Part_4: Final Analysis

Farhana Islam

The ER department has provided you with a data extract for all the patients who received Laparoscopy Appendectomy & Laparoscopy Cholecystectomy's in the time period as our other datasets. Use this information to answer the following questions.

```
import pandas as pd
from datetime import datetime
df_b = pd.read_csv("ORBooking.csv")
df_P = pd.read_csv("ER - Patient Log.csv")
```

```
# Convert timestamps to datetime
df_P["Patient Admitting - Check In"] = pd.to_datetime(df_P["Patient Admitting -
Check In"])
df_P["Patient Triagne Nurse Visit"] = pd.to_datetime(df_P["Patient Triagne Nurse
Visit"])
df_P["Patient Admit to ER"] = pd.to_datetime(df_P["Patient Admit to ER"])
# Merge the datasets based on the HCID column
merged_df = pd.merge(df_b, df_P, on="HCID", how="left")
merged_df
```

Do you have enough information to check whether if the following targets have been met? (1 Mark)
-Patients checked into the emergency department in 10 minutes.
-Patients seen by a triage nurse within 20 minutes.
-Patients admitted to the ER in 60 minutes.
For the above targets where there is enough information:

```
# Calculate time differences for each target
merged_df["Time Difference Check-in"] = (merged_df["Patient Admitting - Check
In"] - merged_df["Patient Admit to ER"]).dt.total_seconds() / 60
merged_df["Time Difference Triage"] = (merged_df["Patient Triagne Nurse Visit"] -
merged_df["Patient Admitting - Check In"]).dt.total_seconds() / 60
merged_df["Time Difference Admit to ER"] = (merged_df["Patient Admit to ER"] -
merged_df["Patient Admitting - Check In"]).dt.total_seconds() / 60

# Filter data for Laparoscopy Appendectomy and Laparoscopy Cholecystectomy
procedures
appendectomy_df = merged_df[merged_df['Proc Descr Mod'] == 'Laparoscopy
Appendectomy']
cholecystectomy_df = merged_df[merged_df['Proc Descr Mod'] == 'Laparoscopy
Cholecystectomy']

# Check if any patient exceeded the specified time limits
```

```
exceeded_checkin_limit = (merged_df["Time Difference Check-in"] > 10).any()
exceeded_triage_limit = (merged_df["Time Difference Triage"] > 20).any()
exceeded_admit_to_er_limit = (merged_df["Time Difference Admit to ER"] >
60).any()
# Print the results
print("Target: Patients checked into the emergency department in 10 minutes.")
if exceeded_checkin_limit:
    print("Some patients exceeded the 10-minute target for check-in at the
emergency department.")
else:
    print("All patients checked into the emergency department within 10
minutes.")

print("\nTarget: Patients seen by a triage nurse within 20 minutes.")
if exceeded_triage_limit:
    print("Some patients exceeded the 20-minute target for seeing a triage
nurse.")
else:
    print("All patients seen by a triage nurse within 20 minutes.")

print("\nTarget: Patients admitted to the ER in 60 minutes.")
if exceeded_admit_to_er_limit:
    print("Some patients exceeded the 60-minute target for admission to the ER.")
else:
    print("All patients admitted to the ER within 60 minutes.")
```

Calculate if current targets are being met. (1 Mark)

```
# Check if any patient exceeded the specified time limits
exceeded_checkin_limit = (merged_df["Time Difference Check-in"] > 10).sum()
exceeded_triage_limit = (merged_df["Time Difference Triage"] > 20).sum()
exceeded_admit_to_er_limit = (merged_df["Time Difference Admit to ER"] >
60).sum()
```

How many patients are in/ out of target? (1 Mark)

```
# Calculate how many patients are in/out of target
in_target_checkin = len(merged_df) - exceeded_checkin_limit
out_of_target_checkin = exceeded_checkin_limit

in_target_triage = len(merged_df) - exceeded_triage_limit
out_of_target_triage = exceeded_triage_limit

in_target_admit_to_er = len(merged_df) - exceeded_admit_to_er_limit
```

```
out_of_target_admit_to_er = exceeded_admit_to_er_limit
```

```
# Print results
print("Patients checked into the emergency department in 10 minutes:")
print("In target:", in_target_checkin)
print("Out of target:", out_of_target_checkin)

print("\nPatients seen by a triage nurse within 20 minutes:")
print("In target:", in_target_triage)
print("Out of target:", out_of_target_triage)

print("\nPatients admitted to the ER in 60 minutes:")
print("In target:", in_target_admit_to_er)
print("Out of target:", out_of_target_admit_to_er)
```

Merge the ER Dataset with the DI & OR Datasets to and answer the below questions:

```
import pandas as pd

df_b = pd.read_csv("ORBooking.csv")
df_P = pd.read_csv("ER - Patient Log.csv")
df_DI = pd.read_csv("combined_DI_data.csv")

# Merge the datasets based on the HCID column
merged_df = pd.merge(df_b, df_P, on="HCID", how="left")
merged_df = pd.merge(merged_df, df_DI, on="HCID", how="left")

# Display the merged dataframe
merged_df.head(5)
```

What is the average time for the patient journey from the time they are checked in at admitting to when they have surgery? (2 Marks)

```
# Assuming surgery start time is the OR Booking Req DT/Tm in the ORBooking
dataset
merged_df["OR Booking Req DT/Tm"] = pd.to_datetime(merged_df["OR Booking Req
DT/Tm"])
merged_df["Patient Admitting - Check In"] = pd.to_datetime(merged_df["Patient
Admitting - Check In"])
merged_df["Time from Admitting to Surgery"] = (merged_df["OR Booking Req DT/Tm"]
- merged_df["Patient Admitting - Check In"]).dt.total_seconds() / 60
average_time = merged_df["Time from Admitting to Surgery"].mean()
average_time_formatted = "{:.2f}".format(average_time)
```

```
print("Average time for Patient Journey from Admitting to Surgery:",
average_time_formatted, "minutes")
```

Where is the longest wait between steps in the process? (2 Marks)

```
# Convert timestamps to datetime
merged_df["Patient Admitting - Check In"] = pd.to_datetime(merged_df["Patient
Admitting - Check In"])
merged_df["Patient Triagne Nurse Visit"] = pd.to_datetime(merged_df["Patient
Triagne Nurse Visit"])
merged_df["Patient Admit to ER"] = pd.to_datetime(merged_df["Patient Admit to
ER"])
```

```
# Calculate time differences between consecutive steps
wait_times = merged_df[["Patient Admitting - Check In", "Patient Triagne Nurse
Visit", "Patient Admit to ER"]]
wait_times_diff = wait_times.diff(axis=1).drop(columns=["Patient Admitting -
Check In"])
wait_times_diff = wait_times_diff.apply(lambda x: x.dt.total_seconds() / 60)  #
Convert to minutes
longest_wait_step = wait_times_diff.max().idxmax()
print("Longest wait between steps in the process:", longest_wait_step)
```

Relative to all known targets for timeliness (the above and OR Booking Status) which parts of the process have the highest percentage of patients missing their target? (2 Marks)

```
import datetime

# Define target times for each step in minutes
target_times = {
    "Check-in at admitting": datetime.timedelta(minutes=10),
    "Seen by triage nurse": datetime.timedelta(minutes=20),
    "Admitted to ER": datetime.timedelta(minutes=60)
}
# Calculate actual time taken for each step
actual_times = {
    "Check-in at admitting": merged_df["Patient Triagne Nurse Visit"] -
merged_df["Patient Admitting - Check In"],
    "Seen by triage nurse": merged_df["Patient Admit to ER"] - merged_df["Patient
Triagne Nurse Visit"],
    "Admitted to ER": merged_df["Patient Admit to ER"] - merged_df["Patient
Admitting - Check In"]
}
```

```python
# Calculate percentage of patients missing targets for each step
percentage_missing = {}
for step, target_time in target_times.items():
    percentage_missing[step] = (actual_times[step] > target_time).mean() * 100

# Display the percentage of patients missing targets for each step
print("Percentage of patients missing targets for each step:")
for step, percentage in percentage_missing.items():
    print(f"{step}: {percentage:.2f}%")
```

For both types of surgery, does visit to Diagnostic Imaging add a significant amount of time to the overall process? (2 Marks)

```python
import pandas as pd

df_DI = pd.read_csv("combined_DI_data.csv")

# Filter the dataset for patients who underwent surgery
surgery_patients = merged_df["HCID"].unique()

# Filter DI dataset for surgery patients
df_DI_surgery = df_DI[df_DI["HCID"].isin(surgery_patients)]

# Convert timestamps to datetime
df_DI_surgery = df_DI_surgery.copy()
df_DI_surgery["DI Req - Time"] = pd.to_datetime(df_DI_surgery["DI Req - Time"])
df_DI_surgery["DI - Pt in Suite"] = pd.to_datetime(df_DI_surgery["DI - Pt in
Suite"])
```

```python
# Calculate the time taken for DI visit
df_DI_surgery["DI Visit Time"] = (df_DI_surgery["DI - Pt in Suite"] -
df_DI_surgery["DI Req - Time"]).dt.total_seconds() / 60
df_DI_surgery
```

```python
# Calculate the average time taken for DI visit for both types of surgery
avg_di_visit_time = df_DI_surgery.groupby("Req Type - Abdominal")["DI Visit
Time"].mean()

print("Average time taken for DI visit for each type of surgery in minutes:")
print(avg_di_visit_time)
```

```
# Assuming the DI visit time is available in the DI dataset
merged_df["Patient Triagne Nurse Visit"] = pd.to_datetime(merged_df["Patient
Triagne Nurse Visit"])
merged_df["DI - Pt in Suite"] = pd.to_datetime(merged_df["DI - Pt in Suite"])
di_time_effect = (merged_df["DI - Pt in Suite"] - merged_df["Patient Triagne
Nurse Visit"]).dt.total_seconds() / 60
overall_time = merged_df["Time from Admitting to Surgery"]
di_time_effect_formatted = "{:.2f}".format(di_time_effect.mean())
print("Average additional time due to Diagnostic Imaging visit:",
di_time_effect_formatted, "minutes")
```

Using other techniques to aggregate the data, what insights can you gain (2-3 points max). (3 Marks)

Statistical Test

```
from scipy.stats import ttest_ind

# t-test to compare time from admitting to surgery between two surgical types
surgical_type_A = merged_df[merged_df["Proc Descr Mod"] == "Laparoscopy
Appendectomy"]["Time from Admitting to Surgery"]
surgical_type_B = merged_df[merged_df["Proc Descr Mod"] == "Laparoscopy
Cholecystectomy"]["Time from Admitting to Surgery"]

t_statistic, p_value = ttest_ind(surgical_type_A, surgical_type_B)
print("T-test result:")
print("T-statistic:", t_statistic)
print("P-value:", p_value)

# Compare p-value to significance level (α)
alpha = 0.05
if p_value < alpha:
    print("The difference in time from admitting to surgery between the two
surgical types is statistically significant.")
else:
    print("There is no statistically significant difference in time from
admitting to surgery between the two surgical types.")
```

```
# Box plot of time from admitting to surgery by surgical type
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(8, 6))
```

```python
sns.boxplot(x="Proc Descr Mod", y="Time from Admitting to Surgery",
data=merged_df)
plt.xlabel("Surgical Type")
plt.ylabel("Time from Admitting to Surgery (minutes)")
plt.title("Box Plot of Time from Admitting to Surgery by Surgical Type")
plt.show()
plt.figure(figsize=(10, 6))
sns.boxplot(x="Requesting Physician", y="Time from Admitting to Surgery",
data=merged_df)
plt.xlabel("Operating Surgeon")
plt.ylabel("Time from Admitting to Surgery (minutes)")
plt.title("Box Plot of Time from Admitting to Surgery by Surgeon")
plt.xticks(rotation=45)
plt.show()
```

In one paragraph, reflect on the how the process map exercise informed your work on these time series assignments. (1 Mark)

Creating a process map gave me a clear picture of the steps in the ED process and how they relate to each other. This understanding helped me pinpoint where delays or waits might occur. By visualizing the relationships between discrete steps, it became easier to analyze the data for the time series assignments and select relevant variables. This guided a focused investigation into factors influencing process duration, contributing to more insightful analysis and interpret the results more effectively.