import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
  
events = pd.read\_csv(r"C:\Users\sarah\DATASCIENCE\events.csv")  
sessions = pd.read\_csv(r"C:\Users\sarah\DATASCIENCE\sessions.csv")  
applicants = pd.read\_csv(r"C:\Users\sarah\DATASCIENCE\applicants.csv")  
  
# events.shape  
# sessions.shape  
# applicants.shape  
  
print("\nShape of events tab with duplicate:")  
print(events.shape)  
  
# drop duplicate rows  
events.drop\_duplicates(keep='first'**,** inplace=True)  
sessions.drop\_duplicates(keep='first'**,** inplace=True)  
applicants.drop\_duplicates(keep='first'**,** inplace=True)  
  
# events.shape  
# sessions.shape  
# applicants.shape  
  
print("\nShape of events tab without duplicate:")  
print(events.shape)  
  
events['event\_datetime'].dtype  
  
frames = [events**,** applicants**,** sessions]  
  
# join tabs with 'session\_id' and 'applicant\_id' as common key  
data = pd.merge(events**,**sessions**,** on=['session\_id'**,** 'applicant\_id']**,** how='left')  
data = pd.merge(data**,**applicants**,** on=['session\_id'**,** 'applicant\_id']**,** how='left')  
  
# Remove useless columns  
data = data.drop(["first\_name"**,** 'last\_name']**,** axis=**1**)  
# create new dataframe with event\_type: "submitted test results" or "end\_of\_underwriting" not made by 'Recruiter  
  
submission\_df = data[(data["event\_type"] == "submitted test results") |  
 ((data["event\_type"] == "end\_of\_underwriting") & (data["event\_user"] != Recruiter))]  
  
  
submission\_df.reset\_index(inplace=True)  
  
# check and fix data types  
# submission\_df.dtypes  
  
submission = submission\_df.copy()  
  
submission.loc[:**,** 'event\_datetime'] = pd.to\_datetime(submission['event\_datetime'])  
submission.loc[:**,** 'birth\_date'] = pd.to\_datetime(submission['birth\_date'])  
  
submission = submission.sort\_values(by=['session\_id'**,** 'event\_datetime'])  
submission.reset\_index(inplace=True)  
  
  
# Vectors of submission time before and after the modification in underwiting application on March 15, 2259:  
  
submission\_time\_before = []  
submission\_time\_after = []  
  
modification\_date = pd.Timestamp('2259-03-15'**,** tz='UTC')  
  
submission['delta\_time'] = np.nan  
submission['Before\_After'] = np.nan  
submission['event\_datetime'].dtype  
  
for i in range(len(submission) - **1**):  
 if (submission.iloc[i + **1**]['event\_type'] == “submitted test results' and submission.iloc[i]['event\_type'] == 'end\_of\_underwriting' and  
 submission.iloc[i + **1**]['session\_id'] == submission.iloc[i]['session\_id']):  
 delta\_time = submission.iloc[i + **1**]['event\_datetime'] - submission.iloc[i]['event\_datetime']  
 if delta\_time.seconds > **0** and delta\_time.seconds < **7200**:  
 submission['delta\_time'][i] = delta\_time  
  
 # before modification  
 if submission.iloc[i]['event\_datetime'] < modification\_date:  
 Before\_After='Before'  
 submission\_time\_before.append(delta\_time)  
  
 # after modification  
 if submission.iloc[i]['event\_datetime'] >= modification\_date:  
 Before\_After = 'After'  
 submission\_time\_after.append(delta\_time)  
 submission['Before\_After'][i] = Before\_After  
  
# delete unrelevant rows:  
submission = submission.dropna(subset=['Before\_After'])  
  
# reset the index after dropping rows  
submission.reset\_index(drop=True**,** inplace=True)  
  
  
  
# Let's remove outliers to get a precise result.  
# We can easily see that the process mostly take minutes, so values bigger than 2 hours will be removed. It will also drop negative results.  
  
submission\_time\_before = [delta for delta in submission\_time\_before if delta.seconds < **7200**]  
submission\_time\_after = [delta for delta in submission\_time\_after if delta.seconds < **7200**]  
  
# Calculate the average  
average\_time\_before = sum(submission\_time\_before**,** pd.Timedelta(**0**)) / len(submission\_time\_before)  
average\_time\_after = sum(submission\_time\_after**,** pd.Timedelta(**0**)) / len(submission\_time\_after)  
  
# Calculate the median  
median\_time\_before = pd.Series(submission\_time\_before).median()  
median\_time\_after = pd.Series(submission\_time\_after).median()  
  
  
# Print the results  
  
# print("Average Time before modification:", average\_time\_before)  
# print("Average Time after modification:", average\_time\_after)  
# print("Median Time before modification:", median\_time\_before)  
# print("Median Time after modification:", median\_time\_after)  
  
  
  
# in a more readable format (minutes:secondes):  
# Convert average and median times to total seconds  
average\_sec\_before = int(average\_time\_before.total\_seconds())  
median\_sec\_before = int(median\_time\_before.total\_seconds())  
average\_sec\_after = int(average\_time\_after.total\_seconds())  
median\_sec\_after = int(median\_time\_after.total\_seconds())  
  
# Format average and median times  
formatted\_avg\_before = f"{(average\_sec\_before % **3600**) // **60**:02d}:{average\_sec\_before % **60**:02d}"  
formatted\_avg\_after = f"{(average\_sec\_after % **3600**) // **60**:02d}:{average\_sec\_after % **60**:02d}"  
formatted\_median\_before = f"{(median\_sec\_before % **3600**) // **60**:02d}:{median\_sec\_before % **60**:02d}"  
formatted\_median\_after = f"{(median\_sec\_after % **3600**) // **60**:02d}:{median\_sec\_after % **60**:02d}"  
  
# Print the results  
print("Average Time before modification:"**,** formatted\_avg\_before)  
print("Average Time after modification:"**,** formatted\_avg\_after)  
print()  
print("Median Time before modification:"**,** formatted\_median\_before)  
print("Median Time after modification:"**,** formatted\_median\_after)  
  
# Considering these results, we observe a significant increase in the average time after the change implementation.  
# This suggests that the change might have had a negative impact.  
# The median shows better results than the mean but is still twice higher than before the modification.  
  
  
# Now we will try to understand if those bad results are related to some difference between datasets parameters before and after the modifications.  
  
  
# Age of the applicants before and after:  
  
# Ensure the datetime columns are in the correct format  
submission['event\_datetime'] = pd.to\_datetime(submission['event\_datetime']).dt.tz\_localize(None)  
submission['birth\_date'] = pd.to\_datetime(submission['birth\_date']).dt.tz\_localize(None)  
  
submission['applicant\_Age'] = (submission['event\_datetime'] - submission['birth\_date']).astype('<m8[Y]')  
# print(submission)  
  
# Calculate the average age before and after modification:  
average\_age\_before = submission[submission['Before\_After'] == 'Before']['applicant\_Age'].mean()  
average\_age\_after = submission[submission['Before\_After'] == 'After']['applicant\_Age'].mean()  
  
print(f"Average age before the modification: {average\_age\_before:.2f} years")  
print(f"Average age after the modification: {average\_age\_after:.2f} years")  
  
# We got a very close result before and after the modification (68.4 vs 67.7),  
# so this parameter can't explain why the submission time increased after the modification.  
  
  
  
# Now let's seee the submission time per gender:  
# Filter the DataFrame for 'Before' and 'After'  
before\_df = submission[submission['Before\_After'] == 'Before']  
after\_df = submission[submission['Before\_After'] == 'After']  
  
# Count the number of men and women for 'Before'  
male\_count\_before = before\_df[before\_df['gender'] == 'male'].shape[**0**]  
female\_count\_before = before\_df[before\_df['gender'] == 'female'].shape[**0**]  
  
# Count the number of men and women for 'After'  
male\_count\_after = after\_df[after\_df['gender'] == 'male'].shape[**0**]  
female\_count\_after = after\_df[after\_df['gender'] == 'female'].shape[**0**]  
  
print("\nGender counts before change:")  
print(f"Male: {male\_count\_before}")  
print(f"Female: {female\_count\_before}")  
  
print("\nGender counts after change':")  
print(f"Male: {male\_count\_after}")  
print(f"Female: {female\_count\_after}")  
  
# Here again, no significant difference in the gender distribution  
  
  
# Now let's see if the difference per Recruiter:  
# Calculate the average delta\_time for each Recruiter\_name for "Before"  
average\_delta\_time\_before = submission[submission['Before\_After'] == 'Before'].groupby(Recruiter\_name')['delta\_time'].mean()  
  
# Calculate the average delta\_time for each Recruiter\_name for "After"  
average\_delta\_time\_after = submission[submission['Before\_After'] == 'After'].groupby(Recruiter\_name')['delta\_time'].mean()  
  
print("\nAverage delta\_time before by Recruiter\_name:")  
print(average\_delta\_time\_before)  
  
print("\nAverage delta\_time after by Recruiter\_name:")  
print(average\_delta\_time\_after)  
  
  
# We can see it as a plot:  
# Convert timedelta to total seconds for plotting  
average\_delta\_time\_before\_seconds = average\_delta\_time\_before.apply(lambda x: x.total\_seconds())  
average\_delta\_time\_after\_seconds = average\_delta\_time\_after.apply(lambda x: x.total\_seconds())  
  
# Plotting the averages  
fig**,** ax = plt.subplots(**2, 1,** figsize=(**12, 10**))  
  
# Plot for 'Before'  
ax[**0**].bar(average\_delta\_time\_before\_seconds.index**,** average\_delta\_time\_before\_seconds.values**,** color='blue')  
ax[**0**].set\_title('Average Delta Time Before Modification per 'Recruiter Name')  
ax[**0**].set\_xlabel('Recruiter Name')  
ax[**0**].set\_ylabel('Average Delta Time (seconds)')  
ax[**0**].tick\_params(axis='x'**,** rotation=**90**)  
  
# Plot for 'After'  
ax[**1**].bar(average\_delta\_time\_after\_seconds.index**,** average\_delta\_time\_after\_seconds.values**,** color='green')  
ax[**1**].set\_title('Average Delta Time After Modification per 'Recruiter Name')  
ax[**1**].set\_xlabel('Recruiter Name')  
ax[**1**].set\_ylabel('Average Delta Time (seconds)')  
ax[**1**].tick\_params(axis='x'**,** rotation=**90**)  
  
# Adjust layout  
plt.tight\_layout()  
  
  
# Define the width of each bar  
bar\_width = **0.35**# Define the x positions for the bars  
x\_before = np.arange(len(average\_delta\_time\_before\_seconds))  
x\_after = np.arange(len(average\_delta\_time\_after\_seconds))  
  
# Plotting the averages for both "Before" and "After" on separate plots  
plt.figure(figsize=(**12, 8**))  
  
# Plot for 'Before'  
plt.bar(x\_before**,** average\_delta\_time\_before\_seconds.values**,** width=bar\_width**,** color='blue'**,** label='Before')  
  
# Plot for 'After'  
plt.bar(x\_after + bar\_width**,** average\_delta\_time\_after\_seconds.values**,** width=bar\_width**,** color='green'**,** alpha=**0.5,** label='After')  
  
plt.title('Average Delta Time Before and After Modification per 'Recruiter Name')  
plt.xlabel('Recruiter Name')  
plt.ylabel('Average Delta Time (seconds)')  
plt.xticks(np.arange(max(len(x\_before)**,** len(x\_after)))**,** average\_delta\_time\_before\_seconds.index**,** rotation=**90**)  
plt.legend()  
  
# Show the plot  
plt.tight\_layout()  
plt.show()



