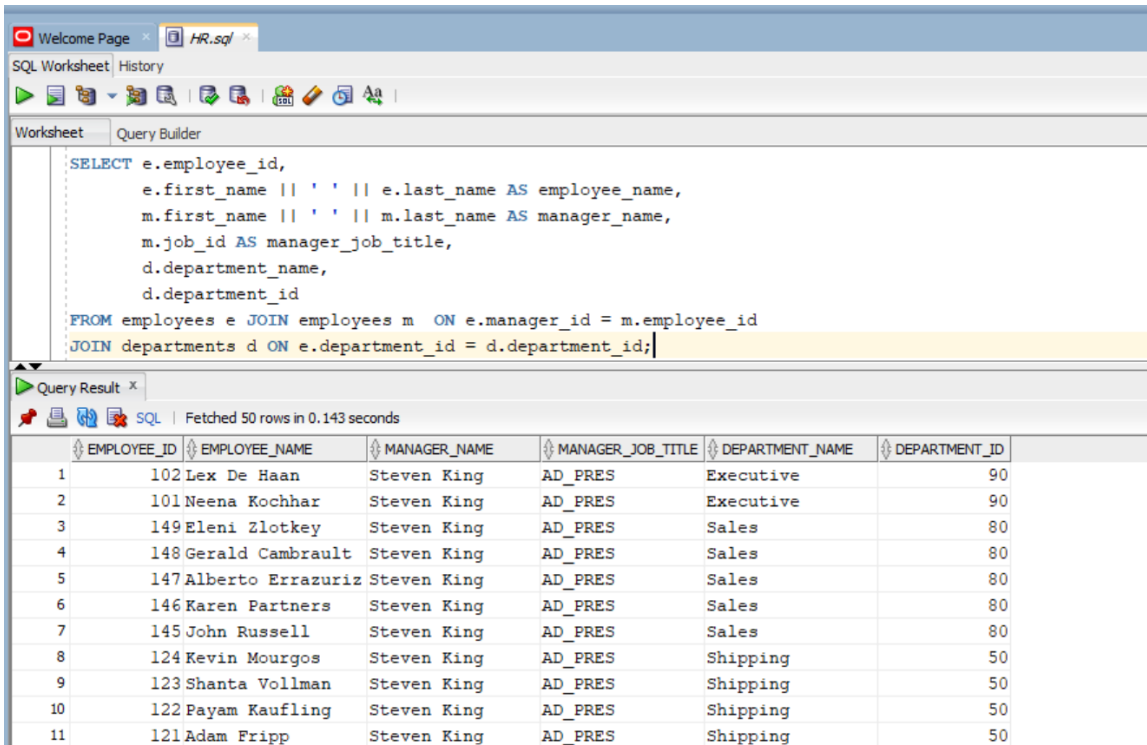


## Lab 5 - INLAB

23K-0594

### TASK 1:

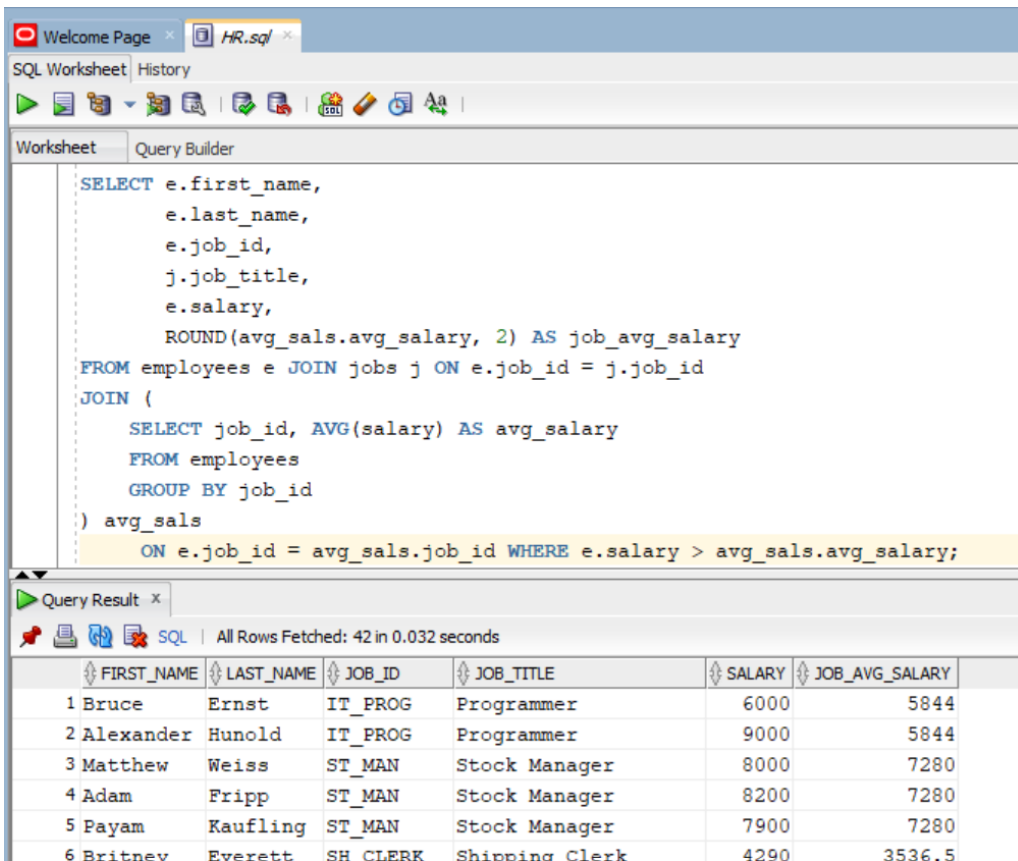


The screenshot shows a SQL Worksheet interface with a query and its results. The query is a JOIN between employees, managers, and departments. The results table has 11 rows and 6 columns: EMPLOYEE\_ID, EMPLOYEE\_NAME, MANAGER\_NAME, MANAGER\_JOB\_TITLE, DEPARTMENT\_NAME, and DEPARTMENT\_ID.

```
SELECT e.employee_id,
       e.first_name || ' ' || e.last_name AS employee_name,
       m.first_name || ' ' || m.last_name AS manager_name,
       m.job_id AS manager_job_title,
       d.department_name,
       d.department_id
FROM employees e JOIN employees m ON e.manager_id = m.employee_id
JOIN departments d ON e.department_id = d.department_id;
```

EMPLOYEE_ID	EMPLOYEE_NAME	MANAGER_NAME	MANAGER_JOB_TITLE	DEPARTMENT_NAME	DEPARTMENT_ID
1	102 Lex De Haan	Steven King	AD_PRES	Executive	90
2	101 Neena Kochhar	Steven King	AD_PRES	Executive	90
3	149 Eleni Zlotkey	Steven King	AD_PRES	Sales	80
4	148 Gerald Cambrault	Steven King	AD_PRES	Sales	80
5	147 Alberto Errazuriz	Steven King	AD_PRES	Sales	80
6	146 Karen Partners	Steven King	AD_PRES	Sales	80
7	145 John Russell	Steven King	AD_PRES	Sales	80
8	124 Kevin Mourgos	Steven King	AD_PRES	Shipping	50
9	123 Shanta Vollman	Steven King	AD_PRES	Shipping	50
10	122 Payam Kaufling	Steven King	AD_PRES	Shipping	50
11	121 Adam Fripp	Steven King	AD_PRES	Shipping	50

### Task 2:



The screenshot shows a SQL Worksheet interface with a query and its results. The query is a JOIN between employees, jobs, and a subquery for average salary by job\_id. The results table has 6 rows and 6 columns: FIRST\_NAME, LAST\_NAME, JOB\_ID, JOB\_TITLE, SALARY, and JOB\_AVG\_SALARY.

```
SELECT e.first_name,
       e.last_name,
       e.job_id,
       j.job_title,
       e.salary,
       ROUND(avg_sals.avg_salary, 2) AS job_avg_salary
FROM employees e JOIN jobs j ON e.job_id = j.job_id
JOIN (
    SELECT job_id, AVG(salary) AS avg_salary
    FROM employees
    GROUP BY job_id
) avg_sals
ON e.job_id = avg_sals.job_id WHERE e.salary > avg_sals.avg_salary;
```

FIRST_NAME	LAST_NAME	JOB_ID	JOB_TITLE	SALARY	JOB_AVG_SALARY
1 Bruce	Ernst	IT_PROG	Programmer	6000	5844
2 Alexander	Hunold	IT_PROG	Programmer	9000	5844
3 Matthew	Weiss	ST_MAN	Stock Manager	8000	7280
4 Adam	Fripp	ST_MAN	Stock Manager	8200	7280
5 Payam	Kaufling	ST_MAN	Stock Manager	7900	7280
6 Britnev	Everett	SH_CLERK	Shipping Clerk	4290	3536.5

### Task 3:

The screenshot shows a SQL Worksheet interface with a query in the 'Worksheet' tab. The query is as follows:

```
SELECT d.department_name,  
       l.city,  
       COUNT(e.employee_id) AS num_employees FROM departments d  
JOIN locations l ON d.location_id = l.location_id  
JOIN employees e ON d.department_id = e.department_id  
WHERE l.city = 'London' GROUP BY d.department_name, l.city HAVING COUNT(e.employee_id) > 5;
```

The 'Query Result' tab shows the following columns: DEPARTM..., CITY, and NUM\_EMP... The status bar indicates 'All Rows Fetched: 0 in 0.032 seconds'.

### Task 4:

The screenshot shows a SQL Worksheet interface with a query in the 'Worksheet' tab. The query is as follows:

```
SELECT e.first_name || ' ' || e.last_name AS employee_name,  
       m.first_name || ' ' || m.last_name AS manager_name,  
       e.hire_date AS employee_hire_date,  
       m.hire_date AS manager_hire_date  
FROM employees e JOIN employees m ON e.manager_id = m.employee_id  
WHERE EXTRACT(YEAR FROM e.hire_date) = EXTRACT(YEAR FROM m.hire_date);
```

The 'Query Result' tab shows a table with 4 columns: EMPLOYEE\_NAME, MANAGER\_NAME, EMPLOYEE\_HIRE\_DATE, and MANAGER\_HIRE\_DATE. The status bar indicates 'All Rows Fetched: 17 in 0.015 seconds'.

	EMPLOYEE_NAME	MANAGER_NAME	EMPLOYEE_HIRE_DATE	MANAGER_HIRE_DATE
1	Payam Kaufling	Steven King	01-MAY-03	17-JUN-03
2	Valli Pataballa	Alexander Hunold	05-FEB-06	03-JAN-06
3	Daniel Faviet	Nancy Greenberg	16-AUG-02	17-AUG-02
4	Alexis Bull	Adam Fripp	20-FEB-05	10-APR-05
5	James Marlow	Adam Fripp	16-FEB-05	10-APR-05
6	Mozhe Atkinson	Adam Fripp	30-OCT-05	10-APR-05
7	Laura Bissot	Adam Fripp	20-AUG-05	10-APR-05
8	Britney Everett	Shanta Vollman	03-MAR-05	10-OCT-05
9	Stephen Stiles	Shanta Vollman	26-OCT-05	10-OCT-05
10	Donald OConnell	Kevin Mourgos	21-JUN-07	16-NOV-07
11	Louise Doran	Karen Partners	15-DEC-05	05-JAN-05
12	Lindsey Smith	Karen Partners	10-MAR-05	05-JAN-05

## Task 5:

The screenshot shows an SQL Worksheet interface with a query editor and a query result pane. The query is designed to find the department with the highest total salary by joining the employees and departments tables, grouping by department name, and ordering by total salary in descending order. The result pane shows one row: Sales with a total salary of 308235.

```
SELECT department_name, total_salary
FROM (
    SELECT d.department_name,
           SUM(e.salary) AS total_salary
    FROM employees e
    JOIN departments d
      ON e.department_id = d.department_id
    GROUP BY d.department_name
    ORDER BY total_salary DESC
) WHERE ROWNUM = 1;
```

Query Result x

All Rows Fetched: 1 in 0.008 seconds

DEPARTMENT_NAME	TOTAL_SALARY
1 Sales	308235

## Task 6:

The screenshot shows an SQL Worksheet interface with a query editor and a query result pane. The query joins the employees, employee\_projects, and projects tables to find employees who have worked on projects for less than 50 hours in the year 2023. The result pane shows two rows: Neena Kochhar with 20 hours on the AI System project, and Steven King with 30 hours on the AI System project.

```
SELECT e.first_name || ' ' || e.last_name AS employee_name,
       p.project_name,
       SUM(ep.hours_worked) AS total_hours
FROM employees e
JOIN employee_projects ep ON e.employee_id = ep.emp_id
JOIN projects p ON ep.project_id = p.project_id
WHERE TO_CHAR(p.start_date, 'YYYY') = '2023'
GROUP BY e.first_name || ' ' || e.last_name, p.project_name
HAVING SUM(ep.hours_worked) < 50;
```

Query Result x

All Rows Fetched: 2 in 0.024 seconds

EMPLOYEE_NAME	PROJECT_NAME	TOTAL_HOURS
1 Neena Kochhar	AI System	20
2 Steven King	AI System	30

## Task 7:

The screenshot shows the SQL Developer interface with a query window titled 'HR.sql'. The query is as follows:

```
SELECT p.project_name,  
       COUNT(DISTINCT ep.emp_id) AS num_employees FROM projects p  
JOIN employee_projects ep ON p.project_id = ep.project_id  
GROUP BY p.project_name HAVING COUNT(DISTINCT ep.emp_id) = (  
    SELECT MAX(COUNT(DISTINCT emp_id))  
    FROM employee_projects  
    GROUP BY project_id  
);
```

The query result is displayed below the editor, showing one row:

PROJECT_NAME	NUM_EMPLOYEES
1 AI System	2

## Task 8:

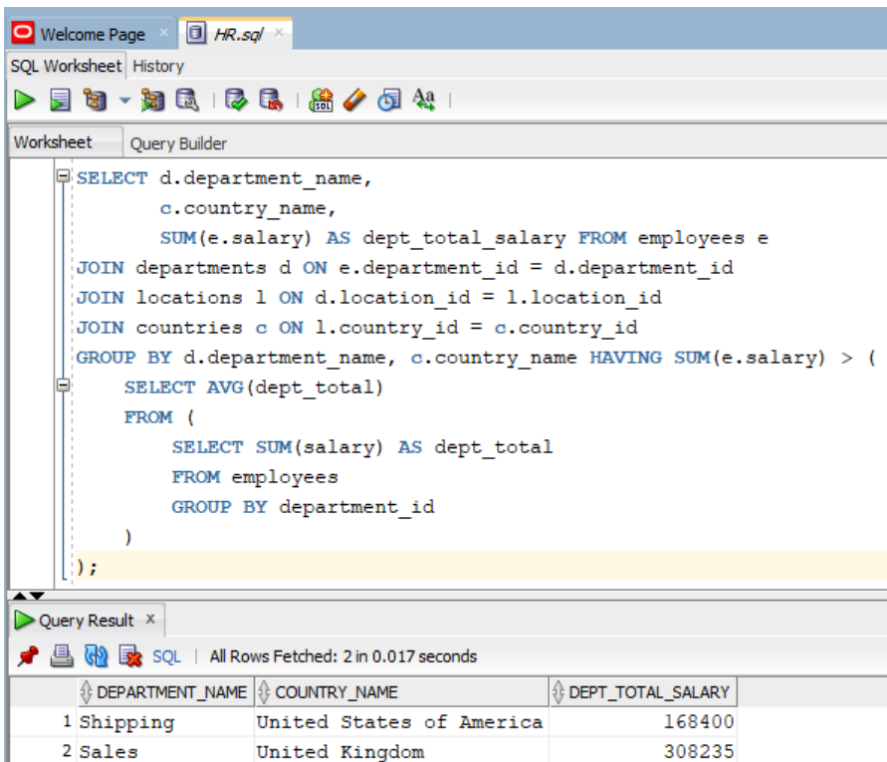
The screenshot shows the SQL Developer interface with a query window titled 'HR.sql'. The query is as follows:

```
SELECT e.first_name,  
       e.last_name,  
       (e.salary + NVL(e.salary * e.commission_pct, 0)) AS total_salary FROM employees e  
JOIN (  
    SELECT department_id,  
           AVG(salary + NVL(salary * commission_pct, 0)) AS dept_avg_salary  
    FROM employees  
    GROUP BY department_id  
) dept_avg  
ON e.department_id = dept_avg.department_id  
WHERE (e.salary + NVL(e.salary * e.commission_pct, 0)) < dept_avg.dept_avg_salary;
```

The query result is displayed below the editor, showing 8 rows:

FIRST_NAME	LAST_NAME	TOTAL_SALARY
1 Neena	Kochhar	17000
2 Lex	De Haan	17000
3 David	Austin	4800
4 Valli	Pataballa	4800
5 Diana	Lorentz	4620
6 John	Chen	8200
7 Ismael	Sciarra	7700
8 Jose Manuel Urman		7800

## Task 9:



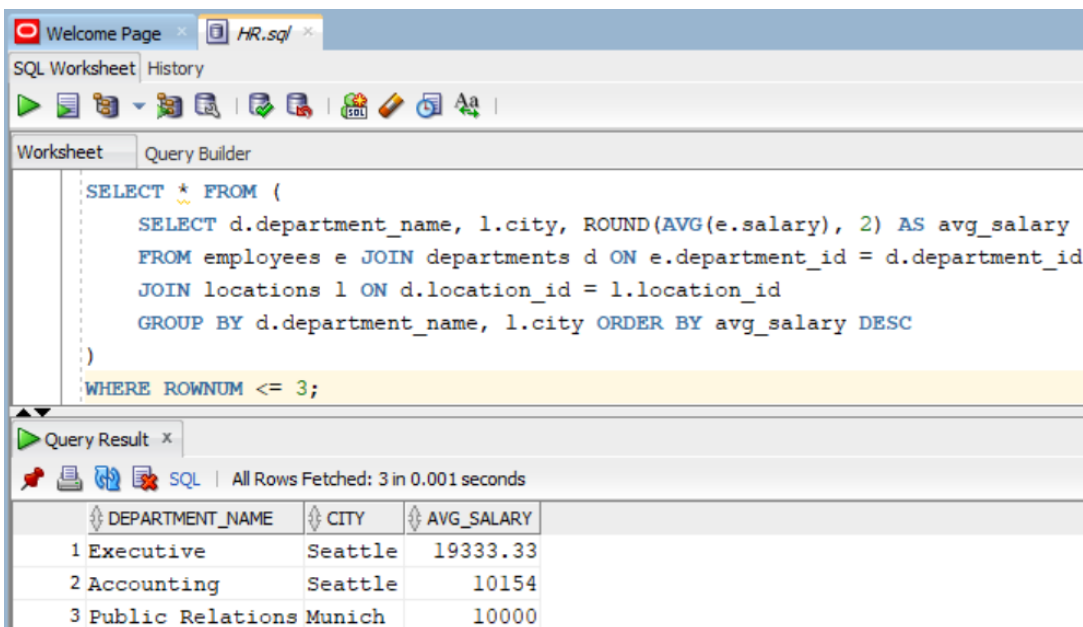
```
SELECT d.department_name,
       c.country_name,
       SUM(e.salary) AS dept_total_salary FROM employees e
JOIN departments d ON e.department_id = d.department_id
JOIN locations l ON d.location_id = l.location_id
JOIN countries c ON l.country_id = c.country_id
GROUP BY d.department_name, c.country_name HAVING SUM(e.salary) > (
  SELECT AVG(dept_total)
  FROM (
    SELECT SUM(salary) AS dept_total
    FROM employees
    GROUP BY department_id
  )
);
```

Query Result

SQL | All Rows Fetched: 2 in 0.017 seconds

DEPARTMENT_NAME	COUNTRY_NAME	DEPT_TOTAL_SALARY
1 Shipping	United States of America	168400
2 Sales	United Kingdom	308235

## Task 10:



```
SELECT * FROM (
  SELECT d.department_name, l.city, ROUND(AVG(e.salary), 2) AS avg_salary
  FROM employees e JOIN departments d ON e.department_id = d.department_id
  JOIN locations l ON d.location_id = l.location_id
  GROUP BY d.department_name, l.city ORDER BY avg_salary DESC
)
WHERE ROWNUM <= 3;
```

Query Result

SQL | All Rows Fetched: 3 in 0.001 seconds

DEPARTMENT_NAME	CITY	AVG_SALARY
1 Executive	Seattle	19333.33
2 Accounting	Seattle	10154
3 Public Relations	Munich	10000