

Summarizing data

INTERMEDIATE SQL



Jasmin Ludolf

Data Science Content Developer,
DataCamp

Summarizing data

- Aggregate functions return a single value



Aggregate functions

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

```
SELECT AVG(budget)
FROM films;
```

```
| avg          |
|-----|
| 39902826.2684... |
```

```
SELECT SUM(budget)
FROM films;
```

```
| sum          |
|-----|
| 181079025606 |
```

Aggregate functions

```
SELECT MIN(budget)
FROM films;
```

```
|min|
|---|
|218|
```

```
SELECT MAX(budget)
FROM films;
```

```
|max          |
|-----|
|12215500000|
```

Non-numerical data

Numerical fields only

- `AVG()`
- `SUM()`

Various data types

- `COUNT()`
- `MIN()`
- `MAX()`

Non-numerical data

`MIN()` <-> `MAX()`

Minimum <-> Maximum

Lowest <-> Highest

A <-> Z

1715 <-> 2022

0 <-> 100

Non-numerical data

```
SELECT MIN(country)
FROM films;
```

```
|min      |
|-----|
|Afghanistan|
```

```
SELECT MAX(country)
FROM films;
```

```
|max      |
|-----|
|West Germany|
```

Aliasing when summarizing

```
SELECT MIN(country)
FROM films;
```

```
|min      |
|-----|
|Afghanistan|
```

```
SELECT MIN(country) AS min_country
FROM films;
```

```
|min_country|
|-----|
|Afghanistan|
```


Let's practice!
INTERMEDIATE SQL

Summarizing subsets

INTERMEDIATE SQL



Jasmin Ludolf

Data Science Content Developer,
DataCamp

Using WHERE with aggregate functions

```
SELECT AVG(budget) AS avg_budget
FROM films
WHERE release_year >= 2010;
```

```
| avg_budget          |
|-----|
| 41072235.18324607... |
```

Using WHERE with aggregate functions

```
SELECT SUM(budget) AS sum_budget
FROM films
WHERE release_year = 2010;
```

```
|sum_budget|
|-----|
|8942365000|
```

```
SELECT MIN(budget) AS min_budget
FROM films
WHERE release_year = 2010;
```

```
|min_budget|
|-----|
|65000      |
```

Using WHERE with aggregate functions

```
SELECT MAX(budget) AS max_budget
FROM films
WHERE release_year = 2010;
```

```
|max_budget|
|-----|
|6000000000|
```

```
SELECT COUNT(budget) AS count_budget
FROM films
WHERE release_year = 2010;
```

```
|count_budget|
|-----|
|194          |
```

ROUND()

- Round a number to a specified decimal

```
SELECT AVG(budget) AS avg_budget
FROM films
WHERE release_year >= 2010;
```

```
| avg_budget |
|-----|
| 41072235.18324607... |
```

```
ROUND(number_to_round, decimal_places)
```

```
SELECT ROUND(AVG(budget), 2) AS avg_budget
FROM films
WHERE release_year >= 2010;
```

```
| avg_budget |
|-----|
| 41072235.18 |
```

ROUND() to a whole number

```
SELECT ROUND(AVG(budget)) AS avg_budget
FROM films
WHERE release_year >= 2010;
```

```
| avg_budget |
|-----|
| 41072235  |
```

```
SELECT ROUND(AVG(budget), 0) AS avg_budget
FROM films
WHERE release_year >= 2010;
```

```
| avg_budget |
|-----|
| 41072235  |
```

ROUND() using a negative parameter

```
SELECT ROUND(AVG(budget), -5) AS avg_budget
FROM films
WHERE release_year >= 2010;
```

```
| avg_budget |
|-----|
| 41100000  |
```

- Numerical fields only

Let's practice!

INTERMEDIATE SQL

Aliasing and arithmetic

INTERMEDIATE SQL



Jasmin Ludolf

Data Science Content Developer,
DataCamp

Arithmetic

`+` , `-` , `*` , and `/`

```
SELECT (4 + 3);
```

```
|7|
```

```
SELECT (4 - 3);
```

```
|1|
```

```
SELECT (4 * 3);
```

```
|12|
```

```
SELECT (4 / 3);
```

```
|1|
```

Arithmetic

```
SELECT (4 / 3);
```

```
|1|
```

```
SELECT (4.0 / 3.0);
```

```
|1.333...|
```

Aggregate functions vs. arithmetic

Aggregate functions

title	ticket_price	fees	tax
The Host	5	1	0.5
The Mask	5	1	0.5
Titanic	6	2	0.6

Arithmetic

title	ticket_price	fees	tax
The Host	5	1	0.5
The Mask	5	1	0.5
Titanic	6	2	0.6

Aliasing with arithmetic

```
SELECT (gross - budget)
FROM films;
```

```
|?column?|
|-----|
|null    |
|2900000 |
|null    |
|...     |
```

```
SELECT (gross - budget) AS profit
FROM films;
```

```
|profit |
|-----|
|null   |
|2900000|
|null   |
|...    |
```

Aliasing with functions

```
SELECT MAX(budget), MAX(duration)
FROM films;
```

```
| max          | max |
|-----|---|
| 12215500000 | 334 |
```

```
SELECT MAX(budget) AS max_budget,
       MAX(duration) AS max_duration
FROM films;
```

```
| max_budget | max_duration |
|-----|-----|
| 12215500000 | 334          |
```

Order of execution

- Step 1: FROM
 - Step 2: WHERE
 - Step 3: SELECT (aliases are defined here)
 - Step 4: LIMIT
-
- Aliases defined in the SELECT clause cannot be used in the WHERE clause due to order of execution

```
SELECT budget AS max_budget  
FROM films  
WHERE max_budget IS NOT NULL;
```

```
column "max_budget" does not exist  
LINE 5: WHERE max_budget IS NOT NULL;  
                ^
```


Let's practice!
INTERMEDIATE SQL