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 [85]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
df=pd.read_csv("C:/Users/user/Downloads/cscpopendata.csv")
df
```

Out[2]:

	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]:

	CDPHId	ProductName	CSFId	CSF	CompanyId	CompanyName	BrandName	PrimaryCategor
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	

5 rows × 22 columns

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

Out[4]:

	CDPHId	ProductName	CSFId	CSF	CompanyId	CompanyName	BrandName	Prir
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	

5 rows × 22 columns

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 [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 [10]:
```
df.duplicated().sum()
```

Out[10]:
```
254
```

In [12]:
```
df.drop_duplicates()
```

Out[12]:

	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 [63]:
```
df.drop(columns=["CDPHId","CSFId","CSF","DiscontinuedDate","ChemicalDateRemove
df
```

Out[63]:

	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 [70]:
```
df["BrandName"].fillna("Merle Norman",inplace=True)
```

In [79]:
```
df['CasNumber'].fillna('Unknown',inplace=True)
```

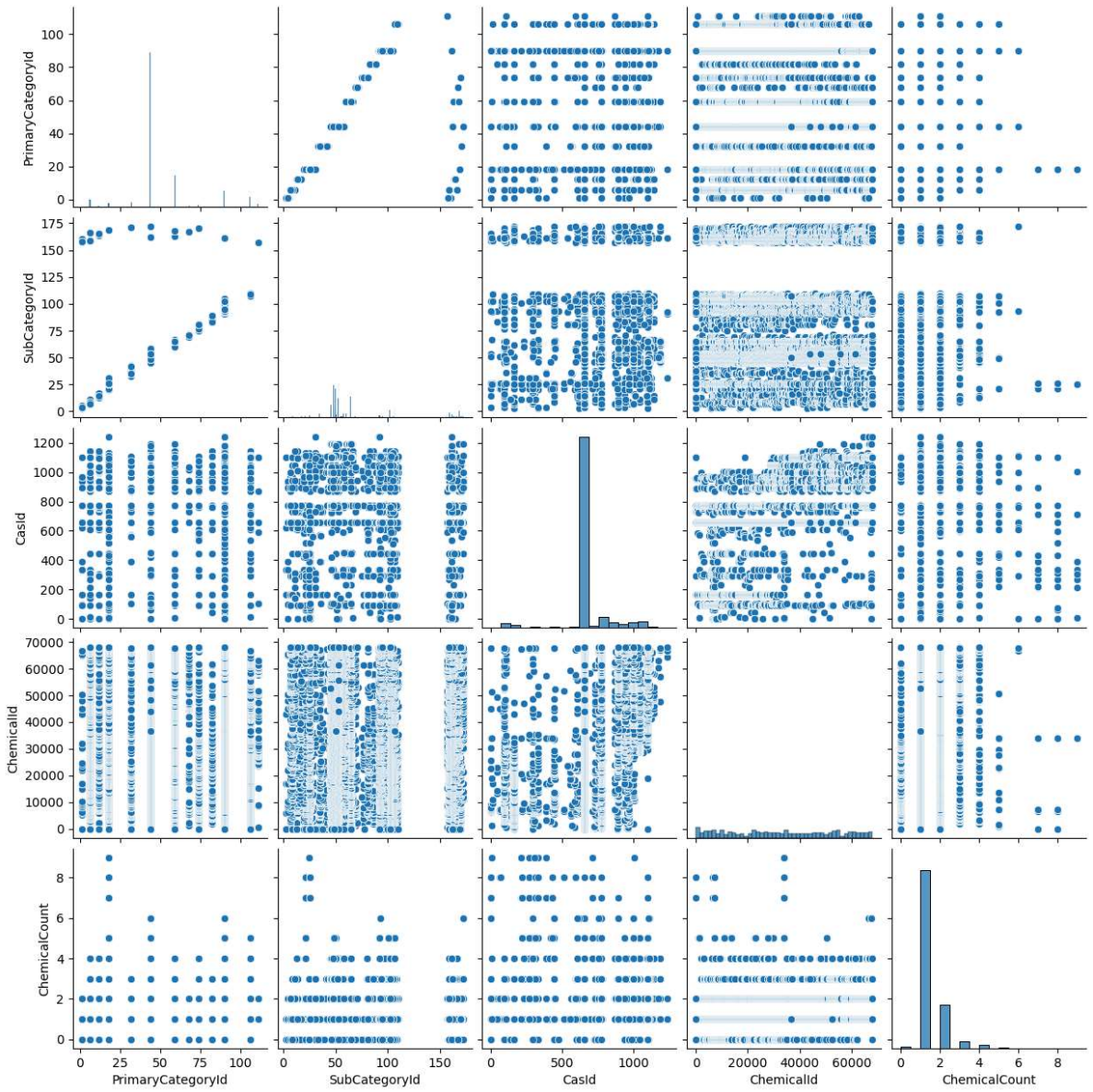
In [80]:
```
df.isnull().sum()
```

Out[80]:
```
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
```

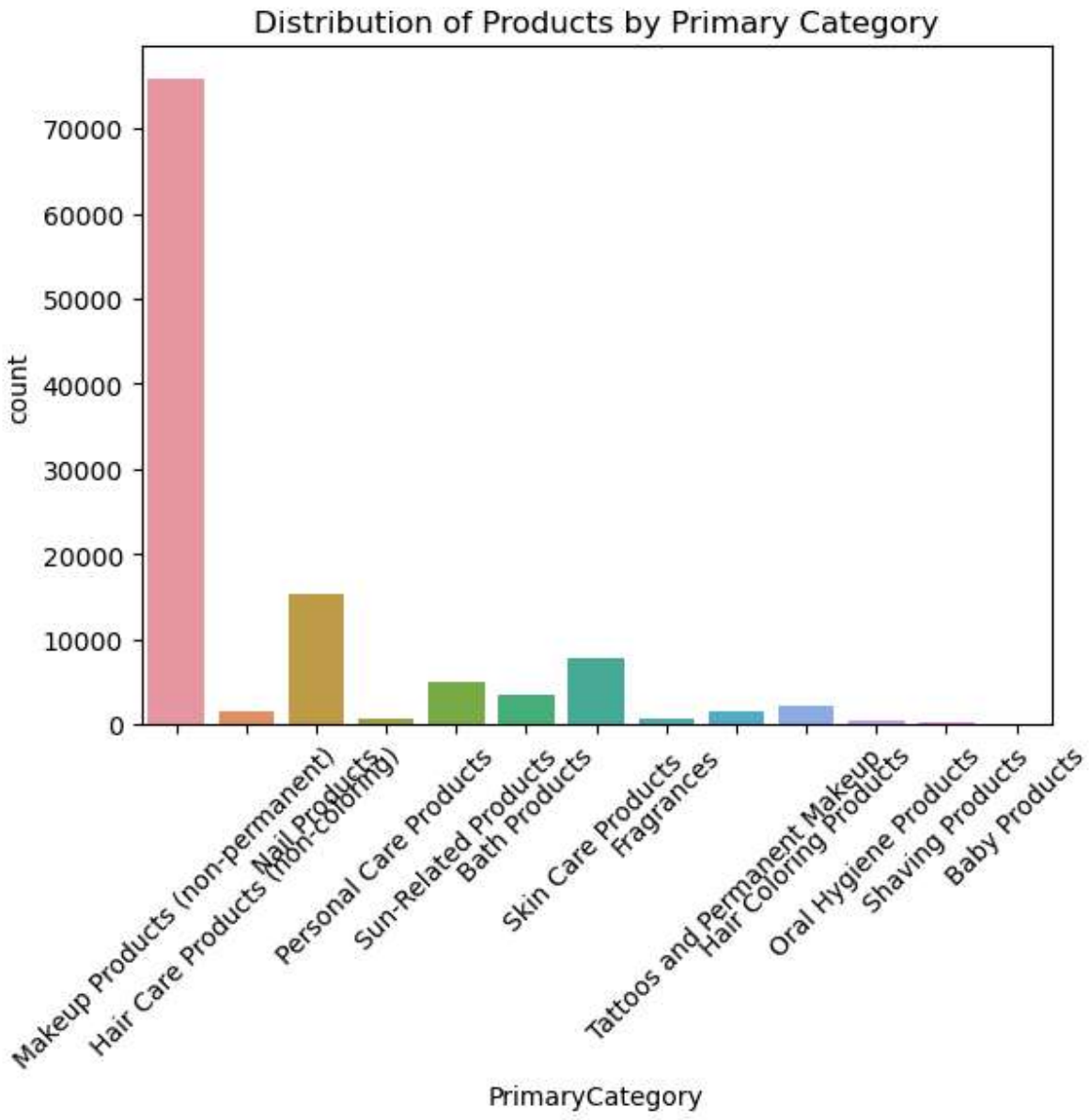
In [83]:
```
date_columns = ['InitialDateReported','MostRecentDateReported','ChemicalCreate
for col in date_columns:
    df[col] = pd.to_datetime(df[col])
df["MostRecentDateReported"]
```

Out[83]:
```
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]
```

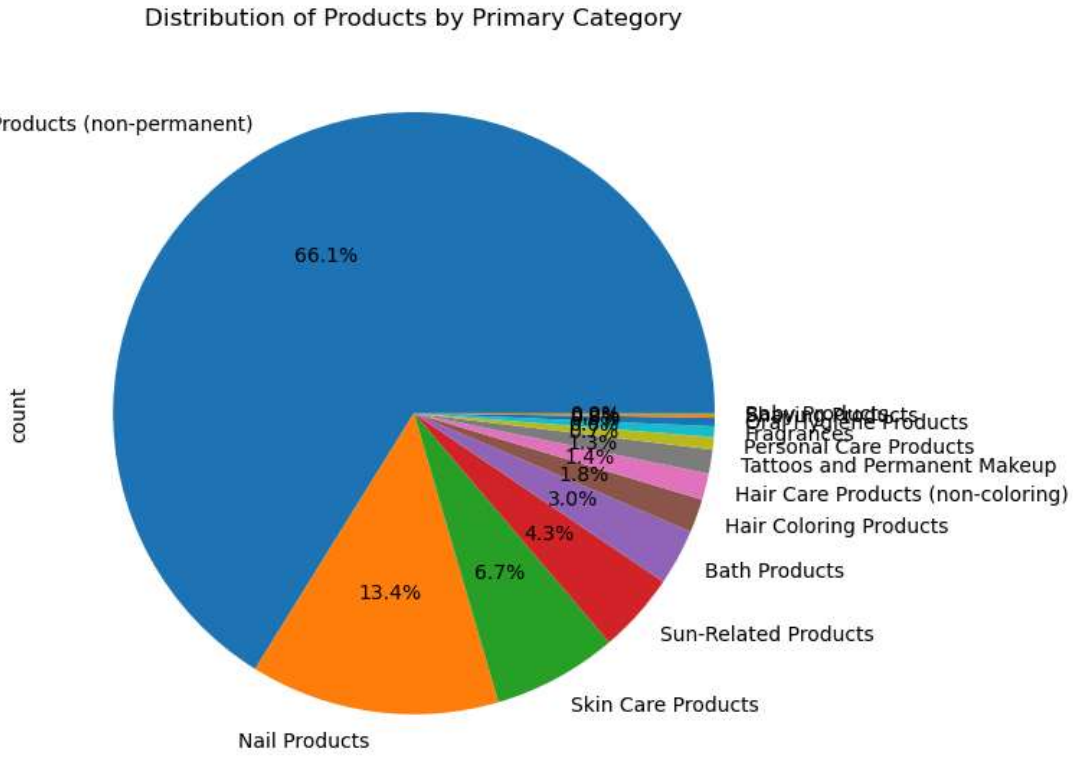
In [81]:
```
sns.pairplot(df)
plt.show()
```



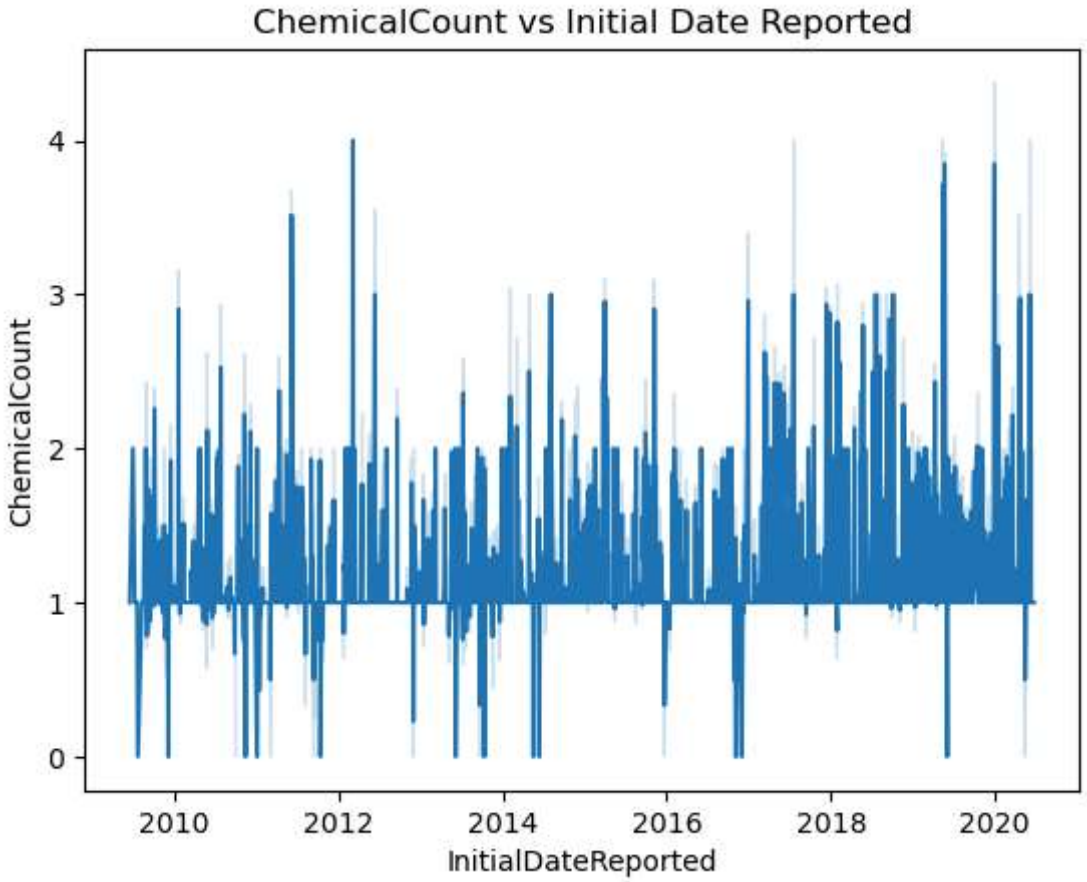
In [33]:
```
sns.countplot(x='PrimaryCategory', data=df)
plt.xticks(rotation=45)
plt.title('Distribution of Products by Primary Category')
plt.show()
```



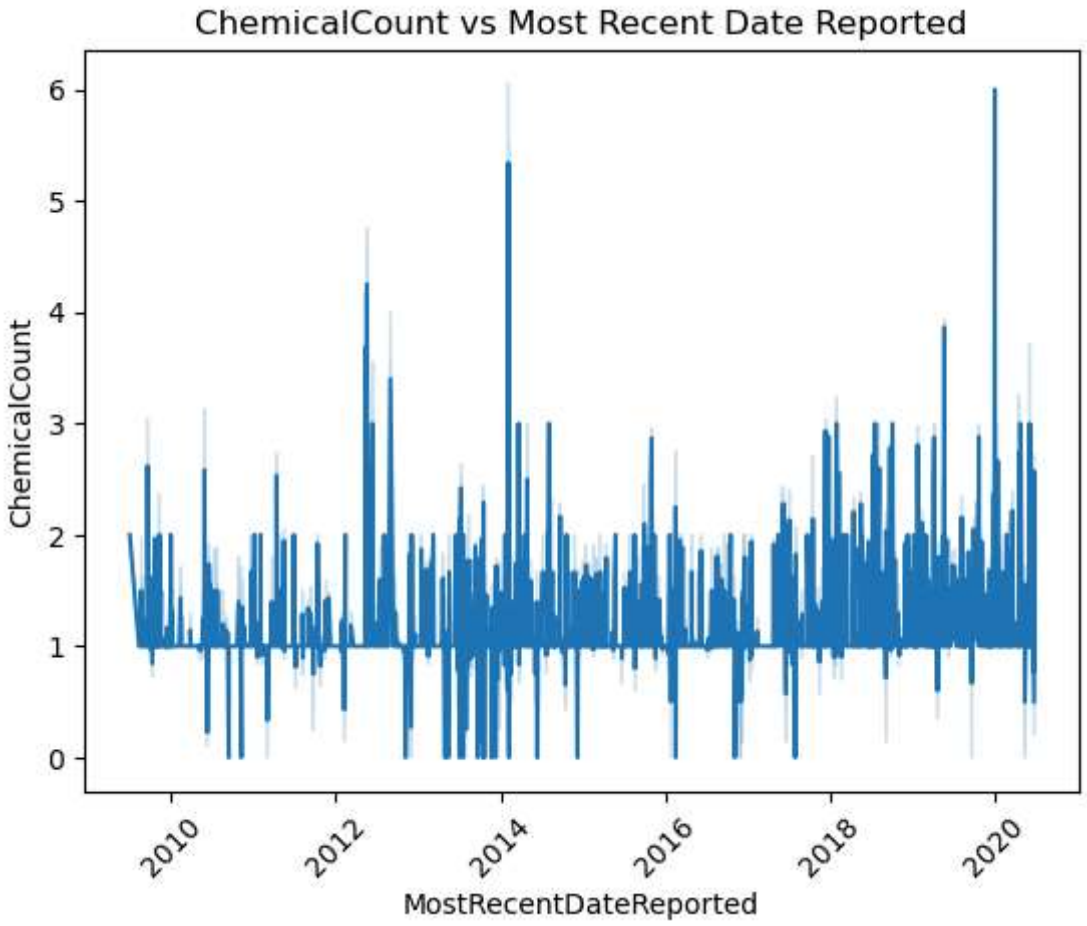
In [104]:
```
plt.figure(figsize=(8,8))
df['PrimaryCategory'].value_counts().plot(kind='pie', autopct='%1.1f%%')
plt.title('Distribution of Products by Primary Category')
plt.tight_layout()
plt.show()
```



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



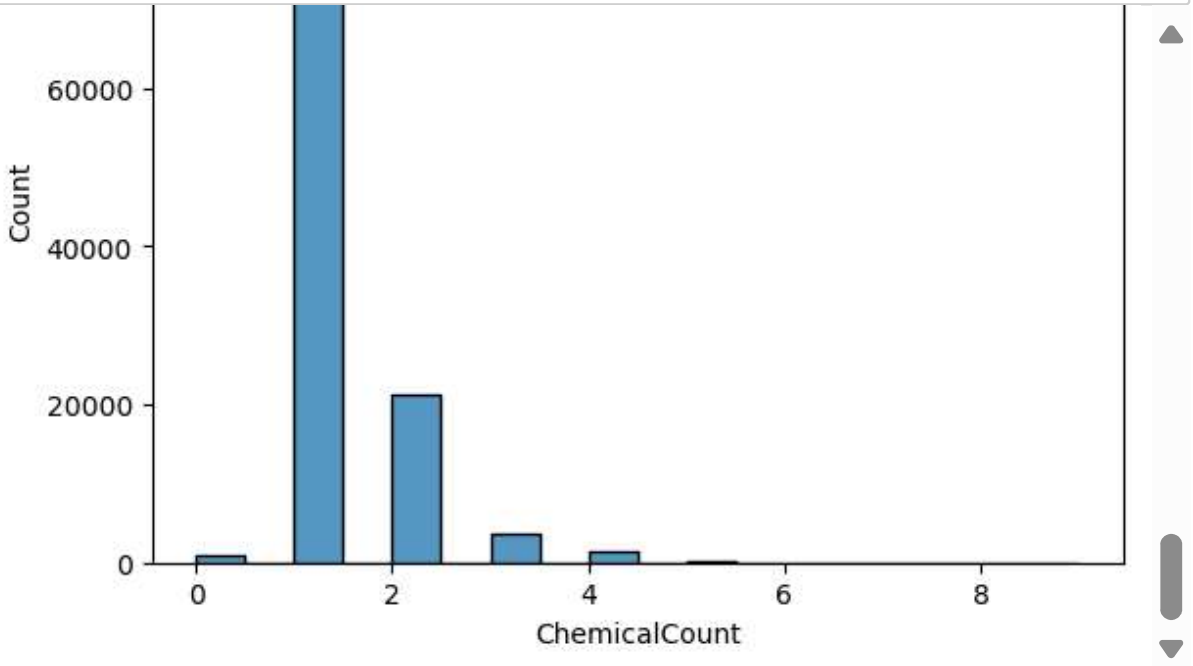
In [86]:
```
sns.lineplot(x='MostRecentDateReported', y='ChemicalCount', data=df)
plt.title('ChemicalCount vs Most Recent Date Reported')
plt.xticks(rotation=45)
plt.show()
```



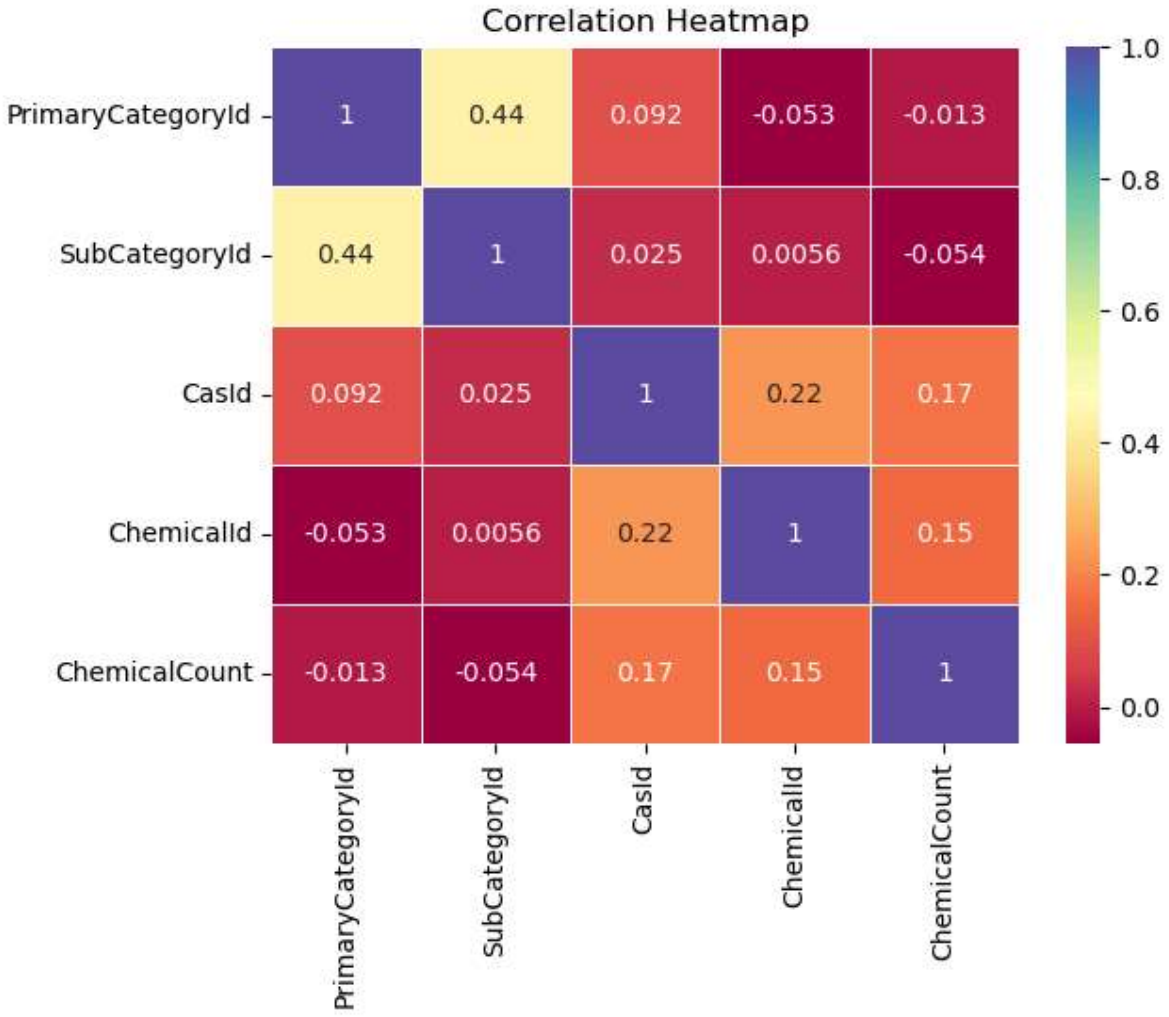
In [98]:
```
df.select_dtypes(include="number").columns
```

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

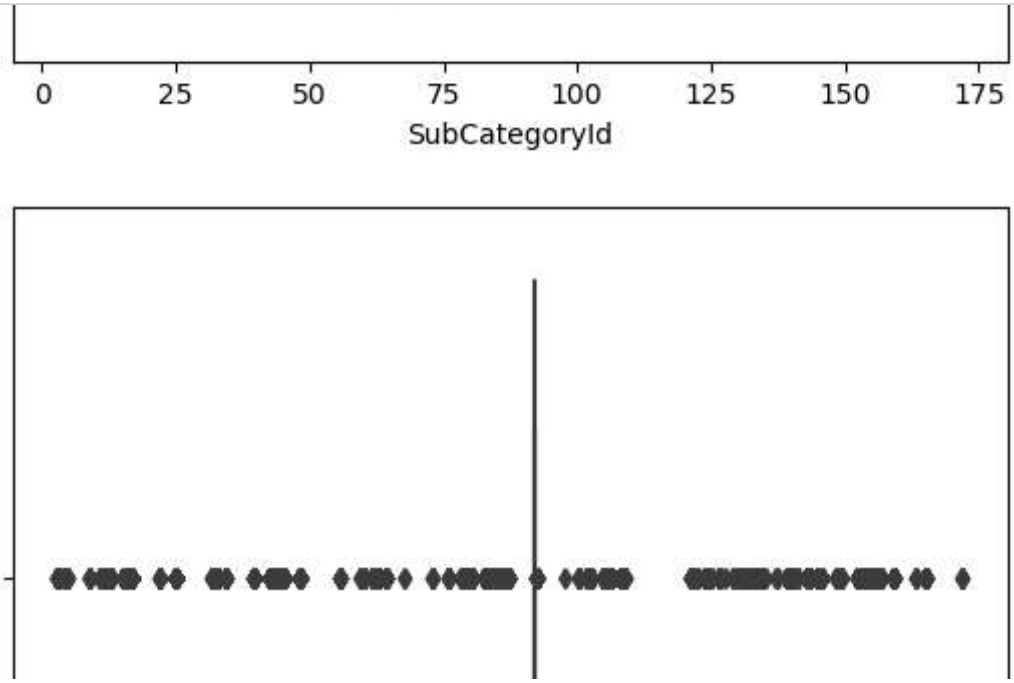
In [97]:
```
for i in df.select_dtypes(include="number").columns:
    sns.histplot(data=df,x=i)
    plt.show()
```



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()
```

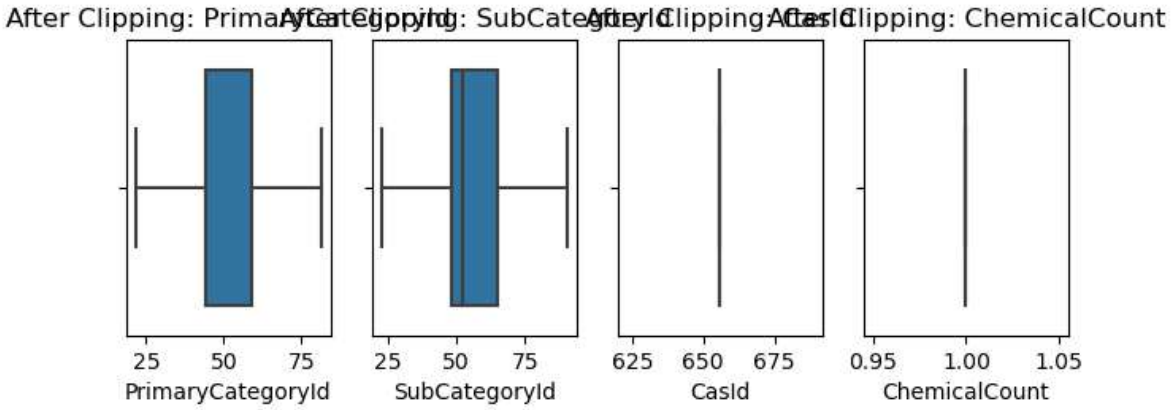


In [127]:
```
num_col=['PrimaryCategoryId','SubCategoryId','CasId','ChemicalCount']
for col in num_col:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    df[col] = df[col].clip(lower=lower_bound, upper=upper_bound)
    clipped_values = df[(df[col] == lower_bound) | (df[col] == upper_bound)]
print(f"Clipped values in {col}: {clipped_values.shape[0]}")
```

```
Clipped values in ChemicalCount: 114635
```

In [128]:
```
for i, col in enumerate(num_col, len(num_col) + 1):
    plt.subplot(2, len(num_col), i)  # Create subplot for each column
    sns.boxplot(data=df, x=col)
    plt.title(f'After Clipping: {col}')

plt.tight_layout()  # Adjust layout to prevent overlap
plt.show()
```



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

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

In [130]:
```
from sklearn.preprocessing import LabelEncoder
```

In [131]:
```
encoder=LabelEncoder()
```

In [143]:
```
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['ChemicalId']=encoder.fit_transform(data['ChemicalId'])
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 [144]:
```
data
```

Out[144]:

	ProductName	CompanyName	BrandName	PrimaryCategoryId	PrimaryCategory	SubCate
0	30981	372	82	44.0	5	
1	13104	252	1023	21.5	3	
2	13104	252	1023	21.5	3	
3	22843	372	82	44.0	5	
4	1328	372	82	44.0	5	
...	...	...	...	...	...	
114630	13606	579	2577	44.0	5	
114631	13606	579	2577	44.0	5	
114632	13606	579	2577	44.0	5	
114633	13606	579	2577	44.0	5	
114634	21965	527	1728	21.5	1	

114635 rows × 16 columns

In [145]:
```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif
```

In [164]:
```
X = data.drop(columns=['ChemicalCount'])
y = data["ChemicalCount"]
selector = SelectKBest(chi2, k=14)
X_new = selector.fit_transform(X, y)
print("Selected features:", selected_features)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[164], line 4
      2 y = data["ChemicalCount"]  # Target column
      3 selector = SelectKBest(chi2, k=14)
----> 4 X_new = selector.fit_transform(X, y)
      5 selected_features = X.columns[selector.get_support()]
      6 print("Selected features:", selected_features)

File ~\anaconda3\Lib\site-packages\sklearn\utils\_set_output.py:140, in _wrap
_method_output.<locals>.wrapped(self, X, *args, **kwargs)
    138 @wraps(f)
    139 def wrapped(self, X, *args, **kwargs):
--> 140     data_to_wrap = f(self, X, *args, **kwargs)
    141     if isinstance(data_to_wrap, tuple):
    142         # only wrap the first output for cross decomposition
    143         return_tuple = (
    144             _wrap_data_with_container(method, data_to_wrap[0], X, sel
f),
    145             *data_to_wrap[1:],
    146         )

File ~\anaconda3\Lib\site-packages\sklearn\base.py:918, in TransformerMixin.f
it_transform(self, X, y, **fit_params)
    915     return self.fit(X, **fit_params).transform(X)
    916 else:
    917     # fit method of arity 2 (supervised transformation)
--> 918     return self.fit(X, y, **fit_params).transform(X)

File ~\anaconda3\Lib\site-packages\sklearn\base.py:1151, in _fit_context.<loc
als>.decorator.<locals>.wrapper(estimator, *args, **kwargs)
   1144     estimator._validate_params()
   1146 with config_context(
   1147     skip_parameter_validation=(
   1148         prefer_skip_nested_validation or global_skip_validation
   1149     )
   1150 ):
-> 1151     return fit_method(estimator, *args, **kwargs)

File ~\anaconda3\Lib\site-packages\sklearn\feature_selection\_univariate_sele
ction.py:498, in _BaseFilter.fit(self, X, y)
    480 @_fit_context(prefer_skip_nested_validation=True)
    481 def fit(self, X, y):
    482     """Run score function on (X, y) and get the appropriate features.
    483
    484     Parameters
   (...)
    496         Returns the instance itself.
    497     """
--> 498     X, y = self._validate_data(
    499         X, y, accept_sparse=["csr", "csc"], multi_output=True
    500     )
    502     self._check_params(X, y)
    503     score_func_ret = self.score_func(X, y)

File ~\anaconda3\Lib\site-packages\sklearn\base.py:621, in BaseEstimator._val
idate_data(self, X, y, reset, validate_separately, cast_to_ndarray, **check_p
```

```
arams)
    619         y = check_array(y, input_name="y", **check_y_params)
    620     else:
--> 621         X, y = check_X_y(X, y, **check_params)
    622     out = X, y
    624 if not no_val_X and check_params.get("ensure_2d", True):

File ~\anaconda3\Lib\site-packages\sklearn\utils\validation.py:1147, in check
_X_y(X, y, accept_sparse, accept_large_sparse, dtype, order, copy, force_all_
finite, ensure_2d, allow_nd, multi_output, ensure_min_samples, ensure_min_fea
tures, y_numeric, estimator)
   1142         estimator_name = _check_estimator_name(estimator)
   1143     raise ValueError(
   1144         f"{estimator_name} requires y to be passed, but the target y
is None"
   1145     )
-> 1147 X = check_array(
   1148     X,
   1149     accept_sparse=accept_sparse,
   1150     accept_large_sparse=accept_large_sparse,
   1151     dtype=dtype,
   1152     order=order,
   1153     copy=copy,
   1154     force_all_finite=force_all_finite,
   1155     ensure_2d=ensure_2d,
   1156     allow_nd=allow_nd,
   1157     ensure_min_samples=ensure_min_samples,
   1158     ensure_min_features=ensure_min_features,
   1159     estimator=estimator,
   1160     input_name="X",
   1161 )
   1163 y = _check_y(y, multi_output=multi_output, y_numeric=y_numeric, estim
ator=estimator)
   1165 check_consistent_length(X, y)

File ~\anaconda3\Lib\site-packages\sklearn\utils\validation.py:797, in check_
array(array, accept_sparse, accept_large_sparse, dtype, order, copy, force_al
l_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_min_features, estim
ator, input_name)
    793 pandas_requires_conversion = any(
    794     _pandas_dtype_needs_early_conversion(i) for i in dtypes_orig
    795 )
    796 if all(isinstance(dtype_iter, np.dtype) for dtype_iter in dtypes_ori
g):
--> 797     dtype_orig = np.result_type(*dtypes_orig)
    798 elif pandas_requires_conversion and any(d == object for d in dtypes_o
rig):
    799     # Force object if any of the dtypes is an object
    800     dtype_orig = object

File <__array_function__ internals>:200, in result_type(*args, **kwargs)

TypeError: The DType <class 'numpy.dtype[datetime64]'> could not be promoted
by <class 'numpy.dtype[float64]'>. This means that no common DType exists for
the given inputs. For example they cannot be stored in a single array unless
the dtype is `object`. The full list of DTypes is: (<class 'numpy.dtype[int3
2]'>, <class 'numpy.dtype[int32]'>, <class 'numpy.dtype[int32]'>, <class 'num
```

```
py.dtype[float64]'>, <class 'numpy.dtype[int32]'>, <class 'numpy.dtype[float6
4]'>, <class 'numpy.dtype[int32]'>, <class 'numpy.dtype[int64]'>, <class 'num
py.dtype[int32]'>, <class 'numpy.dtype[int64]'>, <class 'numpy.dtype[int3
2]'>, <class 'numpy.dtype[datetime64]'>, <class 'numpy.dtype[datetime64]'>, <
class 'numpy.dtype[datetime64]'>, <class 'numpy.dtype[datetime64]'>)
```

In [ ]: