

## 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;
```

The screenshot shows the Oracle SQL Developer interface. The 'Worksheet' tab contains the problem statement and the SQL query. The 'Query Result' tab displays the output of the query, which lists 16 employees from the 'employees' table. The columns shown are LAST\_NAME, FIRST\_NAME, SALARY, HIRE\_DATE, DEPARTMENT\_ID, and MANAGER\_ID. The results are ordered by department\_id (Nulls first), salary DESC, and last\_name.

LAST_NAME	FIRST_NAME	SALARY	HIRE_DATE	DEPARTMENT_ID	MANAGER_ID
Grant	Kimberely	7000	24-MAY-99	(null)	149
Russell	John	14000	01-OCT-96	80	100
Partners	Karen	13500	05-JAN-97	80	100
Errazuriz	Alberto	12000	10-MAR-97	80	100
Ozer	Lisa	11500	11-MAR-97	80	148
Abel	Eileen	11000	11-MAY-96	80	149
Cambrault	Gerald	11000	15-OCT-99	80	100
Vishney	Clara	10500	11-NOV-97	80	147
Bloom	Harrison	10000	23-MAR-98	80	148
King	Janette	10000	30-JAN-96	80	146
Tucker	Peter	10000	30-JAN-97	80	145
Fox	Taylor	9800	24-JAN-98	80	149
Bernstein	David	9500	24-MAR-97	80	145
Greene	Danielle	9500	19-MAR-99	80	147
Sully	Patrick	9500	04-MAR-96	80	146
Hall	Peter	9000	20-AUG-97	80	145

The screenshot shows the Oracle SQL Developer interface. The 'Worksheet' tab contains a query builder with the following code:

```

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;

```

The 'Query Result' tab displays the output of the query, which is a table of employee data. The columns are LAST\_NAME, FIRST\_NAME, SALARY, HIRE\_DATE, DEPARTMENT\_ID, and MANAGER\_ID. The data is as follows:

LAST_NAME	FIRST_NAME	SALARY	HIRE_DATE	DEPARTMENT_ID	MANAGER_ID
Olsen	Christopher	8000	30-MAR-98	80	145
Smith	Lindsey	8000	10-MAR-97	80	146
Cambrault	Nanette	7500	09-DEC-98	80	145
Doran	Louise	7500	15-DEC-97	80	146
Smith	William	7400	23-FEB-99	80	148
Bates	Elizabeth	7300	24-MAR-99	80	148
Sevall	Sarah	7000	03-NOV-98	80	146
Tuvault	Oliver	7000	23-NOV-99	80	145
Greenberg	Nancy	12000	17-AUG-94	100	101
Faviet	Daniel	9000	16-ADG-94	100	108
Chen	John	8200	28-SEP-97	100	108
Uman	Jose Manuel	7800	07-MAR-98	100	108
Sclarra	Ismael	7700	30-SEP-97	100	108
Popp	Luis	6900	07-DEC-99	100	108
Higgins	Shelley	12000	07-JUN-94	110	101
Gietz	William	8300	07-JUN-94	110	205

## 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 sensitivity not removed

SELECT UPPER(last\_name)||', '|LOWER(first\_name) AS name,

Case

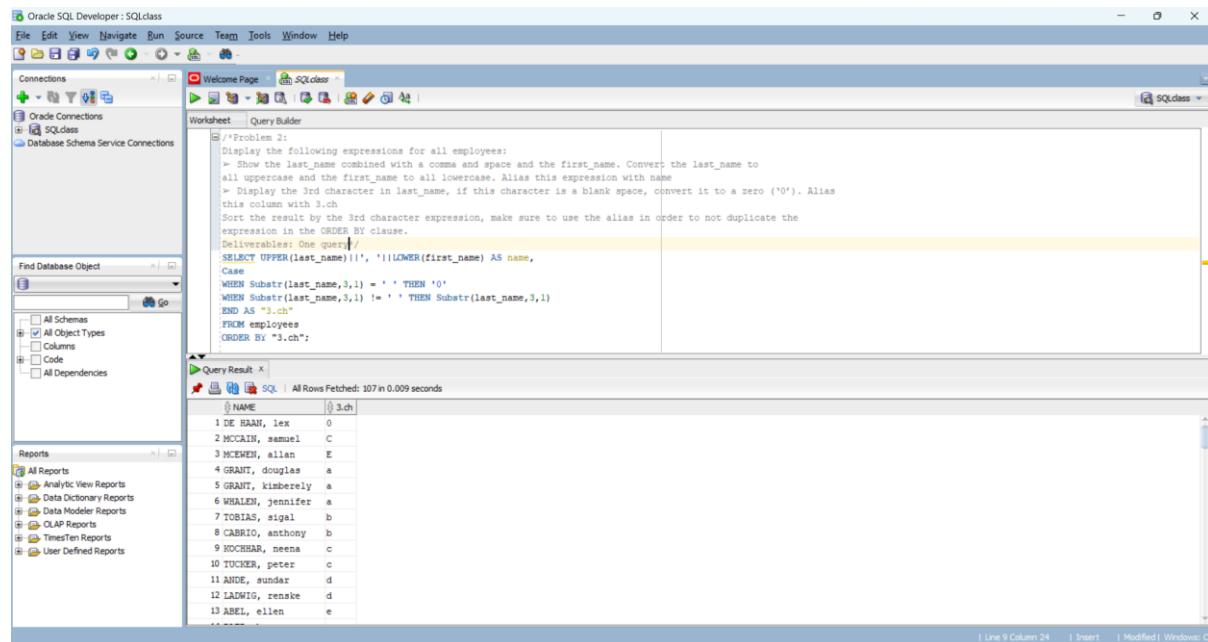
WHEN Substr(last\_name,3,1) = '' THEN '0'

WHEN Substr(last\_name,3,1) != '' THEN Substr(last\_name,3,1)

END AS "3.ch"

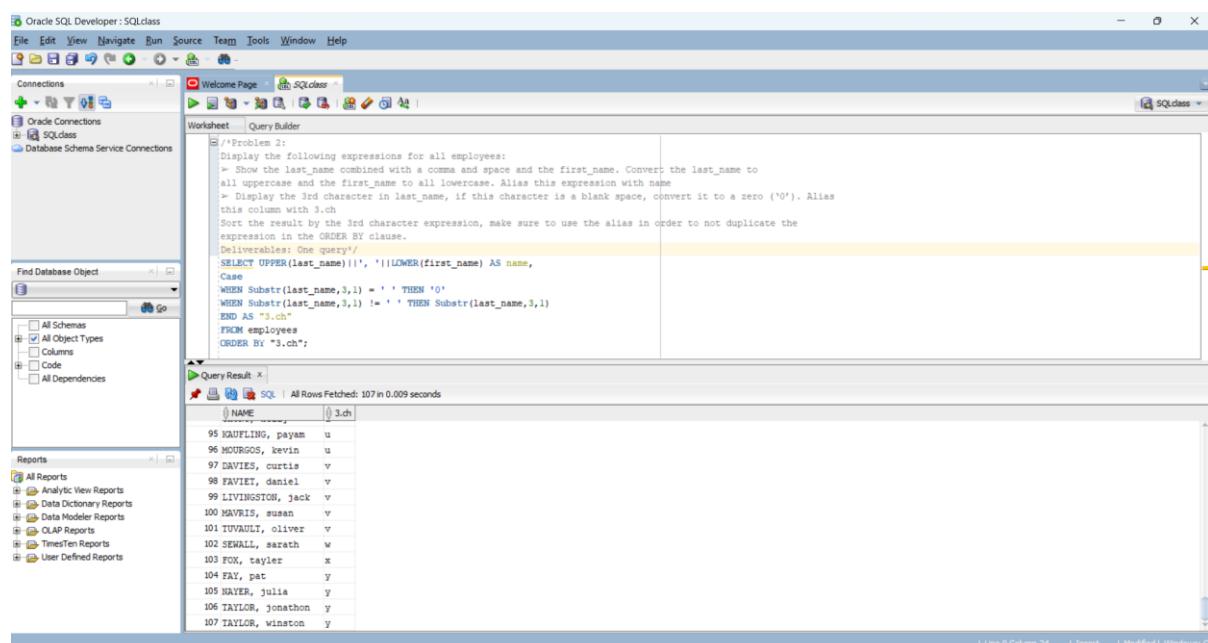
FROM employees

ORDER BY "3.ch";



The screenshot shows the Oracle SQL Developer interface with the SQL Worksheet tab selected. The code area contains the provided PL/SQL block. The Query Result pane displays the output of the query, which is a list of employee names and their corresponding '3.ch' values. The results are as follows:

NAME	3.ch
DE HAAN, lex	0
MCCAIN, samuel	C
MCNEELEN, allan	E
GRANT, douglas	a
GRANT, kimbereley	a
WHALEN, jennifer	a
TORIAS, sigal	b
CABRIO, anthony	b
KOCHEAR, meena	c
TUCKER, peter	c
ANDE, sundar	d
LADwig, reneske	d
ABEL, ellen	e



The screenshot shows the Oracle SQL Developer interface with the SQL Worksheet tab selected. The code area contains the provided PL/SQL block. The Query Result pane displays the output of the query, which is a list of employee names and their corresponding '3.ch' values. The results are as follows:

NAME	3.ch
KAUFLING, payam	u
HOUTGROS, kevin	u
DAVIES, curtis	v
FAVIET, daniel	v
LIVINGSTON, jack	v
MAVRIS, susan	v
TUVAULT, oliver	v
SEWALL, sarath	w
FOX, taylor	x
FAY, pat	y
HAYER, julia	y
TAYLOR, jonathon	y
TAYLOR, winston	y

## Solution 2: case sensitivity removed /the case ignored

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

Case

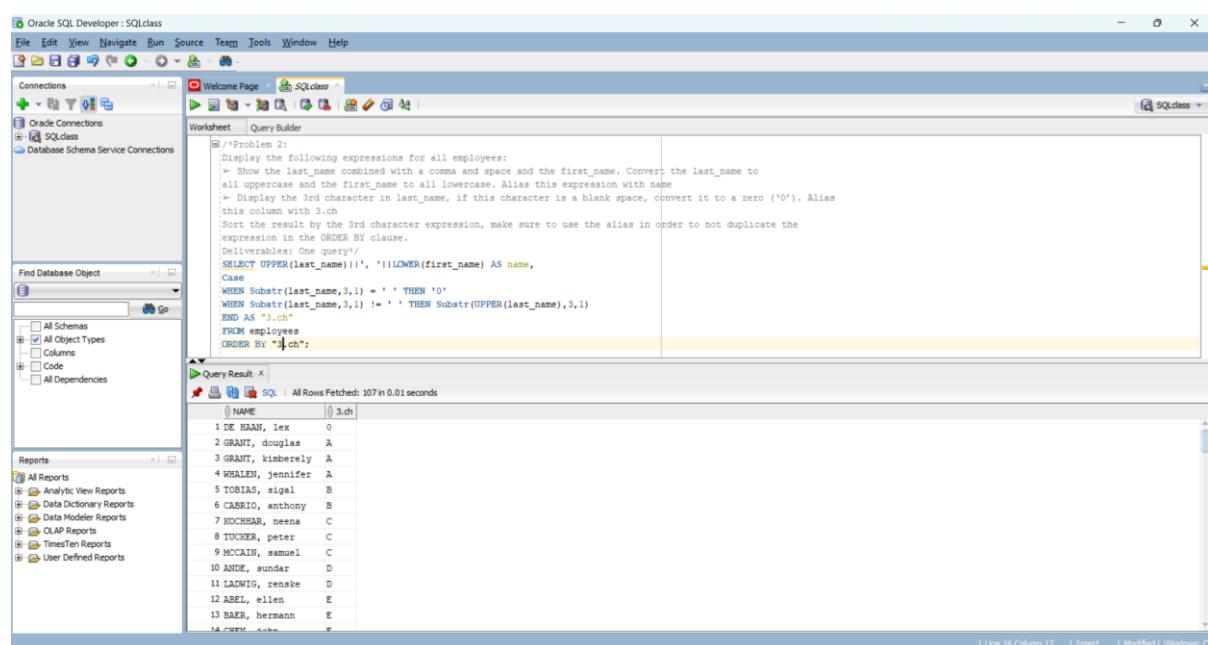
```
WHEN Substr(last_name,3,1) = '' THEN '0'
```

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

```
END AS "3.ch"
```

```
FROM employees
```

```
ORDER BY "3.ch";
```



The screenshot shows the Oracle SQL Developer interface. The top menu bar includes File, Edit, View, Navigate, Run, Source, Team, Tools, Window, and Help. The left sidebar contains sections for Connections, Find Database Object, and Reports. The main area has tabs for Worksheet and Query Builder. The Worksheet tab displays the following SQL code:

```
/*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/  
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";
```

The Query Result tab shows the output of the query, which is a table with two columns: NAME and 3.ch. The data is as follows:

NAME	3.ch
DE HAAN, lex	0
GRANT, douglas	A
GRANT, Kimberly	A
WHALEN, jennifer	A
TOBIAS, sigal	B
CARRIO, anthony	B
KOCHEMAR, neena	C
TUCKER, peter	C
MCCAIN, samuel	C
ANDERSON, sundar	D
LADWIG, renske	D
ABEL, ellen	E
BAER, hermann	E
CURRY, sacha	F

The screenshot shows the Oracle SQL Developer interface. The 'Worksheet' tab is active, displaying a query for 'Problem 3'. The query is as follows:

```

/*Problem 3:
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*/
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";

```

The 'Query Result' tab shows the output of the query, which is a list of employee names and their 3rd character status. The data is as follows:

NAME	3.ch
1. clinton, kerrey	
95 KAUFLING, payam	U
96 MOURGOS, kevin	U
97 DAVIES, curtis	V
98 FAVIET, daniel	V
99 LIVINGSTON, jack	V
100 MAVRIS, susan	V
101 TUVAULT, oliver	V
102 SENNELL, sarath	W
103 FOX, taylor	X
104 FAY, pat	Y
105 NAYER, julia	Y
106 TAYLOR, jonathon	Y
107 TAYLOR, winston	Y

### 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. Sort 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;`

The screenshot shows the Oracle SQL Developer interface. The Worksheet tab contains a query with three numbered steps. Step 1 displays employees with a length of 5 and either second or fourth character as 'r'. Step 2 displays employees with a salary of 7000 and either 'a' or 's' in the last name. Step 3 shows common records ordered by last name. The Query Result tab shows the output of the third step, listing 11 employees with their last names, first names, and salaries.

```
/*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.
   Sort 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

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;
```

LAST_NAME	FIRST_NAME	SALARY
Bates	Elizabeth	7300
Ernst	Bruce	6000
Fripp	Adam	8200
Gates	Timothy	2900
Grant	Douglas	2800
Grant	Kimberly	7000
Jones	Vance	2800
Mayer	Julia	3200
Olsen	Christopher	8000
Patel	Joshua	2500
Urman	Jose Manuel	7800

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;`

Oracle SQL Developer : SQLclass

File Edit View Navigate Run Source Team Tools Window Help

Connections Oracle Connections Database Schema Service Connections

Find Database Object All Schemas All Object Types All Columns Code All Dependencies Reports All Reports Analytic View Reports Data Dictionary Reports Data Modeler Reports OLAP Reports TimesTen Reports User Defined Reports

Worksheet Query Builder

```
/*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.
Sort 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
*/
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;
```

Query Result SQL All Rows Fetched: 32 in 0.005 seconds

LAST_NAME	FIRST_NAME	SALARY
Abel	Ellen	11000
Baer	Hermann	10000
Bates	Elizabeth	7300
Bernstein	David	9500
Cambrault	Nanette	7500
Cambrault	Gerald	11000
De Haan	Lex	17000
Doran	Louise	7500
Errazuriz	Alberto	12000
Faviet	Daniel	9000
Grant	Kimberly	7000
Hall	Peter	9000
Hartstein	Michael	13000
Richter	Shelley	15000

| Line 16 Column 20 | Insert | Modified | Windows: O

Oracle SQL Developer : SQLclass

File Edit View Navigate Run Source Team Tools Window Help

Connections Oracle Connections Database Schema Service Connections

Find Database Object All Schemas All Object Types All Columns Code All Dependencies Reports All Reports Analytic View Reports Data Dictionary Reports Data Modeler Reports OLAP Reports TimesTen Reports User Defined Reports

Worksheet Query Builder

```
/*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.
Sort 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
*/
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;
```

Query Result SQL All Rows Fetched: 32 in 0.005 seconds

LAST_NAME	FIRST_NAME	SALARY
Christensen	Christene	6000
Fartherns	Karen	13500
Raphaely	Den	11000
Russell	John	14000
Sclarra	Ismael	7700
Sewall	Sarah	7000
Smith	Lindsey	8000
Smith	William	7400
Sully	Patrick	9500
Taylor	Jonathon	8600
Tuvault	Oliver	7000
Urman	Jose Manuel	7800
Vishney	Clara	10500
Weiss	Matthew	8000

| Line 16 Column 20 | Insert | Modified | Windows: O

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;
```

The screenshot shows the Oracle SQL Developer interface. The 'Worksheet' tab contains the following query:

```
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.  
Sort 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
```

The 'Query Result' tab shows the output of the query:

LAST_NAME	FIRST_NAME	SALARY
Bates	Elizabeth	7300
Grant	Kimberly	7000
Olsen	Christopher	8000
Urman	Jose Manuel	7800

## 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;

The screenshot shows the Oracle SQL Developer interface with the following details:

- Connections:** Oracle Connections, SQLclass
- Worksheet:** Query Builder
- Query:**

```
--> PROCEDURE v;
--> 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;
```
- Query Result:** All Rows Fetched: 23 in 0.003 seconds

LEN_LAST	LEN_FIRST	COUNT(LENGTH(LAST_NAME))
1	10	5
2	9	5
3	9	7
4	8	5
5	8	6
6	7	5
7	7	6
8	7	7
9	6	5
10	6	6
11	6	7
12	6	8
13	5	4
14	5	5
15	5	6
16	5	7
17	5	8
18	5	9

The screenshot shows the Oracle SQL Developer interface with the following details:

- Connections:** Oracle Connections, SQLclass
- Worksheet:** Query Builder
- Query:**

```
--> PROCEDURE v;
--> 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;
```
- Query Result:** All Rows Fetched: 23 in 0.003 seconds

LEN_LAST	LEN_FIRST	COUNT(LENGTH(LAST_NAME))
6	7	5
7	7	6
8	7	7
9	6	5
10	6	6
11	6	7
12	6	8
13	5	4
14	5	5
15	5	6
16	5	7
17	5	8
18	5	9
19	5	11
20	4	4
21	4	5
22	4	6
23	4	7

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');

Oracle SQL Developer : SQLclass

File Edit View Navigate Run Source Team Tools Window Help

Connections Oracle Connections SQLclass Database Schema Service Connections

Worksheet Query Builder

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.
*/
```

```
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');
```

Script Output | Query Result

LAST_NAME	DEPARTMENT_ID	SALARY	COMM_PCT	CURRENT_FULL_SAL
Hartstein	20	13000	0	13000
Fay	20	6000	0	6000
Mavris	40	6500	0	6500
Baer	70	10000	0	10000
Russell	80	14000	0.4	19600
Partners	80	13500	0.3	17550
Errazuriz	80	12000	0.3	15600
Cambrault	80	11000	0.3	14300
Zlotkey	80	10500	0.2	12600
Tucker	80	10000	0.3	13000
Bernstein	80	9500	0.25	11750
Hall	80	9000	0.25	11250
Olsen	80	8000	0.2	9600
Cambrault	80	7500	0.2	9000
Tuvault	80	7000	0.15	8050
King	80	10000	0.35	13500
Sully	80	9500	0.35	12825
...	...	...	...	...

Line 11 Column 47 | Insert | Modified | Windows: O

Oracle SQL Developer : SQLclass

File Edit View Navigate Run Source Team Tools Window Help

Connections Oracle Connections SQLclass Database Schema Service Connections

Worksheet Query Builder

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.
*/
```

```
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');
```

Script Output | Query Result

LAST_NAME	DEPARTMENT_ID	SALARY	COMM_PCT	CURRENT_FULL_SAL
Vishney	80	10500	0.25	13125
Greene	80	9500	0.15	10925
Marvins	80	7200	0.1	7920
Lee	80	6800	0.1	7480
Ande	80	6400	0.1	7040
Banda	80	6200	0.1	6820
Ozer	80	11500	0.25	14375
Bloom	80	10000	0.2	12000
Fox	80	9600	0.2	11520
Smith	80	7400	0.15	8510
Bates	80	7300	0.15	8395
Kumar	80	6100	0.1	6710
Abel	80	11000	0.3	14300
Button	80	8800	0.25	11000
Taylor	80	8400	0.2	10320
Livingston	80	8400	0.2	10080
Johnson	80	6200	0.1	6820
...	...	...	...	...

Line 11 Column 47 | Insert | Modified | Windows: O

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

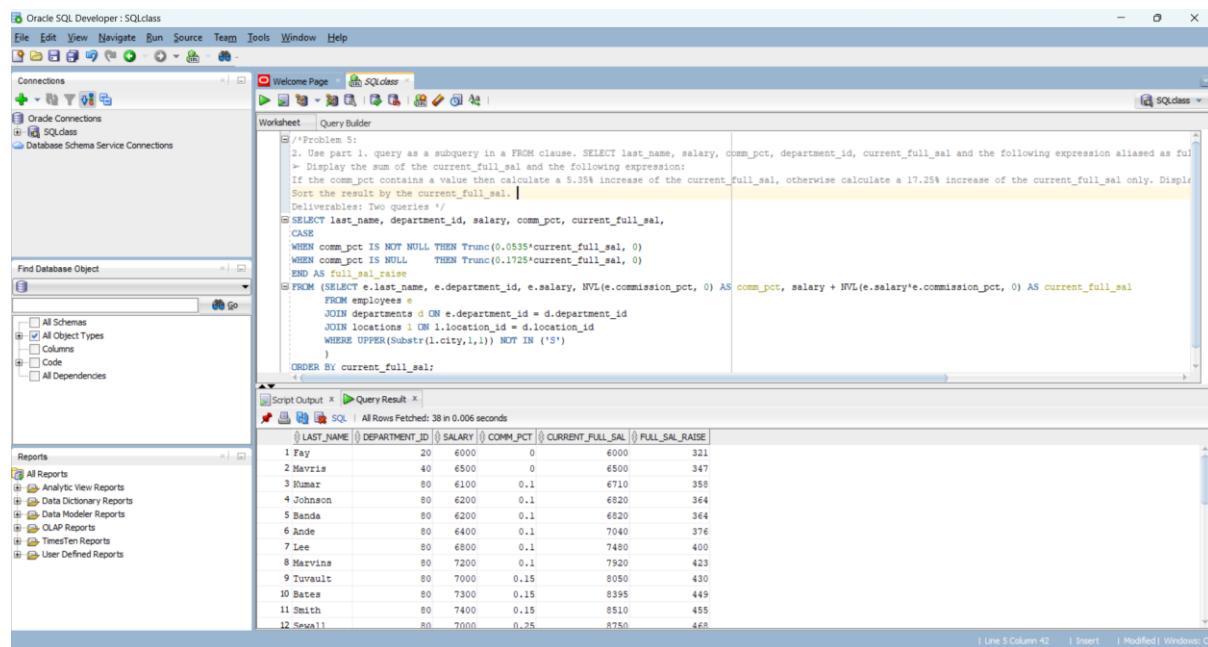
JOIN departments d ON e.department\_id = d.department\_id

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;



The screenshot shows the Oracle SQL Developer interface. The top menu bar includes File, Edit, View, Navigate, Run, Source, Team, Tools, Window, and Help. The left sidebar contains sections for Connections, Find Database Object (with filters for All Schemas, All Object Types, Code, and All Dependencies), and Reports. The central workspace has a 'Worksheet' tab open, displaying the SQL query. The 'Script Output' tab at the bottom shows the execution results:

LAST_NAME	DEPARTMENT_ID	SALARY	COMM_PCT	CURRENT_FULL_SAL	FULL_SAL_RAISE
1 Fay	20	6000	0	6000	321
2 Mavris	40	6500	0	6500	347
3 Kumar	80	6100	0.1	6710	358
4 Johnson	80	6200	0.1	6620	364
5 Banda	80	6200	0.1	6620	364
6 Ande	80	6400	0.1	7040	376
7 Lee	80	6800	0.1	7480	400
8 Marvin	80	7200	0.1	7920	423
9 Turault	80	7000	0.15	8050	430
10 Bates	80	7300	0.15	8395	449
11 Smith	80	7400	0.15	8510	455
12残缺	80	7000	0.25	8750	468

The screenshot shows the Oracle SQL Developer interface. In the Worksheet tab, there is a multi-line SQL script. The first part of the script is a comment block starting with `/*FromLine 5`. It contains instructions for part 1, including a subquery in a FROM clause, a CASE statement, and a WHERE clause. The second part of the script is a SELECT statement that joins the employees table with the departments table using an outer join on department\_id. It also includes a WHERE clause to filter locations. The final ORDER BY clause is `ORDER BY current_full_sal;`. In the Query Result tab, the output shows a table with columns: LAST\_NAME, DEPARTMENT\_ID, SALARY, COMM\_PCT, CURRENT\_FULL\_SAL, and FULL\_SAL\_RAISE. The data consists of 38 rows of employee information.

LAST_NAME	DEPARTMENT_ID	SALARY	COMM_PCT	CURRENT_FULL_SAL	FULL_SAL_RAISE
1 Jitney	60	10500	0.2	12600	674
28 Sully	80	9500	0.35	12625	666
29 Tucker	80	10000	0.3	13000	695
30 Hartstein	20	13000	0	13000	695
31 Vishney	80	10500	0.25	13125	702
32 King	80	10000	0.35	13500	722
33 Abel	80	11000	0.3	14300	765
34 Cambrault	80	11000	0.3	14300	765
35 Ozer	80	11500	0.25	14375	769
36 Errazuriz	80	12000	0.3	15600	834
37 Partners	80	13500	0.3	17550	938
38 Russell	80	14000	0.4	19600	1048

**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;
```

Oracle SQL Developer : SQLclass

File Edit View Navigate Run Source Team Window Help

Connections Oracle Connections Database Schema Service Connections

Worksheet Query Builder

```
/*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.
*/
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;
```

Script Output | Query Result | All Rows Fetched: 107 in 0.014 seconds

LAST_NAME	DEPARTMENT_ID	SALARY	COMM_PCT	CURRENT_FULL_SAL	DEPARTMENT_NAME
Whalen	10	4400	0	4400	Administration
Hartstein	20	13000	0	13000	Marketing
Fay	20	6000	0	6000	Marketing
Raphaely	30	11000	0	11000	Purchasing
Khoo	30	3100	0	3100	Purchasing
Baida	30	2900	0	2900	Purchasing
Tobias	30	2800	0	2800	Purchasing
Himuro	30	2600	0	2600	Purchasing
Colmenares	30	2500	0	2500	Purchasing
Mavris	40	6500	0	6500	Human Resources
Weiss	50	8000	0	8000	Shipping
Fripp	50	8200	0	8200	Shipping
Kaufling	50	7900	0	7900	Shipping
Vollman	50	6500	0	6500	Shipping
Mourgos	50	5800	0	5800	Shipping
Mayer	50	3200	0	3200	Shipping
Mikkilineni	50	2700	0	2700	Shipping
Landry	50	2400	0	2400	Shipping
Markle	50	2200	0	2200	Shipping

| Line 4 Column 82 | Insert | Modified | Windows: C

Oracle SQL Developer : SQLclass

File Edit View Navigate Run Source Team Window Help

Connections Oracle Connections Database Schema Service Connections

Worksheet Query Builder

```
/*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.
*/
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;
```

Script Output | Query Result | All Rows Fetched: 107 in 0.014 seconds

LAST_NAME	DEPARTMENT_ID	SALARY	COMM_PCT	CURRENT_FULL_SAL	DEPARTMENT_NAME
Bates	80	7300	0.15	8355	Sales
Kumar	80	6100	0.1	6710	Sales
Abel	80	11000	0.3	14300	Sales
Hutton	80	8800	0.25	11000	Sales
Taylor	80	8600	0.2	10320	Sales
Livingston	80	8400	0.2	10080	Sales
Johnson	80	6200	0.1	6820	Sales
King	90	24000	0	24000	Executive
Kochhar	90	17000	0	17000	Executive
De Haan	90	17000	0	17000	Executive
Greenberg	100	12000	0	12000	Finance
Faviet	100	9000	0	9000	Finance
Chen	100	8200	0	8200	Finance
Sciarras	100	7700	0	7700	Finance
Urman	100	7800	0	7800	Finance
Popp	100	6900	0	6900	Finance
Higgins	110	12000	0	12000	Accounting
Gietz	110	8300	0	8300	Accounting
Grant	(null)	7000	0.15	8050	(null)

| Line 4 Column 82 | Insert | Modified | Windows: C

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 ;
```

The screenshot shows the Oracle SQL Developer interface. The top menu bar includes File, Edit, View, Navigate, Run, Source, Team, Tools, Window, and Help. The left sidebar contains sections for Connections, Find Database Object, and Reports. The main workspace has a 'Worksheet' tab open, displaying the SQL query. Below the worksheet is a 'Script Output' tab showing the execution time (12 ms) and a 'Query Result' tab displaying the output of the query. The output table is as follows:

DEPARTMENT_NAME	SUM_CURRENT_SAL	SUM_SALARY_RAISE
1 Accounting	20300	1086
2 Administration	4400	235
3 Executive	58000	3102
4 Finance	51600	2758
5 Human Resources	6500	347
6 IT	28800	1538
7 Marketing	19000	1016
8 Public Relations	10000	535
9 Purchasing	24900	1329
10 Sales	377140	20163
11 Shipping	156400	8345
12 (null)	8050	430

**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:

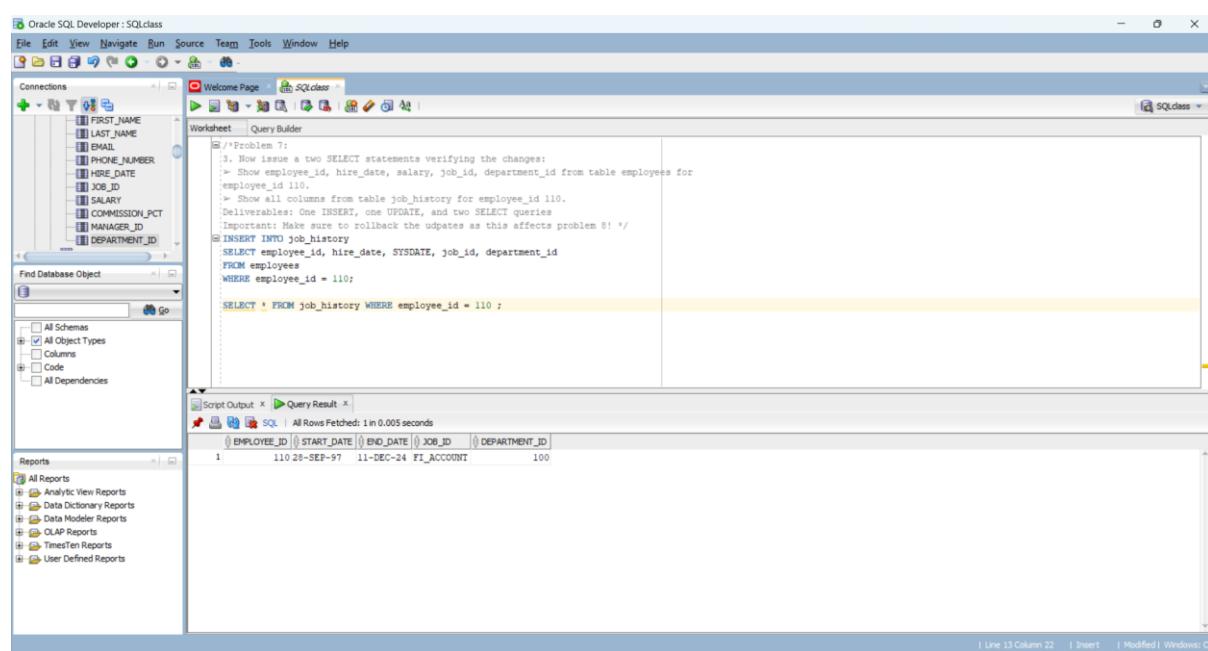
- Show employee\_id, hire\_date, salary, job\_id, department\_id from table employees for employee\_id 110.
- Show all columns from table job\_history for employee\_id 110.

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

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

3.

```
SELECT * FROM job_history WHERE employee_id = 110 ;
```



The screenshot shows the Oracle SQL Developer interface. The 'Worksheet' tab is active, displaying the following SQL code:

```
/*Problem 7:  
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 updates as this affects problem 8! */  
INSERT INTO job_history  
SELECT employee_id, hire_date, SYSDATE, job_id, department_id  
FROM employees  
WHERE employee_id = 110;  
  
SELECT * FROM job_history WHERE employee_id = 110 ;
```

The 'Script Output' tab shows the results of the last query:

EMPLOYEE_ID	START_DATE	END_DATE	JOB_ID	DEPARTMENT_ID
1	110-SEP-97	11-DEC-24	FI_ACCOUNT	100

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

The screenshot shows the Oracle SQL Developer interface. The Worksheet pane contains a script with several UPDATE statements. The first UPDATE statement is:

```

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;

```

The Query Result pane shows the output of the query:

EMPLOYEE_ID	HIRE_DATE	SALARY	JOB_ID	DEPARTMENT_ID
1	110-28-SEF-97	12100	AC_MGR	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;

Oracle SQL Developer : SQLclass

File Edit View Navigate Run Source Team Tools Window Help

Connections Oracle Connections SQLclass Database Schema Service Connections

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

Worksheet Query Builder

```
/*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
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,
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 employee;
```

Query Result X SQL | All Rows Fetched: 107 in 0.092 seconds

LAST_NAME	SALARY	HIRE_DATE	LAG_1	LAG_2	LAG_3
King	24000	17-JUN-87	0	0	0
Whalen	4400	17-SEP-87	24000	0	0
Kochhar	17000	21-SEP-87	4400	24000	0
Hunold	9000	03-JAN-90	17000	4400	24000
Ernst	4000	21-MAY-91	9000	17000	4400
De Haan	17000	13-JAN-93	6000	9000	17000
Mavris	6500	07-JUN-94	17000	6000	9000
Baer	10000	07-JUN-94	6500	17000	6000
Higgins	12000	07-JUN-94	10000	6500	17000
Gietz	8300	07-JUN-94	12000	10000	6500
Faviet	9000	16-AUG-94	8300	12000	10000
Greenberg	12000	17-AUG-94	9000	8300	12000
Raphaely	11000	07-DEC-94	12000	9000	8300
Kaufmann	7600	01-MAY-95	11000	12000	9000

| Line 16 Column 14 | Insert | Modified | Windows | O

Oracle SQL Developer : SQLclass

File Edit View Navigate Run Source Team Tools Window Help

Connections Oracle Connections SQLclass Database Schema Service Connections

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

Worksheet Query Builder

```
/*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
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,
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 employee;
```

Query Result X SQL | All Rows Fetched: 107 in 0.092 seconds

LAST_NAME	SALARY	HIRE_DATE	LAG_1	LAG_2	LAG_3
Popp	6500	07-DEC-99	7000	5800	11000
Gepp	2400	12-DEC-99	6900	7000	5800
Perkins	2500	19-DEC-99	2400	6900	7000
Johnson	4200	04-JAN-00	2500	2400	6900
Grant	2600	13-JAN-00	6200	2500	2400
Marvins	7200	24-JAN-00	2600	6200	2500
Zlotkey	10500	29-JAN-00	7200	2600	6200
Geoni	2800	03-FEB-00	10500	7200	2600
Philtanker	2200	06-FEB-00	2800	10500	7200
Lee	6800	23-FEB-00	2200	2800	10500
Markle	2200	08-MAR-00	6800	2200	2800
Ande	6400	24-MAR-00	2200	6800	2200
Banda	6200	21-APR-00	6400	2200	6800
Kumar	6100	21-APR-00	6200	6400	2200

| Line 16 Column 14 | Insert | Modified | Windows | O

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.

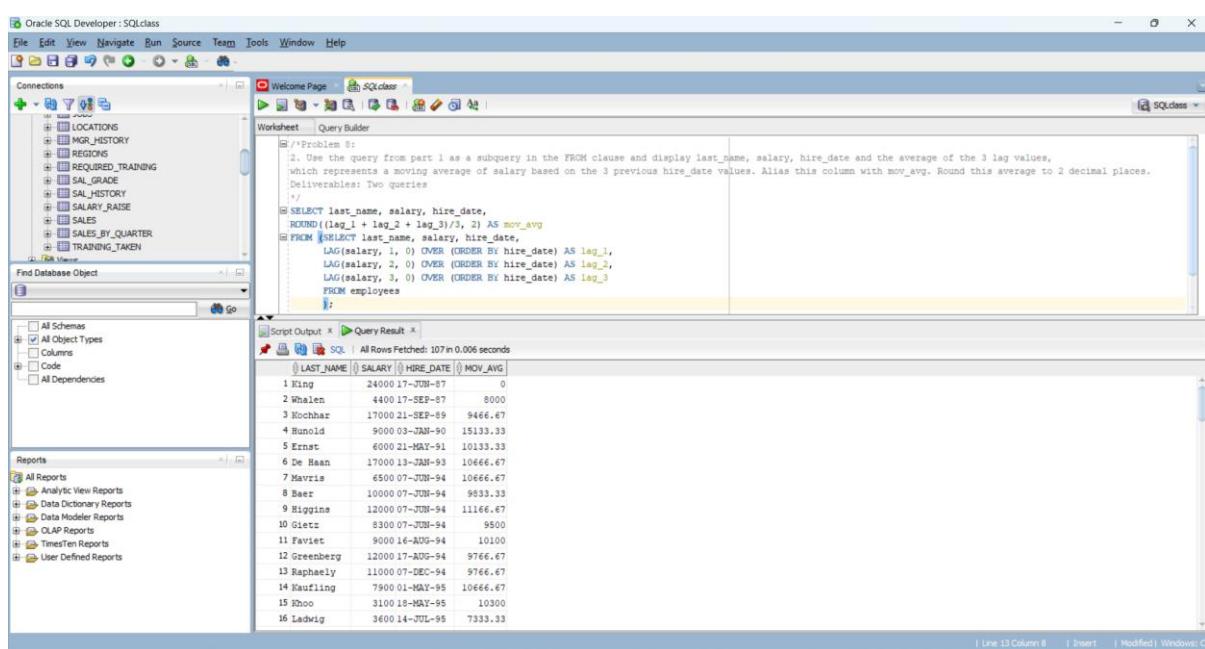
Deliverables: Two queries

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
      );

```



The screenshot shows the Oracle SQL Developer interface with the following details:

- Toolbar:** File, Edit, View, Navigate, Run, Source, Team, Tools, Window, Help.
- Connections:** A tree view showing various database objects like LOCATIONS, MGR\_HISTORY, PAYROLL, REQUIRED\_TRAINING, SAL\_GRADE, SAL\_HISTORY, SALARY\_RAISE, SALES, and SALES\_BY\_QUARTER.
- Worksheet:** Displays the SQL query from the code block above.
- Script Output:** Shows the execution results of the query.
- Query Result:** A table showing the results of the query. The columns are LAST\_NAME, SALARY, HIRE\_DATE, and MOV\_AVG. The data is as follows:

LAST_NAME	SALARY	HIRE_DATE	MOV_AVG
King	24000	17-JUN-87	0
Whalen	4400	21-SEP-87	8000
Kochhar	17000	21-SEP-87	9466.47
Hunold	9000	03-JAN-90	15133.33
Ernst	6000	21-MAY-91	10133.33
De Haan	17000	13-JAN-93	10666.47
Mavris	6500	07-JUN-94	10666.47
Baer	10000	07-JUN-94	9833.33
Higgins	12000	07-JUN-94	11166.47
Gietz	8300	07-JUN-94	9500
Faviet	9000	16-AUG-94	10100
Greenberg	12000	17-APR-94	9766.47
Raphaely	11000	07-DEC-94	9766.47
Kaufling	7900	01-MAY-95	10666.47
Shoo	3100	18-MAY-95	10300
Ladwig	3600	14-JUL-95	7333.33

Oracle SQL Developer : SQLclass

File Edit View Navigate Run Source Team Tools Window Help

Connections

Worksheet Query Builder

```
/*Problem c:
1. 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 X

All Rows Fetched: 107 in 0.005 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 Marvin	7200	24-JAN-00	3766.67
100 Zlotkey	10500	29-JAN-00	5333.33
101 Geroni	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

| Line 13 Column 8 | Insert | Modified | Windows | C

-THE END-