PANDAS PART 5

Comparison with SQL Queries

NOTE: Use 'shoppingmall.csv' and 'shopping_customers.csv' datasets.

NOTE: In Jupyter, to see only 20 rows at a time:

pd.set_option("display.max_rows", 20)

To see all rows:

pd.set_option("display.max_rows", None)

[76]:
```
import pandas as pd
```

[77]:
```
# load the data set into a data frame
customers = pd.read_csv('D:/aatish/datasets/shoppingmall.csv')
```

[5]:
```
# SQL -> select * from customers
customers
```

t[5]:

	ID	Gender	Age	Income	Score
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40
...	...	...	...	...	...
195	196	Female	35	120	79
196	197	Female	45	126	28
197	198	Male	32	126	74
198	199	Male	32	137	18
199	200	Male	30	137	83

200 rows × 5 columns

[78]:
```
# SQL -> select * from customers limit 8
customers.head(8)
```

[78]:

	ID	Gender	Age	Income	Score
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40
5	6	Female	22	17	76
6	7	Female	35	18	6
7	8	Female	23	18	94

[79]:
```
# SQL -> select * from customers where Income=23
customers[customers.Income==23]
```

[79]:

	ID	Gender	Age	Income	Score
18	19	Male	52	23	29
19	20	Female	35	23	98

[80]:
```
# SQL -> select ID from customers where Income=23
customers[customers.Income==23].ID
```

[80]:
```
18    19
19    20
Name: ID, dtype: int64
```

[81]:
```
# SQL -> select * from customers where Income=23 and Score=98
customers[(customers.Income==23) & (customers.Score==98)]
```

[81]:

	ID	Gender	Age	Income	Score
19	20	Female	35	23	98

[82]:
```
# SQL -> select ID, Age from customers where Income=23 and Score =98
customers[(customers.Income==23) & (customers.Score==98)][['ID', 'Age']]
```

[82]:

	ID	Age
19	20	35

[83]:
```
# SQL -> select mean(Age), max(Age), min(Age) from customers
customers.agg({'Age': ['mean', 'max', 'min']})
```

[83]:

	Age
mean	38.85
max	70.00
min	18.00

[84]:
```
# SQL -> select distinct Income from customers
# customers.Income.unique()
customers['Income'].unique()
```

[84]:
```
array([ 15,  16,  17,  18,  19,  20,  21,  23,  24,  25,  28,  29,  30,
        33,  34,  37,  38,  39,  40,  42,  43,  44,  46,  47,  48,  49,
        50,  54,  57,  58,  59,  60,  61,  62,  63,  64,  65,  67,  69,
        70,  71,  72,  73,  74,  75,  76,  77,  78,  79,  81,  85,  86,
        87,  88,  93,  97,  98,  99, 101, 103, 113, 120, 126, 137],
      dtype=int64)
```

[85]:
```
# SQL -> select * from customers where Gender='Female' order by Score
customers[customers.Gender=='Female'].sort_values('Score')
```

[85]:

	ID	Gender	Age	Income	Score
140	141	Female	57	75	5
22	23	Female	46	25	5
2	3	Female	20	16	6
6	7	Female	35	18	6
136	137	Female	44	73	7
...	...	...	...	...	...
163	164	Female	31	81	93
7	8	Female	23	18	94
167	168	Female	33	86	95
19	20	Female	35	23	98
11	12	Female	35	19	99

112 rows × 5 columns

[86]:
```
# SQL -> select * from customers where Gender='Female' order by Score desc
customers[customers.Gender=='Female'].sort_values('Score', ascending=False)
```

[86]:

	ID	Gender	Age	Income	Score
11	12	Female	35	19	99
19	20	Female	35	23	98
167	168	Female	33	86	95
7	8	Female	23	18	94
163	164	Female	31	81	93
...	...	...	...	...	...
136	137	Female	44	73	7
6	7	Female	35	18	6
2	3	Female	20	16	6
22	23	Female	46	25	5
140	141	Female	57	75	5

112 rows × 5 columns

[87]:
```
# SQL -> select Gender, Age, count(*) from customers groupby Gender, Age
customers.groupby(['Gender', 'Age']).size().to_frame('Count').reset_index()
```

[87]:

	Gender	Age	Count
0	Female	18	1
1	Female	19	2
2	Female	20	2
3	Female	21	4
4	Female	22	2
...	...	...	...
82	Male	66	1
83	Male	67	3
84	Male	68	1
85	Male	69	1
86	Male	70	2

87 rows × 3 columns

[94]:
```
# SQL -> select Gender, Age, count(*) from customers
# group by Gender, Age order by Age desc
customers.groupby(['Gender', 'Age']).size().to_frame('Count')
.reset_index().sort_values('Age', ascending=False)
```

[94]:

	Gender	Age	Count
86	Male	70	2
85	Male	69	1
42	Female	68	2
84	Male	68	1
83	Male	67	3
...	...	...	...
2	Female	20	2
1	Female	19	2
44	Male	19	6
0	Female	18	1
43	Male	18	3

[89]:
```
# SQL -> select * from customers where Age in (20, 30, 40)
customers[customers.Age.isin([20,30,40])]
```

[89]:

	ID	Gender	Age	Income	Score
2	3	Female	20	16	6
9	10	Female	30	19	72
17	18	Male	20	21	66
28	29	Female	40	29	31
37	38	Female	30	34	73
39	40	Female	20	37	75
77	78	Male	40	54	48
93	94	Female	40	60	40
99	100	Male	20	61	49
122	123	Female	40	69	58
127	128	Male	40	71	95
134	135	Male	20	73	5

[90]:
```
# SQL -> select * from customers where Age not in (20, 30, 40)
customers[~customers.Age.isin([20,30,40])]
```

[90]:

	ID	Gender	Age	Income	Score
0	1	Male	19	15	39
1	2	Male	21	15	81
3	4	Female	23	16	77
4	5	Female	31	17	40
5	6	Female	22	17	76
...	...	...	...	...	...
194	195	Female	47	120	16
195	196	Female	35	120	79
196	197	Female	45	126	28
197	198	Male	32	126	74
198	199	Male	32	137	18

182 rows × 5 columns

[91]:
```
# to retrieve top 10 rows after ordering on Score in descending order
# SQL -> select * from customers orderby Score desc limit 10
customers.nlargest(10, columns='Score')
```

[91]:

	ID	Gender	Age	Income	Score
11	12	Female	35	19	99
19	20	Female	35	23	98
145	146	Male	28	77	97
185	186	Male	30	99	97
127	128	Male	40	71	95
167	168	Female	33	86	95
7	8	Female	23	18	94
141	142	Male	32	75	93
163	164	Female	31	81	93
33	34	Male	18	33	92

[92]:
```
# to retrieve the next 10 rows after ordering on Score into descending order
# SQL -> select * from customers orderby Score desc limit 10 offset 10
customers.nlargest(20, columns='Score').tail(10)
```

[92]:

	ID	Gender	Age	Income	Score
41	42	Male	24	38	92
173	174	Male	36	87	92
123	124	Male	39	69	91
193	194	Female	38	113	91
149	150	Male	34	78	90
179	180	Male	35	93	90
155	156	Female	27	78	89
135	136	Female	29	73	88
151	152	Male	39	78	88
183	184	Female	29	98	88

[43]:
```
# load shopping_customers.csv into another dataframe
shop_customers = pd.read_csv('D:/aatish/datasets/shopping_customers.csv')
```

[75]:
```
# SQL -> select * from customers WHERE Income>50 UNION ALL
# select * from shopping_customers WHERE Income<45
pd.concat([customers[customers.Income>50],shop_customers[shop_customers.Income<45]])
```

[75]:

	ID	Gender	Age	Income	Score
74	75	Male	59	54	47
75	76	Male	26	54	54
76	77	Female	45	54	53
77	78	Male	40	54	48
78	79	Female	23	54	52
...	...	...	...	...	...
15	216	Male	64	26	79
16	217	Female	67	26	35
17	218	Male	67	27	66

[45]:
```
# SQL -> select * from customers WHERE Income>50 UNION
# select * from shopping_customers WHERE Income<45
pd.concat([customers[customers.Income>50],shop_customers[shop_customers.Income<45]])
.drop_duplicates()
```

[45]:

	ID	Gender	Age	Income	Score
74	75	Male	59	54	47
75	76	Male	26	54	54
76	77	Female	45	54	53
77	78	Male	40	54	48
78	79	Female	23	54	52
...	...	...	...	...	...
15	216	Male	64	26	79
16	217	Female	67	26	35
17	218	Male	67	27	66
18	219	Male	20	28	29
19	220	Female	23	24	98

[51]:
```
# SQL -> insert into customers VALUES(500, 'Male', 50,30,20)
customers = customers.append({'ID':500, 'Gender':'Male', 'Age':50, 'Income':30,
                              'Score':20}, ignore_index=True)
```

[50]:
```
customers
```

[50]:

	ID	Gender	Age	Income	Score
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40
...	...	...	...	...	...
195	196	Female	35	120	79
196	197	Female	45	126	28
197	198	Male	32	126	74
198	199	Male	32	137	18
199	200	Male	30	137	83

200 rows × 5 columns

[60]:
```
# SQL -> update customers set  Score=11 where Score= 10
customers.loc[customers['Score']==10, 'Score'] = 11
customers[customers.Score==11]  # to view them
```

[60]:

	ID	Gender	Age	Income	Score
128	129	Male	59	71	11
138	139	Male	19	74	11
172	173	Male	36	87	11

[73]:
```
# SQL -> delete from customers where Score=11
customers = customers.drop(customers[customers.Score==11].index)
```

[74]:
```
# to chek the rows with Score 11 are deleted or not
customers[customers.Score==11]
```

[74]:

	ID	Gender	Age	Income	Score