*Credit card * :- Credit cards are financial tools that allow individuals to make purchases on credit. Credit card come with a predetermined credit limit that represents the maximum amount a cardholder can borrow.

*Credit Card offer certain benefits :- *

1.**Convenience**: Credit cards provide a convenient payment method for both online and in-person transactions.

2.**Reward and facility** : access to airport lounge , discount at dining centre and cashback offers.

3.**Security**: Credit cards offer enhanced security features transactions. Most credit cards have built-in fraud protection.

Before exploratory data analysis and feature engineering we have to consider certain hypothesis.

1. An applicant must have a minimum age of 18 years.
2. An applicant must have minimum income level.
3. Have a stable source of money.
4. An applicant shall have citizenship card and other identity proof.

Exploratory data analysis of dataset

```
# importing libraries for data manipulation and data visualisation

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns


# reading credit_card data

credit_card = pd.read_csv("/content/Credit_card.csv")


#to display top 5 rows

credit_card.head()
```

ype_Income	EDUCATION	Marital_status	Housing_type	Birthday_count	Employed_days
Pensioner	Higher education	Married	House / apartment	-18772.0	365243
Commercial associate	Higher education	Married	House / apartment	-13557.0	-586
Commercial associate	Higher education	Married	House / apartment	NaN	-586
Commercial associate	Higher education	Married	House / apartment	-13557.0	-586
Commercial associate	Higher education	Married	House / apartment	-13557.0	-586

```
# to display bottom 5 rows

credit_card.tail()
```

	Ind_ID	GENDER	Car_Owner	Propert_Owner	CHILDREN	Annual_income	Type_Inc
1543	5028645	F	N	Y	0	NaN	Commer assoc
1544	5023655	F	N	N	0	225000.0	Commer assoc
1545	5115992	M	Y	Y	2	180000.0	Worl
1546	5118219	M	Y	N	0	270000.0	Worl
1547	5053790	F	Y	Y	0	225000.0	Worl

```
# To check the datatype

credit_card.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1548 entries, 0 to 1547
Data columns (total 18 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Ind_ID           1548 non-null   int64  
 1   GENDER           1541 non-null   object 
 2   Car_Owner        1548 non-null   object 
 3   Propert_Owner    1548 non-null   object 
 4   CHILDREN         1548 non-null   int64  
 5   Annual_income    1525 non-null   float64
 6   Type_Income      1548 non-null   object 
 7   EDUCATION        1548 non-null   object 
 8   Marital_status   1548 non-null   object 
 9   Housing_type     1548 non-null   object 
 10  Birthday_count   1526 non-null   float64
 11  Employed_days    1548 non-null   int64  
 12  Mobile_phone     1548 non-null   int64  
 13  Work_Phone       1548 non-null   int64  
 14  Phone            1548 non-null   int64  
 15  EMAIL_ID         1548 non-null   int64  
 16  Type_Occupation  1060 non-null   object 
 17  Family_Members   1548 non-null   int64  
dtypes: float64(2), int64(8), object(8)
memory usage: 217.8+ KB
```

```
# To display categorical and numeical

credit_card.describe(include = "all")
```

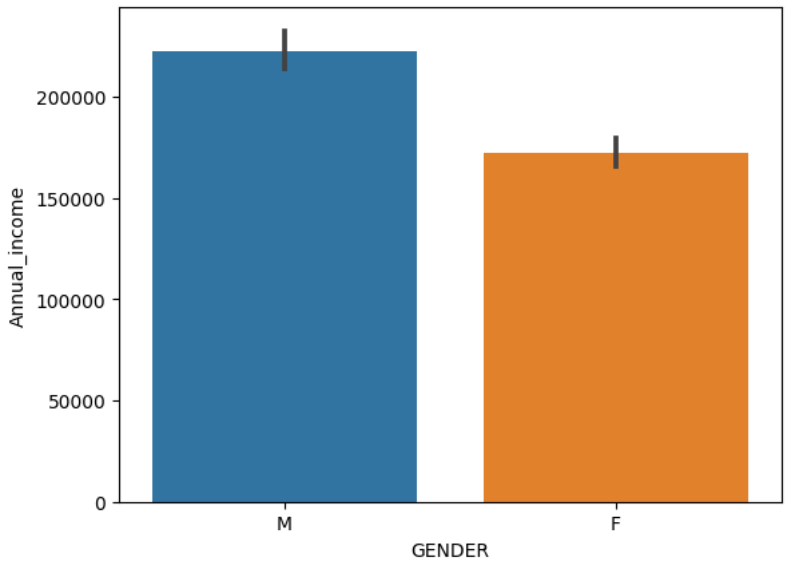
	Ind_ID	GENDER	Car_Owner	Propert_Owner	CHILDREN	Annual_income
count	1.548000e+03	1541	1548	1548	1548.000000	1.525000e+03
unique	NaN	2	2	2	NaN	NaN
top	NaN	F	N	Y	NaN	NaN
freq	NaN	973	924	1010	NaN	NaN
mean	5.078920e+06	NaN	NaN	NaN	0.412791	1.913993e+05
std	4.171759e+04	NaN	NaN	NaN	0.776691	1.132530e+05
min	5.008827e+06	NaN	NaN	NaN	0.000000	3.375000e+04
25%	5.045070e+06	NaN	NaN	NaN	0.000000	1.215000e+05
50%	5.078842e+06	NaN	NaN	NaN	0.000000	1.665000e+05
75%	5.115673e+06	NaN	NaN	NaN	1.000000	2.250000e+05
max	5.150412e+06	NaN	NaN	NaN	14.000000	1.575000e+06

To understand the data we need to plot the data in relation to each other

```
# To plot gender and annual income
```

```
sns.barplot(data=credit_card,x="GENDER",y="Annual_income")
```

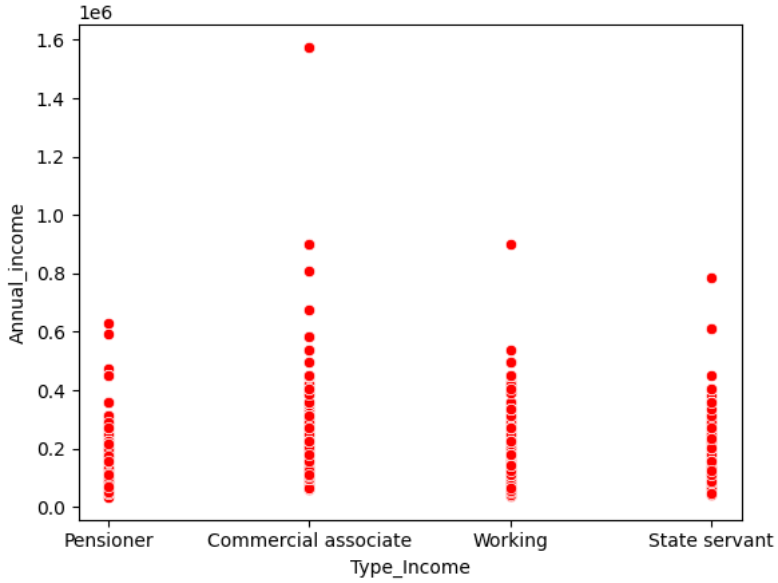
```
<Axes: xlabel='GENDER', ylabel='Annual_income'>
```



The annual income of male is more compared to female.

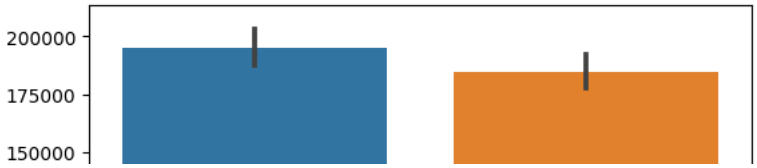
```
# To plot annual_income and Type_income

sns.scatterplot(data=credit_card,x="Type_Income",y="Annual_income",marker = "o",color = "r")
```

```
<Axes: xlabel='Type_Income', ylabel='Annual_income'>
```




```
# To plot Property_Owner and Annual_income

sns.barplot(data = credit_card,x = "Propert_Owner",y="Annual_income")
```

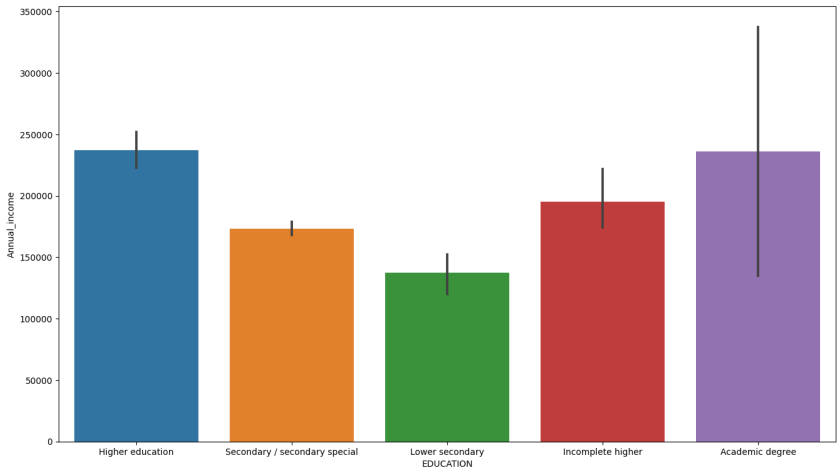
```
<Axes: xlabel='Propert_Owner', ylabel='Annual_income'>
```



The number of people owning property is more than the people not owning property


```
# To plot educational status and annual income
plt.figure(figsize=([16,9]))
sns.barplot(data=credit_card,x="EDUCATION",y="Annual_income")
```

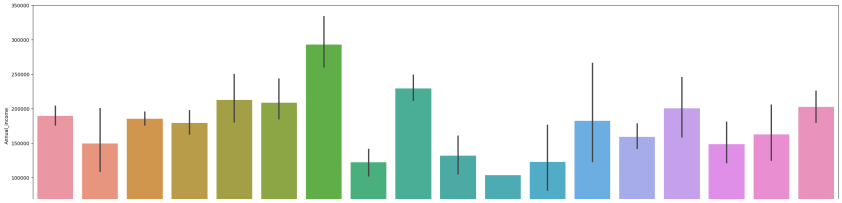
```
<Axes: xlabel='EDUCATION', ylabel='Annual_income'>
```



The annual income for higher education people is highest and lowest for lower secondary education. These shows eduaction and income are directly proportional to each others.

```
# To plot type_of_occupation and annual income
plt.figure(figsize=([30,9]))
sns.barplot(data=credit_card,x="Type_Occupation",y="Annual_income")
```

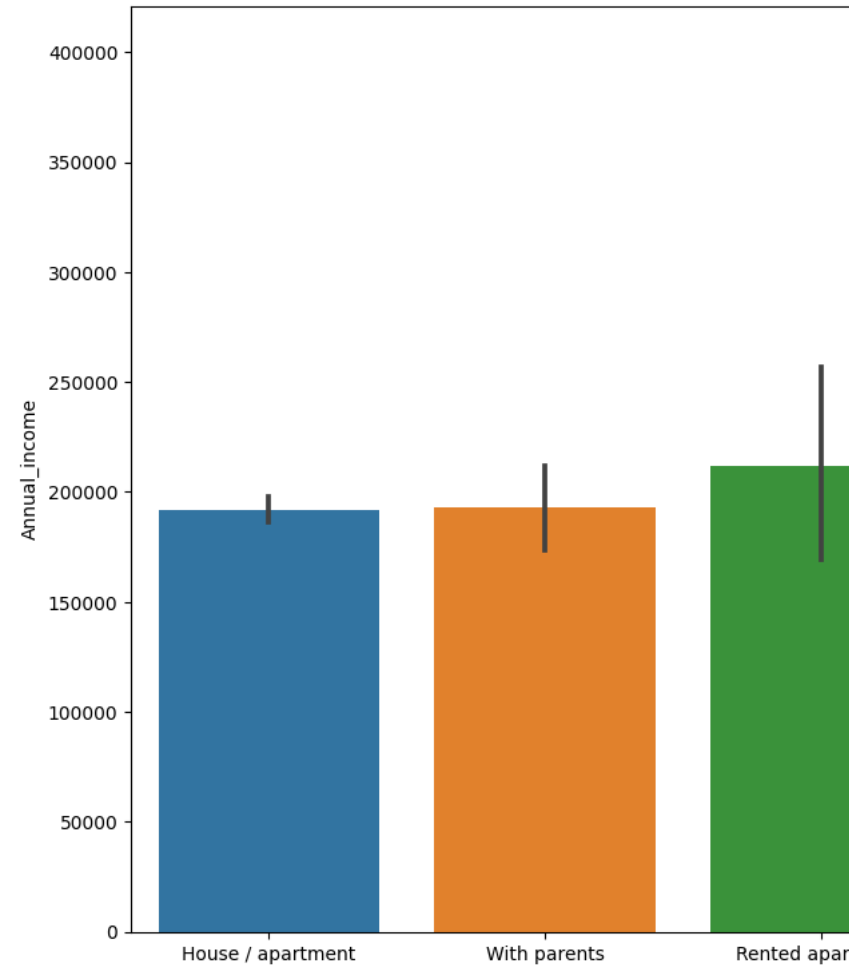
```
<Axes: xlabel='Type_Occupation', ylabel='Annual_income'>
```



The annual income is highest for Managers and lowest for IT staffs

```
# To plot annual income and Housing_type
plt.figure(figsize=([16,9]))
sns.barplot(data=credit_card,x="Housing_type",y="Annual_income")
```

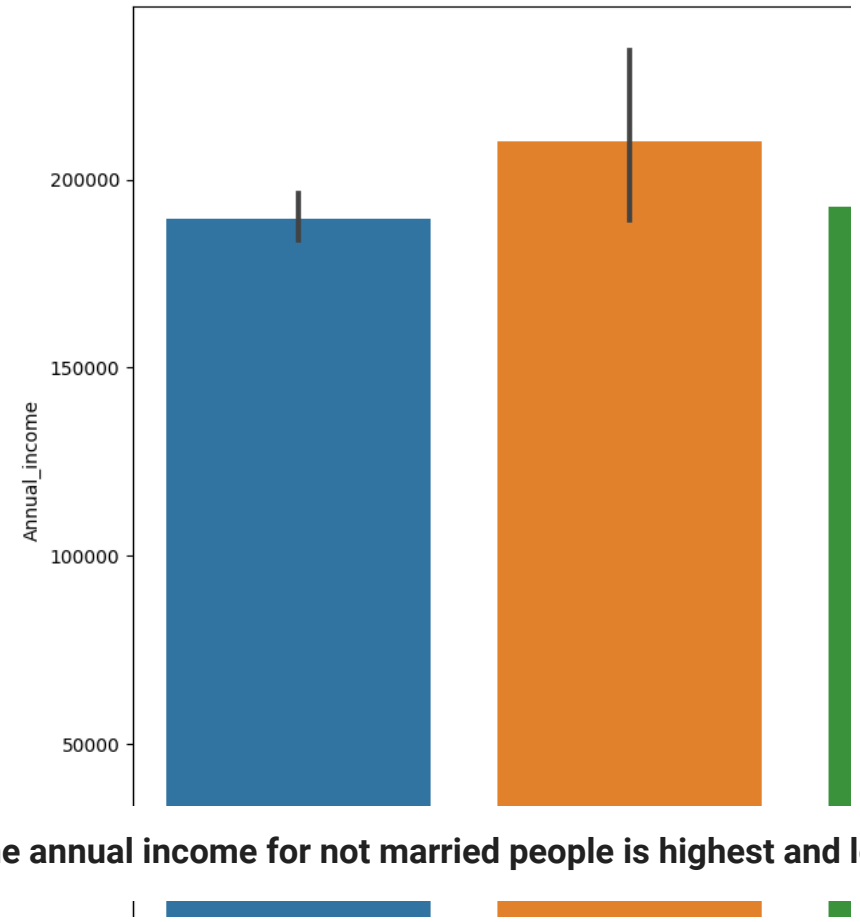
```
<Axes: xlabel='Housing_type', ylabel='Annual_income'>
```



The annual income is highest for people living in office apartment and lowest for people living in cooperative apartment.

```
# To plot annual_income and Marital_status

plt.figure(figsize=([16,9]))
sns.barplot(data=credit_card,x="Marital_status",y="Annual_income")
```

```
<Axes: xlabel='Marital_status', ylabel='Annual_income'>
```



The annual income for not married people is highest and lowest for widow people.

Data Cleaning

```
# To find the null values

credit_card.isnull().sum()
```

```
Ind_ID              0
GENDER              7
Car_Owner           0
Propert_Owner       0
CHILDREN            0
Annual_income      23
Type_Income         0
EDUCATION           0
Marital_status      0
Housing_type        0
Birthday_count     22
Employed_days       0
Mobile_phone        0
Work_Phone          0
Phone               0
EMAIL_ID            0
Type_Occupation   488
Family_Members      0
dtype: int64
```

```
# To drop duplicate values for Ind_ID because it is unique and there can be no duplicate values

credit_card= credit_card.drop_duplicates(["Ind_ID"])


# droping unneccesary rows

credit_card = credit_card.dropna(subset=["Annual_income","Birthday_count","GENDER"])


credit_card["Age"] = -(credit_card["Birthday_count"])//365
```

```
<ipython-input-32-6d7439257861>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  credit_card["Age"] = -(credit_card["Birthday_count"])//365
```

```
# dropping unneccesary columns
credit_card = credit_card.drop(['Mobile_phone','Work_Phone','Phone','EMAIL_ID','Type_Occupation','Birthday_count'],axis=1)
```

```
# parsing data

object_credit_card = pd.DataFrame(credit_card.dtypes =='object').reset_index()
object_type = object_credit_card[object_credit_card[0] == True]['index']
object_type
```

```
1            GENDER
2         Car_Owner
3     Propert_Owner
6       Type_Income
7         EDUCATION
8    Marital_status
9      Housing_type
Name: index, dtype: object
```

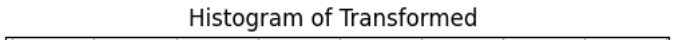
Feature Engineering

```
credit_card['Employment_Status'] = credit_card['Employed_days'].apply(lambda x: 1 if x < 0 else 0)
```

```
credit_card = credit_card.drop(["Employed_days"],axis=1)
```

```
# To import labelencoder to convert categorical data into numerical data

from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
for i in credit_card:
    if credit_card[i].dtypes == 'object':
        credit_card[i] = labelencoder.fit_transform(credit_card[i])
```

```
# transformation of data

import scipy.stats as stats
# Select the column to transform
column_name = ['Annual_income']

for j in column_name:
    # Apply Box-Cox transformation
    transformed_data, lambda_val = stats.boxcox(credit_card[j])
    credit_card[j] = transformed_data

    # Plot histogram of transformed data
    plt.hist(transformed_data, bins=10)
    plt.xlabel('Transformed ')
    plt.ylabel('Frequency')
    plt.title('Histogram of Transformed ')
    plt.grid()
    plt.show()
```



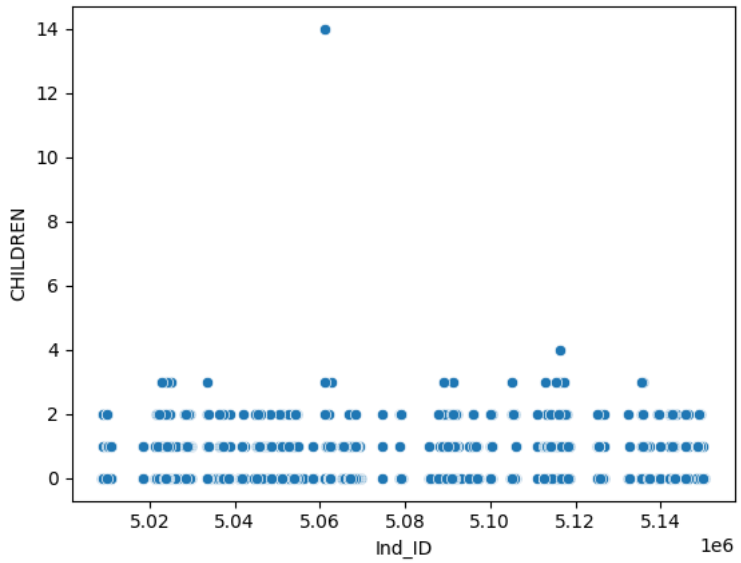
Visualisation of data


```
credit_card.head()
```

	Ind_ID	GENDER	Car_Owner	Propert_Owner	CHILDREN	Annual_income	Type_Income
0	5008827	1	1	1	0	7.368859	1
1	5009744	0	1	0	0	7.581871	0
4	5009752	0	1	0	0	7.581871	0
6	5009754	0	1	0	0	7.581871	0
7	5009894	0	0	0	0	7.368859	1

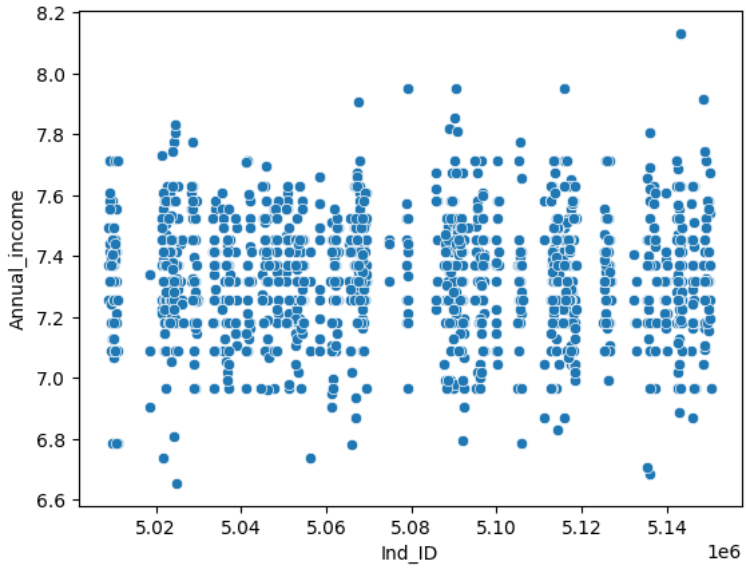
```
sns.scatterplot(data=credit_card,x="Ind_ID",y="CHILDREN")
```

```
<Axes: xlabel='Ind_ID', ylabel='CHILDREN'>
```



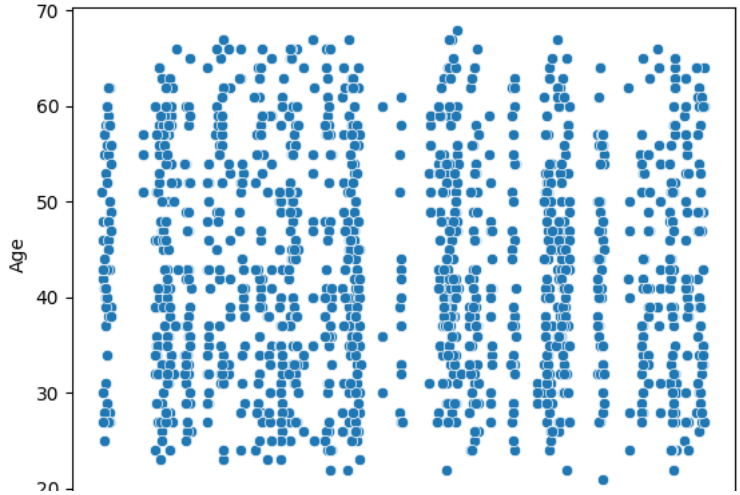

```
sns.scatterplot(data=credit_card,x="Ind_ID",y="Annual_income")
```

```
<Axes: xlabel='Ind_ID', ylabel='Annual_income'>
```



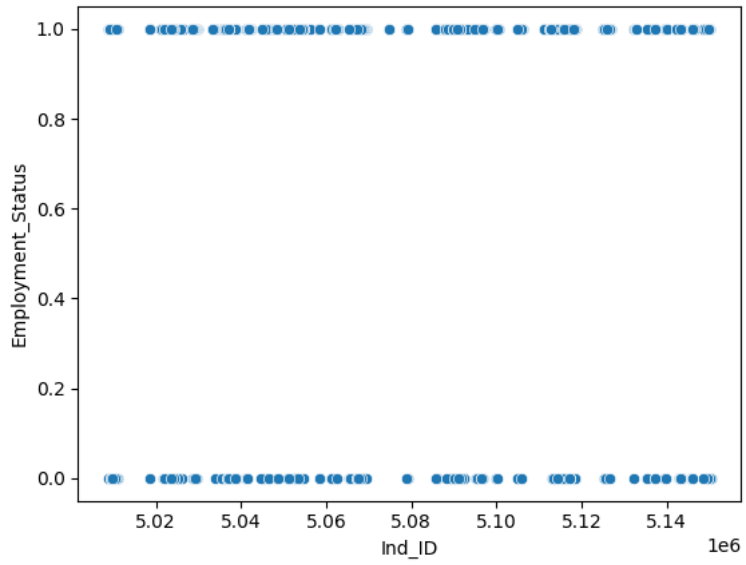

```
sns.scatterplot(data=credit_card,x="Ind_ID",y="Age")
```

```
<Axes: xlabel='Ind_ID', ylabel='Age'>
```



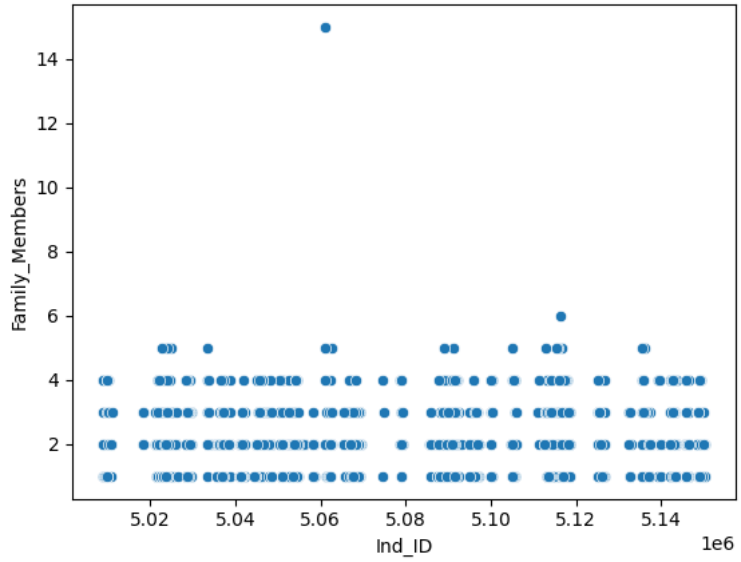

```
sns.scatterplot(data=credit_card,x="Ind_ID",y="Employment_Status")
```

```
<Axes: xlabel='Ind_ID', ylabel='Employment_Status'>
```



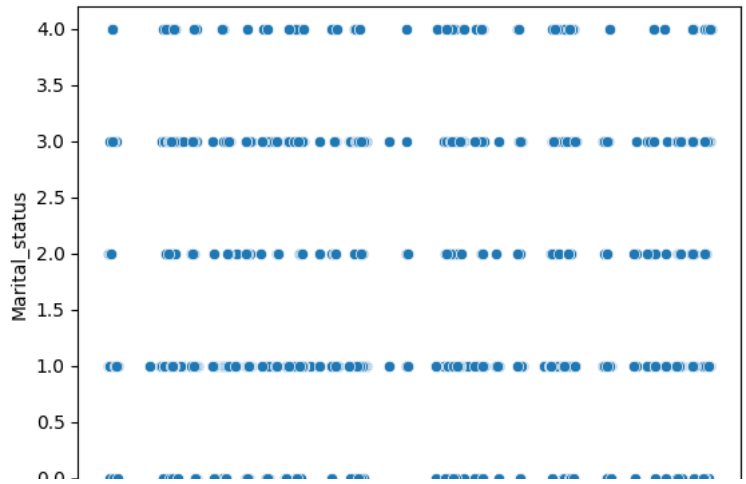

```
sns.scatterplot(data=credit_card,x="Ind_ID",y="Family_Members")
```

```
<Axes: xlabel='Ind_ID', ylabel='Family_Members'>
```



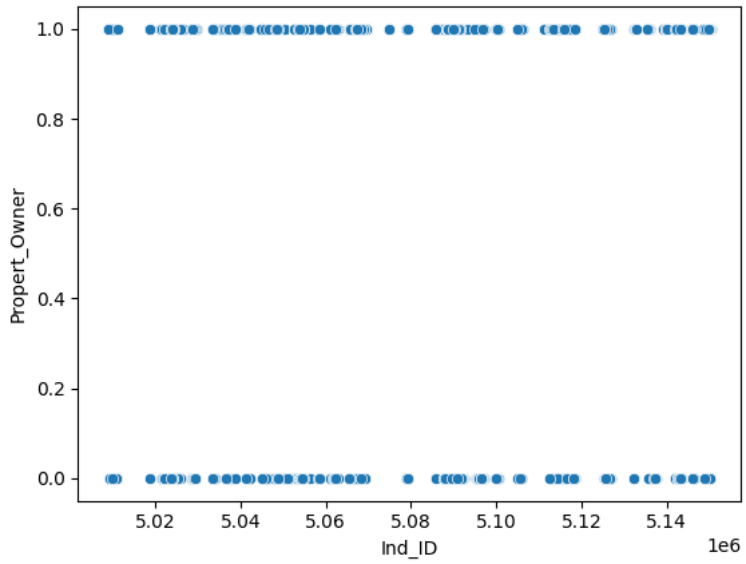

```
sns.scatterplot(data=credit_card,x="Ind_ID",y="Marital_status")
```

```
<Axes: xlabel='Ind_ID', ylabel='Marital_status'>
```




```
sns.scatterplot(data=credit_card,x="Ind_ID",y="Propert_Owner")
```

```
<Axes: xlabel='Ind_ID', ylabel='Propert_Owner'>
```

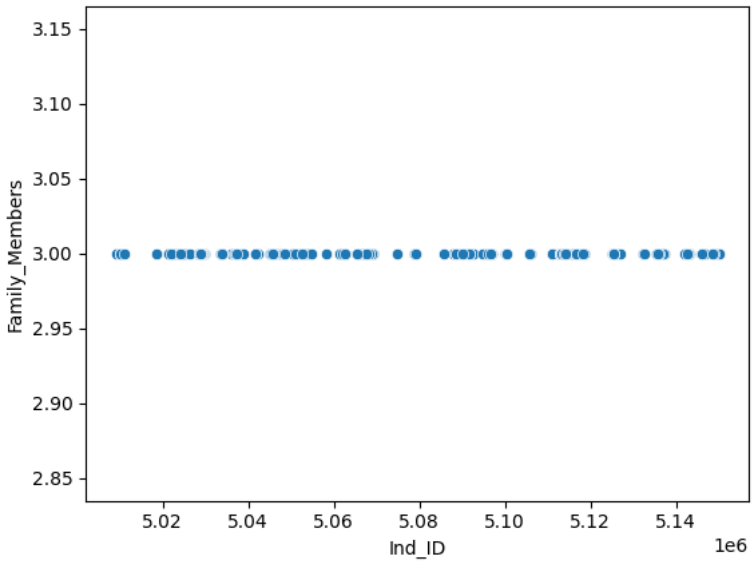



```
# to remove outliers from the dataset

upper = credit_card["CHILDREN"].quantile(0.995)
lower = credit_card["CHILDREN"].quantile(0.005)
credit_card = credit_card[(credit_card["CHILDREN"]>lower) & (credit_card["CHILDREN"]<upper)]


upper = credit_card["Family_Members"].quantile(0.995)
lower = credit_card["Family_Members"].quantile(0.005)
credit_card = credit_card[(credit_card["Family_Members"]>lower)&(credit_card["Family_Members"]<upper)]


upper = credit_card["Annual_income"].quantile(0.995)
lower = credit_card["Annual_income"].quantile(0.005)
credit_card = credit_card[(credit_card["Annual_income"]>lower)&(credit_card["Annual_income"]<upper)]


# to check whether outliers is removed from CHILDREN or not

sns.scatterplot(data=credit_card,x="Ind_ID",y="CHILDREN")
```

```
<Axes: xlabel='Ind_ID', ylabel='CHILDREN'>
```

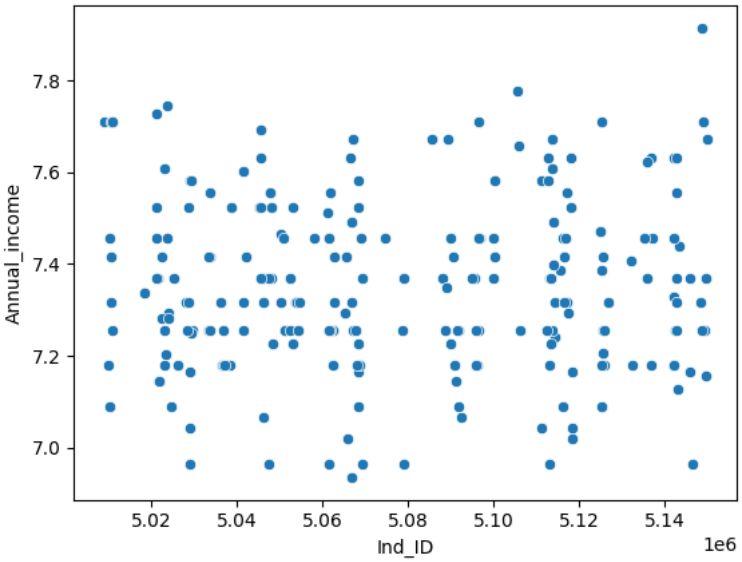

```
# to check whether outliers is removed from Family_Members or not

sns.scatterplot(data=credit_card,x="Ind_ID",y="Family_Members")
```

```
<Axes: xlabel='Ind_ID', ylabel='Family_Members'>
```




```
# To check whether outliers are present in Annual_income or not

sns.scatterplot(data=credit_card,x="Ind_ID",y="Annual_income")
```

```
<Axes: xlabel='Ind_ID', ylabel='Annual_income'>
```

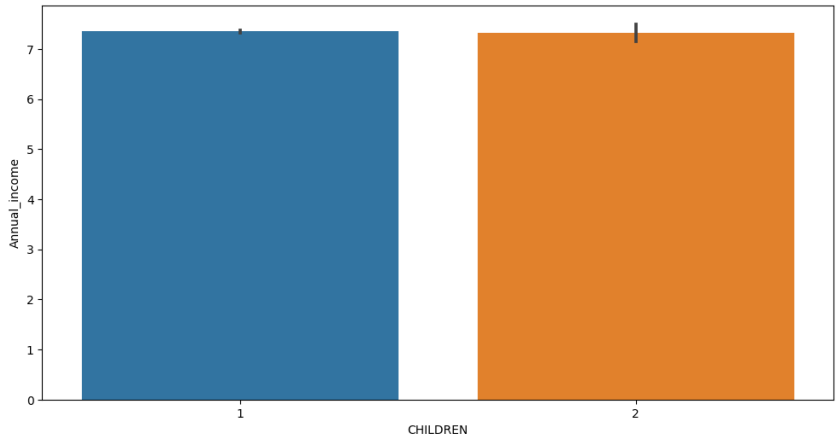



```
# lets check outliers

from scipy.stats import zscore
t = zscore(credit_card["Annual_income"])
threshold = 3
outliers = credit_card[np.abs(t)>threshold]
```

Visualisation of data

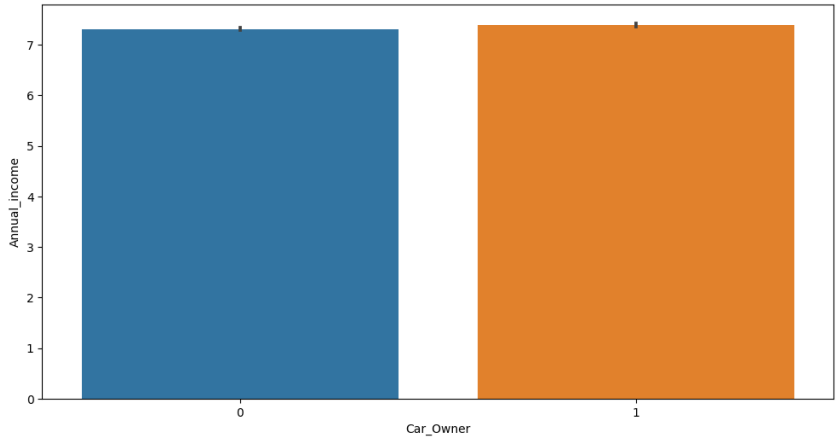
```
plt.figure(figsize=([12,6]))
sns.barplot(data=credit_card,x='CHILDREN',y="Annual_income")
```

```
<Axes: xlabel='CHILDREN', ylabel='Annual_income'>
```



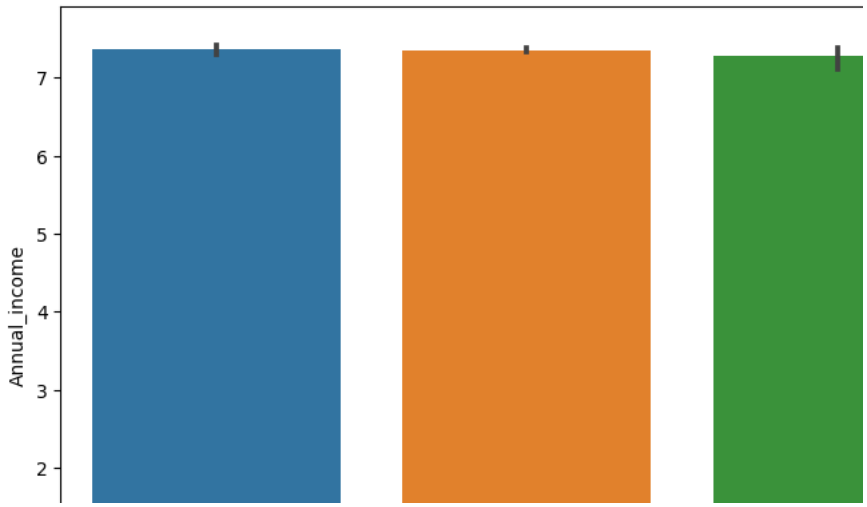

```
plt.figure(figsize=([12,6]))
sns.barplot(data=credit_card,x="Car_Owner",y="Annual_income")
```

```
<Axes: xlabel='Car_Owner', ylabel='Annual_income'>
```



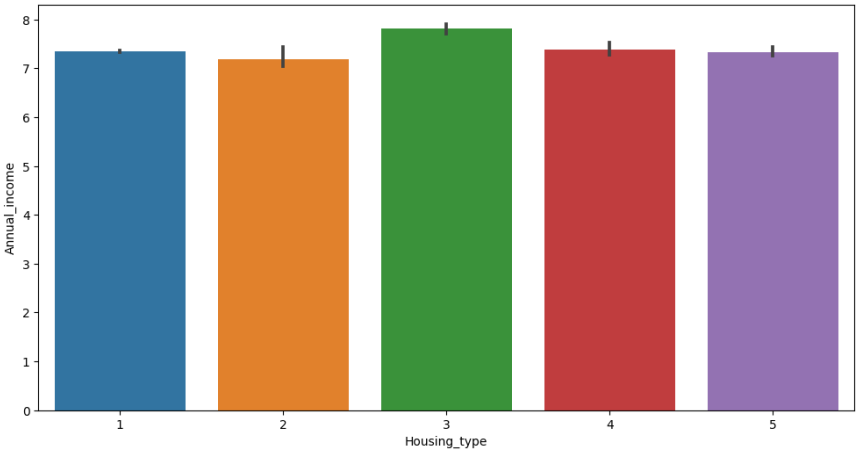

```
plt.figure(figsize=([12,6]))
sns.barplot(data=credit_card,x="Marital_status",y="Annual_income")
```

```
<Axes: xlabel='Marital_status', ylabel='Annual_income'>
```



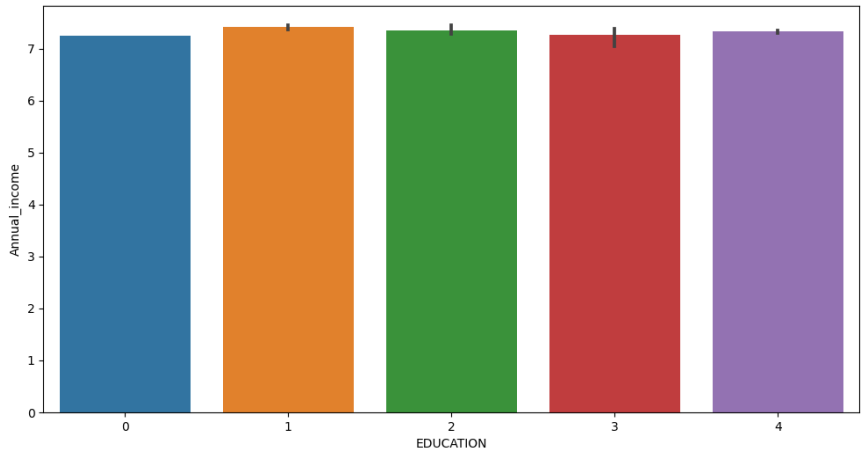

```
plt.figure(figsize=([12,6]))
sns.barplot(data=credit_card,x="Housing_type",y="Annual_income")
```

```
<Axes: xlabel='Housing_type', ylabel='Annual_income'>
```




```
plt.figure(figsize=([12,6]))
sns.barplot(data=credit_card,x="EDUCATION",y="Annual_income")
```

```
<Axes: xlabel='EDUCATION', ylabel='Annual_income'>
```



✓ 0s completed at 10:33 PM