```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import pandas as pd
from statsmodels.tsa.statespace.sarimax import SARIMAX
from matplotlib import pyplot as plt
from sklearn.metrics import mean_squared_error
import numpy as np


train_path = '/content/drive/My Drive/forecasting-unit-sales-vit-task-2/train.csv'
test_path = '/content/drive/My Drive/forecasting-unit-sales-vit-task-2/test.csv'
sample_path = '/content/drive/My Drive/forecasting-unit-sales-vit-task-2/sample_submission.csv'


train_data = pd.read_csv(train_path)
test_data = pd.read_csv(test_path)
sample_data = pd.read_csv(sample_path)


train_data.head(), test_data.head(), sample_data.head()
```

```
(                       ID        date     Item Id  \
 0  2022-04-12_B09KDTS4DC  2022-04-12  B09KDTS4DC
 1  2022-04-12_B09MR2MLZH  2022-04-12  B09MR2MLZH
 2  2022-04-12_B09KSYL73R  2022-04-12  B09KSYL73R
 3  2022-04-12_B09KT5HMNY  2022-04-12  B09KT5HMNY
 4  2022-04-12_B09KTF8ZDQ  2022-04-12  B09KTF8ZDQ

                                            Item Name  ad_spend anarix_id  \
 0  NapQueen Elizabeth 8" Gel Memory Foam Mattress...       NaN  NAPQUEEN
 1  NapQueen 12 Inch Bamboo Charcoal Queen Size Me...       NaN  NAPQUEEN
 2     NapQueen Elsa 8" Innerspring Mattress, Twin XL       NaN  NAPQUEEN
 3        NapQueen Elsa 6" Innerspring Mattress, Twin       NaN  NAPQUEEN
 4     NapQueen Elsa 6" Innerspring Mattress, Twin XL       NaN  NAPQUEEN

    units  unit_price
 0    0.0         0.0
 1    0.0         0.0
 2    0.0         0.0
 3    0.0         0.0
 4    0.0         0.0  ,
                       ID        date     Item Id  \
 0  2024-07-01_B09KDR64LT  2024-07-01  B09KDR64LT
 1  2024-07-01_B09KDTS4DC  2024-07-01  B09KDTS4DC
 2  2024-07-01_B09KDTHJ6V  2024-07-01  B09KDTHJ6V
 3  2024-07-01_B09KDQ2BWY  2024-07-01  B09KDQ2BWY
 4  2024-07-01_B09KDYY3SB  2024-07-01  B09KDYY3SB

                                            Item Name  ad_spend anarix_id  \
 0  NapQueen Elizabeth 10" Gel Memory Foam Mattres...       NaN  NAPQUEEN
 1  NapQueen Elizabeth 8" Gel Memory Foam Mattress...       NaN  NAPQUEEN
 2  NapQueen Elizabeth 12" Gel Memory Foam Mattres...       NaN  NAPQUEEN
 3  NapQueen Elizabeth 12" Gel Memory Foam Mattres...       NaN  NAPQUEEN
 4  NapQueen Elizabeth 10" Gel Memory Foam Mattres...    101.72  NAPQUEEN

    unit_price
 0         0.0
 1         0.0
 2         0.0
 3         0.0
 4      1094.5  ,
                       ID  units    TARGET
 0  2024-07-01_B09KDTS4DC    NaN -2.923211
 1  2024-07-02_B09KDTS4DC    NaN -1.992157
 2  2024-07-03_B09KDTS4DC    NaN -6.185509
 3  2024-07-04_B09KDTS4DC    NaN -5.878580
 4  2024-07-05_B09KDTS4DC    NaN -5.962086)
```

```
# Function to train SARIMA model on aggregated data and make predictions
def train_sarima_and_predict_aggregated(train_data, test_data):
    # Aggregate the training data by date
    aggregated_train_data = train_data.groupby('date').agg({'units': 'sum'}).reset_index()

    # Set date as index
    aggregated_train_data.set_index('date', inplace=True)
    # Train SARIMA model on aggregated data
    sarima_model = SARIMAX(aggregated_train_data['units'], order=(1, 1, 1), seasonal_order=(1, 1, 1, 12))
    sarima_results = sarima_model.fit(disp=False)

    # Forecast for each unique date in test set
    test_dates = test_data['date'].unique()
    forecast = sarima_results.get_forecast(steps=len(test_dates))
    predicted_units = forecast.predicted_mean

    # Map predictions back to test_data
    date_predictions = pd.DataFrame({'date': test_dates, 'predicted_units': predicted_units})

    return date_predictions

# Load the datasets
train_data = pd.read_csv('/content/drive/My Drive/forecasting-unit-sales-vit-task-2/train.csv')
test_data = pd.read_csv('/content/drive/My Drive/forecasting-unit-sales-vit-task-2/test.csv')
submission_data = pd.read_csv('/content/drive/My Drive/forecasting-unit-sales-vit-task-2/sample_submission.csv')

# Convert date columns to datetime format
train_data['date'] = pd.to_datetime(train_data['date'])
test_data['date'] = pd.to_datetime(test_data['date'])

# Ensure the data is sorted by date
train_data = train_data.sort_values('date')
test_data = test_data.sort_values('date')

# Select relevant columns
train_data = train_data[['date', 'Item Id', 'units', 'unit_price']]
test_data = test_data[['date', 'Item Id', 'unit_price']]

# Handle missing values
train_data.dropna(subset=['Item Id', 'units'], inplace=True)

# Get predictions for aggregated data
aggregated_predictions = train_sarima_and_predict_aggregated(train_data, test_data)

# Merge predictions with test_data on date using suffixes
test_data = test_data.merge(aggregated_predictions, on='date', how='left', suffixes=('_test', '_pred'))

# Ensure the column name 'predicted_units' is correct
if 'predicted_units_pred' in test_data.columns:
    test_data.rename(columns={'predicted_units_pred': 'predicted_units'}, inplace=True)

# Create the ID column
test_data['ID'] = test_data['date'].dt.strftime('%Y-%m-%d') + '_' + test_data['Item Id']

# Filter the necessary columns and remove NaN values
sample_data = test_data[['ID', 'predicted_units']].rename(columns={'predicted_units': 'TARGET'}).dropna()

# Save the submission file
sample_data.to_csv('/content/drive/My Drive/forecasting-unit-sales-vit-task-2/sampleSubmission.csv', index=False)

print(sample_data.head())
```

```
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency information was pr
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency information was pr
  self._init_dates(dates, freq)
                      ID      TARGET
0  2024-07-01_B09KDR64LT  1006.09442
1  2024-07-01_B0BNL11QD8  1006.09442
2  2024-07-01_B0BDRS6R5Z  1006.09442
3  2024-07-01_B0BNL4L4K5  1006.09442
4  2024-07-01_B0BNL3J36Z  1006.09442
```