FUNCTIONS

Belangrijke info

AS: het keyword AS wordt gebruikt om een veld een anderen naam te geven in een weergave of een nieuw samengesteld veld een naam te geven. AS is dus wat noemt een **alias** van een bestaand of nieuw veld.

In MySQL kunnen we daarnaast ook functies toepassen. Functies worden toegepast op bestaande velden in. Functies worden gekenmerkt door de naamgeving en ronde haakjes. Bijvoorbeeld: de functie MIN() haalt de laagste waarde uit een kolom.

Hieronder zie je een lijst van veel gebruikte functies

COUNT	De MySQL COUNT-aggregatiefunctie wordt gebruikt om het aantal rijen in een databasetabel te tellen.
MAX	Met de MySQL MAX-aggregatiefunctie kunnen we de hoogste (maximale) waarde voor een bepaalde kolom selecteren.
MIN	Met de MySQL MIN-aggregatiefunctie kunnen we de laagste (minimum) waarde voor een bepaalde kolom selecteren.
AVG	De MySQL AVG-aggregatiefunctie selecteert de gemiddelde waarde voor een bepaalde tabelkolom.
SUM	Met de MySQL SUM-aggregatiefunctie kunt u het totaal voor een numerieke kolom selecteren.
SQRT	Dit wordt gebruikt om een vierkantswortel van een bepaald getal te genereren.
RAND	Dit wordt gebruikt om een willekeurig getal te genereren met behulp van de MySQL-opdracht.
CONCAT	Dit wordt gebruikt om elke tekenreeks binnen een MySQL-opdracht samen te voegen .
LENGTH	Geeft de lengte van een strin gerug
CURDATE	Retourneert de huidige datum
CURTIME	Retourneert de huidige tijd
ADDDATE	Telt dagen bij een datum op
ADDTIME	Telt tijd bij tijd op
DATEFORMAT	Hier kan je de weergave van een datum mee wijzigen
DATE_SUB	Hier kan je 2 data van elkaar aftrekken
DATE	Geeft de datum van vandaag terug
WEEKDAY	Retourneert het weekdag nummer (index)
WEEK	Retourneert het weeknummer (index)

MySQL telt dus heel wat BUILT-IN functions. Er zijn er een paar honderd. Alle functies kun je hieronder terugvinden via de volgende link:
MySQL :: MySQL 8.4 Reference Manual :: 14.1 Built-In Function and Operator Reference

- Selecteer het hoogste bedrag uit de tabel payment

```
SELECT max(amount)
from payment
```

- Hoeveel rijen telt de tabel payment

```
SELECT max(amount)
from payment
```

- Selecteer de gemiddelde prijs van de bedragen uit de tabel payment

```
SELECT avg(amount)
FROM payment
```

KLASSIKALE OEFENINGEN

1. Film Titels in Hoofdletters

Schrijf een query die de titels van alle films in hoofdletters retourneert.

Query:

```
SELECT UPPER(title) AS film_titel
FROM film;
```

2. Lengte van de Film Titel

Haal de titels van alle films op en geef ook de lengte van elke titel weer.

Query:

```
SELECT title, LENGTH(title) AS titel_lengte
FROM film;
```

3. Film Titels zonder Spaties aan de Randen

Geef de film titels weer waarbij eventuele leidende en afsluitende spaties worden verwijderd.

Query:

```
SELECT TRIM(title) AS titel_bijgesneden
FROM film;
```

4. Films die een Bepaalde Tekst bevatten

Haal de titels van alle films op die het woord ‘love’ in hun titel hebben (ongeacht hoofdletters).

Query:

```
SELECT title
FROM film
WHERE LOWER(title) LIKE '%love%';
```

5. Eerste en Laatste Teken van Acteursnamen

Toon de voornaam van elke acteur, samen met het eerste en laatste teken van hun voornaam.

Query:

```
SELECT first_name, LEFT(first_name, 1) AS eerste_karakter, RIGHT(first_name,
1) AS laatste_karakter
FROM actor;
```

SELECT count(distinct last_name) as aantal

FROM actor

GROUP BY

Belangrijke info

Een group by groepeert rijen/velden die identiek zijn aan elkaar tot één rij.
Let op: bij een GROUP BY dien je altijd ALLE VELDEN over te nemen die in het SELECT gedeelte staan ZONDER de velden die van een functie zouden voorzien zijn.

Voorbeeld van een group by :

- Selecteer alle familienamen uit de tabel actor en voeg daarna een tweede kolom toe die het aantal per naam weergeeft. Sorteer dan volgens het aantal van Z-A.
  ```
  SELECT last_name, count(last_name) as aantal
  FROM actor
  GROUP BY last_name
  ORDER BY aantal DESC
  ```
- Selecteer alle familienamen en voornamen uit de tabel actor en voeg daarna een tweede kolom toe die het aantal per naam weergeeft. Sorteer dan volgens het aantal van Z-A.
  ```
  SELECT last_name, first_name, count(last_name) as aantal
  FROM actor
  GROUP BY last_name, first_name
  ORDER BY aantal DESC
  ```

HAVING

Belangrijke info

Een having clause wordt gebruikt wanneer we uit het resultaat van een group by dienen te FILTEREN. Een having is eigenlijk de where voor een GROUP BY

- Selecteer alle familienamen uit de actor tabel. Geef dan de lijst weer met familienamen die meer dan 1 x voorkomen!

```sql
select last_name
from actor
group by last_name
having count(*) > 1
```

Oefening 1

Vraag: Selecteer alle verschillende `rental_rate` waarden uit de `film` tabel.

```
SELECT DISTINCT rental_rate
FROM film;
```

Oefening 2

Vraag: Toon alle `actors` met de achternaam `SMITH` of `JONES`.

```
SELECT *
FROM actor
WHERE last_name = 'SMITH' OR last_name = 'JONES';
```

Oefening 3

Vraag: Selecteer de `film_id` en `title` van alle films met een `rental_rate` van meer dan 2.99, gesorteerd op `title`.

```
SELECT film_id, title
FROM film
WHERE rental_rate > 2.99
ORDER BY title;
```

Oefening 4

Vraag: Tel het aantal films per `rating` en toon enkel diegenen met meer dan 100 films.

```
SELECT rating, COUNT(*) AS film_count
FROM film
GROUP BY rating
HAVING COUNT(*) > 100;
```

Oefening 5

Vraag: Selecteer alle `customer_id` waar de `email` `NULL` is.

```
SELECT customer_id
FROM customer
WHERE email IS NULL;
```

Oefening 6

Vraag: Selecteer alle films (`title`) met een `rental_duration` van 5 dagen en een `replacement_cost` tussen 15 en 25.

```
SELECT title
FROM film
WHERE rental_duration = 5 AND replacement_cost BETWEEN 15 AND 25;
```

Oefening 7

Vraag: Toon alle `inventory_id`'s waarvan de `store_id` niet gelijk is aan 2.

```sql
SELECT inventory_id
FROM inventory
WHERE NOT store_id = 2;
```

Oefening 8

Vraag: Selecteer de verschillende `store_id` waarden uit de `staff` tabel.

```sql
SELECT DISTINCT store_id
FROM staff;
```

Oefening 9

Vraag: Selecteer het aantal klanten (`customer_id`) per `store_id`.

```sql
SELECT store_id, COUNT(customer_id) AS customer_count
FROM customer
GROUP BY store_id;
```

Oefening 10

Vraag: Toon alle `film` titels waarvan de `replacement_cost` groter is dan 20, gesorteerd op aflopende `replacement_cost`.

```sql
SELECT title
FROM film
WHERE replacement_cost > 20
ORDER BY replacement_cost DESC;
```

Oefening 11

Vraag: Selecteer alle `category_id`'s van de `category` tabel waarvan de `name` niet `Children` is.

```sql
SELECT category_id
FROM category
WHERE NOT name = 'Children';
```

Oefening 12

Vraag: Tel het aantal `inventory_id` per `film_id` en toon enkel resultaten met meer dan 20 exemplaren.

```sql
SELECT film_id, COUNT(inventory_id) AS inventory_count
FROM inventory
GROUP BY film_id
HAVING COUNT(inventory_id) > 20;
```

Oefening 13

Vraag: Selecteer de `actor_id` en `first_name` voor acteurs met de achternaam die begint met `K`.

```sql
SELECT actor_id, first_name
FROM actor
WHERE last_name LIKE 'K%';
```

Oefening 14

Vraag: Toon het totaal aantal films per `rental_duration` en toon alleen de gevallen met een `rental_duration` van meer dan 3 dagen.

```sql
SELECT rental_duration, COUNT(*) AS film_count
FROM film
GROUP BY rental_duration
HAVING rental_duration > 3;
```

Oefening 15

Vraag: Selecteer alle `rental_id` waarbij de `return_date` nog niet is ingevuld (NULL).

```sql
SELECT rental_id
FROM rental
WHERE return_date IS NULL;
```

Oefening 16

Vraag: Selecteer alle verschillende `ratings` uit de `film` tabel.

```sql
SELECT DISTINCT rating
FROM film;
```

Oefening 17

Vraag: Tel het aantal klanten per `store_id` waarbij `active` gelijk is aan 1.

```sql
SELECT store_id, COUNT(customer_id) AS active_customer_count
FROM customer
WHERE active = 1
GROUP BY store_id;
```

Oefening 18

Vraag: Toon alle `actor_id`'s waarvan de `last_name` niet gelijk is aan `DAVIS` of `JOHNSON`.

```sql
SELECT actor_id
FROM actor
WHERE last_name NOT IN ('DAVIS', 'JOHNSON');
```

Oefening 19

Vraag: Selecteer alle `payment_id`'s waarbij de `amount` groter is dan 5 en de `payment_date` is in 2005.

```sql
SELECT payment_id
FROM payment
WHERE amount > 5 AND payment_date LIKE '2005%';
```

Oefening 20

Vraag: Toon de `city_id` en tel het aantal steden in elke `country_id` met meer dan 5 steden.

```
SELECT country_id, COUNT(city_id) AS city_count
FROM city
GROUP BY country_id
HAVING COUNT(city_id) > 5;
```

Oefening 21

Vraag: Selecteer alle `film_id` waarbij de `description` niet `NULL` is.

```
SELECT film_id
FROM film
WHERE description IS NOT NULL;
```

Oefening 22

Vraag: Toon de `store_id` en tel hoeveel `inventory_id`'s elke `store` heeft.

```
SELECT store_id, COUNT(inventory_id) AS inventory_count
FROM inventory
GROUP BY store_id;
```

Oefening 23

Vraag: Selecteer alle `actor_id` en `last_name` waar de `first_name` gelijk is aan `NICK`.

```
SELECT actor_id, last_name
FROM actor
WHERE first_name = 'NICK';
```

Oefening 24

Vraag: Toon alle `customer_id` waarbij `create_date` gelijk is aan `2006-02-14`.

```
SELECT customer_id
FROM customer
WHERE create_date = '2006-02-14';
```

Oefening 25

Vraag: Tel het aantal verschillende `film_id`'s per `store_id` in de `inventory` tabel.

```
SELECT store_id, COUNT(DISTINCT film_id) AS unique_film_count
FROM inventory
GROUP BY store_id;
```

Oefening 26

Vraag: Selecteer de `rental_id` en `rental_date` voor alle verhuurtransacties in februari 2005.

```sql
SELECT rental_id, rental_date
FROM rental
WHERE rental_date LIKE '2005-02%';
```

Oefening 27

Vraag: Tel het aantal `address_id` per `city_id` en toon enkel steden met meer dan 3 adressen.

```sql
SELECT city_id, COUNT(address_id) AS address_count
FROM address
GROUP BY city_id
HAVING COUNT(address_id) > 3;
```

Oefening 28

Vraag: Selecteer alle `customer_id` waar de klant niet actief is (`active` = 0).

```sql
SELECT customer_id
FROM customer
WHERE active = 0;
```

Oefening 29

Vraag: Selecteer de `title` en `film_id` voor films met een `length` tussen 90 en 120 minuten.

```sql
SELECT title, film_id
FROM film
WHERE length BETWEEN 90 AND 120;
```

Oefening 30

Vraag: Toon alle verschillende `store_id` waarden uit de `customer` tabel.

```sql
SELECT DISTINCT store_id
FROM customer;
```