D610 Task 2

.round(3)

Research Question

A. Summarize the real-data research question you identified in Task 1. Your summary should include justification for the research question you identified in Task 1, a description of the context in which the research question exists, and a discussion of your hypothesis.

My research question for the capstone project is: “To what extent do time of day, weekend or not, and the presence of a highway affect injury rates in vehicle crashes?” I chose this research question because vehicle crashes are a continuous safety concern in California, where large populations and busy highways contribute to frequent accidents. Understanding when and where injury rates are highest can inform interventions aimed at improving public health and road safety.

The context for this question is the state of California, which experiences a significant number of vehicle accidents each year. Despite existing research on traffic safety, there is a limited analysis of how the combination of time of day, weekend versus weekday status, and highway versus non-highway locations influences injury rates. By analyzing crash data, including variables such as city, type of collision, time, day, and location, I aim to address this gap. I plan to use publicly available crash data from state agencies to ensure findings are grounded in real, recent incidents.

My hypothesis is that the time of day, whether or not the crash occurred on a weekend, and whether it happened on a highway statistically significantly affect injury rates in vehicle crashes. I believe nighttime, weekends, and highways were associated with higher injury rates due to factors such as reduced visibility, increased traffic volume, or higher speeds. My daily observation of frequent highway crashes and the broader trends in California traffic support the expectation that these variables will prove significant in the analysis.

Data Collection

B. Report on your data-collection process by describing the relevant data you collected, discussing **one** advantage and **one** disadvantage of the data-gathering methodology you used, and discussing how you overcame any challenges you encountered during the process of collecting your data.

My data collection process includes relevant sources that contain accurate and consistent data. I needed access to large datasets containing car crashes throughout the state. I noticed that all of this information was present on the California government website. I found information regarding crashes dating back to 2016.

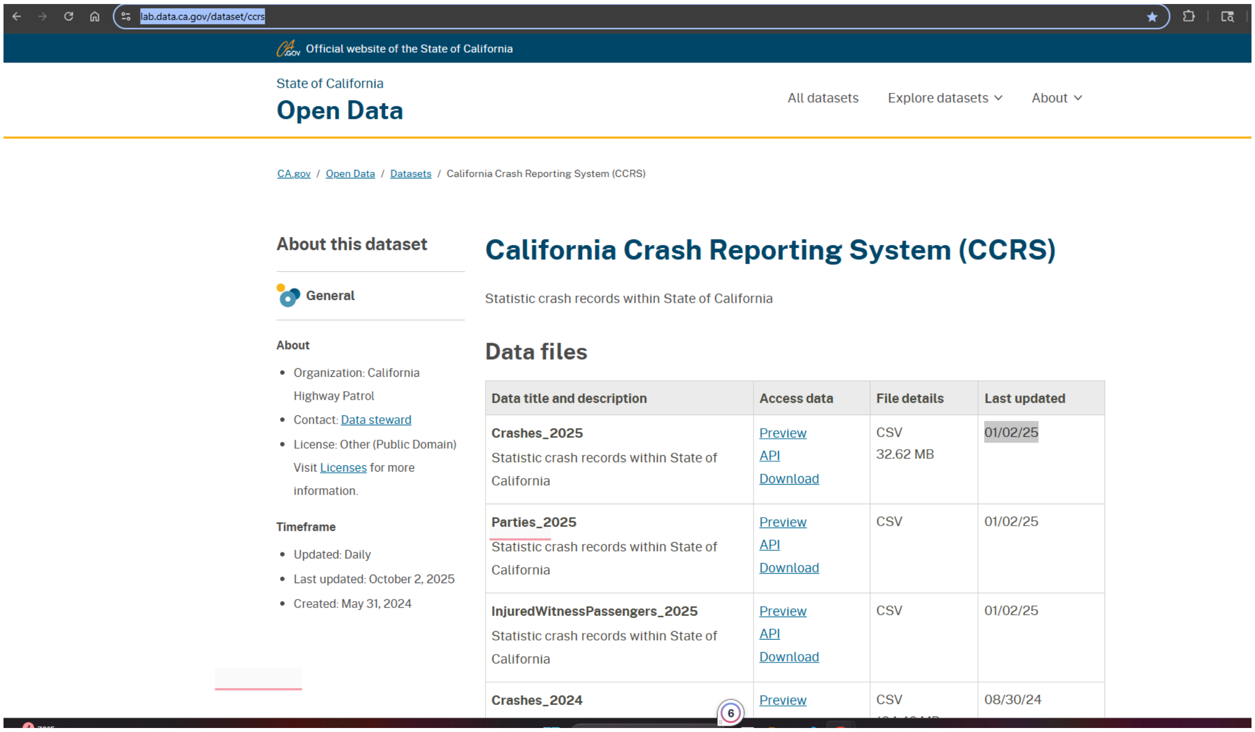
One advantage of this data-gathering methodology is that, since it is official California government data, we can assume that it contains a substantial amount, if not all, of the data on crashes in California. One disadvantage of this method is that the 2025 dataset does not get regularly ingested. According to the website, the 2025 datasets were last updated on January 2, 2025.

One main challenge was finding datasets that not only contained data in California, but also datasets that were not mock data that accurately represented real-world vehicle crashes. Upon searching, I noticed that the California government website features the California Crash Reporting System (CCRS) and was able to locate the dataset required for my analysis.

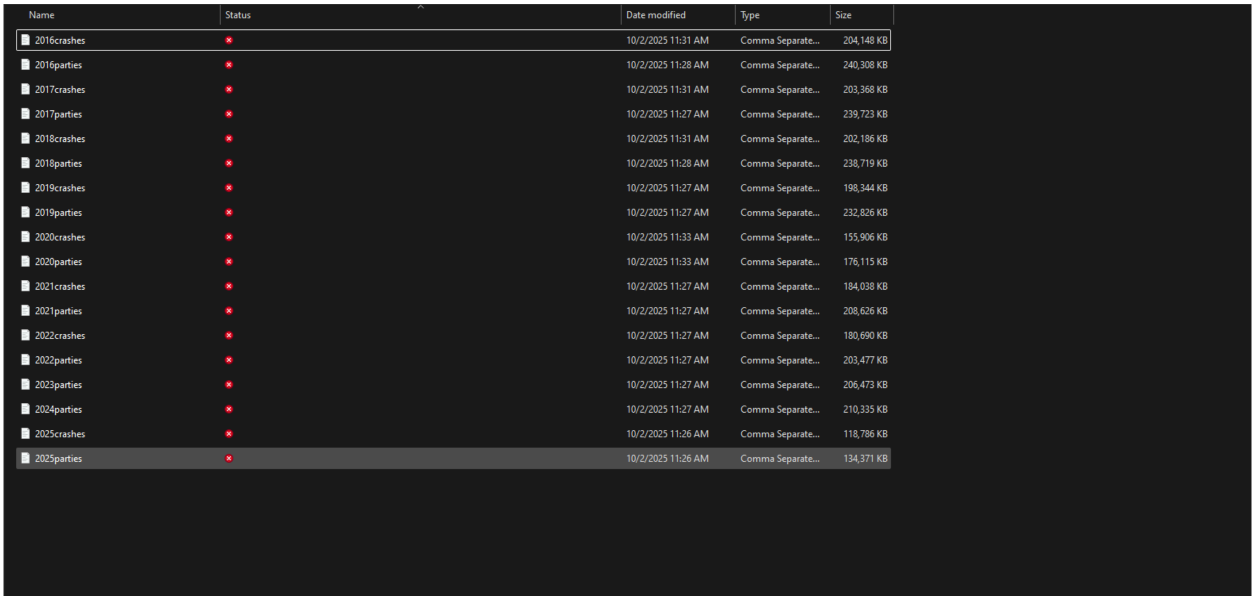
Data Extraction and Preparation

C. Describe your data extraction and preparation process and provide screenshots to illustrate *each* step. Explain the tools and techniques you used for data extraction and data preparation, including how these tools and techniques were used on the data. Justify why you used these particular tools and techniques, including **one** advantage and **one** disadvantage of using them with your data-extraction and preparation methods.

1. On the website, https://lab.data.ca.gov/dataset/ccrs, I downloaded the crashes and parties CSV files due to the files having the independent and dependent variables respectively.

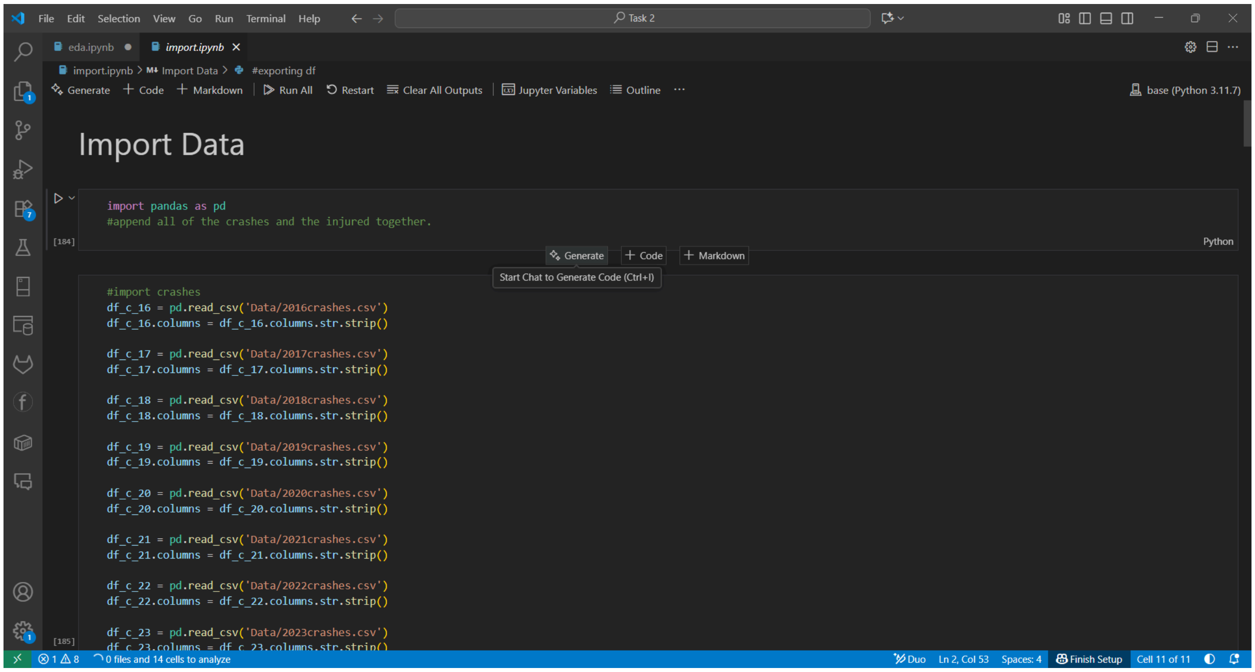


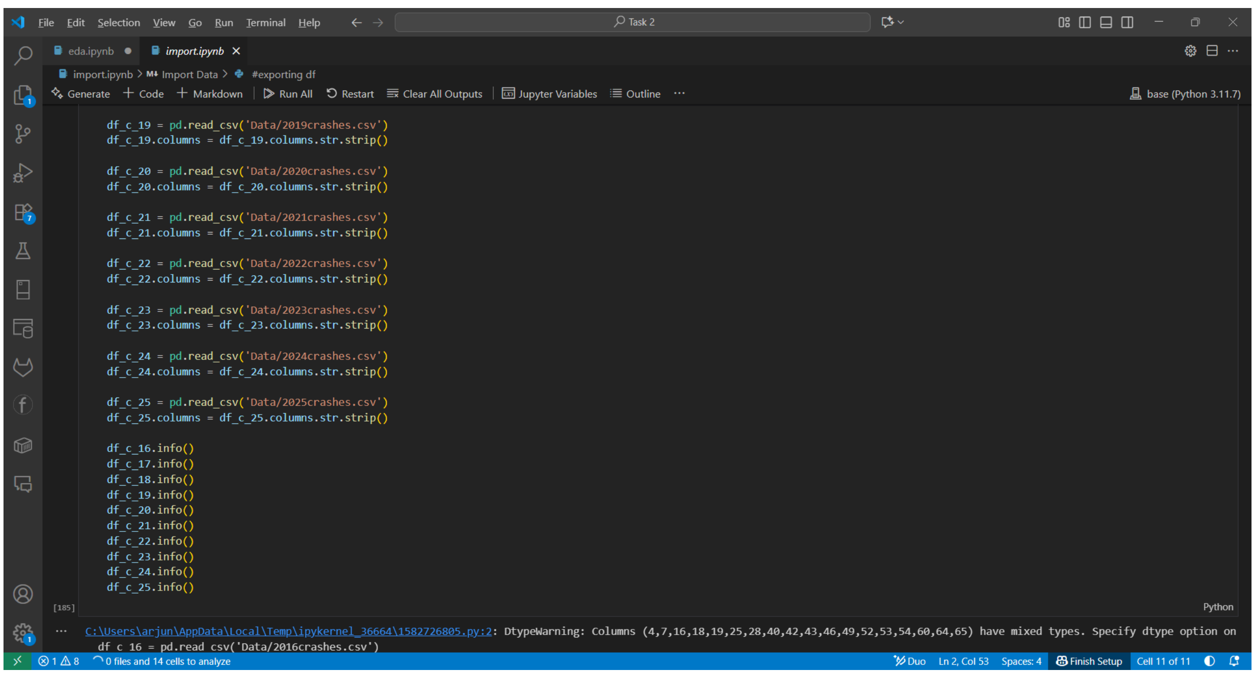
2. I then changed the names of the files and added them to a folder which I am saving as the Data Lake.

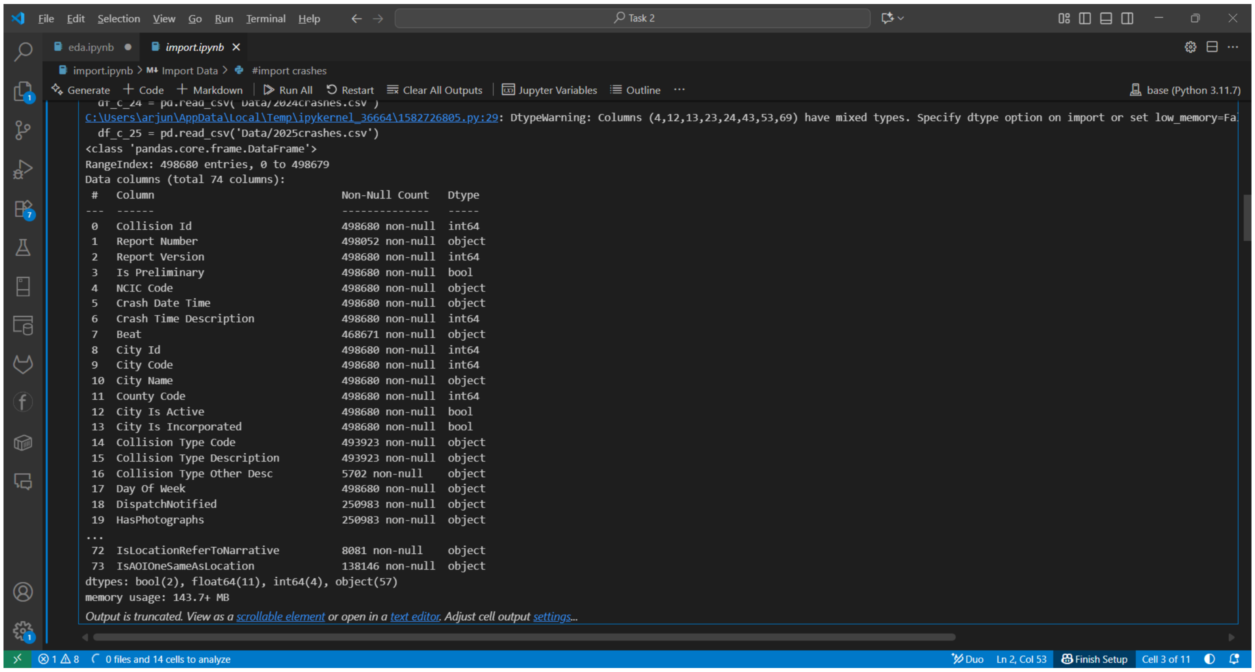


Name	Status	Date modified	Type	Size
2016crashes		10/2/2025 11:31 AM	Comma Separate...	204,148 KB
2016parties		10/2/2025 11:28 AM	Comma Separate...	240,308 KB
2017crashes		10/2/2025 11:31 AM	Comma Separate...	203,368 KB
2017parties		10/2/2025 11:27 AM	Comma Separate...	239,723 KB
2018crashes		10/2/2025 11:31 AM	Comma Separate...	202,186 KB
2018parties		10/2/2025 11:28 AM	Comma Separate...	238,719 KB
2019crashes		10/2/2025 11:27 AM	Comma Separate...	198,344 KB
2019parties		10/2/2025 11:27 AM	Comma Separate...	232,826 KB
2020crashes		10/2/2025 11:33 AM	Comma Separate...	155,906 KB
2020parties		10/2/2025 11:33 AM	Comma Separate...	176,115 KB
2021crashes		10/2/2025 11:27 AM	Comma Separate...	184,038 KB
2021parties		10/2/2025 11:27 AM	Comma Separate...	208,626 KB
2022crashes		10/2/2025 11:27 AM	Comma Separate...	180,690 KB
2022parties		10/2/2025 11:27 AM	Comma Separate...	203,477 KB
2023parties		10/2/2025 11:27 AM	Comma Separate...	206,473 KB
2024parties		10/2/2025 11:27 AM	Comma Separate...	210,335 KB
2025crashes		10/2/2025 11:26 AM	Comma Separate...	118,786 KB
2025parties		10/2/2025 11:26 AM	Comma Separate...	134,371 KB

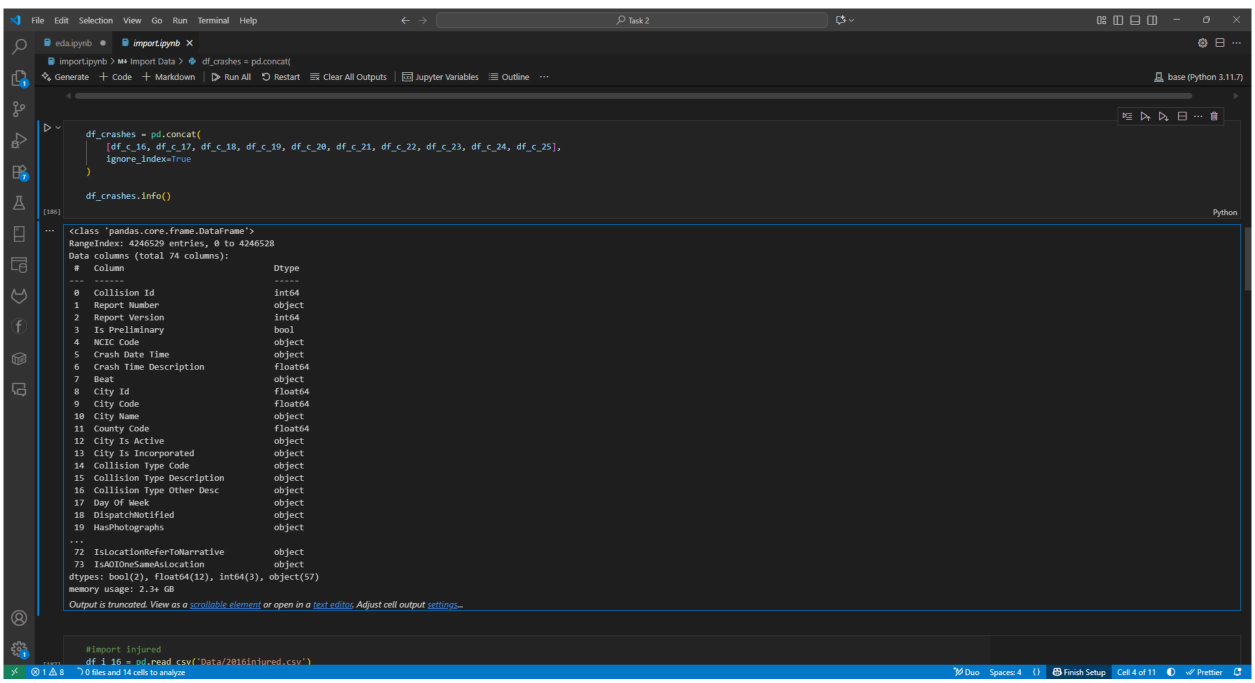
3. Upon importing, I went to my IDE. Here, I am using the Python library Pandas, and I will be working with dataframes for the various CSV files that need to be imported. I wish to use pandas dataframes as it allows for easy use and changes to the table whenever needed. It also allows for visualization and can effectively be used alongside other libraries which will be see soon. One disadvantage is that it operates primarily in-memory. This means that I will need to continuously check in-memory usage especially since the datafram is huge. I then import all the crash files, which provide the required variables: day of the week, Collision ID, and Crash Time Description. (Pandas, 2025)


File Edit Selection View Go Run Terminal Help
Task 2
eda.ipynb
import.ipynb
import.ipynb > Import Data > #exporting df
Generate + Code + Markdown Run All Restart Clear All Outputs Jupyter Variables Outline
base (Python 3.11.7)
Import Data
import pandas as pd
#append all of the crashes and the injured together.
[184]
Python
Generate + Code + Markdown
Start Chat to Generate Code (Ctrl+I)
#import crashes
df_c_16 = pd.read_csv('Data/2016crashes.csv')
df_c_16.columns = df_c_16.columns.str.strip()
df_c_17 = pd.read_csv('Data/2017crashes.csv')
df_c_17.columns = df_c_17.columns.str.strip()
df_c_18 = pd.read_csv('Data/2018crashes.csv')
df_c_18.columns = df_c_18.columns.str.strip()
df_c_19 = pd.read_csv('Data/2019crashes.csv')
df_c_19.columns = df_c_19.columns.str.strip()
df_c_20 = pd.read_csv('Data/2020crashes.csv')
df_c_20.columns = df_c_20.columns.str.strip()
df_c_21 = pd.read_csv('Data/2021crashes.csv')
df_c_21.columns = df_c_21.columns.str.strip()
df_c_22 = pd.read_csv('Data/2022crashes.csv')
df_c_22.columns = df_c_22.columns.str.strip()
df_c_23 = pd.read_csv('Data/2023crashes.csv')
[185]
0 files and 14 cells to analyze
Duo Ln 2, Col 53 Spaces: 4 Finish Setup Cell 11 of 11

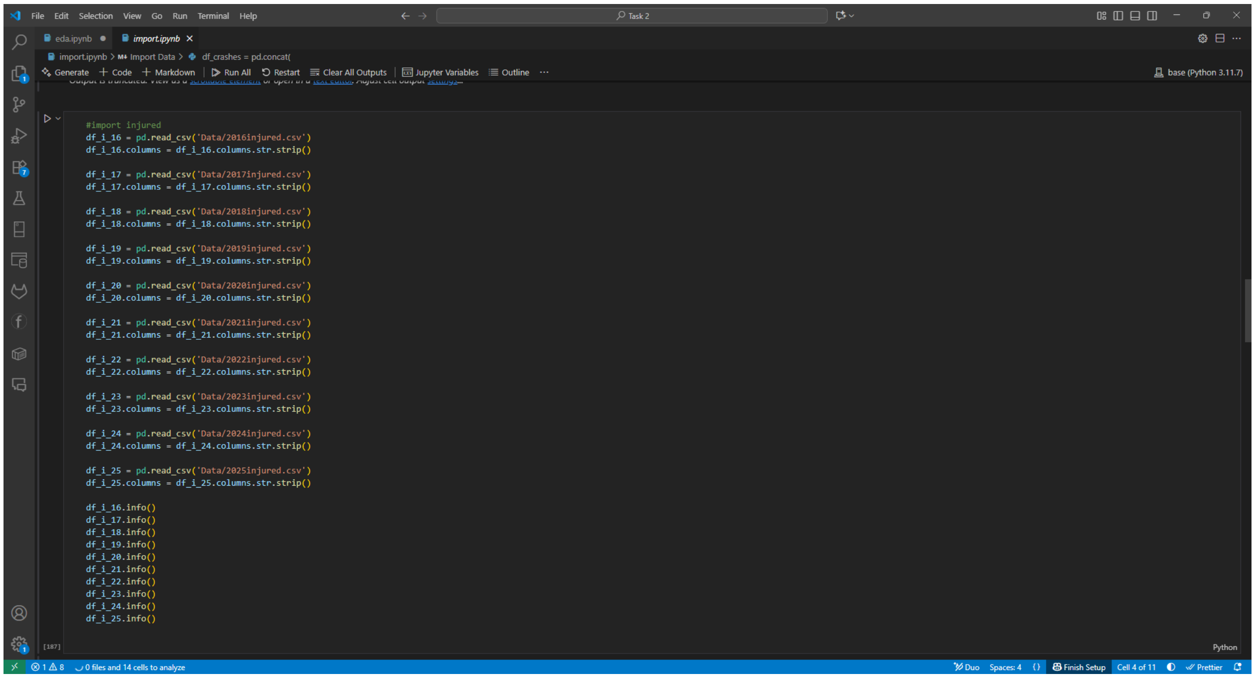

File Edit Selection View Go Run Terminal Help
Task 2
eda.ipynb
import.ipynb
import.ipynb > Import Data > #exporting df
Generate + Code + Markdown Run All Restart Clear All Outputs Jupyter Variables Outline
base (Python 3.11.7)
df_c_19 = pd.read_csv('Data/2019crashes.csv')
df_c_19.columns = df_c_19.columns.str.strip()
df_c_20 = pd.read_csv('Data/2020crashes.csv')
df_c_20.columns = df_c_20.columns.str.strip()
df_c_21 = pd.read_csv('Data/2021crashes.csv')
df_c_21.columns = df_c_21.columns.str.strip()
df_c_22 = pd.read_csv('Data/2022crashes.csv')
df_c_22.columns = df_c_22.columns.str.strip()
df_c_23 = pd.read_csv('Data/2023crashes.csv')
df_c_23.columns = df_c_23.columns.str.strip()
df_c_24 = pd.read_csv('Data/2024crashes.csv')
df_c_24.columns = df_c_24.columns.str.strip()
df_c_25 = pd.read_csv('Data/2025crashes.csv')
df_c_25.columns = df_c_25.columns.str.strip()
df_c_16.info()
df_c_17.info()
df_c_18.info()
df_c_19.info()
df_c_20.info()
df_c_21.info()
df_c_22.info()
df_c_23.info()
df_c_24.info()
df_c_25.info()
[185]
Python
C:\Users\arjun\AppData\Local\Temp\ipykernel_36664\1582726805.py:2: DtypeWarning: Columns (4,7,16,18,19,25,28,40,42,43,46,49,52,53,54,60,64,65) have mixed types. Specify dtype option on
df_c_16 = pd.read_csv('Data/2016crashes.csv')
0 files and 14 cells to analyze
Duo Ln 2, Col 53 Spaces: 4 Finish Setup Cell 11 of 11



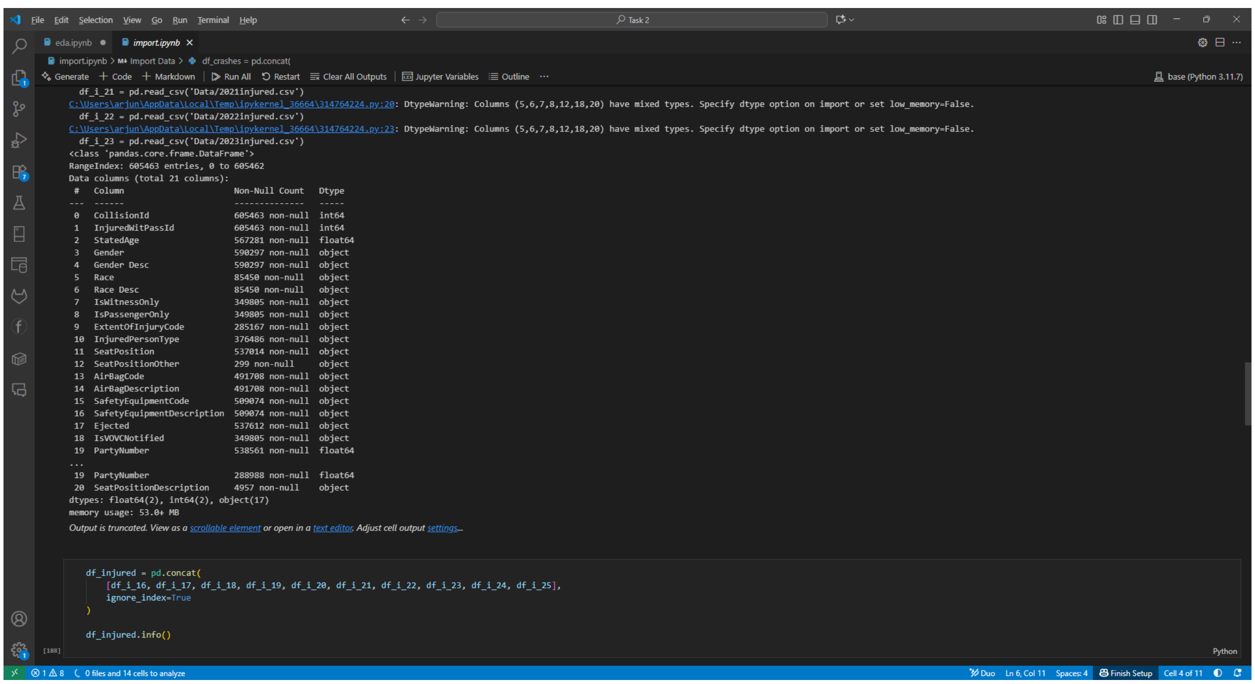
4. I then combine these dataframes into one larger dataframe called df_crashes.



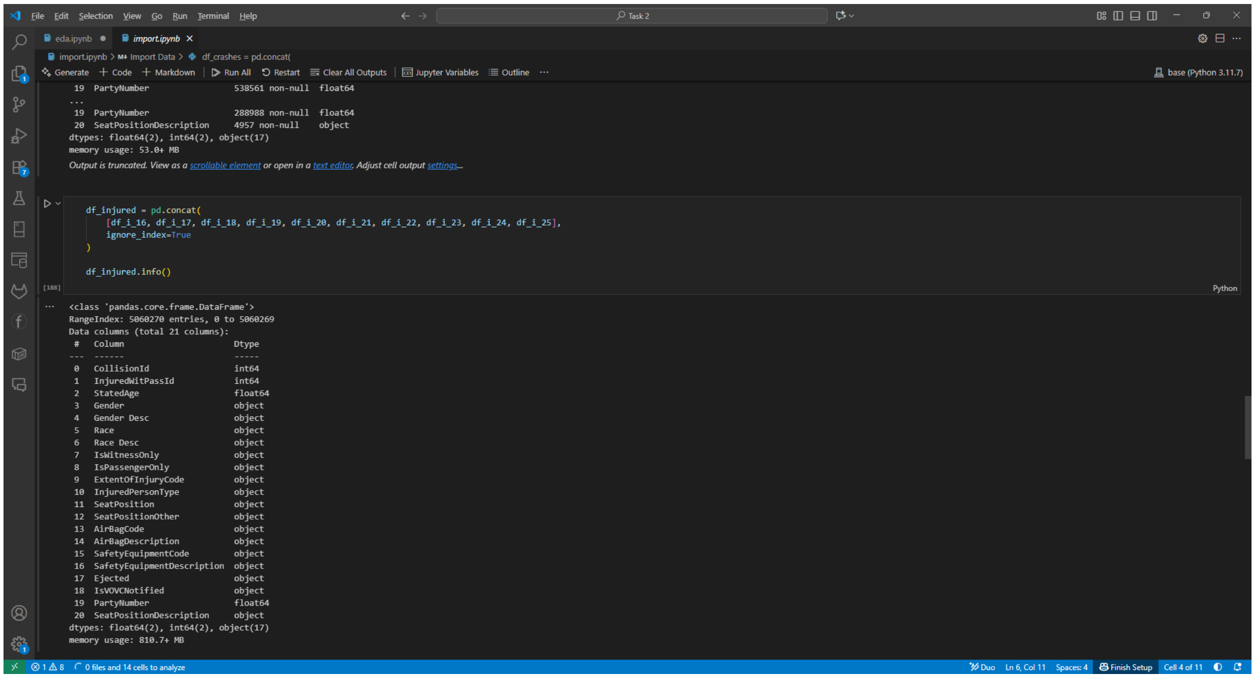
5. I then do steps 3 and 4 for the injured csv files which will give us IsHighwayRelated and ExtentOfInjury variables.



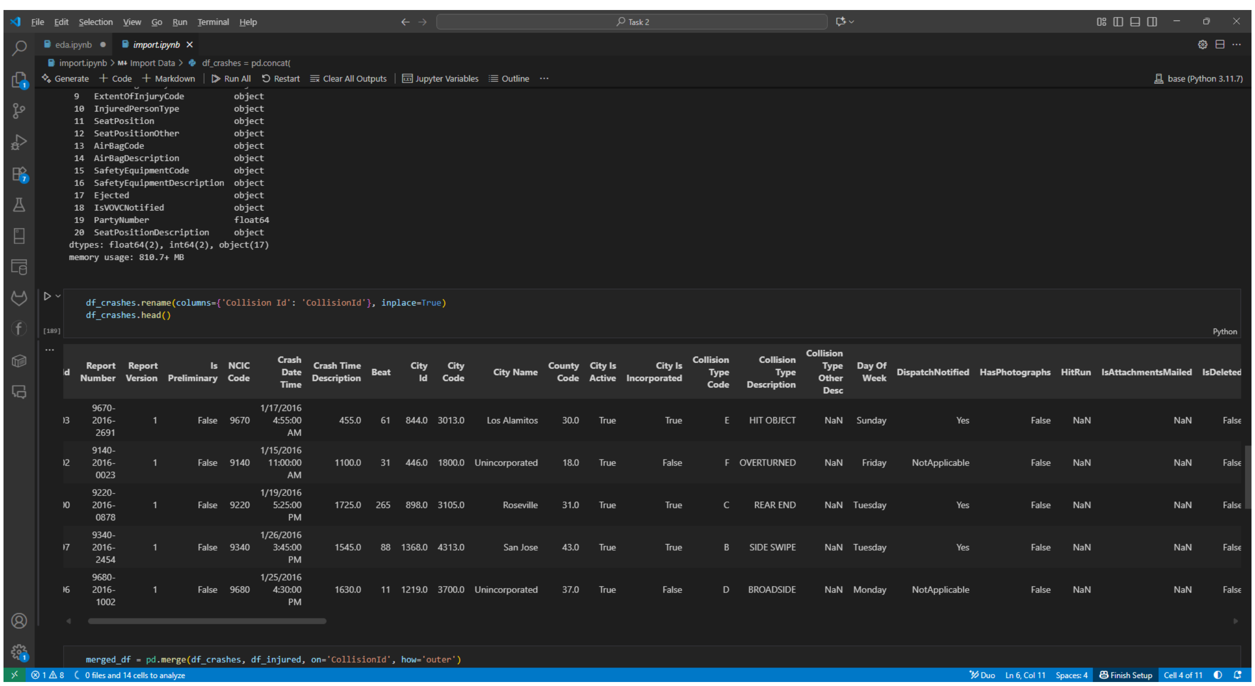
#import injured
df_i_16 = pd.read_csv('Data/2016injured.csv')
df_i_16.columns = df_i_16.columns.str.strip()
df_i_17 = pd.read_csv('Data/2017injured.csv')
df_i_17.columns = df_i_17.columns.str.strip()
df_i_18 = pd.read_csv('Data/2018injured.csv')
df_i_18.columns = df_i_18.columns.str.strip()
df_i_19 = pd.read_csv('Data/2019injured.csv')
df_i_19.columns = df_i_19.columns.str.strip()
df_i_20 = pd.read_csv('Data/2020injured.csv')
df_i_20.columns = df_i_20.columns.str.strip()
df_i_21 = pd.read_csv('Data/2021injured.csv')
df_i_21.columns = df_i_21.columns.str.strip()
df_i_22 = pd.read_csv('Data/2022injured.csv')
df_i_22.columns = df_i_22.columns.str.strip()
df_i_23 = pd.read_csv('Data/2023injured.csv')
df_i_23.columns = df_i_23.columns.str.strip()
df_i_24 = pd.read_csv('Data/2024injured.csv')
df_i_24.columns = df_i_24.columns.str.strip()
df_i_25 = pd.read_csv('Data/2025injured.csv')
df_i_25.columns = df_i_25.columns.str.strip()
df_i_16.info()
df_i_17.info()
df_i_18.info()
df_i_19.info()
df_i_20.info()
df_i_21.info()
df_i_22.info()
df_i_23.info()
df_i_24.info()
df_i_25.info()
[187]



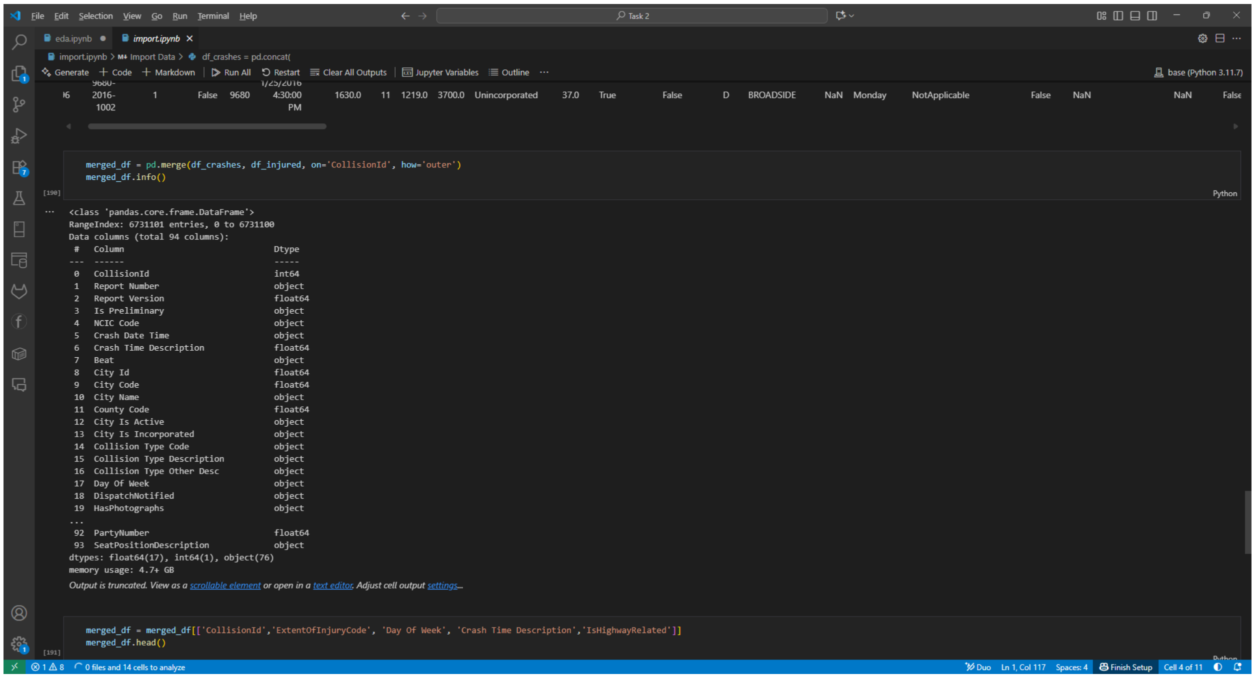
df_i_21 = pd.read_csv('Data/2021injured.csv')
C:\Users\arjun\AppData\Local\Temp\ipykernel_36664\314764224.py:20: DtypeWarning: Columns (5,6,7,8,12,18,20) have mixed types. Specify dtype option on import or set low_memory=False.
df_i_22 = pd.read_csv('Data/2022injured.csv')
C:\Users\arjun\AppData\Local\Temp\ipykernel_36664\314764224.py:23: DtypeWarning: Columns (5,6,7,8,12,18,20) have mixed types. Specify dtype option on import or set low_memory=False.
df_i_23 = pd.read_csv('Data/2023injured.csv')
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 605463 entries, 0 to 605462
Data columns (total 21 columns):
Column Non-Null Count Dtype
0 CollisionId 605463 non-null int64
1 InjuredWitPassId 605463 non-null int64
2 StatedAge 567281 non-null float64
3 Gender 590297 non-null object
4 Gender Desc 590297 non-null object
5 Race 85450 non-null object
6 Race Desc 85450 non-null object
7 IsWitnessOnly 349805 non-null object
8 IsPassengerOnly 349805 non-null object
9 ExtentOfInjuryCode 285167 non-null object
10 InjuredPersonType 376486 non-null object
11 SeatPosition 537014 non-null object
12 SeatPositionOther 299 non-null object
13 AirBagCode 491708 non-null object
14 AirBagDescription 491708 non-null object
15 SafetyEquipmentCode 509074 non-null object
16 SafetyEquipmentDescription 509074 non-null object
17 Ejected 537612 non-null object
18 IsVOVCNotified 349805 non-null object
19 PartyNumber 538561 non-null float64
...
19 PartyNumber 288988 non-null float64
20 SeatPositionDescription 4957 non-null object
dtypes: float64(2), int64(2), object(17)
memory usage: 53.0+ MB
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
df_injured = pd.concat(
[df_i_16, df_i_17, df_i_18, df_i_19, df_i_20, df_i_21, df_i_22, df_i_23, df_i_24, df_i_25],
ignore_index=True
)
df_injured.info()
[188]



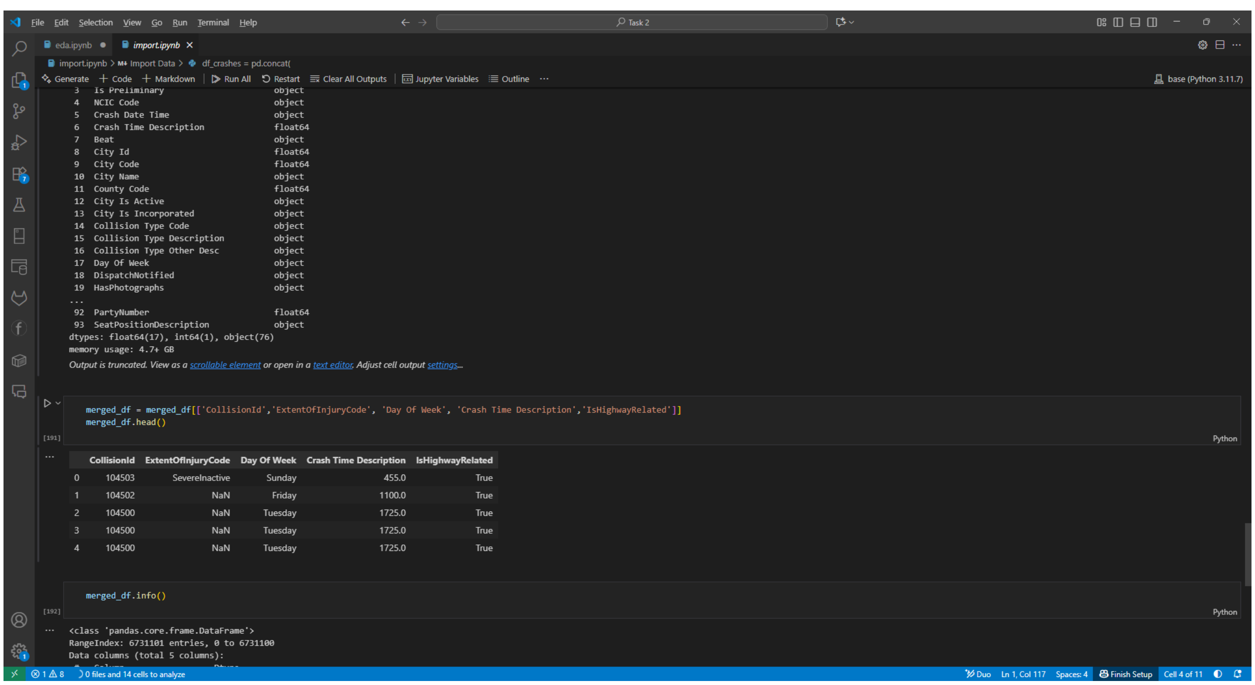
6. Rename “Collision Id” in crashes dataframe to “CollisionId” to allow a join between the two dataframes.



7. Merge the two dataframes together

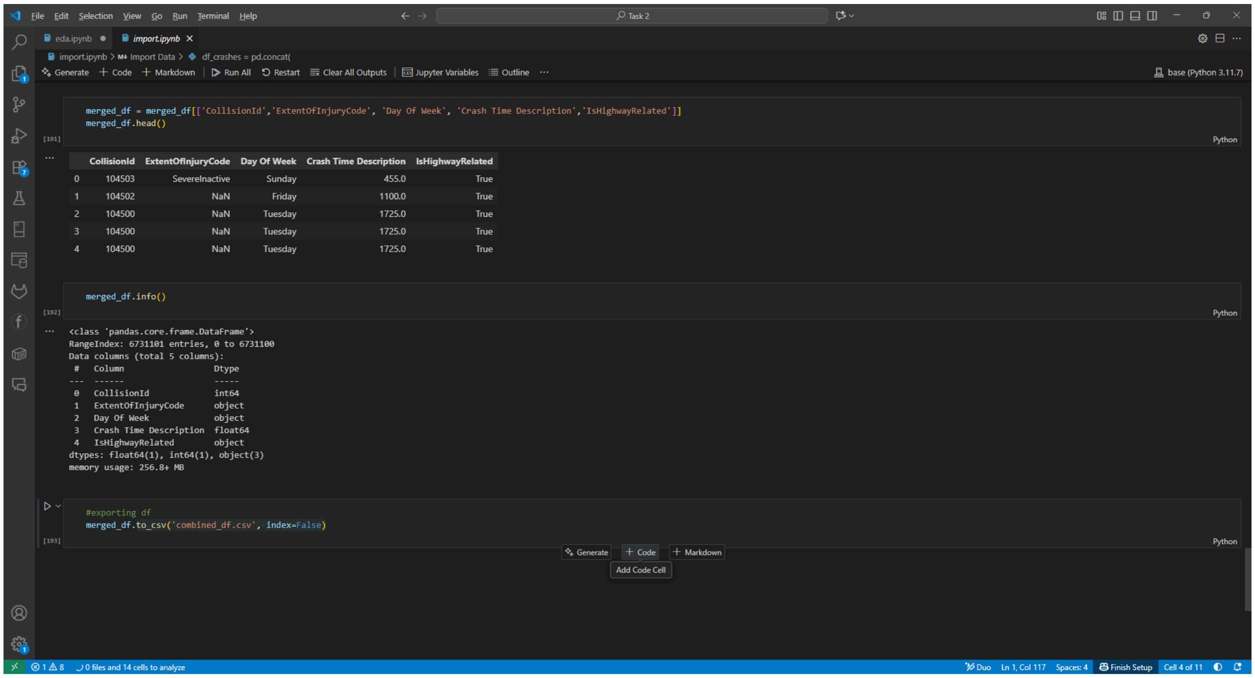


8. Only keep the required variables in the dataframes

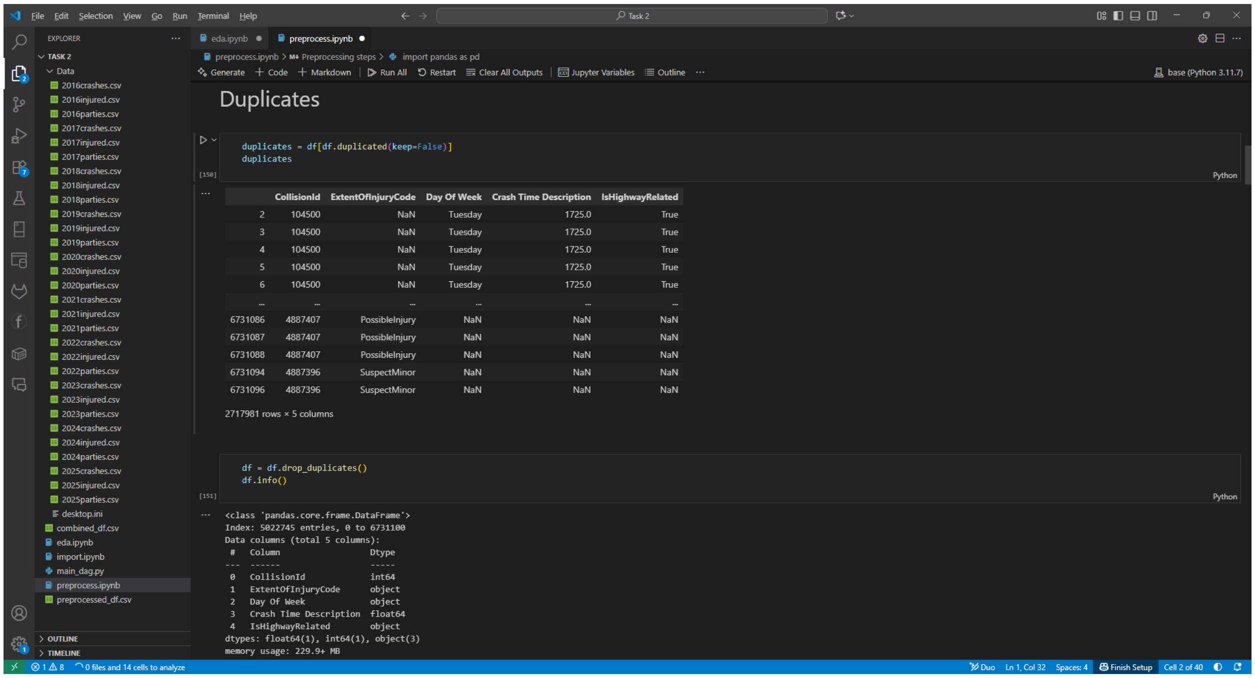


9. I then exported the data into a csv file to be used by another notebook to allow for easier readability. I also would not have to rerun all the code everytime I

made a mistake.



10. I have now started the preprocessing steps. After importing the relevant csv file that was exported in the last step, I will remove the duplicates using the drop method. (McKinney, 2022)



Duplicates

```
duplicates = df[df.duplicated(keep=False)]
duplicates
```

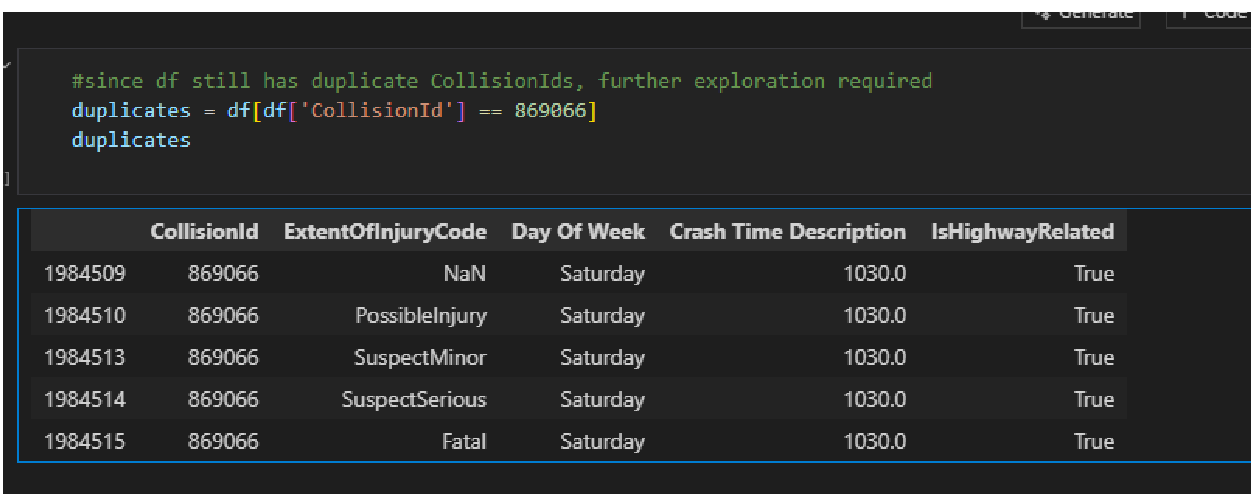
	CollisionId	ExtentOfInjuryCode	Day Of Week	Crash Time Description	IsHighwayRelated
2	104500	NaN	Tuesday	1725.0	True
3	104500	NaN	Tuesday	1725.0	True
4	104500	NaN	Tuesday	1725.0	True
5	104500	NaN	Tuesday	1725.0	True
6	104500	NaN	Tuesday	1725.0	True
...	...	...	...	...	...
6731086	4887407	PossibleInjury	NaN	NaN	NaN
6731087	4887407	PossibleInjury	NaN	NaN	NaN
6731088	4887407	PossibleInjury	NaN	NaN	NaN
6731094	4887396	SuspectMinor	NaN	NaN	NaN
6731096	4887396	SuspectMinor	NaN	NaN	NaN

2717981 rows × 5 columns

```
df = df.drop_duplicates()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 5022745 entries, 0 to 6731100
Data columns (total 5 columns):
 #   Column                  Dtype  
---  ------                  -----  
 0   CollisionId             int64  
 1   ExtentOfInjuryCode      object 
 2   Day Of Week             object 
 3   Crash Time Description  float64
 4   IsHighwayRelated        object 
dtypes: float64(1), int64(1), object(3)
memory usage: 229.9+ MB
```

Upon removing duplicates, I noticed that there were still duplicates within the CollisionID column. Since CollisionID is being used as my unique identifier for rows, this was a problem. However, upon further inspection, it was found that there were multiple collisionIDs, but they were unique (see CollisionId 869066)



```
#since df still has duplicate CollisionIds, further exploration required
duplicates = df[df['CollisionId'] == 869066]
duplicates
```

	CollisionId	ExtentOfInjuryCode	Day Of Week	Crash Time Description	IsHighwayRelated
1984509	869066	NaN	Saturday	1030.0	True
1984510	869066	PossibleInjury	Saturday	1030.0	True
1984513	869066	SuspectMinor	Saturday	1030.0	True
1984514	869066	SuspectSerious	Saturday	1030.0	True
1984515	869066	Fatal	Saturday	1030.0	True

Upon further evaluation, I concluded that there may have been multiple people in both cars, resulting in various different ExtentOfInjuryCode values. I decided to drop CollisionId and replace it with a new unique identifier. This should be fine for the analysis as this allows us to keep a lot of the ExtentofInjuryCode data while removing the CollisionId data which may not be too useful.

```python
#upon further eval, learned that the reason why there were many different collision ids is because there may be many people in a car which is creating this issue.
#will remove CollisionId and replace with regular id
df["ID"] = range(1, len(df) + 1)

df = df.drop('CollisionId', axis=1)

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 5022745 entries, 0 to 6731100
Data columns (total 5 columns):
 #   Column                 Dtype
---  ------                 -----
 0   ExtentOfInjuryCode     object
 1   Day Of Week            object
 2   Crash Time Description float64
 3   IsHighwayRelated       object
 4   ID                     int64
dtypes: float64(1), int64(1), object(3)
memory usage: 229.9+ MB
```

11. I then renamed the columns, allowing me to have shorter, more concise variable names which more accurately descibed the variables.

```python
#rename columns
df = df.rename(columns={'ID': 'ID', 'ExtentOfInjuryCode': 'is_injured', 'Day Of Week': 'is_weekend', 'Crash Time Description': 'crash_time', 'IsHighwayRelated': 'is_highway'})
```

12. I changed the column types. The reason why is to save space in memory. We are dealing with a dataset that has over 5 million rows, and when altered a column from int64 to int32 this ends up saving a large chunk of memory.

```
#changing coolumn types
df.info()
```

[159]

```
<class 'pandas.core.frame.DataFrame'>
Index: 5022745 entries, 0 to 6731100
Data columns (total 5 columns):
 #   Column      Dtype  
---  ------      -----  
 0   ID          int64  
 1   is_injured  object 
 2   is_weekend  object 
 3   crash_time  float64
 4   is_highway  object 
dtypes: float64(1), int64(1), object(3)
memory usage: 229.9+ MB
```

```
df['ID'] = df['ID'].astype('int16')
df['crash_time'] = df['crash_time'].astype('float32')
```

[160]

```
df.info()
```

[161]

```
<class 'pandas.core.frame.DataFrame'>
Index: 5022745 entries, 0 to 6731100
Data columns (total 5 columns):
 #   Column      Dtype  
---  ------      -----  
 0   ID          int16  
 1   is_injured  object 
 2   is_weekend  object 
 3   crash_time  float32
 4   is_highway  object 
dtypes: float32(1), int16(1), object(3)
memory usage: 182.0+ MB
```

13. Next, we will impute missing values. I will input the categorical values using the mode, while the continuous variables will be represented by the mean or median, depending on the data distribution. (I imputed is_injured, is_weekend, and is_highway with the mode, while imputing crash_time with median as the

graph is skewed). I found the shape of the graph using the seaborn library for crash_time. One advantage of using this is that it can quickly spin up visualizations, requiring minimal setup. The disadvantage to using this is that the library does not allow for a lot of customization. (Seaborn, 2025)

Missing Values

```
columns = ['ID','is_injured', 'is_weekend','crash_time','is_highway'
for i in columns:
    print(f" {i} : {df[i].isnull().sum()}")
```

[62]

```
ID : 0
is_injured : 3134929
is_weekend : 13962
crash_time : 13964
is_highway : 13965
```

```
#since is_injured is categorical
injured_mode = df['is_injured'].mode()[0]

df['is_injured'] = df['is_injured'].fillna(injured_mode)

df['is_injured'].isnull().sum()
```

[63]

```
0
```

```
#since is_weekend is categorical
weekend_mode = df['is_weekend'].mode()[0]

df['is_weekend'] = df['is_weekend'].fillna(weekend_mode)

df['is_weekend'].isnull().sum()
```

[64]

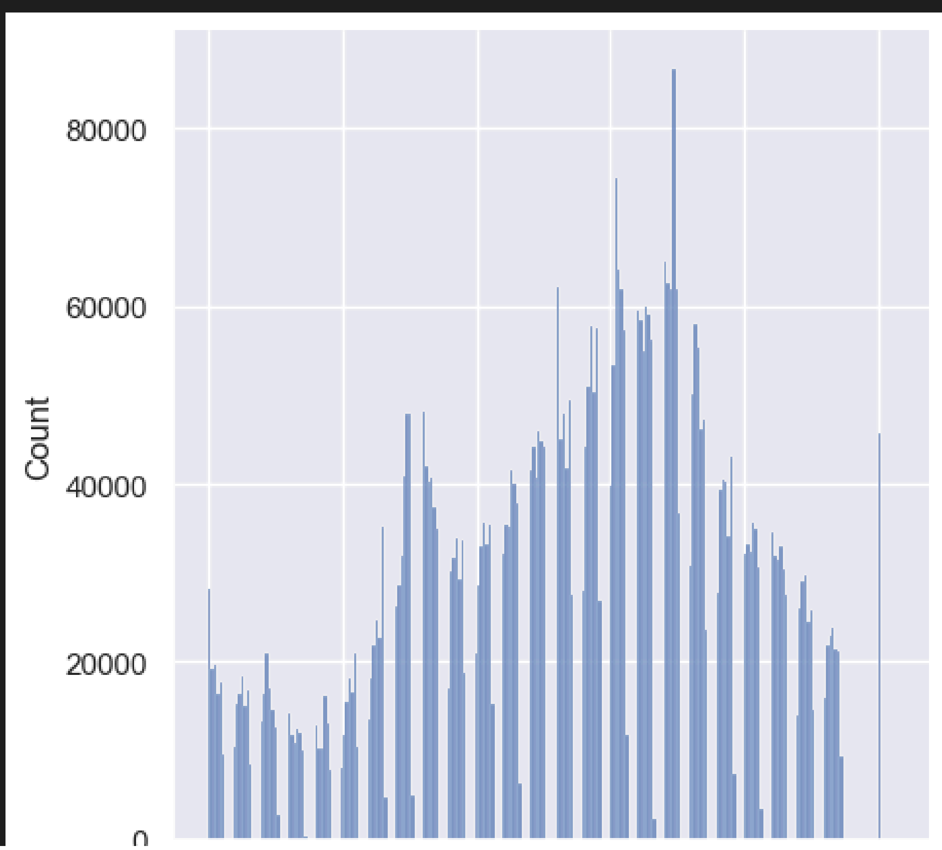
```
#since is_highway is categorical
highway_mode = df['is_highway'].mode()[0]

df['is_highway'] = df['is_highway'].fillna(highway_mode)

df['is_highway'].isnull().sum()
```

```
0
```

```
#shape of crash time
sns.displot(data=df, x=df['crash_time'])
```

```
c:\Users\arjun\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: Future
  with pd.option_context('mode.use_inf_as_na', True):
```

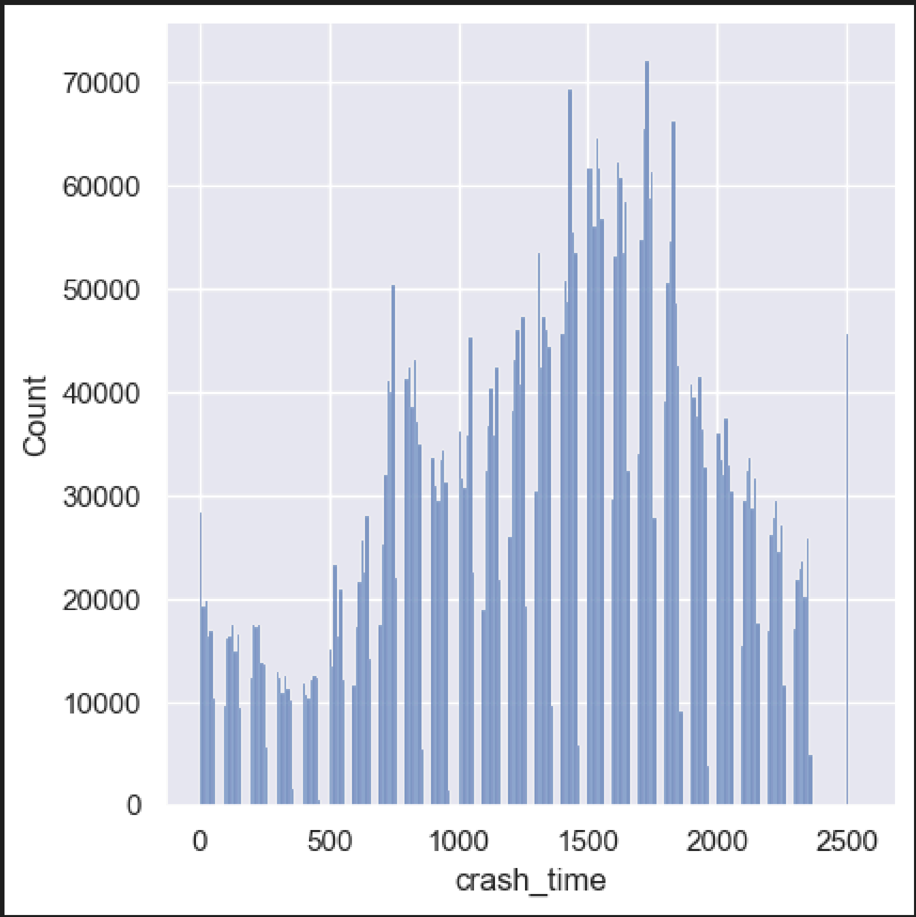
```
<seaborn.axisgrid.FacetGrid at 0x1f084bfcb90>
```



```python
#median imputation for crash_time
df['crash_time'] = df['crash_time'].fillna(df['crash_time'].median())

sns.displot(data=df, x=df['crash_time'])
```

```
c:\Users\arjun\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning
  with pd.option_context('mode.use_inf_as_na', True):
```

```
<seaborn.axisgrid.FacetGrid at 0x1f100ed1650>
```



14. I will recode variables. is_injured, is weekend, and is_highway will be recoded as booleans.

Recoding Variables

```python
df['is_injured'].value_counts()
```

Python

```
is_injured
ComplaintOfPainInactive    3705811
PossibleInjury              541083
SuspectMinor                350747
OtherVisibleInactive        246118
SuspectSerious               93054
SevereInactive               50548
Fatal                        35384
Name: count, dtype: int64
```

```python
#will group ComplaintOfPainInactive and OtherVisibleInactive and SevereInactive together as 0 and the rest as 1
df['is_injured'] = df['is_injured'].map({'ComplaintOfPainInactive': 0, 'PossibleInjury': 1, 'SuspectMinor': 1, 'OtherVisibleInactive': 0,'SuspectSerious': 1, 'SevereInactive': 0, "Fatal": 1 })

df['is_injured'].value_counts()
```

Python

```
is_injured
0    4002477
1    1020268
Name: count, dtype: int64
```

```python
df['is_weekend'].value_counts()
```

Python

```
is_weekend
Friday       824803
Thursday     734785
Wednesday    720892
Saturday     714833
Tuesday      714649
Monday       686935
Sunday       625848
Name: count, dtype: int64
```

```python
##will group Saturday and Sunday together as 1 and the rest as 0
df['is_weekend'] = df['is_weekend'].map({'Sunday': 1, 'Monday': 0, 'Tuesday': 0, 'Wednesday': 0, 'Thursday': 0, 'Friday': 0,'Saturday': 1})

df['is_weekend'].value_counts()
```

[172]

```
is_weekend
0    3682064
1    1340681
Name: count, dtype: int64
```

```python
df['is_highway'].value_counts()
```

[173]

```
is_highway
False    3037860
True     1984885
Name: count, dtype: int64
```

```python
df['is_highway'] = df['is_highway'].replace({True: 1, False: 0})
```

[174]

```python
df['is_highway'].value_counts()
```

[175]

```
is_highway
0    3037860
1    1984885
Name: count, dtype: int64
```

15. I wanted to change the crash_time to the nearest hour. The reason for this is that this will give us a better understanding of the time of day a crash occurs.

```python
df['crash_time'] = df['crash_time'].round(decimals= -2)
df['crash_time'].value_counts()
```

```
crash_time
1600.0    372109
1700.0    353653
1500.0    341142
1400.0    338550
1800.0    338313
1200.0    270793
800.0     268419
1300.0    257131
1900.0    227939
2000.0    213830
1100.0    212992
1000.0    210849
700.0     196803
900.0     186089
2100.0    182331
2200.0    175891
600.0     144319
2300.0    132092
200.0     105935
0.0       100740
100.0      94710
500.0      89514
400.0      72652
300.0      70107
2500.0     45671
2400.0     20162
2600.0         9
Name: count, dtype: int64
```

```python
#converting 2400 to 0, 2500 to 100, 2600 to 200
def standardize_time(val):
    if val >= 2400:
        return int(val - 2400)
    return int(val)

df["crash_time"] = df["crash_time"].apply(standardize_time)
```

16. I noticed that although crash_time was in military hours, there were values that were at 2500 or 2600. I simply returned these as 100 or 200, respectively. There may, however, be issues if it was done on a Sunday, then this would count as a day for the weekend and not a weekday, as the crash occurred on a Monday. I decided that it was important to leave it on the day it occurred, as many people still regard 1 or 2am on a Monday as part of the weekend.

```
#converting 2400 to 0, 2500 to 100, 2600 to 200
def standardize_time(val):
    if val >= 2400:
        return int(val - 2400)
    return int(val)

df["crash_time"] = df["crash_time"].apply(standardize_time)
```

17. This data was then exported into another notebook for further analysis.

```
df.to_csv('preprocessed_df.csv', index=False)
```

Analysis

D. Report on your data-analysis process by describing the analysis techniques you used to appropriately analyze the data. Include the calculations you performed and their outputs. Justify how you selected the analysis techniques you used, including **one** advantage and **one** disadvantage of these techniques.

1. I employed several different techniques for analysis. Since I am examining the effect of the independent variables is_weekend, crash_time, and is_highway on the dependent variable, I first wish to review the descriptive statistics. One advantage of this technique is it provides me a quick way to get a high level overview of the data. The disadvantage of this method is that if there are many independent categorical variables, not much information can be recovered from them.

```python
df.describe()
```

✓ 1.6s Python

	ID	is_injured	is_weekend	crash_time	is_highway
count	5.022745e+06	5.022745e+06	5.022745e+06	5.022745e+06	5.022745e+06
mean	5.459663e+01	2.031296e-01	2.669220e-01	1.307383e+03	3.951793e-01
std	1.893999e+04	4.023282e-01	4.423513e-01	5.917926e+02	4.888892e-01
min	-3.276800e+04	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
25%	-1.636800e+04	0.000000e+00	0.000000e+00	9.000000e+02	0.000000e+00
50%	1.520000e+02	0.000000e+00	0.000000e+00	1.400000e+03	0.000000e+00
75%	1.646000e+04	0.000000e+00	1.000000e+00	1.800000e+03	1.000000e+00
max	3.276700e+04	1.000000e+00	1.000000e+00	2.300000e+03	1.000000e+00

Since many of the variables are categorical, there is not much to use from the descriptive statistics. Not much important information can be gathered from crash_time's descriptive statistics.

2. Another method is to look at the relationships between the dependent variable and the independent variables.

I decided to use logistic regression as is_injured is a categorical variable and crash_time is a continuous variable. These are the results form the model. One advantage of this method is the efficiency and speed of setting up the regression model. This allows for quick analysis and understanding. A disadvantage of this is that since the model only contains a single predictor variable, this limits the model's ability to predict possible outcomes.

I have the results of the analysis below:

```python
#find relationship between is_injured and crash_time

# Extract features and target
X = df[['crash_time']]  # Needs to be a 2D array/DataFrame
y = df['is_injured']    # Binary target variable

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

# Create and fit logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# Predict on test set
y_pred = model.predict(X_test)

# Evaluate model performance
print(classification_report(y_test, y_pred))
```

✓ 8.5s

	precision	recall	f1-score	support
0	0.80	1.00	0.89	800582
1	0.00	0.00	0.00	203967
accuracy			0.80	1004549
macro avg	0.40	0.50	0.44	1004549
weighted avg	0.64	0.80	0.71	1004549

The model is great at predicting if someone is not_injured, but is not great at predicting if someone is injured. This means that the accuracy is misleading. This could be due to the large number of not-injured cases (800,582)and a relatively small number of injured cases (203967).

3. I also created contingency tables and performed Chi-squared tests for the categorical variables. (is_weekend and is_highway). One advantage of the Chi-squared test is that it does not assume a normal distribution, making it suitable for analyzing categorical data without requiring specific distributional constraints. One disadvantage is that it does not indicate the strength or direction of the relationship, only that one exists.

```python
# Create a contingency table for is_weekend and is_injured
contingency_table = pd.crosstab(df['is_weekend'], df['is_injured'])

# Perform Chi-square test
chi2, p, dof, expected = chi2_contingency(contingency_table)

print('is_weekend:')
print("Chi-square statistic:", chi2)
print("p-value:", p)
```

✓ 0.3s

```
is_weekend:
Chi-square statistic: 227.16265096121217
p-value: 2.4782063721989298e-51
```

For the p-value, it can be inferred that there is overwhelming evidence to show that the two variables are not independent. The data strongly suggests that injury occurrence depends on whether or not it is the weekend.

```python
# Create a contingency table for is_highway and is_injured
contingency_table = pd.crosstab(df['is_highway'], df['is_injured'])

# Perform Chi-square test
chi2, p, dof, expected = chi2_contingency(contingency_table)

print('is_highway:')
print("Chi-square statistic:", chi2)
print("p-value:", p)
```

✓ 0.2s

```
is_highway:
Chi-square statistic: 34392.193906757995
p-value: 0.0
```

Similarly, for the is_highway variable, it was found that the p-value is so small, it was recorded as 0. This too shows that the data strongly suggests that injury occurrence depends on whether or not the collision happens on the highway.

Outcomes and Implications

E. Summarize the implications of your data analysis by discussing the results in the context of the research question, including **one** limitation of your analysis. Within the context of your research question, recommend a course of action based on your results. Then propose **two** directions or approaches for future study of the dataset.

The data analysis provides strong evidence supporting the research question that time of day, weekend status, and presence of a highway significantly affect injury rates in vehicle crashes in California. Logistic regression showed that while crash_time alone is insufficient to accurately predict injuries due to class imbalance, the model efficiently distinguishes between non-injured and injured cases. Chi-squared tests confirmed statistically significant associations between injury and categorical variables, such as is_weekend and is_highway, with overwhelming evidence rejecting their independence.

One limitation of the analysis is the heavy class imbalance between injured and non-injured cases, which affects the predictive power of the logistic regression model and may give misleading accuracy metrics. Is_injured was also heavily imputed (close to 35%), which may also skew the results.

Based on these results, a recommended course of action is to incorporate additional relevant predictor variables and apply techniques to address class imbalance (such as resampling or class-weight adjustments) to improve model performance for injury risks. Adding interaction terms or nonlinear modeling could also enhance insights.

Two directions for future study include:

1. Expanding the analysis to include other environmental factors (weather, road conditions) to better understand influences on injury occurrence.
2. Applying advanced machine learning algorithms or ensemble methods that can handle imbalanced data more effectively and potentially reveal complex

nonlinear relationships for improved risk prediction.

These steps will deepen the understanding of factors affecting injury outcomes and support the development of improved traffic safety policies and interventions in California.

F. Acknowledge sources, using in-text citations and references, for content that is quoted, paraphrased, or summarized.

1. "W3schools.Com." W3Schools Online Web Tutorials, W3Schools, www.w3schools.com/python/default.asp. Accessed 31 Oct. 2025.
2. "California Crash Reporting System (CCRS)." *California Crash Reporting System (CCRS) | CA Open Data*, lab.data.ca.gov/dataset/ccrs. Accessed 31 Oct. 2025.
3. "Pandas Documentation." *Pandas Documentation - Pandas 2.3.3 Documentation*, pandas.pydata.org/docs/. Accessed 31 Oct. 2025.
4. "Statistical Data Visualization." *Seaborn*, seaborn.pydata.org/. Accessed 31 Oct. 2025.
5. McKinney, Wes. *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and Jupyter*. O'Reilly Media, Inc, 2022.

G. Demonstrate professional communication in the content and presentation of your submission.