SQL

INTRODUCTION

LECTURE 11

Contents to be covered

Operators:

- IN Operators
- Between Operators
- Like Operators
- Orderby clause

Aggregate Functions:

- Avg
- Min
- Max
- Sum
- Count
- Group by Clause
- Group by with Having Clause

Scalar Functions

IN Operator

IN operator is used to check whether an attributes contains one or more specified values

Syntax:

```
AttributeName IN(Value1, Value2, Value3)
```

For example Query : List the employee residing in Delhi or Mumbai

```
SELECT * FROM Employee where Empcity IN('Delhi', 'Mumbai')
```

Example :IN Operator

SELECT * FROM Employee where Empcity IN('Delhi', 'Mumbai')

Relation: Employee

Employee_id	Employee_Name	Employee_City
E001	David	Delhi
E002	Peter	Delhi
E003	Jane	Mumbai
E004	Nammy	Patna
E005	John	Chennai

Query Result

Employee_id	Employee_Name	Employee_City
E001	David	Delhi
E002	Peter	Delhi
E003	Jane	Mumbai

Example :IN Operator

SELECT * FROM Employee where Empcity NOT IN('Delhi', 'Mumbai')

Relation: Employee

Employee_id	Employee_Name	Employee_City
E001	David	Delhi
E002	Peter	Delhi
E003	Jane	Mumbai
E004	Nammy	Patna
E005	John	Chennai

Query Result

Employee_id	Employee_Name	Employee_City
E004	Nammy	Patna
E005	John	Chennai

BETWEEN Operator

BETWEEN Operator is used to fetch the data lying within specified range

Syntax:

AttributeName BETWEEN Value1 AND Value2

For example: **List all the employees having salary within 20,000 to 50,000 range .**

Select * from employee where Emp_sal BETWEEN 20000 AND 50000;

Example :BETWEEN Operator

SELECT * FROM Employee where Emp_Sal BETWEEN 20000 AND 50000

Relation: Employee

Emp_id	Emp_Name	Emp_Sal
E001	David	24000
E002	Peter	30000
E003	Jane	15000
E004	Nammy	45000
E005	John	80000

Query Result

Emp_id	Emp_Name	Emp_Sal
E001	David	24000
E002	Peter	30000
E004	Nammy	45000

LIKE operator

LIKE operator is used for string comparison.

The operator “like” uses patterns that are described using two special characters:

- percent (%). The % character matches any substring.
- underscore (_). The _ character matches any character.

For Example: **Find the details of all employees whose name has the substring “an”.**

select * **from** ***Employee*** **where** Emp_Name **like** ‘% an%‘

Example :LIKE Operator

SELECT * FROM Employee where Emp_name like %an%

Relation: Employee

Emp_id	Emp_Name	Emp_Sal
E001	Mane	24000
E002	Peter	30000
E003	Jane	15000
E004	Taney	45000
E005	John	80000

Query Result

Emp_id	Emp_Name	Emp_Sal
E001	Mane	24000
E002	Jane	30000
E004	Tane	45000

Example :LIKE Operator

SELECT * FROM Employee where Emp_Sal like '_ane';

Relation: Employee

Emp_id	Emp_Name	Emp_Sal
E001	David	24000
E002	Peter	30000
E003	Jane	15000
E004	Nane	45000
E005	John	80000

Query Result

Emp_id	Emp_Name	Emp_Sal
E003	Jane	15000
E004	Nane	45000

ORDER BY operator

Orderby clause is used to arrange the elements of an attribute in ascending(asc) or descending order(desc)

For each attribute; ascending order is the default.

- Example: **order by** Emp_name **desc**

List the names of all employees in alphabetic order having salary more than 30000

select Emp_name from Employee where Emp_sal>30000 order by Emp_name

Aggregate Functions

Aggregate Functions

These functions operate on the multiset of values of a column of a relation, and return a value

avg: average value
min: minimum value
max: maximum value
sum: sum of values
count: number of values

Aggregate Functions (Cont.)

Find the average salary of employees working in Mumbai.

select avg (Emp_sal) **from** *employee* **where Emp_city** = 'Mumbai'

Find the number of records in the employee relation.

select count (*) from employee

Find the number of departments in the organisation.

select count (distinct Dept_name) from Employee

Aggregate Functions (Cont.)

Write a query to get the total salaries payable to employees.

SELECT SUM(salary) FROM employees;

Write a query to list the number of jobs available in the employees table.

SELECT COUNT(DISTINCT jobdesc) FROM employees;

Write a query to get the maximum salary of an employee working as a Programmer

SELECT MAX(salary) FROM employees WHERE jobdesc = 'Programmer';

Aggregate Functions (The Groupby Clause)

The function to divide the records into groups and returns an aggregate for each group.

Display the total employees department wise.

SELECT Emp_dept, count(*) as Emp_cnt FROM Employee group by Emp_Dept;

Emp_id	Emp_Name	Emp_Sal	Emp_Dept
E001	David	24000	IT
E002	Peter	30000	HR
E003	Jane	15000	IT
E004	Nane	45000	HR
E005	John	80000	HR
E006	Mane	30000	Mkt

Results

Emp_Dept	Emp_Cnt
IT	2
HR	3
Mkt	1

Aggregate Functions – Having Clause

Find the names of the cities where the average salary of employees is more than Rs 15000.

select *Emp_name,* **avg** (Emp_Sal) **from** *Employee* **group by** *Emp_name*
having avg (Emp_Sal) > 15000.

Scalar functions

These functions are based on user input, these too returns single value.

1. UCASE() : It converts the value of a field to uppercase
2. LCASE() : It converts the value of a field to lowercase
3. MID():The MID() function extracts texts from the text field

   Syntax: Select MID(Colname, start, length) from tablename;

   For example: Fetching first four characters of names of students from the Students table.

   Select MID(Name,1,4) from employee

Scalar functions

4. LEN(): It returns the length of the value in a text field
5. ROUND(): It returns the round off value upto certain decimal places.
   - Syntax: Select Round(Col,decimal) from tablename
   - For example: Select Round(Emp_sal,0) from employee;

NOW(): It returns current date and time.

Practice Exercise

Consider the patient relation consisting following attributes:

Attribute	Datatype	Details
id	numeric	patient id number
pname	character	patient name
p_age	numeric	patient age
gender	character	M/F
diagnosis	character	disease diagnosed
ccode	character	city code
cpaid	logical	consultation paid
nooftests	numeric	Number of test done

Practice Exercise(Contd..)

Query1: List the number of patients of each disease.

SELECT diagnosis, COUNT(*) FROM patient GROUP BY diagnosis

Query 2: List the number of female patients of each city.

SELECT ccode, COUNT(*) FROM patients WHERE gender="F" GROUP BY ccode

Practice Exercise(Contd..)

Query 3: List the average consultation fees paid for each diagnosis. The list should not contain diagnosis in which patients are less than 3.

SELECT AVG(cpaid), diagnosis FROM patients GROUP BY diagnosis HAVING COUNT(*) >= 3

Query 4: List the male patients of age more than 60 in ascending order of their names.

SELECT pname FROM patient WHERE gender="M" and p_age=60 ORDER BY name

Practice Exercise(Contd..)

Query 5: List the patients name who are belongs to delhi, Vadodara or Chennai and contain *kaur* in their names

SELECT pname from patients where ccode IN('Delhi',' Vadodara','Chennai') and pname LIKE %kaur%;

Query 6: Find the names of the patients who have paid the consultation ranging from 1000 to 10000 and number of test performed were 5.

Select pname from patients where nooftests=5 and cpaid between 1000 and 10000

Practice Exercise(Contd..)

Query 7: List the name of patient whose maximum test has done for diagnosing a disease.

SELECT pname, max(nooftests) from patient;

Query 8: how many boys of age below 10 are having diabetes.

SELECT COUNT(*) from patients where gender='M' and diagnosis='diabetes' and p_age<=10;