

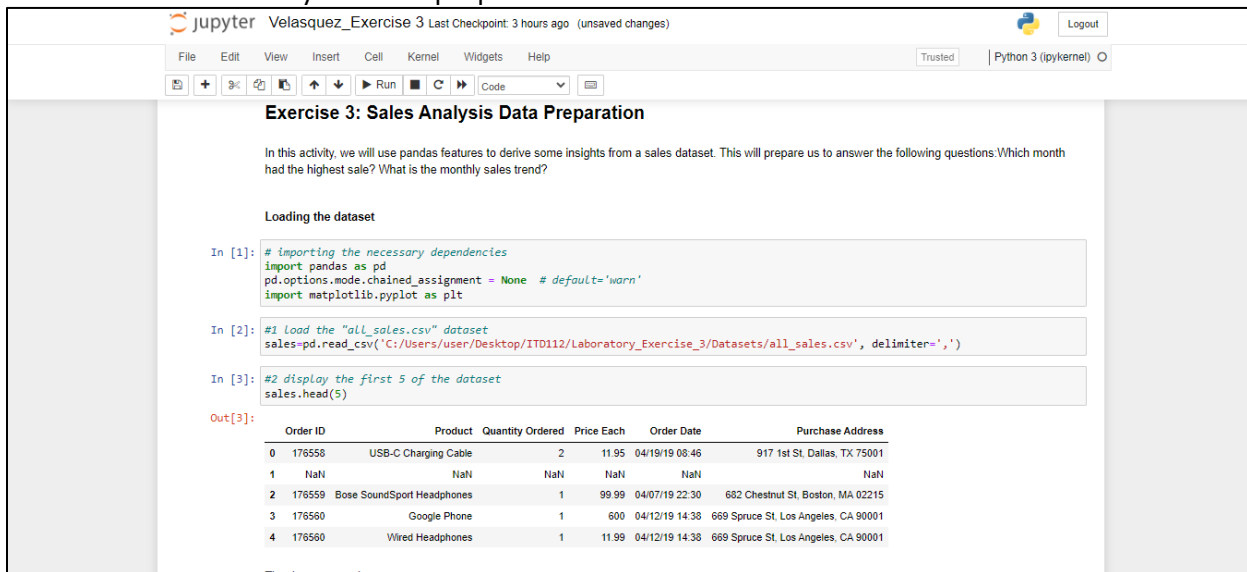
ITD 112 Laboratory Exercises #3

Name: Clint Joshua O. Velasquez

Complete the codes of the following exercises. Screenshot your output. Submit in pdf format.

Source Code Link: <https://github.com/kiyojiii/ITD112>

Exercise 3: Sales Analysis Data preparation



Exercise 3: Sales Analysis Data Preparation

In this activity, we will use pandas features to derive some insights from a sales dataset. This will prepare us to answer the following questions: Which month had the highest sale? What is the monthly sales trend?

Loading the dataset

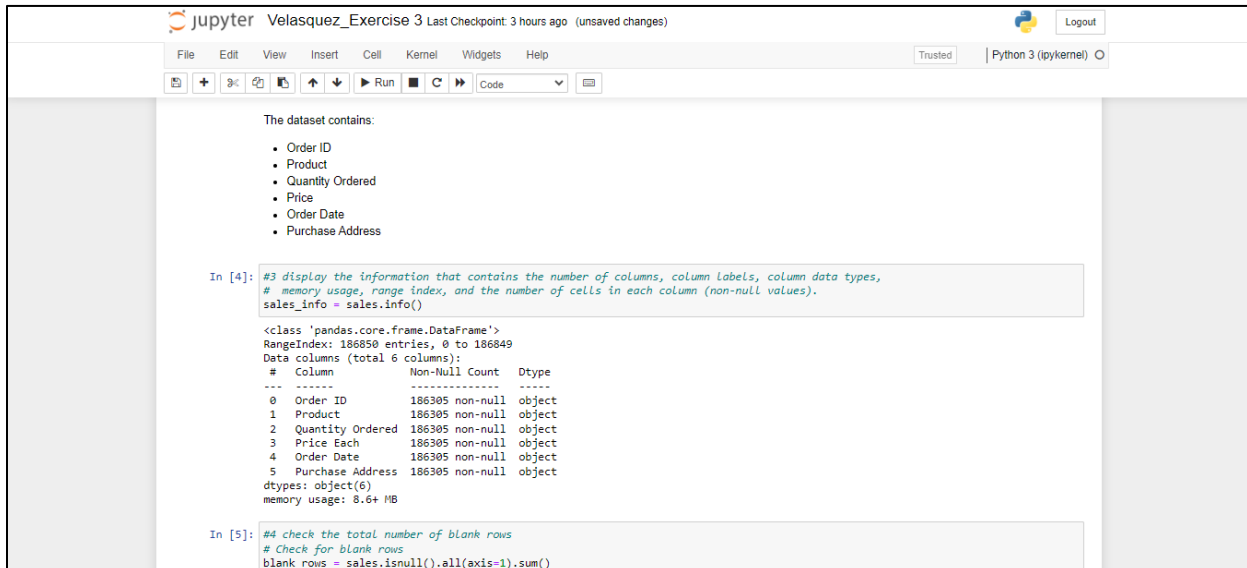
```
In [1]: # importing the necessary dependencies
import pandas as pd
pd.options.mode.chained_assignment = None  # default='warn'
import matplotlib.pyplot as plt
```

```
In [2]: #1 Load the "all_sales.csv" dataset
sales=pd.read_csv('C:/Users/user/Desktop/ITD112/Laboratory_Exercise_3/Datasets/all_sales.csv', delimiter=',')
```

```
In [3]: #2 display the first 5 of the dataset
sales.head(5)
```

Out[3]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001
1	NaN	NaN	NaN	NaN	NaN	NaN
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215
3	176560	Google Phone	1	600	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001



The dataset contains:

- Order ID
- Product
- Quantity Ordered
- Price
- Order Date
- Purchase Address

```
In [4]: #3 display the information that contains the number of columns, column labels, column data types,
# memory usage, range index, and the number of cells in each column (non-null values).
sales_info = sales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 186850 entries, 0 to 186849
Data columns (total 6 columns):
#   Column                Non-Null Count  Dtype
---  --
0   Order ID              186305 non-null  object
1   Product               186305 non-null  object
2   Quantity Ordered      186305 non-null  object
3   Price Each            186305 non-null  object
4   Order Date            186305 non-null  object
5   Purchase Address      186305 non-null  object
dtypes: object(6)
memory usage: 8.6+ MB
```

```
In [5]: #4 check the total number of blank rows
# Check for blank rows
blank_rows = sales.isnull().all(axis=1).sum()
```

Jupyter Velasquez_Exercise 3 Last Checkpoint: 3 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

```
In [5]: #4 check the total number of blank rows
# Check for blank rows
blank_rows = sales.isnull().all(axis=1).sum()

# Display the total number of blank rows
print(f"Total number of blank rows: {blank_rows}")

Total number of blank rows: 545

In [6]: #5 Remove all nulls/blanks using dropna() method.
# Remove all rows with null or blank values
sales = sales.dropna()

# Reset the index after dropping rows
sales = sales.reset_index(drop=True)

# Show the data
sales.head()
```

Out[6]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001
1	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215
2	176560	Google Phone	1	600	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001
3	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001
4	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001

Jupyter Velasquez_Exercise 3 Last Checkpoint: 3 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

```
In [7]: #6 Create the 'Month' column from the 'Order Date' column using
# sales['Month'] = sales['Order Date'].str[0:2]
# display the first 5 of the dataset

# Create the 'Month' column
sales['Month'] = sales['Order Date'].str[0:2]

# Display the first 5 rows of the dataset
sales.head(5)
```

Out[7]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001	04
1	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	04
2	176560	Google Phone	1	600	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	04
3	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	04
4	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	04

```
In [8]: #7 Convert data type of Month Column from object to integer using
# sales['Month'] = sales['Month'].astype('int32')
# There will be ValueError: invalid literal for int() with base 10: '04'

# Convert data type of Month Column from object to integer
sales['Month'] = sales['Month'].astype('int32', errors='ignore')

# Show data
sales
```

Out[8]:

Jupyter Velasquez_Exercise 3 Last Checkpoint: 3 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Out[8]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001	04
1	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	04
2	176560	Google Phone	1	600	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	04
3	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	04
4	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	04
...
186300	259353	AAA Batteries (4-pack)	3	2.99	09/17/19 20:56	840 Highland St, Los Angeles, CA 90001	09
186301	259354	iPhone	1	700	09/01/19 16:00	216 Dogwood St, San Francisco, CA 94016	09
186302	259355	iPhone	1	700	09/23/19 07:39	220 12th St, San Francisco, CA 94016	09
186303	259356	34in Ultrawide Monitor	1	379.99	09/19/19 17:30	511 Forest St, San Francisco, CA 94016	09
186304	259357	USB-C Charging Cable	1	11.95	09/30/19 00:18	250 Meadow St, San Francisco, CA 94016	09

186305 rows x 7 columns

```
In [9]: #8 Create Or_dum dataframe to check the "Or's" in the Month Column using
# Or_dum = sales[sales['Month']=="04"]
# display Or_dum first 5 rows
Or_dum = sales[sales['Month']=="04"]
Or_dum.head(5)
```

Out[9]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month
517	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Or
1166	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Or

Jupyter Velasquez_Exercise 3 Last Checkpoint: 3 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

```
In [9]: # Create Or_dum dataframe to check the "Or's" in the Month Column using
# Or_dum = sales[sales['Month']!= "Or"]
# display Or_dum first 5 rows
Or_dum = sales[sales['Month'] == "Or"]
Or_dum.head(5)
```

Out[9]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month
517	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Or
1146	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Or
1152	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Or
2869	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Or
2884	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Or

```
In [10]: #9 Remove "Or's" from the Month Column using
# sales = sales[sales['Month']!= "Or"]
# Remove 'Or' values from the Month Column
sales = sales[sales['Month'] != 'Or']
sales.head()
```

Out[10]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001	04
1	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	04
2	176560	Google Phone	1	600	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	04
3	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	04
4	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	04

Jupyter Velasquez_Exercise 3 Last Checkpoint: 3 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

```
In [11]: #10 Check whether "Or's" from the Month Column still exist. (Or's must be gone)
# Check if 'Or's' still exist in the Month Column
or_exist = any(sales['Month'] == 'Or')
print("Do 'Or's' still exist in the Month Column?", or_exist)

Do 'Or's' still exist in the Month Column? False
```

```
In [12]: #10 Convert data type of Month Column from object to integer using
# sales['Month'] = sales['Month'].astype('int32')
# There will be ValueError: invalid literal for int() with base 10: 'Or'
sales['Month'] = sales['Month'].astype('int32', errors='ignore')
sales.head(5)
```

Out[12]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001	4
1	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	4
2	176560	Google Phone	1	600	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4
3	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4
4	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	4

```
In [13]: #11 check the new number of columns
# display sales first 5 rows
# Check the number of columns
num_columns = sales.shape[1]
print(f"Number of columns: {num_columns}")

# Display the first 5 rows of the dataset
sales.head(5)
```

Jupyter Velasquez_Exercise 3 Last Checkpoint: 3 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Number of columns: 7

Out[13]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001	4
1	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	4
2	176560	Google Phone	1	600	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4
3	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4
4	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	4

```
In [14]: #12 Convert Quantity Ordered and price each columns to the correct data type
# sales['Quantity Ordered'] = pd.to_numeric(sales['Quantity Ordered'])
# sales['Price Each'] = pd.to_numeric(sales['Price Each'])
# display .info()
# Convert 'Quantity Ordered' to numeric
sales['Quantity Ordered'] = pd.to_numeric(sales['Quantity Ordered'], errors='coerce')

# Convert 'Price Each' to numeric
sales['Price Each'] = pd.to_numeric(sales['Price Each'], errors='coerce')

# Display information about the dataset
sales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 185950 entries, 0 to 186304
Data columns (total 7 columns):
 #   Column              Non-Null Count  Dtype
---  ---
 0   Order ID            185950 non-null  object
 1   Product              185950 non-null  object
 2   Quantity Ordered     185950 non-null  object
 3   Price Each           185950 non-null  object
 4   Order Date           185950 non-null  object
 5   Purchase Address     185950 non-null  object
 6   Month                185950 non-null  object
```

Jupyter Velasquez_Exercise 3 Last Checkpoint: 3 hours ago (autosaved) Python 3 (ipykernel)

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 185950 entries, 0 to 186304
Data columns (total 7 columns):
 #   Column              Non-Null Count  Dtype  
---  --
 0   Order ID            185950 non-null  object  
 1   Product             185950 non-null  object  
 2   Quantity Ordered    185950 non-null  int64   
 3   Price Each          185950 non-null  float64  
 4   Order Date          185950 non-null  object  
 5   Purchase Address    185950 non-null  object  
 6   Month               185950 non-null  int32   
dtypes: float64(1), int32(1), int64(1), object(4)
memory usage: 10.6+ MB

In [15]: #13 Add Cost column, where Cost = sales['Quantity Ordered'] * sales['Price Each']
# display sales first 5 rows
# Add the 'Cost' column
sales['Cost'] = sales['Quantity Ordered'] * sales['Price Each']

# Display the first 5 rows of the updated dataset
sales.head()

Out[15]:
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Cost
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001	4	23.90
1	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	4	99.99
2	176560	Google Phone	1	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	600.00
3	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	11.99
4	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	4	11.99

Jupyter Velasquez_Exercise 3 Last Checkpoint: 3 hours ago (autosaved) Python 3 (ipykernel)

```
Out[15]:
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Cost
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001	4	23.90
1	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	4	99.99
2	176560	Google Phone	1	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	600.00
3	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	11.99
4	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	4	11.99

```
In [16]: # Ensure the 'Month' column is in integer format (if not already done)
sales['Month'] = sales['Month'].astype(int)

# Clean and convert the 'Cost' column to numeric
sales['Cost'] = pd.to_numeric(sales['Cost'], errors='coerce')

# Drop rows with NaN values in the 'Cost' column
sales.dropna(subset=['Cost'], inplace=True)

# Calculate the total cost for each month
monthly_total_cost = sales.groupby('Month')['Cost'].sum()

# Display the total cost for each month
print("Total Monthly Cost for Each Month:")
print(monthly_total_cost)

# Calculate the sum of total monthly cost
total_cost_sum = monthly_total_cost.sum()

# Display the sum of total monthly cost
print(f"Sum of Total Monthly Cost: ${total_cost_sum}")

Total Monthly Cost for Each Month:
Month
1    1822256.73
2    2202022.42
3    2807100.38
4    3390670.24
5    3152606.75
6    2577802.26
7    2647775.76
8    2244467.88
9    2097560.13
10   3336396.00
```

Jupyter Velasquez_Exercise 3 Last Checkpoint: 3 hours ago (autosaved) Python 3 (ipykernel)

```
In [16]: # Ensure the 'Month' column is in integer format (if not already done)
sales['Month'] = sales['Month'].astype(int)

# Clean and convert the 'Cost' column to numeric
sales['Cost'] = pd.to_numeric(sales['Cost'], errors='coerce')

# Drop rows with NaN values in the 'Cost' column
sales.dropna(subset=['Cost'], inplace=True)

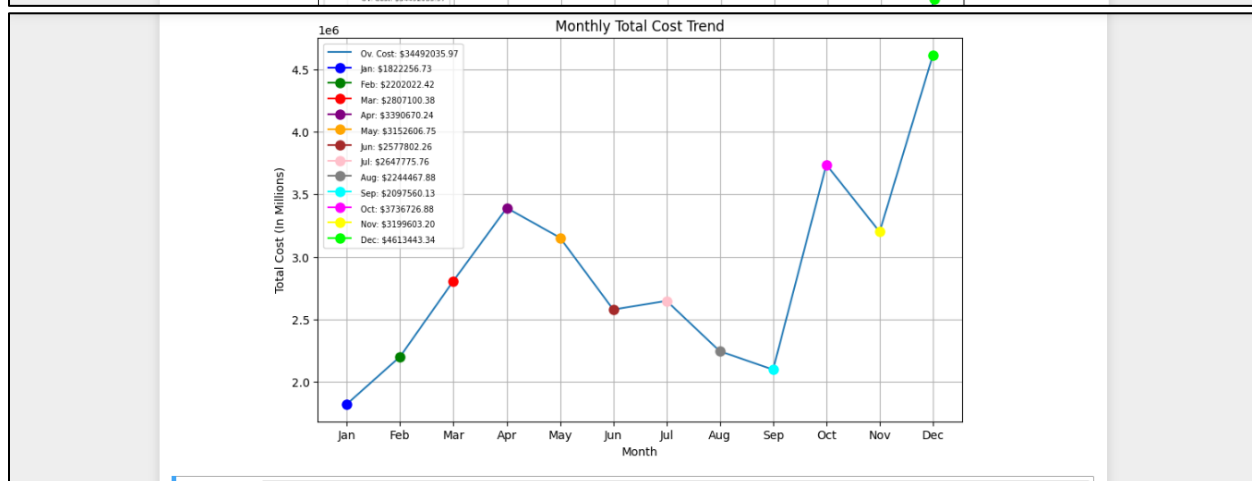
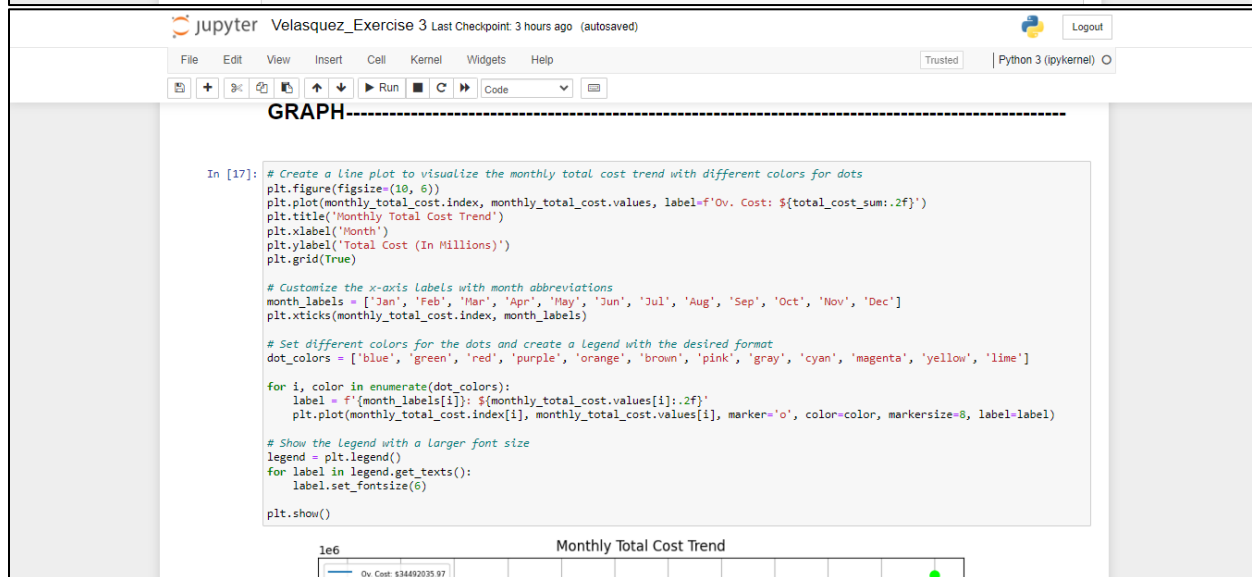
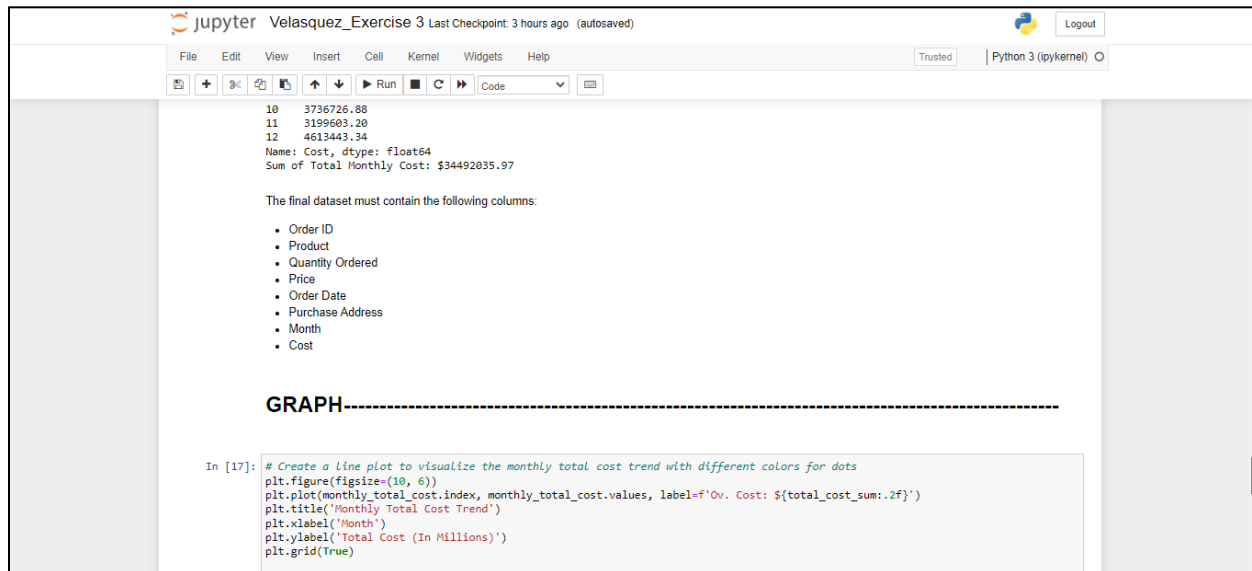
# Calculate the total cost for each month
monthly_total_cost = sales.groupby('Month')['Cost'].sum()

# Display the total cost for each month
print("Total Monthly Cost for Each Month:")
print(monthly_total_cost)

# Calculate the sum of total monthly cost
total_cost_sum = monthly_total_cost.sum()

# Display the sum of total monthly cost
print(f"Sum of Total Monthly Cost: ${total_cost_sum}")

Total Monthly Cost for Each Month:
Month
1    1822256.73
2    2202022.42
3    2807100.38
4    3390670.24
5    3152606.75
6    2577802.26
7    2647775.76
8    2244467.88
9    2097560.13
10   3336396.00
```



```
Jupyter Velasquez_Exercise 3 Last Checkpoint: 3 minutes ago (unsaved changes)
Python 3 (ipykernel)

In [49]: # Group the data by the 'Month' column and count the number of sales per month
monthly_sales_count = sales.groupby('Month').size()

# Calculate the sum of total monthly sales
total_sales_sum = monthly_sales_count.sum()

# Define colors for each month
colors = ['skyblue', 'coral', 'green', 'salmon', 'seagreen', 'pink',
          'gray', 'blue', 'cyan', 'green', 'yellow', 'coral']

# Create a bar plot to visualize the number of sales per month
plt.figure(figsize=(10, 6))
bars = plt.bar(monthly_sales_count.index, monthly_sales_count.values, color=colors)
plt.title(f'Number of Sales per Month (Overall Sales: {total_sales_sum})')
plt.xlabel('Month')
plt.ylabel('Number of Sales')
plt.grid(axis='y')

# Customize the x-axis labels with month abbreviations
month_labels = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']
plt.xticks(monthly_sales_count.index, month_labels)

# Add values on top of each bar
for bar, value in zip(bars, monthly_sales_count.values):
    plt.text(bar.get_x() + bar.get_width() / 2, value, str(value), ha='center', va='bottom')

# Create a legend with the specified format [Month]:[number of sales]
legend_labels = [f'{month_labels[i]}: {monthly_sales_count.values[i]}' for i in range(len(month_labels))]

# Create a legend for the bars with labels for each month
plt.legend(bars, legend_labels, title='Monthly Sales', fontsize=7)
```

