CAPSTONE PROJECT 1

Power Sector Generation Performance & Gap Analysis in India (2018-2024)

INDEX

1. Title Page

Project Title: Power Sector Generation Performance & Gap Analysis in India
(2018-2024).
Submitted by: Rachuri Ramesh, PGA 46, Imarticus Learning, Hyderabad.
Tools Used: Excel, Python, SQL, Power BI.

2. Abstract / Executive Summary

This project demonstrates the end-to-end development of an interactive dashboard for analyzing outage patterns, coal stock trends, and generation performance across thermal and hydro power stations in Northern India.

Data was sourced from Kaggle, cleaned and processed using Python, further transformed with SQL, and finally visualized in Power BI. The final dashboards provide stakeholders with actionable insights into generation gaps, outage frequencies, and coal dependencies, supporting operational efficiency and strategic planning.

3. Introduction

Background

Power generation reliability in Northern India faces challenges due to frequent outages and fluctuating coal stock availability. These issues not only affect grid stability but also hinder efficient power supply to consumers.

Need for Dash boarding

Currently, performance and outage data are primarily tracked through **static Excel reports**, which lack **real-time updates, interactivity, and drill-down analysis**. This limits decision-makers' ability to quickly identify problem areas, compare performance across stations, and forecast resource requirements.

Objective

The project aims to build an **interactive, data-driven dashboard** using **Python (for preprocessing and EDA), SQL (for structured data storage and queries), and Power BI (for visualization)**. The dashboard will provide:

- **Unit-level insights** (availability, outages, coal stock usage).
- **Station-level comparisons** (generation trends, reliability, efficiency).
- **Interactive filters and KPIs** for decision-makers to monitor and act in real time.

4. Problem Statement

- **Issue**: Data from multiple sources is fragmented and inconsistent.
- **Challenge**: Outages and coal stock variations are not effectively visualized for decision-making.
- **Goal**: Create a consolidated, interactive dashboard with real-time insights into capacity utilization, outage types, and coal stock levels.

5. Data Sources

- **Primary Data**: Kaggle dataset (https://www.kaggle.com/datasets/vangap/india-power-generation)(6 Excel/CSV files consolidated into one workbook).
- **Attributes**: region, sector, state, station, station_type, unit.

6. Methodology

Step 1: Data Preparation in Excel

- All six Kaggle CSV/Excel files were merged into a single Excel workbook.
- Initial formatting and consistency checks were performed.

Step 2: Data Cleaning & EDA in Python

- Used **Pandas & NumPy** for missing value treatment (Python File).
- Performed exploratory data analysis (EDA):
  - Null value heat map
  - Outage frequency distribution
  - Coal stock descriptive statistics
- Data exported to SQL-ready format.

Step 3: SQL Data Transformation

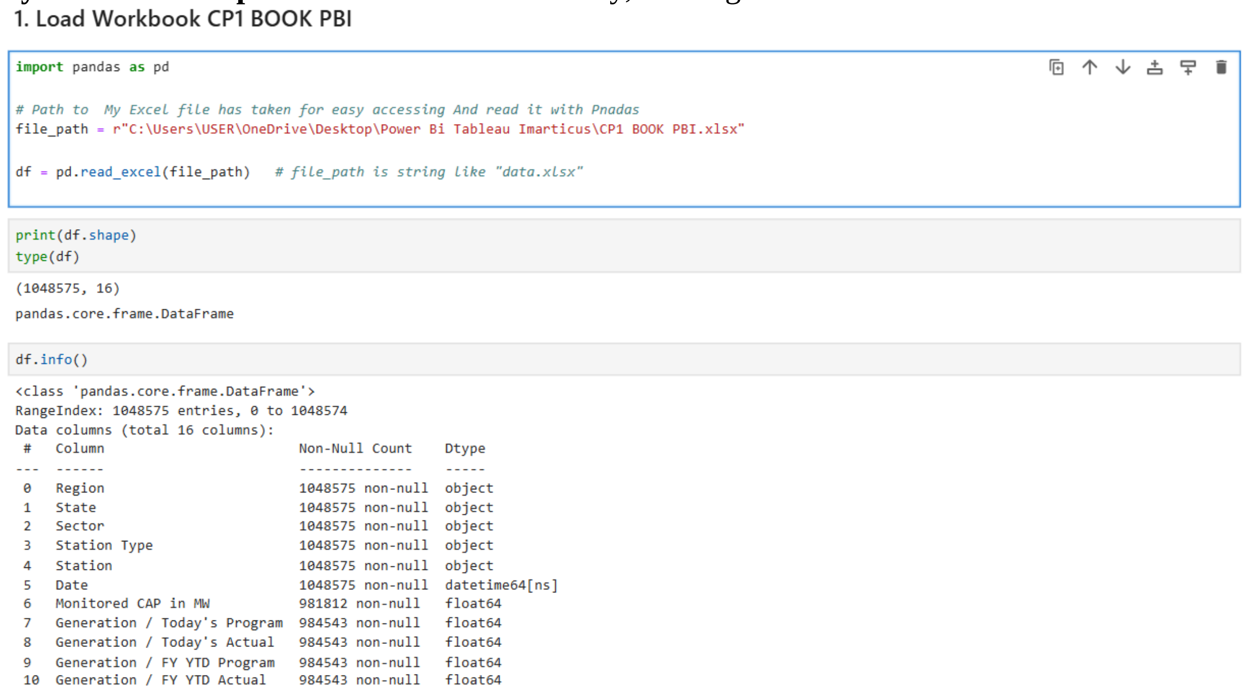
- Imported cleaned dataset into SQL database.
- Performed operations:
  - Aggregation (generation by sector/state)
  - Filtering outage types
  - Joins across unit/station tables
- Generated structured tables for visualization.

Step 4: Dashboard Development in Power BI

- Connected SQL database to Power BI Dash Board.
- Designed dashboards with:
  - **KPIs**: capacity utilization, scheduled vs. actual generation, coal stock levels
  - **Filters/Slicers**: state, sector (thermal/hydro), outage type
  - **Visuals**: line charts (trends), bar charts (sector comparison), maps (state-wise outages), donut charts (outage type contribution)

7. Implementation (Screenshots to be added by you)

- **Python EDA Outputs**: Null treatment summary, missing values handled.



1. Load Workbook CP1 BOOK PBI

```
import pandas as pd

# Path to  My Excel file has taken for easy accessing And read it with Pnadas
file_path = r"C:\Users\USER\OneDrive\Desktop\Power Bi Tableau Imarticus\CP1 BOOK PBI.xlsx"

df = pd.read_excel(file_path)   # file_path is string like "data.xlsx"
```

```
print(df.shape)
type(df)
```

```
(1048575, 16)
pandas.core.frame.DataFrame
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 16 columns):
 #   Column                       Non-Null Count    Dtype
---  ------                       --------------    -----
 0   Region                       1048575 non-null  object
 1   State                        1048575 non-null  object
 2   Sector                       1048575 non-null  object
 3   Station Type                 1048575 non-null  object
 4   Station                      1048575 non-null  object
 5   Date                         1048575 non-null  datetime64[ns]
 6   Monitored CAP in MW          981812 non-null   float64
 7   Generation / Today's Program 984543 non-null   float64
 8   Generation / Today's Actual  984543 non-null   float64
 9   Generation / FY YTD Program  984543 non-null   float64
 10  Generation / FY YTD Actual   984543 non-null   float64
```

```python
import pandas as pd

# Drop fully empty columns
df = df.drop(columns=['Outage Date', 'Expected Date / Sync Date', 'Remarks'])



# Handle partially null columns
df['CAP under outage'] = df['CAP under outage'].fillna(0)  # no outage = 0
df['Coal Stock in Days'] = df['Coal Stock in Days'].fillna(df['Coal Stock in Days'].median())
df['Monitored CAP in MW'] = df['Monitored CAP in MW'].fillna(df['Monitored CAP in MW'].median())
df["Generation / Today's Program"] = df["Generation / Today's Program"].fillna(df["Generation / Today's Program"].median())
df["Generation / Today's Actual"] = df["Generation / Today's Actual"].fillna(df["Generation / Today's Actual"].median())
df['Generation / FY YTD Program'] = df['Generation / FY YTD Program'].fillna(df['Generation / FY YTD Program'].median())
df['Generation / FY YTD Actual'] = df['Generation / FY YTD Actual'].fillna(df['Generation / FY YTD Actual'].median())

# Verify the nulls in df
print(df.isnull().sum())
```

```
Region                          0
State                           0
Sector                          0
Station Type                    0
Station                         0
Date                            0
Monitored CAP in MW             0
Generation / Today's Program    0
Generation / Today's Actual     0
Generation / FY YTD Program     0
Generation / FY YTD Actual      0
Coal Stock in Days              0
CAP under outage                0
```

Observations:

1. Each row represents power generation data for a station on a specific date.
2. Columns like Monitored CAP in MW, Generation / Today's Program/Actual, Coal Stock in Days, and CAP under outage are numeric and ready for aggregation.
3. Categorical columns like Region, State, Sector, Station Type can be used for grouping and filtering in SQL queries.
4. The dataset is cleaned, no nulls in numeric columns, and no duplicates.

- Ready to be moved into SQL for aggregation queries to support Power BI dashboards.

Export Cleaned Data into sql server for next operations:

```python
from sqlalchemy import create_engine
import pyodbc
```

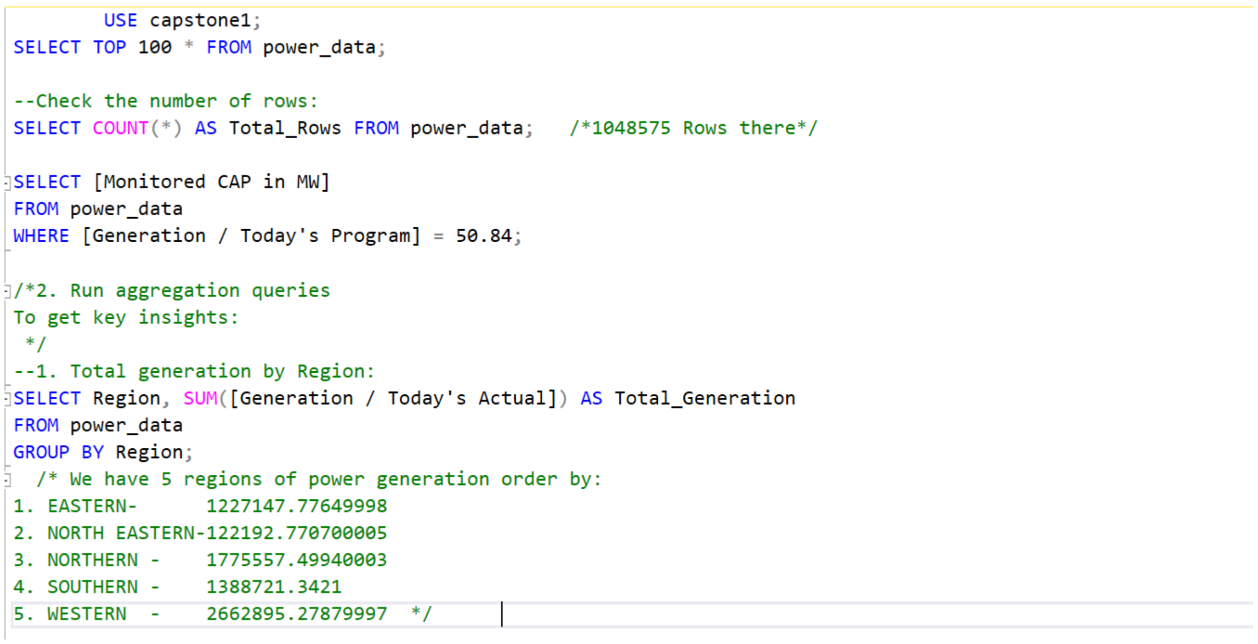
1. Connect Python to SQL Server

```python
from sqlalchemy import create_engine
import urllib

# SQL Server connection details
server = 'DESKTOP-I3RMHOT'
database = 'capstone1'

# Connection string using Windows Authentication
params = urllib.parse.quote_plus(
    f"DRIVER={{ODBC Driver 17 for SQL Server}};"
    f"SERVER={server};"
    f"DATABASE={database};"
    f"Trusted_Connection=yes;"
)

# Create engine
engine = create_engine(f"mssql+pyodbc:///?odbc_connect={params}")

# Store DataFrame in SQL Server
df.to_sql("power_data", engine, if_exists="replace", index=False)

print("DataFrame successfully stored in SQL Server using Windows Authentication!")
```

```
DataFrame successfully stored in SQL Server using Windows Authentication!
```

After cleaning, the DataFrame was successfully stored in SQL Server (database: capstone1, table: power_data) using Windows Authentication. This enables running SQL queries for aggregation, filtering, and grouping, which will be used as the data source for Power BI dashboards and visualizations.

- **SQL Queries**: Example aggregation query for coal stock trends.



```
        USE capstone1;
SELECT TOP 100 * FROM power_data;

--Check the number of rows:
SELECT COUNT(*) AS Total_Rows FROM power_data;   /*1048575 Rows there*/

SELECT [Monitored CAP in MW]
FROM power_data
WHERE [Generation / Today's Program] = 50.84;

/*2. Run aggregation queries
To get key insights:
 */
--1. Total generation by Region:
SELECT Region, SUM([Generation / Today's Actual]) AS Total_Generation
FROM power_data
GROUP BY Region;
  /* We have 5 regions of power generation order by:
1. EASTERN-      1227147.77649998
2. NORTH EASTERN-122192.770700005
3. NORTHERN -    1775557.49940003
4. SOUTHERN -    1388721.3421
5. WESTERN  -    2662895.27879997  */
```

- **Power BI Dashboards**: Screenshots of final visualizations with explanations.

  [AS attached Power Bi file]

8. Results & Analysis

- **Coal Stock Insights**: Certain states showed critical dependency on limited reserves.
- **Outage Patterns**: Reserve shutdowns contributed to major generation gaps.
- **Generation vs. Schedule**: Deviations were higher in thermal compared to hydro.

9. Discussion

- **Benefits**:
  - Improved visibility into outage causes
  - State-wise performance comparison
  - Decision support for coal stock planning
- **Tool Comparison**: Python (data cleaning) + SQL (structured processing) + Power BI (visual storytelling) created a strong pipeline.

10. Risks, Challenges, and Limitations

- Incomplete or inconsistent entries across some stations.

- Limited granularity in outage classification.
- Dependency on historical Kaggle dataset (not real-time).

11. Conclusion

The project successfully transformed raw Kaggle datasets into interactive dashboards. By integrating **Python, SQL, and Power BI**, the workflow achieved reliable data cleaning, structured transformation, and impactful visualization. These dashboards enable stakeholders to monitor coal reserves, identify outage causes, and optimize generation schedules in Northern India.

12. Future Enhancements (Brief Explanation)

1. **Automate daily data refresh with scheduled Python scripts**
    - Currently, the dashboards depend on manual data preparation and uploads. By automating the data pipeline using Python scripts and scheduling them with tools like **Windows Task Scheduler** or **Airflow**, the dataset can be refreshed daily or hourly. This ensures that the dashboards always display the latest operational data without human intervention.
2. **Expand dashboards to cover renewable generation (solar/wind)**
    - The current project focuses mainly on **thermal and hydro power plants**. In future, including **solar and wind generation data** will give a more holistic view of the regional energy mix. This will help stakeholders analyze renewable penetration, seasonal variability, and compare renewable vs. non-renewable performance.
3. **Use Machine Learning to predict outages and coal shortages**
    - With enough historical outage and coal stock data, predictive models can be trained (e.g., regression, time-series forecasting, or classification models) to **anticipate shortages or failures** before they occur.
    - For example:
        - Predicting when coal stock will fall below critical levels.
        - Identifying patterns in forced or reserve outages.
        - Forecasting demand–supply gaps under different scenarios.
    - This predictive capability can help improve **proactive decision-making** and resource allocation.

13. References

- Kaggle Dataset India Power Generation
- Excel Work book
- Python & SQL libraries
- Microsoft Power BI Documentation