Report: Comprehensive EDA Report

This **Exploratory Data Analysis (EDA)** focuses on understanding the stock price dataset, extracting insights, and determining relevant features for predictive modeling. Below is the detailed report that covers visualizations, trends, seasonality, anomalies, feature selection, and data preprocessing decisions.

1. Visualizations of Key Patterns and Relationships in the Data

a. Visualizing the Closing Price Over Time

To understand the overall behavior of the stock, we begin by plotting the closing price over time. This allows us to visualize trends, fluctuations, and possible patterns.

```
import matplotlib.pyplot as plt

# Assuming 'df' is the DataFrame and 'Date' and 'Close' are the
relevant columns
plt.figure(figsize=(12,6))
plt.plot(df['Date'], df['Close'], label='Closing Price', color='blue')
plt.title('Stock Closing Price Over Time')
plt.xlabel('Date')
plt.ylabel('Closing Price')
plt.xticks(rotation=45)
plt.legend()
plt.show()
```

This line plot displays the stock's **closing price** over the entire time period. From this, we can observe:

Trends: Is the stock price moving upward or downward in the long term?
Volatility: How much the stock price fluctuates day-to-day.
Periods of Stability or Decline: Any long-term periods where the stock shows limited movement.

b. Moving Averages (Smoothing the Series)

Next, we visualize moving averages to capture the general trend while smoothing out short-term fluctuations. The **5-day** and **20-day** moving averages will help reveal the short-term and medium-term trends.

```
# Calculate moving averages
df['SMA_5'] = df['Close'].rolling(window=5).mean()
df['SMA_20'] = df['Close'].rolling(window=20).mean()

# Plot closing price with moving averages
plt.figure(figsize=(12,6))
plt.plot(df['Date'], df['Close'], label='Closing Price', color='blue',
alpha=0.5)
plt.plot(df['Date'], df['SMA_5'], label='5-Day Moving Average',
color='green', linewidth=2)
plt.plot(df['Date'], df['SMA_20'], label='20-Day Moving Average',
color='red', linewidth=2)
plt.title('Stock Closing Price with Moving Averages')
plt.xlabel('Date')
plt.ylabel('Price')
plt.xticks(rotation=45)
plt.legend()
plt.show()
```

5-Day Moving Average: Captures short-term fluctuations, showing the immediate trend.
20-Day Moving Average: A smoother curve that identifies medium-term trends, revealing overall stock behavior.

c. Correlation Heatmap

To identify relationships between different features, we generate a **correlation heatmap**. This helps us understand how variables like `Open`, `Close`, `High`, `Low`, and `Volume` correlate with each other.

```
import seaborn as sns

# Calculate correlation matrix
```

```
correlation_matrix = df.corr()

# Plot the heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
fmt='.2f', linewidths=0.5)
plt.title("Correlation Matrix of Features")
plt.show()
```

Key observations might include:

Close vs. Open: The **Close** and **Open** prices usually have a high correlation, indicating that the opening price has a significant influence on the closing price.
Volume: The Volume (number of shares traded) may not always correlate strongly with the stock price itself, but it might be indicative of market activity.

2. Analysis of Trends, Seasonality, and Anomalies

a. Trend and Seasonality Analysis

We can decompose the time series into **trend**, **seasonality**, and **residuals** to understand underlying patterns.

```
import statsmodels.api as sm

# Convert Date to datetime and set as index
df['Date'] = pd.to_datetime(df['Date'])
df.set_index('Date', inplace=True)

# Decompose the time series
decomposed = sm.tsa.seasonal_decompose(df['Close'],
model='multiplicative', period=30)
decomposed.plot()
plt.show()
```

The decomposition will produce:

Trend: The general direction of the stock price over time (upward or downward).
Seasonality: Any periodic fluctuations. In stock data, seasonality could appear if there are cyclical patterns due to the time of the year or other factors.
Residuals: The "noise" or randomness in the data that isn't explained by trend or seasonality.

b. Anomaly Detection

Anomalies can be defined as points where the price deviates significantly from the expected trend. We can detect anomalies by calculating the **rolling mean** and **standard deviation** to flag outliers.

```
# Calculate rolling mean and standard deviation
df['Rolling_Mean'] = df['Close'].rolling(window=20).mean()
df['Rolling_Std'] = df['Close'].rolling(window=20).std()

# Detect anomalies (any points more than 2 standard deviations away
from the rolling mean)
df['Anomalies'] = (df['Close'] > df['Rolling_Mean'] + 2 *
df['Rolling_Std']) | (df['Close'] < df['Rolling_Mean'] - 2 *
df['Rolling_Std'])

# Plot anomalies
plt.figure(figsize=(12,6))
plt.plot(df['Date'], df['Close'], label='Closing Price')
plt.scatter(df[df['Anomalies']].index, df[df['Anomalies']]['Close'],
color='red', label='Anomalies')
plt.title('Stock Closing Price with Anomalies')
plt.xlabel('Date')
plt.ylabel('Price')
plt.xticks(rotation=45)
plt.legend()
plt.show()
```

Anomalies are flagged as points that are more than 2 standard deviations away from the rolling mean. This can help identify periods of unexpected volatility, such as earnings announcements or major market events.

3. Justification for Feature Selection Choices

The following features were selected to build the predictive model:

a. Lagged Features (Previous Prices)

Lag-1, **Lag-2**, **Lag-3**, etc., represent the stock prices on previous days. These features are critical because **past prices** are strong predictors of future prices. In time series forecasting, past data is often the most relevant information for predicting future values.

b. Moving Averages

SMA_5 (5-day Simple Moving Average) and **SMA_20 (20-day Simple Moving Average)** were selected as features to capture short-term and medium-term trends in stock prices. Moving averages are useful for smoothing out noise and revealing the underlying trend.

c. Volume

Volume (number of shares traded) is included because it can serve as an indicator of market sentiment. Higher trading volumes may indicate stronger investor interest and could correlate with stock price movements.

d. High/Low Prices

The **High** and **Low** prices of the day can provide additional context to the closing price, revealing volatility within the day.

4. Clear Documentation of Data Preprocessing Decisions

a. Handling Missing Data

The dataset may contain **missing values**. These missing values were handled by **forward filling** or **backward filling**, depending on the situation.

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

For any rows with missing values in critical features, we may also consider dropping those rows if they cannot be reasonably imputed.

b. Feature Engineering

Lagged Features: We created features representing the stock's price at earlier time points (e.g., `Lag-1`, `Lag-2`, etc.).

```
df['Lag_1'] = df['Close'].shift(1)
df['Lag_2'] = df['Close'].shift(2)
```

Moving Averages: We added simple moving averages (`SMA_5`, `SMA_20`) to capture short-term and long-term trends.

```
df['SMA_5'] = df['Close'].rolling(window=5).mean()
df['SMA_20'] = df['Close'].rolling(window=20).mean()
```

c. Handling Outliers

Anomalies were detected using the **rolling mean** and **standard deviation** method. Points that deviated more than 2 standard deviations from the rolling mean were flagged as anomalies.

Outlier Treatment: If anomalies are detected, we could choose to **remove** them or **replace** them with the rolling mean for further analysis.

d. Feature Scaling

Since stock prices can have vastly different scales (e.g., high-priced vs. low-priced stocks), it may be necessary to **normalize** or **standardize** features, particularly for machine learning models that are sensitive to feature scales (e.g., Linear Regression or Neural Networks).

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
df[['Close', 'Volume']] = scaler.fit_transform(df[['Close',
'Volume']])
```

Conclusion

The **EDA** has provided a detailed understanding of the stock price data, revealing important insights such as trends, seasonality, and anomalies. The selected features, including lagged features and moving averages, are based on strong domain knowledge and are expected to enhance the predictive power of the model. Key preprocessing steps, such as handling missing data, feature engineering, and outlier detection, ensure the quality of the data before model training.

With this EDA complete, we are ready to move on to the next step: **model development and evaluation**.