DATABASE MANAGEMENT SYSTEM

Holiday Assignment - 2311CS020699

1.Game Play Analysis (Solve it in LeetCode)

Table: Activity

Create a Activity table and Insert the given below values and Write a Query for below question :-

1. Write a solution to find the **first login date** for each player from table .

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

The result format is in the following example.

Example 1:

Input:

y table:

r_id	device_id	event_date	games_played
2	2	2016-03-01	5
2	2	2016-05-02	6
3	3	2017-06-25	1
1	1	2016-03-02	0
4	4	2018-07-03	5

Ans.

```
mysql> Select * from Activity;
+-----------+-----------+------------+--------------+
| player_id | device_id | event_date | games_played |
+-----------+-----------+------------+--------------+
|         1 |         2 | 2016-03-01 |            5 |
|         1 |         2 | 2016-05-02 |            6 |
|         2 |         3 | 2017-06-25 |            1 |
|         3 |         1 | 2016-03-02 |            0 |
|         3 |         4 | 2018-07-03 |            5 |
+-----------+-----------+------------+--------------+
5 rows in set (0.02 sec)

mysql> SELECT player_id, Min(event_date) as first_login_date from Activity group by player_id;
+-----------+------------------+
| player_id | first_login_date |
+-----------+------------------+
|         1 | 2016-03-01       |
|         2 | 2017-06-25       |
|         3 | 2016-03-02       |
+-----------+------------------+
3 rows in set (0.00 sec)
```

TASK-2

Find Customer Referee((Solve it in LeetCode)

Find the names of the customer that are **not referred by** the customer with id = 2.

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

Input:

```
+----+ ner table:
       -----+------------+
+----+ ame | referee_id |
       -----+------------+
       ill | null       |
       ane | null       |
 4 |   ex | 2          |
       ill       |
       ack | 1          |
       ark | 2          |
```

Ans.

```
mysql> Select * from customer;
+------+------+-------------+
| id   | name | referee_id |
+------+------+-------------+
|    1 | Will |       NULL |
|    2 | Jane |       NULL |
|    3 | Alex |          2 |
|    4 | Bill |       NULL |
|    5 | Zack |          1 |
|    6 | Mark |          2 |
+------+------+-------------+
6 rows in set (0.00 sec)

mysql> Select name from customer where referee_id != 2 or refere
e_id is null;
+------+
| name |
+------+
| Will |
| Jane |
| Bill |
| Zack |
+------+
4 rows in set (0.00 sec)
```

TASK-3

Big Countries (Solve it in LeetCode)

A country is **big** if:

- it has an area of at least three million (i.e., 3000000 km^2), or
- it has a population of at least twenty-five million (i.e., 25000000).

Write a solution to find the name, population, and area of the **big countries**.

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

Input:

World

```
+-------
          +-----------+---------+------------+--------------+
+-------   | continent | area    | population | gdp          |
          +-----------+---------+------------+--------------+
anistan | Asia      | 652230  | 25500100   | 20343000000  |
ia      | Europe    | 28748   | 2831741    | 12960000000  |
ia      | Africa    | 2381741 | 37100000   | 188681000000 |
+-------rra     | Europe    | 468     | 78115      | 3712000000   |
la      | Africa    | 1246700 | 20609294   | 100990000000 | +-----------+---------+------------+-----------
          --+
```

Ans.

```
mysql> select * from World_table;
+-------------+-----------+---------+------------+--------------+
| name        | continent | area    | population | gdp
|
+-------------+-----------+---------+------------+--------------+
| Afghanistan | Asia      |  652230 |   25500100 |  20343000000 |
| Albania     | Europe    |   28748 |    2831741 |  12960000000 |
| Algeria     | Africa    | 2381741 |   37100000 | 188681000000 |
| Andorra     | Europe    |     468 |      78115 |   3712000000 |
| Angola      | Africa    | 1246700 |   20609294 | 100990000000 |
+-------------+-----------+---------+------------+--------------+
5 rows in set (0.00 sec)

mysql> select name,population,area from World_table where population >= 3000000 or area >= 25000000;
+-------------+------------+---------+
| name        | population | area    |
+-------------+------------+---------+
| Afghanistan |   25500100 |  652230 |
| Algeria     |   37100000 | 2381741 |
| Angola      |   20609294 | 1246700 |
+-------------+------------+---------+
3 rows in set (0.05 sec)
```

TASK-4

Recyclable and low fat products (Solve it in LeetCode)

Write a solution to find the ids of products that are both low fat and recyclable.

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

Input:

cts table:

```
          +----------+------------+
+-------  ict_id  | low_fats | recyclable |
          +----------+------------+
+-------     | Y      | N          |
          | Y      | Y          |
          | N      | Y          |
          | Y      | Y          |
          | N      | N          |
+-------     +----------+------------+
```

Ans.

```
mysql> select * from products;
+------------+----------+------------+
| product_id | low_fats | recyclable |
+------------+----------+------------+
|          0 | Y        | N          |
|          1 | Y        | Y          |
|          2 | N        | Y          |
|          3 | Y        | Y          |
|          4 | N        | N          |
+------------+----------+------------+
5 rows in set (0.00 sec)

mysql> select product_id from products where low_fats == 'Y' a
ERROR 1064 (42000): You have an error in your SQL syntax; chec
t syntax to use near '== 'Y' and recyclable == 'Y'' at line 1
mysql> select product_id from products where low_fats = 'Y' an
+------------+
| product_id |
+------------+
|          1 |
|          3 |
+------------+
2 rows in set (0.00 sec)
```

TASK-5

used in the content of the tweet is **strictly greater** than	Write a solution to find the IDs of the invalid tweets. The tweet is invalid if the number of characters 15.
Input:	

```
s table:
+-------+----------------------------------+
|_id | content                          |
+-------+----------------------------------+
| Let us Code                      |
| More than fifteen chars are here! | +----------------------------------
+-------+
```

Ans.

```
mysql> Select * from Tweets;
+----------+-----------------------------------+
| tweet_id | content                           |
+----------+-----------------------------------+
|        1 | Let us Code                       |
|        2 | More than fifteen chars are here! |
+----------+-----------------------------------+
2 rows in set (0.00 sec)

mysql> select tweet_id from Tweets where lenth(content) > 15;
ERROR 1305 (42000): FUNCTION dbms.lenth does not exist
mysql> select tweet_id from Tweets where length(content) > 15;
+----------+
| tweet_id |
+----------+
|        2 |
+----------+
1 row in set (0.00 sec)
```

Case Study Question: School Database

Scenario:

You are tasked with designing a database for a small school. The school has students, teachers, and classes. The database should help manage the following information:

1. Students' details: Unique ID, name, age, and grade level.
2. Teachers' details: Unique ID, name, and subject specialization.
3. Classes: Each class has a unique ID, subject name, and a teacher assigned.
4. Enrollments: Students enrolled in specific classes.

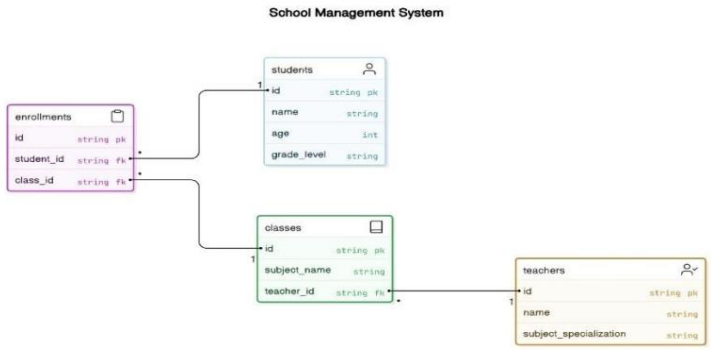
Tasks:

1. **ER Diagram**: Design an ER diagram showing the relationships between Students, Teachers, Classes, and Enrollments.(**Use SmartDraw Tool**)
2. **Schema Design**:
   Write SQL to create the following tables:

- Students (StudentId, Name, Age, GradeLevel) ○ Teachers (TeacherId, Name, SubjectSpecialization)
- Classes (ClassId, SubjectName, TeacherId) ○ Enrollments (EnrollmentId, StudentId, ClassId)

Ans.

1)



2)

```
mysql> desc student;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| studentId  | int         | YES  |     | NULL    |       |
| name       | varchar(35) | YES  |     | NULL    |       |
| age        | int         | YES  |     | NULL    |       |
| gradeLevel | int         | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)

mysql> desc teachers;
+----------------------+-------------+------+-----+---------+-------+
| Field                | Type        | Null | Key | Default | Extra |
+----------------------+-------------+------+-----+---------+-------+
| teacherId            | int         | YES  |     | NULL    |       |
| name                 | varchar(35) | YES  |     | NULL    |       |
| subjectSpecialization | varchar(26) | YES  |     | NULL    |       |
+----------------------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql> desc classes;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| classId     | int         | YES  |     | NULL    |       |
| subjectName | varchar(20) | YES  |     | NULL    |       |
| teacherId   | int         | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql> desc enrollments;
+--------------+------+------+-----+---------+-------+
| Field        | Type | Null | Key | Default | Extra |
+--------------+------+------+-----+---------+-------+
| enrollmentId | int  | YES  |     | NULL    |       |
| studentId    | int  | YES  |     | NULL    |       |
| classId      | int  | YES  |     | NULL    |       |
+--------------+------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

1. Explain different languages present in DBMS.

Ans. In Database Management Systems (DBMS), different types of languages are used to interact with and manipulate data. These languages can be broadly categorized into the following:

1. Data Definition Language (DDL)

- **Purpose**: Used to define the database schema or structure.
- **Functions**: Includes commands to create, alter, and drop tables, views, and indexes.
- **Common Commands**:
    - CREATE: Defines new database objects (e.g., tables, views). ○ ALTER: Modifies existing database objects.
    - DROP: Deletes database objects.
    - TRUNCATE: Removes all records from a table but does not remove the table itself.

2. Data Manipulation Language (DML)

- **Purpose**: Used for querying and modifying data within the database.
- **Functions**: Includes commands for inserting, updating, deleting, and retrieving data.
- **Common Commands**:
    - SELECT: Retrieves data from one or more tables. ○ INSERT: Adds new data into a table.
    - UPDATE: Modifies existing data in a table.
    - DELETE: Removes data from a table.

3. Data Query Language (DQL)

- **Purpose**: A subset of DML, primarily focused on querying data.
- **Functions**: Allows for retrieving data.
- **Common Command**:
    - SELECT: Retrieves specific data from the database according to specified conditions.

4. Data Control Language (DCL)

- **Purpose**: Used to control access and permissions in the database.
- **Functions**: Includes commands for granting and revoking access privileges.
- **Common Commands**:
    - GRANT: Assigns permissions to users.
    - REVOKE: Removes permissions from users.

5. Transaction Control Language (TCL)

- **Purpose**: Used to manage transactions in a DBMS, ensuring data integrity and consistency.
- **Functions**: Includes commands to commit or roll back transactions.
- **Common Commands**:
    - COMMIT: Saves all changes made in the current transaction. ○ ROLLBACK: Undoes the changes made in the current transaction.
    - SAVEPOINT: Sets a point within a transaction to which you can roll back.
    - SET TRANSACTION: Configures the properties of a transaction.

Each of these languages plays a crucial role in managing, manipulating, and securing the data within a database system.

2. What are the different levels of abstraction in the DBMS?

Ans. In a Database Management System (DBMS), the concept of **levels of abstraction** is essential for managing and organizing data in a way that provides simplicity, flexibility, and security. There are three primary levels of abstraction in a DBMS:

1. Physical Level (Internal Level)

- **Description**: This is the lowest level of abstraction, focusing on how data is physically stored in the database system. It deals with the actual storage structures, like files, disk blocks, and data access methods.
- **Purpose**: It defines the internal format of the data, including how it is stored, indexed, and accessed on physical storage media (e.g., hard drives).
- **Key Concepts**:
    - Storage structures such as files and indexes. ○ Data compression techniques and file formats.
    - Access paths like B-trees or hashing.

2. Logical Level (Conceptual Level)

- **Description**: This level represents the logical structure of the data, independent of the physical storage. It defines what data is stored, and the relationships among the data, without specifying how it is stored.
- **Purpose**: It provides an abstract view of the entire database and how the data is logically organized (tables, views, relationships, etc.).
- **Key Concepts**:

- Entities, attributes, and relationships (e.g., tables, keys). ○ Integrity constraints (e.g., primary keys, foreign keys).
- Database schema and data models (e.g., relational model, object-oriented model).

3. View Level (External Level)

- **Description**: The highest level of abstraction, focusing on how users or applications interact with the data. It provides different views of the database tailored to specific user needs.
- **Purpose**: It defines the different perspectives or user-specific views of the data, often involving restrictions or filters to present only the relevant data to users.
- **Key Concepts**:
  - User-specific views (e.g., a customer view showing only their details). ○ Virtual tables or views (i.e., derived from the logical level but not physically stored).
  - Access control (ensuring users only see data they are authorized to view).

3. What are the different data models?

Ans. In a Database Management System (DBMS), a **data model** defines how data is structured, stored, and manipulated. It provides an abstraction for managing data in a database. There are several types of data models, each with its own approach to organizing and representing data. Here are the most common types:

1. Hierarchical Data Model

- **Description**: This model organizes data in a tree-like structure, where each record has a single parent and potentially many children (parent-child relationships).
- **Structure**: It represents data in a hierarchy with nodes, where each node represents a record or entity, and edges represent the relationship between them.
- **Example**: An organizational chart where each department is a parent and employees within the department are children.
- **Advantages**:
  - Efficient for representing data with a clear hierarchical relationship.
  - Fast retrieval of hierarchical data.
- **Disadvantages**:
  - Lack of flexibility in handling complex relationships.
  - Difficult to reorganize data once established.

2. Network Data Model

- **Description**: The network model is an extension of the hierarchical model. It allows more complex relationships by allowing each record to have multiple parent and child relationships (many-to-many relationships).
- **Structure**: Data is organized in a graph structure with nodes representing entities and edges representing relationships. Each record can have multiple parent-child links.
- **Example**: A university database where students can enroll in multiple courses, and a course can have multiple students.
- **Advantages**:
    - Supports many-to-many relationships.
    - More flexible than the hierarchical model.
- **Disadvantages**:
    - Complexity in implementation and navigation.
    - Requires specialized knowledge to query and maintain.

3. Relational Data Model

- **Description**: This is the most widely used data model. It organizes data into tables (relations), where each table consists of rows (records) and columns (attributes).
- **Structure**: Data is represented in tables, with each table having a primary key that uniquely identifies each record. Relationships between tables are established using foreign keys.
- **Example**: A student table with attributes like student ID, name, and course ID, and a course table with course ID and course name.
- **Advantages**:
    - Simple and intuitive design.
    - Supports powerful querying using SQL (Structured Query Language).
    - Data independence and flexibility.
- **Disadvantages**:
    - Performance can degrade with very large datasets. ○ Complex relationships may require multiple tables and joins.

4. Object-Oriented Data Model

- **Description**: This model integrates object-oriented programming principles into databases. Data is represented as objects, similar to how data is structured in object-oriented programming languages.
- **Structure**: Objects have attributes (properties) and methods (functions) that operate on the data. Objects can inherit properties and methods from other objects (inheritance).
- **Example**: A customer object with attributes like name and address, and methods like placeOrder().
- **Advantages**:
    - Supports complex data types and relationships.
    - Provides more flexibility and closer mapping to real-world entities.
- **Disadvantages**:
    - Less mature and widely adopted than relational models.
    - Complex to implement and manage.

5. Entity-Relationship (ER) Model

- **Description**: The ER model is primarily a conceptual model used for designing databases. It uses entities (objects) and relationships to model real-world systems.
- **Structure**: Entities are represented by rectangles, relationships by diamonds, and attributes by ovals. The relationships between entities are depicted by connecting lines.
- **Example**: An ER diagram for a library system, where "Book" and "Author" are entities, and "writtenBy" is the relationship between them.
- **Advantages**:
    - Useful for high-level database design.
    - Simple, easy-to-understand graphical representation.
- **Disadvantages**:
    - Does not specify how data is stored or queried, so it requires conversion to another model (e.g., relational) for implementation.

6. Document Data Model

- **Description**: This model is used in document-based databases, where data is stored in documents rather than tables. Each document is typically represented in formats such as JSON, XML, or BSON.

- **Structure**: Documents can contain nested structures, arrays, and key-value pairs. Each document is self-contained and can represent complex data types.
- **Example**: A MongoDB database stores each user profile as a document, with attributes like name, age, and address.
- **Advantages**:
    - Flexible schema, ideal for unstructured or semi-structured data.
    - Efficient for storing and retrieving JSON-like data.
- **Disadvantages**:
    - Less efficient for handling highly relational data.
    - Can lead to data duplication and integrity issues.

7. Key-Value Data Model

- **Description**: In the key-value data model, data is stored as key-value pairs. Each key is unique, and it maps to a value, which can be a simple value, a set, or a complex object.
- **Structure**: Data is represented as a collection of key-value pairs, often stored in key-value stores like Redis or DynamoDB.
- **Example**: A caching system storing user session data, where each session ID is a key, and the corresponding value is the user data.
- **Advantages**:
    - Simple and fast for storing and retrieving data.
    - Highly scalable and suitable for distributed systems.
- **Disadvantages**:
    - Lack of structure and querying capabilities.
    - Limited support for complex relationships.

8. Columnar Data Model

- **Description**: This model organizes data into columns rather than rows, making it more efficient for read-heavy applications, especially in analytical databases.
- **Structure**: Data is stored in columns, where each column represents an attribute, and all values for that attribute are stored together.
- **Example**: Apache HBase and Google Bigtable are columnar databases, suitable for analytical workloads where large datasets need to be scanned quickly.

- **Advantages**:
    - Efficient for read-heavy workloads and large-scale data analytics.
    - Enables better compression and faster query performance on specific columns.
- **Disadvantages**:
    - Not ideal for transactional workloads. ○ Requires specialized query techniques for optimal performance.