QUESTION 02

02 - A)

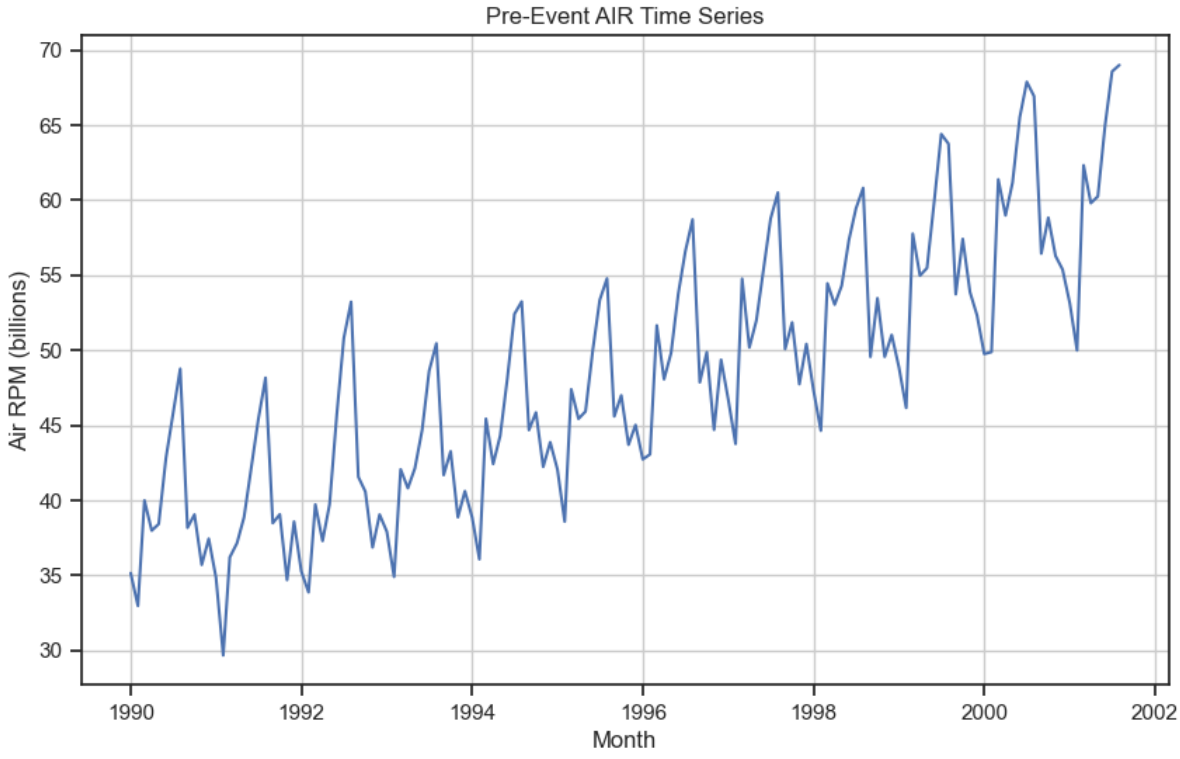
In [18]:
```
import pandas as pd
import numpy as np
from datetime import datetime
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

# Load the data from the CSV file
data = pd.read_csv('air_miles.csv')QUESTION 02
```

In [47]:
```
data['Month'] = pd.to_datetime(data['Month'], format='%b-%y')

df = data[data['Month'] < pd.to_datetime('2001-09-01')]
```

In [48]:
```
plt.figure(figsize=(10, 6))
plt.plot(df['Month'], df['Air RPM (billions)'])
plt.xlabel('Month')
plt.ylabel('Air RPM (billions)')
plt.title('Pre-Event AIR Time Series')
plt.grid(True)
plt.show()
```



The graph shows a definite upward trend, which denotes steady progress over time. Additionally, there are periodic recurring patterns that point to substantial seasonality. These trends imply that modelling could benefit from using linear regression with trend and seasonality. However, the use of linear regression is encouraged since it successfully captures these observable trends and seasonal changes in the absence of significant abnormalities.

02 - B

In [59]:
```
#Calculating Seasonality
monthly_means = df.groupby(df['Month'].dt.month)['Air RPM'].mean()
df = df.join(monthly_means, on=df['Month'].dt.month, rsuffix='_mean
```

In [50]:
```
df['Seasonally_Adjusted'] = df['Air RPM'] - df['Air RPM_mean']
```

In [51]:
```
#Creating Independent Variable and Split Data
df['Month_Index'] = range(1, len(df) + 1)
X = df[['Month_Index']]
y = df['Seasonally_Adjusted']
```

In [52]:
```
#Fit Linear Regression Model
model = LinearRegression()
model.fit(X, y)
```

Out[52]:
```
LinearRegression()
```

In [53]:
```
df['Predicted_Seasonally_Adjusted'] = model.predict(X)
```

In [54]:
```
df['Predicted_Air_RPM'] = df['Predicted_Seasonally_Adjusted'] + df[
```

In [55]:
```
#Evaluation Metrics
mse = np.mean((df['Predicted_Air_RPM'] - df['Air RPM'])**2)
rmse = np.sqrt(mse)
ssr = np.sum((df['Predicted_Air_RPM'] - df['Air RPM'].mean())**2)
sst = np.sum((df['Air RPM'] - df['Air RPM'].mean())**2)
r_squared = ssr / sst
mape = np.mean(np.abs((df['Air RPM'] - df['Predicted_Air_RPM']) / d

print(f'Mean Squared Error (MSE): {mse:.2f}')
print(f'Root Mean Squared Error (RMSE): {rmse:.2f}')
print(f'R-squared (Coefficient of Determination): {r_squared:.2f}')
print(f'Mean Absolute Percentage Error (MAPE): {mape:.2f}%')
```

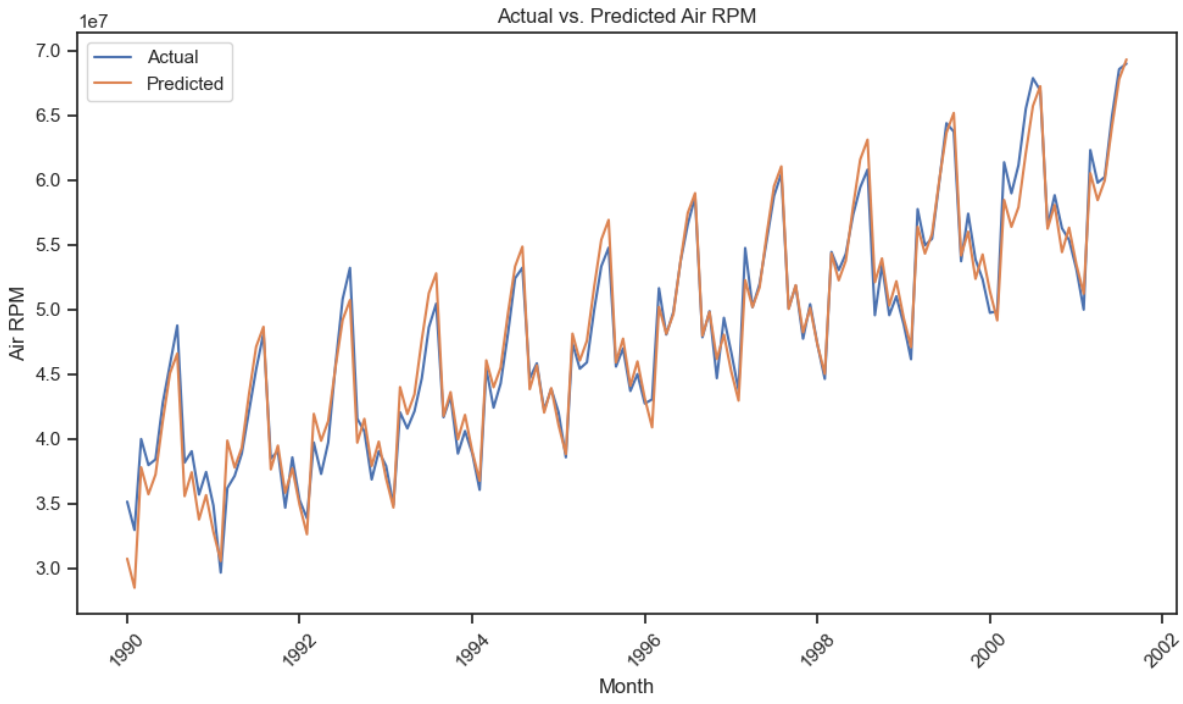
```
Mean Squared Error (MSE): 2204908010075.72
Root Mean Squared Error (RMSE): 1484893.27
R-squared (Coefficient of Determination): 1.01
Mean Absolute Percentage Error (MAPE): 2.55%
```

- The model's predictions will inaccurate and have substantial mistakes, as shown by the high MSE and RMSE values.

- A problem of overfitting is indicated by the R-squared value being above 1, which means that the model is not capturing the underlying patterns but is instead fitting noise.

- Although the MAPE of 2.55% looks appropriate, additional metrics should also be taken into account. It could conceal occasions where the model's predictions differ noticeably from the observed data.

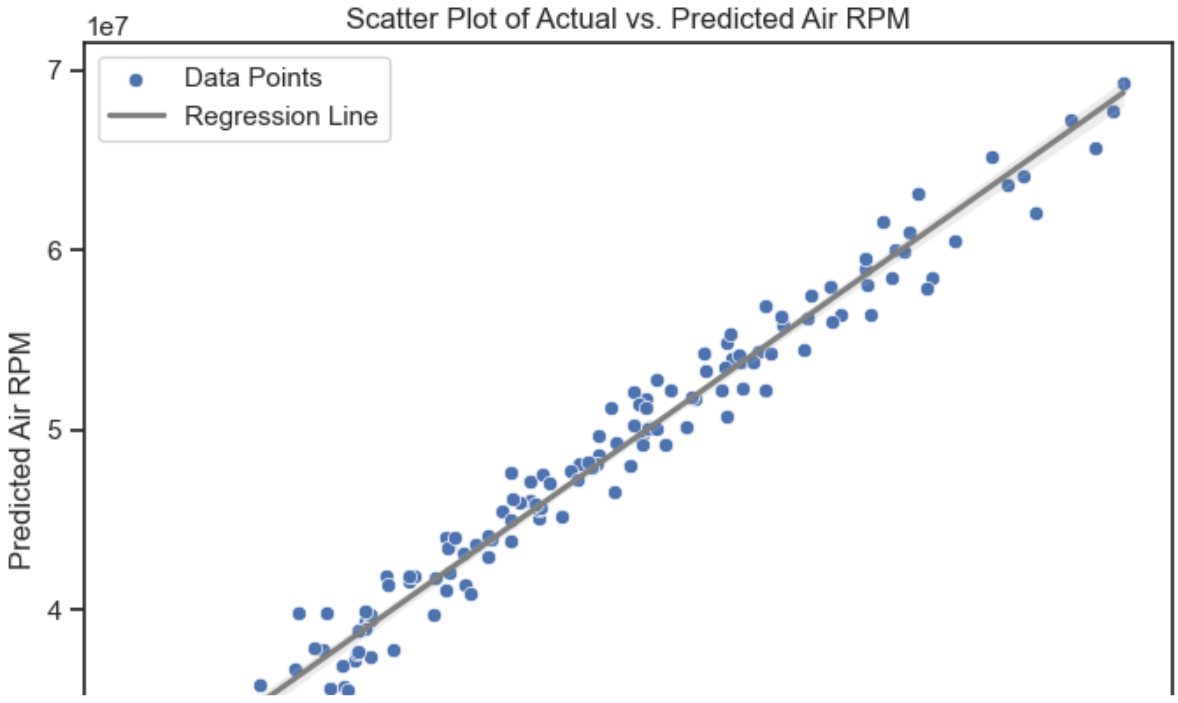
In [56]:
```
plt.figure(figsize=(10, 6))
plt.plot(df['Month'], df['Air RPM'], label='Actual')
plt.plot(df['Month'], df['Predicted_Air_RPM'], label='Predicted')
plt.xlabel('Month')
plt.ylabel('Air RPM')
plt.title('Actual vs. Predicted Air RPM')
plt.legend()
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



The Linear Regression Model with Trend and Seasonality is the best forecasting technique for this circumstance, according to the provided code and data. The seasonality is calculated, a linear regression model is used to capture the trend and seasonality, and several indicators are used to assess the model's success. Both the trend and the seasonal patterns seen in the data are intended to be accounted for by this method.

In [57]:
```
import matplotlib.pyplot as plt
import seaborn as sns

sns.set(style="ticks")
plt.figure(figsize=(8, 6))
sns.scatterplot(x=df['Air RPM'], y=df['Predicted_Air_RPM'], color='
sns.regplot(x=df['Air RPM'], y=df['Predicted_Air_RPM'], scatter=Fal
plt.xlabel('Actual Air RPM')
plt.ylabel('Predicted Air RPM')
plt.title('Scatter Plot of Actual vs. Predicted Air RPM')
plt.legend(['Data Points','Regression Line', ])
plt.show()
```



Each blue dot's position in relation to the regression line represents the observation's prediction error. As opposed to those that deviate further from the line, which indicate bigger prediction errors, the dots that are near to the line reflect accurate forecasts.