19162121031

SMIT R PATEL

BDA SEM 5

PRACTICAL 19

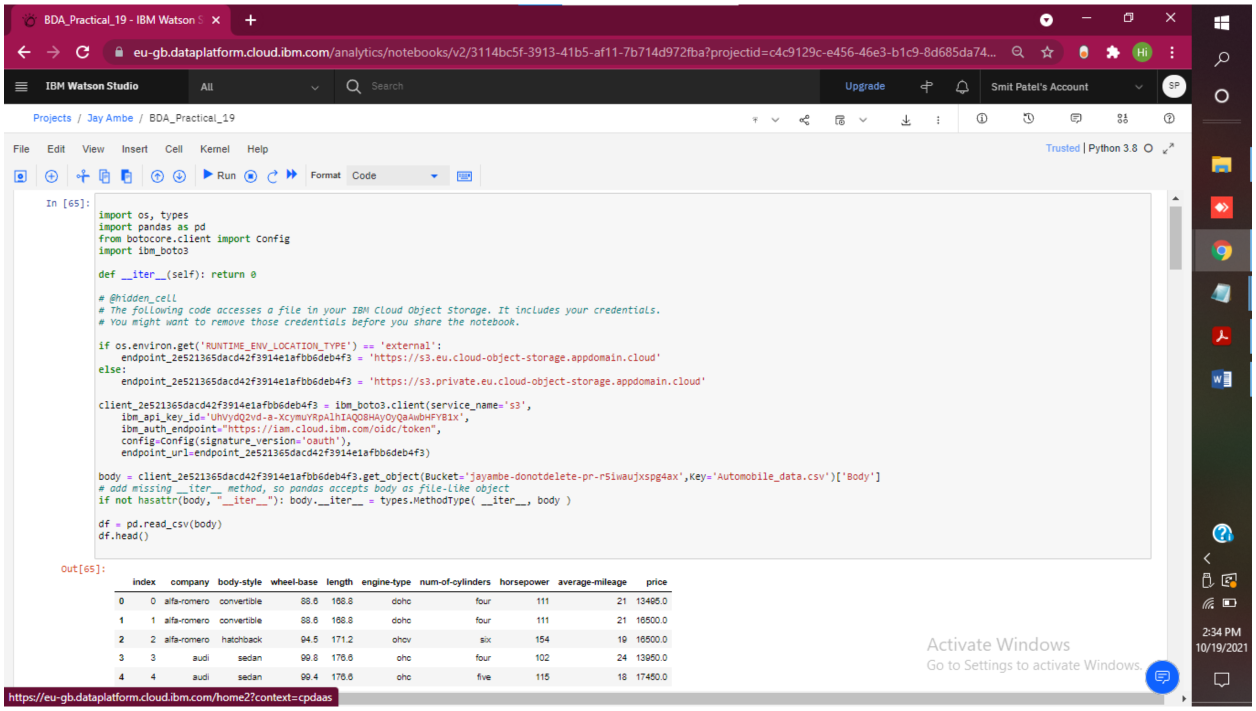
In this exercise, we are using **Automobile Dataset** for data analysis. This Dataset has different characteristics of an auto such as body-style, wheel-base, engine-type, price, mileage, horsepower, etc.

What included in this Pandas exercise?

- It contains 10 questions. The solution is provided for each question.
- Each question includes a specific Pandas topic you need to learn.

When you complete each question, you get more familiar with data analysis using pandas.

Exercise 1: From the given dataset print the first and last five rows

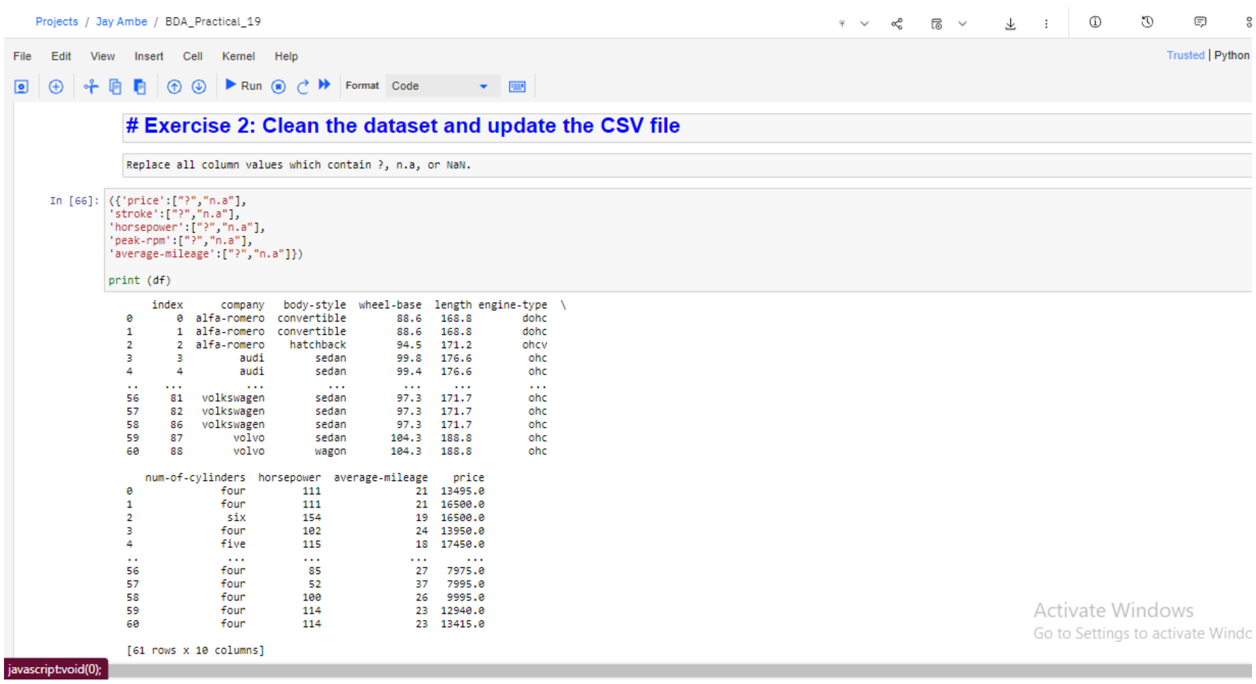


In [65]:

```
import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.

if os.environ.get('RUNTIME_ENV_LOCATION_TYPE') == 'external':
    endpoint_2e521365dacd42f3914e1afbb6deb4f3 = 'https://s3.eu.cloud-object-storage.appdomain.cloud'
else:
    endpoint_2e521365dacd42f3914e1afbb6deb4f3 = 'https://s3.private.eu.cloud-object-storage.appdomain.cloud'

client_2e521365dacd42f3914e1afbb6deb4f3 = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='UhVydQ2vd-a-XcymuYRpAlhIAQO8HAyOyQaAwbHFYB1x',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url=endpoint_2e521365dacd42f3914e1afbb6deb4f3)

body = client_2e521365dacd42f3914e1afbb6deb4f3.get_object(Bucket='jayambe-donotdelete-pr-r5iwaujxspg4ax',Key='Automobile_data.csv')['Body']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

df = pd.read_csv(body)
df.head()
```

Out[65]:

	index	company	body-style	wheel-base	length	engine-type	num-of-cylinders	horsepower	average-mileage	price
0	0	alfa-romero	convertible	88.6	168.8	dohc	four	111	21	13495.0
1	1	alfa-romero	convertible	88.6	168.8	dohc	four	111	21	16500.0
2	2	alfa-romero	hatchback	94.5	171.2	ohcv	six	154	19	16500.0
3	3	audi	sedan	99.8	176.6	ohc	four	102	24	13950.0
4	4	audi	sedan	99.4	176.6	ohc	five	115	18	17450.0



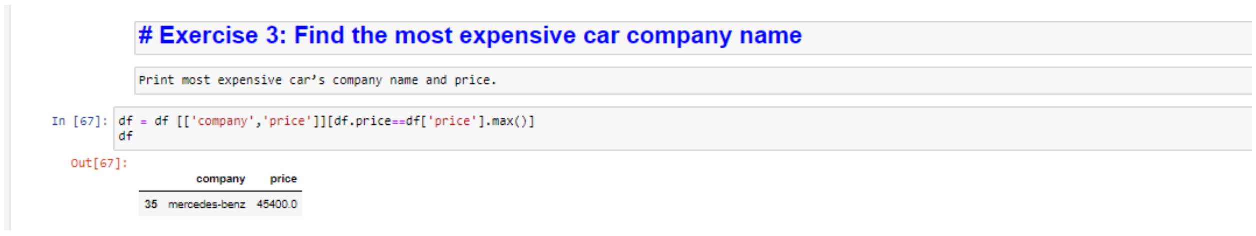
Exercise 2: Clean the dataset and update the CSV file

Replace all column values which contain ?, n.a, or NaN.

```
In [66]: ({'price':["?","n.a"],
'stroke':["?","n.a"],
'horsepower':["?","n.a"],
'peak-rpm':["?","n.a"],
'average-mileage':["?","n.a"]})

print (df)
```

```
    index      company   body-style  wheel-base  length engine-type  \
0       0  alfa-romero  convertible        88.6   168.8        dohc
1       1  alfa-romero  convertible        88.6   168.8        dohc
2       2  alfa-romero    hatchback        94.5   171.2        ohcv
3       3         audi        sedan        99.8   176.6         ohc
4       4         audi        sedan        99.4   176.6         ohc
..    ...          ...          ...         ...     ...         ...
56     81   volkswagen        sedan        97.3   171.7         ohc
57     82   volkswagen        sedan        97.3   171.7         ohc
58     86   volkswagen        sedan        97.3   171.7         ohc
59     87        volvo        sedan       104.3   188.8         ohc
60     88        volvo        wagon       104.3   188.8         ohc

   num-of-cylinders  horsepower  average-mileage    price
0              four         111               21  13495.0
1              four         111               21  16500.0
2               six         154               19  16500.0
3              four         102               24  13950.0
4              five         115               18  17450.0
..              ...         ...              ...      ...
56             four          85               27   7975.0
57             four          52               37   7995.0
58             four         100               26   9995.0
59             four         114               23  12940.0
60             four         114               23  13415.0

[61 rows x 10 columns]
```

Activate Windows
Go to Settings to activate Windc



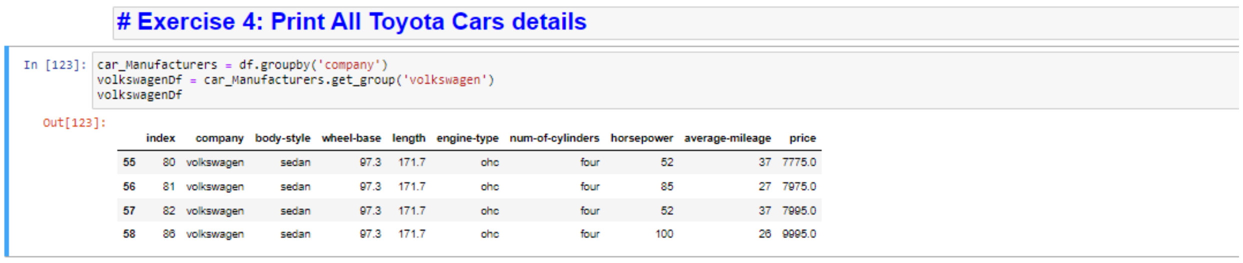
Exercise 3: Find the most expensive car company name

Print most expensive car's company name and price.

```
In [67]: df = df [['company','price']][df.price==df['price'].max()]
df
```

Out[67]:

	company	price
35	mercedes-benz	45400.0

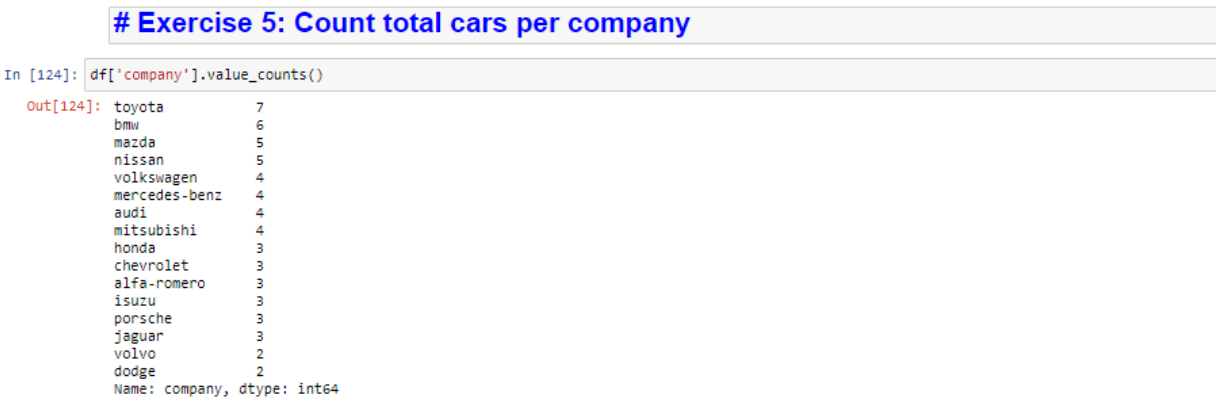


Exercise 4: Print All Toyota Cars details

```
In [123]: car_Manufacturers = df.groupby('company')
volkswagenDf = car_Manufacturers.get_group('volkswagen')
volkswagenDf
```

Out[123]:

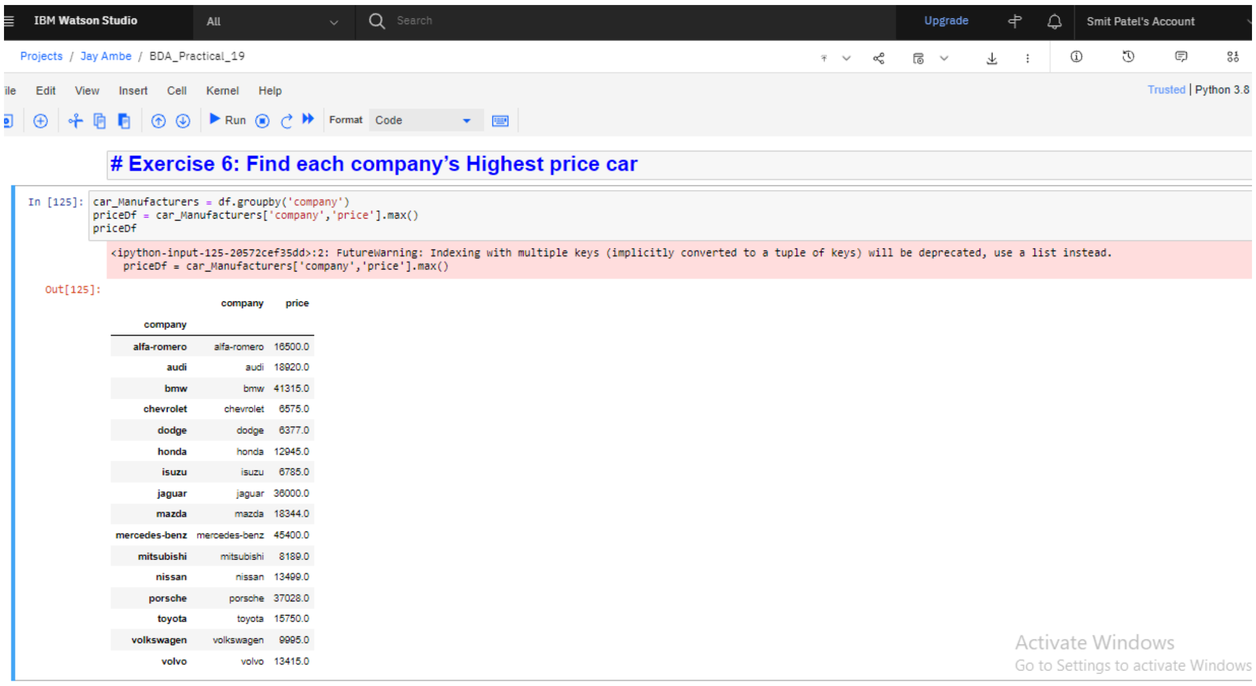
	index	company	body-style	wheel-base	length	engine-type	num-of-cylinders	horsepower	average-mileage	price
55	80	volkswagen	sedan	97.3	171.7	ohc	four	52	37	7775.0
56	81	volkswagen	sedan	97.3	171.7	ohc	four	85	27	7975.0
57	82	volkswagen	sedan	97.3	171.7	ohc	four	52	37	7995.0
58	86	volkswagen	sedan	97.3	171.7	ohc	four	100	26	9995.0



Exercise 5: Count total cars per company

```
In [124]: df['company'].value_counts()
```

```
Out[124]: toyota           7
          bmw              6
          mazda            5
          nissan           5
          volkswagen       4
          mercedes-benz    4
          audi             4
          mitsubishi       4
          honda            3
          chevrolet        3
          alfa-romero      3
          isuzu            3
          porsche          3
          jaguar           3
          volvo            2
          dodge            2
          Name: company, dtype: int64
```


IBM Watson Studio
All
Search
Upgrade
Smit Patel's Account
Projects / Jay Ambe / BDA_Practical_19
ile Edit View Insert Cell Kernel Help
Trusted | Python 3.8
Run
Format Code
Activate Windows
Go to Settings to activate Windows

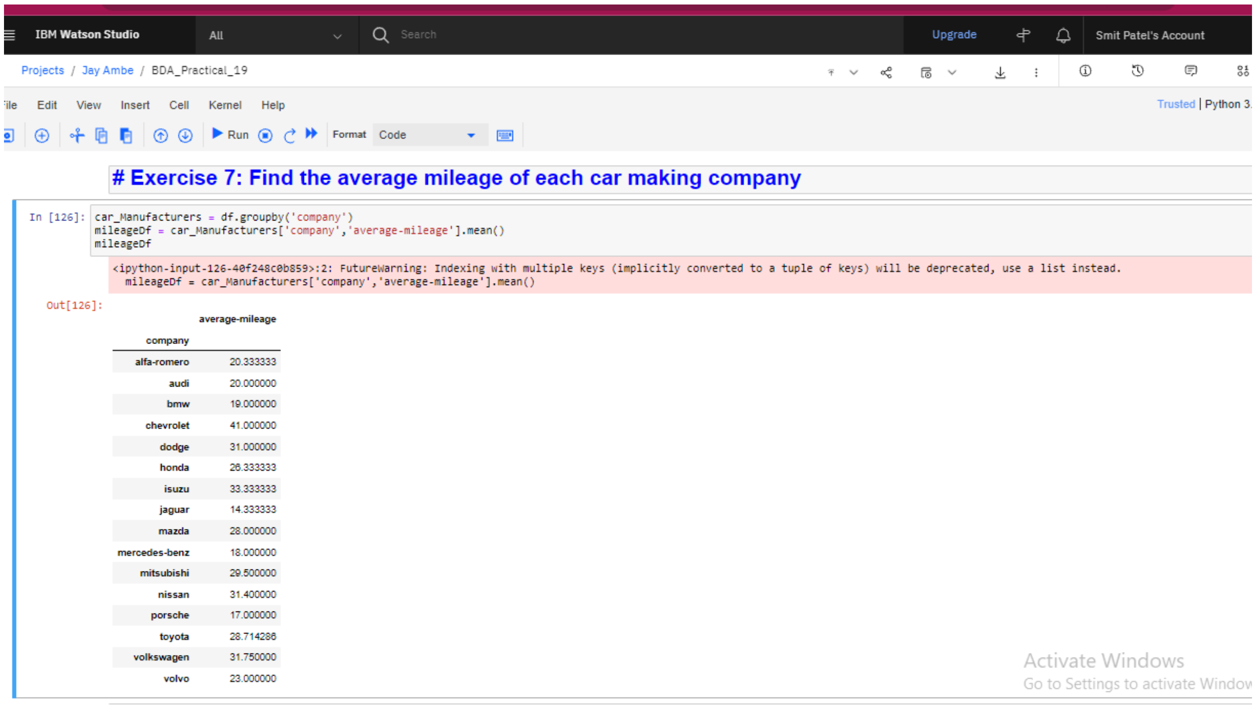
Exercise 6: Find each company's Highest price car

```
In [125]: car_Manufacturers = df.groupby('company')
          priceDf = car_Manufacturers['company','price'].max()
          priceDf
```

```
<ipython-input-125-20572cef35dd>:2: FutureWarning: Indexing with multiple keys (implicitly converted to a tuple of keys) will be deprecated, use a list instead.
  priceDf = car_Manufacturers['company','price'].max()
```

Out[125]:

	company	price
company		
alfa-romero	alfa-romero	16500.0
audi	audi	18920.0
bmw	bmw	41315.0
chevrolet	chevrolet	6575.0
dodge	dodge	6377.0
honda	honda	12945.0
isuzu	isuzu	6785.0
jaguar	jaguar	36000.0
mazda	mazda	18344.0
mercedes-benz	mercedes-benz	45400.0
mitsubishi	mitsubishi	8189.0
nissan	nissan	13499.0
porsche	porsche	37028.0
toyota	toyota	15750.0
volkswagen	volkswagen	9995.0
volvo	volvo	13415.0



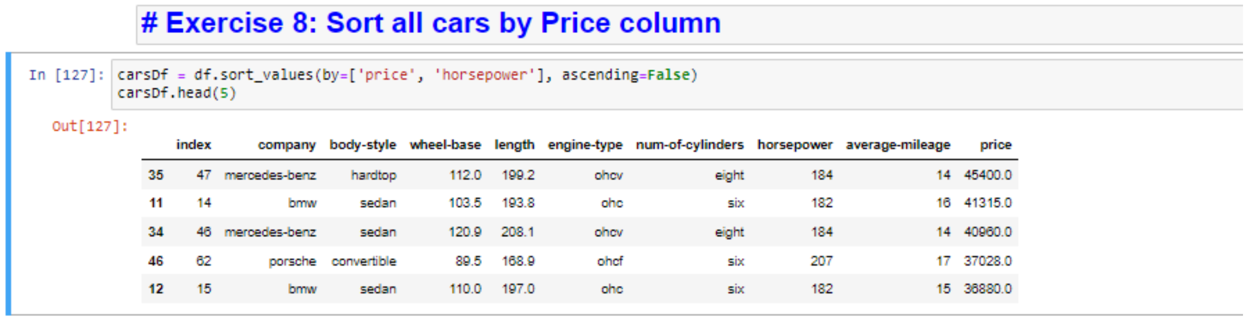
Exercise 7: Find the average mileage of each car making company

```
In [126]: car_Manufacturers = df.groupby('company')
          mileageDf = car_Manufacturers['company','average-mileage'].mean()
          mileageDf
```

```
<ipython-input-126-40f248c0b859>:2: FutureWarning: Indexing with multiple keys (implicitly converted to a tuple of keys) will be deprecated, use a list instead.
  mileageDf = car_Manufacturers['company','average-mileage'].mean()
```

Out[126]:

	average-mileage
company	
alfa-romero	20.333333
audi	20.000000
bmw	19.000000
chevrolet	41.000000
dodge	31.000000
honda	26.333333
isuzu	33.333333
jaguar	14.333333
mazda	28.000000
mercedes-benz	18.000000
mitsubishi	29.500000
nissan	31.400000
porsche	17.000000
toyota	28.714286
volkswagen	31.750000
volvo	23.000000

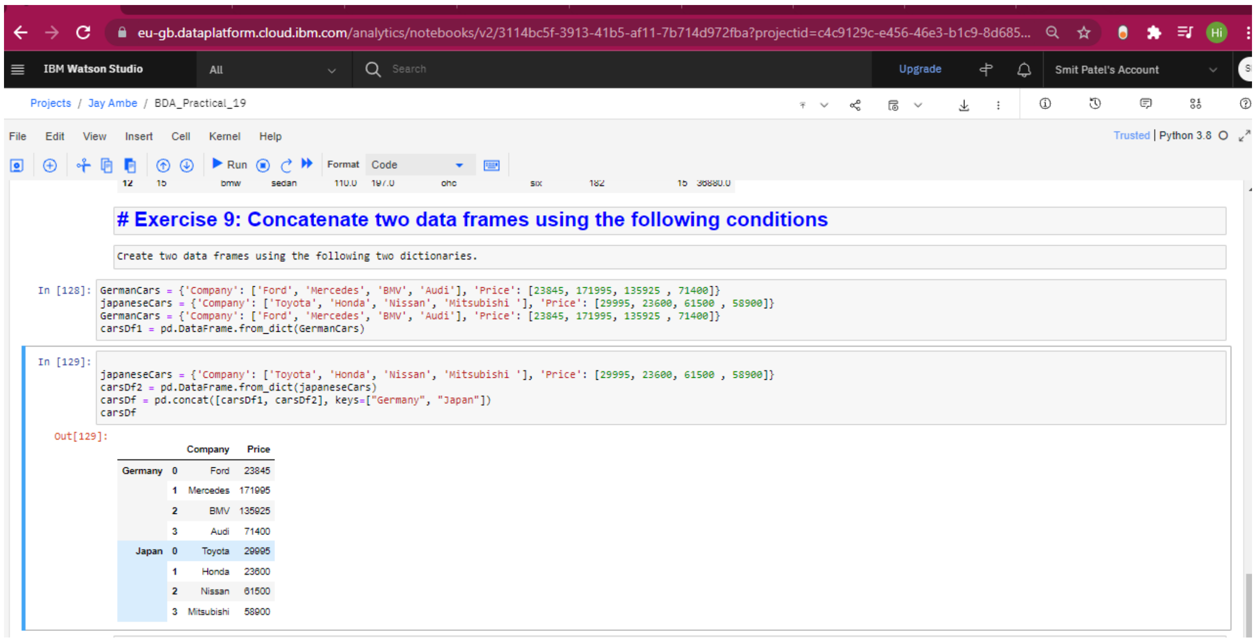


Exercise 8: Sort all cars by Price column

```
In [127]: carsDf = df.sort_values(by=['price', 'horsepower'], ascending=False)
          carsDf.head(5)
```

Out[127]:

	index	company	body-style	wheel-base	length	engine-type	num-of-cylinders	horsepower	average-mileage	price
35	47	mercedes-benz	hardtop	112.0	199.2	ohcv	eight	184	14	45400.0
11	14	bmw	sedan	103.5	193.8	ohc	six	182	16	41315.0
34	46	mercedes-benz	sedan	120.9	208.1	ohcv	eight	184	14	40960.0
46	62	porsche	convertible	89.5	168.9	ohcf	six	207	17	37028.0
12	15	bmw	sedan	110.0	197.0	ohc	six	182	15	36880.0



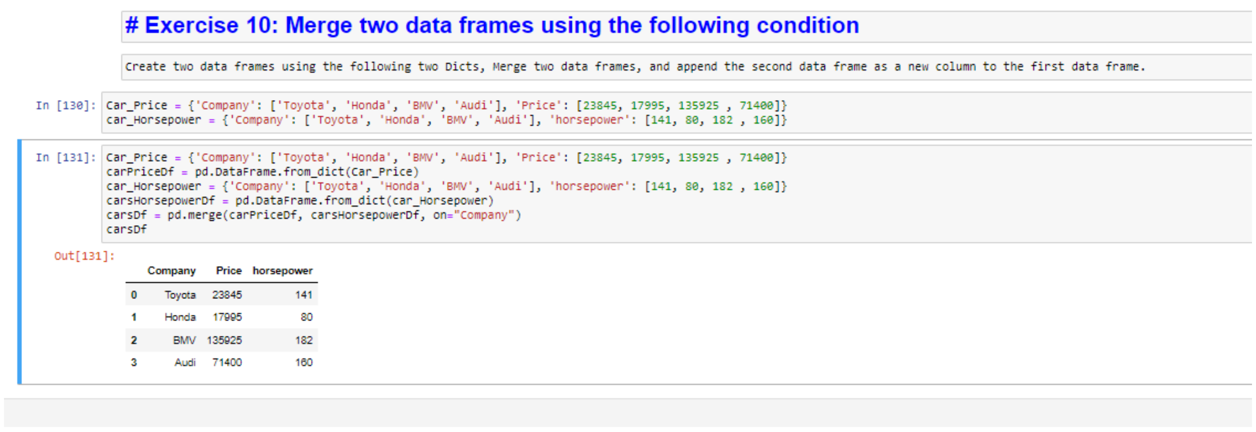
Exercise 9: Concatenate two data frames using the following conditions

```
Create two data frames using the following two dictionaries.
```

```
In [128]: GermanCars = {'Company': ['Ford', 'Mercedes', 'BMV', 'Audi'], 'Price': [23845, 171995, 135925 , 71400]}
          japaneseCars = {'Company': ['Toyota', 'Honda', 'Nissan', 'Mitsubishi '], 'Price': [29995, 23600, 61500 , 58900]}
          GermanCars = {'Company': ['Ford', 'Mercedes', 'BMV', 'Audi'], 'Price': [23845, 171995, 135925 , 71400]}
          carsDf1 = pd.DataFrame.from_dict(GermanCars)
```

```
In [129]:
          japaneseCars = {'Company': ['Toyota', 'Honda', 'Nissan', 'Mitsubishi '], 'Price': [29995, 23600, 61500 , 58900]}
          carsDf2 = pd.DataFrame.from_dict(japaneseCars)
          carsDf = pd.concat([carsDf1, carsDf2], keys=["Germany", "Japan"])
          carsDf
```

```
Out[129]:
                   Company   Price
          Germany 0      Ford   23845
                  1  Mercedes  171995
                  2       BMV  135925
                  3      Audi   71400
          Japan   0    Toyota   29995
                  1     Honda   23600
                  2    Nissan   61500
                  3 Mitsubishi  58900
```



Exercise 10: Merge two data frames using the following condition

```
Create two data frames using the following two Dicts, Merge two data frames, and append the second data frame as a new column to the first data frame.
```

```
In [130]: Car_Price = {'Company': ['Toyota', 'Honda', 'BMV', 'Audi'], 'Price': [23845, 17995, 135925 , 71400]}
          car_Horsepower = {'Company': ['Toyota', 'Honda', 'BMV', 'Audi'], 'horsepower': [141, 80, 182 , 160]}
```

```
In [131]: Car_Price = {'Company': ['Toyota', 'Honda', 'BMV', 'Audi'], 'Price': [23845, 17995, 135925 , 71400]}
          carPriceDf = pd.DataFrame.from_dict(Car_Price)
          car_Horsepower = {'Company': ['Toyota', 'Honda', 'BMV', 'Audi'], 'horsepower': [141, 80, 182 , 160]}
          carsHorsepowerDf = pd.DataFrame.from_dict(car_Horsepower)
          carsDf = pd.merge(carPriceDf, carsHorsepowerDf, on="Company")
          carsDf
```

```
Out[131]:
             Company   Price  horsepower
          0   Toyota   23845         141
          1    Honda   17995          80
          2      BMV  135925         182
          3     Audi   71400         160
```