

SQL LAB 7

BY

Revanth Derangula

AF0366973

Database Schema:

Use the same database scheme created in previous lab.

Assignment 1:

Task 1: 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?

Task 2: 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?

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?

Submission:

Create an SQL script file containing your solutions for all tasks (queries). Name the file "lab_assignment1.sql" Provide comments above each query to indicate the query's purpose.

Assignment 2:

Task1: Create a table Person with PersonID int, FirstName varchar(255), LastName varchar(255) and age (int).

Make PersonID PRIMARY KEY.

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.

Task 3: Insert data to Person table Task

4: Insert data to Employee table Task 5:

Create Union of two tables

Submission:

Create an SQL script file containing your solutions for the task. Name the file "lab_assignment2.sql" Provide comments above the query to indicate the query's purpose.

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 columns: StudentId(ForeignKey), CourseID((ForeignKey), 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.

SOLUTION:

Database Schema:

Use the same database scheme created in previous lab.

Assignment 1:

Task 1: 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 queryfor this?

Code:

```
CREATE TABLE Students_data (  
  ID INT PRIMARY KEY,  
  First_Name VARCHAR(50),  
  Last_Name VARCHAR(50),  
  City VARCHAR(50),  
  Age INT,  
  Birth_Date DATE  
);
```

Code:

```
INSERT INTO Students_data (ID, First_Name, Last_Name, City, Age, Birth_Date)  
VALUES  
(1, 'Aarav', 'Sharma', 'Mumbai', 23, '2000-01-15'),  
(2, 'Vivaan', 'Verma', 'Delhi', 22, '2001-02-22'),  
(3, 'Diya', 'Patel', 'Bangalore', 21, '2002-03-30'),
```

```
(4, 'Aanya', 'Reddy', 'Hyderabad', 20, '2003-04-12'),
(5, 'Ishaan', 'Singh', 'Chennai', 19, '2004-05-19'),
(6, 'Anaya', 'Kumar', 'Pune', 18, '2005-06-05'),
(7, 'Arjun', 'Nair', 'Kochi', 17, '2006-07-20'),
(8, 'Aadhya', 'Mehta', 'Ahmedabad', 16, '2007-08-25'),
(9, 'Aryan', 'Joshi', 'Surat', 15, '2008-09-10'),
(10, 'Anvi', 'Bose', 'Kolkata', 14, '2009-10-18'),
(11, 'Vihaan', 'Das', 'Lucknow', 13, '2010-11-30'),
(12, 'Mira', 'Roy', 'Jaipur', 12, '2011-12-25'),
(13, 'Reyansh', 'Chopra', 'Chandigarh', 11, '2012-01-15'),
(14, 'Aarohi', 'Kapoor', 'Indore', 10, '2013-02-22'),
(15, 'Kabir', 'Malhotra', 'Bhopal', 9, '2014-03-30');
```

Output:

```
mysql> CREATE TABLE Students_data (
  -> ID INT PRIMARY KEY,
  -> First_Name VARCHAR(50),
  -> Last_Name VARCHAR(50),
  -> City VARCHAR(50),
  -> Age INT,
  -> Birth_Date DATE
  -> );
Query OK, 0 rows affected (0.25 sec)

mysql> INSERT INTO Students_data (ID, First_Name, Last_Name, City, Age,
  -> Birth_Date)
  -> VALUES
  -> (1, 'Aarav', 'Sharma', 'Mumbai', 23, '2000-01-15'),
  -> (2, 'Vivaan', 'Verma', 'Delhi', 22, '2001-02-22'),
  -> (3, 'Diya', 'Patel', 'Bangalore', 21, '2002-03-30'),
  -> (4, 'Aanya', 'Reddy', 'Hyderabad', 20, '2003-04-12'),
  -> (5, 'Ishaan', 'Singh', 'Chennai', 19, '2004-05-19'),
  -> (6, 'Anaya', 'Kumar', 'Pune', 18, '2005-06-05'),
  -> (7, 'Arjun', 'Nair', 'Kochi', 17, '2006-07-20'),
  -> (8, 'Aadhya', 'Mehta', 'Ahmedabad', 16, '2007-08-25'),
  -> (9, 'Aryan', 'Joshi', 'Surat', 15, '2008-09-10'),
  -> (10, 'Anvi', 'Bose', 'Kolkata', 14, '2009-10-18'),
  -> (11, 'Vihaan', 'Das', 'Lucknow', 13, '2010-11-30'),
  -> (12, 'Mira', 'Roy', 'Jaipur', 12, '2011-12-25'),
  -> (13, 'Reyansh', 'Chopra', 'Chandigarh', 11, '2012-01-15'),
  -> (14, 'Aarohi', 'Kapoor', 'Indore', 10, '2013-02-22'),
  -> (15, 'Kabir', 'Malhotra', 'Bhopal', 9, '2014-03-30');
Query OK, 15 rows affected (0.01 sec)
Records: 15 Duplicates: 0 Warnings: 0
```

Code:

SELECT * FROM Students_data WHERE birth_date > '2009-06-16'; Output:

```
mysql> select * from Students_data;
```

ID	First_Name	Last_Name	City	Age	Birth_Date
1	Aarav	Sharma	Mumbai	23	2000-01-15
2	Vivaan	Verma	Delhi	22	2001-02-22
3	Diya	Patel	Bangalore	21	2002-03-30
4	Aanya	Reddy	Hyderabad	20	2003-04-12
5	Ishaan	Singh	Chennai	19	2004-05-19
6	Anaya	Kumar	Pune	18	2005-06-05
7	Arjun	Nair	Kochi	17	2006-07-20
8	Aadhya	Mehta	Ahmedabad	16	2007-08-25
9	Aryan	Joshi	Surat	15	2008-09-10
10	Anvi	Bose	Kolkata	14	2009-10-18
11	Vihaan	Das	Lucknow	13	2010-11-30
12	Mira	Roy	Jaipur	12	2011-12-25
13	Reyansh	Chopra	Chandigarh	11	2012-01-15
14	Aarohi	Kapoor	Indore	10	2013-02-22
15	Kabir	Malhotra	Bhopal	9	2014-03-30

```
15 rows in set (0.00 sec)
```



```
mysql> SELECT * FROM Students_data WHERE birth_date > '2009-06-16';
```

ID	First_Name	Last_Name	City	Age	Birth_Date
10	Anvi	Bose	Kolkata	14	2009-10-18
11	Vihaan	Das	Lucknow	13	2010-11-30
12	Mira	Roy	Jaipur	12	2011-12-25
13	Reyansh	Chopra	Chandigarh	11	2012-01-15
14	Aarohi	Kapoor	Indore	10	2013-02-22
15	Kabir	Malhotra	Bhopal	9	2014-03-30

```
6 rows in set (0.00 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?

Code:

```
SELECT * FROM Students_data
WHERE first_name LIKE 'A%' OR first_name LIKE 'J%';
```

Output:

```
mysql> SELECT * FROM Students_data
-> WHERE first_name LIKE 'A%' OR first_name LIKE 'J%';
```

ID	First_Name	Last_Name	City	Age	Birth_Date
1	Aarav	Sharma	Mumbai	23	2000-01-15
4	Aanya	Reddy	Hyderabad	20	2003-04-12
6	Anaya	Kumar	Pune	18	2005-06-05
7	Arjun	Nair	Kochi	17	2006-07-20
8	Aadhya	Mehta	Ahmedabad	16	2007-08-25
9	Aryan	Joshi	Surat	15	2008-09-10
10	Anvi	Bose	Kolkata	14	2009-10-18
14	Aarohi	Kapoor	Indore	10	2013-02-22

```
8 rows in set (0.20 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?

Code:

```
CREATE TABLE Student ( id INT
AUTO_INCREMENT PRIMARY KEY,
first_name VARCHAR(255),
email VARCHAR(255)
);
```

Output:

```
CREATE TABLE Student (
id INT AUTO_INCREMENT PRIMARY KEY,
first_name VARCH' at line 1
mysql> CREATE TABLE Student (
-> id INT AUTO_INCREMENT PRIMARY KEY,
-> first_name VARCHAR(255),
-> email VARCHAR(255)
-> );
Query OK, 0 rows affected (0.05 sec)
```

Code:

```
SELECT * FROM Student
```

WHERE first_name <> 'Alice' AND email LIKE '%@example.com'; Output:


```
mysql> select * from Student;
+----+-----+-----+
| id | first_name | email |
+----+-----+-----+
| 1  | Alice      | alice@example.com |
| 2  | John       | john@example.com   |
| 3  | Jane       | jane@example.com   |
| 4  | Bob        | bob@example.com     |
| 5  | Anna       | anna@example.com   |
+----+-----+-----+
5 rows in set (0.00 sec)

mysql> SELECT * FROM Student
-> WHERE first_name <> 'Alice' AND email LIKE '%@example.com';
+----+-----+-----+
| id | first_name | email |
+----+-----+-----+
| 2  | John       | john@example.com   |
| 3  | Jane       | jane@example.com   |
| 4  | Bob        | bob@example.com     |
| 5  | Anna       | anna@example.com   |
+----+-----+-----+
4 rows in set (0.00 sec)
```

Submission:

Create an SQL script file containing your solutions for all tasks (queries). Name the file "lab_assignment1.sql" Provide comments above each query to indicate the query's purpose.

Assignment 2:

Task1: Create a table Person with PersonID int, FirstName varchar(255), LastName varchar(255) and age (int).

Make PersonID PRIMARY KEY.

Code:

```
CREATE TABLE Person (
  PersonID INT AUTO_INCREMENT PRIMARY KEY,
  FirstName VARCHAR(255),
  LastName VARCHAR(255), Age
  INT
);
```

Task 3: Insert data to Person table

Code:

```
-- Insert data into the Person table
('John', 'Doe', 30),
```

('Jane', 'Smith', 25),
('Alice', 'Johnson', 28),
('Bob', 'Brown', 35),

('Emily', 'Davis',

22); Output:

```
mysql> select * from Person;
```

PersonID	FirstName	LastName	Age
1	John	Doe	30
2	Jane	Smith	25
3	Alice	Johnson	28
4	Bob	Brown	35
5	Emily	Davis	22
6	John	Doe	30
7	Jane	Smith	25
8	Alice	Johnson	28
9	Bob	Brown	35
10	Emily	Davis	22

10 rows in set (0.00 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.

Code:

```
CREATE TABLE Employee ( emp_id INT  
    AUTO_INCREMENT PRIMARY KEY, first_name  
    VARCHAR(255), last_name VARCHAR(255),  
    age INT  
);
```

Task 4: Insert data to Employee table Code:

-- Insert data into the Employee table

```
INSERT INTO Employee (first_name, last_name, age)  
VALUES
```

('Michael', 'Scott', 45);

('Jim', 'Halpert', 32);

('Pam', 'Beesly', 30);

('Dwight', 'Schrute', 38);

('Stanley', 'Hudson', 50);

Output:

```
mysql> select * from Employee;
+-----+-----+-----+-----+
| emp_id | first_name | last_name | age |
+-----+-----+-----+-----+
|      1 | Michael   | Scott    | 45  |
|      2 | Jim       | Halpert  | 32  |
|      3 | Pam       | Beesly   | 30  |
|      4 | Dwight    | Schrute  | 38  |
|      5 | Stanley   | Hudson   | 50  |
+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

Task 5: Create a Union of two tables

Code:

SELECT first_name AS Name, last_name AS Surname, age AS Age FROM
Employee

UNION

SELECT FirstName AS Name, LastName AS Surname, Age FROM Person;

Output:

```
mysql> SELECT first_name AS Name, last_name AS Surname, age AS Age FROM
-> Employee
-> UNION
-> SELECT FirstName AS Name, LastName AS Surname, Age FROM Person;
+-----+-----+-----+
| Name   | Surname | Age |
+-----+-----+-----+
| Michael | Scott   | 45  |
| Jim     | Halpert | 32  |
| Pam     | Beesly  | 30  |
| Dwight  | Schrute | 38  |
| Stanley | Hudson  | 50  |
| John    | Doe     | 30  |
| Jane    | Smith   | 25  |
| Alice   | Johnson | 28  |
| Bob     | Brown   | 35  |
| Emily   | Davis   | 22  |
+-----+-----+-----+
10 rows in set (0.00 sec)
```

Submission:

Create an SQL script file containing your solutions for the task. Name the file "lab_assignment2.sql" Provide comments above the query to indicate the

query's purpose. 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 columns: StudentId(ForeignKey), CourseID((ForeignKey), Grade. you want to find courses where the average grade is below a "C". Generate the theChatGPT prompt for creating the queries for the above requirement.

Code:

-- Create the Course table

```
CREATE TABLE Course (  
  CourseId INT PRIMARY KEY,  
  CourseName VARCHAR(100),  
  CreditHours INT  
);
```

-- Create the Grade table

```
CREATE TABLE Grade (  
  StudentId INT,  
  CourseId INT,  
  Grade CHAR(1),  
  FOREIGN KEY (CourseId) REFERENCES Course(CourseId)  
);
```

Output:

```
mysql> select * from Course;
```

CourseId	CourseName	CreditHours
1	Mathematics	4
2	Physics	3
3	Chemistry	4
4	Biology	3
5	History	2

```
5 rows in set (0.00 sec)
```



```
mysql> select * from Grade;
```

StudentId	CourseId	Grade
101	1	A
102	1	B
103	1	C
101	2	B
102	2	C
103	2	D
101	3	A
102	3	A
103	3	B
101	1	A
102	1	B
103	1	C
101	2	B
102	2	C
103	2	D
101	3	A
102	3	A
103	3	B
101	4	C
102	4	D
103	4	F
101	5	B
102	5	C
103	5	C

```
24 rows in set (0.00 sec)
```

Code:

-- Assuming the following grade point values:

-- 'A' = 4.0

-- 'B' = 3.0

-- 'C' = 2.0

-- 'D' = 1.0

-- 'F' = 0.0

-- SQL query to find courses where the average grade is below a 'C'

```

SELECT c.CourseId, c.CourseName, c.CreditHours
FROM Course c
JOIN Grade g ON c.CourseId = g.CourseId
GROUP BY c.CourseId, c.CourseName, c.CreditHours
HAVING AVG(
CASE g.Grade
WHEN 'A' THEN 4.0
WHEN 'B' THEN 3.0
WHEN 'C' THEN 2.0
WHEN 'D' THEN 1.0
WHEN 'F' THEN 0.0
ELSE NULL END
) < 2.0;

```

Code:

```

mysql> -- Assuming the following grade point values:
mysql> -- 'A' = 4.0
mysql> -- 'B' = 3.0
mysql> -- 'C' = 2.0
mysql> -- 'D' = 1.0
mysql> -- 'F' = 0.0
mysql> -- SQL query to find courses where the average grade is below a 'C'
mysql> SELECT c.CourseId, c.CourseName, c.CreditHours
-> FROM Course c
-> JOIN Grade g ON c.CourseId = g.CourseId
-> GROUP BY c.CourseId, c.CourseName, c.CreditHours
-> HAVING AVG(
-> CASE g.Grade
-> WHEN 'A' THEN 4.0
-> WHEN 'B' THEN 3.0
-> WHEN 'C' THEN 2.0
-> WHEN 'D' THEN 1.0
-> WHEN 'F' THEN 0.0
-> ELSE NULL
-> END
-> ) < 2.0;
+-----+-----+-----+
| CourseId | CourseName | CreditHours |
+-----+-----+-----+
| 4 | Biology | 3 |
+-----+-----+-----+
1 row in set (0.00 sec)

```