In [ ]:

```

```

In [1]:

```
import jaydebeapi, sys, os # allows connection to Netezza and executes :
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 [97]:

```
%run -i 'login.py'
```

```
Connection String: jdbc:netezza://ori-netezza.vumc.org:5480/DENNY_OMO
P_RD
('2022-02-24 12:32:43',)

<Figure size 432x288 with 0 Axes>
```

In [ ]:

```
curs=conn.cursor()
curs.execute("SELECT COUNT(DISTINCT fact_id_2) FROM DENNY_OMOP_RD.dbo.v_
             WHERE relationship_concept_id = 4248584")
curs.fetchall()
```

In [ ]:

```
# Create table using relationship_concept_id = 4248584 from the V_FACT_I
# This will create a 2 x n list of all mom-baby pairs read as 'mom "moth
# There are 62275 unique mom baby pairs

curs=conn.cursor()
curs.execute("CREATE TABLE mom_baby_ids AS \
SELECT f.fact_id_1 AS mom_id, f.fact_id_2 AS child_id \
FROM v_fact_relationship f \
WHERE f.relationship_concept_id = 4248584")
curs.fetchall()
```

In [ ]:

```
#  Using the V_X_PERSON_PII table add mom first and last names, add baby

curs=conn.cursor()
curs.execute("CREATE TABLE mom_baby_ids_2 AS \
SELECT m.mom_id, m.child_id, x.x_fname AS mom_fname, x.x_lname AS mom_ln
x1.x_lname AS child_lname \
FROM mom_baby_ids AS m \
INNER JOIN v_x_person_pii x ON m.mom_id = x.person_id \
INNER JOIN v_x_person_pii x1 ON m.child_id = x1.person_id")
curs.fetchall()
```

```
In [ ]: # Using the V_PERSON table add mom and baby genders

        curs=conn.cursor()
        curs.execute("CREATE TABLE mom_baby_ids_3 AS \
        SELECT m.mom_id, m.child_id, m.mom_fname, m.mom_lname, date(p.birth_date
        p.gender_source_value AS mom_gender, m.child_fname, m.child_lname, date
        p1.gender_source_value AS child_gender \
        FROM mom_baby_ids_2 m \
        INNER JOIN v_person p ON m.mom_id = p.person_id \
        INNER JOIN v_person p1 ON m.child_id = p1.person_id")
        curs.fetchall()
```

```
In [ ]: # Lastly, order the table by mom_ids: MBIS = 'MOM_BABY_IDs'

        curs=conn.cursor()
        curs.execute("CREATE TABLE MBIs AS SELECT m.* FROM mom_baby_ids_3 m ORDI
        curs.fetchall()
```

```
In [ ]: #  There are 62275 unique mom baby pairs

        curs=conn.cursor()
        curs.execute("SELECT count(DISTINCT m.child_id) FROM mom_baby_ids m")
        curs.fetchall()
```

MOM_FNAME	MOM_LNAME	MOM_DOB	MOM_GENDER	CHILD_FNAME	CHILD_LNAME	CHILD
[redacted]	[redacted]	1984-11-05	F	[redacted]	[redacted]	2015-
[redacted]	[redacted]	1984-12-27	F	[redacted]	[redacted]	2008-
[redacted]	[redacted]	1980-10-13	F	[redacted]	[redacted]	2014-
[redacted]	[redacted]	1986-03-17	F	[redacted]	[redacted]	2008-
[redacted]	[redacted]	1986-07-05	F	[redacted]	[redacted]	2021-
[redacted]	[redacted]	1993-02-18	F	[redacted]	[redacted]	2010-
[redacted]	[redacted]	1988-01-10	F	[redacted]	[redacted]	2011-
[redacted]	[redacted]	1987-03-01	F	[redacted]	[redacted]	2019-
[redacted]	[redacted]	1994-08-26	F	[redacted]	[redacted]	2016-
[redacted]	[redacted]	1993-11-11	F	[redacted]	[redacted]	2017-
[redacted]	[redacted]	1989-03-21	F	[redacted]	[redacted]	2010-
[redacted]	[redacted]	1980-03-04	F	[redacted]	[redacted]	2010-
[redacted]	[redacted]	1982-08-21	F	[redacted]	[redacted]	2011-
[redacted]	[redacted]	1983-02-12	F	[redacted]	[redacted]	2007-
[redacted]	[redacted]	1990-09-20	F	[redacted]	[redacted]	2018-
[redacted]	[redacted]	1986-05-18	F	[redacted]	[redacted]	2012-
[redacted]	[redacted]	1977-01-17	F	[redacted]	[redacted]	2010-

MOM_FNAME	MOM_LNAME	MOM_DOB	MOM_GENDER	CHILD_FNAME	CHILD_LNAME	CHILD
[redacted]						

In [ ]:

```
# THE FOLLOWING CONTROL IS A VALIDATION OF THE SCRIPT ITSELF

curs=conn.cursor()
curs.execute("CREATE TABLE testSet AS \
SELECT v.person_id, v.entry_date, v.field_name, v.field_value \
FROM v_x_labor_and_delivery v \
WHERE ((v.field_name LIKE 'Baby Name One' OR v.field_name LIKE 'Infant I
OR v.field_name LIKE 'Infant Gender 1') AND v.person_id = 11821577) ORDI
curs.fetchall()
```

PERSON_ID	ENTRY_DATE	FIELD_NAME	FIELD_VALUE
[redacted]	2014-07-01 12:14:00.000	Infant Gender 1	[redacted]
[redacted]	2014-07-01 12:14:00.000	Baby Name One	[redacted]
[redacted]	2014-07-01 12:14:00.000	Infant Delivery Date Time 1	[redacted]
[redacted]	2016-04-04 22:13:00.000	Infant Gender 1	[redacted]
[redacted]	2016-04-04 22:13:00.000	Infant Delivery Date Time 1	[redacted]
[redacted]	2016-04-04 22:13:00.000	Baby Name One	[redacted]
[redacted]	2017-07-28 09:01:00.000	Baby Name One	[redacted]
[redacted]	2017-07-28 09:01:00.000	Infant Gender 1	[redacted]
[redacted]	2017-07-28 09:01:00.000	Infant Delivery Date Time 1	[redacted]

In [ ]:

```
# Create a second test table from MBIs for retieving baby name, baby gei

curs=conn.cursor()
curs.execute("CREATE TABLE mTest AS SELECT * FROM mom_baby_ids_4 m WHERI
curs.fetchall()
```

MOM_FNAME	MOM_LNAME	MOM_DOB	MOM_GENDER	CHILD_FNAME	CHILD_LNAME	CHILD
[redacted]	[redacted]	1987-11-17	F	[redacted]	[redacted]	2014-
[redacted]	[redacted]	1987-11-17	F	[redacted]	[redacted]	2017-
[redacted]	[redacted]	1987-11-17	F	[redacted]	[redacted]	2016-

In [ ]:
```
# Check to see if the delivery date from the testSet matches the child_
# If so, check against MBIs
# The script:

curs=conn.cursor()
curs.execute("SELECT m.*, date(t.field_value) AS child_name_lnd \
FROM mTest m \
INNER JOIN testSet t ON m.child_dob = t.field_value \
WHERE (t.field_name LIKE 'Infant Delivery Date Time 1' AND m.mom_id = t
curs.fetchall()
```

MOM_FNAME	MOM_LNAME	MOM_DOB	MOM_GENDER	CHILD_FNAME	CHILD_LNAME	CHILD
[redacted]	[redacted]	1987-11-17	F	[redacted]	[redacted]	2014-
[redacted]	[redacted]	1987-11-17	F	[redacted]	[redacted]	2016-
[redacted]	[redacted]	1987-11-17	F	[redacted]	[redacted]	2017-

In [ ]:
```
# Create table including columns for entry_date and child DOB from V_X_

curs=conn.cursor()
curs.execute("CREATE TABLE MBIs_1 AS \
SELECT v.entry_date, m.*, date(v.field_value) AS child_dob_lnd \
FROM MBIs m \
INNER JOIN v_x_labor_and_delivery v ON m.child_dob = v.field_value \
WHERE (v.field_name LIKE 'Infant Delivery Date Time 1' AND m.mom_id = v
curs.fetchall()
```

In [ ]:
```
# Create table to include column for child names from V_X_LABOR_AND_DEL

curs=conn.cursor()
curs.execute("CREATE TABLE MBIs_2 AS \
SELECT m.*, v.field_value AS child_name_lnd \
FROM mbis_1 m \
INNER JOIN v_x_labor_and_delivery v ON m.entry_date = v.entry_date \
WHERE (v.field_name LIKE 'Baby Name One' AND m.mom_id = v.person_id)")
curs.fetchall()
```

In [ ]:
```
# Create table to include child gender from V_X_LABOR_AND_DELIVERY

curs=conn.cursor()
curs.execute("CREATE TABLE MBIs_3 AS \
SELECT m.*, v.field_value AS child_gender_lnd \
FROM mbis_2 m \
INNER JOIN v_x_labor_and_delivery v ON m.entry_date = v.entry_date \
WHERE (v.field_name LIKE 'Infant Gender 1' AND m.mom_id = v.person_id)"
curs.fetchall()
```

```
In [ ]: curs=conn.cursor()
        curs.execute("SELECT * FROM MBIs_3")
        df = pd.DataFrame(curs.fetchall())
        df.iloc[0:10,:]
```

MOM_FNAME	MOM_LNAME	MOM_DOB	MOM_GENDER	CHILD_FNAME	CHILD_LNAME	CHILD
[redacted]	[redacted]	1979-02-13	F	[redacted]	[redacted]	2013-
[redacted]	[redacted]	1985-06-09	F	[redacted]	[redacted]	2017-
[redacted]	[redacted]	1969-05-12	F	[redacted]	[redacted]	2012-
[redacted]	[redacted]	1988-01-08	F	[redacted]	[redacted]	2015-
[redacted]	[redacted]	1992-04-21	F	[redacted]	[redacted]	2013-
[redacted]	[redacted]	1978-03-04	F	[redacted]	[redacted]	2011-
[redacted]	[redacted]	1990-01-16	F	[redacted]	[redacted]	2017-
[redacted]	[redacted]	1988-04-13	F	[redacted]	[redacted]	2016-
[redacted]	[redacted]	1979-12-16	F	[redacted]	[redacted]	2013-
[redacted]	[redacted]	1988-06-22	F	[redacted]	[redacted]	2012-
[redacted]	[redacted]	1978-02-02	F	[redacted]	[redacted]	2012-
[redacted]	[redacted]	1991-10-11	F	[redacted]	[redacted]	2017-
[redacted]	[redacted]	1984-09-17	F	[redacted]	[redacted]	2013-
[redacted]	[redacted]	1979-07-27	F	[redacted]	[redacted]	2011-
[redacted]	[redacted]	1982-11-14	F	[redacted]	[redacted]	2013-
[redacted]	[redacted]	1994-08-16	F	[redacted]	[redacted]	2017-
[redacted]	[redacted]	1979-05-09	F	[redacted]	[redacted]	2011-
[redacted]	[redacted]	1983-07-29	F	[redacted]	[redacted]	2011-
[redacted]	[redacted]	1986-09-23	F	[redacted]	[redacted]	2016-
[redacted]	[redacted]	1984-06-15	F	[redacted]	[redacted]	2013-

Investigating mothers with multiple births for errors in "twins"-1

There are 46 mothers with 3 children

```
In [ ]: import pandas as pd

        curs=conn.cursor()
        curs.execute("SELECT m.mom_id \
        FROM   mbis_3 m \
        GROUP  BY m.mom_id \
        HAVING  count(*) = 3")
        df = pd.DataFrame(curs.fetchall())
        df.iloc[0:10,:]
```

Mother IDs with no errors in v_x_labor_and_delivery:

[redacted]

Mother IDs with errors in v_x_labor_and_delivery:

[redacted]

```
In [ ]: curs=conn.cursor()
        curs.execute("SELECT * FROM MBIs_3 m WHERE m.mom_id = 13421855 ORDER BY
        df = pd.DataFrame(curs.fetchall())
        df.iloc[0:3,:]
```

```
In [ ]: curs=conn.cursor()
        curs.execute("SELECT v.person_id, v.entry_date, v.field_value \
        FROM v_x_labor_and_delivery v \
        WHERE ((v.field_name LIKE 'Baby Name One' OR v.field_name LIKE 'Infant I
        OR v.field_name LIKE 'Infant Gender 1') AND v.person_id = 13421855) \
        ORDER BY v.person_id ASC, entry_date")
        df = pd.DataFrame(curs.fetchall())
        df.iloc[0:10,:]
```

Investigating mothers with multiple births for errors in "twins"-2

There are 3 mothers with 4 children

```
In [ ]: curs=conn.cursor()
        curs.execute("SELECT m.mom_id \
        FROM   mbis_3 m \
        GROUP  BY m.mom_id \
        HAVING  count(*) = 4")
        df = pd.DataFrame(curs.fetchall())
        df.iloc[:,:]
```

```
In [ ]: curs=conn.cursor()
        curs.execute("SELECT * FROM MBIs_3 m WHERE m.mom_id = 6326732 ORDER BY r
        df = pd.DataFrame(curs.fetchall())
        df.iloc[:,:]
```

```
In [ ]:
```

```
In [ ]: curs=conn.cursor()
        curs.execute("SELECT v.person_id, v.entry_date, v.field_value \
        FROM v_x_labor_and_delivery v \
        WHERE ((v.field_name LIKE 'Baby Name One' OR v.field_name LIKE 'Infant I
        OR v.field_name LIKE 'Infant Gender 1') AND v.person_id = 6326732) \
        ORDER BY v.person_id ASC, entry_date")
        df = pd.DataFrame(curs.fetchall())
        df.iloc[:,:]
```

```
In [ ]:
```

```
In [ ]: curs=conn.cursor()
        curs.execute("SELECT * FROM CO_CSV_COUNTS")
        CO_df = pd.DataFrame(curs.fetchall())
        CO_df.iloc[:,:]
```

```
In [ ]: curs=conn.cursor()
        curs.execute("SELECT * FROM OBS_CSV_COUNTS")
        CO_df = pd.DataFrame(curs.fetchall())
        CO_df.iloc[:,:]
```

```
In [ ]: curs=conn.cursor()
        curs.execute("SELECT * FROM MEAS_CSV_COUNTS")
        CO_df = pd.DataFrame(curs.fetchall())
        CO_df.iloc[:,:]
```

```
In [ ]: curs=conn.cursor()
        curs.execute("SELECT * FROM PO_CSV_COUNTS")
        CO_df = pd.DataFrame(curs.fetchall())
        CO_df.iloc[:,:]
```

```
In [ ]: curs=conn.cursor()
        curs.execute("SELECT * FROM codes_and_codeids")
        CO_df = pd.DataFrame(curs.fetchall())
        CO_df.iloc[:,:]
```

Create a table to hold the designated OUD codes: ICDCodes

Purpose: To filter tables by the specified OUD codes

NOTE: The original table was created in DBeaver. For future use the file path will need to be adjusted to reflect the location of the source file "ICD9-10_Codes.csv"

```
In [ ]: curs=conn.cursor()
        curs.execute("CREATE TABLE ICDCodes (code varchar(255), descript varchar
        curs.fetchall()

        curs=conn.cursor()
        curs.execute("INSERT INTO ICDCodes \
        SELECT * FROM \
        EXTERNAL '/Users/davidfoutch/Desktop/ICD9-10_Codes.csv' \
        USING \
        (DELIMITER ',' \
        SKIPROWS 1 \
        logdir '/Users/davidfoutch/Desktop/ICD9-10_Codes.csv' \
        ENCODING 'internal' \
        REMOTESOURCE 'JDBC' \
        ESCAPECHAR '\' \
        QUOTEDVALUE 'DOUBLE'))
        curs.fetchall()
```

To inspect the ICDCodes table run:

In [ ]:

```
curs=conn.cursor()
curs.execute("SELECT * FROM  ICDCodes LIMIT 10")
df=pd.DataFrame(curs.fetchall())
df.iloc[:,:]
```

Create a table to hold the OUD codes and their integer concept IDs

Purpose: To have a list where code concept IDs may potentially be useful where ICD9-10 codes are not listed in the table of interest

NOTE: This table is unnecessary as all tables of interest listed the OUD codes in a *_SOURCE_VALUE column.

In [ ]:

```
curs=conn.cursor()
curs.execute("CREATE TABLE codes_and_codeIDs AS \
SELECT c.concept_id,i.code \
FROM v_concept c  \
INNER JOIN icdcodes i ON i.code = c.concept_code \
SELECT * FROM codes_and_codeIDs")
curs.fetchall()
```

OUD code locations:

1. Create a table that reports the counts for each OUD code in the V_CONDITION_OCCURRENCE table

Purpose: To discover where OUD codes are located

In [ ]:

```
curs=conn.cursor()
curs.execute("CREATE TABLE CO_CSV_COUNTS AS \
SELECT c.condition_source_value, COUNT(*) \
FROM v_condition_occurrence c \
INNER JOIN codes_and_codeIDs i ON i.code = c.condition_source_value \
WHERE YEAR(c.condition_start_date) >= 2010 \
GROUP BY c.condition_source_value")
curs.fetchall()
```

To inspect the CO_CSV_COUNTS table run:

In [ ]:

```
curs=conn.cursor()
curs.execute("SELECT * FROM  CO_CSV_COUNTS LIMIT 10")
df=pd.DataFrame(curs.fetchall())
df.iloc[:,:]
```

2. Create a table that reports the counts for each OUD code in the V_OBSERVATION table

Purpose: To discover where OUD codes are located

In [ ]:

```
curs=conn.cursor()
curs.execute("CREATE TABLE OBS_CSV_COUNTS AS \
SELECT p.observation_source_value, COUNT(*) \
FROM v_observation p \
INNER JOIN codes_and_codeIDs i ON i.code = p.observation_source_value \
WHERE YEAR(p.observation_date) >= 2010 \
GROUP BY p.observation_source_value")
curs.fetchall()
```

To inspect the OBS_CSV_COUNTS table run:

In [ ]:

```
curs=conn.cursor()
curs.execute("SELECT * FROM OBS_CSV_COUNTS LIMIT 100")
df=pd.DataFrame(curs.fetchall())
df.iloc[:,:]
```

3. Create a table that reports the counts for each OUD code in the V_MEASUREMENT table

Purpose: To discover where OUD codes are located

```
In [ ]: curs=conn.cursor()
        curs.execute("CREATE TABLE MEAS_CSV_COUNTS AS \
        SELECT p.value_source_value, COUNT(*)value \
        FROM v_measurement p \
        INNER JOIN codes_and_codeIDs i ON i.code = p.value_source_value \
        WHERE YEAR(p.measurement_date) >= 2010 \
        GROUP BY p.value_source_value")
        curs.fetchall()
```

To inspect the MEAS_CSV_COUNTStable run:

```
In [ ]: curs=conn.cursor()
        curs.execute("SELECT * FROM MEAS_CSV_COUNTS LIMIT 10")
        df=pd.DataFrame(curs.fetchall())
        df.iloc[:,:]
```

4. Create a table that reports the counts for each OUD code in the V_PROCEDURE_OCCURRENCE table

Purpose: To discover where OUD codes are located

```
In [ ]: curs=conn.cursor()
        curs.execute("CREATE TABLE PO_CSV_COUNTS AS \
        SELECT p.procedure_source_value, COUNT(*) \
        FROM v_procedure_occurrence p \
        INNER JOIN codes_and_codeIDs i ON i.code = p.procedure_source_value \
        WHERE YEAR(p.procedure_date) >= 2010 \
        GROUP BY p.procedure_source_value")
        curs.fetchall()
```

To inspect the PO_CSV_COUNTStable run:

In [ ]:

```
curs=conn.cursor()
curs.execute("SELECT * FROM PO_CSV_COUNTS LIMIT 10")
df=pd.DataFrame(curs.fetchall())
df.iloc[:,:]
```

"Sanity Check":

NOTE: Considering the results from 1-4 above it may seem that atttention should be focused on the condition_occurrence and observation tables. It should be noted by far the most represented OUD code in the table is Z79.891 with 107983 appearances. The result implies that this table perhaps is best used to investigate the subset of cases where Z79.891 was assigned to persons. Furthermore, of the mother IDs derived from the fact_relationship table only 306 of these occur in the observation table. In other words, the observation table gives us scant new information with respect to person_ids with OUD entries condition_occurrence table.

Supporting evidence:

In [ ]:

```
curs=conn.cursor()
curs.execute("SELECT count(DISTINCT p.person_id)\
FROM v_observation p\
WHERE p.observation_source_value = 'Z79.891'")
curs.fetchall()  # 18194 unique person ids

curs=conn.cursor()
curs.execute("CREATE TABLE temp AS\
SELECT DISTINCT p.person_id\
FROM v_observation p\
WHERE p.observation_source_value = 'Z79.891'")
curs.fetchall()

curs=conn.cursor()
curs.execute("SELECT count(DISTINCT p.person_id)\
FROM temp p\
WHERE p.observation_source_value = 'Z79.891'")
# 306 unique mother ids that derive from the fact_relationship table
```

5. Create a table that reports the mom IDs appearing in the V_PROCEDURE_OCCURRENCE table

Purpose: To discover which mother IDs have OUD codes on or before the baby DOB

Condition_source table: This table is a subset of the condition_occurrence table. Records in this table are selected from v_condition_occurrence based on mother IDs (person_id).

```
In [ ]:
curs=conn.cursor()
curs.execute("CREATE TABLE condition_source AS \
SELECT m.mom_id,m.child_dob,c.condition_source_value,c.condition_start_
FROM mom_baby_ids_3 m \
INNER JOIN v_condition_occurrence c ON c.person_id = m.mom_id")
curs.fetchall()
```

To inspect the CONDITION_SOURCE table run:

```
In [ ]:
curs=conn.cursor()
curs.execute("SELECT * FROM CONDITION_SOURCE LIMIT 10")
df=pd.DataFrame(curs.fetchall())
df.iloc[:,:]
```

NOTE: There are fewer moms from the fact_relationship DB in the V_CONDITION_OCCURRENCE table than there are in the V_PERSON.

```
In [50]:
curs=conn.cursor()
curs.execute("SELECT count(DISTINCT mom_id) FROM condition_source") #47
curs.fetchall()
```

```
Out[50]: [(47599,)]
```

```
In [51]:
curs=conn.cursor()
curs.execute("SELECT count(DISTINCT mom_id) FROM mom_baby_ids") #47635
curs.fetchall()
```

```
Out[51]: [(47635,)]
```

6. Create a table that reports the mom IDs appearing in the

CONDITION_SOURCE table that are also assigned an OUD code

```
In [ ]: curs=conn.cursor()
        curs.execute("CREATE TABLE  oud_condition_source AS \
        SELECT c.mom_id,c.child_dob,c.condition_source_value,c.condition_start_
        FROM condition_source c \
        INNER JOIN codes_and_codeIDs p ON p.code = c.condition_source_value")
        curs.fetchall()
```

NOTE: There are 1754 moms with oud codes in the condition_occurrence table:

```
In [ ]: curs=conn.cursor()
        curs.execute("SELECT count(DISTINCT mom_id) FROM oud_condition_source")
        df=pd.DataFrame(curs.fetchall())
        df.iloc[:,:]
```

To inspect the OUD_CONDITION_SOURCE table run:

```
In [ ]: curs=conn.cursor()
        curs.execute("SELECT * FROM OUD_CONDITION_SOURCE LIMIT 10")
        df=pd.DataFrame(curs.fetchall())
        df.iloc[:,:]
```

"Sanity Check":

SC.1: Select all records where the date the condition is entered is strictly AFTER baby date of births

NOTE: There are 884 unique records.

```
In [ ]: curs=conn.cursor()
        curs.execute("CREATE TABLE greater_dob AS \
        SELECT * FROM oud_condition_source c WHERE YEAR(c.condition_start_date)
        curs.fetchall()
```

To inspect the GREATER_DOB table run:

```
In [ ]: curs=conn.cursor()
        curs.execute("SELECT * FROM GREATER_DOB LIMIT 10")
        df=pd.DataFrame(curs.fetchall())
        df.iloc[:,:]
```

SC.2: Select all records where the date the condition is entered is ON OR BEFORE baby date of births

NOTE: There are 1555 unique records.

```
In [ ]: curs=conn.cursor()
        curs.execute("CREATE TABLE lesser_dob AS \
        SELECT * FROM oud_condition_source c WHERE YEAR(c.condition_start_date)
        df=pd.DataFrame(curs.fetchall())
        df.iloc[:,:]
```

To inspect the LESSER_DOB table run:

```
In [ ]: curs=conn.cursor()
        curs.execute("SELECT * FROM LESSER_DOB LIMIT 10")
        df=pd.DataFrame(curs.fetchall())
        df.iloc[:,:]
```

SC.3: Compare the unique mom ids in the three groups

```
In [57]: curs=conn.cursor()
         curs.execute("SELECT count(DISTINCT mom_id) FROM oud_condition_source")
         curs.fetchall()

Out[57]: [(1754,)]
```

```
In [58]: curs=conn.cursor()
         curs.execute("SELECT count(DISTINCT mom_id) FROM greater_dob")
         curs.fetchall()

Out[58]: [(884,)]
```

```
In [59]: curs=conn.cursor()
         curs.execute("SELECT count(DISTINCT mom_id) FROM lesser_dob")
         curs.fetchall()

Out[59]: [(1555,)]
```

SC.4: Inspect the intersection of the two dob groups.

NOTE: There are 685 mom IDs in the intersection.

In [61]:

```
curs=conn.cursor()
curs.execute("SELECT count(DISTINCT l.mom_id) \
FROM lesser_dob l \
INNER JOIN greater_dob g ON g.mom_id = l.mom_id")
curs.fetchall()
```

Out[61]:

```
[(685,)]
```

Looking at the numbers we find a difference 1754 - 1555 = 199. The difference between the total number of unique mom ids (1754) and the number of unique mom ids associated with oud code start dates on or before the baby DOB (1555) is 199. This implies...

- There are 199 records where mom oud code entry dates are STRICTLY after baby DOB
- There are 685 records where the mom IDs have oud code entry dates BOTH before and after baby date of birth

Illustrate with Venn diagram:

In [84]:

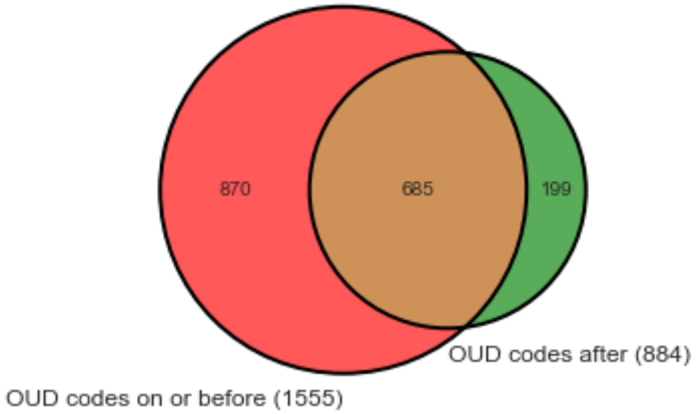
```
#Import libraries
from matplotlib_venn import venn2, venn2_circles, venn2_unweighted
from matplotlib_venn import venn3, venn3_circles
from matplotlib import pyplot as plt
%matplotlib inline

venn2(subsets = (870, 199, 685), set_labels = ('OUD codes on or before
venn2_circles(subsets = (870, 199, 685))
```

Out[84]:

```
<matplotlib_venn._common.VennDiagram at 0x7faa217aad30>
```

Out[84]:

```
[<matplotlib.patches.Circle at 0x7faa217ba2e0>,
 <matplotlib.patches.Circle at 0x7faa331224c0>]
```



Check the diagram: 1555 + 199 = 1754 (good).
Check the diagram: 884 + 870 = 1754 (good).

NOTE: This is naturally different than the following counts

> SELECT count(*) FROM oud_condition_source --106280
> SELECT count(*) FROM greater_dob --47910
> SELECT count(*) FROM lesser_dob --58370
> 47910 + 58370 = 106280

SC.5: Simplify the query by only including records where oud code entry or start date is <= baby dob

```
In [ ]: curs=conn.cursor()
        curs.execute("CREATE TABLE oud_CS2 AS \
        SELECT * FROM oud_condition_source c WHERE YEAR(c.condition_start_date)
        curs.fetchall()
```

To inspect the OUD_CS2 table run:

NOTE: There should only be 1555 unique mom IDs.

```
In [86]: curs=conn.cursor()
         curs.execute("SELECT count(DISTINCT mom_id) FROM oud_CS2")
         curs.fetchall()
```

```
Out[86]: [(1555,)]
```

End of "Sanity check"

---

7. Create a table that reports by year the unique mother IDs that have an assigned an OUD code on or before the baby DOB

```
In [ ]: curs=conn.cursor()
        curs.execute("CREATE TABLE endResult AS \
        SELECT YEAR(m.child_dob), COUNT(DISTINCT m.mom_id) \
        FROM oud_CS2 m \
        INNER JOIN dates d ON YEAR(m.child_dob) = d.dates \
        GROUP BY YEAR(m.child_dob)")
        curs.fetchall()
```

To inspect the ENDRESULT table run:

In [91]:

```
curs=conn.cursor()
curs.execute("SELECT * FROM endresult")
df=pd.DataFrame(curs.fetchall())
df.sort_values(by=[0], ascending=True)
```

Out[91]:

	0	1
2	2010	32
9	2011	58
5	2012	83
12	2013	94
1	2014	139
3	2015	178
11	2016	178
10	2017	203
4	2018	165
6	2019	165
8	2020	155
0	2021	191
7	2022	16

8. Create a table that reports the count of mother IDs that have with one child, two children, three children, etc.

In [ ]:

```
curs=conn.cursor()
curs.execute("CREATE TABLE reports AS \
SELECT mom_id, COUNT(mom_id) \
FROM mom_baby_ids_3 \
GROUP BY mom_id \
HAVING COUNT(mom_id)>1")
curs.fetchall()
```

In [ ]:

```
curs=conn.cursor()
curs.execute("DROP TABLE REPORTS")
curs.fetchall()
```

To inspect the REPORTS table run:

In [103]:
```
curs=conn.cursor()
curs.execute("SELECT * FROM REPORTS LIMIT 10")
df=pd.DataFrame(curs.fetchall())
df.iloc[:,:]
```

Out[103]:

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

To inspect the exact counts of mothers with n-children in reports run:

In [104]:
```
curs=conn.cursor()
curs.execute("SELECT count(DISTINCT mom_id) FROM reports")
curs.fetchall()
```

Out[104]:
```
[(11521,)]
```

"Sanity Check":

NOTE: Each code below is intended to be run individually.

NOTE: Total is 1657 which implies that mothers with OUD codes on or before baby dob occur in multiple years.

In [ ]:
```
SELECT count(DISTINCT c.mom_id) FROM oud_CS2 c WHERE YEAR(c.child_dob) =
SELECT DISTINCT mom_id FROM oud_CS2 WHERE YEAR(child_dob) = 2010 #32
SELECT count(DISTINCT c.mom_id) FROM oud_CS2 c WHERE YEAR(c.child_dob) =
SELECT DISTINCT mom_id FROM oud_CS2 WHERE YEAR(child_dob) = 2011 #58
SELECT count(DISTINCT c.mom_id) FROM oud_CS2 c WHERE YEAR(c.child_dob) =
SELECT DISTINCT mom_id FROM oud_CS2 WHERE YEAR(child_dob) = 2012 #83
SELECT count(DISTINCT c.mom_id) FROM oud_CS2 c WHERE YEAR(c.child_dob) =
SELECT DISTINCT mom_id FROM oud_CS2 WHERE YEAR(child_dob) = 2013 #94
SELECT count(DISTINCT c.mom_id) FROM oud_CS2 c WHERE YEAR(c.child_dob) =
SELECT DISTINCT mom_id FROM oud_CS2 WHERE YEAR(child_dob) = 2014 #139
SELECT count(DISTINCT c.mom_id) FROM oud_CS2 c WHERE YEAR(c.child_dob) =
SELECT DISTINCT mom_id FROM oud_CS2 WHERE YEAR(child_dob) = 2015 #178
SELECT count(DISTINCT c.mom_id) FROM oud_CS2 c WHERE YEAR(c.child_dob) =
SELECT DISTINCT mom_id FROM oud_CS2 WHERE YEAR(child_dob) = 2016 #178
SELECT count(DISTINCT c.mom_id) FROM oud_CS2 c WHERE YEAR(c.child_dob) =
SELECT DISTINCT mom_id FROM oud_CS2 WHERE YEAR(child_dob) = 2017 #203
SELECT count(DISTINCT c.mom_id) FROM oud_CS2 c WHERE YEAR(c.child_dob) =
SELECT DISTINCT mom_id FROM oud_CS2 WHERE YEAR(child_dob) = 2018 #165
SELECT count(DISTINCT c.mom_id) FROM oud_CS2 c WHERE YEAR(c.child_dob) =
SELECT DISTINCT mom_id FROM oud_CS2 WHERE YEAR(child_dob) = 2019 #165
SELECT count(DISTINCT c.mom_id) FROM oud_CS2 c WHERE YEAR(c.child_dob) =
SELECT DISTINCT mom_id FROM oud_CS2 WHERE YEAR(child_dob) = 2020 #155
SELECT count(DISTINCT c.mom_id) FROM oud_CS2 c WHERE YEAR(c.child_dob) =
SELECT DISTINCT mom_id FROM oud_CS2 WHERE YEAR(child_dob) = 2021 #191
SELECT count(DISTINCT c.mom_id) FROM oud_CS2 c WHERE YEAR(c.child_dob) =
SELECT DISTINCT mom_id FROM oud_CS2 WHERE YEAR(child_dob) = 2022 #16
Total is 1657 which implies that mothers with oud codes on or before ba
```

In [108]:
```
curs=conn.cursor()
curs.execute("SELECT count(DISTINCT c.mom_id) FROM oud_CS2 c WHERE YEAR
curs.fetchall()
```

Out[108]:
```
[(32,)]
```

See 7. above:

image info

https://www.cdc.gov/opioids/basics/epidemic.html" (https://www.cdc.gov/opioids/basics/epidemic.html")

https://www.cdc.gov/mmwr/preview/mmwrhtml/mm6450a3.htm (https://www.cdc.gov/mmwr/preview/mmwrhtml/mm6450a3.htm)

https://www.tn.gov/health/nas.html (https://www.tn.gov/health/nas.html)

In [ ]: