Assignment No. 1

```
# importing libraries
import pandas as pd
import numpy as np
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

# Loading dataset
df = pd.read_csv("covid_19_clean_complete_2022.csv")
df.drop('Province/State', axis=1, inplace=True)
df.head()

#  data preprocessing - detecting NaN values and using describe() function
df.isna().any()
df.describe()

# shape of dataset (dimensions)
df.shape

# data formatting and normalization
df.dtypes

df['Date'] = pd.to_datetime(df['Date'])
df['Country/Region'] = df['Country/Region'].astype('string')
df.dtypes

#  handling categorical values
# dropping the categorical variable column
# df['new_col'] = df['some_col'].map({'value_1': 1, 'value_2': 2})
df = df.drop(['WHO Region','Country/Region'], axis=1)
df.head()
```

Output

```
Country/Region    False
Lat                True
Long               True
Date              False
Confirmed         False
Deaths            False
Recovered         False
Active            False
```

```
WHO Region          True
dtype: bool
```

	Lat	Long	Confirmed	Deaths	Recovered	Active
count	213348.000000	213348.000000	2.148940e+05	214894.000000	2.148940e+05	2.148940e+05
mean	20.528131	22.735337	4.578132e+05	9310.764693	1.079987e+05	3.405037e+05
std	25.899139	76.304185	2.708770e+06	47497.835275	8.470111e+05	2.516382e+06
min	-71.949900	-178.116500	0.000000e+00	0.000000	0.000000e+00	-1.638280e+05
25%	6.426991	-27.932425	2.530000e+02	2.000000	0.000000e+00	1.600000e+01
50%	22.233350	21.752000	5.223000e+03	71.000000	4.500000e+01	1.243000e+03
75%	41.166070	88.658375	9.892275e+04	1675.000000	5.115750e+03	2.644675e+04
max	71.706900	178.065000	7.925051e+07	958144.000000	3.097475e+07	7.829236e+07

```
Country/Region    object
Lat               float64
Long              float64
Date              object
Confirmed         int64
Deaths            int64
Recovered         int64
Active            int64
WHO Region        object
dtype: object
```

Assignment No. 2

```
# Commented out IPython magic to ensure Python compatibility.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
# %matplotlib inline
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

df = pd.read_csv("Academic-Performance-Dataset.csv")
df

df.shape

df.dtypes

df.isna().sum()
cols_with_na = []
for col in df.columns:
    if df[col].isna().any():
        cols_with_na.append(col)

cols_with_na
for col in cols_with_na:
    col_dt = df[col].dtypes
    if (col_dt == 'int64' or col_dt == 'float64'):
        outliers = (df[col] < 0) | (100 < df[col])
        df.loc[outliers, col] = np.nan
        df[col] = df[col].fillna(df[col].mean())
    else:
        df[col] = df[col].fillna(method='ffill')
df
df['Total
Marks']=df['Phy_marks']+df['Che_marks']+df['EM1_marks']+df['PPS_marks']+df['SME_m
arks']
df['Percentage']=df['Total Marks']/5

df
import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = (9, 6)
df_list = ['Attendence', 'Phy_marks', 'Che_marks', 'EM1_marks', 'PPS_marks',
'SME_marks']
```

```
fig, axes = plt.subplots(2, 3)
fig.set_dpi(120)

count=0
for r in range(2):
    for c in range(3):
        _ = df[df_list[count]].plot(kind = 'box', ax=axes[r,c])
        count+=1
Q1 = df['Che_marks'].quantile(0.25)
Q3 = df['Che_marks'].quantile(0.75)
IQR = Q3 - Q1

Lower_limit = Q1 - 1.5 * IQR
Upper_limit = Q3 + 1.5 * IQR

print(f'Q1 = {Q1}, Q3 = {Q3}, IQR = {IQR}, Lower_limit = {Lower_limit},
Upper_limit = {Upper_limit}')

df[(df['Che_marks'] < Lower_limit) | (df['Che_marks'] > Upper_limit)]
def BinningFunction(column, cut_points, labels = None) :
    break_points=[column.min()] + cut_points + [column.max( )]
    print('Gradding According to percentage \n>60 = F \n60-70 = B \n70-80 =
A\n80-100 = O')
    return pd.cut(column, bins=break_points, labels=labels, include_lowest=True)

cut_points=[60, 70, 80]
labels=['F', 'B', 'A', 'O']
df['Grade']=BinningFunction(df['Percentage'], cut_points, labels)

df
```

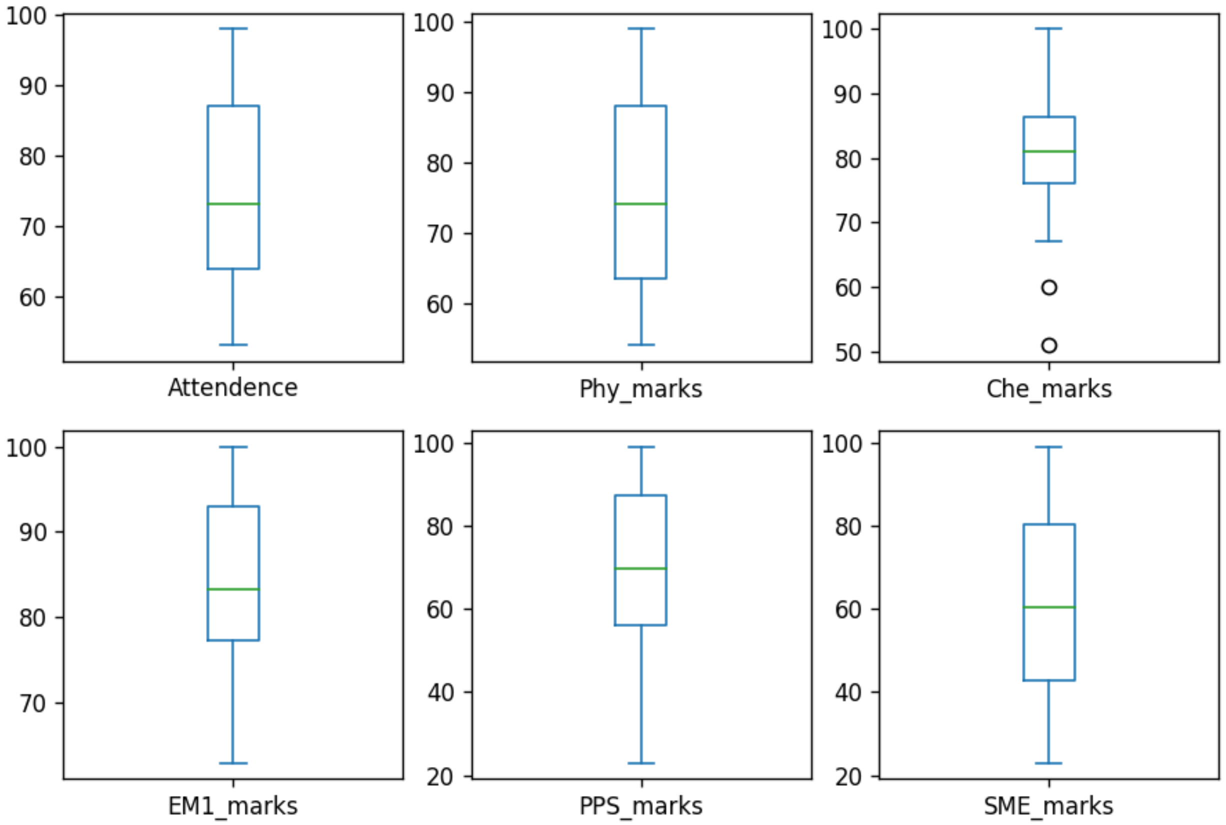
Output

```
(20, 12)
Rollno       int64
Name         object
Gender       object
Branch       object
```

```
Attendence    float64
Phy_marks     float64
Che_marks     float64
EM1_marks     float64
PPS_marks     float64
SME_marks     float64
Total Marks     int64
Percentage    float64
dtype: object
```

```
Rollno        0
Name          2
Gender        0
Branch        0
Attendence    0
Phy_marks     1
Che_marks     3
EM1_marks     2
PPS_marks     1
SME_marks     0
Total Marks   0
Percentage    0
dtype: int64
```

```
['Name', 'Phy_marks', 'Che_marks', 'EM1_marks', 'PPS_marks']
```



```
Q1 = 76.0, Q3 = 86.25, IQR = 10.25, Lower_limit = 60.625, Upper_limit =
101.625
```

```
Rollno  Name    Gender Branch  Attendence       Phy_marks       Che_marks       EM1_marks
        PPS_marks       SME_marks       Total Marks     Percentage

7       8       Ishaan  M       ENTC    75.0    66.0    51.0    83.0    69.611111       76.0
        345.611111      69.122222

14      15      Maryam          F       IT      64.0    87.0    60.0    90.0    65.000000       90.0
        392.000000      78.400000
```

```
Gradding According to percentage
>60 = F
60-70 = B
70-80 = A
80-100 = O
```

Assignment No. 3

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

df = pd.read_csv("iris.csv")
df.head()

'Iris-setosa'
setosa = df['Species'] == 'Iris-setosa'
df[setosa].describe()
'Iris-versicolor'
versicolor = df['Species'] == 'Iris-versicolor'
df[versicolor].describe()
'Iris-virginica'
virginica = df['Species'] == 'Iris-virginica'
df[virginica].describe()

df.dtypes
df.dtypes.value_counts()
```

Output

```
Iris-setosa

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

Iris-versicolor

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        1.800000Iris-virginica
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
```

```
Id              int64
SepalLengthCm   float64
SepalWidthCm    float64
PetalLengthCm   float64
PetalWidthCm    float64
Species         object
dtype: object
float64   4
int64     1
object    1
dtype: int64
```

Assignment No. 4

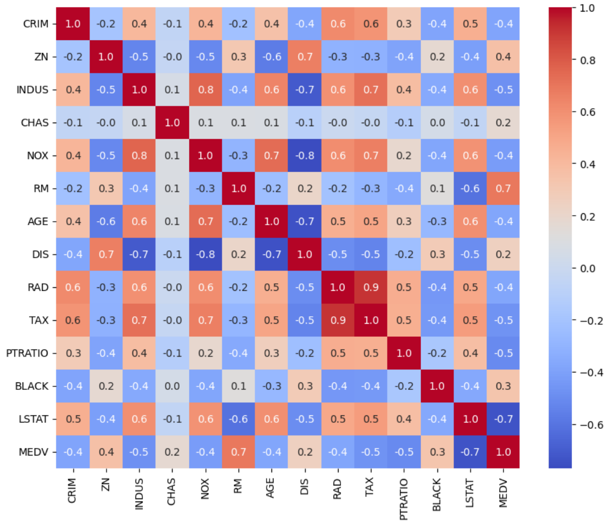
```
# Commented out IPython magic to ensure Python compatibility.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
# %matplotlib inline

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

# Importing DataSet and take a look at Data
Boston = pd.read_csv("boston.csv")
Boston.head()

Boston.info()
Boston.describe()

Boston.plot.scatter('RM', 'MEDV', figsize=(6, 6));

plt.subplots(figsize=(10,8))
sns.heatmap(Boston.corr(), cmap = 'coolwarm', annot = True, fmt = '.1f');

X = Boston[Boston.columns[:-1]]
Y = Boston['MEDV']

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler

# Split DataSet
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3)
sc_X = StandardScaler()
X_train_ = sc_X.fit_transform(X_train)
X_test_ = sc_X.transform(X_test)

print(f'Train Dataset Size - X: {X_train.shape}, Y: {Y_train.shape}')
print(f'Test  Dataset Size - X: {X_test.shape}, Y: {Y_test.shape}')

# Model Building
lm = LinearRegression()
lm.fit(X_train_, Y_train)
predictions = lm.predict(X_test_)
```

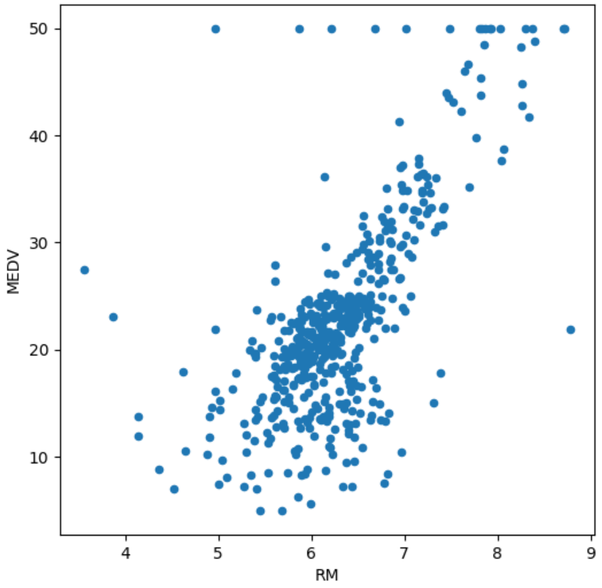
```
# Model Visualization
plt.figure(figsize=(6, 6));
plt.scatter(Y_test, predictions);
plt.xlabel('Y Test');
plt.ylabel('Predicted Y');
plt.title('Test vs Prediction');

plt.figure(figsize=(6, 6));
sns.regplot(x = X_test['RM'], y = predictions, scatter_kws={'s':5});
plt.scatter(X_test['RM'], Y_test, marker = '+');
plt.xlabel('Average number of rooms per dwelling');
plt.ylabel('Median value of owner-occupied homes');
plt.title('Regression Line Tracing');

from sklearn import metrics

print('Mean Absolute Error:', metrics.mean_absolute_error(Y_test, predictions))
print('Mean Square Error:', metrics.mean_squared_error(Y_test, predictions))
print('Root Mean Square Error:', np.sqrt(metrics.mean_squared_error(Y_test,
predictions)))

# Model Coefficients
coefficients = pd.DataFrame(lm.coef_.round(2), X.columns)
coefficients.columns = ['Coefficients']
coefficients
```

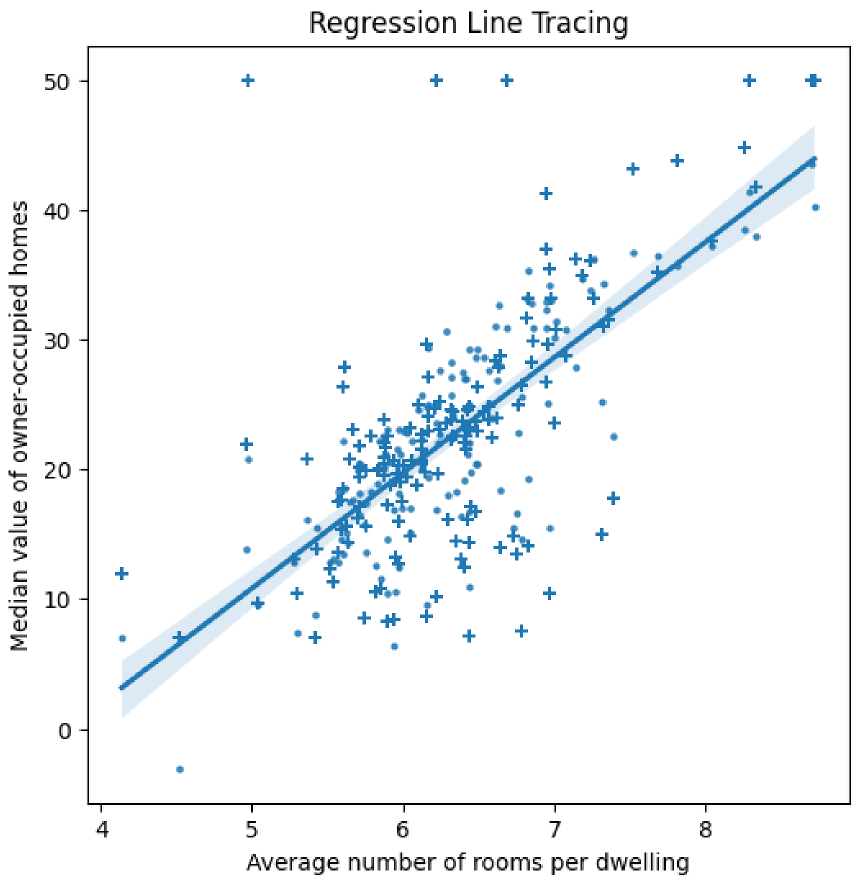
Output

```
dtypes: float64(12), int64(2)

memory usage: 55.5 KB

CRIM  ZN       INDUS        CHAS NOX    RM      AGE    DIS     RAD    TAX     PTRATIO
      BLACK           LSTAT MEDV

count 506.000000   506.000000   506.000000   506.000000   506.000000   506.000000
      506.000000   506.000000   506.000000   506.000000   506.000000   506.000000
      506.000000   506.000000

mean  3.613524     11.363636    11.136779    0.069170     0.554695     6.284634
      68.574901    3.795043     9.549407     408.237154   18.455534    356.674032
      12.653063    22.532806
```

```
std    8.601545     23.322453    6.860353     0.253994     0.115878     0.702617
       28.148861    2.105710     8.707259     168.537116   2.164946     91.294864
       7.141062     9.197104

min    0.006320     0.000000     0.460000     0.000000     0.385000     3.561000
       2.900000     1.129600     1.000000     187.000000   12.600000    0.320000
       1.730000     5.000000

25%    0.082045     0.000000     5.190000     0.000000     0.449000     5.885500
       45.025000    2.100175     4.000000     279.000000   17.400000    375.377500
       6.950000     17.025000

50%    0.256510     0.000000     9.690000     0.000000     0.538000     6.208500
       77.500000    3.207450     5.000000     330.000000   19.050000    391.440000
       11.360000    21.200000

75%    3.677083     12.500000    18.100000    0.000000     0.624000     6.623500
       94.075000    5.188425     24.000000    666.000000   20.200000    396.225000
       16.955000    25.000000

max    88.976200    100.000000   27.740000    1.000000     0.871000     8.780000
       100.000000   12.126500    24.000000    711.000000   22.000000    396.900000
       37.970000    50.000000
```






Regression Line Tracing
Median value of owner-occupied homes
Average number of rooms per dwelling
0
10
20
30
40
50
4
5
6
7
8

Assignment No. 5

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

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

df = pd.read_csv('Social_Network_Ads.csv')
df.head()
df.info()
df.describe()

X = df[['Age', 'EstimatedSalary']]
Y = df['Purchased']

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25,
random_state = 0)
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)

print(f'Train Dataset Size - X: {X_train.shape}, Y: {Y_train.shape}')
print(f'Test  Dataset Size - X: {X_test.shape}, Y: {Y_test.shape}')

from sklearn.linear_model import LogisticRegression

lm = LogisticRegression(random_state = 0, solver='lbfgs' )
lm.fit(X_train, Y_train)
predictions = lm.predict(X_test)

plt.figure(figsize=(6, 6));
sns.regplot(x = X_test[:, 1], y = predictions, scatter_kws={'s':5});
plt.scatter(X_test[:, 1], Y_test, marker = '+');
plt.xlabel("User's Estimated Salary");
plt.ylabel('Ads Purchased');
plt.title('Regression Line Tracing');
```

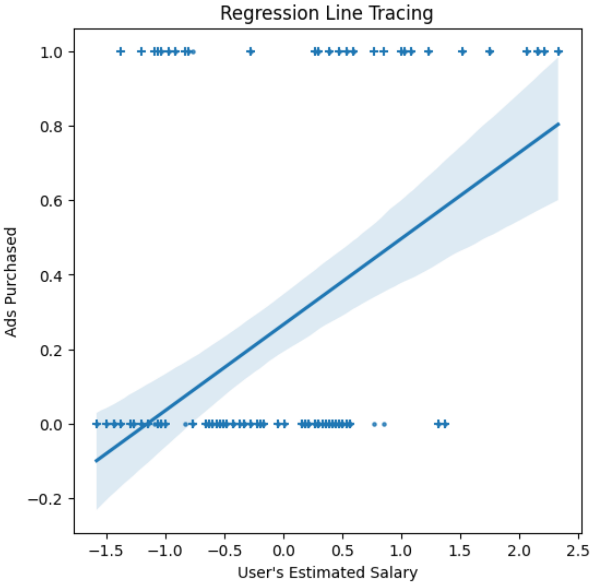
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

cm = confusion_matrix(Y_test, predictions)
print(f'''Confusion matrix :\n
               | Positive Prediction\t| Negative Prediction
---------------+------------------------+----------------------
Positive Class | True Positive (TP) {cm[0, 0]}\t| False Negative (FN) {cm[0, 1]}
---------------+------------------------+----------------------
Negative Class | False Positive (FP) {cm[1, 0]}\t| True Negative (TN) {cm[1,
1]}\n\n''')

cm = classification_report(Y_test, predictions)
print('Classification report : \n', cm)

# Visualizing the Training set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, Y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:,
0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:,
1].max() + 1, step = 0.01))

plt.figure(figsize=(9, 7.5));
plt.contourf(X1, X2, lm.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.6, cmap = ListedColormap(('red', 'green')));
plt.xlim(X1.min(), X1.max());
plt.ylim(X2.min(), X2.max());
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                color = ListedColormap(('red', 'green'))(i), label = j);
plt.title('Logistic Regression (Training set)');
plt.xlabel('Age');
plt.ylabel('Estimated Salary');
plt.legend();
plt.show();

# Visualizing the Test set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, Y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:,
0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:,
1].max() + 1, step = 0.01))
```

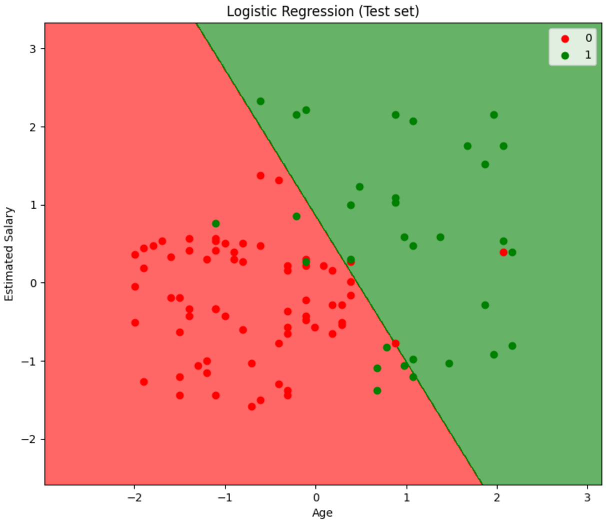
```
plt.figure(figsize=(9, 7.5));
plt.contourf(X1, X2, lm.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.6, cmap = ListedColormap(('red', 'green')));
plt.xlim(X1.min(), X1.max());
plt.ylim(X2.min(), X2.max());
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                color = ListedColormap(('red', 'green'))(i), label = j);
plt.title('Logistic Regression (Test set)');
plt.xlabel('Age');
plt.ylabel('Estimated Salary');
plt.legend();
plt.show();
```

Output

```
User ID  Gender   Age      EstimatedSalary   Purchased

0        15624510        Male      19.0      19000.0 0

1        15810944        Male      35.0      20000.0 0

2        15668575        Female    26.0      43000.0 0

3        15603246        Female    27.0      57000.0 0

4        15804002        Male      19.0      76000.0 0

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 400 entries, 0 to 399

Data columns (total 5 columns):

 #   Column          Non-Null Count  Dtype

---  ------          --------------  -----

 0   User ID         400 non-null    int64

 1   Gender          400 non-null    object

 2   Age             400 non-null    float64

 3   EstimatedSalary  400 non-null    float64

 4   Purchased       400 non-null    int64

dtypes: float64(2), int64(2), object(1)

memory usage: 15.8+ KB

User ID  Age       EstimatedSalary   Purchased
```

count	4.000000e+02	400.000000	400.000000	400.000000
mean	1.569154e+07	37.655000	69742.500000	0.357500
std	7.165832e+04	10.482877	34096.960282	0.479864
min	1.556669e+07	18.000000	15000.000000	0.000000
25%	1.562676e+07	29.750000	43000.000000	0.000000
50%	1.569434e+07	37.000000	70000.000000	0.000000
75%	1.575036e+07	46.000000	88000.000000	1.000000
max	1.581524e+07	60.000000	150000.000000	1.000000





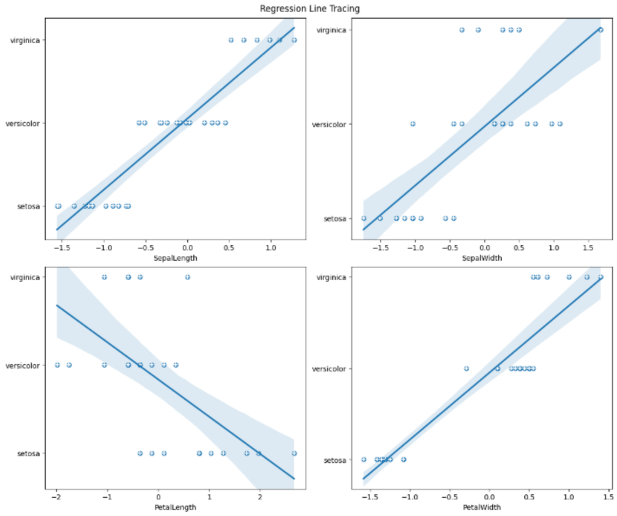
Assignment No. 6

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

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

df = pd.read_csv('iris.csv')
df.head()

X = df.iloc[:, :4].values
Y = df['Species'].values

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2,
random_state = 0)
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)

print(f'Train Dataset Size - X: {X_train.shape}, Y: {Y_train.shape}')
print(f'Test  Dataset Size - X: {X_test.shape}, Y: {Y_test.shape}')

from sklearn.naive_bayes import GaussianNB

classifier = GaussianNB()
classifier.fit(X_train, Y_train)
predictions = classifier.predict(X_test)

mapper = {'setosa': 0, 'versicolor': 1, 'virginica': 2}
predictions_ = [mapper[i] for i in predictions]

fig, axs = plt.subplots(2, 2, figsize = (12, 10), constrained_layout = True);
_ = fig.suptitle('Regression Line Tracing')

for i in range(4):
    x, y = i // 2, i % 2
    _ = sns.regplot(x = X_test[:, i], y = predictions_, ax=axs[x, y])
    _ = axs[x, y].scatter(X_test[:, i][::-1], Y_test[::-1], marker = '+',
color="white")
```

```
    _ = axs[x, y].set_xlabel(df.columns[i + 1][:-2])

from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

cm = confusion_matrix(Y_test, predictions)
print(f'''Confusion matrix :\n
               | Positive Prediction\t| Negative Prediction
---------------+------------------------+----------------------
Positive Class | True Positive (TP) {cm[0, 0]}\t| False Negative (FN) {cm[0, 1]}
---------------+------------------------+----------------------
Negative Class | False Positive (FP) {cm[1, 0]}\t| True Negative (TN) {cm[1,
1]}\n\n''')

cm = classification_report(Y_test, predictions)
print('Classification report : \n', cm)
```

Output

```
Id      SepalLengthCm   SepalWidthCm    PetalLengthCm   PetalWidthCm    Species

0       1       5.1     3.5     1.4     0.2     setosa

1       2       4.9     3.0     1.4     0.2     setosa

2       3       4.7     3.2     1.3     0.2     setosa

3       4       4.6     3.1     1.5     0.2     setosa

4       5       5.0     3.6     1.4     0.2     setosa
```



Assignment No. 7

```
# import required module
from sklearn.feature_extraction.text import TfidfVectorizer

# assign documents
d0 = 'Just Code It'
d1 = 'Code'
d2 = 'Programming'

# merge documents into a single corpus
string = [d0, d1, d2]

# create object
tfidf = TfidfVectorizer()

# get tf-df values
result = tfidf.fit_transform(string)

# get idf values
print('\nidf values:')
for ele1, ele2 in zip(tfidf.get_feature_names(), tfidf.idf_):
    print(ele1, ':', ele2)

# get indexing
print('\nWord indexes:')
print(tfidf.vocabulary_)

# display tf-idf values
print('\ntf-idf value:')
print(result)

# in matrix form
print('\ntf-idf values in matrix form:')
print(result.toarray())
```

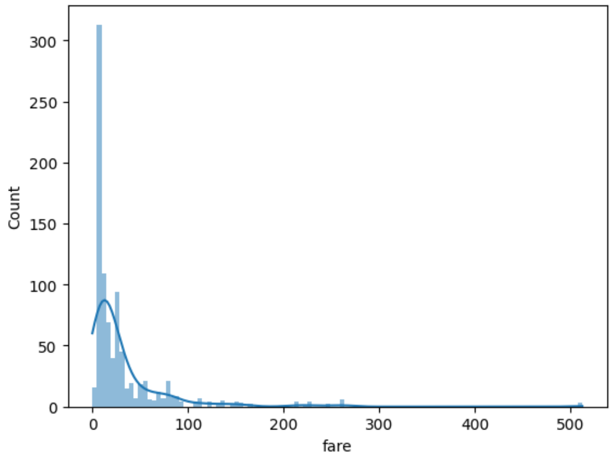
Output:-

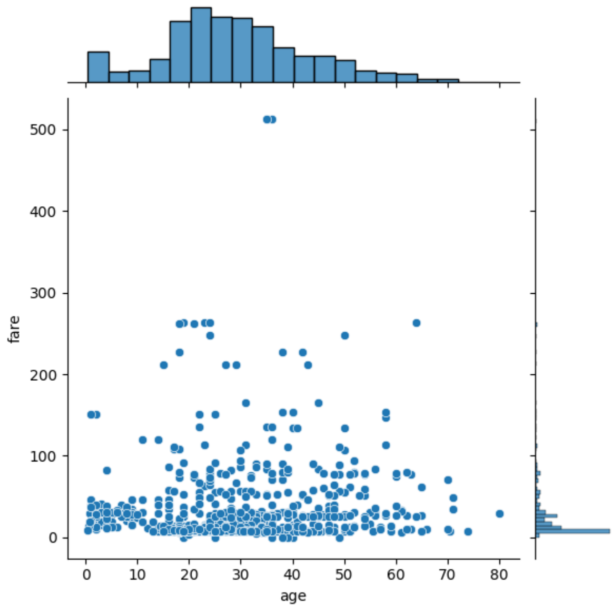
```
Word indexes:
{'just': 2, 'code': 0, 'it': 1, 'programming': 3}
tf-idf value:
  (0, 1)	0.6227660078332259
  (0, 0)	0.4736296010332684
  (0, 2)	0.6227660078332259
  (1, 0)	1.0
  (2, 3)	1.0

tf-idf values in matrix form:
[[0.4736296  0.62276601 0.62276601 0.        ]
 [1.         0.         0.         0.        ]
 [0.         0.         0.         1.        ]]
```

Assignment No. 8

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

dataset = sns.load_dataset('titanic')

dataset.head()

sns.histplot(dataset['fare'], kde=True, linewidth=0);

sns.jointplot(x='age', y='fare', data=dataset);
```

Output:



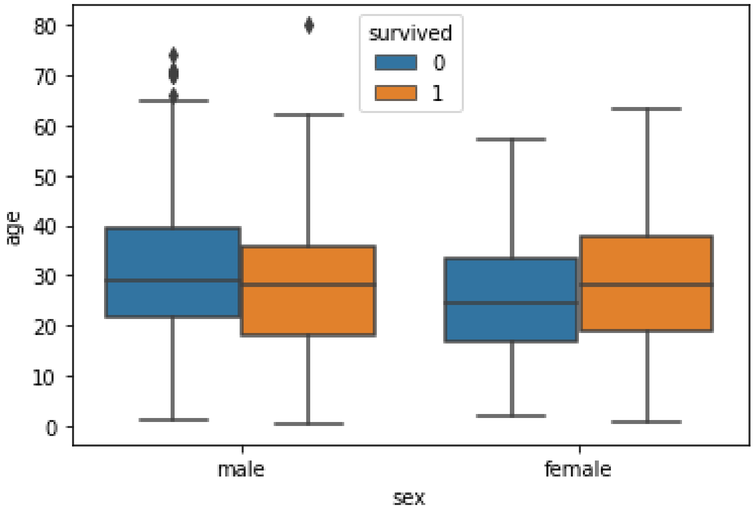


Assignment No. 9

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

dataset = sns.load_dataset('titanic')

dataset.head()

sns.boxplot(x='sex', y='age', data=dataset, hue="survived");
```

Output

```
survived  pclass    sex       age     sibsp   parch   fare      embarkedclass  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

4         0         3         male    35.0    0       0         8.0500  S      Third   man      True     NaN
          Southampton         no      True
```



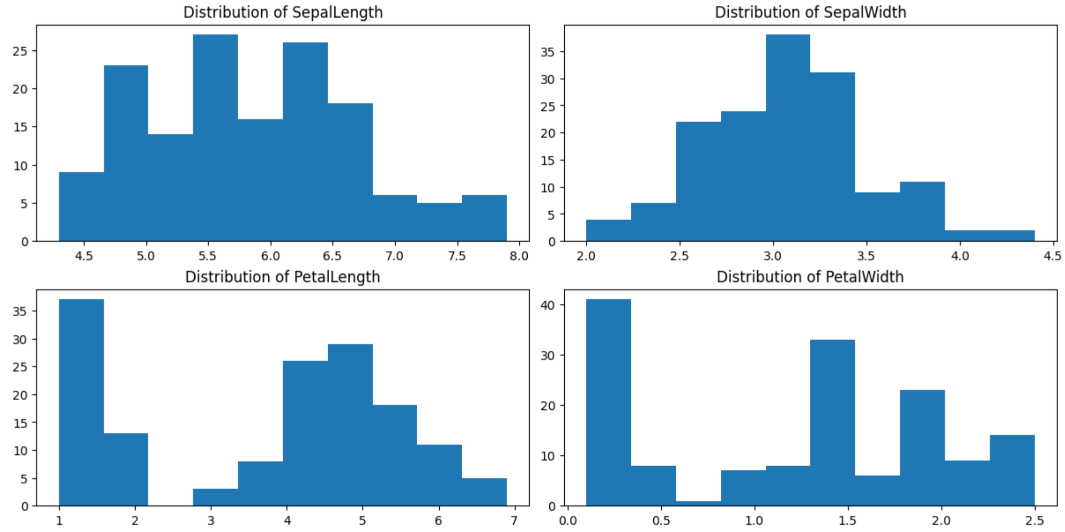
Assignment No. 10

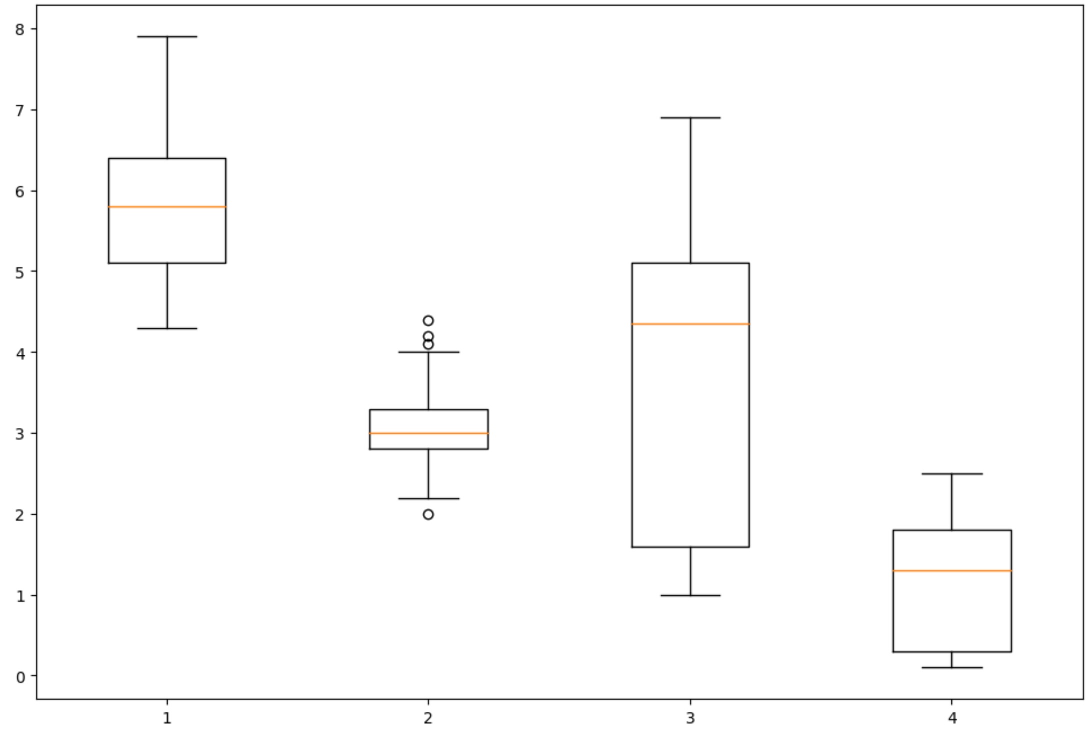
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

df = pd.read_csv('iris.csv')
df.head()

df.info()

np.unique(df["Species"])

df.describe()

import seaborn as sns
import matplotlib
import matplotlib.pyplot as plt

fig, axes = plt.subplots(2, 2, figsize=(12, 6), constrained_layout = True)

for i in range(4):
    x, y = i // 2, i % 2
    _ = axes[x, y].hist(df[df.columns[i + 1]])
    _ = axes[x, y].set_title(f"Distribution of {df.columns[i + 1][:-2]}")

data_to_plot = df[df.columns[1:-1]]

fig, axes = plt.subplots(1, figsize=(12,8))
bp = axes.boxplot(data_to_plot)
```

Output

```
Id      SepalLengthCm    SepalWidthCm    PetalLengthCm    PetalWidthCm    Species

0       1       5.1     3.5     1.4     0.2     setosa

1       2       4.9     3.0     1.4     0.2     setosa

2       3       4.7     3.2     1.3     0.2     setosa

3       4       4.6     3.1     1.5     0.2     setosa

4       5       5.0     3.6     1.4     0.2     setosa
```

```
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
```





```
// WC_Runner.java
package com.wc;
import java.io.IOException;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.TextInputFormat;
import org.apache.hadoop.mapred.TextOutputFormat;
public class WC_Runner {
public static void main(String[] args) throws IOException {
JobConf conf = new JobConf(WC_Runner.class);
conf.setJobName("WordCount");
conf.setOutputKeyClass(Text.class);
conf.setOutputValueClass(IntWritable.class);
conf.setMapperClass(WC_Mapper.class);
conf.setCombinerClass(WC_Reducer.class);
conf.setReducerClass(WC_Reducer.class);
conf.setInputFormat(TextInputFormat.class);
conf.setOutputFormat(TextOutputFormat.class);
FileInputFormat.setInputPaths(conf,new Path(args[0]));
FileOutputFormat.setOutputPath(conf,new Path(args[1]));
JobClient.runJob(conf);
}
}
```

Assignment No. 11

```
// WC_Mapper.java
package com.wc;
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;
public class WC_Mapper extends MapReduceBase implements
Mapper<LongWritable,Text,Text,IntWritable>{
private final static IntWritable one = new IntWritable(1);
private Text word = new Text();
public void map(
LongWritable key,
Text value,
OutputCollector<Text,IntWritable> output,
Reporter reporter
) throws IOException {
String line = value.toString();
StringTokenizer tokenizer = new StringTokenizer(line);
while (tokenizer.hasMoreTokens()){
word.set(tokenizer.nextToken());
output.collect(word, one);
}
}
}
// WC_Reducer.java
package com.wc;
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;
public class WC_Reducer extends MapReduceBase implements
Reducer<Text,IntWritable,Text,IntWritable> {
public void reduce(
Text key,
Iterator<IntWritable> values,
OutputCollector<Text,IntWritable> output,
Reporter reporter
) throws IOException {
int sum=0;
while (values.hasNext()) {
sum += values.next().get();
}
output.collect(key,new IntWritable(sum));
}
}
```

Input:

```
HDFS is a storage unit of Hadoop
MapReduce is a processing tool for Hadoop
```

Output:

```
HDFS 1
Hadoop 2
MapReduce 1
a 2
for 1
is 2
of 1
processing 1
storage 1
tool 1
unit 1
```

Assignment No. 12

```
// SalesCountryRunner.java
package SalesCountry;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
public class SalesCountryRunner {
public static void main(String[] args) {
JobClient my_client = new JobClient();
// Create a configuration object for the job
JobConf job_conf = new JobConf(SalesCountryDriver.class);
// Set a name of the Job
job_conf.setJobName("SalePerCountry");
// Specify data type of output key and value
job_conf.setOutputKeyClass(Text.class);
job_conf.setOutputValueClass(IntWritable.class);
// Specify names of Mapper and Reducer Class
job_conf.setMapperClass(SalesCountry.SalesMapper.class);
job_conf.setReducerClass(SalesCountry.SalesCountryReducer.class);
// Specify formats of the data type of Input and output
job_conf.setInputFormat(TextInputFormat.class);
job_conf.setOutputFormat(TextOutputFormat.class);
// Set input and output directories using command line arguments,
//arg[0] = name of input directory on HDFS, and arg[1] = name of output
directory to be created to store the output file.
FileInputFormat.setInputPaths(job_conf, new Path(args[0]));
FileOutputFormat.setOutputPath(job_conf, new Path(args[1]));
my_client.setConf(job_conf);
try {
// Run the job
JobClient.runJob(job_conf);
} catch (Exception e) {
e.printStackTrace();
}
}
}
// SalesMapper.java
package SalesCountry;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;
public class SalesMapper extends MapReduceBase implements Mapper<LongWritable, Text, Text,
IntWritable> {
private final static IntWritable one = new IntWritable(1);
public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable>
output, Reporter reporter) throws IOException {
String valueString = value.toString();
String[] SingleCountryData = valueString.split(",");
output.collect(new Text(SingleCountryData[7]), one);
}
}
// SalesCountryReducer.java
package SalesCountry;
```

```
import java.io.IOException;
import java.util.*;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;
public class SalesCountryReducer extends MapReduceBase implements Reducer<Text, IntWritable,
Text, IntWritable> {
public void reduce(Text t_key, Iterator<IntWritable> values,
OutputCollector<Text,IntWritable> output, Reporter reporter) throws IOException {
Text key = t_key;
int frequencyForCountry = 0;
while (values.hasNext()) {
// replace type of value with the actual type of our value
IntWritable value = (IntWritable) values.next();
frequencyForCountry += value.get();
}
output.collect(key, new IntWritable(frequencyForCountry));
}
}
```

Output:

```
Argentina 1
Australia 38
Austria 7
Bahrain 1
Belgium 8
Bermuda 1
Brazil 5
Bulgaria 1
```

CO 1

Canada 76

Cayman Isls 1

China 1

Costa Rica 1

Country 1

Czech Republic 3

Denmark 15

Dominican Republic 1

Finland 2

France 27

Germany 25

Greece 1

Guatemala 1

Hong Kong 1

Hungary 3

Iceland 1

India 2

Ireland 49

Israel 1

Italy 15

Japan 2

Jersey 1

Kuwait 1

Latvia 1

Luxembourg 1

Malaysia 1

Malta 2

Mauritius 1

Moldova 1

Monaco 2

Netherlands 22

New Zealand 6

Norway 16

Philippines 2

Poland 2

Romania 1

Russia 1

South Africa 5

South Korea 1

Spain 12

Sweden 13

Switzerland 36

Thailand 2

The Bahamas 2

Turkey 6

Ukraine 1

United Arab Emirates 6

United Kingdom 100

United States 462

Assignment No. 13

```
// MaxTemperatureDriver.java
package MaxMinTemp;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;
public class MaxTemperatureDriver extends Configured implements Tool{
public int run(String[] args) throws Exception {
if(args.length !=2) {
System.err.println("Usage: MaxTemperatureDriver <input path>
<outputpath>");
System.exit(-1);
}
Job job = new Job();
job.setJarByClass(MaxTemperatureDriver.class);
job.setJobName("Max Temperature");
FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job,new Path(args[1]));
job.setMapperClass(MaxTemperatureMapper.class);
job.setReducerClass(MaxTemperatureReducer.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
System.exit(job.waitForCompletion(true) ? 0:1);
boolean success = job.waitForCompletion(true);
return success ? 0 : 1;
}
public static void main(String[] args) throws Exception {
MaxTemperatureDriver driver = new MaxTemperatureDriver();
int exitCode = ToolRunner.run(driver, args);
System.exit(exitCode);
}
}
// MaxTemperatureMapper.java
package MaxMinTemp;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
public class MaxTemperatureMapper extends Mapper<LongWritable, Text, Text,
IntWritable> {
private static final int MISSING = 9999;
@Override
public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
String line = value.toString();
String year = line.substring(15, 19);
int airTemperature;
if (line.charAt(87) == '+') { // parseInt doesn't like leading plus signs
airTemperature = Integer.parseInt(line.substring(88, 92));
} else {
airTemperature = Integer.parseInt(line.substring(87, 92));
}
String quality = line.substring(92, 93);
if (airTemperature != MISSING && quality.matches("[01459]")) {
context.write(new Text(year), new IntWritable(airTemperature));
}
}
}
// MaxTemperatureReducer.java
package MaxMinTemp;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
public class MaxTemperatureReducer extends Reducer<Text, IntWritable, Text,
IntWritable> {
@Override
public void reduce(Text key, Iterable<IntWritable> values, Context context)
throws IOException, InterruptedException {
int maxValue = Integer.MIN_VALUE;
for (IntWritable value : values) {
maxValue = Math.max(maxValue, value.get());
}
context.write(key, new IntWritable(maxValue));
}
}
```

OUTPUT:

```
1901 317
1902 244
1903 289
1904 256
1905 283
1906 294
1907 283
1908 289
1909 278
1910 294
1911 306
1912 322
1913 300
1914 333
1915 294
1916 278
1917 317
1918 322
1919 378
1920 294
```