Exercise 5 : Perform ARIMA on air pollution data

Step 1 : Read test CSV File with PM2.5 Values

In [6]:
```
import pandas as pd
import matplotlib.pyplot as plt


train_data = pd.read_csv("30201130PM25_trainex5.csv")
test_data = pd.read_csv("30201130PM25_testex5.csv")
```

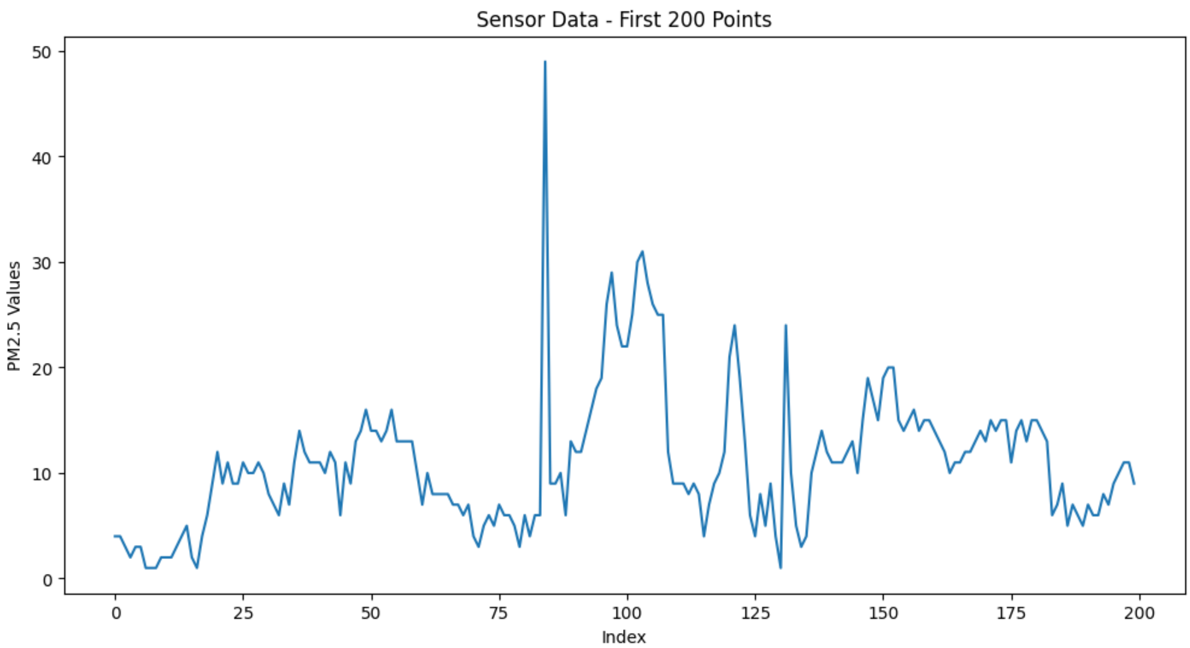
Step 2 : Create lag and residual columns for the dataframe

In [7]:
```
train_data['Lag1'] = train_data['pm25'].shift(1)
train_data['Lag2'] = train_data['pm25'].shift(2)


train_data['Residuals'] = train_data['pm25'].diff()
```

Step 3 Plot Data to analyze TREND (plot first 200 items for clear analysis)

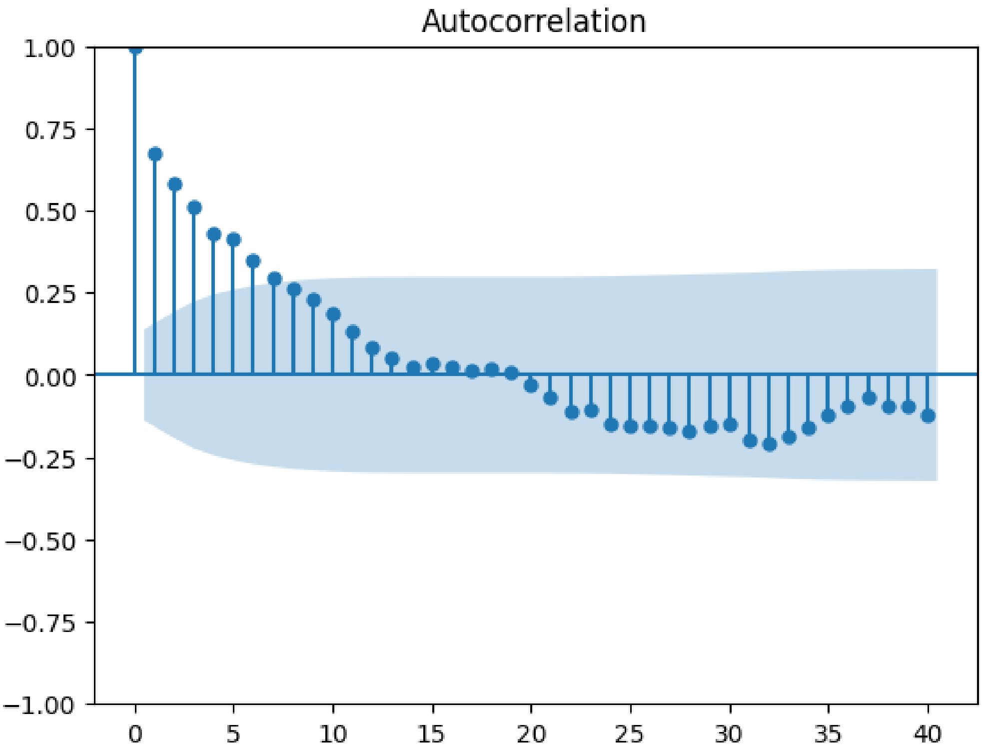
Analyze whether data is stationary or not. If not stationary plot autocorrelation plot for residuals and determine 'd' (differencing order)

In [6]:
```
first_200_points = train_data.head(200)
plt.figure(figsize=(12, 6))
plt.plot(first_200_points.index, first_200_points['pm25'])  # Using index as x-axis
plt.title("Sensor Data - First 200 Points")
plt.xlabel("Index")
plt.ylabel("PM2.5 Values")
plt.show()
```



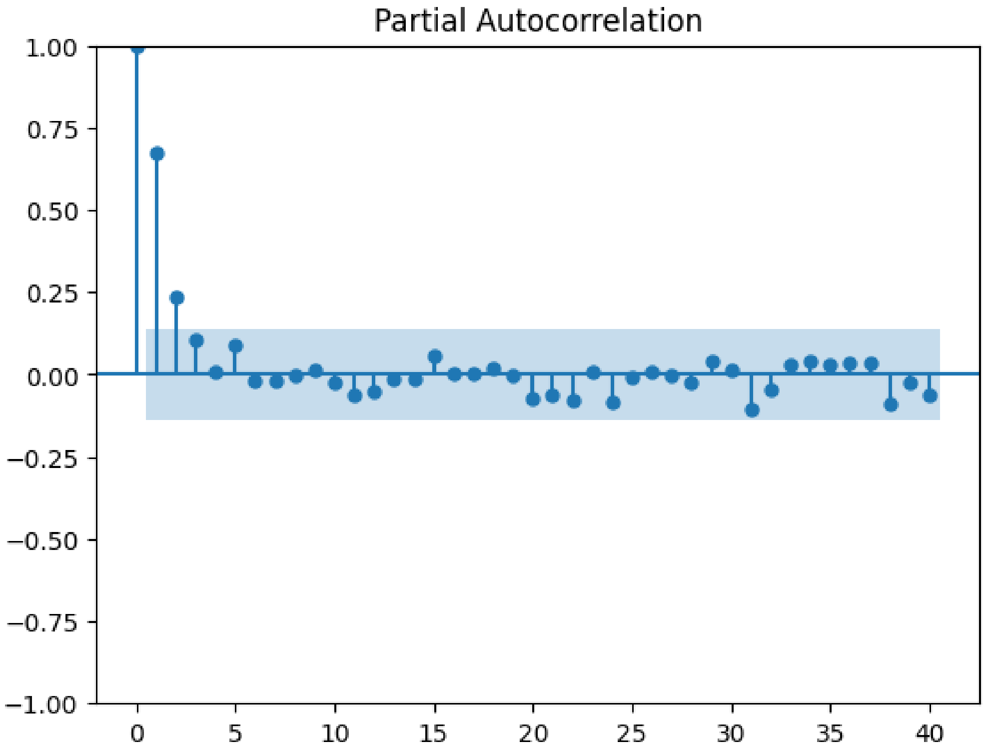
Step 4 : Plot Auto Correlation plot of the data and determine 'q' value(plot first 200 items for clear analysis)

In [10]:
```
import statsmodels.api as sm
from statsmodels.graphics.tsaplots import plot_acf

acf_plot = plot_acf(train_data['pm25'][:200], lags=40)
acf_plot.show()
```



Step 5 : Plot Partial Auto Correlation Plot of the data and determine 'p' value(plot first 200 items for clear analysis)

In [8]:
```
from statsmodels.graphics.tsaplots import plot_pacf


pacf_plot = plot_pacf(train_data['pm25'][:200], lags=40)
pacf_plot.show()
```



Step 6 : Fit ARIMA model to the data

In [12]:
```
import statsmodels.api as sm
from statsmodels.tsa.arima.model import ARIMA

p = 1
d = 1
q = 2

model = ARIMA(train_data['pm25'], order=(p, d, q))
model_fit = model.fit()
model_fit.summary()
```

Out[12]:

SARIMAX Results

Dep. Variable:	pm25	**No. Observations:**	8000
Model:	ARIMA(1, 1, 2)	**Log Likelihood**	-65932.965
Date:	Tue, 31 Oct 2023	**AIC**	131873.930
Time:	07:05:10	**BIC**	131901.878
Sample:	0	**HQIC**	131883.496
	- 8000		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.5409	0.009	58.396	0.000	0.523	0.559
ma.L1	-1.0841	0.010	-108.969	0.000	-1.104	-1.065
ma.L2	0.1414	0.008	18.148	0.000	0.126	0.157
sigma2	8.452e+05	2091.808	404.045	0.000	8.41e+05	8.49e+05

Ljung-Box (L1) (Q):	0.14	**Jarque-Bera (JB):**	2228575.63
Prob(Q):	0.71	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.87	**Skew:**	6.20
Prob(H) (two-sided):	0.00	**Kurtosis:**	83.83

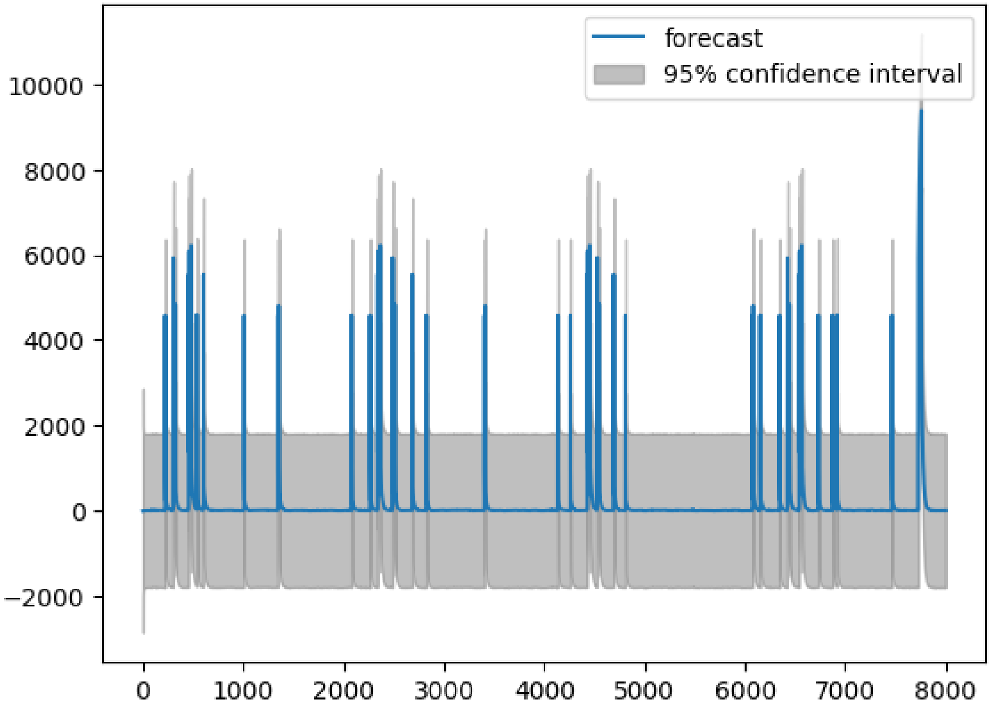
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

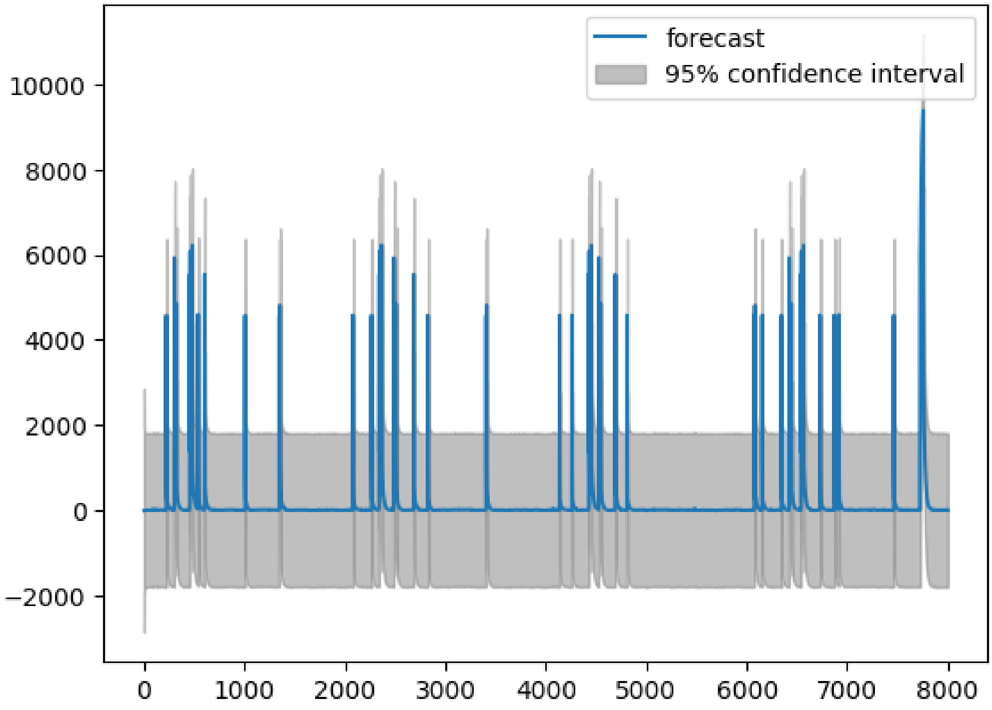
Step 7 : Predict Values using ARIMA model

In [18]:

```
from statsmodels.graphics.tsaplots import plot_predict
start_index = len(train_data)
end_index = start_index + len(test_data) - 1

predictions = model_fit.predict(start=start_index, end=end_index)
plot_predict(model_fit)
```

Out[18]:





Step 8: Evaluating the ARIMA model

In [18]:
```
# Read test CSV File with pm2.5 values
from sklearn.metrics import mean_squared_error
test_values = test_data['pm25']
```

In [50]:
```
mse = mean_squared_error(test_values, predictions)
print(f"Mean Squared Error (MSE): {mse}")
```

```
Mean Squared Error (MSE): 56.903693885477146
```

Step 9 : Try to change p,q,d values and observe model performance (optional)

In [27]:
```
## 1,2,1
p = 1
```

```
d = 2
q = 1

model = ARIMA(train_data['pm25'], order=(p, d, q))
model_fit = model.fit()


start_index = len(train_data)
end_index = start_index + len(test_data) - 1
predictions = model_fit.predict(start=start_index, end=end_index)

mse = mean_squared_error(test_data['pm25'], predictions)

print(f"MSE: {mse}")
```

```
MSE: 81.60256688178424
```

In [28]:
```
#2,2,1
p = 2
d = 2
q = 1

model = ARIMA(train_data['pm25'], order=(p, d, q))
model_fit = model.fit()


start_index = len(train_data)
end_index = start_index + len(test_data) - 1
predictions = model_fit.predict(start=start_index, end=end_index)

mse = mean_squared_error(test_data['pm25'], predictions)

print(f"MSE: {mse}")
```

```
MSE: 72.93351435024334
```

In [34]:
```
#2,2,2
p = 2
d = 2
q = 2

model = ARIMA(train_data['pm25'], order=(p, d, q))
model_fit = model.fit()


start_index = len(train_data)
end_index = start_index + len(test_data) - 1
predictions = model_fit.predict(start=start_index, end=end_index)

mse = mean_squared_error(test_data['pm25'], predictions)

print(f"MSE: {mse}")
```

```
MSE: 57.58995872704161
```

In [35]:
```
#7,2,3
p = 7
d = 2
q = 3

model = ARIMA(train_data['pm25'], order=(p, d, q))
model_fit = model.fit()


start_index = len(train_data)
end_index = start_index + len(test_data) - 1
predictions = model_fit.predict(start=start_index, end=end_index)
```

```python
mse = mean_squared_error(test_data['pm25'], predictions)
print(f"MSE: {mse}")
```

```
/home/student/s1292011/.local/lib/python3.10/site-packages/statsmodels/tsa/statespac
e/sarimax.py:978: UserWarning: Non-invertible starting MA parameters found. Using zer
os as starting parameters.
  warn('Non-invertible starting MA parameters found.'
/home/student/s1292011/.local/lib/python3.10/site-packages/statsmodels/base/model.py:
607: ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check ml
e_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
MSE: 266.3599471565791
```

In [ ]: