Experiment No.05

PART A

(PART A: TO BE REFERRED BY STUDENTS)

A.1 Aim: To study and implement DML select statement with where clause, and, or, not, in, between and like clause.

A.2 Prerequisite:

DML commands of SQL

A.3 Outcome:

After successful completion of this experiment students will be able to

1. Apply knowledge of relational algebra and structural query language to retrieve and manage data in relational databases.

A.4 Theory:

The select statement is used to view records from the database. In order to view only specific records, select statement is used with *where* clause. The SQL WHERE clause is used to specify a condition while fetching the data from single table or joining with multiple tables. If the given condition is satisfied then only it returns specific value from the table. You would use WHERE clause to filter the records and fetching only necessary records.

The WHERE clause is not only used in SELECT statement, but it is also used in UPDATE, DELETE statement, etc.,

SQL syntax for Select statement with where clause:

```
SELECT column_name,column_name
FROM table_name
WHERE column_name operator value;
```

Example:

select * from customers where country='CHINA';

The above query will select customers from the country CHINA.

The SQL requires single quotes around text values (as shown in above example). However numeric fields should not be enclosed in quotes.

Example:

select * from customers where customer_id=1;

The following operators can be used in the WHERE clause:

Operator	Description
=	Equal
<>	Not equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
Between	Between an inclusive range
Like	Search for a pattern
IN	To specify multiple possible values for a column
AND	To combine two conditions – both condition needs to be true
OR	To combine two conditions- either of the two condition has to be true

Using > operator with where clause:

Example:

select employee_id, name, from employee where salary>50000;

Using = operator with where clause:

select employee_id, name,salary from employee where name = ‘Hardik’;

SQL AND- OR operator:

The AND operator allows the existence of multiple conditions in an SQL statement's WHERE clause.

The basic syntax of AND operator with WHERE clause is as follows:

```
SELECT column1, column2, columnN
FROM table_name
```

WHERE [condition1] AND [condition2]...AND [conditionN];

You can combine N number of conditions using AND operator. For an action to be taken by the SQL statement, whether it be a transaction or query, all conditions separated by the AND must be TRUE.

Example:

select employee_id, name, salary from employee where salary >2000 and age <25;

The OR operator is used to combine multiple conditions in an SQL statement's WHERE clause.

The basic syntax of OR operator with WHERE clause is as follows:

```
SELECT column1, column2, columnN
FROM table_name
WHERE [condition1] OR [condition2]...OR [conditionN]
```

You can combine N number of conditions using OR operator. For an action to be taken by the SQL statement, whether it be a transaction or query, only any ONE of the conditions separated by the OR must be TRUE.

Example:

select employee_id, name, salary from employee where salary>2000 or age<25;

The IN operator:

The IN operator allows you to specify multiple values in a WHERE clause.

SQL IN syntax:

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (value1,value2,...);
```

Example:

The following SQL statement selects all customers with a City of "Paris" or "London":

select * from customers where city not in (‘Paris’,’London’);

SQL Between operator:

The BETWEEN operator is used to select values within a range. The values can be number, text or dates.

SQL BETWEEN syntax:

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

Example:

The following SQL statement selects all products with a price BETWEEN 10 and 20:

select * from products where price between 10 and 20;

To display the products outside the range of the previous example, use NOT BETWEEN:

select * from products where price not between 10 and 20;

SQL LIKE operator:

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

SQL LIKE syntax:

```
SELECT column_name(s)
FROM table_name
WHERE column_name LIKE pattern;
```

SQL Wildcard characters:

In SQL, wildcard characters are used with the SQL LIKE operator. SQL wildcards are used to search for data within a table.

With SQL, the wildcards are:

Wildcard	Description
%	A substitute for zero or more characters
_	A substitute for a single character
[*charlist*]	Sets and ranges of characters to match
[^*charlist*] or [!*charlist*]	Matches only a character NOT specified within the brackets

Using LIKE with SQL Wildcard characters:

The following SQL statement selects all customers with a City starting with "ber":

select * from customers where city like ‘ber%’;

The following SQL statement selects all customers with a City containing the pattern "es":

select * from customers where city like ‘%es%’;

The following SQL statement selects all customers with a City starting with any character, followed by "erlin":

select * from customers where city like ‘_erlin’;

The following SQL statement selects all customers with a City starting with "L", followed by any character, followed by "n", followed by any character, followed by "on":

select * from customers where city like ‘L_n_on’;
The following SQL statement selects all customers with a City starting with "b", "s", or "p":

select * from customers where city like ‘[bsp]%’;

A.5 Task: For given tables solve below queries:

category_header

Category_header

Cat_code	Cate_desc
01	super delux
02	delux
03	super fast
04	normal

route_Header

Route_id	Route_no	Cate_code	Origin	Destination	Fare	Distance	Capacity
101	33	01	Madurai	Madras	35	250	50
102	25	02	Trichy	Madurai	40	159	50
103	15	03	Thanjavur	Madurai	59	140	50
104	36	04	Madras	Banglore	79	375	50
105	40	01	Banglore	Madras	80	375	50
106	38	02	Madras	Madurai	39	250	50
107	39	03	Hydrabad	Madras	50	430	50
108	41	04	Madras	Cochin	47	576	50

Place Header:

Place_id	Place_name	Place_address	Bus_station
01	Madras	10, ptc road	Parrys
02	Madurai	21, canal bank road	Kknagar
03	Trichy	11, first cross road	Bheltown
04	Banglore	15, first main road	Cubbon park
05	Hydrabad	115,lake view road	Charminar
06	Thanjavur	12, temple road	Railway jn.

Fleet Header:

Fleet_id	Day	Route_id	Cat_code
01	10-apr-96	101	01
02	10-apr-96	101	01
03	10-apr-96	101	01
04	10-apr-96	102	02
05	10-apr-96	102	03
06	10-apr-96	103	04

Ticket Header:

Fleet_id	Ticket_no	Doi	Dot
01	01	10-apr-96	10-may-96
02	02	12-apr-96	5-may-96
03	03	21-apr-96	15-may-96

Time_travel	Board_place	Origin	Destinition
15:00:00	Parrys	Madrsa	Madurai
09:00:00	Kknagar	Madurai	Madras
21:00:00	Cubbon park	Banglore	Madras

Adults	Children	Total_fare	Route_id
1	1	60	101
2	1	60	102
4	2	400	101

Ticket Detail:

~~Adults~~ ticket_no ~~fleet_id~~	Name	Sex	Age	Fare
01	Charu	F	24	14.00
01	Lathu	F	10	15.55
02	Anand	M	28	17.80
02	Guatham	M	28	16.00
02	Bala	M	09	17.65
05	Sandip	M	30	18.00

Route Detail:

Route_id	Place_id	Nonstop
105	01	N
012	02	S
106	01	S
108	05	N
106	02	N

Queries:

1. Display only those routes that originate in “madras” and terminate in “cochin”.

2. Display only those rows from route_header whose origin begins with ‘m’.
3. Display only those rows whose fare ranges between 30 and 50.
4. Display the fare and the origin for route_no which are greater than 15.
5. Display all details of places whose name begins with ‘m’.
6. Display those routes whose distance is in range of 200 and 400.
7. Find out fleets which travel through route 102 or 103.
8. Find out routes which are non stop.
9. Find out category whose category description starts with ‘s’ and ends with ‘t’.
10. Find out routes which have category code 1, 2 or 4.
11. Display details of place with bus station “charminar”.
12. Display details of those routes whose fare is less than 70 and distance greater than 120.
13. Find out details of tickets issued to female travelers and with age greater than 10.
14. What will be fare of each route after incrementing fare by 10 percent?
15. Find out details of routes with 101 or 105 or 107.
16. Display those routes for which origin is “Madras” and distance is greater than 300 or destination is “Madras” and distance less than 300.
17. Create a new table temp_MPSTME with columns as place_id, place_name and place_address **(column data type and size must be same as columns of place_header table).**
18. Write a query to insert records into temp_MPSTME table from place_header table. Only those records are to be selected for which place_id is between 1 to 4 and place_name starts from ‘m’.

PART B

(PART B: TO BE COMPLETED BY STUDENTS)

(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the Portal or emailed to the concerned lab in charge faculties at the end of the practical in case the there is no Portal access available)

Roll No. I066	Name: Srihari Thyagarajan
Program : B Tech Artificial Intelligence	Division: I
Batch: B3	Date of Experiment: 02/09/2022
Date of Submission: 02/09/2022	Grade :

B.1 Tasks given in PART A to be completed here

B.1.1 : Execution of each SQL query used to complete the task given in PART A :

Query 1:

SELECT * FROM route_header WHERE origin = 'Madras' and destination = 'Cochin';

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Co

route_id	route_no	cat_code	origin	destination	fare	distance	capacity	new_fare
108	41	4	Madras	Cochin	200.00	600	50	57.00
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Query 2 :

SELECT * FROM route_header WHERE origin LIKE 'M%';

Result Grid | Filter Rows: | Edit: | Export/Import: | Wra

route_id	route_no	cat_code	origin	destination	fare	distance	capacity	new_fare
101	33	1	Madurai	Madras	40.00	250	50	50.00
104	36	4	Madras	Bangalore	200.00	600	50	89.00
106	38	2	Madras	Madurai	39.00	250	50	49.00
108	41	4	Madras	Cochin	200.00	600	50	57.00
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Query 3:

SELECT * FROM route_header WHERE fare between 30 and 50;

route_id	route_no	cat_code	origin	destination	fare	distance	capacity	new_fare
101	33	1	Madurai	Madras	40.00	250	50	50.00
102	25	2	Trichy	Madurai	40.00	159	50	50.00
105	40	1	Bangalore	Madras	40.00	375	50	50.00
106	38	2	Madras	Madurai	39.00	250	50	49.00
107	39	3	Hyderabad	Madras	50.00	430	50	60.00
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Query 4:

SELECT fare, origin FROM route_header WHERE route_no >= 15;

fare	origin
40.00	Madurai
40.00	Trichy
59.00	Thanjavur
200.00	Madras
40.00	Bangalore
39.00	Madras
50.00	Hyderabad
200.00	Madras

Query 5:

SELECT * FROM place_header WHERE place_name LIKE 'M%';

Result Grid | Filter Rows: | Edit:

place_id	place_name	place_address	bus_station
1	Madras	10, ptc road	Parrys
2	Madurai	21 Canal Bank road	Kknagar
NULL	NULL	NULL	NULL

Query 6:

SELECT * FROM route_header WHERE distance between 200 and 400;

route_id	route_no	cat_code	origin	destination	fare	distance	capacity	new_fare
101	33	1	Madurai	Madras	40.00	250	50	50.00
105	40	1	Bangalore	Madras	40.00	375	50	50.00
106	38	2	Madras	Madurai	39.00	250	50	49.00
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Query 7:

SELECT * FROM fleet_header WHERE route_id = 102 or route_id = 103;

fleet_id	route_id	cat_code	day
4	102	2	1996-04-10
5	102	3	1996-04-10
6	103	4	1996-04-10
NULL	NULL	NULL	NULL

Query 8:

SELECT * FROM route_detail WHERE nonstop = 'N';

route_id	place_id	nonstop
105	1	N
108	5	N
106	2	N

Query 9:

SELECT * FROM category_header WHERE cat_desc LIKE 's%t';

cat_code	cat_desc
3	super fast
NULL	NULL

Query 10:

SELECT * FROM category_header WHERE cat_code = 1 or cat_code = 2 or cat_code = 4;

cat_code	cat_desc
1	super deluxe
2	deluxe
4	normal
NULL	NULL

Query 11:

SELECT * FROM place_header WHERE bus_station = 'Charminar';

Result Grid | Filter Rows: | Edit:

place_id	place_name	place_address	bus_station
5	Hyderabad	115 Lake View Road	Charminar
NULL	NULL	NULL	NULL

Query 12:

SELECT * FROM route_header WHERE fare <=70 or fare > 120;

route_id	route_no	cat_code	origin	destination	fare	distance	capacity	new_fare
101	33	1	Madurai	Madras	40.00	250	50	50.00
102	25	2	Trichy	Madurai	40.00	159	50	50.00
103	15	3	Thanjavur	Madurai	59.00	140	50	69.00
104	36	4	Madras	Bangalore	200.00	600	50	89.00
105	40	1	Bangalore	Madras	40.00	375	50	50.00
106	38	2	Madras	Madurai	39.00	250	50	49.00
107	39	3	Hyderabad	Madras	50.00	430	50	60.00
108	41	4	Madras	Cochin	200.00	600	50	57.00
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Query 13:

SELECT * FROM ticket_detail where Sex = 'F' and Age > 10;

ticket_no	name	sex	age	fare
1	Charu	F	24	14.00

Query 14:

SELECT fare, fare + 0.1*fare as new_fare_ FROM route_header;

fare	new_fare_
40.00	44.00
40.00	44.00
59.00	64.90
200.00	220.00
40.00	44.00
39.00	42.90
50.00	55.00
200.00	220.00

Query 15:

SELECT * FROM route_header WHERE route_id = 101 or route_id = 105 or route_id = 107;

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell

route_id	route_no	cat_code	origin	destination	fare	distance	capacity	new_fare
101	33	1	Madurai	Madras	40.00	250	50	50.00
105	40	1	Bangalore	Madras	40.00	375	50	50.00
107	39	3	Hyderabad	Madras	50.00	430	50	60.00
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Query 16:

SELECT * FROM route_header WHERE (origin = 'Madras' and distance > 300) or (origin = 'Madras' and distance < 300);

route_id	route_no	cat_code	origin	destination	fare	distance	capacity	new_fare
104	36	4	Madras	Bangalore	200.00	600	50	89.00
106	38	2	Madras	Madurai	39.00	250	50	49.00
108	41	4	Madras	Cochin	200.00	600	50	57.00
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Query 17:

Create Table IF NOT EXISTS temp_MPSTME as (SELECT place_id, place_name, place_address from place_header);

Field	Type	Null	Key	Default	Extra
place_id	int	NO		NULL	
place_name	varchar(20)	NO		NULL	
place_address	varchar(50)	YES		NULL	

Query 18:

SELECT place_id, place_name, place_address

FROM place_header

WHERE (place_id BETWEEN 1 and 4) and (place_name LIKE 'M%');

place_id	place_name	place_address
1	Madras	10, ptc road
2	Madurai	21 Canal Bank road
NULL	NULL	NULL

B.2 Observations and Learning:

From the above experiment, I learnt the application and implementation of various SQL Queries which help in easy retrieval and modification of data from the tables created and displaying them accordingly.

B.3 Conclusion:

The experiment covered SQL Queries on relations created and making necessary changes accordingly. It helped in understanding the approach of non-procedural language.

B.4 Question of curiosity:

1. **Justify the statement, *"Various clauses used with DML helps for better manipulation of the given table"***

Ans) SQL Queries help and facilitate faster, easier and organized retrieval and modification of data as per one's needs. There are various ways in which data can be fetched and modified (having different syntax). It encompasses many methods in which data is retrieved to facilitate and enhance the manipulation of data in the tables.

2. ***Differentiate between 'in' and 'Between' operators used for Databases.***

Ans) The 'Between' operator helps in specifying the range from which data is to be selected, whereas the 'in' operator on the other hand, allows one to specify and check data in multiple values in the table. Instead of writing the 'or' operator over and over again, the 'in' operator can be used.

3. ***What is wild card character used in databases?***

Ans) The wild characters such as LIKE are used to select a contiguous piece of characters in the string and are checked for a specific condition/ pattern.

****