Muhammad Umer Adeeb

Batch 7 - DSAI

Question 1: Data Loading and Inspection (Titanic Dataset)

a) Load the Titanic dataset using Seaborn's load_dataset function.

b) Display the first 10 rows of the dataset.

c) Provide a summary of the dataset's information, including data types and missing values.

```
import seaborn as sns
titanic_data = sns.load_dataset('titanic')
display(titanic_data.head(10))

titanic_data.info()
titanic_data
```

```
   survived  pclass     sex   age  sibsp  parch     fare embarked
class  \
0         0       3    male  22.0      1      0   7.2500        S
Third
1         1       1  female  38.0      1      0  71.2833        C
First
2         1       3  female  26.0      0      0   7.9250        S
Third
3         1       1  female  35.0      1      0  53.1000        S
First
4         0       3    male  35.0      0      0   8.0500        S
Third
5         0       3    male   NaN      0      0   8.4583        Q
Third
6         0       1    male  54.0      0      0  51.8625        S
First
7         0       3    male   2.0      3      1  21.0750        S
Third
8         1       3  female  27.0      0      2  11.1333        S
Third
9         1       2  female  14.0      1      0  30.0708        C
Second

     who  adult_male deck  embark_town alive  alone
0    man        True  NaN  Southampton    no  False
1  woman       False    C    Cherbourg   yes  False
```

```
2  woman       False  NaN  Southampton   yes   True
3  woman       False    C  Southampton   yes  False
4    man        True  NaN  Southampton    no   True
5    man        True  NaN   Queenstown    no   True
6    man        True    E  Southampton    no   True
7  child       False  NaN  Southampton    no  False
8  woman       False  NaN  Southampton   yes  False
9  child       False  NaN    Cherbourg   yes  False

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 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          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB

     survived  pclass     sex   age  sibsp  parch     fare embarked
class  \
0           0       3    male  22.0      1      0   7.2500        S
Third
1           1       1  female  38.0      1      0  71.2833        C
First
2           1       3  female  26.0      0      0   7.9250        S
Third
3           1       1  female  35.0      1      0  53.1000        S
First
4           0       3    male  35.0      0      0   8.0500        S
Third
..        ...     ...     ...   ...    ...    ...      ...      ...
...
886         0       2    male  27.0      0      0  13.0000        S
Second
887         1       1  female  19.0      0      0  30.0000        S
First
```

```
888         0       3  female   NaN      1      2  23.4500        S
Third
889         1       1    male  26.0      0      0  30.0000        C
First
890         0       3    male  32.0      0      0   7.7500        Q
Third

       who  adult_male deck  embark_town alive  alone
0      man        True  NaN  Southampton    no  False
1    woman       False    C    Cherbourg   yes  False
2    woman       False  NaN  Southampton   yes   True
3    woman       False    C  Southampton   yes  False
4      man        True  NaN  Southampton    no   True
..     ...         ...  ...          ...   ...    ...
886    man        True  NaN  Southampton    no   True
887  woman       False    B  Southampton   yes   True
888  woman       False  NaN  Southampton    no  False
889    man        True    C    Cherbourg   yes   True
890    man        True  NaN   Queenstown    no   True

[891 rows x 15 columns]
```

Question 2: Statistical Summary

a) Using the Titanic dataset, display the statistical summary (mean, median, quartiles, etc.) of all numerical columns.

b) Interpret the mean age of the passengers.

```
display(titanic_data.describe())
mean_age = titanic_data['age'].mean()
display(f"The mean age of the passengers is {mean_age:.2f}")
```

```
         survived      pclass         age       sibsp       parch
fare
count  891.000000  891.000000  714.000000  891.000000  891.000000
891.000000
mean     0.383838    2.308642   29.699118    0.523008    0.381594
32.204208
std      0.486592    0.836071   14.526497    1.102743    0.806057
49.693429
min      0.000000    1.000000    0.420000    0.000000    0.000000
0.000000
25%      0.000000    2.000000   20.125000    0.000000    0.000000
7.910400
50%      0.000000    3.000000   28.000000    0.000000    0.000000
14.454200
75%      1.000000    3.000000   38.000000    1.000000    0.000000
31.000000
```

```
max      1.000000    3.000000   80.000000    8.000000    6.000000
512.329200

'The mean age of the passengers is 29.70'
```

Question 3: Handling Missing Data

a) Identify the columns in the Titanic dataset that have missing values.

b) Choose an appropriate method to handle missing values in the age column and apply it.

c) Justify your choice of method.

```
missing_values = titanic_data.isnull().any()
display(missing_values)

titanic_data['age'].fillna(mean_age, inplace=True)      # inplace ----
> When `inplace=True` is specified, the original DataFrame is modified
directly
titanic_data.head(10)

print("The mean (average) age shows the typical age. Using the mean
keeps the overall numbers similar to the original data, \nwhich is
better than filling in missing values with random numbers or zeros.")
```

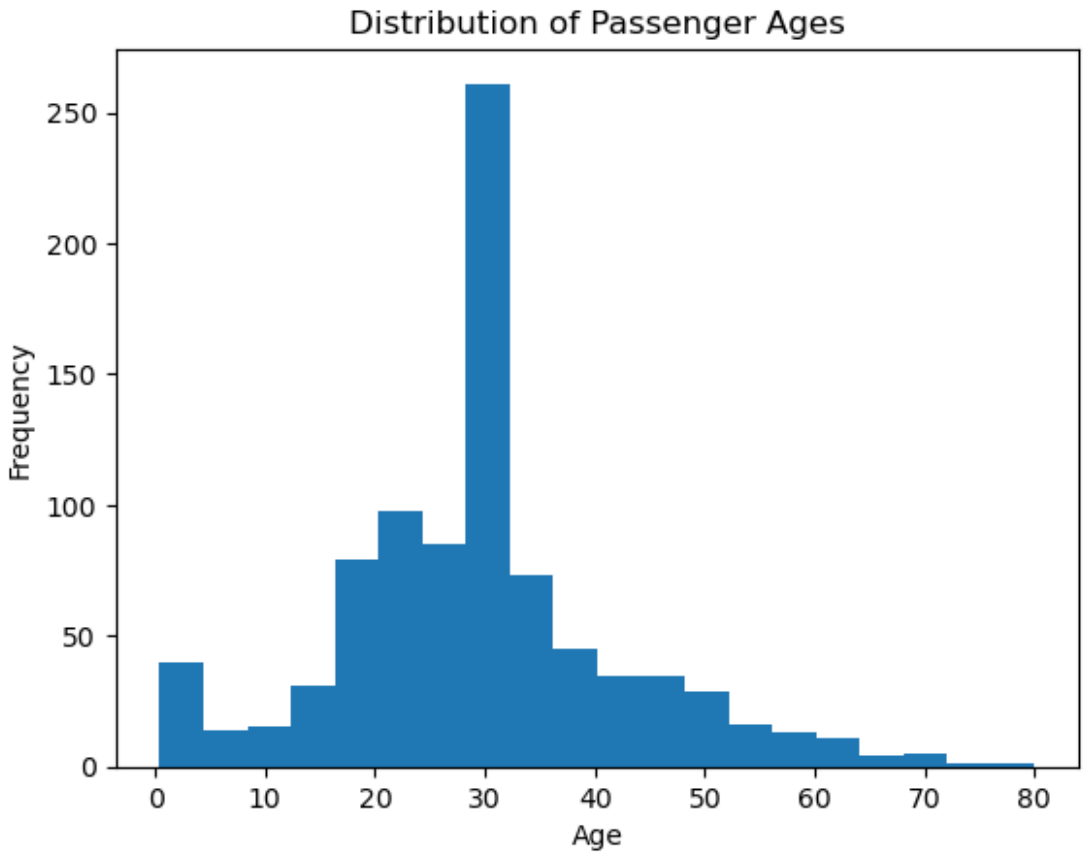
```
survived       False
pclass         False
sex            False
age             True
sibsp          False
parch          False
fare           False
embarked        True
class          False
who            False
adult_male     False
deck            True
embark_town     True
alive          False
alone          False
dtype: bool

The mean (average) age shows the typical age. Using the mean keeps the
overall numbers similar to the original data,
which is better than filling in missing values with random numbers or
zeros.
```

Question 4: Data Visualization - Univariate Analysis

a) Plot a histogram of the age distribution of the Titanic passengers.

b) What does the distribution tell you about the age of the passengers?

```
import matplotlib.pyplot as plt

plt.hist(titanic_data['age'], bins=20)
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.title('Distribution of Passenger Ages')
plt.show()

print("The distribution tells us about the typical age of Titanic
passengers. \nIt appears that the majority of the passengers are in
the range of 20-40 years old. \nWe can observe that there is also a
significant number of children as well as older adults on the ship.")
```




```
The distribution tells us about the typical age of Titanic passengers.

It appears that the majority of the passengers are in the range of 20-
40 years old.
We can observe that there is also a significant number of children as
well as older adults on the ship.
```

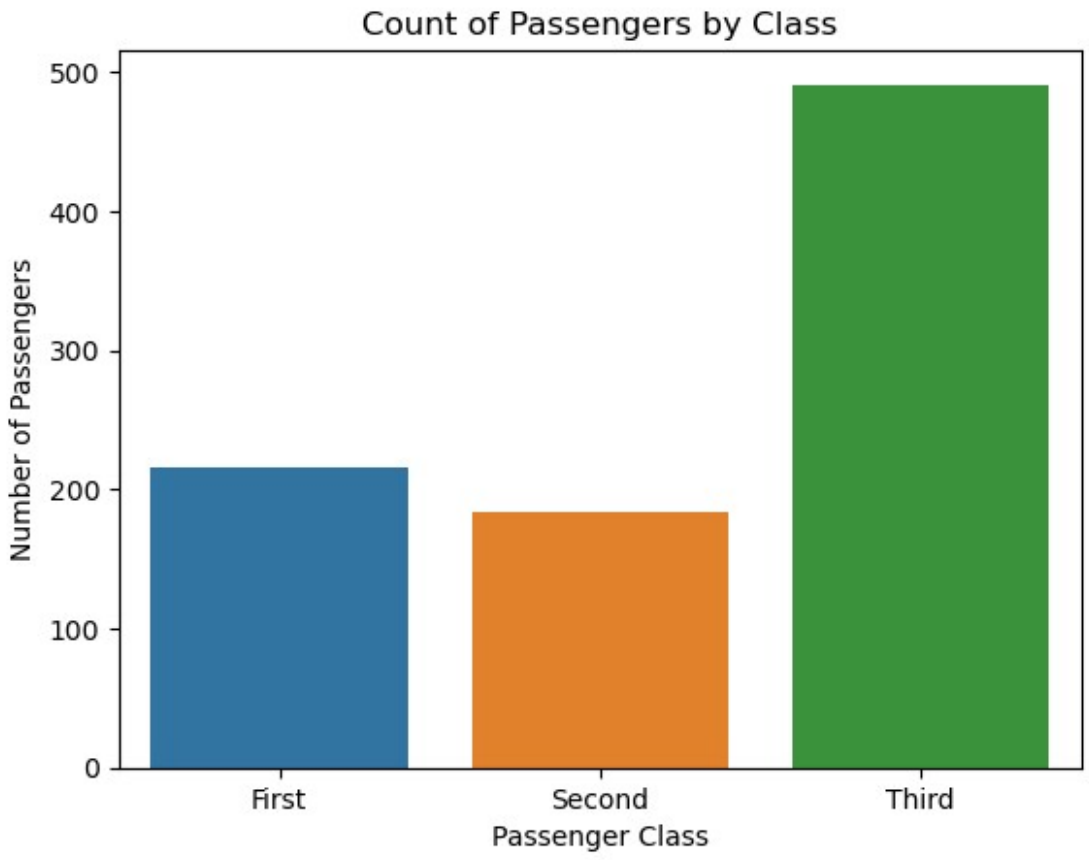
Question 5: Categorical Data Analysis

a) Using the Titanic dataset, create a count plot of the class variable.

b) Which passenger class was the most common?

```
sns.countplot(x='class', data=titanic_data)
plt.title('Count of Passengers by Class')
plt.xlabel('Passenger Class')
plt.ylabel('Number of Passengers')
plt.show()

class_counts = titanic_data['class'].value_counts()
display(class_counts)
most_common_class = class_counts.idxmax()
print(f"The most common passenger class was: {most_common_class}")
```

```
C:\Users\MAK TECH\Anaconda3\Lib\site-packages\seaborn\
categorical.py:641: FutureWarning: The default of observed=False is
deprecated and will be changed to True in a future version of pandas.
Pass observed=False to retain current behavior or observed=True to
adopt the future default and silence this warning.
  grouped_vals = vals.groupby(grouper)
```



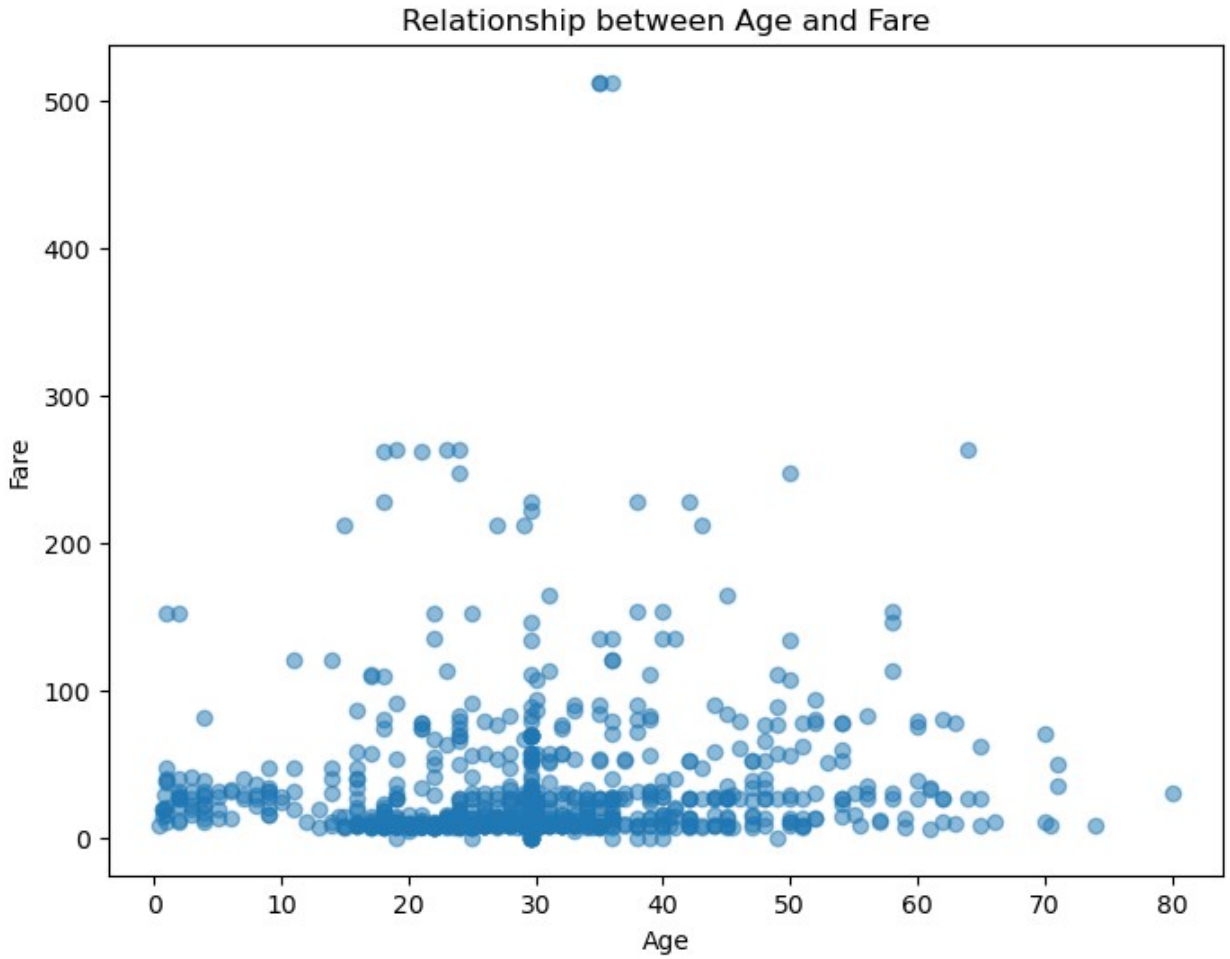
```
class
Third     491
First     216
Second    184
Name: count, dtype: int64

The most common passenger class was: Third
```

Question 6: Bivariate Analysis

a) Create a scatter plot showing the relationship between age and fare.

b) Does there appear to be any correlation between age and fare?

```
plt.figure(figsize=(8, 6))  # Adjust figure size if needed
plt.scatter(titanic_data['age'], titanic_data['fare'], alpha=0.5)
plt.xlabel('Age')
plt.ylabel('Fare')
plt.title('Relationship between Age and Fare')
plt.show()

correlation = titanic_data['age'].corr(titanic_data['fare'])
print('The correlation coefficient between age and fare is',
correlation)
print("Based on the scatter plot and correlation coefficient, there
doesn't appear to be a strong linear correlation between age and fare.
\nIt seems that passengers of various ages can pay different fares,
with some outliers paying very high fares at various ages. \nIf the
correlation value is close to 1, it indicates a strong positive
correlation (as age increases, fare tends to increase) \nIf it's close
to -1, it indicates a strong negative correlation (as age increases,
fare tends to decrease) \nBut since value is around 0 means there's
little to no linear correlation.")
```



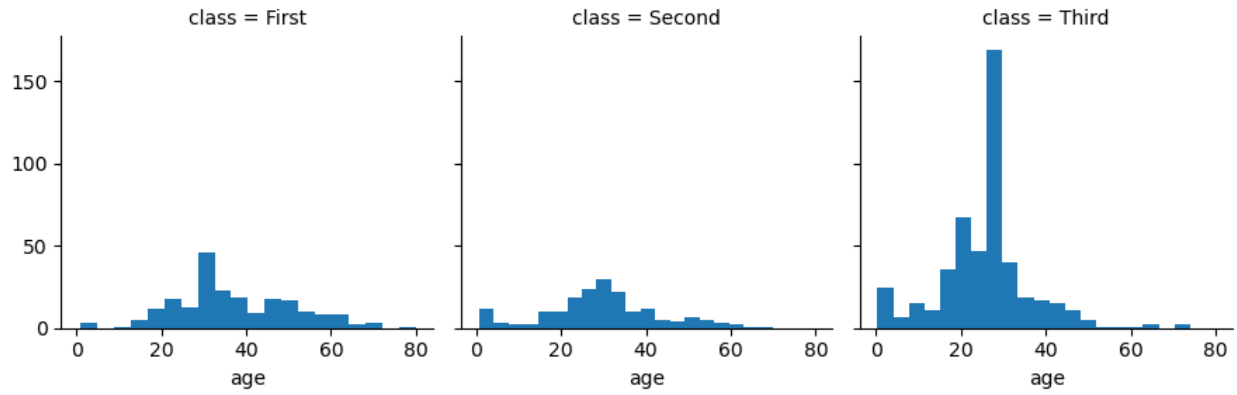

```
The correlation coefficient between age and fare is
0.09156609328505758
Based on the scatter plot and correlation coefficient, there doesn't
appear to be a strong linear correlation between age and fare.
It seems that passengers of various ages can pay different fares, with
some outliers paying very high fares at various ages.
If the correlation value is close to 1, it indicates a strong positive
correlation (as age increases, fare tends to increase)
If it's close to -1, it indicates a strong negative correlation (as
age increases, fare tends to decrease)
But since value is around 0 means there's little to no linear
correlation.
```

Question 7: Faceted Plots

a) Use Seaborn's FacetGrid to create separate histograms of age for each passenger class.

b) Describe any noticeable differences between the classes.

```
age = sns.FacetGrid(titanic_data, col='class')
age.map(plt.hist, 'age', bins=20)
plt.show()

print("Based on the faceted histograms, here's a description of
noticeable differences between passenger classes with respect to
age:")
print(
    "1. Third Class: The third class has a more spread-out age
distribution, it seems like the distribution is a little skewed to the
right"
)
print(
    "2. Second Class: The distribution is more centered and resembles
a normal distribution in its shape."
)
print(
    "3. First Class: The first-class distribution has some
similarities with second class, but it might have a slightly higher
proportion of older adults."
)
```




```
Based on the faceted histograms, here's a description of noticeable
differences between passenger classes with respect to age:
1. Third Class: The third class has a more spread-out age
distribution, it seems like the distribution is a little skewed to the
right
2. Second Class: The distribution is more centered and resembles a
normal distribution in its shape.
3. First Class: The first-class distribution has some similarities
with second class, but it might have a slightly higher proportion of
older adults.
```

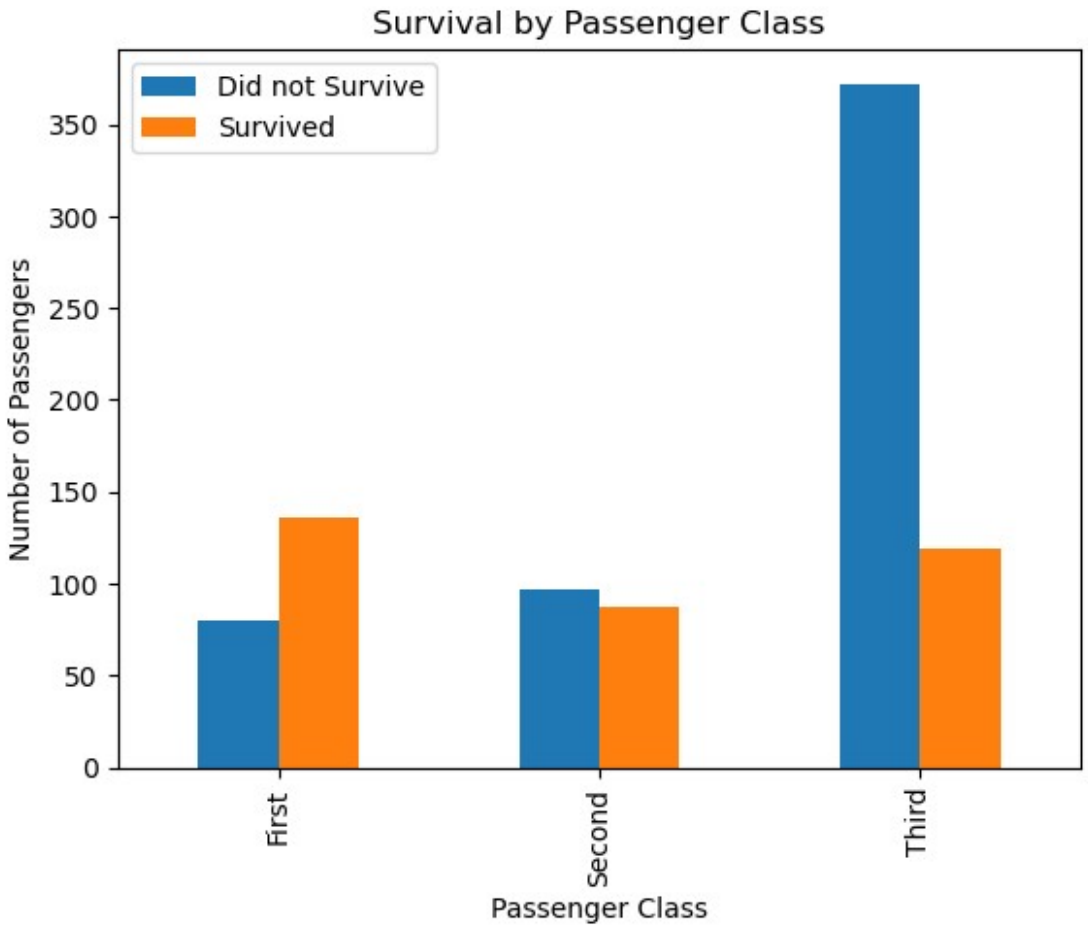
Question 8: Survival Analysis

a) Create a bar plot showing the number of survivors (survived) by passenger class.

b) What conclusions can you draw about survival rates across classes?

```
survivors_by_class = titanic_data.groupby(['class', 'survived'])
['survived'].count().unstack()
survivors_by_class.plot(kind='bar', stacked=False)  # Use stacked=True
for a stacked bar plot
plt.xlabel('Passenger Class')
plt.ylabel('Number of Passengers')
plt.title('Survival by Passenger Class')
plt.legend(['Did not Survive', 'Survived'])  # Add a legend
plt.show()

print(
    "Based on the bar plot, we can draw the following conclusions
about survival rates across classes:"
)
print(
    "1. Passengers in First Class had a higher survival rate compared
to passengers in other classes."
)
print(
    "2. Passengers in Second Class had a slightly lower survival rate
than First Class but higher than Third Class."
)
print(
    "3. Passengers in Third Class had the lowest survival rate among
the three passenger classes."
)
print(
    "This suggests that passenger class played a significant role in
survival on the Titanic."
)
```

```
C:\Users\MAK TECH\AppData\Local\Temp\ipykernel_11008\1574927471.py:1:
FutureWarning: The default of observed=False is deprecated and will be
changed to True in a future version of pandas. Pass observed=False to
retain current behavior or observed=True to adopt the future default
and silence this warning.
  survivors_by_class = titanic_data.groupby(['class', 'survived'])
['survived'].count().unstack()
```




```
Based on the bar plot, we can draw the following conclusions about
survival rates across classes:
1. Passengers in First Class had a higher survival rate compared to
passengers in other classes.
2. Passengers in Second Class had a slightly lower survival rate than
First Class but higher than Third Class.
3. Passengers in Third Class had the lowest survival rate among the
three passenger classes.
This suggests that passenger class played a significant role in
survival on the Titanic.
```

Question 9: Correlation Heatmap

a) Compute the correlation matrix for the numerical variables in the Titanic dataset.

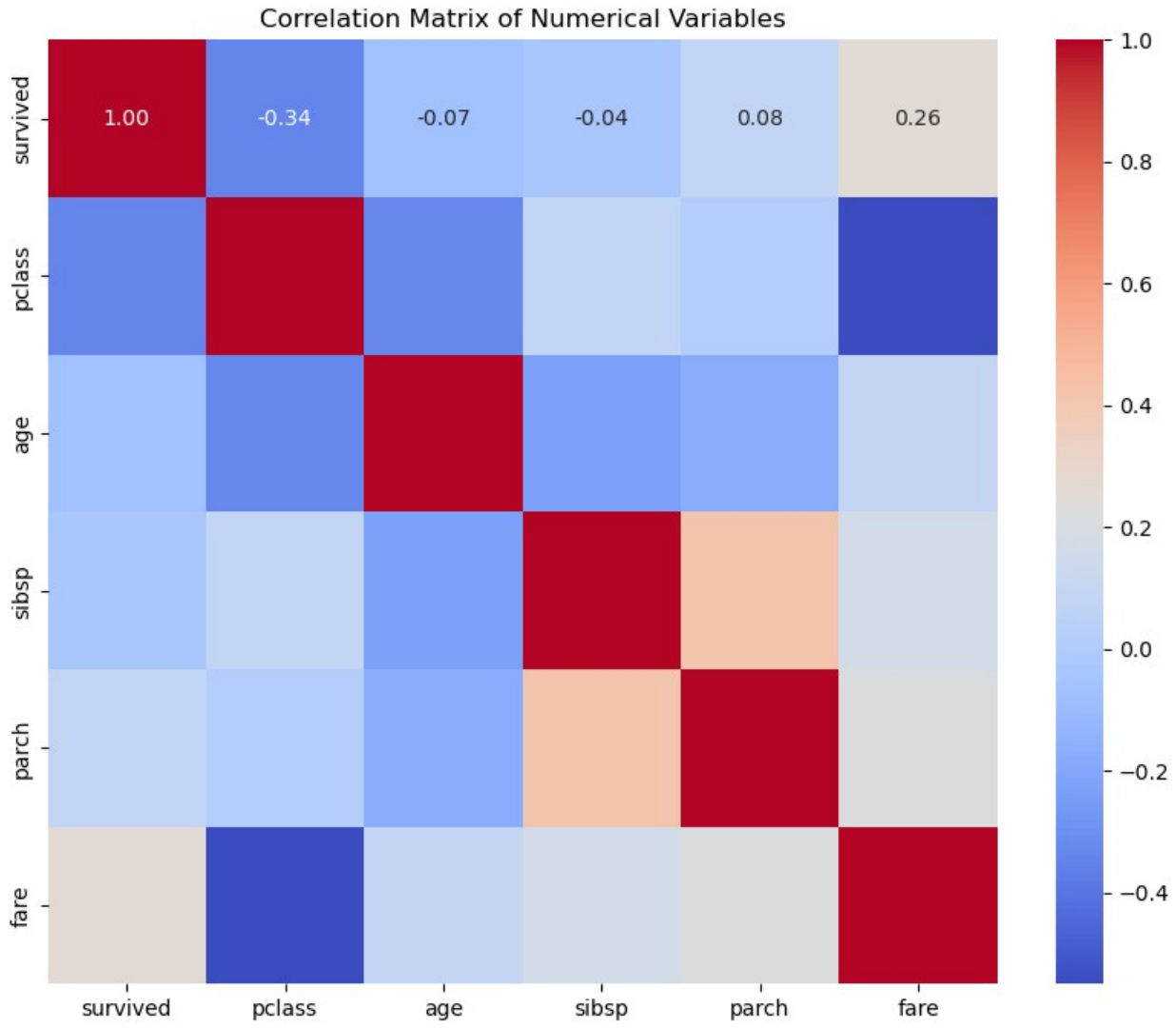
b) Plot a heatmap of the correlation matrix using Seaborn.

c) Identify any strong correlations and discuss them.

```
correlation_matrix =
titanic_data.select_dtypes(include=['number']).corr()
```

```
display(correlation_matrix)

plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
fmt=".2f")
plt.title('Correlation Matrix of Numerical Variables')
plt.show()


print("Based on the heatmap:")
print("- **Fare and Pclass have a moderate negative correlation:**
This means that as the passenger class increases (i.e., 1st class is
higher than 3rd class), the fare tends to be higher. This is expected
as higher-class tickets are generally more expensive.")
print("- **There is a weak positive correlation between age and
fare:** This means that as age increases, the fare tends to increase
slightly. It's not a very strong relationship though.")
print("- **SibSp and Parch have a weak positive correlation:** This
indicates that passengers who have siblings/spouses are more likely to
also have parents/children on board, and vice versa. It makes sense as
families are likely to travel together.")
```

	survived	pclass	age	sibsp	parch	fare
survived	1.000000	-0.338481	-0.069809	-0.035322	0.081629	0.257307
pclass	-0.338481	1.000000	-0.331339	0.083081	0.018443	-0.549500
age	-0.069809	-0.331339	1.000000	-0.232625	-0.179191	0.091566
sibsp	-0.035322	0.083081	-0.232625	1.000000	0.414838	0.159651
parch	0.081629	0.018443	-0.179191	0.414838	1.000000	0.216225
fare	0.257307	-0.549500	0.091566	0.159651	0.216225	1.000000



Based on the heatmap:
- **Fare and Pclass have a moderate negative correlation:** This means that as the passenger class increases (i.e., 1st class is higher than 3rd class), the fare tends to be higher. This is expected as higher-class tickets are generally more expensive.
- **There is a weak positive correlation between age and fare:** This means that as age increases, the fare tends to increase slightly. It's not a very strong relationship though.
- **SibSp and Parch have a weak positive correlation:** This indicates that passengers who have siblings/spouses are more likely to also have parents/children on board, and vice versa. It makes sense as families are likely to travel together.

Question 10: Time Series Data (If Applicable)

Note: Since the Titanic dataset does not include time series data, use the following synthetic data for this question.

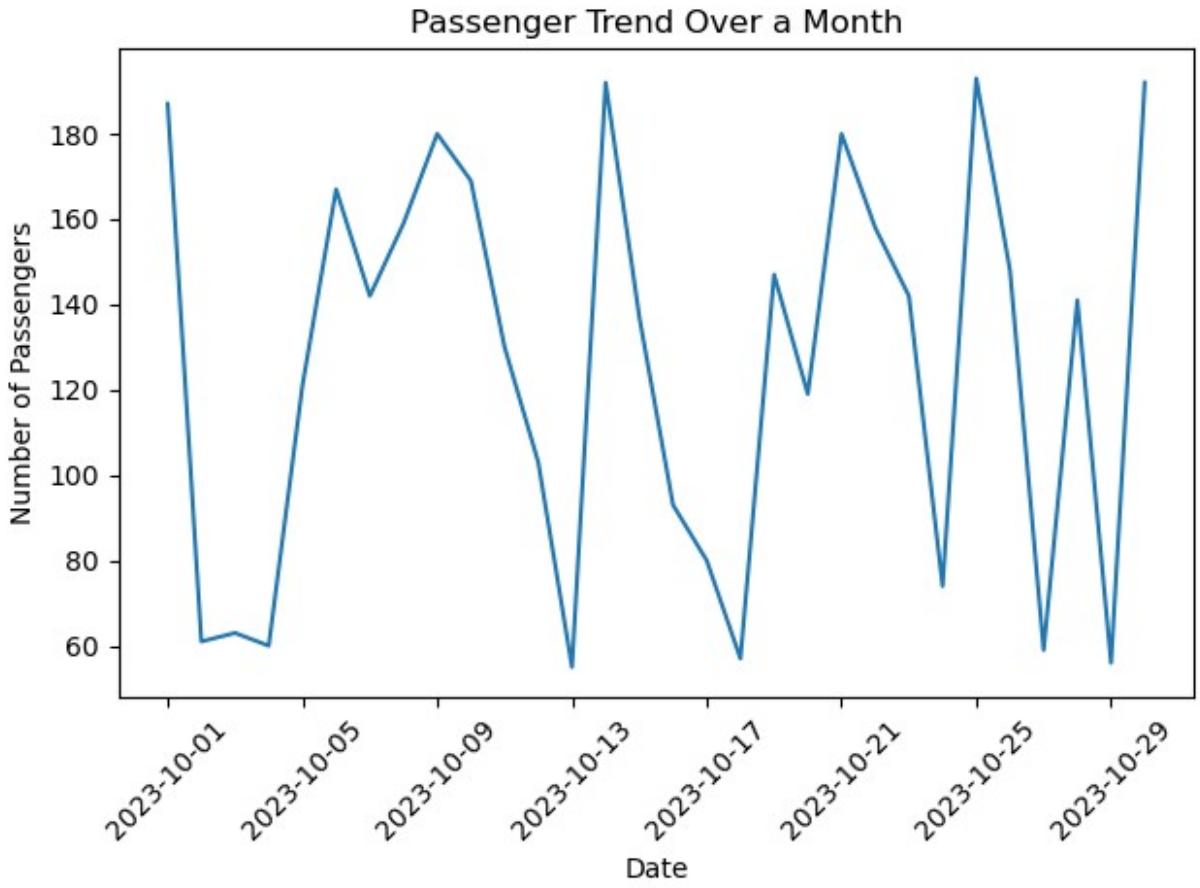
a) Create a Pandas DataFrame containing the number of passengers boarded each day for a month (generate random data).

b) Plot a line chart showing the trend over the month.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

dates = pd.date_range(start='2023-10-01', periods=30, freq='D')
passenger_counts = np.random.randint(50, 200, size=30)  # Random
counts between 50 and 200
df_passengers = pd.DataFrame({'Date': dates, 'Passengers':
passenger_counts})
display(df_passengers)

plt.plot(df_passengers['Date'], df_passengers['Passengers'])
plt.xlabel('Date')
plt.ylabel('Number of Passengers')
plt.title('Passenger Trend Over a Month')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

	Date	Passengers
0	2023-10-01	187
1	2023-10-02	61
2	2023-10-03	63
3	2023-10-04	60
4	2023-10-05	121
5	2023-10-06	167
6	2023-10-07	142
7	2023-10-08	159
8	2023-10-09	180
9	2023-10-10	169
10	2023-10-11	130
11	2023-10-12	103
12	2023-10-13	55
13	2023-10-14	192
14	2023-10-15	137
15	2023-10-16	93
16	2023-10-17	80
17	2023-10-18	57
18	2023-10-19	147
19	2023-10-20	119

```
20 2023-10-21          180
21 2023-10-22          158
22 2023-10-23          142
23 2023-10-24           74
24 2023-10-25          193
25 2023-10-26          148
26 2023-10-27           59
27 2023-10-28          141
28 2023-10-29           56
29 2023-10-30          192
```



Question 11: Data Loading and Inspection (Diabetes Dataset)

a) Load the Diabetes dataset from the provided Kaggle link.

b) Display the last 5 rows of the dataset.

c) Check for missing values and report your findings.

```
pip install pandas kaggle
```

```
Requirement already satisfied: pandas in c:\users\mak tech\anaconda3\
lib\site-packages (2.1.4)
Requirement already satisfied: kaggle in c:\users\mak tech\anaconda3\
lib\site-packages (1.6.17)
Requirement already satisfied: numpy<2,>=1.23.2 in c:\users\mak tech\
anaconda3\lib\site-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\mak
tech\anaconda3\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\mak tech\
anaconda3\lib\site-packages (from pandas) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\mak tech\
anaconda3\lib\site-packages (from pandas) (2023.3)
Requirement already satisfied: six>=1.10 in c:\users\mak tech\
anaconda3\lib\site-packages (from kaggle) (1.16.0)
Requirement already satisfied: certifi>=2023.7.22 in c:\users\mak
tech\anaconda3\lib\site-packages (from kaggle) (2024.7.4)
Requirement already satisfied: requests in c:\users\mak tech\
anaconda3\lib\site-packages (from kaggle) (2.31.0)
Requirement already satisfied: tqdm in c:\users\mak tech\anaconda3\
lib\site-packages (from kaggle) (4.65.0)
Requirement already satisfied: python-slugify in c:\users\mak tech\
anaconda3\lib\site-packages (from kaggle) (5.0.2)
Requirement already satisfied: urllib3 in c:\users\mak tech\anaconda3\
lib\site-packages (from kaggle) (2.0.7)
Requirement already satisfied: bleach in c:\users\mak tech\anaconda3\
lib\site-packages (from kaggle) (4.1.0)
Requirement already satisfied: packaging in c:\users\mak tech\
anaconda3\lib\site-packages (from bleach->kaggle) (23.1)
Requirement already satisfied: webencodings in c:\users\mak tech\
anaconda3\lib\site-packages (from bleach->kaggle) (0.5.1)
Requirement already satisfied: text-unidecode>=1.3 in c:\users\mak
tech\anaconda3\lib\site-packages (from python-slugify->kaggle) (1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\
mak tech\anaconda3\lib\site-packages (from requests->kaggle) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\mak tech\
anaconda3\lib\site-packages (from requests->kaggle) (3.4)
Requirement already satisfied: colorama in c:\users\mak tech\
anaconda3\lib\site-packages (from tqdm->kaggle) (0.4.6)
Note: you may need to restart the kernel to use updated packages.
```

```python
import os
import zipfile

# Create a directory to store the dataset
if not os.path.exists('diabetes-dataset'):
    os.makedirs('diabetes-dataset')

# Change the directory to the created folder
os.chdir('diabetes-dataset')
```

```
# Download the dataset
!kaggle datasets download -d mathchi/diabetes-data-set

# Unzip the downloaded dataset
with zipfile.ZipFile('diabetes-data-set.zip', 'r') as zip_ref:
    zip_ref.extractall()
```

```
Dataset URL: https://www.kaggle.com/datasets/mathchi/diabetes-data-set
License(s): CC0-1.0
Downloading diabetes-data-set.zip to C:\Users\MAK TECH\Desktop\PGD
python\final_datavisualization\diabetes-dataset



  0%|          | 0.00/8.91k [00:00<?, ?B/s]
100%|##########| 8.91k/8.91k [00:00<00:00, 6.05MB/s]
```

```
import pandas as pd

df = pd.read_csv('diabetes.csv')  # Adjust the filename if necessary
display(df.tail())
```

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI
\
763           10      101             76             48      180  32.9

764            2      122             70             27        0  36.8

765            5      121             72             23      112  26.2

766            1      126             60              0        0  30.1

767            1       93             70             31        0  30.4


     DiabetesPedigreeFunction  Age  Outcome
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0
```

```
missing_values = df.isnull().sum()
display(missing_values)
```

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
```

```
Age                         0
Outcome                     0
dtype: int64
```

Question 12: Pair Plot

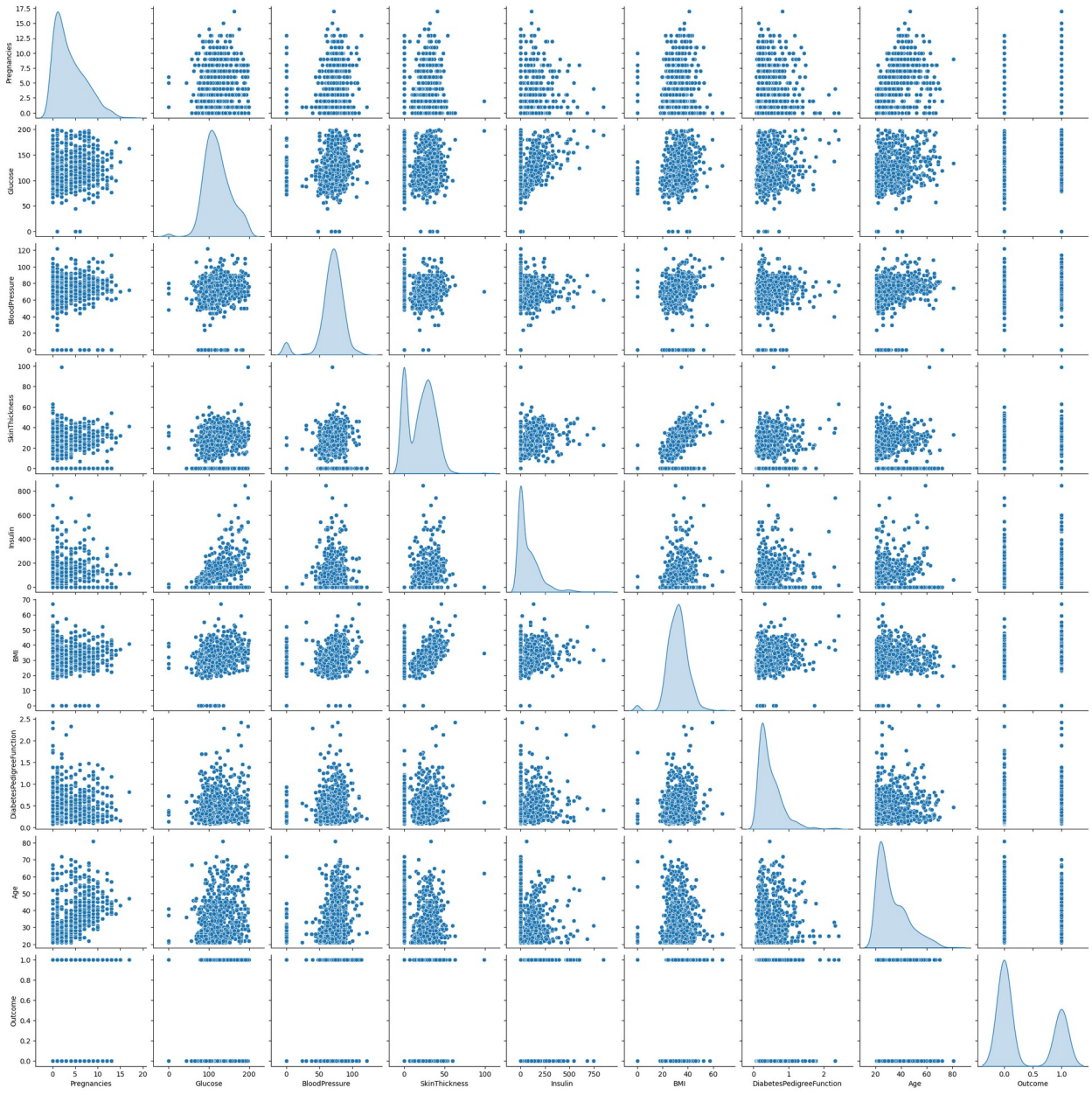
a) Using the Diabetes dataset, create a pair plot of all numerical variables.

b) Highlight any interesting relationships you observe.

```
import seaborn as sns
import matplotlib.pyplot as plt

sns.pairplot(df, diag_kind='kde')  # Use diag_kind='kde' for kernel density estimation plots on the diagonal
plt.show()

print("Based on the pair plot:")
print("- Some variables, like 'Glucose', 'Insulin', and 'BMI', appear to have some positive correlations with 'Outcome' (whether a person has diabetes).")
print("- You can also see some relationships between variables themselves. For example, 'BMI' and 'SkinThickness' might show a positive correlation.")
print("- The diagonal plots (kernel density estimates) give you an idea of the distribution of each variable.")
print("- It's useful to explore these relationships to understand potential patterns and features that might be important in predicting diabetes.")
```

```
C:\Users\MAK TECH\Anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\MAK TECH\Anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\MAK TECH\Anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\MAK TECH\Anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
```

```
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\MAK TECH\Anaconda3\Lib\site-packages\seaborn\
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\MAK TECH\Anaconda3\Lib\site-packages\seaborn\
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\MAK TECH\Anaconda3\Lib\site-packages\seaborn\
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\MAK TECH\Anaconda3\Lib\site-packages\seaborn\
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\MAK TECH\Anaconda3\Lib\site-packages\seaborn\
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



Based on the pair plot:
- Some variables, like 'Glucose', 'Insulin', and 'BMI', appear to have some positive correlations with 'Outcome' (whether a person has diabetes).
- You can also see some relationships between variables themselves. For example, 'BMI' and 'SkinThickness' might show a positive correlation.
- The diagonal plots (kernel density estimates) give you an idea of the distribution of each variable.
- It's useful to explore these relationships to understand potential patterns and features that might be important in predicting diabetes.

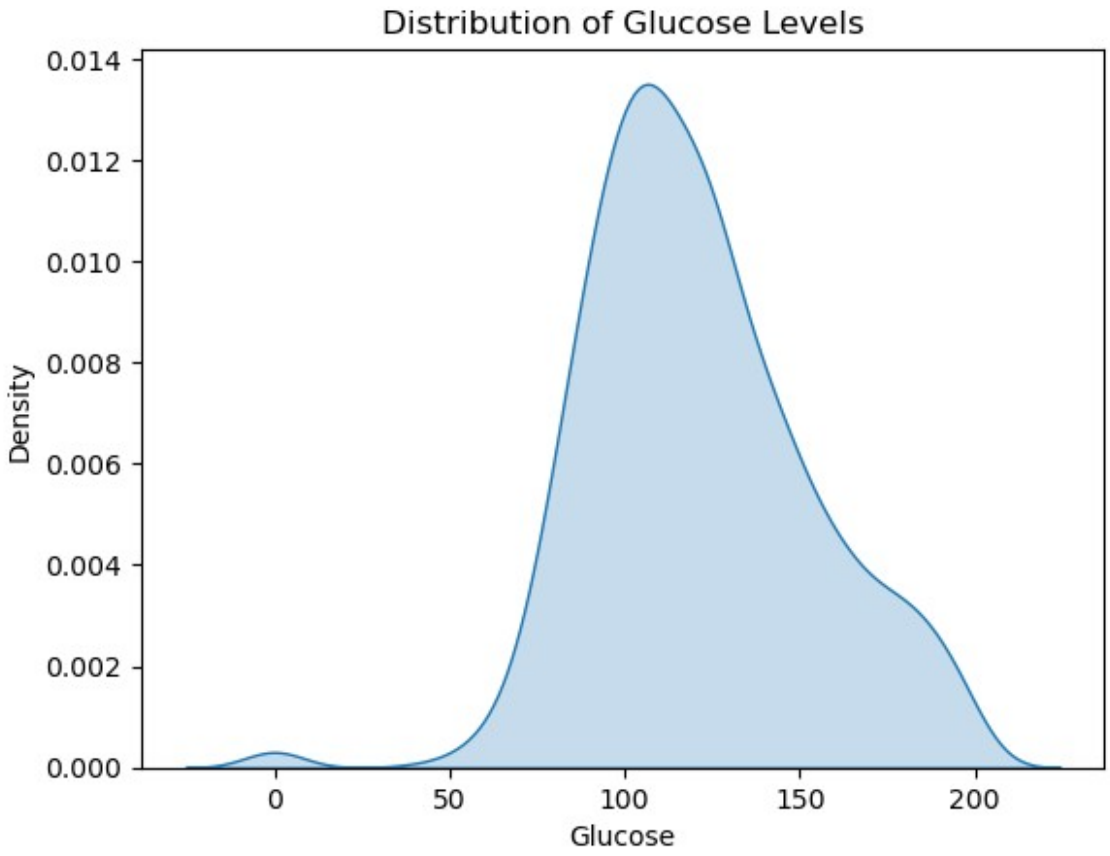
Question 13: Distribution Plots

a) Plot the distribution of the Glucose variable using a KDE plot.

b) Does the Glucose level appear to follow a normal distribution?

```
sns.kdeplot(df['Glucose'], shade=True)
plt.xlabel('Glucose')
plt.ylabel('Density')
plt.title('Distribution of Glucose Levels')
plt.show()

print("Based on the KDE plot, the Glucose level does not appear to
follow a normal distribution. It seems to be skewed to the right.")
```

```
C:\Users\MAK TECH\AppData\Local\Temp\ipykernel_11008\3376586831.py:1:
FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(df['Glucose'], shade=True)
C:\Users\MAK TECH\Anaconda3\Lib\site-packages\seaborn\
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```




```
Based on the KDE plot, the Glucose level does not appear to follow a
normal distribution. It seems to be skewed to the right.
```

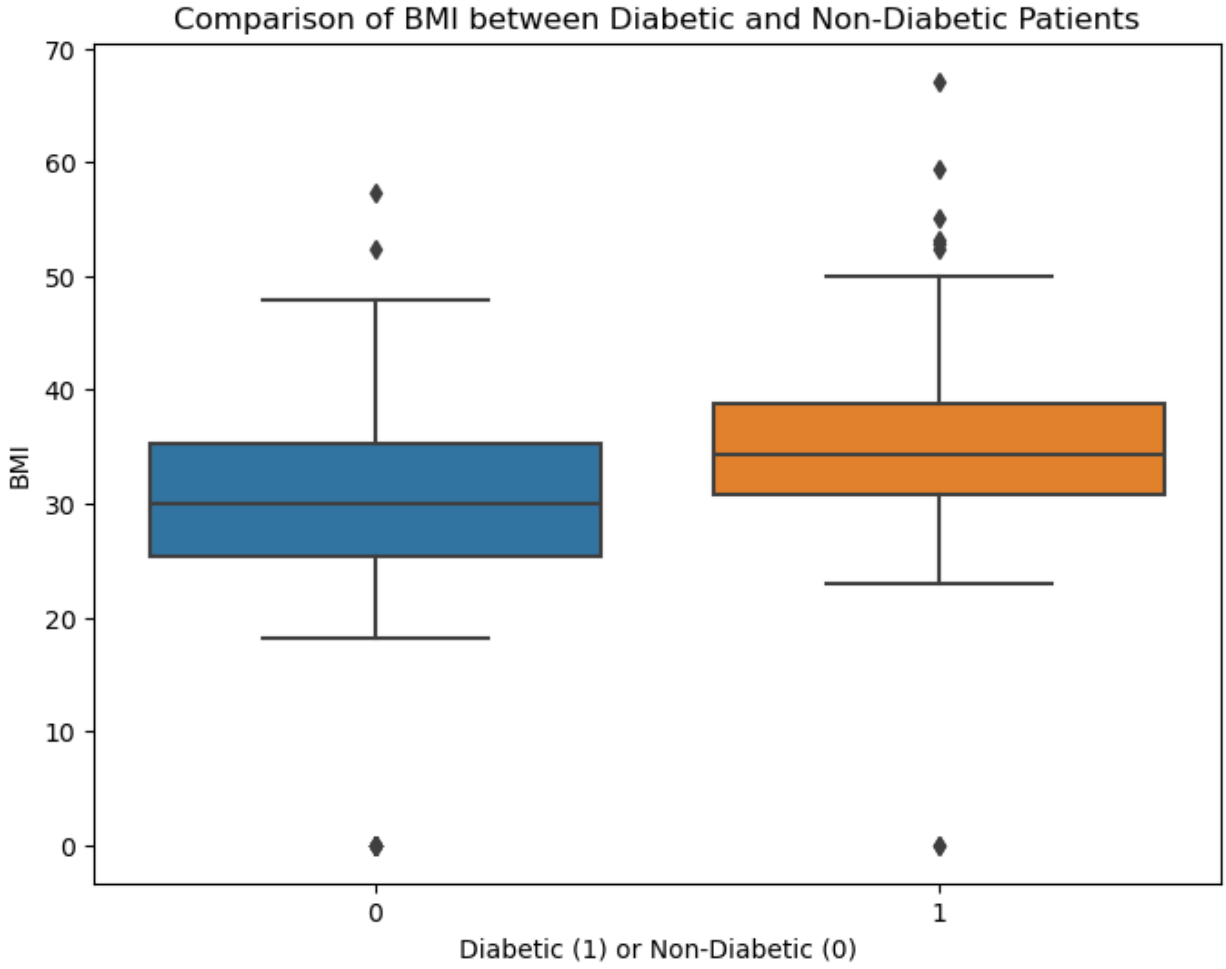
Question 14: Box Plots

a) Create a box plot comparing the BMI of diabetic and non-diabetic patients.

b) What does the box plot tell you about the BMI in relation to diabetes?

```
plt.figure(figsize=(8, 6))
sns.boxplot(x='Outcome', y='BMI', data=df)
plt.xlabel('Diabetic (1) or Non-Diabetic (0)')
plt.ylabel('BMI')
plt.title('Comparison of BMI between Diabetic and Non-Diabetic
Patients')
plt.show()

print("Based on the box plot:")
print("- Patients with diabetes tend to have a slightly higher median
BMI.")
print("- The interquartile range (IQR) of BMI is also wider for
diabetic patients, suggesting more variability in BMI among those with
```

```
diabetes.")
print("- This indicates that higher BMI might be associated with an
increased risk of diabetes.")
```




```
Based on the box plot:
- Patients with diabetes tend to have a slightly higher median BMI.
- The interquartile range (IQR) of BMI is also wider for diabetic
patients, suggesting more variability in BMI among those with
diabetes.
- This indicates that higher BMI might be associated with an increased
risk of diabetes.
```

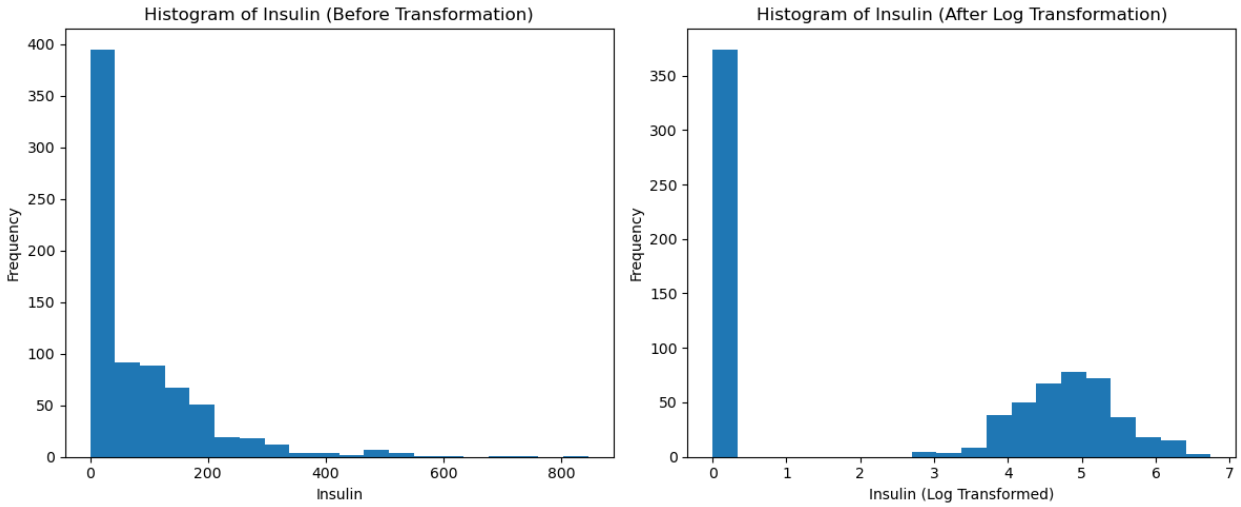
Question 15: Data Transformation

a) Apply a log transformation to the Insulin variable to reduce skewness.

b) Plot the histogram before and after the transformation.

```
import numpy as np

df['Insulin_log'] = np.log(df['Insulin'] + 1)  # Add 1 to avoid log(0)
display(df.head())

plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.hist(df['Insulin'], bins=20)
plt.xlabel('Insulin')
plt.ylabel('Frequency')
plt.title('Histogram of Insulin (Before Transformation)')

plt.subplot(1, 2, 2)
plt.hist(df['Insulin_log'], bins=20)
plt.xlabel('Insulin (Log Transformed)')
plt.ylabel('Frequency')
plt.title('Histogram of Insulin (After Log Transformation)')

plt.tight_layout()
plt.show()
```

```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin
BMI  \
0            6      148             72             35        0  33.6

1            1       85             66             29        0  26.6

2            8      183             64              0        0  23.3

3            1       89             66             23       94  28.1

4            0      137             40             35      168  43.1


   DiabetesPedigreeFunction  Age  Outcome  Insulin_log
0                     0.627   50        1     0.000000
1                     0.351   31        0     0.000000
2                     0.672   32        1     0.000000
3                     0.167   21        0     4.553877
4                     2.288   33        1     5.129899
```



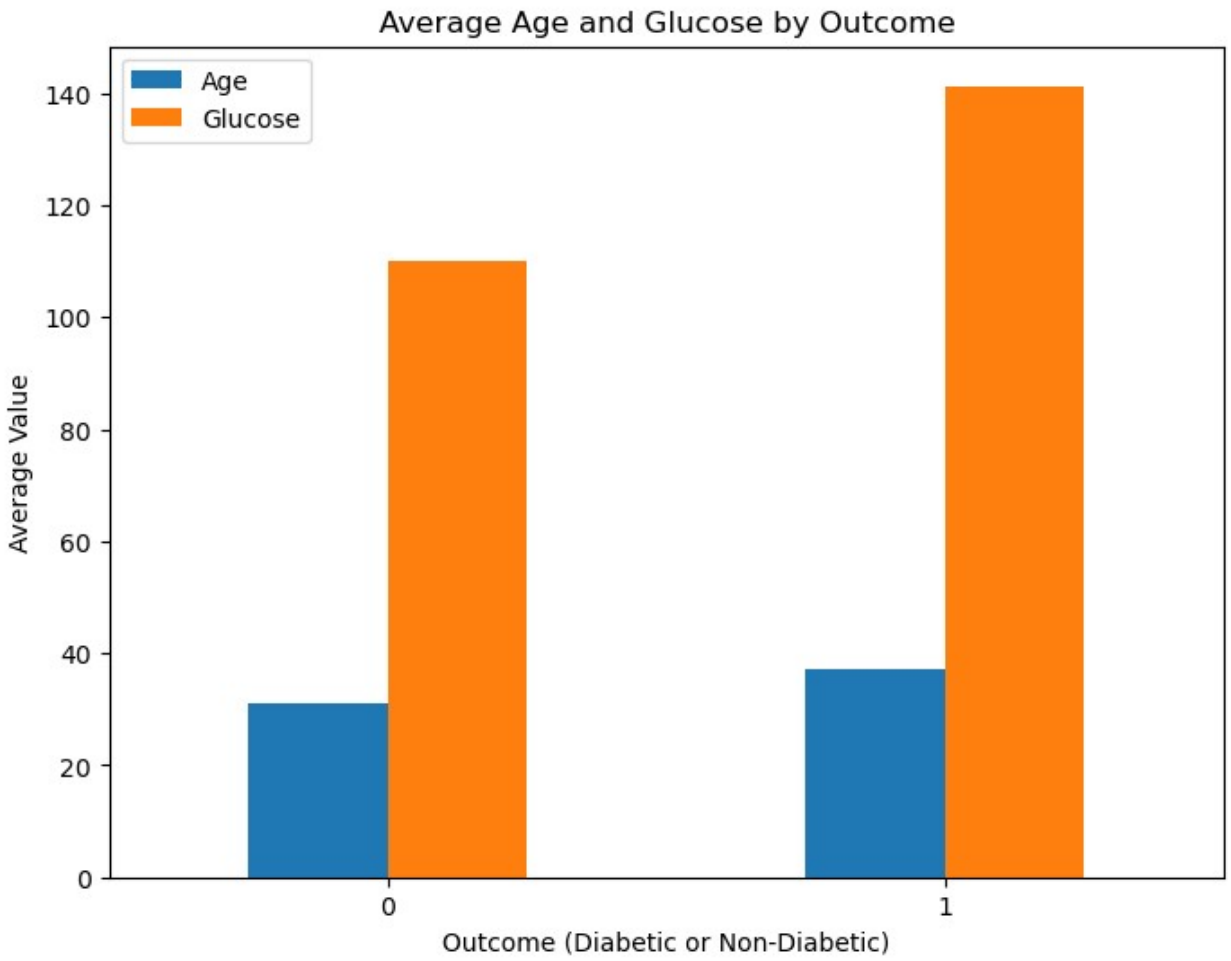
Question 16: Grouped Bar Charts¶

a) Group the Diabetes dataset by Outcome and calculate the average Age and Glucose.

b) Plot a grouped bar chart showing these averages.

```
average_by_outcome = df.groupby('Outcome')[['Age',
'Glucose']].mean().round(2)
display(average_by_outcome)

average_by_outcome.plot(kind='bar', figsize=(8, 6))
plt.xlabel('Outcome (Diabetic or Non-Diabetic)')
plt.ylabel('Average Value')
plt.title('Average Age and Glucose by Outcome')
plt.xticks(rotation=0)
plt.legend()
plt.show()
```

Outcome	Age	Glucose
0	31.19	109.98
1	37.07	141.26

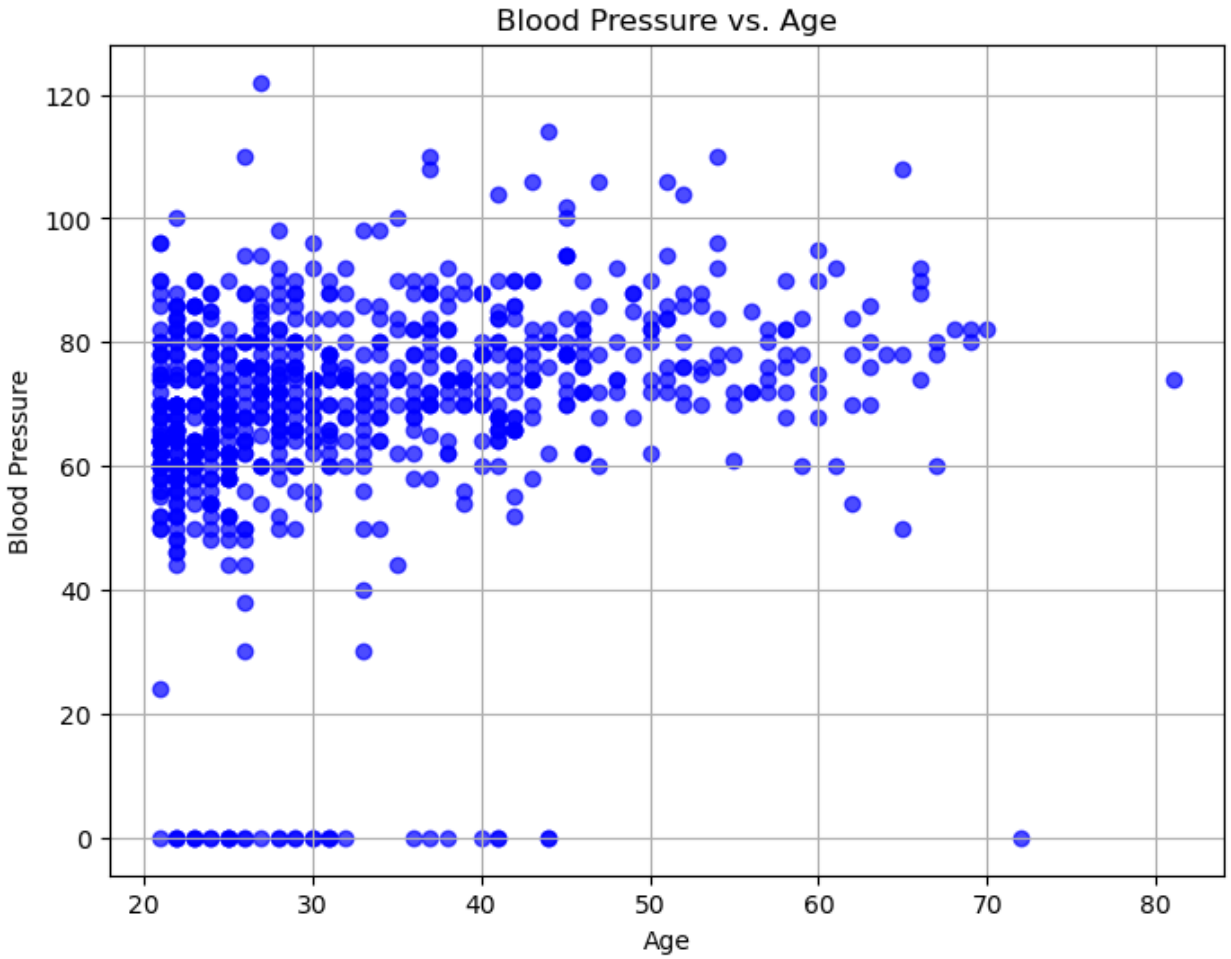


Question 17: Customizing Plots

a) Create a scatter plot of BloodPressure vs. Age.

b) Customize the plot by adding a title, axis labels, and changing the marker style.

```
plt.figure(figsize=(8, 6))
plt.scatter(df['Age'], df['BloodPressure'], marker='o', color='blue',
alpha=0.7)
plt.xlabel('Age')
plt.ylabel('Blood Pressure')
plt.title('Blood Pressure vs. Age')
plt.grid(True)
plt.show()
```



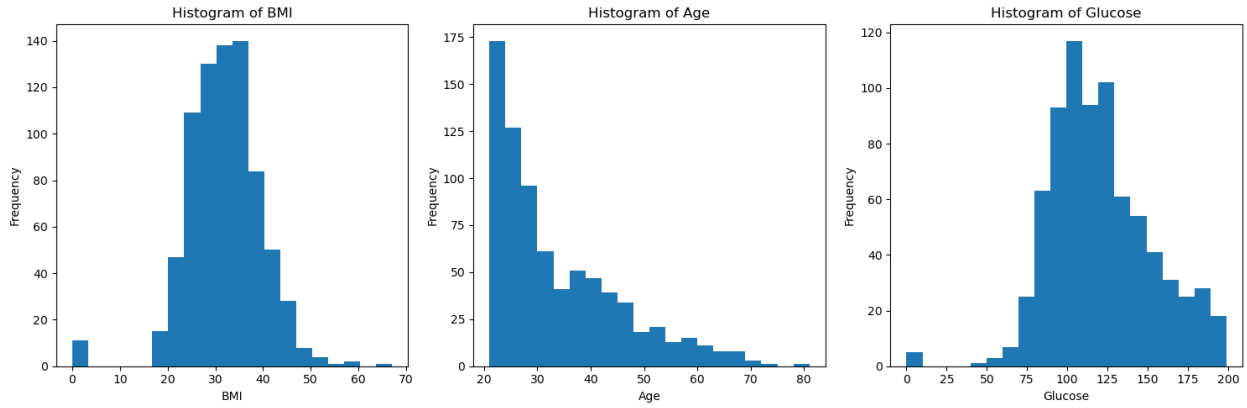
Question 18: Subplots

a) Create subplots of histograms for BMI, Age, and Glucose in a single figure.

b) Ensure that the subplots are well-arranged and labeled.

```
plt.figure(figsize=(15, 5))

plt.subplot(1, 3, 1)
plt.hist(df['BMI'], bins=20)
plt.xlabel('BMI')
plt.ylabel('Frequency')
plt.title('Histogram of BMI')

plt.subplot(1, 3, 2)
plt.hist(df['Age'], bins=20)
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.title('Histogram of Age')
```

```
plt.subplot(1, 3, 3)
plt.hist(df['Glucose'], bins=20)
plt.xlabel('Glucose')
plt.ylabel('Frequency')
plt.title('Histogram of Glucose')

plt.tight_layout()
plt.show()
```



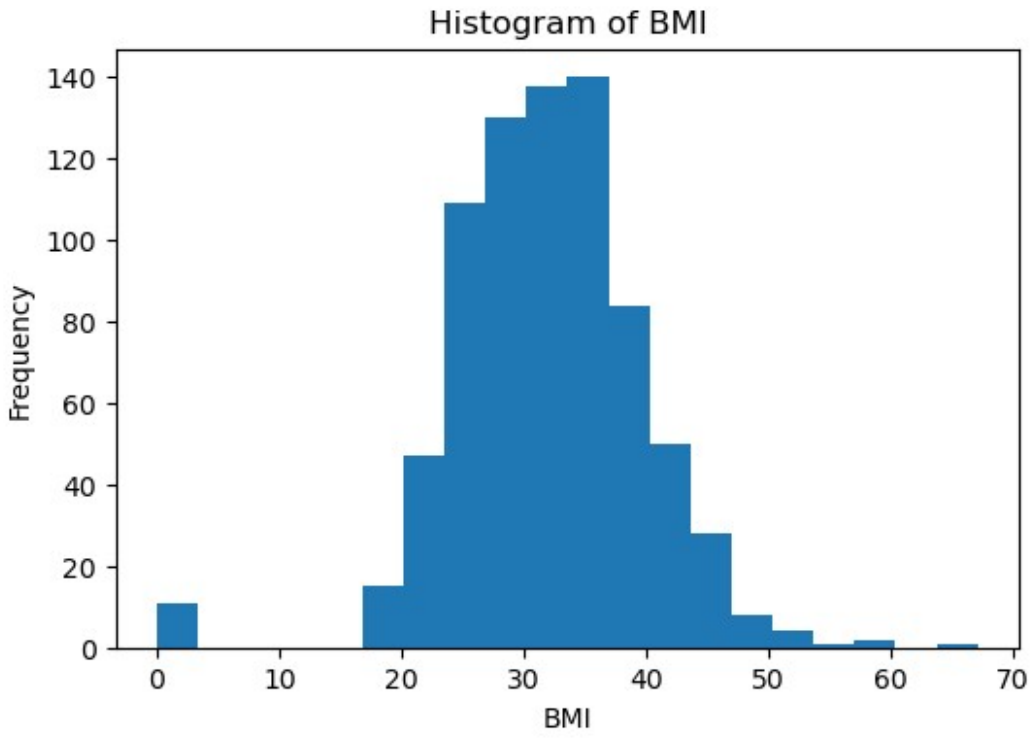
Question 19: Saving and Exporting Plots

a) Save one of your plots from Question 18 as a PNG file.

b) Show the code used to save the plot.

```
plt.figure(figsize=(6, 4))
plt.hist(df['BMI'], bins=20)
plt.xlabel('BMI')
plt.ylabel('Frequency')
plt.title('Histogram of BMI')

plt.savefig('bmi_histogram.png')
print("Plot saved as bmi_histogram.png")
```

```
Plot saved as bmi_histogram.png
```



Question 20: Final Analysis Report

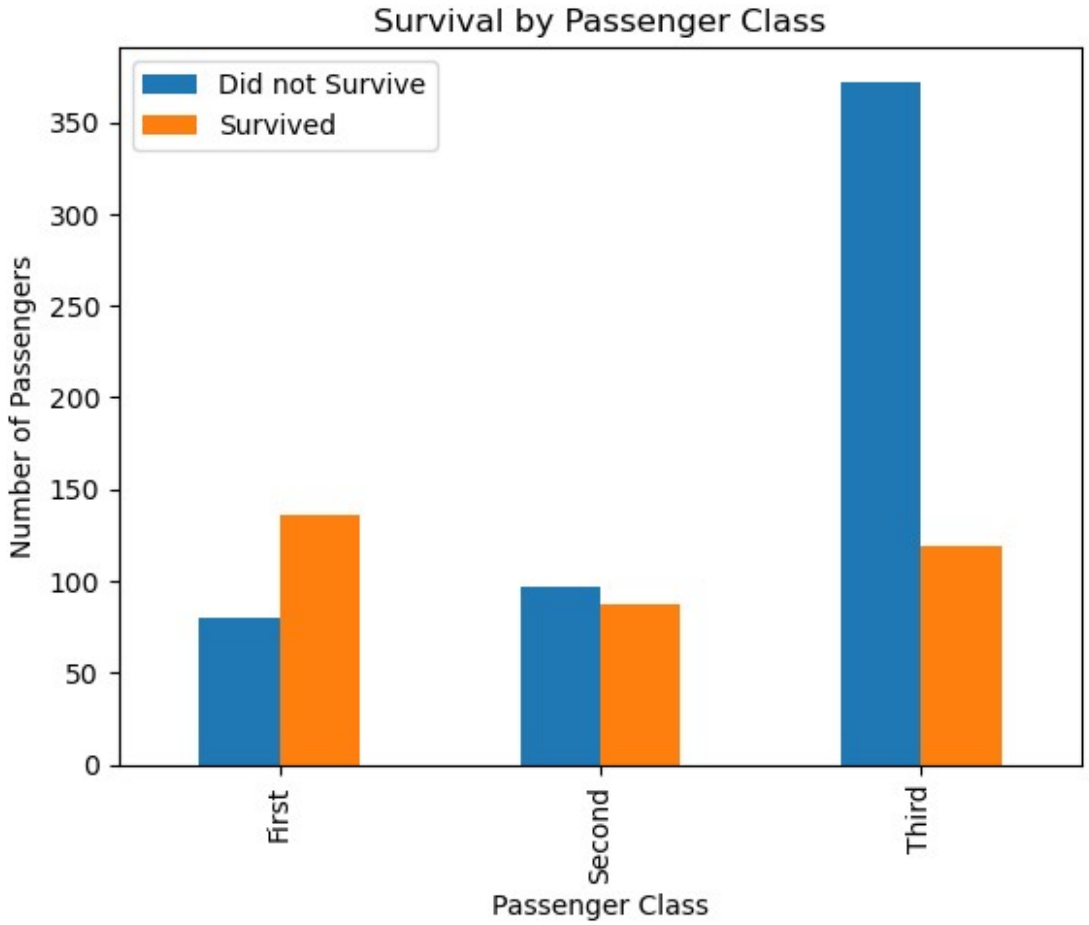
a) Write a brief report summarizing your key findings from the analyses above.

b) Include at least three visualizations to support your conclusions.

```
print("## Final Analysis Report: Titanic and Diabetes Datasets")
print("")
print("**Key Findings:**")
print("")
print("**Titanic Dataset:**")
print("- Passenger class was a major factor in survival, with first-
class passengers having a significantly higher survival rate than
those in third class.")
print("- The majority of passengers were in the age range of 20-40,
with a considerable number of children and older adults on board.")
print("- There appears to be no strong linear correlation between age
and fare.")
print("")
print("**Diabetes Dataset:**")
print("- Glucose, Insulin, and BMI appear to be positively correlated
with the outcome (diabetes diagnosis).")
print("- Patients with diabetes tend to have a higher median BMI and
more variability in BMI compared to non-diabetic patients.")
print("- The distribution of glucose is skewed to the right.")
print("")
```

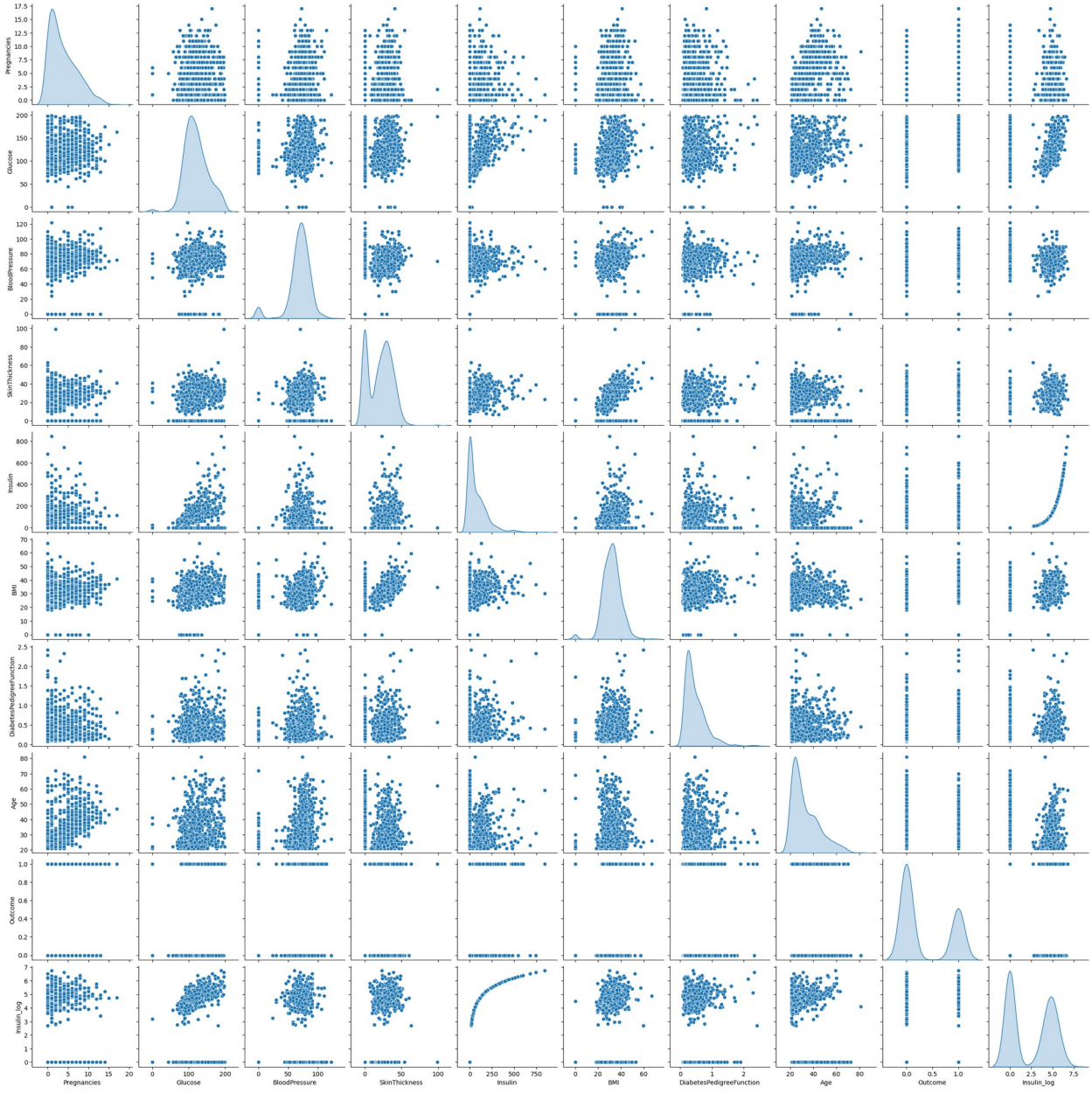
```
print("**Visualizations:**")
print("")
print("**1. Survival by Passenger Class (Titanic):**")
print("This bar plot clearly demonstrates the disparities in survival
rates across passenger classes.")

survivors_by_class = titanic_data.groupby(['class', 'survived'])
['survived'].count().unstack()
survivors_by_class.plot(kind='bar', stacked=False)  # Use stacked=True
for a stacked bar plot
plt.xlabel('Passenger Class')
plt.ylabel('Number of Passengers')
plt.title('Survival by Passenger Class')
plt.legend(['Did not Survive', 'Survived'])  # Add a legend
plt.show()

print("")
print("**2. Pair Plot (Diabetes):**")
print("The pair plot shows potential relationships and correlations
between different variables, especially highlighting the connection
between Glucose, Insulin, BMI, and diabetes.")
sns.pairplot(df, diag_kind='kde')  # Use diag_kind='kde' for kernel
density estimation plots on the diagonal
plt.show()

## Final Analysis Report: Titanic and Diabetes Datasets

**Key Findings:**

**Titanic Dataset:**
- Passenger class was a major factor in survival, with first-class
passengers having a significantly higher survival rate than those in
third class.
- The majority of passengers were in the age range of 20-40, with a
considerable number of children and older adults on board.
- There appears to be no strong linear correlation between age and
fare.

**Diabetes Dataset:**
- Glucose, Insulin, and BMI appear to be positively correlated with
the outcome (diabetes diagnosis).
- Patients with diabetes tend to have a higher median BMI and more
variability in BMI compared to non-diabetic patients.
- The distribution of glucose is skewed to the right.

**Visualizations:**

**1. Survival by Passenger Class (Titanic):**
This bar plot clearly demonstrates the disparities in survival rates
across passenger classes.
```

```
C:\Users\MAK TECH\AppData\Local\Temp\ipykernel_11008\1059397152.py:20:
FutureWarning: The default of observed=False is deprecated and will be
changed to True in a future version of pandas. Pass observed=False to
retain current behavior or observed=True to adopt the future default
and silence this warning.
  survivors_by_class = titanic_data.groupby(['class', 'survived'])
['survived'].count().unstack()
```




```
**2. Pair Plot (Diabetes):**
The pair plot shows potential relationships and correlations between
different variables, especially highlighting the connection between
Glucose, Insulin, BMI, and diabetes.

C:\Users\MAK TECH\Anaconda3\Lib\site-packages\seaborn\
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\MAK TECH\Anaconda3\Lib\site-packages\seaborn\
```

```
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\MAK TECH\Anaconda3\Lib\site-packages\seaborn\
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\MAK TECH\Anaconda3\Lib\site-packages\seaborn\
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\MAK TECH\Anaconda3\Lib\site-packages\seaborn\
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\MAK TECH\Anaconda3\Lib\site-packages\seaborn\
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\MAK TECH\Anaconda3\Lib\site-packages\seaborn\
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\MAK TECH\Anaconda3\Lib\site-packages\seaborn\
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\MAK TECH\Anaconda3\Lib\site-packages\seaborn\
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\MAK TECH\Anaconda3\Lib\site-packages\seaborn\
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```


Pregnancies
Glucose
BloodPressure
SkinThickness
Insulin
BMI
DiabetesPedigreeFunction
Age
Outcome
Insulin_log