PANDAS

NUMPY

MATPLOTLIB & SEABORN

In [ ]:

```
1
```

In [ ]:

```
1
```

In [21]:

```
1 import pandas as pd
2 data = pd.read_csv("Social_Network_Ads.csv", sep = ',')
```

In [22]:

```
1 data.head()
```

Out[22]:

	Age	EstimatedSalary	Purchased
0	19.0	19000.0	0.0
1	35.0	20000.0	0.0
2	26.0	43000.0	0.0
3	27.0	57000.0	0.0
4	19.0	76000.0	0.0

In [23]:

```
1 # data.drop("Purchased", axis =1, inplace = True)
```

In [24]:

```
1 subset = data[['Purchased', 'EstimatedSalary']]
```

In [27]:

```
1 subset.head()
```

Out[27]:

	Purchased	EstimatedSalary
0	0.0	19000.0
1	0.0	20000.0
2	0.0	43000.0
3	0.0	57000.0
4	0.0	76000.0

In [25]:

```
1 data.describe()
```

Out[25]:

	Age	EstimatedSalary	Purchased
count	398.000000	397.000000	396.000000
mean	37.680905	69609.571788	0.356061
std	10.492708	33887.085366	0.479439
min	18.000000	15000.000000	0.000000
25%	30.000000	43000.000000	0.000000
50%	37.000000	70000.000000	0.000000
75%	46.000000	87000.000000	1.000000
max	60.000000	150000.000000	1.000000

In [18]:

```
1 data[['Age','EstimatedSalary']]
```

Out[18]:

	Age	EstimatedSalary
0	19.0	19000.0
1	35.0	20000.0
2	26.0	43000.0
3	27.0	57000.0
4	19.0	76000.0
...	...	...
395	46.0	41000.0
396	51.0	23000.0
397	50.0	20000.0
398	36.0	33000.0
399	49.0	36000.0

400 rows × 2 columns

In [19]:

```
1 data.iloc[: , 1:]
```

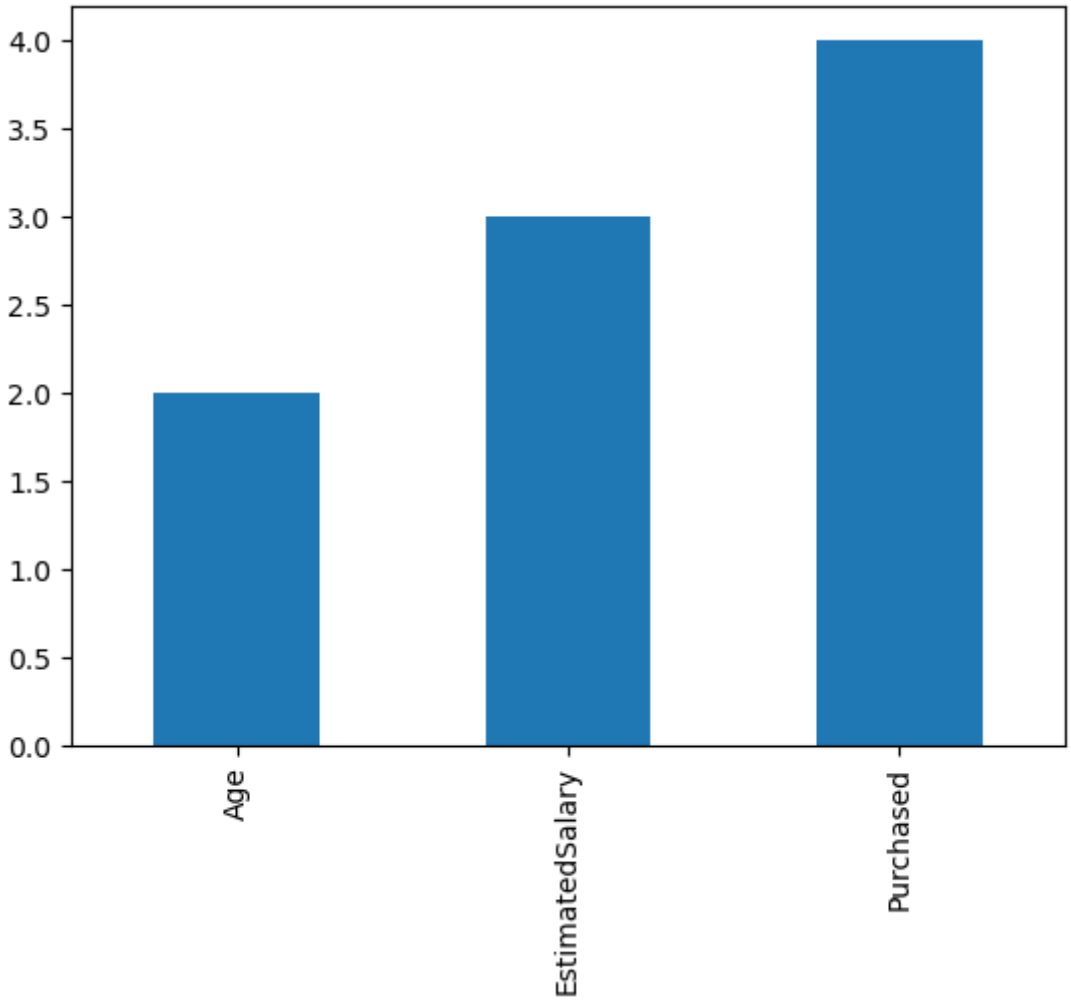
Out[19]:

	EstimatedSalary
0	19000.0
1	20000.0
2	43000.0
3	57000.0
4	76000.0
...	...
395	41000.0
396	23000.0
397	20000.0
398	33000.0
399	36000.0

400 rows × 1 columns

In [28]:

```
1 data.isnull().sum().plot(kind = 'bar');
```



In [41]:

```
1 data['Age']<30
```

Out[41]:

```
0       True
1      False
2       True
3       True
4       True
       ...
395    False
396    False
397    False
398    False
399    False
Name: Age, Length: 400, dtype: bool
```

In [51]:

```
above_30 = data['Age']>30
below_30, above_30 = data[~above_30], data[above_30]
above_30
```

Out[51]:

	Age	EstimatedSalary	Purchased
1	35.0	20000.0	0.0
7	32.0	NaN	1.0
9	35.0	65000.0	0.0
13	32.0	18000.0	0.0
16	47.0	25000.0	1.0
...	...	...	...
395	46.0	41000.0	1.0
396	51.0	23000.0	1.0
397	50.0	20000.0	1.0
398	36.0	33000.0	0.0
399	49.0	36000.0	1.0

288 rows × 3 columns

In [53]:

```
below_30
```

Out[53]:

	Age	EstimatedSalary	Purchased
0	19.0	19000.0	0.0
2	26.0	43000.0	0.0
3	27.0	57000.0	0.0
4	19.0	76000.0	0.0
5	27.0	58000.0	0.0
...	...	...	...
193	19.0	70000.0	0.0
194	28.0	89000.0	0.0
196	30.0	79000.0	0.0
197	20.0	36000.0	0.0
198	26.0	80000.0	0.0

112 rows × 3 columns

In [66]:

```
not_purchased = data['Purchased']==0
purchased, not_purchased = data[~not_purchased], data[not_purchased]
purchased.describe()
```

Out[66]:

	Age	EstimatedSalary	Purchased
count	145.000000	143.000000	141.0
mean	46.234483	85888.111888	1.0
std	8.655453	41537.714900	0.0
min	27.000000	20000.000000	1.0
25%	39.000000	41500.000000	1.0
50%	47.000000	90000.000000	1.0
75%	53.000000	122000.000000	1.0
max	60.000000	150000.000000	1.0

In [70]:

```
not_purchased.describe()
```

Out[70]:

	Age	EstimatedSalary	Purchased
count	253.000000	254.000000	255.0
mean	32.778656	60444.881890	0.0
std	8.025654	24390.493219	0.0
min	18.000000	15000.000000	0.0
25%	26.000000	44000.000000	0.0
50%	34.000000	61000.000000	0.0
75%	38.000000	77000.000000	0.0
max	59.000000	141000.000000	0.0

In [64]:

```
data['Purchased'].unique()
```

Out[64]:

```
array([ 0.,  1., nan])
```

In [72]:

```
not_purchased[not_purchased['EstimatedSalary'] == max(not_purchased['Estimate
```

Out[72]:

	Age	EstimatedSalary	Purchased
284	48.0	141000.0	0.0

In [81]:

```
purchased[purchased['EstimatedSalary'] == min(purchased['EstimatedSalary'].dr
```

Out[81]:

	Age	EstimatedSalary	Purchased
25	47.0	20000.0	1.0
397	50.0	20000.0	1.0

In [54]:

```
less_30[less_30['EstimatedSalary'] == max(less_30['EstimatedSalary'])]
```

Out[54]:

	Age	EstimatedSalary	Purchased
168	29.0	148000.0	1.0

In [96]:

```
data.Age.fillna(data.Age.mean(), inplace = True)
```

In [97]:

```
data.isnull().sum()
```

Out[97]:

```
Age                0
EstimatedSalary    3
Purchased          4
dtype: int64
```

In [ ]:

```

```

In [ ]:

```
data[['Age']].describe()
```

In [ ]:

```
data.head()
```

In [ ]:

```
data.isnull().sum()
```

In [ ]:

```
data.isnull().sum().plot(kind = 'bar');
```

In [ ]:

```
# mising_salary = data['EstimatedSalary'].mean()

# data['EstimatedSalary'].fillna(mising_salary, inplace = True)
```

In [ ]:

```
data['name'] = 'caleb'
```

In [ ]:

```
data.info()
```

In [ ]:

```
data.shape
```

In [ ]:

```
# df = pd.read_csv('')
```

In [ ]:

```
data[data['EstimatedSalary'].isna()]
```

In [ ]:

```
# x = data[data['EstimatedSalary'].isna()]
# val =[23,45,98]
# for i in range(3):
#     data[data['EstimatedSalary'].isna()].fillna(val[i], limit = 1)
```

In [ ]:

```
# x = data[data['EstimatedSalary'].isna()]
x
```

In [ ]:

```
# val = [230,450,980]
# indexx = [7,20,34]
# for i in range(len(val)):
# #     x.loc[indexx[i], 'EstimatedSalary'] = val[i]
```

In [ ]:

```
val = [112,95,11]
indexx = [7,20,34]
for i in range(len(val)):
    data.at[indexx[i], 'EstimatedSalary'] = val[i]
```

In [ ]:

```
for i, u in enumerate(val):
    print(f"i_values: {i}\t u_values: ,{u}")
```

In [ ]:

```
for i in range(len(val)):
    print(f"i_values: {i}\t u_values: ,{val[i]}")
```

In [ ]:

```
data[data['EstimatedSalary'].isna()]
```

In [ ]:

```
data.head(200)
```

In [ ]:

```
data[data['Age'].isna()]
```

In [ ]:

```
data[data['Purchased'].isna()]
```

In [ ]:

```
data[data.isnull().any(axis =1)]
```

In [ ]:

```
data.isnull().sum()
```

In [ ]:

```
data.query('(Age < 30) and (Purchased > 0)')
```

In [ ]:

```
data.query('(Purchased == 1)').Age.plot(kind= 'hist');
```

In [ ]:

```

```

In [ ]:

```
data['Age'].fillna(data['Age'].mean(), inplace = True)
```

In [ ]:

```
# np.where(data['EstimatedSalary'] == "")
```

In [ ]:

```
data.isnull().sum()
```

In [ ]:

```
# data.query("EstimatedSalary == '2'")
```

In [ ]:

```
data.info()
```

In [ ]:

```
data.head()
```

In [ ]:

```
data.info()
```

iloc() and loc funtion

condictional filtering

apply() funtion

groupby() operations

sorting() operations

bool series

pandas merg()

In [ ]:

```
data.info()
```

In [ ]:

```
data.loc[50]
```

In [ ]:

```
data.iloc[:, 0:-2]
```

In [ ]:

```
x = data.iloc[:50, :-1]
y = data.iloc[:50, -1]
```

In [ ]:

```
data['rate'] = [i*2 for i in range(400)] #list comprehension
```

In [ ]:

```
# data['rate'] = rate
```

apply() func

In [ ]:

```
data
```

In [ ]:

```
# data.drop(('Rate'), axis =1, inplace = True)
```

In [ ]:

```
data
```

In [ ]:

```
list(data.columns)
```

In [ ]:

```
cols = list(data.columns)
```

In [ ]:

```
cols[4], cols[1] = 'Rate', 'Estimated_Salary'
```

In [ ]:

```
x,y =1,2
print(x,y)
```

In [ ]:

```
data.columns = cols
```

In [ ]:

```
data
```

In [ ]:

```
data.columns = data.columns.str.title()# converting the columns to a title ca
```

In [ ]:

```
data
```

creating a bool series for data filtering

In [ ]:

```
# creating a numpy nan values to be added to a dataframe
import numpy as np
data_dict = {'age': [1,2,3,4,np.nan,5,np.nan],
             'width': [np.nan,2,3,4,5,6,np.nan]}
```

In [ ]:

```
df = pd.DataFrame.from_dict(data_dict)# creating a dtaframe from dictionary o
```

In [ ]:

```
df
```

In [ ]:

```
data
```

In [ ]:

```
data_series = pd.notnull(data['Age'])
data_series
```

In [ ]:

```
data[~data_series]
```

In [ ]:

```
data_series = pd.isnull(data['Purchased'])
```

In [ ]:

```
data[~data_series]
```

In [ ]:

```
data[~data_series]
```

In [ ]:

```
data_series = pd.isnull(data[data.columns])
```

In [ ]:

```
data[~data_series]
```

In [ ]:

```
data.head()
```

In [ ]:

```
data.info()
```

In [ ]:

```
def squre(X):

    return x*x

df1 = data['Age'].apply(squre)
```

In [ ]:

```
df1
```

Condictional Filtering

In [1]:

```
import pandas as pd
df = pd.read_csv('Churn_Modelling.csv')
df
```

Out[1]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	
0	1	15634602	Hargrave	619	France	Female	42	2	
1	2	15647311	Hill	608	Spain	Female	41	1	8
2	3	15619304	Onio	502	France	Female	42	8	15
3	4	15701354	Boni	699	France	Female	39	1	
4	5	15737888	Mitchell	850	Spain	Female	43	2	12
...	...	...	...	...	...	...	...	...	
9995	9996	15606229	Obijiaku	771	France	Male	39	5	
9996	9997	15569892	Johnstone	516	France	Male	35	10	5
9997	9998	15584532	Liu	709	France	Female	36	7	
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	7
9999	10000	15628319	Walker	792	France	Female	28	4	13

10000 rows × 14 columns

In [2]:

```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   RowNumber        10000 non-null  int64  
 1   CustomerId       10000 non-null  int64  
 2   Surname          10000 non-null  object 
 3   CreditScore      10000 non-null  int64  
 4   Geography        10000 non-null  object 
 5   Gender           10000 non-null  object 
 6   Age              10000 non-null  int64  
 7   Tenure           10000 non-null  int64  
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64  
 10  HasCrCard        10000 non-null  int64  
 11  IsActiveMember   10000 non-null  int64  
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64  
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

In [3]:

```
1 df.Exited.unique() # used to check the unique values
```

Out[3]:

```
array([1, 0])
```

In [6]:

```
1 df.Age.nunique() # number of unique values
```

Out[6]:

```
70
```

single conditions

In [10]:

```
1 exited = df[df['Exited'] == 1]
```

Out[10]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure
0	1	15634602	Hargrave	619	France	Female	42	2
2	3	15619304	Onio	502	France	Female	42	8
5	6	15574012	Chu	645	Spain	Male	44	8
7	8	15656148	Obinna	376	Germany	Female	29	4
16	17	15737452	Romeo	653	Germany	Male	58	1
...	...	...	...	...	...	...	...	...
9981	9982	15672754	Burbidge	498	Germany	Male	42	3
9982	9983	15768163	Griffin	655	Germany	Female	46	7
9991	9992	15769959	Ajuluchukwu	597	France	Female	53	4
9997	9998	15584532	Liu	709	France	Female	36	7
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3

2037 rows × 14 columns

In [14]:

```
1 x = df['Exited'] == 1
2 members = df[~x]
```

In [16]:

```
1 members.describe()
```

Out[16]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	N
count	7963.000000	7.963000e+03	7963.000000	7963.000000	7963.000000	7963.000000	
mean	5024.694964	1.569117e+07	651.853196	37.408389	5.033279	72745.296779	
std	2891.682053	7.174423e+04	95.653837	10.125363	2.880658	62848.040701	
min	2.000000	1.556570e+07	405.000000	18.000000	0.000000	0.000000	
25%	2526.500000	1.562882e+07	585.000000	31.000000	3.000000	0.000000	
50%	5042.000000	1.569154e+07	653.000000	36.000000	5.000000	92072.680000	
75%	7525.500000	1.575335e+07	718.000000	41.000000	7.000000	126410.280000	
max	10000.000000	1.581569e+07	850.000000	92.000000	10.000000	221532.800000	

In [17]:

```
one_condition = df[df['Exited'] == 1]



print(one_condition.shape[0])
one_condition.head()
```

```
2037
```

Out[17]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Bal
0	1	15634602	Hargrave	619	France	Female	42	2	
2	3	15619304	Onio	502	France	Female	42	8	1596(
5	6	15574012	Chu	645	Spain	Male	44	8	1137!
7	8	15656148	Obinna	376	Germany	Female	29	4	1150-
16	17	15737452	Romeo	653	Germany	Male	58	1	1326(

multiple columns

In [20]:

```
df.EstimatedSalary.describe()
```

Out[20]:

```
count     10000.000000
mean     100090.239881
std       57510.492818
min          11.580000
25%       51002.110000
50%      100193.915000
75%      149388.247500
max      199992.480000
Name: EstimatedSalary, dtype: float64
```

In [23]:

```
mult_conditions = df[(df['Age'] <= 30) & (df['Geography'] == 'France') &
                     (df['Gender']== 'Female') &
                     (df['EstimatedSalary'] < df['EstimatedSalary'].mean()) &
                     (df['Exited'] == 0)]
mult_conditions
```

Out[23]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure
19	20	15568982	Hao	726	France	Female	24	6
69	70	15755648	Pisano	675	France	Female	21	8
75	76	15780961	Cavenagh	735	France	Female	21	1
226	227	15774393	Ch'ien	694	France	Female	30	9
258	259	15750803	Jess	693	France	Female	30	6
...	...	...	...	...	...	...	...	...
9788	9789	15571756	Ohearn	724	France	Female	28	5
9823	9824	15622658	Lai	551	France	Female	26	2
9860	9861	15716431	Brookes	775	France	Female	30	10
9940	9941	15791972	Bergamaschi	748	France	Female	20	7
9999	10000	15628319	Walker	792	France	Female	28	4

205 rows × 14 columns

In [26]:

```
df.head()
```

Out[26]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Bala
0	1	15634602	Hargrave	619	France	Female	42	2	(
1	2	15647311	Hill	608	Spain	Female	41	1	8380
2	3	15619304	Onio	502	France	Female	42	8	15966(
3	4	15701354	Boni	699	France	Female	39	1	(
4	5	15737888	Mitchell	850	Spain	Female	43	2	12551(

In [27]:

```
1 df.iloc[ : , 1: ]
```

Out[27]:

	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	Num
0	15634602	Hargrave	619	France	Female	42	2	0.00	
1	15647311	Hill	608	Spain	Female	41	1	83807.86	
2	15619304	Onio	502	France	Female	42	8	159660.80	
3	15701354	Boni	699	France	Female	39	1	0.00	
4	15737888	Mitchell	850	Spain	Female	43	2	125510.82	
...	...	...	...	...	...	...	...	...	
9995	15606229	Obijiaku	771	France	Male	39	5	0.00	
9996	15569892	Johnstone	516	France	Male	35	10	57369.61	
9997	15584532	Liu	709	France	Female	36	7	0.00	
9998	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	
9999	15628319	Walker	792	France	Female	28	4	130142.79	

10000 rows × 13 columns

In [24]:

```
1 mult_conditions.shape
```

Out[24]:

```
(205, 14)
```

In [30]:

```
germany = df[df['Geography'] == 'Germany']
germany
```

Out[30]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenu
7	8	15656148	Obinna	376	Germany	Female	29	
15	16	15643966	Goforth	616	Germany	Male	45	
16	17	15737452	Romeo	653	Germany	Male	58	
26	27	15736816	Young	756	Germany	Male	36	
28	29	15728693	McWilliams	574	Germany	Female	43	
...	...	...	...	...	...	...	...	
9982	9983	15768163	Griffin	655	Germany	Female	46	
9984	9985	15696175	Echezonachukwu	602	Germany	Male	35	
9986	9987	15581736	Bartlett	673	Germany	Male	47	
9990	9991	15798964	Nkemakonam	714	Germany	Male	33	
9998	9999	15682355	Sabbatini	772	Germany	Male	42	

2509 rows × 14 columns

In [44]:

```
df[~df.isna()]
```

Out[44]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	
0	1	15634602	Hargrave	619	France	Female	42	2	
1	2	15647311	Hill	608	Spain	Female	41	1	8
2	3	15619304	Onio	502	France	Female	42	8	15
3	4	15701354	Boni	699	France	Female	39	1	
4	5	15737888	Mitchell	850	Spain	Female	43	2	12
...	...	...	...	...	...	...	...	...	
9995	9996	15606229	Obijiaku	771	France	Male	39	5	
9996	9997	15569892	Johnstone	516	France	Male	35	10	5
9997	9998	15584532	Liu	709	France	Female	36	7	
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	7
9999	10000	15628319	Walker	792	France	Female	28	4	13

10000 rows × 14 columns

In [41]:

```
df.isnull().sum()
```

Out[41]:

```
RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

In [38]:

```
CreditScore = [619, 699, 792]

mult_condition_filters_methods = df[
    (df["CreditScore"].isin(CreditScore)) &

    (df["Geography"].str.contains("France")) &
    (df["Age"].between(35, 42))
]
```

In [36]:

```
1 mult_condition_filters_methods
```

Out[36]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	E
0	1	15634602	Hargrave	619	France	Female	42	2	
3	4	15701354	Boni	699	France	Female	39	1	
608	609	15607170	Boyle	699	France	Male	35	5	
935	936	15675316	Avdeeva	619	France	Female	38	3	
1609	1610	15750248	Wright	619	France	Female	35	8	13
1638	1639	15571550	Dore	699	France	Male	39	9	
2295	2296	15664543	Shaw	699	France	Male	40	7	
3114	3115	15636023	O'Donnell	619	France	Female	40	10	
3418	3419	15632272	Lung	792	France	Female	42	2	
5251	5252	15743759	Brooks	619	France	Male	39	5	
5797	5798	15775206	Hunter	699	France	Male	37	10	
6332	6333	15793046	Holden	619	France	Female	35	4	9(
7885	7886	15632344	Jones	792	France	Female	42	0	9
8895	8896	15658972	Foster	699	France	Female	40	8	12
9228	9229	15749679	Beck	699	France	Male	39	2	10

In [34]:

```
1 df[df.CreditScore == 792]
```

Out[34]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	
785	786	15660155	Lorenzo	792	Spain	Male	36	5	
1427	1428	15799966	Chigolum	792	Germany	Female	59	9	1
1701	1702	15605279	Francis	792	France	Male	50	9	
2956	2957	15642885	Gray	792	France	Male	30	8	
3095	3096	15680243	Brown	792	France	Male	19	7	1
3418	3419	15632272	Lung	792	France	Female	42	2	
3617	3618	15663446	Volkova	792	Germany	Female	29	4	1
4211	4212	15615207	Yeh	792	Spain	Male	47	0	
4558	4559	15623730	Ch'iu	792	France	Male	34	1	
4924	4925	15574868	Lowell	792	Germany	Male	36	5	1
5075	5076	15684921	Onuchukwu	792	Spain	Male	25	8	1
6606	6607	15633181	Swinton	792	France	Male	31	6	
7772	7773	15614168	Alexander	792	Germany	Female	50	4	1
7885	7886	15632344	Jones	792	France	Female	42	0	
8139	8140	15770539	Walters	792	France	Male	30	1	1
8444	8445	15793641	Evseyev	792	France	Female	70	3	
8802	8803	15714642	Hawkins	792	Spain	Female	40	7	
9999	10000	15628319	Walker	792	France	Female	28	4	1

using pandas query

In [45]:

```
one_year_query = df.query('CreditScore > 500')
one_year_query
```

Out[45]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	
0	1	15634602	Hargrave	619	France	Female	42	2	
1	2	15647311	Hill	608	Spain	Female	41	1	8
2	3	15619304	Onio	502	France	Female	42	8	15
3	4	15701354	Boni	699	France	Female	39	1	
4	5	15737888	Mitchell	850	Spain	Female	43	2	12
...	...	...	...	...	...	...	...	...	
9995	9996	15606229	Obijiaku	771	France	Male	39	5	
9996	9997	15569892	Johnstone	516	France	Male	35	10	5
9997	9998	15584532	Liu	709	France	Female	36	7	
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	7
9999	10000	15628319	Walker	792	France	Female	28	4	13

9357 rows × 14 columns

In [46]:

```
one_year_query.Tenure.unique()
```

Out[46]:

```
array([ 2,  1,  8,  7,  4,  6,  5,  3,  9, 10,  0])
```

In [51]:

```
df.Tenure.unique()
```

Out[51]:

```
array([ 2,  1,  8,  7,  4,  6,  3, 10,  5,  9,  0])
```

In [52]:

```
df
```

Out[52]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	
0	1	15634602	Hargrave	619	France	Female	42	2	
1	2	15647311	Hill	608	Spain	Female	41	1	8
2	3	15619304	Onio	502	France	Female	42	8	15
3	4	15701354	Boni	699	France	Female	39	1	
4	5	15737888	Mitchell	850	Spain	Female	43	2	12
...	...	...	...	...	...	...	...	...	
9995	9996	15606229	Obijiaku	771	France	Male	39	5	
9996	9997	15569892	Johnstone	516	France	Male	35	10	5
9997	9998	15584532	Liu	709	France	Female	36	7	
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	7
9999	10000	15628319	Walker	792	France	Female	28	4	13

10000 rows × 14 columns

In [59]:

```
bal = [2, 8, 1, 0]
# name = ['Hargrave', 'Onio', 'Mitchell']

mult_conditions_query = df.query(
    'Tenure.isin(@bal) and Surname.str.contains("Mitchell") and EstimatedSalary
)

mult_conditions_query
```

Out[59]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	E
4	5	15737888	Mitchell	850	Spain	Female	43	2	125
9922	9923	15596811	Mitchell	667	France	Male	36	8	139

In [ ]:

```
mult_conditions_query = df.query(
    "Tenure.isin(@bal) and Age.between(10,50) and EstimatedSalary.between(7908
)

mult_conditions_query.head()
```