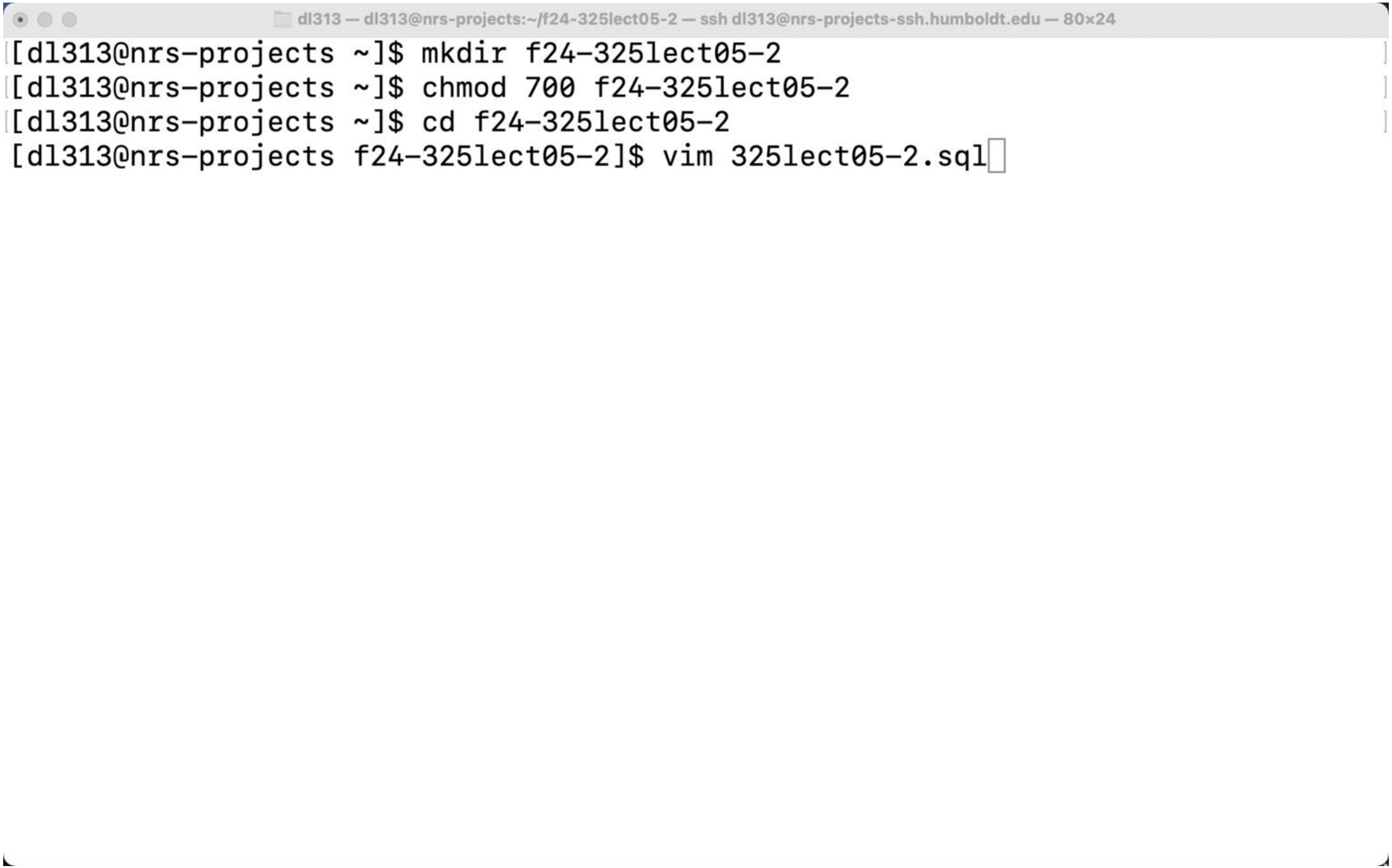
CS 325 Week 6-1

Computed columns, column aliases, aggregate Functions, and sub select in SQL

dl313 — dl313@nrs-projects:~/f24-325lect05-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
[dl313@nrs-projects ~]$ mkdir f24-325lect05-2
[dl313@nrs-projects ~]$ chmod 700 f24-325lect05-2
[dl313@nrs-projects ~]$ cd f24-325lect05-2
[dl313@nrs-projects f24-325lect05-2]$
```


dl313 — dl313@nrs-projects:~/f24-325lect05-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24
[dl313@nrs-projects ~]$ mkdir f24-325lect05-2
[dl313@nrs-projects ~]$ chmod 700 f24-325lect05-2
[dl313@nrs-projects ~]$ cd f24-325lect05-2
[dl313@nrs-projects f24-325lect05-2]$ vim 325lect05-2.sql

dl313 — dl313@nrs-projects:~/f24-325lect05-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
/*=====
    a SQL select statement can do MORE than just relational operations...

    for example, a SELECT clause can project more than just columns.

    for example, you can specify that a computation on a column be
    projected, and then it will be done for each selected row from that
    query
    *   THIS DOES NOT CHANGE the DATA IN THAT TABLE!!!!! it just
        PROJECTS a result with this computation showing

=====*/
~
~
~
~
~
~
~
~
~
~
~
-- INSERT --                                              12,8          All
```

dl313 — dl313@nrs-projects:~/f24-325lect05-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
/*=====
    a SQL select statement can do MORE than just relational operations...

    for example, a SELECT clause can project more than just columns.

    for example, you can specify that a computation on a column be
    projected, and then it will be done for each selected row from that
    query
    *   THIS DOES NOT CHANGE the DATA IN THAT TABLE!!!!! it just
        PROJECTS a result with this computation showing

=====*/

prompt ========
prompt project empl last names and salaries:

select empl_last_name, salary
from   empl;
~
~
~
~
~
-- INSERT --                                          18,13         All
```

dl313 — dl313@nrs-projects:~/f24-325lect05-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
SQL> @ 325lect05-2.sql
========
project empl last names and salaries:

EMPL_LAST_NAME      SALARY
--------------- ----------
King                  5000
Jones                 2975
Blake                 2850
Raimi                 2450
Ford                  3000
Smith                  800
Michaels              1600
Ward                  1250
Martin                1250
Scott                 3000
Turner                1500

EMPL_LAST_NAME      SALARY
--------------- ----------
Adams                 1100
James                  950
Miller                1300
```

dl313 — dl313@nrs-projects:~/f24-325lect05-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
prompt ========
prompt project empl last names and salaries:

select empl_last_name, salary
from   empl;

prompt ========
prompt project empl last names and 2 * salaries:

select empl_last_name, salary * 2
from   empl;
```

-- INSERT -- 35,1 Bot

dl313 — dl313@nrs-projects:~/f24-325lect05-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
SQL> @ 325lect05-2.sql
```

```
========
project empl last names and 2 * salaries:

EMPL_LAST_NAME    SALARY*2
--------------- ----------
King                10000
Jones                5950
Blake                5700
Raimi                4900
Ford                 6000
Smith                1600
Michaels             3200
Ward                 2500
Martin               2500
Scott                6000
Turner               3000

EMPL_LAST_NAME    SALARY*2
--------------- ----------
Adams                2200
James                1900
Miller               2600

14 rows selected.
```

dl313 — dl313@nrs-projects:~/f24-325lect05-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
prompt ========
prompt project empl last names and 2 * salaries:

select empl_last_name, salary * 2
from   empl;

-- empl salaries have NOT changed as a result!!!!!!!!

prompt ========
prompt project empl last names and salaries
prompt (note that projecting a computed column does not change
prompt that column!)

select empl_last_name, salary
from   empl;


```

-- INSERT -- 37,1 95%

dl313 — dl313@nrs-projects:~/f24-325lect05-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
SQL> @ 325lect05-2.sql
```

```
========
project empl last names and salaries
(note that projecting a computed column does not change
that column!)

EMPL_LAST_NAME       SALARY
--------------- ----------
King                   5000
Jones                  2975
Blake                  2850
Raimi                  2450
Ford                   3000
Smith                   800
Michaels               1600
Ward                   1250
Martin                 1250
Scott                  3000
Turner                 1500

EMPL_LAST_NAME       SALARY
--------------- ----------
Adams                  1100
James                   950
Miller                 1300
```

dl313 — dl313@nrs-projects:~/f24-325lect05-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
/*=======
    hey, guess what?
    you can ASK the select statement to use a DIFFERENT column
    heading for the columns in its result;

    ONE way to do this (using SQL and not other means)
    is with a COLUMN ALIAS

    IN a select clause,
    after a projected expression, you can put a blank and then
       the desired alternate title <-- that's the column alias

    *   if you don't surround it with DOUBLE (!!) quotes,
        it will be displayed in all-caps
    *   if you DO surround it with DOUBLE (!!) quotes,
        it will be displayed in the case shown within the quotes
    *   (it CANNOT contain blanks unless it is in double quotes)

    *   THIS DOES NOT CHANGE the COLUMN NAMES IN THAT TABLE!!!!!
        it just projects a DIFFERENT column heading for THAT
        select's results!
=====*/
```

-- INSERT -- 53,1 85%

dl313 — dl313@nrs-projects:~/f24-325lect05-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
    IN a select clause,
    after a projected expression, you can put a blank and then
       the desired alternate title <-- that's the column alias

    *   if you don't surround it with DOUBLE (!!) quotes,
        it will be displayed in all-caps
    *   if you DO surround it with DOUBLE (!!) quotes,
        it will be displayed in the case shown within the quotes
    *   (it CANNOT contain blanks unless it is in double quotes)

    *   THIS DOES NOT CHANGE the COLUMN NAMES IN THAT TABLE!!!!!
        it just projects a DIFFERENT column heading for THAT
        select's results!
=====*/

prompt ========
prompt playing with column aliases!

select empl_last_name, salary * 2 if_raise
from empl;



-- INSERT --                                          61,1          95%
```

dl313 — dl313@nrs-projects:~/f24-325lect05-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
SQL> @ 325lect05-2.sql
```

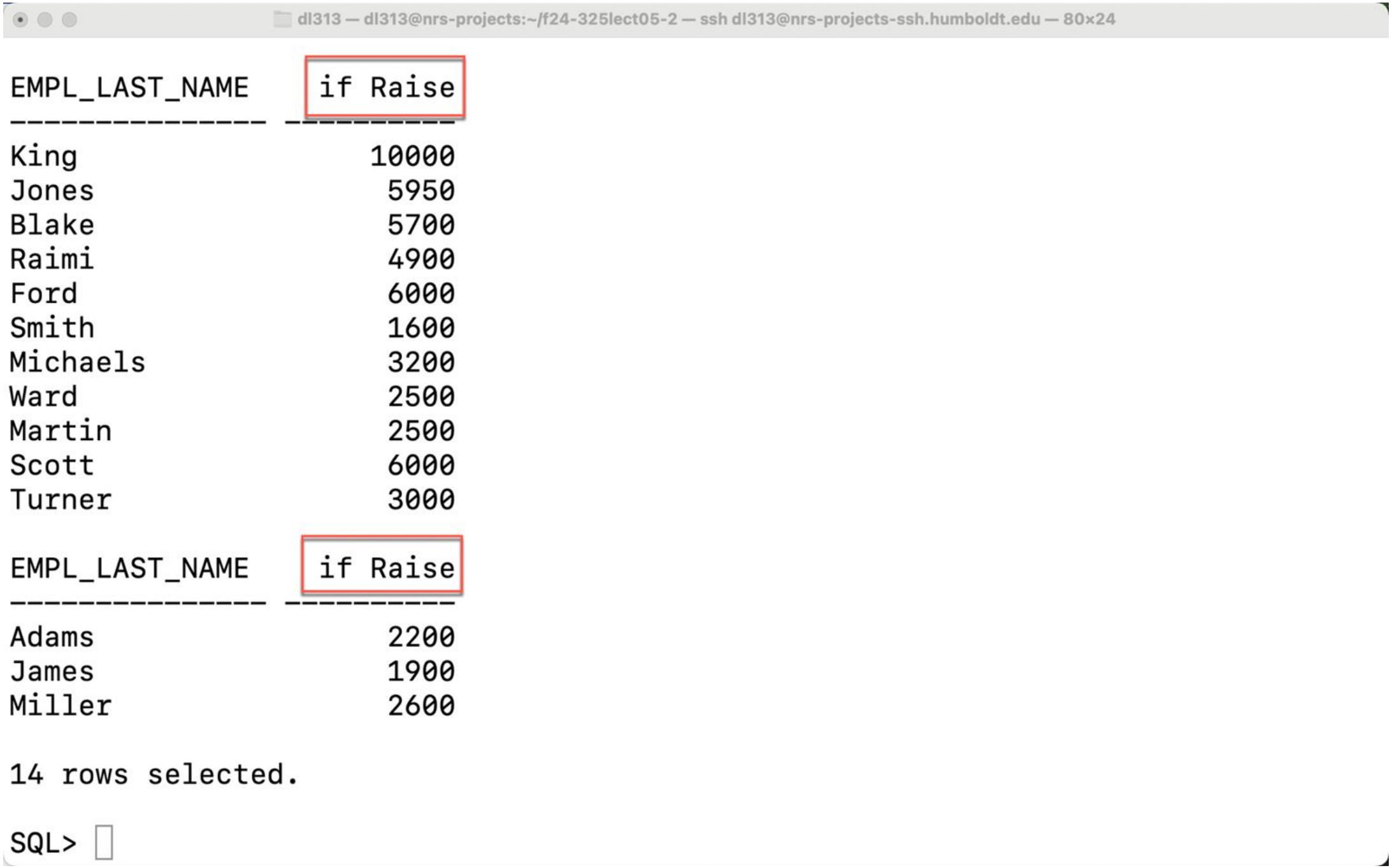
dl313 — dl313@nrs-projects:~/f24-325lect05-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
========
playing with column aliases!

EMPL_LAST_NAME    IF_RAISE
--------------- ----------
King                 10000
Jones                 5950
Blake                 5700
Raimi                 4900
Ford                  6000
Smith                 1600
Michaels              3200
Ward                  2500
Martin                2500
Scott                 6000
Turner                3000

EMPL_LAST_NAME    IF_RAISE
--------------- ----------
Adams                 2200
James                 1900
Miller                2600

14 rows selected.
```

```
    *   if you don't surround it with DOUBLE (!!) quotes,
        it will be displayed in all-caps
    *   if you DO surround it with DOUBLE (!!) quotes,
        it will be displayed in the case shown within the quotes
    *   (it CANNOT contain blanks unless it is in double quotes)

    *   THIS DOES NOT CHANGE the COLUMN NAMES IN THAT TABLE!!!!!
        it just projects a DIFFERENT column heading for THAT
        select's results!
=====*/

prompt ========
prompt playing with column aliases!

select empl_last_name, salary * 2 if_raise
from empl;

select empl_last_name, salary * 2 "if Raise"
from empl;
```

dl313 — dl313@nrs-projects:~/f24-325lect05-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
SQL> @ 325lect05-2.sql
```


dl313 — dl313@nrs-projects:~/f24-325lect05-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24
EMPL_LAST_NAME if Raise
--------------- ----------
King 10000
Jones 5950
Blake 5700
Raimi 4900
Ford 6000
Smith 1600
Michaels 3200
Ward 2500
Martin 2500
Scott 6000
Turner 3000
EMPL_LAST_NAME if Raise
--------------- ----------
Adams 2200
James 1900
Miller 2600
14 rows selected.
SQL>

```
    *   if you DO surround it with DOUBLE (!!) quotes,
        it will be displayed in the case shown within the quotes
    *   (it CANNOT contain blanks unless it is in double quotes)

    *   THIS DOES NOT CHANGE the COLUMN NAMES IN THAT TABLE!!!!!
        it just projects a DIFFERENT column heading for THAT
        select's results!
=====*/

prompt ========
prompt playing with column aliases!

select empl_last_name, salary * 2 if_raise
from empl;

select empl_last_name, salary * 2 "if Raise"
from empl;

select empl_last_name "Employee", salary * 1.1 "if 10% raise"
from   empl;
```

dl313 — dl313@nrs-projects:~/f24-325lect05-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
SQL> @ 325lect05-2.sql
```

dl313 — dl313@nrs-projects:~/f24-325lect05-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
Employee        if 10% raise
--------------- ------------
King                    5500
Jones                 3272.5
Blake                   3135
Raimi                   2695
Ford                    3300
Smith                    880
Michaels                1760
Ward                    1375
Martin                  1375
Scott                   3300
Turner                  1650

Employee        if 10% raise
--------------- ------------
Adams                   1210
James                   1045
Miller                  1430

14 rows selected.

SQL>
```

Question

What is the correct way to display a column alias in mixed case or with spaces in the result set?

A) Use single quotes around the alias name.

B) Use no quotes at all for the alias name with spaces.

C) Use double quotes around the alias name.

D) Use square brackets around the alias name.

dl313 — dl313@nrs-projects:~/f24-325lect05-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
        select's results!
=====*/

prompt ========
prompt playing with column aliases!

select empl_last_name, salary * 2 if_raise
from empl;

select empl_last_name, salary * 2 "if Raise"
from empl;

select empl_last_name "Employee", salary * 1.1 "if 10% raise"
from   empl;

prompt ========
prompt WILL GET ERROR, column alias must be in NO quotes or DOUBLE quotes:

select empl_last_name, salary * 2 'if Raise'
from empl;

```

-- INSERT -- 77,1 Bot

dl313 — dl313@nrs-projects:~/f24-325lect05-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
SQL> @ 325lect05-2.sql
```

```
Smith                            880
Michaels                        1760
Ward                            1375
Martin                          1375
Scott                           3300
Turner                          1650

Employee        if 10% raise
--------------- ------------
Adams                           1210
James                           1045
Miller                          1430

14 rows selected.

========
WILL GET ERROR, column alias must be in NO quotes or DOUBLE quotes:
select empl_last_name, salary * 2 'if Raise'
                                  *
ERROR at line 1:
ORA-00923: FROM keyword not found where expected


SQL>
```

Question

Which of the following correctly creates a column alias without changing the actual column name in the table?

A) SELECT salary * 2 AS new_salary FROM empl;

B) SELECT salary * 2 new_salary FROM empl;

C) SELECT salary * 2 "New Salary" FROM empl;

D) All of the above

dl313 — dl313@nrs-projects:~/f24-325lect05-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
/*=====
     a caveat!

     computations on columns are NOT done for NULL values of that column!
=====*/

prompt ========
prompt computations will NOT be done on NULL values:

select empl_last_name, salary + commission "Total pay"
from  empl;
```

-- INSERT -- 99,1 Bot

dl313 — dl313@nrs-projects:~/f24-325lect05-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
SQL> @ 325lect05-2.sql
```

```
========
computations will NOT be done on NULL values:

EMPL_LAST_NAME   Total pay
--------------- ----------
King
Jones
Blake
Raimi
Ford
Smith
Michaels              1900
Ward                  1750
Martin                2650
Scott
Turner                1500

EMPL_LAST_NAME   Total pay
--------------- ----------
Adams
James
Miller

14 rows selected.
```

dl313@nrs-projects:~

```
SQL> select empl_last_name, COALESCE(salary + commission, 0) "Total pay"
  2  from empl;

EMPL_LAST_NAME   Total pay
-------------- ----------
King                    0
Jones                   0
Blake                   0
Raimi                   0
Ford                    0
Smith                   0
Michaels             1900
Ward                 1750
Martin               2650
Scott                   0
Turner               1500

EMPL_LAST_NAME   Total pay
-------------- ----------
Adams                   0
James                   0
Miller                  0

14 rows selected.

SQL>
```

dl313@nrs-projects:~

```
SQL> select empl_last_name, NVL(salary + commission, 0) "Total pay"
  2  from empl;

EMPL_LAST_NAME   Total pay
-------------- ----------
King                    0
Jones                   0
Blake                   0
Raimi                   0
Ford                    0
Smith                   0
Michaels             1900
Ward                 1750
Martin               2650
Scott                   0
Turner               1500

EMPL_LAST_NAME   Total pay
-------------- ----------
Adams                   0
James                   0
Miller                  0

14 rows selected.

SQL>
```

dl313 — dl313@nrs-projects:~/f24-325lect05-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
/*======
    TABLES can also have aliases!

    IN the from clause,
    you can follow a table name (or table EXPRESSEION) with a blank and
       a name (or something in double-quotes),
       and that becomes the alias for this tables THROUGHOUT that ONE
       SELECT statement!

    *   why??????
        *   it can make join conditions and references to
            columns with the same name shorter to type...
        *   it can allow joining a table to itself (!!)

    *   CLASS STYLE STANDARD: make the alias at least
        SOMEWHAT related to the table name (even if just the first
        letter of its name)
======*/
```

-- INSERT -- 106,9 94%

dl313 — dl313@nrs-projects:~/f24-325lect05-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
    IN the from clause,
    you can follow a table name (or table EXPRESSEION) with a blank and
       a name (or something in double-quotes),
       and that becomes the alias for this tables THROUGHOUT that ONE
       SELECT statement!

    *   why??????
        *   it can make join conditions and references to
            columns with the same name shorter to type...
        *   it can allow joining a table to itself (!!)

    *   CLASS STYLE STANDARD: make the alias at least
        SOMEWHAT related to the table name (even if just the first
        letter of its name)
======*/

prompt ========
prompt empl last names and dept names, NO table aliases:

select empl_last_name, dept_name
from   empl, dept
where  empl.dept_num = dept.dept_num;

-- INSERT --                                              109,57        91%
```

dl313 — dl313@nrs-projects:~/f24-325lect05-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
SQL> @ 325lect05-2.sql
```

dl313 — dl313@nrs-projects:~/f24-325lect05-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
========
empl last names and dept names, NO table aliases:

EMPL_LAST_NAME  DEPT_NAME
--------------- ---------------
Miller          Accounting
Raimi           Accounting
Scott           Research
Jones           Research
Ford            Research
Smith           Research
Martin          Sales
Ward            Sales
Blake           Sales
Michaels        Sales
James           Sales

EMPL_LAST_NAME  DEPT_NAME
--------------- ---------------
Turner          Sales
Adams           Operations
King            Management

14 rows selected.
```

dl313 — dl313@nrs-projects:~/f24-325lect05-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
prompt ========
prompt empl last names and dept names, NO table aliases:

select empl_last_name, dept_name
from   empl, dept
where  empl.dept_num = dept.dept_num;

prompt ========
prompt empl last names and dept names, WITH table aliases
prompt (no difference in the output!):

select empl_last_name, dept_name
from   empl e, dept d
where  e.dept_num = d.dept_num;
```

125,0-1 Bot

dl313 — dl313@nrs-projects:~/f24-325lect05-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
SQL> @ 325lect05-2.sql
```

```
========
empl last names and dept names, WITH table aliases
(no difference in the output!):

EMPL_LAST_NAME  DEPT_NAME
--------------- ---------------
Miller          Accounting
Raimi           Accounting
Scott           Research
Jones           Research
Ford            Research
Smith           Research
Martin          Sales
Ward            Sales
Blake           Sales
Michaels        Sales
James           Sales

EMPL_LAST_NAME  DEPT_NAME
--------------- ---------------
Turner          Sales
Adams           Operations
King            Management
```

Question

What will happen in the following SQL query if the commission column contains NULL values?

```
SELECT empl_last_name, salary + commission AS "Total Pay"
FROM empl;
```

A) The query will treat NULL as 0 and return the sum of salary + 0.
B) The query will return NULL for rows where commission is NULL.
C) The query will raise an error because NULL values cannot be used in computations.
D) The query will replace NULL with the average value of the commission column.

dl313 — dl313@nrs-projects:~/f24-325lect05-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
prompt ========
prompt empl last names and dept names, WITH table aliases
prompt (no difference in the output!):

select empl_last_name, dept_name
from   empl e, dept d
where  e.dept_num = d.dept_num;

prompt ========
prompt WILL GET ERROR; once you have a table alias,
prompt you must USE it THROUGHOUT that select,
prompt EVEN in its SELECT clause...!

select empl_last_name, dept.dept_num, dept_name
from   empl e, dept d
where  e.dept_num = d.dept_num;
```

-- INSERT -- 137,1 Bot

dl313 — dl313@nrs-projects:~/f24-325lect05-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
SQL> @ 325lect05-2.sql
```

```
Ward            Sales
Blake           Sales
Michaels        Sales
James           Sales

EMPL_LAST_NAME  DEPT_NAME
--------------- ---------------
Turner          Sales
Adams           Operations
King            Management

14 rows selected.

========
WILL GET ERROR; once you have a table alias,
you must USE it THROUGHOUT that select,
EVEN in its SELECT clause...!
select empl_last_name, dept.dept_num, dept_name
                       *
ERROR at line 1:
ORA-00904: "DEPT"."DEPT_NUM": invalid identifier


SQL>
```

dl313 — dl313@nrs-projects:~/f24-325lect05-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
prompt ========
prompt WILL GET ERROR; once you have a table alias,
prompt you must USE it THROUGHOUT that select,
prompt EVEN in its SELECT clause...!

select empl_last_name, dept.dept_num, dept_name
from   empl e, dept d
where  e.dept_num = d.dept_num;

prompt ========
prompt now using the needed table alias in select clause:

select empl_last_name, d.dept_num, dept_name
from   empl e, dept d
where  e.dept_num = d.dept_num;
```

-- INSERT -- 145,1 Bot

dl313 — dl313@nrs-projects:~/f24-325lect05-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
SQL> @ 325lect05-2.sql
```

dl313 — dl313@nrs-projects:~/f24-325lect05-2 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
========
now using the needed table alias in select clause:

EMPL_LAST_NAME  DEP DEPT_NAME
--------------- --- ---------------
Miller          100 Accounting
Raimi           100 Accounting
Scott           200 Research
Jones           200 Research
Ford            200 Research
Smith           200 Research
Martin          300 Sales
Ward            300 Sales
Blake           300 Sales
Michaels        300 Sales
James           300 Sales

EMPL_LAST_NAME  DEP DEPT_NAME
--------------- --- ---------------
Turner          300 Sales
Adams           400 Operations
King            500 Management

14 rows selected.
```

Question

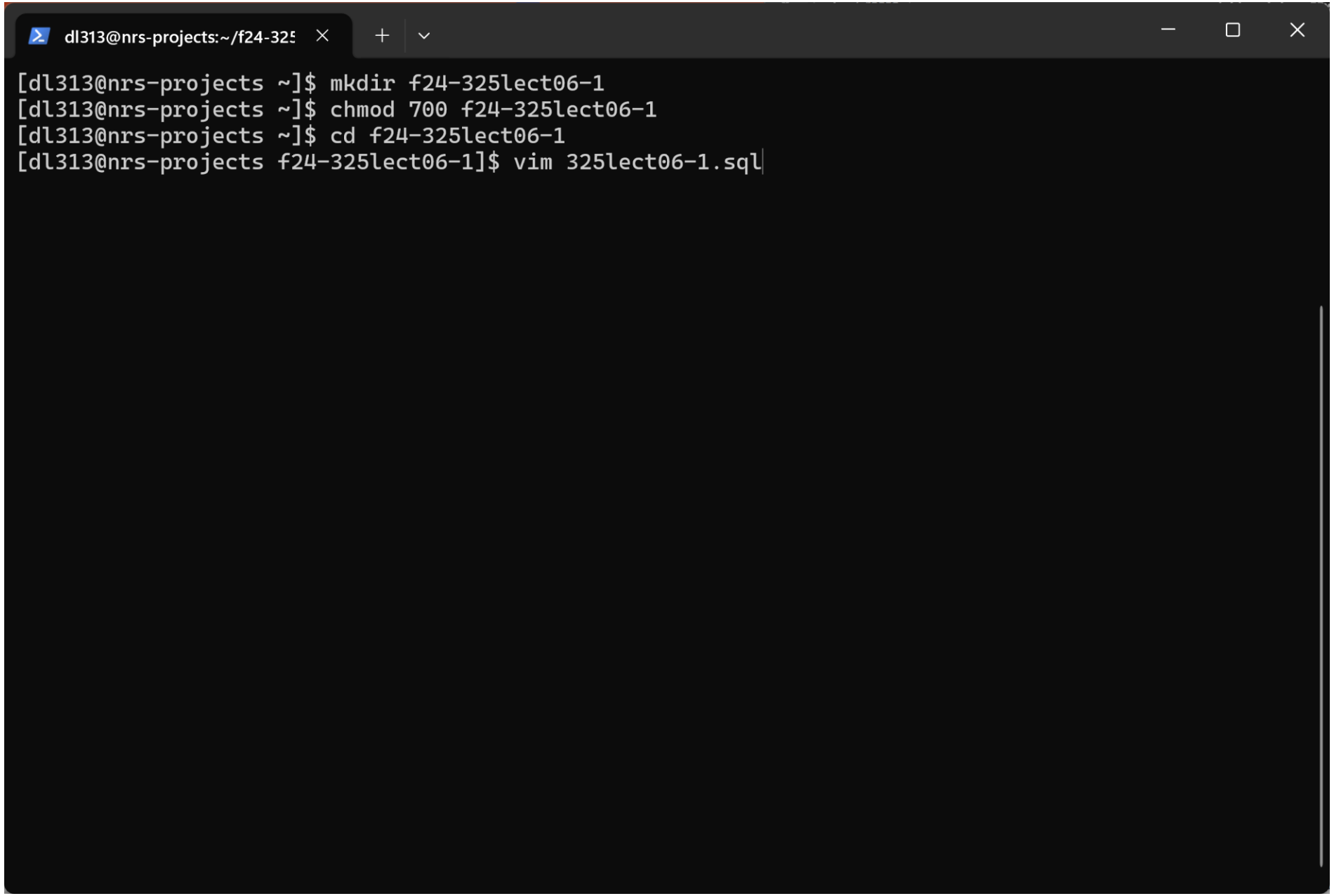
What happens when you try to perform a computation on a column that contains NULL values?

A) The computation will return 0 for NULL values.

B) The computation will be performed, and the result will replace the NULL.

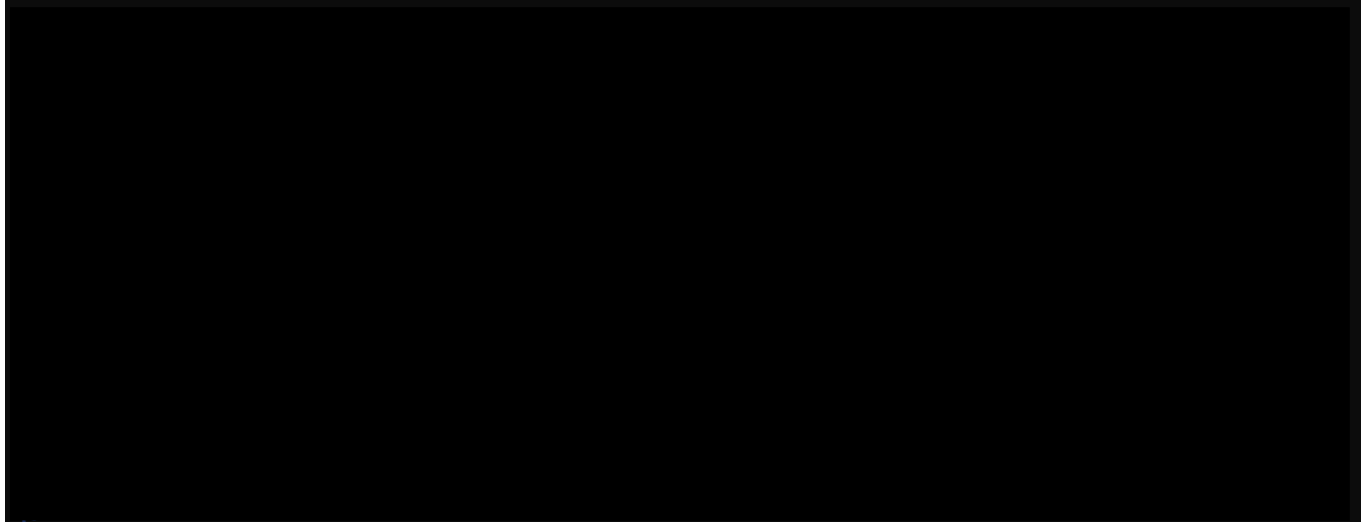
C) The computation will be skipped, and NULL will remain as the result.

D) The query will return an error if NULL values are encountered.


dl313@nrs-projects:~/f24-325
[dl313@nrs-projects ~]$ mkdir f24-325lect06-1
[dl313@nrs-projects ~]$ chmod 700 f24-325lect06-1
[dl313@nrs-projects ~]$ cd f24-325lect06-1
[dl313@nrs-projects f24-325lect06-1]$ vim 325lect06-1.sql

dl313@nrs-projects:~/f24-325 | dl313@nrs-projects:~/f24-325l

```
prompt ========
prompt what if I would like to project, for each customer,
prompt the customer's last name, the name of that customer's employee rep,
prompt and the department location of that employee
```

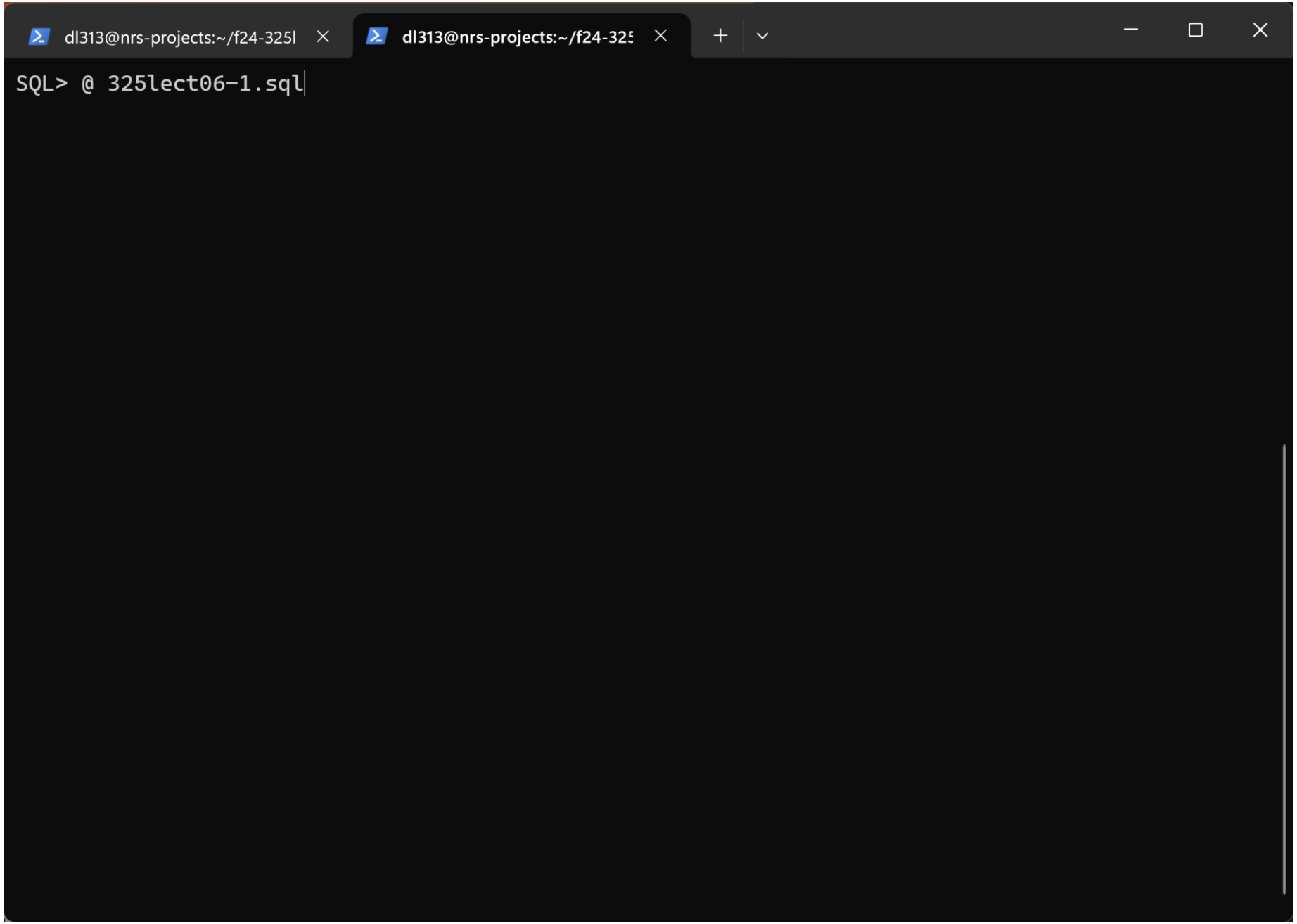


```
~
~
~
~
~
~
~
~
~
~
~
~
"325lect06-1.sql" 18L, 594C written                          14,43         All
```

dl313@nrs-projects:~/f24-325 | dl313@nrs-projects:~/f24-325l

```
prompt ========
prompt what if I would like to project, for each customer,
prompt the customer's last name, the name of that customer's employee rep,
prompt and the department location of that employee

select cust_lname, empl_last_name, dept_loc
from   empl, customer, dept
where  customer.empl_rep = empl.empl_num
       and empl.dept_num = dept.dept_num;
```

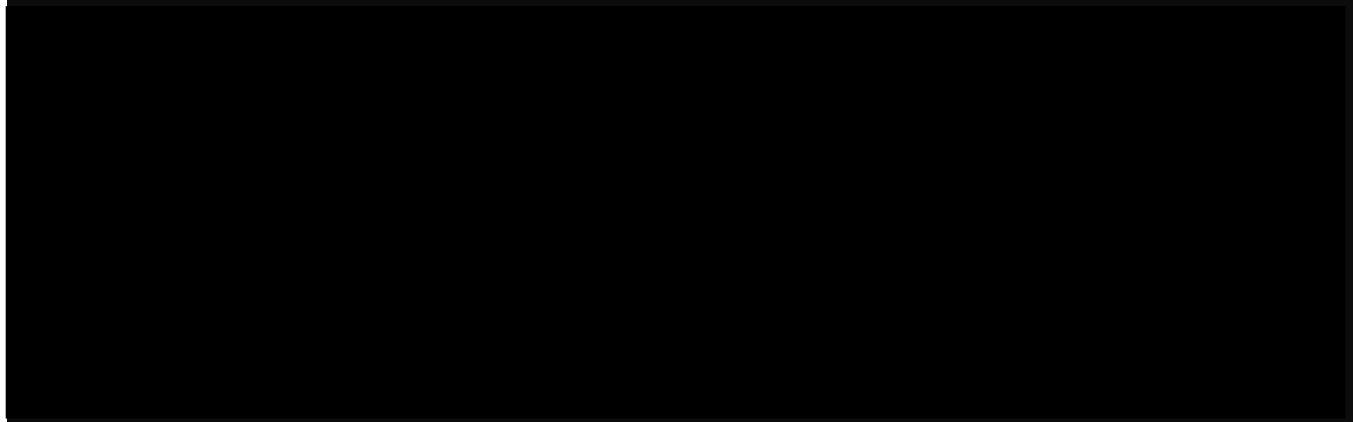
```
~
~
~
~
~
~
~
~
~
~
~
~
"325lect06-1.sql" 18L, 594C written                          14,43         All
```

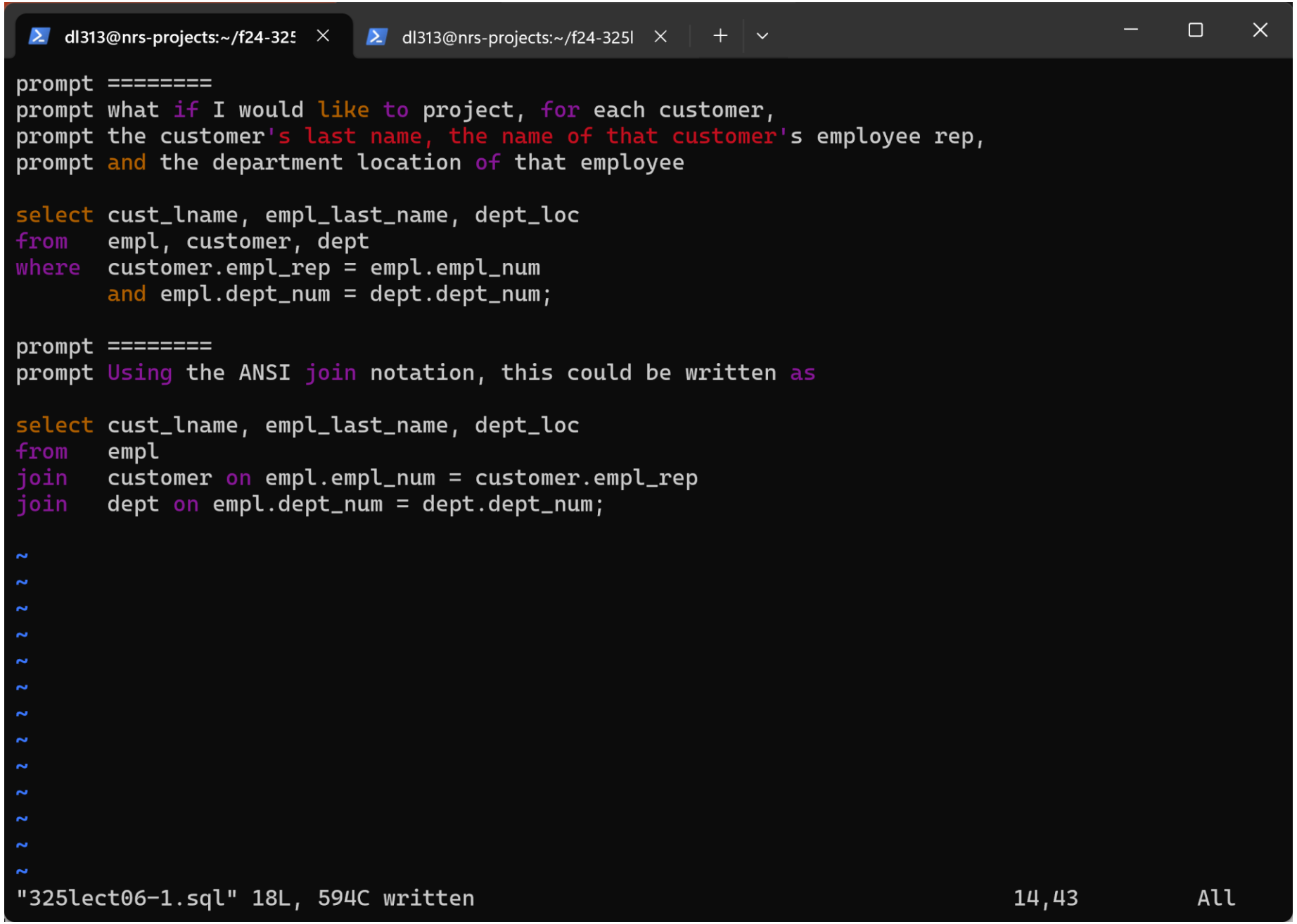

dl313@nrs-projects:~/f24-325l
dl313@nrs-projects:~/f24-325
SQL> @ 325lect06-1.sql

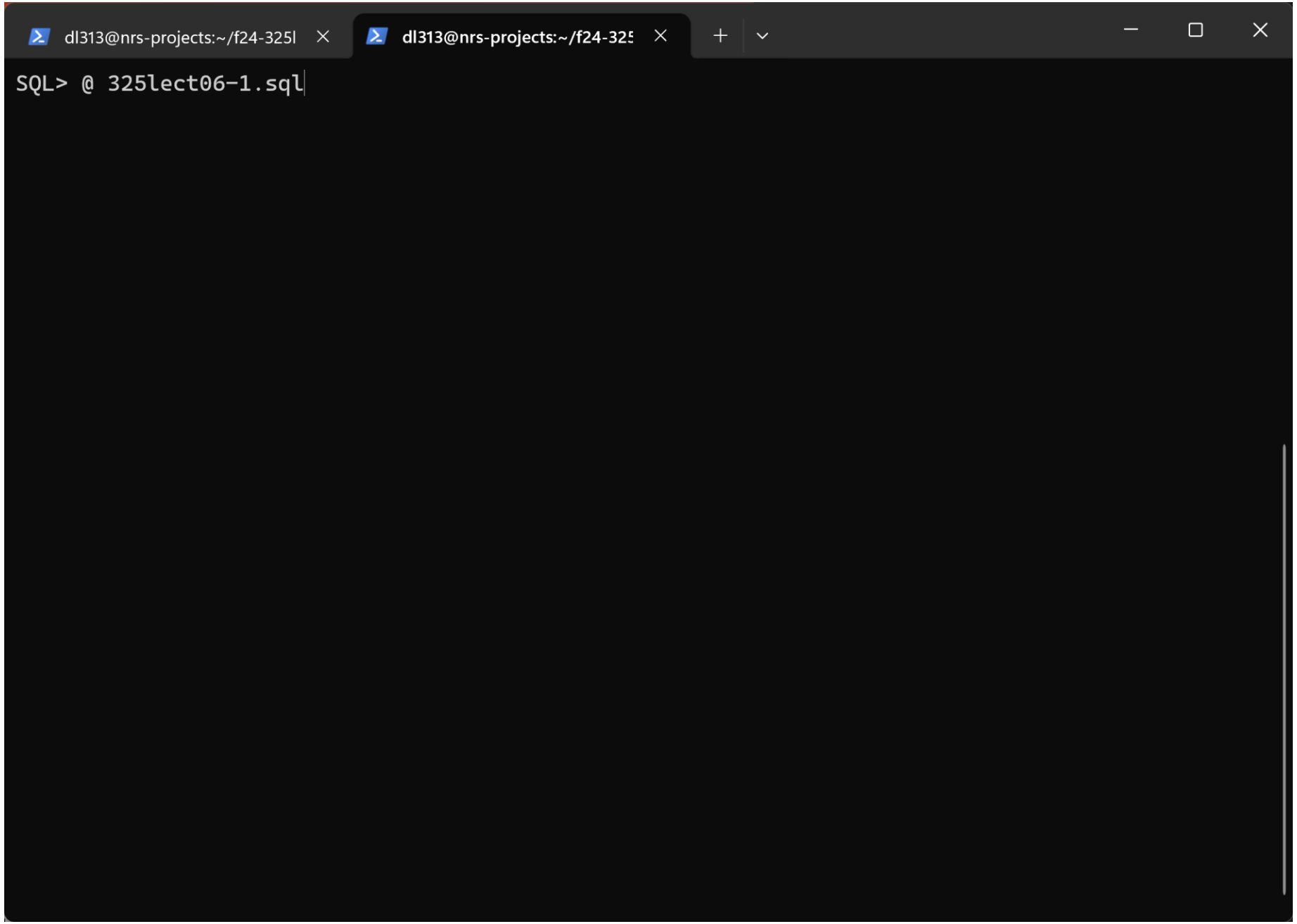
dl313@nrs-projects:~/f24-325l

dl313@nrs-projects:~/f24-325

```
SQL> @ 325lect06-1.sql
========
what if I would like to project, for each customer,
the customer's last name, the name of that customer's employee rep,
and the department location of that employee

CUST_LNAME           EMPL_LAST_NAME  DEPT_LOC
-------------------- --------------- ---------------
Firstly              Michaels        Chicago
Secondly             Martin          Chicago
Thirdly              Michaels        Chicago
```



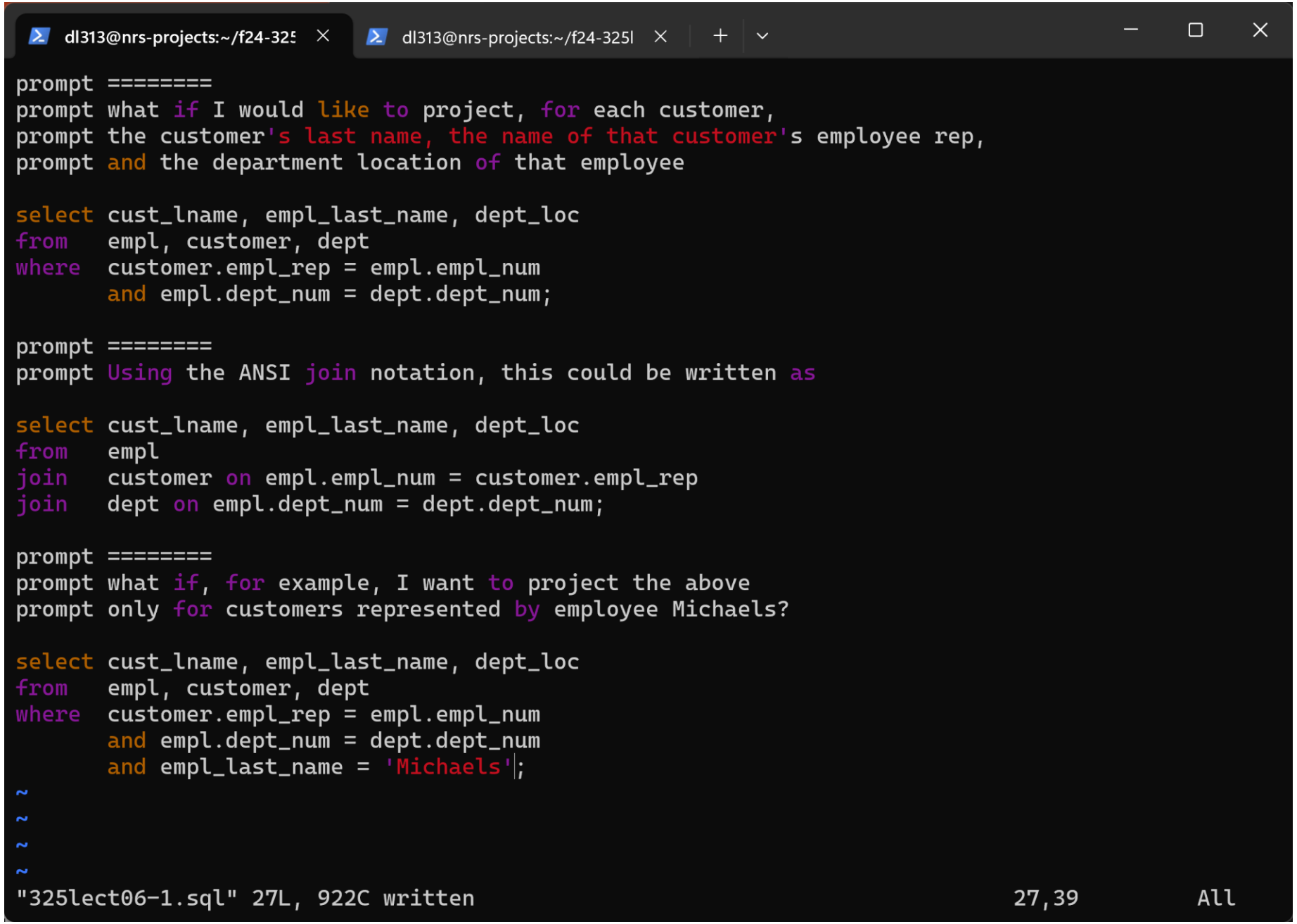

dl313@nrs-projects:~/f24-325
dl313@nrs-projects:~/f24-325l
prompt ========
prompt what if I would like to project, for each customer,
prompt the customer's last name, the name of that customer's employee rep,
prompt and the department location of that employee
select cust_lname, empl_last_name, dept_loc
from empl, customer, dept
where customer.empl_rep = empl.empl_num
and empl.dept_num = dept.dept_num;
prompt ========
prompt Using the ANSI join notation, this could be written as
select cust_lname, empl_last_name, dept_loc
from empl
join customer on empl.empl_num = customer.empl_rep
join dept on empl.dept_num = dept.dept_num;
"325lect06-1.sql" 18L, 594C written
14,43
All

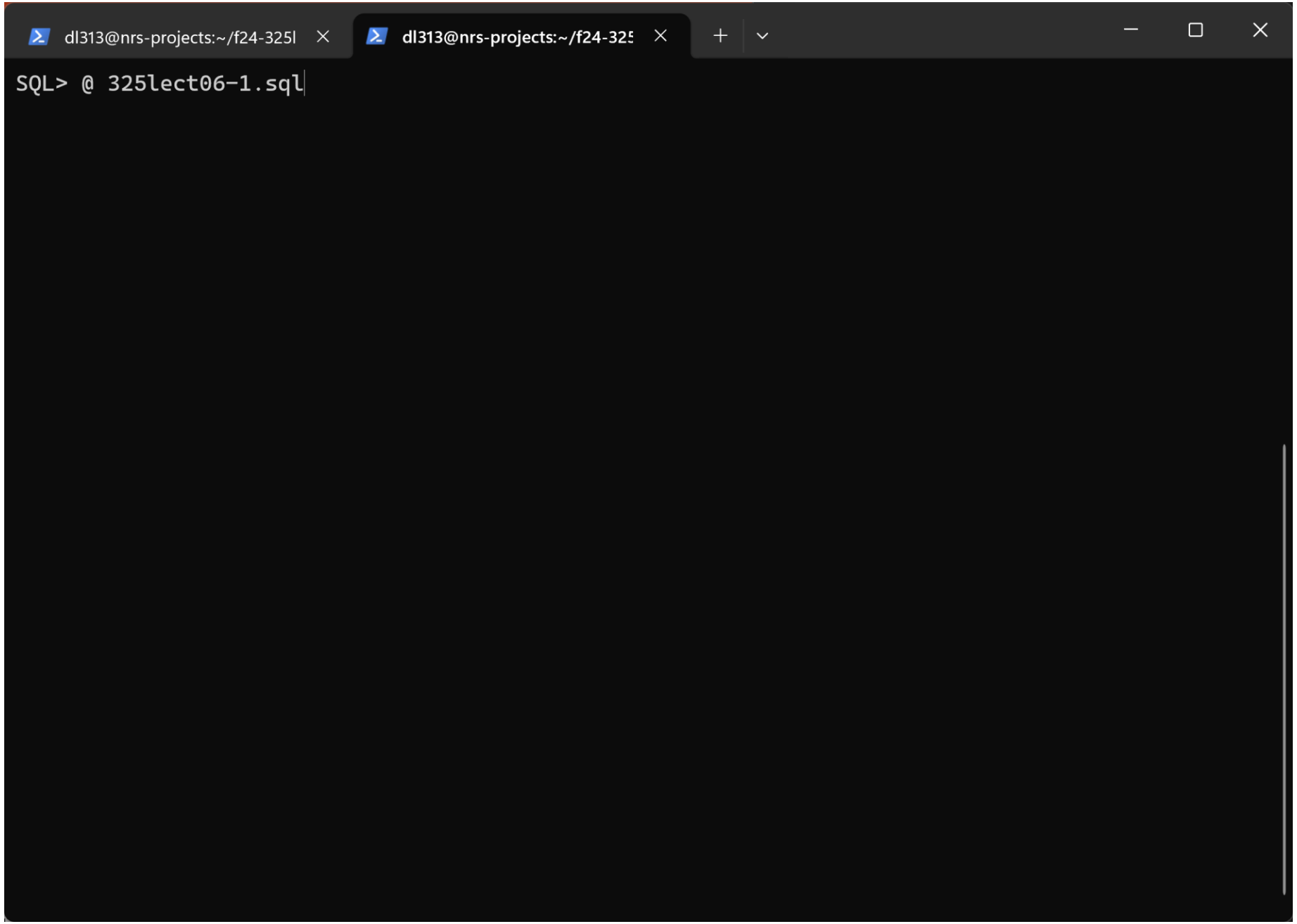

dl313@nrs-projects:~/f24-325l
dl313@nrs-projects:~/f24-325
SQL> @ 325lect06-1.sql

dl313@nrs-projects:~/f24-325l

dl313@nrs-projects:~/f24-325

```
SQL> @ 325lect06-1.sql
========
what if I would like to project, for each customer,
the customer's last name, the name of that customer's employee rep,
and the department location of that employee

CUST_LNAME           EMPL_LAST_NAME  DEPT_LOC
-------------------- --------------- ---------------
Firstly              Michaels        Chicago
Secondly             Martin          Chicago
Thirdly              Michaels        Chicago

========
Using the ANSI join notation, this could be written as

CUST_LNAME           EMPL_LAST_NAME  DEPT_LOC
-------------------- --------------- ---------------
Firstly              Michaels        Chicago
Secondly             Martin          Chicago
Thirdly              Michaels        Chicago

SQL>
```


dl313@nrs-projects:~/f24-325
dl313@nrs-projects:~/f24-325l
prompt ========
prompt what if I would like to project, for each customer,
prompt the customer's last name, the name of that customer's employee rep,
prompt and the department location of that employee
select cust_lname, empl_last_name, dept_loc
from empl, customer, dept
where customer.empl_rep = empl.empl_num
and empl.dept_num = dept.dept_num;
prompt ========
prompt Using the ANSI join notation, this could be written as
select cust_lname, empl_last_name, dept_loc
from empl
join customer on empl.empl_num = customer.empl_rep
join dept on empl.dept_num = dept.dept_num;
prompt ========
prompt what if, for example, I want to project the above
prompt only for customers represented by employee Michaels?
select cust_lname, empl_last_name, dept_loc
from empl, customer, dept
where customer.empl_rep = empl.empl_num
and empl.dept_num = dept.dept_num
and empl_last_name = 'Michaels';
~
~
~
~
"325lect06-1.sql" 27L, 922C written
27,39
All


dl313@nrs-projects:~/f24-325l
dl313@nrs-projects:~/f24-325
SQL> @ 325lect06-1.sql

dl313@nrs-projects:~/f24-325l

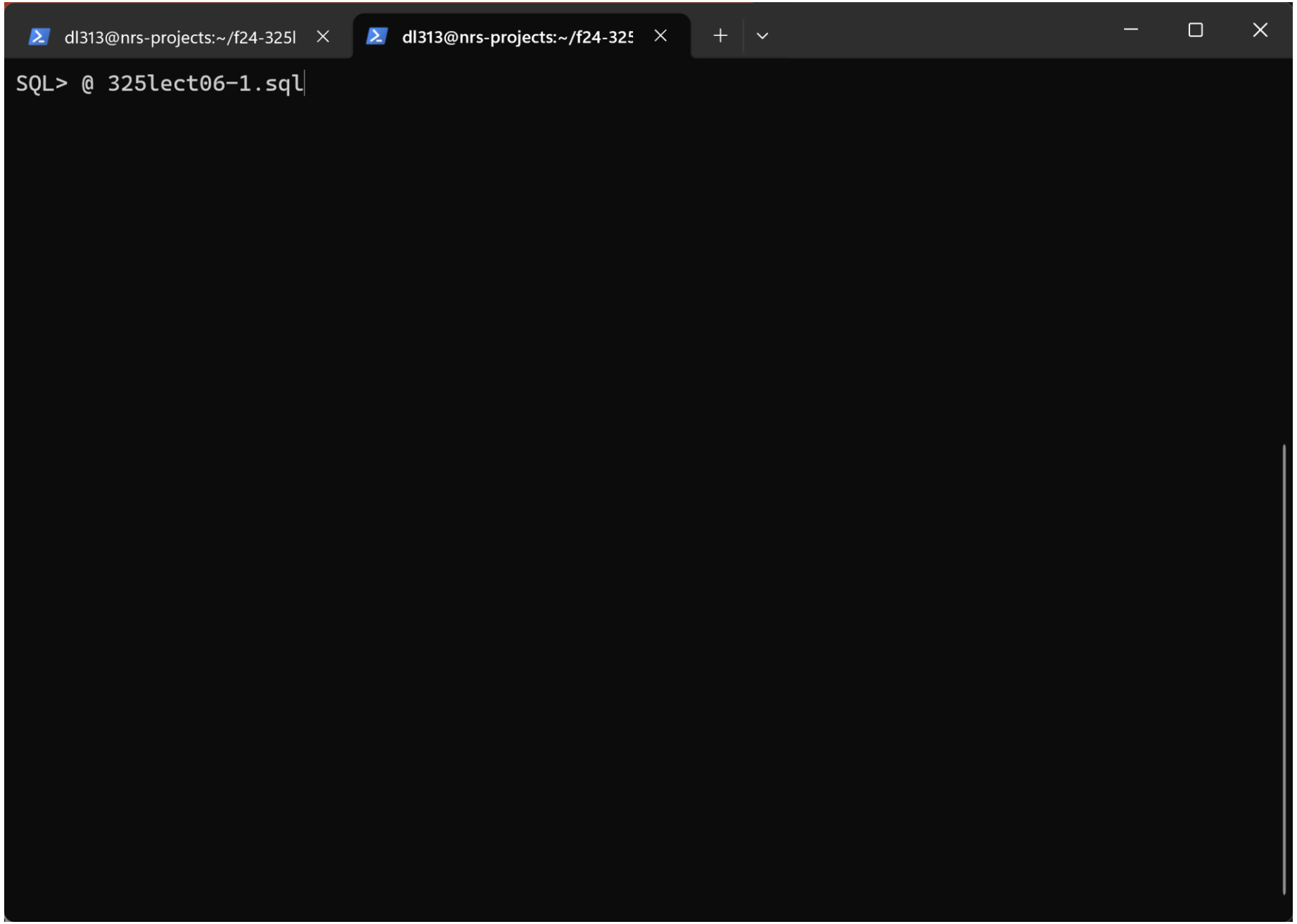
dl313@nrs-projects:~/f24-325

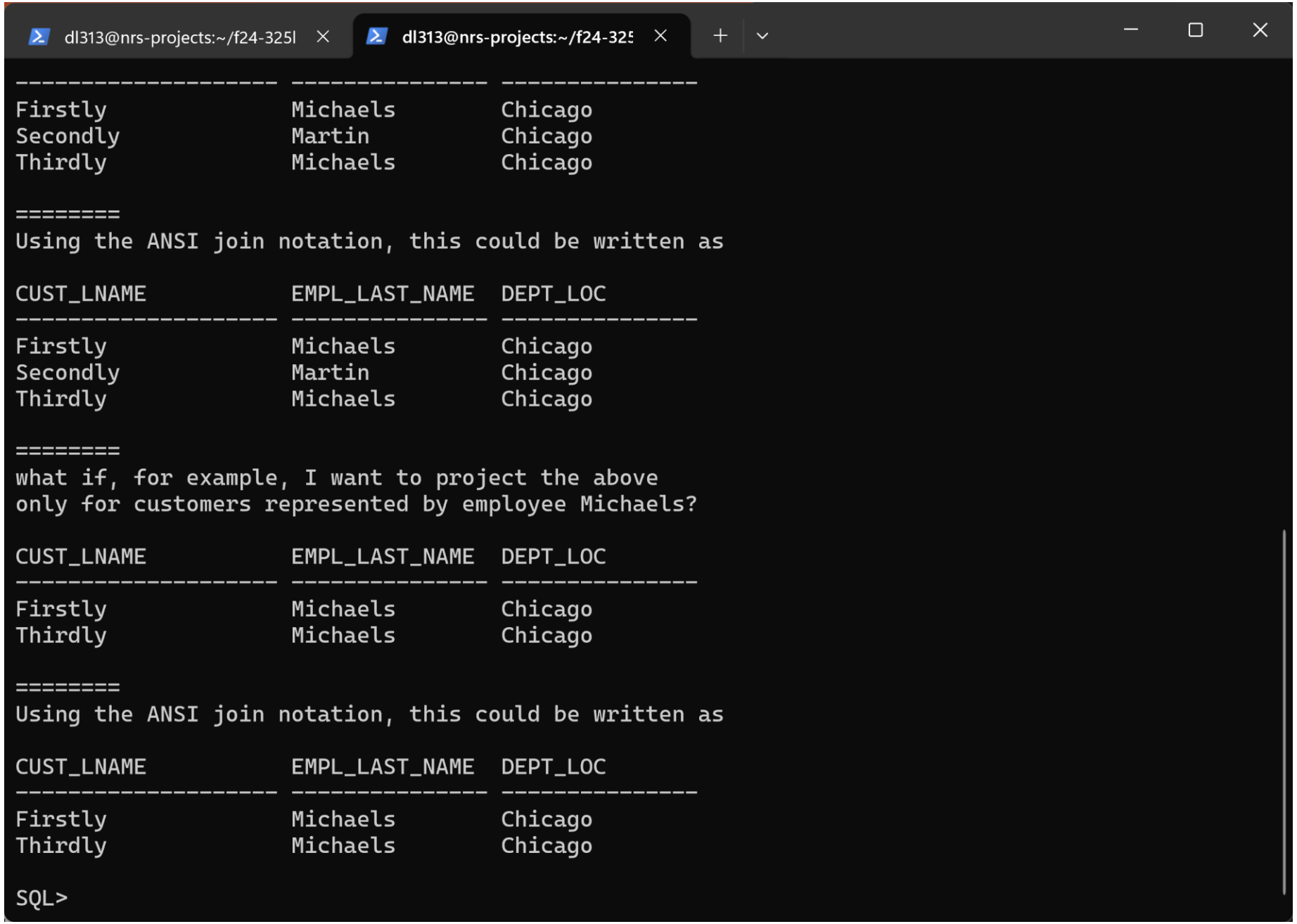
```
SQL> @ 325lect06-1.sql
========
what if I would like to project, for each customer,
the customer's last name, the name of that customer's employee rep,
and the department location of that employee

CUST_LNAME           EMPL_LAST_NAME  DEPT_LOC
-------------------- --------------- ---------------
Firstly              Michaels        Chicago
Secondly             Martin          Chicago
Thirdly              Michaels        Chicago

========
Using the ANSI join notation, this could be written as

CUST_LNAME           EMPL_LAST_NAME  DEPT_LOC
-------------------- --------------- ---------------
Firstly              Michaels        Chicago
Secondly             Martin          Chicago
Thirdly              Michaels        Chicago

========
what if, for example, I want to project the above
only for customers represented by employee Michaels?

CUST_LNAME           EMPL_LAST_NAME  DEPT_LOC
-------------------- --------------- ---------------
Firstly              Michaels        Chicago
Thirdly              Michaels        Chicago

SQL>
```

dl313@nrs-projects:~/f24-325 | dl313@nrs-projects:~/f24-325l

```
prompt ========
prompt what if, for example, I want to project the above
prompt only for customers represented by employee Michaels?

select cust_lname, empl_last_name, dept_loc
from   empl, customer, dept
where  customer.empl_rep = empl.empl_num
       and empl.dept_num = dept.dept_num
       and empl_last_name = 'Michaels';

prompt ========
prompt Using the ANSI join notation, this could be written as

select cust_lname, empl_last_name, dept_loc
from empl
         join customer on empl.empl_num = customer.empl_rep
         join dept on empl.dept_num = dept.dept_num
where empl_last_name = 'Michaels';
```

"325lect06-1.sql" 48L, 1215C written 48,0-1 Bot


dl313@nrs-projects:~/f24-325l
dl313@nrs-projects:~/f24-325
SQL> @ 325lect06-1.sql


dl313@nrs-projects:~/f24-325l
dl313@nrs-projects:~/f24-325
---------- ---------- ----------
Firstly Michaels Chicago
Secondly Martin Chicago
Thirdly Michaels Chicago
========
Using the ANSI join notation, this could be written as
CUST_LNAME EMPL_LAST_NAME DEPT_LOC
---------- ---------- ----------
Firstly Michaels Chicago
Secondly Martin Chicago
Thirdly Michaels Chicago
========
what if, for example, I want to project the above
only for customers represented by employee Michaels?
CUST_LNAME EMPL_LAST_NAME DEPT_LOC
---------- ---------- ----------
Firstly Michaels Chicago
Thirdly Michaels Chicago
========
Using the ANSI join notation, this could be written as
CUST_LNAME EMPL_LAST_NAME DEPT_LOC
---------- ---------- ----------
Firstly Michaels Chicago
Thirdly Michaels Chicago
SQL>

Question

How many join conditions are required when joining three tables in SQL?

A) 1

B) 2

C) 3

D) 4

dl313 — dl313@nrs-projects:~/f24-325lect06-1 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
/*======
     AGGREGATE functions

     ...these give you ONE computation for ALL of the selected
     rows!

     (until we add another SELECT clause later, these result
     in a SINGLE-row result!)

     avg
     min
     max
     sum
     count

     count(col_name) - counts the number of selected rows
        with a NON-NULL value of col_name
     count(*) - counts the number of selecte rows

     YES, these computations tend to SKIP null values...!
=====*/
```

-- INSERT -- 55,5 82%

```
     in a SINGLE-row result!)

     avg
     min
     max
     sum
     count

     count(col_name) - counts the number of selected rows
        with a NON-NULL value of col_name
     count(*) - counts the number of selecte rows

     YES, these computations tend to SKIP null values...!
=====*/

prompt ========
prompt average salary of, and number of rows selected,
prompt with job_title of Manager

select avg(salary), count(*)
from   empl
where  job_title = 'Manager';

-- INSERT --                                          51,11         86%
```

dl313 — dl313@nrs-projects:~/f24-325lect06-1 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

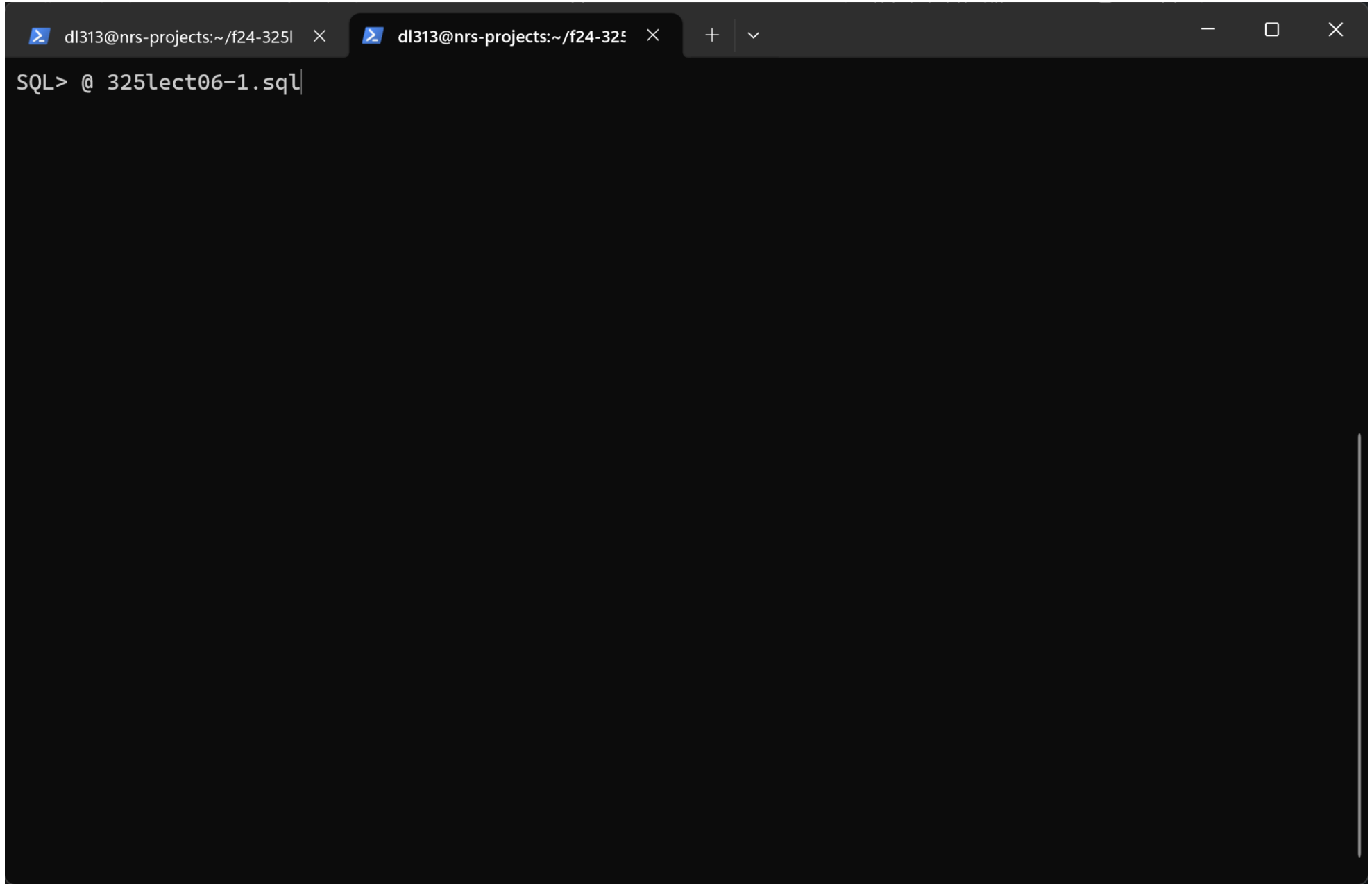
```
SQL> @ 325lect06-1.sql
```

```
only for customers represented by employee Michaels?

CUST_LNAME           EMPL_LAST_NAME  DEPT_LOC
-------------------- --------------- ---------------
Firstly              Michaels        Chicago
Thirdly              Michaels        Chicago

========
Using the ANSI join notation, this could be written as

CUST_LNAME           EMPL_LAST_NAME  DEPT_LOC
-------------------- --------------- ---------------
Firstly              Michaels        Chicago
Thirdly              Michaels        Chicago

========
average salary of, and number of rows selected,
with job_title of Manager

AVG(SALARY)   COUNT(*)
----------- ----------
 2758.33333          3

SQL>
```

dl313@nrs-projects:~/f24-325 | dl313@nrs-projects:~/f24-325l

```
prompt ========
prompt average salary of, and number of rows selected,
prompt with job_title of Manager

select avg(salary), count(*)
from   empl
where  job_title = 'Manager';

prompt ========
prompt average "rounded" salary of, and number of rows selected,
prompt with job_title of Manager

select ROUND(avg(salary), 2) "Average Salary", count(*) "How many rows"
from   empl
where  job_title = 'Manager';
```

"325lect06-1.sql" 102L, 2488C written 89,29 97%


dl313@nrs-projects:~/f24-325l
dl313@nrs-projects:~/f24-325
SQL> @ 325lect06-1.sql

dl313@nrs-projects:~/f24-325l

dl313@nrs-projects:~/f24-325

```
with job_title of Manager

AVG(SALARY)   COUNT(*)
----------- ----------
 2758.33333          3

========
note difference between count(*) and count(commission),
and again note that computations are NOT done with NULL values:

  COUNT(*) COUNT(COMMISSION) SUM(COMMISSION)
---------- ---------------- ---------------
        14                4            2200

========
average salary of, and number of rows selected,
with job_title of Manager

AVG(SALARY)   COUNT(*)
----------- ----------
 2758.33333          3

========
average "rounded" salary of, and number of rows selected,
with job_title of Manager

Average Salary How many rows
-------------- -------------
       2758.33             3

SQL>
```

dl313@nrs-projects:~/f24-325 | dl313@nrs-projects:~/f24-325l

```
prompt average salary of, and number of rows selected,
prompt with job_title of Manager

select avg(salary), count(*)
from   empl
where  job_title = 'Manager';

prompt ========
prompt average "rounded" salary of, and number of rows selected,
prompt with job_title of Manager

select ROUND(avg(salary), 2) "Average Salary", count(*) "How many rows"
from   empl
where  job_title = 'Manager';

prompt ========
prompt average "rounded" salary of, and number of rows selected,
prompt with job_title of Manager to the whole number

select ROUND(avg(salary)) "Average Salary", count(*) "How many rows"
from   empl
where  job_title = 'Manager';
```

"325lect06-1.sql" 110L, 2735C written 97,29 90%

dl313@nrs-projects:~/f24-325l

dl313@nrs-projects:~/f24-325

```
and again note that computations are NOT done with NULL values:

  COUNT(*) COUNT(COMMISSION) SUM(COMMISSION)
---------- ---------------- ---------------
        14                4            2200

========
average salary of, and number of rows selected,
with job_title of Manager

AVG(SALARY)   COUNT(*)
----------- ----------
 2758.33333          3

========
average "rounded" salary of, and number of rows selected,
with job_title of Manager

Average Salary How many rows
-------------- -------------
       2758.33             3

========
average "rounded" salary of, and number of rows selected,
with job_title of Manager to the whole number

Average Salary How many rows
-------------- -------------
          2758             3

SQL>
```

dl313@nrs-projects:~/f24-325 | dl313@nrs-projects:~/f24-325l

```
prompt ========
prompt average "rounded" salary of, and number of rows selected,
prompt with job_title of Manager

select ROUND(avg(salary), 2) "Average Salary", count(*) "How many rows"
from   empl
where  job_title = 'Manager';

prompt ========
prompt average "rounded" salary of, and number of rows selected,
prompt with job_title of Manager to the whole number

select ROUND(avg(salary)) "Average Salary", count(*) "How many rows"
from   empl
where  job_title = 'Manager';

prompt ========
prompt round to the left of the decimal point

select ROUND(avg(salary), -2) "Average Salary", count(*) "How many rows"
from   empl
where  job_title = 'Manager';
```

"325lect06-1.sql" 117L, 2914C written 104,29 91%

dl313@nrs-projects:~/f24-325l

dl313@nrs-projects:~/f24-325

```
average salary of, and number of rows selected,
with job_title of Manager

AVG(SALARY)   COUNT(*)
----------- ----------
 2758.33333          3

========
average "rounded" salary of, and number of rows selected,
with job_title of Manager

Average Salary How many rows
-------------- -------------
       2758.33             3

========
average "rounded" salary of, and number of rows selected,
with job_title of Manager to the whole number

Average Salary How many rows
-------------- -------------
          2758             3

========
round to the left of the decimal point

Average Salary How many rows
-------------- -------------
          2800             3

SQL>
```

dl313@nrs-projects:~/f24-325 | dl313@nrs-projects:~/f24-325l

```
prompt average "rounded" salary of, and number of rows selected,
prompt with job_title of Manager to the whole number

select ROUND(avg(salary)) "Average Salary", count(*) "How many rows"
from   empl
where  job_title = 'Manager';

prompt ========
prompt round to the left of the decimal point

select ROUND(avg(salary), -2) "Average Salary", count(*) "How many rows"
from   empl
where  job_title = 'Manager';

prompt ========
prompt add a Min Hiredate

select ROUND(avg(salary), -2) "Average Salary",
       min(hiredate) "Min Hiredate",
       count(*) "How many rows"
from   empl
where  job_title = 'Manager';
```

"325lect06-1.sql" 126L, 3117C written 113,29 91%

```
dl313@nrs-projects:~/f24-325l   dl313@nrs-projects:~/f24-325

========
average "rounded" salary of, and number of rows selected,
with job_title of Manager

Average Salary How many rows
-------------- -------------
       2758.33             3

========
average "rounded" salary of, and number of rows selected,
with job_title of Manager to the whole number

Average Salary How many rows
-------------- -------------
          2758             3

========
round to the left of the decimal point

Average Salary How many rows
-------------- -------------
          2800             3

========
add a Min Hiredate

Average Salary Min Hired How many rows
-------------- --------- -------------
          2800 02-APR-12             3

SQL>
```

dl313@nrs-projects:~/f24-325 | dl313@nrs-projects:~/f24-325l

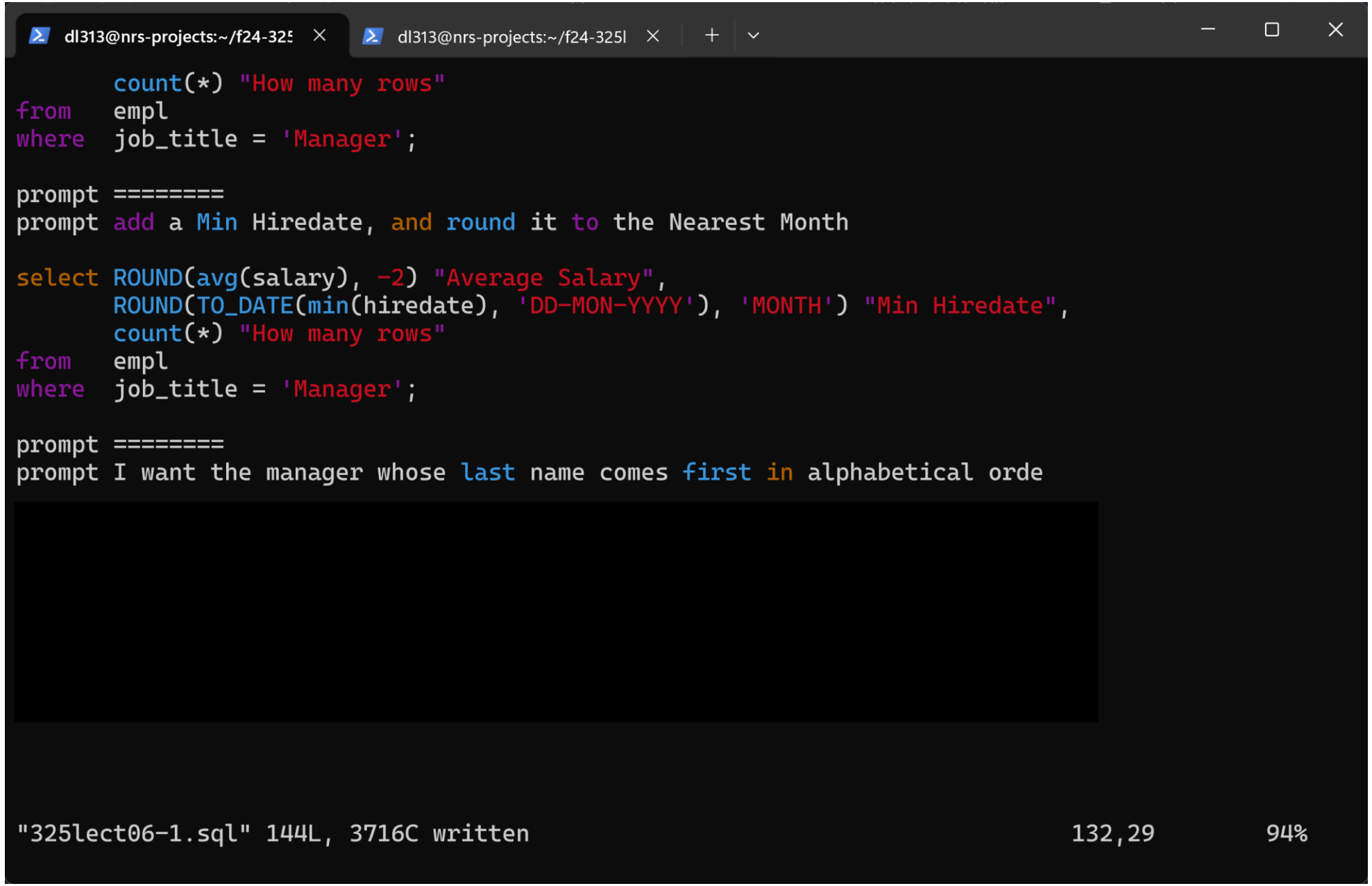
```sql
select ROUND(avg(salary), -2) "Average Salary", count(*) "How many rows"
from   empl
where  job_title = 'Manager';

prompt ========
prompt add a Min Hiredate

select ROUND(avg(salary), -2) "Average Salary",
       min(hiredate) "Min Hiredate",
       count(*) "How many rows"
from   empl
where  job_title = 'Manager';

prompt ========
prompt add a Min Hiredate, and round it to the Nearest Month

select ROUND(avg(salary), -2) "Average Salary",
       ROUND(TO_DATE(min(hiredate), 'DD-MON-YYYY'), 'MONTH') "Min Hiredate",
       count(*) "How many rows"
from   empl
where  job_title = 'Manager';
```

"325lect06-1.sql" 135L, 3397C written 122,29 92%

dl313@nrs-projects:~/f24-325l

dl313@nrs-projects:~/f24-325

```
========
average "rounded" salary of, and number of rows selected,
with job_title of Manager to the whole number

Average Salary How many rows
-------------- -------------
          2758             3

========
round to the left of the decimal point

Average Salary How many rows
-------------- -------------
          2800             3

========
add a Min Hiredate

Average Salary Min Hired How many rows
-------------- --------- -------------
          2800 02-APR-12             3

========
add a Min Hiredate, and round it to the Nearest Month

Average Salary Min Hired How many rows
-------------- --------- -------------
          2800 01-APR-12             3

SQL>
```


dl313@nrs-projects:~/f24-325
dl313@nrs-projects:~/f24-325l
count(*) "How many rows"
from empl
where job_title = 'Manager';
prompt ========
prompt add a Min Hiredate, and round it to the Nearest Month
select ROUND(avg(salary), -2) "Average Salary",
ROUND(TO_DATE(min(hiredate), 'DD-MON-YYYY'), 'MONTH') "Min Hiredate",
count(*) "How many rows"
from empl
where job_title = 'Manager';
prompt ========
prompt I want the manager whose last name comes first in alphabetical orde
"325lect06-1.sql" 144L, 3716C written
132,29
94%

dl313@nrs-projects:~/f24-325 | dl313@nrs-projects:~/f24-325l

```
       count(*) "How many rows"
from   empl
where  job_title = 'Manager';

prompt ========
prompt add a Min Hiredate, and round it to the Nearest Month

select ROUND(avg(salary), -2) "Average Salary",
       ROUND(TO_DATE(min(hiredate), 'DD-MON-YYYY'), 'MONTH') "Min Hiredate",
       count(*) "How many rows"
from   empl
where  job_title = 'Manager';

prompt ========
prompt I want the manager whose last name comes first in alphabetical orde

select ROUND(avg(salary), -2) "Average Salary",
       ROUND(TO_DATE(min(hiredate), 'DD-MON-YYYY'), 'MONTH') "Min Hiredate",
       count(*) "How many rows",
       min(empl_last_name)
from   empl
where  job_title = 'Manager';
```

"325lect06-1.sql" 144L, 3716C written 132,29 94%

```
dl313@nrs-projects:~/f24-325l   dl313@nrs-projects:~/f24-325

          2758             3

========
round to the left of the decimal point

Average Salary How many rows
-------------- -------------
          2800             3

========
add a Min Hiredate

Average Salary Min Hired How many rows
-------------- --------- -------------
          2800 02-APR-12             3

========
add a Min Hiredate, and round it to the Nearest Month

Average Salary Min Hired How many rows
-------------- --------- -------------
          2800 01-APR-12             3

========
I want the manager whose last name comes first in alphabetical orde

Average Salary Min Hired How many rows MIN(EMPL_LAST_N
-------------- --------- ------------- ---------------
          2800 01-APR-12             3 Blake

SQL>
```

dl313@nrs-projects:~/f24-325l

dl313@nrs-projects:~/f24-325

```
Average Salary Min Hired How many rows MIN(EMPL_LAST_N
-------------- --------- ------------- ---------------
          2800 01-APR-12             3 Blake

SQL> select *
  2  from empl;

EMPL EMPL_LAST_NAME  JOB_TITLE  MGR  HIREDATE      SALARY COMMISSION DEP
---- --------------- ---------- ---- --------- ---------- ---------- ---
7839 King            President       17-NOV-11       5000            500
7566 Jones           Manager    7839 02-APR-12       2975            200
7698 Blake           Manager    7839 01-MAY-13       2850            300
7782 Raimi           Manager    7839 09-JUN-12       2450            100
7902 Ford            Analyst    7566 03-DEC-12       3000            200
7369 Smith           Clerk      7902 17-DEC-12        800            200
7499 Michaels        Sales      7698 20-FEB-18       1600        300 300
7521 Ward            Sales      7698 22-FEB-19       1250        500 300
7654 Martin          Sales      7698 28-SEP-18       1250       1400 300
7788 Scott           Analyst    7566 09-NOV-18       3000            200
7844 Turner          Sales      7698 08-SEP-19       1500          0 300

EMPL EMPL_LAST_NAME  JOB_TITLE  MGR  HIREDATE      SALARY COMMISSION DEP
---- --------------- ---------- ---- --------- ---------- ---------- ---
7876 Adams           Clerk      7788 23-SEP-18       1100            400
7900 James           Clerk      7698 03-DEC-17        950            300
7934 Miller          Clerk      7782 23-JAN-16       1300            100

14 rows selected.

SQL>
```

dl313@nrs-projects:~/f24-325 | dl313@nrs-projects:~/f24-325l

```
select ROUND(avg(salary), -2) "Average Salary",
       ROUND(TO_DATE(min(hiredate), 'DD-MON-YYYY'), 'MONTH') "Min Hiredate",
       count(*) "How many rows"
from   empl
where  job_title = 'Manager';

prompt ========
prompt I want the manager whose last name comes first in alphabetical orde

select ROUND(avg(salary), -2) "Average Salary",
       ROUND(TO_DATE(min(hiredate), 'DD-MON-YYYY'), 'MONTH') "Min Hiredate",
       count(*) "How many rows",
       min(empl_last_name)
from   empl
where  job_title = 'Manager';

prompt ========
prompt I want to know the number of unique job titles in the empl table.

select COUNT(DISTINCT job_title) "Unique Job Titles"
from   empl;
```

"325lect06-1.sql" 150L, 3873C written 138,12 94%

```
          2800              3

========
add a Min Hiredate

Average Salary Min Hired How many rows
-------------- --------- -------------
          2800 02-APR-12              3

========
add a Min Hiredate, and round it to the Nearest Month

Average Salary Min Hired How many rows
-------------- --------- -------------
          2800 01-APR-12              3

========
I want the manager whose last name comes first in alphabetical orde

Average Salary Min Hired How many rows MIN(EMPL_LAST_N
-------------- --------- ------------- ---------------
          2800 01-APR-12              3 Blake

========
I want to know the number of unique job titles in the empl table.

Unique Job Titles
-----------------
                5

SQL>
```

dl313@nrs-projects:~/f24-325l

dl313@nrs-projects:~/f24-325

```
SQL> select DISTINCT job_title "Unique Job Titles"
  2  from empl;

Unique Job
----------
Sales
President
Manager
Analyst
Clerk

SQL>
```

dl313 — dl313@nrs-projects:~/f24-325lect06-1 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
     count(col_name) - counts the number of selected rows
        with a NON-NULL value of col_name
     count(*) - counts the number of selecte rows

     YES, these computations tend to SKIP null values...!
=====*/

prompt ========
prompt average salary of, and number of rows selected,
prompt with job_title of Manager

select avg(salary), count(*)
from   empl
where  job_title = 'Manager';

prompt ========
prompt note difference between count(*) and count(commission),
prompt and again note that computations are NOT done with NULL values:

select count(*), count(commission), sum(commission)
from   empl;


-- INSERT --                                          59,1          92%
```

dl313 — dl313@nrs-projects:~/f24-325lect06-1 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
SQL> @ 325lect06-1.sql
```

```
Using the ANSI join notation, this could be written as

CUST_LNAME           EMPL_LAST_NAME  DEPT_LOC
-------------------- --------------- ---------------
Firstly              Michaels        Chicago
Thirdly              Michaels        Chicago

========
average salary of, and number of rows selected,
with job_title of Manager

AVG(SALARY)   COUNT(*)
----------- ----------
 2758.33333          3

========
note difference between count(*) and count(commission),
and again note that computations are NOT done with NULL values:

  COUNT(*) COUNT(COMMISSION) SUM(COMMISSION)
---------- ----------------- ---------------
        14                 4            2200

SQL>
```

dl313 — dl313@nrs-projects:~/f24-325lect06-1 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
SQL> select *
  2  from empl;
```

dl313 — dl313@nrs-projects:~/f24-325lect06-1 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
EMPL EMPL_LAST_NAME   JOB_TITLE   MGR  HIREDATE       SALARY COMMISSION DEP
---- ---------------- ---------- ---- --------- ---------- ---------- ---
7839 King             President       17-NOV-11        5000                500
7566 Jones            Manager    7839 02-APR-12        2975                200
7698 Blake            Manager    7839 01-MAY-13        2850                300
7782 Raimi            Manager    7839 09-JUN-12        2450                100
7902 Ford             Analyst    7566 03-DEC-12        3000                200
7369 Smith            Clerk      7902 17-DEC-12         800                200
7499 Michaels         Sales      7698 20-FEB-18        1600         300    300
7521 Ward             Sales      7698 22-FEB-19        1250         500    300
7654 Martin           Sales      7698 28-SEP-18        1250        1400    300
7788 Scott            Analyst    7566 09-NOV-18        3000                200
7844 Turner           Sales      7698 08-SEP-19        1500           0    300

EMPL EMPL_LAST_NAME   JOB_TITLE   MGR  HIREDATE       SALARY COMMISSION DEP
---- ---------------- ---------- ---- --------- ---------- ---------- ---
7876 Adams            Clerk      7788 23-SEP-18        1100                400
7900 James            Clerk      7698 03-DEC-17         950                300
7934 Miller           Clerk      7782 23-JAN-16        1300                100

14 rows selected.

SQL>
```

SQL time!!!

- SUB-selects? Nested selects?
- the idea here: there are certain places WITHIN a select where you can PLACE a sub-select/nested select;

SQL time!!!

- SUB-selects? Nested selects?
- the idea here: there are certain places WITHIN a select where you can PLACE a sub-select/nested select;
- ...you can place a sub-select where its results would be appropriate!
- ONE such place: is within a WHERE clause, when the Boolean expression for that WHERE clause can use the sub-select's results in determining which rows to select!

dl313 — dl313@nrs-projects:~/f24-325lect06-1 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
-- consider:

prompt =====
prompt the salary of the highest-paid clerk:
prompt =====

select max(salary)
from   empl
where  job_title = 'Clerk';
```

"325lect06-1.sql" 171L, 4088C written 151,27 95%

dl313 — dl313@nrs-projects:~/f24-325lect06-1 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
SQL> @ 325lect06-1.sql
```

```
-------------- --------- ------------- ---------------
          2800 01-APR-12             3 Blake

========
I want to know the number of unique job titles in the empl table.

Unique Job Titles
-----------------
                5


Unique Job Titles
-----------------
                6

=====
the salary of the highest-paid clerk:
=====

MAX(SALARY)
-----------
       1300

SQL>
```

dl313 — dl313@nrs-projects:~/f24-325lect06-1 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
prompt =====
prompt the salary of the highest-paid clerk:
prompt =====

select max(salary)
from   empl
where  job_title = 'Clerk';

-- WHAT IF: I actually want the last name(s) and salary(ies) of
--     the Clerk who has the highest salary?

prompt =====
prompt the does not work:
prompt =====

select empl_last_name, max(salary)
from   empl
where  job_title = 'Clerk';
```

-- INSERT -- 162,28 90%

dl313 — dl313@nrs-projects:~/f24-325lect06-1 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
SQL> @ 325lect06-1.sql
```

```
Unique Job Titles
-----------------
                6

=====
the salary of the highest-paid clerk:
=====

MAX(SALARY)
-----------
       1300

=====
the does not work:
=====
select empl_last_name, max(salary)
       *
ERROR at line 1:
ORA-00937: not a single-group group function


SQL>
```

dl313 — dl313@nrs-projects:~/f24-325lect06-1 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
-- WHAT IF: I actually want the last name(s) and salary(ies) of
--      the Clerk who has the highest salary?

prompt =====
prompt the does not work:
prompt =====

select empl_last_name, max(salary)
from   empl
where  job_title = 'Clerk';

prompt =====
prompt this also does not work:
prompt =====

select empl_last_name, salary
from   empl
where  job_title = 'Clerk'
       and salary = max(salary);
```

"325lect06-1.sql" 193L, 4491C written 159,0-1 88%

dl313 — dl313@nrs-projects:~/f24-325lect06-1 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
SQL> @ 325lect06-1.sql
```

```
MAX(SALARY)
-----------
       1300

=====
the does not work:
=====
select empl_last_name, max(salary)
       *
ERROR at line 1:
ORA-00937: not a single-group group function


=====
this also does not work:
=====
       and salary = max(salary)
                    *
ERROR at line 4:
ORA-00934: group function is not allowed here


SQL>
```

```
prompt =====
prompt this also does not work:
prompt =====

select empl_last_name, salary
from   empl
where  job_title = 'Clerk'
       and salary = max(salary);

-- I can ask for rows where job_title is 'Clerk'
--     and salary is the result of a sub-query,
--     asking for the maximum salary of a clerk:

prompt =====
prompt the name and salary of the highest-paid clerk:
prompt =====

select empl_last_name, salary
from   empl
where  job_title = 'Clerk'
       and salary = (select max(salary)
                     from   empl
                     where  job_title = 'Clerk');
-- INSERT --                                              179,13         89%
```

dl313 — dl313@nrs-projects:~/f24-325lect06-1 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
SQL> @ 325lect06-1.sql
```

dl313 — dl313@nrs-projects:~/f24-325lect06-1 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
select empl_last_name, max(salary)
       *
ERROR at line 1:
ORA-00937: not a single-group group function


=====
this also does not work:
=====
       and salary = max(salary)
                    *
ERROR at line 4:
ORA-00934: group function is not allowed here


=====
the name and salary of the highest-paid clerk:
=====

EMPL_LAST_NAME      SALARY
--------------- ----------
Miller                1300

SQL>
```

dl313 — dl313@nrs-projects:~/f24-325lect06-1 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
select empl_last_name, salary
from   empl
where  job_title = 'Clerk'
       and salary = (select max(salary)
                     from   empl
                     where  job_title = 'Clerk');

-- If I'd like that clerk's manager?

prompt =====
prompt the mgr number of the highest-paid clerk:
prompt =====

select mgr
from   empl
where  job_title = 'Clerk'
       and salary = (select max(salary)
                     from   empl
                     where  job_title = 'Clerk');
```

-- INSERT -- 198,33 92%

dl313 — dl313@nrs-projects:~/f24-325lect06-1 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
SQL> @ 325lect06-1.sql
```

```
=====
       and salary = max(salary)
                    *
ERROR at line 4:
ORA-00934: group function is not allowed here


=====
the name and salary of the highest-paid clerk:
=====

EMPL_LAST_NAME      SALARY
--------------- ----------
Miller                1300

=====
the mgr number of the highest-paid clerk:
=====

MGR
----
7782

SQL>
```

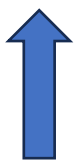
dl313 — dl313@nrs-projects:~/f24-325lect06-1 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
select mgr
from   empl
where  job_title = 'Clerk'
       and salary = (select max(salary)
                     from   empl
                     where  job_title = 'Clerk');

-- Oh, I'd like the NAME of that manager? of the highest-paid clerk?
-- I can use another sub-select:

prompt =====
prompt the last name and mgr number of the highest-paid clerk:
prompt =====

select empl_last_name, empl_num
from   empl
where  empl_num IN (select mgr
                    from   empl
                    where  job_title = 'Clerk'
                           and salary = (select max(salary)
                                         from   empl
                                         where  job_title = 'Clerk'));
```

-- INSERT -- 199,50 92%

dl313 — dl313@nrs-projects:~/f24-325lect06-1 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
select mgr
from   empl
where  job_title = 'Clerk'
       and salary = (select max(salary)
                     from   empl
                     where  job_title = 'Clerk');

-- Oh, I'd like the NAME of that manager? of the highest-paid clerk?
-- I can use another sub-select:

prompt =====
prompt the last name and mgr number of the highest-paid clerk:
prompt =====

select empl_last_name, empl_num
from   empl
where  empl_num IN (select mgr
                    from   empl
                    where  job_title = 'Clerk'
                           and salary = (select max(salary)
                                         from   empl
                                         where  job_title = 'Clerk'));
```



-- INSERT -- 199,50 92%

dl313 — dl313@nrs-projects:~/f24-325lect06-1 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
SQL> @ 325lect06-1.sql
```

dl313 — dl313@nrs-projects:~/f24-325lect06-1 — ssh dl313@nrs-projects-ssh.humboldt.edu — 80×24

```
the name and salary of the highest-paid clerk:
=====

EMPL_LAST_NAME      SALARY
--------------- ----------
Miller                1300

=====
the mgr number of the highest-paid clerk:
=====

MGR
----
7782

=====
the last name and mgr number of the highest-paid clerk:
=====

EMPL_LAST_NAME  EMPL
--------------- ----
Raimi           7782

SQL>
```