Appendix

Software and Tools

1. **Microsoft Power BI Desktop:** Used it for data cleaning, modeling, and creating all visualizations for the report.
2. **Jupyter Notebook:** Used it for writing and executing the Python code for predictive analysis.
3. **Python Libraries:**

- pandas (for data cleaning and analysis)
- numpy (for numerical operations)
- scikit-learn (for building the linear regression model)
- matplotlib (for creating the scatter plot visualization)

Dataset

1. **Employee Name:** The name of the Business Development associate.
2. **Day:** The day of the week the lead generation activities occurred.
3. **Date:** The specific date of the lead generation activities.
4. **Leads:** The total number of leads generated on a given day.
5. **Time spent on LG (mins):** The total time spent on lead generation activities.
6. **Avg Time Per Lead (mins):** The calculated average time spent to generate a single lead.
7. **Daily Team Review:** Indicates whether a team review was attended or missed on a specific day.
8. **No. of Incomplete Leads:** The number of leads that were not completed or fully qualified on a given day.

Data Cleaning and Preparation Steps

1. Combined the three separate Excel files (for Ali, Arya, and Raj) into a single, comprehensive table.
2. Handled rows with all values missing by dropping them that had no leads, as these would have manipulated our analysis
3. Filled the empty cell in the Daily Team Review column with "attended" using **mode imputation method**
4. Created a new Day Type column to categorize each day as a "Weekday" or "Weekend" for analysis.
5. Also converted the Date column to a date format, numeric columns were converted to whole numbers, and the Day column was converted to text string.

Python Code for Linear Regression model

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,  r2_score

# Loading Excel file
file = r"C:\Users\suraj\Desktop\Test.xlsx"
df = pd.read_excel(file_path)
# Storing required columns in a dataframe
df = df[["Employee Name", "Leads", "Time spent on LG (mins)"]]

#preparing the features (X) and the target variable (y) for a machine learning model
X = df[["Time spent on LG (mins)"]]
y = df["Leads"]

# Training linear regression model
model = LinearRegression()
model.fit(X, y)

#  predicts leads from time spent using the trained model, and stores these predictions in a new column.
df["Predicted Leads"] = model.predict(X)
# Calculating the difference (error) between actual and predicted leads
df["Error"] = df["Leads"] - df["Predicted Leads"]

# Calculating % Error
df["% Error"] = ((df["Leads"] - df["Predicted Leads"]) / df["Leads"]) * 100

# Calculating overall accuracy metrics (MSE and R²) for the whole dataset
mse = mean_squared_error(y, df["Predicted Leads"])
r2 = r2_score(y, df["Predicted Leads"])
```

```
metrics_df = pd.DataFrame({
    "Metric": ["Mean Squared Error", "R² Score"],
    "Value": [mse, r2]
})
# Calculating RMSE and R² for each employee separately
employee_metrics = []
for emp, group in df.groupby("Employee Name"):
    y_true = group["Leads"]
    y_pred = group["Predicted Leads"]
# RMSE = root mean squared error. Lower mean square error is considered as good prediction
  rmse = np.sqrt(mean_squared_error(y_true, y_pred))
  r2 = r2_score(y_true, y_pred) if len(group) > 1 else None  # avoid error if single row

    employee_metrics.append({
        "Employee Name": emp,
        "RMSE": rmse,
        "R²": r2
    })
employee_metrics_df = pd.DataFrame(employee_metrics)
#Building a summary comparison table for each employee
comparison_df = df.groupby("Employee Name").agg({
    "Leads": "sum",
    "Predicted Leads": "mean",
    "Error": "mean",
    "% Error": "mean"
}).reset_index()

# Printing Tables
print("Model Performance Metrics (Overall):")
print(metrics_df, "\n")

print(" Per-Employee RMSE and R²:")
print(employee_metrics_df, "\n")
print("Per-Employee Comparison:")
print(comparison_df, "\n")
```

 # Plotting actual leads vs predicted leads for visualization

```
plt.figure(figsize=(8,5))
plt.scatter(X, y, color="blue", label="Actual")
plt.plot(X, df["Predicted Leads"], color="red", linewidth=2, label="Predicted")
plt.xlabel("Time spent on Lead Generation (mins)")
plt.ylabel("Leads Generated")
plt.title("Predictive Analysis: Time vs Leads")
plt.legend()
plt.show()
```

Output:

```
Model Performance Metrics (Overall):
               Metric     Value
0  Mean Squared Error  6.275739
1            R² Score  0.100859

 Per-Employee RMSE and R²:
  Employee Name      RMSE        R²
0           Ali  2.696080  0.193683
1          Arya  2.251647 -0.116976
2           Raj  2.547245  0.055336

Per-Employee Comparison:
  Employee Name  Leads  Predicted Leads     Error    % Error
0           Ali    488        11.672387  0.230052  -4.419519
1          Arya    474        10.641431  0.919545   3.967337
2           Raj    447        12.052035 -1.149596 -16.570057
```