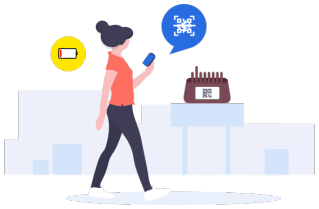

COMP3278 Introduction to Database Management Systems
Assignment1 Part2
Database Application Development for the ABC PowerBank Rental Service
Due Date:
2025 Mar 23 (Sunday) 17:30 pm

The ABC company provides rental service of portable power banks to its registered members. You are a database designer of the ABC company, and you are tasked with designing the database to support their rental management system. The database will track member registrations, power bank inventory, rental transactions, and service locations. The database schema solution shown below has been adopted by the company. The schemas are shown in the following tables, where underlined attributes represent the primary key of each relation:

- **Member** (member_ID, name, email, contact_number)
- **ServiceArea** (service_area_ID, name, parent_area_ID)
  - parent_area_ID: References ServiceArea.service_area_ID, creating a hierarchical structure of service areas.
  - parent_area_ID can accept NULL value (for top-level service areas).
- **ChargingStation** (station_ID, name, location_ID, available_pbs)
  - location_ID: References ServiceArea.service_area_ID, indicating which service area this charging station belongs to.
  - available_pbs: Represents the current count of available power banks at this charging station.
- **PowerBank** (pb_ID, station_ID, status)
  - station_ID: References ChargingStation.station_ID, indicating the charging station where the power bank is located.
  - station_ID can accept NULL value (for power banks not currently at any charging station, e.g., being rented out).
  - status: Indicates the availability of the power bank. There are two possible values: 0 indicates the power bank is unavailable, and 1 indicates the power bank is available in the charging station.
- **Coupon** (coupon_ID, redemption, expiration_date, discount_value, member_ID)
  - member_ID: References Member.member_ID, associating the coupon with a member.
  - redemption: There are two possible values: 0 indicates the coupon hasn't been redeemed, and 1 indicates the coupon has been redeemed.
  - discount_value: The monetary value or percentage discount provided by this coupon.
  - expiration_date: The date after which the coupon becomes invalid if unused.
- **RentalTransaction** (transaction_ID, member_ID, pb_ID, station_ID, start_datetime, end_datetime, payment_amount, status, coupon_ID)
  - member_ID: References Member.member_ID, indicating the member involved in the rental.
  - pb_ID: References PowerBank.pb_ID, indicating the rented power bank.
  - station_ID: References ChargingStation.station_ID, indicating where the power

bank was rented from.
    - start_datetime: When the rental transaction began.
    - end_datetime: When the power bank was returned, can be NULL for ongoing rentals.
    - payment_amount: The cost of the rental, can be NULL for ongoing transactions.
    - coupon_ID: References Coupon.coupon_ID, identifying any coupon applied to this rental.
    - status: Indicates the status of the transaction - 0 indicates an ongoing transaction, and 1 indicates a completed transaction.
- **Reservation** (reservation_ID, member_ID, station_ID, reservation_datetime, collect_datetime, status)
    - member_ID: References Member.member_ID, indicating the member who made the reservation.
    - station_ID: References ChargingStation.station_ID, indicating which charging station is reserved.
    - reservation_datetime: The time when the reservation was made. The reservation is only on hold for 30 mins.
    - collect_datetime: The time when the user collected the power bank from the reserved charging station.
    - status: There are three possible values: 0 indicates the reservation is ongoing, 1 indicates the reservation is completed (the power bank has been collected from the reserved charging station), and 2 indicates the reservation has been canceled.

Requirements

1. [20%] Build the database using MySQL.
    - Using the schemas above, define tables with appropriate constraints.
    - Store the corresponding table definition commands in an SQL file called "tables.sql" (Please include also the referential constraints in the .sql file, make sure that the files can correctly build the necessary tables and constraints when imported to another database).
2. [80%] Answer the following SQL queries and submit as separate SQL files
    - Each query should be saved as a separate SQL file named q1.sql, q2.sql, etc.
    - Make sure your SQL statements produce the exact column names specified in each query.
    - The SQL queries should follow the requirements listed below:

Q1. [8%] Display the *station_ID* and *name* for the charging stations where the *name* begins with "Pokfulam".

- Case-sensitive string matching.
- Sort the results by *station_ID* in ascending order.

Q2. [8%] For the coupons that have been redeemed, display the *coupon_ID*, *discount_value* of coupons, the *member_ID*, and the *name* of the member as well as the *payment_amount* of the involved transaction.

- Sort the records by the *payment_amount* of the corresponding redeemed transaction in descending order, then ascending order of *member_ID*.

Q3. [10%] Display all charging stations (*station_ID*, *name, available_pbs*) and the *name* of the ServiceArea.

- *available_pbs* is the number of *PowerBanks* available in the charging station, this number can be 0.
- Sort the records in descending order of *available_pbs*, then ascending order of *station_ID*.

Q4. [10%] Display the *member_ID*, *name*, and *rental_count* for members who have completed at least one rental transaction.

- *rental_count* is the total number of rental transactions completed by the member.
- Sort the records in descending order of *rental_count*, then ascending order of *member_ID*.

Q5. [12%] Display the *station_ID*, *station_name*, *sum_of_completed_reservation* of the *Reservation* with the *ChargingStation*(s) in the ServiceArea of "Central and West".

- The *ChargingStation*(s) in the sub-ServiceArea(s) of "Central and West" will be included in the result.
- *sum_of_completed_reservation* is the number of the completed reservation for the *ChargingStation*, this number can be 0.
- To simplify this question, you can assume that there is at most a 3-layer hierarchy structure of the *ServiceArea*.
- Sort the results by *sum_of_completed_reservation* in descending order, then *station_ID* in ascending order.

Q6. [12%] Display *station_ID*, *station_name*, and *rental_income* for each of the charging stations in February 2024.

- Sort the results by *rental_income* in descending order, then *station_ID* in ascending order.
- *rental_income* is the sum of the *payment_amount* in a given period.

Q7. [10%] Display all reservations (*reservation_ID*, *member_ID*, *member_name*, *station_ID*, *reservation_datetime, collect_datetime, email*) that have been collected within 15 mins from the *reservation_datetime*.

- *email* is the email of the member making the reservation.
- Sort by *reservation_datetime* in ascending order, then *reservation_ID* in descending order.
- You may use TIMESTAMPDIFF() to solve this question. For example, TIMESTAMPDIFF(MINUTE, '2023-01-01 10:00:00', '2023-01-01 12:30:00') calculates the difference in minutes between 10:00 AM and 12:30 PM on January 1st, 2023, returning 150.

Q8. [10%] Write a query to list all the *service_area_name* and *charging_station_count* of the charging stations on the *ServiceArea* without a parent-service area.

- *service_area_name* is the name of the *SeriveArea*.
- We can assume that there is at most a 3-layer hierarchy structure of the *ServiceArea*.
- *charging_station_count* is the total number of charging stations in the *ServiceArea*.
- Only list the record(s) with *station_count* >= 1.
- Display results in the format of "*service_area_name*: *charging_station_count*".
- Sort the results by the *charging_station_count* in descending order, and then ascending by *service_area_ID*.

Q9. [10%] For a given charging station (assume station_ID = 3 for this query), display the *reservation_ID*, *member_ID*, *reservation_datetime*, and member's contact information (*name*, *email*, *contact_number*) for all completed reservations of the selected charging station.

- Sort by *reservation_datetime* in ascending order, then ascending by *member_ID*.
- Your SQL file should work if we change the station_ID value.

Files to Submit

You must submit exactly 10 SQL files through the Moodle VPL system:

1. tables.sql - Contains all table creation statements
2. q1.sql - Query for Question 1
3. q2.sql - Query for Question 2
4. q3.sql - Query for Question 3
5. q4.sql - Query for Question 4
6. q5.sql - Query for Question 5
7. q6.sql - Query for Question 6
8. q7.sql - Query for Question 7
9. q8.sql - Query for Question 8
10. q9.sql - Query for Question 9

VPL Submission Guidelines

You can either:

- Upload your SQL files using the VPL interface in Moodle, OR
- Write your SQL code directly in the VPL editor

After coding your solutions, click the "Evaluate" button to run the basic test cases

You can make multiple submissions before the deadline to improve your solutions

Important: **The VPL system provides basic test cases to help you verify your solutions and your final grade will be determined by additional hidden test cases that will be used for evaluation**

Make sure your solutions work correctly for all potential valid inputs, not just the sample test cases

SQL File Format Requirements

For all SQL files:

- Use UTF-8 encoding
- Each file must contain exactly ONE SQL statement (ending with a semicolon)
- Do not include comments in your SQL files
- Do not include SET commands or other database configurations

For tables.sql:

- Follow the exact database schema specified in the assignment
- Define all primary keys and foreign keys as required
- Ensure your tables have the correct field names and constraints

For query files (q1.sql to q9.sql):

- Each file should contain exactly one SQL query that answers the corresponding question
- Ensure your query produces output columns with EXACTLY the column names specified in the assignment
- Follow the sorting requirements specified in each question
- Do not use views or temporary tables in your queries

Submission Format Example

Your files should look like this:

tables.sql:

```
CREATE TABLE Member (
        …
);
-- Additional tables with appropriate constraints
```

q1.sql:

```
SELECT …
```

FROM …
WHERE …
ORDER BY …

Please Submit your work through Moodle before the deadline. Late submissions will not be accepted.

If you encounter difficulty in this assignment, please feel free to post your questions on the Moodle forum or contact us. We are very happy to help.

We wish you enjoy learning database technologies in this course!