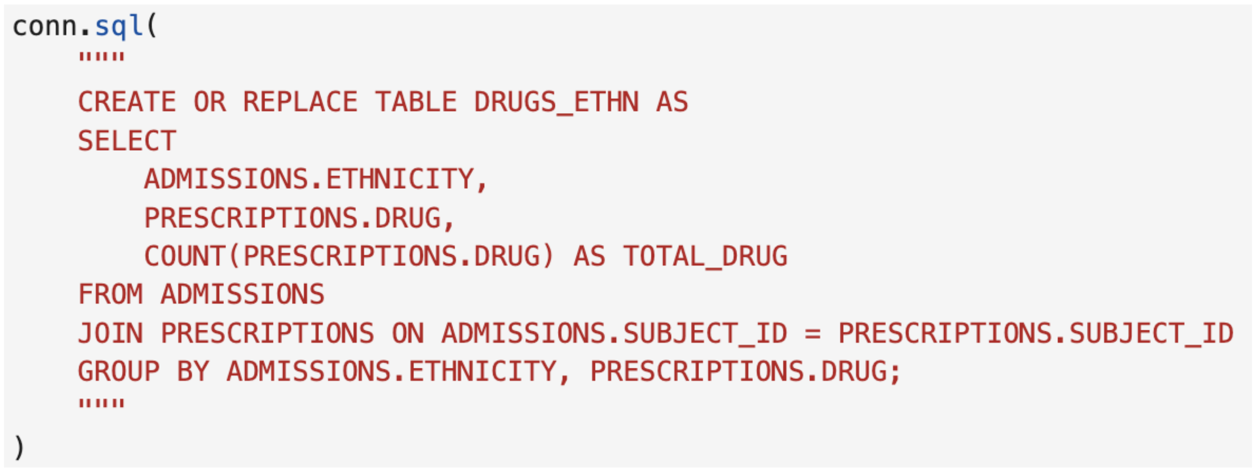
Seeley McGillis
DSE 300: HW 2 Written Answers

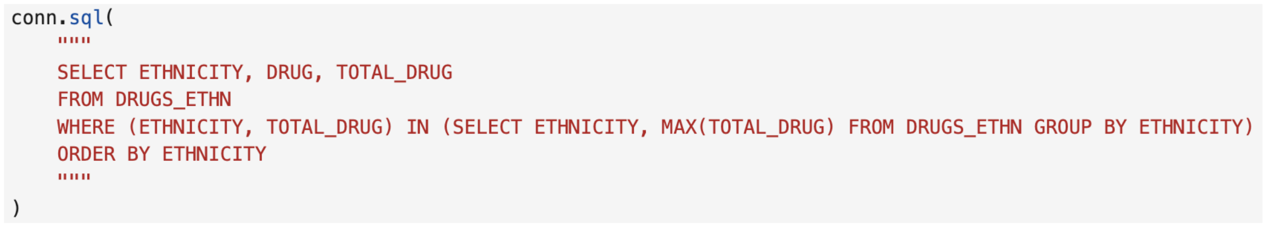
## Part 1: Relational

*Question 1: Create a summary of types of drugs and their total amount used by ethnicity. Report the top usage in each ethnicity group. You may have to make certain assumptions in calculating their total amount.*

A. SQL query



```
conn.sql(
    """
    CREATE OR REPLACE TABLE DRUGS_ETHN AS
    SELECT
        ADMISSIONS.ETHNICITY,
        PRESCRIPTIONS.DRUG,
        COUNT(PRESCRIPTIONS.DRUG) AS TOTAL_DRUG
    FROM ADMISSIONS
    JOIN PRESCRIPTIONS ON ADMISSIONS.SUBJECT_ID = PRESCRIPTIONS.SUBJECT_ID
    GROUP BY ADMISSIONS.ETHNICITY, PRESCRIPTIONS.DRUG;
    """
)
```



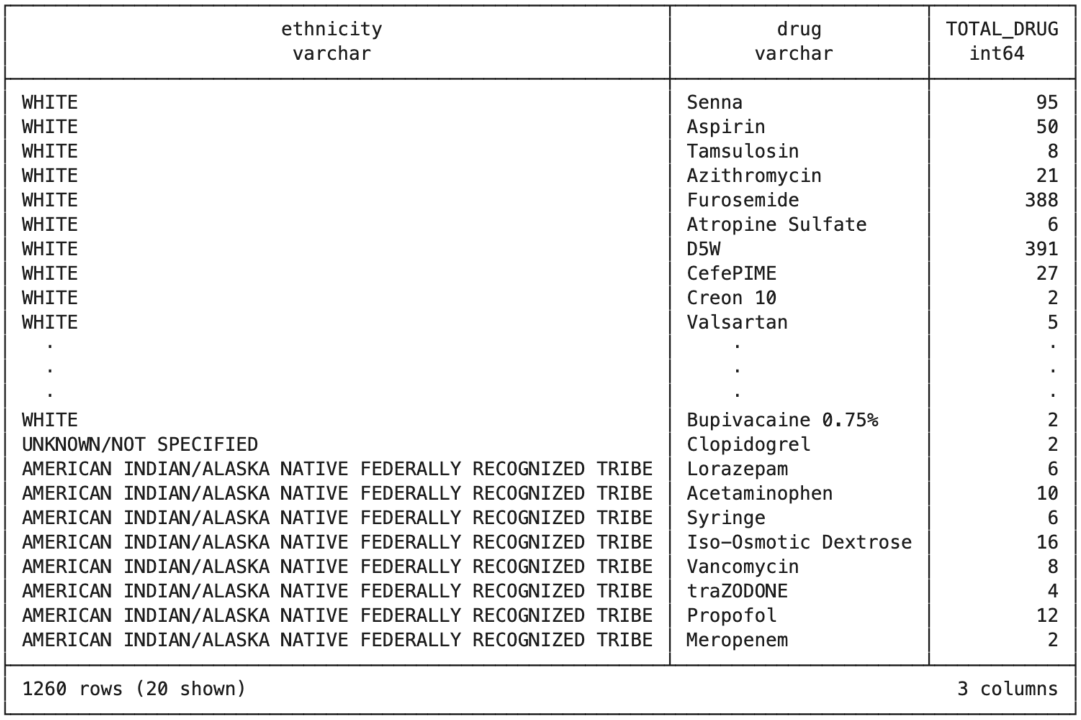
```
conn.sql(
    """
    SELECT ETHNICITY, DRUG, TOTAL_DRUG
    FROM DRUGS_ETHN
    WHERE (ETHNICITY, TOTAL_DRUG) IN (SELECT ETHNICITY, MAX(TOTAL_DRUG) FROM DRUGS_ETHN GROUP BY ETHNICITY)
    ORDER BY ETHNICITY
    """
)
```

B. Brief explanation of the query (i.e., what operations are performed by the major parts of the query)

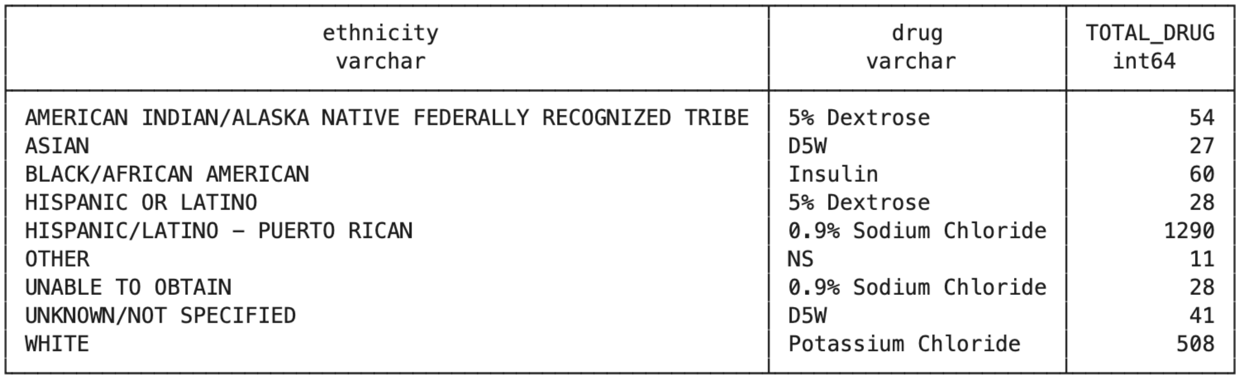
The first query creates a table that reports each drug and the count of how much each ethnicity used them in the records. It does this by creating columns on the ethnicity, the different drugs, and the count of how much that drug is used by that ethnicity. It joins the table admissions which contains ethnicity, and the prescriptions given on the subject id. The table is grouped by ethnicity and the drug type.

The second query creates a table that reports the most used drug by each ethnicity. It does this by selecting the columns in the previously created drugs_ethn table, and reports the max count of drugs used by each ethnicity. The table is grouped by ethnicity. The table is ordered by ethnicity alphabetically.

C. The first several lines of your resulting table(s)



```
┌──────────────────────────────────────────────────────────┬──────────────────────┬────────────┐
│                        ethnicity                         │         drug         │ TOTAL_DRUG │
│                         varchar                          │       varchar        │   int64    │
├──────────────────────────────────────────────────────────┼──────────────────────┼────────────┤
│ WHITE                                                    │ Senna                │         95 │
│ WHITE                                                    │ Aspirin              │         50 │
│ WHITE                                                    │ Tamsulosin           │          8 │
│ WHITE                                                    │ Azithromycin         │         21 │
│ WHITE                                                    │ Furosemide           │        388 │
│ WHITE                                                    │ Atropine Sulfate     │          6 │
│ WHITE                                                    │ D5W                  │        391 │
│ WHITE                                                    │ CefePIME             │         27 │
│ WHITE                                                    │ Creon 10             │          2 │
│ WHITE                                                    │ Valsartan            │          5 │
│   ·                                                      │    ·                 │          · │
│   ·                                                      │    ·                 │          · │
│   ·                                                      │    ·                 │          · │
│ WHITE                                                    │ Bupivacaine 0.75%    │          2 │
│ UNKNOWN/NOT SPECIFIED                                    │ Clopidogrel          │          2 │
│ AMERICAN INDIAN/ALASKA NATIVE FEDERALLY RECOGNIZED TRIBE │ Lorazepam            │          6 │
│ AMERICAN INDIAN/ALASKA NATIVE FEDERALLY RECOGNIZED TRIBE │ Acetaminophen        │         10 │
│ AMERICAN INDIAN/ALASKA NATIVE FEDERALLY RECOGNIZED TRIBE │ Syringe              │          6 │
│ AMERICAN INDIAN/ALASKA NATIVE FEDERALLY RECOGNIZED TRIBE │ Iso-Osmotic Dextrose │         16 │
│ AMERICAN INDIAN/ALASKA NATIVE FEDERALLY RECOGNIZED TRIBE │ Vancomycin           │          8 │
│ AMERICAN INDIAN/ALASKA NATIVE FEDERALLY RECOGNIZED TRIBE │ traZODONE            │          4 │
│ AMERICAN INDIAN/ALASKA NATIVE FEDERALLY RECOGNIZED TRIBE │ Propofol             │         12 │
│ AMERICAN INDIAN/ALASKA NATIVE FEDERALLY RECOGNIZED TRIBE │ Meropenem            │          2 │
├──────────────────────────────────────────────────────────┴──────────────────────┴────────────┤
│ 1260 rows (20 shown)                                                               3 columns │
└──────────────────────────────────────────────────────────────────────────────────────────────┘
```



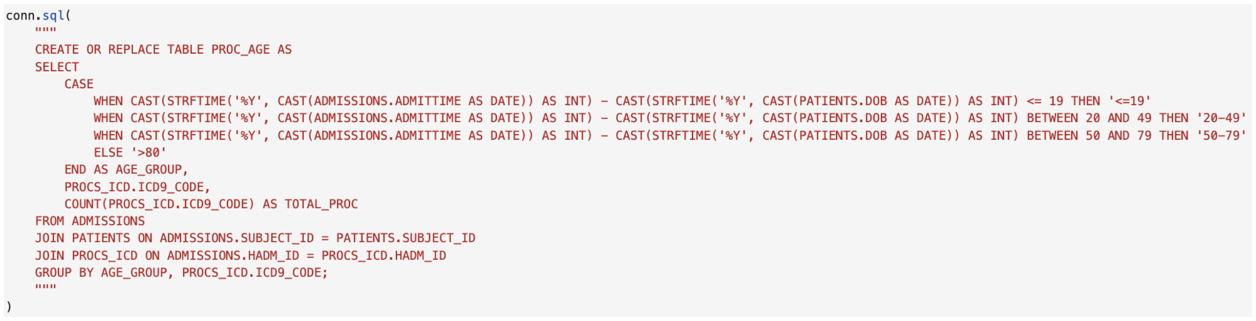
```
┌──────────────────────────────────────────────────────────┬──────────────────────┬────────────┐
│                        ethnicity                         │         drug         │ TOTAL_DRUG │
│                         varchar                          │       varchar        │   int64    │
├──────────────────────────────────────────────────────────┼──────────────────────┼────────────┤
│ AMERICAN INDIAN/ALASKA NATIVE FEDERALLY RECOGNIZED TRIBE │ 5% Dextrose          │         54 │
│ ASIAN                                                    │ D5W                  │         27 │
│ BLACK/AFRICAN AMERICAN                                   │ Insulin              │         60 │
│ HISPANIC OR LATINO                                       │ 5% Dextrose          │         28 │
│ HISPANIC/LATINO - PUERTO RICAN                           │ 0.9% Sodium Chloride │       1290 │
│ OTHER                                                    │ NS                   │         11 │
│ UNABLE TO OBTAIN                                         │ 0.9% Sodium Chloride │         28 │
│ UNKNOWN/NOT SPECIFIED                                    │ D5W                  │         41 │
│ WHITE                                                    │ Potassium Chloride   │        508 │
└──────────────────────────────────────────────────────────┴──────────────────────┴────────────┘
```

D. Summary of findings

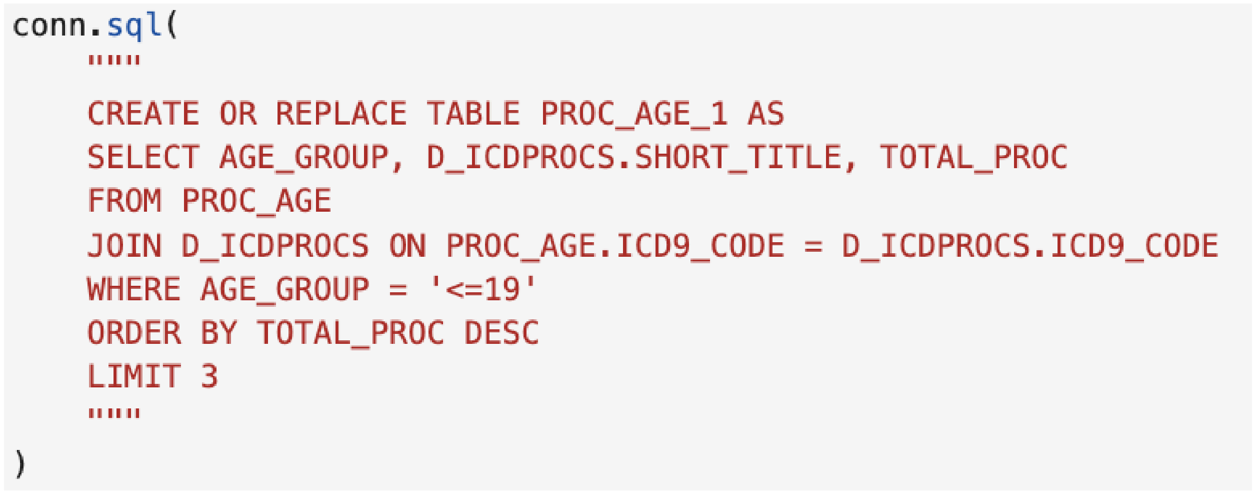
Besides other, unable to obtain, and unknown, each race seems to have a different most used drug. The most prescribed drug appears to be 0.9% Sodium Chloride to Puerto Ricans, followed by Potassium Chloride to white patients. This may mean that white patients are more likely to experience potassium deficiencies, and that Puerto Rican patients may need to replace bodily fluids more.

*Question 2: Create a summary of procedures performed on patients by age groups (<=19, 20-49, 50-79, >80). Report the top three procedures, along with the name of the procedures, performed in each age group.*

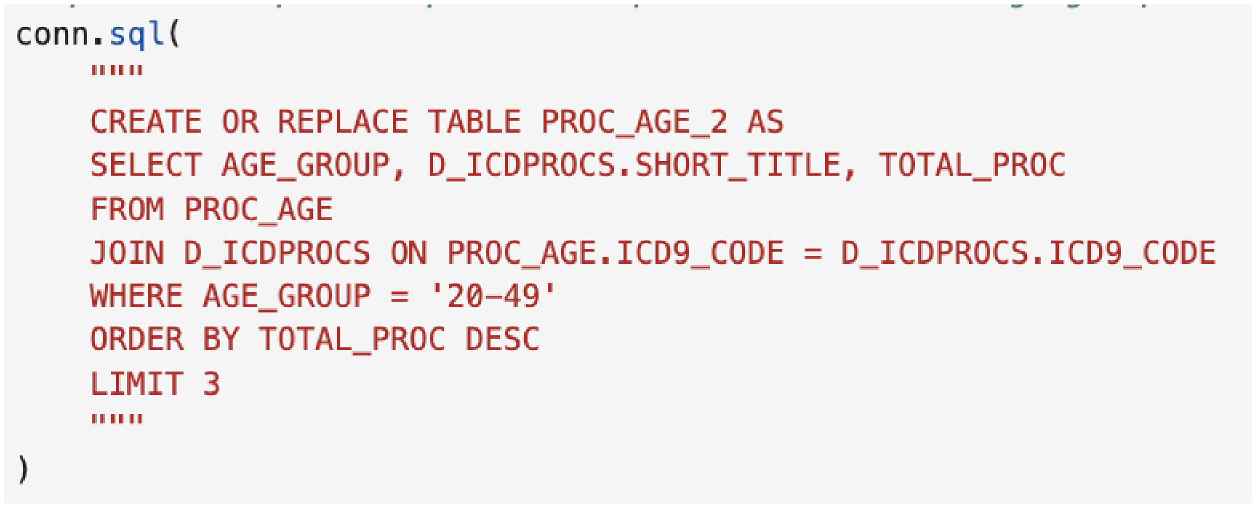
A. SQL query



```
conn.sql(
    """
    CREATE OR REPLACE TABLE PROC_AGE AS
    SELECT
        CASE
            WHEN CAST(STRFTIME('%Y', CAST(ADMISSIONS.ADMITTIME AS DATE)) AS INT) - CAST(STRFTIME('%Y', CAST(PATIENTS.DOB AS DATE)) AS INT) <= 19 THEN '<=19'
            WHEN CAST(STRFTIME('%Y', CAST(ADMISSIONS.ADMITTIME AS DATE)) AS INT) - CAST(STRFTIME('%Y', CAST(PATIENTS.DOB AS DATE)) AS INT) BETWEEN 20 AND 49 THEN '20-49'
            WHEN CAST(STRFTIME('%Y', CAST(ADMISSIONS.ADMITTIME AS DATE)) AS INT) - CAST(STRFTIME('%Y', CAST(PATIENTS.DOB AS DATE)) AS INT) BETWEEN 50 AND 79 THEN '50-79'
            ELSE '>80'
        END AS AGE_GROUP,
        PROCS_ICD.ICD9_CODE,
        COUNT(PROCS_ICD.ICD9_CODE) AS TOTAL_PROC
    FROM ADMISSIONS
    JOIN PATIENTS ON ADMISSIONS.SUBJECT_ID = PATIENTS.SUBJECT_ID
    JOIN PROCS_ICD ON ADMISSIONS.HADM_ID = PROCS_ICD.HADM_ID
    GROUP BY AGE_GROUP, PROCS_ICD.ICD9_CODE;
    """
)
```



```
conn.sql(
    """
    CREATE OR REPLACE TABLE PROC_AGE_1 AS
    SELECT AGE_GROUP, D_ICDPROCS.SHORT_TITLE, TOTAL_PROC
    FROM PROC_AGE
    JOIN D_ICDPROCS ON PROC_AGE.ICD9_CODE = D_ICDPROCS.ICD9_CODE
    WHERE AGE_GROUP = '<=19'
    ORDER BY TOTAL_PROC DESC
    LIMIT 3
    """
)
```



```
conn.sql(
    """
    CREATE OR REPLACE TABLE PROC_AGE_2 AS
    SELECT AGE_GROUP, D_ICDPROCS.SHORT_TITLE, TOTAL_PROC
    FROM PROC_AGE
    JOIN D_ICDPROCS ON PROC_AGE.ICD9_CODE = D_ICDPROCS.ICD9_CODE
    WHERE AGE_GROUP = '20-49'
    ORDER BY TOTAL_PROC DESC
    LIMIT 3
    """
)
```

```
conn.sql(
    """
    CREATE OR REPLACE TABLE PROC_AGE_3 AS
    SELECT AGE_GROUP, D_ICDPROCS.SHORT_TITLE, TOTAL_PROC
    FROM PROC_AGE
    JOIN D_ICDPROCS ON PROC_AGE.ICD9_CODE = D_ICDPROCS.ICD9_CODE
    WHERE AGE_GROUP = '50-79'
    ORDER BY TOTAL_PROC DESC
    LIMIT 3
    """
)
```

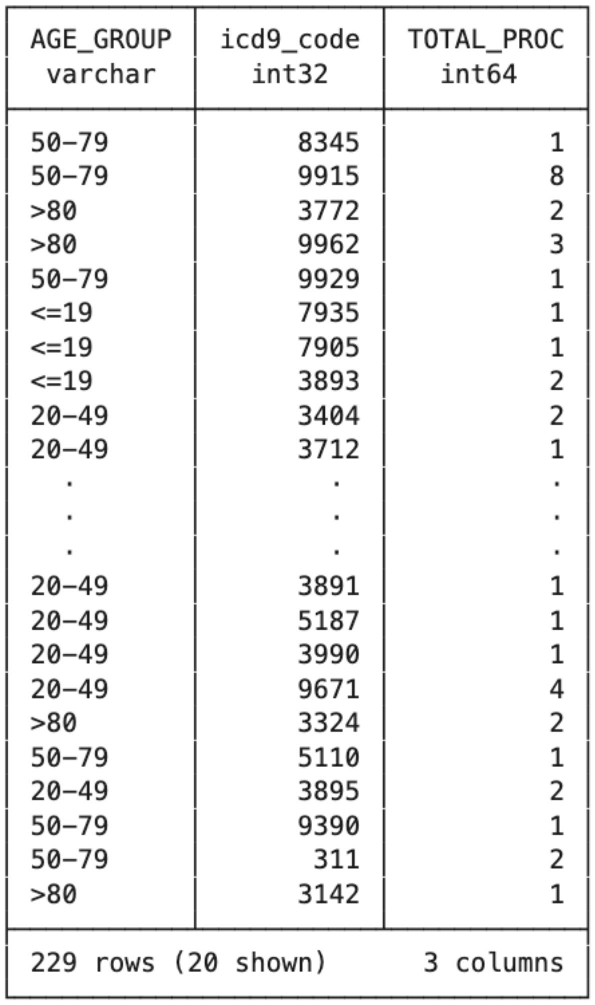
```
conn.sql(
    """
    CREATE OR REPLACE TABLE PROC_AGE_4 AS
    SELECT AGE_GROUP, D_ICDPROCS.SHORT_TITLE, TOTAL_PROC
    FROM PROC_AGE
    JOIN D_ICDPROCS ON PROC_AGE.ICD9_CODE = D_ICDPROCS.ICD9_CODE
    WHERE AGE_GROUP = '>80'
    ORDER BY TOTAL_PROC DESC
    LIMIT 3
    """
)
```

## B. Brief explanation of the query (i.e., what operations are performed by the major parts of the query)

The first query starts by converting the year on the admissions time to a number as well as the year of the birthdate of the patient to a number. The birth year is subtracted from the admissions time in order to get the age of the patient in years. A when statement is used to evaluate which age group the patient falls under based on the obtained age in years. The table proc_age is created with the total amount of each procedure for each age group based on the count of ICD9_CODE. The tables admissions, patients, and procs_icd are joined in order to perform this action, and the table is grouped by age_group and the icd9_code.

The four other queries report the top three procedures performed in each age group. This is done by creating another table, and using the previously created proc_age table. The title of the procedures is extracted from the table d_icdprocs which is joined through the icd9_code. Information is filtered based on the desired age_group, and the table is sorted by the total procedures in descending order.

C. The first several lines of your resulting table



```
┌───────────┬───────────┬────────────┐
│ AGE_GROUP │ icd9_code │ TOTAL_PROC │
│  varchar  │   int32   │   int64    │
├───────────┼───────────┼────────────┤
│ 50-79     │      8345 │          1 │
│ 50-79     │      9915 │          8 │
│ >80       │      3772 │          2 │
│ >80       │      9962 │          3 │
│ 50-79     │      9929 │          1 │
│ <=19      │      7935 │          1 │
│ <=19      │      7905 │          1 │
│ <=19      │      3893 │          2 │
│ 20-49     │      3404 │          2 │
│ 20-49     │      3712 │          1 │
│   ·       │        ·  │          · │
│   ·       │        ·  │          · │
│   ·       │        ·  │          · │
│ 20-49     │      3891 │          1 │
│ 20-49     │      5187 │          1 │
│ 20-49     │      3990 │          1 │
│ 20-49     │      9671 │          4 │
│ >80       │      3324 │          2 │
│ 50-79     │      5110 │          1 │
│ 20-49     │      3895 │          2 │
│ 50-79     │      9390 │          1 │
│ 50-79     │       311 │          2 │
│ >80       │      3142 │          1 │
├───────────┴───────────┴────────────┤
│ 229 rows (20 shown)      3 columns │
└────────────────────────────────────┘
```

```
┌───────────┬──────────────────────────┬────────────┐
│ AGE_GROUP │       short_title        │ TOTAL_PROC │
│  varchar  │         varchar          │   int64    │
├───────────┼──────────────────────────┼────────────┤
│ <=19      │ Venous cath NEC          │          2 │
│ <=19      │ Closed bronchial biopsy  │          1 │
│ <=19      │ Vertebral fx repair      │          1 │
└───────────┴──────────────────────────┴────────────┘
```

```
┌───────────┬──────────────────────────┬────────────┐
│ AGE_GROUP │       short_title        │ TOTAL_PROC │
│  varchar  │         varchar          │   int64    │
├───────────┼──────────────────────────┼────────────┤
│ 20-49     │ Venous cath NEC          │          9 │
│ 20-49     │ Entral infus nutrit sub  │          7 │
│ 20-49     │ Insert endotracheal tube │          6 │
└───────────┴──────────────────────────┴────────────┘
```

```
┌───────────┬──────────────────────────┬────────────┐
│ AGE_GROUP │       short_title        │ TOTAL_PROC │
│  varchar  │         varchar          │   int64    │
├───────────┼──────────────────────────┼────────────┤
│ 50-79     │ Venous cath NEC          │         25 │
│ 50-79     │ Entral infus nutrit sub  │         22 │
│ 50-79     │ Packed cell transfusion  │         13 │
└───────────┴──────────────────────────┴────────────┘
```

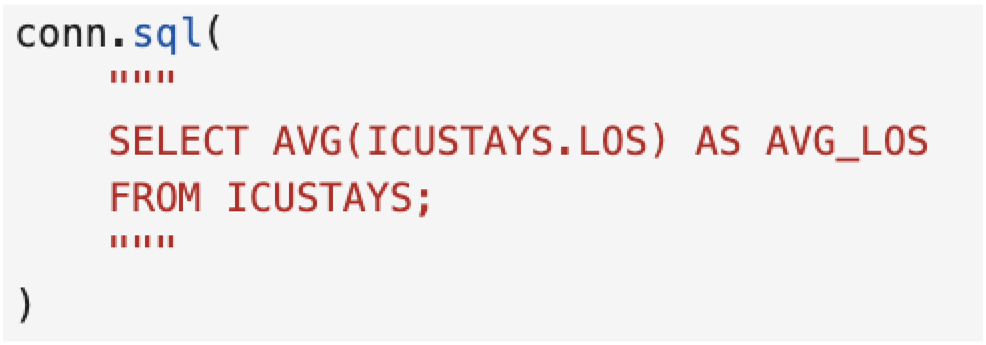
```
┌───────────┬──────────────────────────┬────────────┐
│ AGE_GROUP │       short_title        │ TOTAL_PROC │
│  varchar  │         varchar          │   int64    │
├───────────┼──────────────────────────┼────────────┤
│ >80       │ Venous cath NEC          │         20 │
│ >80       │ Packed cell transfusion  │         13 │
│ >80       │ Insert endotracheal tube │          8 │
└───────────┴──────────────────────────┴────────────┘
```

## D. Summary of findings

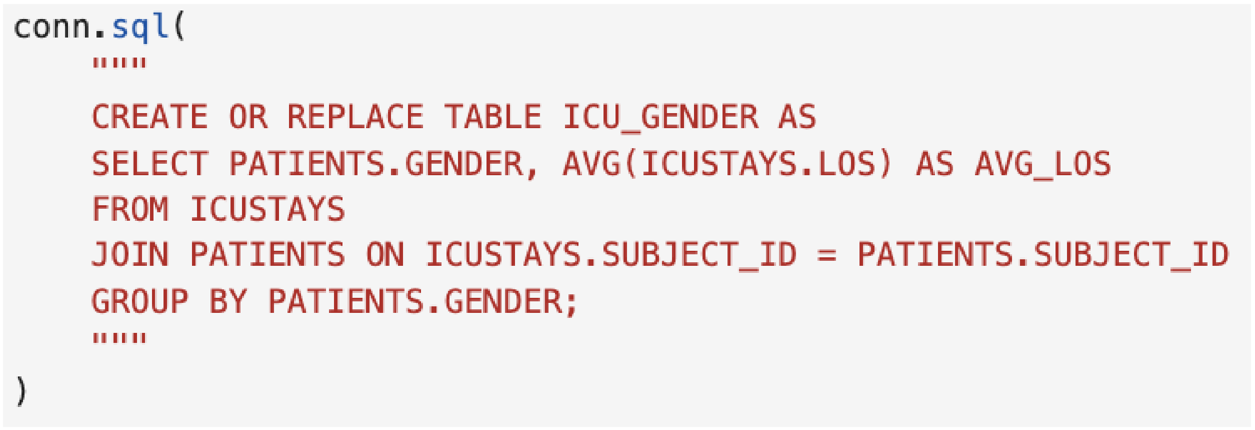
The most popular procedure for every age group is Venous cath NEC. This likely means this procedure is important for all patients and is likely not dependent on the age of the patient. The second and third most popular procedure then varies from age group to age group. The packed cell transfusion is popular in both older age groups, 50-79 and over 80 years old. This might mean that as patients get older they are more likely to require transfusions.

*Question 3: How long do patients stay in the ICU? Is there a difference in the ICU length of stay among gender or ethnicity?*

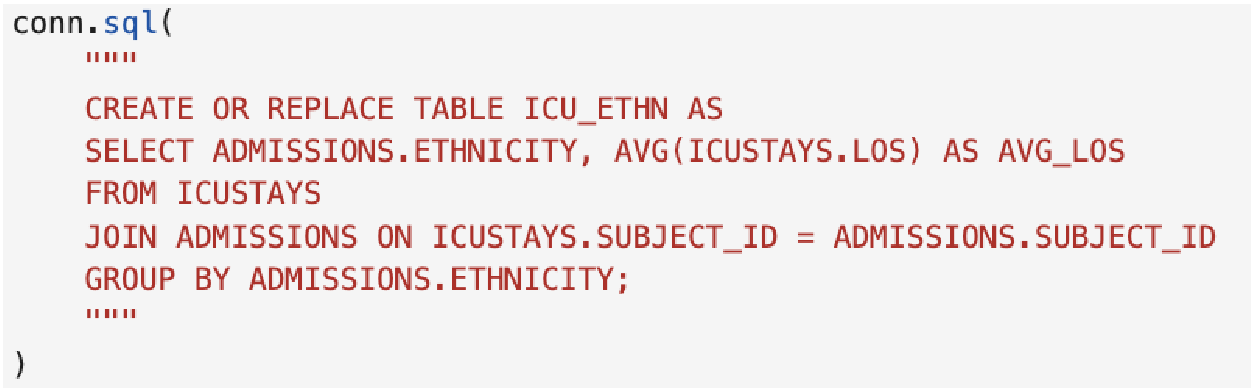
<u>A. SQL query</u>



```
conn.sql(
    """
    SELECT AVG(ICUSTAYS.LOS) AS AVG_LOS
    FROM ICUSTAYS;
    """
)
```



```
conn.sql(
    """
    CREATE OR REPLACE TABLE ICU_GENDER AS
    SELECT PATIENTS.GENDER, AVG(ICUSTAYS.LOS) AS AVG_LOS
    FROM ICUSTAYS
    JOIN PATIENTS ON ICUSTAYS.SUBJECT_ID = PATIENTS.SUBJECT_ID
    GROUP BY PATIENTS.GENDER;
    """
)
```



```
conn.sql(
    """
    CREATE OR REPLACE TABLE ICU_ETHN AS
    SELECT ADMISSIONS.ETHNICITY, AVG(ICUSTAYS.LOS) AS AVG_LOS
    FROM ICUSTAYS
    JOIN ADMISSIONS ON ICUSTAYS.SUBJECT_ID = ADMISSIONS.SUBJECT_ID
    GROUP BY ADMISSIONS.ETHNICITY;
    """
)
```

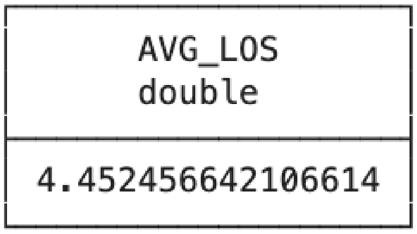
<u>B. Brief explanation of the query (i.e., what operations are performed by the major parts of the query)</u>

The first query simply reports the average length of stay from ICUSTAYS.

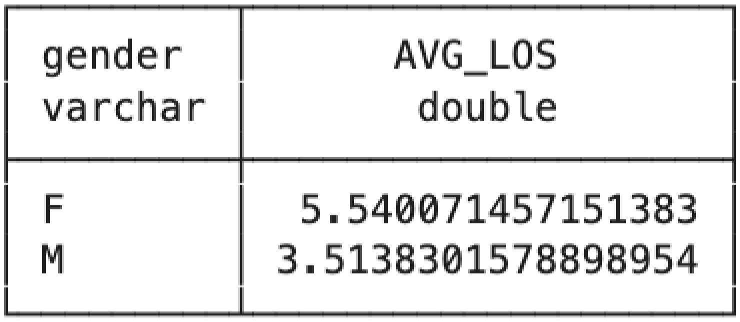
The second query creates the table ICU_GENDER by joining the tables PATIENTS and ICU_STAYS on SUBJECT_ID. The query then reports the average length of stay from ICUSTAYS grouped by the gender of the patients.

The third query creates the table ICU_ETHN by joining the tables ADMISSIONS and ICU_STAYS on SUBJECT_ID. The query then reports the average length of stay from ICUSTAYS grouped by the ethnicity of the patients which is from ADMISSIONS.

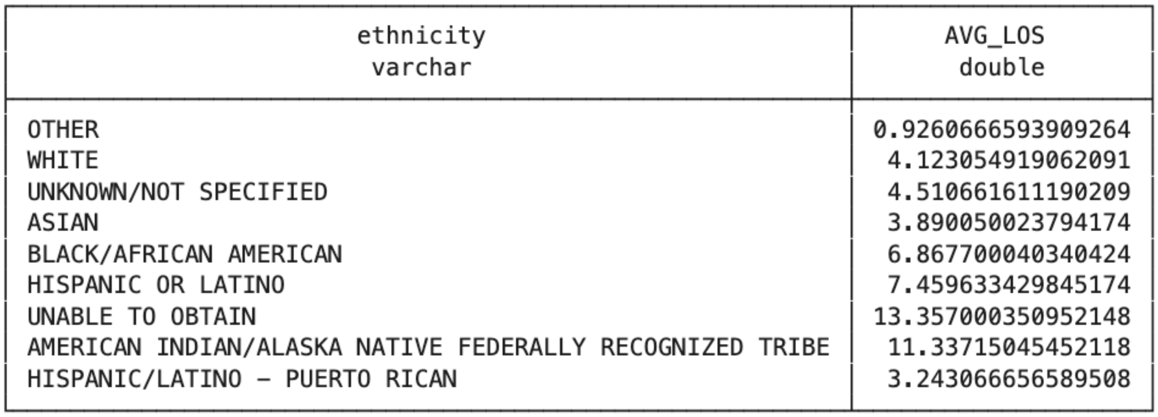
<u>C. The first several lines of your resulting table</u>



| AVG_LOS<br>double |
|---|
| 4.452456642106614 |



| gender<br>varchar | AVG_LOS<br>double |
|---|---|
| F | 5.540071457151383 |
| M | 3.5138301578898954 |



| ethnicity<br>varchar | AVG_LOS<br>double |
|---|---|
| OTHER | 0.9260666593909264 |
| WHITE | 4.123054919062091 |
| UNKNOWN/NOT SPECIFIED | 4.510661611190209 |
| ASIAN | 3.890050023794174 |
| BLACK/AFRICAN AMERICAN | 6.867700040340424 |
| HISPANIC OR LATINO | 7.459633429845174 |
| UNABLE TO OBTAIN | 13.357000350952148 |
| AMERICAN INDIAN/ALASKA NATIVE FEDERALLY RECOGNIZED TRIBE | 11.33715045452118 |
| HISPANIC/LATINO - PUERTO RICAN | 3.243066656589508 |

<u>D. Summary of findings</u>

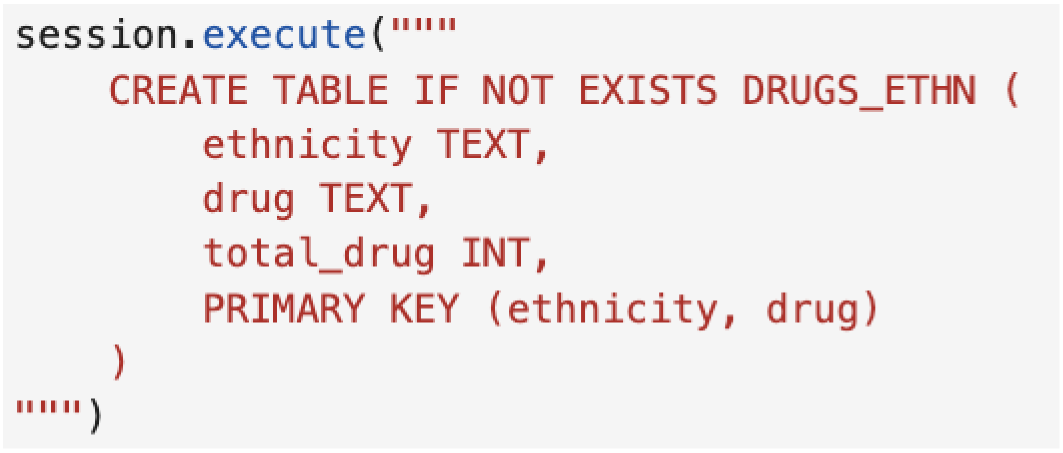
Women spend more time in the ICU than males as well as more than the average person. By ethnicity, Puerto Rican patients spend the least amount of time in the ICU, followed by Asian patients. American Indian/Alaska Native patients spend the most time in the ICU, followed by Hispanic or Latino patients. It is worth noting that Puerto Rican patients and Asian patients spend about one day less than the average length of stay in the ICU, while American Indian/Alaska Native and Hispanic or Latino patients spend more than 5 days more than the average length of stay in the ICU.

## Part 2: Non Relational

No copies of the AWS credentials file are stored on any publicly accessible location, nor is the file in any way shared with anyone outside of DATA_ENG 300 (Spring 2025).

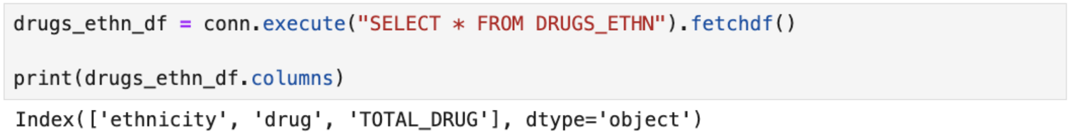
Question 1:

A. Design a Cassandra table for the specific analysis. Report your table creation query



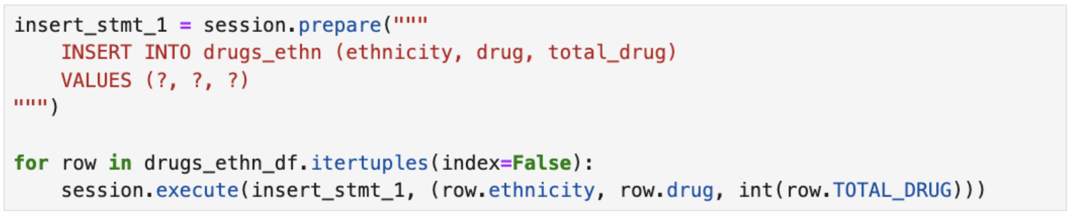
```
session.execute("""
    CREATE TABLE IF NOT EXISTS DRUGS_ETHN (
        ethnicity TEXT,
        drug TEXT,
        total_drug INT,
        PRIMARY KEY (ethnicity, drug)
    )
""")
```

B. Upload the data into the table to facilitate answering the question. Report your code for uploading the data.



```
drugs_ethn_df = conn.execute("SELECT * FROM DRUGS_ETHN").fetchdf()

print(drugs_ethn_df.columns)
```

```
Index(['ethnicity', 'drug', 'TOTAL_DRUG'], dtype='object')
```

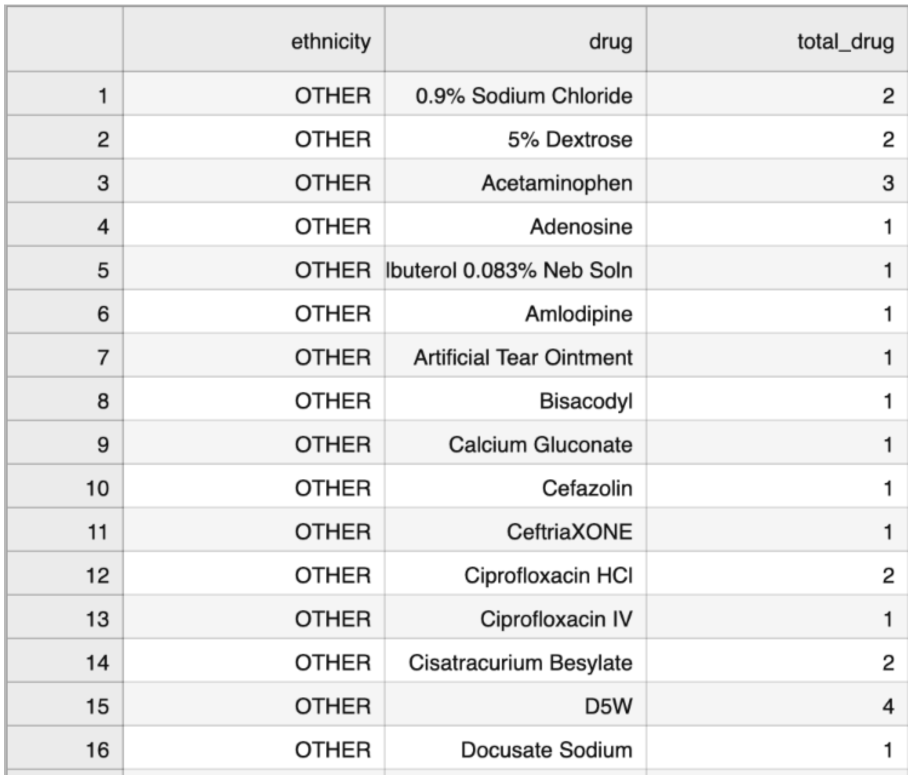


```
insert_stmt_1 = session.prepare("""
    INSERT INTO drugs_ethn (ethnicity, drug, total_drug)
    VALUES (?, ?, ?)
""")

for row in drugs_ethn_df.itertuples(index=False):
    session.execute(insert_stmt_1, (row.ethnicity, row.drug, int(row.TOTAL_DRUG)))
```

C. Report the query for extracting relevant data to answer the question. You may choose to not aggregate within Cassandra. If so, indicate your post-extraction analyses and include the code for reaching the final answer.

```python
import csv

rows = session.execute("SELECT * FROM drugs_ethn")
with open("drugs_ethn.csv", "w", newline="") as f:
    writer = csv.writer(f)
    writer.writerow(["ethnicity", "drug", "total_drug"])
    for row in rows:
        writer.writerow([row.ethnicity, row.drug, int(row.total_drug)])
```

<u>D. Verify that the extraction produces the desired data.</u>

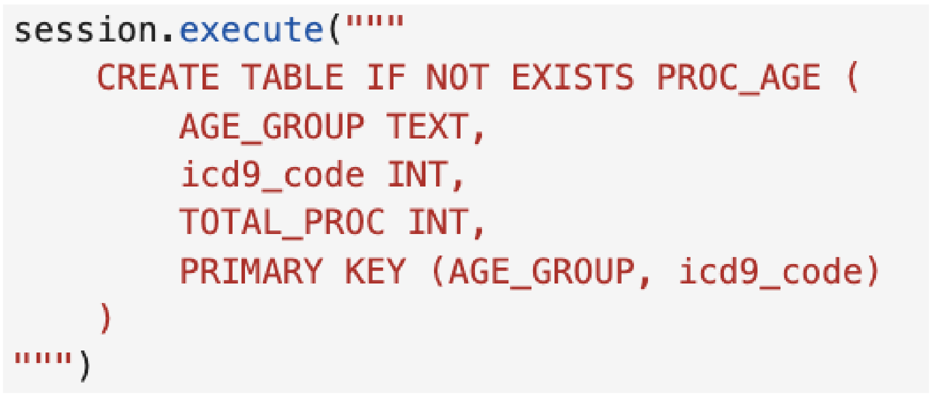


| | ethnicity | drug | total_drug |
|---|---|---|---|
| 1 | OTHER | 0.9% Sodium Chloride | 2 |
| 2 | OTHER | 5% Dextrose | 2 |
| 3 | OTHER | Acetaminophen | 3 |
| 4 | OTHER | Adenosine | 1 |
| 5 | OTHER | lbuterol 0.083% Neb Soln | 1 |
| 6 | OTHER | Amlodipine | 1 |
| 7 | OTHER | Artificial Tear Ointment | 1 |
| 8 | OTHER | Bisacodyl | 1 |
| 9 | OTHER | Calcium Gluconate | 1 |
| 10 | OTHER | Cefazolin | 1 |
| 11 | OTHER | CeftriaXONE | 1 |
| 12 | OTHER | Ciprofloxacin HCl | 2 |
| 13 | OTHER | Ciprofloxacin IV | 1 |
| 14 | OTHER | Cisatracurium Besylate | 2 |
| 15 | OTHER | D5W | 4 |
| 16 | OTHER | Docusate Sodium | 1 |

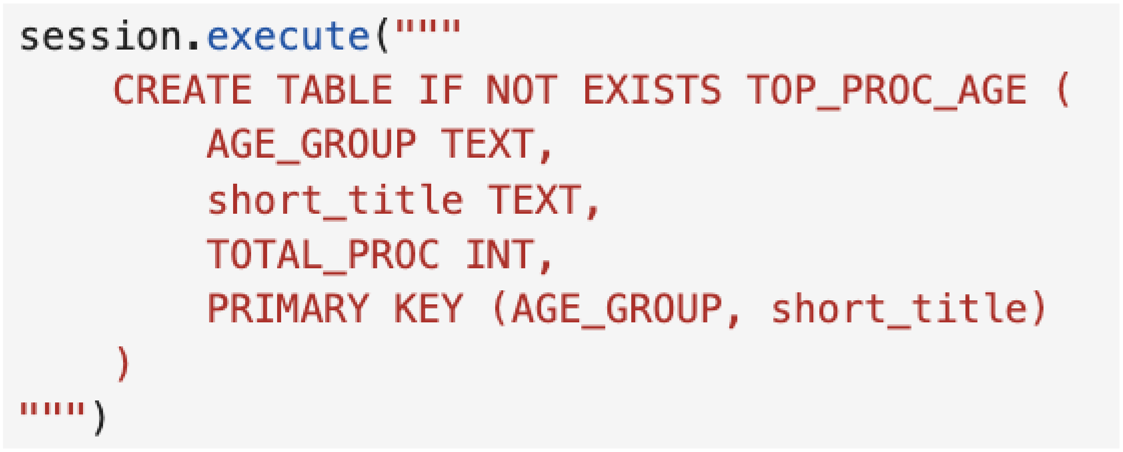
Data was exported to a csv table. Csv table was opened and the table was confirmed.

*Question 2:*

A. Design a Cassandra table for the specific analysis. Report your table creation query.

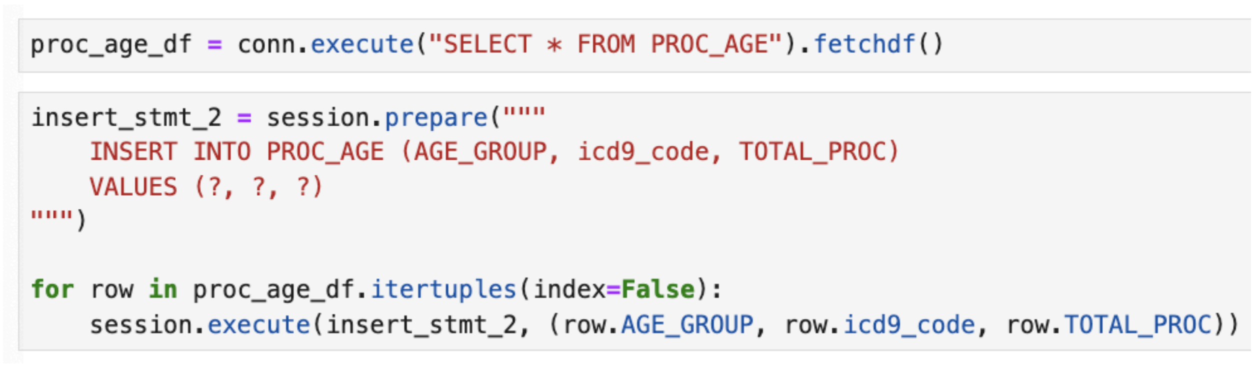


```
session.execute("""
    CREATE TABLE IF NOT EXISTS PROC_AGE (
        AGE_GROUP TEXT,
        icd9_code INT,
        TOTAL_PROC INT,
        PRIMARY KEY (AGE_GROUP, icd9_code)
    )
""")
```



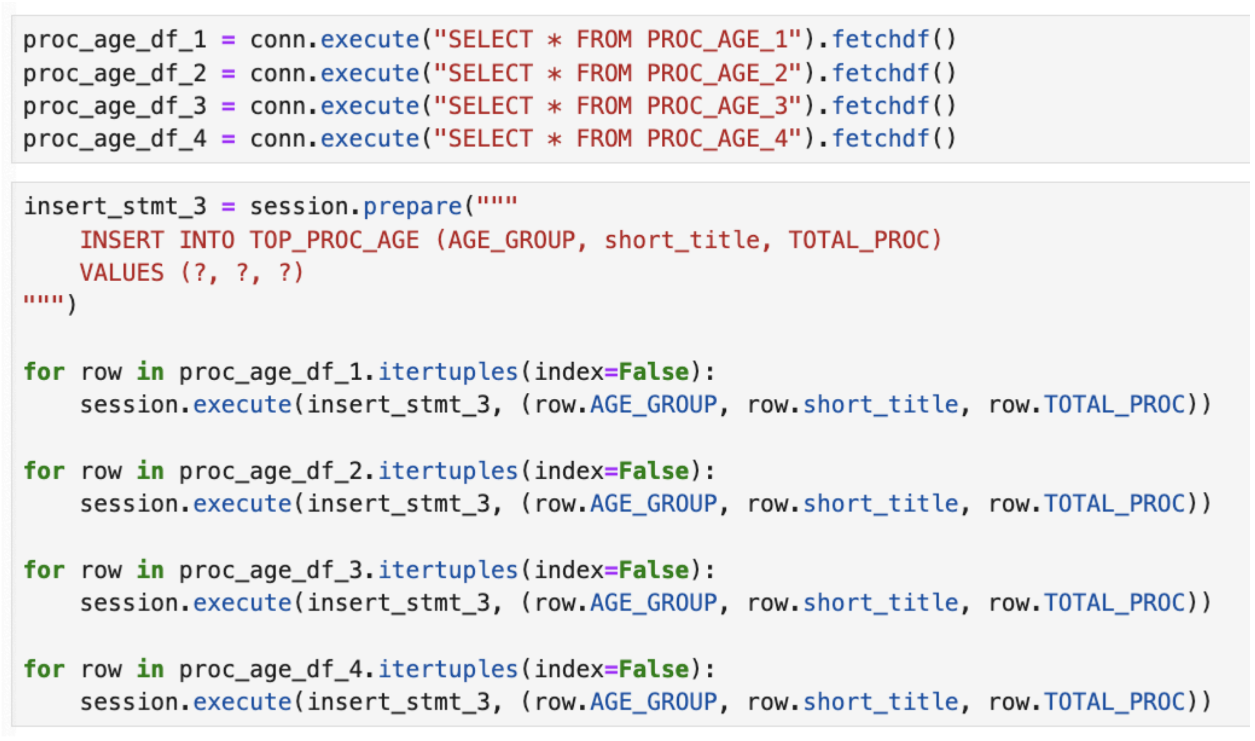
```
session.execute("""
    CREATE TABLE IF NOT EXISTS TOP_PROC_AGE (
        AGE_GROUP TEXT,
        short_title TEXT,
        TOTAL_PROC INT,
        PRIMARY KEY (AGE_GROUP, short_title)
    )
""")
```

B. Upload the data into the table to facilitate answering the question. Report your code for uploading the data.



```
proc_age_df = conn.execute("SELECT * FROM PROC_AGE").fetchdf()
```

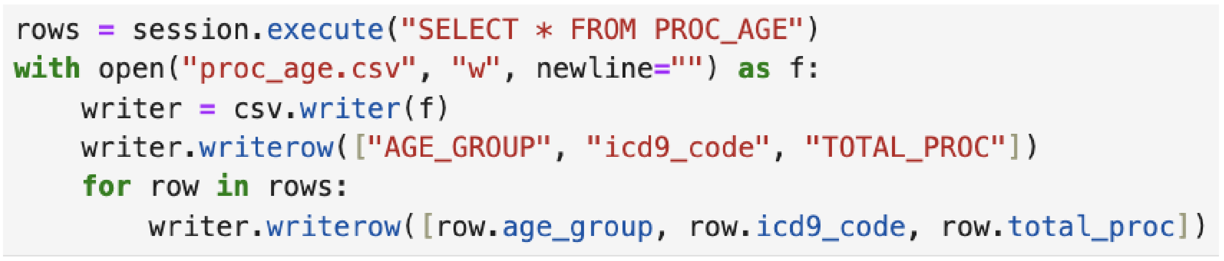
```
insert_stmt_2 = session.prepare("""
    INSERT INTO PROC_AGE (AGE_GROUP, icd9_code, TOTAL_PROC)
    VALUES (?, ?, ?)
""")

for row in proc_age_df.itertuples(index=False):
    session.execute(insert_stmt_2, (row.AGE_GROUP, row.icd9_code, row.TOTAL_PROC))
```



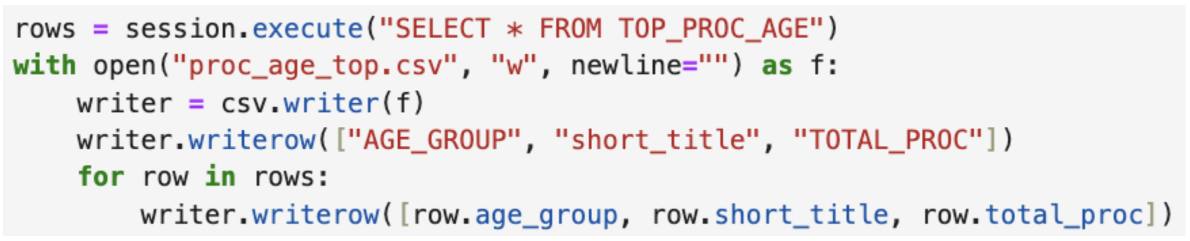
```
proc_age_df_1 = conn.execute("SELECT * FROM PROC_AGE_1").fetchdf()
proc_age_df_2 = conn.execute("SELECT * FROM PROC_AGE_2").fetchdf()
proc_age_df_3 = conn.execute("SELECT * FROM PROC_AGE_3").fetchdf()
proc_age_df_4 = conn.execute("SELECT * FROM PROC_AGE_4").fetchdf()
```

```
insert_stmt_3 = session.prepare("""
    INSERT INTO TOP_PROC_AGE (AGE_GROUP, short_title, TOTAL_PROC)
    VALUES (?, ?, ?)
""")

for row in proc_age_df_1.itertuples(index=False):
    session.execute(insert_stmt_3, (row.AGE_GROUP, row.short_title, row.TOTAL_PROC))

for row in proc_age_df_2.itertuples(index=False):
    session.execute(insert_stmt_3, (row.AGE_GROUP, row.short_title, row.TOTAL_PROC))

for row in proc_age_df_3.itertuples(index=False):
    session.execute(insert_stmt_3, (row.AGE_GROUP, row.short_title, row.TOTAL_PROC))

for row in proc_age_df_4.itertuples(index=False):
    session.execute(insert_stmt_3, (row.AGE_GROUP, row.short_title, row.TOTAL_PROC))
```

C. Report the query for extracting relevant data to answer the question. You may choose to not aggregate within Cassandra. If so, indicate your post-extraction analyses and include the code for reaching the final answer.

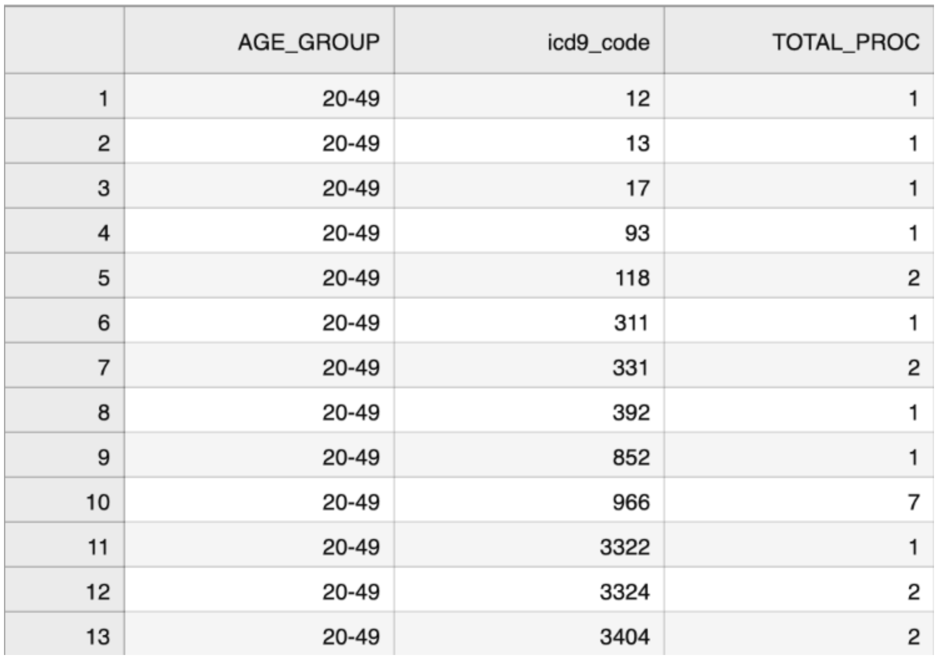


```
rows = session.execute("SELECT * FROM PROC_AGE")
with open("proc_age.csv", "w", newline="") as f:
    writer = csv.writer(f)
    writer.writerow(["AGE_GROUP", "icd9_code", "TOTAL_PROC"])
    for row in rows:
        writer.writerow([row.age_group, row.icd9_code, row.total_proc])
```

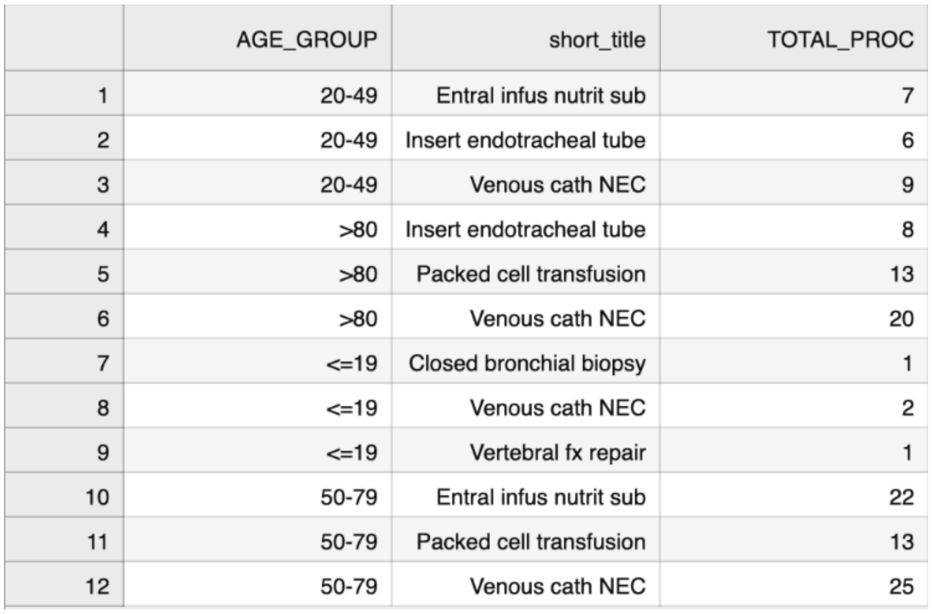


```
rows = session.execute("SELECT * FROM TOP_PROC_AGE")
with open("proc_age_top.csv", "w", newline="") as f:
    writer = csv.writer(f)
    writer.writerow(["AGE_GROUP", "short_title", "TOTAL_PROC"])
    for row in rows:
        writer.writerow([row.age_group, row.short_title, row.total_proc])
```

D. Verify that the extraction produces the desired data.



| | AGE_GROUP | icd9_code | TOTAL_PROC |
|---|---|---|---|
| 1 | 20-49 | 12 | 1 |
| 2 | 20-49 | 13 | 1 |
| 3 | 20-49 | 17 | 1 |
| 4 | 20-49 | 93 | 1 |
| 5 | 20-49 | 118 | 2 |
| 6 | 20-49 | 311 | 1 |
| 7 | 20-49 | 331 | 2 |
| 8 | 20-49 | 392 | 1 |
| 9 | 20-49 | 852 | 1 |
| 10 | 20-49 | 966 | 7 |
| 11 | 20-49 | 3322 | 1 |
| 12 | 20-49 | 3324 | 2 |
| 13 | 20-49 | 3404 | 2 |

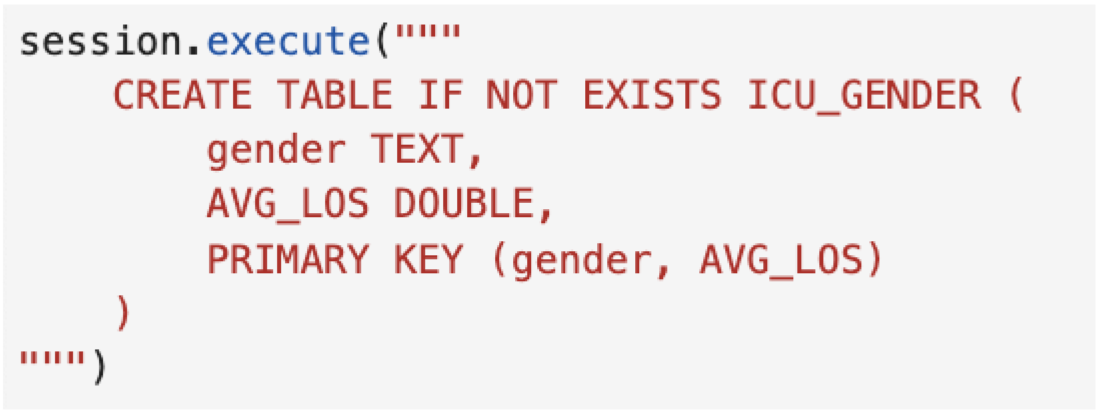


| | AGE_GROUP | short_title | TOTAL_PROC |
|---|---|---|---|
| 1 | 20-49 | Entral infus nutrit sub | 7 |
| 2 | 20-49 | Insert endotracheal tube | 6 |
| 3 | 20-49 | Venous cath NEC | 9 |
| 4 | >80 | Insert endotracheal tube | 8 |
| 5 | >80 | Packed cell transfusion | 13 |
| 6 | >80 | Venous cath NEC | 20 |
| 7 | <=19 | Closed bronchial biopsy | 1 |
| 8 | <=19 | Venous cath NEC | 2 |
| 9 | <=19 | Vertebral fx repair | 1 |
| 10 | 50-79 | Entral infus nutrit sub | 22 |
| 11 | 50-79 | Packed cell transfusion | 13 |
| 12 | 50-79 | Venous cath NEC | 25 |

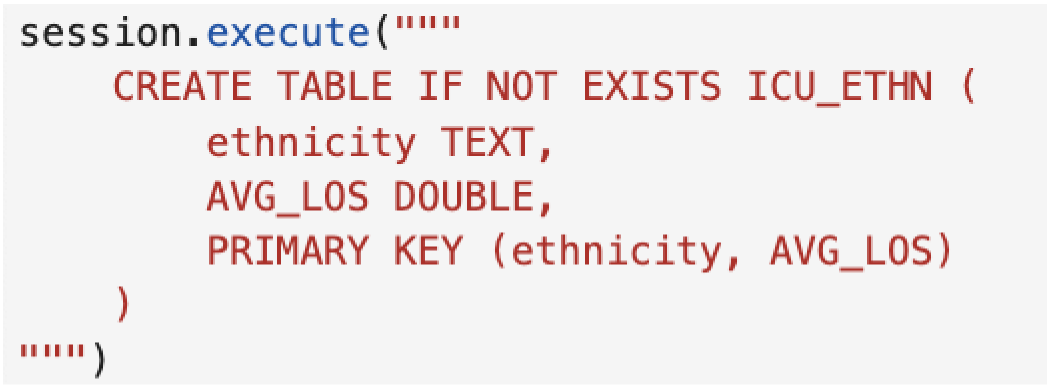
Data was exported to a csv to be confirmed.

*Question 3:*

A. Design a Cassandra table for the specific analysis. Report your table creation query.

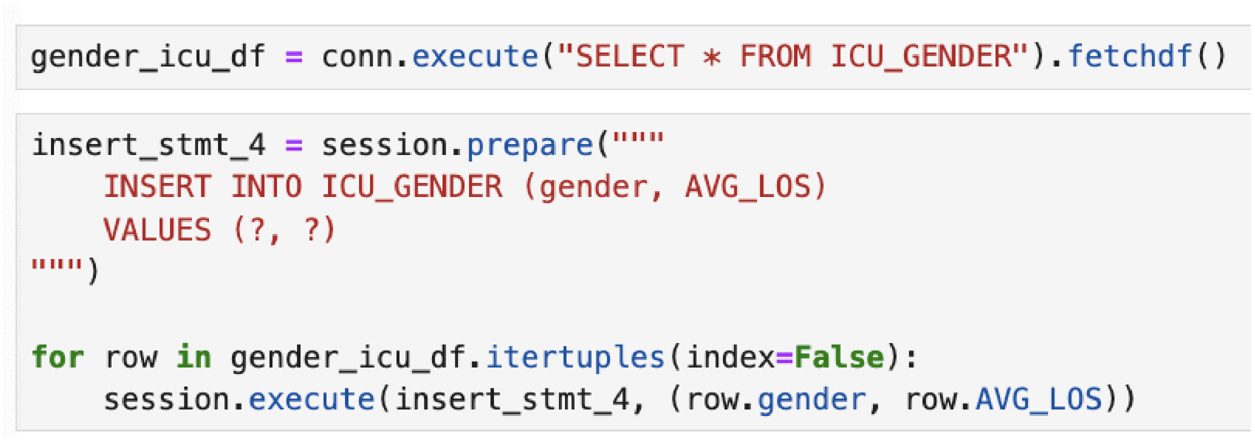


```
session.execute("""
    CREATE TABLE IF NOT EXISTS ICU_GENDER (
        gender TEXT,
        AVG_LOS DOUBLE,
        PRIMARY KEY (gender, AVG_LOS)
    )
""")
```



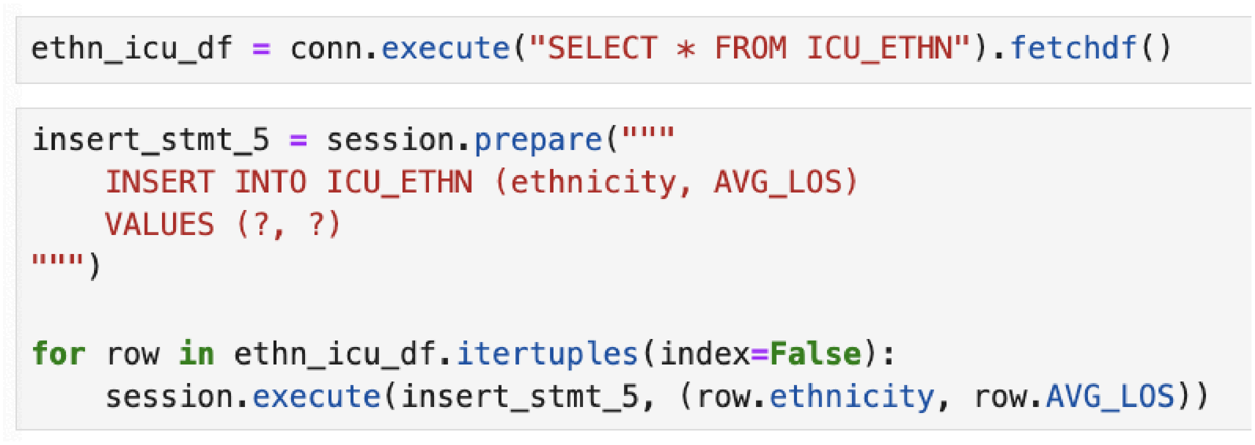
```
session.execute("""
    CREATE TABLE IF NOT EXISTS ICU_ETHN (
        ethnicity TEXT,
        AVG_LOS DOUBLE,
        PRIMARY KEY (ethnicity, AVG_LOS)
    )
""")
```

B. Upload the data into the table to facilitate answering the question. Report your code for uploading the data.



```
gender_icu_df = conn.execute("SELECT * FROM ICU_GENDER").fetchdf()
```

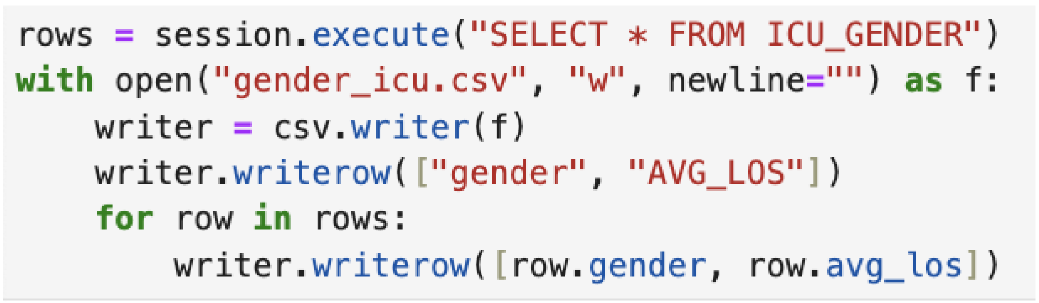
```
insert_stmt_4 = session.prepare("""
    INSERT INTO ICU_GENDER (gender, AVG_LOS)
    VALUES (?, ?)
""")

for row in gender_icu_df.itertuples(index=False):
    session.execute(insert_stmt_4, (row.gender, row.AVG_LOS))
```



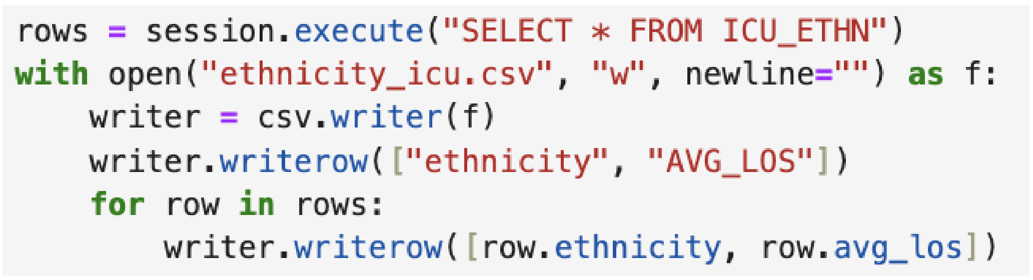
```python
ethn_icu_df = conn.execute("SELECT * FROM ICU_ETHN").fetchdf()
```

```python
insert_stmt_5 = session.prepare("""
    INSERT INTO ICU_ETHN (ethnicity, AVG_LOS)
    VALUES (?, ?)
""")

for row in ethn_icu_df.itertuples(index=False):
    session.execute(insert_stmt_5, (row.ethnicity, row.AVG_LOS))
```

C.  Report the query for extracting relevant data to answer the question. You may choose to not aggregate within Cassandra. If so, indicate your post-extraction analyses and include the code for reaching the final answer.

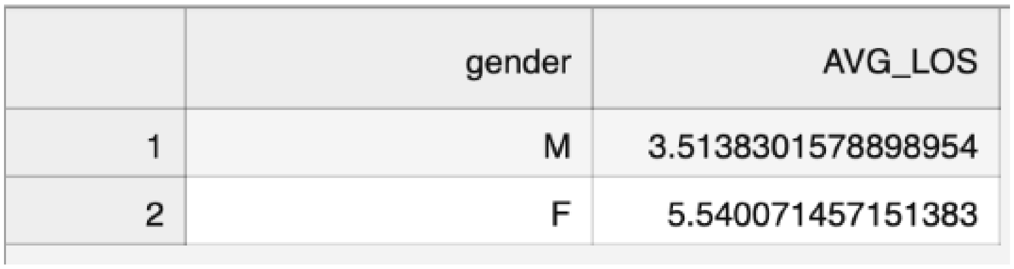


```python
rows = session.execute("SELECT * FROM ICU_GENDER")
with open("gender_icu.csv", "w", newline="") as f:
    writer = csv.writer(f)
    writer.writerow(["gender", "AVG_LOS"])
    for row in rows:
        writer.writerow([row.gender, row.avg_los])
```

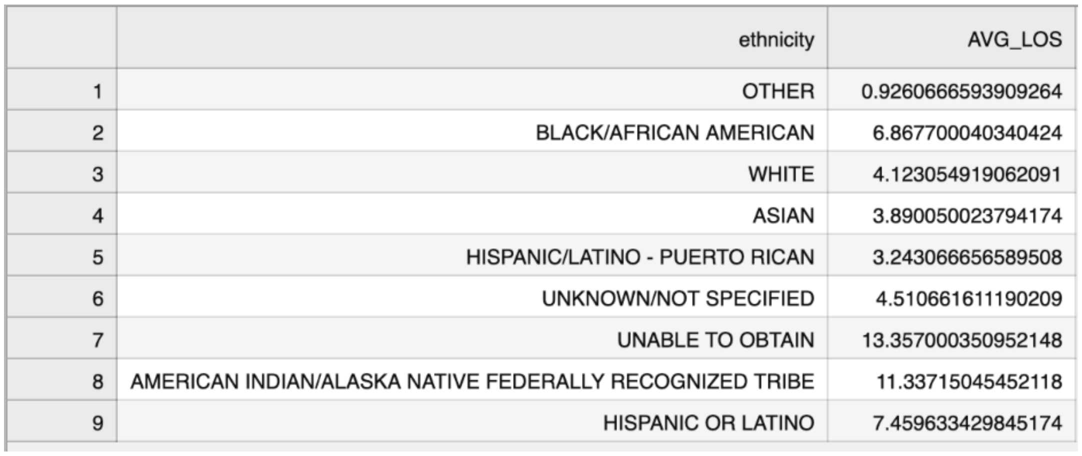


```python
rows = session.execute("SELECT * FROM ICU_ETHN")
with open("ethnicity_icu.csv", "w", newline="") as f:
    writer = csv.writer(f)
    writer.writerow(["ethnicity", "AVG_LOS"])
    for row in rows:
        writer.writerow([row.ethnicity, row.avg_los])
```

D. Verify that the extraction produces the desired data.



| | gender | AVG_LOS |
|---|---|---|
| 1 | M | 3.5138301578898954 |
| 2 | F | 5.540071457151383 |



| | ethnicity | AVG_LOS |
|---|---|---|
| 1 | OTHER | 0.9260666593909264 |
| 2 | BLACK/AFRICAN AMERICAN | 6.867700040340424 |
| 3 | WHITE | 4.123054919062091 |
| 4 | ASIAN | 3.890050023794174 |
| 5 | HISPANIC/LATINO - PUERTO RICAN | 3.243066656589508 |
| 6 | UNKNOWN/NOT SPECIFIED | 4.510661611190209 |
| 7 | UNABLE TO OBTAIN | 13.357000350952148 |
| 8 | AMERICAN INDIAN/ALASKA NATIVE FEDERALLY RECOGNIZED TRIBE | 11.33715045452118 |
| 9 | HISPANIC OR LATINO | 7.459633429845174 |

Data was exported to a csv to be confirmed.

## Generative AI Disclosure:

Homework was initially coded in google colab, where the autocomplete function was used in some instances. Specifically the following line of code: conn.sql('PRAGMA table_info(TABLE NAME);') was generated using this function to simply display the table. The autocomplete function was also used to generate the line of code:

WHEN CAST(STRFTIME('%Y', CAST(ADMISSIONS.ADMITTIME AS DATE)) AS INT) - CAST(STRFTIME('%Y', CAST(PATIENTS.DOB AS DATE)) AS INT) …

This line of code is seen in the creation of the table PROC_AGE in part 1, question two of the homework. This line of code basically extracts the year of the admissions time and the year of the patient's date of birth as integers in order to determine the patient's age at the time of admission.

ChatGPT was used sparingly, but was used mainly in part 2 of this homework assignment. I asked ChatGPT how I could take the previously created table in part 1 of the homework assignment and put it into the cassandra table. ChatGPT gave me the following line of code:

drugs_ethn_df = conn.execute("SELECT * FROM DRUGS_ETHN").fetchdf()

Which I then continued to use to convert the rest of the tables from part 1 of the homework into dataframes which allow the insertion of the dataframes into the cassandra tables.

ChatGPT also helped remedy the following error:
AttributeError: 'Row' object has no attribute 'AGE_GROUP'

Basically the fix was that when inserting information into the cassandra tables, you must use lowercase attribute names.