Building ELT Pipeline with Airflow

For Sydney Airbnb Listings Dataset

Big Data Engineering | University of Technology Sydney

TABLE OF CONTENTS

1. Introduction

1.1 Brief Introduction

Airbnb, a leading online platform, connects guests and hosts worldwide by facilitating short-term and long-term accommodations and homestay experiences. This project aims to build an Extraction, Loading, and Transformation (ETL) pipeline for Sydney Airbnb listings dataset. Directed Acyclic Graph (DAG) in Apache Airflow will be built for orchestration and scheduling tasks, while dbt will handle transformations and design a data warehouse architecture on Postgres based on the Medallion architecture. Finally, the project will explore the data and generate meaningful business insights regarding Sydney listings from the constructed warehouse on Postgres.

1.2 About the Dataset

Three types of datasets will be used in this project. The primary focus is the monthly Airbnb listings data, which was scraped by Inside Airbnb, covering the period from May 2020 to April 2012. This dataset contains listing review scores, listing property attributes, and host information. In addition, Local Governance Aeras (LGAs) mapping data and census data relate to these LGAs from Australian Bureau of Statistics (ABS) are provided the construction of the data warehouse and support business analysis.

1.3 Executive Architecture

The main workflow of this project, shown in Fig1, involves constructing the ELT pipeline, and data warehouse, using Apache Airflow, dbt and PostgreSQL, followed by performing business analysis and generating business insights.

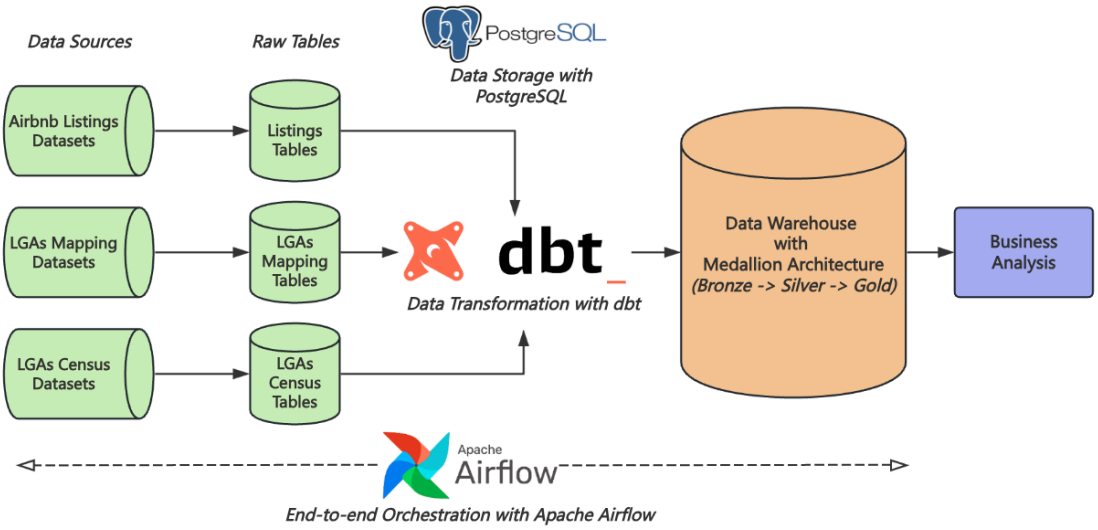

Fig 1. Project Architecture

2. Initial Setup

The project started with setting up Google Could, along with Google Cloud Composer, PostgreSQL instance, and Airflow. In addition, DBeaver was also be set up as a SQL client to access PostgreSQL, and Data Build Tool (dbt) was used to perform data transformations and design a data warehouse with Medallion architecture.

2.1 Google Cloud

1. **Google Cloud Composer**: is used to orchestrate the Airflow DAGs and handle ELT pipeline. In this project , a Composer environment '*bde*' was created, and *pandas* and *apache-airflow-providers-postgres* were added into the environment PyPI packages (Fig 2).

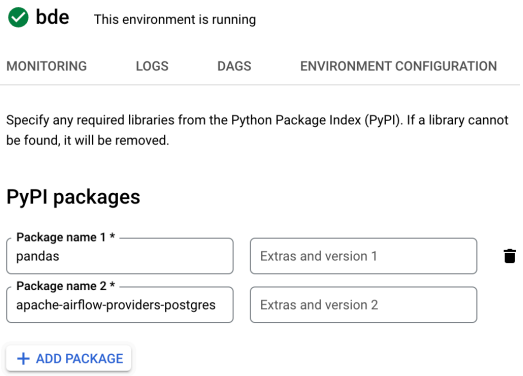

Fig 2. PyPI packages

2. **PostgreSQL:** a managed instance provided by Google Cloud, is where the raw data and data warehouse stored. A PostgreSQL instance was created with configurations:
    a. Cloud SQL edition: Enterprise
    b. Edition preset: Sandbox
    c. Database version: PostgreSQL 16
    d. Instance ID: bde
    e. Password: password of choice
    f. Region: asutralia-southeast1 (Sydney)
    g. Storage Capacity: 100 GB
    h. Connections: Enabled Private IP with default network, and enabled private path (Fig 3)

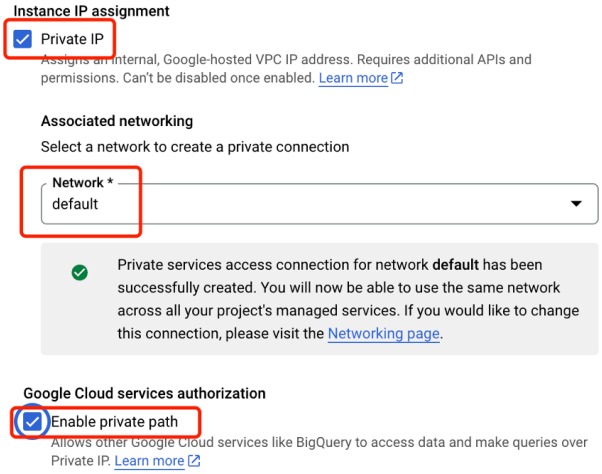

Fig 3. Connection configurations

After the creation of PostgreSQL instance, retrieved the public and private IP address (Fig 4).

Instance ID	Issues	Cloud SQL edition	Type	Public IP address	Private IP address
bde		Enterprise	PostgreSQL 16	34.87.192.196	10.125.0.3

Fig 4. IP addresses

3. **Apache Airflow:** deployed on Google Could Composer, is responsible for loading data into PostgreSQL, triggering dbt for transformations, and managing dependencies across the pipeline. This can be accessed by navigating to Airflow webserver in Google Composer environment. Navigated to Admin, Connections, and added a new record by specifying the Host with private IP address retrieved from PostgreSQL instance, with login as ‘postgres’ and port as ‘5432’ (Fig 5).

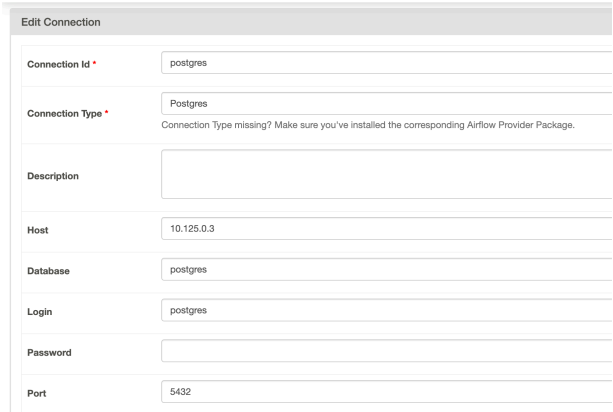

Fig 5. Connecting Airflow with Postgres

2.2 DBeaver

DBeaver is used as a SQL client to access PostgreSQL database, and performing data queries. The PostgreSQL was connected by specifying the Host with public IP address retrieved from PostgreSQL instance, and the Password with instance Password (Fig 6).

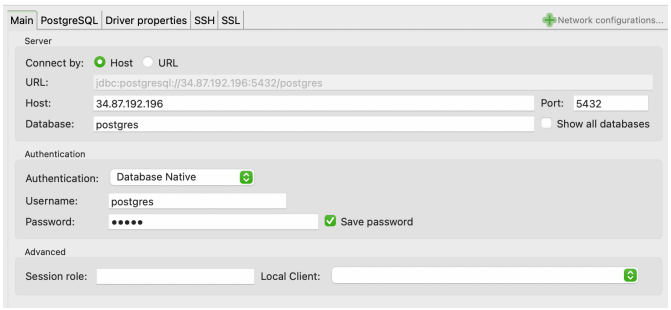

Fig 6. Connecting DBeaver with Postgres

2.3 Data Build Tool (dbt)

The dbt cloud is used for the ELT transformations, where raw data from PostgreSQL is processed, and it acts as a tool to design and construct a data warehouse with Medallion architecture. In dbt cloud, a project was created with environments and repository setups.

For environments, set up a connection with PostgreSQL by specifying the Sever Hostname with public IP address retrieved from PostgreSQL instance, along with Port and Database name as Fig 7. Then, created corresponding deployment environment and selected a managed repository with a specified name (Fig 8).

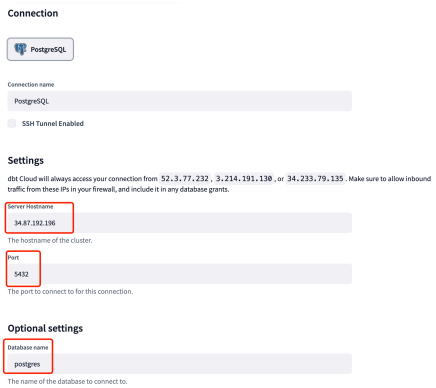

Fig 7. Connecting dbt with Postgres

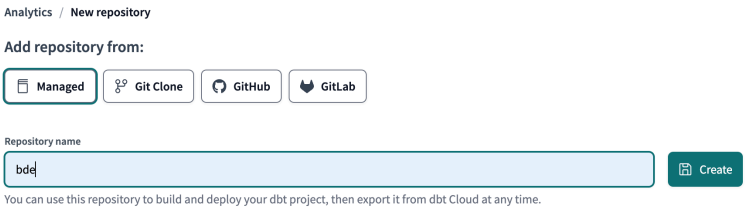

Fig 8. Adding repository

3. Data Ingestion with Airflow

3.1 Initial Datasets and Schema Preparation

Corresponding code source: part_1.sql

1. **Uploaded datasets into Google Cloud Airflow storage bucket**: The first month of Airbnb listings data (*listing/05-2020.csv*) and the reference datasets (*NSW_LGA/NSW_LGA_CODE.csv*, *NSW_LGA/NSW_SUBURB.csv*, *Census_LGA/2016Census_G01_NSW_LGA.csv*, *Census_LGA/2016Census_G02_NSW_LGA.csv*) were uploaded into Airflow storage bucket, under the directory of */data/airbnb/*(Fig 9).

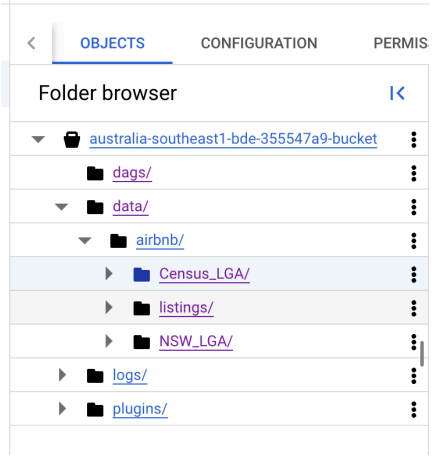

Fig 9. Uploading source files

2. **Set up Bronze schema and tables in Postgres using DBeaver**: The datasets would be extracted with their original forms. All columns of each tables would be represented by text format. Checked if the tables were created successfully (Fig 10).

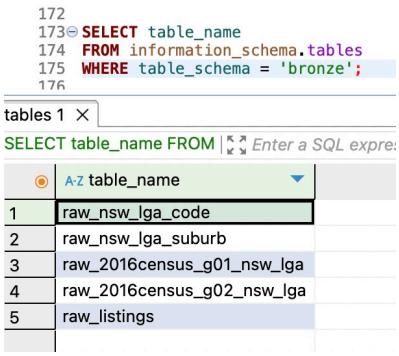

Fig 10. Checking tables

3.2 Build an Airflow DAG for Data Ingestion

Corresponding code source: bde_airbnb_dag.py

This step is to perform data extraction and loading based on the initial datasets. A DAG that read the data from Airflow storage bucket and loads it into raw tables with the bronze schema on Postgres, was built by the following steps:

1. **Import Libraries**: The required libraries regarding data manipulation, airflow configuration, and datetime setting were imported.
2. **DAG Settings**: Firstly, the DAG default arguments (*dag_default_args*) were defined, by specifying the fields like owner, start_date, retries and retry_delay. A DAG was later defined by setting *default_args= dag_default_args* and *schedule_interval=None*.
3. **Load Environment Variables**: The paths for the raw files in Airflow storage bucket would be used through different operators.
4. **Custom Logic for Operator**: Python functions to extract raw data from storage bucket as string format and load them into Postgres in their original forms. It also kept track of each steps with logging package and move the processed files to their archive folders.
5. **DAG Operators and Pipeline Setups**: The Python Operators were set up, specifying the corresponding task_id and python_callable, set *provide_context=True*, so that the function can call the environment variables to retrieve the paths for raw files. Finally, the tasks were chained as the task pipeline for the DAG, and the file was saved as '*bde_airbnb_dag.py'*.
6. **Run the DAG in Airflow**: Uploaded the DAG file into 'dags' folder in Airflow storage bucket and checked the graph of the DAG on Airflow webserver (Fig 11). Then, Triggered the DAG.

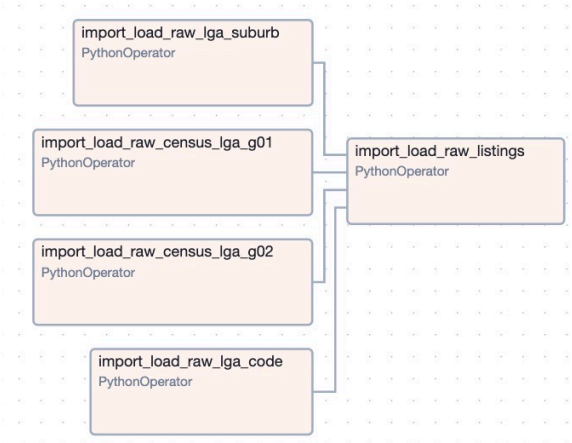

Fig 11. Graph of DAG

7. **Checked the Loaded Data on Postgres**: After the DAG was successfully executed, checked the loaded data on Postgres using DBeaver (Fig 12).

A-Z table_name	123 num_rows
raw_nsw_lga_code	129
raw_nsw_lga_suburb	4,470
raw_2016census_g01_nsw_lga	132
raw_2016census_g02_nsw_lga	132
raw_listings	37,562

Fig 12. Checking data loaded

4. Data Transformation with dbt

4.1 Source Configuration

Corresponding code source: dbt_project/models/sources.yml

Under the models folder in project's working directory, *sources.yml* was created to specify raw data sources and corresponding tables from Postgres.

4.2 Bronze Layer

Corresponding code source: dbt_project/models/bronze

The Bronze Layer under Medallion architecture represents raw data, ingested and stored as close to the original format as possible. These tables were retrieved directly from PostgreSQL and were stored in bronze models as string to ensure the original format: *b_census_g01*, *b_census_g02*, *b_lga_code*, *b_lga_suburb*, *b_listings*.

4.3 Silver Layer

Corresponding code source: dbt_project/models/silver/

The Silver Layer under Medallion architecture is where data undergoes cleaning, transformation, and normalization, creating initial dimension tables and a base facts table. The facts table contains the numerical information of the dataset, and the dimension tables offer descriptive information and context to the numbers from facts table.

The Silver tables and their corresponding transformations:

1. *s_census_g01*: created from *b_census_g01*, with transformations of consistent naming and converting datatypes, and specifying unique key '*lga_code'*.
2. *s_census_g02*: created from *b_census_g02*, with transformations of consistent naming and converting datatypes, and specifying unique key '*lga_code'*.
3. *s_lga*: created from *b_lga_code*, with transformations of consistent naming and converting datatypes, and specifying unique key '*lga_code'*. Additionally, a column '*script_date'* was added to keep track of the time when records was created in the table.

4. *s_lga_suburb*: created from *b_lga_code*. A surrogate key '*suburb_id'*, generated from '*suburb_name'* and '*lga_name',* was added to be the unique key of the table.
5. *s_host*: derived from *b_listings*. It only kept the columns related to host dimension. Columns like '*scraped_date'*, '*host_sinc'e* were formatted as date. Converted '*host_is_superhost'* to binary column. Retrieved '*suburb_id'* by mapping '*host_neighbourhood'* on '*suburb_name'* from *s_suburb*. Handled deduplicates hosts, only keeping the latest entry of *'host_id'*.
6. *s_listing_property*: derived from *b_listings*. It only kept the columns of listing property attributes. The transformations performed in this table include converting and formatting data types, and retrieving '*lga_code'* by mapping '*listing_neighbourhood'* with '*lga_name'* from *s_lga*.
7. *s_facts*: derived from *b_listings*. It only kept the columns of numerical information and reference IDs. The table primarily underwent data types formatting. Besides, a '*month*' column was retrieved from '*scraped_date*', as the focus of this project is observing the monthly changes of the datasets.

4.4 Snapshots

Corresponding code source: dbt_project/snapshots/

In this project, the host and listing property data may change slowly over time, and the Slowly Changing Dimension (SCD) Type 2 technique is an effective method to manage these changes. Specifically, the changes of host and listing property tables can be tracked, along with meta columns '*dbt_valid_from'* and '*dbt_valid_to'* within each snapshot, which can later be used later to track history records. The related snapshot tables are listed below:

1. *host_snapshot*: created from *s_host*, using the 'timestamp' strategy with and '*scraped_date'* as the '*update_at'* field to track changes when '*scraped_date'* updates. A surrogate key, '*host_record_id'* (generated from '*host_id'* and '*month'),* added as the unique key.
2. *listing_property_snapshot*: created from *s_listing_property*, using the same strategy to track the historic changes. A surrogate key '*listin_record_id'*, (generated from '*listing_id'* and '*month'),* acted as the unique key.

4.5 Gold Layer

Corresponding code source: dbt_project/models/gold/

The Gold Layer is designed for business analysis and reporting. It includes **Star Schema** with dimension tables and a facts table, which enables data query, and **Data Mart**, containing views that serve for different business use cases. Moveover, the SCD Type 2 was applied, by retrieving columns '*valid_from'* and '*valid_to*' from '*dbt_valid_from'* and '*dbt_valid_to'* generated by each snapshot.

The Gold tables and their corresponding transformations:

Star Schema:

1. *facts*: directly retrieved from *s_facts*, and the same surrogate key '*listin_record_id'* was generated as the unique key in this table (Fig 13).

Fig 13. Schema of facts table

2. *dim_host*: retrieved from *host_snapshot*, with two columns '*valid_from'* and '*valid_to*' to track historical records of the host table, and a starting point of year 1900 was set for '*valid_from*' to standardize the historical data tracking (Fig 14).

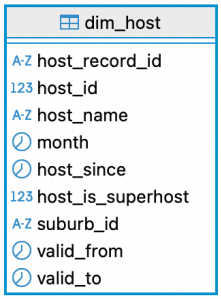

Fig 14. Schema of dim_host table

3. *dim_listing_property*: retrieved from *listing_property_snapshot*, with two columns '*valid_from'* and '*valid_to*' to track historical records of the current table, and a starting point of year 1900 was set for '*valid_from*' to standardize the historical data tracking (Fig 15).

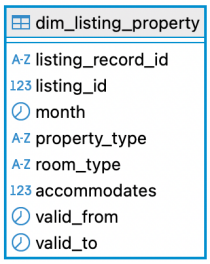

Fig 15. Schema of dim_listing_property table

4. *dim_lga*: directly retrieved from *s_lga* (Fig 16).

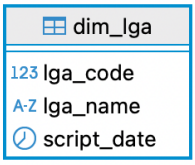

Fig 16. Schema of dim_lga table

5. *dim_suburb*: directly retrieved from *s_suburb* (Fig 17).

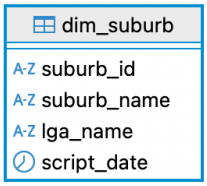

Fig 17. Schema of dim_suburb table

Mart

1. *census_g01*: retrieved from *s_census_g01*, providing LGA-level demographic information.
2. *census_g02*: derived from *s_census_g02*, providing LGA-level community information.
3. *listing_neighbourhood*: This view aims to generate insights on a monthly basis for each listing neighborhood, with the metrics active listings rates, pricing

trends, distinct hosts, super-host rates, review scores, listing activity changes, total stays, and estimated revenue (Fig 17).

listing_n	mont	act_list	min_act_p	max_act_p	med_act_p	avg_act_p	total_h	sup_h	sup_host_	percent_chang
Bayside	2020-05-01	100	0	15,367	86	182.0148514851	1,220	137	11.2295081967	[NULL]
Bayside	2020-06-01	100	0	14,478	84	178.7433795712	1,201	136	11.3238967527	-1.8564356436
Bayside	2020-07-01	100	0	14,376	85	192.5425844347	1,041	138	13.2564841499	-14.1235813367
Bayside	2020-08-01	100	17	2,904	79	110.6945144928	1,051	136	12.9400570885	1.3215859031
Bayside	2020-09-01	100	17	2,903.71	78	107.7148004627	1,157	140	12.1002592913	11.6666666667
Bayside	2020-10-01	100	17	2,904	78	108.7817448777	1,141	145	12.708150745	-1.8170019468
Bayside	2020-11-01	100	14	2,904	78	112.5416384563	1,115	144	12.9147982063	-2.3793787178
Bayside	2020-12-01	100	15	2,904	80	116.8160951105	1,128	141	12.5	1.0832769127
Bayside	2021-01-01	100	15	2,904	80	128.8322237017	1,128	146	12.9432624113	0.6028131279
Bayside	2021-02-01	0	15	2,904	80	124.7214334009	1,111	146	13.1413141314	-1.5312916112
Bayside	2021-03-01	0	15	2,904	80	126.8485477178	1,106	143	12.9294755877	-2.2312373225
Bayside	2021-04-01	0	15	2,904	80	120.0456460674	1,087	144	13.2474701012	-1.5214384509
Blacktown	2020-05-01	100	23	1,099	63	90.3018867925	239	31	12.9707112971	[NULL]
Blacktown	2020-06-01	100	23	1,099	62	90.674267101	234	32	13.6752136752	-3.4591194969
Blacktown	2020-07-01	100	23	2,164	66	123.6863468635	212	31	14.6226415094	-11.7263843648
Blacktown	2020-08-01	100	20	2,000	60	98.1520567376	216	29	13.4259259259	4.0590405904
Blacktown	2020-09-01	100	20	2,000	60	92.3031974922	220	30	13.6363636364	2.4822695035
Blacktown	2020-10-01	100	18	2,000	64	98.9654448399	216	30	13.8888888889	-2.76816609

Fig 17. Screenshot of listing_neighbourhood view

4. *property_type*: This view aims to generate insights on a monthly basis in terms of listing property attributes, with the metrics active listings rates, pricing trends, distinct hosts, super-host rates, review scores, listing activity changes, total stays, and estimated revenue (Fig 18).

property_type	room_type	accommodates	month	act_listing_rate	min_act_price	max_act_price	med_act_price	avg_act_price
Aparthotel	Entire home/apt	2	2020-05-01	100	165	165	165	165
Aparthotel	Entire home/apt	2	2020-06-01	100	164	164	164	164
Aparthotel	Entire home/apt	2	2020-08-01	100	101	101	101	101
Aparthotel	Entire home/apt	2	2020-09-01	100	106.43	107	106.715	106.715
Aparthotel	Entire home/apt	2	2020-10-01	100	120	120	120	120
Aparthotel	Entire home/apt	2	2020-11-01	100	126	126	126	126
Aparthotel	Entire home/apt	2	2020-12-01	100	164	164	164	164
Aparthotel	Entire home/apt	2	2021-01-01	100	133	133	133	133
Aparthotel	Entire home/apt	2	2021-02-01	100	139	139	139	139
Aparthotel	Entire home/apt	2	2021-03-01	100	206	206	206	206
Aparthotel	Entire home/apt	2	2021-04-01	100	161	161	161	161
Aparthotel	Entire home/apt	3	2020-06-01	100	99	109	104	104
Aparthotel	Entire home/apt	3	2020-07-01	100	98	108	103	103
Aparthotel	Entire home/apt	3	2020-08-01	100	54	56	54	54.3333333333
Aparthotel	Entire home/apt	3	2020-09-01	100	45	55	52.5	51.6666666667
Aparthotel	Entire home/apt	3	2020-10-01	100	56	113	58.29	69.658
Aparthotel	Entire home/apt	3	2020-11-01	100	64	69	65	65.75
Aparthotel	Entire home/apt	3	2020-12-01	100	102	156	114	121.5

Fig 18. Screenshot of property_type view

5. *host_neighbourhood*: This view aims to generate insights on a monthly basis for each host neighborhood, with the metrics distinct hosts, total estimated revenue and estimated revenue per host (Fig 19).

host_neighbourhoo	month	host_cnt	total_estimated_revenue	estimated_revenue_per_
ARMIDALE REGIONAL	2020-05-01	1	1,800	1,800
ARMIDALE REGIONAL	2020-06-01	1	3,600	3,600
ARMIDALE REGIONAL	2020-07-01	1	5,400	5,400
ARMIDALE REGIONAL	2020-08-01	1	7,200	7,200
ARMIDALE REGIONAL	2020-09-01	1	9,000	9,000
ARMIDALE REGIONAL	2020-10-01	2	12,480	6,240
ARMIDALE REGIONAL	2020-11-01	2	14,840	7,420
ARMIDALE REGIONAL	2020-12-01	2	31,788	15,894
ARMIDALE REGIONAL	2021-01-01	2	25,400	12,700
ARMIDALE REGIONAL	2021-02-01	2	18,700	9,350
ARMIDALE REGIONAL	2021-03-01	2	37,800	18,900
ARMIDALE REGIONAL	2021-04-01	2	42,600	21,300
BATHURST REGIONAL	2020-05-01	10	188,177	18,817.7
BATHURST REGIONAL	2020-06-01	9	328,206	36,467.3333333333
BATHURST REGIONAL	2020-07-01	9	490,412	54,490.2222222222
BATHURST REGIONAL	2020-08-01	9	1,013,901	112,655.6666666667
BATHURST REGIONAL	2020-09-01	9	1,604,481	178,275.6666666667

Fig 19. Screenshot of host_neighbourhood view

4.6 Deploy the dbt Job

After designing the data warehouse in development environment, the project can be deployed to the production:

1. Set up a production environment (deployment type as Production), and configured the connection and deployment credentials as same with development environment, shown as Fig 20.

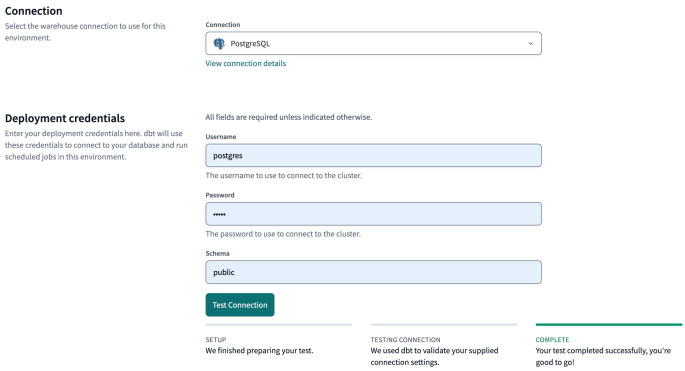

Fig 20. Production environment connection configurations

2. Created a job for production (Deploy >> Jobs >> Create job >> Deploy job) to execute the models defined.
3. Triggered the job manually to test and validate the transformations, and retrieved the required variables (Account ID, Job ID, URL from API trigger, and API token from Account settings >> API tokens >> Personal tokens >> Create personal access token) for Airflow to set up dbt job task.

5. End-to-End Orchestration with Airflow

Corresponding code source: bde_airbnb_dag.py

5.1 Configuration with Airflow

1. **Intergrate dbt job with Airflow**: Configured the following variables retrieved from dbt Cloud in Airflow (Fig 21). This ensures Airflow to authentic and communicate with dbt Cloud, then trigger and monitor dbt jobs directly within Airflow.

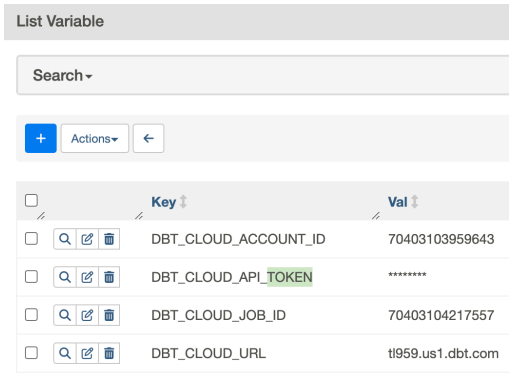

Fig 21. List of added variables in Airflow

2. **Set up a DAG task to trigger dbt job**: Defined a task in Airflow DAG and reference the required variables to trigger the dbt Cloud jobs via API token. The task function would later be used as the transformation part of the pipeline.

5.2 ELT Pipeline Orchestration

The ELT end-to-end orchestration with Airflow can be explained by Fig 22. The task regarding reference datasets were set to run initially, after this, ensuring all necessary mapping tables are available before processing the Airbnb listings data. Once the reference datasets were in place, listings dataset would be extracted and loaded into Postgres, then a dbt job was triggered to perform predefined transformations, processing data through the bronze, silver and gold models. The process was iterated monthly in chronological order. This setup maintains data integrity through the pipeline, and automates the end-to-end ELT workflow from ingestion through transformation with PostgreSQL and dbt Cloud.

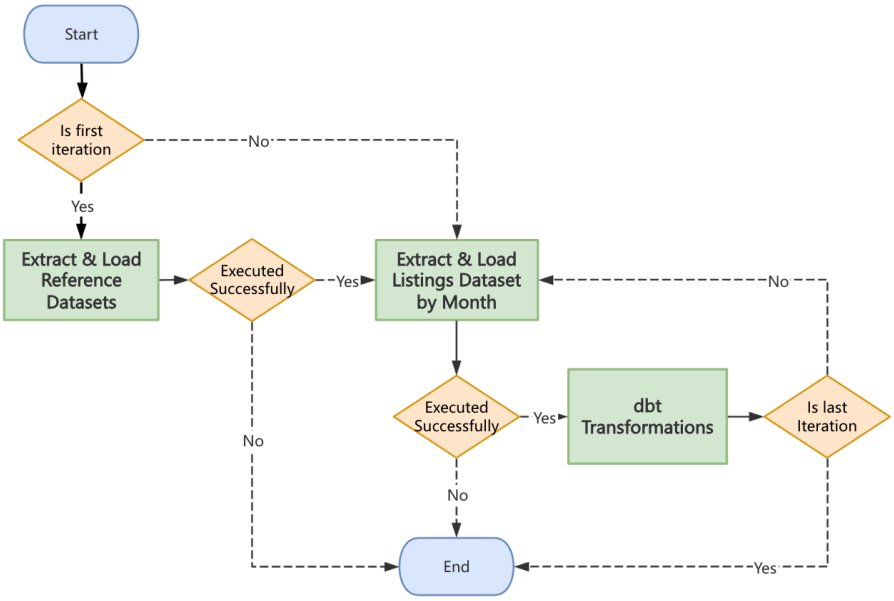

Fig 22. FlowChart of the pipeline

6. Ad-hoc Analysis

Corresponding code source: part_4.sql

1. Explored the demographic differences between the top 3 and lowest 3 performing LGAs in terms of estimated revenue per active listing over the last 12 months. Key insights (Fig 23) indicated that high-performing LGAs tent to have an older demographic, smaller household sizes, and a higher percentage of English-only speakers at home, and a greater proportion of residents with higher education (completion of Year 12 or above).

lga_name	est_rev_per_li	rk_top	rk_bottom	tot_p_p	average_household_size	percent_lang_home_eng	median_age_	percent_65_74yr	percent_75
Mosman	87,455.0261326861	1	29	28,475	2.4	77.9280070237	42	10.5355575066	5.415276
Northern Beaches	70,827.2633865763	2	28	252,878	2.7	79.7902545892	40	8.8738443044	5.171663
Woollahra	61,034.022830385	3	27	54,240	2.3	75.6544985251	39	10.1972713864	5.495943
Burwood	12,615.0624804178	27	3	36,809	2.9	31.4135129995	33	6.7483495884	4.884674
Blacktown	11,356.4528967254	28	2	336,962	3.2	53.6932947929	33	6.3912251233	2.870946
Fairfield	9,710.3825694444	29	1	198,817	3.3	24.8233299969	36	7.6718791653	4.326088

Fig 23. Screenshot of analysis result 1

2. Explored the correlation between the estimated revenue per active listing in a neighbourhood and its median age. As shown in Fig 24, there is correlation between the median age of a community with its listing revenue, and high performing neighbourhoods tend to have an older population.

lga_name	est_rev_per_listing	rk_top	median_age_persons
Mosman	87,455.0261326861	1	42
Northern Beaches	70,827.2633865763	2	40
Woollahra	61,034.022830385	3	39
Waverley	52,727.115652032	4	35
Hunters Hill	47,303.2228571429	5	43
Sutherland Shire	40,519.0586956522	6	40
Randwick	39,427.479227446	7	34
Lane Cove	38,430.7299014778	8	36
North Sydney	37,664.7732232893	9	37
Willoughby	37,007.5185991678	10	37
Inner West	32,969.6710578569	11	36
Sydney	32,509.5866366571	12	32
Cumberland	27,623.6309987358	13	32
Canada Bay	25,721.4296006944	14	36
Hornsby	22,224.5413993711	15	40
The Hills Shire	20,429.1702823529	16	38
Bayside	20,279.2964443378	17	35
Penrith	20,004.7930241935	18	34
Campbelltown	19,394.6398648649	19	34
Georges River	18,139.0148015123	20	37
Ryde	17,991.9995865922	21	36
Camden	15,568.7784507042	22	33
Liverpool	15,551.3397391304	23	33
Parramatta	14,365.3152231604	24	34
Canterbury-Banks	13,938.2752313625	25	35
Strathfield	12,740.3723048327	26	32
Burwood	12,615.0624804178	27	33
Blacktown	11,356.4528967254	28	33
Fairfield	9,710.3825694444	29	36

Fig 24. Screenshot of analysis result 2

3. Analyzed the popularity of different type of listing in top-5 performing neighbourhoods by total number of stays. From Fig 25, the most favored type of

listing was entire apartment with a capacity of 2, receiving more than 1,950,000 stays over 12 months.

property_type	room_type	accommodates	num_stays
Entire apartment	Entire home/apt	2	1,950,190
Entire apartment	Entire home/apt	4	1,909,396
Private room in apartment	Private room	2	1,445,948
Apartment	Entire home/apt	2	1,345,656
Apartment	Entire home/apt	4	1,330,965
Apartment	Private room	2	1,022,974
Entire house	Entire home/apt	8	645,525
Entire house	Entire home/apt	6	636,162
Entire apartment	Entire home/apt	3	511,354
Private room in apartment	Private room	1	491,256
Private room in house	Private room	2	482,428
House	Entire home/apt	8	438,039
House	Entire home/apt	6	436,661
Apartment	Entire home/apt	3	351,197
Apartment	Private room	1	350,250
House	Private room	2	341,583
Entire apartment	Entire home/apt	6	336,921
Entire apartment	Entire home/apt	5	297,407
Entire house	Entire home/apt	4	295,005
Entire house	Entire home/apt	7	263,465
Apartment	Entire home/apt	6	231,049

Fig 25. Screenshot of analysis result 3

4. Analyzed how their listings distribution across Sydney LGAs for host with multiple listings. According to Fig 26, findings indicate that host with less listings tend to have concentrate them within the same LGA. For host with 2 or 3 listings, about 80% kept their properties within the same aera.

num_listing_group	total_host	percent_same_lga
a.listing_2	3,232	82.301980198
b.listing_3	827	79.2019347037
c.listing_4_5	505	67.7227722772
d.listing_5_9	284	59.8591549296
e.listing_10_29	184	40.7608695652
f.listing_30ov	63	12.6984126984

Fig 26. Screenshot of analysis result 4

5. Compared Airbnb listing revenue to property mortgage repayment costs across Sydney's LGAs over 12 months to identify where hosts are most likely able to cover mortgage costs. From Fig 27, only a portion of hosts could fully cover their mortgage repayments with listing revenue, with Northern Beaches having the highest percentage—over 67% of hosts able to meet these costs through their listings.

A-Z lga_name	123 total_host	123 percent_h_c_rep
Northern Beaches	4,223	67.511247928
Mosman	340	62.0588235294
Sutherland Shire	492	59.5528455285
Waverley	4,119	56.1544064093
North Sydney	958	55.4279749478
Sydney	5,576	55.0035868006
Randwick	2,434	54.1906327034
Woollahra	1,262	49.6830427892
Hunters Hill	55	49.0909090909
Inner West	1,888	48.3050847458
Lane Cove	228	48.2456140351
Willoughby	352	43.75
Canada Bay	329	41.9452887538
Cumberland	337	41.2462908012
Hornsby	319	37.9310344828
Penrith	111	36.036036036
Camden	37	35.1351351351
Ryde	414	33.3333333333
Bayside	1,058	32.0415879017
The Hills Shire	230	31.7391304348
Canterbury-Bankstown	420	31.1904761905
Burwood	139	29.4964028777
Parramatta	377	29.4429708223
Georges River	260	28.0769230769
Liverpool	93	27.9569892473
Campbelltown	54	27.7777777778
Strathfield	127	27.5590551181
Fairfield	40	17.5
Blacktown	213	16.9014084507

Fig 27. Screenshot of analysis result 5

7. Conclusion

This project has illustrated the orchestration of an end-to-end ELT pipeline by Apache Airflow, with PostgreSQL for data storage and dbt for data transformation. This approach generated a well-structured data warehouse, focusing on Airbnb listings data in Sydney, with Medallion architecture, where bronze, silver, gold layers were built and Slowly Changing Dimensions Type 2 technique was applied. The orchestration also ensures the data pipeline to run smoothly and sequentially, with build-in and customized automation and fault-tolerance.

The ad-hoc analysis based on the constructed data warehouse, indicated key insights into the Sydney Airbnb market. It showed that top-performing areas often have older residents and higher education levels, along with a link between median age and listing revenue of a neighbourhood. The analysis also identified the entire apartment with capacity of 2 being the most popular listing type. Additionally, host with multiple listings often concentrate their properties in same area, while for those with single listing, face challenges in covering the property mortgage repayment with their Airbnb listing revenue.

8. Appendix

1. Assignment brief and source files used in this project are part of the learning materials from Big Data Engineering of University of Technology Sydney (UTS).
2. Data Source for Sydney Listings Datasets: Inside Airbnb.
3. Data Source for LGA Reference Datasets: Australian Bureau of Statistics.