Data Analysis Assignment for New Joiner

Overview

Welcome to your **Data Analysis Assignment**! This two-week task will help you get hands-on experience with **Pandas, NumPy, Matplotlib, Seaborn, and GitHub**. By the end of this assignment, you will have performed data cleaning, exploratory analysis, and generated meaningful insights from real-world e-commerce sales data.

Objective

This assignment is designed to assess your ability to:

- Load and clean real-world datasets
- Perform exploratory data analysis (EDA)
- Calculate key business KPIs
- Visualize data using **Matplotlib** and **Seaborn**
- Collaborate using **GitHub**
- Automate data updates to **Google Sheets** every **6 hours** using a cron job
- **(Bonus Task)** Deploy and run the analysis using **Djongo framework**

📂 Dataset

Dataset Name: E-commerce Sales Data
Source: Kaggle
Format: CSV
Size: ~10,000 rows

Key Columns:

- Order ID
- Date
- Customer ID
- Product Category
- Product Name
- Quantity Ordered
- Price Each
- Order Total
- Payment Method
- Order Status
- Country
- City

📍 Week 1: Data Understanding & Preprocessing

Task 1: Setup & GitHub Integration

Instructions:

1. **Fork the GitHub repository** (or create a new repo).
2. Clone the repo and create a **Jupyter Notebook (`data_analysis.ipynb`)**.
3. Upload the dataset to the repository.
4. Install required libraries:

```
pip install pandas numpy matplotlib seaborn gspread oauth2client djongo

```

5. Load the dataset using Pandas and print the first few rows.

Deliverables:

- ✅ GitHub repository with dataset and Jupyter Notebook
- ✅ First dataset preview in Jupyter Notebook

Task 2: Data Cleaning & Preprocessing

Instructions:

1. Load the dataset using **Pandas**.
2. Identify missing values, duplicates, and inconsistencies.
3. Handle missing data using **NumPy**:

```
df.fillna(np.nan, inplace=True)

```

4. Convert date fields to proper datetime format.
5. Standardize categorical variables (convert text to lowercase, remove spaces).
6. Create **new useful columns** (e.g., extracting month, weekday from date).

Deliverables:

- ✅ Cleaned dataset stored as `cleaned_data.csv`
- ✅ Jupyter Notebook with Pandas & NumPy cleaning operations
- ✅ GitHub push with updated notebook and cleaned dataset

Task 3: Exploratory Data Analysis (EDA)

Instructions:

1. Perform **descriptive statistics** using `.describe()`.
2. Identify **correlations** using Pandas `.corr()`.

3. **Visualize trends using Matplotlib & Seaborn:**
   - Sales trend over time (line chart)
   - Top-selling products (bar chart)
   - Order distribution by country (pie chart)
   - Payment method usage (bar chart)
4. **Plot a heatmap** to show feature correlations:

```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10,6))
sns.heatmap(df.corr(), annot=True, cmap="coolwarm")
plt.show()

```

Deliverables:

- ✅ At least **4 visualizations** in Jupyter Notebook
- ✅ GitHub push with updated **notebook & insights report (** `EDA_Report.md` **)**

📍 Week 2: Business Insights & KPI Calculation

Task 4: Business Insights & KPI Calculation

Instructions:

1. Define **3–5 key business metrics**, such as:
   - **Total Revenue** = SUM(Order Total)
   - **Total Orders** = COUNT(Order ID)
   - **Average Order Value (AOV)** = Total Revenue / Total Orders
   - **Most profitable city**
   - **Repeat customer rate**
2. Use **Pandas operations** to calculate KPIs.
3. Summarize findings and insights in a markdown report.

Deliverables:

- ✅ Summary of KPIs in Jupyter Notebook
- ✅ GitHub push with `business_insights.md` report

Task 5: Google Sheets Integration & Cron Job

Instructions:

1. Set up **Google Sheets API** authentication.
2. Write a **Python script** to update the Google Sheet every **6 hours** using a cron job:

```
import gspread
from oauth2client.service_account import ServiceAccountCredentials
import pandas as pd

# Authenticate and connect to Google Sheets
scope = ["https://spreadsheets.google.com/feeds", "https://www.googleapis.com/auth/drive"]
creds = ServiceAccountCredentials.from_json_keyfile_name("credentials.json", scope)
client = gspread.authorize(creds)

# Open the Google Sheet and update data
sheet = client.open("Ecommerce Sales Report").sheet1
df = pd.read_csv("cleaned_data.csv")
sheet.update([df.columns.values.tolist()] + df.values.tolist())

```

3. Schedule a **cron job** to run this script every **6 hours**:

```
crontab -e
# Add the following line to schedule every 6 hours
0 */6 * * * /usr/bin/python3 /path/to/update_google_sheets.py

```

Deliverables:

- ✅ Automated Google Sheets update every **6 hours**
- ✅ GitHub push with `update_google_sheets.py` script
- ✅ Documentation on setting up cron job

Bonus Task: Implement the Analysis Using Djongo (Optional)

Instructions:

1. Set up a **Mongodb database with Django**.
2. Create a Django project and configure it to use Django as the database backend.
3. Load the cleaned dataset into the **Mongodb database**.
4. Create a celery worker and beat to run the above task as cronetab .
5. run the task as scheduler for very 6 hr.

Deliverables:

- ✅ Django project with **Django integration**
- ✅ celery task and display dataset
- ✅ GitHub push with project files and documentation

🎯 Evaluation Criteria

✅ **GitHub Collaboration:** Proper commits & organized repo ✅ **Technical Skills:** Pandas, NumPy, Matplotlib, Google Sheets API, Djongo ✅ **Problem-Solving:** Data cleaning, analysis, and insights ✅ **Automation & Reporting:** Scheduled updates to Google Sheets ✅ **Bonus Implementation:** Successful deployment using Djongo

---

Jupyter install using docker in macbook

Steps to Install Jupyter Notebook using Docker on macOS

Running Jupyter Notebook in a **Docker container** ensures a clean, isolated environment without affecting your local system. Below are the **step-by-step instructions** to set up and run Jupyter Notebook using Docker on a MacBook.

📌 Step 1: Install Docker Desktop

1. **Download & Install Docker Desktop** for macOS from the official website:
2. 👉 Docker Desktop for Mac
3. **Enable Docker in macOS:** After installation, open **Docker Desktop** and ensure it is running.

📌 Step 2: Pull Jupyter Docker Image

Docker provides official **Jupyter Notebook images**. You can pull a **base image** or one with popular Python libraries pre-installed.

Option 1: Basic Jupyter Notebook

```
docker pull jupyter/base-notebook

```

Option 2: Jupyter Notebook with SciPy, Pandas, and Matplotlib

```
docker pull jupyter/scipy-notebook

```

Option 3: Jupyter Notebook with Data Science Libraries (Recommended)

```
docker pull jupyter/datascience-notebook

```

> - *The `datascience-notebook` image includes NumPy, Pandas, SciPy, Matplotlib, Scikit-learn, and Seaborn.*

📌 Step 3: Run Jupyter Notebook in Docker

Once the image is downloaded, you can run the container using:

```
docker run -p 8888:8888 -v ~/jupyter:/home/jovyan/work --name jupyter-container jupyter/datascience-notebook

```

- `-p 8888:8888` → Maps port 8888 inside the container to port 8888 on your Mac.

- `-v ~/jupyter:/home/jovyan/work` → Mounts the local `~/jupyter` directory to the container for persistent storage.
- `--name jupyter-container` → Names the container `jupyter-container` for easy management.

📌 Step 4: Access Jupyter Notebook

1. After running the command, Docker will generate a **token-based URL**.
2. Copy and paste the URL (e.g., `http://127.0.0.1:8888/?token=yourtoken`) into your **browser**.
3. You should see the **Jupyter Notebook dashboard**.

📌 Step 5: Stop & Restart Jupyter Notebook

To stop the Jupyter container:

```
docker stop jupyter-container

```

To start the Jupyter container again:

```
docker start jupyter-container

```

To remove the container (if needed):

```
docker rm -f jupyter-container

```

📌 Step 6: Running Jupyter Notebook in Detached Mode

If you want to run Jupyter in the **background (detached mode)**:

```
docker run -d -p 8888:8888 -v ~/jupyter:/home/jovyan/work --name jupyter-container jupyter/datascience-notebook

```

> 🔹 Now, you can **close the terminal**, and Jupyter Notebook will keep running.

📌 Additional Resources

- 📖 Official Jupyter Docker Stacks Documentation:
- 👉 Jupyter Docker Stacks — Docker Stacks documentation
- 📖 Docker Docs for macOS:
- 👉 Mac