

Retrieve all data from the dataset

SELECT \* FROM dataset\_1 ;

dataset\_1 1 X

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

Grid

Text

Record

|    | ABC destination | ABC passanger | ABC weather | 123 temperature | ABC time | ABC coupon         | ABC expiration | ABC gender |
|----|-----------------|---------------|-------------|-----------------|----------|--------------------|----------------|------------|
| 1  | No Urgent Place | Alone         | Sunny       | 55              | 2PM      | Restaurant(<20)    | 1d             | Female     |
| 2  | No Urgent Place | Friend(s)     | Sunny       | 80              | 10AM     | Coffee House       | 2h             | Female     |
| 3  | No Urgent Place | Friend(s)     | Sunny       | 80              | 10AM     | Carry out & Take a | 2h             | Female     |
| 4  | No Urgent Place | Friend(s)     | Sunny       | 80              | 2PM      | Coffee House       | 2h             | Female     |
| 5  | No Urgent Place | Friend(s)     | Sunny       | 80              | 2PM      | Coffee House       | 1d             | Female     |
| 6  | No Urgent Place | Friend(s)     | Sunny       | 80              | 6PM      | Restaurant(<20)    | 2h             | Female     |
| 7  | No Urgent Place | Friend(s)     | Sunny       | 55              | 2PM      | Carry out & Take a | 1d             | Female     |
| 8  | No Urgent Place | Kid(s)        | Sunny       | 80              | 10AM     | Restaurant(<20)    | 2h             | Female     |
| 9  | No Urgent Place | Kid(s)        | Sunny       | 80              | 10AM     | Carry out & Take a | 2h             | Female     |
| 10 | No Urgent Place | Kid(s)        | Sunny       | 80              | 10AM     | Bar                | 1d             | Female     |
| 11 | No Urgent Place | Kid(s)        | Sunny       | 80              | 2PM      | Restaurant(<20)    | 1d             | Female     |
| 12 | No Urgent Place | Kid(s)        | Sunny       | 55              | 2PM      | Restaurant(<20)    | 1d             | Female     |
| 13 | No Urgent Place | Kid(s)        | Sunny       | 55              | 6PM      | Coffee House       | 2h             | Female     |
| 14 | Home            | Alone         | Sunny       | 55              | 6PM      | Bar                | 1d             | Female     |
| 15 | Home            | Alone         | Sunny       | 55              | 6PM      | Restaurant(20-50)  | 1d             | Female     |
| 16 | Home            | Alone         | Sunny       | 80              | 6PM      | Coffee House       | 2h             | Female     |
| 17 | Work            | Alone         | Sunny       | 55              | 7AM      | Coffee House       | 2h             | Female     |
| 18 | Work            | Alone         | Sunny       | 55              | 7AM      | Bar                | 1d             | Female     |
| 19 | Work            | Alone         | Sunny       | 80              | 7AM      | Restaurant(20-50)  | 1d             | Female     |
| 20 | Work            | Alone         | Sunny       | 80              | 7AM      | Carry out & Take a | 2h             | Female     |
| 21 | Work            | Alone         | Sunny       | 55              | 7AM      | Restaurant(<20)    | 1d             | Female     |
| 22 | Work            | Alone         | Sunny       | 55              | 7AM      | Coffee House       | 2h             | Female     |

Refresh Save Cancel Export data 200 200+

200 row(s) fetched - 0.012s (0.011s fetch), on 2024-04-12 at 17:10:01

Retrieve all data from the dataset

# SELECT \* FROM dataset\_1;

dataset\_1

|     | destination     | passanger | weather | temperature | time | coupon                | expiration | gender | age | maritalStatus     | ... | CarryAway | RestaurantLessThan20 | Restaurant2 |
|-----|-----------------|-----------|---------|-------------|------|-----------------------|------------|--------|-----|-------------------|-----|-----------|----------------------|-------------|
| 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                  |             |
| ... | ...             | ...       | ...     | ...         | ...  | ...                   | ...        | ...    | ... | ...               | ... | ...       | ...                  |             |

ENG 5:10 PM

Fetch the data of first 10 rows from the dataset

SQL query editor showing the query: `SELECT * FROM dataset_1 LIMIT 10;`

Results table (10 rows):

|    | destination     | passanger | weather | temperature | time | coupon             | expiration | gender |
|----|-----------------|-----------|---------|-------------|------|--------------------|------------|--------|
| 1  | No Urgent Place | Alone     | Sunny   | 55          | 2PM  | Restaurant(<20)    | 1d         | Female |
| 2  | No Urgent Place | Friend(s) | Sunny   | 80          | 10AM | Coffee House       | 2h         | Female |
| 3  | No Urgent Place | Friend(s) | Sunny   | 80          | 10AM | Carry out & Take a | 2h         | Female |
| 4  | No Urgent Place | Friend(s) | Sunny   | 80          | 2PM  | Coffee House       | 2h         | Female |
| 5  | No Urgent Place | Friend(s) | Sunny   | 80          | 2PM  | Coffee House       | 1d         | Female |
| 6  | No Urgent Place | Friend(s) | Sunny   | 80          | 6PM  | Restaurant(<20)    | 2h         | Female |
| 7  | No Urgent Place | Friend(s) | Sunny   | 55          | 2PM  | Carry out & Take a | 1d         | Female |
| 8  | No Urgent Place | Kid(s)    | Sunny   | 80          | 10AM | Restaurant(<20)    | 2h         | Female |
| 9  | No Urgent Place | Kid(s)    | Sunny   | 80          | 10AM | Carry out & Take a | 2h         | Female |
| 10 | No Urgent Place | Kid(s)    | Sunny   | 80          | 10AM | Bar                | 1d         | Female |

Fetch the data of first 10 rows from the dataset

Python code snippet:

```
# SELECT * FROM dataset_1 LIMIT 10;
dataset_1.head(10)
```

Results table (10 rows):

|   | destination     | passanger | weather | temperature | time | coupon                | expiration | gender | age | maritalStatus     | ... | CarryAway | RestaurantLessThan20 | Restaurant20To5 |
|---|-----------------|-----------|---------|-------------|------|-----------------------|------------|--------|-----|-------------------|-----|-----------|----------------------|-----------------|
| 0 | No Urgent Place | Alone     | Sunny   | 55          | 2PM  | Restaurant(<20)       | 1d         | Female | 21  | Unmarried partner | ... | NaN       | 4~8                  | 1~              |
| 1 | No Urgent Place | Friend(s) | Sunny   | 80          | 10AM | Coffee House          | 2h         | Female | 21  | Unmarried partner | ... | NaN       | 4~8                  | 1~              |
| 2 | No Urgent Place | Friend(s) | Sunny   | 80          | 10AM | Carry out & Take away | 2h         | Female | 21  | Unmarried partner | ... | NaN       | 4~8                  | 1~              |
| 3 | No Urgent Place | Friend(s) | Sunny   | 80          | 2PM  | Coffee House          | 2h         | Female | 21  | Unmarried partner | ... | NaN       | 4~8                  | 1~              |
| 4 | No Urgent Place | Friend(s) | Sunny   | 80          | 2PM  | Coffee House          | 1d         | Female | 21  | Unmarried partner | ... | NaN       | 4~8                  | 1~              |
| 5 | No Urgent Place | Friend(s) | Sunny   | 80          | 6PM  | Restaurant(<20)       | 2h         | Female | 21  | Unmarried partner | ... | NaN       | 4~8                  | 1~              |

Select passenger and weather column from the dataset

SELECT passenger , weather FROM dataset\_1 ;

dataset\_11 X

SELECT passenger , weather FROM data: Enter a SQL expression to filter results (use Ctrl+Space)

|    | ABC passenger | ABC weather |
|----|---------------|-------------|
| 1  | Alone         | Sunny       |
| 2  | Friend(s)     | Sunny       |
| 3  | Friend(s)     | Sunny       |
| 4  | Friend(s)     | Sunny       |
| 5  | Friend(s)     | Sunny       |
| 6  | Friend(s)     | Sunny       |
| 7  | Friend(s)     | Sunny       |
| 8  | Kid(s)        | Sunny       |
| 9  | Kid(s)        | Sunny       |
| 10 | Kid(s)        | Sunny       |
| 11 | Kid(s)        | Sunny       |
| 12 | Kid(s)        | Sunny       |
| 13 | Kid(s)        | Sunny       |
| 14 | Alone         | Sunny       |
| 15 | Alone         | Sunny       |
| 16 | Alone         | Sunny       |
| 17 | Alone         | Sunny       |
| 18 | Alone         | Sunny       |
| 19 | Alone         | Sunny       |
| 20 | Alone         | Sunny       |
| 21 | Alone         | Sunny       |

Refresh

Save

Cancel

Export data

200

200 row(s) fetched - 0.002s (0.001s fetch), on 2024-04-12 at 17:15:02

Select passenger and weather column from the dataset

```
# SELECT passenger , weather FROM dataset_1;

dataset_1[['passanger','weather']]
```

|       | passanger | weather |
|-------|-----------|---------|
| 0     | Alone     | Sunny   |
| 1     | Friend(s) | Sunny   |
| 2     | Friend(s) | Sunny   |
| 3     | Friend(s) | Sunny   |
| 4     | Friend(s) | Sunny   |
| ...   | ...       | ...     |
| 12679 | Partner   | Rainy   |
| 12680 | Alone     | Rainy   |
| 12681 | Alone     | Snowy   |
| 12682 | Alone     | Snowy   |

Select unique value for passenger from the dataset

The screenshot shows a SQLite Test.db interface. At the top, a tab labeled "Script.sql" contains the SQL query: `SELECT DISTINCT passenger FROM dataset_1 ;`. Below the query editor, a tab labeled "dataset\_1 1" displays the results of the query. The results are shown in a table with a grid icon on the left. The table has one column, "passanger", and four rows of data: "Alone", "Friend(s)", "Kid(s)", and "Partner". The "passanger" column header is highlighted in blue, and the "Alone" row is also highlighted in blue. The table is titled "ABC passanger".

| passanger |
|-----------|
| Alone     |
| Friend(s) |
| Kid(s)    |
| Partner   |

Select unique value for passanger from the dataset

```
# SELECT DISTINCT passenger FROM dataset_1;

dataset_1['passanger'].unique()

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

Retrieve all the data from the dataset whose destination is work

SELECT \* FROM dataset\_1 WHERE destination == 'Work' ;

dataset\_11 X

SELECT \* FROM dataset\_1 WHERE destir Enter a SQL expression to filter results (use Ctrl+Space)

|    | ABC destination | ABC passanger | ABC weather | 123 temperature | ABC time | ABC coupon            | ABC expiration | ABC ge |
|----|-----------------|---------------|-------------|-----------------|----------|-----------------------|----------------|--------|
| 1  | Work            | Alone         | Sunny       | 55              | 7AM      | Coffee House          | 2h             | Femal  |
| 2  | Work            | Alone         | Sunny       | 55              | 7AM      | Bar                   | 1d             | Femal  |
| 3  | Work            | Alone         | Sunny       | 80              | 7AM      | Restaurant(20-50)     | 1d             | Femal  |
| 4  | Work            | Alone         | Sunny       | 80              | 7AM      | Carry out & Take away | 2h             | Femal  |
| 5  | Work            | Alone         | Sunny       | 55              | 7AM      | Restaurant(<20)       | 1d             | Femal  |
| 6  | Work            | Alone         | Sunny       | 55              | 7AM      | Coffee House          | 2h             | Femal  |
| 7  | Work            | Alone         | Sunny       | 55              | 7AM      | Coffee House          | 2h             | Male   |
| 8  | Work            | Alone         | Sunny       | 55              | 7AM      | Bar                   | 1d             | Male   |
| 9  | Work            | Alone         | Sunny       | 80              | 7AM      | Restaurant(20-50)     | 1d             | Male   |
| 10 | Work            | Alone         | Sunny       | 80              | 7AM      | Carry out & Take away | 2h             | Male   |
| 11 | Work            | Alone         | Sunny       | 55              | 7AM      | Restaurant(<20)       | 1d             | Male   |
| 12 | Work            | Alone         | Sunny       | 55              | 7AM      | Coffee House          | 2h             | Male   |
| 13 | Work            | Alone         | Sunny       | 55              | 7AM      | Coffee House          | 2h             | Male   |
| 14 | Work            | Alone         | Sunny       | 55              | 7AM      | Bar                   | 1d             | Male   |
| 15 | Work            | Alone         | Sunny       | 80              | 7AM      | Restaurant(20-50)     | 1d             | Male   |
| 16 | Work            | Alone         | Sunny       | 80              | 7AM      | Carry out & Take away | 2h             | Male   |
| 17 | Work            | Alone         | Sunny       | 55              | 7AM      | Restaurant(<20)       | 1d             | Male   |
| 18 | Work            | Alone         | Sunny       | 55              | 7AM      | Coffee House          | 2h             | Male   |
| 19 | Work            | Alone         | Sunny       | 55              | 7AM      | Coffee House          | 2h             | Male   |
| 20 | Work            | Alone         | Sunny       | 55              | 7AM      | Bar                   | 1d             | Male   |

Retrieve all the data from the dataset whose destination is work

```
# SELECT * FROM dataset_1 WHERE destination == 'Work';  
dataset_1[dataset_1['destination']=='Work']
```

|       | destination | passanger | weather | temperature | time | coupon                | expiration | gender | age | maritalStatus     | ... | CarryAway | RestaurantLessThan20 | Restaurant20 |
|-------|-------------|-----------|---------|-------------|------|-----------------------|------------|--------|-----|-------------------|-----|-----------|----------------------|--------------|
| 16    | Work        | Alone     | Sunny   | 55          | 7AM  | Coffee House          | 2h         | Female | 21  | Unmarried partner | ... | NaN       | 4~8                  |              |
| 17    | Work        | Alone     | Sunny   | 55          | 7AM  | Bar                   | 1d         | Female | 21  | Unmarried partner | ... | NaN       | 4~8                  |              |
| 18    | Work        | Alone     | Sunny   | 80          | 7AM  | Restaurant(20-50)     | 1d         | Female | 21  | Unmarried partner | ... | NaN       | 4~8                  |              |
| 19    | Work        | Alone     | Sunny   | 80          | 7AM  | Carry out & Take away | 2h         | Female | 21  | Unmarried partner | ... | NaN       | 4~8                  |              |
| 20    | Work        | Alone     | Sunny   | 55          | 7AM  | Restaurant(<20)       | 1d         | Female | 21  | Unmarried partner | ... | NaN       | 4~8                  |              |
| ...   | ...         | ...       | ...     | ...         | ...  | ...                   | ...        | ...    | ... | ...               | ... | ...       | ...                  | ...          |
| 18255 | Work        | Alone     | Sunny   | 80          | 7AM  | Restaurant(20-50)     | 1d         | Male   | 21  | Married           | ... | NaN       | 4~8                  |              |

## Sorting entire data based on coupon

\*<SQLite Test.db> Script.sql X

```
SELECT * FROM dataset_1 ORDER BY COUPON ;
```

dataset\_11 X

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

|    | ABC destination | ABC passanger | ABC weather | 123 temperature | ABC time | ABC coupon | ABC expiration | ABC ge |
|----|-----------------|---------------|-------------|-----------------|----------|------------|----------------|--------|
| 1  | No Urgent Place | Kid(s)        | Sunny       | 80              | 10AM     | Bar        | 1d             | Femal  |
| 2  | Home            | Alone         | Sunny       | 55              | 6PM      | Bar        | 1d             | Femal  |
| 3  | Work            | Alone         | Sunny       | 55              | 7AM      | Bar        | 1d             | Femal  |
| 4  | No Urgent Place | Friend(s)     | Sunny       | 80              | 10AM     | Bar        | 1d             | Male   |
| 5  | Home            | Alone         | Sunny       | 55              | 6PM      | Bar        | 1d             | Male   |
| 6  | Work            | Alone         | Sunny       | 55              | 7AM      | Bar        | 1d             | Male   |
| 7  | No Urgent Place | Friend(s)     | Sunny       | 80              | 10AM     | Bar        | 1d             | Male   |
| 8  | Home            | Alone         | Sunny       | 55              | 6PM      | Bar        | 1d             | Male   |
| 9  | Work            | Alone         | Sunny       | 55              | 7AM      | Bar        | 1d             | Male   |
| 10 | No Urgent Place | Kid(s)        | Sunny       | 80              | 10AM     | Bar        | 1d             | Male   |
| 11 | Home            | Alone         | Sunny       | 55              | 6PM      | Bar        | 1d             | Male   |
| 12 | Work            | Alone         | Sunny       | 55              | 7AM      | Bar        | 1d             | Male   |
| 13 | No Urgent Place | Friend(s)     | Sunny       | 80              | 10AM     | Bar        | 1d             | Male   |
| 14 | Home            | Alone         | Sunny       | 55              | 6PM      | Bar        | 1d             | Male   |
| 15 | Work            | Alone         | Sunny       | 55              | 7AM      | Bar        | 1d             | Male   |
| 16 | No Urgent Place | Friend(s)     | Sunny       | 80              | 10AM     | Bar        | 1d             | Male   |
| 17 | Home            | Alone         | Sunny       | 55              | 6PM      | Bar        | 1d             | Male   |
| 18 | Work            | Alone         | Sunny       | 55              | 7AM      | Bar        | 1d             | Male   |
| 19 | No Urgent Place | Kid(s)        | Sunny       | 80              | 10AM     | Bar        | 1d             | Femal  |
| 20 | Home            | Alone         | Sunny       | 55              | 6PM      | Bar        | 1d             | Femal  |

Refresh Save Cancel Export data 200 200+

## Sorting entire data based on coupon

```
# SELECT * FROM dataset_1 ORDER BY COUPON;
```

```
dataset_1.sort_values('coupon')
```

|       | destination     | passanger | weather | temperature | time | coupon          | expiration | gender | age    | maritalStatus     | ... | CarryAway | RestaurantLessThan20 | Restaurar |
|-------|-----------------|-----------|---------|-------------|------|-----------------|------------|--------|--------|-------------------|-----|-----------|----------------------|-----------|
| 11702 | Home            | Partner   | Sunny   | 30          | 10PM | Bar             | 2h         | Female | 50plus | Married partner   | ... | 4~8       | 1~3                  |           |
| 9930  | No Urgent Place | Alone     | Snowy   | 30          | 2PM  | Bar             | 1d         | Female | 21     | Single            | ... | gt8       | gt8                  |           |
| 10632 | Home            | Alone     | Rainy   | 55          | 6PM  | Bar             | 1d         | Male   | 21     | Single            | ... | gt8       | less1                |           |
| 7997  | No Urgent Place | Friend(s) | Rainy   | 55          | 10PM | Bar             | 2h         | Male   | 26     | Unmarried partner | ... | 4~8       | never                |           |
| 11166 | Work            | Alone     | Snowy   | 30          | 7AM  | Bar             | 1d         | Female | 41     | Married partner   | ... | gt8       | 1~3                  |           |
| ...   | ...             | ...       | ...     | ...         | ...  | ...             | ...        | ...    | ...    | ...               | ... | ...       | ...                  |           |
| 10476 | Home            | Alone     | Sunny   | 80          | 6PM  | Restaurant(<20) | 1d         | Female | 31     | Unmarried partner | ... | 1~3       | 1~3                  |           |

## Rename the specific column of the dataset

SQLite Test.db> Script.sql

```
SELECT destination as 'Destination' , passanger as 'Passanger' from dataset_1 ;
```

dataset\_1 1

SELECT destination as 'Destination' , pas Enter a SQL expression to filter results (use Ctrl+Space)

|    | ABC Destination | ABC Passanger |
|----|-----------------|---------------|
| 1  | No Urgent Place | Alone         |
| 2  | No Urgent Place | Friend(s)     |
| 3  | No Urgent Place | Friend(s)     |
| 4  | No Urgent Place | Friend(s)     |
| 5  | No Urgent Place | Friend(s)     |
| 6  | No Urgent Place | Friend(s)     |
| 7  | No Urgent Place | Friend(s)     |
| 8  | No Urgent Place | Kid(s)        |
| 9  | No Urgent Place | Kid(s)        |
| 10 | No Urgent Place | Kid(s)        |
| 11 | No Urgent Place | Kid(s)        |
| 12 | No Urgent Place | Kid(s)        |
| 13 | No Urgent Place | Kid(s)        |
| 14 | Home            | Alone         |
| 15 | Home            | Alone         |
| 16 | Home            | Alone         |
| 17 | Work            | Alone         |
| 18 | Work            | Alone         |
| 19 | Work            | Alone         |
| 20 | Work            | Alone         |
| 21 | Work            | Alone         |

Refresh Save Cancel Export data 200 200+

### Rename the specific column of the dataset

```
''' df.columns=df.columns.str.replace('destination', 'Destination') # 1st method
df.rename(columns = {'passanger':'Passanger'}, inplace = True) # 2nd method
df
'''

" df.columns=df.columns.str.replace('destination', 'Destination') # 1st method\n df.rename(columns = {'passanger':'Passanger'}, inplace = True) # 2nd method\n df\n"
```

```
# SELECT destination as 'Destination' , passanger as 'Passanger' from dataset_1;

dataset_1.rename(columns={'destination': 'Destination', 'passanger': 'Passanger'}, inplace = True)
dataset_1
```

|   | Destination     | passanger | weather | temperature | time | coupon                | expiration | gender | age | maritalStatus     | ... | CarryAway | RestaurantLessThan20 | Restaurant2 |
|---|-----------------|-----------|---------|-------------|------|-----------------------|------------|--------|-----|-------------------|-----|-----------|----------------------|-------------|
| 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                  |             |

## Grouping the occupation

The screenshot shows a SQLite Test.db interface. The top panel displays a SQL query: `SELECT occupation From dataset_1 GROUP BY occupation ;`. Below the query, the results are shown in a table with 21 rows, each representing an occupation. The table has two columns: an index (1-21) and the occupation name. The occupations listed are: Architecture & Engineering, Arts Design Entertainment Sports & Media, Building & Grounds Cleaning & Maintenance, Business & Financial, Community & Social Services, Computer & Mathematical, Construction & Extraction, Education&Training&Library, Farming Fishing & Forestry, Food Preparation & Serving Related, Healthcare Practitioners & Technical, Healthcare Support, Installation Maintenance & Repair, Legal, Life Physical Social Science, Management, Office & Administrative Support, Personal Care & Service, Production Occupations, Protective Service, and Retired.

| Index | 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                                   |

## Grouping the occupation

```
# SELECT occupation From dataset_1 GROUP BY occupation ;
```

```
dataset_1.groupby('occupation').size().reset_index(name='count')
```

|   | 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   |



## Find the average of temperature by grouping the weather

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

dataset\_11 X

SELECT weather , AVG(temperature) as a | Enter a SQL expression to filter results (use Ctrl+Space)

|   | ABC weather | 123 average_temp |
|---|-------------|------------------|
| 1 | Rainy       | 55               |
| 2 | Snowy       | 30               |
| 3 | Sunny       | 68.9462707319    |
|   |             |                  |
|   |             |                  |
|   |             |                  |
|   |             |                  |
|   |             |                  |
|   |             |                  |

## Find the average of temperature by grouping the weather

```
] : # SELECT weather , AVG(temperature) as average_temp FROM dataset_1 GROUP BY weather;
```

```
dataset_1.groupby('weather')['temperature'].mean().reset_index(name='average_temp')
```

```
] :
```

|   | weather | average_temp |
|---|---------|--------------|
| 0 | Rainy   | 55.000000    |
| 1 | Snowy   | 30.000000    |
| 2 | Sunny   | 68.946271    |

## Find the count of temperature by grouping the weather

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

dataset\_11 X

SELECT weather , Count(temperature) as | Enter a SQL expression to filter results (use Ctrl+Space)

|   | ABC weather | 123 count_temp |
|---|-------------|----------------|
| 1 | Rainy       | 1,210          |
| 2 | Snowy       | 1,405          |
| 3 | Sunny       | 10,069         |
|   |             |                |
|   |             |                |
|   |             |                |
|   |             |                |
|   |             |                |

## Find the count of temperature by grouping the weather

```
# SELECT weather , Count(temperature) as count_temp FROM dataset_1 GROUP BY weather;
```

```
dataset_1.groupby('weather')['temperature'].count().reset_index(name='count_temp')
```

```
weather count_temp
0    Rainy      1210
1    Snowy      1405
2    Sunny     10069
```

```
SELECT weather , Count(DISTINCT temperature) as count_distinct_temp FROM dataset_1 GROUP BY weather;
```

dataset\_11 X

```
SELECT weather , Count(DISTINCT temp) Enter a SQL expression to filter results (use Ctrl+Space)
```

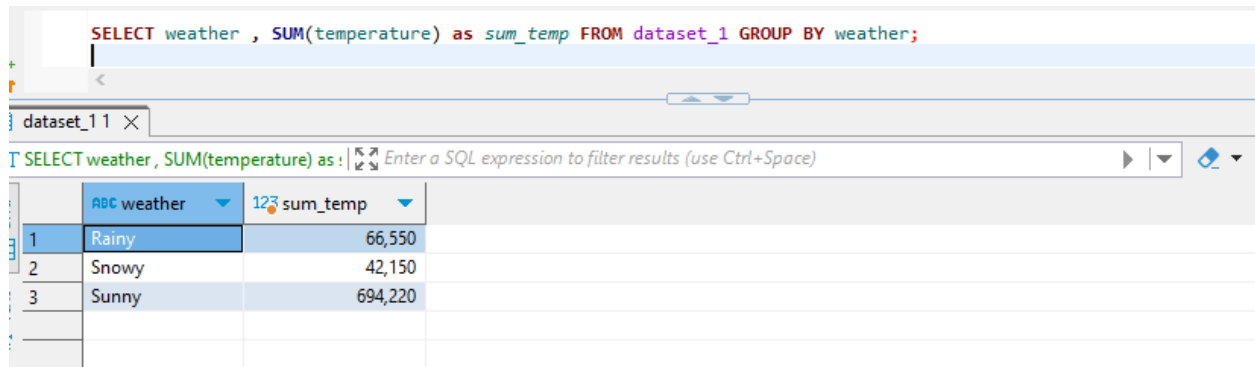
|   | weather | count_distinct_temp |
|---|---------|---------------------|
| 1 | Rainy   | 1                   |
| 2 | Snowy   | 1                   |
| 3 | Sunny   | 3                   |

```
# SELECT weather , Count(DISTINCT temperature) as count_distinct_temp FROM dataset_1 GROUP BY weather;
```

```
dataset_1.groupby('weather')['temperature'].nunique().reset_index(name='count_distinct_temp')
```

```
weather count_distinct_temp
0    Rainy                1
1    Snowy                1
2    Sunny                3
```

Find the sum of temperature by grouping the weather



The screenshot shows a SQL query editor with the following query: `SELECT weather , SUM(temperature) as sum_temp FROM dataset_1 GROUP BY weather;` The query is executed, and the results are displayed in a table with two columns: 'weather' and 'sum\_temp'. The results are as follows:

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

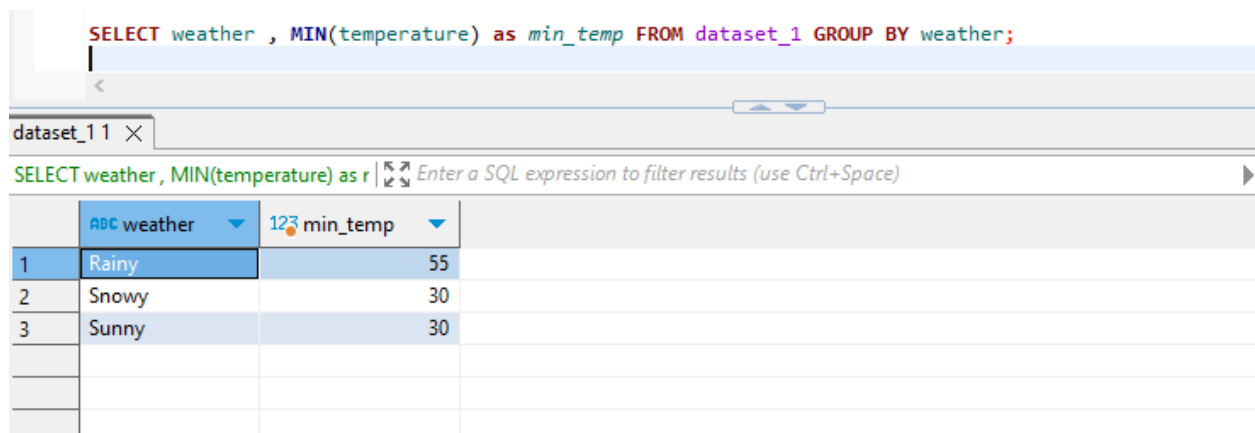
Find the sum of temperature by grouping the weather

```
] : # SELECT weather , SUM(temperature) as sum_temp FROM dataset_1 GROUP BY weather;  
  
dataset_1.groupby('weather')['temperature'].sum().reset_index(name='sum_temp')
```

```
] :
```

|   | weather | sum_temp |
|---|---------|----------|
| 0 | Rainy   | 66550    |
| 1 | Snowy   | 42150    |
| 2 | Sunny   | 694220   |

Find the minimum temperature by grouping the weather



The screenshot shows a SQL query editor with the following query: `SELECT weather , MIN(temperature) as min_temp FROM dataset_1 GROUP BY weather;` The query is executed, and the results are displayed in a table with two columns: 'weather' and 'min\_temp'. The results are as follows:

|   | weather | min_temp |
|---|---------|----------|
| 1 | Rainy   | 55       |
| 2 | Snowy   | 30       |
| 3 | Sunny   | 30       |

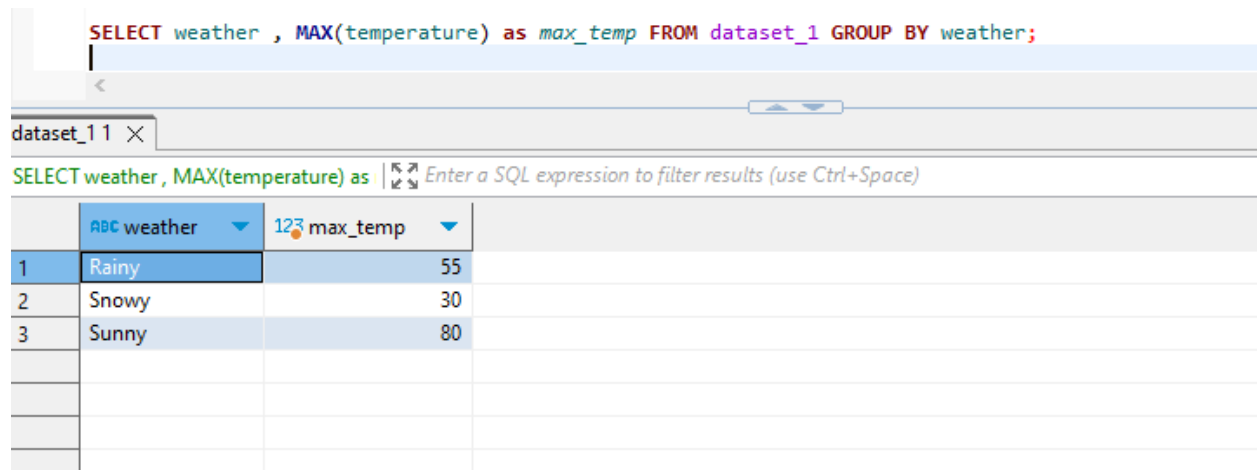
## Find the minimum temperature by grouping the weather

```
: # SELECT weather , MIN(temperature) as min_temp FROM dataset_1 GROUP BY weather;

dataset_1.groupby('weather')['temperature'].min().reset_index(name='min_temp')
```

```
:      weather  min_temp
0    Rainy         55
1    Snowy         30
2    Sunny         30
```

## Find the maximum temperature by grouping the weather



The screenshot shows a SQL editor with a query to find the maximum temperature by weather type. The query is: `SELECT weather , MAX(temperature) as max_temp FROM dataset_1 GROUP BY weather;`. Below the query, there is a table with 3 columns: `weather`, `max_temp`, and an unnamed column. The table has 3 rows of data: Rainy (55), Snowy (30), and Sunny (80).

|   | weather | max_temp |  |
|---|---------|----------|--|
| 1 | Rainy   | 55       |  |
| 2 | Snowy   | 30       |  |
| 3 | Sunny   | 80       |  |
|   |         |          |  |
|   |         |          |  |
|   |         |          |  |
|   |         |          |  |

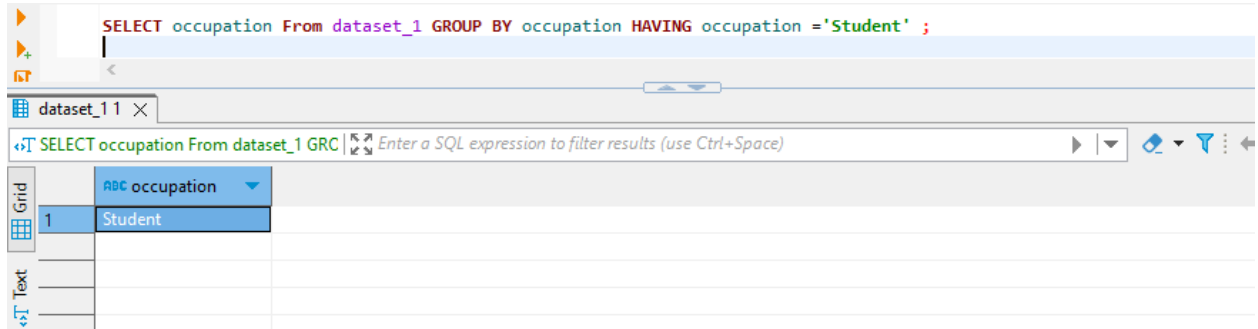
## Find the maximum temperature by grouping the weather

```
] : # SELECT weather , MAX(temperature) as max_temp FROM dataset_1 GROUP BY weather;

dataset_1.groupby('weather')['temperature'].max().reset_index(name='max_temp')
```

```
] :      weather  max_temp
0    Rainy         55
1    Snowy         30
2    Sunny         80
```

Groups the filtered DataFrame where 'occupation' is 'student' and counts the occurrences of each group



```
SELECT occupation From dataset_1 GROUP BY occupation HAVING occupation = 'Student' ;
```

dataset\_11

SELECT occupation From dataset\_1 GRC

|   | ABC occupation |
|---|----------------|
| 1 | Student        |

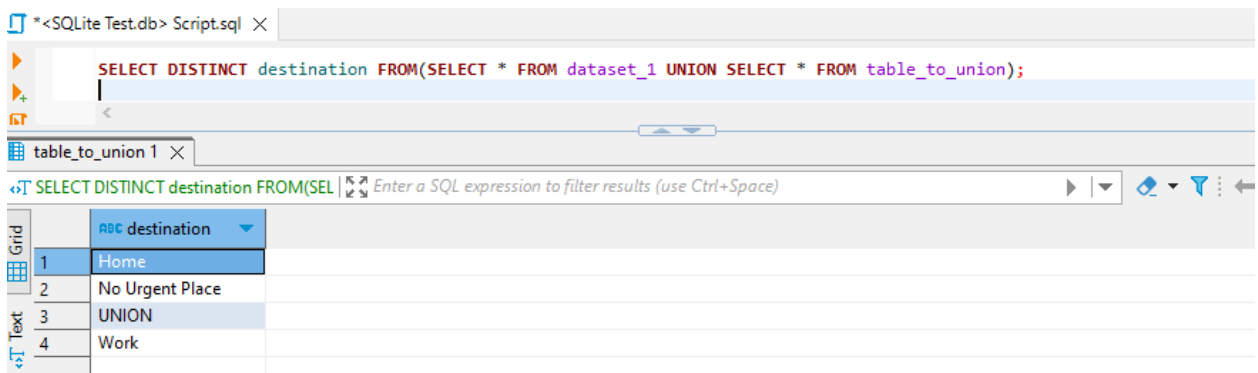
Groups the filtered DataFrame where 'occupation' is 'student' and counts the occurrences of each group.

```
# SELECT occupation From dataset_1 GROUP BY occupation HAVING occupation = 'Student' ;

dataset_1.groupby('occupation').filter(lambda x: x['occupation'].iloc[0] ==
'Student').groupby('occupation').size()

occupation
Student    1584
dtype: int64
```

Join the DataFrames and extract unique values from the 'Destination' column



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

table\_to\_union 1

SELECT DISTINCT destination FROM(SELECT

|   | ABC destination |
|---|-----------------|
| 1 | Home            |
| 2 | No Urgent Place |
| 3 | UNION           |
| 4 | Work            |

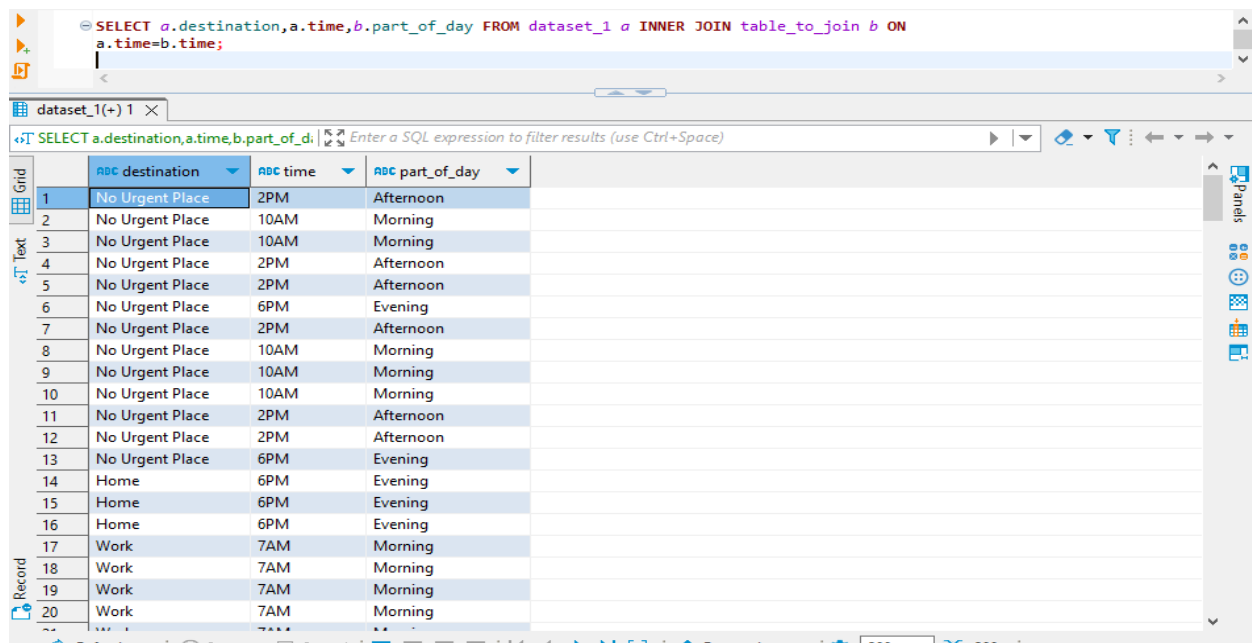
## Join the dataframes and extracting unique values from the 'Destination' column

```
[21]: # SELECT DISTINCT destination FROM(SELECT * FROM dataset_1 UNION SELECT * FROM table_to_union);
```

```
pd.concat([dataset_1, table_to_union])['Destination'].drop_duplicates()
```

```
[21]: 0    No Urgent Place
      13           Home
      16           Work
      0           NaN
      Name: Destination, dtype: object
```

## Merge the two DataFrames using inner join



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;`

dataset\_1(+1) X

SQL Expression: `SELECT a.destination, a.time, b.part_of_day`

|    | destination     | time | part_of_day |
|----|-----------------|------|-------------|
| 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     |
| 19 | Work            | 7AM  | Morning     |
| 20 | Work            | 7AM  | Morning     |

## Merge the two dataframes using inner join

```
# 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(dataset_1, table_to_join, 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       |

Select the column destination and passenger where the 'passenger' is 'Alone'

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

|    | ABC destination | ABC passanger |
|----|-----------------|---------------|
| 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         |
| 18 | Work            | Alone         |
| 19 | Work            | Alone         |
| 20 | Work            | Alone         |
| 21 | Work            | Alone         |

Select the column destination and passanger where the 'passenger' is 'Alone'

```
3]: # SELECT destination , passanger FROM(SELECT*FROM dataset_1 WHERE passanger = 'Alone');  
dataset_1[dataset_1['passanger'] == 'Alone'][['Destination', 'passanger']]
```

```
3]:
```

|       | 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     |

Retrieve the data that include only rows where the 'weather' column starts with the string 'Sun'.

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

|    | destination     | passanger | weather | 123 temperature | time | coupon                | expiration | gender |
|----|-----------------|-----------|---------|-----------------|------|-----------------------|------------|--------|
| 1  | No Urgent Place | Alone     | Sunny   | 55              | 2PM  | Restaurant(<20)       | 1d         | Femal  |
| 2  | No Urgent Place | Friend(s) | Sunny   | 80              | 10AM | Coffee House          | 2h         | Femal  |
| 3  | No Urgent Place | Friend(s) | Sunny   | 80              | 10AM | Carry out & Take away | 2h         | Femal  |
| 4  | No Urgent Place | Friend(s) | Sunny   | 80              | 2PM  | Coffee House          | 2h         | Femal  |
| 5  | No Urgent Place | Friend(s) | Sunny   | 80              | 2PM  | Coffee House          | 1d         | Femal  |
| 6  | No Urgent Place | Friend(s) | Sunny   | 80              | 6PM  | Restaurant(<20)       | 2h         | Femal  |
| 7  | No Urgent Place | Friend(s) | Sunny   | 55              | 2PM  | Carry out & Take away | 1d         | Femal  |
| 8  | No Urgent Place | Kid(s)    | Sunny   | 80              | 10AM | Restaurant(<20)       | 2h         | Femal  |
| 9  | No Urgent Place | Kid(s)    | Sunny   | 80              | 10AM | Carry out & Take away | 2h         | Femal  |
| 10 | No Urgent Place | Kid(s)    | Sunny   | 80              | 10AM | Bar                   | 1d         | Femal  |
| 11 | No Urgent Place | Kid(s)    | Sunny   | 80              | 2PM  | Restaurant(<20)       | 1d         | Femal  |
| 12 | No Urgent Place | Kid(s)    | Sunny   | 55              | 2PM  | Restaurant(<20)       | 1d         | Femal  |
| 13 | No Urgent Place | Kid(s)    | Sunny   | 55              | 6PM  | Coffee House          | 2h         | Femal  |
| 14 | Home            | Alone     | Sunny   | 55              | 6PM  | Bar                   | 1d         | Femal  |
| 15 | Home            | Alone     | Sunny   | 55              | 6PM  | Restaurant(20-50)     | 1d         | Femal  |
| 16 | Home            | Alone     | Sunny   | 80              | 6PM  | Coffee House          | 2h         | Femal  |
| 17 | Work            | Alone     | Sunny   | 55              | 7AM  | Coffee House          | 2h         | Femal  |
| 18 | Work            | Alone     | Sunny   | 55              | 7AM  | Bar                   | 1d         | Femal  |
| 19 | Work            | Alone     | Sunny   | 80              | 7AM  | Restaurant(20-50)     | 1d         | Femal  |
| 20 | Work            | Alone     | Sunny   | 80              | 7AM  | Carry out & Take away | 2h         | Femal  |



Retrieve the data that include only rows where the 'weather' column starts with the string 'Sun'.

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

[24]:

|       | Destination     | passanger | weather | temperature | time | coupon                | expiration | gender | age | maritalStatus     | ... | CarryAway | RestaurantLessThan20 | Restaurant2 |
|-------|-----------------|-----------|---------|-------------|------|-----------------------|------------|--------|-----|-------------------|-----|-----------|----------------------|-------------|
| 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                  |             |
| ...   | ...             | ...       | ...     | ...         | ...  | ...                   | ...        | ...    | ... | ...               | ... | ...       | ...                  |             |
| 12673 | Home            | Alone     | Sunny   | 30          | 6PM  | Carry out &           | 1d         | Male   | 26  | Single            | ... | 1~3       | 4~8                  |             |

Retrieve the unique data that include 'temperature' within the range from 29 to 75

The screenshot shows a data analysis interface. At the top, a SQL query is entered: `SELECT DISTINCT temperature FROM dataset_1 WHERE temperature BETWEEN 29 AND 75 ;`. Below the query, the results are displayed in a grid view. The grid has two columns: '123 temperature' and '55'. The first row contains the value '55', and the second row contains the value '30'.

▼ Retrieve the unique data that include 'temperature' within the range from 29 to 75

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

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

Select the 'occupation' column that include values 'Sales & Related' or 'Management'

The screenshot shows a data visualization tool interface. At the top, a SQL query is entered: `SELECT occupation FROM dataset_1 WHERE occupation IN('Sales & Related','Management');`. Below the query editor, a table titled 'dataset\_1' is displayed. The table has two columns: 'ABC occupation' and an unnamed column. The 'ABC occupation' column contains the value 'Sales & Related' for all 21 rows. The unnamed column is empty. The interface includes a search bar with the text 'Enter a SQL expression to filter results (use Ctrl+Space)', a 'Refresh' button, a 'Save' button, a 'Cancel' button, and an 'Export data' button. A pagination control shows '200' and '200+'.

|    | 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 |  |
| 18 | Sales & Related |  |
| 19 | Sales & Related |  |
| 20 | Sales & Related |  |
| 21 | Sales & Related |  |

Select the 'occupation' column that include values 'Sales & Related' or 'Management'

The screenshot shows a Jupyter Notebook cell with the following code: `# SELECT occupation FROM dataset_1 WHERE occupation IN('Sales & Related','Management');` and `dataset_1[dataset_1['occupation'].isin(['Sales & Related', 'Management'])]['occupation']`. The output of the code is a list of 'Sales & Related' values, indexed from 193 to 12682. The output is displayed in a table with two columns: an index and the value 'Sales & Related'.

```
[26]: # SELECT occupation FROM dataset_1 WHERE occupation IN('Sales & Related','Management');
dataset_1[dataset_1['occupation'].isin(['Sales & Related', 'Management'])]['occupation']
```

```
[26]:
```

|       | 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 |

