Airbnb Data Analysis

Low Level Design Document

Divyansh Verma
Data Analyst Intern
iNeuron AI

Table of Contents

1. Introduction **3**
1.1. What is Low-Level Design Document? 3
1.2. Scope 3
2. Architecture **4**
Architecture Description 4
3. Data Description **5**
3.1 Data Transformation 5
3.2. Data Insertion into Database 5
3.3. Export Data from Database 6
3.4. Deployment 6
4. Unit Test Cases **9**

1. Introduction

This document provides a detailed low-level design for the AirBnB data analysis project. It covers the architecture, data transformation, data insertion into the database, data export, deployment, and unit test cases for the analysis.

1.1. What is a Low-Level Design Document?

A Low-Level Design (LLD) document provides the detailed implementation logic for each component of the project. It translates the high-level design into a detailed blueprint for developers to follow.

1.2. Scope

The scope of this document includes:

- Detailed architecture description.
- Data transformation processes.
- Data insertion and export from the database.
- Deployment procedures.
- Unit test cases for each component.

2. Architecture

Architecture Description

The architecture consists of the following components:

- Data Ingestion: Fetching data from a CSV file.
- Data Preprocessing: Cleaning and transforming the data for analysis.
- Exploratory Data Analysis (EDA): Analyzing data to uncover insights.
- Modeling and Analysis: Building models to answer research questions.
- Visualization: Creating visual representations of the findings.
- Deployment: Deploying the analysis and results.

3. Data Description

The dataset contains information about AirBnB listings in San Diego, California for 2019. Key attributes include:

- `listing_id`
- `host_id`
- `neighborhood`
- `latitude`
- `longitude`
- `price`
- `reviews`
- `overall_satisfaction`
- `bedrooms`
- `bathrooms`
- `accommodates`

3.1 Data Transformation

1. **Data Cleaning**:
    - Handling missing values.
    - Removing duplicates.
    - Correcting data entries.
2. **Feature Engineering**:
    - Creating new features like `total_customers` (reviews * accommodates) and `total_earnings` (price * reviews).
3. **Data Normalization**:
    - Normalizing numerical features for better model performance.

3.2. Data Insertion into Database

1. **Database Setup**:
    - Choose a database (e.g., PostgreSQL, MySQL).
    - Define schema for storing the dataset.
2. **Data Insertion**:
    - Use `pandas` to load data into the database.

3.
```
from sqlalchemy import create_engine

engine = create_engine('postgresql://username:password@localhost:5432/airbnb')
```

```
airbnb_data.to_sql('airbnb_listings', engine, index=False,
if_exists='replace')
```

3.3. Export Data from Database

1. **Data Retrieval**:
    - Query the database to retrieve transformed data.
    - ```
      query = "SELECT * FROM airbnb_listings"

      airbnb_data = pd.read_sql(query, engine)
      ```

3.4. Deployment

Environment Setup:

- Set up a Python environment with required libraries (`numpy`, `pandas`, `matplotlib`, `seaborn`, `statsmodels`, `sqlalchemy`).

Deployment Script:

- Create a deployment script to automate the analysis process.
      ```

```
# deployment_script.py

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import statsmodels.api as sm

from sqlalchemy import create_engine


# Data Ingestion

airbnb_data =
pd.read_csv("https://raw.githubusercontent.com/divyansh-193/airbn
b_analysis/main/airbnb_prices.csv")


# Data Transformation

airbnb_data['total_customers'] = airbnb_data['reviews'] *
airbnb_data['accommodates']

airbnb_data['total_earnings'] = airbnb_data['price'] *
airbnb_data['reviews']


# Database Connection

engine =
create_engine('postgresql://username:password@localhost:5432/airb
nb')


# Data Insertion
```

```
airbnb_data.to_sql('airbnb_listings', engine, index=False,
if_exists='replace')



# Analysis and Visualization

# (Include EDA, modeling, and visualization code here)



# Export Data

transformed_data = pd.read_sql("SELECT * FROM airbnb_listings",
engine)
```

4. Unit Test Cases

4.1. Test Data Ingestion:

```
def test_data_ingestion():

    data =
pd.read_csv("https://raw.githubusercontent.com/divyansh-193/airbn
b_analysis/main/airbnb_prices.csv")

    assert not data.empty, "Data ingestion failed"
```

4.2. Test Data Transformation:

```
def test_data_transformation():

    data['total_customers'] = data['reviews'] *
data['accommodates']

    data['total_earnings'] = data['price'] * data['reviews']

    assert 'total_customers' in data.columns, "Feature
engineering failed"

    assert 'total_earnings' in data.columns, "Feature engineering
failed"
```

4.3. Test Data Insertion:

```
def test_data_insertion():

    engine =
create_engine('postgresql://username:password@localhost:5432/airb
nb')

    data.to_sql('airbnb_listings', engine, index=False,
if_exists='replace')
```

```
    inserted_data = pd.read_sql("SELECT * FROM airbnb_listings",
engine)

    assert not inserted_data.empty, "Data insertion failed"
```

4.4. Test Data Retrieval:

```
def test_data_retrieval():

    engine =
create_engine('postgresql://username:password@localhost:5432/airb
nb')

    retrieved_data = pd.read_sql("SELECT * FROM airbnb_listings",
engine)

    assert not retrieved_data.empty, "Data retrieval failed"
```

4.5. Test Visualization:

```
def test_visualization():

    # Code to test visualization functions

  Pass
```