SQL Part02

SQL LIKE Operator

- The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

- SQL LIKE Syntax
- SELECT *column_name(s)*
  FROM *table_name*
  WHERE *column_name* LIKE *pattern*;

SQL Wildcards

- In SQL, wildcard characters are used with the SQL LIKE operator.
- SQL wildcards are used to search for data within a table.

Wildcard	Description
%	A substitute for zero or more characters
_	A substitute for a single character

- Example
- SELECT * FROM Customers
  WHERE City LIKE 'L_n_on';

The IN Operator

- The IN operator allows you to specify multiple values in a WHERE clause.

- SQL IN Syntax
- SELECT *column_name(s)*
  FROM *table_name*
  WHERE *column_name* IN (*value1*,*value2*,...);

- Example
- SELECT * FROM Customers
  WHERE City IN ('Paris','London');

SQL BETWEEN Operator

- The BETWEEN operator selects values within a range. The values can be numbers, text, or dates.

- SQL BETWEEN Syntax
  - SELECT *column_name(s)*
    FROM *table_name*
    WHERE *column_name* BETWEEN *value1* AND *value2;*

- Example
- SELECT * FROM Products
  WHERE Price BETWEEN 10 AND 20;
-
  Example
- SELECT * FROM Products
  WHERE Price NOT BETWEEN 10 AND 20;

SQL Aliases

- SQL aliases are used to give a database table, or a column in a table, a temporary name.
- Basically aliases are created to make column names more readable.

- SQL Alias Syntax for Columns
- SELECT *column_name* AS *alias_name*
  FROM *table_name;*

- SQL Alias Syntax for Tables
- SELECT *column_name(s)*
  FROM *table_name* AS *alias_name;*

- Example
- SELECT CustomerName AS Customer, ContactName AS Contact Person
  FROM Customers;

- Example
- SELECT o.OrderID, o.OrderDate, c.CustomerName
  FROM Customers AS c, Orders AS o
  WHERE c.CustomerName="Around the Horn" AND
  c.CustomerID=o.CustomerID;

SQL ALTER TABLE Statement

- The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.
-

  ALTER TABLE table_name
  ADD column_name datatype

- ALTER TABLE table_name
  DROP COLUMN column_name

- Now we want to add a column named "DateOfBirth" in the "Persons" table.
- ALTER TABLE Persons
  ADD DateOfBirth date
- Now we want to change the data type of the column named "DateOfBirth" in the "Persons" table.
- ALTER TABLE Persons
  ALTER COLUMN DateOfBirth year

- Next, we want to delete the column named "DateOfBirth" in the "Persons" table.
- ALTER TABLE Persons
  DROP COLUMN DateOfBirth

The DROP TABLE Statement

- The DROP TABLE statement is used to delete a table.
    - DROP TABLE table_name

SQL Views

- A view is a virtual table.
- A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

- SQL CREATE VIEW Syntax
  - CREATE VIEW view_name AS
    SELECT column_name(s)
    FROM table_name
    WHERE condition
- Select * from view name

SQL Functions- Aggregate Functions

- SQL aggregate functions return a single value, calculated from values in a column.
- Useful aggregate functions:
- AVG() - Returns the average value
- COUNT() - Returns the number of rows
- MAX() - Returns the largest value
- MIN() - Returns the smallest value
- SUM() - Returns the sum

- COUNT()- The COUNT function returns the total number of values in the specified field.
- It works on both numeric and non-numeric data types.

- COUNT (*) is a special implementation of the COUNT function that returns the count of all the rows in a specified table. COUNT (*) also considers Nulls and duplicates.

reference_ number	transaction_ date	return_date	membership_ number	movie_id	movie_ returned
11	20-06-2012	NULL	1	1	0
12	22-06-2012	25-06-2012	1	2	0
13	22-06-2012	25-06-2012	3	2	0
14	21-06-2012	24-06-2012	2	2	0
15	23-06-2012	NULL	3	3	0

Let's suppose that we want to get the number of times that the movie with id 2 has been rented out

```
SELECT COUNT(`movie_id`)  FROM `movierentals` WHERE `movie_id` = 2;
```

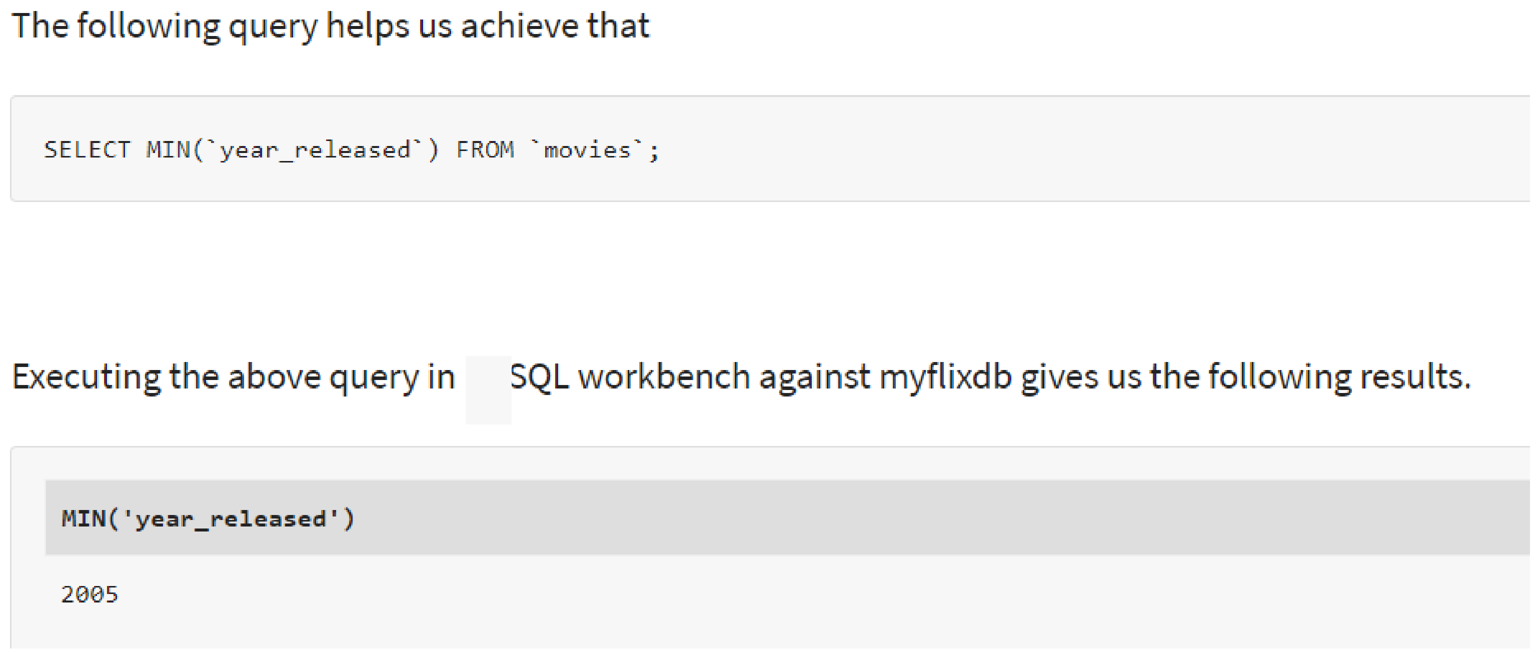
Executing the above query in SQL workbench against myflixdb gives us the following results.

```
COUNT('movie_id')

3
```

MIN() function

- The MIN function returns the smallest value in the specified table field.
- As an example, let's suppose we want to know the year in which the oldest movie in our library was released, we can use SQL's MIN function to get the desired information.



The following query helps us achieve that

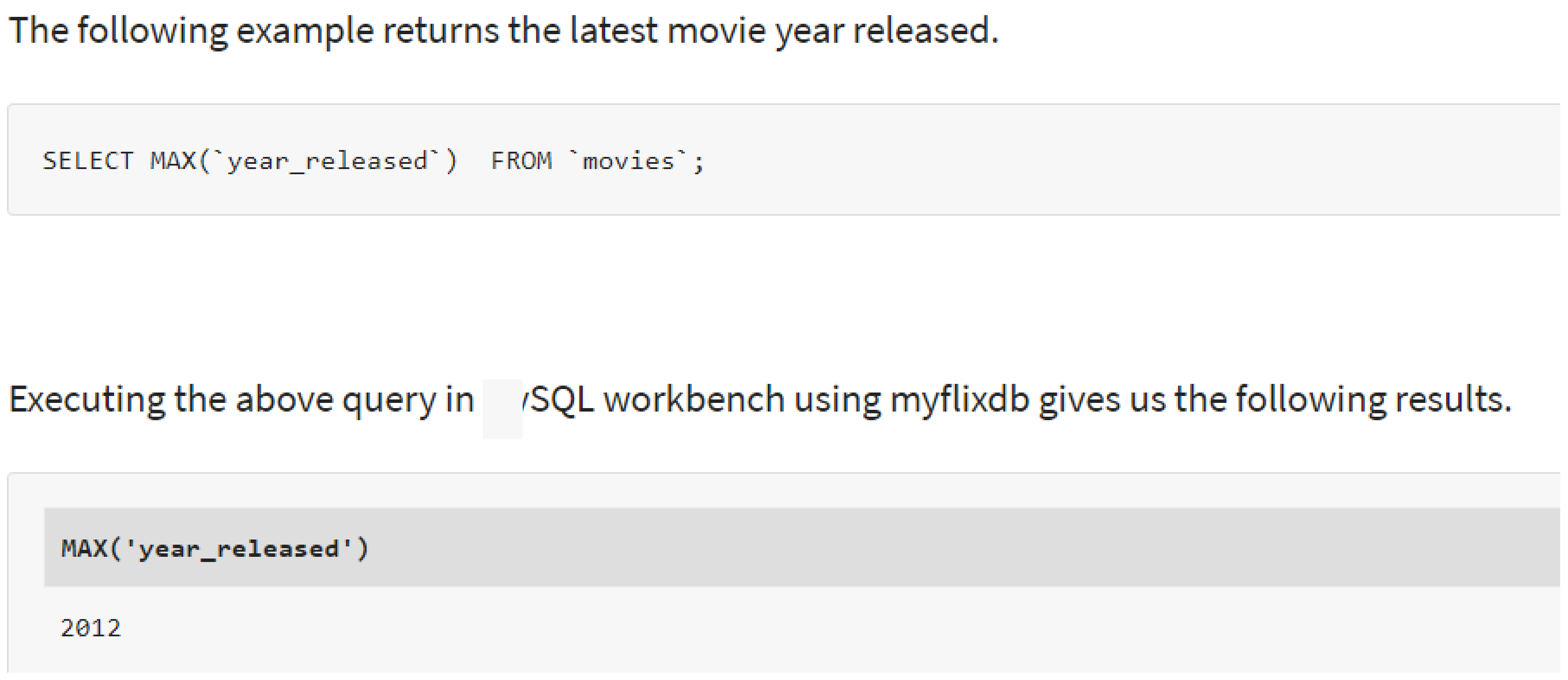
```
SELECT MIN(`year_released`) FROM `movies`;
```

Executing the above query in SQL workbench against myflixdb gives us the following results.

```
MIN('year_released')
2005
```

MAX function

- Just as the name suggests, the MAX function is the opposite of the MIN function. It **returns the largest value from the specified table field**.


The following example returns the latest movie year released.

```
SELECT MAX(`year_released`)  FROM `movies`;
```

Executing the above query in ySQL workbench using myflixdb gives us the following results.

```
MAX('year_released')
2012
```

SUM function

- SUM function
- Suppose we want a report that gives total amount of payments made so far.
- We can use the SQL SUM function which returns the sum of all the values in the specified column.
- SUM works on numeric fields only.
- Null values are excluded from the result returned.

The following table shows the data in payments table-

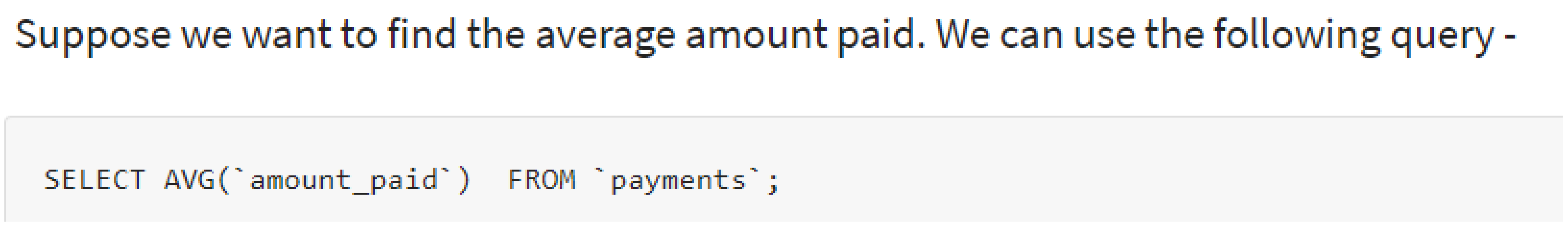
payment_ id	membership_ number	payment_ date	description	amount_ paid	external_ reference _number
1	1	23-07-2012	Movie rental payment	2500	11
2	1	25-07-2012	Movie rental payment	2000	12
3	3	30-07-2012	Movie rental payment	6000	NULL

```
SELECT SUM(`amount_paid`) FROM `payments`;
```

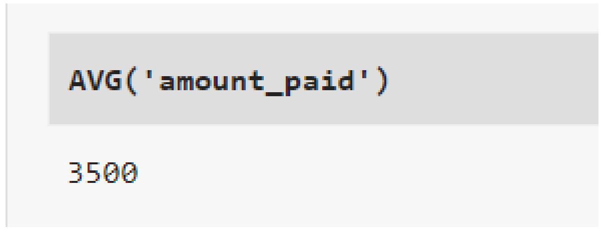
```
SUM('amount_paid')
10500
```

AVG function

- SQL AVG function **returns the average of the values in a specified column**. Just like the SUM function, it **works only on numeric data types**.


Suppose we want to find the average amount paid. We can use the following query -

```
SELECT AVG(`amount_paid`)  FROM `payments`;
```



```
AVG('amount_paid')

3500
```

- SELECT AVG(Price) AS PriceAverage FROM Products;
- SELECT COUNT(CustomerID) AS OrdersFromCustomerID7 FROM Orders WHERE CustomerID=7;

- SELECT MAX(Price) AS HighestPrice FROM Products;

SQL GROUP BY Statement

- The GROUP BY statement is used in conjunction with the aggregate functions to group the result-set by one or more columns.

- SQL GROUP BY Syntax
  - SELECT column_name, aggregate_function(column_name)
    FROM table_name
    WHERE column_name operator value
    GROUP BY column_name;

- SELECT Shippers.ShipperName,COUNT(Orders.OrderID) AS NumberOfOrders FROM Orders
  LEFT JOIN Shippers
  ON Orders.ShipperID=Shippers.ShipperID
  GROUP BY ShipperName;