TITANIC DATA ANALYSIS

"USING PYTHON"

by Matplotlib Group

OUR TEAM

01. TOBIAS MIKHA SULISTIYO
02. DAUD IBADURAHMAN
03. FITRI ALFAQRINA
04. PUTRI REGHINA HILMI PRASASTI

TITANIC DATA

Titanic data is passenger data that boarded the Titanic. In this data, there is information about Ticket, Age, Gender, And Passenger Status whether it is safe or not. We can use this data because there are interesting things contained in this data

USE CASE : TITANIC DATA ANALYSIS

Use Case Summary

OBJECTIVE STATEMENT

- Get insight about how many passengers of the Titanic ship who survived
- To find out what actions must be taken so that the same thing does not happen again on other ships
- To get the insight about how to cleansing Titanic data
- To get the new knowledge about how to do imputation in the missing data of Titanic data

SOURCE DATA

Online dataset by kaggle, Titanic.csv

CHALLENGES

- Large size of data, so it can't be managed by Excel or Spreadsheet
- Data have a lot of missing values
- Not all data in the Titanic data can be analyzed
- Python is a case sensitive programming language, so the writing of the syntax must be considered

METHODOLOGY/ ANALYTHIC TECHNIQUIE

- Descriptive analysis
- Graph analysis

BENEFITS

- As a basic material for learning Data Science for begginers
- As additional knowledge about how to get the insight from a data

EXCPECTED OUTCOME

- Know how to import, imputation, and modify Titanic data using Python
- Know how to analyze dat in graphical form
- Conclusion from the results of the titanic data analysis

Data Cleansing

DATA UNDERSTANDING

- Dataset of the Titanic shipwreck was a massive disaster to know the number of survivors and their details.
- Source data : Online dataset by kaggle, Titanic.csv
- The dataset has 12 columns and 891 rows.

DATA UNDERSTANDING

- Data Dictionary
- PassengerId : Number of passenger
- Survived : Outcome of survival (0 = No; 1 = Yes)
- Pclass : Socio-economic class (1 = Upper; 2 = Middle; 3 = Lower)
- Name : First name and last name of passenger
- Sex : Sex of passenger (Male or Female)
- Age : Age of passenge
- SibSp : Number of siblings and spouses of passenger aboard
- Parch : Number of parents and children of passenger aboard
- Ticket : Ticket number of passenger
- Fare : Fare paid by the passenger
- Cabin : Cabin number of the passenger
- Embarked : Part of embarkation of the passenger (C = Cherbourg; Q = Queenstown; S = Southampton)

Data Cleansing

DATA PREPARATION

Code Used :

- Python in Google colab (https://colab.research.google.com/).
- Packages : Pandas, Numpy, Matplotlib, and Seaborn

Data Cleansing

DATA CLEANSING

- Check for each column and find if any column with missing values.
- There are about 20% of Null Age and 2 rows of Null Embark in the data. We need to inputation them.
- Need to remove of Cabin column as there is have a lot of unique data of Cabin.
- Need to manipulation of Column SibSp and Column Parch.
- Check for value that a the relationship between column Sex and column Survived.

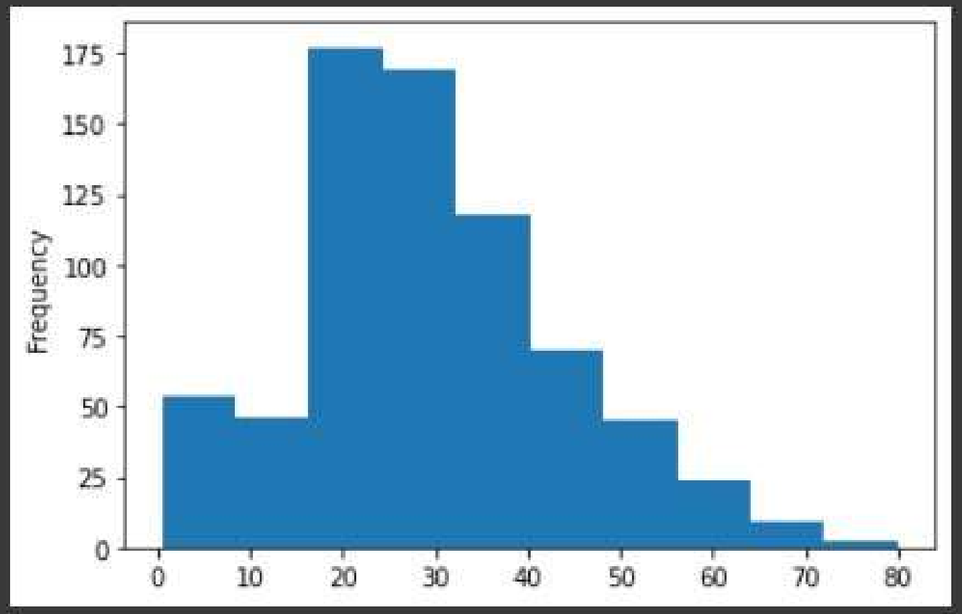
Data Cleansing

INPUTATION OF COLUMN AGE AND COLUMN EMBARK

COLUMN AGE

Because the Age column has a skewness distribution then we will inputation (enter data) in the Age column with the median (the middle value after sorted).

```
df['Age'].plot(kind='hist');
```



COLUMN AGE

The middle value after sorted can be seen by displaying a description of the quantitative statistics data for Age column.

```
df['Age'].describe()
```

```
count    714.000000
mean      29.699118
std       14.526497
min        0.420000
25%       20.125000
50%       28.000000
75%       38.000000
max       80.000000
Name: Age, dtype: float64
```

COLUMN EMBARK

Show the proportion of the Embarked column. It turns out that the Embarked column is the categorical data. We will inputation in the Embark column with the value that often arise (modus)

```
df.Embarked.value_counts()
```

```
S    644
C    168
Q     77
Name: Embarked, dtype: int64
```

```
df.Embarked.describe()
```

```
count      889
unique       3
top          S
freq       644
Name: Embarked, dtype: object
```

Data Cleansing

COLUMN SIBSP AND COLUMN PARCH

Data Cleansing

02.

COLUMN SIBSP AND COLUMN PARCH

In the SibSp and Parch column, we will do manipulation data. It is not to change the data value but to make this data easier to read by machine.

Column SibSp (sibling Spouse) means the column that states the number of siblings or the number of partners carried by Passenger. Column Parch (Parent Children) means column which states the number of parents or the number of children carried by Passenger

We will make a new column that displays whether he is alone or with his family.

Data Cleansing

02.

COLUMN SIBSP AND COLUMN PARCH

we want to make a column who shows passengers alone or going with family, we made a code and show like this:

```
df['Status'] = df['SibSp'] + df['Parch']
```

```
df['Status'][df['Status']>0] = 'With Familiy'
df['Status'][df['Status']==0] = 'Without Family'
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

Data Cleansing

02.

COLUMN SIBSP AND COLUMN PARCH

Voila, we had a column that show passengers with or without family, we checked with 'df.head()'

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

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked	Status
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	S	With Familiy
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C	With Familiy
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	S	Without Family
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	S	With Familiy
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	S	Without Family

EXPLORATORY DATA ANALYSIS

04.

Exloratory Data Analysis

EMBARKED

Exploratory Data Analysis

02.

EMBARKED

After analyzed, it turns out that there are 2 missing embarked data from our titanic data.

Therefore, we can used imputation to fill the 2 data missing in Embarked column

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

```
Survived      0
Pclass        0
Name          0
Sex           0
Age         177
SibSp         0
Parch         0
Ticket        0
Fare          0
Cabin       687
Embarked      2
dtype: int64
```

Exploratory Data Analysis

02.

EMBARKED

```
df.Embarked.value_counts()
```

```
S    644
C    168
Q     77
Name: Embarked, dtype: int64
```

Embarked column proportion

If we are going to do imputation on the Embarked column, we should check the data type of the Embarked column first.

The data in the Embarked column is in the form of categorical data, then the imputation uses mode

Exploratory Data Analysis

02.

EMBARKED

From the proportion of the Embarked column, S is the data that appears frequently, then S is the mode in the Embarked column. Then we can use category S to fill in 2 missing data from the Embarked column

```
val = df.Embarked.mode().values[0]
df['Embarked'] = df.Embarked.fillna(val)
```

Exploratory Data Analysis

02.

EMBARKED

It can be seen that there are 891 data from the 891 Titanic data that we used. It means that there is no more missing data in the Embarked column.

But, the data in the embarked column still an object data type.

```
df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 891 entries, 1 to 891
Data columns (total 11 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Survived  891 non-null    int64
 1   Pclass    891 non-null    int64
 2   Name      891 non-null    object
 3   Sex       891 non-null    object
 4   Age       714 non-null    float64
 5   SibSp     891 non-null    int64
 6   Parch     891 non-null    int64
 7   Ticket    891 non-null    object
 8   Fare      891 non-null    float64
 10  Embarked  891 non-null    object
memory usage: 83.5+ KB
```

Exploratory Data Analysis

02.

EMBARKED

Therefore, to facilitate the analysis process, we will convert the object type to a numeric type.

```
df.Embarked = df.Embarked.map({'S':0, 'C':1, 'Q':2})
```

We use a value of 0 for S, value of 1 for C and value of 2 for Q chategory.

```
df.Embarked.value_counts()

S    646
C    168
Q     77
```

Before

```
df.Embarked.value_counts()

0    646
1    168
2     77
Name: Embarked, dtype: int64
```

After

04.

Exloratory Data Analysis

AGE

Exploratory Data Analysis

02.

AGE

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 891 entries, 1 to 891
Data columns (total 11 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Survived  891 non-null    int64
 1   Pclass    891 non-null    int64
 2   Name      891 non-null    object
 3   Sex       891 non-null    object
 4   Age       714 non-null    float64
 5   SibSp     891 non-null    int64
 6   Parch     891 non-null    int64
 7   Ticket    891 non-null    object
 8   Fare      891 non-null    float64
 9   Cabin     204 non-null    object
 10  Embarked  891 non-null    int64
dtypes: float64(2), int64(5), object(4)
memory usage: 83.5+ KB
```

On our Titanic data, there is 891 data. but, our data in column age just have 714 data.

To fill the 177 data left, we used to fill the data with inputation.

Exploratory Data Analysis

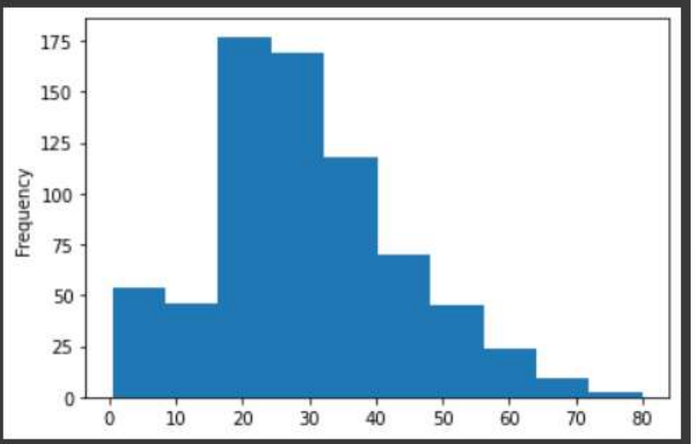
02.

AGE

Show our age data

Histogram

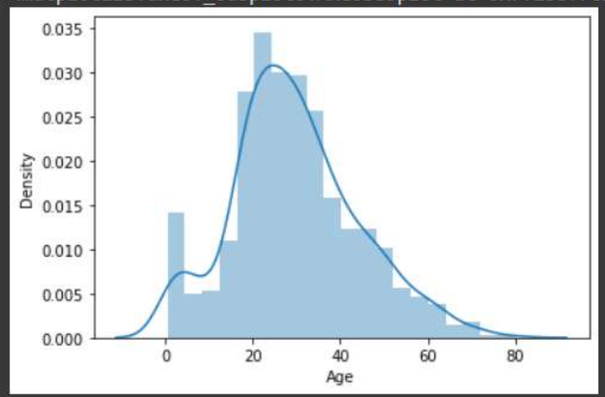
```
df['Age'].plot(kind='hist');
```



Distplot

```
sns.distplot(df['Age'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f1b8f7cb5d0>
```



Age Value

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

```
24.00    30
22.00    27
18.00    26
19.00    25
28.00    25
         ..
36.50     1
55.50     1
0.92      1
23.50     1
74.00     1
Name: Age, Length: 88, dtype: int64
```

Exploratory Data Analysis

02.

AGE

Because the age column had an skewness distribution, we do an inputation (adding data) for the column age with median. And then, our Age data had 891 like others data

```
val = df.Age.median()
df['Age'] = df.Age.fillna(val)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 891 entries, 1 to 891
Data columns (total 11 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Survived  891 non-null    int64
 1   Pclass    891 non-null    int64
 2   Name      891 non-null    object
 3   Sex       891 non-null    object
 4   Age       891 non-null    float64
 5   SibSp     891 non-null    int64
 6   Parch     891 non-null    int64
 7   Ticket    891 non-null    object
 8   Fare      891 non-null    float64
 9   Cabin     204 non-null    object
 10  Embarked  891 non-null    int64
dtypes: float64(2), int64(5), object(4)
memory usage: 83.5+ KB
```

Exploratory Data Analysis

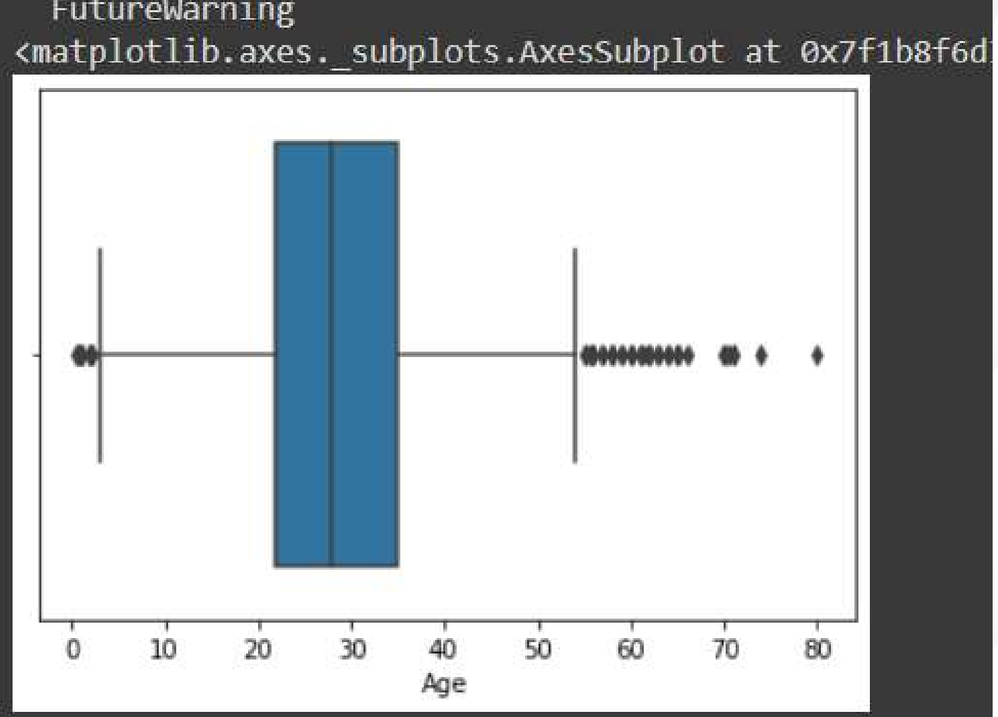
02.

AGE

we made an outlier visualization for column age with boxplot.

```
sns.boxplot(df['Age'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_deco
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f1b8f6d
```



04.

Exloratory Data Analysis

CABIN

Exploratory Data Analysis

02.

CABIN

```
[ ]  df['Cabin'].value_counts()

B96 B98        4
G6             4
C23 C25 C27    4
C22 C26        3
F33            3
              ..
E34            1
C7             1
C54            1
E36            1
C148           1
Name: Cabin, Length: 147, dtype: int64
```

We can see that our data in cabin column had a unique data and the data isn't informative to know about data survive. because the data isn't informative, we can drop (clear) the data

Exploratory Data Analysis

02.

CABIN

If there is too many data missing, then you should drop (clear) the data from the column to avoid bias data.

```
df.drop('Cabin', axis=1, inplace = True)
```

we use 'df.drop()' to clear the cabin data and we use 'inplace = True" to clear cabin column permanently.

Exploratory Data Analysis

02.

CABIN

Voila!

We finally clear the cabin data, so there's no bias data in our column because cabin data isn't that informative so we throw it out.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 891 entries, 1 to 891
Data columns (total 10 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Survived  891 non-null    int64  
 1   Pclass    891 non-null    int64  
 2   Name      891 non-null    object 
 3   Sex       891 non-null    object 
 4   Age       891 non-null    float64
 5   SibSp     891 non-null    int64  
 6   Parch     891 non-null    int64  
 7   Ticket    891 non-null    object 
 8   Fare      891 non-null    float64
 9   Embarked  891 non-null    int64  
dtypes: float64(2), int64(5), object(3)
memory usage: 76.6+ KB
```

04.

Exloratory Data Analysis

NAME

NAME

```
df['Name'].value_counts()
```

```
Braund, Mr. Owen Harris                     1
Boulos, Mr. Hanna                           1
Frolicher-Stehli, Mr. Maxmillian            1
Gilinski, Mr. Eliezer                       1
Murdlin, Mr. Joseph                         1
                                           ..
Kelly, Miss. Anna Katherine "Annie Kate"    1
McCoy, Mr. Bernard                          1
Johnson, Mr. William Cahoone Jr             1
Keane, Miss. Nora A                         1
Dooley, Mr. Patrick                         1
Name: Name, Length: 891, dtype: int64
```

We can see that our data in Name column had too many unique data .

Exploratory Data Analysis

02.

NAME

The Name data isn't informative to know about data survive. Because the data isn't informative, we can drop (clear) it.

```
df.drop('Name', axis=1, inplace=True)
```

we use 'df.drop()' to clear the Name column and use 'inplace = True" to clear Name column permanently.

NAME

Finally !
The name column has been droped from Titanic data.

So we can move on to the next step.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 891 entries, 1 to 891
Data columns (total 9 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Survived  891 non-null    int64  
 1   Pclass    891 non-null    int64  
 2   Sex       891 non-null    object 
 3   Age       891 non-null    float64
 4   SibSp     891 non-null    int64  
 5   Parch     891 non-null    int64  
 6   Ticket    891 non-null    object 
 7   Fare      891 non-null    float64
 8   Embarked  891 non-null    int64  
dtypes: float64(2), int64(5), object(2)
memory usage: 69.6+ KB
```

04.

Exloratory Data Analysis

SEX

SEX

The data in the Sex column still an object data type.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 891 entries, 1 to 891
Data columns (total 9 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Survived  891 non-null    int64
 2   Sex       891 non-null    object
 4   SibSp     891 non-null    int64
 5   Parch     891 non-null    int64
 6   Ticket    891 non-null    object
 7   Fare      891 non-null    float64
 8   Embarked  891 non-null    int64
dtypes: float64(2), int64(5), object(2)
memory usage: 69.6+ KB
```

SEX

Therefore, to facilitate the analysis process, we will convert the object type to a numeric type.

```
df.Sex = df.Sex.map({'male':0, 'female':1})
```

We use a value of 0 for male and a value of 1 for female.

```
df['Sex'].value_counts()

male      577
female    314
```

Before

```
df['Sex'].value_counts()

0    577
1    314
Name: Sex, dtype: int64
```

After

04.

Exploratory Data Analysis

TICKET

Exploratory Data Analysis

02.

TICKET

```
df['Ticket'].value_counts()
```

```
347082      7
CA. 2343    7
1601        7
3101295     6
CA 2144     6
           ..
9234        1
19988       1
2693        1
PC 17612    1
370376      1
Name: Ticket, Length: 681, dtype: int64
```

We can see that our data in Ticket column had too many unique data .

Exploratory Data Analysis

02.

TICKET

The Ticket data isn't informative to know about data survive. Because the data isn't informative, we can drop (clear) it.

```
df.drop('Ticket', axis=1, inplace=True)
```

we use 'df.drop()' to clear the Ticket column and use 'inplace = True" to clear Ticket column permanently.

Exploratory Data Analysis

02.

TICKET

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 891 entries, 1 to 891
Data columns (total 8 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Survived  891 non-null    int64
 1   Pclass    891 non-null    int64
 2   Sex       891 non-null    int64
 3   Age       891 non-null    float64
 4   SibSp     891 non-null    int64
 5   Parch     891 non-null    int64
 6   Fare      891 non-null    float64
 7   Embarked  891 non-null    int64
dtypes: float64(2), int64(6)
memory usage: 62.6 KB
```

Finally !

The Ticket column has been droped permanently from Titanic data..

04.

Exloratory Data Analysis

RELATIONSHIP BETWEEN SEX AND SURVIVED COLUMN

Exploratory Data Analysis

02.

SEX SURVIVED COUNTS

```
df.Sex = df.Sex.map({'male':0, 'female':1})
```

```
df['Sex'].value_counts()
```

```
0    577
1    314
Name: Sex, dtype: int64
```

```
[ ]  df.Sex[df['Survived']==1].value_counts()
```

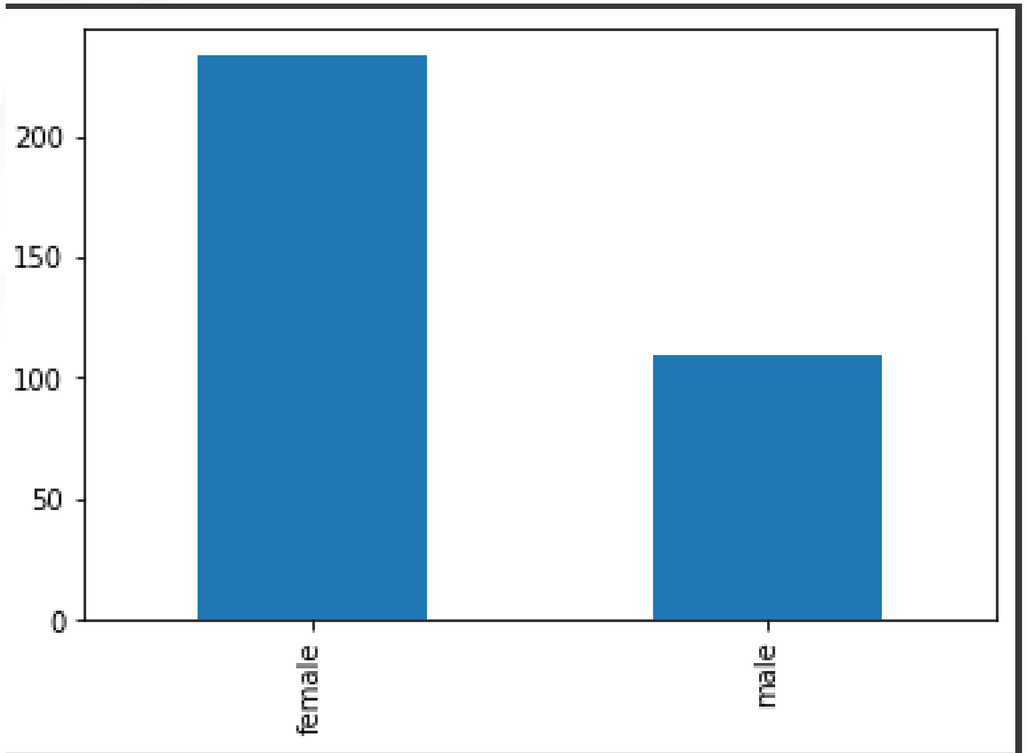
```
female    233
male      109
Name: Sex, dtype: int64
```

In this information we can see that from 891 passengers consisting of 577 Male and 314 Female. There are only 342 Passengers who survived consisting of 109 Male And 233 Female. We need to pay attention to the *difference* in the number of *passengers who boarded* the ship and the *passengers who survived*

Exploratory Data Analysis

02.

SEX SURVIVED COUNTS




```
female    233
male      109
Name: Sex, dtype: int64
```

From these data we can see that the number of female passengers who survived was greater than the number of male passengers who survived. We can analyze that at that time, female passengers were prioritized to be rescued over male passengers.

04.

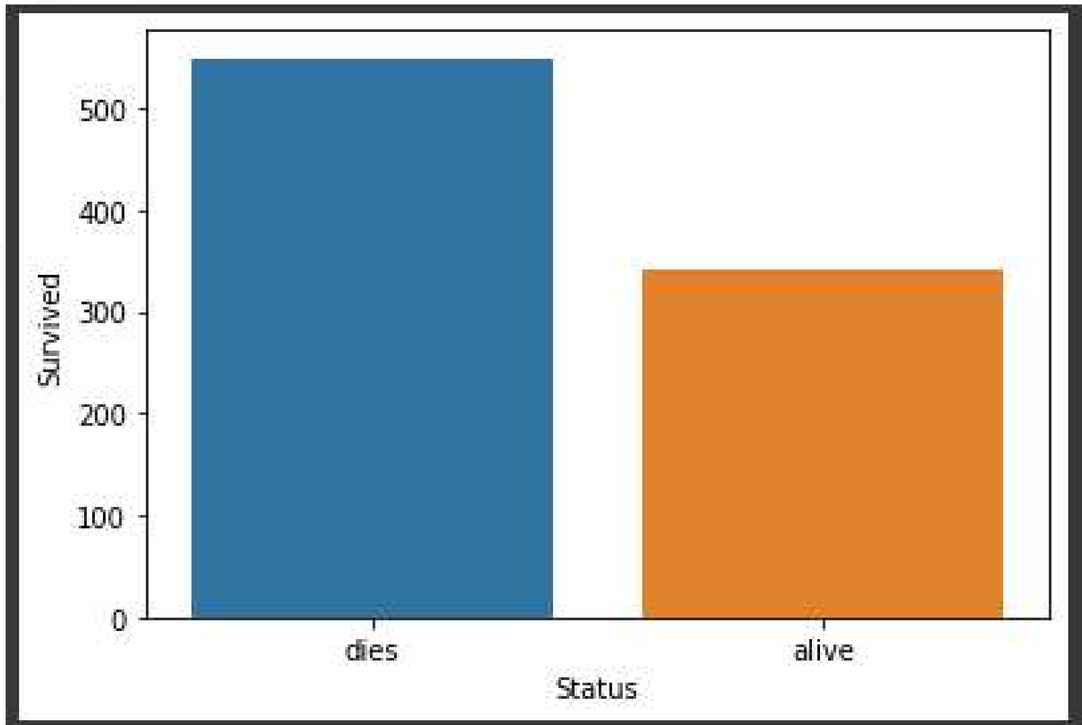
Exloratory Data Analysis

SURVIVED STATUS

Exploratory Data Analysis

02.

SURVIVED STATUS



	Survived	Status
0	549	dies
1	342	alive

From these data we can see that the difference number passengers who *dies* and *alive*. From these differences we can conclude that because more people died than alive, the safety and security of the Titanic was very bad.

CONCLUSION

CONCLUSION

From some of the existing data, we can conclude that not all data can be used for analysis. of the age range of passengers who boarded the titanic on average were productive ages (16-65 years). It can be concluded that they boarded the titanic to go to work at their destination.

CONCLUSION

From passenger data, we can see that more passengers died than survivors. So it can be concluded that the safety on the Titanic ship is bad. In addition, the number of female passengers who survived was more so that it can be concluded that female passengers are prioritized to be rescued than male passengers.

THANK YOU