Name:Vaibhav kumar gupta

Date:-12-02-2025

SQL Concepts: Foreign Key, Primary Key, Normalization, ER, and Procedures

1. Primary Key

A **Primary Key** is a unique identifier for a record in a table. It ensures that each row in the table can be uniquely identified.

Characteristics:

- Must contain unique values.
- Cannot have NULL values.
- A table can have only one primary key, which may consist of single or multiple columns (Composite Key).

Example:

```
CREATE TABLE Employees (
   EmployeeID INT PRIMARY KEY,
   Name VARCHAR(100),
   Department VARCHAR(50)
);
```

---

2. Foreign Key

A **Foreign Key** is a column or set of columns in one table that establishes a relationship with the **Primary Key** of another table.

Characteristics:

- Ensures referential integrity between tables.
- Can have duplicate values.
- Can have NULL values if not marked as NOT NULL.

Example:

```
CREATE TABLE Orders (
   OrderID INT PRIMARY KEY,
   EmployeeID INT,
   OrderDate DATE,
   FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID)
);
```

---

3. Normalization

Normalization is the process of organizing data to reduce redundancy and improve data integrity.

Forms of Normalization:

1. **1NF (First Normal Form)** - Ensures atomicity (each column should contain atomic values) and uniqueness.
2. **2NF (Second Normal Form)** - Follows 1NF and removes partial dependencies (every non-key column must depend on the entire primary key).
3. **3NF (Third Normal Form)** - Follows 2NF and removes transitive dependencies.

---

4. Entity-Relationship (ER) Model

The **ER Model** represents data as **Entities**, **Attributes**, and **Relationships**.

Components of ER Model:

- **Entities:** Objects in the database (e.g., Student, Teacher).
- **Attributes:** Characteristics of an entity (e.g., StudentID, Name).
- **Relationships:** Connections between entities (e.g., "Student Enrolls in Course").

---

5. Stored Procedures

A **Stored Procedure** is a precompiled SQL code that can be executed multiple times.

Advantages:

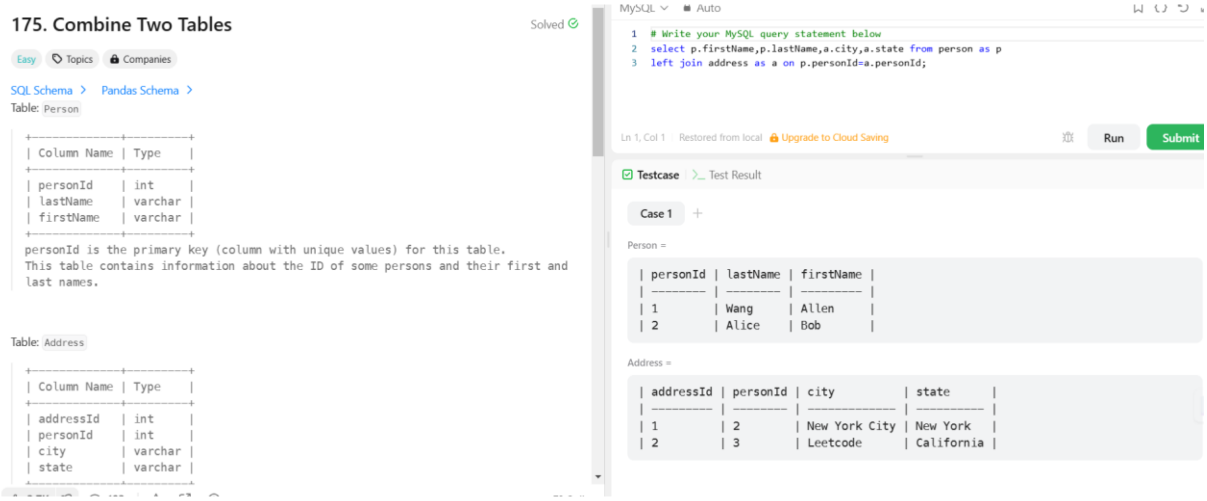
- Improves performance.
- Enhances security.
- Reduces redundancy.

Example:

```
DELIMITER $$

CREATE PROCEDURE GetEmployeeDetails()

BEGIN

   SELECT * FROM Employees;

END $$

DELIMITER ;
```

CALL GetEmployeeDetails();

Coding:-



175. Combine Two Tables

Table: `Person`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| personId    | int     |
| lastName    | varchar |
| firstName   | varchar |
+-------------+---------+
personId is the primary key (column with unique values) 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 |
+-------------+---------+
```

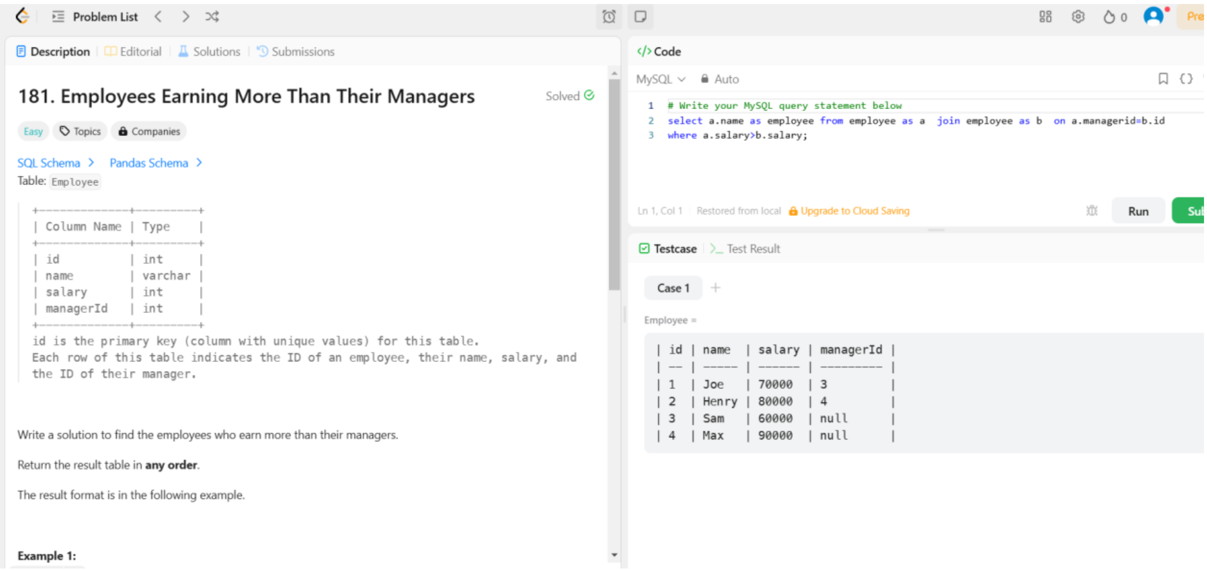
```sql
# Write your MySQL query statement below
select p.firstName,p.lastName,a.city,a.state from person as p
left join address as a on p.personId=a.personId;
```

Person =

```
| personId | lastName | firstName |
| -------- | -------- | --------- |
| 1        | Wang     | Allen     |
| 2        | Alice    | Bob       |
```

Address =

```
| addressId | personId | city          | state      |
| --------- | -------- | ------------- | ---------- |
| 1         | 2        | New York City | New York   |
| 2         | 3        | Leetcode      | California |
```



181. Employees Earning More Than Their Managers

Table: `Employee`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| id          | int     |
| name        | varchar |
| salary      | int     |
| managerId   | int     |
+-------------+---------+
id is the primary key (column with unique values) for this table.
Each row of this table indicates the ID of an employee, their name, salary, and
the ID of their manager.
```

Write a solution to find the employees who earn more than their managers.

Return the result table in **any order**.

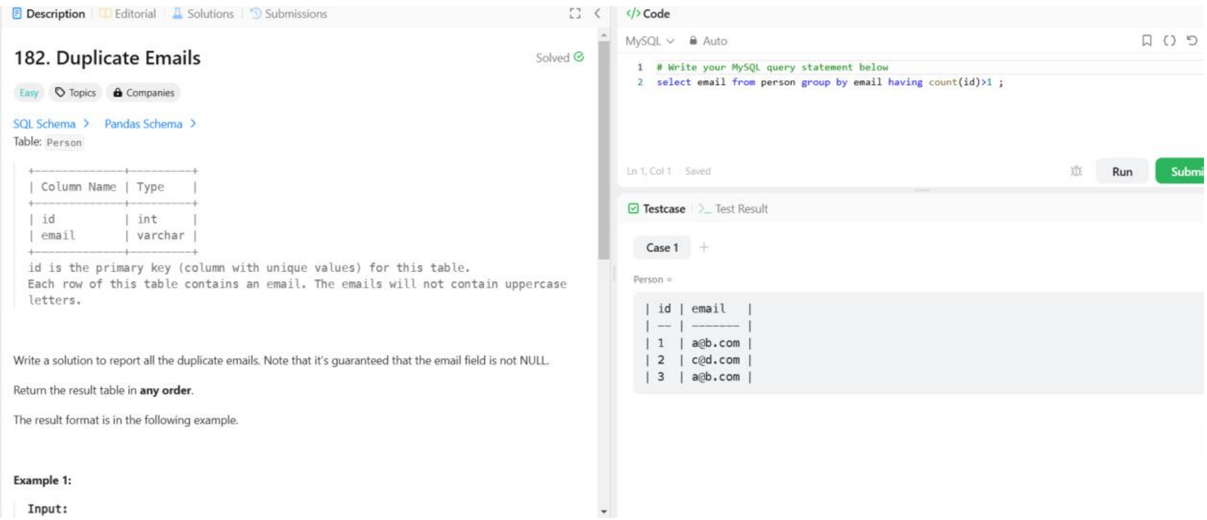
The result format is in the following example.

Example 1:

```sql
# Write your MySQL query statement below
select a.name as employee from employee as a  join employee as b  on a.managerid=b.id
where a.salary>b.salary;
```

Employee =

```
| id | name  | salary | managerId |
| -- | ----- | ------ | --------- |
| 1  | Joe   | 70000  | 3         |
| 2  | Henry | 80000  | 4         |
| 3  | Sam   | 60000  | null      |
| 4  | Max   | 90000  | null      |
```



182. Duplicate Emails

Table: `Person`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| id          | int     |
| email       | varchar |
+-------------+---------+
id is the primary key (column with unique values) for this table.
Each row of this table contains an email. The emails will not contain uppercase
letters.
```

Write a solution to report all the duplicate emails. Note that it's guaranteed that the email field is not NULL.

Return the result table in **any order**.

The result format is in the following example.

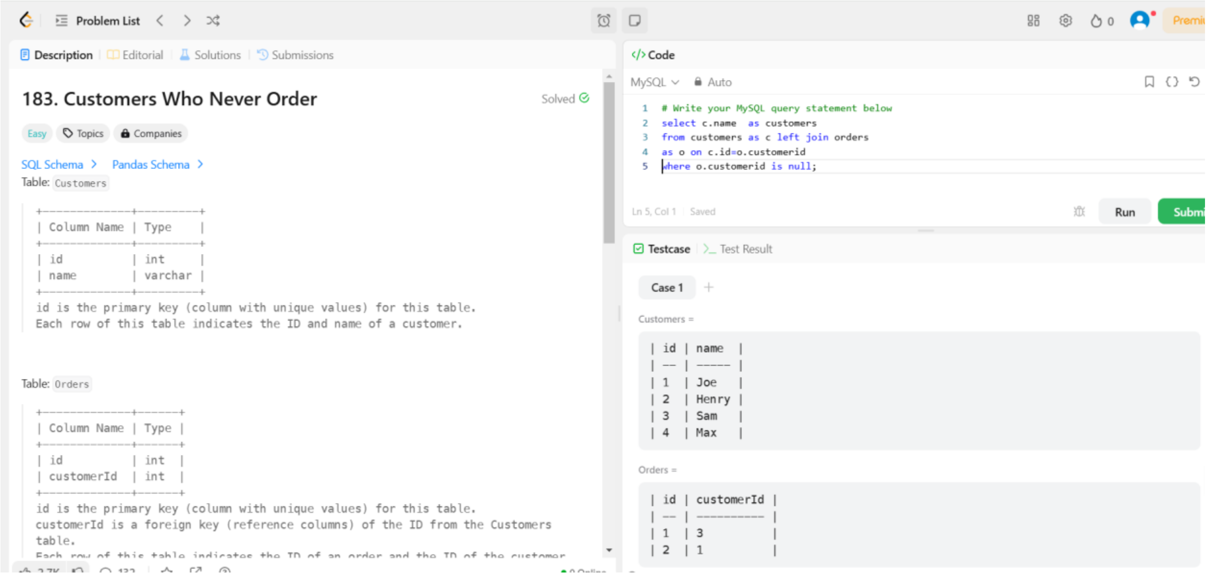
Example 1:

Input:

```sql
# Write your MySQL query statement below
select email from person group by email having count(id)>1 ;
```

Person =

```
| id | email   |
| -- | ------- |
| 1  | a@b.com |
| 2  | c@d.com |
| 3  | a@b.com |
```



183. Customers Who Never Order

SQL Schema > Pandas Schema >

Table: `Customers`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| id          | int     |
| name        | varchar |
+-------------+---------+
id is the primary key (column with unique values) 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 with unique values) for this table.
customerId is a foreign key (reference columns) of the ID from the Customers
table.
Each row of this table indicates the ID of an order and the ID of the customer
```

```sql
# Write your MySQL query statement below
select c.name  as customers
from customers as c left join orders
as o on c.id=o.customerid
where o.customerid is null;
```

Customers =

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

Orders =

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