-- Flipkart Data Analysis using SQL
-- Solutions of 20 business problems

---

1 Understand how many unique users are active on the platform each day to track daily engagement trends (DAU – Daily Active Users).

```
SELECT activity_date, COUNT(DISTINCT user_id) AS daily_active_users
FROM user_activity
GROUP BY activity_date
ORDER BY activity_date;
```

2 Identify how many distinct users use the platform each month to analyze user base growth and retention (MAU – Monthly Active Users).

```
SELECT DATE_TRUNC('month', activity_date) AS month, COUNT(DISTINCT user_id) AS monthly_active_users
FROM user_activity
GROUP BY month
ORDER BY month;
```

3 Monitor how many new users are signing up on the platform to evaluate acquisition performance (New Signups).

```
SELECT activity_date, COUNT(DISTINCT user_id) AS new_signups
FROM user_activity
WHERE is_new_user = TRUE
GROUP BY activity_date
ORDER BY activity_date;
```

4. Measure the month-over-month growth or decline in the user base to adjust growth strategies effectively (MoM – Month-over-Month Growth).

```
WITH monthly_users AS (
        SELECT DATE_TRUNC('month', activity_date) AS month,
COUNT(DISTINCT user_id) AS user_count
  FROM user_activity
  GROUP BY month
)
SELECT month, user_count,
    LAG(user_count) OVER (ORDER BY month) AS prev_month_users,
     ROUND(((user_count - LAG(user_count) OVER (ORDER BY month))
* 100.0) / NULLIF(LAG(user_count) OVER (ORDER BY month), 0), 2) AS
mom_growth_percent
FROM monthly_users;
```

5. Analyze which traffic sources (e.g., Organic, Ads, Referral) bring the most users to optimize marketing efforts (Traffic Source Analysis).

```
SELECT source, COUNT(DISTINCT user_id) AS users
FROM user_activity
GROUP BY source
ORDER BY users DESC;
```

6. Understand whether users prefer using the app or the web version to improve UX and prioritize platform-specific features (Device Preference – App vs Web).

```
SELECT platform, COUNT(DISTINCT user_id) AS users
FROM user_activity
GROUP BY platform;
```

7.Identify how many users are visiting for the first time versus returning users to understand retention and loyalty (New vs Returning Users).

```
SELECT is_new_user, COUNT(DISTINCT user_id) AS users
FROM user_activity
```

```
GROUP BY is_new_user;
```

8. Determine how many users signed up through referrals to assess the effectiveness of the referral program (Referral Signups).

```
SELECT COUNT(DISTINCT user_id) AS referral_signups
FROM user_activity
WHERE referral_user_id IS NOT NULL;
```

9.Analyze the user drop-off rate at each stage of the funnel (View → Cart → Purchase) to optimize conversion (Conversion Funnel Analysis).

```
SELECT
    COUNT(DISTINCT CASE WHEN page_viewed IS NOT NULL THEN user_id END) AS viewed,
      COUNT(DISTINCT CASE WHEN added_to_cart = TRUE THEN user_id END) AS added_to_cart,
    COUNT(DISTINCT CASE WHEN purchased = TRUE THEN user_id END) AS purchased
FROM user_activity;
```

10. Measure how much time users spend on the platform on average to evaluate engagement levels (Average Session Duration).

```
SELECT AVG(session_duration_seconds) AS avg_session_duration_seconds
FROM user_activity;
```

11.Understand how many pages users view in a single session to measure content or product catalog effectiveness (Pages per Session).

```
SELECT session_id, COUNT(page_viewed) AS pages_per_session
```

```
FROM user_activity
GROUP BY session_id
ORDER BY pages_per_session DESC;
```

12. Analyze how frequently each user returns to the platform by counting sessions per user (Session Frequency per User).

```
SELECT user_id, COUNT(DISTINCT session_id) AS session_count
FROM user_activity
GROUP BY user_id
ORDER BY session_count DESC;
```

13. Determine how many sessions result in just one page view to measure bounce rate and identify UX issues (Bounce Rate).

```
SELECT COUNT(*) FILTER (WHERE session_id IN (
      SELECT session_id FROM user_activity GROUP BY session_id
HAVING COUNT(*) = 1
)) * 100.0 / COUNT(*) AS bounce_rate_percent
FROM user_activity;
```

14.Track the percentage of users who click on any element after landing on a page to understand UI performance (Click-Through Rate – CTR).

```
SELECT COUNT(*) FILTER (WHERE click_event = TRUE) * 100.0 /
COUNT(*) AS click_through_rate_percent
FROM user_activity;
```

15. Calculate the proportion of users who added items to the cart but did not purchase to assess conversion leakage (Cart Abandonment Rate).

```
SELECT
```

```
    COUNT(DISTINCT CASE WHEN added_to_cart = TRUE THEN user_id END) AS added_to_cart,
    COUNT(DISTINCT CASE WHEN purchased = TRUE THEN user_id END) AS purchased,
    (COUNT(DISTINCT CASE WHEN added_to_cart = TRUE THEN user_id END) - COUNT(DISTINCT CASE WHEN purchased = TRUE THEN user_id END)) * 100.0 / NULLIF(COUNT(DISTINCT CASE WHEN added_to_cart = TRUE THEN user_id END), 0) AS cart_abandonment_rate
FROM user_activity;
```

16. Measure the success rate of items added to the cart that led to a purchase to evaluate checkout performance (Cart-to-Purchase Rate).

```
SELECT
  COUNT(*) FILTER (WHERE added_to_cart = TRUE AND purchased = TRUE) * 100.0 / NULLIF(COUNT(*) FILTER (WHERE added_to_cart = TRUE), 0) AS cart_to_purchase_rate_percent
FROM user_activity;
```

17. Track how often users use coupons and how that translates into successful purchases to assess promotional impact (Coupon Conversion Rate).

```
SELECT
  COUNT(*) FILTER (WHERE coupon_used = TRUE AND purchased = TRUE) * 100.0 / NULLIF(COUNT(*) FILTER (WHERE coupon_used = TRUE), 0) AS coupon_conversion_rate
FROM user_activity;
```

18. Analyze how many users make repeat purchases to evaluate loyalty and customer retention (Repeat Purchase Rate – RPR).

```
SELECT COUNT(*) * 100.0 / NULLIF((SELECT COUNT(DISTINCT user_id) FROM user_activity WHERE purchased = TRUE), 0) AS repeat_purchase_rate_percent
FROM (
  SELECT user_id
  FROM user_activity
  WHERE purchased = TRUE
  GROUP BY user_id
  HAVING COUNT(*) > 1
) AS repeat_users;
```

19. Calculate the total value of all completed purchases to measure gross merchandise value and overall sales performance (GMV – Gross Merchandise Value).

```
SELECT SUM(order_value) AS gross_merchandise_value
FROM user_activity
WHERE purchased = TRUE;
```

20. Determine the total revenue remaining after returns to understand true earnings and profitability (Net Revenue).

```
SELECT SUM(order_value) FILTER (WHERE purchased = TRUE AND returned = FALSE) AS net_revenue
FROM user_activity;
```
