Aggregate Functions

- SQL Queries don't just access raw data, they can also perform calculations on the raw data to answer specific data questions.
- Calculations performed on multiple rows of a table are called **aggregates**.
- Preview: Docs Returns the number of rows that match the specified criteria.
- `COUNT()` : count the number of rows
- `SUM()`: the sum of the values in a column
- `MAX()` `MIN()` : the largest/smallest value
- `AVG()` : the average of the values in a column
- `ROUND()` : round the values in the column

Count()

Count() is a function that takes the name of a column as an argument and counts the number of non-empty values in that column.
SELECT COUNT(*) FROM fake_apps;

Sum()

SQL makes it easy to add all values in a particular column using `SUM()`. It is a function that takes the name of a column as an argument and returns the sum of all the values in that column.
SELECT SUM(downloads) FROM fake_apps;

Max() / Min()

The `MAX()` and `MIN()` functions return the highest and lowest values in a column, respectively.
SELECT MAX(downloads)
FROM fake_apps;

Average()
SQL uses the `AVG()` function to quickly calculate the average value of a particular column.
SELECT AVG(downloads) FROM fake_apps;

Round()

- By default, SQL tries to be as precise as possible without rounding. We can make the result table easier to read using the Round()
- Round() function takes two arguments inside the parenthesis:
  a column name an integer

  SELECT ROUND(price, 2) FROM fake_apps;

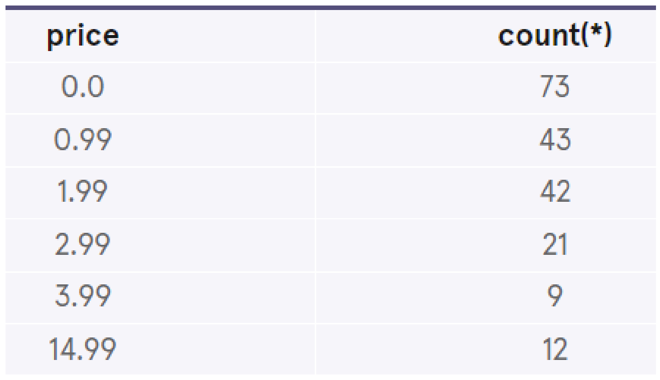
Group By/Order by

- to calculate an aggregate for data with certain characteristics.
- to arrange identical data into groups.

```
SELECT AVG(imdb_rating)
FROM movies
WHERE year = 1999;

SELECT AVG(imdb_rating)
FROM movies
WHERE year = 2000;

SELECT AVG(imdb_rating)
FROM movies
WHERE year = 2001;

SELECT year,
  AVG(imdb_rating)
FROM movies
GROUP BY year
ORDER BY year;
```

- The `GROUP BY` statement comes after any `WHERE`_ statements, but before Order by or `LIMIT`

SELECT price, COUNT(*) FROM fake_apps GROUP BY price;



price	count(*)
0.0	73
0.99	43
1.99	42
2.99	21
3.99	9
14.99	12

- Sometimes, we want to GROUP BY a calculation done on a column.

```
SELECT ROUND(imdb_rating),
   COUNT(name)
FROM movies
GROUP BY ROUND(imdb_rating)
ORDER BY ROUND(imdb_rating);
—-------------------------------------------------------
SELECT ROUND(imdb_rating),
   COUNT(name)
FROM movies
GROUP BY 1     //group by first column imdb_rating
ORDER BY 1;
```

Having

- In addition to being able to group data using GROUP BY SQL also allows you to filter which groups to include and which to exclude.
- `HAVING` is very similar to `WHERE`.
- **Where clause, want to filter the rows;**
- **Having clause, want to *filter groups*.**

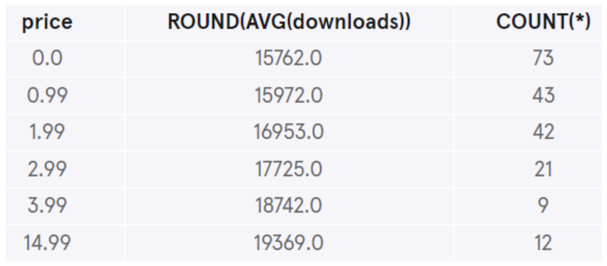
```
SELECT year,
   genre,
   COUNT(name)
FROM movies
GROUP BY 1, 2
HAVING COUNT(name) > 10;
```

1. When we want to limit the results of a query based on values of the individual rows, use `WHERE`.
2. When we want to limit the results of a query based on an aggregate property, use `HAVING`.

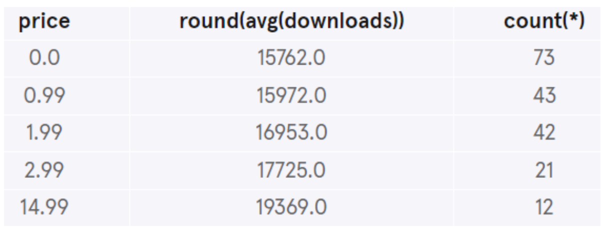
`HAVING` statement always comes after `GROUP BY`, but before Order By and Limit

```
SELECT price,
   ROUND(AVG(downloads)),
   COUNT(*)
FROM fake_apps
GROUP BY price;
```

```
select price,
round(avg(downloads)),
count(*)
from fake_apps
group by price
having count(*)>10
```



price	ROUND(AVG(downloads))	COUNT(*)
0.0	15762.0	73
0.99	15972.0	43
1.99	16953.0	42
2.99	17725.0	21
3.99	18742.0	9
14.99	19369.0	12



price	round(avg(downloads))	count(*)
0.0	15762.0	73
0.99	15972.0	43
1.99	16953.0	42
2.99	17725.0	21
14.99	19369.0	12

GROUP BY - is a clause used with aggregate functions to combine data from one or more columns.

HAVING - limit the results of a query based on an aggregate property.

Link

https://www.codecademy.com/learn/learn-sql/modules/learn-sql-aggregate-functions/cheatsheet