**ORDER BY and GROUP BY**

1. **Highest Rated Movies**

The numbers in the **ravg** column are averages over multiple ratings systems. It's a rough measure of how highly critics and audience members rated each movie.

Write a query to select the **5 highest rated movies** according to this ravg metric.

* select \* from movies

Order by ravg DESC

Limit 5;

* 2 movies had English titles

1. **Lowest Rated Movies**

Now write a query to select the 5 lowest rated movies.

Notice the different ratio of English titles here.

🡪 Select \* from movies

Order by ravg

Limit 5;

* Here all of 5 movies where having English titles.

1. **Highest Grossing Movies**

Write a query to select the 5 movies with the highest revenue.

🡪 select \* from movies

order by revenue desc

limit 5;

1. **Lowest Grossing Movies**

Now let's look at the 5 movies with the lowest revenue.

🡪 select \* from movies

order by revenue

limit 5;

1. **Movies that lost money**

Write a query to select all movies that lost money.

In other words, the movies WHERE revenue < budget.

* select \* from movies

where revenue < budget;

1. **Sort by Ratings**

Now order the movies that lost money by their ratings average.

Use the **DESC** flag so the highest rated are first.

* select \* from movies

where revenue < budget

order by ravg desc;

------------------------------------------------------------------------------------------------------

In **SQL**, the **GROUP BY** clause will group together rows in a table based on the values in a given column. This enables us to perform calculations on each group, such as finding the **sum**, **average** and **count** of columns using the **SUM()**, **AVG()**, and **COUNT()** functions.

1. **Group by Substrate**

Notice the **substrate** column. This indicates if the vehicle travels through air, water or on land.

Let's **group** the vehicles based on the **substrate** column by adding the following to the end of our query:

**🡪** select \* from vehicles

group by substrate;

1. **Average Speed**

The power of **GROUP BY** is that it enables us to perform calculations on groups of rows.

Let's calculate the **average speed** for each propellant group by replacing **\*** in our **SELECT** command with the following **AVG** function:

**🡪** SELECT avg(speed) FROM vehicles

group by propellant;

1. **More Columns**

We are now seeing average speeds, but it's hard to tell which average goes with which propellant.

The **SELECT** command allows us to list as many columns as we want, separated by **commas**.

Let's include the **propellant** column before **AVG(speed)** like this:

🡪 select propellant, avg(speed) from vehicles

group by propellant;

1. **Passengers per Substrate**

Awesome! Let's try another grouping calculation.

Write a query that will return the **average number of passengers** for **each substrate**. Include the substrate column.

🡪 select substrate, avg(passengers) from vehicles

group by substrate ;

1. **Counting Vehicles**

Now let's do one more, but this time we'll use the **COUNT()** function instead of AVG().

Write a query to retrieve the **substrate** column and count **the number of vehicles** for **each substrate**.

🡪 select substrate, COUNT(vehicle) from vehicles

group by substrate ;

In the previous section we learned how to use the **GROUP BY** clause to perform grouping calculations.

Here we will extend this understanding by exploring how to **rename** and **order** calculated columns.

1. **Continent Populations**

The **countries** table includes the population of each country. But what if we wanted to know the population of each continent? **GROUP BY** to the rescue!!

Use **GROUP BY** and **SUM()** to calculate the **total population** of **each continent**. Select only the continent column and the sum.

🡪SELECT continent, SUM(population) FROM countries

GROUP BY continent;

1. **Counting Countries**

Use **COUNT()** to write a query to show the **number of countries** for each continent. Select the continent and the count.

🡪SELECT continent, COUNT(country) FROM countries

GROUP BY continent;

1. **Counting Prizes**

Does the Nobel Prize Committee award the same number of prizes each year? Let's find out!

Write a query to **count** the **number of prizes** for **each year**. Select the year and the count.

🡪SELECT year, COUNT(\*) FROM nobel\_prizes

GROUP BY year;

1. **Rename Column**

**SQL** allows us to rename columns by using the **AS** operator.

Try replacing **COUNT(\*)** with the following to rename that calculated column as **prize\_count**:

**🡪**SELECT year, COUNT(\*) AS prize\_count

FROM nobel\_prizes

GROUP BY year;

1. **Order by prize\_count**

What if we wanted to find the year where they gave out the highest number of prizes?

Now that we have a nice **prize\_count** name for that calculated column, we can use the ORDER BY clause to sort the results.

🡪SELECT year, COUNT(\*) AS prize\_count

FROM nobel\_prizes

GROUP BY year

ORDER BY prize\_count DESC;

1. **Counting Categories**

Fifteen prizes is a lot, but those numbers represent **people**. Many prizes are given to multiple collaborators. Let's count the number of **categories** awarded each year.

While we are at it, let's rename that column to **category\_count**.

🡪SELECT year, COUNT(category) AS category\_count

FROM nobel\_prizes

GROUP BY year

ORDER BY category\_count DESC;

**7) Counting DISTINCT**

**Wait a minute!!** Those are the same numbers we got last time!

This is because by default **COUNT** will tally up all values even if there are duplicates. But SQL provides us with the **DISTINCT** flag which will tell COUNT to ignore duplicates.

🡪SELECT year, COUNT(DISTINCT category) AS category\_count

FROM nobel\_prizes

GROUP BY year

ORDER BY category\_count DESC;

**8) Movie Revenue**

Let's switch over to movies and apply everything we've learned so far.

Write a query to show which MPAA category generates the most average revenue for movies.

Call the calculated column **avg\_revenue**, and order by that value from greatest to least.

🡪SELECT mpaa, AVg(revenue) AS avg\_revenue

FROM movies

GROUP BY mpaa

ORDER BY avg\_revenue DESC;