Bike rental shop - SQL Case study

Introduction:

Emily is the shop owner, and she would like to gather data to help her grow the business. She has hired you as an SQL specialist to get the answers to her business questions such as How many bikes does the shop own by category? What was the rental revenue for each month? etc. The answers are hidden in the database. You just need to figure out how to get them out using SQL.

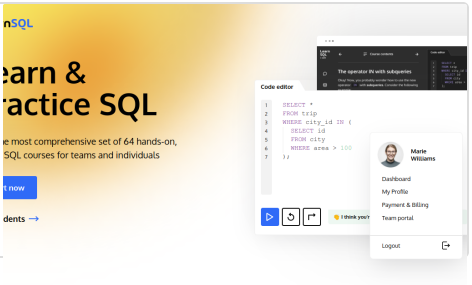
Case Study URL

This case study is taken from LearnSQL.com. It is part of the "November 2023 SQL Challenge" course 👇

Log in | LearnSQL.com

Learn the SQL standard and other SQL dialects comprehensively or simply upskill yourself with our interactive online SQL courses.

https://learnsql.com/course/2023-advanced-sql-practice-challenges/november-challenge/introduction/introduction/



Understanding the database:

The shop's database consists of 5 tables:

- `customer`
- `bike`
- `rental`
- `membership_type`
- `membership`

The customer table

In the first table, `customer`, you'll find details about the customers of the bike rental shop. This table contains the following columns:

- `id`: The unique ID of each customer.
- `name`: The customer's name.
- `email`: The customer's email address.

The bike table

In the `bike` table, you'll find information about bikes the rental shop owns.
This table contains the following columns:

- `id`: The unique ID of the bike.
- `model`: The model of the bike.
- `category`: The type of bike (e.g., mountain bike, road bike, hybrid, electric).
- `price_per_hour`: The rental price per hour for the bike.
- `price_per_day`: The rental price per day for the bike.
- `status`: The status of the bike (available, rented, out of service).

The rental table

The `rental` table matches customers with bikes they have rented. This table has the following columns:

- `id`: The unique ID of the rental entry.
- `customer_id`: The ID of the customer who rented the bike.
- `bike_id`: The ID of the bike rented.
- `start_timestamp`: The date and time when the rental started.
- `duration`: The duration of the rental in minutes.
- `total_paid`: The total amount paid for the rental.

The membership_type table

The `membership_type` table has information about the different membership types for purchase. This table contains the following columns:

- `id`: The unique ID of the membership type.
- `name`: The name of the membership type.
- `description`: A description of the membership type.
- `price`: The price of the membership type.

The membership table

The `membership` table has information about individual memberships purchased by customers. This table contains the following columns:

- `id`: The unique ID of the membership.
- `membership_type_id`: The ID of the membership type purchased.
- `customer_id`: The ID of the customer who purchased the membership.
- `start_date`: The start date of the membership.
- `end_date`: The end date of the membership.
- `total_paid`: The total amount paid for the membership.

**** You will find the sql scripts to create this dataset in the txt file*

"DATASET-Bike_Rental_Case_Study.txt"

Problem Statements

Following are the business questions that Emily wants answers to. Use SQL to answer them. All the best.

1. Emily would like to know **how many bikes the shop owns by category**. Can you get this for her?

   Display the **category name** and the **number of bikes the shop owns in each category** (call this column `number_of_bikes`). Show only the categories where the number of bikes is greater than `2`.

2. Emily needs a list of **customer names with the total number of memberships purchased by each**.

   For each customer, display the **customer's name** and the **count of memberships purchased** (call this column `membership_count`). Sort the results by `membership_count`, starting with the customer who has purchased the highest number of memberships.

   Keep in mind that some customers may not have purchased any memberships yet. In such a situation, display `0` for the `membership_count`.

3. Emily is working on a special offer for the winter months. Can you help her prepare a list of **new rental prices**?

   For each bike, display its **ID**, **category**, **old price per hour** (call this column `old_price_per_hour`), **discounted price per hour** (call it `new_price_per_hour`), **old price per day** (call it `old_price_per_day`), and **discounted price per day** (call it `new_price_per_day`).

   Electric bikes should have a **10% discount for hourly** rentals and a **20% discount for daily** rentals. Mountain bikes should have a **20% discount for hourly** rentals and a **50% discount for daily** rentals. All other bikes should have a **50% discount** for all types of rentals.

   Round the new prices to `2` decimal digits.

4. Emily is looking for **counts of the rented bikes and of the available bikes in each category**.

   Display the **number of available bikes** (call this column `available_bikes_count`) and the **number of rented bikes** (call this column `rented_bikes_count`) by bike category.

5. Emily is preparing a sales report. She needs to know the **total revenue from rentals by month, the total by year, and the all-time across all the years**.

Display the total revenue from rentals for each month, the total for each year, and the total across all the years. **Do not take memberships into account.** There should be 3 columns: `year`, `month`, and `revenue`.

Sort the results **chronologically**. Display the year total after all the month totals for the corresponding year. Show the all-time total as the last row.

The resulting table looks something like this:

year	month	revenue
2022	11	200.00
2022	12	150.00
2022	null	350.00
2023	1	110.00
...		
2023	10	335.00
2023	null	1370.00
null	null	1720.00

6. Emily has asked you to get the **total revenue from memberships** for each combination of year, month, and membership type.

   Display the **year**, the **month**, the name of the **membership type** (call this column `membership_type_name`), and the **total revenue** (call this column `total_revenue`) for every combination of year, month, and membership type. Sort the results by year, month, and name of membership type.

7. Next, Emily would like data about **memberships purchased in 2023**, with subtotals and grand totals for all the different combinations of membership types and months.

   Display the **total revenue from memberships purchased in 2023 for each combination of month and membership type**. Generate subtotals and grand totals for all possible combinations. There should be 3 columns: `membership_type_name`, `month`, and `total_revenue`.

   Sort the results by membership type name **alphabetically** and then **chronologically** by month.

8. Now it's time for the final task.

   Emily wants to **segment customers based on the number of rentals** and see the **count of customers in each segment**. Use your SQL skills to get this!

   Categorize customers based on their rental history as follows:

   - Customers who have had more than 10 rentals are categorized as `'more than 10'`.
   - Customers who have had 5 to 10 rentals (inclusive) are categorized as `'between 5 and 10'`.
   - Customers who have had fewer than 5 rentals should be categorized as `'fewer than 5'`.

   Calculate the number of customers in each category. Display two columns: `rental_count_category` (the rental count category) and `customer_count` (the number of customers in each category).

*** Solutions to all these problems can be found in

“SOLUTION-Bike_Rental_Case_Study.txt” file.