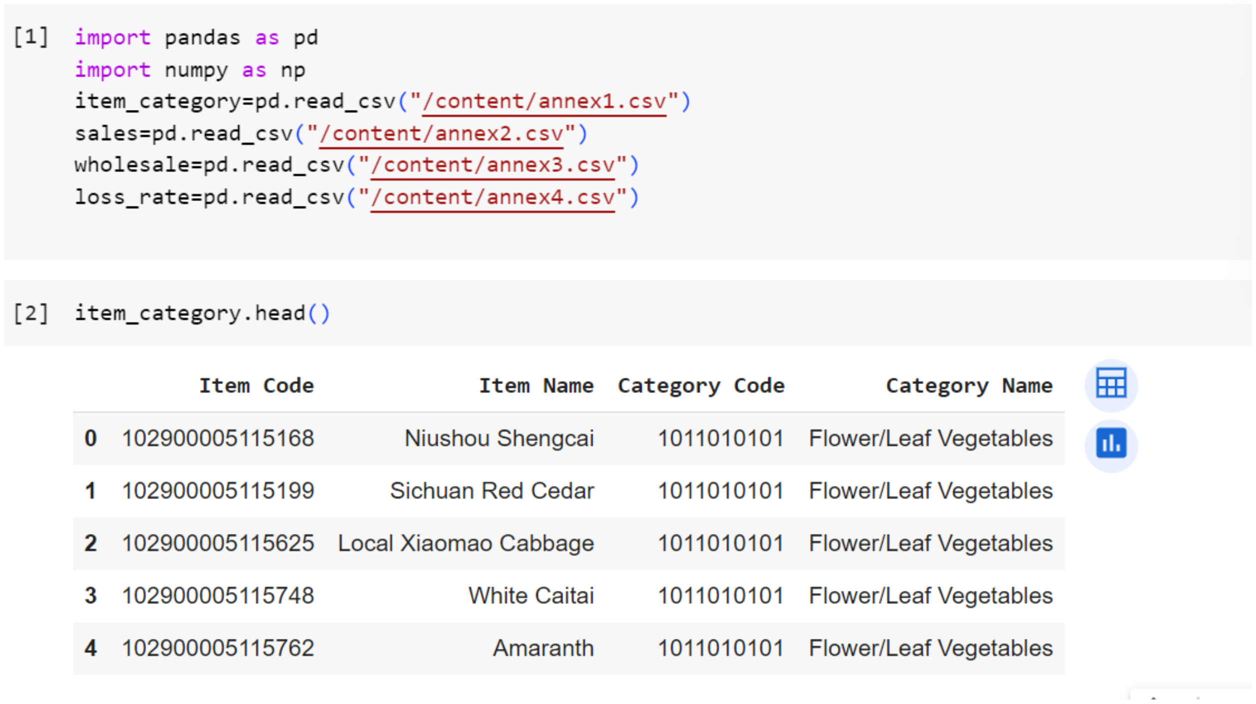
Sales Analysis Project- Python

In today's competitive business landscape, data-driven insights are paramount for organizations to thrive. Sales analysis plays a pivotal role in understanding market trends, customer behaviors, and optimizing business strategies. As businesses accumulate vast amounts of sales data, the need for robust analytical tools becomes increasingly essential. Python, with its versatility and powerful libraries, has emerged as a go-to language for data analysis and visualization.

The Sales Analysis Project aims to leverage the capabilities of Python to provide comprehensive insights into sales performance, customer segmentation, product trends, and forecasting. By harnessing the potential of Python's libraries such as Pandas, NumPy, Matplotlib, and Seaborn, this project endeavors to empower businesses with actionable intelligence derived from their sales data.

Through this project, users will be able to:

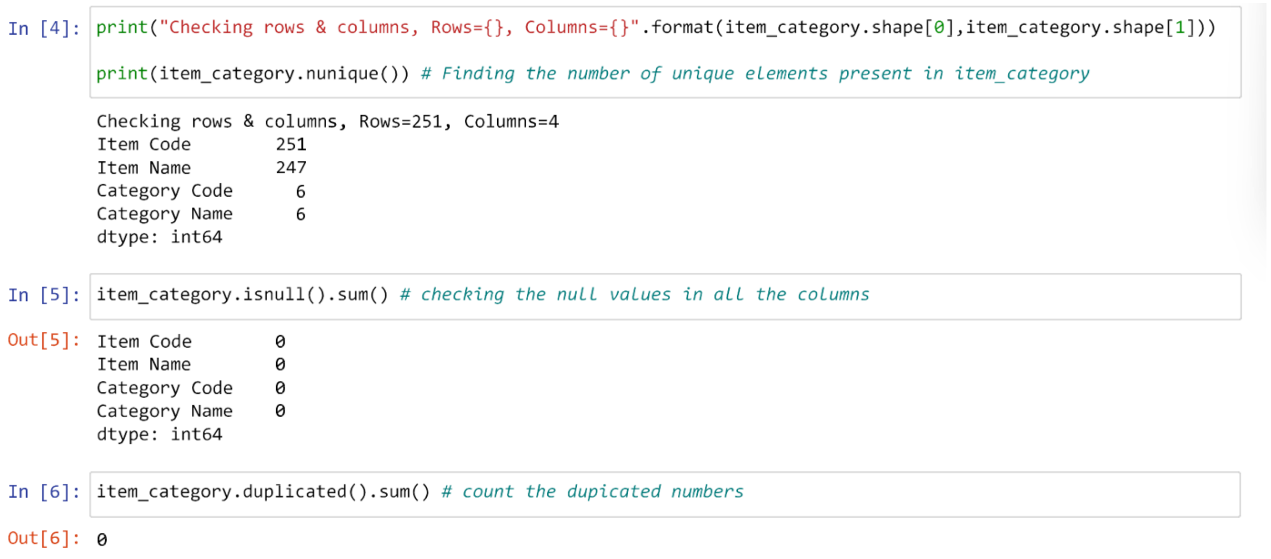
Explore Data: Dive deep into sales datasets to understand the underlying patterns, anomalies, and correlations.
Visualize Trends: Create insightful visualizations and charts to represent sales trends, seasonality, and geographical distribution.
Perform Segmentation: Segment customers based on various attributes such as demographics, purchase history, and preferences to tailor marketing strategies.
Forecast Sales: Utilize time-series analysis techniques to forecast future sales trends and anticipate demand fluctuations.
Evaluate Performance: Assess the effectiveness of sales campaigns, promotional activities, and product launches through rigorous analysis.
Generate Reports: Generate customizable reports and dashboards to communicate findings effectively to stakeholders.
By embarking on this Sales Analysis Project, businesses can gain a competitive edge by making data-driven decisions, optimizing resource allocation, and enhancing customer satisfaction. With Python as the backbone of the project, users can leverage its flexibility, scalability, and community support to adapt to evolving business needs and extract maximum value from their sales data.



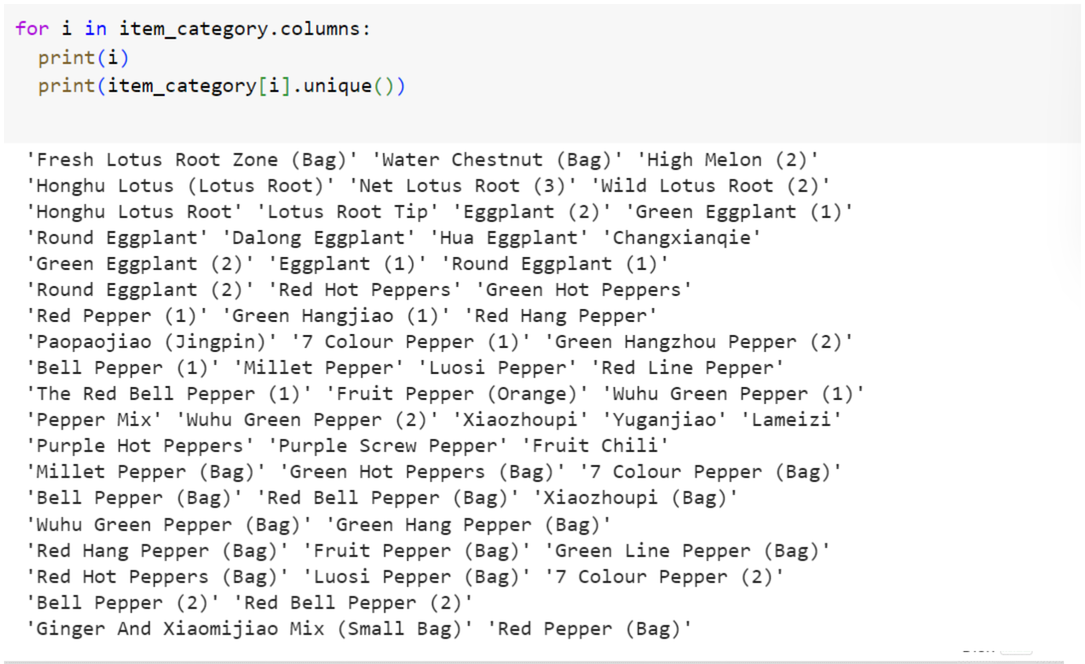
```
[1]  import pandas as pd
     import numpy as np
     item_category=pd.read_csv("/content/annex1.csv")
     sales=pd.read_csv("/content/annex2.csv")
     wholesale=pd.read_csv("/content/annex3.csv")
     loss_rate=pd.read_csv("/content/annex4.csv")
```

```
[2]  item_category.head()
```

	Item Code	Item Name	Category Code	Category Name
0	102900005115168	Niushou Shengcai	1011010101	Flower/Leaf Vegetables
1	102900005115199	Sichuan Red Cedar	1011010101	Flower/Leaf Vegetables
2	102900005115625	Local Xiaomao Cabbage	1011010101	Flower/Leaf Vegetables
3	102900005115748	White Caitai	1011010101	Flower/Leaf Vegetables
4	102900005115762	Amaranth	1011010101	Flower/Leaf Vegetables

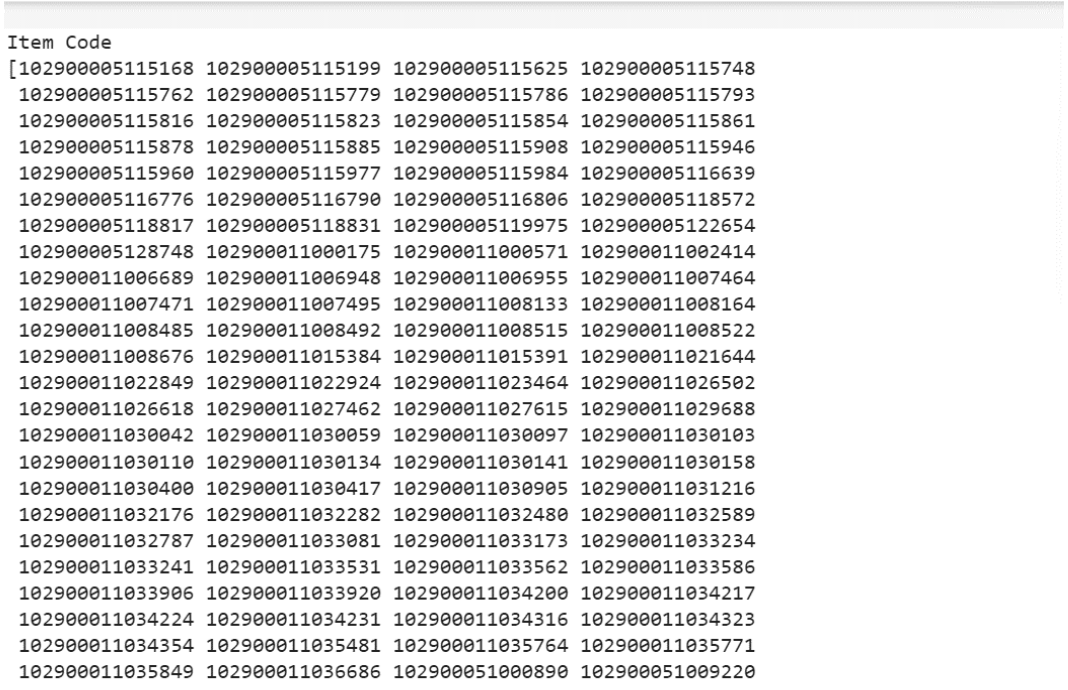


```
In [4]: print("Checking rows & columns, Rows={}, Columns={}".format(item_category.shape[0],item_category.shape[1]))

        print(item_category.nunique()) # Finding the number of unique elements present in item_category

        Checking rows & columns, Rows=251, Columns=4
        Item Code        251
        Item Name        247
        Category Code      6
        Category Name      6
        dtype: int64

In [5]: item_category.isnull().sum() # checking the null values in all the columns

Out[5]: Item Code        0
        Item Name        0
        Category Code    0
        Category Name    0
        dtype: int64

In [6]: item_category.duplicated().sum() # count the dupicated numbers

Out[6]: 0
```



```python
for i in item_category.columns:
  print(i)
  print(item_category[i].unique())
```

```
 'Fresh Lotus Root Zone (Bag)' 'Water Chestnut (Bag)' 'High Melon (2)'
 'Honghu Lotus (Lotus Root)' 'Net Lotus Root (3)' 'Wild Lotus Root (2)'
 'Honghu Lotus Root' 'Lotus Root Tip' 'Eggplant (2)' 'Green Eggplant (1)'
 'Round Eggplant' 'Dalong Eggplant' 'Hua Eggplant' 'Changxianqie'
 'Green Eggplant (2)' 'Eggplant (1)' 'Round Eggplant (1)'
 'Round Eggplant (2)' 'Red Hot Peppers' 'Green Hot Peppers'
 'Red Pepper (1)' 'Green Hangjiao (1)' 'Red Hang Pepper'
 'Paopaojiao (Jingpin)' '7 Colour Pepper (1)' 'Green Hangzhou Pepper (2)'
 'Bell Pepper (1)' 'Millet Pepper' 'Luosi Pepper' 'Red Line Pepper'
 'The Red Bell Pepper (1)' 'Fruit Pepper (Orange)' 'Wuhu Green Pepper (1)'
 'Pepper Mix' 'Wuhu Green Pepper (2)' 'Xiaozhoupi' 'Yuganjiao' 'Lameizi'
 'Purple Hot Peppers' 'Purple Screw Pepper' 'Fruit Chili'
 'Millet Pepper (Bag)' 'Green Hot Peppers (Bag)' '7 Colour Pepper (Bag)'
 'Bell Pepper (Bag)' 'Red Bell Pepper (Bag)' 'Xiaozhoupi (Bag)'
 'Wuhu Green Pepper (Bag)' 'Green Hang Pepper (Bag)'
 'Red Hang Pepper (Bag)' 'Fruit Pepper (Bag)' 'Green Line Pepper (Bag)'
 'Red Hot Peppers (Bag)' 'Luosi Pepper (Bag)' '7 Colour Pepper (2)'
 'Bell Pepper (2)' 'Red Bell Pepper (2)'
 'Ginger And Xiaomijiao Mix (Small Bag)' 'Red Pepper (Bag)'
```



```
Item Code
[102900005115168 102900005115199 102900005115625 102900005115748
 102900005115762 102900005115779 102900005115786 102900005115793
 102900005115816 102900005115823 102900005115854 102900005115861
 102900005115878 102900005115885 102900005115908 102900005115946
 102900005115960 102900005115977 102900005115984 102900005116639
 102900005116776 102900005116790 102900005116806 102900005118572
 102900005118817 102900005118831 102900005119975 102900005122654
 102900005128748 102900011000175 102900011000571 102900011002414
 102900011006689 102900011006948 102900011006955 102900011007464
 102900011007471 102900011007495 102900011008133 102900011008164
 102900011008485 102900011008492 102900011008515 102900011008522
 102900011008676 102900011015384 102900011015391 102900011021644
 102900011022849 102900011022924 102900011023464 102900011026502
 102900011026618 102900011027462 102900011027615 102900011029688
 102900011030042 102900011030059 102900011030097 102900011030103
 102900011030110 102900011030134 102900011030141 102900011030158
 102900011030400 102900011030417 102900011030905 102900011031216
 102900011032176 102900011032282 102900011032480 102900011032589
 102900011032787 102900011033081 102900011033173 102900011033234
 102900011033241 102900011033531 102900011033562 102900011033586
 102900011033906 102900011033920 102900011034200 102900011034217
 102900011034224 102900011034231 102900011034316 102900011034323
 102900011034354 102900011035481 102900011035764 102900011035771
 102900011035849 102900011036686 102900051000890 102900051009220
```

```
Item Name
['Niushou Shengcai' 'Sichuan Red Cedar' 'Local Xiaomao Cabbage'
 'White Caitai' 'Amaranth' 'Yunnan Shengcai' 'Zhuyecai' 'Chinese Cabbage'
 'Nanguajian' 'Shanghaiqing' 'Radish Leaves' 'Niushou Youcai'
 'Garden Chrysanthemum' 'Caidian Quinoa Artemisia' 'Caixin' 'Muercai'
 'Wandoujian' 'Yunnan Lettuces' 'Machixian' 'Local Spinach'
 'Yellow Xincai (1)' 'Black Rapeseed' 'Local Shanghaiqing' 'Spinach'
 'Wawacai' 'Hongshujian' 'Zhijiang Red Bolt' 'Huanghuacai' 'Kuaicai'
 'Suizhou Bubble Green' 'Panax Notoginseng' 'Dongmenkou Xiaobaicai'
 'Foreign Garland Chrysanthemum ' 'Ice Grass' 'Perilla' 'Mint'
 'The Dandelion' 'Siguajian' 'Naibaicai' 'Mustard' 'Big Broccoli'
 'Miantiaocai' 'Sweet Chinese Cabbage' 'Jicai' 'Malan Head' 'Ganlanye'
 'Hongshan Caitai' 'The Local Yellow Youcai' 'Green Caitai'
 'Xiaoqingcai (1)' 'Fresh Rice Dumplings Leaves' 'Aihao' 'Naibai Caimiao'
 'Juhua Youcai' 'Shuanggou Cabbage' 'Zhijiang Red Bolt (Bag)'
 'Yunnan Lettuce (Bag)' 'Yunnan Leaf Lettuce (Bag)'
 'Garden Chrysanthemum (Bag)' 'Spinach (Bag)' 'Caixin (Bag)'
 'Shanghai Green (Bag)' 'Xiaoqingcai (2)' 'Hongshan Shoutidai'
 'Hongshan Gift Box' 'Yuxingcao (Bag)' 'Ice Grass (Box)' 'Basil (Bag)'
 'Xiangtianhongcaitai (Bag)' 'Artemisia Stelleriana' 'Yuxingcao '
 'Zhuyecai (Bag)' 'Chuncai' 'Sophora Japonica' 'Hongshujian (Bag)'
 'Caidian Quinoa Artemisia (Bag)' 'Red Coral (Leaf)' 'Red Oak Leaf'
 'Green Butter' 'Powcan Mountain Chinese Cabbage ' 'Huangxincai (2)'
 'Amaranth (Bag)' 'Chinese Cabbage (Bag)' 'Xiaoqingcai (Bag)'
```

```
 'Xixia Xianggu Mushroom (Bag)' 'Mushroom And Leaf Mix (Bag)'
 'Velvet Antler Mushroom (Box)' 'Chinese Caterpillar Fungus Flowers'
 'The Crab Flavor Mushroom (Bag)' 'Haixian Mushroom (Bag) (2)'
 'Embroidered Aureus' 'Embroidered Aureus (Bag)'
 'Needle Mushroom (Bag) (3)' 'Needle Mushroom (Bag) (2)'
 'Needle Mushroom (Box)' 'The White Mushroom (2)'
 'The Crab Flavor Mushroom (2)' 'The White Mushroom (Box)'
 'The Crab Flavor Mushroom (Box)' 'Haixian Mushroom (Bag) (4)'
 'Haixian Mushroom (Bunch)' 'Haixian Mushroom (Bag) (3)'
 'Chinese Caterpillar Fungus Flowers (Box) (2)'
 'Hfyg Haixian Mushroom (Bunch)']
Category Code
[1011010101 1011010201 1011010402 1011010501 1011010504 1011010801]
Category Name
['Flower/Leaf\xa0Vegetables' 'Cabbage' 'Aquatic Tuberous Vegetables'
 'Solanum' 'Capsicum' 'Edible Mushroom']
```

In [8]:
```
# checking the data type of item category table
item_category.dtypes
```

Out[8]:
```
Item Code         int64
Item Name        object
Category Code     int64
Category Name    object
dtype: object
```

In [9]:
```
# It will give stats of the numerical values columns
item_category.describe().astype(int)
```

Out[9]:

	Item Code	Category Code
count	251	251
mean	-2147483648	1011010414
std	-2147483648	291
min	-2147483648	1011010101
25%	-2147483648	1011010101
50%	-2147483648	1011010501
75%	-2147483648	1011010801
max	-2147483648	1011010801

In [10]:
```
sales.head()
```

Out[10]:

	Date	Time	Item Code	Quantity Sold (kilo)	Unit Selling Price (RMB/kg)	Sale or Return	Discount (Yes/No)
0	2020-07-01	09:15:07.924	102900005117056	0.396	7.6	sale	No
1	2020-07-01	09:17:27.295	102900005115960	0.849	3.2	sale	No
2	2020-07-01	09:17:33.905	102900005117056	0.409	7.6	sale	No
3	2020-07-01	09:19:45.450	102900005115823	0.421	10.0	sale	No
4	2020-07-01	09:20:23.686	102900005115908	0.539	8.0	sale	No

In [11]:
```
sales.dtypes
# we have to change the wrong datatype to right data type
# for eg- Time is in object datatype we have to change into timeseries
```

Out[11]:
```
Date                             object
Time                             object
Item Code                         int64
Quantity Sold (kilo)            float64
Unit Selling Price (RMB/kg)     float64
Sale or Return                   object
Discount (Yes/No)                object
dtype: object
```

```
In [12]: sales["Date"]=pd.to_datetime(sales["Date"]) # converting the datatype to datetime
```

```
In [13]: sales.dtypes
```

```
Out[13]: Date                              datetime64[ns]
         Time                                      object
         Item Code                                  int64
         Quantity Sold (kilo)                     float64
         Unit Selling Price (RMB/kg)              float64
         Sale or Return                            object
         Discount (Yes/No)                         object
         dtype: object
```

```
In [14]: sales.isnull().sum()
```

```
Out[14]: Date                           0
         Time                           0
         Item Code                      0
         Quantity Sold (kilo)           0
         Unit Selling Price (RMB/kg)    0
         Sale or Return                 0
         Discount (Yes/No)              0
         dtype: int64
```

```
In [15]: sales.duplicated().sum()  # check the number of duplicated rows in sales table
```

```
Out[15]: 0
```

```
In [16]: print("Checking rows & columns, Rows={}, Columns={}".format(sales.shape[0], sales.shape[1]))
         print(sales.nunique()) # Finding the number of unique elements present in sales
```

```
         Checking rows & columns, Rows=878503, Columns=7
         Date                             1085
         Time                           849632
         Item Code                         246
         Quantity Sold (kilo)             2794
         Unit Selling Price (RMB/kg)       264
         Sale or Return                      2
         Discount (Yes/No)                   2
         dtype: int64
```

```
In [17]: sales.head()
```

Out[17]:

	Date	Time	Item Code	Quantity Sold (kilo)	Unit Selling Price (RMB/kg)	Sale or Return	Discount (Yes/No)
0	2020-07-01	09:15:07.924	102900005117056	0.396	7.6	sale	No
1	2020-07-01	09:17:27.295	102900005115960	0.849	3.2	sale	No
2	2020-07-01	09:17:33.905	102900005117056	0.409	7.6	sale	No
3	2020-07-01	09:19:45.450	102900005115823	0.421	10.0	sale	No
4	2020-07-01	09:20:23.686	102900005115908	0.539	8.0	sale	No

In [18]:
```
wholesale.head()
```

Out[18]:

	Date	Item Code	Wholesale Price (RMB/kg)
0	2020-07-01	102900005115762	3.88
1	2020-07-01	102900005115779	6.72
2	2020-07-01	102900005115786	3.19
3	2020-07-01	102900005115793	9.24
4	2020-07-01	102900005115823	7.03

In [19]:
```
# checking the data type of whole sale  table
wholesale.dtypes
```

Out[19]:
```
Date                         object
Item Code                     int64
Wholesale Price (RMB/kg)    float64
dtype: object
```

In [20]:
```
wholesale.isnull().sum() # checking the null
```

Out[20]:
```
Date                        0
Item Code                   0
Wholesale Price (RMB/kg)    0
dtype: int64
```

In [21]:
```
wholesale["Date"]=pd.to_datetime(wholesale["Date"]) # converting the date column from
```

In [22]:
```
wholesale.dtypes # after converting the Date column from object data type into datetime
```

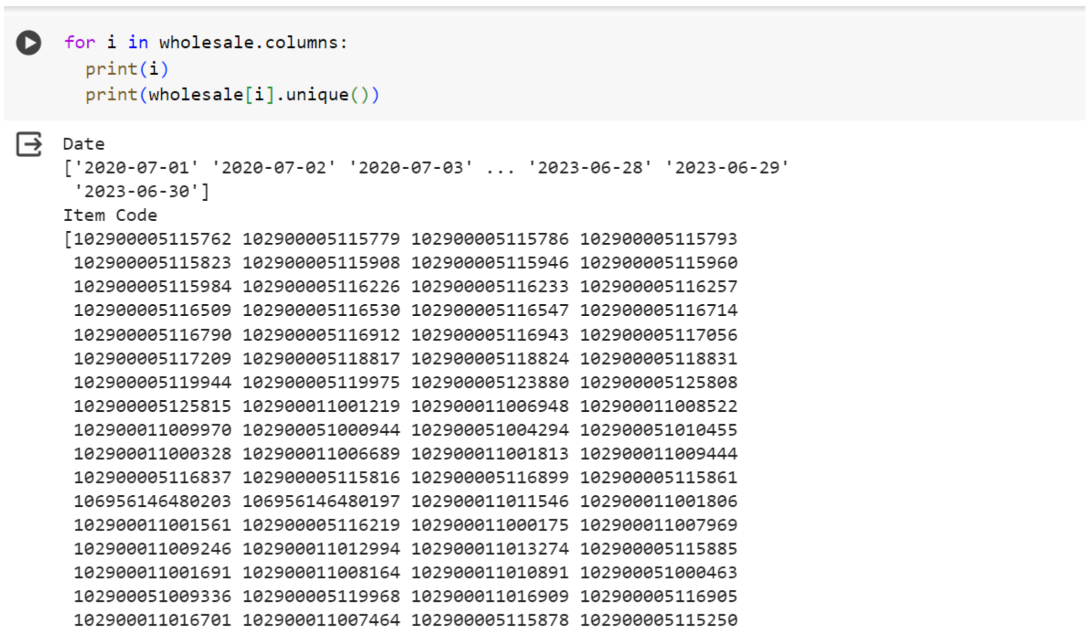
Out[22]:
```
Date                        datetime64[ns]
Item Code                            int64
Wholesale Price (RMB/kg)           float64
dtype: object
```

In [23]:
```
print("Checking rows & columns, Rows={}, Columns={}".format(wholesale.shape[0], wholesale.shape[1]))
print(wholesale.nunique()) # Finding the number of unique elements present in item_category
```

```
Checking rows & columns, Rows=55982, Columns=3
Date                        1091
Item Code                    251
Wholesale Price (RMB/kg)    2380
dtype: int64
```

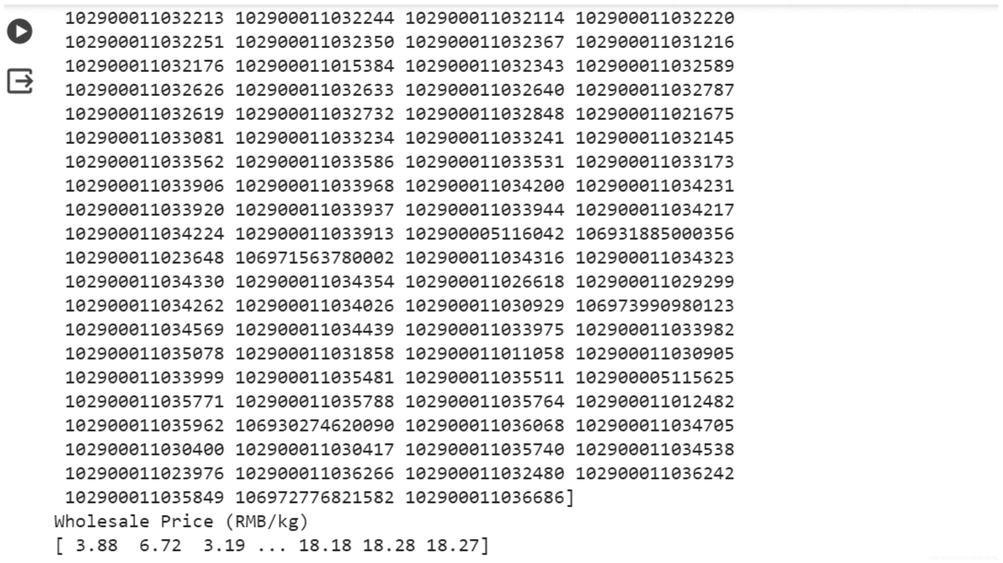
In [24]:
```
wholesale.duplicated().sum()
```

Out[24]:
```
0
```

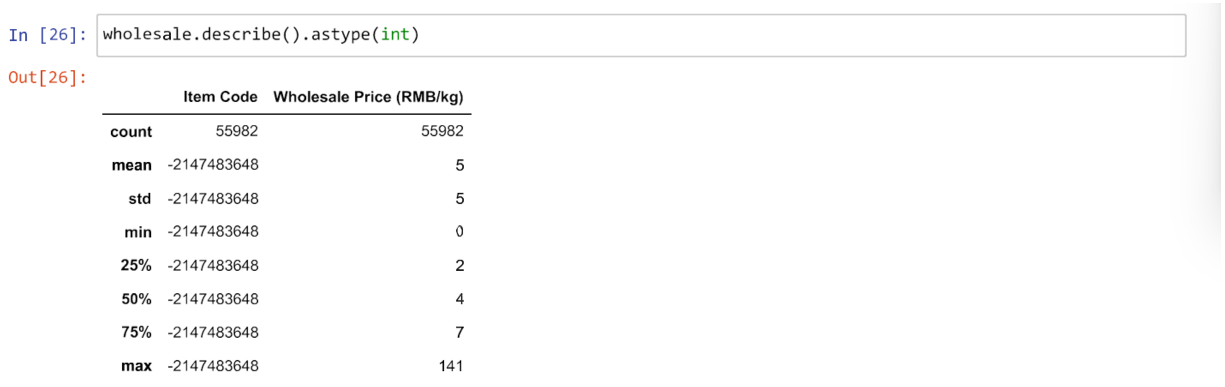


```
for i in wholesale.columns:
  print(i)
  print(wholesale[i].unique())
```

```
Date
['2020-07-01' '2020-07-02' '2020-07-03' ... '2023-06-28' '2023-06-29'
 '2023-06-30']
Item Code
[102900005115762 102900005115779 102900005115786 102900005115793
 102900005115823 102900005115908 102900005115946 102900005115960
 102900005115984 102900005116226 102900005116233 102900005116257
 102900005116509 102900005116530 102900005116547 102900005116714
 102900005116790 102900005116912 102900005116943 102900005117056
 102900005117209 102900005118817 102900005118824 102900005118831
 102900005119944 102900005119975 102900005123880 102900005125808
 102900005125815 102900011001219 102900011006948 102900011008522
 102900011009970 102900051000944 102900051004294 102900051010455
 102900011000328 102900011006689 102900011001813 102900011009444
 102900005116837 102900005115816 102900005116899 102900005115861
 106956146480203 106956146480197 102900011011546 102900011001806
 102900011001561 102900005116219 102900011000175 102900011007969
 102900011009246 102900011012994 102900011013274 102900005115885
 102900011001691 102900011008164 102900011010891 102900051000463
 102900051009336 102900005119968 102900011016909 102900005116905
 102900011016701 102900011007464 102900005115878 102900005115250
```



```
 102900011032213 102900011032244 102900011032114 102900011032220
 102900011032251 102900011032350 102900011032367 102900011031216
 102900011032176 102900011015384 102900011032343 102900011032589
 102900011032626 102900011032633 102900011032640 102900011032787
 102900011032619 102900011032732 102900011032848 102900011021675
 102900011033081 102900011033234 102900011033241 102900011032145
 102900011033562 102900011033586 102900011033531 102900011033173
 102900011033906 102900011033968 102900011034200 102900011034231
 102900011033920 102900011033937 102900011033944 102900011034217
 102900011034224 102900011033913 102900005116042 106931885000356
 102900011023648 106971563780002 102900011034316 102900011034323
 102900011034330 102900011034354 102900011026618 102900011029299
 102900011034262 102900011034026 102900011030929 106973990980123
 102900011034569 102900011034439 102900011033975 102900011033982
 102900011035078 102900011031858 102900011011058 102900011030905
 102900011033999 102900011035481 102900011035511 102900005115625
 102900011035771 102900011035788 102900011035764 102900011012482
 102900011035962 106930274620090 102900011036068 102900011034705
 102900011030400 102900011030417 102900011035740 102900011034538
 102900011023976 102900011036266 102900011032480 102900011036242
 102900011035849 106972776821582 102900011036686]
Wholesale Price (RMB/kg)
[ 3.88  6.72  3.19 ... 18.18 18.28 18.27]
```



```
In [26]: wholesale.describe().astype(int)
```

Out[26]:

	Item Code	Wholesale Price (RMB/kg)
count	55982	55982
mean	-2147483648	5
std	-2147483648	5
min	-2147483648	0
25%	-2147483648	2
50%	-2147483648	4
75%	-2147483648	7
max	-2147483648	141

In [27]:
```
loss_rate.head()
```

Out[27]:

	Item Code	Item Name	Loss Rate (%)
0	102900005115168	Niushou Shengcai	4.39
1	102900005115199	Sichuan Red Cedar	10.46
2	102900005115250	Xixia Black Mushroom (1)	10.80
3	102900005115625	Local Xiaomao Cabbage	0.18
4	102900005115748	White Caitai	8.78

In [28]:
```
loss_rate.dtypes
```

Out[28]:
```
Item Code          int64
Item Name         object
Loss Rate (%)    float64
dtype: object
```

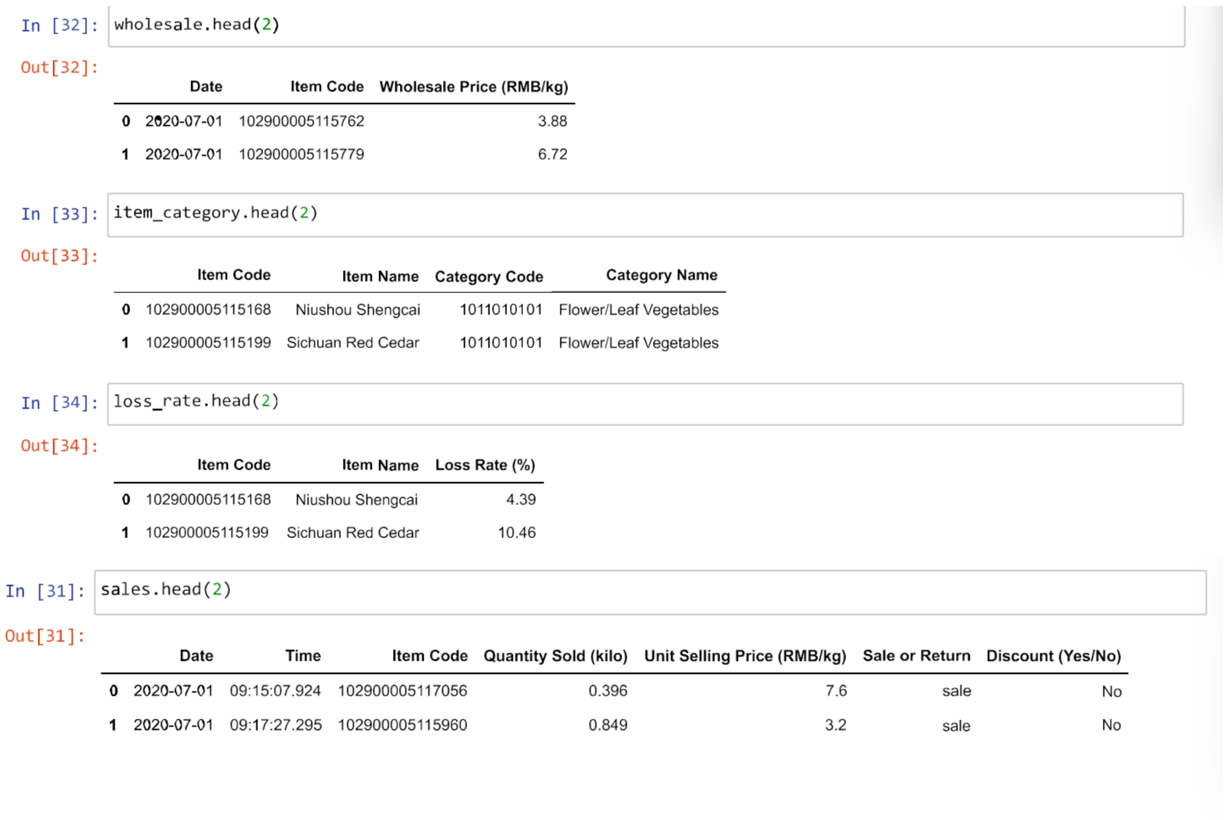
In [29]:
```
loss_rate.isnull().sum()
```

Out[29]:
```
Item Code        0
Item Name        0
Loss Rate (%)    0
dtype: int64
```

In [30]:
```
loss_rate.describe().astype(int)
```

Out[30]:

	Item Code	Loss Rate (%)
count	251	251
mean	-2147483648	9
std	-2147483648	5
min	-2147483648	0
25%	-2147483648	8
50%	-2147483648	9
75%	-2147483648	11
max	-2147483648	29



```
In [32]: wholesale.head(2)
```

Out[32]:

	Date	Item Code	Wholesale Price (RMB/kg)
0	2020-07-01	102900005115762	3.88
1	2020-07-01	102900005115779	6.72

```
In [33]: item_category.head(2)
```

Out[33]:

	Item Code	Item Name	Category Code	Category Name
0	102900005115168	Niushou Shengcai	1011010101	Flower/Leaf Vegetables
1	102900005115199	Sichuan Red Cedar	1011010101	Flower/Leaf Vegetables

```
In [34]: loss_rate.head(2)
```

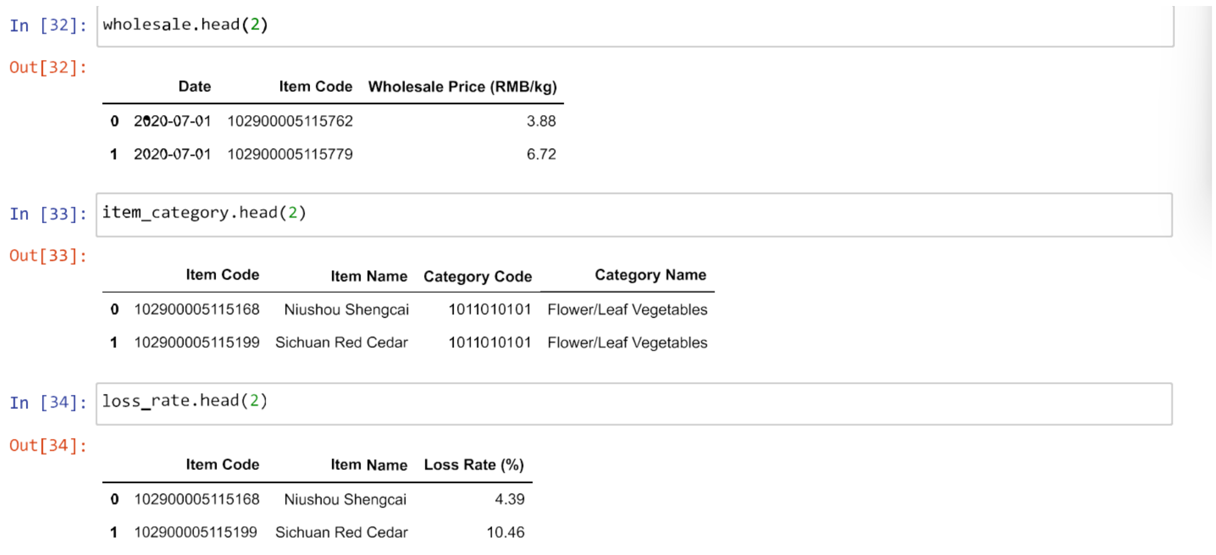
Out[34]:

	Item Code	Item Name	Loss Rate (%)
0	102900005115168	Niushou Shengcai	4.39
1	102900005115199	Sichuan Red Cedar	10.46

```
In [31]: sales.head(2)
```

Out[31]:

	Date	Time	Item Code	Quantity Sold (kilo)	Unit Selling Price (RMB/kg)	Sale or Return	Discount (Yes/No)
0	2020-07-01	09:15:07.924	102900005117056	0.396	7.6	sale	No
1	2020-07-01	09:17:27.295	102900005115960	0.849	3.2	sale	No



```
In [32]: wholesale.head(2)
```

Out[32]:

	Date	Item Code	Wholesale Price (RMB/kg)
0	2020-07-01	102900005115762	3.88
1	2020-07-01	102900005115779	6.72

```
In [33]: item_category.head(2)
```

Out[33]:

	Item Code	Item Name	Category Code	Category Name
0	102900005115168	Niushou Shengcai	1011010101	Flower/Leaf Vegetables
1	102900005115199	Sichuan Red Cedar	1011010101	Flower/Leaf Vegetables

```
In [34]: loss_rate.head(2)
```

Out[34]:

	Item Code	Item Name	Loss Rate (%)
0	102900005115168	Niushou Shengcai	4.39
1	102900005115199	Sichuan Red Cedar	10.46

In [35]:
```python
sales_wholesale_combine_data = pd.merge(sales,wholesale,how="left",on=["Item Code","Date"])
sales_wholesale_combine_data.head()
```

Out[35]:

	Date	Time	Item Code	Quantity Sold (kilo)	Unit Selling Price (RMB/kg)	Sale or Return	Discount (Yes/No)	Wholesale Price (RMB/kg)
0	2020-07-01	09:15:07.924	102900005117056	0.396	7.6	sale	No	4.32
1	2020-07-01	09:17:27.295	102900005115960	0.849	3.2	sale	No	2.10
2	2020-07-01	09:17:33.905	102900005117056	0.409	7.6	sale	No	4.32
3	2020-07-01	09:19:45.450	102900005115823	0.421	10.0	sale	No	7.03
4	2020-07-01	09:20:23.686	102900005115908	0.539	8.0	sale	No	4.60

In [36]:
```python
sales_wholesale_combine_data.isnull().sum()
```

Out[36]:
```
Date                           0
Time                           0
Item Code                      0
Quantity Sold (kilo)           0
Unit Selling Price (RMB/kg)    0
Sale or Return                 0
Discount (Yes/No)              0
Wholesale Price (RMB/kg)       0
dtype: int64
```

In [37]:
```python
sales_wholesale_category = pd.merge(sales_wholesale_combine_data,item_category,how="left",on="Item Code")
sales_wholesale_category.head()
```

Out[37]:

	Date	Time	Item Code	Quantity Sold (kilo)	Unit Selling Price (RMB/kg)	Sale or Return	Discount (Yes/No)	Wholesale Price (RMB/kg)	Item Name	Category Code	Category Na
0	2020-07-01	09:15:07.924	102900005117056	0.396	7.6	sale	No	4.32	Paopaojiao (Jingpin)	1011010504	Capsic
1	2020-07-01	09:17:27.295	102900005115960	0.849	3.2	sale	No	2.10	Chinese Cabbage	1011010101	Flower/Leaf Vegetat
2	2020-07-01	09:17:33.905	102900005117056	0.409	7.6	sale	No	4.32	Paopaojiao (Jingpin)	1011010504	Capsic
3	2020-07-01	09:19:45.450	102900005115823	0.421	10.0	sale	No	7.03	Shanghaiqing	1011010101	Flower/Leaf Vegetat
4	2020-07-01	09:20:23.686	102900005115908	0.539	8.0	sale	No	4.60	Caixin	1011010101	Flower/Leaf Vegetat

```
In [38]: final_data = pd.merge(sales_wholesale_category,loss_rate,how="left",on=["Item Code","Item Name"])
         final_data.head()
```

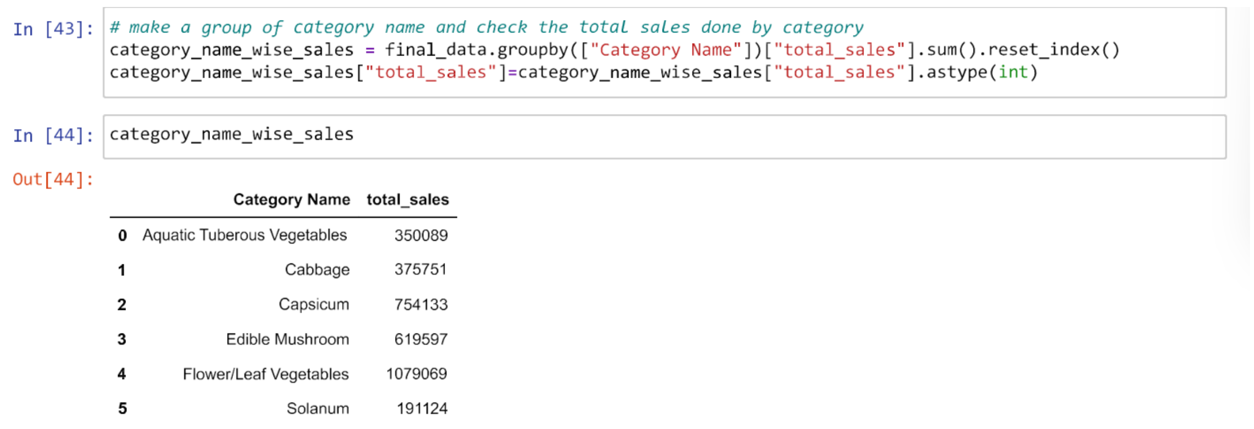
Out[38]:

	Date	Time	Item Code	Quantity Sold (kilo)	Unit Selling Price (RMB/kg)	Sale or Return	Discount (Yes/No)	Wholesale Price (RMB/kg)	Item Name	Category Code	Category Na
0	2020-07-01	09:15:07.924	102900005117056	0.396	7.6	sale	No	4.32	Paopaojiao (Jingpin)	1011010504	Capsic
1	2020-07-01	09:17:27.295	102900005115960	0.849	3.2	sale	No	2.10	Chinese Cabbage	1011010101	Flower/Leaf Vegetat
2	2020-07-01	09:17:33.905	102900005117056	0.409	7.6	sale	No	4.32	Paopaojiao (Jingpin)	1011010504	Capsic
3	2020-07-01	09:19:45.450	102900005115823	0.421	10.0	sale	No	7.03	Shanghaiqing	1011010101	Flower/Leaf Vegetat
4	2020-07-01	09:20:23.686	102900005115908	0.539	8.0	sale	No	4.60	Caixin	1011010101	Flower/Leaf Vegetat

```
In [39]: final_data["total_sales"]=final_data["Quantity Sold (kilo)"]*final_data["Unit Selling Price (RMB/kg)"]
```

```
In [41]: final_data.isnull().sum()

Out[41]: Date                             0
         Time                             0
         Item Code                        0
         Quantity Sold (kilo)             0
         Unit Selling Price (RMB/kg)      0
         Sale or Return                   0
         Discount (Yes/No)                0
         Wholesale Price (RMB/kg)         0
         Item Name                        0
         Category Code                    0
         Category Name                    0
         Loss Rate (%)                    0
         total_sales                      0
         dtype: int64
```

```
In [42]: final_data["Item Name"].nunique()
         final_data["Category Name"].nunique()

Out[42]: 6
```



```
In [43]: # make a group of category name and check the total sales done by category
         category_name_wise_sales = final_data.groupby(["Category Name"])["total_sales"].sum().reset_index()
         category_name_wise_sales["total_sales"]=category_name_wise_sales["total_sales"].astype(int)

In [44]: category_name_wise_sales

Out[44]:
```

	Category Name	total_sales
0	Aquatic Tuberous Vegetables	350089
1	Cabbage	375751
2	Capsicum	754133
3	Edible Mushroom	619597
4	Flower/Leaf Vegetables	1079069
5	Solanum	191124

Dataset

- https://drive.google.com/file/d/15cbdh2go--8f-T7f5qXDaql_2Zxkbwaq/view?usp=drive_link
- https://drive.google.com/file/d/1MBoFohSMjIOf0iRiRn-iWEvrfIZw58y2/view?usp=drive_link
- https://drive.google.com/file/d/1rGq11aho5Cs5YHsp46H2iR3Uxw43QG2A/view?usp=drive_link
- https://drive.google.com/file/d/1uYmMCmtE0FeZVRsu6Kt8LER23TpcptW-/view?usp=drive_link

Conclusion

In conclusion, the Sales Analysis Project underscores the transformative potential of data-driven approaches in shaping the future of sales and marketing. By embracing Python as a catalyst for innovation, businesses can navigate uncertainty, capitalize on opportunities, and chart a course toward sustained success in an ever-changing landscape.