DMPM Assignment 3

Name: Saniya S. Inamdar

SRN: 201900913

Roll no. : 17

CODE:

```
library(dplyr)
library(tidyverse)
library(dplyr)
library(ggplot2)
library(Hmisc)
dirtydf <- read.csv("dirty.csv")

head(dirtydf)
summary(dirtydf)
glimpse(dirtydf)

na_count <-sapply(dirtydf, function(y) sum(length(which(is.na(y)))))
na_count <- data.frame(na_count)
na_count


par( mfrow= c(1,3) )
boxplot(dirtydf$Age,main="Age")
boxplot(dirtydf$Weight,main="Wieght")
boxplot(dirtydf$Price,main="Price")

outlier_values <- boxplot.stats(dirtydf$Price)$out  # outlier values.
print(outlier_values)

omitdf<-na.omit(dirtydf)
cat("Percentage of missing values in the na omitted dataset",mean(is.na(omitdf)),"%")

#imputing age with median values and weight w/ mean values
dirtydf$Weight = impute(dirtydf$Weight, fun = mean) # mean imputation
dirtydf$Age = impute(dirtydf$Age, fun = median) # median imputation

sum(is.na(dirtydf$Weight))
sum(is.na(dirtydf$Age))

dirtydf<-dirtydf[!is.na(dirtydf),]
sum(is.na(dirtydf))
```

OUTPUT:

```
> summary(dirtydf)
     Price            Age              KM           FuelType
 Min.   : 4350   Min.   : 1.0   Min.   :     1   Length:1436
 1st Qu.: 8450   1st Qu.:44.0   1st Qu.: 43000   Class :character
 Median : 9900   Median :61.0   Median : 63390   Mode  :character
 Mean   :10731   Mean   :56.1   Mean   : 68533
 3rd Qu.:11950   3rd Qu.:70.0   3rd Qu.: 87021
 Max.   :32500   Max.   :80.0   Max.   :243000
                 NA's   :6
       HP           MetColor        Automatic             CC
 Min.   : 69.0   Min.   :0.0000   Min.   :0.00000   Min.   :1300
 1st Qu.: 90.0   1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:1400
 Median :110.0   Median :1.0000   Median :0.00000   Median :1600
 Mean   :101.5   Mean   :0.6748   Mean   :0.05571   Mean   :1567
 3rd Qu.:110.0   3rd Qu.:1.0000   3rd Qu.:0.00000   3rd Qu.:1600
 Max.   :192.0   Max.   :1.0000   Max.   :1.00000   Max.   :2000
                                                    NA's   :3
     Doors           Weight
 Min.   :2.000   Min.   :1000
 1st Qu.:3.000   1st Qu.:1040
 Median :4.000   Median :1066
 Mean   :4.033   Mean   :1072
 3rd Qu.:5.000   3rd Qu.:1085
 Max.   :5.000   Max.   :1615
                 NA's   :6
```

By observing the summary we can say that there are 3 columns with missing values.

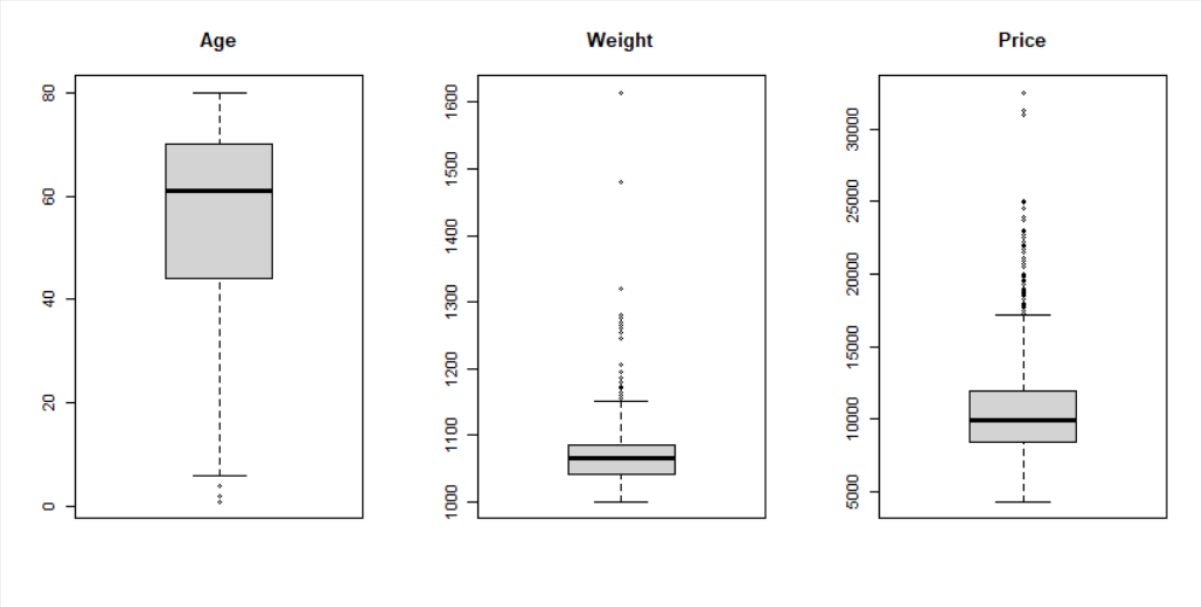
```
> glimpse(dirtydf)
Rows: 1,436
Columns: 10
$ Price     <int> 13500, 13750, 13950, 14950, 13750, 12950, 16900, 18600, 215~
$ Age       <int> 23, 23, NA, 26, 30, 32, 27, 30, 27, NA, 25, 22, 25, 31, 32,~
$ KM        <int> 46986, 72937, 41711, 48000, 38500, 61000, 94612, 75889, 197~
$ FuelType  <chr> "Diesel", "Diesel", "Diesel", "Diesel", " ", "Diesel", "Die~
$ HP        <int> 90, 90, 90, 90, 90, 90, 90, 90, 192, 69, 192, 192, 192, 192~
$ MetColor  <int> 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1,~
$ Automatic <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ CC        <int> 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 1800, 1900,~
$ Doors     <int> 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,~
$ Weight    <int> 1165, 1165, 1165, 1165, 1170, 1170, 1245, 1245, 1185, 1105,~
>
```

Glimpse() gives a peek into the dataset and an overall info of the features, also the dimensions of the dataset.

```
> na_count <-sapply(dirtydf, function(y) sum(length(which(is.na(y)))))
> na_count <- data.frame(na_count)
> na_count
          na_count
Price            0
Age              6
KM               0
FuelType         0
HP               0
MetColor         0
Automatic        0
CC               3
Doors            0
Weight           6
```

This is the function I created for a simple view of the name of the clomns and the count of missing values.

Now let's analyze the Outliers using boxplot():



The points above and below the "whiskers" denote the outlier points

Let's look at the outliers of the "Price" column.

```
> outlier_values <- boxplot.stats(dirtydf$Price)$out  # outlier values.
> print(outlier_values)
  [1] 18600 21500 20950 19950 19600 21500 22500 22000 22750 17950 17495 17950
 [13] 19000 17950 17950 21950 17950 20500 21950 18950 18750 17950 17950 18950
 [25] 22250 18950 19950 18750 18450 18950 17250 17950 17450 17950 21950 22250
 [37] 19950 18900 19950 18750 17450 18990 18500 18500 19450 18800 17450 17950
 [49] 32500 31000 31275 24950 24950 22950 24990 21950 17900 19250 22250 18950
 [61] 19950 18950 21750 17950 18450 23000 19900 23950 19950 18500 18950 20500
 [73] 24500 19450 20950 19950 18450 19500 21750 19500 18900 19750 19750 18950
 [85] 20750 19600 19500 17650 19950 19950 20950 20500 17795 18245 23750 19500
 [97] 18950 21950 19950 18950 19950 21950 22500 18500 18700 21125 21500 17795
[109] 18245 18950
```

These outliers can be removed or treated.

Now let's take care of the NULL values of the dataset;

1. One way to do this is by dropping all the missing values in the dataset:

```
> omitdf<-na.omit(dirtydf)
> cat("Percentage of missing values in the na omitted dataset",mean(is.na(omitd
f)),"%")
Percentage of missing values in the na omitted dataset 0 %
>
```

2. Another way is to impute the missing values
   We have 3 columns with missing values: Age(6), Weight(6), CC(3)
   Since, Age is an integer, it's quite logical to impute the NULL values of this column by its median.
   And Weight can have a floating-point wo we can impute this column's missing values by its mean.
   Since the number of missing values in CC column are very less, i.e. 3, so it won't much affect the dataset, o I am going to drop those missing values.

```
> #imputing age with median values and weight w/ mean values
> dirtydf$Weight = impute(dirtydf$Weight, fun = mean) # mean imputation
> dirtydf$Age = impute(dirtydf$Age, fun = median) # median imputation
> sum(is.na(dirtydf$Weight))
[1] 0
> sum(is.na(dirtydf$Age))
[1] 0
> dirtydf=na.omit(dirtydf$CC)
> sum(is.na(dirtydf))
[1] 0
```

The dirty dataset is cleaned!!!

END OF PART 1

For the 2nd part I have cleaned a "credit default" dataset using Python and it's libraries.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from pandas_profiling import ProfileReport
```

In [2]:
```
df = pd.read_csv('dataset/train.csv')
# shuffle the DataFrame rows
df = df.sample(frac = 1)
df
```

Out[2]:

	customer_id	name	age	gender	owns_car	owns_house	no_of_children	net_yearly_income	no_of_days_employed	occupa
19189	CST_134259	Alison	32	F	N	N	0.0	126169.76	117.0	
39015	CST_131984	Rod	38	F	N	Y	1.0	218984.38	365250.0	
23613	CST_144573	Angela	51	M	N	Y	2.0	220004.97	749.0	
15843	CST_142563	Ross	26	M	Y	N	1.0	194414.03	1949.0	
31789	CST_147145	Reynolds	54	M	N	Y	0.0	210161.00	1043.0	
...	...	...	...	...	...	...	...	...	...	
26564	CST_106369	Terhi Kinnunen	29	F	N	N	0.0	104092.91	3489.0	
4063	CST_165818	Sheppard	35	M	N	Y	0.0	114944.02	881.0	Se
7132	CST_132286	Deepa	35	F	N	Y	0.0	138636.96	365247.0	
24235	CST_108722	Jonathan Leff	28	M	Y	Y	0.0	238468.93	218.0	
30686	CST_134710	Silvia	45	F	N	Y	0.0	202168.00	1283.0	

45528 rows × 19 columns

In [3]:
```
np.sum(df.isnull().any(axis=1))
```

Out[3]: 2019

In [4]:
```
df.isnull().any()
```

Out[4]:
```
customer_id                False
name                       False
age                        False
gender                     False
owns_car                    True
owns_house                 False
no_of_children              True
net_yearly_income          False
no_of_days_employed         True
occupation_type            False
total_family_members        True
migrant_worker              True
yearly_debt_payments        True
credit_limit               False
credit_limit_used(%)       False
credit_score                True
prev_defaults              False
default_in_last_6months    False
credit_card_default        False
dtype: bool
```

In [5]:
```
report = ProfileReport(df)
report.to_notebook_iframe()
```

Overview

Dataset statistics

Number of variables	20
Number of observations	45528
Missing cells	2057
Missing cells (%)	0.2%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	6.9 MiB
Average record size in memory	160.0 B

Variable types

Numeric	10
Categorical	8
Boolean	2

Alerts

`customer_id` has a high cardinality: 45528 distinct values	**High cardinality**
`name` has a high cardinality: 4010 distinct values	**High cardinality**
`no_of_children` is highly correlated with `total_family_members`	**High correlation**
`net_yearly_income` is highly correlated with `credit_limit`	**High correlation**
`total_family_members` is highly correlated with `no_of_children`	**High correlation**

`credit_limit` is highly correlated with `net_yearly_income`	**High correlation**
`prev_defaults` is highly correlated with `default_in_last_6months` and 1 other fields (default_in_last_6months, credit_card_default)	**High correlation**

In [6]:
```
df.drop(['name'],axis=1,inplace=True)
```

In [7]:
```
df
```

Out[7]:

	customer_id	age	gender	owns_car	owns_house	no_of_children	net_yearly_income	no_of_days_employed	occupation_type
19189	CST_134259	32	F	N	N	0.0	126169.76	117.0	Unknown
39015	CST_131984	38	F	N	Y	1.0	218984.38	365250.0	Unknown
23613	CST_144573	51	M	N	Y	2.0	220004.97	749.0	Laborers
15843	CST_142563	26	M	Y	N	1.0	194414.03	1949.0	Drivers
31789	CST_147145	54	M	N	Y	0.0	210161.00	1043.0	Drivers
...	...	...	...	...	...	...	...	...	...
26564	CST_106369	29	F	N	N	0.0	104092.91	3489.0	Unknown
4063	CST_165818	35	M	N	Y	0.0	114944.02	881.0	Security staff
7132	CST_132286	35	F	N	Y	0.0	138636.96	365247.0	Unknown
24235	CST_108722	28	M	Y	Y	0.0	238468.93	218.0	Unknown
30686	CST_134710	45	F	N	Y	0.0	202168.00	1283.0	Core staff

45528 rows × 18 columns

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
customer_id                  0
age                          0
gender                       0
owns_car                   547
owns_house                   0
no_of_children             774
net_yearly_income            0
no_of_days_employed        463
occupation_type              0
total_family_members        83
migrant_worker              87
yearly_debt_payments        95
credit_limit                 0
credit_limit_used(%)         0
credit_score                 8
prev_defaults                0
default_in_last_6months      0
credit_card_default          0
dtype: int64
```

In [9]:
```
df.skew().sort_values(ascending=False)
```

Out[9]:
```
net_yearly_income          203.683504
credit_limit               200.387167
prev_defaults                4.681004
default_in_last_6months      4.103720
credit_card_default          3.066570
no_of_children               1.827606
yearly_debt_payments         1.721201
migrant_worker               1.673767
no_of_days_employed          1.667675
total_family_members         0.924824
age                          0.003975
credit_limit_used(%)        -0.127449
credit_score                -0.302517
dtype: float64
```

In [10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 45528 entries, 19189 to 30686
Data columns (total 18 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   customer_id              45528 non-null  object
 1   age                      45528 non-null  int64
 2   gender                   45528 non-null  object
 3   owns_car                 44981 non-null  object
 4   owns_house               45528 non-null  object
 5   no_of_children           44754 non-null  float64
 6   net_yearly_income        45528 non-null  float64
 7   no_of_days_employed      45065 non-null  float64
 8   occupation_type          45528 non-null  object
 9   total_family_members     45445 non-null  float64
 10  migrant_worker           45441 non-null  float64
 11  yearly_debt_payments     45433 non-null  float64
 12  credit_limit             45528 non-null  float64
 13  credit_limit_used(%)     45528 non-null  int64
 14  credit_score             45520 non-null  float64
 15  prev_defaults            45528 non-null  int64
 16  default_in_last_6months  45528 non-null  int64
 17  credit_card_default      45528 non-null  int64
dtypes: float64(8), int64(5), object(5)
memory usage: 6.6+ MB
```

In [11]:
```
#net_yearly_income & credit_limit
df['net_yearly_income'] = np.log(df['net_yearly_income'] )
df['credit_limit'] = np.log(df['credit_limit'])
```

In [12]:
```
#ordinal encoding of various columns
df['gender'].replace('M', 0, inplace=True)
df['gender'].replace('F', 1, inplace=True)
df['gender'].replace('XNA', 1, inplace=True)
df['owns_car'].replace('N', 0, inplace=True)
df['owns_car'].replace('Y', 1, inplace=True)
df['owns_house'].replace('N', 0, inplace=True)
df['owns_house'].replace('Y', 1, inplace=True)
```

In [13]:
```
from sklearn import preprocessing

label_encoder = preprocessing.LabelEncoder()

df['occupation_type']= label_encoder.fit_transform(df['occupation_type'])
df.head()
```

Out[13]:

	customer_id	age	gender	owns_car	owns_house	no_of_children	net_yearly_income	no_of_days_employed	occupation_type
19189	CST_134259	32	1	0.0	0	0.0	11.745384	117.0	17
39015	CST_131984	38	1	0.0	1	1.0	12.296756	365250.0	17
23613	CST_144573	51	0	0.0	1	2.0	12.301405	749.0	8
15843	CST_142563	26	0	1.0	0	1.0	12.177745	1949.0	4
31789	CST_147145	54	0	0.0	1	0.0	12.255629	1043.0	4

In [14]:
```
corr = df.corr()
```

In [15]:
```
import plotly.express as px
fig = px.imshow(corr)
fig.show()
```

In [ ]: