# Data Validation

## I. Duplicated rows

To understand the scale and structure of the raw event data, I first checked the number of events, users, and sessions in the staging table:

```
SELECT
COUNT(*)AS event_rows,
COUNT(DISTINCT user_pseudo_id)AS users,
COUNT(DISTINCT session_key)AS sessions
FROM `funnel-analysis-482106.analytics.events_flat`;
```

Next, I verified whether the session-level mart contained duplicate sessions by comparing total rows to distinct session keys:

```
SELECT
COUNT(*)AS rows_no,
COUNT(DISTINCT session_key)AS distinct_sessions
FROM `funnel-analysis-482106.analytics.sessions_funnel`;
```

To identify duplicated sessions, I ran:

```
SELECT
  session_key,
COUNT(*)AS rows_per_session
FROM `funnel-analysis-482106.analytics.sessions_funnel`
GROUPBY session_key
HAVINGCOUNT(*)>1;
```

This analysis revealed **268 duplicated sessions**. Further inspection showed that the duplicated records had different event dates, which happens when a single GA4 session spans multiple calendar days (for example, overnight sessions).

To inspect these duplicated sessions in more detail, I joined the duplicated session keys back to the session table:

```
WITH duplicated_sessionsAS (
SELECT
   session_key,
COUNT(*)AS rows_per_session
FROM `funnel-analysis-482106.analytics.sessions_funnel`
GROUPBY session_key
HAVINGCOUNT(*)>1
)
SELECT*
FROM `funnel-analysis-482106.analytics.sessions_funnel`AS main_table
INNERJOIN duplicated_sessions
ON main_table.session_key= duplicated_sessions.session_key
ORDERBY main_table.session_key;
```

To resolve the issue, duplicated sessions were merged by aggregating all rows for the same `session_key`. Funnel step flags were combined using `MAX`, and revenue was summed at the session level. Before applying this logic, I verified that summing revenue would not result in double counting. The check confirmed that revenue never appeared in more than one row per duplicated session:

```
SELECT
COUNT(*)AS duplicated_sessions,
  COUNTIF(revenue_sum> revenue_max)AS sessions_where_sum_exceeds
_max
FROM (
SELECT
   session_key,
SUM(revenue)AS revenue_sum,
MAX(revenue)AS revenue_max
FROM `funnel-analysis-482106.analytics.sessions_funnel`
GROUPBY session_key
HAVINGCOUNT(*)>1
```

```
);
```

Since `revenue_sum` never exceeded `revenue_max`, it was safe to aggregate revenue using `SUM` when deduplicating sessions.

| Row | duplicated_sessio... | sessions_where_... |
|---|---|---|
| 1 | 268 | 0 |

After deduplication, I validated that funnel steps were coherent at the session level (sessions with purchase should not systematically exceed sessions reaching earlier steps).

```
SELECT
SUM(has_view_item)AS view_sessions,
SUM(has_add_to_cart)AS cart_sessions,
SUM(has_begin_checkout)AS checkout_sessions,
SUM(has_purchase)AS purchase_sessions
FROM `funnel-analysis-482106.analytics.sessions_funnel_deduped`;
```

I spot-checked a sample of duplicated sessions to confirm that acquisition fields (source/medium/campaign) were consistent across the duplicated rows before relying on `ANY_VALUE`."

## II. GA4 didn't capture all the add to cart events

The initial funnel exploration revealed an unexpected pattern: the number of `begin_checkout` and `purchase` events was significantly higher than the number of `add_to_cart` events for certain periods. To investigate this, I compared daily counts of key e-commerce events directly at the raw GA4 event level:

```
SELECT
  PARSE_DATE('%Y%m%d', event_date)AS dt,
  COUNTIF(event_name='add_to_cart')AS add_to_cart,
  COUNTIF(event_name='begin_checkout')AS begin_checkout,
  COUNTIF(event_name='purchase')AS purchase
FROM `bigquery-public-data.ga4_obfuscated_sample_ecommerce.events_*
```

```
`
WHERE _TABLE_SUFFIXBETWEEN'20201101'AND'20201128'
GROUPBY1
ORDERBY1;
```

This analysis showed that, during parts of November 2020, `add_to_cart` events were severely underreported or missing, while downstream events continued to be recorded. This broke the expected funnel logic and indicated a tracking issue rather than actual user behavior.

To address this, the analysis window was restricted to **December 2020–January 2021**, where event coverage across all funnel steps was consistent and the funnel behaved as expected. This ensured that conversion rates and drop-off analysis were based on reliable data.

Alternative approaches could have included redefining funnel steps (for example, treating `begin_checkout` as the first intent signal), modeling incomplete steps with imputation, or flagging affected days and excluding them dynamically at query time. For this project, narrowing the timeframe was the most transparent and defensible choice, balancing data quality with analytical clarity.

| Row | dt | add_to_cart | begin_checkout | purchase |
|---|---|---|---|---|
| 1 | 2020-11-01 | 4 | 127 | 14 |
| 2 | 2020-11-02 | 0 | 219 | 49 |
| 3 | 2020-11-03 | 0 | 233 | 49 |
| 4 | 2020-11-04 | 0 | 218 | 46 |
| 5 | 2020-11-05 | 0 | 195 | 26 |
| 6 | 2020-11-06 | 0 | 269 | 58 |
| 7 | 2020-11-07 | 0 | 150 | 31 |
| 8 | 2020-11-08 | 0 | 132 | 32 |
| 9 | 2020-11-09 | 0 | 211 | 52 |
| 10 | 2020-11-10 | 0 | 266 | 64 |
| 11 | 2020-11-11 | 0 | 248 | 64 |
| 12 | 2020-11-12 | 0 | 355 | 54 |
| 13 | 2020-11-13 | 0 | 605 | 79 |
| 14 | 2020-11-14 | 0 | 320 | 30 |