



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

**Batch: A3 Roll No.: 1911034**

**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 –



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

**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>| From <table_name>  
Group by <attribute_name>
```



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

**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.



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

**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:*



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

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*;



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

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 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;



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

## **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);





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

### Implementation details

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

Create:-

```
Command Prompt - mysql -u root -p

+-----+-----+-----+-----+-----+-----+
| asc_bank | varchar(10) | YES | | NULL | | |
+-----+-----+-----+-----+-----+-----+
7 rows in set (0.00 sec)

mysql> use proppsys;
Database changed
mysql> create table builder(name_b varchar(10),age int,id_no_b int,exp float);de
sc builder;
Query OK, 0 rows affected (0.28 sec)

+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| name_b | varchar(10) | YES | | NULL | |
| age | int(11) | YES | | NULL | |
| id_no_b | int(11) | YES | | NULL | |
| exp | float | YES | | NULL | |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.01 sec)

mysql> _
```

Insert:-

```
mysql> create table Property_1(area int,location varchar(10),cost float,no_of_ro
oms int,type_p varchar(10),p_id float);
Query OK, 0 rows affected (0.33 sec)

mysql> INSERT INTO Property_1(area,location,cost,no_of_rooms,type_p,p_id)VALUES
(550,'delhi', 456789, 5, 'rental',6743);
Query OK, 1 row affected (0.08 sec)

mysql> INSERT INTO Property_1(area,location,cost,no_of_rooms,type_p,p_id)VALUES
(200,'mumbai', 21333.99, 4, 'rental',2399);
Query OK, 1 row affected (0.18 sec)

mysql> INSERT INTO Property_1(area,location,cost,no_of_rooms,type_p,p_id)VALUES
(340,'pune', 950000, 2, 'ownership',4210);
Query OK, 1 row affected (0.07 sec)

mysql> select * from Property_1;

+-----+-----+-----+-----+-----+-----+
| area | location | cost | no_of_rooms | type_p | p_id |
+-----+-----+-----+-----+-----+-----+
| 550 | delhi | 456789 | 5 | rental | 6743 |
| 200 | mumbai | 21334 | 4 | rental | 2399 |
| 340 | pune | 950000 | 2 | ownership | 4210 |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

Delete:-



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

```
mysql> UPDATE Property_1 SET location = 'punjab', area= '350' WHERE p_id= 4210;
Query OK, 1 row affected (0.07 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> select * from Property_1;
+-----+-----+-----+-----+-----+-----+
| area | location | cost   | no_of_rooms | type_p | p_id |
+-----+-----+-----+-----+-----+-----+
| 550 | delhi    | 456789 | 5           | rental | 6743 |
| 350 | punjab   | 950000 | 2           | ownership | 4210 |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

**Update:-**

```
mysql> DELETE FROM 'Property_1' WHERE 'p_id' = 2399;
Query OK, 1 row affected (0.11 sec)

mysql> select * from Property_1;
+-----+-----+-----+-----+-----+-----+
| area | location | cost   | no_of_rooms | type_p | p_id |
+-----+-----+-----+-----+-----+-----+
| 550 | delhi    | 456789 | 5           | rental | 6743 |
| 340 | pune     | 950000 | 2           | ownership | 4210 |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

**Original property table:-**

	area	location	cost	no_of_rooms	type_p	p_id
▶	200	mumbai	2355.16	4	rental	1210
	350	pune	1297.66	3	ownership	2133
	410	thane	3500.01	2	rental	2159
	300	churchgate	3500.77	3	rental	2211
	250	andheri	2200.22	2	rental	3221
	500	ghatkopar	1799.66	5	ownshership	7651
	NULL	NULL	NULL	NULL	NULL	NULL

**Original customer table:-**



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

	name_c	age	id_no	budget	type_p	no_of_emi	asc_bank
▶	ashwini	48	1122	5000	ownership	12	HDFC Bank
	Aditi	19	1210	9000	rental	7	ICPC Bank
	Dhruvi	19	1998	10000	rental	9	HDFC Bank
	Samiksha	19	2133	4500	ownership	8	Canara
	Pinky	21	9987	2300	rental	9	Baroda
	Siddhi	22	9989	3000	ownership	10	Canara
	NULL	NULL	NULL	NULL	NULL	NULL	NULL

**sum function**

select sum(cost) as t\_cost from property;

	t_cost
▶	14653.47998046875

**EXISTS clause**

select name\_c , id\_no FROM Customer WHERE exists (select \* from property  
WHERE property.p\_id = customer.id\_no);

	name_c	id_no
▶	Aditi	1210
	Samiksha	2133
*	NULL	NULL

**IN Clause**

select name\_c , id\_no , asc\_bank FROM customer WHERE asc\_bank IN ("HDFC  
Bank");

	name_c	id_no	asc_bank
▶	ashwini	1122	HDFC Bank
	Dhruvi	1998	HDFC Bank
*	NULL	NULL	NULL

**BETWEEN clause**

select \* from customer where budget BETWEEN 8000 AND 11000



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

	name_c	age	id_no	budget	type_p	no_of_emi	asc_bank
▶	Aditi	19	1210	9000	rental	7	ICPC Bank
	Dhruvi	19	1998	10000	rental	9	HDFC Bank
•	NULL	NULL	NULL	NULL	NULL	NULL	NULL

**LIKE clause**

select \* from customer WHERE name\_c LIKE '\_i%';

	name_c	age	id_no	budget	type_p	no_of_emi	asc_bank
▶	Pinky	21	9987	2300	rental	9	Baroda
	Siddhi	22	9989	3000	ownership	10	Canara
•	NULL	NULL	NULL	NULL	NULL	NULL	NULL

**Column Alias**

select location AS p\_loc FROM property WHERE no\_of\_rooms>2;

	p_loc
▶	mumbai
	pune
	churchgate
	ghatkopar

**DISTINCT clause**

select DISTINCT type\_p from Customer;

	type_p
▶	ownership
	rental

**UNION**

SELECT area, name\_c, location, id\_no FROM property LEFT OUTER JOIN customer  
ON customer.id\_no = property.p\_id

**UNION**

SELECT area, name\_c, location, id\_no FROM property RIGHT OUTER JOIN  
customer ON customer.id\_no = property.p\_id;



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

	area	name_c	location	id_no
▶	200	Aditi	mumbai	1210
	350	Samiksha	pune	2133
	410	NULL	thane	NULL
	300	NULL	churchgate	NULL
	250	NULL	andheri	NULL
	500	NULL	ghatkopar	NULL
	NULL	ashwini	NULL	1122
	NULL	Dhruvi	NULL	1998
	NULL	Pinky	NULL	9987
	NULL	Siddhi	NULL	9989

UNION ALL

SELECT area, name\_c, location, id\_no FROM property LEFT OUTER JOIN customer  
ON customer.id\_no = property.p\_id

UNION ALL

SELECT area, name\_c, location, id\_no FROM property RIGHT OUTER JOIN  
customer ON customer.id\_no = property.p\_id;

	area	name_c	location	id_no
▶	200	Aditi	mumbai	1210
	350	Samiksha	pune	2133
	410	NULL	thane	NULL
	300	NULL	churchgate	NULL
	250	NULL	andheri	NULL
	500	NULL	ghatkopar	NULL
	NULL	ashwini	NULL	1122
	200	Aditi	mumbai	1210
	NULL	Dhruvi	NULL	1998
	350	Samiksha	pune	2133
	NULL	Pinky	NULL	9987
	NULL	Siddhi	NULL	9989



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

ANY clause

Select name\_c , no\_of\_emi FROM Customer WHERE no\_of\_emi = ANY(select  
no\_of\_emi from customer where no\_of\_emi > 8);

	name_c	no_of_emi
▶	ashwini	12
	Dhruvi	9
	Pinky	9
	Siddhi	10





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

**Conclusion:** we have successfully learned and implemented:-

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

**Post Lab Questions**

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

- a) Delete
- b) Truncate
- c) Update

**ANS:-d) Create**

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

- a. retrieve all student who his grade has not been awarded



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

SELECT \* FROM student WHERE grade=NULL

- b. Find the names of all instructors in the Computer Science department**

SELECT name  
FROM instructors WHERE department=computer science

- c. Find the names of all student whose name starts with 'S'.**

SELECT \*  
FROM student WHERE name LIKE '%S'

- d. find the names of instructors with salary amounts between \$90,000 and \$100,000.**

Select name

From instructors WHERE salary>90000 and salary<100000

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

SELECT \*

FROM student AS record ORDER BY dep\_name , total creditASC ,  
name