

# MySQL Appunti

---

## SELECT Syntax

---

Select di alcune colonne (*column1*, *column2*)

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

Select di tutte le colonne di una tabella

```
SELECT *
FROM table_name;
```

## SELECT WHERE Syntax

---

Select di alcune colonne (*column1*, *column2*) con una condizione

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

Select di tutte le colonne di una tabella

```
SELECT *
FROM table_name;
WHERE condition;
```

Le *condition* sono fatte da:

- Uguaglianza:

`column = valore`

- Maggiore e minore

`column > valore`

`column < valore`

- Tra 2 valori

`column BETWEEN valore1 AND valore2`

- Non è nullo

`IS NOT NULL`

- Operatori:

- AND
- OR
- NOT per negare

## INSERT INTO Syntax

---

Inserimento specificando le colonne in cui andranno inseriti i valori

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

Inserimento di tutti i valori di tutte le colonne della tabella

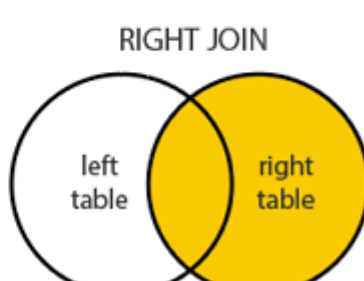
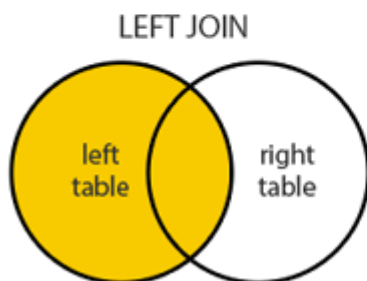
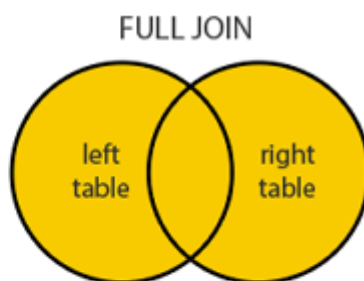
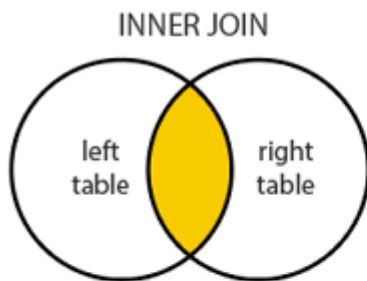
```
INSERT INTO table_name  
VALUES (value1, value2, value3, ...);
```

Inserimento con il primo valore lasciato a default per il db

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

## JOIN Syntax

---



## INNER JOIN

```
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name;
```

## LEFT JOIN

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name = table2.column_name;
```

## RIGHT JOIN

```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name = table2.column_name;
```

## FULL JOIN

```
SELECT column_name(s)
FROM table1
CROSS JOIN table2;
```

# CREATE Syntax

---

## Creare un DB

```
CREATE DATABASE databasename;
```

## Creare una tabella

```
CREATE TABLE table_name (  
    column1 datatype constraint,  
    column2 datatype constraint,  
    column3 datatype constraint,  
    CONSTRAINT nomeCostraint UNIQUE (column1, column2)  
);
```

Constraint inseribili nella stessa linea:

- 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

## AUTO\_INCREMENT

```
CREATE TABLE Persons (  
    Personid int NOT NULL AUTO_INCREMENT,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255),  
    Age int,  
    PRIMARY KEY (Personid)  
);
```

## PRIMARY KEY

Primary key semplice

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

```
    FirstName varchar(255),  
    Age int,  
    PRIMARY KEY (ID)  
);
```

Primary key composita

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

## FOREIGN KEY

Foreign key semplice

```
CREATE TABLE Orders (  
    OrderID int NOT NULL,  
    OrderNumber int NOT NULL,  
    PersonID int,  
    PRIMARY KEY (OrderID),  
    FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)  
);
```

Primary key composita

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

## MySQL Types

---

### DATE types

- DATE - format YYYY-MM-DD
- DATETIME - format: YYYY-MM-DD HH:MI:SS
- TIMESTAMP - format: YYYY-MM-DD HH:MI:SS
- YEAR - format YYYY or YY

## NORMAL types

- `CHAR(size)` A FIXED length string (can contain letters, numbers, and special characters). The size 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 size 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 size parameter specifies the column length in bytes. Default is 1
- `VARBINARY(size)` Equal to `VARCHAR()` , but stores binary byte strings. The size 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