Big Data Engineering: Assignment 3

Data Pipelines with Airflow.

Oliver Salman 13881750

Contents table

High level overview of the project

The aim of this project is to showcase understanding of the necessary steps in creating a data pipeline within a business context. The project introduced us to the concepts of ELT, a framework of how data is extracted, loaded and transformed between sources and destinations, as well as how we can implement scheduling using Airflow to make these processes efficient and guaranteed. Furthermore, the project was created within a business environment where key performance indicators (KPIs) were created to track certain performance characteristics within the data. Finally, ad-hoc analysis of our dataset was explored, a typical request from clients and stakeholders interested in exploratory analysis on the data. The project uses Sydney Airbnb listing data as well as census data for Sydney’s LGAs.

Explanation for the different steps of your project

This project consisted of four distinct stages in the pipeline. The first of which consisted of downloading the data from the GCP storage and formatting the schema of each dataset in the staging phase. Part 2 of the project consisted of designing a star schema and loading the data into the star schema. Part 3 of the project consisted of creating key performance indicators from the data in the star schema, and pushing the data into a data mart. Part 4 consisted of answering various ad-hoc analysis from data in the data mart. This report will now go into some detail about the different steps mentioned in the project.

Part 1: Download the datasets

Downloading the datasets consisted of firstly creating a storage integration to the GCP storage where the datasets and the DAG reside. This allows Snowflake to read and write data to and from the GCP storage area. The datasets were then loaded into an external table from the GCP storage. The listing data needed to be loaded in partitions as schemas in months 5, 6, and 7 were different to the rest of the data. Airflow was used for this section, which schedules the downloading of datasets as tasks, which can be executed manually or automatically.

Part 2: Design and populate a data warehouse following an ELT pattern

Designing the data warehouse consisted of firstly defining the star schema, defining column names and data types, and then loading the data into the star schema whilst handling slowly changing dimensions. Data columns from the above mentioned data files need to be renamed to their proper name. In snowflake, data loaded into tables from an external source have default column and row names, renaming these default values to their corresponding value is necessary in the creation of the star schema.

From here, all data has been formatted correctly and is ready to be loaded into the star schema. Loading into the star schema requires the handling of slowly changing dimensions. Slowly changing dimensions are dimensions of data that can slowly change over time with new data being merged into the dataset. I have decided to handle these cases using type 1 handling. Type 1 handling handles slowly changing dimensions by replacing the old data in the dataset with the new data. This method of handling means the dataset does not contain a history of duplicate data, but doesn't show the history of changing data, which could be useful for later analysis and/or debugging. Furthermore, the dimension table in the star schema was created by joining census G01, G02, and LGA datasets on the LGA_code field. The fact table in the star schema was created by joining the listing data partitions on id, year, and month. The fact table and dimension table is then joined together on the listings neighborhood and the census lga name. The star schema was completed by joining the combined listing data with the combined census data on lga name. Below is a code snippet that successfully created the data warehouse.

```
-- star schema with listings and census joined
CREATE OR REPLACE TABLE data_warehouse.listing_final as (
SELECT *
FROM data_warehouse.listings_agg
    FULL OUTER JOIN data_warehouse.census_agg
    on trim(lower(listings_agg.neighbourhood_cleansed)) = trim(lower(census_agg.lga_name))
);
```

Part 3: Design and populate a data mart

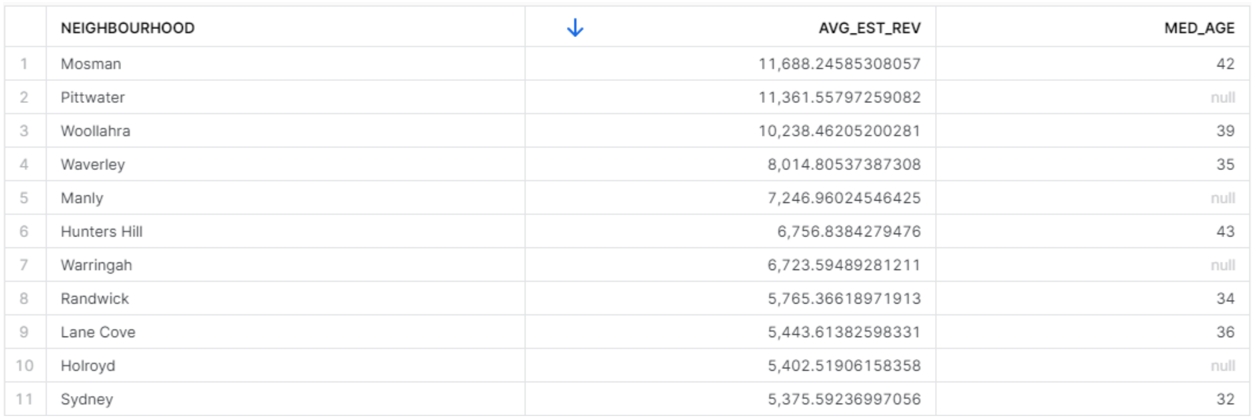
Data marts contain a subset of the data contained in the data warehouse. This section, we build a data mart containing specific KPIs derived from the star schema in the data warehouse. KPIs measure specific performance related characteristics of the data over different domains. The KPIs were constructed using common table expressions (CTEs), which query over a subset of the data we are interested in. This section was completed in airflow which schedules the task of creating the data mart when new data arrives in the data warehouse.

Part 4: Ad-hoc analysis

The Ad-hoc analysis required us to identify some questions which relate the listing data to the census data. Constructing the queries for the ad-hoc analysis was conducted in a similar manner to designing the KPIs in the previous question. This section focuses on SQL query design and optimisation, where both listing and census data are used to show analysis, trends and categories.

Answers to the business questions with supporting evidence

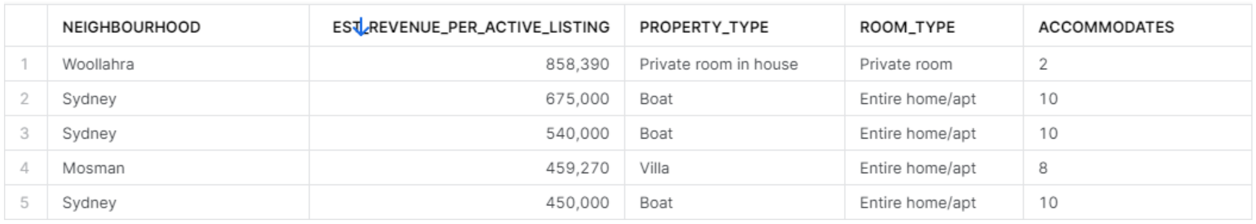
What are the main differences from a population point of view (i.g. higher population of under 30s) between the best performing neighbourhood_cleansed and the worst (in terms of estimated revenue per active listings) over the last 12 months?
This question was answered by grouping neighborhoods, and the median age of that neighborhood by the estimated revenue per active listing. Ordering results by the estimated revenue in descending, saw a slight correlation in estimated revenue and median age, with listings with a higher estimated revenue having a slightly older population. The top 10 neighborhoods by estimated average revenue have an average median age of 37.4. Neighborhoods ranking 11-20 have an average median age of 37.1. Neighborhoods ranking 21-30 have an average median age of 35.4. Neighborhoods ranking 31-38 have an average median age of 33.5.



	NEIGHBOURHOOD	AVG_EST_REV	MED_AGE
1	Mosman	11,688.24585308057	42
2	Pittwater	11,361.55797259082	null
3	Woollahra	10,238.46205200281	39
4	Waverley	8,014.80537387308	35
5	Manly	7,246.96024546425	null
6	Hunters Hill	6,756.8384279476	43
7	Warringah	6,723.59489281211	null
8	Randwick	5,765.36618971913	34
9	Lane Cove	5,443.61382598331	36
10	Holroyd	5,402.51906158358	null
11	Sydney	5,375.59236997056	32

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

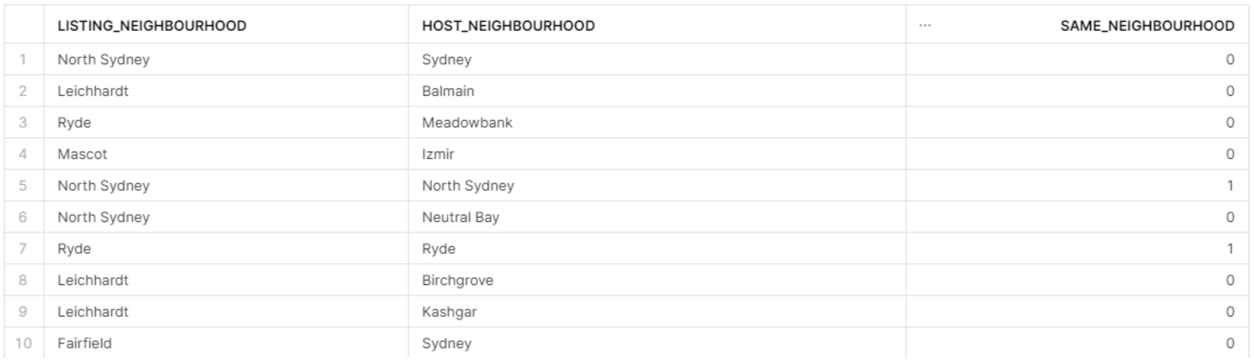
The result to this question was firstly answered by creating a CTE that consisted of neighborhood, property type, room type, accommodates and estimated revenue per active listing, a calculated field. The results show that neighborhoods Woollahra, Sydney and Mosman were of that of the top 5 neighborhoods in terms of highest estimated revenue per active listing, along with their corresponding property type, room type and number of accommodates.



	NEIGHBOURHOOD	EST_REVENUE_PER_ACTIVE_LISTING	PROPERTY_TYPE	ROOM_TYPE	ACCOMMODATES
1	Woollahra	858,390	Private room in house	Private room	2
2	Sydney	675,000	Boat	Entire home/apt	10
3	Sydney	540,000	Boat	Entire home/apt	10
4	Mosman	459,270	Villa	Entire home/apt	8
5	Sydney	450,000	Boat	Entire home/apt	10

Do hosts with multiple listings are more inclined to have their listings in the same "neighborhood" as where they live?

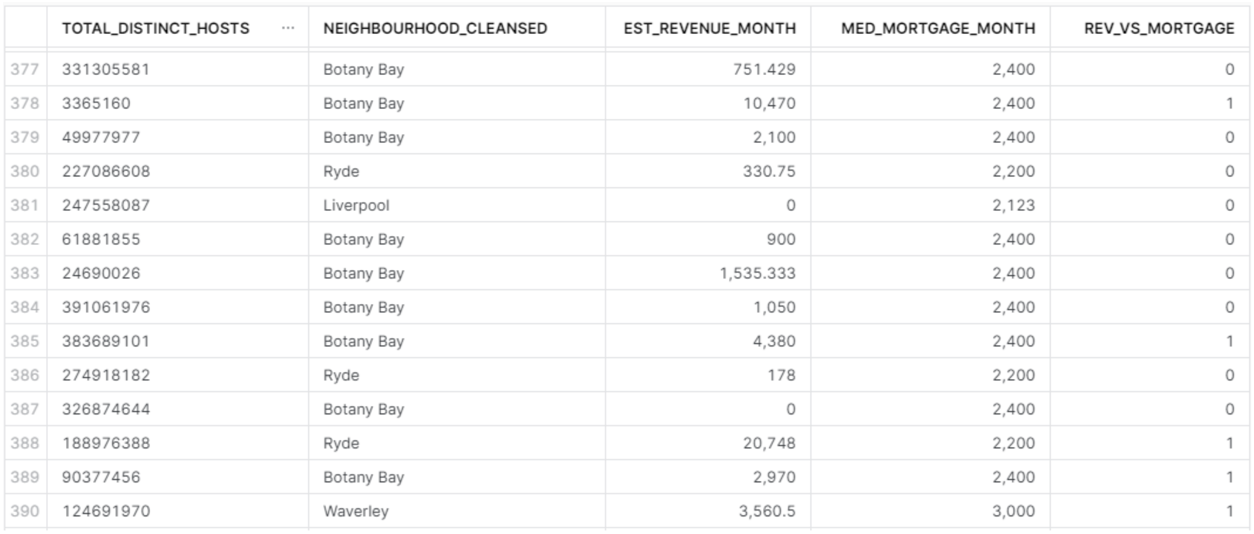
The result to this question was answered by creating a nested CTE that consisted of neighborhood, host location, the total number of superhosts, the total number of hosts. These aggregations were then queried on in the nested CTE, finding the case when the listing neighborhood is equal to the host neighborhood, and flagging this case as 1 if true and 0 else. The results show that 170 superhosts with multiple listings have a listing in the same neighborhood as where they live, out of a total of ~8.5K superhosts. Therefore, with only roughly 2% of superhosts having listings in the same location as where they live, there is no discerning inclination for superhosts to have their listings in the same neighborhood as to where they live.



	LISTING_NEIGHBOURHOOD	HOST_NEIGHBOURHOOD	SAME_NEIGHBOURHOOD
1	North Sydney	Sydney	0
2	Leichhardt	Balmain	0
3	Ryde	Meadowbank	0
4	Mascot	Izmir	0
5	North Sydney	North Sydney	1
6	North Sydney	Neutral Bay	0
7	Ryde	Ryde	1
8	Leichhardt	Birchgrove	0
9	Leichhardt	Kashgar	0
10	Fairfield	Sydney	0

For hosts with a unique listing, does their estimated revenue over the last 12 months can cover the annualized median mortgage repayment of their listing's "neighbourhood_cleansed"?

This question was answered by firstly finding the estimated monthly revenue for each host with a unique listing, the neighborhood of the listing, and its associated median monthly mortgage repayment. The estimated monthly revenue for each host with a unique listing was calculated by grouping each distinct host, neighborhood and median mortgage repayment. The data from this query was queried on through using a CTE, where an additional field is now calculated. To find hosts where their estimated annual revenue covers the annual median mortgage repayment, a case when function was used, where the case when the estimated annual revenue is larger than the median annual mortgage repayment, it is flagged with 1, otherwise 0. Overall, only 36% of hosts with a unique listing have their revenue cover the annual median mortgage repayment of their listing.



	TOTAL_DISTINCT_HOSTS	NEIGHBOURHOOD_CLEANSED	EST_REVENUE_MONTH	MED_MORTGAGE_MONTH	REV_VS_MORTGAGE
377	331305581	Botany Bay	751.429	2,400	0
378	3365160	Botany Bay	10,470	2,400	1
379	49977977	Botany Bay	2,100	2,400	0
380	227086608	Ryde	330.75	2,200	0
381	247558087	Liverpool	0	2,123	0
382	61881855	Botany Bay	900	2,400	0
383	24690026	Botany Bay	1,535.333	2,400	0
384	391061976	Botany Bay	1,050	2,400	0
385	383689101	Botany Bay	4,380	2,400	1
386	274918182	Ryde	178	2,200	0
387	326874644	Botany Bay	0	2,400	0
388	188976388	Ryde	20,748	2,200	1
389	90377456	Botany Bay	2,970	2,400	1
390	124691970	Waverley	3,560.5	3,000	1

Any issues/bugs you faced and how you solved them

Parts of this project were difficult and time consuming, and of which, came with many issues.

The first issue I faced was in the download phase where I would load all listing data into one table and format the columns according to the first row in the download. After formatting the table I found that most of the data was not in the correct column. After carefully inspecting each csv file in excel I found that months 5,6, and 7 had different columns to the rest of the months in the listing data. From here i had two options to correctly download and format all the listing data; to read the files in the DAG file using pandas, to format and concatenate all the data, and then write back to the GCP storage for snowflake to read, or to format directly into snowflake, making sure i format each unique month that was different, making sure that all the columns are the same for all. I started with using python and airflow to format the data, but quickly ran into trouble when connecting to the GCP storage account. As a result, I opted for the latter option by individually formatting the months that were not the same in Snowflake. This was a tedious process with lots of quality control measures, but all the listing data ended up being correctly formatted and ready to be used for the upcoming parts of the project.

Another issue I found was when creating the data warehouse and handling slowly changing dimensions (SCD). Creating the data warehouse consisted of loading all available data into the star schema, whilst handling SCD. The first of the issues arose when designing the star schema, as this concept was new to me. I was unsure how to join the necessary columns together, and how they should be merging properly so no data is lost. Though, this was solved by reading online resources and learning what different types of join functions do to data. This is where I found my second issue, how to handle slowly changing dimensions. After some research, I realized for the use case of this assignment, a type 1 method of handling SCDs is only necessary. A type 1 method of handling SCDs doesn't remember a records’ history, meaning if a row has been updated, the new values will replace the old, and the old will be removed. This was a tedious process of using the merge function along with individually specifying each column and its corresponding value, but it is the only way Snowflake supports its merge statements.

Another issue I faced was creating the KPIs in part 3. The resulting tables had an unusually high amount of rows, which led me to believe this wasn’t correct. I was creating a sequential CTE that contained the information about every single KPI. I resolved this issue by distinguishing each CTE from the others, and focusing on only one KPI at a time. This way I am able to track if the number of rows is correct or not. After creating a CTE for each KPI, I was able to merge all the select statements and common tables into one CTE, where I knew the data was behaving as expected.

Another issue I faced was in part 3 question 3, where we were asked to make host_neighbourhood comparable to lga_name and to use this new column as an aggregate in the KPI. Though I had the correct logic in matching host_neighbourhood to lga_name, I could not figure out the snowflake syntax to insert the values into the new column on the condition.