Report of Project 1

To understand before starting working on it, we briefly explore the data.

We import the basic libraries that we will be using throughout the program, namely pandas, numpy and copy. We set Matplotlib to plot inline, which means that the output plot will appear immediately under each code cell.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
import copy
%matplotlib inline
```

We read the excel file containing data into a dataframe ‘df’

```
[71] df = pd.read_excel('Credit_cards_App_data.xlsx')
     df.head()
```

The next step is to gather some information about different columns in our DataFrame. We use ‘info()’ which gives us information about the number of rows, columns, column data types, memory usage, etc.

```
print(df_copy.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 18 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Application id           1000 non-null   int64
 1   first_name               1000 non-null   object
 2   last_name                1000 non-null   object
 3   email                    1000 non-null   object
 4   gender                   1000 non-null   object
 5   address                  1000 non-null   object
 6   age                      1000 non-null   int64
 7   tdecision                1000 non-null   object
 8   empstaus                 1000 non-null   object
 9   ExCus (Customer in Past) 1000 non-null   object
 10  Source                   1000 non-null   object
 11  Salary                   1000 non-null   int64
 12  ExDebt (Liability)       1000 non-null   int64
 13  Booking                  699 non-null    object
 14  INT_ID                   1000 non-null   int64
 15  Prev_ID                  1000 non-null   object
 16  AGT_ID                   1000 non-null   object
 17  Booking_Amt              699 non-null    float64
dtypes: float64(1), int64(5), object(12)
memory usage: 140.8+ KB
None
```

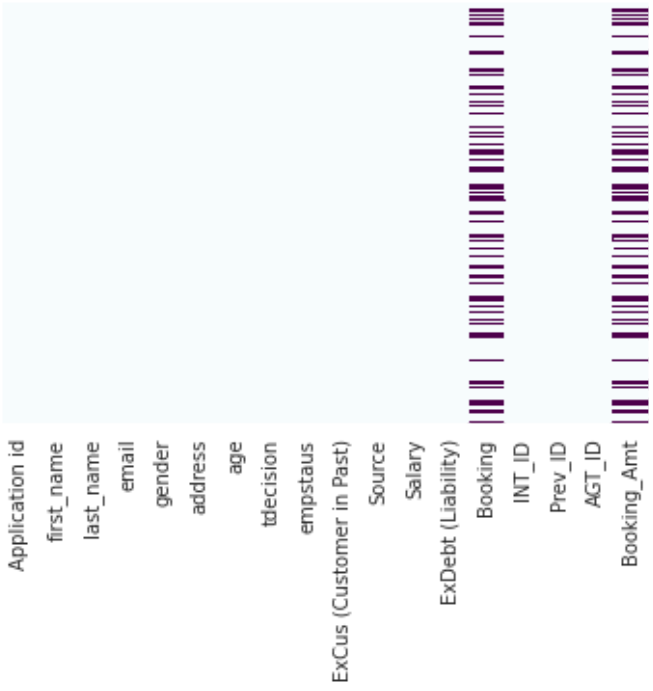
0s completed a

The columns with object dtype are the possible categorical features in our dataset.

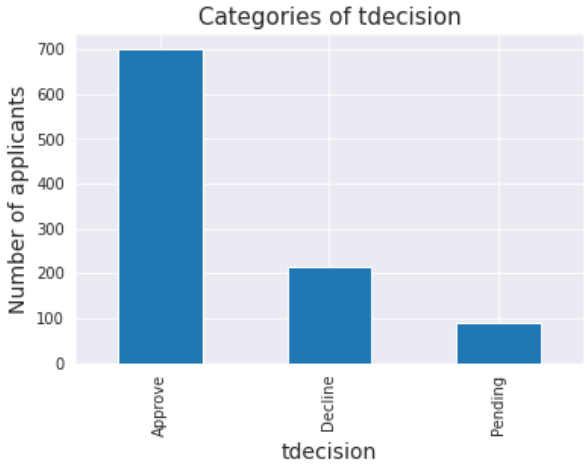
We check the total number of missing values in the DataFrame and the column-wise distribution of null values

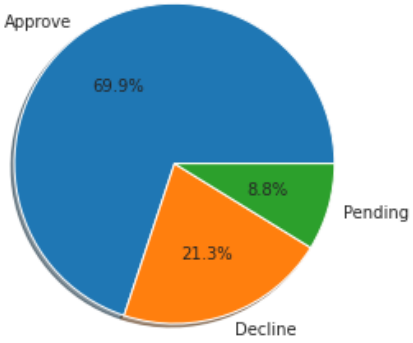
```
[76] df_copy.isnull().values.sum()                # returns total number of null values in df

602

     df_copy.isnull().sum()                       # returns number of null values in each column

Application id                0
first_name                    0
last_name                     0
email                         0
gender                        0
address                       0
age                           0
tdecision                     0
empstaus                      0
ExCus (Customer in Past)      0
Source                        0
Salary                        0
ExDebt (Liability)            0
Booking                     301
INT_ID                        0
Prev_ID                       0
AGT_ID                        0
Booking_Amt                 301
dtype: int64
```

We visualise the missing data using a heatmap. We observe that approximately 30% values of ‘Booking’ are missing. We will use the mode of the current values to replace the missing values.



We visualise tdecision as bar and pie charts. We also display the frequency of categories and the percentages of the same.






```
The number of values in tdecision:
Approve    699
Decline    213
Pending     88
Name: tdecision, dtype: int64
```

```
Approve    69.9
Decline    21.3
Pending     8.8
Name: tdecision, dtype: float64
```

We encode the categorical features of 'tdecision' to numeric quantities using one-hot-encoding. scikit-learn also supports one hot encoding via LabelBinarizer and OneHotEncoder in its preprocessing module. We can conveniently use it here because we only have 3 categories and hence will not face the curse of dimensionality.

```
[91] dfmatrix = df_copy.copy()

     from sklearn.preprocessing import LabelBinarizer

     lb = LabelBinarizer()
     lb_results = lb.fit_transform(dfmatrix['tdecision'])
     lb_results_df = pd.DataFrame(lb_results, columns=lb.classes_)

     print(lb_results_df)
```

```
     Approve  Decline  Pending
0          0        0        1
1          1        0        0
2          1        0        0
3          1        0        0
4          1        0        0
..       ...      ...      ...
995        0        0        1
996        1        0        0
997        0        0        1
998        0        0        1
999        0        1        0

[1000 rows x 3 columns]
```

We create a new dataframe ‘clean_df’ after replacing NaN values by the mode and dropping columns we do not need.

```
[97] df_copy['Booking'].fillna(df_copy['Booking'].mode()[0], inplace=True)
```

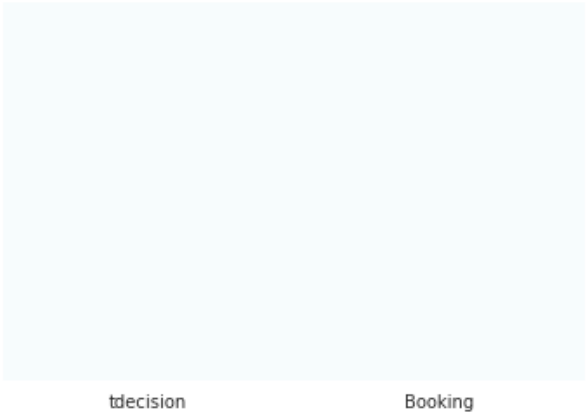
Creating new dataframe

```
[99] clean_df = pd.DataFrame(df_copy.iloc[:,[7,13]])      # clean df only has required columns
```

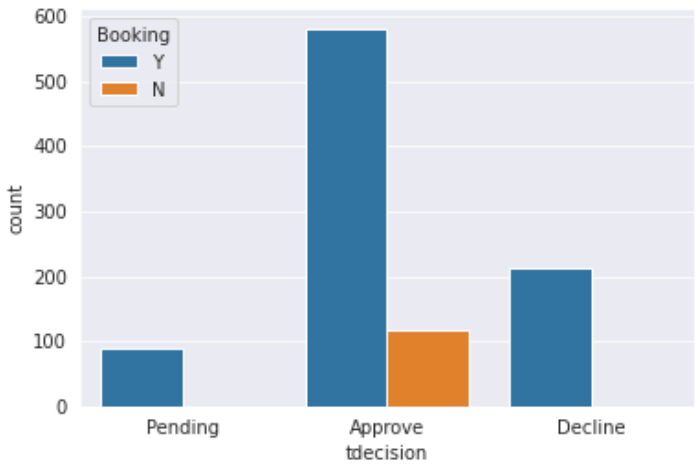
```
[100] clean_df.head()
```

	tdecision	Booking
0	Pending	Y
1	Approve	Y
2	Approve	Y
3	Approve	Y
4	Approve	Y

We ensure that all the NaN values have been replaced and visualise it using a heatmap.



We also visualise booking status according to tdecision categories as a bar plot



Last, we summarise the above as a form of a dataframe ‘status_df’

	tdecision	Y	N
0	Approve	581	118
1	Pending	88	0
2	Decline	213	0