

# **Week 4**

MySQL Aggregate Functions

# Functions

- A function is code that performs a ***specific*** task and that ***can be reused*** together later in a program
- For example these are functions you have already used:

## JavaScript Functions

```
window.prompt();  
  
parseInt(input);  
  
console.log(input);  
  
arr.pop();  
  
arr.push();
```

## MySQL Function

```
CONCAT(column_1, column_2)
```

# Functions Dissected

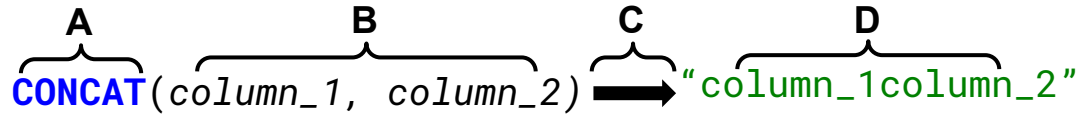
Functions...

- A. ...are identified & called by their names
- B. ..."take in" data (i.e. Inputs)
- C. ...complete some process
- D. ..."return" some data, changed



A diagram illustrating the function `parseInt`. A bracket labeled 'A' is above `parseInt`. A bracket labeled 'B' is above the input `"15"`. A thick white arrow labeled 'C' points from the input to the output `15`, which is bracketed by 'D'.

```
parseInt("15"); → 15
```



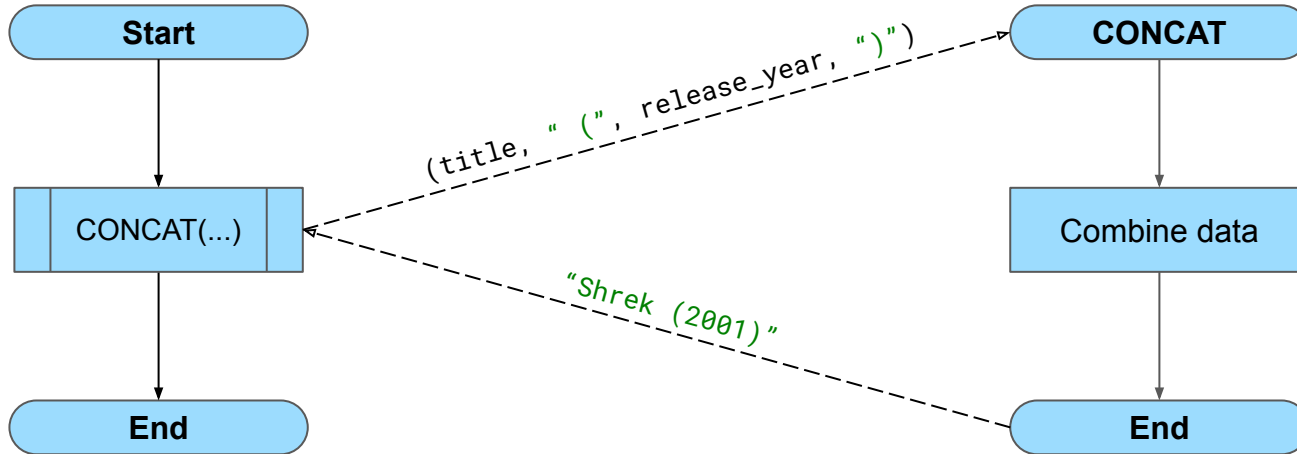
A diagram illustrating the function `CONCAT`. A bracket labeled 'A' is above `CONCAT`. A bracket labeled 'B' is above the inputs `column_1` and `column_2`. A thick black arrow labeled 'C' points from the inputs to the output `"column_1column_2"`, which is bracketed by 'D'.

```
CONCAT(column_1, column_2) → "column_1column_2"
```

# Function Flowchart

```
SELECT CONCAT(title, " (", release_year, ")") FROM movie;
```

movie_id	title	runtime	genre	release_year
7	Shrek	89	Animation	2001



# SQL Aggregate Functions<sup>1</sup>

- **Function** - Reusable code that performs a specific task
- **Aggregate** - A whole formed by combining several elements
- **Aggregate Function** - Reusable code that combines several data values to return a single value
  - Sum data, average data, count data

Aggregate functions...

- ...are identified & called by their names
- ...“take in” data from multiple rows
- ...complete some process that condenses the data into a single number
- ...then “returns” the processed single number

<sup>1</sup> [https://www.w3schools.com/sql/sql\\_aggregate\\_functions.asp](https://www.w3schools.com/sql/sql_aggregate_functions.asp)

# SQL Aggregate Functions<sup>1</sup>

**MAX():** **SELECT MAX(runtime) FROM** movie;

---

**MIN():** **SELECT MIN(release\_year) FROM** movie;

---

**AVG():** **SELECT AVG(runtime) FROM** movie;

---

**SUM():** **SELECT SUM(runtime) FROM** movie;

---

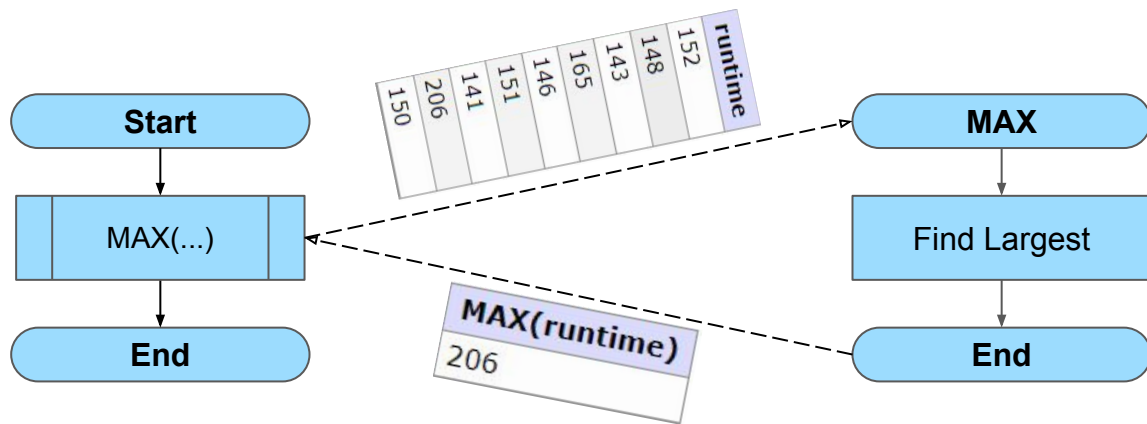
**COUNT():** **SELECT COUNT(\*) FROM** movie;

<sup>1</sup> [https://www.w3schools.com/sql/sql\\_aggregate\\_functions.asp](https://www.w3schools.com/sql/sql_aggregate_functions.asp)

# MAX()<sup>1</sup> Function

- **MAX()** returns the largest value of the selected column
  - Example: Find the longest running film

```
SELECT MAX(runtime) FROM movie;
```



runtime
152
148
143
165
146
151
141
206
150

## Syntax

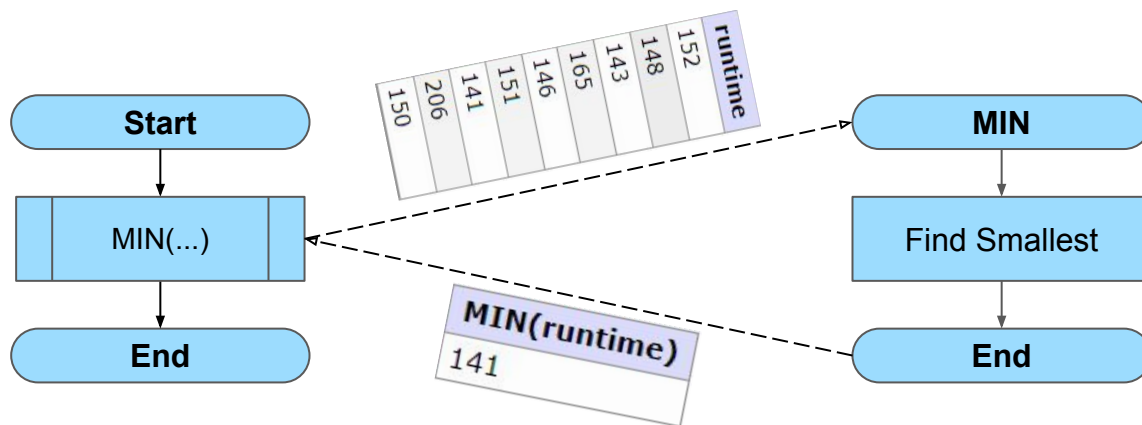
```
SELECT MAX(column) FROM table_name;
```

<sup>1</sup> [https://www.w3schools.com/sql/sql\\_min\\_max.asp](https://www.w3schools.com/sql/sql_min_max.asp)

# MIN()<sup>1</sup> Function

- **MIN()** returns the smallest value of the selected column
  - Example: Find the shortest running film

```
SELECT MIN(runtime) FROM movie;
```



runtime
152
148
143
165
146
151
141
206
150

## Syntax

```
SELECT MIN(column) FROM table_name;
```

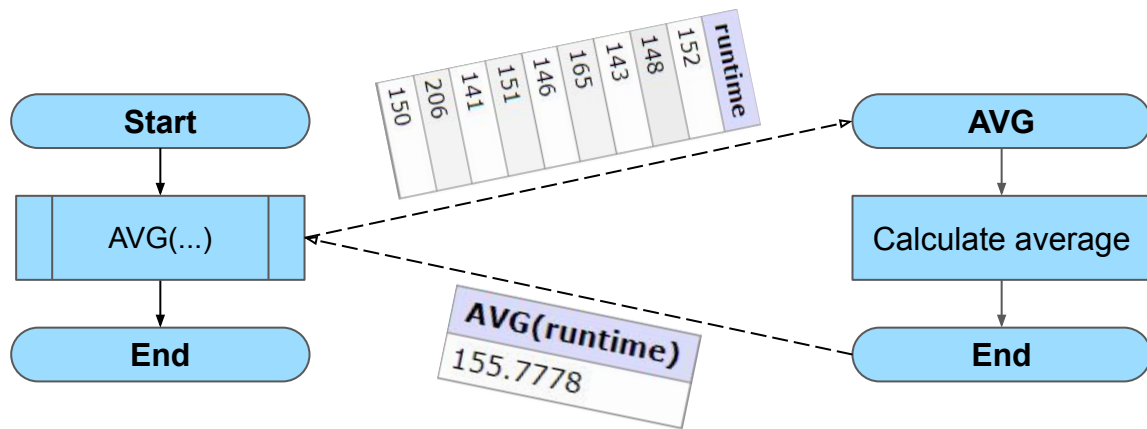
<sup>1</sup> [https://www.w3schools.com/sql/sql\\_min\\_max.asp](https://www.w3schools.com/sql/sql_min_max.asp)



# AVG()<sup>1</sup> Function

- **AVG()** returns the smallest value of the selected column
  - Example: Find the average runtime

```
SELECT AVG(runtime) FROM movie;
```



## Syntax

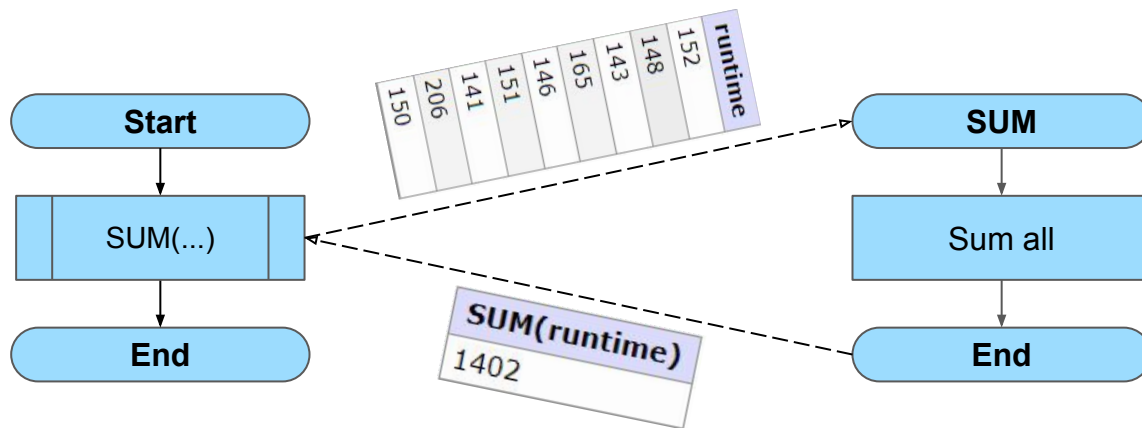
```
SELECT AVG(column) FROM table_name;
```

<sup>1</sup>[https://www.w3schools.com/sql/sql\\_avg.asp](https://www.w3schools.com/sql/sql_avg.asp)

# SUM()<sup>1</sup> Function

- **SUM()** returns the smallest value of the selected column
  - Example: Find the total runtime of movies combined

```
SELECT SUM(runtime) FROM movie;
```



## Syntax

```
SELECT SUM(column) FROM table_name;
```

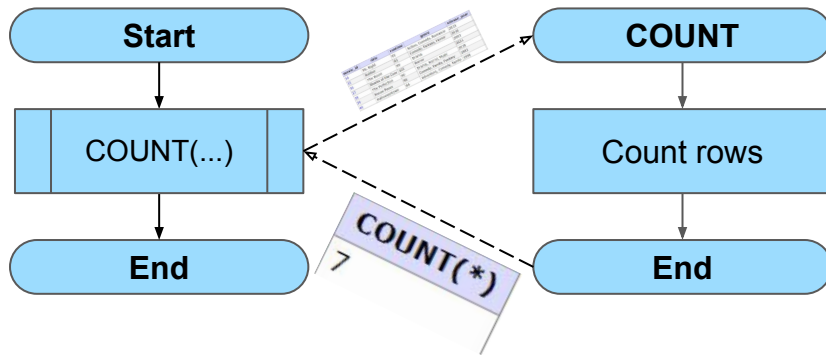
<sup>1</sup> [https://www.w3schools.com/sql/sql\\_sum.asp](https://www.w3schools.com/sql/sql_sum.asp)

# COUNT()<sup>1</sup> Function

- **COUNT()** returns the number of rows that were selected
  - Example: How many movies are in the database?

**SELECT COUNT(\*) FROM movie;**

movie_id	title	runtime	genre	release_year
34	Mr. Right	95	Action, Comedy, Romance	2015
35	Rubber	82	Comedy, Fantasy, Horror	2010
36	The Room	99	Drama	2003
37	Sharks of the Corn	105	Horror	2021
38	The Perfection	90	Drama, Horror, Music	2018
39	Hocus Pocus	96	Comedy, Family, Fantasy	1993
40	Halloweentown	84	Adventure, Comedy, Family	1998



## Syntax

**SELECT COUNT(column) FROM table\_name;**

<sup>1</sup> [https://www.w3schools.com/sql/sql\\_count.asp](https://www.w3schools.com/sql/sql_count.asp)

# GROUP BY<sup>1</sup>

- **GROUP BY** statement groups rows with the same values into summary rows
  - Example: Count all the movies grouped by their release year

```
SELECT COUNT(movie_id), release_year
FROM movie
GROUP BY release_year;
```

COUNT(movie_id)	release_year
1	2024
19	2023
5	2022
5	2021
5	2020
6	2019
6	2018

- **GROUP BY** can be used with the aggregate functions to group the result-set by one or more columns

**COUNT()**, **MAX()**,  
**MIN()**, **SUM()**, **AVG()**

## Syntax

```
SELECT AGG_FUNCTION(column) FROM table_name
GROUP BY column_1, column_2, ...;
```

<sup>1</sup> [https://www.w3schools.com/sql/sql\\_groupby.asp](https://www.w3schools.com/sql/sql_groupby.asp)

# HAVING<sup>1</sup>

- **HAVING**, added because **WHERE** cannot be used with aggregate functions
  - Example: Count all the movies grouped by their release year

```
SELECT COUNT(movie_id), release_year
FROM movie GROUP BY release_year
HAVING COUNT(movie_id)>5;
```

COUNT(movie_id)	release_year
19	2023
6	2019
6	2018
7	2012
7	2004
6	2000

## Syntax

```
SELECT AGG_FUNCTION(column) FROM table_name
GROUP BY column(s) HAVING condition(s);
```

<sup>1</sup> [https://www.w3schools.com/sql/sql\\_having.asp](https://www.w3schools.com/sql/sql_having.asp)

# SELECT Syntax

```
SELECT column(s), FUNCTION(column), ...  
  FROM table_name  
  WHERE condition(s)  
  GROUP BY column(s)  
  HAVING condition(s)  
  ORDER BY column(s) ASC|DESC  
  LIMIT num_limit  
  OFFSET num_offset;
```

## **Next (Week 5)**

- **Lesson - Multiple Tables**
- **Lab 5 (8%)**

# Terminology

- **Aggregate Functions** - function that performs a calculation on a set of values, and returns a single value
  - In SQL - **MAX**, **MIN**, **AVG**, **SUM**, **COUNT**



# SQL Keywords

**MAX()** : **SELECT MAX**(runtime) **FROM** movie;

---

**MIN()** : **SELECT MIN**(release\_year) **FROM** movie;

---

**AVG()** : **SELECT AVG**(runtime) **FROM** movie;

---

**SUM()** : **SELECT SUM**(runtime) **FROM** movie;

---

**COUNT()** : **SELECT COUNT**(\*) **FROM** movie;

---

**GROUP BY** : **SELECT ... GROUP BY** release\_year;

---

**HAVING** : **SELECT ... HAVING COUNT**(movie\_id)>5;

# Practice

- [W3Schools SQL](#)  
MAX, MIN, AVG, SUM, COUNT, GROUP BY, HAVING
- [SQL BOLT](#)
  - SQL Lesson 8: A short note on NULLs
  - SQL Lesson 9: Queries with expressions
  - SQL Lesson 10: Queries with aggregates (Pt. 1)
  - SQL Lesson 11: Queries with aggregates (Pt. 2)