Database: A database is simply a structured collection of data

- The data within a database are naturally related, for example, a product belongs to a product category and is associated with multiple tags. Hence, we use the term relational database.
- In a relational database, we model data like products, categories, tags, etc., using tables. A table contains columns and rows, much like a spreadsheet.
- Tables can relate to one another table using various types of relationships, like one-to-one and one-to-many.
- Because we handle a substantial amount of data, we need a way, to efficiently define a databases, tables, and process data. Moreover, we want to transform data into valuable information.

SQL – the language of the relational database
SQL is composed of three parts:

1. Data definition language (DDL) includes statements for defining the database and its objects such as tables, views, triggers, stored procedures, etc.
2. Data manipulation language (DML) contains statements for updating and querying data.
3. Data control language (DCL) allows you to grant permissions to users to access specific data in the database

SQL – Connect command
-Use cmd or powershell and type: mysql –u root –p then wait for the password
-If you are using mysql shell then: \connect --mysql -u root -p

Using DB command
-CREATE DATABASE IF NOT EXISTS classicmodels
-SHOW DATABASES:
-USE <db_name>: this will show all db
-SHOW TABLES: this will show all tables in the selected db

Section 1. Querying data
Introduction to MySQL SELECT FROM statement:
-The SELECT statement allows you to select data from one or more tables. To write a SELECT statement in MySQL, you use this syntax:

```
SELECT select_list FROM table_name; or SELECT select_list FROM
table_name LIMIT 10;
```

-First, specify one or more columns from which you want to select data after the SELECT keyword. If the select_list has multiple columns, you need to separate them by a comma (,)
-When executing the SELECT statement, MySQL evaluates the FROM clause before the SELECT clause:
- use SELECT * FROM <table_name>\G to print in vertical line
-In MySQL, the SELECT statement doesn't require the FROM clause. It means that you can have a SELECT statement without the FROM clause like this: SELECT 1 + 1;
-MySQL has many built-in functions like string functions, date functions, and math functions. You can use the SELECT statement to execute one of these functions. For example, the following statement uses the NOW() function in the SELECT statement to return the current date and time of the server where MySQL server is running: SELECT NOW();
Column alias
-By default, MySQL uses the expression specified in the SELECT clause as the column name of the result set. To change a column name of the result set, you can use a column alias:
-SELECT expression AS column_alias;
To assign an alias to a column, you place the AS keyword after the expression followed by a column alias. The AS keyword is optional, so you can skip it like this:

```
SELECT expression column_alias;
```

-If the column alias contains spaces, you need to place it inside quotes like this:
`SELECT CONCAT('Jane',' ','Doe') AS 'Full name';`

ORDER BY clause
- To sort the rows in the result set, you add the ORDER BY clause to the SELECT statement.

```
    SELECT select_list FROM table_name ORDER BY column1 [ASC|DESC],
column2 [ASC|DESC], ...;
```

- When executing the SELECT statement with an ORDER BY clause, MySQL always evaluates the ORDER BY clause after the FROM and SELECT clauses: FROM-> SELECT -> ORDER BY
-Using column alias and expression: `SELECT orderNumber, orderLineNumber, quantityOrdered * priceEach AS subtotal FROM orderdetails ORDER BY subtotal DESC;`

Using MySQL ORDER BY clause to sort data using a custom list
-The FIELD() function returns the index (position) of a value within a list of values.
Here's the syntax of the FIELD() function: FIELD(value, value1, value2, ...)
Code language: SQL (Structured Query Language) (sql)
In this syntax:

- value: The value for which you want to find the position.
- value1, value2, ...: A list of values against which you want to compare the specified value.

The FIELD() function returns the position of the value in the list of values value1, value2, and so on. If the value is not found in the list, the FIELD() function returns 0.

```
SELECT  orderNumber, status FROM orders ORDER BY FIELD(status, 'In
Process', 'On Hold', 'Cancelled', 'Resolved', 'Disputed', 'Shipped');
```

MySQL ORDER BY and NULL
In MySQL, NULL comes before non-NULL values. Therefore, when you the ORDER BY clause with the ASC option, NULLs appear first in the result set.

WHERE clause
-The WHERE clause allows you to specify a search condition for the rows returned by a query.
The following shows the syntax of the WHERE clause:-
`SELECT select_list FROM table_name WHERE search_condition;`
-The search_condition is a combination of one or more expressions using the logical operator AND, OR and NOT.
-In MySQL, a predicate is a Boolean expression that evaluates to TRUE, FALSE, or UNKNOWN.
-The SELECT statement will include any row that satisfies the search_condition in the result set.
-Besides the SELECT statement, you can use the WHERE clause in the UPDATE or DELETE statement to specify which rows to update or delete.
-When executing a SELECT statement with a WHERE clause, MySQL evaluates the WHERE clause after the FROM clause and before the SELECT and ORDER BY clauses:
FROM -> WHERE -> SELECT -> ORDER BY

1) Using the WHERE clause with equality operator example
   ```
   SELECT lastname, firstname, jobtitle FROM employees WHERE jobtitle =
   'Sales Rep';
   ```
2) Using the WHERE clause with the AND operator
   ```
   SELECT lastname, firstname, jobtitle, officeCode FROM employees WHERE
   jobTitle='Sales Rep' AND officeCode=1;
   ```

3) Using MySQL WHERE clause with OR operator:

```
      SELECT lastName, firstName, jobTitle,vofficeCode FROM employees
WHERE jobtitle = 'Sales Rep' OR officeCode = 1 ORDER BY officeCode ,
jobTitle;
```

4) Using the WHERE clause with the BETWEEN operator example

-The BETWEEN operator returns TRUE if a value is in a range of values: expression BETWEEN low AND high

```
      SELECT lastName, firstName, jobTitle, officeCode FROM employees
WHERE officeCode BETWEEN 3 AND 7 ORDER by lastName DESC;
```

5) Using the WHERE clause with the LIKE operator example

-The LIKE operator evaluates to TRUE if a value matches a specified pattern.

-You can use two wildcards with LIKE:

- % - Represents zero, one, or multiple characters
- _ - Represents a single character (MS Access uses a question mark (?) instead)

CustomerName starting with "a": LIKE 'a%'

CustomerName ending with "a": LIKE '%a';

CustomerName that have "or" in any position: LIKE '%or%';

CustomerName that starts with "a" and are at least 3 characters in length: LIKE 'a__%';

6) Using the WHERE clause with the IN operator example

-The IN operator returns TRUE if a value matches any value in a list.

-value IN (value1, value2,...)

-The following example uses the WHERE clause with the IN operator to find employees m who are located in the offices with the codes 1, 2, and 3:

```
SELECT firstName, lastName, officeCode FROM employees WHERE
officeCode IN(1,2,3) ORDER BY officeCode;
```

7) Using MySQL WHERE clause with the IS NULL operator

-To check if a value is NULL or not, you use the IS NULL operator, not the equal operator (=). The IS NULL operator returns TRUE if a value is NULL.

value IS NULL

In the database world, NULL is a marker that indicates that a value is missing or unknown. NULL is not equivalent to the number 0 or an empty string.

The following statement uses the WHERE clause with the IS NULL operator to get the rows with the values in the reportsTo column are NULL:

```
SELECT lastName, firstName, reportsTo FROM employees WHERE reportsTo
IS NULL;
```

8) Using MySQL WHERE clause with comparison operators

The following table shows the comparison operators that you can use to form the expression in the WHERE clause.

OPERATOR	DESCRIPTION
=	Equal to. You can use it with almost any data type.
<> OR !=	Not equal to
<	Less than. You typically use it with numeric and date/time data types.
>	Greater than.
<=	Less than or equal to
>=	Greater than or equal to

The following query uses the not equal to (<>) operator to find all employees who are not the Sales Rep: `SELECT lastname, firstname, jobtitle FROM employees WHERE jobtitle <> 'Sales Rep';`

```
DISTINCT clause
```

When querying data from a table, you may get duplicate rows. To remove these duplicate rows, you use the DISTINCT clause in the SELECT statement.
SELECT DISTINCT select_list FROM table_name WHERE search_condition ORDER BY sort_expression;
FROM → WHERE→ SELECT → DISTINCT → ORDER BY

AND clause
MySQL doesn't have a built-in Boolean type. Instead, it uses the number zero as FALSE and non-zero values as TRUE.
The AND operator is a logical operator that combines two or more Boolean expressions and returns 1, 0, or NULL: A AND B
In this expression, A and B are called operands. They can be literal values or expressions.

```
SELECT customername, country, state, creditlimit FROM customers WHERE country='USA' AND state='CA' AND creditlimit >100000;
```

IN clause
The IN operator allows you to determine if a value matches any value in a list of values.
Here's the syntax of the IN operator: value IN (value1, value2, value3,...)
The IN operator returns 1 (true) if the value equals any value in the list (value1, value2, value3,...). Otherwise, it returns 0.
In this syntax:

- First, specify the value to test on the left side of the IN operator. The value can be a column or an expression.
- Second, specify a comma-separated list of values to match in the parentheses.

The IN operator is functionally equivalent to a combination of multiple OR operators:
value = value1 OR value = value2 OR value = value3 OR ...

```
SELECT officeCode, city, phone, country FROM offices WHERE country IN ('usa','france');
Use the IN operator to form a condition for the WHERE clause.
```

NOT IN clause
The NOT operator negates the IN operator: value NOT IN (value1, value2, value2)
The following example uses the NOT IN operator to find the offices that are not located in France and the USA:

```
SELECT officeCode, city, phone FROM offices WHERE country NOT IN ('USA' 'France') ORDER BY city;
```

BETWEEN Operator
The BETWEEN operator is a logical operator that specifies whether a value is in a range or not.
Here's the syntax of the BETWEEN operator:
value BETWEEN low AND high;
The BETWEEN operator returns 1 if:
value >= low AND value <= highCode language: SQL (Structured Query Language) (sql)
Otherwise, it returns 0.
If the value, low, or high is NULL, the BETWEEN operator returns NULL

```
Eg: SELECT productCode,productName,buyPrice FROM products WHERE
```

```
    buyPrice < 20 OR buyPrice > 100;
```

LIKE Operator

The LIKE operator is a logical operator that tests whether a string contains a specified pattern or not.
In this syntax, if the expression matches the pattern, the LIKE operator returns 1. Otherwise, it returns 0.
MySQL provides two wildcard characters for constructing patterns: Percentage % and underscore _ .

- The percentage (%) wildcard matches any string of zero or more characters.
- The underscore (_) wildcard matches any single character.

1) LIKE operator with wildcard(%)

Eg: SELECT `employeeNumber, lastName, firstName FROM employees`
`WHERE firstName LIKE 'a%';`
'a%' = The word will start from 'a';
'an%' = word will start from 'an';
'on%' = word will end with 'on';
%on% = if the string contains substring;

2) LIKE operator with wildcard(_)

SELECT employeeNumber, lastName, firstName FROM employees WHERE firstname LIKE 'T_m';
employees whose first names start with the letter T , end with the letter m, and contain any single character

3) NOT LIKE

want to search for employees whose last names don't start with the letter B, you can use the NOT LIKE, Eg:
SELECT employeeNumber, lastName, firstName FROM employees
WHERE lastName NOT LIKE 'B%';
Note: The pattern is not case-sensitive. Therefore, the b% and B% patterns return the same result.

4) ESCAPE clause

We can use the ESCAPE clause to specify the escape character so that the LIKE operator interprets the wildcard character as a literal character.If you don't specify the escape character explicitly, the backslash character (\) is the default escape character.
For example, if you want to find products whose product codes contain the string _20 , you can use the pattern %_20% with the default escape character:
SELECT productCode, productName FROM products WHERE productCode LIKE '%_20%';

Alternatively, you can specify a different escape character e.g., $ using the ESCAPE clause:
SELECT productCode, productName
FROM products
WHERE productCode LIKE '%$_20%' ESCAPE '$';