In [65]:
```
#Label Encoding
df.classification = df.classification.astype('category').cat.codes
df.head()
```

Out[65]:

	education	age	capital-gain	race	capital-loss	hours-per-week	gender	classification
0	Bachelors	39	2174.0	White	0	40	1	0
1	Bachelors	50	NaN	White	0	13	1	0
2	HS-grad	38	NaN	White	0	40	1	0
3	11th	53	NaN	Black	0	40	1	0
4	Bachelors	28	0.0	Black	0	40	0	0

Categorical Ordinal Encoding

In [66]:
```
from pandas.api.types import CategoricalDtype
categories=['Preschool', '1st-4th', '5th-6th', '7th-8th', '10th', '9th', '12th', '11th',
            'Some-college', 'HS-grad', 'Bachelors', 'Masters', 'Doctorate']
Dtype = CategoricalDtype(categories=categories, ordered=True)
```

In [67]:
```
df.education=df.education.astype(Dtype).cat.codes
```

In [68]:
```
df.head()
```

Out[68]:

	education	age	capital-gain	race	capital-loss	hours-per-week	gender	classification
0	10	39	2174.0	White	0	40	1	0
1	10	50	NaN	White	0	13	1	0
2	9	38	NaN	White	0	40	1	0
3	7	53	NaN	Black	0	40	1	0
4	10	28	0.0	Black	0	40	0	0

One Hot Encoding

In [47]:
```
df = pd.get_dummies(df,columns=['race'])
```

In [48]:
```
df.head()
```

Out[48]:

	education	age	capital-gain	capital-loss	hours-per-week	gender	classification	race_Amer-Indian-Eskimo	race_Asian-Pac-Islander	race_Black	race_Other	race_White
0	Bachelors	39	2174.0	0	40	1	0	0	0	0	0	1
1	Bachelors	50	NaN	0	13	1	0	0	0	0	0	1
2	HS-grad	38	NaN	0	40	1	0	0	0	0	0	1
3	11th	53	NaN	0	40	1	0	0	0	1	0	0
4	Bachelors	28	0.0	0	40	0	0	0	0	1	0	0

TASK #1: Data Integration

In [3]:
```
uber1 = pd.read_csv('uber1.csv')
uber2 = pd.read_csv('uber2.csv')
uber3 = pd.read_csv('uber3.csv')
```

In [7]:
```
uber1.head()
```

Out[7]:

	Unnamed: 0	Date/Time	Lat	Lon	Base
0	42	5/1/2014 3:50:00	40.7299	-73.9868	B02512
1	43	5/1/2014 3:57:00	40.7762	-73.9499	B02512
2	44	5/1/2014 3:58:00	40.7538	-73.9774	B02512
3	45	5/1/2014 3:58:00	40.6819	-73.6850	B02512
4	46	5/1/2014 4:04:00	40.7580	-73.9892	B02512

In [8]:
```
# Concatenate uber1, uber2, and uber3: row_concat
row_concat = pd.concat([uber1,uber2,uber3])
```

In [10]:
```
row_concat.shape
```

Out[10]: (312, 5)

In [11]:
```

df1 = pd.DataFrame(
    {
        "A": ["A0", "A1", "A2", "A3"],
        "B": ["B0", "B1", "B2", "B3"],
        "C": ["C0", "C1", "C2", "C3"],
        "D": ["D0", "D1", "D2", "D3"],
    },
    index=[0, 1, 2, 3],
)
df2 = pd.DataFrame(
    {
        "A": ["A4", "A5", "A6", "A7"],
        "B": ["B4", "B5", "B6", "B7"],
        "C": ["C4", "C5", "C6", "C7"],
        "D": ["D4", "D5", "D6", "D7"],
    },
    index=[4, 5, 6, 7],
)
df3 = pd.DataFrame(
    {
        "B": ["B2", "B3", "B6", "B7"],
        "D": ["D2", "D3", "D6", "D7"],
        "F": ["F2", "F3", "F6", "F7"],
    },
    index=[2, 3, 6, 7],
)
```

In [14]:
```
print(df1.head())
print(df2.head())
print(df3.head())

```

```
    A   B   C   D
0  A0  B0  C0  D0
1  A1  B1  C1  D1
2  A2  B2  C2  D2
3  A3  B3  C3  D3
    A   B   C   D
4  A4  B4  C4  D4
5  A5  B5  C5  D5
6  A6  B6  C6  D6
7  A7  B7  C7  D7
    B   D   F
2  B2  D2  F2
3  B3  D3  F3
6  B6  D6  F6
7  B7  D7  F7
```

In [15]:
```
pd.concat([df1, df3], axis=1)
```

Out[15]:

	A	B	C	D	B	D	F
0	A0	B0	C0	D0	NaN	NaN	NaN
1	A1	B1	C1	D1	NaN	NaN	NaN
2	A2	B2	C2	D2	B2	D2	F2
3	A3	B3	C3	D3	B3	D3	F3
6	NaN	NaN	NaN	NaN	B6	D6	F6
7	NaN	NaN	NaN	NaN	B7	D7	F7

In [16]:
```
pd.concat([df1, df3], axis=1, join="inner")
```

Out[16]:

	A	B	C	D	B	D	F
2	A2	B2	C2	D2	B2	D2	F2
3	A3	B3	C3	D3	B3	D3	F3

In [17]:
```
# Merging
left = pd.DataFrame(
    {
        "key": ["K0", "K1", "K2", "K3"],
        "A": ["A0", "A1", "A2", "A3"],
        "B": ["B0", "B1", "B2", "B3"],
    }
)


right = pd.DataFrame(
    {
        "key": ["K0", "K1", "K2", "K3"],
        "C": ["C0", "C1", "C2", "C3"],
        "D": ["D0", "D1", "D2", "D3"],
    }
)
```

In [18]:
```
pd.merge(left, right, on="key")
```

Out[18]:

	key	A	B	C	D
0	K0	A0	B0	C0	D0
1	K1	A1	B1	C1	D1
2	K2	A2	B2	C2	D2
3	K3	A3	B3	C3	D3

In [19]:
```
left = pd.DataFrame(
    {
        "key1": ["K0", "K0", "K1", "K2"],
        "key2": ["K0", "K1", "K0", "K1"],
        "A": ["A0", "A1", "A2", "A3"],
        "B": ["B0", "B1", "B2", "B3"],
    }
)


right = pd.DataFrame(
    {
        "key1": ["K0", "K1", "K1", "K2"],
        "key2": ["K0", "K0", "K0", "K0"],
        "C": ["C0", "C1", "C2", "C3"],
        "D": ["D0", "D1", "D2", "D3"],
    }
)
```

In [20]:
```
pd.merge(left, right, how="left", on=["key1", "key2"])
```

Out[20]:

	key1	key2	A	B	C	D
0	K0	K0	A0	B0	C0	D0
1	K0	K1	A1	B1	NaN	NaN
2	K1	K0	A2	B2	C1	D1
3	K1	K0	A2	B2	C2	D2
4	K2	K1	A3	B3	NaN	NaN

In Python Pandas, concat and merge are used to combine dataframes in different ways.

concat is used to concatenate dataframes along either rows (axis=0) or columns (axis=1). The concatenated dataframes simply have their rows or columns stacked on top of each other, with no regard for shared columns or indices.

merge, on the other hand, is used to combine dataframes based on common columns or indices. The resulting dataframe contains only the rows where there is a match between the two input dataframes. The merge operation allows you to specify the type of join to perform (e.g., inner join, outer join, left join, right join) and how to handle overlapping data.

In general, concat is a simpler operation and is useful when you want to combine dataframes without any regard for shared columns or indices. merge is more powerful and is useful when you want to combine dataframes based on shared columns or indices.

In [22]:
```
#Reshaping the data using melt
airquality = pd.read_csv('airquality.csv')
# Print the head of airquality
airquality.head()

```

Out[22]:

	Ozone	Solar.R	Wind	Temp	Month	Day
0	41.0	190.0	7.4	67	5	1
1	36.0	118.0	8.0	72	5	2
2	12.0	149.0	12.6	74	5	3
3	18.0	313.0	11.5	62	5	4
4	NaN	NaN	14.3	56	5	5

In [24]:
```
# Melt airquality: airquality_melt
airquality_melt = pd.melt(airquality, id_vars=['Month', 'Day'])

```

In [25]:
```
# Print the head of airquality_melt
airquality_melt.head()
```

Out[25]:

	Month	Day	variable	value
0	5	1	Ozone	41.0
1	5	2	Ozone	36.0
2	5	3	Ozone	12.0
3	5	4	Ozone	18.0
4	5	5	Ozone	NaN

In [26]:
```
# Melt airquality: airquality_melt
airquality_melt = pd.melt(airquality, id_vars=['Month', 'Day'],
                          var_name='measurement', value_name='reading')

```

In [27]:
```
# Print the head of airquality_melt
airquality_melt.head()
```

Out[27]:

	Month	Day	measurement	reading
0	5	1	Ozone	41.0
1	5	2	Ozone	36.0
2	5	3	Ozone	12.0
3	5	4	Ozone	18.0
4	5	5	Ozone	NaN

Pivot Data

In [28]:
```
airquality_pivot = airquality_melt.pivot_table(index=['Month', 'Day'],
                                               columns='measurement',
                                               values='reading')
```

In [29]:
```
airquality_pivot.head()
```

Out[29]:

	measurement	Ozone	Solar.R	Temp	Wind
Month	**Day**				
	1	41.0	190.0	67.0	7.4
	2	36.0	118.0	72.0	8.0
5	**3**	12.0	149.0	74.0	12.6
	4	18.0	313.0	62.0	11.5
	5	NaN	NaN	56.0	14.3

In [30]:
```
# Reset the index of airquality_pivot: airquality_pivot
airquality_pivot = airquality_pivot.reset_index()
```