ASSIGNMENT 1

Name: Manasi Dattu Hire

Roll no: 331070

Prn no: 22320015

Aim: Design any database with at least 3 entities and relationships between them. Apply DCL and DDL commands. Draw suitable ER/EER diagram for the system.

Objective

1. To learn ER diagram design
2. To design EER for one small application
3. Understand and execute DDL and DCL commands

Theory:-

Entity Relationship Diagram:

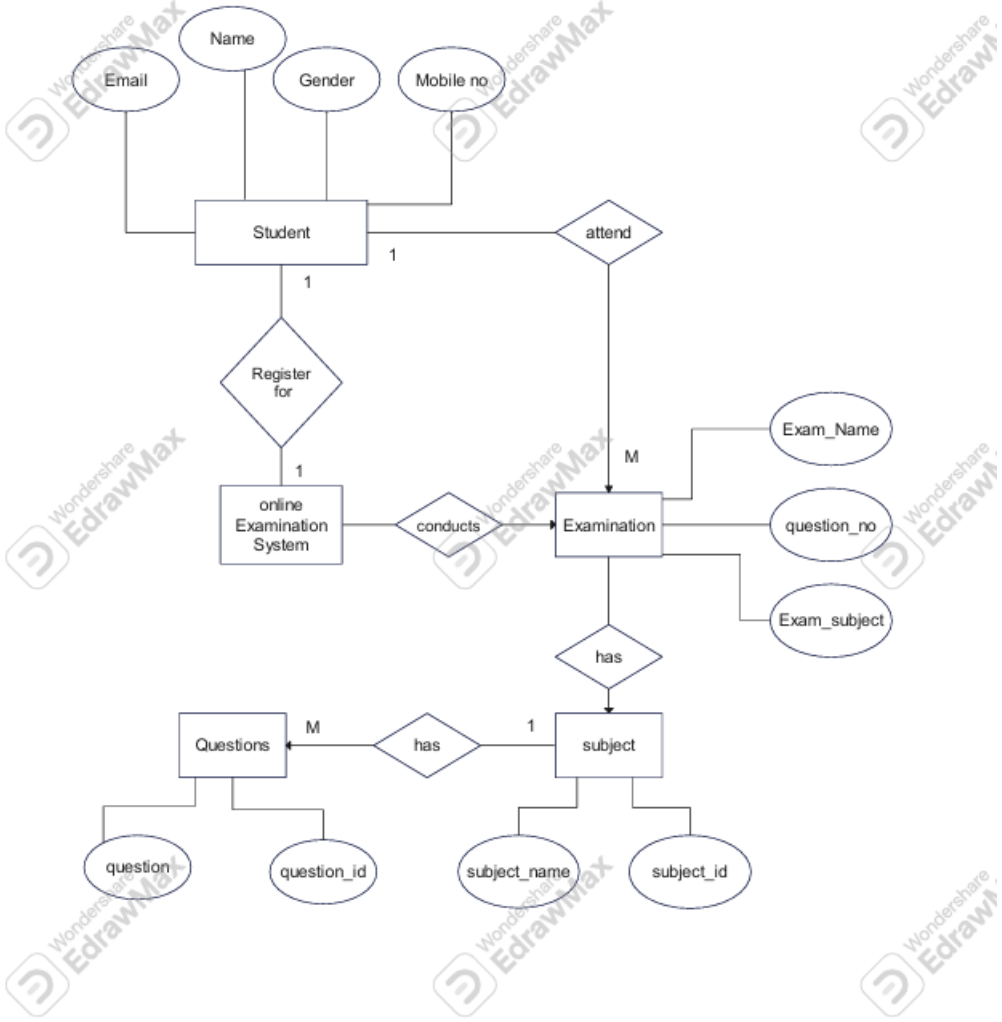
An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of the E-R model are: entity set and relationship set.

What is an ER Diagram?

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system. ER Diagrams are most often used to design or debug relational

databases in the fields of software engineering, business information systems, education and research.

ER diagram:



DDL Commands:

Data Definition Language(DDL) is a subset of SQL and a part of DBMS(Database Management System). DDL consist of Commands to commands like CREATE, ALTER, TRUNCATE and DROP. These commands are used to create or modify the tables in SQL.

DDL Commands:

1. Create
2. Alter
3. Rename
4. Drop

Create:

This command is used to create a new table in SQL. The user has to give information like table name, column names, and their data types.

Syntax –

```
CREATE TABLE table_name
(
column_1 datatype,
column_2 datatype,
column_3 datatype, ....
);
```

Alter:

This command is used to add, delete or change columns in the existing table. The user needs to know the existing table name and can add, delete or modify tasks easily.

Syntax -

ALTER TABLE table_name ADD column_name datatype;

Rename:

It is used to rename the existing name to another desired name.

Syntax -

RENAME table table_name to new_table_name

Drop:

This command is used to remove an existing table along with its structure from the Database.

Syntax: –

Syntax to drop an existing table. DROP TABLE table_name;

DCL Commands:

DCL includes commands such as GRANT and REVOKE which mainly deal with the rights, permissions, and other controls of the database system.

Grant:

This command gives users access privileges to the database.

Syntax:

GRANT privileges_names ON object TO user;

Revoke:

This command withdraws the user's access privileges given by using the Grant command.

Syntax -

REVOKE privileges ON object FROM user;

DDL Statements:

1. Student Table

```
CREATE TABLE Student (
   student_id INT PRIMARY KEY AUTO_INCREMENT,
   name VARCHAR(255),
   email VARCHAR(255),
   gender CHAR(1),
   mobile_no VARCHAR(15)
);
```

2. **Online_Examination_Center Table**

```
CREATE TABLE Online_Examination_Center (
   center_id INT PRIMARY KEY AUTO_INCREMENT,
   center_name VARCHAR(255)
);
```

3. Examination Table

```
CREATE TABLE Examination (
    exam_id INT PRIMARY KEY AUTO_INCREMENT,
    exam_name VARCHAR(255),
    exam_subject VARCHAR(255),
    question_no INT,
    center_id INT,
    FOREIGN KEY (center_id) REFERENCES
Online_Examination_Center(center_id)
);
```

4. Subject Table

```
CREATE TABLE Subject (
    subject_id INT PRIMARY KEY AUTO_INCREMENT,
    subject_name VARCHAR(255)
);
```

5. Questions Table

```
CREATE TABLE Questions (
    question_id INT PRIMARY KEY AUTO_INCREMENT,
    question TEXT,
    subject_id INT,
    FOREIGN KEY (subject_id) REFERENCES
Subject(subject_id)
);
```

6. Student_Attend_Examination Table

```
CREATE TABLE Student_Attend_Examination (
    student_id INT,
    exam_id INT,
    PRIMARY KEY (student_id, exam_id),
    FOREIGN KEY (student_id) REFERENCES
Student(student_id),
    FOREIGN KEY (exam_id) REFERENCES
Examination(exam_id)
);
```

7. Examination_Has_Subject Table

```
CREATE TABLE Examination_Has_Subject (
    exam_id INT,
    subject_id INT,
    PRIMARY KEY (exam_id, subject_id),
    FOREIGN KEY (exam_id) REFERENCES
Examination(exam_id),
    FOREIGN KEY (subject_id) REFERENCES
Subject(subject_id)
);
```

8. Register Table

```
CREATE TABLE Register (
    student_id INT,
    center_id INT,
    PRIMARY KEY (student_id, center_id),
    FOREIGN KEY (student_id) REFERENCES
Student(student_id),
    FOREIGN KEY (center_id) REFERENCES
Online_Examination_Center(center_id)
);
```

Conclusion:

We have studied and created a database with at least 3 entities and relationships between them with creation of ER/EER diagram for the system

ASSIGNMENT 2

Name: Manasi Dattu Hire

Roll no: 331070

Prn no: 22320015

Aim: Design and implement a database and apply at least 10 different DML queries for the following task. For a given input string display only those records which match the given pattern or a phrase in the search string. Make use of wild characters and LIKE operators for the same. Make use of Boolean and arithmetic operators wherever necessary

```
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 17
Server version: 8.0.39 MySQL Community Server - GPL

Copyright (c) 2000, 2024, 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> mysql -u root -p
    -> 1234
    -> show databases
    -> \c
mysql> CREATE DATABASE exam_db;
Query OK, 1 row affected (0.03 sec)

mysql> USE exam_db;
Database changed
mysql> CREATE TABLE Student (
    ->     student_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     name VARCHAR(255),
    ->     email VARCHAR(255),
    ->     gender CHAR(1),
    ->     mobile_no VARCHAR(15)
    -> );
Query OK, 0 rows affected (0.06 sec)
```

```
mysql> SHOW TABLES;
+------------------+
| Tables_in_exam_db |
+------------------+
| student          |
+------------------+
1 row in set (0.01 sec)

mysql> CREATE TABLE Online_Examination_Center (
    ->     center_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     center_name VARCHAR(255)
    -> );
Query OK, 0 rows affected (0.04 sec)

mysql> CREATE TABLE Examination (
    ->     exam_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     exam_name VARCHAR(255),
    ->     exam_subject VARCHAR(255),
    ->     question_no INT,
    ->     center_id INT,
    ->     FOREIGN KEY (center_id) REFERENCES Online_Examination_Center(center_id)
    -> );
Query OK, 0 rows affected (0.05 sec)

mysql> CREATE TABLE Subject (
    ->     subject_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     subject_name VARCHAR(255)
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql>
mysql> CREATE TABLE Questions (
    ->     question_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     question TEXT,
    ->     subject_id INT,
    ->     FOREIGN KEY (subject_id) REFERENCES Subject(subject_id)
    -> );
Query OK, 0 rows affected (0.04 sec)
```

```
mysql> CREATE TABLE Student_Attend_Examination (
    ->     student_id INT,
    ->     exam_id INT,
    ->     PRIMARY KEY (student_id, exam_id),
    ->     FOREIGN KEY (student_id) REFERENCES Student(student_id),
    ->     FOREIGN KEY (exam_id) REFERENCES Examination(exam_id)
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> CREATE TABLE Examination_Has_Subject (
    ->     exam_id INT,
    ->     subject_id INT,
    ->     PRIMARY KEY (exam_id, subject_id),
    ->     FOREIGN KEY (exam_id) REFERENCES Examination(exam_id),
    ->     FOREIGN KEY (subject_id) REFERENCES Subject(subject_id)
    -> );
Query OK, 0 rows affected (0.04 sec)

mysql>
mysql> CREATE TABLE Register (
    ->     student_id INT,
    ->     center_id INT,
    ->     PRIMARY KEY (student_id, center_id),
    ->     FOREIGN KEY (student_id) REFERENCES Student(student_id),
    ->     FOREIGN KEY (center_id) REFERENCES Online_Examination_Center(center_id)
    -> );
Query OK, 0 rows affected (0.04 sec)
```

```
mysql> show tables;
+----------------------------+
| Tables_in_exam_db          |
+----------------------------+
| examination                |
| examination_has_subject    |
| online_examination_center  |
| questions                  |
| register                   |
| student                    |
| student_attend_examination |
| subject                    |
+----------------------------+
8 rows in set (0.00 sec)

mysql> show tables;
```

```
mysql> INSERT INTO Student (name, email, gender, mobile_no)
    -> VALUES ('John Doe', 'john.doe@example.com', 'M', '1234567890');
Query OK, 1 row affected (0.03 sec)

mysql> INSERT INTO Online_Examination_Center (center_name)
    -> VALUES ('Pune Center');
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO Examination (exam_name, exam_subject, question_no, center_id)
    -> VALUES ('Mid-term Exam', 'Mathematics', 50, 1);
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO Subject (subject_name)
    -> VALUES ('Mathematics');
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO Questions (question, subject_id)
    -> VALUES ('What is 2+2?', 1);
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO Student_Attend_Examination (student_id, exam_id)
    -> VALUES (1, 1);
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO Examination_Has_Subject (exam_id, subject_id)
    -> VALUES (1, 1);
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO Register (student_id, center_id)
    -> VALUES (1, 1);
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM Student;
+------------+----------+----------------------+--------+------------+
| student_id | name     | email                | gender | mobile_no  |
+------------+----------+----------------------+--------+------------+
|          1 | John Doe | john.doe@example.com | M      | 1234567890 |
+------------+----------+----------------------+--------+------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT * FROM Student;
+------------+----------+----------------------+--------+------------+
| student_id | name     | email                | gender | mobile_no  |
+------------+----------+----------------------+--------+------------+
|          1 | John Doe | john.doe@example.com | M      | 1234567890 |
+------------+----------+----------------------+--------+------------+
1 row in set (0.00 sec)

mysql> SELECT e.exam_name, e.exam_subject
    -> FROM Student_Attend_Examination sae
    -> JOIN Examination e ON sae.exam_id = e.exam_id
    -> WHERE sae.student_id = 1;
+---------------+--------------+
| exam_name     | exam_subject |
+---------------+--------------+
| Mid-term Exam | Mathematics  |
+---------------+--------------+
1 row in set (0.00 sec)

mysql>
```

```
mysql> INSERT INTO Student_Attend_Examination (student_id, exam_id)
    -> VALUES (1, 1);
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO Examination_Has_Subject (exam_id, subject_id)
    -> VALUES (1, 1);
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO Register (student_id, center_id)
    -> VALUES (1, 1);
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM Student;
+------------+----------+----------------------+--------+------------+
| student_id | name     | email                | gender | mobile_no  |
+------------+----------+----------------------+--------+------------+
|          1 | John Doe | john.doe@example.com | M      | 1234567890 |
+------------+----------+----------------------+--------+------------+
1 row in set (0.00 sec)

mysql> SELECT e.exam_name, e.exam_subject
    -> FROM Student_Attend_Examination sae
    -> JOIN Examination e ON sae.exam_id = e.exam_id
    -> WHERE sae.student_id = 1;
+---------------+--------------+
| exam_name     | exam_subject |
+---------------+--------------+
| Mid-term Exam | Mathematics  |
+---------------+--------------+
1 row in set (0.00 sec)

mysql> UPDATE Student
    -> SET mobile_no = '0987654321'
    -> WHERE student_id = 1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql>
```

```
mysql> UPDATE Student
    -> SET mobile_no = '0987654321'
    -> WHERE student_id = 1;
Query OK, 0 rows affected (0.00 sec)
Rows matched: 1  Changed: 0  Warnings: 0

mysql> UPDATE Examination
    -> SET exam_name = 'Final Exam'
    -> WHERE exam_id = 1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
mysql> DELETE FROM Questions
    -> WHERE question_id = 1;
Query OK, 1 row affected (0.01 sec)

mysql> DELETE FROM Register
    -> WHERE student_id = 1 AND center_id = 1;
Query OK, 1 row affected (0.01 sec)
```

```
mysql> GRANT SELECT, INSERT ON online_exam_system.* TO 'exam_user'@'localhost';
ERROR 1410 (42000): You are not allowed to create a user with GRANT
mysql> GRANT SELECT, INSERT ON exam_db.* TO 'exam_user'@'localhost';
ERROR 1410 (42000): You are not allowed to create a user with GRANT
mysql> GRANT SELECT, INSERT ON exam_db.* TO 'root'@'localhost';
Query OK, 0 rows affected (0.01 sec)

mysql> GRANT ALL PRIVILEGES ON exam_db.* TO 'root'@'localhost';
Query OK, 0 rows affected (0.01 sec)

mysql> REVOKE INSERT ON exam_db.* FROM 'root'@'localhost';
Query OK, 0 rows affected (0.01 sec)

mysql> REVOKE ALL PRIVILEGES ON exam_db.* FROM 'root'@'localhost';
Query OK, 0 rows affected (0.01 sec)

mysql>
```

```
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO Student (name, email, gender, mobile_no) VALUES ('Jane Doe', 'jane.doe@example.com', 'F', '9876543210');
Query OK, 1 row affected (0.00 sec)

mysql> COMMIT;
Query OK, 0 rows affected (0.01 sec)

mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO Questions (question, subject_id) VALUES ('What is the capital of France?', 1);
Query OK, 1 row affected (0.00 sec)

mysql> ROLLBACK;
Query OK, 0 rows affected (0.00 sec)

mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO Student (name, email, gender, mobile_no) VALUES ('Mark Doe', 'mark.doe@example.com', 'M', '5647382910');
Query OK, 1 row affected (0.00 sec)

mysql> SAVEPOINT student_insert;
Query OK, 0 rows affected (0.00 sec)

mysql> ROLLBACK TO student_insert;
Query OK, 0 rows affected (0.00 sec)

mysql> RELEASE SAVEPOINT student_insert;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

Conclusion:

We successfully completed the assignment by designing the database and applying the different DML queries also we performed Boolean and arithmetic operators with the wildcard characters.

Assignment 3

Name: Manasi Dattu Hire

Roll no: 331070

Prn no: 22320015

Aim:

Execute the aggregate functions like count, sum, avg etc. on the suitable database. Make use of built in functions according to the need of the database chosen. Retrieve the data from thedatabase based on time and date functions like now (), date (), day () etc. Use group by and having clauses

1)AVG

The AVG() function returns the average value of a numeric column.

Syntax:

SELECT AVG(column_name)

FROM table_name

WHERE condition

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| exam_db            |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)

mysql> use exam_db;
Database changed
mysql> SELECT AVG(question_no) AS avg_questions_per_exam
    -> FROM Examination;
+------------------------+
| avg_questions_per_exam |
+------------------------+
|                   NULL |
+------------------------+
1 row in set (0.07 sec)
```

2)count()

The COUNT() function returns the number of rows that matches a specified criterion.

Syntax :

SELECT COUNT(column_name)

FROM table_name

WHERE condition

```
mysql> SELECT COUNT(student_id) AS total_students
    -> FROM Student;
+----------------+
| total_students |
+----------------+
|              0 |
+----------------+
1 row in set (0.02 sec)
```

3)min max

The MIN() function returns the smallest value of the selected column.

The MAX() function returns the largest value of the selected column.

Syntax :

SELECT MIN(column_name)

FROM table_name

WHERE condition;

```
mysql> SELECT MIN(question_no) AS min_questions, MAX(question_no) AS max_questions
    -> FROM Examination;
+---------------+---------------+
| min_questions | max_questions |
+---------------+---------------+
|          NULL |          NULL |
+---------------+---------------+
1 row in set (0.00 sec)
```

4)date

MySQL comes with the following data types for storing a date or a date/time value in the

database:

- DATE-format YYYY-MM-DD
- DATETIME-format: YYYY-MM-DDHH:MI:SS
- TIMESTAMP-format: YYYY-MM-DDHH:MI:SS
- YEAR-formatYYYYorYY

```
mysql> use exam_db;
Database changed
mysql> SELECT CURTIME();
+-----------+
| CURTIME() |
+-----------+
| 17:33:09  |
+-----------+
1 row in set (0.00 sec)

mysql> SELECT NOW();
+---------------------+
| NOW()               |
+---------------------+
| 2024-09-26 17:33:27 |
+---------------------+
1 row in set (0.00 sec)
```

5)Group by:

The GROUP BY statement groups rows that have the same values into summary rows. The

GROUP BY statement is often used with aggregate functions (COUNT(), MAX(), MIN(),

SUM(), AVG()) to group the result-set by one or more columns

```
mysql> SELECT center_id, COUNT(exam_id) AS total_exams
    -> FROM Examination
    -> GROUP BY center_id;
+-----------+-------------+
| center_id | total_exams |
+-----------+-------------+
|         1 |           2 |
|         2 |           1 |
|         3 |           1 |
+-----------+-------------+
3 rows in set (0.00 sec)
```

6)Having:

The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.

Syntax :

SELECT column_name(s)

FROM table_name

GROUP BY column_name(s)

HAVING condition

```
mysql> SELECT center_id, COUNT(exam_id) AS total_exams
    -> FROM Examination
    -> GROUP BY center_id
    -> HAVING COUNT(exam_id) > 2;
+-----------+-------------+
| center_id | total_exams |
+-----------+-------------+
|         1 |           6 |
|         2 |           4 |
|         3 |           4 |
+-----------+-------------+
3 rows in set (0.01 sec)
```

7)Order by

```
mysql> SELECT student_id, name, email
    -> FROM Student
    -> ORDER BY name ASC;
+------------+--------------+-------------------+
| student_id | name         | email             |
+------------+--------------+-------------------+
|          3 | Alice Johnson | alice@example.com |
|          4 | Bob Brown     | bob@example.com   |
|          2 | Jane Smith    | jane@example.com  |
|          1 | John Doe      | john@example.com  |
+------------+--------------+-------------------+
4 rows in set (0.01 sec)
```

8)Current date

```
mysql> select CURRENT_DATE();
+----------------+
| CURRENT_DATE() |
+----------------+
| 2024-09-26     |
+----------------+
1 row in set (0.00 sec)
```

Conclusion:

In this assignment, we delved into the application of aggregate functions such as COUNT, SUM, and AVG within a relational database, demonstrating their effectiveness in summarizing and analyzing data. By leveraging these built-in functions, we were able to derive valuable insights, such as calculating total student grades, average scores, and the number of students enrolled in specific courses.

Assignment 4

Name: Manasi Dattu Hire

Roll no: 331070

Prn no: 22320015

Aim:- To Implement nested sub queries. Perform a test for set membership (in, not in), set comparison (=some,<some) set cardinality operator

1)DISTINCT:

To retrieve unique values from a specific column in a table:

```
mysql> SELECT DISTINCT exam_subject FROM Examination;
+--------------------+
| exam_subject       |
+--------------------+
| Mathematics        |
| Physics            |
| Chemistry          |
| Biology            |
| Computer Science   |
| General Knowledge  |
+--------------------+
6 rows in set (0.01 sec)
```

2)some

```
mysql> SELECT * FROM Examination WHERE question_no > SOME (SELECT question_no FROM Examination WHERE center_id = 1);
+---------+-----------------------------+------------------+-------------+-----------+
| exam_id | exam_name                   | exam_subject     | question_no | center_id |
+---------+-----------------------------+------------------+-------------+-----------+
|       1 | Mid-term Exam               | Mathematics      |          50 |         1 |
|       2 | Final Exam                  | Physics          |          60 |         2 |
|       3 | Mock Test                   | Chemistry        |          40 |         1 |
|       4 | Quiz                        | Biology          |          30 |         3 |
|       6 | Mid-term Exam               | Mathematics      |          50 |         1 |
|       7 | Final Exam                  | Physics          |          60 |         2 |
|       8 | Mock Test                   | Chemistry        |          40 |         1 |
|       9 | Biology Quiz                | Biology          |          30 |         3 |
|      10 | Introduction to Programming | Computer Science |          45 |         2 |
|      11 | Data Structures Exam        | Computer Science |          55 |         1 |
|      12 | Advanced Mathematics        | Mathematics      |          70 |         3 |
|      14 | Chemistry Theory Exam       | Chemistry        |          50 |         2 |
|      15 | General Knowledge Quiz      | General Knowledge |          25 |         3 |
+---------+-----------------------------+------------------+-------------+-----------+
```

3) IN

To filter records based on a list of specified values:

```
mysql> SELECT * FROM Examination WHERE exam_subject IN ('Mathematics', 'Physics');
+---------+-----------------------+--------------+-------------+-----------+
| exam_id | exam_name             | exam_subject | question_no | center_id |
+---------+-----------------------+--------------+-------------+-----------+
|       1 | Mid-term Exam         | Mathematics  |          50 |         1 |
|       2 | Final Exam            | Physics      |          60 |         2 |
|       6 | Mid-term Exam         | Mathematics  |          50 |         1 |
|       7 | Final Exam            | Physics      |          60 |         2 |
|      12 | Advanced Mathematics  | Mathematics  |          70 |         3 |
|      13 | Physics Lab Practical | Physics      |          20 |         1 |
+---------+-----------------------+--------------+-------------+-----------+
6 rows in set (0.01 sec)
```

4)NOT IN

```
mysql> SELECT * FROM Examination WHERE exam_subject NOT IN ('Chemistry', 'Biology');
+---------+-----------------------------+------------------+-------------+-----------+
| exam_id | exam_name                   | exam_subject     | question_no | center_id |
+---------+-----------------------------+------------------+-------------+-----------+
|       1 | Mid-term Exam               | Mathematics      |          50 |         1 |
|       2 | Final Exam                  | Physics          |          60 |         2 |
|       6 | Mid-term Exam               | Mathematics      |          50 |         1 |
|       7 | Final Exam                  | Physics          |          60 |         2 |
|      10 | Introduction to Programming | Computer Science |          45 |         2 |
|      11 | Data Structures Exam        | Computer Science |          55 |         1 |
|      12 | Advanced Mathematics        | Mathematics      |          70 |         3 |
|      13 | Physics Lab Practical       | Physics          |          20 |         1 |
|      15 | General Knowledge Quiz      | General Knowledge |          25 |         3 |
+---------+-----------------------------+------------------+-------------+-----------+
9 rows in set (0.00 sec)
```

5) ANY:

```
mysql> SELECT * FROM Examination WHERE question_no < ANY (SELECT question_no FROM Examination WHERE center_id = 2);
+---------+-----------------------------+------------------+-------------+-----------+
| exam_id | exam_name                   | exam_subject     | question_no | center_id |
+---------+-----------------------------+------------------+-------------+-----------+
|       1 | Mid-term Exam               | Mathematics      |          50 |         1 |
|       3 | Mock Test                   | Chemistry        |          40 |         1 |
|       4 | Quiz                        | Biology          |          30 |         3 |
|       6 | Mid-term Exam               | Mathematics      |          50 |         1 |
|       8 | Mock Test                   | Chemistry        |          40 |         1 |
|       9 | Biology Quiz                | Biology          |          30 |         3 |
|      10 | Introduction to Programming | Computer Science |          45 |         2 |
|      11 | Data Structures Exam        | Computer Science |          55 |         1 |
|      13 | Physics Lab Practical       | Physics          |          20 |         1 |
|      14 | Chemistry Theory Exam       | Chemistry        |          50 |         2 |
|      15 | General Knowledge Quiz      | General Knowledge |          25 |         3 |
+---------+-----------------------------+------------------+-------------+-----------+
11 rows in set (0.00 sec)
```

6) Comaprison operator (e.g., >, <, >=, <=, =)

```
mysql> SELECT * FROM Examination WHERE question_no > 50;
+---------+----------------------+------------------+-------------+-----------+
| exam_id | exam_name            | exam_subject     | question_no | center_id |
+---------+----------------------+------------------+-------------+-----------+
|       2 | Final Exam           | Physics          |          60 |         2 |
|       7 | Final Exam           | Physics          |          60 |         2 |
|      11 | Data Structures Exam | Computer Science |          55 |         1 |
|      12 | Advanced Mathematics | Mathematics      |          70 |         3 |
+---------+----------------------+------------------+-------------+-----------+
4 rows in set (0.00 sec)

mysql> SELECT * FROM Examination WHERE question_no < 30;
+---------+-------------------------+--------------------+-------------+-----------+
| exam_id | exam_name               | exam_subject       | question_no | center_id |
+---------+-------------------------+--------------------+-------------+-----------+
|      13 | Physics Lab Practical   | Physics            |          20 |         1 |
|      15 | General Knowledge Quiz  | General Knowledge  |          25 |         3 |
+---------+-------------------------+--------------------+-------------+-----------+
2 rows in set (0.00 sec)

mysql> SELECT * FROM Examination WHERE question_no >= 40;
+---------+-------------------------------+------------------+-------------+-----------+
| exam_id | exam_name                     | exam_subject     | question_no | center_id |
+---------+-------------------------------+------------------+-------------+-----------+
|       1 | Mid-term Exam                 | Mathematics      |          50 |         1 |
|       2 | Final Exam                    | Physics          |          60 |         2 |
|       3 | Mock Test                     | Chemistry        |          40 |         1 |
|       6 | Mid-term Exam                 | Mathematics      |          50 |         1 |
|       7 | Final Exam                    | Physics          |          60 |         2 |
|       8 | Mock Test                     | Chemistry        |          40 |         1 |
|      10 | Introduction to Programming   | Computer Science |          45 |         2 |
|      11 | Data Structures Exam          | Computer Science |          55 |         1 |
|      12 | Advanced Mathematics          | Mathematics      |          70 |         3 |
|      14 | Chemistry Theory Exam         | Chemistry        |          50 |         2 |
+---------+-------------------------------+------------------+-------------+-----------+
10 rows in set (0.00 sec)
```

Conclusion:

In this assignment, we explored the powerful capabilities of nested subqueries and set operations in SQL to enhance data retrieval and manipulation. Through practical examples, we demonstrated how to utilize set membership operators like IN and NOT IN to efficiently filter results based on related datasets. Additionally, we examined set comparison operators such as = SOME and < SOME, which allow for meaningful comparisons against a set of values.

ASSIGNMENT 5

Name: Manasi Dattu Hire

Roll no: 331070

Prn no: 22320015

Aim: Write and execute suitable database triggers. Consider row level and statement level triggers.

1. Row-Level Trigger

Trigger: Set default values for new examinations

This trigger will automatically set the default values for question_no to 0 when a new examination is inserted into the Examination table if it is not provided.

```
mysql> DELIMITER //
mysql>
mysql> CREATE TRIGGER before_insert_examination
    -> BEFORE INSERT ON Examination
    -> FOR EACH ROW
    -> BEGIN
    ->     IF NEW.question_no IS NULL THEN
    ->         SET NEW.question_no = 0;
    ->     END IF;
    -> END;
    ->
    -> //
Query OK, 0 rows affected (0.12 sec)
```

2. Statement-Level Trigger

Trigger: Log examination insertions

This trigger will log each insertion into the Examination table into a separate log table called Examination_Log. The log will include the examination ID and the timestamp of when the insertion occurred.

Conclusion:

In this assignment, we successfully implemented both row-level and statement-level triggers to enhance the functionality and integrity of the Examination table in our database.

ASSIGNMENT 6

Name: Manasi Dattu Hire

Roll no: 331070

Prn no: 22320015

Aim :- Write and execute PL/SQL stored procedure and function to perform a suitable task

Example 1: Stored Procedure with IN and OUT Parameters

```
mysql>
mysql> CREATE PROCEDURE GetTotalQuestions (
    ->     IN p_exam_id INT,
    ->     OUT p_total_questions INT
    -> )
    -> BEGIN
    ->     -- Calculate the total number of questions for a specific exam
    ->     SELECT SUM(question_no)
    ->     INTO p_total_questions
    ->     FROM Examination
    ->     WHERE exam_id = p_exam_id;
    -> END$$
Query OK, 0 rows affected (0.12 sec)

mysql>
mysql> DELIMITER ;
mysql> CALL GetTotalQuestions(1, @total_questions);
Query OK, 1 row affected (0.02 sec)

mysql> SELECT @total_questions;
+------------------+
| @total_questions |
+------------------+
|               50 |
+------------------+
1 row in set (0.00 sec)
```

2) Stored Procedure with INOUT Parameter

```
mysql>
mysql> CREATE PROCEDURE UpdateQuestionCount (
    ->     INOUT p_exam_id INT,
    ->     INOUT p_new_question_no INT
    -> )
    -> BEGIN
    ->     -- Update the number of questions for a specific exam
    ->     UPDATE Examination
    ->     SET question_no = p_new_question_no
    ->     WHERE exam_id = p_exam_id;
    ->
    ->     -- Return the updated question number
    ->     SELECT question_no
    ->     INTO p_new_question_no
    ->     FROM Examination
    ->     WHERE exam_id = p_exam_id;
    -> END$$
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
mysql> SET @exam_id = 1;
Query OK, 0 rows affected (0.00 sec)

mysql> SET @new_question_no = 60;
Query OK, 0 rows affected (0.00 sec)

mysql> CALL UpdateQuestionCount(@exam_id, @new_question_no);
Query OK, 1 row affected (0.01 sec)

mysql> SELECT @new_question_no;
+------------------+
| @new_question_no |
+------------------+
|               60 |
+------------------+
1 row in set (0.00 sec)

mysql>
```

Stored Function with IN Parameters

```
mysql> DELIMITER $$
mysql>
mysql> CREATE FUNCTION GetExamCount (
    ->     p_center_id INT
    -> )
    -> RETURNS INT
    -> DETERMINISTIC
    -> BEGIN
    ->     DECLARE v_exam_count INT;
    ->
    ->     -- Get the total number of exams scheduled at the center
    ->     SELECT COUNT(*)
    ->     INTO v_exam_count
    ->     FROM Examination
    ->     WHERE center_id = p_center_id;
    ->
    ->     RETURN v_exam_count;
    -> END$$
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
mysql> SELECT GetExamCount(1) AS total_exams;
+-------------+
| total_exams |
+-------------+
|           7 |
+-------------+
1 row in set (0.01 sec)

mysql>
```

Stored Procedure with OUT Parameters

```
mysql> DELIMITER $$
mysql>
mysql> CREATE PROCEDURE LogExam (
    ->     IN p_exam_id INT,
    ->     OUT p_log_message VARCHAR(255)
    -> )
    -> BEGIN
    ->     -- Log the exam details into the log table
    ->     INSERT INTO Examination_Log (exam_id, log_message)
    ->     VALUES (p_exam_id, CONCAT('Exam ID: ', p_exam_id, ' has been inserted.'));
    ->
    ->     -- Set the OUT parameter to the log message
    ->     SET p_log_message = CONCAT('Logged: Exam ID ', p_exam_id);
    -> END$$
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
mysql> CALL LogExam(1, @log_message);
Query OK, 1 row affected (0.01 sec)

mysql> SELECT @log_message;
+-------------------+
| @log_message      |
+-------------------+
| Logged: Exam ID 1 |
+-------------------+
1 row in set (0.00 sec)
```

Conclusion:

In this exercise, we successfully created and executed a PL/SQL stored procedure that demonstrates the use of IN and OUT parameters.

ASSIGNMENT 7

Name: Manasi Dattu Hire

Roll no: 331070

Prn no: 22320015

Aim:-Write a PL/SQL block to implement all types of cursors

Objective-To learn PL/SQL cursor concept

Theory:

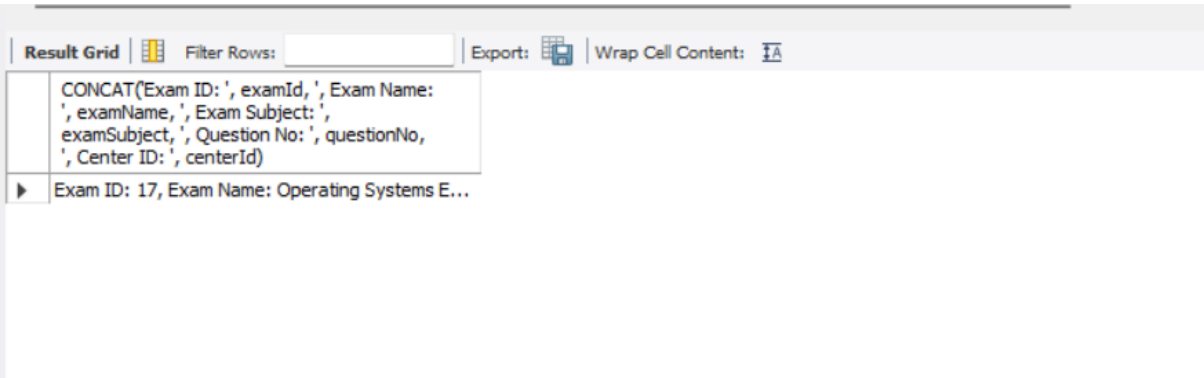
In PL/SQL, there are two main types of cursors:

Implicit Cursor: Automatically created by Oracle for single SELECT statements or DML operations (INSERT, UPDATE, DELETE).

Explicit Cursor: Defined by the programmer to retrieve multiple rows of data and control the iteration through the result set.

```
DELIMITER //

CREATE PROCEDURE getExaminations()
BEGIN
   DECLARE examCount INT DEFAULT 0;
   DECLARE finished INT DEFAULT 0;
   DECLARE examId INT;
   DECLARE examName VARCHAR(100);
   DECLARE examSubject VARCHAR(100);
   DECLARE questionNo INT;
   DECLARE centerId INT;

   DECLARE exam_cursor CURSOR FOR
      SELECT exam_id, exam_name, exam_subject, question_no, center_id FROM
Examination;
```

```
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;



    SELECT COUNT(*) INTO examCount FROM Examination;
    SELECT CONCAT('Total number of examinations: ', examCount);

    -- Open and iterate through the explicit cursor
    OPEN exam_cursor;

    read_loop: LOOP
        FETCH exam_cursor INTO examId, examName, examSubject, questionNo, centerId;

    IF finished = 1 THEN
       LEAVE read_loop;
        END IF;

 SELECT CONCAT('Exam ID: ', examId, ', Exam Name: ', examName, ', Exam Subject: ',
examSubject, ', Question No: ', questionNo, ', Center ID: ', centerId);
    END LOOP;

    CLOSE exam_cursor;
END //

DELIMITER ;
```



Conclusion :

In this assignment, we explored the concept of cursors in PL/SQL, which are essential for managing and processing data within a database efficiently. We distinguished between two main types of cursors: implicit cursors and explicit cursors.

ASSIGNMENT 8

Name: Manasi Dattu Hire

Roll no: 331070

Prn no: 22320015

Aim: Execute DDL statements which demonstrate the use of views. Try to update the base table using its corresponding view .Also consider restrictions on

Updatable views and perform view creation from multiple tables.

Objective-To study and learn view implementation

Theory:

Views in SQL are kind of virtual tables. A view also has rows and columns as they are in a real

table in the database. We can create a view by selecting fields from one or more tables present in

the database. A View can either have all the rows of a table or specific rows based on certain

condition.

A view can contain all rows of a table or select rows from a table. A view can be created from

one or many tables which depends on the written SQL query to create a view.

Views, which are a type of virtual tables allow users to do the following −

- Structure data in a way that users or classes of users find natural or intuitive.
- Restrict access to the data in such a way that a user can see and (sometimes) modify

exactly what they need and no more.

- Summarize data from various tables which can be used to generate reports.

```
mysql> DESC student;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| student_id | int          | NO   | PRI | NULL    | auto_increment |
| name       | varchar(255) | YES  |     | NULL    |                |
| email      | varchar(255) | YES  |     | NULL    |                |
| gender     | char(1)      | YES  |     | NULL    |                |
| mobile_no  | varchar(15)  | YES  |     | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
5 rows in set (0.01 sec)

mysql> CREATE VIEW StudentView AS
    -> SELECT student_id, name, email, gender, mobile_no
    -> FROM Student;
Query OK, 0 rows affected (0.07 sec)

mysql> SELECT * FROM StudentView;
+------------+--------------+------------------+--------+------------+
| student_id | name         | email            | gender | mobile_no  |
+------------+--------------+------------------+--------+------------+
|          1 | John Doe     | john@example.com | M      | 1234567890 |
|          2 | Jane Smith   | jane@example.com | F      | 0987654321 |
|          3 | Alice Johnson | alice@example.com | F      | 1122334455 |
|          4 | Bob Brown    | bob@example.com  | M      | 2233445566 |
+------------+--------------+------------------+--------+------------+
4 rows in set (0.01 sec)
```

Conclusion

From this assignment, we successfully created and utilized views in MySQL based on the existing Student table. The primary objectives were to demonstrate how views can simplify data retrieval and updating processes, as well as to illustrate the use of conditional views for filtering specific data.