BUILDING ELT DATA PIPELINES WITH AIRFLOW

Simon Lim

Big Data Engineering

1 CONTEXTS AND OBJECTIVES OF THE PROJECT

Airbnb is an online-based marketing company that provides services to people seeking accommodation (Airbnb guests) to people looking for renting their properties (Airbnb hosts). Airbnb offers a variety of the rental property's options, including apartments, homes, boats, townhouses or private rooms. While Airbnb records millions of various information in 191 countries, including density of rentals across regions, price variations across rentals and host-guest interactions (e.g., number of reviews). In this project, only data in Sydney are used with specific date range from May 2020 to April 2021

In addition, the Census of Population and Housing (Census) is Australia's largest statistical data collection that is undertaken and managed by the Australian Bureau of Statistics (ABS). Census aims at accurately collecting data regarding the key characteristics of Australians in each region. The data of key characteristics generated from Census will be used along with Airbnb data, in order to extract insights and further analyse and answer business questions.

In terms of the following contexts, the objective of the project is to build production-ready data pipelines with Airflow, Data Build Tool and Postgres SQL. The procedure of the project includes building production-ready data pipelines with Airflow, processing and cleaning data using DBT ELT pipelines, design the architecture of a data warehouse on Postgres and finally analyse and extract valuable insights. In this project, Airflow is used to load data, DBT is used to design a data warehouse and Postgres is used to analyse and answer business questions.

2 PRESENTATION OF DATASET & PREDICTIVE ISSUES

There are three kinds of datasets generated from Airbnb and Census, including 12 months of Airbnb listing data for Sydney, G01("Selected Person Characteristics by Sex") and G02 ("Selected Medians and Averages") Census data and lastly a dataset, containing LGAs code, LGA names and suburb names, which will help to join other datasets in later stage. Airbnb listing dataset contains information, such as host_name, host_since, host_neighbouthood, listing_neighbourhood, property_type, room_type, accommodates, price, availabilities and number and scores of reviews. Perhaps, it is assumed that Airbnb listing dataset can be snapshotted into three dimensions, including host, property type and room type. Furthermore, Census tables (i.e., G01 and G02) and LGAs tables can be also used as dimensions tables. Lastly, Airbnb listing dataset contains some null values and data quality issues in some columns (e.g., string values in numeric column). In this regard, it is predicted that when loading data in Airflow, wrong data type of column can potentially raise an error. Therefore, appropriate matched data type for each column would be necessary when loading data in Airflow.

3 PIPELINE OF THE PROJECT

PART 0. Setting IP Address on Airflow and Postgres

- Public and private IP address were obtained and applied into Airflow (private) and Postgres (public).

PART 1. Data Loading into Postgres using Airflow

1. Airbnb listing dataset, LGA datasets and Census datasets were uploaded into Airflow storage bucket.
2. A raw schema and the relevant raw tables, which represent Airbnb, Census and LGA datasets, were created on Postgres. The relevant raw tables include raw.listings, raw.nsw_lga, raw.nsw_suburb, raw.g01, raw.g02.
3. An Airflow Dag file, that was used to read the datasets and insert the raw data into the raw schema, was created. The names of columns in raw tables on Postgres and column names in Dag file were identical with appropriate data types based on values of each column. The Dag file was then uploaded into Airflow storage bucket, which was automatically processed in Airflow and updated on Postgres.

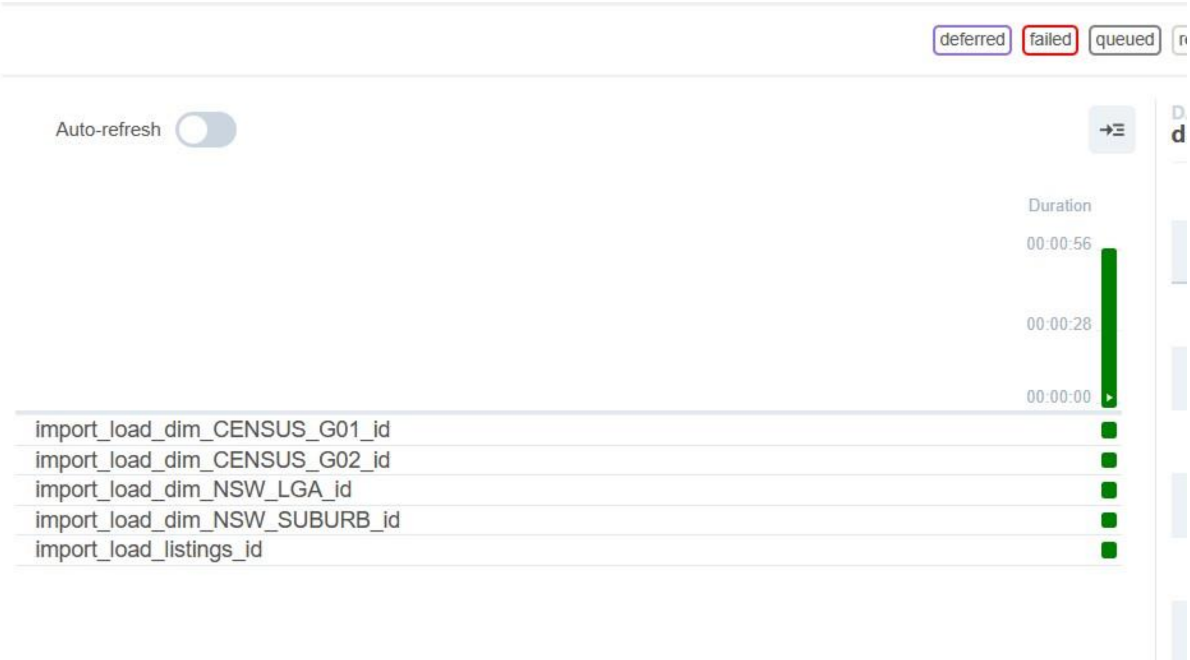

Figure 1. Loading Data using Airflow Dag

PART 1. Issues in Data Loading into Postgres using Airflow (Solved/Unsolved)

- As discussed in presentation of dataset stage, there were some data quality issues in listing dataset. For example, there was 't' in 'price' column, which was supposed to include only numeric values. I decided to set its data type as varchar at this stage but would essentially need to be converted into float type in data transformation stage (solved).

		F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
15104		#######	f		Sydney	Private roc	Private roc	1	40	t		0	4	100	10	10	10	10
15105		#######	f	Turrella	Bayside	Private roc	Private roc	2	65	t		0	2	100	10	10	10	10
15106	Shi	#######	f		Northern E	Private roc	Private roc	2	79	t		29	0					
15107	J 8	#######	t	Lidcombe	Bayside	Entire apar	Entire hor	4	118	t		0	105	96	9	9	10	10
15108		#######	f	Beaconsfie	Sydney	Private roc	Private roc	1	62	t		0	4	85	9	8	9	10
15109		#######	t		Waverley	Private roc	Private roc	2	75	t		27	15	100	10	10	10	10
15110		#######	f		Penrith	Private roc	Private roc	2	55	t		30	0					
15111		#######	f		Sydney	Shared roc	Shared roc	1	30	t		0	0					
15112		#######	f	Randwick	Randwick	Private roc	Private roc	1	45	t		0	2	100	9	9	10	10
15113		#######	f		Ryde	Entire apar	Entire hor	4	180	t		0	8	98	10	10	10	10
15114	/2(	f		Bayside	Entire hou	Entire hor	4	150	t	0	4	100	10	10	10	10	10	
15115		#######	f		Sydney	Private roc	Private roc	2	120	t		0	59	97	10	10	10	10
15116	ro	#######	f	Bondi Bea	Waverley	Entire apar	Entire hor	3	75	t		0	4	93	9	5	9	9
15117		#######	f	Camperdo	Inner Wes	Entire apar	Entire hor	1	50	t		0	0					
15118		#######	f		Georges R	Private roc	Private roc	2	50	t		0	0					
15119		#######	f		North Syd	Private roc	Private roc	2	34	t		0	0					
15120	a	#######	f	Coogee	Randwick	Entire apar	Entire hor	2	130	t		0	1	80	10	8	10	8
15121		#######	f		Cumberlar	Shared roc	Shared roc	1	70	t		0	0					
15122		#######	f		Cumberlar	Shared roc	Shared roc	1	15	t		0	0					
15123		#######	f	Mosman	Mosman	Private roc	Private roc	1	50	t		15	7	100	10	10	10	10
15124		#######	t		Inner Wes	Entire gue	Entire hor	2	108	t		10	73	96	10	10	10	10
15125		#######	f	Stanmore	Inner Wes	Private roc	Private roc	1	30	t		0	0					

Figure 2. Data Quality Issue in listing dataset

PART 2. Design a Data Warehouse using DBT

- In order to design a data warehouse on Postgres, four different layers were created, including Raw, Staging, Warehouse and Datamart.

1. **Raw/Snapshot (table)**

- Five raw tables were already created in Airflow loading stage, including raw.listings, raw.nsw_lag, raw.nsw_suburb, raw.g01, raw.g02.
- Three different dimensions from raw.listings were snapshotted with each dimension representing host, property_type and room_type. Mutually, 'host_id' was used as a unique key for all three snapshots and 'scraped_date' was used as an updated date.

2. **Staging (view)**

- The goal of staging was to clean, transform and rename data from raw and snapshot.
- 5 raw tables and 3 snapshot tables were transformed into staging views with names of stg_G01, stg_G02, stg_Host, stg_LGA, stg_listing, stg_Property, stg_room and stg_Suburb.

a. **stg_G01 and stg_G02**: A column, 'lga_code_2016' was transformed from data format of 'LGA*****' to '*****' using SUBSTRING function and then its column name was renamed as 'lga_code' with data type integer.

b. **Stg_LGA**: There was no change from a raw table.

c. **Stg_Suburb**: All columns ('lga_name' and 'suburb_name') were in capital letters. Hence, they were appropriately changed to match the format with 'lga_name' in stg_LGA. Only first letter in each word was set as a capital letter and other letters were fixed as lower cases.

```
{{
    config(
        unique_key='lga_name',
        materialized='view'
    )
}}

with

source  as (

    select * from {{source('raw','nsw_suburb')}}

),

renamed as (
    select
        CONCAT(UPPER(SUBSTRING(lga_name, 1, 1)), LOWER(SUBSTRING(lga_name, 2))) AS lga_name,
        CONCAT(UPPER(SUBSTRING(suburb_name, 1, 1)), LOWER(SUBSTRING(suburb_name, 2))) AS suburb_name
    from source
)

select * from renamed
```

Figure 3. Modifying Data Format in stg_Suburb

d. **Stg_Host**: There were some null values in several columns, including ‘host_name’, ‘host_since’, ‘host_is_superhost’ and ‘host_neighbourhood’. Null values were cleaned by filling them with default values, ‘unknown’ for string values, 0 for numeric values and ‘false’ for Boolean values. Also, date columns such as ‘host_since’, ‘scraped_date’, ‘dbt_valid_from’ and ‘dbt_valid_to’ were converted to date type.

e. **Stg_Property:** Some data type transformations were performed at this stage. For example, ‘price’ column was converted from varchar type to float type and ‘has_availablity’ was converted to Boolean type and ‘scraped_date’ was converted to date type.

f. **Stg_room:** There were some null values in integer columns, including ‘review_scores_rating’, ‘review_scores_accuracy’, ‘review_scores_cleanliness’, ‘review_scores_checkin’, ‘review_scores_communication’ and ‘review_scores_value’. Null values were filled with 0. Also, date columns were converted to date type like other snapshots.

g. **Stg_listing:** This staging view included all the cleaned and transformed columns from stg_host, stg_property and stg_room. It would be used as a fact table in warehouse stage.

```sql
renamed as (
    SELECT
    listing_id,
    SCRAPED_DATE::date as SCRAPED_DATE,
    HOST_ID,
    CASE
    WHEN HOST_NAME = 'Na' THEN 'unknown'
    WHEN HOST_NAME = 'NaN' Then 'unknown'
    ELSE HOST_NAME
    END AS HOST_NAME,
    CASE WHEN HOST_SINCE = 'NaN' THEN '1900-01-01'::date ELSE to_date(HOST_SINCE, 'DD/MM/YYYY') END AS HOST_SINCE,
    CASE WHEN HOST_IS_SUPERHOST = 'NaN' THEN 'false'::boolean ELSE HOST_IS_SUPERHOST::boolean END AS HOST_IS_SUPERHOST,
    CASE WHEN HOST_NEIGHBOURHOOD = 'NaN' THEN 'unknown' ELSE HOST_NEIGHBOURHOOD END AS HOST_NEIGHBOURHOOD,
    LISTING_NEIGHBOURHOOD,
    PROPERTY_TYPE,
    ROOM_TYPE,
    ACCOMMODATES,
    PRICE::FLOAT as PRICE,
    HAS_AVAILABILITY::boolean,
    AVAILABILITY_30,
    NUMBER_OF_REVIEWS,
    CASE WHEN REVIEW_SCORES_RATING = 'NaN' THEN 0 ELSE REVIEW_SCORES_RATING END AS REVIEW_SCORES_RATING,
    CASE WHEN REVIEW_SCORES_ACCURACY = 'NaN' THEN 0 ELSE REVIEW_SCORES_ACCURACY END AS REVIEW_SCORES_ACCURACY,
    CASE WHEN REVIEW_SCORES_CLEANLINESS = 'NaN' THEN 0 ELSE REVIEW_SCORES_CLEANLINESS END AS REVIEW_SCORES_CLEANLINESS,
    CASE WHEN REVIEW_SCORES_CHECKIN = 'NaN' THEN 0 ELSE REVIEW_SCORES_CHECKIN END AS REVIEW_SCORES_CHECKIN,
    CASE WHEN REVIEW_SCORES_COMMUNICATION = 'NaN' THEN 0 ELSE REVIEW_SCORES_COMMUNICATION END AS REVIEW_SCORES_COMMUNICATION,
    CASE WHEN REVIEW_SCORES_VALUE = 'NaN' THEN 0 ELSE REVIEW_SCORES_VALUE END AS REVIEW_SCORES_VALUE
    FROM source
)

select * from renamed
```

Figure 4. Data Cleaning and Transformation in stg_listing

3. Warehouse (table)

- While all of dimension staging views were directly transformed into warehouse table without any change, this stage focused on a fact table, which is fact_listings. All the columns in the fact table were brought from corresponding dimension tables. Any value that was not included in corresponding dimension tables was set as a default values (i.e., ‘unknown’, 0 or ‘false’).

```sql
with check_dimensions as
(select
    listing_id,
    SCRAPED_DATE,
    case when LISTING_NEIGHBOURHOOD in (select LISTING_NEIGHBOURHOOD from {{ ref('stg_Host') }}) then LISTING_NEIGHBOURHOOD else 'unknown' end as LISTING_NEIGHBOURHOOD,
    case when host_id in (select distinct host_id from {{ ref('stg_Host') }}) then host_id else 0 end as host_id,
    case when HOST_NAME in (select HOST_NAME from {{ ref('stg_Host') }}) then HOST_NAME else 'unknown' end as HOST_NAME,
    case when HOST_SINCE in (select HOST_SINCE from {{ ref('stg_Host') }}) then HOST_SINCE else '1900-01-01' end as HOST_SINCE,
    case when HOST_IS_SUPERHOST in (select HOST_IS_SUPERHOST from {{ ref('stg_Host') }}) then HOST_IS_SUPERHOST else 'false' end as HOST_IS_SUPERHOST,
    case when HOST_NEIGHBOURHOOD in (select HOST_NEIGHBOURHOOD from {{ ref('stg_Host') }}) then HOST_NEIGHBOURHOOD else 'unknown' end as HOST_NEIGHBOURHOOD,
    case when PROPERTY_TYPE in (select PROPERTY_TYPE from {{ ref('stg_Property') }}) then PROPERTY_TYPE else 'unknown' end as PROPERTY_TYPE,
    case when ROOM_TYPE in (select ROOM_TYPE from {{ ref('stg_room') }}) then ROOM_TYPE else 'unknown' end as ROOM_TYPE,
    case when price in (select price from {{ ref('stg_Property') }}) then price else 0 end as price,
    case when ACCOMMODATES in (select ACCOMMODATES from {{ ref('stg_Property') }}) then ACCOMMODATES else 0 end as ACCOMMODATES,
    case when HAS_AVAILABILITY in (select HAS_AVAILABILITY from {{ ref('stg_Property') }}) then HAS_AVAILABILITY else 'false' end as HAS_AVAILABILITY,
    case when AVAILABILITY_30 in (select AVAILABILITY_30 from {{ ref('stg_Property') }}) then AVAILABILITY_30 else 0 end as AVAILABILITY_30,
    case when NUMBER_OF_REVIEWS in (select NUMBER_OF_REVIEWS from {{ ref('stg_room') }}) then NUMBER_OF_REVIEWS else 0 end as NUMBER_OF_REVIEWS,
    case when REVIEW_SCORES_RATING in (select REVIEW_SCORES_RATING from {{ ref('stg_room') }}) then REVIEW_SCORES_RATING else 0 end as REVIEW_SCORES_RATING
from {{ ref('stg_listing') }})
```

Figure 5. Checking Dimensions in fact_listings table

- Lastly, the fact table was joined with few dimension tables, including stg_LGA (on listing_neighbourhood, host_neighbourhood = lga_name) and stg_Suburb (on host_neighbourhood = lga_name)
- Particularly, ‘host_neighbourhood’ included values that are in either ‘lga_name’ or ‘suburb_name’. Hence, both ‘lga_name’ and ‘suburb_name’ were used to match values in ‘host_neighbourhood’.

```
select
    a.listing_id,
    a.SCRAPED_DATE as date,
    a.LISTING_NEIGHBOURHOOD,
    b.lga_code as LISTING_NEIGHBOURHOOD_LGA_CODE,
    a.host_id,
    a.HOST_NAME,
    a.HOST_SINCE,
    a.HOST_IS_SUPERHOST,
    a.HOST_NEIGHBOURHOOD,
    c.lga_code as HOST_NEIGHBOURHOOD_LGA_CODE,
    case when a.HOST_NEIGHBOURHOOD in (select suburb_name from {{ ref('stg_Suburb') }}) then a.LISTING_NEIGHBOURHOOD
    else c.lga_name
    end as HOST_NEIGHBOURHOOD_LGA_NAME,
    a.PROPERTY_TYPE,
    a.ROOM_TYPE,
    a.price,
    a.ACCOMMODATES,
    a.HAS_AVAILABILITY,
    a.AVAILABILITY_30,
    a.NUMBER_OF_REVIEWS,
    a.REVIEW_SCORES_RATING
from check_dimensions a
left join {{ ref('stg_LGA') }} b  on a.LISTING_NEIGHBOURHOOD = b.lga_name
left join {{ ref('stg_LGA') }} c  on a.HOST_NEIGHBOURHOOD = c.lga_name
left join {{ ref('stg_Suburb') }} d  on a.HOST_NEIGHBOURHOOD = d.lga_name
```

Figure 6. The Final Fact_Listing Table

4. Datamart (view)

- For the datamart, 3 different views were created, including dm_listing_neighbourhood, dm_property_type and dm_host_neighbourhood.
- The fact listing table was mainly used to create the datamart views.

a. Dm_listing_neighbourhood

In this view, for ‘listing_neighbourhood’ and ‘month/year’ (grouped by and ordered by), the following columns and values were produced:

- active listings
- active listings rate

- minimum, maximum, median and average of price for active listings
- superhost rate
- average of review scores rating for active listings
- percentage change for active listings
- percentage change for inactive listings
- total number of stays
- average estimated revenue per active listings

	listing_neighbourhood	month_year	active_listings	active_listings_rate	min_price_active_listings	max_price_active_listing
1	Bayside	11/2020	7	100	55	235
2	Bayside	1/2021	1,502	100	0	2,904
3	Bayside	12/2020	6	100	20	250
4	Bayside	2/2021	21	100	28	211
5	Bayside	3/2021	16	100	32	350
6	Bayside	4/2021	45	100	25	457
7	Bayside	5/2020	351	100	0	2,090
8	Bayside	6/2020	21	100	30	801
9	Bayside	7/2020	32	100	0	174
10	Bayside	8/2020	20	100	22	380
11	Bayside	9/2020	15	100	25	120
12	Blacktown	11/2020	4	100	28	176
13	Blacktown	1/2021	285	100	0	2,000
14	Blacktown	2/2021	5	100	25	99
15	Blacktown	3/2021	4	100	50	323
16	Blacktown	4/2021	11	100	38	270
17	Blacktown	5/2020	76	100	31	881
18	Blacktown	6/2020	1	100	75	75
19	Blacktown	7/2020	6	100	0	150
20	Blacktown	8/2020	2	100	29	35
21	Blacktown	9/2020	1	100	39	39
22	Burwood	11/2020	5	100	59	116
23	Burwood	1/2021	251	100	16	530
24	Burwood	12/2020	2	100	77	109
25	Burwood	2/2021	5	100	33	165
26	Burwood	3/2021	2	100	114	191
27	Burwood	4/2021	11	100	37	260
28	Burwood	5/2020	66	100	0	250
29	Burwood	6/2020	4	100	48	120

Figure 7. dm_listing_neighbourhood table

b. dm_property_type

In this view, for 'property_type', 'room type', 'acoomodates' and 'month/year' (grouped by and ordered by), the following columns and values were produced:

- active listings
- active listings rate
- minimum, maximum, median and average of price for active listings
- superhost rate
- average of review scores rating for active listings

- percentage change for active listings
- percentage change for inactive listings
- total number of stays
- average estimated revenue per active listings

	property_type	room_type	accommodates	month_year	active_listings	active_listings_rate	min_price_active_li
1	Apartment	Entire home/apt	1	5/2020	1	100	
2	Apartment	Entire home/apt	1	5/2020	1	100	
3	Apartment	Entire home/apt	1	5/2020	2	100	
4	Apartment	Entire home/apt	1	5/2020	1	100	
5	Apartment	Entire home/apt	1	5/2020	2	100	
6	Apartment	Entire home/apt	1	5/2020	1	100	
7	Apartment	Entire home/apt	1	5/2020	4	100	
8	Apartment	Entire home/apt	1	5/2020	4	100	
9	Apartment	Entire home/apt	1	6/2020	1	100	
10	Apartment	Entire home/apt	1	6/2020	1	100	
11	Apartment	Entire home/apt	1	7/2020	2	100	
12	Apartment	Entire home/apt	2	5/2020	23	100	
13	Apartment	Entire home/apt	2	5/2020	6	100	
14	Apartment	Entire home/apt	2	5/2020	14	100	
15	Apartment	Entire home/apt	2	5/2020	4	100	
16	Apartment	Entire home/apt	2	5/2020	11	100	
17	Apartment	Entire home/apt	2	5/2020	2	100	
18	Apartment	Entire home/apt	2	5/2020	9	100	
19	Apartment	Entire home/apt	2	5/2020	69	100	
20	Apartment	Entire home/apt	2	5/2020	7	100	
21	Apartment	Entire home/apt	2	5/2020	1	100	
22	Apartment	Entire home/apt	2	5/2020	17	100	
23	Apartment	Entire home/apt	2	5/2020	100	100	
24	Apartment	Entire home/apt	2	5/2020	98	100	
25	Apartment	Entire home/apt	2	5/2020	24	100	
26	Apartment	Entire home/apt	2	5/2020	1	100	
27	Apartment	Entire home/apt	2	5/2020	65	100	
28	Apartment	Entire home/apt	2	5/2020	12	100	
29	Apartment	Entire home/apt	2	5/2020	2	100	

Figure 8. dm_property_type table

c. dm_host_neighbourhood

In this view, for 'host_neighbourhood_lga_name' and 'month/year' (grouped by and ordered by), the following columns and values were produced:

- number of distinct host
- estimated revenue
- estimated revenue per distinct host

	host_neighbourhood_lga	month_year	distinct_host_count	estimated_revenue	estimated_revenue_per_l
1	Bayside	1/2021	82	249,701	3,045.1341463415
2	Bayside	4/2021	2	13,710	6,855
3	Bayside	5/2020	14	26,595	1,899.6428571429
4	Bayside	9/2020	1	765	765
5	Blacktown	1/2021	1	4,380	4,380
6	Blacktown	4/2021	1	1,750	1,750
7	Blacktown	5/2020	1	105	105
8	Burwood	1/2021	49	151,511	3,092.0612244898
9	Burwood	5/2020	12	50,613	4,217.75
10	Burwood	8/2020	1	2,970	2,970
11	Campbelltown	2/2021	1	2,520	2,520
12	Campbelltown	5/2020	1	3,996	3,996
13	Canada Bay	1/2021	19	115,681	6,088.4736842105
14	Canada Bay	5/2020	1	5,220	5,220
15	Canterbury-Bankstown	1/2021	3	11,340	3,780
16	Canterbury-Bankstown	5/2020	3	8,093	2,697.6666666667
17	Cumberland	1/2021	21	24,228	1,153.7142857143
18	Cumberland	2/2021	3	1,536	512
19	Cumberland	3/2021	1	810	810
20	Cumberland	5/2020	12	66,913	5,576.0833333333
21	Cumberland	8/2020	1	52	52
22	Cumberland	9/2020	1	300	300
23	Fairfield	5/2020	1	2,590	2,590
24	Georges River	1/2021	12	21,935	1,827.9166666667
25	Georges River	5/2020	1	0	0
26	Georges River	6/2020	1	150	150
27	Hornsby	1/2021	1	11,790	11,790
28	Hornsby	9/2020	1	3,750	3,750
29	Hunters Hill	1/2021	35	236,189	6,748.2571428571
30	Hunters Hill	4/2021	2	1,565	782.5

Figure 9. dm_host_neighbourhood table

PART 2. Issues in Design a Data Warehouse using DBT

- In fact_listings table, there were some null values in columns ‘host_neighbourhood_lga_name’ and ‘host_neighbourhood_lga_code’. This issue arises because there were some values in ‘host_neighbourhood’ that were not matched with values in ‘lga_name’ or ‘suburb_name’ when joining tables, (stg_listings and stg_suburb and stg_LGA). Those null values in two columns were filled with default values (solved).

PART 3. Ad-hoc Analysis

1. The table below shows a list of distinct listing_neighbourhood, their performance based on average estimated revenue per active listings and median age of people. Accordingly, the best performing listing_neighbourhood is Hunters Hill and the worst performing listing_neighbourhood is Fairfield. From a population perspective, Hunters Hill tends to have higher median age than the worst performing Fairfield.

	listing_neighbourhood	avg_revenue	median_age_persons
1	Hunters Hill	7,857.1999667	43
2	Northern Beaches	5,681.2330119401	40
3	Mosman	3,919.5919245041	42
4	Waverley	3,686.7846291197	35
5	Woollahra	3,487.1565936375	39
6	Sutherland Shire	3,333.3831408015	40
7	Canada Bay	2,645.2659772957	36
8	Randwick	2,488.3020364869	34
9	Willoughby	2,458.4105146524	37
10	North Sydney	2,404.9034191745	37
11	Sydney	2,376.9173294213	32
12	Hornsby	2,331.1462382026	40
13	Inner West	2,078.2279006705	36
14	Lane Cove	1,902.2320213379	36
15	Campbelltown	1,591.4880823958	34
16	Cumberland	1,553.5542596843	32
17	Penrith	1,522.8351332126	34
18	Bayside	1,458.7827223108	35
19	Parramatta	1,365.1519759942	34
20	Camden	1,359.5890985325	33
21	Strathfield	1,335.9962593633	32
22	Burwood	1,254.7331347448	33
23	Ryde	1,238.8316927723	36
24	The Hills Shire	1,159.8553440693	38
25	Liverpool	1,144.5918752077	33
26	Canterbury-Bankstown	1,133.6217766955	35
27	Blacktown	991.2735326954	33
28	Georges River	967.3319531639	37
29	Fairfield	745.4125948237	36

Figure 10. The performance of listing_neighbourhood based on average estimated revenue per active listings

2. The table below shows the top 5 best type of listing (property_type, room_type and accommodates) based on the number of stays for the top 5 "listing_neighbourhood" (Hunter Hill, Northern Beaches, Mosman, Waverley and Woollahra). Accordingly, the best property type is 'Entire apartment' and the best room type is 'Entire home/apt' and the optimal accommodates are 2 or 4.

	listing_neighbourhood	property_type	room_type	accommodates	total_number_of_stays
1	Waverley	Entire apartment	Entire home/apt	2	24,415
2	Waverley	Entire apartment	Entire home/apt	4	23,565
3	Waverley	Private room in apartm	Private room	2	20,629
4	Northern Beaches	Entire apartment	Entire home/apt	4	14,936
5	Northern Beaches	Entire apartment	Entire home/apt	2	13,830

Figure 11. Top 5 best type of listing

3. The table below shows lga_name, distinct host_id and number of listings for each host. Accordingly, hosts with multiple listings are more inclined to have their listings in the same LGA where they live.

	host_neighbourhood_lga_name	host_id	num_listings
1	Bayside	33,617,827	6
2	Burwood	167,340,072	21
3	Burwood	255,894,977	12
4	Burwood	293,274,101	18
5	Canterbury-Bankstown	293,274,101	10
6	Cumberland	47,437,319	6
7	Cumberland	293,274,101	16
8	Fairfield	293,274,101	10
9	Inner West	10,441,624	6
10	Inner West	42,440,269	14
11	Inner West	83,630,467	12
12	Inner West	167,340,072	10
13	Lane Cove	10,132,237	8
14	Lane Cove	32,803,646	6
15	Lane Cove	108,189,888	10
16	Lane Cove	137,278,159	9
17	Lane Cove	150,050,000	8
18	Lane Cove	201,706,548	15
19	Mosman	99,141,846	8
20	Mosman	148,745,808	13
21	Mosman	162,174,353	6
22	North Sydney	86,333,173	9
23	North Sydney	153,195,376	7
24	North Sydney	210,813,631	7
25	North Sydney	229,925,439	11
26	North Sydney	297,512,659	15
27	Randwick	292,913	18
28	Randwick	2,450,066	87
29	Randwick	3,192,478	6
30	Randwick	68,788,391	7

Figure 12. Number of listings for lga_name and host_id

4. The table below shows host_id, lga_name, total_estimated_revenue, total_median_mortgage_repay for 2020 and 2021. While some hosts can cover their estimated revenue over the last 12 months by the total median mortgage repayment, some hosts cannot cover their estimated revenue by the total median mortgage repayment.

	host_id	year	host_neighbourhood_lga_nam	total_estimated_revenue	total_median_mortgage_repay	can_cover_mortgage
1	33,294	2,021	Willoughby	30,906.6828282828	23,016	Yes
2	197,987	2,021	Mosman	47,874.1597356544	24,000	Yes
3	201,452	2,020	Randwick	32,341.5958358432	26,000	Yes
4	217,799	2,021	Woollahra	68,801.9365881033	25,600	Yes
5	279,955	2,021	Mosman	47,874.1597356544	24,000	Yes
6	284,711	2,021	Sydney	36,601.2690045249	27,489	Yes
7	316,181	2,020	Waverley	27,614.1393037518	30,000	No
8	326,805	2,021	Willoughby	30,906.6828282828	23,016	Yes
9	333,581	2,021	Sydney	36,601.2690045249	27,489	Yes
10	333,594	2,021	Inner West	12,897.0556006494	18,200	No
11	426,921	2,021	Waverley	27,614.1393037518	30,000	No
12	442,913	2,021	North Sydney	18,391.4782229965	20,800	No
13	453,747	2,021	Waverley	27,614.1393037518	30,000	No
14	476,047	2,021	Waverley	27,614.1393037518	30,000	No
15	501,973	2,021	Waverley	27,614.1393037518	30,000	No
16	503,366	2,021	Randwick	32,341.5958358432	26,000	Yes
17	519,412	2,021	Randwick	32,341.5958358432	26,000	Yes
18	528,837	2,020	Waverley	27,614.1393037518	30,000	No
19	528,837	2,021	Waverley	27,614.1393037518	30,000	No
20	560,537	2,021	Inner West	12,897.0556006494	18,200	No
21	586,347	2,020	Inner West	12,897.0556006494	18,200	No
22	627,555	2,021	Mosman	47,874.1597356544	24,000	Yes
23	665,503	2,021	Waverley	27,614.1393037518	30,000	No
24	675,165	2,021	Woollahra	68,801.9365881033	25,600	Yes
25	691,032	2,021	Sydney	36,601.2690045249	27,489	Yes
26	707,197	2,021	Randwick	32,341.5958358432	26,000	Yes
27	707,921	2,021	Waverley	27,614.1393037518	30,000	No
28	727,275	2,021	Waverley	27,614.1393037518	30,000	No
29	738,056	2,021	Ryde	7,454.9102564103	6,600	Yes

Figure 13. Total estimated revenue and total median mortgage repayment for each host_id and year

4 REFERENCE

docs.getdbt.com. (2023). *strategy | dbt Developer Hub*. [online] Available at: https://docs.getdbt.com/reference/resource-configs/strategy [Accessed 8 Nov. 2023].

Coginiti. (n.d.). *SQL Date Formats*. [online] Available at: https://www.coginiti.co/tutorials/intermediate/sql-date-formats/ [Accessed 8 Nov. 2023].