

Grouping Data with SQL - Lab

Introduction

In this lab, you'll query data from a table populated with Babe Ruth's career hitting statistics. Then you'll use aggregate functions to pull interesting information from the table that basic queries cannot track.

Objectives

- Describe the relationship between aggregate functions and `GROUP BY` statements
- Use `Group BY` statements in SQL to apply aggregate functions like: `COUNT` , `MAX` , `MIN` , and `SUM`
- Create an alias in a SQL query
- Use the `HAVING` clause to compare different aggregates
- Compare the difference between the `WHERE` and `HAVING` clause

Babe Ruth - Career Hitting Statistics

The database you will be working with in this lab is located in the file `babe_ruth.db` . This database contains a single table, `babe_ruth_stats` . The table schema is:

```
CREATE TABLE babe_ruth_stats (  
  id INTEGER PRIMARY KEY,  
  year INTEGER,  
  team TEXT,  
  league TEXT,  
  doubles INTEGER,  
  triples INTEGER,  
  hits INTEGER,  
  HR INTEGER,  
  games INTEGER,  
  runs INTEGER,  
  RBI INTEGER,  
  at_bats INTEGER,  
  BB INTEGER,  
  SB INTEGER,  
  SO INTEGER,  
  AVG REAL  
)
```

The table contains the following data:

year	team	league	doubles	triples	hits	HR	games	runs	RBI	at_bats	BB	SB	SO	AVG
1914	"BOS"	"AL"	1	0	2	0	5	1	2	10	0	0	4	0.2
1915	"BOS"	"AL"	10	1	29	4	42	16	21	92	9	0	23	0.315
1916	"BOS"	"AL"	5	3	37	3	67	18	15	136	10	0	23	0.272
1917	"BOS"	"AL"	6	3	40	2	52	14	12	123	12	0	18	0.325
1918	"BOS"	"AL"	26	11	95	11	95	50	66	317	58	6	58	0.3
1919	"BOS"	"AL"	34	12	139	29	130	103	114	432	101	7	58	0.322
1920	"NY"	"AL"	36	9	172	54	142	158	137	458	150	14	80	0.376
1921	"NY"	"AL"	44	16	204	59	152	177	171	540	145	17	81	0.378
1922	"NY"	"AL"	24	8	128	35	110	94	99	406	84	2	80	0.315
1923	"NY"	"AL"	45	13	205	41	152	151	131	522	170	17	93	0.393
1924	"NY"	"AL"	39	7	200	46	153	143	121	529	142	9	81	0.378
1925	"NY"	"AL"	12	2	104	25	98	61	66	359	59	2	68	0.29
1926	"NY"	"AL"	30	5	184	47	152	139	146	495	144	11	76	0.372
1927	"NY"	"AL"	29	8	192	60	151	158	164	540	137	7	89	0.356
1928	"NY"	"AL"	29	8	173	54	154	163	142	536	137	4	87	0.323
1929	"NY"	"AL"	26	6	172	46	135	121	154	499	72	5	60	0.345
1930	"NY"	"AL"	28	9	186	49	145	150	153	518	136	10	61	0.359
1931	"NY"	"AL"	31	3	199	46	145	149	163	534	128	5	51	0.373
1932	"NY"	"AL"	13	5	156	41	133	120	137	457	130	2	62	0.341
1933	"NY"	"AL"	21	3	138	34	137	97	103	459	114	4	90	0.301
1934	"NY"	"AL"	17	4	105	22	125	78	84	365	104	1	63	0.288
1935	"BOS"	"NL"	0	0	13	6	28	13	12	72	20	0	24	0.181

As you can see, each record in this table represents statistics for a baseball season.

Connect to the Database

Import `sqlite3` and `pandas`. Then, connect to the database in the `babe_ruth.db` file.

```
In [1]: # import relevant libraries
import sqlite3
import pandas as pd
# connecting to the database
conn = sqlite3.connect('babe_ruth.db')
```

Now, write SQL queries to answer questions about the data in the `babe_ruth_stats` table. You can display all results using `pandas` for readability.

Total Seasons

Return the total number of years that Babe Ruth played professional baseball

```
In [2]: ▶ # calculating total number of years Babe Ruth played professional baseball
q = """SELECT COUNT(*) AS num_years
FROM babe_ruth_stats;
"""

pd.read_sql(q,conn)
```

Out[2]:

	num_years
0	22

Seasons with NY

Return the total number of years Babe Ruth played with the NY Yankees (i.e. where the team value is "NY").

```
In [3]: ▶ # calculating total number of years Babe Ruth played with NY Yankees
q = """SELECT COUNT(*) AS num_years
FROM babe_ruth_stats
WHERE team = 'NY';
"""


pd.read_sql(q,conn)
```

Out[3]:

	num_years
0	15

Most Home Runs

Return the row with the most HR that Babe Ruth hit in one season.

In [4]:  *# Calculating the rows with most HR that Babe Ruth hit in one season*


```
q= """
SELECT *
FROM babe_ruth_stats
WHERE HR=(SELECT MAX(HR)
FROM babe_ruth_stats);
"""
pd.read_sql(q,conn)
```

Out[4]:

	id	year	team	league	doubles	triples	hits	HR	games	runs	RBI	at_bats	BB	SB
0	14	1927	NY	AL	29	8	192	60	151	158	164	540	137	7

Least HR

Select the row with the least number of HR hit in one season.

In [5]:  *# Calculating the rows with Least HR that Babe Ruth hit in one season*


```
q= """
SELECT *
FROM babe_ruth_stats
WHERE HR=(SELECT MIN(HR)
FROM babe_ruth_stats);
"""
pd.read_sql(q,conn)
```

Out[5]:

	id	year	team	league	doubles	triples	hits	HR	games	runs	RBI	at_bats	BB	SB
0	1	1914	BOS	AL	1	0	2	0	5	1	2	10	0	0

Total HR

Return the total number of HR hit by Babe Ruth during his career.

In [6]:  *# Calculating the total number of HR hit by Babe Ruth during his career*

```
q= """
SELECT SUM(HR) AS total_HR_hits
FROM babe_ruth_stats;
"""
pd.read_sql(q,conn)
```

Out[6]:

	total_HR_hits
0	714

Five Worst HR Seasons With at Least 100 Games Played

Above you saw that Babe Ruth hit 0 home runs in his first year when he played only five games. To avoid this and other extreme outliers, first filter the data to include only those years in which Ruth played in at least 100 games. Then, select all of the columns for the 5 worst seasons, in terms of the number of home runs, where he played over 100 games.

```
In [7]: ▶ # calculating 5 worst HR seasons with at least 100 games played
q="""
SELECT *
FROM babe_ruth_stats
WHERE games >=100
order by HR
LIMIT 5;
"""
pd.read_sql(q,conn)
```

Out[7]:

	id	year	team	league	doubles	triples	hits	HR	games	runs	RBI	at_bats	BB	SB
0	21	1934	NY	AL	17	4	105	22	125	78	84	365	104	1
1	6	1919	BOS	AL	34	12	139	29	130	103	114	432	101	7
2	20	1933	NY	AL	21	3	138	34	137	97	103	459	114	4
3	9	1922	NY	AL	24	8	128	35	110	94	99	406	84	2
4	10	1923	NY	AL	45	13	205	41	152	151	131	522	170	17

Average Batting Average

Select the average, AVG, of Ruth's batting averages. The header of the result would be AVG(AVG) which is quite confusing, so provide an alias of career_average.

```
In [8]: ▶ # Calculating the Average Battling Average
q = """
SELECT AVG(AVG) AS career_average
FROM babe_ruth_stats;
"""
pd.read_sql(q, conn)
```

Out[8]:

	career_average
0	0.322864

Number of Years with Over 300 Times On Base

We want to know the years in which Ruth successfully reached base over 300 times. We need to add `hits` and `BB` to calculate how many times Ruth reached base. Simply add the two columns together (ie: `SELECT [columnName] + [columnName] AS ...`) and give this value an alias of `on_base`. Select the `year` and `on_base` for only those years with an `on_base` over 300.

```
In [9]: ▶ # calculating the Number of Years with over 300 times on Base
q = """
SELECT year, hits + BB AS on_base
FROM babe_ruth_stats
WHERE on_base > 300
ORDER BY on_base DESC;
"""
pd.read_sql(q, conn)
```

Out[9]:

	year	on_base
0	1923	375
1	1921	349
2	1924	342
3	1927	329
4	1926	328
5	1931	327
6	1920	322
7	1930	322
8	1928	310

Total Years and Hits Per Team

Select the total number of years played (as `num_seasons`) and total hits (as `total_hits`) Babe Ruth had for each team he played for. The result should have 2 rows, one for each team.

```
In [10]: # calculate the total number of years played  
q = """  
SELECT team, COUNT(*) AS num_seasons, SUM(hits) AS total_hits  
FROM babe_ruth_stats  
GROUP BY team;  
"""  
pd.read_sql(q, conn)
```

Out[10]:

	team	num_seasons	total_hits
0	BOS	7	355
1	NY	15	2518

Teams with More than 10 Seasons

Repeat the above query, this time only including teams where he played for more than 10 years.

Hint: Think about whether this filtering occurs before or after the `GROUP BY`. If before, that's a `WHERE`. If after, that's a `HAVING`.

```
In [11]: # Calculate teams played with more than 10 seasons  
q = """  
SELECT team, COUNT(*) AS num_seasons, SUM(hits) AS total_hits  
FROM babe_ruth_stats  
GROUP BY team  
HAVING num_seasons > 10;  
"""  
pd.read_sql(q, conn)
```

Out[11]:

	team	num_seasons	total_hits
0	NY	15	2518

Team with Highest Average At Bats

Select the name of the team and the average at bats per season (as `average_at_bats`), for the team where he averaged the highest at bats.

```
In [12]: # Calculate team with highest Average at Bats  
q="""  
SELECT team,AVG(at_bats) AS average_at_bats  
FROM babe_ruth_stats  
GROUP BY team  
ORDER BY average_at_bats DESC  
LIMIT 1;  
"""  
pd.read_sql(q, conn)
```

Out[12]:

	team	average_at_bats
0	NY	481.133333

Teams with Average At Bats Over 100

Repeat the above query, this time returning all teams where the `average_at_bats` was over 100.

```
In [13]: # Calculate team with Average at Bats over 100  
q="""  
SELECT team,AVG (at_bats) AS average_at_bats  
FROM babe_ruth_stats  
GROUP BY team  
HAVING average_at_bats > 100;  
"""  
pd.read_sql(q, conn)
```

Out[13]:

	team	average_at_bats
0	BOS	168.857143
1	NY	481.133333

Summary

Well done! In this lab, you continued to add complexity to SQL statements, which included using some aggregate functions, the `GROUP BY` statement, and the `HAVING` statement. You wrote queries that showed Babe Ruth's total years and home runs per team as well as selected only years that met a minimum value of our calculated on base attribute.