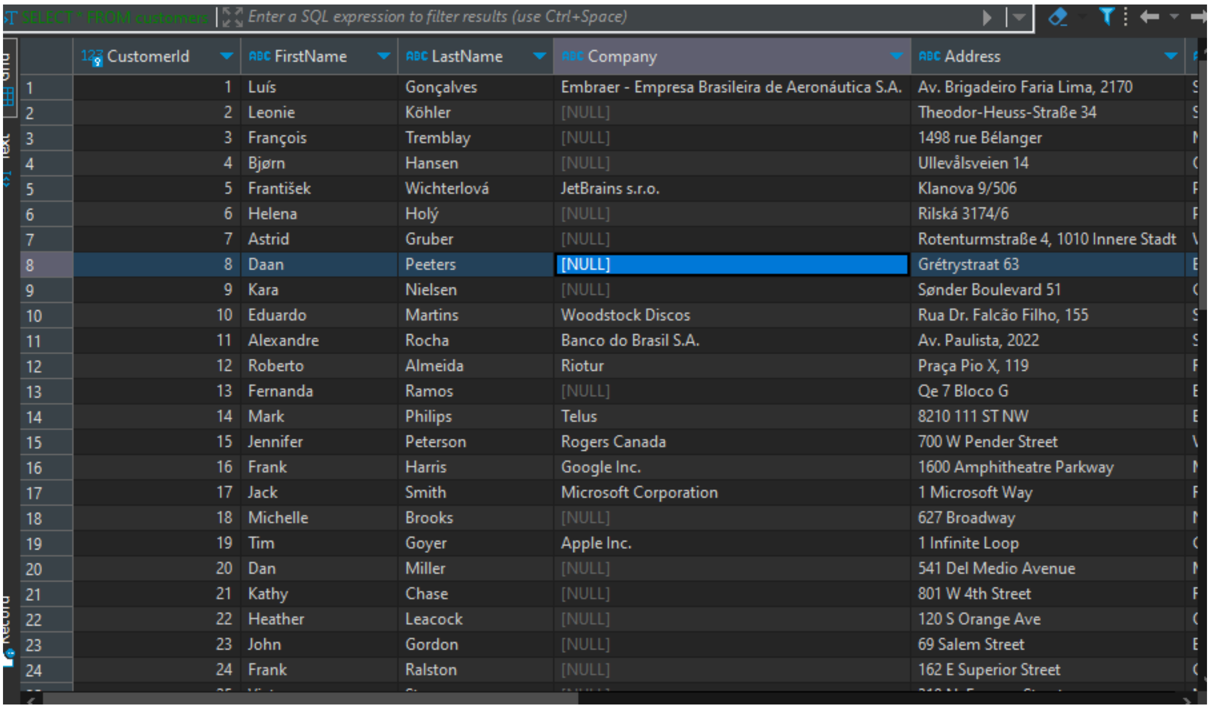
Comparison of SQL and PYTHON

SQL query

SELECT * FROM customers;

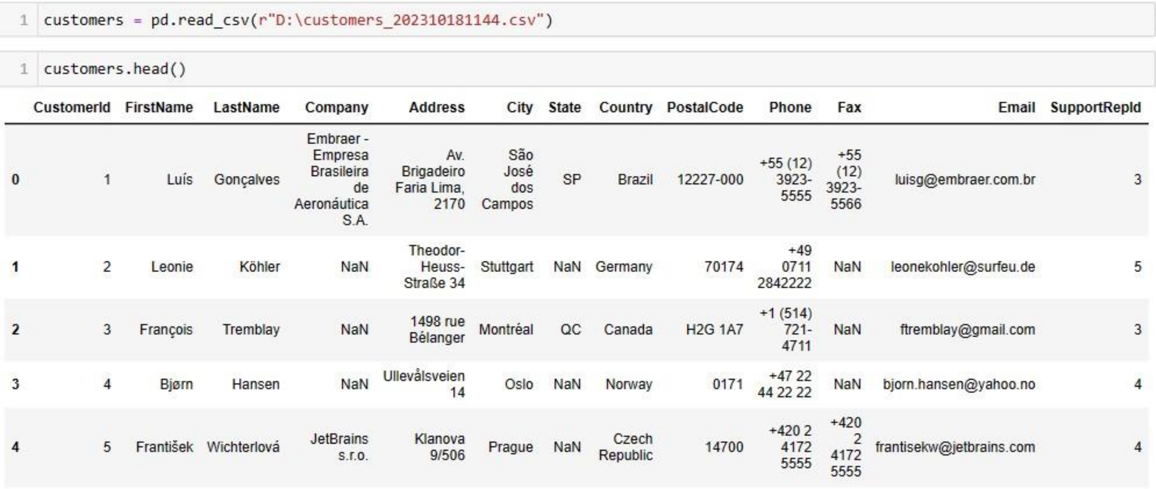


SELECT * FROM customers | Enter a SQL expression to filter results (use Ctrl+Space)

	CustomerId	FirstName	LastName	Company	Address
1	1	Luís	Gonçalves	Embraer - Empresa Brasileira de Aeronáutica S.A.	Av. Brigadeiro Faria Lima, 2170
2	2	Leonie	Köhler	[NULL]	Theodor-Heuss-Straße 34
3	3	François	Tremblay	[NULL]	1498 rue Bélanger
4	4	Bjørn	Hansen	[NULL]	Ullevålsveien 14
5	5	František	Wichterlová	JetBrains s.r.o.	Klanova 9/506
6	6	Helena	Holý	[NULL]	Rilská 3174/6
7	7	Astrid	Gruber	[NULL]	Rotenturmstraße 4, 1010 Innere Stadt
8	8	Daan	Peeters	[NULL]	Grétrystraat 63
9	9	Kara	Nielsen	[NULL]	Sønder Boulevard 51
10	10	Eduardo	Martins	Woodstock Discos	Rua Dr. Falcão Filho, 155
11	11	Alexandre	Rocha	Banco do Brasil S.A.	Av. Paulista, 2022
12	12	Roberto	Almeida	Riotur	Praça Pio X, 119
13	13	Fernanda	Ramos	[NULL]	Qe 7 Bloco G
14	14	Mark	Philips	Telus	8210 111 ST NW
15	15	Jennifer	Peterson	Rogers Canada	700 W Pender Street
16	16	Frank	Harris	Google Inc.	1600 Amphitheatre Parkway
17	17	Jack	Smith	Microsoft Corporation	1 Microsoft Way
18	18	Michelle	Brooks	[NULL]	627 Broadway
19	19	Tim	Goyer	Apple Inc.	1 Infinite Loop
20	20	Dan	Miller	[NULL]	541 Del Medio Avenue
21	21	Kathy	Chase	[NULL]	801 W 4th Street
22	22	Heather	Leacock	[NULL]	120 S Orange Ave
23	23	John	Gordon	[NULL]	69 Salem Street
24	24	Frank	Ralston	[NULL]	162 E Superior Street

PYTHON

Customers



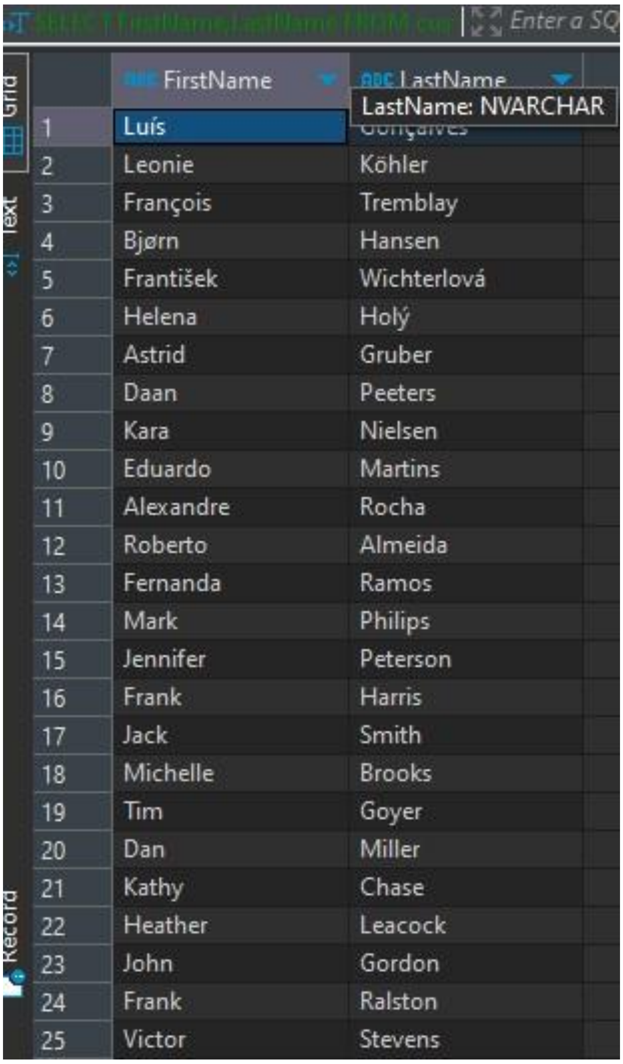
```
customers = pd.read_csv(r"D:\customers_202310181144.csv")
```

```
customers.head()
```

	CustomerId	FirstName	LastName	Company	Address	City	State	Country	PostalCode	Phone	Fax	Email	SupportRepId
0	1	Luís	Gonçalves	Embraer - Empresa Brasileira de Aeronáutica S.A.	Av. Brigadeiro Faria Lima, 2170	São José dos Campos	SP	Brazil	12227-000	+55 (12) 3923-5555	+55 (12) 3923-5566	luisg@embraer.com.br	3
1	2	Leonie	Köhler	NaN	Theodor-Heuss-Straße 34	Stuttgart	NaN	Germany	70174	+49 0711 2842222	NaN	leonekohler@surfeu.de	5
2	3	François	Tremblay	NaN	1498 rue Bélanger	Montréal	QC	Canada	H2G 1A7	+1 (514) 721-4711	NaN	ftremblay@gmail.com	3
3	4	Bjørn	Hansen	NaN	Ullevålsveien 14	Oslo	NaN	Norway	0171	+47 22 44 22 22	NaN	bjorn.hansen@yahoo.no	4
4	5	František	Wichterlová	JetBrains s.r.o.	Klanova 9/506	Prague	NaN	Czech Republic	14700	+420 2 4172 5555	+420 2 4172 5555	frantisekw@jetbrains.com	4

SQL query

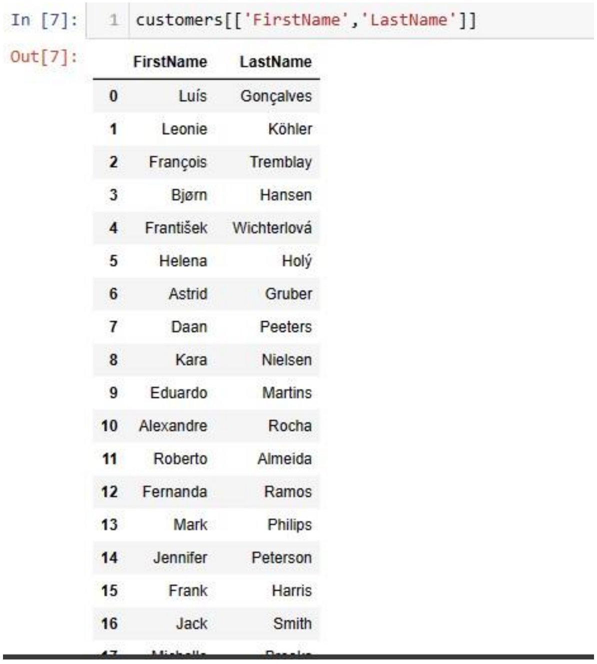
SELECT FirstName,LastName FROM customers;



	FirstName	LastName
1	Luís	Gonçalves
2	Leonie	Köhler
3	François	Tremblay
4	Bjørn	Hansen
5	František	Wichterlová
6	Helena	Holý
7	Astrid	Gruber
8	Daan	Peeters
9	Kara	Nielsen
10	Eduardo	Martins
11	Alexandre	Rocha
12	Roberto	Almeida
13	Fernanda	Ramos
14	Mark	Philips
15	Jennifer	Peterson
16	Frank	Harris
17	Jack	Smith
18	Michelle	Brooks
19	Tim	Goyer
20	Dan	Miller
21	Kathy	Chase
22	Heather	Leacock
23	John	Gordon
24	Frank	Ralston
25	Victor	Stevens

PYTHON

Customers[['FirstName','LastName']]

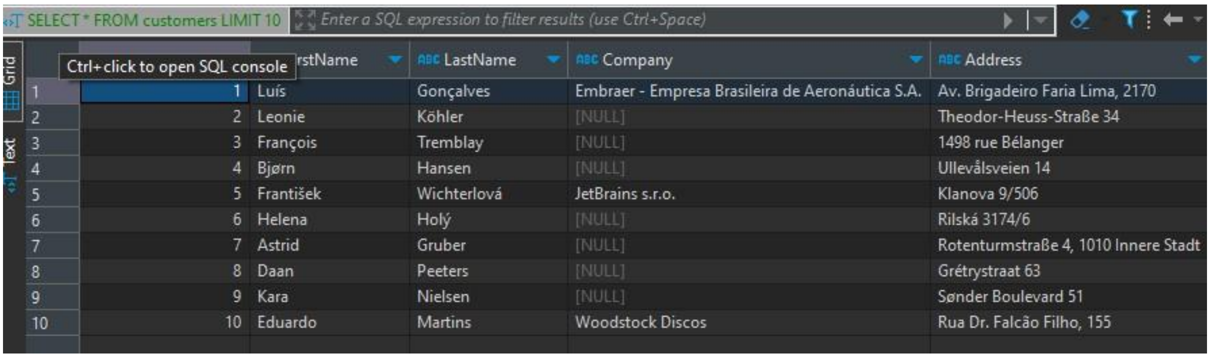


```
In [7]:  customers[['FirstName','LastName']]
Out[7]:
```

	FirstName	LastName
0	Luís	Gonçalves
1	Leonie	Köhler
2	François	Tremblay
3	Bjørn	Hansen
4	František	Wichterlová
5	Helena	Holý
6	Astrid	Gruber
7	Daan	Peeters
8	Kara	Nielsen
9	Eduardo	Martins
10	Alexandre	Rocha
11	Roberto	Almeida
12	Fernanda	Ramos
13	Mark	Philips
14	Jennifer	Peterson
15	Frank	Harris
16	Jack	Smith
17	Michelle	Brooks

SQL query

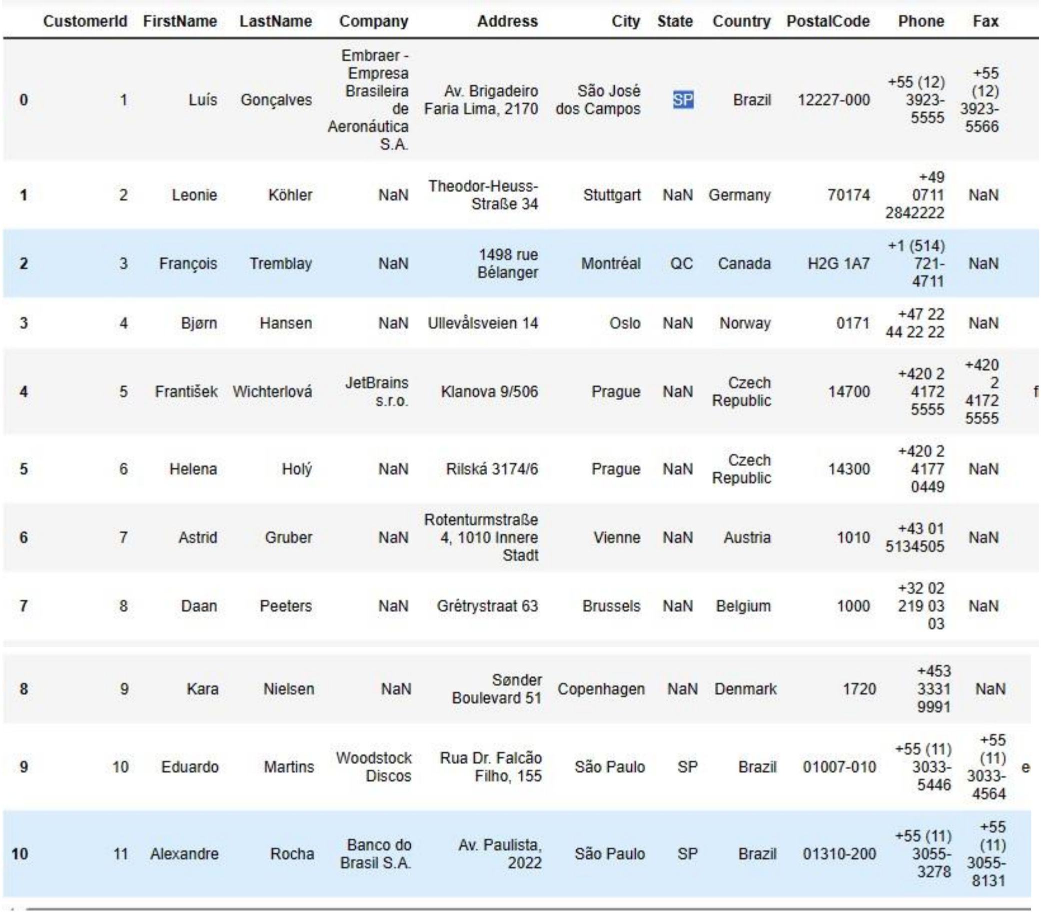
SELECT * FROM customers LIMIT 10;


SELECT * FROM customers LIMIT 10
Enter a SQL expression to filter results (use Ctrl+Space)
Grid
Text
Ctrl+click to open SQL console

		rstName	LastName	Company	Address
1	1	Luís	Gonçalves	Embraer - Empresa Brasileira de Aeronáutica S.A.	Av. Brigadeiro Faria Lima, 2170
2	2	Leonie	Köhler	[NULL]	Theodor-Heuss-Straße 34
3	3	François	Tremblay	[NULL]	1498 rue Bélanger
4	4	Bjørn	Hansen	[NULL]	Ullevålsveien 14
5	5	František	Wichterlová	JetBrains s.r.o.	Klanova 9/506
6	6	Helena	Holý	[NULL]	Rilská 3174/6
7	7	Astrid	Gruber	[NULL]	Rotenturmstraße 4, 1010 Innere Stadt
8	8	Daan	Peeters	[NULL]	Grétrystraat 63
9	9	Kara	Nielsen	[NULL]	Sønder Boulevard 51
10	10	Eduardo	Martins	Woodstock Discos	Rua Dr. Falcão Filho, 155

PYTHON

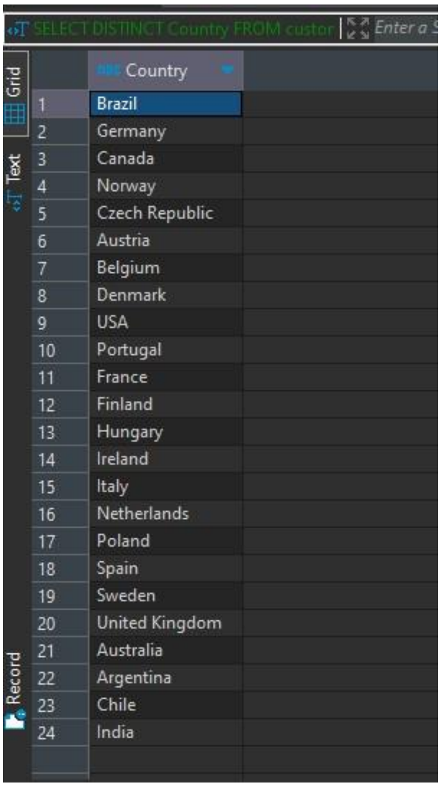
Customer[:11]



	CustomerId	FirstName	LastName	Company	Address	City	State	Country	PostalCode	Phone	Fax
0	1	Luís	Gonçalves	Embraer - Empresa Brasileira de Aeronáutica S.A.	Av. Brigadeiro Faria Lima, 2170	São José dos Campos	SP	Brazil	12227-000	+55 (12) 3923-5555	+55 (12) 3923-5566
1	2	Leonie	Köhler	NaN	Theodor-Heuss-Straße 34	Stuttgart	NaN	Germany	70174	+49 0711 2842222	NaN
2	3	François	Tremblay	NaN	1498 rue Bélanger	Montréal	QC	Canada	H2G 1A7	+1 (514) 721-4711	NaN
3	4	Bjørn	Hansen	NaN	Ullevålsveien 14	Oslo	NaN	Norway	0171	+47 22 44 22 22	NaN
4	5	František	Wichterlová	JetBrains s.r.o.	Klanova 9/506	Prague	NaN	Czech Republic	14700	+420 2 4172 5555	+420 2 4172 5555
5	6	Helena	Holý	NaN	Rilská 3174/6	Prague	NaN	Czech Republic	14300	+420 2 4177 0449	NaN
6	7	Astrid	Gruber	NaN	Rotenturmstraße 4, 1010 Innere Stadt	Vienne	NaN	Austria	1010	+43 01 5134505	NaN
7	8	Daan	Peeters	NaN	Grétrystraat 63	Brussels	NaN	Belgium	1000	+32 02 219 03 03	NaN
8	9	Kara	Nielsen	NaN	Sønder Boulevard 51	Copenhagen	NaN	Denmark	1720	+453 3331 9991	NaN
9	10	Eduardo	Martins	Woodstock Discos	Rua Dr. Falcão Filho, 155	São Paulo	SP	Brazil	01007-010	+55 (11) 3033-5446	+55 (11) 3033-4564
10	11	Alexandre	Rocha	Banco do Brasil S.A.	Av. Paulista, 2022	São Paulo	SP	Brazil	01310-200	+55 (11) 3055-3278	+55 (11) 3055-8131

SQL query

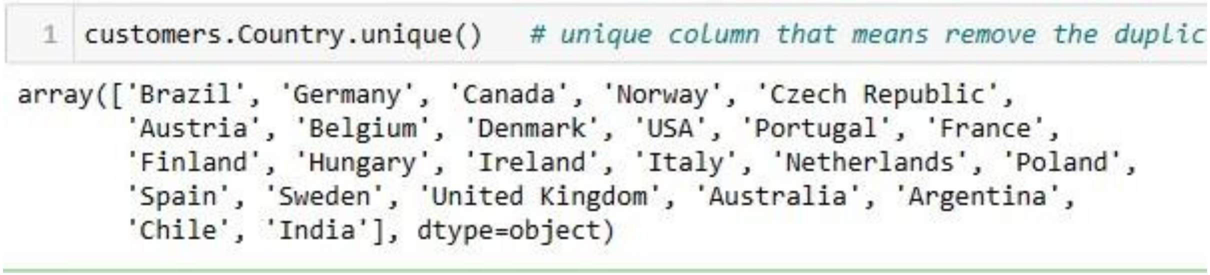
SELECT DISTINT   Country FROM customers;



	Country
1	Brazil
2	Germany
3	Canada
4	Norway
5	Czech Republic
6	Austria
7	Belgium
8	Denmark
9	USA
10	Portugal
11	France
12	Finland
13	Hungary
14	Ireland
15	Italy
16	Netherlands
17	Poland
18	Spain
19	Sweden
20	United Kingdom
21	Australia
22	Argentina
23	Chile
24	India

PYTHON

Customers.Country.unique()

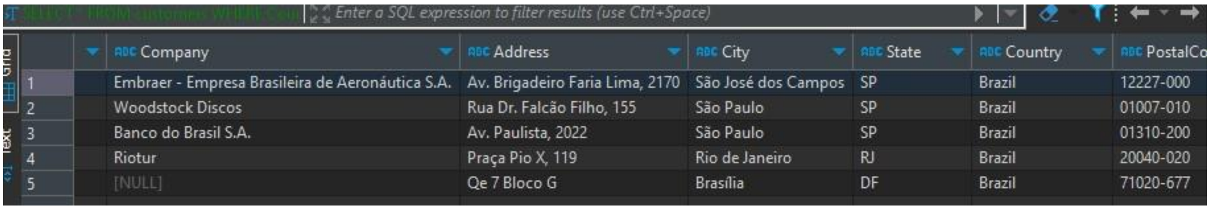


```
customers.Country.unique()    # unique column that means remove the duplic
```

```
array(['Brazil', 'Germany', 'Canada', 'Norway', 'Czech Republic',
       'Austria', 'Belgium', 'Denmark', 'USA', 'Portugal', 'France',
       'Finland', 'Hungary', 'Ireland', 'Italy', 'Netherlands', 'Poland',
       'Spain', 'Sweden', 'United Kingdom', 'Australia', 'Argentina',
       'Chile', 'India'], dtype=object)
```

SQL query

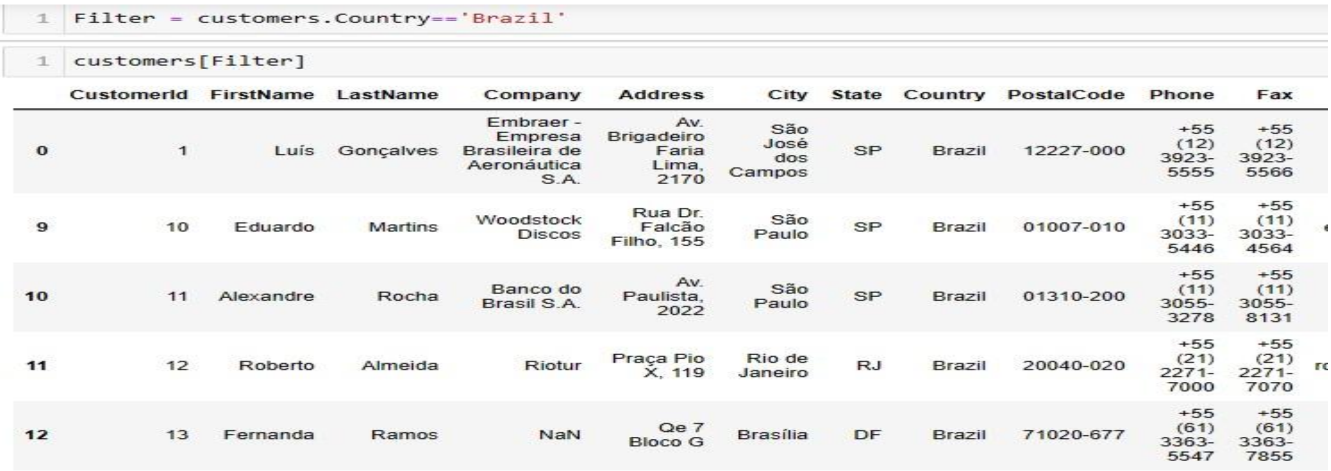
SELECT * FROM customer WHERE country =’Brazil’;


SELECT * FROM customers WHERE Coun
Enter a SQL expression to filter results (use Ctrl+Space)
Grid
Text

		Company	Address	City	State	Country	PostalCo
1		Embraer - Empresa Brasileira de Aeronáutica S.A.	Av. Brigadeiro Faria Lima, 2170	São José dos Campos	SP	Brazil	12227-000
2		Woodstock Discos	Rua Dr. Falcão Filho, 155	São Paulo	SP	Brazil	01007-010
3		Banco do Brasil S.A.	Av. Paulista, 2022	São Paulo	SP	Brazil	01310-200
4		Riotur	Praça Pio X, 119	Rio de Janeiro	RJ	Brazil	20040-020
5		[NULL]	Qe 7 Bloco G	Brasília	DF	Brazil	71020-677

PYTHON

Customers[filter]=customers.Country==’Brazil’



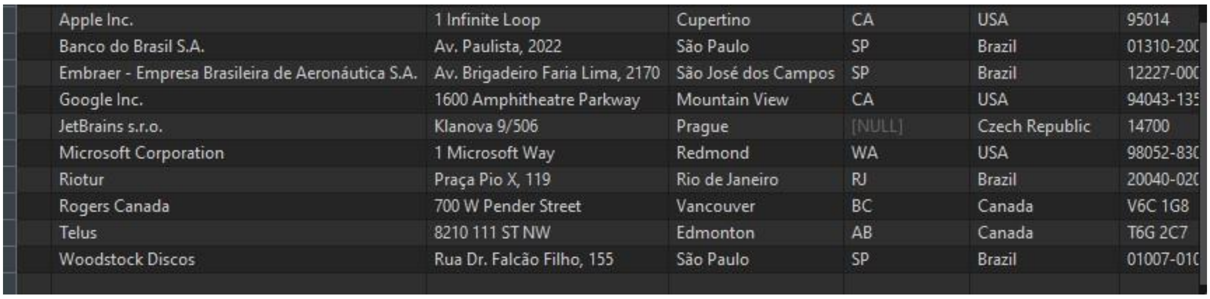
```
Filter = customers.Country=='Brazil'
```

```
customers[Filter]
```

	CustomerId	FirstName	LastName	Company	Address	City	State	Country	PostalCode	Phone	Fax
0	1	Luís	Gonçalves	Embraer - Empresa Brasileira de Aeronáutica S.A.	Av. Brigadeiro Faria Lima, 2170	São José dos Campos	SP	Brazil	12227-000	+55 (12) 3923-5555	+55 (12) 3923-5566
9	10	Eduardo	Martins	Woodstock Discos	Rua Dr. Falcão Filho, 155	São Paulo	SP	Brazil	01007-010	+55 (11) 3033-5446	+55 (11) 3033-4564
10	11	Alexandre	Rocha	Banco do Brasil S.A.	Av. Paulista, 2022	São Paulo	SP	Brazil	01310-200	+55 (11) 3055-3278	+55 (11) 3055-8131
11	12	Roberto	Almeida	Riotur	Praça Pio X, 119	Rio de Janeiro	RJ	Brazil	20040-020	+55 (21) 2271-7000	+55 (21) 2271-7070
12	13	Fernanda	Ramos	NaN	Qe 7 Bloco G	Brasília	DF	Brazil	71020-677	+55 (61) 3363-5547	+55 (61) 3363-7855

SQL query

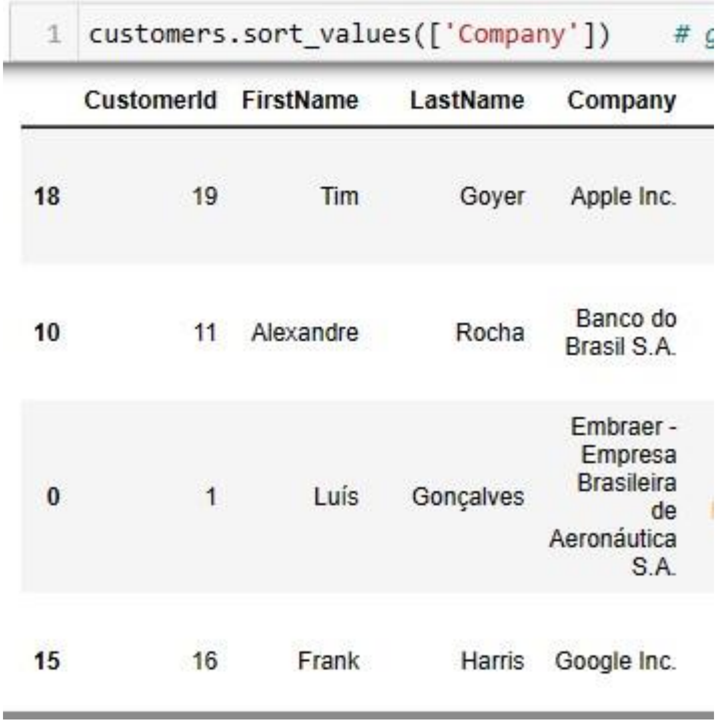
SELECT * FROM customers ORDER BY company;



Apple Inc.	1 Infinite Loop	Cupertino	CA	USA	95014
Banco do Brasil S.A.	Av. Paulista, 2022	São Paulo	SP	Brazil	01310-20
Embraer - Empresa Brasileira de Aeronáutica S.A.	Av. Brigadeiro Faria Lima, 2170	São José dos Campos	SP	Brazil	12227-00
Google Inc.	1600 Amphitheatre Parkway	Mountain View	CA	USA	94043-13
JetBrains s.r.o.	Klanova 9/506	Prague	[NULL]	Czech Republic	14700
Microsoft Corporation	1 Microsoft Way	Redmond	WA	USA	98052-83
Riotur	Praça Pio X, 119	Rio de Janeiro	RJ	Brazil	20040-02
Rogers Canada	700 W Pender Street	Vancouver	BC	Canada	V6C 1G8
Telus	8210 111 ST NW	Edmonton	AB	Canada	T6G 2C7
Woodstock Discos	Rua Dr. Falcão Filho, 155	São Paulo	SP	Brazil	01007-01

PYTHON

Customers.sort_values([Company])

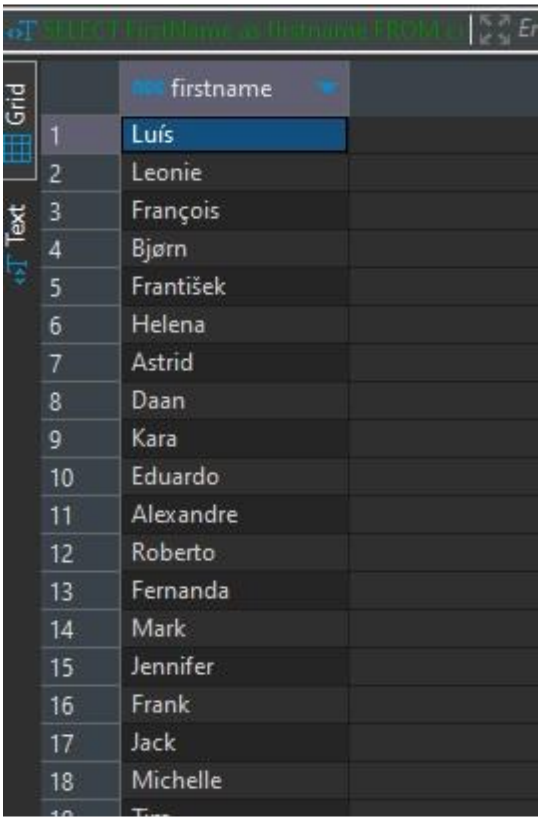


```
customers.sort_values(['Company'])    # g
```

	CustomerId	FirstName	LastName	Company
18	19	Tim	Goyer	Apple Inc.
10	11	Alexandre	Rocha	Banco do Brasil S.A.
0	1	Luís	Gonçalves	Embraer - Empresa Brasileira de Aeronáutica S.A.
15	16	Frank	Harris	Google Inc.

SQL query

SELECT Firstname as firstname FROM customers;

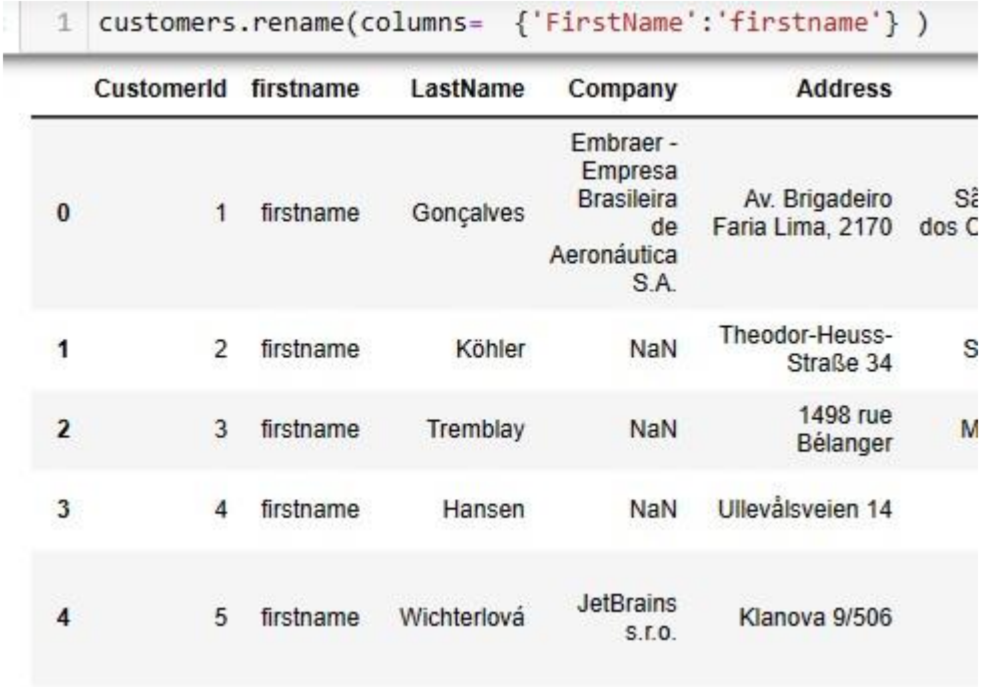


SELECT FirstName as firstname FROM cu

	firstname
1	Luís
2	Leonie
3	François
4	Bjørn
5	František
6	Helena
7	Astrid
8	Daan
9	Kara
10	Eduardo
11	Alexandre
12	Roberto
13	Fernanda
14	Mark
15	Jennifer
16	Frank
17	Jack
18	Michelle
19	Tim

PYTHON

Customers.rename(columns={FirstName' :'firstname'})

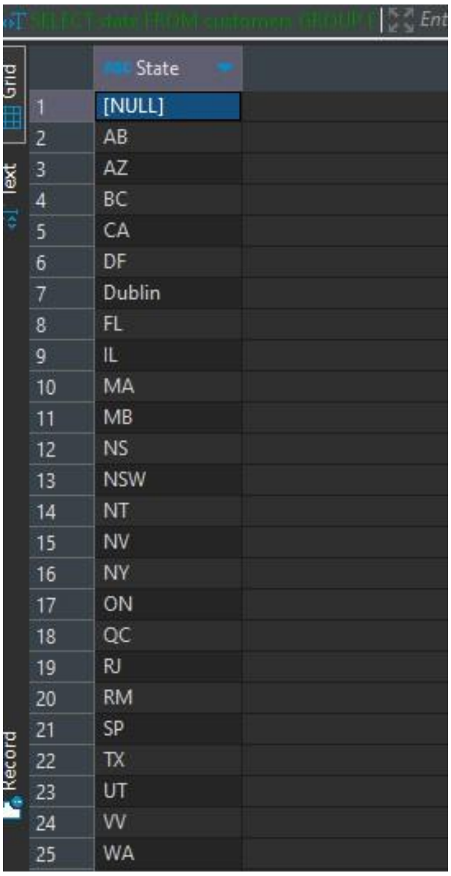


```
customers.rename(columns=  {'FirstName':'firstname'} )
```

	CustomerId	firstname	LastName	Company	Address	
0	1	firstname	Gonçalves	Embraer - Empresa Brasileira de Aeronáutica S.A.	Av. Brigadeiro Faria Lima, 2170	Sã dos C
1	2	firstname	Köhler	NaN	Theodor-Heuss-Straße 34	S
2	3	firstname	Tremblay	NaN	1498 rue Bélanger	M
3	4	firstname	Hansen	NaN	Ullevålsveien 14	
4	5	firstname	Wichterlová	JetBrains s.r.o.	Klanova 9/506	

SQL query

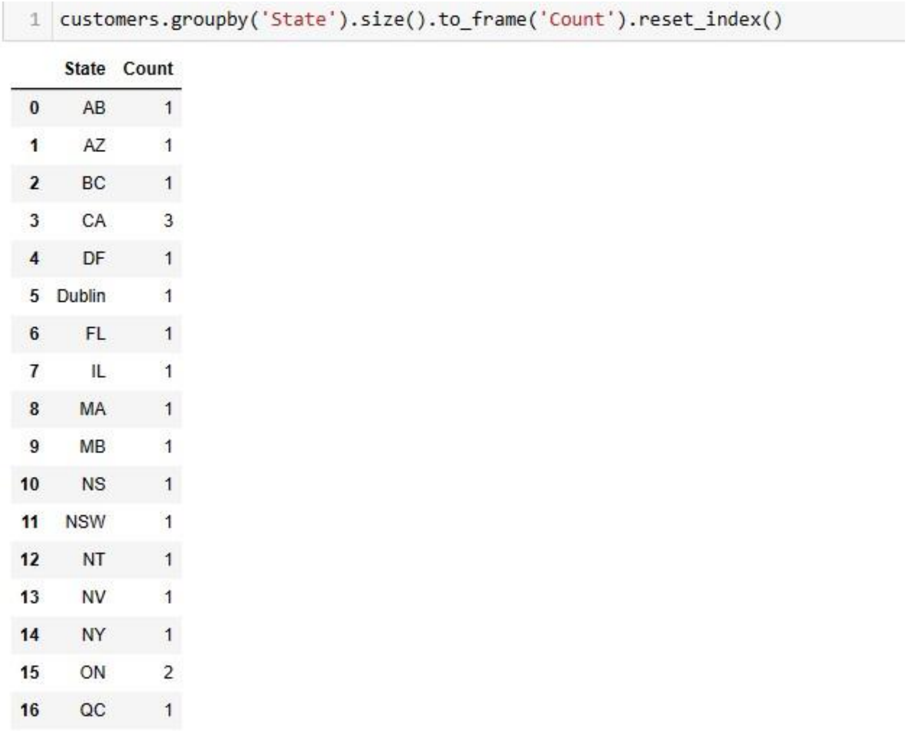
SELECT state FROM customers GROUP BY state;



	State
1	[NULL]
2	AB
3	AZ
4	BC
5	CA
6	DF
7	Dublin
8	FL
9	IL
10	MA
11	MB
12	NS
13	NSW
14	NT
15	NV
16	NY
17	ON
18	QC
19	RJ
20	RM
21	SP
22	TX
23	UT
24	VV
25	WA

PYTHON

Customers.groupby(‘state’).size().to_frame(‘count’).reset_index()

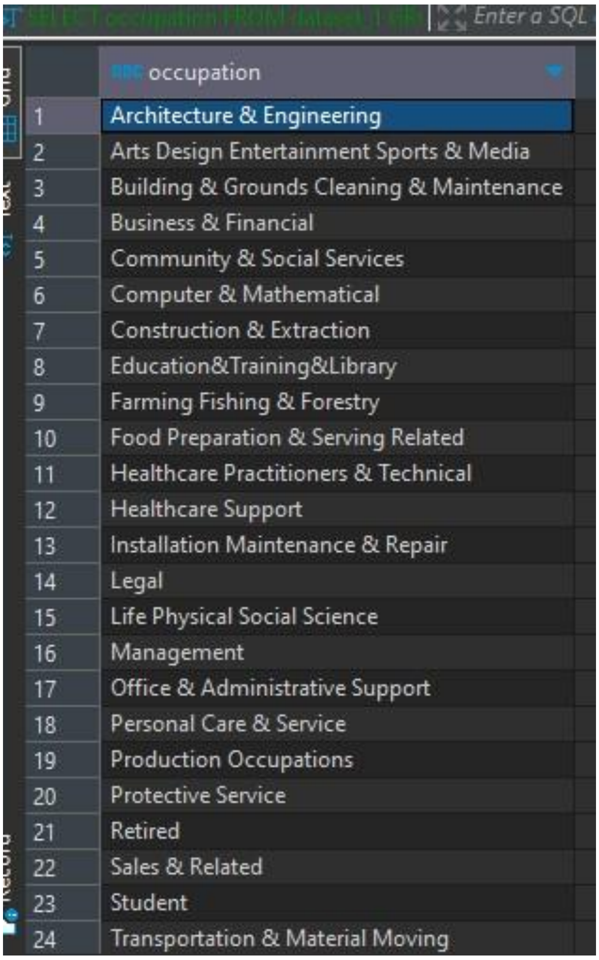


```
customers.groupby('State').size().to_frame('Count').reset_index()
```

	State	Count
0	AB	1
1	AZ	1
2	BC	1
3	CA	3
4	DF	1
5	Dublin	1
6	FL	1
7	IL	1
8	MA	1
9	MB	1
10	NS	1
11	NSW	1
12	NT	1
13	NV	1
14	NY	1
15	ON	2
16	QC	1

SQL query

SELECT occupation FROM dataset_1 GROUP BY occupation ;

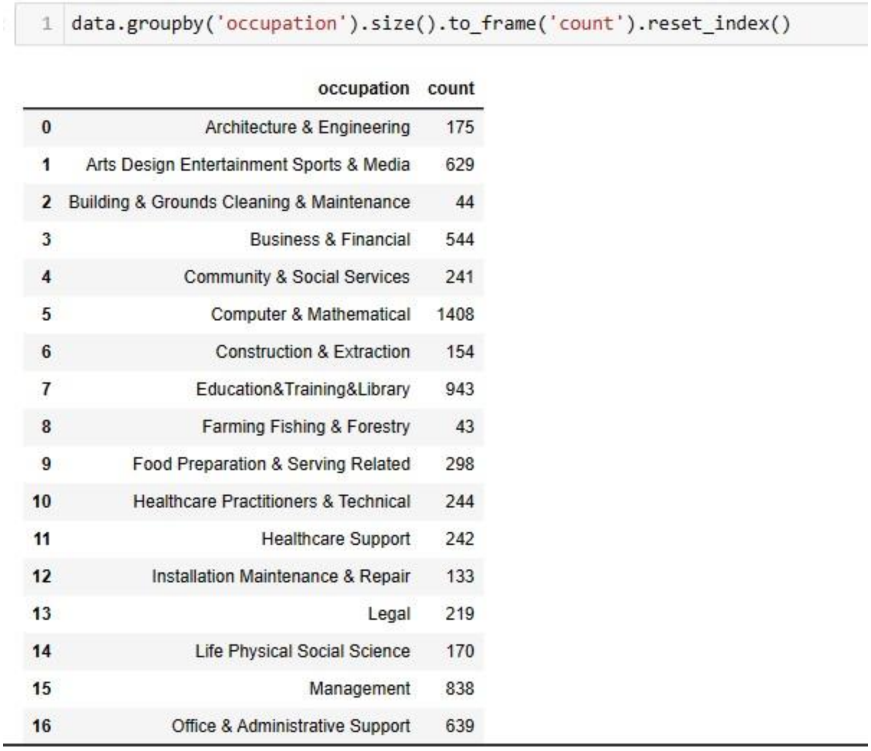


```
SELECT occupation FROM dataset_1 GR
```

	occupation
1	Architecture & Engineering
2	Arts Design Entertainment Sports & Media
3	Building & Grounds Cleaning & Maintenance
4	Business & Financial
5	Community & Social Services
6	Computer & Mathematical
7	Construction & Extraction
8	Education&Training&Library
9	Farming Fishing & Forestry
10	Food Preparation & Serving Related
11	Healthcare Practitioners & Technical
12	Healthcare Support
13	Installation Maintenance & Repair
14	Legal
15	Life Physical Social Science
16	Management
17	Office & Administrative Support
18	Personal Care & Service
19	Production Occupations
20	Protective Service
21	Retired
22	Sales & Related
23	Student
24	Transportation & Material Moving

PYTHON

data.groupby(‘occupation’).size().to_frame(‘count).reset_index()

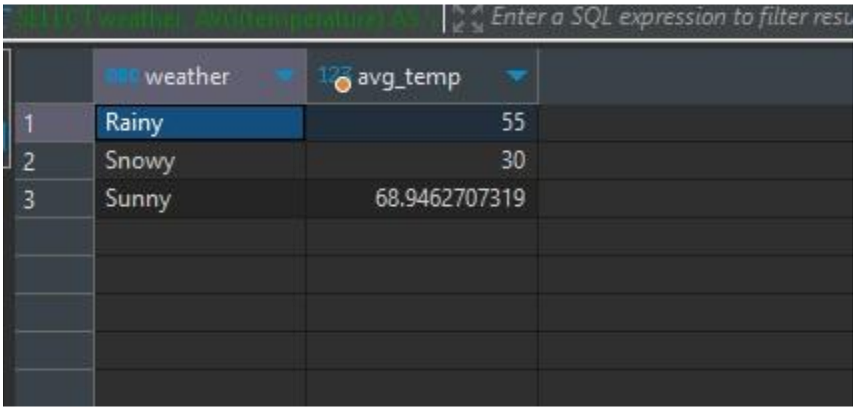


```python
data.groupby('occupation').size().to_frame('count').reset_index()
```

	occupation	count
0	Architecture & Engineering	175
1	Arts Design Entertainment Sports & Media	629
2	Building & Grounds Cleaning & Maintenance	44
3	Business & Financial	544
4	Community & Social Services	241
5	Computer & Mathematical	1408
6	Construction & Extraction	154
7	Education&Training&Library	943
8	Farming Fishing & Forestry	43
9	Food Preparation & Serving Related	298
10	Healthcare Practitioners & Technical	244
11	Healthcare Support	242
12	Installation Maintenance & Repair	133
13	Legal	219
14	Life Physical Social Science	170
15	Management	838
16	Office & Administrative Support	639

SQL query

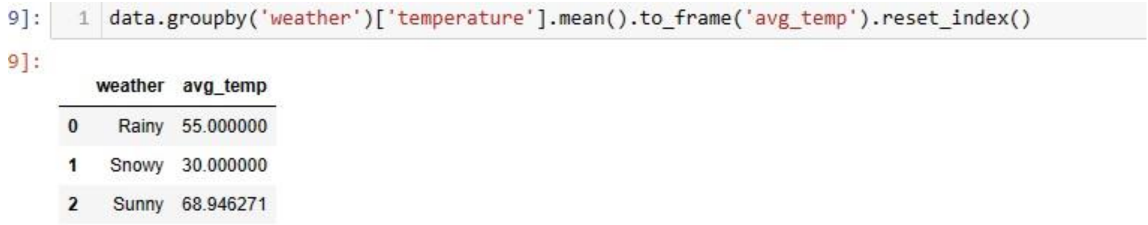
SELECT weather ,AVG(temperature) as avg_temp FROM dataset_1 GROUP BY weather;


SELECT weather, AVG(temperature) AS
Enter a SQL expression to filter resu

	weather	avg_temp
1	Rainy	55
2	Snowy	30
3	Sunny	68.9462707319

PYTHON

data.groupby(‘weather’)[‘temperature].mean().to_frame(‘avg temp’).reset_index()

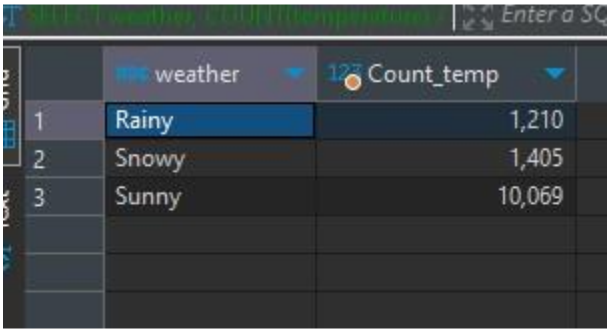


```
9]: data.groupby('weather')['temperature'].mean().to_frame('avg_temp').reset_index()
9]:
```

	weather	avg_temp
0	Rainy	55.000000
1	Snowy	30.000000
2	Sunny	68.946271

SQL query

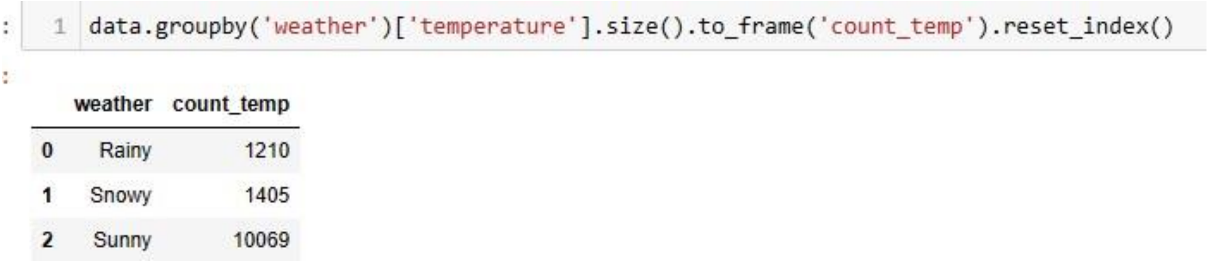
SELECT weather ,COUNT(temperature) as count_temp FROM dataset_1 GROUP BY weather;



	weather	Count_temp
1	Rainy	1,210
2	Snowy	1,405
3	Sunny	10,069

PYTHON

data.groupby(‘weather’)[‘temperature].mean().to_frame(‘avg temp’).reset_index()

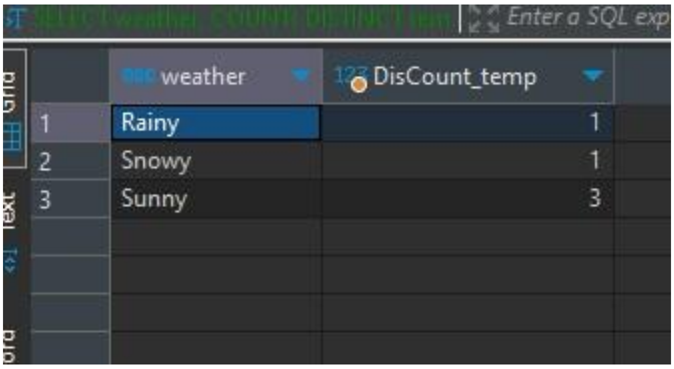


```
: data.groupby('weather')['temperature'].size().to_frame('count_temp').reset_index()
:
```

	weather	count_temp
0	Rainy	1210
1	Snowy	1405
2	Sunny	10069

SQL query

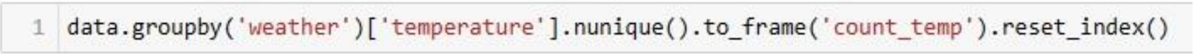
SELECT weather ,COUNT(DISTINTtemperature) as Discount_temp FROM dataset_1 GROUP BY weather;



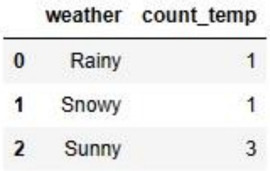
	weather	DisCount_temp
1	Rainy	1
2	Snowy	1
3	Sunny	3

PYTHON

data.groupby(‘weather’)[‘temperature].nuniqe().to_frame(‘count_temp’).reset_index()



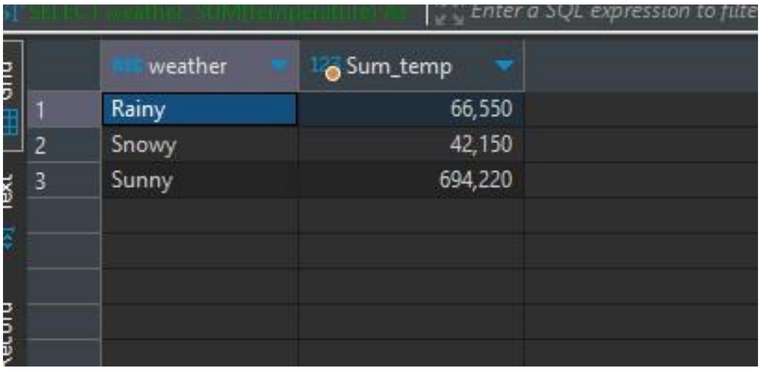
```
data.groupby('weather')['temperature'].nunique().to_frame('count_temp').reset_index()
```



	weather	count_temp
0	Rainy	1
1	Snowy	1
2	Sunny	3

SQL query

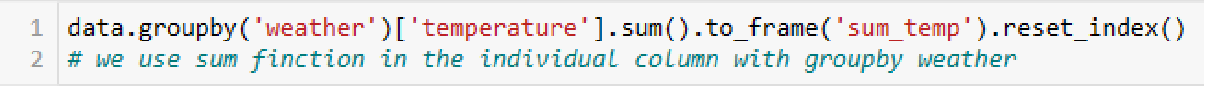
SELECT weather ,SUM(temperature) as Sum_temp FROM dataset_1 GROUP BY weather;


SELECT weather, SUM(temperature) as
Enter a SQL expression to filte
Grid
Text
Record

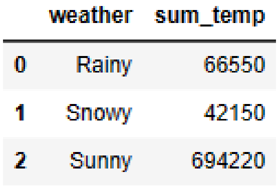
	weather	Sum_temp
1	Rainy	66,550
2	Snowy	42,150
3	Sunny	694,220

PYTHON

data.groupby(‘weather’)[‘temperature].sum().to_frame(‘sum_ temp’).reset_index()



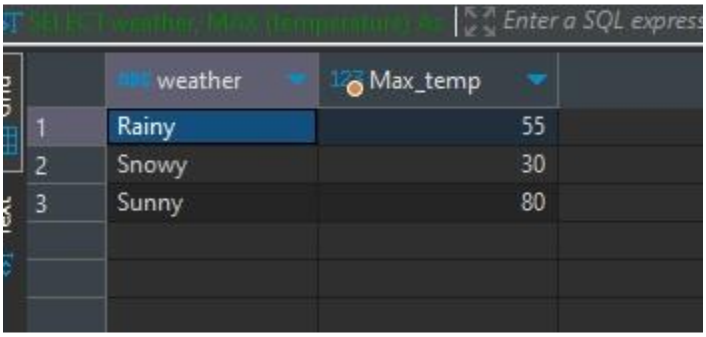
```
data.groupby('weather')['temperature'].sum().to_frame('sum_temp').reset_index()
# we use sum fiction in the individual column with groupby weather
```



	weather	sum_temp
0	Rainy	66550
1	Snowy	42150
2	Sunny	694220

SQL query

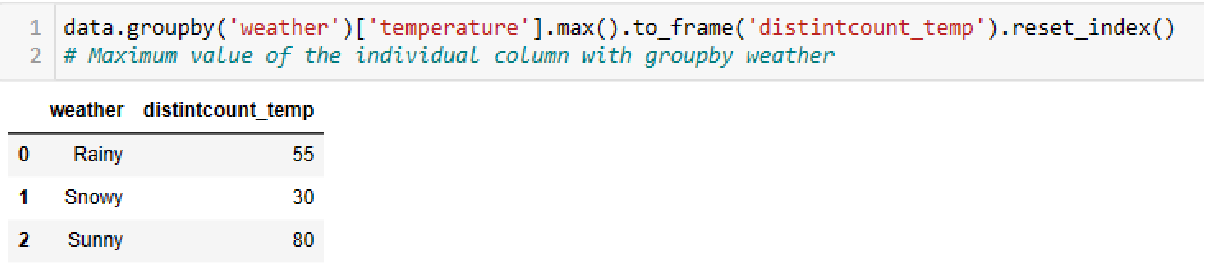
SELECT weather ,MAX(temperature) as Max_temp FROM dataset_1 GROUP BY weather;



	weather	Max_temp
1	Rainy	55
2	Snowy	30
3	Sunny	80

PYTHON

data.groupby(‘weather’)[‘temperature].max().to_frame(‘max_ temp’).reset_index()

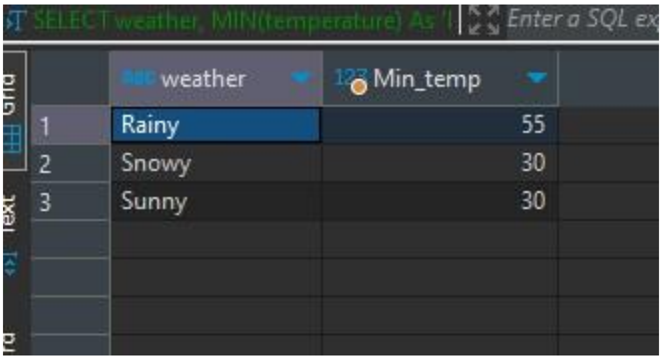


```python
data.groupby('weather')['temperature'].max().to_frame('distintcount_temp').reset_index()
# Maximum value of the individual column with groupby weather
```

	weather	distintcount_temp
0	Rainy	55
1	Snowy	30
2	Sunny	80

SQL query

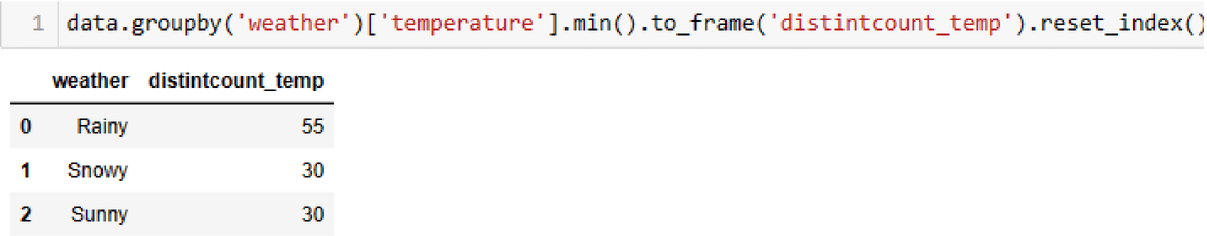
SELECT weather ,MIN(temperature) as Min_temp FROM dataset_1 GROUP BY weather;



	weather	Min_temp
1	Rainy	55
2	Snowy	30
3	Sunny	30

PYTHON

data.groupby(‘weather’)[‘temperature].min().to_frame(‘min_ temp’).reset_index()

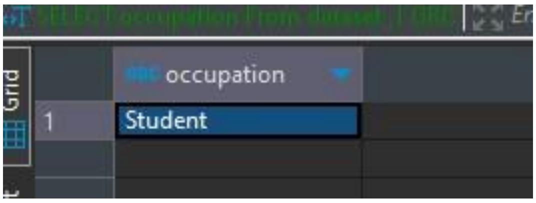


```python
data.groupby('weather')['temperature'].min().to_frame('distintcount_temp').reset_index()
```

	weather	distintcount_temp
0	Rainy	55
1	Snowy	30
2	Sunny	30

SQL query

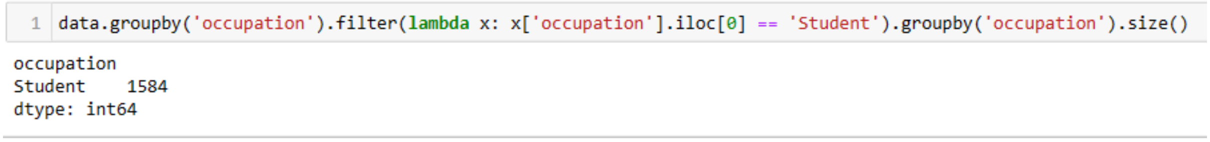
SELECT occupation FROM dataset_1 GROUP BY occupation HAVING occupation='Student';



	occupation
1	Student

PYTHON

data.groupby(‘occupation’).filter(lambda x:x[‘occupation’].iloc[0]==’student‘.groupby(‘occupation’).size()

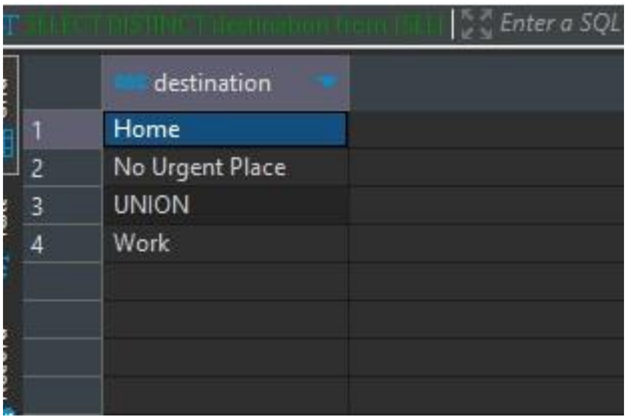


```
data.groupby('occupation').filter(lambda x: x['occupation'].iloc[0] == 'Student').groupby('occupation').size()
```

```
occupation
Student     1584
dtype: int64
```

SQL query

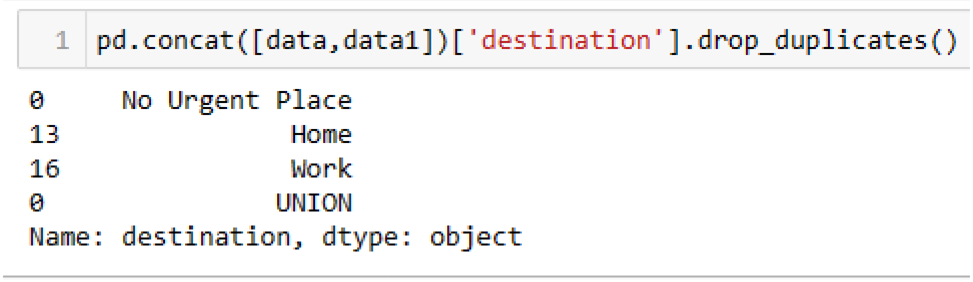
SELECT DISTINCT destination FROM(SELECT * FROM dataset_1 UNION SELECT * FROM table_to_union;



	destination
1	Home
2	No Urgent Place
3	UNION
4	Work

PYTHON

pd.concat([data,data1])[‘destination’].drop_duplicates()

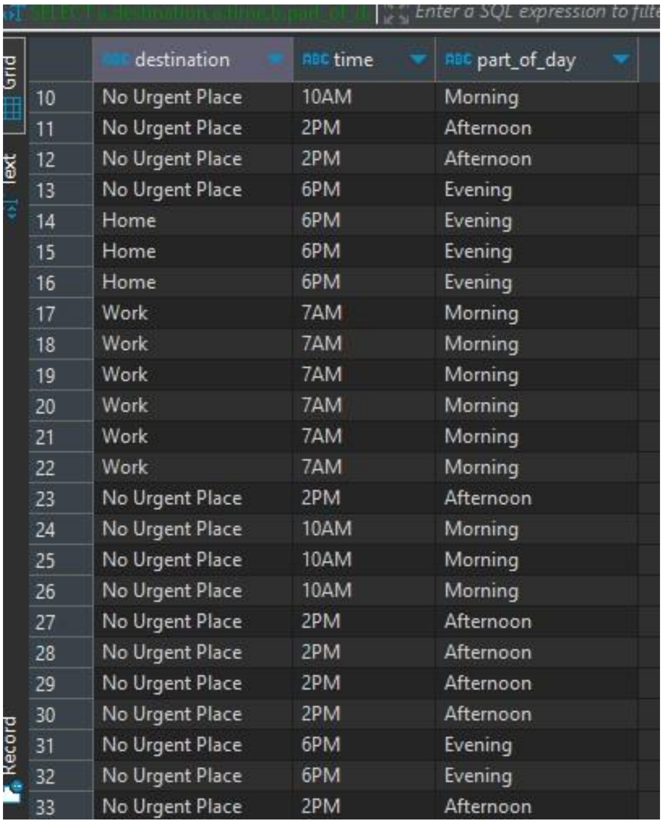


```
pd.concat([data,data1])['destination'].drop_duplicates()
```

```
0     No Urgent Place
13               Home
16               Work
0               UNION
Name: destination, dtype: object
```

SQL query

SELECT a.destination,a.time,b.part_of_day FROM dataset_1 a INNER JOIN table_to_join b ON a.time=b.time ;

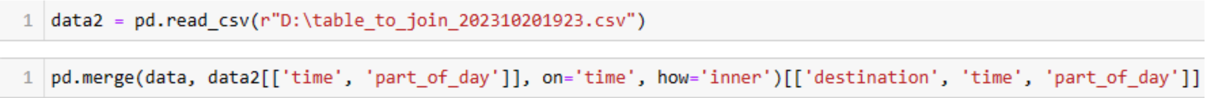


SELECT a.destination,a.time,b.part_of_d
Enter a SQL expression to filt
Grid
Text
Record

	destination	time	part_of_day
10	No Urgent Place	10AM	Morning
11	No Urgent Place	2PM	Afternoon
12	No Urgent Place	2PM	Afternoon
13	No Urgent Place	6PM	Evening
14	Home	6PM	Evening
15	Home	6PM	Evening
16	Home	6PM	Evening
17	Work	7AM	Morning
18	Work	7AM	Morning
19	Work	7AM	Morning
20	Work	7AM	Morning
21	Work	7AM	Morning
22	Work	7AM	Morning
23	No Urgent Place	2PM	Afternoon
24	No Urgent Place	10AM	Morning
25	No Urgent Place	10AM	Morning
26	No Urgent Place	10AM	Morning
27	No Urgent Place	2PM	Afternoon
28	No Urgent Place	2PM	Afternoon
29	No Urgent Place	2PM	Afternoon
30	No Urgent Place	2PM	Afternoon
31	No Urgent Place	6PM	Evening
32	No Urgent Place	6PM	Evening
33	No Urgent Place	2PM	Afternoon

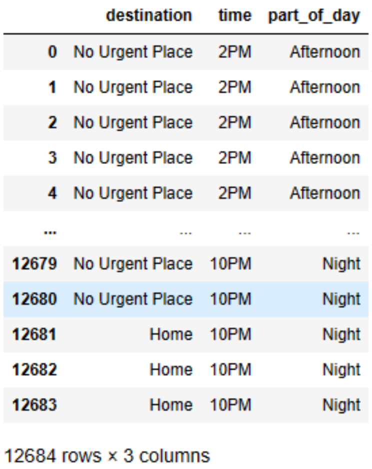
PYTHON

pd.merge(data,data2[[‘time’,’part_of_day’]],on=’time,how=’inner’)[[‘destination’,’time’,’part_of_day’]]



```
data2 = pd.read_csv(r"D:\table_to_join_202310201923.csv")
```

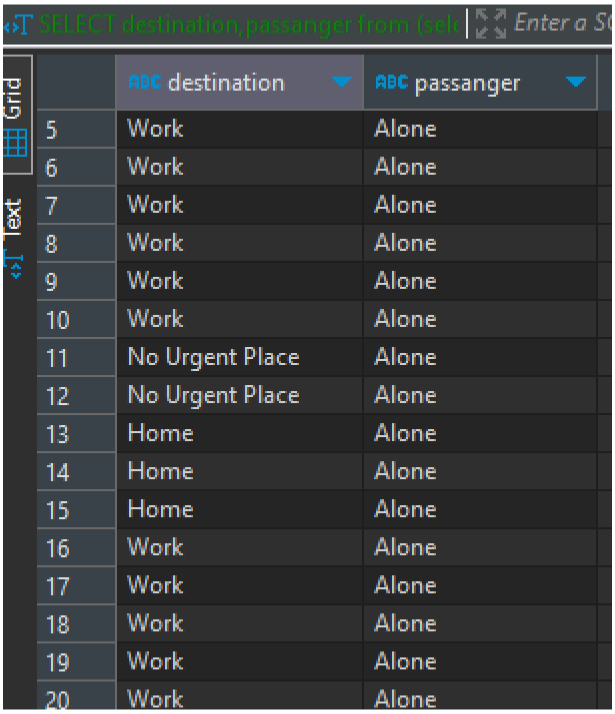
```
pd.merge(data, data2[['time', 'part_of_day']], on='time', how='inner')[['destination', 'time', 'part_of_day']]
```



```
           destination   time  part_of_day
    0  No Urgent Place    2PM    Afternoon
    1  No Urgent Place    2PM    Afternoon
    2  No Urgent Place    2PM    Afternoon
    3  No Urgent Place    2PM    Afternoon
    4  No Urgent Place    2PM    Afternoon
  ...              ...    ...          ...
12679  No Urgent Place   10PM        Night
12680  No Urgent Place   10PM        Night
12681             Home   10PM        Night
12682             Home   10PM        Night
12683             Home   10PM        Night

12684 rows × 3 columns
```

SQL query

SELECT destination ,passenger FROM(SELECT*FROM dataset_1 WHERE passenger = 'Alone');

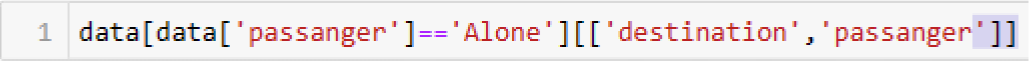


SELECT destination,passanger from (sele

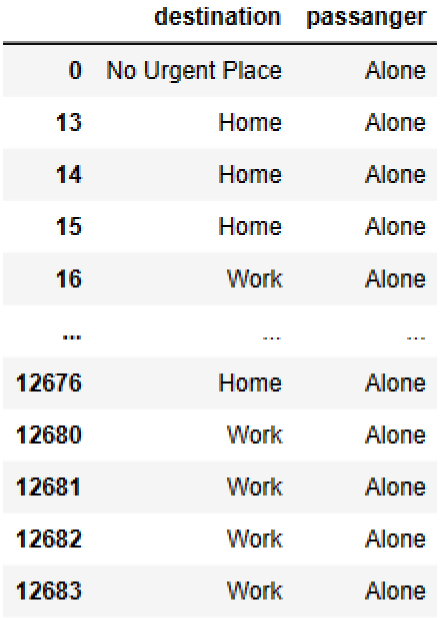
	destination	passanger
5	Work	Alone
6	Work	Alone
7	Work	Alone
8	Work	Alone
9	Work	Alone
10	Work	Alone
11	No Urgent Place	Alone
12	No Urgent Place	Alone
13	Home	Alone
14	Home	Alone
15	Home	Alone
16	Work	Alone
17	Work	Alone
18	Work	Alone
19	Work	Alone
20	Work	Alone

PYTHON

data[data[‘passanger’]==’Alone’][[‘destination’,’passanger’]]



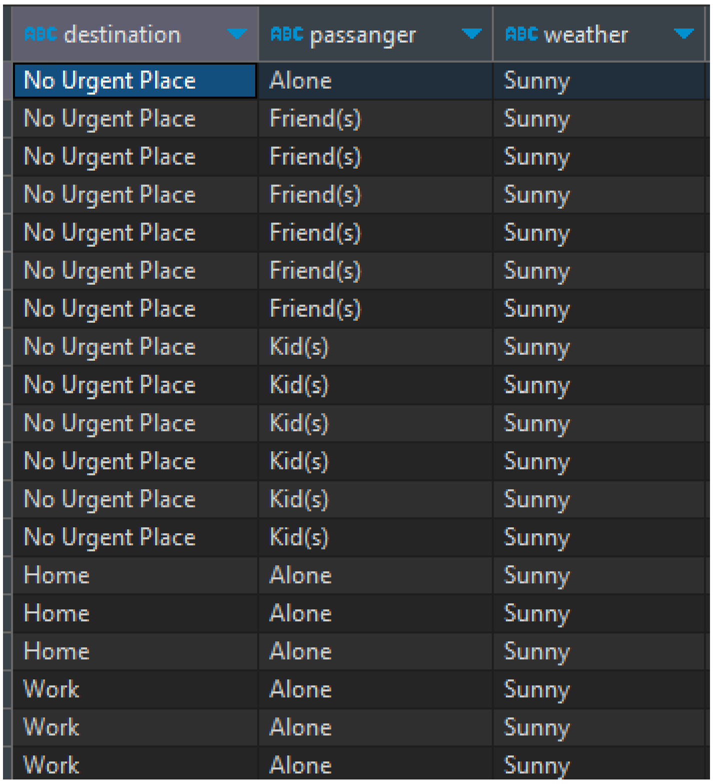
```
data[data['passanger']=='Alone'][['destination','passanger']]
```



	destination	passanger
0	No Urgent Place	Alone
13	Home	Alone
14	Home	Alone
15	Home	Alone
16	Work	Alone
...	...	...
12676	Home	Alone
12680	Work	Alone
12681	Work	Alone
12682	Work	Alone
12683	Work	Alone

SQL query

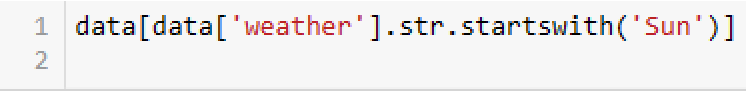
SELECT * FROM dataset_1 WHERE weather LIKE 'Sun%';



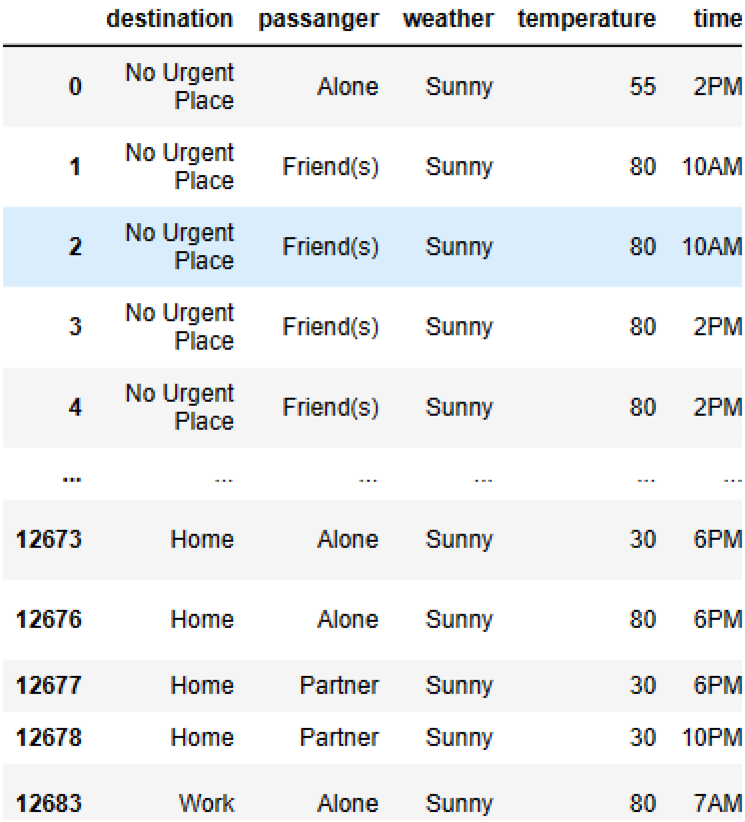
destination	passanger	weather
No Urgent Place	Alone	Sunny
No Urgent Place	Friend(s)	Sunny
No Urgent Place	Friend(s)	Sunny
No Urgent Place	Friend(s)	Sunny
No Urgent Place	Friend(s)	Sunny
No Urgent Place	Friend(s)	Sunny
No Urgent Place	Friend(s)	Sunny
No Urgent Place	Kid(s)	Sunny
No Urgent Place	Kid(s)	Sunny
No Urgent Place	Kid(s)	Sunny
No Urgent Place	Kid(s)	Sunny
No Urgent Place	Kid(s)	Sunny
No Urgent Place	Kid(s)	Sunny
Home	Alone	Sunny
Home	Alone	Sunny
Home	Alone	Sunny
Work	Alone	Sunny
Work	Alone	Sunny
Work	Alone	Sunny

PYTHON

data[data[‘weather’].str.starswith(‘sun’)]



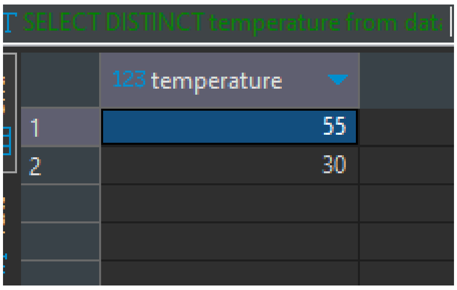
```
data[data['weather'].str.startswith('Sun')]

```



	destination	passanger	weather	temperature	time
0	No Urgent Place	Alone	Sunny	55	2PM
1	No Urgent Place	Friend(s)	Sunny	80	10AM
2	No Urgent Place	Friend(s)	Sunny	80	10AM
3	No Urgent Place	Friend(s)	Sunny	80	2PM
4	No Urgent Place	Friend(s)	Sunny	80	2PM
...	...	...	...	...	...
12673	Home	Alone	Sunny	30	6PM
12676	Home	Alone	Sunny	80	6PM
12677	Home	Partner	Sunny	30	6PM
12678	Home	Partner	Sunny	30	10PM
12683	Work	Alone	Sunny	80	7AM

SQL query

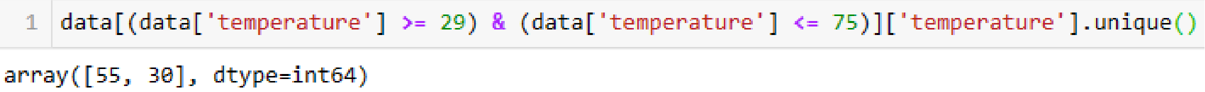
SELECT DISTINT temperature FROM dataset_1 temperature BETWEEN 29 AND 75;



	temperature
1	55
2	30

PYTHON

data[(data[‘temperature’]>29&(data[‘temperature’]<=75)][‘temperature’].unique()

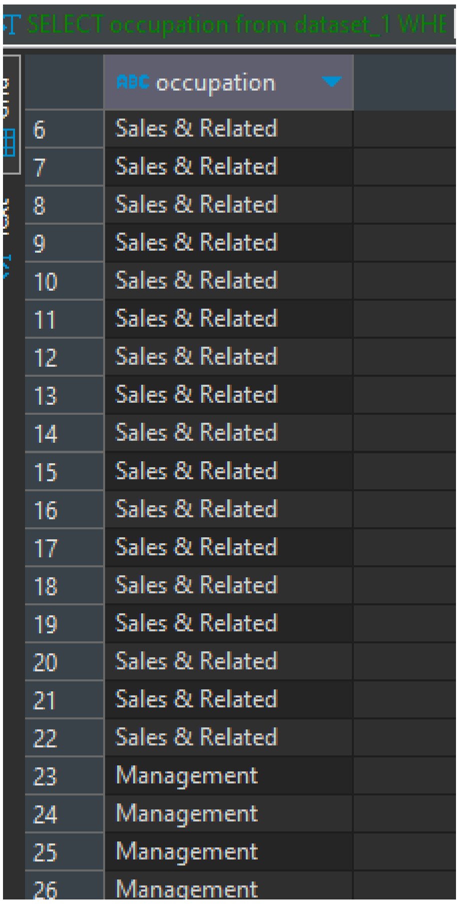


```
data[(data['temperature'] >= 29) & (data['temperature'] <= 75)]['temperature'].unique()
```

```
array([55, 30], dtype=int64)
```

SQL query

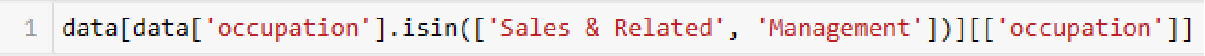
SELECT occupation FROM dataset_1 WHERE occupation IN('Sales & Related','Management';



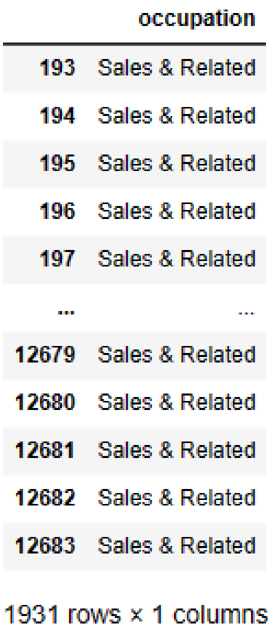
	occupation
6	Sales & Related
7	Sales & Related
8	Sales & Related
9	Sales & Related
10	Sales & Related
11	Sales & Related
12	Sales & Related
13	Sales & Related
14	Sales & Related
15	Sales & Related
16	Sales & Related
17	Sales & Related
18	Sales & Related
19	Sales & Related
20	Sales & Related
21	Sales & Related
22	Sales & Related
23	Management
24	Management
25	Management
26	Management

PYTHON

data[data[‘occupation’].isin([‘sales&Related’,’Management’])][[‘occupation’]]



```
data[data['occupation'].isin(['Sales & Related', 'Management'])][['occupation']]
```



```
            occupation
  193  Sales & Related
  194  Sales & Related
  195  Sales & Related
  196  Sales & Related
  197  Sales & Related
  ...              ...
12679  Sales & Related
12680  Sales & Related
12681  Sales & Related
12682  Sales & Related
12683  Sales & Related

1931 rows × 1 columns
```