

Coffee sales

```
import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from statsmodels.tsa.statespace.sarimax import SARIMAX

import warnings

warnings.filterwarnings('ignore')


# Load the dataset

try:

    df = pd.read_csv("index.csv")


# Convert date columns

df['date'] = pd.to_datetime(df['date'])

df['datetime'] = pd.to_datetime(df['datetime'])


# Add time features

df['month'] = df['date'].dt.to_period('M')

df['day'] = df['date'].dt.day_name()

df['hour'] = df['datetime'].dt.hour


#-----

# 1. TIME SERIES EDA

#-----


# Daily Sales Trend

daily_sales = df.groupby('date')['money'].sum().reset_index()

plt.figure(figsize=(14,6))
```

```
sns.lineplot(data=daily_sales, x='date', y='money')  
plt.title("Daily Coffee Sales")  
plt.xlabel("Date")  
plt.ylabel("Sales")  
plt.xticks(rotation=45)  
plt.tight_layout()  
plt.show()
```

```
# Monthly Sales by Product
```

```
monthly_sales = df.groupby(['month', 'coffee_name'])['money'].sum().unstack().fillna(0)  
monthly_sales.plot(kind='line', figsize=(14, 6), title='Monthly Sales by Coffee Type')  
plt.xticks(rotation=45)  
plt.tight_layout()  
plt.show()
```

```
#-----
```

```
# 2. FORECAST NEXT DAY/WEEK/MONTH SALES
```

```
#-----
```

```
# Use total daily sales for forecasting
```

```
df_forecast = daily_sales.set_index('date')  
model = SARIMAX(df_forecast, order=(1,1,1), seasonal_order=(1,1,1,7))  
results = model.fit()
```

```
# Forecast next 30 days
```

```
forecast = results.get_forecast(steps=30)  
forecast_df = forecast.predicted_mean.reset_index()  
forecast_df.columns = ['date', 'predicted_sales']
```

```
# Plot Forecast
```

```
plt.figure(figsize=(14,6))
```

```
plt.plot(df_forecast.index, df_forecast['money'], label='Historical')
```

```
plt.plot(forecast_df['date'], forecast_df['predicted_sales'], label='Forecast', color='red')
```

```
plt.title("Sales Forecast- Next 30 Days")
```

```
plt.xlabel("Date")
```

```
plt.ylabel("Sales")
```

```
plt.legend()
```

```
plt.tight_layout()
```

```
plt.show()
```

```
#-----
```

```
# 3. SPECIFIC CUSTOMER PURCHASES
```

```
#-----
```

```
# Replace 'ANON-0000-0000-0001' with any card ID
```

```
customer_id = 'ANON-0000-0000-0001'
```

```
customer_data = df[df['card'] == customer_id]
```

```
print(f"\nPurchase History for Customer: {customer_id}")
```

```
print(customer_data[['date', 'coffee_name', 'money']])
```

```
# Visualize
```

```
plt.figure(figsize=(10,5))
```

```
sns.countplot(data=customer_data, x='coffee_name',  
order=customer_data['coffee_name'].value_counts().index)
```

```
plt.title(f"Coffee Types Purchased by {customer_id}")
```

```
plt.xticks(rotation=45)
```

```
plt.tight_layout()
```

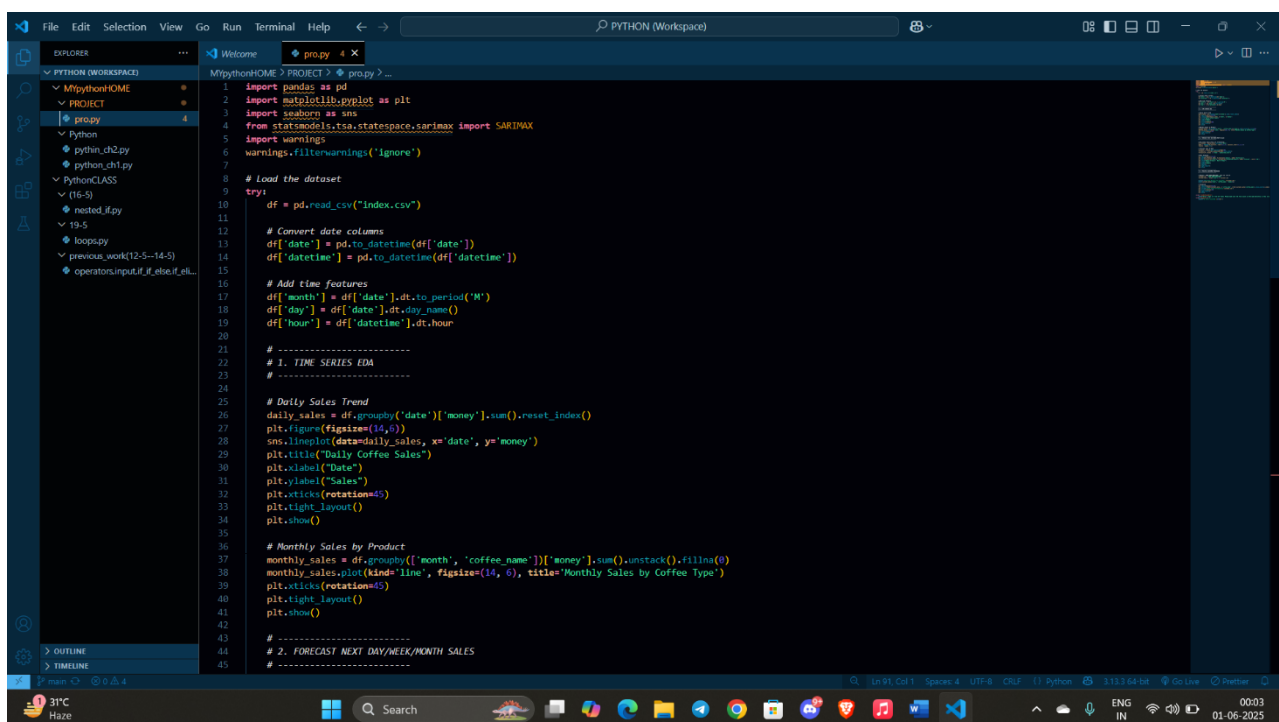
```
plt.show()
```

except FileNotFoundError:

```
    print("Error: index.csv file not found. Please make sure the file exists in the same directory  
as this script.")
```

except Exception as e:

```
    print(f"An error occurred: {str(e)}")
```



The screenshot shows a Visual Studio Code editor window titled 'PYTHON (Workspace)'. The Explorer sidebar on the left shows a project structure with folders 'PYTHON (WORKSPACE)', 'MyPythonHOME', 'PROJECT', and 'propy'. The 'propy' folder is selected, showing files like 'python_ch2.py', 'python_ch1.py', 'PythonCLASS', 'nested_if.py', 't9-5', 'loops.py', 'previous_work(12-5--14-5)', and 'operator.inputs_if_else_if_el...'. The main editor area displays a Python script with the following content:

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4 from statsmodels.tsa.statespace.sarimax import SARIMAX
5 import warnings
6 warnings.filterwarnings('ignore')
7
8 # Load the dataset
9 try:
10     df = pd.read_csv("index.csv")
11
12     # Convert date columns
13     df['date'] = pd.to_datetime(df['date'])
14     df['datetime'] = pd.to_datetime(df['datetime'])
15
16     # Add time features
17     df['month'] = df['date'].dt.to_period('M')
18     df['day'] = df['date'].dt.day_name()
19     df['hour'] = df['datetime'].dt.hour
20
21     # -----
22     # 1. TIME SERIES EDA
23     # -----
24
25     # Daily Sales Trend
26     daily_sales = df.groupby('date')['money'].sum().reset_index()
27     plt.figure(figsize=(14,6))
28     sns.lineplot(data=daily_sales, x='date', y='money')
29     plt.title("Daily Coffee Sales")
30     plt.xlabel("Date")
31     plt.ylabel("Sales")
32     plt.xticks(rotation=45)
33     plt.tight_layout()
34     plt.show()
35
36     # Monthly Sales by Product
37     monthly_sales = df.groupby(['month', 'coffee_name'])['money'].sum().unstack().fillna(0)
38     monthly_sales.plot(kind='line', figsize=(14, 6), title='Monthly Sales by Coffee Type')
39     plt.xticks(rotation=45)
40     plt.tight_layout()
41     plt.show()
42
43     # -----
44     # 2. FORECAST NEXT DAY/WEEK/MONTH SALES
45     # -----
```

```
File Edit Selection View Go Run Terminal Help PYTHON (Workspace)
EXPLORER
PYTHON (WORKSPACE)
MypythonHOME
PROJECT
propy
Python
python_ch2.py
python_ch1.py
PythonCLASS
(16.5)
nested_if.py
19.5
loops.py
previous_work(12.5--14.5)
operators.input_if_else_if_etc...
OUTLINE
TIMELINE
# Use total daily sales for forecasting
df_forecast = daily_sales.set_index('date')
model = SARIMAX(df_forecast, order=(1,1,1), seasonal_order=(1,1,1,7))
results = model.fit()

# Forecast next 30 days
forecast = results.get_forecast(steps=30)
forecast_df = forecast.predicted_mean.reset_index()
forecast_df.columns = ['date', 'predicted_sales']

# Plot Forecast
plt.figure(figsize=(14,6))
plt.plot(df_forecast.index, df_forecast['money'], label='Historical')
plt.plot(forecast_df['date'], forecast_df['predicted_sales'], label='Forecast', color='red')
plt.title("Sales Forecast - Next 30 Days")
plt.xlabel("Date")
plt.ylabel("Sales")
plt.legend()
plt.tight_layout()
plt.show()

# -----
# 3. SPECIFIC CUSTOMER PURCHASES
# -----

# Replace 'ANON-0000-0000-0001' with any card ID
customer_id = 'ANON-0000-0000-0001'
customer_data = df[df['card'] == customer_id]

print(f"\nPurchase History for Customer: {customer_id}")
print(customer_data[['date', 'coffee_name', 'money']])

# Visualize
plt.figure(figsize=(10,5))
sns.countplot(data=customer_data, x='coffee_name', order=customer_data['coffee_name'].value_counts().index)
plt.title(f"Coffee Types Purchased by {customer_id}")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

except FileNotFoundError:
    print("Error: index.csv file not found. Please make sure the file exists in the same directory as this script.")
except Exception as e:
    print(f"An error occurred: {str(e)}")
```

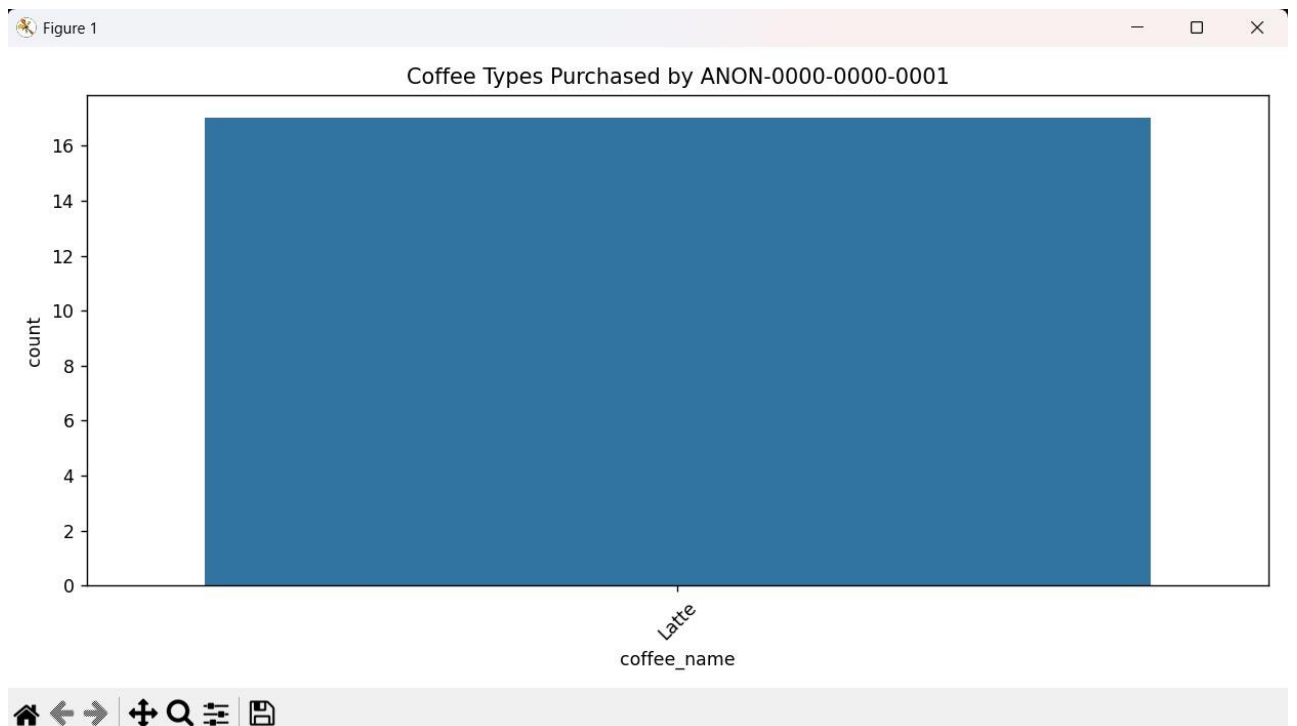
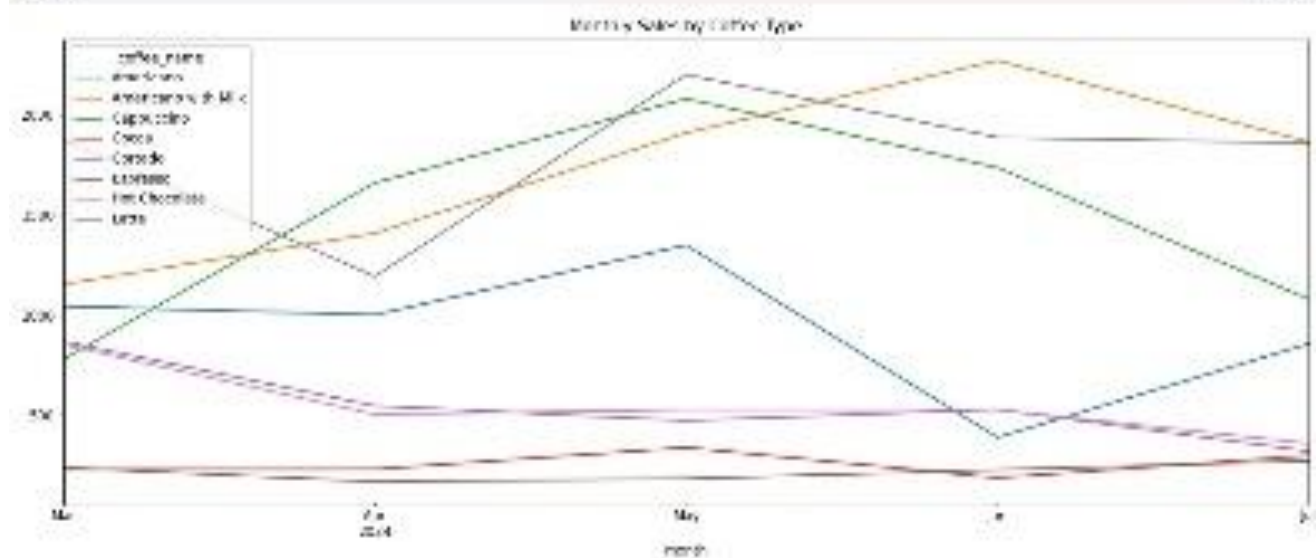
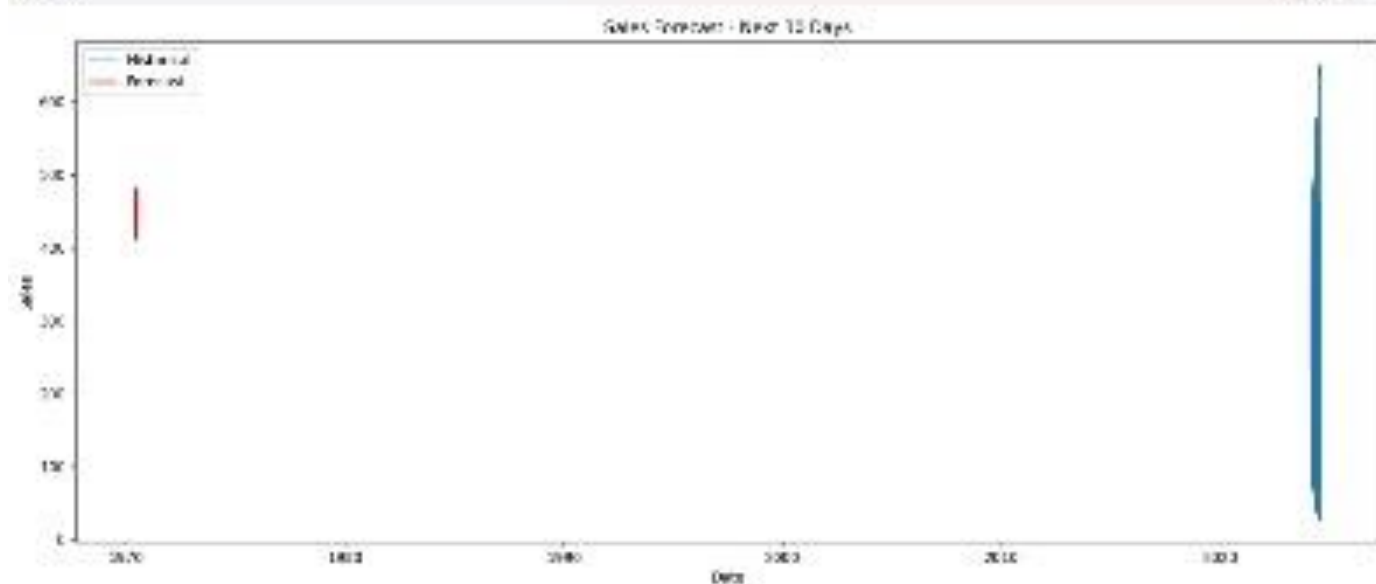


Figure 1



Navigation icons: back, forward, search, etc.

Figure 2



Navigation icons: back, forward, search, etc.

