Chapter 3 Filtering

In [1]:

```
import pandas as pd

# Read titanic dataset
tnc = pd.read_csv("./datasets/titanic.csv")

# Print dataframe
tnc.head()
```

Out[1]:

	pclass	survived	name	gender	age	sibsp	parch	ticket	fare	cabin	em
0	1	1	Allen, Miss. Elisabeth Walton	female	29	0	0	24160	211.3375	B5	
1	1	1	Allison, Master. Hudson Trevor	male	0.9167	1	2	113781	151.55	C22 C26	
2	1	0	Allison, Miss. Helen Loraine	female	2	1	2	113781	151.55	C22 C26	
3	1	0	Allison, Mr. Hudson Joshua Creighton	male	30	1	2	113781	151.55	C22 C26	
4	1	0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25	1	2	113781	151.55	C22 C26	

Filter records using boolean series

We can filter out rows by passing boolean series as an argument to the Dataframe. All the rows that have corresponding boolean value as True are extracted.

Syntax: df[boolean_series]

In [2]:

```
# Generate boolean series for first 5 passengers
boolean = [True, False, True, True, False]
```

```
# Print only the records that corresponds to True
tnc.head()[boolean]
```

Out[2]:

	pclass	survived	name	gender	age	sibsp	parch	ticket	fare	cabin	emba
0	1	1	Allen, Miss. Elisabeth Walton	female	29	0	0	24160	211.3375	B5	
2	1	0	Allison, Miss. Helen Loraine	female	2	1	2	113781	151.55	C22 C26	
3	1	0	Allison, Mr. Hudson Joshua Creighton	male	30	1	2	113781	151.55	C22 C26	

To generate boolean series we can take the help of comparison operators in python

Useful comparison operators are ==, !=, >, >=, < and <=

Ex: df["age"] == 18

In [3]:

```
# Generate boolean series for male passengers
tnc["gender"] == "male"
```

Out[3]:

```
0        False
1         True
2        False
3         True
4        False
         ...  
1304     False
1305     False
1306      True
1307      True
1308      True
Name: gender, Length: 1309, dtype: bool
```

In [4]:

```
# Filter out male passengers from the dataframe using the series
tnc[tnc["gender"] == "male"]
```

Out[4]:

	pclass	survived	name	gender	age	sibsp	parch	ticket	fare	cabin
1	1	1	Allison, Master. Hudson Trevor	male	0.9167	1	2	113781	151.55	C22 C26
3	1	0	Allison, Mr. Hudson Joshua Creighton	male	30	1	2	113781	151.55	C22 C26
5	1	1	Anderson, Mr. Harry	male	48	0	0	19952	26.55	E12
7	1	0	Andrews, Mr. Thomas Jr	male	39	0	0	112050	0	A36
9	1	0	Artagaveytia, Mr. Ramon	male	71	0	0	PC 17609	49.5042	?
...	...	...	...	...	...	...	...	...	...	...
1302	3	0	Yousif, Mr. Wazli	male	?	0	0	2647	7.225	?
1303	3	0	Yousseff, Mr. Gerious	male	?	0	0	2627	14.4583	?
1306	3	0	Zakarian, Mr. Mapriededer	male	26.5	0	0	2656	7.225	?
1307	3	0	Zakarian, Mr. Ortin	male	27	0	0	2670	7.225	?
1308	3	0	Zimmerman, Mr. Leo	male	29	0	0	315082	7.875	?

843 rows × 14 columns

In [5]:

```
# Find the count of male passengers
len(tnc[tnc["gender"] == "male"])
```

Out[5]:

```
843
```

In [6]:

```
# Extract the passengers who survived
tnc[tnc["survived"] == 1]
```

Out[6]:

	pclass	survived	name	gender	age	sibsp	parch	ticket	fare	cabin
0	1	1	Allen, Miss. Elisabeth Walton	female	29	0	0	24160	211.3375	B5
1	1	1	Allison, Master. Hudson Trevor	male	0.9167	1	2	113781	151.55	C22 C26
5	1	1	Anderson, Mr. Harry	male	48	0	0	19952	26.55	E12
6	1	1	Andrews, Miss. Kornelia Theodosia	female	63	1	0	13502	77.9583	D7
8	1	1	Appleton, Mrs. Edward Dale (Charlotte Lamson)	female	53	2	0	11769	51.4792	C101
...	...	...	...	...	...	...	...	...	...	...
1261	3	1	Turkula, Mrs. (Hedwig)	female	63	0	0	4134	9.5875	
1277	3	1	Vartanian, Mr. David	male	22	0	0	2658	7.225	
1286	3	1	Whabee, Mrs. George Joseph (Shawneene Abi-Saab)	female	38	0	0	2688	7.2292	
1290	3	1	Wilkes, Mrs. James (Ellen Needs)	female	47	1	0	363272	7	
1300	3	1	Yasbeck, Mrs. Antoni (Selini Alexander)	female	15	1	0	2659	14.4542	

500 rows × 14 columns

In [7]:

```
# Extract the passengers who didn't survive
tnc[tnc["survived"] != 1]
```

Out[7]:

	pclass	survived	name	gender	age	sibsp	parch	ticket	fare	cabin
2	1	0	Allison, Miss. Helen Loraine	female	2	1	2	113781	151.55	C22 C26
3	1	0	Allison, Mr. Hudson Joshua Creighton	male	30	1	2	113781	151.55	C22 C26
4	1	0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25	1	2	113781	151.55	C22 C26
7	1	0	Andrews, Mr. Thomas Jr	male	39	0	0	112050	0	A36
9	1	0	Artagaveytia, Mr. Ramon	male	71	0	0	PC 17609	49.5042	?
...	...	...	...	...	...	...	...	...	...	...
1304	3	0	Zabour, Miss. Hileni	female	14.5	1	0	2665	14.4542	?
1305	3	0	Zabour, Miss. Thamine	female	?	1	0	2665	14.4542	?
1306	3	0	Zakarian, Mr. Mapriededer	male	26.5	0	0	2656	7.225	?
1307	3	0	Zakarian, Mr. Ortin	male	27	0	0	2670	7.225	?
1308	3	0	Zimmerman, Mr. Leo	male	29	0	0	315082	7.875	?

809 rows × 14 columns

In [8]:

```
# Extract the passengers whose age is greater than or equal to 18
# tnc[tnc["age"] >= 18]
```

In [9]:

```
# Verify that the datatype of age is string (python object)
tnc.age
```

```
Out[9]: 0           29
        1       0.9167
        2            2
        3           30
        4           25
                 ...
        1304      14.5
        1305         ?
        1306      26.5
        1307        27
        1308        29
        Name: age, Length: 1309, dtype: object
```

```
In [10]: # Since it is an object we cannot perform comparison operators on strings.
         # Only equality operations can be performed on strings.
         tnc[tnc.age == "18"]
```

Out[10]:

	pclass	survived	name	gender	age	sibsp	parch	ticket	fare	cabi
11	1	1	Astor, Mrs. John Jacob (Madeleine Talmadge Force)	female	18	1	0	PC 17757	227.525	C6 C6
198	1	1	Marvin, Mrs. Daniel Warner (Mary Graham Carmic...	female	18	1	0	113773	53.1	D3
228	1	0	Penasco y Castellana, Mr. Victor de Satode	male	18	1	0	PC 17758	108.9	C6
250	1	1	Ryerson, Miss. Emily Borie	female	18	2	2	PC 17608	262.375	B5 B5 B6 B6
270	1	1	Smith, Mrs. Lucien Philip (Mary Eloise Hughes)	female	18	1	0	13695	60	C3
289	1	1	Taussig, Miss. Ruth	female	18	0	2	110413	79.65	E6
326	2	0	Andrew, Mr. Edgardo Samuel	male	18	0	0	231945	11.5	
331	2	0	Bailey, Mr. Percy Andrew	male	18	0	0	29108	11.5	
386	2	0	Davies, Mr. Charles Henry	male	18	0	0	S.O.C. 14879	73.5	
394	2	0	Dibden, Mr. William	male	18	0	0	S.O.C. 14879	73.5	
395	2	1	Doling, Miss. Elsie	female	18	0	1	231919	23	
405	2	0	Fahlstrom, Mr. Arne Jonas	male	18	0	0	236171	13	
408	2	0	Fillbrook, Mr. Joseph Charles	male	18	0	0	C.A. 15185	10.5	
445	2	0	Hiltunen, Miss. Marta	female	18	1	1	250650	13	

	pclass	survived	name	gender	age	sibsp	parch	ticket	fare	cabi
558	2	1	Silven, Miss. Lyyli Karoliina	female	18	0	2	250652	13	
607	3	1	Abrahim, Mrs. Joseph (Sophie Halaut Easu)	female	18	0	0	2657	7.2292	
612	3	1	Aks, Mrs. Sam (Leah Rosen)	female	18	0	1	392091	9.35	
619	3	0	Allum, Mr. Owen George	male	18	0	0	2223	8.3	
636	3	0	Arnold-Franchi, Mrs. Josef (Josefine Franchi)	female	18	1	0	349237	17.8	
661	3	1	Badman, Miss. Emily Louisa	female	18	0	0	A/4 31416	8.05	
665	3	0	Barbara, Miss. Saiide	female	18	0	1	2691	14.4542	
676	3	0	Bjorklund, Mr. Ernst Herbert	male	18	0	0	347090	7.75	
695	3	0	Burns, Miss. Mary Delia	female	18	0	0	330963	7.8792	
698	3	0	Cacic, Mr. Jego Grga	male	18	0	0	315091	8.6625	
717	3	0	Chronopoulos, Mr. Demetrios	male	18	1	0	2680	14.4542	
719	3	1	Cohen, Mr. Gurshon 'Gus'	male	18	0	0	A/5 3540	8.05	
786	3	0	Edvardsson, Mr. Gustaf Hjalmar	male	18	0	0	349912	7.775	
799	3	0	Fischer, Mr. Eberhard Thelander	male	18	0	0	350036	7.7958	
809	3	0	Ford, Mr. Edward Watson	male	18	2	2	W./C. 6608	34.375	

	pclass	survived	name	gender	age	sibsp	parch	ticket	fare	cabi
859	3	0	Hegarty, Miss. Hanora 'Nora'	female	18	0	0	365226	6.75	
938	3	0	Klasen, Mr. Klas Albin	male	18	1	1	350404	7.8542	
1045	3	0	Myhrman, Mr. Pehr Fabian Oliver Malkolm	male	18	0	0	347078	7.75	
1060	3	1	Nilsson, Miss. Berta Olivia	female	18	0	0	347066	7.775	
1130	3	0	Pettersson, Miss. Ellen Natalia	female	18	0	0	347087	7.775	
1157	3	0	Rosblom, Mr. Viktor Richard	male	18	1	1	370129	20.2125	
1205	3	1	Sjoblom, Miss. Anna Sofia	female	18	0	0	3101265	7.4958	
1260	3	1	Turja, Miss. Anna Sofia	female	18	0	0	4138	9.8417	
1273	3	0	Vander Planke, Miss. Augusta Maria	female	18	2	0	345764	18	
1288	3	0	Wiklund, Mr. Jakob Alfred	male	18	1	0	3101267	6.4958	

In [11]:

```
# Check for numeric columns in dataframe that do not contain '?'
def check(col):
    for row in tnc[col]:
        if row == '?':
            return False
    return True

for col in tnc.columns:
    if(check(col) and tnc[col].dtype == "int64"):
        print(col)
```

```
pclass
survived
sibsp
parch
```

In [12]:

```
# Filter records whose count of siblings or spouse is greater than 1
tnc[tnc["sibsp"] > 1]
```

Out[12]:

	pclass	survived	name	gender	age	sibsp	parch	ticket	fare	cabin	er
8	1	1	Appleton, Mrs. Edward Dale (Charlotte Lamson)	female	53	2	0	11769	51.4792	C101	
42	1	1	Brown, Mrs. John Murray (Caroline Lane Lamson)	female	59	2	0	11769	51.4792	C101	
79	1	1	Cornell, Mrs. Robert Clifford (Malvina Helen L...	female	55	2	0	11770	25.7	C101	
111	1	1	Fortune, Miss. Alice Elizabeth	female	24	3	2	19950	263	C23 C25 C27	
112	1	1	Fortune, Miss. Ethel Flora	female	28	3	2	19950	263	C23 C25 C27	
...	...	...	...	...	...	...	...	...	...	...	
1208	3	0	Skoog, Miss. Mabel	female	9	3	2	347088	27.9	?	
1209	3	0	Skoog, Miss. Margit Elizabeth	female	2	3	2	347088	27.9	?	
1273	3	0	Vander Planke, Miss. Augusta Maria	female	18	2	0	345764	18	?	
1274	3	0	Vander Planke, Mr. Julius	male	31	3	0	345763	18	?	
1275	3	0	Vander Planke, Mr. Leo Edmondus	male	16	2	0	345764	18	?	

99 rows × 14 columns

In [13]:

```
# Read houses dataset
houses = pd.read_csv("./datasets/kc_house_data.csv")

# Print dataframe
houses.head()
```

Out[13]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	fl
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	

5 rows × 21 columns

In [14]:

```
# Extract houses whose price is greater than 5,000,000
price_cond = houses["price"] > 5_000_000
houses[price_cond]
```

Out[14]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lo
1164	1247600105	20141020T000000	5110800.0	5	5.25	8010	45517
1315	7558700030	20150413T000000	5300000.0	6	6.00	7390	24829
1448	8907500070	20150413T000000	5350000.0	5	5.00	8000	23985
3914	9808700762	20140611T000000	7062500.0	5	4.50	10040	37325
4411	2470100110	20140804T000000	5570000.0	5	5.75	9200	35069
7252	6762700020	20141013T000000	7700000.0	6	8.00	12050	27600
9254	9208900037	20140919T000000	6885000.0	6	7.75	9890	31374

7 rows × 21 columns

In [15]:

```
# Extract houses whose no. of bedrooms is greater than or equal to 10
bdroom_cond = houses["bedrooms"] >= 10
houses[bdroom_cond]
```

Out[15]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_l
8757	1773100755	20140821T000000	520000.0	11	3.00	3000	49
13314	627300145	20140814T000000	1148000.0	10	5.25	4590	109
15161	5566100170	20141029T000000	650000.0	10	2.00	3610	119
15870	2402100895	20140625T000000	640000.0	33	1.75	1620	60
19254	8812401450	20141229T000000	660000.0	10	3.00	2920	37

5 rows × 21 columns

In [16]:

```
# Extract houses whose no. of bathrooms is less than 1
bathroom_cond = houses["bathrooms"] < 1
houses[bathroom_cond]
```

Out[16]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lo
208	1222000055	20141123T000000	180250.0	2	0.75	900	960
264	2123039032	20141027T000000	369900.0	1	0.75	760	1007
350	7325600160	20140604T000000	299000.0	1	0.75	560	1212
465	8658300340	20140523T000000	80000.0	1	0.75	430	505
569	4045500710	20141218T000000	405000.0	2	0.75	1160	1502
...	...	...	...	...	...	...	.
19344	2114700090	20150301T000000	151000.0	2	0.75	720	504
19452	3980300371	20140926T000000	142000.0	0	0.00	290	2087
20017	1025039168	20140923T000000	290000.0	1	0.75	740	128
21610	1523300141	20140623T000000	402101.0	2	0.75	1020	135
21612	1523300157	20141015T000000	325000.0	2	0.75	1020	107

86 rows × 21 columns

In [17]:

```
# Extract houses whose grade is less than or equal to 7
grd_cond = houses["grade"] <= 7
houses[grd_cond]
```

```
Out[17]:
              id             date     price  bedrooms  bathrooms  sqft_living  sqft_lo
    0   7129300520  20141013T000000  221900.0         3       1.00         1180     565
    1   6414100192  20141209T000000  538000.0         3       2.25         2570     724
    2   5631500400  20150225T000000  180000.0         2       1.00          770    1000
    3   2487200875  20141209T000000  604000.0         4       3.00         1960     500
    6   1321400060  20140627T000000  257500.0         3       2.25         1715     681
  ...          ...              ...       ...       ...        ...          ...       .
21584    952006823  20141202T000000  380000.0         3       2.50         1260      90
21585   3832050760  20140828T000000  270000.0         3       2.50         1870     500
21602    844000965  20140626T000000  224000.0         3       1.75         1500    1196
21610   1523300141  20140623T000000  402101.0         2       0.75         1020     135
21612   1523300157  20141015T000000  325000.0         2       0.75         1020     107

11294 rows × 21 columns
```

Filter records within a range

To filter records within a range we make use of the method:

Dataframe.column.between(min, max)

Ex: df["age"].between(13, 19)

```
In [18]:
# Extract houses whose latitude lie in the range 47.30 to 47.35
lat_cond = houses["lat"].between(47.30, 47.35)
houses[lat_cond]
```

Out[18]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lo
6	1321400060	20140627T000000	257500.0	3	2.25	1715	681
18	16000397	20141205T000000	189000.0	2	1.00	1200	985
19	7983200060	20150424T000000	230000.0	3	1.00	1250	977
22	7137970340	20140703T000000	285000.0	5	2.50	2270	630
25	1202000200	20141103T000000	233000.0	3	2.00	1710	469
...	...	...	...	...	...	...	.
21479	6181500120	20140623T000000	312891.0	5	3.00	2300	821
21511	3304030220	20150421T000000	480000.0	4	2.50	2940	917
21585	3832050760	20140828T000000	270000.0	3	2.50	1870	500
21589	7570050450	20140910T000000	347500.0	3	2.50	2540	476
21602	844000965	20140626T000000	224000.0	3	1.75	1500	1196

1340 rows × 21 columns

Filter records whose values lie in a set

To filter records based on whether their values lie in a set we make use of the method:

Dataframe.column.isin([val1, val2, ..., valN])

Ex: df["age"].isin(10,20,30,40)

In [19]:

```
# Extract houses built in the years 1940, 1965 and 2008
yr_cond = houses["yr_built"].isin([1940, 1965, 2008])
houses[yr_cond]
```

Out[19]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_l
3	2487200875	20141209T000000	604000.0	4	3.00	1960	50
10	1736800520	20150403T000000	662500.0	3	2.50	3560	97
56	9478500640	20140819T000000	292500.0	4	2.50	2250	44
69	1802000060	20140612T000000	1325000.0	5	2.25	3200	201
74	3444100400	20150316T000000	349000.0	3	1.75	1790	505
...	...	...	...	...	...	...	
21594	5087900040	20141017T000000	350000.0	4	2.75	2500	59
21600	249000205	20141015T000000	1537000.0	5	3.75	4470	80
21601	5100403806	20150407T000000	467000.0	3	2.50	1425	11
21607	2997800021	20150219T000000	475000.0	3	2.50	1310	12
21612	1523300157	20141015T000000	325000.0	2	0.75	1020	10

710 rows × 21 columns

Filter records by combining conditions

We can combine the multiple conditions used while filtering using the bitwise **AND &** and **OR |** operators.

Ex: Return records of illegal child marriages of Indian girl children

```
teenage = df["age"] <= 21
indian = df["country"] == "India"
girl = df["gender"] == "female"
married = df["married"] == True
```

Combine the logic:

```
df[married & indian & teenage & girl]
```

In [20]:

```
# Extract male passengers who survived having siblings or spouse count greater than
male = tnc["gender"] == "male"
double = tnc["sibsp"] >= 2
survived = tnc["survived"] == 1

# Combine all the logic
tnc[male & survived & double]
```

Out[20]:

	pclass	survived	name	gender	age	sibsp	parch	ticket	fare	cabin	e
119	1	1	Frauenthal, Dr. Henry William	male	50	2	0	PC 17611	133.65	?	
249	1	1	Ryerson, Master. John Borie	male	13	2	2	PC 17608	262.375	B57 B59 B63 B66	
339	2	1	Becker, Master. Richard F	male	1	2	1	230136	39	F4	
641	3	1	Asplund, Master. Edvin Rojj Felix	male	3	4	2	347077	31.3875	?	
935	3	1	Kink-Heilmann, Mr. Anton	male	29	3	1	315153	22.025	?	
1003	3	1	McCoy, Mr. Bernard	male	?	2	0	367226	23.25	?	

In [21]:

```python
# Extract passengers who are male of pclass not equal to 1 (or) are women of pclass
male = tnc["gender"] == "male"
male_pclass = tnc["pclass"] != 1

female = tnc["gender"] == "female"
female_pclass = tnc["pclass"] == 1

tnc[(male & male_pclass) | (female & female_pclass)]
```

Out[21]:

	pclass	survived	name	gender	age	sibsp	parch	ticket	fare	cabin
0	1	1	Allen, Miss. Elisabeth Walton	female	29	0	0	24160	211.3375	B5
2	1	0	Allison, Miss. Helen Loraine	female	2	1	2	113781	151.55	C22 C26
4	1	0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25	1	2	113781	151.55	C22 C26
6	1	1	Andrews, Miss. Kornelia Theodosia	female	63	1	0	13502	77.9583	D7
8	1	1	Appleton, Mrs. Edward Dale (Charlotte Lamson)	female	53	2	0	11769	51.4792	C101
...	...	...	...	...	...	...	...	...	...	...
1302	3	0	Yousif, Mr. Wazli	male	?	0	0	2647	7.225	?
1303	3	0	Yousseff, Mr. Gerious	male	?	0	0	2627	14.4583	?
1306	3	0	Zakarian, Mr. Mapriededer	male	26.5	0	0	2656	7.225	?
1307	3	0	Zakarian, Mr. Ortin	male	27	0	0	2670	7.225	?
1308	3	0	Zimmerman, Mr. Leo	male	29	0	0	315082	7.875	?

808 rows × 14 columns

Filter records that do not meet the required conditions

To extract all the records that doesn't satisfy the required conditions we can make use of bitwise negation ~ operator.

In [22]:

```
# Extract houses that are new
```

```python
# Note that a house is said to be new if it is built or renovated in the year 2014
yr_blt = houses["yr_built"] >= 2014
yr_rnt = houses["yr_renovated"] >= 2014

houses[yr_blt | yr_rnt]
```

Out[22]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lo
42	7203220400	20140707T000000	861990.0	5	2.75	3595	563
63	9528102996	20141207T000000	549000.0	3	1.75	1540	104
133	8929000270	20140512T000000	453246.0	3	2.50	2010	228
139	2767603505	20140507T000000	519950.0	3	2.25	1170	124
211	1025049114	20140717T000000	625504.0	3	2.25	1270	156
...	...	...	...	...	...	...	.
21598	8956200760	20141013T000000	541800.0	4	2.50	3118	786
21602	844000965	20140626T000000	224000.0	3	1.75	1500	1196
21604	9834201367	20150126T000000	429000.0	3	2.00	1490	112
21605	3448900210	20141014T000000	610685.0	4	2.50	2520	602
21609	6600060120	20150223T000000	400000.0	4	2.50	2310	581

704 rows × 21 columns

In [23]:

```python
# Extract houses that are old
# Note that a house is said to be old if it is not new
houses[~(yr_blt | yr_rnt)]
```

Out[23]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lo
0	7129300520	20141013T000000	221900.0	3	1.00	1180	565
1	6414100192	20141209T000000	538000.0	3	2.25	2570	724
2	5631500400	20150225T000000	180000.0	2	1.00	770	1000
3	2487200875	20141209T000000	604000.0	4	3.00	1960	500
4	1954400510	20150218T000000	510000.0	3	2.00	1680	808
...	...	...	...	...	...	...	.
21607	2997800021	20150219T000000	475000.0	3	2.50	1310	129
21608	263000018	20140521T000000	360000.0	3	2.50	1530	113
21610	1523300141	20140623T000000	402101.0	2	0.75	1020	135
21611	291310100	20150116T000000	400000.0	3	2.50	1600	238
21612	1523300157	20141015T000000	325000.0	2	0.75	1020	107

20909 rows × 21 columns

Filter records based on whether they are null or non-null

Note: NaN (Not a Number) refers to missing or null values in pandas nomenclature.

We can filter records if they are NaN (null values) using the **Dataframe.column.isna()** method.

And if they are not NaN (non null values) using the **Dataframe.column.notna()** method.

In [24]:

```
# Read sales dataset
sales = pd.read_csv("./datasets/sales.csv")

# Print dataframe
sales
```

Out[24]:

	rating	shipping_zip	billing_zip
0	5.0	NaN	81220.0
1	4.5	94931.0	94931.0
2	NaN	92625.0	92625.0
3	4.5	10003.0	10003.0
4	4.0	NaN	92660.0
5	NaN	NaN	NaN
6	NaN	60007.0	60007.0

In [25]:

```
# Extract sales that have null ratings
na = sales["rating"].isna()
sales[na]
```

Out[25]:

	rating	shipping_zip	billing_zip
2	NaN	92625.0	92625.0
5	NaN	NaN	NaN
6	NaN	60007.0	60007.0

In [26]:

```
# Extract sales that have non-null shipping_zip
na = sales["shipping_zip"].notna()
sales[na]
```

Out[26]:

	rating	shipping_zip	billing_zip
1	4.5	94931.0	94931.0
2	NaN	92625.0	92625.0
3	4.5	10003.0	10003.0
6	NaN	60007.0	60007.0