



UNIVERSITÀ  
DEGLI STUDI  
DI PADOVA

DIPARTIMENTO DI MATEMATICA

LAUREA TRIENNALE IN INFORMATICA

BASI DI DATI - LABORATORIO 1

---

## Introduzione a PostgreSQL

---

Massimiliano de Leoni  
Alessandro Padella  
Samuel Cognolato

deleoni@math.unipd.it  
alessandro.padella@phd.unipd.it  
samuel.cognolato@studenti.unipd.it

# Indice

<b>1</b>	<b>Introduzione a pgAdmin4</b>	<b>3</b>
1.1	Creazione database . . . . .	7
1.2	Importazione database . . . . .	10
<b>2</b>	<b>Visualizzazione e manipolazione dei dati</b>	<b>12</b>
2.1	Query di base . . . . .	15
<b>3</b>	<b>Esercizi sulle query</b>	<b>17</b>
<b>4</b>	<b>Altri Esercizi</b>	<b>17</b>
<b>5</b>	<b>Soluzioni query</b>	<b>19</b>
<b>6</b>	<b>Soluzioni altri esercizi</b>	<b>20</b>

## Elenco delle figure

1	Homepage di pgAdmin4. . . . .	3
2	pgAdmin4 senza server connessi. . . . .	3
3	Accesso allo strumento di creazione di un nuovo server . . . . .	4
4	Impostazione del nome del server. . . . .	4
5	Impostazione dell'indirizzo del server. . . . .	5
6	Possibile errore in linux . . . . .	5
7	Corrispondenti righe del file dopo la modifica. . . . .	6
8	Inserimento della password. . . . .	6
9	Dashboard iniziale di pgAdmin4 . . . . .	7
10	Creazione di un nuovo database. . . . .	7
11	Inserimento del nome del database. . . . .	8
12	Visualizzazione del nuovo database. . . . .	8
13	Procedura per aprire il Query Tool. . . . .	9
14	Esecuzione di una query. . . . .	9
15	Importazione di un file <code>sql</code> nel Query Tool. . . . .	11
16	Selezione del file da importare. . . . .	11
17	Query eseguita con successo. . . . .	12
18	Schema del database. . . . .	12
19	Procedura per la visualizzazione dei dati di una tabella . . . . .	13
20	Interfaccia per la visualizzazione dei dati . . . . .	13
21	Procedura per aprire il Query Tool dalla barra degli strumenti. . . . .	14
22	Descrizione dei vari pannelli che compongono il Query Tool. . . . .	14
23	Esecuzione e visualizzazione dei risultati della prima query. . . . .	15
24	Tabella <b>LIBRO</b> completa. . . . .	21
25	Tabella <b>LIBRO</b> dopo la cancellazione degli autori con cognome uguale a ‘Rossi’. . . . .	22
26	Selezione di tutti i libri di Pirandello. . . . .	23
27	Tabella autore dopo l’aggiornamento del nome di “Pirandello”. . . . .	23
28	Caratteristiche del libro “Il fu Mattia Pascal” dopo l’aggiornamento dell’autore . . . . .	24

## 1 Introduzione a pgAdmin4

Una volta avviato il server, aprendo l'applicazione pgAdmin4, sarà possibile accedere all'interfaccia web da browser. La pagina si aprirà automaticamente sul browser predefinito. A questo punto avremmo accesso alla schermata principale di pgAdmin4.

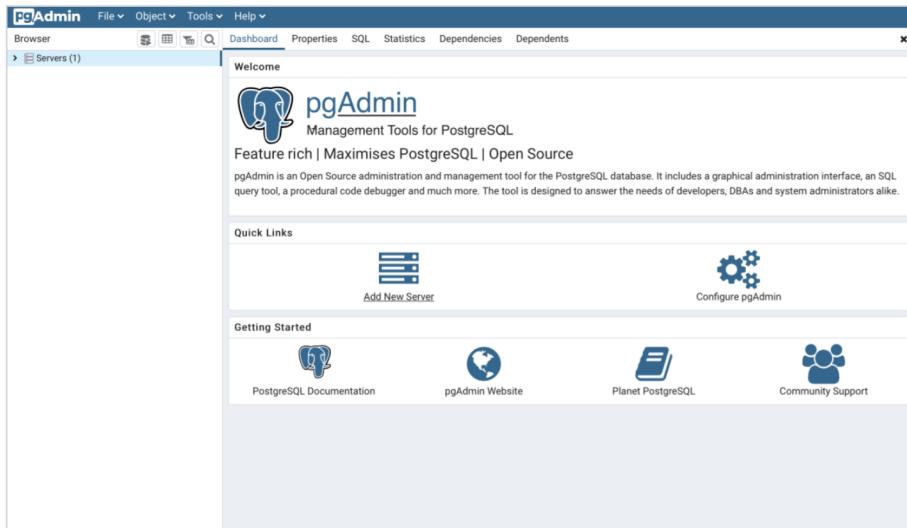


Figura 1: Homepage di pgAdmin4.

Nel menù a sinistra sono visualizzati i server connessi. Nel caso in cui accanto a “Server” non sia indicato alcun numero tra parentesi (vedi Figura 2), e anche dopo aver fatto doppio click su “Servers” non compare alcun numero, allora significa che nessun server è collegato con pgAdmin4.

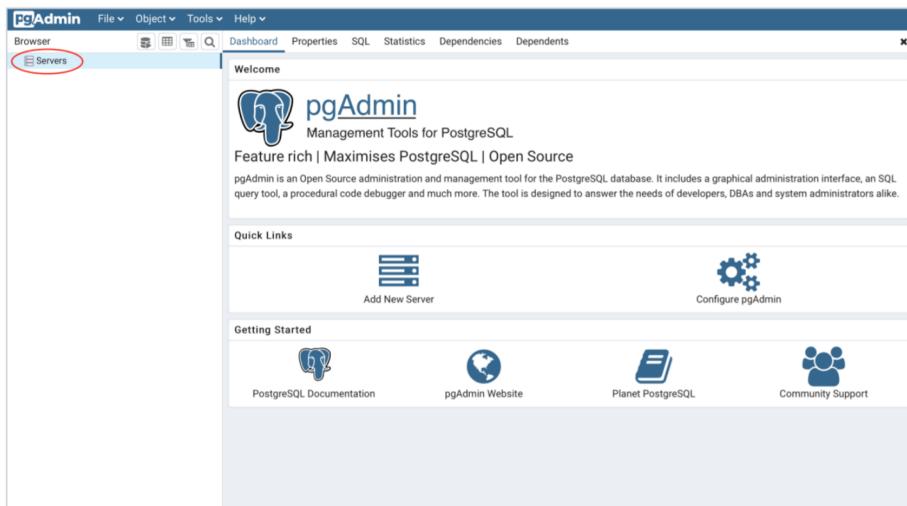


Figura 2: pgAdmin4 senza server connessi.

Per collegare il server locale manualmente, è necessario creare un nuovo server selezionando **Object** dalla barra del menù in alto e scegliere **Create** > **Server** come mostrato in Figura 3.

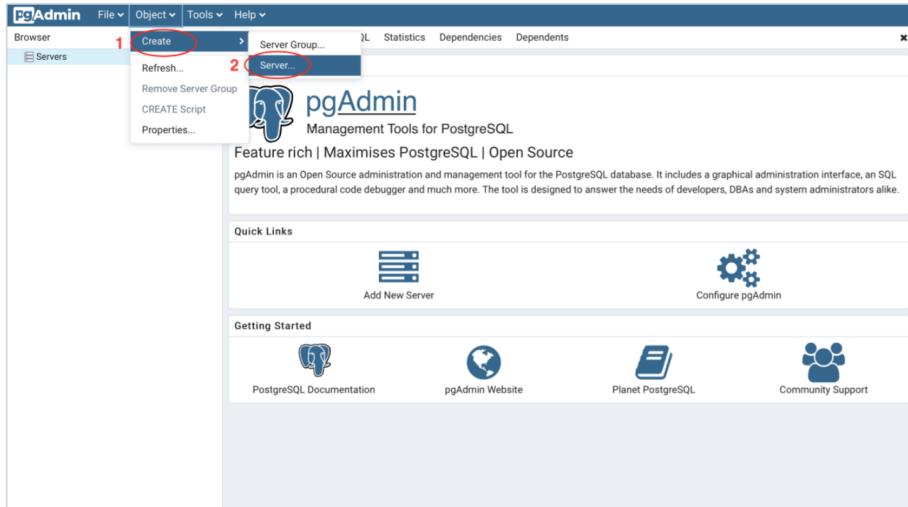


Figura 3: Accesso allo strumento di creazione di un nuovo server

Nel pannello per la creazione del nuovo server è necessario indicare un nome (Figura 4), quindi aprire il pannello “Connection” ed indicare l’indirizzo del server che nel nostro caso sarà “localhost”, come mostrato in Figura 5.

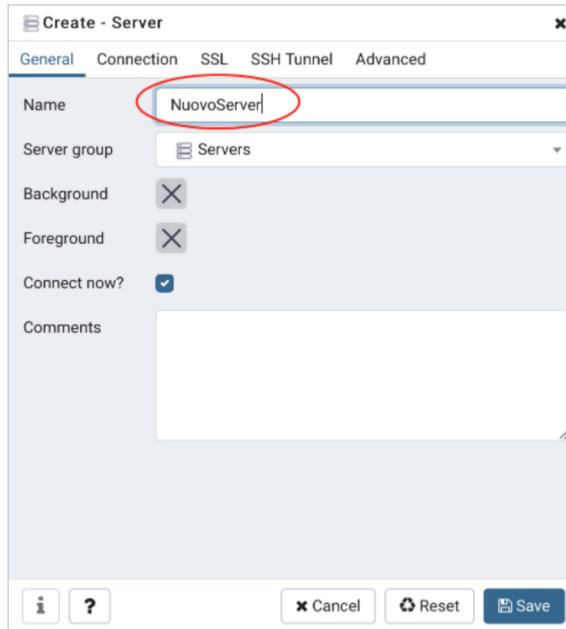


Figura 4: Impostazione del nome del server.

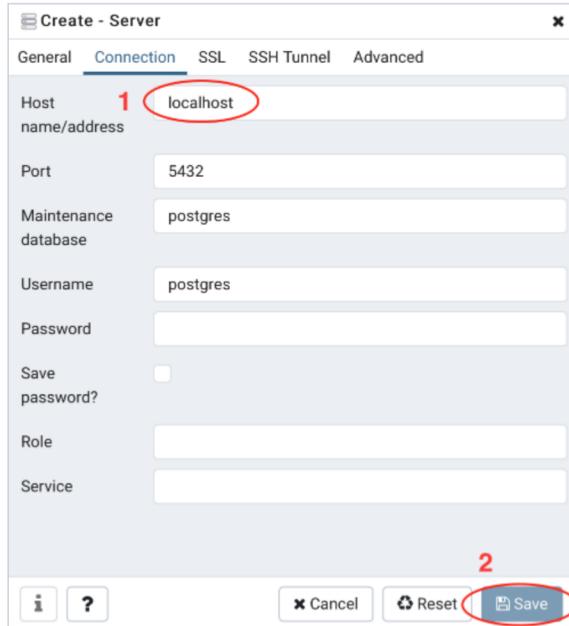


Figura 5: Impostazione dell'indirizzo del server.

**Nota:** In linux può apparire il seguente errore.

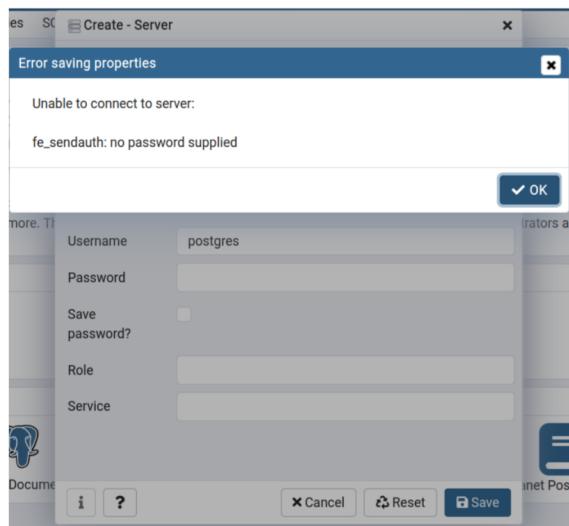


Figura 6: Possibile errore in linux

Tale errore è dovuto al fatto che nel file di configurazione *pg\_hba.conf* è esplicitata la necessità di una password. Per ovviare a ciò è sufficiente inserire la password nel corrispettivo campo, oppure modificare il file *pg\_hba.conf*. La modifica può essere effettuata lanciando il seguente comando da terminale:

```
01 | sudo gedit /etc/postgresql/12/main/pg_hba.conf
```

dove *gedit* è l'editor di testo selezionato. Se non presente basta installarlo o selezionarne un altro. Successivamente modificare le linee relative a *IPv4 local connections* e *IPv6 local connections* sostituendo il parametro *md5* con *trust*. Ottenendo la nuova configurazione:

```
89 # "local" is for Unix domain socket connections only
90 local    all        all                                peer
91 # IPv4 local connections:
92 host     all        all      127.0.0.1/32            trust
93 # IPv6 local connections:
94 host     all        all      ::1/128               trust
```

Figura 7: Corrispondenti righe del file dopo la modifica.

Salvare, e lanciare il comando per aggiornare le configurazioni:

```
01 | sudo service postgresql reload
```

Infine, riavviare pgAdmin.

Una volta connesso il server manualmente, o nel caso in cui sia già presente tale collegamento, nel pannello a sinistra sarà visualizzato il numero di server presenti tra parentesi a fianco alla scritta “Servers”. Cliccando su “Servers” si accederà alla lista dei server disponibili e sarà possibile selezionare quindi il proprio server locale cliccando sul relativo nome (“PostgreSQL 12” nel caso venga creato di default, oppure il nome impostato durante la connessione manuale). Per accedere potrebbe essere necessario inserire la password impostata durante l’installazione di PostgreSQL. In alcune installazioni potrebbe non essere impostata alcuna password di default e in questo caso si accederà direttamente alla *Dashboard*.

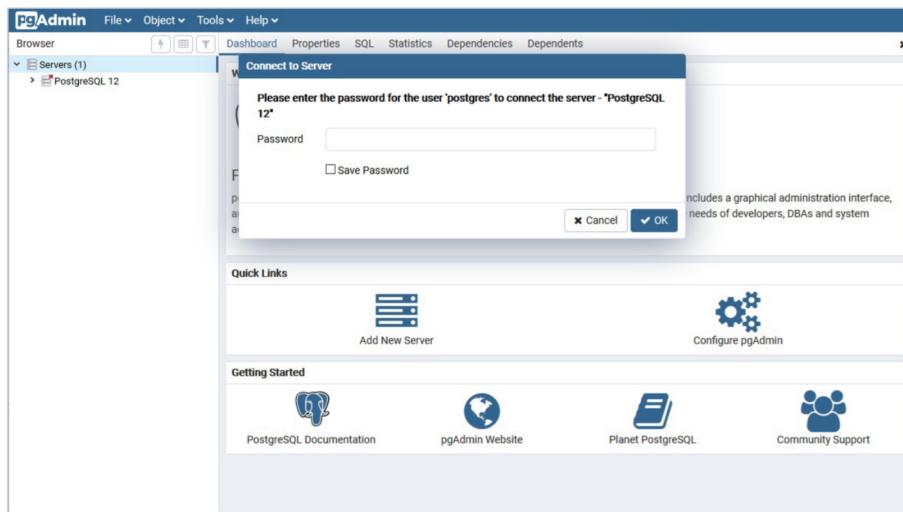


Figura 8: Inserimento della password.

Questo ci porterà alla schermata mostrata in Figura 9, in cui è presente un database di default chiamato “*postgres*”.

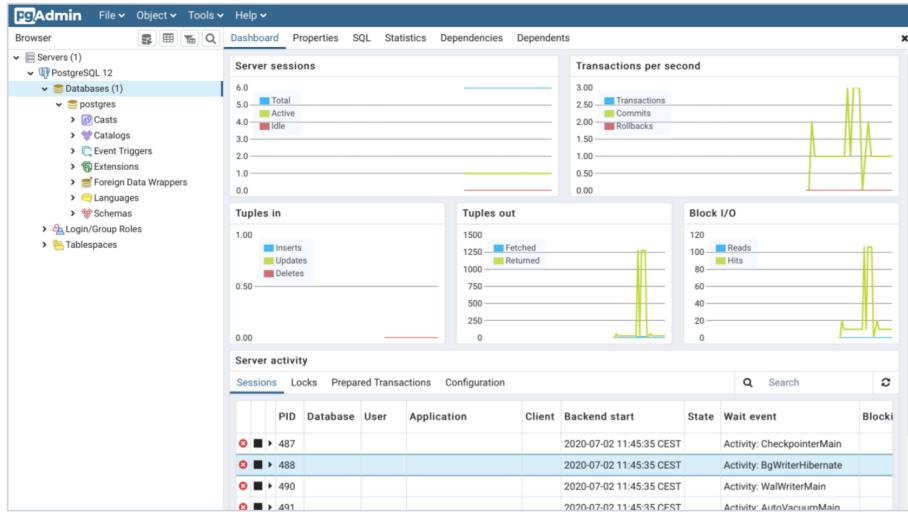


Figura 9: Dashboard iniziale di pgAdmin4

## 1.1 Creazione database

Per creare un nuovo database è necessario selezionare dal menù sulla sinistra la voce **Databases** e quindi cliccare dal menù in alto **Object** > **Create** > **Database**.

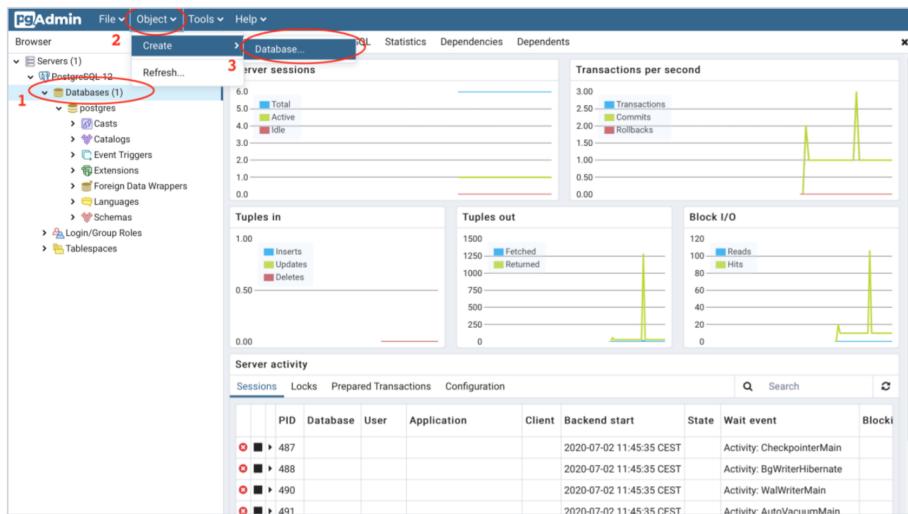


Figura 10: Creazione di un nuovo database.

In questo modo avremo accesso alla finestra per la creazione di un nuovo database. Da qui possiamo scegliere il nome. Per questo laboratorio lo chiameremo “sampleDB” e premendo **Save** lo andremo a salvare sul nostro server locale.

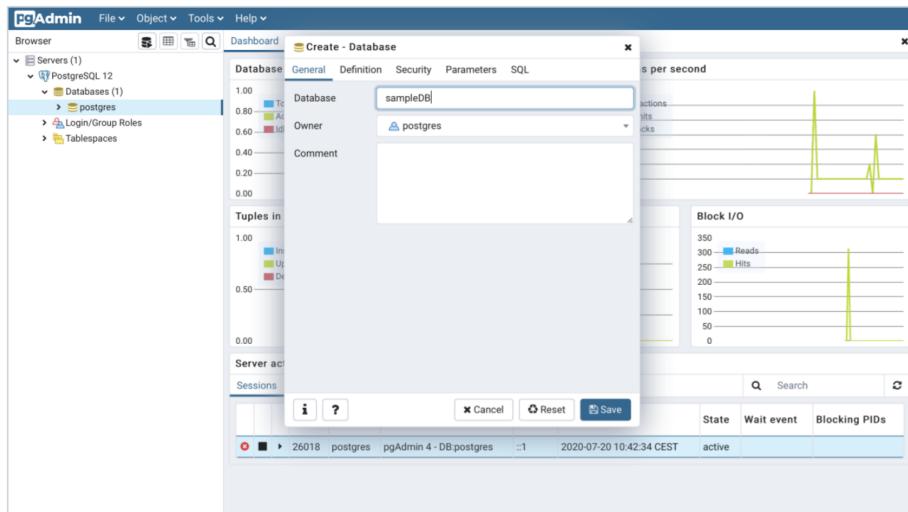


Figura 11: Inserimento del nome del database.

Una volta creato, comparirà nel menù a sinistra un nuovo database con il nome che abbiamo scelto. In questo caso il nome sarà “sampleDB”, come mostrato in Figura 12.

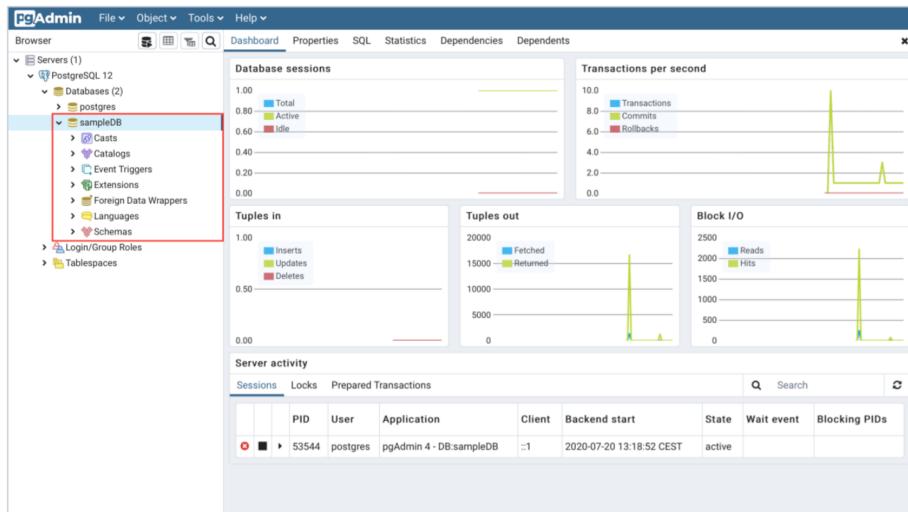


Figura 12: Visualizzazione del nuovo database.

A questo punto abbiamo creato un nuovo database vuoto. Per poterlo popolare con dei dati dobbiamo innanzitutto creare delle tabelle. Per farlo ci sono due alternative: utilizzare l’interfaccia grafica, oppure definire una query SQL manualmente. Per questo laboratorio definiremo manualmente la query di creazione. Per farlo dobbiamo selezionare il nostro database dal menù e aprire il **Query Tool** dal menù **Tools** nella barra in alto (Figura 13).

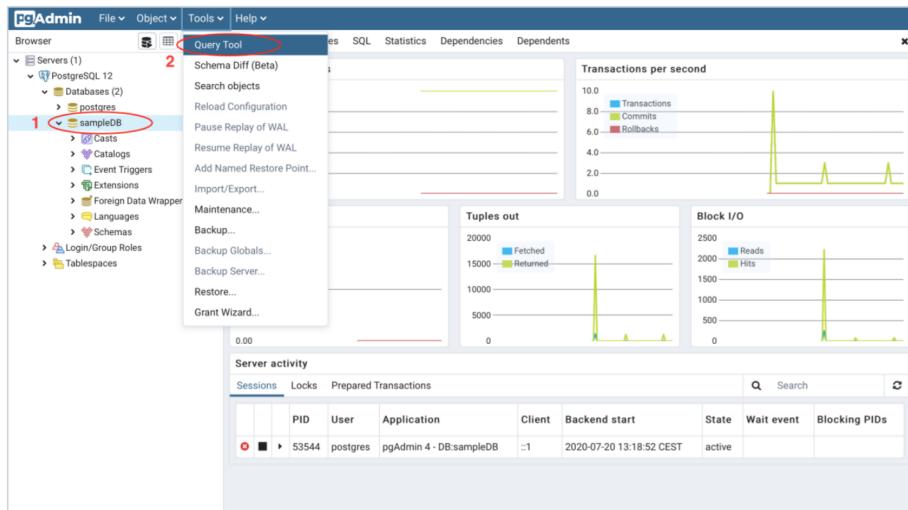


Figura 13: Procedura per aprire il Query Tool.

La schermata per l'inserimento delle query si presenta come un normale editor in cui possiamo inserire il nostro codice SQL e successivamente eseguirlo. Per fare questo occorre selezionare il pulsante **Esegui** mostrato in Figura 14.

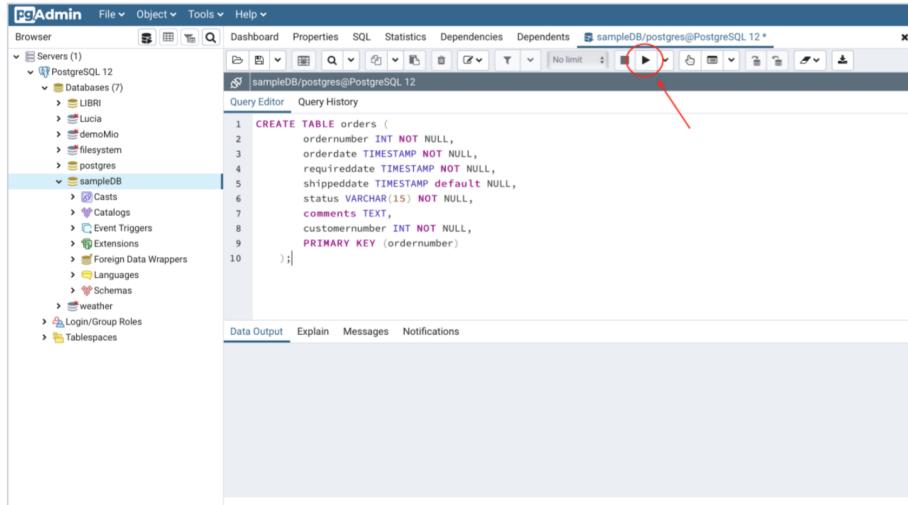


Figura 14: Esecuzione di una query.

Definiamo quindi le seguenti tabelle con i relativi campi dati:

```

01 | CREATE TABLE orders (
02 |     ordernumber INT NOT NULL,
03 |     orderdate TIMESTAMP NOT NULL,
04 |     requireddate TIMESTAMP NOT NULL,
05 |     shippeddate TIMESTAMP default NULL,
06 |     status VARCHAR(15) NOT NULL,

```

```

07 |     comments TEXT,
08 |     customernumber INT NOT NULL,
09 |     PRIMARY KEY (ordernumber)
10 | );

```

```

01 | CREATE TABLE products (
02 |     productcode VARCHAR(15) NOT NULL,
03 |     productname VARCHAR(70) NOT NULL,
04 |     productline VARCHAR(50) NOT NULL,
05 |     productscale VARCHAR(10) NOT NULL,
06 |     productvendor VARCHAR(50) NOT NULL,
07 |     productdescription TEXT NOT NULL,
08 |     quantityinstock SMALLINT NOT NULL,
09 |     buyprice DOUBLE PRECISION NOT NULL,
10 |     msrp DOUBLE PRECISION NOT NULL,
11 |     PRIMARY KEY (productcode)
12 | );

```

Creiamo quindi una terza tabella `orderdetails` che contiene due riferimenti alle tabelle precedenti tramite le due chiavi esterne a `ordernumber` sulla tabella `orders` e `productcode` sulla tabella `products`.

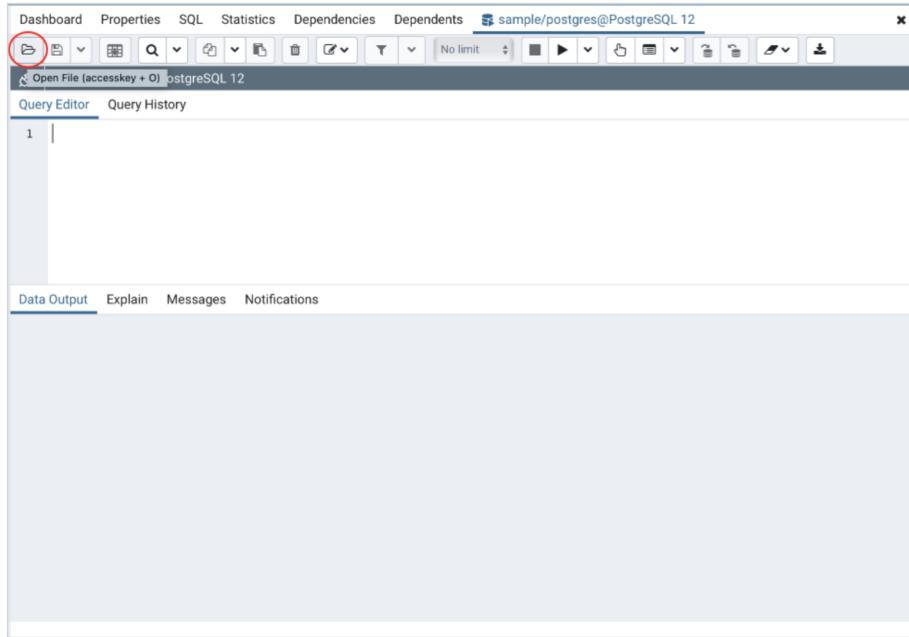
```

01 | CREATE TABLE orderdetails (
02 |     ordernumber INT NOT NULL,
03 |     productcode VARCHAR(15) NOT NULL,
04 |     quantityordered INT NOT NULL,
05 |     priceeach DOUBLE PRECISION NOT NULL,
06 |     orderlinenumber SMALLINT NOT NULL,
07 |     PRIMARY KEY (ordernumber,productcode),
08 |     FOREIGN KEY (ordernumber) REFERENCES orders(ordernumber),
09 |     FOREIGN KEY (productcode) REFERENCES products(productcode)
10 | );

```

## 1.2 Importazione database

Una volta create le prime tabelle, importiamo il resto del database e i relativi comandi di inserimento dei dati contenuti nel file `sample.sql`, disponibile su Moodle. Per farlo apriamo il `Query Tool` e selezioniamo l'icona della cartella posta sulla sinistra della barra dei comandi (come mostrato in Figura 15).

Figura 15: Importazione di un file `sql` nel Query Tool.

Una volta trovato il file desiderato navigando il *file system*, è necessario selezionarlo e quindi premere `Select` per proseguire con l'importazione dello script.

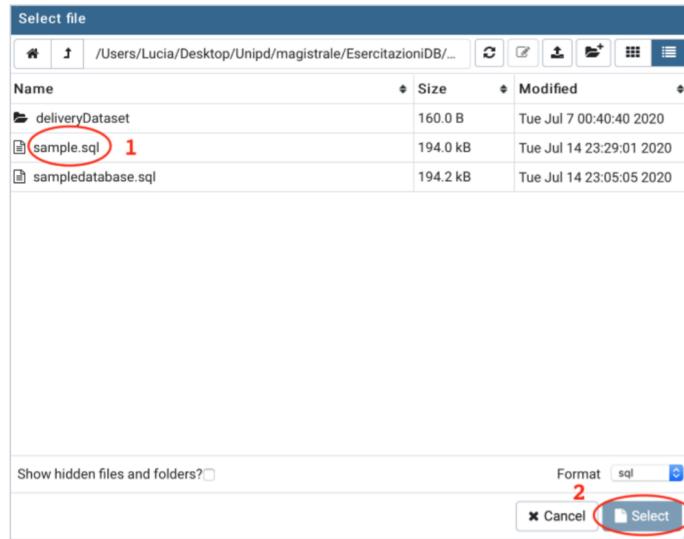


Figura 16: Selezione del file da importare.

Se il file viene importato correttamente, la query sarà visibile ed editabile nella sezione `Query Editor` e sarà possibile eseguirla normalmente come indicato precedentemente. Se l'esecuzione avviene con successo, verrà mostrato un avviso in basso a destra come in Figura 17.

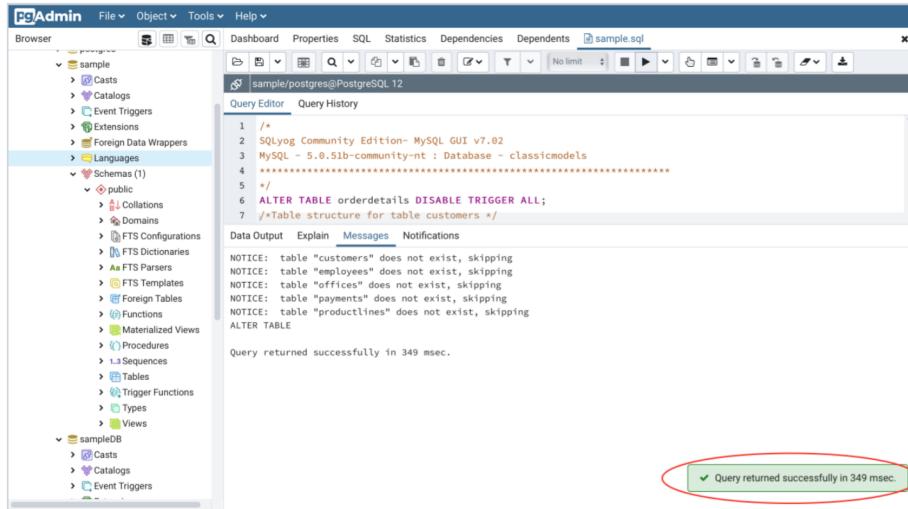


Figura 17: Query eseguita con successo.

A questo punto abbiamo creato un database che rappresenta un sistema di gestione aziendale in cui si tiene traccia di impiegati, uffici, prodotti, linee produttive e clienti. Tutte queste informazioni vengono quindi utilizzate per rappresentare gli ordini gestiti e relativi dettagli. Lo schema in Figura 18 illustra tutte le tabelle presenti e le loro relazioni.

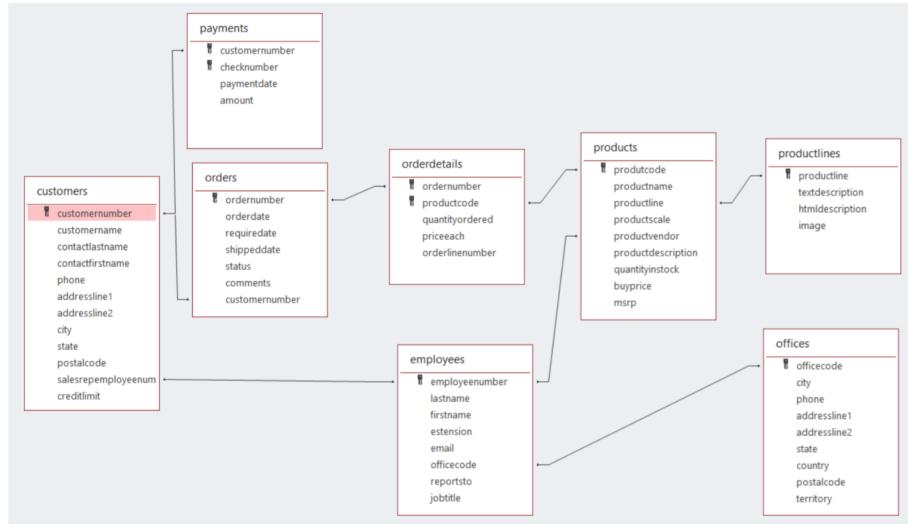


Figura 18: Schema del database.

## 2 Visualizzazione e manipolazione dei dati

Avendo ora a disposizione un database completo di dati, possiamo visualizzare i dati contenuti nelle tabelle. Per farlo, selezioniamo per esempio la tabella **customers** e premiamo tasto destro. Dal menù selezioniamo **View/Edit Data** > **All Rows**, come mostrato in Figura 19.

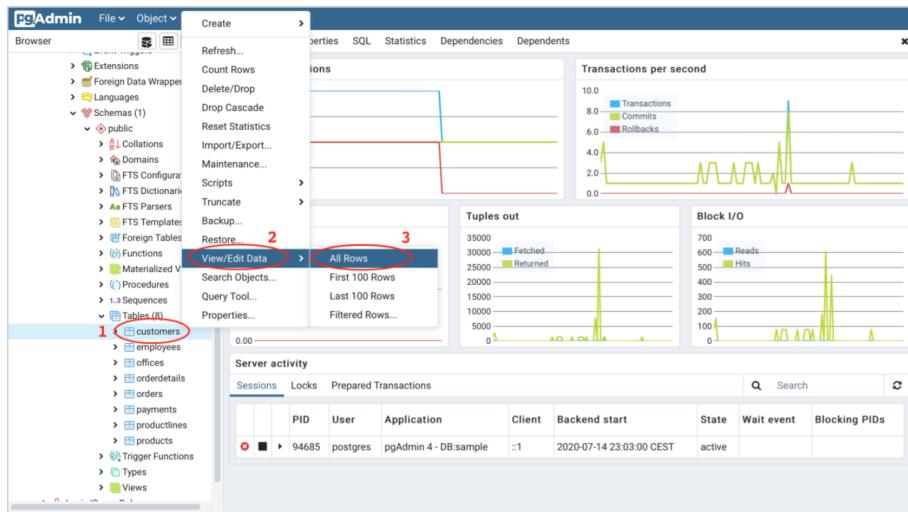


Figura 19: Procedura per la visualizzazione dei dati di una tabella

Nel pannello principale (1) in Figura 20) chiamato “Query Editor” viene mostrata la query eseguita automaticamente per selezionare tutti i valori della tabella selezionata.

Nel pannello in basso ((2) in Figura 20) vengono invece illustrati i dati selezionati in forma tabellare.

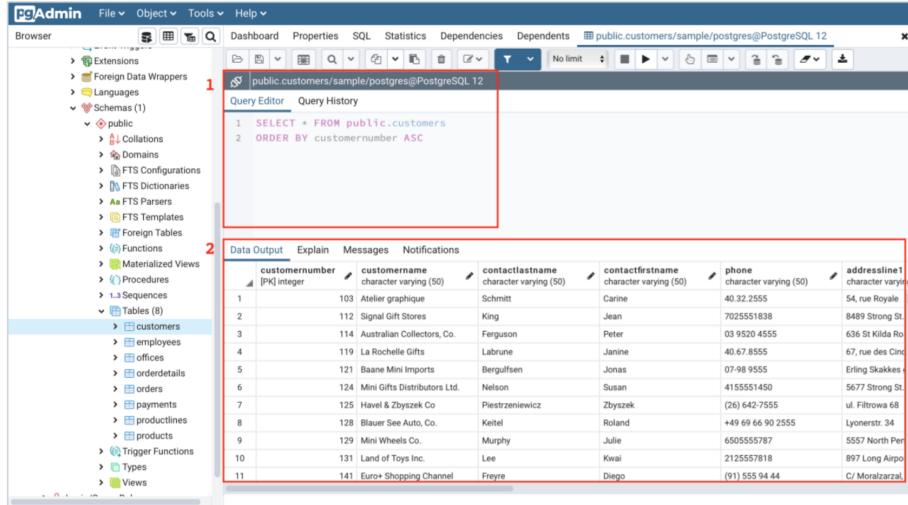


Figura 20: Interfaccia per la visualizzazione dei dati

Per eseguire delle query personalizzate è necessario accedere al **Query Tool** utilizzato in precedenza per la creazione delle tabelle. Tramite questo pannello è infatti possibile definire manualmente le query per poter interrogare le tabelle.

Il **Query Tool** può essere aperto tramite il menù a comparsa premendo tasto destro sul nome delle tabelle, oppure dalla barra del menù in alto selezionando **Tools** > **Query Tool**.

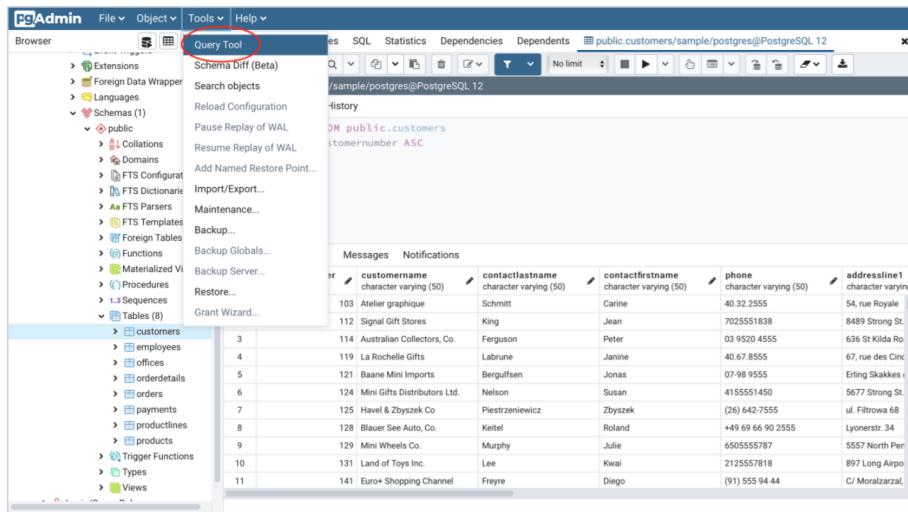


Figura 21: Procedura per aprire il Query Tool dalla barra degli strumenti.

L’interfaccia comprende varie sezioni (vedi Figura 22):

1. Una barra degli strumenti che comprende varie funzionalità come salvare, avviare e fermare la query e molte altre;
2. Un pannello editor in cui è possibile definire le queries;
3. Un pannello chiamato “Query History” che ci permette di visualizzare lo storico delle query eseguite;
4. Un pannello di controllo dove verranno visualizzati i risultati delle queries oppure eventuali messaggi di errore.

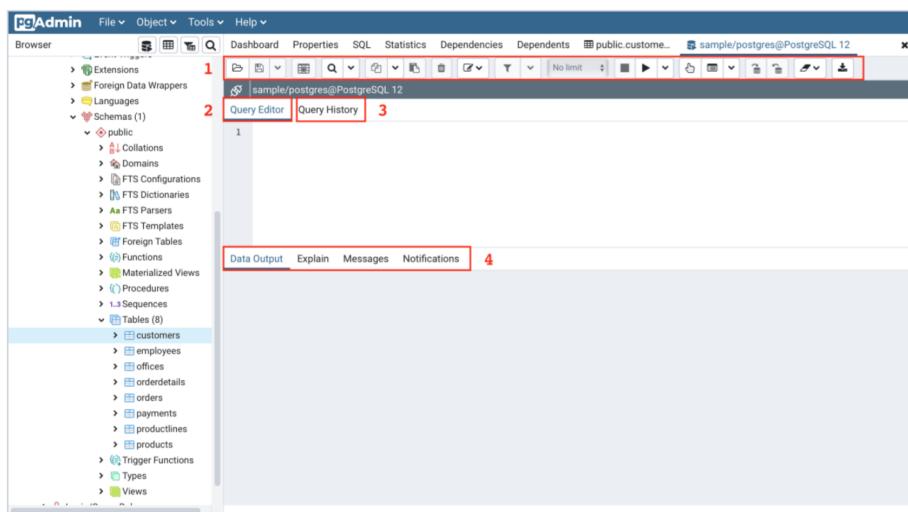


Figura 22: Descrizione dei vari pannelli che compongono il Query Tool.

## 2.1 Query di base

1. Trovare il nome (`customername`) di tutti i clienti.

```
01 | SELECT customername
02 | FROM customers;
```

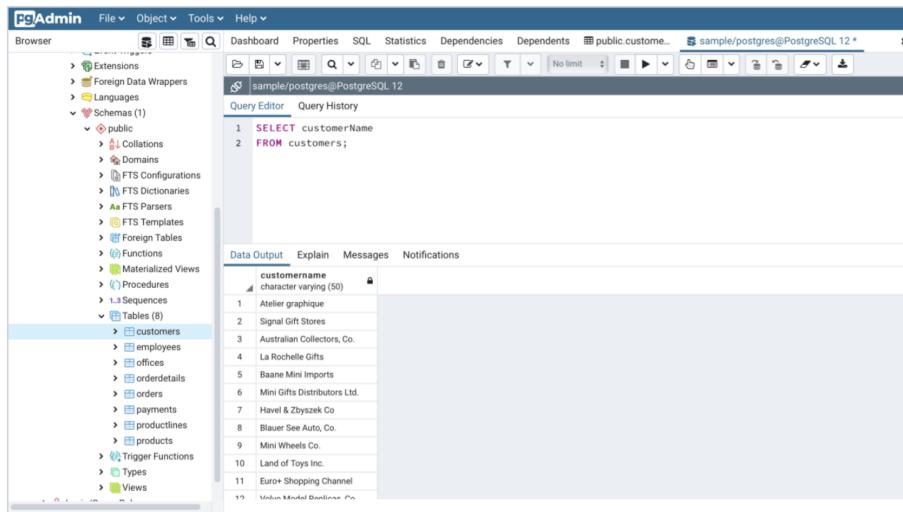


Figura 23: Esecuzione e visualizzazione dei risultati della prima query.

2. Trovare la città (`city`) dove ha sede il cliente “Frau da Collezione”.

```
01 | SELECT city
02 | FROM customers
03 | WHERE (customername='Frau da Collezione');
```

3. Trovare nome (`firstname`), cognome (`lastname`) e la email del presidente (`jobtitle = 'President'`).

```
01 | SELECT firstname, lastname, email
02 | FROM employees
03 | WHERE (jobtitle='President');
```

4. Trovare il Nome e il limite di credito (`creditlimit`) di tutti i clienti che hanno un limite di credito superiore a 100000.

```
01 | SELECT customername, creditlimit
02 | FROM customers
03 | WHERE (creditlimit>100000);
```

5. Trovare tutte le linee di prodotti (ogni linea deve comparire una sola volta).

```
01 |   SELECT DISTINCT productline  
02 |   FROM productlines;
```

6. Trovare il nome e il cognome dei dipendenti in ordine alfabetico, prima secondo il cognome e poi secondo il nome.

```
01 |   SELECT lastname,firstname  
02 |   FROM employees  
03 |   ORDER BY lastname,firstname;
```

7. Trovare per ogni dipendente il suo cognome e quello delle persone che dirige.

```
01 |   SELECT dirigente.lastname AS Dirigente,  
02 |           dipendente.lastname AS Dipendente  
03 |   FROM employees AS dipendente, employees AS dirigente  
04 |   WHERE dipendente.reportsto=dirigente.employeenumber;
```

8. Restituire il nome (`customername`) e il limite di credito (`creditlimit`) di tutti i clienti, ordinati rispetto al limite di credito decrescente.

```
01 |   SELECT customername AS Nome,  
02 |           creditlimit AS credito  
03 |   FROM customers  
04 |   ORDER BY creditlimit DESC;
```

### 3 Esercizi sulle query

**Query 1** Capire se esiste un dipendente con il ruolo di dirigente “Sale Manager (EMEA)” a “Paris”;

**Query 2** Trovare il nome dei clienti che hanno ordinato prodotti della linea “Planes”;

**Query 3** Trovare il nome di tutti i clienti della sede di “Tokyo”;

**Query 4** Trovare il codice cliente di tutti i clienti che non hanno eseguito ordini nell’anno 2005.  
(Il formato standard di un valore di tipo *timestamp* è “yyyy-mm-dd”);

**Query 5** Restituire il nome delle città con almeno 2 dipendenti.

**Query 6** Restituire tutte le linee di prodotti con il numero di prodotti a loro associati.

**Query 7** Restituire le linee di prodotto in ordine decrescente di guadagno considerando un solo prodotto venduto per ogni ordine.

**Query 8** Come la query precedente considerando anche la quantità di prodotti dell’ordine (*quantityordered*).

### 4 Altri Esercizi

**Esercizio 1** Definire un attributo che permetta di rappresentare stringhe di lunghezza massima pari a 256 caratteri, su cui non sono ammessi valori nulli e con valore di default “sconosciuto”.

**Esercizio 2** Dare le definizioni SQL delle tabelle:

AUTORE (Nome, Cognome, DataNascita, Nazionalità)  
LIBRO (TitoloLibro, NomeAutore, CognomeAutore, Lingua)

Per il vincolo foreign key specificare una politica di CASCADE sulla cancellazione e di SET NULL sulle modifiche.

**Esercizio 3** Dato lo schema dell’esercizio 3, spiegare cosa può capitare con l’esecuzione dei seguenti comandi di aggiornamento:

1. delete from AUTORE where Cognome = ‘Rossi’
2. update LIBRO set NomeAutore= ‘Umberto’ where CognomeAutore = ‘Eco’
3. insert into AUTORE(Nome,Cognome) values (‘Antonio’,‘Bianchi’)
4. update AUTORE set Nome = ‘Giovanni’ where Cognome = ‘Pirandello’

Per testare il funzionamento delle query indicate, viene fornito il file `libri.sql`, che contiene i dati per il popolamento delle tabelle “libro” ed “autore”.

**Esercizio 4** Dare le definizioni SQL delle tabelle:

SERVIZI\_SOCIALI (Città, Servizio, Anno, Spesa)  
POSIZIONE (Città, Regione, Abitanti)

in modo che non sia possibile avere due città con lo stesso nome, e SERVIZI\_SOCIALI.Città si riferisca a POSIZIONE.Città. Ad esempio, la tupla (Padova, Babysitting, 2019, 30000) in SERVIZI\_SOCIALI indica che ”Padova” ha speso nel 2019 la cifra di 30000€ per il servizio sociale ”Babysitting”.

**Esercizio 5** Dato lo schema dell'esercizio 4, definire le seguenti query:

1. Restituire le città che forniscono almeno due servizi sociali.
2. Restituire le città che forniscono esattamente un servizio sociale.

Per testare il funzionamento delle query, viene fornito il file **servizi.sql**, che contiene i dati per il popolamento delle tabelle “SERVIZI\_SOCIALI” e “POSIZIONE”.

## 5 Soluzioni query

### Query 1

```

01 | SELECT employees.lastname
02 | FROM employees,offices
03 | WHERE employees.officeCode=offices.officecode AND employees.jobTitle='Sale
    Manager (EMEA)'
04 | AND offices.city='Paris';

```

### Query 2

```

01 | SELECT DISTINCT customerName
02 | FROM customers,orders,orderdetails,products
03 | WHERE customers.customerNumber=orders.customerNumber
04 | AND orders.orderNumber=orderdetails.orderNumber
05 | AND orderdetails.productCode=products.productCode
06 | AND productline='Planes';

```

### Query 3

```

01 | SELECT customerName
02 | FROM offices,customers,employees
03 | WHERE employees.officeCode=offices.officecode
04 | AND employees.employeeNumber=customers.salesRepEmployeeNumber
05 | AND offices.city='Tokyo';

```

### Query 4

```

01 | SELECT customernumber
02 | FROM customers
03 | EXCEPT (SELECT DISTINCT customernumber
04 |     FROM orders
05 |     WHERE orders.orderdate >= '2005-01-01' AND
06 |             orders.orderdate <= '2005-12-31');

```

### Query 5

```

01 | SELECT DISTINCT of1.city
02 | FROM offices AS of1, employees AS emp1, offices AS of2, employees AS emp2
03 | WHERE of1.officecode = emp1.officecode AND
04 |     of2.officecode = emp2.officecode AND
05 |     of1.city = of2.city AND
06 |     emp1.employeeNumber <> emp2.employeeNumber;

```

## Query 6

```

01 |   SELECT pl.productline, COUNT(*)
02 |   FROM productlines AS pl, products as pr
03 |   WHERE pr.productline = pl.productline
04 |   GROUP BY pl.productline;

```

## Query 7

```

01 |   SELECT pl.productline AS proline, SUM(priceeach) AS total
02 |   FROM productlines AS pl, products AS pr, orders AS ord, orderdetails AS det
03 |   WHERE pl.productline=pr.productline AND
04 |       ord.ordernumber=det.ordernumber AND
05 |       pr.productcode=det.productcode
06 |   GROUP BY pl.productline
07 |   ORDER BY total DESC

```

## Query 8

```

01 |   SELECT pl.productline AS proline, SUM(priceeach*quantityordered) AS total
02 |   FROM productlines AS pl, products AS pr, orders AS ord, orderdetails AS det
03 |   WHERE pl.productline=pr.productline AND
04 |       ord.ordernumber=det.ordernumber AND
05 |       pr.productcode=det.productcode
06 |   GROUP BY pl.productline
07 |   ORDER BY total DESC

```

## 6 Soluzioni altri esercizi

### Esercizio 1

```

01 |   CREATE DOMAIN STRING as character varying (256)
02 |   default 'sconosciuto' NOT NULL

```

### Esercizio 2

```

01 |   CREATE TABLE autore(
02 |     Nome character(20),
03 |     Cognome character(20),
04 |     DataNascita date,
05 |     Nazionalita character(20),
06 |     PRIMARY KEY (Nome,Cognome)
07 |   );
08 |
09 |   CREATE TABLE libro(
10 |     TitoloLibro character(30) PRIMARY KEY,
11 |     NomeAutore character(20),

```

```

12 |     CognomeAutore character(20),
13 |     Lingua character(20),
14 |     FOREIGN KEY (NomeAutore,CognomeAutore)--Al primo argomento di FOREIGN
KEY corrisponde il primo attributo argomento di REFERENCES
15 |     REFERENCES AUTORE (Nome,Cognome)
16 |     ON DELETE CASCADE -- Tutte le righe della tabella interna
corrispondenti alla riga cancellata vengono cancellate
17 |     ON UPDATE SET NULL
18 | );

```

### Esercizio 3

- Prima di eseguire il comando `delete` possiamo selezionare l'intero contenuto della tabella `LIBRO` e, come mostrato in Figura 24, osserviamo che ci sono alcuni libri in cui il cognome dell'autore è “Rossi”.

The screenshot shows the pgAdmin interface with the 'LIBRI' database selected. In the left sidebar, under 'Tables (2)', the 'autore' table is selected. In the main area, a query editor window displays the following SQL command:

```
1 SELECT * FROM libro
```

The results show a table with four columns: titololibro, nomeautore, cognomeautore, and lingua. The rows are:

	titololibro	nomeautore	cognomeautore	lingua
1	Il barone rampante	Italo	Calvino	Italiano
2	Le citta invisibili	Italo	Calvino	Italiano
3	Le cosmicomiche	Italo	Calvino	Italiano
4	Il cavaliere inesistente	Italo	Calvino	Italiano
5	Marcovaldo	Italo	Calvino	Italiano
6	Il visconte dimezzato	Italo	Calvino	Italiano
7	Ifu Mattia Pascal	Luigi	Pirandello	Italiano
8	Uno, nessuno e centomila	Luigi	Pirandello	Italiano
9	Novelle per un anno	Luigi	Pirandello	Italiano
10	I vecchi e i giovani	Luigi	Pirandello	Italiano
11	Zibaldone	Giacomo	Leopardi	Italiano
12	Operette Morali	Giacomo	Leopardi	Italiano
13	Camorra	Luca	Rossi	Italiano
14	Arrivederci mafia	Luca	Rossi	Italiano
15	Catturandi	Luca	Rossi	Italiano
16	I disarmati	Luca	Rossi	Italiano

Figura 24: Tabella LIBRO completa.

Il comando `delete`, cancella dalla tabella `AUTORE` tutte le tuple con `cognome = 'Rossi'`. A causa della politica cascade anche le tuple di `LIBRO` con `cognomeautore = 'Rossi'` verranno eliminate.

Possiamo quindi verificarlo selezionando nuovamente tutti i libri dalla tabella `LIBRO` e notare che come atteso, non compare alcun libro in cui il `cognomeautore` sia uguale a “Rossi” come mostrato in Figura 25.

	titololibro	nomeautore	cognomeautore	lingua
1	Il barone rampante	Italo	Calvino	Italiano
2	Le città invisibili	Italo	Calvino	Italiano
3	Le cosmonomiche	Italo	Calvino	Italiano
4	Il cavaliere inesistente	Italo	Calvino	Italiano
5	Marcovaldo	Italo	Calvino	Italiano
6	Il visconte dimezzato	Italo	Calvino	Italiano
7	Il fu Mattia Pascal	Luigi	Pirandello	Italiano
8	Uno, nessuno e centomila...	Luigi	Pirandello	Italiano
9	Novelle per un anno	Luigi	Pirandello	Italiano
10	I vecchi e i giovani	Luigi	Pirandello	Italiano
11	Zibaldone	Giacomo	Leopardi	Italiano
12	Operette Morali	Giacomo	Leopardi	Italiano
13	The Da Vinci code	Dan	Brown	Inglese
14	Inferno	Dan	Brown	Inglese
15	The lost symbol	Dan	Brown	Inglese
16	Angels and demons	...	Brown	Inglese

Figura 25: Tabella LIBRO dopo la cancellazione degli autori con cognome uguale a ‘Rossi’.

2. Il comando è ininfluente : nella tabella LIBRO infatti la condizione `cognomeautore = 'Eco'` non è mai verificata
3. Il comando aggiunge una nuova tupla alla tabella AUTORE. Non ha alcun effetto sulla tabella LIBRO.
4. Per osservare quanto accade, prima di eseguire l'aggiornamento, selezioniamo tutti i libri che hanno `cognomeAutore = 'Pirandello'` come mostrato in Figura 26, per avere dei dati di riferimento.

The screenshot shows the PgAdmin interface. In the left sidebar, under the 'LIBRI' database, the 'Tables' section is expanded, showing 'autore'. In the main pane, the 'Query Editor' contains the following SQL code:

```
1 SELECT * FROM libro WHERE cognomeautore = 'Pirandello'
```

The results table shows four rows of data:

titololibro	nomeautore	cognomeautore	lingua
Il fu Mattia Pascal	Luigi	Pirandello	Italiano
Uno, nessuno e centomila	Luigi	Pirandello	Italiano
Novelle per un anno	Luigi	Pirandello	Italiano
I vecchi e i giovani	Luigi	Pirandello	Italiano

Figura 26: Selezione di tutti i libri di Pirandello.

Osserviamo ad esempio che il libro intitolato “Il fu Mattia Pascal” ha come autore “Pirandello”. A questo punto eseguiamo l’aggiornamento e controlliamo che abbia avuto effetto sulla tabella AUTORE.

The screenshot shows the PgAdmin interface after an update. The 'Query Editor' contains the following SQL code:

```
1 UPDATE autore SET nome = 'Giovanni' WHERE cognome = 'Pirandello';
2 SELECT * FROM autore;
```

The results table shows the updated data, with the row for 'Pirandello' now having 'Giovanni' in the 'nome' column instead of 'Luigi'.

nome	cognome	datanascita	nazionalita
Italo	Calvino	1923-10-15	Italia
Giacomo	Leopardi	1798-06-29	Italia
Luca	Rossi	1977-04-15	Italia
Dan	Brown	1964-06-22	StatiUniti
Agatha	Christie	1890-09-15	RegnoUnito
Mary	Shelley	1797-08-30	RegnoUnito
Giovanni	Pirandello	1867-06-28	Italia

Figura 27: Tabella autore dopo l’aggiornamento del nome di “Pirandello”.

Le modifiche alla tabella autore hanno una politica SET NULL pertanto tutti i libri che prima della modifica avevano come autore Pirandello ora dovrebbero avere nomeAutore e cognomeautore uguali a NULL. Possiamo verificarlo selezionando per esempio il libro con

`titololibro = 'Il fu Mattia Pascal'` che, prima della modifica, era assegnato all'autore “Pirandello”. Dopo l'aggiornamento possiamo notare che a causa della politica SET NULL questo libro ha `cognomeautore = NULL` e `nomeautore = NULL` (vedi Figura 28).

titololibro	nomeautore	cognomeautore	lingua
Il fu Mattia Pascal	[null]	[null]	Italiano

Figura 28: Caratteristiche del libro “Il fu Mattia Pascal” dopo l'aggiornamento dell'autore

## Esercizio 4

```

01 | CREATE TABLE POSIZIONE(
02 |   Citta VARCHAR(32) PRIMARY KEY,
03 |   Regione VARCHAR(32),
04 |   Abitanti INT
05 | );
06 |
07 | CREATE TABLE SERVIZI_SOCIALI(
08 |   Citta VARCHAR(32) REFERENCES POSIZIONE(Citta),
09 |   Servizio VARCHAR(32),
10 |   Anno INT,
11 |   Spesa DOUBLE PRECISION,
12 |   PRIMARY KEY (Citta, Servizio, Anno)
13 | );

```

## Esercizio 5

- Restituire le città che forniscono almeno due servizi sociali:

```

01 | SELECT DISTINCT S1.Citta
02 | FROM SERVIZI_SOCIALI S1, SERVIZI_SOCIALI S2
03 | WHERE S1.Citta=S2.Citta AND S1.Servizio<>S2.Servizio

```

Nota bene: in questa query è necessario l'utilizzo di `DISTINCT` per eliminare tutte le tuple duplicate.

2. Restituire le città che forniscono esattamente un servizio sociale:

```
01 |   SELECT Citta
02 |   FROM SERVIZI_SOCIALI
03 |   EXCEPT(
04 |       SELECT S1.Citta
05 |       FROM SERVIZI_SOCIALI S1, SERVIZI_SOCIALI S2
06 |       WHERE S1.Citta=S2.Citta AND S1.Servizio<>S2.Servizio
07 |   )
```

In questo caso non è necessario DISTINCT per via della proprietà di EXCEPT di lavorare sugli insiemi.