1. **Create a dictionary for student database and perform following operations.**

Dictionary For students

In [2]:
```
import pandas as pd
```

In [14]:
```
student = {
    "Name":["Rahul","Nayan","Priyanshi","Monika","Parnal","Neeraj"],
    "Year":"Third",
    "Age":['20','21','22','20','18',23],
    "Marks":[90,92,98,100,85,27]
}
```

In [15]:
```
df = pd.DataFrame(student)
```

In [17]:
```
# Top 3 rows
print(df.head(3))
```

```
        Name   Year Age  Marks
0      Rahul  Third  20     90
1      Nayan  Third  21     92
2  Priyanshi  Third  22     98
```

In [18]:
```
# Last 3 rows
print(df.tail(3))
```

```
     Name   Year Age  Marks
3  Monika  Third  20    100
4  Parnal  Third  18     85
5  Neeraj  Third  23     27
```

In [19]:
```
print(df.shape)
```

```
(6, 4)
```

In [22]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   Name    6 non-null      object
 1   Year    6 non-null      object
 2   Age     6 non-null      object
 3   Marks   6 non-null      int64 
dtypes: int64(1), object(3)
memory usage: 320.0+ bytes
```

In [23]:
```
# Overall Statisctics of the database
df.describe()
```

Out[23]:

	Marks
count	6.000000
mean	82.000000
std	27.488179
min	27.000000
25%	86.250000
50%	91.000000
75%	96.500000
max	100.000000

In [24]:
```
# Check Null values inside the db.
df.isnull().sum()
```

Out[24]:
```
Name      0
Year      0
Age       0
Marks     0
dtype: int64
```

In [34]:
```
#Find total no of students having marks between 90 to 100.
df[df['Marks'].between(90,100)][['Name','Marks']]
```

Out[34]:

	Name	Marks
0	Rahul	90
1	Nayan	92
2	Priyanshi	98
3	Monika	100

2.Perform following operation on Ecommerce purchase website

1. Display Top 10 Rows of The Dataset

2. Display last 10 rows of dataset.

```
In [1]: import pandas as pd
        import numpy as np
        df=pd.read_csv('C:/Users/khyati/Documents/pandas python/Ecommerce-Purchases')
        df.head(10)
```

Out[1]:

	Address	Lot	AM or PM	Browser Info	Company	Credit Card	CC Exp Date	CC Security Code	CC Provider	Email	Job
0	16629 Pace Camp Apt. 448\nAlexisborough, NE 77...	46 in	PM	Opera/9.56.(X11; Linux x86_64; sl-SI) Presto/2...	Martinez-Herman	6011929061123406	02/20	900	JCB 16 digit	pdunlap@yahoo.com	Scientist, product/process development
1	9374 Jasmine Spurs Suite 508\nSouth John, TN 8...	28 rn	PM	Opera/8.93.(Windows 98; Win 9x 4.90; en-US) Pr...	Fletcher, Richards and Whitaker	3337758169645356	11/18	561	Mastercard	anthony41@reed.com	Drilling engineer
2	Unit 0065 Box 5052\nDPO AP 27450	94 vE	PM	Mozilla/5.0 (compatible; MSIE 9.0; Windows NT ...	Simpson, Williams and Pham	675957666125	08/19	699	JCB 16 digit	amymiller@morales-harrison.com	Customer service manager
3	7780 Julia Fords\nNew Stacy, WA 45798	36 vm	PM	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_0 ...	Williams, Marshall and Buchanan	6011578504430710	02/24	384	Discover	brent16@olson-robinson.info	Drilling engineer

```
In [7]: #2.Check Last 10 Rows of The Dataset
        df.tail(10)
```

Out[7]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hours-per-week	native-country	income
48832	32	Private	34066	10th	6	Married-civ-spouse	Handlers-cleaners	Husband	Amer-Indian-Eskimo	Male	0	0	40	United-States	<=50K
48833	43	Private	84661	Assoc-voc	11	Married-civ-spouse	Sales	Husband	White	Male	0	0	45	United-States	<=50K
48834	32	Private	116138	Masters	14	Never-married	Tech-support	Not-in-family	Asian-Pac-Islander	Male	0	0	11	Taiwan	<=50K

3. Check Datatype of Each Column

4. Check null values in the dataset

```
In [19]: #3.Check Datatype of Each Column
         print(df.dtypes)
```

```
Address               object
Lot                   object
AM or PM              object
Browser Info          object
Company               object
Credit Card            int64
CC Exp Date           object
CC Security Code       int64
CC Provider           object
Email                 object
Job                   object
IP Address            object
Language              object
Purchase Price       float64
dtype: object
```

```
In [21]: #4.Check null values in the dataset
         df.isnull().count()
```

```
Out[21]: Address              10000
         Lot                  10000
         AM or PM             10000
         Browser Info         10000
         Company              10000
         Credit Card          10000
         CC Exp Date          10000
         CC Security Code     10000
         CC Provider          10000
         Email                10000
         Job                  10000
         IP Address           10000
         Language             10000
         Purchase Price       10000
         dtype: int64
```

5. How many rows and columns are there in our Dataset? 6. Highest and Lowest Purchase Prices.

7. Average Purchase Price 8. How many people have French 'fr' as their Language?

```
In [22]: #5.How many rows and columns are there in our Dataset?
         df.shape

Out[22]: (10000, 14)

In [26]: #6.Highest and Lowest Purchase Prices.
         print(df['Purchase Price'].max())
         print(df['Purchase Price'].min())

         99.99
         0.0

In [29]: #7.Average Purchase Price
         print(df['Purchase Price'].mean())

         50.347302

In [ ]:

In [41]: #8.How many people have French 'fr' as their Language?
         len(df[df['Language'] == 'fr'])

Out[41]: 1097
```

9. Job Title Contains Engineer

10. Find The Email of the person with the following IP Address: 132.207.160.22 11. How many People have Mastercard as their Credit Card Provider and made a purchase above 50.?

```
In [43]: #9.How many people have French 'fr' as their Language?
         df[df['Language'] == 'fr'].count()

Out[43]: Address             1097
         Lot                 1097
         AM or PM            1097
         Browser Info        1097
         Company             1097
         Credit Card         1097
         CC Exp Date         1097
         CC Security Code    1097
         CC Provider         1097
         Email               1097
         Job                 1097
         IP Address          1097
         Language            1097
         Purchase Price      1097
         dtype: int64

In [ ]:

In [49]: #10.Job Title Contains Engineer
         len(df[df['Job'].str.contains('engineer',case = False)])

Out[49]: 984

In [52]: #11.Find The Email of the person with the following IP Address: 132.207.160.22
         df[df['IP Address']=='132.207.160.22'].Email

Out[52]: 2    amymiller@morales-harrison.com
         Name: Email, dtype: object
```

12. Find the email of the person with the following Credit Card Number: 4664825258997302

13. How many people purchase during the AM and how many people purchase during PM?

14. How many people have a credit card that expires in 2020?

15. What are the top 5 most popular email providers (e.g. gmail.com, yahoo.com, etc...)

```
In [56]: #13.Find the email of the person with the following Credit Card Number: 4664825258997302
         df[df['Credit Card']==4664825258997302].Email

Out[56]: 9992    bberry@wright.net
         Name: Email, dtype: object

In [62]: #14.How many people purchase during the AM and how many people purchase during PM?
         df['AM or PM'].value_counts()

Out[62]: PM    5068
         AM    4932
         Name: AM or PM, dtype: int64

In [ ]:

In [65]: #15.How many people have a credit card that expires in 2020?
         len(df[df['CC Exp Date'].apply(lambda x:x[3:]=='20')])

Out[65]: 988
```

```
In [69]: #16.What are the top 5 most popular email providers (e.g. gmail.com, yahoo.com, etc...)
         list = []
         for email in df['Email']:

             list.append(email.split('@')[1])

In [71]: df['Email Provider'] = list

In [72]: df['Email Provider'].value_counts()

Out[72]: hotmail.com           1638
         yahoo.com             1616
         gmail.com             1605
         smith.com               42
         williams.com            37
                                ...
         booker.com               1
         woods-allen.biz          1
         richards-wilson.com      1
         morris-thomas.com        1
         wade-garner.com          1
         Name: Email Provider, Length: 3416, dtype: int64

In [74]: df['Email Provider'].value_counts().head(5)

Out[74]: hotmail.com    1638
         yahoo.com      1616
         gmail.com      1605
         smith.com        42
         williams.com     37
         Name: Email Provider, dtype: int64
```

3 .Perform following operation on Employee salary dataset

4. Getting Information About Our Dataset Like Total Number Rows, Total Number of Columns, Datatypes of Each Column And Memory Requirement

```
In [16]: #4. Getting Information About Our Dataset Like Total Number Rows, Total Number of Columns,
         #Datatypes of Each Column And Memory Requirement
         df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 148654 entries, 0 to 148653
Data columns (total 13 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   Id                148654 non-null  int64
 1   EmployeeName      148654 non-null  object
 2   JobTitle          148654 non-null  object
 3   BasePay           148049 non-null  object
 4   OvertimePay       148654 non-null  object
 5   OtherPay          148654 non-null  object
 6   Benefits          112495 non-null  object
 7   TotalPay          148654 non-null  float64
 8   TotalPayBenefits  148654 non-null  float64
 9   Year              148654 non-null  int64
 10  Notes             0 non-null       float64
 11  Agency            148654 non-null  object
 12  Status            38119 non-null   object
dtypes: float64(3), int64(2), object(8)
memory usage: 14.7+ MB
```

5. Check Null Values In The Dataset 6. Drop ID, Notes, Agency, and Status Columns

```
In [7]: #5. Check Null Values In The Dataset
        df.isnull().sum()
```

```
Out[7]: Id                     0
        EmployeeName           0
        JobTitle               0
        BasePay              605
        OvertimePay            0
        OtherPay               0
        Benefits           36159
        TotalPay               0
        TotalPayBenefits       0
        Year                   0
        Notes             148654
        Agency                 0
        Status            110535
        dtype: int64
```

```
In [23]: #6. Drop ID, Notes, Agency, and Status Columns
         df.drop(['Id','Notes','Agency','Status'],axis=1,inplace=True)
         df.columns
```

```
Out[23]: Index(['EmployeeName', 'JobTitle', 'BasePay', 'OvertimePay', 'OtherPay',
                'Benefits', 'TotalPay', 'TotalPayBenefits', 'Year'],
               dtype='object')
```

7. Get Overall Statistics About The Dataframe 8. Find Occurrence of The Employee Names (Top 5) 9. Find The Number of Unique Job Titles

```
In [24]: #7. Get Overall Statistics About The Dataframe
         df.describe(include='all')
```

Out[24]:

	EmployeeName	JobTitle	BasePay	OvertimePay	OtherPay	Benefits	TotalPay	TotalPayBenefits	Year
count	148654	148654	148049.0	148654.0	148654.0	112495.0	148654.000000	148654.000000	148654.000000
unique	110811	2159	109900.0	66555.0	84968.0	99635.0	NaN	NaN	NaN
top	Kevin Lee	Transit Operator	0.0	0.0	0.0	0.0	NaN	NaN	NaN
freq	13	7036	875.0	66103.0	35218.0	1053.0	NaN	NaN	NaN
mean	NaN	NaN	NaN	NaN	NaN	NaN	74768.321972	93692.554811	2012.522643
std	NaN	NaN	NaN	NaN	NaN	NaN	50517.005274	62793.533483	1.117538
min	NaN	NaN	NaN	NaN	NaN	NaN	-618.130000	-618.130000	2011.000000
25%	NaN	NaN	NaN	NaN	NaN	NaN	36168.995000	44065.650000	2012.000000
50%	NaN	NaN	NaN	NaN	NaN	NaN	71426.610000	92404.090000	2013.000000
75%	NaN	NaN	NaN	NaN	NaN	NaN	105839.135000	132876.450000	2014.000000
max	NaN	NaN	NaN	NaN	NaN	NaN	567595.430000	567595.430000	2014.000000

```
In [25]: #8. Find Occurrence of The Employee Names (Top 5)
         df['EmployeeName'].value_counts().head(5)
```

```
Out[25]: Kevin Lee      13
         Richard Lee    11
         Steven Lee     11
         William Wong   11
         Stanley Lee     9
         Name: EmployeeName, dtype: int64
```

```
In [26]: #9. Find The Number of Unique Job Titles
         df['JobTitle'].nunique()
```

```
Out[26]: 2159
```

10. Total Number of Job Titles Contain Captain

11. Display All the Employee Names From Fire Department

```
In [29]: #10.Total Number of Job Titles Contain Captain
def capString(x):
    if "captain" in x.lower():
        return True
    else:
        return False

sum(df["JobTitle"].apply(lambda x: capString(x)))
```

```
Out[29]: 552
```

```
In [37]: #11. Display All the Employee Names From Fire Department
df[df['JobTitle'].str.contains('FIRE DEPARTMENT', case=False)]
```

Out[37]:

	EmployeeName	JobTitle	BasePay	OvertimePay	OtherPay	Benefits	TotalPay	TotalPayBenefits	Year
4	PATRICK GARDNER	DEPUTY CHIEF OF DEPARTMENT,(FIRE DEPARTMENT)	134401.6	9737.0	182234.59	NaN	326373.19	326373.19	2011
6	ALSON LEE	BATTALION CHIEF, (FIRE DEPARTMENT)	92492.01	89062.9	134426.14	NaN	315981.05	315981.05	2011
8	MICHAEL MORRIS	BATTALION CHIEF, (FIRE DEPARTMENT)	176932.64	86362.68	40132.23	NaN	303427.55	303427.55	2011
9	JOANNE HAYES-WHITE	CHIEF OF DEPARTMENT, (FIRE DEPARTMENT)	285262.0	0.0	17115.73	NaN	302377.73	302377.73	2011
10	ARTHUR KENNEY	ASSISTANT CHIEF OF DEPARTMENT, (FIRE DEPARTMENT)	194999.39	71344.88	33149.9	NaN	299494.17	299494.17	2011
...	...	...	...	...	...	...	...	...	...
32623	JAMES BARDEN	BATTALION CHIEF, (FIRE DEPARTMENT)	0.0	1290.69	5802.68	NaN	7093.37	7093.37	2011
36162	Joanne Hayes-White	Chief, Fire Department	296943.01	0.0	17816.59	72047.88	314759.60	386807.48	2012
72926	Joanne M Hayes-White	Chief, Fire Department	313686.01	0.0	23236.0	85431.39	336922.01	422353.40	2013

Find Minimum, Maximum, and Average BasePay 13. Replace 'Not Provided' inEmployeeName' Column to NaN

```
In [39]: #12. Find Minimum, Maximum, and Average BasePay
x=df['TotalPay'].max()
y=df['TotalPay'].min()
z=df['TotalPay'].mean()
print(x ,y, z)
```

```
567595.43 -618.13 74768.32197169267
```

```
In [3]: #13. Replace 'Not Provided' in EmployeeName' Column to NaN
df['EmployeeName'] = df['EmployeeName'].replace(r'(?i)Not Provided', np.nan, regex=True)
df.tail()
```

Out[3]:

	Id	EmployeeName	JobTitle	BasePay	OvertimePay	OtherPay	Benefits	TotalPay	TotalPayBenefits	Year	Notes	Agency	Status
148649	148650	Roy I Tillery	Custodian	0.00	0.00	0.00	0.00	0.00	0.00	2014	NaN	San Francisco	PT
148650	148651	NaN	Not provided	Not Provided	Not Provided	Not Provided	Not Provided	0.00	0.00	2014	NaN	San Francisco	NaN
148651	148652	NaN	Not provided	Not Provided	Not Provided	Not Provided	Not Provided	0.00	0.00	2014	NaN	San Francisco	NaN
148652	148653	NaN	Not provided	Not Provided	Not Provided	Not Provided	Not Provided	0.00	0.00	2014	NaN	San Francisco	NaN
148653	148654	Joe Lopez	Counselor, Log Cabin Ranch	0.00	0.00	-618.13	0.00	-618.13	-618.13	2014	NaN	San Francisco	PT

14. Drop The Rows Having 5 Missing Values 15. Find Job Title of ALBERT PARDINI 16. How Much ALBERT PARDINI Make (Include Benefits)? 17.Display Name of The Person Having The Highest BasePay.

```
In [2]: #14.Drop The Rows Having 5 Missing Values
df.drop(df[df.isnull().sum(axis=1)>=5].index,axis=0, inplace=True)
df
```

Out[2]:

	Id	EmployeeName	JobTitle	BasePay	OvertimePay	OtherPay	Benefits	TotalPay	TotalPayBenefits	Year	Notes	Agency	Status
0	1	NATHANIEL FORD	GENERAL MANAGER-METROPOLITAN TRANSIT AUTHORITY	167411.18	0.0	400184.25	NaN	567595.43	567595.43	2011	NaN	San Francisco	NaN
1	2	GARY JIMENEZ	CAPTAIN III (POLICE DEPARTMENT)	155966.02	245131.88	137811.38	NaN	538909.28	538909.28	2011	NaN	San Francisco	NaN
2	3	ALBERT PARDINI	CAPTAIN III (POLICE DEPARTMENT)	212739.13	106088.18	16452.6	NaN	335279.91	335279.91	2011	NaN	San Francisco	NaN
3	4	CHRISTOPHER CHONG	WIRE ROPE CABLE MAINTENANCE MECHANIC	77916.0	56120.71	198306.9	NaN	332343.61	332343.61	2011	NaN	San Francisco	NaN
4	5	PATRICK GARDNER	DEPUTY CHIEF OF DEPARTMENT, (FIRE DEPARTMENT)	134401.6	9737.0	182234.59	NaN	326373.19	326373.19	2011	NaN	San Francisco	NaN

```
In [8]: #15. Find Job Title of ALBERT PARDINI
df[df['EmployeeName'] == 'ALBERT PARDINI'][['EmployeeName','JobTitle']]
```

Out[8]:

	EmployeeName	JobTitle
2	ALBERT PARDINI	CAPTAIN III (POLICE DEPARTMENT)

```
In [9]: #16. How Much ALBERT PARDINI Make (Include Benefits)?
df[df['EmployeeName'] == 'ALBERT PARDINI'][['EmployeeName','TotalPayBenefits']]
```

Out[9]:

	EmployeeName	TotalPayBenefits
2	ALBERT PARDINI	335279.91

```
In [15]: #17.Display Name of The Person Having The Highest BasePay
emp=df[df['BasePay']== df.TotalPayBenefits.max()]
print(f'{emp["EmployeeName"].values[0]} is the highest paid employee with a base pay of {emp["BasePay"].values[0]}')
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
```

21. Find Top 5 Most Common Jobs

```
In [14]: #21. Find Top 5 Most Common Jobs
         df['JobTitle'].value_counts().head(5)

Out[14]: Transit Operator                7036
         Special Nurse                   4389
         Registered Nurse                3736
         Public Svc Aide-Public Works    2518
         Police Officer 3                2421
         Name: JobTitle, dtype: int64
```

4 .Perform following operation on Income Database .

1. Getting Information About Our Dataset Like Total Number Rows, Total Number of Columns,Datatypes of Each Column And Memory Requirement

```
In [8]: #3.Find Shape of Our Dataset (Number of Rows And Number of Columns)
        df.shape

Out[8]: (48842, 15)
```

```
In [9]: #4.Getting Information About Our Dataset Like Total Number Rows, Total Number of Columns,
        #Datatypes of Each Column And Memory Requirement
        df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48842 entries, 0 to 48841
Data columns (total 15 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   age              48842 non-null  int64
 1   workclass        48842 non-null  object
 2   fnlwgt           48842 non-null  int64
 3   education        48842 non-null  object
 4   educational-num  48842 non-null  int64
 5   marital-status   48842 non-null  object
 6   occupation       48842 non-null  object
 7   relationship     48842 non-null  object
 8   race             48842 non-null  object
 9   gender           48842 non-null  object
 10  capital-gain     48842 non-null  int64
 11  capital-loss     48842 non-null  int64
 12  hours-per-week   48842 non-null  int64
 13  native-country   48842 non-null  object
 14  income           48842 non-null  object
dtypes: int64(6), object(9)
memory usage: 5.6+ MB
```

2.Fetch Random Sample From the Dataset (50%) ,Check Null Values In The Dataset .

```
In [10]: #5.Fetch Random Sample From the Dataset (50%)
         df.sample(frac=0.5, random_state=10)
```

Out[10]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hours-per-week	native-country	income
28113	34	Private	339142	HS-grad	9	Separated	Handlers-cleaners	Unmarried	White	Female	0	0	40	United-States	<=50K
40486	69	Private	130413	Bachelors	13	Widowed	Exec-managerial	Not-in-family	White	Female	2346	0	15	United-States	<=50K
45549	26	Local-gov	72594	Some-college	10	Married-civ-spouse	Protective-serv	Husband	White	Male	0	0	55	United-States	>50K
3173	57	Local-gov	212303	Masters	14	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	40	United-States	>50K
46828	18	Private	137646	11th	7	Never-married	Other-service	Own-child	White	Female	0	0	15	United-States	<=50K
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
13121	26	Federal-gov	269353	Assoc-voc	11	Never-married	Adm-clerical	Not-in-family	Other	Male	0	0	55	United-States	<=50K
6802	46	Private	45857	HS-grad	9	Divorced	Sales	Not-in-family	White	Female	0	0	33	United-States	<=50K
23454	64	?	140237	Preschool	1	Married-civ-spouse	?	Husband	White	Male	0	0	40	United-States	<=50K
24541	25	Private	187540	Some-college	10	Never-married	Craft-repair	Not-in-family	White	Male	0	0	40	United-States	<=50K
5206	30	Private	247328	HS-grad	9	Separated	Protective-serv	Not-in-family	White	Male	0	0	40	United-States	>50K

24421 rows × 15 columns

```
In [12]: #8. Drop all The Missing Values
         df.isin(['?']).sum()

Out[12]: age                 0
         workclass        2799
         fnlwgt              0
         education           0
         educational-num     0
         marital-status      0
         occupation       2809
         relationship        0
         race                0
         gender              0
         capital-gain        0
         capital-loss        0
         hours-per-week      0
         native-country    857
         income              0
         dtype: int64

In [15]: df.replace('?', np.nan, inplace=True)

In [16]: df.isnull().sum()*100 / df.shape[0]

Out[16]: age              0.000000
         workclass        5.730724
```

```
In [18]: #9. Check for duplicated data and drop them.
         dup = df.duplicated().any()
         print('is there any duplicate value in dataset ->', dup)

In [11]: #6.Check Null Values In The Dataset
         df.isnull().sum()

Out[11]: age               0
         workclass         0
         fnlwgt            0
         education         0
         educational-num   0
         marital-status    0
         occupation        0
         relationship      0
         race              0
         gender            0
         capital-gain      0
         capital-loss      0
         hours-per-week    0
         native-country    0
         income            0
         dtype: int64

In [12]: #7.Perform Data Cleaning [ Replace '?' with Python ]
         df.isin(['?']).sum()

Out[12]: age                 0
         workclass        2799
         fnlwgt              0
         education           0
         educational-num     0
         marital-status      0
         occupation       2809
         relationship        0
         race                0
         gender              0
         capital-gain        0
         capital-loss        0
         hours-per-week      0
         native-country    857
```

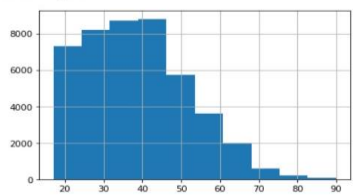
8. **Get Overall Statistics About The Dataframe**
9. **Drop The Columns education-num, capital-gain, and capital-loss**
10. **What Is The Distribution of Age Column?**
11. **1.Find Total Number of Persons Having Age Between 17 To 48**

```
In [23]: #12. What Is The Distribution of Age Column?
         df['age'].describe()

Out[23]: count    45175.000000
         mean        38.556170
         std         13.215349
         min         17.000000
         25%         28.000000
         50%         37.000000
         75%         47.000000
         max         90.000000
         Name: age, dtype: float64

In [24]: df['age'].hist()

Out[24]: <AxesSubplot:>
```




```
In [25]: #13. Find Total Number of Persons Having Age Between 17 To 48 (Inclusive) Using Between Method
         df['age'].between(17, 48).sum()

Out[25]: 34858
```

In [21]:
```
#10. Get Overall Statistics About The Dataframe
df.describe(include='all')
```

Out[21]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hou
count	45175.000000	45175	4.517500e+04	45175	45175.000000	45175	45175	45175	45175	45175	45175.000000	45175.000000	45175.(
unique	NaN	7	NaN	16	NaN	7	14	6	5	2	NaN	NaN	
top	NaN	Private	NaN	HS-grad	NaN	Married-civ-spouse	Craft-repair	Husband	White	Male	NaN	NaN	
freq	NaN	33262	NaN	14770	NaN	21042	6010	18653	38859	30495	NaN	NaN	
mean	38.556170	NaN	1.897388e+05	NaN	10.119314	NaN	NaN	NaN	NaN	NaN	1102.576270	88.687593	40.(
std	13.215349	NaN	1.056524e+05	NaN	2.551740	NaN	NaN	NaN	NaN	NaN	7510.249876	405.156611	12.(
min	17.000000	NaN	1.349200e+04	NaN	1.000000	NaN	NaN	NaN	NaN	NaN	0.000000	0.000000	1.(
25%	28.000000	NaN	1.173925e+05	NaN	9.000000	NaN	NaN	NaN	NaN	NaN	0.000000	0.000000	40.(
50%	37.000000	NaN	1.783120e+05	NaN	10.000000	NaN	NaN	NaN	NaN	NaN	0.000000	0.000000	40.(
75%	47.000000	NaN	2.379030e+05	NaN	13.000000	NaN	NaN	NaN	NaN	NaN	0.000000	0.000000	45.(
max	90.000000	NaN	1.490400e+06	NaN	16.000000	NaN	NaN	NaN	NaN	NaN	99999.000000	4356.000000	99.(

In [22]:
```
#11.Drop The Columns education-num, capital-gain, and capital-loss
df.drop(['educational-num', 'capital-gain', 'capital-loss'], axis=1, inplace=True)
df.head(3)
```

Out[22]:

	age	workclass	fnlwgt	education	marital-status	occupation	relationship	race	gender	hours-per-week	native-country	income
0	25	Private	226802	11th	Never-married	Machine-op-inspct	Own-child	Black	Male	40	United-States	<=50K
1	38	Private	89814	HS-grad	Married-civ-spouse	Farming-fishing	Husband	White	Male	50	United-States	<=50K
2	28	Local-gov	336951	Assoc-acdm	Married-civ-spouse	Protective-serv	Husband	White	Male	40	United-States	>50K

12. What is The Distribution of Workclass Column? 15. How Many Persons Having Bachelors and Masters Degree? 16. Bivariate Analsis

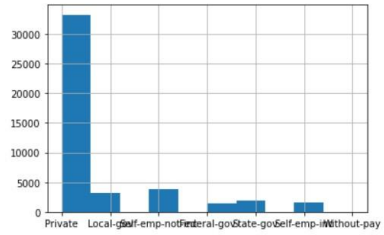
17. Replace Salary Values With 0 and 1

In [26]:
```
#14.What is The Distribution of Workclass Column?
df['workclass'].describe()
```

Out[26]:
```
count       45175
unique          7
top       Private
freq        33262
Name: workclass, dtype: object
```

In [27]:
```
df['workclass'].hist()
```

Out[27]: <AxesSubplot:>

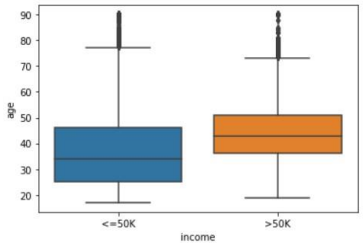


In [28]:
```
#15.How Many Persons Having Bachelors and Masters Degree?
df[(df['education']=='Bachelors') | (df['education']=='Masters')].shape[0]
```

Out[28]: 10072

```
In [30]: #16.Bivariate Analsis
         sns.boxplot(x='income', y='age', data=df)
```

```
Out[30]: <AxesSubplot:xlabel='income', ylabel='age'>
```




```
In [31]: #17.Replace salary Values With 0 and 1
         df['income'].value_counts()
```

```
Out[31]: <=50K    37155
         >50K     11687
         Name: income, dtype: int64
```

18. Which Workclass Getting The Highest Salary?

19.How Has Better Chance To Get Salary greater than 50K Male or Female.

```
In [33]: #18.Which Workclass Getting The Highest Salary?
         highest_sal = df.groupby('workclass')['encoded_income'].mean().sort_values(ascending=False)

         print( highest_sal.values[0])
```

```
0.5533923303834808
```

```
In [34]: #19.Who Has Better Chance To Get Salary greater than 50K Male or Female?
         total_50k = df[df['encoded_income']==1]['gender'].value_counts().sort_values(ascending= False)

         percentage_male = total_50k[0]*100/(total_50k[0]+total_50k[1])
         percentage_female = total_50k[1]*100/(total_50k[0]+total_50k[1])

         print('Male chance -> ', percentage_male, 'Female chance -> ', percentage_female)
```

```
Male chance ->  84.863523573201 Female chance ->  15.136476426799007
```

20.Convert Workclass columns datatype into category datatype.

```
In [35]: #20.Covert workclass Columns Datatype To Category Datatype
         df['workclass'] = df['workclass'].astype('category')
         df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48842 entries, 0 to 48841
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   age              48842 non-null  int64
 1   workclass        48842 non-null  category
 2   fnlwgt           48842 non-null  int64
 3   education        48842 non-null  object
 4   educational-num  48842 non-null  int64
 5   marital-status   48842 non-null  object
 6   occupation       48842 non-null  object
 7   relationship     48842 non-null  object
 8   race             48842 non-null  object
 9   gender           48842 non-null  object
 10  capital-gain     48842 non-null  int64
 11  capital-loss     48842 non-null  int64
 12  hours-per-week   48842 non-null  int64
 13  native-country   48842 non-null  object
 14  income           48842 non-null  object
 15  encoded_income   48842 non-null  int64
dtypes: category(1), int64(7), object(8)
memory usage: 5.6+ MB
```