In [155…
```
import jaydebeapi, sys, os # allows connection to Netezza and executes SQL q
import pandas as pd # for saving results from SQL queries to pandas DFs
import numpy as np # for math operations

import matplotlib.pyplot as plt # for plotting data
plt.style.use('seaborn-whitegrid')
%matplotlib inline
```

In [156…
```
def get_df_from_db(sql):
    cursor = conn.cursor()
    cursor.execute(sql)
    data = cursor.fetchall()
    columnDes = cursor.description
    columnNames = [columnDes[i][0] for i in range(len(columnDes))]
    df = pd.DataFrame([list(i) for i in data],columns=columnNames)
    return df
```

In [157…
```
%run -i 'login.py'
```

```
Connection String: jdbc:netezza://ori-netezza.vumc.org:5480/DENNY_OMOP_SD
('2022-08-24 20:22:43',)
```

Observation table

Initial approach to Observation table:

The initial approach consisted of the following steps:

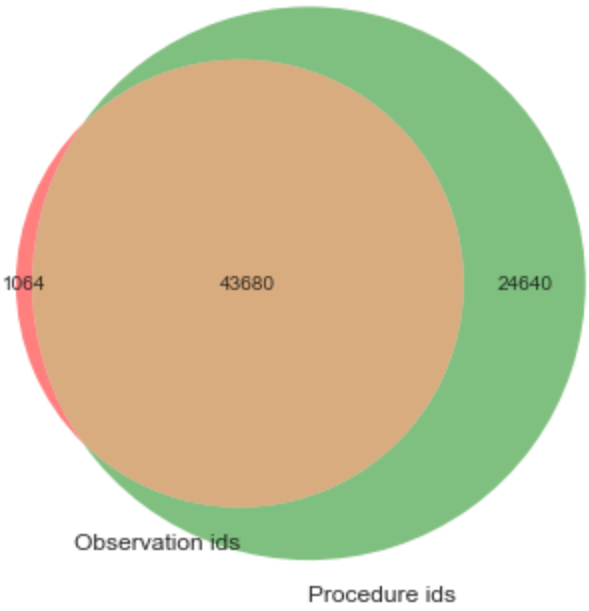
1. Gather the set of entries from the Observation table having concept IDs 40485048 or 4260747 into one table
2. Intersect *that* set of entries with the set of person ids having one or more CPT codes from the Procedure table
3. Use the resulting or intersecting set of Observation person IDs to further filter the table from .1
4. Clean up the EGAs from the table from .3
5. Use the observation date, i.e. the delivery date, and resulting finished EGAs to calculate ETOC, the prepartum period, the pregnancy period, and the postpartum period

---

In [166…
```
import matplotlib.pyplot as plt
from matplotlib_venn import venn2

plt.figure(figsize=(6,6))
```

```
venn2(subsets = (1064, 24640, 43680), set_labels = ('Observation ids', 'Proc
plt.show()
```



Step 1. from above:

In [ ]:
```
curs=conn.cursor()
curs.execute("CREATE TABLE observation_person_ids AS \
SELECT person_id, observation_date, value_as_string FROM v.observation a \
WHERE (a.observation_concept_id = 40485048 OR a.observation_concept_id =4260
ORDER BY person_id, observation_date")
```

In [124…
```
curs=conn.cursor()
curs.execute("SELECT * FROM observation_person_ids ORDER BY person_id LIMIT
df = pd.DataFrame(curs.fetchall())
df.columns = ['person id', 'observation date', 'value as string']
df.iloc[:,:]
```

Out[124]:

	person id	observation date	value as string
0	[redacted]	2012-09-22	36 5/7 weeks
1		2011-05-10	31 5/7 weeks
2		2010-03-15	41 1/7
3		2008-12-01	41 6/7
4		2008-12-01	41 6/7
5		2019-12-03	37w 4d
6		2016-02-29	38 2/7 weeks
7		2016-04-21	38 6/7 weeks
8		2014-05-29	39 weeks
9		2016-07-19	41 1/7 weeks

a) Dimensions of the table from step 1.

In [125…

```
curs=conn.cursor()
curs.execute("SELECT count(DISTINCT person_id), count(*)  FROM observation_p
df = pd.DataFrame(curs.fetchall())
df.columns = ['Distinct person id', 'total entries']
df.iloc[:,:]
```

Out[125]:

	Distinct person id	total entries
0	46370	68362

b) Dimensions of the table from step 1. after grouping.

In [126…

```
curs=conn.cursor()
curs.execute(" \
SELECT count(DISTINCT person_id), count(*) FROM ( \
SELECT person_id, year_month, count(date_counts) FROM ( \
SELECT person_id, year_month, count(observation_date) AS date_counts FROM (
SELECT person_id, cast(year(observation_date) as varchar (8)) || '-' || cast
GROUP BY person_id, year_month ) b \
GROUP BY person_id, year_month ) c \
")
df = pd.DataFrame(curs.fetchall())
df.columns = ['Distinct person id', 'total entries accounting for reps']
df.iloc[:,:]
```

Out[126]:

	Distinct person id	total entries accounting for reps
0	46370	59637

In [129…

```
curs=conn.cursor()
```

```
curs.execute("SELECT * FROM ( \
SELECT person_id, \
year_month, \
count(observation_date) AS invd_counts, \
min(observation_date) AS min_date, \
max(observation_date) AS max_date, \
max(observation_date) - min(observation_date) AS diff FROM ( \
SELECT person_id, cast(year(observation_date) as varchar (8)) || '-' || cast
GROUP BY person_id, year_month ORDER BY diff DESC ) b \
WHERE diff > 2 ORDER BY diff DESC")
df = pd.DataFrame(curs.fetchall())
df.columns = ['person_id','year_month','row_counts','min_date','max_date','c
df.iloc[:,:]
```

Out[129]:

	person_id	year_month	row_counts	min_date	max_date	diff
0	[redacted]	2013-12	2	2013-12-04	2013-12-27	23
1		2014-9	2	2014-09-03	2014-09-26	23
2		2013-2	3	2013-02-03	2013-02-25	22
3		2016-11	2	2016-11-11	2016-11-30	19
4		2010-6	2	2010-06-05	2010-06-24	19
...		...	...	...	...	...
98		2016-7	2	2016-07-14	2016-07-17	3
99		2012-10	2	2012-10-10	2012-10-13	3
100		2011-12	2	2011-12-08	2011-12-11	3
101		2012-12	2	2012-12-11	2012-12-14	3
102		2012-1	3	2012-01-03	2012-01-06	3

103 rows × 6 columns

Procedure table

In [128…

```
curs=conn.cursor()
curs.execute("SELECT * FROM ( \
SELECT person_id, year_month, count(procedure_date) AS row_counts, \
            min(procedure_date) AS min_date, max(procedure_date) AS max_dat
            max(procedure_date) - min(procedure_date) AS diff FROM ( \
SELECT DISTINCT * FROM ( \
SELECT person_id, year_month, procedure_date FROM jn_step_3) a \
)b GROUP BY person_id, year_month ORDER BY diff DESC ) b \
WHERE diff > 2 ORDER BY diff DESC")
df=pd.DataFrame(curs.fetchall())
df.columns = ['person_id','year_month','row_counts','min_date','max_date','c
df.iloc[:,:]
```

Out[128]:

	person_id	year_month	row_counts	min_date	max_date	diff
0	[redacted]	2021-7	2	2021-07-02	2021-07-29	27
1		2005-12	2	2005-12-05	2005-12-23	18
2		2018-3	2	2018-03-04	2018-03-20	16
3		2018-10	2	2018-10-13	2018-10-29	16
4		2017-4	2	2017-04-14	2017-04-28	14
...		...	...	...	...	...
78		2020-9	2	2020-09-21	2020-09-24	3
79		2021-9	2	2021-09-25	2021-09-28	3
80		1999-5	2	1999-05-06	1999-05-09	3
81		2010-11	2	2010-11-03	2010-11-06	3
82		2017-5	3	2017-05-18	2017-05-21	3

83 rows × 6 columns

Here is what I did...

1. WORKING FROM THE PROCEDURE TABLE

In [ ]:
```
curs=conn.cursor()
curs.execute("CREATE TABLE JN_step_2 AS \
                SELECT person_id, cast(year(procedure_date) as varchar (8))
                AS year_date, procedure_date, cpt_codes FROM JN_step_1 \
                ORDER BY procedure_date")
```

In [ ]:
```
curs=conn.cursor()
curs.execute("CREATE TABLE procedure_id_date AS \
                SELECT person_id, year_date, max(procedure_date) AS most_rec
                GROUP BY person_id, year_date \
                ORDER BY person_id")
```

Inspect procedure_id_date

In [26]:
```
curs=conn.cursor()
curs.execute("SELECT *  FROM procedure_id_date LIMIT 20")
df = pd.DataFrame(curs.fetchall())
df.columns = ['person id','year date','most recent proc date']
df.iloc[:,:]
```

Out[26]:

	person id	year date	most recent proc date
0	[redacted]	2020	2020-05-09
1	[redacted]	2014	2014-12-23
2	[redacted]	2017	2017-05-15
3	[redacted]	2016	2016-05-10
4	[redacted]	2019	2019-09-19
5	[redacted]	2017	2017-10-09
6	[redacted]	2019	2019-12-10
7	[redacted]	2006	2006-02-13
8	[redacted]	2007	2007-10-13
9	[redacted]	2008	2008-07-01
10	[redacted]	2013	2013-06-27
11	[redacted]	2008	2008-01-06
12	[redacted]	2016	2016-09-23
13	[redacted]	2017	2017-12-04
14	[redacted]	2002	2002-02-28
15	[redacted]	2021	2021-07-17
16	[redacted]	2017	2017-11-27
17	[redacted]	2002	2002-09-12
18	[redacted]	2016	2016-08-29
19	[redacted]	2018	2018-03-20

Inspect procedure_id_date table dimensions

In [24]:
```
curs=conn.cursor()
curs.execute("SELECT COUNT(DISTINCT person_id), COUNT(*) FROM procedure_id_c
df=pd.DataFrame(curs.fetchall())
df.columns = ['invd person ids','total rows']
df.iloc[:,:]
```

Out[24]:

	invd person ids	total rows
0	69317	90695

Validation for the procedure_id_date table

In [32]:
```
curs.execute("SELECT person_id, count(most_recent_date) as counts FROM proce
```

```
                GROUP BY person_id ORDER BY counts desc LIMIT 10")
df = pd.DataFrame(curs.fetchall())
df.columns = ['person id','counts']
df.iloc[:,:]
```

Out[32]:

	person id	counts
0	[redacted]	9
1	[redacted]	8
2	[redacted]	8
3	[redacted]	8
4	[redacted]	7
5	[redacted]	7
6	[redacted]	7
7	[redacted]	7
8	[redacted]	7
9	[redacted]	7

In [35]:

```
curs.execute("SELECT * FROM procedure_id_date WHERE person_id = '2535740'")
df = pd.DataFrame(curs.fetchall())
df.columns = ['person id','year date','most recent proc date']
df.iloc[:,:]
```

Out[35]:

	person id	year date	most recent proc date
0	[redacted]	1999	1999-02-19
1	[redacted]	2001	2001-12-17
2	[redacted]	2004	2004-01-21
3	[redacted]	2006	2006-01-02
4	[redacted]	2007	2007-10-01
5	[redacted]	2009	2009-10-06
6	[redacted]	2011	2011-09-24
7	[redacted]	2013	2013-08-11
8	[redacted]	2016	2016-02-28

2. WORKING FROM THE OBSERVATION TABLE

In [150…

```
curs=conn.cursor()
curs.execute("CREATE TABLE observation_person_ids AS \
```

```
                SELECT person_id, observation_date, value_as_string FROM v.c
                WHERE (a.observation_concept_id = 40485048 OR a.observation_
```

Inspect observation_person_ids

In [37]:
```
curs=conn.cursor()
curs.execute("SELECT *  FROM observation_person_ids LIMIT 20")
df = pd.DataFrame(curs.fetchall())
df.columns = ['person id', 'observation date', 'value_as_string']
df.iloc[:,:]
```

Out[37]:

	person id	observation date	value_as_string
0	[redacted]	2020-10-25	39w
1	[redacted]	2017-11-06	36w 6d
2	[redacted]	2019-10-06	37w 5d
3	[redacted]	2021-06-24	39w
4	[redacted]	2019-03-31	39w 2d
5	[redacted]	2020-06-12	39w 1d
6	[redacted]	2019-03-08	38w 5d
7	[redacted]	2018-09-02	39w 2d
8	[redacted]	2018-08-13	31w
9	[redacted]	2018-08-13	31w
10	[redacted]	2021-05-09	40w 5d
11	[redacted]	2021-01-03	38w 5d
12	[redacted]	2017-06-27	40w 2d
13	[redacted]	2019-12-09	41w 1d
14	[redacted]	2019-01-11	26w 5d
15	[redacted]	2019-01-11	26w 5d
16	[redacted]	2020-06-26	39w 2d
17	[redacted]	2017-10-01	39w 2d
18	[redacted]	2018-01-18	41w 1d
19	[redacted]	2020-04-10	40w 6d

Inspect observation_person_ids table dimensions

In [151…
```
curs=conn.cursor()
curs.execute("SELECT count(DISTINCT person_id), count(*)  FROM observation_p
df = pd.DataFrame(curs.fetchall())
```

```
df.columns = ['Distinct person id', 'total entries']
df.iloc[:,:]
```

Out[151]:

	Distinct person id	total entries
0	46370	68362

Creating the preliminary observation table

The preliminary observation table has the following elements:

- person ids
- observation date (most recent)
- egas (unfinished)
- procedure date (as delivery date)

The next three temp_observation* tables are intermediates leading up to the creation of the preliminary_observation table

---

In [ ]:
```
12345 2011 2011-04-06  44 wks
12345 2011 2011-04-06  44 weeks

12345 2011 2011-04-06




group by personid, yearindex, ega
```

In [4]:
```
curs.execute("CREATE TABLE temp_observation AS \
                SELECT person_id, cast(year(observation_date) as varchar (8)
                observation_date, value_as_string AS ega FROM observation_pe
```

In [337…
```
curs.execute("SELECT person_id, cast(year(observation_date) as varchar (8))
                observation_date, value_as_string AS ega FROM observation_pe
df=pd.DataFrame(curs.fetchall())
df.iloc[:,:]
```

Out[337]:

	0	1	2	3
0	[redacted]	2017	2017-09-03	40w 1d
1		2020	2020-07-12	40w 2d
2		2021	2021-05-22	39w 3d
3		2020	2020-11-18	36w 6d
4		2018	2018-05-10	39w 6d
...		...	...	...
68357		2021	2021-12-19	39w 4d
68358		2021	2021-01-02	40w 2d
68359		2021	2021-01-21	39w 3d
68360		2019	2019-05-09	40w
68361		2020	2020-08-11	15w 3d

68362 rows × 4 columns

In [ ]:
```
curs.execute("CREATE TABLE temp_observation_2 AS \
                SELECT person_id, year_date, max(observation_date) AS most_r
                SELECT person_id, cast(year(observation_date) as varchar (8)
                observation_date, value_as_string AS ega FROM observation_pe
                GROUP BY person_id, year_date")
```

In [343…
```
curs.execute("SELECT person_id, year_date, max(observation_date) AS most_rec
                SELECT person_id, cast(year(observation_date) as varchar (8)
                observation_date, value_as_string AS ega FROM observation_pe
                GROUP BY person_id, year_date")
df=pd.DataFrame(curs.fetchall())
df.iloc[:,:]
```

Out[343]:

	0	1	2
0	[redacted]	2017	2017-09-03
1	[redacted]	2020	2020-07-12
2	[redacted]	2021	2021-05-22
3	[redacted]	2020	2020-11-18
4	[redacted]	2018	2018-05-10
...	[redacted]	...	...
59495	[redacted]	2019	2019-08-11
59496	[redacted]	2018	2018-05-28
59497	[redacted]	2021	2021-04-28
59498	[redacted]	2021	2021-06-09
59499	[redacted]	2019	2019-10-25

59500 rows × 3 columns

In [ ]:
```
curs.execute("CREATE TABLE temp_observation_3 AS \
                SELECT b.person_id, b.year_date, b.observation_date, \
                a.most_recent_observation, b.ega FROM temp_observation_2 a \
                INNER JOIN temp_observation b \
                ON a.person_id = b.person_id AND a.year_date = b.year_date \
                AND a.most_recent_observation = b.observation_date")
df=pd.DataFrame(curs.fetchall())
df.iloc[:,:]
```

NOTE: temp_observation_3 should still have duplicate rows for identical observation dates and EGA string per person / identical observation dates and different EGA string per person

In [344…
```
curs.execute("SELECT b.person_id, b.year_date, b.observation_date, \
                a.most_recent_observation, b.ega FROM temp_observation_2 a \
                INNER JOIN temp_observation b \
                ON a.person_id = b.person_id AND a.year_date = b.year_date \
                AND a.most_recent_observation = b.observation_date")
df=pd.DataFrame(curs.fetchall())
df.iloc[:,:]
```

Out[344]:

	0	1	2	3	4
0	1305101	2019	2019-06-01	2019-06-01	34w 6d
1	110363	2018	2018-07-16	2018-07-16	39w
2	821494	2017	2017-12-08	2017-12-08	41w
3	5047719	2021	2021-11-05	2021-11-05	40w 6d
4	4741234	2021	2021-11-04	2021-11-04	40w 6d
...	...	...	...	...	...
67524	3028400	2021	2021-09-16	2021-09-16	39w 6d
67525	2720763	2018	2018-11-07	2018-11-07	41w
67526	110291	2020	2020-03-17	2020-03-17	39w 2d
67527	4310931	2020	2020-03-27	2020-03-27	39w
67528	2280932	2021	2021-12-15	2021-12-15	40w 4d

67529 rows × 5 columns

In [8]:
```
curs.execute("CREATE TABLE preliminary_observation AS \
                SELECT * FROM ( \
                SELECT a.person_id, a.most_recent_observation, a.ega, \
                b.most_recent_date AS delivery FROM temp_observation_3 a \
                INNER JOIN procedure_id_date b \
                ON a.person_id = b.person_id AND a.year_date = b.year_date)
```

Inspect preliminary_observation

In [184…
```
curs=conn.cursor()
curs.execute("SELECT * FROM preliminary_observation  LIMIT 10")
df=pd.DataFrame(curs.fetchall())
df.columns = ['person id','most recent obs date','ega','delivery']
df.iloc[:,:]
```

Out[184]:

	person id	most recent obs date	ega	delivery
0	4698527	2021-02-13	39w	2021-02-13
1	4193800	2020-08-08	38w 3d	2020-08-08
2	2385324	2018-06-06	39w 3d	2018-06-06
3	4048693	2019-02-08	36w 3d	2019-02-08
4	4193800	2019-02-03	39w	2019-02-03
5	5112630	2021-05-24	39w 3d	2021-05-24
6	4126118	2018-07-30	34w 3d	2018-07-30
7	487903	2020-10-17	39w 2d	2020-10-17
8	4211848	2018-10-18	41w 5d	2018-10-18
9	4271225	2020-02-07	38w 4d	2020-02-07

Inspect preliminary_observation dimensions

In [185…

```
curs=conn.cursor()
curs.execute("SELECT COUNT(DISTINCT person_id), \
              COUNT(*) FROM preliminary_observation")
df=pd.DataFrame(curs.fetchall())
df.columns = ['invd person ids','total rows']
df.iloc[:,:]
```

Out[185]:

	invd person ids	total rows
0	45169	65175

Validation of the observation_subset_on_procedure_ids_dates

Validation: Count the number of delivery dates associated with a person id and an observation date

Intention: This table shows how many delivery dates from the procedure table are associated with a person id and an observation date. The question to answer here is this: Where do these multiple entries stem from?

In [190…

```
curs.execute("SELECT person_id, year_date, count(delivery) AS counts FROM (
```

```
                SELECT person_id, cast(year(most_recent_observation) as varc
                delivery FROM preliminary_observation) a \
                GROUP BY  person_id, year_date \
                ORDER BY counts desc LIMIT 10")
df=pd.DataFrame(curs.fetchall())
df.columns = ['person id','year date','delivery count']
df.iloc[:,:]
```

Out[190]:

	person id	year date	delivery count
0	[redacted]	2011	4
1		2019	4
2		2011	3
3		2010	3
4		2011	3
5		2013	3
6		2012	3
7		2013	3
8		2011	3
9		2013	3

Inspecting the results from the procedure table:

In [347…

```
curs.execute("SELECT * FROM procedure_id_date WHERE person_id = '1865888'")
df=pd.DataFrame(curs.fetchall())
df.columns = ['person id','year date','most recent delivery date']
df.iloc[:,:]
```

Out[347]:

	person id	year date	most recent delivery date
0	[redacted]	2011	2011-11-08

Inspecting the results from the observation table:

In [191…

```
curs.execute("SELECT * FROM observation_person_ids WHERE person_id = '186588
df=pd.DataFrame(curs.fetchall())
df.columns = ['person id','observation date','ega']
df.iloc[:,:]
```

Out[191]:

	person id	observation date	ega
0	[redacted]	2011-11-08	40 6/7
1	[redacted]	2011-11-08	40 6/7
2	[redacted]	2011-11-08	40 6/7 weeks
3	[redacted]	2011-11-08	40 6/7 weeks

Inspecting the results from joining the 'observation table' on the 'procedure table':

---

curs.execute("SELECT * FROM preliminary_observation
WHERE person_id = '1865888'") df=pd.DataFrame(curs.fetchall()) df.columns = ['person id','most recent observation date','ega','most recent delivery date'] df.iloc[:,:]

Note: These 'duplicates' are of little concern since a 'distinct' function can be used to eliminate duplicate rows after the EGAs are cleaned up

---

Clean the EGAs into a useable form for calculating periods and cohorts

---

In [310…

```
curs=conn.cursor()
curs.execute("UPDATE preliminary_observation \
                SET ega = REPLACE(ega,'22  4','22 4')")
```

List of string patterns that were replaced (run code above with different corrections from below):

- ' weeks','weeks',' wks','wks','1d-6d',' days',' 0/7-6/7','w ','w','.0-.6','/1-','+1-+6',''. =', ' 27','413/7','~',' ~','approx 32','unknon,'est 34-36 by pt','41 and 4','41 60/7',' 35',' 28','236d', '26.7','27 1ay','28-32','32 and 4','32 3/','33-35','34 by first U/S','36-39','362d','36 5 EGA','38 22', '41 & 1 day','404','400d','40.','40 and 2','40 6v','41 & 1 day','40 45/7','40 3 Days', '40 1 day','40 1EGA','40 + 2','39-40','39 5ays','39 1

/7','381','38-41','38+ 2','38 5/8','37 5 6','37 37', '29 + 6','22 4','-','4657 2','424','12 20 2012','10/7/09' 'unknon, term','unknon','no dating','TERM','Delivered',''

Inspect the table to see the results (Note: kept table name preliminary_observation)

In [311…

```
curs=conn.cursor()
curs.execute("SELECT * FROM preliminary_observation ORDER BY person_id LIMIT
df = pd.DataFrame(curs.fetchall())
df.columns = ['person ids', 'observation date', 'egas', 'delivery']
df.iloc[:,:]
```

Out[311]:

	person ids	observation date	egas	delivery
0	[redacted]	2012-09-22	36 5	2012-09-22
1		2011-05-10	31 5	2011-05-10
2		2008-12-01	41 6	2008-12-01
3		2010-03-15	41 1	2010-03-15
4		2008-12-01	41 6	2008-12-01
5		2016-02-29	38 2	2016-02-29
6		2019-12-03	37 4	2019-12-03
7		2016-04-21	38 6	2016-04-21
8		2014-05-29	39	2014-05-29
9		2016-07-19	41 1	2016-07-19
10		2010-02-06	36 4	2010-02-06
11		2014-08-25	40 2	2014-08-25
12		2017-09-01	40 4	2017-09-01
13		2019-11-21	40 1	2019-11-21
14		2016-02-03	39 3	2016-02-03
15		2011-10-25	39	2011-10-24
16		2009-08-17	39 1	2009-08-17
17		2007-06-12	37 4	2007-06-12
18		2020-05-08	32 6	2020-05-08
19		2021-02-28	40 3	2021-02-28

Working with NA and empty ega entries; Remove duplicate rows

Count the number of NAs

In [313…
```
curs=conn.cursor()
curs.execute("SELECT count(*) FROM preliminary_observation WHERE ega = 'NA'"
df=pd.DataFrame(curs.fetchall())
df.columns = ['total NAs']
df.iloc[:,:]
```

Out[313]:

	total NAs
0	237

Remove rows with NA and inspect the new table dimensions

In [ ]:
```
curs=conn.cursor()
curs.execute("CREATE TABLE preliminary_observation_2 AS SELECT * FROM prelim
```

In [318…
```
curs=conn.cursor()
curs.execute("SELECT count(DISTINCT person_id), count(*) FROM preliminary_ob
df=pd.DataFrame(curs.fetchall())
df.columns = ['invd person_ids','total rows']
df.iloc[:,:]
```

Out[318]:

	invd person_ids	total rows
0	45087	64938

Remove duplicate rows from observation_minus_na_egas and inspect the new dimensions

In [319…
```
curs.execute("CREATE TABLE observation_working_set AS \
                SELECT DISTINCT * FROM preliminary_observation_2")
```

In [320…
```
curs.execute("SELECT count(DISTINCT person_id), count(*) FROM (\
                SELECT DISTINCT * FROM observation_working_set) a")
df=pd.DataFrame(curs.fetchall())
df.columns = ['invd person_ids','total rows']
df.iloc[:,:]
```

Out[320]:

	invd person_ids	total rows
0	45087	60256

In [351…
```
curs.execute("SELECT * FROM observation_working_set LIMIT 10")
df=pd.DataFrame(curs.fetchall())
```

```
df.columns = ['person_ids','most recent observation', 'ega', 'most recent de
df.iloc[:,:]
```

Out[351]:

	person_ids	most recent observation	ega	most recent delivery
0	[redacted]	2021-03-29	38 5	2021-03-28
1		2020-11-16	37 3	2020-11-16
2		2021-05-13	37 1	2021-05-13
3		2017-06-07	39	2017-06-07
4		2017-07-19	40 5	2017-07-19
5		2018-01-15	37	2018-01-15
6		2021-02-15	41	2021-02-15
7		2020-12-16	38 5	2020-12-16
8		2021-06-06	40 4	2021-06-06
9		2020-11-22	39 3	2020-11-22

Working with empty ega entries

Save blank ega entries to separate table

In [321…

```
curs.execute("CREATE TABLE observation_blank_ega AS \
                SELECT * FROM observation_working_set WHERE ega = ' '")
```

Remove blank entries from observation_working_set and inspect dimensions

In [322…

```
curs.execute("CREATE TABLE observation_working_set_2 AS \
                SELECT * FROM observation_working_set WHERE ega != ' '")
```

In [323…

```
curs.execute("SELECT count(DISTINCT person_id), count(*) FROM observation_wo
df=pd.DataFrame(curs.fetchall())
df.columns = ['invd person_ids','total rows']
df.iloc[:,:]
```

Out[323]:

	invd person_ids	total rows
0	44856	59852

Calculate periods for assigning cohorts using observation dates and egas

---

Calculating periods using observation date per WQW. Assuming that observation date ~ delivery date.

In [ ]:
```
curs.execute("CREATE TABLE observation_with_periods AS \
                SELECT * FROM ( \
                SELECT person_id, most_recent_observation, ega, \
                ISNULL(CAST(regexp_extract(ega,'^[0-9]+') AS int),0)*7 AS eg
                ISNULL(CAST(regexp_extract(ega,' [0-9]') AS int),0) AS ega_d
                most_recent_observation - (ega_weeks_in_days + ega_days + 7*
                most_recent_observation - (ega_weeks_in_days + ega_days) AS
                delivery, \
                delivery + 365 AS postpartum \
                FROM observation_working_set_2) a")
```

In [325…
```
curs=conn.cursor()
curs.execute("SELECT count(DISTINCT person_id), count(*) FROM observation_wi
df=pd.DataFrame(curs.fetchall())
df.columns = ['invd person_ids','total rows']
df.iloc[:,:]
```

Out[325]:

	invd person_ids	total rows
0	44856	59852

In [334…
```
curs.execute("SELECT * FROM observation_with_periods LIMIT 10")
df=pd.DataFrame(curs.fetchall())
df.columns = ['person_id','observation_date','ega','egas_weeks_in_days','ega
              'prepartum start','etoc','delivey','postpartum end']
df.iloc[:,:]
```

Out[334]:

	person_id	observation_date	ega	egas_weeks_in_days	egas_days	prepartum start
0	[redacted]	2021-03-29	385	266	5	2020-03-11
1	[redacted]	2020-11-16	373	259	3	2019-11-08
2	[redacted]	2021-05-13	371	259	1	2020-05-06
3	[redacted]	2017-06-07	39	273	0	2016-05-18
4	[redacted]	2017-07-19	405	280	5	2016-06-17
5	[redacted]	2018-01-15	37	259	0	2017-01-09
6	[redacted]	2021-02-15	41	287	0	2020-01-13
7	[redacted]	2020-12-16	385	266	5	2019-11-29
8	[redacted]	2021-06-06	404	280	4	2020-05-06
9	[redacted]	2020-11-22	393	273	3	2019-10-31

Create table containing the set of drugs identified in the drug exposure table using the drugs of interest

mprint1_druglist was created earlier (refer to 00_mprint_project1_....ipynb); used bash to read text file with manual cleanup

In [160…

```
sql="SELECT * FROM drug_exposure_results LIMIT 10"
df = get_df_from_db(sql)
df.iloc[:,:]
```

Out[160]:

	PERSON_ID	DRUG_EXPOSURE_START_DATE	DRUG_SOURCE_VALUE
0	[redacted]	2013-06-28	atenolol
1	[redacted]	2009-01-09	carvedilol
2	[redacted]	2009-01-24	esomeprazole
3	[redacted]	2014-11-02	aripiprazole
4	[redacted]	2012-12-19	hydromorphone
5	[redacted]	2016-02-16	clopidogrel
6	[redacted]	2013-02-25	tamsulosin
7	[redacted]	2005-03-07	aspirin
8	[redacted]	2015-10-11	tacrolimus
9	[redacted]	2003-07-30	amitriptyline

In [ ]:
```
curs.execute("CREATE TABLE drug_exposure_results AS \
                SELECT a.person_id,drug_exposure_start_date,a.drug_source_va
                FROM v_drug_exposure a \
                INNER JOIN mprint1_druglist b ON lower(a.drug_source_value)
                LIKE '%' || lower(b.drug_names) || '%'")
```

In [7]:
```
curs.execute("CREATE TABLE observation_drug_results AS \
                SELECT * FROM drug_exposure_results \
                WHERE person_id IN ( SELECT person_id FROM observation_with_
```

In [12]:
```
curs.execute("CREATE TABLE preliminary_observation_cohorts AS \
SELECT *  FROM ( \
SELECT person_id, drug_name, ISNULL(period_index,0) AS period_index FROM ( \
SELECT *, CASE \
        WHEN a.drug_exposure_start_date BETWEEN a.prepartum AND a.etoc THEN
        WHEN a.drug_exposure_start_date BETWEEN a.etoc AND a.delivery  THEN
    END AS period_index \
FROM ( \
    SELECT a.person_id, a.prepartum,a.etoc,a.delivery,a.postpartum,\
    b.drug_exposure_start_date, b.drug_source_value AS drug_name \
    FROM observation_with_periods a \
    INNER JOIN observation_drug_results b \
    ON a.person_id = b.person_id) a \
) b \
) c \
WHERE period_index > 0")
```

In [13]:
```
sql="SELECT * FROM preliminary_observation_cohorts LIMIT 20"
df = get_df_from_db(sql)
df.iloc[:,:]
```

Out[13]:

	PERSON_ID	DRUG_NAME	PERIOD_INDEX
0	[redacted]	labetalol	2
1	[redacted]	ibuprofen	2
2	[redacted]	ibuprofen	2
3	[redacted]	morphine	2
4	[redacted]	lidocaine	2
5	[redacted]	ondansetron	2
6	[redacted]	lidocaine	2
7	[redacted]	labetalol	2
8	[redacted]	lidocaine	2
9	[redacted]	oxycodone	2
10	[redacted]	propylthiouracil	2
11	[redacted]	hydromorphone	2
12	[redacted]	fentanyl	2
13	[redacted]	ondansetron	2
14	[redacted]	fentanyl	2
15	[redacted]	ondansetron	2
16	[redacted]	ibuprofen	2
17	[redacted]	ibuprofen	2
18	[redacted]	ondansetron	2
19	[redacted]	ondansetron	2

Observation: SD counts

In [18]:
```
curs.execute("CREATE TABLE  observation_sd_counts AS \
SELECT drug_source_value AS drug_name, \
count(DISTINCT person_id) AS sd_counts \
FROM observation_drug_results \
GROUP BY drug_source_value \
ORDER BY sd_counts desc")
```

In [20]:
```
sql="SELECT count(DISTINCT person_id) AS id_counts, \
             count(DISTINCT drug_source_value) AS drug_counts, \
             count(*) AS total_count FROM observation_drug_results"
df = get_df_from_db(sql)
df.iloc[:,:]
```

Out[20]:

	ID_COUNTS	DRUG_COUNTS	TOTAL_COUNT
0	44848	198	6860824

In [22]:
```
sql="SELECT * FROM observation_sd_counts ORDER BY sd_counts desc LIMIT 20"
df = get_df_from_db(sql)
df.iloc[:,:]
```

Out[22]:

	DRUG_NAME	SD_COUNTS
0	ibuprofen	44333
1	lidocaine	43395
2	ondansetron	40665
3	fentanyl	36441
4	oxycodone	29331
5	hydrocodone	21111
6	morphine	19837
7	naloxone	18568
8	nitrofurantoin	9013
9	hydromorphone	8736
10	midazolam	7837
11	remifentanil	7463
12	metoclopramide	6260
13	sertraline	5839
14	haloperidol	5764
15	gentamicin	5583
16	omeprazole	5309
17	aspirin	4667
18	labetalol	4519
19	naproxen	4221

Observation: BioVu counts

In [ ]:
```
curs.execute("CREATE TABLE observation_biovu_counts AS \
              SELECT drug_source_value AS drug_name, count(DISTINCT person_id)
              FROM observation_drug_results  WHERE person_id IN( SELECT persor
              GROUP BY drug_source_value \
              ORDER BY biovu_counts desc")
```

In [143…
```
sql="SELECT count(DISTINCT person_id), count(*) FROM observation_drug_result
```

```
df = get_df_from_db(sql)
df.iloc[:,:]
```

Out[143]:

	COUNT	COUNT
0	14334	2935259

In [30]:
```
sql="SELECT * FROM observation_biovu_counts ORDER BY biovu_counts desc LIMIT
df = get_df_from_db(sql)
df.iloc[:,:]
```

Out[30]:

	DRUG_NAME	BIOVU_COUNTS
0	ibuprofen	14162
1	lidocaine	13961
2	ondansetron	12985
3	fentanyl	11837
4	oxycodone	10767
5	hydrocodone	7539
6	morphine	6814
7	naloxone	5220
8	hydromorphone	3510
9	nitrofurantoin	3350
10	midazolam	3273
11	remifentanil	3250
12	metoclopramide	2609
13	sertraline	2429
14	omeprazole	2373
15	ciprofloxacin	2021
16	aspirin	2007
17	gentamicin	1939
18	naproxen	1885
19	haloperidol	1815

Observation: MEGA counts

In [38]:
```
curs.execute("drop table observation_mega_counts")
```

In [39]:
```
curs.execute("CREATE TABLE observation_mega_counts AS \
                SELECT drug_source_value AS drug_name, count(DISTINCT persor
```

```
                FROM observation_drug_results WHERE person_id \
                IN( SELECT person_id FROM v_genotype_results_assay WHERE ass
                GROUP BY drug_source_value \
                ORDER BY mega_counts desc")
```

In [142…
```
sql="SELECT count(DISTINCT person_id), count(*) FROM observation_drug_result
                IN( SELECT person_id FROM v_genotype_results_assay WHERE ass
df = get_df_from_db(sql)
df.iloc[:,:]
```

Out[142]:

	COUNT	COUNT
0	4432	1256936

In [40]:
```
sql="SELECT * FROM observation_mega_counts ORDER BY mega_counts desc LIMIT 2
df = get_df_from_db(sql)
df.iloc[:,:]
```

Out[40]:

	DRUG_NAME	MEGA_COUNTS
0	ibuprofen	4378
1	lidocaine	4301
2	ondansetron	4014
3	fentanyl	3714
4	oxycodone	3589
5	hydrocodone	2704
6	morphine	2409
7	remifentanil	1467
8	hydromorphone	1410
9	naloxone	1406
10	nitrofurantoin	1294
11	midazolam	1282
12	metoclopramide	1063
13	omeprazole	893
14	ciprofloxacin	857
15	sertraline	822
16	naproxen	786
17	sulfamethoxazole / trimethoprim	782
18	aspirin	762
19	gentamicin	735

Joining the observation_*_counts tables

In [44]:
```
curs.execute("CREATE TABLE observation_sd_biovu_mega AS \
                SELECT a.*, ISNULL(b.biovu_counts,0) AS biovu_counts, \
                ISNULL(c.mega_counts,0) AS mega_counts FROM observation_sd_c
                LEFT JOIN observation_biovu_counts B ON a.drug_name = b.drug
                LEFT JOIN observation_mega_counts c ON a.drug_name = c.drug_
                ORDER BY a.drug_name asc")
```

In [49]:
```
curs.execute("SELECT * FROM observation_sd_biovu_mega ORDER BY drug_name asc
df = get_df_from_db(sql)
df.iloc[:,:]
```

Out[49]:

	DRUG_NAME	MEGA_COUNTS
0	ibuprofen	4378
1	lidocaine	4301
2	ondansetron	4014
3	fentanyl	3714
4	oxycodone	3589
5	hydrocodone	2704
6	morphine	2409
7	remifentanil	1467
8	hydromorphone	1410
9	naloxone	1406
10	nitrofurantoin	1294
11	midazolam	1282
12	metoclopramide	1063
13	omeprazole	893
14	ciprofloxacin	857
15	sertraline	822
16	naproxen	786
17	sulfamethoxazole / trimethoprim	782
18	aspirin	762
19	gentamicin	735

Calculating cohort 1 and cohort 2

In [74]:
```
curs.execute("CREATE TABLE preliminary_observation_cohorts_index1 AS \
```

```
                SELECT DISTINCT * FROM preliminary_observation_cohorts \
                WHERE period_index = 1")
```

In [ ]:
```
# curs.execute("SELECT count(DISTINCT person_id) FROM preliminary_observatio
# df=pd.DataFrame(curs.fetchall())
# df.columns = ['id counts']
# df.iloc[:,:]
```

In [75]:
```
curs.execute("CREATE TABLE preliminary_observation_cohorts_index2 AS \
                SELECT DISTINCT * FROM preliminary_observation_cohorts \
                WHERE period_index = 2")
```

In [ ]:
```
# sql="SELECT count(DISTINCT person_id) FROM preliminary_observation_cohorts
# df = get_df_from_db(sql)
# df.iloc[:,:]
```

Calculating cohort 2

In [77]:
```
curs.execute("CREATE TABLE temp_observation_cohort2 AS \
                SELECT * FROM ( \
                SELECT a.person_id, a.drug_name, a.period_index AS period_1,
                b.period_index AS period_2 \
                FROM preliminary_observation_cohorts_index1 a \
                INNER JOIN preliminary_observation_cohorts_index2 b \
                ON a.person_id = b.person_id AND a.drug_name = b.drug_name)
```

In [132…
```
sql="SELECT count(DISTINCT person_id), count(DISTINCT drug_name), count(*) F
df = get_df_from_db(sql)
df.iloc[:,:]
```

Out[132]:

	COUNT	COUNT	COUNT
0	4336	140	9834

In [82]:
```
sql="SELECT * FROM temp_observation_cohort2 LIMIT 20"
df = get_df_from_db(sql)
df.iloc[:,:]
```

Out[82]:

	PERSON_ID	DRUG_NAME	PERIOD_1	PERIOD_2
0	[redacted]	sertraline	1	2
1	[redacted]	gentamicin	1	2
2	[redacted]	tacrolimus	1	2
3	[redacted]	ondansetron	1	2
4	[redacted]	escitalopram	1	2
5	[redacted]	amitriptyline	1	2
6	[redacted]	lansoprazole	1	2
7	[redacted]	ibuprofen	1	2
8	[redacted]	sertraline	1	2
9	[redacted]	ydromorphone	1	2
10	[redacted]	oxycodone	1	2
11	[redacted]	lidocaine	1	2
12	[redacted]	ondansetron	1	2
13	[redacted]	morphine	1	2
14	[redacted]	citalopram	1	2
15	[redacted]	omeprazole	1	2
16	[redacted]	hydrocodone	1	2
17	[redacted]	oxycodone	1	2
18	[redacted]	ibuprofen	1	2
19	[redacted]	hydrocodone	1	2

Calculating cohort 1

In [80]:
```
curs.execute("CREATE TABLE temp_observation_cohort1 AS \
                SELECT a.* FROM preliminary_observation_cohorts_index2  a \
                LEFT JOIN temp_observation_cohort2 b \
                ON (a.person_id = b.person_id \
                AND a.drug_name = b.drug_name AND a.period_index = b.period_
                WHERE b.person_id IS NULL")
```

In [ ]:
```
# sql="SELECT count(DISTINCT person_id), count(DISTINCT drug_name), count(*)
# df = get_df_from_db(sql)
# df.iloc[:,:]
```

In [135…
```
sql="SELECT * FROM temp_observation_cohort1 LIMIT 20"
df = get_df_from_db(sql)
df.iloc[:,:]
```

Out[135]:

	DRUG_NAME	COUNT
0	glipizide	117
1	l-methylfolate	8
2	tamoxifen	7
3	mercaptopurine	14
4	efavirenz	1
5	nicotine	1347
6	haloperidol	3756
7	propafenone	7
8	diclofenac	177
9	valproic acid	11
10	atazanavir	73
11	fluvoxamine	13
12	celecoxib	43
13	hydrocodone	13482
14	isoniazid	14
15	oxazepam	4
16	doxorubicin	11
17	methylphenidate	52
18	sertraline	1773
19	paliperidone	3

Calculating cohort 2 SD

In [84]:
```
curs.execute("CREATE TABLE observation_sd_cohort2 as \
SELECT drug_name, count(DISTINCT person_id) FROM temp_observation_cohort2 \
GROUP BY drug_name")
```

In [150…
```
sql="SELECT count(DISTINCT person_id), count(DISTINCT drug_name), count(*) F
df = get_df_from_db(sql)
df.iloc[:,:]
```

Out[150]:

	COUNT	COUNT	COUNT
0	4336	140	9834

In [86]:
```
sql="SELECT * FROM observation_sd_cohort2 ORDER BY count desc LIMIT 20"
df = get_df_from_db(sql)
df.iloc[:,:]
```

Out[86]:

	DRUG_NAME	COUNT
0	ibuprofen	1673
1	ondansetron	1294
2	lidocaine	1009
3	fentanyl	621
4	oxycodone	611
5	hydrocodone	478
6	sertraline	351
7	morphine	258
8	omeprazole	208
9	nitrofurantoin	184
10	citalopram	183
11	naloxone	182
12	escitalopram	142
13	naproxen	127
14	labetalol	122
15	aspirin	121
16	fluoxetine	111
17	bupropion	106
18	hydromorphone	84
19	ciprofloxacin	77

Calculating cohort 2 BioVU

In [88]:
```
curs.execute("CREATE TABLE observation_biovu_cohort2 AS \
SELECT drug_name, count(DISTINCT person_id) FROM temp_observation_cohort2 \
WHERE person_id IN( SELECT person_id FROM v_biovu_inds ) \
GROUP BY drug_name")
```

In [151…
```
sql="SELECT count(DISTINCT person_id), count(DISTINCT drug_name), count(*) F
WHERE person_id IN( SELECT person_id FROM v_biovu_inds )"
df = get_df_from_db(sql)
df.iloc[:,:]
```

Out[151]:

	COUNT	COUNT	COUNT
0	2061	126	4706

In [89]:
```
sql="SELECT * FROM observation_biovu_cohort2 ORDER BY count desc LIMIT 20"
df = get_df_from_db(sql)
df.iloc[:,:]
```

Out[89]:

	DRUG_NAME	COUNT
0	ibuprofen	792
1	ondansetron	575
2	lidocaine	426
3	oxycodone	288
4	fentanyl	264
5	hydrocodone	242
6	sertraline	177
7	morphine	126
8	omeprazole	119
9	nitrofurantoin	91
10	citalopram	88
11	aspirin	75
12	naproxen	71
13	escitalopram	71
14	bupropion	60
15	labetalol	56
16	fluoxetine	55
17	ciprofloxacin	47
18	naloxone	46
19	hydromorphone	44

Calculating cohort 2 MEGA

In [90]:
```
curs.execute("drop table observation_mega_cohort2")
```

In [91]:
```
curs.execute("CREATE TABLE observation_mega_cohort2 AS \
SELECT drug_name, count(DISTINCT person_id) FROM temp_observation_cohort2 \
WHERE person_id IN( SELECT person_id FROM v_genotype_results_assay WHERE ass
GROUP BY drug_name")
```

In [154…
```
sql="SELECT count(DISTINCT person_id), count(DISTINCT drug_name), count(*) F
WHERE person_id IN( SELECT person_id FROM v_genotype_results_assay WHERE ass
df = get_df_from_db(sql)
df.iloc[:,:]
```

Out[154]:

	COUNT	COUNT	COUNT
0	813	105	2109

In [92]:
```
sql="SELECT * FROM observation_mega_cohort2 ORDER BY count desc LIMIT 20"
df = get_df_from_db(sql)
df.iloc[:,:]
```

Out[92]:

	DRUG_NAME	COUNT
0	ibuprofen	362
1	ondansetron	242
2	lidocaine	185
3	oxycodone	136
4	hydrocodone	121
5	fentanyl	116
6	morphine	68
7	sertraline	62
8	omeprazole	59
9	citalopram	34
10	nitrofurantoin	32
11	naproxen	32
12	aspirin	28
13	escitalopram	28
14	hydromorphone	28
15	labetalol	25
16	ciprofloxacin	23
17	bupropion	22
18	naloxone	20
19	hydroxychloroquine	18

Calculating cohort 1 SD

In [95]:
```
curs.execute("CREATE TABLE observation_sd_cohort1 AS \
SELECT a.person_id, a.drug_name FROM preliminary_observation_cohorts_index2
LEFT JOIN preliminary_observation_cohorts_index1 b USING (person_id, drug_na
WHERE b.period_index IS NULL ")
```

In [115…
```
sql="SELECT * FROM observation_sd_cohort1 LIMIT 20"
df = get_df_from_db(sql)
```

```
df.iloc[:,:]
```

Out[115]:

	PERSON_ID	DRUG_NAME
0	[redacted]	ondansetron
1	[redacted]	oxycodone
2	[redacted]	labetalol
3	[redacted]	ondansetron
4	[redacted]	fentanyl
5	[redacted]	nitrofurantoin
6	[redacted]	ibuprofen
7	[redacted]	oxycodone
8	[redacted]	lidocaine
9	[redacted]	nitrofurantoin
10	[redacted]	fentanyl
11	[redacted]	lansoprazole
12	[redacted]	nitrofurantoin
13	[redacted]	omeprazole
14	[redacted]	nitrofurantoin
15	[redacted]	fentanyl
16	[redacted]	ondansetron
17	[redacted]	fentanyl
18	[redacted]	labetalol
19	[redacted]	lidocaine

In [136…

```
sql="SELECT count(DISTINCT person_id), count(DISTINCT drug_name), count(*) F
df = get_df_from_db(sql)
df.iloc[:,:]
```

Out[136]:

	COUNT	COUNT	COUNT
0	44725	169	274331

In [106…

```
curs.execute("CREATE TABLE temp_observation_cohort1 as \
SELECT drug_name, count(DISTINCT person_id) FROM observation_sd_cohort1 \
GROUP BY drug_name")
```

In [108…

```
sql="SELECT * FROM temp_observation_cohort1 ORDER BY count desc LIMIT 20"
df = get_df_from_db(sql)
df.iloc[:,:]
```

Out[108]:

	DRUG_NAME	COUNT
0	lidocaine	39989
1	ondansetron	37732
2	ibuprofen	36654
3	fentanyl	33363
4	oxycodone	23336
5	naloxone	17113
6	morphine	16023
7	hydrocodone	13482
8	nitrofurantoin	5284
9	gentamicin	4276
10	haloperidol	3756
11	remifentanil	3382
12	labetalol	3189
13	hydromorphone	2867
14	metoclopramide	2729
15	aspirin	2628
16	omeprazole	2149
17	sertraline	1773
18	mirtazapine	1712
19	midazolam	1617

Calculating cohort 1 BioVU

In [116…

```
curs.execute("CREATE TABLE observation_biovu_cohort1 as \
SELECT drug_name, count(DISTINCT person_id) FROM observation_sd_cohort1 \
WHERE person_id IN( SELECT person_id FROM v_biovu_inds ) \
GROUP BY drug_name")
```

In [117…

```
sql="SELECT * FROM observation_biovu_cohort1 ORDER BY count desc LIMIT 20"
df = get_df_from_db(sql)
df.iloc[:,:]
```

Out[117]:

	DRUG_NAME	COUNT
0	lidocaine	12590
1	ibuprofen	11822
2	ondansetron	11716
3	fentanyl	10482
4	oxycodone	8629
5	morphine	5111
6	naloxone	4679
7	hydrocodone	4526
8	nitrofurantoin	1828
9	gentamicin	1354
10	remifentanil	1270
11	aspirin	1005
12	labetalol	989
13	metoclopramide	989
14	haloperidol	980
15	hydromorphone	948
16	omeprazole	836
17	sertraline	680
18	glibenclamide	589
19	mirtazapine	453

In [145…
```
sql="SELECT count(DISTINCT person_id), count(DISTINCT drug_name), count(*) F
WHERE person_id IN( SELECT person_id FROM v_biovu_inds )"
df = get_df_from_db(sql)
df.iloc[:,:]
```

Out[145]:

	COUNT	COUNT	COUNT
0	14300	158	89307

Calculating cohort 1 MEGA

In [118…
```
curs.execute("drop table observation_mega_cohort1")
```

In [119…
```
curs.execute("CREATE TABLE observation_mega_cohort1 as \
SELECT drug_name, count(DISTINCT person_id) FROM observation_sd_cohort1 \
WHERE person_id IN( SELECT person_id FROM v_genotype_results_assay WHERE ass
GROUP BY drug_name")
```

In [148…
```
sql="SELECT count(DISTINCT person_id), count(DISTINCT drug_name), count(*) F
WHERE person_id IN( SELECT person_id FROM v_genotype_results_assay WHERE ass
df = get_df_from_db(sql)
df.iloc[:,:]
```

Out[148]:

	COUNT	COUNT	COUNT
0	4423	141	28387

In [120…
```
sql="SELECT * FROM observation_mega_cohort1 ORDER BY count desc LIMIT 20"
df = get_df_from_db(sql)
df.iloc[:,:]
```

Out[120]:

	DRUG_NAME	COUNT
0	lidocaine	3712
1	ibuprofen	3633
2	ondansetron	3499
3	fentanyl	3154
4	oxycodone	2827
5	morphine	1666
6	hydrocodone	1557
7	naloxone	1156
8	nitrofurantoin	709
9	remifentanil	593
10	gentamicin	447
11	labetalol	393
12	metoclopramide	388
13	hydromorphone	376
14	aspirin	365
15	omeprazole	290
16	haloperidol	221
17	glibenclamide	220
18	sertraline	201
19	nicotine	149

In [ ]:
```
curs.execute("CREATE TABLE observation_summary_table AS \
SELECT a.*, ISNULL(b.biovu_counts,0) AS biovu_counts, ISNULL(c.mega_counts,0
ISNULL(d.count,0) AS cohort1_sd_counts, ISNULL(e.count,0) AS cohort1_biovu_c
ISNULL(g.count,0) AS cohort2_sd_counts, ISNULL(h.count,0) AS cohort2_biovu_c
FROM observation_sd_counts a \
```

```
LEFT JOIN observation_biovu_counts b ON a.drug_name = b.drug_name \
LEFT JOIN observation_mega_counts c ON a.drug_name = c.drug_name \
LEFT JOIN temp_observation_cohort1 d ON a.drug_name = d.drug_name \
LEFT JOIN observation_biovu_cohort1 e ON a.drug_name = e.drug_name \
LEFT JOIN observation_mega_cohort1 f ON a.drug_name = f.drug_name \
LEFT JOIN observation_sd_cohort2 g ON a.drug_name = g.drug_name \
LEFT JOIN observation_biovu_cohort2 h ON a.drug_name = h.drug_name \
LEFT JOIN observation_mega_cohort2 i ON a.drug_name = i.drug_name \
ORDER BY a.drug_name asc")
```

In [ ]:
```
curs.execute("CREATE TABLE summary_table_index_counts AS \
SELECT *, \
CASE \
        WHEN a.sd_counts >= a.biovu_counts THEN 0 \
        ELSE 1 \
        END AS sd_biovu_count_index,b \
CASE \
        WHEN a.biovu_counts >= mega_counts THEN 0 \
        ELSE 1 \
        END AS biovu_mega_count_index, \
CASE \
        WHEN a.cohort1_sd_counts >= cohort1_biovu_counts THEN 0 \
        ELSE 1 \
        END AS cohort1_sd_biovu_count_index, \
CASE \
        WHEN a.cohort1_biovu_counts >= cohort1_mega_counts THEN 0 \
        ELSE 1 \
        END AS cohort1_biovu_mega_count_index, \
CASE \
        WHEN a.cohort2_sd_counts >= cohort2_biovu_counts THEN 0 \
        ELSE 1 \
        END AS cohort2_sd_biovu_count_index, \
CASE \
        WHEN a.cohort2_biovu_counts >= cohort2_mega_counts THEN 0 \
        ELSE 1 \
        END AS cohort2_biovu_mega_count_index \
FROM observation_summary_table a")
```

In [ ]:
```
sql="SELECT sum(a.sd_biovu_count_index) AS sd_biovu_sum, \
sum(a.biovu_mega_count_index) AS biovu_mega_sum, \
sum(a.cohort1_sd_biovu_count_index) AS cohort1_sd_biovu_sum, \
sum(a.cohort1_biovu_mega_count_index) AS cohort1_biovu_mega_sum, \
sum(a.cohort2_sd_biovu_count_index) AS cohort2_sd_biovu_sum, \
sum(a.cohort2_biovu_mega_count_index) AS cohort2_biovu_mega_sum \
FROM summary_table_index_counts a"
df = get_df_from_db(sql)
df.iloc[:,:]
```

In [131…
```
sql="SELECT * FROM summary_table_index_counts WHERE biovu_mega_count_index =
df = get_df_from_db(sql)
df.iloc[:,:]
```

Out[131]:

	DRUG_NAME	SD_COUNTS	BIOVU_COUNTS	MEGA_COUNTS	COHORT1_SD_COUN
0	nilotinib	1	0	1	
1	paliperidone	29	7	9	

In [ ]:

In [ ]: