## horizontal line

Building data pipelines with Airflow

Werner Schott – 10930194

Word Count (without titles, references, etc.): 2,487

[horizontal line 0](#_Toc73100839)

[Business Requirement 2](#_Toc73100840)

[Data Understanding 2](#_Toc73100841)

[Data Description 2](#_Toc73100842)

[Airbnb 2](#_Toc73100843)

[ABS Census Population 2](#_Toc73100844)

[Data Limitations 2](#_Toc73100845)

[Data Cleansing 3](#_Toc73100846)

[Airbnb 3](#_Toc73100847)

[ABS 3](#_Toc73100848)

[ELT Process 4](#_Toc73100849)

[Raw Data Table Definition and Data Load 4](#_Toc73100850)

[Airflow Issues 4](#_Toc73100851)

[Data Warehouse Table Definition, Data Transformation and Load 5](#_Toc73100852)

[Airbnb 5](#_Toc73100853)

[Dimension Tables 6](#_Toc73100854)

[ABS 6](#_Toc73100855)

[Data Mart 8](#_Toc73100856)

[Business Questions 9](#_Toc73100857)

[Conclusion and Learning 10](#_Toc73100858)

[Appendix A: Source Data 11](#_Toc73100859)

[Airbnb 11](#_Toc73100860)

[ABS 2016 Census 12](#_Toc73100861)

[G1 12](#_Toc73100862)

[G2 15](#_Toc73100863)

[Appendix B: Airflow Errors 16](#_Toc73100864)

[Airflow can only load 3 months at a time 16](#_Toc73100865)

[DAG Cannot Be Refreshed 17](#_Toc73100866)

[Appendix C: Airbnb Data: Candidate Entities 19](#_Toc73100867)

[Appendix D: Bibliography 19](#_Toc73100868)

# Business Requirement

There are two sets of requirements in this task: business and technical. The business requirements are about enablement: enabling the business to automatically ingest their data into an accessible and useable arrangement where they may examine and use it to form insights and answer questions. This ‘what’ they want to achieve. The specific questions they would like to answer and the areas they would like to explore, could be thought of as ‘analytical objectives. These should guide and drive the data selection, processing, and storage, as depicted below.

Application

Description automatically generated with low confidence

The second requirement set is comprised of this assignment’s technical requirements and constraints: use prescribed datasets, build production-ready data pipelines, design and populate a database, and create a DataMart to facilitate views of the data that are interesting and necessary to the business—in this case to answer business questions and the necessary KPIs.

The technological constraints are use Airflow for scheduling/integration, load raw data and transform using SQL, use PostgreSQL as a target for ETL/ELT, to persist the data, Python as the backend language, and SQL to create database and analytical data structures to enable BI. The architectural constraints are use an ELT/ETL integration pattern and a data warehouse dimensional data model.

# Data Understanding

## Data Description

### Airbnb

There are twelve monthly ‘listing’ files. Each file holds data from various entities (e.g. listing, host, property) including mainly numeric variables, but also categorical and a few identifiers. There are many null values, but an identifier crucial to linking ABS data and building necessary queries is complete: neighbourhood\_cleansed. Three of the twelve files had different structures and thus had to be transformed to conform to a single structure. For a list of fields and datatypes, refer to [Appendix A](#_Appendix_A).

### ABS Census Population

There are two files. The first (G1) has one geographic identifier (LGA\_CODE\_2016) and a hundred and eight numeric. However only eleven of these were used and thus included in the data warehouse. The second (G2) has the same geographic identifier and eight numeric. However, only one is used and thus loaded. For a field and datatype list refer to [Appendix A](#_Appendix_A)

### Data Limitations

To answer some questions, both datasets must be joined using an appropriate key. While the airbnb listings files have several candidate keys with LGAs or suburbs that could work, many have missing values, and no column aligns consistently with the census data. For instance ‘host\_location’ is often the correct suburb but has null values, and sometimes the host has multiple properties, so it is incorrect. The ‘best’ column—no missing values—is neighbourhood\_cleansed. However, this lists LGAs (Sutherland Shire), inaccurately defined LGAs (Hornsby is Hornsby Shire), and some extinct LGAs due to LGA/council mergers (Canterbury is now City of Canterbury-Bankstown). As such, only 63% of it matches with Census data. Moreover, as Census data has LGA codes, I sourced the names for said codes[[1]](#footnote-1), but there was no list that had all codes—many were missing. Thus, I had to manually create one from various sources, the latest published in 2018[[2]](#footnote-2).

Another major limitation is the way that ABS embeds dimensionality into their data columns. This is an issue with the data issues from many other government agencies. There seems to be a general ignorance about the virtues of dimensional modelling and publishing of the data. This attracts many problems. For instance, the data below combines sex and age into three categories. However, M and L do not equal P. What is the residual 3?



The explanatory notes call this file:



What has been ‘selected’ seems to be random and inconsistent. On the page there are no explanatory notes:

Graphical user interface, website

Description automatically generated

I checked the Census 2016 data dictionary, which did not mention this file[[3]](#footnote-3).

The other problem is if I wanted to know the breakdown between set and females, I cannot. So I cannot use the sex and age dimensionality separately. Thus I used the total, hoping it was best.

## Data Cleansing

### Airbnb

May to July 2020 had a different number of columns to the rest. May and June had thirty-four extra fields, missing two. July also had thirty-four extra. Not all the same as the other two, six being unique. Extra fields were deleted, and missing fields were added as empty columns, to conform to the raw table schema common to other files.

### ABS

To successfully join the ABS and Airbnb data sets using LGA, major and manual cleaning was done, mapping neighbourhood\_cleansed values to actual LGA names. One limitation with this approach was that some neighbourhood\_cleansed values corresponded to two LGAs e.g. parts of Carlton are in Georges River LGA and some in Bayside LGA. Thus qualitative decisions had to be made on a case-by-case basis, depending on the actual address, sometimes determined by the geocodes. However, in a real production system, this process would need to be formalised to ensure uniformity and thus data integrity.

## ELT Process

### Raw Data Table Definition and Data Load

Data was loaded from local files using Airflow, loading all columns as text. Here, tasks and their dependencies were defined and arranged in DAG[[4]](#footnote-4) format, including task execution scheduling and the order of relative task execution. These tasks included the identification of the file locations, the creation of a schema, the creation of database tables, and the population of these tables with the file data. The DAGs were:

1. abs\_raw\_data\_load\_neu
2. airbnb\_raw\_data\_may\_to\_jul\_2020\_updated
3. airbnb\_raw\_data\_load\_aug\_to\_oct\_2020
4. airbnb\_raw\_data\_load\_nov\_2020\_to\_jan\_2021
5. airbnb\_raw\_data\_load\_feb\_to\_apr\_2021

The names are not uniform as these were the final versions that worked, and I was afraid of the files becoming corrupted if I renamed them. Please see [Airflow Issues](#_Airflow_Issues).

The first load included all raw files for ABS and Airbnb, including the manually created LGA mapping file (lga\_raw). Excluding May-June listing files that had to be conformed to the structure of the other nine, these were loaded in their original format, with the addition of ‘month’ and ‘year’ columns to differentiate the file and to serve as keys to join the fact table to the time dimension table. The tables created/populated are as follows:

1. airbnb\_raw
2. g1\_raw
3. g2\_raw
4. lga\_raw

### Airflow Issues

All ABS data was able to be loaded using a single DAG file. However, with Airbnb data, the system crashed and produced several errors. See [Appendix B](#_Appendix_B:_Airflow) for screenshots and details. In summary, Airflow crashed processing a single DAG file to load Airbnb data, but if that file was split into four, loading three months at a time, using the same code in the DAG, it worked.

Some root-cause analysis was done but it was difficult to identify the cause. Likely suspects were Docker’s small memory allocation (using my Windows version I was not able to increase this amount) and the inefficiency of WSL2’s[[5]](#footnote-5) memory allocation and usage. Trouble-shooting this issue unfortunately took days.

I also found an issue with Airflow where, once errors were fixed and the code was updated (shown as such in Airflow), it would still throw up the old error. The workaround I found was to copy and paste the new code into a new file and restart the webserver/container. This became very tedious in trying to trouble-shoot bugs, but eventually provided with a way for the files to work. I thought it may have been the browser cache, but I kept refreshing this to no difference. I also cleared the cache files produced by Airflow.

### Data Warehouse Table Definition, Data Transformation and Load

A new set of DAGs was used to define the schema and fact and dimension tables that made us the data warehouse, as well as to transform the data and to load the tables. All data was loaded from the raw files, which was selected and transformed to make it conform to the data warehouse tables. This meant updating all datatypes from text to their correspoding type, where applicable.

#### Airbnb

What were first thought to be necessary Airbnb entities were modelled. These are listed in [Appendix C: Airbnb Data: Candidate Entities](#_Appendix_C:_Airbnb). However, for simplicity and practicality, only listing, host, and property were used. The data was arranged into a star schema model shown in the below ERD diagram, with a single fact table and three-dimension tables.

Diagram

Description automatically generated

A ‘time’ dimension table was created to store month/year information. After defining the entities themselves, the main principle applied to determine which columns should go where was traditional for data warehousing: categorical variables are stored as dimension values and metrics are stored as facts. This meant that the fact table store both listing and property entity data.

Both applied method for determining where data is stored are slightly subjective as, strictly speaking, all this data can be defined as facts in numeric, temporal and categoric form. However, this model was adopted to meet the assignment requirements.

This model also makes things like managing discrete dimensional data easier e.g. updating slowly changing dimensions, however w less nornalised structure would save having to define and maintain joins between tables, and also the additional time and processing power required to service them. However, in a serious distributed production environment, the priority may be to store the data in a workable format, rather than to optimise processing or storage.

Towards the middle of the assignment, I realised that neighbourhood\_cleansed was a property attribute, not host. By this time, most of the code had been written and it was risky to change it all. However, there is no material difference either way—only semantic and practical (perhaps a number of extra joins). Each table was allocated a system-generated primary key, as only host and time had useable natural keys.

##### Dimension Tables

To ensure there are no errors when loading dimension table data when a key already exists, ‘UPSERT’ statements were included, which provide instructions to either insert (if key does not exist), or update (if it does).

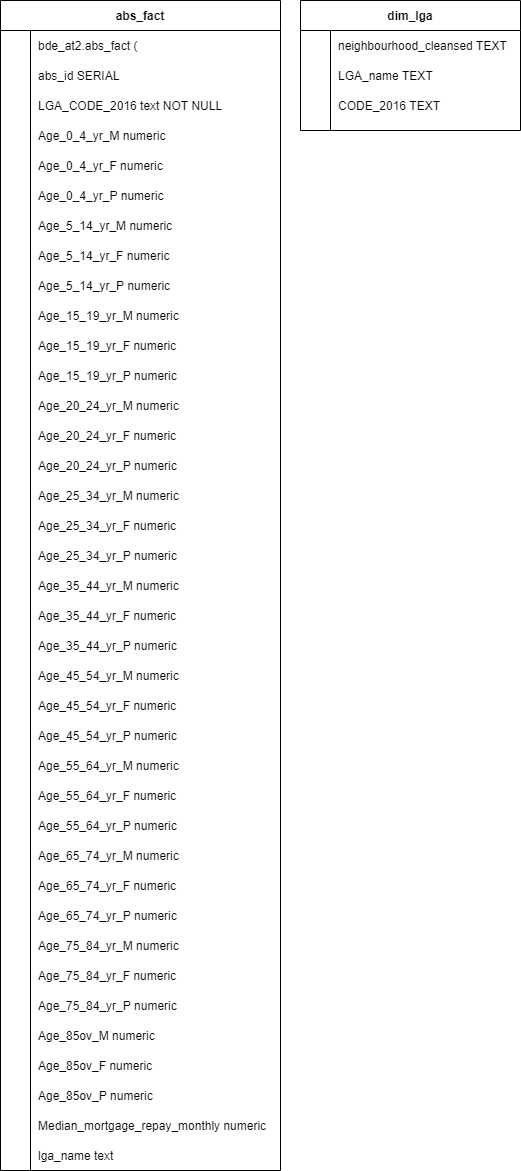
Text

Description automatically generated

#### ABS

As both ABS files had the exactly the same identifiers, they were loaded into a single fact table, using the LGA code as a natural key. As can be seen, only this code, age-related fields (G1), and the ‘mortgage\_repay’ field (G2) were loaded into the data warehouse. As mentioned at the beginning, this was an educated guess based on the business requirement, which translated to the views to be defined in the DataMart to answer the business. This arrangement turned out to be suitable, however, only the ‘total’ fields were used e.g. age\_5\_14\_yr\_p, which included all sexes.

A dim\_lga table was also created to store lga\_raw data. However, as Airbnb data and G1/G2 data had LGAs not featured in the LGA list (some were deprecated) a foreign-key-constraint was not able to be set linking this dimension to either fact table. It was however used successfully by joining it to these to answer question 4c. The ERD for these tables are shown below.



## Data Mart

To populate the DataMart, materialised views were created in a separate schema ‘at2\_datamart’, named to roughly describe what the KPIs are.

Text

Description automatically generated

These scripts can be run over and over as they will either create the view the first time, and then only update the view every other time.

Text

Description automatically generated

In structuring queries to create these views (and also the business question views), it was preferred to use various linked ctes rather than many joins in one query, as these proved to perform better.

## Business Questions

I have defined views for all answers Q4\_a- Q4\_d. To disambiguate, I will refer to them by name.

### 4.a

It is difficult to make reliable inferences using age groups as many underlying and unrelated factors mat drive performance. But to try, the groups must be examined in aggregate form to examine the data in a better relative context. View q4\_a\_1\_revenue\_lga\_g1 shows that regardless of age, the greatest projected revenue is in places either near the beach, the bay, or the CBD. All these views are sorted in descending order by estimated revenue.

Table

Description automatically generated with low confidence

q4\_a\_2\_revenue\_lga\_g1\_prop has calculated proportions of the age brackets to their total, for each LGA. This will help to show relative differences. However, it is not clear without understanding group bias that may exist i.e. 0\_4 and 85\_plu are not likely to be naturally the largest brackets.

Table

Description automatically generated with medium confidence

As such, q4\_a\_3\_population\_bracket\_proportion shows the average value for each bracket.



Also q4\_a\_4\_population\_bracket\_proportion\_average shows the overall.



So we know anything below 9% is negatively biased and the opposite above 9%. Thus, age brackets show nothing insightful, following Sydney demographic patterns (average age 30). As can be seen, brackets 25\_34 is the most populous, with 35\_44, and 45\_54 following. The real pattern here is location, which is a function of property price and thus rental price. The notable exception is Hunters Hill, which is extremely expensive. However, these houses are so expensive, they would be unlikely candidates for Airbnb.

### 4.b

q4\_b\_2\_best\_listing\_type shows that the Northern beaches, eastern suburbs, and CBD top the ranks, with entire apartments being more popular (and abundant) in places with higher population density (CBD, Waverly) and houses being in the greatest demand in the Northern Beaches.

Graphical user interface, application, table

Description automatically generated

Although revenue if a function of number of stays, q4\_b\_3\_number\_of\_stays shows a slightly different result, showing the same trend.

Graphical user interface, text, application

Description automatically generated

Still Northern beaches, eastern suburbs, and CBD, but showing slightly cheaper suburbs, which have volume over price.

### 4.c

# Appendix A: Source Data

## Airbnb

| **#** | **Column** | **Dtype** |
| --- | --- | --- |
| 1 | id | int64 |
| 2 | listing\_url | object |
| 3 | scrape\_id | int64 |
| 4 | last\_scraped | object |
| 5 | name | object |
| 6 | description | object |
| 7 | neighborhood\_overview | object |
| 8 | picture\_url | object |
| 9 | host\_id | int64 |
| 10 | host\_url | object |
| 11 | host\_name | object |
| 12 | host\_since | object |
| 13 | host\_location | object |
| 14 | host\_about | object |
| 15 | host\_response\_time | object |
| 16 | host\_response\_rate | object |
| 17 | host\_acceptance\_rate | object |
| 18 | host\_is\_superhost | object |
| 19 | host\_thumbnail\_url | object |
| 20 | host\_picture\_url | object |
| 21 | host\_neighbourhood | object |
| 22 | host\_listings\_count | float64 |
| 23 | host\_total\_listings\_count | float64 |
| 24 | host\_verifications | object |
| 25 | host\_has\_profile\_pic | object |
| 26 | host\_identity\_verified | object |
| 27 | neighbourhood | object |
| 28 | neighbourhood\_cleansed | object |
| 29 | neighbourhood\_group\_cleansed | float64 |
| 30 | latitude | float64 |
| 31 | longitude | float64 |
| 32 | property\_type | object |
| 33 | room\_type | object |
| 34 | accommodates | int64 |
| 35 | bathrooms | float64 |
| 36 | bathrooms\_text | object |
| 37 | bedrooms | float64 |
| 38 | beds | float64 |
| 39 | amenities | object |
| 40 | price | object |
| 41 | minimum\_nights | int64 |
| 42 | maximum\_nights | int64 |
| 43 | minimum\_minimum\_nights | int64 |
| 44 | maximum\_minimum\_nights | int64 |
| 45 | minimum\_maximum\_nights | int64 |
| 46 | maximum\_maximum\_nights | int64 |
| 47 | minimum\_nights\_avg\_ntm | float64 |
| 48 | maximum\_nights\_avg\_ntm | float64 |
| 49 | calendar\_updated | float64 |
| 50 | has\_availability | object |
| 51 | availability\_30 | int64 |
| 52 | availability\_60 | int64 |
| 53 | availability\_90 | int64 |
| 54 | availability\_365 | int64 |
| 55 | calendar\_last\_scraped | object |
| 56 | number\_of\_reviews | int64 |
| 57 | number\_of\_reviews\_ltm | int64 |
| 58 | number\_of\_reviews\_l30d | int64 |
| 59 | first\_review | object |
| 60 | last\_review | object |
| 61 | review\_scores\_rating | float64 |
| 62 | review\_scores\_accuracy | float64 |
| 63 | review\_scores\_cleanliness | float64 |
| 64 | review\_scores\_checkin | float64 |
| 65 | review\_scores\_communication | float64 |
| 66 | review\_scores\_location | float64 |
| 67 | review\_scores\_value | float64 |
| 68 | license | float64 |
| 69 | instant\_bookable | object |
| 70 | calculated\_host\_listings\_count | int64 |
| 71 | calculated\_host\_listings\_count\_entire\_homes | int64 |
| 72 | calculated\_host\_listings\_count\_private\_rooms | int64 |
| 73 | calculated\_host\_listings\_count\_shared\_rooms | int64 |
| 74 | reviews\_per\_month | float64 |

## ABS 2016 Census

### G1

| **#** | **Column** | **Dtype** |
| --- | --- | --- |
| 1 | LGA\_CODE\_2016 | object |
| 2 | Tot\_P\_M | int64 |
| 3 | Tot\_P\_F | int64 |
| 4 | Tot\_P\_P | int64 |
| 5 | Age\_0\_4\_yr\_M | int64 |
| 6 | Age\_0\_4\_yr\_F | int64 |
| 7 | Age\_0\_4\_yr\_P | int64 |
| 8 | Age\_5\_14\_yr\_M | int64 |
| 9 | Age\_5\_14\_yr\_F | int64 |
| 10 | Age\_5\_14\_yr\_P | int64 |
| 11 | Age\_15\_19\_yr\_M | int64 |
| 12 | Age\_15\_19\_yr\_F | int64 |
| 13 | Age\_15\_19\_yr\_P | int64 |
| 14 | Age\_20\_24\_yr\_M | int64 |
| 15 | Age\_20\_24\_yr\_F | int64 |
| 16 | Age\_20\_24\_yr\_P | int64 |
| 17 | Age\_25\_34\_yr\_M | int64 |
| 18 | Age\_25\_34\_yr\_F | int64 |
| 19 | Age\_25\_34\_yr\_P | int64 |
| 20 | Age\_35\_44\_yr\_M | int64 |
| 21 | Age\_35\_44\_yr\_F | int64 |
| 22 | Age\_35\_44\_yr\_P | int64 |
| 23 | Age\_45\_54\_yr\_M | int64 |
| 24 | Age\_45\_54\_yr\_F | int64 |
| 25 | Age\_45\_54\_yr\_P | int64 |
| 26 | Age\_55\_64\_yr\_M | int64 |
| 27 | Age\_55\_64\_yr\_F | int64 |
| 28 | Age\_55\_64\_yr\_P | int64 |
| 29 | Age\_65\_74\_yr\_M | int64 |
| 30 | Age\_65\_74\_yr\_F | int64 |
| 31 | Age\_65\_74\_yr\_P | int64 |
| 32 | Age\_75\_84\_yr\_M | int64 |
| 33 | Age\_75\_84\_yr\_F | int64 |
| 34 | Age\_75\_84\_yr\_P | int64 |
| 35 | Age\_85ov\_M | int64 |
| 36 | Age\_85ov\_F | int64 |
| 37 | Age\_85ov\_P | int64 |
| 38 | Counted\_Census\_Night\_home\_M | int64 |
| 39 | Counted\_Census\_Night\_home\_F | int64 |
| 40 | Counted\_Census\_Night\_home\_P | int64 |
| 41 | Count\_Census\_Nt\_Ewhere\_Aust\_M | int64 |
| 42 | Count\_Census\_Nt\_Ewhere\_Aust\_F | int64 |
| 43 | Count\_Census\_Nt\_Ewhere\_Aust\_P | int64 |
| 44 | Indigenous\_psns\_Aboriginal\_M | int64 |
| 45 | Indigenous\_psns\_Aboriginal\_F | int64 |
| 46 | Indigenous\_psns\_Aboriginal\_P | int64 |
| 47 | Indig\_psns\_Torres\_Strait\_Is\_M | int64 |
| 48 | Indig\_psns\_Torres\_Strait\_Is\_F | int64 |
| 49 | Indig\_psns\_Torres\_Strait\_Is\_P | int64 |
| 50 | Indig\_Bth\_Abor\_Torres\_St\_Is\_M | int64 |
| 51 | Indig\_Bth\_Abor\_Torres\_St\_Is\_F | int64 |
| 52 | Indig\_Bth\_Abor\_Torres\_St\_Is\_P | int64 |
| 53 | Indigenous\_P\_Tot\_M | int64 |
| 54 | Indigenous\_P\_Tot\_F | int64 |
| 55 | Indigenous\_P\_Tot\_P | int64 |
| 56 | Birthplace\_Australia\_M | int64 |
| 57 | Birthplace\_Australia\_F | int64 |
| 58 | Birthplace\_Australia\_P | int64 |
| 59 | Birthplace\_Elsewhere\_M | int64 |
| 60 | Birthplace\_Elsewhere\_F | int64 |
| 61 | Birthplace\_Elsewhere\_P | int64 |
| 62 | Lang\_spoken\_home\_Eng\_only\_M | int64 |
| 63 | Lang\_spoken\_home\_Eng\_only\_F | int64 |
| 64 | Lang\_spoken\_home\_Eng\_only\_P | int64 |
| 65 | Lang\_spoken\_home\_Oth\_Lang\_M | int64 |
| 66 | Lang\_spoken\_home\_Oth\_Lang\_F | int64 |
| 67 | Lang\_spoken\_home\_Oth\_Lang\_P | int64 |
| 68 | Australian\_citizen\_M | int64 |
| 69 | Australian\_citizen\_F | int64 |
| 70 | Australian\_citizen\_P | int64 |
| 71 | Age\_psns\_att\_educ\_inst\_0\_4\_M | int64 |
| 72 | Age\_psns\_att\_educ\_inst\_0\_4\_F | int64 |
| 73 | Age\_psns\_att\_educ\_inst\_0\_4\_P | int64 |
| 74 | Age\_psns\_att\_educ\_inst\_5\_14\_M | int64 |
| 75 | Age\_psns\_att\_educ\_inst\_5\_14\_F | int64 |
| 76 | Age\_psns\_att\_educ\_inst\_5\_14\_P | int64 |
| 77 | Age\_psns\_att\_edu\_inst\_15\_19\_M | int64 |
| 78 | Age\_psns\_att\_edu\_inst\_15\_19\_F | int64 |
| 79 | Age\_psns\_att\_edu\_inst\_15\_19\_P | int64 |
| 80 | Age\_psns\_att\_edu\_inst\_20\_24\_M | int64 |
| 81 | Age\_psns\_att\_edu\_inst\_20\_24\_F | int64 |
| 82 | Age\_psns\_att\_edu\_inst\_20\_24\_P | int64 |
| 83 | Age\_psns\_att\_edu\_inst\_25\_ov\_M | int64 |
| 84 | Age\_psns\_att\_edu\_inst\_25\_ov\_F | int64 |
| 85 | Age\_psns\_att\_edu\_inst\_25\_ov\_P | int64 |
| 86 | High\_yr\_schl\_comp\_Yr\_12\_eq\_M | int64 |
| 87 | High\_yr\_schl\_comp\_Yr\_12\_eq\_F | int64 |
| 88 | High\_yr\_schl\_comp\_Yr\_12\_eq\_P | int64 |
| 89 | High\_yr\_schl\_comp\_Yr\_11\_eq\_M | int64 |
| 90 | High\_yr\_schl\_comp\_Yr\_11\_eq\_F | int64 |
| 91 | High\_yr\_schl\_comp\_Yr\_11\_eq\_P | int64 |
| 92 | High\_yr\_schl\_comp\_Yr\_10\_eq\_M | int64 |
| 93 | High\_yr\_schl\_comp\_Yr\_10\_eq\_F | int64 |
| 94 | High\_yr\_schl\_comp\_Yr\_10\_eq\_P | int64 |
| 95 | High\_yr\_schl\_comp\_Yr\_9\_eq\_M | int64 |
| 96 | High\_yr\_schl\_comp\_Yr\_9\_eq\_F | int64 |
| 97 | High\_yr\_schl\_comp\_Yr\_9\_eq\_P | int64 |
| 98 | High\_yr\_schl\_comp\_Yr\_8\_belw\_M | int64 |
| 99 | High\_yr\_schl\_comp\_Yr\_8\_belw\_F | int64 |
| 100 | High\_yr\_schl\_comp\_Yr\_8\_belw\_P | int64 |
| 101 | High\_yr\_schl\_comp\_D\_n\_g\_sch\_M | int64 |
| 102 | High\_yr\_schl\_comp\_D\_n\_g\_sch\_F | int64 |
| 103 | High\_yr\_schl\_comp\_D\_n\_g\_sch\_P | int64 |
| 104 | Count\_psns\_occ\_priv\_dwgs\_M | int64 |
| 105 | Count\_psns\_occ\_priv\_dwgs\_F | int64 |
| 106 | Count\_psns\_occ\_priv\_dwgs\_P | int64 |
| 107 | Count\_Persons\_other\_dwgs\_M | int64 |
| 108 | Count\_Persons\_other\_dwgs\_F | int64 |
| 109 | Count\_Persons\_other\_dwgs\_P | int64 |

### G2

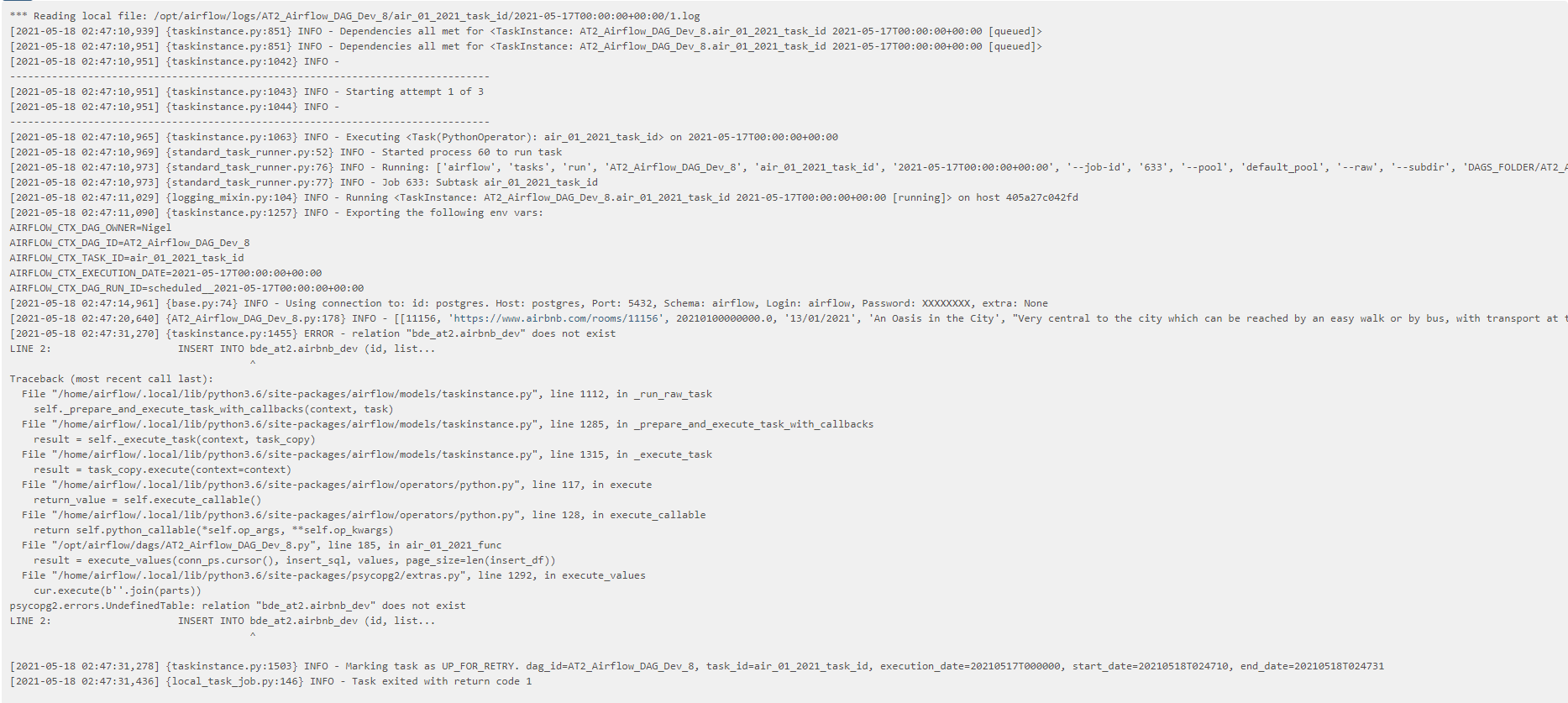
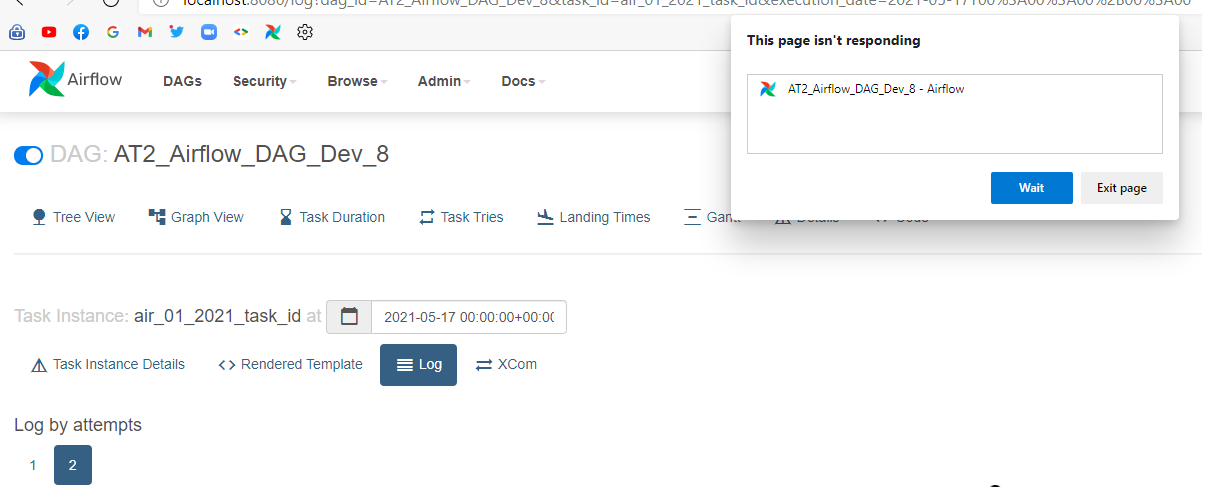
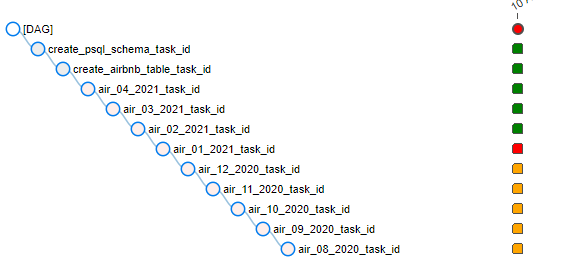
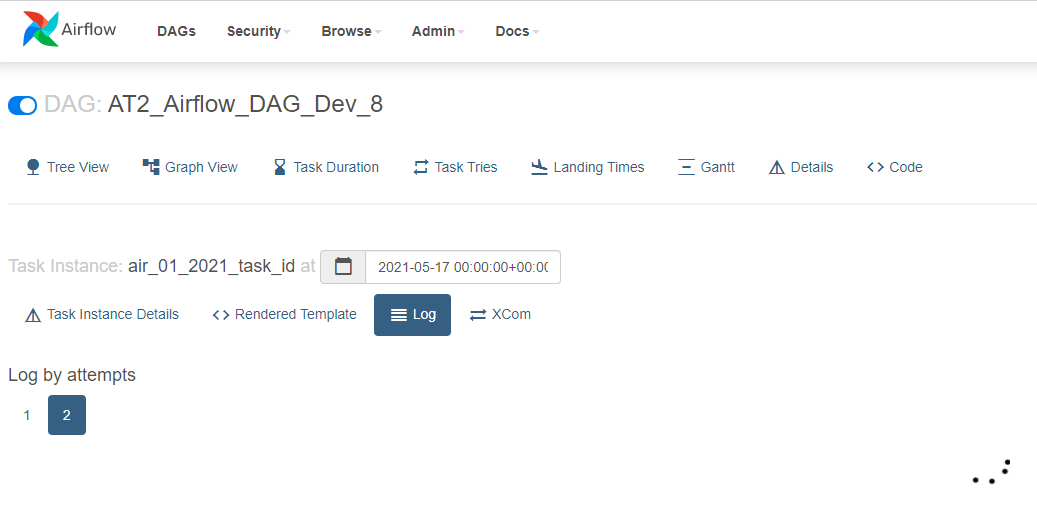
| **#** | **Column** | **Dtype** |
| --- | --- | --- |
| 1 | LGA\_CODE\_2016 | object |
| 2 | Median\_age\_persons | int64 |
| 3 | Median\_mortgage\_repay\_monthly | int64 |
| 4 | Median\_tot\_prsnl\_inc\_weekly | int64 |
| 5 | Median\_rent\_weekly | int64 |
| 6 | Median\_tot\_fam\_inc\_weekly | int64 |
| 7 | Average\_num\_psns\_per\_bedroom | int64 |
| 8 | Median\_tot\_hhd\_inc\_weekly | int64 |
| 9 | Average\_household\_size | int64 |

# 

# Appendix B: Airflow Errors

## Airflow can only load 3 months at a time

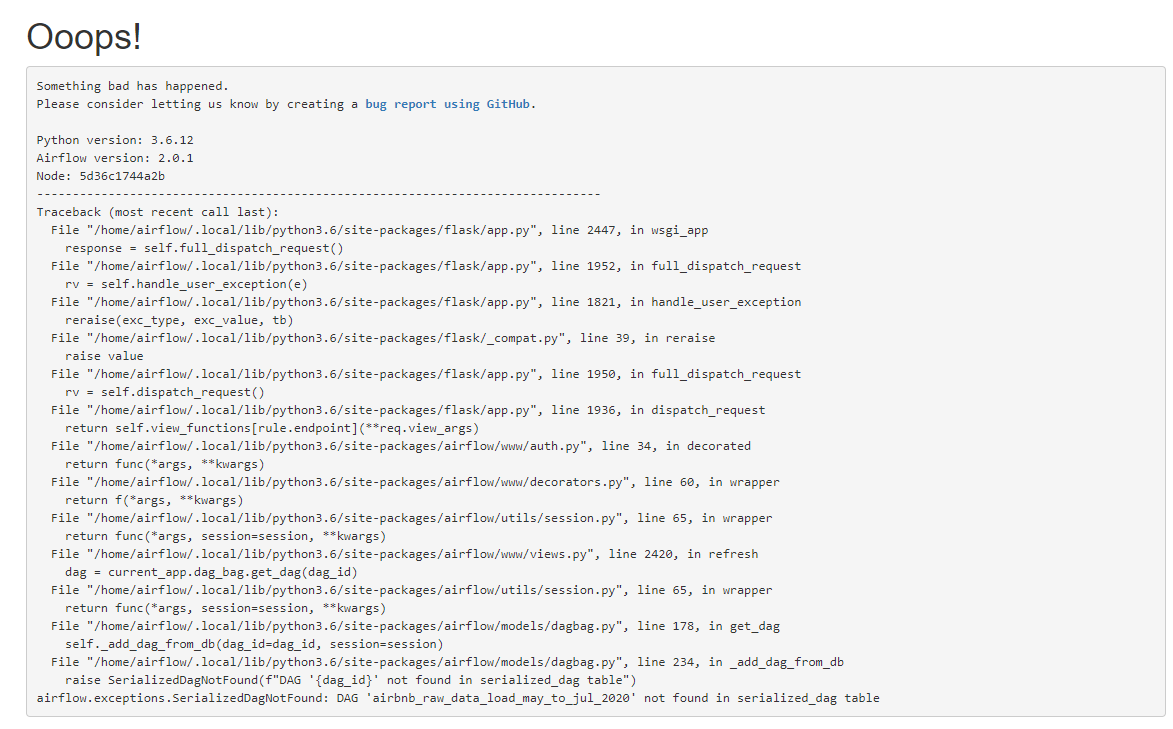
Occurrence: When a single file is used to create table for and load all Airbnb data.



Solution: Used four separate files with the same code.

## DAG Cannot Be Refreshed

Occurrence: when code was updated, and the DAG was refreshed.



Solution: Copy and paste code into a fresh file and restart container/webserver.

# Appendix C: Airbnb Data: Candidate Entities



# Appendix D: Bibliography

Statistics, c=AU; o=Commonwealth of Australia; ou=Australian Bureau of. “Details - Overview.” c=AU; o=Commonwealth of Australia; ou=Australian Bureau of Statistics, September 13, 2016. <https://www.abs.gov.au/AUSSTATS/abs@.nsf/DetailsPage/1270.0.55.003July%202016?OpenDocument>.

Kimball Group. “Kimball Group | Dimensional Data Warehousing Experts.” Accessed May 19, 2021. <https://www.kimballgroup.com/>.

1. New South Wales Local Government Area ASGS Edition 2016 in .csv Format , https://www.abs.gov.au/AUSSTATS/abs@.nsf/DetailsPage/1270.0.55.003July%202016?OpenDocument [↑](#footnote-ref-1)
2. [LGA\_Codes\_and\_Names\_NSW\_2018.csv](https://www.abs.gov.au/AUSSTATS/abs@.nsf/DetailsPage/1270.0.55.003July%202016?OpenDocument), [↑](#footnote-ref-2)
3. 2901.0 - Census of Population and Housing: Census Dictionary [↑](#footnote-ref-3)
4. Directed Acyclic Graph [↑](#footnote-ref-4)
5. Windows Subsystem for Linux [↑](#footnote-ref-5)