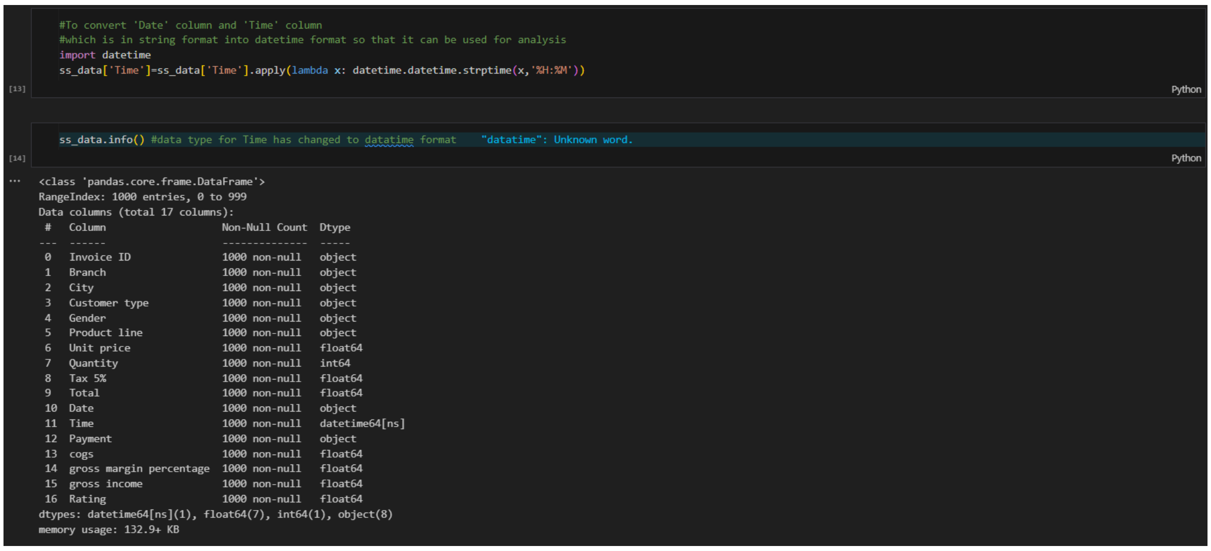
Code Implementation (Python)



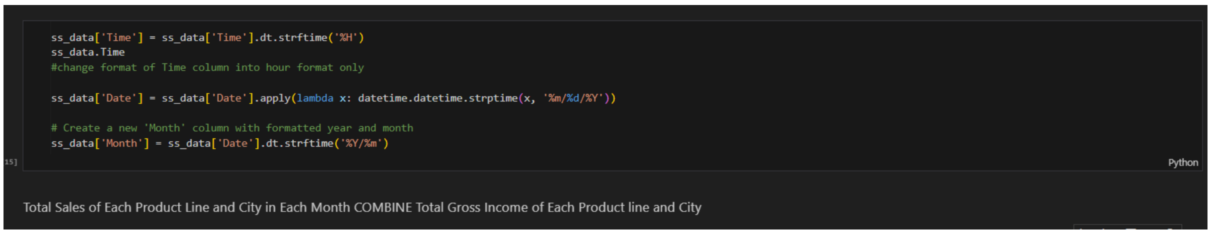
```
#To convert 'Date' column and 'Time' column
#which is in string format into datetime format so that it can be used for analysis
import datetime
ss_data['Time']=ss_data['Time'].apply(lambda x: datetime.datetime.strptime(x,'%H:%M'))
```

[13] Python

```
ss_data.info() #data type for Time has changed to datatime format    "datatime": Unknown word.
```

[14] Python

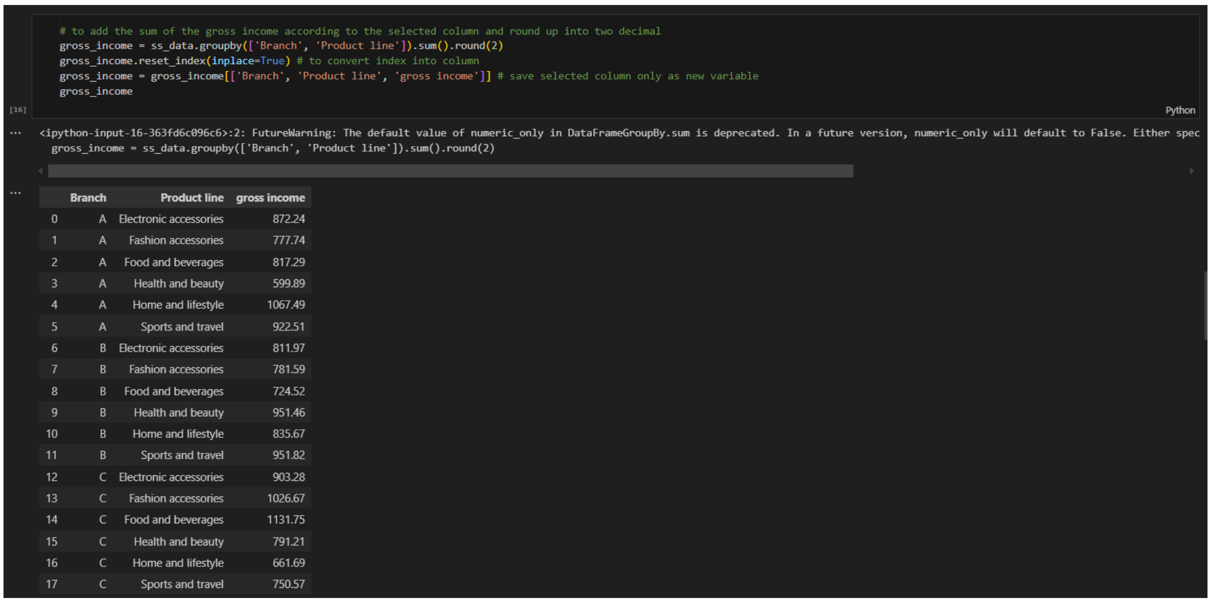
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 17 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Invoice ID               1000 non-null   object
 1   Branch                   1000 non-null   object
 2   City                     1000 non-null   object
 3   Customer type            1000 non-null   object
 4   Gender                   1000 non-null   object
 5   Product line             1000 non-null   object
 6   Unit price               1000 non-null   float64
 7   Quantity                 1000 non-null   int64
 8   Tax 5%                   1000 non-null   float64
 9   Total                    1000 non-null   float64
 10  Date                     1000 non-null   object
 11  Time                     1000 non-null   datetime64[ns]
 12  Payment                  1000 non-null   object
 13  cogs                     1000 non-null   float64
 14  gross margin percentage  1000 non-null   float64
 15  gross income             1000 non-null   float64
 16  Rating                   1000 non-null   float64
dtypes: datetime64[ns](1), float64(7), int64(1), object(8)
memory usage: 132.9+ KB
```



```
ss_data['Time'] = ss_data['Time'].dt.strftime('%H')
ss_data.Time
#change format of Time column into hour format only

ss_data['Date'] = ss_data['Date'].apply(lambda x: datetime.datetime.strptime(x, '%m/%d/%Y'))

# Create a new 'Month' column with formatted year and month
ss_data['Month'] = ss_data['Date'].dt.strftime('%Y/%m')
```

15] Python

Total Sales of Each Product Line and City in Each Month COMBINE Total Gross Income of Each Product line and City

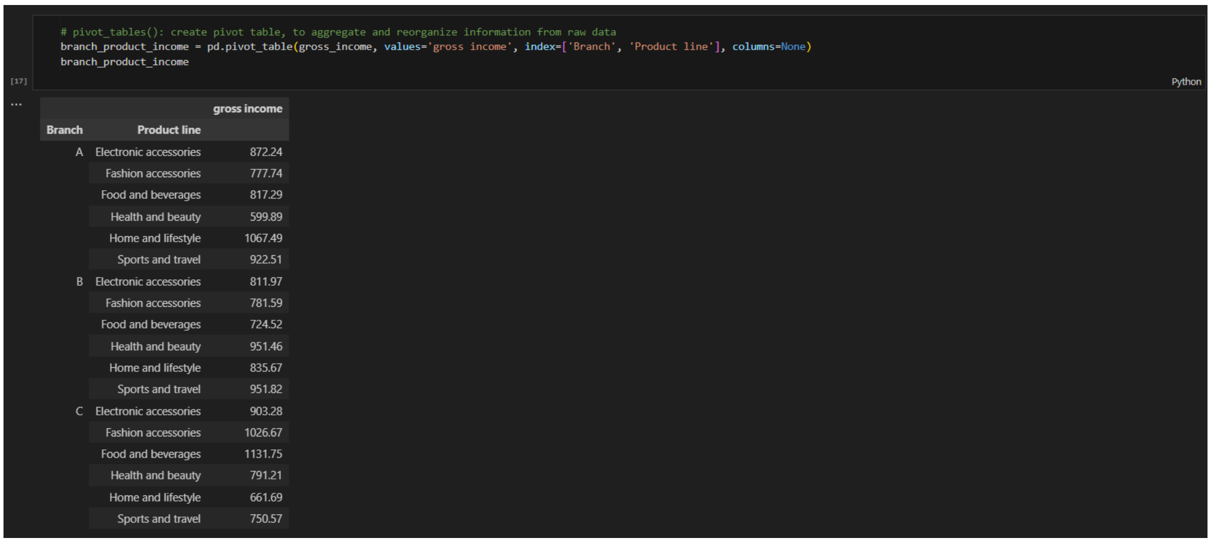


```
# to add the sum of the gross income according to the selected column and round up into two decimal
gross_income = ss_data.groupby(['Branch', 'Product line']).sum().round(2)
gross_income.reset_index(inplace=True) # to convert index into column
gross_income = gross_income[['Branch', 'Product line', 'gross income']] # save selected column only as new variable
gross_income
```

[16] Python

```
<ipython-input-16-363fd6c096c6>:2: FutureWarning: The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future version, numeric_only will default to False. Either spec
  gross_income = ss_data.groupby(['Branch', 'Product line']).sum().round(2)
```

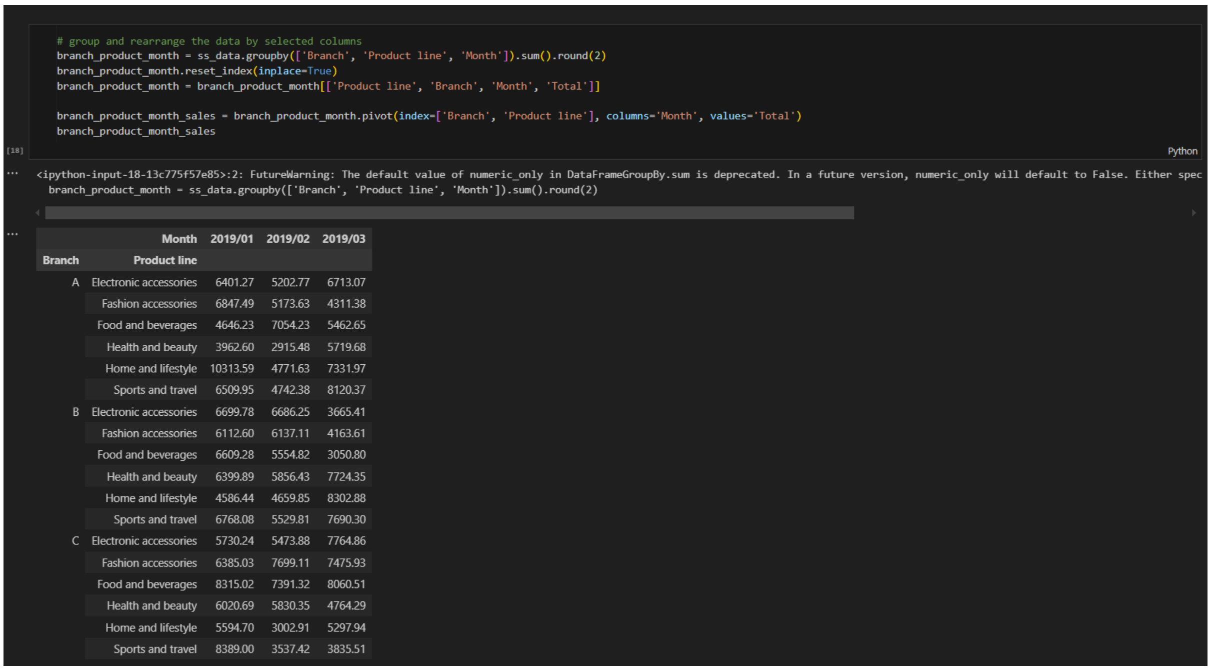
	Branch	Product line	gross income
0	A	Electronic accessories	872.24
1	A	Fashion accessories	777.74
2	A	Food and beverages	817.29
3	A	Health and beauty	599.89
4	A	Home and lifestyle	1067.49
5	A	Sports and travel	922.51
6	B	Electronic accessories	811.97
7	B	Fashion accessories	781.59
8	B	Food and beverages	724.52
9	B	Health and beauty	951.46
10	B	Home and lifestyle	835.67
11	B	Sports and travel	951.82
12	C	Electronic accessories	903.28
13	C	Fashion accessories	1026.67
14	C	Food and beverages	1131.75
15	C	Health and beauty	791.21
16	C	Home and lifestyle	661.69
17	C	Sports and travel	750.57



```
# pivot_tables(): create pivot table, to aggregate and reorganize information from raw data
branch_product_income = pd.pivot_table(gross_income, values='gross income', index=['Branch', 'Product line'], columns=None)
branch_product_income
```

[17] Python

		gross income
Branch	**Product line**	
A	Electronic accessories	872.24
	Fashion accessories	777.74
	Food and beverages	817.29
	Health and beauty	599.89
	Home and lifestyle	1067.49
	Sports and travel	922.51
B	Electronic accessories	811.97
	Fashion accessories	781.59
	Food and beverages	724.52
	Health and beauty	951.46
	Home and lifestyle	835.67
	Sports and travel	951.82
C	Electronic accessories	903.28
	Fashion accessories	1026.67
	Food and beverages	1131.75
	Health and beauty	791.21
	Home and lifestyle	661.69
	Sports and travel	750.57

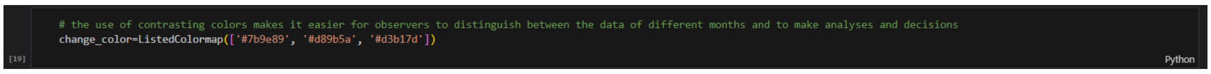


```
# group and rearrange the data by selected columns
branch_product_month = ss_data.groupby(['Branch', 'Product line', 'Month']).sum().round(2)
branch_product_month.reset_index(inplace=True)
branch_product_month = branch_product_month[['Product line', 'Branch', 'Month', 'Total']]

branch_product_month_sales = branch_product_month.pivot(index=['Branch', 'Product line'], columns='Month', values='Total')
branch_product_month_sales
```

[18] Python

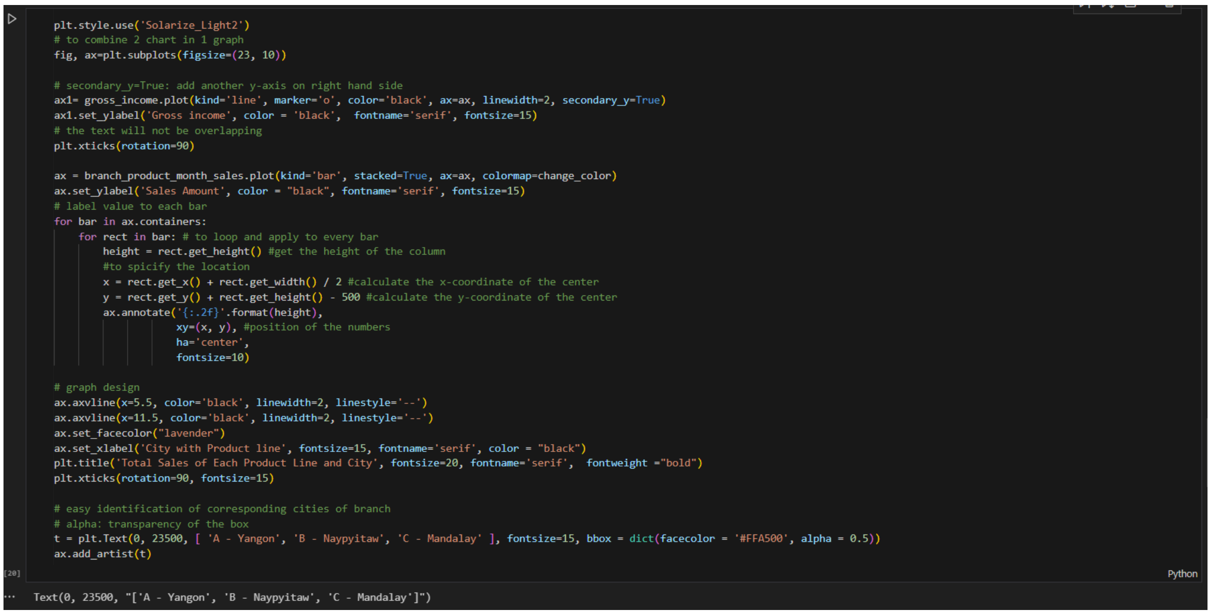
```
<ipython-input-18-13c775f57e85>:2: FutureWarning: The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future version, numeric_only will default to False. Either spec
  branch_product_month = ss_data.groupby(['Branch', 'Product line', 'Month']).sum().round(2)
```

	Month	**2019/01**	**2019/02**	**2019/03**
Branch	**Product line**			
A	Electronic accessories	6401.27	5202.77	6713.07
	Fashion accessories	6847.49	5173.63	4311.38
	Food and beverages	4646.23	7054.23	5462.65
	Health and beauty	3962.60	2915.48	5719.68
	Home and lifestyle	10313.59	4771.63	7331.97
	Sports and travel	6509.95	4742.38	8120.37
B	Electronic accessories	6699.78	6686.25	3665.41
	Fashion accessories	6112.60	6137.11	4163.61
	Food and beverages	6609.28	5554.82	3050.80
	Health and beauty	6399.89	5856.43	7724.35
	Home and lifestyle	4586.44	4659.85	8302.88
	Sports and travel	6768.08	5529.81	7690.30
C	Electronic accessories	5730.24	5473.88	7764.86
	Fashion accessories	6385.03	7699.11	7475.93
	Food and beverages	8315.02	7391.32	8060.51
	Health and beauty	6020.69	5830.35	4764.29
	Home and lifestyle	5594.70	3002.91	5297.94
	Sports and travel	8389.00	3537.42	3835.51



```
# the use of contrasting colors makes it easier for observers to distinguish between the data of different months and to make analyses and decisions
change_color=ListedColormap(['#7b9e89', '#d89b5a', '#d3b17d'])
```

[19] Python

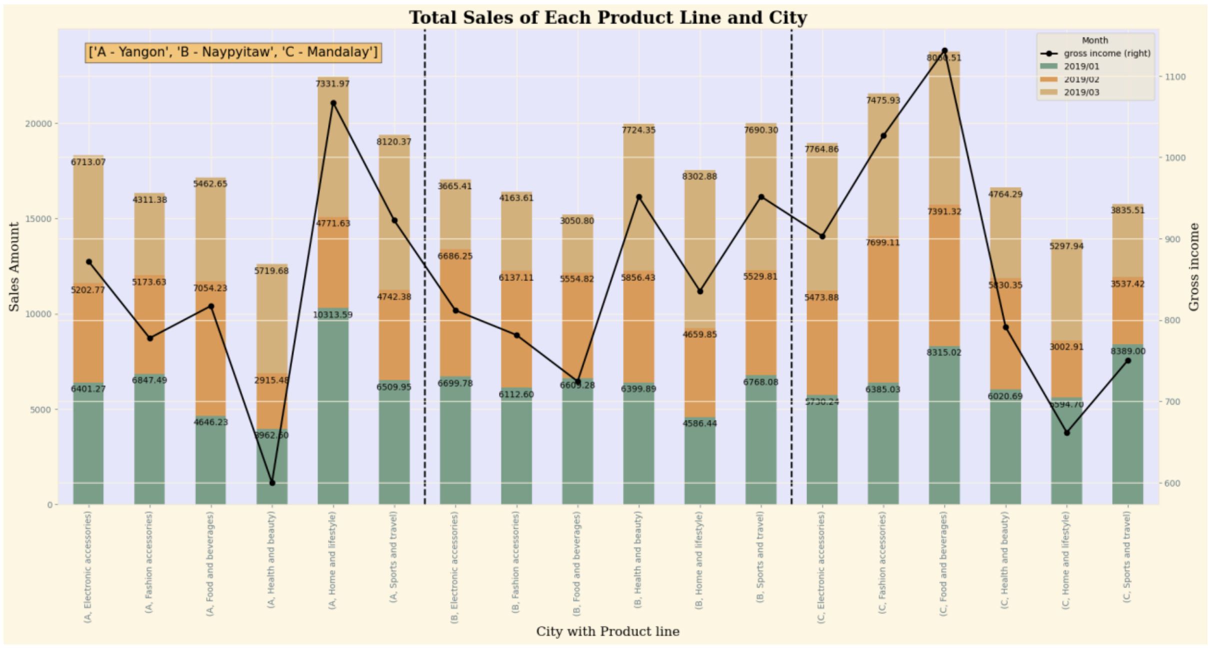


```
plt.style.use('Solarize_Light2')
# to combine 2 chart in 1 graph
fig, ax=plt.subplots(figsize=(23, 10))

# secondary_y=True: add another y-axis on right hand side
ax1= gross_income.plot(kind='line', marker='o', color='black', ax=ax, linewidth=2, secondary_y=True)
ax1.set_ylabel('Gross income', color = 'black',  fontname='serif', fontsize=15)
# the text will not be overlapping
plt.xticks(rotation=90)

ax = branch_product_month_sales.plot(kind='bar', stacked=True, ax=ax, colormap=change_color)
ax.set_ylabel('Sales Amount', color = "black", fontname='serif', fontsize=15)
# label value to each bar
for bar in ax.containers:
    for rect in bar: # to loop and apply to every bar
        height = rect.get_height() #get the height of the column
        #to spicify the location
        x = rect.get_x() + rect.get_width() / 2 #calculate the x-coordinate of the center
        y = rect.get_y() + rect.get_height() - 500 #calculate the y-coordinate of the center
        ax.annotate('{:.2f}'.format(height),
                    xy=(x, y), #position of the numbers
                    ha='center',
                    fontsize=10)

# graph design
ax.axvline(x=5.5, color='black', linewidth=2, linestyle='--')
ax.axvline(x=11.5, color='black', linewidth=2, linestyle='--')
ax.set_facecolor("lavender")
ax.set_xlabel('City with Product line', fontsize=15, fontname='serif', color = "black")
plt.title('Total Sales of Each Product Line and City', fontsize=20, fontname='serif',  fontweight ="bold")
plt.xticks(rotation=90, fontsize=15)

# easy identification of corresponding cities of branch
# alpha: transparency of the box
t = plt.Text(0, 23500, [ 'A - Yangon', 'B - Naypyitaw', 'C - Mandalay' ], fontsize=15, bbox = dict(facecolor = '#FFA500', alpha = 0.5))
ax.add_artist(t)
```

[20]

Python

```
Text(0, 23500, "['A - Yangon', 'B - Naypyitaw', 'C - Mandalay']")
```


Total Sales of Each Product Line and City
['A - Yangon', 'B - Naypyitaw', 'C - Mandalay']
Month
gross income (right)
2019/01
2019/02
2019/03
Sales Amount
Gross income
City with Product line
(A, Electronic accessories)
(A, Fashion accessories)
(A, Food and beverages)
(A, Health and beauty)
(A, Home and lifestyle)
(A, Sports and travel)
(B, Electronic accessories)
(B, Fashion accessories)
(B, Food and beverages)
(B, Health and beauty)
(B, Home and lifestyle)
(B, Sports and travel)
(C, Electronic accessories)
(C, Fashion accessories)
(C, Food and beverages)
(C, Health and beauty)
(C, Home and lifestyle)
(C, Sports and travel)