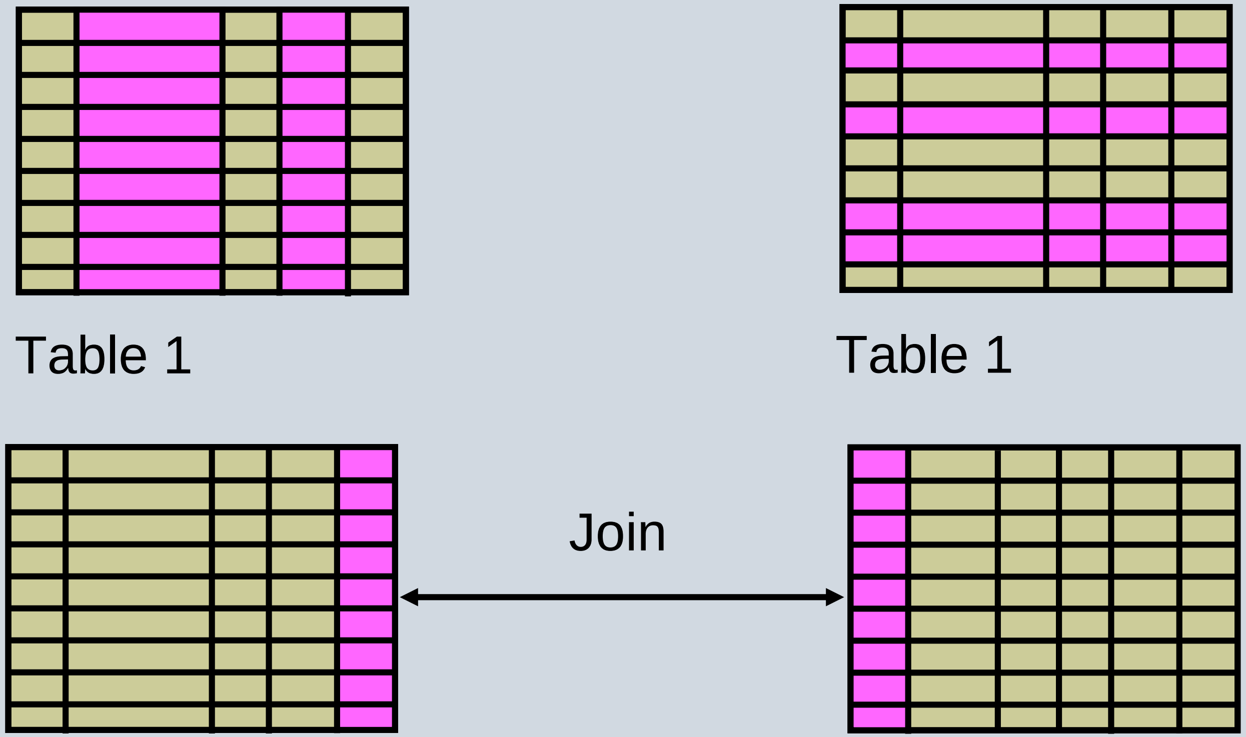
Retrieving Data Using the SQL SELECT Statement

Capabilities of SQL `SELECT` Statements



Projection

Selection

Table 1

Table 1

Join

Table 1

Table 2

Basic `SELECT` Statement

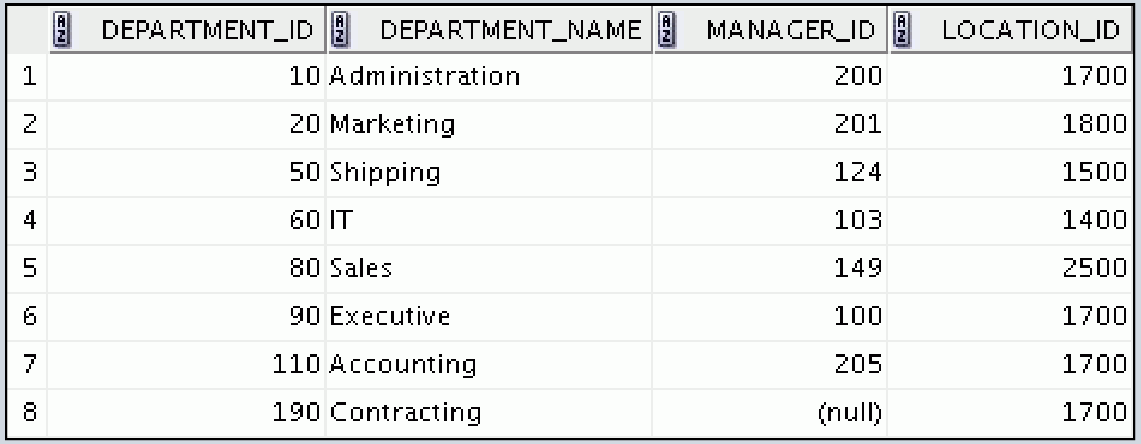
```
SELECT {*|[DISTINCT] column|expression [alias],...}
FROM    table;
```

- `SELECT` identifies the columns to be displayed.
- `FROM` identifies the table containing those columns.

SELECT <attribute list> {*|[DISTINCT] *column*|*expression* [*alias*],...}
FROM <table list> {[*alias*],...}
WHERE <condition>;

Selecting All Columns

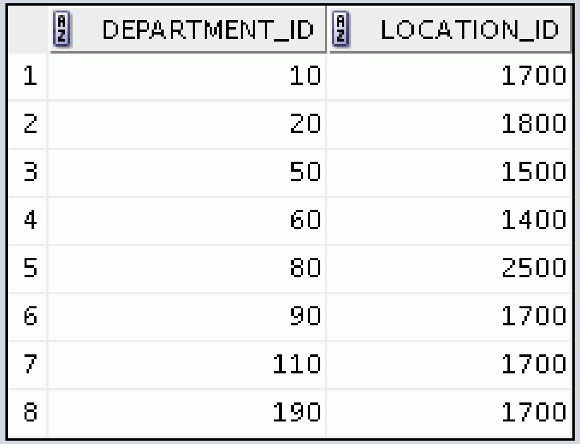
```
SELECT *
FROM   departments;
```



	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700
8	190	Contracting	(null)	1700

Selecting Specific Columns

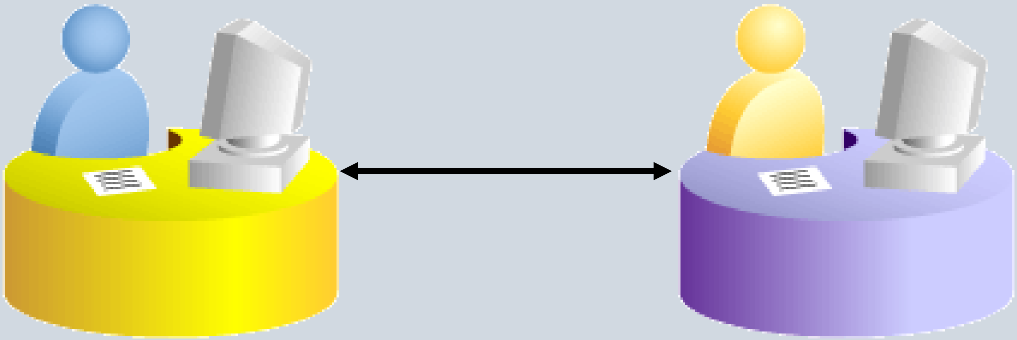
```
SELECT department_id, location_id
FROM   departments;
```



	DEPARTMENT_ID	LOCATION_ID
1	10	1700
2	20	1800
3	50	1500
4	60	1400
5	80	2500
6	90	1700
7	110	1700
8	190	1700

Referencing Another User’s Tables

- Tables belonging to other users are not in the user’s schema.
- You should use the owner’s name as a prefix to those tables.



USERA

USERB

```
SELECT *
FROM userB.employees;
```

```
SELECT *
FROM userA.employees;
```

Writing SQL Statements

- SQL statements are not case sensitive.
- SQL statements can be entered on one or more lines.
- Keywords cannot be abbreviated or split across lines.
- Clauses are usually placed on separate lines.
- Indents are used to enhance readability.
- In SQL Developer, SQL statements can be optionally terminated by a semicolon (;). Semicolons are required when you execute multiple SQL statements.
- In SQL*Plus, you are required to end each SQL statement with a semicolon (;).

Column Heading Defaults

- SQL Developer:
  - Default heading alignment: Left-aligned
  - Default heading display: Uppercase
- SQL*Plus:
  - Character and Date column headings are left-aligned.
  - Number column headings are right-aligned.
  - Default heading display: Uppercase

Arithmetic Expressions

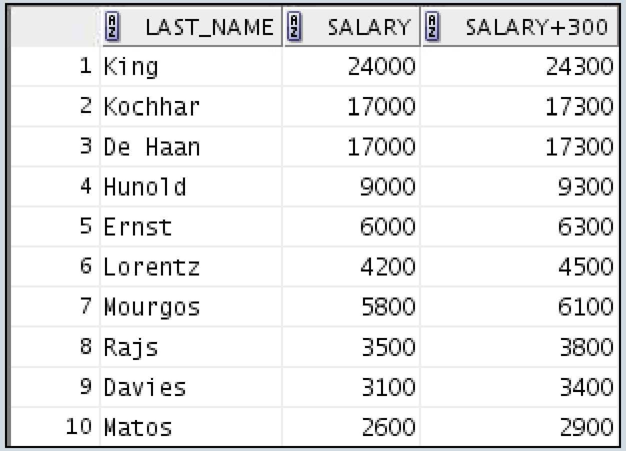
Create expressions with number and date data by using arithmetic operators.

Operator	Description
+	Add
-	Subtract
*	Multiply
/	Divide

Using Arithmetic Operators

Show all the last names, salaries and new salaries of employees if you make 300 dollar increase in their salaries

```
SELECT last_name, salary, salary + 300
FROM   employees;
```



	LAST_NAME	SALARY	SALARY+300
1	King	24000	24300
2	Kochhar	17000	17300
3	De Haan	17000	17300
4	Hunold	9000	9300
5	Ernst	6000	6300
6	Lorentz	4200	4500
7	Mourgos	5800	6100
8	Rajs	3500	3800
9	Davies	3100	3400
10	Matos	2600	2900

...

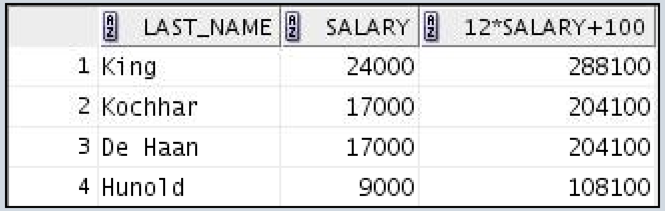
Operator Precedence

Please be careful for the usage of parenthesis when you are applying arithmetical operations.

Which one is correct for the following query:
Find the annual salaries of all employees after increasing salaries with 100 dollar.

```
SELECT last_name, salary, 12*salary+100
FROM   employees;
```

1

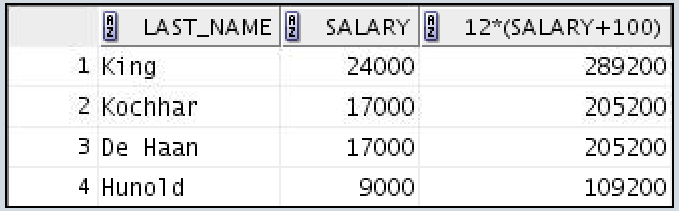


	LAST_NAME	SALARY	12*SALARY+100
1	King	24000	288100
2	Kochhar	17000	204100
3	De Haan	17000	204100
4	Hunold	9000	108100

...

```
SELECT last_name, salary, 12*(salary+100)
FROM   employees;
```

2



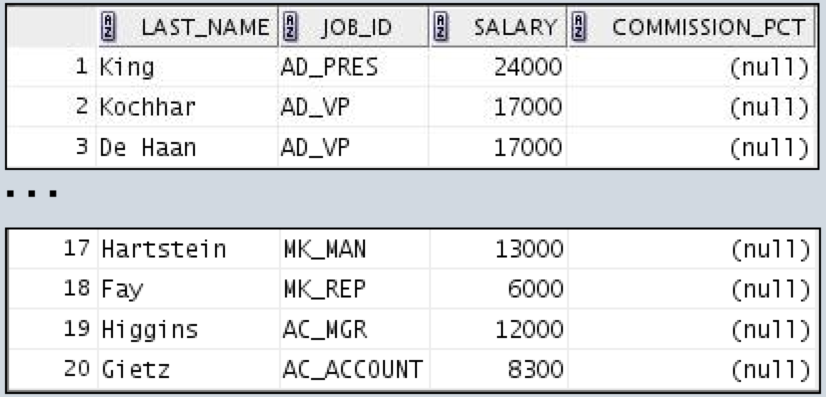
	LAST_NAME	SALARY	12*(SALARY+100)
1	King	24000	289200
2	Kochhar	17000	205200
3	De Haan	17000	205200
4	Hunold	9000	109200

...

Defining a Null Value

- Null is a value that is unavailable, unassigned, unknown, or inapplicable.
- Null is not the same as zero or a blank space.

```
SELECT last_name, job_id, salary, commission_pct
FROM   employees;
```



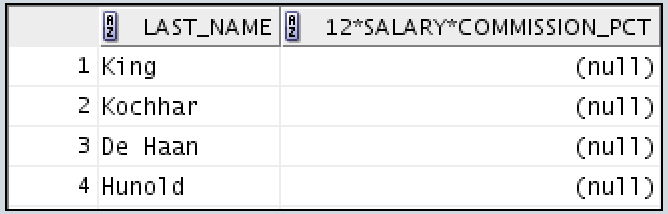
	LAST_NAME	JOB_ID	SALARY	COMMISSION_PCT
1	King	AD_PRES	24000	(null)
2	Kochhar	AD_VP	17000	(null)
3	De Haan	AD_VP	17000	(null)
...				
17	Hartstein	MK_MAN	13000	(null)
18	Fay	MK_REP	6000	(null)
19	Higgins	AC_MGR	12000	(null)
20	Gietz	AC_ACCOUNT	8300	(null)

Null Values in Arithmetic Expressions

Arithmetic expressions containing a null value evaluate to null.

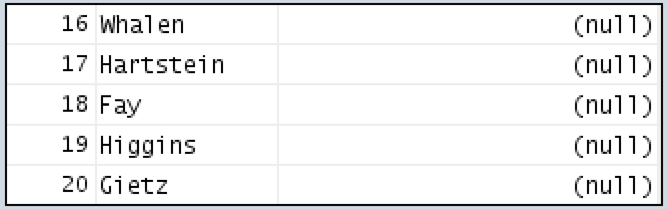
Find the Annual Commissions of all employees.

```
SELECT last_name, 12*salary*commission_pct
FROM   employees;
```



	LAST_NAME	12*SALARY*COMMISSION_PCT
1	King	(null)
2	Kochhar	(null)
3	De Haan	(null)
4	Hunold	(null)

…



16	Whalen	(null)
17	Hartstein	(null)
18	Fay	(null)
19	Higgins	(null)
20	Gietz	(null)

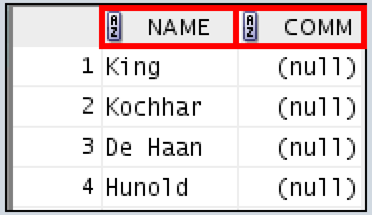
Defining a Column Alias

A column alias:

- Renames a column heading
- Is useful with calculations
- Immediately follows the column name (There can also be the optional `AS` keyword between the column name and the alias.)
- Requires double quotation marks if it contains spaces or special characters, or if it is case-sensitive

Using Column Aliases

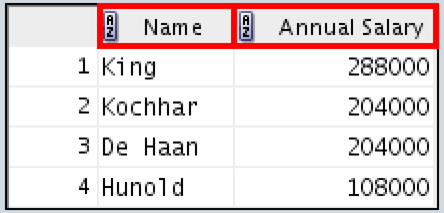
```
SELECT last_name AS name, commission_pct comm
FROM   employees;
```



	NAME	COMM
1	King	(null)
2	Kochhar	(null)
3	De Haan	(null)
4	Hunold	(null)

…

```
SELECT last_name "Name" , salary*12 "Annual Salary"
FROM   employees;
```



	Name	Annual Salary
1	King	288000
2	Kochhar	204000
3	De Haan	204000
4	Hunold	108000

…

Concatenation Operator

A concatenation operator:

- Links columns or character strings to other columns
- Is represented by two vertical bars (||)
- Creates a resultant column that is a character expression

Q: List The employee last name and job_id as EMPLOYEES.

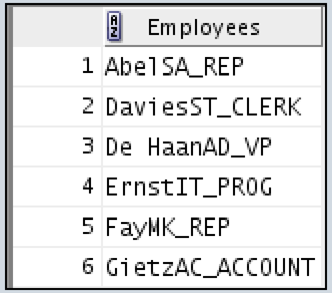
Concatenation Operator

A concatenation operator:

- Links columns or character strings to other columns
- Is represented by two vertical bars (||)
- Creates a resultant column that is a character expression

List The employee last name and job_id as EMPLOYEES

```
SELECT   last_name||job_id AS "Employees"
FROM     employees;
```



	Employees
1	AbelSA_REP
2	DaviesST_CLERK
3	De HaanAD_VP
4	ErnstIT_PROG
5	FayMK_REP
6	GietzAC_ACCOUNT

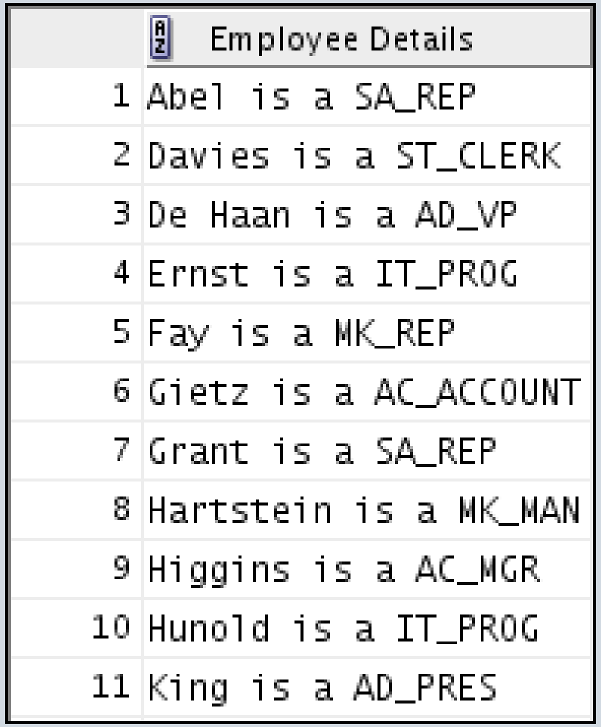
…

Literal Character Strings

- A literal is a character, a number, or a date that is included in the `SELECT` statement.
- Date and character literal values must be enclosed within single quotation marks.
- Each character string is output once for each row returned.

Using Literal Character Strings

Q: How can you produce the following output?

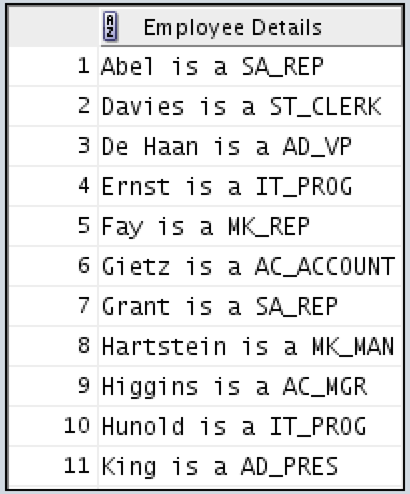


	Employee Details
1	Abel is a SA_REP
2	Davies is a ST_CLERK
3	De Haan is a AD_VP
4	Ernst is a IT_PROG
5	Fay is a MK_REP
6	Gietz is a AC_ACCOUNT
7	Grant is a SA_REP
8	Hartstein is a MK_MAN
9	Higgins is a AC_MGR
10	Hunold is a IT_PROG
11	King is a AD_PRES

…

Using Literal Character Strings

```
SELECT last_name ||' is a '||job_id
       AS "Employee Details"
FROM   employees;
```



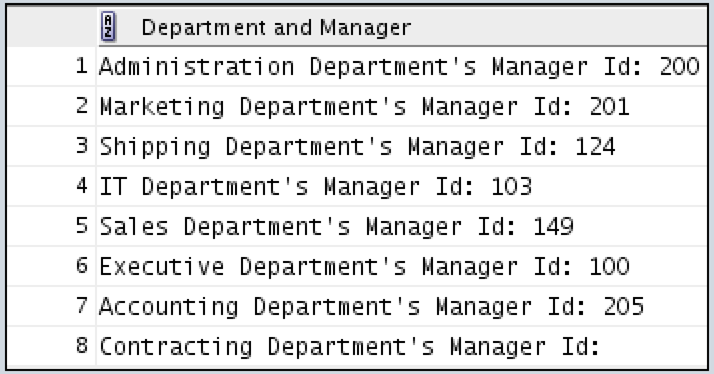
	Employee Details
1	Abel is a SA_REP
2	Davies is a ST_CLERK
3	De Haan is a AD_VP
4	Ernst is a IT_PROG
5	Fay is a MK_REP
6	Gietz is a AC_ACCOUNT
7	Grant is a SA_REP
8	Hartstein is a MK_MAN
9	Higgins is a AC_MGR
10	Hunold is a IT_PROG
11	King is a AD_PRES

…

Alternative Quote (q) Operator

- Specify your own quotation mark delimiter.
- Select any delimiter.
- Increase readability and usability.

```
SELECT department_name || q'[ Department's Manager Id: ]'
       || manager_id
       AS "Department and Manager"
FROM departments;
```



	Department and Manager
1	Administration Department's Manager Id: 200
2	Marketing Department's Manager Id: 201
3	Shipping Department's Manager Id: 124
4	IT Department's Manager Id: 103
5	Sales Department's Manager Id: 149
6	Executive Department's Manager Id: 100
7	Accounting Department's Manager Id: 205
8	Contracting Department's Manager Id:

Duplicate Rows

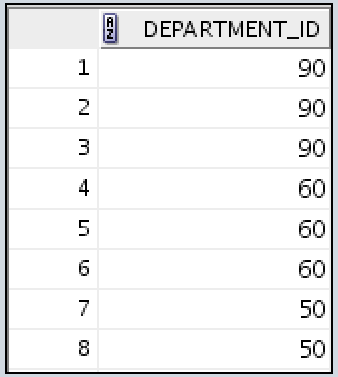
The default display of queries is all rows, including duplicate rows. 1

To eliminate the duplicate rows, use the DISTINCT keyword. 2

Q: Find all the department ids where the employees are working on.

1

```
SELECT department_id
FROM   employees;
```



	DEPARTMENT_ID
1	90
2	90
3	90
4	60
5	60
6	60
7	50
8	50

…

Duplicate Rows

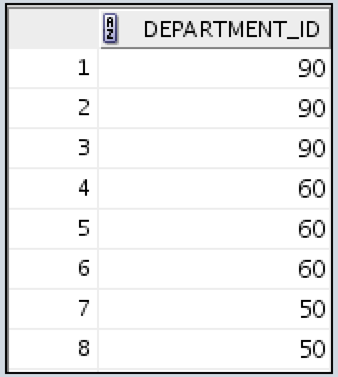
The default display of queries is all rows, including duplicate rows. 1

To eliminate the duplicate rows, use the DISTINCT keyword. 2

Find all the department ids where the employees are working on.

1

```
SELECT department_id
FROM   employees;
```

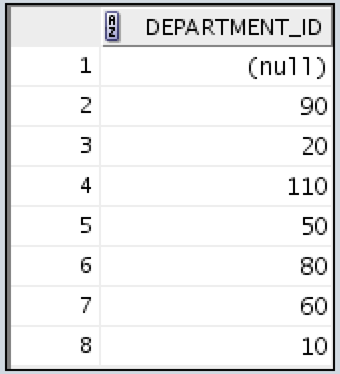


	DEPARTMENT_ID
1	90
2	90
3	90
4	60
5	60
6	60
7	50
8	50

...

2

```
SELECT DISTINCT department_id
FROM   employees;
```

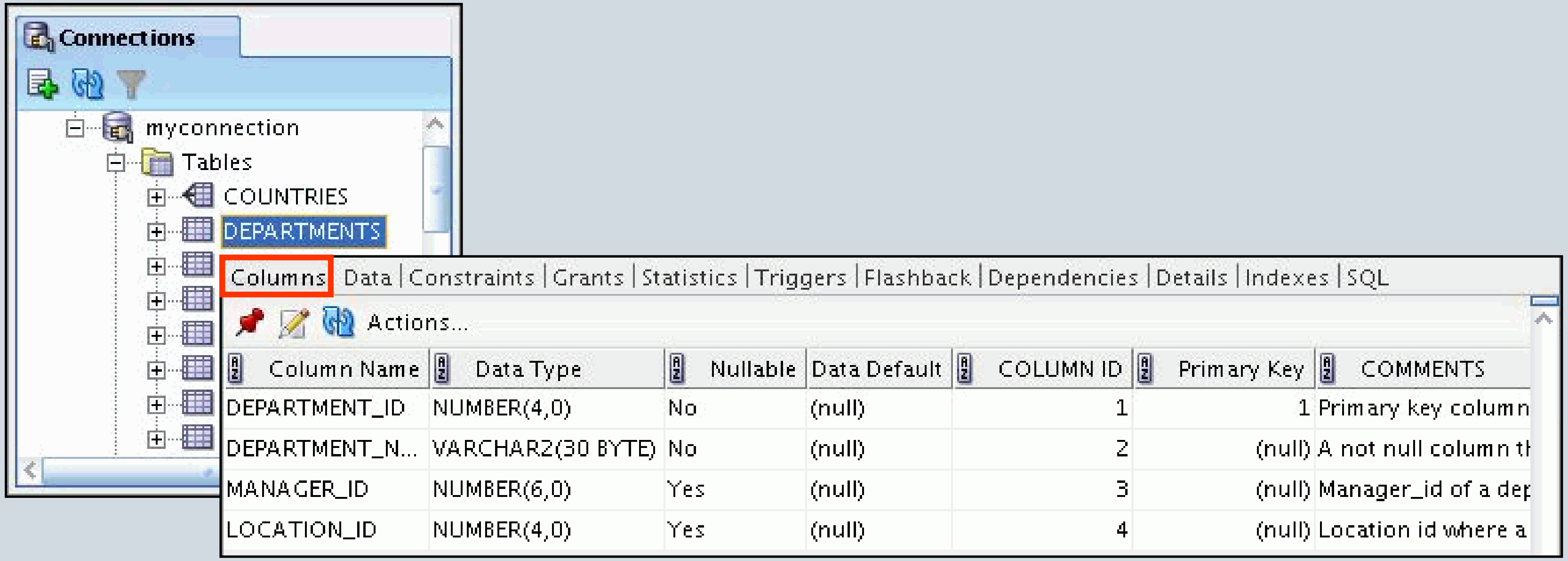


	DEPARTMENT_ID
1	(null)
2	90
3	20
4	110
5	50
6	80
7	60
8	10

Displaying the Table Structure

- Use the `DESCRIBE` command to display the structure of a table.
- Or, select the table in the Connections tree and use the Columns tab to view the table structure.

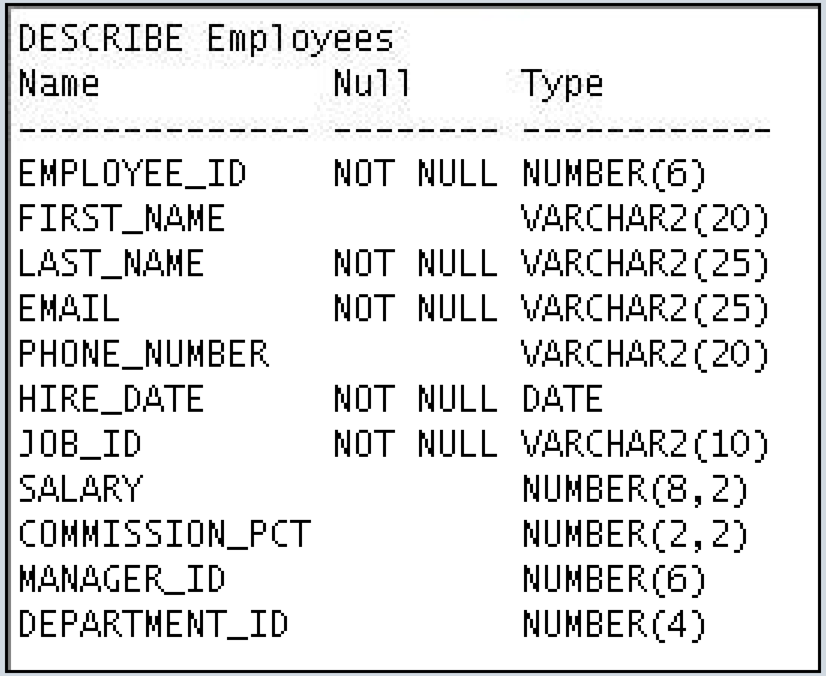
```
DESC[RIBE] tablename
```



Column Name	Data Type	Nullable	Data Default	COLUMN ID	Primary Key	COMMENTS
DEPARTMENT_ID	NUMBER(4,0)	No	(null)	1	1	Primary key column
DEPARTMENT_N...	VARCHAR2(30 BYTE)	No	(null)	2	(null)	A not null column th
MANAGER_ID	NUMBER(6,0)	Yes	(null)	3	(null)	Manager_id of a dep
LOCATION_ID	NUMBER(4,0)	Yes	(null)	4	(null)	Location id where a

Using the `DESCRIBE` Command

```
DESCRIBE employees
```



```
DESCRIBE Employees
Name           Null     Type
-------------- -------- ------------
EMPLOYEE_ID    NOT NULL NUMBER(6)
FIRST_NAME              VARCHAR2(20)
LAST_NAME      NOT NULL VARCHAR2(25)
EMAIL          NOT NULL VARCHAR2(25)
PHONE_NUMBER            VARCHAR2(20)
HIRE_DATE      NOT NULL DATE
JOB_ID         NOT NULL VARCHAR2(10)
SALARY                  NUMBER(8,2)
COMMISSION_PCT          NUMBER(2,2)
MANAGER_ID              NUMBER(6)
DEPARTMENT_ID           NUMBER(4)
```

Restricting and Sorting Data

Limiting the Rows That Are Selected

- Restrict the rows that are returned by using the `WHERE` clause:

```
SELECT *|{[DISTINCT] column|expression [alias],...}
FROM   table
[WHERE logical expression(s)];
```

- The `WHERE` clause follows the `FROM` clause.

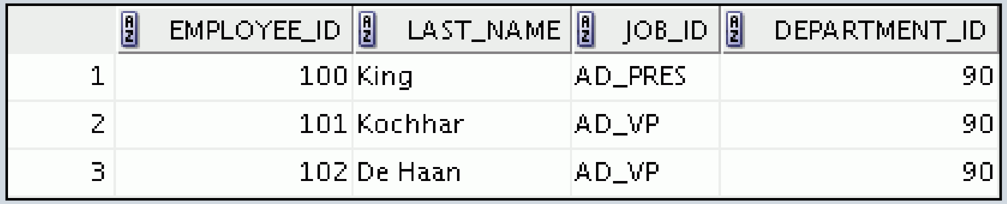
Using the `WHERE` Clause

Q: Find the employees jobs and departments for only the employees working in department 90.

Using the **WHERE** Clause

Find the employees jobs and departments for only the employees working in department 90.

```
SELECT employee_id, last_name, job_id, department_id
FROM   employees
WHERE  department_id = 90 ;
```



	EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	100	King	AD_PRES	90
2	101	Kochhar	AD_VP	90
3	102	De Haan	AD_VP	90

Character Strings and Dates

- Character strings and date values are enclosed with single quotation marks.
- Character values are case-sensitive and date values are format-sensitive.
- The default date display format is `DD-MON-RR`.

Q: Find the employees whose last name is Whalen.

Character Strings and Dates

Find the employees whose last name is Whalen.

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

Q: Find the employees hired on Feb, 17, 1996.

Character Strings and Dates

Find the employees hired on Feb, 17, 1996.

```
SELECT last_name
FROM   employees
WHERE  hire_date = '17-FEB-96' ;
```

Comparison Operators

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to
`BETWEEN ...AND...`	Between two values (inclusive)
`IN(set)`	Match any of a list of values
`LIKE`	Match a character pattern
`IS NULL`	Is a null value

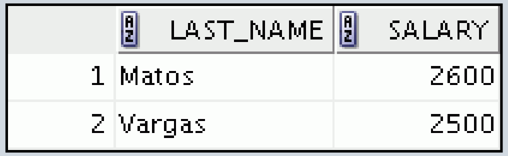
Using Comparison Operators

Q: List the employees whose salary is less than 3000.

Using Comparison Operators

List the employees whose salary is less than 3000.

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



	LAST_NAME	SALARY
1	Matos	2600
2	Vargas	2500

RELATIONAL ALGEBRA

Relational algebra is a set of operations group in order to handle the queries in relational data model.

We can divide these operations into two. One of them is set operations from mathematical set theory. (UNION, INTERSECTION, DIFFERENCE, CARTESIAN PRODUCT). The other group related only relational operations (SELECT, PROJECT, and JOIN).

Classification of Relational Operations

A **projection** operation produces a result table with - Only some of the columns of its input table.
A **selection** operation produces a result table with -
All of the columns of the input table -Only those rows of its input table that satisfy some criteria.
A **join** or **product** operation produces a result table by - Combining the columns of two input tables.
A **set** operation produces a result table by - Combining rows from one or the other of its input tables

Select Operation: SELECTION OF ROWS –(*where stmt. in SQL*)

It is used to select a subset of tuples in a relation. It corresponds to *where* clause in SQL.

$$\delta_{\textbf{<General Condition>}}\textbf{(<Relation Name>)}$$

- Select tuples for all employees who work either in dept 60 and salary is over than $5,000 or work in department 80 and salary is over than $10,000.

$\delta_{\text{(department\_id = 60 AND salary >5000) OR (department\_id = 80 AND salary > 10000)}}$ (EMPLOYEES)

```
SELECT *
FROM   employees
WHERE (department_id = 60 AND salary >5000) OR
      (department_id = 80 AND salary > 10000);
```

Project Operation: SELECTION OF COLUMNS –(*select stmt. in SQL*)

Selects certain columns from operation. It corresponds to *select* clause in SQL.

$$\pi_{\text{<attribute list>}} \text{(<relation name>)}$$

- Retrieve the first name, last name and salary of employees who worked on department 80.

$$\pi_{\text{last\_name, first\_name, salary}} \; (\delta_{\text{(department\_id = 80)}} \text{(EMPLOYEES))}$$

or

$$\text{DEPT-EMPS} \leftarrow \delta_{\text{(department\_id = 80)}} \text{(EMPLOYEE)}$$

$$\text{RESULT} \leftarrow \pi_{\text{last\_name, first\_name, salary}} \; \text{(DEPT-EMPS)}$$

```
SELECT last_name, first_name, salary
FROM   employees
WHERE department_id = 80 ;
```

SQL

Range Conditions Using the BETWEEN Operator

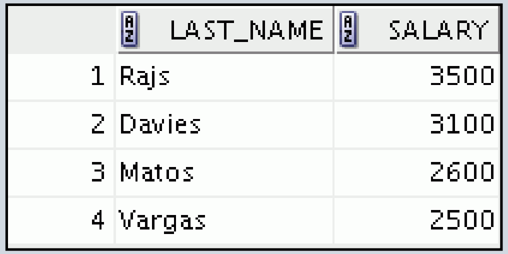
Use the `BETWEEN` operator to display rows based on a range of values:

List the employees whose salary is between 2500 & 3500.

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

Lower limit

Upper limit



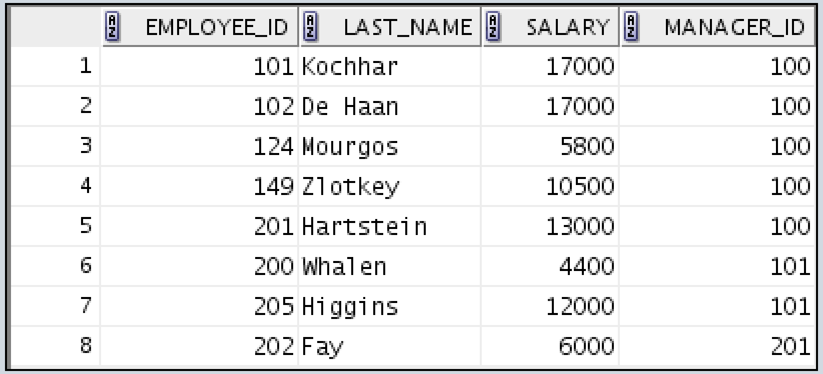
	LAST_NAME	SALARY
1	Rajs	3500
2	Davies	3100
3	Matos	2600
4	Vargas	2500

Membership Condition Using the IN Operator

Use the IN operator to test for values in a list:

Find all the employees of managers 100, 101, 201.

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



	EMPLOYEE_ID	LAST_NAME	SALARY	MANAGER_ID
1	101	Kochhar	17000	100
2	102	De Haan	17000	100
3	124	Mourgos	5800	100
4	149	Zlotkey	10500	100
5	201	Hartstein	13000	100
6	200	Whalen	4400	101
7	205	Higgins	12000	101
8	202	Fay	6000	201

Pattern Matching 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.

Q: Find all the employees whose name starts with S.

Pattern Matching Using the `LIKE` Operator

- `%` denotes zero or many characters.
- `_` denotes one character.

Find all the employees whose name starts with S.

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

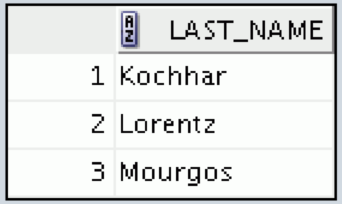
Q: Find the employees whose second character in the last name is ‘o’.

Combining Wildcard Characters

- You can combine the two wildcard characters (%, _) with literal characters for pattern matching:

Find the employees whose second character in the last name is ‘o’.

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



	LAST_NAME
1	Kochhar
2	Lorentz
3	Mourgos

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

Using the `NULL` Conditions

Test for nulls with the `IS NULL` operator.

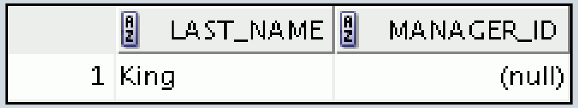
Q: Find the employees who do not have any manager.

Using the NULL Conditions

Test for nulls with the IS NULL operator.

Find the employees who do not have any manager.

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



	LAST_NAME	MANAGER_ID
1	King	(null)

Defining Conditions Using the Logical Operators

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 condition is false

Using the AND Operator

AND requires both the component conditions to be true:

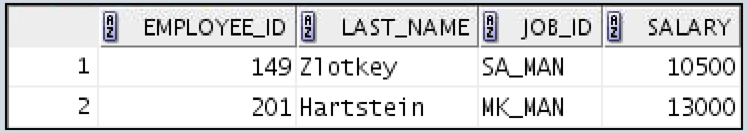
Q: List all kinds of MANagers whose salary is greater than 10000.

Using the AND Operator

AND requires both the component conditions to be true:

List all kinds of MANagers whose salary is greater than 10000.

```
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
1	149	Zlotkey	SA_MAN	10500
2	201	Hartstein	MK_MAN	13000

Using the OR Operator

OR requires either component condition to be true:

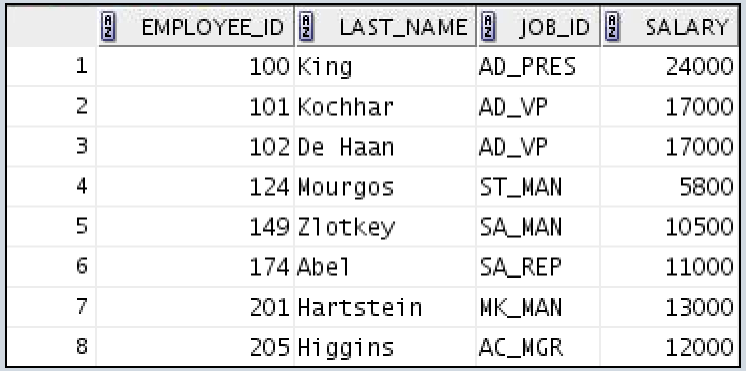
Q: List all MANagers or employees whose salary is greater than 10000.

Using the OR Operator

OR requires either component condition to be true:

List all MANagers or employees whose salary is greater than 10000.

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



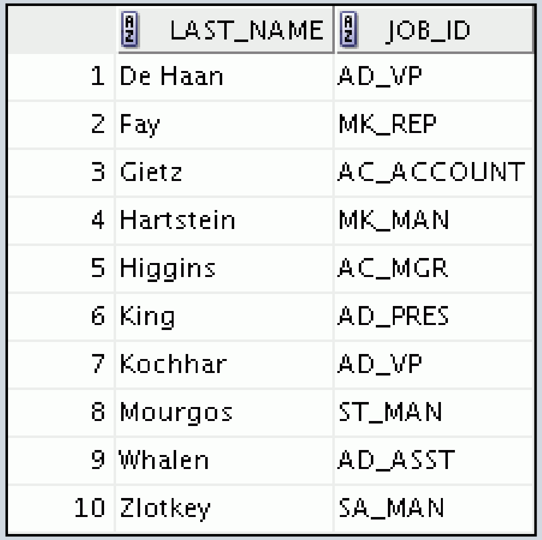
	EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	100	King	AD_PRES	24000
2	101	Kochhar	AD_VP	17000
3	102	De Haan	AD_VP	17000
4	124	Mourgos	ST_MAN	5800
5	149	Zlotkey	SA_MAN	10500
6	174	Abel	SA_REP	11000
7	201	Hartstein	MK_MAN	13000
8	205	Higgins	AC_MGR	12000

Q: List all the employees who are not the IT_PROG, ST_CLERK or SA_REP.

Using the NOT Operator

List all the employees who are not the IT_PROG, ST_CLERK or SA_REP.

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



	LAST_NAME	JOB_ID
1	De Haan	AD_VP
2	Fay	MK_REP
3	Gietz	AC_ACCOUNT
4	Hartstein	MK_MAN
5	Higgins	AC_MGR
6	King	AD_PRES
7	Kochhar	AD_VP
8	Mourgos	ST_MAN
9	Whalen	AD_ASST
10	Zlotkey	SA_MAN

Rules of Precedence

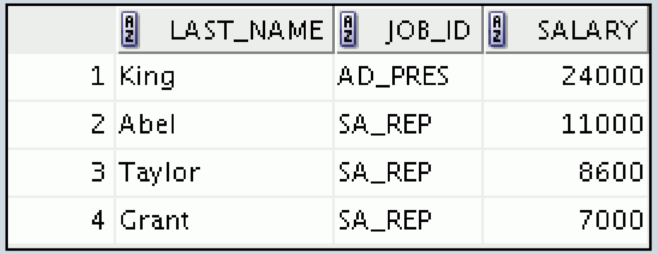
Operator	Meaning
1	Arithmetic operators
2	Concatenation operator
3	Comparison conditions
4	IS [NOT] NULL, LIKE, [NOT] IN
5	[NOT] BETWEEN
6	Not equal to
7	NOT logical condition
8	AND logical condition
9	OR logical condition

You can use parentheses to override rules of precedence.

Rules of Precedence

1

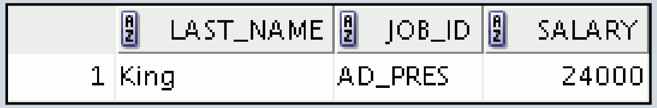
```
SELECT last_name, job_id, salary
FROM   employees
WHERE  job_id = 'SA_REP'
OR     job_id = 'AD_PRES'
AND    salary > 15000;
```



	LAST_NAME	JOB_ID	SALARY
1	King	AD_PRES	24000
2	Abel	SA_REP	11000
3	Taylor	SA_REP	8600
4	Grant	SA_REP	7000

2

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

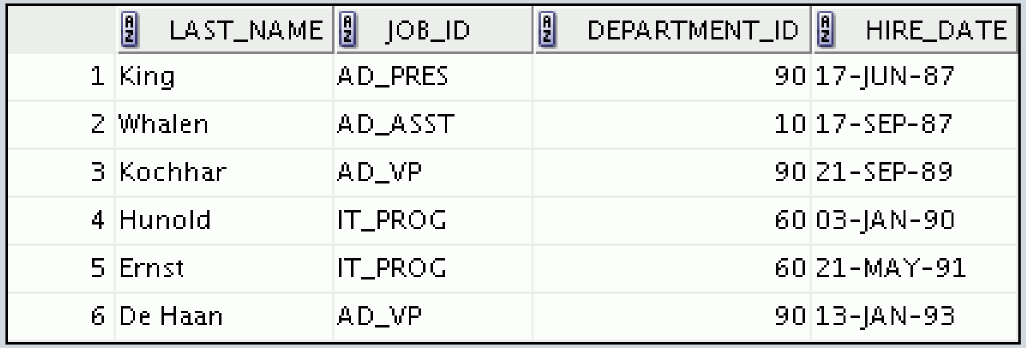


	LAST_NAME	JOB_ID	SALARY
1	King	AD_PRES	24000

Using the ORDER BY Clause

- Sort the 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 ;
```



	LAST_NAME	JOB_ID	DEPARTMENT_ID	HIRE_DATE
1	King	AD_PRES	90	17-JUN-87
2	Whalen	AD_ASST	10	17-SEP-87
3	Kochhar	AD_VP	90	21-SEP-89
4	Hunold	IT_PROG	60	03-JAN-90
5	Ernst	IT_PROG	60	21-MAY-91
6	De Haan	AD_VP	90	13-JAN-93

…

Sorting

- 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

- 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

- Sorting by using the column’s numeric position:

```
SELECT   last_name, job_id, department_id, hire_date
FROM     employees
ORDER BY 3;
```

3

- Sorting by multiple columns:

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

4