Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

100 SQL Problems

Zomato

Flipkart

Dating App

```
SELECT *
FROM orders
WHERE
```

Gowtham

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

Problem 1: Find Zomato Customers Who Ordered More Than 5 Times but Rated Only Once

Problem Statement:

Identify customers who placed **more than 5 orders** but gave ratings for **only one order** in the last 6 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE zomato_orders (
    order_id INT,
    customer_id INT,
    customer_name VARCHAR(50),
    order_date DATE
);

CREATE TABLE zomato_ratings (
    rating_id INT,
    customer_id INT,
    rating INT,
    rating_date DATE
);

INSERT INTO zomato_orders VALUES
(1, 1001, 'Gowtham', '2025-01-01'),
(2, 1001, 'Gowtham', '2025-01-10'),
(3, 1001, 'Gowtham', '2025-01-20'),
(4, 1001, 'Gowtham', '2025-02-01'),
(5, 1001, 'Gowtham', '2025-02-15'),
(6, 1001, 'Gowtham', '2025-02-25');

INSERT INTO zomato_ratings VALUES
(1, 1001, 4, '2025-01-10');
```

Solution:

```
WITH order_counts AS (
   SELECT customer_id, COUNT(*) AS total_orders
   FROM zomato_orders
   WHERE order_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
   GROUP BY customer_id
),
rating_counts AS (
   SELECT customer_id, COUNT(*) AS total_ratings
   FROM zomato_ratings
   WHERE rating_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
   GROUP BY customer_id
)
SELECT o.customer_id
FROM order_counts o
LEFT JOIN rating_counts r ON o.customer_id = r.customer_id
WHERE o.total_orders > 5 AND COALESCE(r.total_ratings, 0) = 1;
```

Explanation:

- Count orders and ratings per customer in last 6 months.
- Use LEFT JOIN to include customers with zero ratings.
- Filters customers with many orders but only one rating.
- Useful to find users who need nudges to rate more.

Problem 2: Find Flipkart Customers Who Purchased Products in More Than 3 Categories

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

Problem Statement:

Identify customers who bought products from **more than 3 different categories** in the last year.

Create & Insert DDL (MySQL):

```
CREATE TABLE flipkart_purchases (
   purchase_id INT,
   customer_id INT,
   customer_name VARCHAR(50),
   product_category VARCHAR(50),
   purchase_date DATE
);

INSERT INTO flipkart_purchases VALUES
(1, 201, 'Gowtham', 'Electronics', '2024-09-10'),
(2, 201, 'Gowtham', 'Books', '2024-10-15'),
(3, 201, 'Gowtham', 'Fashion', '2025-01-20'),
(4, 201, 'Gowtham', 'Home & Kitchen', '2025-03-10');
```

Solution:

```
SELECT customer_id, customer_name, COUNT(DISTINCT product_category) AS
category_count
FROM flipkart_purchases
WHERE purchase_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
GROUP BY customer_id, customer_name
HAVING category_count > 3;
```

Explanation:

- Counts distinct categories purchased per customer.

- Filters customers with purchases across multiple categories.
- Useful to identify customers with broad interests.

Problem 3: Identify Dating App Users Who Matched With More Than 5 Users But Sent Messages to Less Than 3

Problem Statement:

Find users who matched with **more than 5 users** but sent messages to **less than 3** of them in the last 3 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE matches (
  match_id INT,
  user_id INT,
  matched_user_id INT,
  match_date DATE
);

CREATE TABLE messages (
  message_id INT,
  sender_id INT,
  receiver_id INT,
  message_date DATE
);

INSERT INTO matches VALUES
(1, 301, 401, '2025-01-01'),
(2, 301, 402, '2025-01-05'),
(3, 301, 403, '2025-01-10'),
(4, 301, 404, '2025-01-15'),
(5, 301, 405, '2025-01-20'),
(6, 301, 406, '2025-01-25');

INSERT INTO messages VALUES
(1, 301, 401, '2025-01-10'),
(2, 301, 402, '2025-01-15');
```

Solution:

```
WITH match_counts AS (
    SELECT user_id, COUNT(*) AS total_matches
    FROM matches
    WHERE match_date >= DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
    GROUP BY user_id
),
message_counts AS (
    SELECT sender_id, COUNT(DISTINCT receiver_id) AS messages_sent
    FROM messages
    WHERE message_date >= DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
    GROUP BY sender_id
)
SELECT m.user_id
FROM match_counts m
LEFT JOIN message_counts msg ON m.user_id = msg.sender_id
WHERE m.total_matches > 5 AND COALESCE(msg.messages_sent, 0) < 3;
```

Explanation:

- Counts matches and messages sent per user.
- Filters users with many matches but low messaging activity.
- Helps identify users not engaging despite matches.

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

Problem 4: Find Zomato Customers Who Ordered Only on Weekends in Last 3 Months

Problem Statement:

Identify customers who placed orders **only on Saturdays and Sundays** in the last 3 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE zomato_orders (
    order_id INT,
    customer_id INT,
    customer_name VARCHAR(50),
    order_date DATE
);

INSERT INTO zomato_orders VALUES
(1, 1001, 'Gowtham', '2025-04-05'), -- Saturday
(2, 1001, 'Gowtham', '2025-04-06'), -- Sunday
(3, 1001, 'Gowtham', '2025-04-12'); -- Saturday
```

Solution:

```
WITH order_days AS (
    SELECT customer_id,
           MAX(CASE WHEN DAYOFWEEK(order_date) BETWEEN 2 AND 6 THEN 1 ELSE 0
END) AS weekday_order,
           MAX(CASE WHEN DAYOFWEEK(order_date) IN (1,7) THEN 1 ELSE 0 END) AS
weekend_order
    FROM zomato_orders
    WHERE order_date >= DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
    GROUP BY customer_id
)
SELECT customer_id
FROM order_days
```

```
WHERE weekend_order = 1 AND weekday_order = 0;
```

Explanation:

- Uses `DAYOFWEEK()` to classify orders as weekday or weekend.
- Filters customers ordering only on weekends.
- Useful for targeted weekend promotions.

Problem 5: Find Flipkart Customers Who Returned More Than 20% of Orders in Last 6 Months

Problem Statement:

Identify customers who had a **return rate greater than 20%** of their total orders in the last 6 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE flipkart_orders (
   order_id INT,
   customer_id INT,
   customer_name VARCHAR(50),
   order_status VARCHAR(20),
   order_date DATE
);

INSERT INTO flipkart_orders VALUES
(1, 501, 'Gowtham', 'Delivered', '2025-01-10'),
(2, 501, 'Gowtham', 'Returned', '2025-02-10'),
(3, 501, 'Gowtham', 'Delivered', '2025-03-15'),
(4, 502, 'Anu', 'Delivered', '2025-01-20'),
```

(5, 502, 'Anu', 'Returned', '2025-02-25');

Solution:

```
WITH order_stats AS (
   SELECT customer_id, customer_name,
          COUNT(*) AS total_orders,
          SUM(CASE WHEN order_status = 'Returned' THEN 1 ELSE 0 END) AS returned_orders
   FROM flipkart_orders
   WHERE order_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
   GROUP BY customer_id, customer_name
)
SELECT customer_id, customer_name, total_orders, returned_orders,
      (returned_orders / total_orders) AS return_rate
FROM order_stats
WHERE (returned_orders / total_orders) > 0.20;
```

Explanation:

- Calculates total and returned orders per customer.
- Filters customers with return rate above 20%.
- Useful for targeting and understanding return behavior.

If you want me to continue with more such examples or adjust complexity, let me know!

Problem 6: Identify Users Who Took Rides in Multiple Cities in the Same Month

Problem Statement:

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

Find users who took rides in **more than one city within the same calendar month** in the last 6 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE user_rides (

    ride_id INT,

    user_id INT,

    user_name VARCHAR(50),

    city VARCHAR(50),

    ride_date DATE

);


INSERT INTO user_rides VALUES

(1, 601, 'Gowtham', 'Chennai', '2025-02-15'),

(2, 601, 'Gowtham', 'Bangalore', '2025-02-20'),

(3, 602, 'Anu', 'Chennai', '2025-03-10'),

(4, 602, 'Anu', 'Chennai', '2025-03-15');
```

Solution:

```
SELECT user_id, user_name, DATE_FORMAT(ride_date, '%Y-%m') AS ride_month,
COUNT(DISTINCT city) AS city_count

FROM user_rides

WHERE ride_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)

GROUP BY user_id, user_name, ride_month
```

```
HAVING city_count > 1;
```

Explanation:

- **DATE_FORMAT(ride_date, '%Y-%m'):** Extracts year and month from ride date.
- **COUNT(DISTINCT city):** Counts unique cities visited by user in that month.
- **HAVING city_count > 1:** Only users with rides in multiple cities in same month included.

This helps companies analyze multi-city user mobility, an important analytic for regional marketing or pricing.

Problem 7: Identify Customers Who Paid Cash for All Rides in the Last 6 Months

Problem Statement:

Find customers who **paid cash for every ride** in the last 6 months (no digital payments).

Create & Insert DDL (MySQL):

```
CREATE TABLE ride_payments (
   payment_id INT,
   customer_id INT,
   customer_name VARCHAR(50),
   payment_mode VARCHAR(20),
   ride_date DATE
);

INSERT INTO ride_payments VALUES
(1, 701, 'Gowtham', 'Cash', '2025-01-10'),
(2, 701, 'Gowtham', 'Cash', '2025-03-05'),
(3, 702, 'Anu', 'Credit Card', '2025-02-15'),
(4, 702, 'Anu', 'Cash', '2025-04-20');
```

Solution:

```
SELECT customer_id, customer_name
FROM ride_payments
WHERE ride_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
GROUP BY customer_id, customer_name
HAVING SUM(payment_mode != 'Cash') = 0;
```

Explanation:

- **Filtering last 6 months** payments.
- **Grouping by customer** to aggregate their payment modes.
- **HAVING SUM(payment_mode != 'Cash') = 0:** ensures the customer never used any mode other than cash.
- Result: Customers who always paid cash.

Problem 8: List Drivers with Highest Cancellation Rates in the Last Month

Problem Statement:

Identify drivers with the **highest cancellation rates** (percentage of cancelled rides to total rides) in the last month.

Create & Insert DDL (MySQL):

```
CREATE TABLE driver_ride_status (
  ride_id INT,
  driver_id INT,
  driver_name VARCHAR(50),
  ride_status VARCHAR(20),
    ride_date DATE
);

INSERT INTO driver_ride_status VALUES
(1, 801, 'Mani', 'Completed', '2025-06-01'),
(2, 801, 'Mani', 'Cancelled', '2025-06-02'),
(3, 802, 'Karthik', 'Cancelled', '2025-06-01'),
(4, 802, 'Karthik', 'Cancelled', '2025-06-05'),
(5, 802, 'Karthik', 'Completed', '2025-06-10');
```

Solution:

```
SELECT driver_id, driver_name,
       SUM(CASE WHEN ride_status = 'Cancelled' THEN 1 ELSE 0 END) * 100.0 / COUNT(*) AS
cancellation_rate
FROM driver_ride_status
WHERE ride_date >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH)
GROUP BY driver_id, driver_name
ORDER BY cancellation_rate DESC
LIMIT 10;
```

Explanation:

- Calculate cancellation rate as `(Cancelled rides / Total rides) * 100`.
- Use conditional aggregation inside `SUM(CASE WHEN ...)` for cancellations.
- Filter last month's rides.
- Order by cancellation rate descending and limit top 10 drivers.

Problem 9: Identify Users Who Spent More Than ₹10,000 on Rides Last Month

Problem Statement:

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

Find users who spent over ₹10,000 on rides in the last month.

Create & Insert DDL (MySQL):

```
CREATE TABLE user_ride_payments (
    payment_id INT,
    user_id INT,
    user_name VARCHAR(50),
    amount_paid DECIMAL(10,2),
    payment_date DATE
);

INSERT INTO user_ride_payments VALUES
(1, 901, 'Gowtham', 3500.00, '2025-06-05'),
(2, 901, 'Gowtham', 7000.00, '2025-06-20'),
(3, 902, 'Anu', 5000.00, '2025-06-15'),
(4, 902, 'Anu', 3000.00, '2025-06-18');
```

Solution:

```
SELECT user_id, user_name, SUM(amount_paid) AS total_spent
FROM user_ride_payments
WHERE payment_date >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH)
GROUP BY user_id, user_name
HAVING total_spent > 10000;
```

Explanation:

- Sum total payments per user over the last month.
- Filter users whose total spending exceeds ₹10,000.
- Useful for identifying high-value customers.

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

Problem 10: Gowtham Wants to Find Customers Who Used Only Bike Rides for 2 Months Continuously

Problem Statement:

Find customers who used **only bike rides (no car or other vehicle types)** for **2 consecutive months** in the last 6 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE ride_types (
   ride_id INT,
   customer_id INT,
   customer_name VARCHAR(50),
   ride_date DATE,
   vehicle_type VARCHAR(20) -- e.g., 'Bike', 'Car', 'Auto'
);

INSERT INTO ride_types VALUES
(1, 1001, 'Gowtham', '2025-01-15', 'Bike'),
(2, 1001, 'Gowtham', '2025-02-10', 'Bike'),
(3, 1001, 'Gowtham', '2025-03-05', 'Car'),
(4, 1002, 'Anu', '2025-01-20', 'Bike'),
(5, 1002, 'Anu', '2025-02-25', 'Bike');
```

Solution:

```
WITH monthly_usage AS (
   SELECT customer_id, customer_name, DATE_FORMAT(ride_date, '%Y-%m') AS
ride_month,
        SUM(CASE WHEN vehicle_type != 'Bike' THEN 1 ELSE 0 END) AS non_bike_count
   FROM ride_types
   WHERE ride_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
   GROUP BY customer_id, customer_name, ride_month
),
consecutive_months AS (
   SELECT customer_id, customer_name, ride_month,
        DATE_FORMAT(STR_TO_DATE(ride_month, '%Y-%m') - INTERVAL ROW_NUMBER()
OVER (PARTITION BY customer_id ORDER BY ride_month) MONTH, '%Y-%m') AS grp
   FROM monthly_usage
```

```
    WHERE non_bike_count = 0
)
SELECT customer_id, customer_name
FROM consecutive_months
GROUP BY customer_id, customer_name, grp
HAVING COUNT(*) >= 2;
```

Explanation:

- `monthly_usage` calculates per month if any non-bike rides exist.
- Filter months with only bike rides (`non_bike_count = 0`).
- `consecutive_months` uses window functions to detect consecutive months.
- Groups consecutive months by adjusting dates with row numbers.
- Final select filters customers with at least 2 consecutive months.

This combines aggregation, conditional sums, window functions, and date manipulation.

Problem 11: Find Customers Who Referred Friends Frequently but Had Low Ride Activity

Problem Statement:

Identify customers who referred **more than 5 friends** in the last year but completed **less than 10 rides**.

Create & Insert DDL (MySQL):

```
CREATE TABLE referrals (
    referral_id INT,
    customer_id INT,
    referral_date DATE
);
```

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

```
CREATE TABLE rides (
    ride_id INT,
    customer_id INT,
    ride_status VARCHAR(20),
    ride_date DATE
);

INSERT INTO referrals VALUES
(1, 1001, '2024-08-01'),
(2, 1001, '2024-09-15'),
(3, 1002, '2024-10-20');

INSERT INTO rides VALUES
(1, 1001, 'Completed', '2025-01-10'),
(2, 1001, 'Completed', '2025-02-12'),
(3, 1002, 'Completed', '2025-03-15');
```

Solution:

```
WITH referral_counts AS (
    SELECT customer_id, COUNT(*) AS referral_count
    FROM referrals
    WHERE referral_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
    GROUP BY customer_id
),
ride_counts AS (
    SELECT customer_id, COUNT(*) AS ride_count
    FROM rides
    WHERE ride_status = 'Completed' AND ride_date >= DATE_SUB(CURDATE(), INTERVAL 1
YEAR)
    GROUP BY customer_id
)
SELECT r.customer_id
FROM referral_counts r
LEFT JOIN ride_counts rc ON r.customer_id = rc.customer_id
WHERE r.referral_count > 5 AND (rc.ride_count < 10 OR rc.ride_count IS NULL);
```

Explanation:

- `referral_counts` gets referral numbers per customer.

- `ride_counts` gets completed rides per customer.
- Join both and filter for customers with >5 referrals but <10 rides.
- Handles null ride counts for customers with referrals but no rides.

Problem 12: Identify Customers with Increasing Monthly Ride Counts Over Last 6 Months

Problem Statement:

Find customers whose **number of rides increased every month** for the last 6 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE monthly_rides (
  customer_id INT,
  customer_name VARCHAR(50),
  ride_date DATE
);

INSERT INTO monthly_rides VALUES
(1, 'Gowtham', '2024-12-15'),
(1, 'Gowtham', '2025-01-20'),
(1, 'Gowtham', '2025-02-10'),
(1, 'Gowtham', '2025-03-25'),
(2, 'Anu', '2024-12-01'),
(2, 'Anu', '2025-01-05');
```

Solution:

```
WITH rides_per_month AS (
  SELECT customer_id, customer_name, DATE_FORMAT(ride_date, '%Y-%m') AS
ride_month, COUNT(*) AS rides_count
  FROM monthly_rides
  WHERE ride_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
  GROUP BY customer_id, customer_name, ride_month
),
lagged AS (
   SELECT customer_id, customer_name, ride_month, rides_count,
          LAG(rides_count) OVER (PARTITION BY customer_id ORDER BY ride_month) AS
prev_month_rides
   FROM rides_per_month
)
SELECT DISTINCT customer_id, customer_name
FROM lagged
GROUP BY customer_id, customer_name
HAVING MIN(rides_count > prev_month_rides) = 1;
```

Explanation:

- `rides_per_month`: Counts rides per customer per month.
- `lagged`: Uses `LAG()` window function to get previous month's ride count.
- `HAVING MIN(rides_count > prev_month_rides) = 1`: Checks if rides increased every month (true for all rows).

This query demonstrates advanced window functions and monthly trend analysis.

Problem 13: Find Customers Who Cancelled More Than 5 Rides But Have a High Completion Rate

Problem Statement:

Identify customers who have **cancelled more than 5 rides** but maintain a **completion rate above 80%** in the last 6 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE customer_ride_status (
   ride_id INT,
   customer_id INT,
   customer_name VARCHAR(50),
    ride_status VARCHAR(20),  -- 'Completed' or 'Cancelled'
    ride_date DATE
);

INSERT INTO customer_ride_status VALUES
(1, 1001, 'Gowtham', 'Completed', '2025-01-10'),
(2, 1001, 'Gowtham', 'Cancelled', '2025-02-15'),
(3, 1001, 'Gowtham', 'Cancelled', '2025-03-20'),
(4, 1001, 'Gowtham', 'Completed', '2025-04-05'),
(5, 1002, 'Anu', 'Cancelled', '2025-01-10'),
(6, 1002, 'Anu', 'Cancelled', '2025-02-20'),
(7, 1002, 'Anu', 'Completed', '2025-03-15');
```

Solution:

```
WITH ride_summary AS (
    SELECT customer_id, customer_name,
           COUNT(*) AS total_rides,
           SUM(CASE WHEN ride_status = 'Cancelled' THEN 1 ELSE 0 END) AS cancelled_rides,
           SUM(CASE WHEN ride_status = 'Completed' THEN 1 ELSE 0 END) AS
completed_rides
    FROM customer_ride_status
    WHERE ride_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
    GROUP BY customer_id, customer_name
)
SELECT customer_id, customer_name, cancelled_rides, completed_rides,
      (completed_rides / total_rides) * 100 AS completion_rate
FROM ride_summary
WHERE cancelled_rides > 5
  AND (completed_rides / total_rides) >= 0.80;
```

Explanation:

- We summarize rides per customer using a CTE:
    - `total_rides`: total number of rides in the last 6 months.
    - `cancelled_rides`: count of cancelled rides.

- `completed_rides`: count of completed rides.

- We calculate **completion rate** as `completed_rides / total_rides`.
- We filter customers who have:
  - More than 5 cancelled rides.
  - Completion rate at least 80%.

Why this is important:
Sometimes customers cancel a lot but still use the service frequently and complete most rides. This query helps identify such valuable customers balancing cancellations with high usage.

Problem 14: Identify Drivers With Consistently High Ratings but Low Number of Rides

Problem Statement:

Find drivers who have an **average rating above 4.8** but completed **less than 20 rides** in the last month.

Create & Insert DDL (MySQL):

```
CREATE TABLE driver_ratings (
   driver_id INT,
   driver_name VARCHAR(50),
   ride_id INT,
   rating DECIMAL(2,1),
   ride_date DATE
);

INSERT INTO driver_ratings VALUES
(501, 'Mani', 1, 4.9, '2025-06-01'),
(501, 'Mani', 2, 5.0, '2025-06-03'),
(502, 'Karthik', 3, 4.7, '2025-06-05'),
(503, 'Sathish', 4, 4.9, '2025-06-07');
```

Solution:

```
SELECT driver_id, driver_name,
     AVG(rating) AS avg_rating,
     COUNT(*) AS ride_count
FROM driver_ratings
WHERE ride_date >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH)
GROUP BY driver_id, driver_name
HAVING avg_rating > 4.8
  AND ride_count < 20;
```

Explanation:

- We calculate:
  - Average rating per driver in the last month.
  - Total rides completed by driver.
- We filter:
  - Drivers with average rating above 4.8.
  - Drivers who completed fewer than 20 rides.

Why this matters:
Identifying drivers who provide great service but have low engagement helps in planning incentives to motivate more rides or understand capacity issues.

Problem 15: Find Customers Who Took Rides in Multiple Cities and Spent Over ₹5000 Last Month

Problem Statement:

Find customers who took rides in **more than one city** and spent **more than ₹5000** in the last month.

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

Create & Insert DDL (MySQL):

```
CREATE TABLE ride_payments (
   payment_id INT,
   customer_id INT,
   customer_name VARCHAR(50),
   city VARCHAR(50),
   amount DECIMAL(10,2),
   payment_date DATE
);

INSERT INTO ride_payments VALUES
(1, 1001, 'Gowtham', 'Chennai', 3000.00, '2025-06-05'),
(2, 1001, 'Gowtham', 'Bangalore', 2500.00, '2025-06-20'),
(3, 1002, 'Anu', 'Chennai', 2000.00, '2025-06-10');
```

Solution:

```
WITH city_count AS (
   SELECT customer_id, customer_name, COUNT(DISTINCT city) AS cities_visited
   FROM ride_payments
   WHERE payment_date >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH)
   GROUP BY customer_id, customer_name
),
total_spent AS (
   SELECT customer_id, customer_name, SUM(amount) AS total_amount
   FROM ride_payments
   WHERE payment_date >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH)
   GROUP BY customer_id, customer_name
)
SELECT c.customer_id, c.customer_name, c.cities_visited, t.total_amount
FROM city_count c
JOIN total_spent t ON c.customer_id = t.customer_id
WHERE c.cities_visited > 1
  AND t.total_amount > 5000;
```

Explanation:

- `city_count` counts unique cities per customer last month.

- `total_spent` sums amount spent per customer last month.
- Join both to find customers who visited multiple cities **and** spent over ₹5000.

Why useful:
This identifies high-spending frequent travelers crossing city boundaries — valuable for targeted marketing or loyalty programs.

Problem 16: Identify Drivers Who Had a Ride Cancellation Rate Above 20% and Completed More Than 50 Rides Last Month

Problem Statement:

Find drivers whose **cancellation rate is above 20%** but also completed **more than 50 rides** in the last month.

Create & Insert DDL (MySQL):

```
CREATE TABLE driver_rides (
   ride_id INT,
   driver_id INT,
   driver_name VARCHAR(50),
   ride_status VARCHAR(20),
   ride_date DATE
);

INSERT INTO driver_rides VALUES
(1, 601, 'Mani', 'Completed', '2025-06-01'),
(2, 601, 'Mani', 'Cancelled', '2025-06-02'),
(3, 601, 'Mani', 'Completed', '2025-06-03'),
(4, 602, 'Ravi', 'Cancelled', '2025-06-01'),
(5, 602, 'Ravi', 'Cancelled', '2025-06-02'),
(6, 602, 'Ravi', 'Completed', '2025-06-05');
-- Assume Mani has 10% cancellation rate, Ravi 40%
```

Solution:

```
WITH ride_stats AS (
   SELECT driver_id, driver_name,
          COUNT(*) AS total_rides,
          SUM(CASE WHEN ride_status = 'Cancelled' THEN 1 ELSE 0 END) AS cancelled_rides,
          SUM(CASE WHEN ride_status = 'Completed' THEN 1 ELSE 0 END) AS
completed_rides
   FROM driver_rides
   WHERE ride_date >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH)
   GROUP BY driver_id, driver_name
)
SELECT driver_id, driver_name,
       cancelled_rides * 100.0 / total_rides AS cancellation_rate,
       completed_rides
FROM ride_stats
WHERE cancellation_rate > 20
  AND completed_rides > 50;
```

Explanation:

- Aggregate rides per driver last month with counts for cancelled and completed rides.
- Calculate cancellation rate as `(cancelled_rides / total_rides) * 100`.
- Filter drivers with cancellation rate > 20% but completed rides > 50.

Use case:
Identifies drivers with high cancellation but also high workload — useful for performance reviews or targeted coaching.

Problem 17: Find Customers with At Least 3 Different Payment Methods in the Last Year

Problem Statement:

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

Identify customers who have used **3 or more distinct payment methods** (e.g., Cash, Credit Card, UPI) in the past 12 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE payments (
    payment_id INT,
    customer_id INT,
    customer_name VARCHAR(50),
    payment_method VARCHAR(50),
    payment_date DATE
);

INSERT INTO payments VALUES
(1, 1001, 'Gowtham', 'Cash', '2024-08-01'),
(2, 1001, 'Gowtham', 'Credit Card', '2024-10-15'),
(3, 1001, 'Gowtham', 'UPI', '2025-03-20'),
(4, 1002, 'Anu', 'Cash', '2024-09-05'),
(5, 1002, 'Anu', 'Cash', '2025-01-25');
```

Solution:

```
SELECT customer_id, customer_name, COUNT(DISTINCT payment_method) AS
distinct_methods
FROM payments
WHERE payment_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
GROUP BY customer_id, customer_name
HAVING distinct_methods >= 3;
```

Explanation:

- The query filters payments made in the last year.
- It groups payments by customer to count unique payment methods per customer.
- The **HAVING** clause ensures only customers with 3 or more distinct methods are included.

This reveals customers who diversify payment options, useful for targeting multi-channel payment promotions.

Problem 18: Find Users with Increasing Average Ride Distance for 4 Consecutive Months

Problem Statement:

Find users whose **average ride distance increases every month** for 4 consecutive months.

Create & Insert DDL (MySQL):

```
CREATE TABLE ride_distances (
    ride_id INT,
    user_id INT,
    user_name VARCHAR(50),
    ride_date DATE,
    distance_km DECIMAL(5,2)
);

INSERT INTO ride_distances VALUES
(1, 2001, 'Gowtham', '2025-01-10', 5.5),
(2, 2001, 'Gowtham', '2025-02-15', 6.0),
(3, 2001, 'Gowtham', '2025-03-20', 7.2),
(4, 2001, 'Gowtham', '2025-04-25', 7.5),
(5, 2002, 'Anu', '2025-01-12', 3.0),
(6, 2002, 'Anu', '2025-02-15', 2.5);
```

Solution:

```
WITH monthly_avg AS (
    SELECT user_id, user_name, DATE_FORMAT(ride_date, '%Y-%m') AS ride_month,
          AVG(distance_km) AS avg_distance
    FROM ride_distances
    WHERE ride_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
    GROUP BY user_id, user_name, ride_month
),
ranked AS (
    SELECT user_id, user_name, ride_month, avg_distance,
           LAG(avg_distance) OVER (PARTITION BY user_id ORDER BY ride_month) AS
prev_avg_distance,
           LEAD(avg_distance) OVER (PARTITION BY user_id ORDER BY ride_month) AS
next_avg_distance
    FROM monthly_avg
)
SELECT user_id, user_name
FROM ranked
GROUP BY user_id, user_name
HAVING MIN(avg_distance > IFNULL(prev_avg_distance, 0)) = 1
   AND MIN(avg_distance < IFNULL(next_avg_distance, 100000)) = 1;
```

Explanation:

- `monthly_avg` calculates average ride distance per user per month.
- `ranked` uses window functions `LAG` and `LEAD` to compare current month's average with previous and next months.
- The **HAVING** clause ensures the average distances strictly increase across consecutive months.
- Null handling is done with `IFNULL` to handle edge months.

This type of time-series trend analysis is key in behavioral analytics.

Problem 19: Find Customers with Ride Frequency Drop of More Than 50% Month-over-Month

Problem Statement:

Identify customers whose **ride count dropped by more than 50% compared to the previous month** at least once in the last 6 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE monthly_rides (
    customer_id INT,
    customer_name VARCHAR(50),
    ride_date DATE
);

INSERT INTO monthly_rides VALUES
(1, 'Gowtham', '2025-01-10'),
(1, 'Gowtham', '2025-01-15'),
(1, 'Gowtham', '2025-02-20'),
(1, 'Gowtham', '2025-03-05'),
(2, 'Anu', '2025-01-12'),
(2, 'Anu', '2025-02-12'),
(2, 'Anu', '2025-02-15'),
(2, 'Anu', '2025-03-20');
```

Solution:

```
WITH monthly_counts AS (
    SELECT customer_id, customer_name, DATE_FORMAT(ride_date, '%Y-%m') AS
ride_month, COUNT(*) AS rides_count
    FROM monthly_rides
    WHERE ride_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
    GROUP BY customer_id, customer_name, ride_month
),
lagged_counts AS (
    SELECT customer_id, customer_name, ride_month, rides_count,
           LAG(rides_count) OVER (PARTITION BY customer_id ORDER BY ride_month) AS
prev_rides
    FROM monthly_counts
)
SELECT DISTINCT customer_id, customer_name
FROM lagged_counts
WHERE prev_rides IS NOT NULL
  AND rides_count < prev_rides / 2;
```

Explanation:

- `monthly_counts` aggregates rides per customer per month.

- `lagged_counts` gets previous month's rides for comparison using `LAG()`.
- The final filter selects customers with more than 50% drop compared to previous month.
- `DISTINCT` removes duplicates as drop may occur multiple times.

This detects churn risk or engagement drop-offs.

Problem 20: Identify Customers Who Took Rides in All Cities Where Service Operates

Problem Statement:

Find customers who have taken rides in **every city** where the company operates in the last year.

Create & Insert DDL (MySQL):

```
CREATE TABLE rides (
   ride_id INT,
   customer_id INT,
   customer_name VARCHAR(50),
   city VARCHAR(50),
   ride_date DATE
);

CREATE TABLE cities (
   city_name VARCHAR(50)
);

INSERT INTO rides VALUES
(1, 3001, 'Gowtham', 'Chennai', '2025-01-10'),
(2, 3001, 'Gowtham', 'Coimbatore', '2025-02-15'),
(3, 3001, 'Gowtham', 'Bangalore', '2025-03-20'),
(4, 3002, 'Anu', 'Chennai', '2025-01-12'),
(5, 3002, 'Anu', 'Bangalore', '2025-03-15');

INSERT INTO cities VALUES
('Chennai'), ('Coimbatore'), ('Bangalore');
```

Solution:

```
SELECT customer_id, customer_name
FROM rides
WHERE ride_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
GROUP BY customer_id, customer_name
HAVING COUNT(DISTINCT city) = (SELECT COUNT(*) FROM cities);
```

Explanation:

- The query counts distinct cities each customer has taken rides in last year.
- It compares this count to total cities served by company.
- Customers whose count equals total cities are returned.

This helps identify **loyal, widely active customers** for special rewards.

Problem 21: Find Drivers Who Completed More Than 5 Rides in Each City They Operate In Last Month

Problem Statement:

Identify drivers who have completed **more than 5 rides in every city** they operated in during the last month.

Create & Insert DDL (MySQL):

```
CREATE TABLE driver_rides (
   ride_id INT,
   driver_id INT,
   driver_name VARCHAR(50),
   city VARCHAR(50),
   ride_status VARCHAR(20),
   ride_date DATE
);

INSERT INTO driver_rides VALUES
(1, 401, 'Mani', 'Chennai', 'Completed', '2025-06-05'),
(2, 401, 'Mani', 'Chennai', 'Completed', '2025-06-06'),
(3, 401, 'Mani', 'Coimbatore', 'Completed', '2025-06-07'),
(4, 401, 'Mani', 'Coimbatore', 'Completed', '2025-06-08'),
(5, 401, 'Mani', 'Coimbatore', 'Completed', '2025-06-09'),
(6, 402, 'Karthik', 'Chennai', 'Completed', '2025-06-01'),
(7, 402, 'Karthik', 'Chennai', 'Cancelled', '2025-06-02');
```

Solution:

```
WITH rides_per_city AS (
   SELECT driver_id, driver_name, city,
          COUNT(CASE WHEN ride_status = 'Completed' THEN 1 END) AS completed_rides
   FROM driver_rides
   WHERE ride_date >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH)
   GROUP BY driver_id, driver_name, city
),
cities_operated AS (
   SELECT driver_id, COUNT(DISTINCT city) AS city_count
   FROM driver_rides
   WHERE ride_date >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH)
   GROUP BY driver_id
),
cities_with_5plus_rides AS (
   SELECT driver_id, COUNT(*) AS cities_with_5plus
   FROM rides_per_city
   WHERE completed_rides > 5
   GROUP BY driver_id
)
SELECT d.driver_id, d.driver_name
FROM cities_operated c
JOIN cities_with_5plus_rides d ON c.driver_id = d.driver_id
WHERE c.city_count = d.cities_with_5plus;
```

Explanation:

- `rides_per_city` counts completed rides per driver per city.

- `cities_operated` counts how many distinct cities each driver operated in last month.
- `cities_with_5plus_rides` counts cities where drivers completed more than 5 rides.
- Final step joins these counts and selects drivers who completed over 5 rides in **all** cities they operated.

This query combines aggregation and filtering across grouped data to ensure the driver meets the threshold in every city they work in.

Problem 22: Find Customers Who Used Promo Codes More Than 3 Times but Had Less Than 10 Completed Rides

Problem Statement:

Identify customers who have used promo codes more than 3 times but completed fewer than 10 rides in the last 6 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE promo_usage (
   usage_id INT,
   customer_id INT,
   customer_name VARCHAR(50),
   promo_code VARCHAR(50),
   usage_date DATE
);

CREATE TABLE ride_history (
   ride_id INT,
   customer_id INT,
   ride_status VARCHAR(20),
   ride_date DATE
);

INSERT INTO promo_usage VALUES
(1, 501, 'Gowtham', 'SAVE50', '2025-01-10'),
(2, 501, 'Gowtham', 'SAVE50', '2025-02-15'),
(3, 501, 'Gowtham', 'FREERIDE', '2025-03-20'),
(4, 501, 'Gowtham', 'SAVE50', '2025-04-25');

INSERT INTO ride_history VALUES
(1, 501, 'Completed', '2025-01-10'),
(2, 501, 'Completed', '2025-02-15'),
(3, 501, 'Cancelled', '2025-03-20'),
(4, 501, 'Completed', '2025-05-01');
```

Solution:

```
WITH promo_counts AS (
   SELECT customer_id, customer_name, COUNT(*) AS promo_usage_count
   FROM promo_usage
   WHERE usage_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
   GROUP BY customer_id, customer_name
),
completed_rides AS (
   SELECT customer_id, COUNT(*) AS completed_ride_count
   FROM ride_history
   WHERE ride_status = 'Completed' AND ride_date >= DATE_SUB(CURDATE(), INTERVAL 6
MONTH)
   GROUP BY customer_id
)
SELECT p.customer_id, p.customer_name
FROM promo_counts p
LEFT JOIN completed_rides r ON p.customer_id = r.customer_id
WHERE p.promo_usage_count > 3 AND (r.completed_ride_count < 10 OR
r.completed_ride_count IS NULL);
```

Explanation:

- `promo_counts` tallies promo code uses per customer.
- `completed_rides` counts successful rides per customer.
- Final query identifies customers using promos frequently but with low ride completion.

- This helps detect potential abuse or customers attracted by discounts but not loyal riders.

Problem 23: Find Top 3 Cities by Average Ride Distance for Each Month

Problem Statement:

Identify the **top 3 cities** with the highest **average ride distance** for each month in the past year.

Create & Insert DDL (MySQL):

```
CREATE TABLE ride_data (
   ride_id INT,
   city VARCHAR(50),
   ride_date DATE,
   distance_km DECIMAL(5,2)
);

INSERT INTO ride_data VALUES
(1, 'Chennai', '2025-01-05', 12.5),
(2, 'Coimbatore', '2025-01-10', 15.0),
(3, 'Chennai', '2025-02-15', 10.0),
(4, 'Bangalore', '2025-02-20', 20.0),
(5, 'Coimbatore', '2025-03-25', 18.0);
```

Solution:

```
WITH monthly_city_avg AS (
   SELECT city, DATE_FORMAT(ride_date, '%Y-%m') AS ride_month,
        AVG(distance_km) AS avg_distance
   FROM ride_data
   WHERE ride_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
   GROUP BY city, ride_month
),
ranked AS (
   SELECT city, ride_month, avg_distance,
        ROW_NUMBER() OVER (PARTITION BY ride_month ORDER BY avg_distance DESC)
AS rank
    FROM monthly_city_avg
)
SELECT city, ride_month, avg_distance
FROM ranked
WHERE rank <= 3
ORDER BY ride_month, rank;
```

Explanation:

- Calculate average ride distance per city per month.
- Use window function `ROW_NUMBER()` to rank cities by average distance each month.
- Filter to top 3 cities per month.
- This query demonstrates time-based ranking and filtering in SQL.

Problem 24: Identify Customers Who Increased Their Monthly Spending by More Than 30% Consecutively for 3 Months

Problem Statement:

Find customers whose **monthly spending increased by more than 30% month-over-month for 3 consecutive months** in the last 6 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE monthly_spending (
    customer_id INT,
    customer_name VARCHAR(50),
    spend_date DATE,
    amount_spent DECIMAL(10,2)
);
```

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

```
INSERT INTO monthly_spending VALUES
(1, 'Gowtham', '2024-12-15', 2000.00),
(1, 'Gowtham', '2025-01-15', 3000.00),
(1, 'Gowtham', '2025-02-15', 4200.00),
(2, 'Anu', '2025-01-10', 1500.00),
(2, 'Anu', '2025-02-10', 1200.00);
```

Solution:

```
WITH monthly_totals AS (
   SELECT customer_id, customer_name, DATE_FORMAT(spend_date, '%Y-%m') AS
spend_month,
          SUM(amount_spent) AS total_spent
   FROM monthly_spending
   WHERE spend_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
   GROUP BY customer_id, customer_name, spend_month
),
spend_growth AS (
   SELECT customer_id, customer_name, spend_month, total_spent,
          LAG(total_spent) OVER (PARTITION BY customer_id ORDER BY spend_month) AS
prev_month_spent
   FROM monthly_totals
),
growth_flag AS (
   SELECT customer_id, customer_name, spend_month,
          CASE WHEN prev_month_spent IS NOT NULL AND total_spent > 1.3 *
prev_month_spent THEN 1 ELSE 0 END AS increased_30_percent
   FROM spend_growth
)
SELECT customer_id, customer_name
FROM growth_flag
GROUP BY customer_id, customer_name
HAVING SUM(increased_30_percent) >= 3;
```

Explanation:

- Calculate total monthly spending per customer.
- Use `LAG()` to get previous month spending.

- Flag months where spending increased by more than 30%.
- Group and filter customers with at least 3 such consecutive increases.
- Useful for spotting rapidly growing customers.

Problem 25: Find Drivers Who Completed Rides in At Least 3 Different Cities in the Last Year

Problem Statement:

Identify drivers who have completed rides in **at least 3 different cities** in the past year.

Create & Insert DDL (MySQL):

```
CREATE TABLE driver_ride_data (
    driver_id INT,
    driver_name VARCHAR(50),
    city VARCHAR(50),
    ride_status VARCHAR(20),
    ride_date DATE
);

INSERT INTO driver_ride_data VALUES
(1, 'Mani', 'Chennai', 'Completed', '2024-10-10'),
(1, 'Mani', 'Coimbatore', 'Completed', '2024-11-15'),
(1, 'Mani', 'Bangalore', 'Completed', '2025-01-20'),
(2, 'Karthik', 'Chennai', 'Completed', '2025-02-25');
```

Solution:

```
SELECT driver_id, driver_name, COUNT(DISTINCT city) AS city_count
FROM driver_ride_data
WHERE ride_status = 'Completed'
  AND ride_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
GROUP BY driver_id, driver_name
HAVING city_count >= 3;
```

Explanation:

- Count distinct cities where drivers completed rides in last year.
- Filter drivers with rides in 3 or more cities.
- This helps in analyzing driver mobility and cross-city activity.

Problem 26: Identify Customers Who Frequently Used Promo Codes but Had a Low Retention Rate

Problem Statement:

Find customers who used promo codes **more than 5 times** in the last year but had **less than 3 rides** in the last 3 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE promo_usage (
   usage_id INT,
   customer_id INT,
   customer_name VARCHAR(50),
   promo_code VARCHAR(50),
   usage_date DATE
);

CREATE TABLE rides (
   ride_id INT,
   customer_id INT,
   ride_status VARCHAR(20),
   ride_date DATE
);

INSERT INTO promo_usage VALUES
(1, 1001, 'Gowtham', 'DISCOUNT10', '2024-08-01'),
(2, 1001, 'Gowtham', 'DISCOUNT10', '2024-09-15'),
(3, 1001, 'Gowtham', 'SAVE20', '2025-01-20'),
(4, 1002, 'Anu', 'DISCOUNT10', '2024-11-05');

INSERT INTO rides VALUES
(1, 1001, 'Completed', '2025-04-01'),
(2, 1001, 'Completed', '2025-05-10'),
(3, 1002, 'Completed', '2025-04-15'),
(4, 1002, 'Completed', '2025-04-20');
```

Solution:

```
WITH promo_counts AS (
   SELECT customer_id, customer_name, COUNT(*) AS promo_usage_count
   FROM promo_usage
   WHERE usage_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
   GROUP BY customer_id, customer_name
),
recent_ride_counts AS (
   SELECT customer_id, COUNT(*) AS recent_ride_count
   FROM rides
   WHERE ride_status = 'Completed' AND ride_date >= DATE_SUB(CURDATE(), INTERVAL 3
MONTH)
   GROUP BY customer_id
)
SELECT p.customer_id, p.customer_name
FROM promo_counts p
LEFT JOIN recent_ride_counts r ON p.customer_id = r.customer_id
WHERE p.promo_usage_count > 5 AND (r.recent_ride_count < 3 OR r.recent_ride_count IS
NULL);
```

Explanation:

- `promo_counts` counts the number of promo code usages per customer in the past year.
- `recent_ride_counts` counts rides completed in the last 3 months.
- Customers who use promo codes frequently but have low ride activity recently are filtered.

- Helps identify customers attracted by promotions but not retained effectively.

Problem 27: Find Top 5 Cities by Total Revenue Generated in Each Quarter

Problem Statement:

Identify the **top 5 cities by total revenue** for each quarter in the last year.

Create & Insert DDL (MySQL):

```
CREATE TABLE ride_revenue (
    ride_id INT,
    city VARCHAR(50),
    ride_date DATE,
    revenue DECIMAL(10,2)
);

INSERT INTO ride_revenue VALUES
(1, 'Chennai', '2024-01-10', 1200.00),
(2, 'Bangalore', '2024-02-15', 1300.00),
(3, 'Chennai', '2024-03-20', 1100.00),
(4, 'Coimbatore', '2024-04-05', 900.00),
(5, 'Bangalore', '2024-04-25', 1500.00);
```

Solution:

```
WITH quarterly_revenue AS (
    SELECT city,
           CONCAT(YEAR(ride_date), '-Q', QUARTER(ride_date)) AS quarter,
           SUM(revenue) AS total_revenue
    FROM ride_revenue
    WHERE ride_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
    GROUP BY city, quarter
),
ranked_cities AS (
    SELECT city, quarter, total_revenue,
        ROW_NUMBER() OVER (PARTITION BY quarter ORDER BY total_revenue DESC) AS
rank
    FROM quarterly_revenue
)
SELECT city, quarter, total_revenue
FROM ranked_cities
WHERE rank <= 5
ORDER BY quarter, rank;
```

Explanation:

- Calculates revenue per city per quarter.
- Uses window function to rank cities by revenue within each quarter.
- Selects top 5 cities for every quarter.
- Useful for strategic regional revenue analysis.

Problem 28: Identify Customers Who Increased Their Ride Frequency but Decreased Average Spend per Ride in the Last 6 Months

Problem Statement:

Find customers who have **increased the number of rides month-over-month** but have a **decreasing average spend per ride** during the last 6 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE customer_ride_spending (
    ride_id INT,
    customer_id INT,
    customer_name VARCHAR(50),
    ride_date DATE,
    amount_spent DECIMAL(10,2)
);

INSERT INTO customer_ride_spending VALUES
(1, 1001, 'Gowtham', '2024-12-15', 250.00),
(2, 1001, 'Gowtham', '2025-01-15', 200.00),
(3, 1001, 'Gowtham', '2025-02-15', 150.00),
(4, 1002, 'Anu', '2025-01-10', 300.00),
(5, 1002, 'Anu', '2025-02-10', 310.00);
```

Solution:

```
WITH monthly_stats AS (
   SELECT customer_id, customer_name, DATE_FORMAT(ride_date, '%Y-%m') AS
ride_month,
          COUNT(*) AS ride_count,
          AVG(amount_spent) AS avg_spend
   FROM customer_ride_spending
   WHERE ride_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
   GROUP BY customer_id, customer_name, ride_month
),
ranked_stats AS (
   SELECT customer_id, customer_name, ride_month, ride_count, avg_spend,
          LAG(ride_count) OVER (PARTITION BY customer_id ORDER BY ride_month) AS
prev_ride_count,
          LAG(avg_spend) OVER (PARTITION BY customer_id ORDER BY ride_month) AS
prev_avg_spend
   FROM monthly_stats
)
SELECT DISTINCT customer_id, customer_name
FROM ranked_stats
WHERE ride_count > prev_ride_count
  AND avg_spend < prev_avg_spend;
```

Explanation:

- Calculates monthly ride count and average spend per customer.
- Uses window functions to get previous month stats.
- Finds customers with increasing ride frequency but decreasing average spend.

- Identifies users who may be shifting to cheaper rides or shorter distances.

Problem 29: Find Drivers With the Highest Ratio of Night-Time Rides (10 PM - 5 AM)

Problem Statement:

Identify drivers who completed the highest **percentage of rides between 10 PM and 5 AM** in the last 3 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE driver_rides (
   ride_id INT,
   driver_id INT,
   driver_name VARCHAR(50),
   ride_start_time TIME,
   ride_status VARCHAR(20),
   ride_date DATE
);

INSERT INTO driver_rides VALUES
(1, 501, 'Mani', '22:30:00', 'Completed', '2025-04-15'),
(2, 501, 'Mani', '23:45:00', 'Completed', '2025-04-20'),
(3, 502, 'Karthik', '18:30:00', 'Completed', '2025-05-10'),
(4, 502, 'Karthik', '02:00:00', 'Completed', '2025-05-15');
```

Solution:

```
WITH ride_counts AS (
   SELECT driver_id, driver_name,
         COUNT(*) AS total_rides,
         SUM(CASE WHEN (ride_start_time >= '22:00:00' OR ride_start_time < '05:00:00') AND
ride_status = 'Completed' THEN 1 ELSE 0 END) AS night_rides
   FROM driver_rides
   WHERE ride_date >= DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
   GROUP BY driver_id, driver_name
)
SELECT driver_id, driver_name,
       (night_rides / total_rides) * 100 AS night_ride_percentage
FROM ride_counts
ORDER BY night_ride_percentage DESC
LIMIT 10;
```

Explanation:

- Counts total and night-time rides per driver.
- Night rides defined as rides starting between 10 PM and 5 AM (handling the overnight window).
- Calculates percentage and ranks drivers by night ride ratio.
- Useful for managing night shift drivers or special incentives.

Problem 30: Find Customers Who Took the Most Number of Rides But Had the Lowest Average Rating in Last 6 Months

Problem Statement:

Identify customers with the **highest ride counts** but who gave the **lowest average ride ratings** in the last 6 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE ride_feedback (
   ride_id INT,
   customer_id INT,
   customer_name VARCHAR(50),
   rating INT,
   ride_date DATE
);
```

```
INSERT INTO ride_feedback VALUES
(1, 1001, 'Gowtham', 2, '2025-01-10'),
(2, 1001, 'Gowtham', 1, '2025-02-10'),
(3, 1001, 'Gowtham', 3, '2025-03-15'),
(4, 1002, 'Anu', 5, '2025-01-05'),
(5, 1002, 'Anu', 4, '2025-02-20');
```

Solution:

```
WITH customer_stats AS (
  SELECT customer_id, customer_name,
      COUNT(*) AS ride_count,
      AVG(rating) AS avg_rating
  FROM ride_feedback
  WHERE ride_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
  GROUP BY customer_id, customer_name
)
SELECT customer_id, customer_name, ride_count, avg_rating
FROM customer_stats
ORDER BY ride_count DESC, avg_rating ASC
LIMIT 10;
```

Explanation:

- Aggregates ride count and average rating per customer.
- Orders to find customers with high ride counts but low average ratings.
- Helps identify potentially dissatisfied frequent customers for follow-up.

Problem 31: Find Customers Who Made At Least One Ride Every Week for the Last 12 Weeks

Problem Statement:

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

Identify customers who completed **at least one ride every week** for the past 12 weeks.

Create & Insert DDL (MySQL):

```
CREATE TABLE rides (
    ride_id INT,
    customer_id INT,
    customer_name VARCHAR(50),
    ride_date DATE,
    ride_status VARCHAR(20)
);

INSERT INTO rides VALUES
(1, 1001, 'Gowtham', '2025-03-01', 'Completed'),
(2, 1001, 'Gowtham', '2025-03-08', 'Completed'),
(3, 1001, 'Gowtham', '2025-05-15', 'Completed'),
(4, 1002, 'Anu', '2025-04-02', 'Completed');
```

Solution:

```
WITH weeks AS (
    SELECT DISTINCT YEARWEEK(ride_date, 1) AS year_week
    FROM rides
    WHERE ride_date >= DATE_SUB(CURDATE(), INTERVAL 12 WEEK)
),
customer_weeks AS (
    SELECT customer_id, customer_name, YEARWEEK(ride_date, 1) AS year_week
    FROM rides
    WHERE ride_status = 'Completed'
      AND ride_date >= DATE_SUB(CURDATE(), INTERVAL 12 WEEK)
    GROUP BY customer_id, customer_name, year_week
),
customer_active_weeks AS (
    SELECT customer_id, customer_name, COUNT(DISTINCT year_week) AS active_weeks
    FROM customer_weeks
    GROUP BY customer_id, customer_name
),
total_weeks AS (
    SELECT COUNT(*) AS total_weeks FROM weeks
)
SELECT c.customer_id, c.customer_name
FROM customer_active_weeks c, total_weeks t
```

```
WHERE c.active_weeks = t.total_weeks;
```

Explanation:

- `weeks` generates all distinct weeks in the last 12 weeks.
- `customer_weeks` identifies weeks where each customer completed rides.
- `customer_active_weeks` counts number of active weeks per customer.
- `total_weeks` counts total weeks in the interval.
- Final query returns customers whose active weeks equal total weeks, meaning they took rides every week.

This query combines windowed date functions, grouping, and careful filtering to capture consistent weekly user activity.

Problem 32: Identify Drivers Who Improved Their Cancellation Rate by At Least 10% Month Over Month for 3 Consecutive Months

Problem Statement:

Find drivers whose **cancellation rate improved (decreased) by at least 10% month-over-month** for 3 consecutive months.

Create & Insert DDL (MySQL):

```
CREATE TABLE driver_rides (
    ride_id INT,
    driver_id INT,
    driver_name VARCHAR(50),
    ride_status VARCHAR(20),
    ride_date DATE
);
```

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

```
INSERT INTO driver_rides VALUES
(1, 501, 'Mani', 'Cancelled', '2025-01-10'),
(2, 501, 'Mani', 'Completed', '2025-01-15'),
(3, 501, 'Mani', 'Cancelled', '2025-02-05'),
(4, 501, 'Mani', 'Completed', '2025-02-10'),
(5, 501, 'Mani', 'Completed', '2025-03-12');
```

Solution:

```
WITH monthly_stats AS (
   SELECT driver_id, driver_name, DATE_FORMAT(ride_date, '%Y-%m') AS ride_month,
        COUNT(*) AS total_rides,
        SUM(CASE WHEN ride_status = 'Cancelled' THEN 1 ELSE 0 END) AS cancelled_rides,
        SUM(CASE WHEN ride_status = 'Cancelled' THEN 1 ELSE 0 END) / COUNT(*) AS
cancellation_rate
   FROM driver_rides
   WHERE ride_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
   GROUP BY driver_id, driver_name, ride_month
),
rate_diff AS (
   SELECT driver_id, driver_name, ride_month, cancellation_rate,
        LAG(cancellation_rate) OVER (PARTITION BY driver_id ORDER BY ride_month) AS
prev_rate
   FROM monthly_stats
),
improvements AS (
   SELECT driver_id, driver_name, ride_month,
        CASE WHEN prev_rate IS NOT NULL AND prev_rate - cancellation_rate >= 0.10 THEN
1 ELSE 0 END AS improved_10_percent
   FROM rate_diff
)
SELECT driver_id, driver_name
FROM improvements
GROUP BY driver_id, driver_name
HAVING SUM(improved_10_percent) >= 3;
```

Explanation:

- `monthly_stats` computes cancellation rate per driver per month.

- `rate_diff` calculates month-over-month difference using `LAG()`.
- `improvements` flags months with ≥10% improvement.
- Final selects drivers with 3 or more such improvements consecutively.

This problem uses window functions and trend detection relevant for performance monitoring.

Problem 33: Find Customers Who Used Multiple Payment Modes but Never Used Credit Card

Problem Statement:

Identify customers who have used **more than one payment method** in last year but **never used Credit Card**.

Create & Insert DDL (MySQL):

```
CREATE TABLE customer_payments (
   payment_id INT,
   customer_id INT,
   customer_name VARCHAR(50),
   payment_method VARCHAR(50),
   payment_date DATE
);

INSERT INTO customer_payments VALUES
(1, 1001, 'Gowtham', 'Cash', '2024-08-10'),
(2, 1001, 'Gowtham', 'UPI', '2024-09-15'),
(3, 1002, 'Anu', 'Credit Card', '2024-10-10'),
(4, 1002, 'Anu', 'Cash', '2024-11-05');
```

Solution:

```
WITH payment_summary AS (
   SELECT customer_id, customer_name,
        COUNT(DISTINCT payment_method) AS distinct_methods,
        SUM(CASE WHEN payment_method = 'Credit Card' THEN 1 ELSE 0 END) AS
credit_card_used
   FROM customer_payments
   WHERE payment_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
   GROUP BY customer_id, customer_name
)
SELECT customer_id, customer_name
FROM payment_summary
WHERE distinct_methods > 1
  AND credit_card_used = 0;
```

Explanation:

- `payment_summary` counts distinct payment methods and tracks credit card usage.
- Filters customers with multiple payment modes but no credit card.
- Useful for segmenting payment behavior and designing marketing offers.

Problem 34: Identify Drivers with Increasing Average Ratings Over Last 4 Months

Problem Statement:

Find drivers whose **average rating has increased every month** for the last 4 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE driver_ratings (
   driver_id INT,
   driver_name VARCHAR(50),
   rating DECIMAL(2,1),
   rating_date DATE
);

INSERT INTO driver_ratings VALUES
(1, 'Mani', 4.5, '2025-01-10'),
(1, 'Mani', 4.6, '2025-02-15'),
(1, 'Mani', 4.7, '2025-03-20'),
(1, 'Mani', 4.8, '2025-04-25'),
(2, 'Karthik', 4.5, '2025-01-10'),
(2, 'Karthik', 4.4, '2025-02-15');
```

Solution:

```
WITH monthly_avg AS (
   SELECT driver_id, driver_name, DATE_FORMAT(rating_date, '%Y-%m') AS rating_month,
        AVG(rating) AS avg_rating
   FROM driver_ratings
   WHERE rating_date >= DATE_SUB(CURDATE(), INTERVAL 4 MONTH)
   GROUP BY driver_id, driver_name, rating_month
),
lagged AS (
   SELECT driver_id, driver_name, rating_month, avg_rating,
        LAG(avg_rating) OVER (PARTITION BY driver_id ORDER BY rating_month) AS
prev_avg_rating
   FROM monthly_avg
)
SELECT driver_id, driver_name
FROM lagged
GROUP BY driver_id, driver_name
HAVING MIN(avg_rating > COALESCE(prev_avg_rating, 0)) = 1;
```

Explanation:

- Calculate average rating per driver per month.
- Use `LAG()` to get previous month's average rating.
- Check if current month's rating is greater than previous month's for all months.
- Drivers passing this test have consistently improving ratings.

Problem 35: Find Customers Who Have Been Inactive for More Than 6 Months but Returned in Last Month

Problem Statement:

Identify customers who were inactive for over 6 months but completed at least one ride in the last month.

Create & Insert DDL (MySQL):

```
CREATE TABLE rides (
    ride_id INT,
    customer_id INT,
    customer_name VARCHAR(50),
    ride_date DATE,
    ride_status VARCHAR(20)
);

INSERT INTO rides VALUES
(1, 1001, 'Gowtham', '2024-01-10', 'Completed'),
(2, 1001, 'Gowtham', '2024-12-01', 'Completed'),
(3, 1002, 'Anu', '2024-05-20', 'Completed'),
(4, 1002, 'Anu', '2025-06-10', 'Completed');
```

Solution:

```
WITH last_active AS (
    SELECT customer_id, customer_name, MAX(ride_date) AS last_ride_date
    FROM rides
    GROUP BY customer_id, customer_name
)
SELECT la.customer_id, la.customer_name
FROM last_active la
JOIN rides r ON la.customer_id = r.customer_id
WHERE la.last_ride_date <= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
  AND r.ride_date >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH)
GROUP BY la.customer_id, la.customer_name;
```

Explanation:

- Find last ride date per customer.
- Filter customers inactive for 6+ months (`last_ride_date <= DATE_SUB(...)`).
- Check if they had any rides in the last month.
- This identifies customers who churned and returned.

Problem 36: Find Top 5 Restaurants on Zomato by Average Rating with More Than 100 Reviews Last Month

Problem Statement:

Identify the top 5 restaurants on Zomato with the highest average rating who received **more than 100 reviews** in the last month.

Create & Insert DDL (MySQL):

```
CREATE TABLE zomato_reviews (
    review_id INT,
    restaurant_id INT,
    restaurant_name VARCHAR(100),
    rating DECIMAL(2,1),
    review_date DATE
);

INSERT INTO zomato_reviews VALUES
(1, 101, 'Spicy Delight', 4.8, '2025-06-05'),
(2, 101, 'Spicy Delight', 4.9, '2025-06-10'),
(3, 102, 'Curry House', 4.7, '2025-06-15'),
(4, 103, 'Tandoori Treat', 4.9, '2025-06-20');
```

Solution:

```
WITH monthly_reviews AS (
    SELECT restaurant_id, restaurant_name,
           AVG(rating) AS avg_rating,
           COUNT(*) AS review_count
    FROM zomato_reviews
    WHERE review_date >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH)
    GROUP BY restaurant_id, restaurant_name
)
SELECT restaurant_id, restaurant_name, avg_rating, review_count
FROM monthly_reviews
WHERE review_count > 100
ORDER BY avg_rating DESC
LIMIT 5;
```

Explanation:

- The query filters reviews within the last month.
- It aggregates average ratings and counts the number of reviews per restaurant.
- Filters only restaurants with more than 100 reviews for reliability.
- Orders by average rating descending and limits to the top 5.
- Helps users identify trending popular restaurants with significant feedback.

Problem 37: Identify Flipkart Customers Who Purchased Electronics and Fashion Items in the Same Month

Problem Statement:

Find Flipkart customers who purchased both **Electronics** and **Fashion** category products in the same month during the last 6 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE flipkart_purchases (
    purchase_id INT,
    customer_id INT,
    customer_name VARCHAR(50),
    product_category VARCHAR(50),
    purchase_date DATE
);

INSERT INTO flipkart_purchases VALUES
(1, 201, 'Gowtham', 'Electronics', '2025-02-10'),
(2, 201, 'Gowtham', 'Fashion', '2025-02-15'),
(3, 202, 'Anu', 'Fashion', '2025-03-05'),
(4, 202, 'Anu', 'Electronics', '2025-04-10');
```

Solution:

```
WITH monthly_categories AS (
    SELECT customer_id, customer_name, DATE_FORMAT(purchase_date, '%Y-%m') AS
purchase_month,
          GROUP_CONCAT(DISTINCT product_category) AS categories
    FROM flipkart_purchases
    WHERE purchase_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
    GROUP BY customer_id, customer_name, purchase_month
)
SELECT DISTINCT customer_id, customer_name
FROM monthly_categories
WHERE categories LIKE '%Electronics%'
  AND categories LIKE '%Fashion%';
```

Explanation:

- Groups purchases per customer per month, concatenating distinct categories.
- Filters customers who bought both Electronics and Fashion in the same month.
- Useful for cross-category marketing and recommendations.

Problem 38: Find Dating App Users Who Sent Messages to at Least 3 Different Users in a Week

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

Problem Statement:

Identify dating app users who sent messages to **3 or more distinct users** within the same week in the last 3 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE messages (
    message_id INT,
    sender_id INT,
    sender_name VARCHAR(50),
    receiver_id INT,
    message_date DATE
);

INSERT INTO messages VALUES
(1, 301, 'Gowtham', 401, '2025-04-01'),
(2, 301, 'Gowtham', 402, '2025-04-02'),
(3, 301, 'Gowtham', 403, '2025-04-03'),
(4, 302, 'Anu', 404, '2025-05-01');
```

Solution:

```
SELECT sender_id, sender_name, YEARWEEK(message_date, 1) AS year_week,
COUNT(DISTINCT receiver_id) AS unique_receivers
FROM messages
WHERE message_date >= DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
GROUP BY sender_id, sender_name, year_week
HAVING unique_receivers >= 3;
```

Explanation:

- Groups messages by sender and week (`YEARWEEK`).
- Counts distinct receivers per sender per week.
- Filters users who messaged 3 or more unique users in a single week.

- Helps identify highly active or social users.

Problem 39: Identify Flipkart Customers Who Returned More Than 10% of Their Orders Last Quarter

Problem Statement:

Find Flipkart customers whose **return rate** (returned orders / total orders) exceeded **10%** last quarter.

Create & Insert DDL (MySQL):

```
CREATE TABLE orders (
   order_id INT,
   customer_id INT,
   customer_name VARCHAR(50),
   order_status VARCHAR(20), -- 'Delivered' or 'Returned'
   order_date DATE
);

INSERT INTO orders VALUES
(1, 401, 'Gowtham', 'Delivered', '2025-03-10'),
(2, 401, 'Gowtham', 'Returned', '2025-03-15'),
(3, 402, 'Anu', 'Delivered', '2025-04-05'),
(4, 402, 'Anu', 'Returned', '2025-04-10');
```

Solution:

```
WITH order_stats AS (
   SELECT customer_id, customer_name,
        COUNT(*) AS total_orders,
        SUM(CASE WHEN order_status = 'Returned' THEN 1 ELSE 0 END) AS
returned_orders,
        SUM(CASE WHEN order_status = 'Returned' THEN 1 ELSE 0 END)/COUNT(*) AS
return_rate
   FROM orders
```

```
    WHERE order_date BETWEEN DATE_FORMAT(DATE_SUB(CURDATE(), INTERVAL 1
QUARTER), '%Y-%m-01') AND LAST_DAY(DATE_SUB(CURDATE(), INTERVAL 1 MONTH))
    GROUP BY customer_id, customer_name
)
SELECT customer_id, customer_name, total_orders, returned_orders, return_rate
FROM order_stats
WHERE return_rate > 0.10;
```

Explanation:

- Calculates total and returned orders per customer last quarter.
- Computes return rate as ratio.
- Filters customers exceeding 10% return rate.
- Useful for detecting quality issues or problem customers.

Problem 40: Identify Dating App Users Who Liked and Messaged the Same User Within 24 Hours

Problem Statement:

Find dating app users who **liked and messaged the same user within 24 hours** in the last month.

Create & Insert DDL (MySQL):

```
CREATE TABLE likes (
    like_id INT,
    sender_id INT,
    receiver_id INT,
    like_time DATETIME
);

CREATE TABLE messages (
    message_id INT,
    sender_id INT,
    receiver_id INT,
    message_time DATETIME
);

INSERT INTO likes VALUES
(1, 501, 601, '2025-06-01 10:00:00'),
(2, 502, 602, '2025-06-05 09:00:00');

INSERT INTO messages VALUES
(1, 501, 601, '2025-06-01 15:00:00'),
(2, 502, 603, '2025-06-05 10:00:00');
```

Solution:

```
SELECT l.sender_id, l.receiver_id
FROM likes l
JOIN messages m ON l.sender_id = m.sender_id AND l.receiver_id = m.receiver_id
WHERE l.like_time BETWEEN DATE_SUB(CURDATE(), INTERVAL 1 MONTH) AND
CURDATE()
  AND m.message_time BETWEEN l.like_time AND DATE_ADD(l.like_time, INTERVAL 24
HOUR);
```

Explanation:

- Joins likes and messages on same sender and receiver.
- Filters events in the last month.
- Ensures messages happened within 24 hours after the like.
- Identifies highly engaged users with prompt interaction.

Problem 41: Find Zomato Customers Who Ordered From the Same Restaurant More Than 3 Times but Left Low Ratings

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

Problem Statement:

Identify customers who placed **more than 3 orders** from the **same restaurant** but gave an average rating less than **3** in the last 6 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE zomato_orders (
    order_id INT,
    customer_id INT,
    customer_name VARCHAR(50),
    restaurant_id INT,
    restaurant_name VARCHAR(100),
    order_date DATE,
    rating INT
);

INSERT INTO zomato_orders VALUES
(1, 1001, 'Gowtham', 201, 'Spicy Delight', '2025-01-10', 2),
(2, 1001, 'Gowtham', 201, 'Spicy Delight', '2025-02-15', 3),
(3, 1001, 'Gowtham', 201, 'Spicy Delight', '2025-03-05', 1),
(4, 1002, 'Anu', 202, 'Tandoori Treat', '2025-01-12', 4);
```

Solution:

```
WITH customer_restaurant_stats AS (
    SELECT customer_id, customer_name, restaurant_id, restaurant_name,
           COUNT(*) AS order_count,
           AVG(rating) AS avg_rating
    FROM zomato_orders
    WHERE order_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
    GROUP BY customer_id, customer_name, restaurant_id, restaurant_name
)
SELECT customer_id, customer_name, restaurant_id, restaurant_name, order_count,
avg_rating
FROM customer_restaurant_stats
WHERE order_count > 3 AND avg_rating < 3;
```

Explanation:

- Aggregates order count and average rating per customer per restaurant.
- Filters those with more than 3 orders but low average ratings.
- Useful for detecting dissatisfied repeat customers to improve service.

Problem 42: Find Flipkart Customers Who Frequently Bought from Electronics but Never Purchased Accessories

Problem Statement:

Identify customers who bought **Electronics products more than 5 times** but never purchased any **Accessories** category items in the last year.

Create & Insert DDL (MySQL):

```
CREATE TABLE flipkart_purchases (
    purchase_id INT,
    customer_id INT,
    customer_name VARCHAR(50),
    product_category VARCHAR(50),
    purchase_date DATE
);

INSERT INTO flipkart_purchases VALUES
(1, 301, 'Gowtham', 'Electronics', '2024-11-10'),
(2, 301, 'Gowtham', 'Electronics', '2025-01-15'),
(3, 302, 'Anu', 'Accessories', '2024-12-05'),
(4, 301, 'Gowtham', 'Electronics', '2025-03-20');
```

Solution:

```
WITH electronics_purchases AS (
    SELECT customer_id, customer_name, COUNT(*) AS electronics_count
    FROM flipkart_purchases
    WHERE product_category = 'Electronics'
```

```
     AND purchase_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
   GROUP BY customer_id, customer_name
),
accessory_purchases AS (
   SELECT DISTINCT customer_id
   FROM flipkart_purchases
   WHERE product_category = 'Accessories'
     AND purchase_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
)
SELECT e.customer_id, e.customer_name, e.electronics_count
FROM electronics_purchases e
LEFT JOIN accessory_purchases a ON e.customer_id = a.customer_id
WHERE e.electronics_count > 5 AND a.customer_id IS NULL;
```

Explanation:

- Counts electronics purchases per customer.
- Checks which customers bought accessories.
- Filters customers who bought electronics often but never accessories.
- Valuable for targeted accessory marketing to these customers.

Problem 43: Identify Dating App Users Who Sent Messages to At Least 5 Different Users But Received Replies from Only One

Problem Statement:

Find dating app users who messaged **5 or more distinct users** but received replies from **only one** user in the last 3 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE messages (
    message_id INT,
    sender_id INT,
    sender_name VARCHAR(50),
    receiver_id INT,
    receiver_name VARCHAR(50),
    message_date DATE
);

CREATE TABLE replies (
    reply_id INT,
    sender_id INT,
    receiver_id INT,
    reply_date DATE
);

INSERT INTO messages VALUES
(1, 401, 'Gowtham', 501, 'Sita', '2025-04-01'),
(2, 401, 'Gowtham', 502, 'Ravi', '2025-04-02'),
(3, 401, 'Gowtham', 503, 'Priya', '2025-04-03'),
(4, 401, 'Gowtham', 504, 'Karan', '2025-04-04'),
(5, 401, 'Gowtham', 505, 'Deepa', '2025-04-05');

INSERT INTO replies VALUES
(1, 501, 401, '2025-04-06');
```

Solution:

```
WITH sent_counts AS (
    SELECT sender_id, sender_name, COUNT(DISTINCT receiver_id) AS distinct_receivers
    FROM messages
    WHERE message_date >= DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
    GROUP BY sender_id, sender_name
),
reply_counts AS (
    SELECT receiver_id, COUNT(DISTINCT sender_id) AS distinct_repliers
    FROM replies
    WHERE reply_date >= DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
    GROUP BY receiver_id
)
SELECT s.sender_id, s.sender_name
FROM sent_counts s
LEFT JOIN reply_counts r ON s.sender_id = r.receiver_id
WHERE s.distinct_receivers >= 5
```

```
AND (r.distinct_repliers = 1 OR r.distinct_repliers IS NULL);
```

Explanation:

- Counts distinct users messaged per sender.
- Counts distinct users who replied to the sender.
- Filters users who messaged many but got very few replies.
- Useful for understanding user engagement or mismatches.

Problem 44: Find Flipkart Customers Who Purchased Products Across the Maximum Number of Categories

Problem Statement:

Identify customers who purchased products in the **highest number of distinct categories** during the last year.

Create & Insert DDL (MySQL):

```
CREATE TABLE flipkart_purchases (
    purchase_id INT,
    customer_id INT,
    customer_name VARCHAR(50),
    product_category VARCHAR(50),
    purchase_date DATE
);

INSERT INTO flipkart_purchases VALUES
(1, 501, 'Gowtham', 'Electronics', '2024-08-10'),
(2, 501, 'Gowtham', 'Books', '2024-09-15'),
(3, 501, 'Gowtham', 'Fashion', '2025-01-20'),
(4, 502, 'Anu', 'Books', '2025-02-10');
```

Solution:

```
WITH category_counts AS (
   SELECT customer_id, customer_name, COUNT(DISTINCT product_category) AS
category_count
   FROM flipkart_purchases
   WHERE purchase_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
   GROUP BY customer_id, customer_name
),
max_count AS (
   SELECT MAX(category_count) AS max_category_count FROM category_counts
)
SELECT customer_id, customer_name, category_count
FROM category_counts
WHERE category_count = (SELECT max_category_count FROM max_count);
```

Explanation:

- Counts distinct categories purchased per customer.
- Finds the maximum number of categories bought by any customer.
- Selects customers with that maximum.
- Useful for identifying highly diverse shoppers.

Problem 45: Identify Dating App Users Who Liked More Than 20 Profiles but Matched With Less Than 5

Problem Statement:

Find dating app users who **liked over 20 profiles** but matched with **fewer than 5** users in the last 3 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE likes (
    like_id INT,
    user_id INT,
    liked_user_id INT,
    like_date DATE
);

CREATE TABLE matches (
    match_id INT,
    user_id INT,
    matched_user_id INT,
    match_date DATE
);

INSERT INTO likes VALUES
(1, 301, 401, '2025-03-01'),
(2, 301, 402, '2025-03-02'),
(3, 301, 403, '2025-03-03'),
(4, 301, 404, '2025-03-04'),
(5, 301, 405, '2025-03-05');

INSERT INTO matches VALUES
(1, 301, 401, '2025-03-10'),
(2, 301, 402, '2025-03-12');
```

Solution:

```
WITH like_counts AS (
    SELECT user_id, COUNT(*) AS total_likes
    FROM likes
    WHERE like_date >= DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
    GROUP BY user_id
),
match_counts AS (
    SELECT user_id, COUNT(*) AS total_matches
    FROM matches
    WHERE match_date >= DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
    GROUP BY user_id
)
SELECT l.user_id, l.total_likes, COALESCE(m.total_matches, 0) AS total_matches
FROM like_counts l
LEFT JOIN match_counts m ON l.user_id = m.user_id
WHERE l.total_likes > 20
  AND COALESCE(m.total_matches, 0) < 5;
```

Explanation:

- Counts likes and matches per user separately.
- Joins to combine counts.
- Filters users with many likes but few matches.
- Helps identify potential issues in user engagement or app algorithm.

Problem 46: Find Zomato Restaurants With Increasing Monthly Order Counts for 4 Consecutive Months

Problem Statement:

Identify restaurants on Zomato whose **monthly order counts increased consecutively for 4 months**.

Create & Insert DDL (MySQL):

```
CREATE TABLE zomato_orders (
  order_id INT,
  restaurant_id INT,
  restaurant_name VARCHAR(100),
  order_date DATE
);

INSERT INTO zomato_orders VALUES
(1, 101, 'Spicy Delight', '2025-01-10'),
(2, 101, 'Spicy Delight', '2025-02-10'),
(3, 101, 'Spicy Delight', '2025-03-10'),
(4, 101, 'Spicy Delight', '2025-04-10'),
(5, 102, 'Tandoori Treat', '2025-01-15'),
(6, 102, 'Tandoori Treat', '2025-02-15'),
(7, 102, 'Tandoori Treat', '2025-03-15');
```

Solution:

```
WITH monthly_counts AS (
   SELECT restaurant_id, restaurant_name, DATE_FORMAT(order_date, '%Y-%m') AS
order_month,
        COUNT(*) AS orders_count
   FROM zomato_orders
   WHERE order_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
   GROUP BY restaurant_id, restaurant_name, order_month
),
lagged AS (
   SELECT restaurant_id, restaurant_name, order_month, orders_count,
        LAG(orders_count) OVER (PARTITION BY restaurant_id ORDER BY order_month) AS
prev_month_orders
   FROM monthly_counts
)
SELECT restaurant_id, restaurant_name
FROM lagged
GROUP BY restaurant_id, restaurant_name
HAVING MIN(orders_count > COALESCE(prev_month_orders, 0)) = 1
  AND COUNT(*) >= 4;
```

Explanation:

- Calculate monthly order counts per restaurant.
- Use `LAG()` window function to compare current and previous months' orders.
- Filter restaurants with strictly increasing order counts for at least 4 months.
- Valuable for identifying trending popular restaurants.

Problem 47: Identify Flipkart Customers Who Purchased Electronics and Subsequently Purchased Accessories Within 15 Days

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

Problem Statement:

Find customers who bought **Electronics products** and then purchased **Accessories** within 15 days afterward in the last 6 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE flipkart_purchases (
    purchase_id INT,
    customer_id INT,
    customer_name VARCHAR(50),
    product_category VARCHAR(50),
    purchase_date DATE
);

INSERT INTO flipkart_purchases VALUES
(1, 301, 'Gowtham', 'Electronics', '2025-01-01'),
(2, 301, 'Gowtham', 'Accessories', '2025-01-10'),
(3, 302, 'Anu', 'Electronics', '2025-02-05'),
(4, 302, 'Anu', 'Accessories', '2025-03-01');
```

Solution:

```
WITH electronics_purchases AS (
    SELECT customer_id, customer_name, purchase_date
    FROM flipkart_purchases
    WHERE product_category = 'Electronics'
      AND purchase_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
),
accessory_purchases AS (
    SELECT customer_id, customer_name, purchase_date
    FROM flipkart_purchases
    WHERE product_category = 'Accessories'
      AND purchase_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
)
SELECT DISTINCT e.customer_id, e.customer_name
FROM electronics_purchases e
JOIN accessory_purchases a ON e.customer_id = a.customer_id
WHERE a.purchase_date BETWEEN e.purchase_date AND DATE_ADD(e.purchase_date,
INTERVAL 15 DAY);
```

Explanation:

- Separates electronics and accessory purchases.
- Joins on customer to find accessory purchases within 15 days after electronics.
- Useful for understanding cross-category purchase behavior and planning bundled offers.

Problem 48: Find Dating App Users Who Sent Messages to the Same User Multiple Times but Received No Replies

Problem Statement:

Identify dating app users who messaged the **same user multiple times** but never received any reply in the last 3 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE messages (
    message_id INT,
    sender_id INT,
    sender_name VARCHAR(50),
    receiver_id INT,
    message_date DATE
);

CREATE TABLE replies (
    reply_id INT,
    sender_id INT,
    receiver_id INT,
    reply_date DATE
);

INSERT INTO messages VALUES
(1, 401, 'Gowtham', 501, '2025-04-01'),
(2, 401, 'Gowtham', 501, '2025-04-05'),
(3, 402, 'Anu', 502, '2025-05-01');

INSERT INTO replies VALUES
```

(1, 501, 401, '2025-04-10');

Solution:

```
WITH multiple_messages AS (
   SELECT sender_id, sender_name, receiver_id, COUNT(*) AS msg_count
   FROM messages
   WHERE message_date >= DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
   GROUP BY sender_id, sender_name, receiver_id
   HAVING msg_count > 1
),
received_replies AS (
   SELECT DISTINCT receiver_id, sender_id
   FROM replies
   WHERE reply_date >= DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
)
SELECT m.sender_id, m.sender_name, m.receiver_id
FROM multiple_messages m
LEFT JOIN received_replies r ON m.sender_id = r.receiver_id AND m.receiver_id = r.sender_id
WHERE r.sender_id IS NULL;
```

Explanation:

- Identifies users who messaged the same person multiple times.
- Checks if any replies were received from those users.
- Filters those with zero replies despite multiple messages.
- This helps spot one-sided conversations or possible harassment cases.

Problem 49: Identify Flipkart Customers Who Frequently Purchased High-Value Items (Over ₹10,000) but Rarely Used Offers

Problem Statement:

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

Find customers who bought products priced over ₹10,000 **more than 5 times** but used offers **less than 2 times** in the last year.

Create & Insert DDL (MySQL):

```
CREATE TABLE purchases (
    purchase_id INT,
    customer_id INT,
    customer_name VARCHAR(50),
    product_price DECIMAL(10,2),
    used_offer BOOLEAN,
    purchase_date DATE
);

INSERT INTO purchases VALUES
(1, 601, 'Gowtham', 12000.00, TRUE, '2024-11-10'),
(2, 601, 'Gowtham', 15000.00, FALSE, '2025-01-20'),
(3, 601, 'Gowtham', 20000.00, FALSE, '2025-02-15'),
(4, 602, 'Anu', 11000.00, TRUE, '2025-03-10');
```

Solution:

```
WITH high_value_purchases AS (
    SELECT customer_id, customer_name,
           COUNT(*) AS high_value_count,
           SUM(CASE WHEN used_offer THEN 1 ELSE 0 END) AS offer_usage_count
    FROM purchases
    WHERE product_price > 10000
      AND purchase_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
    GROUP BY customer_id, customer_name
)
SELECT customer_id, customer_name, high_value_count, offer_usage_count
FROM high_value_purchases
WHERE high_value_count > 5
  AND offer_usage_count < 2;
```

Explanation:

- Counts high-value purchases and offer usage per customer.

- Filters customers with many expensive buys but low offer utilization.
- Helps target customers who value premium products but don't seek discounts.

Problem 50: Find Dating App Users Who Swiped Right on the Same User More Than 3 Times but Never Matched

Problem Statement:

Identify users who swiped right **more than 3 times** on the same profile but never formed a match in the last 6 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE swipes (
  swipe_id INT,
  user_id INT,
  swiped_user_id INT,
  swipe_direction VARCHAR(10), -- 'right' or 'left'
  swipe_date DATE
);

CREATE TABLE matches (
  match_id INT,
  user_id INT,
  matched_user_id INT,
  match_date DATE
);

INSERT INTO swipes VALUES
(1, 701, 801, 'right', '2024-12-10'),
(2, 701, 801, 'right', '2025-01-10'),
(3, 701, 801, 'right', '2025-02-15'),
(4, 701, 801, 'right', '2025-03-05'),
(5, 702, 802, 'right', '2025-01-12');

INSERT INTO matches VALUES
(1, 702, 802, '2025-02-01');
```

Solution:

```
WITH right_swipes AS (
    SELECT user_id, swiped_user_id, COUNT(*) AS right_swipe_count
    FROM swipes
    WHERE swipe_direction = 'right'
      AND swipe_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
    GROUP BY user_id, swiped_user_id
    HAVING right_swipe_count > 3
),
matches AS (
    SELECT user_id, matched_user_id
    FROM matches
    WHERE match_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
)
SELECT r.user_id, r.swiped_user_id
FROM right_swipes r
LEFT JOIN matches m ON r.user_id = m.user_id AND r.swiped_user_id = m.matched_user_id
WHERE m.user_id IS NULL;
```

Explanation:

- Counts right swipes per user per profile.
- Filters users swiping right more than 3 times on the same profile.
- Left joins matches to check if a match occurred.
- Selects users with frequent swipes but no matches.
- Useful for improving matching algorithms or user experience.

Problem 51: Find Zomato Customers Who Frequently Ordered Late Night (10 PM to 2 AM) but Have Low Average Tip Amount

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

Problem Statement:

Identify customers who placed **more than 10 orders between 10 PM and 2 AM** in the last 6 months but have an average tip amount less than ₹20.

Create & Insert DDL (MySQL):

```
CREATE TABLE zomato_orders (
    order_id INT,
    customer_id INT,
    customer_name VARCHAR(50),
    order_time TIME,
    tip_amount DECIMAL(5,2),
    order_date DATE
);

INSERT INTO zomato_orders VALUES
(1, 1001, 'Gowtham', '22:30:00', 10.00, '2025-01-15'),
(2, 1001, 'Gowtham', '23:15:00', 5.00, '2025-02-10'),
(3, 1001, 'Gowtham', '01:45:00', 15.00, '2025-03-05'),
(4, 1002, 'Anu', '21:00:00', 25.00, '2025-01-20');
```

Solution:

```
WITH late_night_orders AS (
    SELECT customer_id, customer_name,
           COUNT(*) AS order_count,
           AVG(tip_amount) AS avg_tip
    FROM zomato_orders
    WHERE (order_time >= '22:00:00' OR order_time <= '02:00:00')
      AND order_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
    GROUP BY customer_id, customer_name
)
SELECT customer_id, customer_name, order_count, avg_tip
FROM late_night_orders
WHERE order_count > 10 AND avg_tip < 20;
```

Explanation:

- The query filters orders placed during late-night hours (10 PM to 2 AM), taking care to handle the overnight time window by using OR condition.
- Aggregates order count and average tip amount per customer.
- Filters for customers who order frequently late at night but tip below ₹20 on average.
- Useful for targeted promotion or service enhancement for late-night diners.

Problem 52: Identify Flipkart Customers Who Have Made Increasing Monthly Purchases But Decreasing Average Order Value in Last 4 Months

Problem Statement:

Find customers who have increased their **monthly purchase count every month** but whose **average order value has decreased month-over-month** in the last 4 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE flipkart_orders (
   order_id INT,
   customer_id INT,
   customer_name VARCHAR(50),
   order_date DATE,
   order_value DECIMAL(10,2)
);

INSERT INTO flipkart_orders VALUES
(1, 201, 'Gowtham', '2025-01-15', 2500.00),
(2, 201, 'Gowtham', '2025-02-15', 2200.00),
(3, 201, 'Gowtham', '2025-03-15', 2000.00),
(4, 201, 'Gowtham', '2025-04-15', 1800.00);
```

Solution:

```
WITH monthly_stats AS (
    SELECT customer_id, customer_name, DATE_FORMAT(order_date, '%Y-%m') AS
order_month,
           COUNT(*) AS order_count,
           AVG(order_value) AS avg_order_value
    FROM flipkart_orders
    WHERE order_date >= DATE_SUB(CURDATE(), INTERVAL 4 MONTH)
    GROUP BY customer_id, customer_name, order_month
),
lagged_stats AS (
    SELECT customer_id, customer_name, order_month, order_count, avg_order_value,
           LAG(order_count) OVER (PARTITION BY customer_id ORDER BY order_month) AS
prev_order_count,
           LAG(avg_order_value) OVER (PARTITION BY customer_id ORDER BY order_month)
AS prev_avg_order_value
    FROM monthly_stats
)
SELECT DISTINCT customer_id, customer_name
FROM lagged_stats
WHERE order_count > COALESCE(prev_order_count, 0)
  AND avg_order_value < COALESCE(prev_avg_order_value, 100000);
```

Explanation:

- `monthly_stats` computes monthly order counts and average order value per customer.
- `lagged_stats` uses window functions to compare each month's stats with the previous month.
- Filters customers with increasing order counts but decreasing average order value.
- This can highlight customers buying more but cheaper products or smaller orders.

Problem 53: Find Dating App Users Who Liked Profiles But Never Sent Messages

Problem Statement:

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

Identify dating app users who have liked at least 10 profiles but **never sent a message** to any matched user in the last 3 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE likes (
   like_id INT,
   user_id INT,
   liked_user_id INT,
   like_date DATE
);

CREATE TABLE messages (
   message_id INT,
   sender_id INT,
   receiver_id INT,
   message_date DATE
);

INSERT INTO likes VALUES
(1, 301, 401, '2025-03-01'),
(2, 301, 402, '2025-03-05'),
(3, 301, 403, '2025-03-10');

INSERT INTO messages VALUES
(1, 302, 401, '2025-03-15');
```

Solution:

```
WITH like_counts AS (
   SELECT user_id, COUNT(*) AS total_likes
   FROM likes
   WHERE like_date >= DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
   GROUP BY user_id
),
message_senders AS (
   SELECT DISTINCT sender_id
   FROM messages
   WHERE message_date >= DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
)
SELECT l.user_id, l.total_likes
FROM like_counts l
```

```
LEFT JOIN message_senders m ON l.user_id = m.sender_id
WHERE l.total_likes >= 10 AND m.sender_id IS NULL;
```

Explanation:

- Counts total likes per user.
- Finds distinct users who sent messages.
- Filters users with many likes but zero messages sent.
- Important to identify users who are passive or hesitant to engage beyond liking.

Problem 54: Identify Flipkart Customers Who Frequently Bought Products but Never Reviewed

Problem Statement:

Find Flipkart customers who have made **more than 20 purchases** in the last year but **never submitted a product review**.

Create & Insert DDL (MySQL):

```
CREATE TABLE purchases (
   purchase_id INT,
   customer_id INT,
   customer_name VARCHAR(50),
   purchase_date DATE
);

CREATE TABLE reviews (
   review_id INT,
   customer_id INT,
   product_id INT,
   review_date DATE
);
```

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

```
INSERT INTO purchases VALUES
(1, 401, 'Gowtham', '2024-08-10'),
(2, 401, 'Gowtham', '2024-09-15'),
(3, 401, 'Gowtham', '2024-10-20');

INSERT INTO reviews VALUES
(1, 402, 101, '2025-01-05');
```

Solution:

```
WITH purchase_counts AS (
   SELECT customer_id, customer_name, COUNT(*) AS total_purchases
   FROM purchases
   WHERE purchase_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
   GROUP BY customer_id, customer_name
),
reviewers AS (
   SELECT DISTINCT customer_id
   FROM reviews
   WHERE review_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
)
SELECT p.customer_id, p.customer_name, p.total_purchases
FROM purchase_counts p
LEFT JOIN reviewers r ON p.customer_id = r.customer_id
WHERE p.total_purchases > 20 AND r.customer_id IS NULL;
```

Explanation:

- Aggregates purchase counts per customer.
- Identifies customers who submitted reviews.
- Finds customers with many purchases but no reviews.
- Valuable to target for feedback campaigns.

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

Problem 55: Find Dating App Users Who Matched With More Than 10 Users But Messaged Fewer Than 3

Problem Statement:

Identify dating app users who have matched with **more than 10 users** but sent messages to **fewer than 3** of them in the last 6 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE matches (
    match_id INT,
    user_id INT,
    matched_user_id INT,
    match_date DATE
);

CREATE TABLE messages (
    message_id INT,
    sender_id INT,
    receiver_id INT,
    message_date DATE
);

INSERT INTO matches VALUES
(1, 501, 601, '2024-12-01'),
(2, 501, 602, '2025-01-15'),
(3, 501, 603, '2025-02-20');

INSERT INTO messages VALUES
(1, 501, 601, '2025-01-05');
```

Solution:

```
WITH match_counts AS (
    SELECT user_id, COUNT(DISTINCT matched_user_id) AS total_matches
    FROM matches
    WHERE match_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
    GROUP BY user_id
),
message_counts AS (
    SELECT sender_id, COUNT(DISTINCT receiver_id) AS total_messages
    FROM messages
    WHERE message_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
    GROUP BY sender_id
)
SELECT m.user_id, m.total_matches, COALESCE(msg.total_messages, 0) AS total_messages
FROM match_counts m
LEFT JOIN message_counts msg ON m.user_id = msg.sender_id
WHERE m.total_matches > 10 AND COALESCE(msg.total_messages, 0) < 3;
```

Explanation:

- Counts distinct matches per user.
- Counts distinct message recipients per user.
- Filters users with many matches but low messaging activity.
- Helps improve engagement strategies.

Problem 56: Find Zomato Restaurants That Had a Drop in Average Rating by More Than 0.5 in Consecutive Months

Problem Statement:

Identify restaurants on Zomato whose **average rating dropped by more than 0.5** between two consecutive months in the last 6 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE zomato_ratings (
    restaurant_id INT,
    restaurant_name VARCHAR(100),
    rating DECIMAL(2,1),
    rating_date DATE
);

INSERT INTO zomato_ratings VALUES
(1, 101, 'Spicy Delight', 4.8, '2025-03-10'),
(2, 101, 'Spicy Delight', 4.2, '2025-04-10'),
(3, 102, 'Tandoori Treat', 4.5, '2025-03-15'),
(4, 102, 'Tandoori Treat', 4.6, '2025-04-15');
```

Solution:

```
WITH monthly_avg AS (
   SELECT restaurant_id, restaurant_name, DATE_FORMAT(rating_date, '%Y-%m') AS
rating_month,
          AVG(rating) AS avg_rating
   FROM zomato_ratings
   WHERE rating_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
   GROUP BY restaurant_id, restaurant_name, rating_month
),
rating_diff AS (
   SELECT restaurant_id, restaurant_name, rating_month, avg_rating,
          LAG(avg_rating) OVER (PARTITION BY restaurant_id ORDER BY rating_month) AS
prev_avg_rating
   FROM monthly_avg
)
SELECT restaurant_id, restaurant_name, rating_month, avg_rating, prev_avg_rating,
      (prev_avg_rating - avg_rating) AS rating_drop
FROM rating_diff
WHERE prev_avg_rating IS NOT NULL
  AND (prev_avg_rating - avg_rating) > 0.5;
```

Explanation:

- Calculates average monthly ratings per restaurant.
- Uses `LAG()` window function to compare current month's rating with previous month.
- Filters restaurants with rating drop exceeding 0.5 points.
- Helps identify restaurants facing quality issues or customer dissatisfaction.

Problem 57: Find Flipkart Customers Who Purchased Electronics and Returned More Than 20% of Them

Problem Statement:

Identify customers who bought **Electronics** category items and had a **return rate greater than 20%** in the last year.

Create & Insert DDL (MySQL):

```
CREATE TABLE flipkart_orders (
   order_id INT,
   customer_id INT,
   customer_name VARCHAR(50),
   product_category VARCHAR(50),
   order_status VARCHAR(20),
   order_date DATE
);

INSERT INTO flipkart_orders VALUES
(1, 401, 'Gowtham', 'Electronics', 'Delivered', '2024-09-10'),
(2, 401, 'Gowtham', 'Electronics', 'Returned', '2024-10-15'),
(3, 401, 'Gowtham', 'Electronics', 'Delivered', '2025-01-20'),
(4, 402, 'Anu', 'Fashion', 'Delivered', '2025-02-10');
```

Solution:

```
WITH customer_orders AS (
   SELECT customer_id, customer_name,
        COUNT(*) AS total_orders,
        SUM(CASE WHEN order_status = 'Returned' THEN 1 ELSE 0 END) AS returned_orders
   FROM flipkart_orders
   WHERE product_category = 'Electronics'
     AND order_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
   GROUP BY customer_id, customer_name
)
SELECT customer_id, customer_name, total_orders, returned_orders,
      (returned_orders / total_orders) AS return_rate
```

```
FROM customer_orders
WHERE return_rate > 0.20;
```

Explanation:

- Filters electronics orders in last year.
- Calculates total and returned orders per customer.
- Calculates return rate and filters customers exceeding 20%.
- Useful for identifying customers with potential dissatisfaction or misuse.

Problem 58: Identify Dating App Users Who Matched With More Than 5 Users and Sent Messages to All

Problem Statement:

Find dating app users who matched with **more than 5 users** and sent messages to **all** matched users in the last 3 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE matches (
    match_id INT,
    user_id INT,
    matched_user_id INT,
    match_date DATE
);

CREATE TABLE messages (
    message_id INT,
    sender_id INT,
    receiver_id INT,
    message_date DATE
);
```

```
INSERT INTO matches VALUES
(1, 501, 601, '2025-02-01'),
(2, 501, 602, '2025-02-15'),
(3, 501, 603, '2025-03-01'),
(4, 501, 604, '2025-03-15'),
(5, 501, 605, '2025-04-01'),
(6, 501, 606, '2025-04-10');

INSERT INTO messages VALUES
(1, 501, 601, '2025-02-10'),
(2, 501, 602, '2025-02-20'),
(3, 501, 603, '2025-03-05'),
(4, 501, 604, '2025-03-20'),
(5, 501, 605, '2025-04-05'),
(6, 501, 606, '2025-04-15');
```

Solution:

```
WITH match_counts AS (
   SELECT user_id, COUNT(DISTINCT matched_user_id) AS total_matches
   FROM matches
   WHERE match_date >= DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
   GROUP BY user_id
),
message_counts AS (
   SELECT sender_id, COUNT(DISTINCT receiver_id) AS total_messages
   FROM messages
   WHERE message_date >= DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
   GROUP BY sender_id
)
SELECT m.user_id
FROM match_counts m
JOIN message_counts msg ON m.user_id = msg.sender_id
WHERE m.total_matches > 5
  AND m.total_matches = msg.total_messages;
```

Explanation:

- Counts distinct matches and messages per user.

- Filters users with more than 5 matches.
- Ensures messaging count matches match count, implying all matched users were messaged.
- Useful for identifying highly engaged users.

Problem 59: Find Flipkart Customers Who Bought the Same Product More Than Twice But Reviewed Only Once

Problem Statement:

Identify customers who purchased the **same product more than twice** but have submitted only **one review** for that product in the last year.

Create & Insert DDL (MySQL):

```
CREATE TABLE purchases (
    purchase_id INT,
    customer_id INT,
    customer_name VARCHAR(50),
    product_id INT,
    purchase_date DATE
);

CREATE TABLE reviews (
    review_id INT,
    customer_id INT,
    product_id INT,
    review_date DATE
);

INSERT INTO purchases VALUES
(1, 701, 'Gowtham', 1001, '2024-08-10'),
(2, 701, 'Gowtham', 1001, '2024-10-05'),
(3, 701, 'Gowtham', 1001, '2025-01-15'),
(4, 702, 'Anu', 1002, '2024-09-10');

INSERT INTO reviews VALUES
```

(1, 701, 1001, '2024-10-10');

Solution:

```
WITH purchase_counts AS (
   SELECT customer_id, product_id, COUNT(*) AS purchase_count
   FROM purchases
   WHERE purchase_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
   GROUP BY customer_id, product_id
),
review_counts AS (
   SELECT customer_id, product_id, COUNT(*) AS review_count
   FROM reviews
   WHERE review_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
   GROUP BY customer_id, product_id
)
SELECT p.customer_id, p.product_id, p.purchase_count, COALESCE(r.review_count, 0) AS
review_count
FROM purchase_counts p
LEFT JOIN review_counts r ON p.customer_id = r.customer_id AND p.product_id = r.product_id
WHERE p.purchase_count > 2 AND COALESCE(r.review_count, 0) = 1;
```

Explanation:

- Counts purchases and reviews per customer-product pair.
- Filters pairs where purchase count is more than 2 but review count is exactly 1.
- Helps identify customers potentially reluctant to review multiple times despite repeat purchases.

Problem 60: Identify Dating App Users Who Received More Than 10 Likes But Never Matched

Problem Statement:

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

Find dating app users who were **liked by more than 10 different users** but never formed a match in the last 6 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE likes (
   like_id INT,
   liked_user_id INT,
   liker_id INT,
   like_date DATE
);

CREATE TABLE matches (
   match_id INT,
   user_id INT,
   matched_user_id INT,
   match_date DATE
);

INSERT INTO likes VALUES
(1, 801, 901, '2025-01-10'),
(2, 801, 902, '2025-02-15'),
(3, 801, 903, '2025-03-20'),
(4, 801, 904, '2025-04-25'),
(5, 801, 905, '2025-05-10');

INSERT INTO matches VALUES
(1, 801, 906, '2025-03-15');
```

Solution:

```
WITH like_counts AS (
   SELECT liked_user_id, COUNT(DISTINCT liker_id) AS total_likes
   FROM likes
   WHERE like_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
   GROUP BY liked_user_id
),
matched_users AS (
   SELECT DISTINCT user_id
   FROM matches
   WHERE match_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
)
```

```
SELECT l.liked_user_id, l.total_likes
FROM like_counts l
LEFT JOIN matched_users m ON l.liked_user_id = m.user_id
WHERE l.total_likes > 10
  AND m.user_id IS NULL;
```

Explanation:

- Counts distinct likers per user.
- Identifies users who had matches.
- Filters users with high likes but no matches.
- Useful for platform matchmaking quality analysis.

Problem 61: Identify Zomato Customers Who Ordered from More Than 3 Different Cuisines but Only Rated One Cuisine

Problem Statement:

Find customers who ordered from **more than 3 distinct cuisines** but gave ratings for only **one cuisine** in the last 6 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE zomato_orders (
    order_id INT,
    customer_id INT,
    customer_name VARCHAR(50),
    cuisine VARCHAR(50),
    order_date DATE
);
```

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

```
CREATE TABLE zomato_ratings (
    rating_id INT,
    customer_id INT,
    cuisine VARCHAR(50),
    rating INT,
    rating_date DATE
);

INSERT INTO zomato_orders VALUES
(1, 1001, 'Gowtham', 'South Indian', '2025-01-10'),
(2, 1001, 'Gowtham', 'North Indian', '2025-02-15'),
(3, 1001, 'Gowtham', 'Chinese', '2025-03-20'),
(4, 1001, 'Gowtham', 'Italian', '2025-04-25');

INSERT INTO zomato_ratings VALUES
(1, 1001, 'South Indian', 4, '2025-01-15');
```

Solution:

```
WITH cuisine_orders AS (
    SELECT customer_id, customer_name, COUNT(DISTINCT cuisine) AS cuisine_count
    FROM zomato_orders
    WHERE order_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
    GROUP BY customer_id, customer_name
),
cuisine_ratings AS (
    SELECT customer_id, COUNT(DISTINCT cuisine) AS rated_cuisines
    FROM zomato_ratings
    WHERE rating_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
    GROUP BY customer_id
)
SELECT o.customer_id, o.customer_name, o.cuisine_count, COALESCE(r.rated_cuisines, 0)
AS rated_cuisines
FROM cuisine_orders o
LEFT JOIN cuisine_ratings r ON o.customer_id = r.customer_id
WHERE o.cuisine_count > 3 AND COALESCE(r.rated_cuisines, 0) = 1;
```

Explanation:

- Counts distinct cuisines ordered and rated per customer.

- Filters customers who ordered from many cuisines but rated only one.
- This may indicate limited feedback despite diverse usage, useful for improving review requests.

Problem 62: Find Flipkart Customers Who Purchased More Than 10 Items in a Single Category But Never Bought From Others

Problem Statement:

Identify customers who purchased **more than 10 products in a single category** but never purchased from other categories in the last year.

Create & Insert DDL (MySQL):

```
CREATE TABLE flipkart_purchases (
   purchase_id INT,
   customer_id INT,
   customer_name VARCHAR(50),
   product_category VARCHAR(50),
   purchase_date DATE
);

INSERT INTO flipkart_purchases VALUES
(1, 501, 'Gowtham', 'Books', '2024-09-10'),
(2, 501, 'Gowtham', 'Books', '2024-10-15'),
(3, 501, 'Gowtham', 'Books', '2025-02-20'),
(4, 502, 'Anu', 'Electronics', '2025-03-15'),
(5, 502, 'Anu', 'Fashion', '2025-04-10');
```

Solution:

```
WITH category_counts AS (
   SELECT customer_id, customer_name, product_category, COUNT(*) AS purchase_count
   FROM flipkart_purchases
   WHERE purchase_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
```

```
    GROUP BY customer_id, customer_name, product_category
),
total_categories AS (
    SELECT customer_id, COUNT(DISTINCT product_category) AS category_count
    FROM flipkart_purchases
    WHERE purchase_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
    GROUP BY customer_id
)
SELECT c.customer_id, c.customer_name, c.product_category, c.purchase_count
FROM category_counts c
JOIN total_categories t ON c.customer_id = t.customer_id
WHERE c.purchase_count > 10 AND t.category_count = 1;
```

Explanation:

- Counts purchases per customer per category.
- Counts distinct categories per customer.
- Filters customers who bought over 10 items in one category only.
- Helps identify niche or focused shoppers.

Problem 63: Identify Dating App Users Who Sent Messages to a User More Than Twice But Were Never Matched

Problem Statement:

Find users who sent messages to the **same user more than twice** but never formed a match with that user in the last 6 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE messages (
    message_id INT,
    sender_id INT,
    receiver_id INT,
    message_date DATE
);

CREATE TABLE matches (
    match_id INT,
    user_id INT,
    matched_user_id INT,
    match_date DATE
);

INSERT INTO messages VALUES
(1, 401, 501, '2025-01-10'),
(2, 401, 501, '2025-01-15'),
(3, 401, 501, '2025-02-10');

INSERT INTO matches VALUES
(1, 401, 502, '2025-03-01');
```

Solution:

```
WITH multiple_messages AS (
    SELECT sender_id, receiver_id, COUNT(*) AS message_count
    FROM messages
    WHERE message_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
    GROUP BY sender_id, receiver_id
    HAVING message_count > 2
),
matched_pairs AS (
    SELECT user_id, matched_user_id
    FROM matches
    WHERE match_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
)
SELECT m.sender_id, m.receiver_id
FROM multiple_messages m
LEFT JOIN matched_pairs mp ON m.sender_id = mp.user_id AND m.receiver_id =
mp.matched_user_id
WHERE mp.user_id IS NULL;
```

Explanation:

- Finds user pairs with multiple messages.
- Checks if they matched.
- Filters pairs with multiple messages but no match.
- Useful to understand messaging behavior and potential app improvements.

Problem 64: Find Flipkart Customers Who Purchased in All Top 5 Categories by Sales in Last Year

Problem Statement:

Identify customers who made at least one purchase in each of the **top 5 product categories by sales volume** in the last year.

Create & Insert DDL (MySQL):

```
CREATE TABLE flipkart_purchases (
   purchase_id INT,
   customer_id INT,
   customer_name VARCHAR(50),
   product_category VARCHAR(50),
   purchase_date DATE
);

INSERT INTO flipkart_purchases VALUES
(1, 501, 'Gowtham', 'Electronics', '2024-08-10'),
(2, 501, 'Gowtham', 'Books', '2024-09-15'),
(3, 501, 'Gowtham', 'Fashion', '2025-01-20'),
(4, 501, 'Gowtham', 'Home & Kitchen', '2025-03-10'),
(5, 501, 'Gowtham', 'Sports', '2025-04-05');
```

Solution:

```
WITH category_sales AS (
   SELECT product_category, COUNT(*) AS sales_count
```

```
    FROM flipkart_purchases
    WHERE purchase_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
    GROUP BY product_category
    ORDER BY sales_count DESC
    LIMIT 5
),
customer_category_purchases AS (
    SELECT customer_id, customer_name, product_category
    FROM flipkart_purchases
    WHERE purchase_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
    GROUP BY customer_id, customer_name, product_category
),
customer_top_categories AS (
    SELECT c.customer_id, c.customer_name, COUNT(DISTINCT c.product_category) AS
categories_bought
    FROM customer_category_purchases c
    JOIN category_sales s ON c.product_category = s.product_category
    GROUP BY c.customer_id, c.customer_name
),
total_top_categories AS (
    SELECT COUNT(*) AS total_categories FROM category_sales
)
SELECT customer_id, customer_name
FROM customer_top_categories
WHERE categories_bought = (SELECT total_categories FROM total_top_categories);
```

Explanation:

- Identifies top 5 product categories by sales.
- Counts how many of these top categories each customer purchased from.
- Selects customers who purchased from **all** top categories.
- Useful for identifying highly engaged customers with diverse interests.

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

Problem 65: Identify Dating App Users Who Received Messages from More Than 10 Users but Replied to Less Than 3

Problem Statement:

Find users who were messaged by **more than 10 distinct users** but replied to **less than 3** of them in the last 3 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE messages (
    message_id INT,
    sender_id INT,
    receiver_id INT,
    message_date DATE
);

CREATE TABLE replies (
    reply_id INT,
    sender_id INT,
    receiver_id INT,
    reply_date DATE
);

INSERT INTO messages VALUES
(1, 401, 501, '2025-04-01'),
(2, 402, 501, '2025-04-02'),
(3, 403, 501, '2025-04-03'),
(4, 404, 501, '2025-04-04'),
(5, 405, 501, '2025-04-05'),
(6, 406, 501, '2025-04-06'),
(7, 407, 501, '2025-04-07'),
(8, 408, 501, '2025-04-08'),
(9, 409, 501, '2025-04-09'),
(10, 410, 501, '2025-04-10'),
(11, 411, 501, '2025-04-11');

INSERT INTO replies VALUES
(1, 501, 401, '2025-04-12'),
(2, 501, 402, '2025-04-13');
```

Solution:

```
WITH received_messages AS (
    SELECT receiver_id, COUNT(DISTINCT sender_id) AS senders_count
    FROM messages
    WHERE message_date >= DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
    GROUP BY receiver_id
),
reply_counts AS (
    SELECT sender_id, COUNT(DISTINCT receiver_id) AS replied_count
    FROM replies
    WHERE reply_date >= DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
    GROUP BY sender_id
)
SELECT r.receiver_id, r.senders_count, COALESCE(rc.replied_count, 0) AS replied_count
FROM received_messages r
LEFT JOIN reply_counts rc ON r.receiver_id = rc.sender_id
WHERE r.senders_count > 10 AND COALESCE(rc.replied_count, 0) < 3;
```

Explanation:

- Counts distinct senders messaging a user.
- Counts distinct users a user replied to.
- Filters users who received many messages but replied to very few.
- Important to identify less responsive users or possible UX issues.

Problem 66: Find Zomato Restaurants With Consistently Increasing Monthly Revenue for 5 Consecutive Months

Problem Statement:

Identify restaurants on Zomato whose **monthly revenue increased consecutively for 5 months** in the last year.

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

Create & Insert DDL (MySQL):

```
CREATE TABLE zomato_orders (
    order_id INT,
    restaurant_id INT,
    restaurant_name VARCHAR(100),
    order_date DATE,
    order_amount DECIMAL(10,2)
);

INSERT INTO zomato_orders VALUES
(1, 101, 'Spicy Delight', '2024-12-10', 5000.00),
(2, 101, 'Spicy Delight', '2025-01-15', 6000.00),
(3, 101, 'Spicy Delight', '2025-02-10', 7000.00),
(4, 101, 'Spicy Delight', '2025-03-05', 8000.00),
(5, 101, 'Spicy Delight', '2025-04-15', 9000.00),
(6, 102, 'Tandoori Treat', '2024-12-20', 3000.00),
(7, 102, 'Tandoori Treat', '2025-01-25', 2800.00);
```

Solution:

```
WITH monthly_revenue AS (
    SELECT restaurant_id, restaurant_name,
           DATE_FORMAT(order_date, '%Y-%m') AS month,
           SUM(order_amount) AS total_revenue
    FROM zomato_orders
    WHERE order_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
    GROUP BY restaurant_id, restaurant_name, month
),
lagged_revenue AS (
    SELECT restaurant_id, restaurant_name, month, total_revenue,
           LAG(total_revenue) OVER (PARTITION BY restaurant_id ORDER BY month) AS
prev_month_revenue
    FROM monthly_revenue
),
consistent_growth AS (
    SELECT restaurant_id, restaurant_name,
           SUM(CASE WHEN total_revenue > COALESCE(prev_month_revenue, 0) THEN 1 ELSE
0 END) AS growth_months,
           COUNT(*) AS total_months
    FROM lagged_revenue
    GROUP BY restaurant_id, restaurant_name
)
SELECT restaurant_id, restaurant_name
FROM consistent_growth
WHERE growth_months >= 5 AND total_months >= 5;
```

Explanation:

- Calculates monthly revenue per restaurant.
- Uses `LAG()` to compare current month revenue to previous.
- Sums months with revenue growth.
- Filters restaurants with growth in at least 5 months consecutively.
- Useful to identify restaurants with strong upward trends.

Problem 67: Find Flipkart Customers Who Purchased Electronics and Subsequently Bought Warranty Within 30 Days

Problem Statement:

Identify customers who bought an **Electronics product** and purchased a **warranty product** within 30 days after that purchase in the last 6 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE flipkart_purchases (
    purchase_id INT,
    customer_id INT,
    customer_name VARCHAR(50),
    product_category VARCHAR(50),
    purchase_date DATE
);
```

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

```
INSERT INTO flipkart_purchases VALUES
(1, 301, 'Gowtham', 'Electronics', '2025-02-01'),
(2, 301, 'Gowtham', 'Warranty', '2025-02-20'),
(3, 302, 'Anu', 'Electronics', '2025-03-05'),
(4, 302, 'Anu', 'Warranty', '2025-04-10');
```

Solution:

```
WITH electronics_purchases AS (
  SELECT customer_id, customer_name, purchase_date
  FROM flipkart_purchases
  WHERE product_category = 'Electronics'
    AND purchase_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
),
warranty_purchases AS (
  SELECT customer_id, customer_name, purchase_date
  FROM flipkart_purchases
  WHERE product_category = 'Warranty'
    AND purchase_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
)
SELECT DISTINCT e.customer_id, e.customer_name
FROM electronics_purchases e
JOIN warranty_purchases w ON e.customer_id = w.customer_id
WHERE w.purchase_date BETWEEN e.purchase_date AND DATE_ADD(e.purchase_date,
INTERVAL 30 DAY);
```

Explanation:

- Separates Electronics and Warranty purchases.
- Joins on customer, filtering warranty purchases within 30 days post electronics purchase.
- Useful to identify successful warranty upselling.

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

Problem 68: Identify Dating App Users Who Received Messages From Over 10 Users But Replied to Less Than 3 in Last 3 Months

Problem Statement:

Find users who received messages from **more than 10 distinct users** but replied to **fewer than 3** distinct users in the last 3 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE messages (
    message_id INT,
    sender_id INT,
    receiver_id INT,
    message_date DATE
);

CREATE TABLE replies (
    reply_id INT,
    sender_id INT,
    receiver_id INT,
    reply_date DATE
);

INSERT INTO messages VALUES
(1, 401, 501, '2025-04-01'),
(2, 402, 501, '2025-04-02'),
(3, 403, 501, '2025-04-03'),
(4, 404, 501, '2025-04-04'),
(5, 405, 501, '2025-04-05'),
(6, 406, 501, '2025-04-06'),
(7, 407, 501, '2025-04-07'),
(8, 408, 501, '2025-04-08'),
(9, 409, 501, '2025-04-09'),
(10, 410, 501, '2025-04-10'),
(11, 411, 501, '2025-04-11');

INSERT INTO replies VALUES
(1, 501, 401, '2025-04-12'),
(2, 501, 402, '2025-04-13');
```

Solution:

```
WITH received_messages AS (
   SELECT receiver_id, COUNT(DISTINCT sender_id) AS sender_count
   FROM messages
   WHERE message_date >= DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
   GROUP BY receiver_id
),
reply_counts AS (
   SELECT sender_id, COUNT(DISTINCT receiver_id) AS reply_count
   FROM replies
   WHERE reply_date >= DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
   GROUP BY sender_id
)
SELECT r.receiver_id, r.sender_count, COALESCE(rc.reply_count, 0) AS reply_count
FROM received_messages r
LEFT JOIN reply_counts rc ON r.receiver_id = rc.sender_id
WHERE r.sender_count > 10 AND COALESCE(rc.reply_count, 0) < 3;
```

Explanation:

- Counts distinct senders messaging a user.
- Counts distinct users that the user replied to.
- Filters users with many incoming messages but low reply counts.
- Useful for identifying unresponsive or passive users.

Problem 69: Find Flipkart Customers Who Purchased Products Across 5 or More Categories and Returned Less Than 5% of Orders

Problem Statement:

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

Identify customers who bought products in **5 or more categories** and had a **return rate less than 5%** in the last year.

Create & Insert DDL (MySQL):

```
CREATE TABLE purchases (
   purchase_id INT,
   customer_id INT,
   customer_name VARCHAR(50),
   product_category VARCHAR(50),
   order_status VARCHAR(20),
   purchase_date DATE
);

INSERT INTO purchases VALUES
(1, 601, 'Gowtham', 'Electronics', 'Delivered', '2024-08-10'),
(2, 601, 'Gowtham', 'Books', 'Delivered', '2024-09-15'),
(3, 601, 'Gowtham', 'Fashion', 'Delivered', '2025-01-20'),
(4, 601, 'Gowtham', 'Home & Kitchen', 'Delivered', '2025-03-10'),
(5, 601, 'Gowtham', 'Sports', 'Delivered', '2025-04-05'),
(6, 601, 'Gowtham', 'Electronics', 'Returned', '2024-12-10');
```

Solution:

```
WITH category_count AS (
   SELECT customer_id, customer_name, COUNT(DISTINCT product_category) AS
distinct_categories
   FROM purchases
   WHERE purchase_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
   GROUP BY customer_id, customer_name
),
return_stats AS (
   SELECT customer_id,
        COUNT(*) AS total_orders,
        SUM(CASE WHEN order_status = 'Returned' THEN 1 ELSE 0 END) AS returned_orders
   FROM purchases
   WHERE purchase_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
   GROUP BY customer_id
)
SELECT c.customer_id, c.customer_name, c.distinct_categories,
     r.total_orders, r.returned_orders,
     (r.returned_orders / r.total_orders) AS return_rate
```

```
FROM category_count c
JOIN return_stats r ON c.customer_id = r.customer_id
WHERE c.distinct_categories >= 5
  AND (r.returned_orders / r.total_orders) < 0.05;
```

Explanation:

- Counts distinct categories purchased per customer.
- Calculates return rate per customer.
- Filters customers with high category diversity and low returns.
- Identifies loyal and satisfied customers with broad interests.

Problem 70: Find Dating App Users Who Swiped Right on More Than 20 Profiles but Had Matches With Less Than 5 Profiles

Problem Statement:

Identify users who swiped right **on more than 20 profiles** but matched with **fewer than 5** in the last 6 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE swipes (
    swipe_id INT,
    user_id INT,
    swiped_user_id INT,
    swipe_direction VARCHAR(10), -- 'right' or 'left'
    swipe_date DATE
);

CREATE TABLE matches (
    match_id INT,
    user_id INT,
    matched_user_id INT,
    match_date DATE
);

INSERT INTO swipes VALUES
(1, 701, 801, 'right', '2024-12-10'),
(2, 701, 802, 'right', '2025-01-10'),
(3, 701, 803, 'right', '2025-02-15'),
(4, 701, 804, 'right', '2025-03-05'),
(5, 701, 805, 'right', '2025-03-10'),
(6, 701, 806, 'right', '2025-03-15'),
(7, 701, 807, 'right', '2025-03-20'),
(8, 701, 808, 'right', '2025-04-05'),
(9, 701, 809, 'right', '2025-04-10'),
(10, 701, 810, 'right', '2025-04-15'),
(11, 701, 811, 'right', '2025-04-20'),
(12, 701, 812, 'right', '2025-04-25'),
(13, 701, 813, 'right', '2025-04-30'),
(14, 701, 814, 'right', '2025-05-05'),
(15, 701, 815, 'right', '2025-05-10'),
(16, 701, 816, 'right', '2025-05-15'),
(17, 701, 817, 'right', '2025-05-20'),
(18, 701, 818, 'right', '2025-05-25'),
(19, 701, 819, 'right', '2025-05-30'),
(20, 701, 820, 'right', '2025-06-01'),
(21, 701, 821, 'right', '2025-06-05');

INSERT INTO matches VALUES
(1, 701, 801, '2025-02-01'),
(2, 701, 802, '2025-02-15'),
(3, 701, 803, '2025-03-01'),
(4, 701, 804, '2025-03-15');
```

Solution:

```
WITH swipe_counts AS (
    SELECT user_id, COUNT(*) AS right_swipes
    FROM swipes
    WHERE swipe_direction = 'right'
      AND swipe_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
    GROUP BY user_id
),
match_counts AS (
    SELECT user_id, COUNT(*) AS total_matches
    FROM matches
    WHERE match_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
    GROUP BY user_id
)
SELECT s.user_id, s.right_swipes, COALESCE(m.total_matches, 0) AS total_matches
FROM swipe_counts s
LEFT JOIN match_counts m ON s.user_id = m.user_id
WHERE s.right_swipes > 20
  AND COALESCE(m.total_matches, 0) < 5;
```

Explanation:

- Counts total right swipes per user.
- Counts total matches per user.
- Filters users who swiped right frequently but matched rarely.
- Helps identify potential algorithm or engagement issues.

Problem 71: Identify Zomato Customers Who Placed Orders on Weekends But Never on Weekdays in Last 6 Months

Problem Statement:

Find customers who placed **orders only on weekends (Saturday and Sunday)** and never on weekdays in the last 6 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE zomato_orders (
    order_id INT,
    customer_id INT,
    customer_name VARCHAR(50),
    order_date DATE
);

INSERT INTO zomato_orders VALUES
(1, 1001, 'Gowtham', '2025-01-05'), -- Sunday
(2, 1001, 'Gowtham', '2025-01-11'), -- Saturday
(3, 1001, 'Gowtham', '2025-02-02'), -- Sunday
(4, 1002, 'Anu', '2025-01-06'),     -- Monday
(5, 1002, 'Anu', '2025-01-12');    -- Sunday
```

Solution:

```
WITH order_days AS (
    SELECT customer_id, customer_name,
           MAX(CASE WHEN DAYOFWEEK(order_date) IN (2,3,4,5,6) THEN 1 ELSE 0 END) AS
weekday_order,
           MAX(CASE WHEN DAYOFWEEK(order_date) IN (1,7) THEN 1 ELSE 0 END) AS
weekend_order
    FROM zomato_orders
    WHERE order_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
    GROUP BY customer_id, customer_name
)
SELECT customer_id, customer_name
FROM order_days
WHERE weekend_order = 1 AND weekday_order = 0;
```

Explanation:

- `DAYOFWEEK()` returns day number (1=Sunday, 7=Saturday).
- `weekday_order` flags if customer ordered on any weekday.
- `weekend_order` flags if customer ordered on weekend.
- Filters customers who ordered only on weekends.
- Useful for targeted weekend promotions.

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

Problem 72: Find Flipkart Customers Who Frequently Bought Products But Had Increasing Return Rates Every Quarter

Problem Statement:

Identify customers who made **at least 10 purchases each quarter** but whose **return rates increased quarter over quarter** in the last year.

Create & Insert DDL (MySQL):

```
CREATE TABLE flipkart_orders (
   order_id INT,
   customer_id INT,
   customer_name VARCHAR(50),
   order_status VARCHAR(20),
   order_date DATE
);

INSERT INTO flipkart_orders VALUES
(1, 201, 'Gowtham', 'Delivered', '2024-10-10'),
(2, 201, 'Gowtham', 'Returned', '2024-10-15'),
(3, 201, 'Gowtham', 'Delivered', '2025-01-10'),
(4, 201, 'Gowtham', 'Returned', '2025-01-15'),
(5, 201, 'Gowtham', 'Returned', '2025-04-05'),
(6, 201, 'Gowtham', 'Delivered', '2025-04-10');
```

Solution:

```
WITH quarterly_stats AS (
   SELECT customer_id, customer_name,
        CONCAT(YEAR(order_date), '-Q', QUARTER(order_date)) AS quarter,
        COUNT(*) AS total_orders,
        SUM(CASE WHEN order_status = 'Returned' THEN 1 ELSE 0 END) AS
returned_orders,
        SUM(CASE WHEN order_status = 'Returned' THEN 1 ELSE 0 END) / COUNT(*) AS
return_rate
   FROM flipkart_orders
   WHERE order_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
   GROUP BY customer_id, customer_name, quarter
),
lagged_rates AS (
   SELECT customer_id, customer_name, quarter, return_rate,
          LAG(return_rate) OVER (PARTITION BY customer_id ORDER BY quarter) AS
prev_return_rate,
          total_orders
   FROM quarterly_stats
),
increasing_returns AS (
   SELECT customer_id, customer_name,
          SUM(CASE WHEN return_rate > COALESCE(prev_return_rate, 0) AND total_orders >=
10 THEN 1 ELSE 0 END) AS quarters_with_increase,
          COUNT(*) AS total_quarters
   FROM lagged_rates
   GROUP BY customer_id, customer_name
)
SELECT customer_id, customer_name
FROM increasing_returns
WHERE quarters_with_increase = total_quarters AND total_quarters >= 3;
```

Explanation:

- Computes quarterly total and returned orders per customer.
- Uses `LAG()` to compare return rates quarter-over-quarter.
- Checks if return rate increased in all quarters where customer ordered >=10 times.
- Identifies customers with worsening return behavior.

Problem 73: Find Dating App Users Who Matched But Never Sent a Message in Last 3 Months

Problem Statement:

Identify users who formed matches but **never initiated a message** to their matches in the last 3 months.

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

Create & Insert DDL (MySQL):

```
CREATE TABLE matches (
    match_id INT,
    user_id INT,
    matched_user_id INT,
    match_date DATE
);

CREATE TABLE messages (
    message_id INT,
    sender_id INT,
    receiver_id INT,
    message_date DATE
);

INSERT INTO matches VALUES
(1, 501, 601, '2025-02-01'),
(2, 501, 602, '2025-02-15');

INSERT INTO messages VALUES
(1, 601, 501, '2025-02-20');
```

Solution:

```
WITH matched_pairs AS (
    SELECT user_id, matched_user_id
    FROM matches
    WHERE match_date >= DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
),
sent_messages AS (
    SELECT DISTINCT sender_id, receiver_id
    FROM messages
    WHERE message_date >= DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
)
SELECT m.user_id
FROM matched_pairs m
LEFT JOIN sent_messages s ON m.user_id = s.sender_id AND m.matched_user_id =
s.receiver_id
GROUP BY m.user_id
HAVING COUNT(m.matched_user_id) > 0 AND COUNT(s.receiver_id) = 0;
```

Explanation:

- Lists matched user pairs.
- Identifies who sent messages to whom.
- Finds users who never sent a message to any of their matches.
- Helps improve engagement and prompt users to start conversations.

Problem 74: Find Flipkart Customers Who Purchased Same Product Multiple Times But Reviewed Only Once

Problem Statement:

Identify customers who purchased the **same product multiple times** but submitted **only one review** for it in the last year.

Create & Insert DDL (MySQL):

```
CREATE TABLE purchases (
   purchase_id INT,
   customer_id INT,
   customer_name VARCHAR(50),
   product_id INT,
   purchase_date DATE
);

CREATE TABLE reviews (
   review_id INT,
   customer_id INT,
   product_id INT,
   review_date DATE
);

INSERT INTO purchases VALUES
```

```
(1, 701, 'Gowtham', 1001, '2024-08-10'),
(2, 701, 'Gowtham', 1001, '2024-10-10'),
(3, 701, 'Gowtham', 1001, '2025-01-15');

INSERT INTO reviews VALUES
(1, 701, 1001, '2024-10-12');
```

Solution:

```
WITH purchase_counts AS (
    SELECT customer_id, product_id, COUNT(*) AS purchase_count
    FROM purchases
    WHERE purchase_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
    GROUP BY customer_id, product_id
),
review_counts AS (
    SELECT customer_id, product_id, COUNT(*) AS review_count
    FROM reviews
    WHERE review_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
    GROUP BY customer_id, product_id
)
SELECT p.customer_id, p.product_id, p.purchase_count, COALESCE(r.review_count, 0) AS
review_count
FROM purchase_counts p
LEFT JOIN review_counts r ON p.customer_id = r.customer_id AND p.product_id = r.product_id
WHERE p.purchase_count > 1 AND COALESCE(r.review_count, 0) = 1;
```

Explanation:

- Aggregates purchase and review counts per customer-product.
- Filters customers with multiple purchases but only one review.
- Useful to identify potential under-reviewing users.

Problem 75: Find Dating App Users Who Received More Than 20 Likes But Matched With Less Than 5 Profiles

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

Problem Statement:

Identify users who were **liked by more than 20 different users** but matched with **less than 5 profiles** in the last 6 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE likes (
   like_id INT,
   liked_user_id INT,
   liker_id INT,
   like_date DATE
);

CREATE TABLE matches (
   match_id INT,
   user_id INT,
   matched_user_id INT,
   match_date DATE
);

INSERT INTO likes VALUES
(1, 801, 901, '2025-01-10'),
(2, 801, 902, '2025-02-15'),
(3, 801, 903, '2025-03-20'),
(4, 801, 904, '2025-04-25'),
(5, 801, 905, '2025-05-10'),
(6, 801, 906, '2025-05-20'),
(7, 801, 907, '2025-06-01'),
(8, 801, 908, '2025-06-05'),
(9, 801, 909, '2025-06-10'),
(10, 801, 910, '2025-06-15'),
(11, 801, 911, '2025-06-20'),
(12, 801, 912, '2025-06-25'),
(13, 801, 913, '2025-06-30'),
(14, 801, 914, '2025-07-05'),
(15, 801, 915, '2025-07-10'),
(16, 801, 916, '2025-07-15'),
(17, 801, 917, '2025-07-20'),
(18, 801, 918, '2025-07-25'),
(19, 801, 919, '2025-07-30'),
(20, 801, 920, '2025-08-01'),
(21, 801, 921, '2025-08-05');
```

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

```
INSERT INTO matches VALUES
(1, 801, 901, '2025-02-01'),
(2, 801, 902, '2025-02-15'),
(3, 801, 903, '2025-03-01'),
(4, 801, 904, '2025-03-15');
```

Solution:

```
WITH like_counts AS (
   SELECT liked_user_id, COUNT(DISTINCT liker_id) AS total_likes
   FROM likes
   WHERE like_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
   GROUP BY liked_user_id
),
match_counts AS (
   SELECT user_id, COUNT(DISTINCT matched_user_id) AS total_matches
   FROM matches
   WHERE match_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
   GROUP BY user_id
)
SELECT l.liked_user_id, l.total_likes, COALESCE(m.total_matches, 0) AS total_matches
FROM like_counts l
LEFT JOIN match_counts m ON l.liked_user_id = m.user_id
WHERE l.total_likes > 20 AND COALESCE(m.total_matches, 0) < 5;
```

Explanation:

- Counts distinct likers per user.
- Counts matches per user.
- Filters users with high likes but low matches.
- Helps evaluate matching algorithm effectiveness.

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

Problem 76: Find Zomato Customers Who Increased Their Average Order Value by 20% Month-over-Month for 3 Consecutive Months

Problem Statement:

Identify customers on Zomato who have increased their **average order value by at least 20% month-over-month for 3 consecutive months** in the last 6 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE zomato_orders (
    order_id INT,
    customer_id INT,
    customer_name VARCHAR(50),
    order_date DATE,
    order_amount DECIMAL(10,2)
);

INSERT INTO zomato_orders VALUES
(1, 1001, 'Gowtham', '2025-01-10', 500.00),
(2, 1001, 'Gowtham', '2025-02-10', 650.00),
(3, 1001, 'Gowtham', '2025-03-10', 800.00),
(4, 1002, 'Anu', '2025-01-15', 700.00),
(5, 1002, 'Anu', '2025-02-15', 600.00),
(6, 1002, 'Anu', '2025-03-15', 500.00);
```

Solution:

```
WITH monthly_avg AS (
    SELECT customer_id, customer_name,
           DATE_FORMAT(order_date, '%Y-%m') AS month,
           AVG(order_amount) AS avg_order_value
    FROM zomato_orders
    WHERE order_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
    GROUP BY customer_id, customer_name, month
),
lagged AS (
    SELECT customer_id, customer_name, month, avg_order_value,
        LAG(avg_order_value) OVER (PARTITION BY customer_id ORDER BY month) AS
prev_avg_order_value,
        LAG(avg_order_value, 2) OVER (PARTITION BY customer_id ORDER BY month) AS
prev2_avg_order_value
   FROM monthly_avg
)
SELECT customer_id, customer_name
FROM lagged
WHERE prev2_avg_order_value IS NOT NULL
  AND avg_order_value >= 1.2 * prev_avg_order_value
  AND prev_avg_order_value >= 1.2 * prev2_avg_order_value;
```

Explanation:

- Calculate average monthly order value per customer.
- Use `LAG()` to get previous two months' average order values.
- Filter customers whose avg order value increased by at least 20% consecutively in last 3 months.
- Useful for identifying growing high-value customers.

Problem 77: Identify Flipkart Customers Who Purchased High-Value Products but Used No Offers in Last 6 Months

Problem Statement:

Find customers who purchased products priced above ₹10,000 **at least 5 times** but used **no offers** on these purchases in the last 6 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE purchases (
   purchase_id INT,
   customer_id INT,
   customer_name VARCHAR(50),
    product_price DECIMAL(10,2),
    used_offer BOOLEAN,
    purchase_date DATE
);

INSERT INTO purchases VALUES
(1, 501, 'Gowtham', 15000.00, FALSE, '2025-02-10'),
(2, 501, 'Gowtham', 12000.00, FALSE, '2025-03-15'),
(3, 501, 'Gowtham', 13000.00, FALSE, '2025-04-20'),
(4, 502, 'Anu', 11000.00, TRUE, '2025-03-05');
```

Solution:

```
WITH high_value_no_offer AS (
    SELECT customer_id, customer_name, COUNT(*) AS purchase_count
    FROM purchases
    WHERE product_price > 10000
      AND used_offer = FALSE
      AND purchase_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
    GROUP BY customer_id, customer_name
)
SELECT customer_id, customer_name, purchase_count
FROM high_value_no_offer
WHERE purchase_count >= 5;
```

Explanation:

- Filters high-value purchases with no offer usage.
- Aggregates count per customer.
- Identifies customers purchasing premium products without offers.
- Useful for personalized upselling strategies.

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

Problem 78: Find Dating App Users Who Sent Messages to More Than 5 Matches but Received Replies from Only 1

Problem Statement:

Identify users who messaged **more than 5 matched users** but received replies from **only one** of them in the last 3 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE messages (
   message_id INT,
   sender_id INT,
   receiver_id INT,
   message_date DATE
);

CREATE TABLE replies (
   reply_id INT,
   sender_id INT,
   receiver_id INT,
   reply_date DATE
);

CREATE TABLE matches (
   match_id INT,
   user_id INT,
   matched_user_id INT,
   match_date DATE
);

INSERT INTO messages VALUES
(1, 401, 501, '2025-04-01'),
(2, 401, 502, '2025-04-02'),
(3, 401, 503, '2025-04-03'),
(4, 401, 504, '2025-04-04'),
(5, 401, 505, '2025-04-05'),
(6, 401, 506, '2025-04-06');

INSERT INTO replies VALUES
```

```
(1, 501, 401, '2025-04-07');
```

Solution:

```
WITH sent_messages AS (
   SELECT sender_id, receiver_id
   FROM messages
   WHERE message_date >= DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
),
reply_counts AS (
   SELECT sender_id, COUNT(DISTINCT receiver_id) AS replied_count
   FROM replies
   WHERE reply_date >= DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
   GROUP BY sender_id
),
message_counts AS (
   SELECT sender_id, COUNT(DISTINCT receiver_id) AS sent_count
   FROM sent_messages
   GROUP BY sender_id
)
SELECT m.sender_id
FROM message_counts m
LEFT JOIN reply_counts r ON m.sender_id = r.sender_id
WHERE m.sent_count > 5 AND COALESCE(r.replied_count, 0) = 1;
```

Explanation:

- Counts distinct receivers messaged and replied to per user.
- Filters users with many sent messages but very few replies.
- Helps identify one-sided conversations or engagement challenges.

Problem 79: Find Flipkart Customers Who Purchased Products in Last 3 Months but Never Logged Into App

Problem Statement:

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

Identify customers who made purchases in the last 3 months but have **not logged into the Flipkart app** during the same period.

Create & Insert DDL (MySQL):

```
CREATE TABLE purchases (
   purchase_id INT,
   customer_id INT,
   purchase_date DATE
);

CREATE TABLE app_logins (
   login_id INT,
   customer_id INT,
   login_date DATE
);

INSERT INTO purchases VALUES
(1, 601, '2025-05-01'),
(2, 602, '2025-06-01');

INSERT INTO app_logins VALUES
(1, 602, '2025-06-05');
```

Solution:

```
WITH recent_purchases AS (
   SELECT DISTINCT customer_id
   FROM purchases
   WHERE purchase_date >= DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
),
recent_logins AS (
   SELECT DISTINCT customer_id
   FROM app_logins
   WHERE login_date >= DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
)
SELECT p.customer_id
FROM recent_purchases p
LEFT JOIN recent_logins l ON p.customer_id = l.customer_id
WHERE l.customer_id IS NULL;
```

Explanation:

- Lists customers who purchased recently.
- Lists customers who logged in recently.
- Filters customers who purchased but did not log in.
- Useful for identifying users completing purchases via other channels or guest checkout.

Problem 80: Find Dating App Users Who Swiped Right on Over 30 Profiles But Matched With Less Than 5

Problem Statement:

Identify users who swiped right on **over 30 profiles** but matched with **less than 5** in the last 6 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE swipes (
   swipe_id INT,
   user_id INT,
   swiped_user_id INT,
   swipe_direction VARCHAR(10),
   swipe_date DATE
);

CREATE TABLE matches (
   match_id INT,
   user_id INT,
   matched_user_id INT,
   match_date DATE
);

INSERT INTO swipes VALUES
(1, 701, 801, 'right', '2024-12-10'),
(2, 701, 802, 'right', '2025-01-10'),
-- ... assume many more right swipes ...
```

```
(31, 701, 831, 'right', '2025-06-01');

INSERT INTO matches VALUES
(1, 701, 801, '2025-02-01'),
(2, 701, 802, '2025-02-15'),
(3, 701, 803, '2025-03-01'),
(4, 701, 804, '2025-03-15');
```

Solution:

```
WITH swipe_counts AS (
   SELECT user_id, COUNT(*) AS right_swipes
   FROM swipes
   WHERE swipe_direction = 'right'
     AND swipe_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
   GROUP BY user_id
),
match_counts AS (
   SELECT user_id, COUNT(*) AS total_matches
   FROM matches
   WHERE match_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
   GROUP BY user_id
)
SELECT s.user_id, s.right_swipes, COALESCE(m.total_matches, 0) AS total_matches
FROM swipe_counts s
LEFT JOIN match_counts m ON s.user_id = m.user_id
WHERE s.right_swipes > 30
  AND COALESCE(m.total_matches, 0) < 5;
```

Explanation:

- Aggregates right swipes and matches per user.
- Filters users with high swiping activity but low matches.
- Important for evaluating user experience and matching algorithm performance.

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

Problem 81: Find Zomato Customers Who Increased Their Order Frequency But Decreased Average Tip Amount Month-over-Month for 3 Months

Problem Statement:

Identify customers who placed **more orders each month** but whose **average tip amount decreased month-over-month** consecutively for 3 months in the last 6 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE zomato_orders (
   order_id INT,
   customer_id INT,
   customer_name VARCHAR(50),
   order_date DATE,
   tip_amount DECIMAL(6,2)
);

INSERT INTO zomato_orders VALUES
(1, 1001, 'Gowtham', '2025-01-05', 50.00),
(2, 1001, 'Gowtham', '2025-01-15', 40.00),
(3, 1001, 'Gowtham', '2025-02-10', 30.00),
(4, 1001, 'Gowtham', '2025-02-20', 25.00),
(5, 1001, 'Gowtham', '2025-03-05', 20.00),
(6, 1001, 'Gowtham', '2025-03-15', 15.00);
```

Solution:

```
WITH monthly_stats AS (
   SELECT customer_id, customer_name,
        DATE_FORMAT(order_date, '%Y-%m') AS order_month,
        COUNT(*) AS order_count,
        AVG(tip_amount) AS avg_tip
   FROM zomato_orders
   WHERE order_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
   GROUP BY customer_id, customer_name, order_month
),
lagged_stats AS (
   SELECT customer_id, customer_name, order_month, order_count, avg_tip,
        LAG(order_count, 1) OVER (PARTITION BY customer_id ORDER BY order_month) AS
prev_order_count,
        LAG(avg_tip, 1) OVER (PARTITION BY customer_id ORDER BY order_month) AS
prev_avg_tip,
        LAG(order_count, 2) OVER (PARTITION BY customer_id ORDER BY order_month) AS
prev2_order_count,
        LAG(avg_tip, 2) OVER (PARTITION BY customer_id ORDER BY order_month) AS
prev2_avg_tip
   FROM monthly_stats
)
SELECT customer_id, customer_name
FROM lagged_stats
WHERE prev2_order_count IS NOT NULL
  AND order_count > prev_order_count
  AND prev_order_count > prev2_order_count
  AND avg_tip < prev_avg_tip
  AND prev_avg_tip < prev2_avg_tip;
```

Explanation:

- Computes monthly order counts and average tip per customer.
- Uses `LAG()` to get previous two months' stats.
- Filters customers with strictly increasing order count but decreasing average tip for 3 consecutive months.
- Indicates customers ordering more but tipping less, useful for loyalty and incentive programs.

Problem 82: Find Flipkart Customers Who Purchased Electronics and Fashion in the Same Order But Different Payment Methods

Problem Statement:

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

Identify customers who bought **both Electronics and Fashion products** in the same order but paid with **different payment methods** for each category in the last 6 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE flipkart_order_items (
   order_id INT,
   customer_id INT,
   customer_name VARCHAR(50),
   product_category VARCHAR(50),
   payment_method VARCHAR(50),
   order_date DATE
);

INSERT INTO flipkart_order_items VALUES
(1001, 301, 'Gowtham', 'Electronics', 'Credit Card', '2025-01-10'),
(1001, 301, 'Gowtham', 'Fashion', 'UPI', '2025-01-10'),
(1002, 302, 'Anu', 'Electronics', 'Debit Card', '2025-02-15'),
(1002, 302, 'Anu', 'Fashion', 'Debit Card', '2025-02-15');
```

Solution:

```
WITH order_payments AS (
   SELECT order_id, customer_id, customer_name,
          GROUP_CONCAT(DISTINCT product_category) AS categories,
          COUNT(DISTINCT payment_method) AS payment_methods
   FROM flipkart_order_items
   WHERE order_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
   GROUP BY order_id, customer_id, customer_name
)
SELECT order_id, customer_id, customer_name
FROM order_payments
WHERE categories LIKE '%Electronics%'
  AND categories LIKE '%Fashion%'
  AND payment_methods > 1;
```

Explanation:

- Groups order items by order and customer.

- Checks if order contains both Electronics and Fashion.
- Counts distinct payment methods used.
- Filters orders where multiple payment methods were used.
- Useful for understanding split payments or gift purchases.

Problem 83: Identify Dating App Users Who Swiped Right on Over 15 Profiles But Matched with Less Than 3 Profiles in Last 6 Months

Problem Statement:

Find users who swiped right **on more than 15 profiles** but matched with **fewer than 3** in the last 6 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE swipes (
   swipe_id INT,
   user_id INT,
   swiped_user_id INT,
   swipe_direction VARCHAR(10),
   swipe_date DATE
);

CREATE TABLE matches (
   match_id INT,
   user_id INT,
   matched_user_id INT,
   match_date DATE
);

INSERT INTO swipes VALUES
(1, 701, 801, 'right', '2024-12-01'),
(2, 701, 802, 'right', '2025-01-01'),
-- Add more swipes to exceed 15 for user 701
```

```
(16, 701, 816, 'right', '2025-05-01');

INSERT INTO matches VALUES
(1, 701, 801, '2025-01-15'),
(2, 701, 802, '2025-01-20');
```

Solution:

```
WITH swipe_counts AS (
   SELECT user_id, COUNT(*) AS right_swipes
   FROM swipes
   WHERE swipe_direction = 'right'
     AND swipe_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
   GROUP BY user_id
),
match_counts AS (
   SELECT user_id, COUNT(*) AS total_matches
   FROM matches
   WHERE match_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
   GROUP BY user_id
)
SELECT s.user_id, s.right_swipes, COALESCE(m.total_matches, 0) AS total_matches
FROM swipe_counts s
LEFT JOIN match_counts m ON s.user_id = m.user_id
WHERE s.right_swipes > 15
  AND COALESCE(m.total_matches, 0) < 3;
```

Explanation:

- Counts right swipes and matches per user.
- Filters users with high swiping but few matches.
- Indicates potential gaps in matching algorithms or user preferences.

Problem 84: Find Flipkart Customers Who Purchased Products Using Multiple Payment Methods in Last Month

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

Problem Statement:

Identify customers who made purchases using **more than one payment method** in the last month.

Create & Insert DDL (MySQL):

```
CREATE TABLE purchases (
    purchase_id INT,
    customer_id INT,
    customer_name VARCHAR(50),
    payment_method VARCHAR(50),
    purchase_date DATE
);

INSERT INTO purchases VALUES
(1, 801, 'Gowtham', 'Credit Card', '2025-06-01'),
(2, 801, 'Gowtham', 'UPI', '2025-06-15'),
(3, 802, 'Anu', 'Debit Card', '2025-06-10');
```

Solution:

```
SELECT customer_id, customer_name, COUNT(DISTINCT payment_method) AS
payment_methods_used
FROM purchases
WHERE purchase_date >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH)
GROUP BY customer_id, customer_name
HAVING payment_methods_used > 1;
```

Explanation:

- Counts distinct payment methods per customer.
- Filters customers using multiple payment methods recently.
- Helps understand payment preferences and multi-method use.

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

Problem 85: Identify Dating App Users Who Received Messages from More Than 10 Users But Never Replied

Problem Statement:

Find users who received messages from **more than 10 distinct users** but never replied to any in the last 3 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE messages (
    message_id INT,
    sender_id INT,
    receiver_id INT,
    message_date DATE
);

CREATE TABLE replies (
    reply_id INT,
    sender_id INT,
    receiver_id INT,
    reply_date DATE
);

INSERT INTO messages VALUES
(1, 401, 501, '2025-04-01'),
(2, 402, 501, '2025-04-02'),
(3, 403, 501, '2025-04-03'),
(4, 404, 501, '2025-04-04'),
(5, 405, 501, '2025-04-05'),
(6, 406, 501, '2025-04-06'),
(7, 407, 501, '2025-04-07'),
(8, 408, 501, '2025-04-08'),
(9, 409, 501, '2025-04-09'),
(10, 410, 501, '2025-04-10'),
(11, 411, 501, '2025-04-11');

INSERT INTO replies VALUES ();
-- No replies from user 501
```

Solution:

```
WITH received_messages AS (
   SELECT receiver_id, COUNT(DISTINCT sender_id) AS sender_count
   FROM messages
   WHERE message_date >= DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
   GROUP BY receiver_id
),
reply_counts AS (
   SELECT sender_id, COUNT(*) AS reply_count
   FROM replies
   WHERE reply_date >= DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
   GROUP BY sender_id
)
SELECT r.receiver_id, r.sender_count
FROM received_messages r
LEFT JOIN reply_counts rc ON r.receiver_id = rc.sender_id
WHERE r.sender_count > 10 AND (rc.reply_count IS NULL OR rc.reply_count = 0);
```

Explanation:

- Counts distinct senders who messaged a user.
- Counts replies made by the user.
- Filters users with many incoming messages but zero replies.
- Important for identifying inactive or unresponsive users.

Problem 86: Identify Zomato Customers Who Frequently Ordered High-Value Items But Gave Low Ratings

Problem Statement:

Find customers who ordered items with **order amount greater than ₹1000** more than **5 times** in the last 6 months but gave an **average rating below 3**.

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

Create & Insert DDL (MySQL):

```
CREATE TABLE zomato_orders (
   order_id INT,
   customer_id INT,
   customer_name VARCHAR(50),
   order_amount DECIMAL(10,2),
   rating INT,
   order_date DATE
);

INSERT INTO zomato_orders VALUES
(1, 1001, 'Gowtham', 1200.00, 2, '2025-01-10'),
(2, 1001, 'Gowtham', 1500.00, 3, '2025-02-15'),
(3, 1001, 'Gowtham', 1300.00, 2, '2025-03-20'),
(4, 1002, 'Anu', 1100.00, 4, '2025-01-25');
```

Solution:

```
WITH high_value_orders AS (
   SELECT customer_id, customer_name,
        COUNT(*) AS high_value_order_count,
        AVG(rating) AS avg_rating
   FROM zomato_orders
   WHERE order_amount > 1000
     AND order_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
   GROUP BY customer_id, customer_name
)
SELECT customer_id, customer_name, high_value_order_count, avg_rating
FROM high_value_orders
WHERE high_value_order_count > 5 AND avg_rating < 3;
```

Explanation:

- Filters orders with amount > ₹1000 within 6 months.
- Aggregates count and average rating per customer.
- Filters customers ordering frequently expensive items but rating low.

- Useful for identifying dissatisfied high-spenders for service improvement.

Problem 87: Find Flipkart Customers Who Purchased Electronics Products Across Multiple Cities

Problem Statement:

Identify customers who bought **Electronics products** from **more than 2 distinct cities** in the last year.

Create & Insert DDL (MySQL):

```
CREATE TABLE flipkart_purchases (
    purchase_id INT,
    customer_id INT,
    customer_name VARCHAR(50),
    city VARCHAR(50),
    product_category VARCHAR(50),
    purchase_date DATE
);

INSERT INTO flipkart_purchases VALUES
(1, 301, 'Gowtham', 'Chennai', 'Electronics', '2024-08-10'),
(2, 301, 'Gowtham', 'Bangalore', 'Electronics', '2024-10-15'),
(3, 301, 'Gowtham', 'Hyderabad', 'Electronics', '2025-01-20'),
(4, 302, 'Anu', 'Chennai', 'Electronics', '2025-02-10');
```

Solution:

```
SELECT customer_id, customer_name, COUNT(DISTINCT city) AS city_count
FROM flipkart_purchases
WHERE product_category = 'Electronics'
  AND purchase_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
GROUP BY customer_id, customer_name
HAVING city_count > 2;
```

Explanation:

- Filters electronics purchases within 1 year.
- Counts distinct cities per customer.
- Filters customers purchasing electronics in multiple cities.
- Useful for understanding customer mobility or multi-location purchasing.

Problem 88: Identify Dating App Users Who Swiped Left More Than 50 Times but Matched with More Than 10 Users

Problem Statement:

Find users who swiped left **more than 50 times** but matched with **more than 10 users** in the last 6 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE swipes (
   swipe_id INT,
   user_id INT,
   swiped_user_id INT,
   swipe_direction VARCHAR(10),
   swipe_date DATE
);

CREATE TABLE matches (
   match_id INT,
   user_id INT,
   matched_user_id INT,
   match_date DATE
);

INSERT INTO swipes VALUES
(1, 701, 801, 'left', '2025-01-10'),
(2, 701, 802, 'left', '2025-01-15'),
-- assume more than 50 left swipes
(51, 701, 850, 'left', '2025-06-01');

INSERT INTO matches VALUES
(1, 701, 801, '2025-02-01'),
(2, 701, 802, '2025-02-10'),
(3, 701, 803, '2025-02-15'),
-- assume more than 10 matches
(11, 701, 811, '2025-05-15');
```

Solution:

```
WITH left_swipe_counts AS (
  SELECT user_id, COUNT(*) AS left_swipes
  FROM swipes
  WHERE swipe_direction = 'left'
    AND swipe_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
  GROUP BY user_id
),
match_counts AS (
  SELECT user_id, COUNT(*) AS total_matches
  FROM matches
  WHERE match_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
  GROUP BY user_id
)
SELECT l.user_id, l.left_swipes, COALESCE(m.total_matches, 0) AS total_matches
FROM left_swipe_counts l
LEFT JOIN match_counts m ON l.user_id = m.user_id
WHERE l.left_swipes > 50 AND COALESCE(m.total_matches, 0) > 10;
```

Explanation:

- Counts left swipes and matches per user.
- Filters users with high left swipes but also high matches.
- Useful for understanding user selectivity and matching success.

Problem 89: Find Flipkart Customers Who Used Multiple Payment Methods for a Single Order

Problem Statement:

Identify customers who used **more than one payment method** for the same order in the last 6 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE flipkart_order_payments (
    order_id INT,
    customer_id INT,
    customer_name VARCHAR(50),
    payment_method VARCHAR(50),
    payment_amount DECIMAL(10,2),
    payment_date DATE
);

INSERT INTO flipkart_order_payments VALUES
(1001, 501, 'Gowtham', 'Credit Card', 1500.00, '2025-02-10'),
(1001, 501, 'Gowtham', 'Wallet', 500.00, '2025-02-10'),
(1002, 502, 'Anu', 'Debit Card', 2000.00, '2025-03-05');
```

Solution:

```
SELECT order_id, customer_id, customer_name, COUNT(DISTINCT payment_method) AS
payment_methods_count
FROM flipkart_order_payments
WHERE payment_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
GROUP BY order_id, customer_id, customer_name
HAVING payment_methods_count > 1;
```

Explanation:

- Groups payments by order and customer.

- Counts distinct payment methods per order.
- Filters orders paid using multiple methods.
- Useful for analyzing complex payment behaviors and splits.

Problem 90: Identify Dating App Users Who Matched With More Than 5 Users but Messaged Less Than 2

Problem Statement:

Find users who matched with **more than 5 users** but sent messages to **less than 2** of them in the last 3 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE matches (
    match_id INT,
    user_id INT,
    matched_user_id INT,
    match_date DATE
);

CREATE TABLE messages (
    message_id INT,
    sender_id INT,
    receiver_id INT,
    message_date DATE
);

INSERT INTO matches VALUES
(1, 501, 601, '2025-03-01'),
(2, 501, 602, '2025-03-05'),
(3, 501, 603, '2025-03-10'),
(4, 501, 604, '2025-03-15'),
(5, 501, 605, '2025-03-20'),
(6, 501, 606, '2025-03-25');

INSERT INTO messages VALUES
```

```
(1, 501, 601, '2025-03-10');
```

Solution:

```
WITH match_counts AS (
   SELECT user_id, COUNT(DISTINCT matched_user_id) AS total_matches
   FROM matches
   WHERE match_date >= DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
   GROUP BY user_id
),
message_counts AS (
   SELECT sender_id, COUNT(DISTINCT receiver_id) AS messages_sent
   FROM messages
   WHERE message_date >= DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
   GROUP BY sender_id
)
SELECT m.user_id, m.total_matches, COALESCE(msg.messages_sent, 0) AS messages_sent
FROM match_counts m
LEFT JOIN message_counts msg ON m.user_id = msg.sender_id
WHERE m.total_matches > 5 AND COALESCE(msg.messages_sent, 0) < 2;
```

Explanation:

- Counts distinct matches and messages per user.
- Filters users with many matches but minimal messaging activity.
- Helps identify engagement gaps for product improvement.

Problem 91: Identify Zomato Restaurants with High Order Cancellation Rate but Increasing New Customer Count

Problem Statement:

Find restaurants on Zomato with **order cancellation rate above 10%** in the last 6 months but whose **new customer count increased month-over-month** consecutively for 3 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE zomato_orders (
    order_id INT,
    restaurant_id INT,
    restaurant_name VARCHAR(100),
    customer_id INT,
    order_status VARCHAR(20),
    order_date DATE
);

INSERT INTO zomato_orders VALUES
(1, 101, 'Spicy Delight', 1001, 'Cancelled', '2025-01-10'),
(2, 101, 'Spicy Delight', 1002, 'Delivered', '2025-01-15'),
(3, 101, 'Spicy Delight', 1003, 'Delivered', '2025-02-10'),
(4, 101, 'Spicy Delight', 1004, 'Cancelled', '2025-02-20'),
(5, 101, 'Spicy Delight', 1005, 'Delivered', '2025-03-05'),
(6, 101, 'Spicy Delight', 1006, 'Delivered', '2025-03-15');
```

Solution:

```
WITH monthly_stats AS (
    SELECT restaurant_id, restaurant_name,
           DATE_FORMAT(order_date, '%Y-%m') AS month,
           COUNT(*) AS total_orders,
           SUM(CASE WHEN order_status = 'Cancelled' THEN 1 ELSE 0 END) AS
cancelled_orders,
           COUNT(DISTINCT CASE WHEN order_date = DATE(order_date) THEN customer_id
END) AS unique_customers
    FROM zomato_orders
    WHERE order_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
    GROUP BY restaurant_id, restaurant_name, month
),
cancellation_rate AS (
    SELECT restaurant_id, restaurant_name, month,
           cancelled_orders / total_orders AS cancel_rate,
           unique_customers
    FROM monthly_stats
),
lagged_customers AS (
    SELECT restaurant_id, restaurant_name, month, cancel_rate, unique_customers,
           LAG(unique_customers) OVER (PARTITION BY restaurant_id ORDER BY month) AS
prev_customers,
           LAG(unique_customers, 2) OVER (PARTITION BY restaurant_id ORDER BY month) AS
prev2_customers
    FROM cancellation_rate
)
SELECT restaurant_id, restaurant_name
FROM lagged_customers
WHERE cancel_rate > 0.10
  AND prev2_customers IS NOT NULL
  AND unique_customers > prev_customers
  AND prev_customers > prev2_customers;
```

Explanation:

- Calculates monthly total and cancelled orders and unique customers per restaurant.
- Computes cancellation rate.
- Uses `LAG()` to check if unique customers increased for 3 consecutive months.
- Filters restaurants with high cancellation but growing new customer base.
- Helps identify potential operational issues alongside growth.

Problem 92: Find Flipkart Customers Who Bought High-End Electronics but Used Low Credit Limits

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

Problem Statement:

Identify customers who bought electronics products priced above ₹50,000 but used credit cards with **credit limits less than ₹1,00,000** in the last 6 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE flipkart_purchases (
    purchase_id INT,
    customer_id INT,
    customer_name VARCHAR(50),
    product_category VARCHAR(50),
    product_price DECIMAL(10,2),
    payment_method VARCHAR(50),
    credit_limit DECIMAL(10,2),
    purchase_date DATE
);

INSERT INTO flipkart_purchases VALUES
(1, 501, 'Gowtham', 'Electronics', 60000.00, 'Credit Card', 80000.00, '2025-01-10'),
(2, 502, 'Anu', 'Electronics', 70000.00, 'Credit Card', 120000.00, '2025-02-15');
```

Solution:

```
SELECT customer_id, customer_name, product_price, credit_limit
FROM flipkart_purchases
WHERE product_category = 'Electronics'
  AND product_price > 50000
  AND payment_method = 'Credit Card'
  AND credit_limit < 100000
  AND purchase_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH);
```

Explanation:

- Filters electronics purchases with price above ₹50,000.
- Checks payment method is credit card with credit limits below ₹1,00,000.
- Helps identify customers making high-value purchases on limited credit.

- Useful for credit risk analysis or personalized credit offers.

Problem 93: Identify Dating App Users Who Matched But Had No Message Exchanges in Last 3 Months

Problem Statement:

Find users who formed matches but have **zero message exchanges** with their matches in the last 3 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE matches (
   match_id INT,
   user_id INT,
   matched_user_id INT,
   match_date DATE
);

CREATE TABLE messages (
   message_id INT,
   sender_id INT,
   receiver_id INT,
   message_date DATE
);

INSERT INTO matches VALUES
(1, 501, 601, '2025-02-01'),
(2, 501, 602, '2025-02-15');

INSERT INTO messages VALUES ();
-- No messages exchanged
```

Solution:

```
WITH matched_pairs AS (
   SELECT user_id, matched_user_id
```

```
    FROM matches
    WHERE match_date >= DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
),
message_pairs AS (
    SELECT DISTINCT sender_id, receiver_id
    FROM messages
    WHERE message_date >= DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
)
SELECT m.user_id
FROM matched_pairs m
LEFT JOIN message_pairs msg ON (m.user_id = msg.sender_id AND m.matched_user_id =
msg.receiver_id)
                        OR (m.user_id = msg.receiver_id AND m.matched_user_id = msg.sender_id)
GROUP BY m.user_id
HAVING COUNT(msg.sender_id) = 0;
```

Explanation:

- Lists matched pairs.
- Checks if messages were exchanged in either direction.
- Filters users with no message exchanges.
- Helps identify inactive matched users for engagement improvements.

Problem 94: Find Flipkart Customers Who Purchased Multiple Categories but Returned Only From One Category

Problem Statement:

Identify customers who bought products from **multiple categories** but made returns from **only one specific category** in the last year.

Create & Insert DDL (MySQL):

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

```
CREATE TABLE purchases (
    purchase_id INT,
    customer_id INT,
    customer_name VARCHAR(50),
    product_category VARCHAR(50),
    order_status VARCHAR(20),
    purchase_date DATE
);

INSERT INTO purchases VALUES
(1, 701, 'Gowtham', 'Electronics', 'Delivered', '2024-08-10'),
(2, 701, 'Gowtham', 'Books', 'Delivered', '2024-09-15'),
(3, 701, 'Gowtham', 'Electronics', 'Returned', '2024-10-10'),
(4, 701, 'Gowtham', 'Books', 'Delivered', '2025-01-20');
```

Solution:

```
WITH category_purchases AS (
    SELECT customer_id, product_category, COUNT(*) AS purchase_count
    FROM purchases
    WHERE purchase_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
    GROUP BY customer_id, product_category
),
category_returns AS (
    SELECT customer_id, product_category, COUNT(*) AS return_count
    FROM purchases
    WHERE purchase_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
      AND order_status = 'Returned'
    GROUP BY customer_id, product_category
),
customer_categories AS (
    SELECT customer_id, COUNT(DISTINCT product_category) AS distinct_categories
    FROM category_purchases
    GROUP BY customer_id
),
return_categories AS (
    SELECT customer_id, COUNT(DISTINCT product_category) AS distinct_return_categories
    FROM category_returns
    GROUP BY customer_id
)
SELECT cp.customer_id
FROM customer_categories cp
JOIN return_categories rc ON cp.customer_id = rc.customer_id
```

```
WHERE cp.distinct_categories > 1 AND rc.distinct_return_categories = 1;
```

Explanation:

- Counts purchased and returned categories per customer.
- Filters customers buying multiple categories but returning only from one.
- Helps identify problematic product categories.

Problem 95: Identify Dating App Users Who Received Messages from More Than 15 Users But Sent Messages to Less Than 5

Problem Statement:

Find users who were messaged by **more than 15 distinct users** but sent messages to **fewer than 5** distinct users in the last 3 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE messages (
  message_id INT,
  sender_id INT,
  receiver_id INT,
  message_date DATE
);

INSERT INTO messages VALUES
(1, 401, 501, '2025-04-01'),
(2, 402, 501, '2025-04-02'),
-- assume more than 15 messages to 501
(20, 420, 501, '2025-04-20');

INSERT INTO messages VALUES
(21, 501, 401, '2025-04-15'),
(22, 501, 402, '2025-04-16');
```

Solution:

```
WITH received_messages AS (
    SELECT receiver_id, COUNT(DISTINCT sender_id) AS sender_count
    FROM messages
    WHERE message_date >= DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
    GROUP BY receiver_id
),
sent_messages AS (
    SELECT sender_id, COUNT(DISTINCT receiver_id) AS receiver_count
    FROM messages
    WHERE message_date >= DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
    GROUP BY sender_id
)
SELECT r.receiver_id
FROM received_messages r
LEFT JOIN sent_messages s ON r.receiver_id = s.sender_id
WHERE r.sender_count > 15
  AND (s.receiver_count IS NULL OR s.receiver_count < 5);
```

Explanation:

- Counts distinct users messaging a user and distinct users messaged by that user.
- Filters users with many incoming messages but few outgoing.
- Important for engagement and retention strategies.

Problem 96: Identify Zomato Customers Who Ordered More Than 3 Times from Different Restaurants but Rated Only One Restaurant

Problem Statement:

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

Find customers who placed orders from **more than 3 distinct restaurants** but gave ratings to **only one restaurant** in the last 6 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE zomato_orders (
    order_id INT,
    customer_id INT,
    customer_name VARCHAR(50),
    restaurant_id INT,
    restaurant_name VARCHAR(100),
    order_date DATE
);

CREATE TABLE zomato_ratings (
    rating_id INT,
    customer_id INT,
    restaurant_id INT,
    rating INT,
    rating_date DATE
);

INSERT INTO zomato_orders VALUES
(1, 1001, 'Gowtham', 201, 'Spicy Delight', '2025-01-05'),
(2, 1001, 'Gowtham', 202, 'Tandoori Treat', '2025-02-10'),
(3, 1001, 'Gowtham', 203, 'Green Garden', '2025-03-15'),
(4, 1001, 'Gowtham', 204, 'Curry House', '2025-04-20');

INSERT INTO zomato_ratings VALUES
(1, 1001, 201, 4, '2025-01-10');
```

Solution:

```
WITH order_counts AS (
    SELECT customer_id, customer_name, COUNT(DISTINCT restaurant_id) AS
restaurant_order_count
    FROM zomato_orders
    WHERE order_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
    GROUP BY customer_id, customer_name
),
rating_counts AS (
    SELECT customer_id, COUNT(DISTINCT restaurant_id) AS restaurant_rating_count
    FROM zomato_ratings
    WHERE rating_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
    GROUP BY customer_id
)
SELECT o.customer_id, o.customer_name, o.restaurant_order_count,
COALESCE(r.restaurant_rating_count, 0) AS restaurant_rating_count
FROM order_counts o
LEFT JOIN rating_counts r ON o.customer_id = r.customer_id
WHERE o.restaurant_order_count > 3 AND COALESCE(r.restaurant_rating_count, 0) = 1;
```

Explanation:

- Counts distinct restaurants ordered and rated per customer.
- Filters customers ordering from many restaurants but rating only one.
- Important for encouraging broader feedback and reviews.

Problem 97: Find Flipkart Customers Who Used Cash on Delivery but Converted to Prepaid in Last 3 Months

Problem Statement:

Identify customers who used **Cash on Delivery (COD)** for purchases before 3 months ago but switched to **prepaid payments** in the last 3 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE flipkart_purchases (
    purchase_id INT,
    customer_id INT,
    customer_name VARCHAR(50),
    payment_method VARCHAR(50),
    purchase_date DATE
);

INSERT INTO flipkart_purchases VALUES
```

```
(1, 301, 'Gowtham', 'COD', '2024-12-15'),
(2, 301, 'Gowtham', 'Prepaid', '2025-05-01'),
(3, 302, 'Anu', 'COD', '2025-04-01');
```

Solution:

```
WITH cod_users AS (
  SELECT DISTINCT customer_id, customer_name
  FROM flipkart_purchases
  WHERE payment_method = 'COD'
    AND purchase_date < DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
),
prepaid_users AS (
  SELECT DISTINCT customer_id, customer_name
  FROM flipkart_purchases
  WHERE payment_method = 'Prepaid'
    AND purchase_date >= DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
)
SELECT c.customer_id, c.customer_name
FROM cod_users c
JOIN prepaid_users p ON c.customer_id = p.customer_id;
```

Explanation:

- Lists customers who used COD before 3 months.
- Lists customers who used prepaid recently.
- Finds customers who switched from COD to prepaid.
- Useful for payment behavior analysis and targeting incentives.

Problem 98: Identify Dating App Users Who Received More Than 20 Likes But Matched With Less Than 5 Profiles

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

Problem Statement:

Find users who received **more than 20 likes** but matched with **less than 5 profiles** in the last 6 months.

Create & Insert DDL (MySQL):

```
CREATE TABLE likes (
    like_id INT,
    liked_user_id INT,
    liker_id INT,
    like_date DATE
);

CREATE TABLE matches (
    match_id INT,
    user_id INT,
    matched_user_id INT,
    match_date DATE
);

INSERT INTO likes VALUES
(1, 801, 901, '2025-01-10'),
(2, 801, 902, '2025-02-15'),
(3, 801, 903, '2025-03-20'),
-- Assume more likes to exceed 20
(21, 801, 921, '2025-06-01');

INSERT INTO matches VALUES
(1, 801, 901, '2025-02-01'),
(2, 801, 902, '2025-02-15'),
(3, 801, 903, '2025-03-01'),
(4, 801, 904, '2025-03-15');
```

Solution:

```
WITH like_counts AS (
    SELECT liked_user_id, COUNT(DISTINCT liker_id) AS total_likes
    FROM likes
    WHERE like_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
    GROUP BY liked_user_id
),
match_counts AS (
    SELECT user_id, COUNT(DISTINCT matched_user_id) AS total_matches
    FROM matches
    WHERE match_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
    GROUP BY user_id
)
SELECT l.liked_user_id, l.total_likes, COALESCE(m.total_matches, 0) AS total_matches
FROM like_counts l
LEFT JOIN match_counts m ON l.liked_user_id = m.user_id
WHERE l.total_likes > 20 AND COALESCE(m.total_matches, 0) < 5;
```

Explanation:

- Counts distinct likes and matches per user.
- Filters users with many likes but few matches.
- Useful for understanding gaps in matchmaking algorithms.

Problem 99: Find Flipkart Customers Who Made Frequent Purchases but Used Same Payment Method Always

Problem Statement:

Identify customers who made **more than 10 purchases** in the last 6 months but used **only one payment method** for all orders.

Create & Insert DDL (MySQL):

```
CREATE TABLE flipkart_purchases (
    purchase_id INT,
    customer_id INT,
    customer_name VARCHAR(50),
    payment_method VARCHAR(50),
    purchase_date DATE
);
```

```
INSERT INTO flipkart_purchases VALUES
(1, 901, 'Gowtham', 'Credit Card', '2025-01-10'),
(2, 901, 'Gowtham', 'Credit Card', '2025-02-15'),
(3, 901, 'Gowtham', 'Credit Card', '2025-03-20'),
(4, 902, 'Anu', 'UPI', '2025-01-05'),
(5, 902, 'Anu', 'Debit Card', '2025-02-10');
```

Solution:

```
WITH purchase_counts AS (
  SELECT customer_id, customer_name, COUNT(*) AS total_purchases, COUNT(DISTINCT
payment_method) AS distinct_payment_methods
  FROM flipkart_purchases
  WHERE purchase_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
  GROUP BY customer_id, customer_name
)
SELECT customer_id, customer_name, total_purchases
FROM purchase_counts
WHERE total_purchases > 10 AND distinct_payment_methods = 1;
```

Explanation:

- Aggregates purchase counts and distinct payment methods per customer.
- Filters customers with frequent purchases but a single payment method.
- Helps understand payment loyalty or behavior patterns.

Problem 100: Identify Dating App Users Who Sent Messages to More Than 10 Matches But Received Replies From Less Than 3

Problem Statement:

Find users who messaged **more than 10 matched users** but received replies from **less than 3** in the last 3 months.

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in

Create & Insert DDL (MySQL):

```
CREATE TABLE messages (
    message_id INT,
    sender_id INT,
    receiver_id INT,
    message_date DATE
);

CREATE TABLE replies (
    reply_id INT,
    sender_id INT,
    receiver_id INT,
    reply_date DATE
);

CREATE TABLE matches (
    match_id INT,
    user_id INT,
    matched_user_id INT,
    match_date DATE
);

INSERT INTO messages VALUES
(1, 501, 601, '2025-04-01'),
(2, 501, 602, '2025-04-02'),
(3, 501, 603, '2025-04-03'),
-- Assume more messages sent

INSERT INTO replies VALUES
(1, 601, 501, '2025-04-10'),
(2, 602, 501, '2025-04-12');
```

Solution:

```
WITH message_counts AS (
    SELECT sender_id, COUNT(DISTINCT receiver_id) AS messages_sent
    FROM messages
    WHERE message_date >= DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
    GROUP BY sender_id
),
reply_counts AS (
   SELECT receiver_id, COUNT(DISTINCT sender_id) AS replies_received
   FROM replies
   WHERE reply_date >= DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
   GROUP BY receiver_id
)
SELECT m.sender_id, m.messages_sent, COALESCE(r.replies_received, 0) AS
replies_received
FROM message_counts m
LEFT JOIN reply_counts r ON m.sender_id = r.receiver_id
WHERE m.messages_sent > 10 AND COALESCE(r.replies_received, 0) < 3;
```

Explanation:

- Counts distinct matches messaged and replies received.
- Filters users with many outgoing messages but few replies.
- Helps identify engagement and response rate issues.

Congratulations! By working through these 100 practical SQL problems inspired by real-world scenarios from platforms like Zomato, Flipkart, and dating apps, you've sharpened your data querying skills and developed a deeper understanding of complex business use cases. These challenges simulate the type of problems faced by data analysts and engineers at top tech companies

Remember, mastering SQL is not just about writing queries but also about thinking critically about data patterns, optimizing performance, and solving problems efficiently. Keep practicing, exploring new scenarios, and applying these concepts in your projects and interviews.

With consistent effort and curiosity, you'll be well-prepared to excel in technical interviews and real-world data challenges. Best wishes on your data engineering journey!

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @thedatatech.in
— Gowtham

👤 About the Author: Gowtham SB – Data Engineer, Educator, Creator

Gowtham is one of India's most loved **Tamil tech content creators** — known for explaining **Data Engineering, System Design, and AI** using **local analogies** that stick.

- 💻 10+ years of experience in Big Data & Cloud
- 🎥 Creator of the YouTube channel: **Data Engineering**
- 📚 Author of multiple bestselling books on data & AI

Gowtham SB
www.linkedin.com/in/sbgowtham/   Instagram - @thedatatech.in

- 🧠 Featured in IBM's **Top Data Engineering Influencers** list

- 🌍 Over **2 lakh learners** across YouTube, LinkedIn, and workshops

He believes tech should be taught like a conversation at a tea kada — warm, real, and unforgettable.

You can connect with him on:

- YouTube: Data Engineering
- Instagram: @thedatatech.in
- LinkedIn: https://www.linkedin.com/in/sbgowtham/
- My Books - https://topmate.io/dataengineering

Thank You