SQL QUERIES:

Query 1: Table Creation

Defined six essential tables to support attribution, revenue tracking, and user engagement analysis:

- **`costs`:** Captures daily ad spend across different media channels and operating systems.
- **`installs_google`, `installs_rest`, and `installs_organic`:** Track install events across distinct acquisition sources.
- **`mapping`:** Links anonymous visitor identifiers to unique user IDs.
- **`revenues`:** Stores daily revenue generated by each user.

```
-- created costs table
CREATE TABLE costs (
  date DATE,
  media_channel TEXT,
  operating_system TEXT,
  spend NUMERIC
);
-- created installs_google table
CREATE TABLE installs_google (
  install_time TIMESTAMP,
  media_channel TEXT,
  operating_system TEXT,
  visitor_id TEXT,
  attr_id TEXT
);
-- created installs_organic table
CREATE TABLE installs_organic (
  install_time TIMESTAMP,
  operating_system TEXT,
  visitor_id TEXT
);
-- created installs_rest table
CREATE TABLE installs_rest (
  install_time TIMESTAMP,
  media_channel TEXT,
  operating_system TEXT,
  visitor_id TEXT
);
-- created mapping table
CREATE TABLE mapping (
  id TEXT, -- user_id
  vst_id TEXT, -- visitor_id
  createdat TIMESTAMP
);
-- created revenues table
```

```sql
CREATE TABLE revenues (
  period DATE,
  user_id TEXT,
  revenue NUMERIC
);
```

Query 2: Unified Install Dataset

Consolidated install records from all three sources using **`UNION ALL`** to create a comprehensive install **log (`installs_all`)**. This ensures all events are preserved for downstream analysis, including duplicates.

```sql
-- used UNION ALL to add the tables even if we found the duplicates
CREATE TABLE installs_all AS
SELECT
  install_time::date AS install_date,
  media_channel,
  visitor_id
FROM installs_google
UNION ALL
SELECT
  install_time::date,
  NULL AS media_channel,
  visitor_id
FROM installs_organic
UNION ALL
SELECT
  install_time::date,
  media_channel,
  visitor_id
FROM installs_rest;
```

Query 3: Attribution Modeling

Constructed an `attribution_modeling` table to compute weekly Customer Acquisition Cost (CAC) and Return on Ad Spend (ROAS):

- Merged install and revenue data using visitor-to-user mappings.
- Aggregated weekly spend by media channel from the `costs` table.
- Joined installs and spend data to calculate performance metrics per channel and week.

```sql
CREATE TABLE attribution_modeling AS
WITH installs AS (
  SELECT
    i.install_date,
```

```sql
    DATE_TRUNC('week', i.install_date) AS week,
    COALESCE(i.media_channel, 'organic') AS media_source,
    i.visitor_id,
    m.id AS user_id
  FROM installs_all i
  LEFT JOIN mapping m ON i.visitor_id = m.vst_id
),
installs_with_revenue AS (
  SELECT
    inst.week,
    inst.media_source,
    inst.visitor_id,
    r.revenue::numeric
  FROM installs inst
  LEFT JOIN revenues r ON inst.user_id = r.user_id
),
weekly_costs AS (
  SELECT
    DATE_TRUNC('week', date) AS week,
    media_channel AS media_source,
    SUM(spend)::numeric AS spend
  FROM costs
  GROUP BY 1, 2
),
weekly_metrics AS (
  SELECT
    week,
    media_source,
    COUNT(DISTINCT visitor_id) AS installs,
    SUM(revenue) AS revenue
  FROM installs_with_revenue
  GROUP BY 1, 2
)
SELECT
  m.week,
  m.media_source,
  m.installs,
  m.revenue,
  c.spend,
  ROUND((c.spend / NULLIF(m.installs, 0))::numeric, 2) AS CAC,
  ROUND((m.revenue / NULLIF(c.spend, 0))::numeric, 4) AS ROAS
FROM weekly_metrics m
LEFT JOIN weekly_costs c
 ON m.week = c.week AND m.media_source = c.media_source;
```

Query 4 : User Lifetime Value (LTV) Segmentation

Performed LTV analysis and segmented users based on value tiers:

- 4A: Created a view **`user_ltv`** to compute total revenue (LTV) per user.
- 4B: Developed a materialized view **`ltv_segmented_users`** to classify users into **`High`**, **`Medium`**, or **`Low`** LTV tiers using percentile thresholds. Each user was assigned to their latest media source.
- 4C: Created a summary view **`ltv_distribution_by_source`** to report the number of users in each LTV tier across media channels.

```
-- Step 4A: Total revenue (LTV) per user

CREATE OR REPLACE VIEW user_ltv AS
SELECT
  user_id,
  SUM(revenue) AS ltv
FROM revenues
GROUP BY user_id;


-- STEP 4B: USER LTV SEGMENTATION BY MEDIA SOURCE
CREATE MATERIALIZED VIEW ltv_segmented_users AS
WITH all_installs AS (
  SELECT visitor_id, install_time, media_channel
  FROM installs_google
  UNION ALL
  SELECT visitor_id, install_time, NULL AS media_channel
  FROM installs_organic
  UNION ALL
  SELECT visitor_id, install_time, media_channel
  FROM installs_rest
),
latest_installs AS (
  SELECT
     visitor_id,
     MAX(install_time) AS latest_time
  FROM all_installs
  GROUP BY visitor_id
),
latest_user_installs AS (
  SELECT
     a.visitor_id,
     a.install_time,
     COALESCE(a.media_channel, 'organic') AS media_source
  FROM all_installs a
  JOIN latest_installs l
     ON a.visitor_id = l.visitor_id AND a.install_time = l.latest_time
```

```sql
),
mapped_users AS (
  SELECT
    m.id AS user_id,
    i.media_source
  FROM mapping m
  JOIN latest_user_installs i
    ON m.vst_id = i.visitor_id
)
SELECT
  l.user_id,
  l.ltv,
  CASE
    WHEN l.ltv >= (
      SELECT PERCENTILE_CONT(0.66)
      WITHIN GROUP (ORDER BY ltv)
      FROM user_ltv
    ) THEN 'High'
    WHEN l.ltv >= (
      SELECT PERCENTILE_CONT(0.33)
      WITHIN GROUP (ORDER BY ltv)
      FROM user_ltv
    ) THEN 'Medium'
    ELSE 'Low'
  END AS ltv_tier,
  mu.media_source
FROM user_ltv l
JOIN mapped_users mu
  ON l.user_id = mu.user_id;

-- Step 4C: Count users in each LTV tier by media source
CREATE OR REPLACE VIEW ltv_distribution_by_source AS
SELECT
  ltv_tier,
  media_source,
  COUNT(*) AS user_count
FROM ltv_segmented_users
GROUP BY ltv_tier, media_source
ORDER BY ltv_tier, media_source;
```

Query 5: Step 5: Re-Engagement Effectiveness

Analyzed user re-engagement and its impact on LTV:

- **5A**: Built a materialized view **`reengagements`** to detect users who reinstalled the app after more than 30 days of inactivity.
- **5B**: Summarized re-engagement activity in **`reengagement_summary`** by media source, including re-engagement rates and user counts.
- **5C**: Created the **`reengaged_user_ltv`** view to assess LTV outcomes specifically for re-engaged users and quantified total re-engagements.

```
-- STEP 5A: RE-ENGAGEMENT ANALYSIS
CREATE MATERIALIZED VIEW reengagements AS
WITH combined_installs AS (
  SELECT
    m.id AS user_id,
    i.install_date::timestamp AS install_time,
    COALESCE(i.media_channel, 'organic') AS media_source
  FROM installs_all i
  JOIN mapping m ON i.visitor_id = m.vst_id
)
SELECT
  user_id,
  install_time,
  media_source,
  LAG(install_time) OVER (PARTITION BY user_id ORDER BY install_time) AS previous_install,
  CASE
    WHEN install_time - LAG(install_time) OVER (PARTITION BY user_id ORDER BY install_time)
> INTERVAL '30 days'
    THEN TRUE ELSE FALSE
  END AS is_reengagement
FROM combined_installs;
```

```
-- STEP 5B RE-ENGAGEMENT SUMMARY BY MEDIA SOURCE
CREATE OR REPLACE VIEW reengagement_summary AS
SELECT
  media_source,
  COUNT(*) FILTER (WHERE is_reengagement) AS reengaged_users,
  COUNT(*) AS total_installs,
  COUNT(DISTINCT user_id) AS unique_users,
  ROUND(
    COUNT(*) FILTER (WHERE is_reengagement)::NUMERIC / NULLIF(COUNT(*), 0),
    2
  ) AS reengagement_rate
FROM reengagements
```

```sql
GROUP BY media_source
ORDER BY media_source;

-- STEP 5C: POST-REENGAGEMENT LTV COMPARISON
CREATE OR REPLACE VIEW reengaged_user_ltv AS
SELECT
  r.user_id,
  r.media_source,
  l.ltv
FROM reengagements r
JOIN user_ltv l ON r.user_id = l.user_id
WHERE is_reengagement = TRUE;
SELECT COUNT(DISTINCT user_id) AS total_reengaged_users
FROM reengagements
WHERE is_reengagement = TRUE;
```