





BIKE RENTAL SHOP DATA ANALYSIS

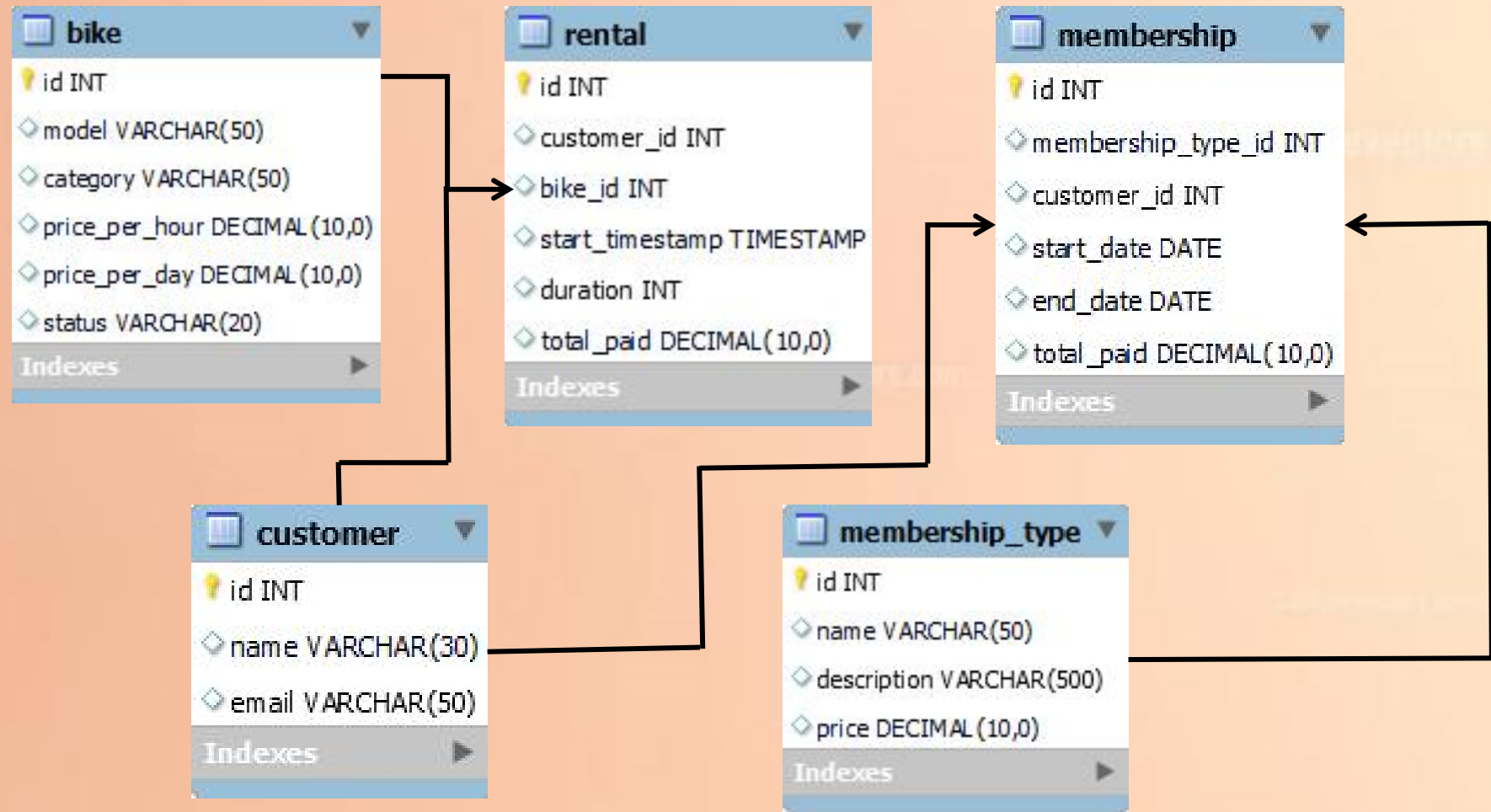
- SANTHOSH KUMAR R

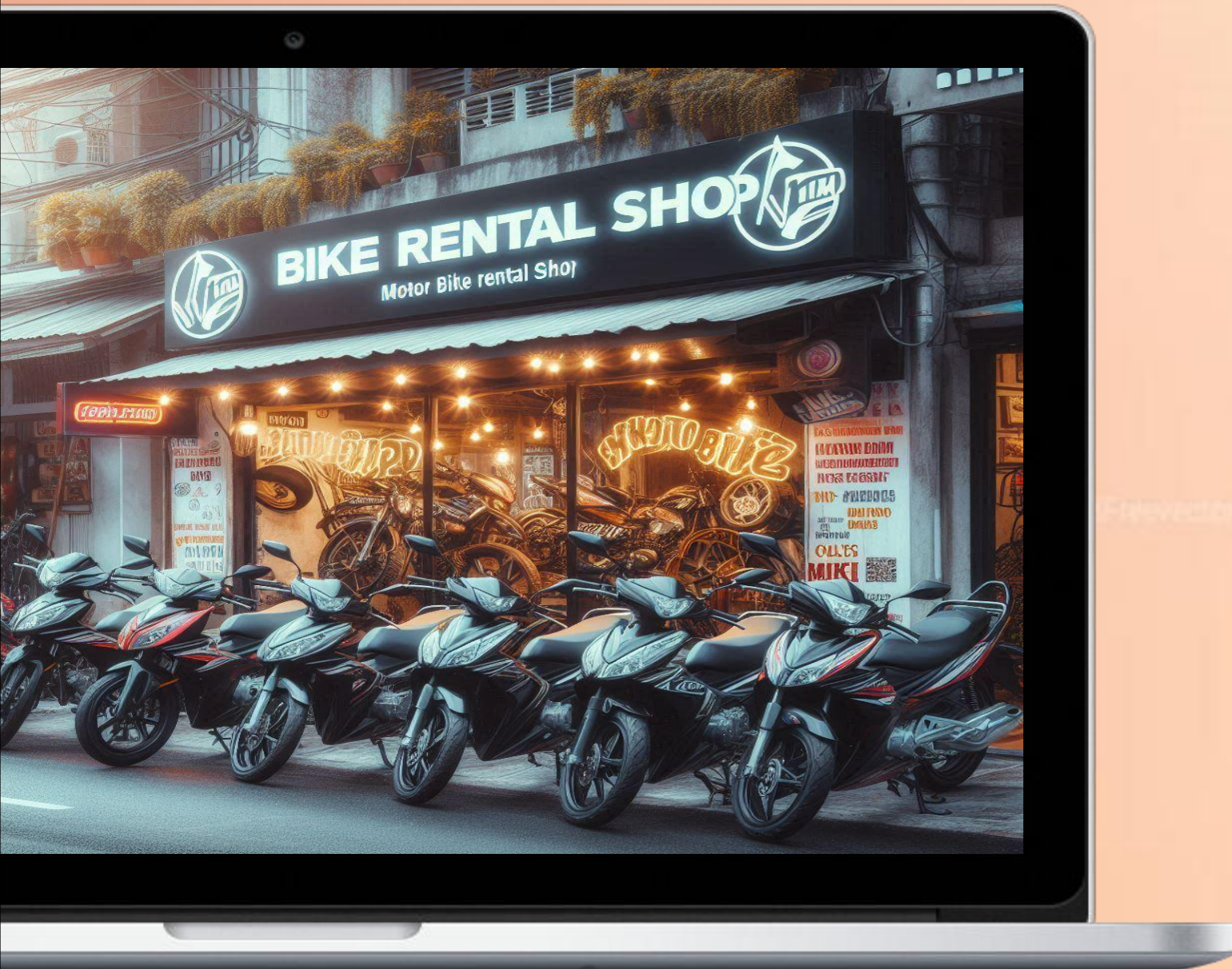


OBJECTIVE

-  This project is to conduct a comprehensive data exploration of a bike rental shop's operations using SQL analysis. This involves analyzing bike inventory by category, examining customer membership patterns, applying discounts for seasonal offers, monitoring bike rental availability, and calculating revenues.
-  Additionally, the project includes segmenting customers based on their rental frequency and revenue generation. Through this analysis, the project aims to provide actionable insights into customer behavior and support effective decision-making for business growth.

DATA MODELING





LET'S BEGIN ANALYSIS USING



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

Display the category name and the number of bikes the shop owns in each category . Show only the categories where the number of bikes is greater than 2.

QUERY :

```
select category
from bike
group by 1
Having count(model) > 2;
```

OUTPUT :

category
mountain bike
road bike

2. Santy 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.

QUERY :

```
select name
,count(mem.id) membership_count
from customer cust
left join membership mem
on cust.id = mem.customer_id
group by 1
order by 2 DESC;
```

OUTPUT :

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

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

For each bike, display its ID, category, old price per hour , discounted price per hour , old price per day , and discounted price per day .

Discounts:

- 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.**

QUERY :

```
with discount as (  
  select model,  
  id,  
  category,  
  price_per_hour as old_price_per_hour,  
  case when category = 'electric' then price_per_hour * 10/100  
    when category = 'mountain bike' then price_per_hour * 20/100  
    else price_per_hour * 50/100 end as hourly_discount,  
  case when category = 'electric' then price_per_day * 20/100  
    when category = 'mountain bike' then price_per_day * 50/100  
    else price_per_day * 50/100 end as day_discount,  
  price_per_day as old_price_per_day  
  from bike  
)  
select model,  
id,  
category,  
old_price_per_hour,  
round(old_price_per_hour - hourly_discount,2) as new_price_per_hour,  
old_price_per_day,  
round(old_price_per_day - day_discount,2) as new_price_per_day  
from discount;
```

OUTPUT :

model	id	category	old_price_per_hour	new_price_per_hour	old_price_per_day	new_price_per_day
Mountain Bike 1	1	mountain bike	10	8.00	50	25.00
Road Bike 1	2	road bike	12	6.00	60	30.00
Hybrid Bike 1	3	hybrid	8	4.00	40	20.00
Electric Bike 1	4	electric	15	13.50	75	60.00
Mountain Bike 2	5	mountain bike	10	8.00	50	25.00
Road Bike 2	6	road bike	12	6.00	60	30.00
Hybrid Bike 2	7	hybrid	8	4.00	40	20.00
Electric Bike 2	8	electric	15	13.50	75	60.00
Mountain Bike 3	9	mountain bike	10	8.00	50	25.00
Road Bike 3	10	road bike	12	6.00	60	30.00



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

Display the number of available bikes and the number of rented bikes by bike category.

QUERY :

```
select category
, count(case
      when status = 'available'
      then 1 end) as available_bikes_count
, count(case
      when status = 'rented'
      then 1 end) as rented_bikes_count
from bike
group by 1;
```

OUTPUT :

category	available_bikes_count	rented_bikes_count
mountain bike	1	1
road bike	3	0
hybrid	0	1
electric	2	0



5. Santy is preparing a sales report. he 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.

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

QUERY :

```
select year(start_timestamp) as year
,month(start_timestamp) as month
,sum(total_paid) as total_revenue
from rental
group by 1,2 with rollup
order by (year is null),1,2;
```



OUTPUT :

year	month	total_revenue
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
NULL	NULL	1720

ORDERING THE RESULT TABLE :



This part uses a boolean expression (year IS NULL) to check if year is NULL.

When year IS NULL is TRUE , it is treated as 1 in sorting.

When year IS NULL is FALSE , it is treated as 0.

Since 0 comes before 1 in ascending order,

this expression places all rows where year is not NULL (regular rows)

before the row where year is NULL (the grand total row).

6.Santy 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 , and the total revenue for every combination of year, month, and membership type.

Sort the results by year, month, and name of membership

QUERY :

```
select year(m.start_date) as year
,month(m.start_date) as month
,mt.name
,sum(m.total_paid) as total_revenue
from membership m
join membership_type mt
on m.membership_type_id = mt.id
group by 1,2,3
order by 1,2,3;
```

OUTPUT :

year	month	name	total_revenue
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

7. Santy 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.

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

QUERY :

```
select mt.name as membership_type_name
,month(m.start_date) as month
,sum(m.total_paid) as total_revenue
from membership as m
join membership_type as mt
on m.membership_type_id = mt.id
where year(m.start_date) = '2023'
group by 1,2 with rollup
order by 1, 2;
```

OUTPUT :

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
Premium Monthly	11	200

8. Santy wants to segment customers based on the number of rentals and see the count of customers in each segment.

Categorize customers based on their rental history as follows:

CATEGORY :

- **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 and customer_count .

QUERY :

```
with customers_based_on_number_of_rentals as
(
    select customer_id ,count(bike_id) as total_count_bike
    from rental as r , bike as b
    where r.bike_id = b.id
    group by 1
)
select case when total_count_bike > 10 then 'more than 10'
           when total_count_bike between 5 and 10 then 'between 5 and 10 '
           else 'fewer than 5' end as rental_count_category
,count(customer_id) as customer_count
from customers_based_on_number_of_rentals
group by 1
order by 2;
```

OUTPUT :

rental_count_category	customer_count
more than 10	1
between 5 and 10	1
fewer than 5	8





THANK YOU