Practical-9

Aim :

1. Identify the column(s) of a given DataFrame which have at least one missing value, count the number of missing values in each column and drop the raws and columns with missing values. Check for the null values. Also remove the duplicate values from the DataFrame. Handle outliers in the Data Frame.
2. Access subset of data through indexing (Select data using labels (column headings)), Slicing(Extract range based subset, subset of rows, subset of columns, select a subset of rows and columns from our DataFrame using iloc method).
3. Perform other data processing on a given dataset.

```
import pandas as pd
import numpy as np
print("12002040701067")
pd.set_option('display.max_rows', None)
#pd.set_option('display.max_columns', None)
df = pd.DataFrame({
'ord_no':[70001,np.nan,70002,70004,np.nan,70005,np.nan,70010,70003,70012,np.nan,70013],
'purch_amt':[150.5,270.65,65.26,110.5,948.5,2400.6,5760,1983.43,2480.4,250.45, 75.29,3045.6],
'ord_date': ['2012-10-05','2012-09-10',np.nan,'2012-08-17','2012-09-10','2012-07-27','2012-09-1
'customer_id':[3002,3001,3001,3003,3002,3001,3001,3004,3003,3002,3001,3001],
'salesman_id':[5002,5003,5001,np.nan,5002,5001,5001,np.nan,5003,5002,5003,np.nan]})
print("Original Orders DataFrame:")
print(df)
print("\nIdentify the columns which have at least one missing value:")
print(df.isna().any())

print("\nNumber of missing values of the said dataframe:")
print(df.isna().sum())
```

```
12002040701067
Original Orders DataFrame:
     ord_no  purch_amt    ord_date  customer_id  salesman_id
0   70001.0     150.50  2012-10-05         3002       5002.0
1       NaN     270.65  2012-09-10         3001       5003.0
2   70002.0      65.26         NaN         3001       5001.0
3   70004.0     110.50  2012-08-17         3003          NaN
4       NaN     948.50  2012-09-10         3002       5002.0
5   70005.0    2400.60  2012-07-27         3001       5001.0
6       NaN    5760.00  2012-09-10         3001       5001.0
7   70010.0    1983.43  2012-10-10         3004          NaN
8   70003.0    2480.40  2012-10-10         3003       5003.0
9   70012.0     250.45  2012-06-27         3002       5002.0
10      NaN      75.29  2012-08-17         3001       5003.0
11  70013.0    3045.60  2012-04-25         3001          NaN

Identify the columns which have at least one missing value:
ord_no          True
purch_amt      False
```

```
ord_date        True
customer_id    False
salesman_id     True
dtype: bool

Number of missing values of the said dataframe:
ord_no         4
purch_amt      0
ord_date       1
customer_id    0
salesman_id    3
dtype: int64
```

```
print("\nDrop the rows where at least one element is missing:")
result = df.dropna()
print(result)

print("\nDrop the columns where at least one element is missing:")
result = df.dropna(axis='columns')
print(result)

# using isnull() function
df.isnull()
print(df.drop_duplicates())
```

```
Drop the rows where at least one element is missing:
    ord_no  purch_amt    ord_date  customer_id  salesman_id
0  70001.0     150.50  2012-10-05         3002       5002.0
5  70005.0    2400.60  2012-07-27         3001       5001.0
8  70003.0    2480.40  2012-10-10         3003       5003.0
9  70012.0     250.45  2012-06-27         3002       5002.0

Drop the columns where at least one element is missing:
    purch_amt  customer_id
0      150.50         3002
1      270.65         3001
2       65.26         3001
3      110.50         3003
4      948.50         3002
5     2400.60         3001
6     5760.00         3001
7     1983.43         3004
8     2480.40         3003
9      250.45         3002
10      75.29         3001
11    3045.60         3001
     ord_no  purch_amt    ord_date  customer_id  salesman_id
0   70001.0     150.50  2012-10-05         3002       5002.0
1       NaN     270.65  2012-09-10         3001       5003.0
2   70002.0      65.26         NaN         3001       5001.0
3   70004.0     110.50  2012-08-17         3003          NaN
4       NaN     948.50  2012-09-10         3002       5002.0
5   70005.0    2400.60  2012-07-27         3001       5001.0
6       NaN    5760.00  2012-09-10         3001       5001.0
7   70010.0    1983.43  2012-10-10         3004          NaN
8   70003.0    2480.40  2012-10-10         3003       5003.0
9   70012.0     250.45  2012-06-27         3002       5002.0
10      NaN      75.29  2012-08-17         3001       5003.0
11  70013.0    3045.60  2012-04-25         3001          NaN
```

```
sample= [15, 101, 18, 7, 13, 16, 11, 21, 5, 15, 10, 9]
outliers = []
def detect_outliers_iqr(data):
    data = sorted(data)
    q1 = np.percentile(data, 25)
    q3 = np.percentile(data, 75)
    # print(q1, q3)
    IQR = q3-q1
    lwr_bound = q1-(1.5*IQR)
    upr_bound = q3+(1.5*IQR)
    # print(lwr_bound, upr_bound)
    for i in data:
        if (i<lwr_bound or i>upr_bound):
            outliers.append(i)
    return outliers# Driver code
sample_outliers = detect_outliers_iqr(sample)
print("Outliers from IQR method: ", sample_outliers)
# Trimming
for i in sample_outliers:
    a = np.delete(sample, np.where(sample==i))
print(a)
```

```
Outliers from IQR method:  [101]
[ 15 101  18   7  13  16  11  21   5  15  10   9]
```

```
# TIP: use the .head() method we saw earlier to make output shorter
# Method 1: select a 'subset' of the data using the column name
print(df['customer_id'])

# Select rows 0, 1, 2 (row 3 is not selected)
print(df[0:3])

# Select columns
print(df.iloc[0:3, 1:4])
```

```
0     3002
1     3001
2     3001
3     3003
4     3002
5     3001
6     3001
7     3004
8     3003
9     3002
10    3001
11    3001
Name: customer_id, dtype: int64
    ord_no  purch_amt    ord_date  customer_id  salesman_id
0  70001.0     150.50  2012-10-05         3002       5002.0
1      NaN     270.65  2012-09-10         3001       5003.0
2  70002.0      65.26         NaN         3001       5001.0
   purch_amt    ord_date  customer_id
0     150.50  2012-10-05         3002
1     270.65  2012-09-10         3001
2      65.26         NaN         3001
```

```
# importing libraries
```

```
import pandas
import scipy
import numpy
from sklearn.preprocessing import MinMaxScaler

pd.set_option('display.max_rows', None)
#pd.set_option('display.max_columns', None)
dataframe = pd.DataFrame({
'ord_no':[70001,np.nan,70002,70004,np.nan,70005,np.nan,70010,70003,70012,np.nan,70013],
'purch_amt':[150.5,270.65,65.26,110.5,948.5,2400.6,5760,1983.43,2480.4,250.45, 75.29,3045.6],
'customer_id':[3002,3001,3001,3003,3002,3001,3001,3004,3003,3002,3001,3001],
'salesman_id':[5002,5003,5001,np.nan,5002,5001,5001,np.nan,5003,5002,5003,np.nan]})
array = dataframe.values

# separate array into input and output components
X = array[:,0:3]
Y = array[:,3]

# initialising the MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))
# learning the statistical parameters for each of the data and transforming
rescaledX = scaler.fit_transform(X)

# summarize transformed data
numpy.set_printoptions(precision=3)
print(rescaledX[0:5,:])
```

```
[[0.    0.015 0.333]
 [  nan 0.036 0.   ]
 [0.083 0.    0.   ]
 [0.25  0.008 0.667]
 [  nan 0.155 0.333]]
```