# InsideAirbnb Austin Data Pipeline Project

**Name: Mouryan Jayasankar**
**UID: 2001459490**

## Introduction

The InsideAirbnb dataset gives a detailed view of short-term rentals and host activity in many large cities. In this project I focus on Austin, Texas, using about 15,000 listings from the 2024 InsideAirbnb Austin extract. This dataset includes property details, host information, prices, availability, and reviews. Because of its size and variety, simple desktop tools are not enough to clean and analyze it efficiently.

To work with this data, I built an end-to-end pipeline on Google Cloud Platform (GCP). The pipeline ingests, cleans, transforms, and then visualizes the Austin Airbnb data. The main goal is to show a working big-data pipeline that is reproducible and scalable, rather than to perform very complex analysis. The design is based on ideas from the course modules on Distributed Processing, Ingest and Storage, and Virtualization. The final output is an interactive Looker Studio dashboard visualizations that lets users explore neighborhoods, pricing trends, and host performance.

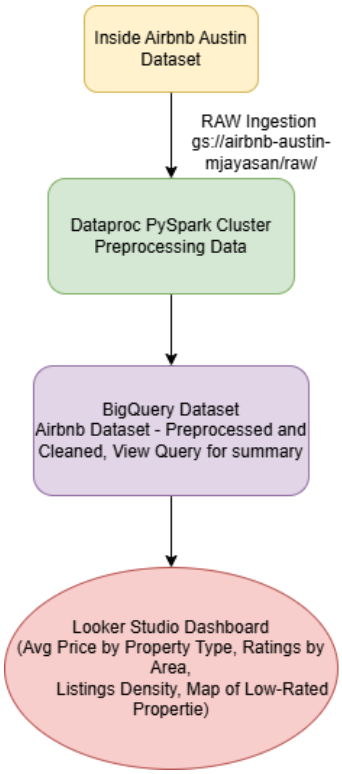
## Background

The Airbnb Austin dataset is a semi-structured file with many different types of information. It has more than 15,000 rows and many columns, including price, location, host details, and reviews. The data is very mixed: some fields are numeric, some are true or false, some are geographic coordinates, others are categories or text. There are also quality problems such as missing values, messy price formats, inconsistent host details, and irregular availability fields. These issues are typical in big data systems.

This project applies three course concepts to handle these challenges. First, distributed processing is used through a PySpark job on Dataproc, which performs cleaning, filling missing values, converting data types, and basic aggregations. Second, ingest and storage are handled by Google Cloud Storage, which holds both the raw files and the cleaned outputs. Third, virtualization is present because the Dataproc cluster runs on virtual machines that are managed for Spark. Together, these steps show how ingestion, transformation, and modeling can work in a practical big data pipeline on a modern cloud platform.

## 3. Methodology

This project was implemented entirely on Google Cloud Platform and follows a straightforward pipeline consisting of storage, distributed processing, data cleaning, warehousing, and visualization. Each component was chosen to highlight a specific big data concept while maintaining simplicity and reproducibility.




### 3.1 Google Cloud Environment Setup

The Google Cloud environment was configured by enabling core services such as Cloud Storage, Dataproc, BigQuery, and IAM. A new project was created inside the course folder, and regional settings were kept consistent across all services to avoid cross-region delays. Service account permissions were adjusted so that Dataproc could read from and write to Cloud Storage and BigQuery. All work was carried out in the web-based GCP Console, without using command-line tools, to keep the setup transparent and easy to document.

### 3.2 Data Upload and Cloud Storage Organization

The InsideAirbnb Austin dataset, which contains about 15,000 rows, was first downloaded locally and then uploaded to Google Cloud Storage. To keep the workflow organized, a simple folder structure was created within the bucket. The raw folder stored the original CSV exactly as downloaded. The processed folder held the cleaned data produced after the Spark job. The scripts folder contained the PySpark code used for data cleaning, and the logs folder captured job outputs and diagnostic information. This layout kept each stage of the pipeline separate and easy to manage, making validation, reruns, and updates more efficient.

### 3.3 Dataproc Cluster Creation

A single-node Dataproc cluster was created to run PySpark jobs. Dataproc provides a managed Spark environment running on virtual machines, which allows large datasets to be processed in a distributed manner without manually configuring Hadoop or Spark. The chosen machine type (n1-standard-4) offered enough memory and CPU for the size of the Airbnb dataset. Idle auto-deletion was enabled to stay within course credit limits. During setup, common issues such as missing IAM permissions and network warnings were resolved to ensure the cluster could run jobs successfully.

### 3.4 Data Cleaning and Transformation with PySpark

The main data preparation work was done using a PySpark script submitted to the Dataproc cluster. The script selected only the columns needed for analysis and removed other fields such as URLs, long descriptions with noisy text, links, and columns with many unwanted symbols. It fixed inconsistent text fields by trimming extra spaces, standardizing letter casing, and cleaning unexpected characters. Unicode characters were normalized to avoid loading issues in BigQuery. Price and percentage fields were cleaned by removing currency or percent symbols and converting them into numeric values. Data types were cast into appropriate numeric or string formats, and missing values were handled using the median for numeric columns and the most frequent value for categorical columns. Rows with invalid geolocation data, such as missing latitude or longitude, were also removed. The goal of this process was not to build complex models but to create a clean, consistent dataset ready for querying and basic visualization.

### 3.5 Loading Cleaned Data to BigQuery

After cleaning, the final PySpark output was written directly to a BigQuery table. A dedicated dataset airbnb_dataset was created inside BigQuery, and the cleaned table preprocessed_listings became the central source for analysis. BigQuery was also used to run several exploratory queries, such as price comparisons, neighbourhood summaries, rating patterns, and host characteristics. These queries helped validate that the cleaning process worked correctly and provided the metrics needed for later visualization.

### 3.6 Visualization Development in Looker Studio

These visualizations allowed the cleaned Airbnb dataset to be explored from different perspectives geographic, economic, and quality-based. Looker Studio automatically refreshed the charts when the underlying BigQuery tables were updated, completing the final stage of the pipeline.

Looker Studio was connected directly to BigQuery to build interactive visualizations. Four main charts were created:

1. **Average price by property type**
2. **Average review rating by neighbourhood**
3. **Listing counts across neighbourhoods**
4. **Map of listings with review scores below 3.0**

## 4. Results

**Fig.1 ( Total number of listing per neighbourhood in Austin )**


**Fig.2 ( Avg. Price per Property Type )**

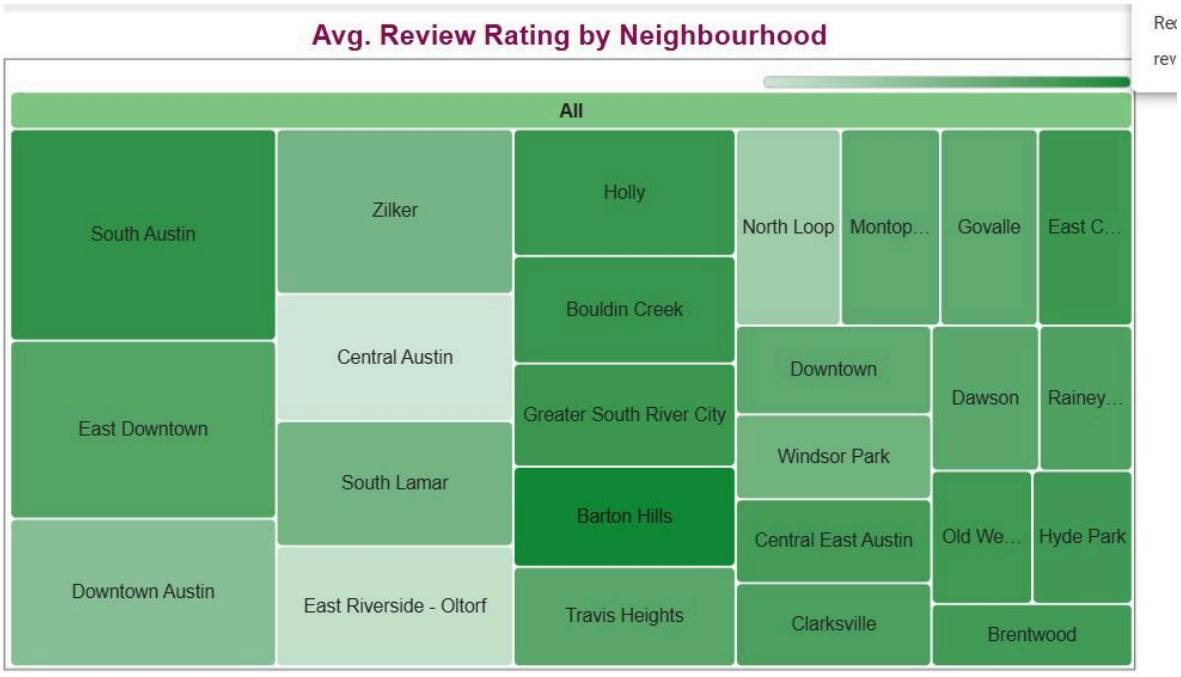



**Fig.3 ( Average Review Rating By Neighbourhood )**


**Fig. 4 ( Average Price per No. of Bedrooms )**

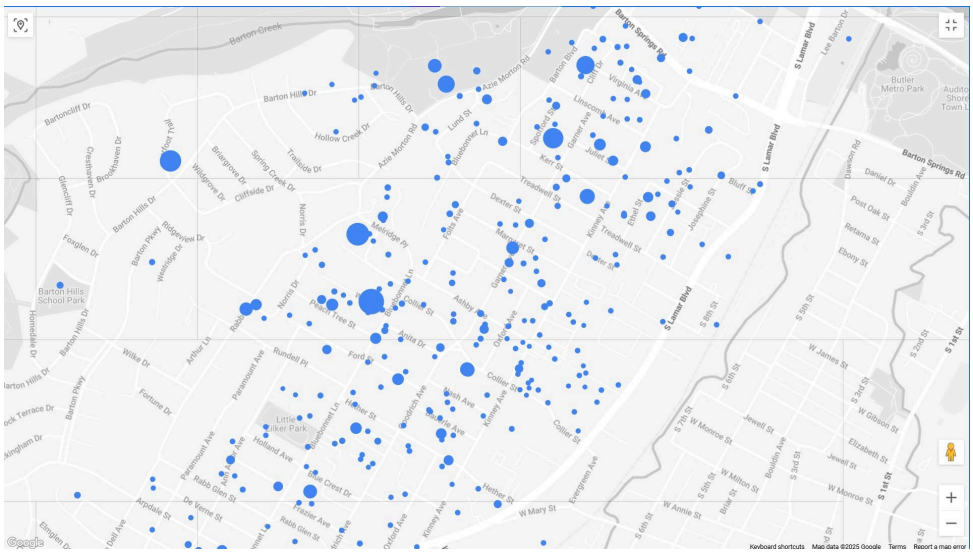



**Fig. 5 ( Average Price of each rental in Zilker Neighbourhood )**

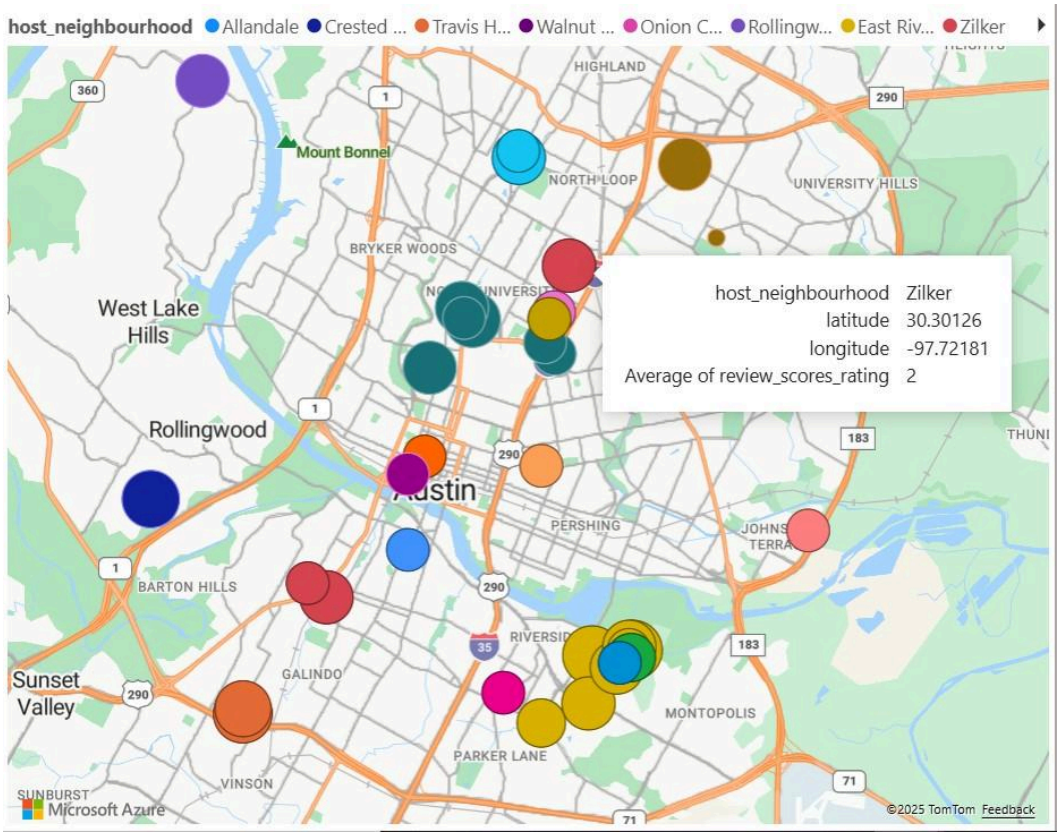



**Fig. 6 ( Neighbourhoods with Reviews Less than 3 out of 5 )**

## 5. Discussion

The pipeline results show clear patterns in how people use Airbnb in Austin and prove that a cloud-based big data system can give useful, real-world insights. Building this system also showed that we had to make some technical trade-offs and design choices, and that our decisions were influenced by the course content and the cloud tools we used.

### 5.1 Results Interpretation

**Fig.1 -** The horizontal bar chart illustrates the number of Airbnb listings grouped by the host_neighbourhood column from the cleaned dataset.
From this, we can infer that South Austin has the highest listing concentration (587), making it the most active rental area, followed by Zilker, Central Austin, and South Lamar, which also show strong listing volumes.

**Fig.2. -** The bar chart displays the average price of listings grouped by the property_type column from the cleaned BigQuery dataset, where the price field was parsed and converted into a numeric value during preprocessing.

From this visualization, we can infer that hotel-based stays such as Room in hotel and Room in boutique hotel command the highest average prices (≈ $3,200+), while categories like Entire home, Treehouse, and Others remain significantly more affordable, indicating strong price variation based on accommodation type.

**Fig.3 -** The above treemap visualizes the average review_scores_rating across different neighbourhoods from the cleaned dataset, using the host_neighbourhood field grouped with the aggregated mean rating.
From the chart, we can infer that neighbourhoods such as South Austin, Zilker, Holly, and East Downtown consistently achieve higher average ratings, indicating strong guest satisfaction, while a few areas show relatively lighter shades, suggesting slightly lower review performance.

**Fig. 4 -** The above bar chart displays the relationship between the number of bedrooms and the average listing price, calculated from the bedrooms and cleaned numeric price fields in the dataset.
From the visualization, we can infer that prices increase steadily with bedroom count, with larger 5-12 bedroom properties commanding significantly higher average rates, indicating strong pricing scalability with property size.

**Fig. 5 -** The above Google Maps bubble visualization displays individual rental prices within the Zilker neighbourhood, using the latitude, longitude, and cleaned numeric price fields filtered from the dataset.
From the distribution and bubble sizes, we can infer that Zilker hosts a dense cluster of rentals with varying price levels, where larger bubbles indicate higher-priced listings concentrated closer to major streets like Barton Springs Rd and BlueBonnet etc.

**Fig. 6** - The above geospatial bubble map highlights listings that have review_scores_rating below 3, plotted using geographic coordinates (latitude, longitude) and grouped by the host_neighbourhood field from the cleaned dataset.
From this visualization, we can infer that low-rated listings are scattered across multiple neighbourhoods including Zilker, North Loop, Barton Hills, and Greater South River City indicating isolated quality issues rather than a neighbourhood-wide pattern.

## 5.2 Connection to Course Modules and Concepts

This project connects closely to several core modules from the course. It applies Data Understanding and Preparation to clean and organize the Airbnb fields, uses Cloud Computing through GCP services, relies on Virtualization via the Dataproc cluster on virtual machines, and follows Ingest and Storage practices by structuring data in Cloud Storage and loading it into BigQuery to visualize using Looker Studio.

**Distributed Processing**

Running PySpark on Dataproc demonstrates how large datasets can be processed in parallel and scaled easily without relying on a single machine. It reflects the course discussions on distributed transformations and cluster-based computation.

**Ingest & Storage**

Cloud Storage holds both the raw and cleaned Airbnb files, showing how cloud systems organize data arrival and long-term storage. The clear folder structure follows the ingest principles taught in class.

**Virtualization**

Dataproc uses virtual machines behind the scenes, but I never had to configure or manage them directly. This matches the course idea that virtualization hides hardware complexity while still providing flexible compute power.

**Cloud Computing**

Using Storage, Dataproc, BigQuery, and Looker Studio together shows how cloud services can replace local servers and simplify big data workflows. It also highlights the convenience and modularity of cloud-native analytics.

## 5.3 Challenges and Privacy Access Issues

A major challenge in the project was dealing with Google Cloud networking and IAM setup. When creating the Dataproc cluster, the system reported that the default VPC network did not exist in my project, so I had to manually create a new default VPC and add basic firewall rules like internal, SSH, and ICMP. Dataproc also required the Cloud Resource Manager API, which was initially disabled, causing validation errors until I enabled it through the API Library. In addition, the cluster could not read from Cloud Storage because the project's compute service account did not have the required permissions. I had to manually grant Storage Object Admin to the account so it could access the staging and temporary buckets. These steps fixing the missing network, enabling necessary APIs, and assigning the correct IAM roles were essential for Dataproc to run successfully and represented the most time-consuming setup challenges in the project.

## 6. Conclusion

This project showed how a simple but well-structured pipeline can turn a raw public dataset like InsideAirbnb Austin into clean, queryable, and visual insights using only managed cloud services. By combining Cloud Storage, Dataproc (PySpark), BigQuery, and Looker Studio, I was able to handle ingestion, cleaning, storage, and visualization in one environment without managing physical servers. The main takeaway is that good data engineering practices, clear folder structure, careful cleaning, and sensible schemas are crucial for making later analysis straightforward. With more time, I would add automation and monitoring, so the pipeline could regularly refresh with new Airbnb data and alert me if any stage fails.

## 7. References

I. Inside Airbnb. (2024). *Inside Airbnb: Get the Data* - Austin listings dataset.
II. Google Cloud. (2025). *Google Cloud Storage - Product Documentation.*
*III. Google Cloud. (2025). Dataproc - Managed Spark and Hadoop Service.*
*IV. Google Cloud. (2025). BigQuery - Data Warehouse Documentation.*
*V. Google Cloud. (2025). Looker Studio - Visualize Your Data.*
*VI. Apache Software Foundation. (2025). Apache Spark: PySpark Documentation.*
*VII. https://github.com/Mouryan-J/AustinAirbnb-GCP-Pipeline.*