Homework #5

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/Jupyter are those included in your homework submission. Use copy/paste to directly move your results from MySQL Workbench/Jupyter 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 lahman2016 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:

1.) (This is a toughie.) Utilize all data from the 2000 to 2016 seasons in the lahman2016 database to answer the following problem. We're going to examine how a baseball player's career batting average changes as the player ages. Batting average is a simple calculation: H/AB. Exclude all data for any player in a season when the player plays more of his games (G) at the pitcher position (POS = 'P') than at all other positions combined in that season (you'll need the 'Fielding' table to figure out this part of the query). You can calculate a player's age in a given year by simply using (birthYear – yearID). Create a "cumulative batting average) for every age that occurs in the data during this time-span (from 2000 to 2016, inclusive). Your output should have 4 columns: age, cumulative hits (including the current age and all ages younger than it), cumulative at-bats (including the current age and all ages younger than it), and cumulative average (including the current age and all ages younger than it). Note that you need to be careful how you're calculating batting average. For example, suppose there are two ages—30 and 31. If total at-bats for all players aged 30 is 200 and total hits for all players aged 30 is 100, then batting average for that year is 0.500. If total at-bats for all players aged 31 is 100 and total hits for all players aged 31 is 10, then batting average for that year ALONE is 0.100. However, cumulative batting average IS NOT simply (0.5 + 0.1) / 2 = 0.3. It is (110/300) = 0.367.

HINT: Your output should include 31 rows from age 19 to age 49.

Query

```
WITH POS_Check1 AS
(
SELECT          *,
                CASE
                WHEN POS = 'P' THEN 1
                ELSE  0
                END AS PCheck
FROM            Fielding
WHERE           yearID >= 2000 AND
                yearID <= 2016
),
POS_Check2 AS
(
SELECT          playerID,
                yearID,
```

```
                PCheck,
                SUM(G) AS tot_g
FROM            POS_Check1
GROUP BY        playerID,
                yearID,
                PCheck
),
POS_Check3 AS
(
SELECT          playerID,
                yearID,
                MAX(CASE WHEN PCheck = 0 THEN tot_g ELSE 0
                END) AS NonP_G,
                MAX(CASE WHEN PCheck = 1 THEN tot_g ELSE 0
                END) AS P_G
FROM            POS_Check2
GROUP BY        playerID,
                yearID
),
POS_Check4 AS
(
SELECT          *
FROM            POS_Check3
WHERE           NonP_G >= P_G
),
ADD_BD AS
(
SELECT          POS_Check4.*,
                Master.birthYear
FROM            POS_Check4
INNER JOIN      Master
ON              POS_Check4.playerID = Master.playerID
),
AGE_AB_H AS
(
SELECT          (ADD_BD.yearID - birthYear) AS Age,
                AB,
                H
FROM            ADD_BD
INNER JOIN      Batting
ON              ADD_BD.playerID=Batting.playerID AND
                ADD_BD.yearID=Batting.yearID
),
AGG_AGE AS
(
SELECT          Age,
                SUM(AB) AS Tot_AB,
```

```
                SUM(H) as Tot_H
FROM            AGE_AB_H
GROUP BY        Age
)

SELECT          Age,
                (SUM(Tot_H) OVER(ORDER BY Age ROWS BETWEEN
                UNBOUNDED PRECEDING AND CURRENT ROW)
                /
                SUM(Tot_AB) OVER(ORDER BY Age ROWS BETWEEN
                UNBOUNDED PRECEDING AND CURRENT ROW)
                ) AS Cumulative_BA
FROM            AGG_AGE;
```

Output

Age	Cumulative_BA
19	0.1765
20	0.2656
21	0.272
22	0.2674
23	0.2655
24	0.2639
25	0.2634
26	0.2637
27	0.2641
28	0.2643
29	0.2647
30	0.2649
31	0.265
32	0.265
33	0.265
34	0.265
35	0.265
36	0.265
37	0.265
38	0.2649
39	0.2649
40	0.2649
41	0.2649
42	0.2649
43	0.2649
44	0.2649
45	0.2649
46	0.2649
47	0.2649
48	0.2649

```
49   0.2649
```

Comment

The use of the two MAX() functions is an example of "pivoting" the data. You did not need to do this in order to get the same answer (you could have used JOINs). Feel free to research pivoting if you're interested.

2.) In 2016, of the players with at least 500 AB, how many players hit more HR than the average number of HR hit by this group of players? (Don't forget to aggregate all AB and HR for each playerID before checking if he has at least 500 AB.)

Query

```
WITH sub1 AS
(
SELECT          playerID,
                SUM(AB) AS abtot,
                SUM(HR) as hrtot
FROM            Batting
WHERE           yearID=2016
GROUP BY        playerID
),
sub2 AS
(
SELECT          playerID,
                abtot,
                hrtot,
                AVG(hrtot) OVER() AS hrav
FROM            sub1
WHERE           abtot >= 500
)
SELECT          COUNT(*)
FROM            sub2
WHERE           hrtot > hrav;
```

Output

```
COUNT(*)
61
```

3.) The season HR record (i.e. the number of HR that a player hits in a single) is an important record in baseball. Starting in 1900, write a query to determine the number of times that the HR record has been broken. The record is "broken" in a given season if a player in that season hits more home runs than any player had it in a single season in all previous seasons. Again, don't forget to aggregate all HR for a player in a given season.

Your final query output should be a single number.

Query

```
WITH sub1 AS
(
SELECT          yearID,
                playerID,
                SUM(HR) AS hrtot
FROM            Batting
WHERE           yearID >= 1900
GROUP BY        yearID,
                playerID
),
sub2 AS
(
SELECT          yearID,
                MAX(hrtot) AS hrtot1
FROM            sub1
GROUP BY        yearID
),
sub3 AS
(
SELECT          yearID,
                hrtot1,
                MAX(hrtot1) OVER(ORDER BY yearID
                ROWS BETWEEN UNBOUNDED PRECEDING AND
                1 PRECEDING) AS hrrec
FROM            sub2
)

SELECT          COUNT(*)
FROM            sub3
WHERE           hrtot1 > hrrec;
```

Output

```
COUNT(*)
10
```

4.) In 2016, how many players hit more HR than their career average number of HR in a season? Again, don't forget to aggregate all HR for a player in a given season.

Query

```
WITH sub1 AS
(
```

```
SELECT          playerID,
                SUM(HR) AS HR_2016
FROM            Batting
WHERE           yearID = 2016
GROUP BY        playerID
),
sub2 AS
(
SELECT          playerID,
                yearID,
                SUM(HR) AS Season_HR
FROM            Batting
GROUP BY        playerID,
                yearID
),
sub3 AS
(
SELECT          sub2.playerID,
                sub2.yearID,
                AVG(Season_HR) OVER(PARTITION BY playerID
                ORDER BY yearID ROWS BETWEEN UNBOUNDED
                PRECEDING AND 1 PRECEDING) AS Career_AV,
                HR_2016
FROM            sub2
INNER JOIN      sub1
ON              sub2.playerID = sub1.playerID
)

SELECT          COUNT(*)
FROM            sub3
WHERE           yearID = 2016 AND
                HR_2016 > Career_AV;
```

Output

```
COUNT(*)
302
```

5.) In 2016, only consider players with at least 500 AB WITH A SINGLE TEAM (i.e. if a player played for multiple teams in 2016 he should be excluded). How many players hit more HR than their team average number of HR AND their individual league (AL vs. NL) average number of HR AND the overall major league (AL and NL combined) average number of HR?

Query

```
WITH sub1 AS
(
SELECT          playerID,
                teamID,
                COUNT(*)
FROM            Batting
WHERE           yearID = 2016
GROUP BY        playerID,
                teamID
HAVING          COUNT(*) = 1
),
sub2 AS
(
SELECT          Batting.playerID,
                Batting.teamID,
                lgID,
                HR,
                AVG(HR) OVER(PARTITION BY Batting.teamID) AS
                Team_Av,
                AVG(HR) OVER(PARTITION BY lgID) AS
                League_Av,
                AVG(HR) OVER() AS MLB_Av
FROM            Batting
INNER JOIN      sub1
ON              Batting.playerID = sub1.playerID
WHERE           yearID = 2016 AND
                AB >= 500
)
SELECT          COUNT(*)
FROM            sub2
WHERE           HR > Team_Av AND
                HR > League_Av AND
                HR > MLB_Av;
```

Output

```
COUNT(*)
41
```

6.) Finally, utilize data from 2013 to 2016. Only consider players that have at least 200 AB in each of those 4 seasons (but don't forget to aggregate all AB for a player in a given season). Of these players, we would like know how many had a "bad" season in 2016. In particular, how many players had fewer hits (H), runs (R), runs batted in (RBI), and more strikeouts (SO) than they did on average during the previous three seasons?

Query

```
WITH sub1 AS
(
SELECT          playerID,
                COUNT(DISTINCT (yearID))
FROM            Batting
WHERE           yearID >= 2013
GROUP BY        playerID
HAVING          COUNT(DISTINCT (yearID)) = 4
),
sub2 AS
(
SELECT          Batting.playerID,
                Batting.yearID,
                SUM(AB) AS AB_Tot
FROM            Batting
INNER JOIN      sub1
ON              Batting.playerID = sub1.playerID
WHERE           yearID >= 2013
GROUP BY        Batting.playerID,
                Batting.yearID
),
sub3 AS
(
SELECT          playerID,
                MIN(AB_Tot)
FROM            sub2
GROUP BY        playerID
HAVING          MIN(AB_Tot) >= 200
),
sub4 AS
(
SELECT          Batting.playerID,
                yearID,
                SUM(H) AS H_tot,
                SUM(R) AS R_tot,
                SUM(RBI) AS RBI_tot,
                SUM(SO) AS SO_tot
FROM            Batting
INNER JOIN      sub3
ON              Batting.playerID = sub3.playerID
WHERE           yearID >= 2013
GROUP BY        Batting.playerID,
                yearID
),
sub5 AS
(
SELECT          sub4.*,
```

```
                AVG(H_tot) OVER(PARTITION BY playerID ORDER
                BY yearID ROWS BETWEEN 3 PRECEDING AND 1
                PRECEDING) AS Av_H,
                AVG(R_tot) OVER(PARTITION BY playerID ORDER
                BY yearID ROWS BETWEEN 3 PRECEDING AND 1
                PRECEDING) AS Av_R,
                AVG(RBI_tot) OVER(PARTITION BY playerID
                ORDER BY yearID ROWS BETWEEN 3 PRECEDING AND
                1 PRECEDING) AS Av_RBI,
                AVG(SO_tot) OVER(PARTITION BY playerID ORDER
                BY yearID ROWS BETWEEN 3 PRECEDING AND 1
                PRECEDING) AS Av_SO
FROM            sub4
)
SELECT          COUNT(*)
FROM            sub5
WHERE           yearID = 2016 AND
                H_tot < Av_H AND
                R_tot < Av_R AND
                RBI_tot < Av_RBI AND
                SO_tot > Av_SO;
```

Output

```
COUNT(*)
14
```