In [1]:
```
import numpy as np
import seaborn as sns
import re
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
```

```
C:\Python310\lib\site-packages\scipy\__init__.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is requir
ed for this version of SciPy (detected version 1.24.1
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

In [2]:
```
data = pd.read_csv("Ecommerce Purchases")
data.head(3)
```

Out[2]:

Lot	AM or PM	Browser Info	Company	Credit Card	CC Exp Date	CC Security Code	CC Provider	Email	Job	IP Address	Langua
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	149.146.147.205	
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	15.160.41.51	
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	132.207.160.22	

In [3]:
```
#Q.2
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Address           10000 non-null  object 
 1   Lot               10000 non-null  object 
 2   AM or PM          10000 non-null  object 
 3   Browser Info      10000 non-null  object 
 4   Company           10000 non-null  object 
 5   Credit Card       10000 non-null  int64  
 6   CC Exp Date       10000 non-null  object 
 7   CC Security Code  10000 non-null  int64  
 8   CC Provider       10000 non-null  object 
 9   Email             10000 non-null  object 
 10  Job               10000 non-null  object 
 11  IP Address        10000 non-null  object 
 12  Language          10000 non-null  object 
 13  Purchase Price    10000 non-null  float64
dtypes: float64(1), int64(2), object(11)
memory usage: 1.1+ MB
```

In [4]:
```
x = data.describe()
x
```

Out[4]:

	Credit Card	CC Security Code	Purchase Price
count	1.000000e+04	10000.000000	10000.000000
mean	2.341374e+15	907.217800	50.347302
std	2.256103e+15	1589.693035	29.015836
min	6.040186e+10	0.000000	0.000000
25%	3.056322e+13	280.000000	25.150000
50%	8.699942e+14	548.000000	50.505000
75%	4.492298e+15	816.000000	75.770000
max	6.012000e+15	9993.000000	99.990000

In [5]:
```
#Q.3
x.loc["mean","Purchase Price"]
```

Out[5]:
```
50.347302
```

In [6]:
```
#Q.4
x.loc[["max","min"],"Purchase Price"]
```

Out[6]:
```
max    99.99
min     0.00
Name: Purchase Price, dtype: float64
```

In [7]:
```
#Q.5
(data["Language"]=="en").value_counts()[True]
```

Out[7]:
```
1098
```

In [8]:
```
(data["Job"]=="Lawyer").value_counts()[True]
```

Out[8]:
```
30
```

In [9]:
```
#Q.6
data.iloc[:,2].value_counts()
```

Out[9]:
```
PM    5068
AM    4932
Name: AM or PM, dtype: int64
```

In [10]:
```
#Q.7
data["Job"].value_counts().head(5)
```

Out[10]:
```
Interior and spatial designer    31
Lawyer                           30
Social researcher                28
Purchasing manager               27
Designer, jewellery              27
Name: Job, dtype: int64
```

In [11]:
```
#Q.8
data.loc[data["Lot"]=="90 WT","Purchase Price"]
```

Out[11]:
```
513    75.1
Name: Purchase Price, dtype: float64
```

In [12]:
```
#Q.10
data.loc[data["Credit Card"]==4926535242672853,"Email"]
```

Out[12]:
```
1234    bondellen@williams-garza.com
Name: Email, dtype: object
```

In [13]:
```
#Q.11
data[(data["CC Provider"]=="American Express")&(data["Purchase Price"]>95)].shape[0]
```

Out[13]:
```
39
```

In [27]:
```
#Q.12
data["Year"]=data["CC Exp Date"].apply(lambda x:int(re.sub(x,x[-2:],x)))
(data["Year"]==25).value_counts()[True]
```

Out[27]:
```
1033
```

In [15]:
```
#Q.13
a = data.Email
a.apply(lambda x:re.findall("\@[a-z]+",x)[0][1:]).value_counts().head(5)
```

Out[15]:
```
hotmail    1638
yahoo      1616
gmail      1605
smith       114
brown        82
Name: Email, dtype: int64
```

In [ ]:
```

```

In [20]:

```
import pandas as pd
import numpy as np
```

In [21]:

```
df=pd.read_csv('Customers.csv')
```

In [22]:

```
df.head()
```

Out[22]:

	CustomerID	Gender	Age	Annual Income ($)	Spending Score (1-100)	Profession	Work Experience	Family Size
0	1	Male	19	15000	39	Healthcare	1	4
1	2	Male	21	35000	81	Engineer	3	3
2	3	Female	20	86000	6	Engineer	1	1
3	4	Female	23	59000	77	Lawyer	0	2
4	5	Female	31	38000	40	Entertainment	2	6

In [23]:

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

Out[23]:

```
CustomerID                0
Gender                    0
Age                       0
Annual Income ($)         0
Spending Score (1-100)    0
Profession               35
Work Experience           0
Family Size               0
dtype: int64
```

In [24]:

```
df.shape
```

Out[24]:

```
(2000, 8)
```

In [25]:

```
df.dtypes
```

Out[25]:

```
CustomerID                int64
Gender                   object
Age                       int64
Annual Income ($)         int64
Spending Score (1-100)    int64
Profession               object
Work Experience           int64
Family Size               int64
dtype: object
```

In [26]:

```
df.describe()
```

Out[26]:

	CustomerID	Age	Annual Income ($)	Spending Score (1-100)	Work Experience	Family Size
count	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000
mean	1000.500000	48.960000	110731.821500	50.962500	4.102500	3.768500
std	577.494589	28.429747	45739.536688	27.934661	3.922204	1.970749
min	1.000000	0.000000	0.000000	0.000000	0.000000	1.000000
25%	500.750000	25.000000	74572.000000	28.000000	1.000000	2.000000
50%	1000.500000	48.000000	110045.000000	50.000000	3.000000	4.000000
75%	1500.250000	73.000000	149092.750000	75.000000	7.000000	5.000000
max	2000.000000	99.000000	189974.000000	100.000000	17.000000	9.000000

In [27]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 8 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   CustomerID              2000 non-null   int64 
 1   Gender                  2000 non-null   object
 2   Age                     2000 non-null   int64 
 3   Annual Income ($)       2000 non-null   int64 
 4   Spending Score (1-100)  2000 non-null   int64 
 5   Profession              1965 non-null   object
 6   Work Experience         2000 non-null   int64 
 7   Family Size             2000 non-null   int64 
dtypes: int64(6), object(2)
memory usage: 125.1+ KB
```

In [28]:

```
row='Rows',len(df)
row
```

Out[28]:

```
('Rows', 2000)
```

In [29]:

```
col=df.shape[1]
col
```

Out[29]:

```
8
```

In [37]:

```
df['Gender'].replace(['Female','Male'],[0,1], inplace = True)
df.head()
```

Out[37]:

	CustomerID	Gender	Age	Annual Income ($)	Spending Score (1-100)	Profession	Work Experience	Family Size
0	1	1	19	15000	39	Healthcare	1	4
1	2	1	21	35000	81	Engineer	3	3
2	3	0	20	86000	6	Engineer	1	1
3	4	0	23	59000	77	Lawyer	0	2
4	5	0	31	38000	40	Entertainment	2	6

In [38]:

```
df.dtypes
```

Out[38]:

```
CustomerID                int64
Gender                    int64
Age                       int64
Annual Income ($)         int64
Spending Score (1-100)    int64
Profession               object
Work Experience           int64
Family Size               int64
dtype: object
```

Importing Libraries

In [297]:
```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
from scipy.stats import norm
import statistics
```

In [298]:
```
df = pd.read_csv('StudentsPerformance.csv')
```

Description of the Data

In [299]:
```
df.head(3)
```

Out[299]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score
0	female	group B	bachelor's degree	standard	none	72.0
1	female	group C	some college	standard	completed	69.0
2	female	group B	master's degree	standard	none	90.0

In [300]:
```
df.shape
```

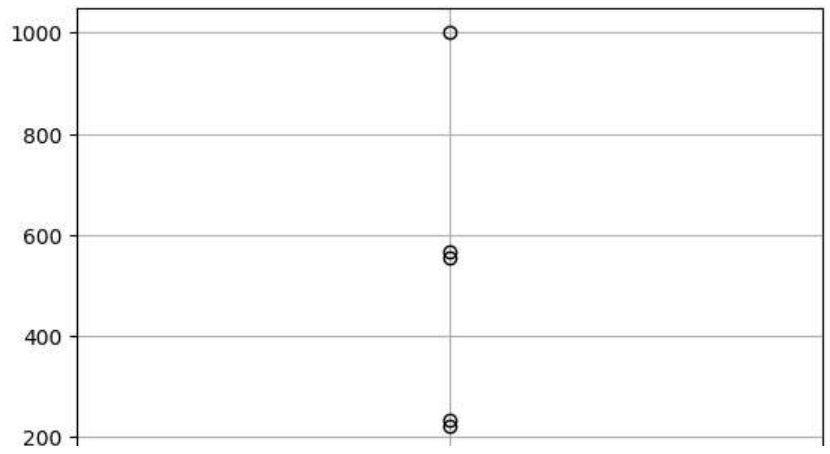
Out[300]:
```
(1000, 6)
```

In [301]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 6 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   gender                       1000 non-null   object 
 1   race/ethnicity               1000 non-null   object 
 2   parental level of education  1000 non-null   object 
 3   lunch                        1000 non-null   object 
 4   test preparation course      1000 non-null   object 
 5   math score                   993 non-null    float64
dtypes: float64(1), object(5)
memory usage: 47.0+ KB
```

In [302]:
```
df.boxplot()
```

Out[302]:
```
<AxesSubplot: >
```



Removing Null Values

In [303]:
```
df.isnull().sum()
```

Out[303]:
```
gender                         0
race/ethnicity                 0
parental level of education    0
lunch                          0
test preparation course        0
math score                     7
dtype: int64
```

In [304]:
```
df.mean(axis=0)
```

Out[304]:
```
math score    68.299094
dtype: float64
```

In [305]:
```
df["math score"].fillna(68.299094, inplace = True)
```

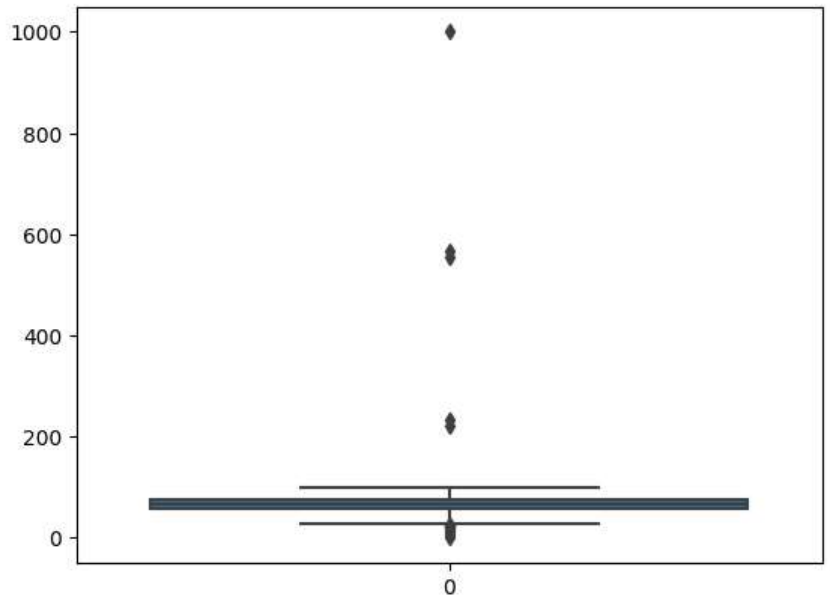
In [306]:
```
df.isnull().sum()
```

Out[306]:
```
gender                         0
race/ethnicity                 0
parental level of education    0
lunch                          0
test preparation course        0
math score                     0
dtype: int64
```

Removing Outliers from the Data

In [307]:
```
sns.boxplot(data=df["math score"])
```

Out[307]:
```
<AxesSubplot: >
```



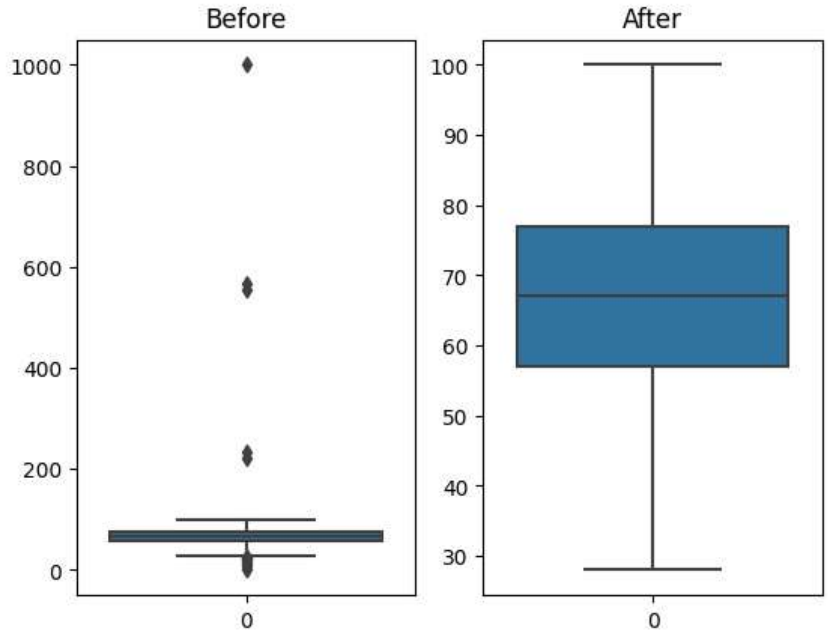
In [308]:
```
Q1 = df['math score'].quantile(0.25)
Q3 = df['math score'].quantile(0.75)
```

In [309]:
```
#Interquartile range
iqr = Q3math - Q1
minm= Q1 - (1.5*iqr)
maxm = Q3 + (1.5*iqr)
print("This is the IQR:",iqr)
print("This is the minimum", minm)
print("This is the maximum", maxm)
```

```
This is the IQR: 20.0
This is the minimum 27.0
This is the maximum 107.0
```

In [310]:
```
df2=df[(df['math score']>minm) & (df['math score']<maxm)]
```

In [311]:
```
fig , axes = plt.subplots(1,2)
sns.boxplot(df['math score'],ax=axes[0])
axes[0].title.set_text('Before')
sns.boxplot(df2['math score'],ax=axes[1])
axes[1].title.set_text('After')
plt.show()
```

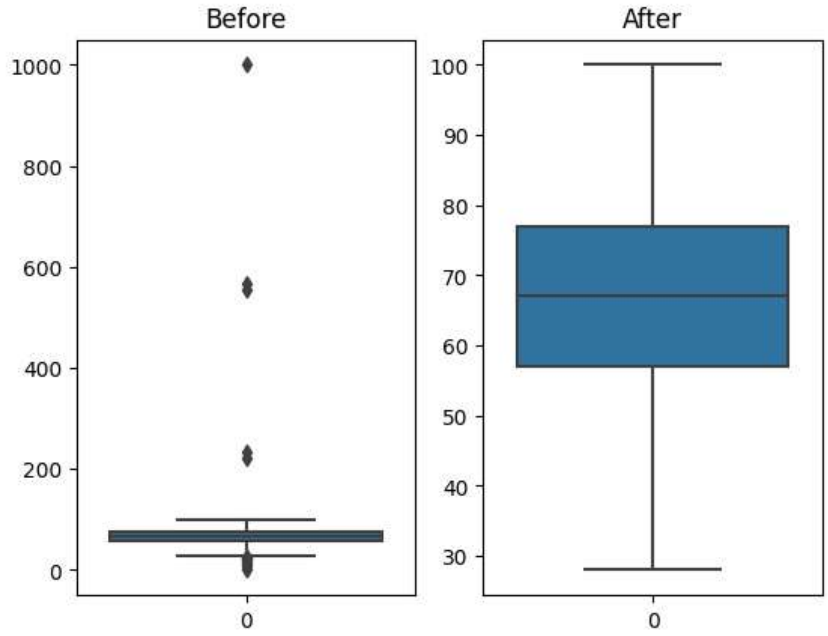


Visualization

In [312]:
```
print("Before Shape:",df.shape)
print("After Shape:",df2.shape)
```

```
Before Shape: (1000, 6)
After Shape: (984, 6)
```

In [313]:
```
fig , axes = plt.subplots(1,2)
sns.boxplot(df['math score'],ax=axes[0])
axes[0].title.set_text('Before')
sns.boxplot(df2['math score'],ax=axes[1])
axes[1].title.set_text('After')
plt.show()
```

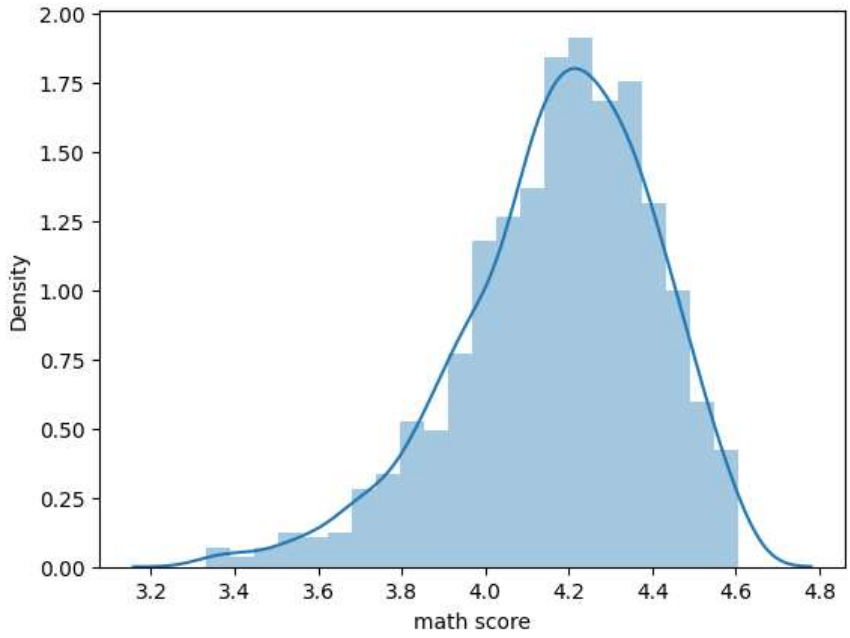


In [314]:
```
print(df.skew(numeric_only=True))
print(df2.skew(numeric_only=True))
```

```
math score    15.889874
dtype: float64
math score   -0.051639
dtype: float64
```

In [315]:
```
t = np.log(df2['math score'])
sns.distplot(t)
```

Out[315]: <AxesSubplot: xlabel='math score', ylabel='Density'>



In [ ]:

In [95]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
from sklearn.datasets import load_iris
%matplotlib inline
```

In [96]:

```
df = pd.read_csv("loan_data_set.csv")
```

In [97]:

```
df.head()
```

Out[97]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coa
0	LP001002	Male	No	0	Graduate	No	5849	
1	LP001003	Male	Yes	1	Graduate	No	4583	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	
4	LP001008	Male	No	0	Graduate	No	6000	

In [98]:

```
df.shape
```

Out[98]:

```
(614, 13)
```

In [99]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Loan_ID            614 non-null    object 
 1   Gender             601 non-null    object 
 2   Married            611 non-null    object 
 3   Dependents         599 non-null    object 
 4   Education          614 non-null    object 
 5   Self_Employed      582 non-null    object 
 6   ApplicantIncome    614 non-null    int64  
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object 
 12  Loan_Status        614 non-null    object 
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

In [100]:

```
df.describe()
```

Out[100]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

In [101]:

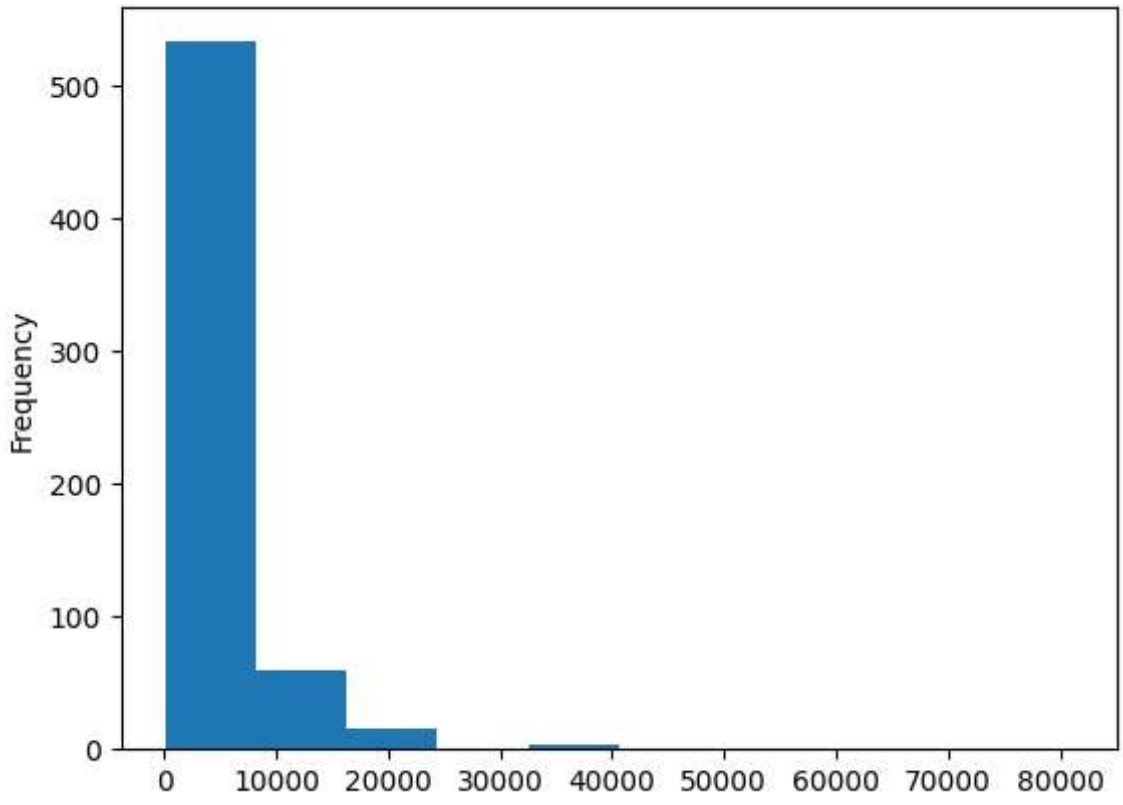
```
df.isna().sum()
```

Out[101]:

```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

In [102]:

```
df["ApplicantIncome"].plot(kind="hist")
plt.show()
```

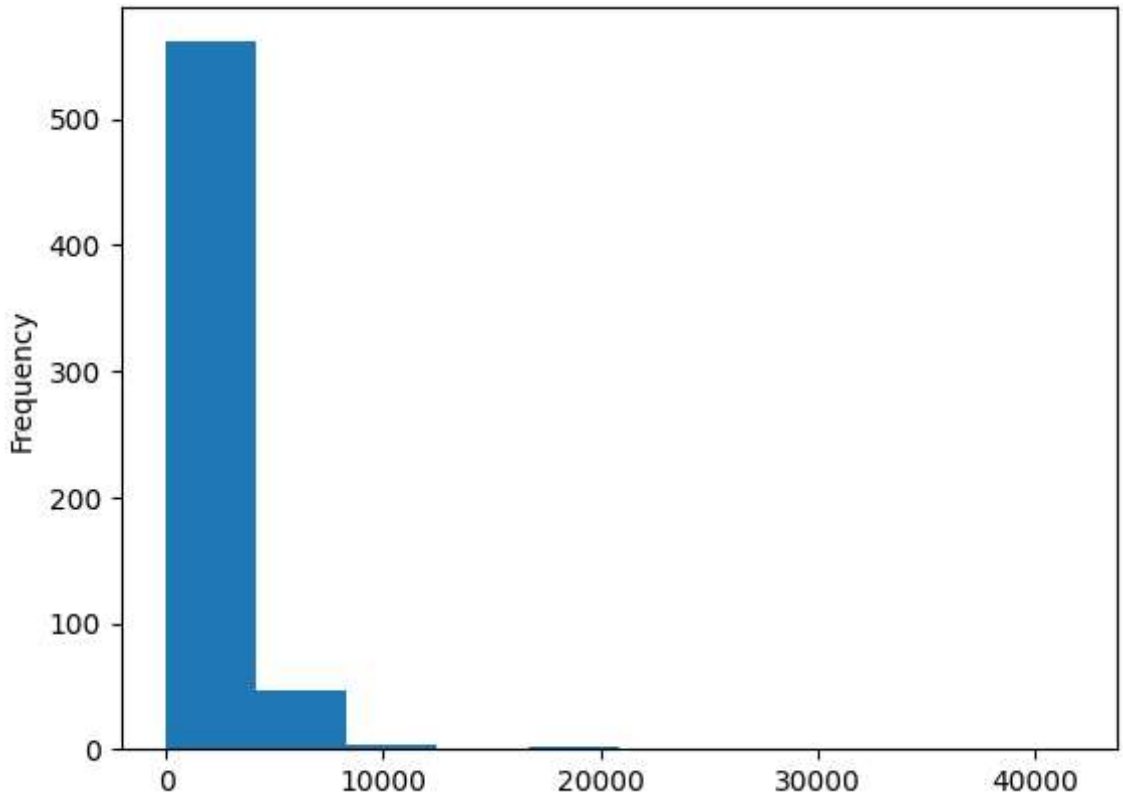


In [103]:

```
df["ApplicantIncome"].fillna(df["ApplicantIncome"].mean(), inplace=True)
```

In [104]:

```
df["CoapplicantIncome"].plot(kind="hist")
plt.show()
```

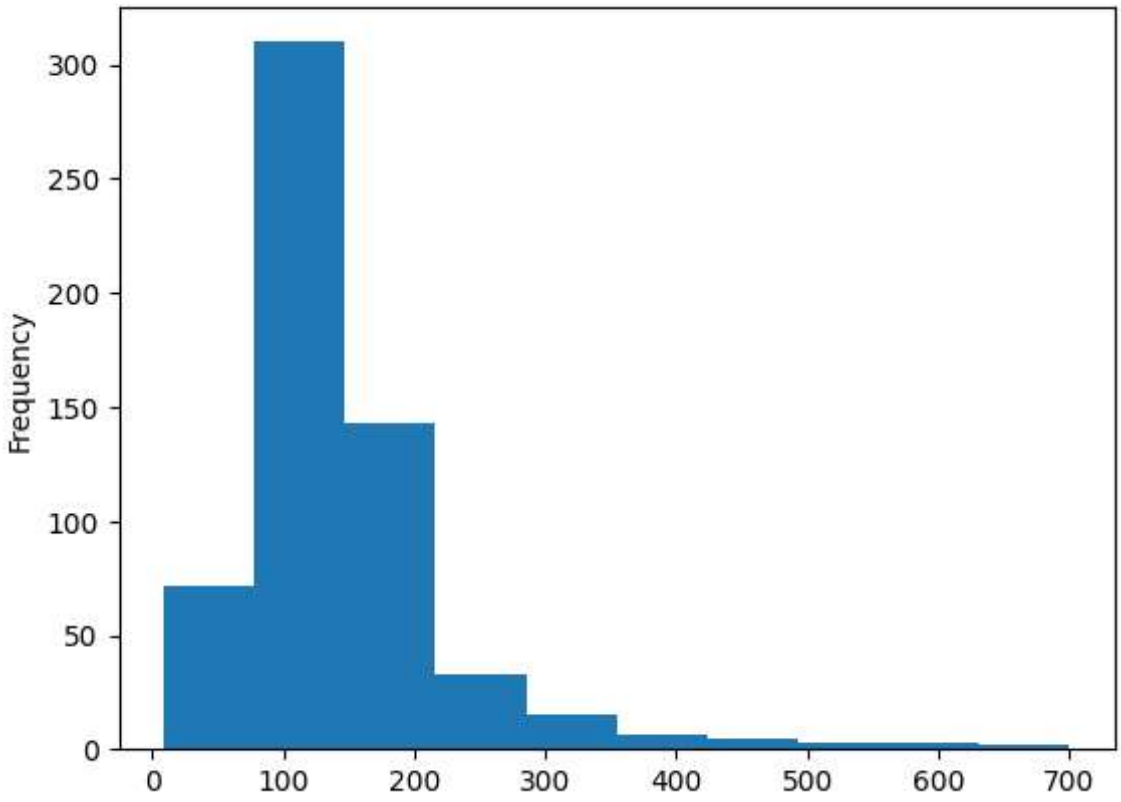


In [105]:

```
df["CoapplicantIncome"].fillna(df["CoapplicantIncome"].mean(), inplace=True)
```

In [106]:

```
df["LoanAmount"].plot(kind="hist")
plt.show()
```

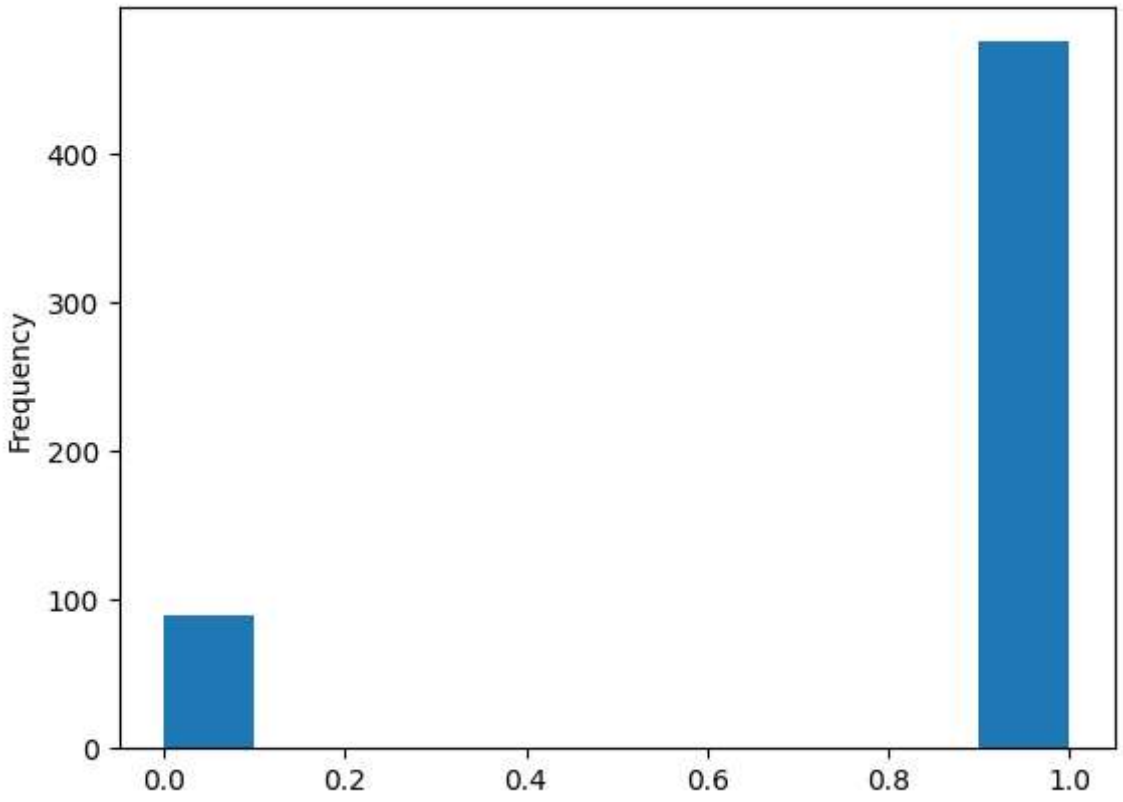


In [107]:

```
df["LoanAmount"].fillna(df["LoanAmount"].mean(), inplace=True)
```

In [108]:

```
df["Credit_History"].plot(kind="hist")
plt.show()
```



In [109]:

```
df["Credit_History"].fillna(np.random.randint(0,2), inplace=True)
```

In [110]:

```
grouped_df = df[["ApplicantIncome", "CoapplicantIncome", "LoanAmount", "Credit_History"]
```

In [111]:

```
mean = grouped_df.mean()
mean
```

Out[111]:

Loan_Status	ApplicantIncome	CoapplicantIncome	LoanAmount	Credit_History
N	5446.078125	1877.807292	150.945488	0.572917
Y	5384.068720	1504.516398	144.349606	0.983412

In [112]:

```
median = grouped_df.median()
median
```

Out[112]:

Loan_Status	ApplicantIncome	CoapplicantIncome	LoanAmount	Credit_History
N	3833.5	268.0	133.5	1.0
Y	3812.5	1239.5	128.0	1.0

In [113]:

```
min = grouped_df.min()
min
```

Out[113]:

Loan_Status	ApplicantIncome	CoapplicantIncome	LoanAmount	Credit_History
N	150	0.0	9.0	0.0
Y	210	0.0	17.0	0.0

In [114]:

```
max = grouped_df.max()
max
```

Out[114]:

Loan_Status	ApplicantIncome	CoapplicantIncome	LoanAmount	Credit_History
N	81000	41667.0	570.0	1.0
Y	63337	20000.0	700.0	1.0

In [115]:

```
std = grouped_df.std()
std
```

Out[115]:

Loan_Status	ApplicantIncome	CoapplicantIncome	LoanAmount	Credit_History
N	6819.558528	4384.060103	83.361163	0.495948
Y	5765.441615	1924.754855	84.361109	0.127872

Iris Data

In [116]:

```
df = pd.read_csv('Iris.csv')
df.head()
```

Out[116]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [117]:

```
tdf = df.groupby(by='Species')
tdf.first()
```

Out[117]:

Species	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
Iris-setosa	1	5.1	3.5	1.4	0.2
Iris-versicolor	51	7.0	3.2	4.7	1.4
Iris-virginica	101	6.3	3.3	6.0	2.5

In [118]:

```
tdf.get_group('Iris-setosa').describe()
```

Out[118]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	50.00000	50.00000	50.000000	50.000000	50.00000
mean	25.50000	5.00600	3.418000	1.464000	0.24400
std	14.57738	0.35249	0.381024	0.173511	0.10721
min	1.00000	4.30000	2.300000	1.000000	0.10000
25%	13.25000	4.80000	3.125000	1.400000	0.20000
50%	25.50000	5.00000	3.400000	1.500000	0.20000
75%	37.75000	5.20000	3.675000	1.575000	0.30000
max	50.00000	5.80000	4.400000	1.900000	0.60000

In [119]:

```
tdf.get_group('Iris-versicolor').describe()
```

Out[119]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	50.00000	50.000000	50.000000	50.000000	50.000000
mean	75.50000	5.936000	2.770000	4.260000	1.326000
std	14.57738	0.516171	0.313798	0.469911	0.197753
min	51.00000	4.900000	2.000000	3.000000	1.000000
25%	63.25000	5.600000	2.525000	4.000000	1.200000
50%	75.50000	5.900000	2.800000	4.350000	1.300000
75%	87.75000	6.300000	3.000000	4.600000	1.500000
max	100.00000	7.000000	3.400000	5.100000	1.800000

In [120]:

```
tdf.get_group('Iris-virginica').describe()
```

Out[120]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	50.00000	50.00000	50.000000	50.000000	50.00000
mean	125.50000	6.58800	2.974000	5.552000	2.02600
std	14.57738	0.63588	0.322497	0.551895	0.27465
min	101.00000	4.90000	2.200000	4.500000	1.40000
25%	113.25000	6.22500	2.800000	5.100000	1.80000
50%	125.50000	6.50000	3.000000	5.550000	2.00000
75%	137.75000	6.90000	3.175000	5.875000	2.30000
max	150.00000	7.90000	3.800000	6.900000	2.50000

In [ ]:

```

```

In [ ]:

```

```

In [145]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_boston
boston_dataset = load_boston()
from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.preprocessing import StandardScaler
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
```

In [146]:
```
boston = load_boston()
boston.keys()
```

Out[146]:
```
dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename', 'data_module'])
```

In [147]:
```
x = pd.DataFrame(boston.data, columns=boston.feature_names)
y = pd.DataFrame(boston.target, columns=['MEDV'])
```

In [148]:
```
x.shape, y.shape
```

Out[148]:
```
((506, 13), (506, 1))
```

In [149]:
```
x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 13 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    float64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    float64
 9   TAX      506 non-null    float64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
dtypes: float64(13)
memory usage: 51.5 KB
```

Basic Stats

In [150]:
```
x.describe()
```

Out[150]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	50
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154	1
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116	
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	1
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000	1
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000	1
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000	2
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	2

In [151]:
```
y.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   MEDV    506 non-null    float64
dtypes: float64(1)
memory usage: 4.1 KB
```

In [152]:
```
y.describe()
```

Out[152]:

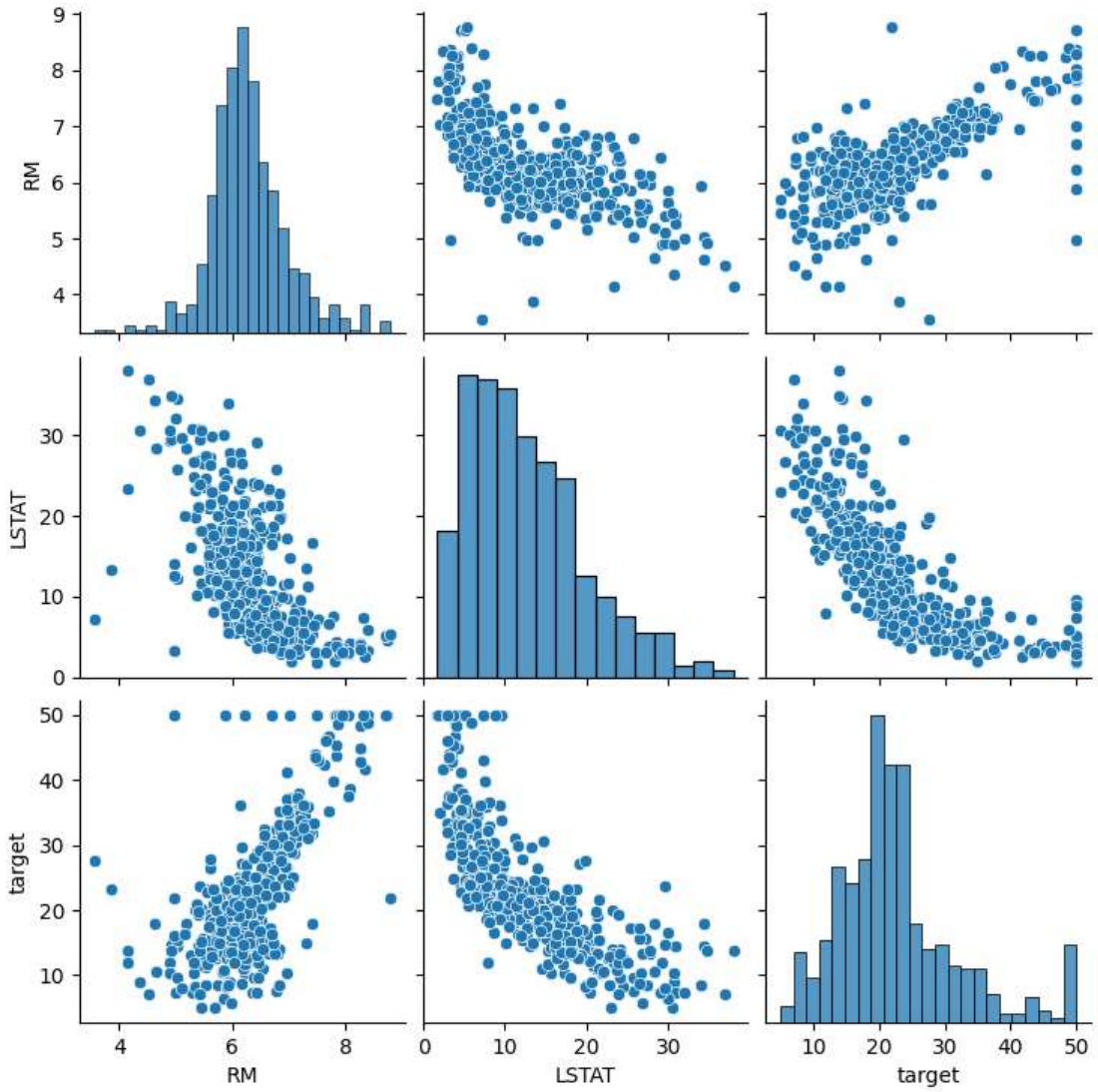
	MEDV
count	506.000000
mean	22.532806
std	9.197104
min	5.000000
25%	17.025000
50%	21.200000
75%	25.000000
max	50.000000

In [153]:
```
df = x
df["target"] = y
df.head()
```

Out[153]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	target
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2

In [154]:
```
df = df[['RM', 'LSTAT', 'target']]
sns.pairplot(df)
plt.show()
```



In [155]:
```
x = df[['RM', 'LSTAT']]
y = df['target']
```

Scale the data

In [156]:
```
scaler = StandardScaler()
x = scaler.fit_transform(x)
```

Split the data

In [157]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, shuffle=True)

x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

Out[157]:
```
((404, 2), (102, 2), (404,), (102,))
```

Linear Regression Modelling

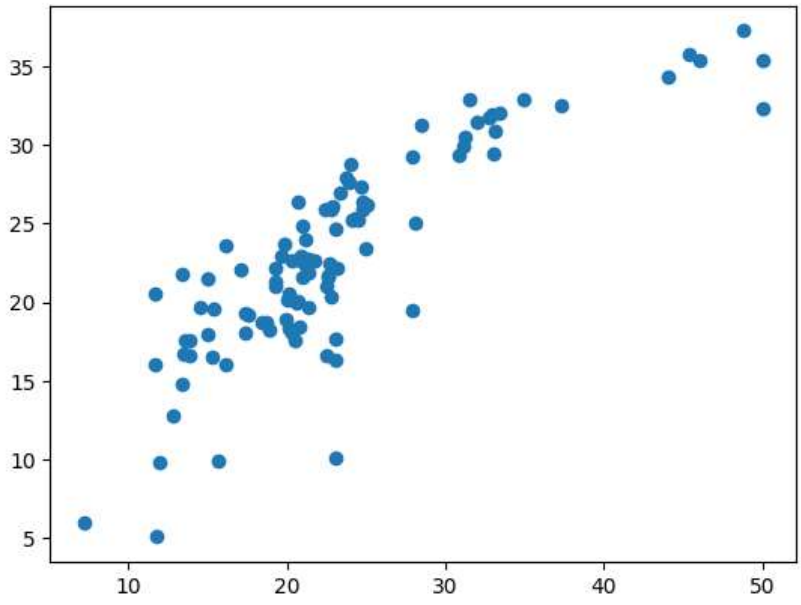
In [158]:
```
lin_model = LinearRegression(n_jobs=-1)
lin_model.fit(x_train, y_train)
```

Out[158]:
```
LinearRegression
LinearRegression(n_jobs=-1)
```

In [159]:
```
y_train_predict = lin_model.predict(x_train)
rmse = (np.sqrt(mean_squared_error(y_train, y_train_predict)))
r2 = r2_score(y_train, y_train_predict)

print("The model performance for training set")
print("--------------------------------------")
print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))
print('The Accuracy is {}'.format(r2*100))
print("\n")

# model evaluation for testing set

y_test_predict = lin_model.predict(x_test)
# root mean square error of the model
rmse = (np.sqrt(mean_squared_error(y_test, y_test_predict)))

# r-squared score of the model
r2 = r2_score(y_test, y_test_predict)

print("The model performance for testing set")
print("--------------------------------------")
print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))
print('The Accuracy is {}'.format(r2*100))
```

```
The model performance for training set
--------------------------------------
RMSE is 5.738419290690856
R2 score is 0.6257505828434102
The Accuracy is 62.575058284341026


The model performance for testing set
--------------------------------------
RMSE is 4.585299654770617
R2 score is 0.6992689116164815
The Accuracy is 69.92689116164816
```

In [160]:
```
plt.scatter(y_test, y_test_predict)
plt.show()
```



Make predictions

In [161]:
```
G=np.array([[18.35,5.701]])
d1=pd.DataFrame(G,columns=['LSTAT','RM'])
```

In [162]:
```
d1.head()
```

Out[162]:

	LSTAT	RM
0	18.35	5.701

In [163]:
```
print(lin_model.predict(d1))
```

```
[60.54204484]
```

In [ ]:

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
```

In [2]:

```
dataset = pd.read_csv('C:/Users/shrey/Desktop/New folder (2)/Social_Network_Ads.csv')
dataset.head()
```

Out[2]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

In [3]:

```
dataset = dataset.drop('User ID', axis=1)
print(dataset.isnull().sum(), '\n')
dataset.info()
```

```
Gender             0
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 4 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Gender           400 non-null    object
 1   Age              400 non-null    int64
 2   EstimatedSalary  400 non-null    int64
 3   Purchased        400 non-null    int64
dtypes: int64(3), object(1)
memory usage: 12.6+ KB
```

In [4]:

```
labelencoder = LabelEncoder()
dataset['Gender'] = labelencoder.fit_transform(dataset['Gender'])
```

In [5]:

```
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

In [6]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0
```

In [7]:

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [8]:

```
classifier = LogisticRegression(random_state=0)
classifier.fit(X_train, y_train)
```

Out[8]:

```
LogisticRegression(random_state=0)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [9]:

```
y_pred = classifier.predict(X_test)
```

In [10]:

```
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm)
```

```
Confusion Matrix:
 [[65  3]
 [ 7 25]]
```

In [11]:

```
TP = cm[0][0]
FP = cm[1][0]
TN = cm[1][1]
FN = cm[0][1]
print(f'TP {TP} FN {FN} \nFP {FP} TN {TN}')
```

```
TP 65 FN 3
FP 7 TN 25
```

In [13]:

```
accuracy = (TP + TN) / (TP + TN + FP + FN)
error_rate = (FP + FN) / (TP + TN + FP + FN)
precision = TP / (TP + FP)
recall = TP / (TP + FN)

print("Accuracy: ", accuracy)
print("Error Rate: ", error_rate)
print("Precision: ", precision)
print("Recall: ", recall)
```

```
Accuracy:  0.9
Error Rate:  0.1
Precision:  0.9027777777777778
Recall:  0.9558823529411765
```

In [ ]:

```

```

In [44]:
```
import pandas as pd
import numpy as np
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix
```

In [45]:
```
iris_df = pd.read_csv(r"C:\Users\hp\Documents\Pathu\iris.csv")
```

In [46]:
```
iris_df.head()
```

Out[46]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [47]:
```
iris_df.isnull()
```

Out[47]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False
...	...	...	...	...	...
145	False	False	False	False	False
146	False	False	False	False	False
147	False	False	False	False	False
148	False	False	False	False	False
149	False	False	False	False	False

150 rows × 5 columns

In [49]:
```
#data preprocessing
iris_df.isnull().sum()
```

Out[49]:
```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

In [50]:
```
iris_df.describe()
```

Out[50]:

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [51]:
```
iris_df.shape
```

Out[51]:
```
(150, 5)
```

In [55]:
```
# Separating features and target variable
X = iris_df.iloc[:, :-1]
y = iris_df.iloc[:, -1]
```

In [56]:
```
# Splitting the data into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state
```

In [57]:
```
# Training the Naive Bayes classifier
gnb = GaussianNB()
gnb.fit(X_train, y_train)
```

Out[57]:
```
GaussianNB()
```

In [58]:
```
# Predicting the class labels for the test set
y_pred = gnb.predict(X_test)
```

In [59]:
```
# Computing the confusion matrix
conf_mat = confusion_matrix(y_test, y_pred)
print("Confusion matrix:\n", conf_mat)

# Computing TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall
TP = conf_mat[1][1]
FP = conf_mat[0][1]
TN = conf_mat[0][0]
FN = conf_mat[1][0]

accuracy = (TP + TN) / (TP + FP + TN + FN)
error_rate = 1 - accuracy
precision = TP / (TP + FP)
recall = TP / (TP + FN)
```

```
Confusion matrix:
 [[19  0  0]
 [ 0  8  5]
 [ 0  1 12]]
```

In [60]:
```
print("TP: ", TP)
print("FP: ", FP)
print("TN: ", TN)
print("FN: ", FN)
print("Accuracy: ", accuracy)
print("Error rate: ", error_rate)
print("Precision: ", precision)
print("Recall: ", recall)
```

```
TP:  8
FP:  0
TN:  19
FN:  0
Accuracy:  1.0
Error rate:  0.0
Precision:  1.0
Recall:  1.0
```

7) Text Analytics

1. Extract Sample document and apply following document preprocessing methods: Tokenization, POS Tagging, stop words removal, Stemming and Lemmatization.
2. Create representation of documents by calculating Term Frequency and Inverse DocumentFrequency.

In [1]:

```
#importing required modules
import numpy as np
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer,WordNetLemmatizer
from collections import Counter
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
document1 = "Modi led the BJP in the 2014 general election which gave the party a majori
in the lower house of Indian parliament, the Lok Sabha, the first time for any single pa
document2 = "Following his party's victory in the 2019 general election, his administrat
revoked the special status of Jammu and Kashmir, introduced the Citizenship Amendment Ac
three controversial farm laws, which prompted widespread protests and sit-ins across the
resulting in a formal repeal of the latter."
print(document1)
```

```
Modi led the BJP in the 2014 general election which gave the party a majo
rity in the lower house of Indian parliament, the Lok Sabha, the first ti
me for any single party since 1984.
```

1. Extract Sample document and apply following document preprocessing methods: Tokenization, POS Tagging, stop words removal, Stemming and Lemmatization.

In [3]:

```
#Tokenization
tokens = nltk.word_tokenize(document1)
print("Word tokenization : ",tokens)
```

```
Word tokenization :  ['Modi', 'led', 'the', 'BJP', 'in', 'the', '2014',
'general', 'election', 'which', 'gave', 'the', 'party', 'a', 'majority',
'in', 'the', 'lower', 'house', 'of', 'Indian', 'parliament', ',', 'the',
'Lok', 'Sabha', ',', 'the', 'first', 'time', 'for', 'any', 'single', 'par
ty', 'since', '1984', '.']
```

In [4]:

```
#POS Tagging
pos_tags = nltk.pos_tag(tokens)
print(f"POS-tags : {pos_tags}",)
```

```
POS-tags : [('Modi', 'NNP'), ('led', 'VBD'), ('the', 'DT'), ('BJP', 'NN
P'), ('in', 'IN'), ('the', 'DT'), ('2014', 'CD'), ('general', 'JJ'), ('el
ection', 'NN'), ('which', 'WDT'), ('gave', 'VBD'), ('the', 'DT'), ('part
y', 'NN'), ('a', 'DT'), ('majority', 'NN'), ('in', 'IN'), ('the', 'DT'),
('lower', 'JJR'), ('house', 'NN'), ('of', 'IN'), ('Indian', 'JJ'), ('parl
iament', 'NN'), (',', ','), ('the', 'DT'), ('Lok', 'NNP'), ('Sabha', 'NN
P'), (',', ','), ('the', 'DT'), ('first', 'JJ'), ('time', 'NN'), ('for',
'IN'), ('any', 'DT'), ('single', 'JJ'), ('party', 'NN'), ('since', 'IN'),
('1984', 'CD'), ('.', '.')]
```

In [5]:

```
#stop words removal
filtered = []
for word in tokens:
    if word not in stopwords.words("english"):
        filtered.append(word)
print(f"Filtered tokens : {filtered}")
```

```
Filtered tokens : ['Modi', 'led', 'BJP', '2014', 'general', 'election',
'gave', 'party', 'majority', 'lower', 'house', 'Indian', 'parliament',
',', 'Lok', 'Sabha', ',', 'first', 'time', 'single', 'party', 'since', '1
984', '.']
```

In [6]:

```
#stemming
stemmer = PorterStemmer()
stemmed_tokens = []
for word in filtered:
    stemmed_tokens.append(stemmer.stem(word))
print(f"Stemmmed tokens : {stemmed_tokens}")
```

```
Stemmmed tokens : ['modi', 'led', 'bjp', '2014', 'gener', 'elect', 'gav
e', 'parti', 'major', 'lower', 'hous', 'indian', 'parliament', ',', 'lo
k', 'sabha', ',', 'first', 'time', 'singl', 'parti', 'sinc', '1984', '.']
```

In [7]:

```
#Lemmatization
lemma  = WordNetLemmatizer()
lemmatized_tokens = []
for word in filtered:
    lemmatized_tokens.append(lemma.lemmatize(word))
print(f"Lemmatized tokens : {lemmatized_tokens}")
```

```
Lemmatized tokens : ['Modi', 'led', 'BJP', '2014', 'general', 'election',
'gave', 'party', 'majority', 'lower', 'house', 'Indian', 'parliament',
',', 'Lok', 'Sabha', ',', 'first', 'time', 'single', 'party', 'since', '1
984', '.']
```

2.Create representation of documents by calculating Term Frequency and Inverse DocumentFrequency.

- Term frequency

Term Frequency (TF) is a metric used in natural language processing and information retrieval to measure the frequency of a term in a document. It indicates how often a particular term appears in a document relative to the total number of terms in that document.

TF can be calculated using different approaches, but one common method is to use the raw count of the term within the document. It can be computed as follows:

TF(term, document) = (Number of times the term appears in the document) / (Total number of terms in the document)

In [8]:

```
total_terms = len(lemmatized_tokens)
term_frequency = Counter(lemmatized_tokens)
for term,freq in term_frequency.items():
    print(f"{term} : {np.round(freq/total_terms,3)}")
```

```
Modi : 0.042
led : 0.042
BJP : 0.042
2014 : 0.042
general : 0.042
election : 0.042
gave : 0.042
party : 0.083
majority : 0.042
lower : 0.042
house : 0.042
Indian : 0.042
parliament : 0.042
, : 0.083
Lok : 0.042
Sabha : 0.042
first : 0.042
time : 0.042
single : 0.042
since : 0.042
1984 : 0.042
. : 0.042
```

In [23]:

```
#instead of repeating all the steps , we will create a single function
```

In [9]:

```
def Preprocess(document):
    words = nltk.word_tokenize(document)
    review = [lemma.lemmatize(word) for word in words if word not in stopwords.words("en
    review = " ".join(review)
    return review
```

In [10]:

```
document1 = Preprocess(document1)
document2 = Preprocess(document2)
```

In [21]:

```
document1
```

Out[21]:

```
'Modi led BJP 2014 general election gave party majority lower house India
n parliament , Lok Sabha , first time single party since 1984 .'
```

In [22]:

```
document2
```

Out[22]:

```
"Following party 's victory 2019 general election , administration revoke
d special status Jammu Kashmir , introduced Citizenship Amendment Act thr
ee controversial farm law , prompted widespread protest sit-in across cou
ntry , resulting formal repeal latter ."
```

- Inverse Document Frequency

Inverse Document Frequency (IDF) is a term commonly used in information retrieval and natural language processing to measure the significance or importance of a term in a collection of documents. IDF is based on the intuition that rare terms that appear in few documents are more informative and carry more weight than common terms that appear in many documents.

IDF is calculated as the logarithm of the total number of documents divided by the number of documents containing a specific term, and then inverted:

IDF(term) = log(N / (1 + DF(term)))

Where:

N is the total number of documents in the collection. DF(term) is the number of documents containing the term.

In [11]:

```
documnets = [document1,document2]
N = len(documnets)
def DF(term,documents=documnets):
    x = 0
    for document in documents:
        if term in document:
            x+=1
    return x
```

In [12]:

```
for term in lemmatized_tokens:
    print(f"IDF {term} : {np.round(np.log(N/DF(term)),3)}")
```

```
IDF Modi : 0.693
IDF led : 0.693
IDF BJP : 0.693
IDF 2014 : 0.693
IDF general : 0.0
IDF election : 0.0
IDF gave : 0.693
IDF party : 0.0
IDF majority : 0.693
IDF lower : 0.693
IDF house : 0.693
IDF Indian : 0.693
IDF parliament : 0.693
IDF , : 0.0
IDF Lok : 0.693
IDF Sabha : 0.693
IDF , : 0.0
IDF first : 0.693
IDF time : 0.693
IDF single : 0.693
IDF party : 0.0
IDF since : 0.693
IDF 1984 : 0.693
IDF . : 0.0
```

Using tfidf

In [13]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [14]:

```
vec = TfidfVectorizer(stop_words=stopwords.words("english"))
```

In [15]:

```
documnets = [document1,document2]
```

In [16]:

```
documnets
```

Out[16]:

```
['Modi led BJP 2014 general election gave party majority lower house Indi
an parliament , Lok Sabha , first time single party since 1984 .',
 "Following party 's victory 2019 general election , administration revok
ed special status Jammu Kashmir , introduced Citizenship Amendment Act th
ree controversial farm law , prompted widespread protest sit-in across co
untry , resulting formal repeal latter ."]
```

In [17]:

```
vec.fit(documnets)
```

Out[17]:

```
TfidfVectorizer(stop_words=['i', 'me', 'my', 'myself', 'we', 'our', 'our
s',
                            'ourselves', 'you', "you're", "you've", "yo
u'll",
                            "you'd", 'your', 'yours', 'yourself', 'yourse
lves',
                            'he', 'him', 'his', 'himself', 'she', "sh
e's",
                            'her', 'hers', 'herself', 'it', "it's", 'it
s',
                            'itself', ...])
```

In [18]:

```
feature_names = vec.get_feature_names()
```

In [19]:

```
matrix = vec.transform(documnets)
```

In [20]:

```
matrix.toarray()
```

Out[20]:

```
array([[0.22339766, 0.22339766, 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.22339766, 0.        , 0.        ,
        0.        , 0.15894928, 0.        , 0.22339766, 0.        ,
        0.        , 0.22339766, 0.15894928, 0.22339766, 0.22339766,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.22339766, 0.22339766, 0.22339766, 0.22339766, 0.22339766,
        0.22339766, 0.31789855, 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.22339766, 0.22339766, 0.22339766,
        0.        , 0.        , 0.        , 0.        , 0.22339766,
        0.        , 0.        ],
       [0.        , 0.        , 0.18725563, 0.18725563, 0.18725563,
        0.18725563, 0.18725563, 0.        , 0.18725563, 0.18725563,
        0.18725563, 0.13323393, 0.18725563, 0.        , 0.18725563,
        0.18725563, 0.        , 0.13323393, 0.        , 0.        ,
        0.18725563, 0.18725563, 0.18725563, 0.18725563, 0.18725563,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.13323393, 0.18725563, 0.18725563, 0.18725563,
        0.18725563, 0.18725563, 0.        , 0.        , 0.        ,
        0.18725563, 0.18725563, 0.18725563, 0.18725563, 0.        ,
        0.18725563, 0.18725563]])
```

In [ ]:

```
```

In [38]:

```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

In [39]:

```
df = sns.load_dataset("titanic")
```

In [40]:

```
df.head(4)
```

Out[40]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_ma
0	0	3	male	22.0	1	0	7.2500	S	Third	man	Tru
1	1	1	female	38.0	1	0	71.2833	C	First	woman	Fals
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	Fals
3	1	1	female	35.0	1	0	53.1000	S	First	woman	Fals

In [41]:

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

Out[41]:

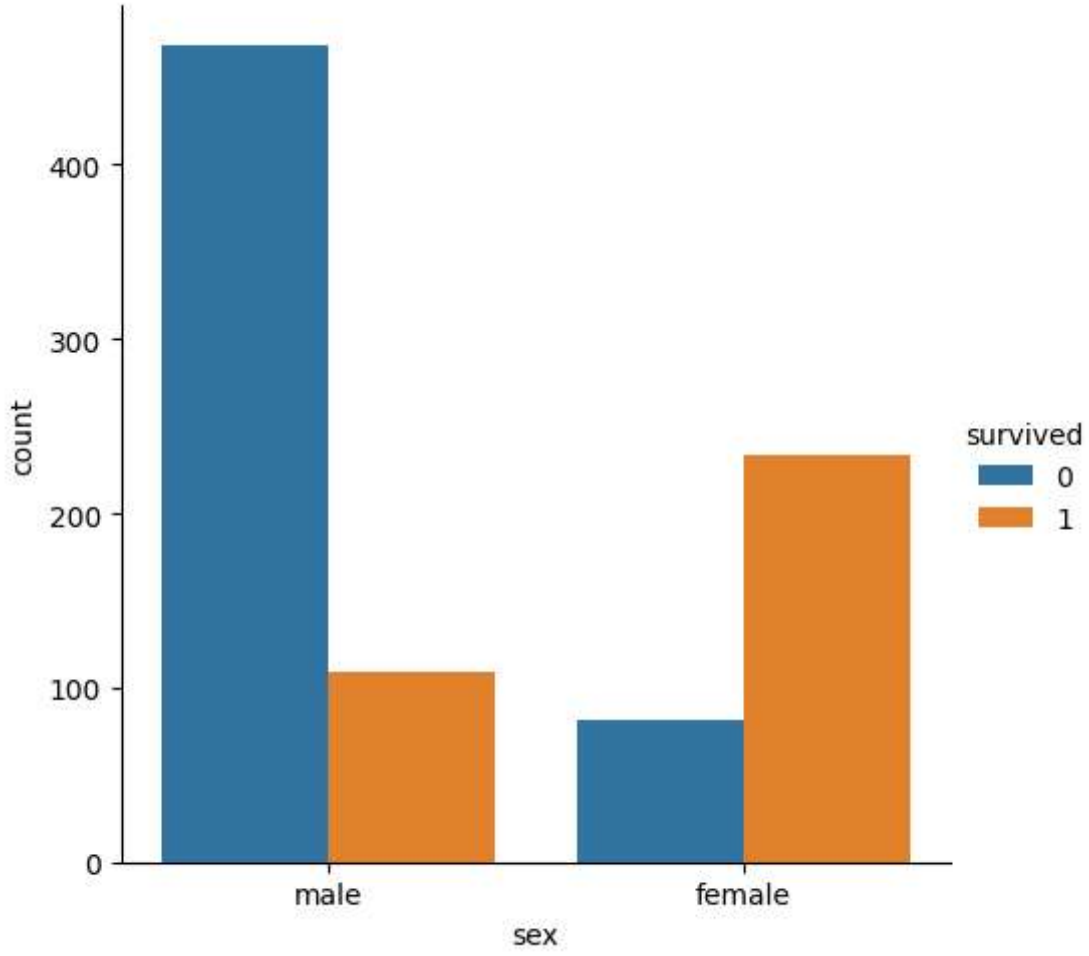
```
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64
```

In [42]:

```
sns.catplot(x = "sex", hue= "survived" , kind= "count", data = df)
```

Out[42]:

```
<seaborn.axisgrid.FacetGrid at 0x2198dcd7130>
```

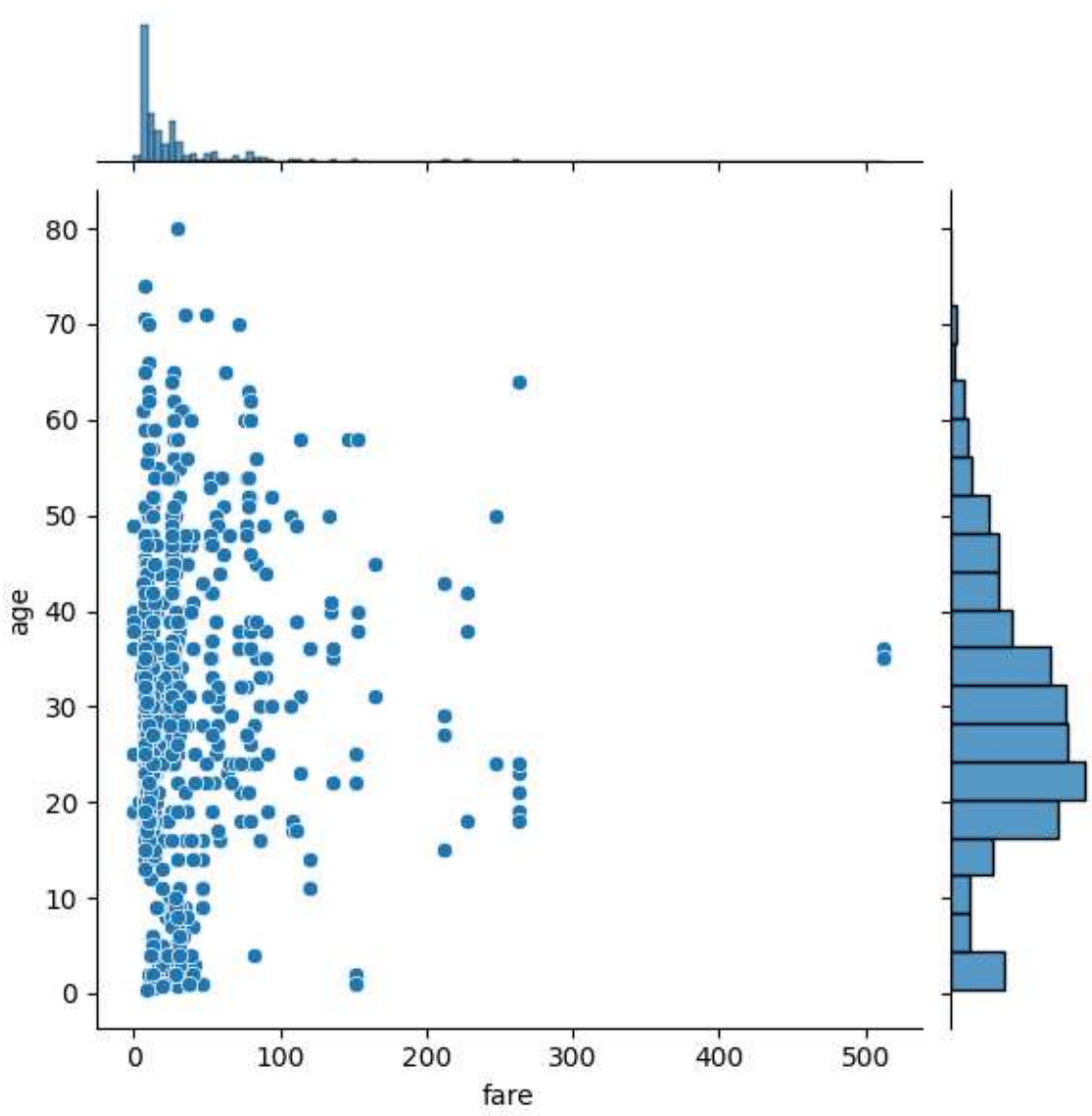


In [43]:

```
sns.jointplot(x="fare",y="age",data=df)
```

Out[43]:

```
<seaborn.axisgrid.JointGrid at 0x2198dd4b310>
```

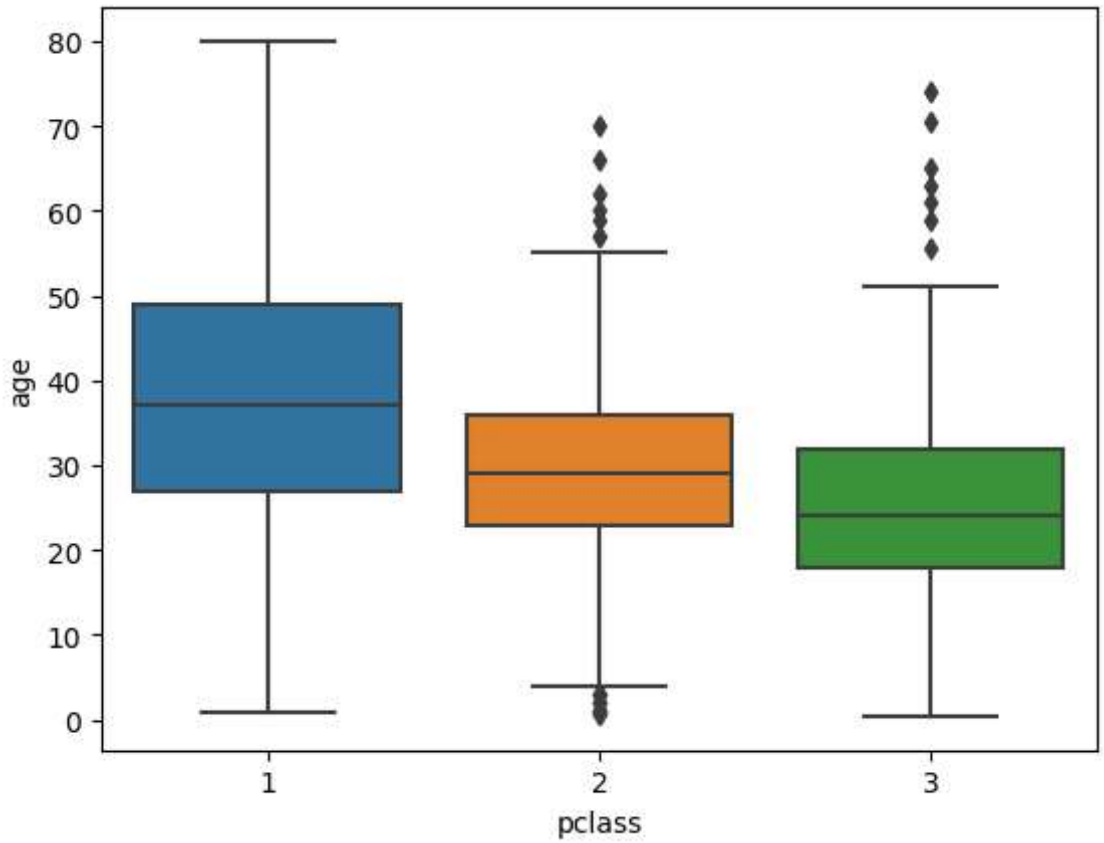


In [44]:

```
sns.boxplot(y='age', x='pclass',data=df)
```

Out[44]:

```
<AxesSubplot: xlabel='pclass', ylabel='age'>
```

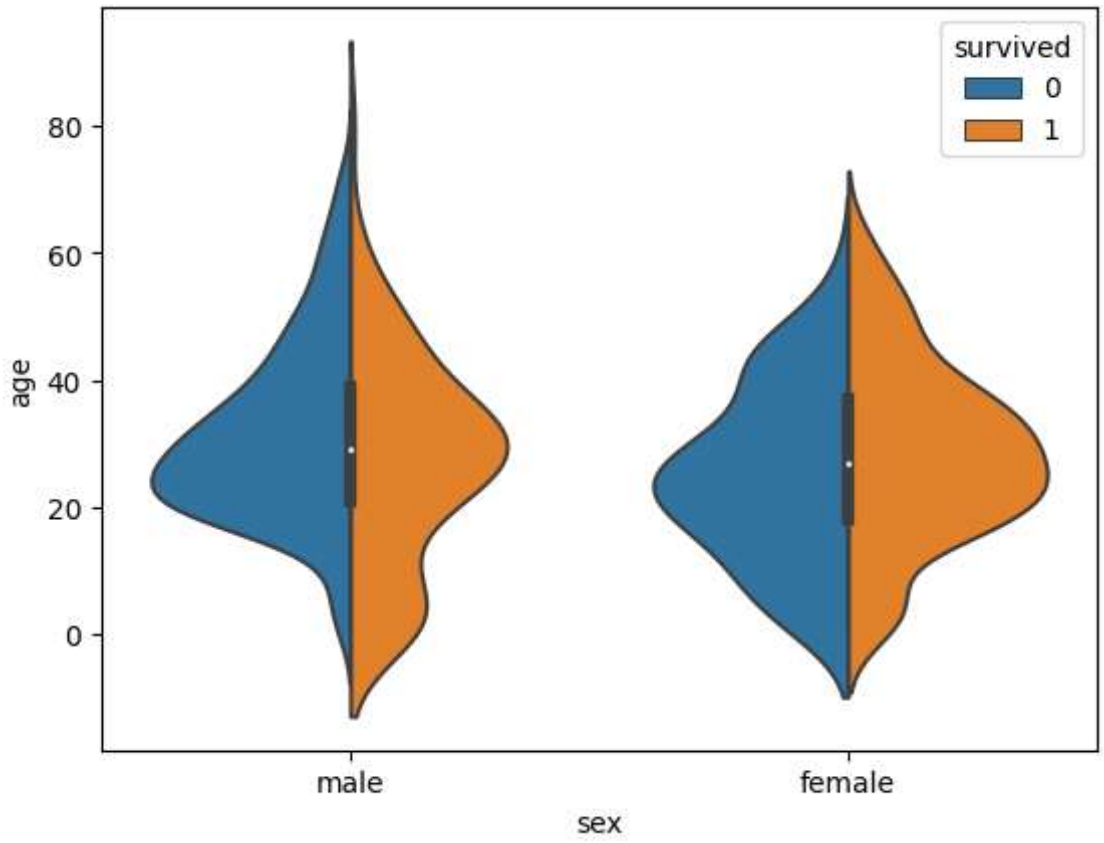


In [45]:

```
sns.violinplot(x="sex", y= "age", hue="survived", data =df , split= True)
```

Out[45]:

```
<AxesSubplot: xlabel='sex', ylabel='age'>
```

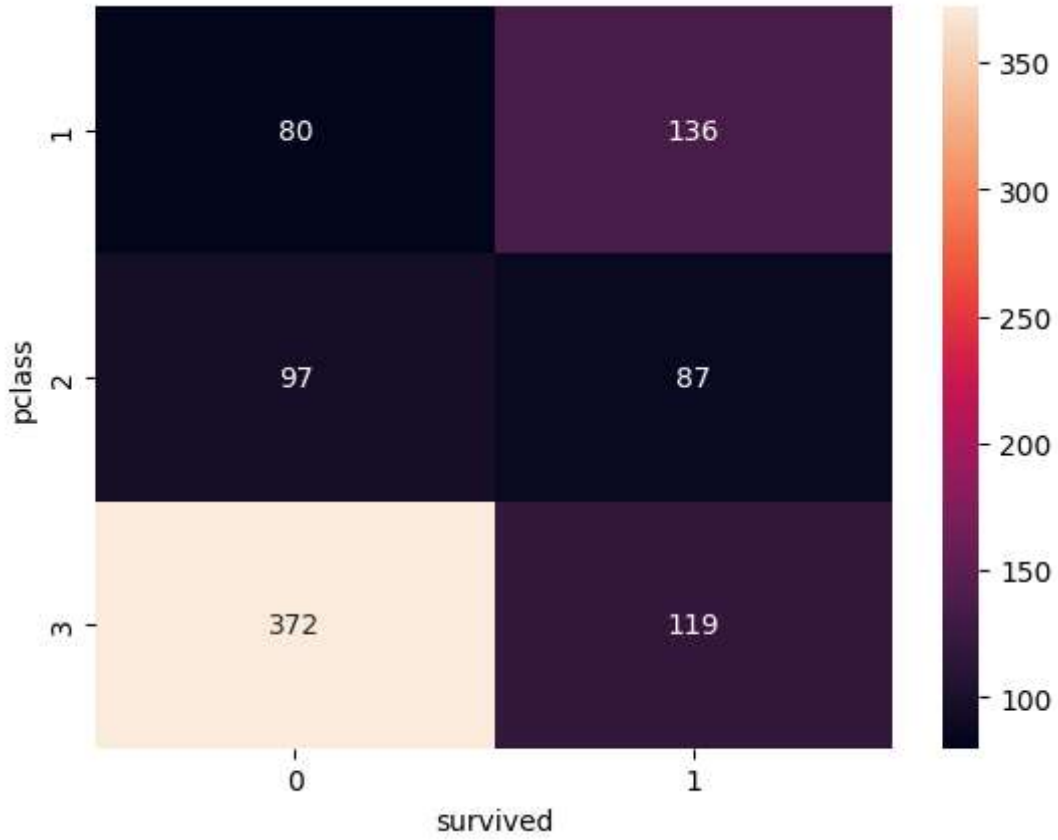


In [46]:

```
group = df.groupby(["pclass","survived"])
pclass_survived = group.size().unstack()
sns.heatmap(pclass_survived, annot = True, fmt="d")
```

Out[46]:

```
<AxesSubplot: xlabel='survived', ylabel='pclass'>
```

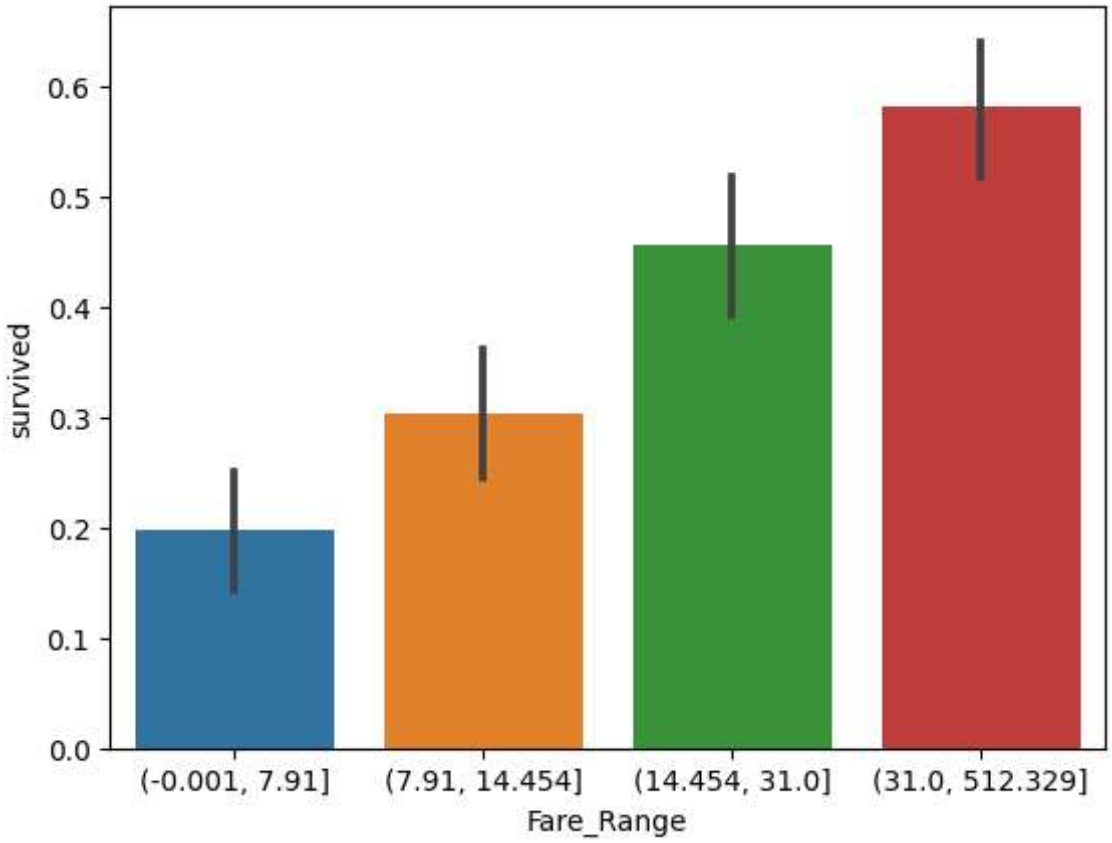


In [47]:

```
df["Fare_Range"] = pd.qcut(df["fare"],4)
sns.barplot(x="Fare_Range" , y = "survived", data = df)
```

Out[47]:

```
<AxesSubplot: xlabel='Fare_Range', ylabel='survived'>
```

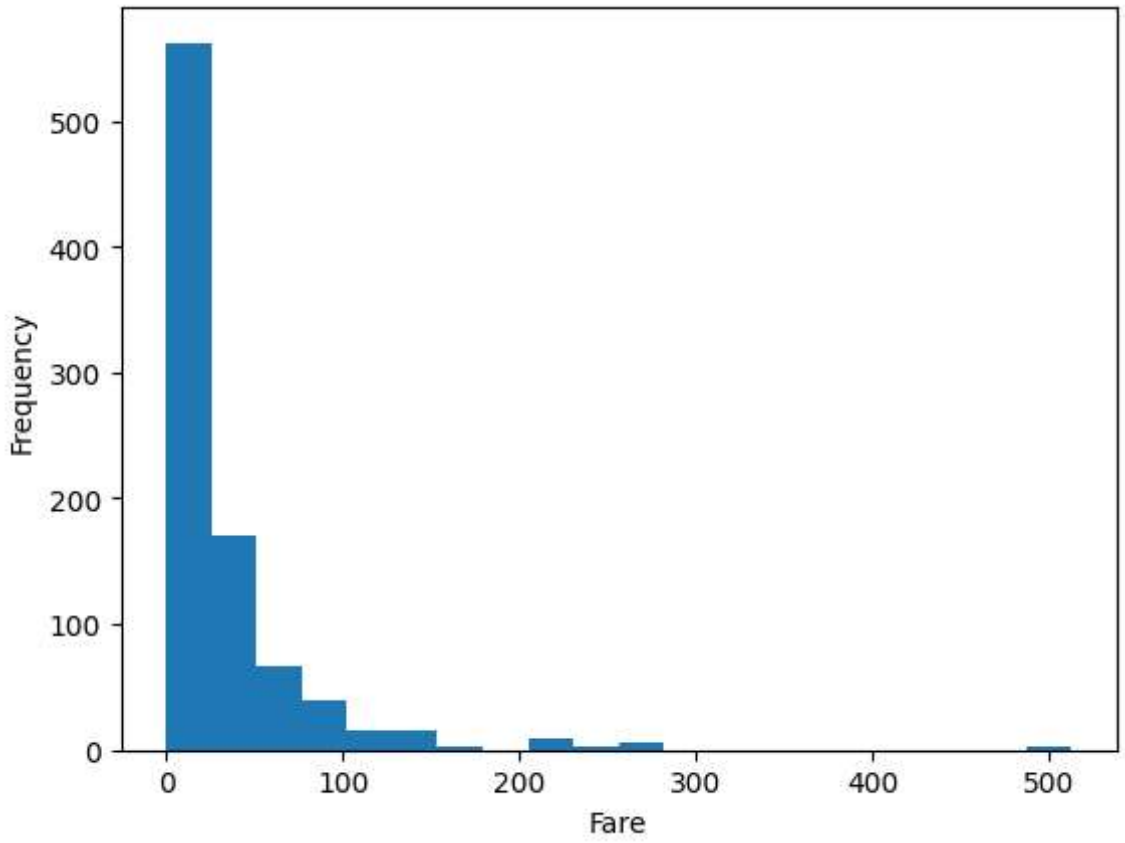


In [48]:

```
df.fare.plot.hist(bins = 20)
plt.xlabel("Fare")
```

Out[48]:

```
Text(0.5, 0, 'Fare')
```



In [ ]:

In [ ]:

In [44]:
```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
```

In [45]:
```
df = sns.load_dataset("titanic")
```

In [46]:
```
df.head(4)
```

Out[46]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False

In [47]:
```
df.isnull().sum()
```

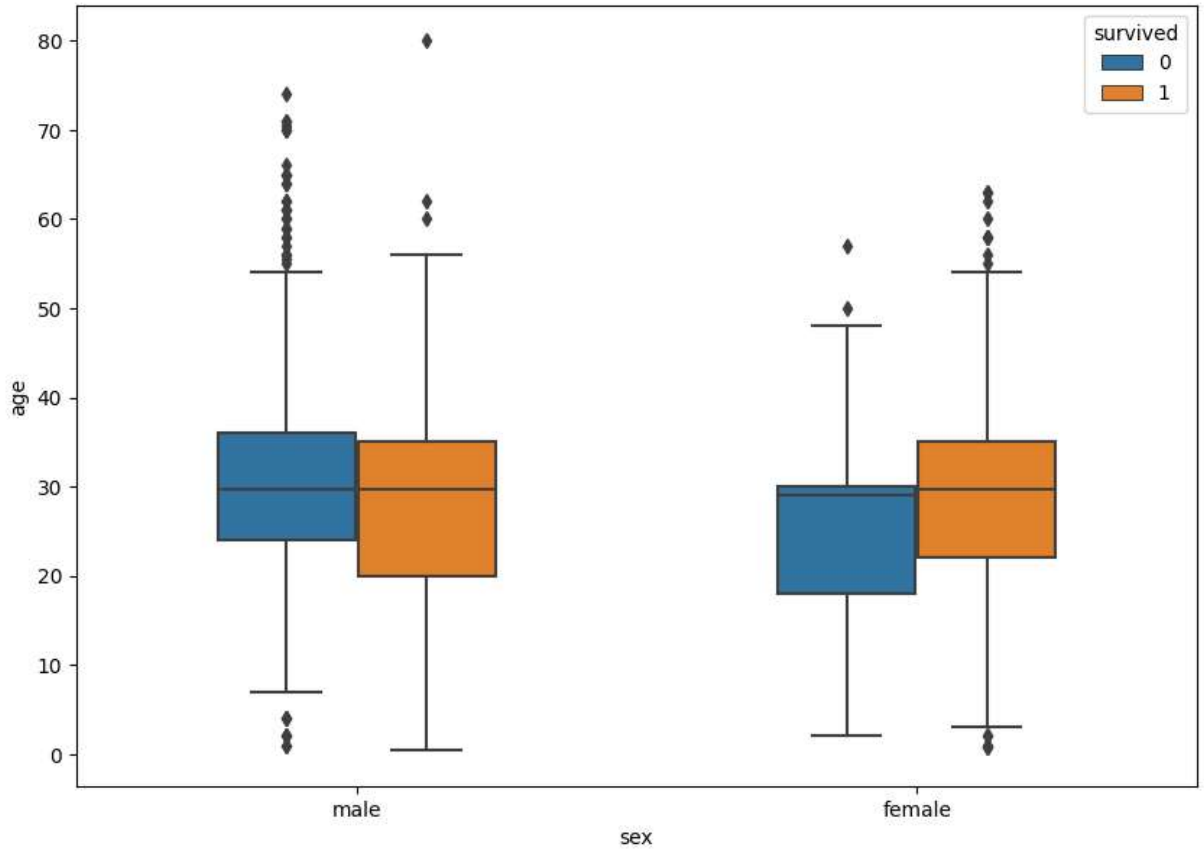
Out[47]:
```
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64
```

In [48]:
```
df["age"] = df["age"].fillna(df["age"].mean())
```

In [49]:
```
df.isnull().sum()
```

Out[49]:
```
survived         0
pclass           0
sex              0
age              0
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64
```

In [50]:
```
plt.figure(figsize=(10,7))
sns.boxplot(data = df , y = "age" , x = "sex" , hue = "survived" , orient = "v", width = 0.5)
plt.show()
```



In [ ]:

In [ ]:

In [35]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [36]:
```
df = pd.read_csv("iris.csv")
```

In [37]:
```
df.head()
```

Out[37]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [38]:
```
df.info()
```

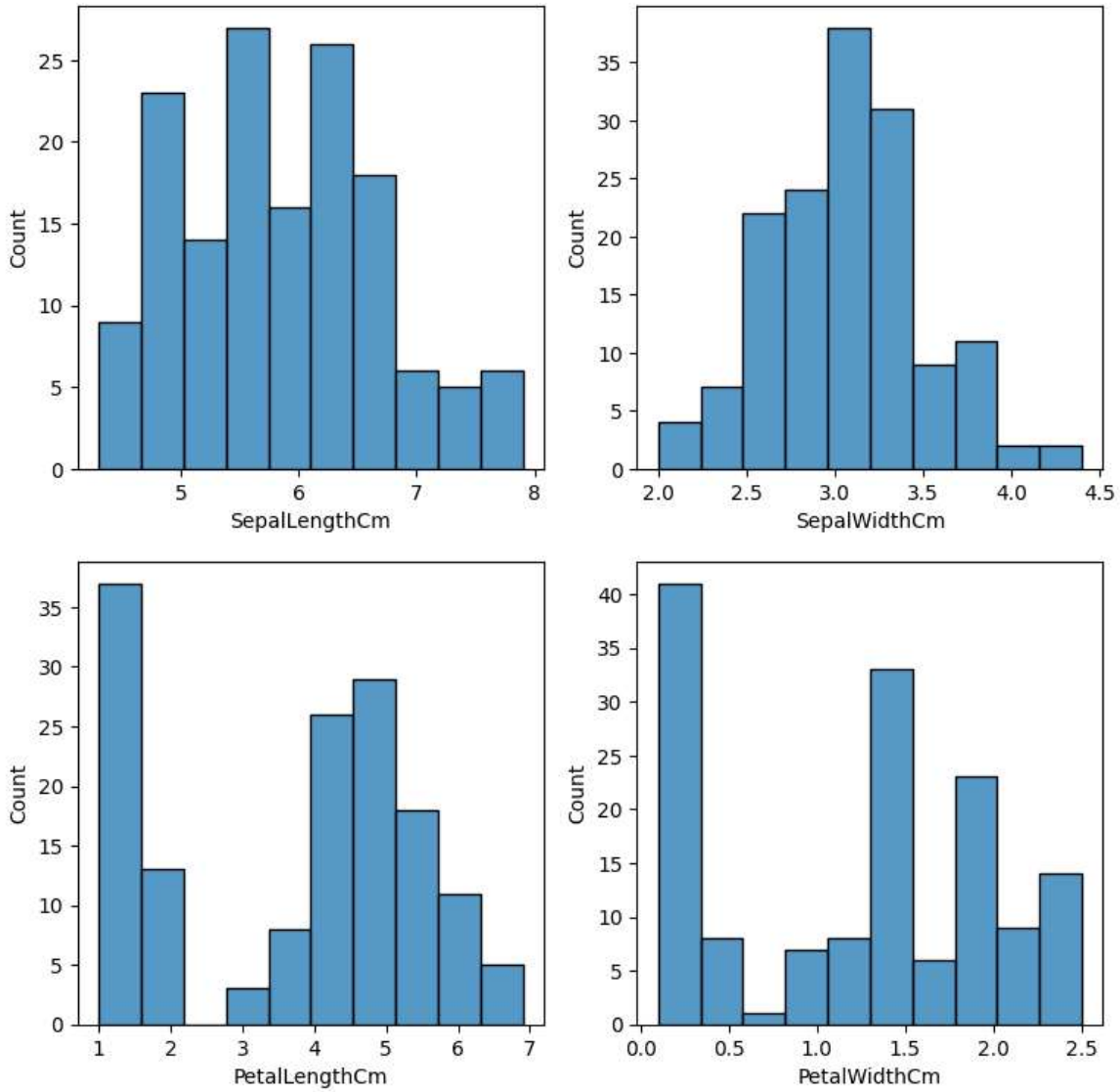
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

In [39]:
```
df.describe()
```

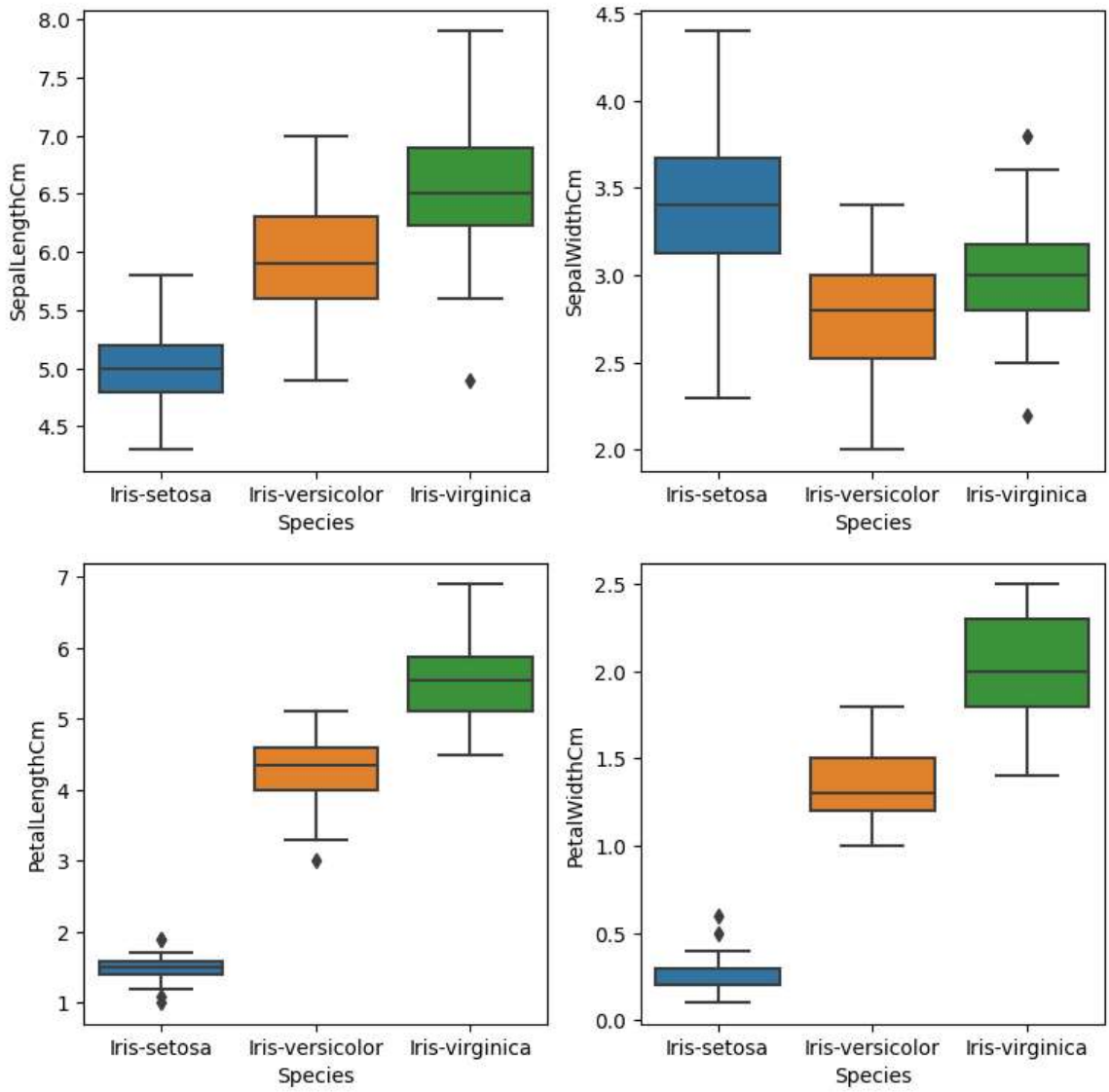
Out[39]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

In [40]:
```
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(9, 9))
axes = axes.flatten()
ax = sns.histplot(x="SepalLengthCm", data=df, bins=10,ax=axes[0])
ax = sns.histplot(x="SepalWidthCm", data=df, bins=10,ax=axes[1])
ax = sns.histplot(x="PetalLengthCm", data=df, bins=10,ax=axes[2])
ax = sns.histplot(x="PetalWidthCm", data=df, bins=10, ax=axes[3])
```



In [41]:
```
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(9, 9))
axes = axes.flatten()
ax = sns.boxplot(x="Species", y="SepalLengthCm", data=df, orient='v', ax=axes[0])
ax = sns.boxplot(x="Species", y="SepalWidthCm", data=df, orient='v', ax=axes[1])
ax = sns.boxplot(x="Species", y="PetalLengthCm", data=df, orient='v', ax=axes[2])
ax = sns.boxplot(x="Species", y="PetalWidthCm", data=df, orient='v', ax=axes[3])
```



Outliers found in

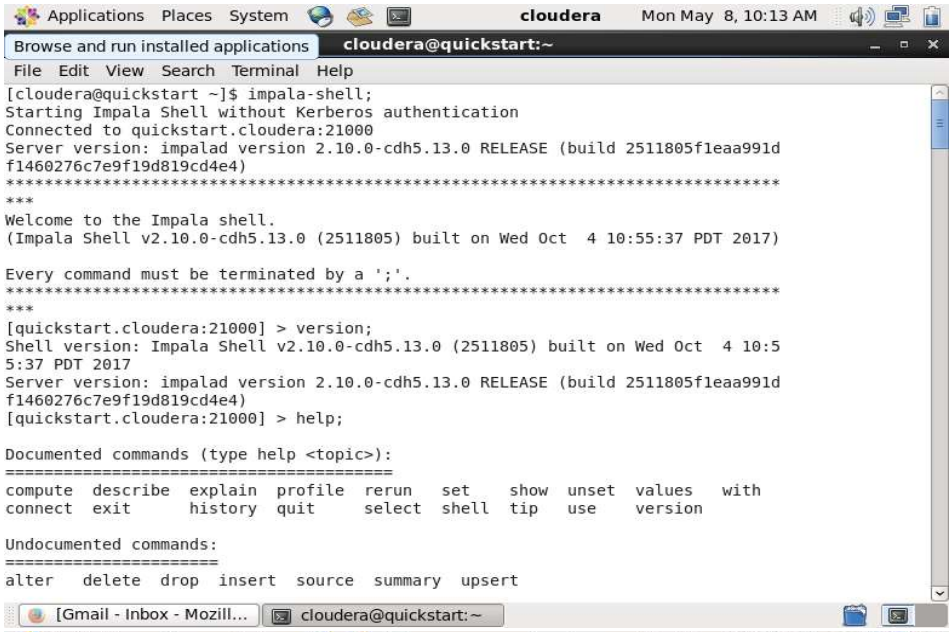
1. Iris Setosa (Petal Length, Petal Width)

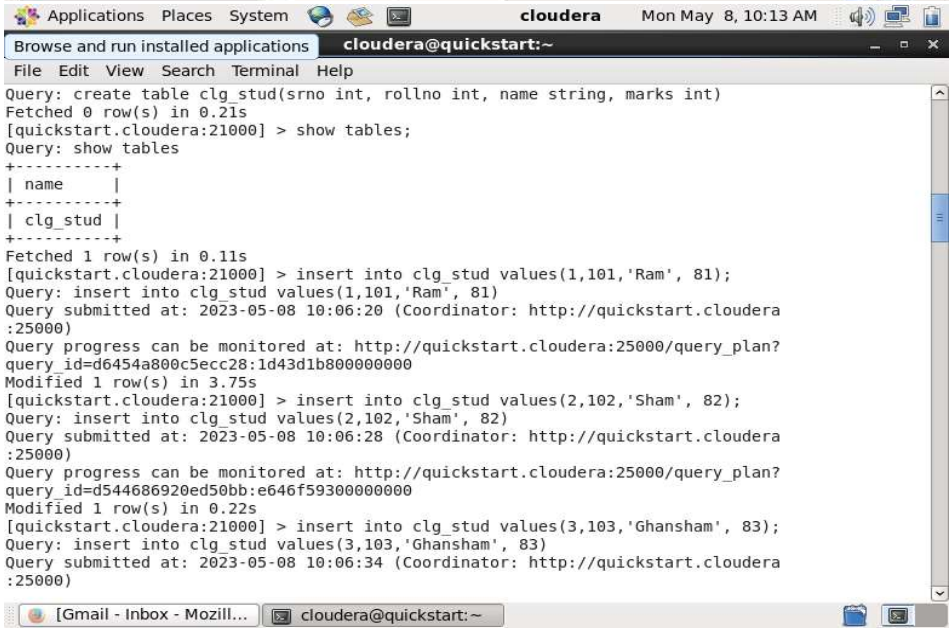
2. Iris Versicolor (Petal Length)

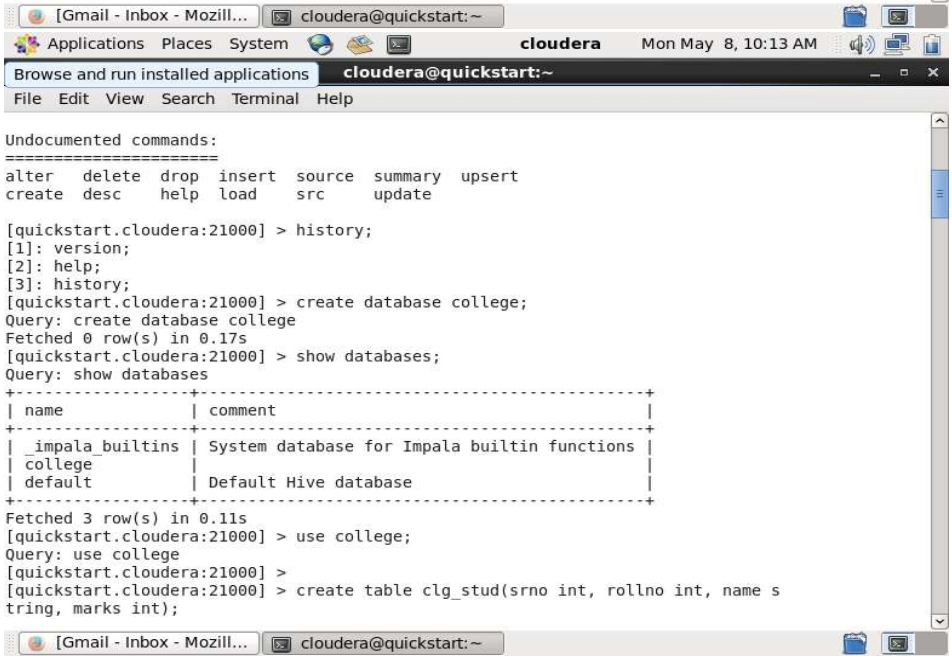
3. Iris Virginica (Sepal Lenght, Sepal Width)

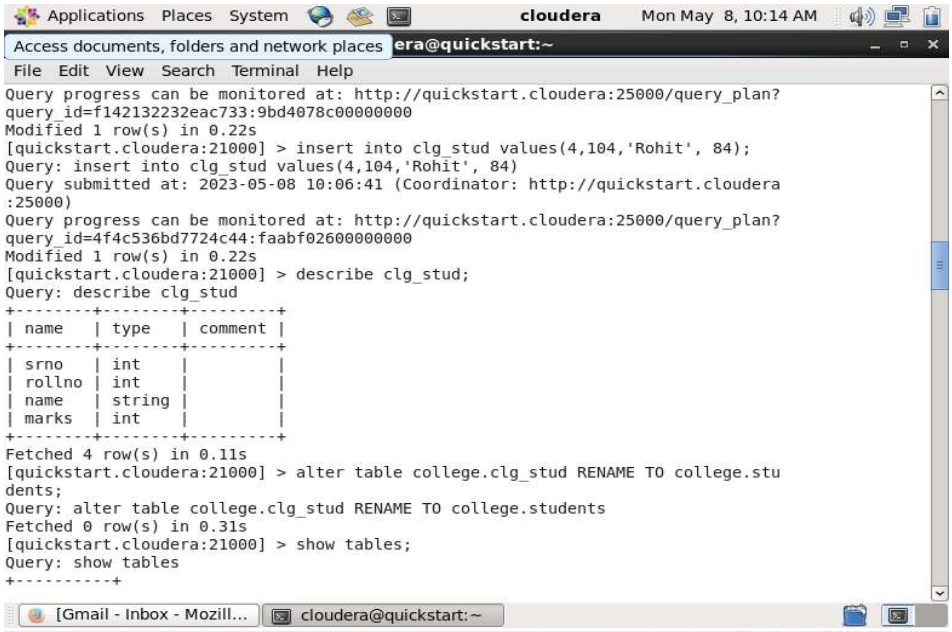
In [21]:

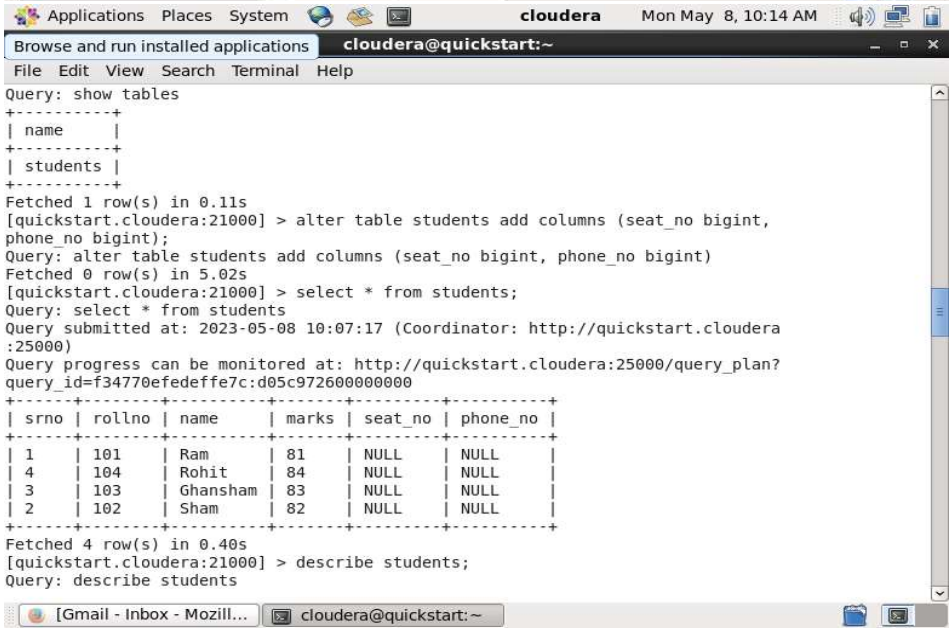
In [ ]:

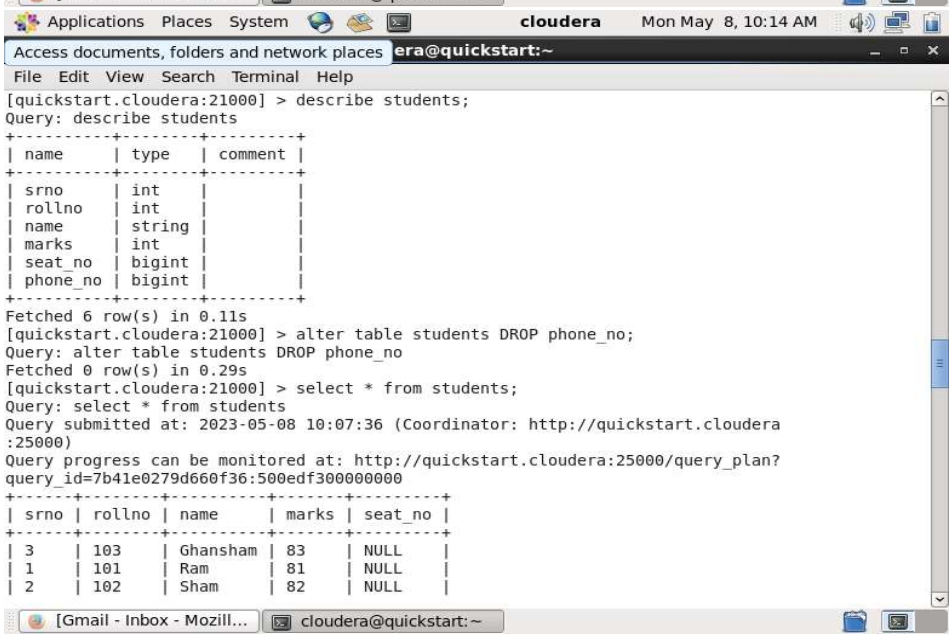

Applications Places System
cloudera
Mon May 8, 10:13 AM
Browse and run installed applications
cloudera@quickstart:~
File Edit View Search Terminal Help
[cloudera@quickstart ~]$ impala-shell;
Starting Impala Shell without Kerberos authentication
Connected to quickstart.cloudera:21000
Server version: impalad version 2.10.0-cdh5.13.0 RELEASE (build 2511805f1eaa991d
f1460276c7e9f19d819cd4e4)
Welcome to the Impala shell.
(Impala Shell v2.10.0-cdh5.13.0 (2511805) built on Wed Oct 4 10:55:37 PDT 2017)
Every command must be terminated by a ';'.
[quickstart.cloudera:21000] > version;
Shell version: Impala Shell v2.10.0-cdh5.13.0 (2511805) built on Wed Oct 4 10:5
5:37 PDT 2017
Server version: impalad version 2.10.0-cdh5.13.0 RELEASE (build 2511805f1eaa991d
f1460276c7e9f19d819cd4e4)
[quickstart.cloudera:21000] > help;
Documented commands (type help <topic>):
compute describe explain profile rerun set show unset values with
connect exit history quit select shell tip use version
Undocumented commands:
alter delete drop insert source summary upsert
[Gmail - Inbox - Mozill...
cloudera@quickstart:~

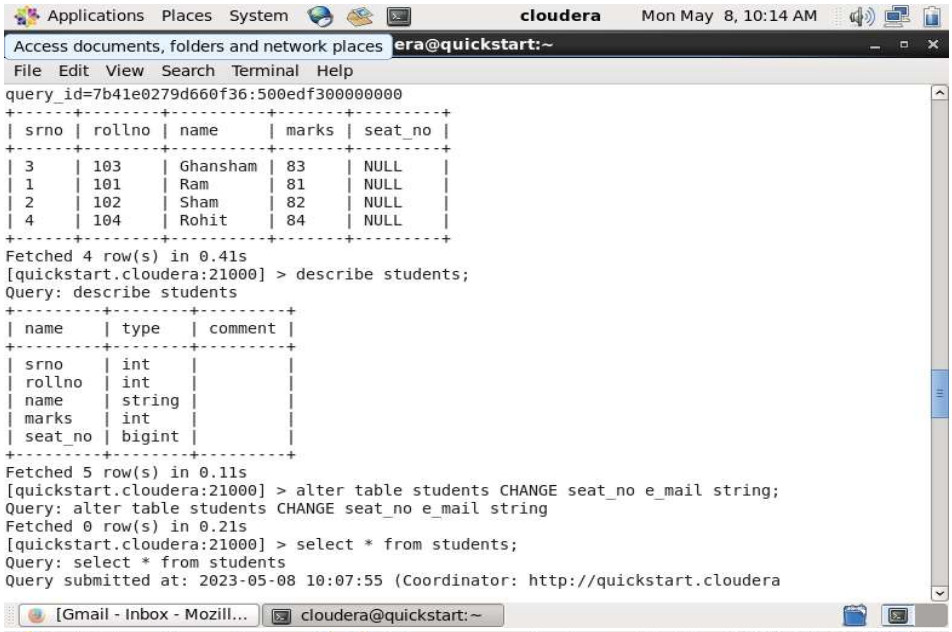

Applications Places System
cloudera
Mon May 8, 10:13 AM
Browse and run installed applications
cloudera@quickstart:~
File Edit View Search Terminal Help
Query: create table clg_stud(srno int, rollno int, name string, marks int)
Fetched 0 row(s) in 0.21s
[quickstart.cloudera:21000] > show tables;
Query: show tables
+----------+
| name |
+----------+
| clg_stud |
+----------+
Fetched 1 row(s) in 0.11s
[quickstart.cloudera:21000] > insert into clg_stud values(1,101,'Ram', 81);
Query: insert into clg_stud values(1,101,'Ram', 81)
Query submitted at: 2023-05-08 10:06:20 (Coordinator: http://quickstart.cloudera
:25000)
Query progress can be monitored at: http://quickstart.cloudera:25000/query_plan?
query_id=d6454a800c5ecc28:1d43d1b800000000
Modified 1 row(s) in 3.75s
[quickstart.cloudera:21000] > insert into clg_stud values(2,102,'Sham', 82);
Query: insert into clg_stud values(2,102,'Sham', 82)
Query submitted at: 2023-05-08 10:06:28 (Coordinator: http://quickstart.cloudera
:25000)
Query progress can be monitored at: http://quickstart.cloudera:25000/query_plan?
query_id=d544686920ed50bb:e646f59300000000
Modified 1 row(s) in 0.22s
[quickstart.cloudera:21000] > insert into clg_stud values(3,103,'Ghansham', 83);
Query: insert into clg_stud values(3,103,'Ghansham', 83)
Query submitted at: 2023-05-08 10:06:34 (Coordinator: http://quickstart.cloudera
:25000)
[Gmail - Inbox - Mozill...
cloudera@quickstart:~

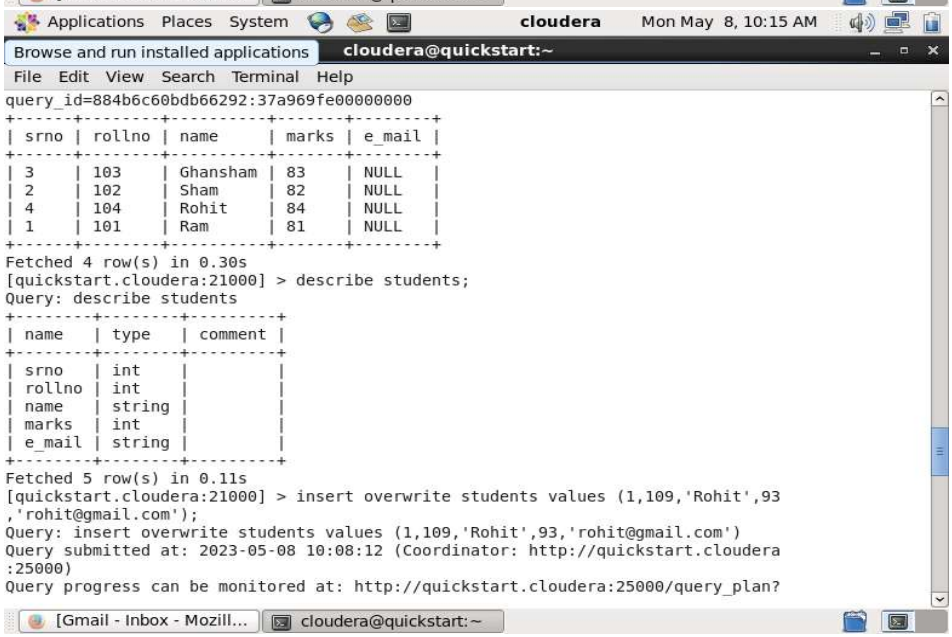

Applications Places System
cloudera
Mon May 8, 10:13 AM
Browse and run installed applications
cloudera@quickstart:~
File Edit View Search Terminal Help
Undocumented commands:
alter delete drop insert source summary upsert
create desc help load src update
[quickstart.cloudera:21000] > history;
[1]: version;
[2]: help;
[3]: history;
[quickstart.cloudera:21000] > create database college;
Query: create database college
Fetched 0 row(s) in 0.17s
[quickstart.cloudera:21000] > show databases;
Query: show databases
name	comment
_impala_builtins	System database for Impala builtin functions
college	
default	Default Hive database
Fetched 3 row(s) in 0.11s
[quickstart.cloudera:21000] > use college;
Query: use college
[quickstart.cloudera:21000] >
[quickstart.cloudera:21000] > create table clg_stud(srno int, rollno int, name s
tring, marks int);
[Gmail - Inbox - Mozill...
cloudera@quickstart:~

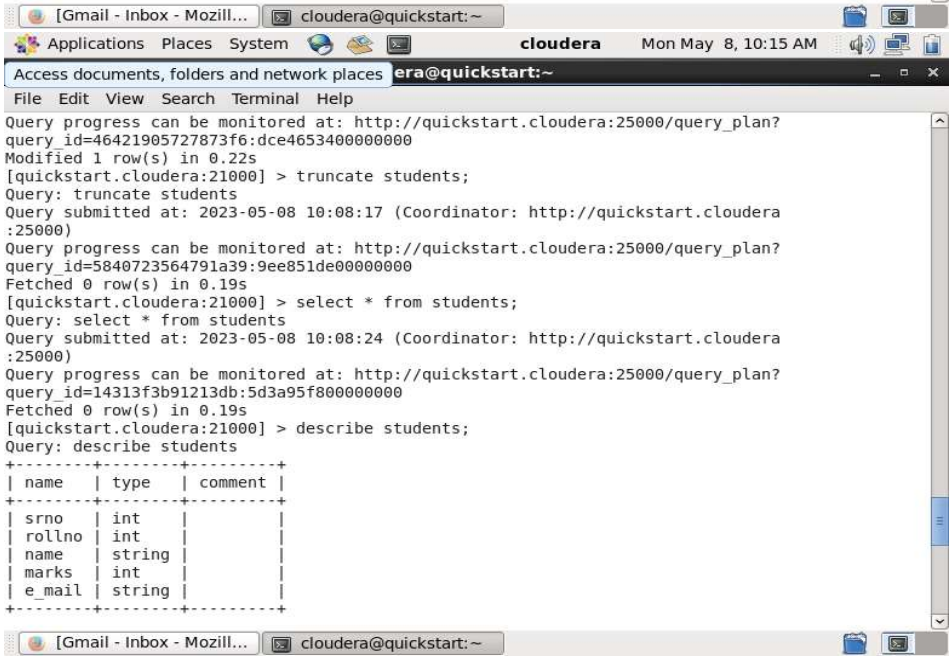

Applications Places System
cloudera
Mon May 8, 10:14 AM
Access documents, folders and network places
era@quickstart:~
File Edit View Search Terminal Help
Query progress can be monitored at: http://quickstart.cloudera:25000/query_plan?
query_id=f142132232eac733:9bd4078c00000000
Modified 1 row(s) in 0.22s
[quickstart.cloudera:21000] > insert into clg_stud values(4,104,'Rohit', 84);
Query: insert into clg_stud values(4,104,'Rohit', 84)
Query submitted at: 2023-05-08 10:06:41 (Coordinator: http://quickstart.cloudera
:25000)
Query progress can be monitored at: http://quickstart.cloudera:25000/query_plan?
query_id=4f4c536bd7724c44:faabf02600000000
Modified 1 row(s) in 0.22s
[quickstart.cloudera:21000] > describe clg_stud;
Query: describe clg_stud
+--------+--------+---------+
| name | type | comment |
+--------+--------+---------+
srno	int	
rollno	int	
name	string	
marks	int	
+--------+--------+---------+
Fetched 4 row(s) in 0.11s
[quickstart.cloudera:21000] > alter table college.clg_stud RENAME TO college.stu
dents;
Query: alter table college.clg_stud RENAME TO college.students
Fetched 0 row(s) in 0.31s
[quickstart.cloudera:21000] > show tables;
Query: show tables
+----------+
[Gmail - Inbox - Mozill...
cloudera@quickstart:~

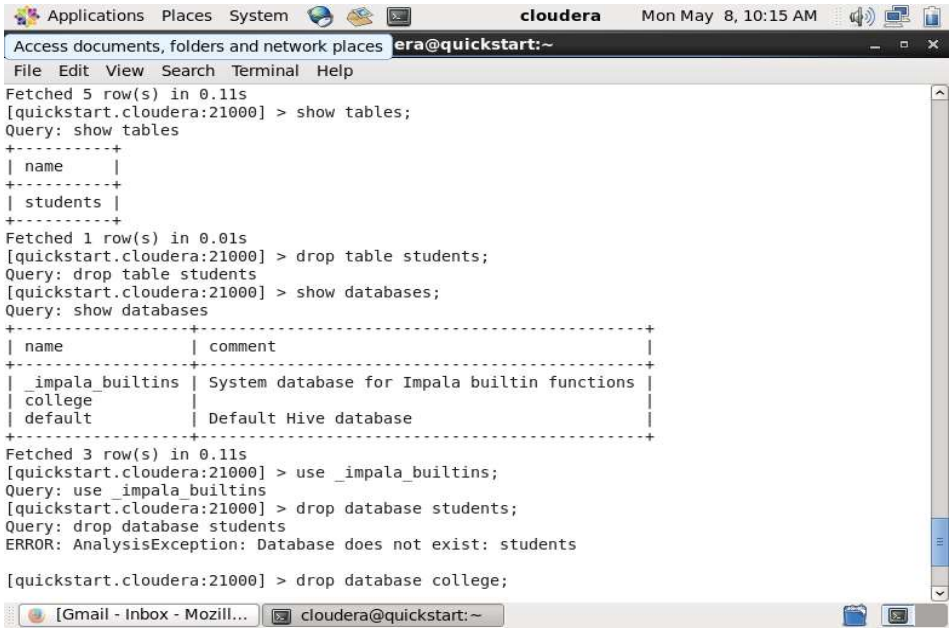

Applications Places System
cloudera
Mon May 8, 10:14 AM
Browse and run installed applications
cloudera@quickstart:~
File Edit View Search Terminal Help
Query: show tables
+----------+
| name |
+----------+
| students |
+----------+
Fetched 1 row(s) in 0.11s
[quickstart.cloudera:21000] > alter table students add columns (seat_no bigint,
phone_no bigint);
Query: alter table students add columns (seat_no bigint, phone_no bigint)
Fetched 0 row(s) in 5.02s
[quickstart.cloudera:21000] > select * from students;
Query: select * from students
Query submitted at: 2023-05-08 10:07:17 (Coordinator: http://quickstart.cloudera
:25000)
Query progress can be monitored at: http://quickstart.cloudera:25000/query_plan?
query_id=f34770efedeffe7c:d05c972600000000
+------+--------+----------+-------+---------+----------+
| srno | rollno | name | marks | seat_no | phone_no |
+------+--------+----------+-------+---------+----------+
1	101	Ram	81	NULL	NULL
4	104	Rohit	84	NULL	NULL
3	103	Ghansham	83	NULL	NULL
2	102	Sham	82	NULL	NULL
+------+--------+----------+-------+---------+----------+
Fetched 4 row(s) in 0.40s
[quickstart.cloudera:21000] > describe students;
Query: describe students
[Gmail - Inbox - Mozill...
cloudera@quickstart:~

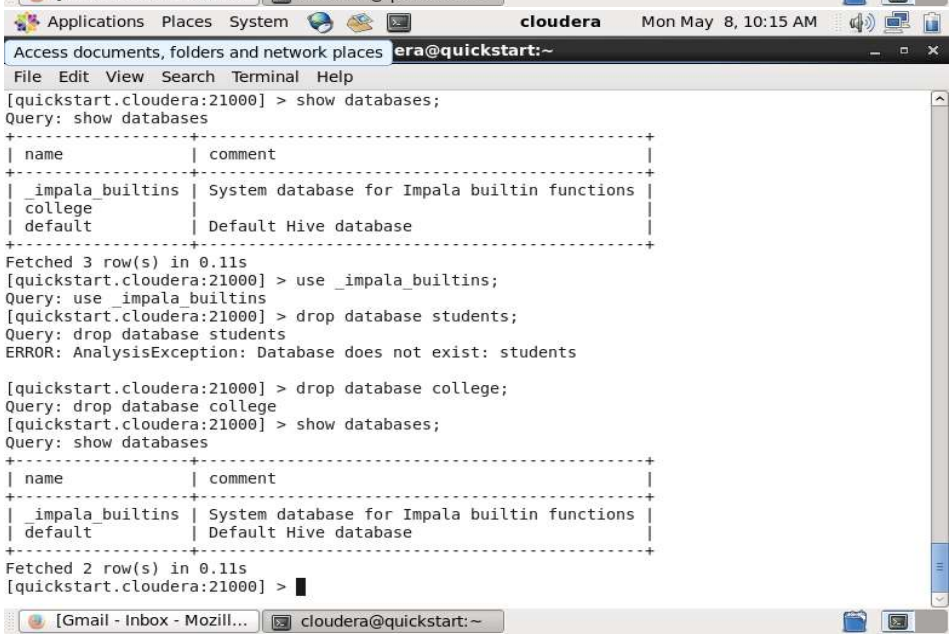

Applications Places System
cloudera
Mon May 8, 10:14 AM
Access documents, folders and network places
era@quickstart:~
File Edit View Search Terminal Help
[quickstart.cloudera:21000] > describe students;
Query: describe students
+----------+--------+---------+
| name | type | comment |
+----------+--------+---------+
srno	int	
rollno	int	
name	string	
marks	int	
seat_no	bigint	
phone_no	bigint	
+----------+--------+---------+
Fetched 6 row(s) in 0.11s
[quickstart.cloudera:21000] > alter table students DROP phone_no;
Query: alter table students DROP phone_no
Fetched 0 row(s) in 0.29s
[quickstart.cloudera:21000] > select * from students;
Query: select * from students
Query submitted at: 2023-05-08 10:07:36 (Coordinator: http://quickstart.cloudera
:25000)
Query progress can be monitored at: http://quickstart.cloudera:25000/query_plan?
query_id=7b41e0279d660f36:500edf300000000
+------+--------+----------+-------+---------+
| srno | rollno | name | marks | seat_no |
+------+--------+----------+-------+---------+
3	103	Ghansham	83	NULL
1	101	Ram	81	NULL
2	102	Sham	82	NULL
[Gmail - Inbox - Mozill...
cloudera@quickstart:~


Applications Places System
cloudera
Mon May 8, 10:14 AM
Access documents, folders and network places
era@quickstart:~
File Edit View Search Terminal Help
query_id=7b41e0279d660f36:500edf300000000
+------+--------+----------+-------+---------+
| srno | rollno | name | marks | seat_no |
+------+--------+----------+-------+---------+
3	103	Ghansham	83	NULL
1	101	Ram	81	NULL
2	102	Sham	82	NULL
4	104	Rohit	84	NULL
+------+--------+----------+-------+---------+
Fetched 4 row(s) in 0.41s
[quickstart.cloudera:21000] > describe students;
Query: describe students
+---------+--------+---------+
| name | type | comment |
+---------+--------+---------+
srno	int	
rollno	int	
name	string	
marks	int	
seat_no	bigint	
+---------+--------+---------+
Fetched 5 row(s) in 0.11s
[quickstart.cloudera:21000] > alter table students CHANGE seat_no e_mail string;
Query: alter table students CHANGE seat_no e_mail string
Fetched 0 row(s) in 0.21s
[quickstart.cloudera:21000] > select * from students;
Query: select * from students
Query submitted at: 2023-05-08 10:07:55 (Coordinator: http://quickstart.cloudera
[Gmail - Inbox - Mozill...
cloudera@quickstart:~


Applications Places System
cloudera
Mon May 8, 10:15 AM
Browse and run installed applications
cloudera@quickstart:~
File Edit View Search Terminal Help
query_id=884b6c60bdb66292:37a969fe00000000
+------+--------+----------+-------+--------+
| srno | rollno | name | marks | e_mail |
+------+--------+----------+-------+--------+
3	103	Ghansham	83	NULL
2	102	Sham	82	NULL
4	104	Rohit	84	NULL
1	101	Ram	81	NULL
+------+--------+----------+-------+--------+
Fetched 4 row(s) in 0.30s
[quickstart.cloudera:21000] > describe students;
Query: describe students
+--------+--------+---------+
| name | type | comment |
+--------+--------+---------+
srno	int	
rollno	int	
name	string	
marks	int	
e_mail	string	
+--------+--------+---------+
Fetched 5 row(s) in 0.11s
[quickstart.cloudera:21000] > insert overwrite students values (1,109,'Rohit',93
,'rohit@gmail.com');
Query: insert overwrite students values (1,109,'Rohit',93,'rohit@gmail.com')
Query submitted at: 2023-05-08 10:08:12 (Coordinator: http://quickstart.cloudera
:25000)
Query progress can be monitored at: http://quickstart.cloudera:25000/query_plan?
[Gmail - Inbox - Mozill...
cloudera@quickstart:~


Applications Places System
cloudera
Mon May 8, 10:15 AM
Access documents, folders and network places
era@quickstart:~
File Edit View Search Terminal Help
Query progress can be monitored at: http://quickstart.cloudera:25000/query_plan?
query_id=46421905727873f6:dce4653400000000
Modified 1 row(s) in 0.22s
[quickstart.cloudera:21000] > truncate students;
Query: truncate students
Query submitted at: 2023-05-08 10:08:17 (Coordinator: http://quickstart.cloudera
:25000)
Query progress can be monitored at: http://quickstart.cloudera:25000/query_plan?
query_id=5840723564791a39:9ee851de00000000
Fetched 0 row(s) in 0.19s
[quickstart.cloudera:21000] > select * from students;
Query: select * from students
Query submitted at: 2023-05-08 10:08:24 (Coordinator: http://quickstart.cloudera
:25000)
Query progress can be monitored at: http://quickstart.cloudera:25000/query_plan?
query_id=14313f3b91213db:5d3a95f800000000
Fetched 0 row(s) in 0.19s
[quickstart.cloudera:21000] > describe students;
Query: describe students
+--------+--------+---------+
| name | type | comment |
+--------+--------+---------+
srno	int	
rollno	int	
name	string	
marks	int	
e_mail	string	
+--------+--------+---------+
[Gmail - Inbox - Mozill...
cloudera@quickstart:~


Applications Places System
cloudera
Mon May 8, 10:15 AM
Access documents, folders and network places
era@quickstart:~
File Edit View Search Terminal Help
Fetched 5 row(s) in 0.11s
[quickstart.cloudera:21000] > show tables;
Query: show tables
+----------+
| name |
+----------+
| students |
+----------+
Fetched 1 row(s) in 0.01s
[quickstart.cloudera:21000] > drop table students;
Query: drop table students
[quickstart.cloudera:21000] > show databases;
Query: show databases
+------------------+--+
| name | comment |
+------------------+--+
_impala_builtins	System database for Impala builtin functions
college	
default	Default Hive database
+------------------+--+
Fetched 3 row(s) in 0.11s
[quickstart.cloudera:21000] > use _impala_builtins;
Query: use _impala_builtins
[quickstart.cloudera:21000] > drop database students;
Query: drop database students
ERROR: AnalysisException: Database does not exist: students
[quickstart.cloudera:21000] > drop database college;
[Gmail - Inbox - Mozill...
cloudera@quickstart:~

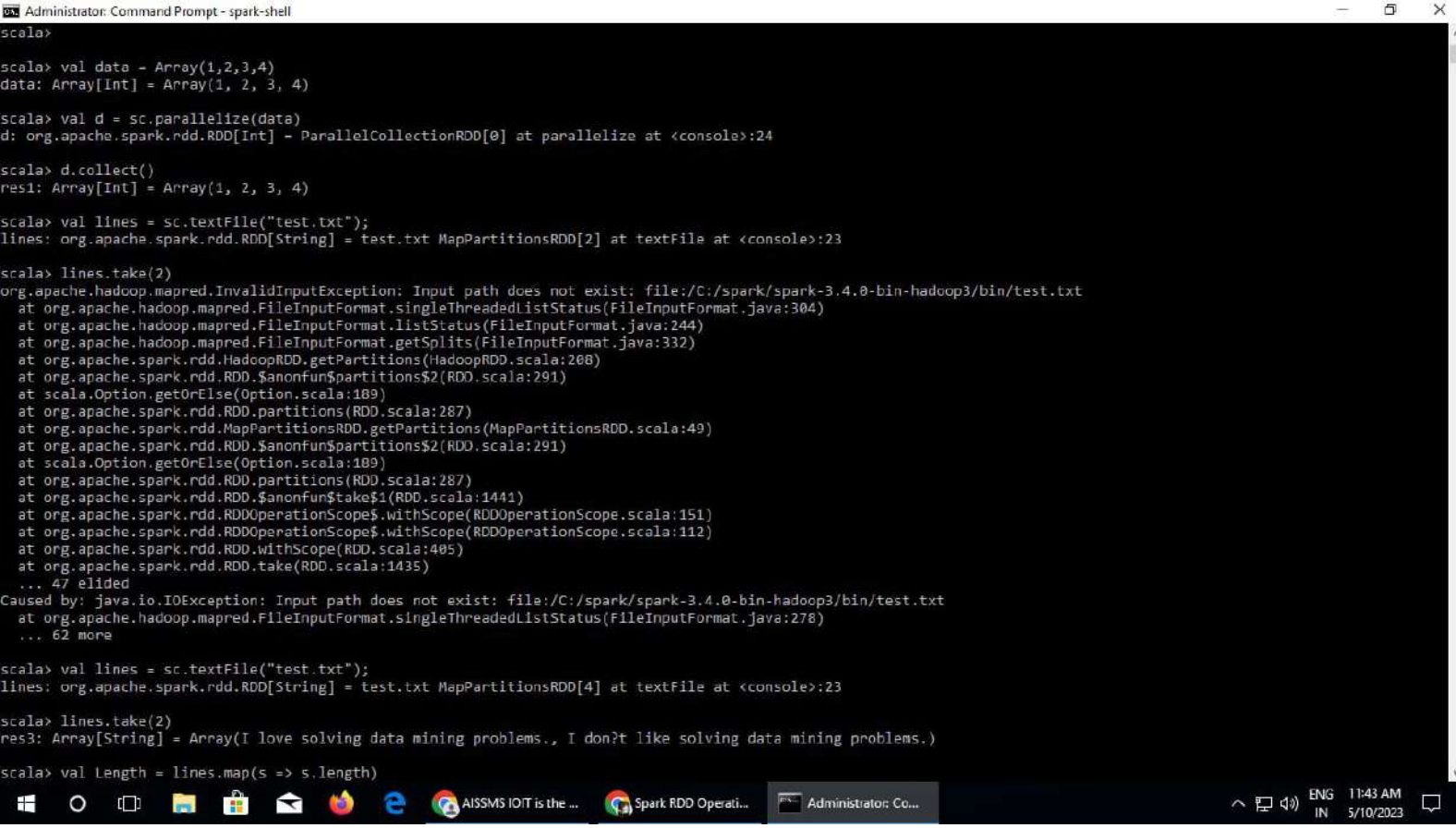

Applications Places System
cloudera
Mon May 8, 10:15 AM
Access documents, folders and network places
era@quickstart:~
File Edit View Search Terminal Help
[quickstart.cloudera:21000] > show databases;
Query: show databases
+------------------+--+
| name | comment |
+------------------+--+
_impala_builtins	System database for Impala builtin functions
college	
default	Default Hive database
+------------------+--+
Fetched 3 row(s) in 0.11s
[quickstart.cloudera:21000] > use _impala_builtins;
Query: use _impala_builtins
[quickstart.cloudera:21000] > drop database students;
Query: drop database students
ERROR: AnalysisException: Database does not exist: students
[quickstart.cloudera:21000] > drop database college;
Query: drop database college
[quickstart.cloudera:21000] > show databases;
Query: show databases
+------------------+--+
| name | comment |
+------------------+--+
| _impala_builtins | System database for Impala builtin functions |
| default | Default Hive database |
+------------------+--+
Fetched 2 row(s) in 0.11s
[quickstart.cloudera:21000] >
[Gmail - Inbox - Mozill...
cloudera@quickstart:~

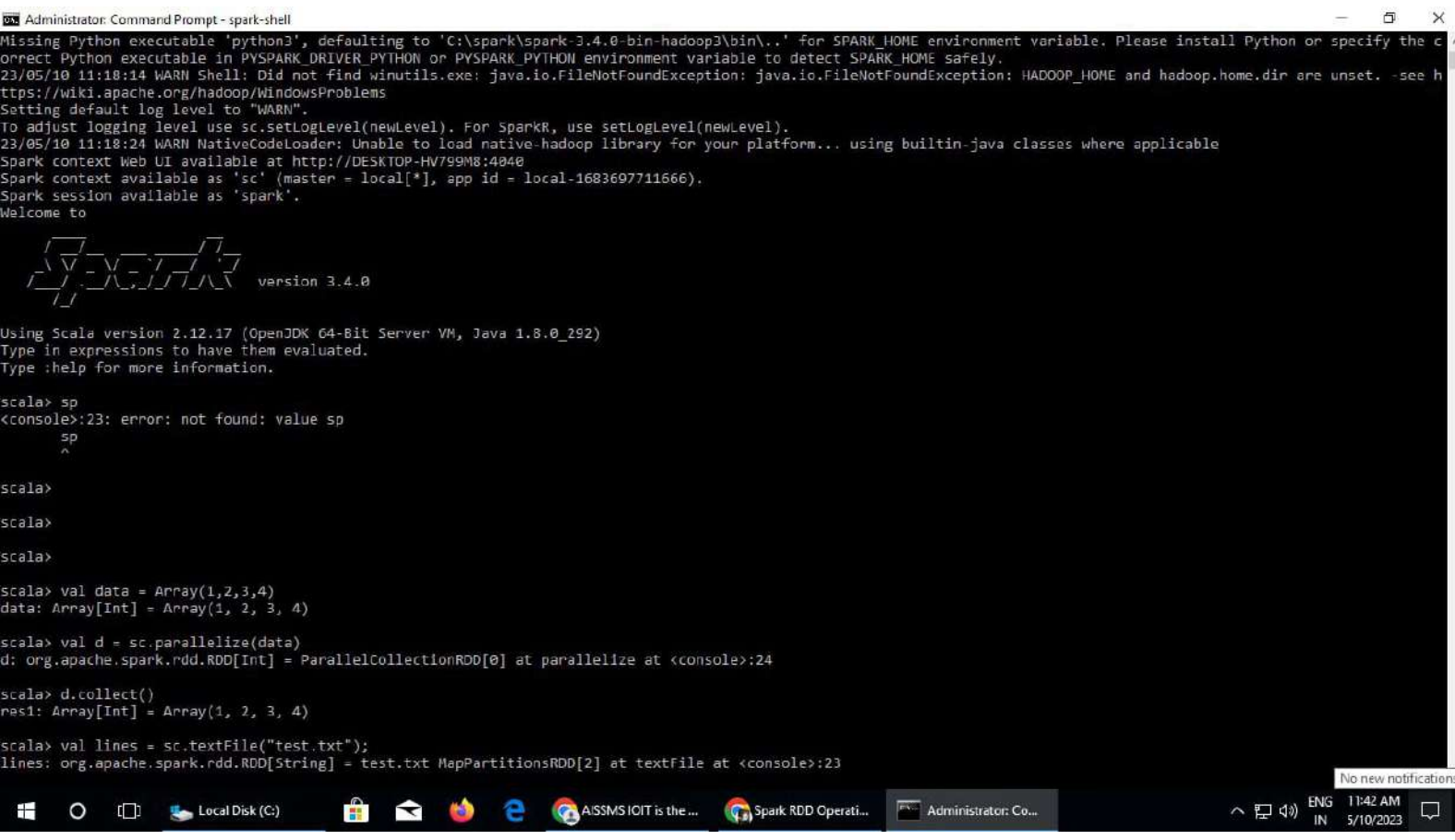
Administrator: Command Prompt - spark-shell

```
res13: Array[Int] = Array(23, 27, 45, 45, 67, 67, 78, 82, 86, 86)

scala> rdd1.sortBy(x -> x, false).collect
res14: Array[Int] = Array(86, 86, 82, 78, 67, 67, 45, 45, 27, 23)

scala> rdd1.reduce((x,y) => x + y)
res15: Int = 606

scala> rdd1.map(x => x + 1).collect
res16: Array[Int] = Array(24, 46, 68, 87, 79, 28, 83, 46, 68, 87)

scala> rdd1.map(x => x + 5).collect
res17: Array[Int] = Array(28, 50, 72, 91, 83, 32, 87, 50, 72, 91)

scala> val rdd2 = sc.parallelize(List(25, 73, 97, 78, 27, 82))
rdd2: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[20] at parallelize at <console>:23

scala> rdd1.union(rdd2).collect
res18: Array[Int] = Array(23, 45, 67, 86, 78, 27, 82, 45, 67, 86, 25, 73, 97, 78, 27, 82)

scala> rdd1.intersection(rdd2).collect
res19: Array[Int] = Array(82, 27, 78)

scala> rdd1.cartesian(rdd2).collect
res20: Array[(Int, Int)] = Array((23,25), (23,73), (23,97), (23,78), (23,27), (23,82), (45,25), (45,73), (45,97), (45,78), (45,27), (45,82), (67,25), (67,73), (67,97),
(67,78), (67,27), (67,82), (86,25), (86,73), (86,97), (86,78), (86,27), (86,82), (78,25), (78,73), (78,97), (78,78), (78,27), (78,82), (27,25), (27,73), (27,97), (27,78
), (27,27), (27,82), (82,25), (82,73), (82,97), (82,78), (82,27), (82,82), (45,25), (45,73), (45,97), (45,78), (45,27), (45,82), (67,25), (67,73), (67,97), (67,78), (67
,27), (67,82), (86,25), (86,73), (86,97), (86,78), (86,27), (86,82))

scala> rdd1.first()
res21: Int = 23

scala> rdd1.take(5)
res22: Array[Int] = Array(23, 45, 67, 86, 78)

scala>
```

Administrator: Command Prompt - spark-shell

```
scala> length.collect()
<console>:23: error: missing argument list for method length in object functions
Unapplied methods are only converted to functions when a function type is expected.
You can make this conversion explicit by writing `length _` or `length(_)` instead of `length`.
       length.collect()
       ^

scala> Length.collect()
res5: Array[Int] = Array(36, 42, 37, 43)

scala> val a = lines.flatMap(line => line.split(" ")).filter(_.length > 5)
a: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[7] at filter at <console>:23

scala> a.collect()
res6: Array[String] = Array(solving, mining, problems., solving, mining, problems., solving, science, problems., solving, science, problems.)

scala> lines.count()
res7: Long = 4

scala> val t = Length.reduce((a,b) => a+b)
t: Int = 158

scala> val rdd1 = sc.parallelize(List(23, 45, 67, 86, 78, 27, 82, 45, 67, 86))
rdd1: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[8] at parallelize at <console>:23

scala> rdd1.collect()
res8: Array[Int] = Array(23, 45, 67, 86, 78, 27, 82, 45, 67, 86)

scala> rdd1.count
res9: Long = 10

scala> rdd1.distinct.collect()
res10: Array[Int] = Array(82, 67, 86, 23, 27, 45, 78)

scala> rdd1.filter(x => x < 50).collect
res11: Array[Int] = Array(23, 45, 27, 45)

scala> rdd1.filter(x => x > 50).collect
res12: Array[Int] = Array(67, 86, 78, 82, 67, 86)

scala> rdd1.sortBy(x => x, true).collect
res13: Array[Int] = Array(23, 27, 45, 45, 67, 67, 78, 82, 86, 86)
```


Administrator: Command Prompt - spark-shell
scala>
scala> val data = Array(1,2,3,4)
data: Array[Int] = Array(1, 2, 3, 4)
scala> val d = sc.parallelize(data)
d: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[0] at parallelize at <console>:24
scala> d.collect()
res1: Array[Int] = Array(1, 2, 3, 4)
scala> val lines = sc.textFile("test.txt");
lines: org.apache.spark.rdd.RDD[String] = test.txt MapPartitionsRDD[2] at textFile at <console>:23
scala> lines.take(2)
org.apache.hadoop.mapred.InvalidInputException: Input path does not exist: file:/C:/spark/spark-3.4.0-bin-hadoop3/bin/test.txt
at org.apache.hadoop.mapred.FileInputFormat.singleThreadedListStatus(FileInputFormat.java:304)
at org.apache.hadoop.mapred.FileInputFormat.listStatus(FileInputFormat.java:244)
at org.apache.hadoop.mapred.FileInputFormat.getSplits(FileInputFormat.java:332)
at org.apache.spark.rdd.HadoopRDD.getPartitions(HadoopRDD.scala:208)
at org.apache.spark.rdd.RDD.$anonfun$partitions$2(RDD.scala:291)
at scala.Option.getOrElse(Option.scala:189)
at org.apache.spark.rdd.RDD.partitions(RDD.scala:287)
at org.apache.spark.rdd.MapPartitionsRDD.getPartitions(MapPartitionsRDD.scala:49)
at org.apache.spark.rdd.RDD.$anonfun$partitions$2(RDD.scala:291)
at scala.Option.getOrElse(Option.scala:189)
at org.apache.spark.rdd.RDD.partitions(RDD.scala:287)
at org.apache.spark.rdd.RDD.$anonfun$take$1(RDD.scala:1441)
at org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.scala:151)
at org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.scala:112)
at org.apache.spark.rdd.RDD.withScope(RDD.scala:405)
at org.apache.spark.rdd.RDD.take(RDD.scala:1435)
... 47 elided
Caused by: java.io.IOException: Input path does not exist: file:/C:/spark/spark-3.4.0-bin-hadoop3/bin/test.txt
at org.apache.hadoop.mapred.FileInputFormat.singleThreadedListStatus(FileInputFormat.java:278)
... 62 more
scala> val lines = sc.textFile("test.txt");
lines: org.apache.spark.rdd.RDD[String] = test.txt MapPartitionsRDD[4] at textFile at <console>:23
scala> lines.take(2)
res3: Array[String] = Array(I love solving data mining problems., I don?t like solving data mining problems.)
scala> val Length = lines.map(s => s.length)
AISSMS IOIT is the ...
Spark RDD Operati...
Administrator: Co...
ENG
IN
11:43 AM
5/10/2023


Administrator: Command Prompt - spark-shell
Missing Python executable 'python3', defaulting to 'C:\spark\spark-3.4.0-bin-hadoop3\bin\..' for SPARK_HOME environment variable. Please install Python or specify the c
orrect Python executable in PYSPARK_DRIVER_PYTHON or PYSPARK_PYTHON environment variable to detect SPARK_HOME safely.
23/05/10 11:18:14 WARN Shell: Did not find winutils.exe: java.io.FileNotFoundException: java.io.FileNotFoundException: HADOOP_HOME and hadoop.home.dir are unset. -see h
ttps://wiki.apache.org/hadoop/WindowsProblems
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
23/05/10 11:18:24 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Spark context Web UI available at http://DESKTOP-HV799M8:4040
Spark context available as 'sc' (master = local[*], app id = local-1683697711666).
Spark session available as 'spark'.
Welcome to
version 3.4.0
Using Scala version 2.12.17 (OpenJDK 64-Bit Server VM, Java 1.8.0_292)
Type in expressions to have them evaluated.
Type :help for more information.
scala> sp
<console>:23: error: not found: value sp
sp
^
scala>
scala>
scala>
scala> val data = Array(1,2,3,4)
data: Array[Int] = Array(1, 2, 3, 4)
scala> val d = sc.parallelize(data)
d: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[0] at parallelize at <console>:24
scala> d.collect()
res1: Array[Int] = Array(1, 2, 3, 4)
scala> val lines = sc.textFile("test.txt");
lines: org.apache.spark.rdd.RDD[String] = test.txt MapPartitionsRDD[2] at textFile at <console>:23
No new notifications
Local Disk (C:)
AISSMS IOIT is the ...
Spark RDD Opereti...
Administrator: Co...
ENG
IN
11:42 AM
5/10/2023