SQL LAB -7
(Qualifiers, Having Clause, Alter Table, Transactions)

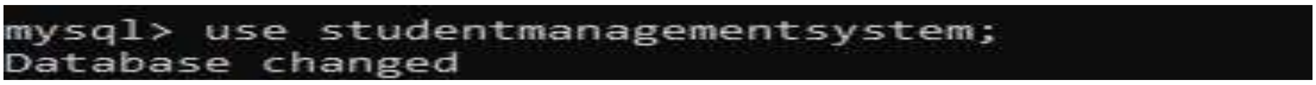
Submitted by: Pravallika Ullikanti

AF ID : AF0366355

Batch ID : ANP-C7281

Database Schema:

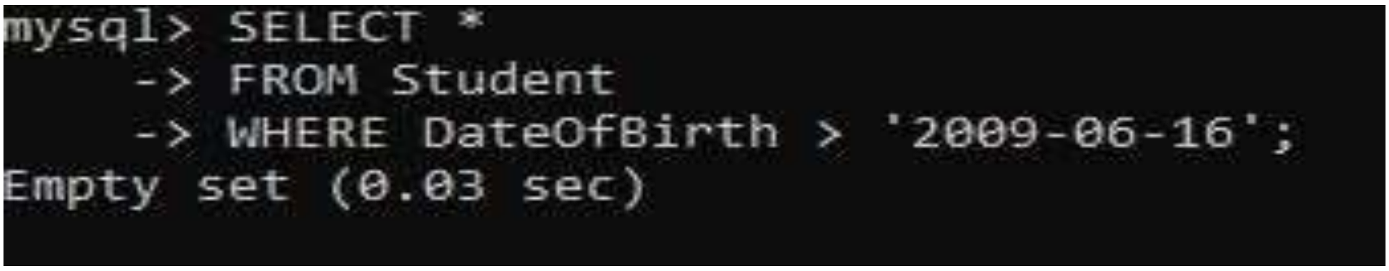
Use the same database scheme created in previous lab.



```
mysql> use studentmanagementsystem;
Database changed
```

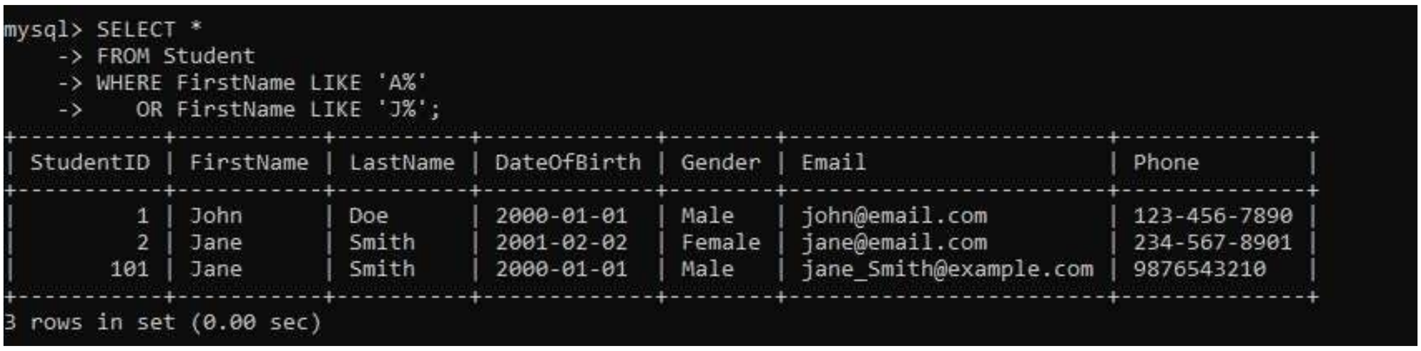
Assignment 1:

Task1: Assume you are managing a database of student records, and you need to retrieve information about students born after June 16, 2009. What will be the sql query for this?



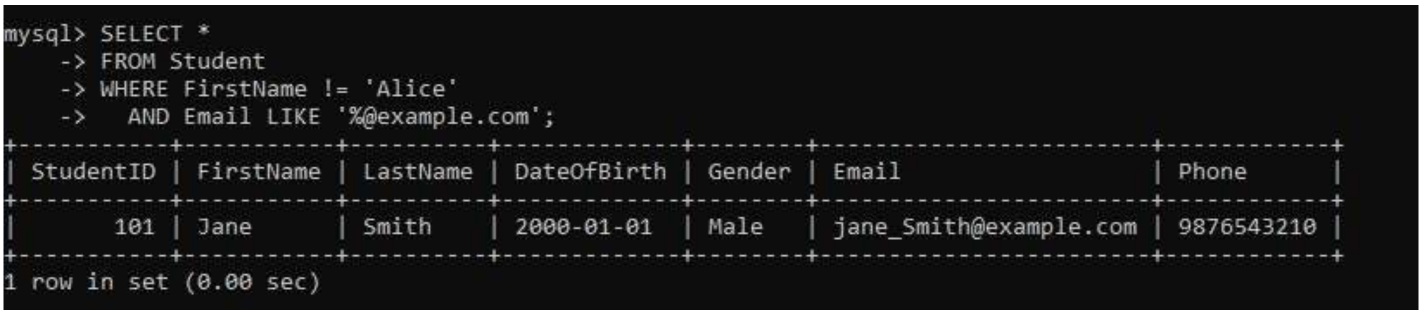
```
mysql> SELECT *
    -> FROM Student
    -> WHERE DateOfBirth > '2009-06-16';
Empty set (0.03 sec)
```

Task2: Assume you have a database containing a "Student" table with information about students, including their first names. You want to retrieve records of students whose first names start with either 'A' or 'J'. To achieve this,what will be your sql query?



```
mysql> SELECT *
    -> FROM Student
    -> WHERE FirstName LIKE 'A%'
    ->    OR FirstName LIKE 'J%';
+-----------+-----------+----------+-------------+--------+--------------------------+--------------+
| StudentID | FirstName | LastName | DateOfBirth | Gender | Email                    | Phone        |
+-----------+-----------+----------+-------------+--------+--------------------------+--------------+
|         1 | John      | Doe      | 2000-01-01  | Male   | john@email.com           | 123-456-7890 |
|         2 | Jane      | Smith    | 2001-02-02  | Female | jane@email.com           | 234-567-8901 |
|       101 | Jane      | Smith    | 2000-01-01  | Male   | jane_Smith@example.com   | 9876543210   |
+-----------+-----------+----------+-------------+--------+--------------------------+--------------+
3 rows in set (0.00 sec)
```

Task 3 .Let's consider a scenario where you have a database with a "Student" table that contains information about students, including their first names and email addresses. You want to retrieve records of students whose first name is not 'Alice' and whose email addresses contain the domain '@example.com'. To achieve this, what will be your sql query?



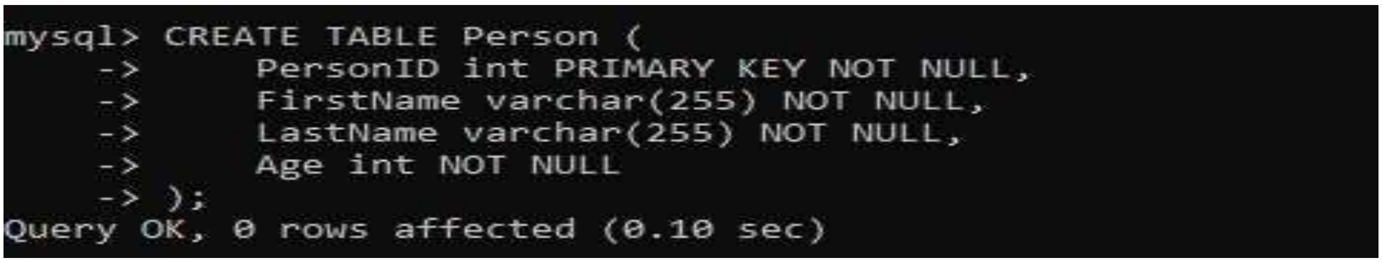
```
mysql> SELECT *
    -> FROM Student
    -> WHERE FirstName != 'Alice'
    ->   AND Email LIKE '%@example.com';
+-----------+-----------+----------+-------------+--------+-----------------------+------------+
| StudentID | FirstName | LastName | DateOfBirth | Gender | Email                 | Phone      |
+-----------+-----------+----------+-------------+--------+-----------------------+------------+
|       101 | Jane      | Smith    | 2000-01-01  | Male   | jane_Smith@example.com | 9876543210 |
+-----------+-----------+----------+-------------+--------+-----------------------+------------+
1 row in set (0.00 sec)
```

Assignment 2:

Task1: Create a table Person with PersonID int, FirstName varchar(255),

LastName varchar(255) and age (int).

Make PersonID PRIMARY KEY.

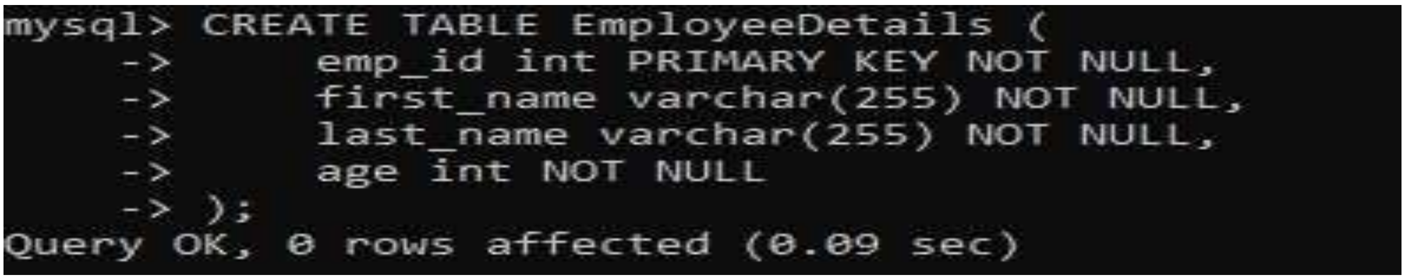


```
mysql> CREATE TABLE Person (
    ->     PersonID int PRIMARY KEY NOT NULL,
    ->     FirstName varchar(255) NOT NULL,
    ->     LastName varchar(255) NOT NULL,
    ->     Age int NOT NULL
    -> );
Query OK, 0 rows affected (0.10 sec)
```

Task2: Create a table Employee with emp_id int, first_name varchar(255)

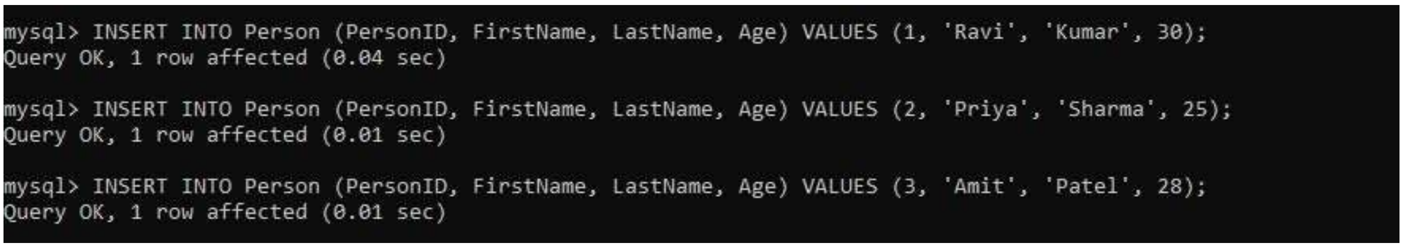
last_name varchar(255) and age (int)

Make emp_id PRIMARY KEY.



```
mysql> CREATE TABLE EmployeeDetails (
    ->     emp_id int PRIMARY KEY NOT NULL,
    ->     first_name varchar(255) NOT NULL,
    ->     last_name varchar(255) NOT NULL,
    ->     age int NOT NULL
    -> );
Query OK, 0 rows affected (0.09 sec)
```

Task 3: Insert data to Person table

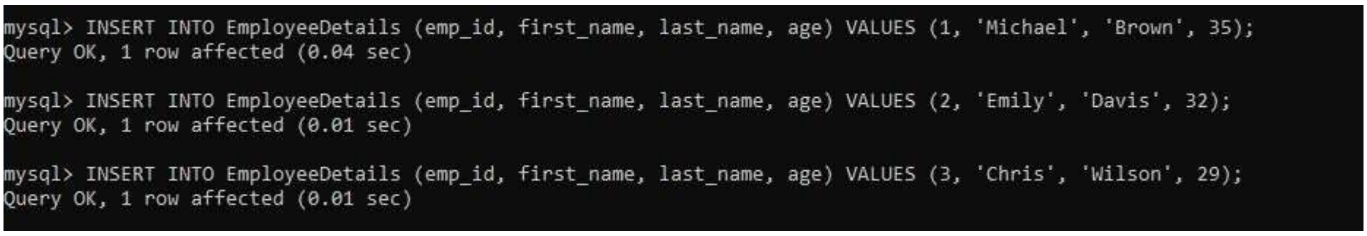


```
mysql> INSERT INTO Person (PersonID, FirstName, LastName, Age) VALUES (1, 'Ravi', 'Kumar', 30);
Query OK, 1 row affected (0.04 sec)

mysql> INSERT INTO Person (PersonID, FirstName, LastName, Age) VALUES (2, 'Priya', 'Sharma', 25);
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO Person (PersonID, FirstName, LastName, Age) VALUES (3, 'Amit', 'Patel', 28);
Query OK, 1 row affected (0.01 sec)
```

Task 4: Insert data to Employee table

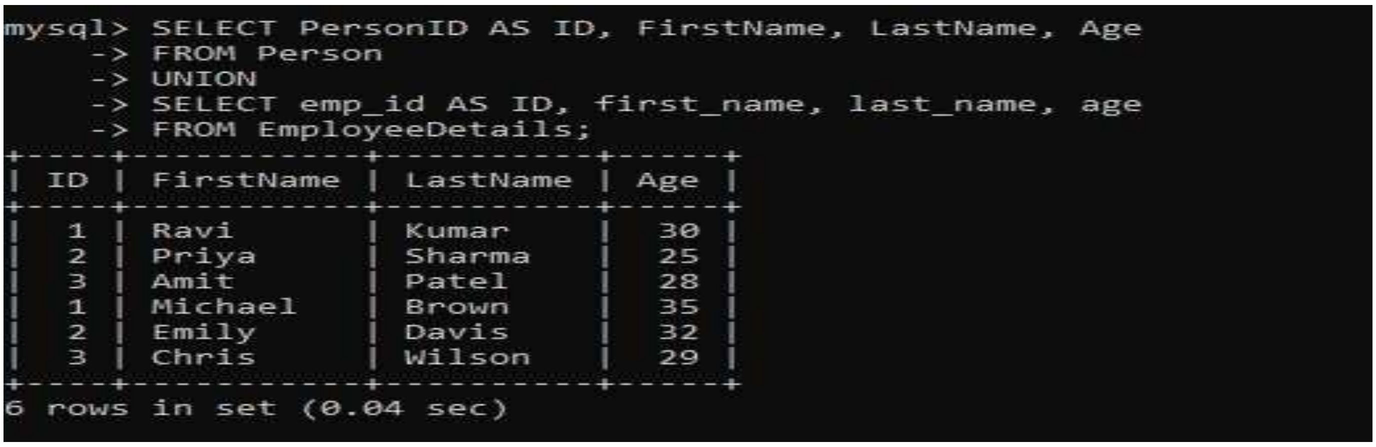


```
mysql> INSERT INTO EmployeeDetails (emp_id, first_name, last_name, age) VALUES (1, 'Michael', 'Brown', 35);
Query OK, 1 row affected (0.04 sec)

mysql> INSERT INTO EmployeeDetails (emp_id, first_name, last_name, age) VALUES (2, 'Emily', 'Davis', 32);
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO EmployeeDetails (emp_id, first_name, last_name, age) VALUES (3, 'Chris', 'Wilson', 29);
Query OK, 1 row affected (0.01 sec)
```

Task 5: Create Union of two tables



```
mysql> SELECT PersonID AS ID, FirstName, LastName, Age
    -> FROM Person
    -> UNION
    -> SELECT emp_id AS ID, first_name, last_name, age
    -> FROM EmployeeDetails;
+----+-----------+-----------+-----+
| ID | FirstName | LastName  | Age |
+----+-----------+-----------+-----+
|  1 | Ravi      | Kumar     |  30 |
|  2 | Priya     | Sharma    |  25 |
|  3 | Amit      | Patel     |  28 |
|  1 | Michael   | Brown     |  35 |
|  2 | Emily     | Davis     |  32 |
|  3 | Chris     | Wilson    |  29 |
+----+-----------+-----------+-----+
6 rows in set (0.04 sec)
```

ChatGPT Exercise

Using ChatGPT generate SQL queries of the below problem .

Scenario 1: In a student grades database with tables for courses and grades, find the courses where the average grade is below a 'C' (consider 'C' as a passing grade).

We have a "Course" table with the following columns:

CourseId, CourseName, CreditHours and "Grade" table with the following a columns: StudentId(Foreign Key),CourseID((Foreign Key),Grade. you want to find courses where the average grade is below a "C". Generate the ChatGPT prompt for creating the queries for the above requirement.

SELECT c.CourseId, c.CourseName

FROM Course c

JOIN Grade g ON c.CourseId = g.CourseId

GROUP BY c.CourseId, c.CourseName

HAVING AVG(g.Grade) < (SELECT AVG(Grade) FROM Grade WHERE Grade = 'C');

This SQL query retrieves the CourseId and CourseName from the Course table where the average grade for that course, calculated from the Grade table, is below a 'C'. It joins the Course and Grade tables on the CourseId, calculates the average grade for each course, and filters the results to only include courses where the average grade is below the average grade of 'C'.