Professor's comment

Problem 1: correct

Problem 1:

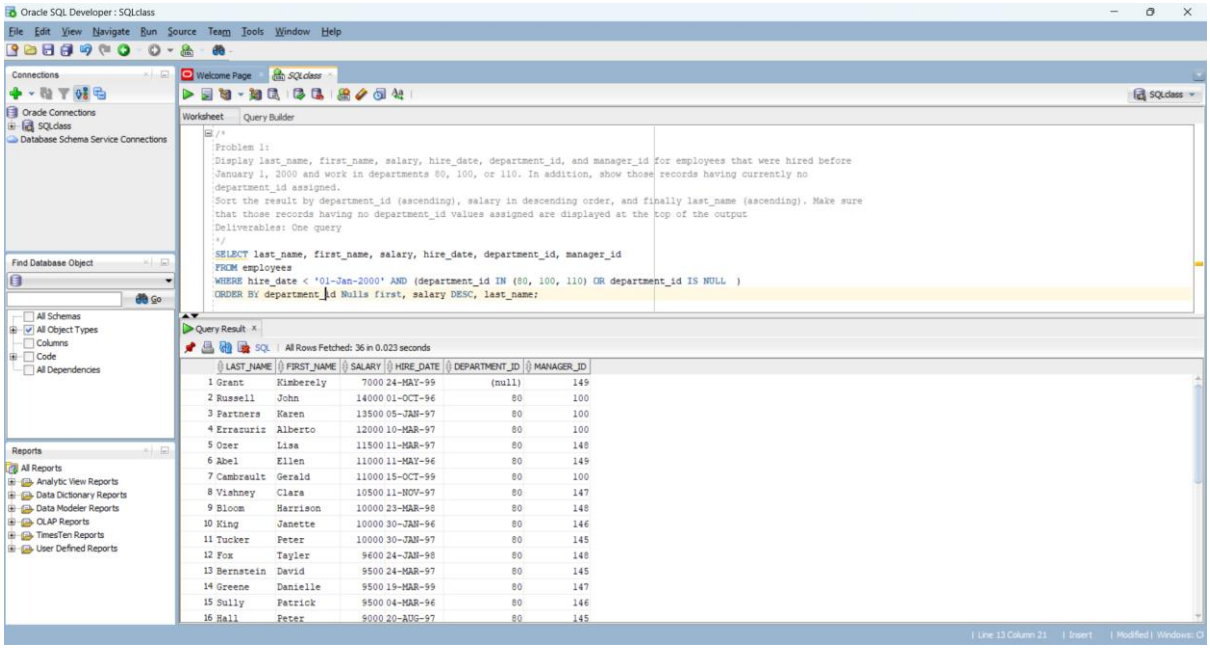
Display last_name, first_name, salary, hire_date, department_id, and manager_id for employees that were hired before January 1, 2000 and work in departments 80, 100, or 110. In addition, show those records having currently no department_id assigned.
Sort the result by department_id (ascending), salary in descending order, and finally last_name (ascending). Make sure that those records having no department_id values assigned are displayed at the top of the output
Deliverables: One query

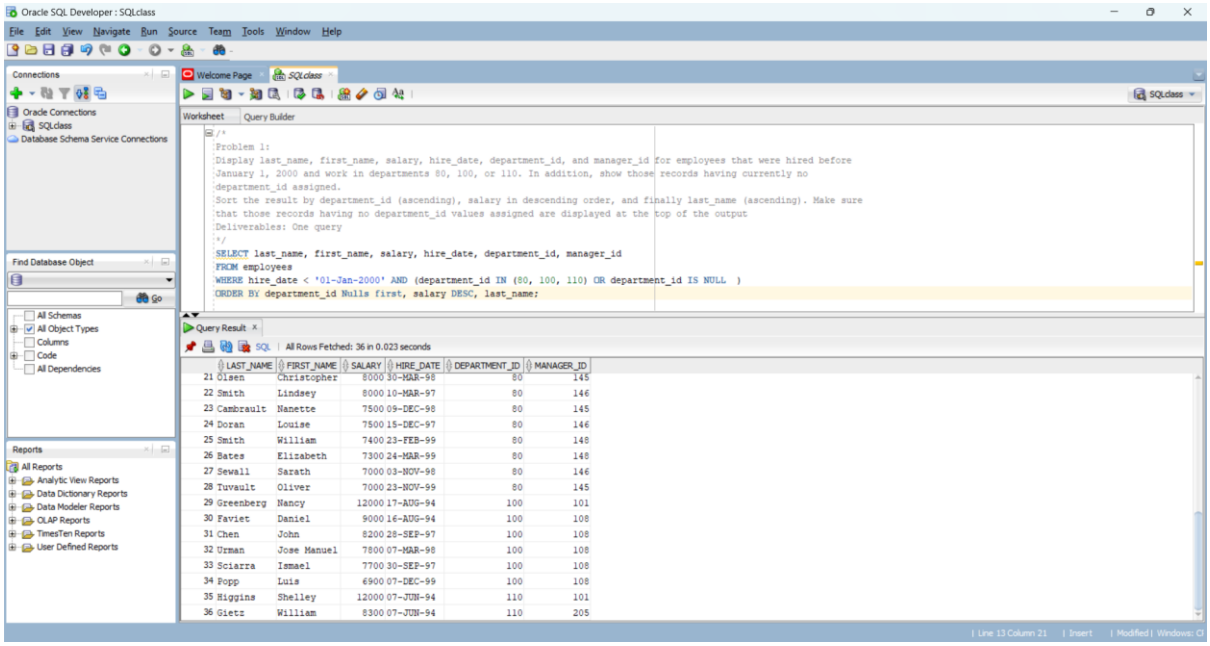
SELECT last_name, first_name, salary, hire_date, department_id, manager_id

FROM employees

WHERE hire_date < '01-Jan-2000' AND (department_id IN (80, 100, 110) OR department_id IS NULL)

ORDER BY department_id Nulls first, salary DESC, last_name;





Problem 2: correct

Problem 2:

Display the following expressions for all employees:

➢ Show the last_name combined with a comma and space and the first_name. Convert the last_name to all uppercase and the first_name to all lowercase. Alias this expression with name

➢ Display the 3rd character in last_name, if this character is a blank space, convert it to a zero (‘0’). Alias this column with 3.ch

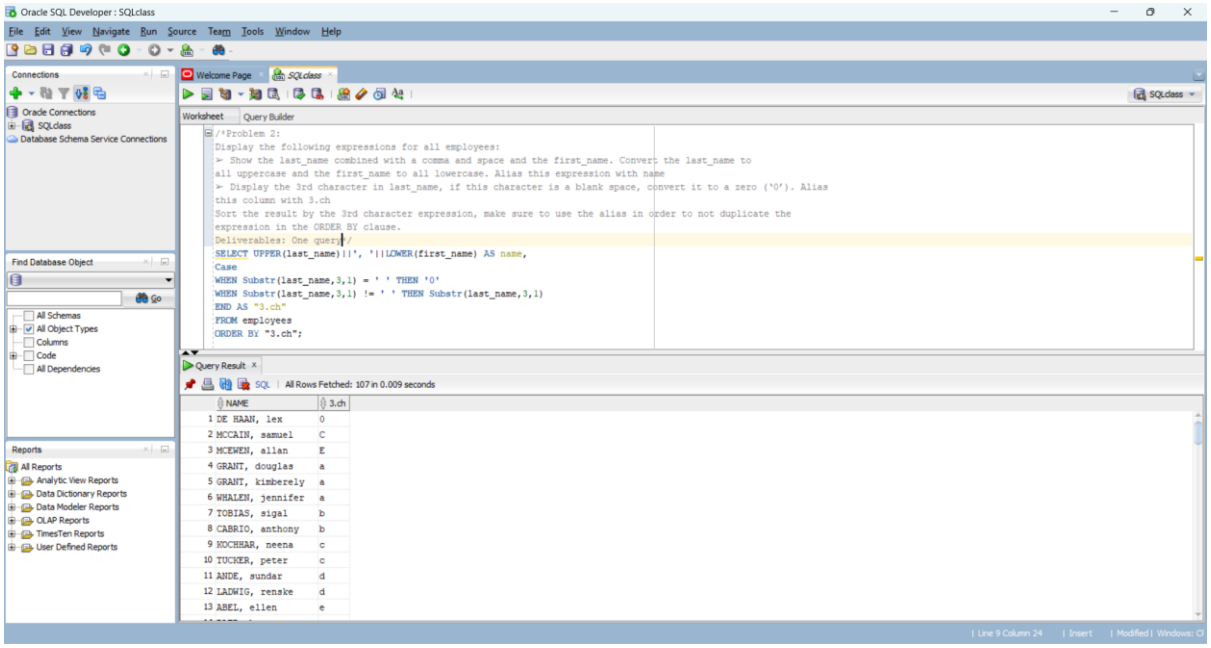
Sort the result by the 3rd character expression, make sure to use the alias in order to not duplicate the expression in the ORDER BY clause.

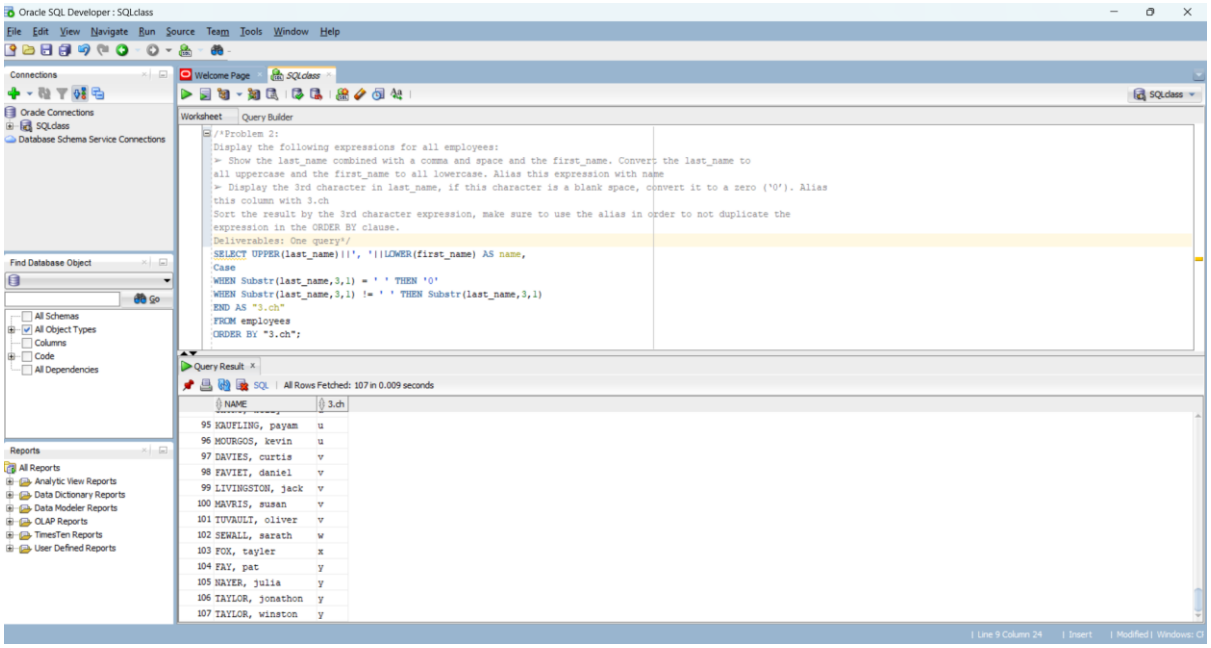
Deliverables: One query

Solution 1: case sensitivty not removed

```
SELECT UPPER(last_name)||', '||LOWER(first_name) AS name,

Case
```

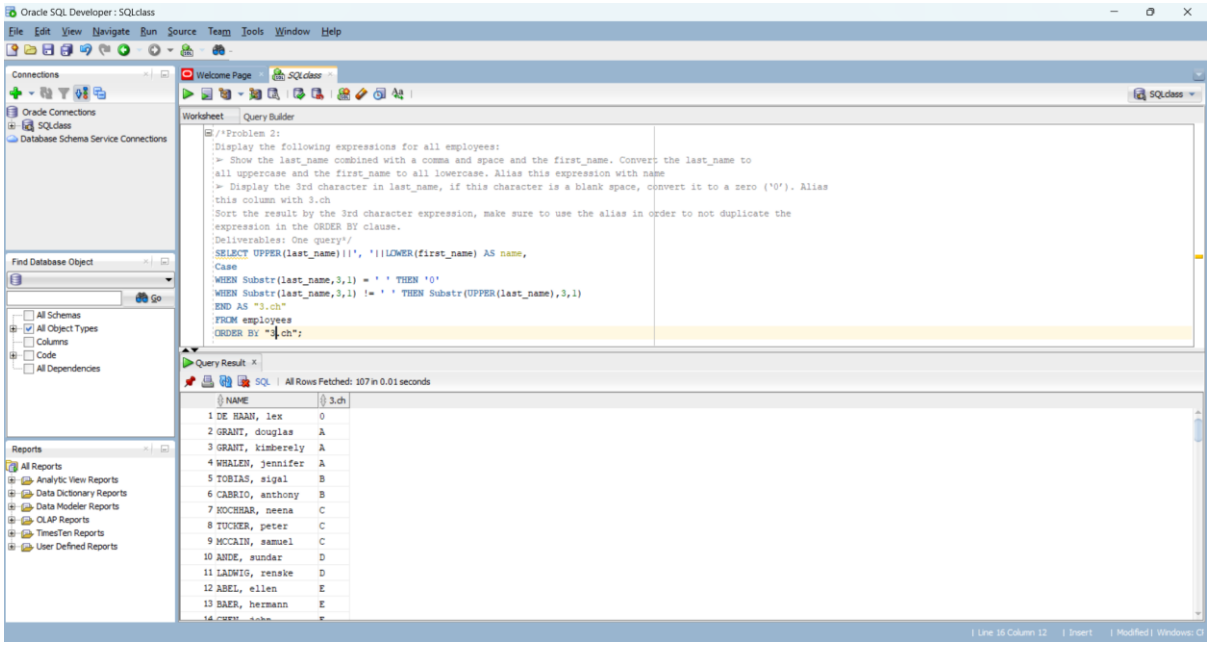
```
WHEN Substr(last_name,3,1) = ' ' THEN '0'

WHEN Substr(last_name,3,1) != ' ' THEN Substr(last_name,3,1)

END AS "3.ch"

FROM employees

ORDER BY "3.ch";
```

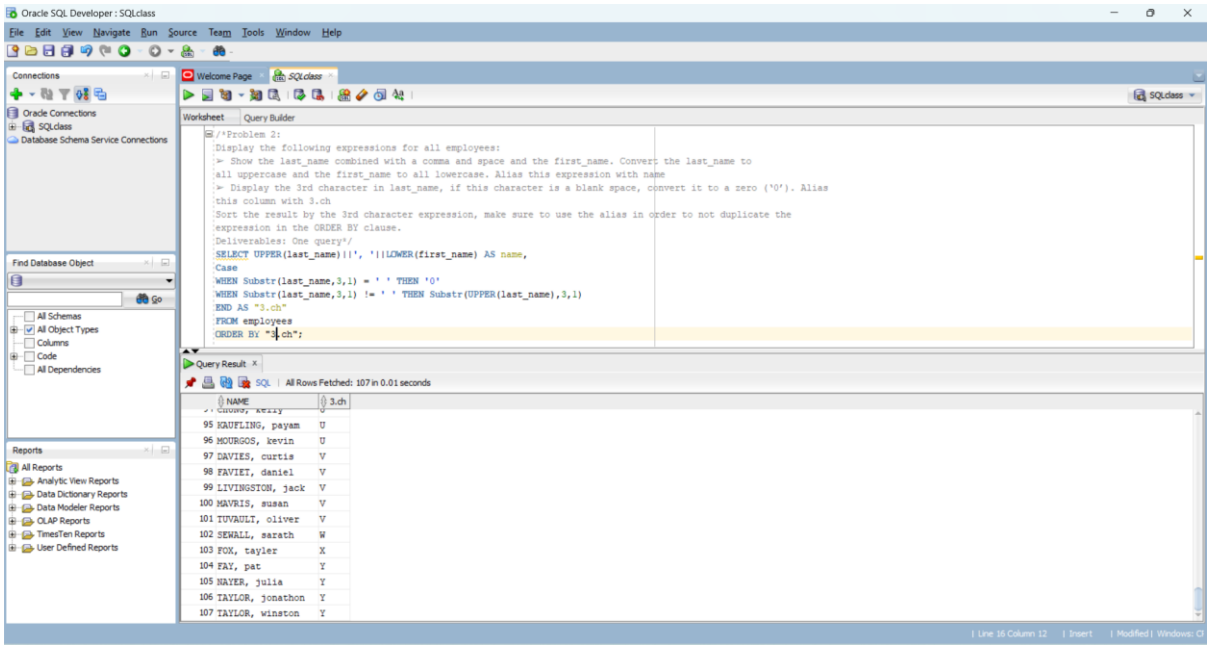




Solution 2: case sensitivty removed /the case ignored

```
SELECT UPPER(last_name)||', '||LOWER(first_name) AS name,
Case
WHEN Substr(last_name,3,1) = ' ' THEN '0'
WHEN Substr(last_name,3,1) != ' ' THEN Substr(UPPER(last_name),3,1)
END AS "3.ch"
FROM employees
ORDER BY "3.ch";
```





Problem 3: correct

Problem 3:

1. Display last_name, first_name, salary for employees under the following conditions: The length of the last_name must equal 5 and either the second character position is the letter r or the fourth character position is the letter e. Sot the result by last_name.

2. Display last_name, first_name, salary for employees under the following conditions: Salary must be greater or equal to 7000 and either the last_name must contain the letter a or s (lower and uppercase letters). Sort the result by last_name.

3. Show the common records (display the same columns) of 1. And 2. Order by last_name

Deliverables: 3 queries

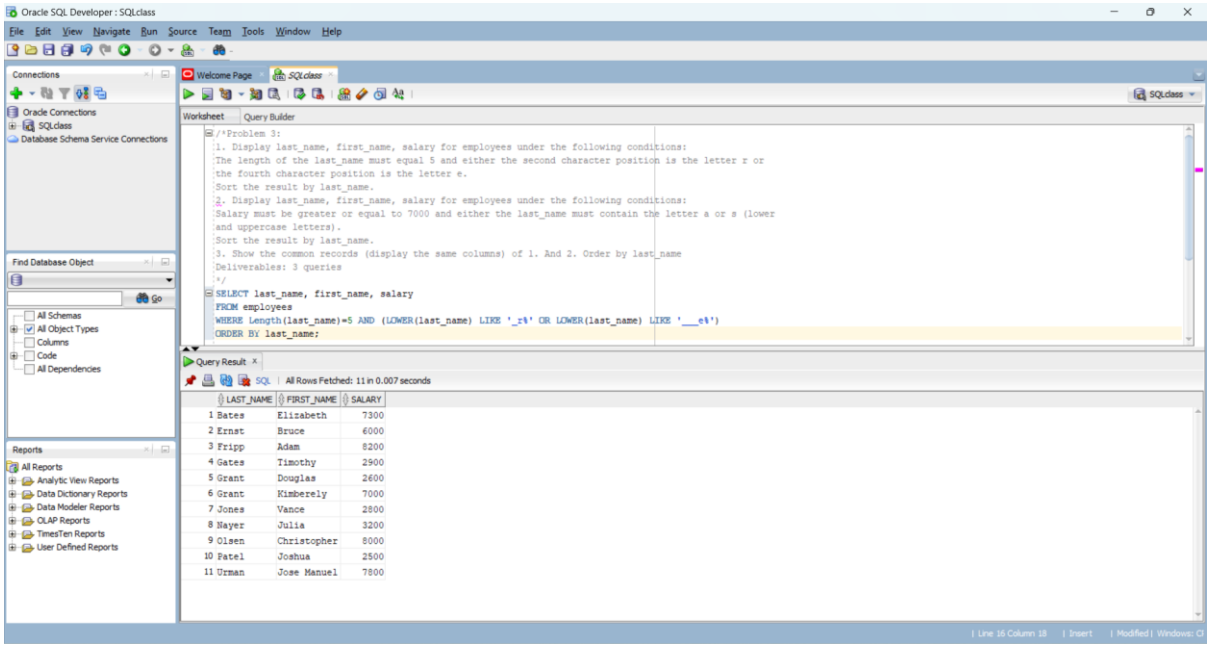
1.

SELECT last_name, first_name, salary

FROM employees

WHERE Length(last_name)=5 AND (LOWER(last_name) LIKE '_r%' OR LOWER(last_name) LIKE '___e%')

ORDER BY last_name;



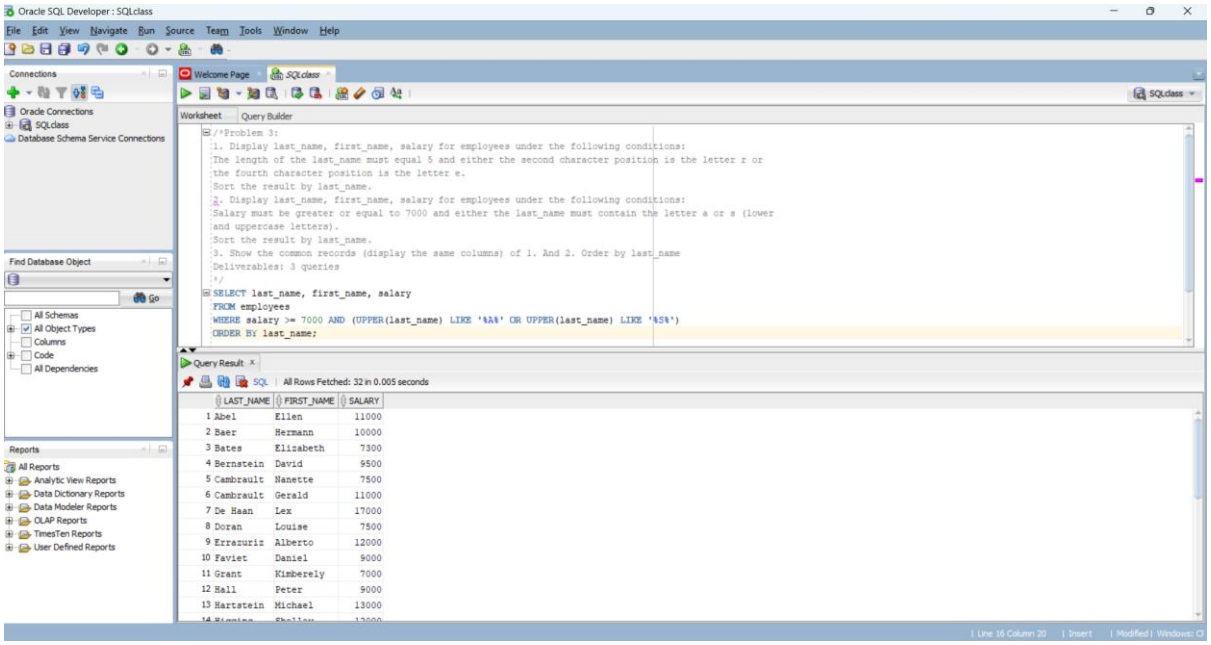
2.

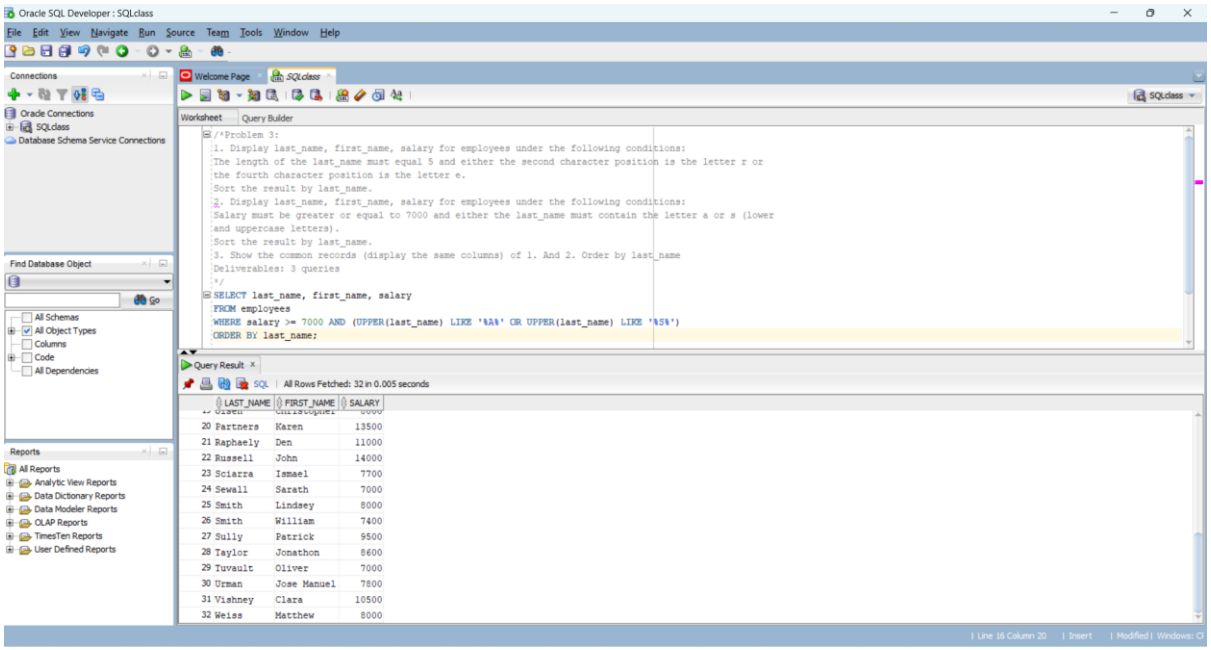
SELECT last_name, first_name, salary

FROM employees

WHERE salary >= 7000 AND (UPPER(last_name) LIKE '%A%' OR UPPER(last_name) LIKE '%S%')

ORDER BY last_name;





3.

SELECT last_name, first_name, salary

FROM employees

WHERE Length(last_name)=5 AND (LOWER(last_name) LIKE '_r%' OR LOWER(last_name) LIKE '___e%')

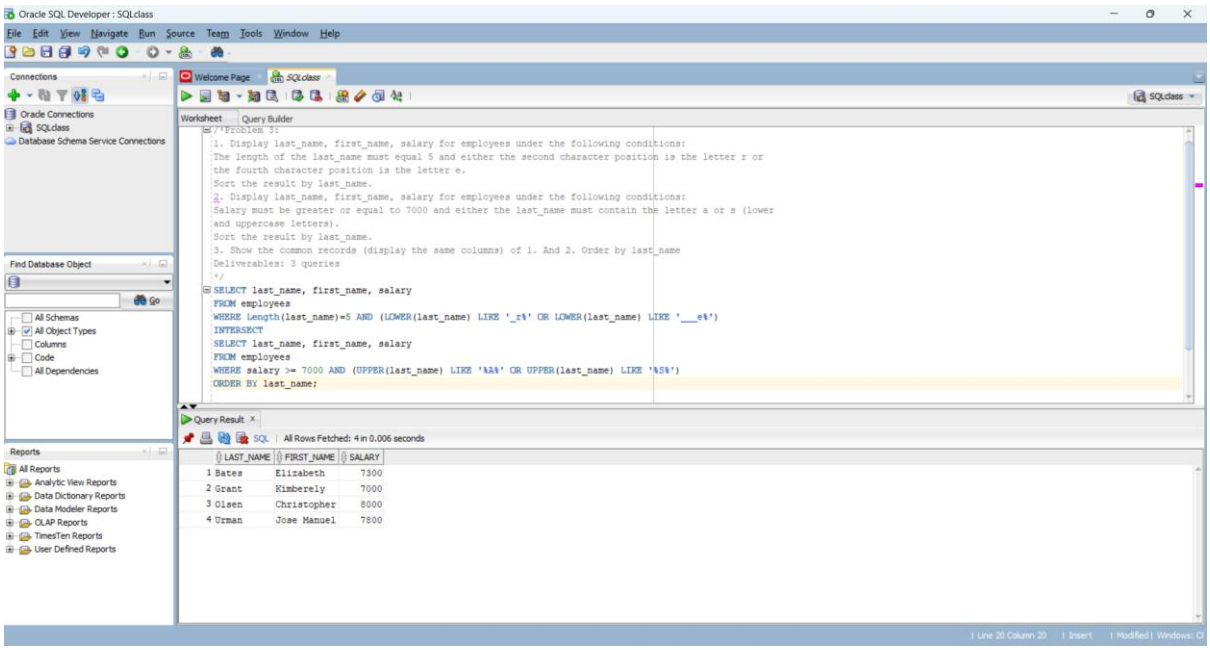
INTERSECT

SELECT last_name, first_name, salary

FROM employees

WHERE salary >= 7000 AND (UPPER(last_name) LIKE '%A%' OR UPPER(last_name) LIKE '%S%')

ORDER BY last_name;



Problem 4: correct

Problem 4:

Show the count of employee records by length of last_name (aliased len_last) and length of the first_name (aliased len_first). Only show those records where the count is greater than 1. Sort the result by length of last_name in descending order and by length of first_name in ascending order.
Deliverables: One query

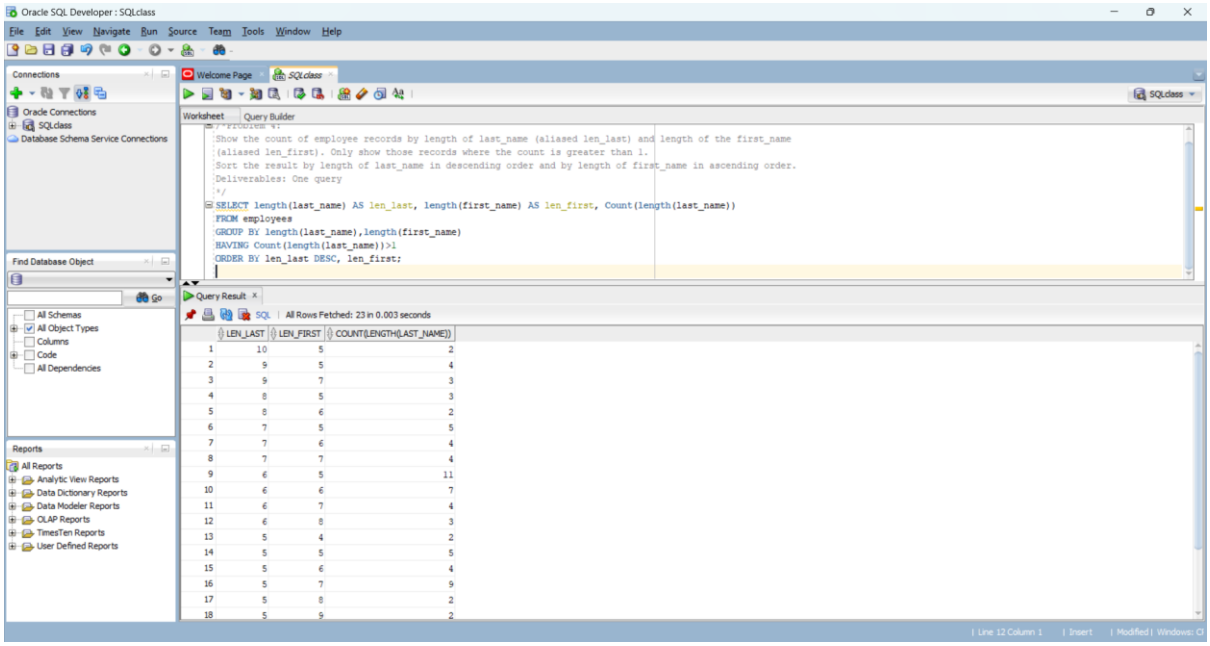
SELECT length(last_name) AS len_last, length(first_name) AS len_first, Count(length(last_name))

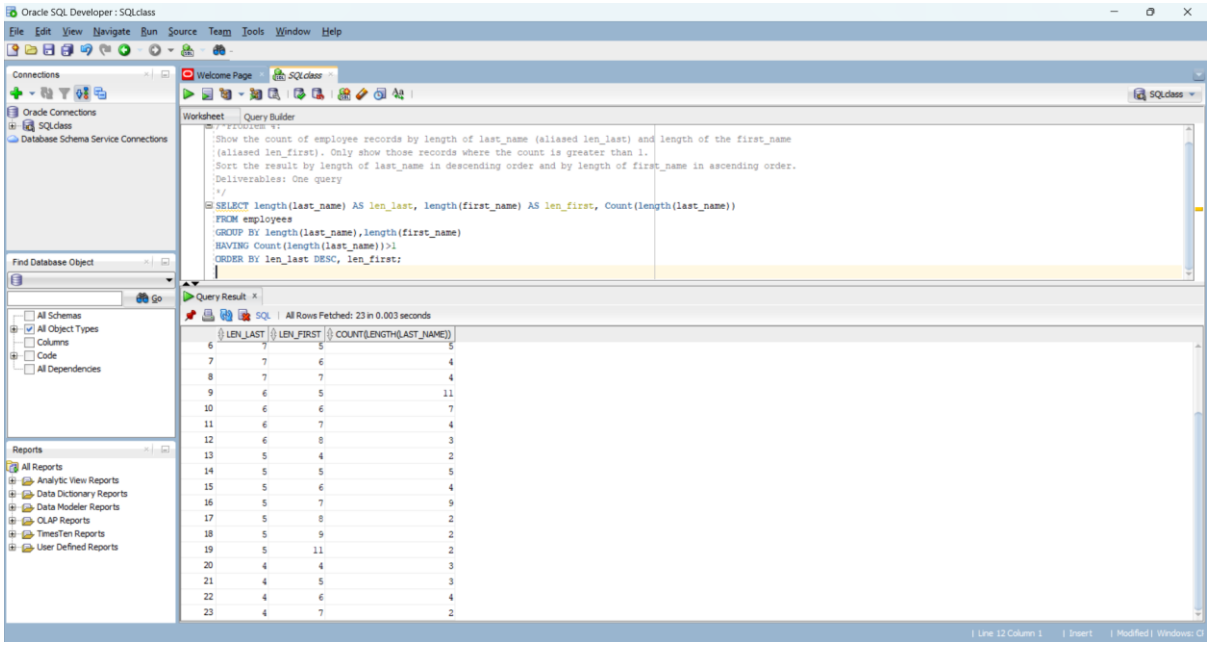
FROM employees

GROUP BY length(last_name),length(first_name)

HAVING Count(length(last_name))>1

ORDER BY len_last DESC, len_first;





Problem 5: In the subquery of the FROM clause, you converted a Null commission_pct into zero. In your main query you test for Null values of comm_pct, but there are no more Null values. (-1)

Also the full_sal_raise should be the current_full_sal plus the salary raise. Just add 1 to the raise values, such as 1.0535 and 1.1725. (-2)

Problem 5:

1. Create a query displaying last_name, department_id, salary, and commission_pct aliased as comm_pct. Show only those records for departments that are located in cities not starting with the letter s. Add the following expression as the last column aliased with current_full_sal:

➢ Add the commission amount (salary multiplied by commission_pct, when commission_pct is null, replace it with a zero) with the salary column together.

2. Use part 1. query as a subquery in a FROM clause. SELECT last_name, salary, comm_pct, department_id, current_full_sal and the following expression aliased as full_sal_raise:

➢ Display the sum of the current_full_sal and the following expression:

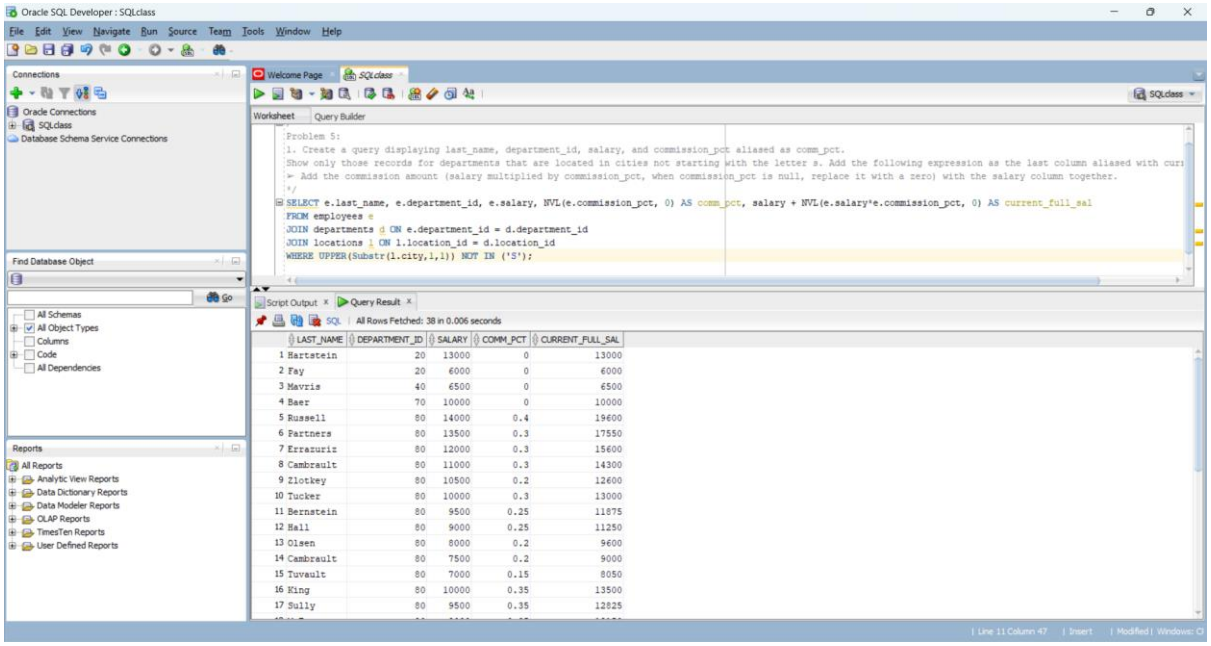
If the comm_pct contains a value then calculate a 5.35% increase of the current_full_sal, otherwise calculate a 17.25% increase of the current_full_sal only. Display this expression as an integer by dropping the decimals.

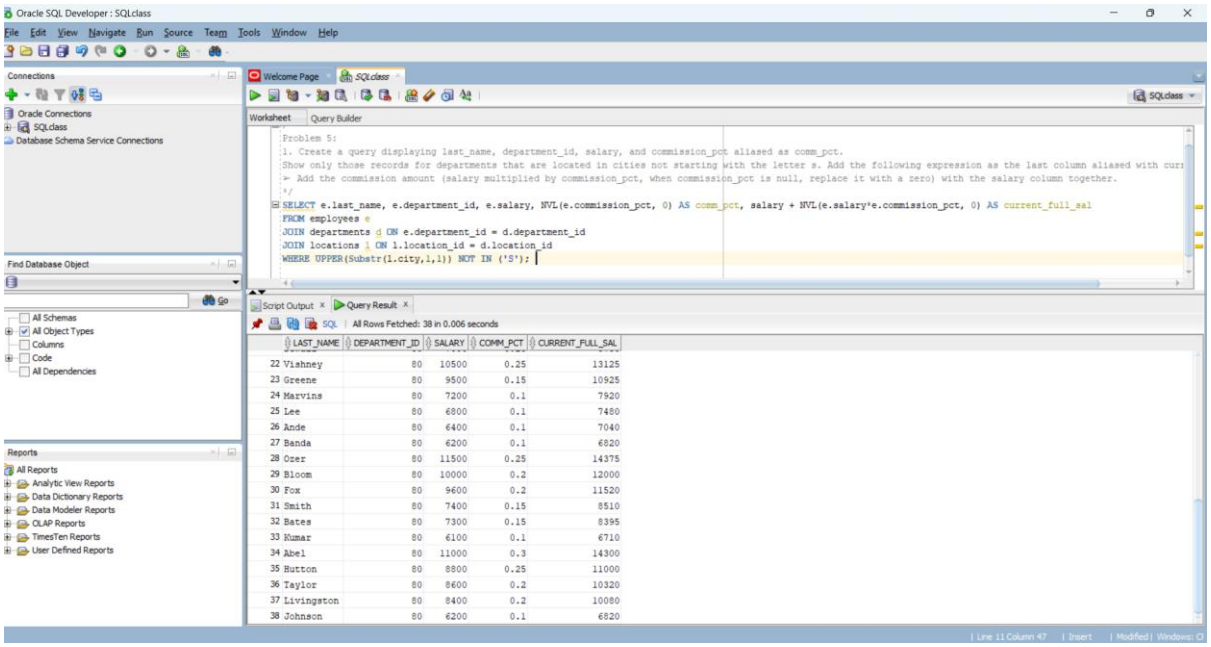
Sort the result by the current_full_sal.

Deliverables: Two queries

1.

```
SELECT e.last_name, e.department_id, e.salary, NVL(e.commission_pct, 0) AS comm_pct,
salary + NVL(e.salary*e.commission_pct, 0) AS current_full_sal

FROM employees e

JOIN departments d ON e.department_id = d.department_id

JOIN locations l ON l.location_id = d.location_id

WHERE UPPER(Substr(l.city,1,1)) NOT IN ('S');
```





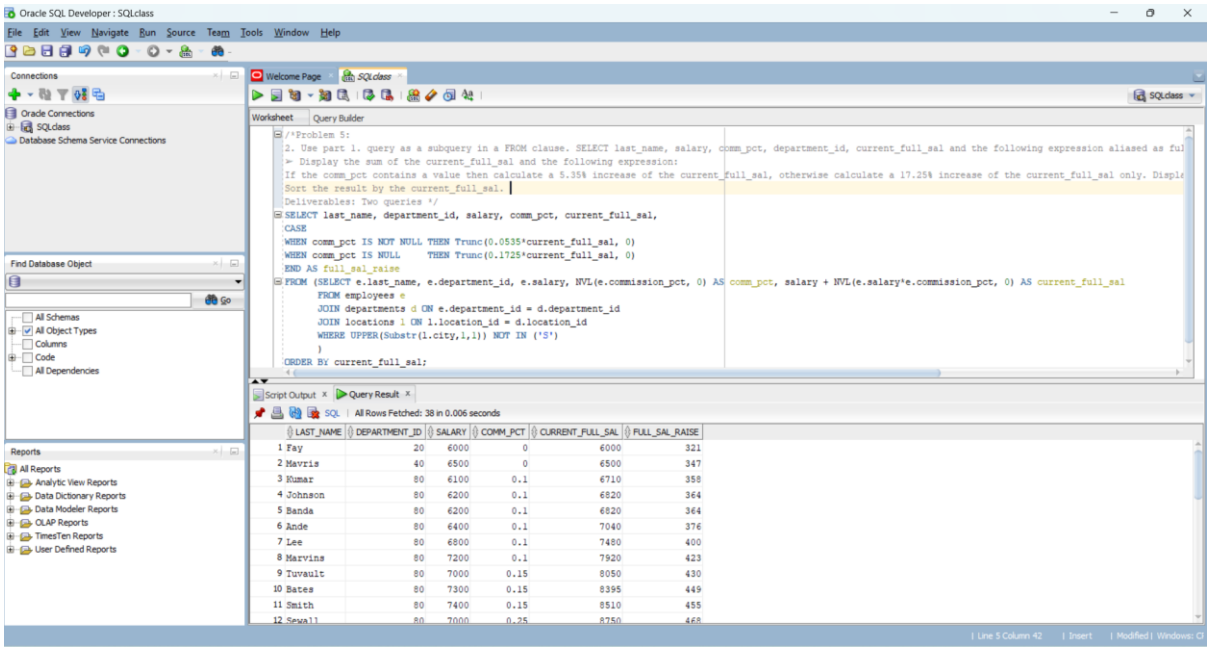
In the subquery of the FROM clause, you converted a Null commission_pct into zero. In your main query you test for Null values of comm_pct, but there are no more Null values. (-1)

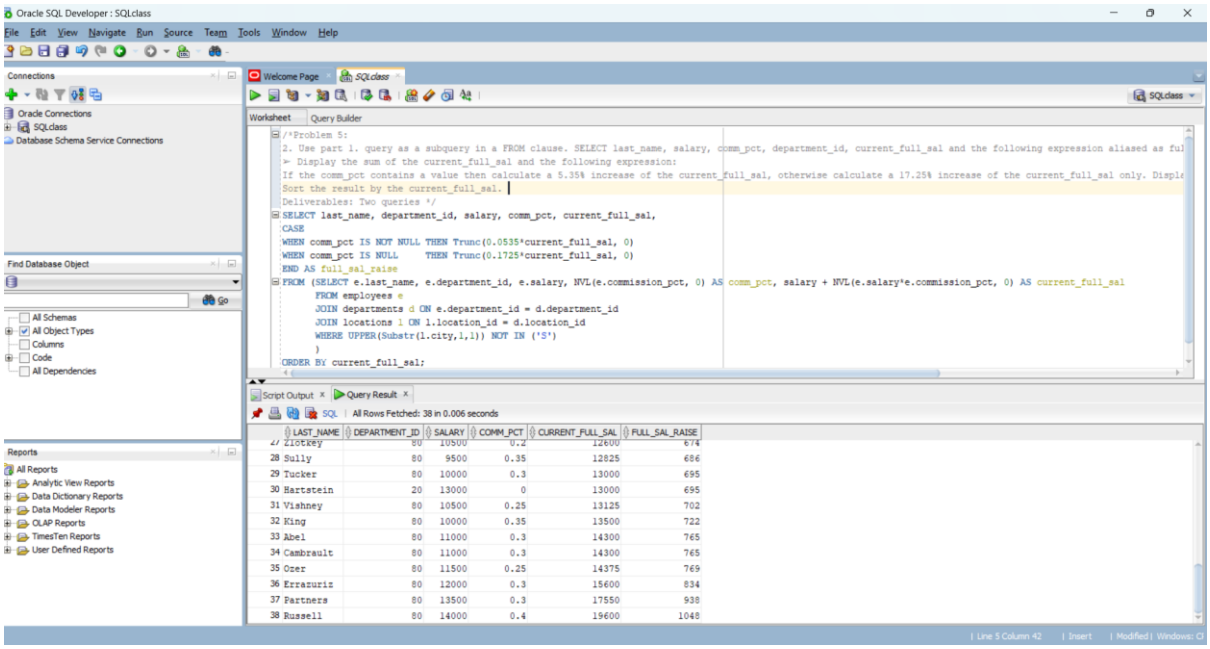
Also the full_sal_raise should be the current_full_sal plus the salary raise. Just add 1 to the raise values, such as 1.0535 and 1.1725. (-2)

2.

SELECT last_name, department_id, salary, comm_pct, current_full_sal,

```
CASE
WHEN comm_pct IS NOT NULL THEN Trunc(0.0535*current_full_sal, 0)
WHEN comm_pct IS NULL      THEN Trunc(0.1725*current_full_sal, 0)
END AS full_sal_raise
FROM (SELECT e.last_name, e.department_id, e.salary, NVL(e.commission_pct, 0) AS comm_pct, salary + NVL(e.salary*e.commission_pct, 0) AS current_full_sal
    FROM employees e
    JOIN departments d ON e.department_id = d.department_id
    JOIN locations l ON l.location_id = d.location_id
    WHERE UPPER(Substr(l.city,1,1)) NOT IN ('S')
    )
ORDER BY current_full_sal;
```





Problem 6: After fixing the errors from problem 5 this query is correct.

Problem 6:

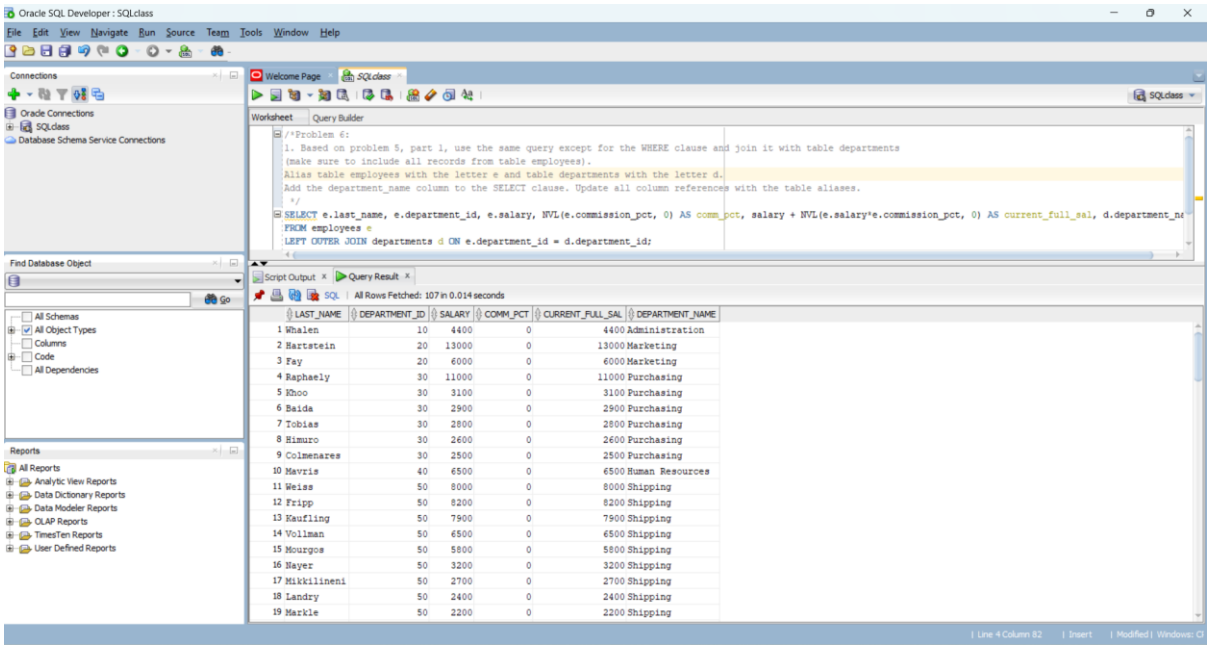
1. Based on problem 5, part 1, use the same query except for the WHERE clause and join it with table departments (make sure to include all records from table employees). Alias table employees with the letter e and table departments with the letter d. Add the department_name column to the SELECT clause. Update all column references with the table aliases.

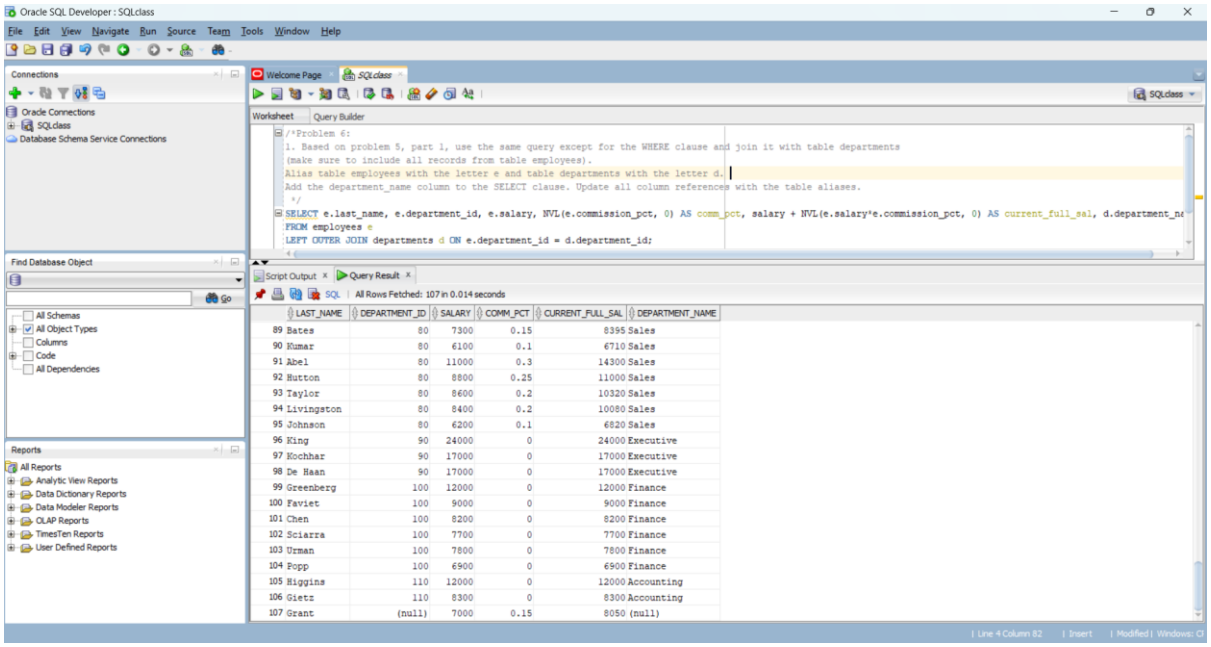
1.

SELECT e.last_name, e.department_id, e.salary, NVL(e.commission_pct, 0) AS comm_pct, salary + NVL(e.salary*e.commission_pct, 0) AS current_full_sal, d.department_name

FROM employees e

LEFT OUTER JOIN departments d ON e.department_id = d.department_id;





2. Use part 1. query as a subquery in a FROM clause. SELECT department_name and the following expressions:

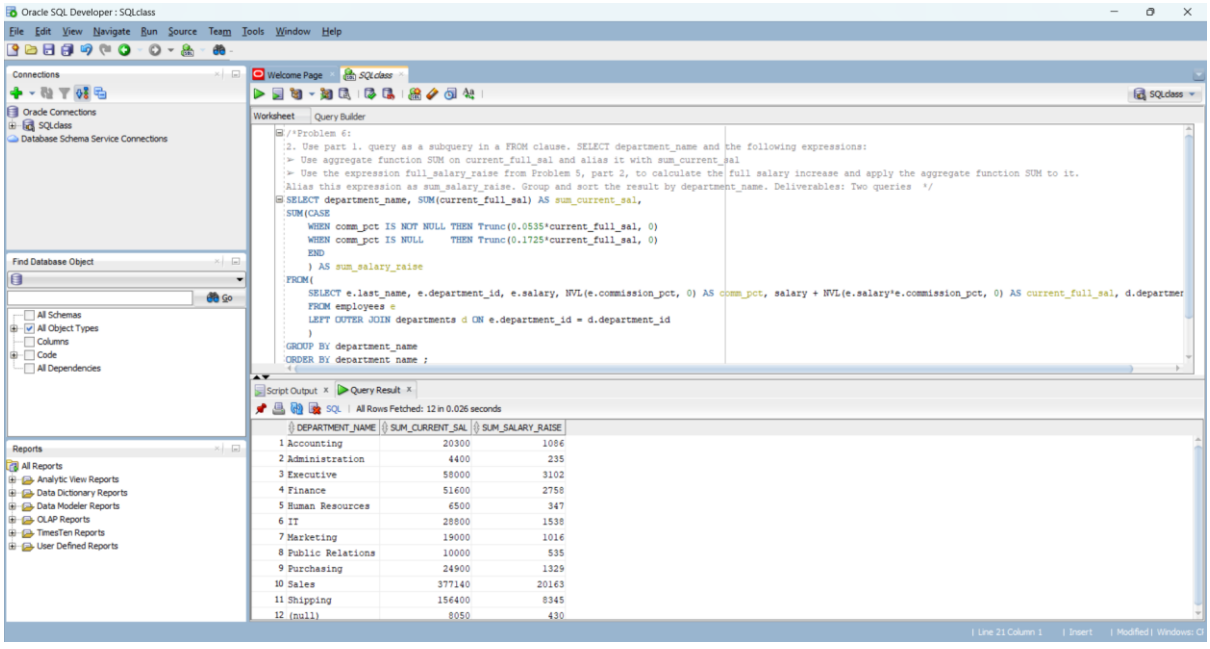
➢ Use aggregate function SUM on current_full_sal and alias it with sum_current_sal

➢ Use the expression full_salary_raise from Problem 5, part 2, to calculate the full salary increase and apply the aggregate function SUM to it. Alias this expression as sum_salary_raise. Group and sort the result by department_name.

Deliverables: Two queries

2.

```
SELECT department_name, SUM(current_full_sal) AS sum_current_sal,
SUM(CASE
    WHEN comm_pct IS NOT NULL THEN Trunc(0.0535*current_full_sal, 0)
    WHEN comm_pct IS NULL     THEN Trunc(0.1725*current_full_sal, 0)
    END
    ) AS sum_salary_raise
FROM(
    SELECT e.last_name, e.department_id, e.salary, NVL(e.commission_pct, 0) AS comm_pct, salary + NVL(e.salary*e.commission_pct, 0) AS current_full_sal, d.department_name
    FROM employees e
    LEFT OUTER JOIN departments d ON e.department_id = d.department_id
    )
GROUP BY department_name
ORDER BY department_name ;
```



Problem 7: You need to use a compound SET clause in order to execute the subquery only once:

```
SET (job_id,salary) = (
                    SELECT job_id, ((min_salary+max_salary)/2)
                    FROM jobs
                    WHERE LOWER(job_title) = 'accounting manager'
                    ), (-1)
```

Problem 7:

Whenever employees change department_id and/or job_id, a copy of the current employee record using selected columns is added to the job_history table. John Chen (employee_id 110) is promoted to accounting manager and moved from the finance to the accounting department.

1. Add a record to table job_history based on employee_id of 110 from table employees. For the start_date column in job_history use the hire_date column from table employees. Use the system date for the end_date column in job_history. Add the remaining columns based on the same column names from table employees. Use a subquery and not the VALUES clause in the INSERT statement.

1.

```
INSERT INTO job_history
SELECT employee_id, hire_date, SYSDATE, job_id, department_id
FROM employees
WHERE employee_id = 110;
```

2. Now modify the employee record having employee_id of 110 as follows:

➢ Update the job_id and salary columns based on data in table jobs. Use a compound SET clause to update both columns using a subquery. Update the job_id to the job_id in table jobs for the job_title of accounting manager and the salary to the average of min_salary and max_salary in table jobs for job_title accounting manager (use subquery).

➢ Update the department_id based on table departments to the new department accounting

You need to use a compound SET clause in order to execute the subquery only once:

```
SET (job_id,salary) = (
                    SELECT job_id, ((min_salary+max_salary)/2)
                    FROM jobs
                    WHERE LOWER(job_title) = 'accounting manager'
                    ), (-1)
```

2.

```
UPDATE employees
SET job_id =
    ( SELECT job_id FROM jobs WHERE LOWER(job_title) = 'accounting manager'),
    salary =
    ( SELECT (min_salary + max_salary)/2 FROM jobs
      WHERE LOWER(job_title) =
```

(SELECT LOWER(job_title) FROM jobs WHERE LOWER(job_title) = 'accounting manager')),

department_id =

(SELECT department_id FROM departments WHERE LOWER(department_name) LIKE '%accounting%')

WHERE employee_id =110;

3. Now issue a two SELECT statements verifying the changes:

➢ Show employee_id, hire_date, salary, job_id, department_id from table employees for employee_id 110.

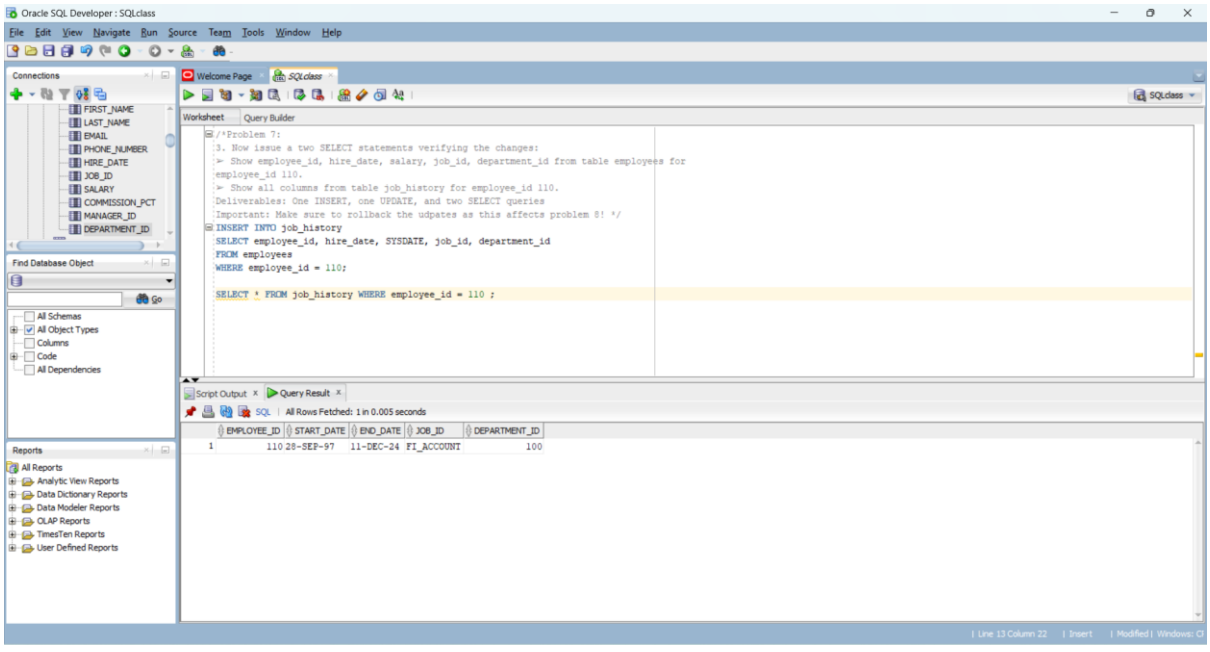
➢ Show all columns from table job_history for employee_id 110.

Deliverables: One INSERT, one UPDATE, and two SELECT queries

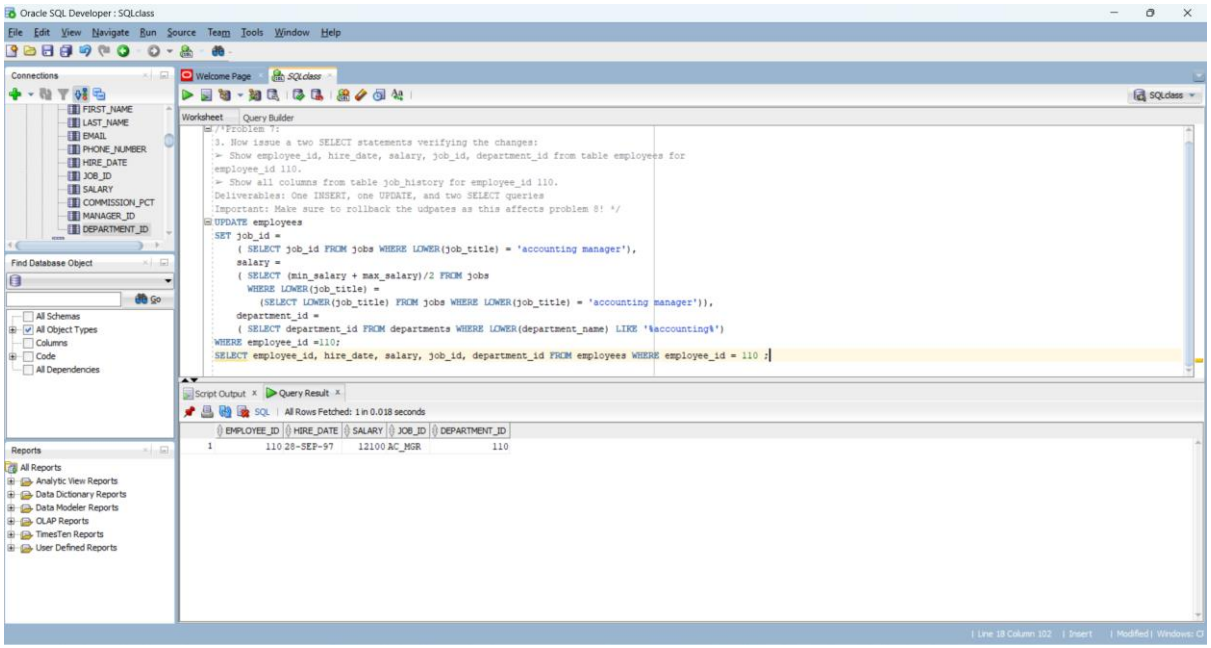
Important: Make sure to rollback the udpates as this affects problem 8!

3.

SELECT * FROM job_history WHERE employee_id = 110 ;



SELECT employee_id, hire_date, salary, job_id, department_id from employees WHERE employee_id = 110;



Problem 8:

1. Create a query showing last_name, salary, and hire_date from table employees. In addition, create three additional calculated columns using the LAG function based on sort order of hire_date, use 0 as the default:

➢ Show the previous salary and alias it with lag_1

➢ Show the next previous salary in relation to lag_1 (using an offset of 2) and alias it with lag_2

➢ Show the next previous salary in relation to lag_2 (using an offset of 3) and alias it with lag_3

1.

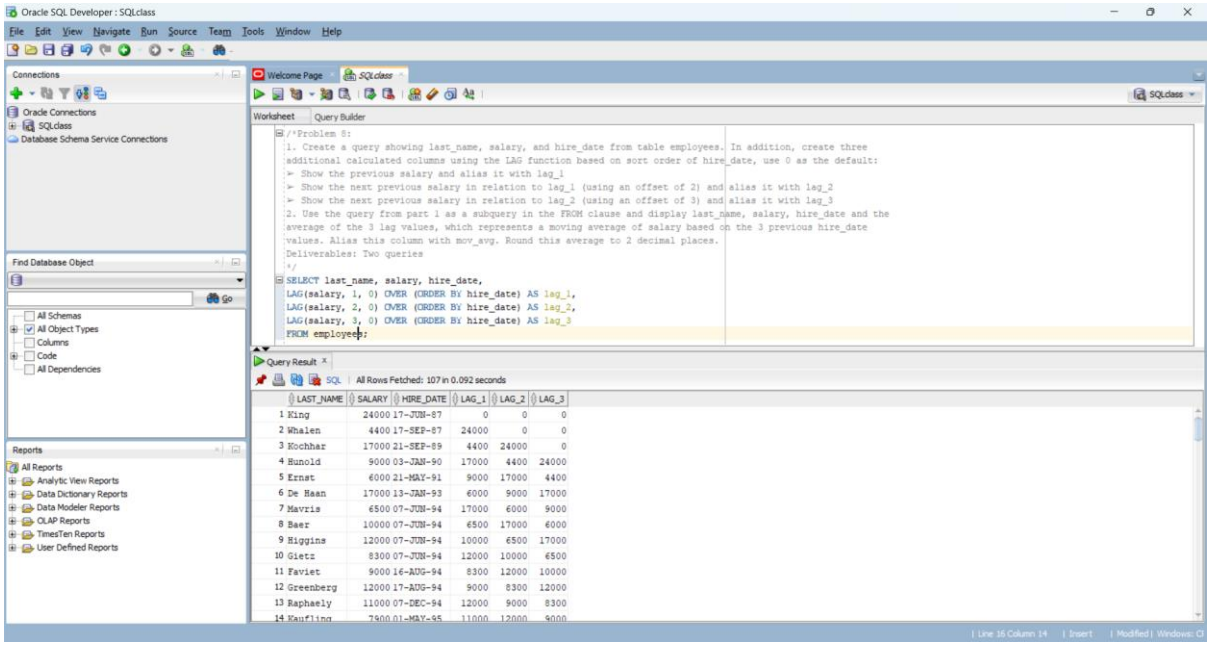
SELECT last_name, salary, hire_date,

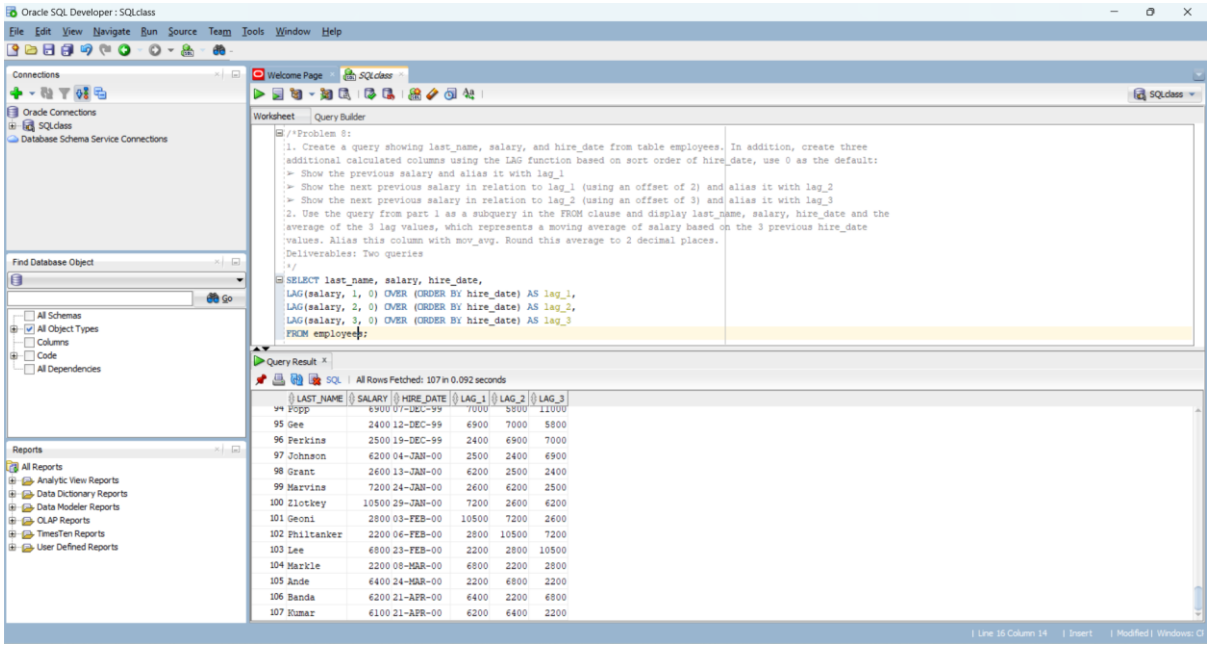
LAG(salary, 1, 0) OVER (ORDER BY hire_date) AS lag_1,

LAG(salary, 2, 0) OVER (ORDER BY hire_date) AS lag_2,

LAG(salary, 3, 0) OVER (ORDER BY hire_date) AS lag_3

FROM employees;



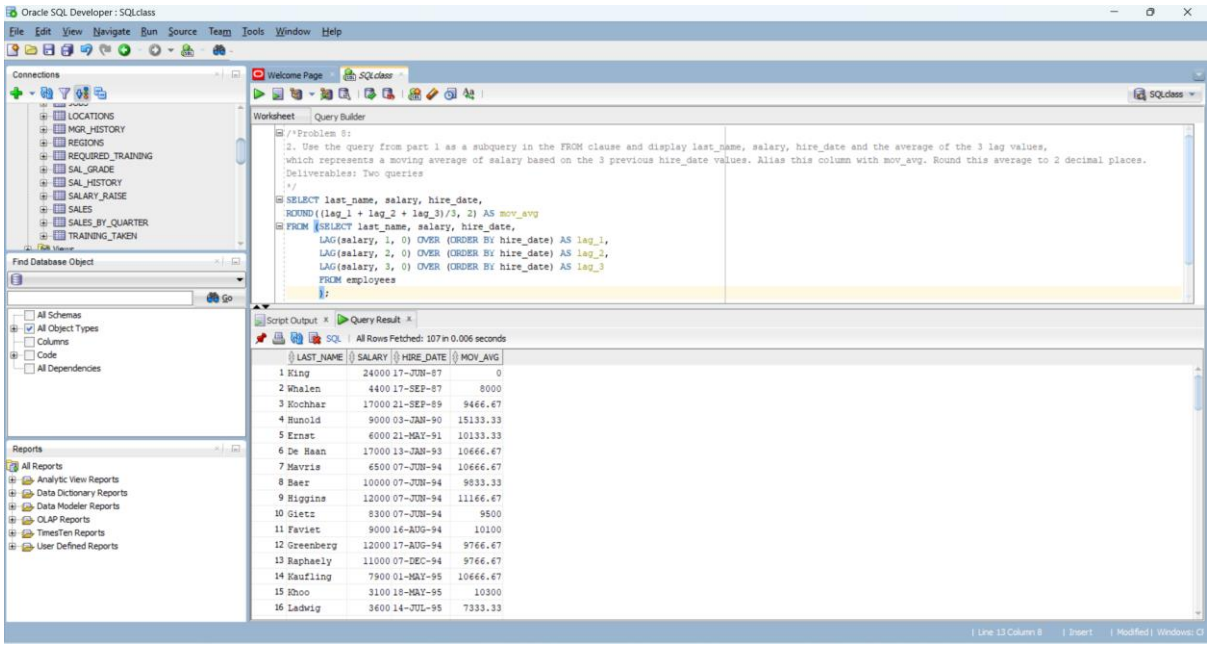


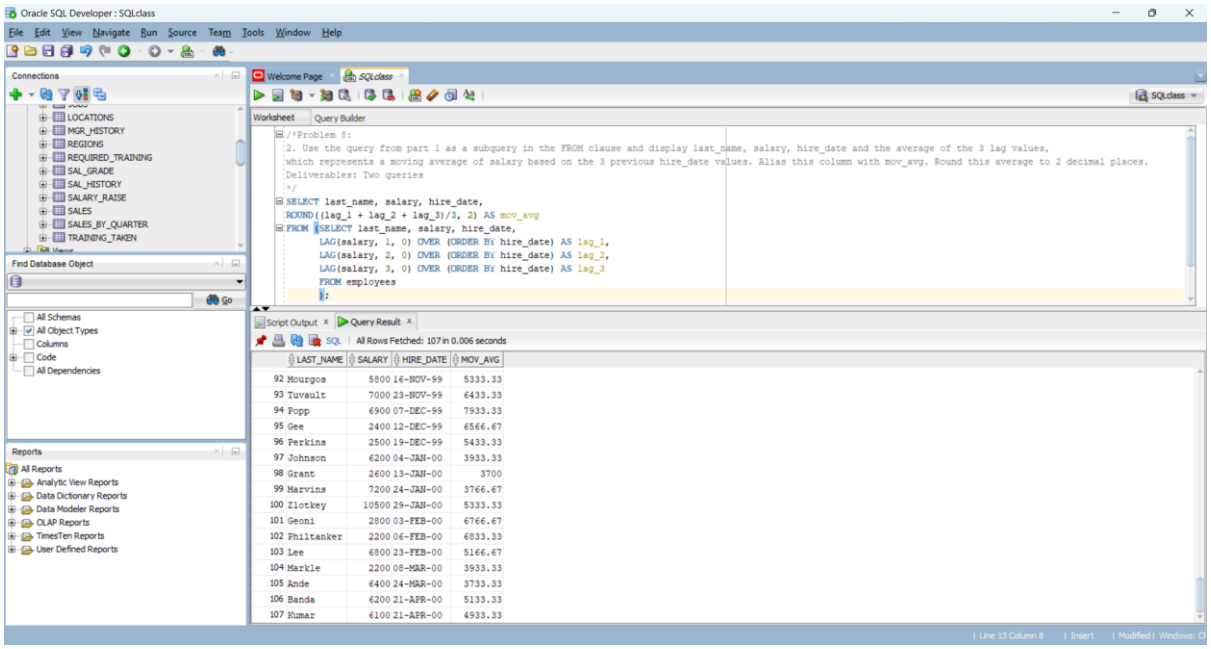
2. Use the query from part 1 as a subquery in the FROM clause and display last_name, salary, hire_date and the average of the 3 lag values, which represents a moving average of salary based on the 3 previous hire_date values. Alias this column with mov_avg. Round this average to 2 decimal places.

Deliverables: Two queries

2.

```
SELECT last_name, salary, hire_date,
ROUND((lag_1 + lag_2 + lag_3)/3, 2) AS mov_avg
FROM (SELECT last_name, salary, hire_date,
        LAG(salary, 1, 0) OVER (ORDER BY hire_date) AS lag_1,
        LAG(salary, 2, 0) OVER (ORDER BY hire_date) AS lag_2,
        LAG(salary, 3, 0) OVER (ORDER BY hire_date) AS lag_3
        FROM employees
        );
```




Oracle SQL Developer : SQLclass
File Edit View Navigate Run Source Team Tools Window Help
Connections
LOCATIONS
MGR_HISTORY
REGIONS
REQUIRED_TRAINING
SAL_GRADE
SAL_HISTORY
SALARY_RAISE
SALES
SALES_BY_QUARTER
TRAINING_TAKEN
Find Database Object
All Schemas
All Object Types
Columns
Code
All Dependencies
Reports
All Reports
Analytic View Reports
Data Dictionary Reports
Data Modeler Reports
OLAP Reports
TimesTen Reports
User Defined Reports
Welcome Page
SQLclass
Worksheet
Query Builder
/*Problem 8:
2. Use the query from part 1 as a subquery in the FROM clause and display last_name, salary, hire_date and the average of the 3 lag values,
which represents a moving average of salary based on the 3 previous hire_date values. Alias this column with mov_avg. Round this average to 2 decimal places.
Deliverables: Two queries
*/
SELECT last_name, salary, hire_date,
ROUND((lag_1 + lag_2 + lag_3)/3, 2) AS mov_avg
FROM (SELECT last_name, salary, hire_date,
LAG(salary, 1, 0) OVER (ORDER BY hire_date) AS lag_1,
LAG(salary, 2, 0) OVER (ORDER BY hire_date) AS lag_2,
LAG(salary, 3, 0) OVER (ORDER BY hire_date) AS lag_3
FROM employees
);
Script Output
Query Result
All Rows Fetched: 107 in 0.006 seconds
LAST_NAME SALARY HIRE_DATE MOV_AVG
92 Mourgos 5800 16-NOV-99 5333.33
93 Tuvault 7000 23-NOV-99 6433.33
94 Popp 6900 07-DEC-99 7933.33
95 Gee 2400 12-DEC-99 6566.67
96 Perkins 2500 19-DEC-99 5433.33
97 Johnson 6200 04-JAN-00 3933.33
98 Grant 2600 13-JAN-00 3700
99 Marvins 7200 24-JAN-00 3766.67
100 Zlotkey 10500 29-JAN-00 5333.33
101 Geoni 2800 03-FEB-00 6766.67
102 Philtanker 2200 06-FEB-00 6833.33
103 Lee 6800 23-FEB-00 5166.67
104 Markle 2200 08-MAR-00 3933.33
105 Ande 6400 24-MAR-00 3733.33
106 Banda 6200 21-APR-00 5133.33
107 Kumar 6100 21-APR-00 4933.33

-THE END-