In [1]:
```
import pandas as pd
sql=pd.read_csv(r"C:\Users\DELL\Downloads\dataset_1_202508041133.csv")
```

In [2]:
```
sql
```

Out[2]:

	destination	passanger	weather	temperature	time	coupon	expiration
0	No Urgent Place	Alone	Sunny	55	2PM	Restaurant(<20)	1d
1	No Urgent Place	Friend(s)	Sunny	80	10AM	Coffee House	2h
2	No Urgent Place	Friend(s)	Sunny	80	10AM	Carry out & Take away	2h
3	No Urgent Place	Friend(s)	Sunny	80	2PM	Coffee House	2h
4	No Urgent Place	Friend(s)	Sunny	80	2PM	Coffee House	1d
...	...	...	...	...	...	...	...
12679	Home	Partner	Rainy	55	6PM	Carry out & Take away	1d
12680	Work	Alone	Rainy	55	7AM	Carry out & Take away	1d
12681	Work	Alone	Snowy	30	7AM	Coffee House	1d
12682	Work	Alone	Snowy	30	7AM	Bar	1d
12683	Work	Alone	Sunny	80	7AM	Restaurant(20-50)	2h

12684 rows × 27 columns

In [3]:
```
sql.head(11)
```

Out[3]:

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

11 rows × 27 columns

In [4]:
```
sql['passanger'].unique()
```

Out[4]:
```
array(['Alone', 'Friend(s)', 'Kid(s)', 'Partner'], dtype=object)
```

In [5]:
```
sql[['weather'],['temperature']]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
File D:\New folder\Lib\site-packages\pandas\core\indexes\base.py:3805, in Index.g
et_loc(self, key)
   3804 try:
-> 3805     return self._engine.get_loc(casted_key)
   3806 except KeyError as err:

File index.pyx:167, in pandas._libs.index.IndexEngine.get_loc()

File index.pyx:173, in pandas._libs.index.IndexEngine.get_loc()

TypeError: '(['weather'], ['temperature'])' is an invalid key

During handling of the above exception, another exception occurred:

InvalidIndexError                         Traceback (most recent call last)
Cell In[5], line 1
----> 1 sql[[         ],[              ]]

File D:\New folder\Lib\site-packages\pandas\core\frame.py:4102, in DataFrame.__ge
titem__(self, key)
   4100 if self.columns.nlevels > 1:
   4101     return self._getitem_multilevel(key)
-> 4102 indexer = self.columns.get_loc(key)
   4103 if is_integer(indexer):
   4104     indexer = [indexer]

File D:\New folder\Lib\site-packages\pandas\core\indexes\base.py:3817, in Index.g
et_loc(self, key)
   3812     raise KeyError(key) from err
   3813 except TypeError:
   3814     # If we have a listlike key, _check_indexing_error will raise
   3815     #  InvalidIndexError. Otherwise we fall through and re-raise
   3816     #  the TypeError.
-> 3817     self._check_indexing_error(key)
   3818     raise

File D:\New folder\Lib\site-packages\pandas\core\indexes\base.py:6059, in Index._
check_indexing_error(self, key)
   6055 def _check_indexing_error(self, key):
   6056     if not is_scalar(key):
   6057         # if key is not a scalar, directly raise an error (the code below
   6058         # would convert to numpy arrays and raise later any way) - GH2992
6
-> 6059         raise InvalidIndexError(key)

InvalidIndexError: (['weather'], ['temperature'])
```

In [6]:
```
sql[['weather','temperature']]
```

Out[6]:

	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

In [7]:
```
sql[sql['destination']=='Home']
```

Out[7]:

	destination	passanger	weather	temperature	time	coupon	expiration
13	Home	Alone	Sunny	55	6PM	Bar	1c
14	Home	Alone	Sunny	55	6PM	Restaurant(20-50)	1c
15	Home	Alone	Sunny	80	6PM	Coffee House	2h
35	Home	Alone	Sunny	55	6PM	Bar	1c
36	Home	Alone	Sunny	55	6PM	Restaurant(20-50)	1c
...	...	...	...	...	...	...	..
12675	Home	Alone	Snowy	30	10PM	Coffee House	2h
12676	Home	Alone	Sunny	80	6PM	Restaurant(20-50)	1c
12677	Home	Partner	Sunny	30	6PM	Restaurant(<20)	1c
12678	Home	Partner	Sunny	30	10PM	Restaurant(<20)	2h
12679	Home	Partner	Rainy	55	6PM	Carry out & Take away	1c

3237 rows × 27 columns

In [9]:
```
sql.sort_values('coupon')
```

Out[9]:

	destination	passanger	weather	temperature	time	coupon	expiration
11702	Home	Partner	Sunny	30	10PM	Bar	2h
9930	No Urgent Place	Alone	Snowy	30	2PM	Bar	1d
10632	Home	Alone	Rainy	55	6PM	Bar	1d
7997	No Urgent Place	Friend(s)	Rainy	55	10PM	Bar	2h
11166	Work	Alone	Snowy	30	7AM	Bar	1d
...	...	...	...	...	...	...	...
10476	Home	Alone	Sunny	80	6PM	Restaurant(<20)	1d
5447	Home	Alone	Sunny	80	10PM	Restaurant(<20)	2h
10478	Home	Alone	Snowy	30	10PM	Restaurant(<20)	2h
5440	No Urgent Place	Alone	Sunny	80	2PM	Restaurant(<20)	2h
0	No Urgent Place	Alone	Sunny	55	2PM	Restaurant(<20)	1d

12684 rows × 27 columns

In [10]:
```
sql.head()
```

Out[10]:

	destination	passanger	weather	temperature	time	coupon	expiration	ge
0	No Urgent Place	Alone	Sunny	55	2PM	Restaurant(<20)	1d	Fe
1	No Urgent Place	Friend(s)	Sunny	80	10AM	Coffee House	2h	Fe
2	No Urgent Place	Friend(s)	Sunny	80	10AM	Carry out & Take away	2h	Fe
3	No Urgent Place	Friend(s)	Sunny	80	2PM	Coffee House	2h	Fe
4	No Urgent Place	Friend(s)	Sunny	80	2PM	Coffee House	1d	Fe

5 rows × 27 columns

In [11]:
```
sql.rename(columns={'destination':'Destination'},inplace=True)
```

In [12]:
```
sql.head()
```

Out[12]:

	Destination	passanger	weather	temperature	time	coupon	expiration	ge
0	No Urgent Place	Alone	Sunny	55	2PM	Restaurant(<20)	1d	Fe
1	No Urgent Place	Friend(s)	Sunny	80	10AM	Coffee House	2h	Fe
2	No Urgent Place	Friend(s)	Sunny	80	10AM	Carry out & Take away	2h	Fe
3	No Urgent Place	Friend(s)	Sunny	80	2PM	Coffee House	2h	Fe
4	No Urgent Place	Friend(s)	Sunny	80	2PM	Coffee House	1d	Fe

5 rows × 27 columns

In [14]:
```
sql.groupby('occupation').size().to_frame('Count').reset_index()
```

Out[14]:

	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

In [19]:
```
sql['occupation'].value_counts()
```

Out[19]:
```
occupation
Unemployed                                    1870
Student                                       1584
Computer & Mathematical                       1408
Sales & Related                               1093
Education&Training&Library                     943
Management                                     838
Office & Administrative Support                639
Arts Design Entertainment Sports & Media       629
Business & Financial                           544
Retired                                        495
Food Preparation & Serving Related             298
Healthcare Practitioners & Technical           244
Healthcare Support                             242
Community & Social Services                    241
Legal                                          219
Transportation & Material Moving               218
Architecture & Engineering                     175
Personal Care & Service                        175
Protective Service                             175
Life Physical Social Science                   170
Construction & Extraction                      154
Installation Maintenance & Repair              133
Production Occupations                         110
Building & Grounds Cleaning & Maintenance       44
Farming Fishing & Forestry                      43
Name: count, dtype: int64
```

In [18]:
```
sql[sql['occupation'].value_counts()]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[18], line 1
----> 1 sql[sql[           ].value_counts()]

File D:\New folder\Lib\site-packages\pandas\core\frame.py:4108, in DataFrame.__ge
titem__(self, key)
   4106     if is_iterator(key):
   4107         key = list(key)
-> 4108     indexer = self.columns._get_indexer_strict(key,          )[1]
   4110 # take() does not accept boolean indexers
   4111 if getattr(indexer, "dtype", None) == bool:

File D:\New folder\Lib\site-packages\pandas\core\indexes\base.py:6200, in Index._
get_indexer_strict(self, key, axis_name)
   6197 else:
   6198     keyarr, indexer, new_indexer = self._reindex_non_unique(keyarr)
-> 6200 self._raise_if_missing(keyarr, indexer, axis_name)
   6202 keyarr = self.take(indexer)
   6203 if isinstance(key, Index):
   6204     # GH 42790 - Preserve name from an Index

File D:\New folder\Lib\site-packages\pandas\core\indexes\base.py:6249, in Index._
raise_if_missing(self, key, indexer, axis_name)
   6247 if nmissing:
   6248     if nmissing == len(indexer):
-> 6249         raise KeyError(f"None of [{key}] are in the [{axis_name}]")
   6251     not_found = list(ensure_index(key)[missing_mask.nonzero()[0]].unique
())
   6252     raise KeyError(f"{not_found} not in index")

KeyError: "None of [Index([1870, 1584, 1408, 1093,  943,  838,  639,  629,  544,
495,  298,  244,\n       242,  241,  219,  218,  175,  175,  175,  170,  154,  1
33,  110,   44,\n       43],\n      dtype='int64')] are in the [columns]"
```

In [17]:
```
sql.value_count('occupation')
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_16356\1773030575.py in ?()
----> 1 sql.value_count('occupation')

D:\New folder\Lib\site-packages\pandas\core\generic.py in ?(self, name)
   6295             and name not in self._accessors
   6296             and self._info_axis._can_hold_identifiers_and_holds_name(nam
e)
   6297         ):
   6298             return self[name]
-> 6299         return object.__getattribute__(self, name)

AttributeError: 'DataFrame' object has no attribute 'value_count'
```

In [20]:
```
sql['weather'].mean('temperature')
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
D:\New folder\Lib\site-packages\pandas\core\generic.py in ?(cls, axis)
    576             return cls._AXIS_TO_AXIS_NUMBER[axis]
    577         except KeyError:
--> 578             raise ValueError(f"No axis named {axis} for object type {cls.
__name__}")

KeyError: 'temperature'

During handling of the above exception, another exception occurred:

ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_16356\2547913748.py in ?()
----> 1 sql['weather'].mean('temperature')

D:\New folder\Lib\site-packages\pandas\core\series.py in ?(self, axis, skipna, nu
meric_only, **kwargs)
   6545         skipna: bool = True,
   6546         numeric_only: bool = False,
   6547         **kwargs,
   6548     ):
-> 6549         return NDFrame.mean(self, axis, skipna, numeric_only, **kwargs)

D:\New folder\Lib\site-packages\pandas\core\generic.py in ?(self, axis, skipna, n
umeric_only, **kwargs)
  12416         skipna: bool_t = True,
  12417         numeric_only: bool_t = False,
  12418         **kwargs,
  12419     ) -> Series | float:
> 12420         return self._stat_function(
  12421             "mean", nanops.nanmean, axis, skipna, numeric_only, **kwargs
  12422         )

D:\New folder\Lib\site-packages\pandas\core\generic.py in ?(self, name, func, axi
s, skipna, numeric_only, **kwargs)
  12373         nv.validate_func(name, (), kwargs)
  12374
  12375         validate_bool_kwarg(skipna, "skipna", none_allowed=False)
  12376
> 12377         return self._reduce(
  12378             func, name=name, axis=axis, skipna=skipna, numeric_only=numer
ic_only
  12379         )

D:\New folder\Lib\site-packages\pandas\core\series.py in ?(self, op, name, axis,
skipna, numeric_only, filter_type, **kwds)
   6435         """
   6436         delegate = self._values
   6437
   6438         if axis is not None:
-> 6439             self._get_axis_number(axis)
   6440
   6441         if isinstance(delegate, ExtensionArray):
   6442             # dispatch to ExtensionArray interface

D:\New folder\Lib\site-packages\pandas\core\generic.py in ?(cls, axis)
    574     def _get_axis_number(cls, axis: Axis) -> AxisInt:
    575         try:
    576             return cls._AXIS_TO_AXIS_NUMBER[axis]
```

```
    577         except KeyError:
--> 578             raise ValueError(f"No axis named {axis} for object type {cls.
__name__}")

ValueError: No axis named temperature for object type Series
```

In [21]:
```
sql.head()
```

Out[21]:

	Destination	passanger	weather	temperature	time	coupon	expiration	ge
0	No Urgent Place	Alone	Sunny	55	2PM	Restaurant(<20)	1d	Fe
1	No Urgent Place	Friend(s)	Sunny	80	10AM	Coffee House	2h	Fe
2	No Urgent Place	Friend(s)	Sunny	80	10AM	Carry out & Take away	2h	Fe
3	No Urgent Place	Friend(s)	Sunny	80	2PM	Coffee House	2h	Fe
4	No Urgent Place	Friend(s)	Sunny	80	2PM	Coffee House	1d	Fe

5 rows × 27 columns

In [22]:
```
mean('temperature')
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[22], line 1
----> 1 mean('temperature')

NameError: name 'mean' is not defined
```

In [23]:
```
from numpy import mean
```

In [24]:
```
mean('temperature')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[24], line 1
----> 1 mean(              )

File D:\New folder\Lib\site-packages\numpy\_core\fromnumeric.py:3904, in mean(a,
axis, dtype, out, keepdims, where)
   3901     else:
   3902         return mean(axis=axis, dtype=dtype, out=out, **kwargs)
-> 3904 return _methods._mean(a, axis=axis, dtype=dtype,
   3905                       out=out, **kwargs)

File D:\New folder\Lib\site-packages\numpy\_core\_methods.py:136, in _mean(a, axi
s, dtype, out, keepdims, where)
    133         dtype = mu.dtype('f4')
    134         is_float16_result = True
--> 136 ret = umr_sum(arr, axis, dtype, out, keepdims, where=where)
    137 if isinstance(ret, mu.ndarray):
    138     with _no_nep50_warning():

TypeError: the resolved dtypes are not compatible with add.reduce. Resolved (dtyp
e('<U11'), dtype('<U11'), dtype('<U22'))
```

In [25]:
```
sql.mean('temperature')
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
D:\New folder\Lib\site-packages\pandas\core\generic.py in ?(cls, axis)
    576             return cls._AXIS_TO_AXIS_NUMBER[axis]
    577         except KeyError:
--> 578             raise ValueError(f"No axis named {axis} for object type {cls.
__name__}")

KeyError: 'temperature'

During handling of the above exception, another exception occurred:

ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_16356\4188329231.py in ?()
----> 1 sql.mean('temperature')

D:\New folder\Lib\site-packages\pandas\core\frame.py in ?(self, axis, skipna, num
eric_only, **kwargs)
  11689         skipna: bool = True,
  11690         numeric_only: bool = False,
  11691         **kwargs,
  11692     ):
> 11693         result = super().mean(axis, skipna, numeric_only, **kwargs)
  11694         if isinstance(result, Series):
  11695             result = result.__finalize__(self, method="mean")
  11696         return result

D:\New folder\Lib\site-packages\pandas\core\generic.py in ?(self, axis, skipna, n
umeric_only, **kwargs)
  12416         skipna: bool_t = True,
  12417         numeric_only: bool_t = False,
  12418         **kwargs,
  12419     ) -> Series | float:
> 12420         return self._stat_function(
  12421             "mean", nanops.nanmean, axis, skipna, numeric_only, **kwargs
  12422         )

D:\New folder\Lib\site-packages\pandas\core\generic.py in ?(self, name, func, axi
s, skipna, numeric_only, **kwargs)
  12373         nv.validate_func(name, (), kwargs)
  12374
  12375         validate_bool_kwarg(skipna, "skipna", none_allowed=False)
  12376
> 12377         return self._reduce(
  12378             func, name=name, axis=axis, skipna=skipna, numeric_only=numer
ic_only
  12379         )

D:\New folder\Lib\site-packages\pandas\core\frame.py in ?(self, op, name, axis, s
kipna, numeric_only, filter_type, **kwds)
  11446         assert filter_type is None or filter_type == "bool", filter_type
  11447         out_dtype = "bool" if filter_type == "bool" else None
  11448
  11449         if axis is not None:
> 11450             axis = self._get_axis_number(axis)
  11451
  11452         def func(values: np.ndarray):
  11453             # We only use this in the case that operates on self.values

D:\New folder\Lib\site-packages\pandas\core\generic.py in ?(cls, axis)
```

```
    574     def _get_axis_number(cls, axis: Axis) -> AxisInt:
    575         try:
    576             return cls._AXIS_TO_AXIS_NUMBER[axis]
    577         except KeyError:
--> 578             raise ValueError(f"No axis named {axis} for object type {cls.
__name__}")

ValueError: No axis named temperature for object type DataFrame
```

In [26]:
```
sql.groupby('weather')['temperature'].mean().to_frame('avg_temp').reset_index()
```

Out[26]:

	weather	avg_temp
0	Rainy	55.000000
1	Snowy	30.000000
2	Sunny	68.946271

In [28]:
```
sql.groupby('weather')['temperature'].mean().to_frame('avg_temp')
```

Out[28]:

weather	avg_temp
Rainy	55.000000
Snowy	30.000000
Sunny	68.946271

In [30]:
```
sql.groupby('weather')['temperature'].mean().to_frame('avg_temp').reset_index()
```

Out[30]:

	weather	avg_temp
0	Rainy	55.000000
1	Snowy	30.000000
2	Sunny	68.946271

In [32]:
```
sql.groupby('weather')['temperature'].count().to_frame('count_temp').reset_index
```

Out[32]:

	weather	count_temp
0	Rainy	1210
1	Snowy	1405
2	Sunny	10069

In [34]:
```
sql.groupby('weather')['temperature'].unique().to_frame('count_distinct_temp').r
```

Out[34]:

	weather	count_distinct_temp
0	Rainy	[55]
1	Snowy	[30]
2	Sunny	[55, 80, 30]

In [35]:
```
sql.groupby('weather')['temperature'].nunique().to_frame('count_distinct_temp').
```

Out[35]:

	weather	count_distinct_temp
0	Rainy	1
1	Snowy	1
2	Sunny	3

In [36]:
```
sql.groupby('weather')['temperature']
```

Out[36]:
```
<pandas.core.groupby.generic.SeriesGroupBy object at 0x00000148DFE32450>
```

In [37]:
```
sql.groupby('weather')['temperature'].nunique()
```

Out[37]:
```
weather
Rainy    1
Snowy    1
Sunny    3
Name: temperature, dtype: int64
```

In [38]:
```
sql.groupby('weather')['temperature'].nunique().reset_index()
```

Out[38]:

	weather	temperature
0	Rainy	1
1	Snowy	1
2	Sunny	3

In [39]:
```
sql.groupby('weather')['temperature'].sum().to_frame('sum_temp').reset_index()
```

Out[39]:

	weather	sum_temp
0	Rainy	66550
1	Snowy	42150
2	Sunny	694220

In [41]:
```
sql.groupby('weather')['temperature'].min().to_frame('min_temp').reset_index()
```

Out[41]:

	weather	min_temp
0	Rainy	55
1	Snowy	30
2	Sunny	30

In [42]:
```
sql.groupby('weather')['temperature'].max().to_frame('max_temp').reset_index()
```

Out[42]:

	weather	max_temp
0	Rainy	55
1	Snowy	30
2	Sunny	80

In [43]:
```
sql.groupby('occupation').filter(lambda x:x['occupation'].iloc[0]=='Student').gr
```

Out[43]:
```
occupation
Student    1584
dtype: int64
```

In [44]:
```
pd.concat([sql,sql1])['destination'].drop_duplicates()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[44], line 1
----> 1 pd.concat([sql,sql1])['destination'].drop_duplicates()

NameError: name 'sql1' is not defined
```

In [45]:
```
pd.merge(sql,sql2[['time','part_of_day']],on='time',how='inner')[['destination',
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[45], line 1
----> 1 pd.merge(sql,sql2[['time','part_of_day']],on='time',how='inner')[['destin
ation','time','part_of_day']]

NameError: name 'sql2' is not defined
```

In [46]:
```
sql[sql['passanger']=='Alone'][['destination','passanger']]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[46], line 1
----> 1 sql[sql[           ]==       ][[             ,           ]]

File D:\New folder\Lib\site-packages\pandas\core\frame.py:4108, in DataFrame.__ge
titem__(self, key)
   4106     if is_iterator(key):
   4107         key = list(key)
-> 4108     indexer = self.columns._get_indexer_strict(key,          )[1]
   4110 # take() does not accept boolean indexers
   4111 if getattr(indexer, "dtype", None) == bool:

File D:\New folder\Lib\site-packages\pandas\core\indexes\base.py:6200, in Index._
get_indexer_strict(self, key, axis_name)
   6197 else:
   6198     keyarr, indexer, new_indexer = self._reindex_non_unique(keyarr)
-> 6200 self._raise_if_missing(keyarr, indexer, axis_name)
   6202 keyarr = self.take(indexer)
   6203 if isinstance(key, Index):
   6204     # GH 42790 - Preserve name from an Index

File D:\New folder\Lib\site-packages\pandas\core\indexes\base.py:6252, in Index._
raise_if_missing(self, key, indexer, axis_name)
   6249     raise KeyError(f"None of [{key}] are in the [{axis_name}]")
   6251 not_found = list(ensure_index(key)[missing_mask.nonzero()[0]].unique())
-> 6252 raise KeyError(f"{not_found} not in index")

KeyError: "['destination'] not in index"
```

In [47]:
```
sql.head()
```

Out[47]:

	Destination	passanger	weather	temperature	time	coupon	expiration	ge
0	No Urgent Place	Alone	Sunny	55	2PM	Restaurant(<20)	1d	Fe
1	No Urgent Place	Friend(s)	Sunny	80	10AM	Coffee House	2h	Fe
2	No Urgent Place	Friend(s)	Sunny	80	10AM	Carry out & Take away	2h	Fe
3	No Urgent Place	Friend(s)	Sunny	80	2PM	Coffee House	2h	Fe
4	No Urgent Place	Friend(s)	Sunny	80	2PM	Coffee House	1d	Fe

5 rows × 27 columns

In [48]:
```
sql[sql['passanger']=='Alone'][['Destination','passanger']]
```

Out[48]:

	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

In [49]:
```
sql[sql['passanger']=='Alone']
```

Out[49]:

	Destination	passanger	weather	temperature	time	coupon	expiration
0	No Urgent Place	Alone	Sunny	55	2PM	Restaurant(<20)	1d
13	Home	Alone	Sunny	55	6PM	Bar	1d
14	Home	Alone	Sunny	55	6PM	Restaurant(20-50)	1d
15	Home	Alone	Sunny	80	6PM	Coffee House	2h
16	Work	Alone	Sunny	55	7AM	Coffee House	2h
...	...	...	...	...	...	...	...
12676	Home	Alone	Sunny	80	6PM	Restaurant(20-50)	1d
12680	Work	Alone	Rainy	55	7AM	Carry out & Take away	1d
12681	Work	Alone	Snowy	30	7AM	Coffee House	1d
12682	Work	Alone	Snowy	30	7AM	Bar	1d
12683	Work	Alone	Sunny	80	7AM	Restaurant(20-50)	2h

7305 rows × 27 columns

In [50]:
```
sql.tail()
```

Out[50]:

	Destination	passanger	weather	temperature	time	coupon	expiration
12679	Home	Partner	Rainy	55	6PM	Carry out & Take away	1d
12680	Work	Alone	Rainy	55	7AM	Carry out & Take away	1d
12681	Work	Alone	Snowy	30	7AM	Coffee House	1d
12682	Work	Alone	Snowy	30	7AM	Bar	1d
12683	Work	Alone	Sunny	80	7AM	Restaurant(20-50)	2h

5 rows × 27 columns

In [51]:
```
sql[sql['weather']['Sunny']]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[51], line 1
----> 1 sql[sql[         ][         ]]

File D:\New folder\Lib\site-packages\pandas\core\series.py:1121, in Series.__geti
tem__(self, key)
   1118     return self._values[key]
   1120 elif key_is_scalar:
-> 1121     return self._get_value(key)
   1123 # Convert generator to list before going through hashable part
   1124 # (We will iterate through the generator there to check for slices)
   1125 if is_iterator(key):

File D:\New folder\Lib\site-packages\pandas\core\series.py:1237, in Series._get_v
alue(self, label, takeable)
   1234     return self._values[label]
   1236 # Similar to Index.get_value, but we do not fall back to positional
-> 1237 loc = self.index.get_loc(label)
   1239 if is_integer(loc):
   1240     return self._values[loc]

File D:\New folder\Lib\site-packages\pandas\core\indexes\range.py:417, in RangeIn
dex.get_loc(self, key)
    415         raise KeyError(key) from err
    416 if isinstance(key, Hashable):
--> 417     raise KeyError(key)
    418 self._check_indexing_error(key)
    419 raise KeyError(key)

KeyError: 'Sunny'
```

In [52]:
```
sql[sql['weather']=='Sunny']
```

Out[52]:

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

10069 rows × 27 columns

In [55]:
```
sql[sql['temperature']>=29 & (sql['temperature']<=75)]['temperature'].unique()
```

Out[55]:
```
array([55, 80, 30])
```

In [56]:
```
sql[sql['occupation'].isin(['Sales & Related','Management'])][['occupation']]
```

Out[56]:

	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

In [ ]: