8.6 Appendix 6

Python (Pandas) script – version 3.8.8.

```
## 1. Preliminaries
# Importing pandas, numpy and matplotlib
import pandas as pd
import numpy as np
import string
import math
from collections import Counter
%matplotlib inline
import random
import matplotlib.pyplot as plt

# Total prescribing dataset uploaded to pandas
data = pd.read_csv (r'/Volumes/Storage 1/allergies.csv')   #load the csv file
df_dat = pd.DataFrame(data, columns=['Description', 'Source', 'Warning Status', 'Context', 'Medications involved in warning checking', 'Severity', 'Importance
Level', 'Provider Type', 'Provider Specialty', 'Override Reason', 'Override Comment', 'Drug-Allergy Allergen', 'Drug-Allergy Reason', 'Drug-Allergy Reactions',
'Drug-Allergy Reason External IDs', 'Drug-Allergy Class Pair for Cross-Sensitivity Warning', 'Drug-Allergy Contraindication Group', 'Order Sets', 'Panels',
'Same OrderSet/Panels?', 'Patient Department', 'Patient Hospital/Clinic', 'Interaction Setting', 'Medications ordered', 'UserID', 'PtID', 'Order ID', 'Phases of Care',
'Date', 'Time', 'Alert ID', 'Alert DAT', 'Patient Age', 'Patient Sex', 'Related Actions'])
df_dat.astype(str).apply(lambda x: x.str.lower())
strip_whitespace = [string.strip() for string in df_dat]

# Removal of all punctuation
def remove_punctuation(sentence: str) -> str:
    return sentence.translate(str.maketrans('', '', string.punctuation))
# Apply removal function
[remove_punctuation(sentence) for sentence in df_dat]

# Updating headers to remove spaces and replace with '_'.
df_dat1 = df_dat.rename(columns = {'Warning Status':'Warning_Status'})
df_dat2 = df_dat1.rename(columns = {'Importance Level' : 'Importance_Level'})
df_dat3 = df_dat2.rename(columns = {'Provider Type':'Provider_Type'})
df_dat4 = df_dat3.rename(columns = {'Provider Specialty':'Provider_Specialty'})
df_dat5 = df_dat4.rename(columns = {'Interaction Setting':'Interaction_Setting'})
df_dat6 = df_dat5.rename(columns = {'Patient Hospital/Clinic': 'Patient_Hospital_Clinic'})
df_dat7 = df_dat6.rename(columns = {'Phases of Care': 'Phases_of_Care'})
df_dat8 = df_dat7.rename(columns = {'Order Sets':'Order_Sets'})
df_dat9 = df_dat8.rename(columns = {'Drug-Allergy Class Pair for Cross-Sensitivity Warning': 'Drug_Allergy_Class_Pair_for_Cross_Sensitivity_Warning'})
df_dat_1 = df_dat9.rename(columns = {'Drug-Allergy Reason': 'Drug_Allergy_Reason'})
df_dat_2 = df_dat_1.rename(columns = {'Drug-Allergy Contraindication Group': 'Drug_Allergy_Contraindication_Group'})
df_dat_3 = df_dat_2.rename(columns = {'Override Reason' : 'Override_Reason'})
df_dat_4 = df_dat_3.rename(columns = {'Override Comment' : 'Override_Comment'})
df_dat_5 = df_dat_4.rename(columns = {'Drug-Allergy Allergen' : 'Drug_Allergy_Allergen'})
df_dat_6 = df_dat_5.rename(columns = {'Drug-Allergy Reactions' : 'Drug_Allergy_Reactions'})
df_dat_7 = df_dat_6.rename(columns = {'Drug-Allergy Reason External IDs' : 'Drug_Allergy_Reason_External_IDs'})
df_dat_8 = df_dat_7.rename(columns = {'Same OrderSet/Panels?' : 'Same_OrderSet_Panels'})
df_dat_9 = df_dat_8.rename(columns = {'Patient Department' : 'Patient_Department'})
df_dat_1_ = df_dat_9.rename(columns = {'Medications ordered' : 'Medications_ordered'})
df_dat_2_ = df_dat_1_.rename(columns = {'Order ID' : 'Order_ID'})
df_dat_3_ = df_dat_2_.rename(columns = {'Alert ID' : 'Alert_ID'})
df_dat_4_ = df_dat_3_.rename(columns = {'Alert DAT' : 'Alert_DAT'})
df_dat_5_ = df_dat_4_.rename(columns = {'Patient Age' : 'Patient_Age'})
df_dat_6_ = df_dat_5_.rename(columns = {'Patient Sex' : 'Patient_Sex'})
df_dat_7_ = df_dat_6_.rename(columns = {'Related Actions' : 'Related_Actions'})
df_data = df_dat_7_.rename(columns = {'Medications involved in warning checking' : 'Medications_Involved_in_Warning_Checking'})

# Applying lowercase to all column names
df_data.columns = map(str.lower, df_data.columns)
pd.set_option('display.max_columns', None)
df_data.head()

# Sanity check of row and column numbers.
df_data.shape

# Reviewing datatypes (dtypes) in the dataset.
df_data.dtypes


## 2. Exploring, cleaning and removing outliers in dataset.
# Checking for missing data
df_data_sum = df_data.isnull().sum()
#print(df_data_sum)

# Replace missing data 'NaNs' with new category value '-999'.
df_nan = df_data.replace(np.nan,-999)
df_nan.head()


### 2.1. Formatting of 'Sex' variable
```

```
# Total count of patients using the 'ptid' variable. (n=5628)
df_pres_ptid = df_data.pivot_table(index = ['ptid'], aggfunc='size')
print (df_pres_ptid)

# Rename 'patient age' to 'age' for ease of use.
df_rename = df_nan.rename({"patient_age":"age"}, axis=1)

# Rename 'patient sex' to 'sex' for ease of use.
df_prescrib = df_rename.rename({"patient_sex":"sex"}, axis=1)

# Count of patients by sex
df_prescrib_count = df_prescrib.pivot_table(index=['sex'], aggfunc='size')
#print (df_prescrib_count)

# Remove outliers where sex is unknown
df_pres_mf = df_prescrib[df_prescrib["sex"].str.contains("Male|Female")==True]
df_pres_mf.head()

# Change sex from integer to 'Female' as '0" and 'Male' as '1'.
df_pres_mf['sex'] = np.where(df_pres_mf['sex'] == 'Female', '0', '1').astype(int)

# Count of patients by sex after outlier removal and reclassification of 'Female' as '0" and 'Male' as '1'.
df_pres_sex = df_pres_mf.pivot_table(index = ['sex'], aggfunc='size')
#print (df_pres_sex)

# Sanity check of row and column numbers.
df_pres_mf.shape

# Removable of commas from 'drug-allergy reason external ids' column.
df_pres_mf["drug_allergy_reason_external_ids"] = df_pres_mf["drug_allergy_reason_external_ids"].str.replace('[^\w\s]','')
df_pres_mf.head(10)
```

2.2 Date and time formatting.

```
# Convert the 'date' column to date time format.
df_pres_mf['date'] = pd.to_datetime(df_pres_mf['date'])
df_pres_mf.head(10)

# Apply time delta function.
df_pres_mf['time'] = pd.to_timedelta(df_pres_mf['time']+':00')
df_pres_mf.head()

# Checking dtype formatting.
df_pres_mf.dtypes

# Remove spaces and the words 'years' and 'months' from
df_pres_mf['age'] = df_pres_mf['age'].map(lambda x: x.lstrip(' ' and ' years ').rstrip(' years ' and ' months'))
df_pres_mf.head(10)

# Add placeholder columns to facilitate separation of m years and n months in to numeric format in months.
df_pres_mf[['year1', 'year2', 'month1', 'month2']] = df_pres_mf.age.str.split(" ",expand=True, n=3)
df_pres_mf.head(20)

# Dropping placeholder columns contatining words 'year' and 'month', while filling missing data with NaNs.
df_pres = df_pres_mf.drop('year2', axis=1)
df_pre = df_pres.drop('month2', axis=1)
df_pre['month1'].replace(to_replace=[None], value=np.nan, inplace=True)
df_pre.head(10)
```

2.3 Convert patient age 'year' and 'month' into months in new age column called 'age in months'.

```
# Convert 'year' to int and time in years to time in months.
df_pre['year1'] = df_pre.year1.astype(int)
df_pre['new_age'] = (df_pre['year1'] * 12).astype(int)

# Convert 'month1' to int
df_pre['month1']  = pd.to_numeric(df_pre['month1'] ,errors='coerce')
df_pre = df_pre.replace(np.nan, 0, regex=True)
df_pre['month1']  = df_pre['month1'] .astype(int)

# Add year in months and month column.
df_pre['age_in_months'] = (df_pre['new_age'] + df_pre['month1'])

# Add a new age in year column by dividing age in months by 12.
df_pre['age_in_years'] = (df_pre['age_in_months']/12).round(2).astype(int)
df_pre.head(50)

# Drop unneccessary columns.
df_pr_y = df_pre.drop('year1', axis=1)
df_pr_m = df_pr_y.drop('month1', axis=1)
df_p = df_pr_m.drop('new_age', axis=1)
df_pr = df_p.drop('age_in_months', axis=1)
df_pr.head(10)

df_pr.shape
```

```
# Checking dtype formatting.
df_pr.dtypes
```

3. Description of data

```
# Total count of missing data entries for each variable
df_pre_missing_data_entries = df_pr.apply(pd.value_counts).fillna(0); df_pre_missing_data_entries
df_pre_m = df_pre_missing_data_entries.iloc[:1,:]
df_pre_m.head()
```

3.1 Count of prescriptions by source column.

```
# Counter of the number of each prescription taken from the 'source' column in descending order.
df_pre_c1b = Counter(df_pr['source'])
df_pre_c2b = pd.DataFrame.from_dict(df_pre_c1b, orient='index').reset_index()
df_pre_c2b.columns = ['source', 'count']
df_pre_c3b = df_pre_c2b.sort_values(by = ['count'], ascending = False)

# Total count of override reasons.
df_pre_c3b.count()
# Include percentage column
df_pre_c3b['%'] = ((df_pre_c3b['count'] / df_pre_c3b['count'].sum())*100).round(2).astype(str) + '%'
# Top ten sources by count.
df_pre_c3_t10 = df_pre_c3b.head(10)
df_pre_c3_t10

# Barchart of top 10 count of warnings by 'source'.
ax=df_pre_c3_t10[['source','count']].plot(kind='bar', x='source', y='count', title='Count by source (Top 10).', figsize=(8,6),legend=False, fontsize=10)
ax.set_xlabel('source',fontsize=10)
ax.set_ylabel('count',fontsize=10)
plt.gcf().set_size_inches(13,8)

# Bottom ten sources by count.
df_pre_c3_t10 = df_pre_c3b.tail(10)
df_pre_c3_t10

# Barchart of bottom 10 count of warnings by 'source'.
ax=df_pre_c3_t10[['source','count']].plot(kind='bar', x='source', y='count', title='Count by source (Bottom 10).', figsize=(8,6),legend=False, fontsize=10)
ax.set_xlabel('source',fontsize=10)
ax.set_ylabel('count',fontsize=10)
plt.gcf().set_size_inches(13,8)
```

3.2 Count of prescriptions by description column.

```
# Counter of the number of each prescription taken from the 'description' column in descending order.
df_pre_c1c = Counter(df_pr['description'])
df_pre_c2c = pd.DataFrame.from_dict(df_pre_c1c, orient='index').reset_index()
df_pre_c2c.columns = ['description', 'count']
df_pre_c3c = df_pre_c2c.sort_values(by = ['count'], ascending = False)
# Total count of override reasons.
df_pre_c3c.count()
# Include percentage column
df_pre_c3c['%'] = ((df_pre_c3c['count'] / df_pre_c3c['count'].sum())*100).round(2).astype(str) + '%'
# Top ten prescriptions by count.
df_pre_c3c_t10 = df_pre_c3c.head(10)
df_pre_c3c_t10

# Barchart of top 10 count of warnings by 'description'.
ax=df_pre_c3c_t10[['description','count']].plot(kind='bar', x='description', y='count', title='Count by description (Top 10).', figsize=(8,6),legend=False,
fontsize=10)
ax.set_xlabel('description',fontsize=10)
ax.set_ylabel('count',fontsize=10)
plt.gcf().set_size_inches(13,8)

# Bottom ten prescriptions by count.
df_pre_c3c_t10 = df_pre_c3c.tail(10)
df_pre_c3c_t10

# Barchart of bottom 10 count of warnings by 'description'.
ax=df_pre_c3c_t10[['description','count']].plot(kind='bar', x='description', y='count', title='Count by description (Bottom 10).', figsize=(8,6),legend=False,
fontsize=10)
ax.set_xlabel('description',fontsize=10)
ax.set_ylabel('count',fontsize=10)
plt.gcf().set_size_inches(13,8)
```

3.3 Count of prescriptions by provider type.

```
# Counter of the number of each 'provider type' in descending order.
df_pre_c1d = Counter(df_pr['provider_type'])
df_pre_c2d = pd.DataFrame.from_dict(df_pre_c1d, orient='index').reset_index()
df_pre_c2d.columns = ['provider_type', 'count']
df_pre_c3d = df_pre_c2d.sort_values(by = ['count'], ascending = False)

# Total count of override reasons.
df_pre_c3d.sum()
```

```
# Include percentage column
df_pre_c3d['%'] = ((df_pre_c3d['count'] / df_pre_c3d['count'].sum())*100).round(2).astype(str) + '%'

# Top ten prescribers by count.
df_pres_c3d_t10 = df_pre_c3d.head(10)
df_pres_c3d_t10

# Barchart of top 10 count of warnings by 'provider type'.
#ax=df_pres_c3d_t10[['provider_type','count']].plot(kind='bar', x='provider_type', y='count', title='Count by provider type (Top 10).', figsize=(8,6),legend=False,
fontsize=10)
ax.set_xlabel('provider_type',fontsize=10)
ax.set_ylabel('count',fontsize=10)
plt.gcf().set_size_inches(13,8)

# Bottom ten 'prescriber types' by count.
df_pres_c3d_b10 = df_pre_c3d.tail(10)
df_pres_c3d_b10

# Barchart of bottom 10 count of warnings by 'provider type'.
#ax=df_pres_c3d_b10[['provider_type','count']].plot(kind='bar', x='provider_type', y='count', title='Count by provider type (Top 10).',
figsize=(8,6),legend=False, fontsize=10)
ax.set_xlabel('provider_type',fontsize=10)
ax.set_ylabel('count',fontsize=10)
plt.gcf().set_size_inches(13,8)
```

3.4 Count by sex

```
# Counter of the number of each 'sex'.
df_pre_c1s = Counter(df_pr['sex'])
df_pre_c2s = pd.DataFrame.from_dict(df_pre_c1s, orient='index').reset_index()
df_pre_c2s.columns = ['sex', 'count']
df_pre_c3s = df_pre_c2s.sort_values(by = ['count'], ascending = False)
# Include percentage column
df_pre_c3s['%'] = ((df_pre_c3s['count'] / df_pre_c3s['count'].sum())*100).round(2).astype(str) + '%'
df_pre_c3s
```

3.5 Count by age_in_years and inclusion of age_range

```
# Counter of the number of each 'age_in_years' in descending order.
df_pre_c1age = Counter(df_pr['age_in_years'])
df_pre_c2age = pd.DataFrame.from_dict(df_pre_c1age, orient='index').reset_index()
df_pre_c2age.columns = ['age_in_years', 'count']
df_pre_c3age = df_pre_c2age.sort_values(by = ['count'], ascending = False)
df_pre_c3age.head(10)

# Shape of 'age_in_years' column.
df_pre_c3age.shape
# Include percentage column
df_pre_c3age['%'] = ((df_pre_c3age['count'] / df_pre_c3age['count'].sum())*100).round(2).astype(str) + '%'

df_pre_c3age.shape

# 'age_in_years' by count.
df_pre_c3age.round(2)
# Ascending order by age
df_pre_c3age_asc_head = df_pre_c3age.sort_values(by = ['age_in_years'], ascending = False)

# Barchart of count of warnings by 'age_in_years' in ascending order.
ax=df_pre_c3age_asc_head[['age_in_years','count']].plot(kind='bar', x='age_in_years', y='count', title='Warning status count by age_in_years in ascending
order.', figsize=(8,6),legend=False, fontsize=10)
ax.set_xlabel('age_in_years',fontsize=10)
ax.set_ylabel("count",fontsize=10)
plt.gcf().set_size_inches(13,8)

# Top 10 'age_in_years' by count.
df_pre_c3age_head1 = df_pre_c3age.head(10)
df_pre_c3age_head1.round(2)

# Barchart of count of warnings by 'age_in_years' in ascending order.

ax=df_pre_c3age_head1[['age_in_years','count']].plot(kind='bar', x='age_in_years', y='count', title='Warning status count by age_in_years (Top 10).',
figsize=(8,6),legend=False, fontsize=10)
ax.set_xlabel('age_in_years',fontsize=10)
ax.set_ylabel("count",fontsize=10)
plt.gcf().set_size_inches(13,8)

# Bottom 10 'age_in_years' by count.
df_pre_c3age_tail = df_pre_c3age.tail()
df_pre_c3age_tail.round(2)

# Barchart of count of warnings by 'age_in_years' in ascending order.
ax=df_pre_c3age_tail[['age_in_years','count']].plot(kind='bar', x='age_in_years', y='count', title='Warning status count by age_in_years (Bottom 10).',
figsize=(8,6),legend=False, fontsize=10)
ax.set_xlabel('age_in_years',fontsize=10)
```

```
ax.set_ylabel("count",fontsize=10)
plt.gcf().set_size_inches(13,8)

# Categorisation by age group
bins = [0,5.999999,15.999999,25.999999,35.999999,45.999999,55.999999,65.999999,80.999999,120.999999]
names = ['0-5', '6-15', '16-25', '26-35', '36-45', '46-55', '56-65', '66-80', '80+']
d = dict(enumerate(names, 1))
df_pre_ager = (df_pr)
df_pre_ager['age_range'] = np.vectorize(d.get)(np.digitize(df_pre_ager['age_in_years'], bins))
df_pre_ager['age_range'] = pd.Categorical(df_pre_ager.age_range)
df_pre_ager['age_range'] = df_pre_ager.age_range.astype(str)
#df_pre_ager.to_csv('df_pre_ager_180821.csv')
df_pre_ager.tail(50)

df_pre_ager.age_in_years.unique()

df_pre_ager[df_pre_ager['index'].map(lambda index: '28112' in index)]
df_pre_ager.iloc[[28112]]

# Sanity check for age_range count
age_rnge = df_pre_ager['age_range'].count()
age_rnge

# Age_range by count.
df_pre_agera = Counter(df_pre_ager['age_range'])
df_pre_ageran = pd.DataFrame.from_dict(df_pre_agera, orient='index').reset_index()
df_pre_ageran.columns = ['age_range', 'count']
df_pre_agerang = df_pre_ageran.sort_values(by = ['count'], ascending = False)
df_pre_agerang

# Include percentage column
df_pre_agerang['%'] = ((df_pre_agerang['count'] / df_pre_agerang['count'].sum())*100).round(2).astype(str) + '%'
#df_pre_agerang.to_csv('df_pre_agerang_180821.csv')
df_pre_agerang.astype(str)

# Ascending order by age_range
df_pre_agerang_asc = df_pre_agerang.sort_values(by = ['age_range'], ascending = True)

# Barchart of count of warnings by 'age_in_years' in ascending order.
ax=df_pre_agerang_asc[['age_range', 'count']].plot(kind='bar', x='age_range', y='count', title='Warning status count by age_range.', figsize=(8,6),legend=False,
fontsize=10)
ax.set_xlabel('age_range',fontsize=10)
ax.set_ylabel('count',fontsize=10)
plt.gcf().set_size_inches(13,8)


df_pre_agerang_asc.info()

# Sanity check for age_range count and type
df_pre_agerang_asc['age_range'].value_counts().astype(str)
```

3.6 Count of prescriptions by provider specialty.

```
# Counter of the number of each 'provider_specialty' in descending order.
df_pre_c1e = Counter(df_pr['provider_specialty'])
df_pre_c2e = pd.DataFrame.from_dict(df_pre_c1e, orient='index').reset_index()
df_pre_c2e.columns = ['provider_specialty', 'count']
df_pre_c3e = df_pre_c2e.sort_values(by = ['count'], ascending = False)

# Shape of provider specialty column.
df_pre_c3e.shape

# Include percentage column
df_pre_c3e['%'] = ((df_pre_c3e['count'] / df_pre_c3e['count'].sum())*100).round(2).astype(str) + '%'

# Top 10 'provider specialty' by count.
df_pre_c3e_head = df_pre_c3e.head(10)
df_pre_c3e_head

# Barchart of top 10 count of warnings by 'provider specialty'.
ax=df_pre_c3e_head[['provider_specialty','count']].plot(kind='bar', x='provider_specialty', y='count', title='Warning status count by provider specialty (Top 10).',
figsize=(8,6),legend=False, fontsize=10)
ax.set_xlabel('provider_specialty',fontsize=10)
ax.set_ylabel("count",fontsize=10)
plt.gcf().set_size_inches(13,8)

# Bottom 10 'provider specialty' by count.
df_pre_c3e_tail = df_pre_c3e.tail(10)
df_pre_c3e_tail

# Barchart of bottom 10 count of warnings by 'provider specialty'.
ax=df_pre_c3e_tail[['provider_specialty','count']].plot(kind='bar', x='provider_specialty', y='count', title='Warning status count by provider specialty (Bottom
10).', figsize=(8,6),legend=False, fontsize=10)
ax.set_xlabel('provider_specialty',fontsize=10)
ax.set_ylabel("count",fontsize=10)
```

```
plt.gcf().set_size_inches(13,8)


### 3.7. Override reason by count
# Counter of the number of each 'override_reason' in descending order.
df_pre_c1f = Counter(df_pr['override_reason'])
df_pre_c2f = pd.DataFrame.from_dict(df_pre_c1f, orient='index').reset_index()
df_pre_c2f.columns = ['override_reason', 'count']
df_pre_c3f = df_pre_c2f.sort_values(by = ['count'], ascending = False)

# Total count of override reasons.
df_pre_c3f.sum()

# Include percentage column
df_pre_c3f['%'] = ((df_pre_c3f['count'] / df_pre_c3f['count'].sum())*100).round(2).astype(str) + '%'

# Top 10 'override_reason' by count.
df_pre_c3f_head = df_pre_c3f.head(10)
df_pre_c3f_head

# Barchart of top 10 count of warnings by 'override_reason'.
ax=df_pre_c3f_head[['override_reason','count']].plot(kind='bar', x='override_reason', y='count', title='Warning status count by override reason (Top 10).',
figsize=(8,6),legend=False, fontsize=10)
ax.set_xlabel('override_reason',fontsize=10)
ax.set_ylabel('count',fontsize=10)
plt.gcf().set_size_inches(13,8)

# Bottom 10 'override_reason' by count.
df_pre_c3f_tail = df_pre_c3f.tail(10)
df_pre_c3f_tail

# Counter of the number of each 'override_reason' in descending order.
# Override reason by count excluding missing data.
df_pre_c3f_nan = df_pre_c3f.replace(-999, np.nan)
df_pre_c3f_nm = df_pre_c3f_nan.dropna()
pd.set_option('display.max_colwidth', None)
# Include percentage column
df_pre_c3f_nm['%'] = ((df_pre_c3f_nm['count'] / df_pre_c3f_nm['count'].sum())*100).round(2).astype(str) + '%'
#df_pre_c3f_nm.to_csv('over_rsn_no_per.csv')
df_pre_c3f_nm

# Barchart of bottom 10 count of warnings by 'override_reason'.
ax=df_pre_c3f_tail[['override_reason','count']].plot(kind='bar', x='override_reason', y='count', title='Warning status count by override reason (Bottom 10).',
figsize=(8,6),legend=False, fontsize=10)
ax.set_xlabel('override_reason',fontsize=10)
ax.set_ylabel("count",fontsize=10)
plt.gcf().set_size_inches(13,8)


## 4. warning levels by size, count and unique value.
# Total size, count and unique counts of warnings in relation to each variable in the dataset.
  # where 'size' is the count including NaN and repeated values, 'count'is the count excluding NaN but including repeats
    # and 'nunique'is the count of unique values, excluding repeats and NaN.
df_pre_overall = df_pr.groupby('warning_status').agg(['size', 'count', 'nunique'])
df_pre_overall


### 4.1. Warnings by count and by source (point of alert trigger)
# Total number of warnings by 'warning status' count
  # where 'size' is the count including NaN and repeated values, 'count'is the count excluding NaN but including repeats
    # and 'nunique'is the count of unique values, excluding repeats and NaN.
df_pre_wa = df_pr.groupby('warning_status')['warning_status'].agg(['size', 'count', 'nunique'])

# Number of warning alerts by count and resetting index.
df_pre_wa = df_pre_wa.reset_index()
df_pre_wa1 = df_pre_wa.rename(columns = {'warning_status':'warning_status'})
df_pre_wa1 = df_pre_wa1[:]
# Include percentage column
df_pre_wa1['%'] = ((df_pre_wa1['count'] / df_pre_wa1['count'].sum())*100).round(2).astype(str) + '%'
#df_pre_wa1.to_csv('df_pre_wa1_150821.csv')
df_pre_wa1

# Total number of warnings by 'warning status' and 'source'.
  # where 'size' is the count including NaN and repeated values, 'count'is the count excluding NaN but including repeats
    # and 'nunique'is the count of unique values, excluding repeats and NaN.
df_pre_so = df_pr.groupby('warning_status')['source'].agg(['size', 'count', 'nunique'])

# Number of warning alerts by source and resetting index.
df_pre_so = df_pre_so.reset_index()
df_pre_so1 = df_pre_so.rename(columns = {'warning_status':'warning_status'})
df_pre_so1 = df_pre_so1[:]
df_pre_so1

# Conformation survey barchart of total count of warnings by 'warning status' and 'source'.
```

```
ax=df_pre_so1[['warning_status','count']].plot(kind='bar', x='warning_status', y='count', title='Warning status by source.', figsize=(8,6),legend=False, fontsize=10)
ax.set_xlabel('warning_status',fontsize=10)
ax.set_ylabel("count",fontsize=10)
plt.gcf().set_size_inches(13,8)
# Conformation survey plot of total count of warnings by 'warning status' and 'source'.
df_pre_so1.plot()
plt.title('Warning status by source.')
plt.xlabel('source')
plt.ylabel('count')
plt.gcf().set_size_inches(13,8)

df_pre_so1.shape

# Count of 'warning status' by 'source',
df_pre_source = df_pr.pivot_table(index = ['warning_status', 'source'], aggfunc = 'count')
df_pre_sour = df_pre_source[['age']]
df_pre_sour0 = df_pre_sour.rename(columns={'age': 'count'}, inplace = True)
df_pre_sour0 = df_pre_sour.sort_values(by = ['count'], ascending = False)

# Warning status versus source.
df_pre_sour1 = df_pre_sour0.reset_index()
df_pre_sour2 = df_pre_sour1.rename(columns = {'warning_status':'warning_status'})

# Top 10 list.
df_pre_sour_t = df_pre_sour1.replace(-999, np.nan)
df_pre_sour_top = df_pre_sour_t.dropna()
pd.set_option('display.max_colwidth', None)
df_pre_sour_to_10 = df_pre_sour_top.head(10)
df_pre_sour_top10 = pd.DataFrame(df_pre_sour_to_10)
df_pre_sour_top10 = df_pre_sour_top10.reset_index(drop=True)

# Include percentage column
df_pre_sour_top10['%'] = ((df_pre_sour_top10['count'] / df_pre_sour_top10['count'].sum())*100).round(2).astype(str) + '%'
df_pre_sour_top10


# Visualisation of top 10 warning distribution versus source data points.
plt.figure(figsize=(9,7))
ax = plt.gca()
x = df_pre_sour_top10['warning_status']
y = df_pre_sour_top10['count']
n = ['Enter Orders', 'Enter Orders', 'Verify Orders', 'One-step Medications', 'Treatment Plan', 'Enter Orders', 'Retro-Allergen', 'Retro-Allergen', 'Resolve Fill Review Prescription Order Flag', 'MAR']
t = np.arange(10)
plt.title('Warning distribution versus source (Top 10)')
plt.ylabel('Warning count by source')
plt.xlabel('Warning alerts')
plt.scatter(x, y, c=t, s=100, alpha=1.0, marker='^')
plt.gcf().set_size_inches(13,8)
#scatter labels
for i, txt in enumerate(n):
    ax.annotate(txt, (x[i],y[i]))
plt.show()

# Source by count.
df_pre_source = df_pr.pivot_table(index = ['source'], aggfunc = 'count')
df_pre_sour = df_pre_source[['age']]
df_pre_sour4 = df_pre_sour.rename(columns={'age': 'count'}, inplace = True)
df_pre_sour4 = df_pre_sour.sort_values(by = ['count'], ascending = False)
df_pre_sour5 = df_pre_sour4.reset_index()
df_pre_sour7 = df_pre_sour5.rename(columns = {'source':'source'})
# Include percentage column
df_pre_sour7['%'] = ((df_pre_sour7['count'] / df_pre_sour7['count'].sum())*100).round(2).astype(str) + '%'
df_pre_sour7

df_pre_sour7.shape

# Barchart of total count of warnings by 'warning status' and 'source'.
ax=df_pre_sour7[['source','count']].plot(kind='bar', x='source', y='count', title='Count by source.', figsize=(8,6),legend=False, fontsize=10)
ax.set_xlabel('source',fontsize=10)
ax.set_ylabel("count",fontsize=10)
plt.gcf().set_size_inches(13,8)
```

4.2. Warnings by description.

```
# Total number of warnings by 'warning status' and 'description'.
  # where 'size' is the count including NaN and repeated values, 'count'is the count excluding NaN but including repeats
    # and 'nunique'is the count of unique values, excluding repeats and NaN.
df_pre_des = df_pr.groupby('warning_status')['description'].agg(['size', 'count', 'nunique'])

# Number of warning alerts by description and resetting index.
df_pre_des = df_pre_des.reset_index()
df_pre_des1 = df_pre_des.rename(columns = {'warning_status':'warning_status'})
df_pre_des1 = df_pre_des1[:]
```

```
df_pre_des1

# Conformation survey barchart of total count of warnings by 'warning status' and 'description'.
ax=df_pre_des1[['warning_status','count']].plot(kind='bar', x='warning_status', y='count', title='Warning status by description.', figsize=(8,6),legend=False,
fontsize=10)
ax.set_xlabel('warning_status',fontsize=10)
ax.set_ylabel("count",fontsize=10)
plt.gcf().set_size_inches(13,8)

# Conformation survey plot of total size, count and unique number of warnings by 'warning status' and 'description'.
df_pre_des1.plot()
plt.title('Warning status by description.')
plt.xlabel('description')
plt.ylabel('count')
plt.gcf().set_size_inches(13,8)

# Count of 'warning status' by 'description',
df_pre_desc = df_pr.pivot_table(index = ['warning_status', 'description'], aggfunc = 'count')
df_pre_descrp = df_pre_desc[['age']]
df_pre_descrp0 = df_pre_descrp.rename(columns={'age': 'count'}, inplace=True)
df_pre_descrp0 = df_pre_descrp.sort_values(by = ['count'], ascending = False)
# Warning status versus description.
df_pre_descrp1 = df_pre_descrp0.reset_index()
df_pre_descrp2 = df_pre_descrp1.rename(columns = {'warning status':'warning_status'})
# Include percentage column
df_pre_descrp1['%'] = ((df_pre_descrp1['count'] / df_pre_descrp1['count'].sum())*100).round(2).astype(str) + '%'
df_pre_descrp1.head()

df_pre_descrp1.count()

# Top 10 list.
df_pre_descrp2_t = df_pre_descrp1.replace(-999, np.nan)
df_pre_descrp2_top = df_pre_descrp2_t.dropna()
pd.set_option('display.max_colwidth', None)
df_pre_descrp2_to_10 = df_pre_descrp2_top.head(10)
df_pre_descrp2_top10 = pd.DataFrame(df_pre_descrp2_to_10)
df_pre_descrp2_top10 = df_pre_descrp2_top10.reset_index(drop=True)
df_pre_descrp2_top10

# Visualisation of top 10 warning distribution versus description of prescription data points.
plt.figure(figsize=(9,7))
ax = plt.gca()
x = df_pre_descrp2_top10['warning_status']
y = df_pre_descrp2_top10['count']
n = ['MORPHINE', 'CODEINE', 'TRAMADOL', 'PENICILLINS', 'NORMAL IMMUNOGLOBULIN HUMAN', 'PENICILLINS', 'CODEINE', 'CO-
CODAMOL (CODEINE AND PARACETAMOL)', 'MORPHINE', 'CODEINE']
t = np.arange(10)
plt.title('Warning distribution versus description (Top 10)')
plt.ylabel('Warning count by description')
plt.xlabel('Warning alerts')
plt.scatter(x, y, c=t, s=100, alpha=1.0, marker='^')
plt.gcf().set_size_inches(13,8)
#scatter labels
for i, txt in enumerate(n):
    ax.annotate(txt, (x[i],y[i]))
plt.show()

# Drug type 'description' by overridden.
df_over = df_pre_descrp2_top[df_pre_descrp2_top['warning_status'].map(lambda warning_status: 'Overridden' in warning_status)]
df_over
#df_over.to_csv('df_over_080821.csv')

# Description by count
df_pre_des1 = df_pr.pivot_table(index = ['description'], aggfunc = 'count')
df_pre_des2 = df_pre_des1[['age']]
df_pre_des3 = df_pre_des2.sort_values(by = ['age'], ascending = False)
df_pre_des4 = df_pre_des3.reset_index()
df_pre_des5 = df_pre_des4.rename(columns={'age': 'count'})
# Include percentage column
df_pre_des5['%'] = ((df_pre_des5['count'] / df_pre_des5['count'].sum())*100).round(2).astype(str) + '%'
df_pre_des5.head(10)

# Top 50 list
df_pre_des5_to_50 = df_pre_des5.head(50)
df_pre_des5_top50 = pd.DataFrame(df_pre_des5_to_50)
df_pre_des5_top50 = df_pre_des5_top50.reset_index(drop=True)
df_pre_des5_top50

# Barchart of total count of warnings by 'warning status' and 'description'.
ax=df_pre_des5_top50[['description','count']].plot(kind='bar', x='description', y='count', title='Count by description (Top 50).', figsize=(8,6),legend=False,
fontsize=10)
ax.set_xlabel('description',fontsize=10)
ax.set_ylabel("count",fontsize=10)
plt.gcf().set_size_inches(13,8)
```

4.3. Warnings by provider type.

```
# Total number of warnings by 'warning status' and 'provider type'.
  # where 'size' is the count including NaN and repeated values, 'count'is the count excluding NaN but including repeats
    # and 'nunique'is the count of unique values, excluding repeats and NaN.
df_pre_proty = df_pr.groupby('warning_status')['provider_type'].agg(['size', 'count', 'nunique'])

# Number of warning alerts by provider type and resetting index.
df_pre_proty = df_pre_proty.reset_index()
df_pre_proty1 = df_pre_proty.rename(columns = {'warning_status':'warning_status'})
df_pre_proty1 = df_pre_proty1[:]
df_pre_proty1
# Conformation survey barchart of total count of warnings by 'warning status' and 'provider type'.
#ax=df_pre_proty1[['warning_status','count']].plot(kind='bar', x='warning_status', y='count', title='Warning status by provider type.', figsize=(8,6),legend=False,
fontsize=10)
ax.set_xlabel('warning_status',fontsize=10)
ax.set_ylabel("count",fontsize=10)
plt.gcf().set_size_inches(13,8)

# Conformation survey plot of total count of warnings by 'warning status' and 'provider type'.
df_pre_proty1.plot()
plt.title('Warning status by provider type.')
plt.xlabel('provider type')
plt.ylabel('count')
plt.gcf().set_size_inches(13,8)

# Count of 'warning status' by 'provider type',
df_pre_pro = df_pr.pivot_table(index = ['warning_status', 'provider_type'], aggfunc = 'count')
df_pre_prot = df_pre_pro[['age']]
df_pre_prot0 = df_pre_prot.rename(columns={'age': 'count'}, inplace=True)
df_pre_prot0 = df_pre_prot.sort_values(by = ['count'], ascending = False)

# Warning status versus provider type.
df_pre_prot1 = df_pre_prot0.reset_index()
df_pre_prot2 = df_pre_prot1.rename(columns = {'warning_status':'warning_status'})
# Include percentage column
df_pre_prot2['%'] = ((df_pre_prot2['count'] / df_pre_prot2['count'].sum())*100).round(2).astype(str) + '%'
df_pre_prot2.head()

df_pre_prot2.count()

# Top 10 list.
df_pre_prot2_t = df_pre_prot2.replace(-999, np.nan)
df_pre_prot2_top = df_pre_prot2_t.dropna()
pd.set_option('display.max_colwidth', None)
df_pre_prot2_to_10 = df_pre_prot2_top.head(10)
df_pre_prot2_top10 = pd.DataFrame(df_pre_prot2_to_10)
df_pre_prot2_top10 = df_pre_prot2_top10.reset_index(drop=True)
df_pre_prot2_top10

# Visualisation of top 10 warnings distribution versus provider type data points.
plt.figure(figsize=(9,7))
ax = plt.gca()
x = df_pre_prot2_top10['warning_status']
y = df_pre_prot2_top10['count']
n = ['Pharmacist', 'Registrar - ST1-2', 'Registrar - ST3+', 'Consultant Anaesthetist', 'Consultant Anaesthetist', 'Technician', 'Staff Nurse', 'Consultant Anaesthetist',
'Registrar - ST1-2', 'Consultant Doctor']
t = np.arange(10)
plt.title('Warning distribution versus provider type (Top 10)')
plt.ylabel('Warning count by provider type')
plt.xlabel('Warning alerts')
plt.scatter(x, y, c=t, s=100, alpha=1.0, marker='^')
plt.gcf().set_size_inches(13,8)
#scatter labels
for i, txt in enumerate(n):
    ax.annotate(txt, (x[i],y[i]))
plt.show()

# Description by count
df_pre_prot1 = df_pr.pivot_table(index = ['provider_type'], aggfunc = 'count')
df_pre_prot2 = df_pre_prot1[['age']]
df_pre_prot3 = df_pre_prot2.sort_values(by = ['age'], ascending = False)
df_pre_prot4 = df_pre_prot3.reset_index()
df_pre_prot5 = df_pre_prot4.rename(columns={'age' : 'count'})
# Drop unknowns and max column width.
df_pre_prot6 = df_pre_prot5.replace(-999, np.nan)
df_pre_prot7 = df_pre_prot6.dropna()
pd.set_option('display.max_colwidth', None)
# Include percentage column
df_pre_prot7['%'] = ((df_pre_prot7['count'] / df_pre_prot7['count'].sum())*100).round(2).astype(str) + '%'
df_pre_prot7.head(10)

# Top 50 list
df_pre_prot5_to_50 = df_pre_prot7.head(50)
df_pre_prot5_top50 = pd.DataFrame(df_pre_prot5_to_50)
```

```
df_pre_prot5_top50 = df_pre_prot5_top50.reset_index(drop=True)
df_pre_prot5_top50

df_pre_prot5_top50.dtypes

# Barchart of total count of warnings by 'warning status' and 'provider type'.
ax=df_pre_prot5_top50[['provider_type','count']].plot(kind='bar', x='provider_type', y='count', title='Count by provider type (Top 50).',
figsize=(8,6),legend=False, fontsize=10)
ax.set_xlabel('provider_type',fontsize=10)
ax.set_ylabel("count",fontsize=10)
plt.gcf().set_size_inches(13,8)
```

4.4. Warnings by provider speciality.

```
# Total number of warnings by 'warning status' and 'provider speciality'.
  # where 'size' is the count including NaN and repeated values, 'count'is the count excluding NaN but including repeats
    # and 'nunique'is the count of unique values, excluding repeats and NaN.
df_pre_prospec = df_pr.groupby('warning_status')['provider_specialty'].agg(['size', 'count', 'nunique'])

# Number of warning alerts by provider specialty and resetting index.
df_pre_prospec = df_pre_prospec.reset_index()
df_pre_prospec1 = df_pre_prospec.rename(columns = {'warning_status':'warning_status'})
df_pre_prospec1 = df_pre_prospec1[:]
df_pre_prospec1

# Conformation survey barchart of total count of warnings by 'warning status' and 'provider specialty'.
ax=df_pre_prospec1[['warning_status','count']].plot(kind='bar', x='warning_status', y='count', title='Warning status by provider specialty.',
figsize=(8,6),legend=False, fontsize=10)
ax.set_xlabel('warning_status',fontsize=10)
ax.set_ylabel("count",fontsize=10)
plt.gcf().set_size_inches(13,8)

# Conformation survey plot of total count of warnings by 'warning status' and 'provider specialty'.
df_pre_prospec1.plot()
plt.title('Warning status by provider specialty.')
plt.xlabel('provider specialty')
plt.ylabel('count')
plt.gcf().set_size_inches(13,8)

# Count of 'warning status' by 'provider speciality'.
df_pre_prspe = df_pr.pivot_table(index = ['warning_status', 'provider_specialty'], aggfunc = 'count')
df_pre_prspec = df_pre_prspe[['age']]
df_pre_prspec0 = df_pre_prspec.rename(columns={'age': 'count'}, inplace=True)
df_pre_prspec0 = df_pre_prspec.sort_values(by = ['count'], ascending = False)

# Warning status versus provider speciality.
df_pre_prspec1 = df_pre_prspec0.reset_index()
df_pre_prspec2 = df_pre_prspec1.rename(columns = {'warning_status':'warning_status'})
# Include percentage column
df_pre_prspec2['%'] = ((df_pre_prspec2['count'] / df_pre_prspec2['count'].sum())*100).round(2).astype(str) + '%'
df_pre_prspec2.head()

df_pre_prspec2.count()

# Top 10 list.
df_pre_prspec2_t = df_pre_prspec2.replace(-999, np.nan)
df_pre_prspec2_top = df_pre_prspec2_t.dropna()
pd.set_option('display.max_colwidth', None)
df_pre_prspec2_to_10 = df_pre_prspec2_top.head(10)
df_pre_prspec2_top10 = pd.DataFrame(df_pre_prspec2_to_10)
df_pre_prspec2_top10 = df_pre_prspec2_top10.reset_index(drop=True)
df_pre_prspec2_top10

# Visualisation of top 10 warnings distribution versus provider speciality data points.
plt.figure(figsize=(9,7))
ax = plt.gca()
x = df_pre_prspec2_top10['warning_status']
y = df_pre_prspec2_top10['count']
n = ['Pharmacy', 'Anes', 'Anes', 'Haem', 'Anes', 'unknown', 'IM', 'Pharmacy', 'EM', 'Neuro']
t = np.arange(10)
plt.title('Warning distribution versus provider specialty (Top 10)')
plt.ylabel('Warning count by provider specialty')
plt.xlabel('Warning alerts')
plt.scatter(x, y, c=t, s=100, alpha=1.0, marker='^')
plt.gcf().set_size_inches(13,8)
#scatter labels
for i, txt in enumerate(n):
    ax.annotate(txt, (x[i],y[i]))
plt.show()

# Provider specialty by count
df_pre_prspec1 = df_pr.pivot_table(index = ['provider_specialty'], aggfunc = 'count')
df_pre_prspec2 = df_pre_prspec1[['age']]
df_pre_prspec3 = df_pre_prspec2.sort_values(by = ['age'], ascending = False)
df_pre_prspec4 = df_pre_prspec3.reset_index()
```

```
df_pre_prspec5 = df_pre_prspec4.rename(columns={'age': 'count'})
# Drop unknowns and max column width.
df_pre_prspec6 = df_pre_prspec5.replace(-999, np.nan)
df_pre_prspec7 = df_pre_prspec6.dropna()
pd.set_option('display.max_colwidth', None)
# Include percentage column
df_pre_prspec7['%'] = ((df_pre_prspec7['count'] / df_pre_prspec7['count'].sum())*100).round(2).astype(str) + '%'
df_pre_prspec7.head(10)

# Top 50 list
df_pre_prspec2_to_50 = df_pre_prspec7.head(50)
df_pre_prspec2_top50 = pd.DataFrame(df_pre_prspec2_to_50)
df_pre_prspec2_top50 = df_pre_prspec2_top50.reset_index(drop=True)
df_pre_prspec2_top50

df_pre_prspec2_top50.dtypes

# Barchart of total count of warnings by 'warning status' and 'provider specialty'.
ax=df_pre_prspec2_top50[['provider_specialty','count']].plot(kind='bar', x='provider_specialty', y='count', title='Count by provider specialty (Top 50).',
figsize=(8,6),legend=False, fontsize=10)
ax.set_xlabel('provider_specialty',fontsize=10)
ax.set_ylabel('count',fontsize=10)
plt.gcf().set_size_inches(13,8)


### 4.5. Warning by phases of care
# Total number of warnings by 'warning status' and 'phases of care'.
  # where 'size' is the count including NaN and repeated values, 'count'is the count excluding NaN but including repeats
    # and 'nunique'is the count of unique values, excluding repeats and NaN.
df_pre_pha = df_pr.groupby('warning_status')['phases_of_care'].agg(['size', 'count', 'nunique'])

# Number of warning alerts by phases of care and resetting index.
df_pre_pha = df_pre_pha.reset_index()
df_pre_pha1 = df_pre_pha.rename(columns = {'warning_status':'warning_status'})
df_pre_pha1 = df_pre_pha1[:]
df_pre_pha1

# Conformation survey barchart of total count of warnings by 'warning status' and 'provider specialty'.
ax=df_pre_pha1[['warning_status','count']].plot(kind='bar', x='warning_status', y='count', title='Warning status by phases of care.', figsize=(8,6),legend=False,
fontsize=10)
ax.set_xlabel('warning_status',fontsize=10)
ax.set_ylabel("count",fontsize=10)
plt.gcf().set_size_inches(13,8)
# Conformation survey plot of total count of warnings by 'warning status' and 'phases of care'.
df_pre_pha.plot()
plt.title('Warning status by phases of care.')
plt.xlabel('phases of care')
plt.ylabel('count')
plt.gcf().set_size_inches(13,8)

# Count of 'warning status' by 'phases of care',
df_pre_phas = df_pr.pivot_table(index=['warning_status', 'phases_of_care'], aggfunc='count')
df_pre_phase = df_pre_phas[['age']]
df_pre_phase0 = df_pre_phase.rename(columns={'age': 'count'}, inplace=True)
df_pre_phase0 = df_pre_phase.sort_values(by = ['count'], ascending = False)

# Warning status versus phases of care.
df_pre_phase1 = df_pre_phase0.reset_index()
df_pre_phase2 = df_pre_phase1.rename(columns = {'warning_status':'warning_status'})

# Top 10 list.
df_pre_phase2_t = df_pre_phase2.replace(-999, np.nan)
df_pre_phase2_top = df_pre_phase2_t.dropna()
pd.set_option('display.max_colwidth', None)
df_pre_phase2_to_10 = df_pre_phase2_top.head(10)
df_pre_phase2_top10 = pd.DataFrame(df_pre_phase2_to_10)
df_pre_phase2_top10 = df_pre_phase2_top10.reset_index(drop=True)
# Include percentage column
df_pre_phase2_top10['%'] = ((df_pre_phase2_top10['count'] / df_pre_phase2_top10['count'].sum())*100).round(2).astype(str) + '%'
df_pre_phase2_top10

# Visualisation of top 10 warnings distribution versus phases of care data points.
plt.figure(figsize=(9,8))
ax = plt.gca()
x = df_pre_phase2_top10['warning_status']
y = df_pre_phase2_top10['count']
n = ['Removed Order: Recovery & Post-Procedure Ward', 'Removed Order: Recovery & Post-Procedure Ward', 'Removed Order: Recovery (only)', 'Removed
Order: Recovery & Post-Procedure Ward', 'Removed Order: Recovery (only)', 'Removed Order: Recovery (only)', 'Removed Order: Intra-Procedure', 'Removed
Order: Recovery & Post-Procedure Ward, Removed Order: Recovery & Post-Procedure Ward', 'Removed Order: Post-Procedure Ward', 'Order 108979332:
Recovery & Post-Procedure Ward, Order 108979325: Intra-Procedure']
t = np.arange(10)
plt.title('Warning distribution versus phases of care (Top 10)')
plt.ylabel('Warning count by phases of care')
plt.xlabel('Warning alerts')
plt.scatter(x, y, c=t, s=100, alpha=1.0, marker='^')
```

```
plt.gcf().set_size_inches(14,10)
#scatter labels
for i, txt in enumerate(n):
    ax.annotate(txt, (x[i],y[i]))
plt.show()

# Phases of care by count
df_pre_phase1 = df_pr.pivot_table(index = ['phases_of_care'], aggfunc = 'count')
df_pre_phase2 = df_pre_phase1[['age']]
df_pre_phase3 = df_pre_phase2.sort_values(by = ['age'], ascending = False)
df_pre_phase4 = df_pre_phase3.reset_index()
df_pre_phase5 = df_pre_phase4.rename(columns={'age': 'count'})
# Drop unknowns and max column width.
df_pre_phase6 = df_pre_phase5.replace(-999, np.nan)
df_pre_phase7 = df_pre_phase6.dropna()
pd.set_option('display.max_colwidth', None)
# Include percentage column
df_pre_phase7['%'] = ((df_pre_phase7['count'] / df_pre_phase7['count'].sum())*100).round(2).astype(str) + '%'
df_pre_phase7.head(10)

# Top 75 list
df_pre_phase2_to_25 = df_pre_phase7.head(25)
df_pre_phase2_top25 = pd.DataFrame(df_pre_phase2_to_25)
df_pre_phase2_top25 = df_pre_phase2_top25.reset_index(drop=True)
df_pre_phase2_top25

df_pre_phase2_top25.dtypes

# Barchart of total count of warnings by 'warning status' and 'phases of care'.
ax=df_pre_phase2_top25[['phases_of_care','count']].plot(kind='bar', x='phases_of_care', y='count', title='Count by phases of care (Top 25).',
figsize=(8,6),legend=False, fontsize=10)
ax.set_xlabel('phases_of_care',fontsize=10)
ax.set_ylabel('count',fontsize=10)
plt.gcf().set_size_inches(13,8)


### 4.6. Warnings by context.
# Total number of warnings by 'warning status' and 'context'.
  # where 'size' is the count including NaN and repeated values, 'count'is the count excluding NaN but including repeats
    # and 'nunique'is the count of unique values, excluding repeats and NaN.
df_pre_contex = df_pr.groupby('warning_status')['context'].agg(['size', 'count', 'nunique'])

# Number of warning alerts by context and resetting index.
df_pre_contex = df_pre_contex.reset_index()
df_pre_contex1 = df_pre_contex.rename(columns = {'warning_status':'warning_status'})
df_pre_contex1 = df_pre_contex1[:]
df_pre_contex1

# Conformation survey barchart of total count of warnings by 'warning status' and 'context'.
ax=df_pre_contex1[['warning_status','count']].plot(kind='bar', x='warning_status', y='count', title='Warning status by context.', figsize=(8,6),legend=False,
fontsize=10)
ax.set_xlabel('warning_status',fontsize=10)
ax.set_ylabel("count",fontsize=10)
plt.gcf().set_size_inches(13,8)

# Conformation survey plot of total count of warnings by 'warning status' and 'context'.
df_pre_contex.plot()
plt.title('Warning status by context.')
plt.xlabel('context')
plt.ylabel('count')
plt.gcf().set_size_inches(13,8)

# Count of 'warning status' by 'context'.
df_pre_conte = df_pr.pivot_table(index = ['warning_status', 'context'], aggfunc = 'count')
df_pre_context = df_pre_conte[['age']]
df_pre_context0 = df_pre_context.rename(columns = {'age': 'count'}, inplace=True)
df_pre_context0 = df_pre_context.sort_values(by = ['count'], ascending = False)

# Warning status versus context - inpatient or outpatient.
df_pre_context1 = df_pre_context0.reset_index()
df_pre_context2 = df_pre_context1.rename(columns = {'warning_status':'warning_status'})

# Top 10 list.
df_pre_context2_t = df_pre_context2.replace(-999, np.nan)
df_pre_context2_top = df_pre_context2_t.dropna()
pd.set_option('display.max_colwidth', None)
df_pre_context2_to_10 = df_pre_context2_top.head(10)
df_pre_context2_top10 = pd.DataFrame(df_pre_context2_to_10)
df_pre_context2_top10 = df_pre_context2_top10.reset_index(drop=True)
# Include percentage column
df_pre_context2_top10['%'] = ((df_pre_context2_top10['count'] / df_pre_context2_top10['count'].sum())*100).round(2).astype(str) + '%'
df_pre_context2_top10

# Visualisation of top 10 warnings distribution versus context data points.
plt.figure(figsize=(9,7))
```

```
ax = plt.gca()
x = df_pre_context2_top10['warning_status']
y = df_pre_context2_top10['count']
n = ['Inpatient', 'Outpatient', 'Inpatient', 'Inpatient', 'unknown', 'Inpatient', 'Outpatient', 'Outpatient', 'Outpatient', 'unknown']
t = np.arange(10)
plt.title('Warning distribution versus context (Top 10)')
plt.ylabel('Warning count by context')
plt.xlabel('Warning alerts')
plt.scatter(x, y, c=t, s=100, alpha=1.0, marker='^')
plt.gcf().set_size_inches(13,8)
#scatter labels
for i, txt in enumerate(n):
    ax.annotate(txt, (x[i],y[i]))
plt.show()

# Context by count
df_pre_context1 = df_pr.pivot_table(index = ['context'], aggfunc = 'count')
df_pre_context2 = df_pre_context1[['age']]
df_pre_context3 = df_pre_context2.sort_values(by = ['age'], ascending = False)
df_pre_context4 = df_pre_context3.reset_index()
df_pre_context5 = df_pre_context4.rename(columns={'age': 'count'})
# Drop unknowns and max column width.
df_pre_context6 = df_pre_context5.replace(-999, np.nan)
df_pre_context7 = df_pre_context6.dropna()
pd.set_option('display.max_colwidth', None)
# Include percentage column
df_pre_context7['%'] = ((df_pre_context7['count'] / df_pre_context7['count'].sum())*100).round(2).astype(str) + '%'
df_pre_context7

# Top 25 list
df_pre_context2_to_25 = df_pre_context7.head(25)
df_pre_context2_top25 = pd.DataFrame(df_pre_context2_to_25)
df_pre_context2_top10 = df_pre_context2_top25.reset_index(drop=True)
df_pre_context2_top25

df_pre_context2_top25.dtypes

# Barchart of total count of warnings by 'warning status' and 'context'.
ax=df_pre_context2_top25[['context','count']].plot(kind='bar', x='context', y='count', title='Count by context.', figsize=(8,6),legend=False, fontsize=10)
ax.set_xlabel('context',fontsize=10)
ax.set_ylabel('count',fontsize=10)
plt.gcf().set_size_inches(13,8)


### 4.7. Warnings by severity.
# Total number of warnings by 'warning status' and 'severity'.
  # where 'size' is the count including NaN and repeated values, 'count'is the count excluding NaN but including repeats
    # and 'nunique'is the count of unique values, excluding repeats and NaN.
df_pre_sever = df_pr.groupby('warning_status')['severity'].agg(['size', 'count', 'nunique'])

# Number of warning alerts by severity and resetting index.
df_pre_sever = df_pre_sever.reset_index()
df_pre_sever1 = df_pre_sever.rename(columns = {'warning_status':'warning_status'})
df_pre_sever1 = df_pre_sever1[:]
df_pre_sever1

# Conformation survey barchart of total count of warnings by 'warning status' and 'severity'.
ax=df_pre_contex1[['warning_status','count']].plot(kind='bar', x='warning_status', y='count', title='Warning status by severity.', figsize=(8,6),legend=False,
fontsize=10)
ax.set_xlabel('warning_status',fontsize=10)
ax.set_ylabel("count",fontsize=10)
plt.gcf().set_size_inches(13,8)
# Conformation survey plot of total count of warnings by 'warning status' and 'severity'.
df_pre_sever.plot()
plt.title('Warning status by severity.')
plt.xlabel('severity')
plt.ylabel('count')
plt.gcf().set_size_inches(13,8)

# Count of 'warning status' by 'severity'.
df_pre_severit = df_pr.pivot_table(index = ['warning_status', 'severity'], aggfunc = 'count')
df_pre_severity = df_pre_severit[['age']]
df_pre_severity0 = df_pre_severity.rename(columns={'age': 'count'}, inplace=True)
df_pre_severity0 = df_pre_severity.sort_values(by = ['count'], ascending = False)

# Warning status versus severity.
df_pre_severity1 = df_pre_severity0.reset_index()
df_pre_severity2 = df_pre_severity1.rename(columns = {'warning_status':'warning_status'})

# Top 10 list.
df_pre_severity2_t = df_pre_severity2.replace(-999, np.nan)
df_pre_severity2_top = df_pre_severity2_t.dropna()
pd.set_option('display.max_colwidth', None)
df_pre_severity2_to_10 = df_pre_severity2_top.head(10)
df_pre_severity2_top10 = pd.DataFrame(df_pre_severity2_to_10)
```

```
df_pre_severity2_top10 = df_pre_severity2_top10.reset_index(drop=True)
# Include percentage column
df_pre_severity2_top10['%'] = ((df_pre_severity2_top10['count'] / df_pre_severity2_top10['count'].sum())*100).round(2).astype(str) + '%'
df_pre_severity2_top10

# Visualisation of top 10 warnings distribution versus severity data points.
plt.figure(figsize=(9,7))
ax = plt.gca()
x = df_pre_severity2_top10['warning_status']
y = df_pre_severity2_top10['count']
n = ['Cross-sensitive Class Match', 'Drug Class Match', 'Ingredient Match', 'Drug Class Match', 'Cross-sensitive Class Match', 'Cross-sensitive Class Match',
'Ingredient Match', 'Drug Class Match', 'Ingredient Match', 'Cross-sensitive Class Match']
t = np.arange(10)
plt.title('Warning distribution versus severity (Top 10)')
plt.ylabel('Warning count by severity')
plt.xlabel('Warning alerts')
plt.scatter(x, y, c=t, s=100, alpha=1.0, marker='^')
plt.gcf().set_size_inches(13,8)
#scatter labels
for i, txt in enumerate(n):
    ax.annotate(txt, (x[i],y[i]))
plt.show()

# Severity by count
df_pre_sever1 = df_pr.pivot_table(index = ['severity'], aggfunc = 'count')
df_pre_sever2 = df_pre_sever1[['age']]
df_pre_sever3 = df_pre_sever2.sort_values(by = ['age'], ascending = False)
df_pre_sever4 = df_pre_sever3.reset_index()
df_pre_sever5 = df_pre_sever4.rename(columns={'age': 'count'})
# Drop unknowns and max column width.
df_pre_sever6 = df_pre_sever5.replace(-999, np.nan)
df_pre_sever7 = df_pre_sever6.dropna()
pd.set_option('display.max_colwidth', None)
# Include percentage column
df_pre_sever7['%'] = ((df_pre_sever7['count'] / df_pre_sever7['count'].sum())*100).round(2).astype(str) + '%'

# Top 25 list
df_pre_severity2_a = df_pre_sever7.head(25)
df_pre_severity2_al = pd.DataFrame(df_pre_severity2_a)
df_pre_severity2_all = df_pre_severity2_al.reset_index(drop=True)
df_pre_severity2_all

df_pre_severity2_all.dtypes

# Barchart of total count of warnings by 'warning status' and 'severity'.
ax=df_pre_severity2_all[['severity','count']].plot(kind='bar', x='severity', y='count', title='Count by severity.', figsize=(8,6),legend=False, fontsize=10)
ax.set_xlabel('severity',fontsize=10)
ax.set_ylabel('count',fontsize=10)
plt.gcf().set_size_inches(13,8)


### 4.8. Warnings by D-A reason.
# Total number of warnings by 'warning status' and 'drug-allergy reason'.
  # where 'size' is the count including NaN and repeated values, 'count'is the count excluding NaN but including repeats
    # and 'nunique'is the count of unique values, excluding repeats and NaN.
df_pre_dareas = df_pr.groupby('warning_status')['drug_allergy_reason'].agg(['size', 'count', 'nunique'])

# Number of warning alerts by D-A reason and resetting index.
df_pre_dareas = df_pre_dareas.reset_index()
df_pre_dareas1 = df_pre_dareas.rename(columns = {'warning_status':'warning_status'})
df_pre_dareas1 = df_pre_dareas1[:]
df_pre_dareas1

# Conformation survey barchart of total count of warnings by 'warning status' and D-A reason.
ax=df_pre_dareas1[['warning_status','count']].plot(kind='bar', x='warning_status', y='count', title='Warning status by D-A reason.', figsize=(8,6),legend=False,
fontsize=10)
ax.set_xlabel('warning_status',fontsize=10)
ax.set_ylabel("count",fontsize=10)
plt.gcf().set_size_inches(13,8)
# Conformation survey plot of total count of warnings by 'warning status' and 'drug-allergy reason'.
df_pre_dareas.plot()
plt.title('Warning status by drug-allergy reason.')
plt.xlabel('drug-allergy reason')
plt.ylabel('count')
plt.gcf().set_size_inches(13,8)

# Count of 'warning status' by D-A reason.
df_pre_dareaso = df_pr.pivot_table(index = ['warning_status', 'drug_allergy_reason'], aggfunc = 'count')
df_pre_dareason = df_pre_dareaso[['age']]
df_pre_dareason0 = df_pre_dareason.rename(columns={'age' : 'count'}, inplace=True)
df_pre_dareason0 = df_pre_dareason.sort_values(by = ['count'], ascending = False)
# Warning status versus D-A reason.
df_pre_dareason1 = df_pre_dareason0.reset_index()
df_pre_dareason2 = df_pre_dareason1.rename(columns = {'warning_status':'warning_status'})
```

```
# Top 10 list.
df_pre_dareason2_t = df_pre_dareason2.replace(-999, np.nan)
df_pre_dareason2_top = df_pre_dareason2_t.dropna()
pd.set_option('display.max_colwidth', None)
df_pre_dareason2_to_10 = df_pre_dareason2_top.head(10)
df_pre_dareason2_top10 = pd.DataFrame(df_pre_dareason2_to_10)
df_pre_dareason2_top10 = df_pre_dareason2_top10.reset_index(drop=True)
# Include percentage column
df_pre_dareason2_top10['%'] = ((df_pre_dareason2_top10['count'] / df_pre_dareason2_top10['count'].sum())*100).round(2).astype(str) + '%'
df_pre_dareason2_top10

# Visualisation of top 10 warnings distribution versus severity data points.
plt.figure(figsize=(9,7))
ax = plt.gca()
x = df_pre_dareason2_top10['warning_status']
y = df_pre_dareason2_top10['count']
n = ['OPIOIDS-MORPHINE ANALOGUES (OXYCODONE,BUPRENORPHINE,CODEINE)', 'TRAMADOL', 'PENICILLINS', 'OPIOIDS-MORPHINE
ANALOGUES (OXYCODONE,BUPRENORPHINE,CODEINE)', 'OPIOIDS-MORPHINE ANALOGUES (OXYCODONE,BUPRENORPHINE,CODEINE)',
'PENICILLINS', 'CORTICOSTEROIDS', 'NSAIDS', 'NORMAL IMMUNOGLOBULIN HUMAN', 'NSAIDS']
t = np.arange(10)
plt.title('Warning distribution versus drug-allergy reason (Top 10)')
plt.ylabel('Warning count by drug-allergy reason')
plt.xlabel('Warning alerts')
plt.scatter(x, y, c=t, s=100, alpha=1.0, marker='^')
plt.gcf().set_size_inches(13,8)
#scatter labels
for i, txt in enumerate(n):
    ax.annotate(txt, (x[i],y[i]))
plt.show()

# Drug-allergy reason by count
df_pre_dareason1 = df_pr.pivot_table(index = ['drug_allergy_reason'], aggfunc = 'count')
df_pre_dareason2 = df_pre_dareason1[['age']]
df_pre_dareason3 = df_pre_dareason2.sort_values(by = ['age'], ascending = False)
df_pre_dareason4 = df_pre_dareason3.reset_index()
df_pre_dareason5 = df_pre_dareason4.rename(columns={'age': 'count'})
# Drop unknowns and max column width.
df_pre_dareason6 = df_pre_dareason5.replace(-999, np.nan)
df_pre_dareason7 = df_pre_dareason6.dropna()
pd.set_option('display.max_colwidth', None)
# Include percentage column
df_pre_dareason7['%'] = ((df_pre_dareason7['count'] / df_pre_dareason7['count'].sum())*100).round(2).astype(str) + '%'
df_pre_dareason7.head()

# Top 25 list
df_pre_dareason2_to_25 = df_pre_dareason7.head(25)
df_pre_dareason2_top25 = pd.DataFrame(df_pre_dareason2_to_25)
df_pre_dareason2_top25 = df_pre_dareason2_top25.reset_index(drop=True)
df_pre_dareason2_top25

df_pre_dareason2_top25.shape

# Barchart of total count of warnings by 'warning status' and 'drug-allergy reason'.
ax=df_pre_dareason2_top25[['count', 'drug_allergy_reason']].plot(kind='bar', x='drug_allergy_reason', y='count', title='Count by drug-allergy reason (Top 25).',
figsize=(8,6),legend=False, fontsize=10)
ax.set_xlabel('drug_allergy_reason',fontsize=10)
ax.set_ylabel('count',fontsize=10)
plt.gcf().set_size_inches(13,8)


### 4.9. Warnings by drug-allergy class pair for cross-sensitivity warning.
# Total number of warnings by 'warning status' and 'drug-allergy class pair for cross-sensitivity warning'.
  # where 'size' is the count including NaN and repeated values, 'count'is the count excluding NaN but including repeats
    # and 'nunique'is the count of unique values, excluding repeats and NaN.
df_pre_daclass = df_pr.groupby('warning_status')['drug_allergy_class_pair_for_cross_sensitivity_warning'].agg(['size', 'count', 'nunique'])

# Number of warning alerts by drug-allergy class pair for cross-sensitivity warning and resetting index.
df_pre_daclass = df_pre_daclass.reset_index()
df_pre_daclass1 = df_pre_daclass.rename(columns = {'warning_status':'warning_status'})
df_pre_daclass1 = df_pre_daclass1[:]
df_pre_daclass1

# Conformation survey barchart of total count of warnings by 'warning status' and drug-allergy class pair for cross-sensitivity warning.
ax=df_pre_daclass1[['warning_status','count']].plot(kind='bar', x='warning_status', y='count', title='Warning status by drug-allergy class pair for cross-sensitivity
warning.', figsize=(8,6),legend=False, fontsize=10)
ax.set_xlabel('warning_status',fontsize=10)
ax.set_ylabel("count",fontsize=10)
plt.gcf().set_size_inches(13,8)

# Conformation survey plot of total count of warnings by 'warning status' and 'drug-allergy class pair for cross-sensitivity warning'.
df_pre_daclass.plot()
plt.title('Warning status by drug-allergy class pair for cross-sensitivity warning.')
plt.xlabel('drug-allergy class pair for cross-sensitivity warning')
plt.ylabel('count')
plt.gcf().set_size_inches(13,8)
```

```
# Count of 'warning status' by 'drug-allergy class pair for cross-sensitivity warning'.
df_pre_daclassp = df_pr.pivot_table(index = ['warning_status', 'drug_allergy_class_pair_for_cross_sensitivity_warning'], aggfunc = 'count')
df_pre_daclasspair = df_pre_daclassp[['age']]
df_pre_daclasspair0 = df_pre_daclasspair.rename(columns={'age' : 'count'}, inplace=True)
df_pre_daclasspair0 = df_pre_daclasspair.sort_values(by = ['count'], ascending = False)

# Warning status versus drug-allergy class pair for cross-sensitivity warning.
df_pre_daclasspair1 = df_pre_daclasspair0.reset_index()
df_pre_daclasspair2 = df_pre_daclasspair1.rename(columns = {'warning_status':'warning_status'})

# Top 10 list.
df_pre_daclasspair2_t = df_pre_daclasspair2.replace(-999, np.nan)
df_pre_daclasspair2_top = df_pre_daclasspair2_t.dropna()
pd.set_option('display.max_colwidth', None)
df_pre_daclasspair2_to_10 = df_pre_daclasspair2_top.head(10)
df_pre_daclasspair2_top10 = pd.DataFrame(df_pre_daclasspair2_to_10)
df_pre_daclasspair2_top10 = df_pre_daclasspair2_top10.reset_index(drop=True)
# Include percentage column
df_pre_daclasspair2_top10['%'] = ((df_pre_daclasspair2_top10['count'] / df_pre_daclasspair2_top10['count'].sum())*100).round(2).astype(str) + '%'
df_pre_daclasspair2_top10

# Visualisation of top 10 warnings distribution versus drug-allergy class pair for cross-sensitivity warning data points.
plt.figure(figsize=(9,7))
ax = plt.gca()
x = df_pre_daclasspair2_top10['warning_status']
y = df_pre_daclasspair2_top10['count']
n = ['OPIOIDS-MORPHINE ANALOGUES (OXYCODONE,BUPRENORPHINE,CODEINE) - TRAMADOL', 'OPIOIDS-FENTANYL AND PETHIDINE
ANALOGUES - OPIOIDS-MORPHINE ANALOGUES (OXYCODONE,BUPRENORPHINE,CODEINE)', 'OPIOIDS-FENTANYL AND PETHIDINE
ANALOGUES - OPIOIDS-MORPHINE ANALOGUES (OXYCODONE,BUPRENORPHINE,CODEINE)', 'CEPHALOSPORINS - PENICILLINS',
'CEPHALOSPORINS - PENICILLINS', 'CEPHALOSPORINS - PENICILLINS', 'NSAIDS - SALICYLATES', 'OPIOIDS-MORPHINE ANALOGUES
(OXYCODONE,BUPRENORPHINE,CODEINE) - TRAMADOL', 'OPIOIDS-FENTANYL AND PETHIDINE ANALOGUES - TRAMADOL',
'CEPHALOSPORINS - PENICILLINS']
t = np.arange(10)
plt.title('Warning distribution versus drug-allergy class pair for cross-sensitivity warning warning (Top 10)')
plt.ylabel('Warning count by drug-allergy class pair for cross-sensitivity warning')
plt.xlabel('Warning alerts')
plt.scatter(x, y, c=t, s=100, alpha=1.0, marker='^')
plt.gcf().set_size_inches(13,8)
#scatter labels
for i, txt in enumerate(n):
    ax.annotate(txt, (x[i],y[i]))
plt.show()

# drug-allergy class pair for cross-sensitivity warning by count
df_pre_daclasspair1 = df_pr.pivot_table(index = ['drug_allergy_class_pair_for_cross_sensitivity_warning'], aggfunc = 'count')
df_pre_daclasspair2 = df_pre_daclasspair1[['age']]
df_pre_daclasspair3 = df_pre_daclasspair2.sort_values(by = ['age'], ascending = False)
df_pre_daclasspair4 = df_pre_daclasspair3.reset_index()
df_pre_daclasspair5 = df_pre_daclasspair4.rename(columns={'age': 'count'})
# Drop unknowns and max column width.
df_pre_daclasspair6 = df_pre_daclasspair5.replace(-999, np.nan)
df_pre_daclasspair7 = df_pre_daclasspair6.dropna()
pd.set_option('display.max_colwidth', None)
# Include percentage column
df_pre_daclasspair7['%'] = ((df_pre_daclasspair7['count'] / df_pre_daclasspair7['count'].sum())*100).round(2).astype(str) + '%'
df_pre_daclasspair7.head()

# Top 25 list
df_pre_daclasspair2_to_25 = df_pre_daclasspair7.head(25)
df_pre_daclasspair2_top25 = pd.DataFrame(df_pre_daclasspair2_to_25)
df_pre_daclasspair2_top25 = df_pre_daclasspair2_top25.reset_index(drop=True)
df_pre_daclasspair2_top25

df_pre_daclasspair2_top25.shape

# Barchart of total count of warnings by 'warning status' and 'drug-allergy class pair for cross-sensitivity warning'.
ax=df_pre_daclasspair2_top25[['drug_allergy_class_pair_for_cross_sensitivity_warning','count']].plot(kind='bar',
x='drug_allergy_class_pair_for_cross_sensitivity_warning', y='count', title='Count by drug-allergy class pair for cross-sensitivity warning (Top 25).',
figsize=(8,6),legend=False, fontsize=10)
ax.set_xlabel('drug_allergy_class_pair_for_cross_sensitivity_warning',fontsize=10)
ax.set_ylabel('count', fontsize=10)
plt.gcf().set_size_inches(13,8)
```

4.10. Warnings by D-A contra group.

```
# Total number of warnings by 'warning status' and 'drug-allergy contraindication group'.
  # where 'size' is the count including NaN and repeated values, 'count'is the count excluding NaN but including repeats
    # and 'nunique'is the count of unique values, excluding repeats and NaN.
df_pre_dacontra = df_pr.groupby('warning_status')['drug_allergy_contraindication_group'].agg(['size', 'count', 'nunique'])
# Number of warning alerts by D-A contra group and resetting index.
df_pre_dacontra = df_pre_dacontra.reset_index()
df_pre_dacontra1 = df_pre_dacontra.rename(columns = {'warning_status':'warning_status'})
df_pre_dacontra1 = df_pre_dacontra1[:]
```

```
df_pre_dacontra1

# Conformation survey barchart of total count of warnings by 'warning status' and D-A contra group.
ax=df_pre_daclass1[['warning_status','count']].plot(kind='bar', x='warning_status', y='count', title='Warning status by D-A contra group.',
figsize=(8,6),legend=False, fontsize=10)
ax.set_xlabel('warning_status',fontsize=10)
ax.set_ylabel("count",fontsize=10)
plt.gcf().set_size_inches(13,8)

# Conformation survey plot of total count of warnings by 'warning status' and 'drug-allergy contraindication group'.
df_pre_dacontra.plot()
plt.title('Warning status by drug-allergy contraindication group.')
plt.xlabel('drug-allergy contraindication group')
plt.ylabel('count')
plt.gcf().set_size_inches(13,8)

# Count of D-A contra group.
df_pre_dacontragrp = df_pr.pivot_table(index = ['warning_status', 'drug_allergy_contraindication_group'], aggfunc = 'count')
df_pre_dacontragroup = df_pre_dacontragrp[['age']]
df_pre_dacontragroup0 = df_pre_dacontragroup.rename(columns={'age': 'count'}, inplace=True)
df_pre_dacontragroup0 = df_pre_dacontragroup.sort_values(by = ['count'], ascending = False)

# Warning status versus drug-allergy contraindication group.
df_pre_dacontragroup1 = df_pre_dacontragroup0.reset_index()
df_pre_dacontragroup2 = df_pre_dacontragroup1.rename(columns = {'warning_status':'warning_status'})

# Top 10 list.
df_pre_dacontragroup2_t = df_pre_dacontragroup2.replace(-999, np.nan)
df_pre_dacontragroup2_top = df_pre_dacontragroup2_t.dropna()
pd.set_option('display.max_colwidth', None)
df_pre_dacontragroup2_to_10 = df_pre_dacontragroup2_top.head(10)
df_pre_dacontragroup2_top10 = pd.DataFrame(df_pre_dacontragroup2_to_10)
df_pre_dacontragroup2_top10 = df_pre_dacontragroup2_top10.reset_index(drop=True)
# Include percentage column
df_pre_dacontragroup2_top10['%'] = ((df_pre_dacontragroup2_top10['count'] / df_pre_dacontragroup2_top10['count'].sum())*100).round(2).astype(str) + '%'
df_pre_dacontragroup2_top10

df_pre_dacontragroup2_top10.dtypes

# Visualisation of warning distribution versus drug-allergy contraindication grouping.
plt.figure(figsize=(9,7))
ax = plt.gca()
x = df_pre_dacontragroup2_top10['warning_status']
y = df_pre_dacontragroup2_top10['count']
n = ['Allergies', 'Allergies', 'Adverse Reactions/Drug Intolerances', 'Allergies', 'Allergies', 'Adverse Reactions/Drug Intolerances', 'Adverse Reactions/Drug
Intolerances', 'Adverse Reactions/Drug Intolerances', 'Allergies', 'Adverse Reactions/Drug Intolerances']
t = np.arange(10)
plt.title('Warning distribution versus drug-allergy contraindication group')
plt.ylabel('Warning count by drug-allergy contraindication group')
plt.xlabel('Warning alerts')
plt.scatter(x, y, c=t, s=100, alpha=1.0, marker='^')
plt.gcf().set_size_inches(13,8)
#scatter labels
for i, txt in enumerate(n):
    ax.annotate(txt, (x[i],y[i]))
plt.show()

# Overall drug-allergy contraindication group by count
df_pre_dacontragroup1 = df_pr.pivot_table(index = ['drug_allergy_contraindication_group'], aggfunc = 'count')
df_pre_dacontragroup2 = df_pre_dacontragroup1[['age']]
df_pre_dacontragroup3 = df_pre_dacontragroup2.sort_values(by = ['age'], ascending = False)
df_pre_dacontragroup4 = df_pre_dacontragroup3.reset_index()
df_pre_dacontragroup5 = df_pre_dacontragroup4.rename(columns={'age': 'count'})
# Drop unknowns and max column width.
df_pre_dacontragroup6 = df_pre_dacontragroup5.replace(0, np.nan)
df_pre_dacontragroup7 = df_pre_dacontragroup6.dropna()
pd.set_option('display.max_colwidth', None)
# Include percentage column
df_pre_dacontragroup7['%'] = ((df_pre_dacontragroup7['count'] / df_pre_dacontragroup7['count'].sum())*100).round(2).astype(str) + '%'
df_pre_dacontragroup7

df_pre_dacontragroup7.shape

# Barchart of total count of warnings by 'warning status' and 'drug-allergy contraindication group'.
ax=df_pre_dacontragroup7[['drug_allergy_contraindication_group','count']].plot(kind='bar', x='drug_allergy_contraindication_group', y='count', title='Warning
alert count by drug-allergy contraindication group.', figsize=(8,6),legend=False, fontsize=10)
ax.set_xlabel('drug-allergy-contraindication_group',fontsize=10)
ax.set_ylabel('count',fontsize=10)
plt.gcf().set_size_inches(13,8)
```

4.11 Warnings by order set.

```
# Total number of warnings by 'warning status' and 'order sets'.
  # where 'size' is the count including NaN and repeated values, 'count'is the count excluding NaN but including repeats
    # and 'nunique'is the count of unique values, excluding repeats and NaN.
```

```
df_pre_order = df_pr.groupby('warning_status')['order_sets'].agg(['size', 'count', 'nunique'])

# Number of warning alerts by order sets and resetting index.
df_pre_order = df_pre_order.reset_index()
df_pre_order1 = df_pre_order.rename(columns = {'warning_status':'warning_status'})
df_pre_order1 = df_pre_order1[:]
df_pre_order1

# Conformation survey barchart of total count of warnings by 'warning status' and order sets.
ax=df_pre_order1[['warning_status','count']].plot(kind='bar', x='warning_status', y='count', title='Warning status by order sets.', figsize=(8,6),legend=False,
fontsize=10)
ax.set_xlabel('warning_status',fontsize=10)
ax.set_ylabel("count",fontsize=10)
plt.gcf().set_size_inches(13,8)
# Conformation survey plot of total count of warnings by 'warning status' and 'order sets'.
df_pre_order.plot()
plt.title('Warning status by order sets.')
plt.xlabel('order sets')
plt.ylabel('count')
plt.gcf().set_size_inches(13,8)

# Count of 'warning status' by 'order sets'.
df_pre_orderse = df_pr.pivot_table(index = ['warning_status', 'order_sets'], aggfunc = 'count')
df_pre_orderset = df_pre_orderse[['age']]
df_pre_orderset0 = df_pre_orderset.rename(columns={'age': 'count'}, inplace = True)
df_pre_orderset0 = df_pre_orderset.sort_values(by = ['count'], ascending = False)

# Warning status versus order sets.
df_pre_orderset1 = df_pre_orderset0.reset_index()
df_pre_orderset2 = df_pre_orderset1.rename(columns = {'warning_status':'warning_status'})

# Top 10 list.
df_pre_orderset2_t = df_pre_orderset2.replace(-999, np.nan)
df_pre_orderset2_top = df_pre_orderset2_t.dropna()
pd.set_option('display.max_colwidth', None)
df_pre_orderset2_to_10 = df_pre_orderset2_top.head(10)
df_pre_orderset2_top10 = pd.DataFrame(df_pre_orderset2_to_10)
df_pre_orderset2_top10 = df_pre_orderset2_top10.reset_index(drop=True)
# Include percentage column
df_pre_orderset2_top10['%'] = ((df_pre_orderset2_top10['count'] / df_pre_orderset2_top10['count'].sum())*100).round(2).astype(str) + '%'
df_pre_orderset2_top10

df_pre_orderset2_top10.shape

# Visualisation of warning distribution versus order sets.
plt.figure(figsize=(9,7))
ax = plt.gca()
x = df_pre_orderset2_top10['warning_status']
y = df_pre_orderset2_top10['count']
n = ['UCLH IP ANAESTHETIC POST-PROCEDURE ORDERS - ADULTS', 'UCLH IP ANAESTHESIA DAY CASE POST PROCEDURE ORDERS -
ADULTS', 'UCLH IP ANAESTHESIA POST-PROCEDURE NEURO ORDERS - ADULT', 'UCLH IP ANAESTHESIA POST-PROCEDURE OBSTETRIC
ORDERS', 'ED UCLH MEDICATIONS', 'UCLH IP ANAESTHETIC POST-PROCEDURE ORDERS - ADULTS', 'UCLH IP ANAESTHETIC POST-
PROCEDURE ORDERS - ADULTS', 'UCLH IP ANAESTHESIA DAY CASE POST PROCEDURE ORDERS - ADULTS', 'UCLH CT CONTRAST', 'UCLH
IP ANAESTHETIC POST-PROCEDURE ORDERS - ADULTS']
t = np.arange(10)
plt.title('Warning distribution versus order sets')
plt.ylabel('Warning count by order sets')
plt.xlabel('Warning alerts')
plt.scatter(x, y, c=t, s=100, alpha=1.0, marker='^')
plt.gcf().set_size_inches(13,8)
#scatter labels
for i, txt in enumerate(n):
    ax.annotate(txt, (x[i],y[i]))
plt.show()

# Order sets by count
df_pre_orderset1 = df_pr.pivot_table(index = ['order_sets'], aggfunc = 'count')
df_pre_orderset2 = df_pre_orderset1[['age']]
df_pre_orderset3 = df_pre_orderset2.sort_values(by = ['age'], ascending = False)
df_pre_orderset4 = df_pre_orderset3.reset_index()
df_pre_orderset5 = df_pre_orderset4.rename(columns={'age': 'count'})
# Drop unknowns and max column width.
df_pre_orderset6 = df_pre_orderset5.replace(-999, np.nan)
df_pre_orderset7 = df_pre_orderset6.dropna()
pd.set_option('display.max_colwidth', None)
# Include percentage column
df_pre_orderset7['%'] = ((df_pre_orderset7['count'] / df_pre_orderset7['count'].sum())*100).round(2).astype(str) + '%'
df_pre_orderset7.head()

# Top 25 list
df_pre_orderset2_to_25 = df_pre_orderset7.head(25)
df_pre_orderset2_top25 = pd.DataFrame(df_pre_orderset2_to_25)
df_pre_orderset2_top25 = df_pre_orderset2_top25.reset_index(drop=True)
df_pre_orderset2_top25
```

```
df_pre_orderset2_top25.shape

# Barchart of total count of warnings by 'warning status' and 'order sets'.
ax=df_pre_orderset2_top25[['order_sets','count']].plot(kind='bar', x='order_sets', y='count', title='Warning alert count by order sets.', figsize=(8,6),legend=False,
fontsize=10)
ax.set_xlabel('order_sets',fontsize=10)
ax.set_ylabel('count',fontsize=10)
plt.gcf().set_size_inches(13,8)


### 4.12 Warnings by interaction setting
# Total number of warnings by 'warning status' and 'order sets'.
  # where 'size' is the count including NaN and repeated values, 'count'is the count excluding NaN but including repeats
    # and 'nunique'is the count of unique values, excluding repeats and NaN.
df_pre_inter = df_pr.groupby('warning_status')['interaction_setting'].agg(['size', 'count', 'nunique'])

# Number of warning alerts by order sets and resetting index.
df_pre_inter = df_pre_inter.reset_index()
df_pre_inter1 = df_pre_inter.rename(columns = {'warning_status':'warning_status'})
df_pre_inter1 = df_pre_inter1[:]
df_pre_inter1

# Conformation survey barchart of total count of warnings by 'warning status' and interaction setting.
ax=df_pre_inter1[['warning_status','count']].plot(kind='bar', x='warning_status', y='count', title='Warning status by interaction setting.',
figsize=(8,6),legend=False, fontsize=10)
ax.set_xlabel('warning_status',fontsize=10)
ax.set_ylabel('count',fontsize=10)
plt.gcf().set_size_inches(13,8)

# Conformation survey plot of total count of warnings by 'warning status' and 'interaction setting'.
df_pre_inter1.plot()
plt.title('Warning status by interaction setting.')
plt.xlabel('interaction setting')
plt.ylabel('count')
plt.gcf().set_size_inches(13,8)

# Count of 'warning status' by interaction setting.
df_pre_interse = df_pr.pivot_table(index = ['warning_status', 'interaction_setting'], aggfunc = 'count')
df_pre_interset = df_pre_orderse[['age']]
df_pre_interset0 = df_pre_interset.rename(columns={'age': 'count'}, inplace = True)
df_pre_interset0 = df_pre_interset.sort_values(by = ['count'], ascending = False)
# Warning status versus interaction setting.
df_pre_interset1 = df_pre_interset0.reset_index()
df_pre_interset2 = df_pre_interset1.rename(columns = {'warning_status':'warning_status'})

# Top 10 list
df_pre_interset2_t = df_pre_interset2.replace(-999, np.nan)
df_pre_interset2_top = df_pre_interset2_t.dropna()
pd.set_option('display.max_colwidth', None)
df_pre_interset2_to_10 = df_pre_interset2_top.head(10)
df_pre_interset2_top10 = pd.DataFrame(df_pre_interset2_to_10)
df_pre_interset2_top10 = df_pre_interset2_top10.reset_index(drop=True)
# Include percentage column
df_pre_interset2_top10['%'] = ((df_pre_interset2_top10['count'] / df_pre_interset2_top10['count'].sum())*100).round(2).astype(str) + '%'
df_pre_interset2_top10

df_pre_interset2_top10.shape

# Visualisation of warning distribution versus interaction settings.
plt.figure(figsize=(9,7))
ax = plt.gca()
x = df_pre_interset2_top10['warning_status']
y = df_pre_interset2_top10['count']
n = ['UCLH IP ANAESTHETIC POST-PROCEDURE ORDERS - ADULTS', 'UCLH IP ANAESTHESIA DAY CASE POST PROCEDURE ORDERS -
ADULTS', 'UCLH IP ANAESTHESIA POST-PROCEDURE NEURO ORDERS - ADULT', 'UCLH IP ANAESTHESIA POST-PROCEDURE OBSTETRIC
ORDERS', 'ED UCLH MEDICATIONS', 'UCLH IP ANAESTHETIC POST-PROCEDURE ORDERS - ADULTS', 'UCLH IP ANAESTHETIC POST-
PROCEDURE ORDERS - ADULTS', 'UCLH IP ANAESTHESIA DAY CASE POST PROCEDURE ORDERS - ADULTS', 'UCLH CT CONTRAST', 'UCLH
IP ANAESTHETIC POST-PROCEDURE ORDERS - ADULTS']
t = np.arange(10)
plt.title('Warning distribution versus order sets')
plt.ylabel('Warning count by order sets')
plt.xlabel('Warning alerts')
plt.scatter(x, y, c=t, s=100, alpha=1.0, marker='^')
plt.gcf().set_size_inches(13,8)
#scatter labels
for i, txt in enumerate(n):
    ax.annotate(txt, (x[i],y[i]))
plt.show()

# Interaction setting by count
df_pre_interset1 = df_pr.pivot_table(index = ['interaction_setting'], aggfunc = 'count')
df_pre_interset2 = df_pre_interset1[['age']]
df_pre_interset3 = df_pre_interset2.sort_values(by = ['age'], ascending = False)
df_pre_interset4 = df_pre_interset3.reset_index()
df_pre_interset5 = df_pre_interset4.rename(columns={'age': 'count'})
```

```
# Drop unknowns and max column width.
df_pre_interset6 = df_pre_interset5.replace(-999, np.nan)
df_pre_interset7 = df_pre_interset6.dropna()
pd.set_option('display.max_colwidth', None)
# Include percentage column
df_pre_interset7['%'] = ((df_pre_interset7['count'] / df_pre_interset7['count'].sum())*100).round(2).astype(str) + '%'
df_pre_interset7

df_pre_interset7.dtypes

# Barchart of total count of warnings by 'warning status' and 'interaction setting'.
ax=df_pre_interset7[['interaction_setting','count']].plot(kind='bar', x='interaction_setting', y='count', title='Count by order sets.', figsize=(8,6),legend=False,
fontsize=10)
ax.set_xlabel('interaction_setting',fontsize=10)
ax.set_ylabel('count',fontsize=10)
plt.gcf().set_size_inches(13,8)
```

4.13. Warnings by patient_hospital/clinic

```
# Total number of warnings by 'warning status' and 'patient hospital/clinic'.
  # where 'size' is the count including NaN and repeated values, 'count'is the count excluding NaN but including repeats
    # and 'nunique'is the count of unique values, excluding repeats and NaN.
df_pre_pathc = df_pr.groupby('warning_status')['patient_hospital_clinic'].agg(['size', 'count', 'nunique'])
# Number of warning alerts by patient hospital/clinic and resetting index.
df_pre_pathc = df_pre_pathc.reset_index()
df_pre_pathc1 = df_pre_pathc.rename(columns = {'warning_status':'warning_status'})
df_pre_pathc1 = df_pre_pathc1[:]
df_pre_pathc1

# Conformation survey barchart of total count of warnings by 'warning status' and patient hospital/clinic.
ax=df_pre_pathc1[['warning_status','count']].plot(kind='bar', x='warning_status', y='count', title='Warning status by patient hospital/clinic.',
figsize=(8,6),legend=False, fontsize=10)
ax.set_xlabel('warning_status',fontsize=10)
ax.set_ylabel("count",fontsize=10)
plt.gcf().set_size_inches(13,8)

# Conformation survey plot of total count of warnings by 'warning status' and 'patient hospital/clinic'.
df_pre_pathc1.plot()
plt.title('Warning_status_by_patient_hospital_clinic.')
plt.xlabel('patient_hospital_clinic')
plt.ylabel('count')
plt.gcf().set_size_inches(13,8)

# Count of 'warning status' by 'patient hospital/clinic'.
df_pre_pathcli = df_pr.pivot_table(index = ['warning_status', 'patient_hospital_clinic'], aggfunc = 'count')
df_pre_pathclin = df_pre_pathcli[['age']]
df_pre_pathclin0 = df_pre_pathclin.rename(columns={'age': 'count'}, inplace = True)
df_pre_pathclin0 = df_pre_pathclin.sort_values(by = ['count'], ascending = False)

# Warning status versus patient hospital/clinic.
df_pre_pathclin1 = df_pre_pathclin0.reset_index()
df_pre_pathclin2 = df_pre_pathclin1.rename(columns = {'warning_status':'warning_status'})

df_pre_pathclin.shape

# Top 10 list.
df_pre_pathclin2_t = df_pre_pathclin2.replace(-999, np.nan)
df_pre_pathclin2_top = df_pre_pathclin2_t.dropna()
pd.set_option('display.max_colwidth', None)
df_pre_pathclin2_to_10 = df_pre_pathclin2_top.head(10)
df_pre_pathclin2_top10 = pd.DataFrame(df_pre_pathclin2_to_10)
df_pre_pathclin2_top10 = df_pre_pathclin2_top10.reset_index(drop=True)
# Include percentage column
df_pre_pathclin2_top10['%'] = ((df_pre_pathclin2_top10['count'] / df_pre_pathclin2_top10['count'].sum())*100).round(2).astype(str) + '%'
df_pre_pathclin2_top10

df_pre_pathclin2_top10.dtypes

# Visualisation of warning distribution versus patient hospital/clinic.
plt.figure(figsize=(9,7))
ax = plt.gca()
x = df_pre_pathclin2_top10['warning_status']
y = df_pre_pathclin2_top10['count']
n = ['UNIVERSITY COLLEGE HOSPITAL CAMPUS', 'UNIVERSITY COLLEGE HOSPITAL CAMPUS', 'MACMILLAN CANCER CENTRE',
'UNIVERSITY COLLEGE HOSPITAL CAMPUS', 'QUEEN SQUARE CAMPUS', 'WESTMORELAND STREET', 'UNIVERSITY COLLEGE HOSPITAL
CAMPUS', 'WESTMORELAND STREET', 'QUEEN SQUARE CAMPUS', 'WESTMORELAND STREET']
t = np.arange(10)
plt.title('Warning distribution versus patient hospital/clinic')
plt.ylabel('Warning count by patient hospital/clinic')
plt.xlabel('Warning alerts')
plt.scatter(x, y, c=t, s=100, alpha=1.0, marker='^')
plt.gcf().set_size_inches(13,8)
#scatter labels
for i, txt in enumerate(n):
    ax.annotate(txt, (x[i],y[i]))
```

```
plt.show()

# patient hospital/clinic by count
df_pre_pathclinic1 = df_pr.pivot_table(index = ['patient_hospital_clinic'], aggfunc = 'count')
df_pre_pathclinic2 = df_pre_pathclinic1[['age']]
df_pre_pathclinic3 = df_pre_pathclinic2.sort_values(by = ['age'], ascending = False)
df_pre_pathclinic4 = df_pre_pathclinic3.reset_index()
df_pre_pathclinic5 = df_pre_pathclinic4.rename(columns={'age' : 'count'})

# Drop unknowns and max column width.
df_pre_pathclinic6 = df_pre_pathclinic5.replace(-999, np.nan)
df_pre_pathclinic7 = df_pre_pathclinic6.dropna()
pd.set_option('display.max_colwidth', None)

# Include percentage column
df_pre_pathclinic7['%'] = ((df_pre_pathclinic7['count'] / df_pre_pathclinic7['count'].sum())*100).round(2).astype(str) + '%'
df_pre_pathclinic7

df_pre_pathclinic7.shape

# Barchart of total count of warnings by 'warning status' and 'patient hospital/clinic'.
ax=df_pre_pathclinic7[['patient_hospital_clinic','count']].plot(kind='bar', x='patient_hospital_clinic', y='count', title='Count by patient hospital/clinic.',
figsize=(8,6),legend=False, fontsize=10)
ax.set_xlabel('patient_hospital_clinic',fontsize=10)
ax.set_ylabel('count',fontsize=10)
plt.gcf().set_size_inches(13,8)


### 4.14. Warnings by drug_allergy_reactions
# Total number of warnings by 'warning status' and 'drug_allergy_reactions'.
  # where 'size' is the count including NaN and repeated values, 'count'is the count excluding NaN but including repeats
    # and 'nunique'is the count of unique values, excluding repeats and NaN.
df_pre_dar = df_pr.groupby('warning_status')['drug_allergy_reactions'].agg(['size', 'count', 'nunique'])

# Number of warning alerts by drug_allergy_reactions and resetting index.
df_pre_c1_dar = df_pre_dar.reset_index()
df_pre_c1_dar = df_pre_c1_dar.rename(columns = {'warning_status':'warning_status'})
df_prec_1_dar = df_pre_c1_dar[:]
df_pre_c1_dar

# Conformation survey barchart of total count of warnings by 'warning status' and drug_allergy_reactions.
ax=df_pre_c1_dar[['warning_status','count']].plot(kind='bar', x='warning_status', y='count', title='Warning status by drug_allergy_reactions.',
figsize=(8,6),legend=False, fontsize=10)
ax.set_xlabel('warning_status',fontsize=10)
ax.set_ylabel("count",fontsize=10)
plt.gcf().set_size_inches(13,8)

# Conformation survey plot of total count of warnings by 'warning status' and 'drug_allergy_reactions'.
df_pre_c1_dar.plot()
plt.title('Warning_status_by_drug_allergy_reactions.')
plt.xlabel('drug_allergy_reactions')
plt.ylabel('count')
plt.gcf().set_size_inches(13,8)


# Count of 'warning status' by 'drug_allergy_reactions'.
df_pre_c2_dar = df_pr.pivot_table(index = ['warning_status', 'drug_allergy_reactions'], aggfunc = 'count')
df_pre_c2_dar_a = df_pre_c2_dar[['age']]
df_pre_c2_dar_b = df_pre_c2_dar_a.rename(columns={'age': 'count'}, inplace = True)
df_pre_c2_dar_b = df_pre_c2_dar_a.sort_values(by = ['count'], ascending = False)

# Warning status versus drug_allergy_reactions.
df_pre_c2_dar_c = df_pre_c2_dar_b.reset_index()
df_pre_c2_dar_d = df_pre_c2_dar_c.rename(columns = {'warning_status':'warning_status'})

df_pre_c2_dar_a.shape

# Top 10 list.
df_pre_c2_dar_d_t = df_pre_c2_dar_d.replace(-999, np.nan)
df_pre_c2_dar_d_top = df_pre_c2_dar_d_t.dropna()
pd.set_option('display.max_colwidth', None)
df_pre_c2_dar_d_to_10 = df_pre_c2_dar_d_top.head(10)
df_pre_c2_dar_d_top10 = pd.DataFrame(df_pre_c2_dar_d_to_10)
df_pre_c2_dar_d_top10 = df_pre_c2_dar_d_top10.reset_index(drop=True)
# Include percentage column
df_pre_c2_dar_d_top10['%'] = ((df_pre_c2_dar_d_top10['count'] / df_pre_c2_dar_d_top10['count'].sum())*100).round(2).astype(str) + '%'
df_pre_c2_dar_d_top10

df_pre_c2_dar_d_top10.dtypes

# Visualisation of warning distribution versus drug_allergy_reactions.
plt.figure(figsize=(9,7))
ax = plt.gca()
x = df_pre_c2_dar_d_top10['warning_status']
y = df_pre_c2_dar_d_top10['count']
```

```
n = ['Other (see comments)', 'Rash, itching or hives', 'Other (see comments)', 'Other (see comments)', 'Rash, itching or hives', 'Rash, itching or hives',
'Anaphylaxis', 'Shortness of breath', 'Other (see comments)', 'Swelling']
t = np.arange(10)
plt.title('Warning distribution versus drug_allergy_reactions')
plt.ylabel('Warning count by drug_allergy_reactions')
plt.xlabel('Warning alerts')
plt.scatter(x, y, c=t, s=100, alpha=1.0, marker='^')
plt.gcf().set_size_inches(13,8)
#scatter labels
for i, txt in enumerate(n):
    ax.annotate(txt, (x[i],y[i]))
plt.show()

# drug_allergy_reactions by count
df_pre_c2_dar_d1 = df_pr.pivot_table(index = ['drug_allergy_reactions'], aggfunc = 'count')
df_pre_c2_dar_d2 = df_pre_c2_dar_d1[['age']]
df_pre_c2_dar_d3 = df_pre_c2_dar_d2.sort_values(by = ['age'], ascending = False)
df_pre_c2_dar_d4 = df_pre_c2_dar_d3.reset_index()
df_pre_c2_dar_d5 = df_pre_c2_dar_d4.rename(columns={'age' : 'count'})
# Drop unknowns and max column width.
df_pre_c2_dar_d6 = df_pre_c2_dar_d5.replace(-999, np.nan)
df_pre_c2_dar_d7 = df_pre_c2_dar_d6.dropna()
pd.set_option('display.max_colwidth', None)
# Include percentage column
df_pre_c2_dar_d7['%'] = ((df_pre_c2_dar_d7['count'] / df_pre_c2_dar_d7['count'].sum())*100).round(2).astype(str) + '%'
df_pre_c2_dar_d7

df_pre_c2_dar_d7.shape

# Barchart of total count of warnings by 'warning status' and 'drug_allergy_reactions'.
ax=df_pre_c2_dar_d7[['drug_allergy_reactions','count']].plot(kind='bar', x='drug_allergy_reactions', y='count', title='Count by drug_allergy_reactions.',
figsize=(8,6),legend=False, fontsize=10)
ax.set_xlabel('drug_allergy_reactions',fontsize=10)
ax.set_ylabel('count',fontsize=10)
plt.gcf().set_size_inches(13,8)
# Convert dataframe to csv.
#df_pre_c2_dar_d7.to_csv('df_pre_c2_dar_d7_120821.csv')
df_pre_c2_dar_d7

## 5. Dimensionality reduction
# Hierarchical indexing by 'description', 'warning status', 'provider type', 'provider specialty', 'interaction setting', 'patient hospital/clinic', 'phases of care',
'context', 'drug_allergy_reactions', override_reason', 'override_comment', 'related_actions', age_in_years', 'age_range', 'sex'.
df_pre_h1 = df_pr.set_index(['description', 'warning_status', 'source', 'severity', 'provider_type', 'provider_specialty', 'interaction_setting',
'patient_hospital_clinic', 'phases_of_care', 'context', 'drug_allergy_contraindication_group', 'drug_allergy_reason',
'drug_allergy_class_pair_for_cross_sensitivity_warning', 'drug_allergy_reactions', 'order_sets', 'override_reason', 'override_comment', 'related_actions',
'age_in_years', 'age_range', 'sex'])
# Reduction of variables for exploration purposes.
df_pre_h1_short = pd.DataFrame(df_pre_h1).reset_index()
df_pre_s = df_pre_h1_short[['description', 'warning_status', 'source', 'severity', 'provider_type', 'provider_specialty', 'interaction_setting', 'patient_hospital_clinic',
'phases_of_care', 'context', 'drug_allergy_contraindication_group', 'drug_allergy_reason', 'drug_allergy_class_pair_for_cross_sensitivity_warning',
'drug_allergy_reactions', 'order_sets', 'override_reason', 'override_comment', 'related_actions', 'age_in_years', 'age_range', 'sex']]
df_pre_s.head()


## 6. Crosstab analysis
# Crosstab of warning status by 'sex' and patient 'age_in_years'.
df_pre_s_pat = pd.crosstab([df_pre_s.sex, df_pre_s.warning_status], df_pre_s.age_in_years.round(2), margins = True)
df_pre_s_pat = df_pre_s_pat.reset_index()
df_pre_s_pat

df_pre_s_pat.shape

# Crosstab of warning status and context.
df_pre_s_context = pd.crosstab(df_pre_s.warning_status, df_pre_s.context, margins=True)
df_pre_s_context = df_pre_s_context.reset_index()
df_pre_s_context


# Crosstab of warning status and provider_type.
df_pre_s_provider_type = pd.crosstab(df_pre_s.warning_status, df_pre_s.provider_type, margins=True)
df_pre_s_provider_type = df_pre_s_provider_type.reset_index(
df_pre_s_provider_type


# Crosstab of warning status and provider_specialty.
df_pre_s_provider_specialty = pd.crosstab(df_pre_s.warning_status, df_pre_s.provider_specialty, margins=True)
df_pre_s_provider_specialty = df_pre_s_provider_specialty.reset_index()
df_pre_s_provider_specialty


# Overview of crosstab of warning status and override reason.
df_pre_s_override_reason = pd.crosstab(df_pre_s.warning_status, df_pre_s.override_reason, margins=True)
df_pre_s_override_reason = df_pre_s_override_reason.reset_index()
df_pre_s_override_reason
```

6.1. Crosstab analysis - description

```
# Warning status versus context - inpatient or outpatient, by description.
df_pre_s_cr_interdes = pd.crosstab([df_pre_s.warning_status, df_pre_s.context], df_pre_s.description, margins=True)
df_pre_s_cr_interdes = df_pre_s_cr_interdes.reset_index()
df_pre_s_cr_interdes
```

```
df_pre_s_cr_interdes.shape
```

6.2. Crosstab analysis - source

```
# Warning status versus context - inpatient or outpatient, by source.
df_pre_s_cr_intersour = pd.crosstab([df_pre_s.warning_status, df_pre_s.context], df_pre_s.source, margins=True)
df_pre_s_cr_intersour = df_pre_s_cr_intersour.reset_index()
df_pre_s_cr_intersour
```

```
df_pre_s_cr_intersour.shape
```

6.3. Crosstab analysis - severity

```
# Warning status versus context - inpatient or outpatient, by severity.
df_pre_s_cr_intersever = pd.crosstab([df_pre_s.warning_status, df_pre_s.context], df_pre_s.severity, margins=True)
df_pre_s_cr_intersever = df_pre_s_cr_intersever.reset_index()
df_pre_s_cr_intersever
```

```
df_pre_s_cr_intersever.shape
```

6.4. Crosstab analysis - provider type

```
# Warning status versus context - inpatient or outpatient, by provider type.
df_pre_s_cr_proty = pd.crosstab([df_pre_s.warning_status, df_pre_s.context], df_pre_s.provider_type, margins=True)
df_pre_s_cr_proty = df_pre_s_cr_proty.reset_index()
df_pre_s_cr_proty
```

```
df_pre_s_cr_proty.shape
```

6.5. Crosstab analysis - provider speciality

```
# Warning status versus context - inpatient or outpatient, by provider speciality.
df_pre_s_cr_prosp = pd.crosstab([df_pre_s.warning_status, df_pre_s.context], df_pre_s.provider_specialty, margins=True)
df_pre_s_cr_prosp = df_pre_s_cr_prosp.reset_index()
df_pre_s_cr_prosp
```

```
df_pre_s_cr_prosp.shape
```

6.6. Crosstab analysis - patient hospital/clinic

```
# Warning status versus context - inpatient or outpatient, by patient_hospital/clinic.
df_pre_s_cr_interphc = pd.crosstab([df_pre_s.warning_status, df_pre_s.context], df_pre_s.patient_hospital_clinic, margins=True)
df_pre_s_cr_interphc = df_pre_s_cr_interphc.reset_index()
df_pre_s_cr_interphc
```

```
df_pre_s_cr_interphc.shape
```

6.7. Crosstab analysis - phases of care

```
# Warning status versus context - inpatient or outpatient, by phases of care.
df_pre_s_cr_interpoc = pd.crosstab([df_pre_s.warning_status, df_pre_s.context], df_pre_s.phases_of_care, margins=True)
df_pre_s_cr_interpoc = df_pre_s_cr_interpoc.reset_index()
df_pre_s_cr_interpoc
```

```
df_pre_s_cr_interpoc.shape
```

6.8. Crosstab analysis - drug-allergy contraindication group

```
# Warning status versus context - inpatient or outpatient, by drug-allergy_contraindication_group.
df_pre_s_cr_interdacg = pd.crosstab([df_pre_s.warning_status, df_pre_s.context], df_pre_s.drug_allergy_contraindication_group, margins=True)
df_pre_s_cr_interdacg = df_pre_s_cr_interdacg.reset_index()
df_pre_s_cr_interdacg
```

```
df_pre_s_cr_interdacg.shape
```

6.9. Crosstab analysis - drug-allergy reason

```
# Warning status versus context - inpatient or outpatient, by drug_allergy_reason.
df_pre_s_cr_interdar = pd.crosstab([df_pre_s.warning_status, df_pre_s.context], df_pre_s.drug_allergy_reason, margins=True)
df_pre_s_cr_interdar = df_pre_s_cr_interdar.reset_index()
df_pre_s_cr_interdar
```

```
df_pre_s_cr_interdar.shape
```

6.10. Crosstab analysis - drug-allergy class pair for cross-sensitivity warning

```
# Warning status versus context - inpatient or outpatient, by drug_allergy_class_pair_for_cross_sensitivity_warning.
```

```
df_pre_s_cr_interdacp = pd.crosstab([df_pre_s.warning_status, df_pre_s.context], df_pre_s.drug_allergy_class_pair_for_cross_sensitivity_warning,
margins=True)
df_pre_s_cr_interdacp = df_pre_s_cr_interdacp.reset_index()
df_pre_s_cr_interdacp

df_pre_s_cr_interdacp.shape
```

6.11. Crosstab analysis - interaction setting

```
# Warning status versus context - inpatient or outpatient, by interaction setting.
df_pre_s_cr_interorst = pd.crosstab([df_pre_s.warning_status, df_pre_s.context], df_pre_s.order_sets, margins=True)
df_pre_s_cr_interorst = df_pre_s_cr_interorst.reset_index()
df_pre_s_cr_interorst

df_pre_s_cr_interorst.shape
```

7. Override analysis

```
# Overrides have the most override reasons of the five warning status alerts.
# Override analysis for exploration purposes - isolate all override warning alert rows for analysis.
df_pre_o1 = pd.DataFrame(df_pre_s, columns = ['description', 'warning_status', 'provider_type', 'provider_specialty', 'interaction_setting',
'patient_hospital_clinic', 'phases_of_care', 'context', 'override_reason', 'override_comment', 'related_actions', 'drug_allergy_reactions', 'age_in_years', 'age_range',
'sex'])
df_pre_o2 = df_pre_o1.loc[df_pre_o1['warning_status'].astype(str) == 'Overridden']
df_pre_o2 = df_pre_o1[df_pre_o1['warning_status'].map(lambda warning_status: 'Overridden' in warning_status)]
df_pre_o2.head()

df_pre_o2.shape

# Count of override reason for overridden alerts.
df_pre_o2_c1 = Counter(df_pre_o2['override_reason'])
df_pre_o2_c2 = pd.DataFrame.from_dict(df_pre_o2_c1, orient='index').reset_index()
df_pre_o2_c2.columns = ['override_reason', 'count']
df_pre_o2_c3 = df_pre_o2_c2.sort_values(by = ['count'], ascending = False)
# Include percentage column
df_pre_o2_c3['%'] = ((df_pre_o2_c3['count'] / df_pre_o2_c3['count'].sum())*100).round(2).astype(str) + '%'
df_pre_o2_c3

# Count of override reason for overridden alerts - missing data removed.
df_pre_o2_c1a = Counter(df_pre_o2['override_reason'])
df_pre_o2_c2a = pd.DataFrame.from_dict(df_pre_o2_c1a, orient='index').reset_index()
df_pre_o2_c2a.columns = ['override_reason', 'count']
df_pre_o2_c3a = df_pre_o2_c2a.sort_values(by = ['count'], ascending = False)
df_pre_o2_c3a1 = df_pre_o2_c3a.replace(-999, np.nan)
df_pre_o2_c3a2 = df_pre_o2_c3a1.dropna()
pd.set_option('display.max_colwidth', None)
# Include percentage column
df_pre_o2_c3a2['%'] = ((df_pre_o2_c3a2['count'] / df_pre_o2_c3a2['count'].sum())*100).round(2).astype(str) + '%'
df_pre_o2_c3a2

# Description versus override reason by description
df_pre_oa2 = pd.crosstab(df_pre_o2.override_reason, df_pre_o2.description, margins=True)
df_pre_oa2 = df_pre_oa2.reset_index()
df_pre_oa2

# Description versus override reason by provider_type
df_pre_oa3 = pd.crosstab(df_pre_o2.override_reason, df_pre_o2.provider_type, margins=True)
df_pre_oa3 = df_pre_oa3.reset_index()
df_pre_oa3

# Description versus override reason by provider specialty
df_pre_oa4 = pd.crosstab(df_pre_o2.override_reason, df_pre_o2.provider_specialty, margins=True)
df_pre_oa4 = df_pre_oa4.reset_index()
df_pre_oa4

# Description versus override reason by interaction setting.
df_pre_oa5 = pd.crosstab(df_pre_o2.override_reason, df_pre_o2.interaction_setting, margins=True)
df_pre_oa5 = df_pre_oa5.reset_index()
df_pre_oa5

# Description versus override reason by patient_hospital_clinic.
df_pre_oa6 = pd.crosstab(df_pre_o2.override_reason, df_pre_o2.patient_hospital_clinic, margins=True)
df_pre_oa6 = df_pre_oa6.reset_index()
df_pre_oa6

# Description versus override reason by phases of care.
df_pre_oa7 = pd.crosstab(df_pre_o2.override_reason, df_pre_o2.phases_of_care, margins=True)
df_pre_oa7 = df_pre_oa7.reset_index()
df_pre_oa7

# Description versus override reason by context.
df_pre_oa8 = pd.crosstab(df_pre_o2.override_reason, df_pre_o2.context, margins=True)
df_pre_oa8 = df_pre_oa8.reset_index()
df_pre_oa8

# Description versus override reason by override_comment.
```

```
df_pre_oa9 = pd.crosstab(df_pre_o2.override_reason, df_pre_o2.override_comment, margins=True)
df_pre_oa9 = df_pre_oa9.reset_index()
df_pre_oa9

# Description versus override reason by related_actions.
df_pre_oa10 = pd.crosstab(df_pre_o2.override_reason, df_pre_o2.related_actions, margins=True)
df_pre_oa10 = df_pre_oa10.reset_index()
df_pre_oa10

# Description versus override reason by age_range.
df_pre_oa11 = pd.crosstab(df_pre_o2.override_reason, df_pre_o2.age_range, margins=True)
df_pre_oa11 = df_pre_oa11.reset_index()
df_pre_oa11

# Description versus override reason by sex.
df_pre_oa12 = pd.crosstab(df_pre_o2.override_reason, df_pre_o2.sex, margins=True)
df_pre_oa12 = df_pre_oa12.reset_index()
df_pre_oa12
```

8. Removed analysis

```
# Removed analysis for exploration purposes.
df_pre_r1 = pd.DataFrame(df_pre_s, columns = ['description', 'warning_status', 'provider_type', 'provider_specialty', 'interaction_setting',
'patient_hospital_clinic', 'phases_of_care', 'context', 'override_reason', 'override_comment', 'related_actions', 'drug_allergy_reactions', 'age_in_years', 'age_range',
'sex'])
df_pre_o2 = df_pre_o1.loc[df_pre_o1['warning_status'].astype(str) == 'Overridden']
df_pre_r2 = df_pre_r1[df_pre_r1['warning_status'].map(lambda warning_status: 'Removed' in warning_status)]
df_pre_r2.head()

df_pre_r2.shape

# Count of override reason for removed alerts
df_pre_r2_c1 = Counter(df_pre_r2['override_reason'])
df_pre_r2_c2 = pd.DataFrame.from_dict(df_pre_r2_c1, orient='index').reset_index()
df_pre_r2_c2.columns = ['override_reason', 'count']
df_pre_r2_c3 = df_pre_r2_c2.sort_values(by = ['count'], ascending = False)
# Include percentage column
df_pre_r2_c3['%'] = ((df_pre_r2_c3['count'] / df_pre_r2_c3['count'].sum())*100).round(2).astype(str) + '%'
df_pre_r2_c3

# Count of override reason for removed alerts - missing data removed.
df_pre_r2_c1a = Counter(df_pre_r2['override_reason'])
df_pre_r2_c2a = pd.DataFrame.from_dict(df_pre_r2_c1a, orient='index').reset_index()
df_pre_r2_c2a.columns = ['override_reason', 'count']
df_pre_r2_c3a = df_pre_r2_c2a.sort_values(by = ['count'], ascending = False)
df_pre_r2_c3a1 = df_pre_r2_c3a.replace(-999, np.nan)
df_pre_r2_c3a2 = df_pre_r2_c3a1.dropna()
pd.set_option('display.max_colwidth', None)
# Include percentage column
df_pre_r2_c3a2['%'] = ((df_pre_r2_c3a2['count'] / df_pre_r2_c3a2['count'].sum())*100).round(2).astype(str) + '%'
df_pre_r2_c3a2

# Removed reason by description
df_pre_ra2 = pd.crosstab(df_pre_r2.override_reason, df_pre_r2.description, margins=True)
df_pre_ra2 = df_pre_ra2.reset_index()
df_pre_ra2

# Removed reason by provider_type
df_pre_ra3 = pd.crosstab(df_pre_r2.override_reason, df_pre_r2.provider_type, margins=True)
df_pre_ra3 = df_pre_ra3.reset_index()
df_pre_ra3

# Removed reason by provider specialty
df_pre_ra4 = pd.crosstab(df_pre_r2.override_reason, df_pre_r2.provider_specialty, margins=True)
df_pre_ra4 = df_pre_ra4.reset_index()
df_pre_ra4

# Removed reason by interaction setting.
df_pre_ra5 = pd.crosstab(df_pre_r2.override_reason, df_pre_r2.interaction_setting, margins=True)
df_pre_ra5 = df_pre_ra5.reset_index()
df_pre_ra5

# Removed reason by patient_hospital_clinic.
df_pre_ra6 = pd.crosstab(df_pre_r2.override_reason, df_pre_r2.patient_hospital_clinic, margins=True)
df_pre_ra6 = df_pre_ra6.reset_index()
df_pre_ra6

# Removed reason by phases of care.
df_pre_ra7 = pd.crosstab(df_pre_r2.override_reason, df_pre_r2.phases_of_care, margins=True)
df_pre_ra7 = df_pre_ra7.reset_index()
df_pre_ra7

# Removed reason by context.
df_pre_ra8 = pd.crosstab(df_pre_r2.override_reason, df_pre_r2.context, margins=True)
df_pre_ra8 = df_pre_oa8.reset_index()
```

```
df_pre_ra8

# Removed reason by override_comment.
df_pre_ra9 = pd.crosstab(df_pre_r2.override_reason, df_pre_r2.override_comment, margins=True)
df_pre_ra9 = df_pre_ra9.reset_index()
df_pre_ra9

# Removed reason by related_actions.
df_pre_ra10 = pd.crosstab(df_pre_r2.override_reason, df_pre_r2.related_actions, margins=True)
df_pre_ra10 = df_pre_ra10.reset_index()
df_pre_ra10

# Removed reason by age_in_years.
df_pre_ra11 = pd.crosstab(df_pre_r2.override_reason, df_pre_r2.age_in_years, margins=True)
df_pre_ra11 = df_pre_ra11.reset_index()
df_pre_ra11

# Removed reason by sex.
df_pre_ra12 = pd.crosstab(df_pre_r2.override_reason, df_pre_r2.sex, margins=True)
df_pre_ra12 = df_pre_ra12.reset_index()
df_pre_ra12
```

9. Cancelled analysis

```
# Cancelled analysis for exploration purposes.
df_pre_c1 = pd.DataFrame(df_pre_s, columns = ['description', 'warning_status', 'provider_type', 'provider_specialty', 'interaction_setting',
'patient_hospital_clinic', 'phases_of_care', 'context', 'override_reason', 'override_comment', 'related_actions', 'drug_allergy_reactions', 'age_in_years', 'age_range',
'sex'])
df_pre_o2 = df_pre_o1.loc[df_pre_o1['warning_status'].astype(str) == 'Overridden']
df_pre_c2 = df_pre_c1[df_pre_c1['warning_status'].map(lambda warning_status: 'Cancelled' in warning_status)]
df_pre_c2.head()

df_pre_c2.shape

# Count of cancelled reason for removed alerts.
df_pre_c2_c1 = Counter(df_pre_c2['override_reason'])
df_pre_c2_c2 = pd.DataFrame.from_dict(df_pre_c2_c1, orient='index').reset_index()
df_pre_c2_c2.columns = ['override_reason', 'count']
df_pre_c2_c3 = df_pre_c2_c2.sort_values(by = ['count'], ascending = False)
# Include percentage column
df_pre_c2_c3['%'] = ((df_pre_c2_c3['count'] / df_pre_c2_c3['count'].sum())*100).round(2).astype(str) + '%'
df_pre_c2_c3
```

10. Counts

10.1. Count by description

```
df_pre_s

# Description count
df_1_1 = pd.DataFrame({'description_count': df_pre_s['description']})
df_1_2 = df_1_1.loc[df_1_1['description_count'].isin(df_pre_s['description']),'description_count'].value_counts()
df_1_3 = pd.DataFrame(df_1_2)
df_1_3 = df_1_3.reset_index()
df_1_3


# Description count
df_2_1 = pd.DataFrame({'warning_count': df_pre_s['warning_status']})
df_2_2 = df_2_1.loc[df_2_1['warning_count'].isin(df_pre_s['warning_status']),'warning_count'].value_counts()
df_2_3 = pd.DataFrame(df_2_2)
df_2_3 = df_2_3.reset_index()
# Include percentage column
df_2_3['%'] = ((df_2_3['warning_count'] / df_2_3['warning_count'].sum())*100).round(2).astype(str) + '%'
#df_2_3.to_csv('description_no_per.csv')
df_2_3
```

10.2 Removed merging and counts

```
# Drug-Allergy reaction count relating to 'removed' alert warnings.
df_3_1 = pd.DataFrame({'da_reaction_count': df_pre_r2['drug_allergy_reactions']})
df_3_2 = df_3_1.loc[df_3_1['da_reaction_count'].isin(df_pre_r2['drug_allergy_reactions']),'da_reaction_count'].value_counts()
df_3_3 = pd.DataFrame(df_3_2)
df_3_4 = df_3_3.reset_index()
df_3_5 = df_3_4.replace(-999, np.nan)
df_3_6 = df_3_5.dropna()
pd.set_option('display.max_colwidth', None)
# Include percentage column
df_3_6['%'] = ((df_3_6['da_reaction_count'] / df_3_6['da_reaction_count'].sum())*100).round(2).astype(str) + '%'
#df_3_6.to_csv('DA_reaction_r_1.csv')

# Total drug-Allergy reaction count.
df_4_1 = pd.DataFrame({'da_reaction_count': df_pre_s['drug_allergy_reactions']})
df_4_2 = df_4_1.loc[df_4_1['da_reaction_count'].isin(df_pre_s['drug_allergy_reactions']),'da_reaction_count'].value_counts()
df_4_3 = pd.DataFrame(df_4_2)
df_4_4 = df_4_3.reset_index()
```

```
df_4_5 = df_4_4.replace(-999, np.nan)
df_4_6 = df_4_5.dropna()
pd.set_option('display.max_colwidth', None)
# Include percentage column
df_4_6['%'] = ((df_4_6['da_reaction_count'] / df_4_6['da_reaction_count'].sum())*100).round(2).astype(str) + '%'
#df_4_6.to_csv('DA_reaction_r_2.csv')

#  Merging total count and 'Removed' count tabulations.
df_5_m = pd.merge(df_4_6, df_3_6, on = 'index', how='left', suffixes=('_total', '_removed'))
# Missing data 'NaN' changed to '1' in order to complete calclation.
df_5_mr = df_5_m.replace(np.nan, 1)
# Converting column values to floats.
df_5_mr['da_reaction_count_total'] = df_5_mr['da_reaction_count_total'].astype(int)
df_5_mr['da_reaction_count_removed'] = df_5_mr['da_reaction_count_removed'].astype(int)
# Creating a new column named 'da_reaction_count_removed'.
# Treat 'NaN's divided by 'da_reaction_count_total' as the original value in 'da_reaction_count_total'.
def removed_rate_(da_reaction_count_removed, da_reaction_count_total):
    return da_reaction_count_removed / da_reaction_count_total
df_5_mr['removed_rate_%'] = (removed_rate_(df_5_mr['da_reaction_count_removed'], df_5_mr['da_reaction_count_total']) *100).round(2)
#df_5_mr.to_csv('DA_reaction_r_3.csv')
df_5_mr

# Drug-Allergy reaction count relating to 'removed' alert warnings - includes missing data.
df_3_1m = pd.DataFrame({'da_reaction_count': df_pre_r2['drug_allergy_reactions']})
df_3_2m = df_3_1m.loc[df_3_1m['da_reaction_count'].isin(df_pre_r2['drug_allergy_reactions']),'da_reaction_count'].value_counts()
df_3_3m = pd.DataFrame(df_3_2m)
df_3_4m = df_3_3m.reset_index()
# Total drug-Allergy reaction count.
df_4_1m = pd.DataFrame({'da_reaction_count': df_pre_s['drug_allergy_reactions']})
df_4_2m = df_4_1m.loc[df_4_1m['da_reaction_count'].isin(df_pre_s['drug_allergy_reactions']),'da_reaction_count'].value_counts()
df_4_3m = pd.DataFrame(df_4_2m)
df_4_4m = df_4_3m.reset_index()
# Include percentage column
df_4_4m['%'] = ((df_4_4m['da_reaction_count'] / df_4_4m['da_reaction_count'].sum())*100).round(2).astype(str) + '%'
df_3_4m['%'] = ((df_3_4m['da_reaction_count'] / df_4_4m['da_reaction_count'].sum())*100).round(2).astype(str) + '%'
#df_3_6m.to_csv('DA_reaction_r_1m.csv')
#df_4_6m.to_csv('DA_reaction_r_2m.csv')

df_3_4m.shape

df_4_4m.shape

#  Merging total count and 'Removed' count tabulations.
df_5_mm = pd.merge(df_4_4m, df_3_4m, on = 'index', how='left', suffixes=('_total', '_removed'))
# Converting column values to floats.
df_5_mm['da_reaction_count_total'] = df_5_mm['da_reaction_count_total'].astype(int)
df_5_mm['da_reaction_count_removed'] = df_5_mm['da_reaction_count_removed'].astype(int)
# Creating a new column named 'da_reaction_count_removed'.
def removed_rate_(da_reaction_count_removed, da_reaction_count_total):
    return da_reaction_count_removed / da_reaction_count_total
df_5_mm['removed_rate_%'] = (removed_rate_(df_5_mm['da_reaction_count_removed'], df_5_mm['da_reaction_count_total']) *100).round(2)
#df_5_mm.to_csv('DA_reaction_r_3m.csv')
df_5_mm

df_5_mm['da_reaction_count_total'].sum()
```

10.3 Overridden merging and counts

```
# Drug-Allergy reaction count relating to 'overridden' alert warnings.
df_6_1 = pd.DataFrame({'da_reaction_count': df_pre_o2['drug_allergy_reactions']})
df_6_2 = df_6_1.loc[df_6_1['da_reaction_count'].isin(df_pre_o2['drug_allergy_reactions']),'da_reaction_count'].value_counts()
df_6_3 = pd.DataFrame(df_6_2)
df_6_4 = df_6_3.reset_index()
df_6_5 = df_6_4.replace(-999, np.nan)
df_6_6 = df_6_5.dropna()
pd.set_option('display.max_colwidth', None)
# Include percentage column
df_6_6['%'] = ((df_6_6['da_reaction_count'] / df_6_6['da_reaction_count'].sum())*100).round(2).astype(str) + '%'

# Total drug-Allergy reaction count.
df_7_1 = pd.DataFrame({'da_reaction_count': df_pre_s['drug_allergy_reactions']})
df_7_2 = df_7_1.loc[df_7_1['da_reaction_count'].isin(df_pre_s['drug_allergy_reactions']),'da_reaction_count'].value_counts()
df_7_3 = pd.DataFrame(df_7_2)
df_7_4 = df_7_3.reset_index()
df_7_5 = df_7_4.replace(-999, np.nan)
df_7_6 = df_7_5.dropna()
pd.set_option('display.max_colwidth', None)
# Include percentage column
df_7_6['%'] = ((df_7_6['da_reaction_count'] / df_7_6['da_reaction_count'].sum())*100).round(2).astype(str) + '%'
df_6_6['%'] = ((df_6_6['da_reaction_count'] / df_7_6['da_reaction_count'].sum())*100).round(2).astype(str) + '%'
#df_6_6.to_csv('DA_reaction_o_1.csv')
#df_7_6.to_csv('DA_reaction_o_2.csv')

#  Merging total count and 'Overridden' count tabulations.
df_8_m = pd.merge(df_7_6, df_6_6, on = 'index', how='left', suffixes=('_total', '_overridden'))
```

```
# Missing data 'NaN' changed to '1' in order to complete calclation.
df_8_mo = df_8_m.replace(np.nan, 1)
# Converting column values to floats.
df_8_mo['da_reaction_count_total'] = df_8_mo['da_reaction_count_total'].astype(int)
df_8_mo['da_reaction_count_overridden'] = df_8_mo['da_reaction_count_overridden'].astype(int)
# Creating a new column named 'da_reaction_count_overridden'.
# Treat 'NaN's divided by 'da_reaction_count_total' as the original value in 'da_reaction_count_total'.
def overridden_rate_(da_reaction_count_overridden, da_reaction_count_total):
    return da_reaction_count_overridden / da_reaction_count_total
df_8_mo['overridden_rate_%'] = (overridden_rate_(df_8_mo['da_reaction_count_overridden'], df_8_mo['da_reaction_count_total']) *100).round(2)
#df_8_mo.to_csv('DA_reaction_o_3.csv')
df_8_mo

#  Drug-Allergy reaction count relating to 'overridden' alert warnings - includes missing data.
df_6_1m = pd.DataFrame({'da_reaction_count': df_pre_o2['drug_allergy_reactions']})
df_6_2m = df_6_1m.loc[df_6_1m['da_reaction_count'].isin(df_pre_o2['drug_allergy_reactions']),'da_reaction_count'].value_counts()
df_6_3m = pd.DataFrame(df_6_2m)
df_6_4m = df_6_3m.reset_index()
# Total drug-Allergy reaction count.
df_7_1m = pd.DataFrame({'da_reaction_count': df_pre_s['drug_allergy_reactions']})
df_7_2m = df_7_1m.loc[df_7_1m['da_reaction_count'].isin(df_pre_s['drug_allergy_reactions']),'da_reaction_count'].value_counts()
df_7_3m = pd.DataFrame(df_7_2m)
df_7_4m = df_7_3m.reset_index()
# Include percentage column
df_7_4m['%'] = ((df_7_4m['da_reaction_count'] / df_7_4m['da_reaction_count'].sum())*100).round(2).astype(str) + '%'
df_6_4m['%'] = ((df_6_4m['da_reaction_count'] / df_7_4m['da_reaction_count'].sum())*100).round(2).astype(str) + '%'
#df_6_4m.to_csv('DA_reaction_o_1m.csv')
#df_7_4m.to_csv('DA_reaction_o_2m.csv')

df_6_4m.shape

df_7_4m.shape

#  Merging total count and 'Overridden' count tabulations.
df_8_mm = pd.merge(df_7_4m, df_6_4m, on = 'index', how='left', suffixes=('_total', '_overridden'))
# Creating a new column named 'da_reaction_count_overridden'.
def overridden_rate_(da_reaction_count_overridden, da_reaction_count_total):
    return da_reaction_count_overridden / da_reaction_count_total
df_8_mm['overridden_rate_%'] = (overridden_rate_(df_8_mm['da_reaction_count_overridden'], df_8_mm['da_reaction_count_total']) *100).round(2)
#df_8_mm.to_csv('DA_reaction_o_3m.csv')
df_8_mm

df_8_mm['da_reaction_count_total'].sum()
```

10.4 Provider type, provider specialty and description dimensionality reduction and counts

```
#  Provider type and description dimensionality reduction and counts.
df_a = df_pre_s[['provider_type', 'provider_specialty', 'description', 'context', 'warning_status', 'drug_allergy_reactions']]
df_a = pd.DataFrame(df_a)
df_a['provider_type_cat1'] = df_a['provider_type']
df_a['provider_type_cat2'] = df_a['provider_type']
df_a['provider_specialty_cat1'] = df_a['provider_specialty']
df_a['provider_specialty_cat2'] = df_a['provider_specialty']
df_a['description_cat1'] = df_a['description']
df_a['description_cat2'] = df_a['description']
df_a['drug_allergy_reactions'] = df_a['drug_allergy_reactions']
df_a.head(50)
```

Provider type cat 1 and cat 2 (subset)

```
#  Provider type high level dimensionality reduction (category 1) to medical_prescriber and non_medical_prescriber.
df_a_c1 = df_a.provider_type_cat1.replace(['Consultant Anaesthetist', 'Registrar - ST1-2', 'Registrar - ST3+', 'Registrar Anaesthetist - ST3+', 'Foundation Year Doctor', 'Consultant Doctor', 'Associate Specialty Grade Doctor', 'Specialty Grade Doctor', 'Registrar Surgeon ST1-2', 'Foundation Year 2 Doctor', 'Registrar Anaesthetist - ST1-2', 'Specialty Grade Anaesthetist', 'Registrar Surgeon ST3+', 'Associate Specialty Grade Surgeon', 'Foundation Year Surgeon', 'Doctor', 'Associate Specialty Grade Anaesthetist', 'General Practitioner', 'Consultant Surgeon', 'Registrar Pathologist ST3+', 'Foundation  Year 2 Surgeon', 'Consultant Psychiatrist', 'Specialty Grade Surgeon', 'Consultant Radiologist', 'Dental Registrar - ST1-2', 'Specialty Grade Dentist', 'Consultant Dentist', 'Registrar Radiologist ST3+', 'Consultant Dental Surgeon', 'Dental Registrar - ST3+', 'Registrar Dental Surgeon - ST3+', 'External Doctor', 'Dental Student', 'Specialty Grade Dental Surgeon'], 'medical_prescriber')
df_a_c1 = pd.DataFrame(df_a_c1)
df_a_ca1 = df_a_c1.provider_type_cat1.replace(['AHP Scientist', 'Pharmacist', 'Staff Nurse', 'Technician', 'Pharmacist Independent Prescriber', 'Clinical Nurse Specialist', 'Nurse Practitioner', 'Charge Nurse', 'Senior Staff Nurse', 'AHP Scientist', 'Anaesthesia Associate', 'Nurse Prescriber', 'AHP Consultant', 'Midwife', 'Pre-Registration Pharmacist', 'AHP Manager', 'Physician Associate', 'Radiographer (Diagnostics)', 'Theatre Practitioner', 'Dietitian', 'Advanced Care Practitioner', 'HCA', 'Art, Music or Drama Therapist', 'Research Nurse', 'Nurse Manager', 'Pre-Registration Nurse', 'Nursing Student', 'Assistant', 'Modern Matron', 'Physiotherapist', 'Practitioner', 'Clinical Trials Practitioner', 'Midwife Lead', 'Nursing Assistant', 'Nurse Consultant', 'Nurse Anaesthetist', 'Dental Therapy Student', 'Radiographer (Therapeutics)', 'Nursing Assistant Practitioner', 'Midwife Matron', 'Scientist', 'Science Consultant', 'Instructor'], 'non_medical_prescriber')
df_a_ca1 = pd.DataFrame(df_a_ca1)
df_a_cat1 = pd.DataFrame(df_a_ca1)
df_a_cat1

#  Count by provider type (cat 1)
df_pt_c1 = df_a_cat1.groupby(['provider_type_cat1']).size().reset_index(name = 'count')
df_pt_c_1 = df_pt_c1.replace(-999, np.nan)
df_pt_ca1 = df_pt_c_1.dropna()
pd.set_option('display.max_colwidth', None)
```

```
# Include percentage column
df_pt_c1['%'] = ((df_pt_c1['count'] / df_pt_c1['count'].sum())*100).round(2).astype(str) + '%'
df_pt_c1

#  Provider type subset dimensionality reduction (category 2).
df_a_c2 = df_a.provider_type_cat2.replace(['AHP Scientist', 'Anaesthesia Associate', 'AHP Consultant', 'AHP Manager', 'Physician Associate', 'Radiographer
(Diagnostics)', 'Theatre Practitioner', 'Dietitian', 'Advanced Care Practitioner', 'HCA', 'Art, Music or Drama Therapist', 'Assistant', 'Physiotherapist', 'Practitioner',
'Clinical Trials Practitioner', 'Nursing Assistant', 'Dental Therapy Student', 'Radiographer (Therapeutics)', 'Nursing Assistant Practitioner'
], 'allied_health_professional')
df_a_c2 = pd.DataFrame(df_a_c2)
df_a_ca2 = df_a_c2.provider_type_cat2.replace(['Consultant Anaesthetist', 'Consultant Doctor', 'Consultant Surgeon', 'Consultant Psychiatrist', 'Consultant
Dentist'], 'consultant')
df_a_ca2 = pd.DataFrame(df_a_ca2)
df_a_ca2_1 = df_a_ca2.provider_type_cat2.replace(['Foundation Year Doctor', 'Foundation Year 2 Doctor', 'Foundation Year Surgeon', 'Foundation  Year 2
Surgeon', 'Dental Student'], 'junior_doctor')
df_a_ca2_1 = pd.DataFrame(df_a_ca2_1)
df_a_ca2_2 = df_a_ca2_1.provider_type_cat2.replace(['Doctor', 'General Practitioner', 'External Doctor'], 'other_doctor')
df_a_ca2_2 = pd.DataFrame(df_a_ca2_2)
df_a_cat2_3 = df_a_ca2_2.provider_type_cat2.replace(['Pharmacist', 'Pharmacist Independent Prescriber'], 'pharmacist')
df_a_cat2_3 = pd.DataFrame(df_a_cat2_3)
df_a_cat2_4 = df_a_cat2_3.provider_type_cat2.replace(['Pre-Registration Nurse', 'Nursing Student'], 'pre_registration_nurse')
df_a_cat2_4 = pd.DataFrame(df_a_cat2_4)
df_a_cat2_5 = df_a_cat2_4.provider_type_cat2.replace(['Staff Nurse', 'Clinical Nurse Specialist', 'Nurse Practitioner', 'Charge Nurse', 'Senior Staff Nurse', 'Nurse
Prescriber', 'Midwife', 'Research Nurse', 'Nurse Manager', 'Modern Matron', 'Midwife Lead', 'Nurse Consultant', 'Nurse Anaesthetist', 'Midwife Matron'],
'registered_nurse')
df_a_cat2_5 = pd.DataFrame(df_a_cat2_5)
df_a_cat2_6 = df_a_cat2_5.provider_type_cat2.replace(['Registrar - ST1-2', 'Registrar - ST3+', 'Registrar Anaesthetist - ST3+', 'Associate Specialty Grade
Doctor', 'Specialty Grade Doctor', 'Registrar Surgeon ST1-2', 'Registrar Anaesthetist - ST1-2', 'Specialty Grade Anaesthetist', 'Registrar Surgeon ST3+',
'Associate Specialty Grade Surgeon', 'Associate Specialty Grade Anaesthetist', 'Registrar Pathologist ST3+', 'Specialty Grade Surgeon', 'Consultant Radiologist',
'Dental Registrar - ST1-2', 'Specialty Grade Dentist', 'Registrar Radiologist ST3+', 'Dental Registrar - ST3+', 'Registrar Dental Surgeon - ST3+', 'Specialty Grade
Dental Surgeon'], 'registrar')
df_a_cat2_6 = pd.DataFrame(df_a_cat2_6)
df_a_cat2_7 = df_a_cat2_6.provider_type_cat2.replace(['Scientist', 'Science Consultant', 'Instructor'], 'other')
df_a_cat2 = pd.DataFrame(df_a_cat2_7)
df_a_cat2

#  Count by provider type (cat 2)
df_pt_c2 = df_a_cat2.groupby(['provider_type_cat2']).size().reset_index(name = 'count')

# Include percentage column
df_pt_c2['%'] = ((df_pt_c2['count'] / df_pt_c2['count'].sum())*100).round(2).astype(str) + '%'
#df_pt_c2.to_csv('df_pt_c2_150821.csv')
df_pt_c2

df_a_cat2.shape

#  Combined provider type cat 1 and cat 2.
df_pt_comcat = pd.concat([df_a_cat1, df_a_cat2], axis=1)
df_pt_comcat
```

Provider speciality cat 1 and cat 2 (subset)

```
#  Provider speciality high level dimensionality reduction (category 1).
df_b_c1 = df_a.provider_specialty_cat1.replace(['Anes', 'Anes, ICU', 'EM', 'ICU', 'ICU, Cardio', 'Radiology, ICU', 'ICU, Radiology'], 'anes_&_emergency_med')
df_b_c1 = pd.DataFrame(df_b_c1)
df_b_ca1 = df_b_c1.provider_specialty_cat1.replace(['OPD', 'OPD, OPD', 'OPD, GI, Radiology', 'IM', 'Clin Pharm', 'Endo', 'Allergy', 'IM, GI', 'Geri Med,
Neuro', 'GI', 'Uro', 'Geri Med', 'ID', 'ENT', 'Rheum', 'SLT', 'Respiratory', 'Psychiatry', 'Diag Rad', 'MedMicro', 'NeuroPhys', 'Radiology', 'RadThera', 'Haem,
Neuro, Neuro, Neuro, Diag Rad, IM, AVM, Neuro, Haem, Uro, Uro, Haem, Neuro Surg, Neuro Surg, Haem, EM, Haem', 'Derm', 'Neuro, Haem', 'Cardio',
'MedVir', 'Uro, Radiology', 'Diag Rad, Radiology', 'Derm, Plastic Surg, GI', 'GUM', 'OMPath', 'ENT, Neuro Surg', 'Radiology, GI', 'Path', 'Ophth', 'AVM', 'Anat
Path', 'Haem', 'Haem, Paeds Onc', 'Haem, Oncology', 'Neuro', 'Neuro, GI'], 'gen_medicine_&_other_med_spec')
df_b_ca1 = pd.DataFrame(df_b_ca1)
df_b_ca1_1 = df_b_ca1.provider_specialty_cat1.replace(['Neuro Surg', 'Orthopedics', 'Cardiac Surg', 'Vasc Surg', 'Oral Surgery', 'Dent', 'ResDen', 'Periodontics',
'ResDen, Endodontics, Periodontics, Prosthodont', 'SCDent', 'ResDen, Endodontics', 'Endodontics', 'Orthodontics', 'PaedDent', 'Gen Surg', 'Gen Surg, Radiology,
OPD', 'Neuro Surg, Neuro', 'Neuro Surg, GI'], 'gen_surg_&_other_surgical_specialty')
df_b_ca1_1 = pd.DataFrame(df_b_ca1_1)
df_b_ca1_2 = df_b_ca1_1.provider_specialty_cat1.replace(['GP'], 'general_practice')
df_b_ca1_2 = pd.DataFrame(df_b_ca1_2)
df_b_ca1_3 = df_b_ca1_2.provider_specialty_cat1.replace(['Paeds', 'Neonatology', 'Paeds Endo'], 'neonatology_paediatrics')
df_b_ca1_3 = pd.DataFrame(df_b_ca1_3)
df_b_ca1_4 = df_b_ca1_3.provider_specialty_cat1.replace(['OBGYN', 'Mat' ], 'obstetrics_gynaecology')
df_b_ca1_4 = pd.DataFrame(df_b_ca1_4)
df_b_ca1_5 = df_b_ca1_4.provider_specialty_cat1.replace(['Oncology', 'Med Onc', 'Paeds Onc', 'Clin Onc', 'Med Onc, Clin Onc', 'Med Onc, Oncology',
'Oncology, Med Onc', 'Oncology, Clin Onc', 'GI, Oncology, Gen Surg', 'Radiology, Oncology', 'Palliative'], 'oncology_&_palliative_care')
df_b_ca1_5 = pd.DataFrame(df_b_ca1_5)
df_b_ca1_6 = df_b_ca1_5.provider_specialty_cat1.replace(['Diet', 'Rehab', 'PT'], 'other_non_medical_specialty')
df_b_ca1_6 = pd.DataFrame(df_b_ca1_6)
df_b_cat1 = df_b_ca1_6.provider_specialty_cat1.replace(['Pharmacy', 'Pharmacy, Haem'], 'pharmacy')
df_b_cat1 = pd.DataFrame(df_b_cat1)
df_b_cat1

#  Count by provider speciality (cat 1)
df_ps_c1 = df_b_cat1.groupby(['provider_specialty_cat1']).size().reset_index(name = 'count')
pd.set_option('display.max_colwidth', None)
# Include percentage column
df_ps_c1['%'] = ((df_ps_c1['count'] / df_ps_c1['count'].sum())*100).round(2).astype(str) + '%'
```

```
#df_ps_c1.to_csv('df_ps_c1_150821.csv')
df_ps_c1

#  Provider specialty subset dimensionality reduction (category 2).
df_b_c2 = df_a.provider_specialty_cat2.replace(['Anes', 'Anes, ICU'], 'anaesthetics')
df_b_c2 = pd.DataFrame(df_b_c2)
df_b_ca2_1 = df_b_c2.provider_specialty_cat2.replace(['EM', 'ICU', 'ICU, Cardio', 'Radiology, ICU', 'ICU, Radiology'], 'emergency_medicine')
df_b_ca2_1 = pd.DataFrame(df_b_ca2_1)
df_b_ca_2_2 = df_b_ca2_1.provider_specialty_cat2.replace(['IM', 'Clin Pharm', 'Endo', 'Allergy', 'IM, GI', 'Geri Med, Neuro'], 'general_medicine')
df_b_ca_2_2 = pd.DataFrame(df_b_ca_2_2)
df_b_ca_2_3 = df_b_ca_2_2.provider_specialty_cat2.replace(['GP'], 'general_practice')
df_b_ca_2_3 = pd.DataFrame(df_b_ca_2_3)
df_b_cat_2_4 = df_b_ca_2_3.provider_specialty_cat2.replace(['Haem', 'Haem, Paeds Onc', 'Haem, Oncology'], 'haematology')
df_b_cat_2_4 = pd.DataFrame(df_b_cat_2_4)
df_b_cat_2_5 = df_b_cat_2_4.provider_specialty_cat2.replace(['Paeds', 'Neonatology', 'Paeds Endo'], 'neonatology_paediatrics')
df_b_cat_2_5 = pd.DataFrame(df_b_cat_2_5)
df_b_cat_2_6 = df_b_cat_2_5.provider_specialty_cat2.replace(['Neuro', 'Neuro, GI', 'Neuro Surg, Neuro', 'Neuro Surg, GI'], 'neurology')
df_b_cat_2_6 = pd.DataFrame(df_b_cat_2_6)
df_b_cat_2_7 = df_b_cat_2_6.provider_specialty_cat2.replace(['OBGYN', 'Mat'], 'obstetrics_gynaecology')
df_b_cat_2_7 = pd.DataFrame(df_b_cat_2_7)
df_b_cat_2_8 = df_b_cat_2_7.provider_specialty_cat2.replace(['Oncology', 'Med Onc', 'Paeds Onc', 'Clin Onc', 'Med Onc, Clin Onc', 'Med Onc, Oncology',
'Oncology, Med Onc', 'Oncology, Clin Onc', 'GI, Oncology, Gen Surg', 'Radiology, Oncology'], 'oncology')
df_b_cat_2_8 = pd.DataFrame(df_b_cat_2_8)
df_b_cat_2_9 = df_b_cat_2_8.provider_specialty_cat2.replace(['GI', 'Uro', 'Geri Med', 'ID', 'ENT', 'Rheum', 'SLT', 'Respiratory', 'Psychiatry', 'Diag Rad',
'MedMicro', 'NeuroPhys', 'Radiology', 'RadThera', 'Haem, Neuro, Neuro, Neuro, Diag Rad, IM, AVM, Neuro, Haem, Uro, Uro, Haem, Neuro Surg, Neuro Surg,
Haem, EM, Haem', 'Derm', 'Neuro, Haem', 'Cardio', 'MedVir', 'Uro, Radiology', 'Diag Rad, Radiology', 'Derm, Plastic Surg, GI', 'GUM', 'OMPath', 'ENT, Neuro
Surg', 'Radiology, GI', 'Path', 'Ophth', 'AVM', 'Diet', 'Anat Path'], 'other_medical_specialty')
df_b_cat_2_9 = pd.DataFrame(df_b_cat_2_9)
df_b_cat_2_10 = df_b_cat_2_9.provider_specialty_cat2.replace(['Diet', 'Rehab', 'PT'], 'other_non_medical_specialty')
df_b_cat_2_10 = pd.DataFrame(df_b_cat_2_10)
df_b_cat_2_11 = df_b_cat_2_10.provider_specialty_cat2.replace(['Neuro Surg', 'Orthopedics', 'Cardiac Surg', 'Vasc Surg', 'Oral Surgery', 'Dent', 'ResDen',
'Periodontics', 'ResDen, Endodontics, Periodontics, Prosthodont', 'SCDent', 'ResDen, Endodontics', 'Endodontics', 'Orthodontics', 'PaedDent'],
'other_surgical_specialty')
df_b_cat_2_11 = pd.DataFrame(df_b_cat_2_11)
df_b_cat_2_12 = df_b_cat_2_11.provider_specialty_cat2.replace(['OPD', 'OPD, OPD', 'OPD, GI, Radiology'], 'outpatient_department')
df_b_cat_2_12 = pd.DataFrame(df_b_cat_2_12)
df_b_cat_2_13 = df_b_cat_2_12.provider_specialty_cat2.replace(['Palliative'], 'palliative_care')
df_b_cat_2_13 = pd.DataFrame(df_b_cat_2_13)
df_b_cat_2_14 = df_b_cat_2_13.provider_specialty_cat2.replace(['Pharmacy', 'Pharmacy, Haem'], 'pharmacy')
df_b_cat_2_14 = pd.DataFrame(df_b_cat_2_14)
df_b_cat_2_15 = df_b_cat_2_14.provider_specialty_cat2.replace(['Gen Surg', 'Gen Surg, Radiology, OPD'], 'surgery')
df_b_cat_2_15 = pd.DataFrame(df_b_cat_2_15)
df_b_cat_2_16 = df_b_cat_2_15.provider_specialty_cat2.replace(['Scientist', 'Science Consultant', 'Instructor'], 'other')
df_b_cat2 = pd.DataFrame(df_b_cat_2_16)
df_b_cat2

#  Count by provider speciality (cat 2)
df_ps_c2 = df_b_cat2.groupby(['provider_specialty_cat2']).size().reset_index(name = 'count')
pd.set_option('display.max_colwidth', None)

# Include percentage column
df_ps_c2['%'] = ((df_ps_c2['count'] / df_ps_c2['count'].sum())*100).round(2).astype(str) + '%'
df_ps_c2


#  Combined provider speciality cat 1 and cat 2.
df_ps_comcat = pd.concat([df_b_cat1, df_b_cat2], axis=1)
df_ps_comcat
```

10.5 D-A reaction dimensionality reduction and associated counts, percentages and rates

```
#  D_A reaction dimensionality reduction.
df_e_da1 = df_pre_s.replace(['Anaphylaxis'], 'anaphylaxis')
df_e_da1 = pd.DataFrame(df_e_da1)
df_e_da1_1 = df_e_da1.replace(['Anaphylaxis, Swelling', 'Anaphylaxis, Other (see comments)', 'Anaphylaxis, Rash, itching or hives, Shortness of breath,
Swelling', 'Anaphylaxis, Rash, itching or hives', 'Anaphylaxis, Rash, itching or hives, Swelling'], 'anaphylaxis_&_other')
df_e_da1_1 = pd.DataFrame(df_e_da1_1)
df_e_da1_2 = df_e_da1_1.replace(['Diarrhoea', 'Gastrointestinal bleeding', 'Diarrhoea, Rash, itching or hives, Swelling', 'Rash, itching or hives, Diarrhoea',
'Diarrhoea, Other (see comments)', 'Other (see comments), Gastrointestinal bleeding', 'Rash, itching or hives, Gastrointestinal bleeding', 'GI intolerance'], 'gi
intolerance_&_other')
df_e_da1_2 = pd.DataFrame(df_e_da1_2)
df_e_da1_3 = df_e_da1_2.replace(['Other (see comments)'], 'other')
df_e_da1_3 = pd.DataFrame(df_e_da1_3)
df_e_da1_4 = df_e_da1_3.replace(['Rash, itching or hives', 'Rash, itching or hives, Swelling', 'Rash, itching or hives, Other (see comments)', 'Rash', 'Other (see
comments), Rash, itching or hives', 'Rash, itching or hives, Diarrhoea, Other (see comments)'], 'rash, itching or hives _&_other')
df_e_da1_4 = pd.DataFrame(df_e_da1_4)
df_e_da1_5 = df_e_da1_4.replace(['Shortness of breath'], 'shortness of breath')
df_e_da1_5 = pd.DataFrame(df_e_da1_5)
df_e_da1_6 = df_e_da1_5.replace(['Rash, itching or hives, Shortness of breath', 'Shortness of breath, Other (see comments)', 'Shortness of breath, Swelling',
'Rash, itching or hives, Shortness of breath, Swelling', 'Swelling, Shortness of breath'], 'shortness of breath_&_other')
df_e_da1_6 = pd.DataFrame(df_e_da1_6)
df_e_da1_7 = df_e_da1_6.replace(['Swelling', 'Swelling, Other (see comments)', 'Swelling, Rash, itching or hives'], 'swelling_&_other')
df_e_da1_7 = pd.DataFrame(df_e_da_7)
df_e_da1_7.head(50)
```

```
df_e_da1_7.drug_allergy_reactions.unique()

#  Count by DA reactions (Cat 2)
df_e_da2 = df_e_da1_7.groupby(['drug_allergy_reactions']).size().reset_index(name = 'count')
pd.set_option('display.max_colwidth', None)
# Include percentage column
df_e_da2['%'] = ((df_e_da2['count'] / df_e_da2['count'].sum())*100).round(2).astype(str) + '%'
#df_e_da2.to_csv('df_e_da2_150821.csv')
df_e_da2.head(50)

df_e_da2.drug_allergy_reactions.unique()
```

Description cat 1 and cat 2 (subset)

```
#  Provider description high level dimensionality reduction (category 1).
df_c_ca1 = df_a.description_cat1.replace(['DROPERIDOL', 'CYCLIZINE', 'METOCLOPRAMIDE'], 'anti-emetic')
df_c_ca1 = pd.DataFrame(df_c_ca1)
df_c_ca1_1 = df_c_ca1.description_cat1.replace(['PENICILLINS', 'CEFPODOXIME', 'CHLORAMPHENICOL', 'CHLORAMPHENICOL WITH
HYDROCORTISONE', 'MACROLIDES', 'MUPIROCIN', 'QUINOLONE ANTIBIOTICS', 'SULFADIAZINE', 'CO-TRIMOXAZOLE (TRIMETHOPRIM
AND SULFAMETHOXAZOLE)', 'PIPERACILLIN WITH TAZOBACTAM', 'CO-AMOXICLAV (AMOXICILLIN AND CLAVULANIC ACID)',
'AMOXICILLIN', 'MEROPENEM', 'FLUCLOXACILLIN', 'CLARITHROMYCIN', 'CICLOSPORIN', 'ERYTHROMYCIN', 'CLAVULANIC ACID',
'VANCOMYCIN', 'CEFTAZIDIME', 'CEPHALOSPORINS', 'CEFALEXIN', 'SULFONAMIDES', 'PHENOXYMETHYLPENICILLIN', 'SULFUR AND
SULFONAMIDES', 'CEFTRIAXONE', 'CEFUROXIME', 'BENZYLPENICILLIN', 'NITROFURANTOIN', 'TAZOBACTAM', 'PIPERACILLIN',
'CEFOTAXIME', 'PIVMECILLINAM', 'CIPROFLOXACIN', 'GENTAMICIN', 'DOXYCYCLINE', 'TRIMETHOPRIM', 'CEFAZOLIN', 'CEFRADINE',
'SULFAMETHOXAZOLE', 'CEFTAZIDIME AND AVIBACTAM', 'METRONIDAZOLE', 'RIFAMPICIN', 'OXYTETRACYCLINE', 'TETRACYCLINE',
'AMPICILLIN', 'NORFLOXACIN', 'GENTAMICIN WITH HYDROCORTISONE', 'NEOMYCIN', 'AMINOGLYCOSIDES', 'ISONIAZID', 'CEFACLOR',
'CLINDAMYCIN', 'TETRACYCLINES', 'CEFADROXIL', 'CO-FLUAMPICIL (FLUCLOXACILLIN AND AMPICILLIN)', 'PIVAMPICILLIN',
'NITROFURAN DERIVATIVES'], 'antibiotic')
df_c_ca1_1 = pd.DataFrame(df_c_ca1_1)
df_c_ca1_2 = df_c_ca1_1.description_cat1.replace(['CHLORPHENAMINE', 'CETIRIZINE', 'ANTIHISTAMINES (PHENOTHIAZINE)'], 'antihistamine')
df_c_ca1_2 = pd.DataFrame(df_c_ca1_2)
df_c_ca1_3 = df_c_ca1_2.description_cat1.replace(['CORTICOSTEROIDS'], 'corticosteroid')
df_c_ca1_3 = pd.DataFrame(df_c_ca1_3)
df_c_ca1_4 = df_c_ca1_3.description_cat1.replace(['NORMAL IMMUNOGLOBULIN HUMAN', 'IMMUNOGLOBULINS', 'ANTI-D (RHO)
IMMUNOGLOBULIN'], 'immunoglobulin')
df_c_ca1_4 = pd.DataFrame(df_c_ca1_4)
df_c_ca1_5 = df_c_ca1_4.description_cat1.replace(['ASPIRIN', 'IBUPROFEN', 'NSAIDS','NSAID', 'PARECOXIB', 'DICLOFENAC', 'NAPROXEN',
'PARACETAMOL', 'CO-PROXAMOL (DEXTROPROPOXYPHENE AND PARACETAMOL)', 'SALICYLATES', 'DICLOFENAC SODIUM AND
MISOPROSTOL', 'MEFENAMIC ACID', 'SALICYLIC ACID', 'PARACETAMOL WITH CAFFEINE', 'KETOPROFEN', 'SALICYLIC ACID WITH
MUCOPOLYSACCHARIDE POLYSULFATE', 'IBUPROFEN WITH PHENYLEPHRINE HYDROCHLORIDE', 'ASPIRIN WITH CAFFEINE'], 'NSAID')
df_c_ca1_5 = pd.DataFrame(df_c_ca1_5)
df_c_ca1_6 = df_c_ca1_5.description_cat1.replace(['MORPHINE', 'CODEINE', 'TRAMADOL', 'CO-CODAMOL (CODEINE AND PARACETAMOL)',
'PETHIDINE', 'DIAMORPHINE', 'DIHYDROCODEINE', 'BUPRENORPHINE', 'OXYCODONE', 'FENTANYL', 'OPIOIDS-MORPHINE ANALOGUES
(OXYCODONE,BUPRENORPHINE,CODEINE)', 'CO-DYDRAMOL (DIHYDROCODEINE AND PARACETAMOL)', 'OPIOIDS-FENTANYL AND
PETHIDINE ANALOGUES', 'PAPAVERETUM WITH PAPAVERINE AND CODEINE AND MORPHINE', 'OPIOIDS-METHADONE AND DIPIPANONE
ANALOGUES', 'CODEINE PHOSPHATE WITH PROMETHAZINE HYDROCHLORIDE', 'TRAMADOL WITH PARACETAMOL', 'IBUPROFEN AND
CODEINE', 'OXYCODONE HYDROCHLORIDE WITH NALOXONE HYDROCHLORIDE', 'CO-CODAPRIN (CODEINE AND ASPIRIN)', 'CODEINE
PHOSPHATE WITH ASPIRIN AND CAFFEINE', 'MEPTAZINOL', 'MORPHINE HYDROCHLORIDE LIGHT KAOLIN, BELLADONNA AND
ALUMINIUM HYDROXIDE'], 'opioid')
df_c_ca1_6 = pd.DataFrame(df_c_ca1_6)
df_c_ca1_7 = df_c_ca1_6.description_cat1.replace(['BETAMETHASONE', 'BETAMETHASONE VALERATE WITH CLIOQUINOL',
'DEXAMETHASONE', 'PREDNISOLONE', 'FLUDROCORTISONE', 'HYDROCORTISONE', 'CORTISONE', 'DEXAMETHASONE WITH NEOMYCIN
AND POLYMYXIN', 'METHYLPREDNISOLONE', 'BECLOMETASONE', 'DEXAMETHASONE AND NEOMYCIN AND ACETIC ACID',
'TRIAMCINOLONE', 'DEXAMETHASONE WITH FRAMYCETIN AND GRAMICIDIN', 'BETAMETHASONE VALERATE AND FUSIDIC ACID',
'FLUTICASONE FUROATE AND UMECLIDINIUM AND VILANTEROL', 'PREDNISONE', 'ALCLOMETASONE', 'SODIUM FUSIDATE WITH
HYDROCORTISONE', 'CLOBETASONE BUTYRATE WITH OXYTETRACYCLINE AND NYSTATIN', 'FLUMETASONE PIVALATE WITH
CLIOQUINOL', 'FLUOCINOLONE ACETONIDE WITH CLIOQUINOL', 'HYDROCORTISONE BUTYRATE WITH CHLORQUINALDOL',
'HYDROCORTISONE WITH CLIOQUINOL', 'HYDROCORTISONE WITH MICONAZOLE'], 'steroid')
df_c_ca1_7 = pd.DataFrame(df_c_ca1_7)
df_c_cat1 = df_c_ca1_7.description_cat1.replace(['SIMVASTATIN', 'PEANUTS AND PEANUT OIL', 'AMLODIPINE', 'ATENOLOL', 'ATORVASTATIN',
'FERRIC CARBOXYMALTOSE', 'LANSOPRAZOLE', 'BUMETANIDE', 'OMEPRAZOLE', 'BENDROFLUMETHIAZIDE', 'GABAPENTIN', 'HEPARIN',
'BETA ADRENOCEPTOR ANTAGONISTS', 'AZATHIOPRINE', 'DALTEPARIN', 'SULFASALAZINE', 'IODINE', 'HMG-COA REDUCTASE
INHIBITORS (STATINS)', 'ENOXAPARIN', 'PAPAVERETUM', 'LOW MOLECULAR WEIGHT HEPARIN', 'FUROSEMIDE', 'ITRACONAZOLE',
'MESALAZINE', 'BISOPROLOL', 'CARBAMAZEPINE', 'ONDANSETRON', 'IODINATED CONTRAST MEDIA', 'POSACONAZOLE', 'MIDODRINE',
'FELODIPINE', 'TINZAPARIN', 'THALIDOMIDE', 'CARBOPLATIN', 'INDAPAMIDE', 'PREGABALIN', 'RIVAROXABAN', 'FEXOFENADINE',
'HYOSCINE', 'COLASPASE', 'RITUXIMAB', 'IOHEXOL', 'VORICONAZOLE', 'INSULIN, HUMAN', 'ROSUVASTATIN', 'DONEPEZIL',
'CLONAZEPAM', 'FLUTICASONE PROPIONATE WITH SALMETEROL', 'PACLITAXEL', 'SALBUTAMOL', 'LORAZEPAM', 'CITALOPRAM',
'TETRACAINE', 'ACICLOVIR', 'METHOTREXATE', 'DEFERASIROX', 'DIAZEPAM', 'NIFEDIPIN', 'CO-CARELDOPA (CARBIDOPA AND
LEVODOPA)', 'BUPIVACAINE', 'ACE INHIBITORS', 'BECLOMETASONE EXTRAFINE PARTICLE WITH FORMOTEROL', 'ANGIOTENSIN-II
RECEPTOR ANTAGONISTS', 'OLANZAPINE', 'VINCRISTINE', 'INSULIN - PROTAMINE ZINC - BOVINE', 'TIOTROPIUM', 'TEMAZEPAM',
'DESFERRIOXAMINE', 'PROCHLORPERAZINE', 'FLUTICASONE', 'NICOTINE', 'AMITRIPTYLINE', 'LEVODOPA WITH CARBIDOPA AND
ENTACAPONE', 'THIORIDAZINE', 'BEE VENOM', 'LIDOCAINE', 'PROPRANOLOL', 'PROMETHAZINE', 'SERTRALINE', 'TAMSULOSIN', 'FERROUS
SULFATE', 'SULPIRIDE', 'DAPSONE', 'RISPERIDONE', 'AZITHROMYCIN', 'IMIPENEM', 'VENLAFAXINE', 'PANTOPRAZOLE', 'CARBIMAZOLE',
'FLUCONAZOLE', 'PEGASPARGASE', 'RANITIDINE', 'TAPENTADOL', 'IPRATROPIUM', 'SITAGLIPTIN', 'ADRENALINE', 'DOMPERIDONE', 'IRON',
'CELECOXIB', 'INFLUENZA VIRUS VACCINE', 'OLMESARTAN MEDOXOMIL WITH AMLODIPINE AND HYDROCHLOROTHIAZIDE',
'CIPROFLOXACIN WITH DEXAMETHASONE', 'IRON ISOMALTOSIDE 1000', 'GADOTERIDOL', 'SIROLIMUS', 'LEVOTHYROXINE SODIUM',
'SPIRONOLACTONE', 'IMIPRAMINE', 'CLOPIDOGREL', 'HALOPERIDOL', 'METFORMIN', 'ISOSORBIDE MONONITRATE', 'OXYBUTYNIN',
'ARIPIPRAZOLE', 'METHADONE', 'BICALUTAMIDE"MAGNESIUM SULFATE AND OTHER MAGNESIUM SALTS', 'HEPARIN DERIVATIVES',
'COCONUT OIL AND COCONUT OIL', 'ENALAPRIL', 'LEVETIRACETAM', 'FERROUS FUMARATE', 'METOPROLOL', 'RAMIPRIL', 'IODINE
ANTISEPTICS AND DISINFECTANTS', 'PLASMA - HUMAN FROZEN SOLVENT/DETERGENT TREATED', 'ETOPOSIDE', 'KETOTIFEN',
'ZONISAMIDE', 'DORZOLAMIDE HYDROCHLORIDE WITH TIMOLOL MALEATE', 'LISINOPRIL', 'LORATADINE', 'LEVOMEPROMAZINE',
'ESOMEPRAZOLE', 'IFOSFAMIDE', 'TITANIUM', 'DACARBAZINE', 'AMISULPRIDE', 'ACETYLCYSTEINE', 'TEICOPLANIN', 'APIXABAN',
'PRAVASTATIN', 'BUDESONIDE WITH FORMOTEROL', 'PANCREATIN (LIPASE, AMYLASE, PROTEASE)', 'DIPHENHYDRAMINE',
'ALENDRONIC ACID', 'NORETHISTERONE', 'LOPERAMIDE', 'CARVEDILOL', 'LAMOTRIGINE', 'CARBOCISTEINE', 'ABACAVIR WITH
LAMIVUDINE', 'PHENYLEPHRINE', 'HYDROCHLOROTHIAZIDE', 'HYOSCINE BUTYLBROMIDE', 'GADOLINIUM SALTS AND COMPOUNDS',
```

```
'LENALIDOMIDE', 'ALLOPURINOL', 'AMPHOTERICIN', 'AMIODARONE', 'ALFENTANIL', 'PIZOTIFEN', 'PENICILLAMINE (NOT AN ANTIBIOTIC)',
'ZOLMITRIPTAN', 'GLYCERYL TRINITRATE', 'ETORICOXIB', 'INDOMETACIN', 'IRBESARTAN', 'UMECLIDINIUM WITH VILANTEROL',
'TERBUTALINE', 'GEMCITABINE', 'DEFERIPRONE', 'LENOGRASTIM', 'GLYCOPYRRONIUM', 'INSULIN GLARGINE', 'HYDROCORTISONE WITH
NYSTATIN DIMETICONE AND BENZALKONIUM CHLORIDE', 'HYDROXYCHLOROQUINE', 'CLOZAPINE', 'LOSARTAN', 'METHYL
SALICYLATE WITH MENTHOL', 'LIOTHYRONINE', 'DILTIAZEM', 'MIDAZOLAM', 'FLUOXETINE', 'ZOLEDRONIC ACID', 'MIRTAZAPINE',
'CLOBETASOL', 'CHLORPROMAZINE', 'IODIXANOL', 'CINNAMON AND OTHER INGREDIENTS', 'IDEBENONE', 'DILTIAZEM AND
HYDROCHLOROTHIAZIDE', 'VALACICLOVIR', 'TRIHEXYPHENIDYL', 'FOSFOMYCIN', 'TOPIRAMATE', 'GLATIRAMER', 'BUDESONIDE',
'PROPYLTHIOURACIL', 'LOSARTAN WITH HYDROCHLOROTHIAZIDE', 'ANTIHISTAMINES (ALKYLAMINE )', 'ANAKINRA', 'CALCIUM
CHANNEL BLOCKERS (DIHYDROPYRIDINE)', 'CANDESARTAN', 'OXALIPLATIN', 'WASP VENOM', 'ROFECOXIB', 'ROCURONIUM',
'CINNARIZINE', 'IRON DEXTRAN', 'IRON SUCROSE', 'KETAMINE', 'LETROZOLE', 'BEZAFIBRATE', 'LIDOCAINE WITH CHLORHEXIDINE',
'OXCARBAZEPINE', 'AMINOPHYLLINE', 'SOTALOL', 'TRASTUZUMAB', 'ESTRIOL', 'GELATIN', 'EMTRICITABINE WITH TENOFOVIR
DISOPROXIL', 'EDOXABAN', 'FONDAPARINUX', 'MERCAPTOPURINE', 'LOCAL ANAESTHETICS (PABA -DERIVED ESTER-TYPE )',
'ETHINYLESTRADIOL WITH LEVONORGESTREL', 'NABUMETONE', 'ETANERCEPT', 'CALCIPOTRIOL WITH BETAMETHASONE
DIPROPIONATE', 'CARBAMAZEPINE AND RELATED PRODUCTS', 'HAEMOPHILUS TYPE B AND MENINGOCOCCAL GROUP C CONJUGATE
VACCINE', 'DEXKETOPROFEN', 'NALOXONE', 'DOCETAXEL', 'DOXAZOSIN', 'FENBUFEN', 'SILDENAFIL', 'SALMETEROL', 'SUMATRIPTAN',
'TRANDOLAPRIL', 'VALPROIC ACID', 'UMECLIDINIUM', 'TRIFLUOPERAZINE', 'THYROID HORMONES', 'THEOPHYLLINE', 'PICOSULFATE',
'AMIDOTRIZOIC (DIATRIZOIC) ACID', 'MELOXICAM', 'ATROPINE', 'BETA ADRENERGIC AGONISTS', 'LACTULOSE', 'PIROXICAM',
'APREPITANT', 'MOMETASONE', 'PROTON PUMP INHIBITORS', 'QUADRIVALENT INFLUENZA VACCINE (SANOFI PASTEUR/FLUARIX
TETRA) - GENTAMICIN', 'QUADRIVALENT INFLUENZA VACCINE (SANOFI PASTEUR/FLUARIX TETRA) - STREPTOMYCIN', 'RASBURICASE',
'ALGINIC ACID', 'ALUMINIUM HYDROXIDE', 'NICOTINIC ACID', 'TESTOSTERONE', 'FLUVASTATINESTRADIOL', 'DIGOXIN', 'FLUNARIZINE',
'ETHINYLESTRADIOL WITH DROSPIRENONE', 'CLOBAZAM', 'GLICLAZIDE', 'ESLICARBAZEPINE', 'SOMATROPIN', 'TRANEXAMIC ACID',
'TIBOLONE', 'TACROLIMUS', 'SOLIFENACIN', 'SACUBITRIL WITH VALSARTAN', 'RIVASTIGMINE', 'QUETIAPINE','PYRIMETHAMINE WITH
SULFADOXINE', 'PYRIMETHAMINE', 'UROKINASE', 'VALDECOXIB', 'VITAMIN B12', 'VITAMIN C WITH VITAMINS AND MINERALS',
'ZOPICLONE', 'WARFARIN', 'PROGUANIL AND ATOVAQUONE', 'ATRACURIUM', 'LIDOCAINE HYDROCHLORIDE WITH ADRENALINE',
'CABERGOLINE', 'ISOTRETINOIN''ISOSORBIDE DINITRATE', 'ISAVUCONAZOLE', 'CARBOPROST TROMETAMOL', 'CHLORHEXIDINE',
'INSULIN ASPART''INFLIXIMAB', 'CLOMIPRAMINE', 'COLESTYRAMINE', 'CROMOGLICATE', 'DESOGESTREL', 'FUSIDIC ACID', 'FOLINIC ACID',
'DINOPROSTONE', 'FESOTERODINE', 'DOLUTEGRAVIR WITH ABACAVIR AND LAMIVUDINE', 'DOSULEPIN', 'BENZOYL PEROXIDE',
'MEBEVERINE', 'PERINDOPRIL', 'PAROXETINE', 'OSELTAMIVIR', 'NATALIZUMAB', 'ANTIPSYCHOTIC (PHENOTHIAZINES)',
'ACETAZOLAMIDE', 'ACLIDINIUM', 'ADENOSINE', 'ALIMEMAZINE', 'AMILORIDE', 'ASCORBIC ACID WITH OTHER VITAMINS',
'BENZODIAZEPINES', 'BICALUTAMIDE', 'BISPHOSPHONATES', 'BOTULINUM TOXIN TYPE A', 'CANAGLIFLOZIN', 'CHLORHEXIDINE
GLUCONATE WITH AMETHOCAINE HYDROCHLORIDE', 'CLOMIFENE', 'CYCLOPHOSPHAMIDE', 'DOBUTAMINE', 'DOCUSATE',
'DULOXETINE', 'EPHEDRINE', 'ESTRADIOL', 'ETODOLAC', 'EZETIMIBE WITH SIMVASTATIN', 'FINASTERIDE', 'FISH AND FISH OIL',
'FLUVASTATIN', 'GLIBENCLAMIDE', 'HYDROXYZINE', 'HYOSCINE HYDROBROMIDE', 'HYOSCYAMINE', 'INFLIXIMAB', 'INSULIN ASPART',
'INSULIN DETEMIR', 'ISOSORBIDE DINITRATE', 'ISOTRETINOIN', 'LACOSAMIDE', 'LEFLUNOMIDE', 'LIDOCAINE WITH PRILOCAINE',
'LINACLOTIDE', 'LOCAL ANAESTHETICS, AMIDE-TYPE', 'MACROGOL', 'MAGNESIUM SULFATE AND OTHER MAGNESIUM SALTS',
'METHENAMINE', 'MIRABEGRON', 'NIFEDIPINE', 'OPIOID ANTAGONISTS', 'OXETACAINE', 'OXYMETAZOLINE', 'PHENELZINE',
'PHENYLBUTAZONE', 'PIMECROLIMUS', 'PROGESTERONE', 'PROPOFOL', 'PSEUDOEPHEDRINE', 'PSEUDOEPHEDRINE WITH TRIPROLIDINE',
'REMIFENTANIL', 'ROPINIROLE', 'SENNA', 'SUGAMMADEX', 'SUXAMETHONIUM', 'TETRABENAZINE', 'THIAZIDES', 'TIMOLOL',
'VEDOLIZUMAB', 'VERAPAMIL', 'ZOLPIDEM'], 'other')
df_c_cat1 = pd.DataFrame(df_c_cat1)
df_c_cat1

#  Count by description (cat 1)
df_c_c2 = df_c_cat1.groupby(['description_cat1']).size().reset_index(name = 'count')
pd.set_option('display.max_colwidth', None)
# Include percentage column
df_c_c2['%'] = ((df_c_c2['count'] / df_c_c2['count'].sum())*100).round(2).astype(str) + '%'
#df_c_ca2.to_csv('df_c_ca2_150821.csv')
df_c_c2.head(50)

#  Description subset dimensionality reduction (category 2) - Penicillins.
df_d_cat2 = df_a.description_cat2.replace(['PENICILLINS', 'AMOXICILLIN', 'BENZYLPENICILLIN', 'PIVMECILLINAM', 'AMPICILLIN',
'PIVAMPICILLIN'], 'penicillin')
df_d_cat2 = pd.DataFrame(df_d_cat2)
df_d_cat2

#  Count by description (cat 1)
df_d_c2 = df_d_cat2.groupby(['description_cat2']).size().reset_index(name = 'count')
pd.set_option('display.max_colwidth', None)

# Include percentage column
df_d_c2['%'] = ((df_d_c2['count'] / df_d_c2['count'].sum())*100).round(2).astype(str) + '%'
#df_d_ca2.to_csv('df_d_ca2_test.csv')
df_d_c2

#  Combined description cat 1 and cat 2.
df_d_comcat = pd.concat([df_c_cat1, df_d_cat2], axis=1)
df_d_comcat

#### Combined dataframe
# Concat of dataframes for combined dataframe.
df_e_da1_7['sex'] = df_e_da1_7['sex'].astype(str)
df_comb = pd.concat([df_pt_comcat, df_ps_comcat, df_c_cat1, df_a['context'], df_a['warning_status'], df_pre_ager['age_range'], df_e_da1_7['sex']], axis=1)
# Convert dataframe to csv.
#df_comb.to_csv('df_comb_120821.csv')
df_comb

# ••••• Concat of dataframes for combined dataframe with D-A reactions •••••
df_e_da1_7['sex'] = df_e_da1_7['sex'].astype(str)
df_comb_ada = pd.concat([df_pt_comcat, df_ps_comcat, df_c_cat1, df_a['context'], df_a['warning_status'], df_e_da1_7['drug_allergy_reactions'],
df_e_da1_7['sex'], df_pre_ager['age_range'], df_pr['severity'], df_pr['importance_level'], df_pr['drug_allergy_contraindication_group']], axis=1)
#df_comb_ada.to_csv('df_comb_ada_180821.csv')
df_comb_ada.head(60)

df_comb_ada.drug_allergy_reactions.unique()
```

```
# Count of grouped D-A reactions.
df_comb_ada_a = df_comb_ada.groupby(['drug_allergy_reactions']).size().reset_index(name = 'count')
df_comb_ada_a

df_e_da1.shape

# Count of grouped D-A reactions.
df_comb_ada_a = df_comb_ada.groupby(['drug_allergy_reactions']).size().reset_index(name = 'count')
df_comb_ada_a

df_e_da1.shape

# Count of grouped severity.
df_comb_ada_b = df_comb_ada.groupby(['severity']).size().reset_index(name = 'count')
# Include percentage column
df_comb_ada_b['%'] = ((df_comb_ada_b['count'] / df_comb_ada_b['count'].sum())*100).round(2).astype(str) + '%'
df_comb_ada_b

df_comb_ada_b.shape

## Provider type cat2 by severity - counts and percentages.
df_comb_pt_ada_s = df_comb_ada['provider_type_cat2'].groupby([df_comb_ada['severity'], df_comb_ada['warning_status']]).count().unstack()
df_comb_pt_ada_s['row_count'] = df_comb_pt_ada_s['Overridden'] + df_comb_pt_ada_s['Removed']
# Include percentage column
df_comb_pt_ada_s['%_overridden'] = ((df_comb_pt_ada_s['Overridden'] / df_comb_pt_ada_s['row_count'])*100).round(2).astype(str) + '%'
df_comb_pt_ada_s['%_removed'] = ((df_comb_pt_ada_s['Removed'] / df_comb_pt_ada_s['row_count'])*100).round(2).astype(str) + '%'
# Convert dataframe to csv.
#df_comb_pt_ada_s.to_csv('df_comb_pt_ada_s_240821.csv')
df_comb_pt_ada_s

# Count of grouped importance_level.
df_comb_ada_c = df_comb_ada.groupby(['importance_level']).size().reset_index(name = 'count')
# Include percentage column
df_comb_ada_c['%'] = ((df_comb_ada_c['count'] / df_comb_ada_c['count'].sum())*100).round(2).astype(str) + '%'
df_comb_ada_c

df_comb_ada_c.shape

## Provider type cat2 by importance_level - counts and percentages.
df_comb_pt_ada_il = df_comb_ada['provider_type_cat2'].groupby([df_comb_ada['importance_level'], df_comb_ada['warning_status']]).count().unstack()
df_comb_pt_ada_il['row_count'] = df_comb_pt_ada_il['Overridden'] + df_comb_pt_ada_il['Removed']
# Include percentage column
df_comb_pt_ada_il['%_overridden'] = ((df_comb_pt_ada_il['Overridden'] / df_comb_pt_ada_il['row_count'])*100).round(2).astype(str) + '%'
df_comb_pt_ada_il['%_removed'] = ((df_comb_pt_ada_il['Removed'] / df_comb_pt_ada_il['row_count'])*100).round(2).astype(str) + '%'
# Convert dataframe to csv.
#df_comb_pt_ada_il.to_csv('df_comb_pt_ada_il_240821.csv')
df_comb_pt_ada_il

# Count of grouped drug_allergy_contraindication_group.
df_comb_ada_d = df_comb_ada.groupby(['drug_allergy_contraindication_group']).size().reset_index(name = 'count')
# Include percentage column
df_comb_ada_d['%'] = ((df_comb_ada_d['count'] / df_comb_ada_d['count'].sum())*100).round(2).astype(str) + '%'
df_comb_ada_d

df_comb_ada_d.shape

## Provider type cat2 by drug_allergy_contraindication_group - counts and percentages.
df_comb_pt_ada_dacg = df_comb_ada['provider_type_cat2'].groupby([df_comb_ada['drug_allergy_contraindication_group'],
df_comb_ada['warning_status']]).count().unstack()
df_comb_pt_ada_dacg['row_count'] = df_comb_pt_ada_dacg['Overridden'] + df_comb_pt_ada_dacg['Removed']
# Include percentage column
df_comb_pt_ada_dacg['%_overridden'] = ((df_comb_pt_ada_dacg['Overridden'] / df_comb_pt_ada_dacg['row_count'])*100).round(2).astype(str) + '%'
df_comb_pt_ada_dacg['%_removed'] = ((df_comb_pt_ada_dacg['Removed'] / df_comb_pt_ada_dacg['row_count'])*100).round(2).astype(str) + '%'
# Convert dataframe to csv.
#df_comb_pt_ada_dacg.to_csv('df_comb_pt_ada_dacg_240821.csv')
df_comb_pt_ada_dacg

# ***** Select overridden and removed only. *****

value_list = ['Overridden', 'Removed']
df_comb_ada_0 = df_comb_ada[df_comb_ada.warning_status.isin(value_list)]
df_comb_ada_0

# ***** IMPORTANT - Assign numerical codes to categories. *****
dict0 = {'male' : '1', 'female' : '0'}
#print(dict0)
dict1 = {'non_medical_prescriber' : '1', 'medical_prescriber' : '0'}
#print(dict1)
dict2 = {'pharmacist' : '0', 'registered_nurse' : '1', 'consultant' : '2', 'registrar' : '3',
      'allied_health_professional' : '4', 'junior_doctor' : '5', 'Technician' : '6',
      'Pre-Registration Pharmacist' : '7', 'Consultant Dental Surgeon' : '8', 'pre_registration_nurse' : '9',
      'other_doctor' : '10', 'other' : "11"}
#print(dict2)
dict3 = {'pharmacy' : '0', 'gen_medicine_&_other_med_spec' : '1',
```

```
       'anes_&_emergency_med' : '2', 'gen_surg_&_other_surgical_specialty' : '3',
       'general_practice' : '4', 'obstetrics_gynaecology' : '5',
       'oncology_&_palliative_care' : '6', 'neonatology_paediatrics' : '7',
       'other_non_medical_specialty' : '8'}
#print(dict3)
dict4 = {'pharmacy' : '0', 'neurology' : '1', 'anaesthetics' : '2',
       'other_surgical_specialty' : '3', 'emergency_medicine' : '4',
       'general_practice' : '5', 'general_medicine' : '6', 'obstetrics_gynaecology' : '7',
       'other_medical_specialty' : '8', 'oncology' : '9', 'haematology' : '10',
       'outpatient_department' : '11', 'surgery' : '12', 'neonatology_paediatrics' : '13',
       'palliative_care' : '14', 'other_non_medical_specialty' : '15'}
#print(dict4)
dict5 = {'antibiotic' : '0', 'NSAID' : '1', 'steroid' : '2', 'anti-emetic' : '3', 'corticosteroid' : '4',
     'immunoglobulin' : '5', 'opioid' : '6', 'antihistamine' : '7', 'other' : '8'}
#print(dict5)
dict6 = {'Inpatient' : '0', 'Outpatient' : '1', 'Both Inpatient and Outpatient' : '2'}
#print(dict6)
dict7 = {'Overridden' : '1', 'Removed' : '0'}
#print(dict7)
dict8 = {'anaphylaxis' : '0', 'anaphylaxis_&_other': '1',
      'gi intolerance_&_other' : '2', 'other' : '3',
      'rash, itching or hives _&_other' : '4', 'shortness of breath' : '5',
      'shortness of breath_&_other' : '6', 'swelling_&_other' : '7', '-999' : '8', 'nan' : '9'}
#print(dict8)
dict9 = {'46-55' : '5', '56-65' : '6', '16-25' : '2', '26-35' : '3', '36-45' : '4', '66-80' : '7', '80+' : '8',
     '6-15' : '1', '0-5' : '0', 'nan' : '-999'}
#print(dict9)
dict10 = {'Cross-sensitive Class Match' : '0', 'Drug Class Match' : '1', 'Ingredient Match' : '2', '-999' : '-999'}
#print(dict10)
dict11 = {'High' : '0', 'Very High' : '1', '-999' : '-999'}
#print(dict11)
dict12 = {'Adverse Reactions/Drug Intolerances' : '0', 'Allergies' : '1'}
#print(dict12)
# Remap the values of the dataframe
df_comb_ada_00 = df_comb_ada_0.replace({"sex": dict0})
df_comb_ada_01 = df_comb_ada_00.replace({"provider_type_cat1": dict1})
df_comb_ada_02 = df_comb_ada_01.replace({"provider_type_cat2": dict2})
df_comb_ada_03 = df_comb_ada_02.replace({"provider_specialty_cat1": dict3})
df_comb_ada_04 = df_comb_ada_03.replace({"provider_specialty_cat2": dict4})
df_comb_ada_05 = df_comb_ada_04.replace({"description_cat1": dict5})
df_comb_ada_06 = df_comb_ada_05.replace({"context": dict6})
df_comb_ada_07 = df_comb_ada_06.replace({"warning_status": dict7})
df_comb_ada_08 = df_comb_ada_07.replace({"drug_allergy_reactions": dict8})
df_comb_ada_09 = df_comb_ada_08.replace({"age_range": dict9})
df_comb_ada_10 = df_comb_ada_09.replace({"severity": dict10})
df_comb_ada_11 = df_comb_ada_10.replace({"importance_level": dict11})
df_comb_ada_12 = df_comb_ada_11.replace({"drug_allergy_contraindication_group": dict12})
# Convert dataframe to csv.
#df_comb_ada_12.to_csv('analysis_ready_dfv2_240821.csv')
df_comb_ada_12.head(50)

df_comb_ada_09a = df_comb_ada_12.groupby(['drug_allergy_reactions']).size().reset_index(name = 'count')
pd.set_option('display.max_colwidth', None)
# Include percentage column
df_comb_ada_09a['%'] = ((df_comb_ada_09a['count'] / df_comb_ada_09a['count'].sum())*100).round(2).astype(str) + '%'
df_comb_ada_09a

df_comb_ada_09.shape

# Sanity checking D-A reactions dataframe
df_comb_ada_12.drug_allergy_reactions.unique()


#### Filtered for non_medical_prescriber and medical_prescriber.

# Concat of dataframes for combined dataframe - non-medical-prescriber
v = ['non_medical_prescriber']
df_comb_nmp = df_comb_ada[df_comb_ada.provider_type_cat1.isin(v)]
df_comb_nmp

# Concat of dataframes for combined dataframe - medical-prescriber
v = ['medical_prescriber']
df_comb_mp = df_comb_ada[df_comb_ada.provider_type_cat1.isin(v)]
df_comb_mp

# ***** Concat of dataframes for combined dataframe of removed and overridden. *****
df_comb_ro = pd.concat([df_pt_comcat, df_ps_comcat, df_c_cat1, df_a['context'], df_a['warning_status'], df_pre_ager['age_range']], axis=1)
value_list = ['Overridden', 'Removed']
df_comb_ro = df_comb_ro[df_comb_ro.warning_status.isin(value_list)]
# Convert dataframe to csv.
#df_comb_ro.to_csv('df_comb_ro_120821.csv')
df_comb_ro

# # Filtered for removed and overridden.
# Concat of dataframes for combined dataframe - non-medical-prescriber
```

```
v = ['non_medical_prescriber']
df_comb_ro_nmp = df_comb_ro[df_comb_ro.provider_type_cat1.isin(v)]
df_comb_ro_nmp

# Filtered for removed and overridden.
# Concat of dataframes for combined dataframe - medical-prescriber
v = ['medical_prescriber']
df_comb_ro_mp = df_comb_ro[df_comb_ro.provider_type_cat1.isin(v)]
df_comb_ro_mp

## Dataframe analysis by provider type Cat 1 and Cat 2
## Provider type cat2 (df_comb_ada_0) - counts and percentages.
df_comb_pt_ada_0_cat2 = df_comb_ada_0['provider_type_cat2'].groupby([df_comb_ada_0['provider_type_cat2'],
df_comb_ada_0['warning_status']]).count().unstack()
df_comb_pt_ada_0_cat2 = df_comb_pt_ada_0_cat2.replace(np.nan, 0).astype(np.int64)
df_comb_pt_ada_0_cat2['row_count'] = df_comb_pt_ada_0_cat2['Overridden'] + df_comb_pt_ada_0_cat2['Removed']
# Include percentage column
df_comb_pt_ada_0_cat2['%_overridden'] = ((df_comb_pt_ada_0_cat2['Overridden'] / df_comb_pt_ada_0_cat2['row_count'])*100).round(2).astype(str) + '%'
df_comb_pt_ada_0_cat2['%_removed'] = ((df_comb_pt_ada_0_cat2['Removed'] / df_comb_pt_ada_0_cat2['row_count'])*100).round(2).astype(str) + '%'
# Convert dataframe to csv.
#df_comb_pt_ada_0_cat2.to_csv('df_comb_pt_ada_0_cat2_200821.csv')
df_comb_pt_ada_0_cat2

## Provider type cat2 by age range (df_comb_ada_0) - counts and percentages.
df_comb_pt_ada_0_age = df_comb_ada_0['provider_type_cat2'].groupby([df_comb_ada_0['age_range'], df_comb_ada_0['warning_status']]).count().unstack()
df_comb_pt_ada_0_age = df_comb_pt_ada_0_age.replace(np.nan, 0).astype(np.int64)
df_comb_pt_ada_0_age['row_count'] = df_comb_pt_ada_0_age['Overridden'] + df_comb_pt_ada_0_age['Removed']
# Include percentage column
df_comb_pt_ada_0_age['%_overridden'] = ((df_comb_pt_ada_0_age['Overridden'] / df_comb_pt_ada_0_age['row_count'])*100).round(2).astype(str) + '%'
df_comb_pt_ada_0_age['%_removed'] = ((df_comb_pt_ada_0_age['Removed'] / df_comb_pt_ada_0_age['row_count'])*100).round(2).astype(str) + '%'
# Convert dataframe to csv.
#df_comb_pt_ada_0_age.to_csv('df_comb_pt_ada_0_age_200821.csv')
df_comb_pt_ada_0_age

df_comb_pt_ada_0_sx = df_comb_ada_0['provider_type_cat2'].groupby([df_comb_ada_0['sex'], df_comb_ada_0['warning_status']]).count().unstack()
df_comb_pt_ada_0_sx

## Provider type cat2 by sex (df_comb_ada_0) - counts and percentages.
df_comb_pt_ada_0_sx = df_comb_ada_0['provider_type_cat2'].groupby([df_comb_ada_0['sex'], df_comb_ada_0['warning_status']]).count().unstack()
df_comb_pt_ada_0_sx = df_comb_pt_ada_0_sx.replace(np.nan, 0).astype(np.int64)
df_comb_pt_ada_0_sx['row_count'] = df_comb_pt_ada_0_sx['Overridden'] + df_comb_pt_ada_0_sx['Removed']
# Include percentage column
df_comb_pt_ada_0_sx['%_overridden'] = ((df_comb_pt_ada_0_sx['Overridden'] / df_comb_pt_ada_0_sx['row_count'])*100).round(2).astype(str) + '%'
df_comb_pt_ada_0_sx['%_removed'] = ((df_comb_pt_ada_0_sx['Removed'] / df_comb_pt_ada_0_sx['row_count'])*100).round(2).astype(str) + '%'
# Convert dataframe to csv.
#df_comb_pt_ada_0_sx.to_csv('df_comb_pt_ada_0_sx_200821.csv')
df_comb_pt_ada_0_sx

## Provider type cat2 by provider specialty cat 1 (df_comb_ada_0) - counts and percentages.
df_comb_pt_ada_0_spec = df_comb_ada_0['provider_type_cat2'].groupby([df_comb_ada_0['provider_specialty_cat1'],
df_comb_ada_0['warning_status']]).count().unstack()
df_comb_pt_ada_0_spec = df_comb_pt_ada_0_spec.replace(np.nan, 0).astype(np.int64)
df_comb_pt_ada_0_spec['row_count'] = df_comb_pt_ada_0_spec['Overridden'] + df_comb_pt_ada_0_spec['Removed']
# Include percentage column
df_comb_pt_ada_0_spec['%_overridden'] = ((df_comb_pt_ada_0_spec['Overridden'] / df_comb_pt_ada_0_spec['row_count'])*100).round(2).astype(str) + '%'
df_comb_pt_ada_0_spec['%_removed'] = ((df_comb_pt_ada_0_spec['Removed'] / df_comb_pt_ada_0_spec['row_count'])*100).round(2).astype(str) + '%'
# Convert dataframe to csv.
#df_comb_pt_ada_0_spec.to_csv('df_comb_pt_ada_0_spec_200821.csv')
df_comb_pt_ada_0_spec

## Provider type cat2 by description cat 1 (df_comb_ada_0) - counts and percentages.
df_comb_pt_ada_0_des = df_comb_ada_0['provider_type_cat2'].groupby([df_comb_ada_0['description_cat1'],
df_comb_ada_0['warning_status']]).count().unstack()
df_comb_pt_ada_0_des = df_comb_pt_ada_0_des.replace(np.nan, 0).astype(np.int64)
df_comb_pt_ada_0_des['row_count'] = df_comb_pt_ada_0_des['Overridden'] + df_comb_pt_ada_0_des['Removed']
# Include percentage column
df_comb_pt_ada_0_des['%_overridden'] = ((df_comb_pt_ada_0_des['Overridden'] / df_comb_pt_ada_0_des['row_count'])*100).round(2).astype(str) + '%'
df_comb_pt_ada_0_des['%_removed'] = ((df_comb_pt_ada_0_des['Removed'] / df_comb_pt_ada_0_des['row_count'])*100).round(2).astype(str) + '%'
# Convert dataframe to csv.
#df_comb_pt_ada_0_des.to_csv('df_comb_pt_ada_0_des_200821.csv')
df_comb_pt_ada_0_des

## Provider type cat2 by context (df_comb_ada_0) - counts and percentages.
df_comb_pt_ada_0_con = df_comb_ada_0['provider_type_cat2'].groupby([df_comb_ada_0['context'], df_comb_ada_0['warning_status']]).count().unstack()
df_comb_pt_ada_0_con = df_comb_pt_ada_0_des.replace(np.nan, 0).astype(np.int64)
df_comb_pt_ada_0_con['row_count'] = df_comb_pt_ada_0_con['Overridden'] + df_comb_pt_ada_0_con['Removed']
# Include percentage column
df_comb_pt_ada_0_con['%_overridden'] = ((df_comb_pt_ada_0_con['Overridden'] / df_comb_pt_ada_0_con['row_count'])*100).round(2).astype(str) + '%'
df_comb_pt_ada_0_con['%_removed'] = ((df_comb_pt_ada_0_con['Removed'] / df_comb_pt_ada_0_con['row_count'])*100).round(2).astype(str) + '%'
# Convert dataframe to csv.
#df_comb_pt_ada_0_con.to_csv('df_comb_pt_ada_0_con_200821.csv')
df_comb_pt_ada_0_con

## Provider type cat2 by DA reactions (df_comb_ada_0) - counts and percentages.
```

```
df_comb_pt_ada_0_dar = df_comb_ada_0['provider_type_cat2'].groupby([df_comb_ada_0['drug_allergy_reactions'],
df_comb_ada_0['warning_status']]).count().unstack()
df_comb_pt_ada_0_con = df_comb_pt_ada_0_des.replace(np.nan, 0).astype(np.int64)
df_comb_pt_ada_0_dar['row_count'] = df_comb_pt_ada_0_dar['Overridden'] + df_comb_pt_ada_0_dar['Removed']
# Include percentage column
df_comb_pt_ada_0_dar['%_overridden'] = ((df_comb_pt_ada_0_dar['Overridden'] / df_comb_pt_ada_0_dar['row_count'])*100).round(2).astype(str) + '%'
df_comb_pt_ada_0_dar['%_removed'] = ((df_comb_pt_ada_0_dar['Removed'] / df_comb_pt_ada_0_dar['row_count'])*100).round(2).astype(str) + '%'
# Convert dataframe to csv.
#df_comb_pt_ada_0_dar.to_csv('df_comb_pt_ada_0_dar_200821.csv')
df_comb_pt_ada_0_dar

## Provider type count by context, warning_status, cat1 provider type and cat 1 description.
df_comb_pt_cat1 = df_comb_ada['provider_type_cat1'].groupby([df_comb_ada['description_cat1'], df_comb_ada['context'],
df_comb_ada['warning_status']]).count().unstack()
df_comb_pt_cat1 = df_comb_pt_cat1.replace(np.nan, 0).astype(np.int64)
df_comb_pt_cat1['row_count'] = df_comb_pt_cat1['Cancelled'] + df_comb_pt_cat1['Held'] + df_comb_pt_cat1['Overridden'] + df_comb_pt_cat1['Removed'] +
df_comb_pt_cat1['Viewed']
# Include percentage column
df_comb_pt_cat1['%_overridden'] = ((df_comb_pt_cat1['Overridden'] / df_comb_pt_cat1['row_count'])*100).round(2).astype(str) + '%'
df_comb_pt_cat1['%_removed'] = ((df_comb_pt_cat1['Removed'] / df_comb_pt_cat1['row_count'])*100).round(2).astype(str) + '%'
df_comb_pt_cat1['%_viewed'] = ((df_comb_pt_cat1['Viewed'] / df_comb_pt_cat1['row_count'])*100).round(2).astype(str) + '%'
df_comb_pt_cat1['%_held'] = ((df_comb_pt_cat1['Held'] / df_comb_pt_cat1['row_count'])*100).round(2).astype(str) + '%'
df_comb_pt_cat1['%_cancelled'] = ((df_comb_pt_cat1['Cancelled'] / df_comb_pt_cat1['row_count'])*100).round(2).astype(str) + '%'
# Include percentage by overridden and removed.
df_comb_pt_cat1_toto = df_comb_pt_cat1['Overridden'].sum()
df_comb_pt_cat1_totr = df_comb_pt_cat1['Removed'].sum()
df_comb_pt_cat1['%_overridden_rate'] = ((df_comb_pt_cat1['Overridden'] / df_comb_pt_cat1_toto)*100).round(2).astype(str) + '%'
df_comb_pt_cat1['%_removed_rate'] = ((df_comb_pt_cat1['Removed'] / df_comb_pt_cat1_totr)*100).round(2).astype(str) + '%'
# Convert dataframe to csv.
#df_comb_pt_cat1.to_csv('df_comb_pt_cat1_150821.csv')
df_comb_pt_cat1

df_comb_pt_cat1.shape

## By Provider type Cat 2.
df_comb_pt_1 = df_comb_ada['description_cat1'].groupby([df_comb_ada['provider_type_cat2'], df_comb_ada['context'],
df_comb_ada['warning_status']]).count().unstack()
df_comb_pt_1 = df_comb_pt_1.replace(np.nan, 0).astype(np.int64)
df_comb_pt_1['row_count'] = df_comb_pt_1['Cancelled'] + df_comb_pt_1['Held'] + df_comb_pt_1['Overridden'] + df_comb_pt_1['Removed'] +
df_comb_pt_1['Viewed']
# Include percentage column
df_comb_pt_1['%_overridden'] = ((df_comb_pt_1['Overridden'] / df_comb_pt_1['row_count'])*100).round(2).astype(str) + '%'
df_comb_pt_1['%_removed'] = ((df_comb_pt_1['Removed'] / df_comb_pt_1['row_count'])*100).round(2).astype(str) + '%'
df_comb_pt_1['%_viewed'] = ((df_comb_pt_1['Viewed'] / df_comb_pt_1['row_count'])*100).round(2).astype(str) + '%'
df_comb_pt_1['%_held'] = ((df_comb_pt_1['Held'] / df_comb_pt_1['row_count'])*100).round(2).astype(str) + '%'
df_comb_pt_1['%_cancelled'] = ((df_comb_pt_1['Cancelled'] / df_comb_pt_1['row_count'])*100).round(2).astype(str) + '%'
# Include percentage by overridden and removed.
df_comb_pt_1_toto = df_comb_pt_1['Overridden'].sum()
df_comb_pt_1_totr = df_comb_pt_1['Removed'].sum()
df_comb_pt_1['%_overridden_rate'] = ((df_comb_pt_1['Overridden'] / df_comb_pt_1_toto)*100).round(2).astype(str) + '%'
df_comb_pt_1['%_removed_rate'] = ((df_comb_pt_1['Removed'] / df_comb_pt_1_totr)*100).round(2).astype(str) + '%'
# Convert dataframe to csv.
#df_comb_pt_1.to_csv('df_comb_pt_1_150821.csv')
df_comb_pt_1

## By Provider type Cat 1 and description Cat 1.
df_comb_pt_d1 = df_comb_ada['provider_type_cat1'].groupby([df_comb_ada['provider_type_cat1'], df_comb_ada['description_cat1'],
df_comb_ada['warning_status']]).count().unstack()
df_comb_pt_d1 = df_comb_pt_d1.replace(np.nan, 0).astype(np.int64)
df_comb_pt_d1['row_count'] = df_comb_pt_d1['Cancelled'] + df_comb_pt_d1['Held'] + df_comb_pt_d1['Overridden'] + df_comb_pt_d1['Removed'] +
df_comb_pt_d1['Viewed']
# Include percentage column
df_comb_pt_d1['%_overridden'] = ((df_comb_pt_d1['Overridden'] / df_comb_pt_d1['row_count'])*100).round(2).astype(str) + '%'
df_comb_pt_d1['%_removed'] = ((df_comb_pt_d1['Removed'] / df_comb_pt_d1['row_count'])*100).round(2).astype(str) + '%'
df_comb_pt_d1['%_viewed'] = ((df_comb_pt_d1['Viewed'] / df_comb_pt_d1['row_count'])*100).round(2).astype(str) + '%'
df_comb_pt_d1['%_held'] = ((df_comb_pt_d1['Held'] / df_comb_pt_d1['row_count'])*100).round(2).astype(str) + '%'
df_comb_pt_d1['%_cancelled'] = ((df_comb_pt_d1['Cancelled'] / df_comb_pt_d1['row_count'])*100).round(2).astype(str) + '%'
# Include percentage by overridden and removed.
df_comb_pt_d1_toto = df_comb_pt_d1['Overridden'].sum()
df_comb_pt_d1_totr = df_comb_pt_d1['Removed'].sum()
df_comb_pt_d1['%_overridden_rate'] = ((df_comb_pt_d1['Overridden'] / df_comb_pt_d1_toto)*100).round(2).astype(str) + '%'
df_comb_pt_d1['%_removed_rate'] = ((df_comb_pt_d1['Removed'] / df_comb_pt_d1_totr)*100).round(2).astype(str) + '%'
# Convert dataframe to csv.
#df_comb_pt_d1.to_csv('df_comb_pt_d1_150821.csv')
df_comb_pt_d1

## By Provider type Cat 2 and description Cat 1.
df_comb_pt_de1 = df_comb_ada['provider_type_cat2'].groupby([df_comb_ada['provider_type_cat2'], df_comb_ada['description_cat1'],
df_comb_ada['warning_status']]).count().unstack()
df_comb_pt_de1 = df_comb_pt_de1.replace(np.nan, 0).astype(np.int64)
df_comb_pt_de1['row_count'] = df_comb_pt_de1['Cancelled'] + df_comb_pt_de1['Held'] + df_comb_pt_de1['Overridden'] + df_comb_pt_de1['Removed'] +
df_comb_pt_de1['Viewed']
# Include percentage column
df_comb_pt_de1['%_overridden'] = ((df_comb_pt_de1['Overridden'] / df_comb_pt_de1['row_count'])*100).round(2).astype(str) + '%'
df_comb_pt_de1['%_removed'] = ((df_comb_pt_de1['Removed'] / df_comb_pt_de1['row_count'])*100).round(2).astype(str) + '%'
```

```
df_comb_pt_de1['%_viewed'] = ((df_comb_pt_de1['Viewed'] / df_comb_pt_de1['row_count'])*100).round(2).astype(str) + '%'
df_comb_pt_de1['%_held'] = ((df_comb_pt_de1['Held'] / df_comb_pt_de1['row_count'])*100).round(2).astype(str) + '%'
df_comb_pt_de1['%_cancelled'] = ((df_comb_pt_de1['Cancelled'] / df_comb_pt_de1['row_count'])*100).round(2).astype(str) + '%'
# Include percentage by overridden and removed.
df_comb_pt_de1_toto = df_comb_pt_de1['Overridden'].sum()
df_comb_pt_de1_totr = df_comb_pt_de1['Removed'].sum()
df_comb_pt_de1['%_overridden_rate'] = ((df_comb_pt_de1['Overridden'] / df_comb_pt_de1_toto)*100).round(2).astype(str) + '%'
df_comb_pt_de1['%_removed_rate'] = ((df_comb_pt_de1['Removed'] / df_comb_pt_de1_totr)*100).round(2).astype(str) + '%'
# Convert dataframe to csv.
#df_comb_pt_de1.to_csv('df_comb_pt_de1_150821.csv')
df_comb_pt_de1

## By Provider Specialty Cat 1.
df_comb_ps_1b = df_comb_ada['description_cat1'].groupby([df_comb_ada['provider_specialty_cat1'], df_comb_ada['description_cat1'],
df_comb_ada['warning_status']]).count().unstack()
df_comb_ps_1b = df_comb_ps_1b.replace(np.nan, 0).astype(np.int64)
df_comb_ps_1b['row_count'] = df_comb_ps_1b['Cancelled'] + df_comb_ps_1b['Held'] + df_comb_ps_1b['Overridden'] + df_comb_ps_1b['Removed'] +
df_comb_ps_1b['Viewed']
# Include percentage column
df_comb_ps_1b['%_overridden'] = ((df_comb_ps_1b['Overridden'] / df_comb_ps_1b['row_count'])*100).round(2).astype(str) + '%'
df_comb_ps_1b['%_removed'] = ((df_comb_ps_1b['Removed'] / df_comb_ps_1b['row_count'])*100).round(2).astype(str) + '%'
df_comb_ps_1b['%_viewed'] = ((df_comb_ps_1b['Viewed'] / df_comb_ps_1b['row_count'])*100).round(2).astype(str) + '%'
df_comb_ps_1b['%_held'] = ((df_comb_ps_1b['Held'] / df_comb_ps_1b['row_count'])*100).round(2).astype(str) + '%'
df_comb_ps_1b['%_cancelled'] = ((df_comb_ps_1b['Cancelled'] / df_comb_ps_1b['row_count'])*100).round(2).astype(str) + '%'
# Include percentage by overridden and removed.
df_comb_ps_1b_toto = df_comb_ps_1b['Overridden'].sum()
df_comb_ps_1b_totr = df_comb_ps_1b['Removed'].sum()
df_comb_ps_1b['%_overridden_rate'] = ((df_comb_ps_1b['Overridden'] / df_comb_ps_1b_toto)*100).round(2).astype(str) + '%'
df_comb_ps_1b['%_removed_rate'] = ((df_comb_ps_1b['Removed'] / df_comb_ps_1b_totr)*100).round(2).astype(str) + '%'
# Convert dataframe to csv.
#df_comb_ps_1b.to_csv('df_comb_ps_1b_150821.csv')
df_comb_ps_1b

## By Provider Specialty Cat 1 and description Cat 1.
df_comb_ps_1a = df_comb_ada['provider_specialty_cat1'].groupby([df_comb_ada['provider_specialty_cat1'], df_comb_ada['description_cat1'],
df_comb_ada['warning_status']]).count().unstack()
df_comb_ps_1a = df_comb_ps_1a.replace(np.nan, 0).astype(np.int64)
df_comb_ps_1a['row_count'] = df_comb_ps_1a['Cancelled'] + df_comb_ps_1a['Held'] + df_comb_ps_1a['Overridden'] + df_comb_ps_1a['Removed'] +
df_comb_ps_1a['Viewed']
# Include percentage column
df_comb_ps_1a['%_overridden'] = ((df_comb_ps_1a['Overridden'] / df_comb_ps_1a['row_count'])*100).round(2).astype(str) + '%'
df_comb_ps_1a['%_removed'] = ((df_comb_ps_1a['Removed'] / df_comb_ps_1a['row_count'])*100).round(2).astype(str) + '%'
df_comb_ps_1a['%_viewed'] = ((df_comb_ps_1a['Viewed'] / df_comb_ps_1a['row_count'])*100).round(2).astype(str) + '%'
df_comb_ps_1a['%_held'] = ((df_comb_ps_1a['Held'] / df_comb_ps_1a['row_count'])*100).round(2).astype(str) + '%'
df_comb_ps_1a['%_cancelled'] = ((df_comb_ps_1a['Cancelled'] / df_comb_ps_1a['row_count'])*100).round(2).astype(str) + '%'
# Include percentage by overridden and removed.
df_comb_ps_1a_toto = df_comb_ps_1a['Overridden'].sum()
df_comb_ps_1a_totr = df_comb_ps_1a['Removed'].sum()
df_comb_ps_1a['%_overridden_rate'] = ((df_comb_ps_1a['Overridden'] / df_comb_ps_1a_toto)*100).round(2).astype(str) + '%'
df_comb_ps_1a['%_removed_rate'] = ((df_comb_ps_1a['Removed'] / df_comb_ps_1a_totr)*100).round(2).astype(str) + '%'
# Convert dataframe to csv.
#df_comb_ps_1a.to_csv('df_comb_ps_1a_150821.csv')
df_comb_ps_1a

## Non_medical_prescriber - Provider type count by context, warning_status, cat1 provider type and cat 1 description.
df_comb_pt_cat1_ro_mp = df_comb_ro_mp['provider_type_cat1'].groupby([df_comb_ro_mp['description_cat1'], df_comb_ro_mp['context'],
df_comb_ro_mp['warning_status']]).count().unstack()
df_comb_pt_cat1_ro_mp = df_comb_pt_cat1_ro_mp.replace(np.nan, 0).astype(np.int64)
df_comb_pt_cat1_ro_mp['row_count'] = df_comb_pt_cat1_ro_mp['Overridden'] + df_comb_pt_cat1_ro_mp['Removed']
# Include percentage column
df_comb_pt_cat1_ro_mp['%_overridden'] = ((df_comb_pt_cat1_ro_mp['Overridden'] / df_comb_pt_cat1_ro_mp['row_count'])*100).round(2).astype(str) + '%'
df_comb_pt_cat1_ro_mp['%_removed'] = ((df_comb_pt_cat1_ro_mp['Removed'] / df_comb_pt_cat1_ro_mp['row_count'])*100).round(2).astype(str) + '%'
# Include percentage by overridden and removed.
df_comb_pt_cat1_ro_mp_toto = df_comb_pt_cat1_ro_mp['Overridden'].sum()
df_comb_pt_cat1_ro_mp_totr = df_comb_pt_cat1_ro_mp['Removed'].sum()
df_comb_pt_cat1_ro_mp['%_overridden_rate'] = ((df_comb_pt_cat1_ro_mp['Overridden'] / df_comb_pt_cat1_ro_mp_toto)*100).round(2).astype(str) + '%'
df_comb_pt_cat1_ro_mp['%_removed_rate'] = ((df_comb_pt_cat1_ro_mp['Removed'] / df_comb_pt_cat1_ro_mp_totr)*100).round(2).astype(str) + '%'
# Convert dataframe to csv.
#df_comb_pt_cat1_ro_mp.to_csv('df_comb_pt_cat1_ro_mp_150821.csv')
df_comb_pt_cat1_ro_mp
```

Filtered for medical_prescriber

```
## Medical_prescriber - Provider type count by context, warning_status, cat1 provider type and cat 1 description.
df_comb_pt_cat1_mp = df_comb_mp['provider_type_cat1'].groupby([df_comb_mp['description_cat1'], df_comb_mp['context'],
df_comb_mp['warning_status']]).count().unstack(
df_comb_pt_cat1_mp = df_comb_pt_cat1_mp.replace(np.nan, 0).astype(np.int64)
df_comb_pt_cat1_mp['row_count'] = df_comb_pt_cat1_mp['Cancelled'] + df_comb_pt_cat1_mp['Held'] + df_comb_pt_cat1_mp['Overridden'] +
df_comb_pt_cat1_mp['Removed'] + df_comb_pt_cat1_mp['Viewed']
# Include percentage column
df_comb_pt_cat1_mp['%_overridden'] = ((df_comb_pt_cat1_mp['Overridden'] / df_comb_pt_cat1_mp['row_count'])*100).round(2).astype(str) + '%'
df_comb_pt_cat1_mp['%_removed'] = ((df_comb_pt_cat1_mp['Removed'] / df_comb_pt_cat1_mp['row_count'])*100).round(2).astype(str) + '%'
df_comb_pt_cat1_mp['%_viewed'] = ((df_comb_pt_cat1_mp['Viewed'] / df_comb_pt_cat1_mp['row_count'])*100).round(2).astype(str) + '%'
df_comb_pt_cat1_mp['%_held'] = ((df_comb_pt_cat1_mp['Held'] / df_comb_pt_cat1_mp['row_count'])*100).round(2).astype(str) + '%'
df_comb_pt_cat1_mp['%_cancelled'] = ((df_comb_pt_cat1_mp['Cancelled'] / df_comb_pt_cat1_mp['row_count'])*100).round(2).astype(str) + '%'
```

```
# Include percentage by overridden and removed.
df_comb_pt_cat1_mp_toto = df_comb_pt_cat1_mp['Overridden'].sum()
df_comb_pt_cat1_mp_totr = df_comb_pt_cat1_mp['Removed'].sum()
df_comb_pt_cat1_mp['%_overridden_rate'] = ((df_comb_pt_cat1_mp['Overridden'] / df_comb_pt_cat1_mp_toto)*100).round(2).astype(str) + '%'
df_comb_pt_cat1_mp['%_removed_rate'] = ((df_comb_pt_cat1_mp['Removed'] / df_comb_pt_cat1_mp_totr)*100).round(2).astype(str) + '%'
# Convert dataframe to csv.
#df_comb_pt_cat1_mp.to_csv('df_comb_pt_cat1_mp_120821.csv')
df_comb_pt_cat1_mp
```

Filtered for medical_prescriber - removed and overridden

```
## Non_medical_prescriber - Provider type count by context, warning_status, cat1 provider type and cat 1 description.
df_comb_pt_cat1_ro_mp = df_comb_ro_mp['provider_type_cat1'].groupby([df_comb_ro_mp['description_cat1'], df_comb_ro_mp['context'],
df_comb_ro_mp['warning_status']]).count().unstack()
df_comb_pt_cat1_ro_mp = df_comb_pt_cat1_ro_mp.replace(np.nan, 0).astype(np.int64)
df_comb_pt_cat1_ro_mp['row_count'] = df_comb_pt_cat1_ro_mp['Overridden'] + df_comb_pt_cat1_ro_mp['Removed']
# Include percentage column
df_comb_pt_cat1_ro_mp['%_overridden'] = ((df_comb_pt_cat1_ro_mp['Overridden'] / df_comb_pt_cat1_ro_mp['row_count'])*100).round(2).astype(str) + '%'
df_comb_pt_cat1_ro_mp['%_removed'] = ((df_comb_pt_cat1_ro_mp['Removed'] / df_comb_pt_cat1_ro_mp['row_count'])*100).round(2).astype(str) + '%'
# Include percentage by overridden and removed.
df_comb_pt_cat1_ro_mp_toto = df_comb_pt_cat1_ro_mp['Overridden'].sum()
df_comb_pt_cat1_ro_mp_totr = df_comb_pt_cat1_ro_mp['Removed'].sum()
df_comb_pt_cat1_ro_mp['%_overridden_rate'] = ((df_comb_pt_cat1_ro_mp['Overridden'] / df_comb_pt_cat1_ro_mp_toto)*100).round(2).astype(str) + '%'
df_comb_pt_cat1_ro_mp['%_removed_rate'] = ((df_comb_pt_cat1_ro_mp['Removed'] / df_comb_pt_cat1_ro_mp_totr)*100).round(2).astype(str) + '%'
# Convert dataframe to csv.
#df_comb_pt_cat1_ro_mp.to_csv('df_comb_pt_cat1_ro_mp_150821.csv')
df_comb_pt_cat1_ro_mp
```

Filtered for non_medical_prescriber

```
## Provider type count by context, warning_status, cat1 provider type and cat 1 description.
df_comb_pt_cat1_nmp = df_comb_nmp['provider_type_cat1'].groupby([df_comb_nmp['description_cat1'], df_comb_nmp['context'],
df_comb_nmp['warning_status']]).count().unstack()
df_comb_pt_cat1_nmp = df_comb_pt_cat1_nmp.replace(np.nan, 0).astype(np.int64)
df_comb_pt_cat1_nmp['row_count'] = df_comb_pt_cat1_nmp['Cancelled'] + df_comb_pt_cat1_nmp['Held'] + df_comb_pt_cat1_nmp['Overridden'] +
df_comb_pt_cat1_nmp['Removed'] + df_comb_pt_cat1_nmp['Viewed']
# Include percentage column
df_comb_pt_cat1_nmp['%_overridden'] = ((df_comb_pt_cat1_nmp['Overridden'] / df_comb_pt_cat1_nmp['row_count'])*100).round(2).astype(str) + '%'
df_comb_pt_cat1_nmp['%_removed'] = ((df_comb_pt_cat1_nmp['Removed'] / df_comb_pt_cat1_nmp['row_count'])*100).round(2).astype(str) + '%'
df_comb_pt_cat1_nmp['%_viewed'] = ((df_comb_pt_cat1_nmp['Viewed'] / df_comb_pt_cat1_nmp['row_count'])*100).round(2).astype(str) + '%'
df_comb_pt_cat1_nmp['%_held'] = ((df_comb_pt_cat1_nmp['Held'] / df_comb_pt_cat1_nmp['row_count'])*100).round(2).astype(str) + '%'
df_comb_pt_cat1_nmp['%_cancelled'] = ((df_comb_pt_cat1_nmp['Cancelled'] / df_comb_pt_cat1_nmp['row_count'])*100).round(2).astype(str) + '%'
# Include percentage by overridden and removed.
df_comb_pt_cat1_nmp_toto = df_comb_pt_cat1_nmp['Overridden'].sum()
df_comb_pt_cat1_nmp_totr = df_comb_pt_cat1_nmp['Removed'].sum()
df_comb_pt_cat1_nmp['%_overridden_rate'] = ((df_comb_pt_cat1_nmp['Overridden'] / df_comb_pt_cat1_nmp_toto)*100).round(2).astype(str) + '%'
df_comb_pt_cat1_nmp['%_removed_rate'] = ((df_comb_pt_cat1_nmp['Removed'] / df_comb_pt_cat1_nmp_totr)*100).round(2).astype(str) + '%'
# Convert dataframe to csv.
#df_comb_pt_cat1_nmp.to_csv('df_comb_pt_cat1_nmp_120821.csv')
df_comb_pt_cat1_nmp
```

Filtered for non_medical_prescriber - removed and overridden.

```
## Non_medical_prescriber - Provider type count by context, warning_status, cat1 provider type and cat 1 description.
df_comb_pt_cat1_ro_nmp = df_comb_ro_nmp['provider_type_cat1'].groupby([df_comb_ro_nmp['description_cat1'], df_comb_ro_nmp['context'],
df_comb_ro_nmp['warning_status']]).count().unstack()
df_comb_pt_cat1_ro_nmp = df_comb_pt_cat1_ro_nmp.replace(np.nan, 0).astype(np.int64)
df_comb_pt_cat1_ro_nmp['row_count'] = df_comb_pt_cat1_ro_nmp['Overridden'] + df_comb_pt_cat1_ro_nmp['Removed']
# Include percentage column
df_comb_pt_cat1_ro_nmp['%_overridden'] = ((df_comb_pt_cat1_ro_nmp['Overridden'] / df_comb_pt_cat1_ro_nmp['row_count'])*100).round(2).astype(str) +
'%'
df_comb_pt_cat1_ro_nmp['%_removed'] = ((df_comb_pt_cat1_ro_nmp['Removed'] / df_comb_pt_cat1_ro_nmp['row_count'])*100).round(2).astype(str) + '%'
# Include percentage by overridden and removed.
df_comb_pt_cat1_ro_nmp_toto = df_comb_pt_cat1_ro_nmp['Overridden'].sum()
df_comb_pt_cat1_ro_nmp_totr = df_comb_pt_cat1_ro_nmp['Removed'].sum()
df_comb_pt_cat1_ro_nmp['%_overridden_rate'] = ((df_comb_pt_cat1_ro_nmp['Overridden'] / df_comb_pt_cat1_ro_nmp_toto)*100).round(2).astype(str) + '%'
df_comb_pt_cat1_ro_nmp['%_removed_rate'] = ((df_comb_pt_cat1_ro_nmp['Removed'] / df_comb_pt_cat1_ro_nmp_totr)*100).round(2).astype(str) + '%'
# Convert dataframe to csv.
#df_comb_pt_cat1_ro_nmp.to_csv('df_comb_pt_cat1_ro_nmp_120821.csv')
df_comb_pt_cat1_ro_nmp
```

Survey of provider_type_cat2 using crosstabs

```
## Provider type count by context, warning_status, cat1 provider type and cat 1 description.
df_comb_pt_cat2 = df_comb['provider_type_cat2'].groupby([df_comb['description_cat1'], df_comb['context'], df_comb['warning_status']]).count().unstack()
df_comb_pt_cat2 = df_comb_pt_cat2.replace(np.nan, 0).astype(np.int64)
df_comb_pt_cat2['row_count'] = df_comb_pt_cat2['Cancelled'] + df_comb_pt_cat2['Held'] + df_comb_pt_cat2['Overridden'] + df_comb_pt_cat2['Removed'] +
df_comb_pt_cat2['Viewed']
# Include percentage column
df_comb_pt_cat2['%_overridden'] = ((df_comb_pt_cat2['Overridden'] / df_comb_pt_cat2['row_count'])*100).round(2).astype(str) + '%'
df_comb_pt_cat2['%_removed'] = ((df_comb_pt_cat2['Removed'] / df_comb_pt_cat2['row_count'])*100).round(2).astype(str) + '%'
df_comb_pt_cat2['%_viewed'] = ((df_comb_pt_cat2['Viewed'] / df_comb_pt_cat2['row_count'])*100).round(2).astype(str) + '%'
df_comb_pt_cat2['%_held'] = ((df_comb_pt_cat2['Held'] / df_comb_pt_cat2['row_count'])*100).round(2).astype(str) + '%'
df_comb_pt_cat2['%_cancelled'] = ((df_comb_pt_cat2['Cancelled'] / df_comb_pt_cat2['row_count'])*100).round(2).astype(str) + '%'
# Include percentage by overridden and removed.
```

```
df_comb_pt_cat2_toto = df_comb_pt_cat2['Overridden'].sum()
df_comb_pt_cat2_totr = df_comb_pt_cat2['Removed'].sum()
df_comb_pt_cat2['%_overridden_rate'] = ((df_comb_pt_cat2['Overridden'] / df_comb_pt_cat2_toto)*100).round(2).astype(str) + '%'
df_comb_pt_cat2['%_removed_rate'] = ((df_comb_pt_cat2['Removed'] / df_comb_pt_cat2_totr)*100).round(2).astype(str) + '%'
# Convert dataframe to csv.
#df_comb_pt_cat2.to_csv('df_comb_pt_cat2_120821.csv')
df_comb_pt_cat2

df_comb_pt_cat2.shape
```

Survey of provider_type_cat1 using crosstabs

```
# Crosstabs of Cat 1 provider type, Cat 1 description and context.
df_comb_pt_crss = pd.crosstab([df_comb_ada.context, df_comb_ada.description_cat1, df_comb_ada.warning_status], df_comb_ada.provider_type_cat1,
margins=True)
# Include percentage column
df_comb_pt_crss['%_missing'] = ((df_comb_pt_crss.iloc[:, :1] / 53057)*100).round(2).astype(str) + '%'
df_comb_pt_crss['%_medical_prescriber'] = ((df_comb_pt_crss['medical_prescriber'] / 53057)*100).round(2).astype(str) + '%'
df_comb_pt_crss['%_non_medical_prescriber'] = ((df_comb_pt_crss['non_medical_prescriber'] / 53057)*100).round(2).astype(str) + '%'
df_comb_pt_crss['%_all'] = ((df_comb_pt_crss['All'] / 53057)*100).round(2).astype(str) + '%'
# Convert dataframe to csv.
#df_comb_pt_crss.to_csv('df_comb_pt_crss_110821.csv')
pd.set_option('display.max_rows', 112)
df_comb_pt_crss

df_comb_pt_crss.shape
```

End