**SQL Job Preparation Assignment 5**

1. **Write a query to DISPLAY THE "DEPTNO" AND "SUM OF SALARY" FOR EACH DEPTNO**

**SELECT** DEPTNO, **SUM**(SALARY) **FROM** EMPLOYEES **GROUP BY** DEPTNO;

This query will display the "DEPTNO" and the "SUM of SALARY" for each DEPTNO in the EMPLOYEES table. The GROUP BY clause groups the rows in the table by the DEPTNO column, and the SUM() function calculates the total salary for each group.

-----------------------------------------------------------------------------------------------------------------------

1. **Define INNER JOIN and OUTER JOIN, then use a query to demonstrate each.**

An INNER JOIN in SQL is used to combine rows from two or more tables based on a related column between them. It only returns rows where there is a match in both tables.

An OUTER JOIN, on the other hand, returns all rows from one table and the matching rows from the second table. If there is no match, NULL values will be returned for the right table.

An INNER JOIN query would look like this:

SELECT \*

FROM table1

INNER JOIN table2

ON table1.column = table2.column;

An OUTER JOIN query would look like this:

SELECT \*

FROM table1

LEFT OUTER JOIN table2

ON table1.column = table2.column;

OR

SELECT \*

FROM table1

RIGHT OUTER JOIN table2

ON table1.column = table2.column;

For example, if we have two tables "departments" and "employees", with a common column "dept\_no" in both tables, we can use the INNER JOIN to display the department name and employee name who work in that department:

SELECT departments.dept\_name, employees.name

FROM departments

INNER JOIN employees

And we can use the LEFT OUTER JOIN to display all the department name and the employee name if exist, otherwise it will display null if there is no employee in that department:

ON departments.dept\_no = employees.dept\_no;

SELECT departments.dept\_name, employees.name

FROM departments

LEFT OUTER JOIN employees

ON departments.dept\_no = employees.dept\_no;

Please note that the above queries may not work as is, as it depends on the actual structure of your tables and their relations.

-------------------------------------------------------------------------------------------------------------------------

1. **WHAT DO YOU MEAN BY VIRTUAL TABLES? as well as how to make one.**

A virtual table, also known as a "view," is a virtual representation of one or more tables in a database. It does not store data itself, but instead retrieves data from one or more underlying tables each time it is queried. Whereas on the other hand a virtual table can be used to simplify the structure of a database, to limit the amount of data that is exposed to users, or to create a customized view of the data in a table.

To create a virtual table, you use the "CREATE VIEW" statement in SQL. The basic syntax is:

CREATE VIEW view\_name AS

SELECT column1, column2, ...

FROM table1

WHERE condition;

For example, if we have a table "employees" and we want to create a virtual table that only displays the name and salary of employees whose salary is greater than 50000, we can use the following query:

CREATE VIEW high\_salary\_employees AS

SELECT name, salary

FROM employees

WHERE salary > 50000;

Once the view is created, you can query it just like a regular table:

You can also use the virtual table in JOIN, UPDATE and DELETE statements, but keep in mind that any update will affect the underlying table as well.

It is important to note that the data in a view is not stored physically, so it can't be updated. However, it can be used to read data from the underlying tables.

--------------------------------------------------------------------------------------------------------------------------

1. **Rewrite the below query using Subqueries. SELECT \* FROM EMP WHERE SAL=MAX(SAL)**

The query can be rewritten using a subquery as follows:

**SELECT \* FROM EMP**

**WHERE SAL = (SELECT MAX(SAL) FROM EMP);**

This query first finds the maximum salary in the "EMP" table using a subquery in the SELECT clause. The subquery returns the maximum salary as a single value, which is then compared to the "SAL" column in the main query. The main query returns all rows from the "EMP" table where the "SAL" column matches the maximum salary returned by the subquery.

Alternatively, you can also use the correlated subquery as follows:

**SELECT \* FROM EMP e1**

**WHERE SAL = (SELECT MAX(e2.SAL) FROM EMP e2 WHERE e1.SAL = e2.SAL);**

The correlated subquery is executed for each row of the outer query and it filters the rows of the inner query using the values from the outer query.

In both the above queries, the subquery is executed first and it returns a single value, which is then used by the main query to filter the rows of the table.