Step 1: Exploratory Data Analysis (EDA)

```
import pandas as pd

train_df = pd.read_csv('train.csv')
test_df = pd.read_csv('test.csv')
sample_submission = pd.read_csv('sample_submission.csv')

print("Training Data:")
print(train_df.head())

print("\nTraining Data Statistics:")
print(train_df.describe())

print("\nMissing Values in Training Data:")
print(train_df.isnull().sum())
```

```
Training Data:
                        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

Training Data Statistics:
           ad_spend         units     unit_price
count  77303.000000  83592.000000  101490.000000
mean     110.771470     10.284381     106.750922
std      529.303777     68.945915     425.704733
min        0.000000   -173.000000   -8232.000000
25%        0.000000      0.000000       0.000000
50%        4.230000      1.000000       0.000000
75%       44.310000      5.000000       0.000000
max    47934.990000   9004.000000   21557.390000

Missing Values in Training Data:
ID               0
date             0
Item Id          2
Item Name     1832
ad_spend     24187
anarix_id        0
units        17898
unit_price       0
dtype: int64
```

Step 2: Data Cleaning

```
train_df.dropna(subset=['Item Id'], inplace=True)

train_df['Item Name'].fillna('Unknown', inplace=True)

train_df['ad_spend'].fillna(0, inplace=True)

train_df.dropna(subset=['units'], inplace=True)

train_df = train_df[(train_df['units'] >= 0) & (train_df['ad_spend'] >= 0) & (train_df['unit_price'] >= 0)]

print("\nCleaned Data:")
print(train_df.describe())
print("\nMissing Values in Cleaned Data:")
print(train_df.isnull().sum())
```

```
Cleaned Data:
           ad_spend         units    unit_price
```

```
count  81942.000000  81942.000000  81942.000000
mean     102.770553     10.541298    132.838373
std      514.021257     69.603631    468.829253
min        0.000000      0.000000      0.000000
25%        0.000000      0.000000      0.000000
50%        1.730000      1.000000      0.000000
75%       35.787500      5.000000     20.074554
max    47934.990000   9004.000000  21557.390000

Missing Values in Cleaned Data:
ID            0
date          0
Item Id       0
Item Name     0
ad_spend      0
anarix_id     0
units         0
unit_price    0
dtype: int64
```

Step 3: Feature Engineering

```
train_df['date'] = pd.to_datetime(train_df['date'])

train_df['day_of_week'] = train_df['date'].dt.dayofweek
train_df['month'] = train_df['date'].dt.month

print("Data with Date-Related Features:")
print(train_df.head())
```

```
Data with Date-Related Features:
                         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...       0.0  NAPQUEEN
1  NapQueen 12 Inch Bamboo Charcoal Queen Size Me...       0.0  NAPQUEEN
2     NapQueen Elsa 8" Innerspring Mattress, Twin XL       0.0  NAPQUEEN
3        NapQueen Elsa 6" Innerspring Mattress, Twin       0.0  NAPQUEEN
4     NapQueen Elsa 6" Innerspring Mattress, Twin XL       0.0  NAPQUEEN

   units  unit_price  day_of_week  month
0    0.0         0.0            1      4
1    0.0         0.0            1      4
2    0.0         0.0            1      4
3    0.0         0.0            1      4
4    0.0         0.0            1      4
```

```
def create_lag_features(df, lags, window):
    for lag in lags:
        df[f'lag_{lag}'] = df.groupby('Item Id')['units'].shift(lag)

    df[f'rolling_mean_{window}'] = df.groupby('Item Id')['units'].shift(1).rolling(window=window).mean()
    return df

lags = [1, 7, 14]
window = 7

train_df = create_lag_features(train_df, lags, window)
train_df.dropna(inplace=True)

print("Data with Lag Features and Rolling Averages:")
print(train_df.head())
```

```
Data with Lag Features and Rolling Averages:
                         ID       date     Item Id  \
239  2022-04-26_B09KDTS4DC 2022-04-26  B09KDTS4DC
240  2022-04-26_B09KSXP3HN 2022-04-26  B09KSXP3HN
241  2022-04-26_B09KSYL73R 2022-04-26  B09KSYL73R
242  2022-04-26_B09KTF8ZDQ 2022-04-26  B09KTF8ZDQ
244  2022-04-26_B09KTMKDKJ 2022-04-26  B09KTMKDKJ

                                          Item Name  ad_spend anarix_id  \
239  NapQueen Elizabeth 8" Gel Memory Foam Mattress...       0.0  NAPQUEEN
240       NapQueen Elsa 8" Innerspring Mattress, Queen       0.0  NAPQUEEN
241     NapQueen Elsa 8" Innerspring Mattress, Twin XL       0.0  NAPQUEEN
242     NapQueen Elsa 6" Innerspring Mattress, Twin XL       0.0  NAPQUEEN
244        NapQueen Elsa 8" Innerspring Mattress, Twin       0.0  NAPQUEEN

     units  unit_price  day_of_week  month  lag_1  lag_7  lag_14  \
239    0.0         0.0            1      4    0.0    0.0     0.0
```

```
240    0.0         0.0            1      4    1.0    0.0     0.0
241    0.0         0.0            1      4    0.0    0.0     0.0
242    0.0         0.0            1      4    0.0    0.0     0.0
244    0.0         0.0            1      4    0.0    0.0     0.0

     rolling_mean_7
239        0.000000
240        0.142857
241        0.142857
242        0.142857
244        0.142857
```

Step 4: Model Selection

```
!pip install prophet
```

```
Requirement already satisfied: prophet in /usr/local/lib/python3.10/dist-packages (1.1.5)
Requirement already satisfied: cmdstanpy>=1.0.4 in /usr/local/lib/python3.10/dist-packages (from prophet) (1.2.4)
Requirement already satisfied: numpy>=1.15.4 in /usr/local/lib/python3.10/dist-packages (from prophet) (1.26.4)
Requirement already satisfied: matplotlib>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from prophet) (3.7.1)
Requirement already satisfied: pandas>=1.0.4 in /usr/local/lib/python3.10/dist-packages (from prophet) (2.1.4)
Requirement already satisfied: holidays>=0.25 in /usr/local/lib/python3.10/dist-packages (from prophet) (0.53)
Requirement already satisfied: tqdm>=4.36.1 in /usr/local/lib/python3.10/dist-packages (from prophet) (4.66.4)
Requirement already satisfied: importlib-resources in /usr/local/lib/python3.10/dist-packages (from prophet) (6.4.0)
Requirement already satisfied: stanio<2.0.0,>=0.4.0 in /usr/local/lib/python3.10/dist-packages (from cmdstanpy>=1.0.4->prophet) (0.
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/dist-packages (from holidays>=0.25->prophet) (2.8.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.0.0->prophet) (1.2.1
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.0.0->prophet) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.0.0->prophet) (4.53
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.0.0->prophet) (1.4.
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.0.0->prophet) (24.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.0.0->prophet) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.0.0->prophet) (3.1.2
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.4->prophet) (2024.1)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.4->prophet) (2024.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil->holidays>=0.25->prophet)
```

```
# Model Training

from prophet import Prophet

train_df['date'] = pd.to_datetime(train_df['date'])
prophet_df = train_df[['date', 'units']]
prophet_df.columns = ['ds', 'y']

model = Prophet()
model.fit(prophet_df)

future = model.make_future_dataframe(periods=30)
forecast = model.predict(future)

print(forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail())
```

```
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
DEBUG:cmdstanpy:input tempfile: /tmp/tmptkygl28k/_92dp1t9.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmptkygl28k/4uowc8b_.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_model/prophet_model.bin', 'random', 'seed=1050
09:48:15 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
09:48:25 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
            ds      yhat  yhat_lower  yhat_upper
775 2024-06-26  6.468196  -88.710227   96.862499
776 2024-06-27  5.757797  -86.701120   96.974242
777 2024-06-28  7.777510  -79.800267  108.400903
778 2024-06-29  7.441948  -84.479687   97.335523
779 2024-06-30  8.752223  -85.394688  100.999889
```

Step 5: Modeling - Generating Predictions for Each Item ID:

```
def forecast_per_item(train_df, periods=30):
    all_forecasts = []

    unique_items = train_df['Item Id'].unique()
    for item in unique_items:
        item_df = train_df[train_df['Item Id'] == item]
        item_df = item_df[['date', 'units']]
        item_df.columns = ['ds', 'y']

        if item_df.dropna().shape[0] < 2:
            continue

        model = Prophet()
        model.fit(item_df)

        future = model.make_future_dataframe(periods=periods)
        forecast = model.predict(future)

        forecast['Item Id'] = item
        forecast = forecast[['ds', 'Item Id', 'yhat']]
        forecast.columns = ['date', 'Item Id', 'units']
        all_forecasts.append(forecast)

    return pd.concat(all_forecasts)

forecasts = forecast_per_item(train_df)

print(forecasts)
```

```
10:48:58 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
INFO:prophet:Disabling yearly seasonality. Run prophet with yearly_seasonality=True to override this.
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
INFO:prophet:n_changepoints greater than number of observations. Using 11.
DEBUG:cmdstanpy:input tempfile: /tmp/tmptkygl28k/vvp8a1e1.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmptkygl28k/sjpawl2p.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_model/prophet_model.bin', 'random', 'seed=9
10:48:58 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
10:48:58 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
INFO:prophet:Disabling yearly seasonality. Run prophet with yearly_seasonality=True to override this.
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
INFO:prophet:n_changepoints greater than number of observations. Using 18.
DEBUG:cmdstanpy:input tempfile: /tmp/tmptkygl28k/vdabfm3o.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmptkygl28k/es35tg6v.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_model/prophet_model.bin', 'random', 'seed=8
10:48:58 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
10:48:58 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
INFO:prophet:Disabling yearly seasonality. Run prophet with yearly_seasonality=True to override this.
INFO:prophet:Disabling weekly seasonality. Run prophet with weekly_seasonality=True to override this.
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
INFO:prophet:n_changepoints greater than number of observations. Using 3.
INFO:prophet:Disabling yearly seasonality. Run prophet with yearly_seasonality=True to override this.
INFO:prophet:Disabling weekly seasonality. Run prophet with weekly_seasonality=True to override this.
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
INFO:prophet:n_changepoints greater than number of observations. Using 1.
DEBUG:cmdstanpy:input tempfile: /tmp/tmptkygl28k/g79yl3xo.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmptkygl28k/gh_cp13m.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_model/prophet_model.bin', 'random', 'seed=9
10:48:59 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
10:48:59 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
         date     Item Id      units
0  2022-05-13  B09KTF8ZDQ  23.785650
1  2022-05-14  B09KTF8ZDQ  18.232936
2  2022-05-15  B09KTF8ZDQ  16.486049
3  2022-05-16  B09KTF8ZDQ  18.415384
4  2022-05-17  B09KTF8ZDQ  16.881209
..        ...         ...        ...
28 2024-06-26  B0CY5KFQBD   8.526587
29 2024-06-27  B0CY5KFQBD   8.810689
30 2024-06-28  B0CY5KFQBD   9.094791
31 2024-06-29  B0CY5KFQBD   9.378892
32 2024-06-30  B0CY5KFQBD   9.662994

[81412 rows x 3 columns]
```

Step 6: Hyperparameter Tuning

```
model = Prophet(
    yearly_seasonality=True,
    weekly_seasonality=True,
    daily_seasonality=False,
    seasonality_mode='multiplicative',
    changepoint_prior_scale=0.5
)
model.fit(prophet_df)

future = model.make_future_dataframe(periods=30)
forecast = model.predict(future)

print(forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail())
```

```
DEBUG:cmdstanpy:input tempfile: /tmp/tmptkygl28k/7lvo0d40.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmptkygl28k/ga1h8o_y.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_model/prophet_model.bin', 'random', 'seed=8464
09:50:30 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
09:51:01 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
            ds      yhat  yhat_lower  yhat_upper
775 2024-06-26  7.110620  -80.538341  107.359941
776 2024-06-27  6.260244  -95.752120  103.880291
777 2024-06-28  7.482723  -88.420342  103.945535
778 2024-06-29  6.734497  -83.097399  101.583516
779 2024-06-30  7.194468  -91.949195  106.087959
```

Step 7: Evaluation

```
from sklearn.metrics import mean_squared_error

forecast_df = forecast[['ds', 'yhat']].set_index('ds')
merged_df = prophet_df.set_index('ds').join(forecast_df, rsuffix='_predicted')

mse = mean_squared_error(merged_df['y'], merged_df['yhat'])
print(f"Mean Squared Error: {mse}")
```

```
Mean Squared Error: 5209.340788253142
```

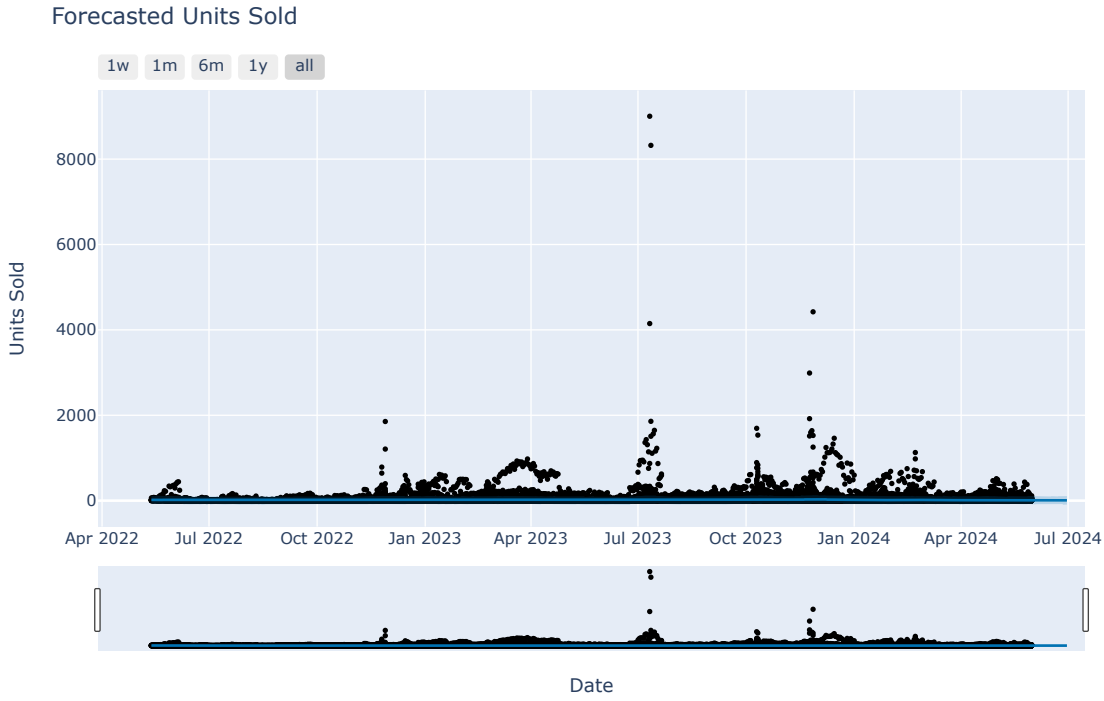
Step 8: Visualize Forecast Trends

```
from prophet.plot import plot_plotly, plot_components_plotly

fig = plot_plotly(model, forecast)
fig.update_layout(title="Forecasted Units Sold", xaxis_title="Date", yaxis_title="Units Sold")
fig.show()

fig_components = plot_components_plotly(model, forecast)
fig_components.show()
```

```
/usr/local/lib/python3.10/dist-packages/_plotly_utils/basevalidators.py:105: FutureWarning: The behavior of DatetimeProperties.to_p
  v = v.dt.to_pydatetime()
```




```
/usr/local/lib/python3.10/dist-packages/_plotly_utils/basevalidators.py:105: FutureWarning:

The behavior of DatetimeProperties.to_pydatetime is deprecated, in a future version this will return a Series containing python dat

/usr/local/lib/python3.10/dist-packages/_plotly_utils/basevalidators.py:105: FutureWarning:

The behavior of DatetimeProperties.to_pydatetime is deprecated, in a future version this will return a Series containing python dat

/usr/local/lib/python3.10/dist-packages/_plotly_utils/basevalidators.py:105: FutureWarning:

The behavior of DatetimeProperties.to_pydatetime is deprecated, in a future version this will return a Series containing python dat
```