UTS UNIVERSITY OF TECHNOLOGY SYDNEY

Building ELT data pipelines with Airflow

Big Data Engineering | Autumn, 2024

Taekjin Jeong (25099654)

<28.10.2024>

Table of Contents

1. Executive Summary

This report presents the design and implementation of a production-ready ELT data pipeline for processing and analysing Airbnb and Census data for Sydney. The project develops a robust data architecture using Apache Airflow and dbt Cloud that facilitates efficient data ingestion, transformation, and management. By structuring the pipeline according to the medallion architecture—where raw data is refined through Bronze, Silver, and Gold layers—the project ensures data consistency, reliability, and scalability for analytical needs. Moreover, Analytical insights derived from this data mart address several business questions, enabling a deeper exploration of trends and correlations such as neighbourhood popularity, revenue potential, and demographic influences on rental activity.

2. Introduction

This project processes Airbnb data (2020.5 ~ 2021.4) and Census data in Sydney stored in PostgreSQL database using Airflow and dbt. The core objective of the pipeline is to transform raw Airbnb and Census datasets into a structured format within a data warehouse, ultimately creating a data mart tailored for data-driven decision-making. The data mart, a refined layer within the Gold tier, provides a high-level view of key metrics and trends for understanding Sydney neighbourhoods' Airbnb activity.

3. Setup architectures (Part 1)

Corresponding file: Part_1.sql, dag_part_1.py

This project builds the necessary architectures (PostgreSQL, Airflow, dbt and DBeaver) to design the Airbnb data pipeline. Google Cloud Platform (GCP) provides scalable, secure data processing and storage infrastructure. We set up Cloud Composer to orchestrate ELT workflows and SQL instances for centralised data storage with PostgreSQL.

- **Step 1**: Cloud Composer and SQL on GCP

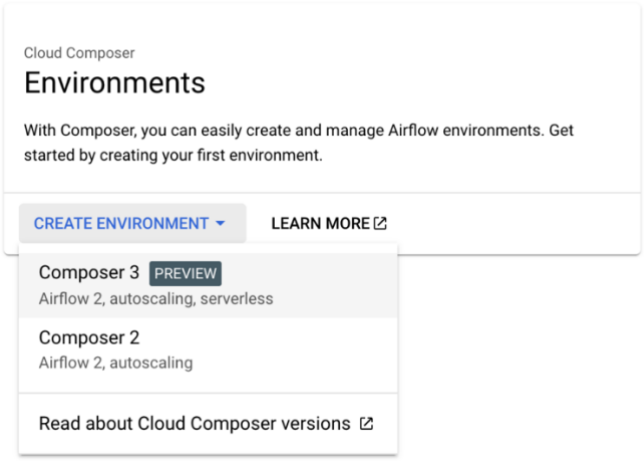


Figure 1. Setup Cloud composer

Pandas and apache-airflow-providers-postgres Pypi packages need to be installed when creating a new environment on GCP. Furthermore, each time the access IP environment changes, the IP address must be registered in SQL connections. As shown in Figure 2, three IP addresses were added for Dbt Cloud to connect to PostgreSQL. It is essential to enable the private path option as well.

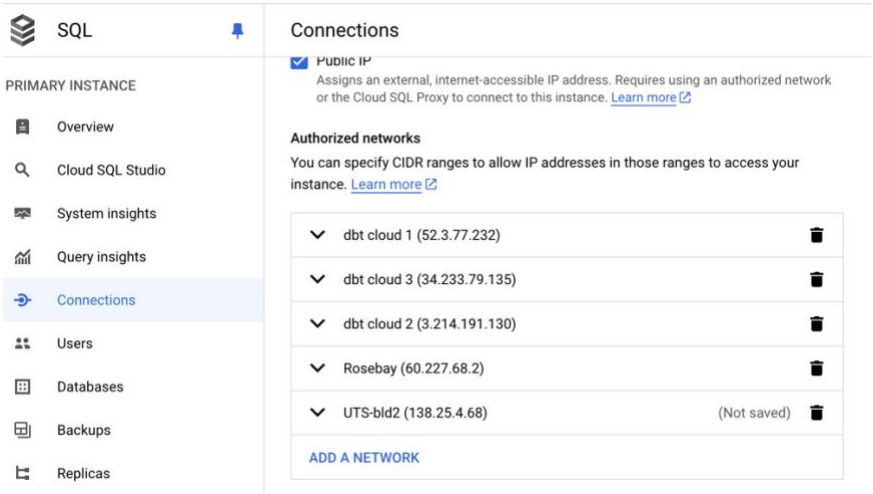


Figure 2. SQL connection

- **Step 2**: Setup PostgreSQL and create schema

The user and password should be enrolled before connecting PostgreSQL using DBeaver; we had addressed connection time-out issues on DBeaver (Figure 3) as we did not add the IP address to different IP environments. There are five specific datasets for this project: Airbnb listing (2020.5~2021.4), 2016Census_G01_NSW_LGA, 2016Census_G02_NSW_LGA,

NSW_LGA_CODE, NSW_LGA_SUBURB. Airbnb listing datasets include 12 months of Airbnb listing data for Sydney, G01 includes demographic data such as the number of people by LGA(local government area) and G02 shows selected medians and averages like ages and mortgage repayments. Other two tables (NSW_LGA_CODE, NSW_LGA_SUBURB) indicate LGA code, name and suburb name.

Figure 3. An error on PostgreSQL

In the ELT (extract-load-transform) process, the data transforms after loading, making it essential for the initial data ingestion phase to handle data of varying types and structures smoothly. To facilitate Airbnb listing datasets, we defined all columns in the BRONZE.RAW_AIRBNB table as VARCHAR. Using VARCHAR data types for all fields at this stage maximises flexibility, ensuring that the ELT process can ingest and store raw data directly without validation or type constraints. Defining different data type could lead to ingestion failures. In practice, storing data for scraped_date and host_since failed through Airflow. After several attempts, we realised that it is vital to avoid setting a specific data type. This approach also accommodates diverse data sources with mixed or inconsistent formats, which are common in raw datasets. However, other tables used different data types as the datasets are snapshot data.

- **Step 3**: Configure Airflow

Cloud composer includes Airflow UI, which provides a straightforward, interactive interface to monitor and manage workflows, allowing users to visualise dependencies, track progress, and troubleshoot task execution in real time. Its intuitive design enhances productivity by simplifying the orchestration of complex data pipelines. Dags files and raw datasets were

uploaded to Cloud storage, then Airflow automatically caught Dags files by Airflow scheduler (Figure 4). Before running the Dag, we added Postgres connection information on Airflow with a private DB IP address.

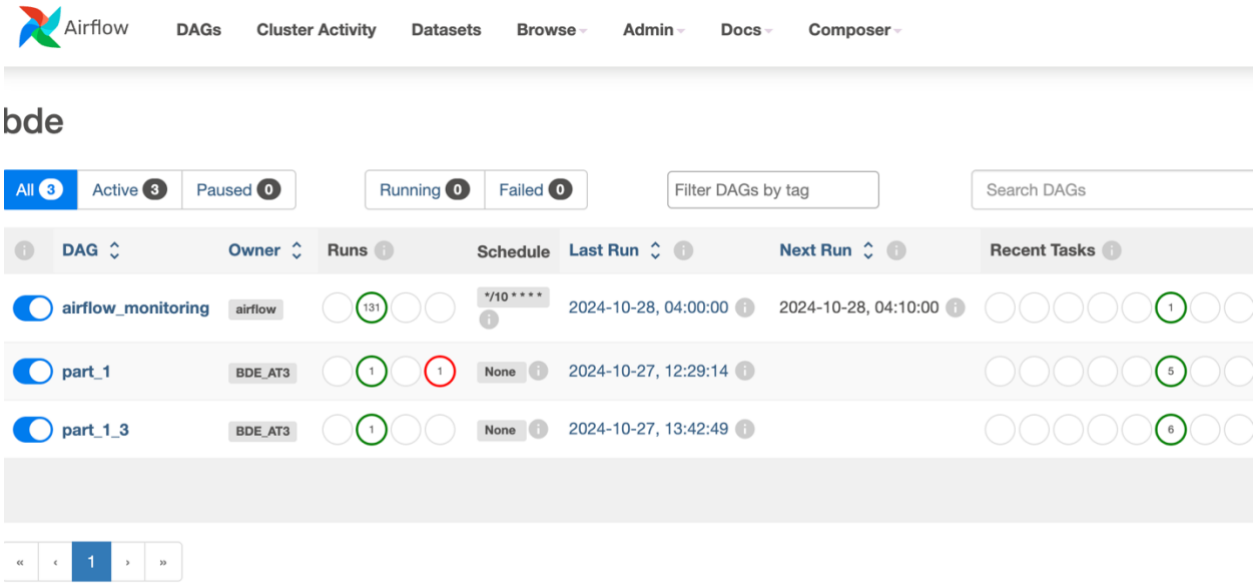


Figure 4. Airflow UI

In Airflow DAGs, task dependencies are defined to ensure the proper sequence of operations, allowing each task to execute in the intended order. By calling functions sequentially rather than all at once, we can manage dependencies effectively and control the data flow, ensuring each step completes successfully before the next begins. This approach also helps in error isolation, making it easier to troubleshoot specific stages if a task fails. Initially, we ran all three functions simultaneously, but when an issue occurred in the raw_airbnb stage, the other datasets were still successfully loaded. This created the inconvenience of needing to delete all data to maintain consistency. Moreover, it prevents potential data conflicts that could arise if tasks are executed simultaneously without control over dependencies. After running the Part_1 Dag, the Airflow DAG operated successfully, and the tables were confirmed to be created correctly shown in Figure 5.

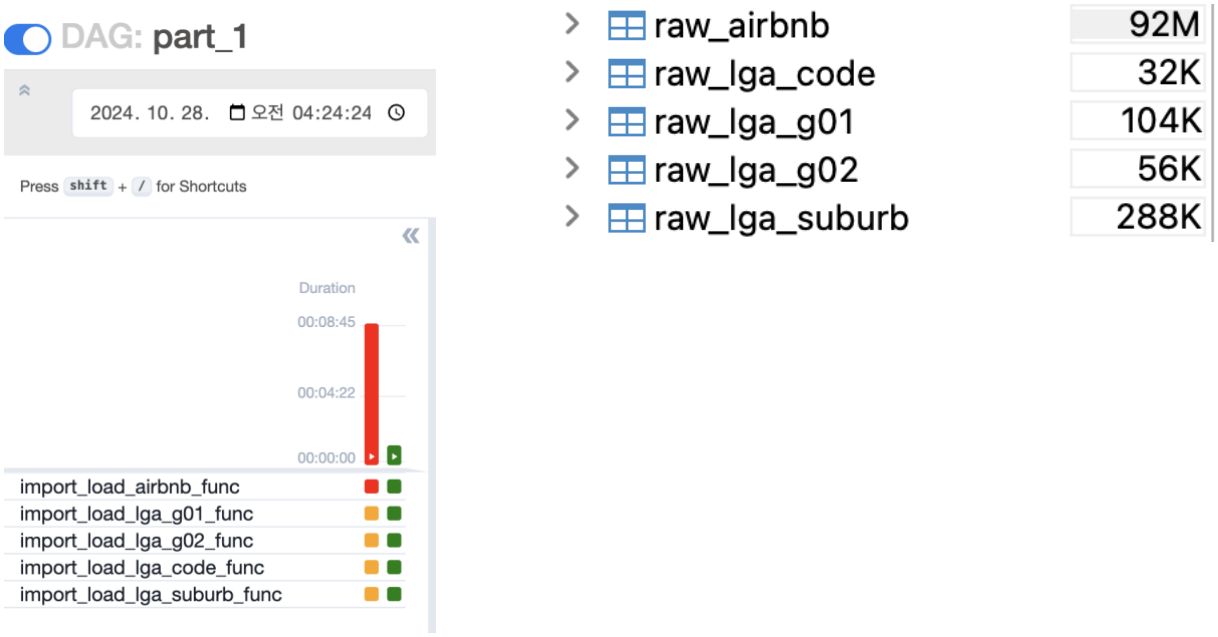


Figure 5. Airflow Dags run and tables

- **Step 4**: Set up dbt Cloud

  This project utilised dbt cloud to create a data warehouse architecture next part. dbt Cloud offers a variety of features, including version control and snapshots, as well as managing and automating data modelling and transformation tasks.

4. Design a Warehouse (Part 2)

Corresponding file: dbt cloud files

The Medallion architecture is a data design pattern used to logically organise data in a lakehouse, aiming to enhance the structure and quality of data incrementally and progressively as it flows through each layer of the architecture (from Bronze to Silver to Gold tables) (Databricks). Before designing data modelling, we configured dbt_project.yml file and source.yml to define the Medallion architecture. (Figure 6).

```yaml
target-path: "target" # directory which will store compiled SQL files
clean-targets: # directories to be removed by `dbt clean`
  - "target"
  - "dbt_packages"

models:
  bde_at_3:
    bronze:
      +materialized: table
      +schema: bronze
    silver:
      +materialized: table
      +schema: silver
    gold:
      star:
        +materialized: table
        +schema: gold
      mart:
        +materialized: view
        +schema: gold
snapshots:
  bde_at_3:
    +schema: silver
    +strategy: timestamp
    +updated_at: column_name
```

Figure 6. dbt_project.yml

In the Bronze layer, we stored raw tables sourced from the PostgreSQL database, capturing unprocessed data in its original form. We stored raw information for the Census data, specifically the g01, g02, and LGA code data, but with the addition of a suburb_id column to uniquely identify each suburb, thereby enriching our data model. Furthermore, we included two additional columns in the Airbnb listing table: Transaction ID and Updated_at. The Updated_at column is particularly useful for tracking changes over time, enabling us to utilise it effectively in our snapshot processes. Figure 7 shows the lineage of the listing table.

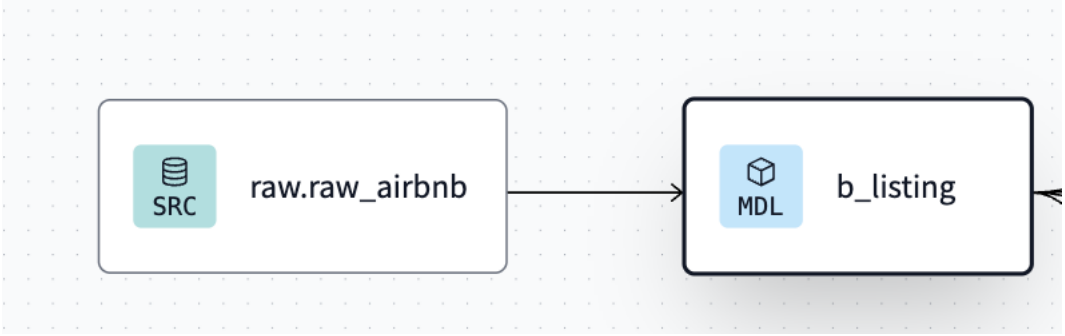


Figure 7. The lineage of b_listing

This project aims to clean and transform listing data in the silver layer. To view the Airbnb listing data from both the host and listing perspectives, we separated it into a host table and a listing table. Since the host_neighbourhood contains a mix of lga_name and suburb_name data

in a host table, it takes time to utilise effectively. To facilitate future data use, we standardised the data using lga_name. Table 1 explains the process of silver layer.

Table 1. Data preprocessing on Silver layer

	Cleansing	Transforming	Adding
S_host	- host_name: replace 'NaN' to 'unknown' - **host_since:** replace 'NaN' to '01/01/2000' - host_is_superhost: replace 'NaN' to 'f' - host_neighbourhood: replace 'NaN' to 'unknown'	- host_since: Change data type to date - host_neighbourhood: standardise text data to lowercase	Lga_code
S_listing	Replace 'Nan' to 0 in number columns (review_scores_rating, accuracy, cleanliness, checkin, communication, values)	- Standardise text data to lowercase (listing_neighbourhood, property_type, room_type) Change number columns' data type to Int or decimal type	Lga_code

Furthermore, the Silver layer contains snapshots. According to dbt documentations, Snapshots implement type-2 Slowly Changing Dimensions over mutable source tables. Kimbell group explains that three additional columns should be added to the dimension row with type 2 changes: 1) row effective date or date/time stamp 2) row expiration date or date/time stamp; and 3) current row indicator. We created updated_at columns in Bronze layer and Dbt snapshots creates these three columns from updated_at. SCDs identify how a row in a table changes over time by the timestamp strategy. Figure 8 shows the snapshot relates to dimensions table in gold layer.

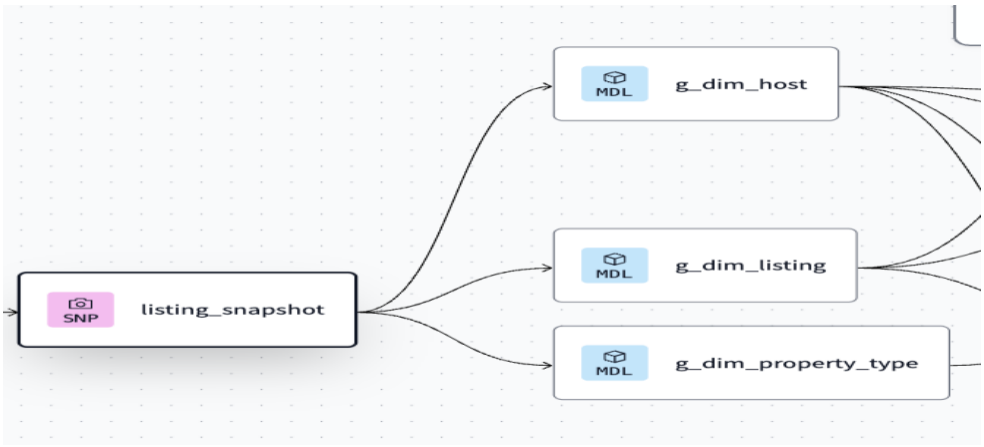


Figure 8. The snapshot

The Gold layer includes star and mart, and star includes five dimensions and a fact table (Table 2).

Table 2. Gold layer's layout

Star	Dimensions	Host, LGA, Property_type, Room_type, Suburb
	Fact	Listing
Mart	dm_listing_neighbourhood, dm_property_type, dm_host_neighbourhood	

LGA data did not need any data preprocessing, so lga dimension table is derived from Bronze layer. Other tables are mostly from listing table in Silver layer. Also, we inserted a query for dimensions to have latest data (Table 3).

Table 3. A sample query for dimensions

```
WITH latest_snapshots AS (
    SELECT *,
        ROW_NUMBER() OVER (PARTITION BY listing_id ORDER BY dbt_updated_at DESC) AS rn
    FROM
        {{ ref('listing_snapshot') }}
)

SELECT
    s.*
FROM
    {{ ref('s_listing') }} s
```

```
LEFT JOIN latest_snapshots ls ON s.listing_id = ls.listing_id AND ls.rn = 1
```

Datamart is materialised as views created from dimensions and fact tables. To create Datamart view, to create a data mart that compares data from the previous month to the current month, we added logic using the LAG function to facilitate month-over-month comparisons. Figure 9 is the result of dm_listing_neighbourhood and stakeholders directly gain insights such as the number of minimum, maximum, median and average price for active listings and average Estimated revenue per active listings. For instance, Sydney recorded the highest total_estimated_revenue, but Hunter Hill stands out with the highest figure when comparing the average estimated revenue per active listing.

3.4s | Returned 29 rows. Change row display Download CSV

listing_neighbourhood	lga_code	month_year	prev_active_listings	active_listings	active_listing_rate	min_price
ryde	16700	05/2020	NULL	6593	100.0	20.0
strathfield	17100	05/2020	NULL	1004	100.0	15.0
sutherland shire	17150	05/2020	NULL	1292	100.0	35.0
sydney	17200	05/2020	NULL	114963	100.0	0.0
the hills shire	17420	05/2020	NULL	676	100.0	24.0
waverley	18050	05/2020	NULL	27777	100.0	5.0
willoughby	18250	05/2020	NULL	4526	100.0	26.0
woollahra	18500	05/2020	NULL	14325	100.0	23.0

Figure 9. The result of a data mart query

5. End to end Orchestration (Part 3)

Corresponding file: dag_part_1_3.py

After completing the setup of the Medallion architecture in the development environment, a job was created in the deployment environment to trigger it through Airflow. Before triggering the job, some variables were added in Airflow (Figure 10).

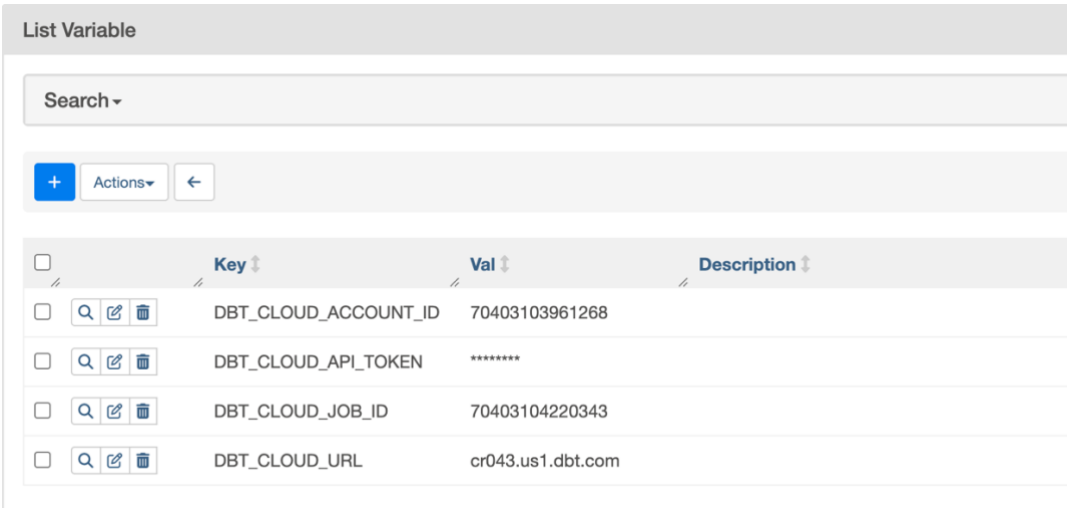


Figure 10. Setup variables on Airflow

When we reran the dag, we addressed two problems: 1) The large data volume prevented storing the data in PostgreSQL as a single DataFrame. 2) Duplicate primary keys in the data for the following month caused storage issues. To address the first issue, data was inserted file-by-file. For the second issue, code was added in Table 4 to remove listing_id as the primary key and replace it with a composite constraint using listing_id and scraped_date. Finally, each file was divided into chunks before insertion to improve performance during data insertion.

Table 4. A query for addressing issues

```
ALTER TABLE BRONZE.RAW_AIRBNB
DROP CONSTRAINT raw_airbnb_pkey,
ADD CONSTRAINT raw_airbnb_pkey PRIMARY KEY (listing_id, scraped_date);
```

Figures 11 through 13 demonstrate the success of the project's orchestration, showcasing the DAG execution results, dbt trigger outcomes, and the final architecture lineage. The total raw Airbnb table contains 412,122 rows (see Figure 14), and each schema (Silver and Gold) has been created accurately (see Figure 15).

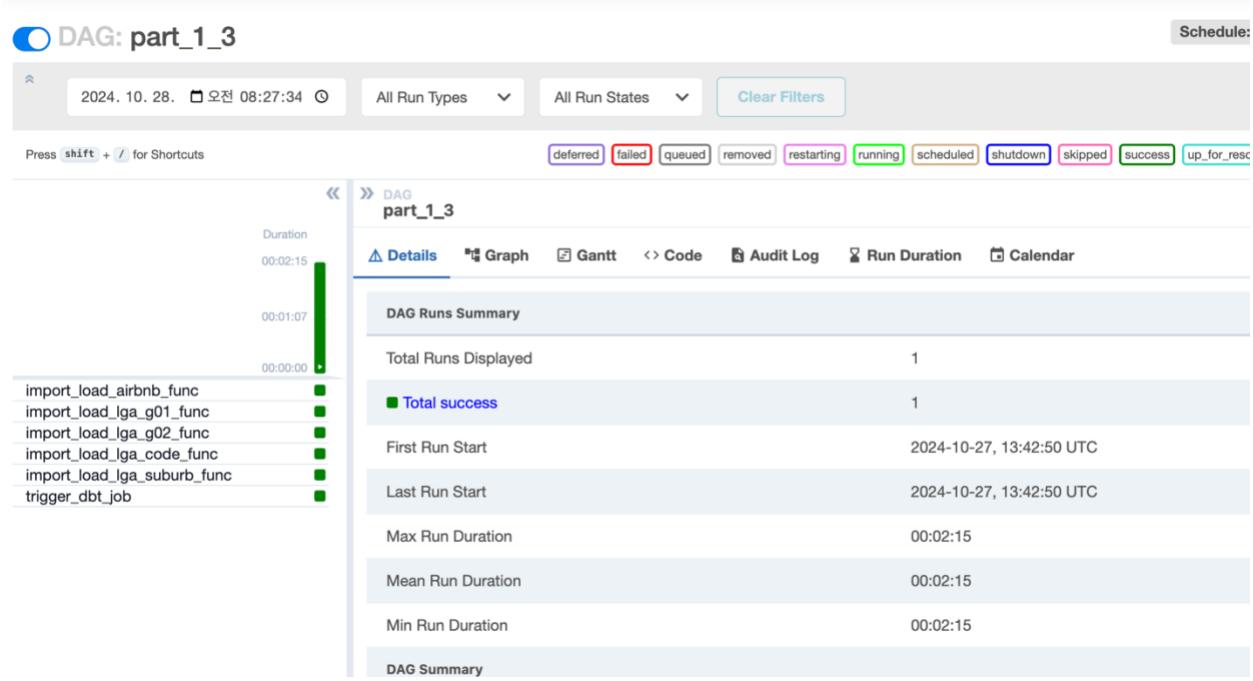


Figure 11. The DAG execution results

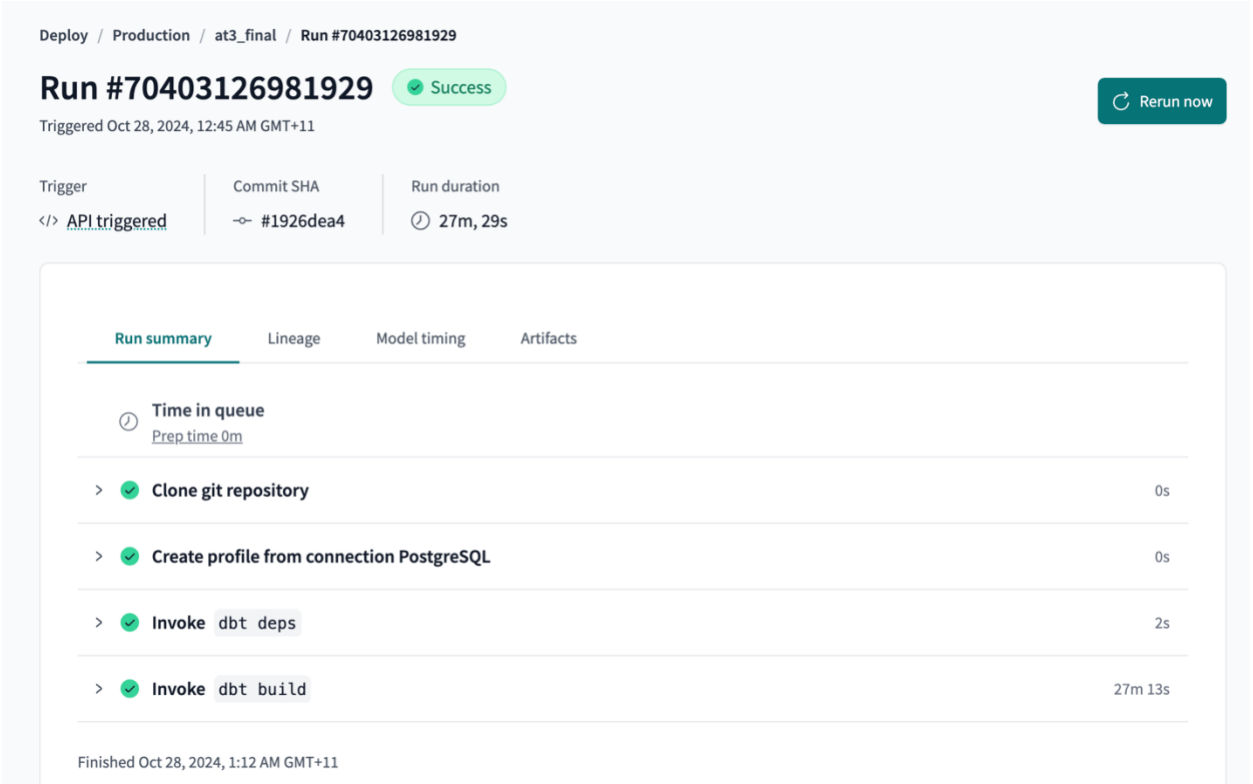


Figure 12. dbt trigger outcomes

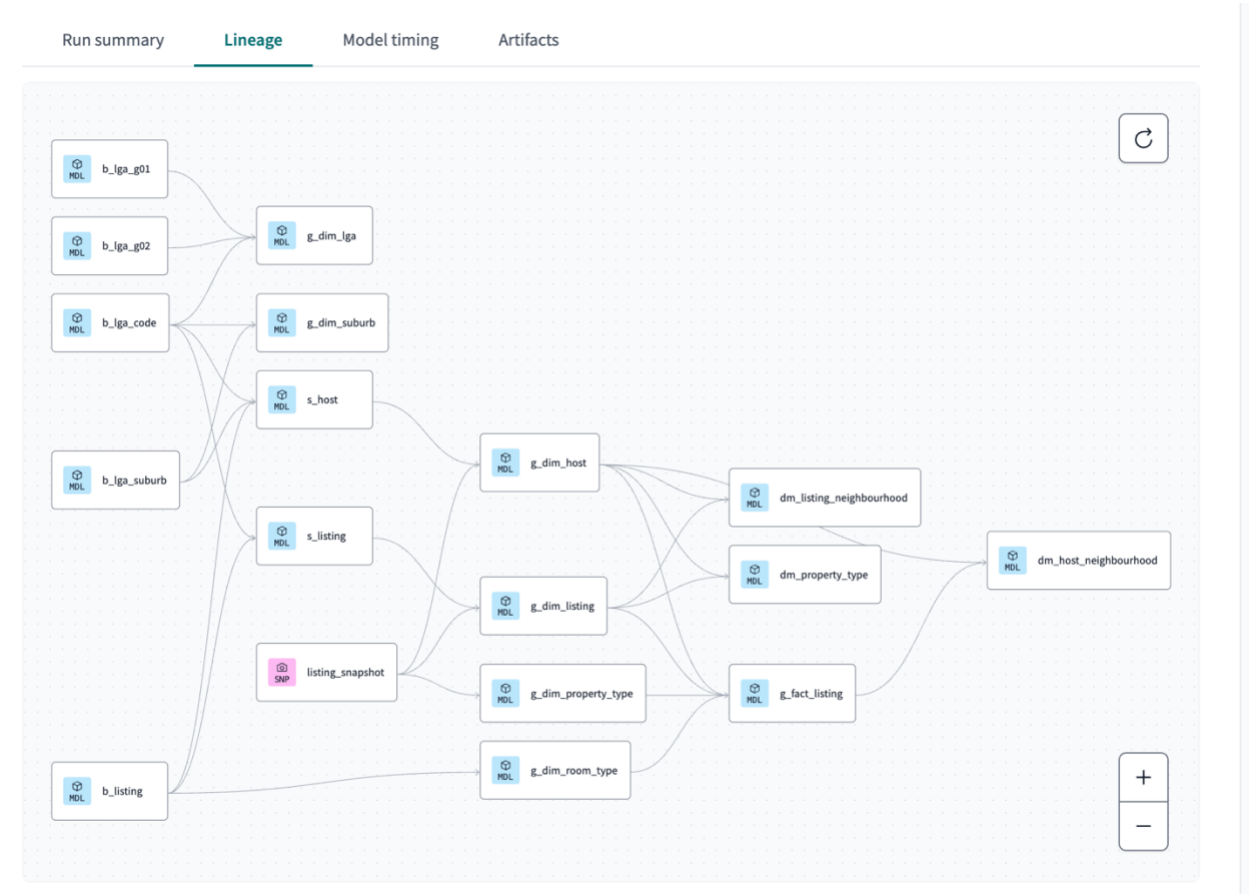


Figure 13. the final architecture lineage

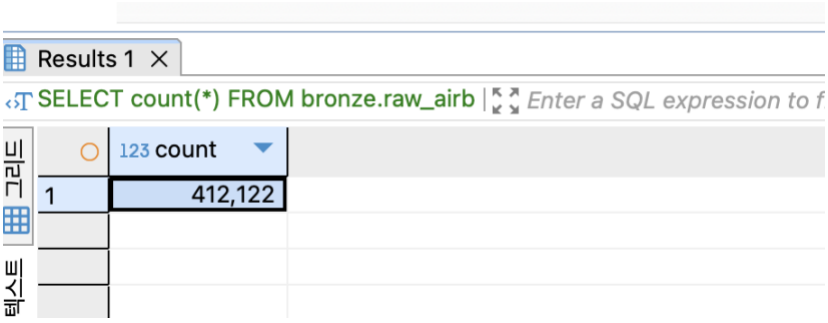


Figure 14. The number of rows

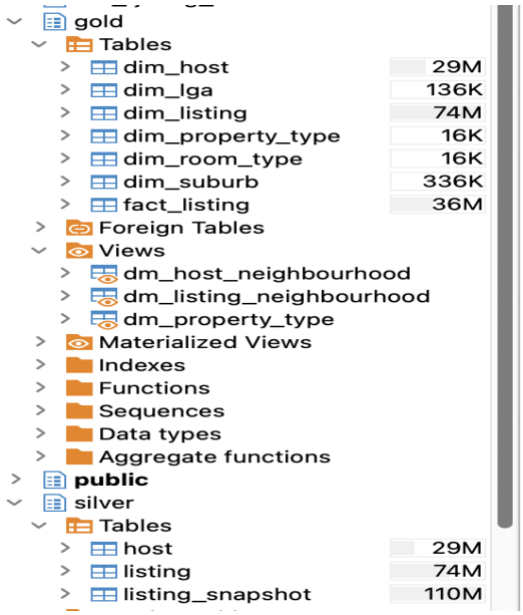
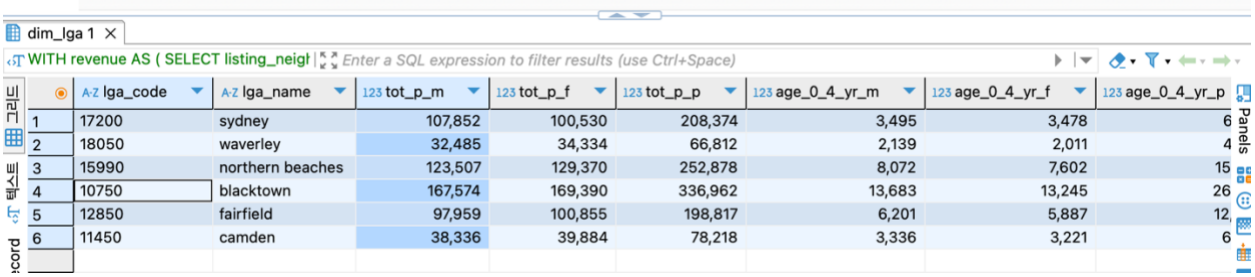


Figure 15. Schemas after orchestration

6. Ad-hoc analysis (Part 4)

Corresponding file: Part_4.sql

a. What are the demographic differences (e.g., age group distribution, household size) between the top 3 performing and lowest 3 performing LGAs based on estimated revenue per active listing over the last 12 months?



	lga_code	lga_name	tot_p_m	tot_p_f	tot_p_p	age_0_4_yr_m	age_0_4_yr_f	age_0_4_yr_p
1	17200	sydney	107,852	100,530	208,374	3,495	3,478	6
2	18050	waverley	32,485	34,334	66,812	2,139	2,011	4
3	15990	northern beaches	123,507	129,370	252,878	8,072	7,602	15
4	10750	blacktown	167,574	169,390	336,962	13,683	13,245	26
5	12850	fairfield	97,959	100,855	198,817	6,201	5,887	12
6	11450	camden	38,336	39,884	78,218	3,336	3,221	6

Figure 16. Question 1 result

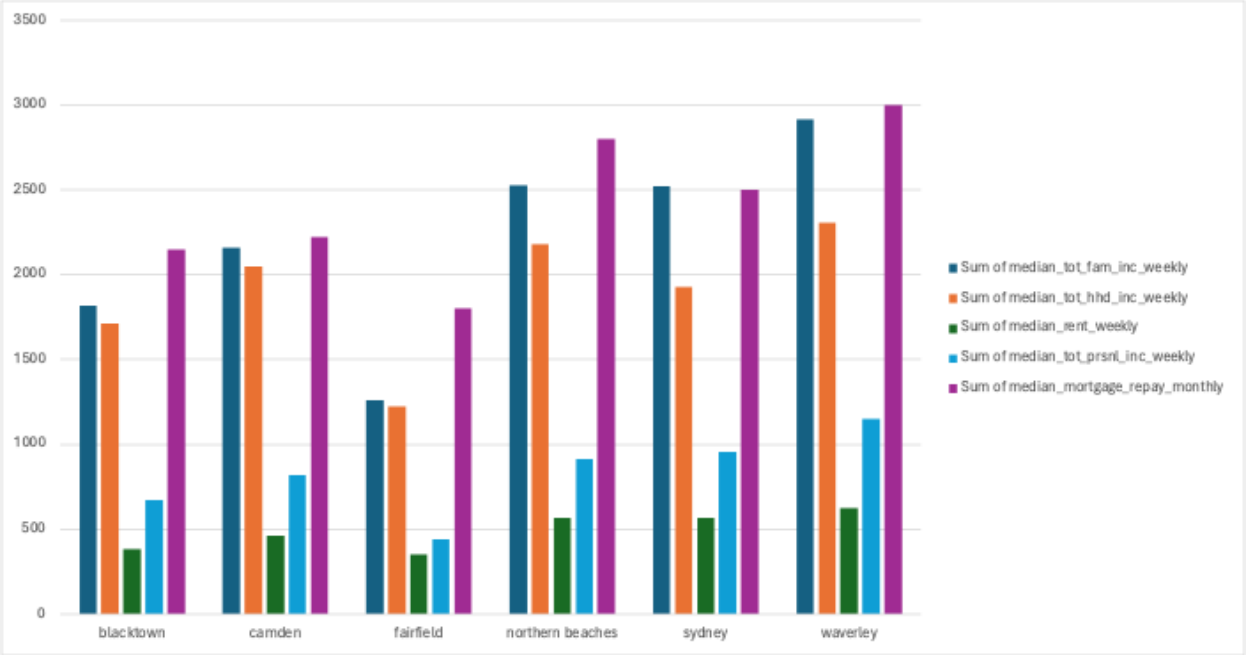


Figure 17. Q1 result's graph

It was observed that the top 3 performing LGAs have notably higher values across several socioeconomic indicators. Figure 17 suggests that households in the top-performing LGAs generally have a higher combined family income, which could indicate a higher capacity for discretionary spending and investment in property. Furthermore, Median Mortgage Repayments are higher top 3 performing LGAs. It implies these areas may reflect a more competitive housing market similar to showing in Median rent. This result highlights that the top-performing LGAs are associated with affluent demographic characteristics.

b. Is there a correlation between the median age of a neighbourhood (from Census data) and the revenue generated per active listing in that neighbourhood?

The correlation coefficient of 0.0556 indicates a very weak positive correlation between the median age of a neighbourhood and the revenue generated per active listing. The relationship is not strong enough to imply any significant connection. However, from the insight of question 1, we compared median rent, median mortgage repayments, and median household income. It results in 0.55, 0.49 and 0.33. This result confirms a strong correlation between real estate-related data and Airbnb revenue. Property values, rental prices, and neighbourhood demographics likely contribute significantly to the revenue generated per listing. Understanding this relationship can help people make informed decisions regarding pricing strategies.

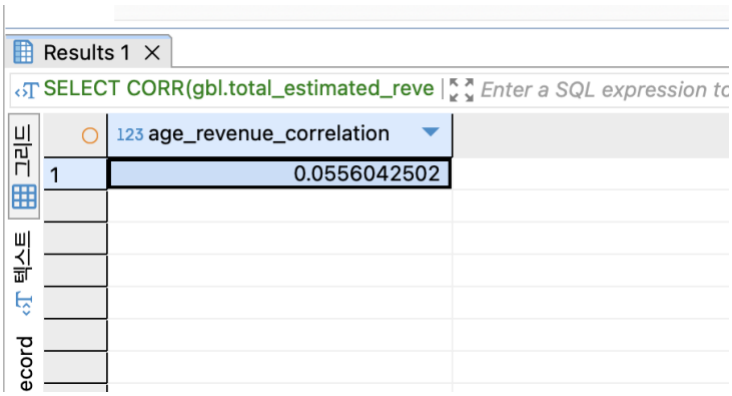


Figure 18. Question 2 result

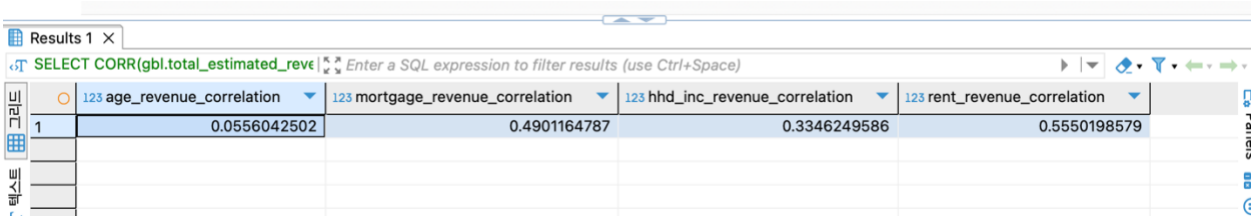


Figure 19. Question 2 result (Further analysis)

c. What will be the best type of listing (property type, room type and accommodates for) for the top 5 "listing_neighbourhood" (in terms of estimated revenue per active listing) to have the highest number of stays?

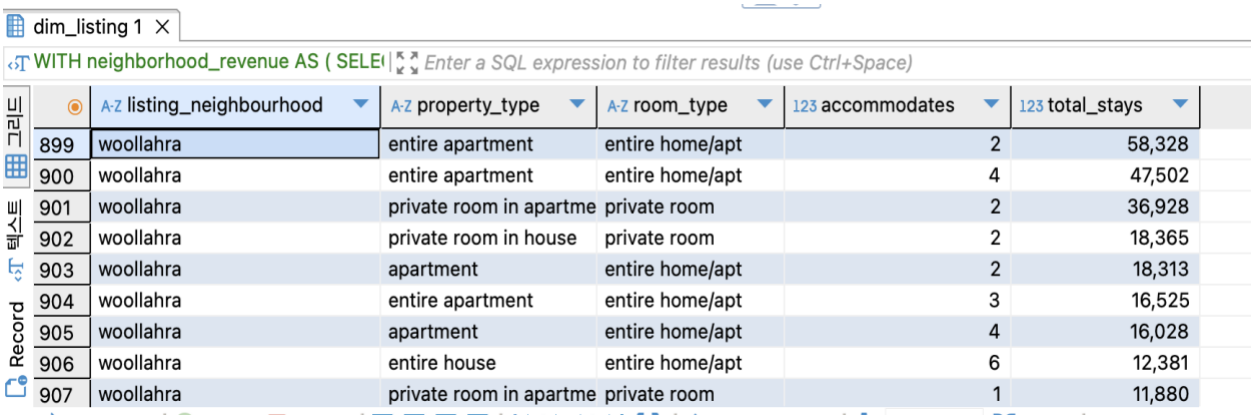


Figure 20. Question 3 result

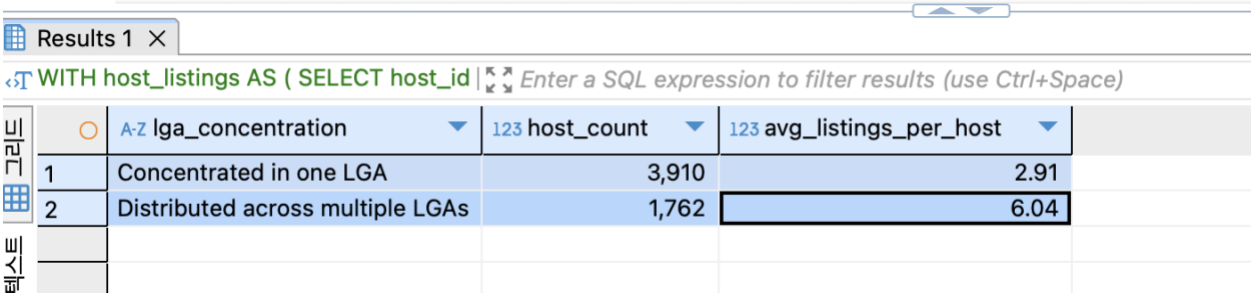
Table 5. Question 3 result

	Listing_neighnourhood	Property_Type	Room_type	accomodates	Total_stay
1	Woollahra	Entire apartment	Entire home/apt	2	58,328
2	Waverley	Entire apartment	Entire home/apt	2	214,352

3	Northern beaches	Entire apartment	Entire home/apt	4	133,597
4	Mosman	Entire apartment	Entire home/apt	2	19,791
5	Campbelltown	Private room in house	Private_room	2	4,730

The analysis indicates that entire apartments are the most successful property type across the top four neighbourhoods, particularly in Waverley. The Northern Beaches also shows promise with a more extensive accommodation capacity of 4 guests, indicating the potential for increased revenue from larger groups. Entire apartments with accommodating 2 guests appear to be the most effective strategy for maximising stays.

d. For, hosts with multiple listings, are their properties concentrated within the same LGA, or are they distributed across different LGAs?

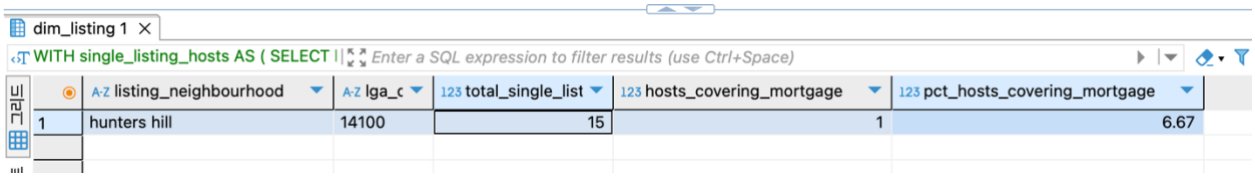


	lga_concentration	host_count	avg_listings_per_host
1	Concentrated in one LGA	3,910	2.91
2	Distributed across multiple LGAs	1,762	6.04

Figure 21. Question 4 result

The result reveals that most hosts with multiple listings (3,910) have their properties concentrated within a single Local Government Area (LGA). These hosts average about 2.91 listings each. Conversely, fewer hosts (1,762) distribute their listings across multiple LGAs, but these hosts maintain a higher average of 6.04 listings each.

e. For hosts with a single Airbnb listing, does the estimated revenue over the last 12 months cover the annualised median mortgage repayment in the corresponding LGA? Which LGA has the highest percentage of hosts that can cover it?



	listing_neighbourhood	lga_c	total_single_list	hosts_covering_mortgage	pct_hosts_covering_mortgage
1	hunters hill	14100	15	1	6.67

Figure 22. Question 5 result

The analysis indicates that hosts with a single Airbnb listing are generally unable to cover their mortgage repayments. In Hunters Hill, only **6.67%** of these hosts can meet their mortgage obligations based on their estimated annual revenue from Airbnb, representing the highest percentage among the analysed areas. This suggests that hosts in other LGAs face even greater challenges in repaying their mortgages, indicating a significant financial strain on those relying solely on Airbnb income.

7. Limitation

One area for improvement addresses the minimal use of the fact table, which typically serves as a central source of quantifiable data, such as activity measures. By relying less on the fact table, we may need more profound insights from granular, event-based data. However, during the business analysis, we identified several additional columns that could enhance the fact table's utility and provide deeper insights. By incorporating these columns, we can capture more granular transaction data. This would enable us to perform more detailed analyses and generate more accurate metrics. For the next step, we plan to redesign the fact table to integrate these enhancements, facilitating better calculations and ultimately leading to improved outcomes in our analyses.

8. Conclusion

This project focuses on data orchestration, data warehouse modelling, and advanced data analysis. In the orchestration phase, we installed Airflow and PostgreSQL in the cloud, creating custom DAG files that integrate seamlessly with dbt Cloud. We implemented the Medallion architecture to design our data warehouse, ensuring a structured and efficient flow of data from raw to refined states. Utilising the insights derived from the Gold layer data, we were able to uncover significant trends and patterns that inform strategic decision-making. Overall, this project not only enhances our understanding of data workflows but also provides actionable insights that can drive business success.

9. References

- Medallion Architecture (https://www.databricks.com/glossary/medallion-architecture)
- Dbt documentation (https://docs.getdbt.com/docs/build/snapshots)
- Kimbell group documentation (https://www.kimballgroup.com/data-warehouse-business-intelligence-resources/kimball-techniques/dimensional-modeling-techniques/type-2/)