Topic: SQL, Database Schema

Created By: Pritey Mehta

Important Notes:

All queries should be submitted in a SQL file with the query number as the file name.

The query definition should be in this file.

1. **Create a database structure for employee leave application. It should include all the employee's information as well as their leave information.**
2. **Write an SQL query to report the movies with an odd-numbered ID and a description that is not "boring". Return the result table ordered by rating in descending order.**

Table: Cinema

```
+----------------+-----------+
| Column Name    | Type      |
+----------------+-----------+
| id             | int       |
| movie          | varchar   |
| description    | varchar   |
| rating         | float     |
+----------------+-----------+
id is the primary key for this table.
```

Each row contains information about the name of a movie, its genre, and its rating.
rating is a 2 decimal places float in the range [0, 10]

The query result format is in the following example.

Example 1:

Input:
Cinema table:
```
+----+------------+-------------+--------+
| id | movie      | description | rating |
+----+------------+-------------+--------+
| 1  | War        | great 3D    | 8.9    |
| 2  | Science    | fiction     | 8.5    |
| 3  | irish      | boring      | 6.2    |
| 4  | Ice song   | Fantacy     | 8.6    |
| 5  | House card | Interesting | 9.1    |
+----+------------+-------------+--------+
```
Output:
```
+----+------------+-------------+--------+
| id | movie      | description | rating |
+----+------------+-------------+--------+
| 5  | House card | Interesting | 9.1    |
| 1  | War        | great 3D    | 8.9    |
+----+------------+-------------+--------+
```
Explanation:
We have three movies with odd-numbered IDs: 1, 3, and 5. The movie with ID = 3 is boring so we do not include it in the answer.

3. **Write an SQL query to swap all 'f' and 'm' values (i.e., change all 'f' values to 'm' and vice versa) with a single update statement and no intermediate temporary tables.Note that you must write a single update statement, do not write any select statement for this problem.**

   Table: Salary

```
+-------------+----------+
| Column Name | Type     |
+-------------+----------+
| id          | int      |
| name        | varchar  |
| sex         | ENUM     |
| salary      | int      |
+-------------+----------+
```
id is the primary key for this table.
The sex column is ENUM value of type ('m', 'f').
The table contains information about an employee.

The query result format is in the following example.

Example 1:

Input:
Salary table:
```
+----+------+-----+--------+
| id | name | sex | salary |
+----+------+-----+--------+
| 1  | A    | m   | 2500   |
| 2  | B    | f   | 1500   |
| 3  | C    | m   | 5500   |
| 4  | D    | f   | 500    |
+----+------+-----+--------+
```
Output:
```
+----+------+-----+--------+
| id | name | sex | salary |
+----+------+-----+--------+
| 1  | A    | f   | 2500   |
| 2  | B    | m   | 1500   |
| 3  | C    | f   | 5500   |
| 4  | D    | m   | 500    |
+----+------+-----+--------+
```
Explanation:
(1, A) and (3, C) were changed from 'm' to 'f'.
(2, B) and (4, D) were changed from 'f' to 'm'.

4. **Write an SQL query to delete all the duplicate emails, keeping only one unique email with the smallest id. Return the result table in any order.**

   Table: Person

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| id          | int     |
```

email	varchar

id is the primary key column for this table.
Each row of this table contains an email. The emails will not contain uppercase letters.

The query result format is in the following example.

Example 1:

Input:
Person table:

id	email
1	john@example.com
2	bob@example.com
3	john@example.com

Output:

id	email
1	john@example.com
2	bob@example.com

Explanation: john@example.com is repeated two times. We keep the row with the smallest Id = 1.

5. **Write an SQL query to report all customers who never order anything. Return the result table in any order.**

Table: Customers

Column Name	Type
id	int
name	varchar

id is the primary key column for this table.

Each row of this table indicates the ID and name of a customer.

Table: Orders

```
+-------------+------+
| Column Name | Type |
+-------------+------+
| id          | int  |
| customerId  | int  |
+-------------+------+
```

id is the primary key column for this table.
customerId is a foreign key of the ID from the Customers table.
Each row of this table indicates the ID of an order and the ID of the customer who ordered it.

The query result format is in the following example.

Example 1:

Input:
Customers table:

```
+----+-------+
| id | name  |
+----+-------+
| 1  | Joe   |
| 2  | Henry |
| 3  | Sam   |
| 4  | Max   |
+----+-------+
```

Orders table:

```
+----+------------+
| id | customerId |
+----+------------+
| 1  | 3          |
| 2  | 1          |
+----+------------+
```

Output:

```
+-----------+
| Customers |
+-----------+
| Henry     |
| Max       |
+-----------+
```

6. **Write an SQL query to create index on the email column.**

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| id          | int     |
| email       | varchar |
+-------------+---------+
```

7. **Create a database schema for student grade system. It contains student data and their grade of each subject based on the different semester.**

8. **Write an SQL query to report the first name, last name, city, and state of each person in the Person table. If the address of a personId is not present in the Address table, report null instead. Return the result table in** any order.

   Table: Person

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| personId    | int     |
| lastName    | varchar |
| firstName   | varchar |
+-------------+---------+
personId is the primary key column for this table.
This table contains information about the ID of some persons
and their first and last names.
```

   Table: Address

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| addressId   | int     |
| personId    | int     |
| city        | varchar |
| state       | varchar |
+-------------+---------+
addressId is the primary key column for this table.
```

Each row of this table contains information about the city and state of one person with ID = PersonId.

The query result format is in the following example.

Example 1:

Input:
```
Person table:
+----------+----------+-----------+
| personId | lastName | firstName |
+----------+----------+-----------+
| 1        | Wang     | Allen     |
| 2        | Alice    | Bob       |
+----------+----------+-----------+
Address table:
+-----------+----------+---------------+------------+
| addressId | personId | city          | state      |
+-----------+----------+---------------+------------+
| 1         | 2        | New York City | New York   |
| 2         | 3        | Leetcode      | California |
+-----------+----------+---------------+------------+
```
Output:
```
+-----------+----------+---------------+----------+
| firstName | lastName | city          | state    |
+-----------+----------+---------------+----------+
| Allen     | Wang     | Null          | Null     |
| Bob       | Alice    | New York City | New York |
+-----------+----------+---------------+----------+
```
Explanation:
There is no address in the address table for the personId = 1 so we return null in their city and state.
addressId = 1 contains information about the address of personId = 2.