Experiment No : 06

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

Batch: B1 **Roll No.: 16010420133** **Experiment No: 06**

Aim: To create nested queries and view for the given database (Virtual lab).

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);

(Autonomous College Affiliated to University of Mumbai)

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:

Nested subqueries:

1. IN:

```
CREATE TABLE employee (
ELoginID text PRIMARY KEY,
Name text UNIQUE,
Password text
);
INSERT into employee values('22', 'SOUMEN','abcdef');
INSERT into employee values('23', 'Men','abcdef');
INSERT into employee values('24', 'Deva','abcd');
INSERT into employee values('25', 'Dev','abc');
SELECT ELoginID FROM employee WHERE Name IN ('SOUMEN','Men','Dev');
```

Output:

	eloginid [PK] text
1	22
2	23
3	25

2. NOT IN:

SELECT ELoginID FROM employee WHERE Name NOT IN ('SOUMEN','Deva','Dev');

Output:

	eloginid [PK] text
1	23

3. ALL:

SELECT Password FROM employee WHERE ELoginID > ALL (SELECT ELoginID FROM employee WHERE Name IN ('SOUMEN'));

Output:

	password text
1	abcdef
2	abcd
3	abc

4. **ANY:**

SELECT Password FROM employee WHERE ELoginID > ANY (SELECT ELoginID FROM employee WHERE Name IN ('Men'));

Output:

	password text
1	abcd
2	abc

5. **EXISTS:**

SELECT Password FROM employee WHERE EXISTS (SELECT ELoginID FROM employee WHERE Name IN ('Dev'));

	password text
1	abcdef
2	abcdef
3	abcd
4	abc

Output:

6. **NOT EXISTS:**

SELECT Password FROM employee WHERE NOT EXISTS (SELECT ELoginID FROM employee WHERE Name IN ('Dev'));

Output:

Data Output | Explain | Mess

	password text

Views:

1. SELECT:

CREATE VIEW employee_view AS SELECT ELoginID, Name FROM employee;
SELECT * FROM employee_view;

Output:

	eloginid text	name text
1	22	SOUMEN
2	23	Men
3	24	Deva
4	25	Dev

2. UPDATE:

DROP VIEW employee_view;
CREATE VIEW employee_view AS SELECT ELoginID, Name, Password FROM employee;
UPDATE employee_view SET Password = 'xyz';
SELECT * FROM employee_view;

Output:

Data Output | Explain | Messages | Notifications

	eloginid text	name text	password text
1	22	SOUMEN	xyz
2	23	Men	xyz
3	24	Deva	xyz
4	25	Dev	xyz

3. INSERT:

```
INSERT into employee_view values('26', 'D','abc');
SELECT * FROM employee_view;
```

Output:

Questions:

1. Explain what are the disadvantages using view on update function.

   A)

   The disadvantage of using views on update function is that allowing updates in views with joins is likely to confuse users because it's not intuitive that they can't update different columns on the same row. After all, it looks like the same table to them. If we want users to be able to use views to update data, base the view on a single table, or use a stored procedure to perform the update. In a complex view, if our update statement affects one base table, then the update succeeded but it may or may not update the data correctly. if our update statement affects more than one table, then the update failed and we will get an error message

2. Can we use where clause with group by clause? Justify your answer

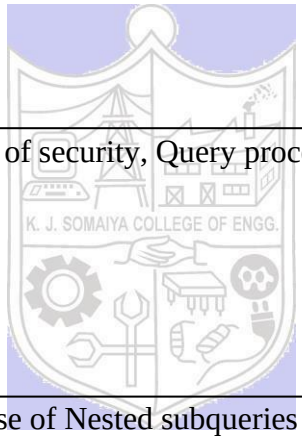
   A)

   Yes, we can use where clause with group by clause. The WHERE clause is used before GROUP BY , because it makes more sense. The filter specified in the WHERE clause is used before grouping. After grouping, you can have a HAVING clause, which is similar to WHERE , except you can filter by aggregate values as well.

3.Can we use having and group by clause without Aggregate functions? Justify your answer.

A)

Yes, we can use having and group by clause without Aggregate functions. When we use GROUP BY and HAVING clause in the SELECT statement without using aggregate functions then it would behave like DISTINCT clause.



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

Conclusion: We illustrated the use of Nested subqueries and Views in the database and implemented the same in our database 'Restaurant Management System' using PostgreSQL pgadmin4 server.

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.

(Autonomous College Affiliated to University of Mumbai)