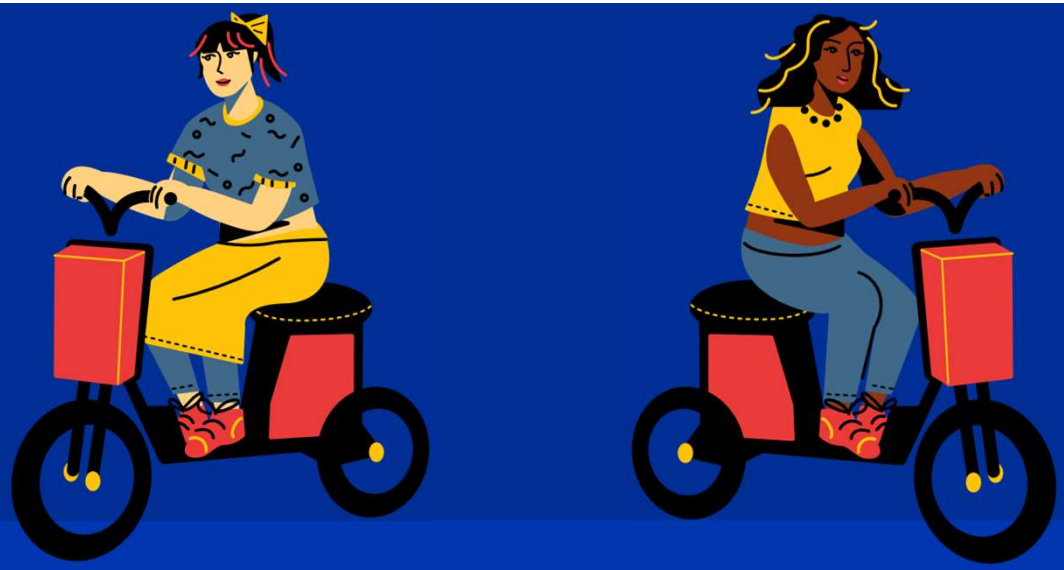
AnkitAnalyst1407

RENTAL BIKES 2024

SQL-PROJECT

AnkitAnalyst1407

CONTENT
RENTAL-BIKES



ABOUT RENTAL-BIKES..

Rental Bikes is a bike sharing service where customers can rent various types of bikes (like electric or mountain bikes) for specified duration, with pricing based on hourly or daily rates.

SQL powers this system by efficiently managing and retrieving data, allowing the rental shop to track bikes availability, customer usage, revenue and inventory- all essential for smooth and organized operations.

RAW DATA

-DataBase and all tables are manually made then data was added in SQL

```sql
CREATE DATABASE
RENTAL_BIKES_2024;
USE RENTAL_BIKES_2024;
drop table if exists customer;
create table customer
(
    id      int primary key,
    name    varchar(30),
    email   varchar(50)
);

drop table if exists bike;
create table bike
(
    id                  int primary key,
    model               varchar(50),
    category            varchar(50),
    price_per_hour      decimal,
    price_per_day       decimal,
    status              varchar(20)
);
drop table if exists rental;
create table rental
```

```sql
);
drop table if exists rental;
create table rental
(
    id                  int primary key,
    customer_id         int references customer(id),
    bike_id             int references bike(id),
    start_timestamp     timestamp,
    duration            int,
    total_paid          decimal
);
drop table if exists membership_type;
create table membership_type
(
    id              int primary key,
    name            varchar(50),
    description     varchar(500),
    price           decimal
);

drop table if exists membership;
create table membership
(
```

```sql

drop table if exists membership;
create table membership
(
    id                  int primary key,
    membership_type_id  int references membership_type(id),
    customer_id         int references customer(id),
    start_date          date,
    end_date            date,
    total_paid          decimal
);
insert into customer values(1,  'John Doe',     'john.doe@ex
insert into customer values(2,  'Alice Smith',  'alice.smith@
insert into customer values(3,  'Bob Johnson',  'bob.johnson
insert into customer values(4,  'Eva Brown',    'eva.brown@
insert into customer values(5,  'Michael Lee',  'michael.le
insert into customer values(6,  'Sarah White',  'sarah.white@ex
insert into customer values(7,  'David Wilson', 'david.wilson@ex
insert into customer values(8,  'Emily Davis',  'emily.davis@exam
insert into customer values(9,  'Daniel Miller', 'daniel.miller@
insert into customer values(10, 'Olivia Taylor', 'olivia.taylor@ex
```

SQL QUES ..

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).

SQL QUERIES THE SOLUTION

```sql
-- Emily would like to know how many bikes the shop
USE RENTAL_BIKES_2024;
SELECT CATEGORY, COUNT(*) AS NUMBER_OF_BIKE
FROM BIKE
GROUP BY CATEGORY;
```

Result Grid | Filter Rows:

CATEGORY	NUMBER_OF_BIKE
mountain bike	3
road bike	3
hybrid	2
electric	2

SQL QUERIES THE SOLUTION

```sql
-- SHOW ONLY WHERE NUMBER IS GREATER THAN 2.
SELECT CATEGORY, COUNT(*) AS NUMBER_OF_BIKE
FROM BIKE
GROUP  BY CATEGORY
HAVING COUNT(*)>2 ;
```

Result Grid | Filter Rows:

CATEGORY	NUMBER_OF_BIKE
mountain bike	3
road bike	3

SQL QUERIES THE SOLUTION

```sql
-- Emily needs a list of customer names with the t
SELECT NAME AS CUSTOMER_NAME,
COUNT(MEMBERSHIP.ID) AS MEMBERSHIP_COUNT
FROM CUSTOMER LEFT JOIN MEMBERSHIP
ON CUSTOMER.ID = MEMBERSHIP.CUSTOMER_ID
GROUP BY CUSTOMER.NAME
ORDER BY MEMBERSHIP_COUNT DESC;
```

Result Grid | Filter Rows:

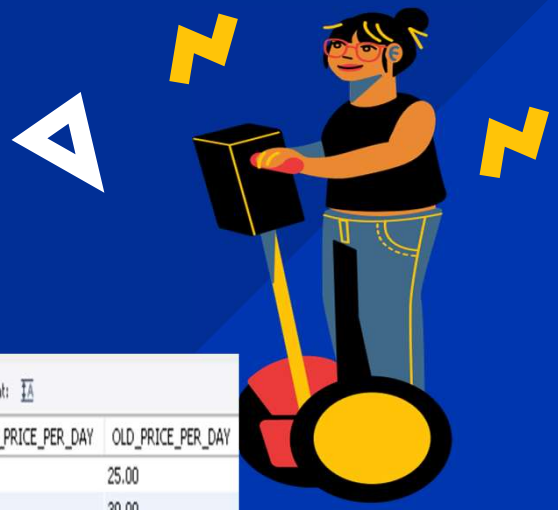
CUSTOMER_NAME	MEMBERSHIP_COUNT
Alice Smith	3
Bob Johnson	3
John Doe	2
Eva Brown	2
Michael Lee	2
Sarah White	0
David Wilson	0
Emily Davis	0
Daniel Miller	0
Olivia Taylor	0

SQL QUERIES THE SOLUTION

```sql
-- Emily is working on a special offer for the winter months. (
SELECT ID, CATEGORY,
PRICE_PER_HOUR AS OLD_PRICE_PER_HOUR, ROUND (
CASE
WHEN CATEGORY = 'ELECTRIC BIKE' THEN PRICE_PER_HOUR * 0.90
WHEN CATEGORY = 'MOUNTAIN BIKE' THEN PRICE_PER_HOUR * 0.80
ELSE PRICE_PER_HOUR * 0.50
END, 2 ) as NEW_PRICE_PER_HOUR,
PRICE_PER_DAY AS OLD_PRICE_PER_DAY, ROUND(
CASE
WHEN CATEGORY = "ELECTRIC BIKE" THEN PRICE_PER_DAY * 0.80
WHEN CATEGORY = "MOUNTAIN BIKE" THEN PRICE_PER_DAY * 0.50
ELSE PRICE_PER_DAY * 0.50
END , 2 ) AS OLD_PRICE_PER_DAY
FROM BIKE;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ID	CATEGORY	OLD_PRICE_PER_HOUR	NEW_PRICE_PER_HOUR	OLD_PRICE_PER_DAY	OLD_PRICE_PER_DAY
1	mountain bike	10	8.00	50	25.00
2	road bike	12	6.00	60	30.00
3	hybrid	8	4.00	40	20.00
4	electric	15	7.50	75	37.50
5	mountain bike	10	8.00	50	25.00
6	road bike	12	6.00	60	30.00
7	hybrid	8	4.00	40	20.00
8	electric	15	7.50	75	37.50
9	mountain bike	10	8.00	50	25.00
10	road bike	12	6.00	60	30.00



SQL QUERIES THE SOLUTION

```sql
-- Emily is looking for counts of the rented bikes and of the available bikes in each category.
USE RENTAL_BIKES_2024;
SELECT CATEGORY,COUNT(
CASE
WHEN STATUS ="AVAILABLE" THEN 1 END) AS AVAILABLE_BIKE_COUNT,
COUNT(
CASE
WHEN STATUS= "RENTED" THEN 1 END) AS RENTED_BIKE_COUNT
FROM BIKE
GROUP BY CATEGORY;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

CATEGORY	AVAILABLE_BIKE_COUNT	RENTED_BIKE_COUNT
mountain bike	1	1
road bike	3	0
hybrid	0	1
electric	2	0

SQL QUERIES THE SOLUTION

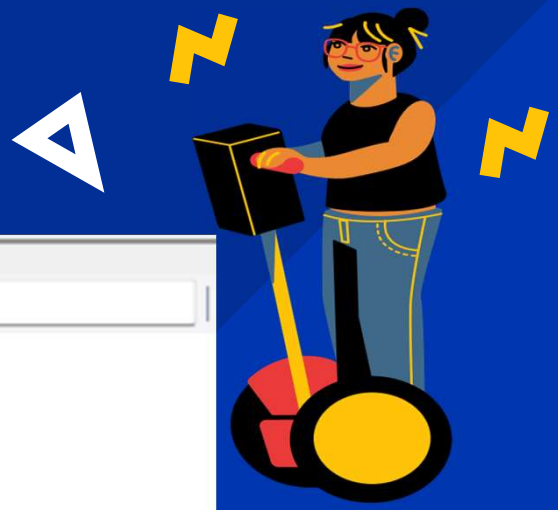
```sql
USE RENTAL_BIKES_2024;
SELECT
EXTRACT(YEAR FROM START_TIMESTAMP) AS YEAR,
EXTRACT(MONTH FROM START_TIMESTAMP) AS MONTH,
SUM(TOTAL_PAID) AS REVENUE
FROM
RENTAL
GROUP BY YEAR, MONTH

UNION ALL
SELECT
EXTRACT(YEAR FROM START_TIMESTAMP) AS YEAR,
NULL AS MONTH,
SUM(TOTAL_PAID) AS REVENUE
FROM
RENTAL
GROUP BY YEAR

UNION ALL
SELECT
NULL AS YEAR,
NULL AS MONTH,
SUM(TOTAL_PAID) AS REVENUE
FROM RENTAL
ORDER BY YEAR, month;
```

Result Grid | Filter Rows:

YEAR	MONTH	REVENUE
NULL	NULL	1720
2022	NULL	350
2022	11	200
2022	12	150
2023	NULL	1370
2023	1	110
2023	2	40
2023	3	110
2023	4	90
2023	5	120
2023	6	115
2023	7	150
2023	8	125
2023	9	175
2023	10	335



SQL QUERIES THE SOLUTION

```sql
-- Emily has asked you to get the total revenue from members
USE RENTAL_BIKES_2024;
SELECT EXTRACT( YEAR FROM START_DATE) AS YEAR,
EXTRACT(MONTH FROM START_DATE) AS MONTH,
NAME AS MEMBERSHIP_TYPE_NAME,
(TOTAL_PAID) AS TOTAL_RENVENUE
FROM MEMBERSHIP
JOIN MEMBERSHIP_TYPE
ON MEMBERSHIP.MEMBERSHIP_TYPE_ID = MEMBERSHIP_TYPE.ID
ORDER BY YEAR, MONTH, MEMBERSHIP_TYPE_NAME;
```

Result Grid | Filter Rows: | Export: | Wrap

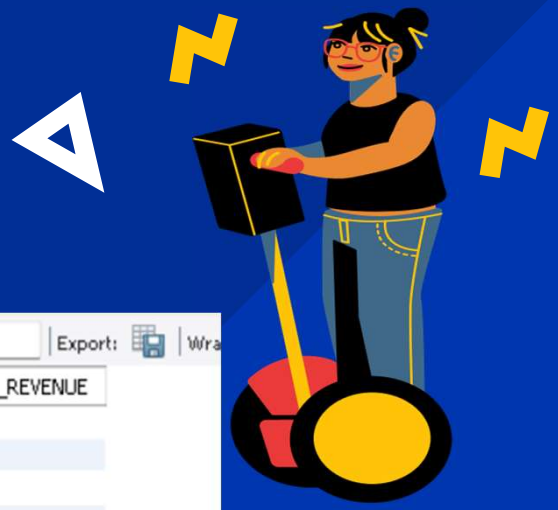
YEAR	MONTH	MEMBERSHIP_TYPE_NAME	TOTAL_RENVENUE
2023	8	Basic Annual	500
2023	8	Basic Monthly	100
2023	8	Premium Monthly	200
2023	9	Basic Annual	500
2023	9	Basic Monthly	100
2023	9	Premium Monthly	200
2023	10	Basic Annual	500
2023	10	Basic Monthly	100
2023	10	Premium Monthly	200
2023	11	Basic Annual	500
2023	11	Basic Monthly	100
2023	11	Premium Monthly	200

SQL QUERIES THE SOLUTION

```sql
-- Emily would like data about memberships purchased in 2023, w
-- types and month
USE RENTAL_BIKES_2024;
SELECT NAME AS MEMBERSHIP_TYPE_NAME,
EXTRACT(MONTH FROM START_DATE) AS MONTH,
SUM(TOTAL_PAID) AS TOTAL_REVENUE
FROM MEMBERSHIP JOIN MEMBERSHIP_TYPE
ON MEMBERSHIP.MEMBERSHIP_TYPE_ID = MEMBERSHIP_TYPE.ID
WHERE EXTRACT( YEAR FROM MEMBERSHIP.START_DATE) = 2023
GROUP BY MEMBERSHIP_TYPE_NAME, month
WITH ROLLUP
ORDER BY MEMBERSHIP_TYPE_NAME, MONTH;
```

Result Grid | Filter Rows: | Export: | Wra

MEMBERSHIP_TYPE_NAME	MONTH	TOTAL_REVENUE
NULL	NULL	3200
Basic Annual	NULL	2000
Basic Annual	8	500
Basic Annual	9	500
Basic Annual	10	500
Basic Annual	11	500
Basic Monthly	NULL	400
Basic Monthly	8	100
Basic Monthly	9	100
Basic Monthly	10	100
Basic Monthly	11	100
Premium Monthly	NULL	800
Premium Monthly	8	200
Premium Monthly	9	200
Premium Monthly	10	200



SQL QUERIES THE SOLUTION

```sql
-- Emily wants to segment customers based on the number of rentals and
-- see the count
USE RENTAL_BIKES_2024;
SELECT RENTAL_COUNT_CATEGORY, COUNT(*) AS CUSTOMER_COUNT
FROM(
SELECT
CASE
WHEN COUNT(RENTAL.ID) > 10 THEN 'MORE THAN 10'
WHEN COUNT(RENTAL.ID) BETWEEN 5 AND 10 THEN 'BETWEEN 5 AND 10'
ELSE 'FEWER THAN 5'
END AS RENTAL_COUNT_CATEGORY
FROM CUSTOMER LEFT JOIN rental
ON CUSTOMER.ID = RENTAL.CUSTOMER_ID
GROUP BY CUSTOMER.ID) AS CUSTOMER_SEGMENTS
GROUP BY RENTAL_COUNT_CATEGORY;
```

Result Grid | Filter Rows:

RENTAL_COUNT_CATEGORY	CUSTOMER_COUNT
MORE THAN 10	1
FEWER THAN 5	8
BETWEEN 5 AND 10	1

AnkitAnalyst1407

THANK YOU

+91 8800601749

Mehra.ankit1407@gmail.com

https://www.linkedin.com/in/ankit1407mehra/

Telangana, Hyderabad