Group functions

INTRODUCTION TO ORACLE SQL

Hadrien Lacroix
Content Developer

Aggregating data

Name	Milliseconds
Restless and Wild	252,051
Breaking The Rules	263,288
Whole Lotta Rosie	323,761
You Oughta Know	249,234
Perfect	188,133
Ironic	229,825
Master Of Puppets	436,453
Twist And Shout	161,123
The Alchemist	509,413
Into The Light	76,303
Midnight Blue	298,631
...	...

Maximum Milliseconds in Track table

MAX(Milliseconds)
5,286,953

Group functions

- `SUM`
- `AVG`
- `MEDIAN`
- `MIN`
- `MAX`
- `COUNT`

SUM

```
SELECT SUM(Milliseconds)
FROM Track
```

```
| SUM(Milliseconds) |
|-------------------|
| 1,378,778,040     |
```

AVG, MEDIAN

```
SELECT AVG(Milliseconds), MEDIAN(Milliseconds)
FROM Track
```

```
| AVG(Milliseconds) | MEDIAN(Milliseconds) |
|-------------------|----------------------|
| 393,599.2         | 255,634              |
```

MIN, MAX

```
SELECT MIN(Milliseconds), MAX(Milliseconds)
FROM Track
```

```
| MIN(Milliseconds) | MAX(Milliseconds) |
|-------------------|-------------------|
| 1,071             | 5,286,953         |
```

COUNT

```
-- Number of rows in a table
SELECT COUNT(*)
FROM Track
```

```
| COUNT(*) |
|----------|
| 3503     |
```

```
-- Number of rows with non-null values
SELECT COUNT(Milliseconds)
FROM Track
```

```
| COUNT(Milliseconds) |
|---------------------|
| 3503                |
```

```
-- Number of distinct non-null values
SELECT COUNT(DISTINCT Milliseconds)
FROM Track
```

```
| COUNT(DISTINCT Milliseconds) |
|------------------------------|
| 3080                         |
```

Column aliases

```
SELECT MIN(Milliseconds)
       SUM(Milliseconds)
FROM Track
```

```
| MIN(Milliseconds) | SUM(Milliseconds) |
|-------------------|-------------------|
| 1,071             | 1,378,778,040     |
```

```
SELECT MIN(Milliseconds) AS minimum
       SUM(Milliseconds) AS "Total Duratio
FROM Track
```

```
| MINIMUM           | Total Duration    |
|-------------------|-------------------|
| 1,071             | 1,378,778,040     |
```

Data types

	Numeric data	Character data	Date data
`AVG`	x		
`SUM`	x		
`MIN`	x	x	x
`MAX`	x	x	x
`COUNT`	x	x	x

Character

`CHAR` , `VARCHAR2`

```
SELECT Name
FROM Track
```

```
| Name            |
|-----------------|
| Ashes To Ashes |
| Layla           |
| Everlong        |
| ...             |
```

Numeric

`NUMBER`

```
SELECT Milliseconds
FROM Track
```

```
| Milliseconds |
|--------------|
| 343719       |
| 205688       |
| 267728       |
| ...          |
```

Date

`DATE` , `DATETIME`

```
SELECT BirthDate
FROM Employee
```

```
| BirthDate  |
|------------|
| 1962-02-18 |
| 1958-12-08 |
| 1973-08-29 |
| ...        |
```

Let's practice!

INTRODUCTION TO ORACLE SQL

Creating groups of data

INTRODUCTION TO ORACLE SQL

Hadrien Lacroix
Content Developer

Grouping data

Composer	Milliseconds
Antonio Vivaldi	199,086
Pearl Jam	122,801
Pearl Jam	65,593
Jimmy Page	401,920
Jimmy Page	386,063
Jimmy Page	132,702
Jimmy Page	189,675
Jimmy Page	126,641
Carlos Santana	126,641
Carlos Santana	296,437
Carlos Santana	882,834
...	...

Average Milliseconds in Track table for each composer

Composer	AVG(Milliseconds)
Antonio Vivaldi	199,086.0
Pearl Jam	94,197.0
Jimmy Page	474,888.3
Carlos Santana	499,234.3
...	...

Group information

Composer	Milliseconds
Antonio Vivaldi	199,086
Pearl Jam	122,801
Pearl Jam	65,593
Jimmy Page	401,920
Jimmy Page	386,063
Jimmy Page	132,702
Jimmy Page	189,675
Jimmy Page	126,641
Carlos Santana	318,432
Carlos Santana	296,437
Carlos Santana	882,834
...	...

What is the average track length of songs written by each composer?

- `GROUP BY`
  - divide the rows in a table into groups
  - use group functions to get summary information for each group

GROUP BY

```
SELECT Composer, AVG(Milliseconds)
FROM Track
GROUP BY Composer
```

```
| Composer         | AVG(Milliseconds) |
|------------------|-------------------|
| Antonio Vivaldi | 199,086.0         |
| Pearl Jam        | 94,197.0          |
| Jimmy Page       | 474,888.3         |
| Carlos Santana   | 499,234.3         |
| ...              | ...               |
```

GROUP BY and WHERE

```
SELECT Composer, AVG(Milliseconds)
FROM Track
WHERE Genre = 1
GROUP BY Composer
```

```
| Composer         | AVG(Milliseconds) |
|------------------|-------------------|
| Antonio Vivaldi | 199,086.0         |
| Pearl Jam        | 94,197.0          |
| Jimmy Page       | 474,888.3         |
| Carlos Santana   | 499,234.3         |
| ...              | ...               |
```

GROUP BY and ORDER BY

```
SELECT Composer, AVG(Milliseconds) AS Average
FROM Track
GROUP BY Composer
ORDER BY AVG(Milliseconds)
```

```
| Composer         | AVG(Milliseconds) |
|------------------|-------------------|
| Pearl Jam        | 94,197.0          |
| Antonio Vivaldi  | 199,086.0         |
| Jimmy Page       | 474,888.3         |
| Carlos Santana   | 499,234.3         |
| ...              | ...               |
```

GROUP BY and ORDER BY

```
SELECT Composer, AVG(Milliseconds) AS Average
FROM Track
GROUP BY Composer
ORDER BY 2
```

```
| Composer         | AVG(Milliseconds) |
|------------------|-------------------|
| Pearl Jam        | 94,197.0          |
| Antonio Vivaldi  | 199,086.0         |
| Jimmy Page       | 474,888.3         |
| Carlos Santana   | 499,234.3         |
| ...              | ...               |
```

GROUP BY and ORDER BY

```
SELECT Composer, AVG(Milliseconds) AS Average
FROM Track
GROUP BY Composer
ORDER BY Average
```

```
| Composer         | Average             |
|------------------|---------------------|
| Pearl Jam        | 94,197.0            |
| Antonio Vivaldi  | 199,086.0           |
| Jimmy Page       | 474,888.3           |
| Carlos Santana   | 499,234.3           |
| ...              | ...                 |
```

Guidelines

Any column or expression in the `SELECT` statement that is not an aggregate function must be in the `GROUP BY` clause

```
SELECT Composer, AVG(Milliseconds), UnitPrice
FROM Track
GROUP BY Composer
```

```
column "track.unitprice" must appear in the GROUP BY clause or be used in an aggregate function
LINE 2: SELECT Composer, AVG(Milliseconds), UnitPrice
```

Guidelines

Any column or expression in the `SELECT` list that is not an aggregate function must be in the `GROUP BY` clause

```
SELECT Composer, AVG(Milliseconds), MAX(UnitPrice)
FROM Track
GROUP BY Composer
```

```
| Composer         | AVG(Milliseconds) | MAX(UnitPrice) |
|------------------|-------------------|----------------|
| Antonio Vivaldi  | 199,086.0         | 0.99           |
| Pearl Jam        | 94,197.0          | 0.99           |
| Jimmy Page       | 474,888.3         | 0.99           |
| Carlos Santana   | 499,234.3         | 0.99           |
| ...              | ...               | ...            |
```

Guidelines

Expressions that are specified in the `GROUP BY` do not have to be included in the `SELECT` statement

```
SELECT AVG(Milliseconds)
FROM Track
GROUP BY Composer
```

```
| AVG(Milliseconds) |
|-------------------|
| 199,086.0         |
| 94,197.0          |
| 474,888.3         |
| 499,234.3         |
| ...               |
```

Multiple columns

```
SELECT Country, City, COUNT(CustomerId)
FROM Customer
GROUP BY Country, City
```

```
| Country   | City                 | COUNT(CustomerId) |
|-----------|----------------------|-------------------|
| Argentina | Buenos Aires         | 1                 |
| Australia | Sidney               | 1                 |
| Austria   | Vienne               | 1                 |
| Belgium   | Brussels             | 1                 |
| Brazil    | Brasilia             | 1                 |
| Brazil    | São José dos Campos | 1                 |
| Brazil    | São Paulo            | 2                 |
| ...       | ...                  | ...               |
```

Let's practice!

INTRODUCTION TO ORACLE SQL

Restricting group results

INTRODUCTION TO ORACLE SQL

Hadrien Lacroix
Content Developer

Back to our example

```
SELECT Composer, AVG(Milliseconds)
FROM Track
WHERE UnitPrice = 0.99
GROUP BY Composer
```

```
| Composer         | AVG(Milliseconds) |
|------------------|-------------------|
| Antonio Vivaldi | 199,086.0         |
| Pearl Jam        | 94,197.0          |
| Jimmy Page       | 474,888.3         |
| Carlos Santana   | 499,234.3         |
| ...              | ...               |
```

What about filtering after grouping?

Limits of WHERE

- `WHERE` can't be used to filter groups
  - Group functions can't be used in `WHERE` clauses

Limits of WHERE example

```
SELECT Composer, AVG(Milliseconds)
FROM Track
GROUP BY Composer
WHERE AVG(Milliseconds) > 200000
```

```
syntax error at or near "WHERE"
LINE 4: WHERE AVG(Milliseconds) > 200000
        ^
```

HAVING

```
SELECT Composer, AVG(Milliseconds)
FROM Track
GROUP BY Composer
HAVING AVG(Milliseconds) > 200000
```

```
| composer     | avg    |
|--------------|--------|
| George Duke | 274155 |
| Miles Davis | 391146 |
| R. Carless  | 251585 |
| ...          | ...    |
```

Restricting group results with HAVING

Composer	Milliseconds
Antonio Vivaldi	199,086
Pearl Jam	122,801
Pearl Jam	65,593
Jimmy Page	401,920
Jimmy Page	386,063
Jimmy Page	132,702
Jimmy Page	189,675
Jimmy Page	126,641
Carlos Santana	126,641
Carlos Santana	296,437
Carlos Santana	882,834
...	...

The maximum song length per artist when it is greater than 200,000 milliseconds

Composer	MAX(Milliseconds)
Jimmy Page	401,920
Carlos Santana	882,834
...	...

HAVING

```
SELECT Composer, MAX(Milliseconds)
FROM Track
GROUP BY Composer
HAVING MAX(Milliseconds) > 200000
```

```
| composer     | max    |
|--------------|--------|
| George Duke | 274155 |
| Miles Davis | 907520 |
| R. Carless  | 251585 |
| ...          | ...    |
```

Another example

```
SELECT Composer, SUM(UnitPrice)
FROM Track
WHERE GenreId = 1
GROUP BY Composer
HAVING COUNT(*) > 4
```

```
| composer       | sum   |
|----------------|------|
| AC/DC          | 7.92 |
| Chris Cornell | 9.90 |
| Eddie Vedder   | 8.91 |
| ...            | ...   |
```

Guidelines

- Different group functions can be used in `SELECT` and `HAVING`
- `HAVING` always has a grouping function

Order of operations:

1. Rows are filtered by `WHERE` and `GROUP BY`
2. Group function applied to the groups
3. The groups are filtered by `HAVING` then outputted

Let's practice!

INTRODUCTION TO ORACLE SQL