CHEMICALS IN COSEMETICS

Introduction

The aim of the project is to identify the amount of chemical substances present in a Cosmetic product.The California Safe Cosmetics Program (CSCP), administered by the California Department of Public Health (CDPH), is a vital initiative designed to protect public health by monitoring and disclosing hazardous ingredients in cosmetic products sold in California. The program, established under the California Safe Cosmetics Act, requires cosmetic manufacturers, packers, and distributors to report products containing ingredients that are known or suspected to cause cancer, birth defects, or other developmental or reproductive harm.

Goal of the Project¶

The goal of this project is to analyze and assess the data reported to the California Safe Cosmetics Program (CSCP) to better understand the presence and prevalence of hazardous ingredients in cosmetic products sold in California. By examining this data, the project aims to: 1)Identify trends in the use of chemicals known or suspected to cause cancer, birth defects, or other developmental or reproductive harm within the cosmetic industry. 2)Evaluate the compliance of manufacturers with the reporting requirements set by the California Safe Cosmetics Act, and assess the completeness of the data collected by the CSCP. 3)Increase awareness about the potential health risks associated with certain cosmetic ingredients, and provide insights into how these chemicals are distributed across different product categories and brands. 4)Highlight gaps in the reporting process, including missing data or products that may not be included, and suggest ways to improve transparency and safety within the cosmetics industry. 5)Support consumer education by presenting the data in a user-friendly format, helping consumers make more informed decisions about the personal care products they use.

In [24]:

```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

In [25]:
```
df=pd.read_csv("C:/Users/user/Downloads/cscpopendata.csv")
df
```

Out[25]:

	CDPHId	ProductName	CSFId	CSF	CompanyId	CompanyName	BrandName	Prir
0	2	ULTRA COLOR RICH EXTRA PLUMP LIPSTICK-ALL SHADES	NaN	NaN	4	New Avon LLC	AVON	
1	3	Glover's Medicated Shampoo	NaN	NaN	338	J. Strickland & Co.	Glover's	
2	3	Glover's Medicated Shampoo	NaN	NaN	338	J. Strickland & Co.	Glover's	
3	4	PRECISION GLIMMER EYE LINER-ALL SHADES �	NaN	NaN	4	New Avon LLC	AVON	
4	5	AVON BRILLIANT SHINE LIP GLOSS-ALL SHADES �	NaN	NaN	4	New Avon LLC	AVON	
...	...	...	...	...	...	...	...	
114630	41523	HYDRA-LIP TRANSLUCENT COLOR LIPSTICK	65001.0	Rosa Soft	1259	Yanbal USA, Inc	YANBAL	
114631	41523	HYDRA-LIP TRANSLUCENT COLOR LIPSTICK	65002.0	Malva Spirit	1259	Yanbal USA, Inc	YANBAL	
114632	41523	HYDRA-LIP TRANSLUCENT COLOR LIPSTICK	65003.0	Rojo Fashion	1259	Yanbal USA, Inc	YANBAL	
114633	41523	HYDRA-LIP TRANSLUCENT COLOR LIPSTICK	65004.0	Terra Mystic	1259	Yanbal USA, Inc	YANBAL	
114634	41524	OLD SPICE GENTLEMENS BLEND ALOE AND WILD SAGE ...	NaN	NaN	86	The Procter & Gamble Company	Old Spice	

114635 rows × 22 columns

Column Description

1)CDPHId: Likely an ID related to the California Department of Public Health (CDPH).

2)ProductName: The name of the product.

3)CSFId: Likely a unique ID for a CSF (could be related to a specific certification or program).

4)CSF: Information about the CSF, possibly indicating a certification, status, or category.

5)CompanyId: Unique identifier for the company.

6)CompanyName: Name of the company.

7)BrandName: Name of the brand associated with the product.

8)PrimaryCategoryId: ID for the primary category of the product.

9)PrimaryCategory: The main category of the product.

10)SubCategoryId: ID for the subcategory of the product.

11)SubCategory: The subcategory under which the product is listed.

12)CasId: Likely referring to an ID related to the Chemical Abstracts Service (CAS).

13)CasNumber: The CAS number associated with the product or chemical.

14)ChemicalId: Unique identifier for the chemical in the product. 15)ChemicalName: The name of the chemical.

16)InitialDateReported: The first date this product/chemical was reported.

17)MostRecentDateReported: The most recent date the product/chemical was reported.

18)DiscontinuedDate: The date when the product or chemical was discontinued.

19)ChemicalCreatedAt: The creation date of the chemical record.

20)ChemicalUpdatedAt: The last update date of the chemical record.

21)ChemicalDateRemoved: Date when the chemical was removed (from a database or list).

22)ChemicalCount: Likely the number of occurrences or items related to the chemical.

In [3]:

```
df.head()
```

Out[3]:

PrimaryCategory	SubCategoryId	...	CasNumber	ChemicalId	ChemicalName	InitialDateReported
Makeup Products (non-permanent)	53	...	13463-67-7	6	Titanium dioxide	06/17/2009
Hair Care Products (non-coloring)	25	...	65996-92-1	4	Distillates (coal tar)	07/01/2009
Hair Care Products (non-coloring)	25	...	140-67-0	5	Estragole	07/01/2009
Makeup Products (non-permanent)	46	...	13463-67-7	7	Titanium dioxide	07/09/2009
Makeup Products (non-permanent)	52	...	13463-67-7	8	Titanium dioxide	07/09/2009

In [4]:
```
df.tail()
```

Out[4]:

PrimaryCategory	SubCategoryId	...	CasNumber	ChemicalId	ChemicalName	InitialDateReported
Makeup Products (non-permanent)	53	...	13463-67-7	68059	Titanium dioxide	06/19/2020
Makeup Products (non-permanent)	53	...	13463-67-7	68060	Titanium dioxide	06/19/2020
Makeup Products (non-permanent)	53	...	13463-67-7	68061	Titanium dioxide	06/19/2020
Makeup Products (non-permanent)	53	...	13463-67-7	68062	Titanium dioxide	06/19/2020
Bath Products	159	...	13463-67-7	68074	Titanium dioxide	06/23/2020

In [5]:
```
df.shape
```

Out[5]:
```
(114635, 22)
```

In [6]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 114635 entries, 0 to 114634
Data columns (total 22 columns):
 #   Column                 Non-Null Count   Dtype  
---  ------                 --------------   -----  
 0   CDPHId                 114635 non-null  int64  
 1   ProductName            114635 non-null  object 
 2   CSFId                  80662 non-null   float64
 3   CSF                    80237 non-null   object 
 4   CompanyId              114635 non-null  int64  
 5   CompanyName            114635 non-null  object 
 6   BrandName              114408 non-null  object 
 7   PrimaryCategoryId      114635 non-null  int64  
 8   PrimaryCategory        114635 non-null  object 
 9   SubCategoryId          114635 non-null  int64  
 10  SubCategory            114635 non-null  object 
 11  CasId                  114635 non-null  int64  
 12  CasNumber              108159 non-null  object 
 13  ChemicalId             114635 non-null  int64  
 14  ChemicalName           114635 non-null  object 
 15  InitialDateReported    114635 non-null  object 
 16  MostRecentDateReported  114635 non-null  object 
 17  DiscontinuedDate       12920 non-null   object 
 18  ChemicalCreatedAt      114635 non-null  object 
 19  ChemicalUpdatedAt      114635 non-null  object 
 20  ChemicalDateRemoved    2985 non-null    object 
 21  ChemicalCount          114635 non-null  int64  
dtypes: float64(1), int64(7), object(14)
memory usage: 19.2+ MB
```

In [170]:

```
df.columns
```

Out[170]:

```
Index(['ProductName', 'CompanyName', 'BrandName', 'PrimaryCategoryId',
       'PrimaryCategory', 'SubCategoryId', 'SubCategory', 'CasId', 'CasNumbe
r',
       'ChemicalId', 'ChemicalName', 'InitialDateReported',
       'MostRecentDateReported', 'ChemicalCreatedAt', 'ChemicalUpdatedAt',
       'ChemicalCount'],
      dtype='object')
```

In [9]:
```
df.describe().T
```

Out[9]:

	count	mean	std	min	25%	50%	75%	m
CDPHId	114635.0	20304.858987	12489.052554	2.0	8717.0	20895.0	31338.50	41524
CSFId	80662.0	32608.658377	19089.443910	1.0	15789.0	32541.0	48717.75	65009
CompanyId	114635.0	450.641532	409.533093	4.0	86.0	297.0	798.00	1391
PrimaryCategoryId	114635.0	51.076294	20.474341	1.0	44.0	44.0	59.00	111
SubCategoryId	114635.0	66.819252	35.822097	3.0	48.0	52.0	65.00	172
CasId	114635.0	674.094107	149.214101	2.0	656.0	656.0	656.00	1242
ChemicalId	114635.0	32837.556959	20439.412299	0.0	13990.0	32055.0	51578.50	68074
ChemicalCount	114635.0	1.288359	0.636418	0.0	1.0	1.0	1.00	9

In [44]:
```
df.describe(include="object")
```

Out[44]:

	ProductName	CSF	CompanyName	BrandName	PrimaryCategory	SubCategory	CasNu
count	114635	80237	114635	114408	114635	114635	1
unique	33716	34326	606	2713	13	89	
top	Eyecolor	Black	L'Oreal USA	SEPHORA	Makeup Products (non-permanent)	Lip Color - Lipsticks, Liners, and Pencils	1346
freq	766	247	5747	3394	75827	16555	

In [26]:
```
df.duplicated().sum()
```

Out[26]:
```
254
```

In [27]:
```
df.drop_duplicates()
```

Out[27]:

	CDPHId	ProductName	CSFId	CSF	CompanyId	CompanyName	BrandName	Prir
0	2	ULTRA COLOR RICH EXTRA PLUMP LIPSTICK-ALL SHADES	NaN	NaN	4	New Avon LLC	AVON	
1	3	Glover's Medicated Shampoo	NaN	NaN	338	J. Strickland & Co.	Glover's	
2	3	Glover's Medicated Shampoo	NaN	NaN	338	J. Strickland & Co.	Glover's	
3	4	PRECISION GLIMMER EYE LINER-ALL SHADES �	NaN	NaN	4	New Avon LLC	AVON	
4	5	AVON BRILLIANT SHINE LIP GLOSS-ALL SHADES �	NaN	NaN	4	New Avon LLC	AVON	
...	...	...	...	...	...	...	...	
114630	41523	HYDRA-LIP TRANSLUCENT COLOR LIPSTICK	65001.0	Rosa Soft	1259	Yanbal USA, Inc	YANBAL	
114631	41523	HYDRA-LIP TRANSLUCENT COLOR LIPSTICK	65002.0	Malva Spirit	1259	Yanbal USA, Inc	YANBAL	
114632	41523	HYDRA-LIP TRANSLUCENT COLOR LIPSTICK	65003.0	Rojo Fashion	1259	Yanbal USA, Inc	YANBAL	
114633	41523	HYDRA-LIP TRANSLUCENT COLOR LIPSTICK	65004.0	Terra Mystic	1259	Yanbal USA, Inc	YANBAL	
114634	41524	OLD SPICE GENTLEMENS BLEND ALOE AND WILD SAGE ...	NaN	NaN	86	The Procter & Gamble Company	Old Spice	

114381 rows × 22 columns

In [16]:
```
df.value_counts("ChemicalCount")# Target Columns
```

Out[16]:
```
ChemicalCount
1    87267
2    21266
3     3528
4     1481
0      869
5      105
8       41
7       36
6       33
9        9
Name: count, dtype: int64
```

In [56]:
```
df.isnull().sum()
```

Out[56]:
```
CDPHId                       0
ProductName                  0
CSFId                    33973
CSF                      34398
CompanyId                    0
CompanyName                  0
BrandName                  227
PrimaryCategoryId            0
PrimaryCategory              0
SubCategoryId                0
SubCategory                  0
CasId                        0
CasNumber                 6476
ChemicalId                   0
ChemicalName                 0
InitialDateReported          0
MostRecentDateReported       0
DiscontinuedDate        101715
ChemicalCreatedAt            0
ChemicalUpdatedAt            0
ChemicalDateRemoved     111650
ChemicalCount                0
dtype: int64
```

In [28]:

```
df.drop(columns=["CDPHId","CSFId","CSF","DiscontinuedDate","ChemicalDateRemove
df
```

Out[28]:

	ProductName	CompanyName	BrandName	PrimaryCategoryId	PrimaryCategory	SubCa
0	ULTRA COLOR RICH EXTRA PLUMP LIPSTICK-ALL SHADES	New Avon LLC	AVON	44	Makeup Products (non-permanent)	
1	Glover's Medicated Shampoo	J. Strickland & Co.	Glover's	18	Hair Care Products (non-coloring)	
2	Glover's Medicated Shampoo	J. Strickland & Co.	Glover's	18	Hair Care Products (non-coloring)	
3	PRECISION GLIMMER EYE LINER-ALL SHADES �	New Avon LLC	AVON	44	Makeup Products (non-permanent)	
4	AVON BRILLIANT SHINE LIP GLOSS-ALL SHADES �	New Avon LLC	AVON	44	Makeup Products (non-permanent)	
...	...	...	...	...	...	
114630	HYDRA-LIP TRANSLUCENT COLOR LIPSTICK	Yanbal USA, Inc	YANBAL	44	Makeup Products (non-permanent)	
114631	HYDRA-LIP TRANSLUCENT COLOR LIPSTICK	Yanbal USA, Inc	YANBAL	44	Makeup Products (non-permanent)	
114632	HYDRA-LIP TRANSLUCENT COLOR LIPSTICK	Yanbal USA, Inc	YANBAL	44	Makeup Products (non-permanent)	
114633	HYDRA-LIP TRANSLUCENT COLOR LIPSTICK	Yanbal USA, Inc	YANBAL	44	Makeup Products (non-permanent)	
114634	OLD SPICE GENTLEMENS BLEND ALOE AND WILD SAGE ...	The Procter & Gamble Company	Old Spice	6	Bath Products	

114635 rows × 16 columns

In [29]:

```
df["BrandName"].fillna("Merle Norman",inplace=True)
```

In [30]:
```
df['CasNumber'].fillna('Unknown',inplace=True)
```

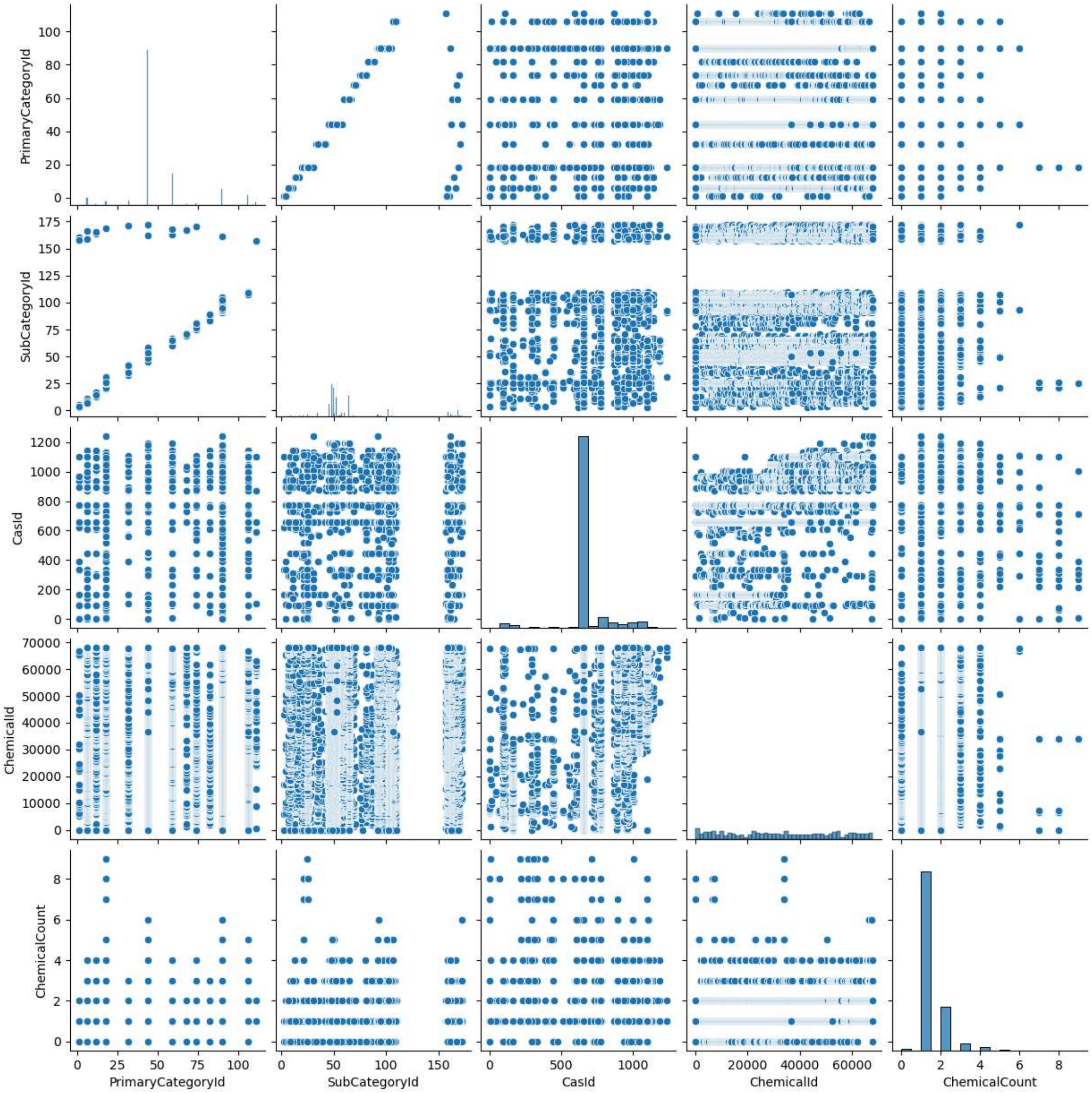
In [69]:
```
df.isnull().sum()
```

Out[69]:
```
ProductName             0
CompanyName             0
BrandName               0
PrimaryCategoryId       0
PrimaryCategory         0
SubCategoryId           0
SubCategory             0
CasId                   0
CasNumber               0
ChemicalId              0
ChemicalName            0
InitialDateReported     0
MostRecentDateReported  0
ChemicalCreatedAt       0
ChemicalUpdatedAt       0
ChemicalCount           0
dtype: int64
```

In [31]:
```
date_columns = ['InitialDateReported','MostRecentDateReported','ChemicalCreate
for col in date_columns:
    df[col] = pd.to_datetime(df[col])
df["MostRecentDateReported"]
```

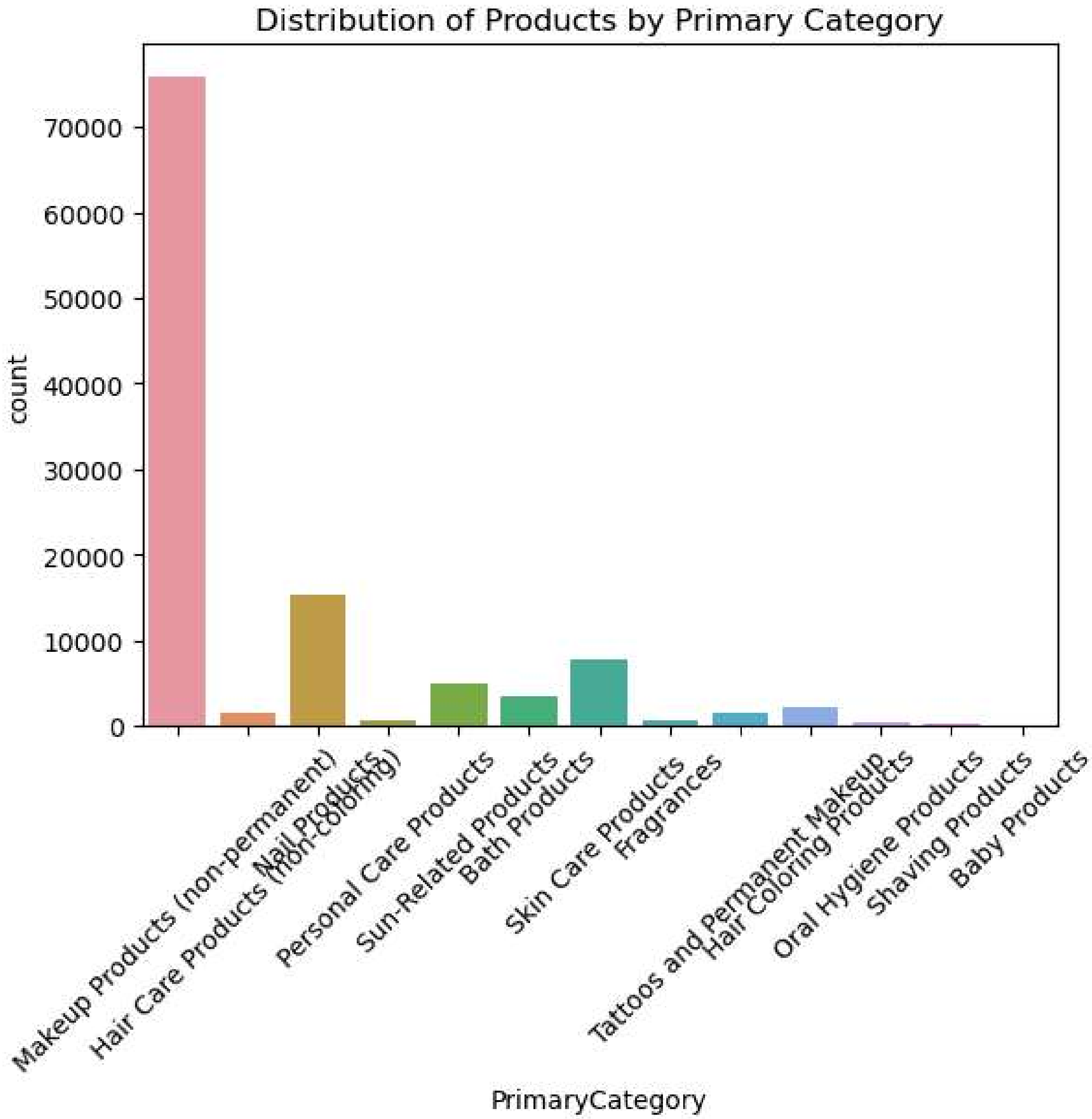
Out[31]:
```
0        2013-08-28
1        2009-07-01
2        2009-07-01
3        2013-08-28
4        2013-08-28
            ...
114630   2020-06-19
114631   2020-06-19
114632   2020-06-19
114633   2020-06-19
114634   2020-06-23
Name: MostRecentDateReported, Length: 114635, dtype: datetime64[ns]
```

In [81]:
```
sns.pairplot(df)
plt.show()
```

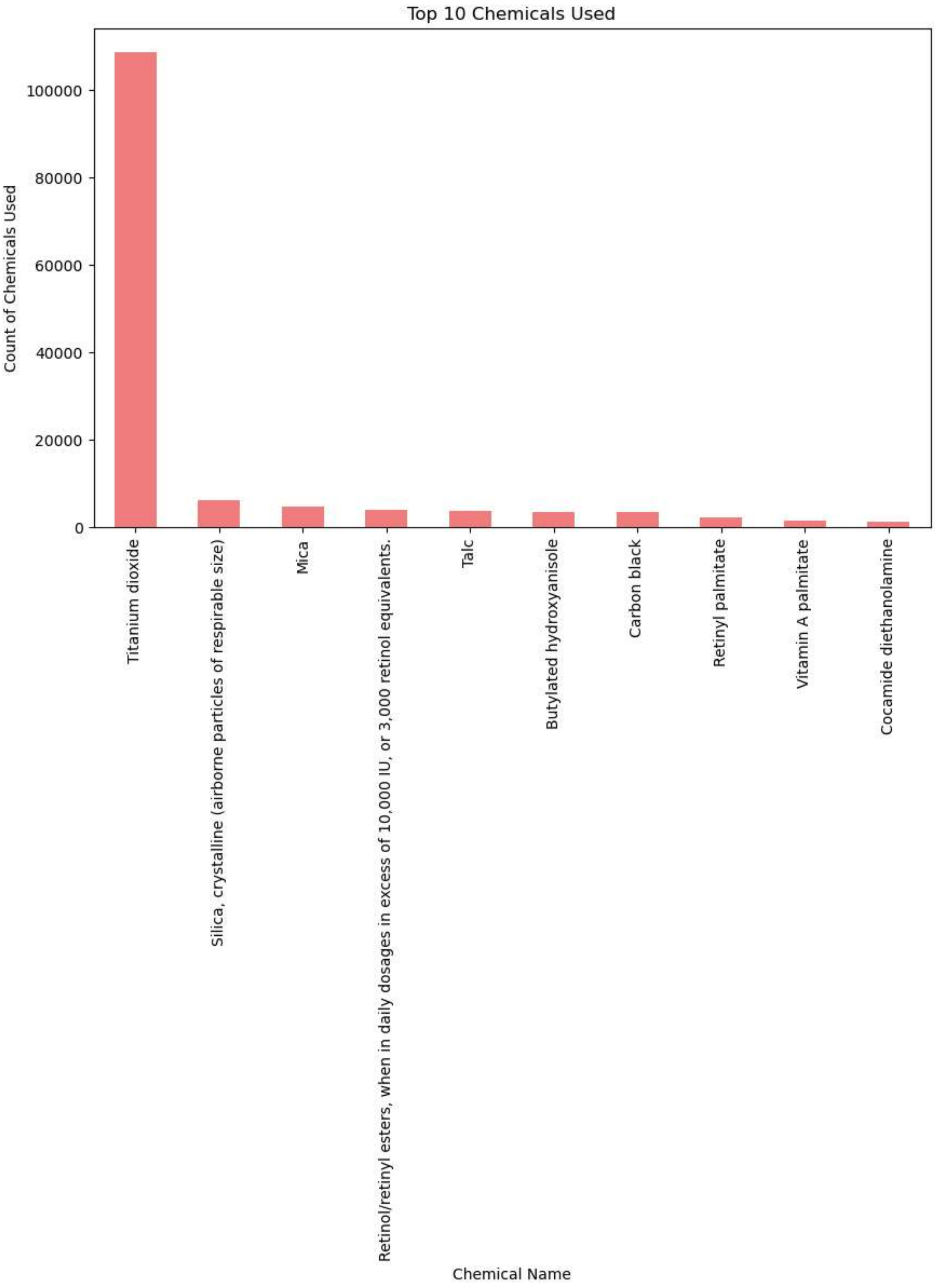


In [33]:

```
sns.countplot(x='PrimaryCategory', data=df)
plt.xticks(rotation=45)
plt.title('Distribution of Products by Primary Category')
plt.show()
```

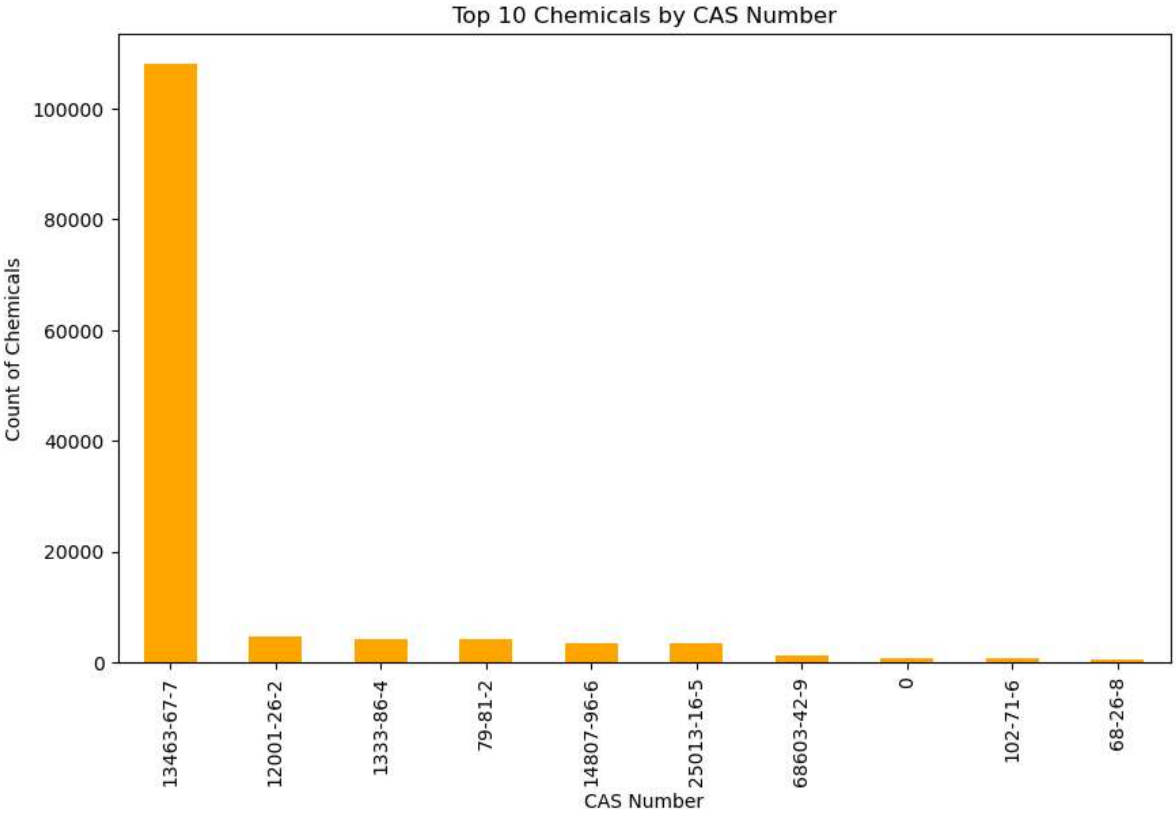


In [3]:
```
plt.figure(figsize=(10, 6))
chemical_counts = df.groupby('ChemicalName')['ChemicalCount'].sum().sort_value
chemical_counts.plot(kind='bar', color='lightcoral')
plt.title('Top 10 Chemicals Used')
plt.xlabel('Chemical Name')
plt.ylabel('Count of Chemicals Used')
plt.show()
```



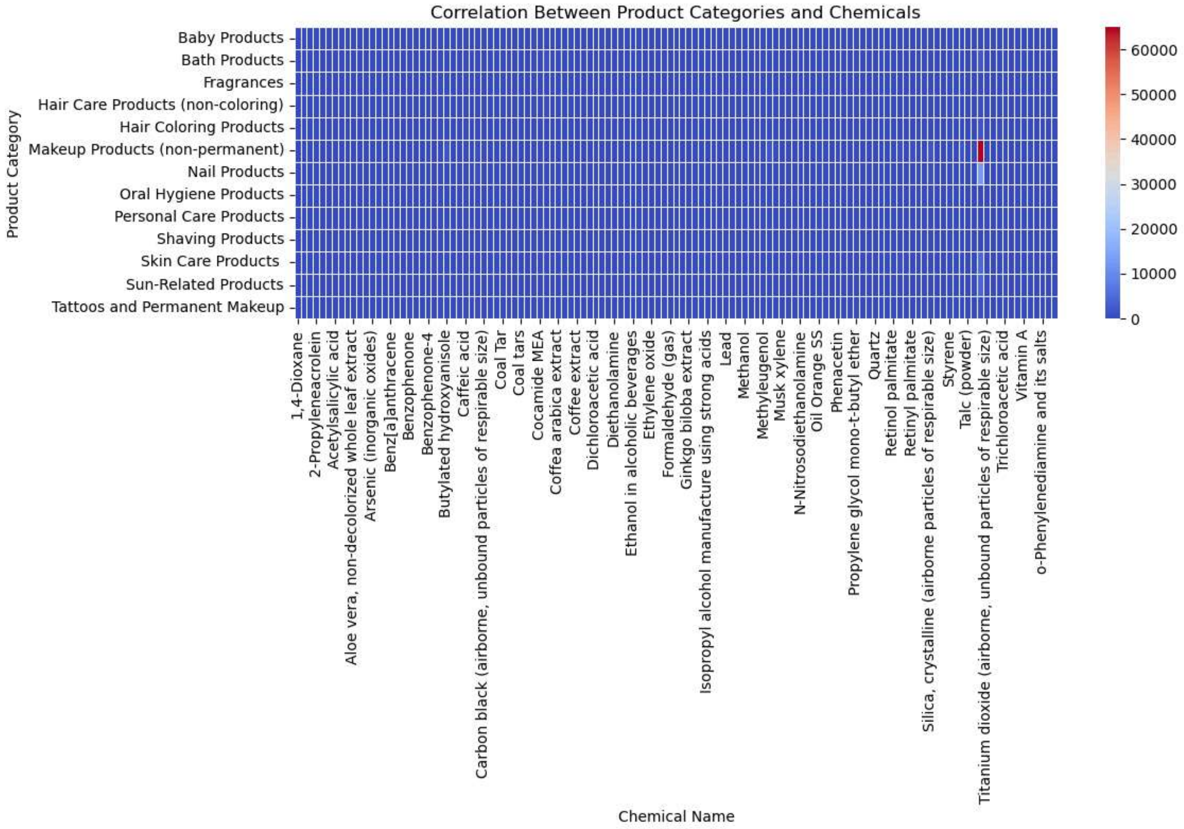
In [4]:

```
plt.figure(figsize=(10, 6))
cas_counts = df.groupby('CasNumber')['ChemicalCount'].sum().sort_values(ascend
cas_counts.plot(kind='bar', color='orange')
plt.title('Top 10 Chemicals by CAS Number')
plt.xlabel('CAS Number')
plt.ylabel('Count of Chemicals')
plt.show()
```

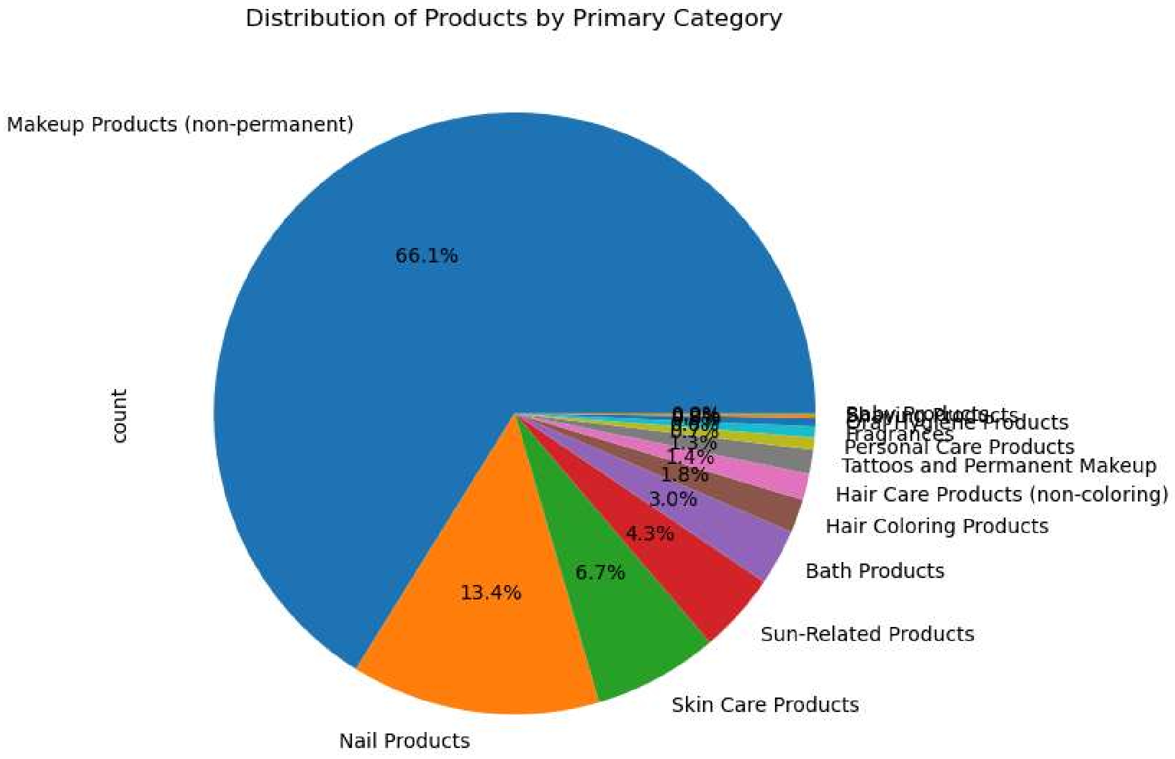


In [6]:

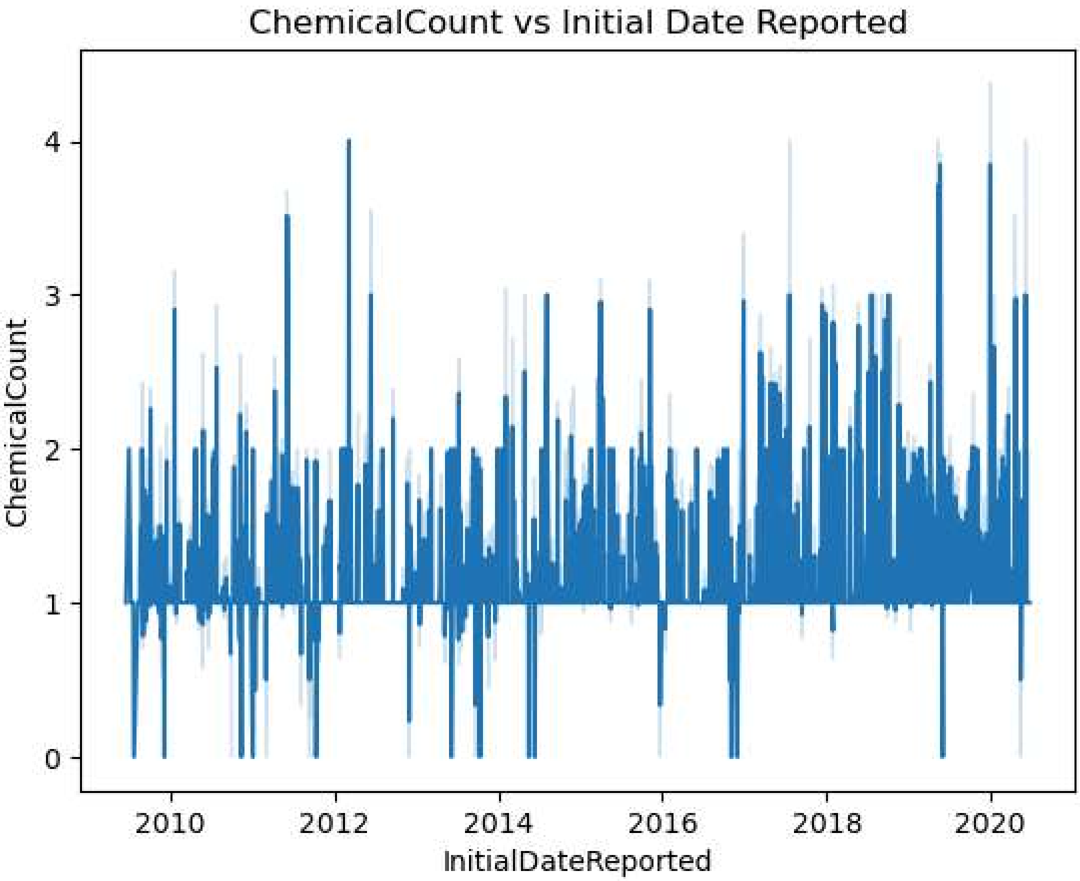
```
category_chemical_counts = df.groupby(['PrimaryCategory', 'ChemicalName']).siz
plt.figure(figsize=(12, 8))
sns.heatmap(category_chemical_counts, cmap="coolwarm", annot=False, fmt="d", l
plt.title('Correlation Between Product Categories and Chemicals')
plt.xlabel('Chemical Name')
plt.ylabel('Product Category')
plt.tight_layout()
plt.show()
```



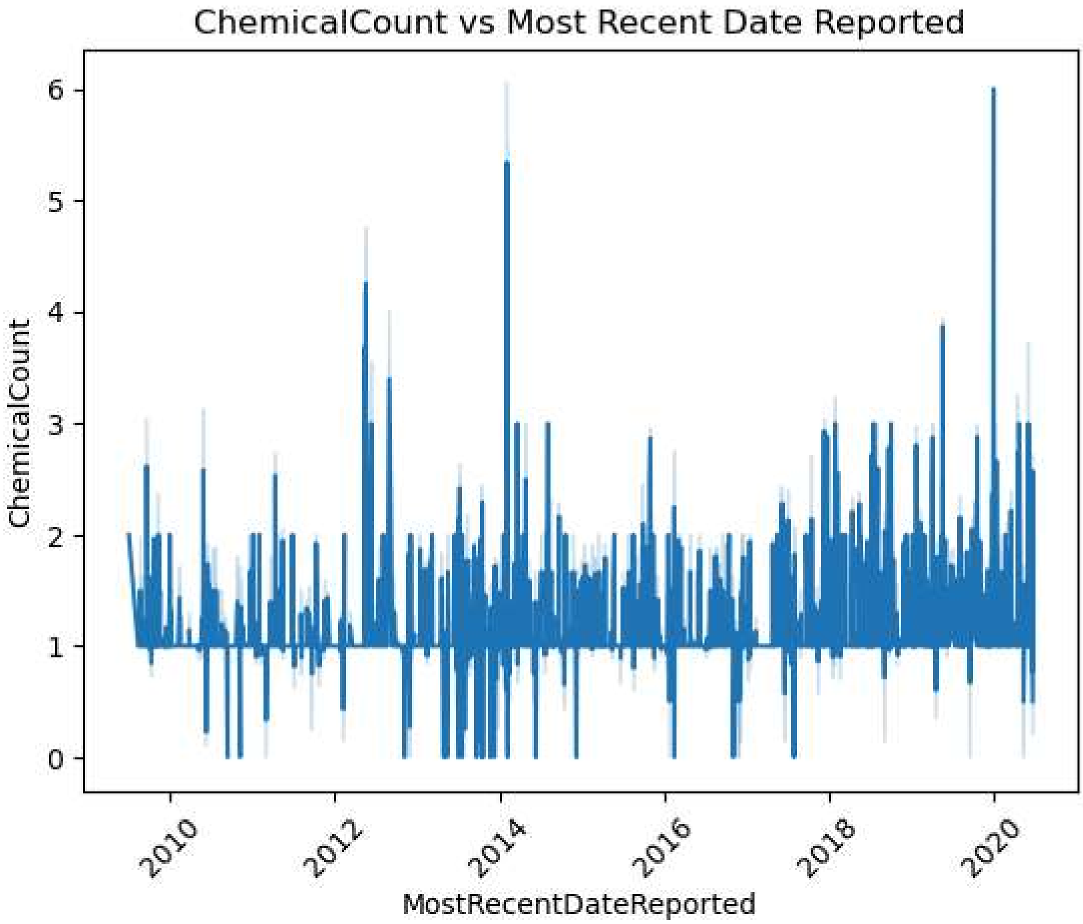
In [104]:
```
plt.figure(figsize=(8,8))
df['PrimaryCategory'].value_counts().plot(kind='pie', autopct='%1.1f%%')
plt.title('Distribution of Products by Primary Category')
plt.tight_layout()
plt.show()
```



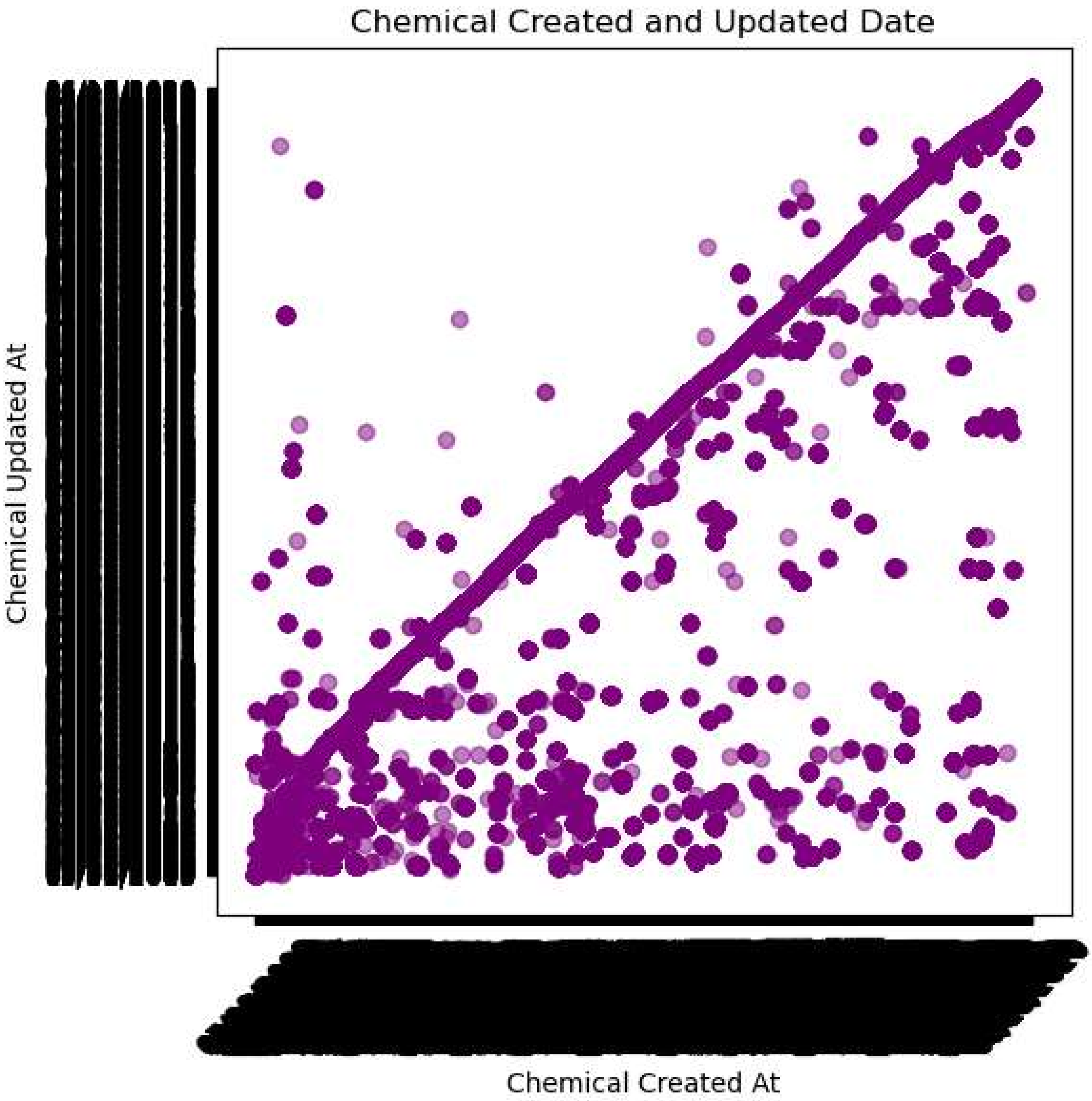
In [84]:
```
sns.lineplot(x='InitialDateReported', y='ChemicalCount', data=df)
plt.title('ChemicalCount vs Initial Date Reported')
plt.show()
```



```
In [86]: sns.lineplot(x='MostRecentDateReported', y='ChemicalCount', data=df)
         plt.title('ChemicalCount vs Most Recent Date Reported')
         plt.xticks(rotation=45)
         plt.show()
```



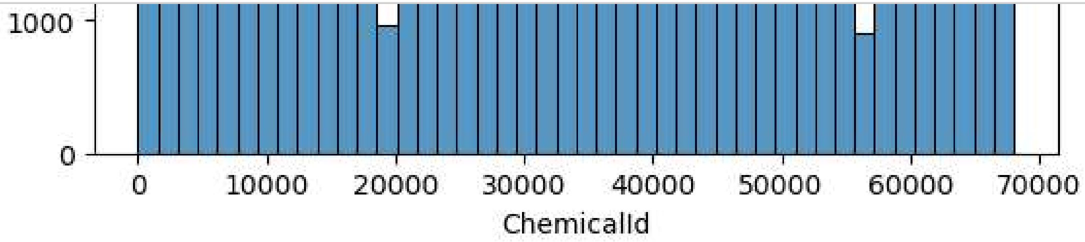
In [17]:
```
plt.figure(figsize=(6, 6))
plt.scatter(df['ChemicalCreatedAt'], df['ChemicalUpdatedAt'], alpha=0.5, c='pu
plt.title('Chemical Created and Updated Date')
plt.xlabel('Chemical Created At')
plt.ylabel('Chemical Updated At')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



In [98]:
```
df.select_dtypes(include="number").columns
```

Out[98]:
```
Index(['PrimaryCategoryId', 'SubCategoryId', 'CasId', 'ChemicalId',
       'ChemicalCount'],
      dtype='object')
```

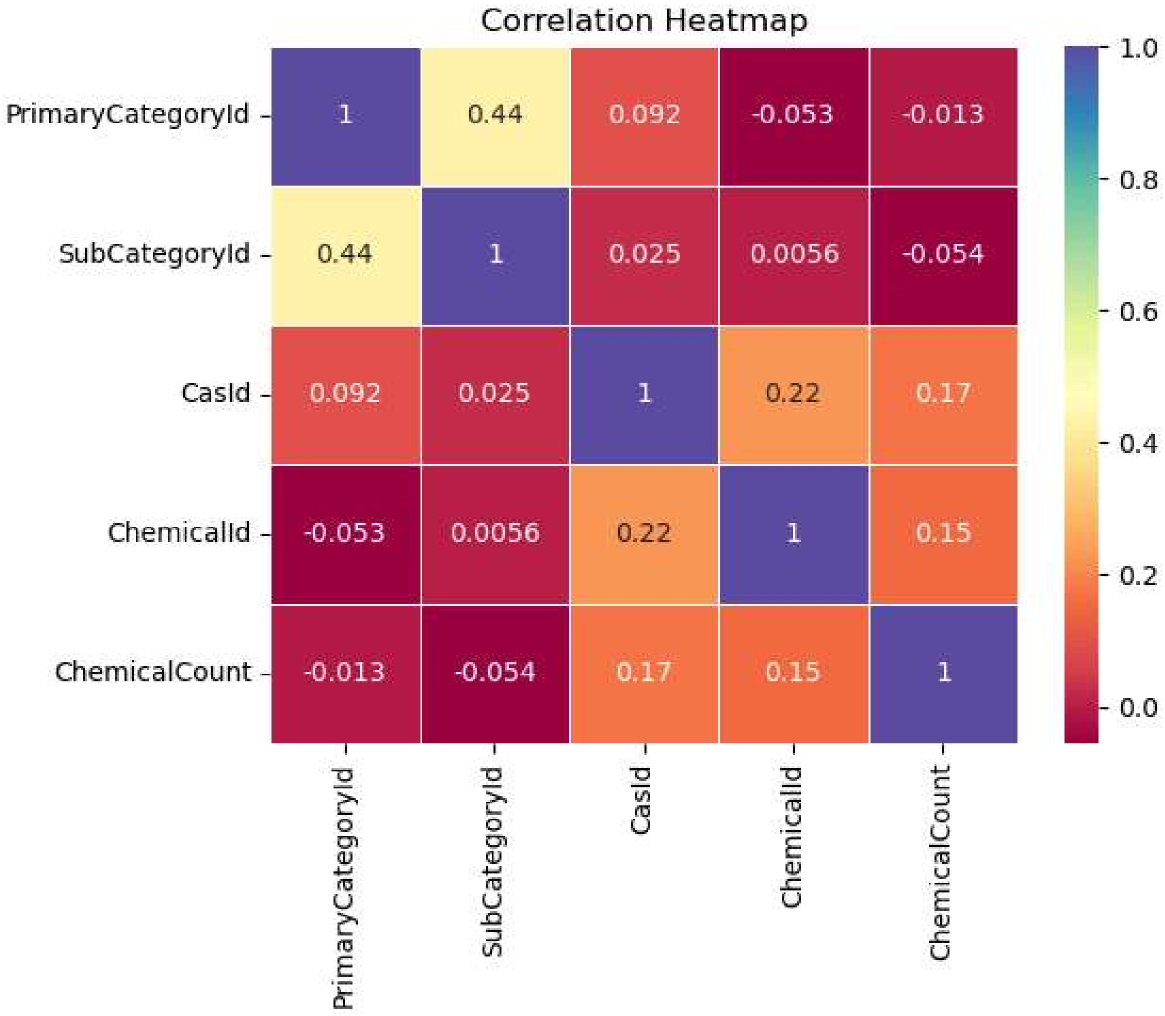
```
In [97]: for i in df.select_dtypes(include="number").columns:
             sns.histplot(data=df,x=i)
             plt.show()
```



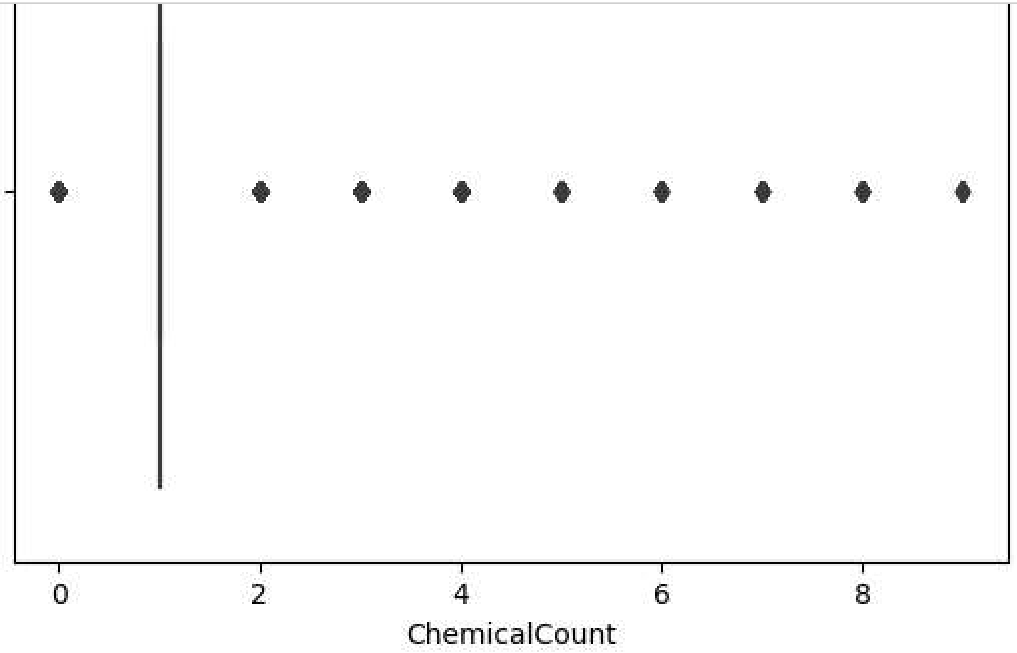
80000
60000
ount

In [102]:

```
s=df.select_dtypes(include="number").corr()
sns.heatmap(s,annot=True, cmap='Spectral', linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()
```



```
In [105]: for i in df.select_dtypes(include="number").columns:
              sns.boxplot(data=df,x=i)
              plt.show()
```



```
In [129]: df.select_dtypes(include=['object']).columns
```

```
Out[129]: Index(['ProductName', 'CompanyName', 'BrandName', 'PrimaryCategory',
                 'SubCategory', 'CasNumber', 'ChemicalName'],
                dtype='object')
```

```
In [51]: df.select_dtypes(include="number").columns
```

```
Out[51]: Index(['PrimaryCategoryId', 'SubCategoryId', 'CasId', 'ChemicalId',
                'ChemicalCount'],
               dtype='object')
```

```
In [87]: from sklearn.preprocessing import LabelEncoder
```

```
In [88]: encoder=LabelEncoder()
```

```
In [89]: data=df.copy()
         data['ProductName']=encoder.fit_transform(data['ProductName'])
         data['CompanyName']=encoder.fit_transform(data['CompanyName'])
         data['BrandName']=encoder.fit_transform(data['BrandName'])
         data['PrimaryCategory']=encoder.fit_transform(data['PrimaryCategory'])
         data['SubCategory']=encoder.fit_transform(data['SubCategory'])
         data['ChemicalName']=encoder.fit_transform(data['ChemicalName'])
         data['CasNumber']=encoder.fit_transform(data['CasNumber'])
```

In [91]:
```
data.drop(columns=['InitialDateReported','MostRecentDateReported','ChemicalCre
data
```

Out[91]:

	ProductName	CompanyName	BrandName	PrimaryCategoryId	PrimaryCategory	SubCate
0	30981	372	82	44	5	
1	13104	252	1023	18	3	
2	13104	252	1023	18	3	
3	22843	372	82	44	5	
4	1328	372	82	44	5	
...	...	...	...	...	...	
114630	13606	579	2578	44	5	
114631	13606	579	2578	44	5	
114632	13606	579	2578	44	5	
114633	13606	579	2578	44	5	
114634	21965	527	1729	6	1	

114635 rows × 10 columns

In [92]:
```
from sklearn.feature_selection import SelectKBest,chi2
from sklearn.feature_selection import f_classif
```

In [109]:
```
x = data.drop(columns=['ChemicalCount'])
y = data["ChemicalCount"]
selector = SelectKBest(score_func=chi2, k=8)
selector.fit_transform(x, y)
datascores=pd.DataFrame(selector.scores_,columns=["Scores"])
datacolumns=pd.DataFrame(x.columns.tolist(),columns=["features"])
result=pd.concat([datacolumns,datascores,],axis=1)
result.sort_values(by="Scores", ascending=False)
```

Out[109]:

	features	Scores
0	ProductName	2.272595e+06
6	CasId	1.841480e+05
7	CasNumber	1.654176e+05
1	CompanyName	7.268965e+04
8	ChemicalName	5.831714e+04
2	BrandName	4.462764e+04
5	SubCategory	6.826602e+03
3	PrimaryCategoryId	4.297863e+03
4	PrimaryCategory	2.907236e+02

In [112]:
```
selected_features = selector.get_support(indices=True)
x_selected = x.iloc[:, selected_features]
```

In [111]:
```
from sklearn.model_selection import train_test_split
```

In [113]:
```
x=data.drop(columns="ChemicalCount")
y=data["ChemicalCount"]
x_train,x_test,y_train,y_test=train_test_split(x_selected,y,test_size=0.2,rand
```

In [115]:
```
from sklearn.preprocessing import StandardScaler
```

In [116]:
```
scaler = StandardScaler()
```

In [117]:
```
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)
```

In [118]:
```
from sklearn.linear_model import LogisticRegression
```

In [119]:
```
model = LogisticRegression()
```

In [120]:
```
model.fit(x_train_scaled, y_train)
```

Out[120]:
```
▾ LogisticRegression
LogisticRegression()
```

In [121]:
```
y_pred=model.predict(x_test_scaled)
```

In [122]:
```
model.score(x_test_scaled,y_test)
```

Out[122]:
```
0.7760718803157849
```

In [123]:
```
from sklearn.metrics import accuracy_score
```

In [124]:
```
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.4f}")
```

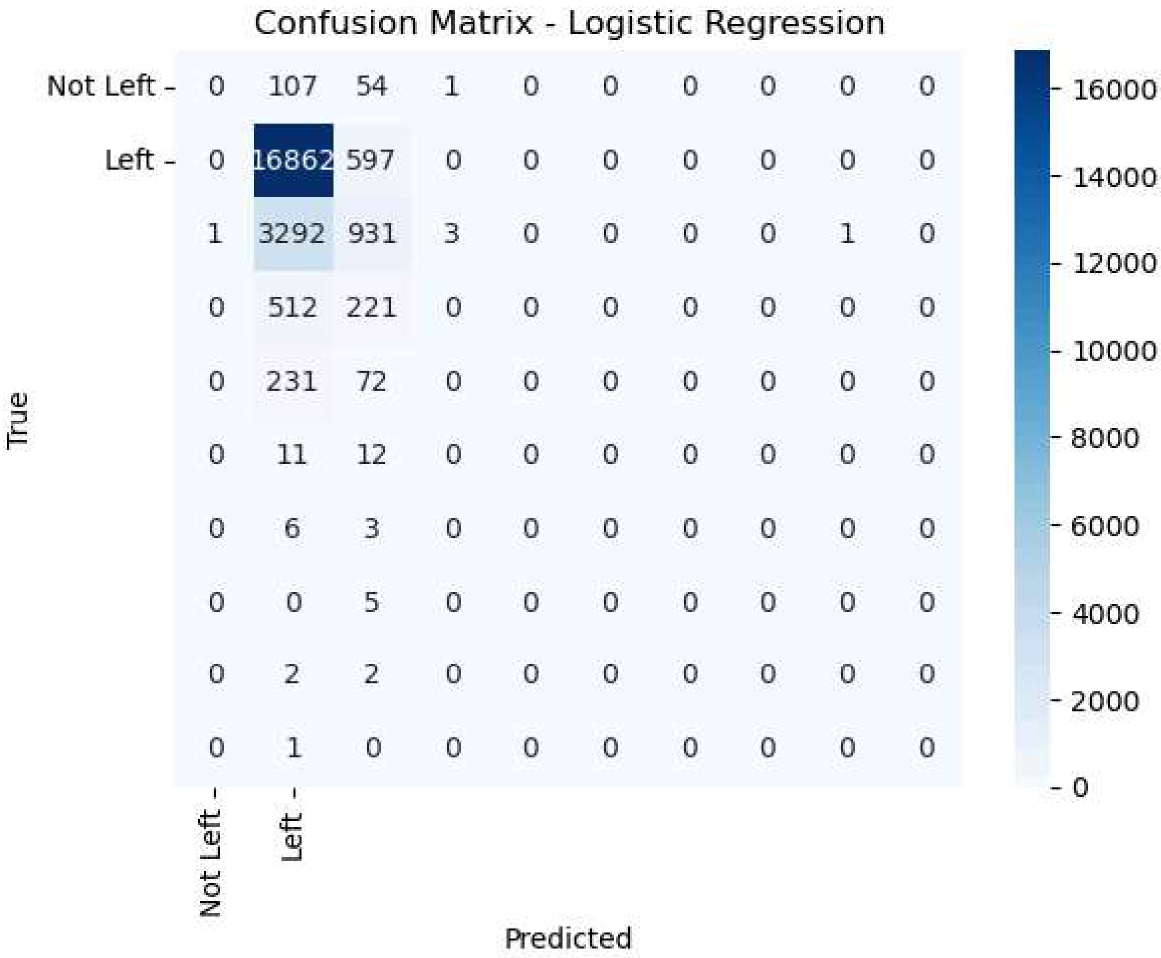
```
Accuracy: 0.7761
```

In [104]:
```
from sklearn.metrics import confusion_matrix
```

In [125]:
```
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm)
```

```
Confusion Matrix:
 [[    0   107    54     1     0     0     0     0     0     0]
 [    0 16862   597     0     0     0     0     0     0     0]
 [    1  3292   931     3     0     0     0     0     1     0]
 [    0   512   221     0     0     0     0     0     0     0]
 [    0   231    72     0     0     0     0     0     0     0]
 [    0    11    12     0     0     0     0     0     0     0]
 [    0     6     3     0     0     0     0     0     0     0]
 [    0     0     5     0     0     0     0     0     0     0]
 [    0     2     2     0     0     0     0     0     0     0]
 [    0     1     0     0     0     0     0     0     0     0]]
```

In [126]:
```
# Visualize Confusion Matrix
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Not Left', 'L
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix - Logistic Regression')
plt.show()
```



```
In [127]: from sklearn.metrics import classification_report
          print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       162
           1       0.80      0.97      0.88     17459
           2       0.49      0.22      0.30      4228
           3       0.00      0.00      0.00       733
           4       0.00      0.00      0.00       303
           5       0.00      0.00      0.00        23
           6       0.00      0.00      0.00         9
           7       0.00      0.00      0.00         5
           8       0.00      0.00      0.00         4
           9       0.00      0.00      0.00         1

    accuracy                           0.78     22927
   macro avg       0.13      0.12      0.12     22927
weighted avg       0.70      0.78      0.72     22927
```

```
In [130]: from sklearn.svm import SVC
```

```
In [133]: model_2=SVC()
          model_2.fit(x_train_scaled, y_train)
```

```
Out[133]: ▾ SVC
          SVC()
```

```
In [134]: y_pred=model_2.predict(x_test_scaled)
          model_2.score(x_test_scaled,y_test)
```

```
Out[134]: 0.8371352553757578
```

```
In [136]: accuracy_2 = accuracy_score(y_test, y_pred)
          print(f"Accuracy: {accuracy_2:.4f}")
```

```
Accuracy: 0.8371
```

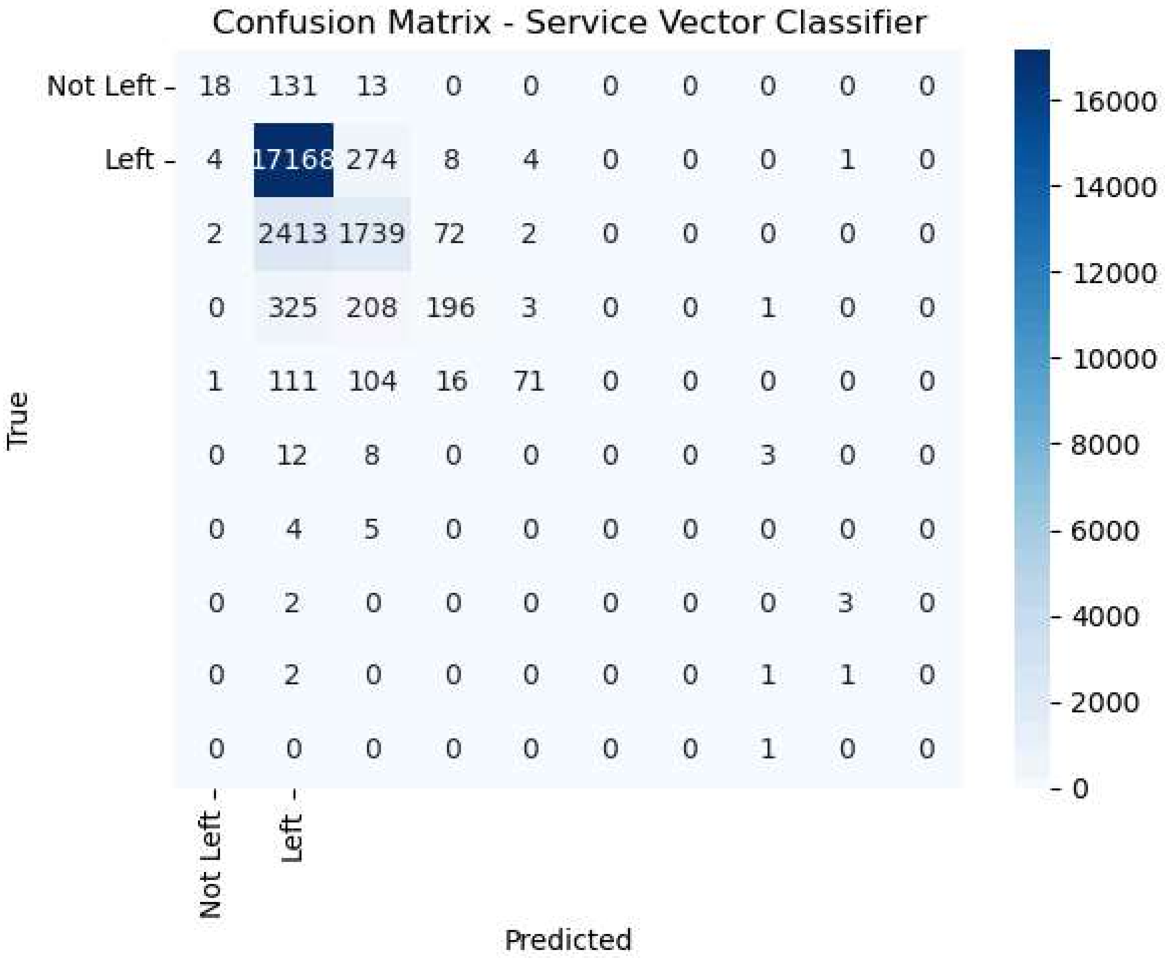
In [137]:

```
cm_svm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm_svm)
```

```
Confusion Matrix:
 [[   18   131    13     0     0     0     0     0     0     0]
 [    4 17168   274     8     4     0     0     0     1     0]
 [    2  2413  1739    72     2     0     0     0     0     0]
 [    0   325   208   196     3     0     0     1     0     0]
 [    1   111   104    16    71     0     0     0     0     0]
 [    0    12     8     0     0     0     0     3     0     0]
 [    0     4     5     0     0     0     0     0     0     0]
 [    0     2     0     0     0     0     0     0     3     0]
 [    0     2     0     0     0     0     0     1     1     0]
 [    0     0     0     0     0     0     0     1     0     0]]
```

In [140]:

```
# Visualize Confusion Matrix
sns.heatmap(cm_svm, annot=True, fmt='d', cmap='Blues', xticklabels=['Not Left'
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix - Service Vector Classifier')
plt.show()
```



In [139]:

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.72      0.11      0.19       162
           1       0.85      0.98      0.91     17459
           2       0.74      0.41      0.53      4228
           3       0.67      0.27      0.38       733
           4       0.89      0.23      0.37       303
           5       0.00      0.00      0.00        23
           6       0.00      0.00      0.00         9
           7       0.00      0.00      0.00         5
           8       0.20      0.25      0.22         4
           9       0.00      0.00      0.00         1

    accuracy                           0.84     22927
   macro avg       0.41      0.23      0.26     22927
weighted avg       0.82      0.84      0.81     22927
```

In [141]:

```
from sklearn.neighbors import KNeighborsClassifier
```

In [142]:

```
knn = KNeighborsClassifier()
```

In [143]:

```
knn.fit(x_train_scaled, y_train)
```

Out[143]:

```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

In [144]:

```
y_pred=knn.predict(x_test_scaled)
knn.score(x_test_scaled,y_test)
```

Out[144]:

```
0.9265058664456755
```

In [145]:

```
accuracy_3 = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy_3:.4f}")
```

```
Accuracy: 0.9265
```

In [ ]: