**FoodPulse\_Kerala\_Analytics**

**SQL Queries**

CREATE TABLE customers (

customer\_id SERIAL PRIMARY KEY,

name VARCHAR(100),

gender VARCHAR(10),

age INT,

city VARCHAR(50),

email VARCHAR(100),

phone VARCHAR(15),

join\_date DATE

);

CREATE TABLE restaurants (

restaurant\_id SERIAL PRIMARY KEY,

restaurant\_name VARCHAR(100),

district VARCHAR(50),

type VARCHAR(50),

contact VARCHAR(20),

address TEXT,

opening\_year INT,

avg\_rating NUMERIC(2,1)

);

CREATE TABLE menu\_items (

menu\_item\_id SERIAL PRIMARY KEY,

restaurant\_id INT REFERENCES restaurants(restaurant\_id),

dish\_name VARCHAR(100),

category VARCHAR(50),

price NUMERIC(6,2),

available BOOLEAN

);

CREATE TABLE orders (

order\_id SERIAL PRIMARY KEY,

customer\_id INT REFERENCES customers(customer\_id),

restaurant\_id INT REFERENCES restaurants(restaurant\_id),

order\_datetime TIMESTAMP,

order\_status VARCHAR(50),

total\_amount DECIMAL(10, 2)

);

CREATE TABLE order\_items (

order\_item\_id SERIAL PRIMARY KEY,

order\_id INT REFERENCES orders(order\_id),

menu\_item\_id INT REFERENCES menu\_items(menu\_item\_id),

quantity INT,

item\_price DECIMAL(10, 2),

total\_price DECIMAL(10, 2)

);

CREATE TABLE payments (

payment\_id SERIAL PRIMARY KEY,

order\_id INT REFERENCES orders(order\_id),

payment\_method VARCHAR(50),

payment\_status VARCHAR(50),

amount\_paid DECIMAL(10, 2),

payment\_time TIMESTAMP

);

CREATE TABLE delivery (

delivery\_id SERIAL PRIMARY KEY,

order\_id INT REFERENCES orders(order\_id),

restaurant\_id INT REFERENCES restaurants(restaurant\_id),

customer\_id INT REFERENCES customers(customer\_id),

delivery\_partner VARCHAR(100),

estimated\_delivery\_time INT,

actual\_delivery\_time INT,

status VARCHAR(50)

);

CREATE TABLE reviews (

review\_id SERIAL PRIMARY KEY,

order\_id INT REFERENCES orders(order\_id),

customer\_id INT REFERENCES customers(customer\_id),

restaurant\_id INT REFERENCES restaurants(restaurant\_id),

rating INT CHECK (rating BETWEEN 1 AND 5),

review\_text TEXT,

review\_time TIMESTAMP

);

COPY customers(customer\_id, name, gender, age, city, email, phone, join\_date)

FROM 'E:/Python/FoodPulse\_Kerala\_Analytics/customers.csv'

DELIMITER ','

CSV HEADER;

COPY restaurants(restaurant\_id, restaurant\_name, district, type, contact, address, opening\_year, avg\_rating)

FROM 'E:/Python/FoodPulse\_Kerala\_Analytics/restaurants.csv'

DELIMITER ','

CSV HEADER;

COPY menu\_items(menu\_item\_id, restaurant\_id, dish\_name, category, price, available)

FROM 'E:/Python/FoodPulse\_Kerala\_Analytics/menu\_items.csv'

DELIMITER ','

CSV HEADER;

COPY orders(order\_id, customer\_id, restaurant\_id, order\_datetime, order\_status, total\_amount)

FROM 'E:\Python\FoodPulse\_Kerala\_Analytics\orders.csv'

DELIMITER ','

CSV HEADER;

COPY order\_items(order\_item\_id, order\_id, menu\_item\_id, quantity, item\_price, total\_price)

FROM 'E:\Python\FoodPulse\_Kerala\_Analytics\order\_items.csv'

DELIMITER ','

CSV HEADER;

COPY payments(payment\_id, order\_id, payment\_method, payment\_status,amount\_paid, payment\_time)

FROM 'E:\Python\FoodPulse\_Kerala\_Analytics\payments.csv'

DELIMITER ','

CSV HEADER;

DROP TABLE IF EXISTS delivery;

CREATE TABLE delivery (

delivery\_id SERIAL PRIMARY KEY,

order\_id INT,

restaurant\_id INT,

customer\_id INT,

delivery\_partner VARCHAR(50),

estimated\_delivery\_time INT,

actual\_delivery\_time REAL,

status VARCHAR(20)

);

COPY delivery(delivery\_id, order\_id, restaurant\_id, customer\_id, delivery\_partner, estimated\_delivery\_time, actual\_delivery\_time, status)

FROM 'E:\Python\FoodPulse\_Kerala\_Analytics\delivery.csv'

DELIMITER ','

CSV HEADER;

COPY reviews(review\_id, order\_id, customer\_id, restaurant\_id, rating, review\_text, review\_time)

FROM 'E:\Python\FoodPulse\_Kerala\_Analytics\reviews.csv'

DELIMITER ','

CSV HEADER;

**QUESTIONS**

**General**

1. **Total number of customers**

SELECT COUNT(\*) AS total\_customers FROM customers;

1. **Total number of orders**

SELECT COUNT(\*) AS total\_orders FROM orders;

1. **Average order value**

SELECT AVG(total\_amount) AS avg\_order\_value FROM orders;

1. **Total revenue generated**

SELECT SUM(amount\_paid) AS total\_revenue FROM payments

WHERE payment\_status = 'Paid';

1. **Most popular dish**

SELECT mi.dish\_name, COUNT(oi.menu\_item\_id) AS order\_count

FROM order\_items oi

JOIN menu\_items mi ON oi.menu\_item\_id = mi.menu\_item\_id

GROUP BY mi.dish\_name

ORDER BY order\_count DESC

LIMIT 5;

1. **Top 5 restaurants by number of orders**

SELECT r.restaurant\_name, COUNT(o.order\_id) AS total\_orders

FROM orders o

JOIN restaurants r ON o.restaurant\_id = r.restaurant\_id

GROUP BY r.restaurant\_name

ORDER BY total\_orders DESC

LIMIT 5;

1. **Revenue by city**

SELECT c.city, SUM(p.amount\_paid) AS city\_revenue

FROM customers c

JOIN orders o ON c.customer\_id = o.customer\_id

JOIN payments p ON o.order\_id = p.order\_id

WHERE p.payment\_status = 'Paid'

GROUP BY c.city

ORDER BY city\_revenue DESC;

1. **Most active customers**

SELECT c.name, COUNT(o.order\_id) AS total\_orders

FROM customers c

JOIN orders o ON c.customer\_id = o.customer\_id

GROUP BY c.name

ORDER BY total\_orders DESC

LIMIT 5;

1. **Monthly order trend**

SELECT DATE\_TRUNC('month', order\_datetime) AS month, COUNT(\*) AS total\_orders

FROM orders

GROUP BY month

ORDER BY month;

1. **Peak order hours**

SELECT EXTRACT(HOUR FROM order\_datetime) AS hour, COUNT(\*) AS order\_count

FROM orders

GROUP BY hour

ORDER BY order\_count DESC;

1. **Cancellation rate**

SELECT

ROUND(100.0 \* COUNT(\*) FILTER (WHERE order\_status = 'Cancelled') / COUNT(\*), 2) AS cancellation\_rate\_percent

FROM orders;

1. **Delayed deliveries percentage**

SELECT

ROUND(100.0 \* COUNT(\*) FILTER (WHERE status = 'Delayed') / COUNT(\*), 2) AS delayed\_delivery\_percent

FROM delivery;

1. **Payment method preference**

SELECT payment\_method, COUNT(\*) AS method\_count

FROM payments

GROUP BY payment\_method

ORDER BY method\_count DESC;

1. **Restaurant with highest rating**

SELECT restaurant\_name, avg\_rating

FROM restaurants

ORDER BY avg\_rating DESC

LIMIT 5;

1. **Total number of menu items per category**

SELECT category, COUNT(\*) AS items\_count

FROM menu\_items

GROUP BY category;

1. **Average delivery time**

SELECT ROUND(AVG(actual\_delivery\_time)) AS avg\_actual\_time

FROM delivery

WHERE status = 'Delivered';

1. **Customer acquisition trend (monthly join)**

SELECT DATE\_TRUNC('month', join\_date) AS month, COUNT(\*) AS new\_customers

FROM customers

GROUP BY month

ORDER BY month;

1. **Best delivery partner (least delays)**

SELECT delivery\_partner,

COUNT(\*) FILTER (WHERE status = 'Delivered') AS successful\_deliveries,

COUNT(\*) FILTER (WHERE status = 'Delayed') AS delays

FROM delivery

GROUP BY delivery\_partner

ORDER BY delays ASC

LIMIT 1;

1. **Most reviewed restaurant**

SELECT r.restaurant\_name, COUNT(re.review\_id) AS review\_count

FROM reviews re

JOIN restaurants r ON re.restaurant\_id = r.restaurant\_id

GROUP BY r.restaurant\_name

ORDER BY review\_count DESC

LIMIT 1;

1. **Average rating by district**

SELECT district, ROUND(AVG(rating), 2) AS avg\_rating

FROM reviews

JOIN restaurants ON reviews.restaurant\_id = restaurants.restaurant\_id

GROUP BY district

ORDER BY avg\_rating DESC;

1. **Most common dish category ordered**

SELECT mi.category, COUNT(\*) AS times\_ordered

FROM order\_items oi

JOIN menu\_items mi ON oi.menu\_item\_id = mi.menu\_item\_id

GROUP BY mi.category

ORDER BY times\_ordered DESC;

1. **Revenue per restaurant**

SELECT r.restaurant\_name, SUM(p.amount\_paid) AS total\_revenue

FROM restaurants r

JOIN orders o ON r.restaurant\_id = o.restaurant\_id

JOIN payments p ON o.order\_id = p.order\_id

WHERE p.payment\_status = 'Paid'

GROUP BY r.restaurant\_name

ORDER BY total\_revenue DESC;

1. **Avg spending per customer**

SELECT c.name, ROUND(SUM(p.amount\_paid), 2) AS total\_spent

FROM customers c

JOIN orders o ON c.customer\_id = o.customer\_id

JOIN payments p ON o.order\_id = p.order\_id

WHERE p.payment\_status = 'Paid'

GROUP BY c.name

ORDER BY total\_spent DESC

LIMIT 10;

1. **Customer with most cancellations**

SELECT c.name, COUNT(\*) AS cancellations

FROM orders o

JOIN customers c ON o.customer\_id = c.customer\_id

WHERE order\_status = 'Cancelled'

GROUP BY c.name

ORDER BY cancellations DESC

LIMIT 1;

1. **Time lag between order and delivery**

SELECT ROUND(AVG(actual\_delivery\_time)) - ROUND(AVG(estimated\_delivery\_time)) AS avg\_delay

FROM delivery

WHERE actual\_delivery\_time IS NOT NULL;

**🍽️ Restaurant Performance & Sales**

1. **Top 10 restaurants with highest number of orders.**

SELECT r.restaurant\_name,

COUNT(o.order\_id) AS total\_orders

FROM orders o

JOIN restaurants r ON o.restaurant\_id = r.restaurant\_id

GROUP BY r.restaurant\_name

ORDER BY total\_orders DESC

LIMIT 10;

1. **Restaurants with the highest total revenue.**

SELECT r.restaurant\_name,

SUM(o.total\_amount) AS total\_revenue

FROM orders o

JOIN restaurants r ON o.restaurant\_id = r.restaurant\_id

WHERE o.order\_status = 'Delivered'

GROUP BY r.restaurant\_name

ORDER BY total\_revenue DESC;

1. **Average order value (AOV) per restaurant.**

SELECT r.restaurant\_name, ROUND(AVG(o.total\_amount), 2) AS avg\_order\_value

FROM orders o

JOIN restaurants r ON o.restaurant\_id = r.restaurant\_id

WHERE o.order\_status = 'Delivered'

GROUP BY r.restaurant\_name

ORDER BY avg\_order\_value DESC;

1. **Most popular dish in each district.**

SELECT sub.district, sub.dish\_name, sub.total\_quantity

FROM (

SELECT r.district, m.dish\_name,

SUM(oi.quantity) AS total\_quantity,

ROW\_NUMBER() OVER (PARTITION BY r.district ORDER BY SUM(oi.quantity) DESC) AS rank

FROM order\_items oi

JOIN menu\_items m ON oi.menu\_item\_id = m.menu\_item\_id

JOIN restaurants r ON m.restaurant\_id = r.restaurant\_id

GROUP BY r.district, m.dish\_name

) sub

WHERE sub.rank = 1;

1. **Top 5 best-rated restaurants (avg rating > 4, minimum 1 reviews).**

SELECT

r.restaurant\_name,

COUNT(rv.review\_id) AS review\_count

FROM

reviews rv

JOIN

restaurants r ON rv.restaurant\_id = r.restaurant\_id

GROUP BY

r.restaurant\_name

HAVING

COUNT(rv.review\_id) >= 1

ORDER BY

review\_count DESC;

1. **Month-wise revenue trend for top-performing restaurants.**

WITH top\_restaurants AS (

SELECT restaurant\_id

FROM orders

WHERE order\_status = 'Delivered'

GROUP BY restaurant\_id

ORDER BY SUM(total\_amount) DESC

LIMIT 5

)

SELECT

r.restaurant\_name,

TO\_CHAR(o.order\_datetime, 'YYYY-MM') AS month,

SUM(o.total\_amount) AS monthly\_revenue

FROM orders o

JOIN restaurants r ON o.restaurant\_id = r.restaurant\_id

WHERE o.restaurant\_id IN (SELECT restaurant\_id FROM top\_restaurants) AND o.order\_status ='Delivered'

GROUP BY r.restaurant\_name, TO\_CHAR(o.order\_datetime, 'YYYY-MM')

ORDER BY r.restaurant\_name, month;

1. **Which restaurant has the highest number of cancelled orders?**

SELECT r.restaurant\_name, COUNT(\*) AS cancelled\_orders

FROM orders o

JOIN restaurants r ON o.restaurant\_id = r.restaurant\_id

WHERE o.order\_status = 'Cancelled'

GROUP BY r.restaurant\_name

ORDER BY cancelled\_orders DESC

LIMIT 1;

**👥 Customer Insights**

1. **Top 10 loyal customers by number of orders.**

SELECT c.name, COUNT(o.order\_id) AS total\_orders

FROM orders o

JOIN customers c ON o.customer\_id = c.customer\_id

GROUP BY c.name

ORDER BY total\_orders DESC

LIMIT 10;

1. **Customers who have spent the most in total.**

SELECT

c.name,

SUM(p.amount\_paid) AS total\_spent

FROM payments p

JOIN orders o ON p.order\_id = o.order\_id

JOIN customers c ON o.customer\_id = c.customer\_id

WHERE p.payment\_status = 'Success'

GROUP BY c.name

ORDER BY total\_spent DESC

LIMIT 10;

1. **Average order frequency per customer (orders per month).**

SELECT

c.name,

ROUND(

(COUNT(o.order\_id) \* 1.0 / DATE\_PART('month', AGE(MAX(o.order\_datetime), MIN(o.order\_datetime))))::numeric,

2

) AS avg\_orders\_per\_month

FROM orders o

JOIN customers c ON o.customer\_id = c.customer\_id

GROUP BY c.name

ORDER BY avg\_orders\_per\_month DESC;

1. **Gender-wise distribution of orders.**

SELECT

c.gender,

COUNT(o.order\_id) AS total\_orders

FROM orders o

JOIN customers c ON o.customer\_id = c.customer\_id

GROUP BY c.gender;

1. **City-wise customer count and average order value.**

SELECT

c.city,

COUNT(DISTINCT c.customer\_id) AS customer\_count,

ROUND(AVG(o.total\_amount), 2) AS avg\_order\_value

FROM customers c

JOIN orders o ON c.customer\_id = o.customer\_id

GROUP BY c.city

ORDER BY customer\_count DESC;

1. **Customers who gave 5-star reviews and their most-ordered restaurant.**

WITH FiveStarReviews AS (

SELECT DISTINCT customer\_id

FROM reviews

WHERE rating = 5

),

CustomerRestaurantOrders AS (

SELECT

o.customer\_id,

o.restaurant\_id,

COUNT(o.order\_id) AS order\_count

FROM orders o

GROUP BY o.customer\_id, o.restaurant\_id

),

MostOrdered AS (

SELECT

customer\_id,

restaurant\_id,

order\_count,

RANK() OVER (PARTITION BY customer\_id ORDER BY order\_count DESC) AS rank

FROM

CustomerRestaurantOrders

)

SELECT c.name, r.restaurant\_name, mo.order\_count

FROM MostOrdered mo

JOIN customers c ON mo.customer\_id = c.customer\_id

JOIN restaurants r ON mo.restaurant\_id = r.restaurant\_id

WHERE

mo.rank = 1 AND mo.customer\_id IN (SELECT customer\_id FROM FiveStarReviews)

ORDER BY mo.order\_count DESC;

**💸 Payments & Revenue**

1. **Total revenue split by payment method.**

SELECT

payment\_method,

SUM(amount\_paid) AS total\_revenue

FROM payments

WHERE payment\_status = 'Paid'

GROUP BY payment\_method

ORDER BY total\_revenue DESC;

1. **Payment failure rate per method.**

SELECT

payment\_method,

COUNT(\*) AS total\_attempts,

COUNT(\*) FILTER (WHERE payment\_status = 'Failed') AS failed\_payments,

ROUND(

(COUNT(\*) FILTER (WHERE payment\_status = 'Failed') \* 100.0 / COUNT(\*))::numeric,

2

) AS failure\_rate\_percent

FROM payments

GROUP BY payment\_method

ORDER BY failure\_rate\_percent DESC;

1. **Orders where payment amount ≠ order amount (data inconsistency check).**

SELECT

o.order\_id,

o.total\_amount AS order\_total,

p.amount\_paid AS payment\_amount

FROM orders o

JOIN payments p ON o.order\_id = p.order\_id

WHERE o.total\_amount <> p.amount\_paid;

1. **Monthly revenue trends and YoY growth.**

SELECT

TO\_CHAR(payment\_time, 'YYYY-MM') AS month,

SUM(amount\_paid) AS monthly\_revenue,

LAG(SUM(amount\_paid)) OVER (ORDER BY TO\_CHAR(payment\_time, 'YYYY-MM')) AS prev\_month\_revenue,

ROUND(

(SUM(amount\_paid) - LAG(SUM(amount\_paid)) OVER (ORDER BY TO\_CHAR(payment\_time, 'YYYY-MM')))

\* 100.0 / NULLIF(LAG(SUM(amount\_paid)) OVER (ORDER BY TO\_CHAR(payment\_time, 'YYYY-MM')), 0),

2

) AS growth\_percentage

FROM payments

WHERE payment\_status = 'Paid'

GROUP BY

TO\_CHAR(payment\_time, 'YYYY-MM')

ORDER BY month;

1. **Number of pending vs completed payments by method.**

SELECT

payment\_method,

COUNT(\*) FILTER (WHERE payment\_status = 'Paid') AS completed,

COUNT(\*) FILTER (WHERE payment\_status = 'Pending') AS pending,

COUNT(\*) FILTER (WHERE payment\_status = 'Failed') AS failed

FROM payments

GROUP BY payment\_method;

**📦 Order & Delivery Analytics**

1. **Average delivery time by delivery partner.**

SELECT

delivery\_partner,

ROUND(AVG((actual\_delivery\_time - estimated\_delivery\_time)::numeric), 2) AS avg\_delay\_minutes,

ROUND(AVG(actual\_delivery\_time::numeric), 2) AS avg\_delivery\_time

FROM delivery

WHERE status = 'Delivered'

GROUP BY delivery\_partner

ORDER BY avg\_delivery\_time;

1. **Orders delayed beyond 10 mins by restaurant.**

SELECT

r.restaurant\_name,

COUNT(d.delivery\_id) AS delayed\_orders

FROM delivery d

JOIN restaurants r ON d.restaurant\_id = r.restaurant\_id

WHERE (actual\_delivery\_time - estimated\_delivery\_time) > 10 AND d.status = 'Delivered'

GROUP BY r.restaurant\_name

ORDER BY delayed\_orders DESC;

1. **Cancellation rate by delivery partner.**

SELECT

delivery\_partner,

COUNT(\*) AS total\_deliveries,

COUNT(\*) FILTER (WHERE status = 'Cancelled') AS cancelled\_deliveries,

ROUND(

(COUNT(\*) FILTER (WHERE status = 'Cancelled') \* 100.0 / COUNT(\*))::numeric,

2

) AS cancellation\_rate\_percent

FROM delivery

GROUP BY delivery\_partner

ORDER BY cancellation\_rate\_percent DESC;

1. **Peak order hours and days.**

SELECT

EXTRACT(DOW FROM order\_datetime) AS day\_of\_week,

EXTRACT(HOUR FROM order\_datetime) AS hour,

COUNT(\*) AS order\_count

FROM orders

GROUP BY day\_of\_week, hour

ORDER BY order\_count DESC

LIMIT 10;

1. **Average number of items per order.**

SELECT

ROUND(AVG(total\_items), 2) AS avg\_items\_per\_order

FROM (

SELECT

order\_id,

SUM(quantity) AS total\_items

FROM

order\_items

GROUP BY

order\_id

) AS order\_summary;

1. **Districts with fastest average delivery times.**

SELECT

r.district,

ROUND(AVG(d.actual\_delivery\_time::numeric), 2) AS avg\_delivery\_time

FROM delivery d

JOIN restaurants r ON d.restaurant\_id = r.restaurant\_id

WHERE d.status = 'Delivered'

GROUP BY r.district

ORDER BY avg\_delivery\_time ASC

LIMIT 10;

1. **Correlation between delivery delays and low customer ratings.**

SELECT

ROUND(AVG((d.actual\_delivery\_time - d.estimated\_delivery\_time)::numeric), 2) AS avg\_delay,

ROUND(AVG(rv.rating::numeric), 2) AS avg\_rating

FROM delivery d

JOIN reviews rv ON d.order\_id = rv.order\_id

WHERE d.status = 'Delivered';

Customer Segmentation

**--Customer Order Summary (CLV)--**

SELECT

c.customer\_id,

c.name,

COUNT(o.order\_id) AS total\_orders,

ROUND(SUM(o.total\_amount)::numeric / NULLIF(COUNT(o.order\_id), 0), 2) AS avg\_order\_value,

DATE\_PART('month', MAX(o.order\_datetime) - MIN(o.order\_datetime)) AS tenure\_months

FROM

customers c

JOIN

orders o ON c.customer\_id = o.customer\_id

WHERE

o.order\_status = 'Delivered'

GROUP BY

c.customer\_id, c.name

ORDER BY

total\_orders DESC;

-- Caluculate Customer Lifetime Value(CLV)--

SELECT

c.customer\_id,

c.name,

COUNT(o.order\_id) AS total\_orders,

ROUND(SUM(o.total\_amount)::numeric / NULLIF(COUNT(o.order\_id), 0), 2) AS avg\_order\_value,

CASE

WHEN DATE\_PART('month', MAX(o.order\_datetime) - MIN(o.order\_datetime)) = 0 THEN 1

ELSE DATE\_PART('month', MAX(o.order\_datetime) - MIN(o.order\_datetime))

END AS tenure\_months,

ROUND(

(COUNT(o.order\_id)::numeric /

NULLIF(

CASE

WHEN DATE\_PART('month', MAX(o.order\_datetime) - MIN(o.order\_datetime)) = 0 THEN 1

ELSE DATE\_PART('month', MAX(o.order\_datetime) - MIN(o.order\_datetime))

END, 0)

)::numeric, 2

) AS purchase\_frequency,

ROUND(

(

(SUM(o.total\_amount)::numeric / NULLIF(COUNT(o.order\_id), 0)) \*

(COUNT(o.order\_id)::numeric /

NULLIF(

CASE

WHEN DATE\_PART('month', MAX(o.order\_datetime) - MIN(o.order\_datetime)) = 0 THEN 1

ELSE DATE\_PART('month', MAX(o.order\_datetime) - MIN(o.order\_datetime))

END,

0)) \*

CASE

WHEN DATE\_PART('month', MAX(o.order\_datetime) - MIN(o.order\_datetime)) = 0 THEN 1

ELSE DATE\_PART('month', MAX(o.order\_datetime) - MIN(o.order\_datetime))

END

)::numeric, 2

) AS customer\_lifetime\_value

FROM

customers c

JOIN

orders o ON c.customer\_id = o.customer\_id

WHERE

o.order\_status = 'Delivered'

GROUP BY

c.customer\_id, c.name

ORDER BY

customer\_lifetime\_value DESC

LIMIT 10;

-- RFM Analysis--

WITH customer\_orders AS (

SELECT

c.customer\_id,

MAX(o.order\_datetime) AS last\_order\_date,

COUNT(o.order\_id) AS frequency,

SUM(o.total\_amount) AS monetary

FROM

customers c

JOIN

orders o ON c.customer\_id = o.customer\_id

WHERE

o.order\_status = 'Delivered'

GROUP BY

c.customer\_id

),

recency\_calc AS (

SELECT

customer\_id,

DATE\_PART('day', CURRENT\_DATE - last\_order\_date) AS recency,

frequency,

monetary

FROM

customer\_orders

)

SELECT

customer\_id,

recency,

frequency,

ROUND(monetary::numeric, 2) AS monetary

FROM

recency\_calc

ORDER BY

recency;

-- Churn Tagging --

WITH customer\_orders AS (

SELECT

c.customer\_id,

MAX(o.order\_datetime) AS last\_order\_date,

COUNT(o.order\_id) AS total\_orders,

SUM(o.total\_amount) AS total\_spent

FROM

customers c

JOIN

orders o ON c.customer\_id = o.customer\_id

WHERE

o.order\_status = 'Delivered'

GROUP BY

c.customer\_id

)

SELECT

customer\_id,

last\_order\_date,

total\_orders,

ROUND(total\_spent::numeric, 2) AS total\_spent,

DATE\_PART('day', CURRENT\_DATE - last\_order\_date) AS days\_since\_last\_order,

CASE

WHEN CURRENT\_DATE - last\_order\_date > INTERVAL '90 days' THEN 'Churned'

ELSE 'Active'

END AS churn\_status

FROM

customer\_orders

ORDER BY

last\_order\_date DESC;

-- Customer Lifetime Value (CLV) Segmentation --

WITH customer\_value AS (

SELECT

c.customer\_id,

c.name,

COUNT(o.order\_id) AS total\_orders,

ROUND(SUM(o.total\_amount)::numeric, 2) AS total\_revenue,

ROUND(SUM(o.total\_amount)::numeric / NULLIF(COUNT(o.order\_id), 0), 2) AS avg\_order\_value

FROM

customers c

JOIN

orders o ON c.customer\_id = o.customer\_id

WHERE

o.order\_status = 'Delivered'

GROUP BY

c.customer\_id, c.name

),

clv\_segmented AS (

SELECT

\*,

NTILE(3) OVER (ORDER BY total\_revenue DESC) AS revenue\_tier

FROM

customer\_value

)

SELECT

customer\_id,

name,

total\_orders,

total\_revenue,

avg\_order\_value,

CASE

WHEN revenue\_tier = 1 THEN 'High Value'

WHEN revenue\_tier = 2 THEN 'Medium Value'

ELSE 'Low Value'

END AS clv\_segment

FROM

clv\_segmented

ORDER BY

total\_revenue DESC;