Homework #1

Instructions:

--You will essentially copy your queries (and output, when requested) into this Word document and then submit the document in Canvas

--SUBMIT ALL QUERIES USING THE EXAMPLE FORMATTING BELOW…

Example Question:

Select "bmi" and "age" for 5 records. Order the output from highest to lowest "bmi." Include your output.

Answer:

Query

```
SELECT    bmi,
          age
FROM      health
ORDER BY  bmi DESC
LIMIT     5;
```

Output

```
bmi       age
6388.49   48
5858.59   54
4324.40   27
4320.96   48
3745.48   66
```

--Notes about the formatting:

- --Use Courier New font (because it is fixed-width)
- --Put each SQL clause on a new line
- --Use all caps for all SQL clauses and keywords
- --Write each field in the SELECT statement on a new line
- --Use tabs to clearly separate SQL clauses from field names, table names, etc.

--***IMPORTANT: Do not use the sanford database for anything other than answering the questions on this assignment. This includes copying the data, displaying your work via screenshots, etc. If I, or the IT department, detect any unauthorized access or usage you will automatically receive an F for the course. Please ask if you are not sure if a specific use is authorized.***

--In general, the assignment will be graded for completeness. However, I reserve the right to grade a question or two for correctness.

Hints:

--All field names and the table name are case-sensitive for the sanford database

--In most cases there is no such thing as a single "right" answer. If two different queries generate the same desired output then both are acceptable.

--To copy output directly from the results window in MySQL Workbench, it is typically easiest to right-click and choose the "tab separated" option

Name: KEY

1.) First, let's perform some simple "sanity checks" on the data. (Note: it is always a good idea to perform sanity checks when working with a new dataset.) Count the total number of records in the lone table, "health," and include the output.

Query

```
SELECT    COUNT(*)
FROM      health;
```

Output

```
155143
```

2.) Next, let's check to see if we can determine the number of unique records in "health." Unfortunately COUNT(DISTINCT(*)) does not work (because when COUNT() and DISTINCT() are combined they only accept one column as input—but, we'll find a workaround next week). Is there a specific field that looks like it might be unique? Count the unique values of "id" and display the result. Based on this result, are all records unique?

Query

```
SELECT    COUNT(DISTINCT(id))
FROM      health;
```

Output

```
155143
```

Written Answer

Yes, since the number of unique values of "id" is equal to the total number of rows in the table, all records are necessarily unique.

3.) Write a query to count the total number of patients for each sex. Sort your output on this count from highest to lowest value. Include your output.

Query

```
SELECT    sex,
          COUNT(sex)
FROM      health
GROUP BY  sex
ORDER BY  COUNT(sex) DESC;
```

Output

```
sex       COUNT(sex)
Female    78503
Male      76639
Unknown   1
```

4.) Next, for each sex, calculate the average incidence of hypertension. Exclude the one patient with 'Unknown' sex. (Note: there is no need for any difficult calculation here—take advantage of the fact that hypertension is defined as a "dummy" variable).

Query

```
SELECT    sex,
          AVG(hypertension)
FROM      health
WHERE     sex != 'Unknown'
GROUP BY  sex;
```

Output

```
sex       AVG(hypertension)
Female    0.5623
Male      0.5916
```

5.) Run the following command:

```
SHOW COLUMNS
FROM health;
```

The output provides important information about the columns in our table, including field name, type, whether the field can be NULL, whether the field is a key, any default values, and any extra information. For additional information, see: https://mariadb.com/kb/en/mariadb/show-columns/. For now we will focus on the "Type" column, specifically for the "age" field. The type of "age" is listed as "varchar(45)"—this means that age is stored as a string (i.e. text) with a maximum length of 45 characters. Hmm—that's a little odd. Wouldn't we typically expect age to be an integer? Run a query that returns all of the distinct values of "age." Is there a specific age value that is not an integer?

Query

```
SELECT    DISTINCT(age)
FROM      health;
```

Written Answer

Yes, the value "90+" is not an integer and is likely the reason why the "age" field is stored as a string.

6.) Well, let's see if the "potentially problematic" age that you found in the previous problem will in fact pose an issue. Take that "problematic" value and run the following query (make sure to put the value in single-quotes to identify it as a string):

```
SELECT 10 + '<your problematic value>';
```

Show your output. What?!?!? How did it just do that magic? Well, MariaDB utilized *implicit type conversion*—it implicitly converted our "problematic" string to an integer and added the integers together (while managing to ignore the "extra" mathematical symbol). HOWEVER, WE TYPICALLY WILL WANT TO AVOID USING IMPLICIT TYPE CONVERSION BECAUSE ITS RESULTS WILL NOT ALWAYS BE WHAT WE ANTICIPATE. Field types are very important in SQL/RDBMS and we will explore them in more detail next week.

Query

```
SELECT    10 + '90+';
```

Output

```
100
```

7.) For now, take advantage of this implicit type conversion and calculate the estimated average age for patients grouped by sex, payor, and hypertension. Be sure to ROUND the average ages to two decimal places (and search the internet if you need some help!). Include the count of patients in each group as well. Only include patients that were alive when the data was collected. Further, exclude the patient with 'Unknown' sex. Finally, only include groups with at least 2,000 records and order your output by sex, payor, and then hypertension (all in ascending order). Are we necessarily underestimating or overestimating the average age of each group?

Query

```
SELECT    sex,
          payor,
          hypertension,
          COUNT(*),
          ROUND(AVG(age),2)
```

```
FROM      health
WHERE     status = 'Alive' AND
          sex != 'Unknown'
GROUP BY  sex,
          payor,
          hypertension
HAVING    COUNT(*) >= 2000
ORDER BY  sex,
          payor,
          hypertension;
```

Output

```
Female    Medicare            0    19409     79.18
Female    Medicare            1    25214     73.31
Female    Private Ins/Other   0    11347     52.36
Female    Private Ins/Other   1    17158     55.69
Male      Medicare            0    13599     76.08
Male      Medicare            1    20689     72.16
Male      Private Ins/Other   0    14583     52.02
Male      Private Ins/Other   1    22989     54.31
```

Written Answer

The "90+" age is capturing all patients at least 90 years old. So, some of these patients will be greater than 90, meaning that we are necessarily underestimating the actual age of each group.

8.) Write a simple query to count the total number of records in the table that have a NULL value for the "a1c" field. Include your output.

Query

```
SELECT    COUNT(*)
FROM      health
WHERE     a1c IS NULL;
```

Output

```
120374
```

9.) Confirm your output from the previous question by writing a query that groups patients by a1c value. Order your output from highest group count. Include only 5 rows in your output.

Query

```
SELECT    a1c,
          COUNT(*)
FROM      health
GROUP BY  a1c
ORDER BY  COUNT(*) DESC
LIMIT     5;
```

Output

```
NULL 120374
6.6  1448
6.5  1448
6.4  1427
6.7  1419
```

10.) Run your query from the previous question for the "visits_sched" field, changing it to include 20 rows in your output. How many records have a NULL value?

Query

```
SELECT    visits_sched,
          COUNT(*)
FROM      health
GROUP BY  visits_sched
ORDER BY  COUNT(*) DESC
LIMIT     20;
```

Output

```
visits_sched   COUNT(*)
2              21668
1              21286
3              18712
4              15638
5              12444
6              10390
7              8624
8              7102
9              5963
10             4859
11             3989
12             3474
13             2862
14             2357
15             2070
16             1772
```

```
NULL            1445
17              1445
18              1173
19              1088
```

Written Answer

Both 0 (if they understand this is not actually a NULL value) and 1445 are acceptable.

11.) Try to confirm your output from the previous question by adapting and re-running your query from Question 8. Does your output from this query seem to confirm or contradict your output from the previous query? If there seems to be a contradiction, do a little ad hoc testing and explain what is happening here.

Query

```
SELECT    COUNT(*)
FROM      health
WHERE     visits_sched IS NULL;
```

Output

```
0
```

Written Answer

Running a query such as...

```
SELECT    COUNT(*)
FROM      health
WHERE     visits_sched = 'NULL';
```

...reveals that the 1,445 records of interest are not coded as NULL values--they are set equal to the string, "NULL".

12.) A stakeholder wants to run an analysis involving three fields: bmi, visits_sched, and visits_miss. However, she wants to exclude all records that have a *missing* value for any of these three fields. How many records will be available for her analysis? (Be sure to test for missing values in all three fields in your query.)

Query

```
SELECT    COUNT(*)
FROM      health
WHERE     bmi IS NOT NULL AND
          visits_sched != 'NULL' AND
```

```
          visits_miss != '';
```

Output

```
144903
```

13.) A different stakeholder is attempting to identify patients who are missing a large proportion of their scheduled appointments (and therefore costing the health system a lot of money). Write a query to count the total number of patients who have had at least 4 scheduled appointments and missed at least 50% of those appointments. Only include patients who are currently alive. (Hint: You may take advantage of implicit type conversion here. However, before running the query to generate your count, it might be reassuring to run some ad hoc queries to check if your calculation is operating as you had hoped.)

Query

```
SELECT    COUNT(*)
FROM      health
WHERE     status = 'Alive' AND
          visits_sched >= 4 AND
          visits_miss / visits_sched >= 0.5;
```

Output

```
875
```

14.) A final stakeholder is interested in comparing the obesity of patients in the health system versus the obesity of patients in the US population. She has provided you with a link (https://www.cdc.gov/nchs/data/hus/hus16.pdf#053) to obtain obesity rate estimates for the US. She is specifically interested in data on page 238 of the document. For men and women between 55 and 64 years old, note the obesity rates for the 2011-2014 timeframe (separately for men and women). Write queries to calculate the respective rates in the dataset. (Hints: You should be able to calculate these obesity rates using two queries. Write one query to calculate the total number of patients in the age range, grouped by sex. Then write another query to count the total number of obese patients in the same age range. You can then calculate the obesity rates. Only include patients labeled as "Alive.") Are your obesity rates similar to those in the report?

Queries

```
SELECT    SEX,
          COUNT(*)
FROM      health
WHERE     (age >= 55 AND
          age <= 64) AND
          status = 'Alive'
```

```
GROUP BY  sex;

SELECT    SEX,
          COUNT(*)
FROM      health
WHERE     (age >= 55 AND
          age <= 64) AND
          status = 'Alive' AND
          bmi >= 30
GROUP BY  sex;
```

Output

```
sex       COUNT(*)
Female    17440
Male      20324

sex       COUNT(*)
Female    10119
Male      12230
```

Written Answer

The obesity rates in the report are 44.4% and 38.1% for females and males, respectively. The obesity rates in the dataset are 10119/17440 = 58.0% for females and 12230/20324 = 60.2% for males. The obesity rates in the dataset are higher than those in the report, especially for males.

15.) A coworker wants to examine a1c levels for patients with diabetes. Grouping patients by sex and payor, he can only utilize groups with at least 1,000 records. What groups will he be able to examine, and what is the average a1c level of each? Be sure to only include "Alive" patients and exclude any seemingly erroneous a1c levels. Order output from highest to lowest a1c level.

Query

```
SELECT    sex,
          payor,
          COUNT(*),
          AVG(a1c)
FROM      health
WHERE     a1c < 20 AND
          status = 'Alive'
GROUP BY  sex,
          payor
HAVING    COUNT(*) > 1000
```

```
ORDER BY  AVG(a1c) DESC;
```

**NOTE: The choice of "a1c < 20" was somewhat arbitrary. I looked online to see what "reasonable" a1c levels are and they typically seemed to fall between 4 and 15. So, I added a little "cushion" and simply restricted them to be less than 20. So, depending on your restriction in the WHERE clause, your results might be slightly different.

Output

Sex	payor	COUNT(*)	AVG(a1c)
Male	Private Ins/Other	7279	7.55052
Female	Private Ins/Other	4765	7.39715
Male	Medicare	9734	7.18547
Female	Medicare	10369	7.04703