0.0.1 Question 2c

Use seaborn to create a **density** histogram showing the distribution of calls by hour.
Include the Kernal Density Estimate (KDE) graph on your histogram.

Be sure that your axes are labeled and that your plot is titled.

In [ ]: ...

```
ax = sns.histplot(calls, x = 'Hour', stat='density',kde=True, label='casual')

ax.set_xlabel("Time of day (Hour)")

ax.set_ylabel("Percent of Calls")

ax.set_title("Distribution of Calls For Each Hour of the Day")


 # Your code above this line

 # Leave this for grading purposes
 ax_3d = plt.gca()
```

0.0.2 Question 2e

i). Use seaborn to construct a box plot showing the distribution of calls by hour.

ii). To better understand the time of day a report occurs we could **stratify the analysis by DayType (i.e. by weekday vs weekends).**

To do this, use seaborn to create side-by-side violin plots comparing the distribution of calls by hour on the weekend vs weekday (here's a reminder of how to create violin plots in Seaborn: https://seaborn.pydata.org/generated/seaborn.violinplot.html)

Note: For aesthetic purposes only the violin plot continues past the end of the whiskers (i.e. past 0 and 24 hours); however it is not possible to get data points outside of the whiskers for this distribution.

In [ ]: ...

```
ax = sns.boxplot(data=calls,y="Hour",saturation=0.5, palette="Set2")

ax.set_title("Analysis of Phone Calls by Hour");

# Your code for boxplot above this line
# Include a title and label both axes
```

In [ ]: ...

```
ax = sns.violinplot(data=calls.sort_values("CVDOW"), x="DayType", y="Hour",
saturation=0.5)

ax.set_title("Stratified Analysis of Phone Calls by Weekend vs Weekday");

# Your code for side-by-side violin plots above this line
# Include a title and label both axes
```

0.0.3 Question 2f

Based on your histogram, boxplot, and violin plots above, what observations can you make about the patterns of calls? Answer each of the following questions:

i). Are there more calls in the day or at night?

ii). What are the most and least popular times?

iii). Do call patterns and/or IQR vary by weekend vs weekday?

We see the standard pattern of limited activity early in the early morning (around 5am). The violin plot has no very clear patterns.

0.0.4 Question 3c

i). Create a series `missing_by_crime` that calculates the fraction of missing (lat/long) data by event type. **Your series should only include events that have missing lat/long data.** Sort the percentages from highest to lowest.

In [ ]: ...

```
missing_by_crime = (missing_lat_lon['CVLEGEND']
                    .value_counts()
                    / calls['CVLEGEND'].value_counts()
                   ).dropna().sort_values(ascending=False)

# Your code above this line
missing_by_crime
```

In [ ]: ...

```
plt.barh(missing_by_crime.index, missing_by_crime)

plt.xlabel("Percent of missing data")

# Your code to create the barplot above this line
```

0.0.4 Question 3d

Based on the plots above, are there any patterns among entries that are missing latitude/longitude data?

Based on the plots above, give your recommendation as to how we should handle the missing data, and justify your answer:

Option 1). Drop rows with missing data

Option 2). Set missing data to NaN

Option 3). Impute data

While some dates have more unlabeled data than others, it seems that a small percentage of Burglary and Fraud calls don't have GPS coordinates.

Because these account for such a small percentage of overall crimes, it seems reasonable to drop this data.

0.0.5 Question 3e

We'll end by visualizing the number of calls as a function of time, to see if there are any seasonal patterns in call volume during the time period the data spans.

Start by grouping the calls dataframe to create a new dataframe called count_by_date, with index EVENT_TS and column CALL_TOT that gives the number of calls on that specific date, sorted in chronological order.

The first 5 rows of your count_by_date dataframe should be:

EVENT_TS	CALL_TOT
2020-12-17	10
2020-12-18	18
2020-12-19	14
2020-12-20	14
2020-12-21	16

Then create a lineplot displaying this data (with x-axis equal to EVENT_TS and y-axis equal to CALL_TOT). Be sure to label your axes.

```
In [ ]: count_by_date = ...
```

count_by_date = calls.groupby("EVENT_TS").agg({"CASENO": "count"}).rename(columns={"CASENO":"CALL_TOT"})

```
        count_by_date.head()
```

```
In [ ]: ...
```

sns.lineplot(data = count_by_date, x="EVENT_TS", y="CALL_TOT")

```
        # Code to create a lineplot above this line
```

```
In [ ]: grader.check("q3e")
```