CMR College of Engineering & Technology
Kandlakoya (V), Medchal Road, Hyderabad - 501 401. Andhra Pradesh. INDIA
Phone No: 08418 - 200699. Fax No: 08418 - 200240.
E-Mail : principal@cmrcet.org , www.cmrcet.org

COURSE FILE

Dept. of Computer Science and Engineering

Subject	:	BIG DATA ANALYTICS
Academic Year	:	2023-2024
Name of Faculty	:	Dr.S.Kirubakaran/N.Surekha
Department	:	CSE
Branch & Year	:	B.Tech CSE IV Year I SEM

CONTENTS

Course Description

Course Description

Course Objectives:

1. To understand big data including the characteristics of big data such as volume, velocity, variety, veracity, and value.
2. Learn techniques for acquiring, storing, and preprocessing large volumes of data from various sources including structured and unstructured data.
3. Understand different storage and management solutions for big data such as Hadoop Distributed File System (HDFS), NoSQL databases.
4. Gain proficiency in various data analysis techniques including descriptive, diagnostic, predictive, and prescriptive analytics
5. Apply big data analytics techniques to real-world problems and case studies across various domains such as healthcare, finance, marketing, and social media.

Course Outcomes:

The student shall be able:

1. Explain Data Science concepts.
2. Explore data and analyze it using R.
3. Implement classification, clustering and feature selection methods with R.
4. Understand Regression Generalized Linear Models.
5. Perform K-means Analysis using R.

Program Outcomes

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

(UGC AUTONOMOUS)

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD-501 401

ASSESSMENT OF PROGRAMME OUTCOMES & PROGRAMME SPECIFIC OUTCOMES

PROGRAMME				**B.TECH (CSE)**			
YEAR	IV	SEM	VII	Academic Year	2022-23	***BATCH***	***2019-202***
Course Code	A30013			Course Name	BMFA		

ARTICULATION

S.No	COs	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**	**PSO1**
1	**CO1**		2		3									
2	**CO2**					2		3						
3	**CO3**								3		2			
4	**CO4**											3	2	
5	**CO5**						3			2				
Average			2		3	2	3	3	3	2	2	3	2	

FINAL ATTAINMENT (70% of External marks + 30% of Internal marks)				
Description	**CO1**	**C02**	**C03**	**CO4**
External Examinations Attainment	3.00	3.00	3.00	3.00
Internal Examinations Attainment	3.00	3.00	3.00	3.00
70% of External Examinations Attainment	2.10	2.10	2.10	2.10
30% of Internal Examinations	0.90	0.90	0.90	0.90
Final Attainment (70% of Ext + 30% of Int)	3.00	3.00	3.00	3.00
Equivalent attainment %	100.0%	100.0%	100.0%	100.0%
Target attainment %	60.0%	60.0%	60.0%	60.0%
Attainment Status (Y/N)	Y	Y	Y	Y

ATTAINMENT OF POs & PSOs THROUGH THE COURSE OUTCOMES

COs	**Attain ment**	PO1	PO2	PO3	PO4	PO5	PO6	P07	PO8	PO9	PO10	PO11	PO12	PSO1
CO1	**3.00**		2		3									
CO2	**3.00**					2		3						
CO3	**3.00**								3		2			
CO4	**3.00**											3	2	
CO5	**3.00**						3			2				
Attainment		-	**3.00**	-	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**	-

(Course Coordinator)

(Programme Coordinator

Syllabus

(A30540) BIG DATA ANALYTICS

(PROFESSIONAL ELECTIVE-V)

B. Tech (CSE)	**L**	**T**	**P**	**C**
	3	**0**	**0**	**3**

Unit-I

Introduction to Big Data:

Introduction- Big Data, Characteristics & Importance of Big Data – Four V's, Relational Database Vs Big Data, Big Data Analytics, Big Data Applications, Introduction to NoSQL Database Systems

Unit- II

Hadoop:

Introduction to Hadoop, History and future of Hadoop **HDFS-** HDFS Architecture and How to load data into HDFS, Rack Awareness, Data node to name node communication, fault- tolerance feature of HDFS, Read data from HDFS, Block Size concept of HDFS,

Unit – III

Map Reduce:

Introduction to Map Reduce and its Architecture, Hadoop Eco System, Setup Hadoop on a Single node, Simple Map Reduce Program, Executing Map Phase – Shuffling and Sorting, Reducing Phase Execution

Unit- IV

PIG:

Introduction to Apache PIG, Data Model and Schema, Load Store and Relational Operators, Processing Data Using Apache PIG, Parameter Substitution, user defined functions

Unit - V

HIVE:

Introduction to HIVE & its Architecture, HIVE Data Types and Table Creation, loading data in HIVE Tables, Managed Tables and External Tables, Querying HIVE Tables, Introduction to R

Text Books:

1. Big Data, Black Book: Covers Hadoop 2, MapReduce, Hive, YARN, Pig, R and Data Visualization, DT Editorial Services, DreamTech
2. Programming Pig by Alan Gates, O'Reilly; 2nd Revised edition
3. Programming Hive by Edward Capriolo, Dean Wampler, Jason Rutherglen, O'Reilly; First edition

Reference Books:

1. Boris lublinsky, Kevin t. Smith, Alexey Yakubovich, "Professional Hadoop Solutions", Wiley, ISBN: 9788126551071, 2015.
2. Chris Eaton, Drikderoos et al., "Understanding Big Data", McGraw Hill,2012
3. Tom White, "HADOOP: The definitive Guide", O Reilly 2012
4. Vignesh Prajapati, "Big Data Analytics with R and Hadoop", Oracle Press, 2014

Outcomes:

Students shall be able to

1.Describe the Big-Data and Big Data Analytics

2.Illustrate the Hadoop Software Frame work and Its Core components (HDFS and Map-Reduce).

3.Demonstrate loading and reading Data from HDFS and Processing using Map-Reduce.

4.Implement Pig Latin Scripts for processing Data.

5.Use Hive Query language for creating and querying tables

****END****

Academic Calendar

CMR COLLEGE OF ENGINEERING & TECHNOLOGY
(UGC AUTONOMOUS)
Kandlakoya, Medchal Road, Hyderabad – 501401.

Date: 05.06.2023

ACADEMIC CALENDAR
B.Tech IV Year - Academic Year 2023-2024

I Semester

S.No.	Description	Period	Duration
1	Commencement of Class Work	03.07.2023	--------
2	First Spell of Instructions	03.07.2023 to 26.08.2023	8Weeks
3	*First Mid Examinations*	*28.08.2023 to 02.09.2023*	1 Week
4	Submission of Mid-I Marks to Exam Branch	09.09.2023	
5	Parent-Teacher Meeting	16.09.2023	
6	Second Spell of Instructions (Including Dusara Vacation)*	04.09.2023 to 04.11.2023	9 Weeks
7	*Second MidExaminations*	*06.11.2023 to 11.11.2023*	1 Week
8	Submission of Mid-II Marks to Exam Branch	18.11.2023	
9	Preparations and Practical Examinations	13.11.2023 to 18.11.2023	1 Week
10	*End Semester & Supplementary Examinations*	*20.11.2023 to 02.12.2023*	2 Weeks

II Semester

S.No	Description	Period	Duration
1	Commencement of Class Work	04.12.2023	--------
2	First Spell of Instructions	04.12.2023 to 27.01.2024	8 Weeks
3	*First Mid Examinations*	*29.01.2024 to 03.02.2024*	1 Week
4	Submission of Mid-I Marks to Exam Branch	10.02.2024	
5	Second Spell of Instructions	05.02.2024 to 30.03.2024	8 Weeks
6	*Second MidExaminations*	*01.04.2024 to 06.04.2024*	1 Week
7	Submission of Mid-II Marks to Exam Branch	13.04.2024	
8	Preparations and Project Evaluation	08.04.2024 to 13.04.2024	1 Weeks
9	*End Semester & Supplementary Examinations*	*15.04.2024 to 27.04.2024*	2 Weeks

***Subjected to declaration by JNTUH/ Govt. of TS**

05/6/2023

PRINCIPAL

Principal
CMR College of Engineering & Technolog
(UGC Autonomous)
Kandlakoya, Medchal Road,Hyderabad,T

Copy to: Secretary Garu/CEO for kind information please

Copy to : 1. Deans
2. IQAC
3. All HODs
4. Administrative Officer
5. Accounts Officer
6. Web Portal In charge
7. ERP In Charge
8. Library
9. Student Notice Boards.

Lesson Plan

CMR College of Engineering & Technology

Department of Computer Science & Engineering

SESSION PLANNER

Academic Year : 2023-24 **Semester :** I **Regulation :** R-18
Course Code : A30540 **Course** : BIG DATA ANALYTICS **Course Credits:** 3
Faculty Name: Dr.S.Kirubakaran/N.Surekha **Semester Start Date:** 12-06-2023 **Semester End Date:** 04.11.2023

S.No	Subject Topic Name/ Sub Topic Name	Books	No. of Periods	Cumulative No. of Periods	Planned Week/Date	Completed Date	Delivery Method (White Board/ PPT/ Video links/ URLs /Animation/ Quiz/ Case study/ Model Show case/ 3DVisualization/Mentimeter/ Kahoot/Google classroom/ NPTEL Videos/Pod Cast/ Hands-on/Demos ...etc)
				UNIT-I			
1	Introduction- Big Data	T1	1	1	3.07.2023 to 8.07.2023		PPT
2	Characteristics & Importance of Big Data	T1	1	2			PPT, NPTEL
3	Four V's	T1	1	3			PPT
4	Relational Database Vs Big Data	T1	2	5	10.07.2023 to 15.07.2023		PPT, WB, Video links
5	Big Data Analytics	T1	1	6			PPT, WB, Hands-on
6	Big Data Applications	T1	2	8	17.07.2023 to 22.07.2023		PPT, WB
7	Introduction to NoSQL Database Systems	T1	1	9			PPT, WB, Video links
				UNIT-II			
8	Introduction to Hadoop	T1,R3	1	10	24.07.2023 to 29.07.2023		PPT, WB,Video Link
9	History and future of Hadoop	T1	1	11			PPT, WB
10	HDFS- HDFS Architecture	T1,R3	1	12			PPT, WB
11	How to load data into HDFS	T1	1	13	31.07.2023 to 05.08.2023		PPT, WB
12	Rack Awareness	T1	1	14			PPT, WB,Video Link

13	Data node to name node communication	T1	1	15	07.08.2023 to 12.08.2023		PPT, WB, Video links
14	fault- tolerance feature of HDFS	T1	1	16			PPT, WB
15	Read data from HDFS	T1	1	17			PPT, WB
16	Block Size concept of HDFS	T1	1	18			
				UNIT-III			
17	Introduction to Map Reduce	T1	1	19	14.08.02023 to 19.08.2023		PPT, WB,Video Link
18	Map Reduce Architecture	T1	1	20			PPT, WB
19	Hadoop Eco System	T1	1	21			PPT, WB
20	Setup Hadoop on a Single node	T1	2	23	21.08.2023 to 26.08.2023		PPT, WB
21	Simple Map Reduce Program	T1	1	24			PPT, WB
22	Executing Map Phase – Shuffling and Sorting	T1	2	26	04.09.2023 to 09.09.2023		PPT, WB,Video Link
23	Reducing Phase Execution	T1	1	27			PPT, WB,Video Link
				UNIT-IV			
24	Introduction to Apache PIG	T1.T2	1	28	11.09.2023 to 16.09.2023		PPT, WB,Video Link
25	Data Model	T1.T2	1	29			PPT, WB
26	Schema	T1	1	30			PPT, WB
27	Load Store and Relational Operators	T1.T2	2	32	18.09.2023 to 23.09.2023		PPT, WB
28	Processing Data Using Apache PIG	T1	1	33			PPT, WB
29	Apache PIG Parameter Substitution	T1	1	34	25.09.2023 to 30.09.2023		PPT, WB
30	User defined functions	T1	2	36			PPT, WB,Video Link
				UNIT-V			
31	Introduction to HIVE	T1,T3	1	37	02.10.2023 to 07.10.2023		PPT, WB,Video Link
32	HIVE Architecture	T1,T3	2	39			PPT, WB,Video Link
33	HIVE Data Types	T1	1	40	09.10.2023 to		PPT, WB

34	HIVE Table Creation	T1	1	41	14.10.2023		PPT, WB
35	loading data in HIVE Tables	T1	1	42			PPT, WB,Video Link
36	Managed Tables and External Tables	T1	1	43	16.10.2023 to 21.10.2023		PPT, WB
37	Querying HIVE Tables	T1	1	44			PPT, WB
38	Introduction to R	T3,R4	1	45			PPT, WB,Video Link
39	Revision(Unit I to Unit III)		3	48	23.10.2023 to 28.10.2023		PPT, WB,Video Link
40	Revision(Unit I to Unit V)		3	51	30.10.2023 to 04.11.2023		PPT, WB,Video Link

TEXT BOOKS

1. Big Data, Black Book: Covers Hadoop 2, MapReduce, Hive, YARN, Pig, R and Data Visualization, DT Editorial Services, DreamTech

2. Programming Pig by Alan Gates, O'Reilly; 2nd Revised edition

3. Programming Hive by Edward Capriolo, Dean Wampler, Jason Rutherglen, O'Reilly; First edition

REFERENCE BOOK

1. Boris lublinsky, Kevin t. Smith, Alexey Yakubovich, “Professional Hadoop Solutions”, Wiley, ISBN: 9788126551071, 2015.

2. Chris Eaton, Drikderoos et al., “Understanding Big Data”, McGraw Hill,2012

3. Tom White, “HADOOP: The definitive Guide”, O Reilly 2012

4. Vignesh Prajapati, “Big Data Analytics with R and Hadoop”, Oracle Press, 2014

Course Outcomes

Students shall be able

CO1: Describe the Big-Data and Big Data Analytics

CO2: Illustrate the Hadoop Software Frame work and Its Core components (HDFS and Map-Reduce).

CO3: Demonstrate loading and reading Data from HDFS and Processing using Map-Reduce.

CO4: Implement Pig Latin Scripts for processing Data.

CO5: Use Hive Query language for creating and querying tables

Faculty Signature **HoD Signature**

Students list

CMR College of Engineering & Technology

Department of Computer Science & Engineering

CSE IV-1 PROFESSIONAL ELECTIVE-5

S.NO:	ROLL NO:	NAME OF THE STUDENT	SUBJECT OPTED
1	20H51A0501	Adki Ashlesha	Big Data Analytics
2	20H51A0502	A.sharon	Big Data Analytics
3	20H51A0503	A Ashwik Rao	Big Data Analytics
4	20H51A0504	ANNAPU REDDY NITHIN KUMAR REDDY	Big Data Analytics
5	20H51A0505	AravelliAbhinav	Big Data Analytics
6	20H51A0506	B.Mani Chandra	Big Data Analytics
7	20H51A0507	Prajnaya	Big Data Analytics
8	20H51A0508	D Sai Venkata Bhaskara Varma	Big Data Analytics
9	20H51A0509	Dupathi Shravani	Big Data Analytics
10	20H51A0510	G.Praneeth	Big Data Analytics
11	20H51A0511	G. Harshitha	Big Data Analytics
12	20H51A0512	Poojitha	Big Data Analytics
13	20H51A0513	Nithin k	Big Data Analytics
14	20H51A0515	Pavan Reddy	Big Data Analytics
15	20H51A0516	Mamidi Varun	Big Data Analytics
16	20H51A0517	M Guru Sai Chawan	Big Data Analytics
17	20H51A0518	P. Varshitha	Big Data Analytics
18	20H51A0519	Pambala Jagan	Big Data Analytics
19	20H51A0520	Ramyasri	Big Data Analytics
20	20H51A0521	PATHPI SREENIDHI	Big Data Analytics
21	20H51A0523	Tammi sai venkat	Big Data Analytics
22	20H51A0524	V.Sri Vidya	Big Data Analytics
23	20h51a0525	V. Datta Sai	Big Data Analytics
24	20H51A0527	Yoddi Sandeep	Big Data Analytics
25	20H51A0528	A.Bhanu Prasad Reddy	Big Data Analytics
26	20H51A0529	A.Sindhuja	Big Data Analytics
27	20H51A0531	Balaji Bhandare	Big Data Analytics
28	20H51A0532	Banoth Naresh	Big Data Analytics
29	20H51A0534	KALA KUSHAL JAIN	Big Data Analytics
30	20H51A0535	Kalluri Rishita	Big Data Analytics
31	20H51A0537	Kolipelli harshitha	Big Data Analytics
32	20H51A0540	Nakka Sreekar	Big Data Analytics
33	20H51A0541	Neelam Shravani	Big Data Analytics
34	20H51A0542	N KEERTHI	Big Data Analytics
35	20H51A0543	P SATWIK	Big Data Analytics
36	20H51A0544	Piska Vinay	Big Data Analytics
37	20H51A0545	POTHARAM ADHARSH	Big Data Analytics
38	20H51A0546	REDDYCHERLA YESHWANTH RAJU	Big Data Analytics
39	20H51A0547	Rishab Agarwal	Big Data Analytics
40	20H51A0548	RUHEENANAAZ	Big Data Analytics
41	20H51A0549	Sandru Abhinaya Reddy	Big Data Analytics
42	20H51A0550	Sreya Srungarapu	Big Data Analytics
43	20H51A0551	S Deepthi	Big Data Analytics
44	20H51A0552	Tammana Sachit	Big Data Analytics
45	20H51A0553	T NIHITH NOVAH	Big Data Analytics
46	20H51A0554	Vattikuti Vijay	Big Data Analytics
47	20H51A0557	ATTELLI BHAGYA SREE	Big Data Analytics
48	20H51A0558	Bachupally Akhil Goud	Big Data Analytics
49	20H51A0559	B.Sathwik	Big Data Analytics
50	20H51A0563	Deepati Honey Kezia	Big Data Analytics

51	20H51A0564	G NAVEEN	Big Data Analytics
52	20H51A0565	VarShitha	Big Data Analytics
53	20H51A0566	INDRAKANTY SREE ANVITA	Big Data Analytics
54	20H51A0567	Sreehaas Kodityala	Big Data Analytics
55	20H51A0568	Maddiveni Priyanka	Big Data Analytics
56	20H51A0570	NANDIREDDY SRIKANTH REDDY	Big Data Analytics
57	20H51A0571	Pallerla Satwik	Big Data Analytics
58	20H51A0572	Pitla Shirisha	Big Data Analytics
59	20H51A0573	Rajuru Grishma	Big Data Analytics
60	20H51A0574	Saba zareen	Big Data Analytics
61	20H51A0576	S Tharun Kumar	Big Data Analytics
62	20H51A0577	S.CHANDANA	Big Data Analytics
63	20H51A0578	VEMPATI VENKATA SAI CHARAN REDDY	Big Data Analytics
64	20h51a0579	V.bhavana	Big Data Analytics
65	20H51A0580	Vuyyuru Namitha	Big Data Analytics
66	20H51A0581	YADAVALI LAXMI NARAYANA	Big Data Analytics
67	20H51A0582	A HARI PRIYA	Big Data Analytics
68	20H51A0583	BANDA SAI RAMAN	Big Data Analytics
69	20H51A0584	Bodduru Pradeep	Big Data Analytics
70	20H51A0586	C ASHWITH	Big Data Analytics
71	20h51a0588	D sunil kumar	Big Data Analytics
72	20H51A0589	Priyanka Ericherla	Big Data Analytics
73	20H51A0590	NISHANTH REDDY ETIKYALA	Big Data Analytics
74	20H51A0591	Farheen	Big Data Analytics
75	20H51A0592	Gangula Sindhu	Big Data Analytics
76	20H51A0593	G. Rama sai charan	Big Data Analytics
77	20H51A0594	Gummadi Suresh Kumar	Big Data Analytics
78	20H51A0595	Harshitha Majety	Big Data Analytics
79	20H51A0596	K.Sai Puneeth	Big Data Analytics
80	20H51A05A1	MOHAMMED MOQEED	Big Data Analytics
81	20H51A05A2	NETHI PRANAY	Big Data Analytics
82	20H51A05A3	Parupati Tanuja Reddy	Big Data Analytics
83	20H51A05A7	Tungathurthi Himesh Baradwaj	Big Data Analytics
84	20H51A05A8	Y. ROHITH REDDY	Big Data Analytics
85	20H51A05A9	Sai Prashanth	Big Data Analytics
86	20H51A05B0	K.Ankita	Big Data Analytics
87	20H51A05B1	Balla Ganesh	Big Data Analytics
88	20H51A05B2	Bandakadi Sneha	Big Data Analytics
89	20H51A05B3	Bathula Vishwani	Big Data Analytics
90	20H51A05B5	shravya	Big Data Analytics
91	20H51A05B6	B.Pravalika	Big Data Analytics
92	20H51A05C0	ErrojuSidhartha	Big Data Analytics
93	20H51A05C2	G. Soniya	Big Data Analytics
94	20H51A05C4	Chitra Bhanu Reddy Gopu	Big Data Analytics
95	20H51A05C6	K RAJA SIMHA REDDY	Big Data Analytics
96	20H51A05C7	K.SAI HARSHA	Big Data Analytics
97	20H51A05C8	m.Yashwanth kumar	Big Data Analytics
98	20H51A05C9	Mitapally Pooja	Big Data Analytics
99	20H51A05D1	ADITYA	Big Data Analytics
100	20H51A05D3	T MANOHAR	Big Data Analytics
101	20H51A05D4	Durga Bhavani	Big Data Analytics
102	20H51A05D7	B.Ravichandra	Big Data Analytics
103	20H51A05D8	Praveen kumar	Big Data Analytics
104	20H51A05D9	D.V.Bhuvaneswar reddy	Big Data Analytics
105	20H51A05E0	G Rohith Reddy	Big Data Analytics
106	20H51A05E1	Jakkidi Santhosh Reddy	Big Data Analytics
107	20H51A05E2	JANANI CHALAPATI	Big Data Analytics
108	20H51A05E5	L. Shriya	Big Data Analytics
109	20H51A05E6	Lokini Navya	Big Data Analytics
110	20H51A05E7	Medi Abhinaya	Big Data Analytics

111	20H51A05E9	Sathwik	Big Data Analytics
112	20H51A05F0	Vinay Reddy	Big Data Analytics
113	20H51A05F1	PONNALA NITHIN VARMA REDDY	Big Data Analytics
114	20H51A05F2	Pranav Kumar	Big Data Analytics
115	20H51A05F3	ARYA PATEL PURUMALLA	Big Data Analytics
116	20H51A05F4	S.Udaya Sri	Big Data Analytics
117	20H51A05F5	Sanjeet tumkur	Big Data Analytics
118	20H51A05F6	Sumit Chelluru	Big Data Analytics
119	20H51A05F7	Bhagya Laxmi	Big Data Analytics
120	20H51A05F8	T Rohith	Big Data Analytics
121	20H51A05F9	Tulugu Tanujha	Big Data Analytics
122	20H51A05G0	Vennam Eshwar	Big Data Analytics
123	20H51A05G1	Yasmeen	Big Data Analytics
124	20H51A05G2	N.KOUSHIK	Big Data Analytics
125	20H51A05G5	Chekuri Sai Venkat	Big Data Analytics
126	20H51A05G6	Deeksha Behara	Big Data Analytics
127	20H51A05G7	D.Saikiran	Big Data Analytics
128	20H51A05H3	Kanukanti Shiva Abhigna	Big Data Analytics
129	20H51A05H5	kola chandu	Big Data Analytics
130	20H51A05H7	M.DEEPAK REDDY	Big Data Analytics
131	20H51A05J2	P NEETHIKA	Big Data Analytics
132	20H51A05J3	Anusha Pulipati	Big Data Analytics
133	20H51A05J4	Rahul Sai Ranganathan	Big Data Analytics
134	20H51A05J7	SWARNA BHAGYASHREE	Big Data Analytics
135	20H51A05K0	AKSHAY TONDE	Big Data Analytics
136	20H51A05K1	B AKASH GOUD	Big Data Analytics
137	20H51A05K2	Bammidi Sharanya	Big Data Analytics
138	20H51A05K4	Ch.Sridham	Big Data Analytics
139	20H51A05K5	G.Bhagath	Big Data Analytics
140	20H51A05K6	G.sai bhargav	Big Data Analytics
141	20H51A05K7	G.Nishith Reddy	Big Data Analytics
142	20H51A05K8	GYARALA SAI KARTHIK GOUD	Big Data Analytics
143	20H51A05L1	M Srikanth	Big Data Analytics
144	20H51A05L2	M NITHIN	Big Data Analytics
145	20H51A05L3	Narla Krishna Koushik	Big Data Analytics
146	20H51A05L4	PAILLA CHETAN DATTA	Big Data Analytics
147	20H51A05L5	P.Akshitha	Big Data Analytics
148	20H51A05L6	Riyaz ahmed	Big Data Analytics
149	20H51A05L7	Sevva Jashwitha	Big Data Analytics
150	20H51A05L8	Siripuram Nikhitha	Big Data Analytics
151	20H51A05L9	Sudhireddy Manikanta Reddy	Big Data Analytics
152	20H51A05M1	Thavutu Ruchitha	Big Data Analytics
153	20H51A05M3	Venkata Sai Natha Reddy Vaddi	Big Data Analytics
154	20H51A05M4	Vandana	Big Data Analytics
155	20H51A05M7	AKSHITH Akkali	Big Data Analytics
156	20H51A05M9	Sree Harsha	Big Data Analytics
157	20H51A05N3	Chennuri Karthik	Big Data Analytics
158	20H51A05N4	CHETTY SHIVA KUMAR SANKEERTH GOUD	Big Data Analytics
159	20H51A05N6	G NITIN	Big Data Analytics
160	20H51A05N9	Nithya sri	Big Data Analytics
161	20H51A05P1	Ksheersagar Nagendra	Big Data Analytics
162	20H51A05P2	L Jatin	Big Data Analytics
163	20H51A05P3	Mallela Jashwanth	Big Data Analytics
164	20H51A05P4	MANGI LAYA	Big Data Analytics
165	20H51A05P5	Meghana	Big Data Analytics
166	20H51A05P6	M.V.Devendranathreddy	Big Data Analytics
167	20H51A05P8	Rajnish Yadav	Big Data Analytics
168	20H51A05P9	Sanikommu chaitra vyshak Reddy	Big Data Analytics
169	20H51A05Q0	YELPULA ABHYUDAY	Big Data Analytics
170	20H51A05Q3	Twinkle Sharma	Big Data Analytics
171	20H51A05Q4	Vivek Khajuria	Big Data Analytics
172	21H55A0509	K Venakata giridhar	Big Data Analytics

HOD CSE

Internal Marks

CMR College of Engineering & Technology

(UGC AUTONOMOUS)

Kandlakoya , Medchal Road - 501401

Department of Computer Science and Engineering

MID-I MARKS LIST

Class : IV B.Tech. I SEM CSE

A.Y.2023-24

Subject:Big Data Analytics(PE-V)...B-1...

S.No	Roll Number	Name of the Candidate	Assignment (5M)	MID Marks (25 M)	Total (30 M)
1	20H51A0501	Adki Ashlesha	5	25	30
2	20H51A0502	A.sharon	5	24	29
3	20H51A0503	A Ashwik Rao	5	23	28
4	20H51A0504	ANNAPU REDDY NITHIN KUMAR REDDY	0	22	22
5	20H51A0505	AravelliAbhinav	5	24	29
6	20H51A0506	B.Mani Chandra	5	24	29
7	20H51A0507	Prajnaya	5	22	27
8	20H51A0508	D Sai Venkata Bhaskara Varma	5	20	25
9	20H51A0509	Dupathi Shravani	5	25	30
10	20H51A0510	G.Praneeth	5	20	25
11	20H51A0511	G. Harshitha	5	23	28
12	20H51A0512	Poojitha	5	20	25
13	20H51A0513	Nithin k	0	22	22
14	20H51A0515	Pavan Reddy	5	21	26
15	20H51A0516	Mamidi Varun	5	22	27
16	20H51A0517	M Guru Sai Chawan	5	21	26
17	20H51A0518	P. Varshitha	5	23	28
18	20H51A0519	Pambala Jagan	5	22	27
19	20H51A0520	Ramyasri	5	17	22
20	20H51A0521	PATHPI SREENIDHI	5	20	25
21	20H51A0523	Tammi sai venkat	5	23	28
22	20H51A0524	V.Sri Vidya	5	21	26
23	20h51a0525	V. Datta Sai	5	20	25
24	20H51A0527	Yoddi Sandeep	5	18	23
25	20H51A0528	A.Bhanu Prasad Reddy	0	17	17
26	20H51A0529	A.Sindhuja	5	17	22
27	20H51A0531	Balaji Bhandare	5	12	17
28	20H51A0532	Banoth Naresh	5	17	22
29	20H51A0534	KALA KUSHAL JAIN	5	20	25
30	20H51A0535	Kalluri Rishita	5	22	27
31	20H51A0537	Kolipelli harshitha	5	17	22
32	20H51A0540	Nakka Sreekar	5	24	29
33	20H51A0541	Neelam Shravani	5	20	25
34	20H51A0542	N KEERTHI	5	23	28
35	20H51A0543	P SATWIK	5	21	26

S.No	Roll Number	Name of the Candidate	Assignment (5M)	MID Marks (25 M)	Total M)
36	20H51A0544	Piska Vinay	5	19	24
37	20H51A0545	POTHARAM ADHARSH	5	18	23
38	20H51A0546	REDDYCHERLA YESHWANTH RAJU	5	AB	05
39	20H51A0547	Rishab Agarwal	5	15	20
40	20H51A0548	RUHEENANAAZ	5	23	28
41	20H51A0549	Sandru Abhinaya Reddy	5	21	26
42	20H51A0550	Sreya Srungarapu	5	23	28
43	20H51A0551	S Deepthi	5	24	29
44	20H51A0552	Tammana Sachit	0	13	13
45	20H51A0553	T NIHITH NOVAH	5	18	23
46	20H51A0554	Vattikuti Vijay	5	20	25
47	20H51A0557	ATTELLI BHAGYA SREE	5	23	28
48	20H51A0558	Bachupally Akhil Goud	5	19	24
49	20H51A0559	B.Sathwik	5	15	20
50	20H51A0563	Deepati Honey Kezia	5	15	20
51	20H51A0564	G NAVEEN	5	21	26
52	20H51A0565	VarShitha	0	19	19
53	20H51A0566	INDRAKANTY SREE ANVITA	5	23	28
54	20H51A0567	Sreehaas Kodityala	5	17	22
55	20H51A0568	Maddiveni Priyanka	5	22	27
56	20H51A0570	NANDIREDDY SRIKANTH REDDY	5	21	26
57	20H51A0571	Pallerla Satwik	5	07	11
58	20H51A0572	Pitla Shirisha	5	17	22
59	20H51A0573	Rajuru Grishma	5	17	22
60	20H51A0574	Saba zareen	5	~~19~~ 19	24
61	20H51A0576	S Tharun Kumar	5	AB.	05
62	20H51A0577	S.CHANDANA	5	22	27
63	20H51A0578	VEMPATI VENKATA SAI CHARAN REDDY	0	15	15
64	20h51a0579	V.bhavana	5	21	26
65	20H51A0580	Vuyyuru Namitha	5	23	28
66	20H51A0581	YADAVALI LAXMI NARAYANA	5	15	20
67	20H51A0582	A HARI PRIYA	5	21	26
68	20H51A0583	BANDA SAI RAMAN	0	AB	00
69	20H51A0584	Bodduru Pradeep	5	12	17
70	20H51A0586	C ASHWITH	5	16	21

.No	Roll Number	Name of the Candidate	Assignment (5M)	MID Marks (25 M)	Total M)
71	20h51a0588	D sunil kumar	5	18	23
72	20H51A0589	Priyanka Ericherla	5	18	23
73	20H51A0590	NISHANTH REDDY ETIKYALA	5	23	28
74	20H51A0591	Farheen	5	22	27
75	20H51A0592	Gangula Sindhu	5	25	30
76	20H51A0593	G. Rama sai charan	5	19	24
77	20H51A0594	Gummadi Suresh Kumar	5	21	26
78	20H51A0595	Harshitha Majety	5	14	19
79	20H51A0596	K.Sai Puneeth	5	18	23
80	20H51A05A1	MOHAMMED MOQEED	5	20	25
81	20H51A05A2	NETHI PRANAY	5	08	13
82	20H51A05A3	Parupati Tanuja Reddy	5	19	24
83	20H51A05A7	Tungathurthi Himesh Baradwaj	5	19	24
84	20H51A05A8	Y. ROHITH REDDY	0	21	21
85	20H51A05A9	Sai Prashanth	5	22	27

Name&Signature of the Faculty	: Dr.S. KIRUBAKARAN.
Department	: CSE
Mobile No :	9677421281

HOD/CSE

CMR College of Engineering & Technology

(UGC AUTONOMOUS)

Kandlakoya , Medchal Road - 501401

Department of Computer Science and Engineering

MID-I MARKS LIST

Class : IV B.Tech. I SEM CSE A.Y.2023-24

SUBJECT : Big Data Analytics(PE-V)..B-II...

S.No	Roll Number	Name of the Candidate	Assignment (5M)	MID Marks (25 M)	Total (30 M)
1	20H51A05B0	K.Ankita	5	21	26
2	20H51A05B1	Balla Ganesh	5	16	21
3	20H51A05B2	Bandakadi Sneha	5	18	23
4	20H51A05B3	Bathula Vishwani	5	19	24
5	20H51A05B5	shravya	5	22	27
6	20H51A05B6	B.Pravalika	5	23	28
7	20H51A05C0	ErrojuSidhartha	5	16	21
8	20H51A05C2	G. Soniya	5	16	21
9	20H51A05C4	Chitra Bhanu Reddy Gopu	5	20	25
10	20H51A05C6	K RAJA SIMHA REDDY	5	18	23
11	20H51A05C7	K.SAI HARSHA	5	22	27
12	20H51A05C8	m.Yashwanth kumar	5	19	24
13	20H51A05C9	Mitapally Pooja	5	17	22
14	20H51A05D1	ADITYA	5	18	23
15	20H51A05D3	T MANOHAR	5	16	21
16	20H51A05D4	Durga Bhavani	5	22	27
17	20H51A05D7	B.Ravichandra	5	20	25
18	20H51A05D8	Praveen kumar	5	23	28
19	20H51A05D9	D.V.Bhuvaneswar reddy	5	12	17
20	20H51A05E0	G Rohith Reddy	5	25	30
21	20H51A05E1	Jakkidi Santhosh Reddy	5	24	29
22	20H51A05E2	JANANI CHALAPATI	5	23	28
23	20H51A05E5	L. Shriya	5	24	29
24	20H51A05E6	Lokini Navya	5	22	27
25	20H51A05E7	Medi Abhinaya	5	24	29
26	20H51A05E9	Sathwik	5	22	27
27	20H51A05F0	Vinay Reddy	5	18	23
28	20H51A05F1	PONNALA NITHIN VARMA REDDY	5	23	28
29	20H51A05F2	Pranav Kumar	5	24	29
30	20H51A05F3	ARYA PATEL PURUMALLA	5	19	24
31	20H51A05F4	S.Udaya Sri	5	23	28
32	20H51A05F5	Sanjeet tumkur	5	20	25
33	20H51A05F6	Sumit Chelluru	5	21	26
34	20H51A05F7	Bhagya Laxmi	5	24	29
35	20H51A05F8	T Rohith	5	17	22

S.No	Roll Number	Name of the Candidate	Assignment (5M)	MID Marks (25 M)	Total (30 M)
36	20H51A05F9	Tulugu Tanujha	5	21	26
37	20H51A05G0	Vennam Eshwar	5	19	24
38	20H51A05G1	Yasmeen	5	23	28
39	20H51A05G2	N.KOUSHIK	A	20	20
40	20H51A05G5	Chekuri Sai Venkat	A	20	20
41	20H51A05G6	Deeksha Behara	5	23	28
42	20H51A05G7	D.Saikiran	5	19	24
43	20H51A05H3	Kanukanti Shiva Abhigna	5	16	21
44	20H51A05H5	kola chandu	5	18	23
45	20H51A05H7	M.DEEPAK REDDY	5	24	29
46	20H51A05J2	P NEETHIKA	5	9	14
47	20H51A05J3	Anusha Pulipati	5	23	28
48	20H51A05J4	Rahul Sai Ranganathan	5	15	20
49	20H51A05J7	SWARNA BHAGYASHREE	5	25	30
50	20H51A05K0	AKSHAY TONDE	A	21	21
51	20H51A05K1	B AKASH GOUD	5	21	26
52	20H51A05K2	Bammidi Sharanya	5	23	28
53	20H51A05K4	Ch.Sridham	5	18	23
54	20H51A05K5	G.Bhagath	5	14	19
55	20H51A05K6	G.sai bhargav	A	16	16
56	20H51A05K7	G.Nishith Reddy	5	25	30
57	20H51A05K8	GYARALA SAI KARTHIK GOUD	5	21	26
58	20H51A05L1	M Srikanth	5	17	22
59	20H51A05L2	M NITHIN	A	A	A
60	20H51A05L3	Narla Krishna Koushik	5	19	24
61	20H51A05L4	PAILLA CHETAN DATTA	5	22	27
62	20H51A05L5	P.Akshitha	5	18	23
63	20H51A05L6	Riyaz ahmed	A	21	21
64	20H51A05L7	Sevva Jashwitha	5	25	30
65	20H51A05L8	Siripuram Nikhitha	5	17	22
66	20H51A05L9	Sudhireddy Manikanta Reddy	5	21	26
67	20H51A05M1	Thavutu Ruchitha	5	24	29
68	20H51A05M3	Venkata Sai Natha Reddy Vaddi	5	14	19
69	20H51A05M4	Vandana	5	25	30
70	20H51A05M7	AKSHITH Akkali	5	25	30

No	Roll Number	Name of the Candidate	Assignment (5M)	MID Marks (25 M)	Total (30 M)
1	20H51A05M9	Sree Harsha	A	14	14
2	20H51A05N3	Chennuri Karthik	5	24	29
3	20H51A05N4	CHETTY SHIVA KUMAR SANKEERTH GOUD	5	23	28
4	20H51A05N6	G NITIN	5	20	25
5	20H51A05N9	Nithya sri	5	23	28
6	20H51A05P1	Ksheersagar Nagendra	A	22	22
7	20H51A05P2	L Jatin	A	20	20
8	20H51A05P3	Mallela Jashwanth	5	23	28
9	20H51A05P4	MANGI LAYA	5	21	26
0	20H51A05P5	Meghana	5	23	28
31	20H51A05P6	M.V.Devendranathreddy	5	14	19
32	20H51A05P8	Rajnish Yadav	5	20	25
33	20H51A05P9	Sanikommu chaitra vyshak Reddy	5	3	8
84	20H51A05Q0	YELPULA ABHYUDAY	A	A	A
85	20H51A05Q3	Twinkle Sharma	5	16	21
86	20H51A05Q4	Vivek Khajuria	5	14	19
87	21H55A0509	K Venakata giridhar	A	23	23

ame&Signature of the Faculty :	N. Sureblha
epartment :	CSE
lobile No :	9052968 333

HOD/CSE

CMR College of Engineering & Technology

(UGC AUTONOMOUS)

Kandlakoya , Medchal Road - 501401

CMR

Department of Computer Science and Engineering

MID-II MARKS LIST

Class : IV B.Tech. I SEM CSE | A.Y.2023-24

Subject:Big Data Analytics(PE-V) Batch-I...............................

S.No	Roll Number	Name of the Candidate	Assignment (5M)	MID Marks (25 M)	Total (30 M)
1	20H51A0501	Adki Ashlesha	5	25	30
2	20H51A0502	A.sharon	5	24	29
3	20H51A0503	A Ashwik Rao	5	21	26
4	20H51A0504	ANNAPU REDDY NITHIN KUMAR REDDY	0	16	16
5	20H51A0505	AravelliAbhinav	5	20	25
6	20H51A0506	B.Mani Chandra	5	21	26
7	20H51A0507	Prajnaya	5	22	27
8	20H51A0508	D Sai Venkata Bhaskara Varma	5	19	24
9	20H51A0509	Dupathi Shravani	5	25	30
10	20H51A0510	G.Praneeth	5	17	22
11	20H51A0511	G. Harshitha	5	20	25
12	20H51A0512	Poojitha	5	20	25
13	20H51A0513	Nithin k	5	18	23
14	20H51A0515	Pavan Reddy	5	19	24
15	20H51A0516	Mamidi Varun	5	21	26
16	20H51A0517	M Guru Sai Chawan	5	21	26
17	20H51A0518	P. Varshitha	5	23	28
18	20H51A0519	Pambala Jagan	5	16	21
19	20H51A0520	Ramyasri	5	18	23
20	20H51A0521	PATHPI SREENIDHI	5	22	27
21	20H51A0523	Tammi sai venkat	5	25	30
22	20H51A0524	V.Sri Vidya	5	21	26
23	20h51a0525	V. Datta Sai	5	18	23
24	20H51A0527	Yoddi Sandeep	5	23	28
25	20H51A0528	A.Bhanu Prasad Reddy	5	15	20
26	20H51A0529	A.Sindhuja	5	10	15
27	20H51A0531	Balaji Bhandare	5	12	17
28	20H51A0532	Banoth Naresh	5	21	26
29	20H51A0534	KALA KUSHAL JAIN	5	21	26
30	20H51A0535	Kalluri Rishita	0	19	19
31	20H51A0537	Kolipelli harshitha	5	06	11
32	20H51A0540	Nakka Sreekar	5	23	28
33	20H51A0541	Neelam Shravani	5	21	26
34	20H51A0542	N KEERTHI	5	20	25
35	20H51A0543	P SATWIK	5	15	20

S.No	Roll Number	Name of the Candidate	Assignment (5M)	MID Marks (25 M)	Total (30 M)
36	20H51A0544	Piska Vinay	5	23	28
37	20H51A0545	POTHARAM ADHARSH	5	14	19
38	20H51A0546	REDDYCHERLA YESHWANTH RAJU	5	19	24
39	20H51A0547	Rishab Agarwal	5	19	24
40	20H51A0548	RUHEENANAAZ	5	23	28
41	20H51A0549	Sandru Abhinaya Reddy	5	15	20
42	20H51A0550	Sreya Srungarapu	5	23	28
43	20H51A0551	S Deepthi	5	15	20
44	20H51A0552	Tammana Sachit	0	12	12
45	20H51A0553	T NIHITH NOVAH	5	21	26
46	20H51A0554	Vattikuti Vijay	5	22	27
47	20H51A0557	ATTELLI BHAGYA SREE	5	24	29
48	20H51A0558	Bachupally Akhil Goud	5	20	25
49	20H51A0559	B.Sathwik	5	18	23
50	20H51A0563	Deepati Honey Kezia	5	18	23
51	20H51A0564	G NAVEEN	5	19	24
52	20H51A0565	VarShitha	5	16	21
53	20H51A0566	INDRAKANTY SREE ANVITA	5	19	24
54	20H51A0567	Sreehaas Kodityala	5	17	22
55	20H51A0568	Maddiveni Priyanka	5	22	27
56	20H51A0570	NANDIREDDY SRIKANTH REDDY	5	21	26
57	20H51A0571	Pallerla Satwik	0	16	16
58	20H51A0572	Pitla Shirisha	5	21	26
59	20H51A0573	Rajuru Grishma	5	19	24
60	20H51A0574	Saba zareen	5	15	20
61	20H51A0576	S Tharun Kumar	5	20	25
62	20H51A0577	S.CHANDANA	5	21	26
63	20H51A0578	VEMPATI VENKATA SAI CHARAN REDDY	0	10	10
64	20h51a0579	V.bhavana	5	17	22
65	20H51A0580	Vuyyuru Namitha	5	20	25
66	20H51A0581	YADAVALI LAXMI NARAYANA	5	18	23
67	20H51A0582	A HARI PRIYA	5	22	27
68	20H51A0583	BANDA SAI RAMAN	5	14	19
69	20H51A0584	Bodduru Pradeep	5	18	23
70	20H51A0586	C ASHWITH	5	23	28

S.No	Roll Number	Name of the Candidate	Assignment (5M)	MID Marks (25 M)	Total (30 M)
71	20h51a0588	D sunil kumar	5	13	18
72	20H51A0589	Priyanka Ericherla	5	19	24
73	20H51A0590	NISHANTH REDDY ETIKYALA	5	25	30
74	20H51A0591	Farheen	5	21	26
75	20H51A0592	Gangula Sindhu	5	25	30
76	20H51A0593	G. Rama sai charan	5	23	28
77	20H51A0594	Gummadi Suresh Kumar	5	21	26
78	20H51A0595	Harshitha Majety	5	20	25
79	20H51A0596	K.Sai Puneeth	5	18	23
80	20H51A05A1	MOHAMMED MOQEED	5	17	22
81	20H51A05A2	NETHI PRANAY	5	02	07
82	20H51A05A3	Parupati Tanuja Reddy	5	19	24
83	20H51A05A7	Tungathurthi Himesh Baradwaj	5	22	27
84	20H51A05A8	Y. ROHITH REDDY	5	21	26
85	20H51A05A9	Sai Prashanth	5	18	23

Name&Signature of the Faculty :	Dr.S.Kirubakaran
Department :	CSE
Mobile No :	9677421281

HOD/CSE

CMR College of Engineering & Technology

(UGC AUTONOMOUS)

Kandlakoya , Medchal Road - 501401

Department of Computer Science and Engineering

MID-II MARKS LIST

Class : IV B.Tech. I SEM CSE

SUBJECT : Big Data Analytics(PE-V) Batch-II.. A.Y.2023-24

S.No	Roll Number	Name of the Candidate	Assignment (5M)	MID Marks (25 M)	Total (30 M)
1	20H51A05B0	K.Ankita	5	24	29
2	20H51A05B1	Balla Ganesh	5	23	28
3	20H51A05B2	Bandakadi Sneha	5	24	29
4	20H51A05B3	Bathula Vishwani	5	20	25
5	20H51A05B5	shravya	5	25	30
6	20H51A05B6	B.Pravalika	5	22	27
7	20H51A05C0	ErrojuSidhartha	5	23	28
8	20H51A05C2	G. Soniya	5	21	26
9	20H51A05C4	Chitra Bhanu Reddy Gopu	5	25	30
10	20H51A05C6	K RAJA SIMHA REDDY	5	23	28
11	20H51A05C7	K.SAI HARSHA	5	21	26
12	20H51A05C8	m.Yashwanth kumar	5	20	25
13	20H51A05C9	Mitapally Pooja	5	24	29
14	20H51A05D1	ADITYA	5	23	28
15	20H51A05D3	T MANOHAR	5	19	24
16	20H51A05D4	Durga Bhavani	5	22	27
17	20H51A05D7	B.Ravichandra	5	21	26
18	20H51A05D8	Praveen kumar	5	24	29
19	20H51A05D9	D.V.Bhuvaneswar reddy	5	15	20
20	20H51A05E0	G Rohith Reddy	5	24	29
21	20H51A05E1	Jakkidi Santhosh Reddy'	5	23	28
22	20H51A05E2	JANANI CHALAPATI	5	24	29
23	20H51A05E5	L. Shriya	5	22	27
24	20H51A05E6	Lokini Navya	5	20	25
25	20H51A05E7	Medi Abhinaya	5	23	28
26	20H51A05E9	Sathwik	5	22	27
27	20H51A05F0	Vinay Reddy	4	18	22
28	20H51A05F1	PONNALA NITHIN VARMA REDDY	5	24	29
29	20H51A05F2	Pranav Kumar	5	24	29
30	20H51A05F3	ARYA PATEL PURUMALLA	5	21	26
31	20H51A05F4	S.Udaya Sri	5	25	30
32	20H51A05F5	Sanjeet tumkur	5	22	27
33	20H51A05F6	Sumit Chelluru	5	20	25
34	20H51A05F7	Bhagya Laxmi	5	14	19
35	20H51A05F8	T Rohith	4	19	23

S.No	Roll Number	Name of the Candidate	Assignment (5M)	MID Marks (25 M)	Total (30 M)
36	20H51A05F9	Tulugu Tanujha	5	21	26
37	20H51A05G0	Vennam Eshwar	5	23	28
38	20H51A05G1	Yasmeen	5	18	23
39	20H51A05G2	N.KOUSHIK	5	18	23
40	20H51A05G5	Chekuri Sai Venkat	AB	17	17
41	20H51A05G6	Deeksha Behara	5	21	26
42	20H51A05G7	D.Saikiran	5	23	28
43	20H51A05H3	Kanukanti Shiva Abhigna	AB	20	20
44	20H51A05H5	kola chandu	5	21	26
45	20H51A05H7	M.DEEPAK REDDY	5	24	29
46	20H51A05J2	P NEETHIKA	5	17	22
47	20H51A05J3	Anusha Pulipati	5	AB	5
48	20H51A05J4	Rahul Sai Ranganathan	4	22	26
49	20H51A05J7	SWARNA BHAGYASHREE	5	AB	5
50	20H51A05K0	AKSHAY TONDE	5	22	27
51	20H51A05K1	B AKASH GOUD	5	24	29
52	20H51A05K2	Bammidi Sharanya	5	24	29
53	20H51A05K4	Ch.Sridham	5	21	26
54	20H51A05K5	G.Bhagath	5	19	24
55	20H51A05K6	G.sai bhargav	5	20	25
56	20H51A05K7	G.Nishith Reddy	5	23	28
57	20H51A05K8	GYARALA SAI KARTHIK GOUD	5	22	27
58	20H51A05L1	M Srikanth	5	21	26
	20H51A05L2	M NITHIN	5	16	21
60	20H51A05L3	Narla Krishna Koushik	5	20	25
61	20H51A05L4	PAILLA CHETAN DATTA	5	23	28
62	20H51A05L5	P.Akshitha	5	16	21
63	20H51A05L6	Riyaz ahmed	5	17	22
64	20H51A05L7	Sevva Jashwitha	5	24	29
65	20H51A05L8	Siripuram Nikhitha	5	17	22
66	20H51A05L9	Sudhireddy Manikanta Reddy	5	24	29
67	20H51A05M1	Thavutu Ruchitha	5	20	25
68	20H51A05M3	Venkata Sai Natha Reddy Vaddi	5	21	26
69	20H51A05M4	Vandana	5	25	30
70	20H51A05M7	AKSHITH Akkali	5	25	30

S.No	Roll Number	Name of the Candidate	Assignment (5M)	MID Marks (25 M)	Total (30 M)
71	20H51A05M9	Sree Harsha	AB	15	15
72	20H51A05N3	Chennuri Karthik	5	25	30
73	20H51A05N4	CHETTY SHIVA KUMAR SANKEERTH GOUD	5	17	22
74	20H51A05N6	G NITIN	5	20	25
75	20H51A05N9	Nithya sri	5	17	22
76	20H51A05P1	Ksheersagar Nagendra	5	22	27
77	20H51A05P2	L Jatin	5	20	25
78	20H51A05P3	Mallela Jashwanth	5	23	28
79	20H51A05P4	MANGI LAYA	5	24	29
80	20H51A05P5	Meghana	5	23	28
81	20H51A05P6	M.V.Devendranathreddy	5	20	25
82	20H51A05P8	Rajnish Yadav	5	24	29
83	20H51A05P9	Sanikommu chaitra vyshak Reddy	AB	11	11
84	20H51A05Q0	YELPULA ABHYUDAY	AB	AB	
85	20H51A05Q3	Twinkle Sharma	5	22	27
86	20H51A05Q4	Vivek Khajuria	5	19	24
87	21H55A0509	K Venakata giridhar	5	18	23

Name&Signature of the Faculty : N·Suneetha
Department : CSE
Mobile No : 9052968333

HOD/CSE

End semester Results

CMR College of Engineering & Technology

* UGC AUTONOMOUS * Approved by AICTE * Accredited by NAAC with 'A' Grade * All B.Tech programs Accredited by NBA *

B.TECH-IV/IV I SEM Regular Results Analysis Held in November 2023: Final Result Curriculum: R18 Rev2

Branch **ELECTRONICS & COMMUNICATION ENGINEERING**

Subject	Reg	App	Passed	Failed	Reg Pass %	Apprd. Pass %
ENTREPRENEURSHIP (C30164)	7	7	6	1	85.71	85.71
AIR POLLUTION & CONTROL (A30163)	74	74	72	2	97.30	97.30
ENVIRONMENTAL PROTECTION AND MANAGEMENT (A30166)	39	39	39	0	100.00	100.00
WASTE TO ENERGY (A30378)	6	6	6	0	100.00	100.00
CLOUD COMPUTING (A30542)	17	17	15	2	88.24	88.24
INTRODUCTION TO DATA SCIENCE (A30559)	10	10	9	1	90.00	90.00
BASICS OF INSURANCE AND TAXATION (C30165)	15	15	13	2	86.67	86.67
MARKETING MANAGEMNET (C30167)	41	41	40	1	97.56	97.56
MAJOR PROJECT PHASE-I (A30428)	249	249	225	24	90.36	90.36
MINI PROJECT-II (A30426)	116	116	116	0	100.00	100.00
SUMMER INTERNSHIP-II (A30427)	133	133	133	0	100.00	100.00
ALL SUBJECTS	249	248	212	36	85.14	85.48

Branch **COMPUTER SCIENCE & ENGINEERING**

Subject	Reg	App	Passed	Failed	Reg Pass %	Apprd. Pass %
BUSINESS MANAGEMENT & FINANCIAL ANALYSIS (A30013)	260	258	247	11	95.00	95.74
DESIGN PATTERNS (A30534)	226	225	214	11	94.69	95.11
MACHINE LEARNING (A30535)	34	34	31	3	91.18	91.18
DATA ANALYTICS WITH R (A30537)	59	59	55	4	93.22	93.22
DEEP LEARNING (A30538)	66	66	61	5	92.42	92.42
ETHICAL HACKING (A30539)	135	134	124	10	91.85	92.54
BIG DATA ANALYTICS (A30540)	172	171	167	4	97.09	97.66
CLOUD COMPUTING (A30542)	88	86	76	10	86.36	88.37
KNOWLEDGE MANAGEMENT (C30162)	50	49	45	4	90.00	91.84
PYTHON PROGRAMMING (A30531)	48	48	45	3	93.75	93.75

Controller of Examinations

Principal

B.TECH-IV/IV I SEM Regular Results Analysis Held in November 2023: Final Result Curriculum: R18 Rev2

Branch **COMPUTER SCIENCE & ENGINEERING**

Subject	Reg	App	Passed	Failed	Reg Pass %	Apprd. Pass %
DISASTER MANAGEMENT AND MITIGATION (A30160)	54	54	53	1	98.15	98.15
ENTREPRENEURSHIP (C30164)	10	10	10	0	100.00	100.00
AIR POLLUTION CONTROL (A30163)	65	65	62	3	95.38	95.38
MARKETING MANAGEMENT (C30167)	23	23	23	0	100.00	100.00
BASICS OF INSURANCE & TAXATION (C30165)	20	19	19	0	95.00	100.00
ENVIRONMENTAL PROTECTION AND MANAGEMENT 30166)	32	32	29	3	90.63	90.63
MAJOR PROJECT PHASE-I (A30552)	260	256	226	30	86.92	88.28
MINI PROJECT-II (A30549)	260	260	257	3	98.85	98.85
ALL SUBJECTS	260	253	215	38	82.69	84.98

Branch **INFORMATION TECHNOLOGY**

Subject	Reg	App	Passed	Failed	Reg Pass %	Apprd. Pass %
BUSINESS MANAGEMENT & FINANCIAL ANALYSIS (A30013)	65	65	60	5	92.31	92.31
DESIGN PATTERNS (A30534)	65	64	58	6	89.23	90.63
HUMAN COMPUTER INTERACTION (A31206)	65	64	61	3	93.85	95.31
BIG DATA ANALYTICS (A30540)	65	65	54	11	83.08	83.08
ASTER MANAGEMENT AND MITIGATION 30160)	30	30	28	2	93.33	93.33
ENTREPRENEURSHIP (C30164)	5	5	5	0	100.00	100.00
KNOWLEDGE MANAGEMENT (C30162)	2	2	2	0	100.00	100.00
PYTHON PROGRAMMING (A30531)	9	9	8	1	88.89	88.89
AIR POLLUTION AND CONTROL (A30163)	32	32	30	2	93.75	93.75
ENVIRONMENTAL PROTECTION MANAGEMENT (A30166)	1	1	1	0	100.00	100.00
WASTE TO ENERGY (A30378)	1	1	1	0	100.00	100.00
INTRODUCTION TO DATA SCIENCE (A30559)	1	1	1	0	100.00	100.00
BASICS OF INSURANCE AND TAXATION (C30165)	6	6	5	1	83.33	83.33

Controller of Examinations

Principal

Internal exam question paper and solutions with scheme

Investigate about Apache PIG parameters Substitution.Solutions

1. The shuffling or shifting phase involves moving data between nodes to ensure that data with the same key (for example, in a MapReduce operation) is grouped together for processing.

   Sorting can help in various scenarios such as preparing data for further analysis, identifying patterns, or optimizing data retrieval.

2. The schema defines the layout of the data, including the data types, relationships between different entities, and constraints on the data

3. It is a collection of tuples where each tuple consists of fields

4. TINYINT,SMALLINT,INT, INTEGER,BIGINT,FLOAT,DOUBLE,BOOLEAN,STRING
5. Dplyr, ggplot2, tidyr, readr, magrittr, lubridate

A Prerequisites:

Make sure you have Java installed on your machine. Hadoop requires Java to run. You can check if Java is installed by running java -version in your terminal.

Install Hadoop:

Extract the downloaded Hadoop archive to a directory of your choice. This will be your Hadoop installation directory.

Set up environment variables:

Add Hadoop's bin directory to your PATH environment variable.

Set the HADOOP_HOME environment variable to point to your Hadoop installation directory.

Configure Hadoop:

Navigate to the Hadoop configuration directory ($HADOOP_HOME/etc/hadoop) and edit the configuration files:

core-site.xml: Configure Hadoop core settings such as the default file system and Hadoop temp directories.

hdfs-site.xml: Configure HDFS settings such as the replication factor.

mapred-site.xml: Configure MapReduce settings.

yarn-site.xml: Configure YARN settings.

Format the NameNode:

Before starting Hadoop, you need to format the Hadoop Distributed File System (HDFS) NameNode. Run the following command:

luaCopy code

```
hdfs namenode -format
```

Start Hadoop:

Start the Hadoop daemons by running the following command:

cssCopy code

```
start-all.sh
```

This command starts the NameNode, DataNode, ResourceManager, NodeManager, and other required daemons.

Access Hadoop UI:

Once Hadoop is running, you can access the Hadoop web interfaces to monitor the cluster:

HDFS: http://localhost:50070/

YARN ResourceManager: http://localhost:8088/

Run Hadoop Jobs:

You can now run MapReduce or other Hadoop jobs on your single-node Hadoop setup. Upload data to HDFS using hdfs dfs -put and run MapReduce jobs using hadoop jar command.

Stop Hadoop:

After you're done experimenting, you can stop Hadoop daemons by running:

arduinoCopy code.

6. B **Shuffling:**

Shuffling refers to the process of moving intermediate key-value pairs from the mappers to the reducers based on their keys.

After the map tasks have completed their processing, the output from each mapper is partitioned into smaller chunks based on the keys.

These key-value pairs with the same key are then grouped together and sent to the appropriate reducer based on the key's hash value or partitioning function.

Shuffling ensures that all key-value pairs with the same key end up at the same reducer, allowing the reducer to aggregate or process them together.

Sorting:

Sorting occurs after the shuffling phase and involves sorting the key-value pairs within each partition or group.

Each reducer receives a sorted list of key-value pairs grouped by key.

Sorting is essential because it allows reducers to process data efficiently. For example, reducers can perform tasks like aggregation, joining, or calculating statistics more effectively when the data is sorted.

The sorting can be done either on the mapper side before shuffling (map-side sorting) or on the reducer side after shuffling (reduce-side sorting), depending on the MapReduce implementation and optimization techniques used.

7. A Schema-On-Read Model:

In the schema-on-read model, data is ingested into the system without a predefined schema. This is common in scenarios where data is semi-structured or unstructured.

Pig can work with data in this model by interpreting the data structure dynamically at the time of processing. It does not enforce a strict schema on the data upfront.

For example, Pig can process data stored in formats like JSON, CSV, or log files, where the schema may vary between different records or files.

Schema-On-Write Model:

In the schema-on-write model, data is ingested into the system with a predefined schema. This is common in structured data scenarios, such as relational databases.

Pig can also work with data in this model by defining a schema upfront and validating data against that schema during loading.

For example, Pig can interact with data stored in structured formats like Avro, Parquet, or ORC, where the schema is defined and enforced during data ingestion.

7. B Defining Parameters: Parameters can be defined in Pig scripts using the DEFINE statement or by passing them directly through the command line.

Example:

pigCopy code

```
-- Define parameters in the script DEFINE input_path '/path/to/input'; DEFINE
output_path '/path/to/output'; -- Or pass parameters from command line -- pig -param
input_path='/path/to/input' -param output_path='/path/to/output' script.pig
```

Accessing Parameters: Parameters can be accessed within Pig scripts using the $ symbol followed by the parameter name.

Example:

pigCopy code

```
-- Access parameters in the script input_data = LOAD '$input_path' USING
PigStorage(',');
```

Command Line Parameter Passing: When invoking Pig scripts from the command line, parameters can be passed using the -param option followed by the parameter name and value.

Example:

cssCopy code

```
pig -param input_path='/path/to/input' -param output_path='/path/to/output' script.pig
```

Default Values: Parameters can have default values specified in case they are not overridden during script invocation.

Example:

pigCopy code

```
-- Define parameters with default values DEFINE input_path '/path/to/default_input';
DEFINE output_path '/path/to/default_output';
```

Use Case: Parameter substitution is useful when you want to run the same Pig script with different input/output paths, configurations, or any other parameters without modifying the script itself. It enhances script portability and makes it easier to manage different environments.

Complex Parameter Substitution: It's also possible to perform more complex parameter substitution using Pig macros or UDFs if needed, allowing for dynamic generation of Pig script components based on parameter values.

8. A The Hive architecture provides a powerful and flexible platform for querying and analyzing large datasets in Hadoop, with support for SQL-like queries, multiple execution engines, and seamless integration with other Hadoop ecosystem components.

8.B

```
CREATE TABLE IF NOT EXISTS student(
Student_Name STRING,
Student_Rollno INT,
Student_Marks FLOAT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';
```

INSERT Query:

```
INSERT INTO TABLE student VALUES ('Dikshant',1,'95'),('Akshat', 2 ,
'96'),('Dhruv',3,'90');
```

1. Big data refers to large volumes of structured, semi-structured, and unstructured data that inundates a business on a day-to-day basis.

2. Volume, velocity, variety, veracity

3. Job tracker: In the classic MapReduce architecture, the JobTracker was responsible for managing and coordinating MapReduce jobs submitted to the Hadoop cluster.

   Task tracker: TaskTrackers were responsible for executing individual Map and Reduce tasks assigned to them by the JobTracker.

4. continue operating seamlessly in the presence of hardware failures, software errors, or other types of disruptions.

5. **Map Function**:
   The map function is the first phase of a MapReduce job. It takes input data and processes it to produce a set of intermediate key-value pairs.

   Reduce Function:

   The reduce function is the second phase of a MapReduce job, following the map phase. It takes the intermediate key-value pairs produced by the map function and performs aggregation or summarization based on keys

6. A **RDMS (Relational Database Management System):** RDBMS is an information management system, which is based on a data model.In RDBMS tables are used for information storage. Each row of the table represents a record and column represents an attribute of data. Organization of data and their manipulation processes are different in RDBMS from other databases. RDBMS ensures ACID (atomicity, consistency, integrity, durability) properties required for designing a database. The purpose of RDBMS is to store, manage, and retrieve data as quickly and reliably as possible. **Hadoop:** It is an open-source software framework used for storing data and running applications on a group of commodity hardware. It has large storage capacity and high processing power. It can manage multiple concurrent processes at the same time. It is used in predictive analysis, data mining and machine learning. It can handle both structured and unstructured form of data. It is more flexible in storing, processing, and managing data than traditional RDBMS. Unlike traditional systems, Hadoop enables multiple analytical processes on the same data at the same time. It

supports scalability very flexibly. Below is a table of differences between RDBMS and Hadoop.

7. **Healthcare Analytics:**

Big data analytics in healthcare involves analyzing large volumes of patient data, medical records, genomic data, and sensor data to improve patient outcomes, personalize treatments, and optimize healthcare delivery.

Applications include disease prediction and prevention, clinical decision support systems, patient monitoring, and drug discovery.

Financial Services:

In the financial services industry, big data analytics is used for fraud detection, risk management, algorithmic trading, customer segmentation, and personalized financial services.

Applications include real-time fraud detection, credit scoring, portfolio optimization, anti-money laundering (AML) compliance, and customer churn prediction.

Retail and E-commerce:

Big data analytics in retail and e-commerce involves analyzing customer transaction data, browsing behavior, social media data, and inventory data to personalize marketing campaigns, optimize pricing, and improve supply chain management.Etc

8. Infrastructure and Scalability:

Building and maintaining the infrastructure required to handle big data processing and storage at scale is a significant challenge. Organizations need to invest in distributed computing systems, storage solutions, and data management tools capable of scaling horizontally to accommodate growing data volumes and processing requirements.

Data Governance and Compliance:

Establishing data governance policies and practices is essential for ensuring data quality, integrity, and compliance with regulatory requirements. Effective data governance frameworks encompass data stewardship, metadata management, data lineage, and audit trails to maintain control and accountability over data assets.

Data Integration and Interoperability:

Integrating and harmonizing data from disparate sources is a complex task. Organizations often struggle with data silos, incompatible data formats, and inconsistent data schemas, hindering seamless data integration and interoperability across systems and applications.

Cultural and Organizational Challenges:

Embracing a data-driven culture and fostering organizational change are critical for leveraging big data effectively. Organizations must overcome resistance to change, foster collaboration between IT and business units, and promote data literacy and analytics skills among employees to drive innovation and decision-making based on data-driven insights.

9. **Infrastructure and Scalability**:

Building and maintaining the infrastructure required to handle big data processing and storage at scale is a significant challenge. Organizations need to invest in distributed computing systems, storage solutions, and data management tools capable of scaling horizontally to accommodate growing data volumes and processing requirements.

Data Governance and Compliance:

Establishing data governance policies and practices is essential for ensuring data quality, integrity, and compliance with regulatory requirements. Effective data governance frameworks encompass data stewardship, metadata management, data lineage, and audit trails to maintain control and accountability over data assets.

Data Integration and Interoperability:

Integrating and harmonizing data from disparate sources is a complex task. Organizations often struggle with data silos, incompatible data formats, and inconsistent data schemas, hindering seamless data integration and interoperability across systems and applications.

Cultural and Organizational Challenges:

Embracing a data-driven culture and fostering organizational change are critical for leveraging big data effectively. Organizations must overcome resistance to change, foster collaboration between IT and business units, and promote data literacy and analytics skills among employees to drive innovation and decision-making based on data-driven insights.

10. **Hadoop Distributed File System (HDFS):**

HDFS is a distributed file system designed to store large volumes of data across multiple commodity servers. It provides high-throughput access to data and ensures fault tolerance through data replication.

MapReduce:

MapReduce is a programming model and processing engine for distributed data processing in Hadoop. It allows users to write parallelizable computations using simple map and reduce functions, which are executed across clusters of machines.

YARN (Yet Another Resource Negotiator):

YARN is a resource management and job scheduling framework introduced in Hadoop 2.0. It separates the resource management and job scheduling functions from the MapReduce framework, allowing Hadoop to support multiple processing models, including MapReduce, Apache Spark, Apache Flink, and others.

Apache Hive:

Hive is a data warehouse infrastructure built on top of Hadoop. It provides a SQL-like interface (HiveQL) for querying and analyzing large datasets stored in HDFS. Hive translates queries into MapReduce or Tez jobs for execution.

Apache Pig:

Pig is a high-level scripting language designed for processing and analyzing large datasets in Hadoop. It provides a simple and expressive language called Pig Latin, which abstracts complex MapReduce operations into a series of data transformations.

Apache HBase:

HBase is a scalable, distributed database that runs on top of Hadoop. It is a NoSQL database designed for storing and managing large volumes of structured data. HBase provides real-time read and write access to data and is well-suited for applications requiring low-latency access to massive datasets.

Apache Spark:

Spark is a fast and general-purpose cluster computing framework for big data processing. It provides high-level APIs in Java, Scala, Python, and R, along with an optimized engine that supports in-memory processing and iterative algorithms. Spark can run standalone or on YARN, and it offers libraries for SQL, streaming, machine learning, and graph processing.

Apache Kafka:

Kafka is a distributed streaming platform designed for building real-time data pipelines and event-driven applications. It provides scalable, fault-tolerant messaging capabilities for handling high-throughput data streams. Kafka integrates seamlessly with Hadoop and other components of the Hadoop ecosystem.

Apache Sqoop:

Sqoop is a tool designed for efficiently transferring bulk data between Hadoop and structured data stores such as relational databases (e.g., MySQL, Oracle). It supports parallel data transfer and can import data from databases into Hadoop (and vice versa) using MapReduce or direct database connections.

Apache Flume:

Flume is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of log data from various sources to centralized data stores like HDFS.

Apache Oozie:

Oozie is a workflow scheduler system for managing Apache Hadoop jobs. It allows users to define and execute workflows composed of Hadoop jobs, Pig scripts, Hive queries, and other types of tasks in a controlled, repeatable manner.

Hall Ticket No. □□□□□□□□□□ Question Paper Code: A30540

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

(AUTONOMOUS)

B.Tech VII Semester Mid-I Examinations August -2023

(Regulation: CMRCET-R18)

Subject Name: BIG DATA ANALYTICS Time: 10:00AM to 11:40AM

Date:02.09.2023 FN Branch: CSE Max Marks: 25

PART A

Answer all FIVE questions (Compulsory)
Each question carries TWO marks. 5x2=10M

1 State Big Data and its importance

2 List the Four V's and Explain about Veracity.

3 Differentiate the Job Tracker and Task Tracker in HDFS

4 Interpret the Fault Tolerance feature of HDFS

5 Critique the Map Function and Reduce Function

PART B

Answer ALL questions.
Each question carries FIVE Marks. 3x5=15M

6. A. Differentiate Relational Data Base and Big Data in detail

OR

6. B. Appraise in detail about Big Data Application

7. A. Justify the statement "Hadoop gives solution to Big Data Issues".Sketch the HDFS architecture and Explain in detail.

OR

7. B. Investigate in detail about the Name Node and Data Node Communication

8. A. Examine the Map Reduce and its Architecture in detail

OR

8. B. Sketch the Hadoop Eco System and Explain in detail.

****

Hall Ticket | | | | | | | | | | | Question Paper Code:A30540

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

(AUTONOMOUS)

B.Tech VII Semester Mid-II Examinations November -2023

(Regulation: CMRCET-R18)

Subject Name: BIG DATA ANALYTICS **Time: 2.00 PM to 3.40 PM**

Date:13-11-2023 **Branch: CSE** **Max Marks: 25**

PART A

Answer all FIVE questions (Compulsory)

Each question carries TWO marks. **5x2=10M**

1. Describe the shuffling and sorting phases
2. What is schema?
3. Define Data models in PIG
4. Explain about data types in Hive
5. List few header files in R and their usage

PART B

Answer ALL questions.

Each question carries FIVE Marks. **3x5=15M**

6.A. Appraise in detail about Hadoop setup on single node.

OR

6.B. Distinguish shuffling, shorting in reducing phase of MapReduce and Explain in detail.

7.A. State Apache PIG and interpret are different data models in it.

OR

7.B. Investigate about Apache PIG parameters Substitution.

8.A. Sketch Hive Architecture and Explain in detail.

OR

8.B. Demonstrate Table Creation and data Loading in Hive

CO attainment sheet

Sample answer booklets

BOOKLET NUMBER :	College Stamp

R18

CMR
GROUP OF INSTITUTIONS
EXPLORE TO INVENT

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

(AUTONOMOUS)

Kandlakoya, Medchal, Hyderabad - 501 401.

312

MID SEMESTER EXAMINATION ANSWER BOOK

Registered No. | 2 | 0 | H | 5 | 1 | A | 0 | 5 | 0 | 9 |

FIRST / SECOND SEMESTER EXAMINATION B.Tech./M.Tech./MBA VII Semester 2/9/2023
(Month and year)

Subject : BDA

Date : 2/9/2023

N. S 02/09/23
Signature of the Invigilator with date

INSTRUCTIONS TO THE CANDIDATES

1. This booklet contains 16 pages. Candidates must ensure it before writing and in case a defective answer book is issued it must be returned to the invigilator and a new and defect free booklet must be obtained.
2. Before the candidate begins to answer, registered number, particulars of year, semester, subject etc., are to be filled in. Failure to enter all or any of these particulars may disqualify the paper from valuation.
3. Candidate is prohibited from
   (a) Writing.
   ☞ anything addressing the examiner in any manner whatsoever, in their answer book.
   ☞ Objectionable/obscene language in the answer book.
   ☞ anything other than their Registered Number on the question paper.
   (b) either seeking or providing any assistance to the fellow candidates in the exam.
   (c) possessing a manuscript or a printed matter, in any form, in the examination hall.
   (d) bringing loose sheets or paper into the examination hall and detaching any paper from the answer book.
   (e) carrying Mobile Phone to Exam Hall.
   Violation of these instructions will be viewed as a case of malpractice, which is a punishable offence.
4. Before beginning to answer any question, candidates must write the correct question number, in the margin only and should not write anything else in the margin.
5. Answers must be written legibly on both sides of the paper. There shall be about 25 lines in each page. It is not necessary to begin each answer on a fresh page. Candidates should not use any other ink, except BLACK or BLUE ink.
6. Rough work, if any, must be separated, from the subject matter, by a line and noted as rough work.
7. The answer book, at the end of the examination, must be handed over to the Assistant Superintendent (Invigilator) by the candidate **This responsibility lies with the candidate only.**
8. Candidates should maintain absolute silence during the time of examination. Misbehavior, in any form, by the candidate, in the examination hall, will attract severe punishment.
9. Candidates are permitted to leave the examination hall only after the expiry of half of the allotted time and candidates will be permitted to carry the question paper only when they are leaving the exam hall in the last half-an-hour.
10. **No additional answer books will be supplied.**

To be filled in by the Examiner only

PART - A / PART - B
MARKS SLIP

PART-A	Q.No.	1	2	3	4	5	—	—	—	—	—	**Part-A Total**
	Marks	2	2	2	2	2						10
PART-B	Q.No.	6	7	8	—	—	—	—	—	—	—	**Part-B Total**
		A | B	A | B	A | B								
	Marks	5 |	5 |	5 |								15
Grand Total in Words :											**GRAND TOTAL**	25

Signature of the Scrutinizer with Date

Signature of the Examiner with Date

:part-A:

1)→ Big data:

→ Big data it is a process Examining & Extracting the data, pattern, Insights, diagram etc. From the large data set is called as Big data. where we use the tranditional data set for managing the data. For the Managing the large data set we are using the Analytical data Base and will make the decision-making & keep the data efficiency. It is used in various Applications like Health care, Education etc.

The Importance of Big data they are:

1) Improved in decision-making
2) Enchaned the customer needs.
3) Innovation of new oppoumities
4) Health care, Education
5) Tramsport
6). goverment

It is about Big data & its Importance.

2) → The Four Vs are:

1) Volume
2) velocity
3) Variety
4) Veracity.

→ Veracity:

→ The Veracity it is a type of Feature which is used in the Big data. the main work of veracity is to handle the data where in the data will get the Variation, Errors, to controll all those we are using veracity so, It is the use Veracity in the Big data.

3) The difference b/w Job Tracker & Task Tracker in HDFS are,

Job Tracker	Task Tracker.
→ The Job Tracker where it is a Task, The main aim of Job Tracker is Trace the data & assign the Job to the particular data & works to be done.	→ The Task tracker is used to Track the Job Tracker & assign the particular task to the each of the data & have to be done. It is about Task Tracker.

→ In HDFs it is observed in Master.	→ In HDFs is occured in Salve.
→ We use the NameNode in the HDFs, For location of the Job.	→ we use the dataNode For the allocation. of the Job.
→ The Scheduling of the Job will be done.	→ It will track the Job according to it will act to the status.

These are the difference b/w the Job Tracker & Task tracker in HDFS.

4). Fault Tolerance:

→ Fault Tolerance where it is a Feature of the HDFs The main aim of Fault Tolerance is to Identify the Errors in the System & according that the Fault Tolerance will be gets.

→ The Fault Tolerance is used in the Rack awarnes where The data is Transformed From one System to another System By using repicla Function we are doing the system & rectify the Error.

→ The Replication Function is used For Identify the Error, & give the data in the Correct form. So, It is about the Fault Tolerance In HDFs.

→ The work of the Fault Tolerance in HDFs.

57. → The MapFunction:

→ The map Function is used For Splitting the data & analysis of data, them By using mapFunction we are mapping the Input data. the main aim of the mapfunction is used Split the Input data & analysis the data carefully in the. HDFs system. So, It is about the map Function of Hadoop.

→ Reduction Function:

→ The Reduction Function is used For the shuffling the data & andysis the Input data & them we are making the Input data into the single data. So, the Reduction Function is used.

→ So, It is about map Function & Reduction Function.

: part-B:

6) The difference b/w the Relational database &
A) Big data they are:

RDBMS (Relatiom data Base System)	Big data
1) Traditional System is declared inside the entpprise level	1) Bigdata System is declared outride the enterprice level.
2) It will range from gigabyte to Terabyte.	2) It will range from petabyte to zettabyte.
3) The time is required for seconds, minutes.	3) The time is taken for the longtime for processing.
4) Data Integration is Easy.	4) Data Integartion is Hard.
5) Normal tools are used In Relational dataBase.	5) Special tools are used In Bigdata.

6> The Relational data Base is stable & relationship.	6> The Bigdata is unstable & unrelationship.
7> The Normal system is used For configure the data.	7> The High system is used For Configure the data.
8> The Normal tools are required For managing the data	8> The Special kind of tools are required For managing the data.
9> It the data is Easy to manage.	9> The data is Hard to manage.
10> Small amount of data is stored.	10> large amount of data is stored.
11> The manageable Volume will be present.	11> The unmanagable Volume in the Bigdata.
12). Tranditimal system is used for the managing.	12> The analytical system is used For managing large data.

13> Centralizied data is used in the Relational dataBase.	13> The disturbed database is used in the Big data.
Eg1 14>. ERP system, Finance, Image processing.	14> Eg1 Image, Vedio, Health care, education, Transport, NLp etc..

7>

A>. The Statement "Hadoop gives solution to Big data" where the above statement states that the, Hadoop system plays an Important role in the Big data for managing the System issues which are occured in the Bigdata.

→ Hadoop is the File system used For the managing the Examing & Extracting the data From the large system For to reach the target code.

→ where it is an open-source system. where it's a programming model where it is written in the Java.

→ where the Hadoop system will be developed with the low-cost amount & the High throughput will be there For the System.

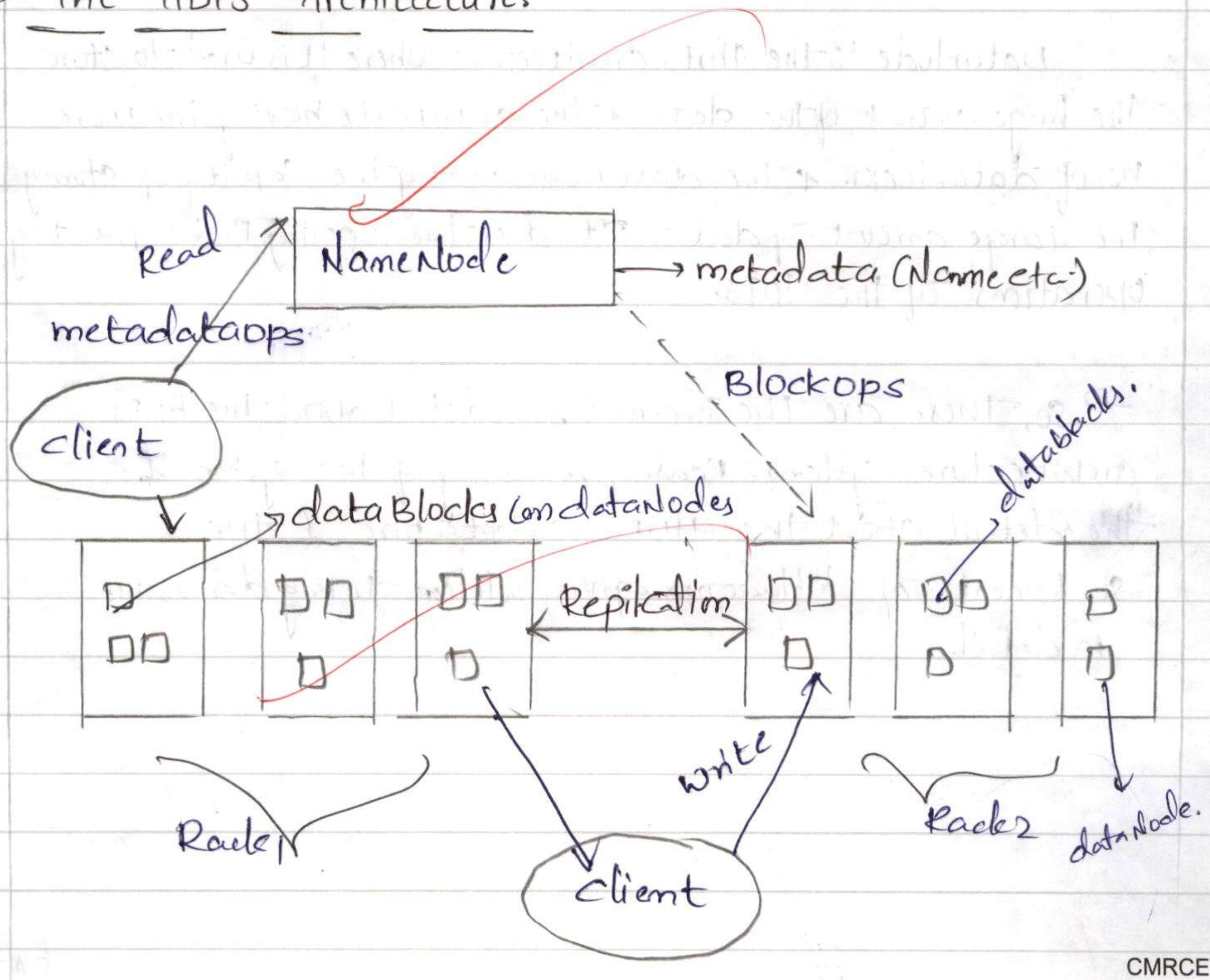
→ There are two main componets Of Hadoop they are:

1> MapReduce

2> HDFS.

→ plays an Important key role In the Bigdata Anaylistics. So, The statement states about the "Hadoop give solution to Big data Issues".

→ The HDFS Architecture:



→ NameNode:

→ NameNode plays an important role in the HDFS. where the NameNode the single type of data will be stored in it. The master where the NameNode is used to the function the Filesystem like opening, the file, Redaling, closing the file. So, the NameNode does, all these operations of the HDFS System of the Architecture.

→ DataNode:

→ DataNode is the HDFS architecture. where it is used to store the huge amount of the data. & the dataNode having the more no. of datablocks. & the dataNode having the Capcity of storing the large amount of data. It does the, opening, Reading, writing operations of the HDFS.

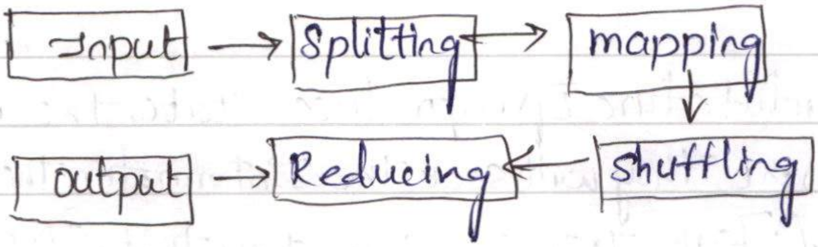
→ So, these are the Functions & in detail about the HDFS architecture system. Reading, & writing of the system. & the detail about the HDFS architecture & the statement of "Hadoop gives Solution to Big data issues" in detail.

87

A). The MapReduce:

→ The MapReduce it is a Important component of Hadoop system. where the Mapreduce plays an important role in the Bigdata. where It is a programming model. It is used For Examining & Extracting the data to get / reach the target code. where they are two Features (or) tasks to be performed they are 1) map:- It is used For the map/splitting the Input value gives the output in the manner of key-value pair. where the 2) Reduce: It is used for analysis the output of the map & covers the key-value pair by the key & gives the output So, It is the mapreduce Function. So, let us discuss some steps of mapreduce Function they are:

→ steps of mapReduce:



→ Splitting:

→ Splitting is used for Split the Input data into the different classes according to there belonging Categories & doing the splitting of Input data.

→ mapping:

→ After the splitting of Input data according to the belonging Fields the System is used For mapping the Input data & arranging.

→ shuffling:

→ shuffling the Input according to the output needed. So, the shuffling Fnction is used for shuffling the system of Input data.

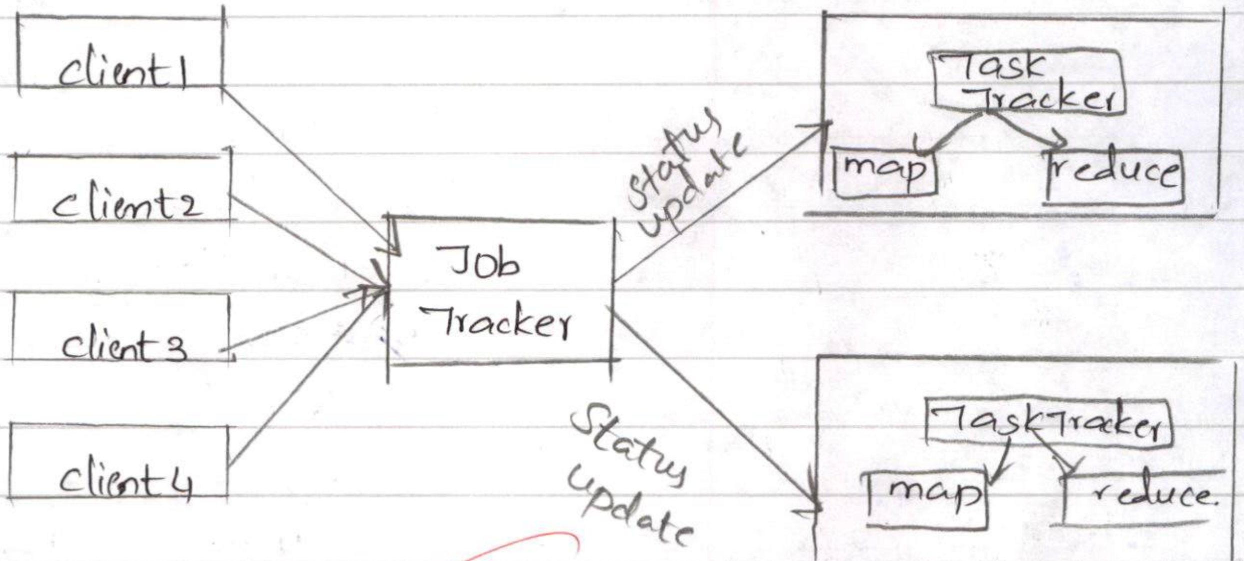
→ Reducing:

→ Removing the unwanted data & giving the output as per needed to the client. So, the Reducing Fnction is used in the mapreduce.

Architecture of mapreduce:

→ The Architecture of mapreduce states the no. of clients will be present They will seek the Information the Job Trackers & Task Trackers according to it. The data is going to be splitted. There are two main tasks are present they are: 1) map 2) Reduce the map is used for the splitting of Input data & the reduce Fnctim is used For shuffling of data & producing the output in the single Form.

diagram:



→ JobTracker:

→ The JobTracker is used for Tracking the particular Jobs assigned to the Input & and it is occured to the master & Name - NameNode is used declaring.

→ TaskTracker:

→ where the Task Tracker is used Tracking the status of the Job & updating the system according. & the dataNodes are used init.

CSE-D.

BOOKLET NUMBER :

College Stamp

R18

CMR
GROUP OF INSTITUTIONS
EXPLORE TO INVENT

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

(AUTONOMOUS)

Kandlakoya, Medchal, Hyderabad - 501 401.

MID SEMESTER EXAMINATION ANSWER BOOK

Registered No.	2	0	H	5	1	A	0	5	0	9

FIRST / [✓ SECOND] SEMESTER EXAMINATION [✓ B.Tech.]/M.Tech./MBA VII Semester NOV 2023 (Month and year)

Subject : BDA

Date : 10/11/2023

10/11

Signature of the Invigilator with date

INSTRUCTIONS TO THE CANDIDATES

1. This booklet contains 16 pages. Candidates must ensure it before writing and in case a defective answer book is issued it must be returned to the invigilator and a new and defect free booklet must be obtained.
2. Before the candidate begins to answer, registered number, particulars of year, semester, subject etc., are to be filled in. Failure to enter all or any of these particulars may disqualify the paper from valuation.
3. Candidate is prohibited from
   - (a) Writing.
     - ☞ anything addressing the examiner in any manner whatsoever, in their answer book.
     - ☞ Objectionable/obscene language in the answer book.
     - ☞ anything other than their Registered Number on the question paper.
   - (b) either seeking or providing any assistance to the fellow candidates in the exam.
   - (c) possessing a manuscript or a printed matter, in any form, in the examination hall.
   - (d) bringing loose sheets or paper into the examination hall and detaching any paper from the answer book.
   - (e) carrying Mobile Phone to Exam Hall.

   Violation of these instructions will be viewed as a case of malpractice, which is a punishable offence.
4. Before beginning to answer any question, candidates must write the correct question number, in the margin only and should not write anything else in the margin.
5. Answers must be written legibly on both sides of the paper. There shall be about 25 lines in each page. It is not necessary to begin each answer on a fresh page. Candidates should not use any other ink, except BLACK or BLUE ink.
6. Rough work, if any, must be separated, from the subject matter, by a line and noted as rough work.
7. The answer book, at the end of the examination, must be handed over to the Assistant Superintendent (Invigilator) by the candidate **This responsibility lies with the candidate only.**
8. Candidates should maintain absolute silence during the time of examination. Misbehavior, in any form, by the candidate, in the examination hall, will attract severe punishment.
9. Candidates are permitted to leave the examination hall only after the expiry of half of the allotted time and candidates will be permitted to carry the question paper only when they are leaving the exam hall in the last half-an-hour.
10. **No additional answer books will be supplied.**

To be filled in by the Examiner only

<table>
<tr><td colspan="22">PART - A / PART - B</td></tr>
<tr><td colspan="22">MARKS SLIP</td></tr>
<tr><td rowspan="2">PART-A</td><td>Q.No.</td><td colspan="2">1</td><td colspan="2">2</td><td colspan="2">3</td><td colspan="2">4</td><td colspan="2">5</td><td colspan="2">—</td><td colspan="2">—</td><td>—</td><td>—</td><td>—</td><td>Part-A Total</td></tr>
<tr><td>Marks</td><td colspan="2">2</td><td colspan="2">2</td><td colspan="2">2</td><td colspan="2">2</td><td colspan="2">2</td><td colspan="2"></td><td colspan="2"></td><td></td><td></td><td></td><td>10</td></tr>
<tr><td rowspan="3">PART-B</td><td>Q.No.</td><td colspan="2">6</td><td colspan="2">7</td><td colspan="2">8</td><td colspan="2">—</td><td colspan="2">—</td><td colspan="2">—</td><td colspan="2">—</td><td>—</td><td>—</td><td>—</td><td>Part-B Total</td></tr>
<tr><td></td><td>A</td><td>B</td><td>A</td><td>B</td><td>A</td><td>B</td><td colspan="2"></td><td colspan="2"></td><td colspan="2"></td><td colspan="2"></td><td></td><td></td><td></td><td rowspan="2">15</td></tr>
<tr><td>Marks</td><td>5</td><td></td><td></td><td>5</td><td>5</td><td></td><td colspan="2"></td><td colspan="2"></td><td colspan="2"></td><td colspan="2"></td><td></td><td></td><td></td></tr>
<tr><td colspan="17">Grand Total in Words :</td><td>GRAND TOTAL</td><td>25</td></tr>
</table>

Signature of the Scrutinizer with Date

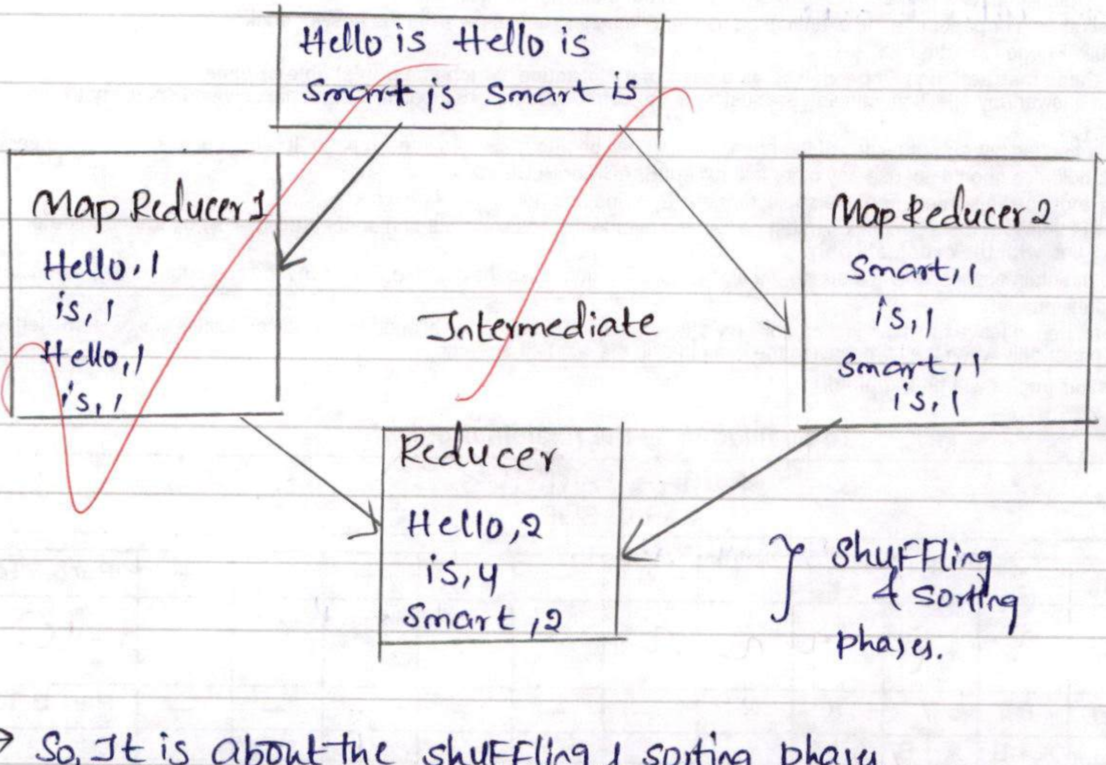
Signature of the Examiner with Date

: part-A :

1). The shuffling & sorting phases:

→ These process are mainly used in the map reduce compiler (or) Map reduce combiner where the process will be done when the map process gets completed then the shuffling & sorting phases will gets occured & then passed in the Reducer so, this known as the shuffling & sorting phases.

→ So, let us discuss with the diagram,



→ So, It is about the shuffling & sorting phases.

2). Schema:

→ where it states the Information of large dataset, where the data present in it will describe the data How is it arranged, in Structural manner & Organized way. where, the organized data will state about the (relational ship in between How they are orggized data.

→ so, It is about the schema & How it works in Big data Analystics.

→ This is about the complete detail Information of schema in big data Analytics.

3). → Data models in pig:

There are 4 types of data models they are

i) atom - where the atom stores the string (or) numerical values in it. so, it is about atom.

ii) bag - It is collection of all the tuples & stores in it. so, It is about bag.

iii) tuple - ordered field of values will be present in the tuples.

iv) map - where, it will state, the key value pairs. so, these are the various data models in pig.

4). The data-types in Hive they are:

1) Int : where it is a type of datatype where it will stores the numerical Values

i) tiny int
ii) Small int
iii) large int

2) Float : Where it will store the nearest Values in the Manner of Float.

i) tiny Float
ii) Small Float
iii) large Float

3) String : the datatype which is "name's" and describes about it.

4) Char : The char, like the place of the Values of it

5) Boolean : It will states wheather they are true (or) False.

6) date : It will state the date. It is also a type of datatype in Hive

so, these are the Various datatypes in Hive.

5). The Header Files in R their usage they are:

i) dplyr - data Impulation of System will be Explained by using this dplyr Files in R.

ii) ggplot2 - It is used in the graphical Representation & plotting of the graph.

iii) base - The normal distribution of graphs will be Explained by the base

iv) readr - Importing of the Files will get place by using the readr.

V) tiydp - used for arranging of Files which are present in the R. So, these are detail description of Files in R & their usage.

: part-B :

6)
A).

Hadoop setup on singlemode:

Hadoop:

→ It is a open source system where it is used store the large data & process the system. Where open source system is based on java. The store data will be disturbed & process will takes parallelly so; the system works properly & Equally time gets saved of it. So, It is known as Hadoop.

→ Hadoop setup will be done by using two modes they are singlemode & multimode.

→ Singlemode: where the Namenode, datanode in working on the single manchine where; the Multimode working on the different machines at a-time.

→ So, let us disuss in detail about the Hadoop setup on single mode step by step process indetail of it.

→ Hadoop setup on singlenode:-

let, us discuss Indetail of it step by step so, let go.

Step1: Installation of Hadoop from tar/
→Extraction of Hadoop From tar & keep into the File so, by using this below command we are doing it.

```
tar -xvf hadoop-2.7.3.tar.gz.
```

Step2: Add the Hadoop & java paths into (.base-profile) file.
→ Add the Hadoop Exacted File & Add the java paths in to the (base-profile) File. so, Salled it and run. after saving check Wheather they are installed (or) not by using

```
java -version -Hadoop version.
```

Step3: Edit the Configuration of the Files.
→ Edit the Files according to the needed configuation for the system required.

Step4: Open crore-site.xml & add the property of configuration to it where it is a type of Hadoop daemon. where the NameNode is running on the cluster. It contains the HDFs & mapreduce init,

Step5: open HdFs-site.xml & Add the property confrguation to it.
→ where it contains the Namenode, dataNode, secondary dataNode, & It also Includes the Replication Factor & Size of the block. So, It Main work of the Hdfs-site.xml.

Step6: open mapred-site.xml & Add the property confrguation to it.
→ where it will run parallel to the JVm Compiler & uses the map size of the process required. where additional we can get the core proces if required.

Step7: open yarn-site.xml & Add the property confirguation tag.
→ Where it will contain ResourceManager & Node Manager. they are application of Memory Management & The operates the program & Analysis.

Step8: Go to the Hadoop & setupNameNode.
→ After Completing all the proces go the Hadoop & directory Name Node have to get setupped according.

Step9: Once the NameNode gets Stepup then go the directory and run the path of it by typing the path as,

```
https://google//doc/html - ./start/dfs.pig.
```

So, this is the detail proces of Hadoop setup on SingleNode. & steps to be Followed.

7) The Apache pig:
B.

→ The Apache pig plays an important key role in the Hadoop. where the people are new to hadoop. they can Easily do the process by taking the help of Apache pig.

→ It Can Easily analysis the large data. And it contains more no. of datatypes.

→ where the user can Easily build his own functions & they are many builtin functions. So, It is about Apache pig.

→ Apache pig parameters substitution:

→ Apache pig it is a type of software system. where the first create one file which is in execution on script and now we can add the different parameters to it. this process called as parameter substitution. It can be done by using the "param".

→ so, let us consider one file name it is as number
12
16
18
20 these are the data present in the number file.

So, we can run the system by using, selecting the specific number .

```
Number = path 'data/load' as
(number:int)
Specific = = 12 filter by Number //selecting specific number.
bump specific; // printing of number.
```

→ After it we have to run in the command by typing the

```
cmd-f/path/to/numbers.pig
```

→ where, we can selecting for any dynamic number, then, we have to the $dynanumber. so, let us discuss in detail of it

```
Number= path 'data/load/' as
(number:int)     $path
specific== $dynanumber // dynamicnumber.
```

→ dump specific; // printing

→ So, After completing we have to run the compiler by typing the Command as,

Commad.

→
```
cmd - param/F/path/to/numbers.pig
```

→ So, this is process for printing the dynamic number - the parameter substution of the pig.

→ So, the param is used in the dynamic Substitution of the numbers.

→ By using above command we are declaring the dynamic number & "$" symbol used.

→ This is the detail description of the Apache pig parameter substitution.

8) → Hive Architecture:

A). → Hive:

→ where the Hive it is a type of platform where it is a data warehouse Infrastructure & used For the SQL queries to be done. It is also used for summarizing the large data. Easily. where it is used For the quering & analyzing the data.

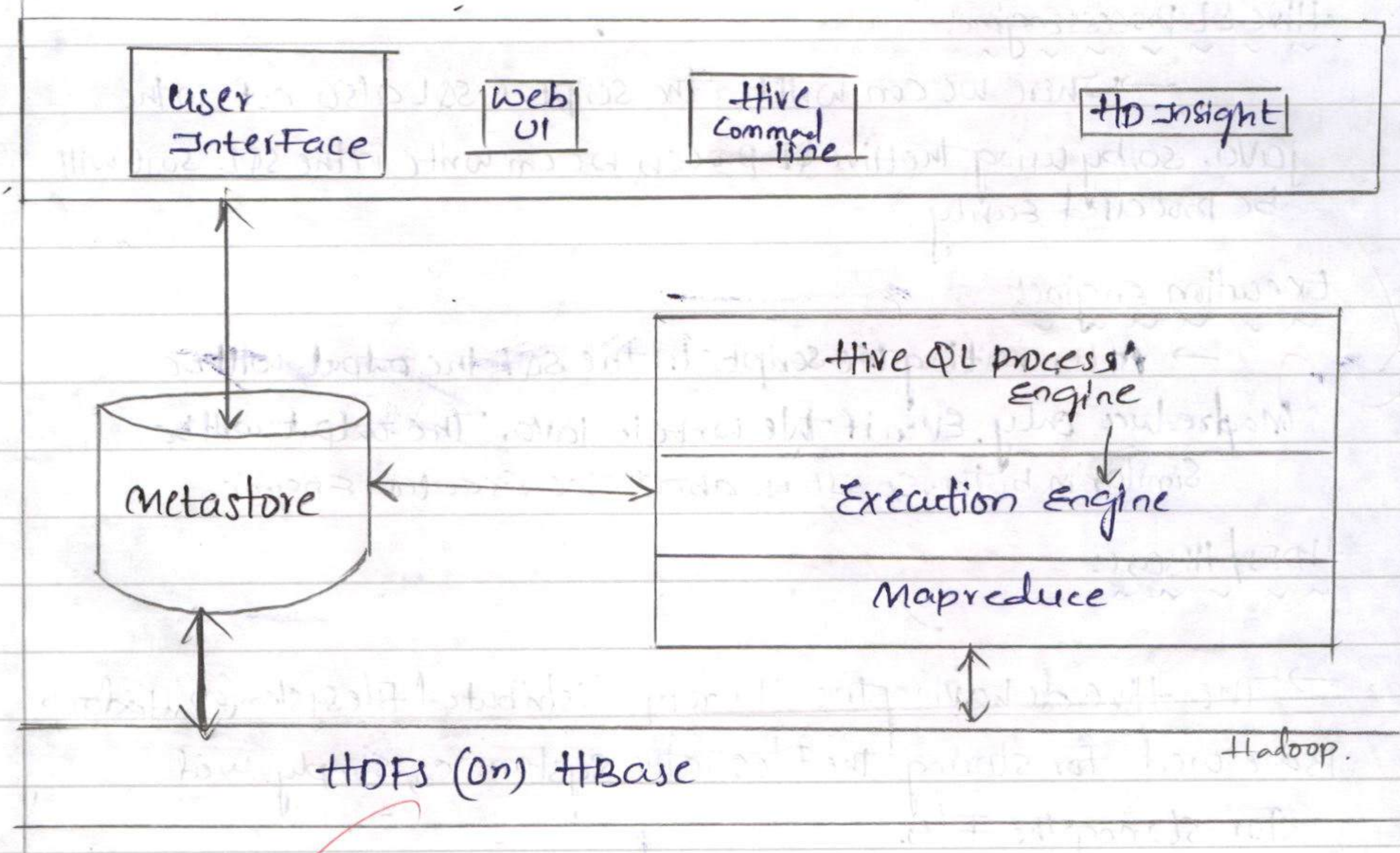
→ The data mainpulation, ddl, dml, dcl, and all types of the procen will be done on it.

→ It supports the SQL languages.

→ First is was discovered by Facebook later it was handover to the Apache Hive under them these proces is done they named it as Hive software. So, let us discuss about the Architecture of Hive.

→ Hive Architecture:

→ let us discuss the Architecture of Hive in detail, in below.



→ This is the Architecture Hive let us, discuss the Each term indetail.

→ User Interface:

→ Hive is a data warehouse Infrastructure where it used to Communicate b/w user & HDFs. The Interface contains the resources as webUI, Hive commandline, HD Insights), gch the process to be done.

→ Metastore:

→ Where it is a type of stoage used For storing the datatypes, Metadata of tables, column in the table. so, simply it will store the data.

→ Hive QL process Engine:

→ Where we can written the script in SQL also not in the java. so, by using the Hive QL process we can write in the SQL so, it will be processed Easily.

→ Execution Engine:

→ After writing the script in the SQL the output will be Mapreduce only. Even if We write in java. The output will be Similar in both. So, It is about the Exection Engine.

→ HDFs/HBase:

→ The Hive ds having the Hadoop distributed Filesystem (or Hadoop Base used For storing The Files in the System. So, Simply used For storing the Files.

→ So, this is the detail discription of Hive & Architecture of Hive.

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

(UGC AUTONOMOUS)

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD-501 401

ASSESSMENT OF COURSE OUTCOMES

PROGRAMME		**B.TECH (CSE)**					
YEAR	IV	SEM	VII	Academic Year	2022-23	BATCH	*2019-2023*
Course Code		A30013		Course Name	BMFA		

ASSESSMENT OF COURSE OUTCOMES THROUGH EXTERNAL EXAMINAION MARKS

CO TARGET VALUES	
>= 50	<50
Y	N

ATTAINMENT TARGET %		
>= 60	<=65 & >55	<55
3	2	1

COURSE OUTCOMES ATTAINMENT

CO	CO1	CO2	CO3	CO4	CO5
No of Students attempted	2008	2008	2008	2008	2008
Number of students attained	1933	1888	1888	1615	1540
CO Attainment %	96.26%	94.02%	94.02%	80.43%	76.69%
Attainment Level	3	3	3	3	3

ASSESSMENT OF COURSE OUTCOMES THROUGH INTERNAL EXAMINAION MARKS

CO TARGET VALUES	
>= 50	<50
Y	N

ATTAINMENT TARGET %		
>= 60	<=60 & >50	<50
3	3	1

COURSE OUTCOMES ATTAINMENT

CO	CO1	CO2	CO3	CO4	CO5
No of Students attempted	258	251	249	263	258
Number of students attained	244	223	223	247	238
CO Attainment %	94.57%	88.84%	91.80%	92.25%	97.30%
Attainment Level	3	3	3	3	3

(Course Coordinator) **(Programme Coordinator)**

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

(UGC AUTONOMOUS)

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD-501 401

ASSESSMENT OF COURSE OUTCOMES (EXTERNAL EXAMINAION)

PROGRAMME **B.TECH (CSE)**

YEAR IV SEM VII Academic Year 2022-23 BATCH ***2019-2023***

Course Code A30013 Course Name BMFA

CO TARGET VALUES	
>= 50	<50
Y	N

ATTAINMENT TARGET %		
>= 60	<=65 & >55	<55
3	2	1

COURSE OUTCOMES ATTAINMENT

CO	CO1	CO2	CO3	CO4	CO5
No of Students attempted	**2008**	**2008**	**2008**	**2008**	**2008**
Number of students attained	**1933**	**1888**	**1888**	**1615**	**1540**
CO Attainment %	**96.26%**	**94.02%**	**94.02%**	**80.43%**	**76.69%**
Attainment Level	**3**	**3**	**3**	**3**	**3**

(Course Coordinator) **(Programme Coordinator)**

DATA FOR EVALUATION OF COURSE OUTCOMES (EXTERNAL EXAMINATIONS) - BMFA

S.No	OMR CODE	Q11			Q12			Q13			Q14			Q15		
		10	%	AT	10	%	AT	10	%	AT	10	%	AT	10	%	AT
1	679583	9	90%	Y	8	80%	Y	9	90%	Y	9	90%	Y	7	70%	Y
2	679585	8	80%	Y	8	80%	Y	9	90%	Y	6	60%	Y	8	80%	Y
3	679587	9	90%	Y	5	50%	Y	7	70%	Y	5	50%	Y	7	70%	Y
4	679590	7	70%	Y	7	70%	Y	8	80%	Y	8	80%	Y	7	70%	Y
5	679592	8	80%	Y	7	70%	Y	8	80%	Y	4	40%	N	3	30%	N
6	679594	8	80%	Y	6	60%	Y	7	70%	Y	6	60%	Y	4	40%	N
7	679596	7	70%	Y	9	90%	Y	8	80%	Y	6	60%	Y	7	70%	Y
8	679598	7	70%	Y	7	70%	Y	8	80%	Y	7	70%	Y	7	70%	Y
9	679600	8	80%	Y	5	50%	Y	7	70%	Y	5	50%	Y	7	70%	Y
10	679602	8	80%	Y	8	80%	Y	6	60%	Y	4	40%	N	3	30%	N
11	679604	8	80%	Y	7	70%	Y	8	80%	Y	6	60%	Y	6	60%	Y
12	679606	8	80%	Y	6	60%	Y	7	70%	Y	5	50%	Y	5	50%	Y
13	679608	8	80%	Y	6	60%	Y	8	80%	Y	3	30%	N	7	70%	Y

Course materials (lecture notes,ppt)

UNIT – I

What is Big Data?

According to Gartner, the definition of Big Data –

"Big data" is high-volume, velocity, and variety information assets that demand cost-effective, innovative forms of information processing for enhanced insight and decision making."

This definition clearly answers the "What is Big Data?" question – Big Data refers to complex and large data sets that have to be processed and analyzed to uncover valuable information that can benefit businesses and organizations.

However, there are certain basic tenets of Big Data that will make it even simpler to answer what is Big Data:

- It refers to a massive amount of data that keeps on growing exponentially with time.
- It is so voluminous that it cannot be processed or analyzed using conventional data processing techniques.
- It includes data mining, data storage, data analysis, data sharing, and data visualization.
- The term is an all-comprehensive one including data, data frameworks, along with the tools and techniques used to process and analyze the data.

The History of Big Data

Although the concept of big data itself is relatively new, the origins of large data sets go back to the 1960s and '70s when the world of data was just getting started with the first data centers and the development of the relational database.

Around 2005, people began to realize just how much data users generated through Facebook, YouTube, and other online services. Hadoop (an open-source framework created specifically to store and analyze big data sets) was developed that same year. NoSQL also began to gain popularity during this time.

The development of open-source frameworks, such as Hadoop (and more recently, Spark) was essential for the growth of big data because they make big data easier to work with and cheaper to store. In the years since then, the volume of big data has skyrocketed. Users are still generating huge amounts of data—but it's not just humans who are doing it.

With the advent of the Internet of Things (IoT), more objects and devices are connected to the internet, gathering data on customer usage patterns and product performance. The emergence of machine learning has produced still more data.

While big data has come far, its usefulness is only just beginning. Cloud computing has expanded big data possibilities even further. The cloud offers truly elastic scalability, where developers can simply spin up ad hoc clusters to test a subset of data.

Benefits of Big Data and Data Analytics

- Big data makes it possible for you to gain more complete answers because you have more information.
- More complete answers mean more confidence in the data—which means a completely different approach to tackling problems.

Types of Big Data

Now that we are on track with what is big data, let's have a look at the types of big data:

a) Structured

Structured is one of the types of big data and By structured data, we mean data that can be processed, stored, and retrieved in a fixed format. It refers to highly organized information that can be readily and seamlessly stored and accessed from a database by simple search engine algorithms. **For instance, the employee table in a company database will be structured as the employee details, their job positions, their salaries, etc.,** will be present in an organized manner.

b) Unstructured

Unstructured data refers to the data that lacks any specific form or structure whatsoever. This makes it very difficult and time-consuming to process and analyze unstructured data. Email is an example of unstructured data. Structured and unstructured are two important types of big data.

c) Semi-structured

Semi structured is the third type of big data. Semi-structured data pertains to the data containing both the formats mentioned above, that is, structured and unstructured data. To be precise, it refers to the data that although has not been classified under a particular repository (database), yet contains vital information or tags that segregate individual elements within the data. Thus we come to the end of types of data.

Characteristics of Big Data

Back in 2001, Gartner analyst Doug Laney listed the **3 'V's of Big Data – Variety, Velocity, and Volume.** Let's discuss the characteristics of big data. These characteristics, isolated, are enough to know what big data is. Let's look at them in depth:

a) Variety

Variety of Big Data refers to structured, unstructured, and semi-structured data that is gathered from multiple sources. While in the past, data could only be collected from spreadsheets and databases, today data comes in an array of forms such as emails, PDFs, photos, videos, audios, SM posts, and so much more. Variety is one of the important characteristics of big data.

b) Velocity

Velocity essentially refers to the speed at which data is being created in real-time. In a broader prospect, it comprises the rate of change, linking of incoming data sets at varying speeds, and activity bursts.

c) Volume

Volume is one of the characteristics of big data. We already know that Big Data indicates huge 'volumes' of data that is being generated on a daily basis from various sources like social media platforms, business processes, machines, networks, human interactions, etc. Such a large amount of data is stored in data warehouses. Thus comes to the end of characteristics of big data.

Why is Big Data Important?

The importance of big data does not revolve around how much data a company has but how a company utilizes the collected data. Every company uses data in its own way; the more efficiently a company uses its data, the more potential it has to grow. The company can take data from any source and analyze it to find answers which will enable:

1. **Cost Savings**: Some tools of Big Data like Hadoop and Cloud-Based Analytics can bring cost advantages to business when large amounts of data are to be stored and these tools also help in identifying more efficient ways of doing business.
2. **Time Reductions: The** high speed of tools like Hadoop and in-memory analytics can easily identify new sources of data which helps businesses analyzing data immediately and make quick decisions based on the learning.
3. **Understand the market conditions**: By analyzing big data you can get a better understanding of current market conditions. For example, by analyzing customers' purchasing behaviors, a company can find out the products that are sold the most and produce products according to this trend. By this, it can get ahead of its competitors.
4. **Control online reputation:** Big data tools can do sentiment analysis. Therefore, you can get feedback about who is saying what about your company. If you want to monitor and improve the online presence of your business, then, big data tools can help in all this.
5. **Using Big Data Analytics to Boost Customer Acquisition and Retention**
   The customer is the most important asset any business depends on. There is no single business that can claim success without first having to establish a solid customer base. However, even with a customer base, a business cannot afford to disregard the high competition it faces. If a business is slow to learn what customers are looking for, then it is very easy to begin offering poor quality products. In the end, loss of clientele will result, and this creates an adverse overall effect on business success. The use of big data allows businesses to observe various customer related patterns and trends. Observing customer behavior is important to trigger loyalty.
6. **Using Big Data Analytics to Solve Advertisers Problem and Offer Marketing Insights**

Big data analytics can help change all business operations. This includes the ability to match customer expectation, changing company's product line and of course ensuring that the marketing campaigns are powerful.

7. **Big Data Analytics As a Driver of Innovations and Product Development**
   Another huge advantage of big data is the ability to help companies innovate and redevelop their products.

Business Intelligence vs Big Data

Although Big Data and Business Intelligence are two technologies used to analyze data to help companies in the decision-making process, there are differences between both of them. They differ in the way they work as much as in the type of data they analyze.

Traditional BI methodology is based on the principle of grouping all business data into a central server. Typically, this data is analyzed in offline mode, after storing the information in an environment called Data Warehouse. The data is structured in a conventional relational database with an additional set of indexes and forms of access to the tables (multidimensional cubes).

A Big Data solution differs in many aspects to BI to use. These are the main differences between Big Data and Business Intelligence:

1. In a Big Data environment, information is stored on a distributed file system, rather than on a central server. It is a much safer and more flexible space.
2. Big Data solutions carry the processing functions to the data, rather than the data to the functions. As the analysis is centered on the information, it´s easier to handle larger amounts of information in a more agile way.
3. Big Data can analyze data in different formats, both structured and unstructured. The volume of unstructured data (those not stored in a traditional database) is growing at levels much higher than the structured data. Nevertheless, its analysis carries different challenges. Big Data solutions solve them by allowing a global analysis of various sources of information.
4. Data processed by Big Data solutions can be historical or come from real-time sources. Thus, companies can make decisions that affect their business in an agile and efficient way.
5. Big Data technology uses parallel mass processing (MPP) concepts, which improves the speed of analysis. With MPP many instructions are executed simultaneously, and since the various jobs are divided into several parallel execution parts, at the end the overall results are reunited and presented. This allows you to analyze large volumes of information quickly.

Big Data vs Data Warehouse

Big Data has become the reality of doing business for organizations today. There is a boom in the amount of structured as well as raw data that floods every organization daily. If this data is managed well, it can lead to powerful insights and quality decision making.

Big data analytics is the process of examining large data sets containing a variety of data types to discover some knowledge in databases, to identify interesting patterns and establish relationships to solve problems, market trends, customer preferences, and other useful information. Companies and businesses that implement Big Data Analytics often reap several business benefits. Companies implement Big Data Analytics because they want to make more informed business decisions.

A data warehouse (DW) is a collection of corporate information and data derived from operational systems and external data sources. A data warehouse is designed to support business decisions by allowing data consolidation, analysis and reporting at different aggregate levels. Data is populated into the Data Warehouse through the processes of extraction, transformation and loading (ETL tools). Data analysis tools, such as business intelligence software, access the data within the warehouse.

Hadoop Environment Big Data Analytics

Hadoop is changing the perception of handling Big Data especially the unstructured data. Let's know how Apache Hadoop software library, which is a framework, plays a vital role in handling Big Data. Apache Hadoop enables surplus data to be streamlined for any distributed processing system across clusters of computers using simple programming models. It truly is made to scale up from single servers to a large number of machines, each and every offering local computation, and storage space. Instead of depending on hardware to provide high-availability, the library itself is built to detect and handle breakdowns at the application layer, so providing an extremely available service along with a cluster of computers, as both versions might be vulnerable to failures.

Hadoop Community Package Consists of

- File system and OS level abstractions
- A MapReduce engine (either MapReduce or YARN)
- The Hadoop Distributed File System (HDFS)
- Java ARchive (JAR) files
- Scripts needed to start Hadoop
- Source code, documentation and a contribution section

Activities performed on Big Data

- **Store** – Big data need to be collected in a seamless repository, and it is not necessary to store in a single physical database.
- **Process** – The process becomes more tedious than traditional one in terms of cleansing, enriching, calculating, transforming, and running algorithms.
- **Access** – There is no business sense of it at all when the data cannot be searched, retrieved easily, and can be virtually showcased along the business lines.

Classification of analytics

Descriptive analytics

Descriptive analytics is a statistical method that is used to search and summarize historical data in order to identify patterns or meaning.

Data aggregation and **data mining** are two techniques used in descriptive analytics to discover historical data. Data is first gathered and sorted by data aggregation in order to make the datasets more manageable by analysts.

Data mining describes the next step of the analysis and involves a search of the data to identify patterns and meaning. Identified patterns are analyzed to discover the specific ways that learners interacted with the learning content and within the learning environment.

Advantages:

- Quickly and easily report on the Return on Investment (ROI) by showing how performance achieved business or target goals.
- Identify gaps and performance issues early - before they become problems.
- Identify specific learners who require additional support, regardless of how many students or employees there are.
- Identify successful learners in order to offer positive feedback or additional resources.
- Analyze the value and impact of course design and learning resources.

Predictive analytics

Predictive Analytics is a statistical method that utilizes algorithms and machine learning to identify trends in data and predict future behaviors

The software for predictive analytics has moved beyond the realm of statisticians and is becoming more affordable and accessible for different markets and industries, including the field of learning & development.

For online learning specifically, predictive analytics is often found incorporated in the Learning Management System (LMS), but can also be purchased separately as specialized software.

For the learner, predictive forecasting could be as simple as a dashboard located on the main screen after logging in to access a course. Analyzing data from past and current progress, visual indicators in the dashboard could be provided to signal whether the employee was on track with training requirements.

Advantages:

- **Personalize the training needs** of employees by identifying their gaps, strengths, and weaknesses; specific learning resources and training can be offered to support individual needs.
- **Retain Talent** by tracking and understanding employee career progression and forecasting what skills and learning resources would best benefit their career paths. Knowing what skills employees need also benefits the design of future training.
- **Support employees** who may be falling behind or not reaching their potential by offering intervention support before their performance puts them at risk.
- **Simplified reporting** and visuals that keep everyone updated when predictive forecasting is required.

Prescriptive analytics

Prescriptive analytics is a statistical method used to generate recommendations and make decisions based on the computational findings of algorithmic models.

Generating automated decisions or recommendations requires specific and unique algorithmic models and clear direction from those utilizing the analytical technique. A recommendation cannot be generated without knowing what to look for or what problem is desired to be solved. In this way, prescriptive analytics begins with a problem.

Example

A Training Manager uses predictive analysis to discover that most learners without a particular skill will not complete the newly launched course. What could be done? Now prescriptive analytics can be of assistance on the matter and help determine options for action. Perhaps an algorithm can detect the learners who require that new course, but lack that particular skill, and send an automated recommendation that they take an additional training resource to acquire the missing skill.

The accuracy of a generated decision or recommendation, however, is only as good as the quality of data and the algorithmic models developed. What may work for one company's training needs may not make sense when put into practice in another company's training department. Models are generally recommended to be tailored for each unique situation and need.

Descriptive vs Predictive vs Prescriptive Analytics

Descriptive Analytics is focused solely on historical data.

You can think of Predictive Analytics as then using this historical data to develop statistical models that will then forecast about future possibilities.

Prescriptive Analytics takes Predictive Analytics a step further and takes the possible forecasted outcomes and predicts consequences for these outcomes.

What Big Data Analytics Challenges

1. Need For Synchronization Across Disparate Data Sources

As data sets are becoming bigger and more diverse, there is a big challenge to incorporate them into an analytical platform. If this is overlooked, it will create gaps and lead to wrong messages and insights.

2. Acute Shortage Of Professionals Who Understand Big Data Analysis

The analysis of data is important to make this voluminous amount of data being produced in every minute, useful. With the exponential rise of data, a huge demand for big data scientists and Big Data analysts has been created in the market. It is important for business organizations to hire a data scientist having skills that are varied as the job of a data scientist is multidisciplinary. Another major challenge faced by businesses is the shortage of professionals who understand Big Data analysis. There is a sharp shortage of data scientists in comparison to the massive amount of data being produced.

3. Getting Meaningful Insights Through The Use Of Big Data Analytics

It is imperative for business organizations to gain important insights from Big Data analytics, and also it is important that only the relevant department has access to this information. A big challenge faced by the companies in the Big Data analytics is mending this wide gap in an effective manner.

4. Getting Voluminous Data Into The Big Data Platform

It is hardly surprising that data is growing with every passing day. This simply indicates that business organizations need to handle a large amount of data on daily basis. The amount and variety of data available these days can overwhelm any data engineer and that is why it is considered vital to make data accessibility easy and convenient for brand owners and managers.

5. Uncertainty Of Data Management Landscape

With the rise of Big Data, new technologies and companies are being developed every day. However, a big challenge faced by the companies in the Big Data analytics is to find out which technology will be best suited to them without the introduction of new problems and potential risks.

6. Data Storage And Quality

Business organizations are growing at a rapid pace. With the tremendous growth of the companies and large business organizations, increases the amount of data produced. The storage of this massive amount of data is becoming a real challenge for everyone. Popular data storage options like data lakes/ warehouses are commonly used to gather and store large quantities of unstructured and structured data in its native format. The real problem arises when a data lakes/ warehouse try to combine unstructured and inconsistent data from diverse sources, it encounters errors. Missing data, inconsistent data, logic conflicts, and duplicates data all result in data quality challenges.

7. Security And Privacy Of Data

Once business enterprises discover how to use Big Data, it brings them a wide range of possibilities and opportunities. However, it also involves the potential risks associated with big data when it comes to the privacy and the security of the data. The Big Data tools used for analysis and storage utilizes the data disparate sources. This eventually leads to a high risk of exposure of the data, making it vulnerable. Thus, the rise of voluminous amount of data increases privacy and security concerns.

Terminologies Used In Big Data Environments

- **As-a-service infrastructure**

Data-as-a-service, software-as-a-service, platform-as-a-service – all refer to the idea that rather than selling data, licences to use data, or platforms for running Big Data technology, it can be provided "as a service", rather than as a product. This reduces the upfront capital investment necessary for customers to begin putting their data, or platforms, to work for them, as the provider bears all of the costs of setting up and hosting the infrastructure. As a customer, as-a-service infrastructure can greatly reduce the initial cost and setup time of getting Big Data initiatives up and running.

- **Data science**

Data science is the professional field that deals with turning data into value such as new insights or predictive models. It brings together expertise from fields including statistics, mathematics, computer science, communication as well as domain expertise such as business knowledge. Data scientist has recently been voted the No 1 job in the U.S., based on current demand and salary and career opportunities.

- **Data mining**

Data mining is the process of discovering insights from data. In terms of Big Data, because it is so large, this is generally done by computational methods in an automated way using methods such as decision trees, clustering analysis and, most recently, machine learning. This can be thought of as using the brute mathematical power of computers to spot patterns in data which would not be visible to the human eye due to the complexity of the dataset.

- **Hadoop**

Hadoop is a framework for Big Data computing which has been released into the public domain as open source software, and so can freely be used by anyone. It consists of a number of modules all tailored for a different vital step of the Big Data process – from file storage (Hadoop File System – HDFS) to database (HBase) to carrying out data operations (Hadoop MapReduce – see below). It has become so popular due to its power and flexibility that it has developed its own industry of retailers (selling tailored versions), support service providers and consultants.

- **Predictive modelling**

At its simplest, this is predicting what will happen next based on data about what has happened previously. In the Big Data age, because there is more data around than ever before, predictions are becoming more and more accurate. Predictive modelling is a core component of most Big Data initiatives, which are formulated to help us choose the course of action which will lead to the most desirable outcome. The speed of modern computers and the volume of data available means that predictions can be made based on a huge number of variables, allowing an ever-increasing number of variables to be assessed for the probability that it will lead to success.

- **MapReduce**

MapReduce is a computing procedure for working with large datasets, which was devised due to difficulty of reading and analysing really Big Data using conventional computing methodologies. As its name suggest, it consists of two procedures – mapping (sorting information into the format needed for analysis – i.e. sorting a list of people according to their age) and reducing (performing an operation, such checking the age of everyone in the dataset to see who is over 21).

UNIT II

NoSQL

NoSQL is a non-relational DMS, that does not require a fixed schema, avoids joins, and is easy to scale. NoSQL database is used for distributed data stores with humongous data storage needs. NoSQL is used for Big data and real-time web apps. For example companies like Twitter, Facebook, Google that collect terabytes of user data every single day.

SQL

Structured Query language (SQL) **pronounced as "S-Q-L" or sometimes as "See-Quel"** is the standard language for dealing with Relational Databases. A relational database defines relationships in the form of tables.

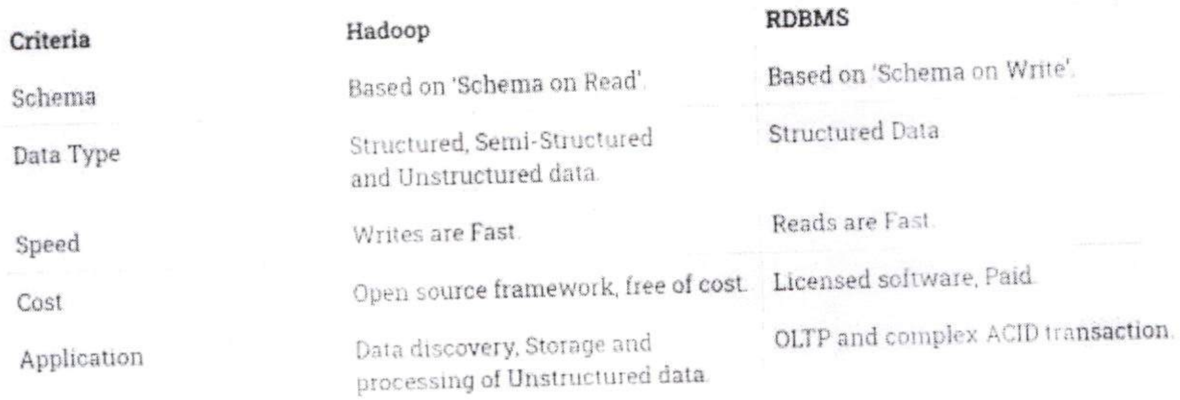
SQL programming can be effectively used to insert, search, update, delete database records.

Comparison of SQL and NoSQL

Parameter	SQL	NOSQL
Definition	SQL databases are primarily called RDBMS or Relational Databases	NoSQL databases are primarily called as Non-relational or distributed database
Design for	Traditional RDBMS uses SQL syntax and queries to analyze and get the data for further insights. They are used for OLAP systems.	NoSQL database system consists of various kind of database technologies. These databases were developed in response to the demands presented for the development of the modern application.
Query Language	Structured query language (SQL)	No declarative query language
Type	SQL databases are table based databases	NoSQL databases can be document based, key-value pairs, graph databases
Schema	SQL databases have a predefined schema	NoSQL databases use dynamic schema for unstructured data.
Ability to scale	SQL databases are vertically scalable	NoSQL databases are horizontally scalable
Examples	Oracle, Postgres, and MS-SQL.	MongoDB, Redis, , Neo4j, Cassandra, Hbase.
Best suited for	An ideal choice for the complex query intensive environment.	It is not good fit complex queries.
Hierarchical data storage	SQL databases are not suitable for hierarchical data storage.	More suitable for the hierarchical data store as it supports key-value pair method.
Variations	One type with minor variations.	Many different types which include key-value stores, document databases, and graph databases.

Development Year	It was developed in the 1970s to deal with issues with flat file storage	Developed in the late 2000s to overcome issues and limitations of SQL databases.
Open-source	A mix of open-source like Postgres & MySQL, and commercial like Oracle Database.	Open-source
Consistency	It should be configured for strong consistency.	It depends on DBMS as some offers strong consistency like MongoDB, whereas others offer only offers eventual consistency, like Cassandra.
Best Used for	RDBMS database is the right option for solving ACID problems.	NoSQL is a best used for solving data availability problems
Importance	It should be used when data validity is super important	Use when it's more important to have fast data than correct data
Best option	When you need to support dynamic queries	Use when you need to scale based on changing requirements
Hardware	Specialized DB hardware (Oracle Exadata, etc.)	Commodity hardware
Network	Highly available network (Infiniband, Fabric Path, etc.)	Commodity network (Ethernet, etc.)
Storage Type	Highly Available Storage (SAN, RAID, etc.)	Commodity drives storage (standard HDDs, JBOD)
Best features	Cross-platform support, Secure and free	Easy to use, High performance, and Flexible tool.
Top Companies Using	Hootsuite, CircleCI, Gauges	Airbnb, Uber, Kickstarter
Average salary	The average salary for any professional SQL Developer is $84,328 per year in the U.S.A.	The average salary for "NoSQL developer" ranges from approximately $72,174 per year
ACID vs. BASE Model	ACID(Atomicity, Consistency, Isolation, and Durability) is a standard for RDBMS	Base (Basically Available, Soft state, Eventually Consistent) is a model of many NoSQL systems

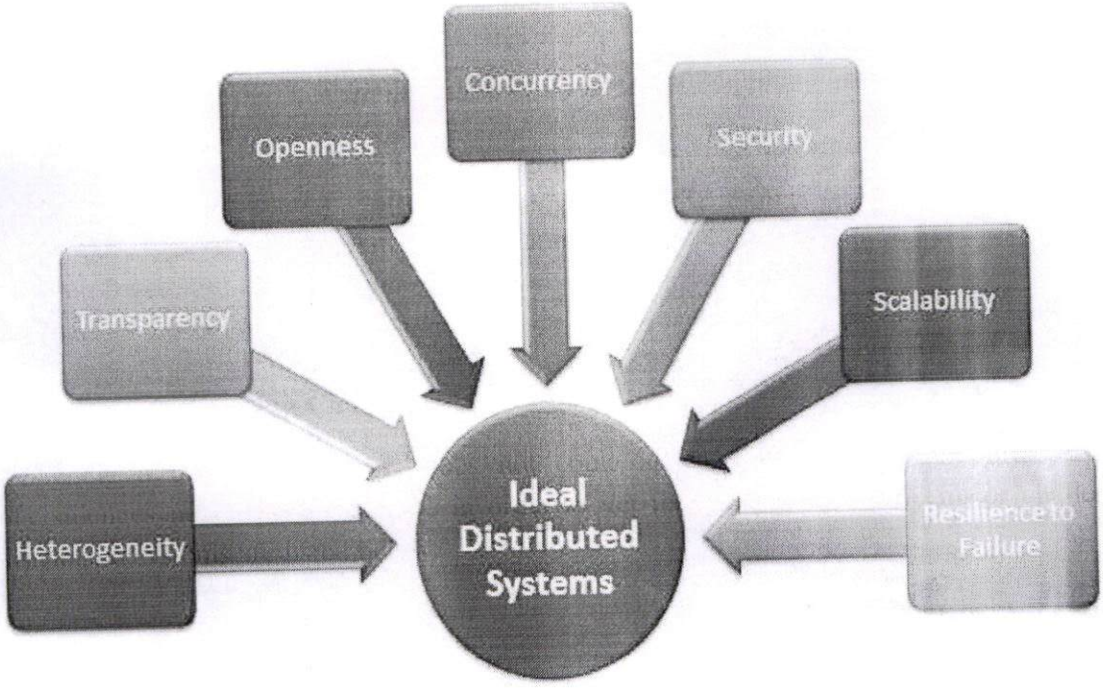
RDBMS Versus Hadoop



Criteria	Hadoop	RDBMS
Schema	Based on 'Schema on Read'.	Based on 'Schema on Write'.
Data Type	Structured, Semi-Structured and Unstructured data.	Structured Data
Speed	Writes are Fast.	Reads are Fast.
Cost	Open source framework, free of cost.	Licensed software, Paid.
Application	Data discovery, Storage and processing of Unstructured data.	OLTP and complex ACID transaction.

Distributed Computing Challenges

Designing a distributed system does not come as easy and straight forward. A number of challenges need to be overcome in order to get the ideal system. The major challenges in distributed systems are listed below:



1. Heterogeneity:

The Internet enables users to access services and run applications over a heterogeneous collection of computers and networks. Heterogeneity (that is, variety and difference) applies to all of the following:

- Hardware devices: computers, tablets, mobile phones, embedded devices, etc.
- Operating System: Ms Windows, Linux, Mac, Unix, etc.
- Network: Local network, the Internet, wireless network, satellite links, etc.
- Programming languages: Java, C/C++, Python, PHP, etc.
- Different roles of software developers, designers, system managers

Different programming languages use different representations for characters and data structures such as arrays and records. These differences must be addressed if programs written in different languages are to be able to communicate with one another. Programs written by different developers cannot communicate with one another unless they use common standards, for example, for network communication and the representation of primitive data items and data structures in messages. For this to happen, standards need to be agreed and adopted – as have the Internet protocols.

Middleware: The term middleware applies to a software layer that provides a programming abstraction as well as masking the heterogeneity of the underlying networks, hardware, operating systems and programming languages. Most middleware is implemented over the Internet protocols, which themselves mask the differences of the underlying networks, but all middleware deals with the differences in operating systems and hardware

Heterogeneity and mobile code: The term mobile code is used to refer to program code that can be transferred from one computer to another and run at the destination – Java applets are an example. Code suitable for running on one computer is not necessarily suitable for running on another because executable programs are normally specific both to the instruction set and to the host operating system.

2. Transparency:

Transparency is defined as the concealment from the user and the application programmer of the separation of components in a distributed system, so that the system is perceived as a whole rather than as a collection of independent components. In other words, distributed systems designers must hide the complexity of the systems as much as they can. Some terms of transparency in distributed systems are:

Access Hide differences in data representation and how a resource is accessed

Location Hide where a resource is located

Migration Hide that a resource may move to another location

Relocation Hide that a resource may be moved to another location while in use

Replication Hide that a resource may be copied in several places

Concurrency Hide that a resource may be shared by several competitive users

Failure Hide the failure and recovery of a resource

Persistence Hide whether a (software) resource is in memory or a disk

3. Openness

The openness of a computer system is the characteristic that determines whether the system can be extended and re-implemented in various ways. The openness of distributed systems is determined primarily by the degree to which new resource-sharing services can be added and be made available for use by a variety of client programs. If the well-defined interfaces for a system are published, it is easier for developers to add new features or replace sub-systems in the future. Example: Twitter and Facebook have API that allows developers to develop their own software interactively.

4. Concurrency

Both services and applications provide resources that can be shared by clients in a distributed system. There is therefore a possibility that several clients will attempt to access a shared resource at the same time. For example, a data structure that records bids for an auction may be accessed very frequently when it gets close to the deadline time. For an object to be safe in a concurrent environment, its operations must be synchronized in such a way that its data remains consistent. This can be achieved by standard techniques such as semaphores, which are used in most operating systems.

5. Security

Many of the information resources that are made available and maintained in distributed systems have a high intrinsic value to their users. Their security is therefore of considerable importance. Security for information resources has three components: ***confidentiality*** (protection against disclosure to unauthorized individuals) ***integrity*** (protection against alteration or corruption), ***availability*** for the authorized (protection against interference with the means to access the resources).

6. Scalability

Distributed systems must be scalable as the number of user increases. The scalability is defined by B. Clifford Neuman as

A system is said to be scalable if it can handle the addition of users and resources without suffering a noticeable loss of performance or increase in administrative complexity

Scalability has 3 dimensions:

- Size
  - Number of users and resources to be processed. Problem associated is overloading
- Geography
  - Distance between users and resources. Problem associated is communication reliability
- Administration
  - As the size of distributed systems increases, many of the system needs to be controlled. Problem associated is administrative mess

7. Failure Handling

Computer systems sometimes fail. When faults occur in hardware or software, programs may produce incorrect results or may stop before they have completed the intended computation. The handling of failures is particularly difficult.

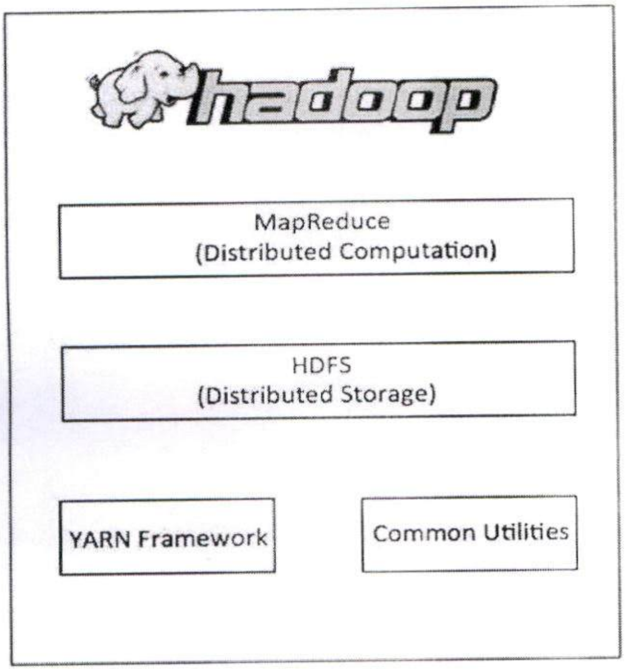
Hadoop Overview

Hadoop is an Apache open source framework written in java that allows distributed processing of large datasets across clusters of computers using simple programming models. The Hadoop framework application works in an environment that provides distributed *storage* and *computation* across clusters of computers. Hadoop is designed to scale up from single server to thousands of machines, each offering local computation and storage.

Hadoop Architecture

At its core, Hadoop has two major layers namely –

- Processing/Computation layer (MapReduce), and
- Storage layer (Hadoop Distributed File System).



MapReduce

MapReduce is a parallel programming model for writing distributed applications devised at Google for efficient processing of large amounts of data (multi-terabyte data-sets), on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner. The MapReduce program runs on Hadoop which is an Apache open-source framework.

Hadoop Distributed File System

The Hadoop Distributed File System (HDFS) is based on the Google File System (GFS) and provides a distributed file system that is designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant. It is highly fault-tolerant and is designed to be deployed on low-cost hardware. It provides high throughput access to application data and is suitable for applications having large datasets.

Apart from the above-mentioned two core components, Hadoop framework also includes the following two modules –

- **Hadoop Common** – These are Java libraries and utilities required by other Hadoop modules.
- **Hadoop YARN** – This is a framework for job scheduling and cluster resource management.

How Does Hadoop Work?

It is quite expensive to build bigger servers with heavy configurations that handle large scale processing, but as an alternative, you can tie together many commodity computers with single-CPU, as a single functional distributed system and practically, the clustered machines can read the dataset in parallel and provide a much higher throughput. Moreover, it is cheaper than one high-end server. So this is the first motivational factor behind using Hadoop that it runs across clustered and low-cost machines.

Hadoop runs code across a cluster of computers. This process includes the following core tasks that Hadoop performs –

- Data is initially divided into directories and files. Files are divided into uniform sized blocks of 128M and 64M (preferably 128M).
- These files are then distributed across various cluster nodes for further processing.
- HDFS, being on top of the local file system, supervises the processing.
- Blocks are replicated for handling hardware failure.
- Checking that the code was executed successfully.
- Performing the sort that takes place between the map and reduce stages.
- Sending the sorted data to a certain computer.
- Writing the debugging logs for each job.

Advantages of Hadoop

- Hadoop framework allows the user to quickly write and test distributed systems. It is efficient, and it automatic distributes the data and work across the machines and in turn, utilizes the underlying parallelism of the CPU cores.

- Hadoop does not rely on hardware to provide fault-tolerance and high availability (FTHA), rather Hadoop library itself has been designed to detect and handle failures at the application layer.
- Servers can be added or removed from the cluster dynamically and Hadoop continues to operate without interruption.
- Another big advantage of Hadoop is that apart from being open source, it is compatible on all the platforms since it is Java based.

Processing Data with Hadoop - Managing Resources and Applications with Hadoop YARN

Yarn divides the task on resource management and job scheduling/monitoring into separate daemons. There is one ResourceManager and per-application ApplicationMaster. An application can be either a job or a DAG of jobs.
The ResourceManger have two components – Scheduler and AppicationManager.

The **scheduler** is a pure scheduler i.e. it does not track the status of running application. It only allocates resources to various competing applications. Also, it does not restart the job after failure due to hardware or application failure. The scheduler allocates the resources based on an abstract notion of a container. A container is nothing but a fraction of resources like CPU, memory, disk, network etc.
Following are the tasks of ApplicationManager:-

- Accepts submission of jobs by client.
- Negotaites first container for specific ApplicationMaster.
- Restarts the container after application failure.

Below are the responsibilities of ApplicationMaster

- Negotiates containers from Scheduler
- Tracking container status and monitoring its progress.

Yarn supports the concept of Resource Reservation via Reservation System. In this, a user can fix a number of resources for execution of a particular job over time and temporal constraints. The Reservation System makes sure that the resources are available to the job until its completion. It also performs admission control for reservation.

Yarn can scale beyond a few thousand nodes via Yarn Federation. YARN Federation allows to wire multiple sub-cluster into the single massive cluster. We can use many independent clusters together for a single large job. It can be used to achieve a large scale system.

Let us summarize how **Hadoop** works step by step:

- Input data is broken into blocks of size **128 Mb** and then blocks are moved to different nodes.
- Once all the blocks of the data are stored on data-nodes, the user can process the data.
- Resource Manager then schedules the program (submitted by the user) on individual nodes.
- Once all the nodes process the data, the output is written back to HDFS.

Interacting with Hadoop Ecosystem

Hadoop Ecosystem Hadoop has an ecosystem that has evolved from its three core components processing, resource management, and storage. In this topic, you will learn the components of the Hadoop ecosystem and how they perform their roles during Big Data processing. The Hadoop ecosystem is continuously growing to meet the needs of Big Data. It comprises the following twelve components:

- HDFS(Hadoop Distributed file system)
- HBase
- Sqoop
- Flume
- Spark
- Hadoop MapReduce
- Pig
- Impala
- Hive
- Cloudera Search
- Oozie
- Hue.

Let us understand the role of each component of the Hadoop ecosystem.

Components of Hadoop Ecosystem

Let us start with the first component HDFS of Hadoop Ecosystem.

HDFS (HADOOP DISTRIBUTED FILE SYSTEM)

- HDFS is a storage layer for Hadoop.
- HDFS is suitable for distributed storage and processing, that is, while the data is being stored, it first gets distributed and then it is processed.
- HDFS provides Streaming access to file system data.
- HDFS provides file permission and authentication.
- HDFS uses a command line interface to interact with Hadoop.

So what stores data in HDFS? It is the HBase which stores data in HDFS.

HBase

- HBase is a NoSQL database or non-relational database .
- HBase is important and mainly used when you need random, real-time, read, or write access to your Big Data.
- It provides support to a high volume of data and high throughput.
- In an HBase, a table can have thousands of columns.

UNIT-III

INTRODUCTION TO MONGODB AND MAPREDUCE PROGRAMMING

MongoDB is a cross-platform, document-oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document.

Database

Database is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases.

Collection

Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.

Document

A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.

The following table shows the relationship of RDBMS terminology with MongoDB.

RDBMS	MongoDB
Database	Database
Table	Collection
Tuple/Row	Document
column	Field
Table Join	Embedded Documents

Primary Key	Primary Key (Default key _id provided by MongoDB itself)
Database Server and Client	
mysqld/Oracle	mongod
mysql/sqlplus	mongo

Sample Document

Following example shows the document structure of a blog site, which is simply a comma separated key value pair.

```
{
  _id: ObjectId(7df78ad8902c)
  title: 'MongoDB Overview',
  description: 'MongoDB is no sql database',
  by: 'tutorials point',
  url: 'http://www.tutorialspoint.com',
  tags: ['mongodb', 'database', 'NoSQL'],
  likes: 100,
  comments: [
    {
      user:'user1',
      message: 'My first comment',
      dateCreated: new Date(2011,1,20,2,15),
      like: 0
```

Why Use MongoDB?

- **Document Oriented Storage** – Data is stored in the form of JSON style documents.
- Index on any attribute
- Replication and high availability
- Auto-Sharding
- Rich queries
- Fast in-place updates
- Professional support by MongoDB

Where to Use MongoDB?

- Big Data
- Content Management and Delivery
- Mobile and Social Infrastructure
- User Data Management
- Data Hub

MongoDB supports many datatypes. Some of them are –

- **String** – This is the most commonly used datatype to store the data. String in MongoDB must be UTF-8 valid.
- **Integer** – This type is used to store a numerical value. Integer can be 32 bit or 64 bit depending upon your server.
- **Boolean** – This type is used to store a boolean (true/ false) value.
- **Double** – This type is used to store floating point values.
- **Min/ Max keys** – This type is used to compare a value against the lowest and highest BSON elements.
- **Arrays** – This type is used to store arrays or list or multiple values into one key.
- **Timestamp** – ctimestamp. This can be handy for recording when a document has been modified or added.
- **Object** – This datatype is used for embedded documents.
- **Null** – This type is used to store a Null value.
- **Symbol** – This datatype is used identically to a string; however, it's generally reserved for languages that use a specific symbol type.
- **Date** – This datatype is used to store the current date or time in UNIX time format. You can specify your own date time by creating object of Date and passing day, month, year into it.
- **Object ID** – This datatype is used to store the document's ID.

- **Binary data** – This datatype is used to store binary data.
- **Code** – This datatype is used to store JavaScript code into the document.
- **Regular expression** – This datatype is used to store regular expression.

The find() Method

To query data from MongoDB collection, you need to use MongoDB's **find()** method.

Syntax

The basic syntax of **find()** method is as follows –

>db.COLLECTION_NAME.find()

find() method will display all the documents in a non-structured way.

Example

Assume we have created a collection named mycol as –

```
> use sampleDB
switched to db sampleDB
> db.createCollection("mycol")
{ "ok" : 1 }
>
```

And inserted 3 documents in it using the insert() method as shown below –

```
> db.mycol.insert([
        {
                title: "MongoDB Overview",
                description: "MongoDB is no SQL database",
                by: "tutorials point",
                url: "http://www.tutorialspoint.com",
                tags: ["mongodb", "database", "NoSQL"],
                likes: 100
        },
        {
                title: "NoSQL Database",
                description: "NoSQL database doesn't have tables",
                by: "tutorials point",
                url: "http://www.tutorialspoint.com",
                tags: ["mongodb", "database", "NoSQL"],
                likes: 20,
                comments: [
                              {
                                        user:"user1",
                                        message: "My first comment",
                                        dateCreated: new Date(2013,11,10,2,35),
                                        like: 0
                              }
                    ]
          }
])
```

Following method retrieves all the documents in the collection –

```
> db.mycol.find()
{ "_id" : ObjectId("5dd4e2cc0821d3b44607534c"), "title" : "MongoDB Overview", "description"
: "MongoDB is no SQL database", "by" : "tutorials point", "url" : "http://www.tutorialspoint.com",
"tags" : [ "mongodb", "database", "NoSQL" ], "likes" : 100 }
{ "_id" : ObjectId("5dd4e2cc0821d3b44607534d"), "title" : "NoSQL Database", "description" :
"NoSQL database doesn't have tables", "by" : "tutorials point", "url" :
"http://www.tutorialspoint.com", "tags" : [ "mongodb", "database", "NoSQL" ], "likes" : 20,
"comments" : [ { "user" : "user1", "message" : "My first comment", "dateCreated" :
ISODate("2013-12-09T21:05:00Z"), "like" : 0 } ] }
>
```

The pretty() Method

To display the results in a formatted way, you can use **pretty()** method.

Syntax

>db.COLLECTION_NAME.find().pretty()

Example

Following example retrieves all the documents from the collection named mycol and arranges them in an easy-to-read format.

```
> db.mycol.find().pretty()
{
          "_id" : ObjectId("5dd4e2cc0821d3b44607534c"),
          "title" : "MongoDB Overview",
          "description" : "MongoDB is no SQL database",
          "by" : "tutorials point",
          "url" : "http://www.tutorialspoint.com",
          "tags" : [
                    "mongodb",
                    "database",
                    "NoSQL"
          ],
          "likes" : 100
}
{
        "_id" : ObjectId("5dd4c2cc0821d3b44607534d"),
        "title" : "NoSQL Database",
        "description" : "NoSQL database doesn't have tables",
        "by" : "tutorials point",
        "url" : "http://www.tutorialspoint.com",
        "tags" : [
                "mongodb",
                "database",
                "NoSQL"
        ],
        "likes" : 20,
        "comments" : [
                {
                        "user" : "user1",
                        "message" : "My first comment",
                        "dateCreated" : ISODate("2013-12-09T21:05:00Z"),
                        "like" : 0
                }
        ]
}
```

The findOne() method

Apart from the find() method, there is **findOne()** method, that returns only one document.

Syntax

```
>db.COLLECTIONNAME.findOne()
```

Example

Following example retrieves the document with title MongoDB Overview.

```
> db.mycol.findOne({title: "MongoDB Overview"})
{
        "_id" : ObjectId("5dd6542170fb13eec3963bf0"),
        "title" : "MongoDB Overview",
        "description" : "MongoDB is no SQL database",
        "by" : "tutorials point",
        "url" : "http://www.tutorialspoint.com",
        "tags" : [
                "mongodb",
                "database",
                "NoSQL"
        ],
        "likes" : 100
}
```

RDBMS Where Clause Equivalents in MongoDB

To query the document on the basis of some condition, you can use following operations.

Operation	**Syntax**	**Example**	**RDBMS Equivalent**
Equality	{<key>:{$eg;<value>}}	db.mycol.find({"by":"tutorials point"}).pretty()	where by = 'tutorials point'
Less Than	{<key>:{$lt:<value>}}	db.mycol.find({"likes":{$lt:50}}).pretty()	where likes < 50
Less Than Equals	{<key>:{$lte:<value>}}	db.mycol.find({"likes":{$lte:50}}).pretty()	where likes <= 50
Greater Than	{<key>:{$gt:<value>}}	db.mycol.find({"likes":{$gt:50}}).pretty()	where likes > 50
Greater Than Equals	{<key>:{$gte:<value>}}	db.mycol.find({"likes":{$gte:50}}).pretty()	where likes >= 50
Not Equals	{<key>:{$ne:<value>}}	db.mycol.find({"likes":{$ne:50}}).pretty()	where likes != 50
Values in an array	{<key>:{$in:[<value1>, <value2>,......<valueN>]}}	db.mycol.find({"name":{$in:["Raj", "Ram", "Raghu"]}}).pretty()	Where name matches any of the value in :["Raj", "Ram", "Raghu"]

Values not in an array	{<key>:{$nin:<value>}}	db.mycol.find({"name":{$nin:["Ramu", "Raghav"]}}).pretty()	Where name values is not in the array :["Ramu", "Raghav"] or, doesn't exist at all

AND in MongoDB

Syntax

To query documents based on the AND condition, you need to use $and keyword. Following is the basic syntax of AND –

>db.mycol.find({ $and: [{<key1>:<value1>}, { <key2>:<value2>}] })

Example

Following example will show all the tutorials written by 'tutorials point' and whose title is 'MongoDB Overview'.

```
> db.mycol.find({$and:[{"by":"tutorials point"},{"title": "MongoDB Overview"}]}).pretty()
{
        "_id" : ObjectId("5dd4e2cc0821d3b44607534c"),
        "title" : "MongoDB Overview",
        "description" : "MongoDB is no SQL database",
        "by" : "tutorials point",
        "url" : "http://www.tutorialspoint.com",
        "tags" : [
                "mongodb",
                "database",
                "NoSQL"
        ],
        "likes" : 100
}
>
```

For the above given example, equivalent where clause will be ' **where by = 'tutorials point' AND title = 'MongoDB Overview'** '. You can pass any number of key, value pairs in find clause.

OR in MongoDB

Syntax

MapReduce:

MapReduce addresses the challenges of distributed programming by providing an abstraction that isolates the developer from system-level details (e.g., locking of data structures, data starvation issues in the processing pipeline, etc.). The programming model specifies simple and well-defined interfaces between a small number of components, and therefore is easy for the programmer to reason about. MapReduce maintains a separation of what computations are to be performed and how those computations are actually carried out on a cluster of machines. The first is under the control of the programmer, while the second is exclusively the responsibility of the execution framework or "runtime". The advantage is that the execution framework only needs to be designed once and verified for correctness—thereafter, as long as the developer expresses computations in the programming model, code is guaranteed to behave as expected. The upshot is that the developer is freed from having to worry about system-level details (e.g., no more debugging race conditions and addressing lock contention) and can instead focus on algorithm or application design.

ich often has multiple cores). Why is MapReduce important? In practical terms, it provides a very effective tool for tackling large-data problems. But beyond that, MapReduce is important in how it has changed the way we organize computations at a massive scale. MapReduce represents the first widely-adopted step away from the von Neumann model that has served as the foundation of computer science over the last half plus century. Valiant called this a bridging model [148], a conceptual bridge between the physical implementation of a machine and the software that is to be executed on that machine. Until recently, the von Neumann model has served us well: Hardware designers focused on efficient implementations of the von Neumann model and didn't have to think much about the actual software that would run on the machines. Similarly, the software industry developed software targeted at the model without worrying about the hardware details. The result was extraordinary growth: chip designers churned out successive generations of increasingly powerful processors, and software engineers were able to develop applications in high-level languages that exploited those processors.

MapReduce can be viewed as the first breakthrough in the quest for new abstractions that allow us to organize computations, not over individual machines, but over entire clusters. As Barroso puts it, the datacenter is the computer. MapReduce is certainly not the first model of parallel computation that has been proposed. The most prevalent model in theoretical computer science, which dates back several decades, is the PRAM. MAPPERS AND REDUCERS Key-value pairs form the basic data structure in MapReduce. Keys and values may be primitives such as integers, floating point values, strings, and raw bytes, or they may be arbitrarily complex structures (lists, tuples, associative arrays, etc.). Programmers typically need to define their own custom data types, although a number of libraries such as Protocol Buffers,5 Thrift,6 and Avro7 simplify the task. Part of the design of MapReduce algorithms involves imposing the key-value structure on arbitrary datasets. For a collection of web pages, keys may be URLs and values may be the actual HTML content. For a graph, keys may represent node ids and values may contain the adjacency lists of those nodes (see Chapter 5 for more details). In some algorithms, input keys are not particularly meaningful and are simply ignored during processing, while in other cases input keys are used to uniquely identify a datum (such as a record id). In Chapter 3, we discuss the role of complex keys and values in the design of various algorithms. In MapReduce, the programmer defines a mapper and a reducer with the following signatures: map: (k1, v1) → [(k2, v2)] reduce: (k2, [v2]) → [(k3, v3)] The convention [. . .] is used throughout this book to denote a list. The input to a MapReduce job starts as data stored on the underlying distributed file system (see Section 2.5). The mapper is applied to every input key-value pair (split across an arbitrary number of files) to generate an arbitrary number of intermediate key-value pairs. The reducer is applied to all values associated with the same intermediate key to generate output key-value pairs.8 Implicit between the map and reduce phases is a distributed "group by" operation on intermediate keys. Intermediate data arrive at each reducer in order, sorted by the key. However, no ordering relationship is guaranteed for keys across different reducers. Output key-value pairs from each reducer are written persistently back onto the distributed file system (whereas intermediate key-value pairs are transient and not preserved). The output ends up in r files on the distributed file system, where r is the number of reducers. For the most part, there is no need to consolidate reducer output, since the r files often serve as input to yet another MapReduce job. Figure 2.2 illustrates this two-stage processing structure. A simple word count algorithm in MapReduce is shown in Figure 2.3. This algorithm counts the number of occurrences of every word in a text collection, which may be the first step in, for example, building a unigram language model (i.e., probability

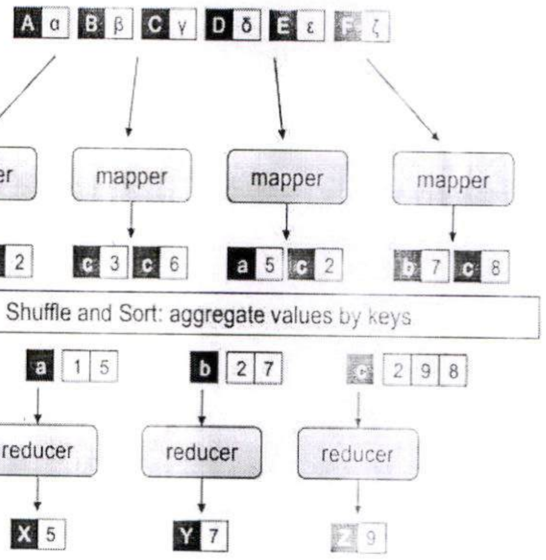


Figure 2.2: Simplified view of MapReduce. Mappers are applied to all input key-value pairs, which generate an arbitrary number of intermediate key-value pairs. Reducers are applied to all values associated with the same key. Between the map and reduce phases lies a barrier that involves a large distributed sort and group by.

distribution over words in a collection). Input key-values pairs take the form of (docid, doc) pairs stored on the distributed file system, where the former is a unique identifier for the document, and the latter is the text of the document itself. The mapper takes an input key-value pair, tokenizes the document, and emits an intermediate key-value pair for every word: the word itself serves as the key, and the integer one serves as the value (denoting that we've seen the word once). The MapReduce execution framework guarantees that all values associated with the same key are brought together in the reducer. Therefore, in our word count algorithm, we simply need to sum up all counts (ones) associated with each word. The reducer does exactly this, and emits final keyvalue pairs with the word as the key, and the count as the value. Final output is written to the distributed file system, one file per reducer. Words within each file will be sorted by alphabetical order, and each file will contain roughly the same number of words. The partitioner, which we discuss later in Section 2.4, controls the assignment of words to reducers. The output can be examined by the programmer or used as input to another MapReduce program.

There are some differences between the Hadoop implementation of MapReduce and Google's implementation.9 In Hadoop, the reducer is presented with a key and an iterator over all values associated with the particular key. The values are arbitrarily ordered. Google's implementation allows the programmer to specify a secondary sort key for ordering the values (if desired)—in which case values associated with each key would be presented to the developer's reduce code in sorted order. Later in Section 3.4 we discuss how to overcome this limitation in Hadoop to perform secondary sorting. Another difference: in Google's implementation the programmer is not allowed to change the key in the reducer. That is, the reducer output key must be exactly the same as the reducer input key. In Hadoop, there is no such restriction, and the reducer can emit an arbitrary number of output key-value pairs (with different keys).

To provide a bit more implementation detail: pseudo-code provided in this book roughly mirrors how MapReduce programs are written in Hadoop. Mappers and reducers are objects that implement the Map and Reduce methods, respectively. In Hadoop, a mapper object is initialized for each map task (associated with a particular sequence of key-value pairs called an input split) and the Map method is called on each key-value pair by the execution framework. In configuring a MapReduce job, the programmer provides a hint on the number of map tasks to run, but the execution framework (see next section) makes the final determination based on the physical layout of the data (more details in Section 2.5 and Section 2.6). The situation is similar for the reduce phase: a reducer object is initialized for each reduce task, and the Reduce method is called once per intermediate key. In contrast with the number of map tasks, the programmer can precisely specify the number of reduce tasks. We will return to discuss the details of Hadoop job execution in Section 2.6, which is dependent on an understanding of the distributed file system (covered in Section 2.5). To reiterate: although the presentation of algorithms in this book closely mirrors the way they would be implemented in Hadoop, our focus is on algorithm design and conceptual understanding—not actual Hadoop programming. For that, we would recommend Tom White's book [154]. What are the restrictions on mappers and reducers? Mappers and reducers can express arbitrary computations over their inputs. However, one must generally be careful about use of external resources since multiple mappers or reducers may be contending for those resources. For example, it may be unwise for a mapper to query an external SQL database, since that would introduce a scalability bottleneck on the number of map tasks that could be run in parallel (since they might all be simultaneously querying the database).10 In general, mappers can emit an arbitrary number of intermediate key-value pairs, and they need not be of the same type as the input key-value pairs. Similarly, reducers can emit an arbitrary number of final key-value pairs, and they can differ in type from the intermediate key-value pairs. Although not permitted in functional programming, mappers and reducers can have side effects. This is a powerful and useful feature: for example, preserving state across multiple inputs is central to the design of many MapReduce algorithms (see Chapter 3). Such algorithms can be understood as having side effects that only change state that is internal to the mapper or reducer. While the correctness of such algorithms may be more difficult to guarantee (since the function's behavior depends not only on the current input but on previous inputs), most potential synchronization problems are avoided since internal state is private only to individual mappers and reducers. In other cases (see Section 4.4 and Section 6.5), it may be useful for mappers or reducers to have external side effects, such as writing files to the distributed file system. Since many mappers and reducers are run in parallel, and the distributed file system is a shared global resource, special care must be taken to ensure that such operations avoid synchronization conflicts. One strategy is to write a temporary file that is renamed upon successful completion of the mapper or reducer .

In addition to the "canonical" MapReduce processing flow, other variations are also possible. MapReduce programs can contain no reducers, in which case mapper output is directly written to disk (one file per mapper). For embarrassingly parallel problems, e.g., parse a large text collection or independently analyze a large number of images, this would be a common pattern. The converse—a MapReduce program with no mappers—is not possible, although in some cases it is useful for the mapper to implement the identity function and simply pass input key-value pairs to the reducers. This has the effect of sorting and regrouping the input for reduce-side processing. Similarly, in some cases it is useful for the reducer to implement the identity function, in which case the program simply sorts and groups mapper output. Finally, running identity mappers and reducers has the effect of regrouping and resorting the input data (which is sometimes useful).

Although in the most common case, input to a MapReduce job comes from data stored on the distributed file system and output is written back to the distributed file system, any other system that satisfies the proper abstractions can serve as a data source or sink. With Google's MapReduce implementation, BigTable [34], a sparse, distributed, persistent multidimensional sorted map, is frequently used as a source of input and as a store of MapReduce output. HBase is an open-source BigTable clone and has similar capabilities. Also, Hadoop has been integrated with existing MPP (massively parallel processing) relational databases, which allows a programmer to write MapReduce jobs over database rows and dump output into a new database table. Finally, in some

UNIT-IV

INTRODUCTION TO HIVE AND PIG

The term 'Big Data' is used for collections of large datasets that include huge volume, high velocity, and a variety of data that is increasing day by day. Using traditional data management systems, it is difficult to process Big Data. Therefore, the Apache Software Foundation introduced a framework called Hadoop to solve Big Data management and processing challenges.

Hadoop

Hadoop is an open-source framework to store and process Big Data in a distributed environment. It contains two modules, one is MapReduce and another is Hadoop Distributed File System (HDFS).

- **MapReduce:** It is a parallel programming model for processing large amounts of structured, semi-structured, and unstructured data on large clusters of commodity hardware.
- **HDFS:**Hadoop Distributed File System is a part of Hadoop framework, used to store and process the datasets. It provides a fault-tolerant file system to run on commodity hardware.

The Hadoop ecosystem contains different sub-projects (tools) such as Sqoop, Pig, and Hive that are used to help Hadoop modules.

- **Sqoop:** It is used to import and export data to and from between HDFS and RDBMS.
- **Pig:** It is a procedural language platform used to develop a script for MapReduce operations.
- **Hive:** It is a platform used to develop SQL type scripts to do MapReduce operations.

Note: There are various ways to execute MapReduce operations:

- The traditional approach using Java MapReduce program for structured, semi-structured, and unstructured data.
- The scripting approach for MapReduce to process structured and semi structured data using Pig.
- The Hive Query Language (HiveQL or HQL) for MapReduce to process structured data using Hive.

What is Hive

Hive is a data warehouse infrastructure tool to process structured data in Hadoop. It resides on top of Hadoop to summarize Big Data, and makes querying and analyzing easy.

Initially Hive was developed by Facebook, later the Apache Software Foundation took it up and developed it further as an open source under the name Apache Hive. It is used by different companies. For example, Amazon uses it in Amazon Elastic MapReduce.

Hive is not

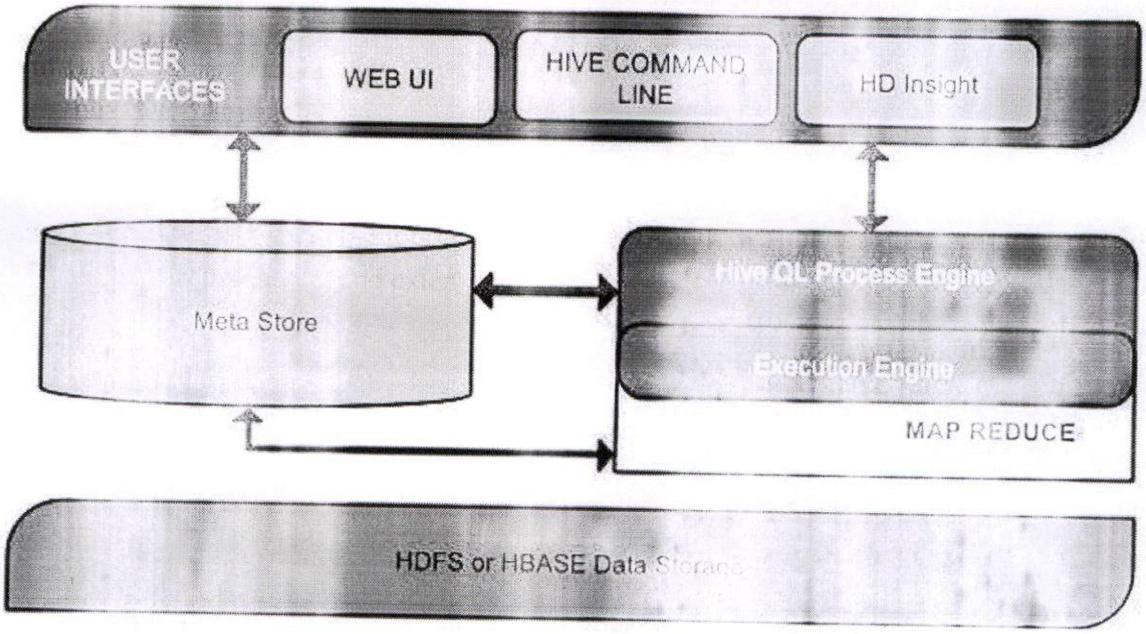
- A relational database
- A design for OnLine Transaction Processing (OLTP)
- A language for real-time queries and row-level updates

Features of Hive

- It stores schema in a database and processed data into HDFS.
- It is designed for OLAP.
- It provides SQL type language for querying called HiveQL or HQL.
- It is familiar, fast, scalable, and extensible.

Architecture of Hive

The following component diagram depicts the architecture of Hive:

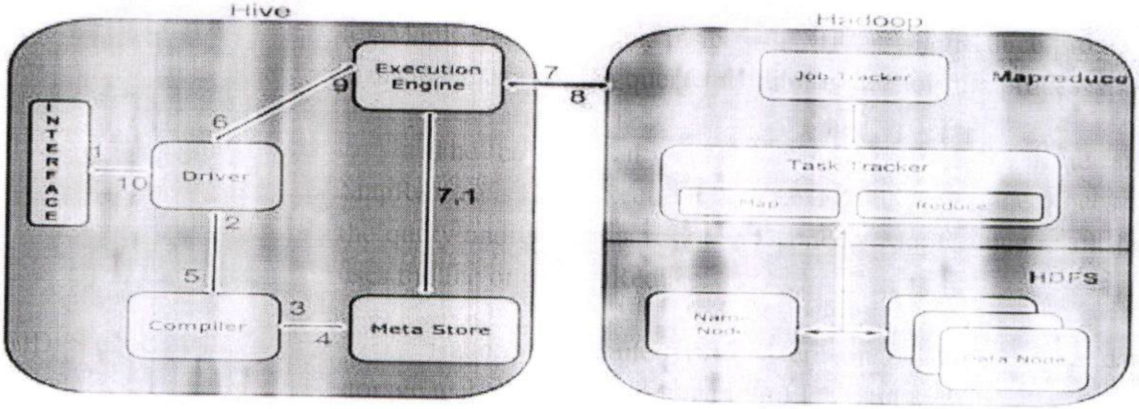


This component diagram contains different units. The following table describes each unit:

Unit Name	Operation
User Interface	Hive is a data warehouse infrastructure software that can create interaction between user and HDFS. The user interfaces that Hive supports are Hive Web UI, Hive command line, and Hive HD Insight (In Windows server).
Meta Store	Hive chooses respective database servers to store the schema or Metadata of tables, databases, columns in a table, their data types, and HDFS mapping.
HiveQL Process Engine	HiveQL is similar to SQL for querying on schema info on the Metastore. It is one of the replacements of traditional approach for MapReduce program. Instead of writing MapReduce program in Java, we can write a query for MapReduce job and process it.
Execution Engine	The conjunction part of HiveQL process Engine and MapReduce is Hive Execution Engine. Execution engine processes the query and generates results as same as MapReduce results. It uses the flavor of MapReduce.
HDFS or HBASE	Hadoop distributed file system or HBASE are the data storage techniques to store data into file system.

Working of Hive

The following diagram depicts the workflow between Hive and Hadoop.



The following table defines how Hive interacts with Hadoop framework:

Step No.	Operation
1	**Execute Query** The Hive interface such as Command Line or Web UI sends query to Driver (any database driver such as JDBC, ODBC, etc.) to execute.
2	**Get Plan** The driver takes the help of query compiler that parses the query to check the syntax and query plan or the requirement of query.
3	**Get Metadata** The compiler sends metadata request to Metastore (any database).
4	**Send Metadata** Metastore sends metadata as a response to the compiler.
5	**Send Plan** The compiler checks the requirement and resends the plan to the driver. Up to here, the parsing and compiling of a query is complete.
6	**Execute Plan** The driver sends the execute plan to the execution engine.
7	**Execute Job** Internally, the process of execution job is a MapReduce job. The execution engine sends the job to JobTracker, which is in Name node and it assigns this job to TaskTracker, which is in Data node. Here, the query executes MapReduce job.
7.1	**Metadata Ops**

	Meanwhile in execution, the execution engine can execute metadata operations with Metastore.
8	**Fetch Result** The execution engine receives the results from Data nodes.
9	**Send Results** The execution engine sends those resultant values to the driver.
10	**Send Results** The driver sends the results to Hive Interfaces.

File Formats in Hive

- File Format specifies how records are encoded in files
- Record Format implies how a stream of bytes for a given record are encoded
- The default file format is **TEXTFILE** – each record is a line in the file
- Hive uses different control characters as delimeters in textfiles
  - ^A (octal 001) , ^B(octal 002), ^C(octal 003), \n
- The term field is used when overriding the default delimiter
  - **FIELDS TERMINATED BY '\001'**
- Supports text files – csv, tsv
- TextFile can contain JSON or XML documents.

ommonly used File Formats –

1. **TextFile format**
   - Suitable for sharing data with other tools
   - Can be viewed/edited manually
2. **SequenceFile**
   - Flat files that stores binary key ,value pair
   - SequenceFile offers a Reader ,Writer, and Sorter classes for reading ,writing, and sorting respectively
   - Supports – Uncompressed, Record compressed (only value is compressed) and Block compressed (both key,value compressed) formats
3. **RCFile**
   - RCFile stores columns of a table in a record columnar way
4. **ORC**
5. **AVRO**

Hive Commands

Hive supports Data definition Language(DDL), Data Manipulation Language(DML) and User defined functions.

Hive DDL Commands

create database

drop database

create table

drop table

alter table

create index

create view

Hive DML Commands

Select

Where

Group By

Order By

Load Data

Join:

- Inner Join
- Left Outer Join
- Right Outer Join
- Full Outer Join

Hive DDL Commands

Create Database Statement

A database in Hive is a namespace or a collection of tables.

```
hive> CREATE SCHEMA userdb;
hive> SHOW DATABASES;
```

Drop database

```
ive> DROP DATABASE IF EXISTS userdb;
```

Creating Hive Tables

Create a table called Sonoo with two columns, the first being an integer and the other a string.

```
hive> CREATE TABLE Sonoo(foo INT, bar STRING);
```

Create a table called HIVE_TABLE with two columns and a partition column called ds. The partition column is a virtual column. It is not part of the data itself but is derived from the partition that a particular dataset is loaded into.By default, tables are assumed to be of text input format and the delimiters are assumed to be ^A(ctrl-a).

```
hive> CREATE TABLE HIVE_TABLE (foo INT, bar STRING) PARTITIONED BY (ds STRING);
```

Browse the table

```
hive>  Show tables;
```

Altering and Dropping Tables

```
hive> ALTER TABLE Sonoo RENAME TO Kafka;
hive> ALTER TABLE Kafka ADD COLUMNS (col INT);
hive> ALTER TABLE HIVE_TABLE ADD COLUMNS (col1 INT COMMENT 'a comment');
hive> ALTER TABLE HIVE_TABLE REPLACE COLUMNS (col2 INT, weight STRING, baz INT COMMENT 'baz replaces new_col1');
```

Hive DML Commands

To understand the Hive DML commands, let's see the employee and employee_department table first.

Employee

EMP ID	Emp Name	Address
1	Rose	US
2	Fred	US
3	Jess	In
4	Frey	Th

Employee Department

Emp ID	Department
1	IT
2	IT
3	Eng
4	Admin

LOAD DATA

1. hive> LOAD DATA LOCAL INPATH './usr/Desktop/kv1.txt' OVERWRITE INTO TABLE Employee;

SELECTS and FILTERS

1. hive> SELECT E.EMP_ID FROM Employee E WHERE E.Address='US';

GROUP BY

1. hive> hive> SELECT E.EMP_ID FROM Employee E GROUP BY E.Addresss;

Adding a Partition

We can add partitions to a table by altering the table. Let us assume we have a table called **employee** with fields such as Id, Name, Salary, Designation, Dept, and yoj.

Syntax:

```
ALTER TABLE table_name ADD [IF NOT EXISTS] PARTITION partition_spec
[LOCATION 'location1'] partition_spec [LOCATION 'location2'] ...;

partition_spec:
: (p_column = p_col_value, p_column = p_col_value, ...)
```

The following query is used to add a partition to the employee table.

```
hive> ALTER TABLE employee
> ADD PARTITION (year=’2012’)
> location '/2012/part2012';
```

Renaming a Partition

The syntax of this command is as follows.

```
ALTER TABLE table_name PARTITION partition_spec RENAME TO PARTITION
partition_spec;
```

The following query is used to rename a partition:

```
hive> ALTER TABLE employee PARTITION (year=’1203’)
  > RENAME TO PARTITION (Yoj=’1203’);
```

Dropping a Partition

The following syntax is used to drop a partition:

```
ALTER TABLE table_name DROP [IF EXISTS] PARTITION partition_spec, PARTITION
partition_spec,...;
```

The following query is used to drop a partition:

```
hive> ALTER TABLE employee DROP [IF EXISTS]
  > PARTITION (year=’1203’);
```

Hive Query Language

The Hive Query Language (HiveQL) is a query language for Hive to process and analyze structured data in a Metastore. This chapter explains how to use the SELECT statement with WHERE clause.

SELECT statement is used to retrieve the data from a table. WHERE clause works similar to a condition. It filters the data using the condition and gives you a finite result. The built-in operators and functions generate an expression, which fulfils the condition.

Syntax

Given below is the syntax of the SELECT query:

```
SELECT [ALL | DISTINCT] select_expr, select_expr, ...
FROM table_reference
[WHERE where_condition]
[GROUP BY col_list]
[HAVING having_condition]
[CLUSTER BY col_list | [DISTRIBUTE BY col_list] [SORT BY col_list]]
[LIMIT number];
```

Example

Let us take an example for SELECT…WHERE clause. Assume we have the employee table as given below, with fields named Id, Name, Salary, Designation, and Dept. Generate a query to retrieve the employee details who earn a salary of more than Rs 30000.

```
+------+--------------+-------------+-------------------+--------+
| ID | Name | Salary | Designation | Dept |
+------+--------------+-------------+-------------------+--------+
|1201 | Gopal | 45000 | Technical manager | TP |
|1202 | Manisha | 45000 | Proofreader | PR |
|1203 | Masthanvali | 40000 | Technical writer | TP |
|1204 | Krian | 40000 | Hr Admin | HR |
|1205 | Kranthi | 30000 | Op Admin | Admin |
+------+--------------+-------------+-------------------+--------+
```

The following query retrieves the employee details using the above scenario:

```
hive> SELECT * FROM employee WHERE salary>30000;
```

On successful execution of the query, you get to see the following response:

```
+------+--------------+-------------+-------------------+--------+
| ID | Name | Salary | Designation | Dept |
+------+--------------+-------------+-------------------+--------+
|1201 | Gopal | 45000 | Technical manager | TP |
|1202 | Manisha | 45000 | Proofreader | PR |
|1203 | Masthanvali | 40000 | Technical writer | TP |
|1204 | Krian | 40000 | Hr Admin | HR |
+------+--------------+-------------+-------------------+--------+
```

JDBC Program

The JDBC program to apply where clause for the given example is as follows.

```
import java.sql.SQLException;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;
import java.sql.DriverManager;

public class HiveQLWhere {
  private static String driverName = "org.apache.hadoop.hive.jdbc.HiveDriver";

  public static void main(String[] args) throws SQLException {

    // Register driver and create driver instance
    Class.forName(driverName);

    // get connection
    Connection con = DriverManager.getConnection("jdbc:hive://localhost:10000/userdb", "",
"");

    // create statement
    Statement stmt = con.createStatement();

    // execute statement
```

```
      Resultset res = stmt.executeQuery("SELECT * FROM employee WHERE salary>30000;");

      System.out.println("Result:");
      System.out.println(" ID \t Name \t Salary \t Designation \t Dept ");

      while (res.next()) {
         System.out.println(res.getInt(1) + " " + res.getString(2) + " " + res.getDouble(3) + " " +
res.getString(4) + " " + res.getString(5));
      }
      con.close();
   }
}
```

Save the program in a file named HiveQLWhere.java. Use the following commands to compile and execute this program.

```
$ javac HiveQLWhere.java
$ java HiveQLWhere
```

Output:

```
ID      Name         Salary     Designation          Dept
1201    Gopal        45000      Technical manager    TP
1202    Manisha      45000      Proofreader          PR
1203    Masthanvali  40000      Technical writer     TP
1204    Krian        40000      Hr Admin             HR
```

The ORDER BY clause is used to retrieve the details based on one column and sort the result set by ascending or descending order.

Syntax

Given below is the syntax of the ORDER BY clause:

```
SELECT [ALL | DISTINCT] select_expr, select_expr, ...
FROM table_reference
[WHERE where_condition]
[GROUP BY col_list]
[HAVING having_condition]
[ORDER BY col_list]]
[LIMIT number];
```

Pig Data Types

Apache Pig supports many data types. A list of Apache Pig Data Types with description and examples are given below.

Type	Description	Example
Int	Signed 32 bit integer	2
Long	Signed 64 bit integer	15L or 15l
Float	32 bit floating point	2.5f or 2.5F
Double	32 bit floating point	1.5 or 1.5e2 or 1.5E2
charArray	Character array	hello javatpoint
byteArray	BLOB(Byte array)	
tuple	Ordered set of fields	(12,43)
bag	Collection f tuples	{(12,43),(54,28)}
map	collection of tuples	[open#apache]

Apache Pig Execution Modes

You can run Apache Pig in two modes, namely, **Local Mode** and **HDFS mode**.

Local Mode

In this mode, all the files are installed and run from your local host and local file system. There is no need of Hadoop or HDFS. This mode is generally used for testing purpose.

MapReduce Mode

MapReduce mode is where we load or process the data that exists in the Hadoop File System (HDFS) using Apache Pig. In this mode, whenever we execute the Pig Latin statements to process the data, a MapReduce job is invoked in the back-end to perform a particular operation on the data that exists in the HDFS.

Apache Pig Execution Mechanisms

Apache Pig scripts can be executed in three ways, namely, interactive mode, batch mode, and embedded mode.

- **Interactive Mode** (Grunt shell) – You can run Apache Pig in interactive mode using the Grunt shell. In this shell, you can enter the Pig Latin statements and get the output (using Dump operator).
- **Batch Mode** (Script) – You can run Apache Pig in Batch mode by writing the Pig Latin script in a single file with **.pig** extension.
- **Embedded Mode** (UDF) – Apache Pig provides the provision of defining our own functions (**U**ser **D**efined **F**unctions) in programming languages such as Java, and using them in our script.
- Given below in the table are some frequently used Pig Commands.

Command	Function
load	Reads data from the system
Store	Writes data to file system
foreach	Applies expressions to each record and outputs one or more records
filter	Applies predicate and removes records that do not return true

Group/cogroup	Collects records with the same key from one or more inputs
join	Joins two or more inputs based on a key
order	Sorts records based on a key
distinct	Removes duplicate records
union	Merges data sets
split	Splits data into two or more sets based on filter conditions
stream	Sends all records through a user-provided binary
dump	Writes output to stdout

limit	Limits the number of records

Complex Types

Type	Description
tuple	It defines an ordered set of fields. Example - (15,12)
bag	It defines a collection of tuples. Example - {(15,12), (12,15)}
map	It defines a set of key-value pairs. Example - [open#apache]

Pig Latin – Relational Operations

The following table describes the relational operators of Pig Latin.

Operator	Description
Loading and Storing	
LOAD	To Load the data from the file system (local/HDFS) into a relation.
STORE	To save a relation to the file system (local/HDFS).
Filtering	

FILTER	To remove unwanted rows from a relation.
DISTINCT	To remove duplicate rows from a relation.
FOREACH, GENERATE	To generate data transformations based on columns of data.
STREAM	To transform a relation using an external program.
Grouping and Joining	
JOIN	To join two or more relations.
COGROUP	To group the data in two or more relations.
GROUP	To group the data in a single relation.
CROSS	To create the cross product of two or more relations.
Sorting	
ORDER	To arrange a relation in a sorted order based on one or more fields (ascending or descending).
LIMIT	To get a limited number of tuples from a relation.
Combining and Splitting	
UNION	To combine two or more relations into a single relation.
SPLIT	To split a single relation into two or more relations.

Diagnostic Operators	
DUMP	To print the contents of a relation on the console.
DESCRIBE	To describe the schema of a relation.
EXPLAIN	To view the logical, physical, or MapReduce execution plans to compute a relation.
ILLUSTRATE	To view the step-by-step execution of a series of statements.

Eval Functions

Given below is the list of **eval** functions provided by Apache Pig.

S.N.	Function & Description
1	AVG() To compute the average of the numerical values within a bag.
2	BagToString() To concatenate the elements of a bag into a string. While concatenating, we can place a delimiter between these values (optional).
3	CONCAT() To concatenate two or more expressions of same type.
4	COUNT() To get the number of elements in a bag, while counting the number of tuples in a bag.
5	COUNT_STAR()

	It is similar to the **COUNT()** function. It is used to get the number of elements in a bag.
6	DIFF() To compare two bags (fields) in a tuple.
7	IsEmpty() To check if a bag or map is empty.
8	MAX() To calculate the highest value for a column (numeric values or chararrays) in a single-column bag.
9	MIN() To get the minimum (lowest) value (numeric or chararray) for a certain column in a single-column bag.
10	PluckTuple() Using the Pig Latin **PluckTuple()** function, we can define a string Prefix and filter the columns in a relation that begin with the given prefix.
11	SIZE() To compute the number of elements based on any Pig data type.
12	SUBTRACT() To subtract two bags. It takes two bags as inputs and returns a bag which contains the tuples of the first bag that are not in the second bag.
13	SUM() To get the total of the numeric values of a column in a single-column bag.
14	TOKENIZE() To split a string (which contains a group of words) in a single tuple and return a bag which contains the output of the split operation.

Apache Pig provides extensive support for **U**ser **D**efined **F**unctions (UDF's). Using these UDF's, we can define our own functions and use them. The UDF support is provided in six programming languages, namely, Java, Jython, Python, JavaScript, Ruby and Groovy.

For writing UDF's, complete support is provided in Java and limited support is provided in all the remaining languages. Using Java, you can write UDF's involving all parts of the processing like data load/store, column transformation, and aggregation. Since Apache Pig has been written in Java, the UDF's written using Java language work efficiently compared to other languages.

In Apache Pig, we also have a Java repository for UDF's named **Piggybank**. Using Piggybank, we can access Java UDF's written by other users, and contribute our own UDF's.

Types of UDF's in Java

While writing UDF's using Java, we can create and use the following three types of functions –

- **Filter Functions** – The filter functions are used as conditions in filter statements. These functions accept a Pig value as input and return a Boolean value.
- **Eval Functions** – The Eval functions are used in FOREACH-GENERATE statements. These functions accept a Pig value as input and return a Pig result.
- **Algebraic Functions** – The Algebraic functions act on inner bags in a FOREACHGENERATE statement. These functions are used to perform full MapReduce operations on an inner bag.

Writing UDF's using Java

To write a UDF using Java, we have to integrate the jar file **Pig-0.15.0.jar**. In this section, we discuss how to write a sample UDF using Eclipse. Before proceeding further, make sure you have installed Eclipse and Maven in your system.

Follow the steps given below to write a UDF function –

- Open Eclipse and create a new project (say **myproject**).
- Convert the newly created project into a Maven project.
- Copy the following content in the pom.xml. This file contains the Maven dependencies for Apache Pig and Hadoop-core jar files.

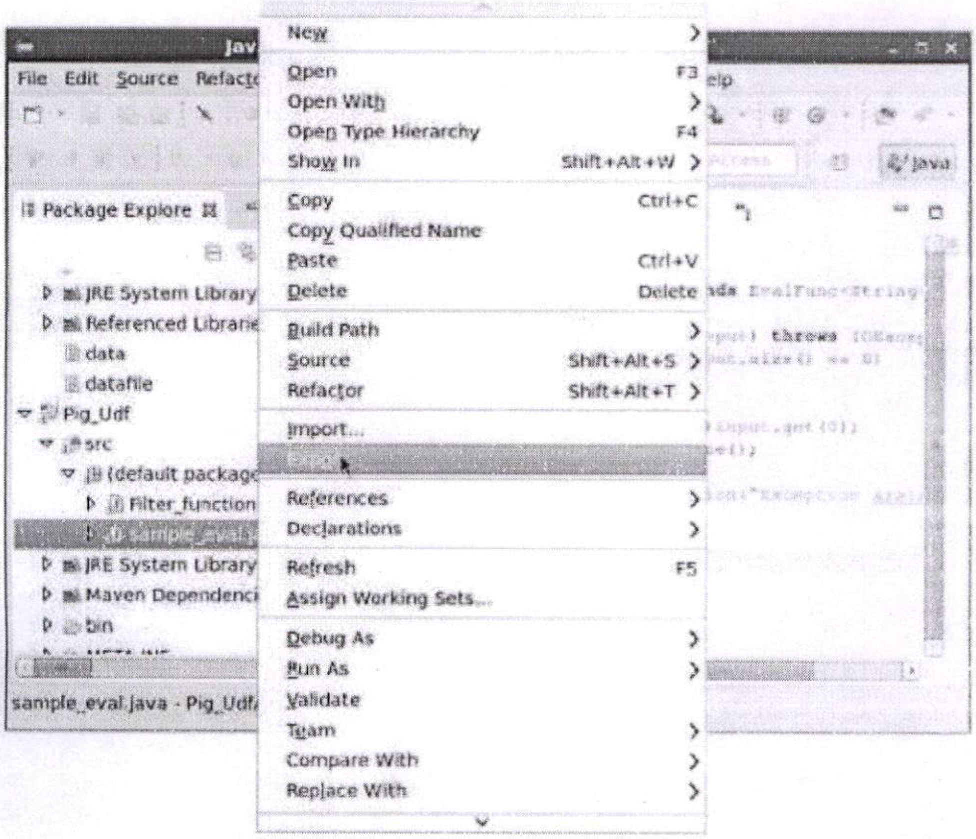
```
<project xmlns = "http://maven.apache.org/POM/4.0.0"
  xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation          =          "http://maven.apache.org/POM/4.0.0http://maven.apache
.org/xsd/maven-4.0.0.xsd">

  <modelVersion>4.0.0</modelVersion>
  <groupId>Pig_Udf</groupId>
  <artifactId>Pig_Udf</artifactId>
  <version>0.0.1-SNAPSHOT</version>

  <build>
    <sourceDirectory>src</sourceDirectory>
    <plugins>
      <plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.3</version>
        <configuration>
          <source>1.7</source>
          <target>1.7</target>
        </configuration>
      </plugin>
    </plugins>
  </build>

  <dependencies>

    <dependency>
      <groupId>org.apache.pig</groupId>
      <artifactId>pig</artifactId>
      <version>0.15.0</version>
    </dependency>

    <dependency>
      <groupId>org.apache.hadoop</groupId>
      <artifactId>hadoop-core</artifactId>
      <version>0.20.2</version>
    </dependency>

  </dependencies>

</project>
```

- Save the file and refresh it. In the **Maven Dependencies** section, you can find the downloaded jar files.
- Create a new class file with name **Sample_Eval** and copy the following content in it.

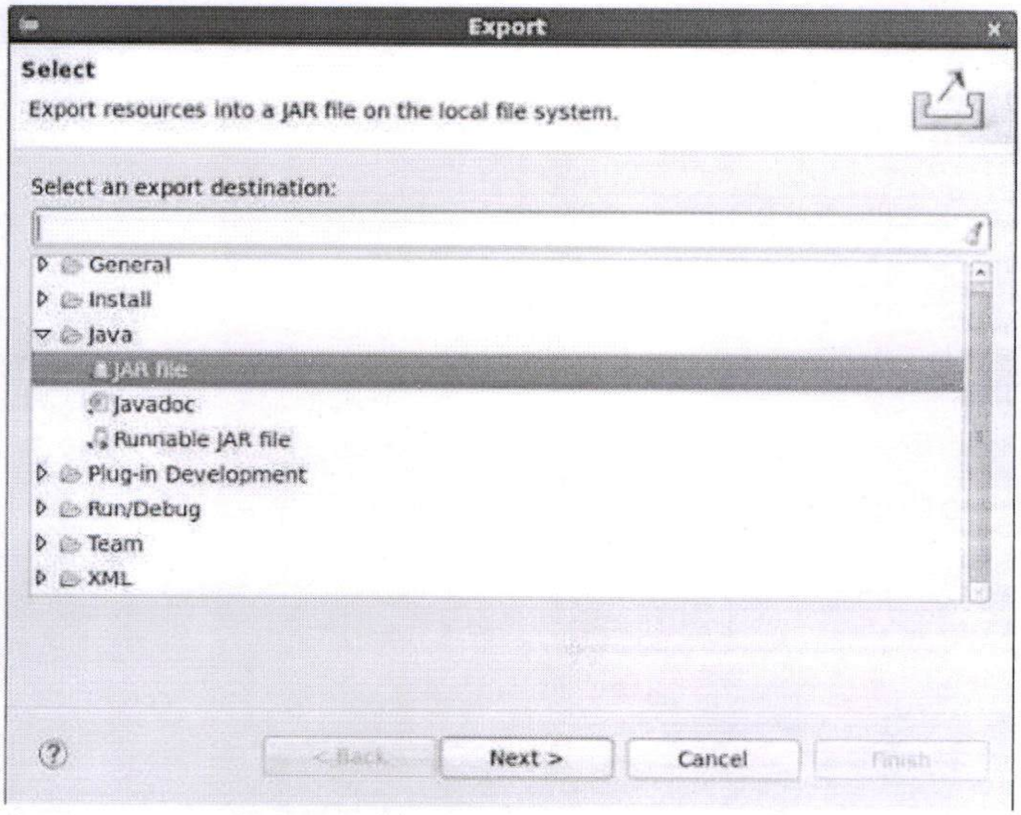
```
import java.io.IOException;
import org.apache.pig.EvalFunc;
import org.apache.pig.data.Tuple;

import java.io.IOException;
import org.apache.pig.EvalFunc;
import org.apache.pig.data.Tuple;

public class Sample_Eval extends EvalFunc<String>{

  public String exec(Tuple input) throws IOException {
    if (input == null || input.size() == 0)
    return null;
    String str = (String)input.get(0);
    return str.toUpperCase();
    }
}
```

While writing UDF's, it is mandatory to inherit the EvalFunc class and provide implementation to **exec()** function. Within this function, the code required for the UDF is written. In the above example, we have return the code to convert the contents of the given column to uppercase.

- After compiling the class without errors, right-click on the Sample_Eval.java file. It gives you a menu. Select **export** as shown in the following screenshot.



- On clicking **export,** you will get the following window. Click on **JAR file.**



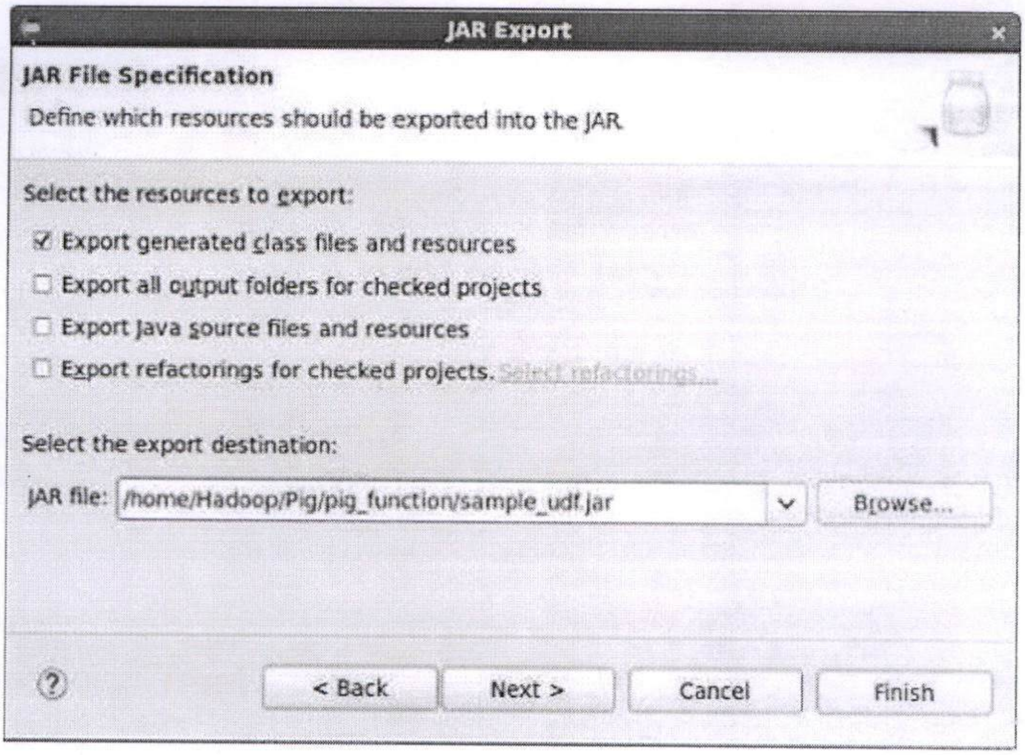
Export

Select

Export resources into a JAR file on the local file system.

Select an export destination:

- Proceed further by clicking **Next>** button. You will get another window where you need to enter the path in the local file system, where you need to store the jar file.



JAR Export

JAR File Specification

Define which resources should be exported into the JAR

Select the resources to export:

☑ Export generated class files and resources

□ Export all output folders for checked projects

□ Export Java source files and resources

□ Export refactorings for checked projects. Select refactorings...

Select the export destination:

JAR file: /home/Hadoop/Pig/pig_function/sample_udf.jar

- Finally click the **Finish** button. In the specified folder, a Jar file **sample_udf.jar** is created. This jar file contains the UDF written in Java.

Using the UDF

After writing the UDF and generating the Jar file, follow the steps given below −

Step 1: Registering the Jar file

After writing UDF (in Java) we have to register the Jar file that contain the UDF using the Register operator. By registering the Jar file, users can intimate the location of the UDF to Apache Pig.

Syntax

Given below is the syntax of the Register operator.

```
REGISTER path;
```

Example

As an example let us register the sample_udf.jar created earlier in this chapter.

Start Apache Pig in local mode and register the jar file sample_udf.jar as shown below.

```
$cd PIG_HOME/bin
$./pig –x local

REGISTER '/$PIG_HOME/sample_udf.jar'
```

Note − assume the Jar file in the path − /$PIG_HOME/sample_udf.jar

Step 2: Defining Alias

After registering the UDF we can define an alias to it using the **Define** operator.

Syntax

Given below is the syntax of the Define operator.

```
DEFINE alias {function | [`command` [input] [output] [ship] [cache] [stderr] ] };
```

Example

Define the alias for sample_eval as shown below.

```
DEFINE sample_eval sample_eval();
```

Step 3: Using the UDF

After defining the alias you can use the UDF same as the built-in functions. Suppose there is a file named emp_data in the HDFS **/Pig_Data/** directory with the following content.

```
001,Robin,22,newyork
002,BOB,23,Kolkata
003,Maya,23,Tokyo
004,Sara,25,London
005,David,23,Bhuwaneshwar
006,Maggy,22,Chennai
```

```
007,Robert,22,newyork
008,Syam,23,Kolkata
009,Mary,25,Tokyo
010,Saran,25,London
011,Stacy,25,Bhuwaneshwar
012,Kelly,22,Chennai
```

And assume we have loaded this file into Pig as shown below.

```
grunt> emp_data = LOAD 'hdfs://localhost:9000/pig_data/emp1.txt' USING PigStorage(',')
  as (id:int, name:chararray, age:int, city:chararray);
```

Let us now convert the names of the employees in to upper case using the UDF **sample_eval**.

```
grunt> Upper_case = FOREACH emp_data GENERATE sample_eval(name);
```

Verify the contents of the relation **Upper_case** as shown below.

grunt> Dump Upper_case;

```
(ROBIN)
(BOB)
(MAYA)
(SARA)
(DAVID)
(MAGGY)
(ROBERT)
(SYAM)
(MARY)
(SARAN)
(STACY)
(KELLY)
```

Parameter substitution in Pig

Earlier I have discussed about writing reusable scripts using Apache Hive, now we see how to achieve same functionality using Pig Latin.

Pig Latin has an option called param, using this we can write dynamic scripts .

Assume ,we have a file called numbers with below data.

12

23

34

12

56

34

57

12

```
Numbers = load ‘/data/numbers’ as (number:int);

specificNumber = filter numbers by number==12;

Dump specificNumber;
```

Usually we write above code in a file .let us assume we have written it in a file called numbers.pig

And we write code from file using

```
Pig –f /path/to/numbers.pig
```

Later if we want to see only numbers equals to 34, then we change second line to

```
specificNumber = filter numbers by number==34;
```

and we re-run the code using same command.
But Its not a good practice to touch the code in production ,so we can make this script dynamic by using –param option of Piglatin.
Whatever values we want to decide at the time of running we make them dynamic .now we want to decide number to be filtered at the time running job,we can write second line like below.

```
specificNumber = filter numbers by number==$dynanumber
```

and we run code like below.

```
Pig –param dynanumber=12  –f numbers.pig
```

Assume we even want to take path at the time of running script, now we write code like below

```
Numbers = load ‘$path’ as (number:int);

specificNumber = filter numbers by number==’$ dynanumber';
```

```
Dump specificNumber;
```

And run like below

```
Pig –param path=/data/path –param dynanumber =34 –f numbers.pig
```

If you feel this code is missing readability, we can specify all these dynamic values in a file like below
##Dyna.params (file name)

```
Path = /data/numbers

dynanumber = 34
```

Then you can run script with param-file option like below.

```
Pig –param-file dyna.params –f numbers.pig
```

Pig Latin provides four different types of diagnostic operators –

- Dump operator
- Describe operator
- Explanation operator
- Illustration operator

Word Count Example Using Pig Script:

```
lines = LOAD '/user/hadoop/HDFS_File.txt' AS (line:chararray);
words = FOREACH lines GENERATE FLATTEN(TOKENIZE(line)) as word;
grouped = GROUP words BY word;
wordcount = FOREACH grouped GENERATE group, COUNT(words);
DUMP wordcount;
```

The above pig script, first splits each line into words using the **TOKENIZE** operator. The tokenize function creates a bag of words. Using the **FLATTEN** function, the bag is converted into a tuple. In the third statement, the words are grouped together so that the count can be computed which is done in fourth statement.

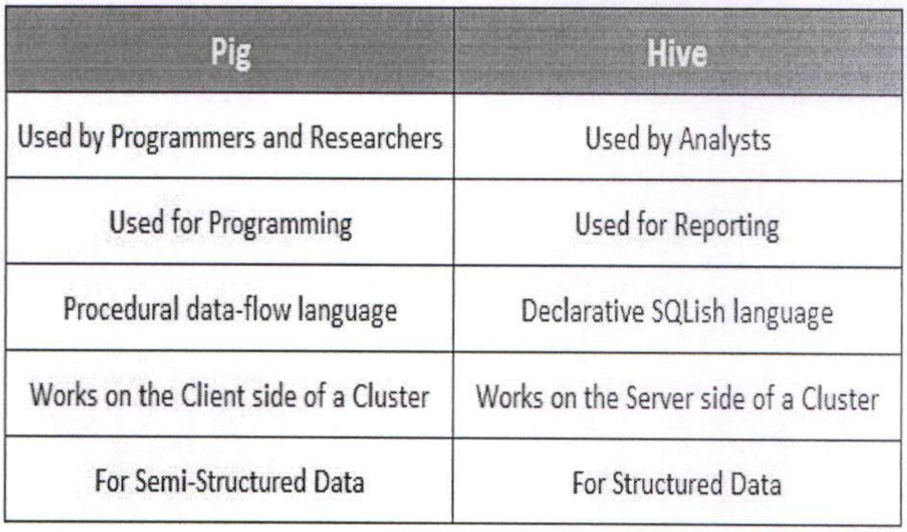
Pig at Yahoo

Pig was initially developed by Yahoo! for its data scientists who were using Hadoop. It was incepted to focus mainly on analysis of large datasets rather than on writing mapper and reduce functions. This allowed users to focus on what they want to do rather than bothering with how its done. On top of this with Pig language you have the facility to write commands in other languages like Java, Python etc. Big applications that can be built on Pig Latin can be custom built for different companies to serve different tasks related to data management. Pig systemizes all the branches of data and relates it in a manner that when the time comes, filtering and searching data is checked efficiently and quickly.

Pig Versus Hive

Pig Vs Hive

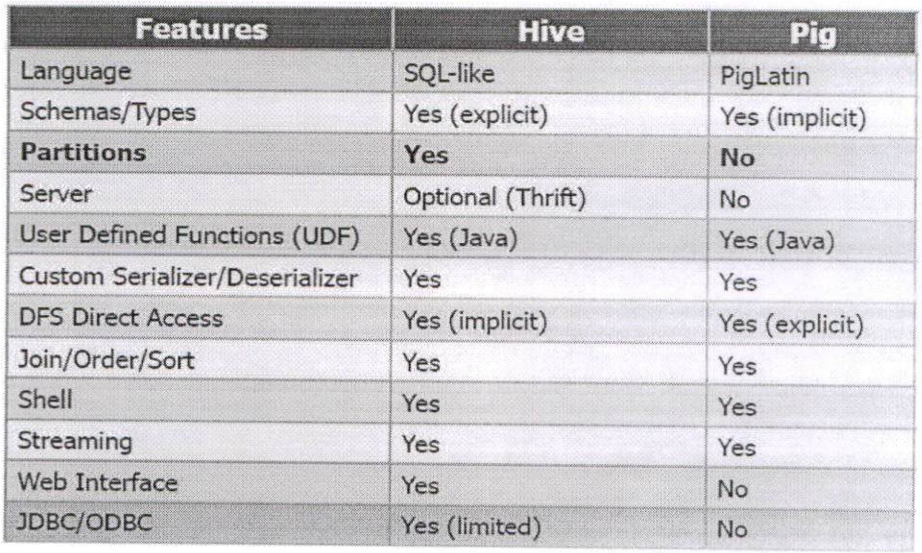
Here are some basic difference between Hive and Pig which gives an idea of which to use depending on the type of data and purpose.



Pig	Hive
Used by Programmers and Researchers	Used by Analysts
Used for Programming	Used for Reporting
Procedural data-flow language	Declarative SQLish language
Works on the Client side of a Cluster	Works on the Server side of a Cluster
For Semi-Structured Data	For Structured Data

Why Go for Hive When Pig is There?

The tabular column below gives a comprehensive comparison between the two. The Hive can be used in places where partitions are necessary and when it is essential to define and create cross-language services for numerous languages.



Features	Hive	Pig
Language	SQL-like	PigLatin
Schemas/Types	Yes (explicit)	Yes (implicit)
Partitions	**Yes**	**No**
Server	Optional (Thrift)	No
User Defined Functions (UDF)	Yes (Java)	Yes (Java)
Custom Serializer/Deserializer	Yes	Yes
DFS Direct Access	Yes (implicit)	Yes (explicit)
Join/Order/Sort	Yes	Yes
Shell	Yes	Yes
Streaming	Yes	Yes
Web Interface	Yes	No
JDBC/ODBC	Yes (limited)	No

Content beyond the syllabus

Results Analysis

CMR College of Engineering & Technology

* UGC AUTONOMOUS * Approved by AICTE * Accredited by NAAC with 'A' Grade * All B.Tech programs Accredited by NBA *

B.TECH-IV/IV I SEM Regular Results Analysis Held in November 2023: Final Result Curriculum: R18 Rev2

Branch **ELECTRONICS & COMMUNICATION ENGINEERING**

Subject	Reg	App	Passed	Failed	Reg Pass %	Apprd. Pass %
ENTREPRENEURSHIP (C30164)	7	7	6	1	85.71	85.71
AIR POLLUTION & CONTROL (A30163)	74	74	72	2	97.30	97.30
ENVIRONMENTAL PROTECTION AND MANAGEMENT (A30166)	39	39	39	0	100.00	100.00
WASTE TO ENERGY (A30378)	6	6	6	0	100.00	100.00
CLOUD COMPUTING (A30542)	17	17	15	2	88.24	88.24
INTRODUCTION TO DATA SCIENCE (A30559)	10	10	9	1	90.00	90.00
BASICS OF INSURANCE AND TAXATION (C30165)	15	15	13	2	86.67	86.67
MARKETING MANAGEMNET (C30167)	41	41	40	1	97.56	97.56
MAJOR PROJECT PHASE-I (A30428)	249	249	225	24	90.36	90.36
MINI PROJECT-II (A30426)	116	116	116	0	100.00	100.00
SUMMER INTERNSHIP-II (A30427)	133	133	133	0	100.00	100.00
ALL SUBJECTS	249	248	212	36	85.14	85.48

Branch **COMPUTER SCIENCE & ENGINEERING**

Subject	Reg	App	Passed	Failed	Reg Pass %	Apprd. Pass %
BUSINESS MANAGEMENT & FINANCIAL ANALYSIS (A30013)	260	258	247	11	95.00	95.74
DESIGN PATTERNS (A30534)	226	225	214	11	94.69	95.11
MACHINE LEARNING (A30535)	34	34	31	3	91.18	91.18
DATA ANALYTICS WITH R (A30537)	59	59	55	4	93.22	93.22
DEEP LEARNING (A30538)	66	66	61	5	92.42	92.42
ETHICAL HACKING (A30539)	135	134	124	10	91.85	92.54
BIG DATA ANALYTICS (A30540)	172	171	167	4	97.09	97.66
CLOUD COMPUTING (A30542)	88	86	76	10	86.36	88.37
KNOWLEDGE MANAGEMENT (C30162)	50	49	45	4	90.00	91.84
PYTHON PROGRAMMING (A30531)	48	48	45	3	93.75	93.75

Controller of Examinations

Principal

CMR College of Engineering & Technology

CMR
EXPLORE TO INVENT

* UGC AUTONOMOUS * Approved by AICTE * Accredited by NAAC with 'A' Grade * All B.Tech programs Accredited by NBA *

B.TECH-IV/IV I SEM Regular Results Analysis Held in November 2023: Final Result Curriculum: R18 Rev2

Branch **COMPUTER SCIENCE & ENGINEERING**

Subject	Reg	App	Passed	Failed	Reg Pass %	Apprd. Pass %
DISASTER MANAGEMENT AND MITIGATION (A30160)	54	54	53	1	98.15	98.15
ENTREPRENEURSHIP (C30164)	10	10	10	0	100.00	100.00
AIR POLLUTION CONTROL (A30163)	65	65	62	3	95.38	95.38
MARKETING MANAGEMENT (C30167)	23	23	23	0	100.00	100.00
BASICS OF INSURANCE & TAXATION (C30165)	20	19	19	0	95.00	100.00
ENVIRONMENTAL PROTECTION AND MANAGEMENT (A30166)	32	32	29	3	90.63	90.63
MAJOR PROJECT PHASE-I (A30552)	260	256	226	30	86.92	88.28
MINI PROJECT-II (A30549)	260	260	257	3	98.85	98.85
ALL SUBJECTS	260	253	215	38	82.69	84.98

Branch **INFORMATION TECHNOLOGY**

Subject	Reg	App	Passed	Failed	Reg Pass %	Apprd. Pass %
BUSINESS MANAGEMENT & FINANCIAL ANALYSIS (A30013)	65	65	60	5	92.31	92.31
DESIGN PATTERNS (A30534)	65	64	58	6	89.23	90.63
HUMAN COMPUTER INTERACTION (A31206)	65	64	61	3	93.85	95.31
BIG DATA ANALYTICS (A30540)	65	65	54	11	83.08	83.08
DISASTER MANAGEMENT AND MITIGATION (A30160)	30	30	28	2	93.33	93.33
ENTREPRENEURSHIP (C30164)	5	5	5	0	100.00	100.00
KNOWLEDGE MANAGEMENT (C30162)	2	2	2	0	100.00	100.00
PYTHON PROGRAMMING (A30531)	9	9	8	1	88.89	88.89
AIR POLLUTION AND CONTROL (A30163)	32	32	30	2	93.75	93.75
ENVIRONMENTAL PROTECTION MANAGEMENT (A30166)	1	1	1	0	100.00	100.00
WASTE TO ENERGY (A30378)	1	1	1	0	100.00	100.00
INTRODUCTION TO DATA SCIENCE (A30559)	1	1	1	0	100.00	100.00
BASICS OF INSURANCE AND TAXATION (C30165)	6	6	5	1	83.33	83.33

Controller of Examinations

Principal

End Exam Question Papers of Previous years

H.T No: | | | | | | | | | | **R18** Course Code: A30540

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

(UGC AUTONOMOUS)

B.Tech VII Semester Regular/Supplementary Examinations December-2022

Course Name: BIG DATA ANALYTICS

(Common for CSE & IT)

Date: 15.12.2022 AN **Time: 3 hours** **Max.Marks: 70**

(Note: Assume suitable data if necessary)

PART-A

Answer all TEN questions (Compulsory)
Each question carries TWO marks. **10x2=20M**

1. Write the characteristics if Big Data. 2 M
2. Define NoSQL database? List few NoSQL database systems. 2 M
3. What is replication factor and what is the default replication factor of Hadoop? 2 M
4. What is HDFS? 2 M
5. What is Mapper Phase? 2 M
6. What is Sorting and shuffling phase? 2 M
7. What is Apache PIG? 2 M
8. Write the procedure for executing pig program. 2 M
9. List the advantages of HIVE. 2 M
10. What is managed table? 2 M

PART-B

Answer the following. Each question carries TEN Marks. **5x10=50M**

11.A). Explain the differences between RDBMS and Big Data? Give suitable applications of each with an example. 10M

OR

11. B). Discuss the real time applications of Big Data and Big Data Analytics with suitable examples. 10M

12. A). Discuss block size concept of HDFS with a neat diagram. 10M

OR

12. B). How the communication takes place between name node and data node? Also explain fault-tolerance of HDFS. 10M

13. A). Implement Map Reduce Program for Word Count Problem. 10M

OR

13. B). Explain Hadoop's ecosystem and write a Hadoop command to copy data from local file system to HDFS and HDFS to local file system. 10M

(P.T.O..)

14. A). i) Explain architecture of PIG and its advantages. 5M

ii) Explain about PIG Relational Operators. 5M

OR

14. B). i) Discuss PIG components. 5M

ii) Explain about pig Load, Store and Relational Operators. 5M

15. A). Explain about HIVE characteristics, architecture and components in detail. 10M

OR

15. B). Explain HIVE data types and demonstrate in creating Table with suitable example. 10M

*****

H.T No: □□□□□□□□□□ **R18** Course Code: A30540

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

(UGC AUTONOMOUS)

B.Tech VII Semester Regular/ Supplementary Examinations November-2023

Course Name: BIG DATA ANALYTICS

(Common for CSE & IT)

Date: 05.12.2023 AN **Time: 3 hours** **Max.Marks: 70**

(Note: Assume suitable data if necessary)

PART-A

Answer all TEN questions (Compulsory)
Each question carries TWO marks. **10x2=20M**

1. List some applications of Big Data. 2 M
2. Write the characteristics of Big Data. 2 M
3. Write about data node and name node. 2 M
4. What does HDFS stand for and what is its purpose? 2 M
5. Write the list of configuration files needs to be edited to setup Hadoop. 2 M
6. How does MapReduce achieve fault tolerance? 2 M
7. List the relational operators used in PIG. 2 M
8. Write the procedure for executing PIG program. 2 M
9. What is meant by an external table? 2 M
10. What are the benefits of using HIVE? 2 M

PART-B

Answer the following. Each question carries TEN Marks. **5x10=50M**

11.A). Explain the differences between RDBMS and Big Data? Give suitable applications of each with an example. 10M

OR

11. B). What is NOSQL Database and explain the features of NOSQL Database? 10M

12. A). Describe the concept of Hadoop's Rack Awareness and discuss about the core components of Hadoop. 10M

OR

12. B). Explain the concept of block size in HDFS along with an illustrative diagram. 10M

13. A). Describe the Architecture of MapReduce and its practical applications. 10M

OR

13. B). Explain in detail about Hadoop setup on a single node. 10M

14. A). Write a Pig Latin Script for word count problem and demonstrate parameter substitution with examples. 10M

OR

14. B). Discuss PIG components and Explain about PIG Load, Store and Relational Operators. 10M

15. A). Describe the following concepts with illustrative examples: 10M
i) Loading data into HIVE Tables, ii) Managed Tables.

OR

15. B). Write about HIVE and Illustrate HIVE Architecture. 10M

*****

]

CO Attainment sheet

Evaluation and CO Assessment tools