Name: Pranav Sumesh

TASK 1:

```
#header files:
import numpy as np
import pandas as pd
import os
import gc
import warnings
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
import statsmodels.api as sm
from sklearn.metrics import mean_squared_error

pd.set_option('display.max_columns', None)
pd.options.display.float_format = '{:.2f}'.format
warnings.filterwarnings('ignore')

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

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

```
train = pd.read_csv('/content/drive/MyDrive/anarix/train.csv')
test = pd.read_csv('/content/drive/MyDrive/anarix/test.csv')
```

Exploratory Data Analysis (EDA):

Data Cleaning:

```
#filling empty cells with 0:
test.unit_price.fillna(0, inplace=True)
test.ad_spend.fillna(0, inplace=True)
train.units.fillna(0, inplace=True)
train.ad_spend.fillna(0, inplace=True)

train.head(10)
```

```
{"type":"dataframe","variable_name":"train"}
```

```
test.head(10)
```


```
{"summary":"{\n  \"name\": \"test\",\n  \"rows\": 2833,\n  \"fields\":
[\n    {\n      \"column\": \"ID\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"num_unique_values\": 2833,\n
\"samples\": [\n          \"2024-07-26_B0CR4C5WXS\",\n
\"2024-07-25_B09KTJRHC7\",\n          \"2024-07-11_B09MR4B13C\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"date\",\n      \"properties\":
{\n        \"dtype\": \"object\",\n        \"num_unique_values\": 28,\
n        \"samples\": [\n          \"2024-07-10\",\n          \"2024-
07-26\",\n          \"2024-07-09\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Item Id\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
155,\n        \"samples\": [\n          \"B0BNL3YY1C\",\n
\"B0BNL1RDVC\",\n          \"B0B31R1ZBV\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Item Name\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 142,\n        \"samples\": [\n
\"NapQueen 2'' 5-Zone Mattress Topper, Full\",\n          \"NapQueen
Elsa 6\\\" Innerspring Mattress, Full\",\n          \"NapQueen 2''
Green Tea Mattress Topper, Twin\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"ad_spend\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
565.6071891141796,\n        \"min\": 0.0,\n        \"max\": 18724.85,\
n        \"num_unique_values\": 1017,\n        \"samples\": [\n
276.25,\n          154.31,\n          6.2\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"anarix_id\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 1,\n        \"samples\": [\n
\"NAPQUEEN\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"unit_price\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 383.5853067975535,\n        \"min\": -
1988.18,\n        \"max\": 6870.0,\n        \"num_unique_values\":
480,\n        \"samples\": [\n          119.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"test"}
```


```
# Converting date columns to datetime:
train['date'] = pd.to_datetime(train['date'], format='%Y-%m-%d')
test['date'] = pd.to_datetime(test['date'], format='%Y-%m-%d')

#Creating the column "orderedrevenueamount" as it wasn't provided in
the dataset:
```

```
train['orderedrevenueamount'] = train['units'] * train['unit_price']
train.head(100)
```

{"type":"dataframe","variable_name":"train"}

```
#Dataset description:
print(train.describe())
print(train.info())
```

```
                                date  ad_spend     units
unit_price  \
count                         101490 101490.00 101490.00   101490.00

mean   2023-07-09 19:17:37.120898560     84.37      8.47      106.75

min              2022-04-12 00:00:00      0.00   -173.00    -8232.00

25%              2023-02-26 00:00:00      0.00      0.00         0.00

50%              2023-07-16 00:00:00      0.72      0.00         0.00

75%              2023-12-13 00:00:00     21.64      3.00         0.00

max              2024-05-31 00:00:00  47934.99   9004.00    21557.39

std                              NaN    464.35     62.69      425.70


       orderedrevenueamount      year     month       day
count             101490.00 101490.00 101490.00 101490.00
mean                3441.39   2023.06      6.05     15.88
min                 -197.67   2022.00      1.00      1.00
25%                    0.00   2023.00      3.00      8.00
50%                    0.00   2023.00      6.00     16.00
75%                    0.00   2023.00      9.00     23.00
max              9420579.43   2024.00     12.00     31.00
std                43949.60      0.61      3.49      8.81
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 101490 entries, 0 to 101489
Data columns (total 12 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   ID                    101490 non-null  object
 1   date                  101490 non-null  datetime64[ns]
 2   Item Id               101488 non-null  object
 3   Item Name             99658 non-null   object
 4   ad_spend              101490 non-null  float64
 5   anarix_id             101490 non-null  object
 6   units                 101490 non-null  float64
 7   unit_price            101490 non-null  float64
 8   orderedrevenueamount  101490 non-null  float64
```

```
 9   year                    101490 non-null  int32
 10  month                   101490 non-null  int32
 11  day                     101490 non-null  int32
dtypes: datetime64[ns](1), float64(4), int32(3), object(4)
memory usage: 8.1+ MB
None
```

Feature Engineering: Extracting year, month, and day from the date column.

```
# Feature Engineering
# Create year, month, and day features from date
train['year'] = train['date'].dt.year
train['month'] = train['date'].dt.month
train['day'] = train['date'].dt.day

test['year'] = test['date'].dt.year
test['month'] = test['date'].dt.month
test['day'] = test['date'].dt.day
```

Model Selection:

For the given task, I have chosen SARIMAX (Seasonal AutoRegressive Integrated Moving Average with eXogenous variables) model as:

- it is ideal for incorporating external factors such as money spent into advertisement and seasonal patterns into the forecasting model,
- the sales data exhibited clear seasonal patterns. SARIMAX can model these seasonal components directly.
- it further enhances the model's predictive accuracy by incorporating exogenous variables.

```
# Grouping by item id and aggregating sales data:
train_grouped = train.groupby(['Item Id', 'year', 'month',
'day']).agg({'units': 'sum', 'ad_spend': 'sum'}).reset_index()

train_grouped['date'] = pd.to_datetime(train_grouped[['year', 'month',
'day']])

# Pivoting data to get time series format:
train_pivot_units = train_grouped.pivot(index='date', columns='Item
Id', values='units').fillna(0)
train_pivot_ad_spend = train_grouped.pivot(index='date', columns='Item
Id', values='ad_spend').fillna(0)
# Extracting the columns of interest for the test dataset:
test_pivot_ad_spend = test.pivot(index='date', columns='Item Id',
values='ad_spend').fillna(0)

models = {}
predictions = {}
```

```
# Fitting a SARIMAX model for each item id and predict
for column in train_pivot_units.columns:
    # Building and fitting the model
    models[column] = sm.tsa.SARIMAX(train_pivot_units[column],
exog=train_pivot_ad_spend[column], order=(5, 1, 0))
    models[column] = models[column].fit(disp=False)

    # Forecasting
    if column in test_pivot_ad_spend.columns:
        predictions[column] =
models[column].forecast(steps=len(test_pivot_ad_spend),
exog=test_pivot_ad_spend[column])
    else:
        predictions[column] = [0] * len(test_pivot_ad_spend)

output = test[['date', 'Item Id']].copy()
output['TARGET'] = 0

# Populating predictions for test dataset
for index, row in test.iterrows():
    item_id = row['Item Id']
    if item_id in predictions:
        output.at[index, 'TARGET'] = predictions[item_id].iloc[-1]

# Saving the output to CSV
output.to_csv('p_output_predictions.csv', index=False)
output.head(10)
```


```
{"summary":"{\n  \"name\": \"output\",\n  \"rows\": 2833,\n
\"fields\": [\n    {\n      \"column\": \"date\",\n
\"properties\": {\n        \"dtype\": \"date\",\n        \"min\":
\"2024-07-01 00:00:00\",\n        \"max\": \"2024-07-28 00:00:00\",\n
\"num_unique_values\": 28,\n        \"samples\": [\n          \"2024-
07-10 00:00:00\",\n          \"2024-07-26 00:00:00\",\n
\"2024-07-09 00:00:00\"\n        ],\n        \"semantic_type\": \"\",\
n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Item Id\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 155,\n
\"samples\": [\n          \"B0BNL3YY1C\",\n          \"B0BNL1RDVC\",\n
\"B0B31R1ZBV\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"TARGET\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 17.02207598082672,\n        \"min\": -7.440420704633116,\n
\"max\": 105.53139631044414,\n        \"num_unique_values\": 153,\n
\"samples\": [\n          0.25759939979945995,\n
0.4580100403844407,\n          -0.1013571916754586\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"output"}
```

TASK 2:

```
# Group by item id and aggregate sales data
new_train_grouped = train.groupby(['Item Id', 'year', 'month',
'day']).agg({'units': 'sum'}).reset_index()

# Combine year, month, and day back into a single date column
new_train_grouped['date'] = pd.to_datetime(new_train_grouped[['year',
'month', 'day']])

# Pivot data to get time series format
new_train_pivot = new_train_grouped.pivot(index='date', columns='Item
Id', values='units').fillna(0)

# Model Building (Using ARIMA Model)
new_models = {}
new_predictions = {}
mse_scores = {}

# Train the model on the entire training data
for column in new_train_pivot.columns:
    # Build and fit the model
    new_models[column] = sm.tsa.ARIMA(new_train_pivot[column],
order=(5, 1, 0))
    new_models[column] = new_models[column].fit()

    # Forecast for the entire range including test dates
    test_dates = test['date'].unique()
    new_predictions[column] =
new_models[column].forecast(steps=len(test_dates))

# Creating the output DataFrame for test dataset
new_output = test[['ID', 'Item Id']].copy()
new_output['TARGET'] = 0

# Populate predictions for test dataset
for index, row in test.iterrows():
    item_id = row['Item Id']
    if item_id in new_predictions:
        new_output.at[index, 'TARGET'] =
new_predictions[item_id].iloc[-1]

# Save the output to CSV
new_output.to_csv('new_output_predictions.csv', index=False)

print(new_output.head())
```

```
                       ID     Item Id  TARGET
0  2024-07-01_B09KDR64LT  B09KDR64LT    0.55
1  2024-07-01_B09KDTS4DC  B09KDTS4DC    2.42
```

```
2  2024-07-01_B09KDTHJ6V  B09KDTHJ6V   -0.00
3  2024-07-01_B09KDQ2BWY  B09KDQ2BWY    0.38
4  2024-07-01_B09KDYY3SB  B09KDYY3SB    0.92
```