Experiment No : 06

Title: To create nested queries and view for the given Database.

Batch:A3 **Roll No.:16010423099** **Experiment No: 06**

Aim: To create nested queries and view for the given database.

Resources needed: PostgreSQL PgAdmin3

Theory:

Nested subqueries:
in clause:
The in connective tests for the set membership, where the set is a collection of values produced by a select clause.
For example to select details of the books written by r.p.jain and d.perry use

```
select book_id, book_name,price from book where author in(„r.p.jain‟, „d. perry‟,‟godse‟);
```

not in:
This connective tests for absence of the set membership.
For example to select details of the books written by authors other than r.p.jain and d.perry use

```
select book_id, book_name,price from book where author not in(„r.p.jain‟,
„d. perry‟,‟godse‟);
```

all:
this keyword is basically used in set comparison query.
It is used in association with relational operators.
"> all" corresponds to the phrase „greater than all‟.
For example to display details of the book that have price greater than all the books published in year 2000 use.

```
Select book_id, book_name, price from book where price >all (select price from book where
pub_year=‟2000‟);
```

any or some:
These keywords are used with relational operators in where clause of set comparison query.
"=some" is identical to in and "<>some" is identical to not in.
">any " is nothing but „greater than at least one‟.

exists and not exists:
exists is the test for non empty set. It is represented by an expression of the form 'exists (select ……. From …….) '. Such expression evaluates to true only if the result evaluvating the subquery represented by the (select ……. From ……) is non empty.
for example to select names of the books for which order is placed use
select book_name from book where exists(select * from order where book_id=order.book_id);

Views:

Views are virtual tables created from already existing tables by selecting certain columns or certain rows. A view can be created from one or many tables. View allows to,

- Restrict access to the data such that a user can only see limited data instead of complete table.
- Summarize data from various tables which can be used to generate reports.

In PostgrSQL, Views are created using the CREATE VIEW statement given bellow.

CREATE [TEMP | TEMPORARY] VIEW view_name AS SELECT column1, column2.....
FROM table_name WHERE [condition];
For example,
Consider COMPANY table having following records:

```
id | name | age | address | salary
 ----+-------+-----+------------+--------
1 | Paul | 32 | California | 20000
2 | Allen | 25 | Texas | 15000
3 | Teddy | 23 | Norway | 20000
4 | Mark | 25 | Rich-Mond | 65000
5 | David | 27 | Texas | 85000
6 | Kim | 22 | South-Hall | 45000
7 | James | 24 | Houston | 10000
```

Following statement creates a view from COMPANY table.

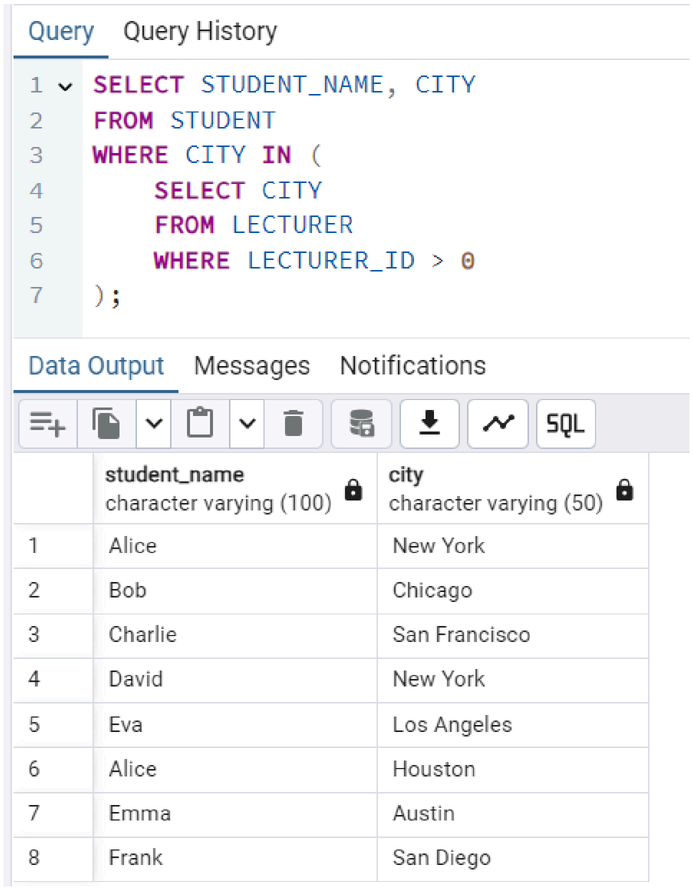
CREATE VIEW COMPANY_VIEW AS SELECT ID, NAME, AGE FROM COMPANY;
Now, query can be written on COMPANY_VIEW in similar way as that of an actual table, as shown below,
SELECT * FROM COMPANY_VIEW;
This would produce the following result:
View can be dropped using “DROP VIEW” statement.

Procedure / Approach /Algorithm / Activity Diagram:

1 Refer different syntax given in theory section and formulate queries consisting of nested sub queries, in , not in, as, group by, having etc clauses and different set operations for your database.
2 Create views from existing tables
Execute SELECT,UPDATE,INSERT statements on views and original table.

Results: (Program printout with output / Document printout as per the format)

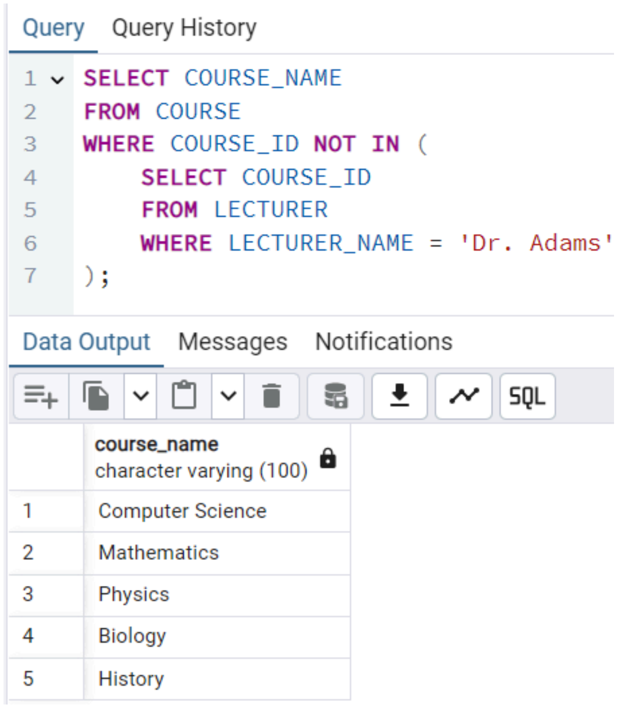
IN:



```sql
SELECT STUDENT_NAME, CITY
FROM STUDENT
WHERE CITY IN (
    SELECT CITY
    FROM LECTURER
    WHERE LECTURER_ID > 0
);
```

	student_name character varying (100)	city character varying (50)
1	Alice	New York
2	Bob	Chicago
3	Charlie	San Francisco
4	David	New York
5	Eva	Los Angeles
6	Alice	Houston
7	Emma	Austin
8	Frank	San Diego

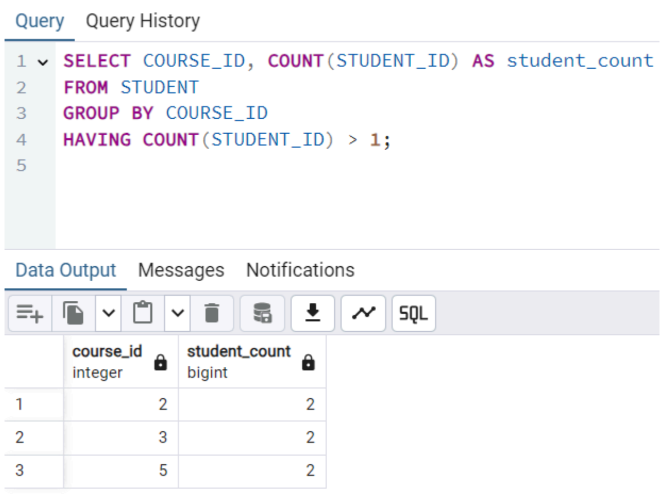
NOT IN:



```sql
SELECT COURSE_NAME
FROM COURSE
WHERE COURSE_ID NOT IN (
    SELECT COURSE_ID
    FROM LECTURER
    WHERE LECTURER_NAME = 'Dr. Adams'
);
```

	course_name character varying (100)
1	Computer Science
2	Mathematics
3	Physics
4	Biology
5	History

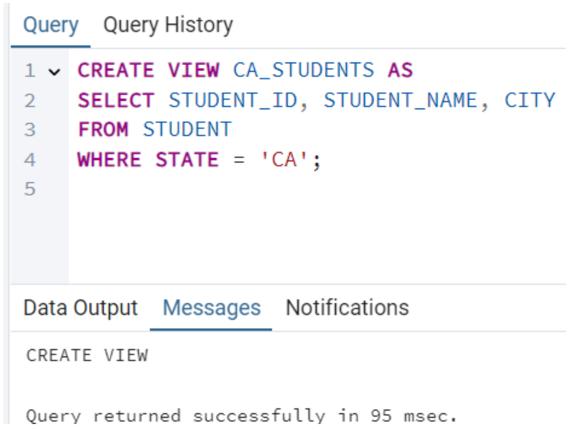
GROUPBY/HAVING:



```sql
SELECT COURSE_ID, COUNT(STUDENT_ID) AS student_count
FROM STUDENT
GROUP BY COURSE_ID
HAVING COUNT(STUDENT_ID) > 1;
```

	course_id integer	student_count bigint
1	2	2
2	3	2
3	5	2

SIMPLE VIEW:



```sql
CREATE VIEW CA_STUDENTS AS
SELECT STUDENT_ID, STUDENT_NAME, CITY
FROM STUDENT
WHERE STATE = 'CA';
```

```
CREATE VIEW

Query returned successfully in 95 msec.
```

Questions:

1. **Explain what are the disadvantages using view on update function.**
   Updating through views can be limited because some views are not updatable if they involve joins, aggregations, or complex subqueries. Additionally, changes made to underlying tables may not reflect correctly if the view logic is complicated.

2 **Can we use where clause with group by clause? Justify your answer**
Yes, the WHERE clause can be used with GROUP BY to filter rows before they are grouped. This ensures only relevant rows are included in the grouping process, optimizing the output.

3 **Can we use having and group by clause without Aggregate functions? Justify your answer**
Technically yes, but the HAVING clause is meant to filter groups, typically based on aggregate results. Without aggregates, its usage becomes redundant, as filtering can usually be achieved with a WHERE clause.

Outcomes:

CO3: Illustrate the concept of security, Query processing, indexing and Normalization for Relational database

Conclusion: (Conclusion to be based on the objectives and outcomes achieved)

Successfully executed the queries with appropriate use of views, WHERE with GROUP BY, and HAVING with or without aggregate functions, ensuring correct filtering and grouping logic on student database.

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of faculty in-charge with date

References:

Books/ Journals/ Websites:

1. Korth, Slberchatz,Sudarshan, :"Database System Concepts", 6th Edition, McGraw – Hill
2. Elmasri and Navathe, " Fundamentals of Database Systems", 5thEdition, PEARSON Education.