**Low-Level Design Document for Airbnb Listing Analysis**

**Introduction**

**What is a Low-Level Design (LLD) Document?**

The Low-Level Design (LLD) document provides detailed, step-by-step specifications of how the components of the project are implemented. It describes the data architecture, transformations, database operations, unit testing, and deployment strategy. While the High-Level Design (HLD) gives a bird’s-eye view of the system, LLD dives into the specifics and technical details required to build the solution.

**1. Scope**

This document focuses on:

* **Data Acquisition** through scraping or connecting to existing datasets.
* **Data Transformation** using Python.
* **Database Operations** with Excel or Power BI integration.
* **Data Export and Reporting** through Power BI.
* **Unit Testing** to ensure correctness and maintainability.
* **Deployment** strategy for visual dashboards and analysis.

**2. Architecture**

**Overview of the Architecture**

The architecture follows a **modular design** that involves:

1. **Data Ingestion**: Load data into Python from an existing dataset (e.g., CSV).
2. **Data Cleaning & Transformation**: Handle missing values, outliers, and create derived features.
3. **Data Storage**: Use Excel to store intermediate results for seamless integration with Power BI.
4. **Data Analysis & Visualization**: Power BI for interactive dashboards.
5. **Deployment**: Publish dashboards on Power BI for stakeholder access.

**3. Architecture Description**

**3.1 Data Description**

The dataset contains Airbnb listings for San Diego, California, including the following columns:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CustomerID** | **ProdTaken** | | **Age** | **TypeofContact** | | **CityTier** | | **DurationOfPitch** | **Occupation** | | **Gender** | |
| **NumberOfPersonVisited** | | **NumberOfFollowups** | | | **ProductPitched** | | **PreferredPropertyStar** | | | **MaritalStatus** | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **NumberOfTrips** | **Passport** | **PitchSatisfactionScore** | **OwnCar** | **NumberOfChildrenVisited** |

|  |  |
| --- | --- |
| **Designation** | **MonthlyIncome** |

**3.2 Data Acquisition / Web Scraping (if applicable)**

If scraping is required:

1. **Python Libraries Used**: BeautifulSoup, requests.
2. **URL Access & Scraping Logic**:
   * Fetch listing data from Airbnb’s public pages.
   * Use requests to send GET requests and parse responses using BeautifulSoup.
3. **Data Parsing**: Extract key details such as prices, availability, and review counts.
4. **Storage**: Store scraped data in a structured CSV or Excel format.

**Example Code Snippet for Scraping:**

python

Copy code

import requests

from bs4 import BeautifulSoup

url = 'https://www.airbnb.com/s/San-Diego--CA'

response = requests.get(url)

soup = BeautifulSoup(response.content, 'html.parser')

# Extract listing prices

prices = [item.text for item in soup.find\_all(class\_='listing-price')]

print(prices)

**3.3 Data Transformation**

Using **Python and Pandas** to clean and transform the data:

* **Handling Missing Values**:
  + Use .fillna() to replace null values for reviews\_per\_month and price.
* **Outlier Detection**:
  + Use IQR method or Z-scores to detect and handle extreme values in prices.
* **Feature Engineering**:
  + Create new columns:
    - **Price per Person**: price / number\_of\_guests.
    - **Booking Frequency**: availability\_365 / 12.
* **Data Export to Excel**:
  + Use pandas.DataFrame.to\_excel() to store transformed data for analysis in Power BI.

**Example Code Snippet:**

python

Copy code

import pandas as pd

# Load dataset

df = pd.read\_csv('airbnb\_san\_diego\_2019.csv')

# Handle missing values

df['reviews\_per\_month'].fillna(0, inplace=True)

# Feature engineering

df['price\_per\_person'] = df['price'] / df['accommodates']

# Export to Excel

df.to\_excel('transformed\_data.xlsx', index=False)

**3.4 Data Insertion into Database**

In this case, **Excel** is used as the intermediate storage format to make the data available for **Power BI** dashboards. The transformed data is saved as transformed\_data.xlsx.

**3.5 Export Data from Database (Excel Integration)**

* **Power BI Setup**:
  + Connect to the **Excel** file from Power BI by choosing **"Get Data" > Excel**.
  + Load data into Power BI and create relationships if necessary (e.g., between neighborhoods and prices).
* **Data Transformation in Power BI**:
  + Perform further transformations in the **Power Query Editor** if required (e.g., filtering neighborhoods).

**3.6 Deployment**

1. **Publish Dashboards**:
   * After developing interactive dashboards, publish them on **Power BI Service** for stakeholder access.
2. **Scheduled Data Refresh**:
   * Schedule regular refreshes in Power BI to pull updated data from Excel.

**4. Unit Test Cases**

Unit tests ensure the correctness of data transformations and calculations. Below are sample test cases:

| **Test Case ID** | **Description** | **Input** | **Expected Output** | **Status** |
| --- | --- | --- | --- | --- |
| TC\_01 | Handle missing values in reviews\_per\_month | Null values | Filled with 0 | Pass/Fail |
| TC\_02 | Calculate price\_per\_person | Price = 100, Guests = 4 | 25 | Pass/Fail |
| TC\_03 | Detect outliers using IQR | Price list | Outliers identified | Pass/Fail |
| TC\_04 | Export data to Excel | Transformed DataFrame | Excel file generated | Pass/Fail |
| TC\_05 | Power BI Excel connection | Valid Excel file | Data loaded successfully | Pass/Fail |

**Example Unit Test Code in Python:**

python

Copy code

import unittest

import pandas as pd

class TestDataTransformation(unittest.TestCase):

def test\_missing\_values(self):

df = pd.DataFrame({'reviews\_per\_month': [5, None, 2]})

df['reviews\_per\_month'].fillna(0, inplace=True)

self.assertEqual(df['reviews\_per\_month'].sum(), 7)

def test\_price\_per\_person(self):

df = pd.DataFrame({'price': [100], 'accommodates': [4]})

df['price\_per\_person'] = df['price'] / df['accommodates']

self.assertEqual(df['price\_per\_person'][0], 25)

if \_\_name\_\_ == '\_\_main\_\_':

unittest.main()

**Conclusion**

This Low-Level Design (LLD) document provides a detailed, technical breakdown of the Airbnb listing analysis project. It covers data ingestion, transformation, storage, and visualization strategies using Python, Excel, and Power BI. Additionally, it outlines test cases to ensure data quality and process reliability, ensuring smooth deployment and actionable insights.