

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO DI MATEMATICA

LAUREA TRIENNALE IN INFORMATICA

BASI DI DATI - LABORATORIO 1

Introduzione a PostgreSQL

Massimiliano de Leoni
Matteo Zavatteri
Alessandro Padella
Matteo Gioele Collu

massimiliano.deleoni@unipd.it
matteo.zavatteri@unipd.it
alessandro.padella@unipd.it
matteogioele.collu@phd.unipd.it

Indice

1	Introduzione a pgAdmin4	3
1.1	Creazione database	7
1.2	Importazione database	10
2	Visualizzazione e manipolazione dei dati	10
2.1	Query di base	13
3	Esercizi sulle query	17
4	Altri Esercizi	17

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 (da Laboratorio).	5
6	Impostazione dell'indirizzo del server (in locale).	5
7	Possibile errore in Windows e Linux	5
8	Corrispondenti righe del file dopo la modifica.	6
9	Inserimento della password.	6
10	Dashboard iniziale di pgAdmin4	7
11	Creazione di un nuovo database.	7
12	Inserimento del nome del database.	8
13	Visualizzazione del nuovo database.	8
14	Procedura per aprire il Query Tool.	9
15	Esecuzione di una query.	9
16	Importazione di un file <code>sql</code> nel Query Tool.	11
17	Selezione del file da importare.	11
18	Query eseguita con successo.	12
19	Schema del database.	12
20	Procedura per la visualizzazione dei dati di una tabella	13
21	Interfaccia per la visualizzazione dei dati	13
22	Procedura per aprire il Query Tool dalla barra degli strumenti.	14
23	Descrizione dei vari pannelli che compongono il Query Tool.	14
24	Esecuzione e visualizzazione dei risultati della prima query.	15

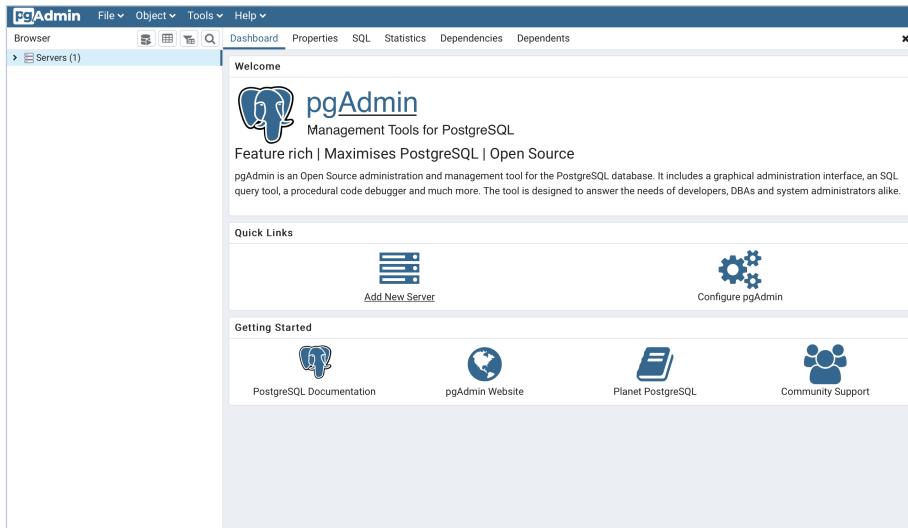


Figura 1: Homepage di pgAdmin4.

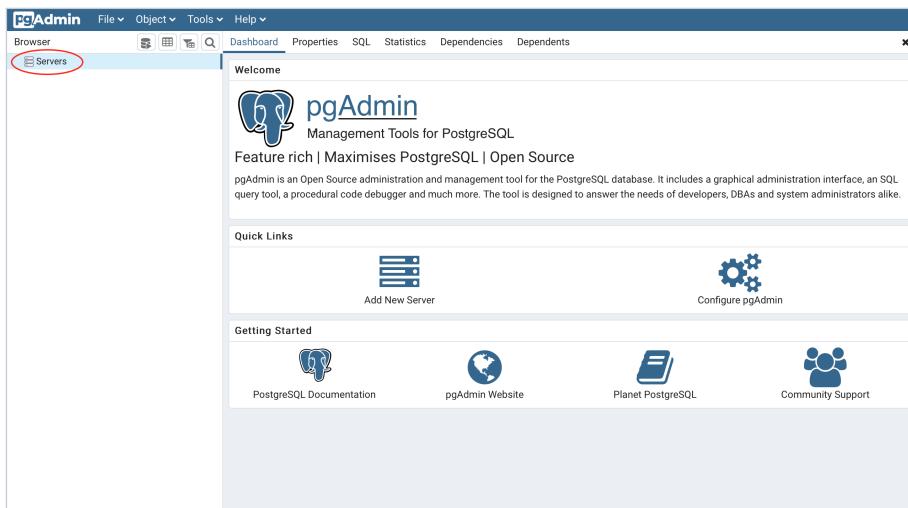


Figura 2: pgAdmin4 senza server connessi.

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 (vedi Figura 1).

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.

Per collegare il server locale manualmente, è necessario creare un nuovo server selezionando [\[Add New Server\]](#) dall'elenco di **Quick Links**, come mostrato in Figura 3.

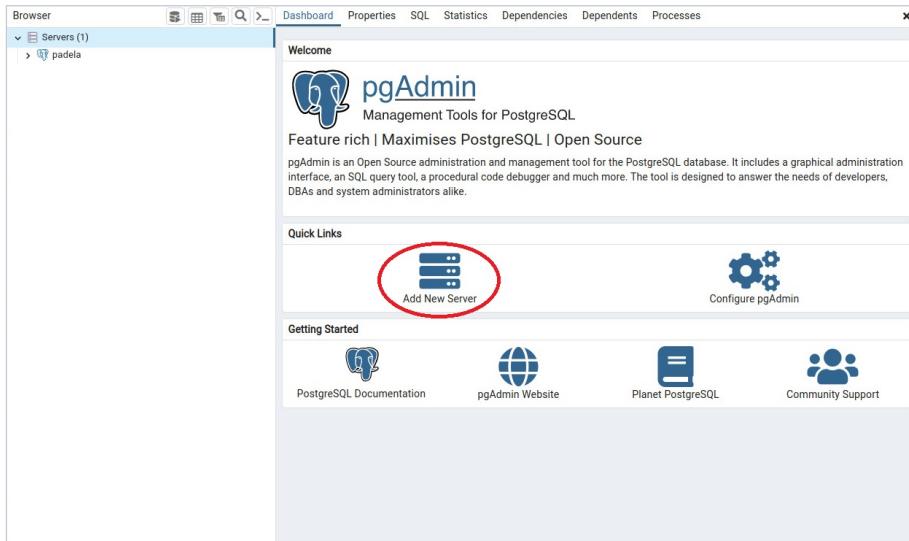


Figura 3: Accesso allo strumento di creazione di un nuovo server

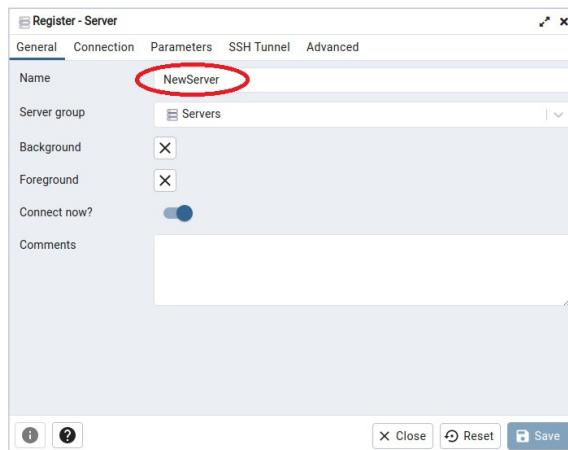


Figura 4: Impostazione del nome del server.

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.

Per collegarsi al server di laboratorio (Figura 5) basterà inserire “postgresql” come “Host name”, indicare come “Maintenance database” e “Username” il proprio nome utente che viene usato per accedere al computer di laboratorio, e infine inserire come “Password” quella recuperata dal link <https://www.studenti.math.unipd.it/account/delivery/> (vedi istruzioni a *Istruzioni per l’accesso a pgAdmin da laboratorio*).

Per accedere in locale dal proprio computer personale, basterà inserire “localhost” in “Host name/address”, come mostrato in Figura 6.

Nota: In Linux e Windows può apparire l’errore in Figura 7.

Soluzione Windows: Inserire nel campo *password* la password impostata nella fase di instal-

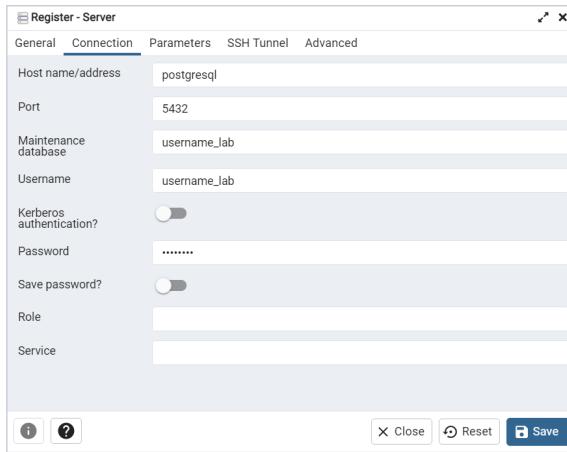


Figura 5: Impostazione dell'indirizzo del server (da Laboratorio).

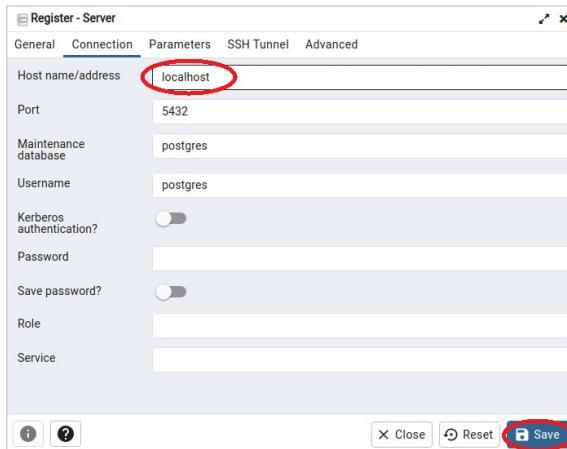


Figura 6: Impostazione dell'indirizzo del server (in locale).

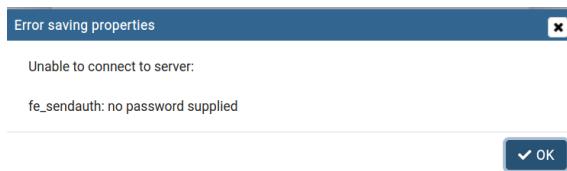


Figura 7: Possibile errore in Windows e Linux

lazione di postgresql.

Soluzione 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:

```

95 # "local" is for Unix domain socket connections only
96 local   all      all                                     peer
97 # IPv4 local connections:
98 host    all      all          127.0.0.1/32            trust
99 # IPv6 local connections:
100 host   all      all          ::1/128               trust

```

Figura 8: Corrispondenti righe del file dopo la modifica.

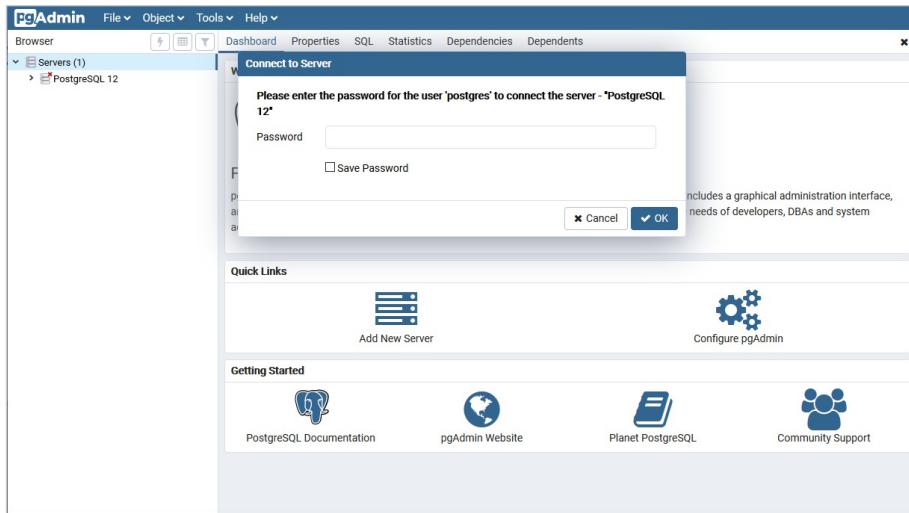


Figura 9: Inserimento della password.

```
01 | sudo gedit /etc/postgresql/15/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 *scram-sha-256* con *trust*. Ottenendo la nuova configurazione mostrata in Figura 8.

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 15” 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 (vedi Figura 9). In alcune installazioni potrebbe non essere impostata alcuna password di default e in questo caso si accederà direttamente alla *Dashboard*.

Questo ci porterà alla schermata mostrata in Figura 10, in cui è presente un database di default chiamato “*postgres*”.

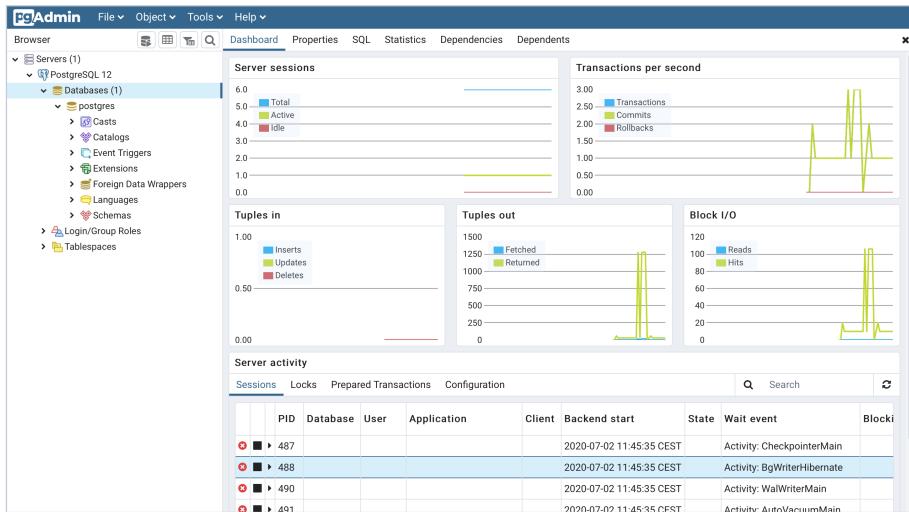


Figura 10: Dashboard iniziale di pgAdmin4

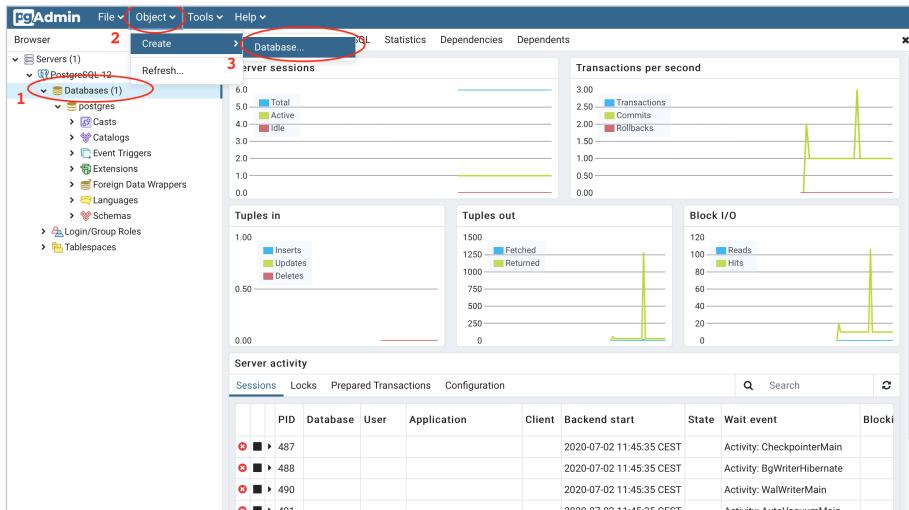


Figura 11: Creazione di un nuovo database.

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** (vedi Figura 11).

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 12). 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 13.

A questo punto abbiamo creato un nuovo database vuoto. Per poterlo popolare con dei dati

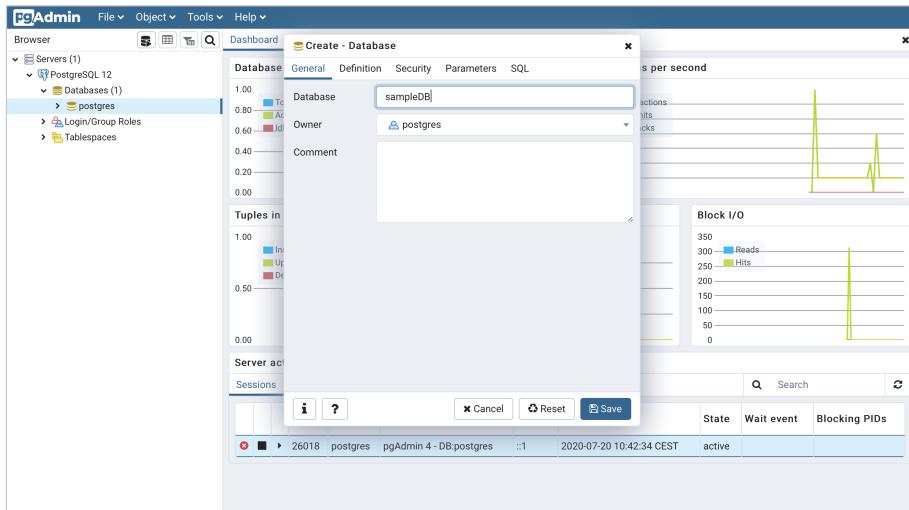


Figura 12: Inserimento del nome del database.

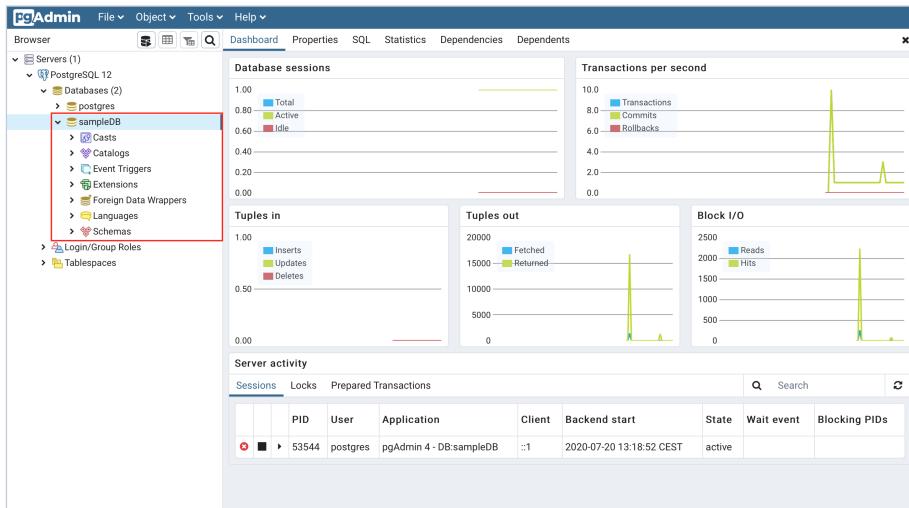


Figura 13: Visualizzazione del nuovo database.

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 14).

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 15.

Definiamo quindi le seguenti tabelle con i relativi campi dati:

```
01 | CREATE TABLE orders (
```

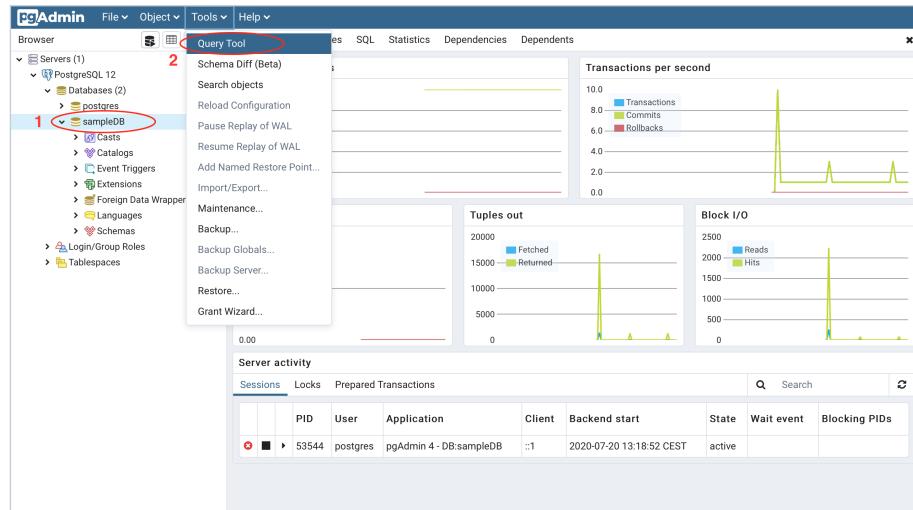


Figura 14: Procedura per aprire il Query Tool.

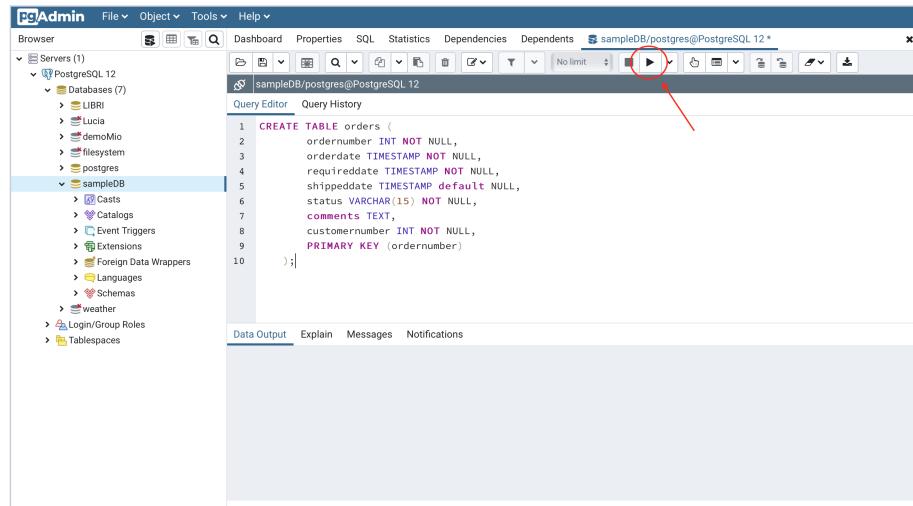


Figura 15: Esecuzione di una query.

```

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 16).

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 17).

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 18.

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 19 illustra tutte le tabelle presenti e le loro relazioni.

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 20. **Nota:** Se le tabelle non sono presenti, si provi ad eseguire il comando **Refresh** prima di **View/Edit Data**.

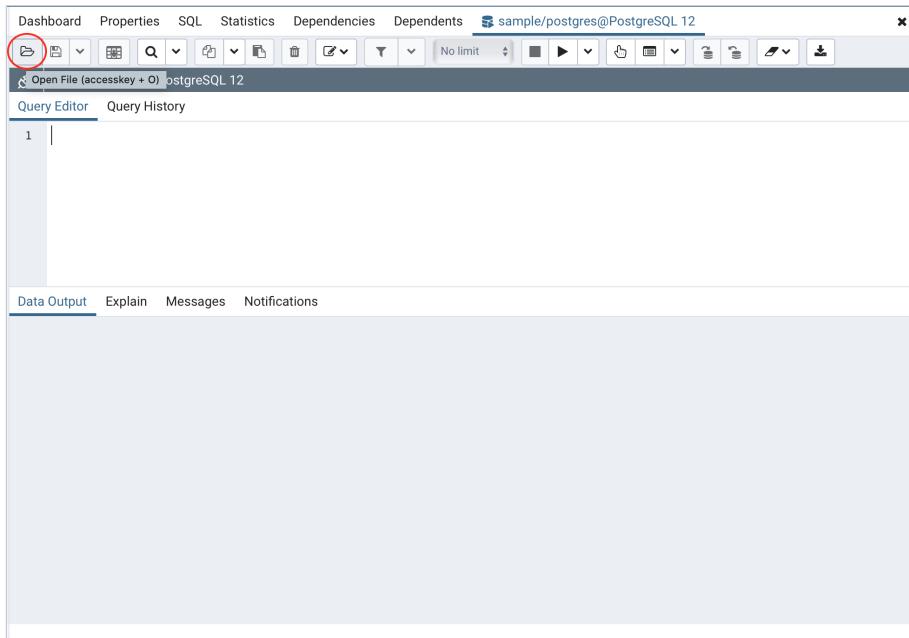


Figura 16: Importazione di un file sql nel Query Tool.

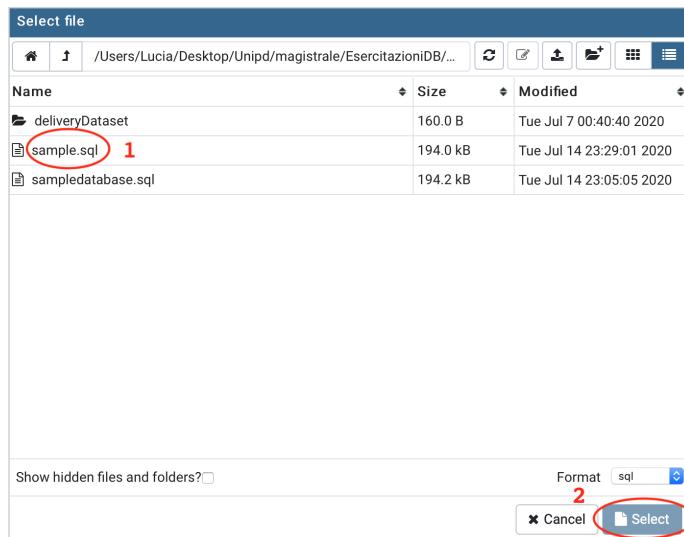


Figura 17: Selezione del file da importare.

Nel pannello principale ((1) in Figura 21) chiamato “Query” viene mostrata la query eseguita automaticamente per selezionare tutti i valori della tabella selezionata.
Nel pannello in basso ((2) in Figura 21) vengono invece illustrati i dati selezionati in forma tabellare. 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

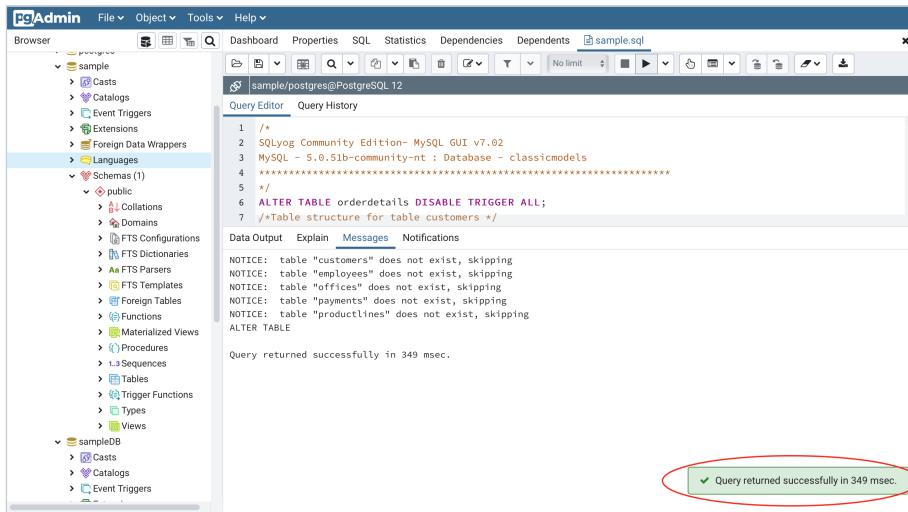


Figura 18: Query eseguita con successo.

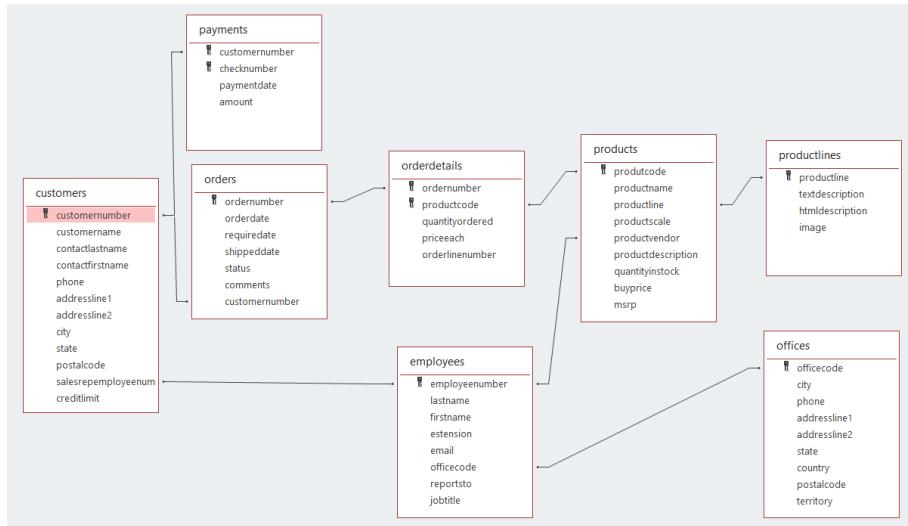


Figura 19: Schema del database.

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**.

L'interfaccia comprende varie sezioni (vedi Figura 23):

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;

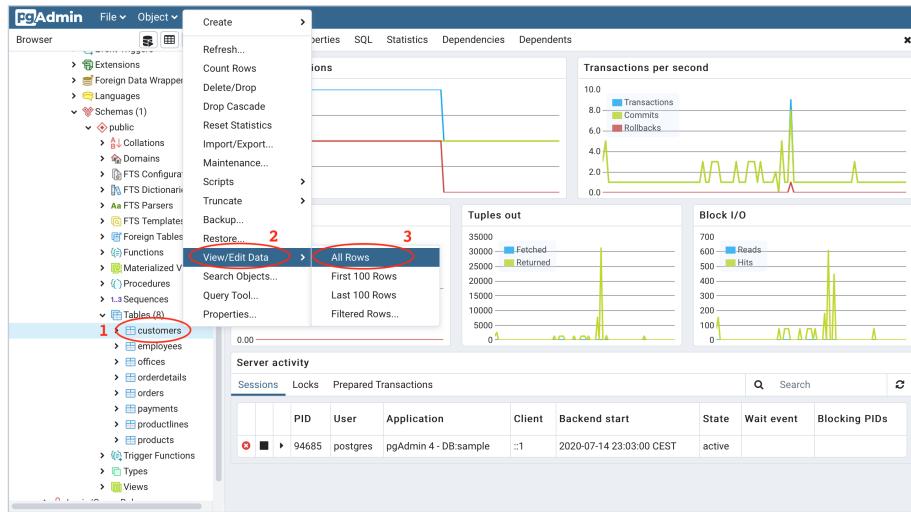


Figura 20: Procedura per la visualizzazione dei dati di una tabella

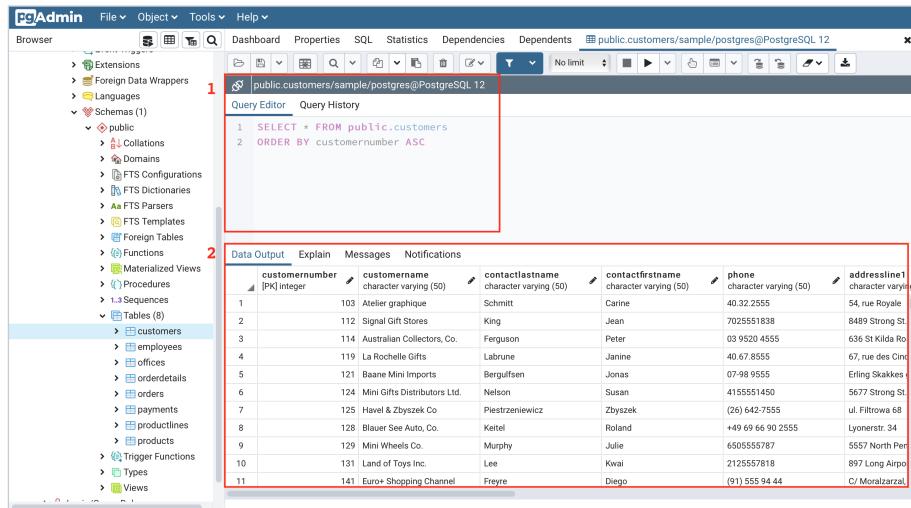


Figura 21: Interfaccia per la visualizzazione dei dati

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.

2.1 Query di base

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

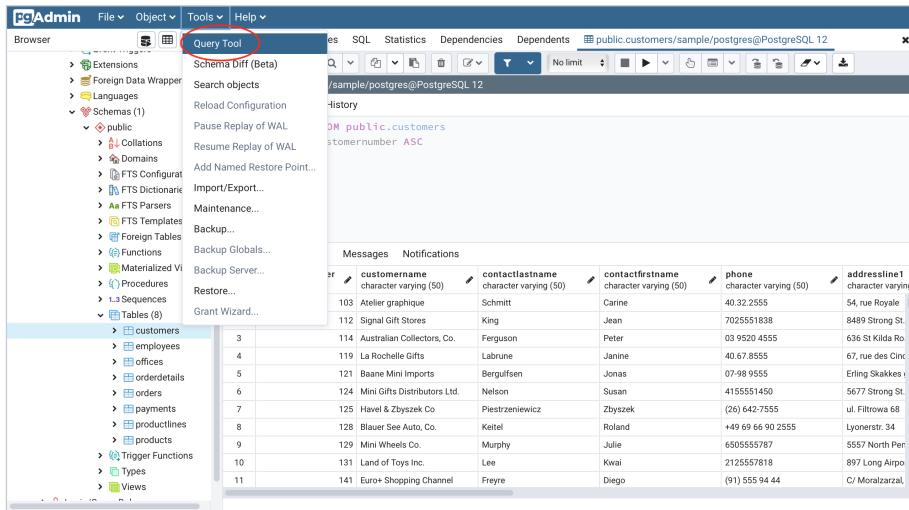


Figura 22: Procedura per aprire il Query Tool dalla barra degli strumenti.

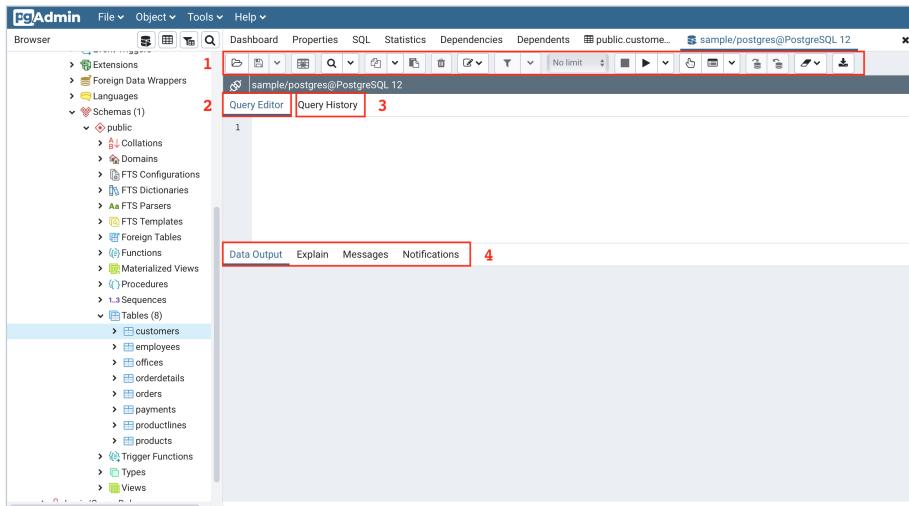


Figura 23: Descrizione dei vari pannelli che compongono il Query Tool.

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

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');
```

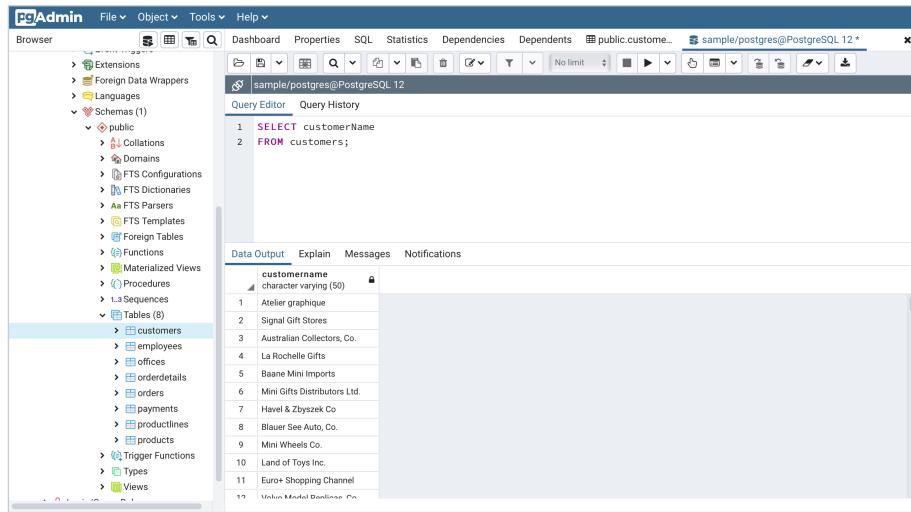


Figura 24: Esecuzione e visualizzazione dei risultati della prima query.

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 2, 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 (Citta, Servizio, Anno, Spesa)
POSIZIONE (Citta, Regione, Abitanti)
```

Utilizzando appunto `Citta` come chiave primaria della tabella `POSIZIONE`, in modo che non sia possibile avere due città con lo stesso nome. Inoltre si vuole che `SERVIZI_SOCIALI.Citta` si riferisca a `POSIZIONE.Citta`. 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”.