Cheatsheets / **Learn SQL**

Aggregate Functions

Aggregate Functions

Aggregate functions perform a calculation on a set of values and return a single value:

- `COUNT()`
- `SUM()`
- `MAX()`
- `MIN()`
- `AVG()`

`COUNT()` Aggregate Function

The `COUNT()` aggregate function returns the total number of rows that match the specified criteria. For instance, to find the total number of employees who have less than 5 years of experience, the given query can be used.

Note: A column name of the table can also be used instead of `*`. Unlike `COUNT(*)`, this variation `COUNT(column)` will not count `NULL` values in that column.

```
SELECT COUNT(*)
FROM employees
WHERE experience < 5;
```

`SUM()` Aggregate Function

The `SUM()` aggregate function takes the name of a column as an argument and returns the sum of all the value in that column.

```
SELECT SUM(salary)
FROM salary_disbursement;
```

`AVG()` Aggregate Function

The `AVG()` aggregate function returns the average value in a column. For instance, to find the average `salary` for the employees who have less than 5 years of experience, the given query can be used.

```
SELECT AVG(salary)
FROM employees
WHERE experience < 5;
```

`ROUND()` Function

The `ROUND()` function will round a number value to a specified number of places. It takes two arguments: a number, and a number of decimal places. It can be combined with other aggregate functions, as shown in the given query. This query will calculate the average rating of movies from 2015, rounding to 2 decimal places.

```
SELECT year,
   ROUND(AVG(rating), 2)
FROM movies
WHERE year = 2015;
```

`GROUP BY` Clause

The `GROUP BY` clause will group records in a result set by identical values in one or more columns. It is often used in combination with aggregate functions to query information of similar records. The `GROUP BY` clause can come after `FROM` or `WHERE` but must come before any `ORDER BY` or `LIMIT` clause.
The given query will count the number of movies per rating.

```
SELECT rating,
   COUNT(*)
FROM movies
GROUP BY rating;
```

Column References

The `GROUP BY` and `ORDER BY` clauses can reference the selected columns by number in which they appear in the `SELECT` statement. The example query will count the number of movies per rating, and will:

- `GROUP BY` column `2` (`rating`)
- `ORDER BY` column `1` (`total_movies`)

```
SELECT COUNT(*) AS 'total_movies',
   rating
FROM movies
GROUP BY 2
ORDER BY 1;
```

`HAVING` Clause

The `HAVING` clause is used to further filter the result set groups provided by the `GROUP BY` clause. `HAVING` is often used with aggregate functions to filter the result set groups based on an aggregate property. The given query will select only the records (rows) from only years where more than 5 movies were released per year.

```
SELECT year,
   COUNT(*)
FROM movies
GROUP BY year
HAVING COUNT(*) > 5;
```

`MAX()` Aggregate Function

The `MAX()` aggregate function takes the name of a column as an argument and returns the largest value in a column. The given query will return the largest value from the `amount` column.

```
SELECT MAX(amount)
FROM transactions;
```

`MIN()` Aggregate Function

The `MIN()` aggregate function returns the smallest value in a column. For instance, to find the smallest value of the `amount` column from the table named `transactions` , the given query can be used.

```
SELECT MIN(amount)
FROM transactions;
```