# SɅRɅSWɅTI

## Department of Computer Engineering

**College of Engineering**

SɅRɅSWɅTI

### Experiment 5

###### Aim: Perform string manipulation operations and aggregate functions with group by.. Having clause.

**Hardware and Software Requirement:** P-IV and above, Oracle

###### Theory:

1. Like Condition:

A LIKE condition specifies a test involving pattern matching. We describe patterns by using two special characters:

* + Percent (%): The % character matches any substring.
  + Underscore ( ): The character matches any character.

###### Syntax:

SELECT columns FROM tables

WHERE column1 **LIKE ’**%\_’;

Example:

List the customers whose name begin with the letters’' SELECT FName,LName

FROM cust

WHERE FName LIKE ‘Ch%’;

The % indicates that any number of character can follow the letter Ch. select lower(name) || upper(name) from try;

select lower('RINA') from dual; select upper(name) from try;

select \* from try where name like '%mal%'

# SɅRɅSWɅTI

## Department of Computer Engineering

**College of Engineering**

SɅRɅSWɅTI

###### Logical operator:

1. OR operator:

The OR condition allows you to create an SQL statement where records are returned when any one of the conditions are met. It can be used in any valid SQL statement - select, insert, update, or delete.

###### Syntax:

SELECT columns FROM tables

WHERE column1 = 'value1' or column2 = 'value2'

The OR condition requires that any of the conditions be must be met for the record to be included in the result set. In this case, column1 has to equal 'value1' OR column2 has to equal 'value2'.

###### Example:

SELECT \*

FROM suppliers

WHERE city = 'New York' or city = 'Newark';

This would return all suppliers that reside in either New York or Newark. Because the \* is used in the select, all fields from the suppliers table would appear in the result set.

1. AND Operator:

The AND operator displays a record if both the first condition and the second condition is true.

###### Syntax:

SELECT columns FROM tables

WHERE column1 = 'value1' and column2 = 'value2'

###### Example:

SELECT \* FROM EMP

WHERE EmpTown = 'London' AND EmpAge > 30

# SɅRɅSWɅTI

## Department of Computer Engineering

**College of Engineering**

SɅRɅSWɅTI

1. NOT Operator:

If you want to find rows that do not satisfy a condition, you can use the logical operator, NOT. NOT results in the reverse of a condition. That is, if a condition is satisfied, then the row is not returned.

**Example:**

If you want to find out the names of the students who do not play football, the query would be like:

SELECT first\_name, last\_name, games FROM student\_details

WHERE NOT games = 'Football' ;

* + ORDER BY clause: Sorting of data in table

The ORDER BY keyword is used to sort the result-set by a specified column. The ORDER BY keyword sort the records in ascending order by default.

If you want to sort the records in a descending order, you can use the DESC keyword

###### Syntax:

SELECT "column\_name" FROM "table\_name" [WHERE "condition"]

ORDER BY "column\_name" [ASC, DESC];

**Aggregate functions**

Aggregate functions return a single result row based on groups of rows, rather than on single rows.

**avg:** average value **min:** minimum value **max:** maximum value **sum:** sum of values

**count:** number of values

Find the average account balance at the Perryridge branch

# SɅRɅSWɅTI

## Department of Computer Engineering

**College of Engineering**

SɅRɅSWɅTI

**select avg** *(balance)*

**from** *account*

**where** *branch\_name =* 'Perryridge'

Find the number of tuples in the

###### select count (\*)

**from** *customer*

Find the number of depositors in the bank

**select count (distinct** *customer\_name)*

**from** *depositor*

###### Aggregate Functions – Group By

**Find the number of depositors for each branch. select** *branch\_name,* **count (distinct** *customer\_name)*

**from** *depositor, account*

**where** *depositor.account\_number = account.account\_number*

**group by** *branch\_name*

###### Find the names of all branches where the average account balance is more than 1,200. select *branch\_name,* avg (*balance*)

**from** *account*

**group by** *branch\_name*

**having avg** (*balance*) *>* 1200

predicates in the **having** clause are applied after the formation of groups whereas predicates in the **where** clause are applied before forming groups

select count(ename),dno from emp group by dno;

###### Conclusion: We have Successfully Performed string manipulation operations and aggregate functions with group by.. Having clause using SQL Command Line Client.

###### Code and Output:

###### 

###### 

###### 

###### 

###### 