Experiment no: - 3

Experiment Name: - Ret rival Commands-II

Aim: - Performing practical by using multiple tables and group functions.

Resource required: - Oracle 9i - iSQLplus

Theory: -

- **DISPLAYING DATA FROM MULTIPLE TABLES**:

Cartesian Products: It is formed when:

- a join condition is omitted
- a join condition is invalid
- All rows in the first table are joined to all rows in the second table.

Syntax: Joining tables

```
SELECT table1.column, table2.column
FROM table1, table2
WHERE table1.column1= table2.column2;
```

- write the join condition in the WHERE clause.

1. Retrieving record with **Equijoin/simple join/ inner join**:

```
e.g.: SELECT employees.employee_id, employees.last_name,
              employees.department_id, departments.department_id,
              departments.location_id
      FROM employees, departments
      WHERE employees.department_id = departments.department_id;
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT-ID	DEPARTMENT-ID	LOCATION_ID
200	Whalen	10	10	1700
201	Hartstein	20	20	1800
202	Fay	30	30	1500

2. Retrieving record with **Non-Equijoins**: contain something other than equality operator.

```
e.g.: SELECT e.last_name, e.salary, j.grade_level
      FROM employees e, job_grades j
      WHERE e.salary
       BETWEEN j.lowest_sal AND j.highest_sal;
```

LAST_NAME	SALARY	GRA
Matros	2500	A
Vargas	2600	A
Lorentz	4200	B

3. **Outer join:** the missing rows can be returned if an outer join operator (+) is used in join condition.

Syntax: SELECT table1.column, table2.column
FROM table1, table2
WHERE table1.column (+) = table2.column;

e.g.: SELECT e.last-name, e.department_id, d.department_name
FROM employees e, departments d
WHERE e.department_id(+) = d.department_id;

LAST_NAME	DEPARTMENT-ID	DEPARTMENT_NAME
Whalen	10	Administration
Matos	50	Shipping
		Contracting

>> LEFT OUTER JOIN

e.g.: SELECT e.last_name, e.department-id, and d.department_name
FROM employees e
LEFT OUTER JOIN departments d
ON (e. department_id = d.department_id);

- ON clause used to join specify column.

>> RIGHT OUTER JOIN

e.g.: SELCET e.last_name, e.department_id, d.department_name
FROM employees e
RIGHT OUTER JOIN departments d
ON (e.department_id = d.department_id);

>> FULL OUTER JOIN

e.g.: SELECT e.last_name, e.department-id, and d.department_name
FROM employees e
FULL OUTER JOIN departments d
ON (e. department_id = d.department_id);

4. **Cross joins:** produces the cross product of two tables.

e.g.: SELECT last_name, department_name
FROM employees
CROSS JOIN departments;

LAST_NAME	DEPARTMENT_NAME
King	Anministration
Kochhar	Administration
De Haan	Administration

5. **Natural Joins:** select record from tables that have equal values in all matched columns.

e .g.: SELECT department_id, department_name‟ location-id, city
FROM departments
NATURAL JOIN locations;

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID	CITY
60	IT	1400	Southlake
50	Shipping	1500	Seattle
90	Executive	1700	Seattle

- **AGGREGATING DATA USING GROUP FUNCTIONS:**

Group Functions: It operate on sets of rows to give one result per group.
Syntax: SELECT [column,] group_function (column), …
FROM table
[WHERE condition]
[GROUP BY column]
[ORDER BY column];

- Types of Group Functions:
1. AVG, COUNT, MAX, MIN, STDDEV, SUM, and VARIANCE
e.g.: SELECT AVG (salary), MAX (salary), MIN (salary), and SUM (salary)
FROM employees
WHERE job_id LIKE „%REP%“;

AVG (SALARY)	MAX (SALARY)	MIN (SALARY)	SUM (SALARY)
8150	11000	6000	32600

2. COUNT (*): returns the number of rows in a table.
e.g.: SELECT COUNT (*)
FROM employees
WHERE department_id = 50;

3. DISTINCT: returns the number of distinct non-null values.
e.g.: SELECT COUNT (DISTINCT department_id)
FROM employees;

Others function with group functions:
- NVL function: forces group functios to include null values.
e.g.: SELECT AVG (NVL (commission_pct, 0))
FROM employees;

- Creating groups of data using **GROUP BY** clause:
Syntax: SELECT column, group_function (column)
From table
[WHERE condition]
[GROUP BY group_by_expression]
[ORDER BY column];
e.g.: SELECT AVG (salary)
FROM employees

GROUP BY department_id;

- The **HAVING** Clause: to restrict groups Syntax:

```
SELECT column,
group_function
        From
        table
        [WH
        ERE
        condi
        tion]
        [GROUP BY
                        group_
        by_expression]
        [HAVING
        group_condition]
        [ORDER BY     column];
```

e.g.:
```
SELECT department_id,
    MAX (salary) FROM
    employees
   GROUP BY
    department_id
    HAVING MAX
    (salary)>10000;
```

- Displays department numbers and maximum salaries for those departments whose maximum salary is greater than $10,000.

- Nesting Group Functions:

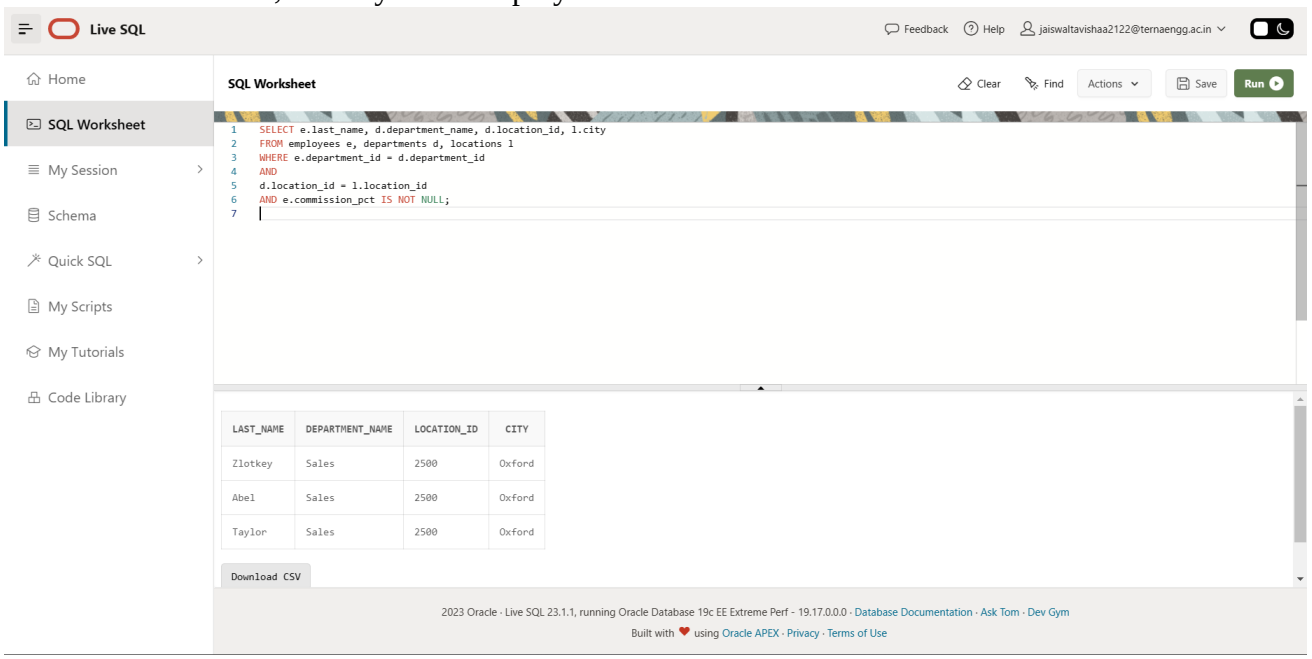
e.g.:
```
SELECT MAX (AVG (salary))
    FROM employees
   GROUP BY department_id;
```

Conclusion:

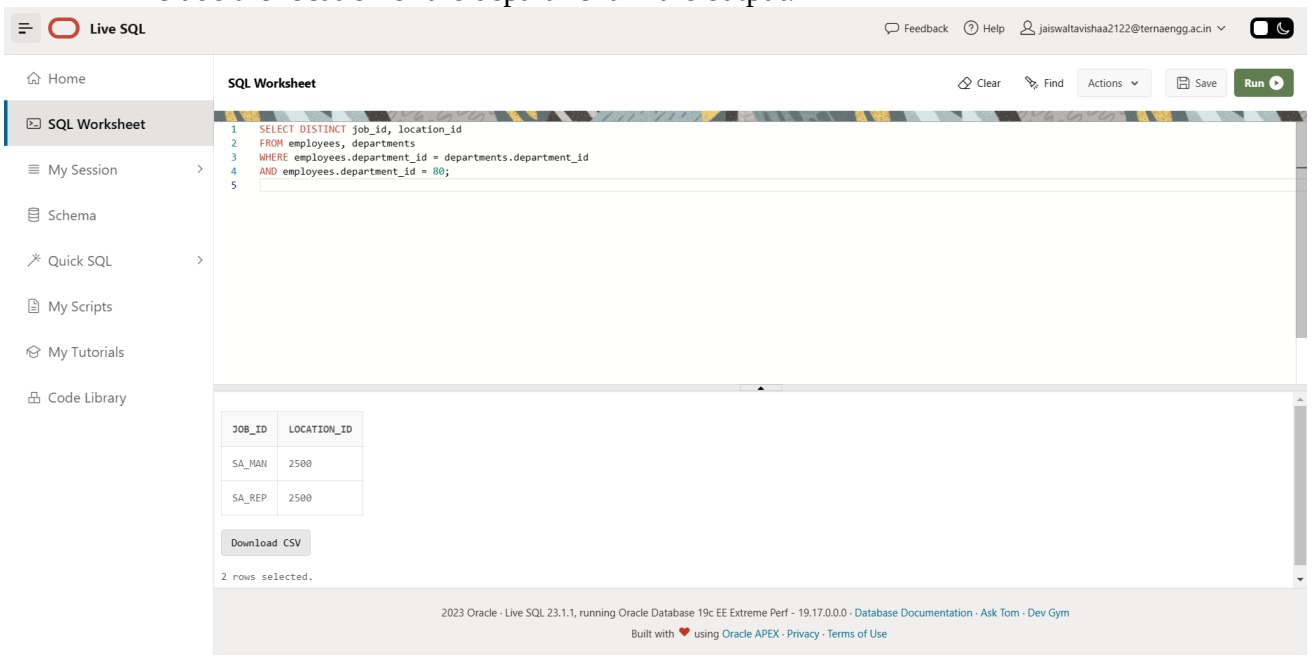
You should have learned how to use joins to display data from multiple tables and group functions.

LAB ASSIGNMENT -3

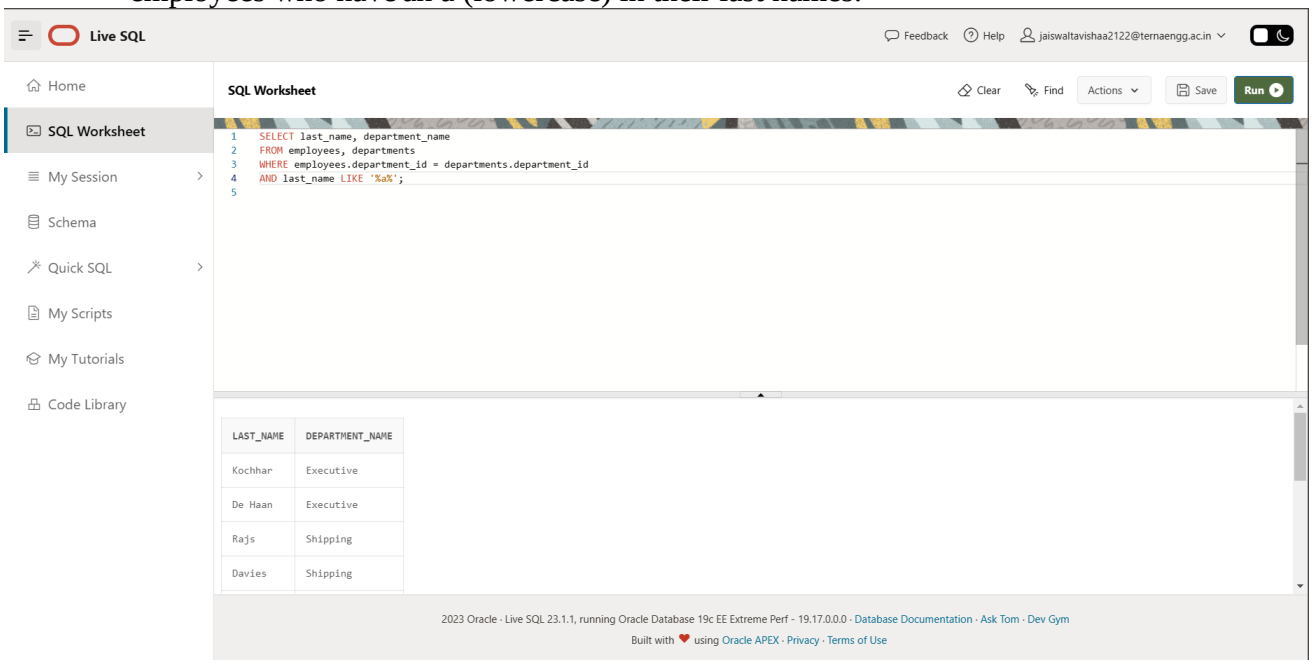
1. Write a query to display the employee last name, department name, location ID, and city of all employees who earn a commission



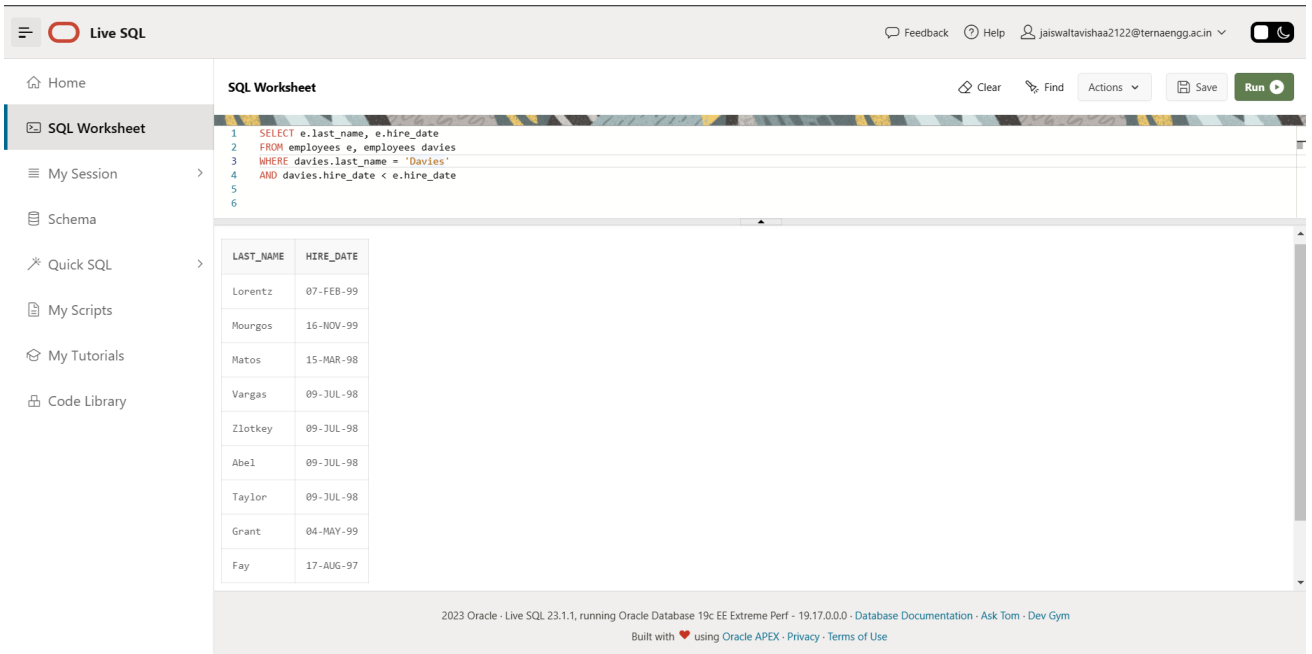
2. Create a unique listing of all jobs that are in department 80. Include the location of the department in the output.



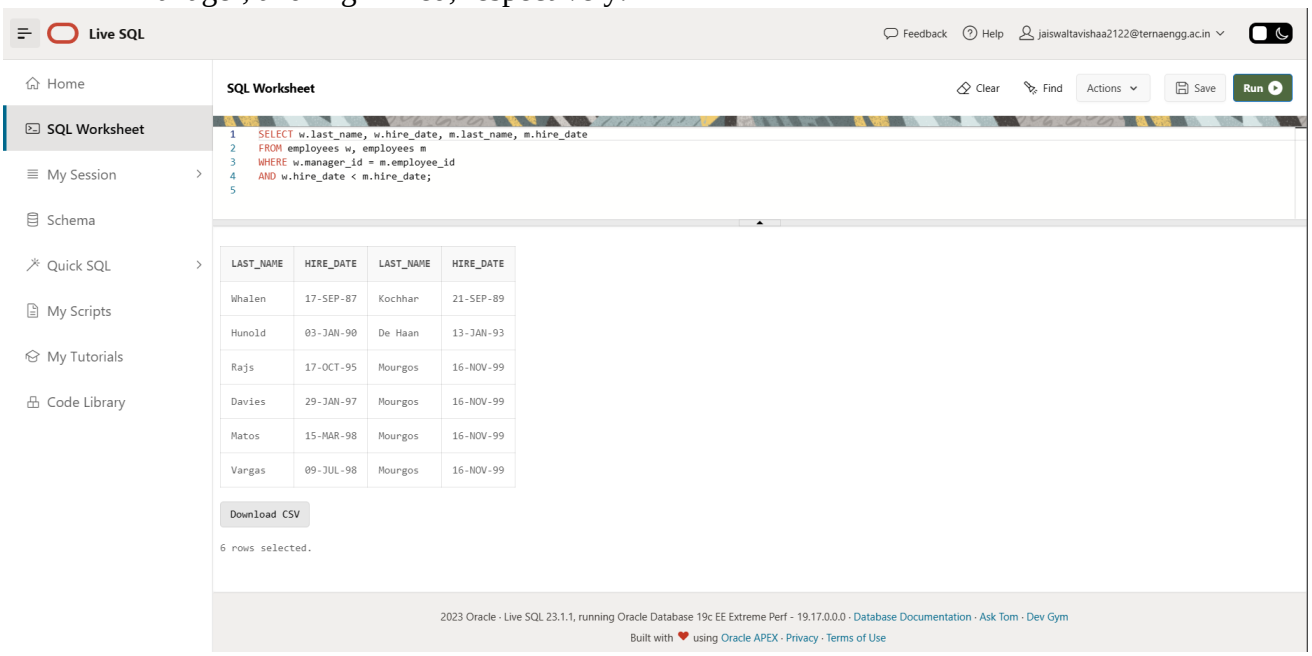
3. Display the employee last name and department name for all employees who have an a (lowercase) in their last names.



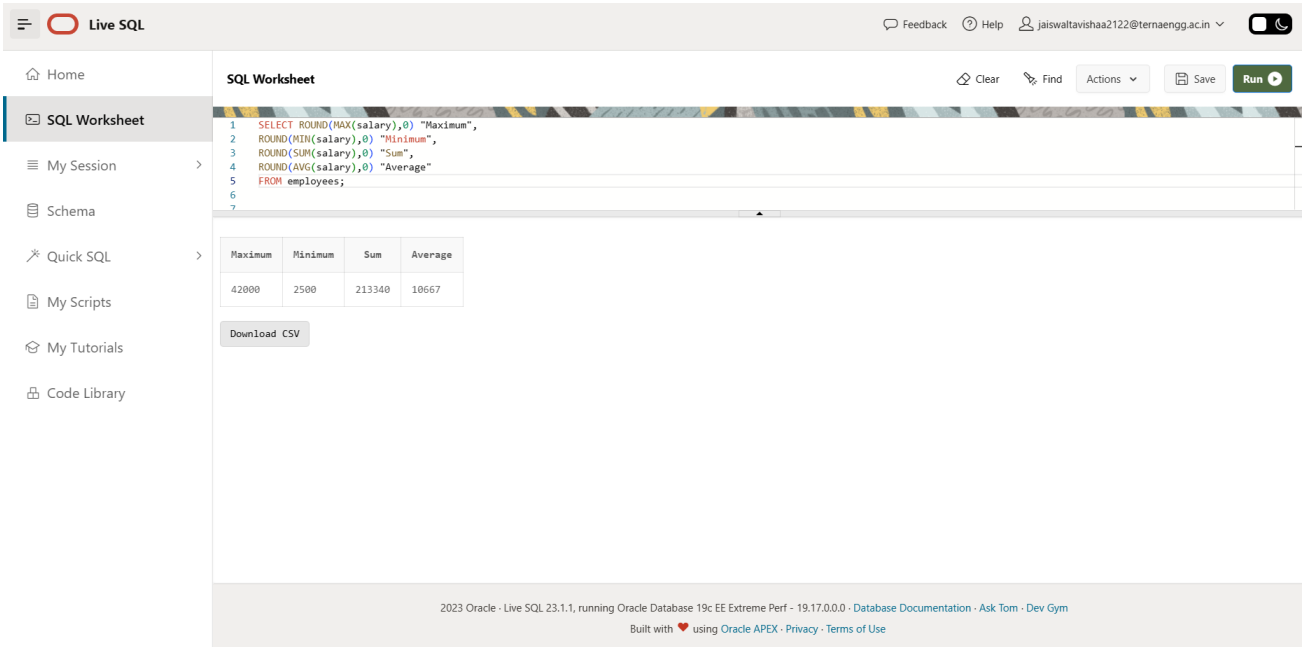
4. Create a query to display the name and hire date of any employee hired after employee Davies.



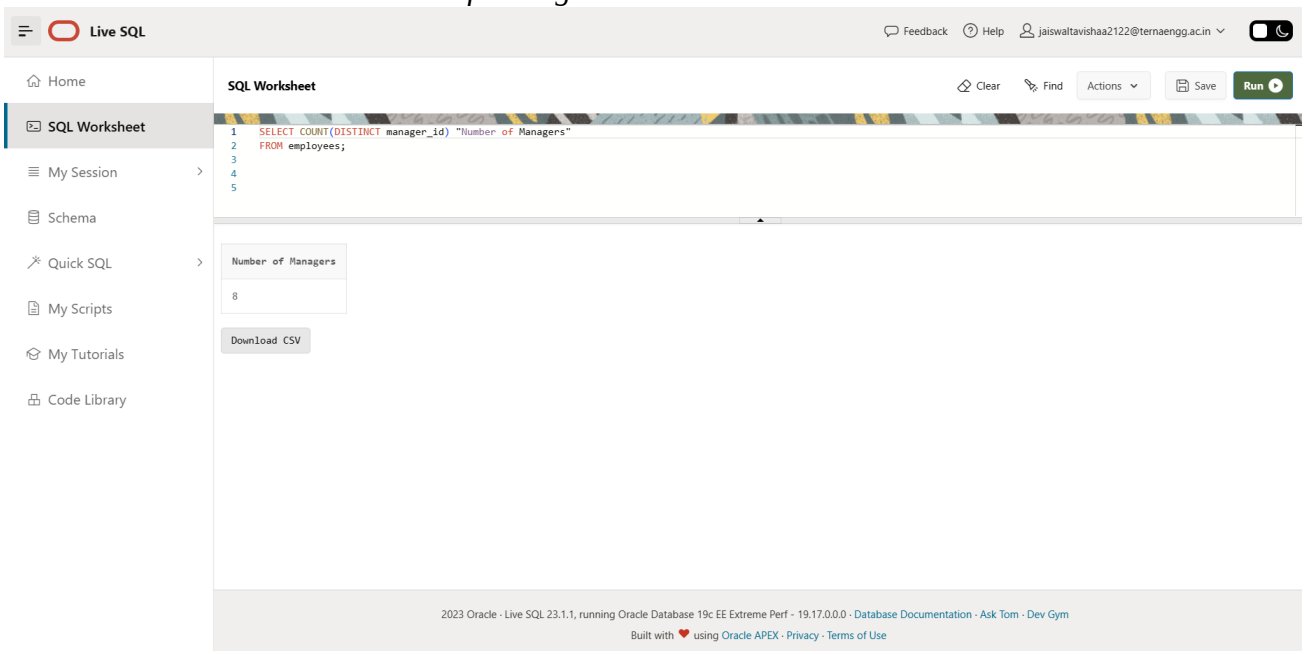
5. Display the names and hire date for all employees who were hired before their managers, along with their manager‟s names and hire dates. Label the column Employee, EMP, Hired, Manager, and Mgr Hired, respectively.



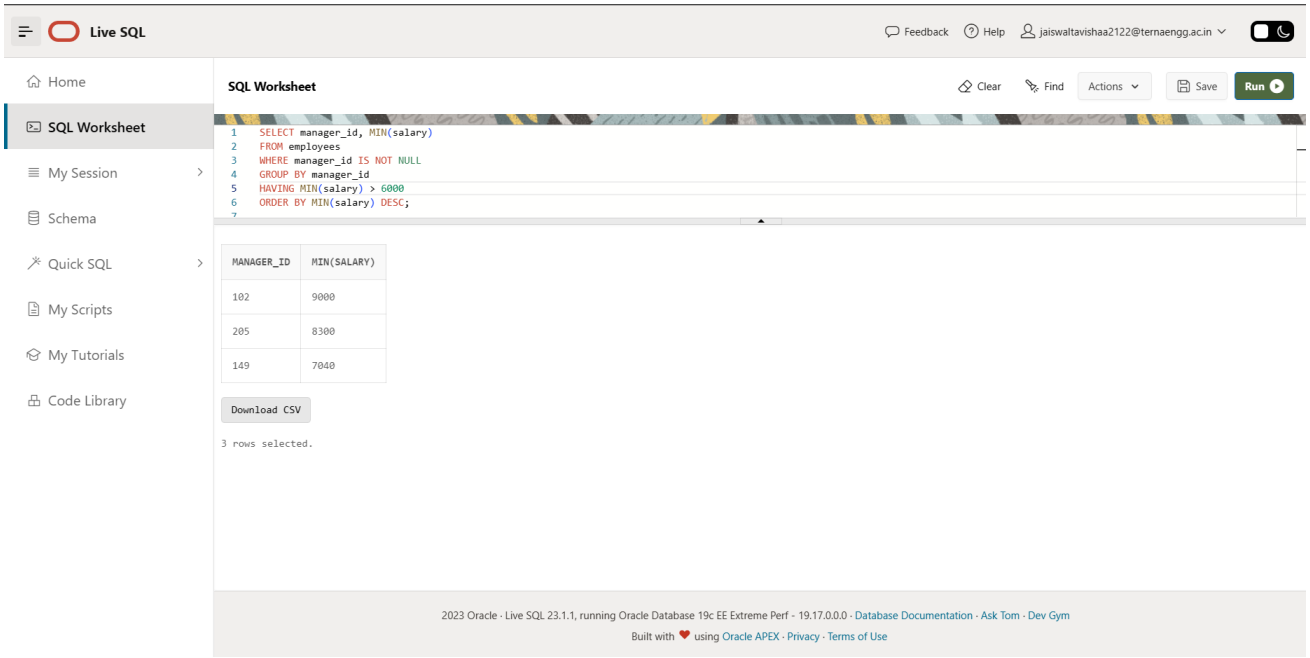
6. Display the highest, lowest, sum, and average salary of all employees. Label the columns Maximum, Minimum, Sum, and Average, respectively. Round your results to the nearest whole number.



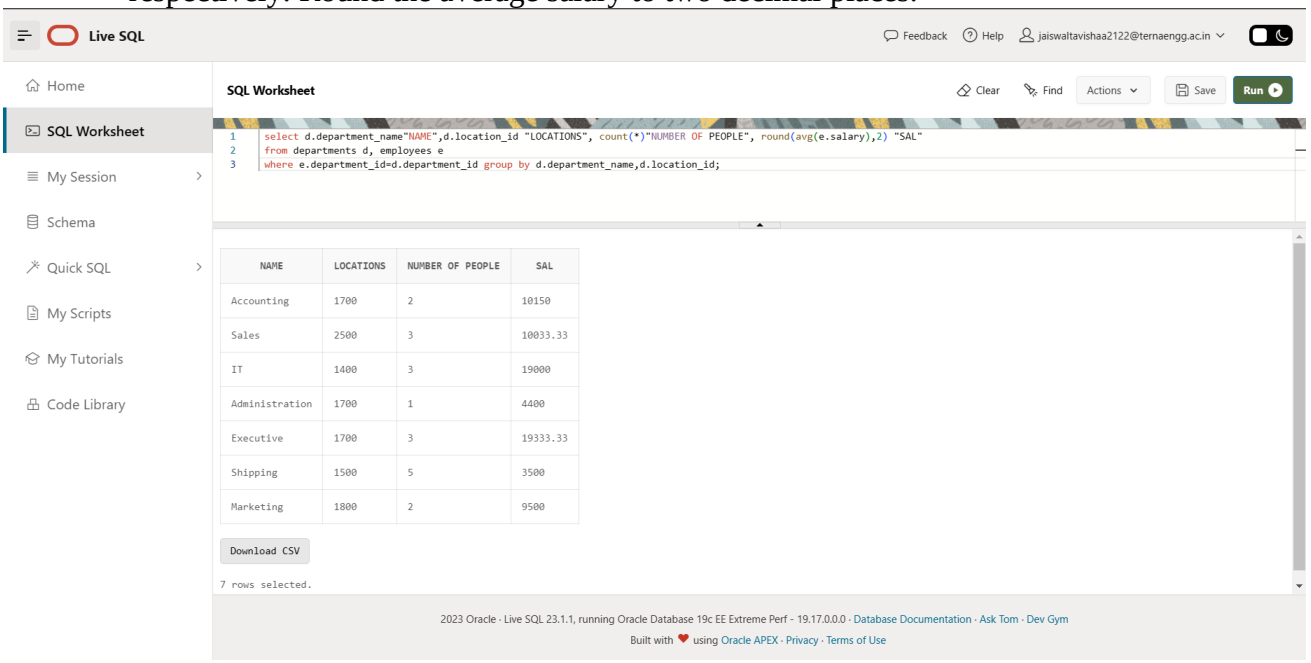
7. Determine the number of managers without listing them. Label the column Number of Managers. *Hint: Use the MANAGER_ID column to determine the number of managers.*



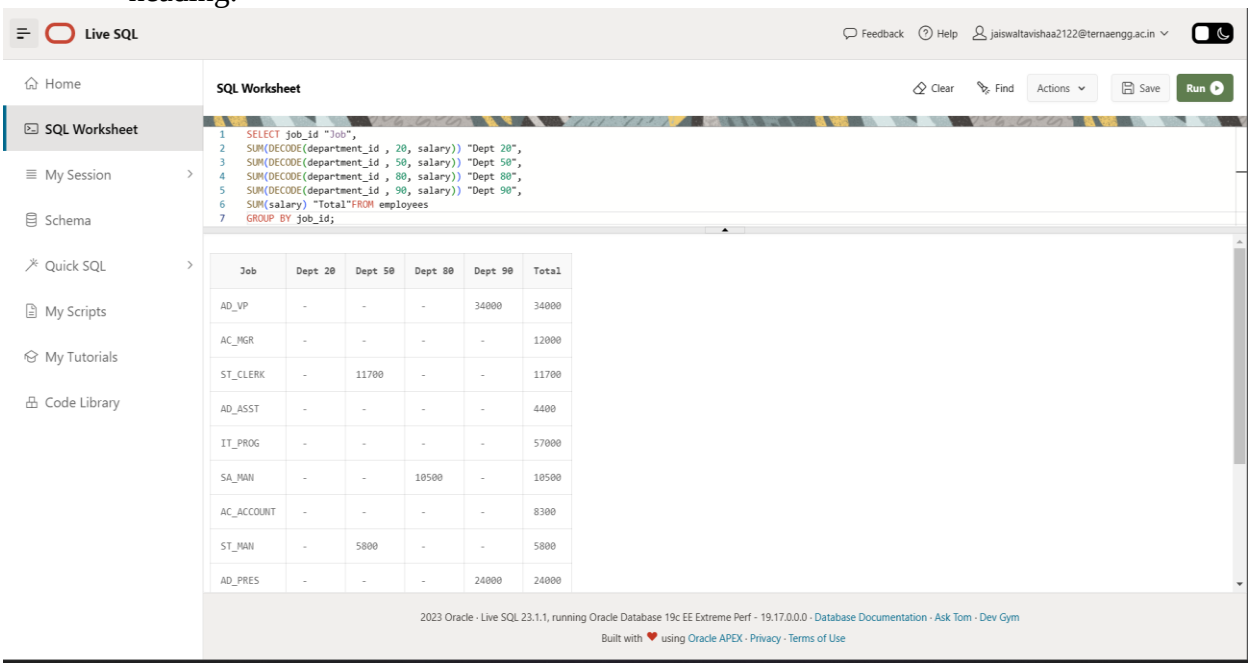
8. Display the manager number and the salary of the lowest paid employee for that manager. Exclude anyone whose manager is not know. Exclude any group where the minimum salary is $6,000 or less. Sort the output in descending order of salary.

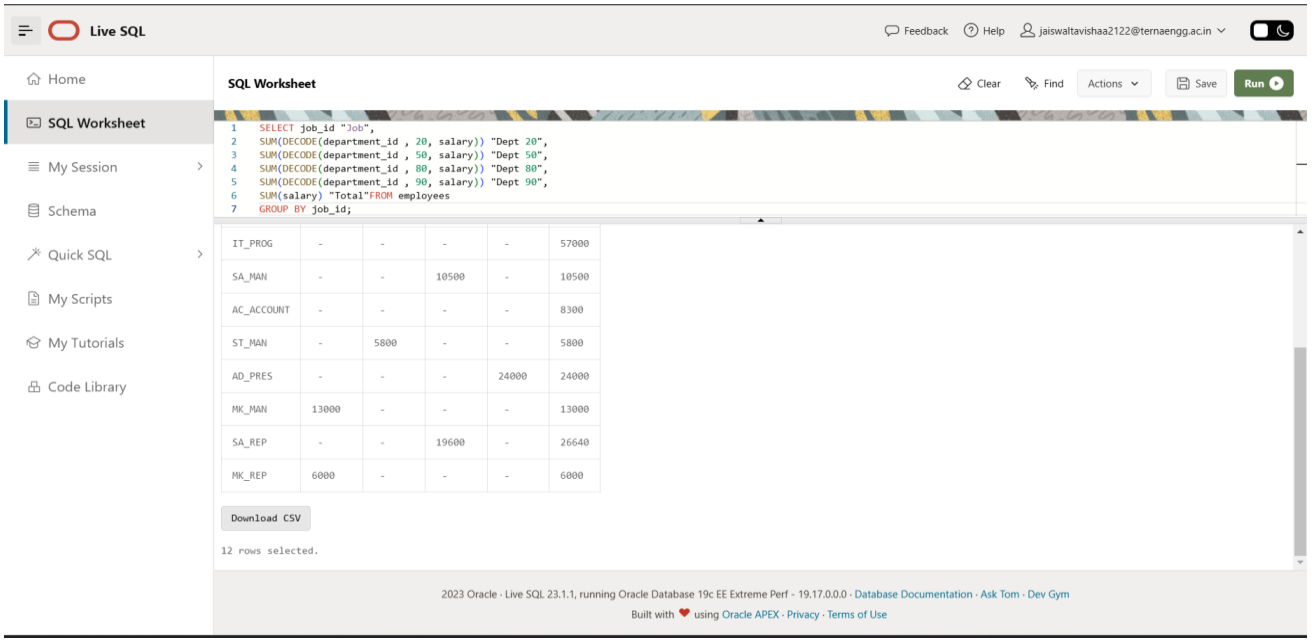


9. Write a query to display each department‟s name, location, number of employees, and the average salary for all employees in that department. Label the columns Name, Location, Number of People, and Salary, respectively. Round the average salary to two decimal places.



10. Create a matrix query to display the job, the salary for that job based on department number, and the total salary for that job, for departments 20, 50, 80, and 90, giving each column an appropriate heading.




Live SQL
Feedback
Help
jaiswaltavishaa2122@ternaengg.ac.in
Home
SQL Worksheet
My Session
Schema
Quick SQL
My Scripts
My Tutorials
Code Library
SQL Worksheet
Clear
Find
Actions
Save
Run
1 SELECT job_id "Job",
2 SUM(DECODE(department_id , 20, salary)) "Dept 20",
3 SUM(DECODE(department_id , 50, salary)) "Dept 50",
4 SUM(DECODE(department_id , 80, salary)) "Dept 80",
5 SUM(DECODE(department_id , 90, salary)) "Dept 90",
6 SUM(salary) "Total"FROM employees
7 GROUP BY job_id;
IT_PROG - - - - 57000
SA_MAN - - 10500 - 10500
AC_ACCOUNT - - - - 8300
ST_MAN - 5800 - - 5800
AD_PRES - - - 24000 24000
MK_MAN 13000 - - - 13000
SA_REP - - 19600 - 26640
MK_REP 6000 - - - 6000
Download CSV
12 rows selected.
2023 Oracle · Live SQL 23.1.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 · Database Documentation · Ask Tom · Dev Gym
Built with using Oracle APEX · Privacy · Terms of Use