1. What is Regression? Explain With Example

A regression model provides a function that describes the relationship between one or more independent variables and a response, dependent, or target variable. For example, the relationship between height and weight may be described by a linear regression model.

2. Give a Brief Note on Linear Regression.

Linear regression is a data analysis technique that predicts the value of unknown data by using another related and known data value. It mathematically models the unknown or dependent variable and the known or independent variable as a linear equation. For instance, suppose that you have data about your expenses and income for last year. Linear regression techniques analyze this data and determine that your expenses are half your income. They then calculate an unknown future expense by halving a future known income.

Some types of regression analysis are more suited to handle complex datasets than others. The following are some examples.

Simple linear regression

Simple linear regression is defined by the linear function:

Y= β0*X + β1 + ε

β0 and β1 are two unknown constants representing the regression slope, whereas ε (epsilon) is the error term.

You can use simple linear regression to model the relationship between two variables, such as these:

- Rainfall and crop yield
- Age and height in children
- Temperature and expansion of the metal mercury in a thermometer #### Multiple linear regression In multiple linear regression analysis, the dataset contains one dependent variable and multiple independent variables. The linear regression line function changes to include more factors as follows:

Y= β0*X0 + β1X1 + β2X2+…… βnXn+ ε

As the number of predictor variables increases, the β constants also increase correspondingly.

Multiple linear regression models multiple variables and their impact on an outcome:

- Rainfall, temperature, and fertilizer use on crop yield
- Diet and exercise on heart disease
- Wage growth and inflation on home loan rates #### Logistic regression Data scientists use logistic regression to measure the probability of an event occurring. The prediction is a value between 0 and 1, where 0 indicates an event that is unlikely to happen, and 1 indicates a maximum likelihood that it will happen. Logistic equations use logarithmic functions to compute the regression line.

These are some examples:

- The probability of a win or loss in a sporting match
- The probability of passing or failing a test
- The probability of an image being a fruit or an animal

3. Take two Variables x, y with data of your own and Find R² for fitting the best line.

In [50]:
```
import warnings
warnings.simplefilter("ignore")
```

In [51]:
```
df1=pd.read_csv("Salary_data.csv")
```

In [52]:
```
df1.head()
```

Out[52]:

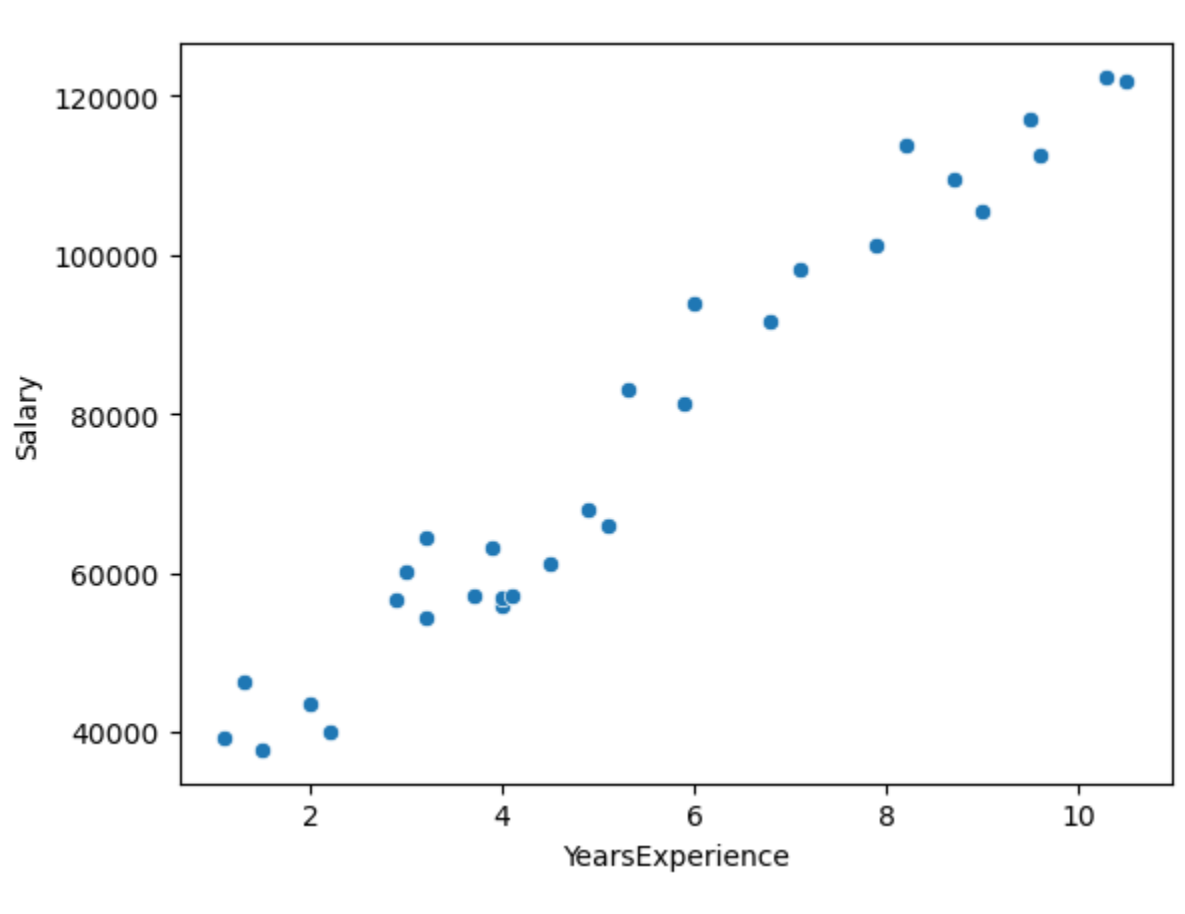
	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0

In [53]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   YearsExperience  30 non-null     float64
 1   Salary           30 non-null     float64
dtypes: float64(2)
memory usage: 612.0 bytes
```

In [54]:
```
sns.scatterplot(x="YearsExperience",y="Salary",data=df1)
```

Out[54]:
```
<Axes: xlabel='YearsExperience', ylabel='Salary'>
```



In [55]:
```
X=df1.drop("Salary",axis=1)
y=df1["Salary"]
```

In [56]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

In [57]:
```
from sklearn.linear_model import LinearRegression
```

In [58]:
```
linear=LinearRegression()
```

In [59]:
```
linear.fit(X_train,y_train)
```

Out[59]:
```
▾ LinearRegression
LinearRegression()
```

In [60]:
```
y_pred=linear.predict(X_test)
```

In [61]:
```
from sklearn.metrics import r2_score
```

In [62]:
```
print(f"r2_Score:{r2_score(y_pred,y_test)}")
```

```
r2_Score:0.9539713548476306
```

4. Build a simple logistic regression model on the 'customer_churn' dataframe, where the the dependent variable is 'Churn' & the independent variable is 'TechSupport'. Store the result in 'log_mod_1'

a. Have a glance at the summary of the model built

b. Predict the result when the value of 'TechSupport' is 'Yes'

c. Predict the result when the value of 'TechSupport' is 'No'

d. Predict the result when the value of 'TechSupport' is 'No internet service'.

In [63]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [64]:
```
df=pd.read_csv("customer_churn.csv")
```

In [65]:
```
df.head()
```

Out[65]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	Paym
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	No	No	No	Month-to-month	Yes	Elect
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	No	No	No	One year	No	M
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	No	No	No	Month-to-month	Yes	M
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	Yes	No	No	One year	No	Ba
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	No	No	No	Month-to-month	Yes	Elect

5 rows × 21 columns

In [66]:
```
df_tech=df[["TechSupport","Churn"]]
```

In [67]:
```
df_tech
```

Out[67]:

	TechSupport	Churn
0	No	No
1	No	No
2	No	Yes
3	Yes	No
4	No	Yes
...	...	...
7038	Yes	No
7039	No	No
7040	No	No
7041	No	Yes
7042	Yes	No

7043 rows × 2 columns

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

Out[68]:

	TechSupport	Churn
0	No	No
1	No	No
2	No	Yes
3	Yes	No
4	No	Yes

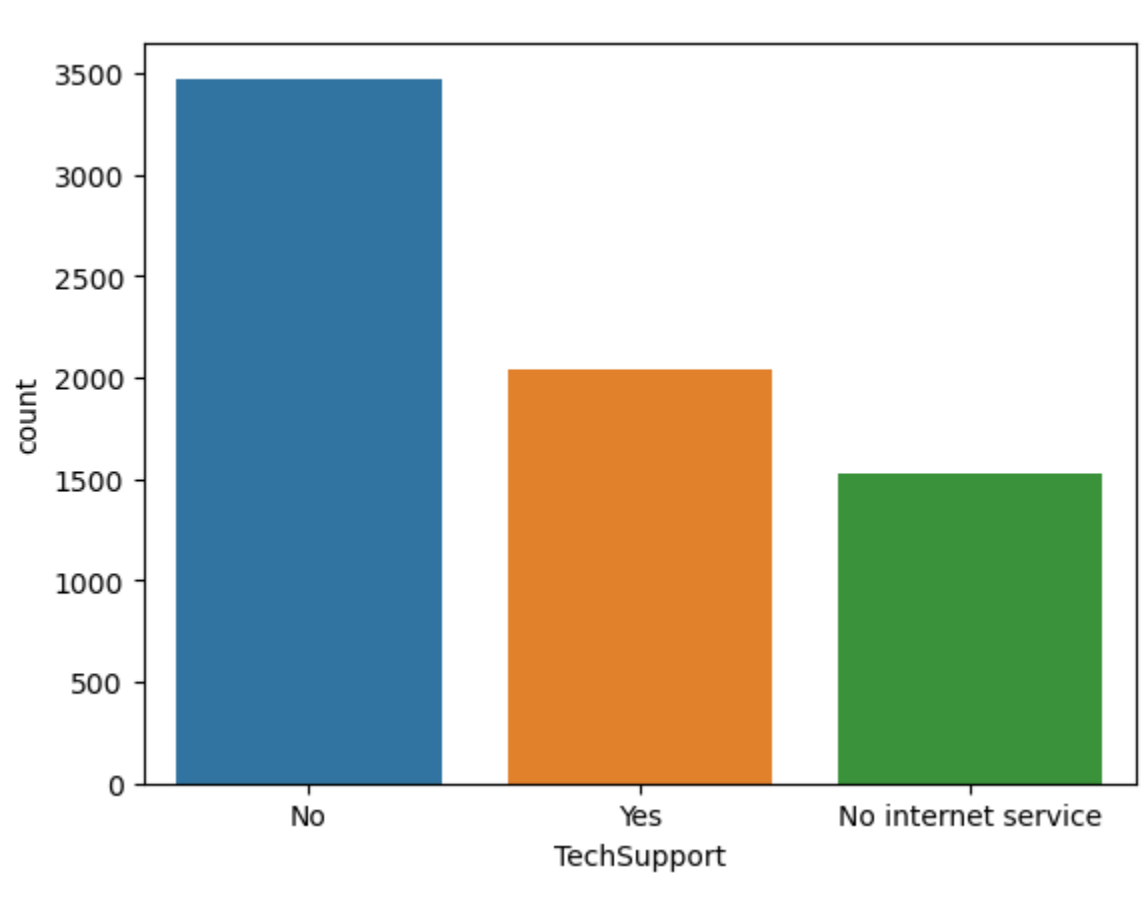
In [69]:
```
df_tech.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 2 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   TechSupport  7043 non-null   object
 1   Churn        7043 non-null   object
dtypes: object(2)
memory usage: 110.2+ KB
```

In [70]:
```
columns=df_tech.columns
```

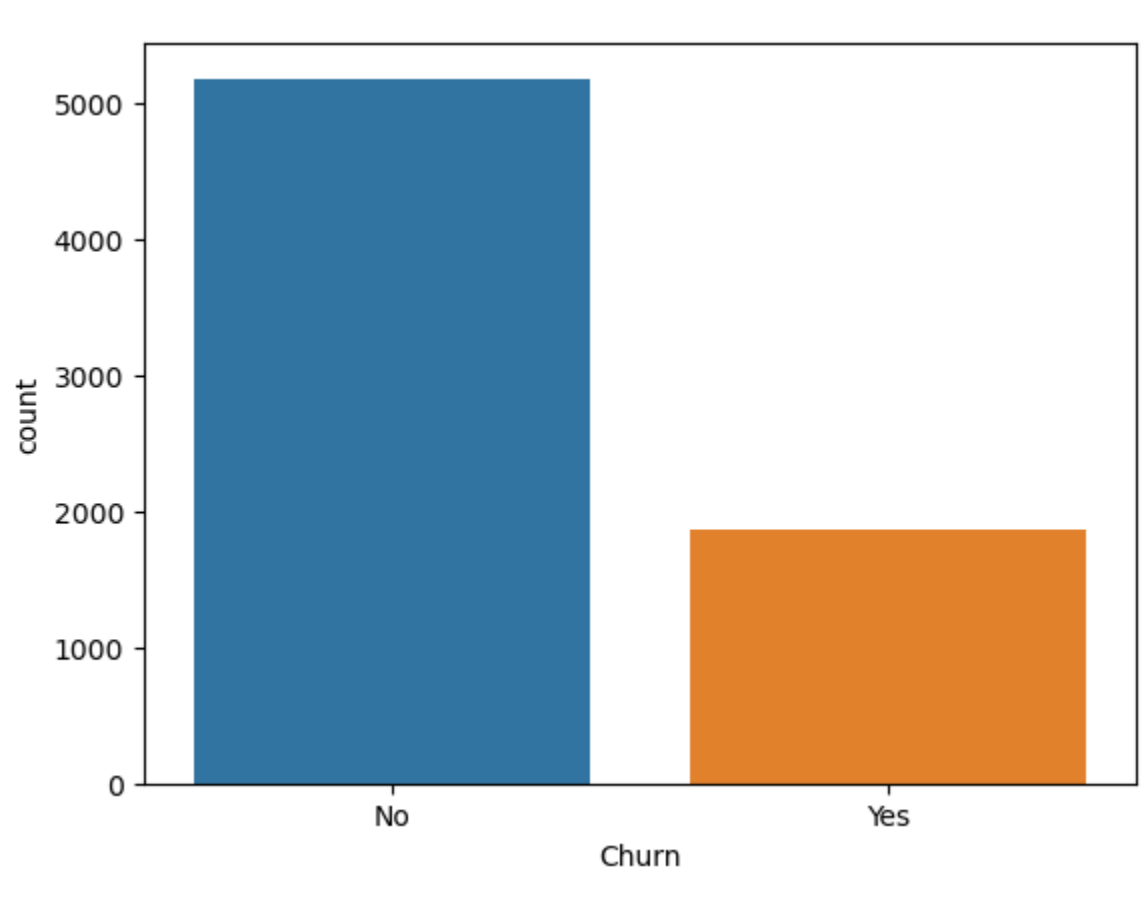
In [71]:
```
sns.countplot(x=df["TechSupport"])
```

Out[71]:
```
<Axes: xlabel='TechSupport', ylabel='count'>
```



In [72]:
```
sns.countplot(x=df["Churn"])
```

Out[72]:
```
<Axes: xlabel='Churn', ylabel='count'>
```



In [73]:
```
from sklearn.preprocessing import LabelEncoder
```

In [74]:
```
le=LabelEncoder()
```

In [75]:
```
df_tech["TechSupport"]=le.fit_transform(df_tech["TechSupport"])
```

In [76]:
```
df_tech["Churn"]=le.fit_transform(df["Churn"])
```

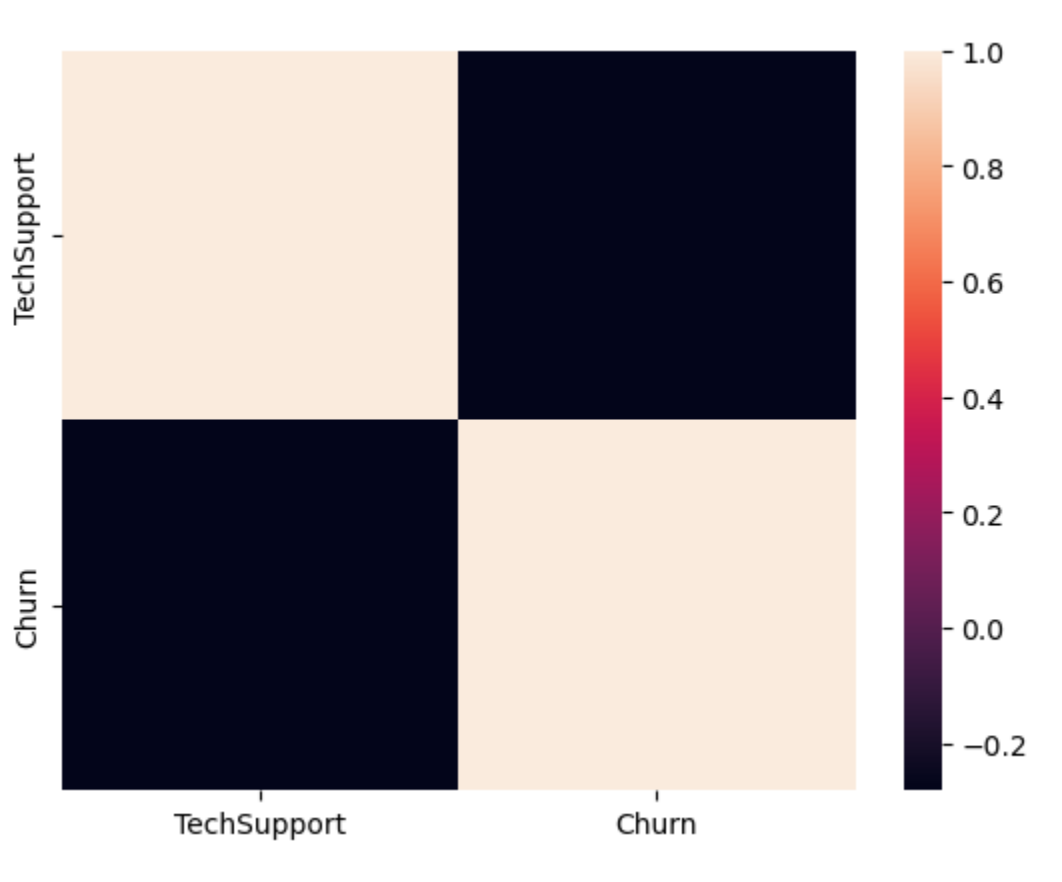
In [77]:
```
df_tech.head()
```

Out[77]:

	TechSupport	Churn
0	0	0
1	0	0
2	0	1
3	2	0
4	0	1

In [78]:
```
sns.heatmap(df_tech.corr())
```

Out[78]:
```
<Axes: >
```



In [79]:
```
X=df_tech.drop("Churn",axis=1)
y=df_tech["TechSupport"]
```

In [80]:
```
from sklearn.model_selection import train_test_split
```

In [81]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

In [82]:
```
from sklearn.linear_model import LogisticRegression
```

In [83]:
```
lr=LogisticRegression()
```

In [84]:
```
lr.fit(X_train,y_train)
```

Out[84]:
```
▾ LogisticRegression
LogisticRegression()
```

In [85]:
```
y_pred=lr.predict(X_test)
```

In [86]:
```
from sklearn.metrics import confusion_matrix,accuracy_score,f1_score
```

In [87]:
```
cm=confusion_matrix(y_test,y_pred)
```

In [88]:
```
cm
```

Out[88]:
```
array([[1162,    0,    0],
       [   0,  482,    0],
       [   0,    0,  681]], dtype=int64)
```

In [89]:
```
print(f"result when the value of ‘TechSupport’ is ‘Yes’:{cm[0,0]}")
print(f"result when the value of ‘TechSupport’ is ‘No’:{cm[1,1]}")
print(f"result when the value of ‘TechSupport’ is ‘No_TechSupport’:{cm[2,2]}")
```

```
result when the value of ‘TechSupport’ is ‘Yes’:1162
result when the value of ‘TechSupport’ is ‘No’:482
result when the value of ‘TechSupport’ is ‘No_TechSupport’:681
```

In [90]:
```
print(f"Accuracy:{accuracy_score(y_test,y_pred)}")
```

```
Accuracy:1.0
```

5. Build a simple logistic regression model on the 'customer_churn' dataframe, where the the dependent variable is 'Dependents' & the independent variable is 'tenure'. Store the result in 'log_mod_2'

a. Have a glance at the summary of the model built

b. Predict the result when the value of 'tenure' is 10

c. Predict the result when the value of 'tenure' is 50

d. Predict the result when the value of 'tenure' is 70

In [91]:
```
df_tenure=df[["tenure","Dependents"]]
```

In [92]:
```
df_tenure.head()
```

Out[92]:

	tenure	Dependents
0	1	No
1	34	No
2	2	No
3	45	No
4	2	No

In [93]:
```
df_tenure.nunique()
```

Out[93]:
```
tenure        73
Dependents     2
dtype: int64
```

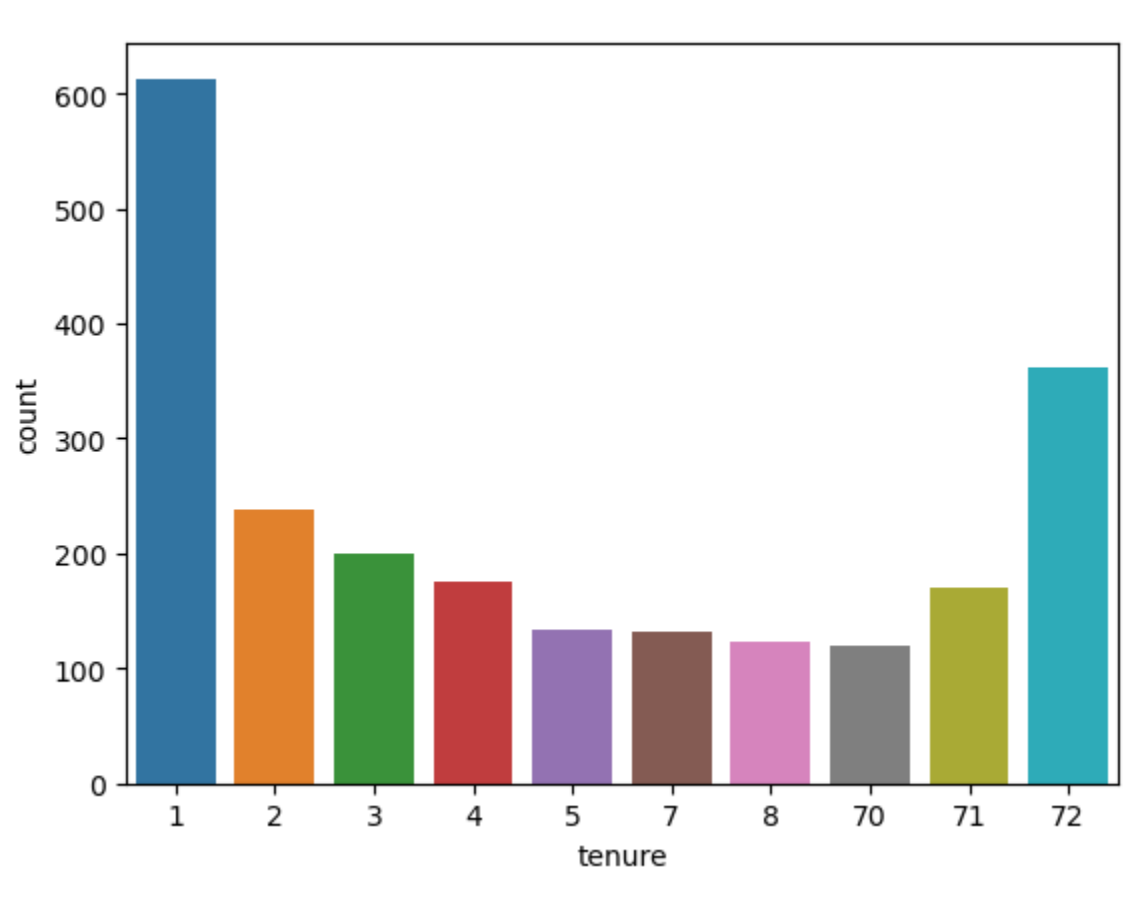
In [94]:
```
count=df_tenure["tenure"].value_counts().reset_index().sort_values("count",ascending=False).head(10)
count
```

Out[94]:

	tenure	count
0	1	613
1	72	362
2	2	238
3	3	200
4	4	176
5	71	170
6	5	133
7	7	131
8	8	123
9	70	119

In [95]:
```
sns.barplot(x="tenure",y="count",data=count)
```

Out[95]:
```
<Axes: xlabel='tenure', ylabel='count'>
```



In [96]:
```
df_tenure["Dependents"]=le.fit_transform(df_tenure["Dependents"])
```

In [97]:
```
df_tenure.head()
```

Out[97]:

	tenure	Dependents
0	1	0
1	34	0
2	2	0
3	45	0
4	2	0

In [98]:
```
X=df_tenure.drop("Dependents",axis=1)
y=df_tenure["Dependents"]
```

In [99]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

In [100…
```
from sklearn.linear_model import LogisticRegression
```

In [101…
```
log_model2=LogisticRegression()
```

In [102…
```
log_model2.fit(X_train,y_train)
```

Out[102]:
```
▾ LogisticRegression
LogisticRegression()
```

In [103…
```
y_pred=log_model2.predict(X_test)
```

In [104…
```
from sklearn.metrics import confusion_matrix,accuracy_score
```

In [105…
```
confusion_matrix(y_test,y_pred)
```

Out[105]:
```
array([[1650,    0],
       [ 675,    0]], dtype=int64)
```

In [106…
```
accuracy_score(y_test,y_pred)
```

Out[106]:
```
0.7096774193548387
```

1.Predict the result when the value of 'tenure' is 10

In [107…
```
df_tenure10=df_tenure[df_tenure["tenure"]==10]
```

In [108…
```
X=df_tenure10.drop("Dependents",axis=1)
y=df_tenure10["Dependents"]
```

In [109…
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

In [110…
```
from sklearn.linear_model import LogisticRegression
```

In [111…
```
log_model2=LogisticRegression()
```

In [112…
```
log_model2.fit(X_train,y_train)
```

Out[112]:
```
▾ LogisticRegression
LogisticRegression()
```

In [113…
```
y_pred=log_model2.predict(X_test)
```

In [114…
```
confusion_matrix(y_test,y_pred)
```

Out[114]:
```
array([[31,  0],
       [ 8,  0]], dtype=int64)
```

In [115…
```
ac=accuracy_score(y_test,y_pred)
```

In [116…
```
print(f"accuracy_score for tenure=10:{ac}")
```

```
accuracy_score for tenure=10:0.7948717948717948
```

2.Predict the result when the value of 'tenure' is 50

In [117…
```
df_tenure50=df_tenure[df_tenure["tenure"]==50]
```

In [118…
```
X=df_tenure50.drop("Dependents",axis=1)
y=df_tenure50["Dependents"]
```

In [119…
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

In [120…
```
from sklearn.linear_model import LogisticRegression
```

In [121…
```
log_model2=LogisticRegression()
```

In [122…
```
log_model2.fit(X_train,y_train)
```

Out[122]:
```
▾ LogisticRegression
LogisticRegression()
```

In [123…
```
y_pred=log_model2.predict(X_test)
```

In [124…
```
confusion_matrix(y_test,y_pred)
```

Out[124]:
```
array([[16,  0],
       [ 7,  0]], dtype=int64)
```

In [125…
```
ac=accuracy_score(y_test,y_pred)
```

In [126…
```
print(f"accuracy_score for tenure is equal to 50:{ac}")
```

```
accuracy_score for tenure is equal to 50:0.6956521739130435
```

3.Predict the result when the value of 'tenure' is 70

In [127…
```
df_tenure70=df_tenure[df_tenure["tenure"]==70]
```

In [128…
```
X=df_tenure70.drop("Dependents",axis=1)
y=df_tenure70["Dependents"]
```

In [129…
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

In [130…
```
from sklearn.linear_model import LogisticRegression
```

In [131…
```
log_model2=LogisticRegression()
```

In [132…
```
log_model2.fit(X_train,y_train)
```

Out[132]:
```
▾ LogisticRegression
LogisticRegression()
```

In [133…
```
y_pred=log_model2.predict(X_test)
```

In [134…
```
confusion_matrix(y_test,y_pred)
```

Out[134]:
```
array([[27,  0],
       [13,  0]], dtype=int64)
```

In [135…
```
ac=accuracy_score(y_test,y_pred)
```

In [136…
```
print(f"accuracy_score for tenure is equal to 70::{ac}")
```

```
accuracy_score for tenure is equal to 70::0.675
```