

## SQL Query

**SELECT\*FROM dataset\_1**

Result:

	destination	passanger	weather	temperature	time	coupon	expiration	gender	age	maritalStatus
1	No Urgent Place	Alone	Sunny	55	2PM	Restaurant(<20)	1d	Female	21	Unmarried
2	No Urgent Place	Friend(s)	Sunny	80	10AM	Coffee House	2h	Female	21	Unmarried
3	No Urgent Place	Friend(s)	Sunny	80	10AM	Carry out & Take away	2h	Female	21	Unmarried
4	No Urgent Place	Friend(s)	Sunny	80	2PM	Coffee House	2h	Female	21	Unmarried
5	No Urgent Place	Friend(s)	Sunny	80	2PM	Coffee House	1d	Female	21	Unmarried
6	No Urgent Place	Friend(s)	Sunny	80	6PM	Restaurant(<20)	2h	Female	21	Unmarried
7	No Urgent Place	Friend(s)	Sunny	55	2PM	Carry out & Take away	1d	Female	21	Unmarried
8	No Urgent Place	Kid(s)	Sunny	80	10AM	Restaurant(<20)	2h	Female	21	Unmarried
9	No Urgent Place	Kid(s)	Sunny	80	10AM	Carry out & Take away	2h	Female	21	Unmarried
10	No Urgent Place	Kid(s)	Sunny	80	10AM	Bar	1d	Female	21	Unmarried
11	No Urgent Place	Kid(s)	Sunny	80	2PM	Restaurant(<20)	1d	Female	21	Unmarried
12	No Urgent Place	Kid(s)	Sunny	55	2PM	Restaurant(<20)	1d	Female	21	Unmarried
13	No Urgent Place	Kid(s)	Sunny	55	6PM	Coffee House	2h	Female	21	Unmarried
14	Home	Alone	Sunny	55	6PM	Bar	1d	Female	21	Unmarried
15	Home	Alone	Sunny	55	6PM	Restaurant(20-50)	1d	Female	21	Unmarried
16	Home	Alone	Sunny	80	6PM	Coffee House	2h	Female	21	Unmarried
17	Work	Alone	Sunny	55	7AM	Coffee House	2h	Female	21	Unmarried

## Python code

df

In [34]:

# SELECT\*FROM dataset\_1

df

Out[34]:

	destination	passanger	weather	temperature	time	coupon	expiration	gender	age	maritalStatus	—	CarryAway	RestaurantLessThan20	R
0	No Urgent Place	Alone	Sunny	55	2PM	Restaurant(<20)	1d	Female	21	Unmarried partner	—	NaN	4-8	
1	No Urgent Place	Friend(s)	Sunny	80	10AM	Coffee House	2h	Female	21	Unmarried partner	—	NaN	4-8	
2	No Urgent Place	Friend(s)	Sunny	80	10AM	Carry out & Take away	2h	Female	21	Unmarried partner	—	NaN	4-8	
3	No Urgent Place	Friend(s)	Sunny	80	2PM	Coffee House	2h	Female	21	Unmarried partner	—	NaN	4-8	
4	No Urgent Place	Friend(s)	Sunny	80	2PM	Coffee House	1d	Female	21	Unmarried partner	—	NaN	4-8	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
12679	Home	Partner	Rainy	55	6PM	Carry out & Take away	1d	Male	26	Single	—	1-3	4-8	
12680	Work	Alone	Rainy	55	7AM	Carry out & Take away	1d	Male	26	Single	—	1-3	4-8	

## SQL query

**SELECT weather,temperature FROM dataset\_1**

Result:

dataset_1 1 ×		
SELECT weather,temperature FROM dataset_1		
	weather	temperature
1	Sunny	55
2	Sunny	80
3	Sunny	80
4	Sunny	80
5	Sunny	80
6	Sunny	80
7	Sunny	55
8	Sunny	80
9	Sunny	80
10	Sunny	80
11	Sunny	80
12	Sunny	55
13	Sunny	55
14	Sunny	55
15	Sunny	55
16	Sunny	80
17	Sunny	55

## Python Code

**df[['weather','temperature']]**

Result:

```
In [4]: # SELECT weather, temperature FROM dataset_1
df[['weather', 'temperature']]
```

```
Out[4]:
```

	weather	temperature
0	Sunny	55
1	Sunny	80
2	Sunny	80
3	Sunny	80
4	Sunny	80
...	...	...
12679	Rainy	55
12680	Rainy	55
12681	Snowy	30
12682	Snowy	30
12683	Sunny	80

12684 rows × 2 columns

## SQL query

**SELECT\*FROM dataset\_1 LIMIT 10**

Result:

dataset_1 1										
SQL SELECT*FROM dataset_1 LIMIT 10 Enter a SQL expression to filter results (use Ctrl+Space)										
	destination	passenger	weather	temperature	time	coupon	expiration	gender	age	
1	No Urgent Place	Alone	Sunny	55	2PM	Restaurant(<20)	1d	Female	21	
2	No Urgent Place	Friend(s)	Sunny	80	10AM	Coffee House	2h	Female	21	
3	No Urgent Place	Friend(s)	Sunny	80	10AM	Carry out & Take away	2h	Female	21	
4	No Urgent Place	Friend(s)	Sunny	80	2PM	Coffee House	2h	Female	21	
5	No Urgent Place	Friend(s)	Sunny	80	2PM	Coffee House	1d	Female	21	
6	No Urgent Place	Friend(s)	Sunny	80	6PM	Restaurant(<20)	2h	Female	21	
7	No Urgent Place	Friend(s)	Sunny	55	2PM	Carry out & Take away	1d	Female	21	
8	No Urgent Place	Kid(s)	Sunny	80	10AM	Restaurant(<20)	2h	Female	21	
9	No Urgent Place	Kid(s)	Sunny	80	10AM	Carry out & Take away	2h	Female	21	
10	No Urgent Place	Kid(s)	Sunny	80	10AM	Bar	1d	Female	21	

## Python Code

`df.head(10)`

result:

```
In [5]: # SELECT*FROM dataset_1 LIMIT 10
df.head(10)
```

Out[5]:

	destination	passanger	weather	temperature	time	coupon	expiration	gender	age	maritalStatus	...	CarryAway	RestaurantLessThan20	Rest
0	No Urgent Place	Alone	Sunny	55	2PM	Restaurant(<20)	1d	Female	21	Unmarried partner	...	NaN		4-8
1	No Urgent Place	Friend(s)	Sunny	80	10AM	Coffee House	2h	Female	21	Unmarried partner	...	NaN		4-8
2	No Urgent Place	Friend(s)	Sunny	80	10AM	Carry out & Take away	2h	Female	21	Unmarried partner	...	NaN		4-8
3	No Urgent Place	Friend(s)	Sunny	80	2PM	Coffee House	2h	Female	21	Unmarried partner	...	NaN		4-8
4	No Urgent Place	Friend(s)	Sunny	80	2PM	Coffee House	1d	Female	21	Unmarried partner	...	NaN		4-8
5	No Urgent Place	Friend(s)	Sunny	80	6PM	Restaurant(<20)	2h	Female	21	Unmarried partner	...	NaN		4-8
6	No Urgent Place	Friend(s)	Sunny	55	2PM	Carry out & Take away	1d	Female	21	Unmarried partner	...	NaN		4-8
7	No Urgent Place	Kid(s)	Sunny	80	10AM	Restaurant(<20)	2h	Female	21	Unmarried partner	...	NaN		4-8
8	No Urgent Place	Kid(s)	Sunny	80	10AM	Carry out & Take away	2h	Female	21	Unmarried partner	...	NaN		4-8
9	No Urgent Place	Kid(s)	Sunny	80	10AM	Bar	1d	Female	21	Unmarried partner	...	NaN		4-8

10 rows x 27 columns

## SQL query

`SELECT DISTINCT passanger FROM dataset_1`

Result:

dataset_1 1 x	
SELECT DISTINCT passanger FROM dataset_1	
Grid	ABC passanger
1	Alone
2	Friend(s)
3	Kid(s)
4	Partner

## Python Code

```
df['passanger'].unique()
```

Result:

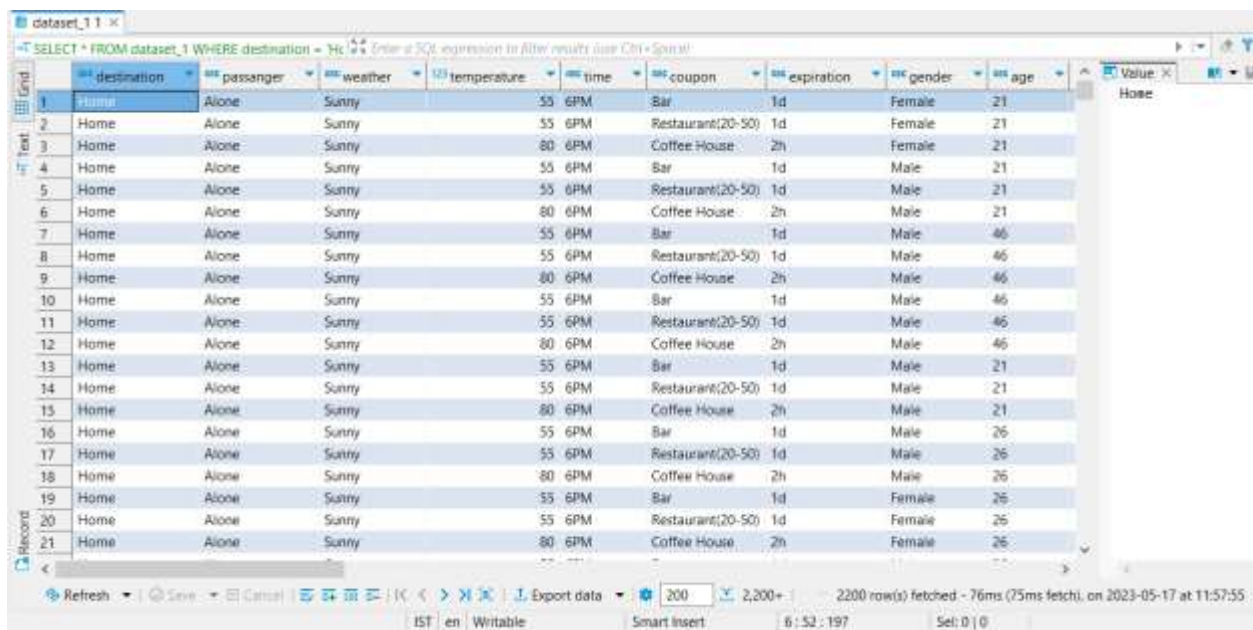
```
In [7]: # SELECT DISTINCT passenger FROM dataset_1
df['passanger'].unique()

Out[7]: array(['Alone', 'Friend(s)', 'Kid(s)', 'Partner'], dtype=object)
```

## SQL query

```
SELECT * FROM dataset_1 WHERE destination = 'Home'
```

Result:



	destination	passanger	weather	temperature	time	coupon	expiration	gender	age
1	Home	Alone	Sunny	55	6PM	Bar	1d	Female	21
2	Home	Alone	Sunny	55	6PM	Restaurant(20-50)	1d	Female	21
3	Home	Alone	Sunny	80	6PM	Coffee House	2h	Female	21
4	Home	Alone	Sunny	55	6PM	Bar	1d	Male	21
5	Home	Alone	Sunny	55	6PM	Restaurant(20-50)	1d	Male	21
6	Home	Alone	Sunny	80	6PM	Coffee House	2h	Male	21
7	Home	Alone	Sunny	55	6PM	Bar	1d	Male	46
8	Home	Alone	Sunny	55	6PM	Restaurant(20-50)	1d	Male	46
9	Home	Alone	Sunny	80	6PM	Coffee House	2h	Male	46
10	Home	Alone	Sunny	55	6PM	Bar	1d	Male	46
11	Home	Alone	Sunny	55	6PM	Restaurant(20-50)	1d	Male	46
12	Home	Alone	Sunny	80	6PM	Coffee House	2h	Male	46
13	Home	Alone	Sunny	55	6PM	Bar	1d	Male	21
14	Home	Alone	Sunny	55	6PM	Restaurant(20-50)	1d	Male	21
15	Home	Alone	Sunny	80	6PM	Coffee House	2h	Male	21
16	Home	Alone	Sunny	55	6PM	Bar	1d	Male	26
17	Home	Alone	Sunny	55	6PM	Restaurant(20-50)	1d	Male	26
18	Home	Alone	Sunny	80	6PM	Coffee House	2h	Male	26
19	Home	Alone	Sunny	55	6PM	Bar	1d	Female	26
20	Home	Alone	Sunny	55	6PM	Restaurant(20-50)	1d	Female	26
21	Home	Alone	Sunny	80	6PM	Coffee House	2h	Female	26

## Python Code

```
df[df['destination']=='Home']
```

Result:

```
In [14]: # SELECT * FROM dataset_1 WHERE destination = 'Home'
df[df['destination']=='Home']
```

Out[14]:

	destination	passanger	weather	temperature	time	coupon	expiration	gender	age	maritalStatus	...	CarryAway
13	Home	Alone	Sunny	55	6PM	Bar	1d	Female	21	Unmarried partner	...	NaN
14	Home	Alone	Sunny	55	6PM	Restaurant(20-50)	1d	Female	21	Unmarried partner	...	NaN
15	Home	Alone	Sunny	80	6PM	Coffee House	2h	Female	21	Unmarried partner	...	NaN
35	Home	Alone	Sunny	55	6PM	Bar	1d	Male	21	Single	...	4-8
36	Home	Alone	Sunny	55	6PM	Restaurant(20-50)	1d	Male	21	Single	...	4-8
...	...	...	...	...	...	...	...	...	...	...	...	...
12675	Home	Alone	Snowy	30	10PM	Coffee House	2h	Male	26	Single	...	1-3
12676	Home	Alone	Sunny	80	6PM	Restaurant(20-50)	1d	Male	26	Single	...	1-3
12677	Home	Partner	Sunny	30	6PM	Restaurant(<20)	1d	Male	26	Single	...	1-3
12678	Home	Partner	Sunny	30	10PM	Restaurant(<20)	2h	Male	26	Single	...	1-3
12679	Home	Partner	Rainy	55	6PM	Carry out & Take away	1d	Male	26	Single	...	1-3

3237 rows × 27 columns

## SQL query

**SELECT \*FROM dataset\_1 ORDER BY coupon**

Result:

dataset\_1\_1 ×

SQL SELECT \*FROM dataset\_1 ORDER BY coupon Enter a SQL expression to filter results (use Ctrl+Space)

	destination	passanger	weather	temperature	time	coupon	expiration	gender	age
1	No Urgent Place	Kid(s)	Sunny	80	10AM	Bar	1d	Female	21
2	Home	Alone	Sunny	55	6PM	Bar	1d	Female	21
3	Work	Alone	Sunny	55	7AM	Bar	1d	Female	21
4	No Urgent Place	Friend(s)	Sunny	80	10AM	Bar	1d	Male	21
5	Home	Alone	Sunny	55	6PM	Bar	1d	Male	21
6	Work	Alone	Sunny	55	7AM	Bar	1d	Male	21
7	No Urgent Place	Friend(s)	Sunny	80	10AM	Bar	1d	Male	46
8	Home	Alone	Sunny	55	6PM	Bar	1d	Male	46
9	Work	Alone	Sunny	55	7AM	Bar	1d	Male	46
10	No Urgent Place	Kid(s)	Sunny	80	10AM	Bar	1d	Male	46
11	Home	Alone	Sunny	55	6PM	Bar	1d	Male	46
12	Work	Alone	Sunny	55	7AM	Bar	1d	Male	46
13	No Urgent Place	Friend(s)	Sunny	80	10AM	Bar	1d	Male	21
14	Home	Alone	Sunny	55	6PM	Bar	1d	Male	21
15	Work	Alone	Sunny	55	7AM	Bar	1d	Male	21
16	No Urgent Place	Friend(s)	Sunny	80	10AM	Bar	1d	Male	26
17	Home	Alone	Sunny	55	6PM	Bar	1d	Male	26
18	Work	Alone	Sunny	55	7AM	Bar	1d	Male	26
19	No Urgent Place	Kid(s)	Sunny	80	10AM	Bar	1d	Female	26
20	Home	Alone	Sunny	55	6PM	Bar	1d	Female	26
21	Work	Alone	Sunny	55	7AM	Bar	1d	Female	26

## Python Code

`df.sort_values('coupon')`

Result:

```
In [3]: # SELECT *FROM dataset_1 ORDER BY coupon  
df.sort_values('coupon')
```

Out[3]:

	destination	passanger	weather	temperature	time	coupon	expiration	gender	age	maritalStatus	...	CarryAway	F
11702	Home	Partner	Sunny	30	10PM	Bar	2h	Female	50plus	Married partner	—	4~8	
9930	No Urgent Place	Alone	Snowy	30	2PM	Bar	1d	Female	21	Single	—	gt8	
10632	Home	Alone	Rainy	55	6PM	Bar	1d	Male	21	Single	—	gt8	
7997	No Urgent Place	Friend(s)	Rainy	55	10PM	Bar	2h	Male	26	Unmarried partner	—	4~8	
11166	Work	Alone	Snowy	30	7AM	Bar	1d	Female	41	Married partner	—	gt8	
...	...	...	...	...	...	...	...	...	...	...	...	...	...
10476	Home	Alone	Sunny	80	6PM	Restaurant(<20)	1d	Female	31	Unmarried partner	—	1~3	
5447	Home	Alone	Sunny	80	10PM	Restaurant(<20)	2h	Female	50plus	Single	—	less1	
10478	Home	Alone	Snowy	30	10PM	Restaurant(<20)	2h	Female	31	Unmarried partner	—	1~3	
5440	No Urgent Place	Alone	Sunny	80	2PM	Restaurant(<20)	2h	Female	50plus	Single	—	less1	
0	No Urgent Place	Alone	Sunny	55	2PM	Restaurant(<20)	1d	Female	21	Unmarried partner	—	NaN	

12684 rows × 27 columns

## SQL query

`SELECT destination as Destination FROM dataset_1`

Result:



dataset_1 1 ×	
SELECT destination as Destination FROM dataset_1	
Grid	Destination
1	No Urgent Place
2	No Urgent Place
3	No Urgent Place
4	No Urgent Place
5	No Urgent Place
6	No Urgent Place
7	No Urgent Place
8	No Urgent Place
9	No Urgent Place
10	No Urgent Place
11	No Urgent Place
12	No Urgent Place
13	No Urgent Place
14	Home
15	Home
16	Home
17	Work
18	Work

### Python Code

```
df.rename(columns={'destination':'Destination'},inplace=True)
```

Result:



In [30]: df

Out[30]:

	Destination
0	No Urgent Place
1	No Urgent Place
2	No Urgent Place
3	No Urgent Place
4	No Urgent Place
...	...
12679	Home
12680	Work
12681	Work
12682	Work
12683	Work

SQL query

SELECT occupation FROM dataset\_1 GROUP BY occupation

Result:

dataset_1 1 x	
SELECT occupation FROM dataset_1 GROUP BY occ	
Grid	ABC 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
25	Unemployed

### Python Code

```
df.groupby('occupation').size().to_frame('Count').reset_index()
```

Result:

```
In [9]: # SELECT occupation FROM dataset_1 GROUP BY occupation
df.groupby('occupation').size().to_frame('Count').reset_index()
```

Out[9]:

	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
17	Personal Care & Service	175
18	Production Occupations	110
19	Protective Service	175
20	Retired	495
21	Sales & Related	1093
22	Student	1584
23	Transportation & Material Moving	218
24	Unemployed	1870

### SQL query

**SELECT weather ,AVG(temperature) as avg\_temp FROM dataset\_1 GROUP BY weather**

Result:

dataset_1 1 ×		
SELECT weather ,AVG(temperature) as avg_temp FR		
Grid	weather	avg_temp
1	Rainy	55
2	Snowy	30
3	Sunny	68.9462707319
Text		

### Python Code

```
df.groupby('weather')['temperature'].mean().to_frame('avg_temp').reset_index()
```

Result:

In [42]:	# SELECT weather ,AVG(temperature) as avg_temp FROM dataset_1 GROUP BY weather df.groupby('weather')['temperature'].mean().to_frame('avg_temp').reset_index()	
Out[42]:	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
```

Result:

dataset_1 1 ×		
SELECT weather ,COUNT( temperature) AS count_te		
Grid	weather	count_temp
1	Rainy	1,210
2	Snowy	1,405
3	Sunny	10,069
Text		

## Python Code

```
df.groupby('weather')['temperature'].size().to_frame('Count_temp').reset_index()
```

Result:

```
In [4]: # SELECT weather ,COUNT( temperature) AS count_temp FROM dataset_1 GROUP BY weather
df.groupby('weather')['temperature'].size().to_frame('Count_temp').reset_index()
```

```
Out[4]:
```

	weather	Count_temp
0	Rainy	1210
1	Snowy	1405
2	Sunny	10069

## SQL query

```
SELECT weather ,COUNT(DISTINCT temperature) AS count_distinct_temp FROM dataset_1 GROUP BY weather
```

Result:

dataset\_1 1

SELECT weather ,COUNT(DISTINCT temperature) AS

Grid

	weather	count_distinct_temp
1	Rainy	1
2	Snowy	1
3	Sunny	3

Text

## Python Code

```
df.groupby('weather')['temperature'].nunique().to_frame('count_distinct_temp').reset_index()
```

Result:

```
In [12]: # SELECT weather ,COUNT(DISTINCT temperature) AS count_distinct_temp FROM dataset_1 GROUP BY weather
df.groupby('weather')['temperature'].nunique().to_frame('count_distinct_temp').reset_index()
```

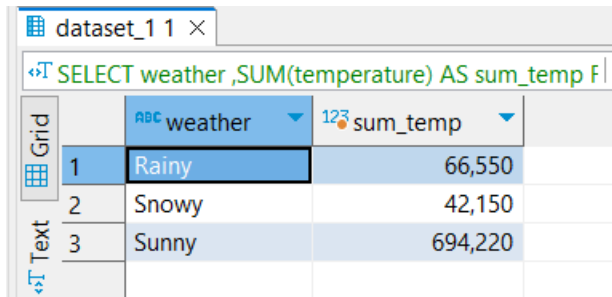
```
Out[12]:
```

	weather	count_distinct_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
```

Result:



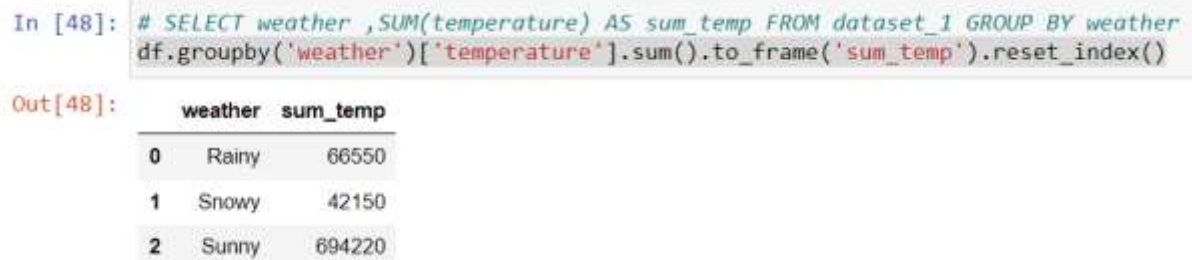
The screenshot shows a database interface with a tab labeled 'dataset\_1 1'. Below the tab, the SQL query 'SELECT weather ,SUM(temperature) AS sum\_temp FROM dataset\_1 GROUP BY weather' is entered. The result is displayed in a table with two columns: 'weather' and 'sum\_temp'. The table has three rows: 'Rainy' with a sum of 66,550, 'Snowy' with a sum of 42,150, and 'Sunny' with a sum of 694,220. The interface also shows a 'Grid' view icon and a 'Text' view icon.

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

## Python Code

```
df.groupby('weather')['temperature'].sum().to_frame('sum_temp').reset_index()
```

Result:



The screenshot shows a Jupyter Notebook cell with the following code and output:

```
In [48]: # SELECT weather ,SUM(temperature) AS sum_temp FROM dataset_1 GROUP BY weather
df.groupby('weather')['temperature'].sum().to_frame('sum_temp').reset_index()
```

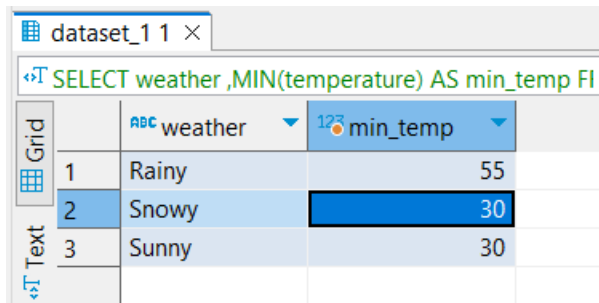
Out[48]:

	weather	sum_temp
0	Rainy	66550
1	Snowy	42150
2	Sunny	694220

### SQL query

```
SELECT weather ,MIN(temperature) AS min_temp FROM dataset_1 GROUP BY weather
```

Result:

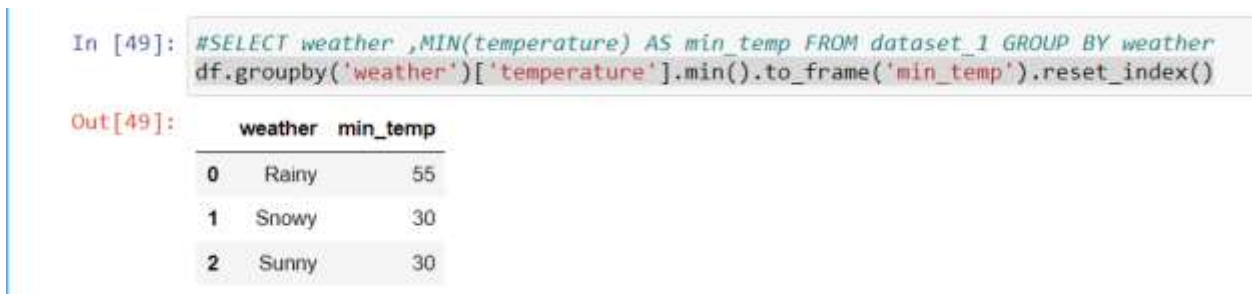


	weather	min_temp
1	Rainy	55
2	Snowy	30
3	Sunny	30

### Python Code

```
df.groupby('weather')['temperature'].min().to_frame('min_temp').reset_index()
```

Result:



```
In [49]: #SELECT weather ,MIN(temperature) AS min_temp FROM dataset_1 GROUP BY weather
df.groupby('weather')['temperature'].min().to_frame('min_temp').reset_index()
```

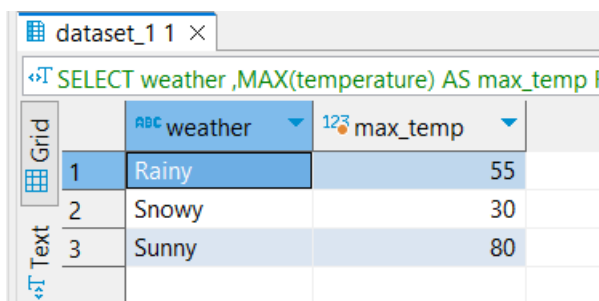
```
Out[49]:
```

	weather	min_temp
0	Rainy	55
1	Snowy	30
2	Sunny	30

### SQL query

```
SELECT weather ,MAX(temperature) AS max_temp FROM dataset_1 GROUP BY weather
```

Result:



	weather	max_temp
1	Rainy	55
2	Snowy	30
3	Sunny	80



### Python Code

```
df.groupby('weather')['temperature'].max().to_frame('max_temp').reset_index()
```

Result:

```
In [50]: #SELECT weather ,MAX(temperature) AS max_temp FROM dataset_1 GROUP BY weather
df.groupby('weather')['temperature'].max().to_frame('max_temp').reset_index()
```

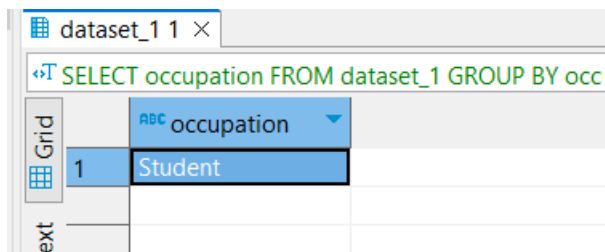
```
Out[50]:
```

	weather	max_temp
0	Rainy	55
1	Snowy	30
2	Sunny	80

### SQL query

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

Result:



The screenshot shows a SQL query interface with a tab labeled 'dataset\_1 1'. The query entered is 'SELECT occupation FROM dataset\_1 GROUP BY occ'. The results are displayed in a table with two columns: 'Grid' and 'ABC occupation'. The 'Grid' column has a value of '1', and the 'ABC occupation' column has a value of 'Student'.

Grid	ABC occupation
1	Student

### Python Code

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

Result:

```
In [14]: #SELECT occupation FROM dataset_1 GROUP BY occupation HAVING occupation='Student'
df.groupby('occupation').filter(lambda x: x['occupation'].iloc[0] == 'Student').groupby('occupation').size()
```

```
Out[14]: occupation
Student    1584
dtype: int64
```

## SQL query

**SELECT DISTINCT destination FROM(SELECT \* FROM dataset\_1 UNION SELECT \* FROM table\_to\_union)**

Result:

table_to_union 1 ×	
SELECT DISTINCT destination FROM(SELECT * FROM	
Grid	ABC destination ▾
1	Home
2	No Urgent Place
3	UNION
4	Work

## Python Code

**pd.concat([df, df1])['destination'].drop\_duplicates()**

Result:

```
In [59]: #SELECT DISTINCT destination FROM(SELECT * FROM dataset_1 UNION SELECT * FROM table_to_union)
pd.concat([df, df1])['destination'].drop_duplicates()

Out[59]: 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**

Result:

dataset_1(+) 1 ×				
SELECT a.destination,a.time,b.part_of_day FROM dataset_1 a INNER JOIN table_to_join b ON a.time=b.time				
Grid	1	2	3	4
	1	No Urgent Place	2PM	Afternoon
	2	No Urgent Place	10AM	Morning
	3	No Urgent Place	10AM	Morning
	4	No Urgent Place	2PM	Afternoon
	5	No Urgent Place	2PM	Afternoon
	6	No Urgent Place	6PM	Evening
	7	No Urgent Place	2PM	Afternoon
	8	No Urgent Place	10AM	Morning
	9	No Urgent Place	10AM	Morning
	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

## Python Code

**pd.merge(df, df2[['time', 'part\_of\_day']], on='time', how='inner')[['destination', 'time', 'part\_of\_day']]**

Result:

```
In [62]: # SELECT a.destination,a.time,b.part_of_day FROM dataset_1 a INNER JOIN table to join b ON a.time=b.time
pd.merge(df, df2[['time', 'part_of_day']], on='time', how='inner')[['destination', 'time', 'part_of_day']]
```

```
Out[62]:
```

	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 x 3 columns

## SQL query

**SELECT destination ,passanger FROM(SELECT\*FROM dataset\_1 WHERE passenger = 'Alone')**

Result:

dataset_1 1 x	
SELECT destination ,passanger FROM(SELECT*FROM	
Grid	Text
Record	
1	No Urgent Place Alone
2	Home Alone
3	Home Alone
4	Home Alone
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

### Python Code

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

Result:

```
In [64]: #SELECT destination ,passenger FROM(SELECT*FROM dataset_1 WHERE passenger = 'Alone')
df[df['passanger'] == 'Alone'][['destination', 'passanger']]
```

```
Out[64]:
```

	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

7305 rows × 2 columns

### SQL query

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

Result:

dataset\_1 1 x

SELECT \* FROM dataset\_1 WHERE weather LIKE 'Sun'

	destination	passenger	weather	temperature	time	coupon	expiration	gender	age
1	No Urgent Place	Alone	Sunny	55	2PM	Restaurant(<20)	1d	Female	21
2	No Urgent Place	Friend(s)	Sunny	80	10AM	Coffee House	2h	Female	21
3	No Urgent Place	Friend(s)	Sunny	80	10AM	Carry out & Take away	2h	Female	21
4	No Urgent Place	Friend(s)	Sunny	80	2PM	Coffee House	2h	Female	21
5	No Urgent Place	Friend(s)	Sunny	80	2PM	Coffee House	1d	Female	21
6	No Urgent Place	Friend(s)	Sunny	80	6PM	Restaurant(<20)	2h	Female	21
7	No Urgent Place	Friend(s)	Sunny	55	2PM	Carry out & Take away	1d	Female	21
8	No Urgent Place	Kid(s)	Sunny	80	10AM	Restaurant(<20)	2h	Female	21
9	No Urgent Place	Kid(s)	Sunny	80	10AM	Carry out & Take away	2h	Female	21
10	No Urgent Place	Kid(s)	Sunny	80	10AM	Bar	1d	Female	21
11	No Urgent Place	Kid(s)	Sunny	80	2PM	Restaurant(<20)	1d	Female	21
12	No Urgent Place	Kid(s)	Sunny	55	2PM	Restaurant(<20)	1d	Female	21
13	No Urgent Place	Kid(s)	Sunny	55	6PM	Coffee House	2h	Female	21
14	Home	Alone	Sunny	55	6PM	Bar	1d	Female	21
15	Home	Alone	Sunny	55	6PM	Restaurant(20-50)	1d	Female	21
16	Home	Alone	Sunny	80	6PM	Coffee House	2h	Female	21
17	Work	Alone	Sunny	55	7AM	Coffee House	2h	Female	21
18	Work	Alone	Sunny	55	7AM	Bar	1d	Female	21
19	Work	Alone	Sunny	80	7AM	Restaurant(20-50)	1d	Female	21
20	Work	Alone	Sunny	80	7AM	Carry out & Take away	2h	Female	21
21	Work	Alone	Sunny	55	7AM	Restaurant(<20)	1d	Female	21
22	Work	Alone	Sunny	55	7AM	Coffee House	2h	Female	21
23	No Urgent Place	Alone	Sunny	55	2PM	Restaurant(<20)	1d	Male	21
24	No Urgent Place	Friend(s)	Sunny	80	10AM	Coffee House	2h	Male	21
25	No Urgent Place	Friend(s)	Sunny	80	10AM	Bar	1d	Male	21
26	No Urgent Place	Friend(s)	Sunny	80	10AM	Carry out & Take away	2h	Male	21
27	No Urgent Place	Friend(s)	Sunny	80	2PM	Coffee House	1d	Male	21
28	No Urgent Place	Friend(s)	Sunny	80	2PM	Coffee House	2h	Male	21
29	No Urgent Place	Friend(s)	Sunny	80	2PM	Coffee House	1d	Male	21

## Python Code

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

Result:

```
In [65]: #SELECT * FROM dataset_1 WHERE weather LIKE 'Sun%'
df[df['weather'].str.startswith('Sun')]
```

```
Out[65]:
```

	destination	passanger	weather	temperature	time	coupon	expiration	gender	age	n
0	No Urgent Place	Alone	Sunny	55	2PM	Restaurant(<20)	1d	Female	21	
1	No Urgent Place	Friend(s)	Sunny	80	10AM	Coffee House	2h	Female	21	
2	No Urgent Place	Friend(s)	Sunny	80	10AM	Carry out & Take away	2h	Female	21	
3	No Urgent Place	Friend(s)	Sunny	80	2PM	Coffee House	2h	Female	21	
4	No Urgent Place	Friend(s)	Sunny	80	2PM	Coffee House	1d	Female	21	
...	...	...	...	...	...	...	...	...	...	...
12673	Home	Alone	Sunny	30	6PM	Carry out & Take away	1d	Male	26	
12676	Home	Alone	Sunny	80	6PM	Restaurant(20-50)	1d	Male	26	
12677	Home	Partner	Sunny	30	6PM	Restaurant(<20)	1d	Male	26	
12678	Home	Partner	Sunny	30	10PM	Restaurant(<20)	2h	Male	26	
12683	Work	Alone	Sunny	80	7AM	Restaurant(20-50)	2h	Male	26	

10069 rows × 27 columns

## SQL query

**SELECT DISTINCT temperature FROM dataset\_1 WHERE temperature BETWEEN 29 AND 75**

Result:

dataset_1 1	×
SELECT DISTINCT temperature FROM dataset_1 WH	
Grid	123 temperature
1	55
2	30

## Python Code

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



Result:

```
In [70]: #SELECT DISTINCT temperature FROM dataset_1 WHERE temperature BETWEEN 29 AND 75
df[(df['temperature'] >= 29) & (df['temperature'] <= 75)]['temperature'].unique()

Out[70]: array([55, 30], dtype=int64)
```

### SQL query

**SELECT occupation FROM dataset\_1 WHERE occupation IN('Sales & Related','Management')**

Result:

dataset_1 1 x	
SELECT occupation FROM dataset_1 WHERE occup:	
Grid	abc occupation
1	Sales & Related
2	Sales & Related
3	Sales & Related
4	Sales & Related
5	Sales & Related
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

## Python Code

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

Result:

```
In [68]: #SELECT occupation FROM dataset 1 WHERE occupation IN('sales and related','Management')
df[df['occupation'].isin(['Sales & Related', 'Management'])][['occupation']]
```

```
Out[68]:
```

	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