```
import pandas as pd
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt
import seaborn as sns

#1. Import claims_data.csv and cust_data.csv which is provided to you and combine the two datasets
#appropriately to create a 360-degree view of the data. Use the same for the subsequent questions.

# Load the dataset
claims_data = pd.read_csv("claims.csv")
cust_data = pd.read_csv("cust_demographics.csv")

df = claims_data.merge(cust_data, left_on="customer_id", right_on="CUST_ID", how="inner")

df.head(5)
```

```
   claim_id  customer_id      incident_cause  claim_date claim_area  \
0  54004764     21868593        Driver error  11/27/2017       Auto
1  33985796     75740424               Crime  10/03/2018       Home
2  53522022     30308357  Other driver error  02/02/2018       Auto
3  63017412     30308357        Driver error  04/04/2018       Auto
4  13015401     47830476      Natural causes  06/17/2018       Auto

  police_report     claim_type claim_amount  total_policy_claims fraudulent  \
0            No  Material only        $2980                  1.0         No
1       Unknown  Material only        $2980                  3.0         No
2            No  Material only      $3369.5                  1.0        Yes
3            No  Material only        $1950                  6.0         No
4            No  Material only        $1680                  1.0         No

    CUST_ID  gender DateOfBirth State       Contact   Segment
0  21868593  Female   12-Jan-79    VT  789-916-8172  Platinum
1  75740424  Female   13-Jan-70    ME  265-543-1264    Silver
2  30308357  Female   11-Mar-84    TN  798-631-4758    Silver
3  30308357  Female   11-Mar-84    TN  798-631-4758    Silver
4  47830476  Female   01-May-86    MA  413-187-7945    Silver
```

```
#2. Perform a data audit for the datatypes and find out if there are any mismatch within the current datatypes
#of the columns and their business significance.

# A2: Perform a data audit
print("\nData Types and Missing Values:")
print(df.info())
print("\nMissing Values Count:")
print(df.isnull().sum())
```

```
Data Types and Missing Values:
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1085 entries, 0 to 1084
Data columns (total 16 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
```

```
 0   claim_id             1085 non-null   int64
 1   customer_id          1085 non-null   int64
 2   incident_cause       1085 non-null   object
 3   claim_date           1085 non-null   object
 4   claim_area           1085 non-null   object
 5   police_report        1085 non-null   object
 6   claim_type           1085 non-null   object
 7   claim_amount         1020 non-null   object
 8   total_policy_claims  1075 non-null   float64
 9   fraudulent           1085 non-null   object
 10  CUST_ID              1085 non-null   int64
 11  gender               1085 non-null   object
 12  DateOfBirth          1085 non-null   object
 13  State                1085 non-null   object
 14  Contact              1085 non-null   object
 15  Segment              1085 non-null   object
dtypes: float64(1), int64(3), object(12)
memory usage: 144.1+ KB
None

Missing Values Count:
claim_id                0
customer_id             0
incident_cause          0
claim_date              0
claim_area              0
police_report           0
claim_type              0
claim_amount           65
total_policy_claims    10
fraudulent              0
CUST_ID                 0
gender                  0
DateOfBirth             0
State                   0
Contact                 0
Segment                 0
dtype: int64
```

```
#Another view
# A2: Perform a data audit
data_audit = pd.DataFrame({
    "Column Name": df.columns,
    "Data Type": df.dtypes.values,
    "Missing Values": df.isnull().sum().values,
    "Unique Values": df.nunique().values
})

print("\nData Audit Table:")
print(data_audit)
```

```
Data Audit Table:
                Column Name Data Type  Missing Values  Unique Values
0                  claim_id     int64               0           1085
1               customer_id     int64               0           1078
2            incident_cause    object               0              5
```

```
3                claim_date    object                0            100
4                claim_area    object                0              2
5             police_report    object                0              3
6                claim_type    object                0              3
7              claim_amount   float64               65            676
8        total_policy_claims   float64               10              8
9                fraudulent    object                0              2
10                  CUST_ID     int64                0           1078
11                   gender    object                0              2
12              DateOfBirth    object                0           1078
13                    State    object                0             50
14                  Contact    object                0           1078
15                  Segment    object                0              3
16  injury_claim_unreported     int64                0              2

#3. Convert the column claim_amount to numeric. Use the appropriate
modules/attributes to remove the $ sign.
# A3: Convert claim_amount to numeric
if 'claim_amount' in df.columns:
    df['claim_amount'] = df['claim_amount'].replace('[\$,]', '',
regex=True).astype(float)

#4. Of all the injury claims, some of them have gone unreported with the police.
#Create an alert flag (1,0) for all such claims.

# a4: Create alert flag for unreported injury claims
df['injury_claim_unreported'] = np.where(df['police_report'] == 'No', 0, 1)


selected_columns = ['claim_id','customer_id', 'claim_amount', 'incident_cause',
'police_report',
                    'injury_claim_unreported']
df[selected_columns].head(5)

   claim_id  customer_id  claim_amount      incident_cause police_report  \
0  54004764     21868593        2980.0        Driver error            No
1  33985796     75740424        2980.0               Crime       Unknown
2  53522022     30308357        3369.5  Other driver error            No
3  63017412     30308357        1950.0        Driver error            No
4  13015401     47830476        1680.0      Natural causes            No

   injury_claim_unreported
0                        0
1                        1
2                        0
3                        0
4                        0

#5. One customer can claim for insurance more than once and in each claim, multiple
categories
#of claims can be involved. However, customer ID should remain unique.
#Retain the most recent observation and delete any duplicated records in the data
based on the customer ID column.

# A5: Retain most recent claim and remove duplicates
df['claim_date'] = pd.to_datetime(df['claim_date'])  # Convert to datetime
df = df.sort_values(by='claim_date',
```

```
ascending=False).drop_duplicates(subset='customer_id', keep='first')
df.head()
```

```
      claim_id  customer_id  incident_cause claim_date claim_area  \
226   49735712     17682060           Crime 2018-10-30       Home
1055  43042986     58451506  Natural causes 2018-10-30       Auto
1077  91386759     65208809  Natural causes 2018-10-30       Auto
354   98795403     38011078           Crime 2018-10-30       Auto
751   25213498     28932340    Driver error 2018-10-30       Auto

     police_report           claim_type  claim_amount  total_policy_claims  \
226        Unknown  Material and injury       17020.0                  1.0
1055            No        Material only        2420.0                  1.0
1077            No        Material only        2290.0                  1.0
354        Unknown        Material only        1940.0                  1.0
751        Unknown        Material only           NaN                  1.0

     fraudulent   CUST_ID  gender DateOfBirth State       Contact   Segment  \
226          No  17682060  Female   21-Nov-74    NV  186-195-3465      Gold
1055         No  58451506    Male   22-Apr-68    FL  673-574-7823      Gold
1077         No  65208809    Male   22-Apr-64    VA  286-128-6132  Platinum
354          No  38011078  Female   20-May-76    NE  271-123-1475      Gold
751          No  28932340    Male   05-Jan-96    LA  652-265-8231      Gold

      injury_claim_unreported
226                         1
1055                        0
1077                        0
354                         1
751                         1
```

```
#6. Check for missing values and impute the missing values with an appropriate
value.
#(mean for continuous and mode for categorical)

# A6: Handle missing values
df.fillna({col: df[col].mean() if df[col].dtype in ['float64', 'int64']
           else df[col].mode()[0] for col in df.columns}, inplace=True)

df.head()
```

```
      claim_id  customer_id  incident_cause claim_date claim_area  \
226   49735712     17682060           Crime 2018-10-30       Home
1055  43042986     58451506  Natural causes 2018-10-30       Auto
1077  91386759     65208809  Natural causes 2018-10-30       Auto
354   98795403     38011078           Crime 2018-10-30       Auto
751   25213498     28932340    Driver error 2018-10-30       Auto

     police_report           claim_type  claim_amount  total_policy_claims  \
226        Unknown  Material and injury  17020.000000                  1.0
1055            No        Material only   2420.000000                  1.0
1077            No        Material only   2290.000000                  1.0
354        Unknown        Material only   1940.000000                  1.0
751        Unknown        Material only  12480.933366                  1.0

     fraudulent   CUST_ID  gender DateOfBirth State       Contact   Segment  \
226          No  17682060  Female   21-Nov-74    NV  186-195-3465      Gold
1055         No  58451506    Male   22-Apr-68    FL  673-574-7823      Gold
```

```
1077          No  65208809    Male   22-Apr-64    VA  286-128-6132  Platinum
354           No  38011078  Female   20-May-76    NE  271-123-1475      Gold
751           No  28932340    Male   05-Jan-96    LA  652-265-8231      Gold

      injury_claim_unreported
226                         1
1055                        0
1077                        0
354                         1
751                         1
```

```
#7. Calculate the age of customers in years. Based on the age, categorize the
customers according to the
#below criteria
#Children < 18
#Youth 18-30
#Adult 30-60
#Senior > 60

# A7: Calculate age and categorize
from datetime import datetime

df['dob'] = pd.to_datetime(df['dob'], format='%d-%b-%y', errors='coerce')
df.loc[df['dob'].dt.year > datetime.now().year, 'dob'] -= pd.DateOffset(years=100)
df['age'] = (pd.to_datetime("today") - df['dob']).dt.days // 365
df.loc[df['age'] < 0, 'age'] = np.nan
df['age_category'] = pd.cut(df['age'], bins=[0, 18, 30, 60, 120],
labels=['Children', 'Youth', 'Adult', 'Senior'])

df.head()
```

```
      claim_id  customer_id  incident_cause claim_date claim_area  \
226   49735712     17682060           Crime 2018-10-30       Home
1055  43042986     58451506  Natural causes 2018-10-30       Auto
1077  91386759     65208809  Natural causes 2018-10-30       Auto
354   98795403     38011078           Crime 2018-10-30       Auto
751   25213498     28932340    Driver error 2018-10-30       Auto

     police_report           claim_type  claim_amount  total_policy_claims  \
226        Unknown  Material and injury  17020.000000                  1.0
1055            No        Material only   2420.000000                  1.0
1077            No        Material only   2290.000000                  1.0
354        Unknown        Material only   1940.000000                  1.0
751        Unknown        Material only  12480.933366                  1.0

     fraudulent   CUST_ID  gender DateOfBirth State       Contact   Segment  \
226          No  17682060  Female   21-Nov-74    NV  186-195-3465      Gold
1055         No  58451506    Male   22-Apr-68    FL  673-574-7823      Gold
1077         No  65208809    Male   22-Apr-64    VA  286-128-6132  Platinum
354          No  38011078  Female   20-May-76    NE  271-123-1475      Gold
751          No  28932340    Male   05-Jan-96    LA  652-265-8231      Gold

      injury_claim_unreported   age age_category        dob
226                         1  50.0        Adult 1974-11-21
1055                        0  56.0        Adult 1968-04-22
1077                        0  60.0        Adult 1964-04-22
354                         1  48.0        Adult 1976-05-20
751                         1  29.0        Youth 1996-01-05
```

```
# 8. What is the average amount claimed by the customers from various segments?

# A8: Average claim amount by segment
avg_claim_by_segment = df.groupby('Segment')['claim_amount'].mean()
print("\nAverage Claim Amount by Segment:")
print(avg_claim_by_segment)
```

```
Average Claim Amount by Segment:
Segment
Gold        12788.392275
Platinum    12370.790860
Silver      12266.176689
Name: claim_amount, dtype: float64
```

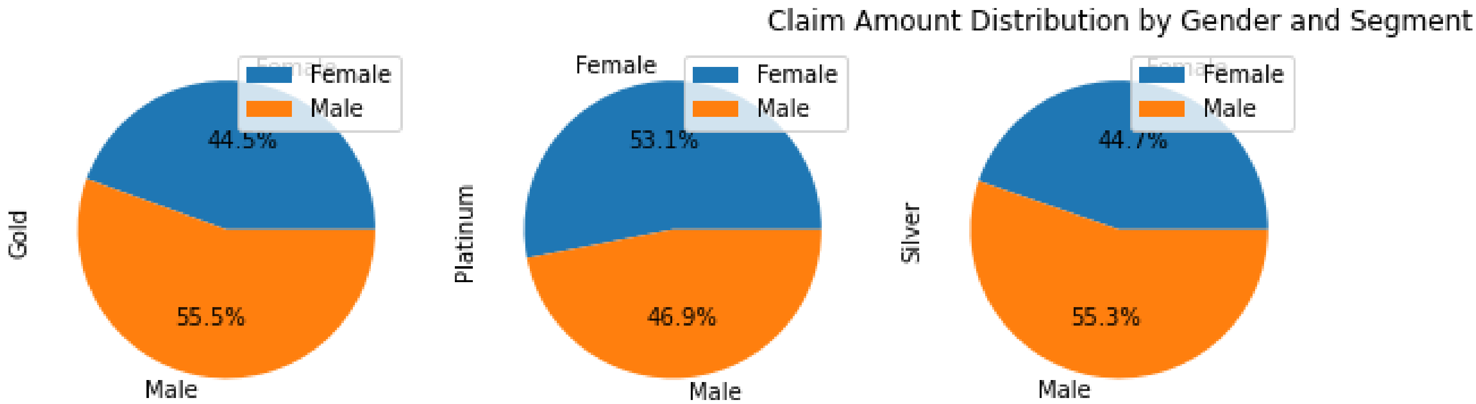
```
#9. What is the total claim amount based on incident cause for all the claims that
have been done at least 20 days
#prior to 1st of October, 2018.

# A9: Total claim amount for incidents before a given date
cutoff_date = pd.to_datetime('2018-09-10')
total_claim_prior = df[df['claim_date'] < cutoff_date].groupby('incident_cause')
['claim_amount'].sum()
print("\nTotal Claim Amount by Incident Cause before October 2018:")
print(total_claim_prior.apply(lambda x: f"{x:,.2f}"))
```

```
Total Claim Amount by Incident Cause before October 2018:
incident_cause
Crime                     721,834.67
Driver error            3,278,791.20
Natural causes          1,312,799.90
Other causes            3,725,236.73
Other driver error      3,315,572.63
Name: claim_amount, dtype: object
```

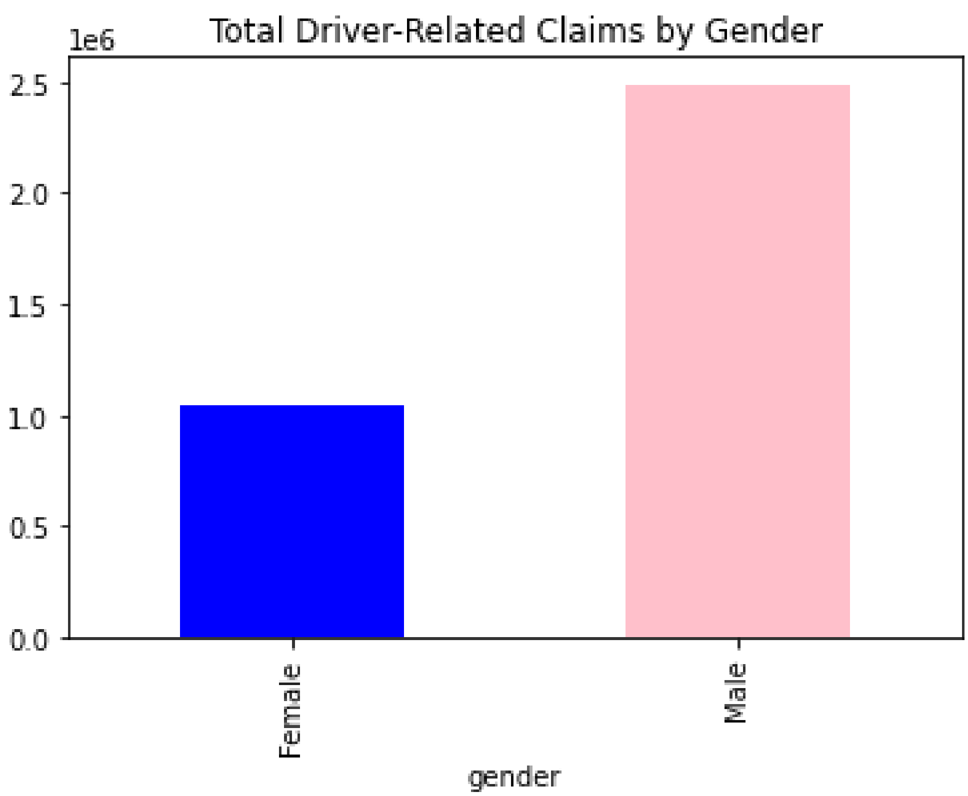
```
#10. How many adults from TX, DE and AK claimed insurance for driver
#related issues and causes?

# A10: Adult claims from TX, DE, AK for driver-related issues
states_filter = df['State'].isin(['TX', 'DE', 'AK'])
adult_driver_claims = df[(df['age_category'] == 'Adult') & states_filter &
                         (df['incident_cause'] == 'Driver error')].shape[0]
print("\nNumber of Adults Claiming for Driver Issues in TX, DE, AK:",
adult_driver_claims)
```

```
Number of Adults Claiming for Driver Issues in TX, DE, AK: 15
```

```
#11. Draw a pie chart between the aggregated value of claim amount based
#on gender and segment. Represent the claim amount as a percentage on the pie chart.

# A11: Pie Chart for Claim Amount by Gender & Segment
claim_pivot = df.groupby(['gender', 'Segment'])['claim_amount'].sum().unstack()
claim_pivot.plot(kind='pie', subplots=True, autopct='%1.1f%%', figsize=(10, 5))
plt.title("Claim Amount Distribution by Gender and Segment")
plt.show()
```

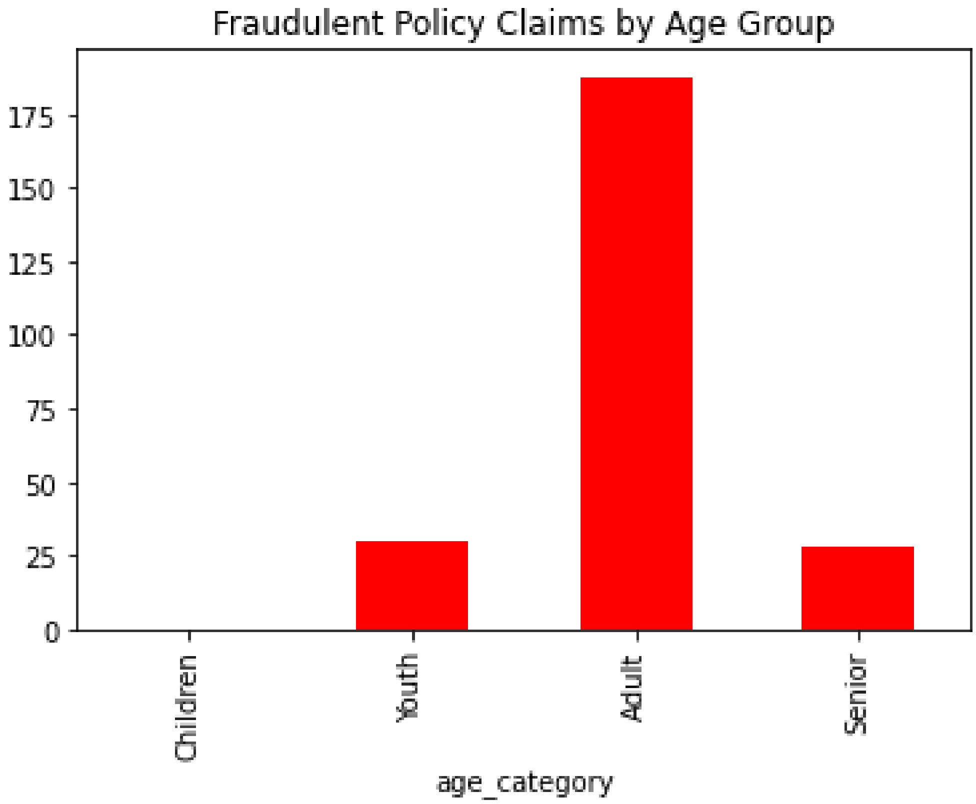



```
#12. Among males and females, which gender had claimed the most for any type of
driver related issues?
#E.g. This metric can be compared using a bar chart

# A12: Bar chart - Driver-related claims by gender
driver_claims_by_gender = df[df['incident_cause'] == 'Driver
error'].groupby('gender')['claim_amount'].sum()
if driver_claims_by_gender.empty:
    print("No driver-related claims found for any gender.")
else:
    driver_claims_by_gender.plot(kind='bar', color=['blue', 'pink'])
    plt.title("Total Driver-Related Claims by Gender")
    plt.show()
```

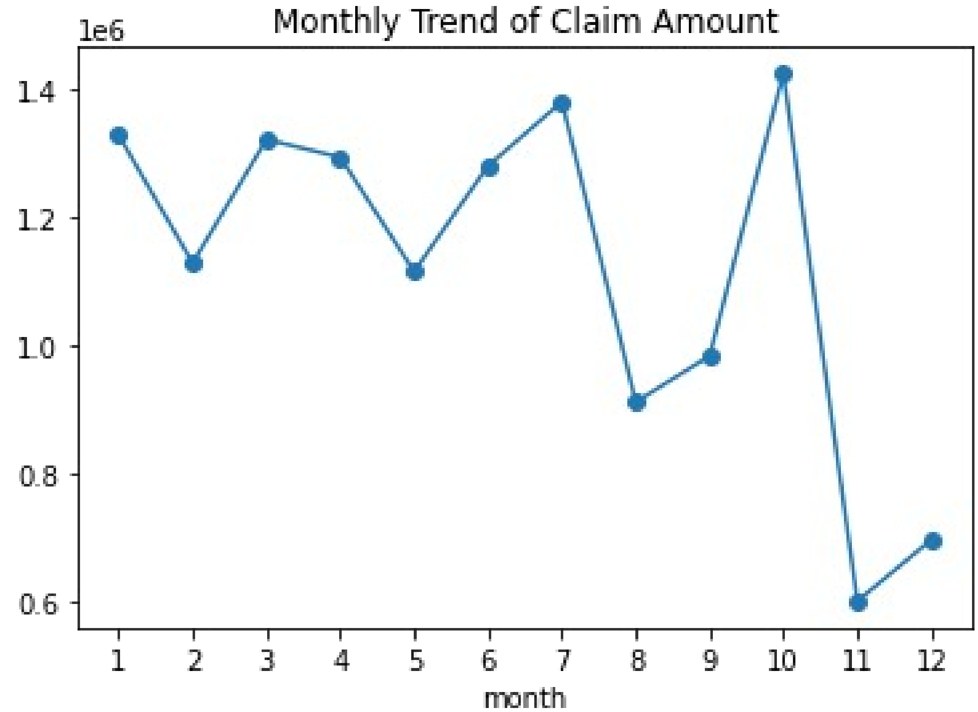



```
#Q13 Which age group had the maximum fraudulent policy claims? Visualize it on a bar
chart.

# A13: Fraudulent claims by age group
```

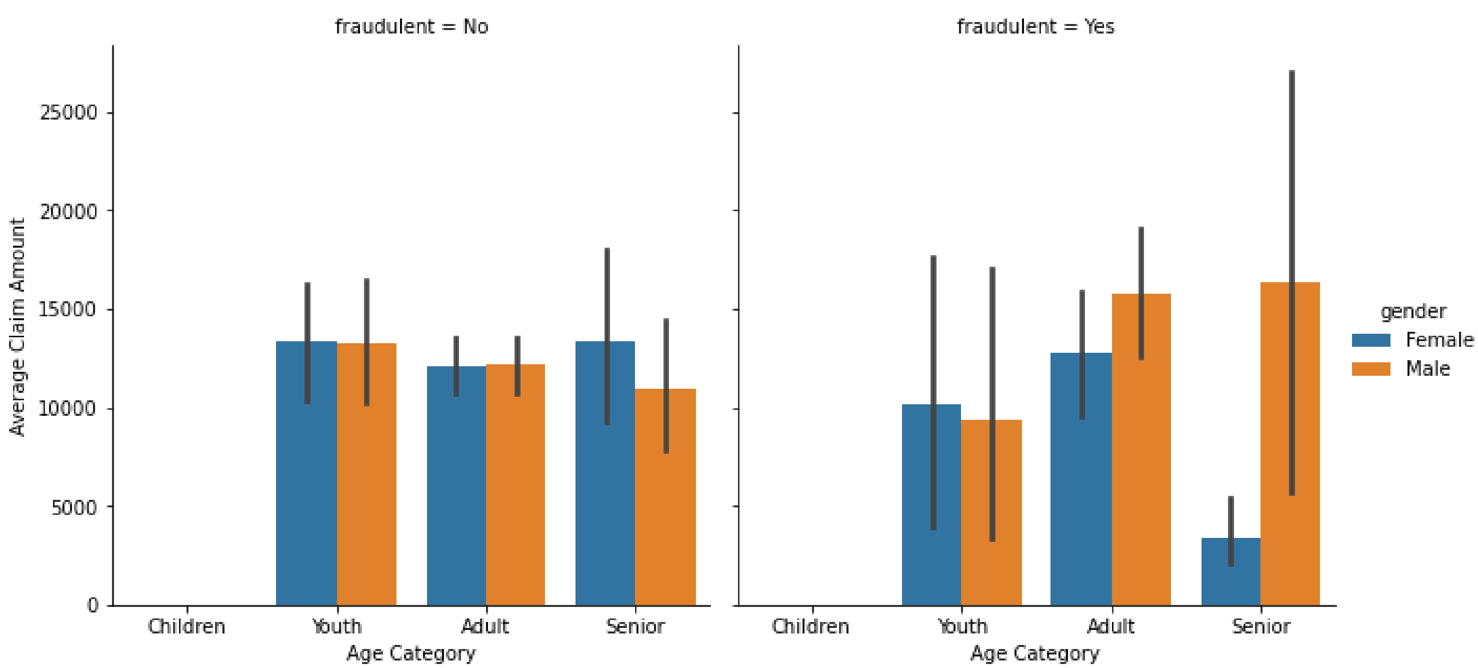
```
fraud_by_age = df[df['fraudulent'] == 'Yes'].groupby('age_category').size()
fraud_by_age.plot(kind='bar', color='red')
plt.title("Fraudulent Policy Claims by Age Group")
plt.show()
```




```
#Q14 Visualize the monthly trend of the total amount that has been claimed by the customers.
#Ensure that on the “month” axis, the month is in a chronological order not alphabetical order.

# A14: Monthly trend of claims
df['month'] = df['claim_date'].dt.month
df.groupby('month')['claim_amount'].sum().plot(kind='line', marker='o')
plt.xticks(range(1, 13))
plt.title("Monthly Trend of Claim Amount")
plt.show()
```




```
#Q15 What is the average claim amount for gender and age categories and suitably
represent the above using
#a facetted bar chart, one facet that represents fraudulent claims and the other for
non-fraudulent claims.

# A15: Facet grid - Claim amount by gender, age category, fraud status
g = sns.catplot(data=df, x='age_category', y='claim_amount', hue='gender',
col='fraudulent', kind='bar')
g.set_axis_labels("Age Category", "Average Claim Amount")
plt.show()
```




```
#Based on the conclusions from exploratory analysis as well as suitable statistical
tests, answer the
```

```python
#below questions. Please include a detailed write-up on the parameters taken into consideration, the Hypothesis
#testing steps, conclusion from the p-values and the business implications of the statements.

# Hypothesis Testing
from scipy.stats import ttest_ind, chi2_contingency

#16 Is there any similarity in the amount claimed by males and females?
# a16: Test similarity in claim amounts between genders
male_claims = df[df['gender'] == 'Male']['claim_amount']
female_claims = df[df['gender'] == 'Female']['claim_amount']
t_stat, p_value = ttest_ind(male_claims, female_claims)
print("\nT-Test for Claim Amounts by Gender: p-value =", p_value)
```

```
T-Test for Claim Amounts by Gender: p-value = 0.3602836601081929
```

```python
#17 Is there any relationship between age category and segment?

# A17: Relationship between age category and segment
age_seg_contingency = pd.crosstab(df['age_category'], df['Segment'])
chi2, p, dof, expected = chi2_contingency(age_seg_contingency)
print("\nChi-Square Test for Age Category and Segment: p-value =", p)
```

```
Chi-Square Test for Age Category and Segment: p-value = 0.9435195542868468
```

```python
#18 The current year has shown a significant rise in claim amounts as
#compared to 2016-17 fiscal average which was $10,000.

# a18: Claim amounts in current year vs historical average
historical_avg = 10000
current_avg = df[df['claim_date'] >= '2018-01-01']['claim_amount'].mean()
t_stat, p_value = ttest_ind(df[df['claim_date'] >= '2018-01-01']['claim_amount'],
[historical_avg]*len(df))
print("\nT-Test for Rise in Claim Amounts: p-value =", p_value)
```

```
T-Test for Rise in Claim Amounts: p-value = 1.046411231276106e-10
```

```python
#Q19 Is there any difference between age groups and insurance claims?

# A19: Difference in insurance claims by age group
age_group_claims = df.groupby('age_category')['claim_amount'].mean()
print("\nAverage Claim Amount by Age Group:")
print(age_group_claims)
```

```
Average Claim Amount by Age Group:
age_category
Children             NaN
Youth       12642.977649
Adult       12632.656681
Senior      11087.681934
Name: claim_amount, dtype: float64
```

```python
#Q20 Is there any relationship between total number of policy claims and the claimed amount?

# A20: Correlation between policy claims and claim amount
correlation = df[['total_policy_claims', 'claim_amount']].corr()
print("\nCorrelation Between Total Policy Claims and Claimed Amount:")
print(correlation)
```

```
Correlation Between Total Policy Claims and Claimed Amount:
                     total_policy_claims  claim_amount
total_policy_claims             1.000000     -0.018001
claim_amount                   -0.018001      1.000000
```