50+ SQL Queries

Business Queries

- Part A — Basic EDA (1–20)

-- 1) How many total bookings do we have?

```
SELECT COUNT(*) AS total_bookings FROM uber_rides;
```

-- 2) What's the date range covered?

```
SELECT MIN(date) AS first_date, MAX(date) AS last_date
FROM uber_rides;
```

-- 3) Which vehicle types are offered?

```
SELECT DISTINCT vehicle_type FROM uber_rides ORDER BY 1;
```

-- 4) Total revenue (sum of booking value)

```
SELECT SUM(booking_value) AS total_revenue
FROM uber_rides;
```

-- 5) Average ride distance overall

```
SELECT ROUND(AVG(ride_distance),2) AS avg_distance_km
FROM uber_rides;
```

-- 6) Top 5 pickup locations by rides

```
SELECT pickup_location, COUNT(*) AS rides
FROM uber_rides
GROUP BY pickup_location
ORDER BY rides DESC
LIMIT 5;


-- 7) Top 5 drop locations by rides

SELECT drop_location, COUNT(*) AS rides
FROM uber_rides
GROUP BY drop_location
ORDER BY rides DESC
LIMIT 5;


-- 8) Distribution of payment methods

SELECT payment_method, COUNT(*) AS rides
FROM uber_rides
GROUP BY payment_method
ORDER BY rides DESC;


-- 9) Average driver rating by vehicle type

SELECT vehicle_type, ROUND(AVG(driver_ratings),2)
AS avg_driver_rating
FROM uber_rides
GROUP BY vehicle_type
ORDER BY avg_driver_rating DESC;


-- 10) Average customer rating by vehicle type
```

```
SELECT vehicle_type, ROUND(AVG(customer_rating),2)
AS avg_customer_rating
FROM uber_rides
GROUP BY vehicle_type
ORDER BY avg_customer_rating DESC;


-- 11) Daily rides trend

SELECT date, COUNT(*) AS rides
FROM uber_rides
GROUP BY date
ORDER BY date;


-- 12) Monthly rides trend

SELECT DATE_TRUNC('month', date) AS month, COUNT(*)
AS rides -- it returns full months
FROM uber_rides
GROUP BY month
ORDER BY month;
-- OR--
SELECT EXTRACT(MONTH FROM date) AS month, COUNT(*)
AS rides --it returns month number
FROM uber_rides
GROUP BY month
ORDER BY month;


-- 13) Monthly revenue trend

SELECT DATE_TRUNC('month', date) AS month,
```

```
SUM(booking_value) AS revenue
FROM uber_rides
GROUP BY month
ORDER BY month;


-- 14) Most common driver cancellation reason

SELECT driver_cancellation_reason, COUNT(*) AS total
FROM uber_rides
WHERE cancelled_rides_by_driver = 1
GROUP BY driver_cancellation_reason
ORDER BY total DESC
LIMIT 1;


-- 15) Most common customer cancellation reason

SELECT reason_for_cancelling_by_customer, COUNT(*) AS cnt
FROM uber_rides
WHERE cancelled_rides_by_customer = 1
GROUP BY reason_for_cancelling_by_customer
ORDER BY cnt DESC
LIMIT 1;


-- 16) Average VTAT & CTAT overall

SELECT ROUND(AVG(avg_vtat),2) AS avg_vtat,
ROUND(AVG(avg_ctat),2) AS avg_ctat
FROM uber_rides;


-- 17) Average VTAT & CTAT by vehicle type
```

```
SELECT vehicle_type, ROUND(AVG(avg_vtat),2) AS avg_vtat,
ROUND(AVG(avg_ctat),2) AS avg_ctat
FROM uber_rides
GROUP BY vehicle_type;


-- 18) Count of incomplete rides and top reasons

SELECT COALESCE(incomplete_rides_reason,'Unknown')
AS reason, COUNT(*) AS cnt
FROM uber_rides
WHERE incomplete_rides = 1
GROUP BY reason
ORDER BY cnt DESC;


-- 19) Highest value booking

SELECT *
FROM uber_rides
ORDER BY booking_value DESC
LIMIT 1;


-- 20) Customers with more than 2 rides

SELECT customer_id, COUNT(*) AS rides
FROM uber_rides
GROUP BY customer_id
HAVING COUNT(*) > 2
ORDER BY rides DESC;
```

- Part B — Intermediate (21-41)

```
-- 21) Revenue by payment method per month
```

```
SELECT DATE(DATE_TRUNC('month', date)) AS month,
payment_method, SUM(booking_value) AS revenue
FROM uber_rides
WHERE payment_method <> 'NA'
GROUP BY month, payment_method
ORDER BY month, revenue DESC;


-- 22) Average booking value by pickup location

SELECT pickup_location, ROUND(AVG(booking_value),2)
AS avg_value
FROM uber_rides
GROUP BY pickup_location
ORDER BY avg_value DESC;


-- 23) Vehicle type with the longest average distance

SELECT vehicle_type, ROUND(AVG(ride_distance),2)
AS avg_km
FROM uber_rides
GROUP BY vehicle_type
ORDER BY avg_km DESC
LIMIT 1;


-- 24) Peak booking hours (top 5)

SELECT time,EXTRACT(HOUR FROM time) AS hours, COUNT(*)
AS rides
FROM uber_rides
GROUP BY time,hours
```

```
ORDER BY rides DESC
LIMIT 5;


-- 25) Weekday vs weekend ride split

SELECT
  CASE
  WHEN EXTRACT(ISODOW FROM date) IN (6,7)
  THEN 'Weekend' ELSE 'Weekday' END AS day_type,
  COUNT(*) AS rides
FROM uber_rides
GROUP BY day_type;
-- ISODOW returns day of week (mon-1,tue-2....,sat-6,sun-7)

-- 26) Completion rate (assuming booking_status='Completed')

SELECT ROUND( SUM(
CASE WHEN booking_status='Completed'
THEN 1 ELSE 0 END)::numeric / COUNT(*) * 100, 2)
AS completion_rate_pct
FROM uber_rides;


-- 27) CTAT vs VTAT gap by vehicle type

SELECT vehicle_type, ROUND(AVG(avg_ctat - avg_vtat),2)
AS avg_ctat_minus_vtat
FROM uber_rides
GROUP BY vehicle_type
ORDER BY avg_ctat_minus_vtat DESC;


-- 28) Revenue per km (efficiency) by vehicle type
```

```
SELECT vehicle_type,
    ROUND(SUM(booking_value) / SUM(ride_distance), 2)
        AS revenue_per_km
FROM uber_rides
GROUP BY vehicle_type
ORDER BY revenue_per_km DESC;


-- 29) Average customer rating by payment method

SELECT payment_method, ROUND(AVG(customer_rating),2)
AS avg_cust_rating
FROM uber_rides
GROUP BY payment_method
ORDER BY avg_cust_rating DESC;


-- 30) Cities (pickup) with highest cancellation rate

SELECT pickup_location,
    ROUND( AVG(
        CASE WHEN cancelled_rides_by_customer= 1 OR
        cancelled_rides_by_driver= 1 THEN 1 ELSE 0 END )*100, 2)
        AS cancellation_rate_pct
FROM uber_rides
GROUP BY pickup_location
ORDER BY cancellation_rate_pct DESC
LIMIT 10;


-- 31) Customers with low average rating (<3)

SELECT customer_id, ROUND(AVG(customer_rating),2)
```

```
AS avg_rating, COUNT(*) AS rides
FROM uber_rides
GROUP BY customer_id
HAVING AVG(customer_rating) > 0 AND AVG(customer_rating) <3
ORDER BY avg_rating;


-- 32) Average booking value by distance bucket

SELECT
  CASE
    WHEN ride_distance < 5 THEN 'Short <5km'
    WHEN ride_distance < 15 THEN 'Medium 5-15km'
    ELSE 'Long >=15km'
  END AS distance_bucket,
  ROUND(AVG(booking_value),2) AS avg_value
FROM uber_rides
GROUP BY distance_bucket
ORDER BY MIN(ride_distance);


-- 33) Most profitable pickup→drop pairs

SELECT pickup_location, drop_location,
ROUND(SUM(booking_value),2)
AS revenue, COUNT(*) AS rides
FROM uber_rides
GROUP BY pickup_location, drop_location
ORDER BY revenue DESC
LIMIT 10;


-- 34) Free-text cleanup example: normalize vehicle types
```

```
SELECT INITCAP(TRIM(REPLACE(vehicle_type, '_', ' ')))
AS normalized_vehicle_type, COUNT(*)
FROM uber_rides
GROUP BY 1
ORDER BY 2 DESC;



-- 35) Hourly conversion: rides completed per hour


SELECT EXTRACT(HOUR FROM time) AS hr,
    SUM(
        CASE WHEN booking_status='Completed' THEN 1 ELSE 0 END)
        AS completed,
    COUNT(*) AS total,
    ROUND( SUM(
        CASE WHEN booking_status='Completed'
        THEN 1 ELSE 0 END)::numeric / COUNT(*) * 100, 2)
        AS completion_pct
FROM uber_rides
GROUP BY hr
ORDER BY hr;



-- 36) Top 10 customers by lifetime value (LTV)


SELECT customer_id, ROUND(SUM(booking_value),2)
AS life_time_value, COUNT(*) AS rides
FROM uber_rides
GROUP BY customer_id
ORDER BY 2 DESC
LIMIT 10;



-- 37) Month-over-month (MoM) growth in rides
```

```
WITH m AS (
  SELECT DATE(DATE_TRUNC('month', date)) AS months,
  COUNT(*) AS rides
  FROM uber_rides
  GROUP BY months
)
SELECT months, rides,
      ROUND( (rides - LAG(rides)
            OVER (ORDER BY months))::numeric / NULLIF(LAG(rides)
            OVER (ORDER BY months),0) * 100, 2) AS mom_growth_pct
FROM m
ORDER BY months;



-- 38) Revenue per customer per month

SELECT DATE(DATE_TRUNC('month', date)) AS month,
customer_id, SUM(booking_value) AS revenue
FROM uber_rides
GROUP BY month, customer_id
ORDER BY month, revenue DESC;



-- 39) Average ratings trend (3-month moving average)

WITH m AS (
  SELECT DATE(DATE_TRUNC('month', date)) AS month,
  AVG(customer_rating) AS avg_rating
  FROM uber_rides
  GROUP BY month
)
SELECT month, avg_rating,
      ROUND(AVG(avg_rating) OVER
```

```
        (ORDER BY month ROWS BETWEEN 2 PRECEDING AND CURRENT ROW),2)
        AS ma3
FROM m
ORDER BY month;



-- 40) Identify routes with low customer ratings (<3.5)

SELECT pickup_location, drop_location, ROUND(AVG(customer_rating),2)
AS avg_rating, COUNT(*) AS rides
FROM uber_rides
GROUP BY pickup_location, drop_location
HAVING AVG(customer_rating) <3.5
ORDER BY avg_rating DESC, rides DESC;

-- 41) Most common payment by pickup location (mode)

SELECT DISTINCT ON (pickup_location)
    pickup_location, payment_method, cnt
FROM (
 SELECT pickup_location, payment_method, COUNT(*) AS cnt,
     RANK() OVER
                (PARTITION BY pickup_location ORDER BY COUNT(*) DESC)
                AS rnk
 FROM uber_rides
 GROUP BY pickup_location, payment_method
) t
WHERE rnk = 1
ORDER BY pickup_location, cnt DESC;
```

- PART C :Advanced (Window, CTE, Segments)

```
-- 1. Rank rides by booking value within each vehicle type
SELECT
  Booking_ID,
  Vehicle_Type,
```

```
    Booking_Value,
    RANK() OVER (PARTITION BY Vehicle_Type ORDER BY Booking_Value DESC)
        AS rank_in_vehicle
FROM uber_rides;

-- 2. Find top 3 customers by spending in each city
SELECT
    Customer_ID,
    Pickup_Location,
    SUM(Booking_Value) AS total_spent,
    DENSE_RANK() OVER
        (PARTITION BY Pickup_Location ORDER BY SUM(Booking_Value) DESC)
        AS rank_in_city
FROM uber_rides
GROUP BY Customer_ID, Pickup_Location;

-- 3. Assign sequential row numbers per customer
SELECT
    Customer_ID,
    Booking_ID,
    ROW_NUMBER() OVER (PARTITION BY Customer_ID
        ORDER BY Date, Time) AS ride_sequence
FROM uber_rides;

-- 4. Running total of revenue per month
SELECT
    DATE_TRUNC('month', Date) AS month,
    SUM(Booking_Value) AS monthly_revenue,
    SUM(SUM(Booking_Value)) OVER (ORDER BY DATE_TRUNC('month', Date))
        AS running_total
FROM uber_rides
GROUP BY month
ORDER BY month;
```

```sql
-- 5. Moving average ride value (last 3 rides per customer)
SELECT
   Customer_ID,
   Booking_ID,
   Booking_Value,
   ROUND(AVG(Booking_Value) OVER (
      PARTITION BY Customer_ID
      ORDER BY Date, Time
      ROWS BETWEEN 2 PRECEDING AND CURRENT ROW
   ),2) AS moving_avg_last3
FROM uber_rides
ORDER BY moving_avg_last3 DESC;



-- 6. Average customer rating vs global average
SELECT
   Customer_ID,
   ROUND(AVG(Customer_Rating),2) AS avg_rating,
   ROUND(AVG(AVG(Customer_Rating)) OVER (),2) AS global_avg_rating
FROM uber_rides
GROUP BY Customer_ID;

-- 7. Peak revenue hour per day
SELECT DISTINCT Date, pickup_hour, daily_revenue,
      RANK() OVER (PARTITION BY Date ORDER BY daily_revenue DESC)
            AS rank_by_day
FROM (
   SELECT
      Date,
      EXTRACT(HOUR FROM Time) AS pickup_hour,
      SUM(Booking_Value) AS daily_revenue
   FROM uber_rides
   GROUP BY Date, pickup_hour
) t
```

```
ORDER BY Date, rank_by_day;

-- 8. Rank drivers by average rating within vehicle type
SELECT
   Vehicle_Type,
   Driver_Ratings,
   RANK() OVER (PARTITION BY Vehicle_Type
        ORDER BY Driver_Ratings DESC) AS driver_rank
FROM uber_rides;


-- 9. Identify peak demand customers (customers with multiple bookings in a single day)
SELECT
   Customer_ID,
   Date,
   COUNT(*) AS rides_day,
   RANK() OVER (PARTITION BY Date ORDER BY COUNT(*) DESC)
        AS rank_in_day
FROM uber_rides
GROUP BY Customer_ID, Date;


-- 10. Average driver rating trend (3-ride moving average)
SELECT
   Driver_Ratings,
   ROUND(AVG(Driver_Ratings) OVER (ORDER BY Date
        ROWS BETWEEN 2 PRECEDING AND CURRENT ROW),2) AS moving_avg_rating
FROM uber_rides;


-- 11. Top 3 vehicle types per city by revenue
SELECT *
FROM (
   SELECT
```

```
    Pickup_Location,
    Vehicle_Type,
    SUM(Booking_Value) AS total_revenue,
    RANK() OVER (PARTITION BY Pickup_Location
                 ORDER BY SUM(Booking_Value) DESC) AS rank_in_city
  FROM uber_rides
  GROUP BY Pickup_Location, Vehicle_Type
) t
WHERE rank_in_city <= 3;
```

- CREATE VIRTUAL TABLE [VIEW]

```
-- Customer Lifetime Value (CLV) View
CREATE VIEW vw_customer_ltv AS
SELECT
  Customer_ID,
  COUNT(*) AS total_rides,
  SUM(Booking_Value) AS total_spent,
  ROUND(AVG(Customer_Rating),2) AS avg_rating,
  MIN(Date) AS first_ride_date,
  MAX(Date) AS last_ride_date
FROM uber_rides
GROUP BY Customer_ID;

--  What this gives you:>>

--total_rides → How many trips a customer has taken
--total_spent → How much revenue they've generated
--avg_rating → Average rating they've given
--first_ride_date & last_ride_date → Helps track customer retention & churn
```