Assignment No. 1

Aim: 1. Design and Develop SQL DDL statements which demonstrate the use of SQL objects such as Table, View, Index, Sequence, Synonym, different constraints etc.

2. Write at least 10 SQL queries on the suitable database application using SQL DML statements.

Software and Hardware Requirements: 1. Intel i3,3.3GHz,4gb ram.

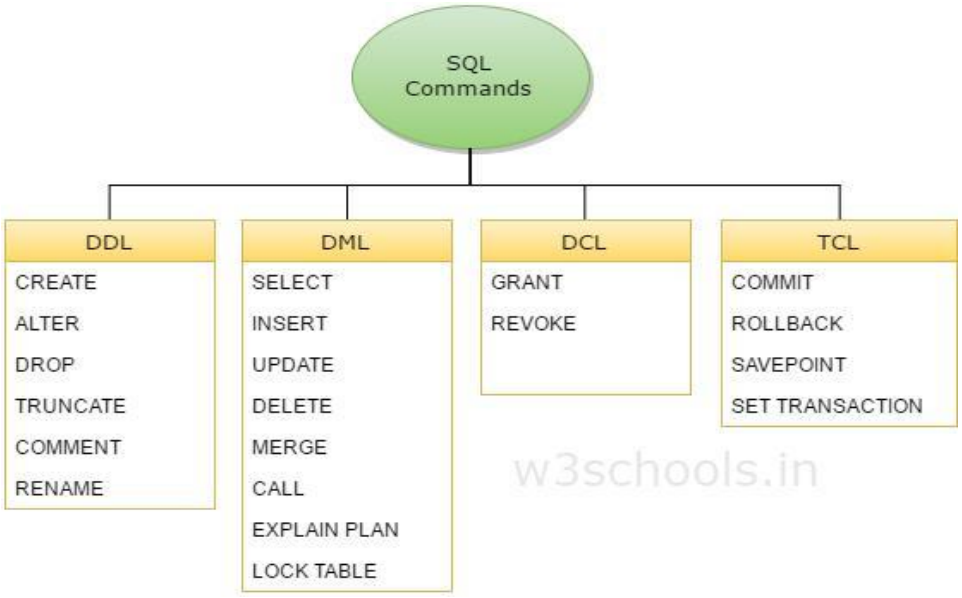
2.Linux OS.

3.Terminal (CLI).

4.MySQL

Theory:

DATA DEFINITION LANGUAGE (DDL) QUERIES



DDL : Data Definition Language (DDL) statements are used to define the database structure or schema.

Data Definition Language (DDL) are used different statements :

CREATE - to create objects in the database

ALTER - alters the structure of the database

DROP - delete objects from the database

TRUNCATE - remove all records from a table, including all spaces allocated for the records are removed

Create Database : From the MySQL command line, enter the command CREATE DATABASE <DATABASENAME>;

Replace <DATABASENAMEs> with the name of your database.

For example, to create a database pune city, you might enter

CREATE DATABASE pune;

Select your database : Once the database has been created, you will need to select it in order to begin editing it. Enter the command

USE pune;

You will see the message Database changed, letting you know that your active database is now pune.

To Display a list of your available databases: Enter the command

SHOW DATABASES;

Create table : We define an SQL relation by using the CREATE TABLE command. The following command creates a relation department in the database.

example:

CREATE TABLE department(dept name VARCHAR(20),building VARCHAR(15),budget INT(12));

Insert values in table : A newly created relation is empty initially. We can use the insert command to load data into the relation. For example, if we wish to insert the fact that there is an instructor named Smith in the Biology department with instructor id 10211 and a salary of $66,000, we write:

INSERT into instructor values (10211, 'Smith', 'Biology', 66000);

Drop table : To remove a relation from an SQL database, we use the drop table command. The drop table command deletes all information about the dropped relation from the database.

example: DROP TABLE <table_name>

Alter Table : We use the alter table command to add attributes to an existing relation. All tuples in the relation are assigned null as the value for the new attribute. The form of the alter table command is

alter table r add AD;

where r is the name of an existing relation, A is the name of the attribute to be added, and D is the type of the added attribute. We can drop attributes from a relation by the command

View : SQL allows a “virtual relation” to be defined by a query, and the relation conceptually contains the result of the query. The virtual relation is not precomputed and stored, but instead is computed by executing the query whenever the virtual relation is used. Any such relation that is not part of the logical model, but is made visible to a user as a virtual relation, is called a view. To create view we use following command :

create view <view_name> as <query_expression>;

where <query_expression> is any legal query expression. The view name is represented by v.

Create Index: A database index is a data structure that improves the speed of operations in a table. Indexes can be created using one or more columns, providing the basis for both rapid random lookups and efficient ordering of access to records.

Simple and Unique Index: You can create a unique index on a table. A unique index means that two rows cannot have the same index value. Here is the syntax to create an Index on a table.

CREATE UNIQUE INDEX index_name ON table_name (column1, column2,...);

Conclusion: In this assignment, we have studied and demonstrated various DDL statements in SQL.

```
// Output:
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| customers          |
| information_schema |
| mysql              |
| performance_schema |
| persinsinfo        |
| sakila             |
| sys                |
| teainds            |
| world              |
+--------------------+
9 rows in set (0.03 sec)
mysql> create database T3
    -> ;
Query OK, 1 row affected (0.04 sec)
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| customers          |
| information_schema |
| mysql              |
| performance_schema |
| persinsinfo        |
| sakila             |
| sys                |
```

```
| t3                 |
| teainds            |
| world              |
+--------------------+
10 rows in set (0.00 sec)
mysql> use t3
Database changed
mysql> create table persons (id int, firstname varchar(255), lastname
varchar(255), city varchar(255), primary key (id));
Query OK, 0 rows affected (0.02 sec)
mysql> show tables;
+--------------+
| Tables_in_t3 |
+--------------+
| persons      |
+--------------+
1 row in set (0.02 sec)
mysql> insert into persons values(1, 'NIlesh', 'sharma','Nashik');
Query OK, 1 row affected (0.01 sec)
mysql> insert into persons values(2, 'Anushka', 'Sharma','Mumbai');
Query OK, 1 row affected (0.00 sec)
mysql> insert into persons values(3, 'Gayatri', 'Jadhav','Chalisgaon');
Query OK, 1 row affected (0.01 sec)
mysq>linsert into persons values(3, 'Gayatri', 'Jadhav','Chalisgaon');
mysql> update persons set firstname='Aishwarya' where id=2;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
mysql> select * from persons;
+----+-----------+----------+------------+
| id | firstname | lastname | city       |
```

```
+----+-----------+----------+------------+
|  1 | NIlesh    | sharma   | Nashik     |
|  2 | Aishwarya | Sharma   | Mumbai     |
|  3 | Gayatri   | Jadhav   | Chalisgaon |
+----+-----------+----------+------------+
3 rows in set (0.00 sec)
mysql> delete from persons where id=1;
Query OK, 1 row affected (0.00 sec)
mysql> alter table persons add emailid varchar(255);
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0
mysql> describe persons;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| id        | int          | NO   | PRI | NULL    |       |
| firstname | varchar(255) | YES  |     | NULL    |       |
| lastname  | varchar(255) | YES  |     | NULL    |       |
| city      | varchar(255) | YES  |     | NULL    |       |
| emailid   | varchar(255) | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
mysql> alter table persons drop emailid;
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0
mysql> alter table persons rename column id to rollno
    -> ;
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0
mysql> describe persons;
```

```
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| rollno    | int          | NO   | PRI | NULL    |       |
| firstname | varchar(255) | YES  |     | NULL    |       |
| lastname  | varchar(255) | YES  |     | NULL    |       |
| city      | varchar(255) | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
mysql> truncate table persons;
Query OK, 0 rows affected (0.02 sec)
mysql> select * from persons;
Empty set (0.00 sec)
mysql> drop table persons;
Query OK, 0 rows affected (0.01 sec)
mysql> show tables;
Empty set (0.00 sec)
mysql> create table persons (id int, firstname varchar(255), lastname
varchar(255), city varchar(255), primary key (id));
Query OK, 0 rows affected (0.01 sec)
5 rows in set (0.00 sec)
mysql> alter table persons add emailid varchar(255);
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0
mysql> select * from persons;
+----+-----------+----------+------------+---------+
| id | firstname | lastname | city       | emailid |
+----+-----------+----------+------------+---------+
|  1 | NIlesh    | sharma   | Nashik     | NULL    |
|  2 | Anushka   | Sharma   | Mumbai     | NULL    |
```

```
|  3 | Sandip    | Kadus    | Ahamadnagr | NULL    |
|  4 | Dhanshree | Bhusare  | Dhamori    | NULL    |
|  5 | Durgesh   | Agrawal  | Manmad     | NULL    |
+----+-----------+----------+------------+---------+
5 rows in set (0.00 sec)
mysql> update persons set emailid="sharma.nilesh101@gmail.com"
where id=1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
mysql> select * from persons;
+----+-----------+----------+------------+--------------------------+
| id | firstname | lastname | city       | emailid                  |
+----+-----------+----------+------------+--------------------------+
|  1 | NIlesh    | sharma   | Nashik     | sharma.nilesh101@gmail.com |
|  2 | Anushka   | Sharma   | Mumbai     | NULL                     |
|  3 | Sandip    | Kadus    | Ahamadnagr | NULL                     |
|  4 | Dhanshree | Bhusare  | Dhamori    | NULL                     |
|  5 | Durgesh   | Agrawal  | Manmad     | NULL                     |
+----+-----------+----------+------------+--------------------------+
5 rows in set (0.00 sec)
mysql> select * from persons order by firstname
    -> ;
+----+-----------+----------+------------+--------------------------+
| id | firstname | lastname | city       | emailid                  |
+----+-----------+----------+------------+--------------------------+
|  2 | Anushka   | Sharma   | Mumbai     | NULL                     |
|  4 | Dhanshree | Bhusare  | Dhamori    | NULL                     |
|  5 | Durgesh   | Agrawal  | Manmad     | NULL                     |
|  1 | NIlesh    | sharma   | Nashik     | sharma.nilesh101@gmail.com |
|  3 | Sandip    | Kadus    | Ahamadnagr | NULL                     |
```

```
+----+-----------+----------+------------+--------------------------+
5 rows in set (0.00 sec)
mysql> select * from persons order by firstname DESC
    -> ;
+----+-----------+----------+------------+--------------------------+
| id | firstname | lastname | city       | emailid                  |
+----+-----------+----------+------------+--------------------------+
|  3 | Sandip    | Kadus    | Ahamadnagr | NULL                     |
|  1 | NIlesh    | sharma   | Nashik     | sharma.nilesh101@gmail.com |
|  5 | Durgesh   | Agrawal  | Manmad     | NULL                     |
|  4 | Dhanshree | Bhusare  | Dhamori    | NULL                     |
|  2 | Anushka   | Sharma   | Mumbai     | NULL                     |
+----+-----------+----------+------------+--------------------------+
5 rows in set (0.00 sec)
mysql> select * from persons order by firstname;
+----+-----------+----------+------------+--------------------------+
| id | firstname | lastname | city       | emailid                  |
+----+-----------+----------+------------+--------------------------+
|  2 | Anushka   | Sharma   | Mumbai     | NULL                     |
|  4 | Dhanshree | Bhusare  | Dhamori    | NULL                     |
|  5 | Durgesh   | Agrawal  | Manmad     | NULL                     |
|  1 | NIlesh    | sharma   | Nashik     | sharma.nilesh101@gmail.com |
|  3 | Sandip    | Kadus    | Ahamadnagr | NULL                     |
+----+-----------+----------+------------+--------------------------+
5 rows in set (0.00 sec)
mysql> select * from persons order by id DESC;
+----+-----------+----------+------------+--------------------------+
| id | firstname | lastname | city       | emailid                  |
+----+-----------+----------+------------+--------------------------+
|  5 | Durgesh   | Agrawal  | Manmad     | NULL                     |
```

```
|  4 | Dhanshree | Bhusare  | Dhamori    | NULL                      |
|  3 | Sandip    | Kadus    | Ahamadnagr | NULL                      |
|  2 | Anushka   | Sharma   | Mumbai     | NULL                      |
|  1 | NIlesh    | sharma   | Nashik     | sharma.nilesh101@gmail.com |
+----+-----------+----------+------------+---------------------------+
5 rows in set (0.00 sec)
mysql> create index i1 on persons (id, firstname);
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0
mysql> help
mysql> help auto_increment
Name: 'AUTO_INCREMENT'
Description:
The AUTO_INCREMENT attribute can be used to generate a unique identity
for new rows:
Examples:
CREATE TABLE animals (
     id MEDIUMINT NOT NULL AUTO_INCREMENT,
     name CHAR(30) NOT NULL,
     PRIMARY KEY (id)
);
INSERT INTO animals (name) VALUES
    ('dog'),('cat'),('penguin'),
    ('lax'),('whale'),('ostrich');
SELECT * FROM animals;
mysql> alter table persons modify column id int AUTO_INCREMENT;
Query OK, 5 rows affected (1.51 sec)
Records: 5  Duplicates: 0  Warnings: 0
mysql> select * from persons;
+----+-----------+----------+------------+---------------------------+
```

```
| id | firstname | lastname | city       | emailid                   |
+----+-----------+----------+------------+---------------------------+
|  1 | NIlesh    | sharma   | Nashik     | sharma.nilesh101@gmail.com |
|  2 | Anushka   | Sharma   | Mumbai     | NULL                      |
|  3 | Sandip    | Kadus    | Ahamadnagr | NULL                      |
|  4 | Dhanshree | Bhusare  | Dhamori    | NULL                      |
|  5 | Durgesh   | Agrawal  | Manmad     | NULL                      |
+----+-----------+----------+------------+---------------------------+
5 rows in set (0.00 sec)

mysql> INSERT INTO Persons (FirstName,LastName)
values('rohit','sharma');

Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO Persons (FirstName,LastName)
values('rohit','sharma');

CREATE TABLE Orders (
    OrderID int NOT NULL,
    OrderNumber int NOT NULL,
    PersonID int,
    PRIMARY KEY (OrderID),
    FOREIGN KEY (PersonID) REFERENCES Persons(id)
);

mysql> create table persons (id int, firstname varchar(255), lastname
varchar(255), city varchar(255), primary key (id));

Query OK, 0 rows affected (0.02 sec)

mysql> insert into persons values(3, 'Gayatri', 'Jadhav','Chalisgaon');

Query OK, 1 row affected (0.00 sec)

mysql> select * from persons;

+----+-----------+----------+------------+
| id | firstname | lastname | city       |
+----+-----------+----------+------------+
|  1 | NIlesh    | sharma   | Nashik     |
```

```
|  2 | Anushka   | Sharma   | Mumbai     |
|  3 | Gayatri   | Jadhav   | Chalisgaon |
+----+-----------+----------+------------+
3 rows in set (0.00 sec)
mysql> CREATE TABLE Orders (
    ->     OrderID int NOT NULL,
    ->     OrderNumber int NOT NULL,
    ->     PersonID int,
    ->     PRIMARY KEY (OrderID),
    ->     FOREIGN KEY (PersonID) REFERENCES Persons(id)
    -> );
Query OK, 0 rows affected (0.02 sec)
mysql> insert into orders values (1,22546,2);
mysql> select * from orders;
+---------+-------------+----------+
| OrderID | OrderNumber | PersonID |
+---------+-------------+----------+
|       1 |       22546 |        2 |
|       2 |       22546 |        1 |
|       3 |       22546 |        3 |
+---------+-------------+----------+
3 rows in set (0.00 sec)
mysql> select firstname , OrderID, OrderNumber from persons inner join
orders on persons.id=orders.personid
    -> ;
+-----------+---------+-------------+
| firstname | OrderID | OrderNumber |
+-----------+---------+-------------+
| NIlesh    |       2 |       22546 |
| Anushka   |       1 |       12345 |
```

```
| Gayatri   |      3 |      53214 |
+-----------+--------+------------+
3 rows in set (0.00 sec)
mysql> select firstname , OrderID, OrderNumber from persons inner join
orders on persons.id=orders.personid
   -> ;
+-----------+---------+-------------+
| firstname | OrderID | OrderNumber |
+-----------+---------+-------------+
| NIlesh    |       2 |       22546 |
| Anushka   |       1 |       12345 |
| Gayatri   |       3 |       53214 |
+-----------+---------+-------------+
3 rows in set (0.00 sec)
mysql> select * from persons inner join orders on
persons.id=orders.personid;
+----+-----------+----------+-----------+---------+-------------+----------+
| id | firstname | lastname | city      | OrderID | OrderNumber | PersonID |
+----+-----------+----------+-----------+---------+-------------+----------+
|  1 | NIlesh    | sharma   | Nashik    |       2 |       22546 |        1 |
|  2 | Anushka   | Sharma   | Mumbai    |       1 |       12345 |        2 |
|  3 | Gayatri   | Jadhav   | Chalisgaon |      3 |       53214 |        3 |
+----+-----------+----------+-----------+---------+-------------+----------+
3 rows in set (0.00 sec)
mysql> create view v1 as select firstname, city from persons;
Query OK, 0 rows affected (0.00 sec)
mysql> select * from v1;
+-----------+-----------+
| firstname | city      |
+-----------+-----------+
| NIlesh    | Nashik    |
```

| Anushka | Mumbai |

| Gayatri | Chalisgaon |

+-----------+------------+

3 rows in set (0.00 sec)

Assignment No. 2

Aim: Design at least 10 SQL queries for suitable database application using SQL DML statements: all types of Join, Sub-Query and View.

Software and Hardware Requirements: 1. Intel i3,3.3GHz,4gb ram.

2.Linux OS.

3.Terminal (CLI).

4.MySQL

Theory:

Introduction to Joins: An SQL JOIN clause is used to combine rows from two or more tables, based on a common field between them.

Types of Join:

1. JOIN: Return rows when there is at least one match in both tables.

SQL JOIN Syntax:

SELECT column_name(s)

FROM table_name1,table_name2

WHERE table_name1.column_name=table_name2.column_name

2. LEFT JOIN: Return all rows from the left table, even if there are no matches in the right table.

SQL LEFT JOIN Syntax:

SELECT column_name(s)

FROM table_name1

LEFT JOIN table_name2

ON table_name1.column_name=table_name2.column_name

3. RIGHT JOIN: Return all rows from the right table, even if there are no matches in the left table.

SQL RIGHT JOIN Syntax:

SELECT column_name(s)

FROM table_name1

RIGHT JOIN table_name2

ON table_name1.column_name=table_name2.column_name

4. FULL JOIN: Return rows when there is a match in one of the tables

SQL FULL JOIN Syntax:

SELECT column_name(s)

FROM table_name1

FULL JOIN table_name2

ON table_name1.column_name=table_name2.column_name

Types of Subqueries

Single Row Sub Query: Sub query which returns single row output. They mark the usage of single row comparison operators, when used in WHERE conditions.

Multiple row sub query: Sub query returning multiple row output. They make use of multiple row comparison operators like IN, ANY, ALL. There can be sub queries returning multiple columns also.

Correlated Sub Query: Correlated subqueries depend on data provided by the outer query.Thistype of subquery also includes subqueries that use the EXISTS operator to test the existence of data rows satisfying specified criteria.

Example:

Emp-id	Ename	City	Post	Salary
1	John	Nashik	Clerk	5000
2	Seema	Aurangabad	Developer	20000
3	Amita	Nagar	Manager	70000
4	Rakesh	Pune	Analyst	50000
5	Samata	Nashik	Tester	35000
6	Ankita	Chandwad	Developer	30000
7	Bhavika	Pune	Team-LR	50000
8	Deepa	Mumbai	CEO	90000
9	Nitin	Nagpur	Clerk	8000
10	Pooja	Pune	Analyst	45000

Display the information of employees, paid more than ‘pooja' from emptable

Select *from emp where salary > (select salaryfrom empwhere name=‘Pooja’) ;

Output of Above Query

Emp-id	Ename	City	Post	Salary
3	Amita	Nagar	Manager	70000
4	Rakesh	Pune	Analyst	50000
7	Bhavika	Pune	Team-LR	50000
8	Deepa	Mumbai	CEO	90000

MySQL Subqueries -Multiple rows with ALL, ANY, IN operator

1. [> ALL] More than the highest value returned by the subquery
2. [< ALL] Less than the lowest value returned by the subquery
3. [< ANY] Less than the highest value returned by the subquery
4. [> ANY] More than the lowest value returned by the subquery
5. [= ANY] Equal to any value returned by the subquery (same as IN)

All Example-

Display the employee name, salary and department no of those employees whose salary

is higher than all developers' salary.

SELECT Ename, salary, deptnoFROM EMPWHERE salary > All (SELECT salaryFROM emp Where post='Developer');

Ename	Salary	deptno
Amita	70000	20
Bhavika	50000	30
Deepa	90000	10
Pooja	45000	20

Output of Above Query

Conculsion: In this assignment, we designed at least 10 SQL queries for suitable database application using SQL DML statements: all types of Join, Sub-Query and View.

//OUTPUT: SQL JOIN

mysql> use database_name

mysql> create table stud_info(Rno int, Name varchar(20), Address varchar(20));

mysql> create table stud_marks(Rno int, DBMS int, TOC int, CN int);

mysql> insert into stud_info values (1,'Abhay','Nashik'), (2,'Sarika','Pune'),

(3,'Riya','Nashik'), (4,'Sachin','Manmad');

mysql> insert into stud_marks

values(1,50,60,55),(2,68,57,76),(3,45,76,70),(5,80,75,85);

mysql> select * from stud_info;

| Rno | Name | Address |

| 1 | Abhay | Nashik |

| 2 | Sarika | Pune |

| 3 | Riya | Nashik |

| 4 | Sachin | Manmad |

mysql> select * from stud_marks;

| Rno | DBMS | TOC | CN |

| 1 | 50 | 60 | 55 |

| 2 | 68 | 57 | 76 |

| 3 | 45 | 76 | 70 |

| 5 | 80 | 75 | 85 |

mysql> select stud_info.Rno, Name,DBMS,TOC,CN from stud_info inner join stud_marks

on stud_info.Rno=stud_marks.Rno;

| Rno | Name | DBMS | TOC | CN |

| 1 | Abhay | 50 | 60 | 55 |

| 2 | Sarika | 68 | 57 | 76 |

| 3 | Riya | 45 | 76 | 70 |

3 rows in set (0.00 sec)

mysql> select stud_info.Rno, Name,DBMS,TOC,CN from stud_info left join stud_marks on

stud_info.Rno=stud_marks.Rno;

| Rno | Name | DBMS | TOC | CN |

| 1 | Abhay | 50 | 60 | 55 |

| 2 | Sarika | 68 | 57 | 76 |

| 3 | Riya | 45 | 76 | 70 |

| 4 | Sachin | NULL | NULL | NULL |

4 rows in set (0.00 sec)

mysql> select stud_info.Rno, Name,DBMS,TOC,CN from stud_info right join stud_marks

on stud_info.Rno=stud_marks.Rno;

| Rno | Name | DBMS | TOC | CN |

| 1 | Abhay | 50 | 60 | 55 |

| 2 | Sarika | 68 | 57 | 76 |

| 3 | Riya | 45 | 76 | 70 |

| NULL | NULL | 80 | 75 | 85 |

4 rows in set (0.00 sec)

mysql> select stud_marks.Rno, Name,DBMS,TOC,CN from stud_info right join

stud_marks on stud_info.Rno=stud_marks.Rno;

| Rno | Name | DBMS | TOC | CN |

| 1 | Abhay | 50 | 60 | 55 |

| 2 | Sarika | 68 | 57 | 76 |

| 3 | Riya | 45 | 76 | 70 |

| 5 | NULL | 80 | 75 | 85 |

4 rows in set (0.00 sec)

//Sub queries.

// Create table EMP

mysql> create table Emp(Eid int, Ename varchar(20), City varchar(20), Post

varchar(15),Salary int, deptno int);

//Insert 10 Rows in the same

mysql> insert into Emp values

(1,'John','Nashik','Clerk',5000,10),

(2,'Seema','Aurangabad','Developer',20000,20),

(3,'Amita','Nagar','Manager',70000,20),

(4,'Rakesh','Pune','Analyst',8000,10),

(5,'Samata','Nashik','Tester',20000,10),

(6,'Anita','Chandwad','Developer',30000,30),

(7,'Bhavika','Pune','Team-LR',50000,30),

(8,'Deepa','Mumbai','CEO',90000,10),

(9,'Nitin','Nagpur','Clerk',8000,30),

(10,'Pooja','Pune','Analyst',45000,20);

mysql> select * from Emp;

| Eid | Ename | City| Post| Salary | deptno |

| 1 | John | Nashik| Clerk| 5000 | 10 |

| 2 | Seema | Aurangabad | Developer | 20000 | 20 |

| 3 | Amita | Nagar| Manager | 70000 | 20 |

| 4 | Rakesh | Pune| Analyst | 8000 | 10 |

| 5 | Samata | Nashik| Tester | 20000 | 10 |

| 6 | Anita | Chandwad | Developer | 30000 | 30 |

| 7 | Bhavika | Pune| Team-LR | 50000 | 30 |

| 8 | Deepa | Mumbai| CEO| 90000 | 10 |

| 9 | Nitin | Nagpur| Clerk | 8000 | 30 |

| 10 | Pooja | Pune| Analyst | 45000 | 20 |

//Display the information of employees, paid more than ‘pooja' from emp table

mysql> select * from Emp where salary>(select Salary from Emp where Ename='Pooja');

+------+---------+--------+---------+--------+--------+

| Eid | Ename | City | Post | Salary | deptno |

+------+---------+--------+---------+--------+--------+

| 3 | Amita | Nagar | Manager | 70000 |20 |

| 7 | Bhavika | Pune | Team-LR | 50000 |30 |

| 8 | Deepa | Mumbai | CEO | 90000 | 10 |

//List the name of the employees, who live in the same city as of ‘Rakesh’

mysql> select * from Emp where City=(select City from Emp where Ename='Rakesh');

+------+---------+------+---------+--------+--------+

| Eid | Ename | City | Post | Salary | deptno |

+------+---------+------+---------+--------+--------+

| 4 | Rakesh | Pune | Analyst | 8000 | 10 |

| 7 | Bhavika | Pune | Team-LR | 50000 | 30 |

| 10 | Pooja | Pune | Analyst | 45000 | 20 |

// Display the information of employees, paid greater salary than average salary throughout the company.

mysql> select * from Emp where Salary>=(select avg(Salary) from Emp);

+------+---------+--------+---------+--------+--------+

| Eid | Ename | City | Post | Salary | deptno |

+------+---------+--------+---------+--------+--------+

| 3 | Amita | Nagar | Manager | 70000 | 20 |

| 7 | Bhavika | Pune | Team-LR | 50000 | 30 |

| 8 | Deepa | Mumbai | CEO| 90000 | 10 |

| 10 | Pooja | Pune | Analyst | 45000 | 20 |

// Display the information of employees, paid less salary than average salary throughout the company.

mysql> select * from Emp where Salary<(select avg(Salary) from Emp);

+------+--------+-----------+----------+--------+--------+

| Eid | Ename | City

| Post

| Salary | deptno |

+------+--------+-----------+----------+--------+--------+

| 1 | John | Nashik | Clerk | 5000 | 10 |

| 2 | Seema | Aurangabad | Developer | 20000 | 20 |

| 4 | Rakesh | Pune| Analyst | 8000 |10 |

| 5 | Samata | Nashik| Tester | 20000 |10 |

| 6 | Anita | Chandwad | Developer | 30000 | 30 |

| 9 | Nitin | Nagpur | Clerk | 8000 | 30 |

//Display the information of employees having maximum salary in company

mysql> select * from Emp where Salary=(select max(Salary) from Emp);

+------+-------+--------+------+--------+--------+

| Eid | Ename | City | Post | Salary | deptno |

+------+-------+--------+------+--------+--------+

| 8 | Deepa | Mumbai | CEO | 90000 | 10 |

//Display the information of employees having minimum salary in company

mysql> select * from Emp where Salary=(select min(Salary) from Emp);

+------+-------+--------+-------+--------+--------+

| Eid | Ename | City | Post | Salary | deptno |

+------+-------+--------+-------+--------+--------+

| 1 | John | Nashik | Clerk | 5000 | 10 |

// IN Example- Display the employee name ,salary and department no of those

employees whose salary is the minimum salary of that department.

mysql> SELECT Ename, salary, deptno FROM EMP WHERE salary IN
(SELECT MIN(salary) FROM emp GROUP BY deptno);

+--------+--------+--------+

| Ename | salary | deptno |

+--------+--------+--------+

| John | 5000 | 10 |

| Seema | 20000 | 20 |

| Rakesh | 8000 | 10 |

| Samata | 20000 | 10 |

| Nitin | 8000 | 30 |

// All Example- Display the employee name, salary and department no of those employees whose salary is higher than all developers salary.

```
mysql> SELECT Ename, salary, deptno FROM EMP WHERE salary > All
( SELECT salary FROM emp Where post= 'Developer');

+---------+--------+--------+

| Ename | salary | deptno |

+---------+--------+--------+

| Amita | 70000 | 20 |

| Bhavika | 50000 | 30 |

| Deepa | 90000 | 10 |

| Pooja | 45000 | 20 |

+---------+--------+--------+

4 rows in set (0.00 sec)
```

// All Example- Display the employee name, salary and department no of those employees whose salary is lower than all developers salary

```
mysql> SELECT Ename, salary, deptno FROM EMP WHERE salary < All
( SELECT salary FROM emp Where post= 'Developer'

+--------+--------+--------+

| Ename | salary | deptno |

+--------+--------+--------+

| John | 5000 | 10 |

| Rakesh | 8000 | 10 |

| Nitin | 8000 | 30 |

+--------+--------+--------+

3 rows in set (0.00 sec)
```

//Any Example- Display the employee name, salary and department no of those employees whose salary is higher than salary of any developers salary

```
mysql> SELECT Ename, salary, deptno FROM EMP WHERE salary >any
( SELECT salary FROM emp Where post= 'Developer')

+---------+--------+--------+

| Ename | salary | deptno |

+---------+--------+--------+
```

| Amita | 70000 |20 |

| Anita | 30000 |30 |

| Bhavika | 50000 |30 |

| Deepa | 90000 |10 |

| Pooja | 45000 |20 |

+---------+--------+--------+

5 rows in set (0.00 sec)

//Any Example- Display the employee name, salary and department no of those employees whose salary is less than salary of developers salary.

mysql> SELECT Ename, salary, deptno FROM EMP WHERE salary <any (SELECT salary FROM emp Where post= 'Developer')

+--------+--------+--------+

| Ename | salary | deptno |

+--------+--------+--------+

| John | 5000 | 10 |

| Seema | 20000 | 20 |

| Rakesh | 8000 | 10 |

| Samata | 20000 | 10 |

| Nitin | 8000 | 30 |

+--------+--------+--------+

Assignment No. 3

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

Software and Hardware Requirements: 1. Intel i3,3.3GHz,4gb ram.

2.Linux OS.

3.Terminal (CLI).

4.MySQL

Theory:

What is NoSQL ?

NoSQL is a non-relational DBMS, that does not require a fixed schema, avoids joins, and is easy to scale. The purpose of using a NoSQL database is 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 collect terabytes of user data every single day. NoSQL database stands for "Not Only SQL" or "Not SQL." Though a better term would be "NoREL", NoSQL caught on. Carl Strozz introduced the NoSQL concept in 1998.

SQL	NOSQL
Relational Database management system	Distributed Database management system
Vertically Scalable	Horizontally Scalable
Fixed or predifined Schema	Dynamic Schema
Not suitable for hierarchical data storage	Best suitable for hierarchical data storage
Can be used for complex queries	Not good for complex queries

MongoDB

Scalable High-Performance Open-source, Document-orientated database.

- Built for Speed
- Rich Document based queries for Easy readability.
- Full Index Support for High Performance.

• Replication and Failover for High Availability.

• Auto Sharding for Easy Scalability.

• Map / Reduce for Aggregation.

Advantages of MongoDB

•Schema less : Number of fields, content and size of the document can be differ from one

document to another.

•No complex joins

•Data is stored as JSON style

•Index on any attribute

•Replication and High availability

Mongo DB Terminologies for RDBMS concepts

RDMS	MongoDB
Database	Database
Table,View	Collection
Row	Document(Json,Bson)
Column	Field
Index	Index
Join	Embedded Document
Foreign Key	Reference
Partition	Shard

Basic Database Operations

use <database name>

switched to database provided with command

• db

To check currently selected database use the command db

• show dbs

Displays the list of databases

• db.dropDatabase()

To Drop the database

•db.createCollection (name)

To create collection

Ex:- db.createCollection(Stud)

• show collections

List out all names of collection in current database

•db.databasename.insert

•({Key : Value})

•Ex:- db.Stud.insert({{Name:”Jiya”})

In mongodb you don't need to create collection. MongoDB creates collection automatically, when you insert some document.

• db.collection.drop() Example:- db.Stud.drop()

MongoDB's db.collection.drop() is used to drop a collection from the database.

CRUD Operations:

• Insert

• Find

• Update

• Delete/Remove

CRUD Operations – Insert

The insert() Method:- To insert data into MongoDB collection, you need to use MongoDB's

insert() or save()method.

Syntax:

>db.COLLECTION_NAME.insert(document)

Example:

>db.stud.insert({name: “Jiya”, age:15})

CRUD Operations – Find

The find() Method- To display data from MongoDB collection. Displays all the documents in a non structured way.

Syntax:

db.COLLECTION_NAME.find()

CRUD Operations – Update

Syntax

db.CollectionName.update (

<query/Condition>,

<update with $set or $unset>,

{

upsert: <boolean>,

multi: <boolean>,

})

Examples:

1> Set age = 25 where id is 100, First Whole document is replaced where condition is matched and only one field is remained as age:25

db.stud.update(

{ _id: 100 },

{ age: 25})

CRUD Operations – Remove

1. Remove All Documents

syntax: db.inventory.remove({})

2. Remove All Documents that Match a Condition

syntax: db.inventory.remove

({ type : "food" })

3. Remove a Single Document that Matches a Condition

syntax: db.inventory.remove

({ type : "food" }, 1)

Conclusion: In this assignment, we Designed and Developed MongoDB Queries using CRUD operations.

```
//Output:
> show dbs
admin  (empty)
local  0.078GB
> use admin
switched to db admin
> db
admin
> db.dropDatabase()
{ "dropped" : "admin", "ok" : 1 }
> db.createCollection('stud')
{ "ok" : 1 }
>  show collections
stud
system.indexes
> db.emp.insert({rno:1,name:'Bhavana'})
WriteResult({ "nInserted" : 1 })
> db.emp.insert({name:'Amit',rno:2})
WriteResult({ "nInserted" : 1 })
> db.emp.insert({rno:3, email_id:'a@gmail.com'})
WriteResult({ "nInserted" : 1 })
> db.emp.find()
{ "_id" : ObjectId("6321a6f722267f027ba09604"), "rno" : 1, "name" :
"Bhavana" }
{ "_id" : ObjectId("6321a70922267f027ba09605"), "name" : "Amit", "rno" :
2 }
{ "_id" : ObjectId("6321a71422267f027ba09606"), "rno" : 3, "email_id" :
"a@gmail.com" }
> db.emp.insert({_id:1,rno:4,name:"Akash"})
WriteResult({ "nInserted" : 1 })
> db.emp.find()
```

```
{ "_id" : ObjectId("6321a6f722267f027ba09604"), "rno" : 1, "name" : "Bhavana" }

{ "_id" : ObjectId("6321a70922267f027ba09605"), "name" : "Amit", "rno" : 2 }

{ "_id" : ObjectId("6321a71422267f027ba09606"), "rno" : 3, "email_id" : "a@gmail.com" }

{ "_id" : 1, "rno" : 4, "name" : "Akash" }

>  db.emp.insert({_id:1,rno:5,name:"Reena"})

WriteResult({

   "nInserted" : 0,

   "writeError" : {

                "code" : 11000,

                "errmsg" : "insertDocument :: caused by :: 11000 E11000 duplicate key error index: admin.emp.$_id_  dup key: { : 1.0 }"

   }

})

> E11000 duplicate key error index: db1.emp.$_id_  dup key: { : 1.0 }

2022-09-14T15:36:35.402+0530 SyntaxError: Unexpected identifier

> db.emp.insert ([{rno:7,name:'a'},{rno:8,name:'b'},{rno:8,name:'c' } ] )

BulkWriteResult({

   "writeErrors" : [ ],

   "writeConcernErrors" : [ ],

   "nInserted" : 3,

   "nUpserted" : 0,

   "nMatched" : 0,

   "nModified" : 0,

   "nRemoved" : 0,

   "upserted" : [ ]

})

>
db.emp.insert({rno:10,name:'Ankit',hobbies:['singing','cricket','swimming']
,age:21})
```

```
WriteResult({ "nInserted" : 1 })
>
> db.emp.insert({rno:11, Name: {Fname:"Bhavana", Mname:"Amit", Lname:"Khivsara"}})
WriteResult({ "nInserted" : 1 })
> db.emp.insert({rno:12, Name: "Janvi", Address:{Flat:501, Building:"Sai Appart", area:"Tidke colony", city: "Nashik", state:"MH", pin:423101}, age:22})
WriteResult({ "nInserted" : 1 })
> db.emp.insert({rno:15, name:'Ravina', dob: ISODate("2019-09-14")})
WriteResult({ "nInserted" : 1 })
> db.emp.insert(
... {rno:17, name:"Ashika",
... date:Date(),
... awards:[{
... name:"Best c -Designer", year:2010, prize:"winner"},
... {name:"Wen site competition",year:2012,prize:"Runner-up"},
... {name:"Fashion show", year:2015,prize:"winner"
...             }],
...  city:"Nashik"})
WriteResult({ "nInserted" : 1 })
> db.emp.find().pretty()
{
    "_id" : ObjectId("6321a6f722267f027ba09604"),
    "rno" : 1,
    "name" : "Bhavana"
}
{ "_id" : ObjectId("6321a70922267f027ba09605"), "name" : "Amit", "rno" : 2 }
{
    "_id" : ObjectId("6321a71422267f027ba09606"),
    "rno" : 3,
```

```
    "email_id" : "a@gmail.com"
}
{ "_id" : 1, "rno" : 4, "name" : "Akash" }
{ "_id" : ObjectId("6321a7b822267f027ba09607"), "rno" : 7, "name" : "a" }
{ "_id" : ObjectId("6321a7b822267f027ba09608"), "rno" : 8, "name" : "b" }
{ "_id" : ObjectId("6321a7b822267f027ba09609"), "rno" : 8, "name" : "c" }
{
    "_id" : ObjectId("6321a7c422267f027ba0960a"),
    "rno" : 10,
    "name" : "Ankit",
    "hobbies" : [
                "singing",
                "cricket",
                "swimming"
    ],
    "age" : 21
}
{
    "_id" : ObjectId("6321a7d722267f027ba0960b"),
    "rno" : 11,
    "Name" : {
                "Fname" : "Bhavana",
                "Mname" : "Amit",
                "Lname" : "Khivsara"
    }
}
{
    "_id" : ObjectId("6321a7e022267f027ba0960c"),
    "rno" : 12,
    "Name" : "Janvi",
```

```
    "Address" : {
                "Flat" : 501,
                "Building" : "Sai Appart",
                "area" : "Tidke colony",
                "city" : "Nashik",
                "state" : "MH",
                "pin" : 423101
    },
    "age" : 22
}
{
    "_id" : ObjectId("6321a7e922267f027ba0960d"),
    "rno" : 15,
    "name" : "Ravina",
    "dob" : ISODate("2019-09-14T00:00:00Z")
}
{
    "_id" : ObjectId("6321a7fe22267f027ba0960e"),
    "rno" : 17,
    "name" : "Ashika",
    "date" : "Wed Sep 14 2022 15:37:58 GMT+0530 (IST)",
    "awards" : [
                {
                 "name" : "Best c -Designer",
                 "year" : 2010,
                 "prize" : "winner"
                },
                {
                 "name" : "Wen site competition",
                 "year" : 2012,
```

```
                "prize" : "Runner-up"
               },
               {
                "name" : "Fashion show",
                "year" : 2015,
                "prize" : "winner"
               }
    ],
    "city" : "Nashik"
}
> db.stud.insert([{rno:1, name:'Ashiti'}, {rno:2,name:'Savita'},
{rno:3,name:'Sagar'}, {rno:4,name:'Reena'},{rno:5,name:'Jivan'}])
BulkWriteResult({
    "writeErrors" : [ ],
    "writeConcernErrors" : [ ],
    "nInserted" : 5,
    "nUpserted" : 0,
    "nMatched" : 0,
    "nModified" : 0,
    "nRemoved" : 0,
    "upserted" : [ ]
})
> db.stud.find()
{ "_id" : ObjectId("6321a81a22267f027ba0960f"), "rno" : 1, "name" :
"Ashiti" }
{ "_id" : ObjectId("6321a81a22267f027ba09610"), "rno" : 2, "name" :
"Savita" }
{ "_id" : ObjectId("6321a81a22267f027ba09611"), "rno" : 3, "name" :
"Sagar" }
{ "_id" : ObjectId("6321a81a22267f027ba09612"), "rno" : 4, "name" :
"Reena" }
```

{ "_id" : ObjectId("6321a81a22267f027ba09613"), "rno" : 5, "name" : "Jivan" }

> db.stud.find({rno:5})

{ "_id" : ObjectId("6321a81a22267f027ba09613"), "rno" : 5, "name" : "Jivan" }

> db.stud.find({rno:5},{name:1})

{ "_id" : ObjectId("6321a81a22267f027ba09613"), "name" : "Jivan" }

> { "_id" : ObjectId("5d83af5aa44331f62bcd836d"), "name" : "Jivan" }

>

> db.stud.find({rno:5},{name:1,_id:0})

{ "name" : "Jivan" }

> { "name" : "Jivan" }

2022-09-14T15:39:21.534+0530 SyntaxError: Unexpected token :

> db.stud.find({rno:5},{name:1,_id:0})

{ "name" : "Jivan" }

> { "name" : "Jivan" }

2022-09-14T15:40:35.559+0530 SyntaxError: Unexpected token :

> db.stud.find({rno:4},{name:,id:0}){"name":"Akash"}

2022-09-14T15:41:37.704+0530 SyntaxError: Unexpected token ,

> db.stud.find({},{name:1,_id:0})

{ "name" : "Ashiti" }

{ "name" : "Savita" }

{ "name" : "Sagar" }

{ "name" : "Reena" }

{ "name" : "Jivan" }

> { "name" : "Ashiti" }

> db.stud.find({rno:{$gt:2}})

{ "_id" : ObjectId("6321a81a22267f027ba09611"), "rno" : 3, "name" : "Sagar" }

{ "_id" : ObjectId("6321a81a22267f027ba09612"), "rno" : 4, "name" : "Reena" }

{ "_id" : ObjectId("6321a81a22267f027ba09613"), "rno" : 5, "name" : "Jivan" }

> db.stud.find({rno:{$lte:2}})

{ "_id" : ObjectId("6321a81a22267f027ba0960f"), "rno" : 1, "name" : "Ashiti" }

{ "_id" : ObjectId("6321a81a22267f027ba09610"), "rno" : 2, "name" : "Savita" }

> db.stud.find({rno:{$ne:2}})

{ "_id" : ObjectId("6321a81a22267f027ba0960f"), "rno" : 1, "name" : "Ashiti" }

{ "_id" : ObjectId("6321a81a22267f027ba09611"), "rno" : 3, "name" : "Sagar" }

{ "_id" : ObjectId("6321a81a22267f027ba09612"), "rno" : 4, "name" : "Reena" }

{ "_id" : ObjectId("6321a81a22267f027ba09613"), "rno" : 5, "name" : "Jivan" }

> db.stud.find({rno:{$in:[1,3,5]}})

{ "_id" : ObjectId("6321a81a22267f027ba0960f"), "rno" : 1, "name" : "Ashiti" }

{ "_id" : ObjectId("6321a81a22267f027ba09611"), "rno" : 3, "name" : "Sagar" }

{ "_id" : ObjectId("6321a81a22267f027ba09613"), "rno" : 5, "name" : "Jivan" }

> db.stud.find().sort({rno:-1})

{ "_id" : ObjectId("6321a81a22267f027ba09613"), "rno" : 5, "name" : "Jivan" }

{ "_id" : ObjectId("6321a81a22267f027ba09612"), "rno" : 4, "name" : "Reena" }

{ "_id" : ObjectId("6321a81a22267f027ba09611"), "rno" : 3, "name" : "Sagar" }

{ "_id" : ObjectId("6321a81a22267f027ba09610"), "rno" : 2, "name" : "Savita" }

{ "_id" : ObjectId("6321a81a22267f027ba0960f"), "rno" : 1, "name" : "Ashiti" }

> db.stud.find().sort({name:1})

{ "_id" : ObjectId("6321a81a22267f027ba0960f"), "rno" : 1, "name" : "Ashiti" }

{ "_id" : ObjectId("6321a81a22267f027ba09613"), "rno" : 5, "name" : "Jivan" }

{ "_id" : ObjectId("6321a81a22267f027ba09612"), "rno" : 4, "name" : "Reena" }

{ "_id" : ObjectId("6321a81a22267f027ba09611"), "rno" : 3, "name" : "Sagar" }

{ "_id" : ObjectId("6321a81a22267f027ba09610"), "rno" : 2, "name" : "Savita" }

> db.stud.find({rno:{$gt:2}},{_id:0}).sort({rno:-1})

{ "rno" : 5, "name" : "Jivan" }

{ "rno" : 4, "name" : "Reena" }

{ "rno" : 3, "name" : "Sagar" }

> db.stud.find()

{ "_id" : ObjectId("6321a81a22267f027ba0960f"), "rno" : 1, "name" : "Ashiti" }

{ "_id" : ObjectId("6321a81a22267f027ba09610"), "rno" : 2, "name" : "Savita" }

{ "_id" : ObjectId("6321a81a22267f027ba09611"), "rno" : 3, "name" : "Sagar" }

{ "_id" : ObjectId("6321a81a22267f027ba09612"), "rno" : 4, "name" : "Reena" }

{ "_id" : ObjectId("6321a81a22267f027ba09613"), "rno" : 5, "name" : "Jivan" }

> db.stud.distinct("rno")

[1, 2, 3, 4, 5]

> [1, 2, 3, 4, 5]

[1, 2, 3, 4, 5]

> db.stud.find().limit(2)

{ "_id" : ObjectId("6321a81a22267f027ba0960f"), "rno" : 1, "name" : "Ashiti" }

{ "_id" : ObjectId("6321a81a22267f027ba09610"), "rno" : 2, "name" : "Savita" }

```
> db.stud.find().skip(2)
{ "_id" : ObjectId("6321a81a22267f027ba09611"), "rno" : 3, "name" : "Sagar" }
{ "_id" : ObjectId("6321a81a22267f027ba09612"), "rno" : 4, "name" : "Reena" }
{ "_id" : ObjectId("6321a81a22267f027ba09613"), "rno" : 5, "name" : "Jivan" }
> db.stud.find({name:/^A/})
{ "_id" : ObjectId("6321a81a22267f027ba0960f"), "rno" : 1, "name" : "Ashiti" }
> db.stud.find({name:/i$/})
{ "_id" : ObjectId("6321a81a22267f027ba0960f"), "rno" : 1, "name" : "Ashiti" }
> db.stud.find({name:/a/})
{ "_id" : ObjectId("6321a81a22267f027ba09610"), "rno" : 2, "name" : "Savita" }
{ "_id" : ObjectId("6321a81a22267f027ba09611"), "rno" : 3, "name" : "Sagar" }
{ "_id" : ObjectId("6321a81a22267f027ba09612"), "rno" : 4, "name" : "Reena" }
{ "_id" : ObjectId("6321a81a22267f027ba09613"), "rno" : 5, "name" : "Jivan" }
> db.stud.findOne()
{
    "_id" : ObjectId("6321a81a22267f027ba0960f")
    "rno" : 1,
    "name" : "Ashiti"
}
> db.stud.find().count()
5
> db.stud.find({rno:{$gt:2}}).count()
3
>  db.stud.insert({rno:8,address:{area:"College Road",city:"Nashik",state:"MH"},name:"Arya"})
```

```
WriteResult({ "nInserted" : 1 })
> db.stud.find({"address.city":"Nashik"})
{ "_id" : ObjectId("6321a99722267f027ba09614"), "rno" : 8, "address" : { "area" : "College Road", "city" : "Nashik", "state" : "MH" }, "name" : "Arya" }
> db.stud.insert({rno:9,hobbies:['singing','dancing','cricket']})
WriteResult({ "nInserted" : 1 })
> db.stud.find({hobbies:'dancing'})
{ "_id" : ObjectId("6321a9a822267f027ba09615"), "rno" : 9, "hobbies" : [ "singing", "dancing", "cricket" ] }
> db.stud.update({rno:1},{$unset:{rno:1}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.stud.update({rno:2},{$set:{rno:22}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.stud.update({rno:50},{$set:{rno:55}},{upsert:true})
WriteResult({
    "nMatched" : 0,
    "nUpserted" : 1,
    "nModified" : 0,
    "_id" : ObjectId("6321a9c92cb29a58d3264181")
})
> //multi:true used to update in multiple documents
>  db.stud.update({rno:5},{$set:{rno:15}},{multiple:true})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.stud.remove({rno:4})
WriteResult({ "nRemoved" : 1 })
> db.stud.remove({rno:4},1)
WriteResult({ "nRemoved" : 0 })
> db.stud.remove({})
WriteResult({ "nRemoved" : 7 })
```

Assignment No. 4

Aim: Unnamed PL/SQL code block: Use of Control structure and Exception handling is mandatory.

Suggested Problem statement:

Consider Tables:

1. Borrower(Roll_no, Name, Date_of_Issue, NameofBook, Status)

2. Fine(Roll_no,Date,Amt)

• Accept roll_no & name of book from user.

• Check the number of days (from date of issue), if days are between 15 to 30 then fine amount will be Rs 5per day.

• If no. of days>;30, per day fine will be Rs 50 per day & for days less than 30, Rs. 5 per day.

• After submitting the book, status will change from I to R.

• If condition of fine is true, then details will be stored into fine table.

• Also handles the exception by named exception handler or user define exception handler.

Software and Hardware Requirements: 1. Intel i3,3.3GHz,4gb ram.

2.Linux OS.

3.Terminal (CLI).

4.MySQL

Theory:

PL/SQL Introduction

PL/SQL is a combination of SQL along with the procedural features of programming languages. Basic Syntax of PL/SQL which is a block-structured language; this means that the PL/SQL programs are divided and written in logical blocks of code. Each block consists of three sub-parts. Every PL/SQL statement ends with a semicolon (;).

Following is the basic structure of a PL/SQL block :

DECLARE <declarations_section>

BEGIN <executable_commands>

EXCEPTION <exception_handling>

END;

Sections	Description
Declarations	• This section starts with the keyword DECLARE. • It is an optional section and defines all variables, cursors, and other elements to be used in the program.
Executable Commands	• This section is enclosed between the keywords BEGIN and END and it is a mandatory section. • It consists of the executable PL/SQL statements of the program. • It should have at least one executable line of code.
Exception Handling	• This section starts with the keyword EXCEPTION. • This optional section contains exception(s) that handle errors in the program.

Anonymous blocks: Unnamed

Anonymous blocks are PL/SQL blocks which do not have any names assigned to them.

They need to be created and used in the same session because they will not be stored in the server as a database objects.

Since they need not to store in the database, they need no compilation steps.

They are written and executed directly, and compilation and execution happen in a single process.

Named blocks:

Named blocks are having a specific and unique name for them.They are stored as the database objects in the server. Since they are available as database objects, they can be referred to or used as long as it is present in the server.

Named blocks are basically of two types:

1. Procedure

2. Function

Stored Procedure Syntax:

CREATE PROCEDURE sp_name([proc_parameter: [IN | OUT | INOUT]

param_namedata_type])

Begin<Declare variable_namedata_type;>

<Control Statements/loops>

SQL executable statements;

End

Stored Procedure-Parameters

IN –is the default mode. When you define an IN parameter in a stored procedure, the calling program has to pass an argument to the stored procedure.

OUT – the value of an OUT parameter can be changed inside the stored procedure and its new value is passed back to the calling program

INOUT – an INOUT parameter is the combination of IN and OUT parameters. It means that the calling program may pass the argument, and the stored procedure can modify the INOUT parameter and pass the new value back to the calling program.

The IN parameter example :

Mysql> DELIMITER //

Mysql> CREATE PROCEDURE Allstud(IN SNameVARCHAR(25))

BEGIN

SELECT *FROM stud where Name=SName;

END

//

Mysql> DELIMITER ;

Mysql> call Allstud(‘Reena’);

The IN and OUT parameter example :

Mysql> CREATE PROCEDURE Allstud(IN Rno1 int,OUT SName VARCHAR)

BEGIN

SELECT Name into SName FROM stud where Rno=RNo1;

END

Mysql> call Allstud(2,@SName)//

Mysql> select @Sname

Date Related Functions required to solve the assignment

CURDATE() - The CURDATE() function returns the current date.

Note:This function returns the current date as a "YYYY-MM-DD" format

Example: SELECT CURDATE();

DATEDIFF() - The DATEDIFF() function returns the difference in days between two date values.

Syntax : DATEDIFF(date1, date2)

Example: SELECT DATEDIFF("2017-06-25", "2017-06-15");

Exception Handling :

Declaring a handler:

• To declare a handler, you use thestatement as follows:

• DECLARE action HANDLER FOR condition_value statement;

• If a condition whose value matches the condition_value, MySQL will execute the statement and continue or exit the current code block based on the action .

The action accepts one of the following values:

1. CONTINUE :the execution of the enclosing code block (BEGIN ... END) continues.

2. EXIT : the execution of the enclosing code block, where the handler is declared, terminates.

Conclusion: In this assignment, we studied and implemented PL/SQL block of code.

//OUTPUT:

```
mysql> use neha;
Database changed
mysql> show tables;
+---------------+
| Tables_in_neha |
+---------------+
| borrower      |
| fine          |
| persons       |
| student       |
| student1      |
+---------------+
5 rows in set (0.03 sec)
mysql> select * from borrower;
+---------+-----------+-------------+------------+-------+
| Roll_no | Name      | DateofoIssue | NameofBook | Satus |
+---------+-----------+-------------+------------+-------+
|       1 | Neha      | 2017-06-25   | Java       | I     |
|       2 | Shantanu  | 2017-07-10   | Networking | I     |
|       3 | Nandini   | 2017-05-22   | MySql      | I     |
|       4 | Vaishnavi | 2017-06-10   | DBMS       | I     |
|       5 | Aniket    | 2017-07-05   | WT         | I     |
|       6 | Komal     | 2017-06-30   | AI         | I     |
+---------+-----------+-------------+------------+-------+
6 rows in set (0.34 sec)
mysql> select * from fine;
Empty set (0.13 sec)
mysql>delimiter $
mysql>create procedure p1(In rno1 int(3),name1 varchar(30))
   ->begin
   ->Declare i_date date
   ->Declare diff int;
   ->select DateofoIssue into i_date from borrower where Roll_no=rno1 and
NameofBook=name1;
   ->SELECT DATEDIFF(CURDATE(),i_date)into diff;
   ->End;
   ->$
Query OK, 1 row affected (0.13 sec)
```

```
mysql> delimiter ;
mysql> call p1(1,'Java');
Query OK, 1 row affected (0.13 sec)
mysql>delimiter $
mysql>create procedure p1(In rno1 int(3),name1 varchar(30))
   ->begin
   ->Declare i_date date
   ->Declare diff int;
   ->select DateofoIssue into i_date from borrower where Roll_no=rno1 and
NameofBook=name1;
   ->SELECT DATEDIFF(CURDATE(),i_date)into diff;
   ->if diff>15 then
   ->Update borrower
   ->set satus='R'
   ->where Roll_no=rno1 and NameofBook=name1;
   ->End if;
   ->End;
   ->End;
   ->$
Query OK, 1 row affected (0.13 sec)
mysql> delimiter ;
mysql> call p1(1,'Java');
Query OK, 1 row affected (0.13 sec)
mysql> delimiter ;
mysql> call p1(2,'Networking');
Query OK, 1 row affected (0.13 sec)
mysql> select * from borrower;
+---------+----------+-------------+-----------+-------+
| Roll_no | Name     | DateofoIssue | NameofBook | Satus |
+---------+----------+-------------+-----------+-------+
|       1 | Neha     | 2017-06-25   | Java       | R     |
|       2 | Shantanu  | 2017-07-10   | Networking | R     |
|       3 | Nandini   | 2017-05-22   | MySql      | I     |
|       4 | Vaishnavi | 2017-06-10   | DBMS       | I     |
|       5 | Aniket    | 2017-07-05   | WT         | I     |
|       6 | Komal     | 2017-06-30   | AI         | I     |
+---------+----------+-------------+-----------+-------+
6 rows in set (0.34 sec)
mysql>delimiter ;
mysql>create procedure p1(In rno1 int(3),name1 varchar(30))
   ->begin
```

```
    ->Declare i_date date
    ->Declare diff int;
    ->Declare fine_amt int;
    ->select DateofoIssue into i_date from borrower where Roll_no=rno1 and
name=name1;
    ->SELECT DATEDIFF(CURDATE(),i_date)into diff;
    ->if(diff>=15 and diff<=30)then
    ->SET fine_amt=diff*5;
    ->insert into fine values(rno1,CURDATE(),fine_amt);
    ->elseif(diff>30)then
    ->SET fine_amt=diff*50;
    ->insert into fine values(rno1,CURDATE(),fine_amt);
    ->End if;
    ->Update borrower set status='R' where Roll_no=rno1 and name=name1;
    ->End;
Query OK, 1 row affected (0.13 sec)
mysql> delimiter ;
mysql> call P3(1,'Java');
Query OK, 1 row affected (0.00 sec)
mysql> select * from fine;
Empty set (0.00 sec)
```

Assignment No. 6

Aim: Design the Database Cursor. Implement all types: Implicit, Explicit, Cursor FOR Loop, and Parameterized Cursor.

Software and Hardware Requirements: 1. Intel i3,3.3GHz,4gb ram.

2.Linux OS.

3.Terminal (CLI).

4.MySQL

Theory:

A cursor is a pointer to this context area. PL/SQL controls the context area through a cursor. A cursor holds the rows (one or more) returned by a SQL statement. The set of rows the cursor holds is referred to as the active set.

Types of Cursors:

There are two types of cursor namely as,

1) Implicit cursor.

2) Explicit cursor.

Why do we need the Cursors?

SELECT statement should return only one row at a time in previous PL/SQL

programs. This is too restrictive in many applications.We use the idea of Cursor to handle the above problem.

Implicit Cursor:

Implicit cursors are automatically created by Oracle whenever an SQL statement is executed, when there is no explicit cursor for the statement. Programmers cannot control the implicit cursors and the information in it.

Whenever a DML statement (INSERT, UPDATE and DELETE) is issued, an implicit cursor is associated with this statement. For INSERT operations, the cursor holds the data that needs to be inserted. For UPDATE and DELETE operations, the cursor identifies the rows that would be affected.

In PL/SQL, you can refer to the most recent implicit cursor as the SQL cursor, which always has attributes such as %FOUND, %ISOPEN,

%NOTFOUND, and %ROWCOUNT.

The SQL cursor has additional attributes, %BULK_ROWCOUNT and

%BULK_EXCEPTIONS, designed for use with the FORALL statement. The following table provides the description of the most used attributes.

Sr No.	Attributes and Description
1	%FOUND Returns TRUE if an INSERT, UPDATE, or DELETE statement affected one or more rows or a SELECT INTO statement returned one or more rows. Otherwise, it returns FALSE.
2	%NOTFOUND The logical opposite of %FOUND. It returns TRUE if an INSERT, UPDATE, or DELETE statement affected no rows, or a SELECT INTO statement returned no rows. Otherwise, it returns FALSE.
3	%ISOPEN Always returns FALSE for implicit cursors, because Oracle closes the SQL cursor automatically after executing its associated SQL statement.
4	%ROWCOUNT Returns the number of rows affected by an INSERT, UPDATE, or DELETE statement, or returned by a SELECT INTO statement.

Any SQL cursor attribute will be accessed as sql%attribute_name as shown below in the example.

```
DECLARE
   total_rows number (2);
 BEGIN
UPDATE customers
SET salary = salary + 500;
   IF sql%notfound THEN dbms_output.put_line('no
      customers selected');
   ELSIF sql%found THEN
      total_rows := sql%rowcount;
      dbms_output.put_line( total_rows || ' customers selected '); END
   IF;
END;
```

Explicit Cursors:

Explicit cursors are programmer-defined cursors for gaining more control over the context area. An explicit cursor should be defined in the declaration section of the PL/SQL Block. It is created on a SELECT Statement which returns more than one row.

The syntax for creating an explicit cursor is,

DECLARE cursor_name CURSOR FOR select_statement;

Working with an explicit cursor includes the following 4 steps

1. Declaring the cursor for initializing the memory

2. Opening the cursor for allocating the memory

3. Fetching the cursor for retrieving the data

4. Closing the cursor to release the allocated memory

Cursors with Parameters:

1. We can pass parameters into a cursor and use them in the query.

2. We can only pass values to the cursor; and cannot pass values out of the cursor through parameters.

3. Only the data type of the parameter is defined, not its length.

4. Optionally, we can also give a default value for the parameter, which will take effect if no value is passed to the cursor.

Cursors with Parameters Example:

DECLARE

cur_emp (par_dept VARCHAR2) CURSOR FOR

SELECT ename, salary

FROM emp WHERE deptno = par_dept;

Open cur_emp(5);

Declaring variables in a Cursor:

In native SQL, the SELECT list may contain both columns and expressions. In PL/SQL, the SELECT list may contain PL/SQL variables, expressions, and even functions as well as host language bind variables (> PL/SQL 2.1).

DECLARE

project_bonus NUMBER := 1000;

Conclusion: In this assignment, we implemented the database Cursor.

//code:

Example 1.

PL/SQL Block :

```
set autoprint on;
set serveroutput on;
set verify off;
declare cursor cu1 is
select Roll,Name from Student;
cursor cu2 is
select Roll from CompDep;
rno int;
nm varchar(20); rno2 int;
begin
open cu1; open cu2;
loop
fetch cu1 into rno,nm; fetch cu2 into rno2;
exit when cu1%found = false; if rno2 <> rno then
insert into CompDep values(rno,nm); end if;
end loop; close cu1; close cu2;
end;
```

Output:

```
SQL> create table CompDep(Roll int,Name varchar(20));
Table created.
SQL> create table Student(Roll int,Name varchar(20));
Table created.
SQL> insert into Student values(1,'a');
1 row created.
SQL> insert into Student values(2,'b');
1 row created.
SQL> insert into Student values(3,'c');
```

1 row created.

SQL> insert into Student values(4,'d');

1 row created.

SQL> insert into CompDep values(2,'b');

1 row created.

SQL> insert into CompDep values(5,'e');

1 row created.

SQL> insert into CompDep values(6,'f');

1 row created.

SQL> @C:\Users\sitrc\assign6.txt // call PL/SQL block code created above

OR

SQL> delimiter /

PL/SQL procedure successfully completed

SQL> select * from CompDep;

ROLL NAME

2 b

e

f

a

b

c

d

7 rows selected.

—----------

Example 2.

mysql> use Vedant;

Reading table information for completion of table and column names

You can turn off this feature to get a quicker startup with -A Database changed

```
mysql>      createtable o_rollcall(roll_no   int,name     varchar(20),address
varchar(20));

Query OK, 0 rows affected (0.28 sec)

mysql>      createtable n_rollcall(roll_no   int,namevarchar(20),address
varchar(20));

Query OK, 0 rows affected (0.27 sec)

mysql> insert into o_rollcall values('1','Hitesh','Nandura');

Query OK, 1 row affected (0.05 sec)

mysql> insert into o_rollcall values('2','Piyush','MP');

Query OK, 1 row affected (0.06 sec)

mysql> insert into o_rollcall values('3','Ashley','Nsk');

Query OK, 1 row affected (0.05 sec)

mysql> insert into o_rollcall values('4','Kalpesh','Dhule');

Query OK, 1 row affected (0.05 sec)

mysql> insert into o_rollcall values('5','Abhi','Satara');

Query OK, 1 row affected (0.04 sec)

mysql> delimiter //

mysql> create procedure p3(in r1 int)

-> begin

-> declare r2 int;

-> declare exit_loop boolean;

-> declare c1 cursor for select roll_no from o_rollcall where roll_no>r1;

-> declare continue handler for not found set exit_loop=true;

-> open c1;

-> e_loop:loop

-> fetch c1 into r2;

-> if not exists(select * from n_rollcall where roll_no=r2)

-> then

-> insert into n_rollcall select * from o_rollcall where roll_no=r2;

-> end if;

-> if exit_loop
```

```
-> then
-> close c1;
-> leave e_loop;
-> end if;
-> end loop e_loop;-> end
-> //
Query OK, 0 rows affected (0.00 sec)
mysql> call p3(3);
-> //
Query OK, 0 rows affected (0.10 sec)
mysql> select * from n_rollcall;
-> //
+_______+_______+________+
| roll_no | name | address |
+_______+_______+________+
| 4 | Kalpesh | Dhule |
| 5 | Abhi |  Satara |
+_______+_______+________+
 2 rows in set (0.00 sec)
mysql> call p3(0);
-> //
Query OK, 0 rows affected (0.22 sec)
mysql> select * from n_rollcall;
-> //
+_______+_______+________+
| roll_no | name | address |
+_______+_______+________+
| 4 | Kalpesh | Dhule |
| 5 | Abhi | Satara |
| 1 | Hitesh | Nandura |
```

| 2 | Piyush | MP |

| 3 | Ashley | Nsk |

+_____+_____+_____+

5 rows in set (0.00 sec)

mysql> insert into o_rollcall values('6','Patil','Kolhapur');

-> //

Query OK, 1 row affected (0.04 sec)

mysql> call p3(4);

-> //

Query OK, 0 rows affected (0.05 sec)

mysql> select * from n_rollcall;

-> //

+_____+_____+_____+

| roll_no | name | address |

+_____+_____+_____+

| 4 | Kalpesh | Dhule |

| 5 | Abhi | Satara |

| 1 | Hitesh | Nandura |

| 2 | Piyush | MP |

| 3 | Ashley | Nsk |

| 6 | Patil | Kolhapur |

+_____+_____+_____+

6 rows in set (0.00 sec)

Assignment No. 7

Aim: Write a program to implement MySQL/Oracle database connectivity with any front end language to implement Database navigation operations (add, delete, edit etc.)

Software and Hardware Requirements: 1. Intel i3,3.3GHz,4gb ram.

2.Linux OS.

3.Terminal (CLI).

4.MySQL

Theory :

How to setup Php as frontend :

Step 1 : place the simple.php & config1.php file in the C:\xampp\htdocs\

in htdocs folder

Step 2: In http://localhost/phpmyadmin/

create the any database as per front end

(simple.php) form having first_name, last_name ,email_id, address etc. attributes.

Setp 3: now run the http://localhost/simple.php code

Conclusion: In this assignment, we wroye a program to implement MySQL/Oracle database connectivity with frontend language to implement Database navigation operations (add, delete, edit etc.)

Code:

config1.php

```
<?php
define('DB_SERVER', 'localhost'); // Name of Serevr
define('DB_USERNAME', 'root'); // User of Database
define('DB_PASSWORD', '');
// Password
define('DB_DATABASE', 'ndd'); // Datanase Name
$db =
mysqli_connect(DB_SERVER,DB_USERNAME,DB_PASSWORD,DB_DATABAS
E);
// Check connection
if (!$db)
{
die("Connection failed: " . mysqli_connect_error());
}
//echo "Connected successfully";
?>
```

simple.php

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Add Record Form</title>
</head>
<center> <br> <br> <h3>Registration Form </h3>
<form name="myForm" action="<?php
method="post" enctype="multipart/form-data" >
echo($_SERVER["PHP_SELF"]);?>"
<table>
```

```
<tr> <td> <label for="firstName">First Name:</label> </td> <td>
<input

type="text" name="first_name" id="firstName"> </td> <td> <input
type="submit"

name="add" value="search"></td> </tr>

<tr> <td> <label for="lastName">Last Name:</label>

type="text" name="last_name" id="lastName" > </td> </tr>

</td>

<td> <input

<tr> <td> <label for="emailAddress">Email Address:</label> </td>
<td> <input

type="text" name="email" id="emailAddress" > </td></tr>

<tr> <td> <label> Upload Image </label> </td> <td> <input
type="file"

name="fileToUpload" id="fileToUpload"> </td></tr>

</table>

<table>

</tr><td><input type="submit" name="add" value="Submit"> </td>
<td><input

type="submit" name="add" value="delete"></td> <td><input
type="submit"

name="add"

value="update"></td>

<td><input

type="submit"

name="add"

value="display"></td></tr>

</table>

</form>

</center>

<br>

<?php
```

```
include("config1.php"); /* file contain the database name user name and
connection
details */
if(isset($_POST['add']))
{
$first_name = mysqli_real_escape_string($db,$_POST['first_name']);
$last_name = mysqli_real_escape_string($db,$_POST['last_name']);
$email = mysqli_real_escape_string($db,$_POST['email']);
$ch = mysqli_real_escape_string($db,$_POST['add']);
$file =
@addslashes(file_get_contents($_FILES["fileToUpload"]["tmp_name"]));
switch ($ch)
{
case "display":
$sql = "SELECT * from persons";
$result = mysqli_query($db, $sql);
?>
<center><h2> Registration </h2> </center>
<center> <table border="2">
<thead>
<tr>
<th>First Name</th>
<th>Last Name </th>
<th>Email Id</th>
<th>Photo</th>
</tr>
</thead>
<tbody>
<?php if (mysqli_num_rows($result)>0)
{
while($row = mysqli_fetch_assoc($result))
```

```
{
$im= '<img
src="'.base64_encode($row['image']).'"
height="100" width="100"/>'; //store image in varibale
echo
"<tr><td>{$row['first_name']}</td><td>{$row['last_name']}</td><td
>{$row['email']}<
/td><td> $im</td></tr>\n";
}
}
else
{
echo '<tr><td colspan="4">No Rows Returned</td></tr>';
} ?>
</tbody>
</table>
<?php
break;
case "Submit":
$sql = "INSERT INTO persons ". "(first_name,last_name,email,image) ". "
VALUES('$first_name','$last_name','$email','$file')";
$result = mysqli_query($db, $sql);
if ($result !=0)
{
echo "Recored is inserted : $result";
}
break;
case "delete":
$sql = " delete from persons where first_name='$first_name' ";
$result = mysqli_query($db, $sql);
if ($result !=0)
```

```
{
echo "Recored is delete : $first_name ";
}
//echo "delete button click ";
break;
case "update":
$sql = "update persons set first_name='$first_name',
last_name='$last_name',email='$email',image='$file' where
first_name='$first_name' ";
$result = mysqli_query($db, $sql);
if ($result !=0)
{
echo "Recored is update : $first_name ";
}
break;
case "search":
$sql = "SELECT * from persons where first_name= '".$first_name."'";
$result = mysqli_query($db, $sql) ;
?>
<center><h2> Registration </h2> </center>
<center> <table border="2">
<thead>
<tr>
<th>First Name</th>
<th>Last Name </th>
<th>Email Id</th>
<th>Photo</th>
</tr>
</thead>
<tbody>
```

```
<?php if (mysqli_num_rows($result)!==0)
{
while($row = mysqli_fetch_assoc($result))
{
$im= '<img
src="'.base64_encode($row['image']).'"
height="100" width="100"/>'; //store image in varibale
echo
"<tr><td>{$row['first_name']}</td><td>{$row['last_name']}</td><td
>{$row['email']}<
/td><td> $im</td></tr>\n";
//$im= '<img
src="'.base64_encode($row['image']).'"
height="10" width="50"/>';
}
}
else
{
echo '<tr><td colspan="4">No Record Found ! </td></tr>';
}
?>
</tbody>
</table>
<?php
break;
default:
echo $ch;
}
} ?>
</body>
</html> <!-- end of form -->
```