```
pip install statsmodels

pip install scipy

Requirement already satisfied: scipy in c:\users\nikhilesh\
jupyter_env\lib\site-packages (1.16.1)
Requirement already satisfied: numpy<2.6,>=1.25.2 in c:\users\
nikhilesh\jupyter_env\lib\site-packages (from scipy) (2.3.2)
Note: you may need to restart the kernel to use updated packages.


[notice] A new release of pip is available: 25.1.1 -> 25.2
[notice] To update, run: python.exe -m pip install --upgrade pip

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

df = pd.read_csv("Churn_Modelling.csv")

df.head()

   RowNumber  CustomerId   Surname  CreditScore Geography  Gender  Age
\
0          1    15634602  Hargrave          619    France  Female   42

1          2    15647311      Hill          608     Spain  Female   41

2          3    15619304      Onio          502    France  Female   42

3          4    15701354      Boni          699    France  Female   39

4          5    15737888  Mitchell          850     Spain  Female   43


   Tenure    Balance  NumOfProducts  HasCrCard  IsActiveMember  \
0       2       0.00              1          1               1
1       1   83807.86              1          0               1
2       8  159660.80              3          1               0
3       1       0.00              2          0               0
4       2  125510.82              1          1               1

   EstimatedSalary  Exited
0        101348.88       1
1        112542.58       0
2        113931.57       1
3         93826.63       0
4         79084.10       0
```

Mean

```
mean = np.mean(df["Age"])
print(mean)
```

```
38.9218
```

Check Data Type of Columns

```
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
```

Mean Absolute Deviation

```
mad = np.sum(np.abs(df["Age"] - mean))/len(df["Age"])

print(mad)
```

```
7.94097904
```

```
print(mean)
```

```
38.9218
```

Check Standard Deviation

```
np.std(df["Age"]) , np.var(df["Age"])
```

```
(np.float64(10.487282048271611), np.float64(109.98308476))
```

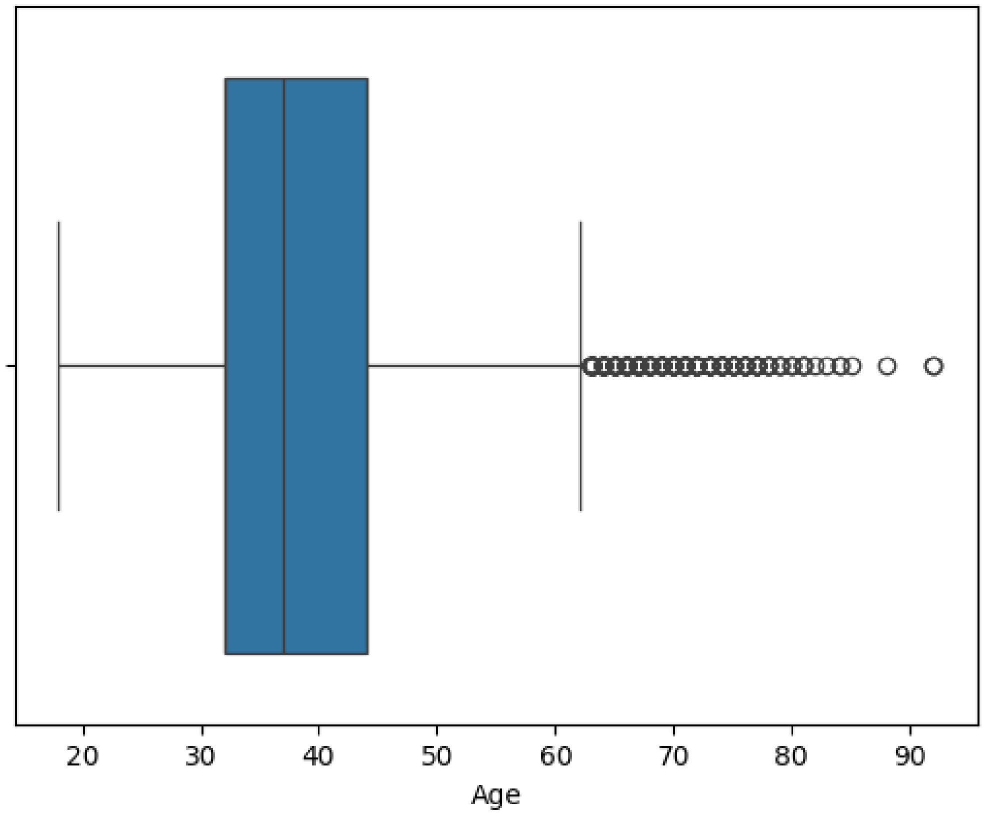
Interquartile Range

```
Q1 = df["Age"].quantile(0.25)
Q3 = df["Age"].quantile(0.75)
IQR = Q3 - Q1

lower_fence = Q1 - 1.5 * IQR
upper_fence = Q3 + 1.5 * IQR

outliers = df[(df["Age"] < lower_fence) | (df["Age"] > upper_fence)]

print("Q1:", Q1, "Q3:", Q3, "IQR:", IQR)
print("Outliers count:", outliers.shape[0])
```

```
Q1: 32.0 Q3: 44.0 IQR: 12.0
Outliers count: 359
```

Interquartile Range with plot

```
sns.boxplot(x="Age",data = df)
plt.show()
```



Check mean , STD , Minimun , Maximum

```
df.describe()

         RowNumber    CustomerId   CreditScore           Age
Tenure  \
count  10000.00000  1.000000e+04  10000.000000  10000.000000
10000.000000
mean    5000.50000  1.569094e+07    650.528800     38.921800
5.012800
std     2886.89568  7.193619e+04     96.653299     10.487806
2.892174
min        1.00000  1.556570e+07    350.000000     18.000000
0.000000
25%     2500.75000  1.562853e+07    584.000000     32.000000
3.000000
50%     5000.50000  1.569074e+07    652.000000     37.000000
5.000000
75%     7500.25000  1.575323e+07    718.000000     44.000000
7.000000
max    10000.00000  1.581569e+07    850.000000     92.000000
10.000000
```

```
             Balance  NumOfProducts    HasCrCard  IsActiveMember  \
count   10000.000000   10000.000000  10000.00000    10000.000000   
mean    76485.889288       1.530200      0.70550        0.515100   
std     62397.405202       0.581654      0.45584        0.499797   
min         0.000000       1.000000      0.00000        0.000000   
25%         0.000000       1.000000      0.00000        0.000000   
50%     97198.540000       1.000000      1.00000        1.000000   
75%    127644.240000       2.000000      1.00000        1.000000   
max    250898.090000       4.000000      1.00000        1.000000   

       EstimatedSalary        Exited  
count     10000.000000  10000.000000  
mean     100090.239881      0.203700  
std       57510.492818      0.402769  
min          11.580000      0.000000  
25%       51002.110000      0.000000  
50%      100193.915000      0.000000  
75%      149388.247500      0.000000  
max      199992.480000      1.000000  
```

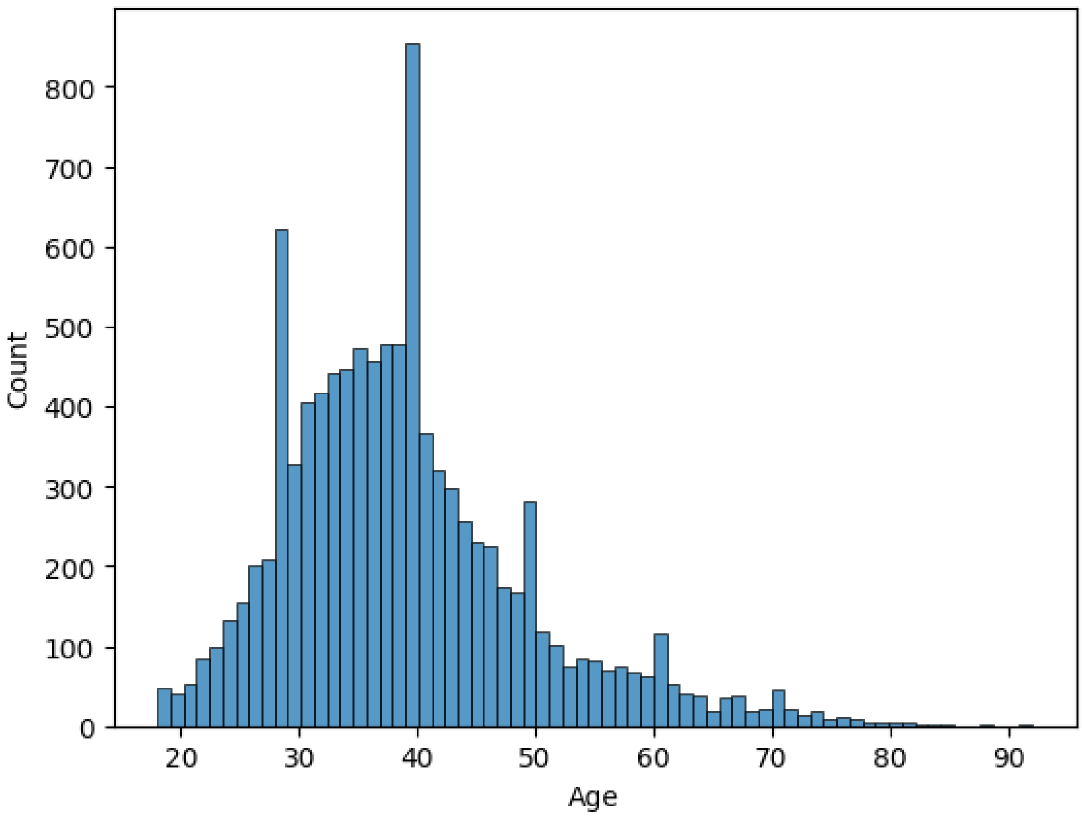
Check Skewness

```
from scipy.stats import skew, kurtosis

print("Skewness:", skew(df["Age"].dropna()))
print("Kurtosis:", kurtosis(df["Age"].dropna()))
```

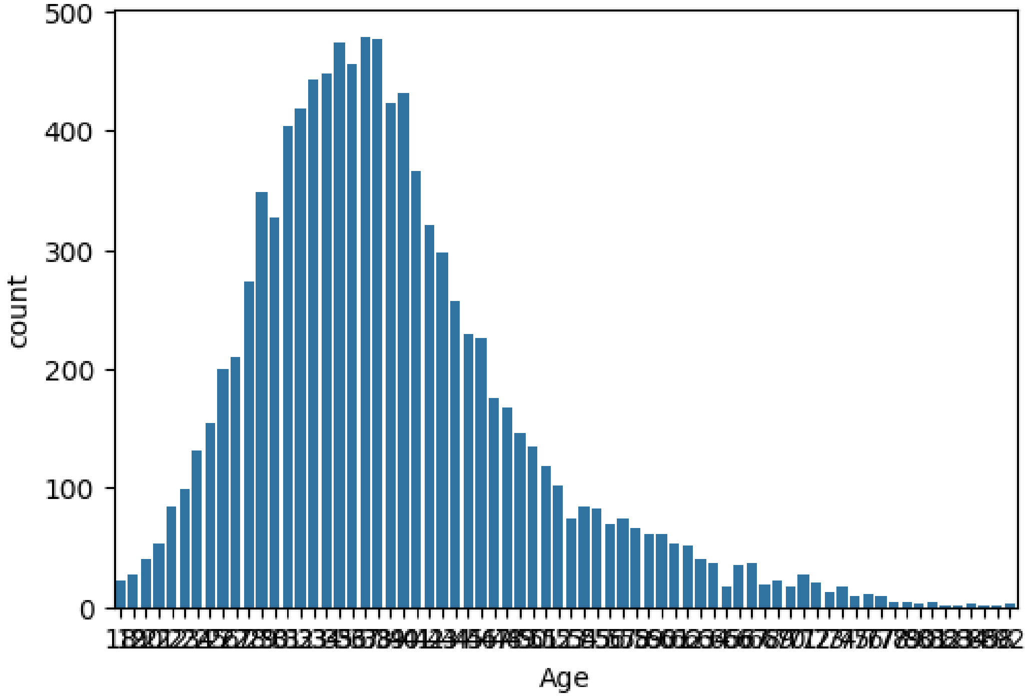
```
Skewness: 1.0111685586628076
Kurtosis: 1.3940495456392599
```

Skewness plot

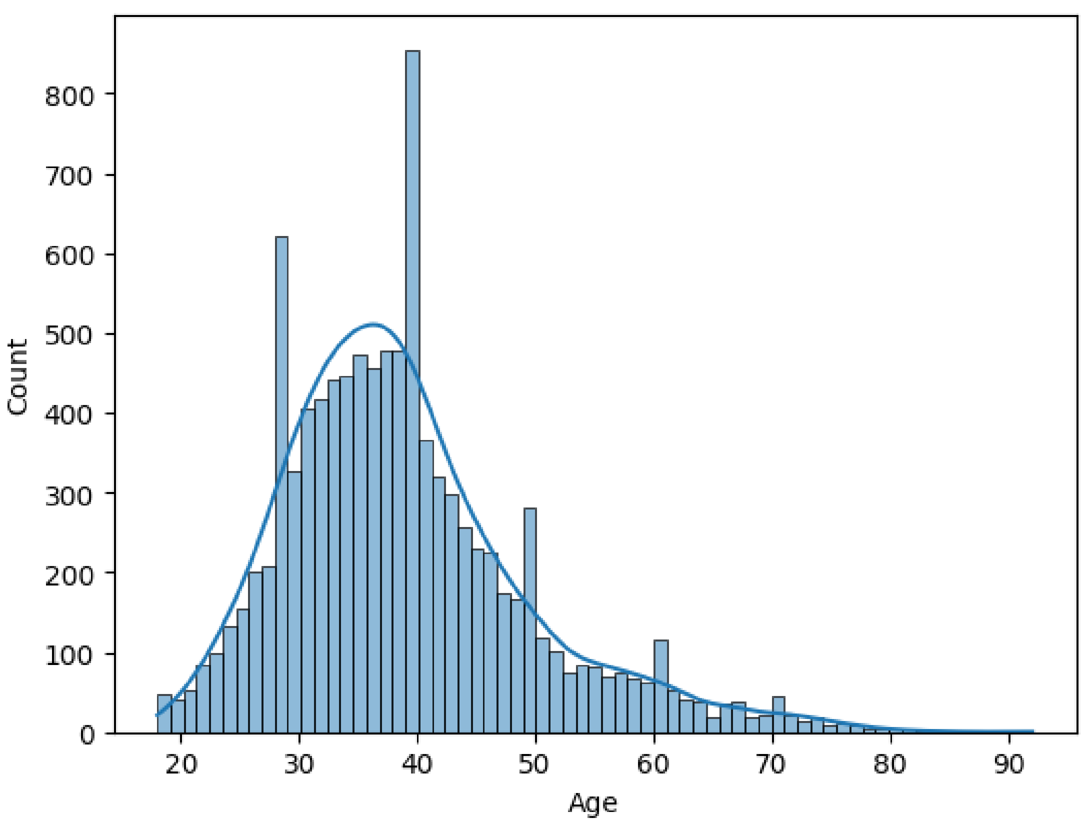
```
sns.histplot(x= "Age" , data= df)
plt.show()
```




```
plt.figure(figsize=(6,4))
sns.countplot(x = "Age" , data = df)
plt.show()
```

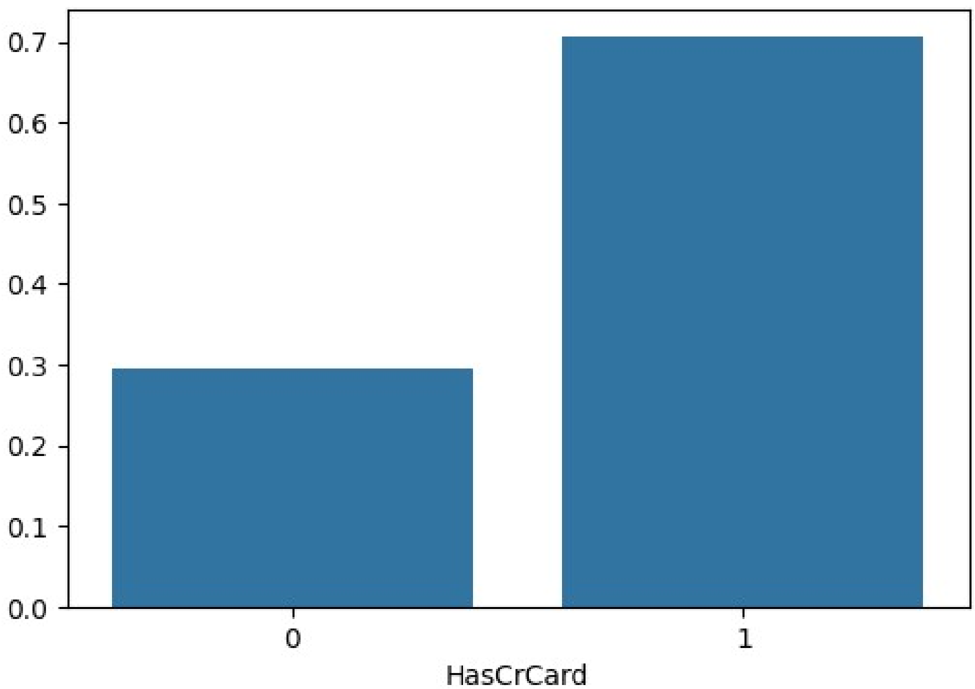



```
sns.histplot(df["Age"].dropna(), kde = True)
plt.show()
```



```
survival_prob = df['HasCrCard'].value_counts(normalize=True)

plt.figure(figsize=(6,4))
sns.barplot(x=survival_prob.index, y=survival_prob.values)
```

```
<Axes: xlabel='HasCrCard'>
```




```
df.tail()
```

```
      RowNumber  CustomerId    Surname  CreditScore Geography  Gender
Age  \
9995       9996    15606229   Obijiaku          771    France    Male
39
9996       9997    15569892  Johnstone          516    France    Male
35
9997       9998    15584532        Liu          709    France  Female
36
9998       9999    15682355  Sabbatini          772   Germany    Male
42
9999      10000    15628319     Walker          792    France  Female
28

      Tenure    Balance  NumOfProducts  HasCrCard  IsActiveMember  \
9995       5       0.00              2          1               0
9996      10   57369.61              1          1               1
9997       7       0.00              1          0               1
9998       3   75075.31              2          1               0
9999       4  130142.79              1          1               0
```

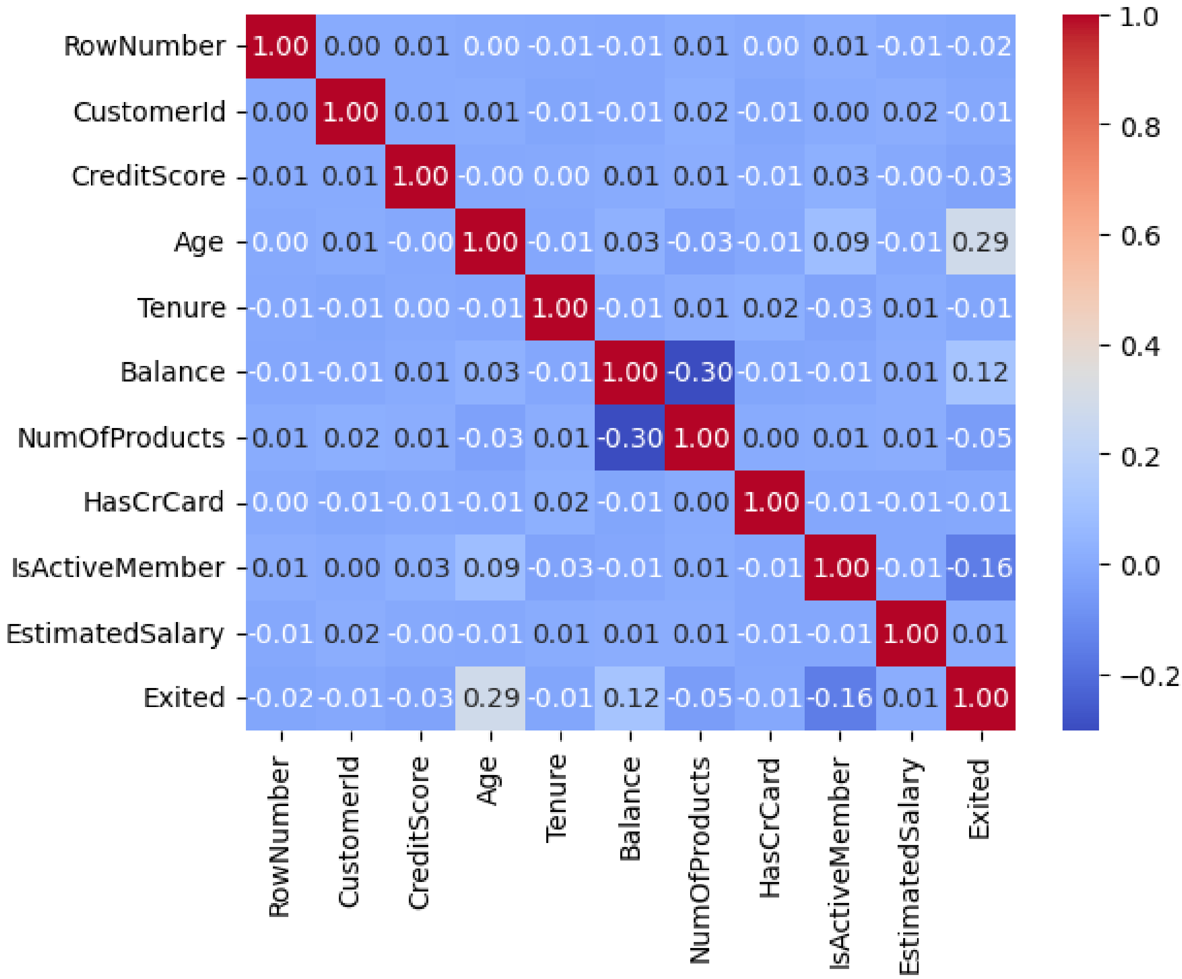
```
      EstimatedSalary  Exited
9995         96270.64       0
9996        101699.77       0
9997         42085.58       1
9998         92888.52       1
9999         38190.78       0
```

Check Correlation

```
corr = df.corr(numeric_only=True)
print(corr['Age'].sort_values(ascending=False))

# Heatmap visualization
sns.heatmap(corr, annot=True, cmap="coolwarm", fmt=".2f")
```

```
Age                1.000000
Exited             0.285323
IsActiveMember     0.085472
Balance            0.028308
CustomerId         0.009497
RowNumber          0.000783
CreditScore       -0.003965
EstimatedSalary   -0.007201
Tenure            -0.009997
HasCrCard         -0.011721
NumOfProducts     -0.030680
Name: Age, dtype: float64
```

```
<Axes: >
```



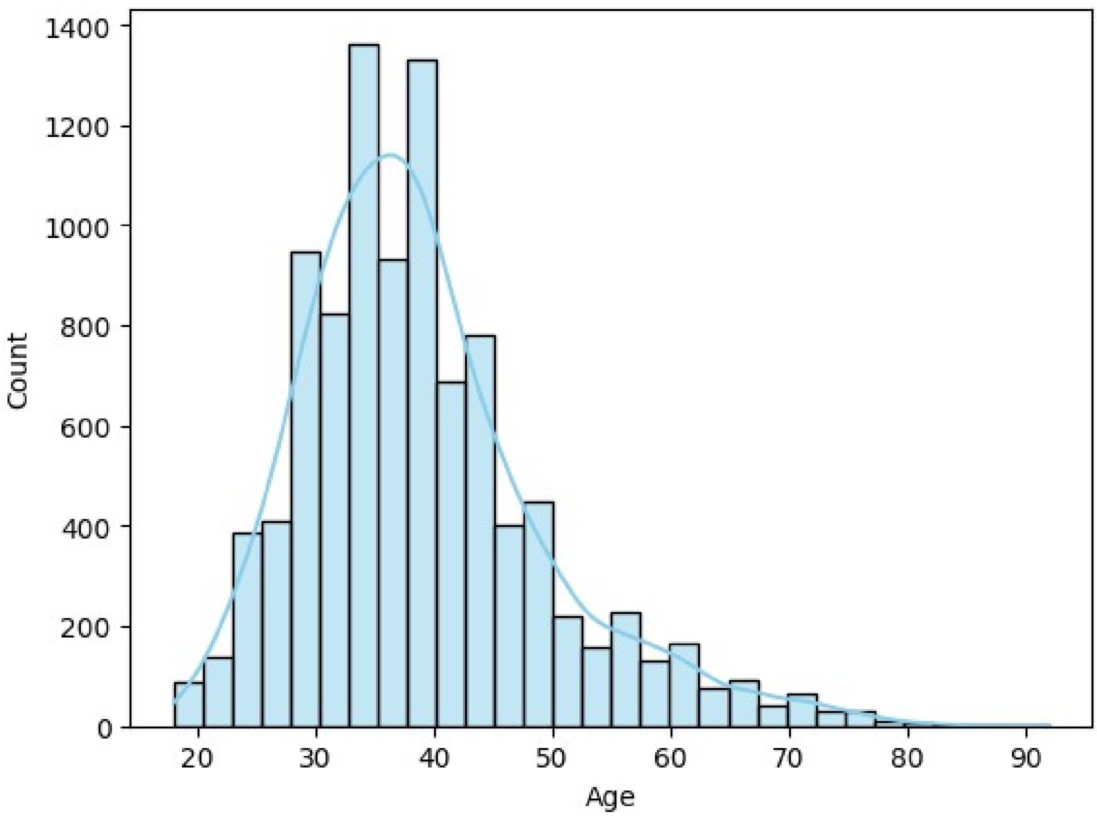
Check P-Vlaue

```
from scipy.stats import chi2_contingency

table = pd.crosstab(df['HasCrCard'], df['Exited'])
print("Contingency Table:\n", table)

# Chi-square test
chi2, p, dof, expected = chi2_contingency(table)

print("\nChi-square Statistic:", chi2)
print("Degrees of Freedom:", dof)
print("p-value:", p)
```

```
Contingency Table:
 Exited        0     1
HasCrCard
0          2332   613
```

```
1          5631  1424

Chi-square Statistic: 0.47133779904440803
Degrees of Freedom: 1
p-value: 0.49237236141554697
```

```
sns.histplot(df["Age"], bins=30, kde=True, color="skyblue")
plt.show()
```




```
df.head()
```

```
   RowNumber  CustomerId   Surname  CreditScore Geography  Gender  Age
\
0          1    15634602  Hargrave          619    France  Female   42

1          2    15647311      Hill          608     Spain  Female   41

2          3    15619304      Onio          502    France  Female   42

3          4    15701354      Boni          699    France  Female   39

4          5    15737888  Mitchell          850     Spain  Female   43


   Tenure    Balance  NumOfProducts  HasCrCard  IsActiveMember  \
```

```
0       2       0.00              1          1               1
1       1   83807.86              1          0               1
2       8  159660.80              3          1               0
3       1       0.00              2          0               0
4       2  125510.82              1          1               1

   EstimatedSalary  Exited
0        101348.88       1
1        112542.58       0
2        113931.57       1
3         93826.63       0
4         79084.10       0
```

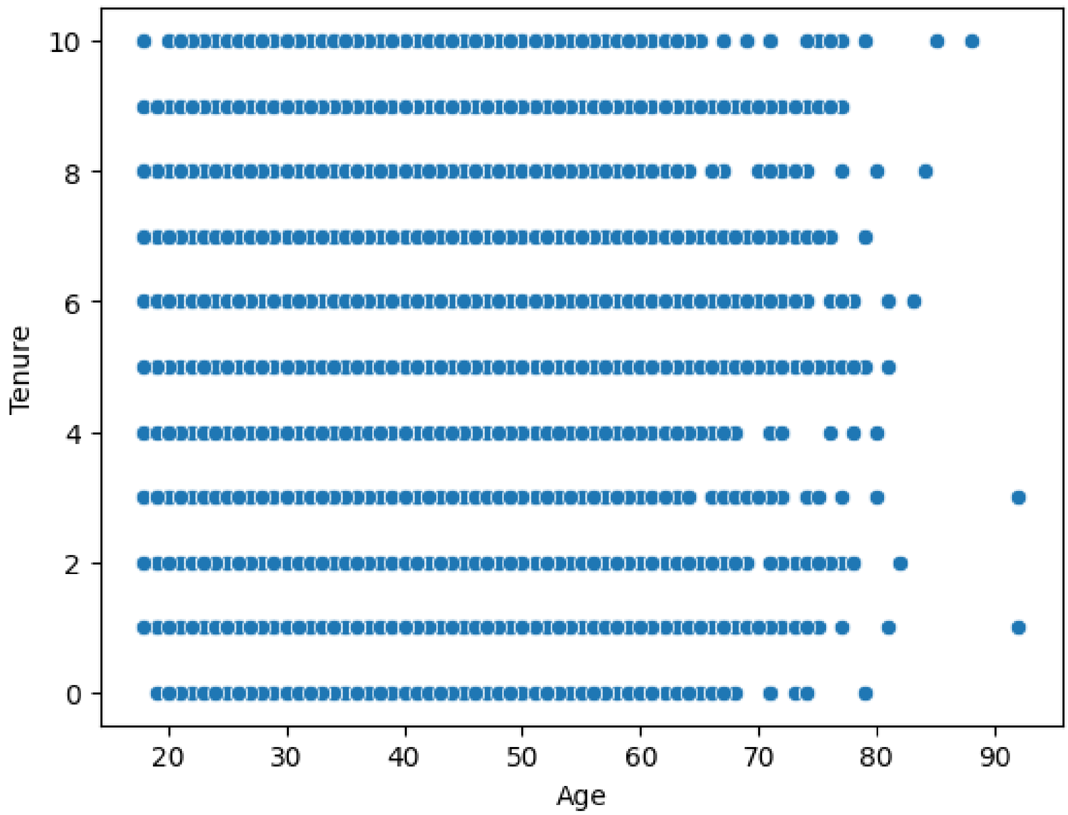
Check Normal Distribution

```
from scipy.stats import normaltest

stat, p = normaltest(df["Age"])
print("Test Statistic:", stat)
print("p-value:", p)

if p < 0.05:
    print("Conclusion: Age distribution is NOT perfectly normal.")
else:
    print("Conclusion: Age distribution follows normal distribution.")

Test Statistic: 1507.7908881363314
p-value: 0.0
Conclusion: Age distribution is NOT perfectly normal.
```

Check Covariance

```
num_df = df[['Age','Tenure']].dropna()

# Covariance matrix
cov_matrix = num_df.cov()
print("Covariance Matrix:\n", cov_matrix)

# Individual covariance value
cov_age_fare = num_df['Age'].cov(num_df['Tenure'])
print("\nCovariance between Age and Fare:", cov_age_fare)

Covariance Matrix:
                 Age    Tenure
Age      109.994084 -0.303229
Tenure    -0.303229  8.364673
```

```
Covariance between Age and Fare: -0.30322936293629343

num_df = df[['Age','Tenure']].dropna()
sns.scatterplot(x='Age', y='Tenure', data=num_df)
plt.show()
```

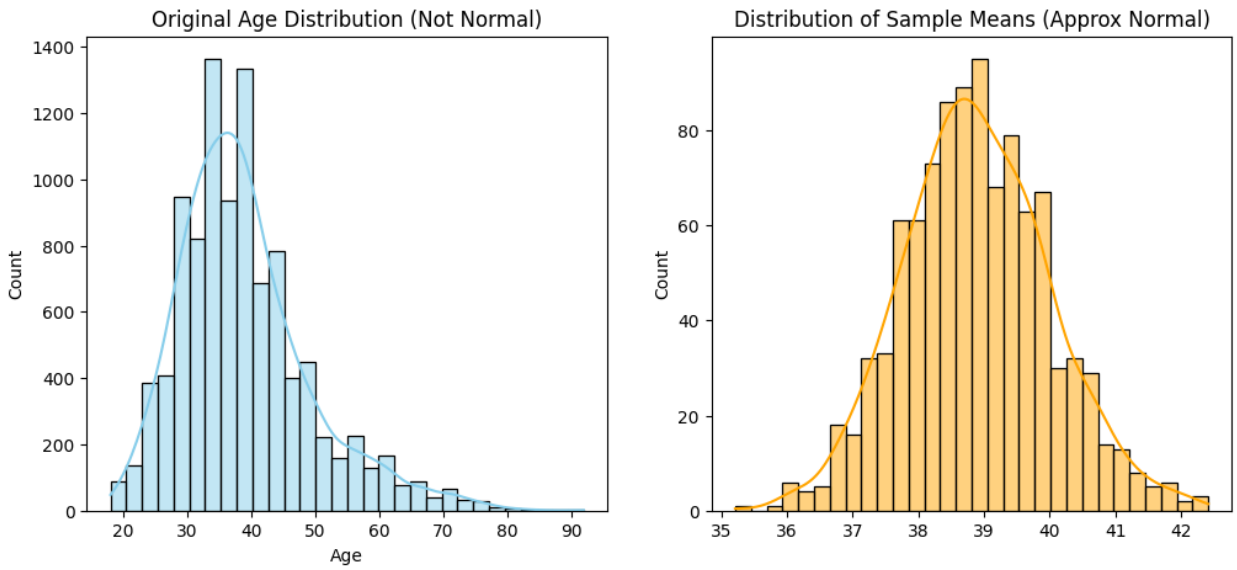


Central Limit Theorem

```
sample_means = []
for i in range(1000):
    sample = np.random.choice(df["Age"], size=80, replace=True)
    sample_means.append(sample.mean())

# Plot original age distribution
plt.figure(figsize=(12,5))

plt.subplot(1,2,1)
sns.histplot(df["Age"], bins=30, kde=True, color="skyblue")
plt.title("Original Age Distribution (Not Normal)")

# Plot sampling distribution of sample means
plt.subplot(1,2,2)
```

```
sns.histplot(sample_means, bins=30, kde=True, color="orange")
plt.title("Distribution of Sample Means (Approx Normal)")
```

```
Text(0.5, 1.0, 'Distribution of Sample Means (Approx Normal)')
```




```
np.mean(sample_means)
```

```
np.float64(38.883925)
```

```
np.mean(df["Age"])
```

```
np.float64(38.9218)
```

Z- Test

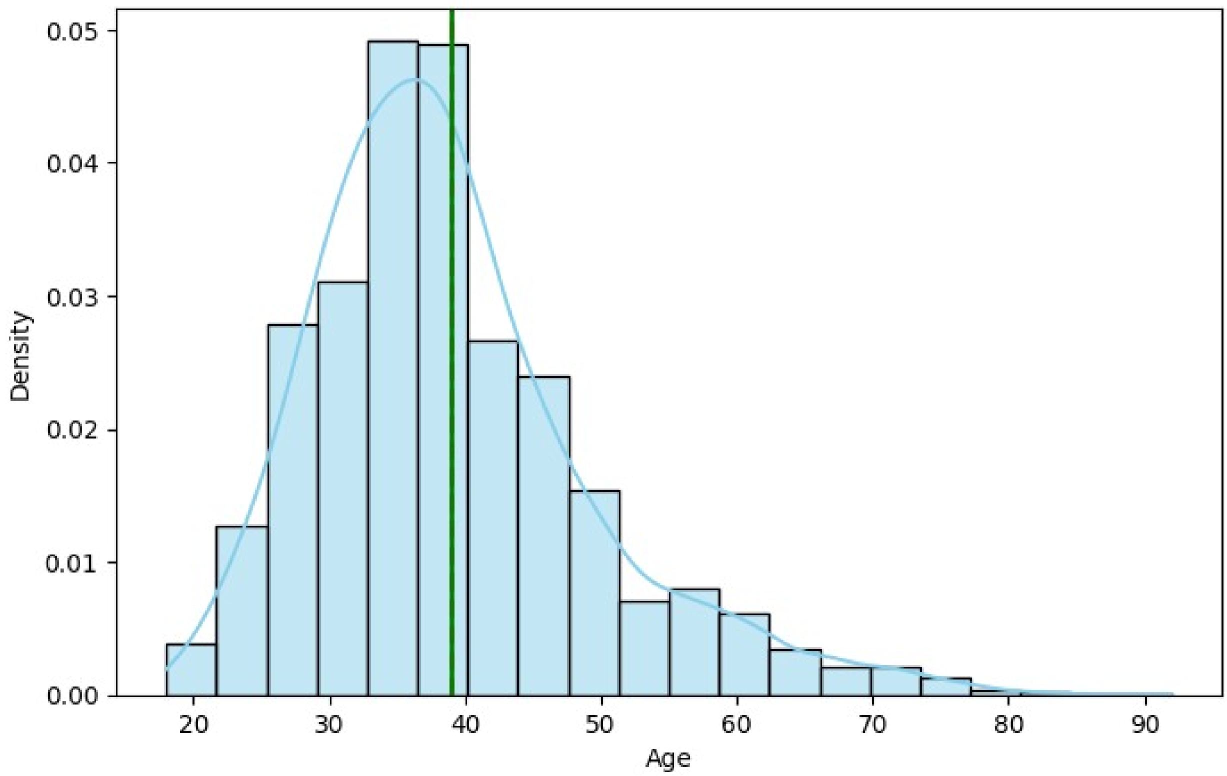
```
from statsmodels.stats.weightstats import ztest

data = df['Age']

# Perform one-sample Z-test (test mean = 5.8)
z_stat, p_val = ztest(data, value=5.8)

print("Z-statistic:", z_stat)
print("p-value:", p_val)

# Decision rule
alpha = 0.05
if p_val < alpha:
    print("Reject Null Hypothesis → Mean is significantly different 
from 5.8")
```

```
else:
    print("Fail to Reject Null → No significant difference from 5.8")
```

```
Z-statistic: 315.81246424142995
p-value: 0.0
Reject Null Hypothesis → Mean is significantly different from 5.8
```

```
data = df['Age']

# Hypothesized population mean
mu = 38.92

# Perform one-sample Z-test
z_stat, p_val = ztest(data, value=mu)

# Plot histogram + KDE
sns.histplot(data, bins=20, kde=True, color="skyblue", stat="density")

# Plot hypothesized mean line
plt.axvline(mu, color='red', linestyle='--', linewidth=2,
label=f"Hypothesized Mean = {mu}")

# Plot sample mean line
sample_mean = np.mean(data)
plt.axvline(sample_mean, color='green', linestyle='-', linewidth=2,
label=f"Sample Mean = {sample_mean:.2f}")
```

```
<matplotlib.lines.Line2D at 0x1c87bc49d10>
```



Z-Test with t_table and t_calculate

```
import scipy.stats as st

z_t = st.norm.ppf(0.95)
print(z_t)

1.6448536269514722

z_score = (10.487282048271611 / np.sqrt(1000))
print(z_score)

0.331636977371342

if z_table < z_score:
    print("h1 is correct")
else:
    print("h0 is correct")

h0 is correct
```

Check Mean with Sample Data

```
sample_mean = df.loc[:4000, "Age"].mean()
print(sample_mean)

38.895526118470386
```

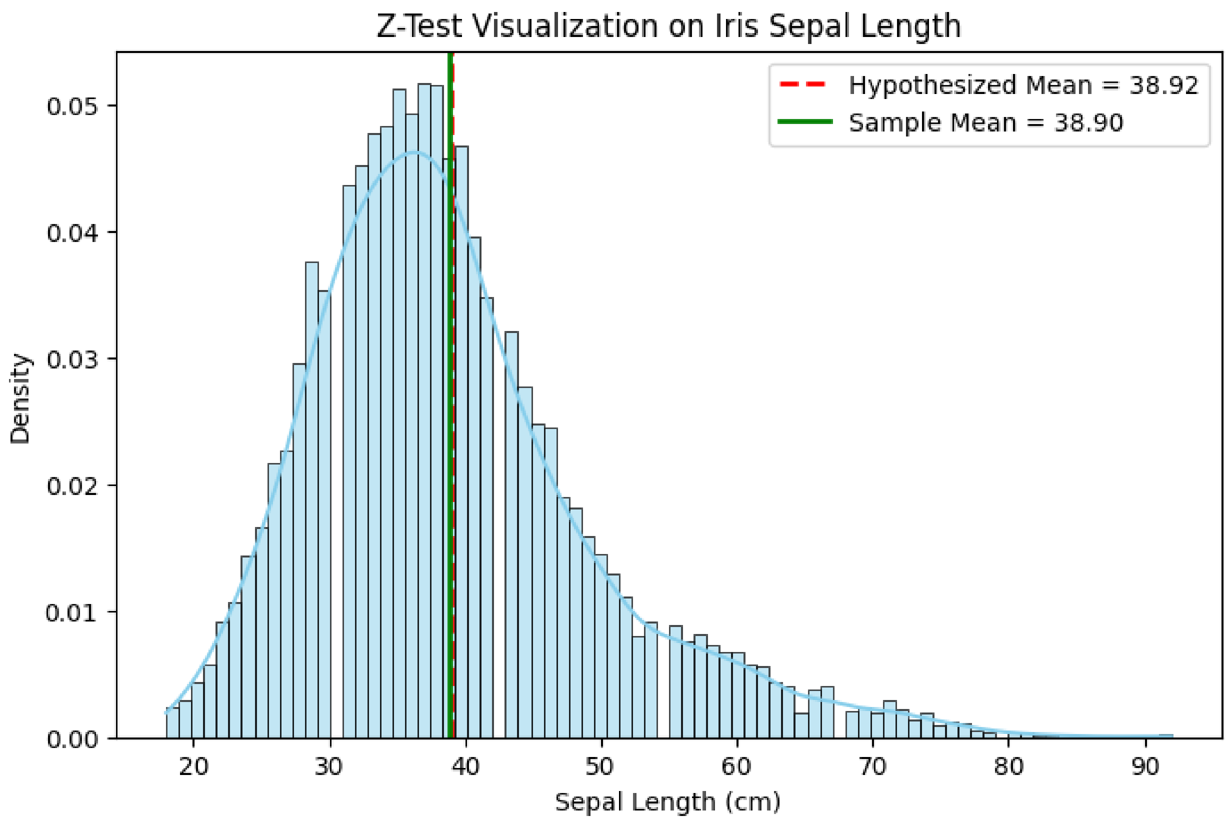
Z-test with t_table and t_Calculate

```
import scipy.stats as st

z_table = st.norm.ppf(0.95)
print(z_table)

1.6448536269514722

std = np.std(df["Age"])
print(std)

10.487282048271611

z_cal = (38.89 - 38.92)/(10.48/np.sqrt(4000))
print(z_cal)

-0.18104643092567743
```

Z-test with Plot

```
from statsmodels.stats.weightstats import ztest

data = df['Age']

# Hypothesized population mean
mu = 38.92

# Perform one-sample Z-test
z_stat, p_val = ztest(data, value=mu)

# Plot histogram + KDE
plt.figure(figsize=(8,5))
sns.histplot(data, bins=80, kde=True, color="skyblue", stat="density")

# Plot hypothesized mean line
plt.axvline(mu, color='red', linestyle='--', linewidth=2,
label=f"Hypothesized Mean = {mu}")

# Plot sample mean line
#sample_mean = np.mean(data)
```

```
sample_mean = df.loc[:4000, "Age"].mean()
plt.axvline(sample_mean, color='green', linestyle='-', linewidth=2,
label=f"Sample Mean = {sample_mean:.2f}")

# Title and labels
plt.title("Z-Test Visualization on Iris Sepal Length")
plt.xlabel("Sepal Length (cm)")
plt.ylabel("Density")
plt.legend()
plt.show()

print("Z-statistic:", z_stat)
print("p-value:", p_val)
```




```
Z-statistic: 0.017162788122422362
p-value: 0.9863067485904023
```

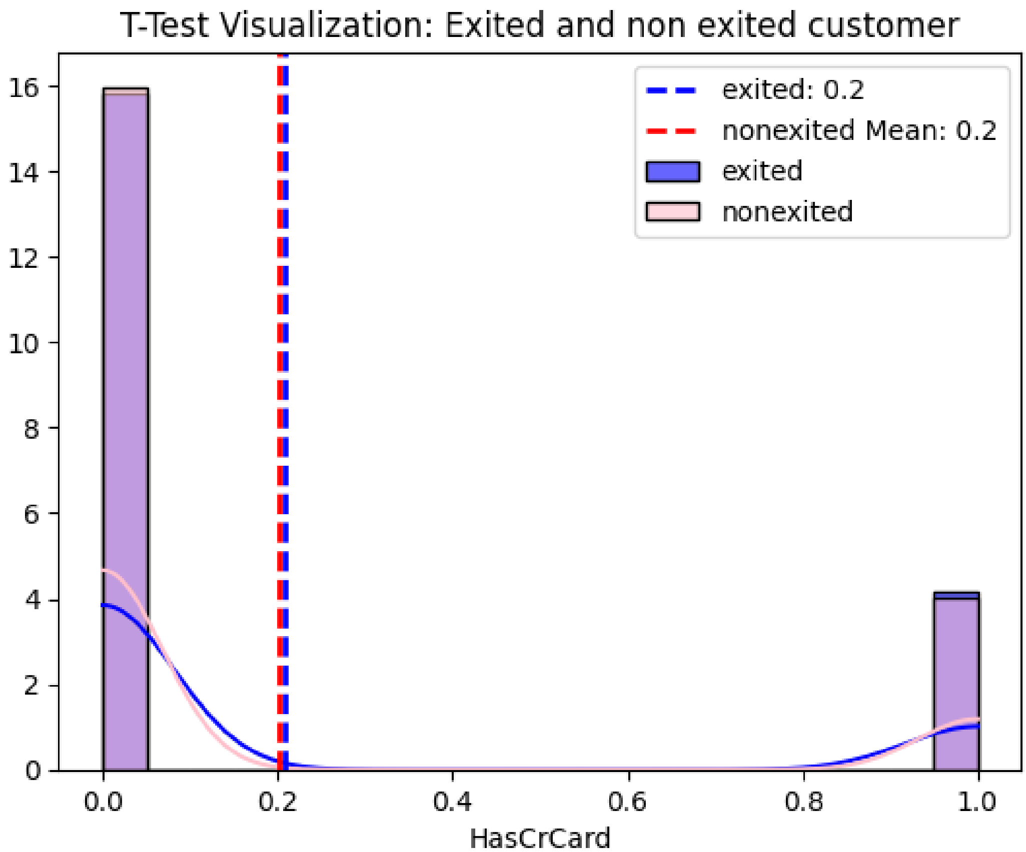
T-Test

```
from scipy.stats import ttest_ind

df = df.dropna(subset=["Exited", "HasCrCard"])
```

```python
# Split male and female groups
exited = df[df["HasCrCard"] == 0]["Exited"]
nonexited = df[df["HasCrCard"] == 1]["Exited"]

# Perform independent T-test
t_stat, p_val = ttest_ind(exited, nonexited)

print("T-statistic:", t_stat)
print("p-value:", p_val)

# Decision rule
alpha = 0.05
if p_val < 0.05:
    print("Reject Null Hypothesis.")
else:
    print("Fail to Reject Null")
```

```
T-statistic: 0.7137233605912553
p-value: 0.47541491837605965
Fail to Reject Null
```

```python
sns.histplot(exited, bins=20, kde=True, color="blue", label="exited",
stat="density", alpha=0.6)
sns.histplot(nonexited, bins=20, kde=True, color="pink",
label="nonexited", stat="density", alpha=0.6)

plt.axvline(exited.mean(), color="blue", linestyle="--", linewidth=2,
label=f"exited: {male_mass.mean():.1f}")
plt.axvline(nonexited.mean(), color="red", linestyle="--",
linewidth=2, label=f"nonexited Mean: {female_mass.mean():.1f}")

plt.title("T-Test Visualization: Exited and non exited customer")
plt.xlabel("HasCrCard")
plt.ylabel("")
plt.legend()
plt.show()
```



Check Causation

```
import statsmodels.api as sm
df.dropna()

# Example: Does total_bill cause tip amount?
X = df["HasCrCard"]     # Independent variable
y = df["Exited"]            # Dependent variable

# Add constant for intercept
X = sm.add_constant(X)

# Fit linear regression model
model = sm.OLS(y, X).fit()

# Summary gives p-values and R-squared (to infer causation with
caution)
print(model.summary())
```

```
                            OLS Regression Results

==============================================================================
```

```
========
Dep. Variable:                Exited   R-squared:
0.000
Model:                           OLS   Adj. R-squared:
-0.000
Method:                Least Squares   F-statistic:
0.5094
Date:               Mon, 15 Sep 2025   Prob (F-statistic):
0.475
Time:                       18:02:07   Log-Likelihood:
-5094.7
No. Observations:              10000   AIC:
1.019e+04
Df Residuals:                   9998   BIC:
1.021e+04
Df Model:                          1

Covariance Type:           nonrobust

======================================================================
========
                 coef    std err          t      P>|t|      [0.025
0.975]
----------------------------------------------------------------------
--------
const          0.2081      0.007     28.045      0.000       0.194
0.223
HasCrCard     -0.0063      0.009     -0.714      0.475      -0.024
0.011
======================================================================
========
Omnibus:                    2043.753   Durbin-Watson:
1.994
Prob(Omnibus):                 0.000   Jarque-Bera (JB):
3619.106
Skew:                          1.471   Prob(JB):
0.00
Kurtosis:                      3.165   Cond. No.
3.45
======================================================================
========

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
```

1. Dep. Variable: Exited

Ye dependent variable hai (target) — matlab hum predict kar rahe hain ki customer Exited (churn kiya) ya nahi.

1. R-squared: 0.000

R^2 measure karta hai ki model dependent variable ka kitna variance explain karta hai.

0.000 matlab model almost kuch bhi explain nahi kar raha (predictive power ≈ zero).

1. F-statistic: 0.5094, Prob (F-statistic): 0.475

F-test check karta hai ki model overall significant hai ya nahi.

p-value (0.475) > 0.05 → model significant nahi hai → explanatory variable meaningful nahi hai.

1. coef (coefficients)

const = 0.2081 → Intercept (baseline probability of exit ~20.8%).

HasCrCard = -0.0063 → Agar customer ke paas credit card hai to exit probability ~0.6% kam hoti hai.

1. P>|t| values (Hypothesis testing)

For HasCrCard, p = 0.475 (> 0.05). Matlab HasCrCard ka effect statistically significant nahi hai.

Hum null hypothesis (no effect) reject nahi kar paate.

1. Confidence Interval [0.025, 0.975]

HasCrCard ka interval = [-0.024, 0.011].

Zero is inside interval → again, effect not significant.

1. Other stats

Durbin-Watson = 1.994 → Residuals ka autocorrelation theek hai (~2 = good).

Omnibus / Jarque-Bera → Residuals normality test (significant → normality issue).

Summary (Simple Words)

Model explain nahi kar raha ($R^2 = 0$).

Predictor (HasCrCard) ka effect statistically insignificant hai (p = 0.475).

Matlab credit card hone ka churn par koi meaningful impact nahi hai is dataset me.

"OLS results show ki HasCrCard ka coefficient -0.0063 hai but p-value 0.475 hai (>0.05). Iska matlab hai ki credit card hone ka customer churn par koi statistically significant impact nahi hai. R^2 value bhi ~0 hai, jo dikhata hai ki model exit variable ko explain nahi kar pa raha."

```
# Probabli

total = len(df)

# Example 1: Probability of survival
p_exited = df['Exited'].sum() / total
```

```python
# Example 2: Probability of male passenger
p_male = (df['Gender'] == 'Male').sum() / total

# Example 3: Joint probability → male AND survived
p_male_exited = len(df[(df['Gender']=='Male') & (df['Exited']==1)]) /
total

# Example 4: Conditional probability → P(Survived | Female)
p_exited_given_female = len(df[(df['Gender']=='Female') &
(df['Exited']==1)]) / (df['Gender']=='Female').sum()

print("P(Exited):", round(p_exited, 3))
print("P(Male):", round(p_male, 3))
print("P(Male ∩ Exited):", round(p_male_exited, 3))
print("P(Exited | Female):", round(p_exited_given_female, 3))
```

```
P(Exited): 0.204
P(Male): 0.546
P(Male ∩ Exited): 0.09
P(Exited | Female): 0.251
```