## 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**