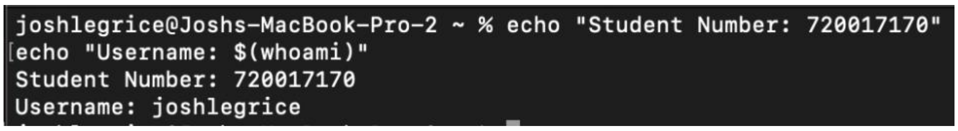
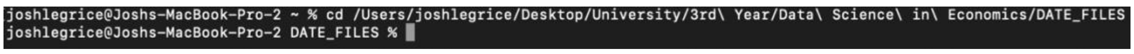
Question 1 – Bash

a) Using "echo", print your student number to the terminal. Additionally, print out your username from the operating system. Take a screenshot and include it in your submission



```
joshlegrice@Joshs-MacBook-Pro-2 ~ % echo "Student Number: 720017170"
[echo "Username: $(whoami)"
Student Number: 720017170
Username: joshlegrice
```

b) Move inside the directory DATE_FILES, which was provided along with unit 2 of the course, which you should store somewhere on your computer.



```
joshlegrice@Joshs-MacBook-Pro-2 ~ % cd /Users/joshlegrice/Desktop/University/3rd\ Year/Data\ Science\ in\ Economics/DATE_FILES
joshlegrice@Joshs-MacBook-Pro-2 DATE_FILES %
```

# cd = changes the current working directory to the file path given

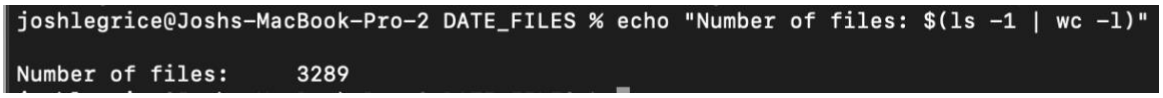
c) Count the number of files in this directory.

# ls -1 = Lists all files in the directory, one per line

| allows the output of the command before to be used as the input to the next command

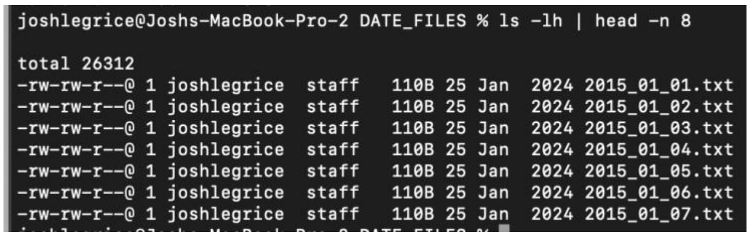
# wc -l = Counts the number of lines in the output of ls -1

$ = allow the code inside the brackets to be executed and not just echoed, like Python f string



```
joshlegrice@Joshs-MacBook-Pro-2 DATE_FILES % echo "Number of files: $(ls -1 | wc -l)"

Number of files:     3289
```

d) Print the names of the first 8 files in this directory, along with information about their ownership, date, and size.



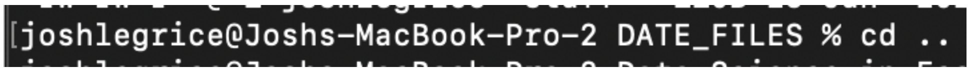
```
joshlegrice@Joshs-MacBook-Pro-2 DATE_FILES % ls -lh | head -n 8

total 26312
-rw-rw-r--@ 1 joshlegrice  staff   110B 25 Jan  2024 2015_01_01.txt
-rw-rw-r--@ 1 joshlegrice  staff   110B 25 Jan  2024 2015_01_02.txt
-rw-rw-r--@ 1 joshlegrice  staff   110B 25 Jan  2024 2015_01_03.txt
-rw-rw-r--@ 1 joshlegrice  staff   110B 25 Jan  2024 2015_01_04.txt
-rw-rw-r--@ 1 joshlegrice  staff   110B 25 Jan  2024 2015_01_05.txt
-rw-rw-r--@ 1 joshlegrice  staff   110B 25 Jan  2024 2015_01_06.txt
-rw-rw-r--@ 1 joshlegrice  staff   110B 25 Jan  2024 2015_01_07.txt
```

# ls -lh = lists the contents of the directory in long (l) format and human-readable (h).

| allows the output of the command before to be used as the input to the next command

# head -n 8 = only displays the first 8 lines of the output

e) Move to the parent directory of this folder

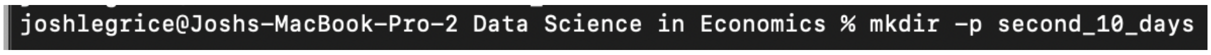


```
[joshlegrice@Joshs-MacBook-Pro-2 DATE_FILES % cd ..
```

# cd = change directory to the file path given

# .. = indicates the parent directory of the current file

f) Create a new directory there, named second_10_days



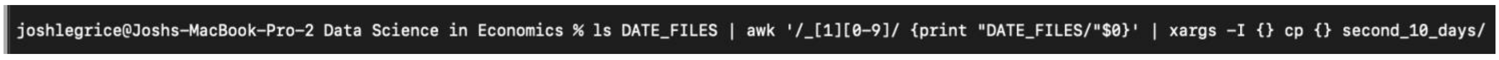
```
joshlegrice@Joshs-MacBook-Pro-2 Data Science in Economics % mkdir -p second_10_days
```

# mkdir = command to make a new directory

# -p = ensures parent directories exist

# second_10_days = the name of the new directory

g) Copy from the DATE_FILES directory the files that are related to the days 10-19 of every month to the newly created directory.



```
joshlegrice@Joshs-MacBook-Pro-2 Data Science in Economics % ls DATE_FILES | awk '/_[1][0-9]/ {print "DATE_FILES/"$0}' | xargs -I {} cp {} second_10_days/
```

# ls DATE_FILES = prints the contents of the directory DATE_FILES

# awk '/_[1] [0-9]/ {print "DATE_FILES/" $0}'

  # /_[1] [0-9]/ = Regular expression to find dates that have _10 to _19

  # {print "DATE_FILES/" $0} = prints out the entire file path of the selected files

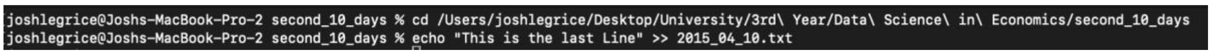
# xargs -I {} cp {} second_10_days/

  # xargs = processes each file one by one

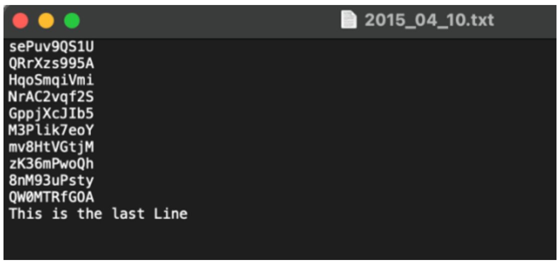
  # -I {} = allows {} to be replaced with filename

  # cp {} second_10_days/ = copies selected file to second_10_days

h) Move inside second_10_days directory, and append the line "This is the last Line" to the end of file 2015_04_10



```
joshlegrice@Joshs-MacBook-Pro-2 second_10_days % cd /Users/joshlegrice/Desktop/University/3rd\ Year/Data\ Science\ in\ Economics/second_10_days
joshlegrice@Joshs-MacBook-Pro-2 second_10_days % echo "This is the last Line" >> 2015_04_10.txt
```



2015_04_10.txt

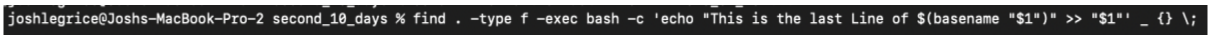
```
sePuv9QS1U
QRrXzs995A
HqoSmqiVmi
NrAC2vqf2S
GppjXcJIb5
M3Plik7eoY
mv8HtVGtjM
zK36mPwoQh
8nM93uPsty
QW0MTRfGOA
This is the last Line
```

# cd ...... = moves to the file path shown

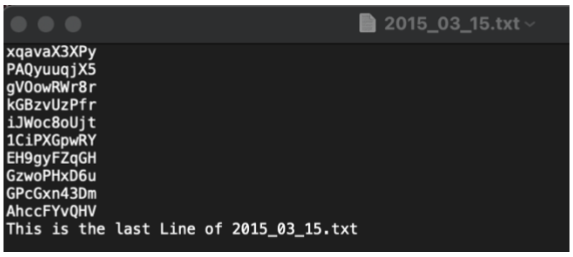
# echo "This is the last Line" = outputs the text This is the last Line

# >> 2015_04_10.txt = appends the echoed text into the file 2015_04_10.txt

i) Write a one-line command to append the line "This is the last Line of X", where X is the name of the file, to the end of every file in the directory second_10_days



```
joshlegrice@Joshs-MacBook-Pro-2 second_10_days % find . -type f -exec bash -c 'echo "This is the last Line of $(basename "$1")" >> "$1"' _ {} \;
```



2015_03_15.txt

```
xqavaX3XPy
PAQyuuqjX5
gVOowRWr8r
kGBzvUzPfr
iJWoc8oUjt
1CiPXGpwRY
EH9gyFZqGH
GzwoPHxD6u
GPcGxn43Dm
AhccFYvQHV
This is the last Line of 2015_03_15.txt
```

# find . = recursively searches the current directory

# -type f = only searches files not directories

# -exec bash – c = runs a bash command for each file found

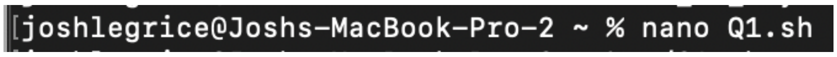
# echo "This is the last Line of $(basename "$1") = outputs 'This is the last Line of (only the name of the file)'

>> "$1"' _ {} \; = appends the output of the echo into the file.

j) Using Bash: create a bash file Q1.sh. Write your code from (i) to it. Run the file Q1.sh including a screenshot showing how this runs on your system. Please explain any steps needed to run this file.

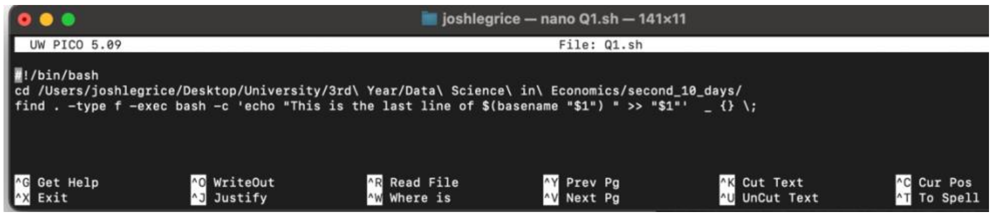
Had to place the .sh file into another directory as if placed in the second_10_days directory, it would write into the Q1.sh file with ' this is the last Line of Q1.sh'

First step = Open an .sh file



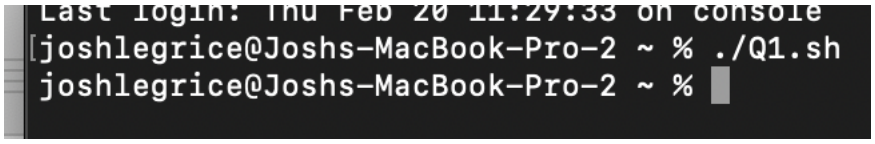
```
joshlegrice@Joshs-MacBook-Pro-2 ~ % nano Q1.sh
```

Second step = Write code in .sh file



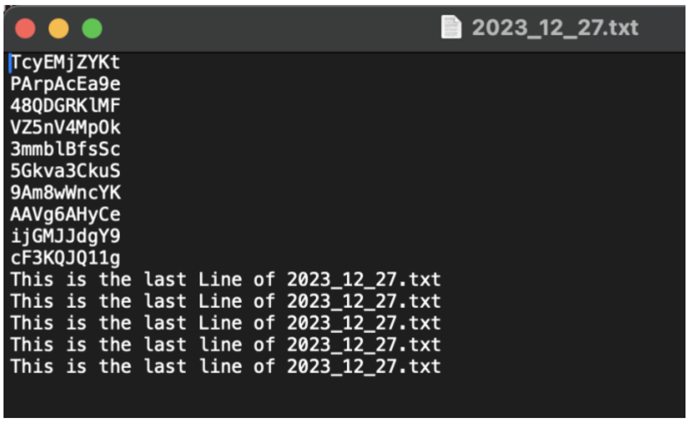
```
#!/bin/bash
cd /Users/joshlegrice/Desktop/University/3rd\ Year/Data\ Science\ in\ Economics/second_10_days/
find . -type f -exec bash -c 'echo "This is the last line of $(basename "$1") " >> "$1"'  _ {} \;
```

Final Step = Run .sh file



```
Last login: Thu Feb 20 11:29:33 on console
[joshlegrice@Joshs-MacBook-Pro-2 ~ % ./Q1.sh
joshlegrice@Joshs-MacBook-Pro-2 ~ %
```

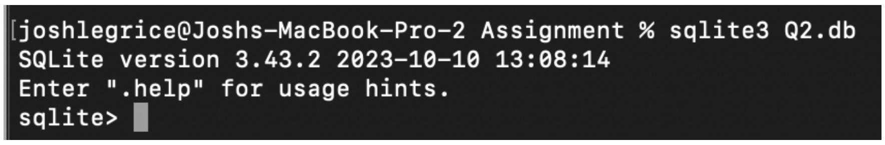
Output in an example file – I ran it a lot to make sure it worked



```
TcyEMjZYKt
PArpAcEa9e
48QDGRKlMF
VZ5nV4MpOk
3mmblBfsSc
5Gkva3CkuS
9Am8wWncYK
AAVg6AHyCe
ijGMJJdgY9
cF3KQJQ11g
This is the last Line of 2023_12_27.txt
This is the last Line of 2023_12_27.txt
This is the last Line of 2023_12_27.txt
This is the last line of 2023_12_27.txt
This is the last line of 2023_12_27.txt
```

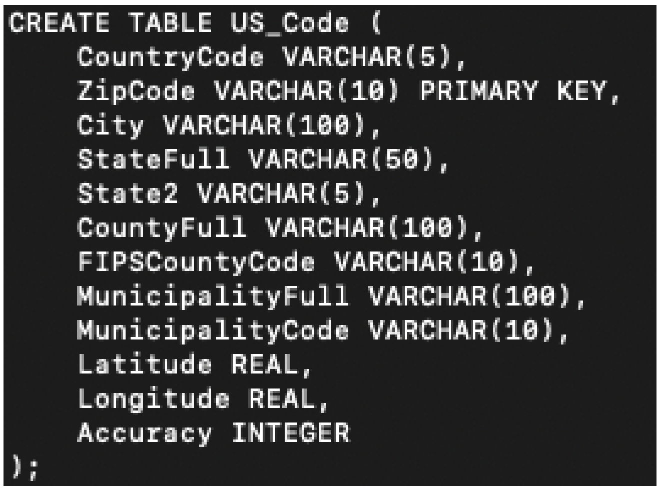
Question 2 – SQL

a) **Create a new database in SQLite named Q2.db**

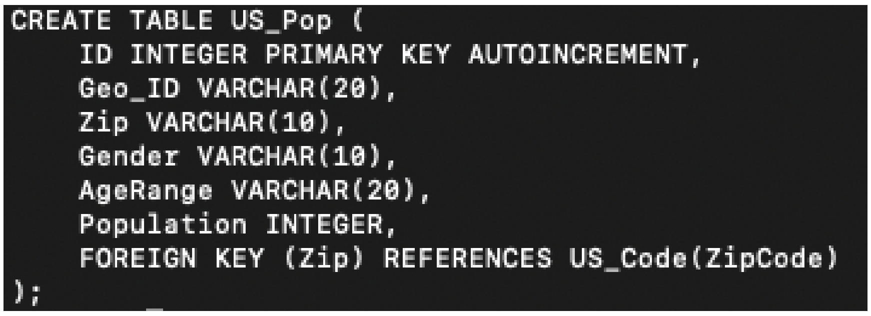


```
[joshlegrice@Joshs-MacBook-Pro-2 Assignment % sqlite3 Q2.db
SQLite version 3.43.2 2023-10-10 13:08:14
Enter ".help" for usage hints.
sqlite> 
```

b) **Create two tables named US_Code and US_Pop with column headings that match these two data frames**

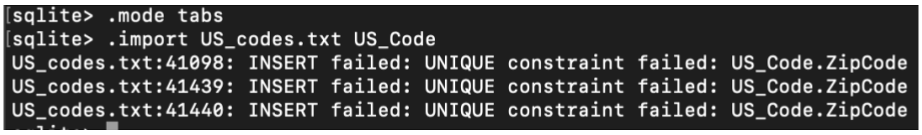


```
CREATE TABLE US_Code (
    CountryCode VARCHAR(5),
    ZipCode VARCHAR(10) PRIMARY KEY,
    City VARCHAR(100),
    StateFull VARCHAR(50),
    State2 VARCHAR(5),
    CountyFull VARCHAR(100),
    FIPSCountyCode VARCHAR(10),
    MunicipalityFull VARCHAR(100),
    MunicipalityCode VARCHAR(10),
    Latitude REAL,
    Longitude REAL,
    Accuracy INTEGER
);
```



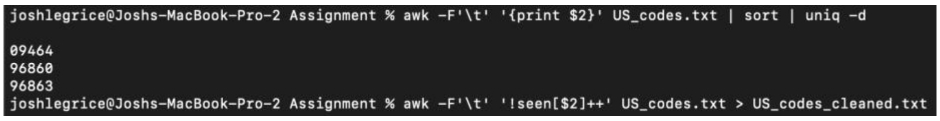
```
CREATE TABLE US_Pop (
    ID INTEGER PRIMARY KEY AUTOINCREMENT,
    Geo_ID VARCHAR(20),
    Zip VARCHAR(10),
    Gender VARCHAR(10),
    AgeRange VARCHAR(20),
    Population INTEGER,
    FOREIGN KEY (Zip) REFERENCES US_Code(ZipCode)
);
```

c) **Insert the data from the two files into the two tables. Make sure you don't insert the column heading from the file US_population.csv. Explain how you did this.**



```
[sqlite> .mode tabs
[sqlite> .import US_codes.txt US_Code
US_codes.txt:41098: INSERT failed: UNIQUE constraint failed: US_Code.ZipCode
US_codes.txt:41439: INSERT failed: UNIQUE constraint failed: US_Code.ZipCode
US_codes.txt:41440: INSERT failed: UNIQUE constraint failed: US_Code.ZipCode
```

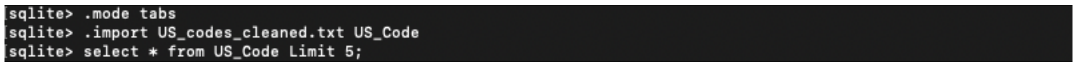
# Had to remove duplicates before loading in the US_Code data due to the above error



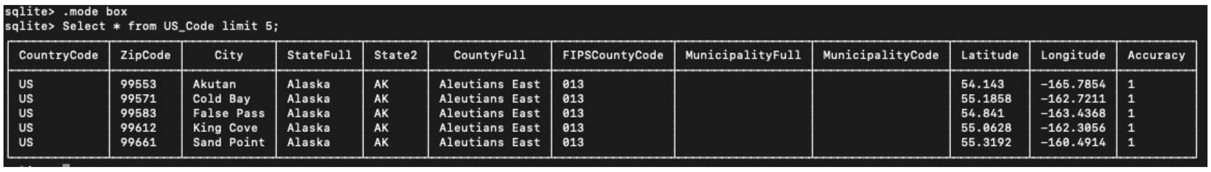
```
joshlegrice@Joshs-MacBook-Pro-2 Assignment % awk -F'\t' '{print $2}' US_codes.txt | sort | uniq -d

09464
96860
96863
joshlegrice@Joshs-MacBook-Pro-2 Assignment % awk -F'\t' '!seen[$2]++' US_codes.txt > US_codes_cleaned.txt
```

# awk -F'\t' '{print $2}' US_codes.txt = Extracts the second column from the .txt file which is ZipCode
# sort = sorts the values within the column
# uniq -d = identifies and prints only the duplicate values = Used to visualise all duplicate values

# awk -F'\t' '!seen[$2]++' US_codes.txt = collects all the non-duplicates in the column into an array
# > US_codes_cleaned.txt = saves the contents of the previous output into a new file



```
[sqlite> .mode tabs
[sqlite> .import US_codes_cleaned.txt US_Code
[sqlite> select * from US_Code Limit 5;
```

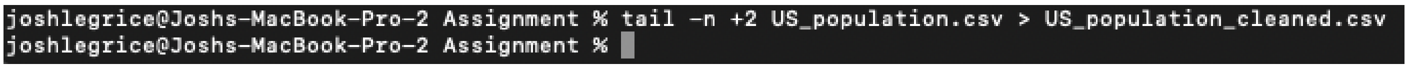


```
sqlite> .mode box
sqlite> Select * from US_Code limit 5;
```

CountryCode	ZipCode	City	StateFull	State2	CountyFull	FIPSCountyCode	MunicipalityFull	MunicipalityCode	Latitude	Longitude	Accuracy
US	99553	Akutan	Alaska	AK	Aleutians East	013			54.143	-165.7854	1
US	99571	Cold Bay	Alaska	AK	Aleutians East	013			55.1858	-162.7211	1
US	99583	False Pass	Alaska	AK	Aleutians East	013			54.841	-163.4368	1
US	99612	King Cove	Alaska	AK	Aleutians East	013			55.0628	-162.3056	1
US	99661	Sand Point	Alaska	AK	Aleutians East	013			55.3192	-160.4914	1

# .mode tabs to set the delimiter to tabs to distinguish columns

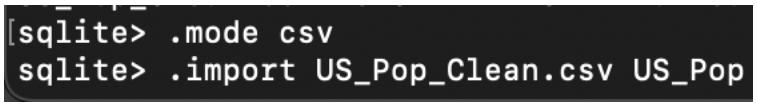
# Remove headers from the US_populations.csv



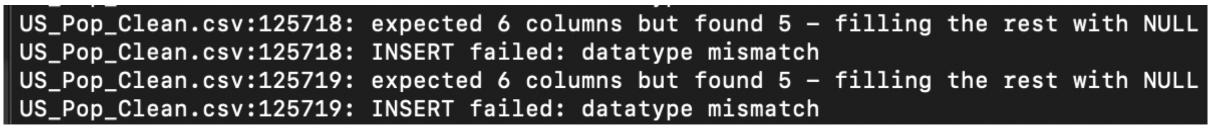
```
joshlegrice@Joshs-MacBook-Pro-2 Assignment % tail -n +2 US_population.csv > US_population_cleaned.csv
joshlegrice@Joshs-MacBook-Pro-2 Assignment % 
```

# tail -n +2 = starts at line 2 and collects all rows
# > US_population_cleaned.csv = moves the new data into the new file

# I had trouble with importing the data straight into the US_Pop table due to this error

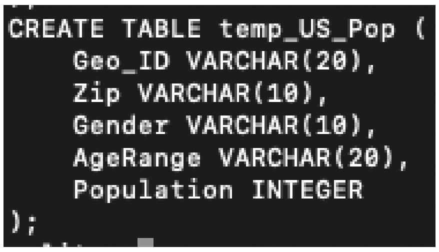


```
sqlite> .mode csv
sqlite> .import US_Pop_Clean.csv US_Pop
```

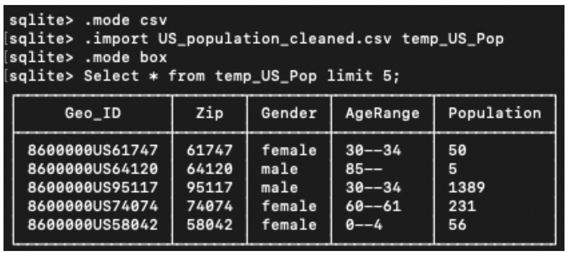


```
US_Pop_Clean.csv:125718: expected 6 columns but found 5 - filling the rest with NULL
US_Pop_Clean.csv:125718: INSERT failed: datatype mismatch
US_Pop_Clean.csv:125719: expected 6 columns but found 5 - filling the rest with NULL
US_Pop_Clean.csv:125719: INSERT failed: datatype mismatch
```

# So, I imported the data into a temporary table and then copied the data into US_Pop

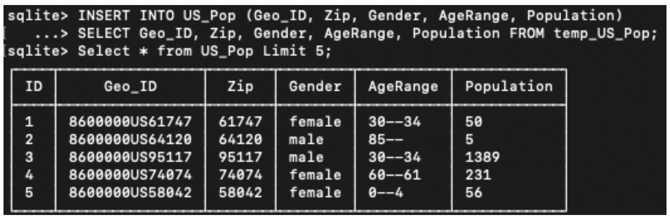


```
CREATE TABLE temp_US_Pop (
    Geo_ID VARCHAR(20),
    Zip VARCHAR(10),
    Gender VARCHAR(10),
    AgeRange VARCHAR(20),
    Population INTEGER
);
```



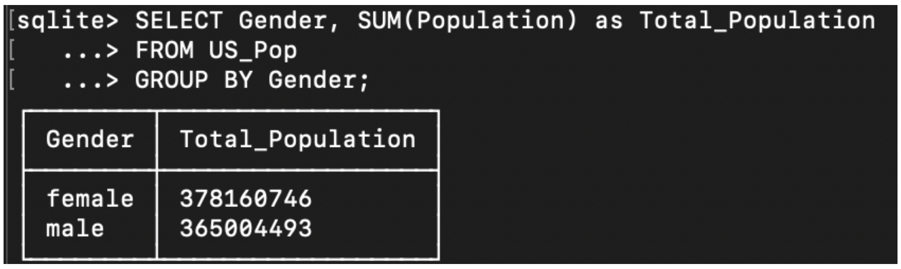
```
sqlite> .mode csv
sqlite> .import US_population_cleaned.csv temp_US_Pop
sqlite> .mode box
sqlite> Select * from temp_US_Pop limit 5;
┌──────────────┬───────┬────────┬──────────┬────────────┐
│    Geo_ID    │  Zip  │ Gender │ AgeRange │ Population │
├──────────────┼───────┼────────┼──────────┼────────────┤
│ 8600000US61747 │ 61747 │ female │ 30--34   │ 50         │
│ 8600000US64120 │ 64120 │ male   │ 85--     │ 5          │
│ 8600000US95117 │ 95117 │ male   │ 30--34   │ 1389       │
│ 8600000US74074 │ 74074 │ female │ 60--61   │ 231        │
│ 8600000US58042 │ 58042 │ female │ 0--4     │ 56         │
└──────────────┴───────┴────────┴──────────┴────────────┘
```

# Inserting data from temp table to US_Pop



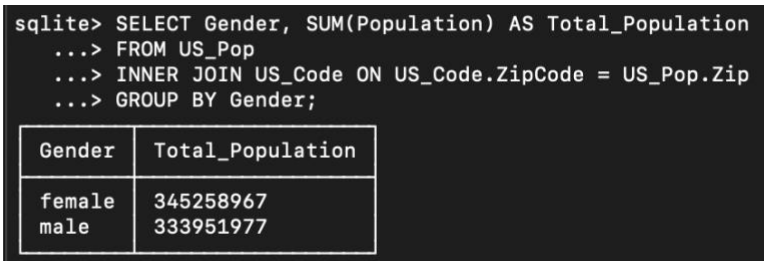
```
sqlite> INSERT INTO US_Pop (Geo_ID, Zip, Gender, AgeRange, Population)
   ...> SELECT Geo_ID, Zip, Gender, AgeRange, Population FROM temp_US_Pop;
sqlite> Select * from US_Pop Limit 5;
┌────┬────────────────┬───────┬────────┬──────────┬────────────┐
│ ID │     Geo_ID     │  Zip  │ Gender │ AgeRange │ Population │
├────┼────────────────┼───────┼────────┼──────────┼────────────┤
│ 1  │ 8600000US61747 │ 61747 │ female │ 30--34   │ 50         │
│ 2  │ 8600000US64120 │ 64120 │ male   │ 85--     │ 5          │
│ 3  │ 8600000US95117 │ 95117 │ male   │ 30--34   │ 1389       │
│ 4  │ 8600000US74074 │ 74074 │ female │ 60--61   │ 231        │
│ 5  │ 8600000US58042 │ 58042 │ female │ 0--4     │ 56         │
└────┴────────────────┴───────┴────────┴──────────┴────────────┘
```

d) Write an SQL query to print the total population per gender (using the US_Pop table only)



```
sqlite> SELECT Gender, SUM(Population) as Total_Population
   ...> FROM US_Pop
   ...> GROUP BY Gender;
┌────────┬──────────────────┐
│ Gender │ Total_Population │
├────────┼──────────────────┤
│ female │ 378160746        │
│ male   │ 365004493        │
└────────┴──────────────────┘
```

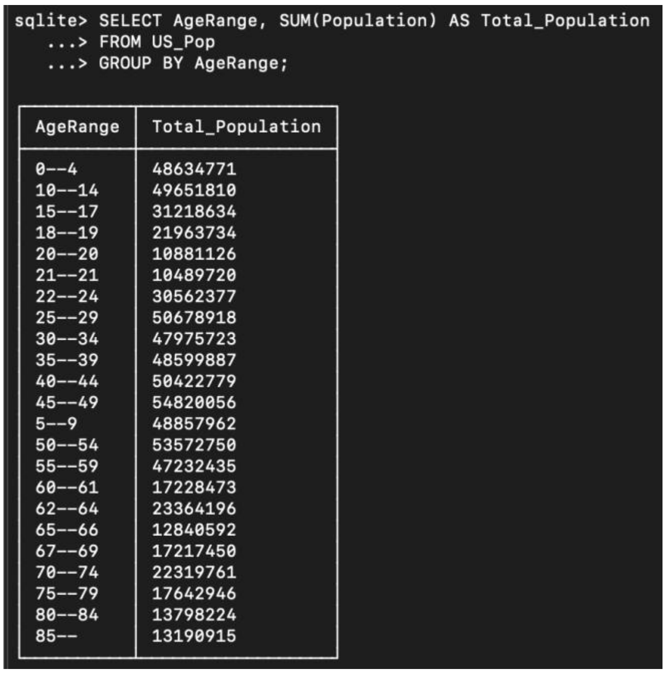
e) **Write an SQL query to print the total population per gender but join the two tables. If you see any difference in your results between this question and part (d), explain why this occurs.**



```
sqlite> SELECT Gender, SUM(Population) AS Total_Population
   ...> FROM US_Pop
   ...> INNER JOIN US_Code ON US_Code.ZipCode = US_Pop.Zip
   ...> GROUP BY Gender;
┌────────┬──────────────────┐
│ Gender │ Total_Population │
├────────┼──────────────────┤
│ female │ 345258967        │
│ male   │ 333951977        │
└────────┴──────────────────┘
```

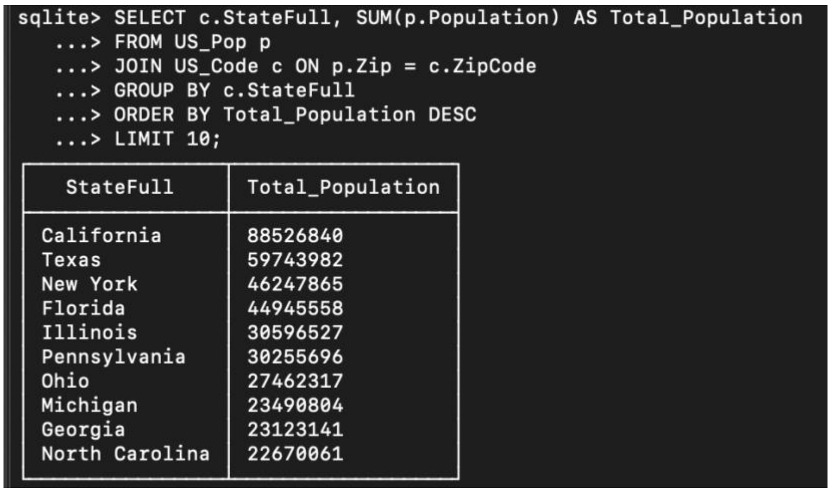
The difference is because INNER JOIN only includes records where zip codes exist in both US_Pop and US_Code, excluding unmatched zip codes from US_Pop. This results in a lower total population in part (e) compared to part (d).

f) **Write an SQL query to print the total population per age group (use the US_Pop table only).**



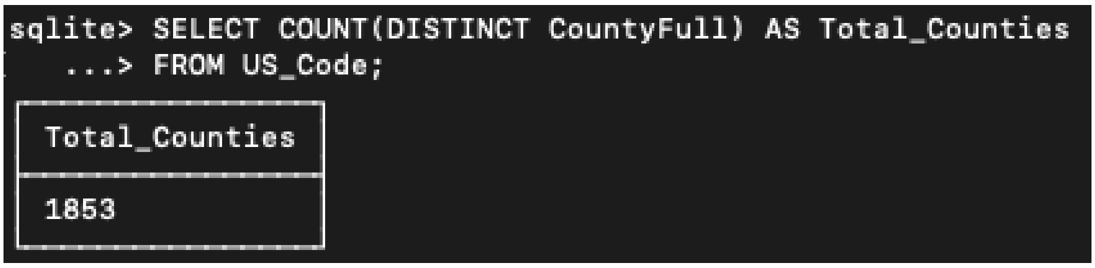
```
sqlite> SELECT AgeRange, SUM(Population) AS Total_Population
   ...> FROM US_Pop
   ...> GROUP BY AgeRange;

┌──────────┬──────────────────┐
│ AgeRange │ Total_Population │
├──────────┼──────────────────┤
│ 0--4     │ 48634771         │
│ 10--14   │ 49651810         │
│ 15--17   │ 31218634         │
│ 18--19   │ 21963734         │
│ 20--20   │ 10881126         │
│ 21--21   │ 10489720         │
│ 22--24   │ 30562377         │
│ 25--29   │ 50678918         │
│ 30--34   │ 47975723         │
│ 35--39   │ 48599887         │
│ 40--44   │ 50422779         │
│ 45--49   │ 54820056         │
│ 5--9     │ 48857962         │
│ 50--54   │ 53572750         │
│ 55--59   │ 47232435         │
│ 60--61   │ 17228473         │
│ 62--64   │ 23364196         │
│ 65--66   │ 12840592         │
│ 67--69   │ 17217450         │
│ 70--74   │ 22319761         │
│ 75--79   │ 17642946         │
│ 80--84   │ 13798224         │
│ 85--     │ 13190915         │
└──────────┴──────────────────┘
```

g) Write an SQL query to print the Top 10 largest states (full name) in terms of population size



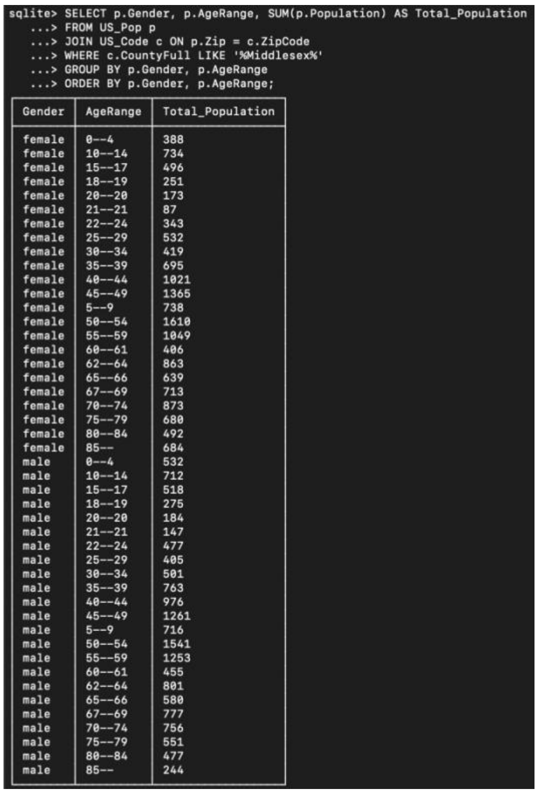
```
sqlite> SELECT c.StateFull, SUM(p.Population) AS Total_Population
   ...> FROM US_Pop p
   ...> JOIN US_Code c ON p.Zip = c.ZipCode
   ...> GROUP BY c.StateFull
   ...> ORDER BY Total_Population DESC
   ...> LIMIT 10;
┌────────────────┬──────────────────┐
│   StateFull    │ Total_Population │
├────────────────┼──────────────────┤
│ California     │ 88526840         │
│ Texas          │ 59743982         │
│ New York       │ 46247865         │
│ Florida        │ 44945558         │
│ Illinois       │ 30596527         │
│ Pennsylvania   │ 30255696         │
│ Ohio           │ 27462317         │
│ Michigan       │ 23490804         │
│ Georgia        │ 23123141         │
│ North Carolina │ 22670061         │
└────────────────┴──────────────────┘
```

h) Write an SQL query to print the number of existing counties (not countries) in the database



```
sqlite> SELECT COUNT(DISTINCT CountyFull) AS Total_Counties
   ...> FROM US_Code;
┌────────────────┐
│ Total_Counties │
├────────────────┤
│ 1853           │
└────────────────┘
```

i) Write an SQL query to print the total population per gender and age group for any counties containing "Middlesex" in their name.



```
sqlite> SELECT p.Gender, p.AgeRange, SUM(p.Population) AS Total_Population
   ...> FROM US_Pop p
   ...> JOIN US_Code c ON p.Zip = c.ZipCode
   ...> WHERE c.CountyFull LIKE '%Middlesex%'
   ...> GROUP BY p.Gender, p.AgeRange
   ...> ORDER BY p.Gender, p.AgeRange;
┌────────┬──────────┬──────────────────┐
│ Gender │ AgeRange │ Total_Population │
├────────┼──────────┼──────────────────┤
│ female │ 0--4     │ 388              │
│ female │ 10--14   │ 734              │
│ female │ 15--17   │ 496              │
│ female │ 18--19   │ 251              │
│ female │ 20--20   │ 173              │
│ female │ 21--21   │ 87               │
│ female │ 22--24   │ 343              │
│ female │ 25--29   │ 532              │
│ female │ 30--34   │ 419              │
│ female │ 35--39   │ 695              │
│ female │ 40--44   │ 1021             │
│ female │ 45--49   │ 1365             │
│ female │ 5--9     │ 738              │
│ female │ 50--54   │ 1610             │
│ female │ 55--59   │ 1049             │
│ female │ 60--61   │ 406              │
│ female │ 62--64   │ 863              │
│ female │ 65--66   │ 639              │
│ female │ 67--69   │ 713              │
│ female │ 70--74   │ 873              │
│ female │ 75--79   │ 680              │
│ female │ 80--84   │ 492              │
│ female │ 85--     │ 684              │
│ male   │ 0--4     │ 532              │
│ male   │ 10--14   │ 712              │
│ male   │ 15--17   │ 518              │
│ male   │ 18--19   │ 275              │
│ male   │ 20--20   │ 184              │
│ male   │ 21--21   │ 147              │
│ male   │ 22--24   │ 477              │
│ male   │ 25--29   │ 405              │
│ male   │ 30--34   │ 501              │
│ male   │ 35--39   │ 763              │
│ male   │ 40--44   │ 976              │
│ male   │ 45--49   │ 1261             │
│ male   │ 5--9     │ 716              │
│ male   │ 50--54   │ 1541             │
│ male   │ 55--59   │ 1253             │
│ male   │ 60--61   │ 455              │
│ male   │ 62--64   │ 801              │
│ male   │ 65--66   │ 580              │
│ male   │ 67--69   │ 777              │
│ male   │ 70--74   │ 756              │
│ male   │ 75--79   │ 551              │
│ male   │ 80--84   │ 477              │
│ male   │ 85--     │ 244              │
└────────┴──────────┴──────────────────┘
```