IMPORT LIBERIES

In [8]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.statespace.sarimax import SARIMAX
from prophet import Prophet
from sklearn.metrics import mean_absolute_error, mean_squared_error
import warnings
warnings.filterwarnings("ignore")
```

DATASET LOADING

In [3]:
```
df=pd.read_csv('/content/time series data.csv')
df.head()
```

Out[3]:

	t	ProductP1	ProductP2	ProductP3	ProductP4	ProductP5	price	temperature
0	1	197	66	266	113	2	1	18
1	2	153	44	264	74	1	2	21
2	3	128	55	317	116	0	1	19
3	4	133	57	390	70	0	2	17
4	5	120	47	440	141	1	1	18

In [4]:
```
df.tail()
```

Out[4]:

	t	ProductP1	ProductP2	ProductP3	ProductP4	ProductP5	price	temperature
95	96	174	226	975	94	0	1	2
96	97	158	201	1141	91	0	2	3
97	98	145	217	881	46	4	1	3
98	99	161	249	741	70	0	1	7
99	100	160	331	854	75	0	1	5

In [5]:
```
df.describe()
```

Out[5]:

	t	ProductP1	ProductP2	ProductP3	ProductP4	ProductP5	price
count	100.000000	100.000000	100.000000	100.0000	100.000000	100.000000	100.000000
mean	50.500000	148.370000	153.440000	631.5800	80.120000	0.550000	1.470000
std	29.011492	18.061989	68.164288	202.8871	36.780891	1.225775	0.501614
min	1.000000	107.000000	42.000000	203.0000	11.000000	0.000000	1.000000
25%	25.750000	134.000000	95.500000	491.7500	55.750000	0.000000	1.000000
50%	50.500000	151.000000	155.500000	611.0000	70.000000	0.000000	1.000000
75%	75.250000	160.000000	200.250000	770.5000	100.250000	0.000000	2.000000
max	100.000000	197.000000	331.000000	1141.0000	194.000000	5.000000	2.000000

In [6]:
```
df.shape
```

Out[6]: (100, 8)

In [7]:
```
df.info()
```

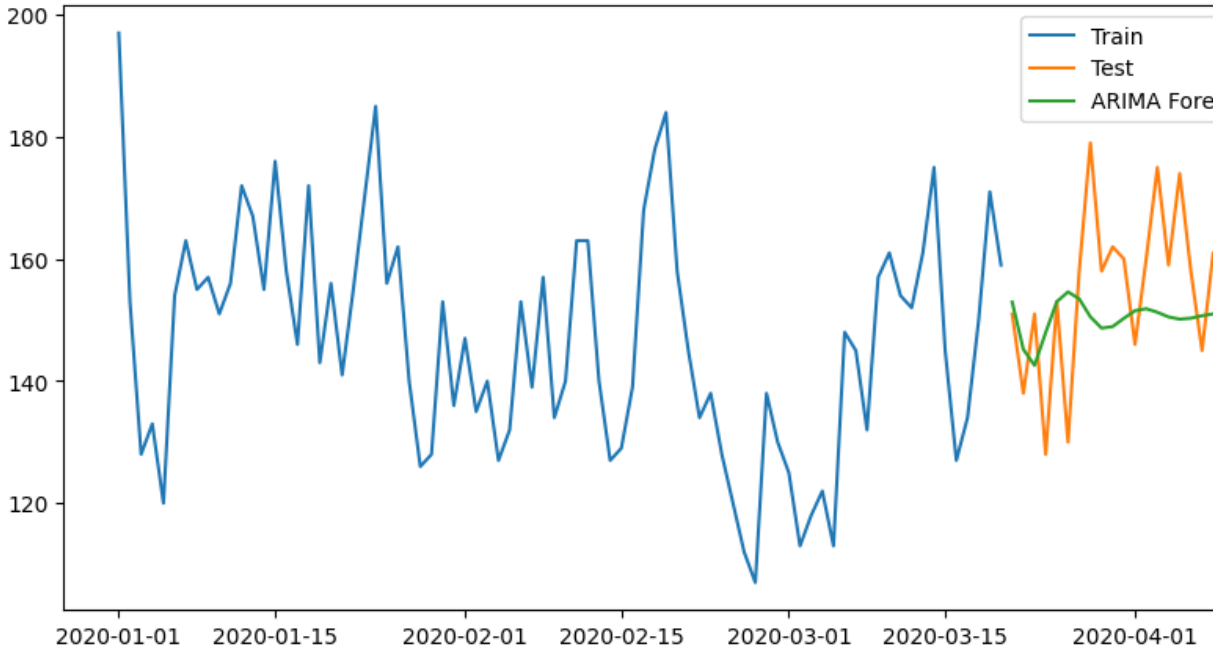
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   t            100 non-null    int64
 1   ProductP1    100 non-null    int64
 2   ProductP2    100 non-null    int64
 3   ProductP3    100 non-null    int64
 4   ProductP4    100 non-null    int64
 5   ProductP5    100 non-null    int64
 6   price        100 non-null    int64
 7   temperature  100 non-null    int64
dtypes: int64(8)
memory usage: 6.4 KB
```

MODEL DEVELOPMENT

In [9]:
```
#PREPARE YOUR DATA
df['Date'] = pd.date_range(start="2020-01-01", periods=len(df), freq='D')
df.set_index('Date', inplace=True)

# Select target variable
ts = df['ProductP1']

# Train-test split
train_size = int(len(ts) * 0.8)
train, test = ts[:train_size], ts[train_size:]
```

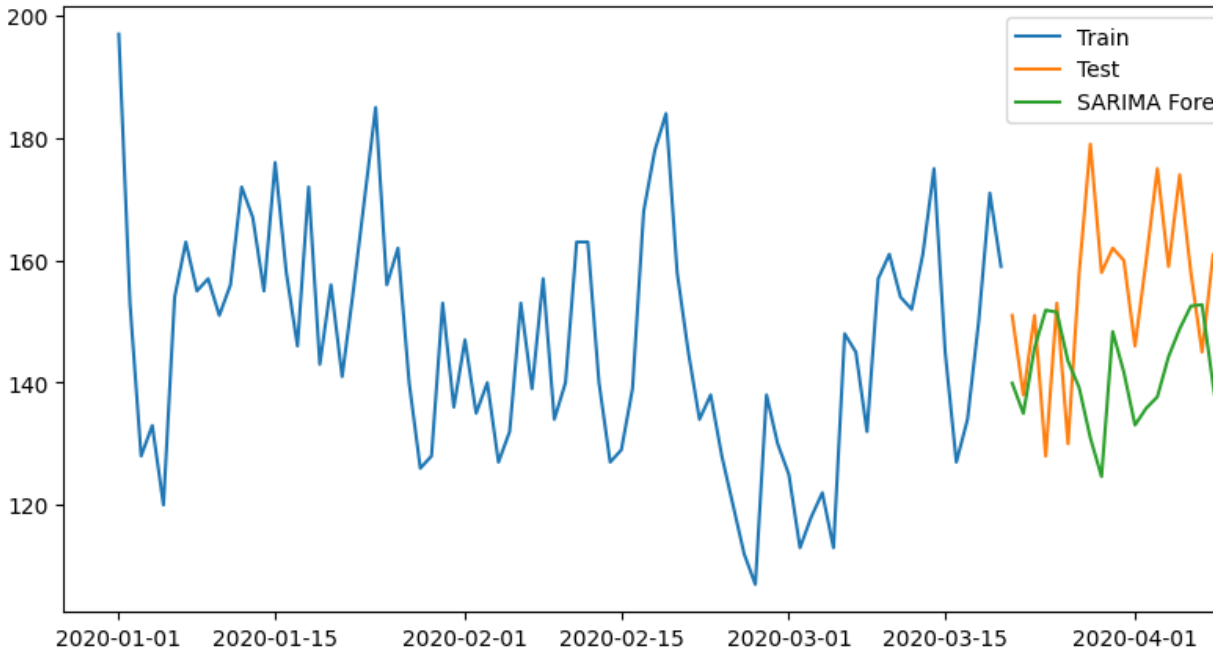
In [10]:
```
#ARIMA MODEL
arima_model = ARIMA(train, order=(5,1,0))
arima_fit = arima_model.fit()
arima_forecast = arima_fit.forecast(steps=len(test))

print("ARIMA MAE:", mean_absolute_error(test, arima_forecast))
print("ARIMA RMSE:", np.sqrt(mean_squared_error(test, arima_forecast)))

plt.figure(figsize=(10,5))
plt.plot(train, label="Train")
plt.plot(test, label="Test")
plt.plot(test.index, arima_forecast, label="ARIMA Forecast")
plt.legend(); plt.show()
```

```
ARIMA MAE: 11.4710168906343
ARIMA RMSE: 13.94277091688454
```



In [11]:
```
#SARIMA MODEL
sarima_model = SARIMAX(train, order=(1,1,1), seasonal_order=(1,1,1,12))
sarima_fit = sarima_model.fit(disp=False)
sarima_forecast = sarima_fit.forecast(steps=len(test))

print("SARIMA MAE:", mean_absolute_error(test, sarima_forecast))
print("SARIMA RMSE:", np.sqrt(mean_squared_error(test, sarima_forecast)))

plt.figure(figsize=(10,5))
plt.plot(train, label="Train")
plt.plot(test, label="Test")
plt.plot(test.index, sarima_forecast, label="SARIMA Forecast")
plt.legend(); plt.show()
```

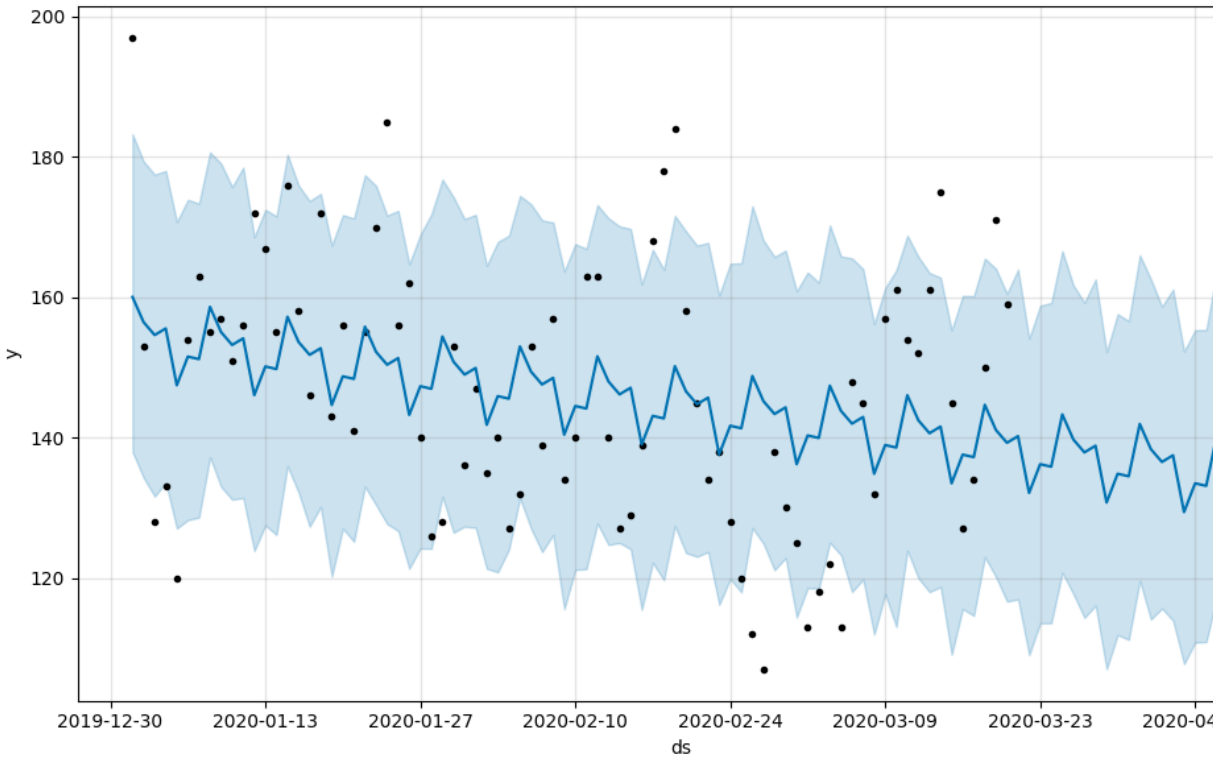
```
SARIMA MAE: 18.570109039841686
SARIMA RMSE: 22.108953029916634
```



In [12]:
```
#PROPHET MODEL
prophet_df = df[['ProductP1']].reset_index()
```

```
prophet_df.columns = ['ds', 'y']

train_prophet = prophet_df.iloc[:train_size]
test_prophet = prophet_df.iloc[train_size:]

prophet_model = Prophet()
prophet_model.fit(train_prophet)

future = prophet_model.make_future_dataframe(periods=len(test_prophet))
forecast = prophet_model.predict(future)

prophet_forecast = forecast['yhat'].iloc[-len(test_prophet):]

print("Prophet MAE:", mean_absolute_error(test_prophet['y'], prophet_forecast))
print("Prophet RMSE:", np.sqrt(mean_squared_error(test_prophet['y'], prophet_for

prophet_model.plot(forecast)
plt.show()
```

```
INFO:prophet:Disabling yearly seasonality. Run prophet with yearly_seasonality=True t
override this.
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to
override this.
DEBUG:cmdstanpy:input tempfile: /tmp/tmpapr100ap/lgf_k_nh.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpapr100ap/odv2j9a3.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.12/dist-packages/prophet/stan_
prophet_model.bin', 'random', 'seed=52139', 'data', 'file=/tmp/tmpapr100ap/lgf_k_nh.j
'init=/tmp/tmpapr100ap/odv2j9a3.json', 'output', 'file=/tmp/tmpapr100ap/
prophet_modelou_t03ko/prophet_model-20250920104548.csv', 'method=optimize',
'algorithm=newton', 'iter=10000']
10:45:48 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
10:45:48 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
Prophet MAE: 20.44555334902817
Prophet RMSE: 23.311301470941018
```



EVALUATION

In [15]:
```
y = df["ProductP1"]

# Train-test split (last 20 points for testing)
train, test = y[:-20], y[-20:]

# Build a simple ARIMA model
model = ARIMA(train, order=(1,1,1))
model_fit = model.fit()

# Forecast
y_pred = model_fit.forecast(steps=len(test))

# Evaluation
mae = mean_absolute_error(test, y_pred)
rmse = np.sqrt(mean_squared_error(test, y_pred))

print("Mean Absolute Error (MAE):", mae)
print("Root Mean Squared Error (RMSE):", rmse)
```

```
Mean Absolute Error (MAE): 13.609480064463478
Root Mean Squared Error (RMSE): 16.171507448504254
```

VISUALIZATION

In [16]:
```
plt.figure(figsize=(10,5))
plt.plot(test.index, test, label="Actual", marker='o')
plt.plot(test.index, y_pred, label="Forecast", marker='x')

plt.title("Actual vs Forecast - ProductP1")
plt.xlabel("Time")
plt.ylabel("ProductP1 Sales")
plt.legend()
```

```
plt.grid(True)
plt.show()
```