STOR 320: Introduction to Data Science

Spring 2025

EDA Group 6

In [2]:
```
import pandas as pd
import numpy as np
from bs4 import BeautifulSoup
import requests
from io import StringIO
import seaborn as sns
import matplotlib.pyplot as plt
!pip install us
import us
```

```
Collecting us
  Downloading us-3.2.0-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: jellyfish in /usr/local/lib/python3.11/dist-pac
kages (from us) (1.1.0)
Downloading us-3.2.0-py3-none-any.whl (13 kB)
Installing collected packages: us
Successfully installed us-3.2.0
```

In [3]:
```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving insurance_claims.csv to insurance_claims.csv
```

In [4]:
```
insurance_claims_df = pd.read_csv("insurance_claims.csv")

url = "https://www.fhwa.dot.gov/policyinformation/statistics/2015/mv1.cfm"
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser')
table = soup.find('table')
all_motor_vehicles_df = pd.read_html(StringIO(str(table)))[0]
```

Creator: Branda Sisoutham

Q1: How does the time of day of reported incidents differ between states with high vs. low vehicle registration?

In [ ]:
```
print(all_motor_vehicles_df.columns.tolist())

# Select and rename the relevant columns
vehicle_reg_df = all_motor_vehicles_df[[('STATE', 'STATE'),
                                         ('ALL MOTOR VEHICLES', 'TOTAL'),
                                         ('ALL MOTOR VEHICLES',
```

```
                                          'PUBLICLY OWNED')]].copy()
vehicle_reg_df.columns = ['state', 'total_vehicles',
                           'public_vehicles']
```

```
[('STATE', 'STATE'), ('AUTOMOBILES', 'PRIVATE AND COMMERCIAL (INCLUDING TAXICA
BS)'), ('AUTOMOBILES', 'PUBLICLY OWNED'), ('AUTOMOBILES', 'TOTAL'), ('BUSES',
'PRIVATE AND COMMERCIAL'), ('BUSES', 'PUBLICLY OWNED'), ('BUSES', 'TOTAL'),
('TRUCKS', 'PRIVATE AND COMMERCIAL'), ('TRUCKS', 'PUBLICLY OWNED'), ('TRUCKS',
'TOTAL'), ('MOTORCYCLES', 'PRIVATE AND COMMERCIAL'), ('MOTORCYCLES', 'PUBLICLY
OWNED (1)'), ('MOTORCYCLES', 'TOTAL'), ('ALL MOTOR VEHICLES', 'PRIVATE AND COM
MERCIAL'), ('ALL MOTOR VEHICLES', 'PUBLICLY OWNED'), ('ALL MOTOR VEHICLES', 'T
OTAL')]
```

In [ ]:

```
# Create a mapping from full state names to abbreviations using the us.states p
state_abbrev_map = {state.name: state.abbr for
                    state in us.states.STATES}

# Add a new column with state abbreviations to match with the insurance claims
vehicle_reg_df['state_abbr'] = vehicle_reg_df[
    'state'].map(state_abbrev_map)
```

In [ ]:

```
# Merge insurance claims with vehicle registration data using the state abbrevi
merged_df = insurance_claims_df.merge(vehicle_reg_df,
                                      left_on='incident_state',
                                      right_on='state_abbr',
                                      how='inner')
merged_df.head(5)
```

Out[ ]:

	months_as_customer	age	policy_number	policy_bind_date	policy_state	policy_csl	policy_
0	328	48	521585	2014-10-17	OH	250/500	
1	228	42	342868	2006-06-27	IN	250/500	
2	134	29	687698	2000-09-06	OH	100/300	
3	256	41	227811	1990-05-25	IL	250/500	
4	228	44	367455	2014-06-06	IL	500/1000	

5 rows × 44 columns

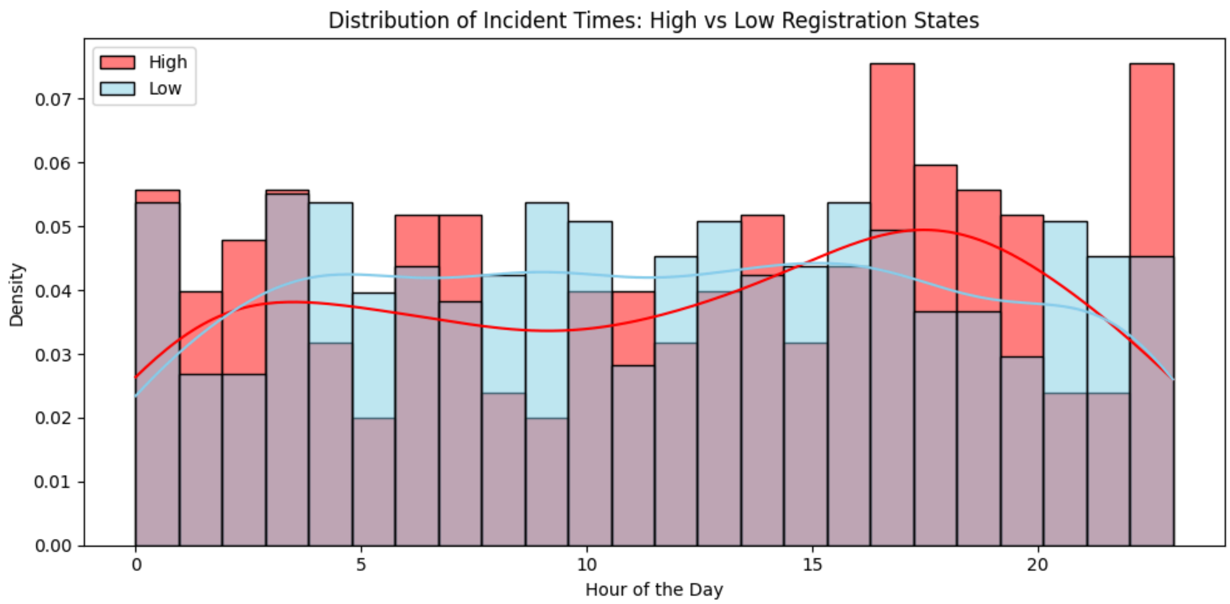
In [ ]:

```
# Calculate the 75th percentile
threshold = merged_df['total_vehicles'].quantile(0.75)

# Create a new column categorizing states as 'Low'
merged_df['registration_category'] = 'Low'
# Reassign to 'High' where the total number of vehicles is greater than or equa
merged_df.loc[merged_df['total_vehicles'] >= threshold,
              'registration_category'] = 'High'
```

In [ ]:

```
# Create a figure for the plot
plt.figure(figsize=(10, 5))

# High registration states
sns.histplot(
    data=merged_df[merged_df['registration_category'] == 'High'],
```

```
    x='incident_hour_of_the_day',
    bins=24, color='red', label='High', stat='density', kde=True)

# Low registration states
sns.histplot(
    data=merged_df[merged_df['registration_category'] == 'Low'],
    x='incident_hour_of_the_day',
    bins=24, color='skyblue', label='Low', stat='density', kde=True )

# Add labels and title
plt.title('Distribution of Incident Times: High vs Low Registration States')
plt.xlabel('Hour of the Day')
plt.ylabel('Density')
plt.legend()
plt.tight_layout()
plt.show()
```



Q2: Do states with a higher percentage of young drivers (16-25) have higher auto insurance claim amounts?

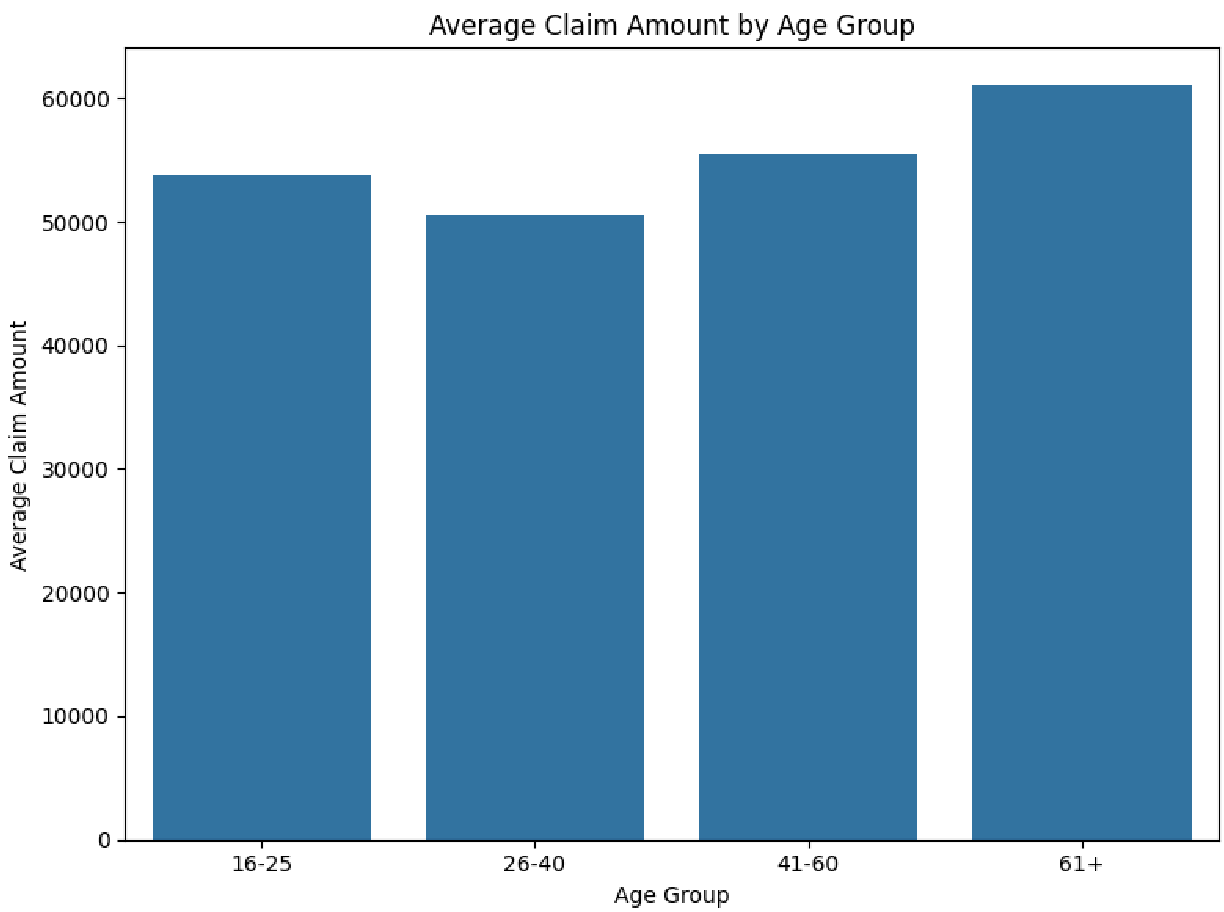
In [ ]:
```
# Create a clean copy of relevant columns
insurance_claims_df_copy = insurance_claims_df[['age',
                                                'incident_state',
                                                'total_claim_amount']].dropna(
# Filter the DataFrame to include only claims from young drivers (ages 16–25)
young_claims = insurance_claims_df_copy[(insurance_claims_df['age'] >= 16) & (
    insurance_claims_df_copy['age'] <= 25)]
```

In [ ]:
```
# Count total number of claims per state (all ages)
total_claims_per_state = insurance_claims_df_copy.groupby('incident_state').si
# Create age groups using pd.cut for visual summaries or comparisons
insurance_claims_df_copy['age_group'] = pd.cut(
    insurance_claims_df['age'],
    bins=[0, 25, 40, 60, 100],
    labels=['16–25', '26–40', '41–60', '61+'],
    right=True
)
```

```
In [ ]: # Keep only necessary columns and calculate average claim amount for each age g
grouped_age_claims = insurance_claims_df_copy[['age_group',
                                               'total_claim_amount']].dropna()
age_group_summary = grouped_age_claims.groupby('age_group')[
    'total_claim_amount'].mean(numeric_only=True).reset_index()
```

```
<ipython-input-29-817033374b7c>:3: FutureWarning: The default of observed=Fals
e is deprecated and will be changed to True in a future version of pandas. Pas
s observed=False to retain current behavior or observed=True to adopt the futu
re default and silence this warning.
  age_group_summary = grouped_age_claims.groupby('age_group')['total_claim_amo
unt'].mean(numeric_only=True).reset_index()
```

```
In [ ]: # Add labels and title
plt.figure(figsize=(8, 6))
sns.barplot(
    data=age_group_summary,
    x='age_group',
    y='total_claim_amount')
plt.title('Average Claim Amount by Age Group')
plt.xlabel('Age Group')
plt.ylabel('Average Claim Amount')
plt.tight_layout()
plt.show()
```



Interpreter: Riley Little

```
In [ ]: insurance_claims_df['policy_deductable'].describe()
```

Out[ ]:

	policy_deductable
count	1000.000000
mean	1136.000000
std	611.864673
min	500.000000
25%	500.000000
50%	1000.000000
75%	2000.000000
max	2000.000000

dtype: float64

Q1: Are policyholders with lower deductibles more likely to be involved in more frequent low-severity incidents compared to those with higher deductibles?

In [ ]:
```
insurance_claims_df['policy_deductable'].value_counts()
```

Out[ ]:

policy_deductable	count
1000	351
500	342
2000	307

dtype: int64

In [ ]:
```
insurance_claims_df['incident_severity'].value_counts()
```

Out[ ]:

incident_severity	count
Minor Damage	354
Total Loss	280
Major Damage	276
Trivial Damage	90

dtype: int64

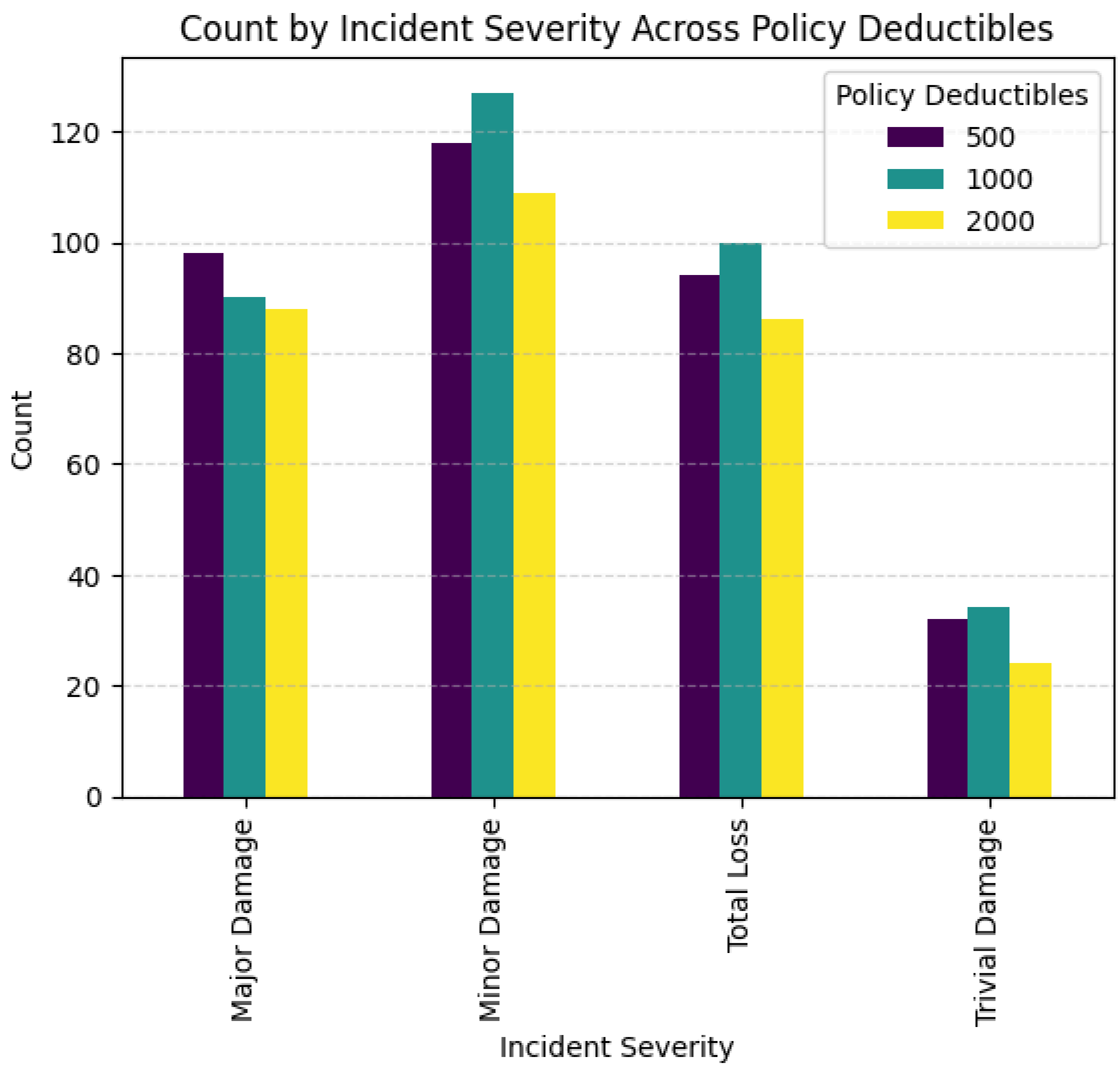
In [ ]:

```
insurance_claims_df['incident_severity'] = insurance_claims_df[
    'incident_severity'].astype('category')
insurance_claims_df['policy_deductable'] = insurance_claims_df[
    'policy_deductable'].astype('category')
```

In [ ]:

```
# Group the data
deduc_counts = insurance_claims_df.groupby(['incident_severity',
                                            'policy_deductable']).size().unsta

# Plot grouped bar chart (stacked=False)
deduc_counts.plot(kind='bar', stacked=False, colormap='viridis')

# Add labels and legend
plt.title("Count by Incident Severity Across Policy Deductibles")
plt.xlabel("Incident Severity")
plt.ylabel("Count")
plt.legend(title="Policy Deductibles")
plt.grid(True, axis='y', linestyle='--', alpha=0.5)

plt.show();
```

```
<ipython-input-19-fcf086778bae>:2: FutureWarning: The default of observed=Fals
e is deprecated and will be changed to True in a future version of pandas. Pas
s observed=False to retain current behavior or observed=True to adopt the futu
re default and silence this warning.
  deduc_counts = insurance_claims_df.groupby(['incident_severity', 'policy_ded
uctable']).size().unstack()
```



Q2: Are policyholders with lower deductibles more likely to submit a fraudulent claim rather than higher deductibles policyholders?

In [ ]:
```
insurance_claims_df['fraud_reported'].value_counts()
```

Out[ ]:

	count
fraud_reported	
N	753
Y	247

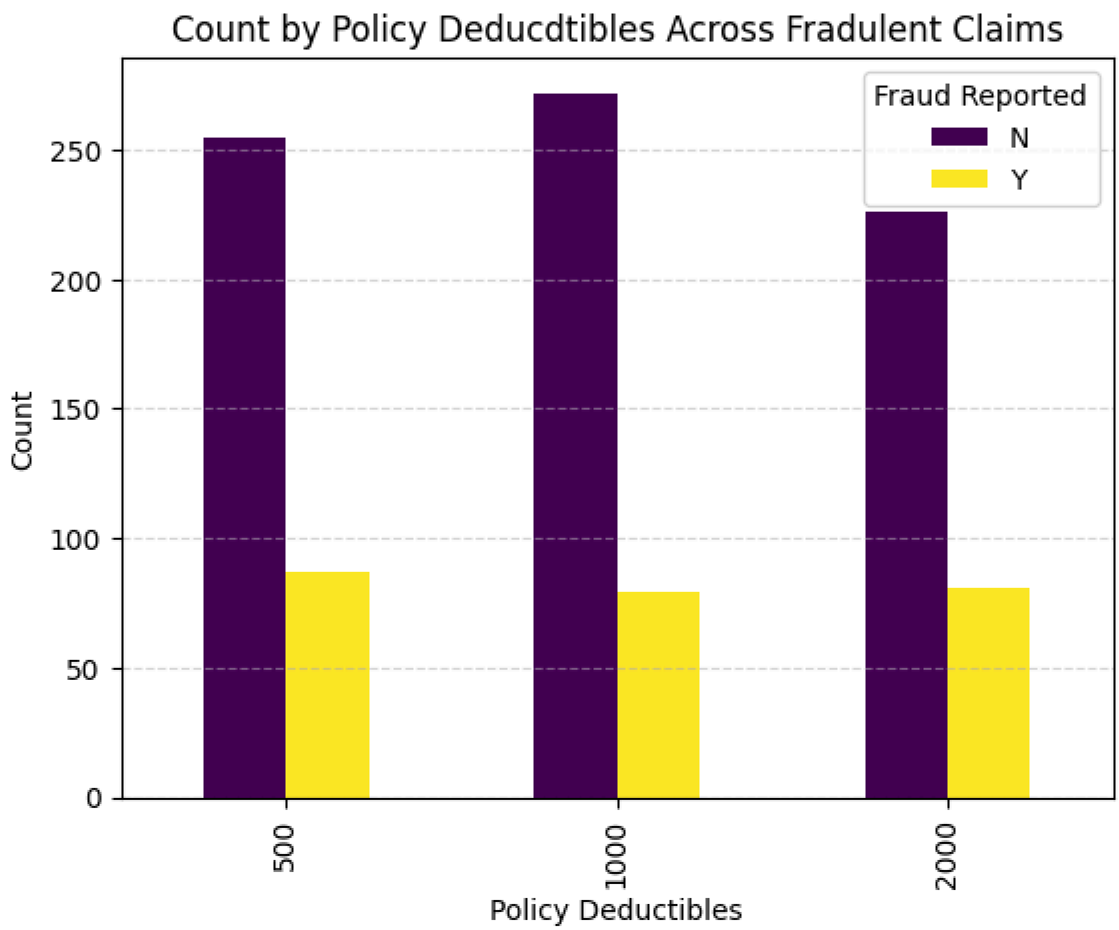
dtype: int64

In [ ]:
```
insurance_claims_df['fraud_reported'] = insurance_claims_df[
    'fraud_reported'].astype('category')
```

In [ ]:
```
# Group the data
deduc_counts = insurance_claims_df.groupby(['policy_deductable',
                                            'fraud_reported']).size().unstack(

# Plot grouped bar chart (stacked=False)
```

```
deduc_counts.plot(kind='bar', stacked=False, colormap='viridis')

# Add labels and legend
plt.title("Count by Policy Deducdtibles Across Fradulent Claims")
plt.xlabel("Policy Deductibles")
plt.ylabel("Count")
plt.legend(title="Fraud Reported")
plt.grid(True, axis='y', linestyle='--', alpha=0.5)

plt.show();
```

```
<ipython-input-22-77188418fe5b>:2: FutureWarning: The default of observed=Fals
e is deprecated and will be changed to True in a future version of pandas. Pas
s observed=False to retain current behavior or observed=True to adopt the futu
re default and silence this warning.
  deduc_counts = insurance_claims_df.groupby(['policy_deductable', 'fraud_repo
rted']).size().unstack()
```

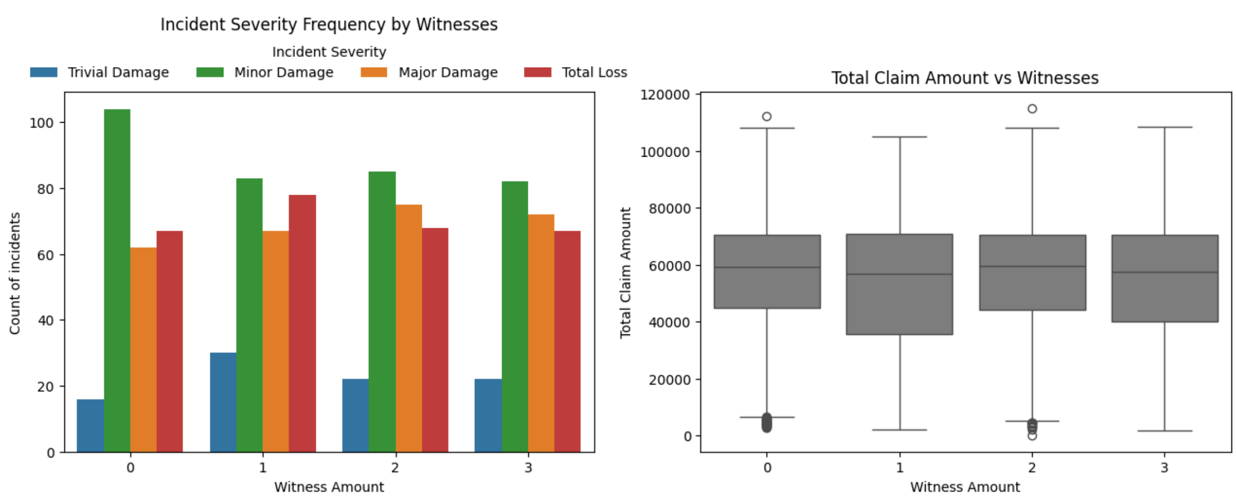


Orator: Jacob Dang

Q1: How does the number of witnesses impact the assessed severity and total claim amount of auto insurance claims?

In [ ]:
```
fig, ax = plt.subplots(1, 2)

# Create bar chart of frequencies by number of witnesses
palette = ['tab:blue', 'tab:green', 'tab:orange', 'tab:red']
```

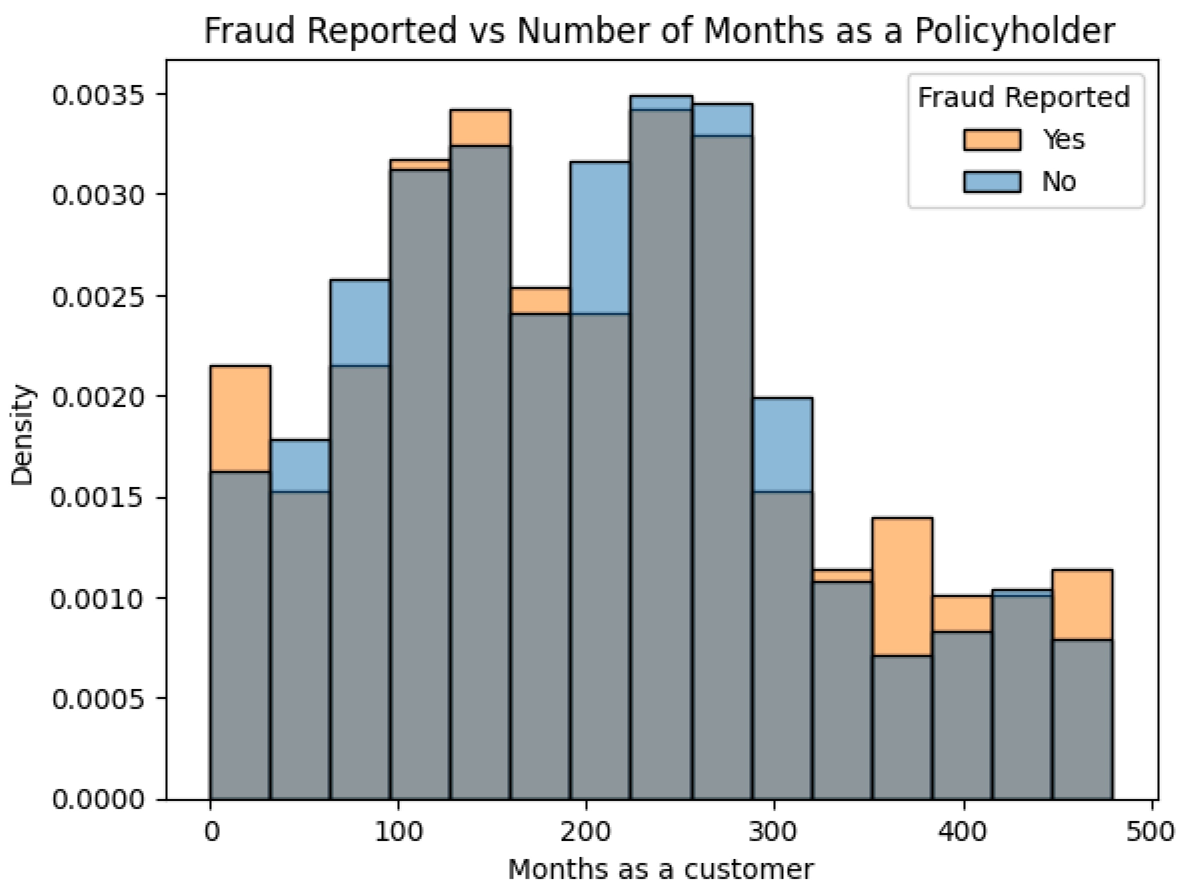
```
sns.countplot(insurance_claims_df, x='witnesses',
              hue='incident_severity',
              hue_order=['Trivial Damage', 'Minor Damage',
                         'Major Damage', 'Total Loss'],
              palette=palette,
              ax=ax[0])

# Move the legend to the top of the graph to avoid overlap
sns.move_legend(
    ax[0], "lower center",
    bbox_to_anchor=(.5, 1), ncol=4,
    title='Incident Severity', frameon=False,
)
# Set title and labels
ax[0].set_title('Incident Severity Frequency by Witnesses', y=1.15)
ax[0].set_xlabel('Witness Amount')
ax[0].set_ylabel('Count of incidents')

# Create boxplot of total claim amount vs witnesses
sns.boxplot(insurance_claims_df, x='witnesses',
            y='total_claim_amount',
            color='grey',ax=ax[1])

# Set title and labels
ax[1].set_title('Total Claim Amount vs Witnesses')
ax[1].set_xlabel('Witness Amount')
ax[1].set_ylabel('Total Claim Amount')

# Set spacing so subplots dont overlap
fig.subplots_adjust(right=2)

plt.show()
```



Q2: Are customers who have held their insurance policies for more months more or less likely to commit fraud than customers who have held their policy for less time?

In [ ]:
```
# Code here for Q2

# Create histogram for fraud vs months as a customer
sns.histplot(insurance_claims_df, x='months_as_customer',
             hue='fraud_reported',
```

```
            common_norm=False, bins=15, stat='density')

# Label and title graph
plt.title('Fraud Reported vs Number of Months as a Policyholder')
plt.xlabel('Months as a customer')
plt.ylabel('Density')
plt.legend( ['Yes', 'No'], title='Fraud Reported')

plt.show()
```



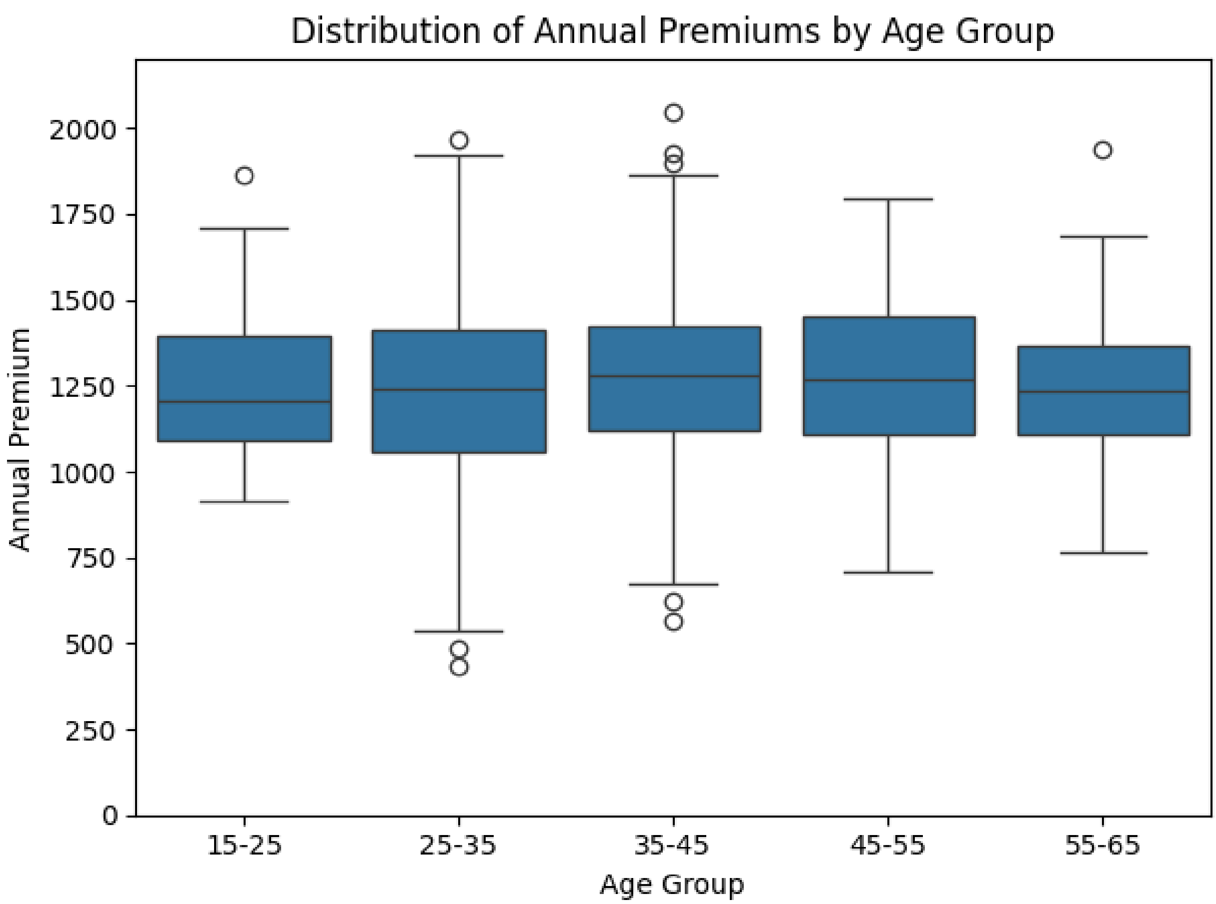
Orator: Yueen Ma

Q1: Are older customers more likely to pay a higher annual premium and higher deductibles?

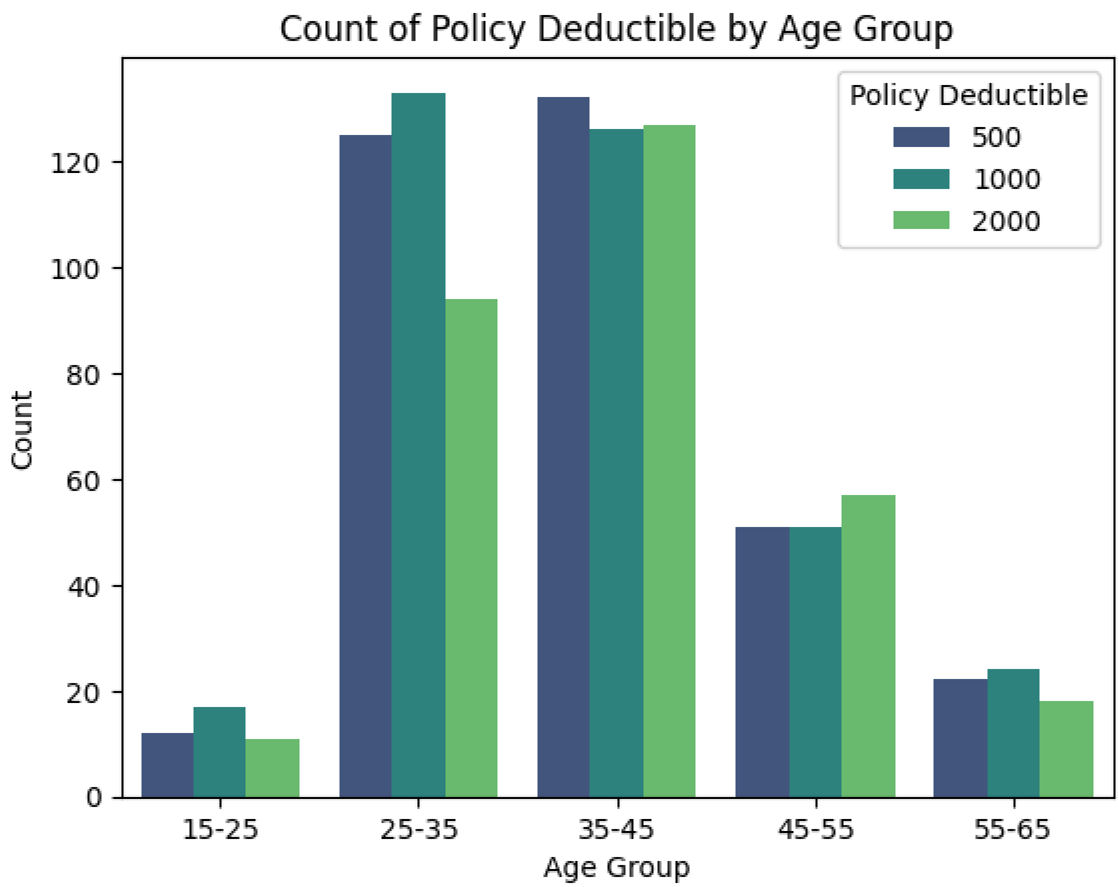
In [ ]:

```
# Code here for Q1

# Create new column for age groups
bins = [15, 25, 35, 45, 55, 65]
labels = ['15-25', '25-35', '35-45', '45-55', '55-65']

insurance_claims_df_copy = insurance_claims_df.copy()
insurance_claims_df_copy['age_group'] = pd.cut(
    insurance_claims_df_copy['age'], bins=bins, labels=labels)

# plot boxplot for annual premium vs age group
sns.boxplot(data=insurance_claims_df_copy, x='age_group',
            y='policy_annual_premium')
```

```python
plt.xlabel('Age Group')
plt.ylabel('Annual Premium')
plt.ylim(0, 2200)
plt.title('Distribution of Annual Premiums by Age Group')
plt.tight_layout()
plt.savefig('premium_age.png')
plt.show()



# count people in each policy deductable across age group
age_group_deductable_size = insurance_claims_df_copy.groupby(
    ['age_group', 'policy_deductable']).size().reset_index(name='count')

# plot bar graph for policy_deductable vs age group
sns.barplot(x='age_group', y='count', hue='policy_deductable',
            data=age_group_deductable_size, palette='viridis')
plt.xlabel('Age Group')
plt.ylabel('Count')
plt.title('Count of Policy Deductible by Age Group')
plt.legend(title="Policy Deductible")
plt.savefig('deductible_age.png')
plt.show()
```




```
<ipython-input-25-007c474ea995>:23: FutureWarning: The default of observed=Fal
se is deprecated and will be changed to True in a future version of pandas. Pa
ss observed=False to retain current behavior or observed=True to adopt the fut
ure default and silence this warning.
  age_group_deductable_size = insurance_claims_df_copy.groupby(['age_group',
'policy_deductable']).size().reset_index(name='count')
```

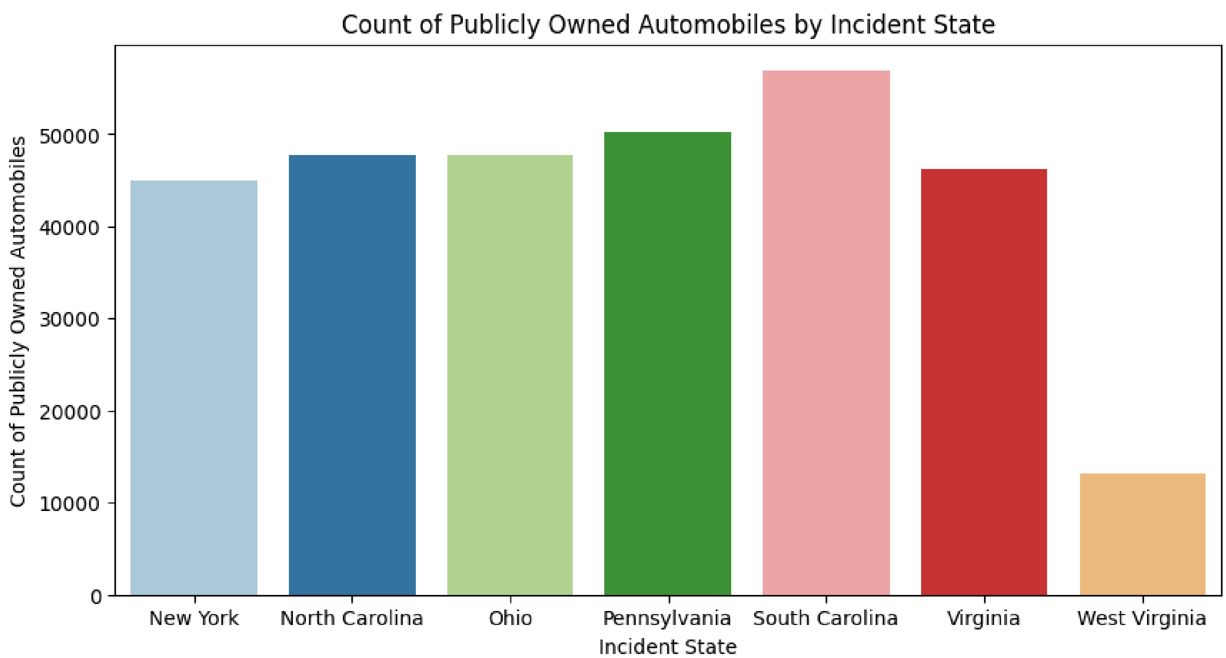


Q2: Is there a correlation between the number of publicly-owned automobiles in a state and the frequency or severity of vehicle claims filed by policyholders?

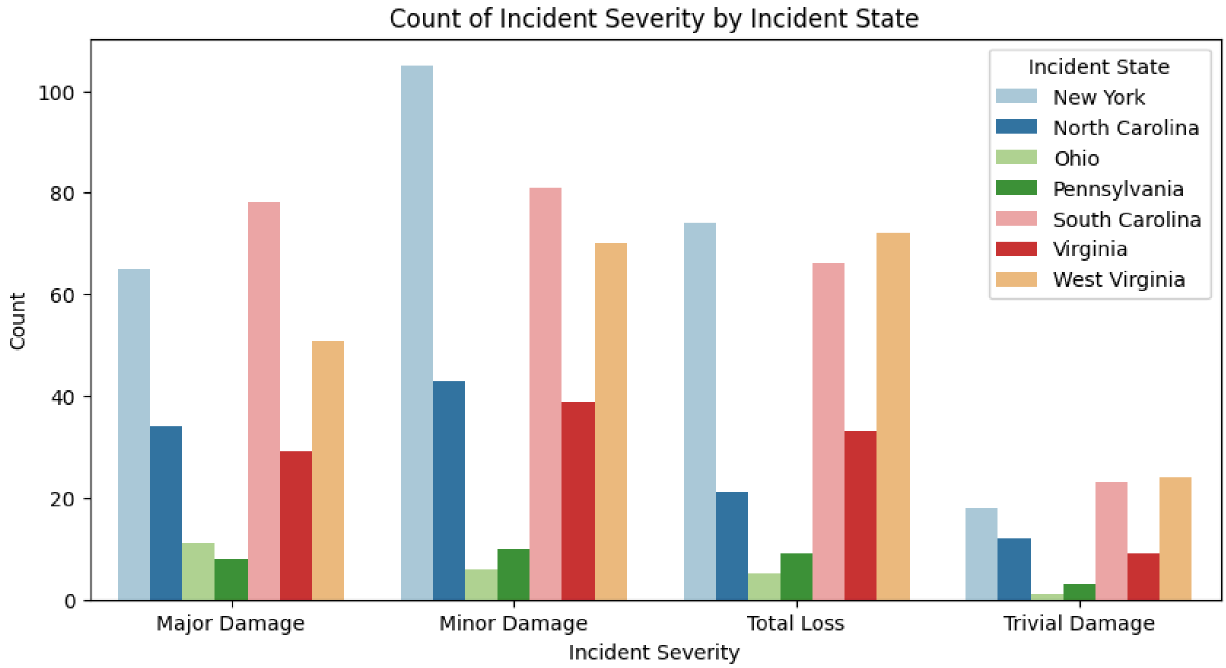
In [ ]:
```
# The incident state in insurance claims data only includes NY, SC, WV, VA, NC,
insurance_claims_df_copy = insurance_claims_df.copy()
insurance_state_value_counts = insurance_claims_df_copy[
    'incident_state'].value_counts()
states = ['New York', 'South Carolina', 'West Virginia',
          'Virginia', 'North Carolina', 'Pennsylvania', 'Ohio']

# get publicly-owned automobiles
publicly_owned_automobiles_df = all_motor_vehicles_df[[('STATE',
                                                         'STATE'),
                                                        ('AUTOMOBILES',
                                                         'PUBLICLY OWNED')]]

# filter publicly_owned_automobiles_df to the same states
filtered_publicly_owned_automobiles_df = publicly_owned_automobiles_df[
    publicly_owned_automobiles_df[('STATE', 'STATE')].isin(states)]

# rename the states
state_name = {'NY': 'New York', 'SC': 'South Carolina',
              'WV': 'West Virginia',
              'VA': 'Virginia', 'NC': 'North Carolina',
              'PA':'Pennsylvania', 'OH': 'Ohio'}
insurance_claims_df_copy['incident_state'] = insurance_claims_df_copy[
    'incident_state'].replace(state_name)
```

```
# plot number of publicly owned automobiles
plt.figure(figsize=(10, 5))
sns.barplot(x=('STATE', 'STATE'), y=('AUTOMOBILES', 'PUBLICLY OWNED'),
            data=filtered_publicly_owned_automobiles_df, palette='Paired')
plt.xlabel('Incident State')
plt.ylabel('Count of Publicly Owned Automobiles')
plt.title('Count of Publicly Owned Automobiles by Incident State')
plt.savefig('publicly_own_automobile_count.png')
plt.show()


# plot group incident severity by incident state
incident_severity_state_df = insurance_claims_df_copy.groupby(
    ['incident_severity', 'incident_state']
).size().reset_index(name='count')
plt.figure(figsize=(10, 5))
sns.barplot(x='incident_severity', y='count', hue='incident_state',
            data=incident_severity_state_df, palette='Paired')
plt.xlabel('Incident Severity')
plt.ylabel('Count')
plt.title('Count of Incident Severity by Incident State')
plt.legend(title='Incident State')
plt.savefig('incident_severity_count.png')
plt.show()
```

```
<ipython-input-26-5a0682f17fe6>:19: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x=('STATE', 'STATE'), y=('AUTOMOBILES', 'PUBLICLY OWNED'), data=
filtered_publicly_owned_automobiles_df, palette='Paired')
```



```
&lt;ipython-input-26-5a0682f17fe6&gt;:28: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
  incident_severity_state_df = insurance_claims_df_copy.groupby(['incident_severity', 'incident_state']).size().reset_index(name='count')
```



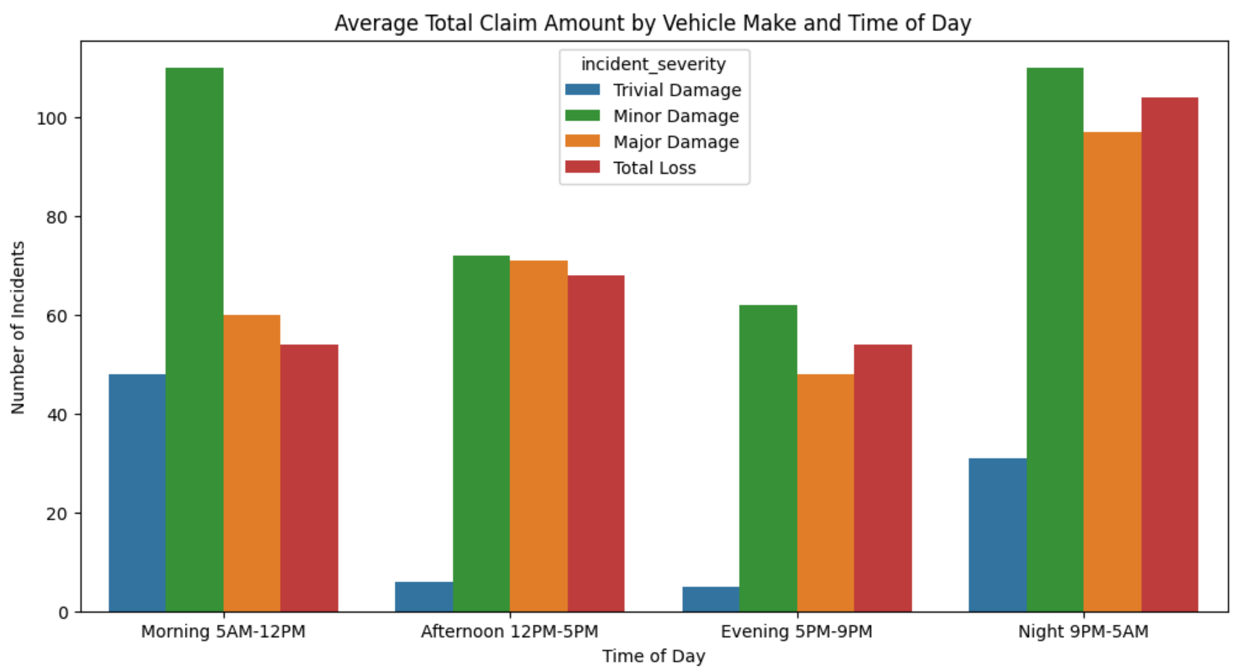
Deliverer: Eleni Kafexhiu

Q1: How do vehicle types and time of day impact the severity of auto insurance claims?

In [ ]:

```python
# Code here for Q1

# Ensure that claim amount column are numeric
insurance_claims_df['total_claim_amount'] = pd.to_numeric(
    insurance_claims_df['total_claim_amount'], errors='coerce')

# Define time of the day
def time_of_day(hour):
    if hour >= 5 and hour < 12:
        return 'Morning 5AM-12PM'
    elif hour >= 12 and hour < 17:
        return 'Afternoon 12PM-5PM'
    elif hour >= 17 and hour < 21:
        return 'Evening 5PM-9PM'
    else:
        return 'Night 9PM-5AM'

# Create new column for time of day
insurance_claims_df['time_of_day'] = insurance_claims_df[
    'incident_hour_of_the_day'].apply(time_of_day)

# Define the desired order for time_of_day
time_order = ['Morning 5AM-12PM', 'Afternoon 12PM-5PM',
```

```
                'Evening 5PM–9PM', 'Night 9PM–5AM']
insurance_claims_df['time_of_day'] = pd.Categorical(
    insurance_claims_df['time_of_day'],categories=time_order,
    ordered=True)

# Create barplot
plt.figure(figsize=(12, 6))
palette = ['tab:blue', 'tab:green', 'tab:orange', 'tab:red']
sns.countplot(insurance_claims_df, x='time_of_day',
              hue='incident_severity',
             hue_order=['Trivial Damage', 'Minor Damage',
                         'Major Damage', 'Total Loss'],
             palette=palette)
plt.title('Average Total Claim Amount by Vehicle Make and Time of Day')
plt.xlabel('Time of Day')
plt.ylabel('Number of Incidents')
plt.show()
```



Q2: How do policyholder tenure and deductible amount impact the total claim amount?

In [ ]:

```
# Code here for Q2

# Convert columns to numeric
insurance_claims_df['months_as_customers'] = pd.to_numeric(
    insurance_claims_df['months_as_customer'], errors = 'coerce')
insurance_claims_df['policy_deductable'] = pd.to_numeric(
    insurance_claims_df['policy_deductable'], errors = 'coerce')

# Create tenure bins
insurance_claims_df['tenure_group'] = pd.cut(
    insurance_claims_df['months_as_customer'],
    bins=[0, 100, 200, 300, 400,
          insurance_claims_df['months_as_customer'].max()],
    labels=['0–100', '101–200', '201–300', '301–400', '401+'])

# Create the boxplot
```

```
plt.figure(figsize=(14, 7))
sns.violinplot(data = insurance_claims_df, x = 'policy_deductable',
               y = 'total_claim_amount', hue = 'tenure_group',
               split = True, palette = 'Set3')
plt.title('Distribution of Claim Amounts by Deductible and Tenure')
plt.xlabel('Policy Deductible ($)')
plt.ylabel('Total Claim Amount ($)')
plt.legend(title = 'Tenure (in months)', bbox_to_anchor=(1.05, 1),
           loc='upper left')
plt.tight_layout()
plt.show()
```

Follow-up Questions

New Questions Based Off Initial Investigation

- Q1: Can we predict the total claim amount using policyholder tenure, deductible amount, number of witnesses, and time of day?
- Q2: Can we predict whether an incident will occur in a high-vehicle-registration state based on the number of witnesses, incident severity, and time of day?
- Q3: Can we predict if a customer will report a fraudulent claim using tenure, deductible, and incident severity?
- Q4: Can we predict whether an incident results in a total loss using time of day, number of witnesses, deductible amount, and age of customer?

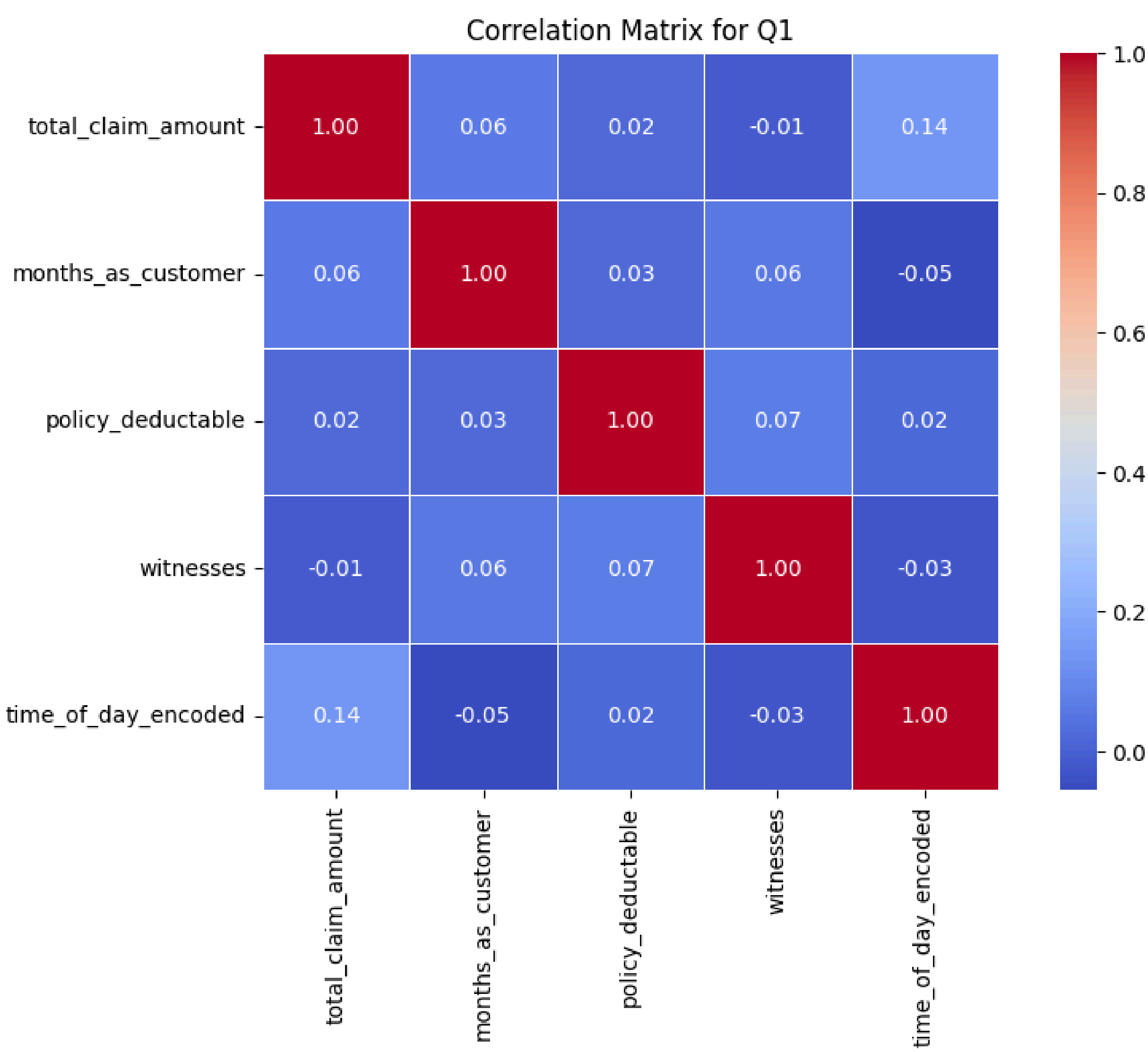
Investigation of Follow-up Questions

Our group decided to investigate Q1 and Q4 in further detail.

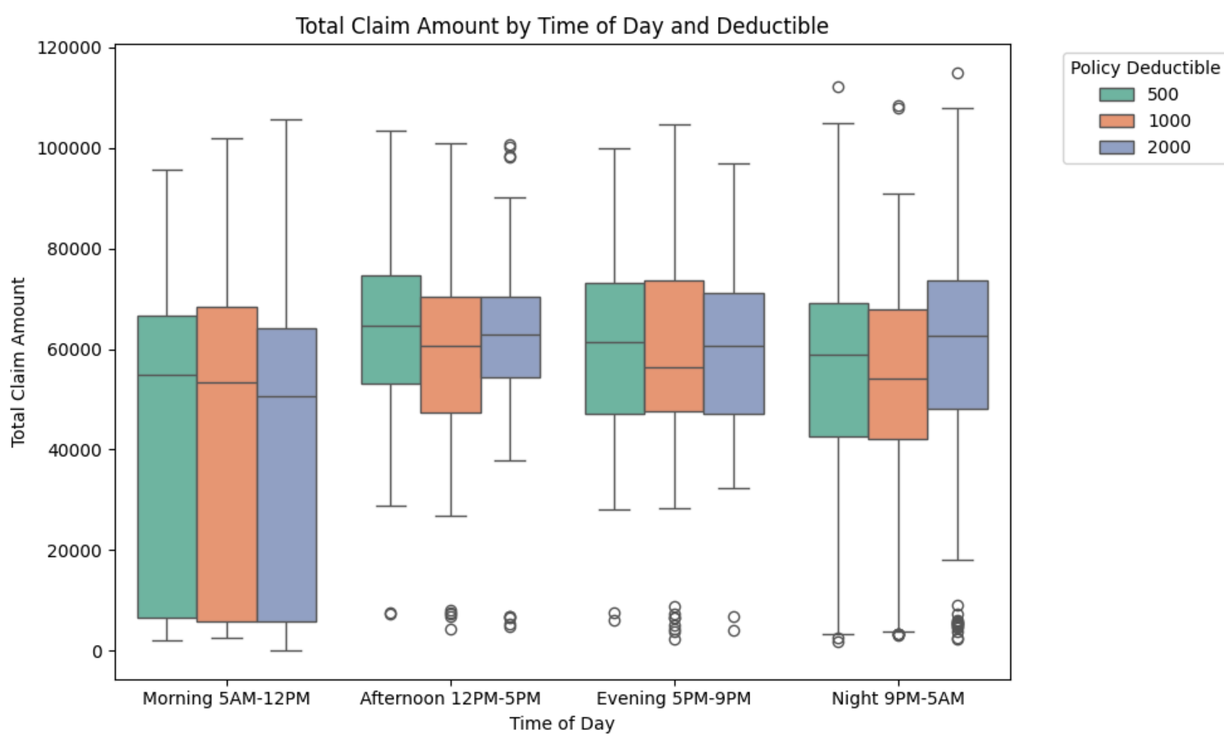
Question #1

In [ ]:

```python
# Map numbers to time of day so we can make a correlation matrix
insurance_claims_df['time_of_day_encoded'] = insurance_claims_df['time_of_day']
    'Morning 5AM–12PM': 1,
    'Afternoon 12PM–5PM': 2,
    'Evening 5PM–9PM': 3,
    'Night 9PM–5AM': 4
})


# Select relevant columns
corr_df = insurance_claims_df[[
    'total_claim_amount',
    'months_as_customer',
    'policy_deductable',
    'witnesses',
    'time_of_day_encoded'
]]

# Plot heatmap
corr = corr_df.corr()
plt.figure(figsize=(10, 6))
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt=".2f",
plt.title('Correlation Matrix for Q1')
plt.show();
```



In [ ]:
```python
plt.figure(figsize=(10, 6))

# Making boxplot
sns.boxplot(
    data=insurance_claims_df,
    x='time_of_day',
    y='total_claim_amount',
    hue='policy_deductable',
    order=['Morning 5AM-12PM',
           'Afternoon 12PM-5PM',
           'Evening 5PM-9PM',
           'Night 9PM-5AM'],
    palette='Set2'
)

# Plots with appropiate labels
plt.title('Total Claim Amount by Time of Day and Deductible')
plt.xlabel('Time of Day')
plt.ylabel('Total Claim Amount')
plt.legend(title='Policy Deductible',
           bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show();
```



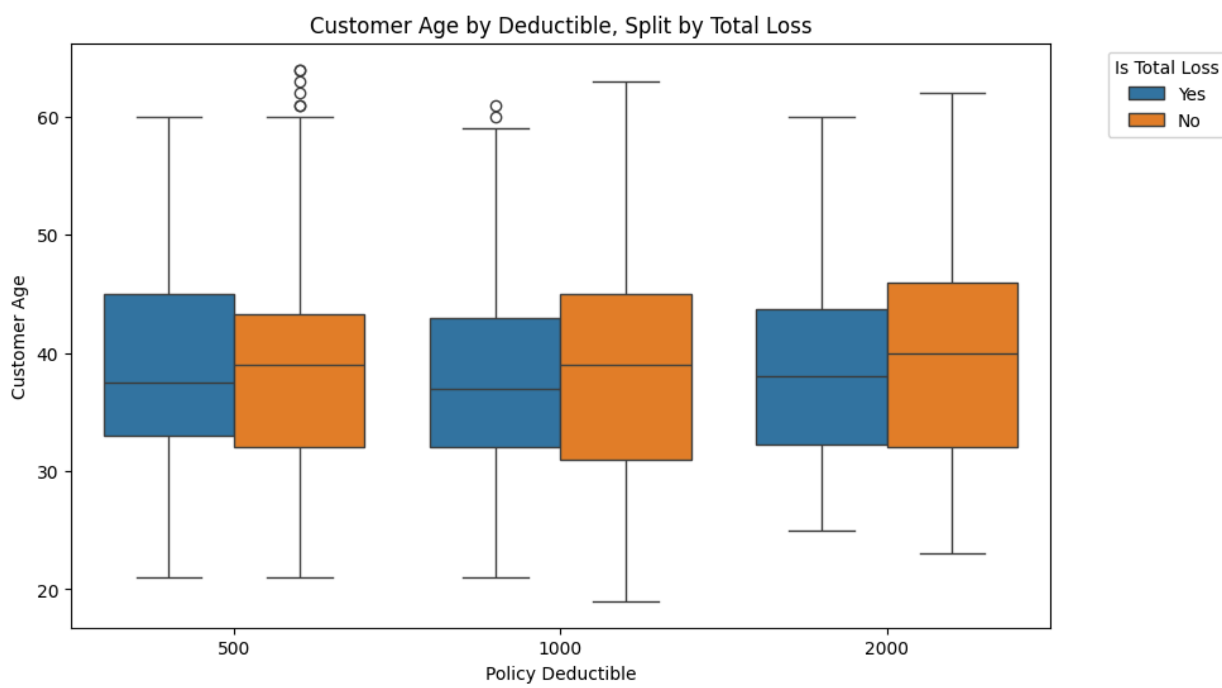
In [ ]:
```python
from statsmodels.stats.outliers_influence import variance_inflation_factor
X = insurance_claims_df[[
    'months_as_customer',
    'policy_deductable',
    'witnesses',
    'time_of_day_encoded'
]]

vif_data = pd.DataFrame()
vif_data['Variable'] = X.columns
vif_data['VIF'] = [variance_inflation_factor(
```

```
    X.values, i) for i in range(X.shape[1])]
print(vif_data)
```

```
               Variable       VIF
0    months_as_customer  3.206454
1     policy_deductable  3.535439
2             witnesses  2.530807
3   time_of_day_encoded  3.534854
```

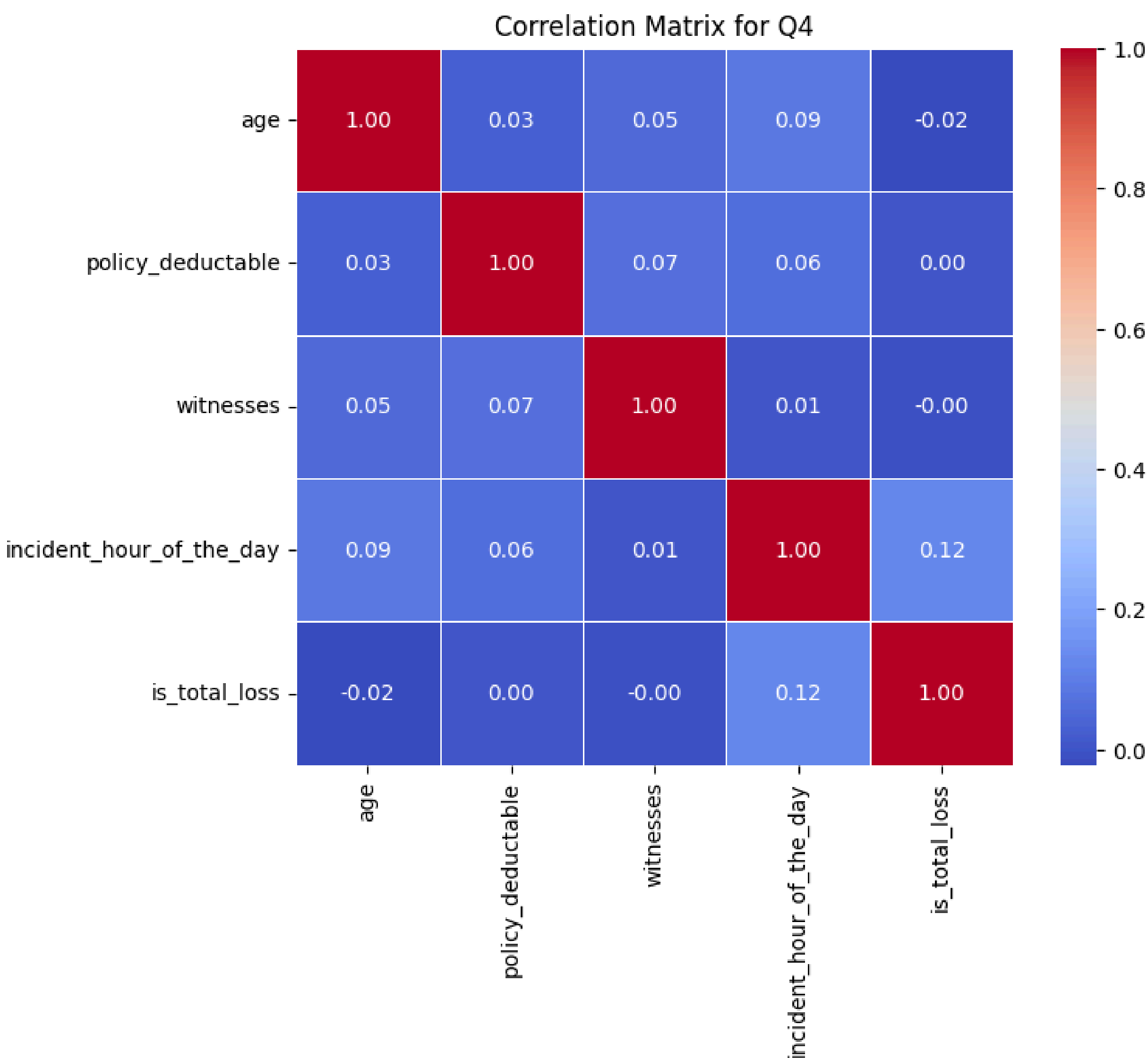
Question #4

In [6]:
```
# Convert boolean to string labels
insurance_claims_df['is_total_loss'] = insurance_claims_df[
    'incident_severity'] == 'Total Loss'
insurance_claims_df['loss_label'] = insurance_claims_df[
    'is_total_loss'].map({False: 'No', True: 'Yes'})

# Create figure
plt.figure(figsize=(10, 6))
sns.boxplot(
    data=insurance_claims_df,
    x='policy_deductable',
    y='age',
    hue='loss_label',
)

plt.title('Customer Age by Deductible, Split by Total Loss')
plt.xlabel('Policy Deductible')
plt.ylabel('Customer Age')
# The legend kept covering up the graphic so I used the bbox to anchor
plt.legend(title='Is Total Loss',
           bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show();
```



In [8]:
```
# Convert boolean to int for correlation
insurance_claims_df['is_total_loss'] = insurance_claims_df[
```

```
    'is_total_loss'].astype(int)


# Select relevant columns
corr_df = insurance_claims_df[[
    'age',
    'policy_deductable',
    'witnesses',
    'incident_hour_of_the_day',
    'is_total_loss',
]]


# Plot heatmap
corr = corr_df.corr()
plt.figure(figsize=(8, 6))
sns.heatmap(corr, annot=True, cmap='coolwarm',
            fmt=".2f", square=True, linewidths=0.5)
plt.title('Correlation Matrix for Q4')
plt.show();
```



In [ ]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
X = insurance_claims_df[[
    'age',
```

```
    'policy_deductable',
    'witnesses',
    'incident_hour_of_the_day',
    'is_total_loss',
]]

#Checking for multicollinearity
vif_data = pd.DataFrame()
vif_data['Variable'] = X.columns
vif_data['VIF'] = [variance_inflation_factor(
    X.values, i) for i in range(X.shape[1])]
print(vif_data)
```

```
                   Variable       VIF
0                       age  6.651078
1         policy_deductable  4.033210
2                 witnesses  2.697148
3  incident_hour_of_the_day  3.710088
4             is_total_loss  1.392873
```

In [ ]:

```
from scipy import stats
insurance_claims_df['is_total_loss'] = insurance_claims_df[
    'is_total_loss'].astype(int)

features = ['age', 'policy_deductable',
            'witnesses', 'incident_hour_of_the_day']

fstats = []
pvals = []
for feature in features:
    group1 = insurance_claims_df.loc[
        insurance_claims_df['is_total_loss'] == 1][feature]
    group2 = insurance_claims_df.loc[
        insurance_claims_df['is_total_loss'] == 0][feature]
    f_statistic, p_value = stats.f_oneway(group1, group2)
    fstats.append(f_statistic)
    pvals.append(p_value)
f_table = pd.DataFrame({'Feature': features,
                        'F Statistic': fstats, 'p-value': pvals})
print('F-test results on features grouped by occurrance of total loss')
f_table
```

```
F-test results on features grouped by occurrance of total loss
```

Out[ ]:

	Feature	F Statistic	p-value
0	age	0.496184	0.481346
1	policy_deductable	0.011203	0.915726
2	witnesses	0.007421	0.931368
3	incident_hour_of_the_day	14.845471	0.000124

Summary

GIVE A 2 PARAGRAPH SUMMARY.

Through our investigation of the initial questions, we identified several key patterns in the data, which helped us refine the focus of our project. One significant change was our decision to move away from analyzing vehicle types after receiving feedback that 'vehicle type' is too broad to yield meaningful insights. We lacked information about the car's age, class, or condition. One key finding was that minor damage incidents occurred most frequently in the morning, likely due to commuting. In contrast, total loss incidents spiked at night, possibly due to factors such as fatigue or low visibility. We also discovered that claim amounts were relatively consistent across different deductible levels, but longer-tenured customers exhibited more variability. This suggests that tenure has a stronger correlation with claim behavior than deductible amounts. Another notable insight was that fraudulent claims were evenly distributed across deductible levels, while older customers tended to have higher average claim amounts. Additionally, a histogram comparing high-registration and low-registration states revealed that high-registration states experienced more claims during peak hours such as rush hour, whereas low-registration states exhibited more balanced claim patterns. All these insights have guided us in forming more targeted follow-up questions, focused on predicting outcomes such as fraud, total loss, or high claim amounts. This will provide us with a clearer path for assessing real-world insurance risk.

We explored several predictive relationships in the dataset in our follow-up questions. The first question focuses on policyholder tenure, deductible amount, number of witnesses, and time of day. Visualizations from Q1, Q2, and Q9 suggest that variations in total claim amount can be partially explained by these factors. Based on the correlation matrix, we see that there is not a strong linear relationship between total claim amount and other variables (i.e, months as customer, policy deductible, witnesses, time of day), but their relationship can be nonlinear. The variance inflation factor of these variables also indicate no serious multicollinearity among predictors. Our second follow-up question investigates whether the number of witnesses, incident severity, and time of day can help explain whether an incident occurs in a state with high vehicle registration. It is motivated by the insights we found in Q3 that states with high registration seem to have more incidents during peak hours, in addition to the other two variables. The third question focuses on variables such as tenure, deductible, and incident severity to help predict whether a customer will report fraudulent claims. Even though we found deductibles to be not a strong predictor of fraudulent claims by itself, we thought pairing it with customer tenure and incident severity might have brought out some patterns. Furthermore, we found that customers who have insurance for a longer time also tend to submit fraudulent claims in Q10. Our last question investigates whether a total-loss incident occurs is predictable by the time of day, number of witnesses, deductible amount, and age of customers. Because we noticed that incident severity varies by time of day, with total-loss incidents spiking at night. Q7 visualization reveals that the 25-35 age group is less likely to choose the highest deductible. Although Q9 shows total loss remains stable across witness counts, combining witness counts with other variables may better explain its variation. The boxplot shows that customers with total loss are slightly younger, suggesting that younger drivers may be more prone to an incident with total loss regardless of deductible amount. The correlation matrix reveals weak relationships, with incident hour showing the highest. There is no serious multicollinearity found between these variables, though age and policy deductible are relatively higher. Though it lacks a clear linear pattern, combining variables may still reveal meaningful results.