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

df=pd.read_excel("/content/drive/MyDrive/datasets/Employee-turnover
(1) (2).xlsx")
df.head()
```

```
   Employee Count  Employee ID              Department
Job Role  \
0             1.0          1.0                   Sales        Sales
Executive
1             1.0          2.0  Research & Development     Research
Scientist
2             1.0          4.0  Research & Development  Laboratory
Technician
3             1.0          5.0  Research & Development     Research
Scientist
4             1.0          7.0  Research & Development  Laboratory
Technician

   Gender   Age Marital Status      Education Education Field  \
0  Female  41.0         Single        College   Life Sciences
1    Male  49.0        Married  Below College   Life Sciences
2    Male  37.0         Single        College           Other
3  Female  33.0        Married         Master   Life Sciences
4    Male  27.0        Married  Below College         Medical

     Business Travel  ... Total Working Years Years At Company  \
0      Travel_Rarely  ...                 8.0              6.0
1  Travel_Frequently  ...                10.0             10.0
2      Travel_Rarely  ...                 7.0              0.0
3  Travel_Frequently  ...                 8.0              8.0
4      Travel_Rarely  ...                 6.0              2.0

   Years In Current Role Years Since Last Promotion  Years With Curr
Manager  \
0                    4.0                         0.0
5.0
1                    7.0                         1.0
7.0
2                    0.0                         0.0
0.0
3                    7.0                         3.0
0.0
4                    2.0                         2.0
2.0
```

```
   Environment Satisfaction  Training Times Last Year Work Life
Balance  \
0                     Medium                         NaN
Bad
1                       High                         NaN
Better
2                        Low                         NaN
Bad
3                  Very High                         NaN
Better
4                        Low                         NaN
Better

   Relationship Satisfaction  Attrition (Yes/No)
0                        Low                 Yes
1                  Very High                  No
2                     Medium                 Yes
3                       High                  No
4                  Very High                  No

[5 rows x 29 columns]
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 29 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   Employee Count               1470 non-null   float64
 1   Employee ID                  1470 non-null   float64
 2   Department                   1470 non-null   object
 3   Job Role                     1470 non-null   object
 4   Gender                       1470 non-null   object
 5   Age                          1470 non-null   float64
 6   Marital Status               1470 non-null   object
 7   Education                    1470 non-null   object
 8   Education Field              1470 non-null   object
 9   Business Travel              1470 non-null   object
 10  Distance From Home (kms)     1439 non-null   object
 11  Job Involvement              1470 non-null   object
 12  Job Level                    1464 non-null   float64
 13  Job Satisfaction             1458 non-null   object
 14  Monthly Income (USD)         1470 non-null   float64
 15  Salary Hike (%)              1470 non-null   float64
 16  Stock Option Level           1470 non-null   float64
 17  Over Time                    1470 non-null   object
 18  No. of Companies Worked      1470 non-null   float64
 19  Total Working Years          1451 non-null   float64
 20  Years At Company             1470 non-null   float64
```

```
 21  Years In Current Role       1470 non-null   float64
 22  Years Since Last Promotion  1470 non-null   float64
 23  Years With Curr Manager     1470 non-null   float64
 24  Environment Satisfaction    1470 non-null   object
 25  Training Times Last Year    664 non-null    float64
 26  Work Life Balance           1470 non-null   object
 27  Relationship Satisfaction   1470 non-null   object
 28  Attrition (Yes/No)          1470 non-null   object
dtypes: float64(14), object(15)
memory usage: 333.2+ KB
```

```
df.isnull().sum()/len(df)
```

```
Employee Count                 0.000000
Employee ID                    0.000000
Department                     0.000000
Job Role                       0.000000
Gender                         0.000000
Age                            0.000000
Marital Status                 0.000000
Education                      0.000000
Education Field                0.000000
Business Travel                0.000000
Distance From Home (kms)       0.021088
Job Involvement                0.000000
Job Level                      0.004082
Job Satisfaction               0.008163
Monthly Income (USD)           0.000000
Salary Hike (%)                0.000000
Stock Option Level             0.000000
Over Time                      0.000000
No. of Companies Worked        0.000000
Total Working Years            0.012925
Years At Company               0.000000
Years In Current Role          0.000000
Years Since Last Promotion     0.000000
Years With Curr Manager        0.000000
Environment Satisfaction       0.000000
Training Times Last Year       0.548299
Work Life Balance              0.000000
Relationship Satisfaction      0.000000
Attrition (Yes/No)             0.000000
dtype: float64
```

```
df.drop(['Training Times Last Year'],axis=1,inplace=True)#Since the
ratio > 0.5
```

```
df[['Distance From Home (kms)','Job Level','Job Satisfaction','Total
Working Years']].describe(include='all')
```

```
        Distance From Home (kms)    Job Level Job Satisfaction  \
count                     1439.0  1464.000000              1458
unique                      30.0          NaN                 4
top                          2.0          NaN         Very High
freq                       208.0          NaN               455
mean                         NaN     2.066940               NaN
std                          NaN     1.107805               NaN
min                          NaN     1.000000               NaN
25%                          NaN     1.000000               NaN
50%                          NaN     2.000000               NaN
75%                          NaN     3.000000               NaN
max                          NaN     5.000000               NaN

        Total Working Years
count           1451.000000
unique                  NaN
top                     NaN
freq                    NaN
mean              11.316334
std                7.786009
min                0.000000
25%                6.000000
50%               10.000000
75%               15.000000
max               40.000000
```

```
df.fillna({'Job Level':df['Job Level'].median()},inplace=True)
df.fillna({'Total Working Years':df['Total Working
Years'].median()},inplace=True)

df["Distance From Home (kms)"]=df['Distance From Home
(kms)'].astype(str).astype(float)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call
last)
<ipython-input-56-2e9726f8e1e9> in <module>
----> 1 df["Distance From Home (kms)"]=df['Distance From Home
(kms)'].astype(str).astype(float)

/usr/local/lib/python3.8/dist-packages/pandas/core/generic.py in
astype(self, dtype, copy, errors)
   5813         else:
   5814             # else, only a single dtype is given
-> 5815             new_data = self._mgr.astype(dtype=dtype,
copy=copy, errors=errors)
   5816             return
self._constructor(new_data).__finalize__(self, method="astype")
   5817
```

```
/usr/local/lib/python3.8/dist-packages/pandas/core/internals/managers.
py in astype(self, dtype, copy, errors)
    416
    417     def astype(self: T, dtype, copy: bool = False, errors: str
= "raise") -> T:
--> 418         return self.apply("astype", dtype=dtype, copy=copy,
errors=errors)
    419
    420     def convert(

/usr/local/lib/python3.8/dist-packages/pandas/core/internals/managers.
py in apply(self, f, align_keys, ignore_failures, **kwargs)
    325                     applied = b.apply(f, **kwargs)
    326                 else:
--> 327                     applied = getattr(b, f)(**kwargs)
    328             except (TypeError, NotImplementedError):
    329                 if not ignore_failures:

/usr/local/lib/python3.8/dist-packages/pandas/core/internals/blocks.py
in astype(self, dtype, copy, errors)
    589         values = self.values
    590
--> 591         new_values = astype_array_safe(values, dtype,
copy=copy, errors=errors)
    592
    593         new_values = maybe_coerce_values(new_values)

/usr/local/lib/python3.8/dist-packages/pandas/core/dtypes/cast.py in
astype_array_safe(values, dtype, copy, errors)
   1307
   1308     try:
-> 1309         new_values = astype_array(values, dtype, copy=copy)
   1310     except (ValueError, TypeError):
   1311         # e.g. astype_nansafe can fail on object-dtype of
strings

/usr/local/lib/python3.8/dist-packages/pandas/core/dtypes/cast.py in
astype_array(values, dtype, copy)
   1255
   1256     else:
-> 1257         values = astype_nansafe(values, dtype, copy=copy)
   1258
   1259     # in pandas we don't store numpy str dtypes, so convert to
object

/usr/local/lib/python3.8/dist-packages/pandas/core/dtypes/cast.py in
astype_nansafe(arr, dtype, copy, skipna)
   1199     if copy or is_object_dtype(arr.dtype) or
is_object_dtype(dtype):
```

```
   1200         # Explicit copy, or required since NumPy can't view
from / to object.
-> 1201         return arr.astype(dtype, copy=True)
   1202
   1203     return arr.astype(dtype, copy=copy)

ValueError: could not convert string to float: '?'
```

```
df[df['Distance From Home (kms)'].str.match('\?')==True]['Distance
From Home (kms)']
```

```
17    ?
18    ?
19    ?
Name: Distance From Home (kms), dtype: object
```

```
df['Distance From Home (kms)'].replace('?',np.NaN,inplace=True)
```

```
df["Distance From Home (kms)"]=df['Distance From Home
(kms)'].astype(str).astype(float)
df.fillna({'Distance From Home (kms)':df['Distance From Home
(kms)'].median()},inplace=True)
```

```
df['Job Satisfaction'].value_counts()
```

```
Very High    455
High         437
Low          288
Medium       278
Name: Job Satisfaction, dtype: int64
```

```
#filling in the most probable value
df.fillna({'Job Satisfaction':'Very High'},inplace=True)
```

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

```
Employee Count                0
Employee ID                   0
Department                    0
Job Role                      0
Gender                        0
Age                           0
Marital Status                0
Education                     0
Education Field               0
Business Travel               0
Distance From Home (kms)      0
Job Involvement               0
Job Level                     0
Job Satisfaction              0
Monthly Income (USD)          0
```

```
Salary Hike (%)                0
Stock Option Level             0
Over Time                      0
No. of Companies Worked        0
Total Working Years            0
Years At Company               0
Years In Current Role          0
Years Since Last Promotion     0
Years With Curr Manager        0
Environment Satisfaction       0
Work Life Balance              0
Relationship Satisfaction      0
Attrition (Yes/No)             0
dtype: int64
```

```
df[df.duplicated()]
```

```
Empty DataFrame
Columns: [Employee Count, Employee ID, Department, Job Role, Gender,
Age, Marital Status, Education, Education Field, Business Travel,
Distance From Home (kms), Job Involvement, Job Level, Job
Satisfaction, Monthly Income (USD), Salary Hike (%), Stock Option
Level, Over Time, No. of Companies Worked, Total Working Years, Years
At Company, Years In Current Role, Years Since Last Promotion, Years
With Curr Manager, Environment Satisfaction, Work Life Balance,
Relationship Satisfaction, Attrition (Yes/No)]
Index: []

[0 rows x 28 columns]
```

```
df['Employee Count'].value_counts()
```

```
1.0    1470
Name: Employee Count, dtype: int64
```

```
#Employee count does not add any value to the data hence it can
dropped.
df.drop(['Employee Count'],axis=1,inplace=True)

output = []
for col in df.columns:
    unique = df[col].nunique()
    colType = str(df[col].dtype)
    categories=df[col].unique()

    output.append([col, unique, colType,categories])

output = pd.DataFrame(output)
output.columns = ['colName','unique','dtype','categories']
output
```

	colName	unique	dtype	categories
0	Employee ID	1470	float64	[1.0, 2.0, 4.0, 5.0, 7.0, 8.0, 10.0, 11.0, 12....
1	Department	3	object	[Sales, Research & Development, Human Resources]
2	Job Role	9	object	[Sales Executive, Research Scientist, Laborato...
3	Gender	2	object	[Female, Male]
4	Age	43	float64	[41.0, 49.0, 37.0, 33.0, 27.0, 32.0, 59.0, 30....
5	Marital Status	3	object	[Single, Married, Divorced]
6	Education	5	object	[College, Below College, Master, Bachelor, Doc...
7	Education Field	6	object	[Life Sciences, Other, Medical, Marketing, Tec...
8	Business Travel	3	object	[Travel_Rarely, Travel_Frequently, Non-Travel]
9	Distance From Home (kms)	29	float64	[1.0, 8.0, 2.0, 3.0, 7.0, 23.0, 27.0, 16.0, 15...
10	Job Involvement	4	object	[High, Medium, Very High, Low]
11	Job Level	5	float64	[2.0, 1.0, 3.0, 4.0, 5.0]
12	Job Satisfaction	4	object	[Very High, Medium, High, Low]
13	Monthly Income (USD)	1349	float64	[5993.0, 5130.0, 2090.0, 2909.0, 3468.0, 3068....
14	Salary Hike (%)	16	float64	[11.0, 23.0, 15.0, 12.0, 13.0, 20.0, 22.0, 21....
15	Stock Option Level	4	float64	[0.0, 1.0, 3.0, 2.0]
16	Over Time	2	object	[Yes, No]
17	No. of Companies Worked	10	float64	[2.0, 1.0, 6.0, 9.0, 0.0, 4.0, 5.0, 7.0, 3.0, ...
18	Total Working Years	40	float64	[8.0, 10.0, 7.0, 6.0, 26.0, 24.0, 22.0, 9.0, 1...
19	Years At Company	37	float64	[6.0, 10.0, 0.0, 8.0, 2.0, 7.0, 1.0, 9.0, 5.0,...
20	Years In Current Role	19	float64	
21	Years Since Last Promotion	16	float64	
22	Years With Curr Manager	18	float64	
23	Environment Satisfaction	4	object	
24	Work Life Balance	4	object	
25	Relationship Satisfaction	4	object	
26	Attrition (Yes/No)	2	object	

```
20  [4.0, 7.0, 0.0, 2.0, 5.0, 9.0, 8.0, 3.0, 6.0, ...
21  [0.0, 1.0, 3.0, 2.0, 7.0, 4.0, 8.0, 6.0, 5.0, ...
22  [5.0, 7.0, 0.0, 2.0, 6.0, 8.0, 3.0, 11.0, 17.0...
23                      [Medium, High, Low, Very High]
24                           [Bad, Better, Good, Best]
25                      [Low, Very High, Medium, High]
26                                          [Yes, No]
```

```
ranks = {"Very High":4, "High":3, "Medium":2, "Low":1}
df.replace(ranks,inplace=True)
```

```
df.rename(columns = {'Attrition (Yes/No)':'Attrition'}, inplace =
True)
```

```
print(np.max(df['Age']))
print(np.min(df['Age']))
```

```
60.0
18.0
```

```
df['Age bins'] = pd.cut(x=df['Age'], bins=[10,20,30,40,50,60])
```

```
df['Age bins'].value_counts()
```

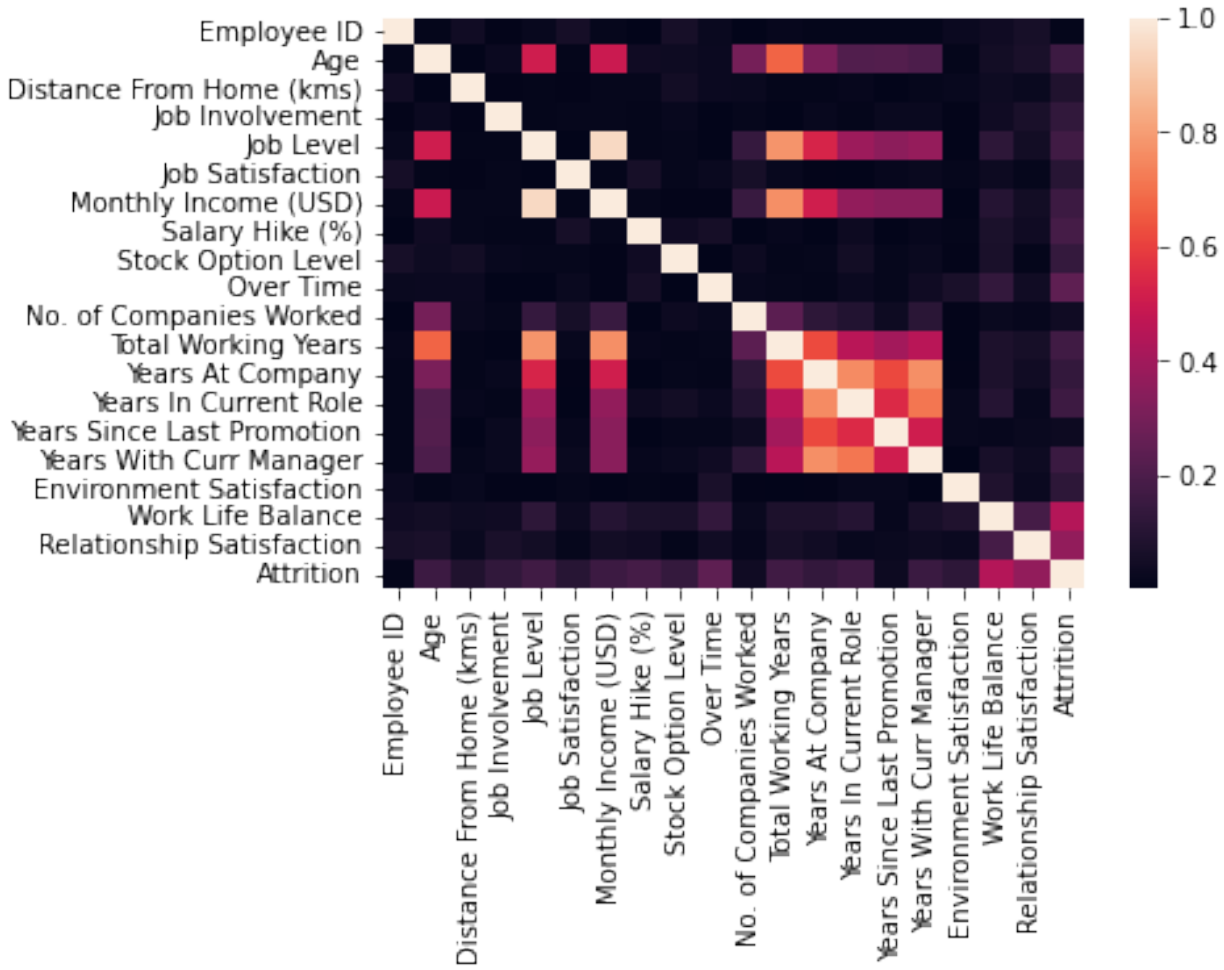
```
(30, 40]    619
(20, 30]    358
(40, 50]    322
(50, 60]    143
(10, 20]     28
Name: Age bins, dtype: int64
```

```
labels = {"Yes":0, "No":1}
df.replace(labels,inplace=True)
```

```
ranks = {"Bad":3, "Good":2, "Better":1, "Best":0}
df.replace(ranks,inplace=True)
```

```
sns.heatmap(df.corr().abs())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdf7568bdf0>
```



Following shows the relation between attrition and it's highly correlated columns

```
pd.crosstab(df['Attrition'], df['Work Life Balance'])
```

Work Life Balance	0	1	2	3
Attrition				
0	6	58	55	118
1	126	762	286	59

```
pd.crosstab(df['Attrition'], df['Relationship Satisfaction'])
```

Relationship Satisfaction	1	2	3	4
Attrition				
0	127	70	38	2
1	219	258	388	368

```
pd.crosstab(df['Attrition'], df['Over Time'])
```

Over Time	0	1
Attrition		

```
0          127  110
1          289  944
```

In [ ]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [ ]:
```
#display the top 5 observations of the dataset
```

In [ ]:
```
df = pd.read_excel("/content/DLA_trips_dataset (1).xlsx")
```

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

Out[ ]:

	booking id	booking_date_time	gender	month	day_of_week	time_of_day	distance_travelled	time_taken	reason	toll	category	commission_base_cost	driver_base_cost	total_tax	total_trip_cost
0	1890001540	43249.919444	Male	May	Tue	0.919444	17	58.0	Office to/from Home	0	Mini	57.73	230.91	21.94	311.00
1	1542149932	43153.925000	Female	February	Thu	0.925000	18	43.0	Late Night Drop	0	Mini	52.04	208.16	19.76	279.96
2	1672692603	43194.882639	Female	April	Wed	0.882639	2	5.0	Office to/from Home	0	Prime	19.70	78.81	7.49	106.00
3	1925600201	43150.932639	Female	March	Thu	0.932639	15	49.0	Office to/from Home	35	Micro	51.24	239.96	21.22	312.00
4	1538065664	43150.479861	Male	February	Mon	0.479861	46	0.0	Office Event	0	Prime Rentals	195.92	783.68	74.45	1054.05

In [ ]:
```
#several unique values in each column
df.nunique()
```

In [ ]:
```
df.tail()
```

Out[ ]:

	booking id	booking_date_time	gender	month	day_of_week	time_of_day	distance_travelled	time_taken	reason	toll	category	commission_base_cost	driver_base_cost	total_tax	total_tri
4945	1901877370	43252.909722	Female	June	Fri	0.909722	29	90.00	Office to/from Home	35	Micro	113.75	490.02	44.97	
4946	1867091987	43243.933333	Female	May	Wed	0.933333	1	2.00	Office to/from Home	0	Mini	15.84	63.36	6.02	
4947	1747312670	43213.971528	Male	April	Tue	0.971528	8	34.00	Office to/from Home	0	Mini	33.78	135.13	12.84	
4948	1635330690	43163.000333	Male	March	Sat	0.000333	10	30.00	Office to/from Home	0	Prime	43.29	173.17	16.45	
4949	OSN_1309565727	43270.966806	Female	June	Tue	0.966806	6	17.00	Office to/from Home	0	Mini	28.99	115.97	11.02	

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

Out[ ]:

	booking_date_time	time_of_day	distance_travelled	time_taken	toll	commission_base_cost	driver_base_cost	total_tax	total_trip_cost	ratings
count	4950.000000	4950.000000	4950.000000	4950.000000	4950.000000	4950.000000	4950.000000	4950.000000	4950.000000	4950.000000
mean	43195.816098	0.686199	11.713333	35.126137	5.408081	47.596505	196.162053	18.381604	262.391305	3.734949
std	53.621694	0.372318	10.336660	25.592956	15.910681	37.159192	155.306959	14.404632	206.539961	1.172378
min	43101.041956	0.000000	0.000000	0.000000	0.000000	0.000000	26.020000	2.440000	34.460000	1.000000
25%	43147.822222	0.273090	4.000000	7.000000	0.000000	22.625000	90.865000	8.620000	122.305000	1.000000
50%	43195.875000	0.900000	8.500000	35.000000	0.000000	37.615000	151.145000	14.350000	203.360000	4.000000
75%	43245.182656	0.930556	16.000000	50.000000	0.000000	58.517500	242.685000	23.067500	325.000000	5.000000
max	43291.199611	0.999306	66.000000	182.000000	140.000000	359.240000	1369.600000	129.120000	1828.120000	5.000000

In [ ]:
```
#identify null values in the data
df.isnull()
```

Out[ ]:

	booking id	booking_date_time	gender	month	day_of_week	time_of_day	distance_travelled	time_taken	reason	toll	category	commission_base_cost	driver_base_cost	total_tax	total_trip_cost	ratin
0	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4945	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	
4946	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	
4947	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	
4948	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	
4949	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	

4950 rows × 16 columns

In [ ]:
```
#used to get the number of missing records in each column
df.isnull().sum()
```

Out[ ]:
```
booking id            0
booking_date_time     0
gender               18
month                 0
day_of_week           0
time_of_day           0
distance_travelled    0
time_taken            0
reason               93
toll                  0
category              0
commission_base_cost  0
driver_base_cost      0
total_tax             0
total_trip_cost      15
ratings               0
dtype: int64
```

In [ ]:
```
#calculate the percentage of missing values in each column
(df.isnull().sum()/(len(df)))*100
```

Out[ ]:
```
booking id            0.000000
booking_date_time     0.000000
gender                0.363636
month                 0.000000
day_of_week           0.000000
time_of_day           0.000000
distance_travelled    0.000000
time_taken            0.000000
reason                1.878788
toll                  0.000000
category              0.000000
commission_base_cost  0.000000
driver_base_cost      0.000000
total_tax             0.000000
total_trip_cost       0.303030
ratings               0.000000
dtype: float64
```

In [ ]:
```
df[df.duplicated()]
```

Out[ ]:

	booking id	booking_date_time	gender	month	day_of_week	time_of_day	distance_travelled	time_taken	reason	toll	category	commission_base_cost	driver_base_cost	total_tax	total_trip_cost	ratin

In [ ]:

In [ ]:

In [ ]:
```
#used to get the number of missing records in each column
df.isnull().sum()
```

Out[ ]:
```
booking id            0
booking_date_time     0
gender               18
month                 0
day_of_week           0
time_of_day           0
distance_travelled    0
time_taken            0
reason               93
toll                  0
category              0
commission_base_cost  0
driver_base_cost      0
total_tax             0
total_trip_cost      15
ratings               0
dtype: int64
```

In [ ]:
```
import numpy as np

# Assuming df is your pandas DataFrame
mean = np.mean(df["total_trip_cost"])

df["total_trip_cost"] = df["total_trip_cost"].fillna(mean)
```

In [ ]:
```
#used to get the number of missing records in each column
df.isnull().sum()
```

Out[ ]:
```
booking id            0
booking_date_time     0
gender               18
month                 0
day_of_week           0
time_of_day           0
distance_travelled    0
time_taken            0
reason               93
toll                  0
category              0
commission_base_cost  0
driver_base_cost      0
total_tax             0
total_trip_cost       0
ratings               0
dtype: int64
```

In [ ]:
```
df = df.dropna(subset=["reason"])
```

In [ ]:
```
#used to get the number of missing records in each column
df.isnull().sum()
```

Out[ ]:
```
booking id            0
booking_date_time     0
gender               17
month                 0
day_of_week           0
time_of_day           0
distance_travelled    0
time_taken            0
reason                0
toll                  0
category              0
commission_base_cost  0
driver_base_cost      0
total_tax             0
total_trip_cost       0
ratings               0
dtype: int64
```

In [ ]:
```
df = df.dropna(subset=["time_of_day"])
```

In [ ]:
```
#used to get the number of missing records in each column
df.isnull().sum()
```

Out[ ]:
```
booking id            0
booking_date_time     0
gender               17
month                 0
day_of_week           0
time_of_day           0
distance_travelled    0
time_taken            0
reason                0
toll                  0
category              0
commission_base_cost  0
driver_base_cost      0
total_tax             0
total_trip_cost       0
ratings               0
dtype: int64
```

In [ ]:
```
df['gender'].value_counts()
```

Out[ ]:
```
Female    2599
Male      2268
Name: gender, dtype: int64
```

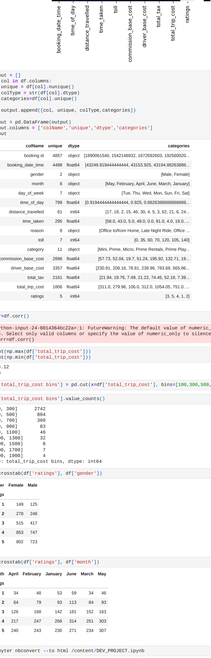
In [ ]:
```
#Filling in the most probable value
df.fillna({'gender':'Female'},inplace=True)
```

In [ ]:
```
#used to get the number of missing records in each column
df.isnull().sum()
```

Out[ ]:
```
booking id            0
booking_date_time     0
gender                0
month                 0
day_of_week           0
time_of_day           0
distance_travelled    0
time_taken            0
reason                0
toll                  0
category              0
commission_base_cost  0
driver_base_cost      0
total_tax             0
total_trip_cost       0
ratings               0
dtype: int64
```

In [ ]:
```
sns.heatmap(df.corr(),annot=True)
```

Out[ ]:
```
<ipython-input-22-8df7bcac526d>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  sns.heatmap(df.corr(),annot=True)
<Axes: >
```



In [ ]:
```
output = []
for col in df.columns:
    unique = df[col].nunique()
    coltype = str(df[col].dtype)
    categories=df[col].unique()

    output.append([col, unique, coltype,categories])

output = pd.DataFrame(output)
output.columns = ['colName', 'unique', 'dtype', 'categories']
output
```

Out[ ]:

	colName	unique	dtype	categories
0	booking id	4857	object	[1890001540, 1542149932, 1672692603, 1925600201,...
1	booking_date_time	4488	float64	[43249.91944444444, 43153.925, 43194.88263888...
2	gender	2	object	[Male, Female]
3	month	6	object	[May, February, April, June, March, January]
4	day_of_week	7	object	[Tue, Thu, Wed, Mon, Sun, Fri, Sat]
5	time_of_day	799	float64	[0.9194444444444444, 0.925, 0.8826388888888889,...
6	distance_travelled	61	int64	[17, 18, 2, 15, 46, 30, 4, 5, 3, 62, 21, 6, 24,...
7	time_taken	295	float64	[58.0, 43.0, 5.0, 49.0, 0.0, 91.0, 4.0, 19.0,...
8	reason	8	object	[Office to/from Home, Late Night Ride, Office ...
9	toll	7	int64	[0, 35, 60, 70, 120, 105, 140]
10	category	11	object	[Mini, Prime, Micro, Prime Rentals, Prime Play,...
11	commission_base_cost	2896	float64	[57.73, 52.04, 19.7, 51.24, 195.92, 132.71, 19....
12	driver_base_cost	3357	float64	[230.91, 208.16, 78.81, 239.96, 783.68, 565.96,...
13	total_tax	2161	float64	[21.94, 19.76, 7.49, 21.22, 74.45, 52.18, 7.39,...
14	total_trip_cost	1806	float64	[311.0, 279.96, 106.0, 312.0, 1054.05, 751.0,...
15	ratings	5	int64	[1, 5, 4, 1, 2]

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

In [ ]:
```
<ipython-input-24-80143640c22a>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  corr=df.corr()
```

In [ ]:
```
print(np.max(df['total_trip_cost']))
print(np.min(df['total_trip_cost']))
```

Out[ ]:
```
1828.12
34.0
```

In [ ]:
```
df['total_trip_cost bins'] = pd.cut(x=df['total_trip_cost'], bins=[100,300,500,700,900,1100,1300,1500,1700,1900])
```

In [ ]:
```
df['total_trip_cost bins'].value_counts()
```

Out[ ]:
```
(100, 300]      2742
(300, 500]       894
(500, 700]       360
(700, 900]        83
(900, 1100]       46
(1100, 1300]      14
(1300, 1500]       8
(1500, 1700]       7
(1700, 1900]       4
Name: total_trip_cost bins, dtype: int64
```

In [ ]:
```
pd.crosstab(df['ratings'], df['gender'])
```

Out[ ]:

gender	Female	Male
ratings		
1	140	125
2	279	248
3	515	417
4	853	747
5	802	723

In [ ]:
```
pd.crosstab(df['ratings'], df['month'])
```

Out[ ]:

month	April	February	January	June	March	May
ratings						
1	34	48	53	59	34	46
2	64	79	93	113	84	93
3	126	168	142	181	152	163
4	217	247	268	314	251	303
5	240	243	230	271	234	307

In [ ]:
```
!jupyter nbconvert --to html /content/DEV_PROJECT.ipynb
```

Out[ ]:
```
[NbConvertApp] Converting notebook /content/DEV_PROJECT.ipynb to html
[NbConvertApp] Writing 803849 bytes to /content/DEV_PROJECT.html
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [171…
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
df=pd.read_csv('/content/UberDatasett.csv')
```

In [172…
```
df
```

Out[172]:

	START_DATE	END_DATE	CATEGORY	START	STOP	MILES	PURPOSE
0	01-01-2016 21:11	01-01-2016 21:17	Business	Fort Pierce	Fort Pierce	5.1	Meal/Entertain
1	01-02-2016 01:25	01-02-2016 01:37	Business	Fort Pierce	Fort Pierce	5.0	NaN
2	01-02-2016 20:25	01-02-2016 20:38	Business	Fort Pierce	Fort Pierce	4.8	Errand/Supplies
3	01-05-2016 17:31	01-05-2016 17:45	Business	Fort Pierce	Fort Pierce	4.7	Meeting
4	01-06-2016 14:42	01-06-2016 15:49	Business	Fort Pierce	West Palm Beach	63.7	Customer Visit
…	…	…	…	…	…	…	…
1151	12/31/2016 13:24	12/31/2016 13:42	Business	Kar?chi	Unknown Location	3.9	Temporary Site
1152	12/31/2016 15:03	12/31/2016 15:38	Business	Unknown Location	Unknown Location	16.2	Meeting
1153	12/31/2016 21:32	12/31/2016 21:50	Business	Katunayake	Gampaha	6.4	Temporary Site
1154	12/31/2016 22:08	12/31/2016 23:51	Business	Gampaha	Ilukwatta	48.2	Temporary Site
1155	Totals	NaN	NaN	NaN	NaN	12204.7	NaN

1156 rows × 7 columns

In [173…
```
df.head()
```

Out[173]:

	START_DATE	END_DATE	CATEGORY	START	STOP	MILES	PURPOSE
0	01-01-2016 21:11	01-01-2016 21:17	Business	Fort Pierce	Fort Pierce	5.1	Meal/Entertain
1	01-02-2016 01:25	01-02-2016 01:37	Business	Fort Pierce	Fort Pierce	5.0	NaN
2	01-02-2016 20:25	01-02-2016 20:38	Business	Fort Pierce	Fort Pierce	4.8	Errand/Supplies
3	01-05-2016 17:31	01-05-2016 17:45	Business	Fort Pierce	Fort Pierce	4.7	Meeting
4	01-06-2016 14:42	01-06-2016 15:49	Business	Fort Pierce	West Palm Beach	63.7	Customer Visit

In [174…
```
df.describe()
```

Out[174]:

	MILES
count	1156.000000
mean	21.115398
std	359.299007
min	0.500000
25%	2.900000
50%	6.000000
75%	10.400000
max	12204.700000

In [175…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1156 entries, 0 to 1155
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   START_DATE  1156 non-null   object 
 1   END_DATE    1155 non-null   object 
 2   CATEGORY    1121 non-null   object 
 3   START       1155 non-null   object 
 4   STOP        1155 non-null   object 
 5   MILES       1156 non-null   float64
 6   PURPOSE     652 non-null    object 
dtypes: float64(1), object(6)
memory usage: 63.3+ KB
```

In [176…
```
df[df.duplicated()]
```

Out[176]:

	START_DATE	END_DATE	CATEGORY	START	STOP	MILES	PURPOSE
492	6/28/2016 23:34	6/28/2016 23:59	Business	Durham	Cary	9.9	Meeting

In [177…
```
# Remove duplicate rows
df = df.drop_duplicates()
```

In [178…
```
df[df.duplicated()]
```

Out[178]:

	START_DATE	END_DATE	CATEGORY	START	STOP	MILES	PURPOSE

In [179…
```
df.isnull().sum()
```

Out[179]:
```
START_DATE      0
END_DATE        1
CATEGORY       35
START           1
STOP            1
MILES           0
PURPOSE       504
dtype: int64
```

In [180…
```
#calculate the percentage of missing values in each column
(df.isnull().sum()/(len(df)))*100
```

Out[180]:
```
START_DATE     0.000000
END_DATE       0.086580
CATEGORY       3.030303
START          0.086580
STOP           0.086580
MILES          0.000000
PURPOSE       43.636364
dtype: float64
```

In [181…
```
df['CATEGORY'].value_counts()
```

Out[181]:
```
Business    1046
Personal      74
Name: CATEGORY, dtype: int64
```

In [182…
```
#filling in the most probable value
df.fillna({'CATEGORY':'Business'},inplace=True)
```

```
<ipython-input-182-879855af01fa>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  df.fillna({'CATEGORY':'Business'},inplace=True)
```

In [183…
```
df['PURPOSE'].value_counts()
```

Out[183]:
```
Meeting            185
Meal/Entertain     160
Errand/Supplies    128
Customer Visit     101
Temporary Site      50
Between Offices     18
Moving               4
Airport/Travel       3
Charity ($)          1
Commute              1
Name: PURPOSE, dtype: int64
```

In [184…
```
#filling in the most probable value
df.fillna({'PURPOSE':'Meeting'},inplace=True)
```

```
<ipython-input-184-9353bd30d714>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  df.fillna({'PURPOSE':'Meeting'},inplace=True)
```

In [185…
```
df.isnull().sum()
```

Out[185]:
```
START_DATE    0
END_DATE      1
CATEGORY      0
START         1
STOP          1
MILES         0
PURPOSE       0
dtype: int64
```

In [186…
```
df = df.dropna(subset=["END_DATE"])
df = df.dropna(subset=["START"])
df = df.dropna(subset=["STOP"])
```

In [187…
```
df.isnull().sum()
```

Out[187]:
```
START_DATE    0
END_DATE      0
CATEGORY      0
START         0
STOP          0
MILES         0
PURPOSE       0
dtype: int64
```

In [188…
```
unknown_locations = df[(df['START'] == 'Unknown Location') | (df['STOP'] == 'Unknown L

# Print the filtered DataFrame
print(unknown_locations)
```

```
            START_DATE          END_DATE  CATEGORY            START  \
108     2/16/2016 3:21    2/16/2016 4:13  Business        Katunayaka
109     2/16/2016 8:29    2/16/2016 9:34  Business  Unknown Location
116    2/16/2016 17:40   2/16/2016 17:44  Business          Nugegoda
117    2/17/2016 13:18   2/17/2016 14:04  Business  Unknown Location
121     2/18/2016 8:19    2/18/2016 8:27  Business  Unknown Location
...                ...               ...       ...               ...
1141  12/29/2016 19:50  12/29/2016 20:10  Business  Unknown Location
1143  12/29/2016 20:53  12/29/2016 21:42  Business           Kar?chi
1144  12/29/2016 23:14  12/29/2016 23:47  Business  Unknown Location
1151  12/31/2016 13:24  12/31/2016 13:42  Business           Kar?chi
1152  12/31/2016 15:03  12/31/2016 15:38  Business  Unknown Location

                  STOP  MILES          PURPOSE
108   Unknown Location   43.7   Customer Visit
109            Colombo   14.1          Meeting
116   Unknown Location    3.6  Errand/Supplies
117            Colombo   14.7   Temporary Site
121   Unknown Location   23.5   Temporary Site
...                ...    ...              ...
1141           Kar?chi    4.1   Customer Visit
1143  Unknown Location    6.4          Meeting
1144           Kar?chi   12.9          Meeting
1151  Unknown Location    3.9   Temporary Site
1152  Unknown Location   16.2          Meeting

[211 rows x 7 columns]
```

In [189…

```python
df = df.drop(unknown_locations.index)

# Reset index after dropping rows
df = df.reset_index(drop=True)
```

In [190…

```python
df
```

Out[190]:

	START_DATE	END_DATE	CATEGORY	START	STOP	MILES	PURPOSE
0	01-01-2016 21:11	01-01-2016 21:17	Business	Fort Pierce	Fort Pierce	5.1	Meal/Entertain
1	01-02-2016 01:25	01-02-2016 01:37	Business	Fort Pierce	Fort Pierce	5.0	Meeting
2	01-02-2016 20:25	01-02-2016 20:38	Business	Fort Pierce	Fort Pierce	4.8	Errand/Supplies
3	01-05-2016 17:31	01-05-2016 17:45	Business	Fort Pierce	Fort Pierce	4.7	Meeting
4	01-06-2016 14:42	01-06-2016 15:49	Business	Fort Pierce	West Palm Beach	63.7	Customer Visit
...	...	...	...	...	...	...	...
938	12/30/2016 16:45	12/30/2016 17:08	Business	Kar?chi	Kar?chi	4.6	Meeting
939	12/30/2016 23:06	12/30/2016 23:10	Business	Kar?chi	Kar?chi	0.8	Customer Visit
940	12/31/2016 1:07	12/31/2016 1:14	Business	Kar?chi	Kar?chi	0.7	Meeting
941	12/31/2016 21:32	12/31/2016 21:50	Business	Katunayake	Gampaha	6.4	Temporary Site
942	12/31/2016 22:08	12/31/2016 23:51	Business	Gampaha	Ilukwatta	48.2	Temporary Site

943 rows × 7 columns

In [191…

```
output = []
for col in df.columns:
    unique = df[col].nunique()
    colType = str(df[col].dtype)
    categories=df[col].unique()

    output.append([col, unique, colType,categories])

output = pd.DataFrame(output)
output.columns = ['colName','unique','dtype','categories']
output
```

Out[191]:

	colName	unique	dtype	categories
0	START_DATE	943	object	[01-01-2016 21:11, 01-02-2016 01:25, 01-02-201...
1	END_DATE	943	object	[01-01-2016 21:17, 01-02-2016 01:37, 01-02-201...
2	CATEGORY	2	object	[Business, Personal]
3	START	175	object	[Fort Pierce, West Palm Beach, Cary, Jamaica, ...
4	STOP	187	object	[Fort Pierce, West Palm Beach, Palm Beach, Car...
5	MILES	229	float64	[5.1, 5.0, 4.8, 4.7, 63.7, 4.3, 7.1, 0.8, 8.3,...
6	PURPOSE	10	object	[Meal/Entertain, Meeting, Errand/Supplies, Cus...

In [192...
```
print(np.max(df['MILES']))
print(np.min(df['MILES']))
```

```
310.3
0.5
```

In [193...
```
df['MILES bins'] = pd.cut(x=df['MILES'], bins=[0,50,100,150,200,250,300,350])
```

In [194...
```
df['MILES bins'].value_counts()
```

Out[194]:
```
(0, 50]       924
(50, 100]       7
(100, 150]      5
(150, 200]      5
(200, 250]      1
(300, 350]      1
(250, 300]      0
Name: MILES bins, dtype: int64
```

In [215...
```
df = df.rename(columns={'MILES': 'distance_travelled'})
```

In [216...
```
df
```

Out[216]:

	START_DATE	END_DATE	CATEGORY	START	STOP	distance_travelled	PURPOSE	M
0	01-01-2016 21:11	01-01-2016 21:17	Business	Fort Pierce	Fort Pierce	5.1	Meal/Entertain	(
1	01-02-2016 01:25	01-02-2016 01:37	Business	Fort Pierce	Fort Pierce	5.0	Meeting	(
2	01-02-2016 20:25	01-02-2016 20:38	Business	Fort Pierce	Fort Pierce	4.8	Errand/Supplies	(
3	01-05-2016 17:31	01-05-2016 17:45	Business	Fort Pierce	Fort Pierce	4.7	Meeting	(
4	01-06-2016 14:42	01-06-2016 15:49	Business	Fort Pierce	West Palm Beach	63.7	Customer Visit	
...	...	...	...	...	...	...	...	
938	12/30/2016 16:45	12/30/2016 17:08	Business	Kar?chi	Kar?chi	4.6	Meeting	(
939	12/30/2016 23:06	12/30/2016 23:10	Business	Kar?chi	Kar?chi	0.8	Customer Visit	(
940	12/31/2016 1:07	12/31/2016 1:14	Business	Kar?chi	Kar?chi	0.7	Meeting	(
941	12/31/2016 21:32	12/31/2016 21:50	Business	Katunayake	Gampaha	6.4	Temporary Site	(
942	12/31/2016 22:08	12/31/2016 23:51	Business	Gampaha	Ilukwatta	48.2	Temporary Site	(

943 rows × 8 columns

In [210…

```
pd.crosstab(df['CATEGORY'], df['PURPOSE'])
```

Out[210]:

PURPOSE	Airport/Travel	Between Offices	Charity ($)	Commute	Customer Visit	Errand/Supplies	Meal/Entertain	I
CATEGORY								
Business	1	18	0	0	92	111	148	
Personal	0	0	1	1	0	0	0	

In [214…

In [208…

In [208…

In [ ]: