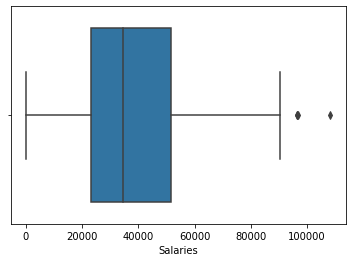
**Practical No. 02**

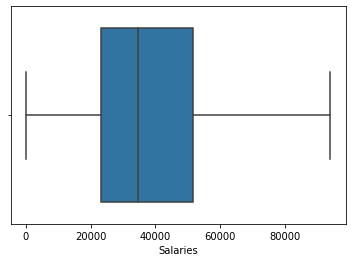
**Code:**

## Type casting #################  
import pandas as pd  
  
data = pd.read\_csv("C:/Data/ethnic diversity.csv")  
data.dtypes  
  
help(data.astype)  
# Now we will convert 'float64' into 'int64' type.  
data.Salaries = data.Salaries.astype('int64')  
data.dtypes  
  
data.age = data.age.astype('float32')  
data.dtypes  
  
### Identify duplicates records in the data ###  
data = pd.read\_csv("C:/Data/mtcars\_dup.csv")  
  
duplicate = data.duplicated()  
duplicate  
sum(duplicate)  
  
# Removing Duplicates  
data1 = data.drop\_duplicates()  
  
############## Outlier Treatment ###############  
import pandas as pd  
import numpy as np  
import seaborn as sns  
  
df = pd.read\_csv("C:/Data/ethnic diversity.csv")  
df.dtypes  
  
# let's find outliers in Salaries  
sns.boxplot(df.Salaries)  
  
sns.boxplot(df.age)  
# No outliers in age column  
  
# Detection of outliers (find limits for salary based on IQR)  
IQR = df['Salaries'].quantile(0.75) - df['Salaries'].quantile(0.25)  
print(IQR)  
lower\_limit = df['Salaries'].quantile(0.25) - (IQR \* 1.5)  
upper\_limit = df['Salaries'].quantile(0.75) + (IQR \* 1.5)  
  
#### ########### 1. Remove (let's trim the dataset) ################  
# Trimming Technique  
# let's flag the outliers in the data setc  
outliers\_df = np.where(df['Salaries'] > upper\_limit, True, np.where(df['Salaries'] < lower\_limit, True, False))  
print(outliers\_df)  
sum(outliers\_df)  
df\_trimmed = df.loc[~(outliers\_df), ]  
df.shape, df\_trimmed.shape  
  
# let's explore outliers in the trimmed dataset  
sns.boxplot(df\_trimmed.Salaries)  
# we see no outiers  
  
############### 2.Replace ###############  
# Now let's replace the outliers by the maximum and minimum limit  
df['df\_replaced'] = pd.DataFrame(np.where(df['Salaries'] > upper\_limit, upper\_limit, np.where(df['Salaries'] < lower\_limit, lower\_limit, df['Salaries'])))  
sns.boxplot(df.df\_replaced)  
  
############### 3. Winsorization ###############  
# pip install feature\_engine   # install the package  
from feature\_engine.outliers import Winsorizer  
winsor = Winsorizer(capping\_method='iqr', # choose  IQR rule boundaries or gaussian for mean and std  
                          tail='both', # cap left, right or both tails  
                          fold=1.5,  
                          variables=['Salaries'])  
  
df\_t = winsor.fit\_transform(df[['Salaries']])  
  
# we can inspect the minimum caps and maximum caps  
# winsor.left\_tail\_caps\_, winsor.right\_tail\_caps\_  
  
# lets see boxplot  
sns.boxplot(df\_t.Salaries)  
  
#### zero variance and near zero variance ######  
  
# If the variance is low or close to zero, then a feature is approximately  
# constant and will not improve the performance of the model.  
# In that case, it should be removed.  
  
df.var() # variance of numeric variables  
  
#################### Missing Values Imputation ################################  
import numpy as np  
import pandas as pd  
  
# load the dataset  
# use modified ethnic dataset  
df = pd.read\_csv('C:/Data/modified ethnic.csv') # for doing modifications  
  
# check for count of NA'sin each column  
df. sum()  
  
# Create an imputer object that fills 'Nan' values  
# Mean and Median imputer are used for numeric data (Salaries)  
# Mode is used for discrete data (ex: Position, Sex, MaritalDesc)  
  
# for Mean, Meadian, Mode imputation we can use Simple Imputer or df.fillna()  
from sklearn.impute import SimpleImputer  
  
# Mean Imputer  
mean\_imputer = SimpleImputer(missing\_values=np.nan, strategy='mean')  
df["Salaries"] = pd.DataFrame(mean\_imputer.fit\_transform(df[["Salaries"]]))  
df["Salaries"].isna().sum()  
  
# Median Imputer  
median\_imputer = SimpleImputer(missing\_values=np.nan, strategy='median')  
df["age"] = pd.DataFrame(median\_imputer.fit\_transform(df[["age"]]))  
df["age"].isna().sum()  # all 2 records replaced by median  
  
df.isna().sum()  
  
# Mode Imputer  
mode\_imputer = SimpleImputer(missing\_values=np.nan, strategy='most\_frequent')  
df["Sex"] = pd.DataFrame(mode\_imputer.fit\_transform(df[["Sex"]]))  
df["MaritalDesc"] = pd.DataFrame(mode\_imputer.fit\_transform(df[["MaritalDesc"]]))  
df.isnull().sum()  # all Sex, MaritalDesc records replaced by mode  
  
###########  
# Discritization  
  
import pandas as pd  
data = pd.read\_csv("C:/Data/ethnic diversity.csv")  
data.head()  
data.describe()  
data['Salaries\_new'] = pd.cut(data['Salaries'], bins=[min(data.Salaries) - 1,  
                                                  data.Salaries.mean(), max(data.Salaries)], labels=["Low","High"])  
data.head()  
data.Salaries\_new.value\_counts()  
  
################## Dummy Variables ###############  
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
  
# we use ethinc diversity dataset  
df = pd.read\_csv("C:/Data/ethnic diversity.csv")  
  
df.columns # column names  
df.shape # will give u shape of the dataframe  
  
# drop emp\_name column  
df.drop(['Employee\_Name','EmpID','Zip'], axis=1, inplace=True)  
df.dtypes  
  
# Create dummy variables  
df\_new = pd.get\_dummies(df)  
df\_new\_1 = pd.get\_dummies(df, drop\_first = True)  
# we have created dummies for all categorical columns  
  
##### One Hot Encoding works  
df.columns  
df = df[['Salaries', 'age', 'Position', 'State','Sex',  
         'MaritalDesc', 'CitizenDesc', 'EmploymentStatus', 'Department','Race']]  
  
  
from sklearn.preprocessing import OneHotEncoder  
# Creating instance of One Hot Encoder  
enc = OneHotEncoder() # initializing method  
  
enc\_df = pd.DataFrame(enc.fit\_transform(df.iloc[:, 2:]).toarray())  
  
# Label Encoder  
from sklearn.preprocessing import LabelEncoder  
# creating instance of labelencoder  
labelencoder = LabelEncoder()  
  
# Data Split into Input and Output variables  
X = df.iloc[:, 0:9]  
  
y = df['Race']  
y = df.iloc[:, 9:] # Alternative approach  
  
df.columns  
  
X['Sex']= labelencoder.fit\_transform(X['Sex'])  
X['MaritalDesc'] = labelencoder.fit\_transform(X['MaritalDesc'])  
X['CitizenDesc'] = labelencoder.fit\_transform(X['CitizenDesc'])  
  
### label encode y ###  
y = labelencoder.fit\_transform(y)  
y = pd.DataFrame(y)  
  
### we have to convert y to data frame so that we can use concatenate function  
# concatenate X and y  
df\_new = pd.concat([X, y], axis =1)  
  
## rename column name  
df\_new.columns  
df\_new = df\_new.rename(columns={0:'Race'})  
  
import pandas as pd  
import numpy as np  
  
### Standardization  
from sklearn.preprocessing import StandardScaler  
d = pd.read\_csv("C:/Data/mtcars.csv")  
  
a = d.describe()  
# Initialise the Scaler  
scaler = StandardScaler()  
# To scale data  
df = scaler.fit\_transform(d)  
# Convert the array back to a dataframe  
dataset = pd.DataFrame(df)  
res = dataset.describe()  
  
### Normalization  
## load data set  
ethnic = pd.read\_csv("C:/Data/ethnic diversity.csv")  
ethnic.columns  
ethnic.drop(['Employee\_Name', 'EmpID', 'Zip'], axis = 1, inplace = True)  
  
a1 = ethnic.describe()  
  
# get dummies  
ethnic = pd.get\_dummies(ethnic, drop\_first = True)  
  
### Normalization function - Custom Function  
# Range converts to: 0 to 1  
def norm\_func(i):  
    x = (i-i.min())/(i.max()-i.min())  
    return(x)  
  
df\_norm = norm\_func(ethnic)  
b = df\_norm.describe()

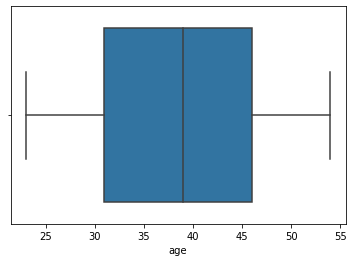
**Outputs:**

****

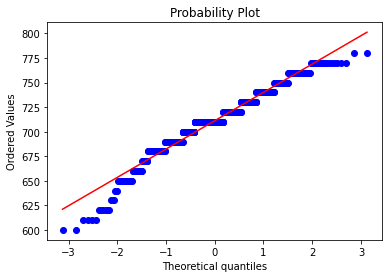
**Box-Plot shows the Outliers in ethnic diversity. Salaries**

****

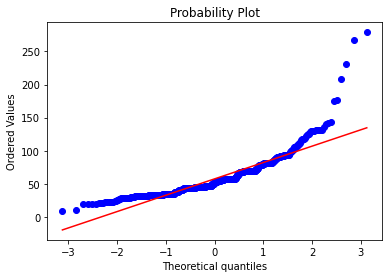
**Box-Plot shows the there is no Outliers in ethnic diversity. Salaries after applying Winsorization**

****

**Box-Plot shows the Outliers in ethnic diversity. Age**

****

**Prob-plot shows the Distribution of education .gmat**

****

**Prob-plot shows the Distribution of education. workex**