NOIDA INSTITUTE OF ENGINEERING AND TECHNOLOGY
GREATER NOIDA-201306
(An Autonomous Institute)
School of Computer Sciences & Engineering in Emerging Technologies

Department of CS&ET (AIML)

LAB FILE

ON

DATABASE MANAGEMENT SYSTEM

(ACSAI-0452)

(4th Semester)

Submitted To:

Roshan Jameel

Submitted By:

Name: VISHAL YADAV
Roll No: 2001331530185

Affiliated to Dr. A.P.J Abdul Kalam Technical University, Uttar Pradesh, Lucknow.

Department of AIML 4th Semester

B. TECH. SECOND YEAR			
Course Code	**ACSAI0452**	**L T P**	**Credit**
Course Title	**Database Management Systems Lab**	**0 0 2**	**1**

List of Experiments:

Sr. No.	Name of Experiment	CO	Date of Experiment	Faculty Sign
1.	Installing ORACLE/ MYSQL/NOSQL.	CO1		
2.	Creating Entity-Relationship Diagram using case tools with Identifying (Entities, attributes, keys and relationships between entities, cardinalities, generalization, specialization etc.)	CO1		
3.	I. Implement DDL commands –Create, Alter, Drop etc. II. Implement DML commands- Insert, Select, Update, Delete	CO2		
4.	I. Implement DCL commands-Grant and Revoke II. Implement TCL commands- Rollback, Commit, Save point III. Implement different type key: -Primary Key, Foreign Key and Unique etc.	CO2		
5.	Converting ER Model to Relational Model (Represent entities and relationships in Tabular form, represent attributes as columns, identifying keys).	CO1, CO2		
6.	Practice Queries using COUNT, SUM, AVG, MAX, MIN, GROUP BY, HAVING, VIEWS Creation and Dropping.	CO2		
7.	Practicing Queries using ANY, ALL, IN, EXISTS, NOT EXISTS, UNION, INTERSECT, CONSTRAINTS etc.	CO2		
8.	Practicing Sub queries (Nested, Correlated) and Joins (Inner, Outer and Equi).	CO2		
9.	**Practicing on Triggers** - creation of trigger, Insertion using trigger, Deletion using trigger, Updating using trigger	CO4		
10.	**Procedures-** Creation of Stored Procedures, Execution of Procedure, and Modification of Procedure	CO4		
11.	**Cursors**- Declaring Cursor, Opening Cursor, Fetching the data, closing the cursor.	CO4		
12.	Study of Open Source NOSQL Database: MongoDB (Installation, Basic CRUD operations, Execution)	CO5		

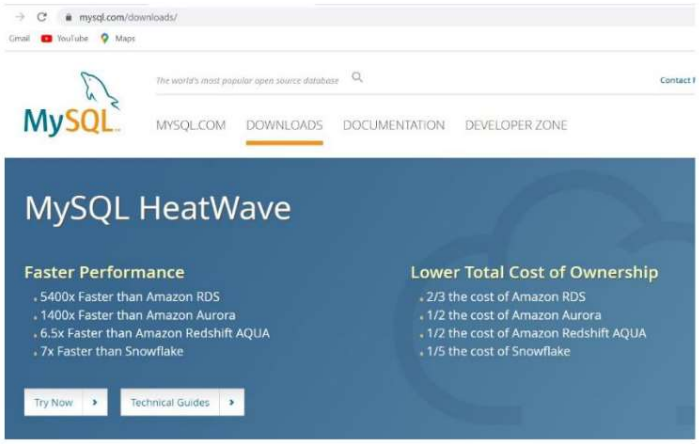
13.	Design and Develop MongoDB Queries using CRUD operations. (Use CRUD operations, SAVE method, logical operators)	CO5		
14.	Implement aggregation and indexing with suitable example using MongoDB.	CO5		
15.	Mini project (Design & Development of Data and Application) for following: - a) Inventory Control System.	CO1		
	b) Material Requirement Processing. c) Hospital Management System. d) Railway Reservation System. e) Personal Information System. f) Web Based User Identification System. g) Timetable Management System. h) Hotel Management System			
Lab Course Outcome: After completion of this course students will be able to				
CO 1	Design and implement the ER, EER model to solve the real-world problem and transform an information model into a relational database schema and to use a data.	K6		
CO 2	Formulate and evaluate query using SQL solutions to a broad range of query and data update problems.	K6		
CO 3	Apply and create PL/SQL blocks, procedure functions, packages and triggers, cursors.	K3, K6		
CO 4	Analyze entity integrity, referential integrity, key constraints, and domain constraints on database.	K4		
CO5	Demonstrate understanding of MongoDB and its query operations.	K3		

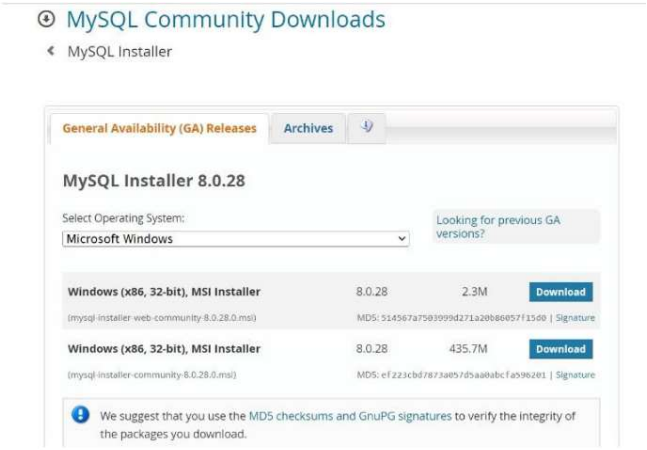
Student Signature **Faculty Signature**

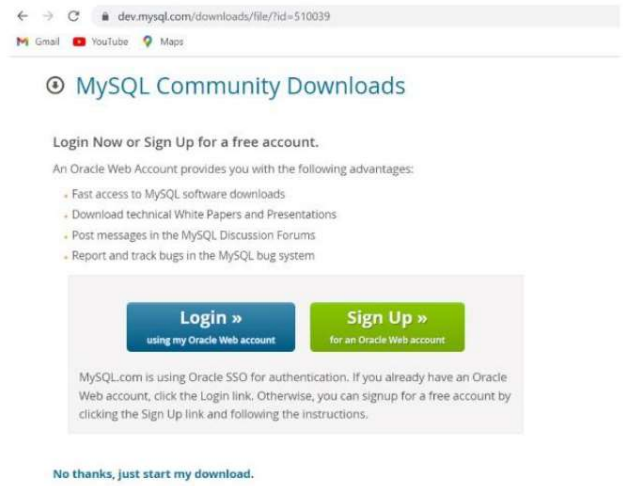
Remarks (If Any)

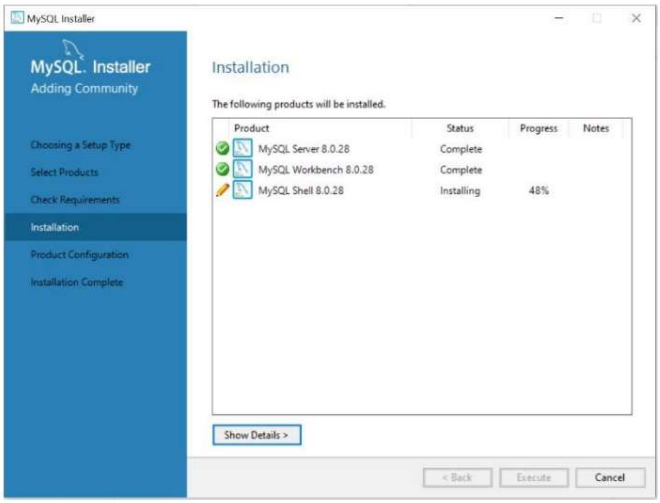
PROGRAM 1

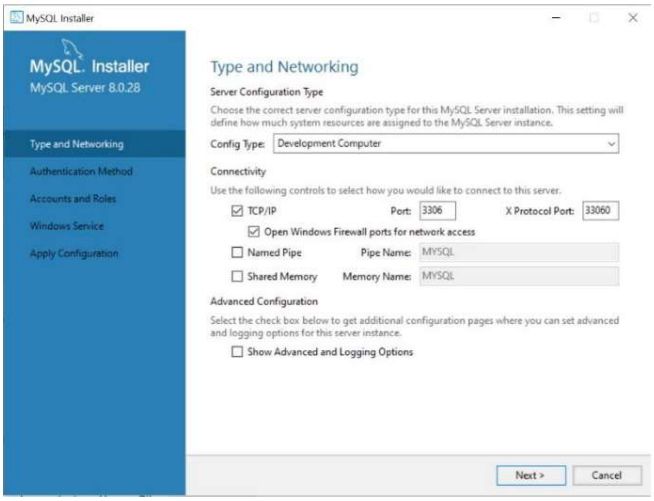
Installing ORACLE/ MYSQL/NOSQL.

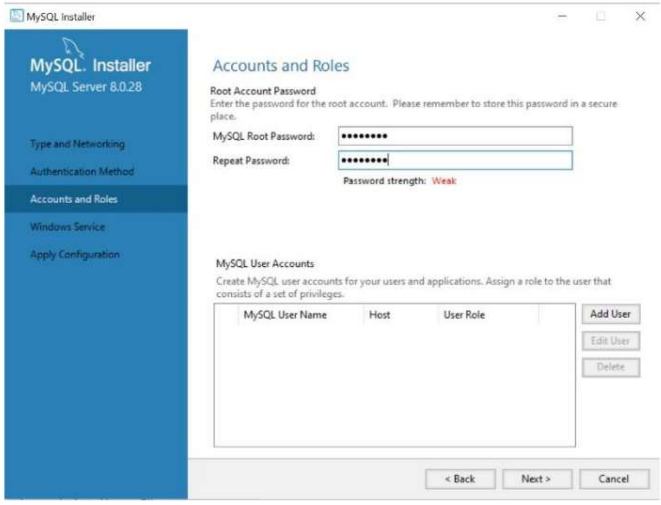


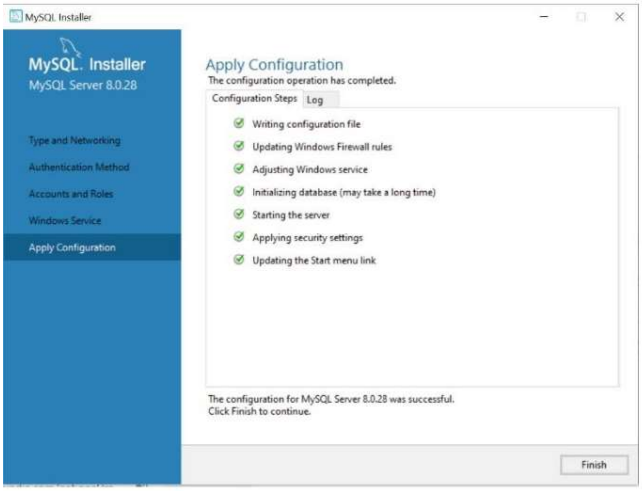


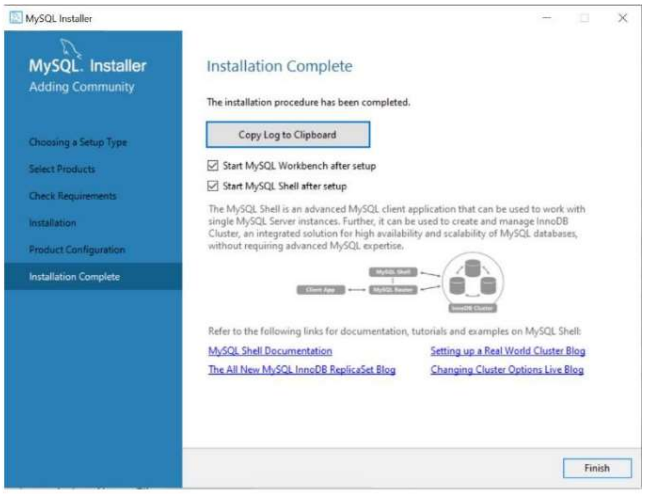


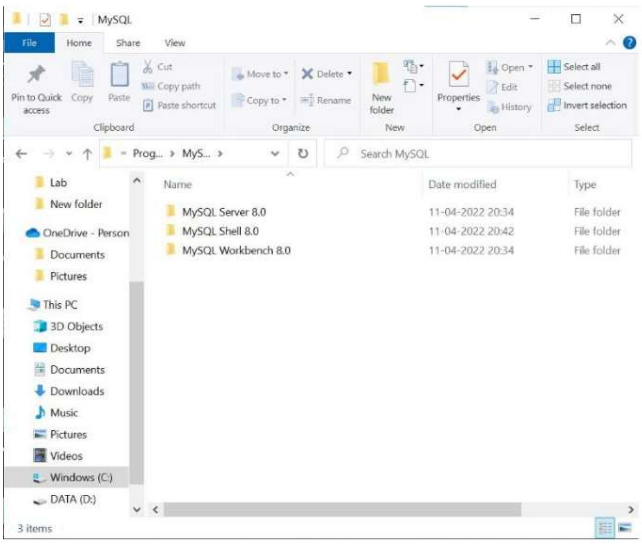


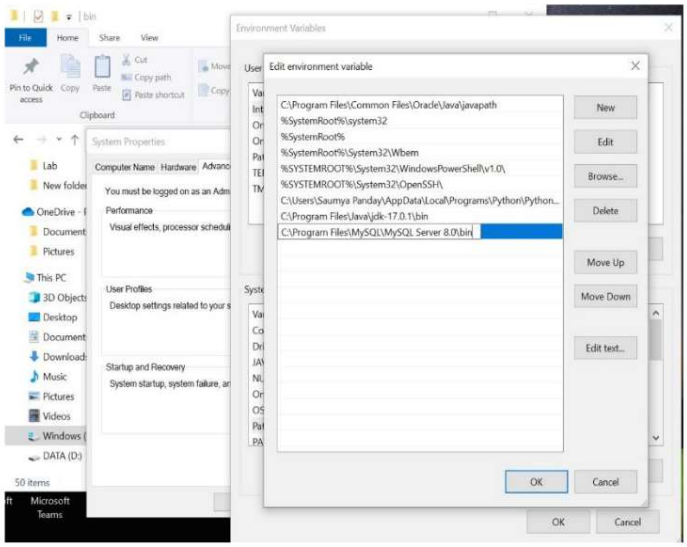




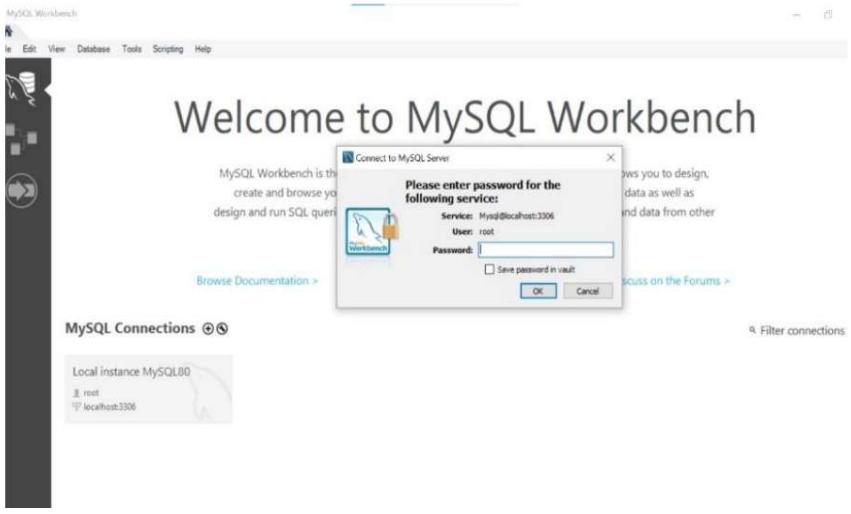

MySQL Installer
MySQL. Installer
MySQL Server 8.0.28
Type and Networking
Authentication Method
Accounts and Roles
Windows Service
Apply Configuration
Apply Configuration
The configuration operation has completed.
Configuration Steps
Log
Writing configuration file
Updating Windows Firewall rules
Adjusting Windows service
Initializing database (may take a long time)
Starting the server
Applying security settings
Updating the Start menu link
The configuration for MySQL Server 8.0.28 was successful.
Click Finish to continue.
Finish


MySQL Installer
MySQL. Installer
Adding Community
Choosing a Setup Type
Select Products
Check Requirements
Installation
Product Configuration
Installation Complete
Installation Complete
The installation procedure has been completed.
Copy Log to Clipboard
Start MySQL Workbench after setup
Start MySQL Shell after setup
The MySQL Shell is an advanced MySQL client application that can be used to work with single MySQL Server instances. Further, it can be used to create and manage InnoDB Cluster, an integrated solution for high availability and scalability of MySQL databases, without requiring advanced MySQL expertise.
Refer to the following links for documentation, tutorials and examples on MySQL Shell:
MySQL Shell Documentation
Setting up a Real World Cluster Blog
The All New MySQL InnoDB ReplicaSet Blog
Changing Cluster Options Live Blog
Finish


MySQL
File
Home
Share
View
Pin to Quick access
Copy
Paste
Cut
Copy path
Paste shortcut
Clipboard
Move to
Copy to
Delete
Rename
Organize
New folder
New
Properties
Open
Edit
History
Select all
Select none
Invert selection
Select
Prog... › MyS... ›
Search MySQL
Lab
New folder
OneDrive - Person
Documents
Pictures
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Windows (C:)
DATA (D:)
Name
Date modified
Type
MySQL Server 8.0
11-04-2022 20:34
File folder
MySQL Shell 8.0
11-04-2022 20:42
File folder
MySQL Workbench 8.0
11-04-2022 20:34
File folder
3 items


Environment Variables
Edit environment variable
C:\Program Files\Common Files\Oracle\Java\javapath
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
C:\Users\Saumya Panday\AppData\Local\Programs\Python\Python...
C:\Program Files\Java\jdk-17.0.1\bin
C:\Program Files\MySQL\MySQL Server 8.0\bin
New
Edit
Browse...
Delete
Move Up
Move Down
Edit text...
OK
Cancel
System Properties
Computer Name
Hardware
Performance
User Profiles
Startup and Recovery
50 items

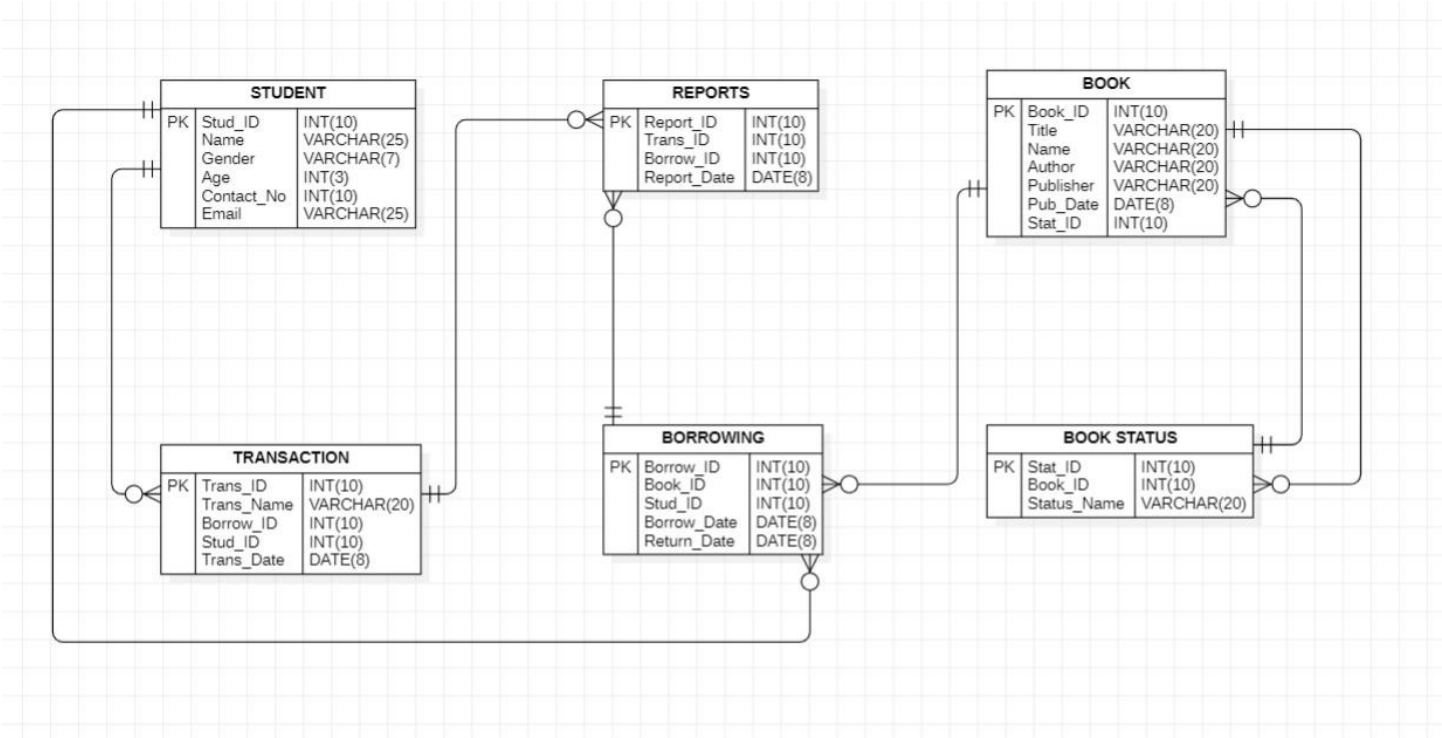

Command Prompt - mysql -u root -p
Microsoft Windows [Version 10.0.19044.1586]
(c) Microsoft Corporation. All rights reserved.
C:\Users\Saumya Panday>mysql --version
mysql Ver 8.0.28 for Win64 on x86_64 (MySQL Community Server - GPL)
C:\Users\Saumya Panday>mysql -u root -p
Enter password: ********
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 8.0.28 MySQL Community Server - GPL
Copyright (c) 2000, 2022, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> show databases
-> show databases;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corres
t line 2
mysql> show databases;
+--------------------+
| Database |
+--------------------+
| information_schema |
| mysql |
| performance_schema |
| sys |
+--------------------+
4 rows in set (0.01 sec)
mysql>


MySQL Workbench
File
Edit
View
Database
Tools
Scripting
Help
Welcome to MySQL Workbench
Connect to MySQL Server
Please enter password for the following service:
Service: Mysql@localhost:3306
User: root
Password:
Save password in vault
OK
Cancel
Browse Documentation >
MySQL Connections
Filter connections
Local instance MySQL80
root
localhost:3306

PROGRAM 2

Creating Entity-Relationship Diagram using case tools with Identifying (Entities, attributes, keys and relationships between entities, cardinalities, generalization, specialization etc.)

ER DIAGRAM ON LIBRARY MANAGEMENT SYSTEM



PROGRAM 3

I. Implement DDL commands –Create, Alter, Drop etc.

II. Implement DML commands- Insert, Select, Update, Delete

DDL COMMANDS

CREATE

```
1  create database ABC;
2  use ABC;
3  create table details(Cust_ID int(20), Cust_Name varchar(30), Cust_Address varchar(40));
4  select * from details;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Cust_ID	Cust_Name	Cust_Address

ALTER

```
5  alter table details add Mobile_no int(12);
6  select * from details;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Cust_ID	Cust_Name	Cust_Address	Mobile_no

DROP

```
7  drop table details;
8  select * from details;
9  drop database ABC;
10 show databases;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Database
information_schema
mysql
performance_schema
sys

TRUNCATE

```
11  truncate table details;
12  truncate database sys;
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	12:57:53	create database ABC	1 row(s) affected	0.031 sec
2	12:58:12	use ABC	0 row(s) affected	0.000 sec
3	13:00:34	create table details(Cust_ID int(20), Cust_Name varchar(30), Cust_Address varchar(40))	0 row(s) affected, 1 warning(s): 1681 Integer display width is deprecated and will be remo...	0.078 sec
4	13:12:43	select * from details LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000
5	13:24:02	alter table details add Mobile_no int(12)	0 row(s) affected, 1 warning(s): 1681 Integer display width is deprecated and will be remo...	0.031 sec
6	13:24:29	select * from details LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000
7	13:28:04	drop table details	0 row(s) affected	0.031 sec
8	13:28:26	select * from details LIMIT 0, 1000	Error Code: 1146. Table 'abc.details' doesn't exist	0.016 sec
9	13:29:22	drop database ABC	0 row(s) affected	0.015 sec
10	13:29:36	show databases	4 row(s) returned	0.000 sec / 0.000

DML COMMANDS

INSERT

```
insert into details values(101,"Abc","Noida",12345),
(102,"Shyam","Delhi",98765),
(103,"Ram","Gr. Noida",09876),
(104,"Xyz","Gzb",67453),
(105,"Kjl","Noida",67843);
```

SELECT

```
select * from details;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Cust_ID	Cust_Name	Cust_Address	Mobile_no
101	Abc	Noida	12345
102	Shyam	Delhi	98765
103	Ram	Gr. Noida	9876
104	Xyz	Gzb	67453
105	Kjl	Noida	67843

```
select * from details where  Cust_Address = "Noida";
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Cust_ID	Cust_Name	Cust_Address	Mobile_no
101	Abc	Noida	12345
105	Kjl	Noida	67843

```
select * from details where  not Cust_Address = "Noida";
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Cust_ID	Cust_Name	Cust_Address	Mobile_no
102	Shyam	Delhi	98765
103	Ram	Gr. Noida	9876
104	Xyz	Gzb	67453

UPDATE

```
update details set Cust_Address = "Ghaziabad" where Cust_Address = "Gzb";
select * from details;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Cust_ID	Cust_Name	Cust_Address	Mobile_no
101	Abc	Noida	12345
102	Shyam	Delhi	98765
103	Ram	Gr. Noida	9876
104	Xyz	Ghaziabad	67453
105	Kjl	Noida	67843

DELETE

```
delete from details where Cust_ID = 104;
select * from details;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Cust_ID	Cust_Name	Cust_Address	Mobile_no
101	Abc	Noida	12345
102	Shyam	Delhi	98765
103	Ram	Gr. Noida	9876
105	Kjl	Noida	67843

PROGRAM 4

1. ***Implement DCL commands-Grant and Revoke***
2. ***Implement TCL commands- Rollback, Commit, Save point***
3. ***Implement different type key: -Primary Key, Foreign Key and Unique etc.***

GRANT

```
1 •  use ABC;
2
3 •  GRANT SELECT ON classes TO'root'@'localhost';
4 •  GRANT UPDATE ON classes TO'root'@'localhost';
5 •  SHOW GRANTS FOR 'root'@'localhost';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Grants for root@localhost
GRANT SELECT, INSERT, UPDATE, DELETE, CR...
GRANT APPLICATION_PASSWORD_ADMIN,AU...
GRANT SELECT, UPDATE ON `abc`.`classes` T...
GRANT PROXY ON ``@`` TO `root`@`localho...

REVOKE

```
6 •  REVOKE SELECT ON classes from'root'@'localhost';
7 •  REVOKE UPDATE ON classes from'root'@'localhost';
8 •  REVOKE DELETE ON classes from'root'@'localhost';
9 •  SHOW GRANTS FOR 'root'@'localhost';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Grants for root@localhost
GRANT SELECT, INSERT, UPDATE, DELETE, CR...
GRANT APPLICATION_PASSWORD_ADMIN,AU...
GRANT PROXY ON ``@`` TO `root`@`localho...

Action Output

#	Time	Action	Message
1	10:58:49	use ABC	0 row(s) affected
2	10:58:59	GRANT SELECT ON classes TO'root'@'localhost'	0 row(s) affected
3	10:59:02	GRANT UPDATE ON classes TO'root'@'localhost'	0 row(s) affected
4	10:59:05	SHOW GRANTS FOR 'root'@'localhost'	4 row(s) returned
5	11:00:07	REVOKE SELECT ON classes from'root'@'localhost'	0 row(s) affected
6	11:00:11	REVOKE UPDATE ON classes from'root'@'localhost'	0 row(s) affected
7	11:00:14	REVOKE DELETE ON classes from'root'@'localhost'	Error Code: 1147. There is no such grant defined for user 'root' on host 'localhost' on tabl...
8	11:00:35	SHOW GRANTS FOR 'root'@'localhost'	3 row(s) returned
9	11:03:54	SHOW GRANTS FOR 'root'@'localhost'	3 row(s) returned

2.

```
17 •  delete from classes where Id_no = 203;
18 •  select * from classes;
19 •  commit;
20 •  select * from classes;
21 •  delete from classes where Id_no = 205;
22 •  rollback;
23 •  select * from classes;
24 •  insert into classes values(203,"SDF",4);
25 •  savepoint A;
```

Action Output

#	Time	Action	Message
13	11:12:02	delete from classes where Id_no = 203	1 row(s) affected
14	11:12:20	select * from classes LIMIT 0, 1000	4 row(s) returned
15	11:12:32	commit	0 row(s) affected
16	11:12:39	select * from classes LIMIT 0, 1000	4 row(s) returned
17	11:13:17	rollback	0 row(s) affected
18	11:13:22	select * from classes LIMIT 0, 1000	4 row(s) returned
19	11:14:14	delete from classes where Id_no = 205	1 row(s) affected
20	11:14:18	rollback	0 row(s) affected
21	11:14:21	select * from classes LIMIT 0, 1000	3 row(s) returned
22	11:16:57	insert into classes values(203,"SDF",4)	1 row(s) affected
23	11:17:11	savepoint A	0 row(s) affected

3.

```
27  create table Information(Person_ID int(5), Name Varchar(20),City Varchar(20), PRIMARY KEY(Person_ID));
28  insert into Information values(401,"DFG","Noida"),
29  (402,"ERTY","delhi"),
30  (403,"FREG","Gzb"),
31  (404,"NBJKDF","Vns");
32  select * from Information;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Person_ID	Name	City
401	DFG	Noida
402	ERTY	delhi
403	FREG	Gzb
404	NBJKDF	Vns
NULL	NULL	NULL

```
40  create table Information1(Person_ID int(5), Name Varchar(20),City Varchar(20),
41  PRIMARY KEY(Person_ID), FOREIGN KEY(Person_ID) REFERENCES Person1(Person_ID));
42  insert into Information1 values(401,"DFG","Noida"),
43  (402,"ERTY","delhi"),
44  (403,"FREG","Gzb"),
45  (404,"NBJKDF","Vns");
46  select * from Information1;
47  ALTER TABLE Information1 ADD FOREIGN KEY (Person_ID) REFERENCES Person1(Person_ID);
48  select * from Information1;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Person_ID	Name	City
401	DFG	Noida
402	ERTY	delhi
403	FREG	Gzb
404	NBJKDF	Vns
NULL	NULL	NULL

```
50  create table Information2(Person_ID int(5), Name Varchar(20),City Varchar(20),
51  UNIQUE(Person_ID));
52  insert into Information2 values(401,"DFG","Noida"),
53  (402,"ERTY","delhi"),
54  (403,"FREG","Gzb"),
55  (404,"NBJKDF","Vns");
56  select * from Information2;
```

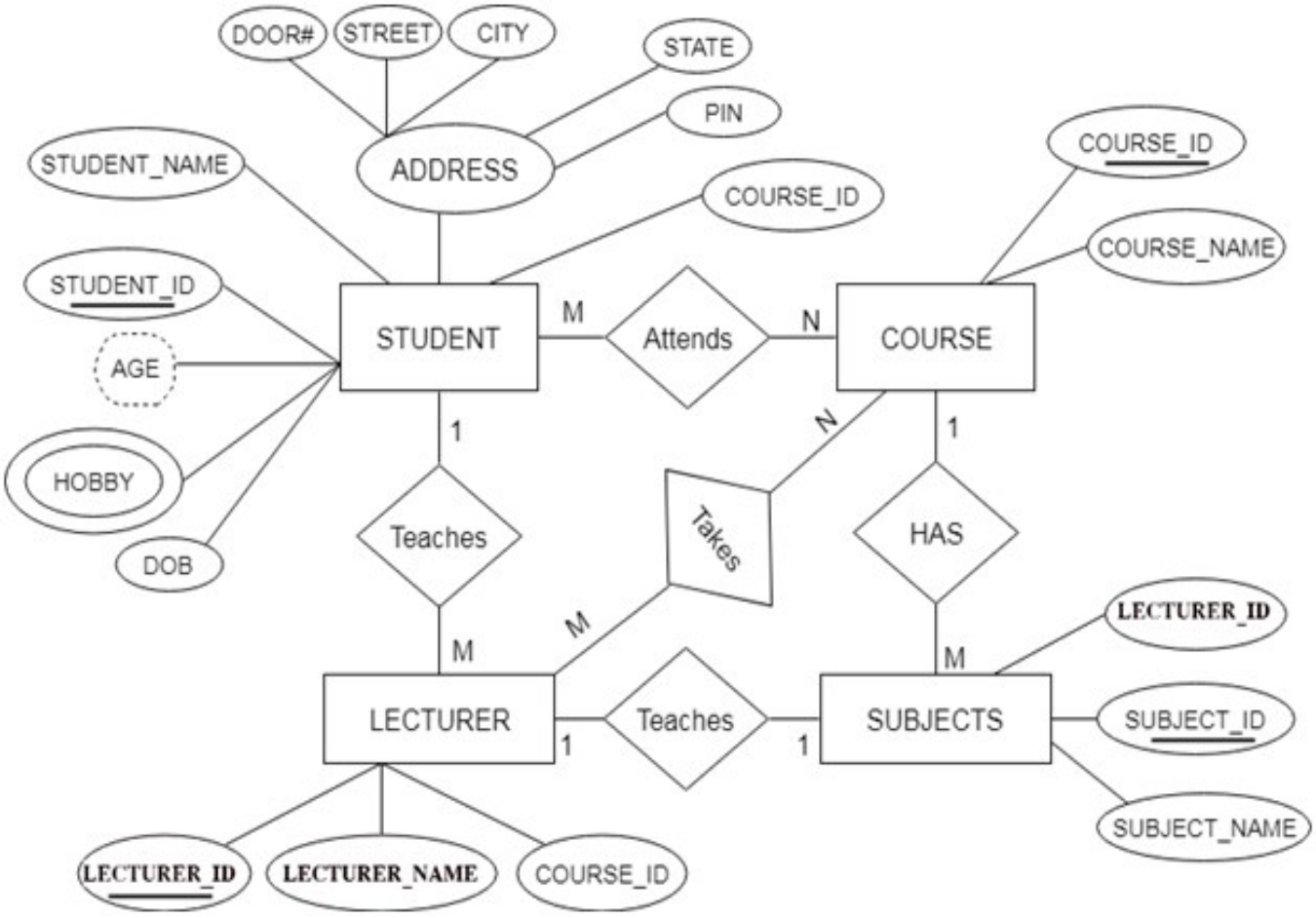
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Person_ID	Name	City
401	DFG	Noida
402	ERTY	delhi
403	FREG	Gzb
404	NBJKDF	Vns

PROGRAM 5

Converting ER Model to Relational Model (Represent entities and relationships in Tabular form, represent attributes as columns, identifying keys).

- Entity type becomes a table.
- All single-valued attribute becomes a column for the table.
- A key attribute of the entity type represented by the primary key.
- The multivalued attribute is represented by a separate table.
- Composite attribute represented by components.
- Derived attributes are not considered in the table.



a. In the given ER diagram, LECTURE, STUDENT, SUBJECT and COURSE forms individual tables.

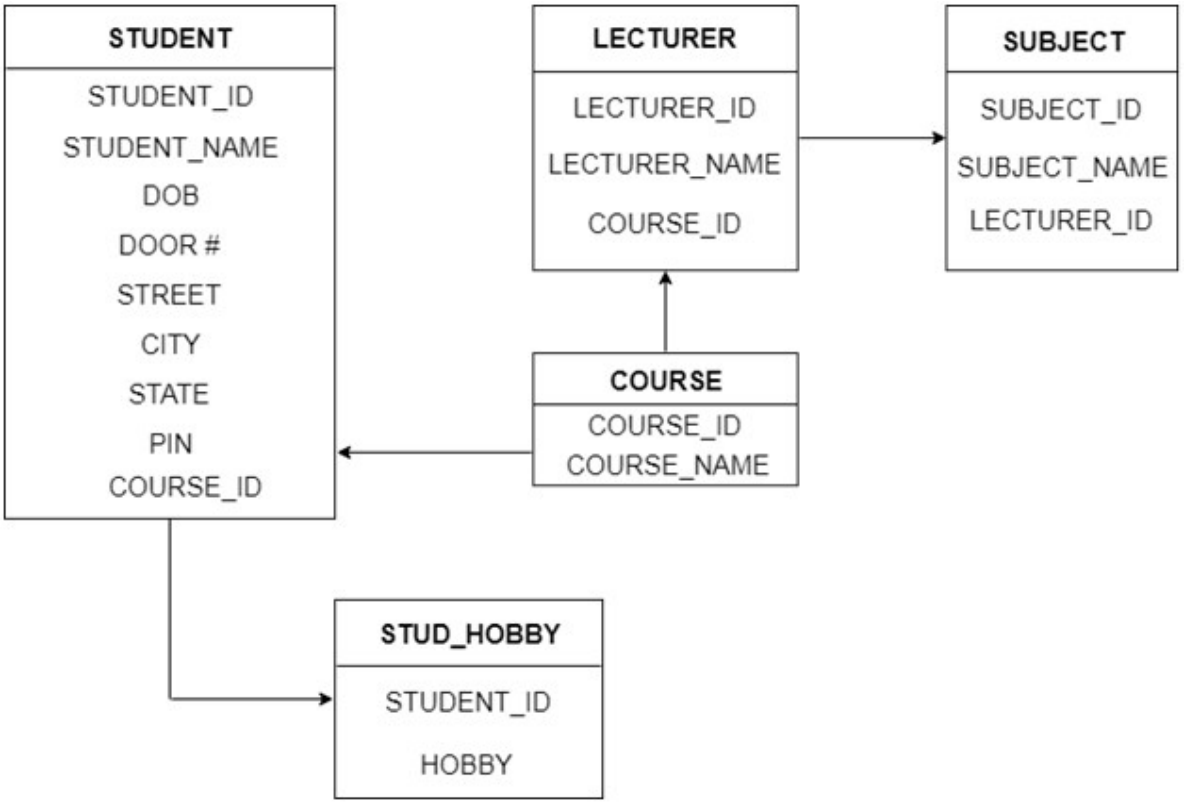
b. In the STUDENT entity, STUDENT_NAME and STUDENT_ID form the column of STUDENT table. Similarly, COURSE_NAME and COURSE_ID form the column of COURSE table and so on.

c. In the given ER diagram, COURSE_ID, STUDENT_ID, SUBJECT_ID, and LECTURE_ID are the key attribute of the entity.

d. In the student table, a hobby is a multivalued attribute. So it is not possible to represent multiple values in a single column of STUDENT table. Hence we create a table STUD_HOBBY with column name STUDENT_ID and HOBBY. Using both the column, we create a composite key.

e. In the given ER diagram, student address is a composite attribute. It contains CITY, PIN, DOOR#, STREET, and STATE. In the STUDENT table, these attributes can merge as an individual column.

f. In the STUDENT table, Age is the derived attribute. It can be calculated at any point of time by calculating the difference between current date and Date of Birth.



PROGRAM 6

Practice Queries using COUNT, SUM, AVG, MAX, MIN, GROUP BY, HAVING, VIEWS Creation and Dropping.

```
1  show databases;
2  use ABC;
3  create table Product_Mast( Product Varchar(20), Company Varchar(20), Qty int(4), Rate int(4), Cost int(4) );
4  insert into Product_Mast values("Item1", "Com1", 2, 10, 20);
5  insert into Product_Mast values("Item2", "Com2", 3, 25, 75);
6  insert into Product_Mast values("Item3", "Com1", 2, 30, 60);
7  insert into Product_Mast values("Item4", "Com3", 5, 10, 50);
8  insert into Product_Mast values("Item5", "Com2", 2, 20, 40);
9  insert into Product_Mast values("Item6", "Com1", 3, 25, 75);
10 insert into Product_Mast values("Item7", "Com1", 5, 30, 150);
11 insert into Product_Mast values("Item8", "Com1", 3, 10, 30);
12 insert into Product_Mast values("Item9", "Com2", 2, 25, 50);
13 insert into Product_Mast values("Item10", "Com3", 4, 30, 120);
14 select count(*) from Product_Mast;
15 select count(*) from Product_Mast where Rate>=20;
16 select count(distinct Company) from Product_Mast;
17 select Company, count(*) from Product_Mast group by Company having Count(*) >2;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

count(*)
10

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

count(*)
7

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

count(distinct Company)
3

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Company	count(*)
Com1	5
Com2	3

```
18 select sum(Qty) from Product_Mast;
19 select sum(Rate) from Product_Mast;
20 select sum(Cost) from Product_Mast;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

sum(Qty)
31

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

sum(Rate)
215

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

sum(Cost)
670

```
21 select avg(Qty) from Product_Mast;
22 select avg(Rate) from Product_Mast;
23 select avg(Cost) from Product_Mast;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

avg(Qty)
3.1000

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

avg(Rate)
21.5000

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

avg(Cost)
67.0000

```
24 •  select max(Qty) from Product_Mast;
25 •  select max(Rate) from Product_Mast;
26 •  select max(Cost) from Product_Mast;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

max(Qty)
5

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

max(Rate)
30

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

max(Cost)
150

```
27 •  select min(Qty) from Product_Mast;
28 •  select min(Rate) from Product_Mast;
29 •  select min(Cost) from Product_Mast;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

min(Qty)
2

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

min(Rate)
10

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

min(Cost)
20

```
30 •  SELECT Company FROM Product_Mast GROUP BY Company;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Company
Com1
Com2
Com3

```
31 •  select Qty from Product_Mast HAVING count(*) > 4;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Qty
2

```
32 •  create view items1 as select Product,Qty,Company from Product_Mast;
33 •  select * from items1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Product	Qty	Company
Item1	2	Com1
Item2	3	Com2
Item3	2	Com1
Item4	5	Com3
Item5	2	Com2
Item6	3	Com1
Item7	5	Com1
Item8	3	Com1
Item9	2	Com2
Item10	4	Com3

```
34 •  drop view items1;
35 •  select * from items1;
```

Action Output

#	Time	Action	Message
44	16:52:46	drop view items1	0 row(s) affected
45	16:53:20	select * from items1 LIMIT 0, 1000	Error Code: 1146. Table 'abc.items1' doesn't exist

PROGRAM 7

Practicing Queries using ANY, ALL, IN, EXISTS, NOT EXISTS, UNION, INTERSECT, CONSTRAINTS etc.

```
use ABC;
create table Employee1(Emp_Id int(4), Emp_Name Varchar(20), City Varchar(20), Salary int(12), Age int(3));
insert into Employee1 values(1, "Angelina", "Chicago", 200000, 30);
insert into Employee1 values(2, "Robert", "Austin", 300000, 26);
insert into Employee1 values(3, "Christian", "Denver", 100000, 42);
insert into Employee1 values(4, "Kristen", "Washington", 500000, 29);
insert into Employee1 values(5, "Russell", "Los angels", 200000, 36);
insert into Employee1 values(6, "Marry", "Canada", 600000, 48);
select * from Employee1;
select Emp_Id,Emp_Name from Employee1 where Emp_Id = Any(select Emp_Id from Employee1 where Salary > 200000);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Emp_Id	Emp_Name
2	Robert
4	Kristen
6	Marry

```
select Emp_Id,Emp_Name,Salary from Employee1 where Emp_Id < All(select Emp_Id from Employee1 where Salary > 400000);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Emp_Id	Emp_Name	Salary
1	Angelina	200000
2	Robert	300000
3	Christian	100000

```
select Emp_Id,Emp_Name,Salary from Employee1 where City in('Chicago','Canada');
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Emp_Id	Emp_Name	Salary
1	Angelina	200000
6	Marry	600000

```
select Emp_Id,Emp_Name,Salary from Employee1 where Exists
(select * From Project where Project.Emp_Id = Employee1.Emp_Id);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Emp_Id	Emp_Name	Salary
1	Angelina	200000
2	Robert	300000
3	Christian	100000
4	Kristen	500000

```
select Emp_Id,Emp_Name,Salary from Employee1 where NOT Exists
(select * From Project where Project.Emp_Id = Employee1.Emp_Id);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Emp_Id	Emp_Name	Salary
5	Russell	200000
6	Marry	600000

```
32 •  select * from Student1
33    UNION
34    Select * from Student2;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Stu_ID	Stu_Name	Class
1	DFFG	3
2	DFFGG	4
3	DFER	5
4	DRFG	2
5	DFFU	4
6	DRUFG	6

```
35 •  select distinct Stu_ID from Student1
36    where Stu_ID in(select Stu_ID from Student1);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Stu_ID
1
2
3
4
5
6

```
37 •  CREATE TABLE Student3(Id INT(4), LastName TEXT NOT NULL, FirstName TEXT NOT NULL, City VARCHAR(35));
38 •  INSERT INTO Student3 VALUES(1, 'Hanks', 'Peter', 'New York');
39 •  INSERT INTO Student3 VALUES(2, NULL, 'Amanda', 'Florida');
```

Output

Action Output

#	Time	Action	Message
93	00:05:59	INSERT INTO Student3 VALUES(1, 'Hanks', 'Peter', 'New York')	1 row(s) affected
94	00:06:27	INSERT INTO Student3 VALUES(2, NULL, 'Amanda', 'Florida')	Error Code: 1048. Column 'LastName' cannot be null

```
40 •  CREATE TABLE ShirtBrands(Id INTEGER, BrandName VARCHAR(40) UNIQUE, Size VARCHAR(30));
41 •  INSERT INTO ShirtBrands(Id, BrandName, Size) VALUES(1, 'Pantaloons', 38), (2, 'Cantabil', 40);
42 •  INSERT INTO ShirtBrands(Id, BrandName, Size) VALUES(1, 'Raymond', 38), (2, 'Cantabil', 40);
```

Context Help | Snipp

Output

Action Output

#	Time	Action	Message
96	00:09:05	INSERT INTO ShirtBrands(Id, BrandName, Size) VALUES(1, 'Pantaloons', 38), (2, 'Can...	2 row(s) affected Records: 2 Duplicates: 0 Warnings: 0
97	00:09:27	INSERT INTO ShirtBrands(Id, BrandName, Size) VALUES(1, 'Raymond', 38), (2, 'Cant...	Error Code: 1062. Duplicate entry 'Cantabil' for key 'shirtbrands.BrandName'

PROGRAM 8

Practicing Sub queries (Nested, Correlated) and Joins (Inner, Outer and Equi).

```
4 •  select Employee1.Emp_Name,Employee1.City from Employee1
5    where Employee1.Emp_Id IN(
6    select Project.Emp_Id
7    from Project
8    where Project.Department = "Development");
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Emp_Name	City
Robert	Austin
Kristen	Washington

```
9 •  select Project.Project_No,Employee1.Emp_Name,
10   Project.Department from Employee1 inner join Project on
11   Project.Emp_Id = Employee1.Emp_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Project_No	Emp_Name	Department
101	Angelina	Testing
102	Robert	Development
103	Christian	Designing
104	Kristen	Development

```
15 •  select Project.Project_No,Project.Department, Employee1.Emp_Name
16    from Project right join Employee1 on
17    Project.Emp_Id = Employee1.Emp_id
18    ORDER BY Employee1.Emp_Id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Project_No	Department	Emp_Name
101	Testing	Angelina
102	Development	Robert
103	Designing	Christian
104	Development	Kristen
NULL	NULL	Russell
NULL	NULL	Marry

```
12 •  select Project.Project_No,Employee1.Emp_Name,Project.Department
13    from Employee1, Project where
14    Project.Emp_Id = Employee1.Emp_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Project_No	Emp_Name	Department
101	Angelina	Testing
102	Robert	Development
103	Christian	Designing
104	Kristen	Development

PROGRAM 9

Practicing on Triggers - creation of trigger, Insertion using trigger, Deletion using trigger, Updating using trigger

```
create table Order_List(Order_Id int(5),Prod_Name Varchar(20),Price int(10));
insert into Order_List values(101,"Keyboard",15000),
(102,"Mouse",5000),
(103,"Printer",25000);
create trigger hike
before insert on Order_List
for each row
set new.price = new.price+2000;
insert into Order_List values(104,"Scanner",3000);
select * from Order_List;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Order_Id	Prod_Name	Price
101	Keyboard	15000
102	Mouse	5000
103	Printer	25000
104	Scanner	5000

```
create table Order_List1(Order_List int);
create trigger hike1
after insert on Order_List
for each row
insert into Order_List1 values(new.price);
insert into Order_List values(105,"Printer",8000);
select * from Order_List;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Order_Id	Prod_Name	Price
101	Keyboard	15000
102	Mouse	5000
103	Printer	25000
104	Scanner	5000
105	Printer	10000

```
DELIMITER $$

CREATE TRIGGER before_salaries_delete
BEFORE DELETE
ON salaries FOR EACH ROW
BEGIN
    INSERT INTO SalaryArchives(employeeNumber,validFrom,amount)
    VALUES(OLD.employeeNumber,OLD.validFrom,OLD.amount);
END$$

DELIMITER ;
SELECT * FROM SalaryArchives;
```

```
DELETE FROM salaries
WHERE employeeNumber = 1002;
SELECT * FROM SalaryArchives;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	employeeNumber	validFrom	amount	deletedAt
NULL	1002	2000-01-01	50000.00	2022-05-06 21:07:06

```
45 •  DELETE FROM salaries;
46 •  SELECT * FROM SalaryArchives;
47
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	employeeNumber	validFrom	amount	deletedAt
NULL	1002	2000-01-01	50000.00	2022-05-06 21:07:06
NULL	1056	2000-01-01	60000.00	2022-05-06 21:10:01
NULL	1076	2000-01-01	70000.00	2022-05-06 21:10:01

```
60 •  CREATE TRIGGER after_salaries_delete
61    AFTER DELETE
62    ON Salaries1 FOR EACH ROW
63    UPDATE SalaryBudgets
64    SET total = total - old.salary;
65 •  DELETE FROM Salaries
66    WHERE employeeNumber = 1002;
67 •  SELECT * FROM SalaryBudgets;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

total
20000.00

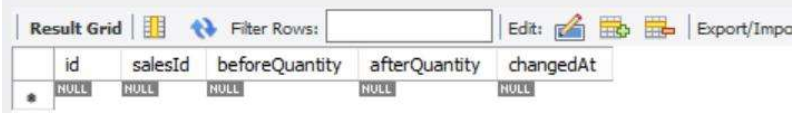
```
85    DELIMITER $$
86 •  CREATE TRIGGER before_sales_update
87    BEFORE UPDATE
88    ON sales FOR EACH ROW
89    BEGIN
90        DECLARE errorMessage VARCHAR(255);
91        SET errorMessage = CONCAT('The new quantity ',
92                            NEW.quantity,
93                            ' cannot be 3 times greater than the current quantity ',
94                            OLD.quantity);
95
96        IF new.quantity > old.quantity * 3 THEN
97            SIGNAL SQLSTATE '45000'
98                SET MESSAGE_TEXT = errorMessage;
99        END IF;
100   END $$
101   DELIMITER ;
102 •  UPDATE sales SET quantity = 150
103   WHERE id = 1;
104 •  SELECT * FROM sales;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | W

id	product	quantity	fiscalYear	fiscalMonth
1	2003 Harley-Davidson Eagle Drag Bike	150	2020	1
2	1969 Corvair Monza	150	2020	1
3	1970 Plymouth Hemi Cuda	200	2020	1
NULL	NULL	NULL	NULL	NULL

```
129   DELIMITER $$
130
131 •  CREATE TRIGGER after_sales_update
132   AFTER UPDATE
133   ON sales FOR EACH ROW
134   BEGIN
135       IF OLD.quantity <> new.quantity THEN
136           INSERT INTO SalesChanges(salesId,beforeQuantity, afterQuantity)
137           VALUES(old.id, old.quantity, new.quantity);
138       END IF;
139   END$$
140
141   DELIMITER ;
```

```
142 •  UPDATE Sales1 SET quantity = 350 WHERE id = 1;
143 •  SELECT * FROM SalesChanges;
```



id	salesId	beforeQuantity	afterQuantity	changedAt
NULL	NULL	NULL	NULL	NULL

PROGRAM 10

Procedures- Creation of Stored Procedures, Execution of Procedure, and Modification of Procedure

```
DELIMITER //

CREATE PROCEDURE GetAllProducts()
BEGIN
    SELECT *  FROM Product_Mast;
END //

DELIMITER ;
CALL GetAllProducts();
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Product	Company	Qty	Rate	Cost
Item1	Com1	2	10	20
Item2	Com2	3	25	75
Item3	Com1	2	30	60
Item4	Com3	5	10	50
Item5	Com2	2	20	40

```
DELIMITER $$

CREATE PROCEDURE GetOrderAmount()
BEGIN
    SELECT
        SUM(Qty * Rate)
    FROM Product_Mast;
END$$

DELIMITER ;
call GetOrderAmount();
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

SUM(Qty * Rate)
670

PROGRAM 11

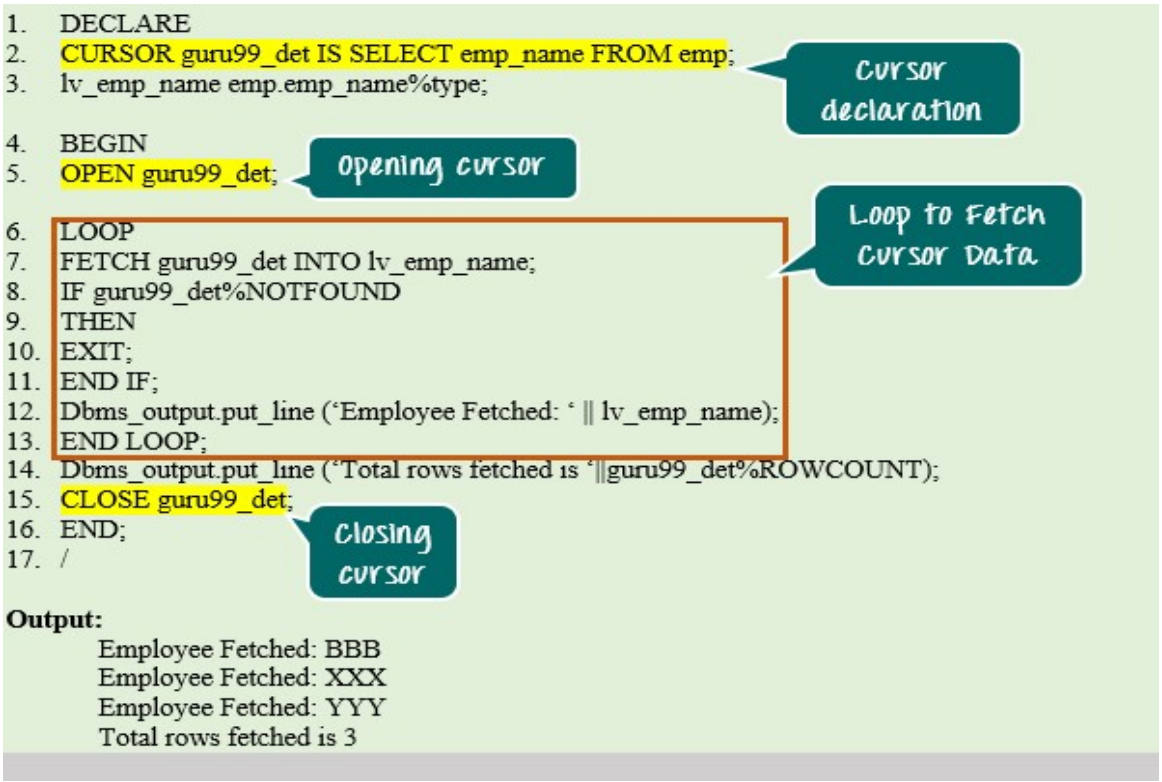
Cursors*- *Declaring Cursor, Opening Cursor, Fetching the data, closing the cursor.

```
DECLARE
CURSOR guru99_det IS SELECT emp_name FROM emp;
lv_emp_name emp.emp_name%type;

BEGIN
OPEN guru99_det;
LOOP
FETCH guru99_det INTO lv_emp_name;
IF guru99_det%NOTFOUND
THEN
EXIT;
END IF;
Dbms_output.put_line('Employee Fetched:'||lv_emp_name);
END LOOP;
Dbms_output.put_line('Total rows fetched is'||guru99_det%R0WCOUNT);
CLOSE guru99_det;
END:
/
```

OUTPUT

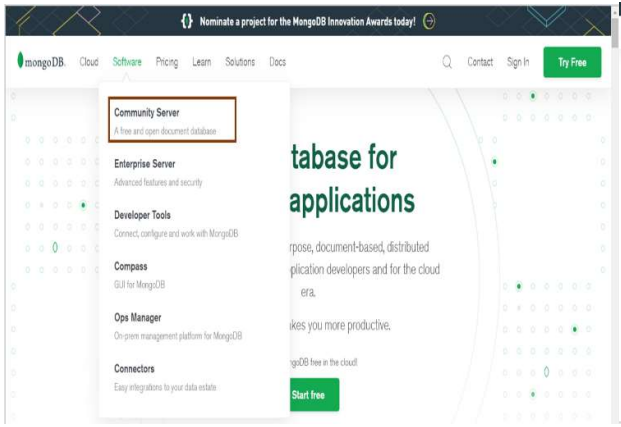
`

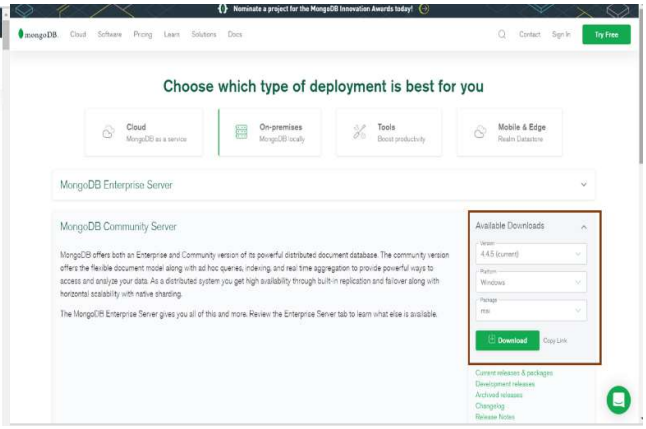


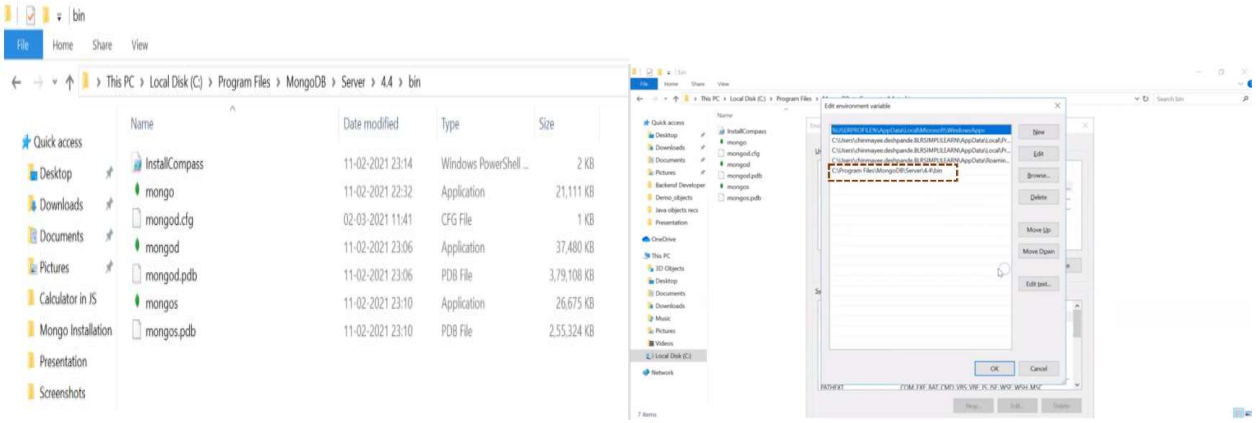
PROGRAM 12

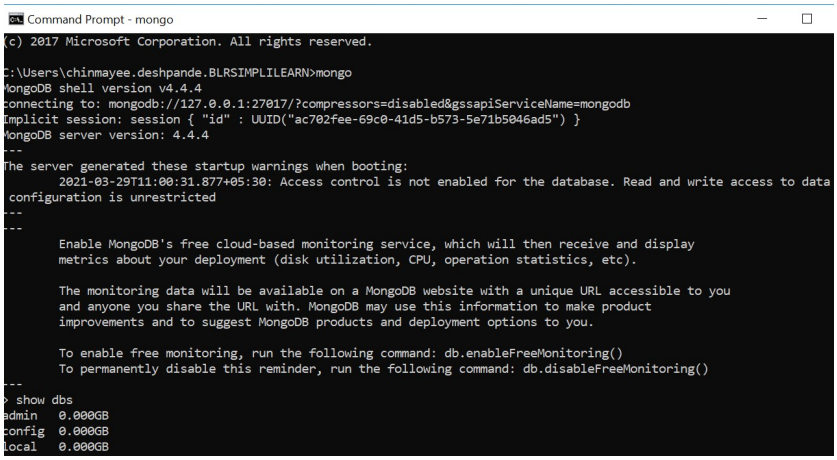
Study of Open Source NOSQL Database: MongoDB (Installation, Basic CRUD operations, Execution)

A Step-by-Step Guide to MongoDB Installation on Window









CRUD OPERATIONS

MongoDB Create Collection

```
>db.createCollection("SSSIT")
>show collections
```

```
SSSIT
```

DROP COLLECTION

```
>db.SSSIT.drop()
```

```
True
```

```
>show collections
```

```
System.indexes
```

MongoDB insert documents

```
db.javatpoint.insert(
 {
  course: "java",
  details: {duration: "6 months",  Trainer: "Sonoo jaiswal"  },
  Batch: [ { size: "Small", qty: 15 }, { size: "Medium", qty: 25 } ],
 category: "Programming language"})
 >db.javatpoint.find()
```

You will get the inserted document in return.

Output:

```
{ "_id" : ObjectId("56482d3e27e53d2dbc93cef8"), "course" :
"java", "details" :
{ "duration" : "6 months", "Trainer" : "Sonoo jaiswal" },
"Batch" :
[ {"size" : "Small", "qty" : 15 }, { "size" : "Medium", "qty"
: 25 } ],
 "category" : "Programming language" }
```

MongoDB update documents

```
db.javatpoint.insert(
 { course: "java",details: {duration: "6 months", Trainer: "Sonoo jaiswal"},
  Batch: [ { size: "Small", qty: 15 }, { size: "Medium", qty: 25 } ],
  category: "Programming language"})
```

>db.javatpoint.find()

Output:

```
{ "_id" : ObjectId("56482d3e27e53d2dbc93cef8"), "course" :
"java", "details" :
{ "duration" : "6 months", "Trainer" : "Sonoo jaiswal" },
"Batch" :
[ {"size" : "Small", "qty" : 15 }, { "size" : "Medium", "qty"
: 25 } ],
 "category" : "Programming language" }
```

Update the existing course "java" into "android":

>db.javatpoint.update({'course':'java'},{$set:{'course':'android'}})

Check the updated document in the collection:

>db.javatpoint.find()

Output:

```
{ "_id" : ObjectId("56482d3e27e53d2dbc93cef8"), "course" :
"android", "details" :
{ "duration" : "6 months", "Trainer" : "Sonoo jaiswal" },
"Batch" :
[ {"size" : "Small", "qty" : 15 }, { "size" : "Medium", "qty"
: 25 } ],
 "category" : "Programming language" }
```

MongoDB Delete documents

Remove all documents

SYNTAX:-

db.javatpoint.remove({})

Remove all documents that match a condition

SYNTAX:-

db.javatpoint.remove({ type : "programming language" })

Remove a single document that match a condition

SYNTAX:-

db.javatpoint.remove({ type : "programming language" }, 1)

PROGRAM 13

Design and Develop MongoDB Queries using CRUD operations. (Use CRUD operations, SAVE method, logical operators)

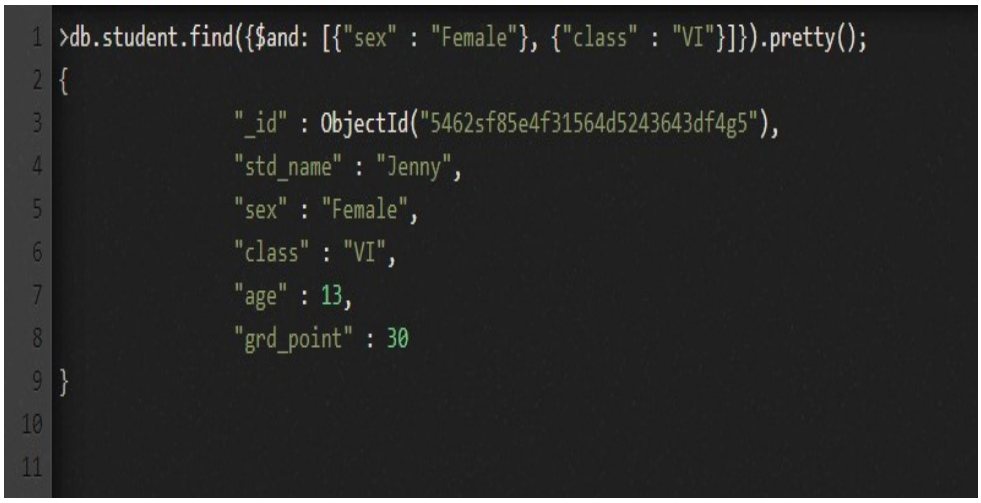
MongoDB $and Operator

Database: JavaTpoint
Collection: Student
Document: Six documents that contain the details **of** the students

```
[
  {
        "std_name" : "Jack",
        "sex" : "Male",
        "class" : "VI",
        "age" : 11,
        "grd_point" : 33
    },
    {
        "std_name" : "Jenny",
        "sex" : "Female",
        "class" : "VI",
        "age" : 13,
        "grd_point" : 30
    },
    {
        "std_name" : "Thomas",
        "sex" : "Male",
        "class" : "V",
        "age" : 11,
        "grd_point" : 35.1257
    },
    {
        "std_name" : "Lassy",
        "sex" : "Female",
        "class" : "X",
        "age" : 17,
        "grd_point" : 36.2514
    },
    {
        "std_name" : "Mia",
        "sex" : "Female",
        "class" : "X",
        "age" : 19,
        "grd_point" : 35.5201
}
    {
       "std_name" : "Mike,
        "sex" : "Male",
```

```
            "class" : "V",
            "age" : 16,
            "grd_point" : 35.5201
        }
]
```

>db.student.find({$and: [{"sex" : "Female"}, {"class" : "VI"}]}).pretty()



MongoDB $not Operator

Database: JavaTpoint
Collection: student
Document: Six documents that contain the details of the students

```
[
    {
        "std_name" : "Jack",
        "sex" : "Male",
        "class" : "VI",
        "age" : 11,
        "Total_marks" : 303
        "Result" : "Pass"
```

```
    },
    {
        "std_name" : "Jenny",
        "sex" : "Female",
        "class" : "VI",
        "age" : 13,
        "Total_marks" : 800
        "Result" : "Pass"
    },
    {
        "std_name" : "Thomas",
        "sex" : "Male",
        "class" : "V",
        "age" : 11,
        " Total_marks" : 200
        " Result" : "Fail"
    },
    {
        "std_name" : "Lassy",
        "sex" : "Female",
        "class" : "X",
        "age" : 17,
        "Total_marks" : 550
        "Result" : "Pass"
    },
    {
        "std_name" : "Mia",
        "sex" : "Female",
        "class" : "X",
        "age" : 19,
        " Total_marks" : 350
        "Result" : "Pass"
    },
    {
        "std_name" : "Mike,
        "sex" : "Male",
        "class" : "V",
        "age" : 16,
        "Total_marks" : 700
        "Result" : "Pass"
    }
]
```

```
db.student.find({"Total_marks" : {$not: {$lt : 400}}}).pretty()
```

OUTPUT

```
>db.student.find({"age" : {$not: {$gt : 12}}}).pretty()
{
                "_Id" : ObjectId("4565d4cd85d4f545ffg43df7"),
                "std_name" : "Jack",
                "sex" : "Male",
                "class" : "VI",
                "age" : 11,
                "Total_marks" : 303
                "Result" : "Pass"
}
{
                "_Id" : ObjectId("4565d4cd85d4f545ffg43df7"),
                "std_name" : "Thomas",
                "sex" : "Male",
                "class" : "V",
                "age" : 11,
                " Total_marks" : 200
                " Result" : "Fail"
}
```

MongoDB $or Operator

Database: JavaTpoint
Collection: student
Document: Five documents that contain the details of the students

```
>db.student.find()
{
        "_id" : ObjectId("56254d4fdf2222265r4g12ds3d65f"),
        "name" : "Mick",
        "Course" : "btech",
        "batch_year" : 2018,
        "language" : ["c++", "java", "python"],
        "personal_details" :   {
                "Father_name" : "Jonny",
                "phone_no" : 8895321456,
                "age" : 23,
                "gender" : "Male",
                "City" : "NewYork",
               }}
{       "_id" : ObjectId("56254d4fdf2222265r4g12ds3d691"),
        "name" : "Zoya",
```

```
        "Course" : "BCA",
        "batch_year" : 2020,
        "language" : ["C#", "JavaScript"],
        "personal_details" :
                {
                "Father_name" : "Henry",
                "phone_no" : 9874563698,
                "age" : 20,
                "gender" : "Female",
                "City" : "London",
                }}
{       "_id" : ObjectId("56254d4fdf2222265r4g12ds3d655"),
        "name" : "Jonny",
        "Course" : "MCA",
        "batch_year" : 2019,
        "language" : ["C#", "java", "PHP"],
        "personal_details" :
                {
                "Father_name" : "Thomas",
                "phone_no" : 7845123698,
                "age" : 24,
                "gender" : "Male",
                "City" : "London",
                }}
{         "_id" : ObjectId("56254d4fdf2222265r4g12ds3d678"),
        "name" : "Oliver",
        "Course" : "BA",
        "batch_year" : 2017,
        "language" : ["c", "PHP"],
        "personal_details" :
                {
                "Father_name" : "William","phone_no" : 9997845123, "age" : 25,
                "gender" : "Male",
                "City" : "Liverpool",
                }}
{
        "_id" : ObjectId("56254d4fdf2222265r4g12ds3d665"),
        "name" : "Mia",
        "Course" : "btech",
        "batch_year" : 2020,
        "language" : ["HTML", "CSS", "PHP"],
        "personal_details" :
                {
                "Father_name" : "Leo",
                "phone_no" : 6312547896,
                "age" : 22,
                "gender" : "Female",
                "City" : "Manchester",
                }
}
>db.student.find({$or: [{Course : "MCA"}, {batch_year : 2018}]}).pretty()
```

```
>db.sutdent.find({$or: [{Course : "MCA"}, {batch_year : 2018}]}).pretty()
{
          "_id" : ObjectId("56254d4fdf2222265r4g12ds3d65f"),
          "name" : "Mick",
          "Coruse" : "btech",
          "batch_year" : 2018,
          "language" : ["c++", "java", "python"],
          "personal_details" :
                    {
                     "Father_name" : "Jonny",
                     "phone_no" : 8895321456,
                     "age" : 23,
                     "gender" : "Male",
                     "City" : "NewYork",
                    }
}
{
          "_id" : ObjectId("56254d4fdf2222265r4g12ds3d655"),
          "name" : "Jonny",
          "Coruse" : "MCA",
          "batch_year" : 2019,
          "language" : ["C#", "java", "PHP"],
          "personal_details" :
                    {
                     "Father_name" : "Thomas",
                     "phone_no" : 7845123698,
                     "age" : 24,
                     "gender" : "Male",
                     "City" : "London",
                    }
}
```

MongoDB $in Operator

Database: JavaTpoint
Collection: student
Document: Five documents that contain the details of the students
>db.student.find()
{ "_id" : ObjectId("56254d4fdf2222265r4g12ds3d65f"),
"name" : "Mick",
"Course" : "btech",
"batch_year" : 2018,
"language" : ["c++", "java", "python"],
"personal_details" :
{
"Father_name" : "Jonny",
"phone_no" : 8895321456,
"age" : 23,
"gender" : "Male",
"City" : "NewYork",
}}
{
"_id" : ObjectId("56254d4fdf2222265r4g12ds3d691"),
"name" : "Zoya",
"Course" : "BCA",
"batch_year" : 2020,
"language" : ["C#", "JavaScript"],
"personal_details" :
{
"Father_name" : "Henry",
"phone_no" : 9874563698,

```
                "age" : 20,
                "gender" : "Female",
                "City" : "London",
                }}
{
        "_id" : ObjectId("56254d4fdf2222265r4g12ds3d655"),
        "name" : "Jonny",
        "Course" : "MCA",
        "batch_year" : 2019,
        "language" : ["C#", "java", "PHP"],
        "personal_details" :
                {
                "Father_name" : "Thomas",
                "phone_no" : 7845123698, "age" : 24, "gender" : "Male", "City" : "London",
                }}
{
        "_id" : ObjectId("56254d4fdf2222265r4g12ds3d678"),
        "name" : "Oliver",
        "Course" : "BA",
        "batch_year" : 2017,
        "language" : ["c", "PHP"],
        "personal_details" :
                {
                "Father_name" : "William",
                "phone_no" : 9997845123,
                "age" : 25,
                "gender" : "Male",
                "City" : "Liverpool",
                }}
{
        "_id" : ObjectId("56254d4fdf2222265r4g12ds3d665"),
        "name" : "Mia", "Course" : "btech","batch_year" : 2020,  "language" : ["HTML", "CSS", "PHP"],
        "personal_details" :
                {
                "Father_name" : "Leo","phone_no" : 6312547896, "age" : 22, "gender" : "Female",
                "City" : "Manchester",
                }}
>db.student.find({name: {$in: ["Jonny", "Mia"]}}).pretty()
```

```
>db.student.find({name: {$in: ["Jonny", "Mia"]}}).pretty()
{
          "_id" : ObjectId("56254d4fdf2222265r4g12ds3d655"),
          "name" : "Jonny",
          "Course" : "MCA",
          "batch_year" : 2019,
          "language" : ["C#", "java", "PHP"],
          "personal_details" :
                    {
                     "Father_name" : "Thomas",
                     "phone_no" : 7845123698,
                     "age" : 24,
                     "gender" : "Male",
                     "City" : "London",
                    }
}
{
          "_id" : ObjectId("56254d4fdf2222265r4g12ds3d665"),
          "name" : "Mia",
          "Course" : "btech",
          "batch_year" : 2020,
          "language" : ["HTML", "CSS", "PHP"],
          "personal_details" :
                    {
                     "Father_name" : "Leo",
                     "phone_no" : 6312547896,
                     "age" : 22,
                     "gender" : "Female",
                     "City" : "Manchester",
                    }
}
```

MongoDB $nor Operator ($nor)

Database: JavaTpoint
Collection: student
Document: Five documents that contain the details **of** the students
>db.student.find()

```
{
        "_id" : ObjectId("56254d4fdf2222265r4g12ds3d65f"),
        "name" : "Mick",
        "Course" : "btech",
        "batch_year" : 2018,
        "language" : ["c++", "java", "python"],
        "personal_details" :
                {
                 "Father_name" : "Jonny",
                 "phone_no" : 8895321456,
                 "age" : 23,
                 "gender" : "Male",
                 "City" : "NewYork",
                }}
{
        "_id" : ObjectId("56254d4fdf2222265r4g12ds3d691"),
        "name" : "Zoya",
        "Course" : "BCA",
        "batch_year" : 2020,
        "language" : ["C#", "JavaScript"],
        "personal_details" :
                {
                 "Father_name" : "Henry",
```

```
            "phone_no" : 9874563698,
            "age" : 20,
            "gender" : "Female",
            "City" : "London",
           }}
{
        "_id" : ObjectId("56254d4fdf2222265r4g12ds3d655"),
        "name" : "Jonny",
        "Course" : "MCA",
        "batch_year" : 2019,
        "language" : ["C#", "java", "PHP"],
        "personal_details" :
            {
            "Father_name" : "Thomas",
            "phone_no" : 7845123698,
            "age" : 24,
            "gender" : "Male",
            "City" : "London",
           }}
{
        "_id" : ObjectId("56254d4fdf2222265r4g12ds3d678"),
        "name" : "Oliver",
        "Course" : "BA",
        "batch_year" : 2017,
        "language" : ["c", "PHP"],
        "personal_details" :
            {
            "Father_name" : "William","phone_no" : 9997845123,"age" : 25,
            "gender" : "Male",
            "City" : "Liverpool",
           }}
{
        "_id" : ObjectId("56254d4fdf2222265r4g12ds3d665"),
        "name" : "Mia",
        "Course" : "btech",
        "batch_year" : 2020,
        "language" : ["HTML", "CSS", "PHP"],
        "personal_details" :
            {
            "Father_name" : "Leo","phone_no" : 6312547896, "age" : 22,"gender" : "Female",
              "City" : "Manchester",
           }}
```

```
db.sutdent.find({$or: [{Course : "MCA"}, {batch_year : 2018}]}).pretty()
{
        "_id" : ObjectId("56254d4fdf2222265r4g12ds3d65f"),
        "name" : "Mick",
        "Coruse" : "btech",
        "batch_year" : 2018,
        "language" : ["c++", "java", "python"],
        "personal_details" :
                {
                 "Father_name" : "Jonny",
                 "phone_no" : 8895321456,
                 "age" : 23,
                 "gender" : "Male",
                 "City" : "NewYork",
                }
}
{
        "_id" : ObjectId("56254d4fdf2222265r4g12ds3d655"),
        "name" : "Jonny",
        "Coruse" : "MCA",
        "batch_year" : 2019,
        "language" : ["C#", "java", "PHP"],
        "personal_details" :
                {
                 "Father_name" : "Thomas",
                 "phone_no" : 7845123698,
                 "age" : 24,
                 "gender" : "Male",
                 "City" : "London",
                }
}
```

PROGRAM 14

Implement aggregation and indexing with suitable example using MongoDB.

Collection: students
Documents: Seven documents that contain the details of the students in the form of field-value pairs

.

```
C:\WINDOWS\system32\cmd.exe - mongo
> db.students.find().pretty()
{
        "_id" : ObjectId("6024aefbf54bd0745f0db733"),
        "name" : "tony",
        "age" : 17,
        "id" : 1,
        "sec" : "A",
        "subject" : [
                "physics",
                "maths"
        ]
}
{
        "_id" : ObjectId("6024aefbf54bd0745f0db734"),
        "name" : "steve",
        "age" : 37,
        "id" : 2,
        "sec" : "A"
}
{
        "_id" : ObjectId("6024aefbf54bd0745f0db735"),
        "name" : "natasha",
        "age" : 17,
        "id" : 3,
        "sec" : "B",
        "subject" : [
                "physics",
                "english"
        ]
}
{
        "_id" : ObjectId("6024aefbf54bd0745f0db736"),
        "name" : "bruce",
        "age" : 21,
        "id" : 4,
        "sec" : "B",
        "subject" : [
                "physics",
                "maths",
                "biology",
                "Chemistry"
        ]
}
{
        "_id" : ObjectId("6024aefbf54bd0745f0db737"),
        "name" : "nick",
        "age" : 40,
        "id" : 5,
        "sec" : "B",
        "subject" : [
                "english"
        ]
}
{
        "_id" : ObjectId("6024aefbf54bd0745f0db738"),
        "name" : "groot",
        "age" : 4,
        "id" : 6,
        "sec" : "A",
        "subject" : [
                "english"
        ]
}
{
        "_id" : ObjectId("6024aefbf54bd0745f0db739"),
        "name" : "thanos",
        "age" : 4,
        "id" : 7,
        "sec" : "A",
        "subject" : [
                "maths",
                "physics",
                "chemistry"
        ]
}
```

SUM(Displaying the total number of students in one section only**)**

db.students.aggregate([{$group:{_id:"$sec",total_st:{$sum:1},max_age:{$max:"$age"}}}])

```
> db.students.aggregate([{$match:{sec:"B"}},{$count:"Total student in sec:B"}])
{ "Total student in sec:B" : 3 }
>
```

Displaying the total number of students in both the sections and maximum age from both section

db.students.aggregate([{$group: {_id:"$sec", total_st: {$sum:1}, max_age:{$max:"$age"} } }])

```
> db.students.aggregate([{$group: {_id:"$sec", total_st: {$sum:1}, max_age:{$max:"$age"} } }])
{ "_id" : "A", "total_st" : 4, "max_age" : 37 }
{ "_id" : "B", "total_st" : 3, "max_age" : 40 }
>
```

Displaying details of students whose age is greater than 30 using match stage

db.students.aggregate([{$match:{age:{$gt:30}}}])

```
C:\WINDOWS\system32\cmd.exe - mongo
> db.students.aggregate([{$match:{age:{$gt:30}}}]) .pretty()
{
        "_id" : ObjectId("6024aefbf54bd0745f0db734"),
        "name" : "steve",
        "age" : 37,
        "id" : 2,
        "sec" : "A"
}
{
        "_id" : ObjectId("6024aefbf54bd0745f0db737"),
        "name" : "nick",
        "age" : 40,
        "id" : 5,
        "sec" : "B",
        "subject" : [
                "english"
        ]
}
```

PROGRAM 15

Mini project (Design & Development of Data and Application) for following: -