Deccan Education Society's (DES)
Pune University, Pune
School of Engineering and Technology
Department of Computer Engineering and Technology
Program: B. Tech in Computer Science and Engineering

Academic Year: 2024-25	**Year:** Second Year	**Semester:** II
PRN No.: 1012412079	**Name: Ratnajeet Patil**	
Subject: Database Management System		
Assignment No.: 6		
Date:		

Lab Assignment: 06

Title: Nested Queries: Create and execute nested SQL queries to address real-world scenarios in any database application.

Execution for:

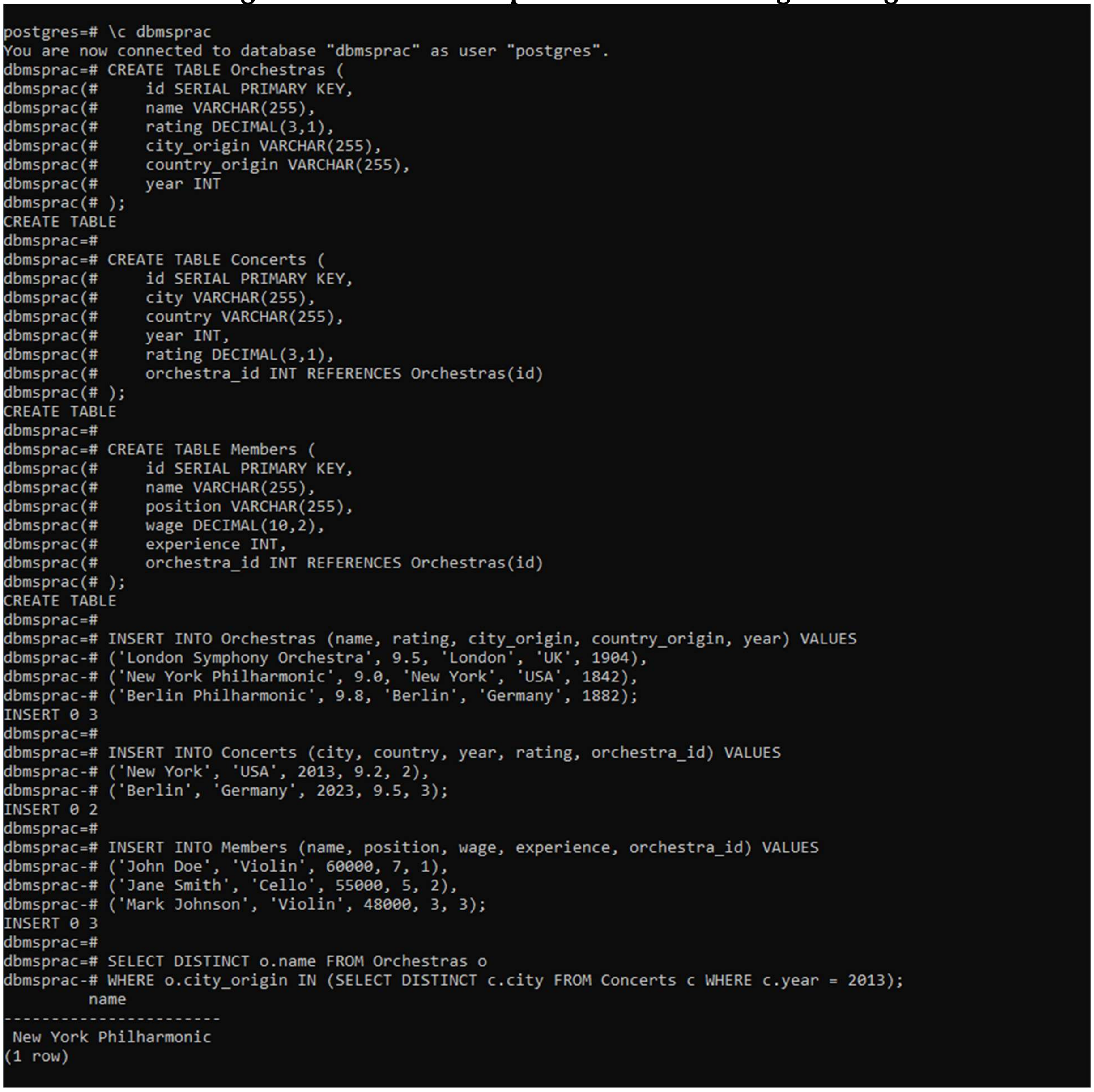
Create following tables and insert sample records in it.

- **Orchestras** (id, name, rating, city_origin, country_origin, year)
- **Concerts** (id, city, country, year, rating, orchestra_id (references the orchestras table)).
- **members** table stores the members of (i.e. musicians playing in) each orchestra. The columns are id, name, position (i.e. the instrument played), wage, experience, and orchestra_id (references the orchestras table).

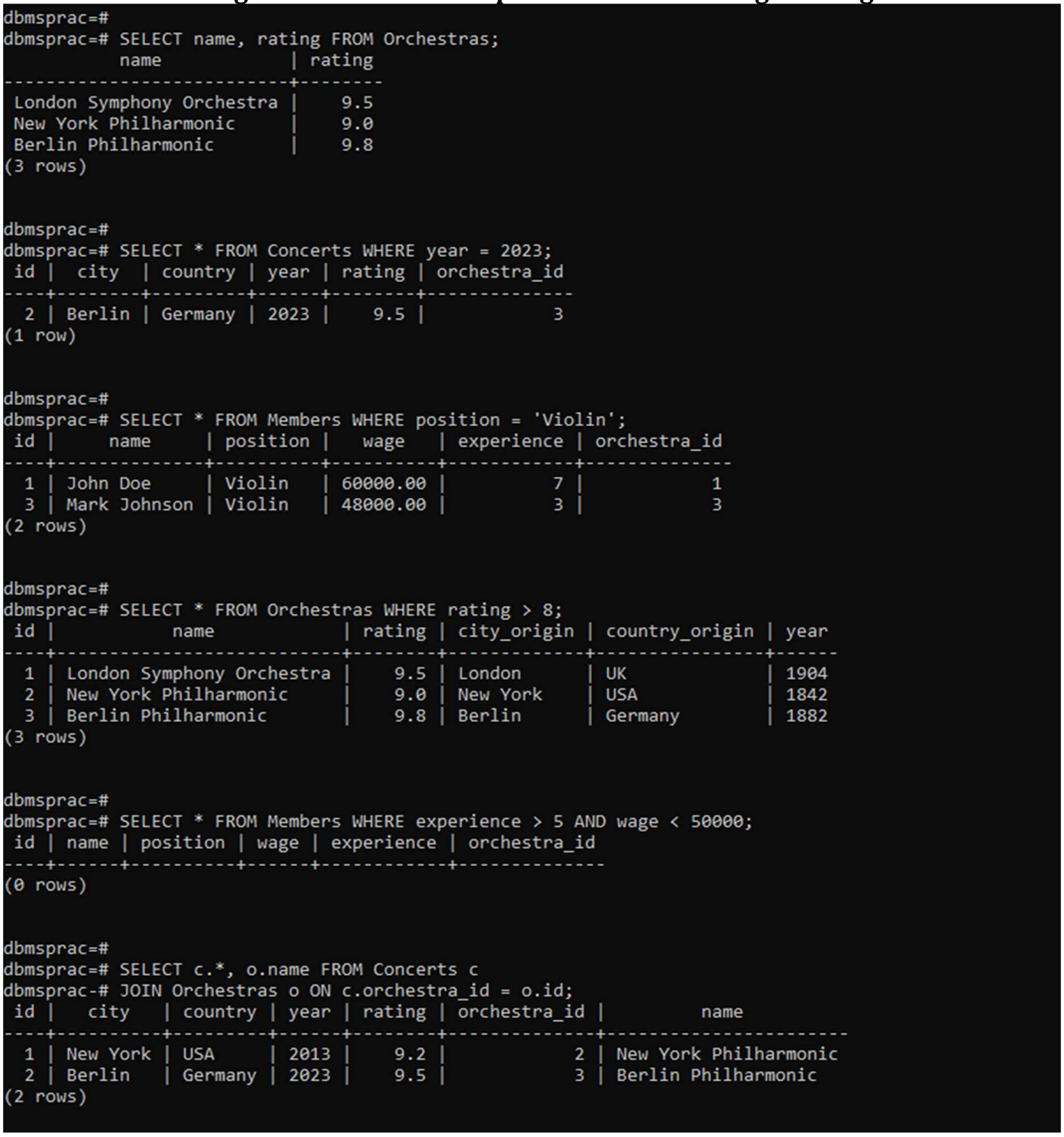
Frame and solve following queries:

1. Select the names of all orchestras that have the same city of origin as any city in which any orchestra performed in 2013.
2. List the names and ratings of all orchestras.
3. Find all concerts held in the year 2023.
4. Display all members who play the violin.
5. Find all orchestras with a rating greater than 8.
6. List members with more than 5 years of experience and a wage less than 50,000.
7. List all concerts along with the name of the orchestra that performed.
8. Display each member's name along with the orchestra name they belong to.
9. Find the average rating of concerts for each orchestra.

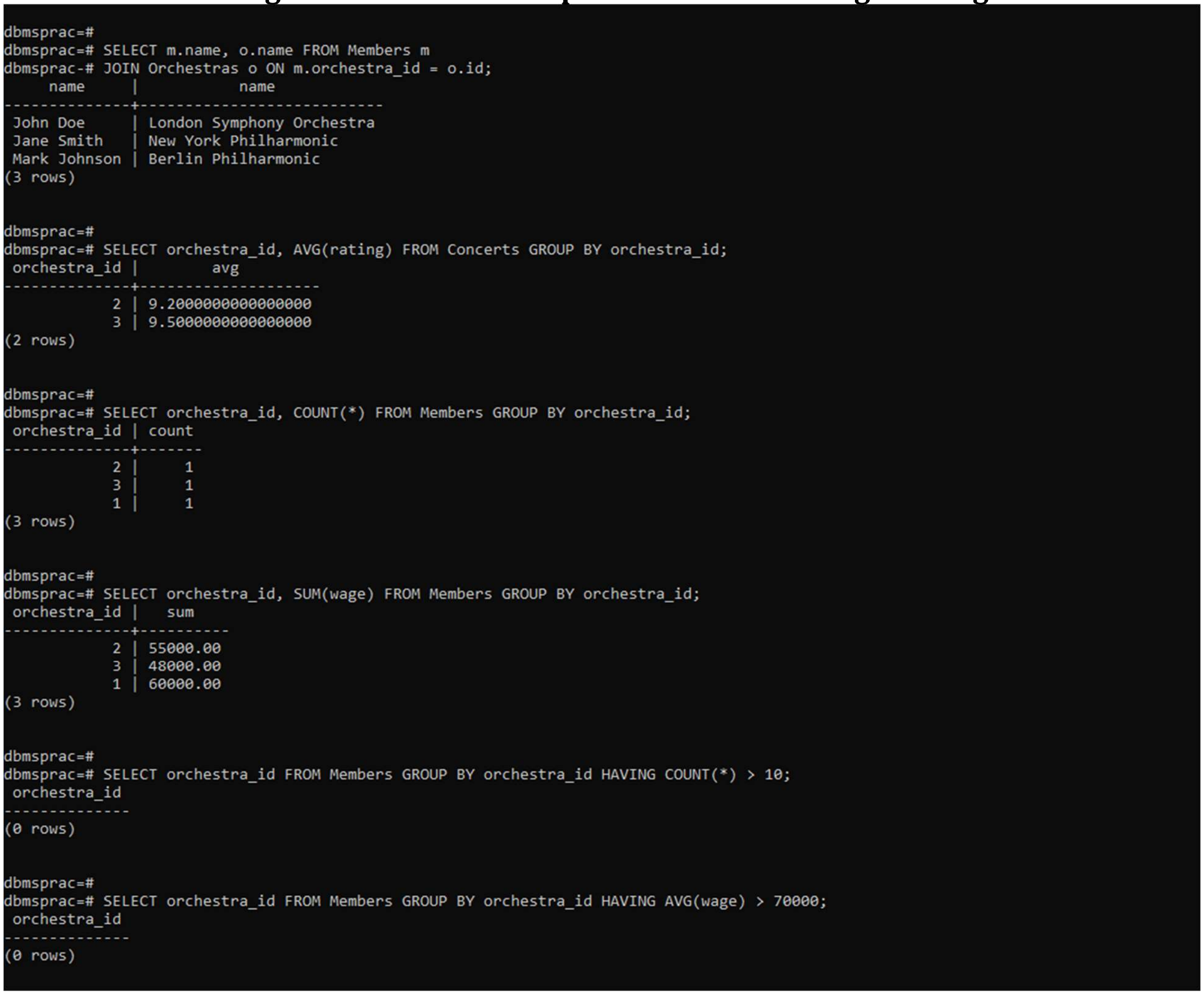
10. Count the number of members in each orchestra.
11. Get the total wage paid to members for each orchestra.
12. List orchestras that have more than 10 members.
13. Find orchestras with an average member wage greater than 70,000.
14. Find orchestras that have never performed a concert.
15. List the names of members who earn more than the average wage of all members.
16. Display the top 5 highest-rated orchestras in descending order.
17. Find the orchestra(s) with the highest average concert rating.
18. List the top 3 orchestras that pay the highest average wage to their members.
19. Find the names of members who earn more than the highest-paid member of any orchestra from Germany.
20. For each orchestra, list the number of concerts held and the total number of members, only for orchestras formed after the year 2000.



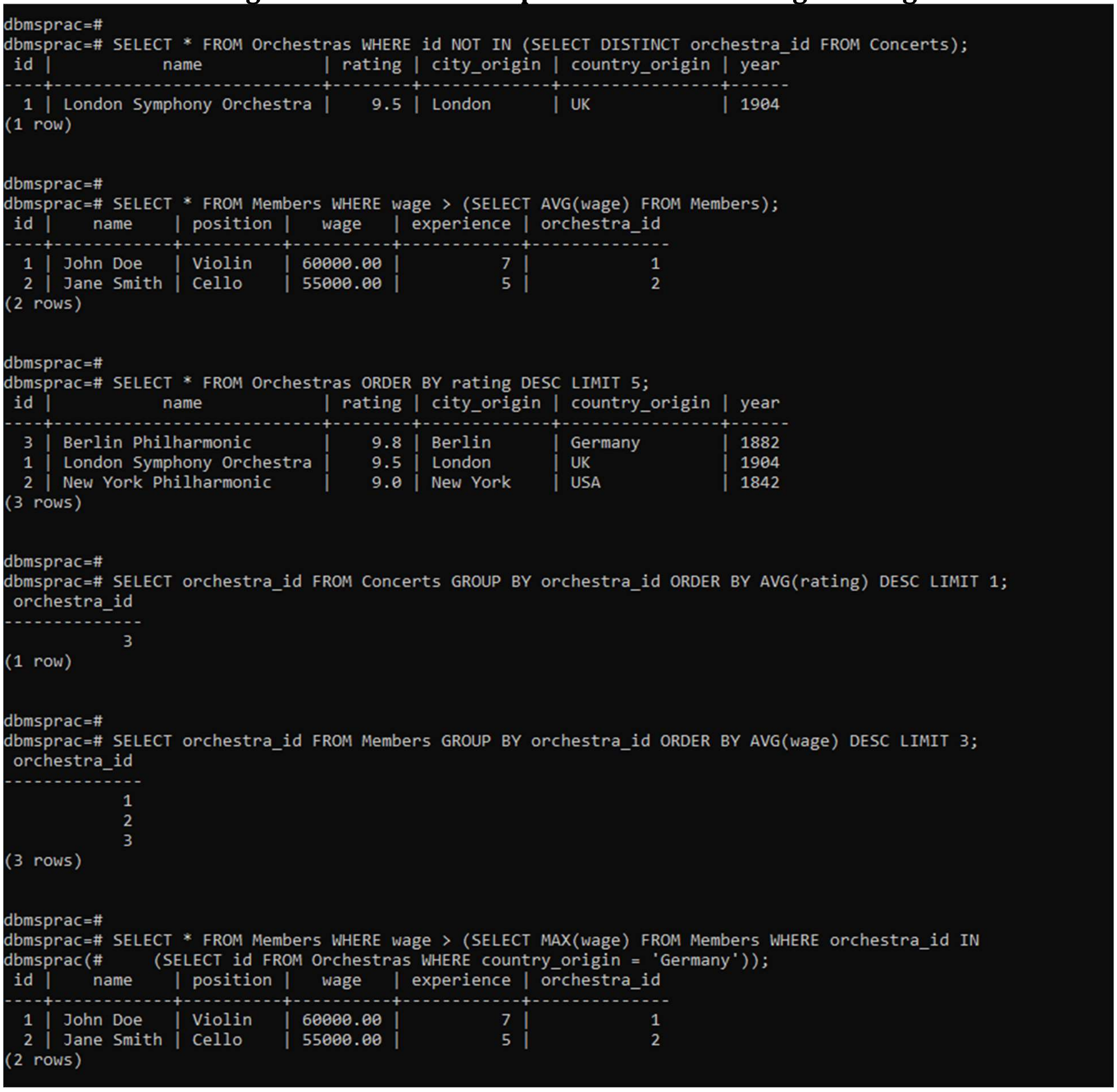
```
postgres=# \c dbmsprac
You are now connected to database "dbmsprac" as user "postgres".
dbmsprac=# CREATE TABLE Orchestras (
dbmsprac(#     id SERIAL PRIMARY KEY,
dbmsprac(#     name VARCHAR(255),
dbmsprac(#     rating DECIMAL(3,1),
dbmsprac(#     city_origin VARCHAR(255),
dbmsprac(#     country_origin VARCHAR(255),
dbmsprac(#     year INT
dbmsprac(# );
CREATE TABLE
dbmsprac=#
dbmsprac=# CREATE TABLE Concerts (
dbmsprac(#     id SERIAL PRIMARY KEY,
dbmsprac(#     city VARCHAR(255),
dbmsprac(#     country VARCHAR(255),
dbmsprac(#     year INT,
dbmsprac(#     rating DECIMAL(3,1),
dbmsprac(#     orchestra_id INT REFERENCES Orchestras(id)
dbmsprac(# );
CREATE TABLE
dbmsprac=#
dbmsprac=# CREATE TABLE Members (
dbmsprac(#     id SERIAL PRIMARY KEY,
dbmsprac(#     name VARCHAR(255),
dbmsprac(#     position VARCHAR(255),
dbmsprac(#     wage DECIMAL(10,2),
dbmsprac(#     experience INT,
dbmsprac(#     orchestra_id INT REFERENCES Orchestras(id)
dbmsprac(# );
CREATE TABLE
dbmsprac=#
dbmsprac=# INSERT INTO Orchestras (name, rating, city_origin, country_origin, year) VALUES
dbmsprac-# ('London Symphony Orchestra', 9.5, 'London', 'UK', 1904),
dbmsprac-# ('New York Philharmonic', 9.0, 'New York', 'USA', 1842),
dbmsprac-# ('Berlin Philharmonic', 9.8, 'Berlin', 'Germany', 1882);
INSERT 0 3
dbmsprac=#
dbmsprac=# INSERT INTO Concerts (city, country, year, rating, orchestra_id) VALUES
dbmsprac-# ('New York', 'USA', 2013, 9.2, 2),
dbmsprac-# ('Berlin', 'Germany', 2023, 9.5, 3);
INSERT 0 2
dbmsprac=#
dbmsprac=# INSERT INTO Members (name, position, wage, experience, orchestra_id) VALUES
dbmsprac-# ('John Doe', 'Violin', 60000, 7, 1),
dbmsprac-# ('Jane Smith', 'Cello', 55000, 5, 2),
dbmsprac-# ('Mark Johnson', 'Violin', 48000, 3, 3);
INSERT 0 3
dbmsprac=#
dbmsprac=# SELECT DISTINCT o.name FROM Orchestras o
dbmsprac-# WHERE o.city_origin IN (SELECT DISTINCT c.city FROM Concerts c WHERE c.year = 2013);
         name
-----------------------
 New York Philharmonic
(1 row)
```



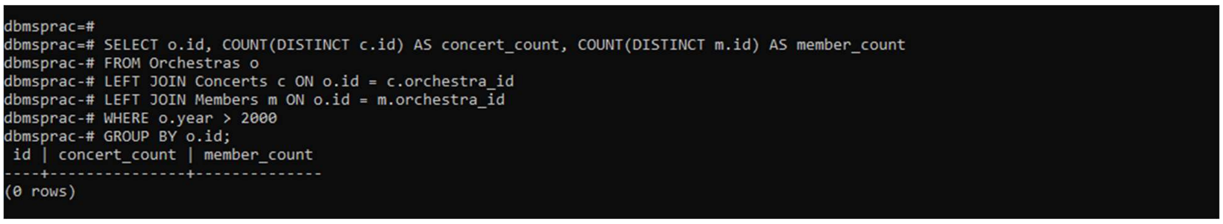
```
dbmsprac=#
dbmsprac=# SELECT name, rating FROM Orchestras;
          name           | rating
-------------------------+--------
 London Symphony Orchestra |    9.5
 New York Philharmonic     |    9.0
 Berlin Philharmonic       |    9.8
(3 rows)


dbmsprac=#
dbmsprac=# SELECT * FROM Concerts WHERE year = 2023;
 id |  city  | country | year | rating | orchestra_id
----+--------+---------+------+--------+-------------
  2 | Berlin | Germany | 2023 |    9.5 |            3
(1 row)


dbmsprac=#
dbmsprac=# SELECT * FROM Members WHERE position = 'Violin';
 id |     name     | position |   wage    | experience | orchestra_id
----+--------------+----------+-----------+------------+-------------
  1 | John Doe     | Violin   | 60000.00 |          7 |            1
  3 | Mark Johnson | Violin   | 48000.00 |          3 |            3
(2 rows)


dbmsprac=#
dbmsprac=# SELECT * FROM Orchestras WHERE rating > 8;
 id |           name            | rating | city_origin | country_origin | year
----+---------------------------+--------+-------------+----------------+------
  1 | London Symphony Orchestra |    9.5 | London      | UK             | 1904
  2 | New York Philharmonic     |    9.0 | New York    | USA            | 1842
  3 | Berlin Philharmonic       |    9.8 | Berlin      | Germany        | 1882
(3 rows)


dbmsprac=#
dbmsprac=# SELECT * FROM Members WHERE experience > 5 AND wage < 50000;
 id | name | position | wage | experience | orchestra_id
----+------+----------+------+------------+-------------
(0 rows)


dbmsprac=#
dbmsprac=# SELECT c.*, o.name FROM Concerts c
dbmsprac-# JOIN Orchestras o ON c.orchestra_id = o.id;
 id |   city   | country | year | rating | orchestra_id |         name
----+----------+---------+------+--------+--------------+-----------------------
  1 | New York | USA     | 2013 |    9.2 |            2 | New York Philharmonic
  2 | Berlin   | Germany | 2023 |    9.5 |            3 | Berlin Philharmonic
(2 rows)
```



```
dbmsprac=#
dbmsprac=# SELECT m.name, o.name FROM Members m
dbmsprac-# JOIN Orchestras o ON m.orchestra_id = o.id;
     name     |           name
--------------+--------------------------
 John Doe     | London Symphony Orchestra
 Jane Smith   | New York Philharmonic
 Mark Johnson | Berlin Philharmonic
(3 rows)


dbmsprac=#
dbmsprac=# SELECT orchestra_id, AVG(rating) FROM Concerts GROUP BY orchestra_id;
 orchestra_id |        avg
--------------+--------------------
            2 | 9.2000000000000000
            3 | 9.5000000000000000
(2 rows)


dbmsprac=#
dbmsprac=# SELECT orchestra_id, COUNT(*) FROM Members GROUP BY orchestra_id;
 orchestra_id | count
--------------+-------
            2 |     1
            3 |     1
            1 |     1
(3 rows)


dbmsprac=#
dbmsprac=# SELECT orchestra_id, SUM(wage) FROM Members GROUP BY orchestra_id;
 orchestra_id |   sum
--------------+----------
            2 | 55000.00
            3 | 48000.00
            1 | 60000.00
(3 rows)


dbmsprac=#
dbmsprac=# SELECT orchestra_id FROM Members GROUP BY orchestra_id HAVING COUNT(*) > 10;
 orchestra_id
--------------
(0 rows)


dbmsprac=#
dbmsprac=# SELECT orchestra_id FROM Members GROUP BY orchestra_id HAVING AVG(wage) > 70000;
 orchestra_id
--------------
(0 rows)
```



```
dbmsprac=#
dbmsprac=# SELECT * FROM Orchestras WHERE id NOT IN (SELECT DISTINCT orchestra_id FROM Concerts);
 id |           name            | rating | city_origin | country_origin | year
----+---------------------------+--------+-------------+----------------+------
  1 | London Symphony Orchestra |    9.5 | London      | UK             | 1904
(1 row)


dbmsprac=#
dbmsprac=# SELECT * FROM Members WHERE wage > (SELECT AVG(wage) FROM Members);
 id |    name    | position |   wage   | experience | orchestra_id
----+------------+----------+----------+------------+--------------
  1 | John Doe   | Violin   | 60000.00 |          7 |            1
  2 | Jane Smith | Cello    | 55000.00 |          5 |            2
(2 rows)


dbmsprac=#
dbmsprac=# SELECT * FROM Orchestras ORDER BY rating DESC LIMIT 5;
 id |           name            | rating | city_origin | country_origin | year
----+---------------------------+--------+-------------+----------------+------
  3 | Berlin Philharmonic       |    9.8 | Berlin      | Germany        | 1882
  1 | London Symphony Orchestra |    9.5 | London      | UK             | 1904
  2 | New York Philharmonic     |    9.0 | New York    | USA            | 1842
(3 rows)


dbmsprac=#
dbmsprac=# SELECT orchestra_id FROM Concerts GROUP BY orchestra_id ORDER BY AVG(rating) DESC LIMIT 1;
 orchestra_id
--------------
            3
(1 row)


dbmsprac=#
dbmsprac=# SELECT orchestra_id FROM Members GROUP BY orchestra_id ORDER BY AVG(wage) DESC LIMIT 3;
 orchestra_id
--------------
            1
            2
            3
(3 rows)


dbmsprac=#
dbmsprac=# SELECT * FROM Members WHERE wage > (SELECT MAX(wage) FROM Members WHERE orchestra_id IN
dbmsprac(#     (SELECT id FROM Orchestras WHERE country_origin = 'Germany'));
 id |    name    | position |   wage   | experience | orchestra_id
----+------------+----------+----------+------------+--------------
  1 | John Doe   | Violin   | 60000.00 |          7 |            1
  2 | Jane Smith | Cello    | 55000.00 |          5 |            2
(2 rows)
```



```
dbmsprac=#
dbmsprac=# SELECT o.id, COUNT(DISTINCT c.id) AS concert_count, COUNT(DISTINCT m.id) AS member_count
dbmsprac-# FROM Orchestras o
dbmsprac-# LEFT JOIN Concerts c ON o.id = c.orchestra_id
dbmsprac-# LEFT JOIN Members m ON o.id = m.orchestra_id
dbmsprac-# WHERE o.year > 2000
dbmsprac-# GROUP BY o.id;
 id | concert_count | member_count
----+---------------+--------------
(0 rows)
```

FAQs:

1. What do you mean by Nested Queries?

   Nested queries, also known as subqueries, are SQL queries placed inside another query. They are used when a query depends on the results of another query for its execution.

2. What is the order of query execution in Nested Queries?
   - The innermost subquery is executed first.
   - The result of the inner query is passed to the outer query.
   - The outer query then uses this result to complete its execution.

3. What do you mean by sub query?

   A subquery is a SQL query that is embedded within another SQL statement. It can be used in SELECT, FROM, or WHERE clauses to retrieve data that will be used by the main query.