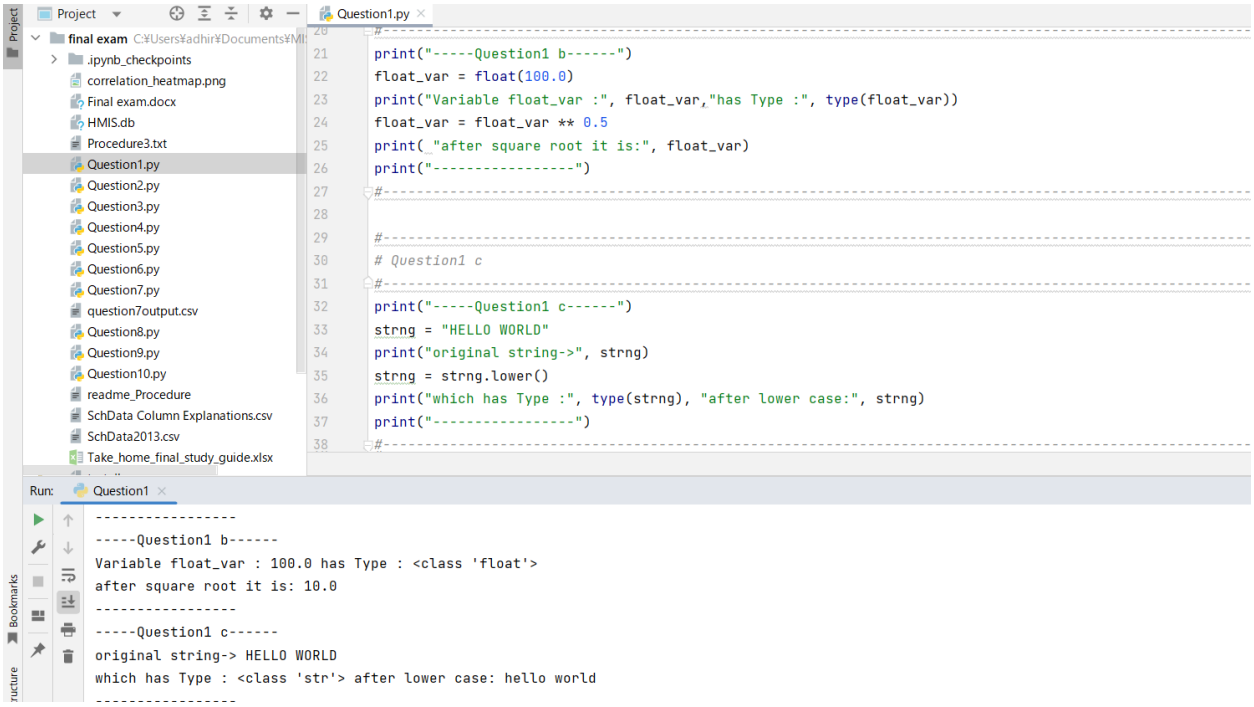
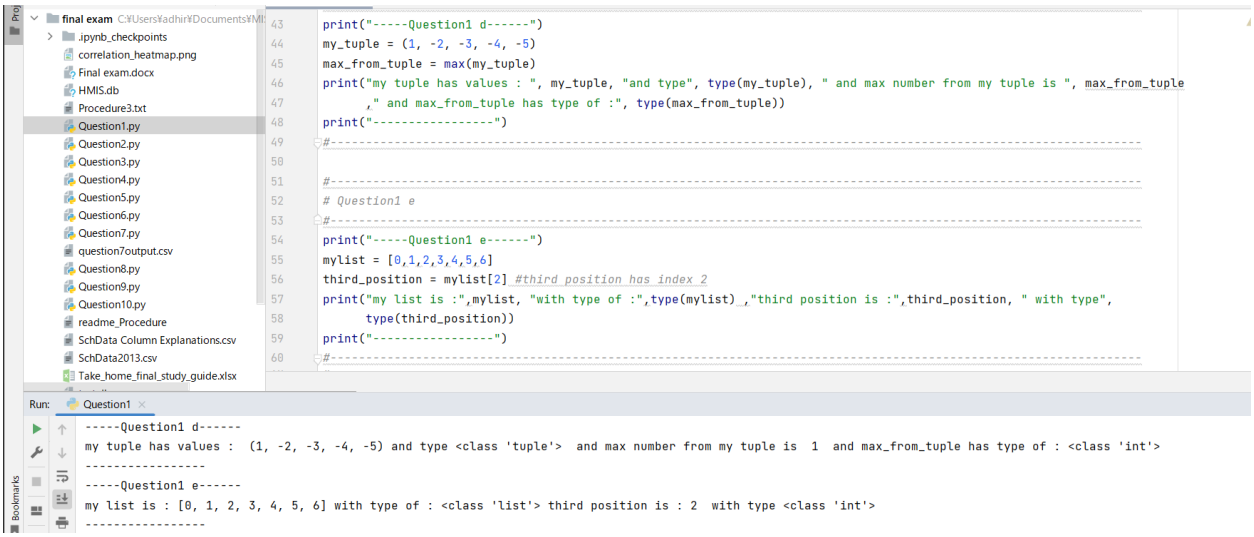
Question1


Project
final exam C:¥Users¥adhir¥Documents¥MI
.ipynb_checkpoints
correlation_heatmap.png
Final exam.docx
HMIS.db
Procedure3.txt
Question1.py
Question2.py
Question3.py
Question4.py
Question5.py
Question6.py
Question7.py
question7output.csv
Question8.py
Question9.py
Question10.py
readme_Procedure
SchData Column Explanations.csv
Question1.py
#----------
Question1 a
#----------
print("-----Question1 a------")
integer1 = int(80)
print("integer 1 is ",integer1)
print("type of integer 1 is ",type(integer1))
mult_by = int(200)
dvyd_by = int(40)
var2 = integer1*mult_by
var3 = integer1//dvyd_by
var4 = integer1*mult_by//dvyd_by
print("Var2 = integer1* 200 :" ,var2,"with type",type(var2), "Var3 integer1/ 40:" ,var3, "with type", type(var3))
print("Var4 = integer1 * 200 / 40:" ,var4,"with type",type(var4))
print("-----------------")
#----------
Run:
Question1
C:\Users\adhir\anaconda\python.exe "C:\Users\adhir\Documents\MIS502\final exam\Question1.py"
-----Question1 a------
integer 1 is 80
type of integer 1 is <class 'int'>
Var2 = integer1* 200 : 16000 with type <class 'int'> Var3 integer1/ 40: 2 with type <class 'int'>
Var4 = integer1 * 200 / 40: 400 with type <class 'int'>

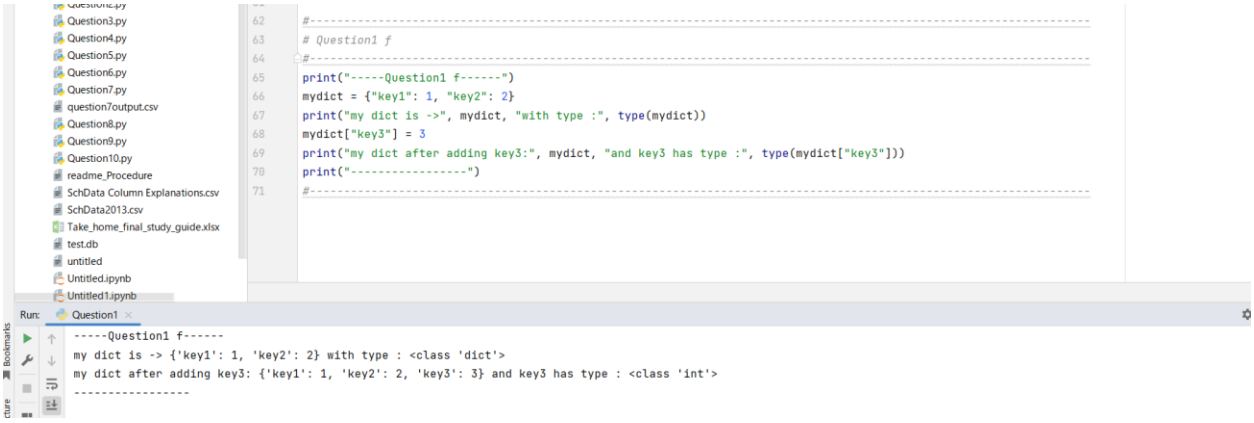

Project
final exam C:¥Users¥adhir¥Documents¥MI
.ipynb_checkpoints
correlation_heatmap.png
Final exam.docx
HMIS.db
Procedure3.txt
Question1.py
Question2.py
Question3.py
Question4.py
Question5.py
Question6.py
Question7.py
question7output.csv
Question8.py
Question9.py
Question10.py
readme_Procedure
SchData Column Explanations.csv
SchData2013.csv
Take_home_final_study_guide.xlsx
Question1.py
print("-----Question1 b------")
float_var = float(100.0)
print("Variable float_var :", float_var,"has Type :", type(float_var))
float_var = float_var ** 0.5
print("after square root it is:", float_var)
print("-----------------")
#----------
#----------
Question1 c
#----------
print("-----Question1 c------")
strng = "HELLO WORLD"
print("original string->", strng)
strng = strng.lower()
print("which has Type :", type(strng), "after lower case:", strng)
print("-----------------")
#----------
Run:
Question1

-----Question1 b------
Variable float_var : 100.0 has Type : <class 'float'>
after square root it is: 10.0

-----Question1 c------
original string-> HELLO WORLD
which has Type : <class 'str'> after lower case: hello world


final exam
.ipynb_checkpoints
correlation_heatmap.png
Final exam.docx
HMIS.db
Procedure3.txt
Question1.py
Question2.py
Question3.py
Question4.py
Question5.py
Question6.py
Question7.py
question7output.csv
Question8.py
Question9.py
Question10.py
readme_Procedure
SchData Column Explanations.csv
SchData2013.csv
Take_home_final_study_guide.xlsx
print("-----Question1 d------")
my_tuple = (1, -2, -3, -4, -5)
max_from_tuple = max(my_tuple)
print("my tuple has values : ", my_tuple, "and type", type(my_tuple), " and max number from my tuple is ", max_from_tuple
," and max_from_tuple has type of :", type(max_from_tuple))
print("-----------------")
# Question1 e
print("-----Question1 e------")
mylist = [0,1,2,3,4,5,6]
third_position = mylist[2] #third position has index 2
print("my list is :",mylist, "with type of :",type(mylist) ,"third position is :",third_position, " with type",
type(third_position))
print("-----------------")
Run:
Question1
-----Question1 d------
my tuple has values : (1, -2, -3, -4, -5) and type <class 'tuple'> and max number from my tuple is 1 and max_from_tuple has type of : <class 'int'>

-----Question1 e------
my list is : [0, 1, 2, 3, 4, 5, 6] with type of : <class 'list'> third position is : 2 with type <class 'int'>


Question3.py
Question4.py
Question5.py
Question6.py
Question7.py
question7output.csv
Question8.py
Question9.py
Question10.py
readme_Procedure
SchData Column Explanations.csv
SchData2013.csv
Take_home_final_study_guide.xlsx
test.db
untitled
Untitled.ipynb
Untitled1.ipynb
# Question1 f
print("-----Question1 f------")
mydict = {"key1": 1, "key2": 2}
print("my dict is ->", mydict, "with type :", type(mydict))
mydict["key3"] = 3
print("my dict after adding key3:", mydict, "and key3 has type :", type(mydict["key3"]))
print("-----------------")
Run:
Question1
-----Question1 f------
my dict is -> {'key1': 1, 'key2': 2} with type : <class 'dict'>
my dict after adding key3: {'key1': 1, 'key2': 2, 'key3': 3} and key3 has type : <class 'int'>

Question2

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
desired_width = 320
pd.set_option('display.width', desired_width)
pd.set_option('display.max_columns', 10)
df = pd.read_csv('SchData2013.csv', low_memory=False)
```

In [3]:
```
print(df.shape)
print(df.info())
print(df.describe())
```

```
(33173, 28)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 33173 entries, 0 to 33172
Data columns (total 28 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   FY                              33173 non-null  object
 1   SurgicalArea                    33173 non-null  object
 2   OperatingRoom                   33173 non-null  object
 3   ORCaseNumber                    33173 non-null  object
 4   SurgicalSpecialty               33173 non-null  object
 5   SSSNo                           33173 non-null  int64
 6   PatientType                     33173 non-null  int64
 7   SchedulePriority                33173 non-null  int64
 8   AddOnIndicator                  33173 non-null  object
 9   CaseDOW                         33173 non-null  object
 10  CaseMonth                       0 non-null      float64
 11  PrimaryProcedure                33173 non-null  object
 12  ScheduledStartDate_Time         33173 non-null  object
 13  CaseStartDate_Time              33173 non-null  object
 14  CaseCreateDate_Time             33173 non-null  object
 15  CaseLevel                       33173 non-null  int64
 16  ASA_Class                       33173 non-null  object
 17  PatientAge                      33173 non-null  int64
 18  PreopDiagnosis                  33173 non-null  object
 19  ScheduledCaseDuration           33173 non-null  int64
 20  PrimaryProcedureSurgeonComment  33173 non-null  object
 21  TotalSurgeryMin                 33173 non-null  int64
 22  PatientInRoomMin                33173 non-null  int64
 23  SetUpMin                        33173 non-null  int64
 24  CleanUpMin                      33173 non-null  int64
 25  SchCreateDays                   33173 non-null  float64
 26  RoomNo                          33173 non-null  object
 27  OTS                             33173 non-null  int64
dtypes: float64(2), int64(11), object(15)
memory usage: 7.1+ MB
None
              SSSNo   PatientType  SchedulePriority  CaseMonth     CaseLevel  ...  PatientInRoomMin      SetUpMin    CleanUp
Min  SchCreateDays           OTS
count  33173.000000  33173.000000      33173.000000        0.0  33173.000000  ...      33173.000000  33173.000000  33173.000
000   33173.000000  33173.000000
mean       7.721129      1.421940          1.960450        NaN      2.963253  ...        150.605523     30.173967     12.481
235      10.808039      0.296175
std        4.300396      0.512332          0.194903        NaN      1.661193  ...        117.708167     30.992437     14.366
459      16.022238      0.456576
min        1.000000      0.000000          1.000000        NaN      0.000000  ...          0.000000      0.000000      0.000
000       0.000000      0.000000
25%        4.000000      1.000000          2.000000        NaN      2.000000  ...         72.000000     20.000000     10.000
000       0.903500      0.000000
50%        8.000000      1.000000          2.000000        NaN      3.000000  ...        121.000000     25.000000     10.000
000       4.877100      0.000000
75%       11.000000      2.000000          2.000000        NaN      4.000000  ...        195.000000     36.000000     15.000
000      14.684000      1.000000
max       18.000000      2.000000          2.000000        NaN      9.000000  ...       1536.000000   1490.000000   1467.000
000     317.906300      1.000000

[8 rows x 13 columns]
```

Proof that null data replaced in data types int64 and float64

In [4]:

```
# Fill missing values in int64 columns with the mean value for that column
int64_columns = df.select_dtypes(include=['int64']).columns
df[int64_columns] = df[int64_columns].fillna(df.mean())

# Fill missing values in float64 columns with the mean value for that column
float64_columns = df.select_dtypes(include=['float64']).columns
df[float64_columns] = df[float64_columns].fillna(df.mean())

print(df['PatientInRoomMin'].isnull().sum())
print(df['SchCreateDays'].isnull().sum())
```

```
C:\Users\adhir\AppData\Local\Temp\ipykernel_8136\2417231367.py:3: FutureWarning: Dropping of nuisance columns in DataFrame r
eductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid column
s before calling the reduction.
  df[int64_columns] = df[int64_columns].fillna(df.mean())
```

```
0
0
```

```
C:\Users\adhir\AppData\Local\Temp\ipykernel_8136\2417231367.py:7: FutureWarning: Dropping of nuisance columns in DataFrame r
eductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid column
s before calling the reduction.
  df[float64_columns] = df[float64_columns].fillna(df.mean())
```

Deleting categorical data

In [7]:

```
df = df.drop('CaseMonth', axis=1)
df = df.dropna()
cat_cols = [col for col in df.columns if df[col].dtype == 'object']
df = df.drop(cat_cols, axis=1)
df = df.reset_index(drop=True)
print(df.info())
```

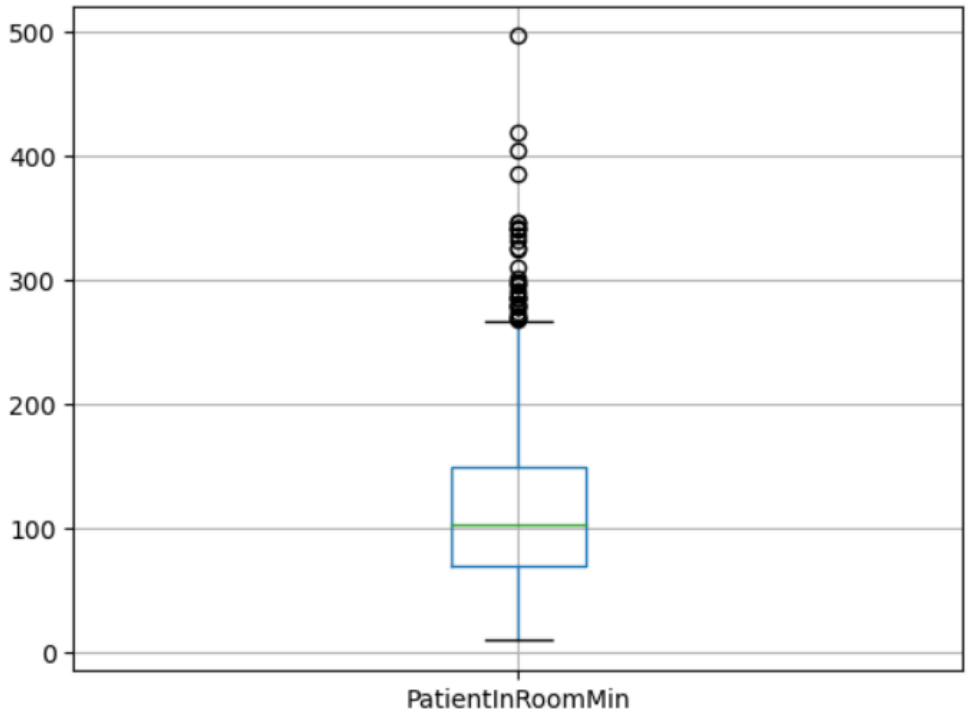
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 33173 entries, 0 to 33172
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   SSSNo                 33173 non-null  int64
 1   PatientType           33173 non-null  int64
 2   SchedulePriority      33173 non-null  int64
 3   CaseLevel             33173 non-null  int64
 4   PatientAge            33173 non-null  int64
 5   ScheduledCaseDuration 33173 non-null  int64
 6   TotalSurgeryMin       33173 non-null  int64
 7   PatientInRoomMin      33173 non-null  int64
 8   SetUpMin              33173 non-null  int64
 9   CleanUpMin            33173 non-null  int64
 10  SchCreateDays         33173 non-null  float64
 11  OTS                   33173 non-null  int64
dtypes: float64(1), int64(11)
memory usage: 3.0 MB
None
```

In [5]:
```
diced_df = df[(df["SurgicalArea"] == "HHOR") & (df["SSSNo"] == 9) & (df["PatientType"] == 2)]

# calculate the mean and median of PatientInRoomMin for the diced dataframe
mean = diced_df["PatientInRoomMin"].mean()
median = diced_df["PatientInRoomMin"].median()

print(diced_df.head(10))
print("Mean of PatientInRoomMin: {:.2f}".format(mean))
print("Median of PatientInRoomMin: {:.2f}".format(median))
```

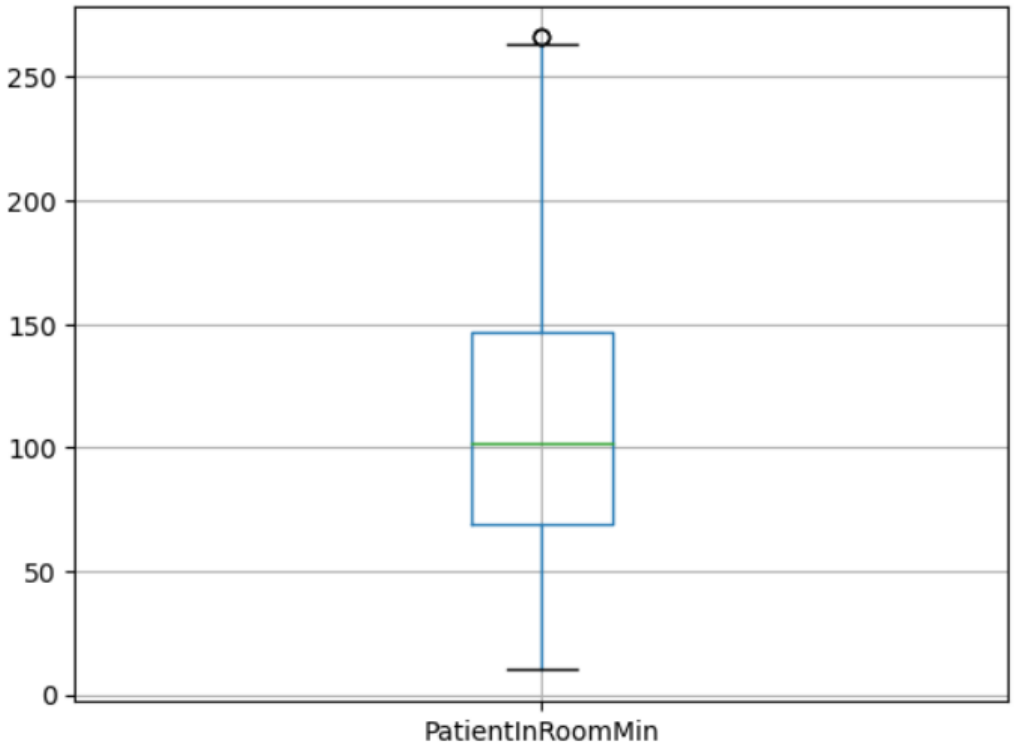
```
        FY SurgicalArea OperatingRoom   ORCaseNumber SurgicalSpecialty  ...  SetUpMin  CleanUpMin  SchCreateDays RoomNo OTS
632   FY13         HHOR      HH OR 02  HH-2012-11642             ORTHO  ...        50          20        32.0819      2   1
1126  FY13         HHOR      HH OR 02   HH-2012-9355             ORTHO  ...        40          10         6.5986      2   0
1268  FY13         HHOR      HH OR 06   HH-2013-2106             ORTHO  ...        60           0         3.9076      6   1
1479  FY13         HHOR      HH OR 10   HH-2013-5954           ORT/TRA  ...        30          14        19.8743     10   0
2052  FY13         HHOR      HH OR 02   HH-2013-7917             ORTHO  ...        30           9         1.7861      2   1
2053  FY13         HHOR      HH OR 02   HH-2013-9049             ORTHO  ...        25          33         4.7083      2   0
2121  FY13         HHOR      HH OR 02  HH-2012-11430             ORTHO  ...        30          10        15.8146      2   0
2125  FY13         HHOR      HH OR 14   HH-2013-4534           ORT/TRA  ...        25           9        18.7250     14   0
2168  FY13         HHOR      HH OR 02   HH-2012-8981             ORTHO  ...        50          20         0.6778      2   1
2241  FY13         HHOR      HH OR 01   HH-2013-3250             ORTHO  ...        19          10         0.8951      1   1

[10 rows x 28 columns]
Mean of PatientInRoomMin: 113.67
Median of PatientInRoomMin: 103.50
```

In [6]:
```
# create a boxplot for PatientInRoomMin
diced_df.boxplot(column=["PatientInRoomMin"])
plt.show()

# calculate the IQR and remove outliers
Q1 = diced_df["PatientInRoomMin"].quantile(0.25)
Q3 = diced_df["PatientInRoomMin"].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5*IQR
upper_bound = Q3 + 1.5*IQR
diced_df = diced_df[(diced_df["PatientInRoomMin"] >= lower_bound) & (diced_df["PatientInRoomMin"] <= upper_bound)]

# create a boxplot for PatientInRoomMin after removing outliers
diced_df.boxplot(column=["PatientInRoomMin"])
plt.show()
```

Before reducing outliers



After reducing outliers :



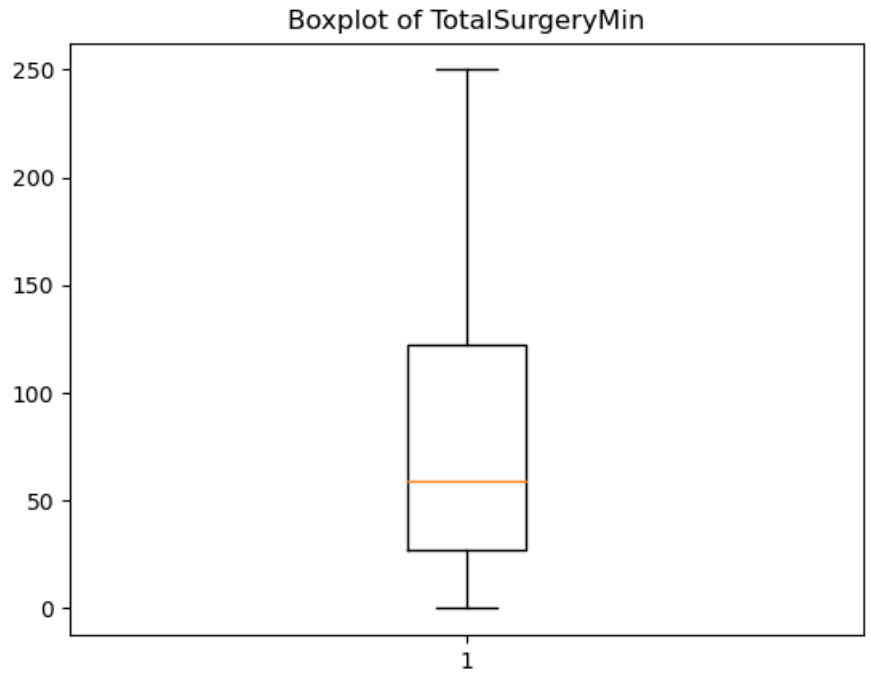
Question 3

Question3.py

```python
import pandas as pd
import matplotlib.pyplot as plt

desired_width = 400
pd.set_option('display.width', desired_width)
pd.set_option('display.max_columns', 10)
df = pd.read_csv('SchData2013.csv', low_memory=False)
print(df.info())

df2 = df.groupby('SurgicalSpecialty')['ScheduledCaseDuration'].median().reset_index()
df2.columns = ['SurgicalSpecialty', 'ScheduledCaseDurationMedian']

merged_df = pd.merge(df, df2, on='SurgicalSpecialty')
filtered_df = merged_df[(merged_df['SurgicalSpecialty'] == 'URO') & (merged_df['TotalSurgeryMin'] <= 250)]

print("Mean of TotalSurgeryMin:", filtered_df['TotalSurgeryMin'].mean())
print("Median of TotalSurgeryMin:", filtered_df['TotalSurgeryMin'].median())

plt.boxplot(filtered_df['TotalSurgeryMin'])
plt.title("Boxplot of TotalSurgeryMin")
plt.show()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 33173 entries, 0 to 33172
Data columns (total 28 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   FY                               33173 non-null  object
 1   SurgicalArea                     33173 non-null  object
 2   OperatingRoom                    33173 non-null  object
 3   ORCaseNumber                     33173 non-null  object
 4   SurgicalSpecialty                33173 non-null  object
 5   SSSNo                            33173 non-null  int64
 6   PatientType                      33173 non-null  int64
 7   SchedulePriority                 33173 non-null  int64
 8   AddOnIndicator                   33173 non-null  object
 9   CaseDOW                          33173 non-null  object
 10  CaseMonth                        0 non-null      float64
 11  PrimaryProcedure                 33173 non-null  object
 12  ScheduledStartDate_Time          33173 non-null  object
 13  CaseStartDate_Time               33173 non-null  object
 14  CaseCreateDate_Time              33173 non-null  object
 15  CaseLevel                        33173 non-null  int64
 16  ASA_Class                        33173 non-null  object
 17  PatientAge                       33173 non-null  int64
 18  PreopDiagnosis                   33173 non-null  object
 19  ScheduledCaseDuration            33173 non-null  int64
 20  PrimaryProcedureSurgeonComment   33173 non-null  object
 21  TotalSurgeryMin                  33173 non-null  int64
 22  PatientInRoomMin                 33173 non-null  int64
```

```
 20  PrimaryProcedureSurgeonComment  33173 non-null  object
 21  TotalSurgeryMin                 33173 non-null  int64
 22  PatientInRoomMin                33173 non-null  int64
 23  SetUpMin                        33173 non-null  int64
 24  CleanUpMin                      33173 non-null  int64
 25  SchCreateDays                   33173 non-null  float64
 26  RoomNo                          33173 non-null  object
 27  OTS                             33173 non-null  int64
dtypes: float64(2), int64(11), object(15)
memory usage: 7.1+ MB
None
Mean of TotalSurgeryMin: 79.97558070279929
Median of TotalSurgeryMin: 59.0

Process finished with exit code 0
```



Question 4

Question4.py

```python
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error

desired_width = 320
pd.set_option('display.width', desired_width)
pd.set_option('display.max_columns', 18)
df = pd.read_csv('SchData2013.csv', low_memory=False)
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 33173 entries, 0 to 33172
Data columns (total 28 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   FY                              33173 non-null  object
 1   SurgicalArea                    33173 non-null  object
 2   OperatingRoom                   33173 non-null  object
 3   ORCaseNumber                    33173 non-null  object
 4   SurgicalSpecialty               33173 non-null  object
 5   SSSNo                           33173 non-null  int64
 6   PatientType                     33173 non-null  int64
 7   SchedulePriority                33173 non-null  int64
 8   AddOnIndicator                  33173 non-null  object
 9   CaseDOW                         33173 non-null  object
 10  CaseMonth                       0 non-null      float64
 11  PrimaryProcedure                33173 non-null  object
 12  ScheduledStartDate_Time         33173 non-null  object
 13  CaseStartDate_Time              33173 non-null  object
 14  CaseCreateDate_Time             33173 non-null  object
 15  CaseLevel                       33173 non-null  int64
 16  ASA_Class                       33173 non-null  object
 17  PatientAge                      33173 non-null  int64
 18  PreopDiagnosis                  33173 non-null  object
 19  ScheduledCaseDuration           33173 non-null  int64
 20  PrimaryProcedureSurgeonComment  33173 non-null  object
 21  TotalSurgeryMin                 33173 non-null  int64
 22  PatientInRoomMin                33173 non-null  int64
```

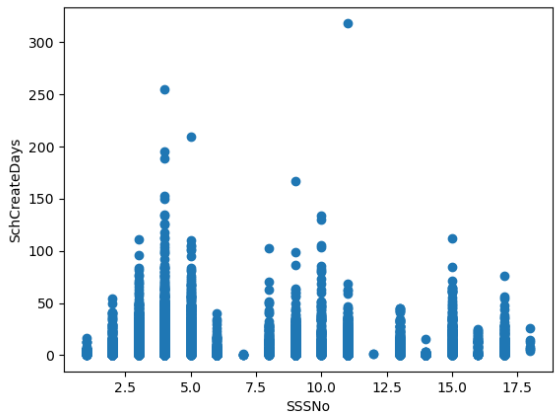
```python
# Identify int64 and float64 variables
int_vars = df.select_dtypes(include=['int64']).columns
float_vars = df.select_dtypes(include=['float64']).columns
print('Int Variables:', int_vars)
print('Float Variables:', float_vars)

# Identify all numeric columns
numeric_cols = df.select_dtypes(include=['int64', 'float64']).columns
#AND
# Replace missing values with column mean
df[numeric_cols] = df[numeric_cols].fillna(df[numeric_cols].mean())

#verifying if there is no null values in any columns
print(df.isnull().sum())

```

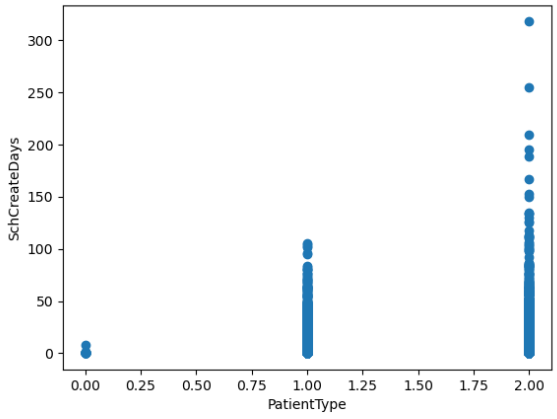
```
dtypes: float64(2), int64(11), object(15)
memory usage: 7.1+ MB
None
Int Variables: Index(['SSSNo', 'PatientType', 'SchedulePriority', 'CaseLevel', 'PatientAge', 'ScheduledCaseDuration', 'TotalSurgeryMin', 'PatientInRoomMin', 'SetUpMin', 'CleanU
Float Variables: Index(['CaseMonth', 'SchCreateDays'], dtype='object')
FY                                  0
SurgicalArea                        0
OperatingRoom                       0
ORCaseNumber                        0
SurgicalSpecialty                   0
SSSNo                               0
PatientType                         0
SchedulePriority                    0
AddOnIndicator                      0
CaseDOW                             0
CaseMonth                       33173
PrimaryProcedure                    0
ScheduledStartDate_Time             0
CaseStartDate_Time                  0
CaseCreateDate_Time                 0
CaseLevel                           0
ASA_Class                           0
PatientAge                          0
PreopDiagnosis                      0
ScheduledCaseDuration               0
PrimaryProcedureSurgeonComment      0
TotalSurgeryMin                     0
PatientInRoomMin                    0
SetUpMin                            0
```

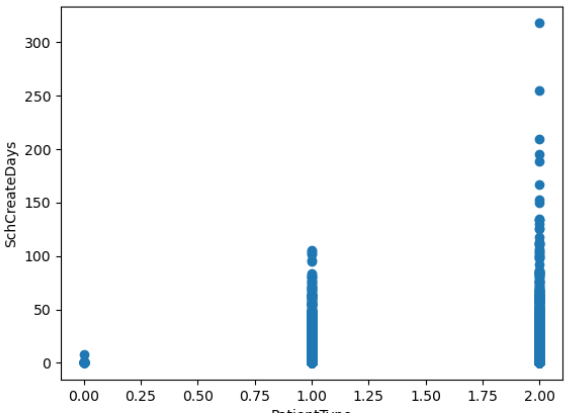
```python
# Slice X
X = df.loc[:, ['SSSNo', 'PatientType', 'SchedulePriority', 'PatientAge',
               'ScheduledCaseDuration', 'TotalSurgeryMin', 'PatientInRoomMin', 'SetUpMin']]

# Slice y
y = df.loc[:, ['SchCreateDays']]

# Split X and y into training and test data subsets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Fit a linear regression model on the training data
reg = LinearRegression().fit(X_train, y_train)

# Predict y values for test data
y_pred = reg.predict(X_test)

# Calculate the R2 score, MAE, MSE, and RMSE
r2 = r2_score(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = mean_squared_error(y_test, y_pred, squared=False)

print("R2 Score:", r2)
print("Mean Absolute Error (MAE):", mae)
print("Mean Squared Error (MSE):", mse)
print("Root Mean Squared Error (RMSE):", rmse)
```

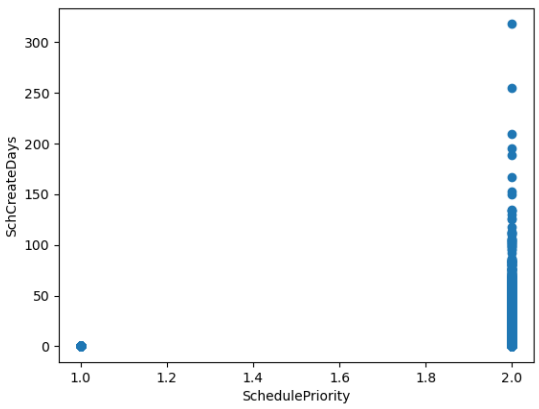
```
PrimaryProcedureSurgeonComment        0
TotalSurgeryMin                       0
PatientInRoomMin                      0
SetUpMin                              0
CleanUpMin                            0
SchCreateDays                         0
RoomNo                                0
OTS                                   0
dtype: int64
R2 Score: 0.10722919276612874
Mean Absolute Error (MAE): 9.68828364218131
Mean Squared Error (MSE): 247.95187348003623
Root Mean Squared Error (RMSE): 15.746487655348295
```

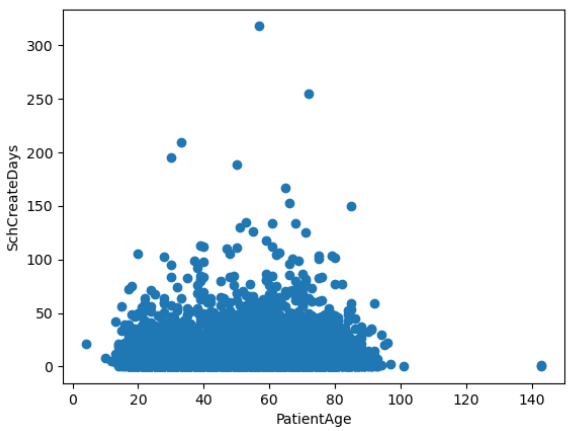
```python
# Scatterplot of X variables and y
for col in X.columns:
    fig, ax = plt.subplots()
    ax.scatter(X_test[col], y_test)
    ax.set_xlabel(col)
    ax.set_ylabel('SchCreateDays')
    plt.show()
```

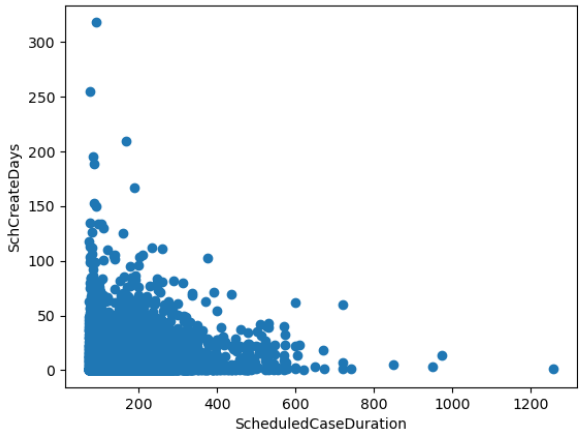

SchCreateDays
SSSNo

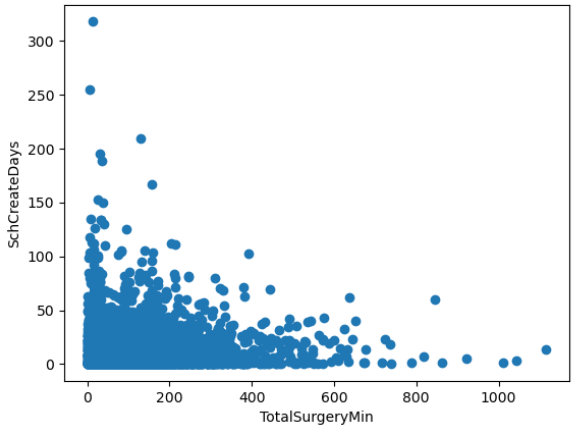

SchCreateDays
PatientType

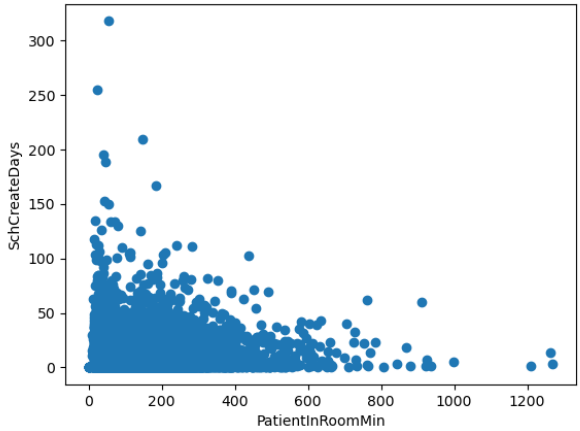

SchCreateDays

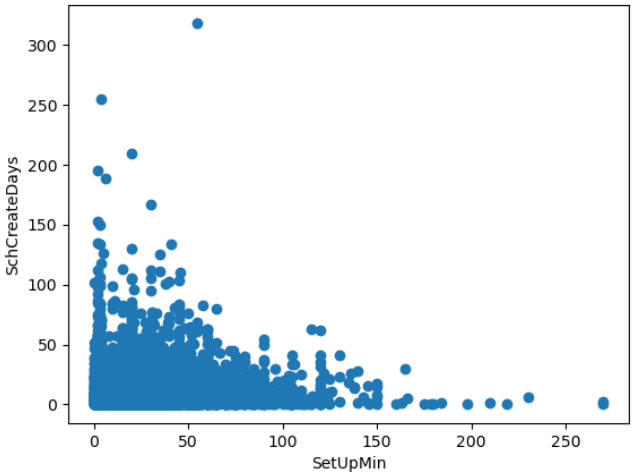

SchCreateDays
SchedulePriority


SchCreateDays
PatientAge


SchCreateDays
ScheduledCaseDuration







Question 5

Performed 3 of them, K means KNN and Linear Regression but for the comment of exam question:

The method I chose, which is K Nearest Neighbors (KNN) regression, is useful for this dataset because it is a non-parametric method that can handle both numerical and categorical data, K Nearest Neighbors (KNN) regression is a good choice for this dataset.

Question5.py

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
from sklearn.cluster import KMeans
from sklearn.metrics import adjusted_rand_score
from sklearn.neighbors import KNeighborsRegressor

desired_width = 320
pd.set_option('display.width', desired_width)
pd.set_option('display.max_columns', 18)
df = pd.read_csv('SchData2013.csv', low_memory=False)
print(df.info())

# Identify int64 and float64 variables
int_vars = df.select_dtypes(include=['int64']).columns
float_vars = df.select_dtypes(include=['float64']).columns
print('Int Variables:', int_vars)
print('Float Variables:', float_vars)
# Identify all numeric columns
numeric_cols = df.select_dtypes(include=['int64', 'float64']).columns
#AND
# Replace missing values with column mean
df[numeric_cols] = df[numeric_cols].fillna(df[numeric_cols].mean())
#verifying if there is no null values in any columns
print(df.isnull().sum())
df = df.drop('CaseMonth', axis=1)
df = df.dropna()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 33173 entries, 0 to 33172
Data columns (total 28 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   FY                              33173 non-null  object
 1   SurgicalArea                    33173 non-null  object
 2   OperatingRoom                   33173 non-null  object
 3   ORCaseNumber                    33173 non-null  object
 4   SurgicalSpecialty               33173 non-null  object
 5   SSSNo                           33173 non-null  int64
 6   PatientType                     33173 non-null  int64
 7   SchedulePriority                33173 non-null  int64
 8   AddOnIndicator                  33173 non-null  object
 9   CaseDOW                         33173 non-null  object
 10  CaseMonth                       0 non-null      float64
 11  PrimaryProcedure                33173 non-null  object
 12  ScheduledStartDate_Time         33173 non-null  object
 13  CaseStartDate_Time              33173 non-null  object
 14  CaseCreateDate_Time             33173 non-null  object
 15  CaseLevel                       33173 non-null  int64
 16  ASA_Class                       33173 non-null  object
 17  PatientAge                      33173 non-null  int64
 18  PreopDiagnosis                  33173 non-null  object
 19  ScheduledCaseDuration           33173 non-null  int64
 20  PrimaryProcedureSurgeonComment  33173 non-null  object
 21  TotalSurgeryMin                 33173 non-null  int64
 22  PatientInRoomMin                33173 non-null  int64
```

```
dtypes: float64(2), int64(11), object(15)
memory usage: 7.1+ MB
None
Int Variables: Index(['SSSNo', 'PatientType', 'SchedulePriority', 'CaseLevel', 'PatientAge', 'ScheduledCaseDuration', 'TotalSurgeryMin', 'PatientInRoomMin', 'SetUpMin', 'CleanUp
Float Variables: Index(['CaseMonth', 'SchCreateDays'], dtype='object')
FY                                0
SurgicalArea                      0
OperatingRoom                     0
ORCaseNumber                      0
SurgicalSpecialty                 0
SSSNo                             0
PatientType                       0
SchedulePriority                  0
AddOnIndicator                    0
CaseDOW                           0
CaseMonth                     33173
PrimaryProcedure                  0
ScheduledStartDate_Time           0
CaseStartDate_Time                0
CaseCreateDate_Time               0
CaseLevel                         0
ASA_Class                         0
PatientAge                        0
PreopDiagnosis                    0
ScheduledCaseDuration             0
PrimaryProcedureSurgeonComment    0
```

```python
#----------Linear Regression-------
num_cols = ['SSSNo', 'PatientType', 'SchedulePriority', 'PatientAge', 'ScheduledCaseDuration',
            'TotalSurgeryMin', 'PatientInRoomMin', 'SetUpMin', 'SchCreateDays']
data = df[num_cols]
X_train, X_test, y_train, y_test = train_test_split(data.iloc[:, :-1], data.iloc[:, -1], test_size=0.2, random_state=42)
# Create a linear regression object
reg = LinearRegression()
# Fit the model using the training data
reg.fit(X_train, y_train)
# Make predictions on the test data
y_pred = reg.predict(X_test)
print("R-squared:", r2_score(y_test, y_pred))
print("Mean Absolute Error:", mean_absolute_error(y_test, y_pred))
print("Mean Squared Error:", mean_squared_error(y_test, y_pred))

#-------------K means clustering analysis---------
# Split the dataset into training and testing sets
X = df[['SSSNo', 'PatientType', 'SchedulePriority', 'PatientAge', 'ScheduledCaseDuration', 'TotalSurgeryMin',
        'PatientInRoomMin', 'SetUpMin']]
y = df['SchCreateDays']
X_train2, X_test2, y_train2, y_test2 = train_test_split(X, y, test_size=0.2, random_state=0)
# Apply KMeans clustering to the data
kmeans = KMeans(n_clusters=3, random_state=0)
kmeans.fit(X_train2)
# Predict the clusters for the test data
y_pred = kmeans.predict(X_test2)
# Evaluate the clustering performance using adjusted rand score
print('Adjusted Rand Score:', adjusted_rand_score(y_test2, y_pred))
#-----------------------------------------------
```

```python
# Apply KNN Regression to the data -----------------
knn = KNeighborsRegressor(n_neighbors=5)
knn.fit(X_train, y_train)
# Predict the values for the test data
y_pred = knn.predict(X_test)
# Evaluate the model performance using R2 score, mean absolute error, and root mean squared error
print('R2 Score:', r2_score(y_test, y_pred))
print('Mean Absolute Error:', mean_absolute_error(y_test, y_pred))
print('Root Mean Squared Error:', mean_squared_error(y_test, y_pred, squared=False))
#-----------------------------------------------
```

```
TotalSurgeryMin                 0
PatientInRoomMin                0
SetUpMin                        0
CleanUpMin                      0
SchCreateDays                   0
RoomNo                          0
OTS                             0
dtype: int64
R-squared: 0.10722919276612874
Mean Absolute Error: 9.68828364218131
Mean Squared Error: 247.95187348003623
C:\Users\adhir\anaconda\lib\site-packages\sklearn\metrics\cluster\_supervised.py:65: UserWarning: Clustering metrics expects discrete values but received continuous values for l
  warnings.warn(msg, UserWarning)
Adjusted Rand Score: 0.00018606851997644266
R2 Score: -0.05154125630289985
Mean Absolute Error: 10.766676521477015
Root Mean Squared Error: 17.089401958452815

Process finished with exit code 0
```

Question6

```python
# sqlite import
import sqlite3
# establish connection to SQLite test.db
con = sqlite3.connect("HMIS.db")
# assign variable to con.cursor() function
cur = con.cursor()
# result of SQL script #1 execution iterated by row
for row in cur.execute(
    """
    SELECT Application || " --- " || Status AS App_Stat_Rural_MA, COUNT(*)
    FROM (
        SELECT Name, Application, Status, City, State, LENGTH(Name) AS name_length, UPPER(State) AS state_upper
        FROM (
            SELECT Name, Application, Status, City, State
            FROM LEADS
            WHERE State = "MA"
        )
        WHERE City NOT IN ("Boston", "Cambridge", "Lowell", "Springfield", "Worcester", "Newton")
        ORDER BY City, Name
    )
    GROUP BY App_Stat_Rural_MA, name_length, state_upper
    ORDER BY COUNT(*) DESC
    LIMIT 10;
"""
):
    # print SQL projection row by row with the 'for loop'
    print(row)
# close the database connection
con.close()
```

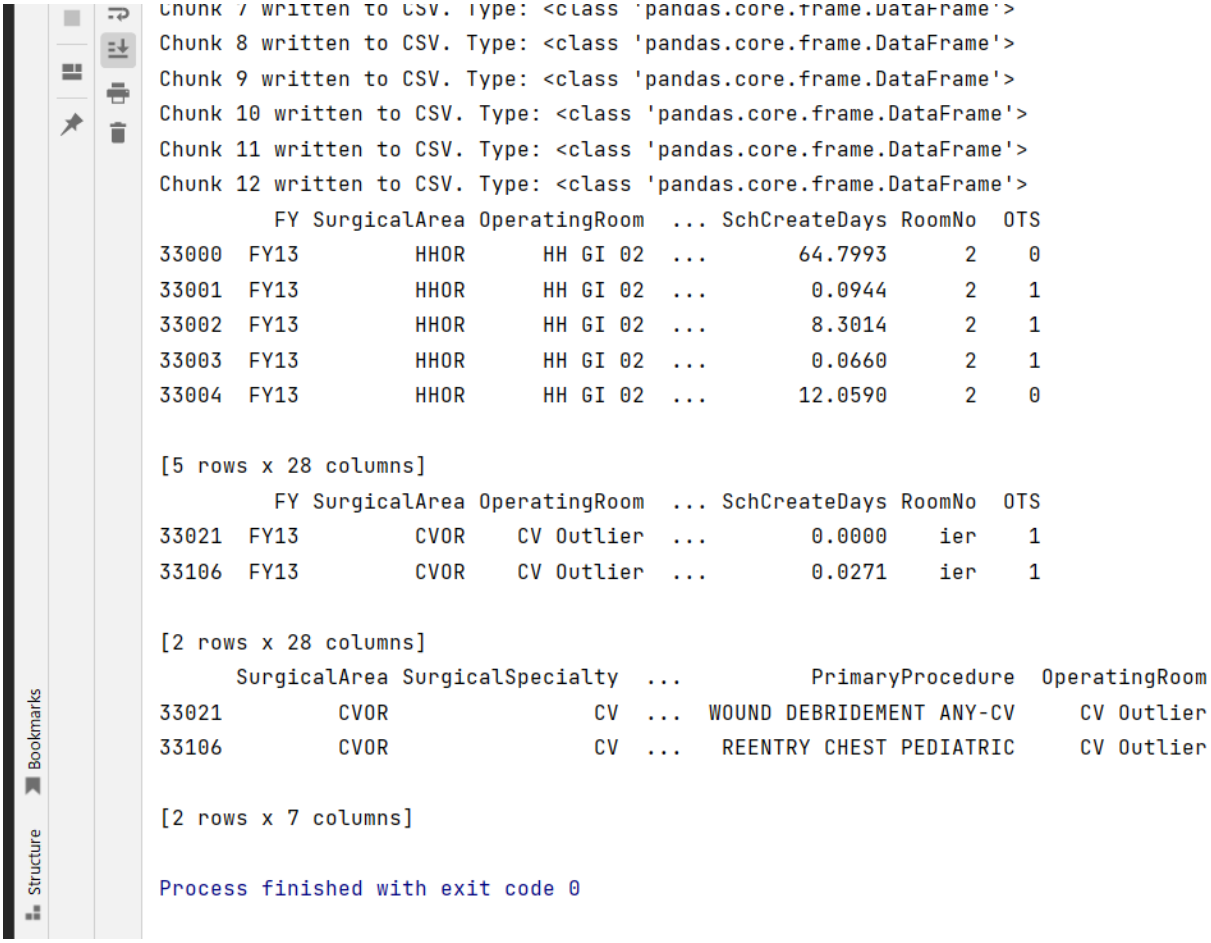
```python
# using a user defined function to execute a SQL script
def exec_sql():
    # print the SQL script to execute
    print(SQL_script)
    # assign cur to the connection.cursor() function
    cur = con.cursor()
    # 'for loop' to  receive SQL projection by row
    for row in cur.execute(SQL_script):
        # print row by row of projection
        print(row)


# establishes connection to SQLite database (open db)
con = sqlite3.connect("HMIS.db")
SQL_script = print(
    """
SELECT Application || " --- " || Status AS App_Stat_Rural_MA, COUNT(*)
FROM ( SELECT Name, Application, Status, City, State, LENGTH(Name) AS name_length, UPPER(State) AS state_upper
        FROM ( SELECT Name, Application, Status, City, State FROM LEADS WHERE State = "MA" )
        WHERE City NOT IN ("Boston", "Cambridge", "Lowell", "Springfield", "Worcester", "Newton")
        ORDER BY City, Name )
GROUP BY App_Stat_Rural_MA, name_length, state_upper
ORDER BY COUNT(*) DESC
LIMIT 10;
""" )
#execute user defined function exec_sql() for SQL script
exec_sql()
con.close()
```

```
('Electronic Data Interchange (EDI) - Clearing House Vendor --- Not Automated', 5)
('Enterprise EMR --- Live and Operational', 5)
('Enterprise Resource Planning --- Not Automated', 5)
('Enterprise Resource Planning --- Not Automated', 5)
('Enterprise Resource Planning --- Not Automated', 5)
('Executive Information System --- Live and Operational', 5)
('Computerized Practitioner Order Entry (CPOE) --- Not Automated', 4)
('Enterprise EMR --- Live and Operational', 4)
('Executive Information System --- Live and Operational', 4)
('Executive Information System --- Live and Operational', 4)

SELECT Application || " --- " || Status AS App_Stat_Rural_MA, COUNT(*)
FROM ( SELECT Name, Application, Status, City, State, LENGTH(Name) AS name_length, UPPER(State) AS state_upper
        FROM ( SELECT Name, Application, Status, City, State FROM LEADS WHERE State = "MA" )
        WHERE City NOT IN ("Boston", "Cambridge", "Lowell", "Springfield", "Worcester", "Newton")
        ORDER BY City, Name )
GROUP BY App_Stat_Rural_MA, name_length, state_upper
ORDER BY COUNT(*) DESC
LIMIT 10;

None
```

Question7

```python
import pandas as pd
chunk_size = 3000
batch_num = 1

# initialize CSV output file
csv_out_file = 'question7output.csv'

for chunk in pd.read_csv('SchData2013.csv', chunksize=chunk_size, iterator=True):

    # filter for SurgicalArea and SurgicalSpecialty
    filtered_chunk = chunk[(chunk['SurgicalArea'] == 'CVOR') | (chunk['SurgicalSpecialty'] == 'CV')]

    # slice and dice the filtered chunk
    sliced_chunk = filtered_chunk[['SurgicalArea', 'SurgicalSpecialty', 'SSSNo', 'PatientAge',
                                   'TotalSurgeryMin', 'PrimaryProcedure', 'OperatingRoom']]

    # write the sliced and diced chunk to CSV
    if batch_num == 1:
        sliced_chunk.to_csv(csv_out_file, header=True, index=False)
    else:
        sliced_chunk.to_csv(csv_out_file, mode='a', header=False, index=False)

    print(f'Chunk {batch_num} written to CSV. Type: {type(sliced_chunk)}')
    batch_num += 1

print(chunk.head())
print(filtered_chunk.head())
print(sliced_chunk.head())
```

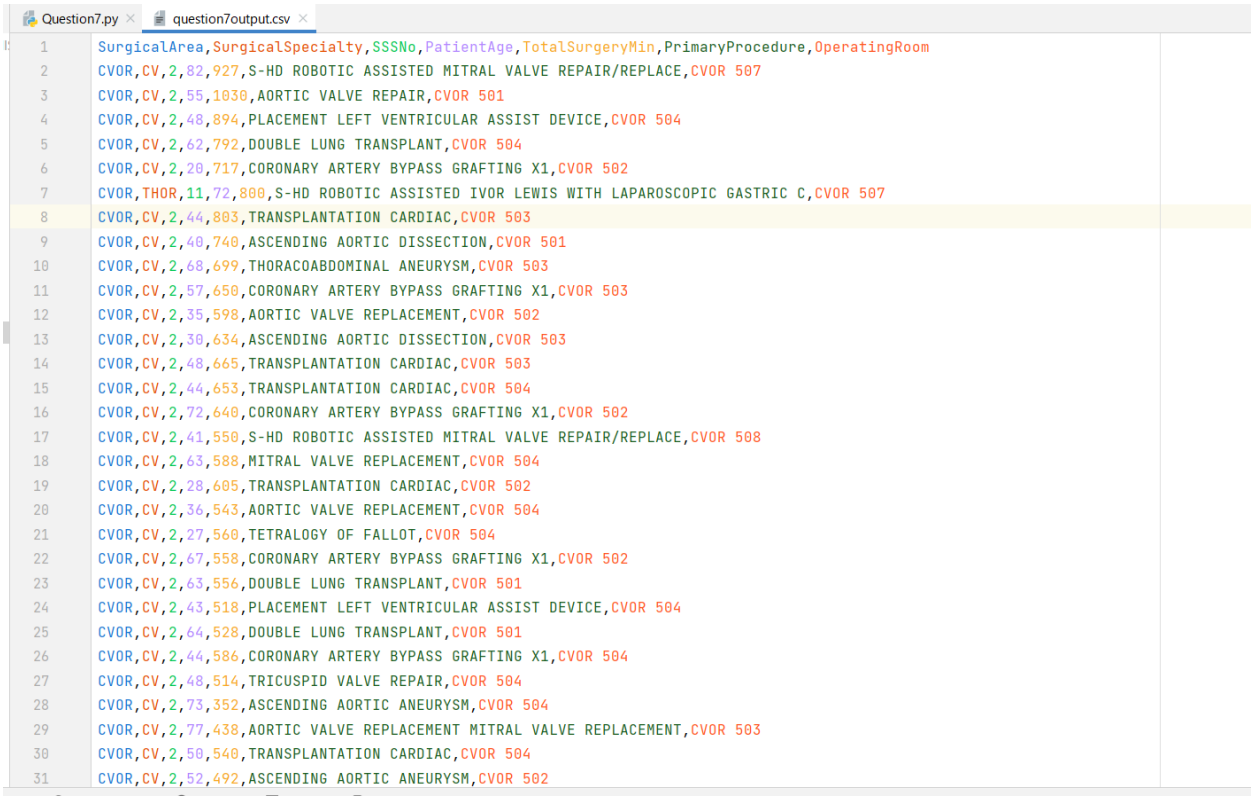
```
Chunk 1 written to CSV. Type: <class 'pandas.core.frame.DataFrame'>
Chunk 2 written to CSV. Type: <class 'pandas.core.frame.DataFrame'>
Chunk 3 written to CSV. Type: <class 'pandas.core.frame.DataFrame'>
Chunk 4 written to CSV. Type: <class 'pandas.core.frame.DataFrame'>
Chunk 5 written to CSV. Type: <class 'pandas.core.frame.DataFrame'>
Chunk 6 written to CSV. Type: <class 'pandas.core.frame.DataFrame'>
Chunk 7 written to CSV. Type: <class 'pandas.core.frame.DataFrame'>
Chunk 8 written to CSV. Type: <class 'pandas.core.frame.DataFrame'>
Chunk 9 written to CSV. Type: <class 'pandas.core.frame.DataFrame'>
Chunk 10 written to CSV. Type: <class 'pandas.core.frame.DataFrame'>
Chunk 11 written to CSV. Type: <class 'pandas.core.frame.DataFrame'>
Chunk 12 written to CSV. Type: <class 'pandas.core.frame.DataFrame'>
         FY SurgicalArea OperatingRoom  ... SchCreateDays RoomNo  OTS
33000  FY13         HHOR      HH GI 02  ...       64.7993      2    0
33001  FY13         HHOR      HH GI 02  ...        0.0944      2    1
33002  FY13         HHOR      HH GI 02  ...        8.3014      2    1
33003  FY13         HHOR      HH GI 02  ...        0.0660      2    1
33004  FY13         HHOR      HH GI 02  ...       12.0590      2    0

[5 rows x 28 columns]
         FY SurgicalArea OperatingRoom  ... SchCreateDays RoomNo  OTS
33021  FY13         CVOR    CV Outlier  ...        0.0000    ier    1
33106  FY13         CVOR    CV Outlier  ...        0.0271    ier    1

[2 rows x 28 columns]
      SurgicalArea SurgicalSpecialty  ...          PrimaryProcedure  OperatingRoom
33021         CVOR               CV  ...  WOUND DEBRIDEMENT ANY-CV     CV Outlier
33106         CVOR               CV  ...   REENTRY CHEST PEDIATRIC     CV Outlier
```



```
Chunk 7 written to CSV. Type: <class 'pandas.core.frame.DataFrame'>
Chunk 8 written to CSV. Type: <class 'pandas.core.frame.DataFrame'>
Chunk 9 written to CSV. Type: <class 'pandas.core.frame.DataFrame'>
Chunk 10 written to CSV. Type: <class 'pandas.core.frame.DataFrame'>
Chunk 11 written to CSV. Type: <class 'pandas.core.frame.DataFrame'>
Chunk 12 written to CSV. Type: <class 'pandas.core.frame.DataFrame'>
         FY SurgicalArea OperatingRoom  ... SchCreateDays RoomNo  OTS
33000  FY13         HHOR      HH GI 02  ...       64.7993      2    0
33001  FY13         HHOR      HH GI 02  ...        0.0944      2    1
33002  FY13         HHOR      HH GI 02  ...        8.3014      2    1
33003  FY13         HHOR      HH GI 02  ...        0.0660      2    1
33004  FY13         HHOR      HH GI 02  ...       12.0590      2    0

[5 rows x 28 columns]
         FY SurgicalArea OperatingRoom  ... SchCreateDays RoomNo  OTS
33021  FY13         CVOR    CV Outlier  ...        0.0000    ier    1
33106  FY13         CVOR    CV Outlier  ...        0.0271    ier    1

[2 rows x 28 columns]
      SurgicalArea SurgicalSpecialty  ...          PrimaryProcedure  OperatingRoom
33021         CVOR                CV  ...  WOUND DEBRIDEMENT ANY-CV     CV Outlier
33106         CVOR                CV  ...   REENTRY CHEST PEDIATRIC     CV Outlier

[2 rows x 7 columns]

Process finished with exit code 0
```

Output CSV

Question7.py | question7output.csv



```
SurgicalArea,SurgicalSpecialty,SSSNo,PatientAge,TotalSurgeryMin,PrimaryProcedure,OperatingRoom
CVOR,CV,2,82,927,S-HD ROBOTIC ASSISTED MITRAL VALVE REPAIR/REPLACE,CVOR 507
CVOR,CV,2,55,1030,AORTIC VALVE REPAIR,CVOR 501
CVOR,CV,2,48,894,PLACEMENT LEFT VENTRICULAR ASSIST DEVICE,CVOR 504
CVOR,CV,2,62,792,DOUBLE LUNG TRANSPLANT,CVOR 504
CVOR,CV,2,20,717,CORONARY ARTERY BYPASS GRAFTING X1,CVOR 502
CVOR,THOR,11,72,800,S-HD ROBOTIC ASSISTED IVOR LEWIS WITH LAPAROSCOPIC GASTRIC C,CVOR 507
CVOR,CV,2,44,803,TRANSPLANTATION CARDIAC,CVOR 503
CVOR,CV,2,40,740,ASCENDING AORTIC DISSECTION,CVOR 501
CVOR,CV,2,68,699,THORACOABDOMINAL ANEURYSM,CVOR 503
CVOR,CV,2,57,650,CORONARY ARTERY BYPASS GRAFTING X1,CVOR 503
CVOR,CV,2,35,598,AORTIC VALVE REPLACEMENT,CVOR 502
CVOR,CV,2,30,634,ASCENDING AORTIC DISSECTION,CVOR 503
CVOR,CV,2,48,665,TRANSPLANTATION CARDIAC,CVOR 503
CVOR,CV,2,44,653,TRANSPLANTATION CARDIAC,CVOR 504
CVOR,CV,2,72,640,CORONARY ARTERY BYPASS GRAFTING X1,CVOR 502
CVOR,CV,2,41,550,S-HD ROBOTIC ASSISTED MITRAL VALVE REPAIR/REPLACE,CVOR 508
CVOR,CV,2,63,588,MITRAL VALVE REPLACEMENT,CVOR 504
CVOR,CV,2,28,605,TRANSPLANTATION CARDIAC,CVOR 502
CVOR,CV,2,36,543,AORTIC VALVE REPLACEMENT,CVOR 504
CVOR,CV,2,27,560,TETRALOGY OF FALLOT,CVOR 504
CVOR,CV,2,67,558,CORONARY ARTERY BYPASS GRAFTING X1,CVOR 502
CVOR,CV,2,63,556,DOUBLE LUNG TRANSPLANT,CVOR 501
CVOR,CV,2,43,518,PLACEMENT LEFT VENTRICULAR ASSIST DEVICE,CVOR 504
CVOR,CV,2,64,528,DOUBLE LUNG TRANSPLANT,CVOR 501
CVOR,CV,2,44,586,CORONARY ARTERY BYPASS GRAFTING X1,CVOR 504
CVOR,CV,2,48,514,TRICUSPID VALVE REPAIR,CVOR 504
CVOR,CV,2,73,352,ASCENDING AORTIC ANEURYSM,CVOR 504
CVOR,CV,2,77,438,AORTIC VALVE REPLACEMENT MITRAL VALVE REPLACEMENT,CVOR 503
CVOR,CV,2,50,540,TRANSPLANTATION CARDIAC,CVOR 504
CVOR,CV,2,52,492,ASCENDING AORTIC ANEURYSM,CVOR 502
```

Question 8

Question8.py

```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
desired_width = 320
pd.set_option('display.width', desired_width)
pd.set_option('display.max_columns', 18)
df = pd.read_csv('SchData2013.csv', low_memory=False)
print(df.info())

# Identify int64 and float64 variables
int_vars = df.select_dtypes(include=['int64']).columns
float_vars = df.select_dtypes(include=['float64']).columns
print('Int Variables:', int_vars)
print('Float Variables:', float_vars)
df = df.drop('CaseMonth', axis=1)
df = df.dropna()
# Identify all numeric columns
numeric_cols = df.select_dtypes(include=['int64', 'float64']).columns
#AND
# Replace missing values with column mean
df[numeric_cols] = df[numeric_cols].fillna(df[numeric_cols].mean())
#verifying if there is no null values in any columns
print(df.isnull().sum())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 33173 entries, 0 to 33172
Data columns (total 28 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   FY                              33173 non-null  object
 1   SurgicalArea                    33173 non-null  object
 2   OperatingRoom                   33173 non-null  object
 3   ORCaseNumber                    33173 non-null  object
 4   SurgicalSpecialty               33173 non-null  object
 5   SSSNo                           33173 non-null  int64
 6   PatientType                     33173 non-null  int64
 7   SchedulePriority                33173 non-null  int64
 8   AddOnIndicator                  33173 non-null  object
 9   CaseDOW                         33173 non-null  object
 10  CaseMonth                       0 non-null      float64
 11  PrimaryProcedure                33173 non-null  object
 12  ScheduledStartDate_Time         33173 non-null  object
 13  CaseStartDate_Time              33173 non-null  object
 14  CaseCreateDate_Time             33173 non-null  object
 15  CaseLevel                       33173 non-null  int64
 16  ASA_Class                       33173 non-null  object
 17  PatientAge                      33173 non-null  int64
 18  PreopDiagnosis                  33173 non-null  object
 19  ScheduledCaseDuration           33173 non-null  int64
 20  PrimaryProcedureSurgeonComment  33173 non-null  object
 21  TotalSurgeryMin                 33173 non-null  int64
```

```
 27  OTS                           33173 non-null  int64
dtypes: float64(2), int64(11), object(15)
memory usage: 7.1+ MB
None
Int Variables: Index(['SSSNo', 'PatientType', 'SchedulePriority', 'CaseLevel', 'PatientAge', 'ScheduledCaseDuration', 'To
Float Variables: Index(['CaseMonth', 'SchCreateDays'], dtype='object')
FY                                0
SurgicalArea                      0
OperatingRoom                     0
ORCaseNumber                      0
SurgicalSpecialty                 0
SSSNo                             0
PatientType                       0
SchedulePriority                  0
AddOnIndicator                    0
CaseDOW                           0
PrimaryProcedure                  0
ScheduledStartDate_Time           0
CaseStartDate_Time                0
CaseCreateDate_Time               0
CaseLevel                         0
ASA_Class                         0
PatientAge                        0
PreopDiagnosis                    0
ScheduledCaseDuration             0
PrimaryProcedureSurgeonComment    0
```

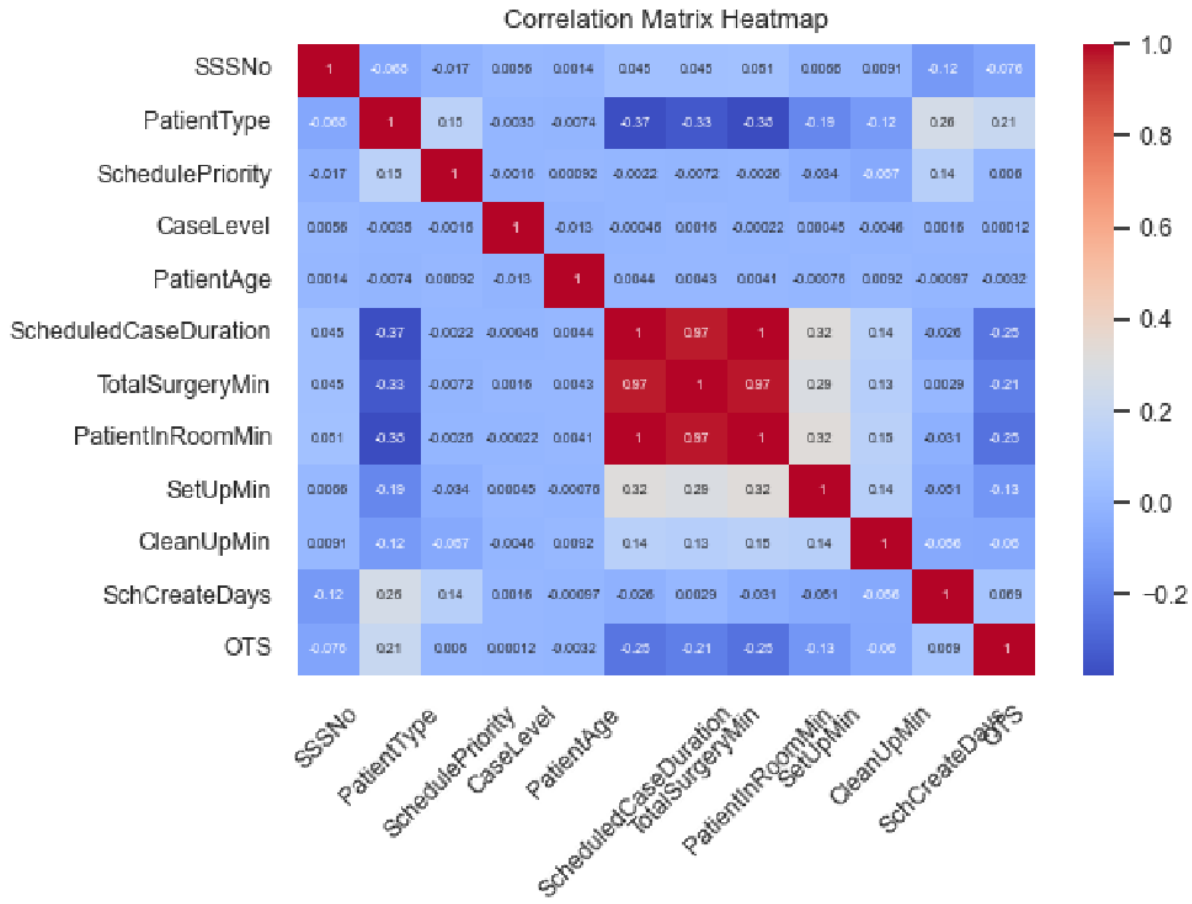
```
RoomNo                            0
OTS                               0
dtype: int64
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 33173 entries, 0 to 33172
Data columns (total 12 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   SSSNo                  33173 non-null  int64
 1   PatientType            33173 non-null  int64
 2   SchedulePriority       33173 non-null  int64
 3   CaseLevel              33173 non-null  int64
 4   PatientAge             33173 non-null  int64
 5   ScheduledCaseDuration  33173 non-null  int64
 6   TotalSurgeryMin        33173 non-null  int64
 7   PatientInRoomMin       33173 non-null  int64
 8   SetUpMin               33173 non-null  int64
 9   CleanUpMin             33173 non-null  int64
 10  SchCreateDays          33173 non-null  float64
 11  OTS                    33173 non-null  int64
dtypes: float64(1), int64(11)
memory usage: 3.0 MB
None
```

Bookmarks

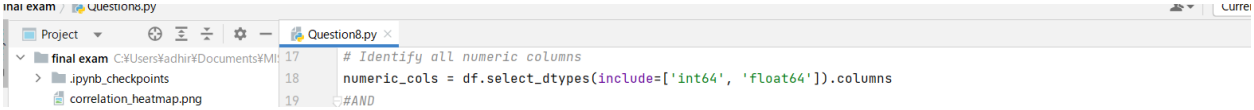
ture

```
df2 = df[numeric_cols]
print(df2.info())

corr_matrix = df2.corr()
print(corr_matrix)

# Create a heatmap using Seaborn
sns.set(font_scale=0.8)
sns.heatmap(corr_matrix, cmap="coolwarm", annot=True, annot_kws={"size": 5})
plt.title("Correlation Matrix Heatmap")
plt.xticks(rotation=45)
plt.yticks(rotation=0)
plt.tight_layout()

# Save the plot to disk
plt.savefig("correlation_heatmap.png")
```

```
None
                            SSSNo  PatientType  SchedulePriority  CaseLevel  PatientAge  ScheduledCaseDuration  TotalSurgeryMin  PatientInRoomMin  SetUpMin  CleanUpMin  SchCreat
SSSNo                    1.000000    -0.068476         -0.017188   0.005638    0.001375               0.045277         0.044706          0.051283  0.006564    0.009114      -0.1
PatientType             -0.068476     1.000000          0.154446  -0.003530   -0.007406              -0.369991        -0.333946         -0.376535 -0.186454   -0.117722       0.2
SchedulePriority        -0.017188     0.154446          1.000000  -0.001603    0.000917              -0.002212        -0.007232         -0.002601 -0.033835   -0.056809       0.1
CaseLevel                0.005638    -0.003530         -0.001603   1.000000   -0.012879              -0.000464         0.001624         -0.000224  0.000454   -0.004590       0.0
PatientAge               0.001375    -0.007406          0.000917  -0.012879    1.000000               0.004357         0.004287          0.004121 -0.000758    0.009183      -0.0
ScheduledCaseDuration    0.045277    -0.369991         -0.002212  -0.000464    0.004357               1.000000         0.972511          0.996709  0.318952    0.141771      -0.0
TotalSurgeryMin          0.044706    -0.333946         -0.007232   0.001624    0.004287               0.972511         1.000000          0.974048  0.294219    0.129837       0.0
PatientInRoomMin         0.051283    -0.376535         -0.002601  -0.000224    0.004121               0.996709         0.974048          1.000000  0.324737    0.145072      -0.0
SetUpMin                 0.006564    -0.186454         -0.033835   0.000454   -0.000758               0.318952         0.294219          0.324737  1.000000    0.135381      -0.0
CleanUpMin               0.009114    -0.117722         -0.056809  -0.004590    0.009183               0.141771         0.129837          0.145072  0.135381    1.000000      -0.0
SchCreateDays           -0.122489     0.259591          0.136325   0.001632   -0.000966              -0.026470         0.002930         -0.030770 -0.051267   -0.056372       1.0
OTS                     -0.075633     0.207286          0.005956   0.000121   -0.003237              -0.247022        -0.211273         -0.253700 -0.131578   -0.060009       0.0

Process finished with exit code 0
```



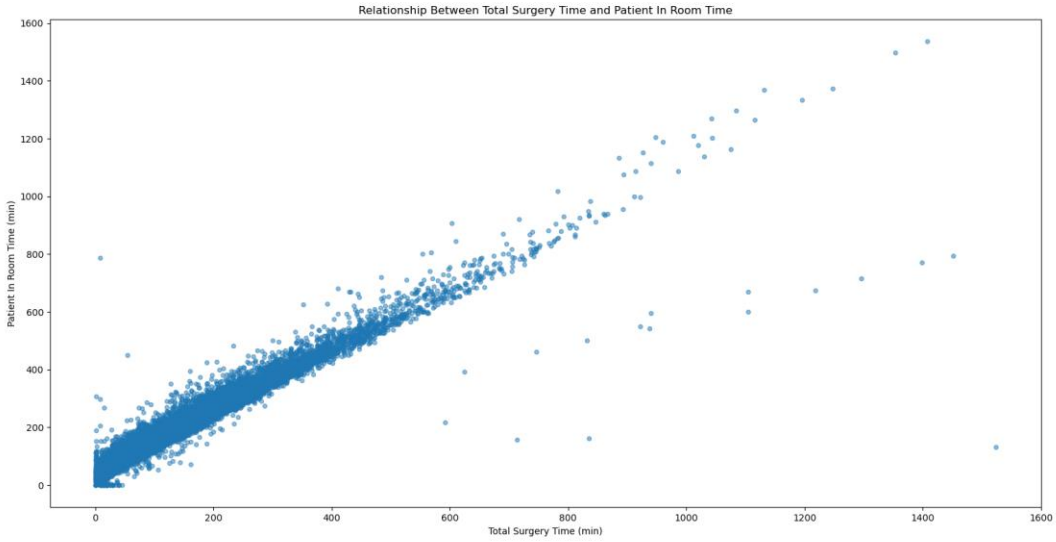
File created locally

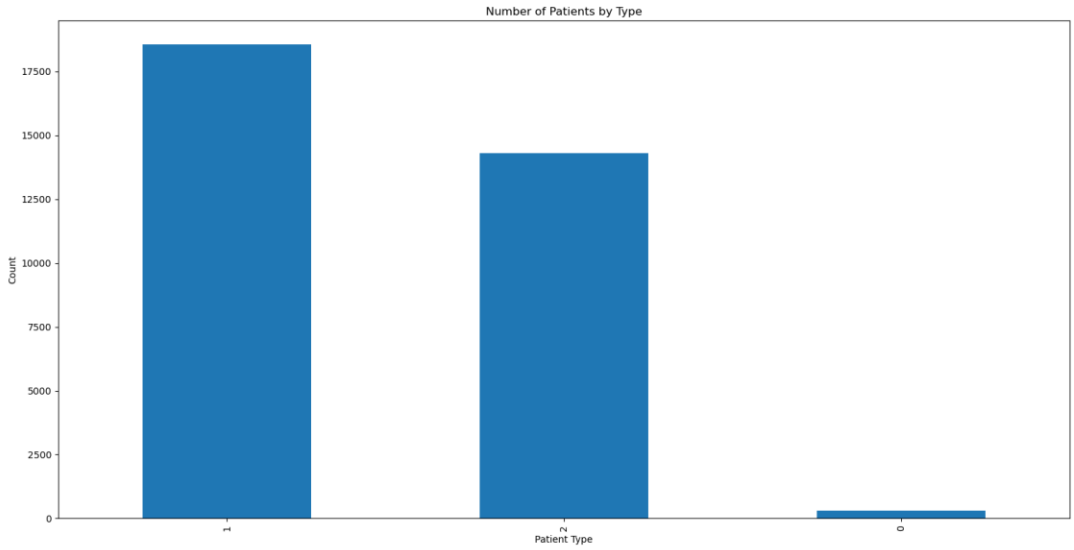


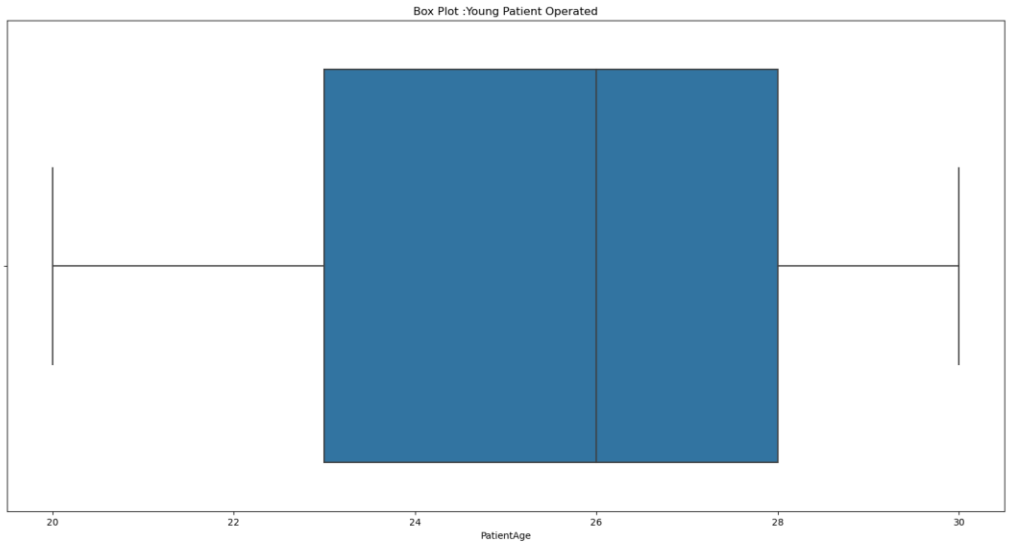
Question9

Question9.py

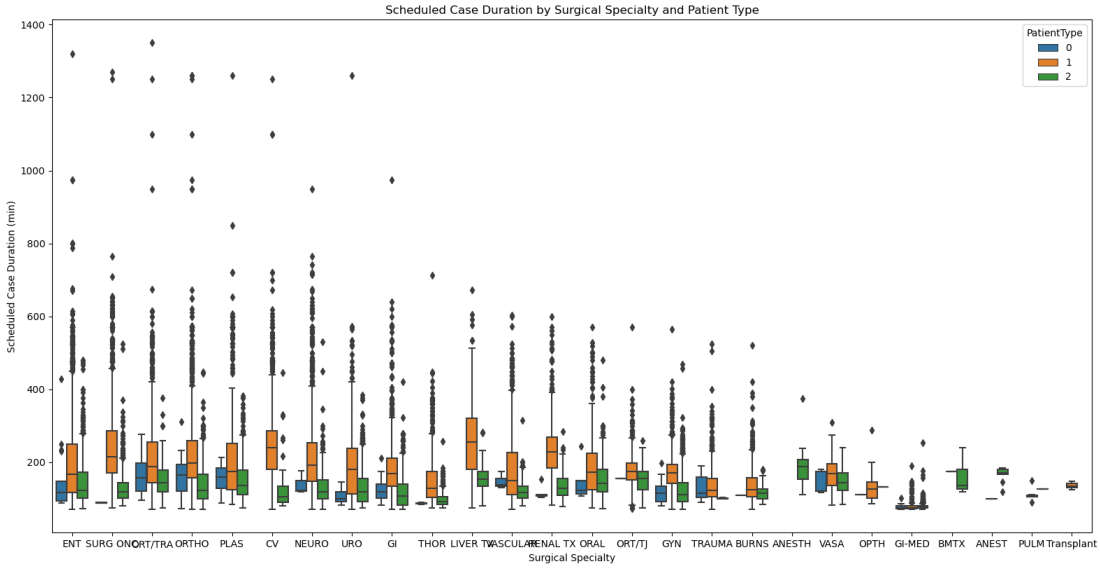
```
import seaborn as sns
import matplotlib.pyplot as plt
df = pd.read_csv('SchData2013.csv', low_memory=False)

# create a scatter plot of TotalSurgeryMin vs. PatientInRoomMin
df.plot(kind='scatter', x='TotalSurgeryMin', y='PatientInRoomMin', alpha=0.5)
plt.xlabel('Total Surgery Time (min)')
plt.ylabel('Patient In Room Time (min)')
plt.title('Relationship Between Total Surgery Time and Patient In Room Time')
plt.show()

# create a bar chart of PatientType counts
df['PatientType'].value_counts().plot(kind='bar')
plt.xlabel('Patient Type')
plt.ylabel('Count')
plt.title('Number of Patients by Type')
plt.show()

df2= df[(df.PatientAge >= 20) & (df.PatientAge <= 30)]
sns.boxplot(df2.PatientAge).set_title('Box Plot :Young Patient Operated')
plt.show()

# create a boxplot of ScheduledCaseDuration by SurgicalSpecialty and PatientType using Seaborn
sns.boxplot(x='SurgicalSpecialty', y='ScheduledCaseDuration', hue='PatientType', data=df)
plt.xlabel('Surgical Specialty')
plt.ylabel('Scheduled Case Duration (min)')
plt.title('Scheduled Case Duration by Surgical Specialty and Patient Type')
plt.show()
```



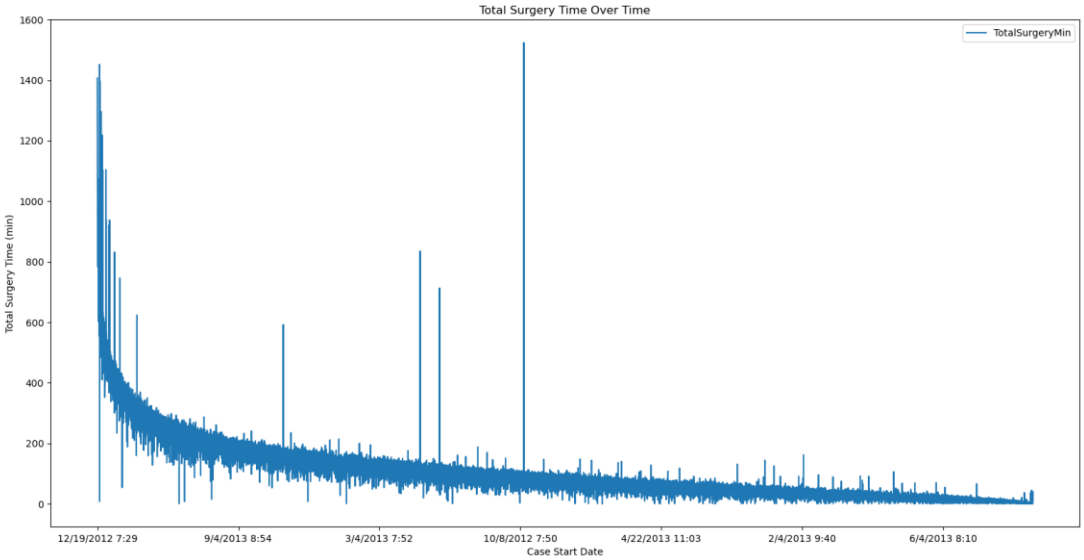


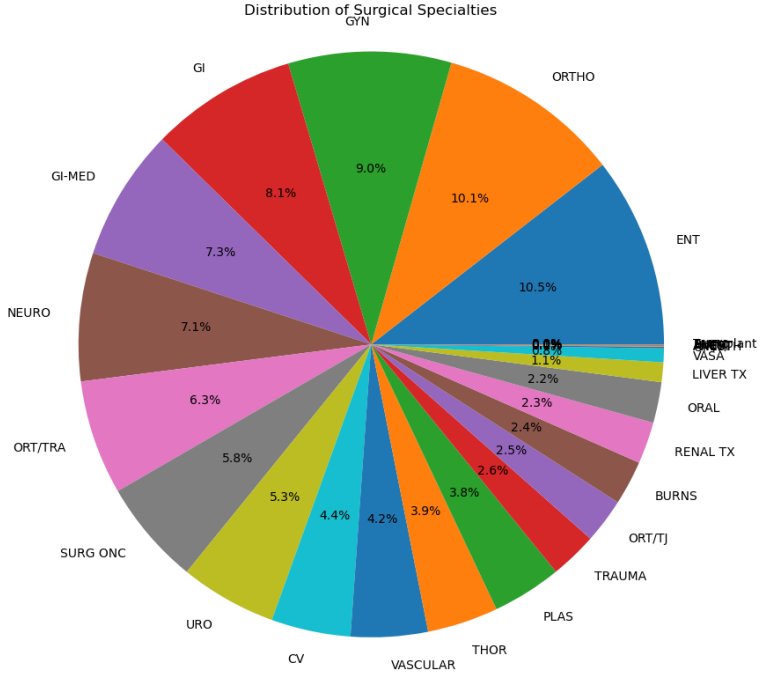


Zoom next image

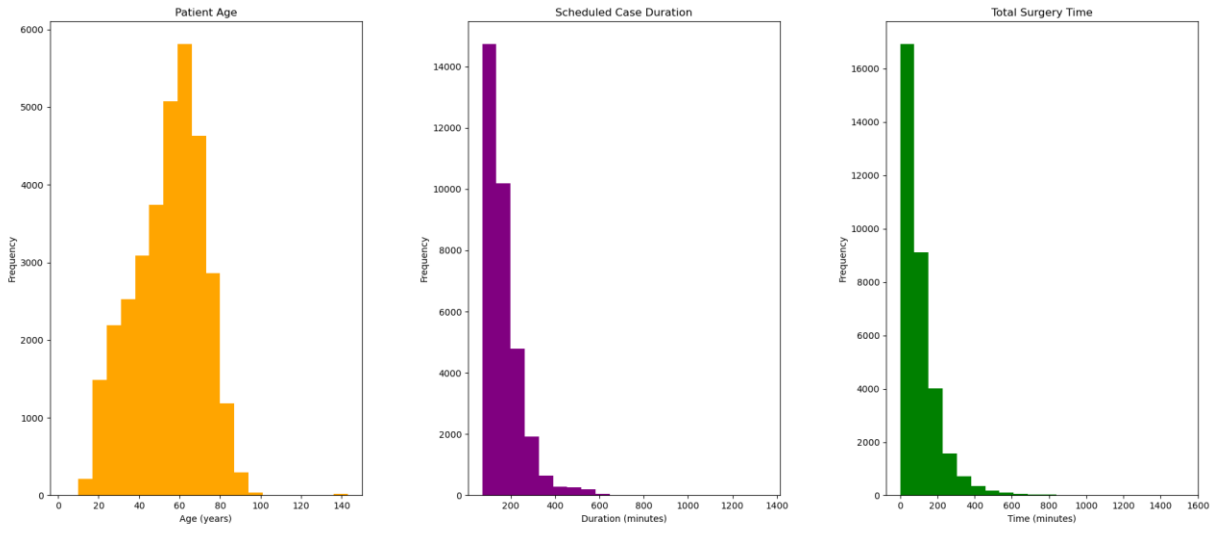



```
# create a line plot of TotalSurgeryMin over time
df.plot(kind='line', x='CaseStartDate_Time', y='TotalSurgeryMin')
plt.xlabel('Case Start Date')
plt.ylabel('Total Surgery Time (min)')
plt.title('Total Surgery Time Over Time')
plt.show()

# Create a series of the counts of each surgical specialty
specialty_counts = df['SurgicalSpecialty'].value_counts()

# Create a pie chart with labels
plt.pie(specialty_counts, labels=specialty_counts.index, autopct='%1.1f%%')
plt.title('Distribution of Surgical Specialties')
plt.axis('equal')
plt.show()
```


Total Surgery Time Over Time
TotalSurgeryMin
Total Surgery Time (min)
1600
1400
1200
1000
800
600
400
200
0
12/19/2012 7:29
9/4/2013 8:54
3/4/2013 7:52
10/8/2012 7:50
4/22/2013 11:03
2/4/2013 9:40
6/4/2013 8:10
Case Start Date


Distribution of Surgical Specialties
GYN
ORTHO
ENT
LIVER TX
ORAL
RENAL TX
BURNS
ORT/TJ
TRAUMA
PLAS
THOR
VASCULAR
CV
URO
SURG ONC
ORT/TRA
NEURO
GI-MED
GI
9.0%
10.1%
10.5%
1.1%
2.2%
2.3%
2.4%
2.5%
2.6%
3.8%
3.9%
4.2%
4.4%
5.3%
5.8%
6.3%
7.1%
7.3%
8.1%

```python
# Create a figure with 3 subplots
fig, axs = plt.subplots(1, 3, figsize=(12, 4))

# Plot the histograms for each column
axs[0].hist(df["PatientAge"], bins=20, color="orange")
axs[0].set_title("Patient Age")
axs[0].set_xlabel("Age (years)")
axs[0].set_ylabel("Frequency")

axs[1].hist(df["ScheduledCaseDuration"], bins=20, color="purple")
axs[1].set_title("Scheduled Case Duration")
axs[1].set_xlabel("Duration (minutes)")
axs[1].set_ylabel("Frequency")

axs[2].hist(df["TotalSurgeryMin"], bins=20, color="green")
axs[2].set_title("Total Surgery Time")
axs[2].set_xlabel("Time (minutes)")
axs[2].set_ylabel("Frequency")

# Adjust the layout and spacing between subplots
plt.tight_layout()

# Display the plot
plt.show()
```



Question 10

```
import nltk
import pandas as pd
import string
from sklearn.feature_extraction. text import TfidfVectorizer, CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix

nltk.download('stopwords')
nltk.download('punkt')
desired_width = 400
pd.set_option('display.width', desired_width)
pd.set_option('display.max_columns', 20)
stopwords_set = set (nltk.corpus.stopwords.words ('english'))
punctuation_set = set(string.punctuation)
print('# stopwords =', len(stopwords_set),'  # punctuations =', len(punctuation_set))
print(' ')
data_file = 'Procedure3.txt' # assigns var data_file
df = pd.read_table (data_file, header=None)
df.columns= ('surgicalArea', 'procedure')
df['proc_cleaned'] =df.procedure.apply(lambda x: ' '.join([word for word in x.split() if word not in stopwords_set and word not
                   in punctuation_set]))
df['proc_cleaned'] = df.proc_cleaned.str.lower()
print(df.head(4))
print(' ')
```

```
TFIDF_vector = TfidfVectorizer()
TFIDF_vector1 = TfidfVectorizer (ngram_range=(1,3))
X = TFIDF_vector.fit_transform (df.proc_cleaned)
X1 = TFIDF_vector1.fit_transform(df.proc_cleaned)
print('TFIDF vector X matrix type = ', type (X), 'X matrix shape = ', X. shape)
print('TFIDF* vector X1 matrix type', type (X1), 'X1 matrix shape = ', X1. shape)

y= df.surgicalArea
X_train, X_test, y_train, y_test = train_test_split(X,y)
X1_train, X1_test, y1_train, y1_test = train_test_split(X1,y)
print(' ')

Logreg = LogisticRegression (max_iter=1000)
Logreg1 = LogisticRegression (max_iter=1000)
Logreg2 = LogisticRegression (max_iter=1000)

Logreg.fit(X_train,y_train)
Logreg1.fit(X1_train,y1_train)


y_pred = Logreg.predict(X_test)
y1_pred = Logreg1.predict(X1_test)
LogregXR2 = Logreg.score (X_test,y_test)
LogregX1R2 = Logreg1.score (X1_test,y1_test)
print('Logistic Regression X R2 ',LogregXR2)
print('Logistic Regression X1 R2', LogregX1R2)
type_err_matrix = confusion_matrix(y_test, y_pred)
type_err_matrix1 = confusion_matrix(y1_test,y1_pred)
```

```
type_err_matrix1 = confusion_matrix(y1_test,y1_pred)
print(type_err_matrix, 'Confusion Matrix [y_test, y_pred]')
print(type_err_matrix1,'Confusion Matrix [y1_test, y1_pred]')

rf = RandomForestClassifier()
rf1 = RandomForestClassifier()
rf2 = RandomForestClassifier()

rf.fit(X_train,y_train)
rf1.fit(X1_train,y1_train)
y2_pred = rf.predict(X_test)
y3_pred = rf1.predict(X1_test)
print('rfscore', rf.score(X_test,y_test))
print('r1fscore', rf1.score(X1_test,y1_test))

type_err_matrix3 = confusion_matrix(y_test,y2_pred)                                # c
type_err_matrix4 = confusion_matrix(y1_test,y3_pred)                               # c
print(type_err_matrix3, ' <---------------- Confusion Matrix [y_test,y2_pred]')    # 
print(type_err_matrix4, ' <---------------- Confusion Matrix [y1_test,y3_pred]')   # 

print('Logistic reg. w/ TFIDF',confusion_matrix(y_test,y_pred))
print('Random forest w/ TFIDF',confusion_matrix(y_test,y2_pred))
print('Logistic reg. w/ TFIDF',confusion_matrix(y1_test,y1_pred))
print('Random forest w/ TFIDF',confusion_matrix(y1_test,y3_pred))
```

o/p

```
# stopwords = 179   # punctuations = 32

[nltk_data]   Package punkt is already up-to-date!
  surgicalArea             procedure          proc_cleaned
0        GENOR  ANKLE BRACHIAL INDEX  ankle brachial index
1        GENOR  ANKLE BRACHIAL INDEX  ankle brachial index
2        GENOR  ANKLE BRACHIAL INDEX  ankle brachial index
3        GENOR  ANKLE BRACHIAL INDEX  ankle brachial index

TFIDF vector X matrix type =  <class 'scipy.sparse._csr.csr_matrix'> X matrix shape =  (38641, 1176)
TFIDF* vector X1 matrix type <class 'scipy.sparse._csr.csr_matrix'> X1 matrix shape =  (38641, 5541)

Logistic Regression X R2  0.9003208777559258
Logistic Regression X1 R2 0.9061173791532967
```

```
[[6519  318]
 [ 645 2179]] Confusion Matrix [y_test, y_pred]
[[6535  335]
 [ 572 2219]] Confusion Matrix [y1_test, y1_pred]
rfscore 0.9058068522927233
r1fscore 0.9096366835731291
[[6484  353]
 [ 557 2267]]  <--------------- Confusion Matrix [y_test,y2_pred]
[[6492  378]
 [ 495 2296]]  <--------------- Confusion Matrix [y1_test,y3_pred]
Logistic reg. w/ TFIDF [[6519  318]
 [ 645 2179]]
Random forest w/ TFIDF [[6484  353]
 [ 557 2267]]
Logistic reg. w/ TFIDF [[6535  335]
 [ 572 2219]]
Random forest w/ TFIDF [[6492  378]
 [ 495 2296]]

Process finished with exit code 0
```