LAB NO.: 4 **Date:**

COMPLEX QUERIES ON SQL

Objectives:

In this lab, student will be able to:

- Use the concept of grouping and ordering of data.
- Implement Complex Queries
- Understand the concept of COMMIT and ROLLBACK.

GROUP BY:

There are circumstances where we would like to apply the aggregate function not only to a single set of tuples, but also to a group of sets of tuples, we specify this in SQL using the group by clause. The attribute or attributes given in the group by clause are used to form group. Tuples with the same value on all attributes in the group by clause are placed in one group.

Syntax:

SELECT columnname, columnname
FROM tablename
GROUP BY columnname;

Example:
SQL> SELECT EMPNO, SUM (SALARY) FROM EMP GROUP BY EMPNO;

JOIN with GROUP BY: This query is used to display a set of fields from two relations by matching a common field in them and group the corresponding records for each and every value of a specified key(s) while displaying.

Syntax: SELECT column_name, aggregate_function(column_name)
FROM relation_1,relation_2
WHERE relation_1.field_x=relation_2.field_y
GROUP BY field_z;

Example:

SQL> SELECT empno,SUM(SALARY) FROM emp,dept WHERE emp.deptno =20
GROUP BY empno;

GROUP BY - HAVING: At times it is useful to state a condition that applies to groups rather than to tuples. For example, we might be interested in only those employees where the count of orders is more than 10. This condition does not apply to a single tuple, rather it applies to each group constructed by the GROUP BY clause. HAVING clause is used to handle such cases.

Syntax: SELECT column_name, aggregate_function(column_name)
FROM table_name WHERE column_name operator value
GROUP BY column_name
HAVING aggregate_function(column_name) operator value;

Example:

```
SELECT Employees.LastName, COUNT(Orders.OrderID) AS NumberOfOrders
FROM (Orders INNER JOIN Employees ON Orders.EmployeeID=Employees.EmployeeID)
GROUP BY LastName
HAVING COUNT (Orders.OrderID) > 10;
```

ORDER BY: This query is used to display a selected set of fields from a relation in an ordered manner base on some field.

Syntax:

```
SELECT column_name
FROM table_name
ORDER BY column_name;
```

Example:

```
SQL> SELECT empno, ename, job
     FROM emp
     ORDER BY job;
```

JOIN with ORDER BY: This query is used to display a set of fields from two relations by matching a common field in them in an ordered manner based on some fields.

Syntax:

```
SELECT column_name1, column_name2
FROM relation_1, relation_2
WHERE relation_1.field_x = relation_2.field_y
ORDER BY field_z;
```

Example:

```
SQL> SELECT empno,ename,job,dname
     FROM emp,dept
     WHERE emp.deptno =dept.dept.no and emp.deptno = 20
     ORDER BY job;
```

WITH Clause: The WITH clause may be processed as an inline view or resolved as a temporary table. The advantage of the latter is that repeated references to the subquery may be more efficient as the data is easily retrieved from the temporary table, rather than being re-queried by each reference.

Syntax:

```
WITH <temporary table name> AS (
SELECT <attributes>
FROM   <table name>
```

GROUP BY <attribute>)
SELECT <attribute/s> from <tablename/s>

Example:

WITH temporaryTable(averageValue) as
(SELECT avg(Salary) from Employee),
SELECT EmployeeID,Name, Salary from Employee
WHERE Employee.Salary > temporaryTable.averageValue;

COMPLEX QUERIES (Derived relations/Subqueries in the from Clause):

SQL allows a subquery expression to be used in the from clause. The key concept applied here is that any select-from-where expression returns a relation as a result and, therefore, can be inserted into another select-from-where anywhere that a relation can appear.

Example:

select branch-name, avg-balance from (select branch-name, avg (balance) as avg-balance from account group by branch-name) where avg-balance > 1200

COMMIT AND ROLLBACK:

Changes due to insert, update, and delete statements are temporarily stored in the database system. They become permanent only after the COMMIT statement has been issued.

- You can use the SQL ROLLBACK statement to rollback (undo) any changes you made to the database before a COMMIT was issued.
- The SQL COMMIT statement saves any changes made to the database. When a COMMIT has been issued, all the changes since the last COMMIT, or since you logged on as the current user, are saved.

Example:

Implementing the save point

SQL> SAVEPOINT S23;

Save point created.

SQL> select * from employee;

EMPNO	ENAME	JOB	DEPTNO	SAL
1	Mathi	AP	1	10000
2	Arjun	ASP	2	15000
3	Gugan	ASP	1	15000
4	Karthik	Prof	2	30000

SQL> INSERT INTO EMPLOYEE VALUES(5,'Akalya','AP',1,10000);

1 row created.

SQL> select * from employee;

EMPNO	ENAME	JOB	DEPTNO	SAL
1	Mathi	AP	1	10000
2	Arjun	ASP	2	15000
3	Gugan	ASP	1	15000
4	Karthik	Prof	2	30000
5	Akalya	AP	1	10000

Implementing the rollback

SQL> rollback;

SQL> select * from employee;

EMPNO	ENAME	JOB	DEPTNO	SAL
1	Mathi	AP	1	10000
2	Arjun	ASP	2	15000
3	Gugan	ASP	1	15000
4	Karthik	Prof	2	30000

Implementing the commit

SQL> COMMIT;

Commit complete.

LAB EXERCISE:

Implement the following Queries on UNIVERSITY Database.

Group By:

1. Find the number of students in each course.
2. Find those departments where the average number of students are greater than 10.
3. Find the total number of courses in each department.
4. Find the names and average salaries of all departments whose average salary is greater than 42000.
5. Find the enrolment of each section that was offered in Spring 2009.

Ordering the display of Tuples (Use ORDER BY ASC/DESC):

6. List all the courses with prerequisite courses, then display course id in increasing order.
7. Display the details of instructors sorting the salary in decreasing order.

Derived Relations:

8. Find the maximum total salary across the departments.
9. Find the average instructors' salaries of those departments where the average salary is greater than 42000.
10. Find the sections that had the maximum enrolment in Spring 2010
11. Find the names of all instructors who teach all students that belong to 'CSE' department.
12. Find the average salary of those department where the average salary is greater than 50000 and total number of instructors in the department are more than 5.

With Clause:

13. Find all departments with the maximum budget.

14. Find all departments where the total salary is greater than the average of the total salary at all departments.
15. Find the sections that had the maximum enrolment in Fall 2009
16. Select the names of those departments where the total credits earned by all the students is greater than the total credits earned by all the students in the Finance Department

(Use ROLLBACK (and SAVEPOINT) to undo the effect of any modification on database before COMMIT)

17. Delete all the instructors of Finance department.
18. Delete all courses in CSE department.
19. Transfer all the students from CSE department to IT department.
20. Increase salaries of instructors whose salary is over $100,000 by 3%, and all others receive a 5% raise
21. Add all instructors to the student relation with tot_creds set to 0.
22. Delete all instructors whose salary is less than the average salary of instructors.