



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

**Batch: A3      Roll No.: 16010121051**

**Experiment / assignment / tutorial No. 4**

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

**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, Silberchatz, Sudarshan : “Database Systems Concept”, 5th Edition , McGraw Hill
4. Elmasri and Navathe,”Fundamentals of database Systems”, 4th Edition PEARSON Education

**Resources used:** Postgres

**Theory:**



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

**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
```



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

**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>| From <table_name>
```

```
Group by <attribute_name>
```

**Example:** select designation, sum( salary) as total\_salary from employee group by Designation;



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

**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) ;
```



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

**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%';
```



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

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



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

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 returns 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



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

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**

**- Simple question based on your application, queries and screen shots for each type:**

**a.insertion of data in customer table**

**Code:**

```
insert into customer values(01,'admin@mail.com','abc123')
```

```
insert into customer values(02,'cust2@mail.com','abc124');
```





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

**insert into customer values(03,'cust3@mail.com','abc125');**

**insert into customer values(04,'cust4@mail.com','abc126');**

**insert into customer values(05,'cust5@mail.com','abc127')**

**select \*from customer**

Data Output	Explain	Messages	Notifications
uid [PK] integer	umail character varying (30)	upassword character varying (10)	
1	1 admin@mail.com	abc123	
2	2 cust2@mail.com	abc124	
3	3 cust3@mail.com	abc125	
4	4 cust4@mail.com	abc126	
5	5 cust5@mail.com	abc127	

**b.insertion of data in sender table**

**insert into sender values('meet','meet@mail.com','ad1','01','201');**

**insert into sender values('riya','riya@mail.com','ad2','02','202');**

**insert into sender values('desai','desai@mail.com','ad3','03','203');**

**insert into sender values('beet','beet@mail.com','ad4','04','204');**

**insert into sender values('pargat','pargat@mail.com','ad5','05','205');**

**select \*from sender**

Data Output						Explain	Messages	Notifications
	sname character varying (20)	umail character varying	saddress character varying (100)	uid [PK] integer	sphone integer			
1	meet	meet@mail.com	ad1		1	201		
2	riya	riya@mail.com	ad2		2	202		
3	desai	desai@mail.com	ad3		3	203		
4	beet	beet@mail.com	ad4		4	204		
5	pargat	pargat@mail.com	ad5		5	205		



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

**c.insertion of data in receiver table**

**insert into receiver values('meet','meet@mail.com','ad11','01','301');**

**insert into receiver values('riya','riya@mail.com','ad22','02','302');**

**insert into receiver values('desai','desai@mail.com','ad33','03','303');**

**insert into receiver values('beet','beet@mail.com','ad44','04','304');**

**insert into receiver values('pargat','pargat@mail.com','ad55','05','305');**

**select \*from receiver**

SQL

Data Output		Explain	Messages	Notifications		
	name character varying (20)	rmail character varying (30)	raddress character varying (100)	rid integer	rphone integer	
1	meet	meet@mail.com	ad11	1	301	
2	riya	riya@mail.com	ad22	2	302	
3	desai	desai@mail.com	ad33	3	303	
4	beet	beet@mail.com	ad44	4	304	
5	pargat	pargat@mail.com	ad55	5	305	



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




**d. inserted values in courier table**

**insert into courier values(01,5);**

**insert into courier values(02,6);**

**select \*from courier**

34

Data Output		Explain	Messages	Notifications
	cid [PK] integer 	cweight integer 		
1	1	5		
2	2	6		

**e. inserted into admin**

**insert into adminn values('231','meet@mail.com',1,01,01);**

**insert into adminn values('232','riya@mail.com',2,02,02);**

**select \*from adminn**

2	231	meet@mail.com	1	1	1
3	232	riya@mail.com	2	2	2









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

**f. added deliveries by insert**

**insert into delivery values('express',27,31,'air',01);**

**insert into delivery values('normal',28,31,'road',02);**

**select \* from delivery**

Data Output						Explain	Messages	Notifications
	 deliverytype character varying (20)	 ordplaceddate integer	 ordrdelivereddate integer	 modetrans character varying (30)	 cid integer			
1	express	27	31	air	1			
2	normal	28	31	road	2			

**g. insert into billingsystem**

**insert into billingsys values(1,987,'cash',2000,01,01);**

**select \* from billingsys**

42

Data Output	Explain	Messages	Notifications
<div><div></div><div>bid</div><div>[PK] integer</div><div></div></div>	<div><div></div><div>bno</div><div>integer</div><div></div></div>	<div><div></div><div>modepay</div><div>character varying (20)</div><div></div></div>	<div><div></div><div>tamount</div><div>integer</div><div></div></div> <div><div></div><div>cid</div><div>integer</div><div></div></div> <div><div></div><div>uid</div><div>integer</div><div></div></div>
1	1	987	cash



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

**f.update customer**

**update customer**

**set uemail='cust1@mail.com'**







**where uid=1**

Data Output	Explain	Messages	Notifications
uid [PK] integer	uemail character varying (30)	upassword character varying (10)	
1	2 cust2@mail.com	abc124	
2	3 cust3@mail.com	abc125	
3	4 cust4@mail.com	abc126	
4	5 cust5@mail.com	abc127	
5	1 cust1@mail.com	abc123	

**g.delete receiver**

**delete from receiver**

**where rid=5**

Data Output						Explain	Messages	Notifications
	 rname character varying (20)	 rmail character varying (30)	 raddress character varying (100)	 rid integer	 rphone integer			
1	meet	meet@mail.com	ad11	1	301			
2	riya	riya@mail.com	ad22	2	302			
3	desai	desai@mail.com	ad33	3	303			
4	beet	beet@mail.com	ad44	4	304			



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

**Conclusion:**

Through this experiment we perform various DML Operations and executing nested queries with various clauses. We use select, insert, update, delete etc commands to manipulate data in our tables under the airline database.

**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

Ans- d)Create

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

- a. Retrieve all student who his grade has not been awarded
  - Select *students* from *course* where *grade=0*;
- b. Find the names of all instructors in the Computer Science department
  - Select *instructore* from *course* where *dept='Comp Sci'*;
- c. Find the names of all student whose name starts with 'S'.
  - Select *students* from *course* where *name like 's%'*;
- d. find the names of instructors with salary amounts between \$90,000 and \$100,000.
  - Select *instructors* from *course* where *salary* between 90000 and 100000;
- e. Find all student sorted by their department name, if there are two students have the same department name, then sort them by total credit in ascending order, then by their "student name" alias.



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

- Select *student, dept, credits* from *course* order by *dept, credits desc, student*