Homework #2

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.

--You will need to manually type field names related to your output

--*****IMPORTANT: As discussed in class, do not extract, copy, move, share, or take screenshots of any of the data in the database. The only query results that should leave MySQL Workbench are those included in your homework submission. Use copy/paste to directly move your results from MySQL Workbench into this Word document. 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.) Continuing with the sanford database, categorize all patients with 'Alive' status into the five blood pressure categories defined here: http://www.heart.org/HEARTORG/Conditions/HighBloodPressure/KnowYourNumbers/Understanding-Blood-Pressure-Readings_UCM_301764_Article.jsp#.WXiqI8aZNPU. Categorize each patient into the HIGHEST group in which he/she falls. For example, if a patient has systolic blood pressure of 150 mm Hg and diastolic blood pressure of 85 mm Hg, we want to categorize that patient as "Hypertension Stage 1" NOT "Prehypertension." Be sure to exclude any records with a negative value for either blood pressure reading. Also, be sure to include an ELSE NULL at the appropriate spot in your query. Your output should include the blood pressure categories (call this "BP_Group"), the count of patients in each category (call this "Total_Patients"), and the percentage of patients in each category who are actually diagnosed with hypertension (call this "Perc_Hyper").

Query

```
SELECT    CASE
          WHEN sbp > 180 OR dbp > 110 THEN 'Hypertensive
               Crisis'
          WHEN sbp >= 160 OR dbp >= 100 THEN 'Hypertension
               Stage 2'
          WHEN (sbp >= 140 AND sbp <= 159) OR (dbp >= 90
               AND dbp <= 99) THEN 'Hypertension Stage 1'
          WHEN (sbp >= 120 AND sbp <= 139) OR (dbp >= 80
               AND dbp <= 89) THEN 'Prehypertension'
          WHEN sbp < 120 AND dbp < 80 THEN 'Normal'
          ELSE NULL
          END AS BP_Group,
          COUNT(*) AS Total_Patients,
          AVG(hypertension) AS Perc_Hyper
FROM      health
WHERE     status = 'Alive' AND
          sbp >= 0 AND
          dbp >= 0
GROUP BY  BP_Group;
```

Output

```
BP_Group                 Total_Patients      Perc_Hyper
Hypertension Stage 1     24327               0.5050
Hypertension Stage 2     6606                0.4416
Hypertensive Crisis      1400                0.3871
Normal                   33021               0.6021
Prehypertension          86126               0.6248
```

Comment

```
The order of the cases in your statement matters greatly
here because we are checking multiple conditions with OR
conditions.
```

2.) It appears that certain values of the "smoke" field are potentially indicative of larger data issues for those associated records. Generate a categorical field called "Smoke_Group" that categorizes all records into one of three groups: "Some Smoke," "No Smoke," and "Unknown." Using the data info sheet, categorize anyone who currently smokes, formerly smoked, or was exposed to smoke as "Some Smoke" (i.e. values 1, 2, 4, 7, 9, and 10). Categorize only those who have clearly never smoked (i.e. value 5) as "No Smoke." Finally, categorize all others as "Unknown." Include the following in your output:

--Count of records in each group (call this "Total")

--Average incidence of hypertension ("Avg_H"), vascular disease ("Avg_V"), and diabetes ("Avg_D")

--Percentage of non-NULL bmi values ("NonNull")

Does it seem as if one of the groups could potentially require further investigation? If so, what would you do?

Query

```
SELECT    CASE
          WHEN smoke IN (1,2,4,7,9,10) THEN 'Some Smoke'
          WHEN smoke IN (5) THEN 'No Smoke'
          ELSE 'Unknown'
          END AS Smoke_Group,
          COUNT(*) AS Total,
          AVG(hypertension) AS Avg_H,
          AVG(vasc_disease) AS Avg_V,
          AVG(diabetes) AS Avg_D,
          COUNT(bmi)/COUNT(*) AS NonNull
FROM      health
GROUP BY  Smoke_Group;
```

Output

```
Smoke_Group    Total     Avg_H     Avg_V     Avg_D     NonNull
No Smoke       78053     0.5594    0.0502    0.2367    0.9319
```

```
Some Smoke      76415      0.5988     0.1170     0.2733     0.9409
Unknown         675        0.0859     0.0030     0.1422     0.3956
```

Written Answer

Yes, it does appear that further investigation would be warranted. Even though the total number of records in the "Unknown" group is relatively small, all of the corresponding rates are significantly different from those in the other groups. For this group of records, we could try to find any abnormalities across all fields in the table. There could be an underlying data issue preventing these records from being collected/inputted properly.

For the remaining queries you will use the "ontime" table in the "airline_ontime" database. Additional information about the data fields can be found here: http://stat-computing.org/dataexpo/2009/the-data.html. We have only included the data for 2007.

3.) Excluding all flights that were cancelled or diverted, what are the average departure and arrival delays across flights?

Query

```
SELECT    AVG(DepDelay),
          AVG(ArrDelay)
FROM      ontime
WHERE     Cancelled = 0 AND
          Diverted = 0;
```

Output

```
AVG(DepDelay)  AVG(ArrDelay)
11.3621        10.1922
```

4.) Next, try to calculate average departure and arrival delays for the same set of flights using the "DepTime," "CRSDepTime," "ArrTime," and "CRSArrTime" fields. Is your output different than the results from #3? If so, do some investigation and determine why this is occurring. (You do not need to resolve the *potential* issue.)

Query

```
SELECT    AVG(DepTime - CRSDepTime),
          AVG(ArrTime - CRSArrTime)
FROM      ontime
WHERE     Cancelled = 0 AND
          Diverted = 0;
```

Output

```
AVG(DepTime - CRSDepTime)     AVG(ArrTime - CRSArrTime)
9.8423                        -11.9238
```

Written Answer

```
Some further investigation will reveal that the issue has
two main causes...
--The difference between these two integer fields does not
necessarily yield a number in minutes (i.e. 2330 - 2100 =
230, but there are really only 150 minutes between these
two times)
--Even if we corrected the first issue by converting the
integers to times, we could never pick out the cases where
the actual date of departure/arrival is different from the
scheduled date of departure/arrival (because we only have
one date for each record)
```

5.) Continue focusing on flights that are not cancelled or diverted. Assume that all aircraft registered through the US FAA have a tail number starting with "N". How many flights were completed by aircraft registered elsewhere?

Query

```
SELECT    COUNT(*)
FROM      ontime
WHERE     Cancelled = 0 AND
          Diverted = 0 AND
          TailNum NOT LIKE 'N%';
```

Output

```
COUNT(*)
250201
```

6.) What is the total distance traveled of all non-cancelled, non-diverted flights that departed from RDU on July 4, 2007?

Query

```
SELECT    SUM(Distance)
FROM      ontime
WHERE     Cancelled = 0 AND
          Diverted = 0 AND
          Origin = 'RDU' AND
          Year = 2007 AND
          Month = 7 AND
          DayofMonth = 4;
```

Output

```
SUM(Distance)
96024
```

7.) From May 15, 2007 to August 15, 2007 (inclusive), how many flights departing from Atlanta (ATL) were cancelled or delayed more than 30 minutes? Run a separate query and calculate the percentage of ATL flight records that are cancelled or delayed more than 30 minutes over that span.

Query

```
SELECT    COUNT(*)
FROM      ontime
WHERE     (Cancelled = 1 OR
          DepDelay > 30) AND
          Origin = ‘ATL’ AND
          DATE(CONCAT(Year,'-',Month,'-',DayofMonth)) >=
          DATE(‘2007-05-15’) AND
          DATE(CONCAT(Year,'-',Month,'-',DayofMonth)) <=
          DATE(‘2007-08-15’);

SELECT    COUNT(*)
FROM      ontime
WHERE     Origin = ‘ATL’ AND
          DATE(CONCAT(Year,'-',Month,'-',DayofMonth)) >=
          DATE(‘2007-05-15’) AND
          DATE(CONCAT(Year,'-',Month,'-',DayofMonth)) <=
          DATE(‘2007-08-15’);
```

Output

```
COUNT(*)
23891

COUNT(*)
109980
```

Written Answer

```
23891/109980 = 21.7% of flights were cancelled or delayed
```

```
30 minutes during this span
```

8.) (This query is not trivial and will require some searching on the internet.) What are the five calendar weeks of 2007 with the greatest number of flight records. Assume that a week starts on Sunday, the "week number" can take on values from 0 to 53, and Week 1 is the 1st week with a Sunday in 2007. Your output should contain the aforementioned "week number," the start date corresponding to that "week number," and the total number of records for that week. To give you a hint, the first line in your output should be:

```
29   2007-07-22     148276
```

You can answer this question with a single query and you CANNOT "hard code" the date when each week number starts—you must use date functions.

Query

```
SELECT    WEEK(DATE(CONCAT(Year,'-',Month,'-',DayofMonth)))
          AS Week_Number,
          STR_TO_DATE(CONCAT('2007',WEEK(DATE(CONCAT(Year,'
          -',Month,'-',DayofMonth))),' Sunday'), '%Y%U %W')
          AS Week_Start_Date,
          COUNT(*) AS Total_Records
FROM      ontime
GROUP BY  Week_Number,
          Week_Start_Date
ORDER BY  Total_Records DESC
LIMIT     5;
```

Output

```
Week_Number    Week_Start_Date     Total_Records
29             2007-07-22          148276
28             2007-07-15          148262
27             2007-07-08          148187
24             2007-06-17          148165
31             2007-08-05          148158
```

Comments

```
In the first part of the question I basically specified
that you should use the default "mode" setting for the
WEEK() function.  While my method of determining
Week_Start_Date is certainly not the only approach, let me
explain why something like the following is NOT correct...

SELECT    WEEK(DATE(CONCAT(Year, '-', Month, '-
          ',DayofMonth)),0) AS Week_Number,
          DATE(CONCAT(Year, '-', Month, '-', DayofMonth))AS
          Week_Start_Date,
          COUNT(*) as   Total_Records
FROM      ontime
GROUP BY  Week_Number
ORDER BY  Total_Records DESC
LIMIT     5;
```

I casually glanced through the submissions and saw a lot of the above query (or some variant of it). Why is this query problematic? BECAUSE WE HAVE FALLEN FOR THE DREADED GROUP BY TRAP—THE LIST OF FIELDS IN OUR GROUP BY CLAUSE DOES NOT MATCH THE LIST OF FIELDS IN OUR SELECT CLAUSE. So, while we might have gotten the correct output in this case, the value returned for Week_Start_Date (associated with each Week_Number) could hypothetically be any date within the corresponding Week_Number. It just so happened in this case that we got lucky and the data were stored in a way such that the optimizer pulled the first day of each associated Week_Number. This is important and something we need to always be aware of when using GROUP BY in MariaDB or MySQL.