Content

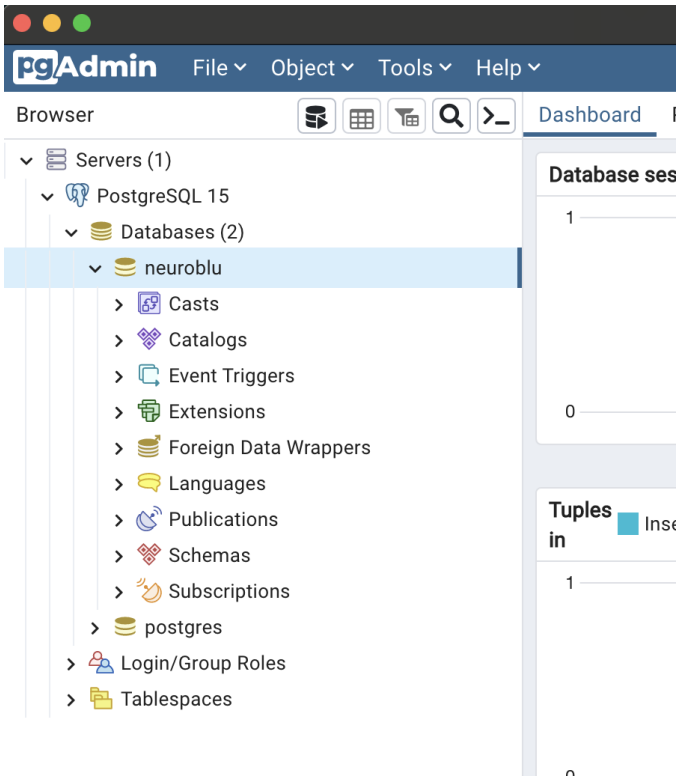
Overview

Task: Holmusk Technical Challenge
Candidate: Beverly Chua (bevjulia@gmail.com)

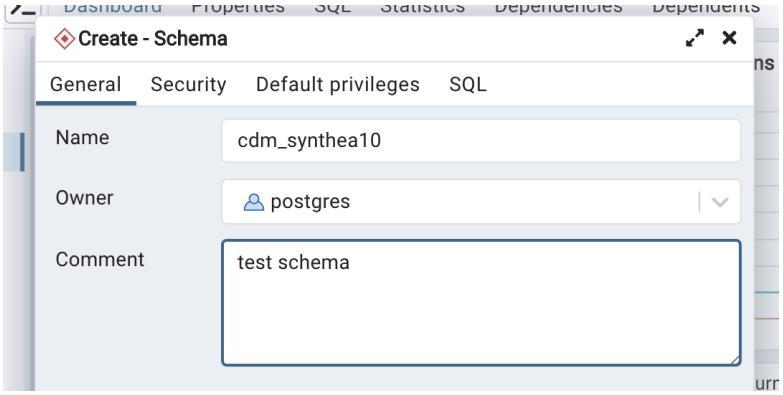
Task 1

Setting Up PostgreSQL 15

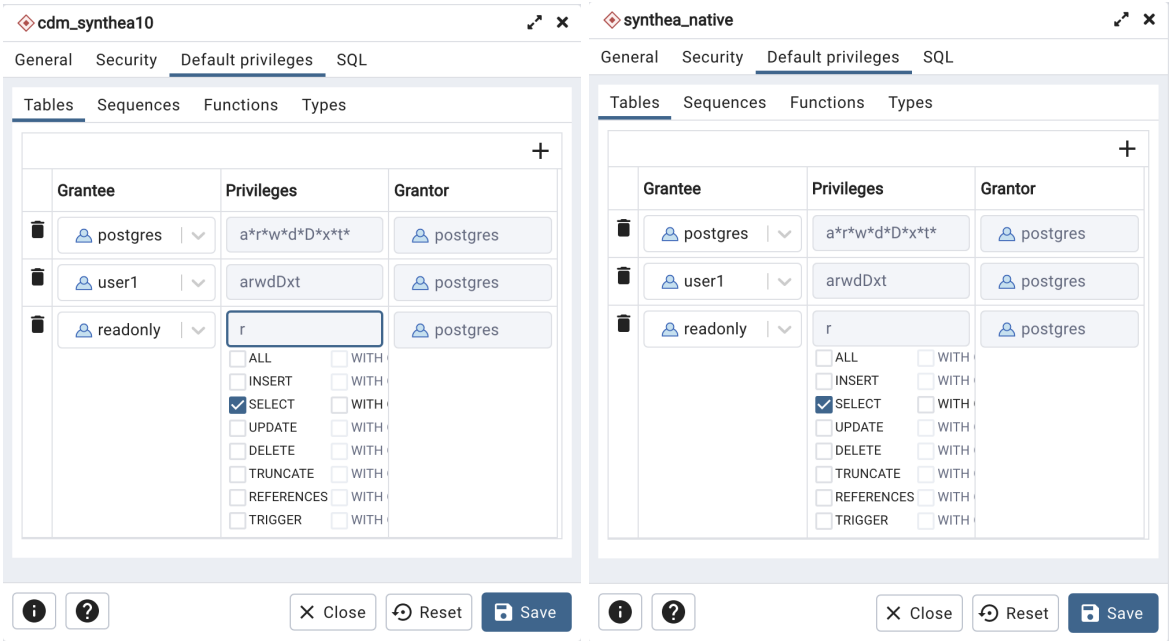
Step 1: Create a dedicated **neuroblu** database for this assignment.



Step 2: Create test schemas **cdm_synthea10** and **synthea_native**.



Step 3: Grant read access to **readonly** user and read/write access to **user1** on both schemas.



OR

```
GRANT CONNECT ON DATABASE neuroblu TO user1;
GRANT CONNECT ON DATABASE neuroblu TO readonly;

GRANT SELECT ON ALL TABLES IN SCHEMA public, cdm_synthea10, synthea_native TO readonly;
GRANT SELECT ON ALL SEQUENCES IN SCHEMA public, cdm_synthea10, synthea_native TO readonly;

GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public, cdm_synthea10, synthea_native TO user1;
GRANT ALL PRIVILEGES ON ALL SEQUENCES IN SCHEMA public, cdm_synthea10, synthea_native TO user1;
```

Query Analysis

A simple query was created to get the patients' conditions and encounters where the conditions were diagnosed.

The original crafted query can be found in **/holmusk_neuroblu/synthea_etl/sql/patient_conditions_and_encounters.sql.** The query plan is as follows:

Query | Query History

```
1  explain analyse
2  select distinct *
3  from synthea_native.patients p
4  inner join synthea_native.conditions c
5  on c.patient = p.id
6  inner join synthea_native.encounters e
7  on e.id = c.encounter
8  and e.patient = c.patient
9  where 1=1
10 and c.start >= '1975-01-01'
11 and e.encounterclass = 'ambulatory'
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	HashAggregate (cost=84.12..86.39 rows=227 width=5007) (actual time=5.451..5.505 rows=24 loops=1)
2	Group Key: p.id, p.birthdate, p.deathdate, p.ssn, p.drivers, p.passport, p.prefix, p.first, p.last, p.suffix, p.maiden, p.marital, p.race, p.ethnicity, p.gender, p.birthplace, p.address, p.city, p.state, p.county, p.zip, p.lat, p.
3	Batches: 1 Memory Usage: 129kB
4	-> Hash Join (cost=37.44..56.88 rows=227 width=5007) (actual time=0.813..1.752 rows=384 loops=1)
5	Hash Cond: (((c.patient)::text = (p.id)::text) AND ((c.encounter)::text = (e.id)::text))
6	-> Seq Scan on conditions c (cost=0.00..13.00 rows=400 width=118) (actual time=0.018..0.153 rows=400 loops=1)
7	Filter: (start >= '1975-01-01'::date)
8	-> Hash (cost=35.58..35.58 rows=124 width=4889) (actual time=0.770..0.772 rows=124 loops=1)
9	Buckets: 1024 Batches: 1 Memory Usage: 83kB
10	-> Hash Join (cost=10.22..35.58 rows=124 width=4889) (actual time=0.041..0.465 rows=124 loops=1)
11	Hash Cond: ((e.patient)::text = (p.id)::text)
12	-> Seq Scan on encounters e (cost=0.00..23.65 rows=124 width=297) (actual time=0.014..0.249 rows=124 loops=1)
13	Filter: ((encounterclass)::text = 'ambulatory'::text)
14	Rows Removed by Filter: 328
15	-> Hash (cost=10.10..10.10 rows=10 width=4592) (actual time=0.011..0.012 rows=2 loops=1)
16	Buckets: 1024 Batches: 1 Memory Usage: 9kB

Total rows: 19 of 19 | Query complete 00:00:00.058 | Ln 11, Col 36

In the above query, we select distinct records as there were duplicated rows in the data. The duplicated rows come from the **synthea_native.encounters** table. From the above query plan, it looks like the first HashAggregate action is taking up some memory and the subsequent Hash Join action is joining on a larger number of rows when the end result is only 19 rows. We want to try and see if we can optimize this.

The optimized query can be found in **/holmusk_neuroblu/synthea_etl/sql/patient_conditions_and_encounters_optimized.sql.** The query plan is as follows:

Query Query History

```
explain analyse
select *
from synthea_native.patients p
inner join synthea_native.conditions c
on c.patient = p.id
inner join (select distinct * from synthea_native.encounters) e
on e.id = c.encounter
and e.patient = c.patient
where 1=1
and c.start >= '1975-01-01'
and e.encounterclass = 'ambulatory'
```

Data Output Messages Notifications

	QUERY PLAN text
3	-> Seq Scan on conditions c (cost=0.00..13.00 rows=400 width=118) (actual time=0.019..0.141 rows=400 loops=1)
4	Filter: (start >= '1975-01-01'::date)
5	-> Hash (cost=40.57..40.57 rows=10 width=4889) (actual time=0.869..0.872 rows=31 loops=1)
6	Buckets: 1024 Batches: 1 Memory Usage: 27kB
7	-> Hash Join (cost=38.52..40.57 rows=10 width=4889) (actual time=0.714..0.787 rows=31 loops=1)
8	Hash Cond: ((encounters.patient)::text = (p.id)::text)
9	-> HashAggregate (cost=28.30..29.12 rows=82 width=297) (actual time=0.692..0.722 rows=31 loops=1)
10	Group Key: encounters.id, encounters.start, encounters.stop, encounters.patient, encounters.organization, encounters.provider, encounters.payer, encounters.encounterclass, encounters.code, encounters.descri
11	Batches: 1 Memory Usage: 72kB
12	-> Seq Scan on encounters (cost=0.00..23.65 rows=124 width=297) (actual time=0.013..0.254 rows=124 loops=1)
13	Filter: ((encounterclass)::text = 'ambulatory'::text)
14	Rows Removed by Filter: 328
15	-> Hash (cost=10.10..10.10 rows=10 width=4592) (actual time=0.011..0.012 rows=2 loops=1)
16	Buckets: 1024 Batches: 1 Memory Usage: 9kB
17	-> Seq Scan on patients p (cost=0.00..10.10 rows=10 width=4592) (actual time=0.005..0.006 rows=2 loops=1)
18	Planning Time: 0.498 ms

Total rows: 19 of 19 Query complete 00:00:00.055 Ln 10, Col 28

We can now see that when we create a subquery to select distinct records from the encounters table first, the grouping functions are taking up less memory and the entire query is sped up from an execution time of 5.742 ms to 1.501 ms. The full query plans can be found in **/holmusk_neuroblu/synthea_etl/sql/query_plans.txt.**

Task 2

Testing the Python package

We have created a package named **querypackage** with 2 functions to query using 1) an SQL string, and 2) a list of patient IDs. The test SQL string and the list of patient IDs is defined in the **test_querypackage.py** file.

Below you can find the result of running **test_querypackage.py** where both functions query the PostgreSQL neuroblu database and return a pandas dataframe.

```
(base_env) beverlychua@Beverlys-MBP-2 holmusk_neuroblu % python test_querypackage.py

==========================================
Testing get_query() function with input
"select * from synthea_native.observations limit 10"
==========================================

Database connection successful
         date                                patient                                 encounter     code  ...                          value    units     type        category
0  1993-06-17  cac10d78-b13d-660e-57ae-4ee7a6c84fc8  a3a3ee14-c5c5-9f51-abbd-df3a13bc1492   8302-2  ...                          169.8       cm  numeric     vital-signs
1  1993-06-17  cac10d78-b13d-660e-57ae-4ee7a6c84fc8  a3a3ee14-c5c5-9f51-abbd-df3a13bc1492  72514-3  ...                            2.0  {score}  numeric     vital-signs
2  1993-06-17  cac10d78-b13d-660e-57ae-4ee7a6c84fc8  a3a3ee14-c5c5-9f51-abbd-df3a13bc1492  29463-7  ...                           77.0       kg  numeric     vital-signs
3  1993-06-17  cac10d78-b13d-660e-57ae-4ee7a6c84fc8  a3a3ee14-c5c5-9f51-abbd-df3a13bc1492  39156-5  ...                           26.7     kg/m2  numeric     vital-signs
4  1993-06-17  cac10d78-b13d-660e-57ae-4ee7a6c84fc8  a3a3ee14-c5c5-9f51-abbd-df3a13bc1492   8462-4  ...                           77.0     mm[Hg]  numeric     vital-signs
5  1993-06-17  cac10d78-b13d-660e-57ae-4ee7a6c84fc8  a3a3ee14-c5c5-9f51-abbd-df3a13bc1492   8480-6  ...                          123.0     mm[Hg]  numeric     vital-signs
6  1993-06-17  cac10d78-b13d-660e-57ae-4ee7a6c84fc8  a3a3ee14-c5c5-9f51-abbd-df3a13bc1492   8867-4  ...                           65.0       /min  numeric     vital-signs
7  1993-06-17  cac10d78-b13d-660e-57ae-4ee7a6c84fc8  a3a3ee14-c5c5-9f51-abbd-df3a13bc1492   9279-1  ...                           14.0       /min  numeric     vital-signs
8  1993-06-17  cac10d78-b13d-660e-57ae-4ee7a6c84fc8  a3a3ee14-c5c5-9f51-abbd-df3a13bc1492  72166-2  ...  Never smoked tobacco (finding)       None    text  social-history
9  1993-06-17  cac10d78-b13d-660e-57ae-4ee7a6c84fc8  a3a3ee14-c5c5-9f51-abbd-df3a13bc1492  76501-6  ...                             No       None    text          survey

[10 rows x 9 columns]

==========================================
Testing get_person() function with input
['cac10d78-b13d-660e-57ae-4ee7a6c84fc8']
==========================================

Database connection successful
                                 id   birthdate   deathdate          ssn    drivers  ...              lon healthcare_expenses healthcare_coverage   fips income
0  cac10d78-b13d-660e-57ae-4ee7a6c84fc8  1959-01-22  2003-05-08  999-83-7658  S99935392  ...  -71.1563728098053           61329.42           974019.54  25025  78384

[1 rows x 27 columns]
(base_env) beverlychua@Beverlys-MBP-2 holmusk_neuroblu %
```

Appendix

Setting Up MySQL Version 8

Initially, the chosen relational database was MySQL. However, due to incompatibility with the recommended R ETL util package to load the Synthea CSV data, PostgreSQL was chosen thereafter.

Nonetheless, below we document the steps for setting up the MySQL database.

We can see the MySQL version downloaded here:

```
(base) beverlychua@Beverlys-MBP-2 ~ % sudo mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 15
Server version: 8.0.31 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

Create dedicated database:

```
mysql> CREATE DATABASE neuroblu;
Query OK, 1 row affected (0.00 sec)
```

Create user1 and grant permissions to access all schema and tables within neuroblu database:

```
mysql> CREATE USER 'user1' IDENTIFIED BY 'user1password';
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> GRANT ALL PRIVILEGES ON neuroblu.* TO 'user1';
Query OK, 0 rows affected (0.00 sec)
```

Create readonly and grant permissions to read schemas and tables in neuroblu database:

```
mysql> CREATE USER 'readonly' IDENTIFIED BY 'readonlypassword';
Query OK, 0 rows affected (0.01 sec)

mysql> GRANT SELECT ON neuroblu.* TO 'readonly';
Query OK, 0 rows affected (0.01 sec)
```