Detailed README / Documentation:

Data Preprocessing Using SQL for Hotel Bookings (Kaggle Dataset)

The goal of this project is to replicate the Python preprocessing for hotel booking data using the PostgreSQL dialect. The PostgreSQL GUI used for this purpose is DBeaver. Additionally, all locally set Python scripts from previous projects, which supported earlier KPI dashboards, have been migrated to Google Colab to improve execution speed and simplify GitHub version control. The PostgreSQL preprocessing steps are carefully compared with the corresponding Python preprocessing scripts for each step.

As the next step, the already preprocessed data are loaded into Power BI. At this stage, it is assumed that Power BI is used only for dashboard creation, either interactive or static, since all preprocessing has been completed outside of Power BI. However, for interactive dashboards, all metrics should be created within Power BI. Based on this assumption, it is crucial to ensure that the cleaned data imported into Power BI are fully vetted. For this reason, a quick yet thorough check is applied within Power BI. These checks are presented in detail below so they can be reused for future operations and to support better project defense.

Quick Check on Data Integrity in Power BI

1. **Number of Columns:** Click Report view → Expand Transform Data → Choose Transform Data → The Power Query Editor opens → Look at the bottom of the screen as the number of columns is displayed → 23 columns must be displayed.
   ***** Ensures no columns were lost during import *****
2. **Number of Rows:** Table view → Look at the bottom of the screen; the number of rows is displayed → 117,316 rows must be displayed.
   ***** Ensures no rows were lost during import *****
3. **Data Types:** Open Power Query Editor with the steps described above and check next to each column header. Ensure that both the order of columns and the data types match the corresponding cleaned PostgreSQL table. Note that Power BI may use simpler data types for quick metric calculations → There are numeric, text, decimal and date columns.

***** Ensures both the order and data types match the cleaned PostgreSQL and Python tables*****

4. **Null Values:** Open Power Query Editor → Choose the View menu → Check Column Quality → Valid must be 100% for each column.

   ***** Confirms all calculations and metrics will function correctly *****

5. **Duplicates:** Remember the initial number of rows (in this case, 117,316) → Open Power Query Editor → Choose Home → Choose all columns with ctrl + A → Remove Rows → Remove Duplicates → Return to Power BI Desktop → Apply changes → Check the row number after duplicates removal → 84,640 rows must be displayed, revealing 32,676 duplicates.

   Note on Duplicates: The duplicates in this dataset were generated due to simplifications during preprocessing. For example, all high-cardinality columns were dropped, and further preprocessing made some rows less unique. Only the arrival_date column remains as a high-cardinality column, so bookings on the same day may appear as duplicates in the final cleaned dataset, but all rows still represent valid, unique bookings.

   ***** Confirms consistency across tools*****

6. **Distinct Dates:** Table view → Click the column header → Check below at the page where the column's distinct values are displayed → 793 distinct date values must be displayed.

   ***** Confirms date columns imported correctly*****

7. **Dates with 100 Bookings:** Report view → Modelling → New table → Copy this code up to DAX field:

```
DatesWith100Bookings =
FILTER (
  SUMMARIZE(
    'public table13',
    'public table13'[arrival_date],
    "BookingCount", COUNTROWS('public table13')
  ),
  [BookingCount] = 100
)
```

   → Commit it clicking the green tick → Insert → Table → Expand the DatesWith100Bookings and drag arrival_date and BookingCount in the newly inserted table → Two dates must be displayed: 2015-09-01 and 2015-12-28.

***** Confirms date columns imported correctly and consistency across tools*****

Future Suggestions

1) The KPIs previously calculated in the LookerStudioKPIDashboard project can also be migrated to Google Colab. This is ideal for simulating the static dashboard originally created in Looker Studio using Power BI.
2) Another option is to explore performing the entire preprocessing directly within Power BI, bypassing Python and PostgreSQL, at least for some tasks, as Power BI can be more limited compared with SQL and Python.
3) Interactive dashboards can be created using Power BI with the already cleaned dataset.
4) The same dataset can be preprocessed using Azure Databricks notebooks, which allow for SQL cells. Completing this will enable demonstrating preprocessing through four different approaches.
5) It is a good idea to convert existing PostgreSQL scripts to MySQL, Oracle SQL, SQL Server, etc., to demonstrate flexibility across different SQL dialects and show that experience with one dialect enables adaptation to others using modern SQL translation tools. Quick validation checks are, of course, necessary to ensure correctness and consistent results across dialects.

Conclusions

This project is part of the hotel_booking Kaggle dataset super project (all related projects are available on GitHub). Depending on the problem being addressed, there are multiple computational environments that can be used to generate meaningful results. For instance:

1) The dataset was preprocessed with Python in a local environment. Most steps were automated so that only Looker Studio reconnections are required to generate static dashboards with updated data. Although these dashboards are static, they are updated periodically, allowing KPI monitoring and providing actionable insights to the company in alignment with domain expert feedback.
2) The dataset was also preprocessed with Python in a cloud environment using Google Colab, demonstrating portability and reproducibility of the workflow.
3) The dataset was preprocessed in a local environment using PostgreSQL via DBeaver, with Power BI connected to the local PostgreSQL server. Using the DirectQuery option

allows Power BI to query the database directly, enabling quick refreshes of updated data without importing the full dataset.