Data Analysis Project: Deal Statistics Analysis Tool

Project Title: Deal Statistics Analysis Tool
Duration: September 2023 - November 2023
Technologies: Python, SQL, Excel
Objective: Create a Python-based tool to analyse and visualise deal statistics, including debt-equity ratios, peers multiples, and macro drivers. The tool will extract data from relational databases and unstructured sources to feed the analysis. Automation of data extraction and processing is key to generate detailed reports and visualisations.

Phase 1: Project Planning and Requirement Analysis

Objective Definition:

The primary objective of the Deal Statistics Analysis Tool is to provide comprehensive analysis and visualisation of deal statistics. This includes metrics such as debt-equity ratios, peers multiples, and macro drivers. The tool aims to automate the extraction, processing, and visualisation of data to assist in portfolio and deal analysis.

Scope Definition:

1. Key Metrics:
    a. Debt-Equity Ratios: Measure the proportion of debt and equity in the financial structure of deals.
    b. Peers Multiples: Compare deal metrics against industry peers to assess relative performance.
    c. Macro Drivers: Analyse external economic factors impacting the deals, such as interest rates, GDP growth, and market trends.
2. Data Sources:
    a. Relational Databases: Structured data from corporate databases, such as MySQL or PostgreSQL.
    b. Unstructured Sources: Data from CSV files, JSON files, web scraping, and other semi-structured formats.
3. End Goals:
    a. Visualisations: Develop bar charts, line charts, and heatmaps to represent key metrics.
    b. Reports: Generate detailed Excel or PDF reports summarising the analysis.
    c. Automation: Automate data extraction and processing workflows.

Stakeholder Meetings

Stakeholder Identification:

1. Internal Stakeholders: Portfolio managers, financial analysts, data scientists.
2. External Stakeholders: Clients, regulatory bodies (if applicable).

Meeting Agenda:

Requirement Gathering:

- Discuss the specific metrics and KPIs needed for the analysis.
- Identify data sources and formats.
- Determine the frequency and format of the reports.

Use Case Scenarios:

- Understand how stakeholders will use the tool.
- Identify any specific customization requirements for different user groups.

Technical Requirements:

- Discuss the technical infrastructure (e.g., databases, servers).
- Determine any software or library preferences.

Outcome:

- Detailed list of requirements and expectations from the tool.
- Clarity on data sources, metrics, and reporting needs.
- Alignment on project timeline and milestones.

Technical Setup

Project Repository:

- Create a GitHub repository named deal-statistics-analysis-tool.
- Define a clear folder structure to organise data, scripts, notebooks, and reports.

Folder Structure:

```
deal-statistics-analysis-tool/
├── data/
│   ├── raw/            # Unprocessed data
│   ├── processed/      # Cleaned and transformed data
```

```
├── notebooks/          # Jupyter notebooks for exploratory analysis
├── scripts/            # Python scripts for various stages of the workflow
├── reports/            # Generated reports
├── README.md           # Project overview and instructions
├── requirements.txt    # List of dependencies
└── LICENSE             # License information
```

Dependencies Installation:

pandas
numpy
sqlalchemy
matplotlib
seaborn
beautifulsoup4
requests
plotly
jupyter

Install dependencies using pip:

pip install -r requirements.txt

Initial Setup:

- Initialise the GitHub repository and push the initial folder structure and requirements.txt.
- Set up a virtual environment for the project to manage dependencies.

Tools and Libraries:

- Python: Main programming language for data extraction, processing, and analysis.
- SQLAlchemy: For connecting to and querying relational databases.
- Pandas and NumPy: For data manipulation and numerical operations.
- Matplotlib and Seaborn: For creating visualisations.
- BeautifulSoup and Requests: For web scraping unstructured data.
- Jupyter Notebooks: For exploratory data analysis and prototyping.

Data Extraction and Processing

Database Connection

Establish connections to Relational Databases:

- Identify relevant relational databases containing deal statistics data (e.g., MySQL, PostgreSQL).
- Obtain necessary credentials and permissions to access these databases.

Write SQL Queries:

- Develop SQL queries to extract key metrics from the databases.
- Ensure queries are optimised for performance and accuracy.

Example SQL Query:

```
SELECT
   deal_id,
   deal_date,
   debt_amount,
   equity_amount,
   industry,
   macro_factor
FROM
   deals
WHERE
   deal_date BETWEEN '2022-01-01' AND '2022-12-31';
```

Example Python Code:

```
from sqlalchemy import create_engine
import pandas as pd

# Database connection
engine = create_engine('postgresql://username:password@hostname/database_name')

# SQL query
query = """
SELECT
   deal_id,
   deal_date,
   debt_amount,
   equity_amount,
   industry,
   macro_factor
FROM
   deals
WHERE
   deal_date BETWEEN '2022-01-01' AND '2022-12-31';
"""

# Execute query and store data in DataFrame
df = pd.read_sql(query, engine)
```

Unstructured Data Extraction

Identify and Parse Unstructured Data Sources:

- Determine sources of unstructured data such as CSV files, JSON files, and web pages.
- Use appropriate libraries to parse and extract data.

CSV/Excel Handling:

Use Pandas to read data from CSV or Excel files.

Example code:

```
df_csv = pd.read_csv('data/raw/deals_data.csv')
df_excel = pd.read_excel('data/raw/deals_data.xlsx')
```

Web Scraping:

Use BeautifulSoup and Requests libraries to scrape data from web pages.

Example code:

```
import requests
from bs4 import BeautifulSoup

url = 'https://example.com/deals'
response = requests.get(url)
soup = BeautifulSoup(response.content, 'html.parser')

# Extract data from HTML
deals = []
for row in soup.find_all('tr'):
    columns = row.find_all('td')
    deal = {
        'deal_id': columns[0].text,
        'deal_date': columns[1].text,
        'debt_amount': columns[2].text,
        'equity_amount': columns[3].text,
        'industry': columns[4].text,
        'macro_factor': columns[5].text
    }
    deals.append(deal)

df_web = pd.DataFrame(deals)
```

Data Cleaning and Transformation

Data Cleaning:

Handle missing values by imputing or removing them

Example:

df.fillna(method='ffill', inplace=True)

Remove duplicates to ensure data integrity:

df.drop_duplicates(inplace=True)

Data Transformation:

Normalise and transform data for consistency:

Example:

```
# Convert date columns to datetime format
df['deal_date'] = pd.to_datetime(df['deal_date'])

# Create new columns for analysis
df['debt_equity_ratio'] = df['debt_amount'] / df['equity_amount']
```

Store Cleaned Data:

Save cleaned and transformed data to a structured format for further analysis.

Example code:

df.to_csv('data/processed/cleaned_deals_data.csv', index=False)

Example Cleaning and Transformation Code:

```
# Handling missing values
df.fillna(method='ffill', inplace=True)

# Removing duplicates
df.drop_duplicates(inplace=True)

# Converting date columns to datetime format
df['deal_date'] = pd.to_datetime(df['deal_date'])

# Creating new columns for analysis
df['debt_equity_ratio'] = df['debt_amount'] / df['equity_amount']
```

```
# Saving cleaned data
df.to_csv('data/processed/cleaned_deals_data.csv', index=False)
```

Data Analysis and Visualization

Analysis Tool Development

Develop Python Scripts for Key Metrics:

- Write Python scripts to calculate key metrics such as debt-equity ratios and peers multiples.
- Integrate macroeconomic data for comprehensive analysis.

Example Calculation Script:

```
import pandas as pd

# Load cleaned data
df = pd.read_csv('data/processed/cleaned_deals_data.csv')

# Calculate debt-equity ratio
df['debt_equity_ratio'] = df['debt_amount'] / df['equity_amount']

# Load macroeconomic data (e.g., interest rates, GDP growth)
macro_data = pd.read_csv('data/raw/macro_data.csv')

# Merge macro data with deal data
df_merged = pd.merge(df, macro_data, on='deal_date')

# Calculate peers multiples
industry_avg = df_merged.groupby('industry')['debt_equity_ratio'].mean().reset_index()
df_merged = pd.merge(df_merged, industry_avg, on='industry', suffixes=('', '_industry_avg'))
df_merged['peers_multiple'] = df_merged['debt_equity_ratio'] /
df_merged['debt_equity_ratio_industry_avg']

# Save analysis results
df_merged.to_csv('data/processed/analysis_results.csv', index=False)
```

Incorporate Macroeconomic Data:

- Collect relevant macroeconomic indicators such as interest rates and GDP growth.
- Integrate this data into the analysis to understand external economic factors' impact.

Visualisation Creation

Create Visualization with Matplotlib and Seaborn:

Develop visualisations to represent key metrics and findings clearly.

Example Visualization Code:

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

# Load analysis results
df = pd.read_csv('data/processed/analysis_results.csv')

# Plot debt-equity ratio over time
plt.figure(figsize=(10, 6))
sns.lineplot(data=df, x='deal_date', y='debt_equity_ratio')
plt.title('Debt-Equity Ratio Over Time')
plt.xlabel('Date')
plt.ylabel('Debt-Equity Ratio')
plt.savefig('reports/debt_equity_ratio_over_time.png')
plt.show()

# Plot peers multiples by industry
plt.figure(figsize=(10, 6))
sns.barplot(data=df, x='industry', y='peers_multiple')
plt.title('Peers Multiples by Industry')
plt.xlabel('Industry')
plt.ylabel('Peers Multiple')
plt.xticks(rotation=45)
plt.savefig('reports/peers_multiples_by_industry.png')
plt.show()
```

Develop Interactive Dashboards (Optional):

Use Plotly or Dash for interactive and dynamic visualisations.

Example:

```
import plotly.express as px

fig = px.line(df, x='deal_date', y='debt_equity_ratio', title='Debt-Equity Ratio Over Time')
fig.show()

fig = px.bar(df, x='industry', y='peers_multiple', title='Peers Multiples by Industry')
fig.show()
```

Automate Analysis Pipeline

Automate Data Extraction, Cleaning, and Analysis:

- Schedule scripts to run automatically using cron jobs or Airflow.
- Ensure the pipeline runs at regular intervals to keep the analysis up-to-date.

Example Cron job:

crontab -e

0 0 * * * /usr/bin/python3 /path/to/project/scripts/automation.py

Airflow Example:

Define a DAG (Directed Acyclic Graph) for the automation pipeline.

Example DAG:

```
from airflow import DAG
from airflow.operators.python_operator import PythonOperator
from datetime import datetime, timedelta

def extract_data():
    # Data extraction logic
    pass

def clean_data():
    # Data cleaning logic
    pass

def analyze_data():
    # Data analysis logic
    pass

def visualize_data():
    # Data visualization logic
    pass

default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'start_date': datetime(2022, 9, 1),
    'email_on_failure': False,
    'email_on_retry': False,
    'retries': 1,
    'retry_delay': timedelta(minutes=5),
```

```
}

dag = DAG('deal_statistics_analysis', default_args=default_args, schedule_interval='@daily')

t1 = PythonOperator(task_id='extract_data', python_callable=extract_data, dag=dag)
t2 = PythonOperator(task_id='clean_data', python_callable=clean_data, dag=dag)
t3 = PythonOperator(task_id='analyze_data', python_callable=analyze_data, dag=dag)
t4 = PythonOperator(task_id='visualize_data', python_callable=visualize_data, dag=dag)

t1 >> t2 >> t3 >> t4
```

Reporting and Documentation

Draft report with following structure:

Create Detailed Report:

- Summarise the analysis findings in a clear and structured format.
- Use visual aids such as charts and tables to enhance the readability of the reports.

Example Report Outline:

Executive Summary:

- Overview of the project and key findings.
- Highlight significant insights and their implications.

Introduction:

- Objectives and scope of the analysis.
- Description of data sources and methodologies used.

Data Analysis:

- Detailed analysis of debt-equity ratios.
- Comparative analysis of peers multiples.
- Impact of macroeconomic drivers on deal statistics.

Visualisations:

- Include charts and graphs illustrating key metrics.
- Example visualisations:
    - Line chart of debt-equity ratio over time.
    - Bar chart of peers multiples by industry.
    - Heatmap of macroeconomic factors vs. deal performance.

Conclusion:

- Summarise key takeaways from the analysis.
- Recommendations based on the findings.

Appendices:

- Detailed tables and additional charts.
- Technical details of data processing and analysis scripts.

Generate Reports in Excel or PDF:

- Use Python libraries such as Pandas, Matplotlib, and FPDF/ReportLab to create and format the reports.
- Example code for generating an Excel report:

```
import pandas as pd
import matplotlib.pyplot as plt

# Load analysis results
df = pd.read_csv('data/processed/analysis_results.csv')

# Create an Excel writer object
writer = pd.ExcelWriter('reports/deal_statistics_report.xlsx', engine='xlsxwriter')

# Write data to Excel
df.to_excel(writer, sheet_name='Analysis Results', index=False)

# Access the XlsxWriter workbook and worksheet objects
workbook = writer.book
worksheet = writer.sheets['Analysis Results']

# Create a chart object
chart = workbook.add_chart({'type': 'line'})

# Configure the chart
chart.add_series({
    'name': 'Debt-Equity Ratio',
    'categories': ['Analysis Results', 1, 1, len(df), 1],
    'values': ['Analysis Results', 1, 3, len(df), 3],
})

# Insert the chart into the worksheet
worksheet.insert_chart('H2', chart)

# Save the Excel file
writer.save()
```

Deployment and Maintenance

Deploy the Analysis Tool:

Choose a deployment platform (e.g., AWS, Heroku, Google Cloud) based on project requirements and scalability needs.

Example Deployment on AWS:

Setup EC2 Instance:

- Launch an EC2 instance with appropriate specifications (e.g., t2.medium).
- Install necessary software (Python, pip, virtual environment).

Deploy Code:

Clone the GitHub repository onto the EC2 instance.

Create and activate a virtual environment. Example code:

```
python3 -m venv venv
source venv/bin/activate
```

Install dependencies:

```
pip install -r requirements.txt
```

Schedule Scripts:

Use cron jobs to automate the execution of data extraction, cleaning, analysis, and visualisation scripts.

Example cron job setup:

```
crontab -e
```

Add the following lines to schedule scripts:

```
0 0 * * * /usr/bin/python3 /path/to/project/scripts/data_extraction.py
1 0 * * * /usr/bin/python3 /path/to/project/scripts/data_cleaning.py
2 0 * * * /usr/bin/python3 /path/to/project/scripts/analysis.py
3 0 * * * /usr/bin/python3 /path/to/project/scripts/visualization.py
```

Set Up CI/CD:

Use a CI/CD tool like GitHub Actions or Jenkins to automate deployment and updates.

Example GitHub Actions workflow:

```
name: CI/CD Pipeline

on:
  push:
    branches:
      - main

jobs:
  build-and-deploy:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v2

      - name: Set up Python
        uses: actions/setup-python@v2
        with:
          python-version: 3.8

      - name: Install dependencies
        run: |
          python -m venv venv
          source venv/bin/activate
          pip install -r requirements.txt

      - name: Run tests
        run: |
          source venv/bin/activate
          python -m unittest discover

      - name: Deploy to EC2
        run: |
          scp -i /path/to/key.pem -r ./* ec2-user@your-ec2-instance:/path/to/deployment
          ssh -i /path/to/key.pem ec2-user@your-ec2-instance "cd /path/to/deployment &&
./deploy.sh"
```

Training and Support

Train End-Users:

- Conduct training sessions for end-users to demonstrate how to use the analysis tool.
- Provide comprehensive user manuals and video tutorials.

User Manual Outline:

Introduction:

- Overview of the tool and its features.

Getting Started:

- Installation and setup instructions.

Using the Tool:

- Step-by-step guide to running the scripts and generating reports.

Interpreting Results:

- Explanation of key metrics and visualisations.

Troubleshooting:

- Common issues and solutions.

Example Training Video Topics:

- Setting up the environment.
- Running data extraction and cleaning scripts.
- Performing data analysis and generating visualisations.
- Interpreting the analysis results.

Provide Support:

- Establish a support channel (e.g., email, Slack) for users to report issues and seek help.
- Regularly update the tool based on user feedback and emerging needs.

Feedback and Iteration

Collect Feedback:

- Gather feedback from users through surveys, feedback forms, and direct interactions.
- Analyse the feedback to identify areas for improvement.

Example Feedback Questions:

- How easy is it to use the tool?
- Are the visualisations clear and helpful?
- What additional features would you like to see?
- Have you encountered any issues or bugs?

Iterate and Improve:

- Prioritise feedback and implement necessary changes in iterative development cycles.
- Release updated versions of the tool regularly, ensuring continuous improvement.

Version Control:

- Use Git for version control to track changes and manage releases.

- Maintain a changelog in the GitHub repository to document updates and improvements.

THE END