Data Engineering 300 Homework 2

Jonathan Wu (jwy5141)
Due: 5/13/2025

Part 1

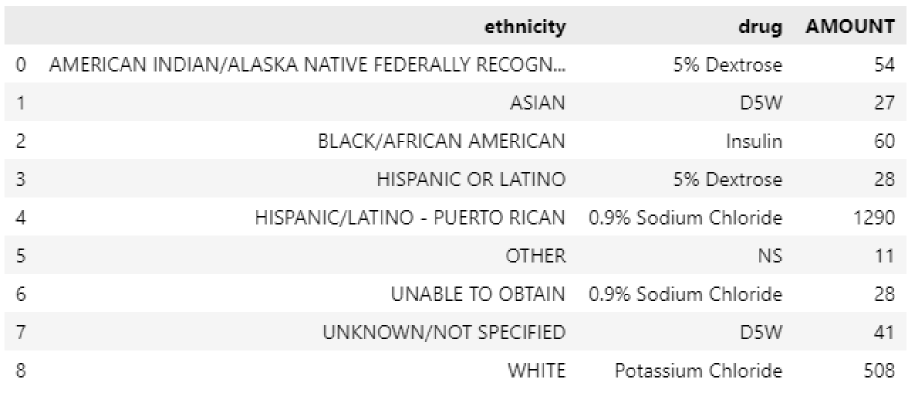
Question 1:

The SQL Query I used is:

```
SELECT
ETHNICITY,DRUG,AMOUNT
FROM (
   SELECT
      ADMISSIONS.ETHNICITY,
      PRESCRIPTIONS.DRUG,
      COUNT(*) AS AMOUNT
   FROM PRESCRIPTIONS
   JOIN ADMISSIONS ON PRESCRIPTIONS.subject_id = ADMISSIONS.subject_id
   GROUP BY ADMISSIONS.ETHNICITY, PRESCRIPTIONS.DRUG
) AS counts
QUALIFY AMOUNT = MAX(AMOUNT) OVER (PARTITION BY ETHNICITY)
ORDER BY ETHNICITY;
```

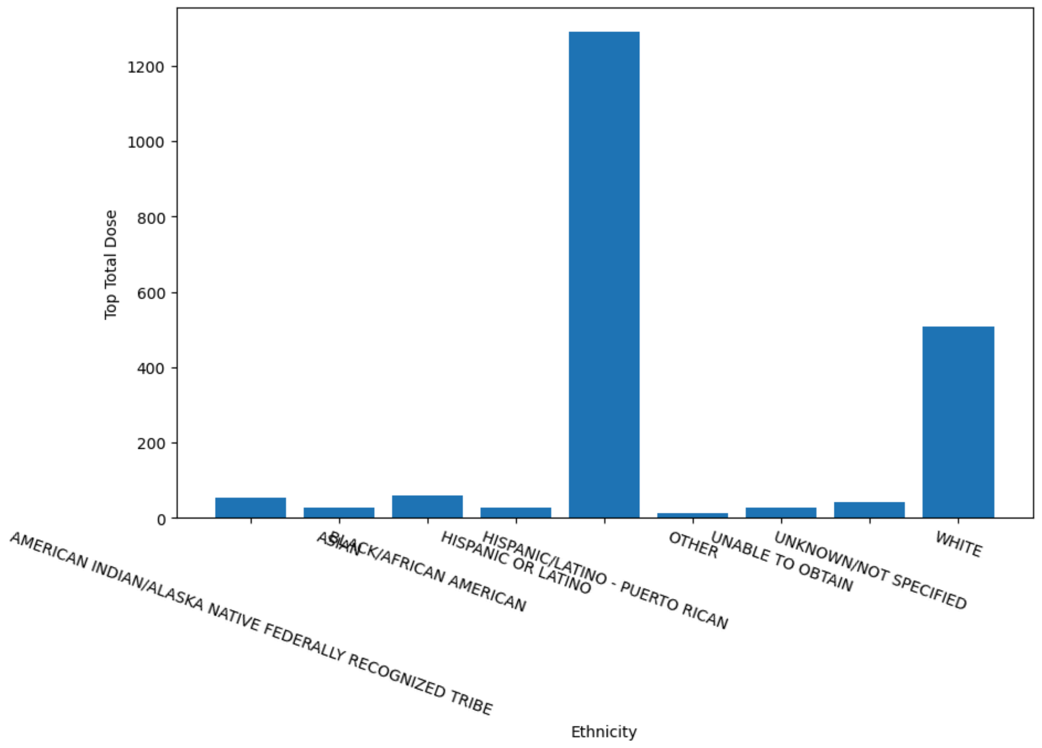
The query joins the admissions and prescriptions tables together and groups by ethnicity and prescribed drugs. I select the maximum count of the prescriptions. This assumes that every drug prescribed has the same potency and "amount" if it has the same name. I essentially assume that every drug has comparable "amounts" even if some drugs are prescribed because they have less strength than others (ie opioids vs ibuprofen)

This is my table:



	ethnicity	drug	AMOUNT
0	AMERICAN INDIAN/ALASKA NATIVE FEDERALLY RECOGN...	5% Dextrose	54
1	ASIAN	D5W	27
2	BLACK/AFRICAN AMERICAN	Insulin	60
3	HISPANIC OR LATINO	5% Dextrose	28
4	HISPANIC/LATINO - PUERTO RICAN	0.9% Sodium Chloride	1290
5	OTHER	NS	11
6	UNABLE TO OBTAIN	0.9% Sodium Chloride	28
7	UNKNOWN/NOT SPECIFIED	D5W	41
8	WHITE	Potassium Chloride	508

I see that Sodium Chloride is frequently prescribed to Hispanic, Latino, and Puerto Rican populations. There is a significant numerical difference between Hispanic/Latino - Puerto Rican prescriptions (1290) and any other ethnicity (less than half for all). There also are lots of prescriptions to White patients, with the most frequent prescription being for Potassium Chloride (508). I wonder if this is a representative sample of the population this study was conducted in because there are basically no African American or Asian ethnicity data points (60 for insulin, 27 for D5W) compared to Hispanic/Latino and White data points.

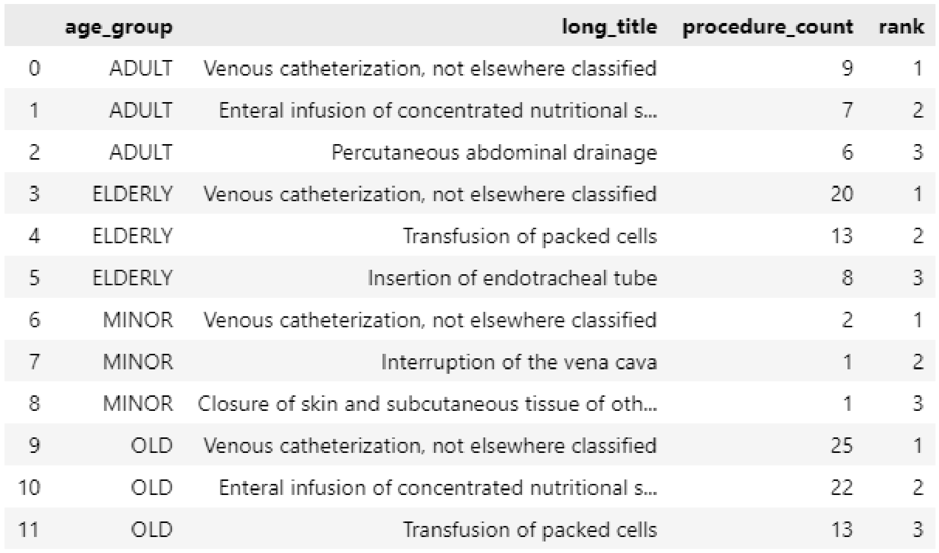


Question 2:

My query was:

```
SELECT *
FROM (
   SELECT
      CASE
         WHEN date_diff('year', cast(dob as DATE), cast(dischtime as DATE)) <= 19 THEN
'MINOR'
         WHEN date_diff('year', cast(dob as DATE), cast(dischtime as DATE)) >= 20 AND
date_diff('year', cast(dob as DATE), cast(dischtime as DATE)) < 50 THEN 'ADULT'
         WHEN date_diff('year', cast(dob as DATE), cast(dischtime as DATE)) >= 50 AND
date_diff('year', cast(dob as DATE), cast(dischtime as DATE)) < 80 THEN 'OLD'
         ELSE 'ELDERLY'
      END AS age_group,
      long_title,
      COUNT(*) AS procedure_count,
        ROW_NUMBER() OVER (
            PARTITION BY
              CASE
                   WHEN date_diff('year', cast(dob as DATE), cast(dischtime as DATE)) <= 19 THEN
'MINOR'
                   WHEN date_diff('year', cast(dob as DATE), cast(dischtime as DATE)) >= 20 AND
date_diff('year', cast(dob as DATE), cast(dischtime as DATE)) < 50 THEN 'ADULT'
                   WHEN date_diff('year', cast(dob as DATE), cast(dischtime as DATE)) >= 50 AND
date_diff('year', cast(dob as DATE), cast(dischtime as DATE)) < 80 THEN 'OLD'
               ELSE 'ELDERLY'
               END
           ORDER BY COUNT(*) DESC
        ) AS rank
    FROM PATIENTS
    JOIN ADMISSIONS ON PATIENTS.subject_id = ADMISSIONS.subject_id
    JOIN PROD_ICD ON ADMISSIONS.hadm_id = PROD_ICD.hadm_id
    JOIN D_ICD ON PROD_ICD.ICD9_CODE = D_ICD.ICD9_CODE
    GROUP BY age_group, long_title
) ranked
WHERE rank <= 3
ORDER BY age_group, procedure_count DESC;
```

This query has a query and a subquery. The subquery takes the age group, procedure name, and the number of procedures and orders it by the number of procedures. The larger query selects the top 3 procedures ordered by age group and count. The subquery joins the patients, admissions, immediate care procedures, names of the procedures.



	age_group	long_title	procedure_count	rank
0	ADULT	Venous catheterization, not elsewhere classified	9	1
1	ADULT	Enteral infusion of concentrated nutritional s...	7	2
2	ADULT	Percutaneous abdominal drainage	6	3
3	ELDERLY	Venous catheterization, not elsewhere classified	20	1
4	ELDERLY	Transfusion of packed cells	13	2
5	ELDERLY	Insertion of endotracheal tube	8	3
6	MINOR	Venous catheterization, not elsewhere classified	2	1
7	MINOR	Interruption of the vena cava	1	2
8	MINOR	Closure of skin and subcutaneous tissue of oth...	1	3
9	OLD	Venous catheterization, not elsewhere classified	25	1
10	OLD	Enteral infusion of concentrated nutritional s...	22	2
11	OLD	Transfusion of packed cells	13	3

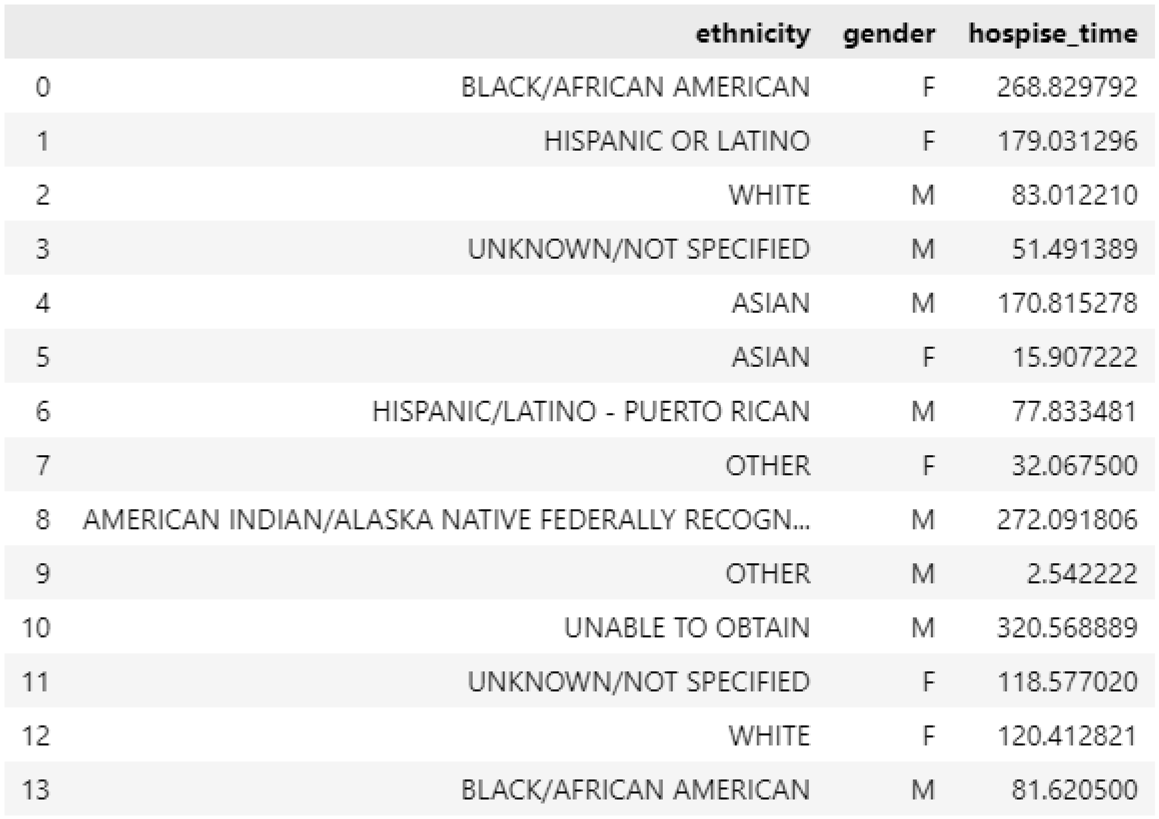
In my results, we can see that there are many of the same procedures between all age groups. Venous catheterization (top across all age groups), transfusion, and infusions are common procedures throughout every group except the age group <= 19 (called minors in my table). Minors don’t have many procedures documented. This might be due to the fact that collecting health data on minors is nearly impossible (maybe unless one passes and the parent gives permission?).

Question 3:

My query was:

```
SELECT ethnicity, gender, avg(epoch(cast(outtime as TIMESTAMP) - cast(intime as
TIMESTAMP)))/3600 AS hospise_time
FROM ICUSTAYS
JOIN PATIENTS on ICUSTAYS.subject_id=PATIENTS.subject_id
JOIN ADMISSIONS on ICUSTAYS.subject_id=ADMISSIONS.subject_id GROUP BY ethnicity,
gender
```

I select the ethnicity, gender, and hospice time joined on ICU stays, patients, and admissions. The hospice time is calculated by casting the entry and exit time as timestamps, then finding the average of the differences. In the question, it asks for how long patients stay in the ICU and if there is a difference in ICU length among gender or ethnicity. I assume this is largely semantic because Cassandra needs to partition on different columns so it doesn’t really make sense to ask for only one Cassandra table for analysis (states “Design *a* Cassandra table”).



	ethnicity	gender	hospise_time
0	BLACK/AFRICAN AMERICAN	F	268.829792
1	HISPANIC OR LATINO	F	179.031296
2	WHITE	M	83.012210
3	UNKNOWN/NOT SPECIFIED	M	51.491389
4	ASIAN	M	170.815278
5	ASIAN	F	15.907222
6	HISPANIC/LATINO - PUERTO RICAN	M	77.833481
7	OTHER	F	32.067500
8	AMERICAN INDIAN/ALASKA NATIVE FEDERALLY RECOGN...	M	272.091806
9	OTHER	M	2.542222
10	UNABLE TO OBTAIN	M	320.568889
11	UNKNOWN/NOT SPECIFIED	F	118.577020
12	WHITE	F	120.412821
13	BLACK/AFRICAN AMERICAN	M	81.620500

Here we see that men seem to spend less time in the hospital than women. Furthermore, patients that are in an unknown (“Unable to obtain”) and Federally recognized tribe ethnicities spend more time in the hospital than other ethnicities such as a different unknown (“Other”) and Hispanic/Latino Puerto Rican ethnicities.

Part 2

Question 1

Cassandra Table:

```
CREATE TABLE IF NOT EXISTS drug_usage_by_ethnicity1 (
   ethnicity TEXT,
   drug TEXT,
   usage_id INT,
   PRIMARY KEY ((ethnicity), drug, usage_id)
);
```

Upload code:

```
prepared = session.prepare("""
    INSERT INTO drug_usage_by_ethnicity1 (ethnicity, drug, usage_id)
    VALUES (?, ?, ?)
""")

for i, row in part1.iterrows():
    session.execute(prepared, (row['ethnicity'], row['drug'], i))
```

Query:

```
rows = session.execute("SELECT ethnicity FROM drug_usage_by_ethnicity1")
ethnicity_list = list(set(row.ethnicity for row in rows))

records = []

for eth in ethnicity_list:
    rows = session.execute("""
        SELECT drug FROM drug_usage_by_ethnicity1 WHERE ethnicity = %s
    """, [eth])
```

Post extraction: I grouped the query using pandas to get the top drugs used and my table is:

	ETHNICITY	DRUG	AMOUNT
0	HISPANIC OR LATINO	5% Dextrose	28
1	UNABLE TO OBTAIN	0.9% Sodium Chloride	28
2	AMERICAN INDIAN/ALASKA NATIVE FEDERALLY RECOGN...	5% Dextrose	54
3	HISPANIC/LATINO - PUERTO RICAN	0.9% Sodium Chloride	1290
4	UNKNOWN/NOT SPECIFIED	D5W	41
5	OTHER	NS	11
6	BLACK/AFRICAN AMERICAN	Insulin	60
7	WHITE	Potassium Chloride	508
8	ASIAN	D5W	27

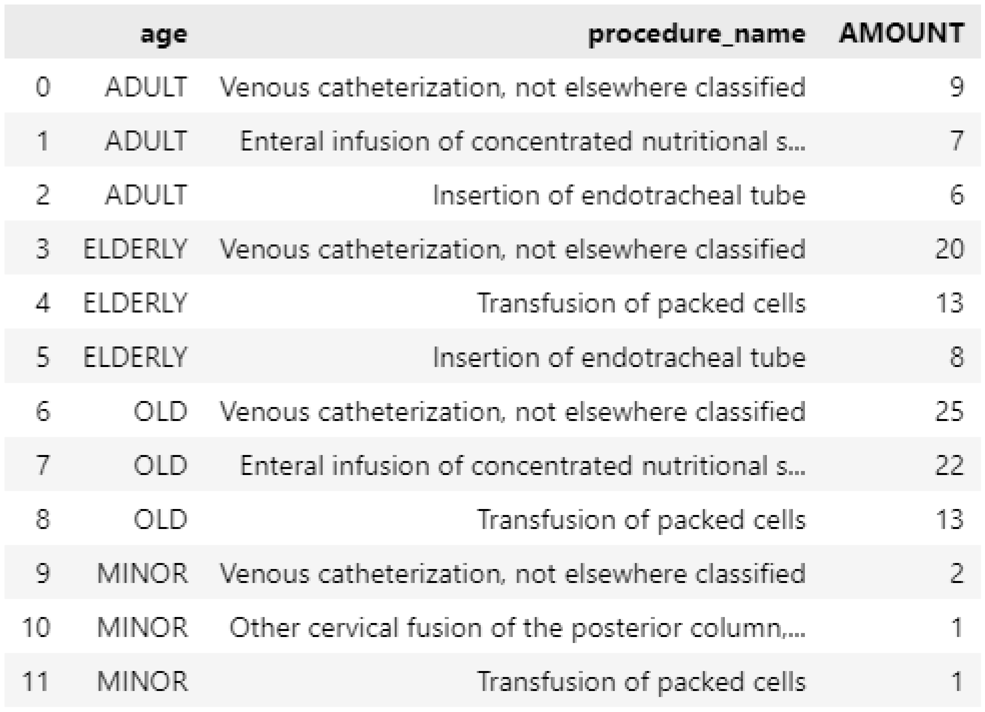
Question 2

Cassandra Table:
CREATE TABLE IF NOT EXISTS question24 (
    procedure_name TEXT,
    age TEXT,
    ID INT,
    PRIMARY KEY ((age), procedure_name, ID)
);

Upload code:
```
prepared = session.prepare("""
    INSERT INTO question24 (procedure_name, AGE, ID)
    VALUES (?, ?, ?)
""")

for i, row in part2.iterrows():
    session.execute(prepared, (row.PROCEDURE_NAME, row.age_group, i))
```

Query:
```
rows = session.execute("SELECT AGE FROM question24")
age_list = list(set(row.age for row in rows))

records = []

for age in age_list:
    rows = session.execute("""
        SELECT procedure_name FROM question24 WHERE age = %s
    """, [age])
```

I used pandas again to group the data together by age group, selecting the top 3 rather than the max count. I added these to a dictionary before turning it into a dataframe seen here:



	age	procedure_name	AMOUNT
0	ADULT	Venous catheterization, not elsewhere classified	9
1	ADULT	Enteral infusion of concentrated nutritional s...	7
2	ADULT	Insertion of endotracheal tube	6
3	ELDERLY	Venous catheterization, not elsewhere classified	20
4	ELDERLY	Transfusion of packed cells	13
5	ELDERLY	Insertion of endotracheal tube	8
6	OLD	Venous catheterization, not elsewhere classified	25
7	OLD	Enteral infusion of concentrated nutritional s...	22
8	OLD	Transfusion of packed cells	13
9	MINOR	Venous catheterization, not elsewhere classified	2
10	MINOR	Other cervical fusion of the posterior column,...	1
11	MINOR	Transfusion of packed cells	1

Question 3

Cassandra Table:

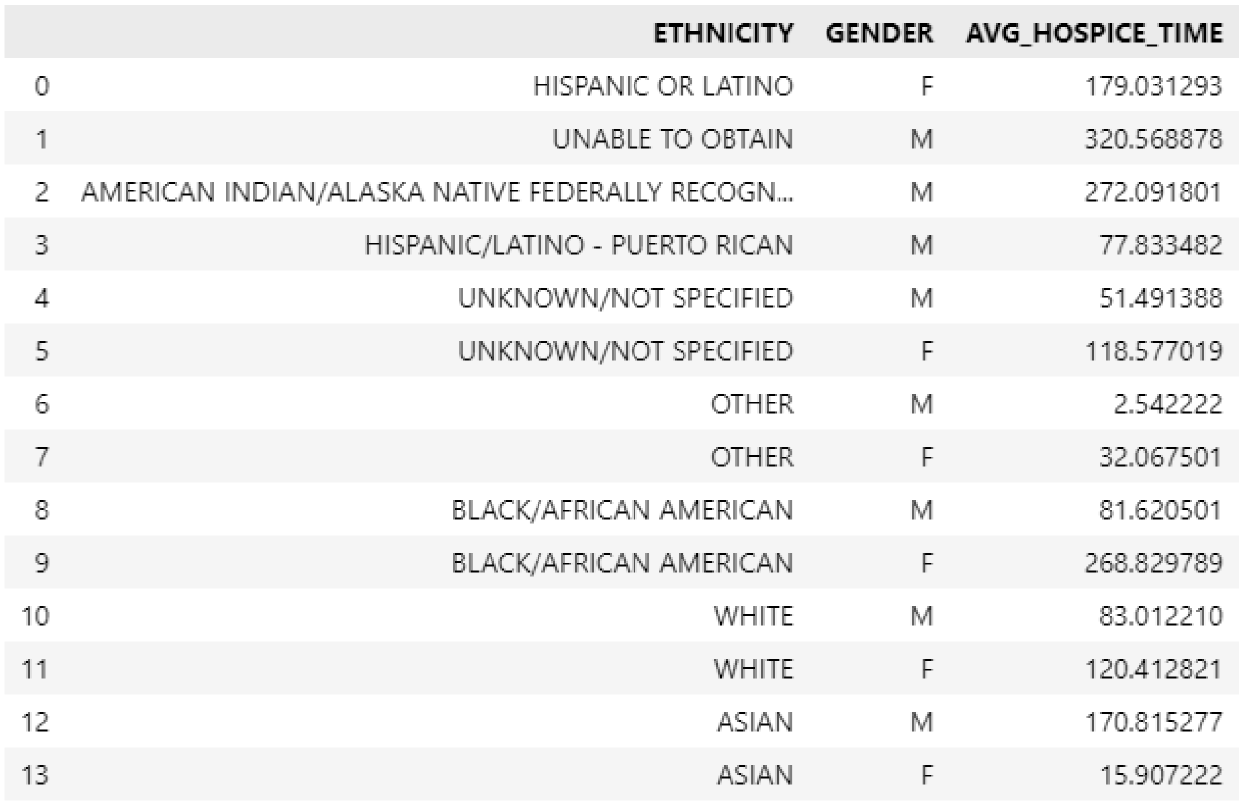
```
CREATE TABLE IF NOT EXISTS question31 (
   ethnicity TEXT,
   gender TEXT,
   hospise_time FLOAT,
   id INT,
   PRIMARY KEY ((ethnicity), gender, hospise_time, id)
);
```

Data Upload code:

```
prepared = session.prepare("""
    INSERT INTO question31 (ethnicity, gender, hospise_time, id)
    VALUES (?, ?, ?,?)
""")

for i, row in part3.iterrows():
    session.execute(prepared, (row['ethnicity'], row['gender'],
row['hospise_time'], i))
```

Query:
```
rows = session.execute("""SELECT ethnicity FROM question31""")
ethnicity_list = list(set(row.ethnicity for row in rows))

records = []

for eth in ethnicity_list:
    rows = session.execute("""SELECT gender FROM question31 WHERE 
ethnicity = %s""", [eth])
    gender_list = list(set(row.gender for row in rows))
    for gen in gender_list:
        # Filter the DataFrame for each group
        rows = session.execute("""
        SELECT hospise_time FROM question31 WHERE ethnicity = %s AND 
gender = %s
        """, [eth, gen])
```

I basically used the same procedure as the previous 2 questions, using mean rather than max on the hospice time. My table has the same contents but a different order than question 3:



	ETHNICITY	GENDER	AVG_HOSPICE_TIME
0	HISPANIC OR LATINO	F	179.031293
1	UNABLE TO OBTAIN	M	320.568878
2	AMERICAN INDIAN/ALASKA NATIVE FEDERALLY RECOGN...	M	272.091801
3	HISPANIC/LATINO - PUERTO RICAN	M	77.833482
4	UNKNOWN/NOT SPECIFIED	M	51.491388
5	UNKNOWN/NOT SPECIFIED	F	118.577019
6	OTHER	M	2.542222
7	OTHER	F	32.067501
8	BLACK/AFRICAN AMERICAN	M	81.620501
9	BLACK/AFRICAN AMERICAN	F	268.829789
10	WHITE	M	83.012210
11	WHITE	F	120.412821
12	ASIAN	M	170.815277
13	ASIAN	F	15.907222

Generative AI Disclosure

I incessantly asked ChatGPT to try and help me set up Docker (technically it's the Dockerfile not letting me access AWS or something?) so my code would run.
Things that were discussed included:

1. How to set up credentials for AWS
2. How to ensure the dockerfile let me have AWS access
3. How to change user permissions (this got to big brain for me to understand)
4. How to open Jupyter Notebook so I could access my contents
5. How to change my Boto3 session to have the AWS login (this did nothing)

I also used it to develop the ranked query used in part 1. My query up to that point is written right above the query in the submitted .ipynb. I also used ChatGPT as an emergency psychiatrist and verbal punching bag as I rage quit over AWS credentials that *still* don't really want to work.
I also prompted ChatGPT as I rage quit over how my first Cassandra table took 18 minutes to load in (whereas the other 2 that I accomplished before my first query took less than a minute).

- ChatGPT said it should take around 30 seconds and I cancelled after the query ran for like 5 minutes

Did I mention I also used it as an emergency psychiatrist as I didn't know why my first Cassandra table TAKES EIGHTEEN WHOLE MINUTES TO UPLOAD DATA TO?
The first table is somewhat designed by ChatGPT as I kept going on a trip over how it took EIGHTEEN minutes. I got to learn from my PsychogistGPT about how using the unique id was a way to not overwrite previous data though.

Side tangent: Have I mentioned the fact that no one has bother explaining how Dockerfiles set up our Jupyter Notebooks and how to properly get the Dockerfile to run so that we can properly run Cassandra queries on the AWS server? I cannot tell you how in the ones and zeros does Docker know how to set up the Jupyter Notebook so I don't need to pip install stuff and set things up. The only person who's bothered to explain it to me so far is my psychologist.