BIG DATA TOOLS FOR MANAGERS

Unit-4 & 5 : Introduction to Python Pandas

python

by
Ankit Velani
Adjunct Faculty, Dept. of MBA,
Siddaganga Institute of Technology,Tumkur

Comparison with Python Data structure

Lists	Tuple	Set	Dictionary
Lists are mutable	Tuples are immutable.	Sets are mutable	Dictionary are mutable
Lists are enclosed within square braces. [ ]	Tuples are enclosed within parenthesis. ( )	Sets are enclosed in curly brackets. { }	Dictionaries are enclosed in curly brackets with key-value pairs. { key : value }
List element can be accessed using index/ range of index	Tuple element can be accessed using index/range of index	Have use iteration like for, while loop to access element	Dictionary element can be accessible using its key
len() function used to get length/size of list	len() function used to get length/size of Tuple	len() function used to get length/size of set	len() function used to get length/size of dictionary
Easy to combine two or more lists with + (plus) operator	Easy to combine two or more Tuple with + (plus) operator	union to be used for combining two set	update function used to combine two dictionary

Exercise

Create a List with given element [10,20,30,40,50,60,70,80,90,100]

Write a python code for :

1. Create a List
2. Print element using print()
3. Print element using iteration (For loop)
4. Multiply list elements with number 2
5. Display first element of list
6. Display last element of list
7. Display first 3 elements of list
8. Display last 3 elements of list

Exercise

# Create a dictionary for Employee data

employee_Name : John

employee_City : Bangalore

employee_Mobile: 9876512345

employee_Email : john@gmail.com

Write a python code for :

1. Create a employee dictionary
2. Display all the key present in dictionary
3. Display all the value present in dictionary
4. Print Dictionary element
5. Access dictionary element using employee_Name
6. Access dictionary element using employee_Email

About Pandas

- Pandas is an open-source Python library providing high-performance data manipulation and analysis tool using its powerful data structure.

- The name panda is derived from the word Panel Data.

- In 2008 panda library introduced for high performance, flexible tool for analysis of data.

- Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Key Features of Pandas

- Fast and efficient DataFrame object with default and customized indexing.
- Tools for loading data into in-memory data objects from different file formats.
- Data alignment and integrated handling of missing data.
- Label-based slicing, indexing and sub-setting of large data sets.
- Columns from a data structure can be deleted or inserted.
- Group by data for aggregation and transformations.
- High performance merging and joining of data.
- Time Series functionality.

Where to get pandas library?

Where to get pandas library?

- Download pandas library from an internet using pip package manager

pip install pandas

how to import pandas library?

import statement used to import the pandas or any other libraries in Python.

```
import pandas
```

or

```
import pandas as pd
```

or

```
import pandas as xyz
```

- **as** keyword helps to create **alias name** of the package.
- This helpful when package name is too large.

Data Frames in R

- Creation of data frame

```
my_data <- data.frame(
          name    = c(“A”,”B”,”C”,”D”),
          age       = c(40,45,70,60),
          gender = c(“F”,”M”,”F”,”M”)
)
```

name, age, gender are the vectors

Output:

Name	Age	gender
A	40	F
B	45	M
C	70	F
D	60	M

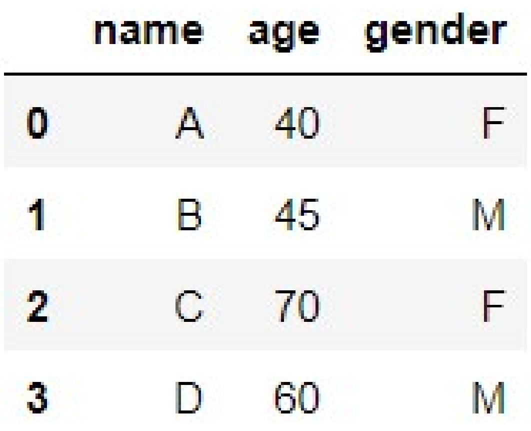
Data Frames in Python

- Creation of data frame in Python

```
import pandas as pd

my_data = pd.DataFrame(
   {
      "name"     :["A","B","C","D"],
      "age"      : [40,45,70,60],
      "gender"  :["F","M","F","M"]
   }
)
```

Output:



	name	age	gender
0	A	40	F
1	B	45	M
2	C	70	F
3	D	60	M

name, age, gender are the dictionary key

Import files using Pandas

- Pandas library has provided different methods for loading datasets with many different formats onto DataFrame.

- *** Once data loaded into memory pandas creates Pandas DataFrame automatically for easy to manipulate data.

- File format pandas support:

  csv, json, excel, table, fwf(fixed with format)

- Read function looks like:

  read_csv()

  read_excel()

  read_table()

Import csv file using Pandas

- Pandas has read_csv functions to read the csv file.

Example:

# Download dataset and save in C:/dataset location

https://drive.google.com/file/d/1EPQhl0wVCZnNP1vx7BE43phOUljzZXGD/view?usp=sharing

Import csv file using Pandas

- Pandas has read_csv functions to read the csv file.

<u>Example:</u>

```
import pandas as pd
pd.read_csv("C:/dataset/VEHICLE_PARK.csv")
```

or

```
import pandas
pandas.read_csv("C:/dataset/VEHICLE_PARK.csv")
```

Import csv file using Pandas

- Pandas has **read_csv()** functions to read the csv file.
- After importing csv file, store the data in any variable.

Example:

```
import pandas as pd
data = pd.read_csv("C:/dataset/VEHICLE_PARK.csv")
```

- Now, data variable having entire csv file data in form of rows & cols.

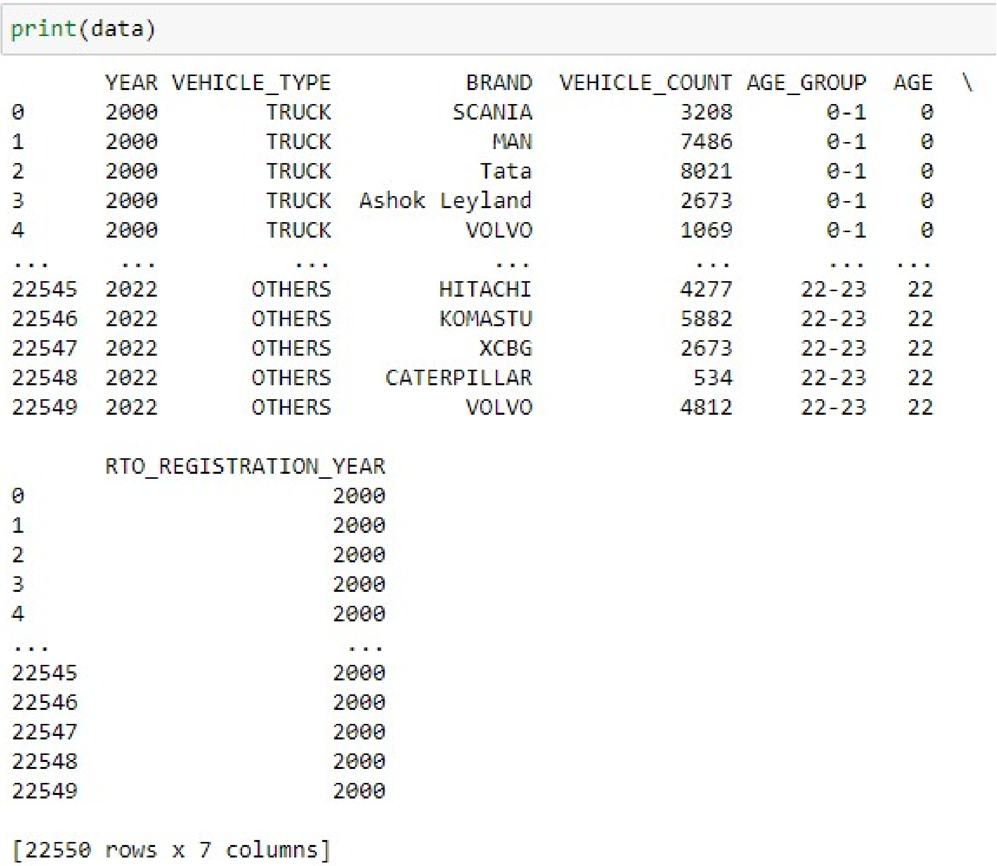
Python Pandas

print() function to display the data frame values

Example:

print(data)

Output:



```
print(data)
```

```
        YEAR VEHICLE_TYPE          BRAND  VEHICLE_COUNT AGE_GROUP  AGE  \
0       2000        TRUCK         SCANIA           3208       0-1    0
1       2000        TRUCK            MAN           7486       0-1    0
2       2000        TRUCK           Tata           8021       0-1    0
3       2000        TRUCK  Ashok Leyland           2673       0-1    0
4       2000        TRUCK          VOLVO           1069       0-1    0
...      ...          ...            ...            ...       ...  ...
22545   2022       OTHERS        HITACHI           4277     22-23   22
22546   2022       OTHERS        KOMASTU           5882     22-23   22
22547   2022       OTHERS           XCBG           2673     22-23   22
22548   2022       OTHERS    CATERPILLAR            534     22-23   22
22549   2022       OTHERS          VOLVO           4812     22-23   22

        RTO_REGISTRATION_YEAR
0                        2000
1                        2000
2                        2000
3                        2000
4                        2000
...                       ...
22545                    2000
22546                    2000
22547                    2000
22548                    2000
22549                    2000

[22550 rows x 7 columns]
```

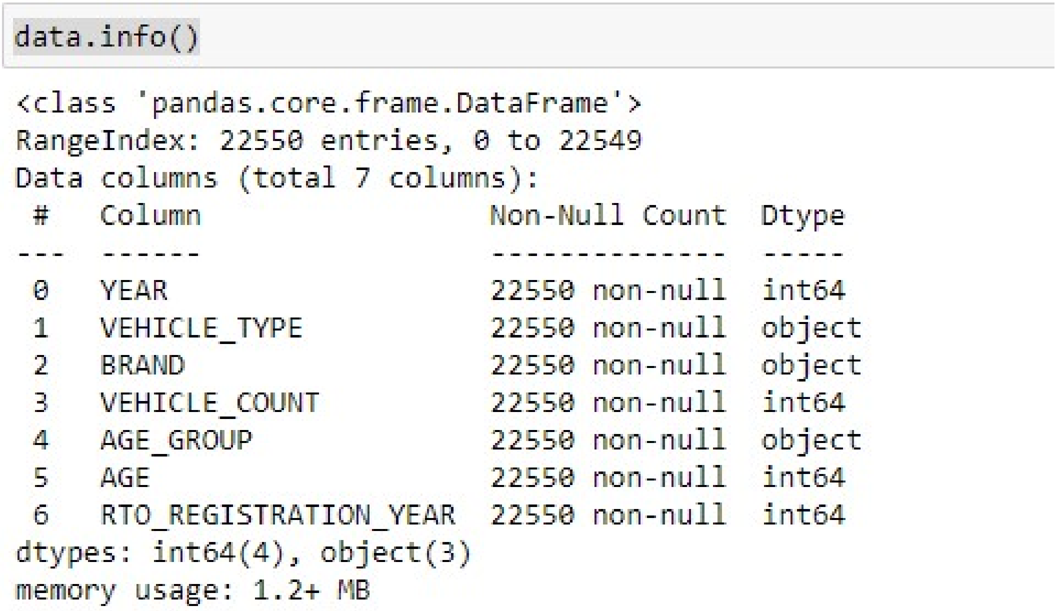
Python Pandas

info() Function to get the structure of Pandas DataFrame

Example:

data.info()

Output:



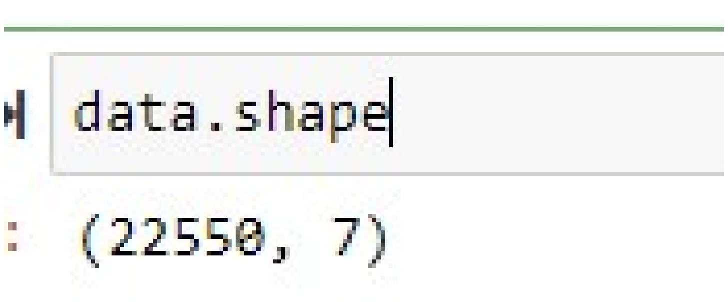
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22550 entries, 0 to 22549
Data columns (total 7 columns):
 #   Column                 Non-Null Count  Dtype 
---  ------                 --------------  ----- 
 0   YEAR                   22550 non-null  int64 
 1   VEHICLE_TYPE           22550 non-null  object
 2   BRAND                  22550 non-null  object
 3   VEHICLE_COUNT          22550 non-null  int64 
 4   AGE_GROUP              22550 non-null  object
 5   AGE                    22550 non-null  int64 
 6   RTO_REGISTRATION_YEAR  22550 non-null  int64 
dtypes: int64(4), object(3)
memory usage: 1.2+ MB
```

Python Pandas

shape is properties to get the no of rows & columns of DataFrame

Output:



```
data.shape
```

```
: (22550, 7)
```

Example:

data.shape

Python Pandas

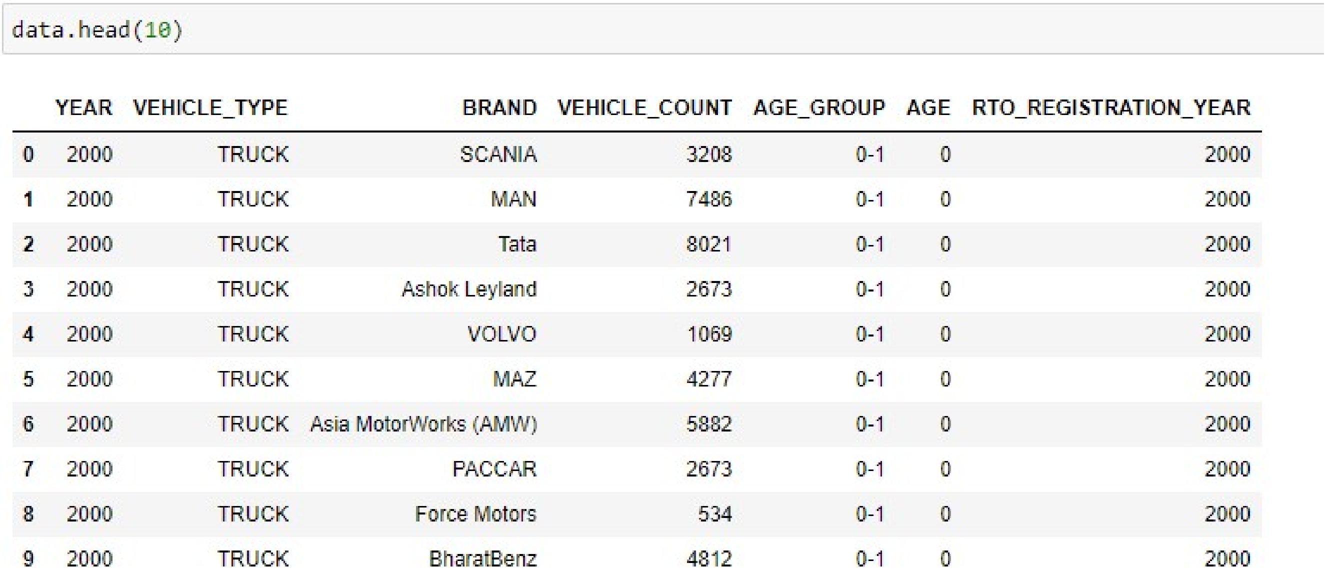
head(n) & **tail(n)** are the functions to display top and bottom rows from the pandas DataFrame.

Output:

Example:

data.head(10)

data.tail(10)



```
data.head(10)
```

	YEAR	VEHICLE_TYPE	BRAND	VEHICLE_COUNT	AGE_GROUP	AGE	RTO_REGISTRATION_YEAR
0	2000	TRUCK	SCANIA	3208	0-1	0	2000
1	2000	TRUCK	MAN	7486	0-1	0	2000
2	2000	TRUCK	Tata	8021	0-1	0	2000
3	2000	TRUCK	Ashok Leyland	2673	0-1	0	2000
4	2000	TRUCK	VOLVO	1069	0-1	0	2000
5	2000	TRUCK	MAZ	4277	0-1	0	2000
6	2000	TRUCK	Asia MotorWorks (AMW)	5882	0-1	0	2000
7	2000	TRUCK	PACCAR	2673	0-1	0	2000
8	2000	TRUCK	Force Motors	534	0-1	0	2000
9	2000	TRUCK	BharatBenz	4812	0-1	0	2000

Python Pandas

Column selection

- **[‘col-name’]** – for selecting single column
- **[[‘col-name-1’,’col-name-2’,…..’col-name-n’]]** - for selecting multiple columns

<u>Example:</u>

```
data['YEAR']    #One column

data[['YEAR', 'AGE_GROUP']]  # Multiple columns
```

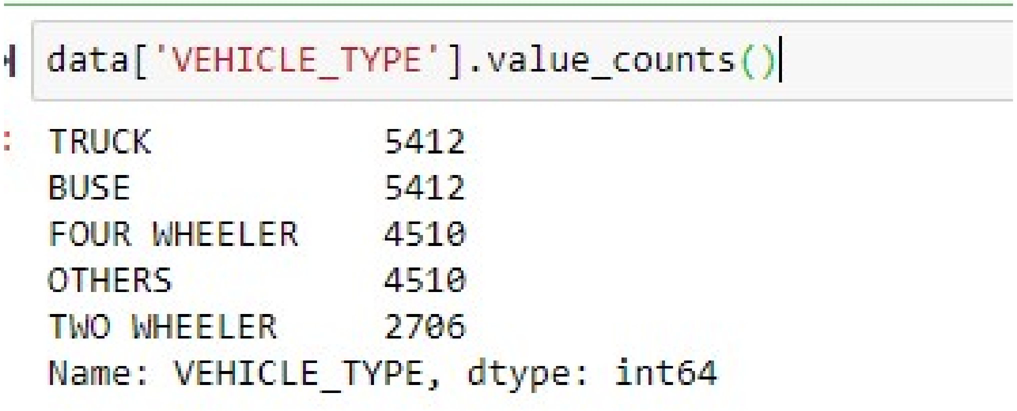
Python Pandas

value_counts(normalize=False) function Gives frequency of each unique value in a column

normalize=True will calculate the percentage of total frequency.

Example:

data['VEHICLE_TYPE'].value_counts()



```
data['VEHICLE_TYPE'].value_counts()
```

```
TRUCK           5412
BUSE            5412
FOUR WHEELER    4510
OTHERS          4510
TWO WHEELER     2706
Name: VEHICLE_TYPE, dtype: int64
```

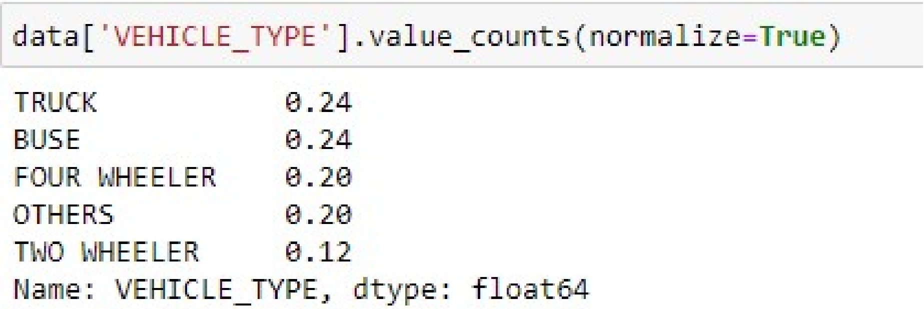
Python Pandas

value_counts(normalize=False) function Gives frequency of each unique value in a column

normalize=True will calculate the percentage of total frequency.

Example:

data['VEHICLE_TYPE']. value_counts(normalize=True)



```
data['VEHICLE_TYPE'].value_counts(normalize=True)
```

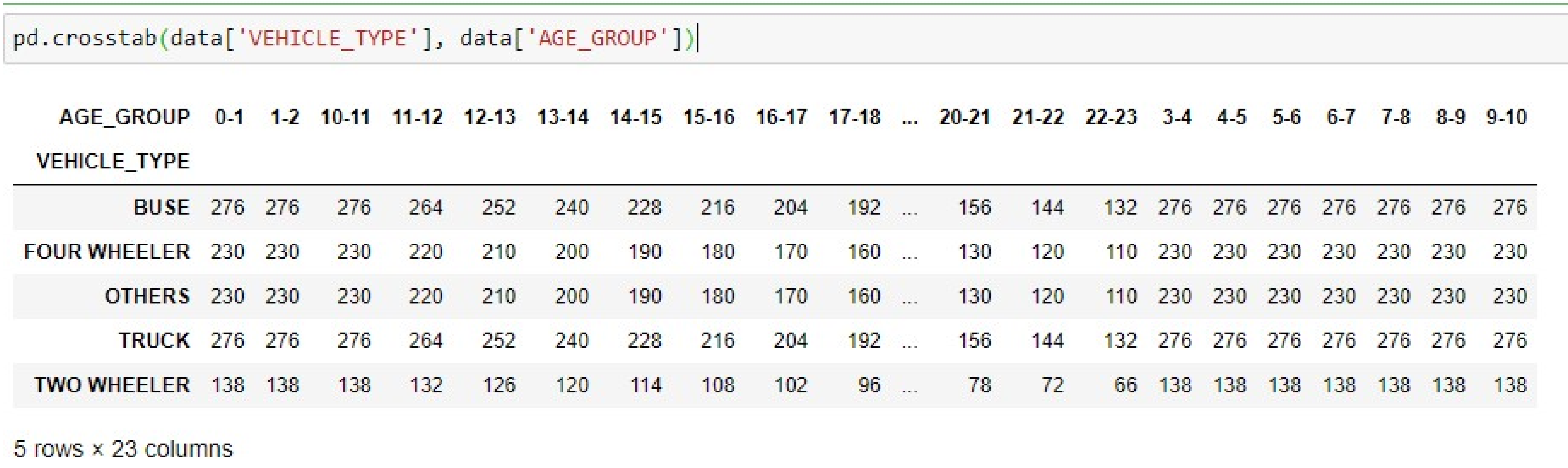
```
TRUCK           0.24
BUSE            0.24
FOUR WHEELER    0.20
OTHERS          0.20
TWO WHEELER     0.12
Name: VEHICLE_TYPE, dtype: float64
```

Python Pandas

crosstab() function Gives frequency of two columns.

Example:

pd.crosstab(data['VEHICLE_TYPE'], data['AGE_GROUP’])



```
pd.crosstab(data['VEHICLE_TYPE'], data['AGE_GROUP'])
```

AGE_GROUP	0-1	1-2	10-11	11-12	12-13	13-14	14-15	15-16	16-17	17-18	...	20-21	21-22	22-23	3-4	4-5	5-6	6-7	7-8	8-9	9-10
VEHICLE_TYPE																					
BUSE	276	276	276	264	252	240	228	216	204	192	...	156	144	132	276	276	276	276	276	276	276
FOUR WHEELER	230	230	230	220	210	200	190	180	170	160	...	130	120	110	230	230	230	230	230	230	230
OTHERS	230	230	230	220	210	200	190	180	170	160	...	130	120	110	230	230	230	230	230	230	230
TRUCK	276	276	276	264	252	240	228	216	204	192	...	156	144	132	276	276	276	276	276	276	276
TWO WHEELER	138	138	138	132	126	120	114	108	102	96	...	78	72	66	138	138	138	138	138	138	138

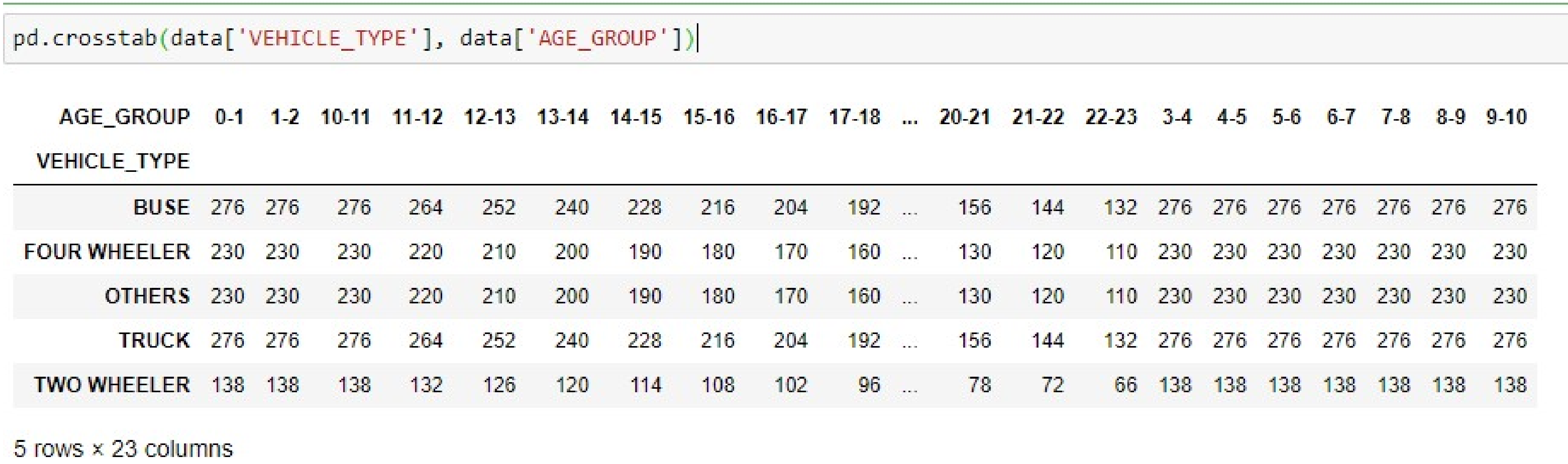
5 rows × 23 columns

Python Pandas

crosstab() function Gives frequency of two columns.

Example:

pd.crosstab(data['VEHICLE_TYPE'], data['AGE_GROUP’])



```
pd.crosstab(data['VEHICLE_TYPE'], data['AGE_GROUP'])
```

AGE_GROUP	0-1	1-2	10-11	11-12	12-13	13-14	14-15	15-16	16-17	17-18	...	20-21	21-22	22-23	3-4	4-5	5-6	6-7	7-8	8-9	9-10
VEHICLE_TYPE																					
BUSE	276	276	276	264	252	240	228	216	204	192	...	156	144	132	276	276	276	276	276	276	276
FOUR WHEELER	230	230	230	220	210	200	190	180	170	160	...	130	120	110	230	230	230	230	230	230	230
OTHERS	230	230	230	220	210	200	190	180	170	160	...	130	120	110	230	230	230	230	230	230	230
TRUCK	276	276	276	264	252	240	228	216	204	192	...	156	144	132	276	276	276	276	276	276	276
TWO WHEELER	138	138	138	132	126	120	114	108	102	96	...	78	72	66	138	138	138	138	138	138	138

5 rows × 23 columns

Python Pandas

sort_values(col-name, ascending=False) function sort the DataFrame based on the specified columns.

By default, sorting is done in ascending order.

Example (Single Column):

```
data.sort_values(["BRAND"], ascending=[False])
```

or

```
data.sort_values("BRAND", ascending=False)
```

Python Pandas

sort_values(col-name, ascending=False) function sort the DataFrame based on the specified columns.

By default, sorting is done in ascending order.

Example (Multi Columns):

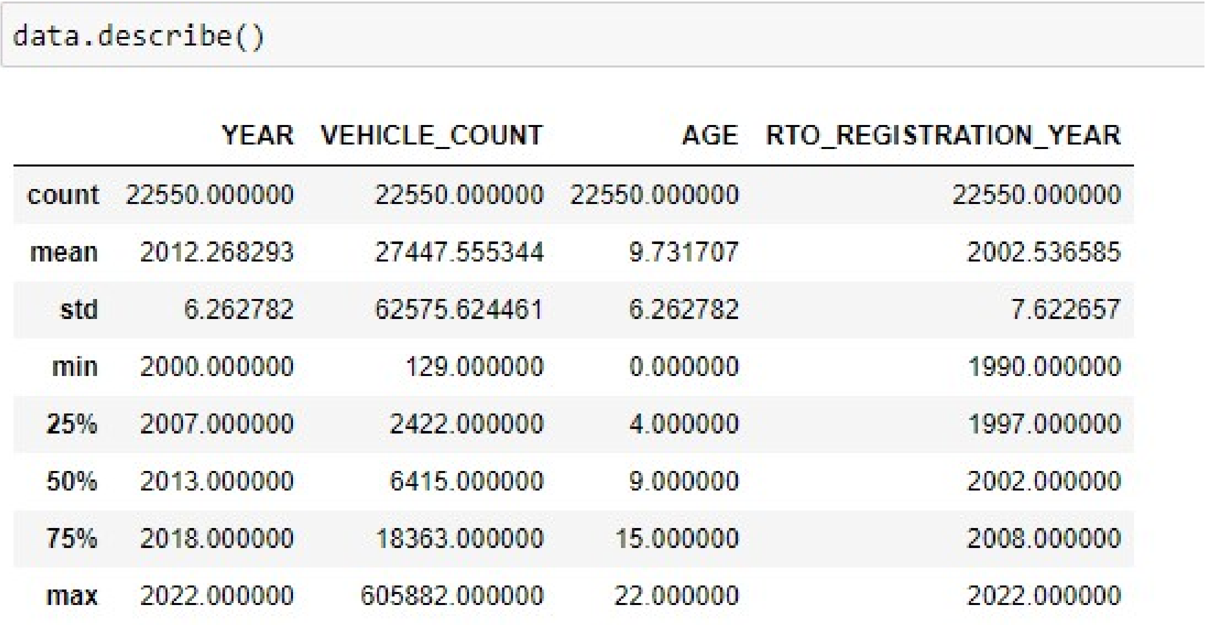
```
data.sort_values(
        ['BRAND', 'VEHICLE_COUNT'],
        ascending=[True, False]
   )
```

Python Pandas

describe() function to display quick summary of the DataFrame. By default, it gives summary for numeric data only.

<u>Example</u>

data.describe()



```
data.describe()
```

	YEAR	VEHICLE_COUNT	AGE	RTO_REGISTRATION_YEAR
count	22550.000000	22550.000000	22550.000000	22550.000000
mean	2012.268293	27447.555344	9.731707	2002.536585
std	6.262782	62575.624461	6.262782	7.622657
min	2000.000000	129.000000	0.000000	1990.000000
25%	2007.000000	2422.000000	4.000000	1997.000000
50%	2013.000000	6415.000000	9.000000	2002.000000
75%	2018.000000	18363.000000	15.000000	2008.000000
max	2022.000000	605882.000000	22.000000	2022.000000

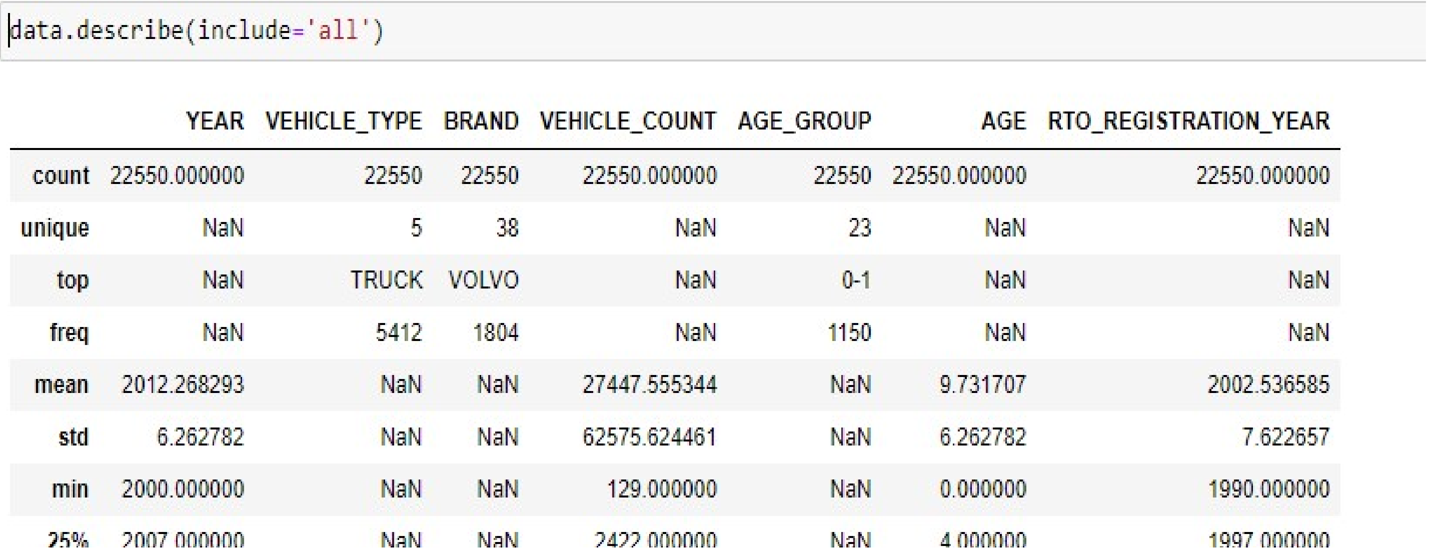
Python Pandas

describe() function to display quick summary of the DataFrame. By default, it gives summary for numeric data only.

describe(include=‘all’) function with include=‘all’ gives summary for all the columns available in DataFrame

Example

data.describe(include='all')



```
data.describe(include='all')
```

```
                YEAR VEHICLE_TYPE  BRAND VEHICLE_COUNT AGE_GROUP           AGE RTO_REGISTRATION_YEAR
count   22550.000000        22550  22550  22550.000000     22550  22550.000000          22550.000000
unique           NaN            5     38           NaN        23           NaN                   NaN
top              NaN        TRUCK  VOLVO           NaN       0-1           NaN                   NaN
freq             NaN         5412   1804           NaN      1150           NaN                   NaN
mean     2012.268293          NaN    NaN  27447.555344       NaN      9.731707           2002.536585
std         6.262782          NaN    NaN  62575.624461       NaN      6.262782              7.622657
min      2000.000000          NaN    NaN    129.000000       NaN      0.000000           1990.000000
25%      2007.000000          NaN    NaN   2422.000000       NaN      4.000000           1997.000000
```

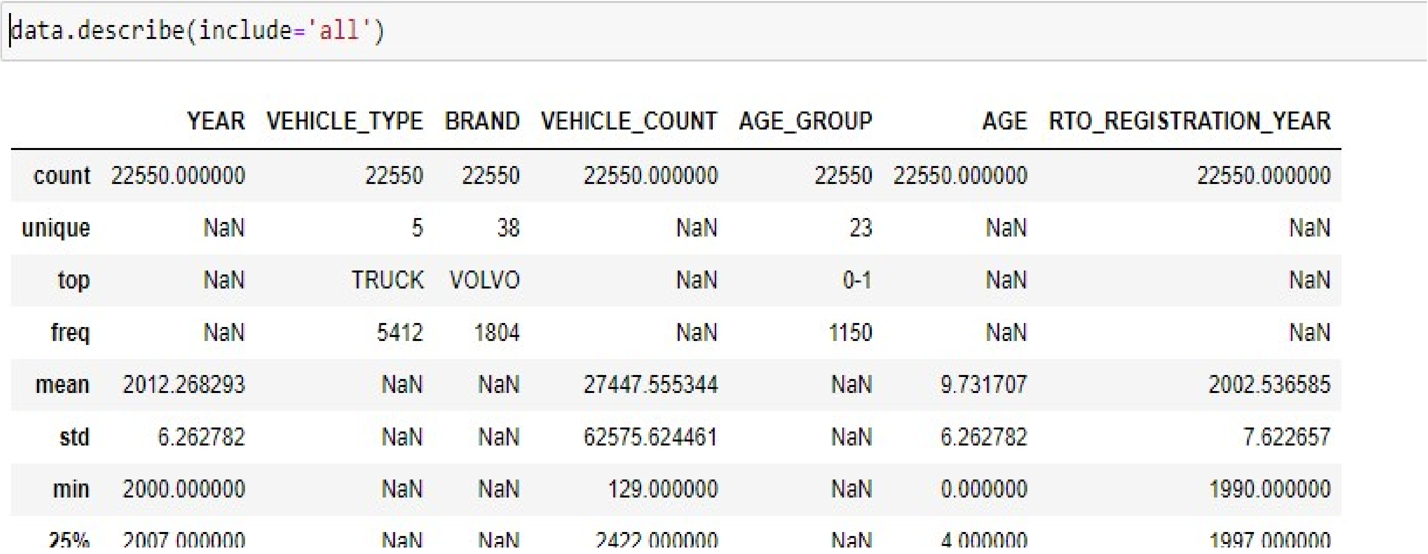
Python Pandas

describe() function to display quick summary of the DataFrame. By default, it gives summary for numeric data only.

describe(include=‘all’) function with include=‘all’ gives summary for all the columns available in DataFrame

Example

data.describe(include='all')



```
data.describe(include='all')
```

	YEAR	VEHICLE_TYPE	BRAND	VEHICLE_COUNT	AGE_GROUP	AGE	RTO_REGISTRATION_YEAR
count	22550.000000	22550	22550	22550.000000	22550	22550.000000	22550.000000
unique	NaN	5	38	NaN	23	NaN	NaN
top	NaN	TRUCK	VOLVO	NaN	0-1	NaN	NaN
freq	NaN	5412	1804	NaN	1150	NaN	NaN
mean	2012.268293	NaN	NaN	27447.555344	NaN	9.731707	2002.536585
std	6.262782	NaN	NaN	62575.624461	NaN	6.262782	7.622657
min	2000.000000	NaN	NaN	129.000000	NaN	0.000000	1990.000000
25%	2007.000000	NaN	NaN	2422.000000	NaN	4.000000	1997.000000

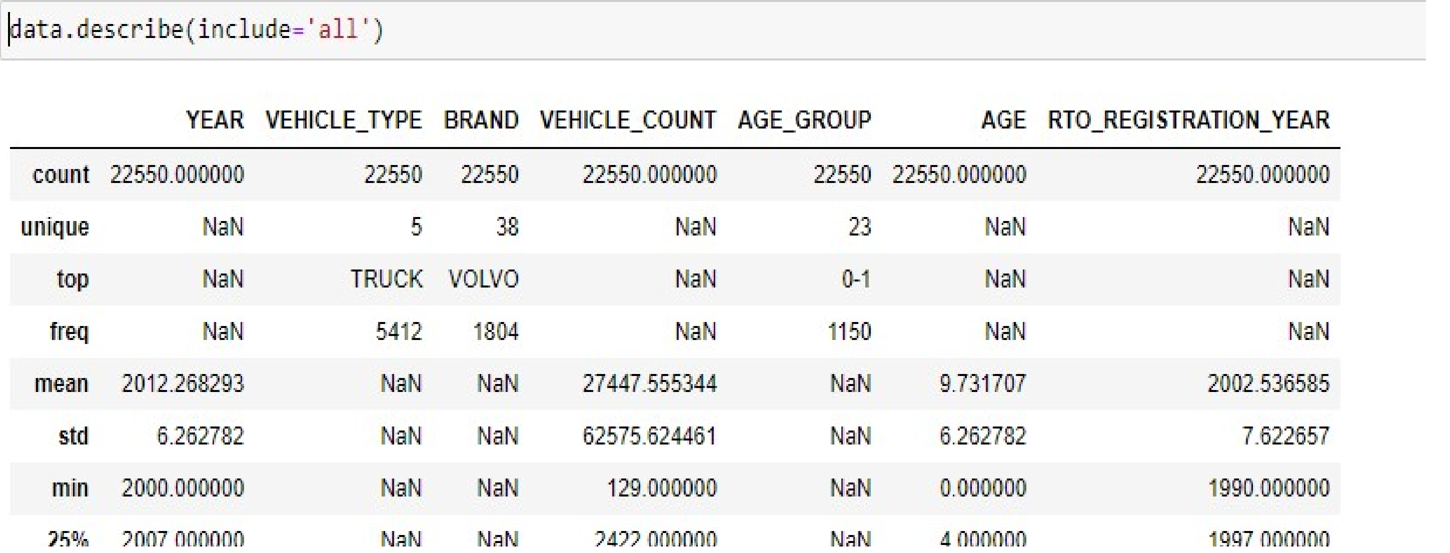
Python Pandas

describe() function to display quick summary of the DataFrame. By default, it gives summary for numeric data only.

describe(include=‘all’) function with include=‘all’ gives summary for all the columns available in DataFrame

Example

data.describe(include='all')



```
data.describe(include='all')
```

```
               YEAR  VEHICLE_TYPE  BRAND  VEHICLE_COUNT  AGE_GROUP           AGE  RTO_REGISTRATION_YEAR
count  22550.000000         22550  22550   22550.000000      22550  22550.000000           22550.000000
unique          NaN             5     38            NaN         23           NaN                    NaN
top             NaN         TRUCK  VOLVO            NaN        0-1           NaN                    NaN
freq            NaN          5412   1804            NaN       1150           NaN                    NaN
mean    2012.268293           NaN    NaN   27447.555344        NaN      9.731707            2002.536585
std        6.262782           NaN    NaN   62575.624461        NaN      6.262782               7.622657
min     2000.000000           NaN    NaN     129.000000        NaN      0.000000            1990.000000
25%     2007.000000           NaN    NaN    2422.000000        NaN      4.000000            1997.000000
```

Python Pandas

Aggregate functions like count, sum, min, max, mean…etc can be applied on DataFrame.

<u>Example</u>

```
#Single Column to Count number of elements present
data['YEAR'].count()
```

```
#Multiple Cols, Count number of elements present
data[['YEAR', 'AGE']].count()
```

```
#all the columns
data.count()
```

Python Pandas

Aggregate functions like count, sum, min, max, mean…etc can be applied on DataFrame.

Example

```
#Single Column, check Avg values for Age column
data['AGE'].mean()

#Multiple Cols, check Avg values for Age, Vehicle Count columns
data[['AGE', 'VEHICLE_COUNT']].mean()

#all the columns (categorical values get excluded)
data.mean()
```

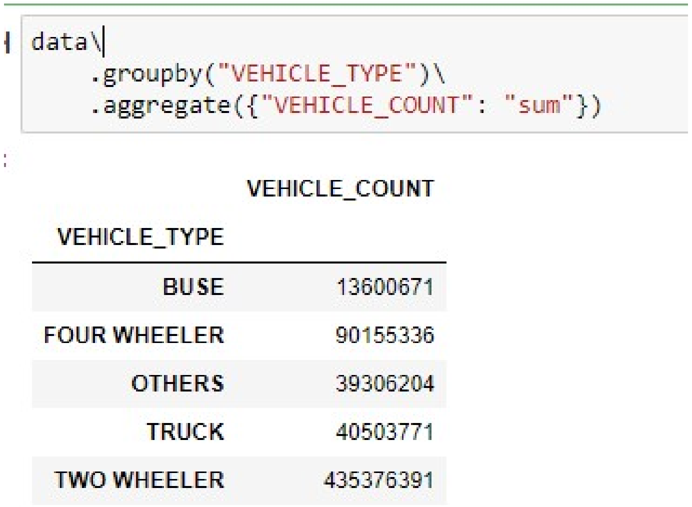
Python Pandas

groupby() functions to group the records based on specified columns and then apply for aggregation (max, min, sum, mean…etc)

Example

#Single Column, get the total number of vehicles by VEHICLE_TYPE

```
data\
    .groupby("VEHICLE_TYPE")\
    .aggregate({"VEHICLE_COUNT": "sum"})
```



```
data\
    .groupby("VEHICLE_TYPE")\
    .aggregate({"VEHICLE_COUNT": "sum"})
```

	VEHICLE_COUNT
VEHICLE_TYPE	
BUSE	13600671
FOUR WHEELER	90155336
OTHERS	39306204
TRUCK	40503771
TWO WHEELER	435376391

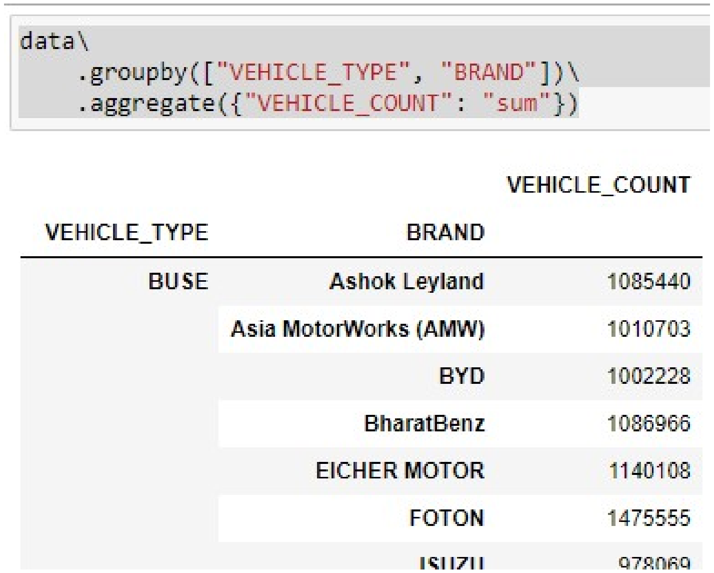
Python Pandas

groupby() functions to group the records based on specified columns and then apply for aggregation (max, min, sum, mean…etc)

Example

#Multi Column, get the total number of vehicles by VEHICLE_TYPE, BRAND

```
data\
    .groupby(["VEHICLE_TYPE", "BRAND"])\
    .aggregate({"VEHICLE_COUNT": "sum"})
```



```
data\
    .groupby(["VEHICLE_TYPE", "BRAND"])\
    .aggregate({"VEHICLE_COUNT": "sum"})
```

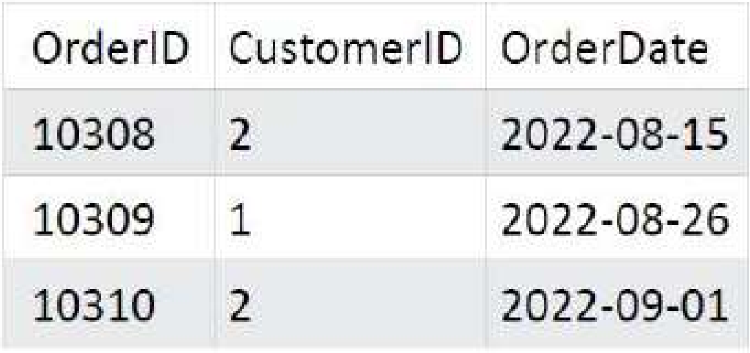
		VEHICLE_COUNT
VEHICLE_TYPE	BRAND	
BUSE	Ashok Leyland	1085440
	Asia MotorWorks (AMW)	1010703
	BYD	1002228
	BharatBenz	1086966
	EICHER MOTOR	1140108
	FOTON	1475555
	ISUZU	978069

Python Pandas

merge() functions to join multiple DataFrame based on the column columns and nature of join.

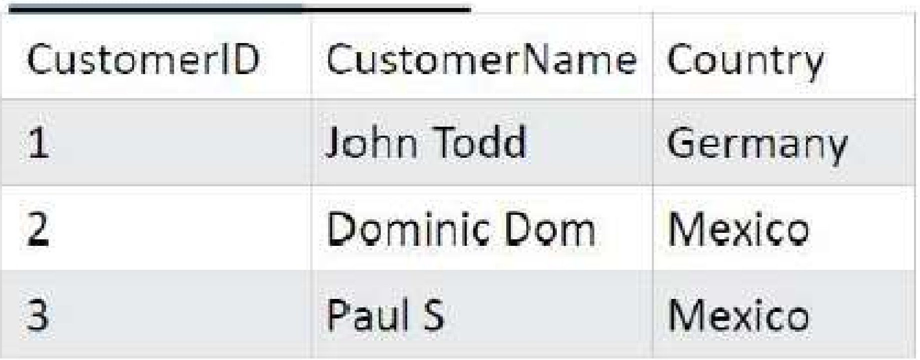
Before doing join, first create two sample dataframe.

DataFrame : Orders



OrderID	CustomerID	OrderDate
10308	2	2022-08-15
10309	1	2022-08-26
10310	2	2022-09-01

DataFrame : Customers



CustomerID	CustomerName	Country
1	John Todd	Germany
2	Dominic Dom	Mexico
3	Paul S	Mexico

Python Pandas

```
# Step-1 Create dictionary
orders_dictionary ={
    "OrderID"         : [10308, 10309, 10310],
    "CustomerID"   : [2,1,2],
    "OrderDate“      : ["2022-08-15", "2022-08-26", "2022-09-01"]
}
```

Python Pandas

```
# Step-1 Create dictionary
orders_dictionary ={
    "OrderID"          : [10308, 10309, 10310],
    "CustomerID"    : [2,1,2],
    "OrderDate“       : ["2022-08-15", "2022-08-26", "2022-09-01"]
}

# Step-2 Create Pandas DataFrame
Orders = pd.DataFrame(orders_dictionary)
```

Python Pandas

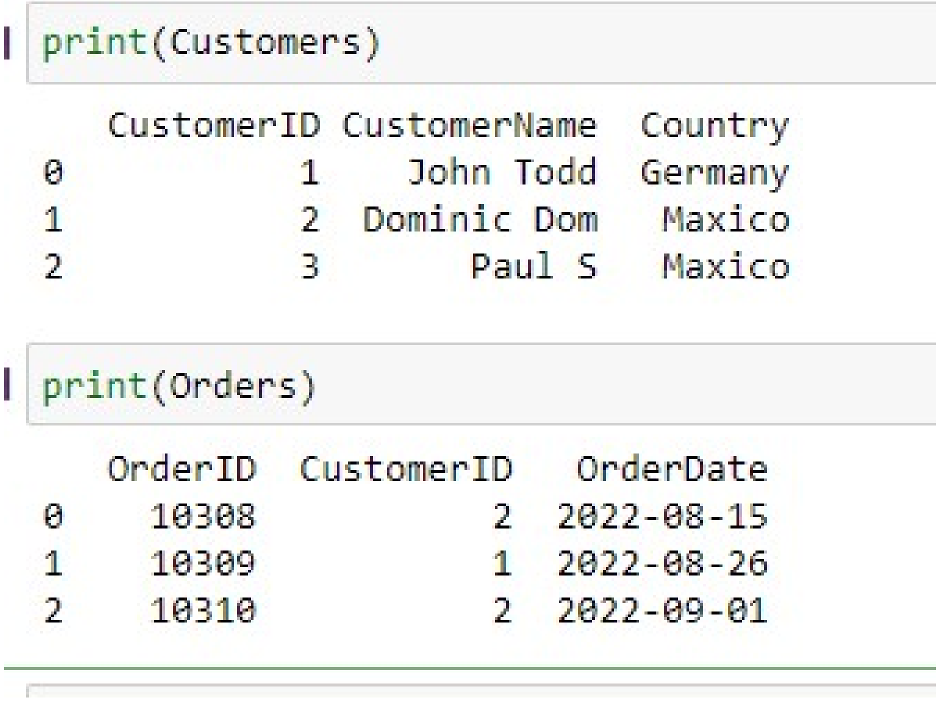
```
# Step-1 Create dictionary
customer_dictionary = {
    "CustomerID"    : [1,2,3],
    "CustomerName" : ["John Todd", "Dominic Dom", "Paul S"],
    "Country"       : ["Germany", "Maxico", "Maxico"]
}

# Step-2 Create Pandas DataFrame
Customers = pd.DataFrame(customer_dictionary)
```

Python Pandas

# Step-3 Display the Orders & Customers DataFrame

```
print(Customers)
print(Orders)
```



```
print(Customers)
```

```
   CustomerID CustomerName  Country
0           1    John Todd  Germany
1           2  Dominic Dom   Maxico
2           3       Paul S   Maxico
```

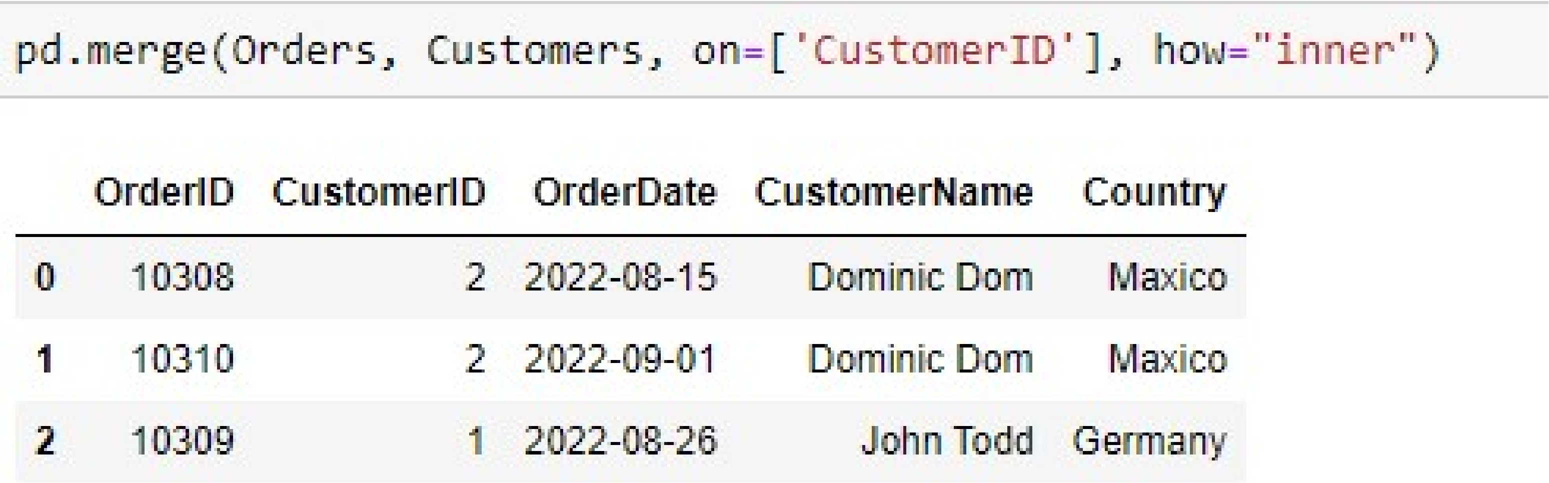
```
print(Orders)
```

```
   OrderID  CustomerID   OrderDate
0    10308           2  2022-08-15
1    10309           1  2022-08-26
2    10310           2  2022-09-01
```

Python Pandas

# Step-4 Perform inner join

```
pd.merge(Orders, Customers, on=['CustomerID'], how="inner")
```



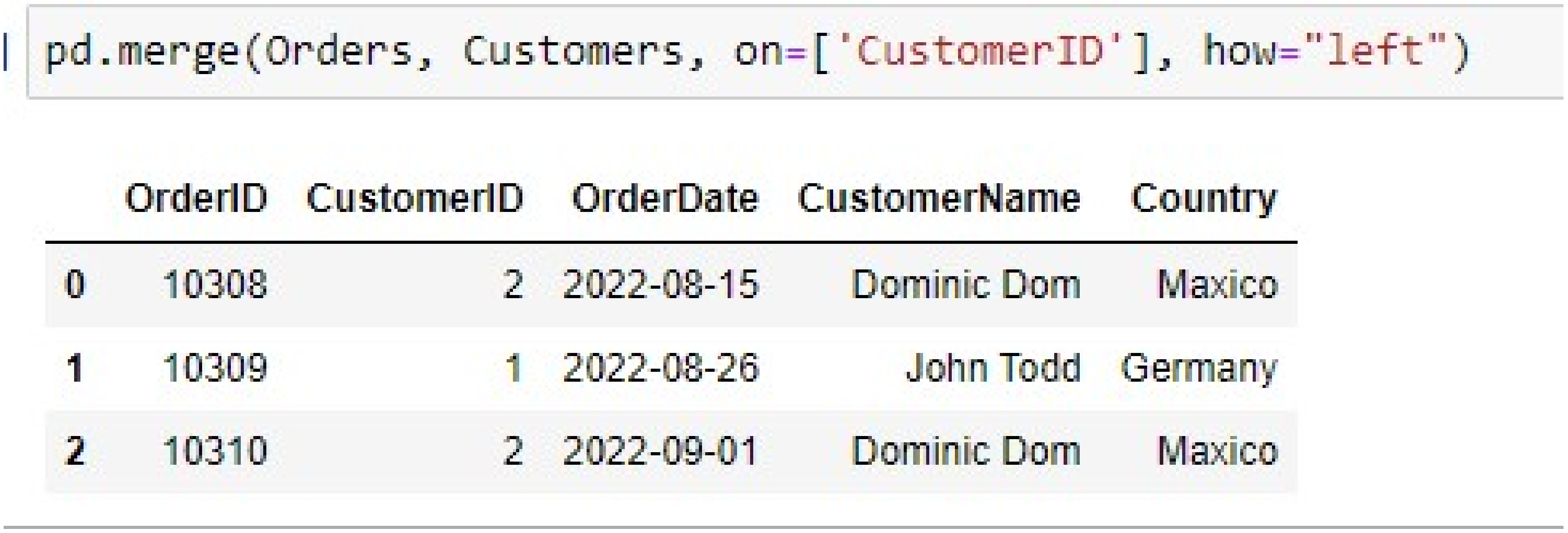
```
pd.merge(Orders, Customers, on=['CustomerID'], how="inner")
```

	OrderID	CustomerID	OrderDate	CustomerName	Country
0	10308	2	2022-08-15	Dominic Dom	Maxico
1	10310	2	2022-09-01	Dominic Dom	Maxico
2	10309	1	2022-08-26	John Todd	Germany

Python Pandas

# Step-5 Perform left join

pd.merge(Orders, Customers, on=['CustomerID'], how="left")



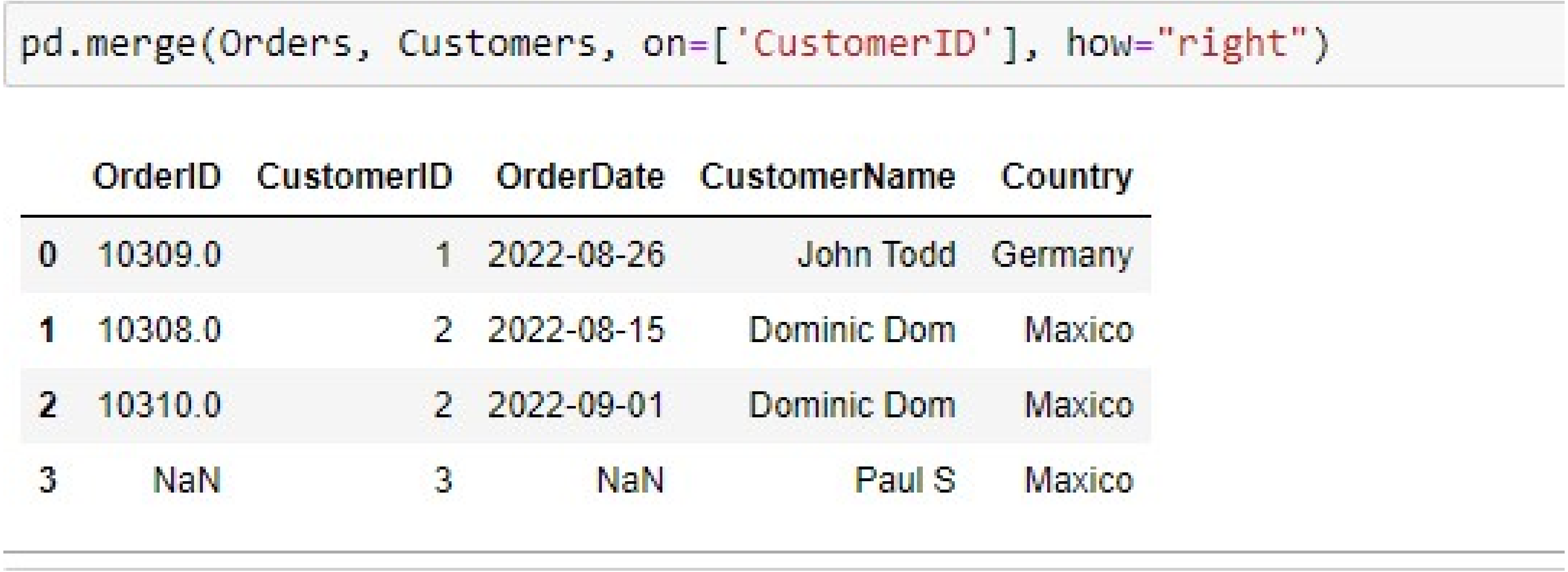
```
pd.merge(Orders, Customers, on=['CustomerID'], how="left")
```

	OrderID	CustomerID	OrderDate	CustomerName	Country
0	10308	2	2022-08-15	Dominic Dom	Maxico
1	10309	1	2022-08-26	John Todd	Germany
2	10310	2	2022-09-01	Dominic Dom	Maxico

Python Pandas

# Step-6 Perform right join

pd.merge(Orders, Customers, on=['CustomerID'], how="right")

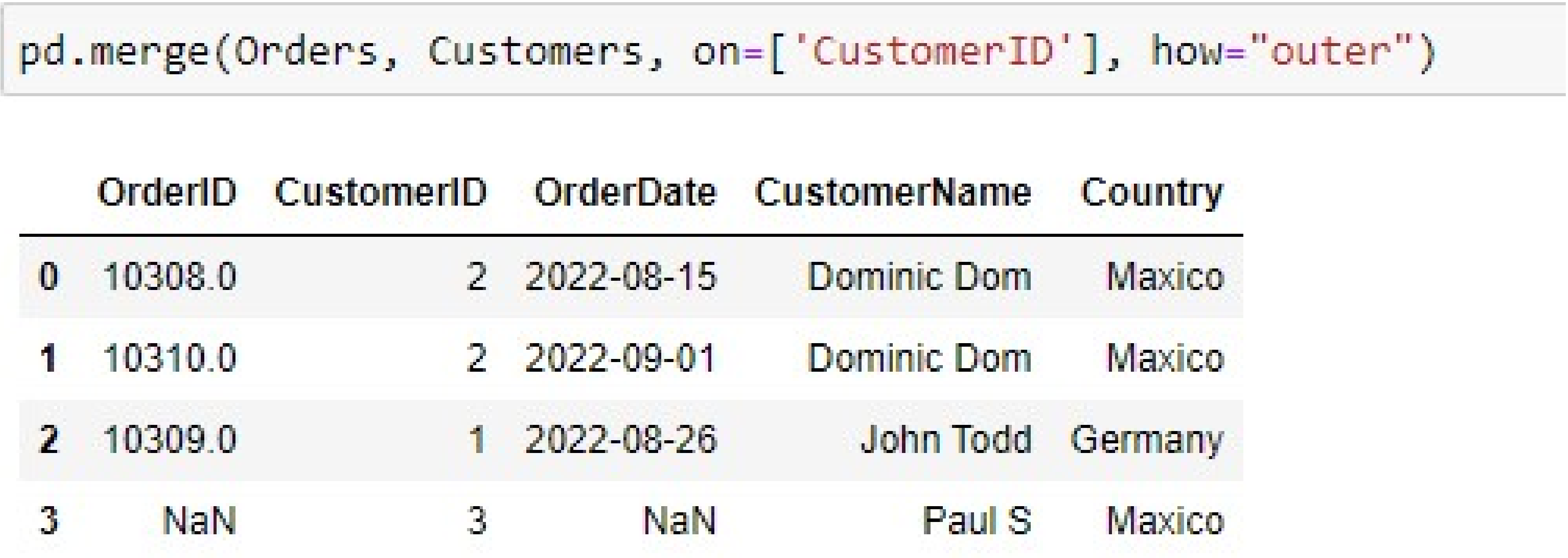


```
pd.merge(Orders, Customers, on=['CustomerID'], how="right")
```

	OrderID	CustomerID	OrderDate	CustomerName	Country
0	10309.0	1	2022-08-26	John Todd	Germany
1	10308.0	2	2022-08-15	Dominic Dom	Maxico
2	10310.0	2	2022-09-01	Dominic Dom	Maxico
3	NaN	3	NaN	Paul S	Maxico

Python Pandas

# Step-7 Perform full join
pd.merge(Orders, Customers, on=['CustomerID'], how="outer")



```
pd.merge(Orders, Customers, on=['CustomerID'], how="outer")
```

	OrderID	CustomerID	OrderDate	CustomerName	Country
0	10308.0	2	2022-08-15	Dominic Dom	Maxico
1	10310.0	2	2022-09-01	Dominic Dom	Maxico
2	10309.0	1	2022-08-26	John Todd	Germany
3	NaN	3	NaN	Paul S	Maxico

Python Pandas

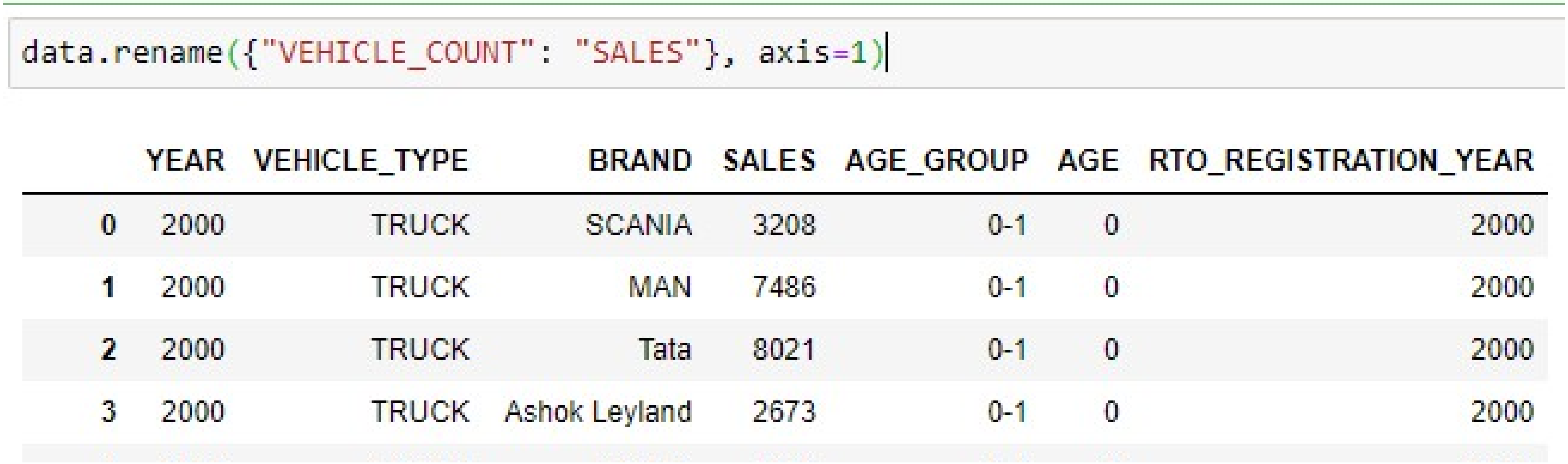
rename() functions to rename the columns in DataFrame

data-frame.rename({“col-name”: “new-col-name”}, axis=1)

- axis=1 represent column axis
- axis=0 represent row axis

Example:

data.rename({"VEHICLE_COUNT": "SALES"}, axis=1)



```
data.rename({"VEHICLE_COUNT": "SALES"}, axis=1)
```

	YEAR	VEHICLE_TYPE	BRAND	SALES	AGE_GROUP	AGE	RTO_REGISTRATION_YEAR
0	2000	TRUCK	SCANIA	3208	0-1	0	2000
1	2000	TRUCK	MAN	7486	0-1	0	2000
2	2000	TRUCK	Tata	8021	0-1	0	2000
3	2000	TRUCK	Ashok Leyland	2673	0-1	0	2000

Python Pandas

rename() functions to rename the columns in DataFrame.

Inplace additional parameters to specify with rename function, to rename column name permanently.

Syntax:

```
data-frame.rename({“col-name”: “new-col-name”},
                  axis=1,
                  inplace=True)
```

Example:

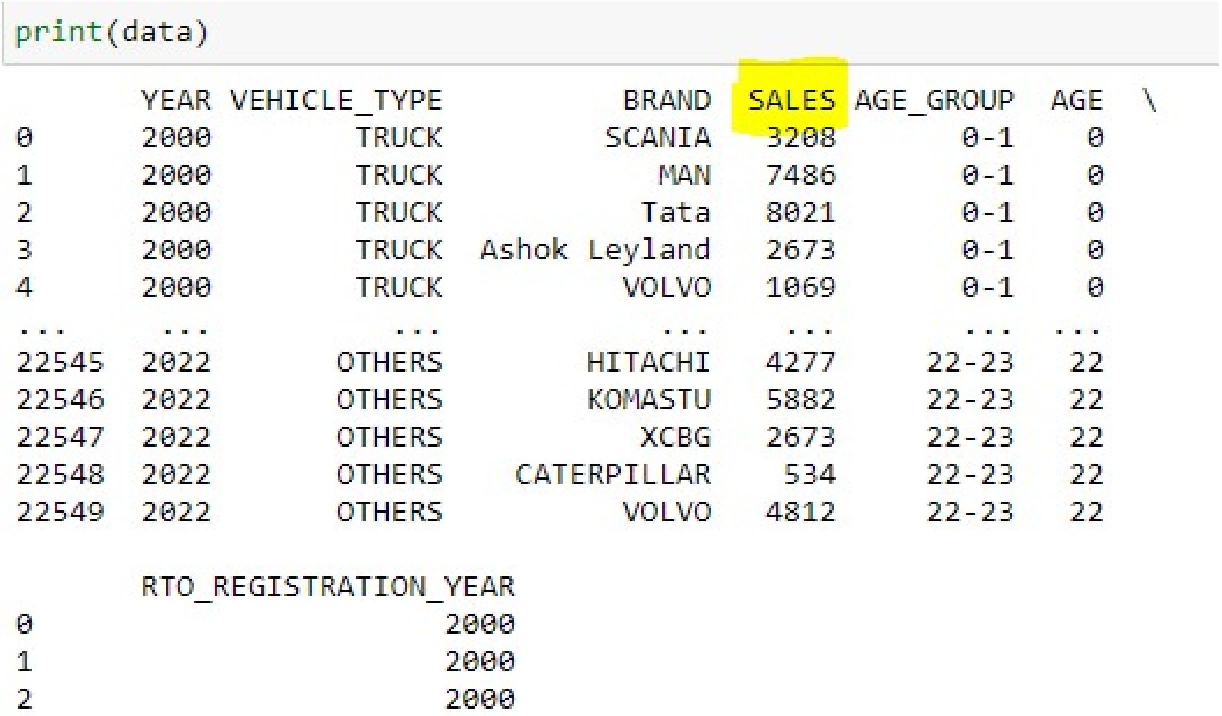
```
data.rename({"VEHICLE_COUNT": "SALES"},
            axis=1,
            inplace=True)
```

Python Pandas

Example:

```
data.rename({"VEHICLE_COUNT": "SALES"},
                    axis=1,
                    inplace=True)


print(data)
```



```
print(data)
```

```
       YEAR VEHICLE_TYPE         BRAND  SALES AGE_GROUP  AGE  \
0      2000        TRUCK        SCANIA   3208       0-1    0
1      2000        TRUCK           MAN   7486       0-1    0
2      2000        TRUCK          Tata   8021       0-1    0
3      2000        TRUCK  Ashok Leyland   2673       0-1    0
4      2000        TRUCK         VOLVO   1069       0-1    0
...     ...          ...           ...    ...       ...  ...
22545  2022       OTHERS       HITACHI   4277     22-23   22
22546  2022       OTHERS       KOMASTU   5882     22-23   22
22547  2022       OTHERS          XCBG   2673     22-23   22
22548  2022       OTHERS   CATERPILLAR    534     22-23   22
22549  2022       OTHERS         VOLVO   4812     22-23   22

       RTO_REGISTRATION_YEAR
0                       2000
1                       2000
2                       2000
```

Python Pandas

drop() functions to delete columns from DataFrame.

Inplace additional parameters to specify with drop function, to delete column name permanently.

Syntax:

```
data-frame.drop({“col-name”: “new-col-name”},
                axis=1,
                inplace=True)
```

Example:

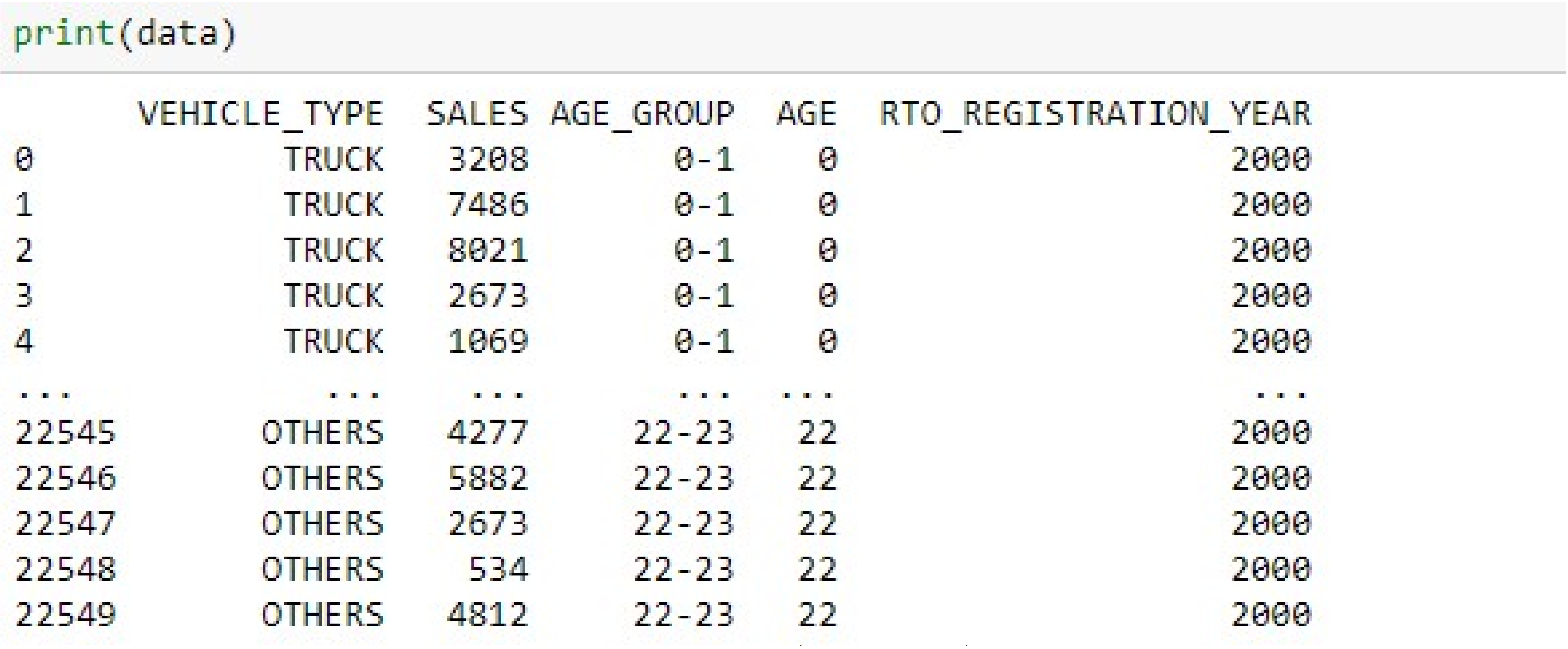
```
data.drop(["YEAR", "BRAND"], axis=1, inplace=True)
```

Python Pandas

Example:

data.drop(["YEAR", "BRAND"], axis=1, inplace=True)

print(data)



```
print(data)
```

```
      VEHICLE_TYPE  SALES AGE_GROUP  AGE  RTO_REGISTRATION_YEAR
0            TRUCK   3208       0-1    0                   2000
1            TRUCK   7486       0-1    0                   2000
2            TRUCK   8021       0-1    0                   2000
3            TRUCK   2673       0-1    0                   2000
4            TRUCK   1069       0-1    0                   2000
...            ...    ...       ...  ...                    ...
22545       OTHERS   4277     22-23   22                   2000
22546       OTHERS   5882     22-23   22                   2000
22547       OTHERS   2673     22-23   22                   2000
22548       OTHERS    534     22-23   22                   2000
22549       OTHERS   4812     22-23   22                   2000
```

Python Pandas

Panda DataFrame allows to write conditions to filter out the data.

Syntax:

data-frame[conditions]

Example: display 10 years old vehicles from DataFrame

```
data[data['AGE']==10]
```

Python Pandas

Panda DataFrame allows to write conditions to filter out the data.

Multiple conditions can be combine with the help of logical operator ie. AND (&), OR (|)

Syntax:

```
data-frame[conditions]
data-frame[(conditions-1) & (conditions-2)] #AND(&) condition
data-frame[(conditions-1) I (conditions-2)] #OR(I) condition
```

Example: display 10 years old FOUR WHEELER vehicles from DataFrame

```
data[ (data['AGE']==10) &
        (data['VEHICLE_TYPE']=='FOUR WHEELER')
   ]
```

Data Visualization

- Python Libraries used for generating graphs
  - **Matplotlib**
  - **Seaborn**

- Download data visualization libraries

```
pip install matplotlib
pip install seaborn
```

Data Visualization

- Importing Matplotlib & Seaborn libraries

```
import matplotlib.pyplot as plt
import seaborn as sn
%matplotlib inline
```

Data Visualization Functions

- **Bar Graphs Syntax**

sn.barplot(x=, y=, data=)

sn.barplot(x=, y=, hue= ,data=)

Data Visualization

- **Histogram**

plt.hist(x, bins=)

- **Distribution or Density Plot**

sn.distplot(x=)

- **Boxplot**

sn.boxplot(x=, data=)

sn.boxplot(x=,y=, data=)

Data Visualization

- **Pairplot**

```
sn.pairplot(df[selected-columns])

sn.pairplot(df)         #For all the columns
```

- **Heatmap**

sn.heatmap(df[selected-columns].corr(), **annot=True**)
***annotate show the data values on graph