

DATABASE SYSTEMS

FACULTY OF AI & MMG

Structured Query Language

MySQL

- MySQL is a very popular open-source relational database management system (RDBMS).
- MySQL is a relational database management system
- MySQL is free & open-source
- MySQL is ideal for both small and large applications
- MySQL is very fast, reliable, scalable, and easy to use
- MySQL is compliant with the ANSI SQL standard
- MySQL was first released in 1995
- MySQL is developed, distributed, and supported by Oracle Corporation
- MySQL is named after co-founder Ulf Michael "Monty" Widenius's daughter: My

Relational Database Management System (RDBMS)

- RDBMS is a program used to maintain a relational database. A relational database is a way to organize data in tables, rows, and columns. It can establish relationships between data points by joining tables.
- RDBMS is the basis for all modern database systems such as MySQL, Microsoft SQL Server, Oracle, and Microsoft Access.
- RDBMS uses SQL queries to access the data in the database.
- SQL is the standard language for dealing with Relational Databases.
- SQL keywords are NOT case sensitive: *select* is the same as *SELECT*
- Some database systems require a semicolon at the end of each SQL statement.

DDL

- CREATE
- DROP
- ALTER
- TRUNCATE

DML

- INSERT
- UPDATE
- DELETE

DCL

- GRANT
- REVOKE

TCL

- COMMIT
- ROLLBACK
- SAVEPOINT

DQL

- SELECT

Structured Query Language (SQL)

Overview of DDL Statements

- **Data Definition Language (DDL):**
 - The CREATE statement is used to create a new database, table, index, or other database objects.
 - Used to define and modify database structures.
 - Common statements: CREATE, ALTER, DROP, TRUNCATE, RENAME, CREATE INDEX, DROP INDEX.

Syntax for Creating a Database:

sql

```
CREATE DATABASE database_name;
```

Example:

sql

```
CREATE DATABASE my_database;
```

This creates a new database named `my_database`.

CREATE STATEMENTS

Make sure you have admin privilege before creating any database. For Checking:

Show databases;

Syntax for Creating a Table:

sql

```
CREATE TABLE table_name (  
    column1 datatype constraints,  
    column2 datatype constraints,  
    ...  
);
```

Example:

sql

```
CREATE TABLE Customers (  
    CustomerID INT PRIMARY KEY,  
    FirstName VARCHAR(50),  
    LastName VARCHAR(50),  
    Email VARCHAR(100)  
);
```

CREATE STATEMENTS

This creates a table named Customers with four columns: *CustomerID*, *FirstName*, *LastName*, and *Email*.

MySQL Data Types

- Each column in a database table is required to have a name and a data type.
- In MySQL there are three main data types: *string*, *numeric*, and *date or time*.

String Data Types

Data type	Description
CHAR(size)	A FIXED length string (can contain letters, numbers, and special characters). The <i>size</i> parameter specifies the column length in characters - can be from 0 to 255. Default is 1
VARCHAR(size)	A VARIABLE length string (can contain letters, numbers, and special characters). The <i>size</i> parameter specifies the maximum column length in characters - can be from 0 to 65535
BINARY(size)	Equal to CHAR(), but stores binary byte strings. The <i>size</i> parameter specifies the column length in bytes. Default is 1
VARBINARY(size)	Equal to VARCHAR(), but stores binary byte strings. The <i>size</i> parameter specifies the maximum column length in bytes.
TINYBLOB	For BLOBs (Binary Large Objects). Max length: 255 bytes
TINYTEXT	Holds a string with a maximum length of 255 characters
TEXT(size)	Holds a string with a maximum length of 65,535 bytes
BLOB(size)	For BLOBs (Binary Large Objects). Holds up to 65,535 bytes of data
MEDIUMTEXT	Holds a string with a maximum length of 16,777,215 characters
MEDIUMBLOB	For BLOBs (Binary Large Objects). Holds up to 16,777,215 bytes of data
LONGTEXT	Holds a string with a maximum length of 4,294,967,295 characters
LOBLOB	For BLOBs (Binary Large Objects). Holds up to 4,294,967,295 bytes of data
ENUM(val1, val2, val3, ...)	A string object that can have only one value, chosen from a list of possible values. You can list up to 65535 values in an ENUM list. If a value is inserted that is not in the list, a blank value will be inserted. The values are sorted in the order you enter them
SET(val1, val2, val3, ...)	A string object that can have 0 or more values, chosen from a list of possible values. You can list up to 64 values in a SET list

MySQL Data Types

- Numeric datatypes

Data type	Description
BIT(<i>size</i>)	A bit-value type. The number of bits per value is specified in <i>size</i> . The <i>size</i> parameter can hold a value from 1 to 64. The default value for <i>size</i> is 1.
TINYINT(<i>size</i>)	A very small integer. Signed range is from -128 to 127. Unsigned range is from 0 to 255. The <i>size</i> parameter specifies the maximum display width (which is 255)
BOOL	Zero is considered as false, nonzero values are considered as true.
BOOLEAN	Equal to BOOL
SMALLINT(<i>size</i>)	A small integer. Signed range is from -32768 to 32767. Unsigned range is from 0 to 65535. The <i>size</i> parameter specifies the maximum display width (which is 255)
MEDIUMINT(<i>size</i>)	A medium integer. Signed range is from -8388608 to 8388607. Unsigned range is from 0 to 16777215. The <i>size</i> parameter specifies the maximum display width (which is 255)
INT(<i>size</i>)	A medium integer. Signed range is from -2147483648 to 2147483647. Unsigned range is from 0 to 4294967295. The <i>size</i> parameter specifies the maximum display width (which is 255)
INTEGER(<i>size</i>)	Equal to INT(<i>size</i>)
BIGINT(<i>size</i>)	A large integer. Signed range is from -9223372036854775808 to 9223372036854775807. Unsigned range is from 0 to 18446744073709551615. The <i>size</i> parameter specifies the maximum display width (which is 255)
FLOAT(<i>size</i> , <i>d</i>)	A floating point number. The total number of digits is specified in <i>size</i> . The number of digits after the decimal point is specified in the <i>d</i> parameter. This syntax is deprecated in MySQL 8.0.17, and it will be removed in future MySQL versions
FLOAT(<i>p</i>)	A floating point number. MySQL uses the <i>p</i> value to determine whether to use FLOAT or DOUBLE for the resulting data type. If <i>p</i> is from 0 to 24, the data type becomes FLOAT(). If <i>p</i> is from 25 to 53, the data type becomes DOUBLE()
DOUBLE(<i>size</i> , <i>d</i>)	A normal-size floating point number. The total number of digits is specified in <i>size</i> . The number of digits after the decimal point is specified in the <i>d</i> parameter
DOUBLE PRECISION(<i>size</i> , <i>d</i>)	
DECIMAL(<i>size</i> , <i>d</i>)	An exact fixed-point number. The total number of digits is specified in <i>size</i> . The number of digits after the decimal point is specified in the <i>d</i> parameter. The maximum number for <i>size</i> is 65. The maximum number for <i>d</i> is 30. The default value for <i>size</i> is 10. The default value for <i>d</i> is 0.

Date and Time Data Types

Data type	Description
DATE	A date. Format: YYYY-MM-DD. The supported range is from '1000-01-01' to '9999-12-31'
DATETIME(<i>fsp</i>)	A date and time combination. Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'. Adding DEFAULT and ON UPDATE in the column definition to get automatic initialization and updating to the current date and time
TIMESTAMP(<i>fsp</i>)	A timestamp. TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC. Automatic initialization and updating to the current date and time can be specified using DEFAULT CURRENT_TIMESTAMP and ON UPDATE CURRENT_TIMESTAMP in the column definition
TIME(<i>fsp</i>)	A time. Format: hh:mm:ss. The supported range is from '-838:59:59' to '838:59:59'
YEAR	A year in four-digit format. Values allowed in four-digit format: 1901 to 2155, and 0000. MySQL 8.0 does not support year in two-digit format.

MYSQL DATA TYPES

Date and Time datatypes

ALTER Statements

- The ALTER statement is used to modify the structure of an existing table, such as adding, deleting, or modifying columns.
- This adds a new column named *PhoneNumber* to the Customers table.

Syntax for Adding a Column:

sql

```
ALTER TABLE table_name  
ADD column_name datatype;
```

Example:

sql

```
ALTER TABLE Customers  
ADD PhoneNumber VARCHAR(15);
```

ALTER Statements

Syntax for Modifying a Column:

sql

```
ALTER TABLE table_name  
MODIFY column_name new_datatype;
```

Example:

sql

```
ALTER TABLE Customers  
MODIFY Email VARCHAR(150);
```

This changes the `Email` column's data type to `VARCHAR(150)`.

ALTER Statements

Syntax for Dropping a Column:

sql

```
ALTER TABLE table_name  
DROP COLUMN column_name;
```

Example:

sql

```
ALTER TABLE Customers  
DROP COLUMN PhoneNumber;
```

This removes the `PhoneNumber` column from the `Customers` table.

MySQL Constraints

- SQL constraints are used to specify rules for data in a table.
- Constraints can be specified when the table is created with the CREATE TABLE statement, or after the table is created with the ALTER TABLE statement.
- This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.
- Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table.

MySQL Constraints

- The following constraints are commonly used in SQL:
 - NOT NULL - Ensures that a column cannot have a NULL value
 - UNIQUE - Ensures that all values in a column are different
 - PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
 - FOREIGN KEY - Prevents actions that would destroy links between tables
 - CHECK - Ensures that the values in a column satisfies a specific condition
 - DEFAULT - Sets a default value for a column if no value is specified
 - CREATE INDEX - Used to create and retrieve data from the database very quickly

MySQL Constraints

- MySQL NOT NULL Constraint
- By default, a column can hold NULL values.
- The NOT NULL constraint enforces a column to NOT accept NULL values.
- **NOT NULL on CREATE TABLE**
- The following SQL ensures that the "ID", "LastName", and "FirstName" columns will NOT accept NULL values when the "Persons" table is created:
- **NOT NULL on ALTER TABLE**
- To create a NOT NULL constraint on the "Age" column when the "Persons" table is already created, use the following SQL:

Example

```
CREATE TABLE Persons (  
    ID int NOT NULL,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255) NOT NULL,  
    Age int  
);
```

Example

```
ALTER TABLE Persons  
MODIFY Age int NOT NULL;
```


MySQL Constraints

- **MySQL DEFAULT Constraint**

- The DEFAULT constraint is used to set a default value for a column.
- The default value will be added to all new records, if no other value is specified.

- **DEFAULT on CREATE TABLE**

- The following SQL sets a DEFAULT value for the "City" column when the "Persons" table is created:

- **DEFAULT on ALTER TABLE**

- To create a DEFAULT constraint on the "City" column when the table is already created, use the following SQL:
- [More examples...](#)

```
CREATE TABLE Persons (  
    ID int NOT NULL,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255),  
    Age int,  
    City varchar(255) DEFAULT 'Sandnes'  
);
```

```
ALTER TABLE Persons  
ALTER City SET DEFAULT 'Sandnes';
```

Syntax for Dropping a Database:

sql

```
DROP DATABASE database_name;
```

Example:

sql

```
DROP DATABASE my_database;
```

This deletes the `my_database` database.

DROP

The DROP statement is used to delete an existing database, table, or index.

Drop Statements

Syntax for Dropping a Table:

sql

```
DROP TABLE table_name;
```

Example:

sql

```
DROP TABLE Customers;
```

This deletes the `Customers` table.

Truncate Statements

- The TRUNCATE statement is used to delete all rows from a table without deleting the table itself.
- This removes all rows from the Customers table but keeps the table structure intact.

Syntax:

sql

```
TRUNCATE TABLE table_name;
```

Example:

sql

```
TRUNCATE TABLE Customers;
```

Syntax:

sql

```
RENAME TABLE old_table_name TO new_table_name;
```

Example:

sql

```
RENAME TABLE Customers TO Clients;
```

This renames the `Customers` table to `Clients`.

RENAME STATEMENTS

The RENAME statement is used to rename a table.

Syntax:

sql

```
CREATE INDEX index_name  
ON table_name (column1, column2, ...);
```

Example:

sql

```
CREATE INDEX idx_lastname  
ON Customers (LastName);
```

CREATE INDEX

- The CREATE INDEX statement is used to create an index on one or more columns of a table, which improves query performance.
- This creates an index named idx_lastname on the LastName column of the Customers table.

Syntax:

sql

```
DROP INDEX index_name ON table_name;
```

Example:

sql

```
DROP INDEX idx_lastname ON Customers;
```

DROP INDEX

- The DROP INDEX statement is used to delete an index.

- This deletes the idx_lastname index from the Customers table.

Summary of DDL Statements:

Statement	Purpose	Example
CREATE	Create a database, table, or index	<code>CREATE TABLE Customers (ID INT, Name VARCHAR(50));</code>
ALTER	Modify the structure of a table	<code>ALTER TABLE Customers ADD Email VARCHAR(100);</code>
DROP	Delete a database, table, or index	<code>DROP TABLE Customers;</code>
TRUNCATE	Delete all rows from a table	<code>TRUNCATE TABLE Customers;</code>
RENAME	Rename a table	<code>RENAME TABLE Customers TO Clients;</code>
CREATE INDEX	Create an index on a table	<code>CREATE INDEX idx_name ON Customers (Name);</code>
DROP INDEX	Delete an index	<code>DROP INDEX idx_name ON Customers;</code>

DATA MANIPULATION LANGUAGE (DML)

- INSERT
- UPDATE
- DELETE

INSERT INTO Statement

- The INSERT INTO statement is used to insert new records in a table.
- INSERT INTO Syntax
 - It is possible to write the INSERT INTO statement in two ways:
 1. Specify both the column names and the values to be inserted:

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

```
INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)
VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway');
```

INSERT INTO Statement

2. If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table.

```
INSERT INTO table_name  
VALUES (value1, value2, value3, ...);
```

Note: A NULL value is different from a zero value or a field that contains spaces. A field with a NULL value is one that has been left blank during record creation!

UPDATE Statement

- The UPDATE statement is used to modify the existing records in a table.
- *Note:*
 - when updating records in a table! Notice the WHERE clause in the UPDATE statement.
 - The WHERE clause specifies which record(s) that should be updated.
 - If you omit the WHERE clause, all records in the table will be updated!

```
UPDATE table_name  
SET column1 = value1, column2 = value2, ...  
WHERE condition;
```

```
UPDATE Customers  
SET ContactName = 'Alfred Schmidt', City = 'Frankfurt'  
WHERE CustomerID = 1;
```

```
UPDATE Customers  
SET PostalCode = 00000;
```

DELETE Statement

```
DELETE FROM table_name WHERE condition;
```

- The DELETE statement is used to delete existing records in a table.
- Note:
 - while deleting records in a table! Notice the WHERE clause in the DELETE statement.
 - The WHERE clause specifies which record(s) should be deleted.
 - If you omit the WHERE clause, all records in the table will be deleted!

```
DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';
```

```
DELETE FROM Customers;
```

DATA CONTROL LANGUAGE (DCL)

- GRANT
- REVOKE

Data Control Language (DCL)

- DCL is used to control access to the data in a database. It includes commands that grant or revoke user permissions.

1.**GRANT** – Provides specific privileges to users.

2.**REVOKE** – Removes specific privileges from users.

Syntax:

```
GRANT privilege_name ON object_name TO user_name;
```

```
REVOKE privilege_name ON object_name FROM user_name;
```

Example:

```
GRANT SELECT, INSERT ON Students TO Ali_123;
```

```
REVOKE INSERT ON Students FROM Ali_123;
```

Transaction Control Language (TCL)

- TCL is used to manage database transactions of the data.

1. COMMIT – Saves all changes made in the current transaction.

```
INSERT INTO Students (RollNo, Name, Age, Department)
VALUES (2501, 'Ali', 22, 'Computer Science');
COMMIT;
```

2. ROLLBACK – Undoes all changes in the current transaction.

```
DELETE FROM Students WHERE RollNo = 2501;
ROLLBACK;
```

3. SAVEPOINT – Sets a save point within a transaction to which you can later roll back.

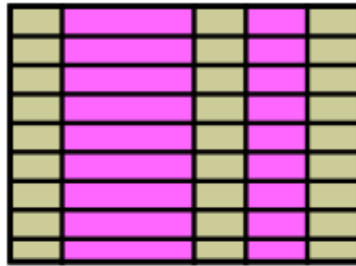
```
INSERT INTO Students (RollNo, Name, Age, Department)
VALUES (2502, 'Ahmed', 21, 'Electrical Engineering');
SAVEPOINT sp1;
DELETE FROM Students WHERE RollNo = 2502;
ROLLBACK TO sp1; -- This will undo the DELETE but keep the INSERT.
```


DATA QUERY LANGUAGE (DQL)

- SELECT

Capabilities of SELECT Statements

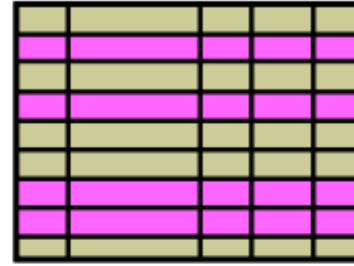
Projection



A 10x5 grid representing a table. The second and fourth columns are highlighted in pink, illustrating the result of a projection operation that selects specific columns from the original table.

Table 1

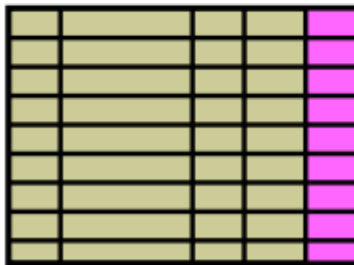
Selection



A 10x5 grid representing a table. The first three rows are highlighted in pink, illustrating the result of a selection operation that filters specific rows from the original table.

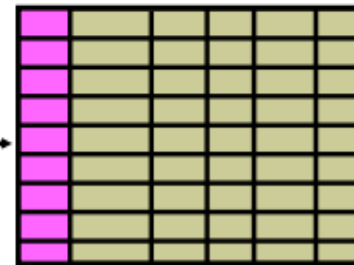
Table 1

Join



A 10x5 grid representing a table. The last two columns are highlighted in pink, representing the result of a join operation where specific columns from both original tables are combined.

Table 1



A 10x5 grid representing a table. The first two columns are highlighted in pink, representing the result of a join operation where specific columns from both original tables are combined.

Table 2

Data Query Language (DQL)

- DQL (Data Query Language) in MySQL is used to retrieve data from the database.
- The main DQL command in SQL is SELECT, which is used to fetch data from tables based on conditions.

Basic Syntax:

sql

```
SELECT column1, column2 FROM table_name WHERE condition;
```

Example:

Retrieve all columns from the `Students` table:

sql

```
SELECT * FROM Students;
```

Retrieve specific columns (`RollNo` , `Name`) from the `Students` table:

sql

```
SELECT RollNo, Name FROM Students;
```

Selecting All Columns

```
SELECT *  
FROM departments;
```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700

8 rows selected.

Selecting Specific Columns

```
SELECT department_id, location_id  
FROM departments;
```

DEPARTMENT_ID	LOCATION_ID
10	1700
20	1800
50	1500
60	1400
80	2500
90	1700
110	1700
190	1700

8 rows selected.

Arithmetic Expressions

Create expressions with number and date data by using arithmetic operators.

Operator	Description
+	Add
-	Subtract
*	Multiply
/	Divide

Using Arithmetic Operators

```
SELECT last_name, salary, salary + 300  
FROM employees;
```

LAST_NAME	SALARY	SALARY+300
King	24000	24300
Kochhar	17000	17300
De Haan	17000	17300
Hunold	9000	9300
Ernst	6000	6300

...
20 rows selected.

WHERE Clause

- The WHERE clause is used to filter records.
- **Note:** The WHERE clause is not only used in SELECT statements, it is also used in UPDATE, DELETE, etc.!

- Syntax:

```
SELECT column1, column2, ...  
FROM table_name  
WHERE condition;
```

```
SELECT employee_id, last_name, job_id, department_id  
FROM employees  
WHERE department_id = 90 ;
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
100	King	AD_PRES	90
101	Kochhar	AD_VP	90
102	De Haan	AD_VP	90

AND, OR and NOT Operators

- The WHERE clause can be combined with AND, OR, and NOT operators.

Operator	Description	Syntax	Example
AND	Returns TRUE if both conditions are true.	<pre>SELECT * FROM table_name WHERE condition1 AND condition2;</pre>	<pre>SELECT * FROM Employees WHERE Department = 'HR' AND Salary > 50000;</pre>
OR	Returns TRUE if at least one condition is true.	<pre>SELECT * FROM table_name WHERE condition1 OR condition2;</pre>	<pre>SELECT * FROM Employees WHERE Department = 'HR' OR Department = 'Finance';</pre>
NOT	Returns TRUE if the condition is false.	<pre>SELECT * FROM table_name WHERE NOT condition;</pre>	<pre>SELECT * FROM Employees WHERE NOT Department = 'HR';</pre>
AND with OR	Used when multiple conditions are needed with different logical constraints.	<pre>SELECT * FROM table_name WHERE (condition1 AND condition2) OR condition3;</pre>	<pre>SELECT * FROM Employees WHERE (Department = 'HR' AND Salary > 50000) OR Department = 'Finance';</pre>

```
SELECT column1, column2, ...  
FROM table_name  
WHERE condition1 AND condition2 AND condition3 ...;
```

```
SELECT column1, column2, ...  
FROM table_name  
WHERE condition1 OR condition2 OR condition3 ...;
```

```
SELECT column1, column2, ...  
FROM table_name  
WHERE NOT condition;
```

```
SELECT * FROM Customers  
WHERE Country = 'Germany' AND (City = 'Berlin' OR City =  
'Stuttgart');
```


Command	Description	Syntax	Example
SELECT	Retrieves data from one or more tables.	SELECT column1, column2 FROM table_name;	SELECT Name, Age FROM Students;
WHERE	Filters records based on a condition.	SELECT column1 FROM table_name WHERE condition;	SELECT * FROM Students WHERE Age > 20;
ORDER BY	Sorts results in ascending or descending order.	SELECT column1 FROM table_name ORDER BY column1 [ASC DESC];	SELECT Name FROM Students ORDER BY Age DESC;
GROUP BY	Groups records based on a column.	SELECT column1, COUNT(*) FROM table_name GROUP BY column1;	SELECT Department, COUNT(*) FROM Students GROUP BY Department;
HAVING	Filters grouped records.	SELECT column1, COUNT(*) FROM table_name GROUP BY column1 HAVING condition;	SELECT Department, COUNT(*) FROM Students GROUP BY Department HAVING COUNT(*) > 5;
LIMIT	Limits the number of rows returned.	SELECT column1 FROM table_name LIMIT number;	SELECT * FROM Students LIMIT 5;
DISTINCT	Removes duplicate values.	SELECT DISTINCT column1 FROM table_name;	SELECT DISTINCT Department FROM Students;
JOIN	Combines data from multiple tables.	SELECT column1 FROM table1 JOIN table2 ON table1.col = table2.col;	SELECT Students.Name, Courses.CourseName FROM Students INNER JOIN Courses ON Students.RollNo = Courses.RollNo;



Tasks 1

- Write 3 MySQL statements each of the following:
- **SELECT**
- **WHERE**
- **ORDER BY**
- **GROUP BY**
- **HAVING**
- **LIMIT**
- **DISTINCT**
- **JOIN**