1. Section A :

A. Rules of Inference:

Propositional Rules:

1. **Modus Ponens**: If p implies q, and p is true, then q must be true.
   Example:
   p: It is raining outside.
   q: The ground is wet.
   If it is true that it is raining outside (p), and we know that if it rains, the ground will be wet (p -> q), then we can conclude that the ground is wet (q).

2. **Modus Tollens**: If p implies q, and q is false, then p must also be false.
   Example:
   p: If it's a weekday, then I have work.
   q: I don't have work today.
   If I know that if it's a weekday (p), then I have work (p -> q), and I don't have work today (not q), then I can conclude that it's not a weekday (not p).

3. **Hypothetical Syllogism**: If p implies q, and q implies r, then p implies r.
   Example:
   p: If it rains, I will stay inside.
   q: If I stay inside, I will play a game.
   r: If it rains, I will play a game.
   If I know that if it rains (p), then I will stay inside (p -> q), and if I stay inside (q), then I will play a game (q -> r), then I can conclude that if it rains (p), then I will play a game (p -> r).

4. **Disjunctive Syllogism**: If p or q is true, and p is false, then q must be true.
   Example:
   p: It's raining outside.
   q: I am staying inside.
   If I know that it's either raining outside (p) or I'm staying inside (q), and it's not raining outside (not p), then I can conclude that I am staying inside (q).

Predicate Rules:

5. **Universal Instantiatio**n: If all members of a set have a certain property, then any one member of that set also has that property.
   Example:
   All dogs bark.
   Mark is a dog.
   Therefore, Mark barks.

If we know that all dogs bark, and Mark is a dog, then we can conclude that Mark barks.

6. **Universal Generalization**: If one member of a set has a certain property, then all members of that set also have that property.
   Example:
   Mark barks.
   Mark is a dog
   Therefore, all dogs bark.
   If we know that Mark barks, then we can conclude that all dogs bark.

7. **Existential Instantiation**: If there exists a member of a set with a certain property, then we can infer the existence of at least one member of that set.
   Example:
   Some dogs are black.
   Therefore, there exists at least one black dog.
   If we know that some dogs are black, then we can conclude that there exists at least one black dog.

8. **Existential Generalization**: If we have shown that a member of a set has a certain property, then we can infer that at least one member of that set has that property.
   Example:
   Mark barks.
   Mark is a dog
   Therefore, some dog barks.
   If we know that Mark barks, then we can conclude that some dog barks.

b. Application of ML in various scenarios:

i. Yes, ML should be applied as approving or rejecting a person's credit card application is mostly based on the information about the person and his cibil score, ability to repay the credit bills and worthiness to take up a credit card. These things are mostly based on statistics and data and hence ML can be useful in carrying out these tasks and filtering out credit worthy people from those not worthy of a credit card. But there may be biases such as giving better credit scores to a men than women due to the data provided.

ii. No, ML should not be completely relies on as human life is at stake and if the data provided is incomplete or inaccurate, then a lot can be lost. It is also important to note that different diseases may have the same symptoms but may have different treatement and precautionary measures, so the data provided needs to take into account all these possibilities and only then be provided to the ML.

iii. Yes, ML can be used to track students' academic performance and provide personalised recommendations for improvement as it mainly relies on individual student data.

Eg:- Which subject is he getting lesser marks?, How interactively is he participating in the classes?, etc.
But sometimes, the data provided might be inaccurate due to reasons like the teacher's teaching, etc.

iv. No, ML might not be a good idea to be used in identifying potential suspects in a criminal investigation as it may pick up patterns in certain criminals and apply it to all the people, which may lead to a biased suspect list.
Eg:- A particular ML may pick a pattern of having a beard in most criminals and may apply it to all the people, and certain innocent people might come in the suspects list for the only reason of having a beard.

In general, we can see that almost in all cases, ML can be applied, but its benefits and risks are associated with it. ML can be applied only after calculating the risk analysis, mitigating as much risk as possible, and providing large, accurate, and precise data for the ML to work on.

c. Turing Test:

Turing Test is the assessment used to test a machine's intelligence. In this test, a human and a machine are made to interact with a human evaluator in natural language. If the evaluator cannot distinguish between the machine's and human responses, the machine is said to have passed the test. The turing test essentially implies that the machine has become intelligent enough to think and respond like a human in some cases.

The test involves a human evaluator who engages in a natural language conversation with a machine and a human. The evaluator does not know which is the machine and which is the human. If the evaluator cannot distinguish the machine's responses from the human's responses, then the machine is said to have passed the Turing Test and demonstrated a level of intelligence that is equivalent to a human.
It does not essentially mean that the machine is as intelligent as humans, it simply implies that machine is able to generate human like responses and is able to mimic humans. Even though this is a breakthrough in the intelligence of machines, they still have a long way to go to acheive human like intelligence.

2. Section B :

a.

```python
import pandas as pd
# importing library

df = pd.read_csv("/content/train.csv")
# reading from the file

print(df.shape) # printing the shape
print(df.head(10)) # printing the first 10 entries
print(df.tail(10)) # printing the last 10 entries
```

```
(891, 12)
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3
5            6         0       3
6            7         0       1
7            8         0       3
8            9         1       3
9           10         1       2

                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0
5                                   Moran, Mr. James    male   NaN      0
6                            McCarthy, Mr. Timothy J    male  54.0      0
7                     Palsson, Master. Gosta Leonard    male   2.0      3
8  Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)  female  27.0      0
9                Nasser, Mrs. Nicholas (Adele Achem)  female  14.0      1

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
```

0s completed at 11:29 PM

```
[9]
4                          Allen, Mr. William Henry    male  35.0      0
5                                 Moran, Mr. James    male   NaN      0
6                          McCarthy, Mr. Timothy J    male  54.0      0
7                   Palsson, Master. Gosta Leonard    male   2.0      3
8  Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)  female  27.0      0
9                Nasser, Mrs. Nicholas (Adele Achem)  female  14.0      1

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S
5      0            330877   8.4583   NaN        Q
6      0             17463  51.8625   E46        S
7      1            349909  21.0750   NaN        S
8      2            347742  11.1333   NaN        S
9      0            237736  30.0708   NaN        C
     PassengerId  Survived  Pclass                                      Name  \
881          882         0       3                          Markun, Mr. Johann
882          883         0       3                 Dahlberg, Miss. Gerda Ulrika
883          884         0       2                Banfield, Mr. Frederick James
884          885         0       3                       Sutehall, Mr. Henry Jr
885          886         0       3         Rice, Mrs. William (Margaret Norton)
886          887         0       2                        Montvila, Rev. Juozas
887          888         1       1                 Graham, Miss. Margaret Edith
888          889         0       3     Johnston, Miss. Catherine Helen "Carrie"
889          890         1       1                         Behr, Mr. Karl Howell
890          891         0       3                           Dooley, Mr. Patrick

        Sex   Age  SibSp  Parch            Ticket     Fare Cabin Embarked
881    male  33.0      0      0            349257   7.8958   NaN        S
882  female  22.0      0      0              7552  10.5167   NaN        S
883    male  28.0      0      0  C.A./SOTON 34068  10.5000   NaN        S
884    male  25.0      0      0   SOTON/OQ 392076   7.0500   NaN        S
885  female  39.0      0      5            382652  29.1250   NaN        Q
886    male  27.0      0      0            211536  13.0000   NaN        S
887  female  19.0      0      0            112053  30.0000   B42        S
888  female   NaN      1      2        W./C. 6607  23.4500   NaN        S
889    male  26.0      0      0            111369  30.0000  C148        C
890    male  32.0      0      0            370376   7.7500   NaN        Q
```

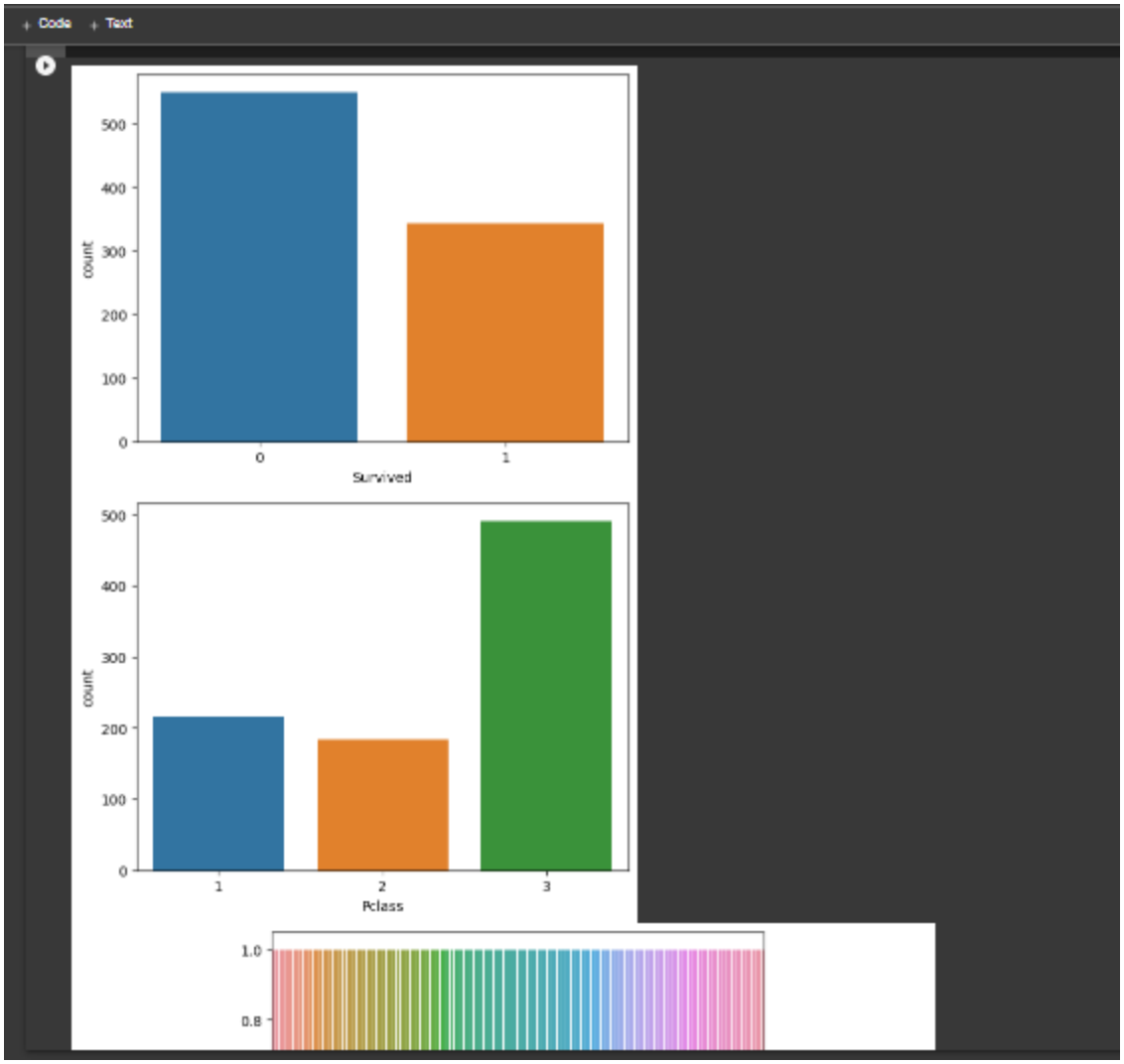
0s completed at 11:29 PM

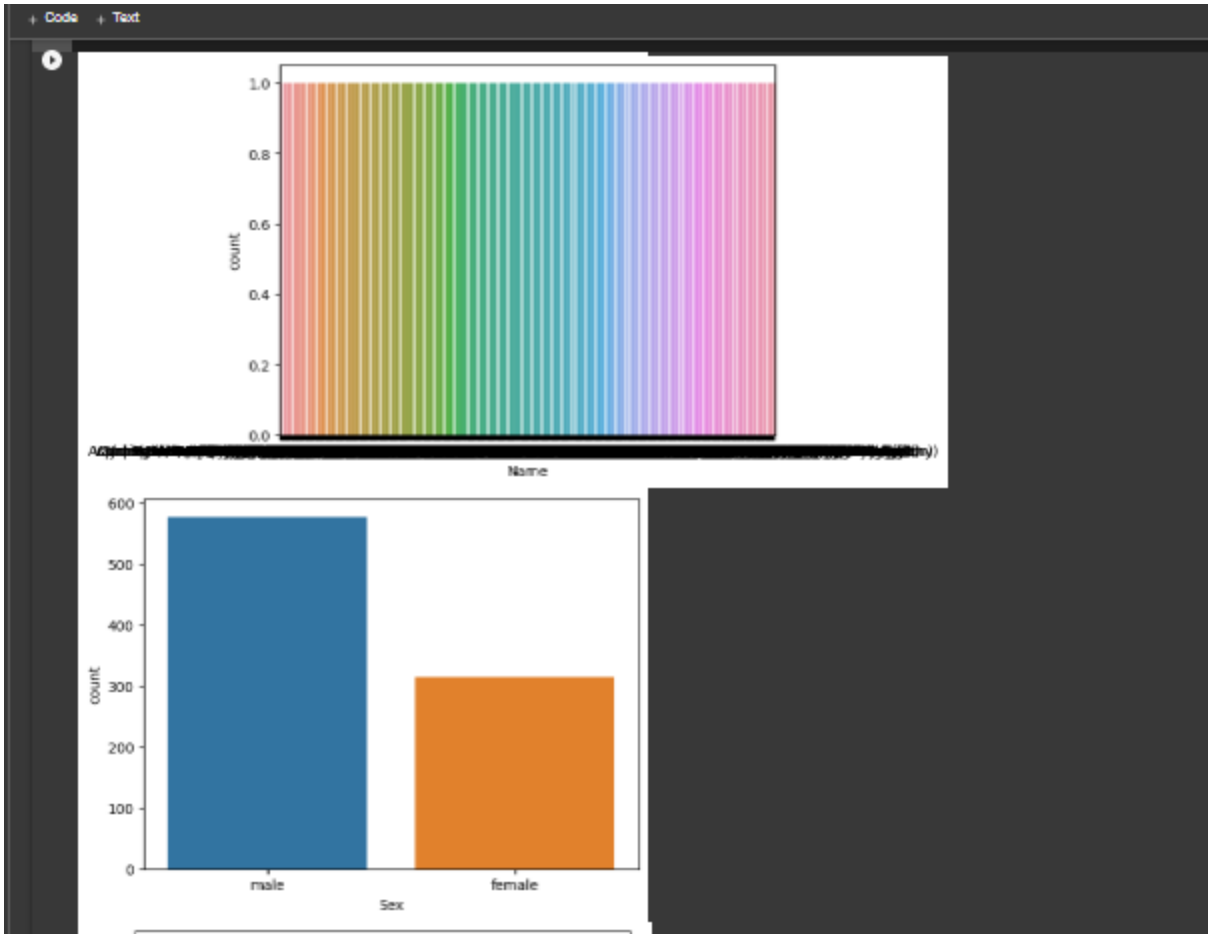
```
print(df.isnull().sum())
# printing the number of null values
```

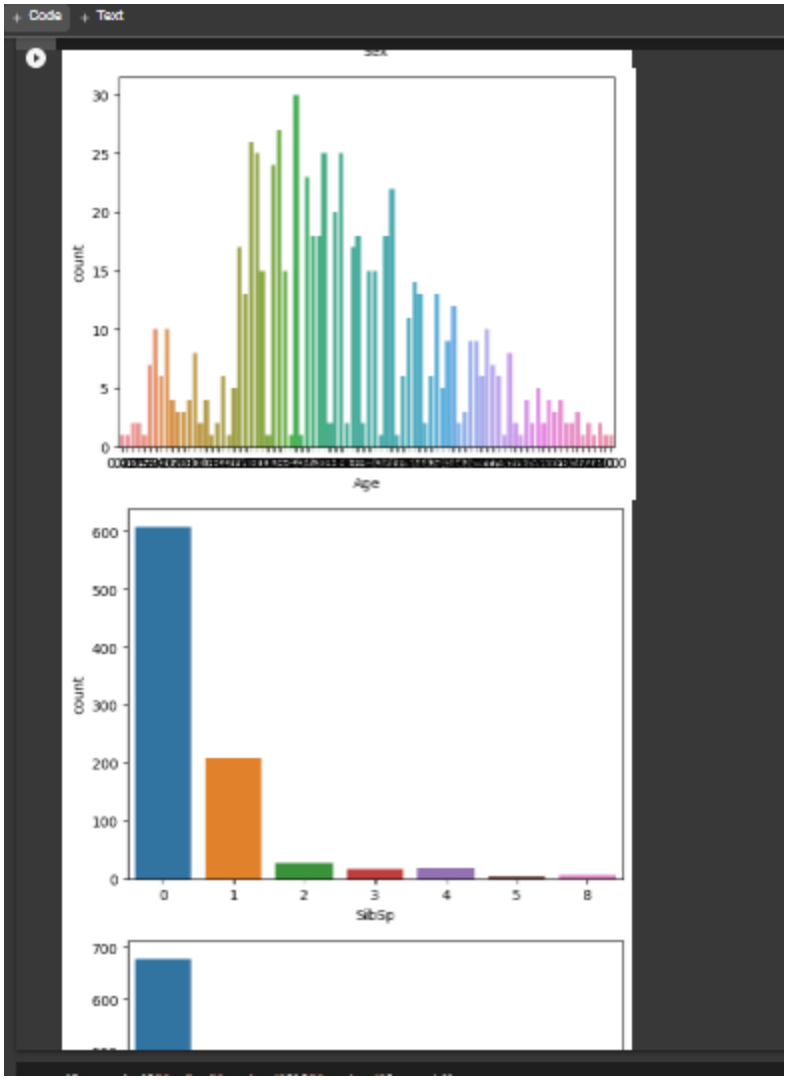
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

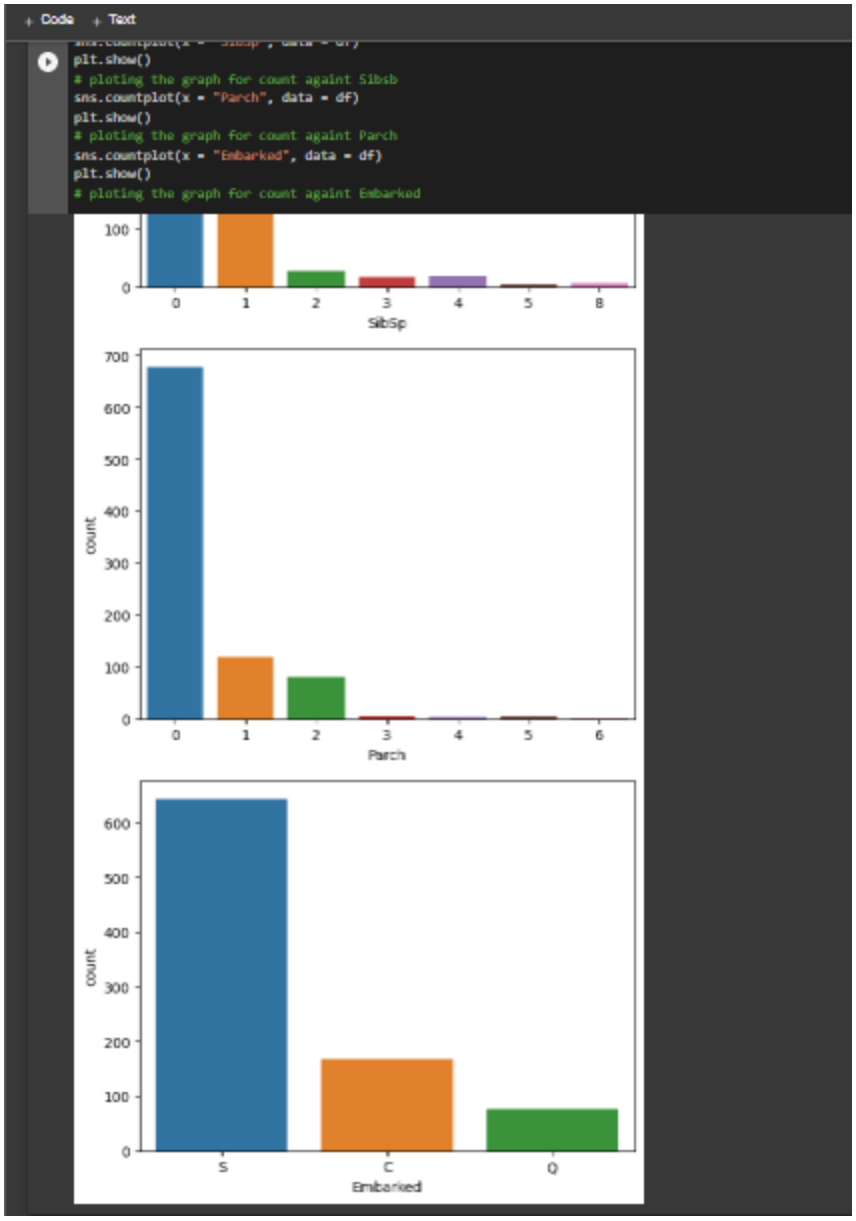
```python
import seaborn as sns
import matplotlib.pyplot as plt
# importing libraries

sns.countplot(x = "Survived", data = df)
plt.show()
# ploting the graph for count againt survived
sns.countplot(x = "Pclass", data = df)
plt.show()
# ploting the graph for count againt Pclass
sns.countplot(x = "Name", data = df)
plt.show()
# ploting the graph for count againt Name
sns.countplot(x = "Sex", data = df)
plt.show()
# ploting the graph for count againt Sex
sns.countplot(x = "Age", data = df)
plt.show()
# ploting the graph for count againt Age
sns.countplot(x = "SibSp", data = df)
plt.show()
# ploting the graph for count againt Sibsb
sns.countplot(x = "Parch", data = df)
plt.show()
# ploting the graph for count againt Parch
sns.countplot(x = "Embarked", data = df)
plt.show()
# ploting the graph for count againt Embarked
```

✓ 0s completed at 11:29 PM


+ Code
+ Text
500
400
300
200
100
0
count
0
1
Survived
500
400
300
200
100
0
count
1
2
3
Pclass
1.0
0.8


+ Code
+ Text
1.0
0.8
0.6
0.4
0.2
0.0
count
Name
600
500
400
300
200
100
0
count
male
female
Sex


+ Code
+ Text
Sex
count
Age
count
SibSp



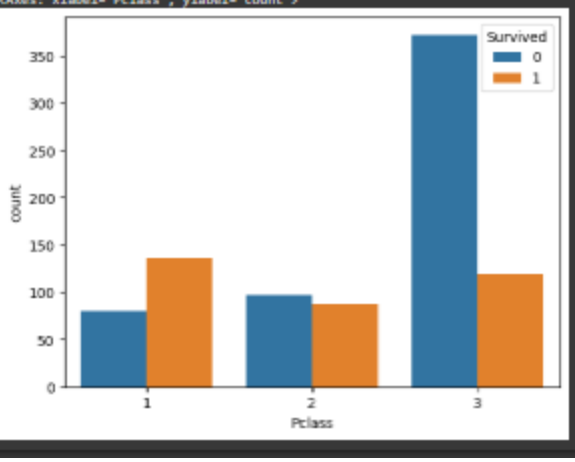
Groups data ny 2 categories, Sex and Survived and shows how many of each sex survived

```
[ ]  df.groupby(["Sex", "Survived"])["Survived"].count()
     #
```

```
Sex     Survived
female  0            81
        1           233
male    0           468
        1           109
Name: Survived, dtype: int64
```

```
sns.countplot(x = "Pclass", hue = "Survived", data = df)
```

```
<Axes: xlabel='Pclass', ylabel='count'>
```



We can see that Pclass 3 had the most people and unfortunately the most casualties.
Pclass 1 had the least deaths and most survivals, thus we can infer that Pclass people were in the most priority

```
[ ]  survived = df[df["Survived"] == 1]
     print("Oldest Survivor: ", survived["Age"].max())
     print("Youngest Survivor: ", survived["Age"].min())
     print("Average age of survivors: ", survived["Age"].mean())
```

```
Oldest Survivor:  80.0
Youngest Survivor:  0.42
Average age of survivors:  28.343689655172415
```

Showing the oldest, youngest, and average age of the survivors

```
[ ]  pd.crosstab(index = [df.Sex, df.Survived], columns = df.Pclass, margins = True)
```

	Pclass	1	2	3	All
Sex	Survived				
female	0	3	6	72	81
	1	91	70	72	233
male	0	77	91	300	468
	1	45	17	47	109
All		216	184	491	891

Crosstab shows the male and female of each Pclass who survived
Women and children are in first priority

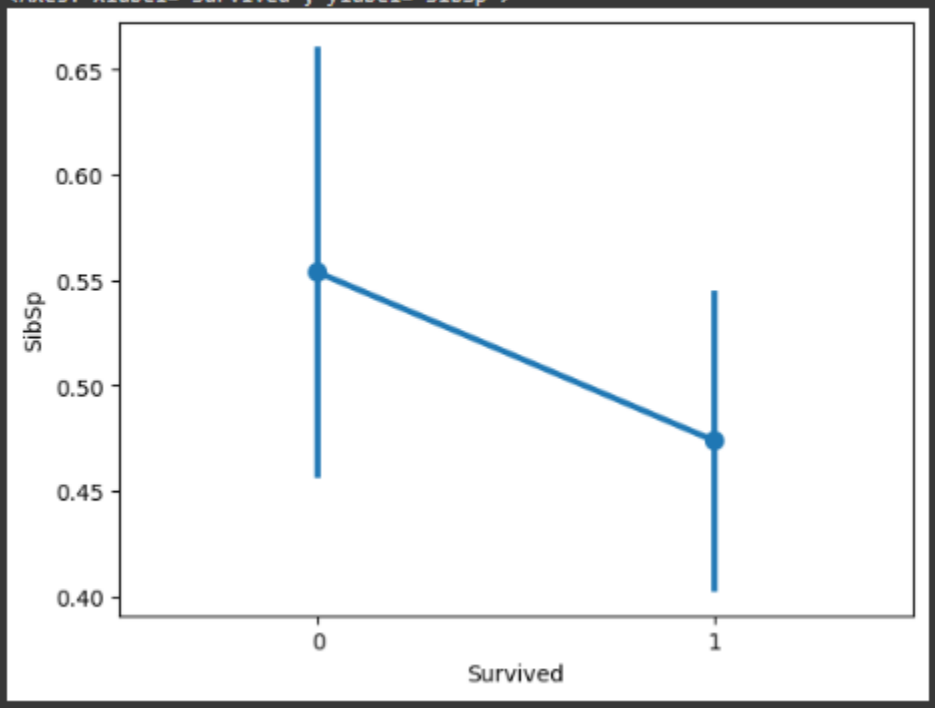
```
[ ]  pd.crosstab(index = df['SibSp'], columns = df['Survived'])
```

Survived	0	1
SibSp		
0	398	210
1	97	112
2	15	13
3	12	4
4	15	3
5	5	0
8	7	0

Factorplot no longer exists so we must use catplot

```
[8]  sns.pointplot(x = 'Survived', y = 'SibSp', data = df)
```

```
<Axes: xlabel='Survived', ylabel='SibSp'>
```



b.

2022570_prolog.pl

File Edit Browse Compile Prolog Pce Help

2022570_prolog.pl

```
% Defining dynamic predicate to store a playlist with its name and list of songs
:- dynamic playlist/2.

% Creating an empty playlist with the name 'My Playlist' and an empty list of songs
playlist('My Playlist', []).

% add_song predicate to add a new song to a playlist
add_song(Name, Title, Artist, UpdatedPlaylist) :-
    % Retrieving the old playlist using the playlist/2 predicate
    playlist(Name, OldPlaylist),
    % Appending the new song to the old playlist to create a new playlist
    append(OldPlaylist, [[Title, Artist]], NewPlaylist),
    % Removing the old playlist from the database
    retract(playlist(Name, OldPlaylist)),
    % Asserting the new playlist with the same name and updated list of songs
    asserta(playlist(Name, NewPlaylist)),
    % Assigning the updated playlist to the UpdatedPlaylist variable
    UpdatedPlaylist = playlist(Name, NewPlaylist),
    % Using cut to prevent backtracking and ensuring that only one solution is found
    !.
% If the add_song predicate fails, it returns false
add_song(_, _, _, _) :-
    fail.

% display_playlist predicate to display the list of songs in a playlist
display_playlist(Name) :-
    % Retrieving the playlist using the playlist/2 predicate
    playlist(Name, Songs),
    % Displaying the name of the playlist
    write('Playlist: '), write(Name), nl,
    % Displaying the list of songs
    write('Songs: '), nl,
    % Calling the display_songs predicate to display each song in the list
    display_songs(Songs).
```

```
% display_playlist predicate to display the list of songs in a playlist
display_playlist(Name) :-
    % Retrieving the playlist using the playlist/2 predicate
    playlist(Name, Songs),
    % Displaying the name of the playlist
    write('Playlist: '), write(Name), nl,
    % Displaying the list of songs
    write('Songs: '), nl,
    % Calling the display_songs predicate to display each song in the list
    display_songs(Songs).

% display_songs predicate to display each song in a list of songs
display_songs([]).
display_songs([[Title, Artist]|Rest]) :-
    % Displaying the title and artist of the song
    write('- '), write(Title), write(' by '), write(Artist), nl,
    % Recursively calling the display_songs predicate for the rest of the list
    display_songs(Rest).
```

SWI-Prolog (AMD64, Multi-threaded, version 9.0.4)

File Edit Settings Run Debug Help

```
Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?-
% c:/Users/Vikranth/Desktop/2022570_HW1/2022570_prolog.pl compiled 0.00 sec, 6 clauses
?- playlist('My Playlist', []).
true.

?- add_song('My Playlist', 'Song 1', 'Artist 1', UpdatedPlaylist).
UpdatedPlaylist = playlist('My Playlist', [['Song 1', 'Artist 1']]).

?- add_song('My Playlist', 'Song 2', 'Artist 2', UpdatedPlaylist).
UpdatedPlaylist = playlist('My Playlist', [['Song 1', 'Artist 1'], ['Song 2', 'Artist 2']]).

?- add_song('My Playlist', 'Song 3', 'Artist 3', UpdatedPlaylist).
UpdatedPlaylist = playlist('My Playlist', [['Song 1', 'Artist 1'], ['Song 2', 'Artist 2'], ['Song 3', 'Artist 3']]).

?- display_playlist('My Playlist').
Playlist: My Playlist
Songs:
- Song 1 by Artist 1
- Song 2 by Artist 2
- Song 3 by Artist 3
true.
```