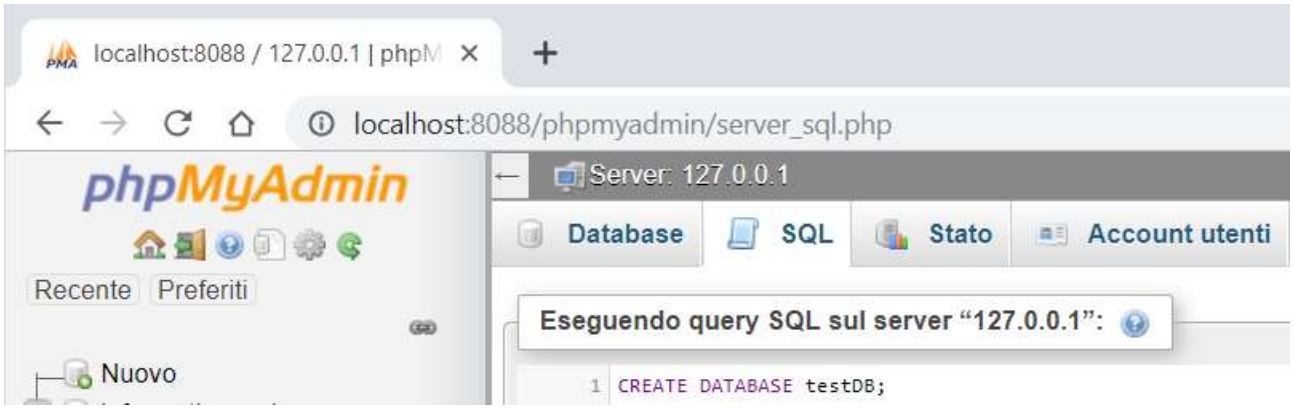
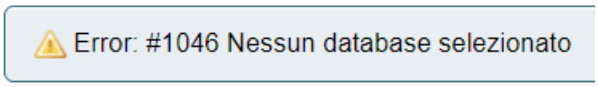
Esercitiamoci[1] con uso MySQL (simulando in locale con XAMPP)

- **Creare DB (esempio da w3schools)**



Al click su 

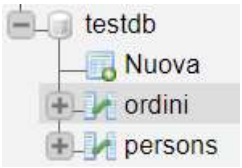


Crea il DATABASE (la directory) pur con Warning che richiederebbe di specificare i privilegi

Dal manuale online si noti la sintassi[2] (senza specifiche vengono usate quelle di default):

```
CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] db_name
    [create_specification] ...

create_specification:
    [DEFAULT] CHARACTER SET [=] charset_name
  | [DEFAULT] COLLATE [=] collation_name
```



- **Creare / popolare / eliminare tabelle**
- **Uso di funzioni: SUM, COUNT, AVG, MAX, MIN**

Creazione tabelle[3] senza PK:

CREATE TABLE Persons (PersonID int, LastName varchar(255), FirstName varchar(255), Address varchar(255), City varchar(255))	CREATE TABLE Ordini (id int, id_cliente int, prezzo int, prodotto varchar(255))

Popoliamo tabella Persons

INSERT INTO Persons VALUES
(1,'Perla','Giada', 'via Giotto 7', 'Milano')

Select * FROM Persons

PersonID	LastName	FirstName	Address	City
1	Perla	Giada	via Giotto 7	Milano

[1] Esercizi tratti da **http://www.mrwebmaster.it/sql/group-by-having_11991.html**
[2] Interessante con uso MySQL Workbench http://www.mysqltutorial.org/mysql-create-database/
[3] ***Usare Ctrl-V per incollare***

Cancellando **DROP TABLE Persons** **meglio** DROP Table **IF EXISTS** [*nomeTab*]
(Con richiesta di conferma per la **rimozione della struttura**)

Reinserendo ed aggiungendo una seconda tupla

INSERT INTO Persons VALUES (1,'Perla','Giada', 'via Giotto 7', 'Milano');
INSERT INTO Persons VALUES (2,'Dori','Marco', 'via Pisa 6', 'Genova')

Select * FROM Persons

PersonID	LastName	FirstName	Address	City
1	Perla	Giada	via Giotto 7	Milano
2	Dori	Marco	via Pisa 6	Genova

Aggiorniamo

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

UPDATE Persons SET LastName='Perlo' WHERE PersonID=1

Select * FROM Persons

PersonID	LastName	FirstName	Address	City
1	Perlo	Giada	via Giotto 7	Milano
2	Dori	Marco	via Pisa 6	Genova

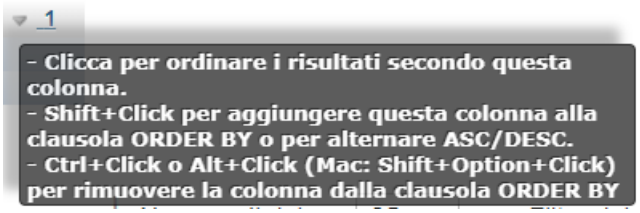
Equivalenza:

SELECT LastName FROM Persons **GROUP BY** LastName ORDER BY LastName ASC;	**SELECT DISTINCT** LastName FROM Persons ORDER BY LastName ASC;

LastName ▲ 1
Dori
Perlo

Nella sua forma di utilizzo più semplice la clausola ***GROUP BY*** produce un risultato analogo a **SELECT DISTINCT**.

Possibilità offerte da phpMyAdmin



Popoliamo la tabella ORDINI

INSERT INTO Ordini VALUES (1, 1, 50, 'Scarpe');
INSERT INTO Ordini VALUES (2, 1,1500, 'Vestito');
INSERT INTO Ordini VALUES (3, 2, 40, 'Scarpe');
INSERT INTO Ordini VALUES (4, 2, 500, 'Vestito');
INSERT INTO Ordini VALUES (5, 1, 60, 'Camicia');
INSERT INTO Ordini VALUES (6, 1,70, 'Cravatta');

select * from Ordini

id	id_cliente	prezzo	prodotto
1	1	50	Scarpe
2	1	1500	Vestito
3	2	40	Scarpe
4	2	500	Vestito
5	1	60	Camicia
6	1	70	Cravatta

Quindi:
1 Perlo scarpe 50 + vestito 1500 + camicia 60 + cravatta 70
2 Dori scarpe 40 + vestito 500

Nb:
al click su SQL 

se selezionata la tabella ordini:

Equivalentemente interpretati:

```
SELECT id_cliente, SUM[4](prezzo) AS spesa
FROM ordini
GROUP BY[5] id_cliente
HAVING spesa >= 1000;
```

id_cliente	spesa
1	1680

SELECT id_cliente, SUM(prezzo) AS spesa
FROM ordini
GROUP BY id_cliente
Having SUM(prezzo)>=1000

```
SELECT id_cliente, SUM(id) AS acquisti, SUM(prezzo) AS spesa
FROM ordini
GROUP BY id_cliente
HAVING spesa >= 1000 OR acquisti >= 2;
```

id_cliente	acquisti	spesa
1	14	1680
2	7	540

SELECT id_cliente, Count(id) AS acquisti, SUM(prezzo) AS spesa
FROM ordini
GROUP BY id_cliente
HAVING SUM(prezzo) >= 1000 OR SUM(id) >= 2;

La clausola Where (applicabile a tuple singole) non può essere usata con funzioni aggregate

Sintassi:

```
SELECT column_name(s)
FROM table_name
WHERE condition                    /* su singole tuple */
GROUP BY column_name(s)
HAVING condition                   /* su gruppi */
ORDER BY column_name(s);
```

In alcuni casi può essere necessario escludere singole tuple dai gruppi (utilizzando una clausola **WHERE**) prima di applicare una **condizione ai gruppi** (utilizzando una clausola HAVING).

La clausola **HAVING** può essere applicata solo alle **colonne presenti anche nella clausola GROUP BY** o in una **funzione di aggregazione**

SELECT LastName, Count(id) AS acquisti, SUM(prezzo) AS spesa
FROM ordini, Persons
WHERE id_cliente = PersonID
GROUP BY LastName
HAVING (SUM(prezzo) >= 1000) OR (Count(id) >= 2);

LastName	acquisti	spesa
Dori	2	540
Perlo	4	1680

```
SELECT id_cliente, MAX(prezzo) AS spesa_massima
FROM ordini
GROUP BY id_cliente;
```

id_cliente	spesa_massima
1	1500
2	500

SELECT LastName, MAX(prezzo) AS spesa_massima
FROM ordini, Persons
Where id_cliente = PersonID
GROUP BY LastName

LastName	spesa_massima
Dori	500
Perlo	1500

Nb: per query parametriche SET @a= 'Dori';

SELECT * FROM Persons Where LastName = @a

PersonID	LastName	FirstName	Address	City
2	Dori	Marco	via Pisa 6	Genova

[4] ***Attenzione: non inserire spazi tra il nome della funzione e le parentesi tonde***

[5] **NB:** si ricordi che tutti i campi che compaiono accanto alla parola SELECT devono essere inclusi nella clausola **GROUP BY** oppure devono essere argomenti di una funzione di aggregazione

```
SELECT LastName, MIN(prezzo) AS spesa_minima
FROM ordini, Persons
Where id_cliente = PersonID
GROUP BY LastName
```

LastName	spesa_minima
Dori	40
Perlo	50

```
SELECT LastName, AVG(prezzo) AS spesa_media
FROM ordini, Persons
Where id_cliente = PersonID
GROUP BY LastName
```

LastName	spesa_media
Dori	270.0000
Perlo	420.0000

```
SELECT LastName, AVG(prezzo) AS spesa_media
FROM ordini, Persons
Where id_cliente = PersonID
GROUP BY LastName
ORDER BY spesa_media DESC;
```

LastName	spesa_media
Perlo	420.0000
Dori	270.0000

Altri raggruppamenti ed ordinamenti

```
SELECT prodotto, prezzo
FROM ordini
GROUP BY prodotto, prezzo
ORDER BY prodotto ASC;
```

prodotto	prezzo
Camicia	60
Cravatta	70
Scarpe	40
Scarpe	50
Vestito	500
Vestito	1500

```
SELECT prodotto, prezzo
FROM ordini
GROUP BY prodotto, prezzo
ORDER BY prezzo DESC;
```

prodotto	prezzo
Vestito	1500
Vestito	500
Cravatta	70
Camicia	60
Scarpe	50
Scarpe	40

```
SELECT prodotto, prezzo
FROM ordini
GROUP BY prodotto, prezzo
ORDER BY prezzo ASC;
```

prodotto	prezzo
Scarpe	40
Scarpe	50
Camicia	60
Cravatta	70
Vestito	500
Vestito	1500

Trovare "il MAX di un conteggio eseguito con uso della funzione COUNT(*)" che in MySQL come in SQL Server non si può fare direttamente scrivendo MAX(COUNT(*))

```
SELECT prodotto, Count(*) AS num FROM ordini Group by prodotto
```

prodotto	num
Camicia	1
Cravatta	1
Scarpe	2
Vestito	2

```
SELECT MAX(y.num) AS massimo
FROM (SELECT prodotto, Count(*) AS num FROM ordini Group by prodotto) y
```

massimo
2

Oppure con alias esplicito della tabella creata al volo dalla select interna[6]:

```
SELECT MAX(y.num) AS massimo
FROM (SELECT prodotto, Count(*) AS num FROM ordini Group by prodotto) AS y
```

[6] Tale subquery che definisce "dinamicamente" una tabella derivata può essere usata anche all'interno della clausola FROM (*table expression* online pg. 13). Altra soluzione con *view* al problema (pg. 6-7).

Equivalente è creare esplicitamente una tabella temporanea ed usarla

```
CREATE Table temp
AS SELECT prodotto, Count(*) AS num
FROM ordini
Group by prodotto;

Select MAX(num) AS massimo
FROM temp
```

CREATE Table temp
AS SELECT prodotto, Count(*) AS num
FROM ordini
Group by prodotto;

Select MAX(num) AS massimo
FROM temp

Dato il seguente schema logico:

biciclette (id, statobici, idUtente) id PK, idUtente FK dove statobici = 1 se è in uso

operazioni (id, dataora, tipoop, idUtente, idBicicletta, idStazione) id PK e idUtente, idBicicletta, idStazione FK

utenti (id, nomeutente, sesso, datanascita, indirizzo, comune, provincia, telefoneo, email, numerocard, saldocard) id PK

stazioni (id, nomestazione, ondirizzo, longitudine, latitudine, slotliberi) id PK

Proporre query:

a. dato il codice di una bicicletta elencare gli utenti che l'hanno utilizzata nel mese corrente
b. mostrare la stazione presso la quale è stato effettuato il maggior numero di noleggi in un dato periodo.

a) Supponendo che il codice della bicicletta sia id=11

```
SELECT distinct nomeutente
FROM operazioni, utenti
WHERE idutente=utenti.id
AND idbicicletta=11 AND month(dataora)=month(curdate())
```

b) è opportuno **creare una tabella temporanea** che contiene per ciascuna stazione il numero di noleggi effettuati (operazioni con tipoop=0) in un dato periodo (ad esempio nei primi due mesi del 2019)

```
CREATE TABLE temp
AS SELECT count(*) AS 'noleggi' ,idstazione
FROM operazioni
WHERE tipoop=0
AND date(dataora) BETWEEN '2019/01/01' AND '2019/02/28'
GROUP by idstazione
```

dalla tabella 'temp' che contiene il numero di noleggi per ciascuna stazione, si seleziona la stazione (o si selezionano le stazioni) con il massimo numero di noleggi tramite la query:

```
SELECT idstazione, noleggi
FROM temp
WHERE noleggi=(SELECT MAX(noleggi) FROM temp)
```

Da soluzione proposta dal prof. Mauro De Berardis - Teramo

Definizione di chiave primaria e chiave esterna

```
CREATE TABLE Persons
(
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255),
PRIMARY KEY (P_Id)
)
```

Crea la struttura

Struttura della tabella | Vista relazioni

	#	Nome	Tipo	Codifica caratteri	Attributi	Null	Predefinito
☐	1	**P_Id** 🔑	int(11)			No	*Nessuno*
☐	2	**LastName**	varchar(255)	latin1_swedish_ci		No	*Nessuno*
☐	3	**FirstName**	varchar(255)	latin1_swedish_ci		Sì	*NULL*
☐	4	**Address**	varchar(255)	latin1_swedish_ci		Sì	*NULL*
☐	5	**City**	varchar(255)	latin1_swedish_ci		Sì	*NULL*

```
CREATE TABLE Orders
(
 id int(11) NOT NULL AUTO_INCREMENT,
 id_cliente int(11) NOT NULL,
 prezzo int,
 prodotto varchar(255),
 PRIMARY KEY (id),
 FOREIGN KEY (id_cliente) REFERENCES
Persons(P_Id)
)
```

Per popolare si esplicitano i campi (la PK è inserita automaticamente):

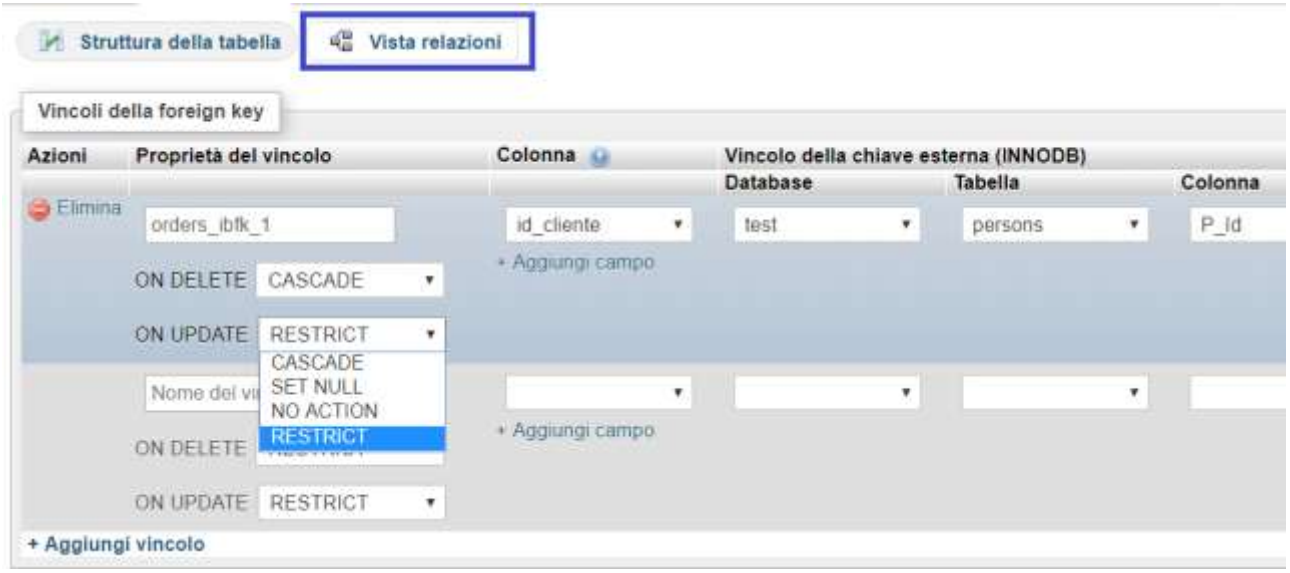
```
INSERT INTO Orders (id_cliente, prezzo, prodotto) VALUES (1, 50, 'Scarpe');
INSERT INTO Orders (id_cliente, prezzo, prodotto)  VALUES (1,1500, 'Vestito');
INSERT INTO Orders  (id_cliente, prezzo, prodotto) VALUES (2, 40, 'Scarpe');
INSERT INTO Orders  (id_cliente, prezzo, prodotto) VALUES (2, 500, 'Vestito');
INSERT INTO Orders  (id_cliente, prezzo, prodotto) VALUES (1, 60, 'Camicia');
INSERT INTO Orders  (id_cliente, prezzo, prodotto) VALUES (1,70, 'Cravatta');
```

Crea la struttura con PK contatore numerico auto-incrementante

Struttura della tabella | Vista relazioni

	#	Nome	Tipo	Codifica caratteri	Attributi	Null	Predefinito	Commenti	Extra
☐	1	**id** 🔑	int(11)			No	*Nessuno*		AUTO_INCREMENT
☐	2	**id_cliente** 🔗	int(11)			No	*Nessuno*		
☐	3	**prezzo**	int(11)			Sì	*NULL*		
☐	4	**prodotto**	varchar(255)	latin1_swedish_ci		Sì	*NULL*		

Di default: Restrict come vincolo di integrità referenziale … la modifica corrisponde ad ALTER TABLE




```
ALTER TABLE `orders` DROP FOREIGN KEY `orders_ibfk_1`; ALTER TABLE `orders` ADD CONSTRAINT `orders_ibfk_1` FOREIGN KEY (`id_cliente`) REFERENCES`persons`(`P_Id`) ON DELETE CASCADE ON UPDATE CASCADE;
```

Esercitarsi con le funzioni di MySQL

Il formato data[7]

Per quanto riguarda ***MySQL*** la soluzione è drastica quanto efficiente: l'unico formato supportato è quello **ISO**, ovvero la struttura **yyyy/mm/dd** dove lo / può essere comunemente rimpiazzato con un -.

Vediamo un semplice esempio legato ad un aggiornamento di un record:

UPDATE nometabella SET campodata = **'2005-09-24'**

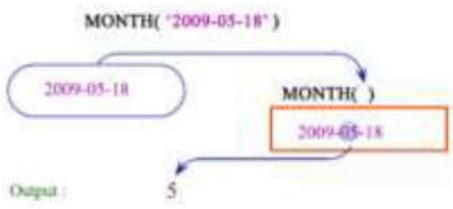
Oppure ricordiamo l'esempio di estrazione del mese corrente nel confronto tra DBMS:

Sintassi Month(*date*)

MySQL

SELECT MONTH(CURDATE());

altro es:



SELECT Prodotto, prezzo, **CURDATE()** as PerDate
FROM Ordini

Prodotto	prezzo	PerDate
Scarpe	50	2019-03-04
Vestito	1500	2019-03-04
Scarpe	40	2019-03-04
Vestito	500	2019-03-04
Camicia	60	2019-03-04
Cravatta	70	2019-03-04

SELECT **NOW();**
risultato 2019-03-04 21:35:40

SELECT Prodotto, prezzo, **DATE_FORMAT**(CURDATE(), "%M %d %Y") as PerDate
FROM Ordini

Prodotto	prezzo	PerDate
Scarpe	50	March 04 2019
Vestito	1500	March 04 2019
Scarpe	40	March 04 2019
Vestito	500	March 04 2019
Camicia	60	March 04 2019
Cravatta	70	March 04 2019

Sintassi:

DATE_FORMAT(*date, format*)

SELECT DATE_FORMAT("2017-06-15", "%M %d %Y");

risultato June 15 2017

Sintassi:

DATEDIFF(*date1, date2*)

SELECT DATEDIFF ("2017-01-01", "2016-12-24");
risultato 8

[7] Da http://new345.altervista.org/DB/SQL/SQL.pdf

Altro esempio online[8]

Data type

```
CREATE TABLE tabella
(
stringa CHAR(20), /* stringa di lunghezza massima 20 caratteri */
numero_int INT, /* numero intero */
numero_dec DECIMAL(5,2), /* numero con 3 cifre intere e 2 decimali:
                            5 indica il numero massimo di cifre intere + 2 byte per il segno
                            e per il punto (che funge da virgola) */
data DATE, /* data in formato AAAA-MM-GG */
testo TEXT /* testo lungo e di lunghezza variabile (fino a 65.000 caratteri) */
);
```

Crea la tabella con struttura:

Struttura della tabella | Relazione vista

	#	Nome	Tipo	Codifica caratteri	Attributi	Null	Predefinito
☐	1	**stringa**	char(20)			Sì	*NULL*
☐	2	**numero_int**	int(11)			Sì	*NULL*
☐	3	**numero_dec**	decimal(5,2)			Sì	*NULL*
☐	4	**data**	date			Sì	*NULL*
☐	5	**testo**	text			Sì	*NULL*

Popolando

```
INSERT INTO tabella
VALUES ('valore_stringa', 20, 200.55, '2019-03-05', 'ecco il testo');
```

Mostra

Pur con warning che avverte l'assenza di PK
(nell'uso del tasto Mostra che esegue una Select estraendo tutti i valori dei campi dalla tabella)

⚠ La selezione corrente non contiene un campo unico. Modifica griglia, checkbox, Modifica, Copia ed Elimina potrebbero non essere disponibili.

è avvenuto l'inserimento

stringa	numero_int	numero_dec	data	testo
valore_stringa	20	200.55	2019-03-05	ecco il testo

Con ulteriori inserimenti si nota l'errore che visualizza il massimo possibile 999.99 nel valore del numero_dec

```
INSERT INTO tabella
VALUES ('valore_stringa2', 2, 22.55, '2019-03-05', 'ecco il testo 2');
INSERT INTO tabella
VALUES ('valore_stringa3', 3, 2.5, '2019-03-05', 'ecco il testo 3');
INSERT INTO tabella
VALUES ('valore_stringa4', 4, 2222.555, '2019-03-05', 'ecco il testo 4');
```

stringa	numero_int	numero_dec	data	testo
valore_stringa	20	200.55	2019-03-05	ecco il testo
valore_stringa2	2	22.55	2019-03-05	ecco il testo 2
valore_stringa3	3	2.50	2019-03-05	ecco il testo 3
valore_stringa4	4	999.99	2019-03-05	ecco il testo 4

[8] Da http://www.corradodamiano.it/appunti/esempi_mysql.htm orientato all'uso di MySQL a linea di comando