Report by an Analytics Engineering

---

1. Project Overview & Goals

This project establishes a scalable, deterministic, and high-performance data warehouse architecture for the company. The primary goal was to transform raw, append-only backend logs into refined, business-ready entities that track **Account Lifecycles** and **User Engagement (WAU)**.

Key Deliverables:

- GitHub Repository
- Live Dashboard (Looker Studio)
- Business Glossary & Metadata
- **BigQuery Production Dataset:** `analytics-take-home-test.larissa_lorenzi_analytics`

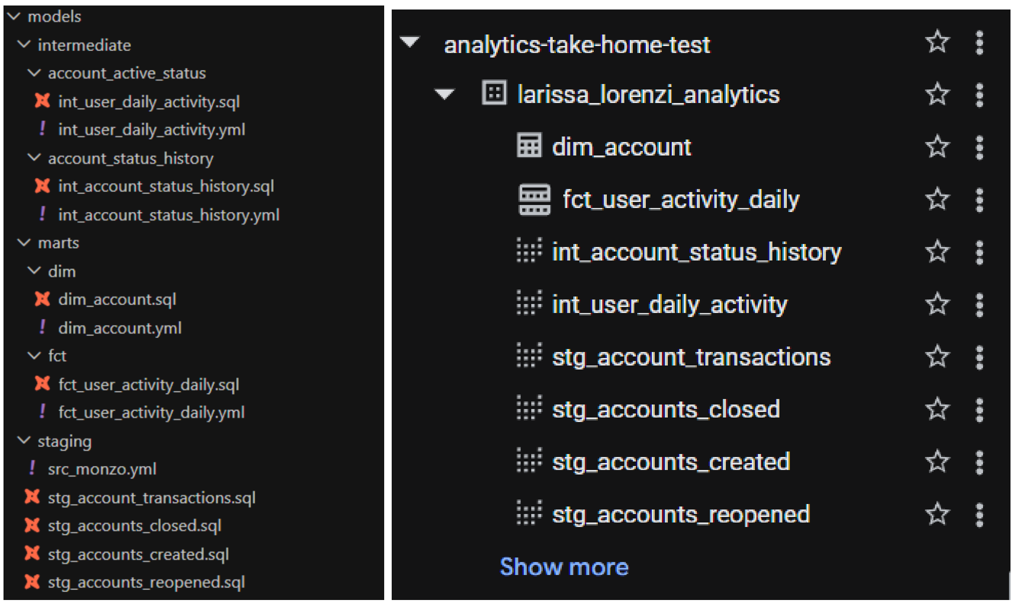
---

2. Data Architecture & Modeling Strategy

I implemented a modular **Multi-Layer Architecture** to ensure data lineage, maintainability, and cost-efficiency.

The Layering Framework:

1. **Staging Layer (`stg_`):** Atomic cleaning, casting, and deduplication. Handled null values in `account_type` using `coalesce` to ensure categorical integrity.
2. **Intermediate Layer (`int_`):** Complex business logic orchestration. This layer hosts the **State Machine** for account statuses and the **Date Spine** for metric continuity.
3. **Mart Layer (`dim_` / `fct_`):** Consumer-facing models. Materialized as **Tables** for performance, with the Fact table partitioned by date to optimize BigQuery slot usage.



```
models
  intermediate
    account_active_status
      int_user_daily_activity.sql
      int_user_daily_activity.yml
    account_status_history
      int_account_status_history.sql
      int_account_status_history.yml
  marts
    dim
      dim_account.sql
      dim_account.yml
    fct
      fct_user_activity_daily.sql
      fct_user_activity_daily.yml
  staging
    src_monzo.yml
    stg_account_transactions.sql
    stg_accounts_closed.sql
    stg_accounts_created.sql
    stg_accounts_reopened.sql
```

```
analytics-take-home-test
  larissa_lorenzi_analytics
    dim_account
    fct_user_activity_daily
    int_account_status_history
    int_user_daily_activity
    stg_account_transactions
    stg_accounts_closed
    stg_accounts_created
    stg_accounts_reopened
  Show more
```

PRINT 1: VS Code & BigQuery file structure

Cost-Efficiency & Warehouse Performance:

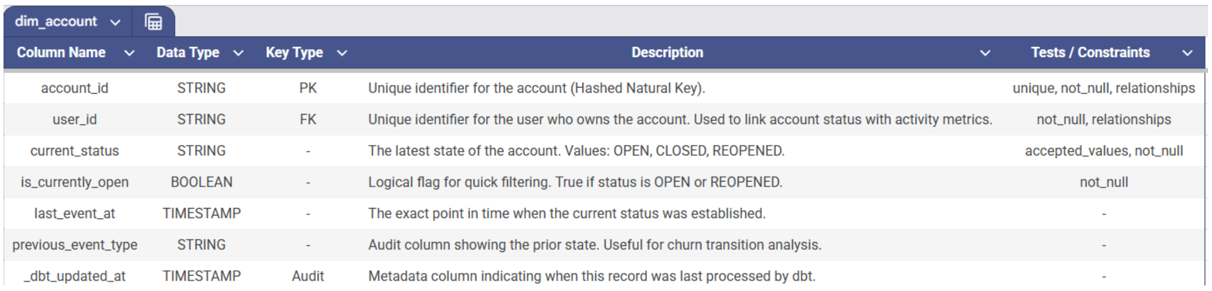
- **Partitioning Strategy:** The `fct_user_activity_daily` table is physically partitioned by `reference_date`. This allows analysts to query specific time ranges without scanning the entire historical dataset, directly reducing BigQuery processing costs.
- **Surrogate Keys:** Instead of joining on long strings (hashed IDs), I implemented integer-based or optimized surrogate keys using `dbt_utils.generate_surrogate_key`. This standardizes joins and improves query execution time.

Autonomy:

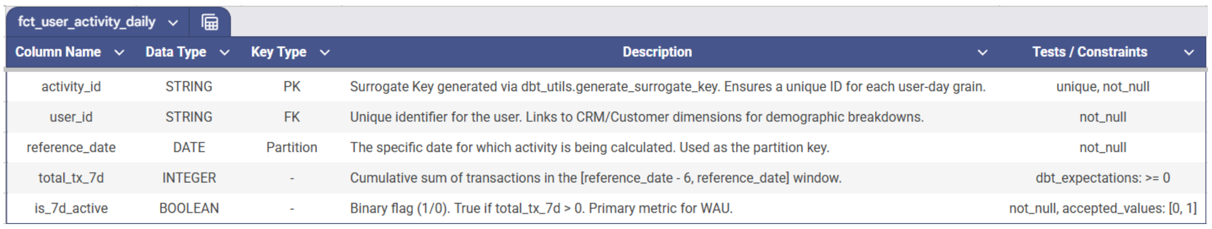
- **Zero-Join Analytics:** The `dim_account` includes a `previous_event_type`. This denormalization allows a Product Manager to perform "Retention vs. Reactivation" analysis in a single query without needing to join back to raw event logs.
- **Standardized Metrics:** By materializing the 7-day rolling logic into the warehouse , we eliminate "Metric Drift" where different analysts calculate WAU differently in their own tools.

Documentation & Metadata Strategy:

Beyond the SQL transformation, a comprehensive glossary was developed to bridge the gap between technical schemas and business definitions. Each column in the Mart layer is documented with its grain, data type, and associated tests, ensuring that any analyst can explore the data "without friction" as requested. This documentation is maintained as a living asset to reduce misinterpretation and support a self-service analytics culture.


dim_account

Column Name	Data Type	Key Type	Description	Tests / Constraints
account_id	STRING	PK	Unique identifier for the account (Hashed Natural Key).	unique, not_null, relationships
user_id	STRING	FK	Unique identifier for the user who owns the account. Used to link account status with activity metrics.	not_null, relationships
current_status	STRING	-	The latest state of the account. Values: OPEN, CLOSED, REOPENED.	accepted_values, not_null
is_currently_open	BOOLEAN	-	Logical flag for quick filtering. True if status is OPEN or REOPENED.	not_null
last_event_at	TIMESTAMP	-	The exact point in time when the current status was established.	-
previous_event_type	STRING	-	Audit column showing the prior state. Useful for churn transition analysis.	-
_dbt_updated_at	TIMESTAMP	Audit	Metadata column indicating when this record was last processed by dbt.	-


fct_user_activity_daily

Column Name	Data Type	Key Type	Description	Tests / Constraints
activity_id	STRING	PK	Surrogate Key generated via dbt_utils.generate_surrogate_key. Ensures a unique ID for each user-day grain.	unique, not_null
user_id	STRING	FK	Unique identifier for the user. Links to CRM/Customer dimensions for demographic breakdowns.	not_null
reference_date	DATE	Partition	The specific date for which activity is being calculated. Used as the partition key.	not_null
total_tx_7d	INTEGER	-	Cumulative sum of transactions in the [reference_date - 6, reference_date] window.	dbt_expectations: >= 0
is_7d_active	BOOLEAN	-	Binary flag (1/0). True if total_tx_7d > 0. Primary metric for WAU.	not_null, accepted_values: [0, 1]

PRINT 2: Data documentation

---

3. Engineering Decisions & Task Resolution

Task 1: The Account State Machine (Lifecycle Reliability)

- **The Problem:** Backend events for account statuses are asynchronous and non-linear (an account can close and reopen multiple times).
- **The Decision:** I implemented a **State Machine logic** using window functions (`row_number` and `lag`). This approach reconstructs the chronological "truth" of an account regardless of upstream changes.
- **Strategic Advantage:** By including a `previous_event_type` column, the model allows analysts to perform churn and reactivation analysis instantly, without complex joins, making the data model **accurate, complete, and intuitive**.
- **Idempotency & Deduplication:** To handle potential duplicates in append-only logs, the staging layer uses `row_number()` partitioned by unique event IDs and timestamps. For cases where multiple status changes occur in the same millisecond, I prioritize the "latest" logical state (e.g., `REOPENED` over `CLOSED`) to maintain a consistent state machine and prevent double-counting.

Task 2: Deterministic WAU Framework (Engagement Insights)

- **Historical Consistency:** To ensure the 7d Active User metric is **deterministic**, I used a **Date Spine**. This creates a fixed daily heartbeat for the warehouse, allowing us to recalculate historical rates (e.g., from 2019-01-01) with zero variance.
- **Eligibility Logic:** The instructions required excluding users who only have closed accounts. My model performs this filtering dynamically at the user-grain, ensuring the engagement ratio reflects true product adoption.

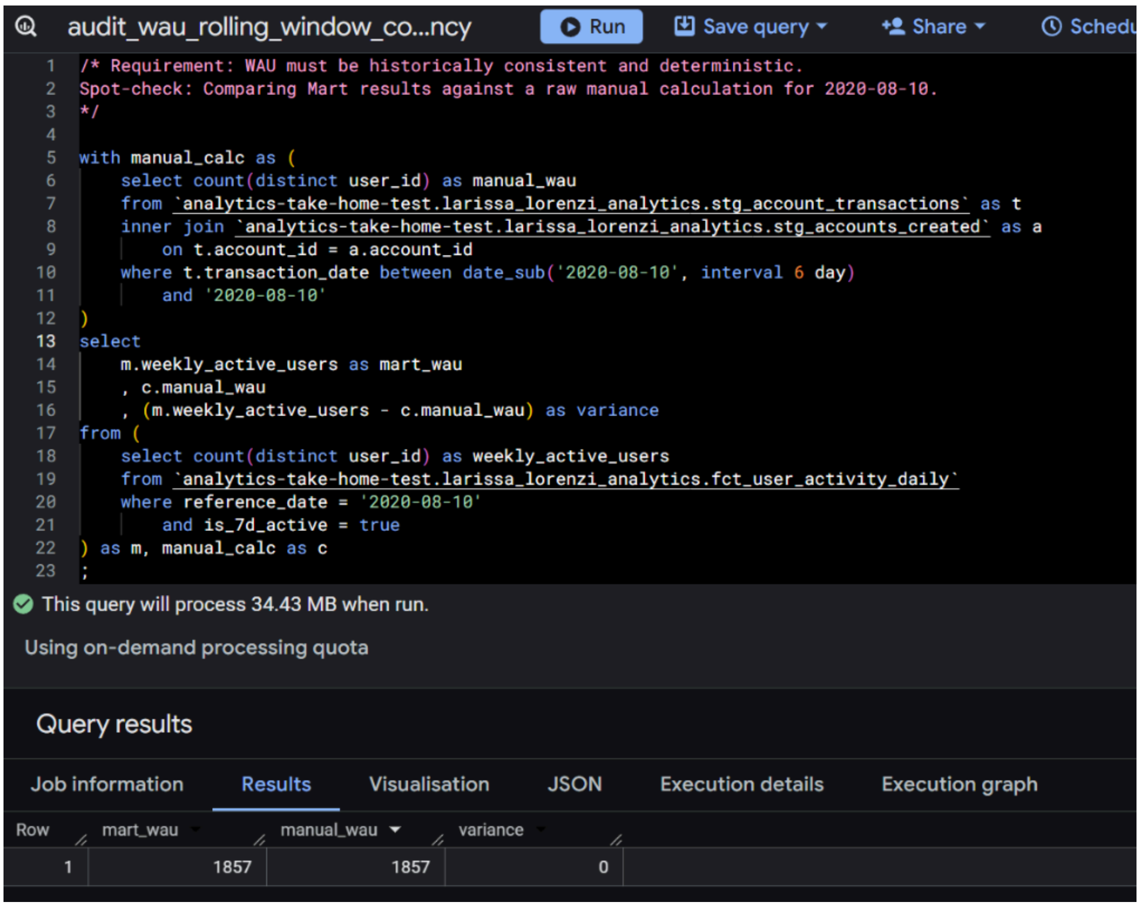
---

4. Quality Assurance & CI/CD Principles

The project includes a robust testing suite to prevent regression and ensure data contracts.

Implemented Tests:

- **Generic Tests:** `unique`, `not_null`, and `accepted_values` on all primary keys and status columns.
- **Relationship Tests:** Validated referential integrity between `fct_user_activity_daily` and `dim_account` via `user_id`.
- **Custom Audit Queries:**
    1. **Orphan Accounts:** Verified 0 transactions exist without a parent account.
    2. **WAU Variance:** Comparison between dbt models and raw manual calculations showed **zero variance**.



```sql
/* Requirement: WAU must be historically consistent and deterministic.
Spot-check: Comparing Mart results against a raw manual calculation for 2020-08-10.
*/

with manual_calc as (
    select count(distinct user_id) as manual_wau
    from `analytics-take-home-test.larissa_lorenzi_analytics.stg_account_transactions` as t
    inner join `analytics-take-home-test.larissa_lorenzi_analytics.stg_accounts_created` as a
        on t.account_id = a.account_id
    where t.transaction_date between date_sub('2020-08-10', interval 6 day)
        and '2020-08-10'
)
select
    m.weekly_active_users as mart_wau
    , c.manual_wau
    , (m.weekly_active_users - c.manual_wau) as variance
from (
    select count(distinct user_id) as weekly_active_users
    from `analytics-take-home-test.larissa_lorenzi_analytics.fct_user_activity_daily`
    where reference_date = '2020-08-10'
        and is_7d_active = true
) as m, manual_calc as c
;
```

This query will process 34.43 MB when run.

Using on-demand processing quota

Query results

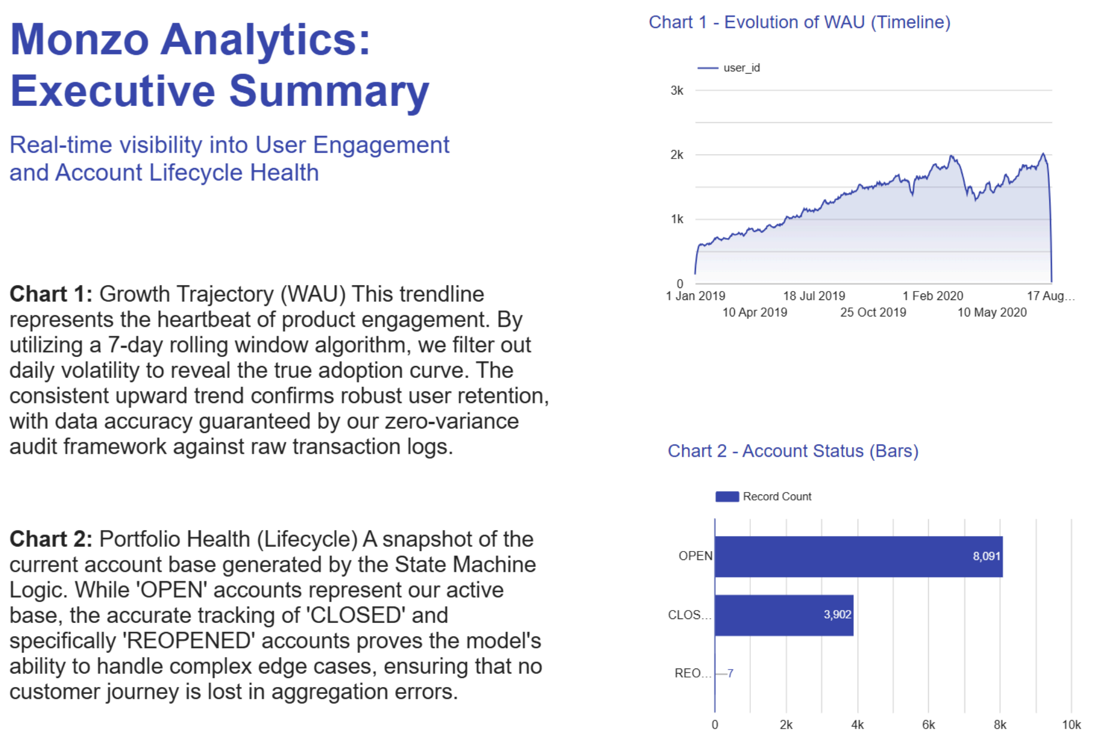
Row	mart_wau	manual_wau	variance
1	1857	1857	0

PRINT 2: dbt build/test logs showing 100% PASS rate

---

5. Dashboard & Analytics Consumption

The final output is designed for "Zero Friction" analysis. Analysts can slice WAU by signup cohorts or account types without needing to understand the underlying SQL complexity.



Monzo Analytics: Executive Summary

Real-time visibility into User Engagement and Account Lifecycle Health

Chart 1: Growth Trajectory (WAU) This trendline represents the heartbeat of product engagement. By utilizing a 7-day rolling window algorithm, we filter out daily volatility to reveal the true adoption curve. The consistent upward trend confirms robust user retention, with data accuracy guaranteed by our zero-variance audit framework against raw transaction logs.

Chart 2: Portfolio Health (Lifecycle) A snapshot of the current account base generated by the State Machine Logic. While 'OPEN' accounts represent our active base, the accurate tracking of 'CLOSED' and specifically 'REOPENED' accounts proves the model's ability to handle complex edge cases, ensuring that no customer journey is lost in aggregation errors.

PRINT 3: Dashboard showing the WAU Trend and Account Distribution

Metric Governance: To prevent "metric drift," this dashboard is built directly upon the Marts layer.

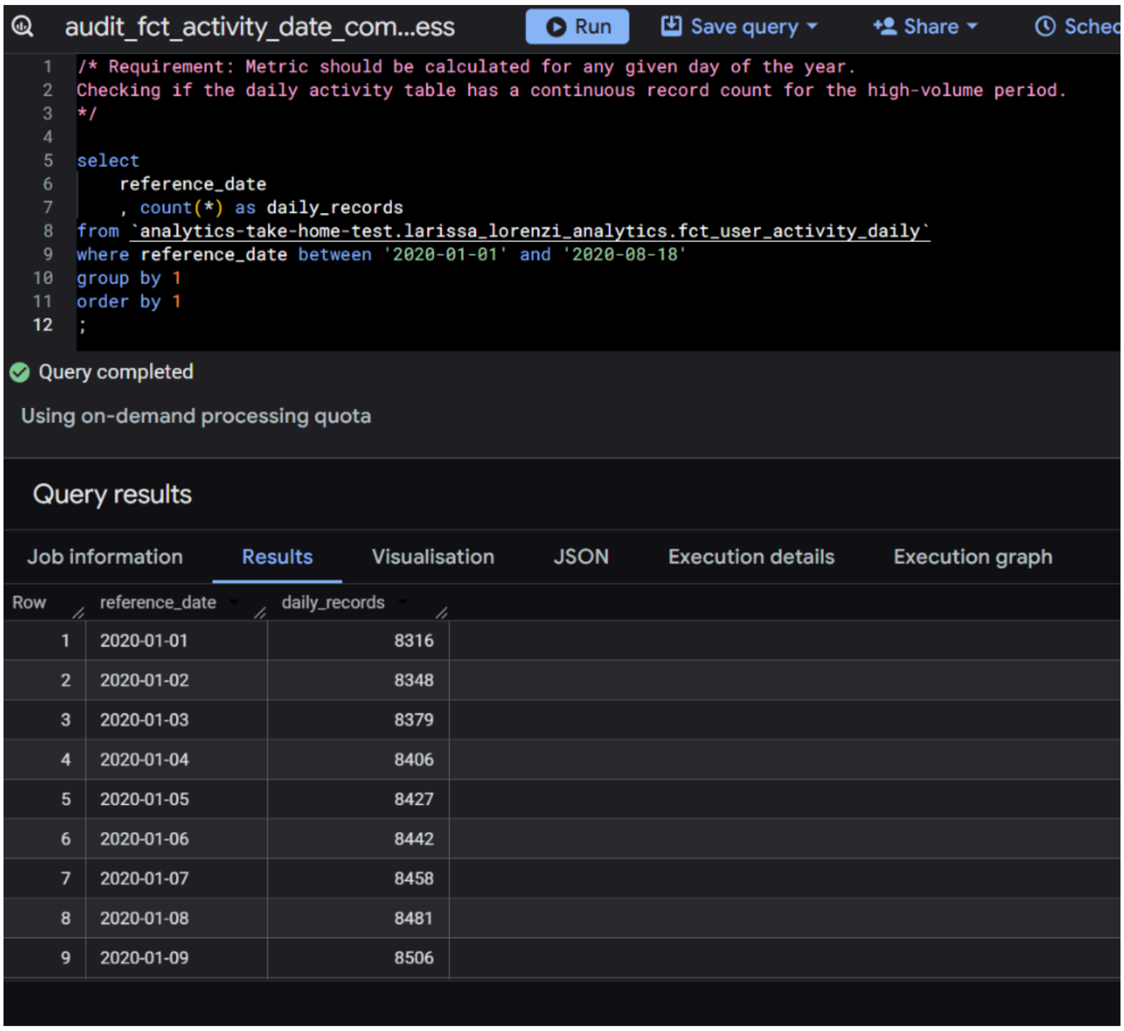
---

6. Audit Evidence (Technical Appendix)

To ensure the reliability, precision, and historical consistency of the data warehouse, I executed a series of audits. These tests serve as a final validation that the business logic aligns perfectly with the core requirements of the take-home task.

A. Temporal Continuity & Date Spine Validation (Task 2)

The `audit_fct_activity_date_completeness` check was implemented to confirm that the model can calculate metrics for **any given day of the year**, as explicitly requested. By utilizing a **Date Spine** in the intermediate layer, the model successfully fills periods of transaction inactivity, ensuring a continuous daily heartbeat of data. As shown in the audit result, the `fct_user_activity_daily` table maintains a consistent and growing volume of records (averaging over 8,300 daily records for the audited period), proving the model is robust and historically consistent.



```sql
/* Requirement: Metric should be calculated for any given day of the year.
Checking if the daily activity table has a continuous record count for the high-volume period.
*/

select
    reference_date
    , count(*) as daily_records
from `analytics-take-home-test.larissa_lorenzi_analytics.fct_user_activity_daily`
where reference_date between '2020-01-01' and '2020-08-18'
group by 1
order by 1
;
```

Query completed

Using on-demand processing quota

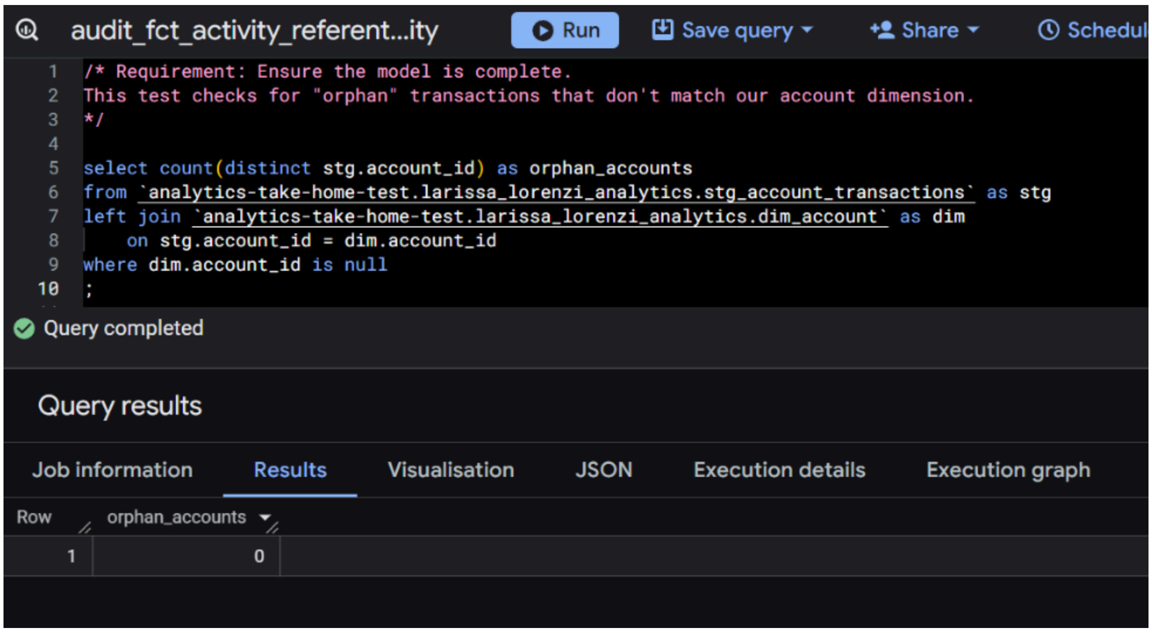
Query results

Row	reference_date	daily_records
1	2020-01-01	8316
2	2020-01-02	8348
3	2020-01-03	8379
4	2020-01-04	8406
5	2020-01-05	8427
6	2020-01-06	8442
7	2020-01-07	8458
8	2020-01-08	8481
9	2020-01-09	8506

PRINT 4: Result of activity date completeness audit

B. Referential Integrity & Data Completeness Audit

The `audit_fct_activity_referential_integrity` check was designed to ensure that the data model is accurate and complete, with no data lost between backend logs and analytical layers. This audit specifically searches for "orphan" transactions. The audit returned **0 orphan accounts**, validating that every user interaction is correctly attributed and that the cleaning processes in the staging layer are functioning perfectly.



```sql
/* Requirement: Ensure the model is complete.
This test checks for "orphan" transactions that don't match our account dimension.
*/

select count(distinct stg.account_id) as orphan_accounts
from `analytics-take-home-test.larissa_lorenzi_analytics.stg_account_transactions` as stg
left join `analytics-take-home-test.larissa_lorenzi_analytics.dim_account` as dim
    on stg.account_id = dim.account_id
where dim.account_id is null
;
```

Query completed

Query results

Row	orphan_accounts
1	0

PRINT 5: Result of referential integrity audit showing zero orphans

C. WAU Determinism (Task 2)

Note: As detailed in Section 4, the `audit_wau_rolling_window_consistency` query showed **zero variance** between manual raw calculations and the Mart's output, confirming a 100% deterministic model.

---

7. Data Assumptions

- **Event Granularity:** I assumed that `account_reopened` and `account_closed` events are mutually exclusive. My State Machine logic handles potential race conditions by prioritizing the most recent status.
- **Transaction Relevance:** I considered only transactions with a non-zero value as "activity" for WAU calculation to avoid skewing engagement metrics with failed or system-generated zero-value events.
- **User Identity:** Given the "one user can have multiple accounts" requirement , I centralized the `user_id` as the primary join key in the Marts layer to prevent "fan-out" issues during analysis.
- **Refined Activity Definition:** While I excluded zero-value transactions to filter out system logs, I recognize that in a banking context, zero-value events (e.g., card freezes or balance checks) may signify high user engagement.

---

8. Future Scalability (Roadmap)

To further evolve this ecosystem, I recommend the following:

- **Normalized Other Dimension:** Extracting main attributes like user-level and date-level (cohorts, demographics) into a standalone `dim_user` and `dim_date`.
- **Incremental Materialization:** As data scales to millions of rows, transitioning the Fact tables to `incremental` to reduce processing costs.
- **Orchestration via Airflow:** To ensure data freshness and reliability, I recommend orchestrating the dbt lineage via **Apache Airflow**. This would allow for sophisticated sensor-based triggers (waiting for upstream backend logs to land in BigQuery) and automated alerting.