Lab Week 1 Continued Section 2 SQL Operators, Restricting and Sorting Data, and Section 3 Retrieving Data from Multiple Tables

Section 2 SQL Operators, Restricting and Sorting Data

- In this section
  - Character strings
  - Comparison operators
  - Rules of precedence
  - Sorting and ordering

Character Strings and Dates

- Character strings and date values are enclosed in single quotation marks.
- Character values are case sensitive and date values are format sensitive.
- The default date format is DD-MON-YY.

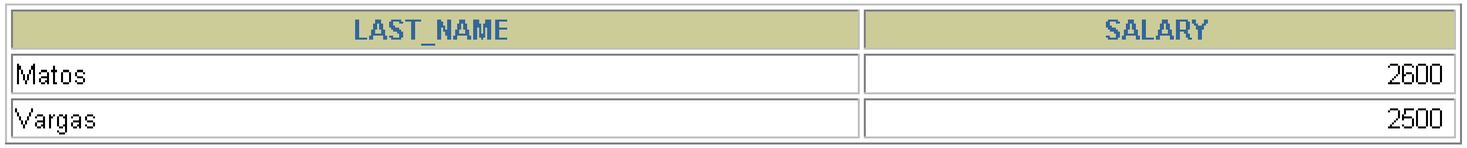
```
SELECT last_name, job_id, department_id
FROM   employees
WHERE  last_name = 'Whalen' ;
```

Comparison Operators

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<> or !=	Not equal to

Xvlqj#wkh#Frpsdulvrq#Rshudwruv

```
SELECT last_name, salary
FROM   employees
WHERE  salary <= 3000 ;
```



LAST_NAME	SALARY
Matos	2600
Vargas	2500

R wkhu#Frp sdulvrq#R shudwruv

Operator	Meaning
BETWEEN ...AND...	Between two values (inclusive)
IN(list)	Match any of a list of values
LIKE	Match a character pattern
IS NULL	Is a null value

Using the BETWEEN Operator

- Use the BETWEEN operator to display rows based on a range of values.

```
SELECT last_name, salary
FROM   employees
WHERE  salary BETWEEN 2500 AND 3500 ;
```

Lower limit Upper limit

LAST_NAME	SALARY
Rajs	3500
Davies	3100
Matos	2600
Vargas	2500

Using the IN Operator

- Use the IN operator to test for values in a list.

```
SELECT employee_id, last_name, salary, manager_id
FROM   employees
WHERE  manager_id IN (100, 101, 201) ;
```

EMPLOYEE_ID	LAST_NAME	SALARY	MANAGER_ID
202	Fay	6000	201
200	Whalen	4400	101
205	Higgins	12000	101
101	Kochhar	17000	100
102	De Haan	17000	100
124	Mourgos	5800	100
149	Zlotkey	10500	100
201	Hartstein	13000	100

8 rows selected.

Using the LIKE Operator

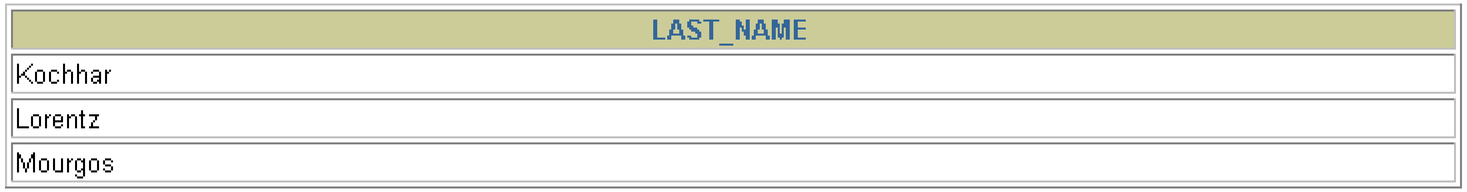
- Use the LIKE operator to perform wildcard searches of valid search string values.
- Search conditions can contain either literal characters or numbers.
- % denotes zero or many characters.
- _ denotes one character.

```
SELECT   first_name
FROM     employees
WHERE    first_name LIKE 'S%' ;
```

Using the LIKE Operator

– You can combine pattern-matching characters:

```
SELECT last_name
FROM   employees
WHERE  last_name LIKE '_o%' ;
```



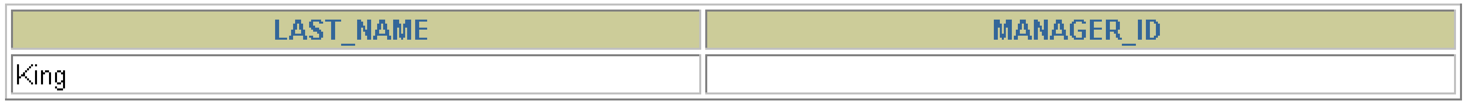
LAST_NAME
Kochhar
Lorentz
Mourgos

– You can use the `ESCAPE` identifier to search for the actual `%` and _ symbols.

Using the IS NULL Operator

- Test for nulls with the `IS NULL` operator.

```
SELECT last_name, manager_id
FROM   employees
WHERE  manager_id IS NULL ;
```



LAST_NAME	MANAGER_ID
King	

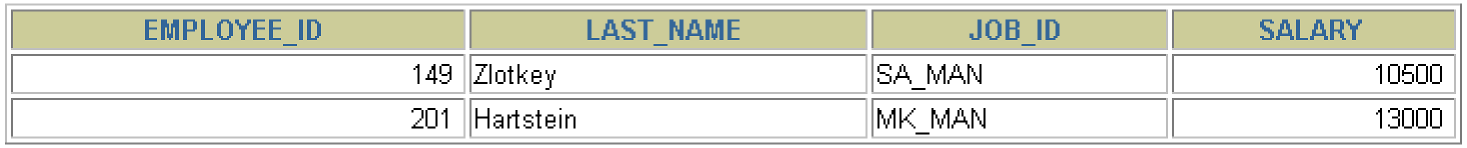
Orjlfdo#Rshudwruv

Operator	Meaning
AND	**Returns TRUE if *both* component conditions are TRUE**
OR	**Returns TRUE if *either* component condition is TRUE**
NOT	**Returns TRUE if the following condition is FALSE**

Using the AND Operator

AND requires both conditions to be true:

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary >=10000
AND    job_id LIKE '%MAN%' ;
```

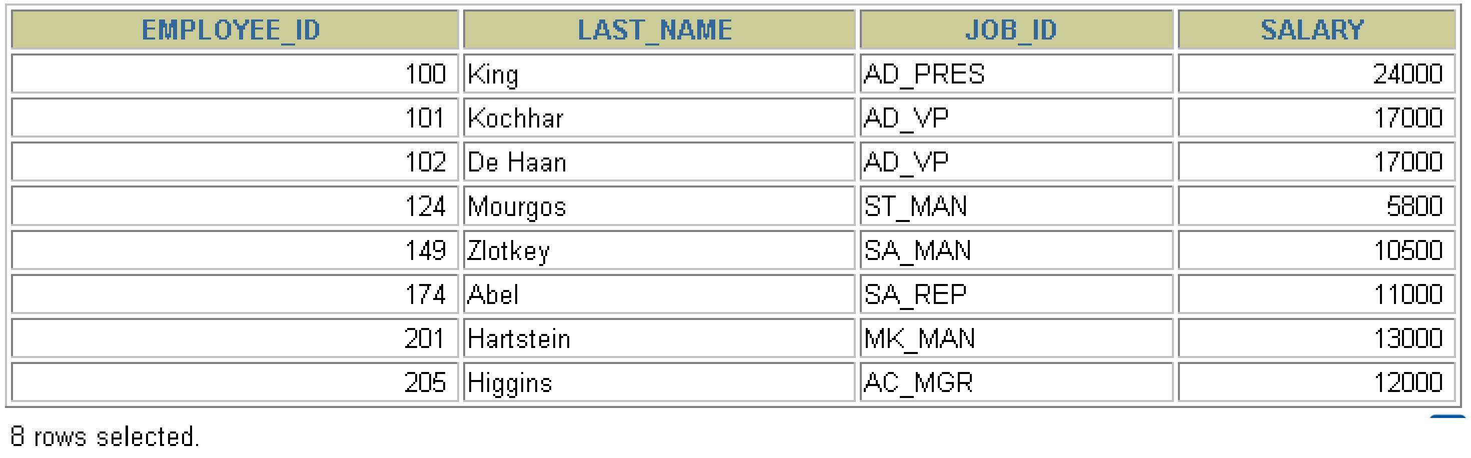


EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
149	Zlotkey	SA_MAN	10500
201	Hartstein	MK_MAN	13000

Using the OR Operator

OR requires either condition to be true:

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary >= 10000
OR     job_id LIKE '%MAN%' ;
```



EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
100	King	AD_PRES	24000
101	Kochhar	AD_VP	17000
102	De Haan	AD_VP	17000
124	Mourgos	ST_MAN	5800
149	Zlotkey	SA_MAN	10500
174	Abel	SA_REP	11000
201	Hartstein	MK_MAN	13000
205	Higgins	AC_MGR	12000

8 rows selected.

Xvlqj#wkh#QRW#Rshudwru

```
SELECT last_name, job_id
FROM   employees
WHERE  job_id
       NOT IN ('IT_PROG', 'ST_CLERK', 'SA_REP') ;
```

LAST_NAME	JOB_ID
King	AD_PRES
Kochhar	AD_VP
De Haan	AD_VP
Mourgos	ST_MAN
Zlotkey	SA_MAN
Whalen	AD_ASST
Hartstein	MK_MAN
Fay	MK_REP
Higgins	AC_MGR
Gietz	AC_ACCOUNT

10 rows selected.

Uxohv#ri#Suhfhghqfh

Order Evaluated	Operator
1	Artimetic operators
2	Concatenation operator
3	Comparison conditions
4	IS [NOT] NULL, LIKE, [NOT] IN
5	[NOT] BETWEEN
6	NOT logical condition
7	AND logical condition
8	OR logical condition

Note: Override rules of precedence by using parentheses.

Uxdhv#ri#Suhfhghqfh

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  job_id = 'SA_REP'
OR     job_id = 'AD_PRES'
AND    salary > 15000;
```

1

LAST_NAME	JOB_ID	SALARY
King	AD_PRES	24000
Abel	SA_REP	11000
Taylor	SA_REP	8600
Grant	SA_REP	7000

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  (job_id = 'SA_REP'
OR     job_id = 'AD_PRES')
AND    salary > 15000;
```

2

LAST_NAME	JOB_ID	SALARY
King	AD_PRES	24000

ORDER BY Clause

– Sort retrieved rows with the `ORDER BY` clause:
  - `ASC`: ascending order, default
  - `DESC`: descending order
– The `ORDER BY` clause comes last in the `SELECT` statement:

```
SELECT    last_name, job_id, department_id, hire_date
FROM      employees
ORDER BY hire_date ;
```

Vruwlqj#lq#Ghvfhqglqj#Rughu

– Sorting in descending order:

```
SELECT   last_name, job_id, department_id, hire_date
FROM     employees
ORDER BY hire_date DESC ;
```

1

– Sorting by column alias:

```
SELECT employee_id, last_name, salary*12 annsal
FROM   employees
ORDER BY annsal ;
```

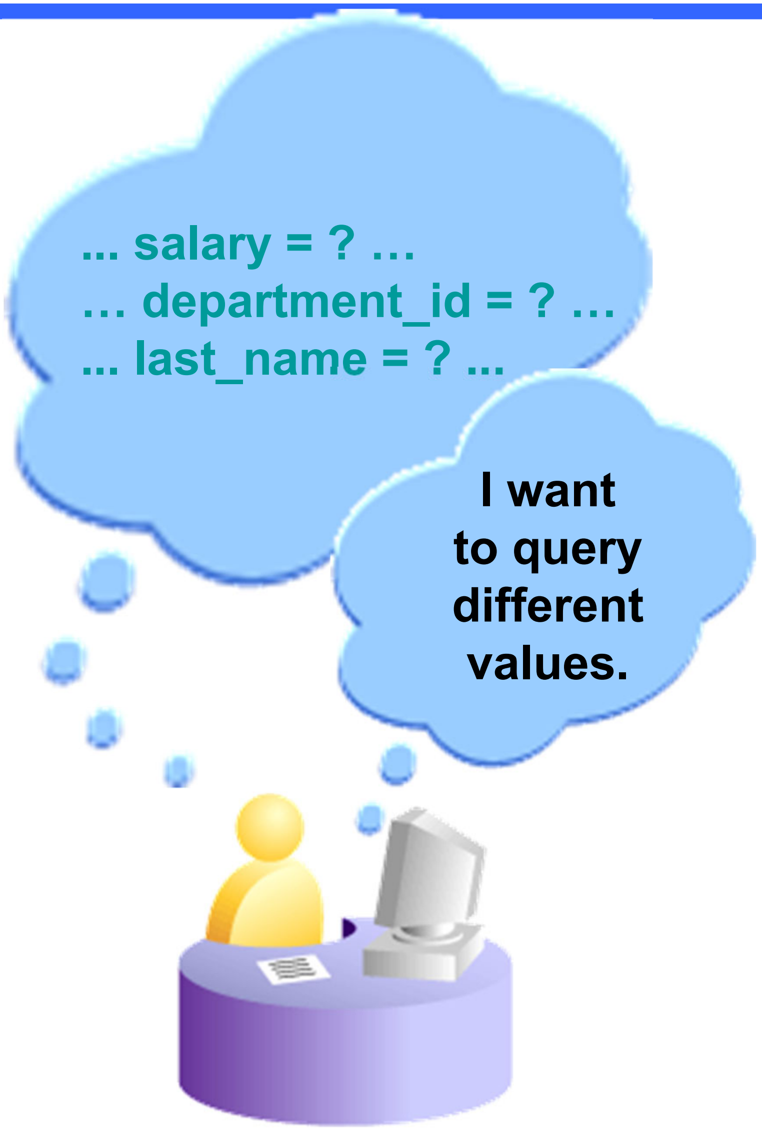
2

– Sorting by multiple columns:

```
SELECT last_name, department_id, salary
FROM   employees
ORDER BY department_id, salary DESC;
```

3

Substitution Variables



Substitution Variables

- Use *i*SQL*Plus substitution variables to:
  - Temporarily store values with single-ampersand (`&`) and double-ampersand (`&&`) substitution
- Use substitution variables to supplement the following:
  - `WHERE` conditions
  - `ORDER BY` clauses
  - Column expressions
  - Table names
  - Entire `SELECT` statements

Using the & Substitution Variable

- Use a variable prefixed with an ampersand (&) to prompt the user for a value:

```
SELECT employee_id, last_name, salary, department_id
FROM   employees
WHERE  employee_id = &employee_num ;
```

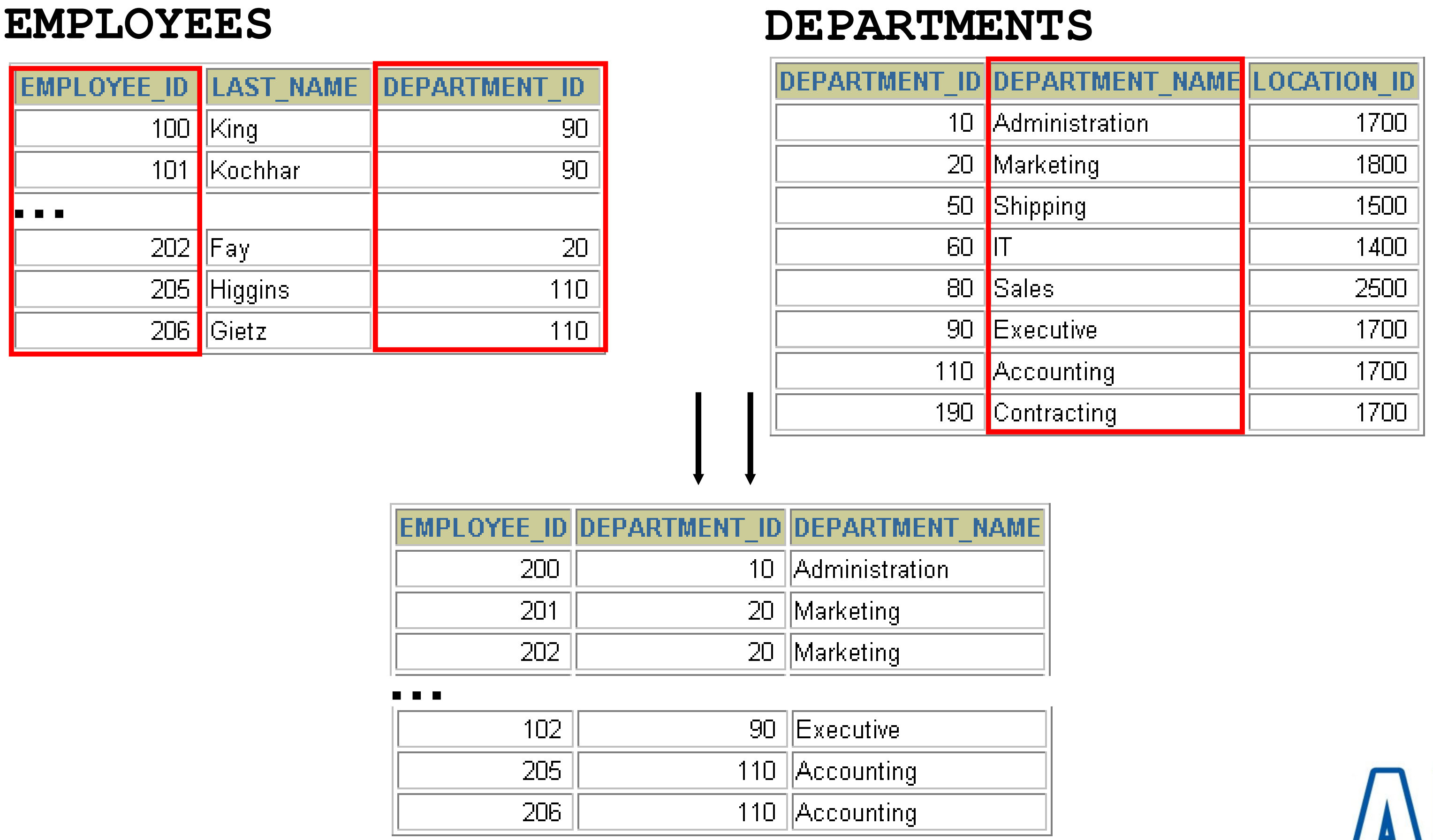
Lab Activities

- Complete Section 2 of SQL lab exercise

Section 3 Retrieving Data from Multiple Tables

- In this section:
  - Using SELECT statements to access data from more than one table using equality and non-equality joins
  - View data that generally does not meet a join condition by using outer joins
  - Join a table to itself

R ewdlqlqj#Gdwd#iurp #P xowlsoh# Wdeohv



EMPLOYEES

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
101	Kochhar	90
...		
202	Fay	20
205	Higgins	110
206	Gietz	110

DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
10	Administration	1700
20	Marketing	1800
50	Shipping	1500
60	IT	1400
80	Sales	2500
90	Executive	1700
110	Accounting	1700
190	Contracting	1700

EMPLOYEE_ID	DEPARTMENT_ID	DEPARTMENT_NAME
200	10	Administration
201	20	Marketing
202	20	Marketing
...		
102	90	Executive
205	110	Accounting
206	110	Accounting

What Is a Join?

- Use a join to query data from more than one table.

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

  – Write the join condition in the WHERE clause.
  – Prefix the column name with the table name when the same column name appears in more than one table.

Cartesian Product

- A Cartesian product is formed when:
  - A join condition is omitted
  - A join condition is invalid
  - All rows in the first table are joined to all rows in the second table
- To avoid a Cartesian product, always include a valid join condition in a WHERE clause.

Jhqhudwlqj#d#Fduwhvldq#Surgxfw

EMPLOYEES (20 rows)

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
101	Kochhar	90
…		
202	Fay	20
205	Higgins	110
206	Gietz	110

20 rows selected.

DEPARTMENTS (8 rows)

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
10	Administration	1700
20	Marketing	1800
50	Shipping	1500
60	IT	1400
80	Sales	2500
90	Executive	1700
110	Accounting	1700
190	Contracting	1700

8 rows selected.

Cartesian product:
20 x 8 = 160 rows

EMPLOYEE_ID	DEPARTMENT_ID	LOCATION_ID
100	90	1700
101	90	1700
102	90	1700
103	60	1700
104	60	1700
107	60	1700
…		

160 rows selected.

Z kdw#lv#dq#HtxlmrlqB

EMPLOYEES

```
 EMPNO ENAME    DEPTNO
------ ------- -------
  7839 KING         10
  7698 BLAKE        30
  7782 CLARK        10
  7566 JONES        20
  7654 MARTIN       30
  7499 ALLEN        30
  7844 TURNER       30
  7900 JAMES        30
  7521 WARD         30
  7902 FORD         20
  7369 SMITH        20
...
14 rows selected.
```

DEPARTMENTS

```
 DEPTNO DNAME      LOC
------- ---------- --------
     10 ACCOUNTING NEW YORK
     30 SALES      CHICAGO
     10 ACCOUNTING NEW YORK
     20 RESEARCH   DALLAS
     30 SALES      CHICAGO
     30 SALES      CHICAGO
     30 SALES      CHICAGO
     30 SALES      CHICAGO
     20 RESEARCH   DALLAS
     20 RESEARCH   DALLAS
...
14 rows selected.
```

Foreign key Primary key

Uhwulhylqj#Uhfrugv#zlwk#Htxlmrlqv

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

```
EMPNO ENAME   DEPTNO DEPTNO LOC
----- ------ ------ ------ ---------
 7839 KING       10     10 NEW YORK
 7698 BLAKE      30     30 CHICAGO
 7782 CLARK      10     10 NEW YORK
 7566 JONES      20     20 DALLAS
...
14 rows selected.
```

Qualifying Ambiguous Column Names

- Use table prefixes to qualify column names that are in multiple tables.
- Improve performance by using table prefixes.
- Distinguish columns that have identical names but reside in different tables by using column aliases.

Using Table Aliases

- Simplify queries by using table aliases.

```
SELECT e.employee_id, e.last_name, d.department_no,
       d.location_id
FROM   employees e, departments d
WHERE  e.department_id=d.department_id;
```

```
SELECT e.empno, e.ename, e.deptno,
       d.deptno, d.loc
FROM   emp e, dept d
WHERE  e.deptno=d.deptno;
```

Dgglwlrqdo#Vhdufk#Frqglwlrqv Xvlqj#wkh#DQG#Rshudwru

EMPLOYEES

```
 EMPNO ENAME      DEPTNO
------ -------- --------
  7839 KING           10
  7698 BLAKE          30
  7782 CLARK          10
  7566 JONES          20
  7654 MARTIN         30
  7499 ALLEN          30
  7844 TURNER         30
  7900 JAMES          30
  7521 WARD           30
  7902 FORD           20
  7369 SMITH          20
...
14 rows selected.
```

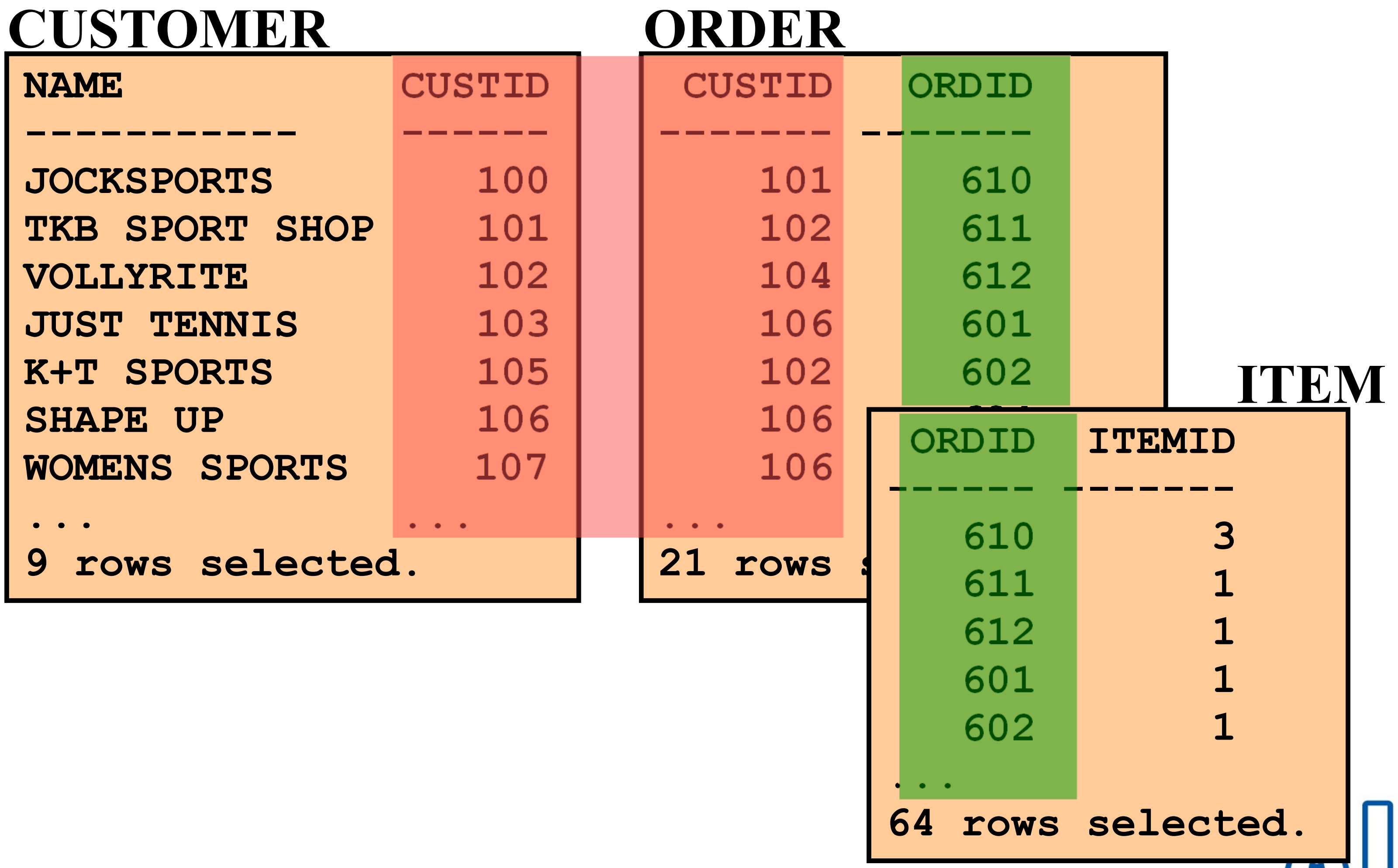
DEPARTMENTS

```
DEPTNO DNAME      LOC
------ ---------- ---------
    10 ACCOUNTING NEW YORK
    30 SALES      CHICAGO
    10 ACCOUNTING NEW YORK
    20 RESEARCH   DALLAS
    30 SALES      CHICAGO
    30 SALES      CHICAGO
    30 SALES      CHICAGO
    30 SALES      CHICAGO
    30 SALES      CHICAGO
    20 RESEARCH   DALLAS
    20 RESEARCH   DALLAS
...
14 rows selected.
```

Additional Search Conditions Using the AND Operator

```
SELECT    e.employee_id, e.last_name,
d.department_no, d.location_id
FROM      EMPLOYEES E, DEPARTMENTS D
WHERE     e.department_id = d.department_id
AND       last_name = 'KING';
```

Mrlqlqj#Pruh#Wkdq#Wzr#Wdeohv



CUSTOMER

```
NAME          CUSTID
------------  ------
JOCKSPORTS       100
TKB SPORT SHOP   101
VOLLYRITE        102
JUST TENNIS      103
K+T SPORTS       105
SHAPE UP         106
WOMENS SPORTS    107
...              ...
9 rows selected.
```

ORDER

```
  CUSTID    ORDID
-------- --------
     101      610
     102      611
     104      612
     106      601
     102      602
     106
     106
...
21 rows s
```

ITEM

```
 ORDID  ITEMID
------ -------
   610       3
   611       1
   612       1
   601       1
   602       1
...
64 rows selected.
```

Joining More Than Two Tables…

```
SELECT   Name, ItemId
FROM Customer C, Ord O, Item I
WHERE    C.custId = O.custId
AND      O.ordId = I.ordId;
```

Q rq0Htxlmrlqv

EMP

```
 EMPNO ENAME         SAL
------ ------- -------
  7839 KING         5000
  7698 BLAKE        2850
  7782 CLARK        2450
  7566 JONES        2975
  7654 MARTIN       1250
  7499 ALLEN        1600
  7844 TURNER       1500
  7900 JAMES         950
...
14 rows selected.
```

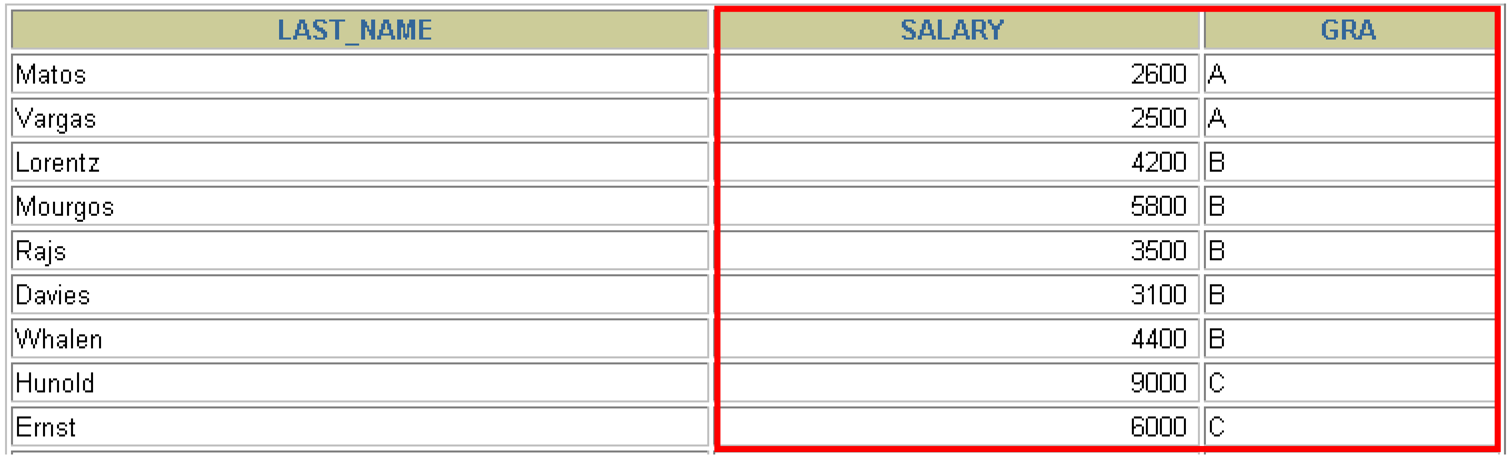
SALGRADE

```
GRADE LOSAL  HISAL
----- ----- ------
1       700   1200
2      1201   1400
3      1401   2000
4      2001   3000
5      3001   9999
```

"salary in the EMP table is between low salary and high salary in the SALGRADE table"

Uhwulhylqj#Uhfrugv#zlwk#Qrq0 Htxlmrlqv

```
SELECT e.last_name, e.salary, j.grade_level
FROM   employees e JOIN job_grades j
ON     e.salary
       BETWEEN j.lowest_sal AND j.highest_sal;
```

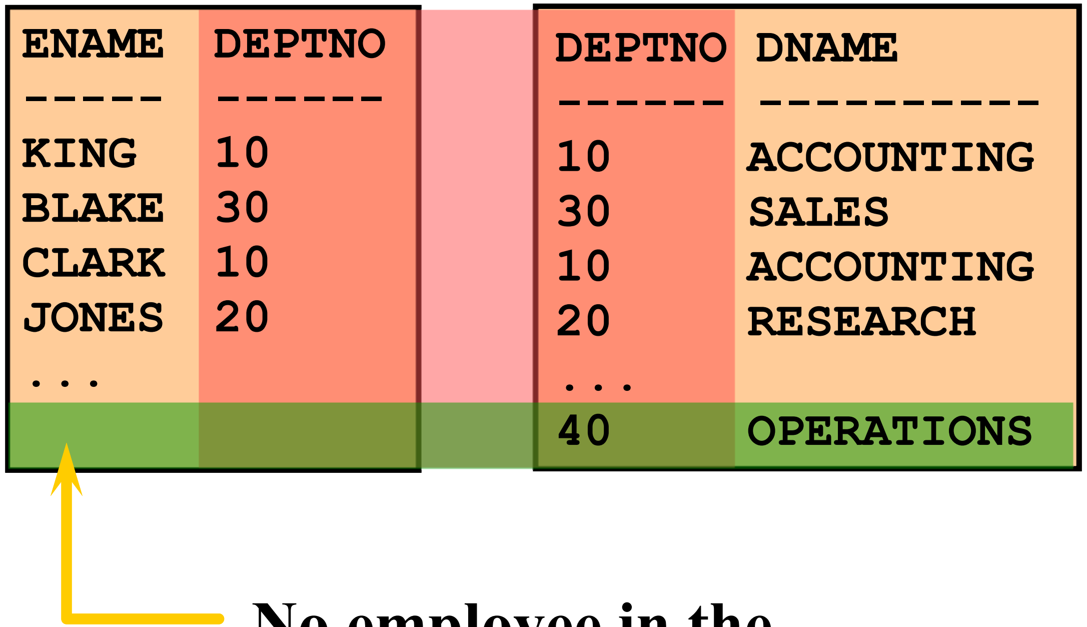


LAST_NAME	SALARY	GRA
Matos	2600	A
Vargas	2500	A
Lorentz	4200	B
Mourgos	5800	B
Rajs	3500	B
Davies	3100	B
Whalen	4400	B
Hunold	9000	C
Ernst	6000	C

...

20 rows selected.

R xwhu#Mrlqv



EMPLOYEES

```
ENAME   DEPTNO
-----   ------
KING    10
BLAKE   30
CLARK   10
JONES   20
...
```

DEPARTMENTS

```
DEPTNO  DNAME
------  ----------
10      ACCOUNTING
30      SALES
10      ACCOUNTING
20      RESEARCH
...
40      OPERATIONS
```

No employee in the OPERATIONS department

Outer Joins

- You use an outer join to also see rows that do not usually meet the join condition.
- Outer join operator is the plus sign (+).

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

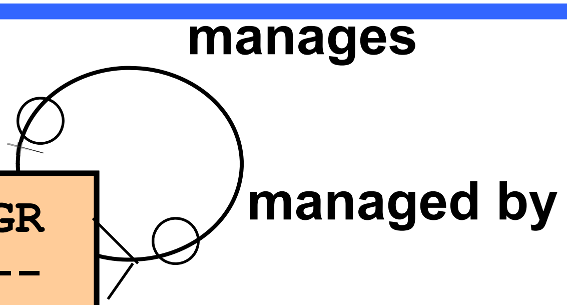
```
SELECT table1.column, table2.column
FROM   table1, table2
WHERE  table1.column = table2.column(+);
```

Xvlqj#Rxwhu#Mrlqv

```
SQL> SELECT e.last_name, d.department_id, d.dname
  2  FROM     employees e, departments d
  3  WHERE    e.department_id(+) = d.department_id
  4  ORDER BY e.department_id;
```

```
ENAME            DEPTNO DNAME
---------- --------- -------------
KING                10 ACCOUNTING
CLARK               10 ACCOUNTING
...
                    40 OPERATIONS
15 rows selected.
```

Self Joins



EMP

EMPNO	ENAME	MGR
7839	KING	
7698	BLAKE	7839
7782	CLARK	7839
7566	JONES	7839
7654	MARTIN	7698
7499	ALLEN	7698

EMP (WORKER)

EMPNO	ENAME	MGR
7839	KING	
7698	BLAKE	7839
7782	CLARK	7839
7566	JONES	7839
7654	MARTIN	7698
7499	ALLEN	7698

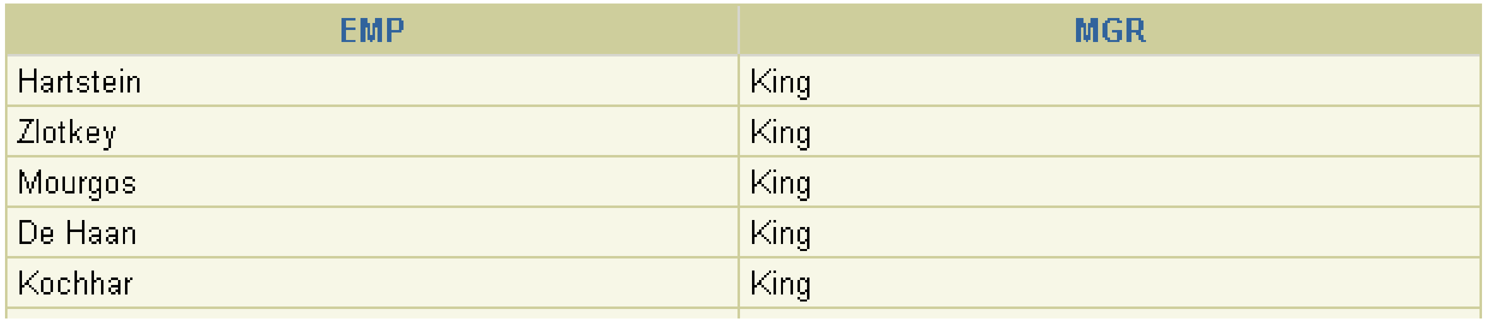
EMP (MANAGER)

EMPNO	ENAME
7839	KING
7839	KING
7839	KING
7698	BLAKE
7698	BLAKE

"MGR in the WORKER table is equal to EMPNO in the MANAGER table"

Joining a Table to Itself (Self Joins)

```
SELECT e.last_name emp, m.last_name mgr
FROM   employees e JOIN employees m
ON     (e.manager_id = m.employee_id);
```



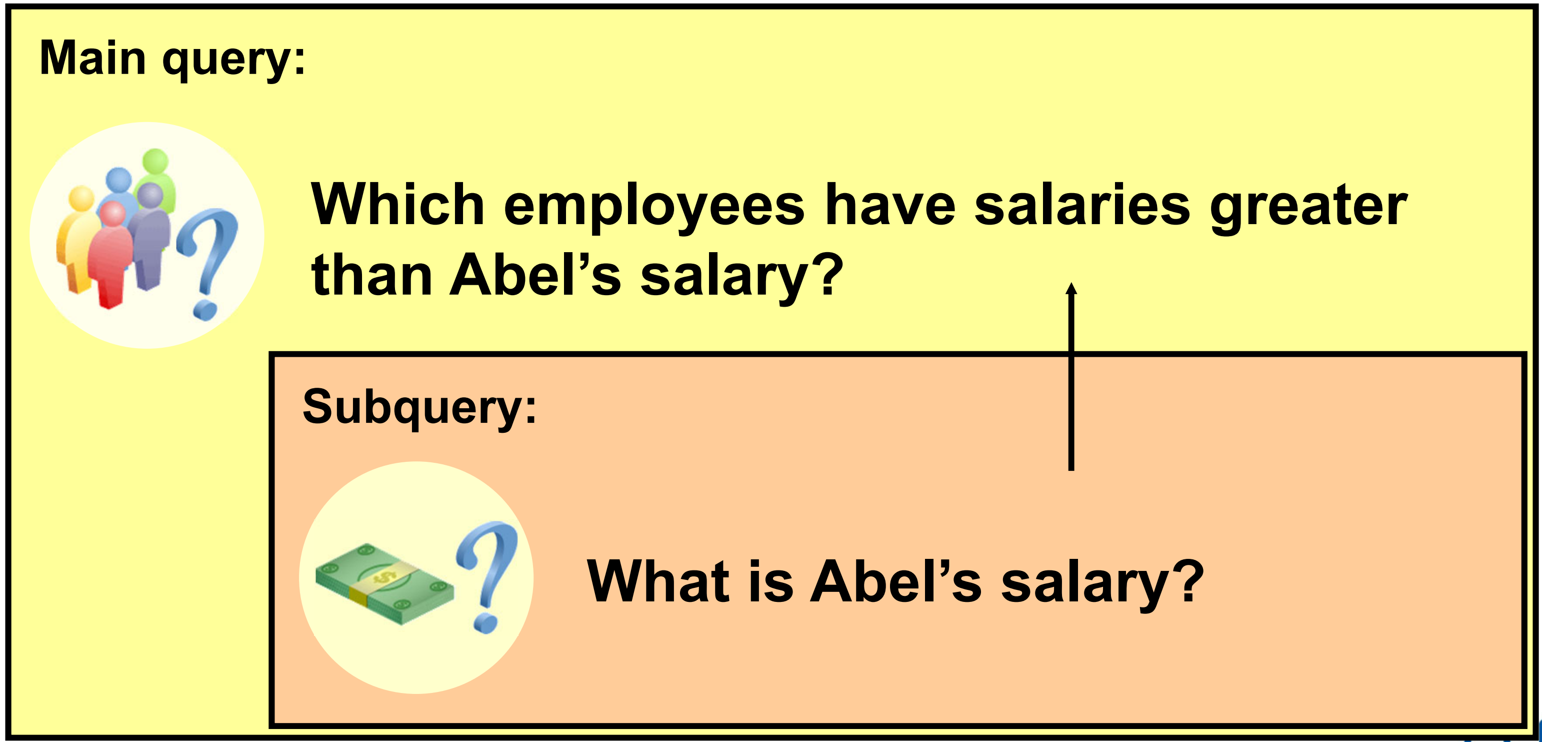
EMP	MGR
Hartstein	King
Zlotkey	King
Mourgos	King
De Haan	King
Kochhar	King

...

19 rows selected.

Using a Subquery to Solve a Problem

- Who has a salary greater than Abel’s?



Main query:

Which employees have salaries greater than Abel’s salary?

Subquery:

What is Abel’s salary?

Subquery Syntax

```
SELECT   select_list
FROM     table
WHERE    expr operator
                    (SELECT     select_list
                     FROM       table);
```

- The subquery (inner query) executes once before the main query (outer query).
- The result of the subquery is used by the main query.

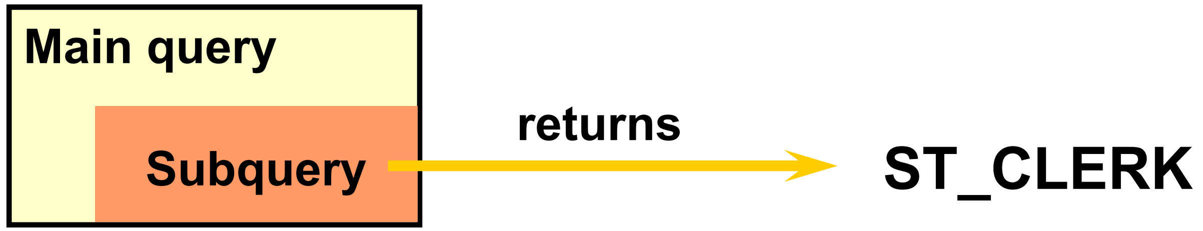
Xvlqj#d#Vxetxhu|

```
SELECT last_name
FROM   employees           11000 <-
WHERE  salary >
                (SELECT salary
                 FROM   employees
                 WHERE  last_name = 'Abel');
```

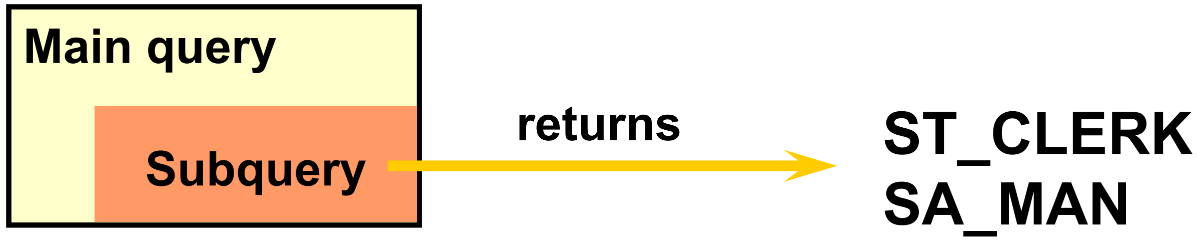
LAST_NAME
King
Kochhar
De Haan
Hartstein
Higgins

Types of Subqueries

- Single-row subquery



- Multiple-row subquery



Executing Single-Row Subqueries

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  job_id =                 ST_CLERK
               (SELECT job_id
                FROM   employees
                WHERE  employee_id = 141)
AND    salary >                   2600
               (SELECT salary
                FROM   employees
                WHERE  employee_id = 143);
```

LAST_NAME	JOB_ID	SALARY
Rajs	ST_CLERK	3500
Davies	ST_CLERK	3100

Multiple-Row Subqueries

- Return more than one row
- Use multiple-row comparison operators

Operator	Meaning
`IN`	Equal to any member in the list
`ANY`	Compare value to each value returned by the subquery
`ALL`	Compare value to every value returned by the subquery

Using the ANY Operator in Multiple-Row Subqueries

```
SELECT employee_id, last_name, job_id, salary
FROM   employees       9000, 6000, 4200
WHERE  salary < ANY
                    (SELECT salary
                     FROM   employees
                     WHERE  job_id = 'IT_PROG')
AND    job_id <> 'IT_PROG';
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
124	Mourgos	ST_MAN	5800
141	Rajs	ST_CLERK	3500
142	Davies	ST_CLERK	3100
143	Matos	ST_CLERK	2600
144	Vargas	ST_CLERK	2500

...

10 rows selected.

Xvlqj#wkh#**ALL** Rshudwru# lq#Pxowlsoh0Urz#Vxetxhulhv

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary < ALL
                    (SELECT salary
                     FROM   employees
                     WHERE  job_id = 'IT_PROG')
AND    job_id <> 'IT_PROG';
```

9000, 6000, 4200

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
141	Rajs	ST_CLERK	3500
142	Davies	ST_CLERK	3100
143	Matos	ST_CLERK	2600
144	Vargas	ST_CLERK	2500

Qxoo#Ydoxhv#lq#d#Vxetxhu|

```
SELECT emp.last_name
FROM   employees emp
WHERE  emp.employee_id NOT IN
                              (SELECT mgr.manager_id
                               FROM   employees mgr);

no rows selected
```

Lab Activities

- Complete Section 3 of SQL lab exercise