SQL

```
CREATE DATABASE databasename; -- crea il database

DROP DATABASE databasename;   -- cancella il database


--sintassi per la creazione delle tabelle

CREATE TABLE table_name (

    column0 datatype NOT NULL AUTO_INCREMENT PRIMARY KEY,
    column1 datatype,
    column2 datatype,
    column3 datatype,
   ....

    FOREIGN KEY (column3) REFERENCES other_table(column3)
);


--sintassi alternativa per la creazione delle tabelle

CREATE TABLE table_name (

    column0 datatype AUTO_INCREMENT,
    column1 datatype,
    column2 datatype,
    column3 datatype,
   ....

   CONSTRAINT PK PRIMARY KEY (column0, column1),

   FOREIGN KEY (column3) REFERENCES other_table(column3)
);


CREATE TABLE table_name (

    column0 datatype AUTO_INCREMENT,
    column1 datatype,
    column2 datatype,
    column3 datatype,
   ....

   CONSTRAINT PK PRIMARY KEY (column0, column1),

   CONSTRAINT FK FOREIGN KEY (column3) REFERENCES other_table(column3)
);
```

```
--sintassi per la modifica delle tabelle
      -- aggiungere un attributo, un vincolo, un un indice
      ALTER TABLE table_name ADD column datatype;
      -- modificare un attributo (nome etipo di dato)
      ALTER TABLE table_name CHANGE columnoOld columnNew datatype;
      -- eliminare un attributo, un vincolo, un un indice
      ALTER TABLE table_name DROP column;


--sintassi per la cancellazione delle tabelle
DROP TABLE table_name; -- cancella la tabella
--sintassi per la inserimento, modifica, cancellazione dei dati
INSERT INTO table_name (column0, column1,…,columnN) VALUES (value1,
value2,…,valueN); -- inserire valori


INSERT INTO table_name VALUES (value1, value2,…,valueN),...,(value11,
value12,…,value1N); -- inserire valori multipli


INSERT INTO table_name SET column0=value1, column1=value1,…,columN=valueN; ---
inserire valori


UPDATE table_name SET columnH=valueH WHERE columnJ=valueJ AND/OR
columnI=valueI; -- aggiornare valori


DELETE FROM table_name WHERE columnJ=valueJ AND/OR columnI=valueI; --
cancellare valori
```

MySQL Data Types (Version 8.0)

String Data Types

CHAR(size)

VARCHAR(size)

BINARY(size)

VARBINARY(size)

TINYBLOB

TINYTEXT

TEXT(size)

BLOB(size)

MEDIUMTEXT

MEDIUMBLOB

LONGTEXT

LONGBLOB

ENUM(val1, val2, val3, ...)

SET(val1, val2, val3, ...)

Numeric Data Types

BIT(size)

TINYINT(size)

BOOL

BOOLEAN

SMALLINT(size)

MEDIUMINT(size)

INT(size)

INTEGER(size)

BIGINT(size)

FLOAT(size, d)

FLOAT(p)

DOUBLE(size, d)

DOUBLE PRECISION(size, d)

DECIMAL(size, d)

DEC(size, d)

Date and Time Data Types

DATE

DATETIME(fsp)

TIMESTAMP(fsp)

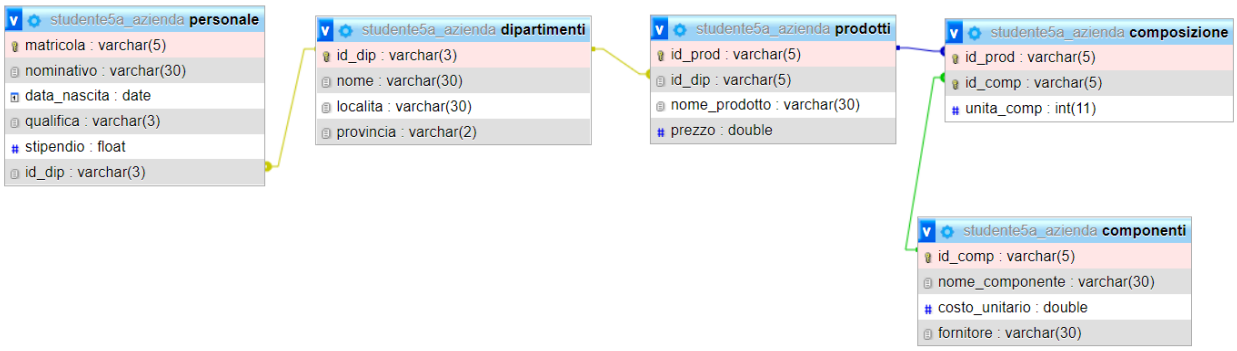
TIME(fsp)

YEAR

--sintassi ordine di esecuzione delle clausole:

```
SELECT      (SELECT table0.* significa tutti i campi della tabella 'table0')
FROM
WHERE
GROUP BY
HAVING
ORDER BY
LIMIT
```

Gli esempi presenti si riferiscono al seguente db-schema che rappresenta un'azienda:



--sintassi per l'interrogazione delle tabelle: (SELECT-FROM-WHERE)

```
SELECT  table0.column0, table0.column1, …….
FROM    table0
WHERE   <condition1>                    /*condition 1*/
AND     <condition2>                    /*condition 1*/
```

--sintassi per l'interrogazione delle tabelle con dati non ripetuti (DISTINCT)

```
SELECT  DISTINCT   table0.column0, table0.column1, …….
FROM    table0
WHERE   <condition1>                    /*condition 1*/
AND     <condition2>                    /*condition 1*/
```

--ALIAS per modificare il nome di campi e/o tabelle

```
SELECT  T0.column0 AS attributo0 , T0.column1 AS attributo1
FROM    table0 AS T0
WHERE   <condition>                    /*condition */
```

--sintassi per campi calcolati. Operazioni algebriche possibili fra colonne diverse: somma(+), sottrazione(-), moltiplicazione (*), divisione(/)

```
SELECT  table0.column0, table0.column1, taable0.column0-table0.column1 AS differenza
FROM    table0
WHERE   <condition>;                   /*condition */
```

Esempio:

Calcolo di un aumento del 10% per il personale del dipartimento d1:

```
SELECT matricola, nominativo, stipendio, stipendio*0.1 as aumento
FROM personale
WHERE id_dip='d1';
```

--sintassi per l'interrogazione delle tabelle con JOIN e ordinamento: (SELECT-FROM-WHERE-ORDER BY)

```
SELECT  table0.column0, table0.column1, table1.column0,…
FROM    table0,table1
WHERE   table0.column0=table1.column0                  /*join*/
/*          PK              FK
FROM    table0 INNER JOIN table1 ON table0.column0=table1.column0 /*join*/
*/
AND     <condition>                                    /*selection*/
ORDER BY <table0.coulmn0,table1.column0> [ASC|DESC];
```

--nella <condition> si possono utilizzare: AND, OR, IN, NOT IN, LIKE, BETWEEN, ALL, ANY >,<,>=,<=,=,<>,!=

--LIKE viene utilizzato nelle operazioni di pattern-matching fra stringhe e consente l'impiego dei seguenti caratteri jolly:

% → qualsiasi stringa

_ → qualsiasi carattere (un solo carattere)

Esempio:

Matricola e nominativo di tutto il personale il cui nominativo inizia con la lettera 'P':

```
SELECT matricola, nominativo
FROM personale
WHERE nominativo LIKE 'P%';
```

Esempio:

Elenco dei prodotti (nome) il cui prezzo è compreso fra 5 e 10 € (compresi gli estremi):

```
SELECT nome_prodotto
FROM prodotti
WHERE prezzo  BETWEEN  5 AND 10;
```

Esempio:

Elenco dei prodotti (nome) il cui prezzo è inferiore al costo unitario di TUTTI componenti

```
SELECT nome_prodotto
FROM prodotti
WHERE prezzo< ALL(SELECT costo_unitario FROM componenti);
```

Esempio:

Elenco dei prodotti (nome) il cui prezzo è inferiore al costo unitario di ALMENO un componente

```
SELECT nome_prodotto
FROM prodotti
WHERE prezzo< ANY(SELECT costo_unitario FROM componenti);
```

--funzioni per operazioni sulle date

YEAR *(data) con data di tipo date restituisce un intero corrispondente all'anno*

MONTH *(data) con data di tipo date restituisce un intero corrispondente al mese (da 1 a 12)*

DAY *(data) con data di tipo date restituisce un intero corrispondente al giorno (da 1 a 31)*

TIMESTAMPDIFF **(**YEAR**,** *data1***,** *data2***)** *fornisce la differenza in anni (oppure in mesi o giorni, minuti, secondi, ore specificando* MONTH, DAY, HOUR, MINUTE, SECOND *come primo parametro) fra data2 e data1.*

CURRENT_DATE *restituisce data attuale*

CURRENT_TIME *restituisce ora attuale*

CURRENT_TIMESTAMP *restituisce data e ora attuale*

Esempio:

mostrare matricola, nominativo ed età di tutto il personale:

SELECT *matricola, nominativo,* TIMESTAMPDIFF*(*YEAR*,data_nascita,*CURRENT_TIMESTAMP*)* AS *eta*

FROM *personale;*

```
--sintassi per OUTER JOIN: LEFT JOIN  (mantiene tutte le righe della tabella table0)

SELECT  table0.column0, table0.column1, table1.column0,…
FROM    table0 LEFT JOIN table1 ON table0.column0=table1.column0
WHERE   table0.column0=table1.column0                    /*join*/
AND     <condition>;                                     /*selection*/

--sintassi per OUTER JOIN: RIGHT JOIN (mantiene tutte le righe della tabella table1)

SELECT  table0.column0, table0.column1, table1.column0,…
FROM    table0 RIGHT JOIN table1 ON table 0.column0=table1.column0
WHERE   table0.column0=table1.column0                    /*join*/
AND     <condition>;

--sintassi per limitare i risultati (LIMIT)

SELECT  *
FROM    table0
WHERE   <condition>                                      /*selection*/
ORDER BY <table0.coulmn0> [ASC|DESC]
LIMIT 3;                              /*restituisce solo i primi tre risultati*/
```

```
Esempio:
SELECT *
FROM personale
ORDER BY nominativo
LIMIT 2;
```

--sintassi per l'interrogazione delle tabelle con raggruppamento (GROUP BY), funzione di aggregazione (COUNT()) e ordinamento (ORDER BY)

```
SELECT   table0.column0, COUNT(*) AS row_number_group
FROM     table0,table1…
WHERE    table0.column1=table1.column1                    /*join*/
AND      <condition>                                      /*selection*/
GROUP BY table0.column0
ORDER BY <table0.coulmn0> [ASC|DESC];
```

--funzioni di aggregazione:

MAX(*coulmn0*)

MIN(*coulmn0*)

AVG(*coulmn0*)

COUNT(*coulmn0*)

SUM(*coulmn0*)

Esempio:

mostrare quante persone lavorano in ciascun dipartimento

```
SELECT id_dip, count(*) AS numero_dipendenti,
FROM personale
GROUP BY id_dip;
```

--sintassi per l'interrogazione delle tabelle con selezione sui raggruppamenti (HAVING)

```
SELECT   table0.column0, FUNZIONE DI AGGREGAZIONE(table0.column0) AS alias_name
FROM     table0, table1…
WHERE    table0.column1=table1.column1                    /*join*/
AND      <condition>                                      /*selection*/
GROUP BY table0.column0
HAVING   <condition>                                  /*group selection*/
```

Esempio:

mostrare quante persone lavorano in ciascun dipartimento ma solo per i dipartimenti in cui lavorano più di 3 persone

```
SELECT id_dip, count(*) AS numero_dipendenti
FROM personale
GROUP BY id_dip;
HAVING numero_dipendenti>3;
```

--subquery: le subquery possono essere utilizzate nel WHERE, nel FROM e nel HAVING

1. Subquery nel WHERE

```
SELECT  table0.column0,
FROM    table0
WHERE   table0.column1 IN
            (SELECT  table1.column2,
            FROM    table1
            WHERE   <condition>);
```

Esempio: *personale che lavora in provincia di Brescia*

```
SELECT personale.matricola, personale.nominativo
FROM personale
WHERE id_dip IN
     (SELECT id_dip
    FROM dipartimenti
    WHERE provincia='BS');
```

2. Subquery nel FROM

```
SELECT  table0.column0, T.column0
FROM    table0, (SELECT table1.column0, table1.column1 FROM table1
WHERE <condition>) AS T
WHERE <condition>;
```

Esempio: *personale che lavora in provincia di Brescia*

```
SELECT personale.matricola, personale.nominativo
FROM personale, (SELECT * FROM dipartimenti WHERE provincia='BS') AS T
WHERE personale.id_dip=T.id_dip;
```

3. Subquery nel HAVING

```
SELECT   table0.column0
FROM     table0
GROUP BY table0.column0
HAVING <condition with subquery>;
```

Esempio: *visualizzare i dipartimenti in cui lo stipendio medio del dipartimento è maggiore rispetto allo stipendio medio complessivo dell'azienda*

```
SELECT personale.id_dip, AVG(stipendio) as
stipendio_medio_per_dipartimento
FROM personale
GROUP BY personale.id_dip
HAVING stipendio_medio_per_dipartimento>
            (SELECT avg(stipendio) as stipendio_medio FROM personale);
```

PHP + MySQL + HTML

Connessione al database MySql

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = new mysqli($servername, $username,
$password);

// Check connection
if ($conn->connect_error) {
 die("Connection failed: " . $conn-
>connect_error);
}
echo "Connected successfully";
?>
```

Creazione Database Php Mysql

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = new mysqli($servername, $username,
$password);
// Check connection
if ($conn->connect_error) {
 die("Connection failed: " . $conn-
>connect_error);
}

// Create database
$sql = "CREATE DATABASE myDB";
if ($conn->query($sql) === TRUE) {
 echo "Database created
successfully";
} else {
 echo "Error creating database: " .
$conn->error;
}

$conn->close();
?>
```

Creazione Tabelle

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username,
$password, $dbname);
// Check connection
if ($conn->connect_error) {
 die("Connection failed: " . $conn-
>connect_error);
}

// sql to create table
$sql = "CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT
PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP DEFAULT
CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
)";

if ($conn->query($sql) === TRUE) {
 echo "Table MyGuests created
successfully";
} else {
 echo "Error creating table: " . $conn-
>error;
}

$conn->close();
?>
```

Inserimento dati

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
 die("Connection failed: " . $conn->connect_error);
}

$sql = "INSERT INTO MyGuests (firstname, lastname, email) VALUES ('John', 'Doe', 'john@example.com')";

if ($conn->query($sql) === TRUE) {
 echo "New record created successfully";
} else {
 echo "Error: " . $sql . "<br>" . $conn->error;
}

$conn->close();
?>
```

Inserire dati multipli

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
 die("Connection failed: " . $conn->connect_error);
}

$sql = "INSERT INTO MyGuests (firstname, lastname, email) VALUES ('John', 'Doe', 'john@example.com');";
$sql .= "INSERT INTO MyGuests (firstname, lastname, email) VALUES ('Mary', 'Moe', 'mary@example.com');";
$sql .= "INSERT INTO MyGuests (firstname, lastname, email) VALUES ('Julie', 'Dooley', 'julie@example.com')";

if ($conn->multi_query($sql) === TRUE) {
 echo "New records created successfully";
} else {
 echo "Error: " . $sql . "<br>" . $conn->error;
}

$conn->close();
?>
```

Query semplice

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
 die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
 // output data of each row
 while($row = $result->fetch_assoc()) {
  echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>";
 }
} else {
 echo "0 results";
}
$conn->close();
?>
```

Cancella Dati

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
 die("Connection failed: " . $conn->connect_error);
}

// sql to delete a record
$sql = "DELETE FROM MyGuests WHERE id=3";

if ($conn->query($sql) === TRUE) {
 echo "Record deleted successfully";
} else {
 echo "Error deleting record: " . $conn->error;
}

$conn->close();
?>
```

Aggiorna dati

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
 die("Connection failed: " . $conn->connect_error);
}

$sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";

if ($conn->query($sql) === TRUE) {
 echo "Record updated successfully";
} else {
 echo "Error updating record: " . $conn->error;
}

$conn->close();
?>
```

FORM HTML E RECUPERO DATI CON PHP

```
<form method="post" action="esegui.php">
<!-- CASELLE DI TESTO -->

Nome<br>
<input type="text"name="nome">
<br>Cognome<br>
<input type="text"name="cognome"><br>


<!-- SELECTBOX -->

Paese<br>
<select name="paese"><option value="I">Italia</option>
<option value="E">Estero</option></select><br>


<!-- RADIO -->

Sesso<br>
<input type="radio"name="sesso"value="M">M<br>
<input type="radio"name="sesso"value="F">F<br>


<!-- CHECKBOX -->

Hobby<br>
<input type="checkbox"name="hobby"value="S">Sport<br>
<input type="checkbox"name="hobby"value="L">Lettura<br>
<input type="checkbox"name="hobby"value="C">Cinema<br>
<input type="checkbox"name="hobby"value="I">Internet<br>


<!-- TEXTAREA -->

Commento<br><textareaname="commento" rows="5"cols="30">
</textarea><br><br>


<!-- SUBMIT -->

<button type="submit">Invia i dati</button>
</form>


// recupero il valore del parametro 'campo1' trasmesso mediante query-string
(metodo GET)

$campo1=$_GET['campo1'];

// recupero il valore del parametro 'campo1' trasmesso mediante il metodo POST

$campo1=$_POST['campo1'];
```