Auto Insurance Claims Analysis

Exploratory Data Analysis (EDA) & Insights

---

1 Introduction

This project explores **Auto Insurance Claims Data** using **Python** to gain insights into claim patt customer behavior, and potential fraud cases. We will analyze trends, detect anomalies, and seg customers based on claim amounts and lifetime value.

Key Objectives:

✔ Understand customer **Lifetime Value (CLV)**
✔ Analyze claim trends **geographically and over time**
✔ Detect **fraudulent claims** using statistical methods
✔ Perform **customer segmentation** for better risk assessment

---

2 Data Loading & Preprocessing

We start by loading the dataset and checking for missing values or inconsistencies.

3 Descriptive Analysis

Exploring basic statistics, distributions, and key metrics like Claim Amounts and CLV.

4 Customer Insights

Understanding customer behavior by analyzing CLV distribution and claim patterns.

5 Claims Analysis

Identifying trends in claim amounts, coverage types, and vehicle categories.

6 Time Series Analysis

Analyzing claim trends over time to detect seasonal patterns.

7 Fraud Detection & Outlier Analysis

Detecting unusual claims and potential fraudulent cases using boxplots and statistical methods.

8 Geographical Analysis

Visualizing claims distribution across different states.

9 Customer Segmentation

Using clustering techniques to group customers based on CLV and claim amounts.

Conclusion

✔ Customers with **higher CLV tend to have higher claim amounts**.
✔ **Certain states** report significantly higher claims than others.
✔ **Outlier detection** helps identify potential **fraudulent claims**.
✔ **Segmentation helps insurers** tailor policies for different risk groups.

Next Steps:

Implement **predictive modeling** for fraud detection and risk assessment.

Auto Insurance Claims - Analysis

This notebook analyzes auto insurance claims using structured data and visualizations.

In [2]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
file_path = r"Corrected_AutoInsuranceClaims2024.csv"
df = pd.read_csv(file_path)

# Display first few rows
df.head()
```

Out[2]:

	Customer	State	Customer Lifetime Value	Response	Coverage	Coverage Index	Education	Educatio Ind
0	AA10041	California	14827.62	No	Basic	0	High School or Below	
1	AA11235	Nevada	4820.44	No	Basic	0	Bachelor	
2	AA16582	Washington	45275.26	Yes	Basic	0	Bachelor	
3	AA30683	California	12375.71	No	Premium	2	Bachelor	
4	AA34092	California	54043.12	No	Extended	1	College	

5 rows × 34 columns

In [3]:
```python
# Descriptive Statistics
print("Basic Statistics:")
print(df.describe())

# Unique customers
print(f"Number of unique customers: {df['Customer'].nunique()}")

# Distribution of Total Claim Amount
plt.figure(figsize=(8, 5))
sns.histplot(df["Total Claim Amount"], bins=30, kde=True)
plt.title("Distribution of Total Claim Amount")
plt.xlabel("Total Claim Amount")
plt.ylabel("Frequency")
plt.show()
```

Basic Statistics:

	Customer Lifetime Value	Coverage Index	Education Index	\
count	9134.000000	9134.000000	9134.000000	
mean	15021.270761	0.480622	1.288373	
std	12893.370722	0.655817	1.079984	
min	3561.610000	0.000000	0.000000	
25%	7495.210000	0.000000	0.000000	
50%	10846.520000	0.000000	1.000000	
75%	16817.502500	1.000000	2.000000	
max	156360.070000	2.000000	4.000000	

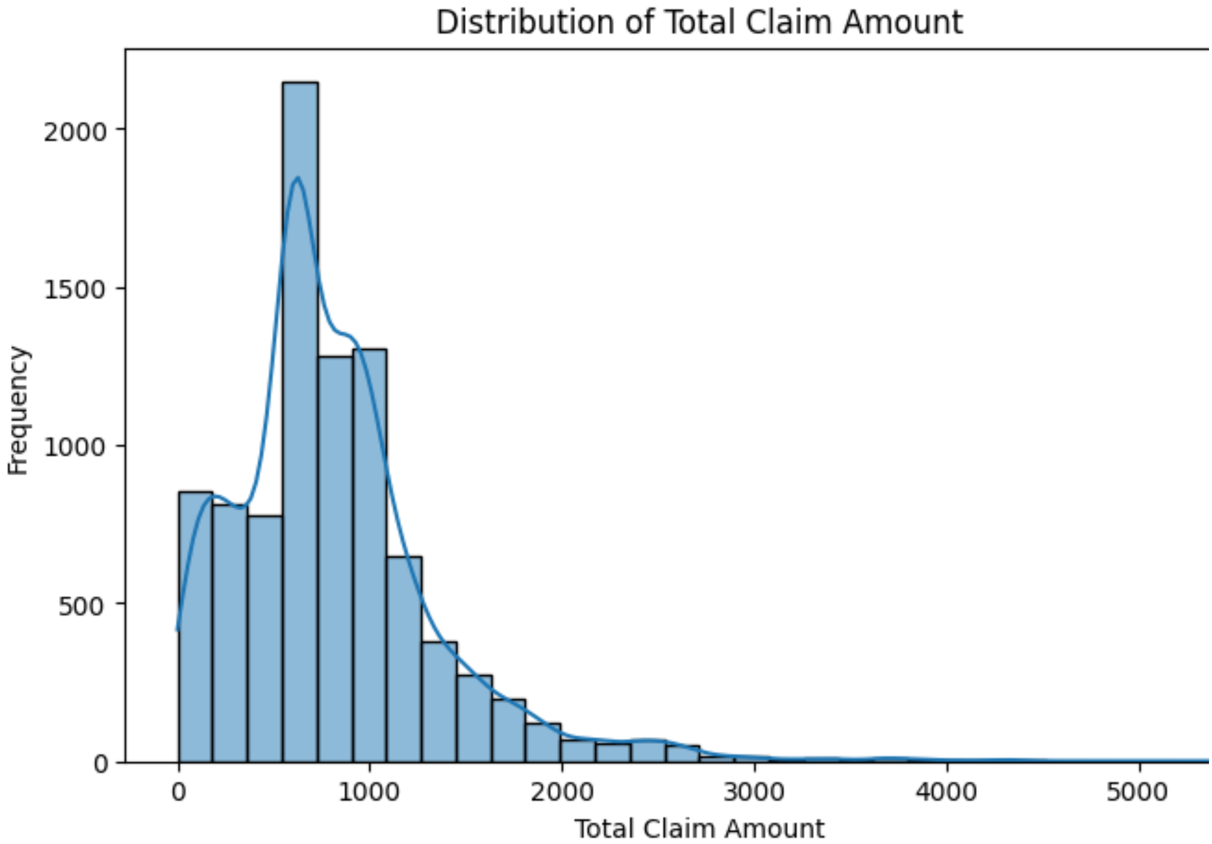
	Employment Status Index	Income	Location Index	\
count	9134.000000	9134.000000	9134.000000	
mean	0.988395	70664.095021	0.979089	
std	0.907454	57007.897853	0.605732	
min	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	1.000000	
50%	1.000000	63593.890000	1.000000	
75%	1.000000	116943.480000	1.000000	
max	4.000000	187613.860000	2.000000	

	Marital Status Index	Monthly Premium Auto	Months Since Last Claim	\
count	9134.000000	9134.000000	9134.000000	
mean	0.879790	174.968190	20.394022	
std	0.636838	64.591363	13.633576	
min	0.000000	113.980000	0.000000	
25%	0.000000	127.880000	8.000000	
50%	1.000000	155.680000	19.000000	
75%	1.000000	204.330000	31.000000	
max	2.000000	558.780000	47.000000	

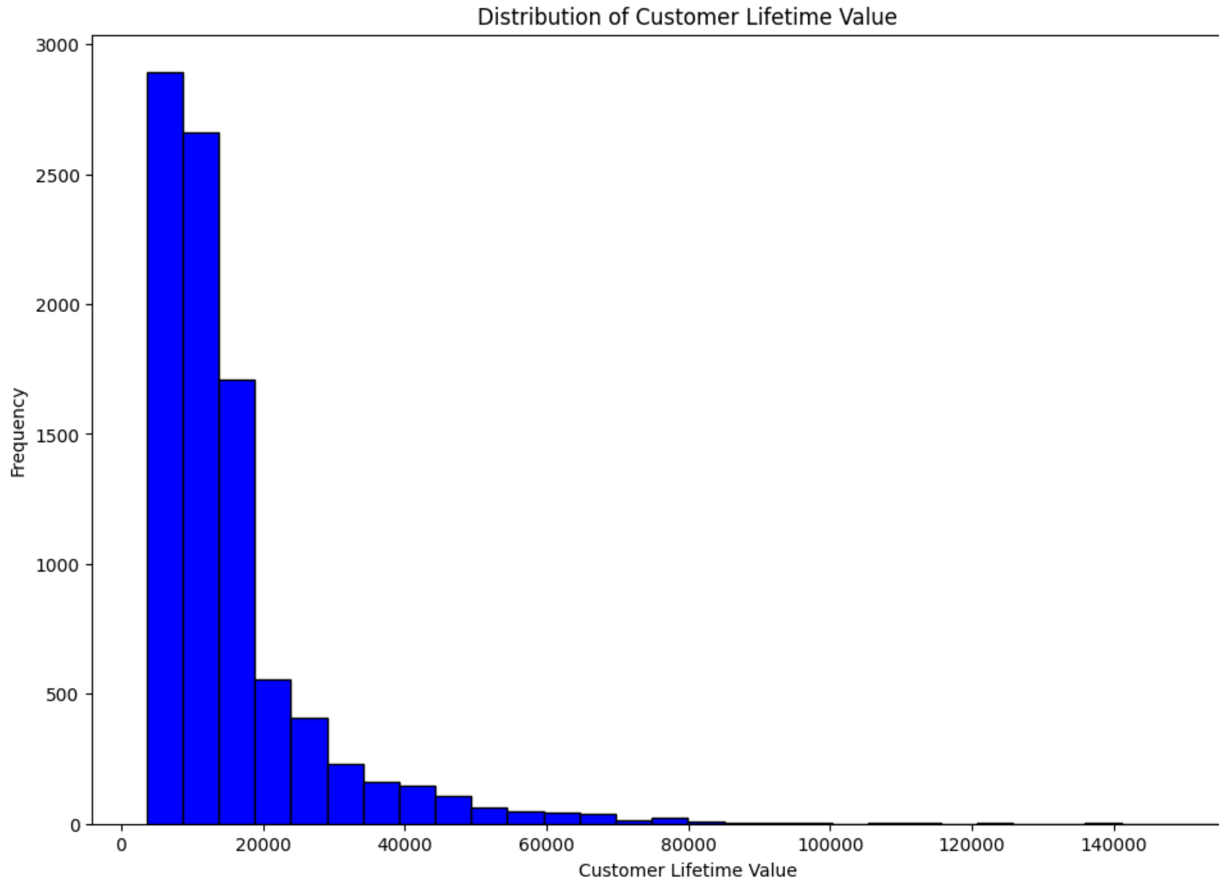
	Months Since Policy Inception	Number of Open Complaints	\
count	9134.000000	9134.000000	
mean	64.912853	0.384388	
std	37.688818	0.910384	
min	0.000000	0.000000	
25%	32.000000	0.000000	
50%	65.000000	0.000000	
75%	96.000000	0.000000	
max	134.000000	5.000000	

	Number of Policies	Policy Type Index	Policy Index	Renew Offer Type
count	9134.000000	9134.000000	9134.000000	9134.000000
mean	2.966170	0.298226	2.214692	1.970221
std	2.390182	0.540451	1.782244	1.007576
min	1.000000	0.000000	0.000000	1.000000
25%	1.000000	0.000000	1.000000	1.000000
50%	2.000000	0.000000	2.000000	2.000000
75%	4.000000	1.000000	3.000000	3.000000
max	9.000000	2.000000	8.000000	4.000000

	Sales Channel Index	Total Claim Amount	Vehicle Class Index	\
count	9134.000000	9134.000000	9134.000000	
mean	1.622071	814.567755	1.552660	
std	0.954878	545.123436	1.455202	
min	0.000000	0.180000	0.000000	
25%	1.000000	510.887500	1.000000	
50%	2.000000	720.475000	1.000000	
75%	2.000000	1027.412500	2.000000	
max	3.000000	5429.160000	5.000000	



In [4]:
```
plt.figure(figsize=(12, 8))
plt.hist(df['Customer Lifetime Value'].dropna(), bins=30, color='blue', ed

# Customize the plot
plt.title('Distribution of Customer Lifetime Value')
plt.xlabel('Customer Lifetime Value')
plt.ylabel('Frequency')
plt.show()
```



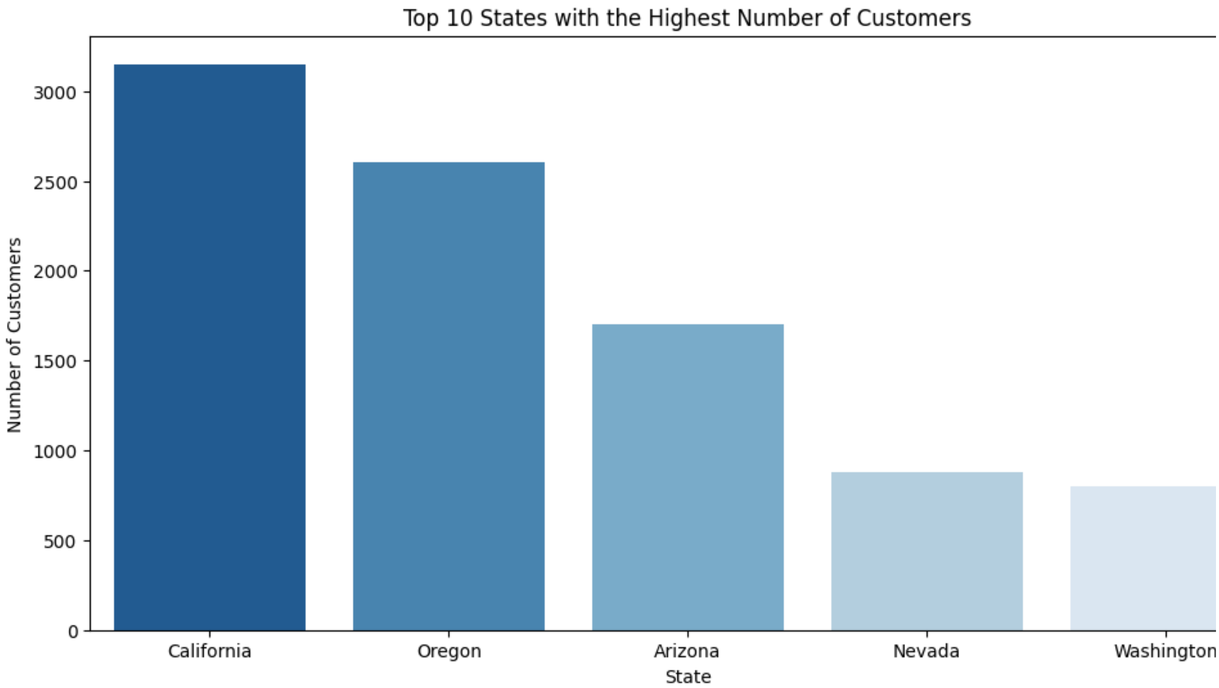
In [5]:
```
import pandas as pd

# Load the dataset
file_path = "Corrected_AutoInsuranceClaims2024.csv"  # Update with your co
df = pd.read_csv(file_path)

# Calculate total claim amounts by state and get the top 10
state_claims = df.groupby("State")["Total Claim Amount"].sum().sort_values

# Display the results
print(state_claims)
```

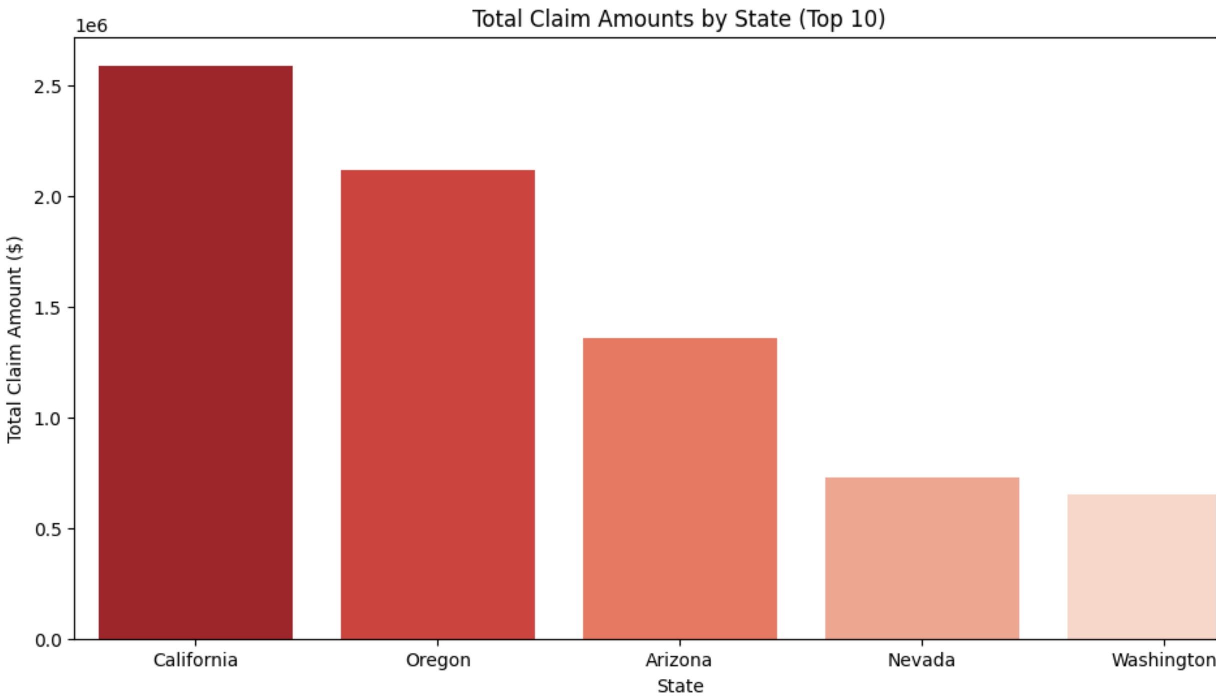
```
State
California    2587939.11
Oregon        2113437.91
Arizona       1359319.18
Nevada         726164.24
Washington     653401.43
Name: Total Claim Amount, dtype: float64
```

Q1: Which state has the highest number of customers a how does that impact total claim amounts?

In [ ]:
```
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

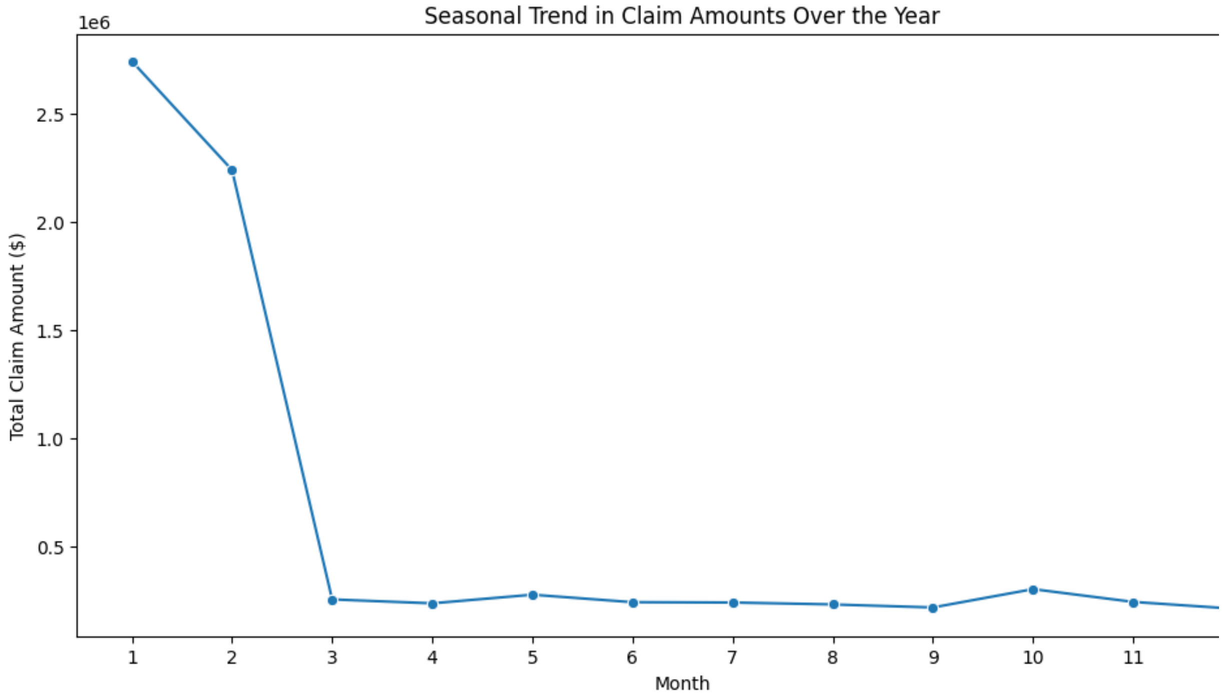
In [7]:
```
plt.figure(figsize=(12, 6))
state_counts = df['State'].value_counts().head(10)  # Top 10 states
sns.barplot(x=state_counts.index, y=state_counts.values, palette='Blues_r
plt.title("Top 10 States with the Highest Number of Customers")
plt.xlabel("State")
plt.ylabel("Number of Customers")
plt.show()

plt.figure(figsize=(12, 6))
state_claims = df.groupby('State')['Total Claim Amount'].sum().sort_value:
sns.barplot(x=state_claims.index, y=state_claims.values, palette='Reds_r'
plt.title("Total Claim Amounts by State (Top 10)")
plt.xlabel("State")
plt.ylabel("Total Claim Amount ($)")
plt.show()
```





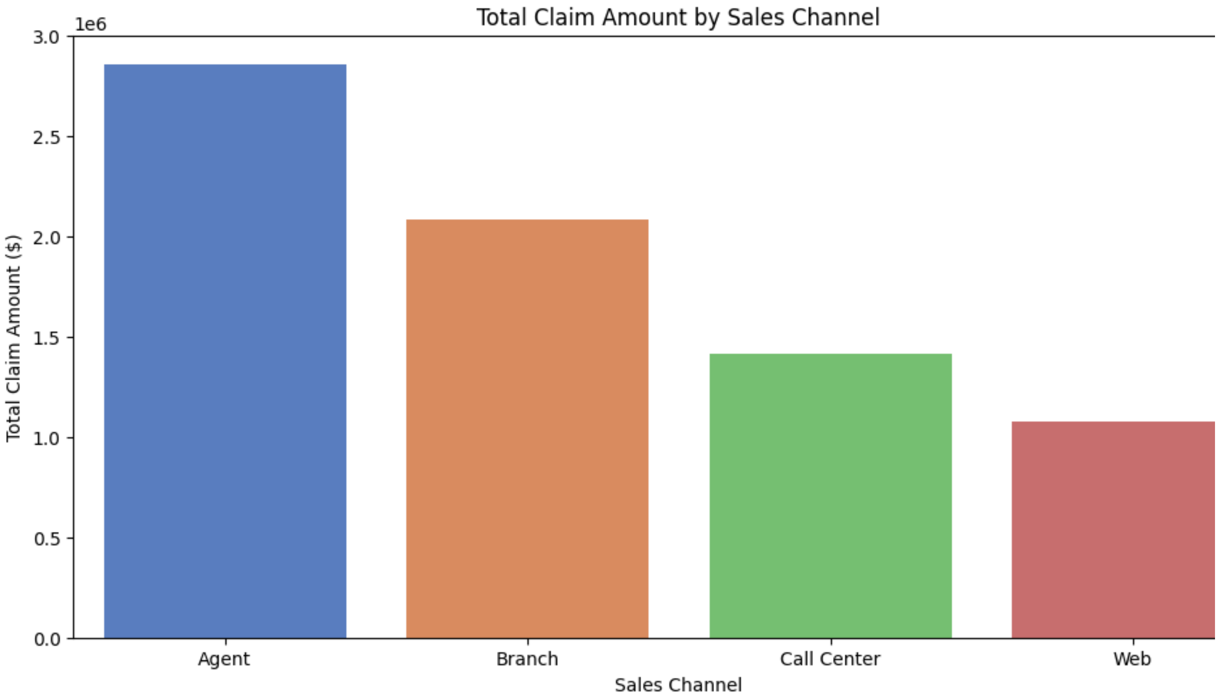
Q2: Is there a seasonal trend in claim amounts over the year?

In [8]:
```
df['Effective To Date'] = pd.to_datetime(df['Effective To Date'], dayfirs
df['Month'] = df['Effective To Date'].dt.month

plt.figure(figsize=(12, 6))
sns.lineplot(x='Month', y='Total Claim Amount', data=df, estimator='sum',
plt.xticks(np.arange(1, 13, 1))
plt.title("Seasonal Trend in Claim Amounts Over the Year")
plt.xlabel("Month")
plt.ylabel("Total Claim Amount ($)")
plt.show()
```



Q3: Which sales channel (e.g., web, search, agent) generates the highest total claim amount?

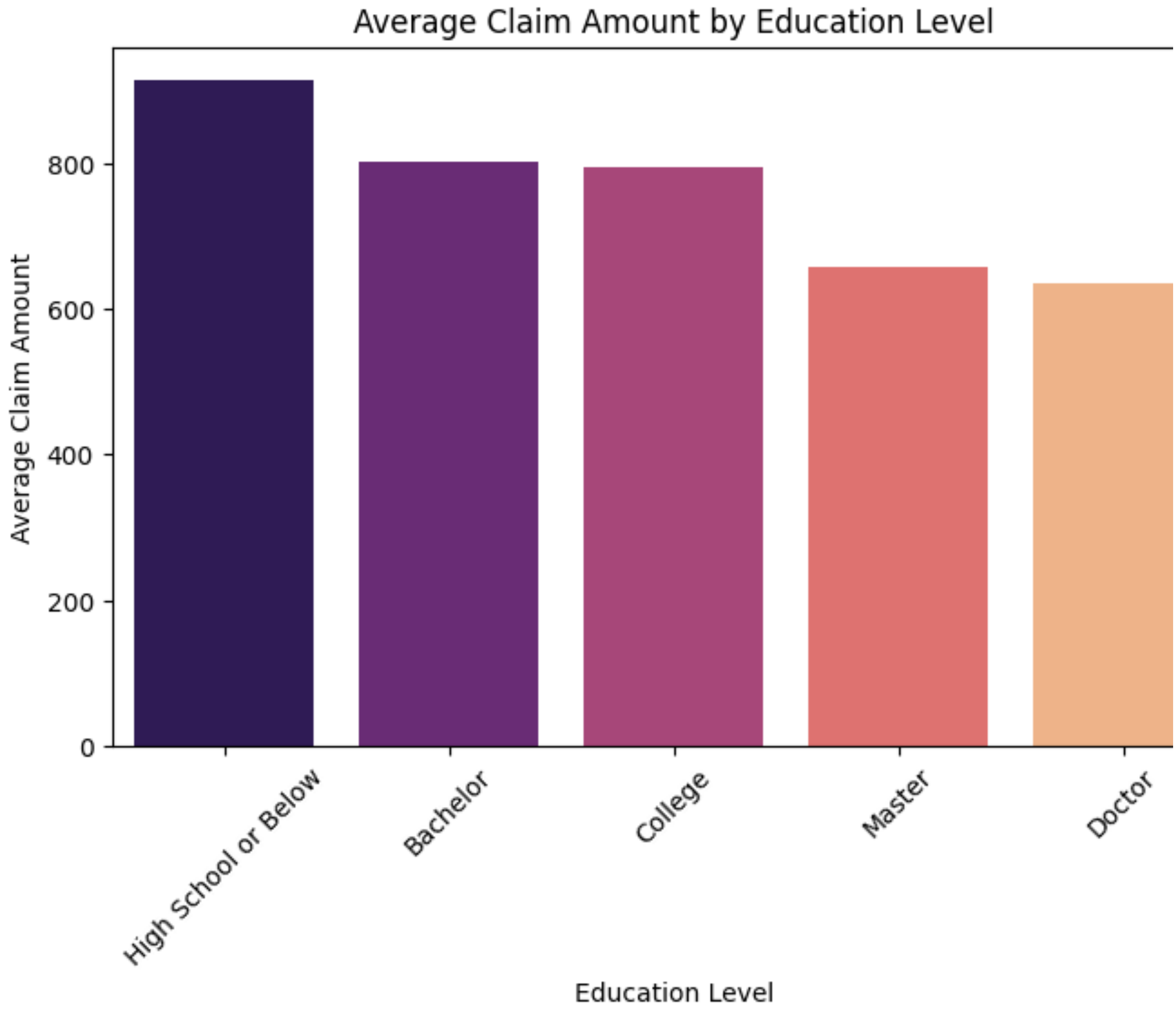
In [9]:
```
plt.figure(figsize=(12, 6))
sales_channel_claims = df.groupby('Sales Channel')['Total Claim Amount'].s
sns.barplot(x=sales_channel_claims.index, y=sales_channel_claims.values, p
plt.title("Total Claim Amount by Sales Channel")
plt.xlabel("Sales Channel")
plt.ylabel("Total Claim Amount ($)")
plt.show()
```



Q4: Education level impact on claim amount

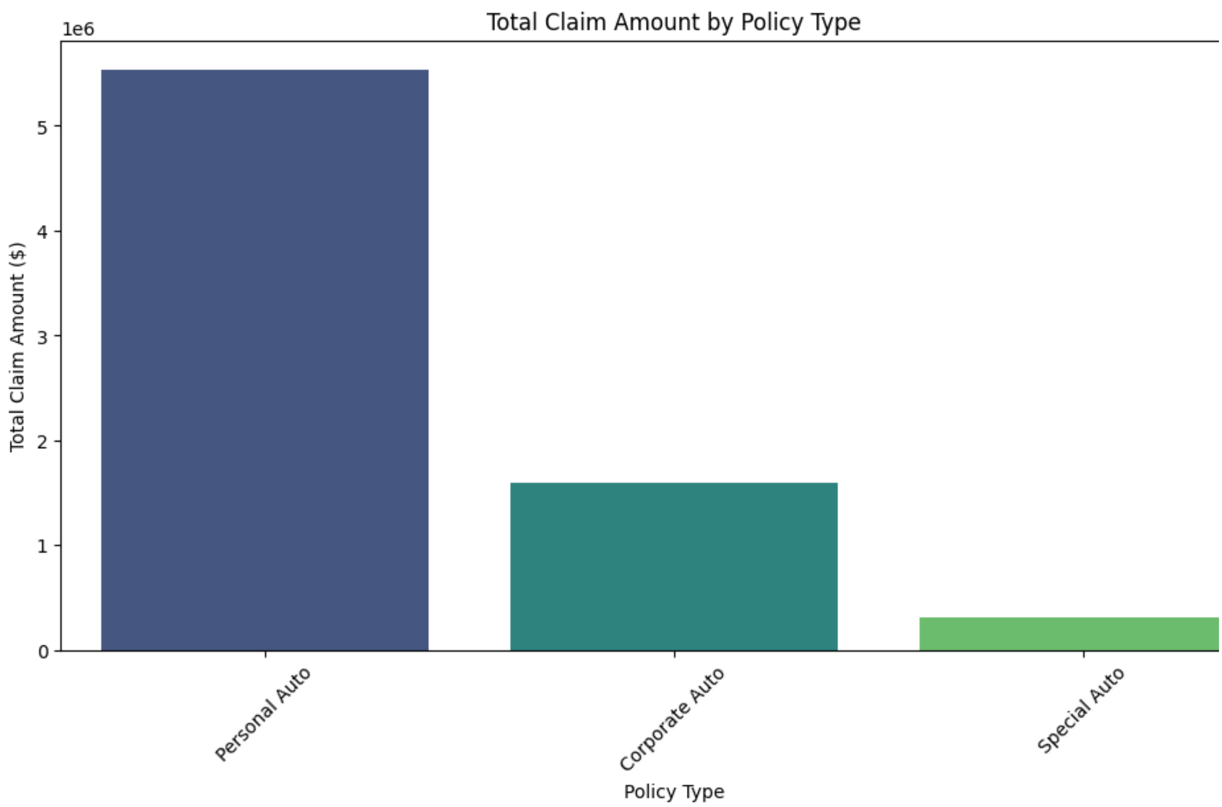
In [10]:
```
# Q4: Education level impact on claim amount
education_claims = df.groupby("Education")["Total Claim Amount"].mean().s
print(education_claims)

# Bar chart
plt.figure(figsize=(8,5))
sns.barplot(x=education_claims.index, y=education_claims.values, palette=
plt.xlabel("Education Level")
plt.ylabel("Average Claim Amount")
plt.title("Average Claim Amount by Education Level")
plt.xticks(rotation=45)
plt.show()
```

```
Education
High School or Below    914.211629
Bachelor                803.130899
College                 795.285162
Master                  657.220040
Doctor                  634.607310
Name: Total Claim Amount, dtype: float64
```



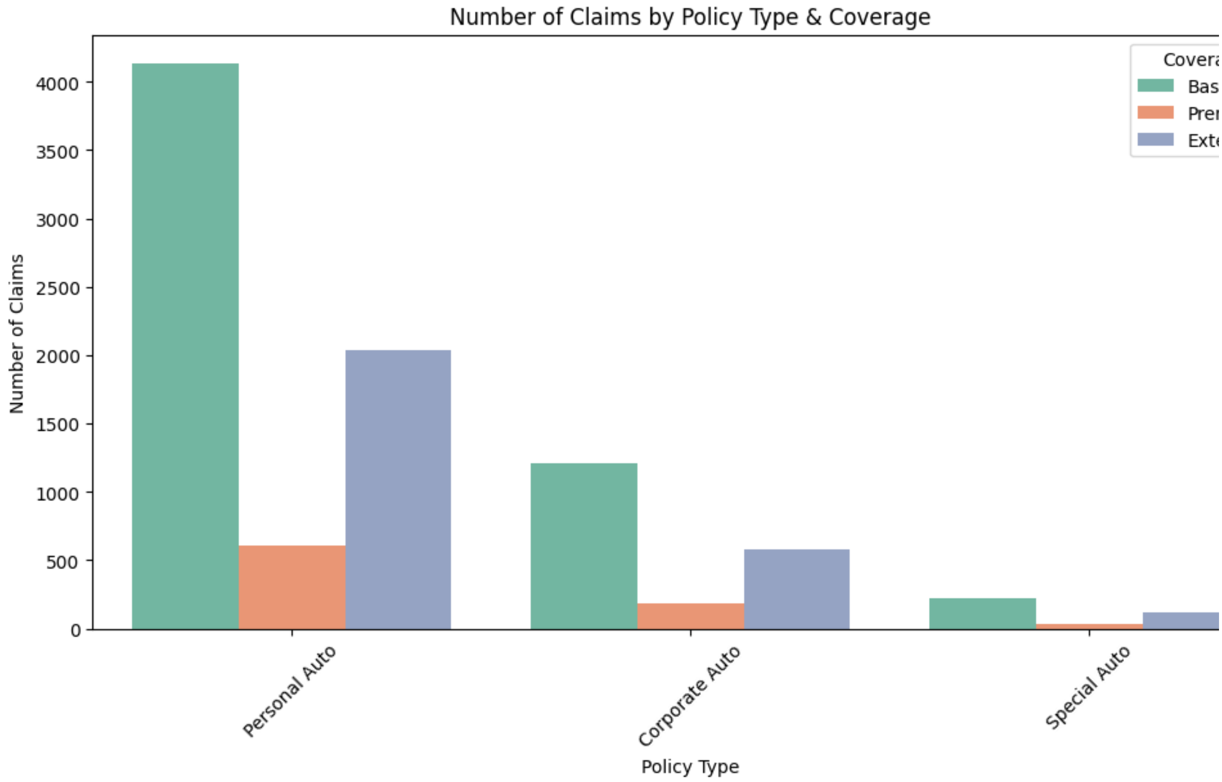
Q5: Do customers with multiple policies file more claims compared to those with a single policy?

```
In [11]: plt.figure(figsize=(12, 6))
policy_claims = df.groupby('Policy Type')['Total Claim Amount'].sum().sor
sns.barplot(x=policy_claims.index, y=policy_claims.values, palette='virid
plt.title("Total Claim Amount by Policy Type")
plt.xlabel("Policy Type")
plt.ylabel("Total Claim Amount ($)")
plt.xticks(rotation=45)
plt.show()
```



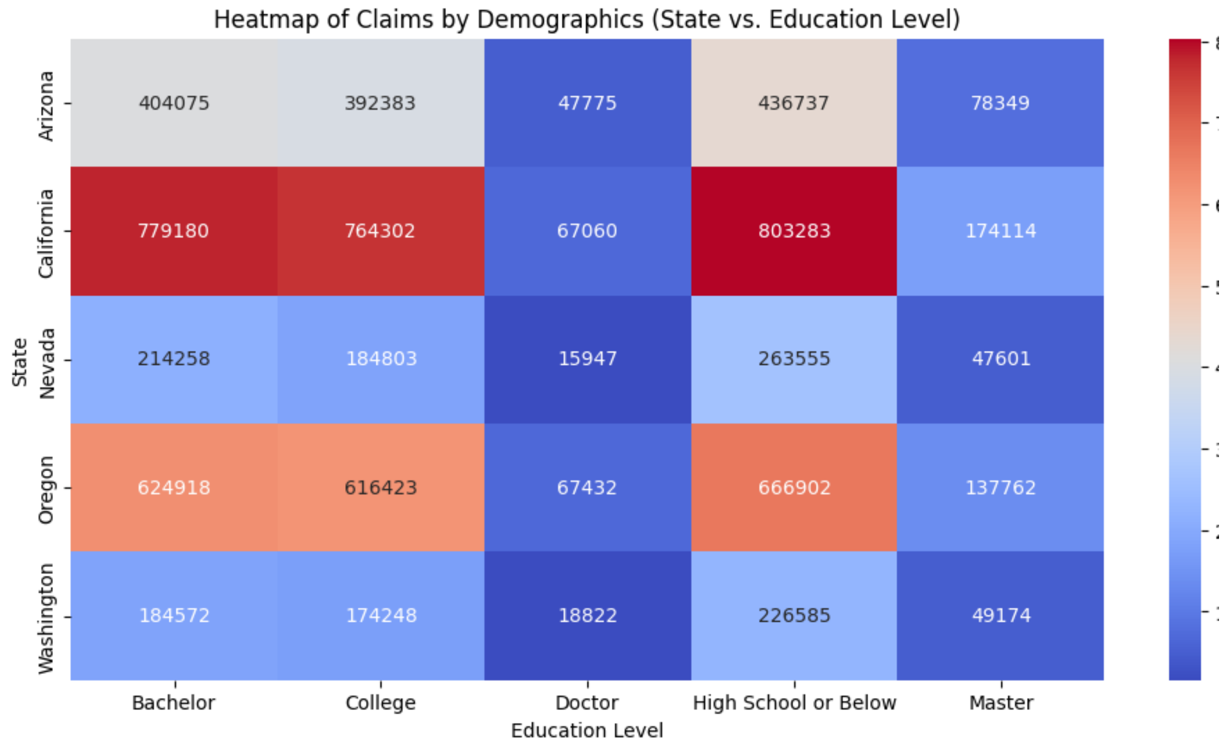
Q6: Claims by Policy Type & Coverage:

In [ ]:
```
plt.figure(figsize=(12, 6))
sns.countplot(x='Policy Type', hue='Coverage', data=df, palette='Set2')
plt.title("Number of Claims by Policy Type & Coverage")
plt.xlabel("Policy Type")
plt.ylabel("Number of Claims")
plt.xticks(rotation=45)
plt.show()
```



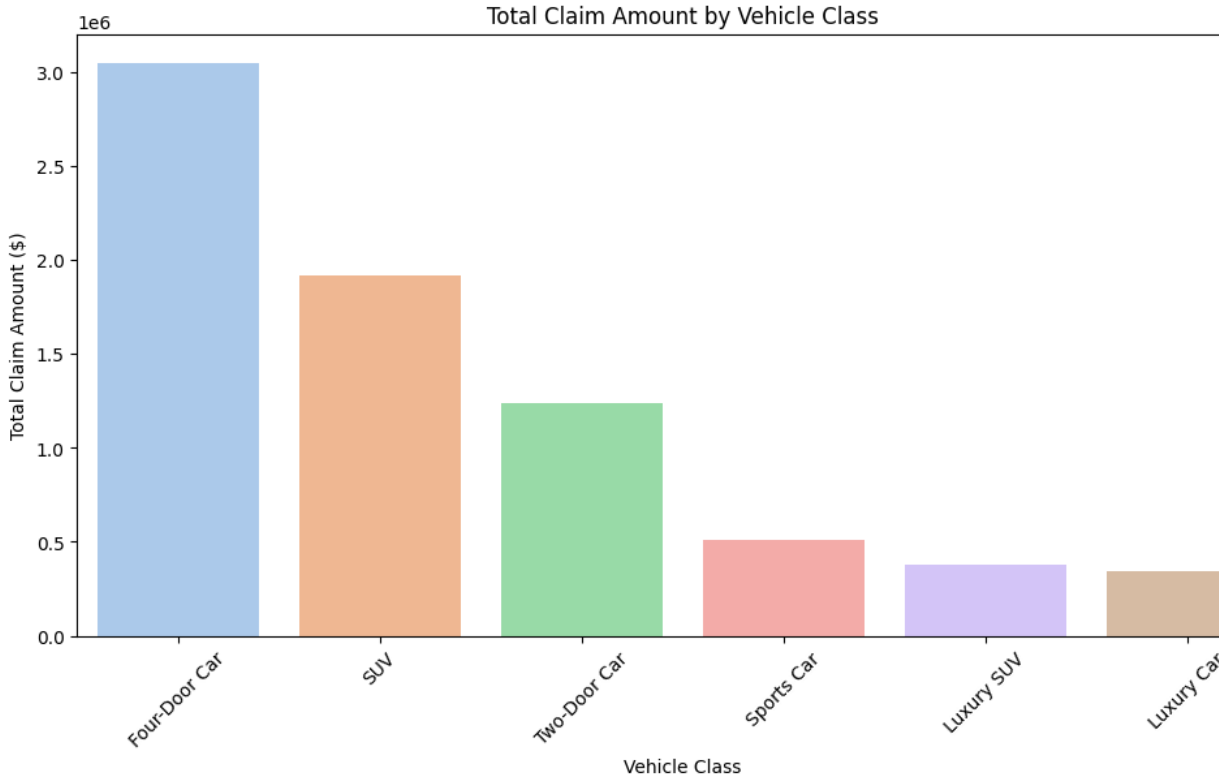
Q7: Claims by Demographics:

In [13]:
```
plt.figure(figsize=(12, 6))
heatmap_data = df.pivot_table(index='State', columns='Education', values=
sns.heatmap(heatmap_data, cmap='coolwarm', annot=True, fmt=".0f")
plt.title("Heatmap of Claims by Demographics (State vs. Education Level)"
plt.xlabel("Education Level")
plt.ylabel("State")
plt.show()
```



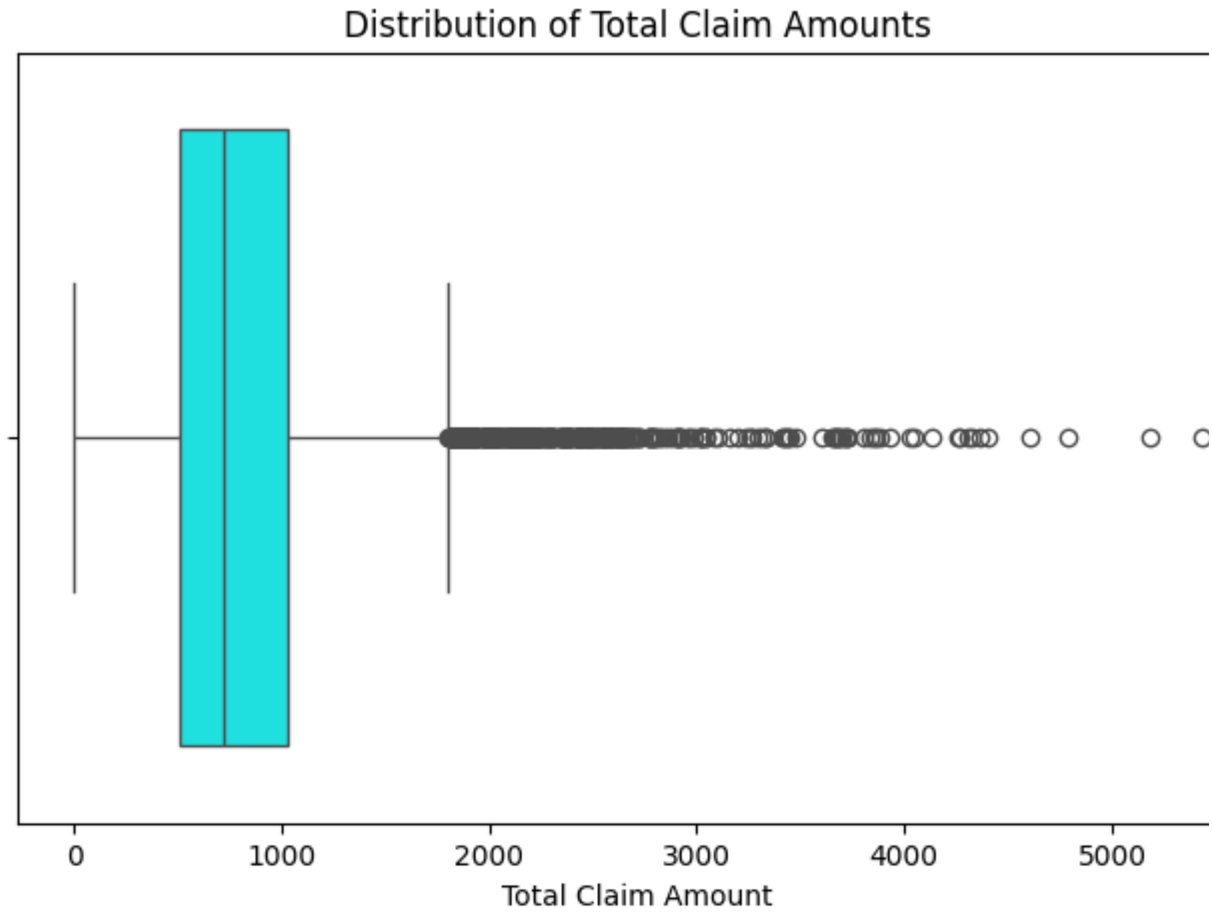
Q8: Vehicle Class vs. Claim Amount:

In [14]:
```
plt.figure(figsize=(12, 6))
vehicle_claims = df.groupby('Vehicle Class')['Total Claim Amount'].sum().s
sns.barplot(x=vehicle_claims.index, y=vehicle_claims.values, palette='pas
plt.title("Total Claim Amount by Vehicle Class")
plt.xlabel("Vehicle Class")
plt.ylabel("Total Claim Amount ($)")
plt.xticks(rotation=45)
plt.show()
```



Q9: Distribution of claim amounts and outliers

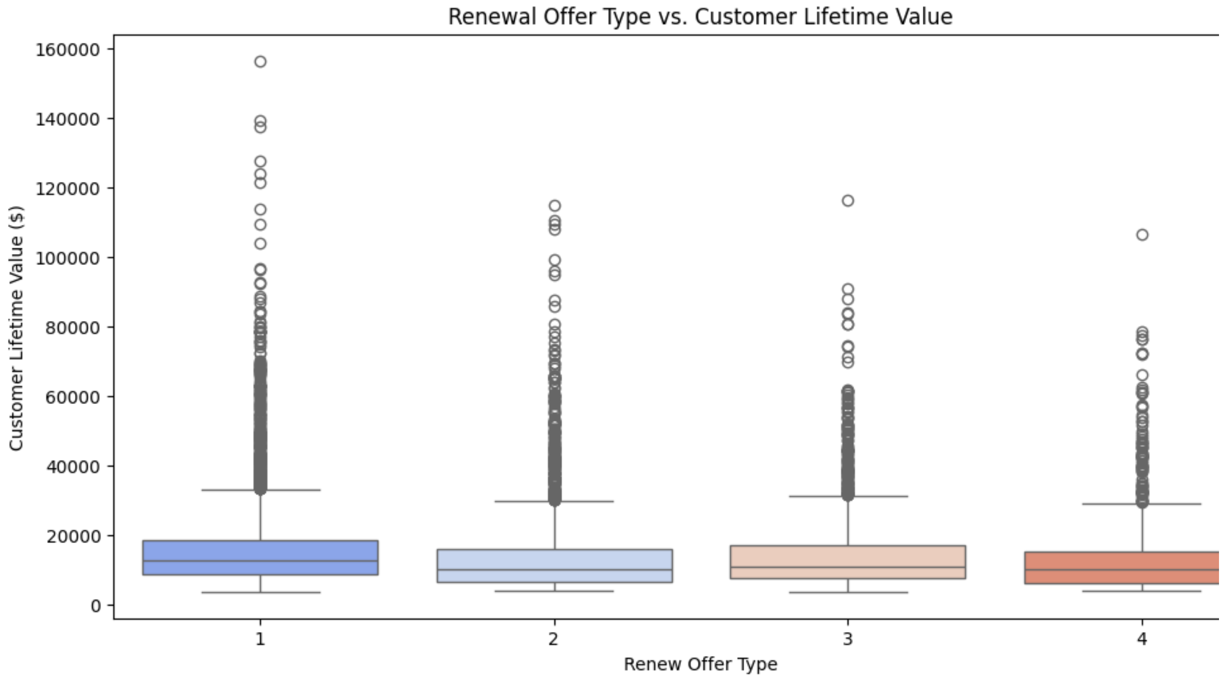
In [15]:
```
# Q9: Distribution of claim amounts and outliers
plt.figure(figsize=(8,5))
sns.boxplot(x=df["Total Claim Amount"], color="cyan")
plt.xlabel("Total Claim Amount")
plt.title("Distribution of Total Claim Amounts")
plt.show()

# Identifying outliers
q1 = df["Total Claim Amount"].quantile(0.25)
q3 = df["Total Claim Amount"].quantile(0.75)
iqr = q3 - q1
lower_bound = q1 - 1.5 * iqr
upper_bound = q3 + 1.5 * iqr

outliers = df[(df["Total Claim Amount"] < lower_bound) | (df["Total Claim
print(f"Number of outliers in Total Claim Amount: {len(outliers)}")
```




```
Number of outliers in Total Claim Amount: 453
```

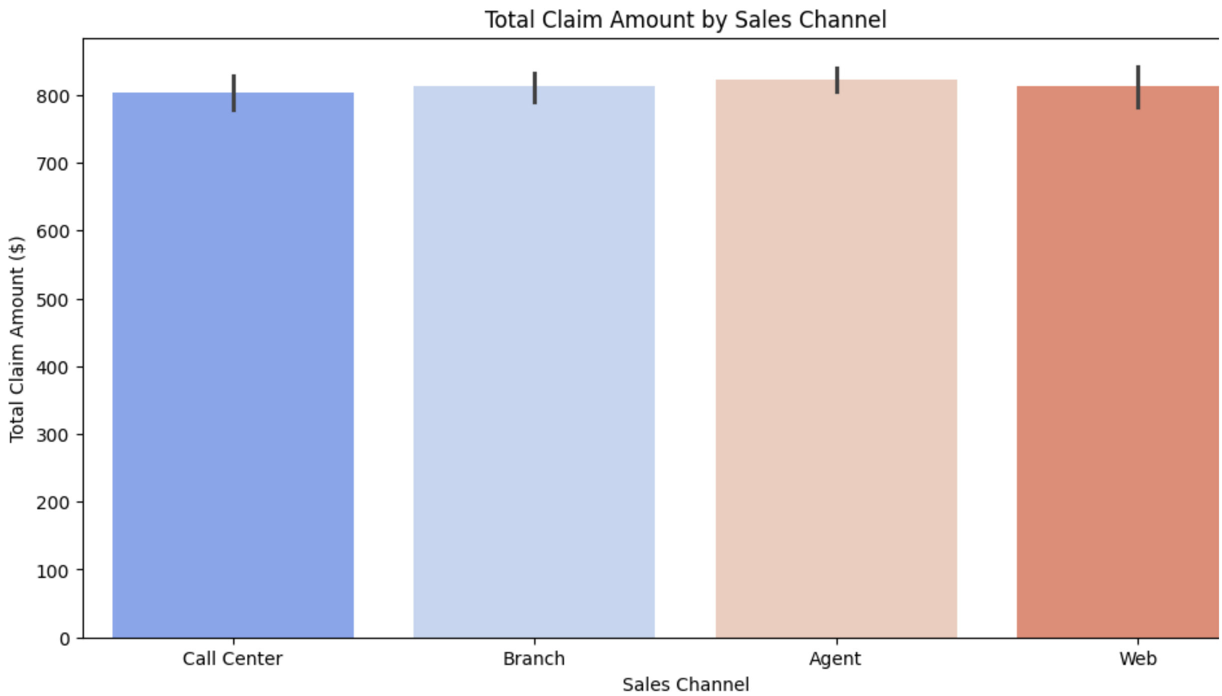
Q10: Renewal Offer Type vs. Customer Lifetime Value:

In [16]:
```
plt.figure(figsize=(12, 6))
sns.boxplot(x='Renew Offer Type', y='Customer Lifetime Value', data=df, pa
plt.title("Renewal Offer Type vs. Customer Lifetime Value")
plt.xlabel("Renew Offer Type")
plt.ylabel("Customer Lifetime Value ($)")
plt.show()
```



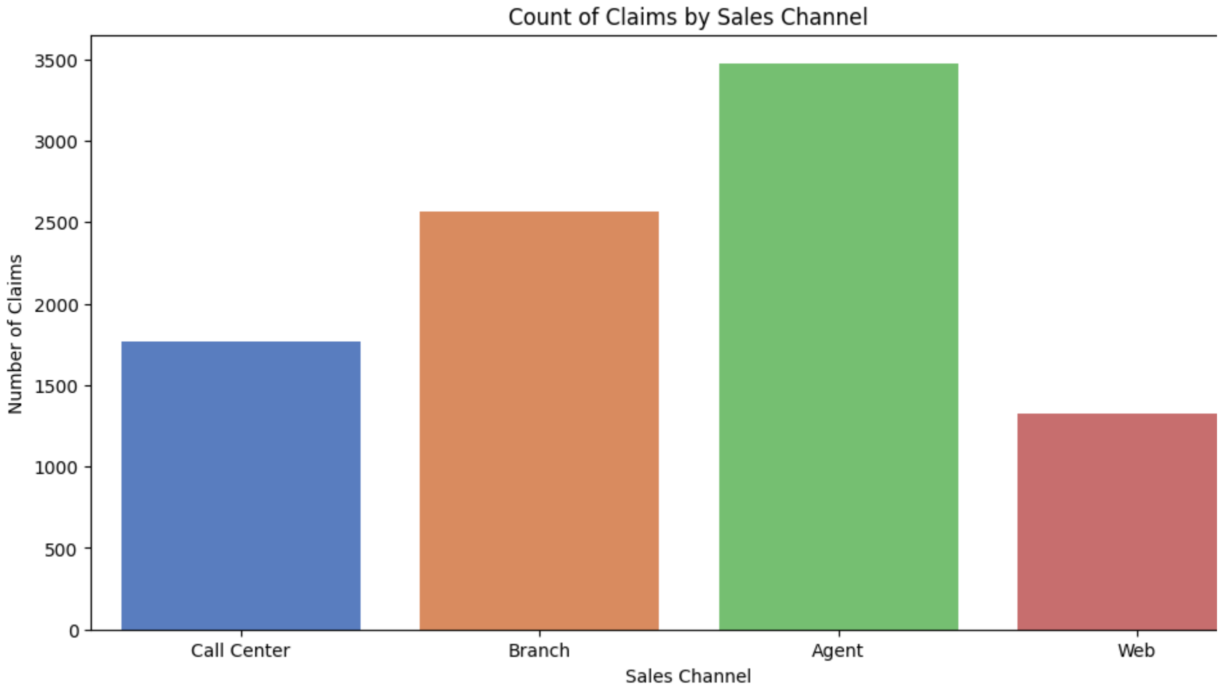
Q11: Sales & Marketing Insights:

In [17]:
```
plt.figure(figsize=(12, 6))
sns.barplot(x='Sales Channel', y='Total Claim Amount', data=df, palette='
plt.title("Total Claim Amount by Sales Channel")
plt.xlabel("Sales Channel")
plt.ylabel("Total Claim Amount ($)")
plt.show()
```



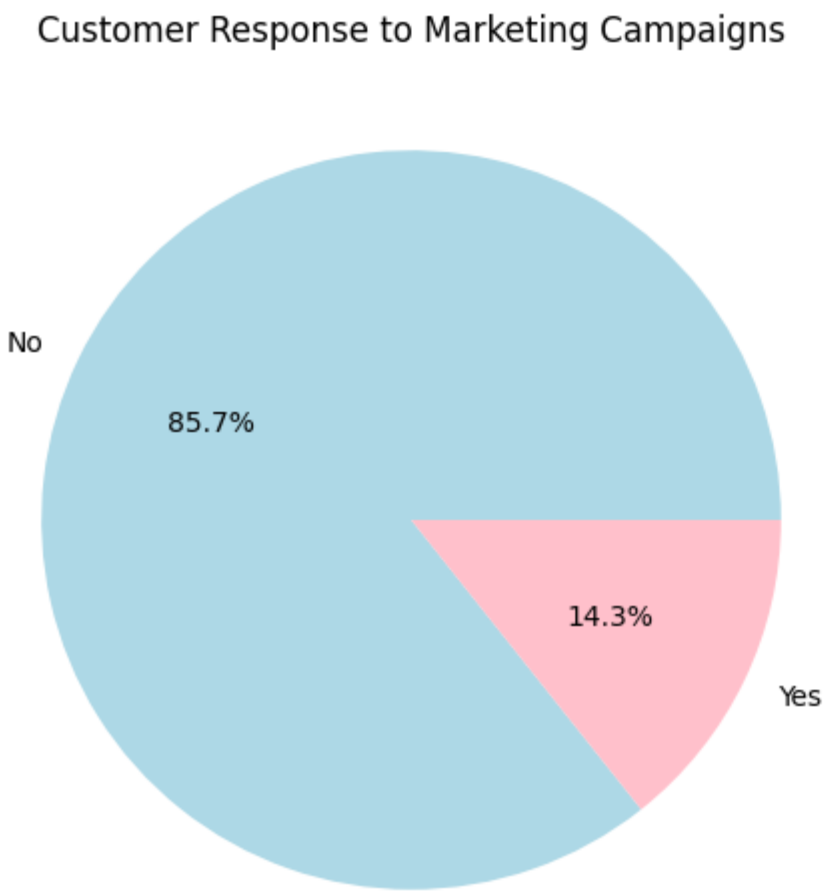
Q12: Sales Channel Effectiveness:

```
In [18]: plt.figure(figsize=(12, 6))
         sns.countplot(x='Sales Channel', data=df, palette='muted')
         plt.title("Count of Claims by Sales Channel")
         plt.xlabel("Sales Channel")
         plt.ylabel("Number of Claims")
         plt.show()
```

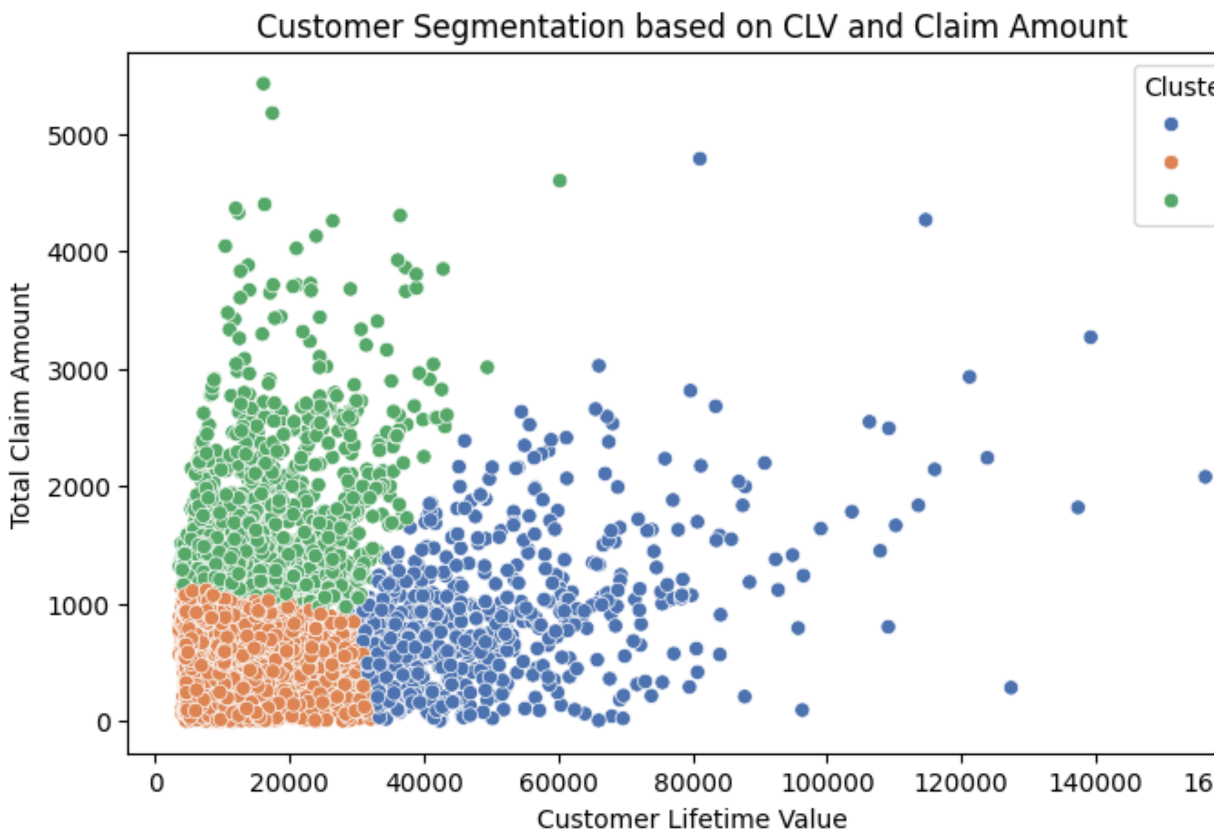


Q13: Response to Marketing Campaigns:

```
In [19]: plt.figure(figsize=(6, 6))
         df['Response'].value_counts().plot.pie(autopct='%1.1f%%', colors=['lightb
         plt.title("Customer Response to Marketing Campaigns")
         plt.ylabel("")
         plt.show()
```



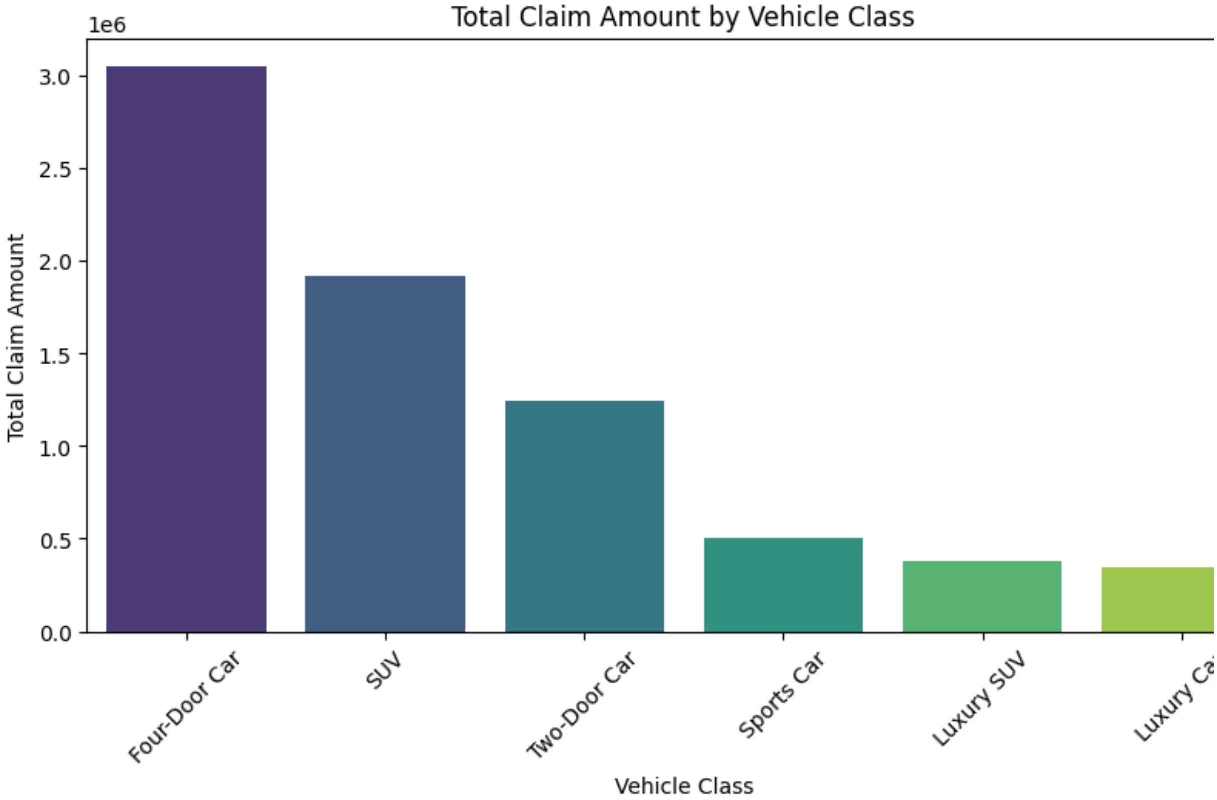

```
In [21]: from sklearn.cluster import KMeans
         from sklearn.preprocessing import StandardScaler

         # Select features for clustering
         features = df[["Customer Lifetime Value", "Total Claim Amount"]].dropna()
         scaler = StandardScaler()
         features_scaled = scaler.fit_transform(features)

         # Apply K-Means clustering
         kmeans = KMeans(n_clusters=3, random_state=42)
         df["Cluster"] = kmeans.fit_predict(features_scaled)

         # Scatter plot of clusters
         plt.figure(figsize=(8, 5))
         sns.scatterplot(x=df["Customer Lifetime Value"], y=df["Total Claim Amount"
         plt.title("Customer Segmentation based on CLV and Claim Amount")
         plt.xlabel("Customer Lifetime Value")
         plt.ylabel("Total Claim Amount")
         plt.legend(title="Cluster")
         plt.show()
```



Additional Business Analysis Questions

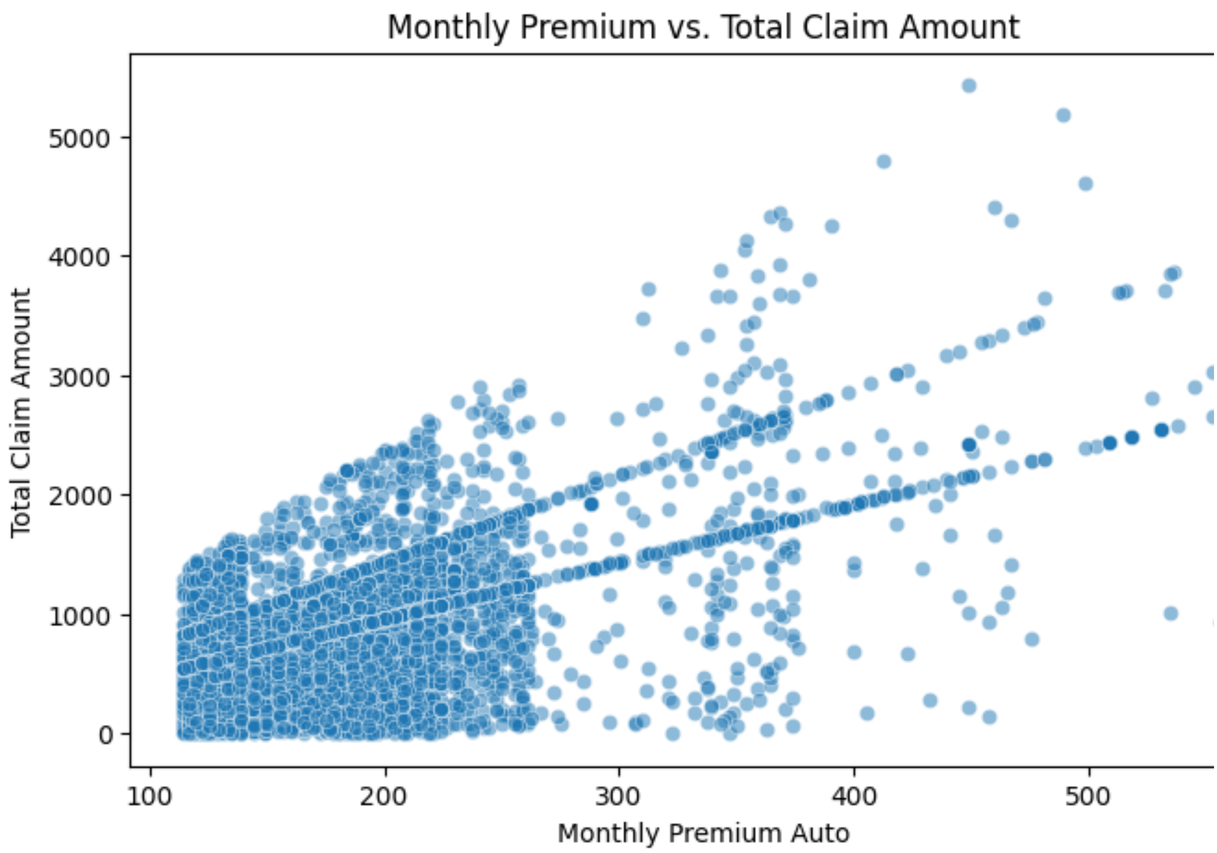
In [22]:
```
# Q1: Which vehicle type has the highest total claim amount?
vehicle_claims = df.groupby("Vehicle Class")["Total Claim Amount"].sum().s
print(vehicle_claims)

# Plot the results
plt.figure(figsize=(10,5))
sns.barplot(x=vehicle_claims.index, y=vehicle_claims.values, palette="vir
plt.xticks(rotation=45)
plt.xlabel("Vehicle Class")
plt.ylabel("Total Claim Amount")
plt.title("Total Claim Amount by Vehicle Class")
plt.show()
```

```
Vehicle Class
Four-Door Car    3050223.08
SUV              1919571.30
Two-Door Car     1240599.38
Sports Car        506924.46
Luxury SUV        377222.70
Luxury Car        345720.95
Name: Total Claim Amount, dtype: float64
```



In [23]:
```
# Q2: Correlation between monthly premium and total claim amount
correlation = df["Monthly Premium Auto"].corr(df["Total Claim Amount"])
print(f"Correlation between Monthly Premium and Total Claim Amount: {corre

# Scatter plot
plt.figure(figsize=(8,5))
sns.scatterplot(x=df["Monthly Premium Auto"], y=df["Total Claim Amount"],
plt.xlabel("Monthly Premium Auto")
plt.ylabel("Total Claim Amount")
plt.title("Monthly Premium vs. Total Claim Amount")
plt.show()
```

```
Correlation between Monthly Premium and Total Claim Amount: 0.63
```

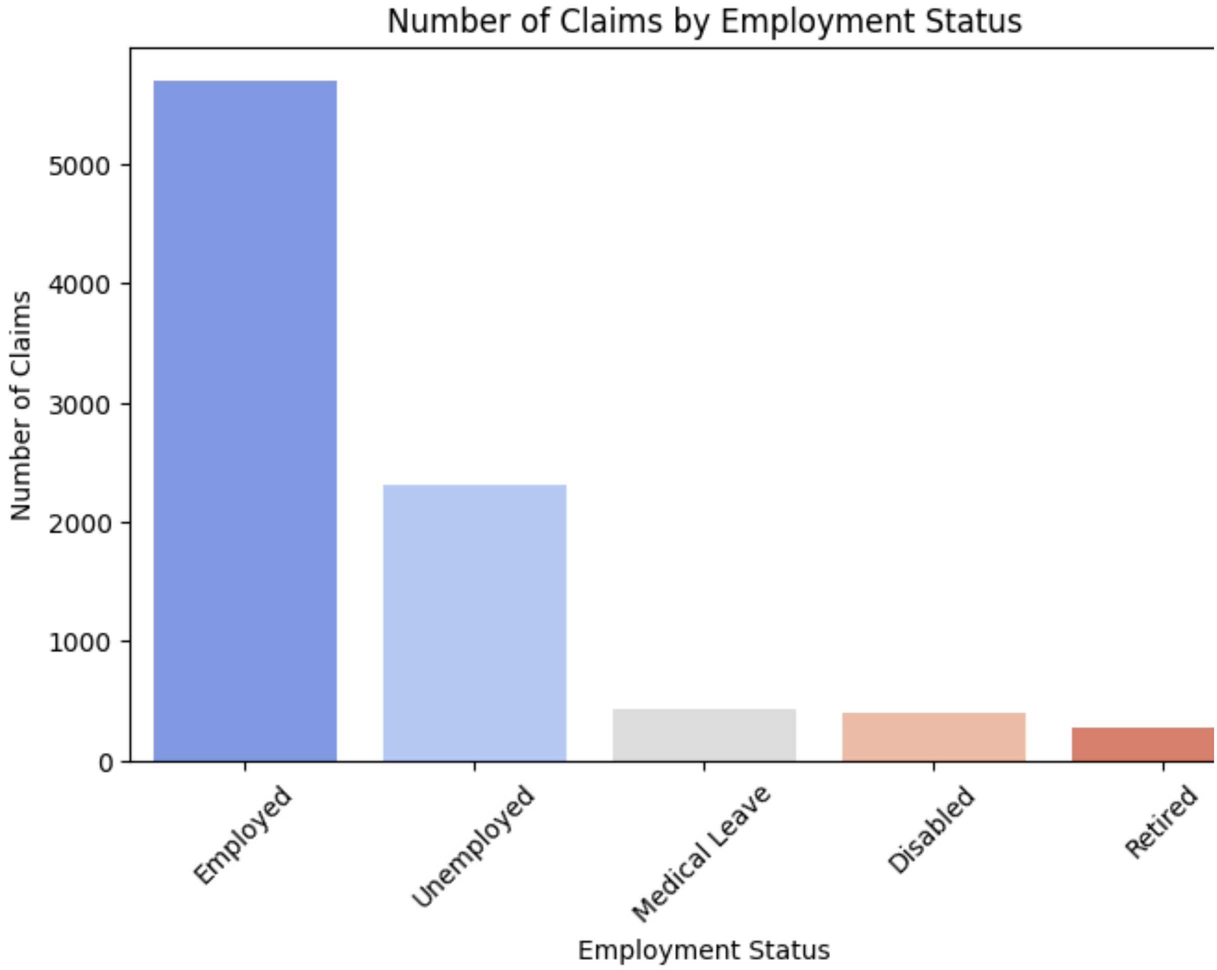



```
In [24]: df.columns
```

```
Out[24]:  Index(['Customer', 'State', 'Customer Lifetime Value', 'Response', 'Coverage
       'Coverage Index', 'Education', 'Education Index', 'Effective To Date'
       'Employment Status', 'Employment Status Index', 'Gender', 'Income',
       'Location', 'Location Index', 'Marital Status', 'Marital Status Index
       'Monthly Premium Auto', 'Months Since Last Claim',
       'Months Since Policy Inception', 'Number of Open Complaints',
       'Number of Policies', 'Policy Type', 'Policy Type Index', 'Policy',
       'Policy Index', 'Renew Offer Type', 'Sales Channel',
       'Sales Channel Index', 'Total Claim Amount', 'Vehicle Class',
       'Vehicle Class Index', 'Vehicle Size', 'Vehicle Size Index', 'Month',
       'Cluster'],
      dtype='object')
```

```
In [25]: # Q3: Employment status with highest claim frequency
employment_claims = df["Employment Status"].value_counts()
print(employment_claims)

# Bar chart
plt.figure(figsize=(8,5))
sns.barplot(x=employment_claims.index, y=employment_claims.values, palett
plt.xlabel("Employment Status")
plt.ylabel("Number of Claims")
plt.title("Number of Claims by Employment Status")
plt.xticks(rotation=45)
plt.show()
```

```
Employment Status
Employed         5698
Unemployed       2317
Medical Leave     432
Disabled          405
Retired           282
Name: count, dtype: int64
```



In [26]:
```
# Q4: Policies per customer for high claim amount customers
high_claim_customers = df[df["Total Claim Amount"] > df["Total Claim Amou
avg_policies = high_claim_customers["Number of Policies"].mean()
print(f"Average number of policies for high claim customers: {avg_policie
```

```
Average number of policies for high claim customers: 2.95
```

In [27]:
```python
# Q5: State-wise claim frequency vs. total claims
state_claim_counts = df["State"].value_counts()
state_total_claims = df.groupby("State")["Total Claim Amount"].sum()

# Combine into one DataFrame
state_analysis = pd.DataFrame({"Claim Frequency": state_claim_counts, "To
state_analysis = state_analysis.sort_values(by="Claim Frequency", ascendi

print(state_analysis.head(10))

# Visualization
fig, ax1 = plt.subplots(figsize=(12,6))

color = 'tab:blue'
ax1.set_xlabel("State")
ax1.set_ylabel("Claim Frequency", color=color)
ax1.bar(state_analysis.index, state_analysis["Claim Frequency"], color=co
ax1.tick_params(axis="y", labelcolor=color)
plt.xticks(rotation=45)

ax2 = ax1.twinx()  # instantiate a second y-axis
color = 'tab:red'
ax2.set_ylabel("Total Claim Amount", color=color)
ax2.plot(state_analysis.index, state_analysis["Total Claim Amount"], colo
ax2.tick_params(axis="y", labelcolor=color)

plt.title("State-wise Claim Frequency vs. Total Claim Amount")
fig.tight_layout()
plt.show()
```

```
            Claim Frequency  Total Claim Amount
State
California             3150          2587939.11
Oregon                 2601          2113437.91
Arizona                1703          1359319.18
Nevada                  882           726164.24
Washington              798           653401.43
```