K. J. Somaiya College of Engineering, Mumbai-77
(Autonomous College Affiliated to University of Mumbai)

Batch: A1 Roll No.: 16010120015
Dikshita 16010120006
Yash 16010120015
Jinay 16010120018
Experiment / assignment / tutorial No. 4

Title: DML – select, insert, update and delete

1.Group by, having clause, aggregate functions, Set Operations
2.Nested queries : AND,OR,NOT, IN, NOT IN, Exists, Not Exists, Between, Like, Alias, ANY,ALL,DISTINCT
3. Update
4. Delete

Objective: To perform various DML Operations and executing nested queries with various clauses.

Expected Outcome of Experiment:
CO 3: Use SQL for Relational database creation, maintenance and query processing

Books/ Journals/ Websites referred:
1. Dr. P.S. Deshpande, SQL and PL/SQL for Oracle 10g.Black book, Dreamtech Press
2. www.db-book.com
3. Korth, Slberchatz, Sudarshan : “Database Systems Concept”, 5th Edition , McGraw Hill
4. Elmasri and Navathe,”Fundamentals of database Systems”, 4th Edition PEARSON Education

Resources used: Postgres

Theory:

Select: The SQL **SELECT** statement is used to fetch the data from a database table which returns this data in the form of a result table. These result tables are called result-sets.

Syntax

The basic syntax of the SELECT statement is as follows –

SELECT column1, column2, columnN FROM table_name;

Here, column1, column2... are the fields of a table whose values you want to fetch. If you want to fetch all the fields available in the field, then you can use the following syntax.

SELECT * FROM table_name;

The following code is an example, which would fetch the ID, Name and Salary fields of the customers available in CUSTOMERS table.

SQL> SELECT ID, NAME, SALARY FROM CUSTOMERS;

Insert: The SQL **INSERT INTO** Statement is used to add new rows of data to a table in the database.

Syntax

There are two basic syntaxes of the INSERT INTO statement which are shown below.

INSERT INTO TABLE_NAME (column1, column2, column3,...columnN)

VALUES (value1, value2, value3,...valueN);

Example

The following statements would create record in the CUSTOMERS table.

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (1, 'Ramesh', 32, 'Ahmedabad', 2000.00);

Update: The SQL **UPDATE** Query is used to modify the existing records in a table. You can use the WHERE clause with the UPDATE query to update the selected rows, otherwise all the rows would be affected.

Syntax:

The basic syntax of the UPDATE query with a WHERE clause is as follows −

UPDATE table_name

SET column1 = value1, column2 = value2...., columnN = valueN

WHERE [condition];

You can combine N number of conditions using the AND or the OR operators.

The following query will update the ADDRESS for a customer whose ID number is 6 in the table.

```
SQL> UPDATE CUSTOMERS
SET ADDRESS = 'Pune'
WHERE ID = 6;
```

Delete: The SQL DELETE Query is used to delete the existing records from a table.

You can use the WHERE clause with a DELETE query to delete the selected rows, otherwise all the records would be deleted.

Syntax

The basic syntax of the DELETE query with the WHERE clause is as follows −

```
DELETE FROM table_name

WHERE [condition];
```

The following code has a query, which will DELETE a customer, whose ID is 6.

```
SQL> DELETE FROM CUSTOMERS
WHERE ID = 6;
```

Clauses and Operators

1. **Group by clause:** These are circumstances where we would like to apply the aggregate functions to a single set of tuples but also to a group of sets of tuples we would like to specify this wish in SQL using the group by clause. The attributes or attributes given by the group by clause are used to form groups. Tuples with the same value on all attributes in the group by clause placed in one group.

Example:.

```
Select<attribute_name,avg(<attribute_name>)as
<new_attribute_name>l From <table_name>
Group by <attribute_name>
```

Example: select designation, sum(salary) as total_salary from employee group by Designation;

2. Having clause: A having clause is like a where clause but only applies only to groups as a whole whereas the where clause applies to the individual rows. A query can contain both where clause and a having clause. In that case

a. The where clause is applied first to the individual rows in the tables or table structures objects in the diagram pane. Only the rows that meet the conditions in the where clause are grouped.

b. The having clause is then applied to the rows in the result set that are produced by grouping. Only the groups that meet the having conditions appear in the query output.

Example:

select dept_no from EMPLOYEE group_by dept_no
having avg (salary) >=all (select avg (salary)
from EMPLOYEE group by dept_no);

3. Aggregate functions: Aggregate functions such as SUM, AVG, count, count (*), MAX and MIN generate summary values in query result sets. An aggregate functions (with the exception of count (*) processes all the selected values in a single column to produce a single result value

Example: select dept_no,count (*)
from EMPLOYEE group by dept_no;

Example: select max (salary)as maximum from EMPLOYEE;

Example: select sum (salary) as total_salary from EMPLOYEE;

Example: Select min (salary) as minsal from EMPLOYEE;

4. Exists and Not Exists: Subqueries introduced with exists and not queries can be used for two set theory operations: Intersection and Difference. The intersection of two sets contains all elements that belong to both of the original sets. The difference contains elements that belong to only first of the two sets.

Example:

Select *from DEPARTMENT
where exists(select * from PROJECT
where DEPARTMENT.dept_no = PROJECT.dept_no) ;

5. **IN and Not In**: SQL allows testing tuples for membership in a relation. The "in" connective tests for set membership where the set is a collection of values produced by select clause. The "not in" connective tests for the absence of set membership. The in and not in connectives can also be used on enumerated sets.

Example:

1. Select fname, mname, lname from employee where designation In ("ceo","manager","hod","assistant")

2. Select fullname from department where relationship not in("brother");

6. Between: The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates. The BETWEEN operator is inclusive. Begin and end values are included.

Syntax:

SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value1 AND value2;

Example:

SELECT * FROM Products WHERE Price BETWEEN 10 AND 20;

7. LIKE: The LIKE **operator** is used in a WHERE clause to search for a specified pattern in a column.

There are two wildcards used in conjunction with the LIKE operator:

- % - The percent sign represents zero, one, or multiple characters
- _ - The underscore represents a single character

Syntax: SELECT *column1, column2, ...*
FROM *table_name*
WHERE *columnN* LIKE *pattern*

Examples:

1. selects all customers with a CustomerName starting with "a":

```
SELECT * FROM Customers
WHERE CustomerName LIKE 'a%';
```

2. selects all customers with a CustomerName that have "r" in the second position:

```
SELECT * FROM Customers
WHERE CustomerName LIKE '_r%';
```

8. Alias: The use of table aliases is to rename a table in a specific SQL statement. The renaming is a temporary change and the actual table name does not change in the database. The column aliases are used to rename a table's columns for the purpose of a particular SQL query.

The basic syntax of a **table** alias is as follows.

```
SELECT column1, column2....

FROM table_name AS alias_name

WHERE [condition];
```

The basic syntax of a **column** alias is as follows.

```
SELECT column_name AS alias_name

FROM table_name

WHERE [condition];
```

Example:

```
SELECT C.ID, C.NAME, C.AGE, O.AMOUNT

FROM CUSTOMERS AS C, ORDERS AS O

WHERE  C.ID = O.CUSTOMER_ID;
```

9. Distinct: The SELECT DISTINCT statement is used to return only distinct (different) values.

Syntax: SELECT DISTINCT *column1*, *column2, ...*
FROM *table_name*;

Example: SELECT DISTINCT Country FROM Customers;

10. Set Operations: 4 different types of SET operations, along with example:

1. UNION
2. UNION ALL
3. INTERSECT
4. MINUS

UNION Operation

UNION is used to combine the results of two or more SELECT statements. However it will eliminate duplicate rows from its resultset. In case of union, number of columns and datatype must be same in both the tables, on which UNION operation is being applied.

Query: SELECT * FROM First

UNION

SELECT * FROM Second;

UNION ALL

This operation is similar to Union. But it also shows the duplicate rows.

Query: SELECT * FROM First

UNION ALL

SELECT * FROM Second;

INTERSECT

Intersect operation is used to combine two SELECT statements, but it only retuns the records which are common from both SELECT statements. In case of **Intersect** the number of columns and datatype must be same.

Query: SELECT * FROM First

INTERSECT

SELECT * FROM Second;

MINUS

The Minus operation combines results of two SELECT statements and return only those in the final result, which belongs to the first set of the result.

Query: SELECT * FROM First

MINUS

SELECT * FROM Second;

11. ANY and ALL: The ANY and ALL operators are used with a WHERE or HAVING clause. The ANY operator returns true if any of the subquery values meet the condition. The ALL operator returns true if all of the subquery values meet the condition.

ANY

SELECT *column_name(s)*
FROM *table_name*
WHERE *column_name operator* ANY
(SELECT *column_name* FROM *table_name* WHERE *condition*);

Example: The following SQL statement returns TRUE and lists the productnames if it finds ANY records in the OrderDetails table that quantity = 10:

SELECT ProductName
FROM Products
WHERE ProductID
= ANY (SELECT ProductID FROM OrderDetails WHERE Quantity = 10);

ALL

SELECT *column_name(s)*
FROM *table_name*
WHERE *column_name operator* ALL
(SELECT *column_name* FROM *table_name* WHERE *condition*);

Example: The following SQL statement returns TRUE and lists the productnames if ALL the records in the OrderDetails table has quantity = 10:

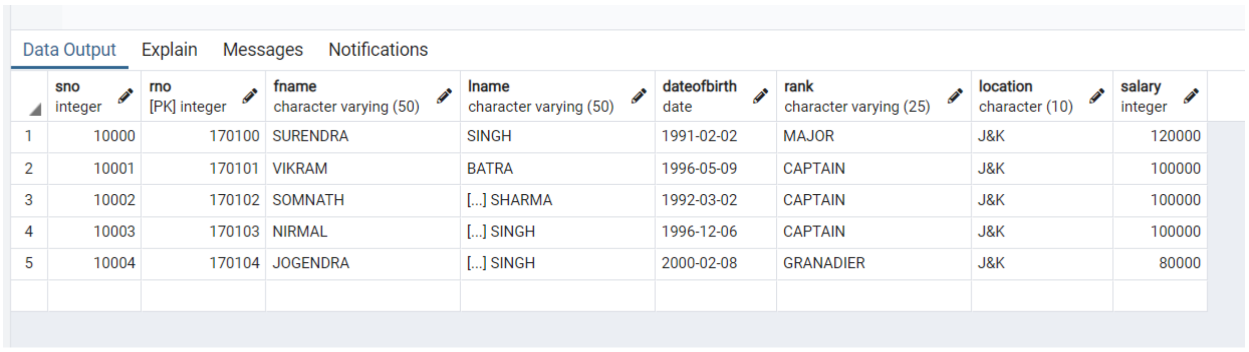
SELECT ProductName
FROM Products
WHERE ProductID
= ALL (SELECT ProductID FROM OrderDetails WHERE Quantity = 10);

Implementation details

```
CREATE TABLE SOLDIERS (SNO INT NOT NULL ,

RNO INT NOT NULL PRIMARY KEY,

FNAME varchar(50) NOT NULL ,

LNAME varchar(50) NOT NULL ,

DateOfBirth DATE NULL ,

RANK varchar(25) NOT NULL,

LOCATION CHAR(10) NOT NULL ,

SALARY INT NOT NULL

);


INSERT INTO SOLDIERS
(SNO, RNO, FNAME, LNAME, DateOfBirth, RANK, location, SALARY)
Values
('10000', '170100', 'SURENDRA' ,'SINGH','02/02/1991', 'MAJOR', 'J&K', '120000');
INSERT INTO SOLDIERS
(SNO, RNO, FNAME, LNAME, DateOfBirth, RANK, location, SALARY)
Values
('10000', '170100', 'SURENDRA' ,'SINGH','02/02/1991', 'MAJOR', 'J&K', '120000');

INSERT INTO SOLDIERS
(SNO, RNO, FNAME, LNAME, DateOfBirth, RANK, location, SALARY)
Values
('10001', '170101', 'VIKRAM','BATRA','05/09/1996', 'CAPTAIN', 'J&K', '100000');

INSERT INTO SOLDIERS
(SNO, RNO, FNAME, LNAME, DateOfBirth, RANK, location, SALARY)
Values
('10002', '170102', 'SOMNATH',' SHARMA ','03/02/1992', 'CAPTAIN', 'J&K', '100000');



INSERT INTO SOLDIERS
```

```
(SNO, RNO, FNAME, LNAME, DateOfBirth, RANK, location, SALARY)
Values
('10003', '170103', 'NIRMAL',' SINGH ','12/06/1996', 'CAPTAIN', 'J&K', '100000');

INSERT INTO SOLDIERS
(SNO, RNO, FNAME, LNAME, DateOfBirth, RANK, location, SALARY)
Values
('10004', '170104', 'JOGENDRA',' SINGH','02/08/2000', 'GRANADIER', 'J&K', '80000');
```

SOLDIER TABLE CREATED AND VALUES ARE INSERTED

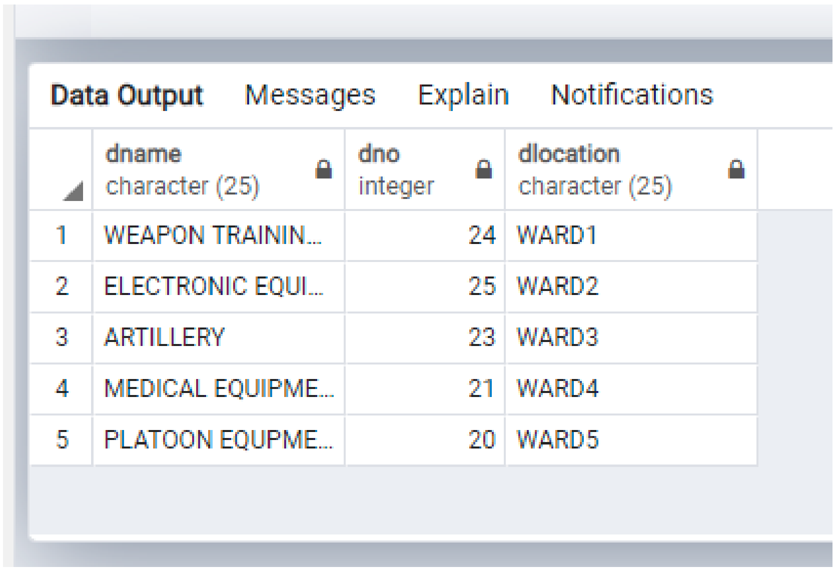


Data Output | Explain | Messages | Notifications

	sno integer	rno [PK] integer	fname character varying (50)	lname character varying (50)	dateofbirth date	rank character varying (25)	location character (10)	salary integer
1	10000	170100	SURENDRA	SINGH	1991-02-02	MAJOR	J&K	120000
2	10001	170101	VIKRAM	BATRA	1996-05-09	CAPTAIN	J&K	100000
3	10002	170102	SOMNATH	[...] SHARMA	1992-03-02	CAPTAIN	J&K	100000
4	10003	170103	NIRMAL	[...] SINGH	1996-12-06	CAPTAIN	J&K	100000
5	10004	170104	JOGENDRA	[...] SINGH	2000-02-08	GRANADIER	J&K	80000

```
INSERT INTO DEPARTMENT
( DNAME, DNO, LOCATION)
Values
('WEAPON TRAINING','24', 'WARD1')

INSERT INTO DEPARTMENT
( DNAME, DNO, LOCATION)
Values
('ELECTRONIC EQUIPMENTS ','25', 'WARD2')

INSERT INTO DEPARTMENT
( DNAME, DNO, LOCATION)
Values
('ARTILLERY','23', 'WARD3')

INSERT INTO DEPARTMENT
( DNAME, DNO, LOCATION)
Values
('MEDICAL EQUIPMENTS ','21', 'WARD4')

INSERT INTO DEPARTMENT
( DNAME, DNO, LOCATION)
```

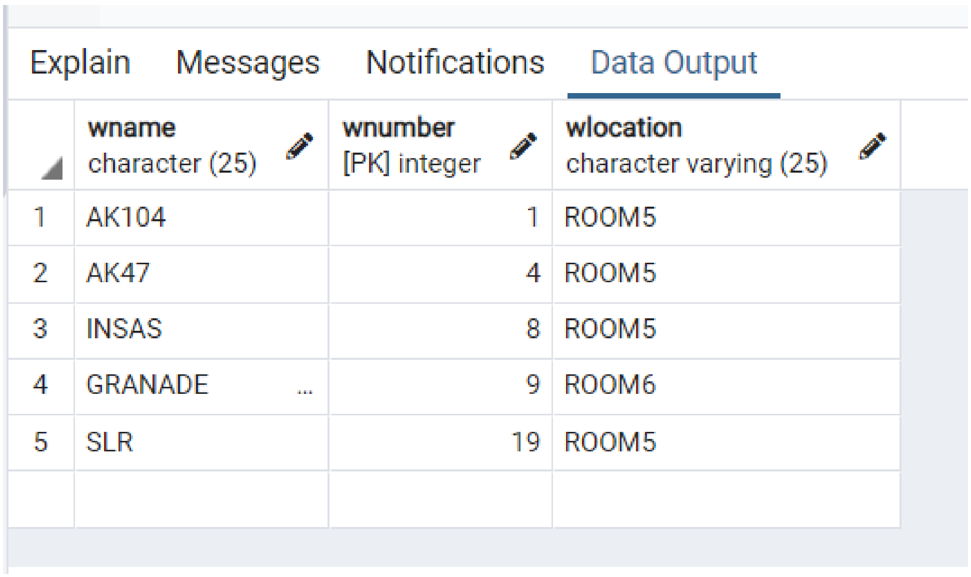
```
Values
('PLATOON EQUPMENTS','20', 'WARD5')
```



Data Output Messages Explain Notifications

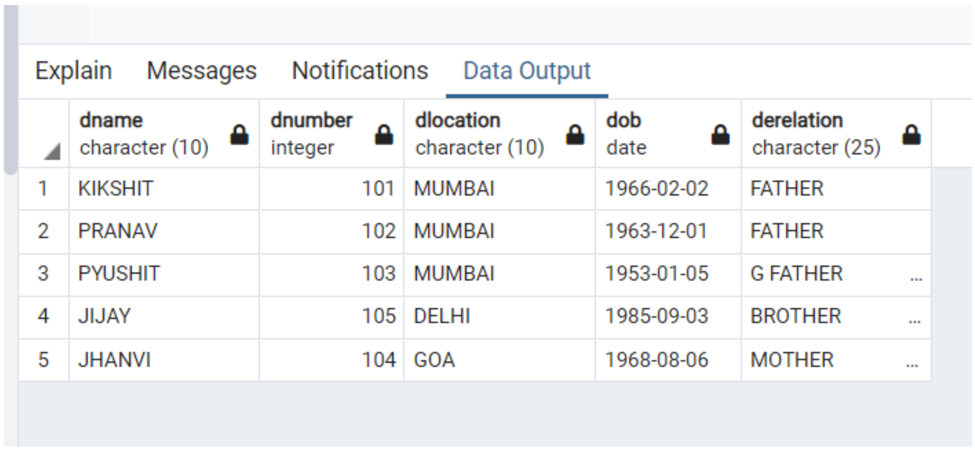
	dname character (25)	dno integer	dlocation character (25)
1	WEAPON TRAININ...	24	WARD1
2	ELECTRONIC EQUI...	25	WARD2
3	ARTILLERY	23	WARD3
4	MEDICAL EQUIPME...	21	WARD4
5	PLATOON EQUPME...	20	WARD5

```
INSERT INTO WEAPONS
( WNAME, WNUMBER,WLOCATION)
Values
('AK47','04', 'ROOM5' )

INSERT INTO WEAPONS
( WNAME, WNUMBER,WLOCATION)
Values
('AK104','01', 'ROOM5' )

INSERT INTO WEAPONS
( WNAME, WNUMBER,WLOCATION)
Values
('GRANADE','09', 'ROOM6' )

INSERT INTO WEAPONS
( WNAME, WNUMBER,WLOCATION)
Values
('SLR','19', 'ROOM5' )

INSERT INTO WEAPONS
( WNAME, WNUMBER,WLOCATION)
Values
('INSAS','08', 'ROOM5' )
```



	wname character (25)	wnumber [PK] integer	wlocation character varying (25)
1	AK104	1	ROOM5
2	AK47	4	ROOM5
3	INSAS	8	ROOM5
4	GRANADE ...	9	ROOM6
5	SLR	19	ROOM5

```
INSERT INTO DEPENDANT
(DNAME, DNUMBER,DLOCATION,DOB, DERELATION )
Values
('KIKSHIT','101','MUMBAI','02/02/1966','FATHER')

INSERT INTO DEPENDANT
(DNAME, DNUMBER,DLOCATION,DOB, DERELATION )
Values
('PRANAV','102','MUMBAI','12/01/1963','FATHER')

INSERT INTO DEPENDANT
(DNAME, DNUMBER,DLOCATION,DOB, DERELATION )
Values
('PYUSHIT','103','MUMBAI','01/05/1953','G FATHER')

INSERT INTO DEPENDANT
(DNAME, DNUMBER,DLOCATION,DOB, DERELATION )
Values
('JIJAY','105','DELHI','09/03/1985','BROTHER')

INSERT INTO DEPENDANT
(DNAME, DNUMBER,DLOCATION,DOB, DERELATION )
Values
('JHANVI','104','GOA','08/06/1968','MOTHER')
```

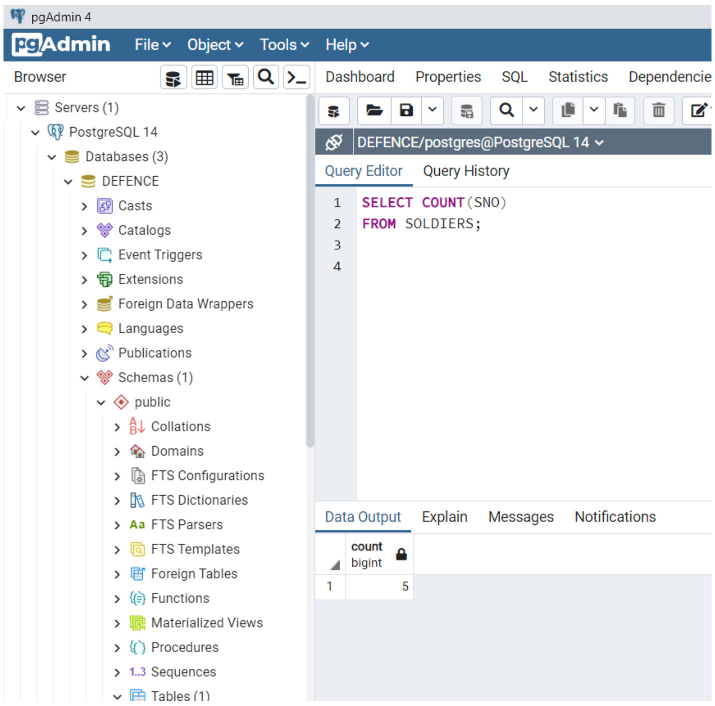


	dname character (10)	dnumber integer	dlocation character (10)	dob date	derelation character (25)
1	KIKSHIT	101	MUMBAI	1966-02-02	FATHER
2	PRANAV	102	MUMBAI	1963-12-01	FATHER
3	PYUSHIT	103	MUMBAI	1953-01-05	G FATHER ...
4	JIJAY	105	DELHI	1985-09-03	BROTHER ...
5	JHANVI	104	GOA	1968-08-06	MOTHER ...

COUNTING SOLDIERS FROM TABLE SOLDIERS :

SELECT COUNT(SNO)

FROM SOLDIERS;

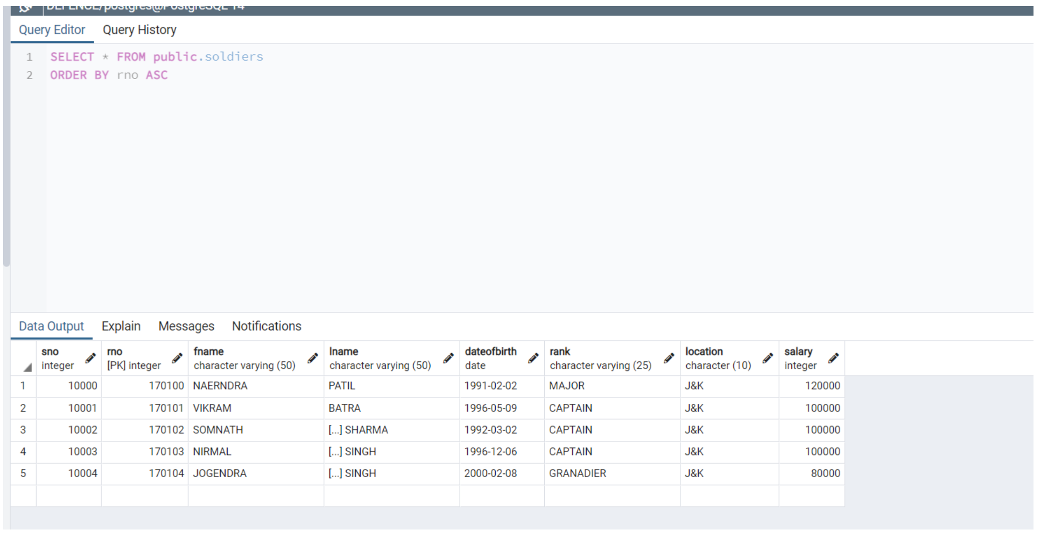


```
SELECT COUNT(SNO)
FROM SOLDIERS;
```

	count bigint
1	5

UPDATING VALUES (SOLDIERS NAME) :

UPDATE SOLDIERS
SET FNAME = 'NAERNDRA' , LNAME = 'PATIL'
WHERE SNO = 10000;



Query Editor | Query History

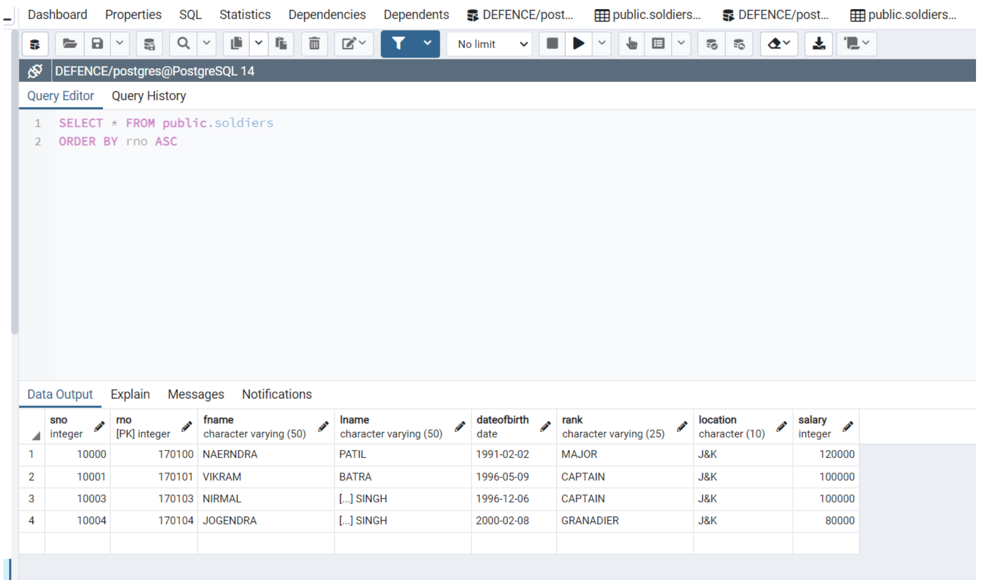
```
SELECT * FROM public.soldiers
ORDER BY rno ASC
```

Data Output | Explain | Messages | Notifications

	sno integer	rno [PK] integer	fname character varying (50)	lname character varying (50)	dateofbirth date	rank character varying (25)	location character (10)	salary integer
1	10000	170100	NAERNDRA	PATIL	1991-02-02	MAJOR	J&K	120000
2	10001	170101	VIKRAM	BATRA	1996-05-09	CAPTAIN	J&K	100000
3	10002	170102	SOMNATH	[...] SHARMA	1992-03-02	CAPTAIN	J&K	100000
4	10003	170103	NIRMAL	[...] SINGH	1996-12-06	CAPTAIN	J&K	100000
5	10004	170104	JOGENDRA	[...] SINGH	2000-02-08	GRANADIER	J&K	80000

DELETING SOLDIER'S NAME FROM TABLE:

DELETE FROM SOLDIERS WHERE FNAME='SOMNATH'



DEFENCE/postgres@PostgreSQL 14

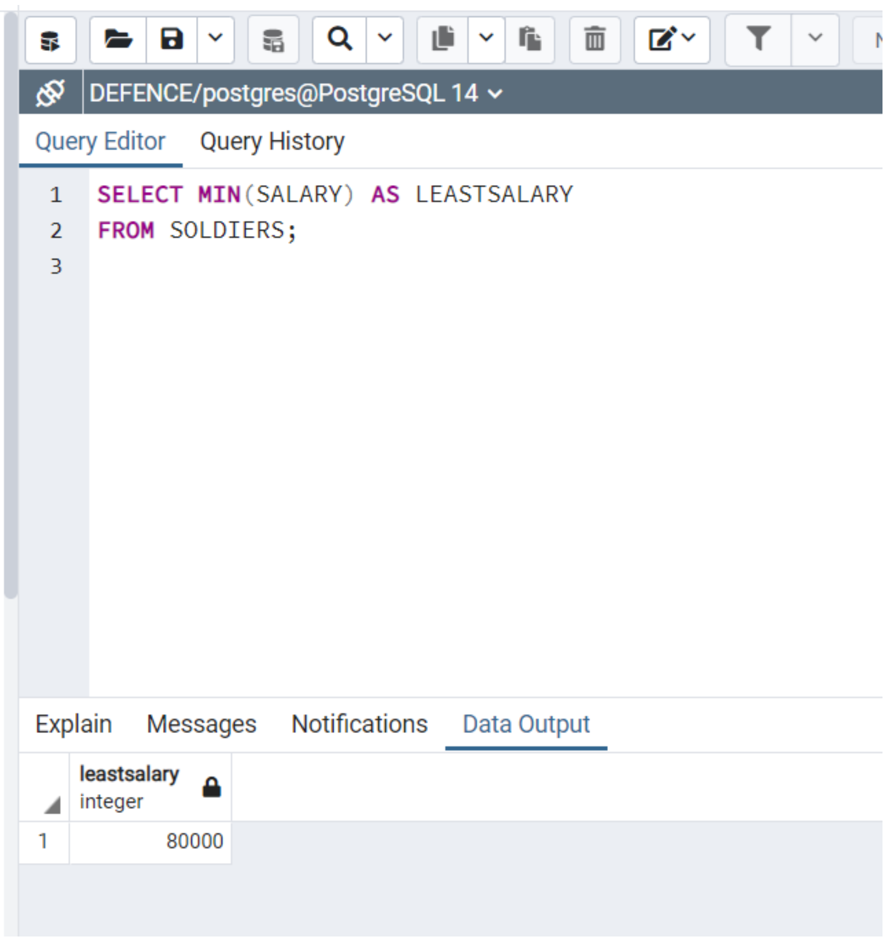
Query Editor | Query History

```
SELECT * FROM public.soldiers
ORDER BY rno ASC
```

Data Output | Explain | Messages | Notifications

	sno integer	rno [PK] integer	fname character varying (50)	lname character varying (50)	dateofbirth date	rank character varying (25)	location character (10)	salary integer
1	10000	170100	NAERNDRA	PATIL	1991-02-02	MAJOR	J&K	120000
2	10001	170101	VIKRAM	BATRA	1996-05-09	CAPTAIN	J&K	100000
3	10003	170103	NIRMAL	[...] SINGH	1996-12-06	CAPTAIN	J&K	100000
4	10004	170104	JOGENDRA	[...] SINGH	2000-02-08	GRANADIER	J&K	80000

**SELECT MIN(SALARY) AS LEASTSALARY
FROM SOLDIERS;**


DEFENCE/postgres@PostgreSQL 14

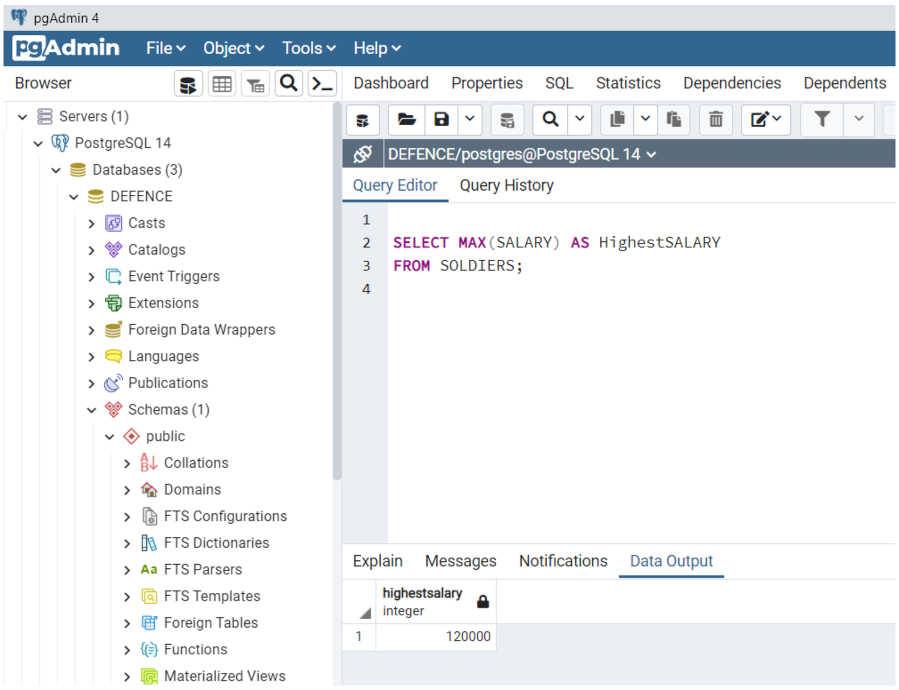
Query Editor | Query History

```
SELECT MIN(SALARY) AS LEASTSALARY
FROM SOLDIERS;
```

Explain | Messages | Notifications | Data Output

```
   leastsalary
   integer
1        80000
```

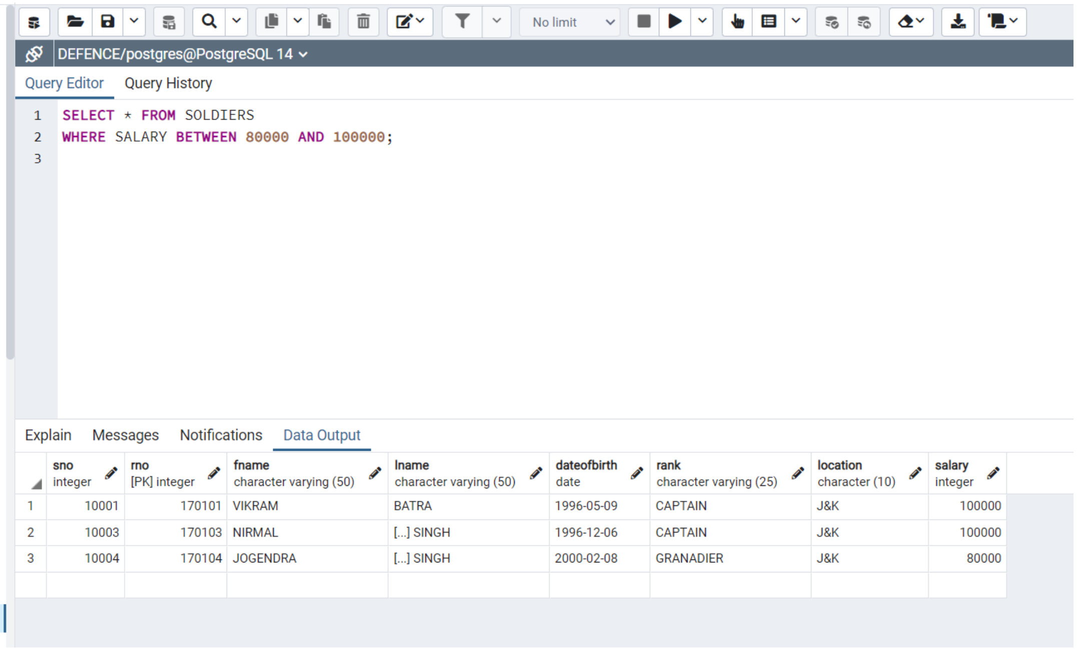
**SELECT MAX(SALARY) AS HighestSALARY
FROM SOLDIERS;**



```
SELECT MAX(SALARY) AS HighestSALARY
FROM SOLDIERS;
```

highestsalary integer
120000

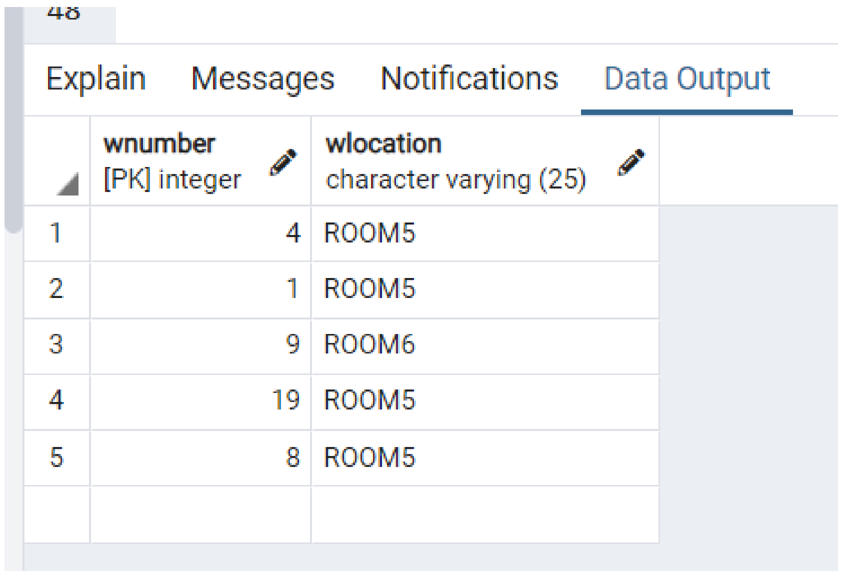
SELECT * FROM SOLDIERS
WHERE SALARY BETWEEN 80000 AND 100000;



```
SELECT * FROM SOLDIERS
WHERE SALARY BETWEEN 80000 AND 100000;
```

	sno integer	rno [PK] integer	fname character varying (50)	lname character varying (50)	dateofbirth date	rank character varying (25)	location character (10)	salary integer
1	10001	170101	VIKRAM	BATRA	1996-05-09	CAPTAIN	J&K	100000
2	10003	170103	NIRMAL	[...] SINGH	1996-12-06	CAPTAIN	J&K	100000
3	10004	170104	JOGENDRA	[...] SINGH	2000-02-08	GRANADIER	J&K	80000

SELECT WNUMBER, WLOCATION FROM WEAPONS;

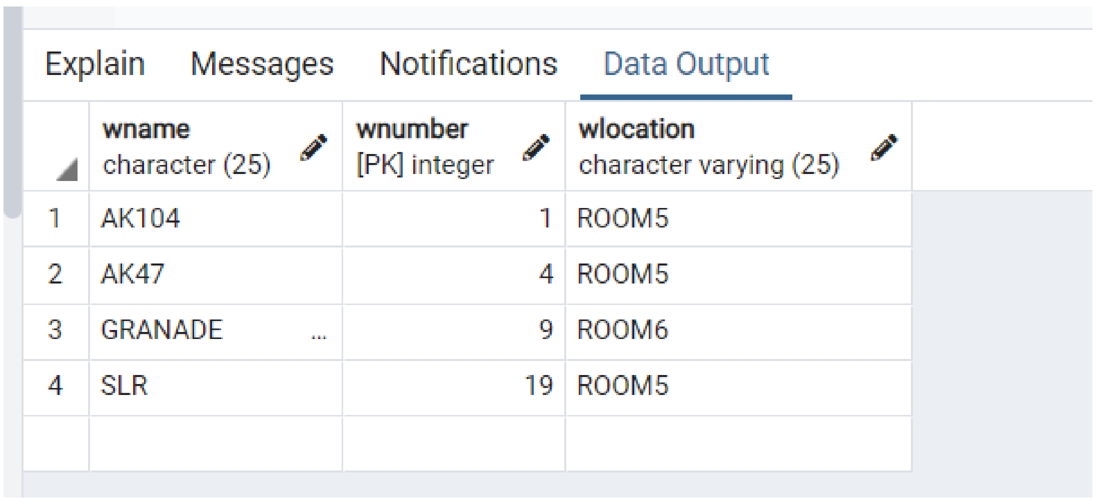


48

Explain Messages Notifications Data Output

	wnumber [PK] integer	wlocation character varying (25)
1	4	ROOM5
2	1	ROOM5
3	9	ROOM6
4	19	ROOM5
5	8	ROOM5

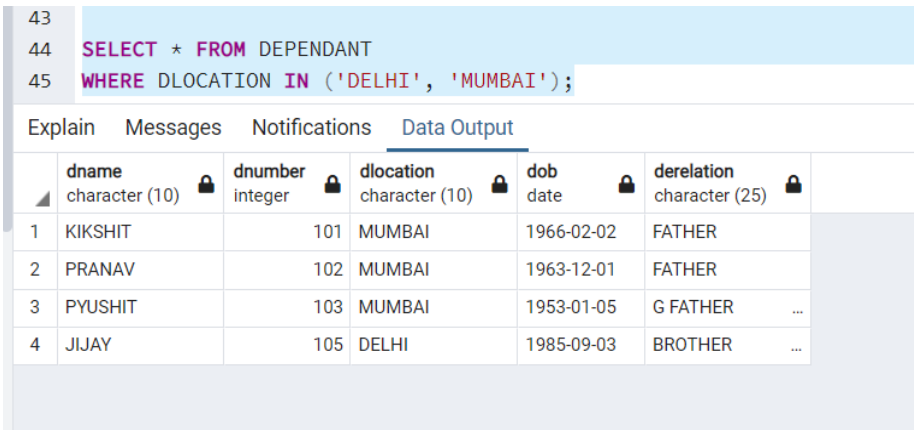
DELETE FROM WEAPONS
WHERE WNAME = INSAS;



Explain Messages Notifications Data Output

	wname character (25)	wnumber [PK] integer	wlocation character varying (25)
1	AK104	1	ROOM5
2	AK47	4	ROOM5
3	GRANADE ...	9	ROOM6
4	SLR	19	ROOM5

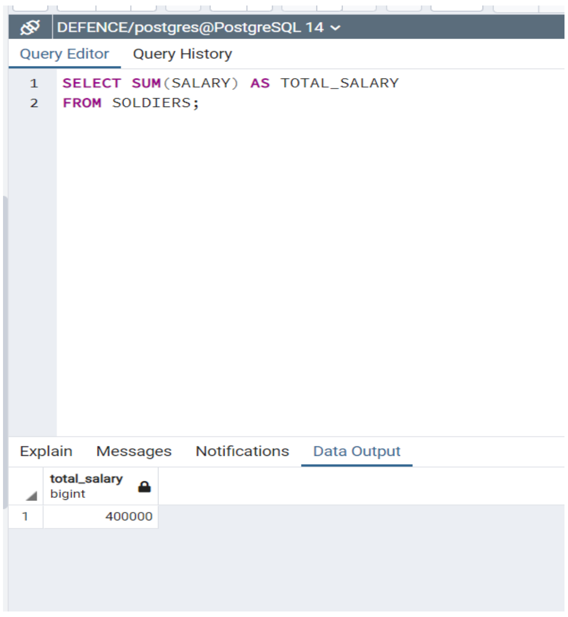
SELECT * FROM DEPENDANT
WHERE DLOCATION IN ('DELHI', 'MUMBAI');



```
SELECT * FROM DEPENDANT
WHERE DLOCATION IN ('DELHI', 'MUMBAI');
```

	dname character (10)	dnumber integer	dlocation character (10)	dob date	derelation character (25)
1	KIKSHIT	101	MUMBAI	1966-02-02	FATHER
2	PRANAV	102	MUMBAI	1963-12-01	FATHER
3	PYUSHIT	103	MUMBAI	1953-01-05	G FATHER
4	JIJAY	105	DELHI	1985-09-03	BROTHER

SELECT SUM(SALARY) AS TOTAL_SALARY
FROM SOLDIERS;



DEFENCE/postgres@PostgreSQL 14

```
SELECT SUM(SALARY) AS TOTAL_SALARY
FROM SOLDIERS;
```

	total_salary bigint
1	400000

Conclusion:

We have successfully performed various DML Operations like insert, update, delete, etc. and also executed nested queries with various queries.

Post Lab Questions

1. **In SQL, which of the following is not a data Manipulation Language Commands?**

   a) Delete
   b) Truncate
   c) Update
   d) Create

ANSWER : B Truncate, D Create

2. **Write SQL query for following statements:**

   a. retrieve all student who his grade has not been awarded
   b. Find the names of all instructors in the Computer Science department
   c. Find the names of all student whose name starts with ‘S’.
   d. find the names of instructors with salary amounts between $90,000 and $100,000.

   e. Find all student sorted by their department name , if there
   f. are two student have the same department name , then sort them by
   g. total credit in ascending order, then by their “student name” alias.

ANSWER:

a. SELECT *

FROM STUDENT

WHERE GRADE = NULL;

b. SELECT NAME

FROM INSTRUCTOR

WHERE DEPARTMENT = “Computer Science”;

c. SELECT NAME

FROM STUDENT

WHERE NAME LIKE “S%”;

d. SELECT NAME

FROM INSTRUCTOR

WHERE SALARY BETWEEN 90000 AND 100000;

e. SELECT NAME AS “student_name”, DEPARTMENT, TOTAL_CREDIT

FROM STUDENT

ORDER BY DEPARTMENT, TOTAL_CREDIT, student_name;