-- Monday Coffee -- Data Analysis SQL(Postgres)

SELECT \* FROM city;

SELECT \* FROM products;

SELECT \* FROM customers;

SELECT \* FROM sales;

-- Reports & Data Analysis

-- Q.1 Coffee Consumers Count

-- How many people in each city are estimated to consume coffee, given that 25% of the population does?

SELECT

city\_name,

ROUND(

(population \* 0.25)/1000000,

2) as coffee\_consumers\_in\_millions,

city\_rank

FROM city

ORDER BY coffee\_consumers\_in\_millions DESC

-- -- Q.2

-- Total Revenue from Coffee Sales

-- What is the total revenue generated from coffee sales across all cities in the last quarter of 2023?

SELECT

ci.city\_name,

SUM(s.total) as total\_revenue

FROM sales as s

JOIN customers as c

ON s.customer\_id = c.customer\_id

JOIN city as ci

ON ci.city\_id = c.city\_id

WHERE

EXTRACT(YEAR FROM s.sale\_date) = 2023

AND

EXTRACT(quarter FROM s.sale\_date) = 4

GROUP BY ci.city\_name

ORDER BY SUM(s.total) DESC

-- Q.3

-- Sales Count for Each Product

-- How many units of each coffee product have been sold?

SELECT

p.product\_name,

COUNT(s.sale\_id) as total\_orders

FROM products as p

LEFT JOIN

sales as s

ON s.product\_id = p.product\_id

GROUP BY p.product\_name

ORDER BY total\_orders DESC

-- Q.4

-- Average Sales Amount per City

-- What is the average sales amount per customer in each city?

SELECT

ci.city\_name,

SUM(s.total) as total\_revenue,

COUNT(DISTINCT s.customer\_id) as total\_cx,

ROUND(

SUM(s.total)::numeric/

COUNT(DISTINCT s.customer\_id)::numeric

,2) as avg\_sale\_pr\_cx

FROM sales as s

JOIN customers as c

ON s.customer\_id = c.customer\_id

JOIN city as ci

ON ci.city\_id = c.city\_id

GROUP BY 1

ORDER BY 2 DESC

-- -- Q.5

-- City Population and Coffee Consumers (25%)

-- Provide a list of cities along with their populations and estimated coffee consumers.

-- return city\_name, total current cx, estimated coffee consumers (25%)

WITH city\_data AS (

SELECT

ci.city\_name,

ROUND((ci.population \* 0.25) / 1000000, 2) AS coffee\_consumers,

COUNT(DISTINCT c.customer\_id) AS unique\_cx

FROM city ci

LEFT JOIN customers c ON ci.city\_id = c.city\_id

LEFT JOIN sales s ON c.customer\_id = s.customer\_id

GROUP BY ci.city\_name, ci.population

)

SELECT

city\_name,

coffee\_consumers AS coffee\_consumer\_in\_millions,

unique\_cx

FROM city\_data;

-- -- Q6

-- Top Selling Products by City

-- What are the top 3 selling products in each city based on sales volume?

SELECT \*

FROM

(

SELECT

ci.city\_name,

p.product\_name,

COUNT(s.sale\_id) as total\_orders,

DENSE\_RANK() OVER(PARTITION BY ci.city\_name ORDER BY COUNT(s.sale\_id) DESC) as rank

FROM sales as s

JOIN products as p

ON s.product\_id = p.product\_id

JOIN customers as c

ON c.customer\_id = s.customer\_id

JOIN city as ci

ON ci.city\_id = c.city\_id

GROUP BY 1, 2

-- ORDER BY 1, 3 DESC

) as t1

WHERE rank <= 3

-- Q.7

-- Customer Segmentation by City

-- How many unique customers are there in each city who have purchased coffee products?

select

c.city\_name,

count(distinct s.customer\_id) distinct\_customer

from sales s

left join customers cu

on cu.customer\_id = s.customer\_id

join city c

on c.city\_id = cu.city\_id

where s.product\_id is not null

group by c.city\_name

order by distinct\_customer desc

-- -- Q.8

-- Average Sale vs Rent

-- Find each city and their average sale per customer and avg rent per customer

select

c.city\_name,

round(sum(s.total)::numeric / count(distinct s.customer\_id)::numeric,2)

as avg\_sales\_per\_customer,

round(c.estimated\_rent ::numeric/count(distinct s.customer\_id)::numeric,2)

as avg\_rent\_per\_customer

from sales s

join customers cu

on cu.customer\_id = s.customer\_id

join city c

on c.city\_id = cu.city\_id

group by c.city\_name,c.estimated\_rent

ORDER BY avg\_sales\_per\_customer DESC;

-- Q.9

-- Monthly Sales Growth

-- Sales growth rate: Calculate the percentage growth (or decline) in sales over different time periods (monthly)

-- by each city

with cte as (

select

c.city\_name,extract(month from sale\_date) as month,

extract(year from sale\_date) as year,sum(s.total) total\_sales

from sales s

join customers cu

on cu.customer\_id = s.customer\_id

join city c

on cu.city\_id = c.city\_id

group by 1,2,3

order by 1,3,2)

, growth\_ratio as (

select

city\_name,total\_sales as cur\_month\_sales,

month,year,

lag(total\_sales,1) over(partition by city\_name order by year,month) as last\_month\_sales

from cte)

select city\_name,

month,year,

cur\_month\_sales,last\_month\_sales,

round((cur\_month\_sales - last\_month\_sales)::numeric / last\_month\_sales ::numeric ,2)\*100 as growth\_rate

from growth\_ratio

-- Q.10

-- Market Potential Analysis

-- Identify top 3 city based on highest sales, return city name, total sale, total rent, total customers, estimated coffee consumer

WITH city\_data AS (

SELECT

ci.city\_name,

SUM(s.total) AS total\_revenue,

COUNT(DISTINCT s.customer\_id) AS total\_cx,

ROUND(SUM(s.total)::numeric / COUNT(DISTINCT s.customer\_id)::numeric, 2) AS avg\_sale\_pr\_cx,

ci.estimated\_rent,

ROUND((ci.population \* 0.25) / 1000000, 3) AS estimated\_coffee\_consumer\_in\_millions

FROM city ci

LEFT JOIN customers c ON ci.city\_id = c.city\_id

LEFT JOIN sales s ON c.customer\_id = s.customer\_id

GROUP BY ci.city\_name, ci.estimated\_rent, ci.population

)

SELECT

city\_name,

total\_revenue,

estimated\_rent AS total\_rent,

total\_cx,

estimated\_coffee\_consumer\_in\_millions,

avg\_sale\_pr\_cx,

ROUND(estimated\_rent::numeric / NULLIF(total\_cx, 0), 2) AS avg\_rent\_per\_cx

FROM city\_data

ORDER BY total\_revenue DESC;

-- Recommendation

City 1: Pune

1.Average rent per customer is very low.

2.Highest total revenue.

3.Average sales per customer is also high.

City 2: Delhi

1.Highest estimated coffee consumers at 7.7 million.

2.Highest total number of customers, which is 68.

3.Average rent per customer is 330 (still under 500).

City 3: Jaipur

1.Highest number of customers, which is 69.

2.Average rent per customer is very low at 156.

3.Average sales per customer is better at 11.6k.