In [15]:

```
import pandas as pd
import numpy as np
from statsmodels.tsa.statespace.sarimax import SARIMAX

# Load the CSV data
train_file_path = r'C:\Users\Giridhar\Downloads\forecasting-unit-sales-vit-task-2\train.c
test_file_path = r'C:\Users\Giridhar\Downloads\forecasting-unit-sales-vit-task-2\test.csv

train_data = pd.read_csv(train_file_path)
test_data = pd.read_csv(test_file_path)

# Handle missing values
def handle_missing_values(data, is_train=True):
    data['Item Id'].fillna('Unknown', inplace=True)
    data['Item Name'].fillna('Unknown Item', inplace=True)
    data['ad_spend'].replace([np.inf, -np.inf], np.nan, inplace=True)
    data['ad_spend'].fillna(0, inplace=True)
    if is_train and 'units' in data.columns:
        data['units'].fillna(0, inplace=True)
    return data

train_data = handle_missing_values(train_data, is_train=True)
test_data = handle_missing_values(test_data, is_train=False)

train_data['units'] = train_data['units'].apply(lambda x: max(x, 0))
train_data['unit_price'] = train_data['unit_price'].apply(lambda x: max(x, 0))

# Create time-based features
def create_time_features(data):
    data['date'] = pd.to_datetime(data['date'])
    data['day_of_week'] = data['date'].dt.dayofweek
    data['month'] = data['date'].dt.month
    data['quarter'] = data['date'].dt.quarter
    return data

train_data = create_time_features(train_data)
test_data = create_time_features(test_data)

# Aggregate data by Item Id and date
def aggregate_data(data, is_train=True):
    agg_dict = {
        'ad_spend': 'sum',
        'unit_price': 'mean'
    }
    if is_train:
        agg_dict['units'] = 'sum'
    return data.groupby(['Item Id', 'date']).agg(agg_dict).reset_index()

train_aggregated = aggregate_data(train_data, is_train=True)
test_aggregated = aggregate_data(test_data, is_train=False)

# Create a dictionary to hold DataFrames for each item
predictions = {}

# List of unique item IDs
item_ids = train_aggregated['Item Id'].unique()

for item_id in item_ids:
    train_item = train_aggregated[train_aggregated['Item Id'] == item_id].copy()
    test_item = test_aggregated[test_aggregated['Item Id'] == item_id].copy()

    if not train_item.empty and not test_item.empty:
        # Prepare data for SARIMA
        train_item_sarima = train_item[['date', 'units']].set_index('date')
        train_item_sarima = train_item_sarima.asfreq('D')

        # Check if there are sufficient data points for model fitting
        if len(train_item_sarima) > 10:  # Adjust threshold as necessary
```

```
            try:
                # Initialize and fit the SARIMA model with simpler parameters
                model_sarima = SARIMAX(train_item_sarima,
                                       order=(1, 1, 1),
                                       seasonal_order=(1, 1, 0, 7),
                                       enforce_stationarity=False,
                                       enforce_invertibility=False)

                model_sarima_fit = model_sarima.fit(disp=False, maxiter=2000)
                print(f"Model fitted successfully for item {item_id}")

                # Create future dataframe and predict
                future_dates = pd.date_range(start=train_item_sarima.index[-1] + pd.Timed
                future = pd.DataFrame(index=future_dates)
                future = future.asfreq('D')

                forecast_sarima = model_sarima_fit.get_forecast(steps=len(future_dates))
                forecast_sarima_mean = forecast_sarima.predicted_mean

                # Debugging print
                print(f"Forecast for item {item_id}: {forecast_sarima_mean.head()}")

                # Add predictions to test_item DataFrame
                test_item['predicted_units_sarima'] = np.ceil(forecast_sarima_mean).astyp

                # Remove negative values (if any) by setting them to zero
                test_item['predicted_units_sarima'] = test_item['predicted_units_sarima']

                # Store the DataFrame in the dictionary
                predictions[item_id] = test_item[['date', 'Item Id', 'predicted_units_sar

            except Exception as e:
                print(f"Failed to fit SARIMA model for item {item_id}: {e}")
                test_item['predicted_units_sarima'] = np.nan
        else:
            print(f"Not enough data to fit model for item {item_id}")

# Combine all predictions into a single DataFrame
if predictions:
    combined_predictions = pd.concat(predictions.values(), ignore_index=True)
    combined_predictions.to_csv('task1_predictions_sarima.csv', index=False)
    print('task1_predictions_sarima.csv')
else:
    print('No predictions were generated.')
```

```
2024-06-03    2.848305
2024-06-04   -0.096140
2024-06-05    0.867445
Freq: D, dtype: float64
Model fitted successfully for item B0CY5LR4VX
Forecast for item B0CY5LR4VX: 2024-06-01    1.014500
2024-06-02    1.983313
2024-06-03    1.116869
2024-06-04    0.918841
2024-06-05    1.184878
Freq: D, dtype: float64
Model fitted successfully for item B0CY5QQ49F
Forecast for item B0CY5QQ49F: 2024-06-01    3.614345e-16
2024-06-02    2.921553e-16
2024-06-03    2.154669e-16
2024-06-04    3.063756e-17
2024-06-05    1.522304e-17
Freq: D, dtype: float64
task1_predictions_sarima.csv
```

In [17]:
```
combined_predictions
```

Out[17]:

	date	Item Id	predicted_units_sarima
0	2024-07-01	B09KDLQ2GW	NaN
1	2024-07-02	B09KDLQ2GW	NaN
2	2024-07-03	B09KDLQ2GW	NaN
3	2024-07-04	B09KDLQ2GW	NaN
4	2024-07-05	B09KDLQ2GW	NaN
...	...	...	...
2828	2024-07-24	B0CY5QQ49F	NaN
2829	2024-07-25	B0CY5QQ49F	NaN
2830	2024-07-26	B0CY5QQ49F	NaN
2831	2024-07-27	B0CY5QQ49F	NaN
2832	2024-07-28	B0CY5QQ49F	NaN

2833 rows × 3 columns

In [ ]:
```
```