1

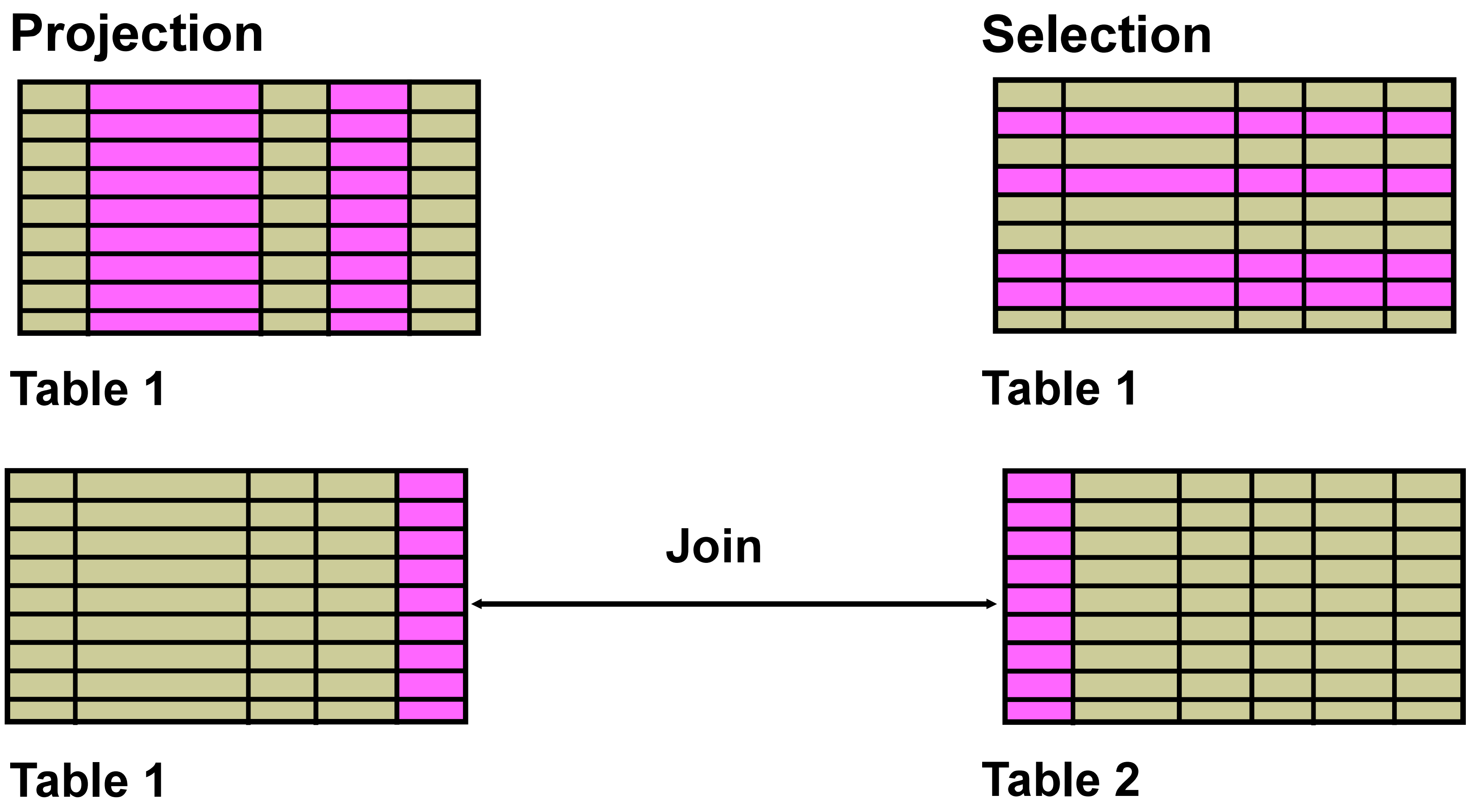
Retrieve Data using the SQL SELECT Statement

Objectives

After completing this lesson, you should be able to do the following:

- List the capabilities of SQL `SELECT` statements
- Execute a basic `SELECT` statement
- Identify and use the key features of Oracle SQL Developer

Capabilities of SQL SELECT Statements



Projection

Selection

Table 1

Table 1

Join

Table 1

Table 2

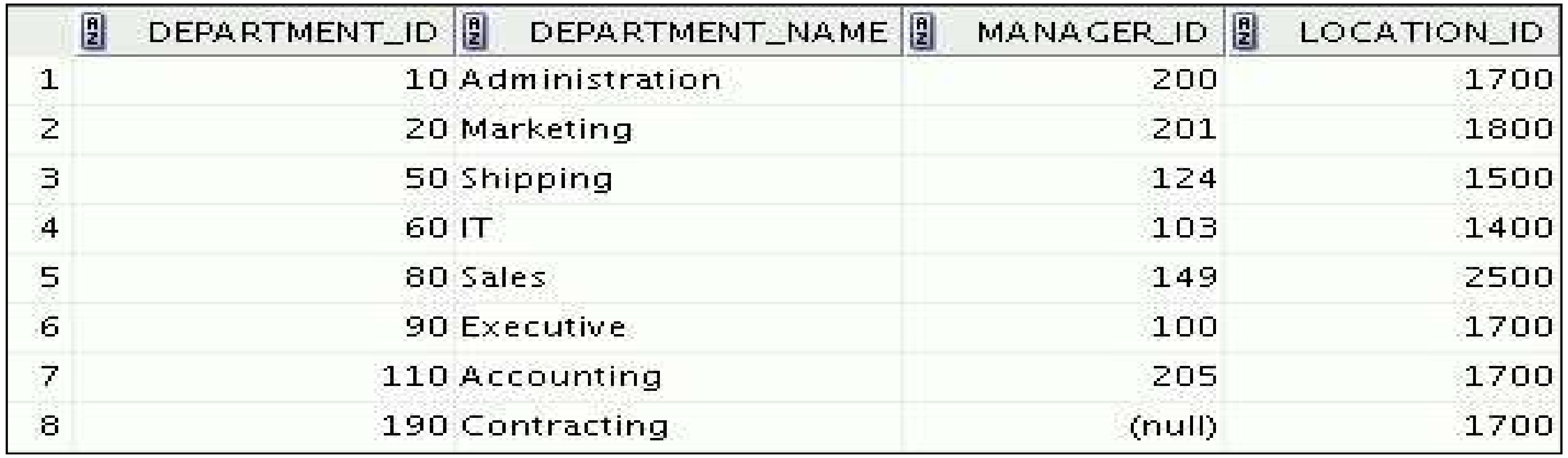
Basic SELECT Statement

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

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

Selecting All Columns

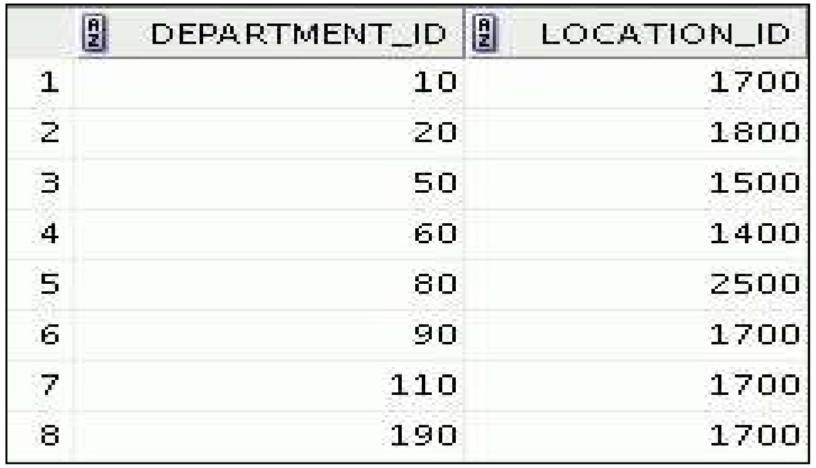
```
SELECT *
FROM   departments;
```



	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

Writing SQL Statements

- SQL statements are not case-sensitive.
- SQL statements can be 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 optionally be terminated by a semicolon (;). Semicolons are required if 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: Center
  - Default heading display: Uppercase
- SQL*Plus:
  - Character and Date column headings are left-aligned
  - Number column headings are right-aligned
  - Default heading display: Uppercase

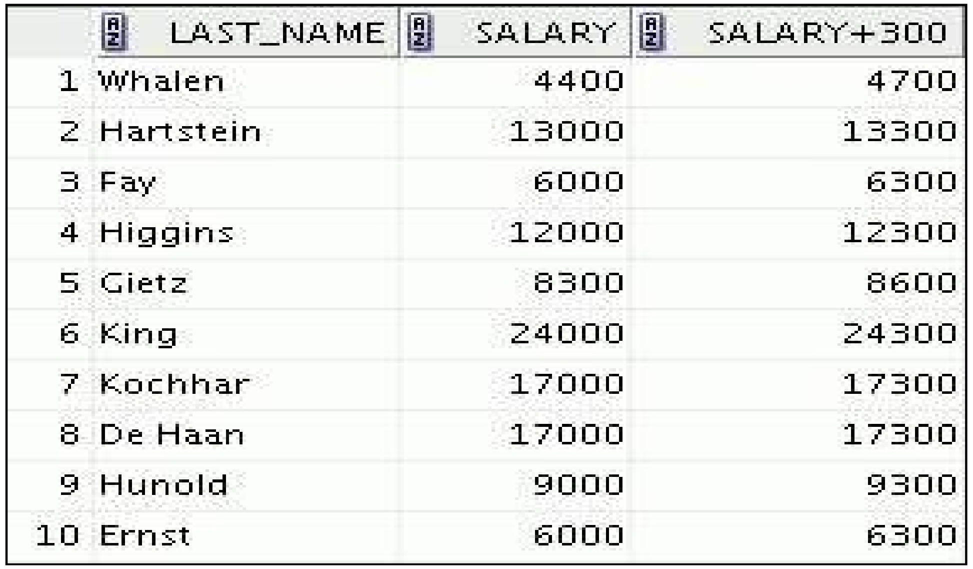
Arithmetic Expressions

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

Operator	Description
+	Add
-	Subtract
*	Multiply
/	Divide

Using Arithmetic Operators

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



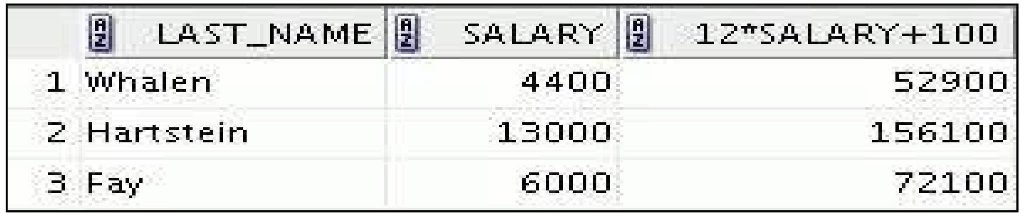
	LAST_NAME	SALARY	SALARY+300
1	Whalen	4400	4700
2	Hartstein	13000	13300
3	Fay	6000	6300
4	Higgins	12000	12300
5	Gietz	8300	8600
6	King	24000	24300
7	Kochhar	17000	17300
8	De Haan	17000	17300
9	Hunold	9000	9300
10	Ernst	6000	6300

...

Operator Precedence

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

1

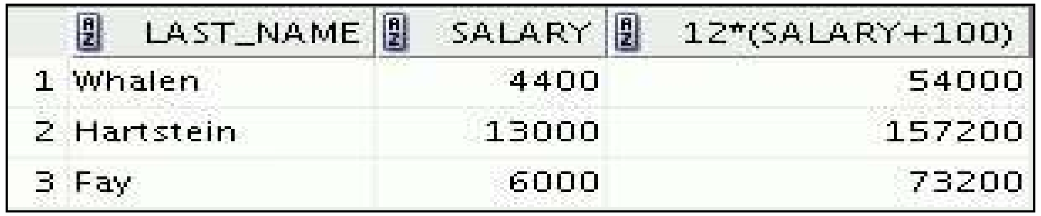


	LAST_NAME	SALARY	12*SALARY+100
1	Whalen	4400	52900
2	Hartstein	13000	156100
3	Fay	6000	72100

…

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

2



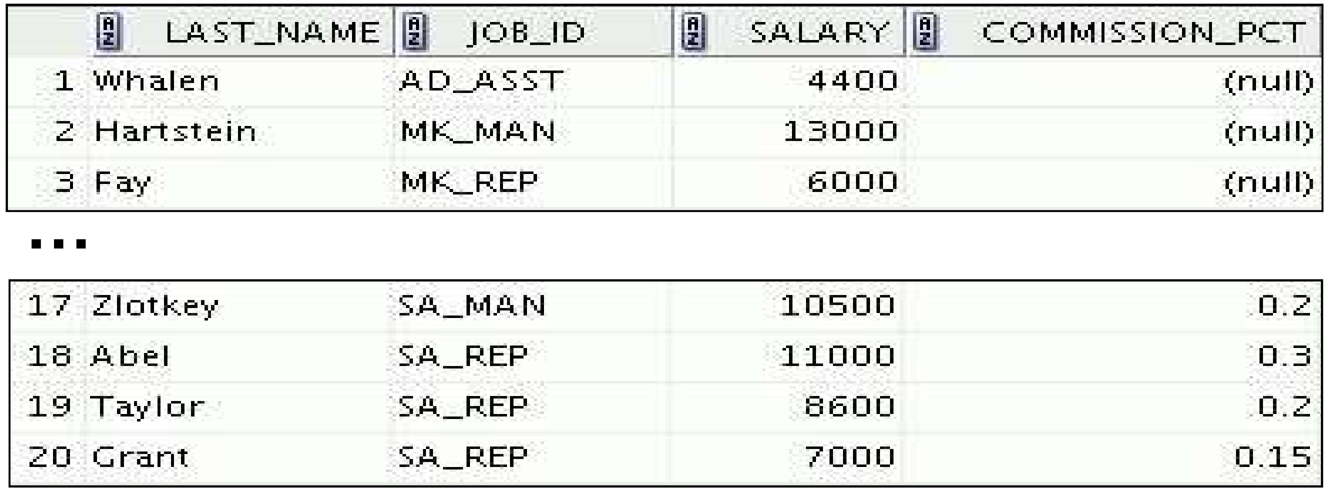
	LAST_NAME	SALARY	12*(SALARY+100)
1	Whalen	4400	54000
2	Hartstein	13000	157200
3	Fay	6000	73200

…

Defining a Null Value

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

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

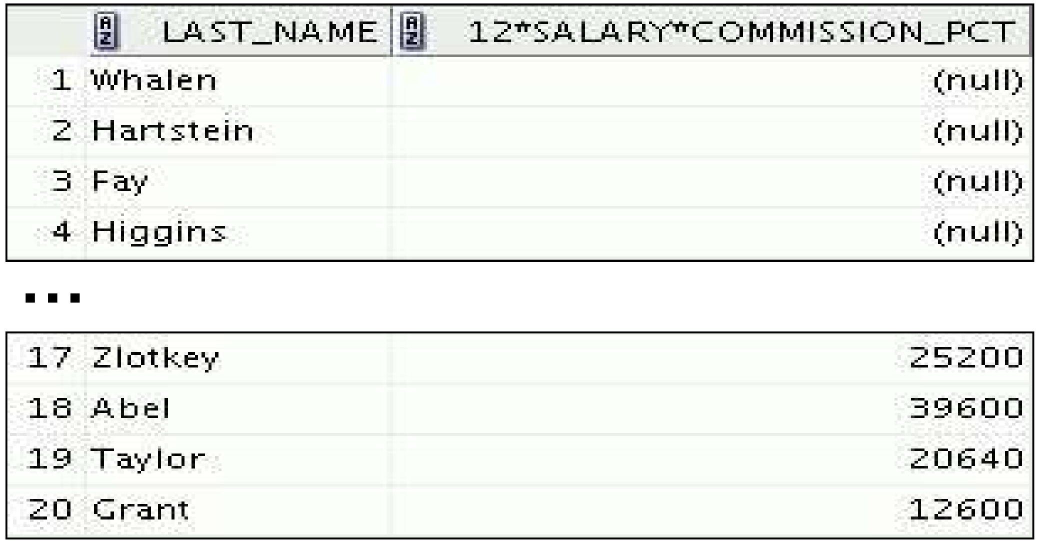


	LAST_NAME	JOB_ID	SALARY	COMMISSION_PCT
1	Whalen	AD_ASST	4400	(null)
2	Hartstein	MK_MAN	13000	(null)
3	Fay	MK_REP	6000	(null)
...				
17	Zlotkey	SA_MAN	10500	0.2
18	Abel	SA_REP	11000	0.3
19	Taylor	SA_REP	8600	0.2
20	Grant	SA_REP	7000	0.15

Null Values

Arithmetic expressions containing a null value evaluate to null.

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



	LAST_NAME	12*SALARY*COMMISSION_PCT
1	Whalen	(null)
2	Hartstein	(null)
3	Fay	(null)
4	Higgins	(null)
...		
17	Zlotkey	25200
18	Abel	39600
19	Taylor	20640
20	Grant	12600

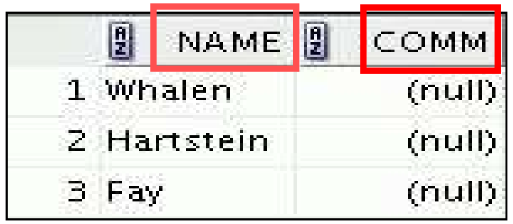
Defining a Column Alias

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 alias.)
- Requires double quotation marks if it contains spaces or special characters, or if it is case-sensitive

Using Column Aliases

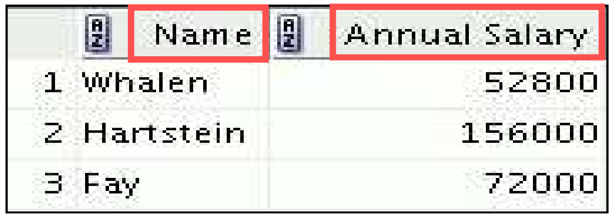
```
SELECT last_name AS name, commission_pct comm
FROM   employees;
```



	NAME	COMM
1	Whalen	(null)
2	Hartstein	(null)
3	Fay	(null)

...

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



	Name	Annual Salary
1	Whalen	52800
2	Hartstein	156000
3	Fay	72000

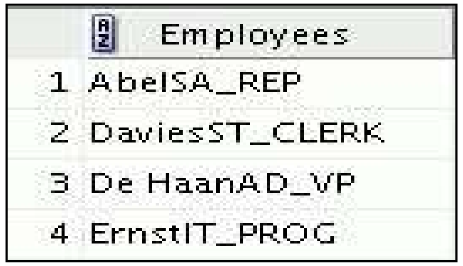
...

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

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



```
    Employees
1 AbelSA_REP
2 DaviesST_CLERK
3 De HaanAD_VP
4 ErnstIT_PROG
```

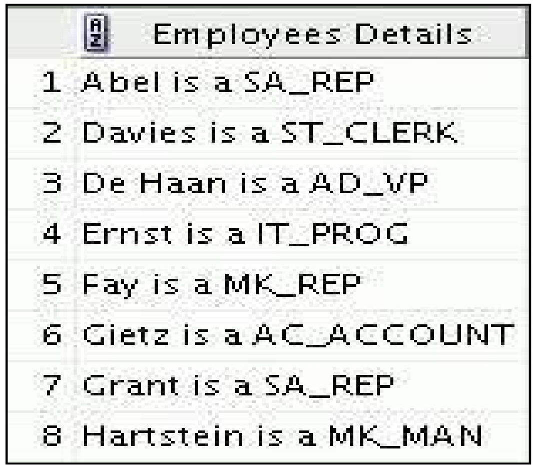
...

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 by single quotation marks.
- Each character string is output once for each row returned.

Using Literal Character Strings

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



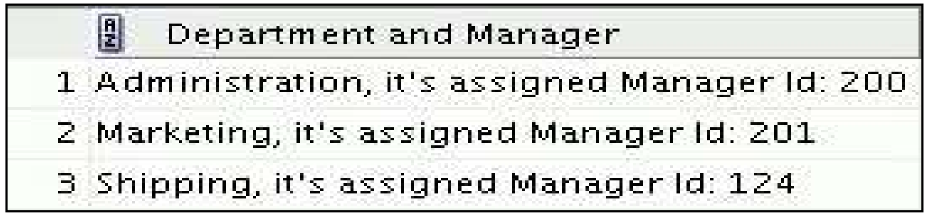
```
   Employees Details
1  Abel is a SA_REP
2  Davies is a ST_CLERK
3  De Haan is a AD_VP
4  Ernst is a IT_PROG
5  Fay is a MK_REP
6  Gietz is a AC_ACCOUNT
7  Grant is a SA_REP
8  Hartstein is a MK_MAN
```

…

Alternative Quote (q) Operator

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

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



```
      Department and Manager
  1 Administration, it's assigned Manager Id: 200
  2 Marketing, it's assigned Manager Id: 201
  3 Shipping, it's assigned Manager Id: 124
```

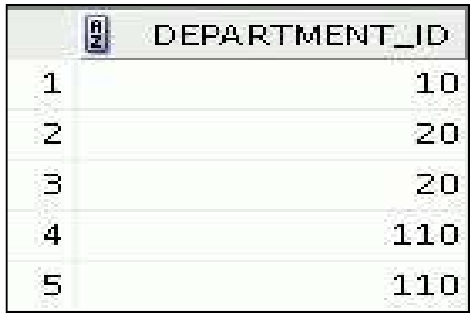
...

Duplicate Rows

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

1

```
SELECT department_id
FROM   employees;
```

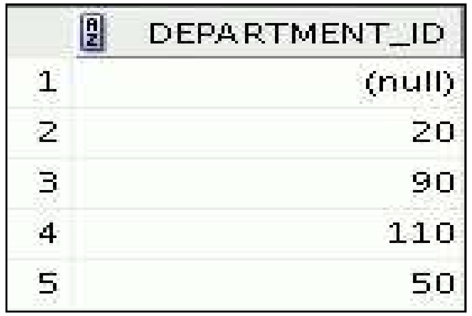


	DEPARTMENT_ID
1	10
2	20
3	20
4	110
5	110

...

2

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



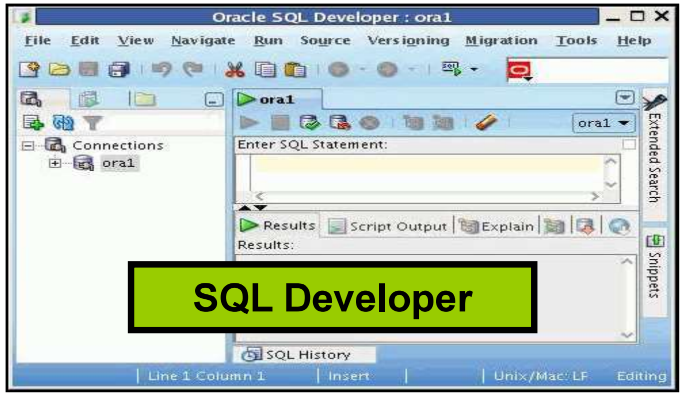
	DEPARTMENT_ID
1	(null)
2	20
3	90
4	110
5	50

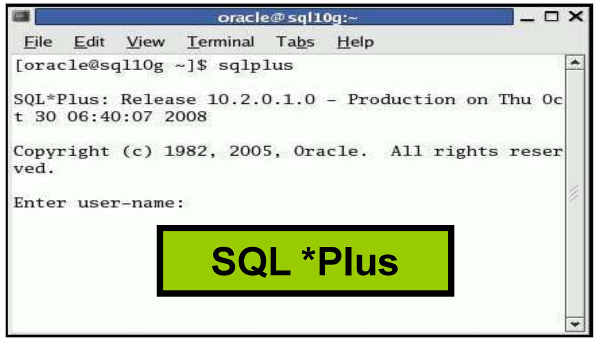
...

Development Environments for SQL

In this course:

- Primarily use Oracle SQL Developer 19.1
- Use SQL*Plus:
  - In case you do not have access to Oracle SQL Developer
  - Or when any command does not run in Oracle SQL Developer

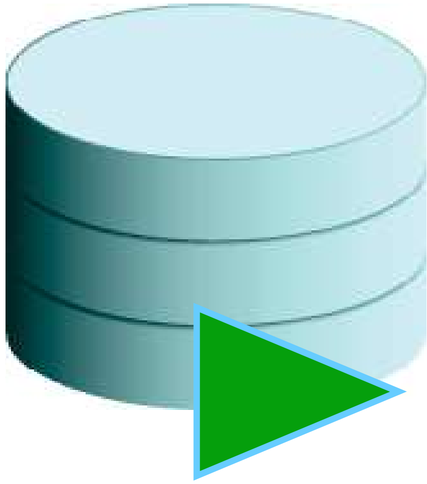

Oracle SQL Developer : ora1
File Edit View Navigate Run Source Versioning Migration Tools Help
Connections
ora1
Enter SQL Statement:
Results Script Output Explain
Results:
Extended Search
Snippets
SQL History
Line 1 Column 1 | Insert | Unix/Mac: LF | Editing
SQL Developer



```
[oracle@sql10g ~]$ sqlplus

SQL*Plus: Release 10.2.0.1.0 - Production on Thu Oc
t 30 06:40:07 2008

Copyright (c) 1982, 2005, Oracle.  All rights reser
ved.

Enter user-name:
```

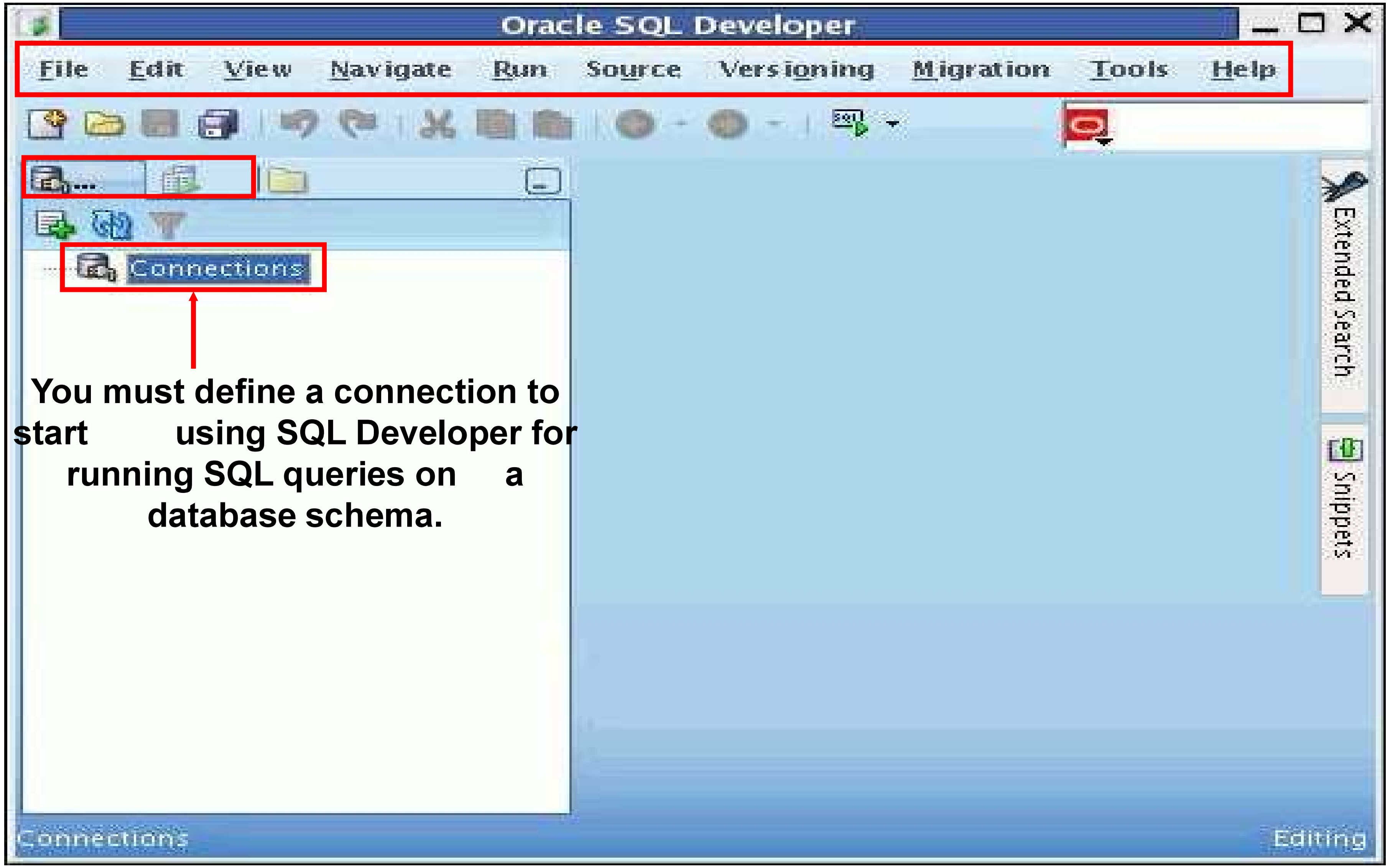
What Is Oracle SQL Developer?

- Oracle SQL Developer is a graphical tool that enhances productivity and simplifies database development tasks.
- You can connect to any target Oracle Database schema by using the standard Oracle Database authentication.



SQL Developer

Oracle SQL Developer Interface

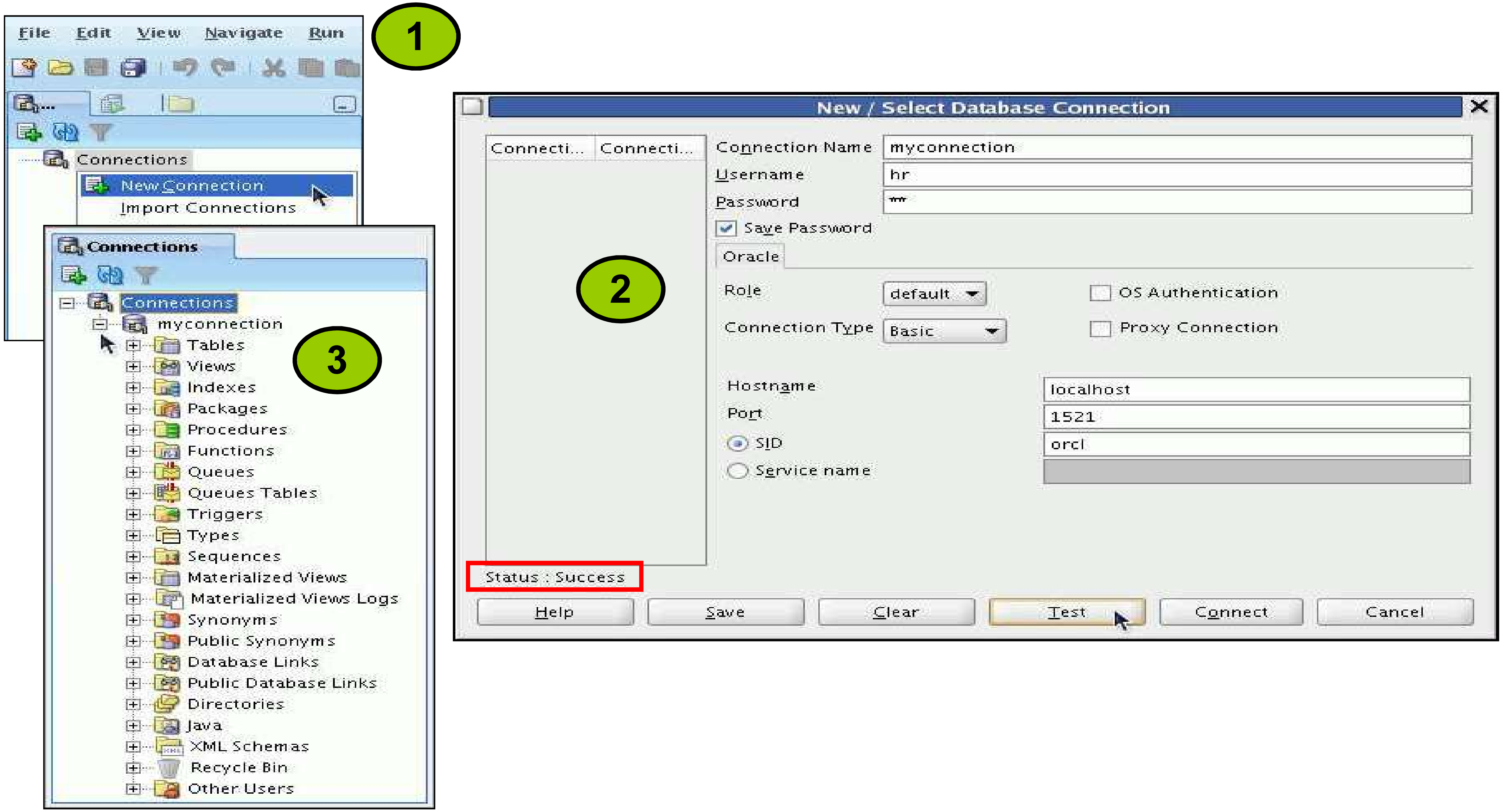


You must define a connection to start using SQL Developer for running SQL queries on a database schema.

Creating a Database Connection

- You must have at least one database connection to use Oracle SQL Developer.
- You can create and test connections for:
  - Multiple databases
  - Multiple schemas
- Oracle SQL Developer automatically imports any connections defined in the `tnsnames.ora` file on your system.
- You can export connections to an XML file.
- Each additional database connection created is listed in the Connections Navigator hierarchy.

Creating a Database Connection



Connection Name: myconnection

Username: hr

Password: **

☑ Save Password

Role: default

Connection Type: Basic

□ OS Authentication

□ Proxy Connection

Hostname: localhost

Port: 1521

● SID: orcl

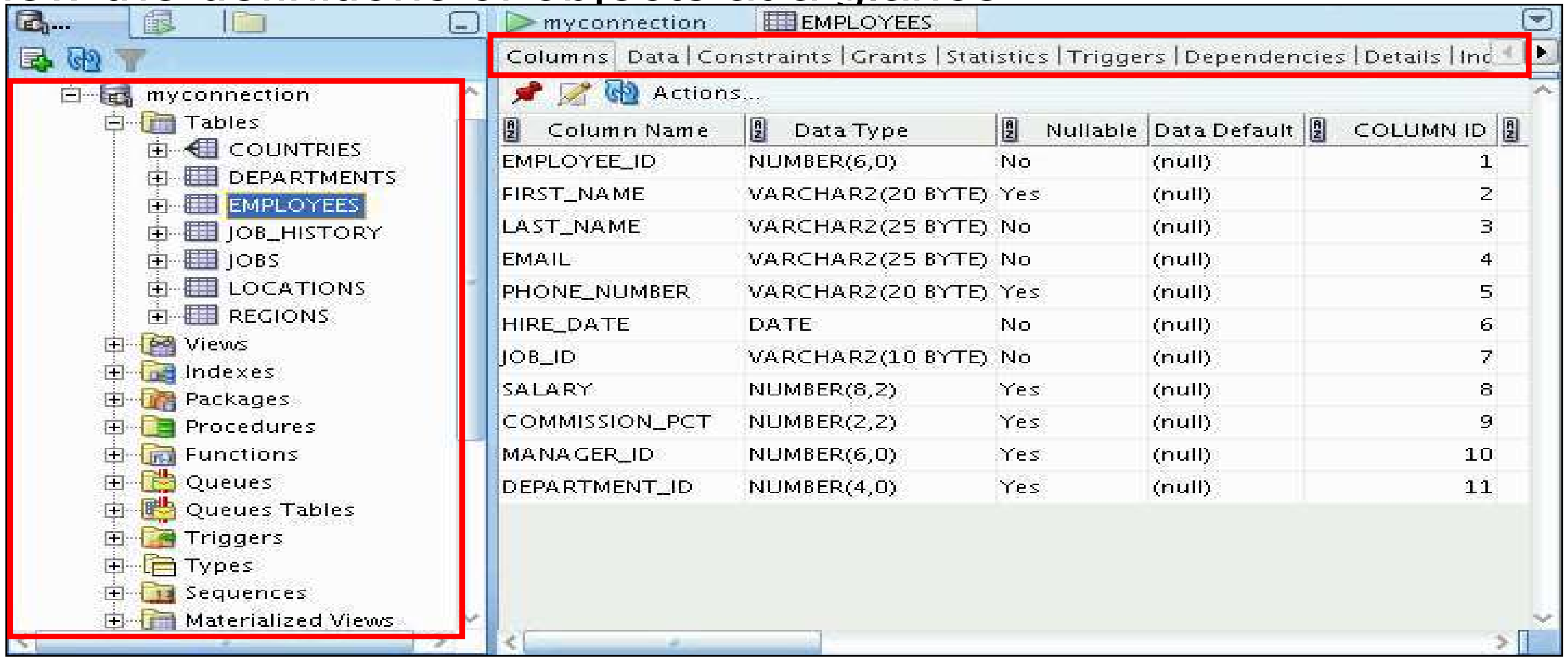
○ Service name

Status : Success

Browsing Database Objects

Use the Connections Navigator to:

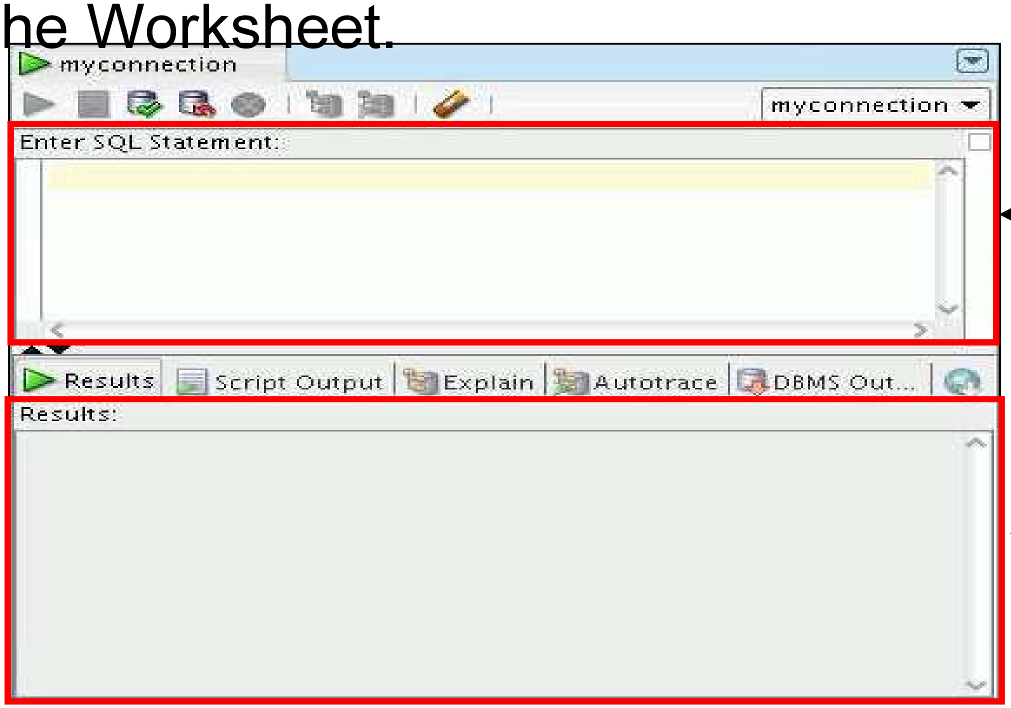
- Browse through many objects in a database schema
- Review the definitions of objects at a glance



Column Name	Data Type	Nullable	Data Default	COLUMN ID
EMPLOYEE_ID	NUMBER(6,0)	No	(null)	1
FIRST_NAME	VARCHAR2(20 BYTE)	Yes	(null)	2
LAST_NAME	VARCHAR2(25 BYTE)	No	(null)	3
EMAIL	VARCHAR2(25 BYTE)	No	(null)	4
PHONE_NUMBER	VARCHAR2(20 BYTE)	Yes	(null)	5
HIRE_DATE	DATE	No	(null)	6
JOB_ID	VARCHAR2(10 BYTE)	No	(null)	7
SALARY	NUMBER(8,2)	Yes	(null)	8
COMMISSION_PCT	NUMBER(2,2)	Yes	(null)	9
MANAGER_ID	NUMBER(6,0)	Yes	(null)	10
DEPARTMENT_ID	NUMBER(4,0)	Yes	(null)	11

Using the SQL Worksheet

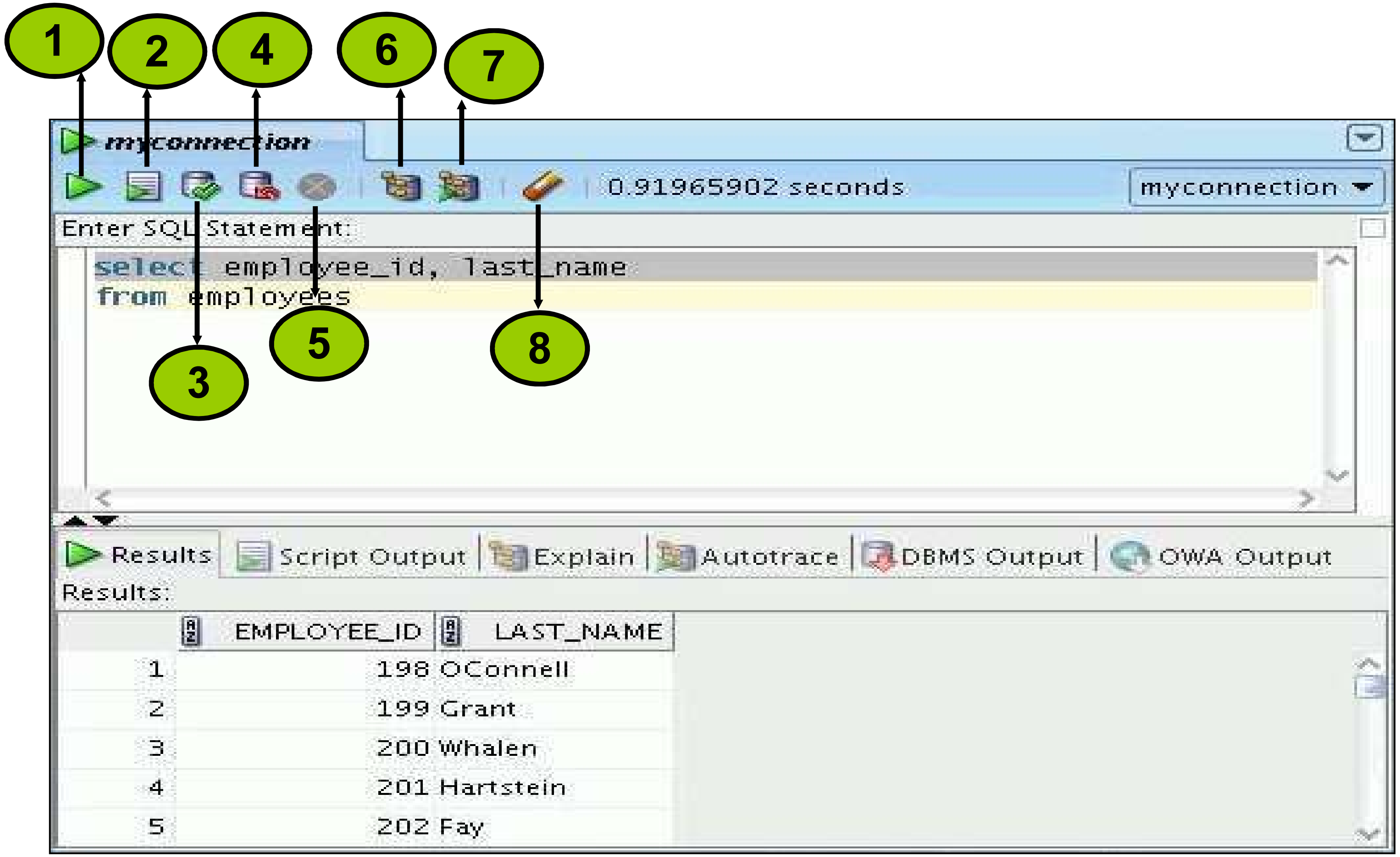
- Use the SQL Worksheet to enter and execute SQL, PL/SQL, and SQL*Plus statements.
- Specify any actions that can be processed by the database connection associated with the Worksheet.



Enter SQL Statements.

Results are shown here.

Using the SQL Worksheet

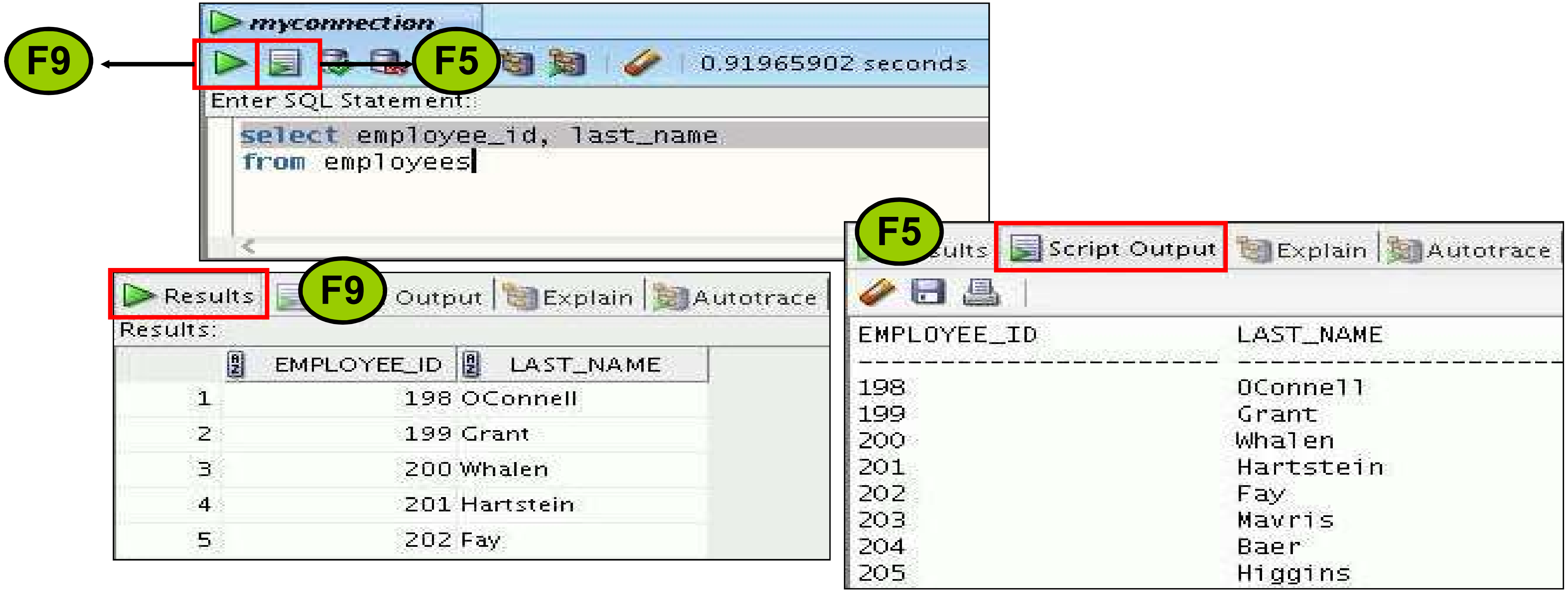


```
select employee_id, last_name
from employees
```

	EMPLOYEE_ID	LAST_NAME
1	198	OConnell
2	199	Grant
3	200	Whalen
4	201	Hartstein
5	202	Fay

Executing SQL Statements

Use the Enter SQL Statement box to enter single or multiple SQL statements.



```
select employee_id, last_name
from employees
```

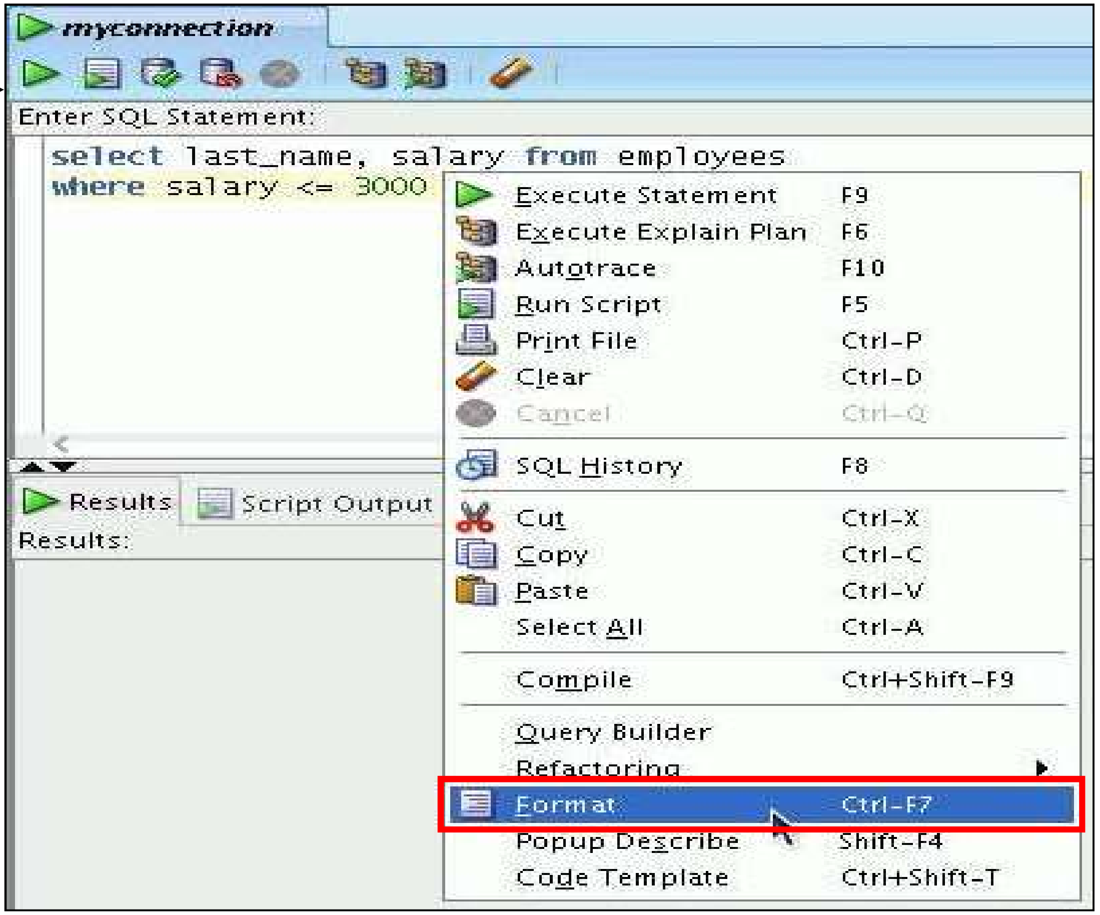
Results:

	EMPLOYEE_ID	LAST_NAME
1	198	OConnell
2	199	Grant
3	200	Whalen
4	201	Hartstein
5	202	Fay

```
EMPLOYEE_ID            LAST_NAME
---------------------- -------------------
198                    OConnell
199                    Grant
200                    Whalen
201                    Hartstein
202                    Fay
203                    Mavris
204                    Baer
205                    Higgins
```

Formatting the SQL Code

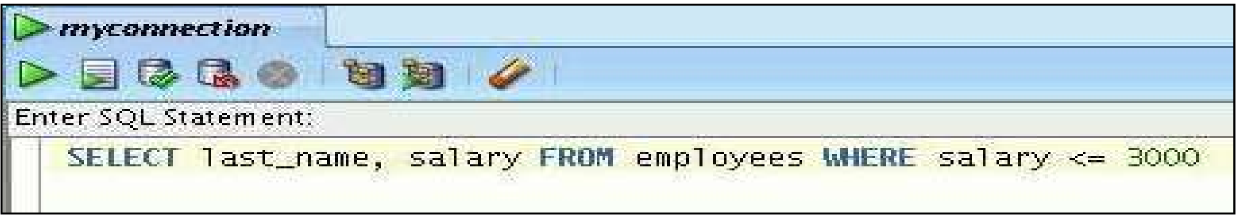
Before formatting



Enter SQL Statement:

```
select last_name, salary from employees
where salary <= 3000
```

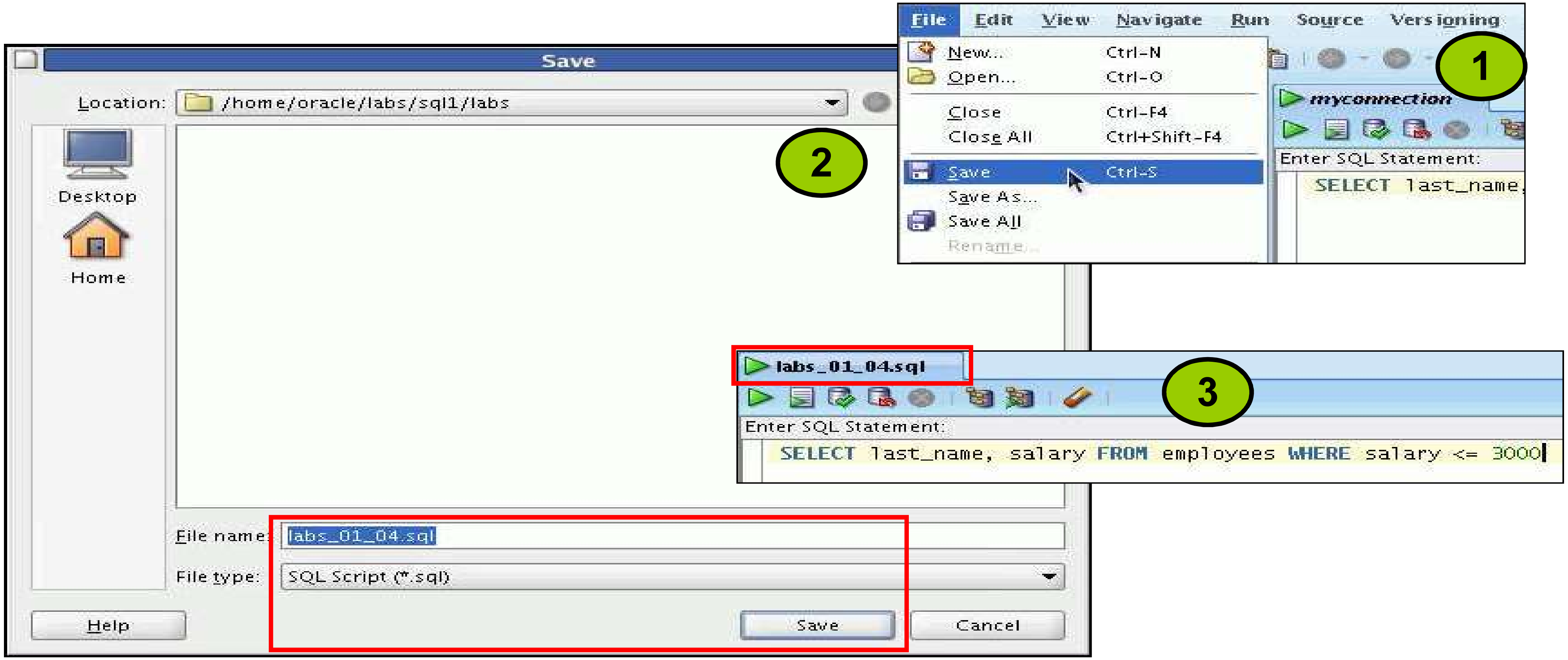
After formatting



Enter SQL Statement:

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

Saving SQL Statements



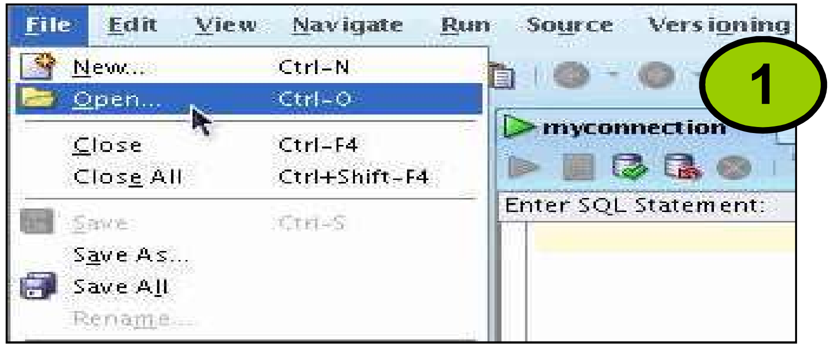
Location: /home/oracle/labs/sql1/labs

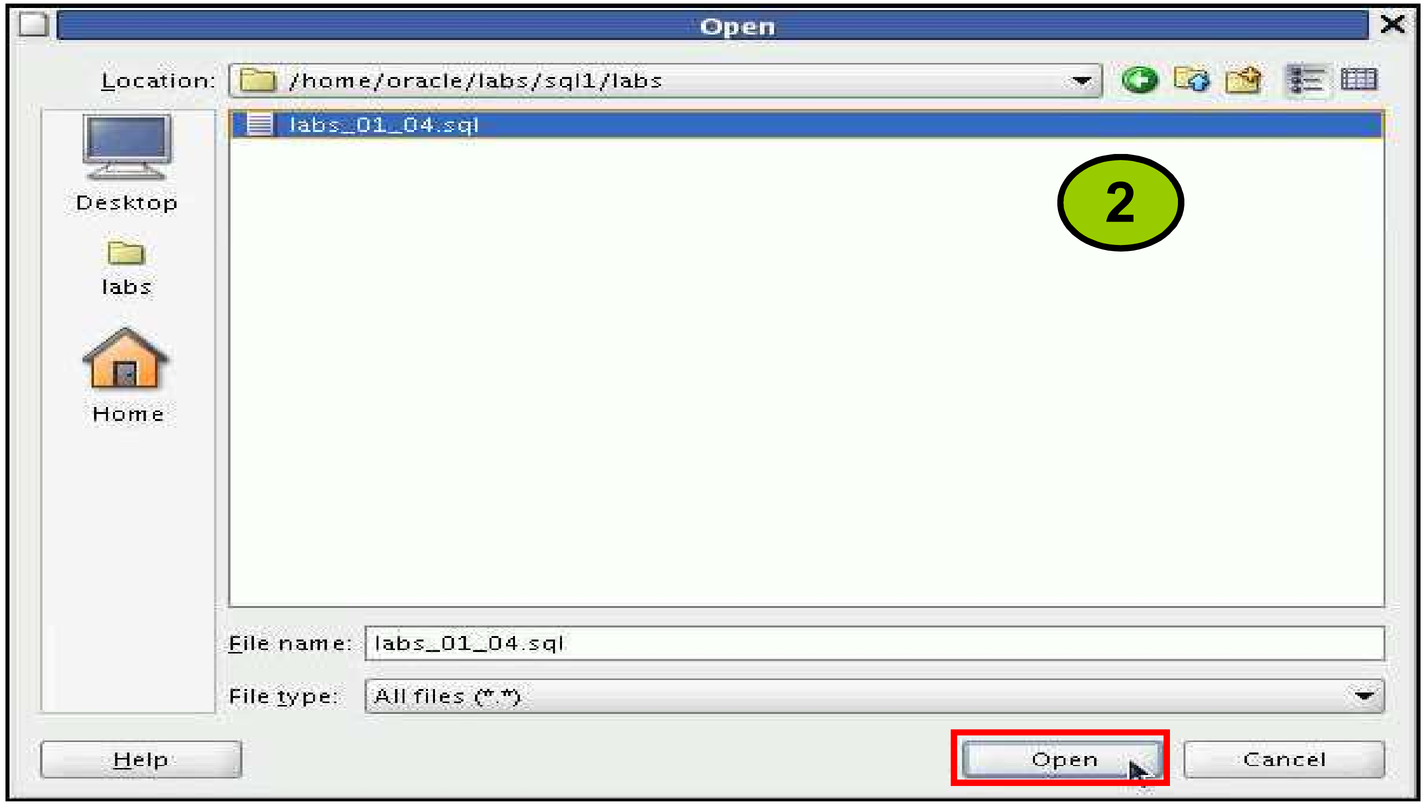
File name: labs_01_04.sql

File type: SQL Script (*.sql)

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

Running Script Files

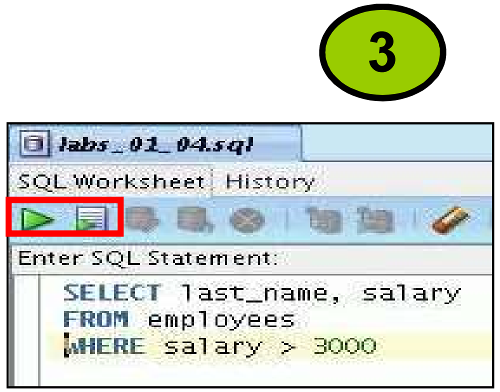




Location: /home/oracle/labs/sql1/labs

File name: labs_01_04.sql

File type: All files (*.*)



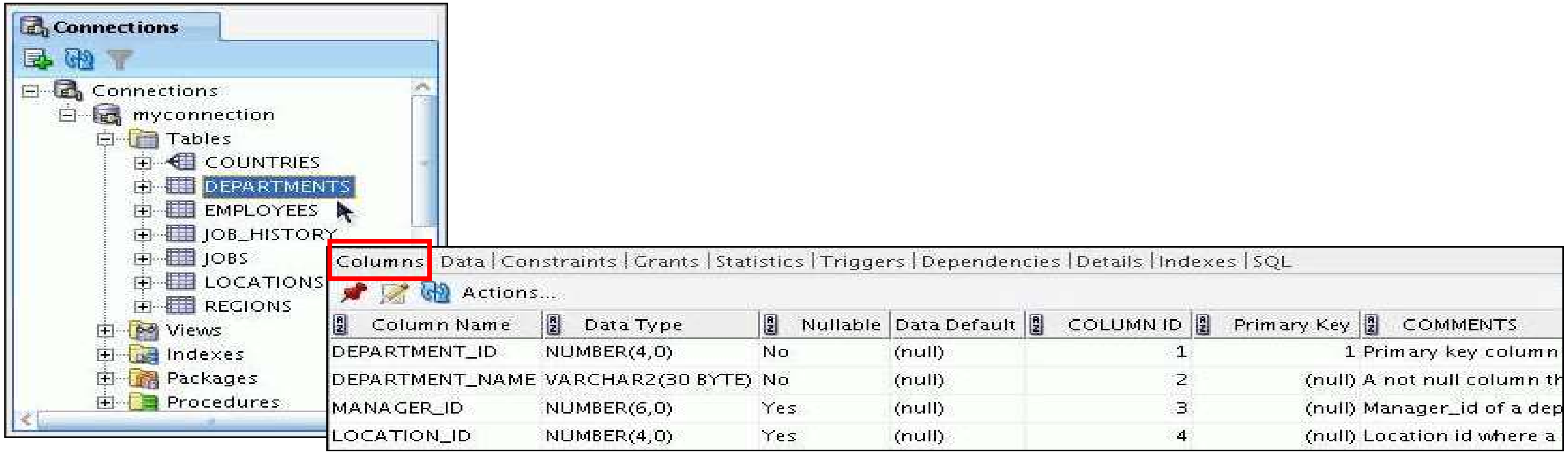
```
SELECT last_name, salary
FROM employees
WHERE salary > 3000
```

Displaying the Table Structure

- Use the `DESCRIBE` command to display the structure of a table.

```
DESC[RIBE] tablename
```

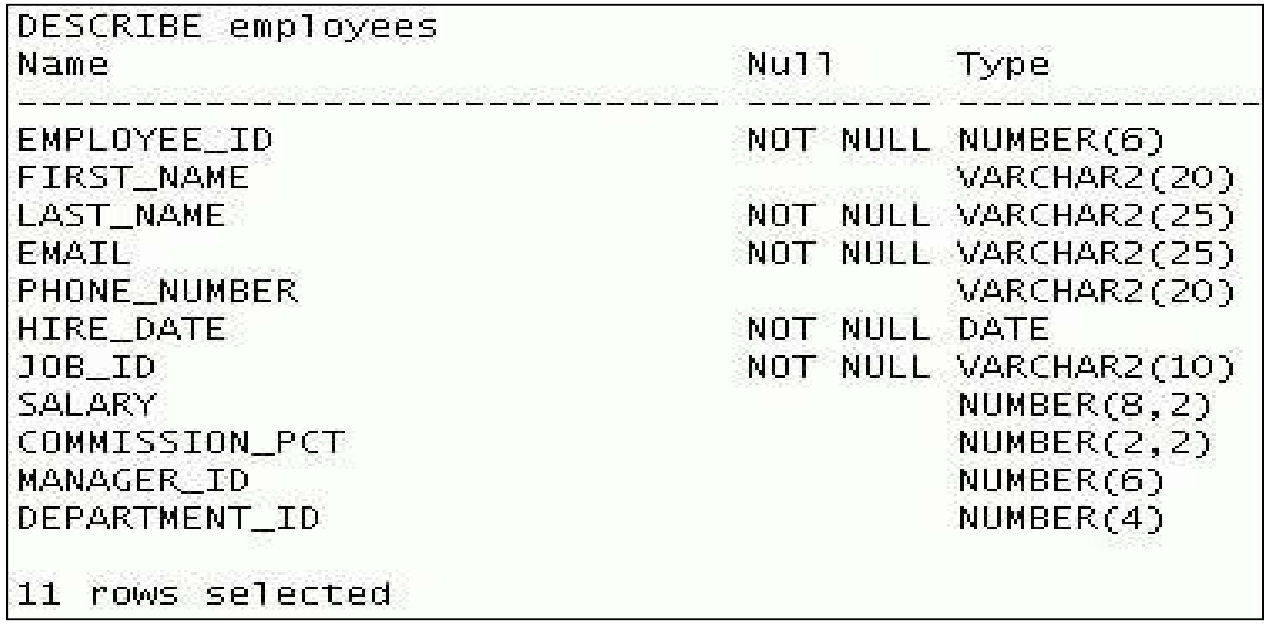
- Or, select the table in the Connections tree and use the Columns tab to view the table structure.



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_NAME	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

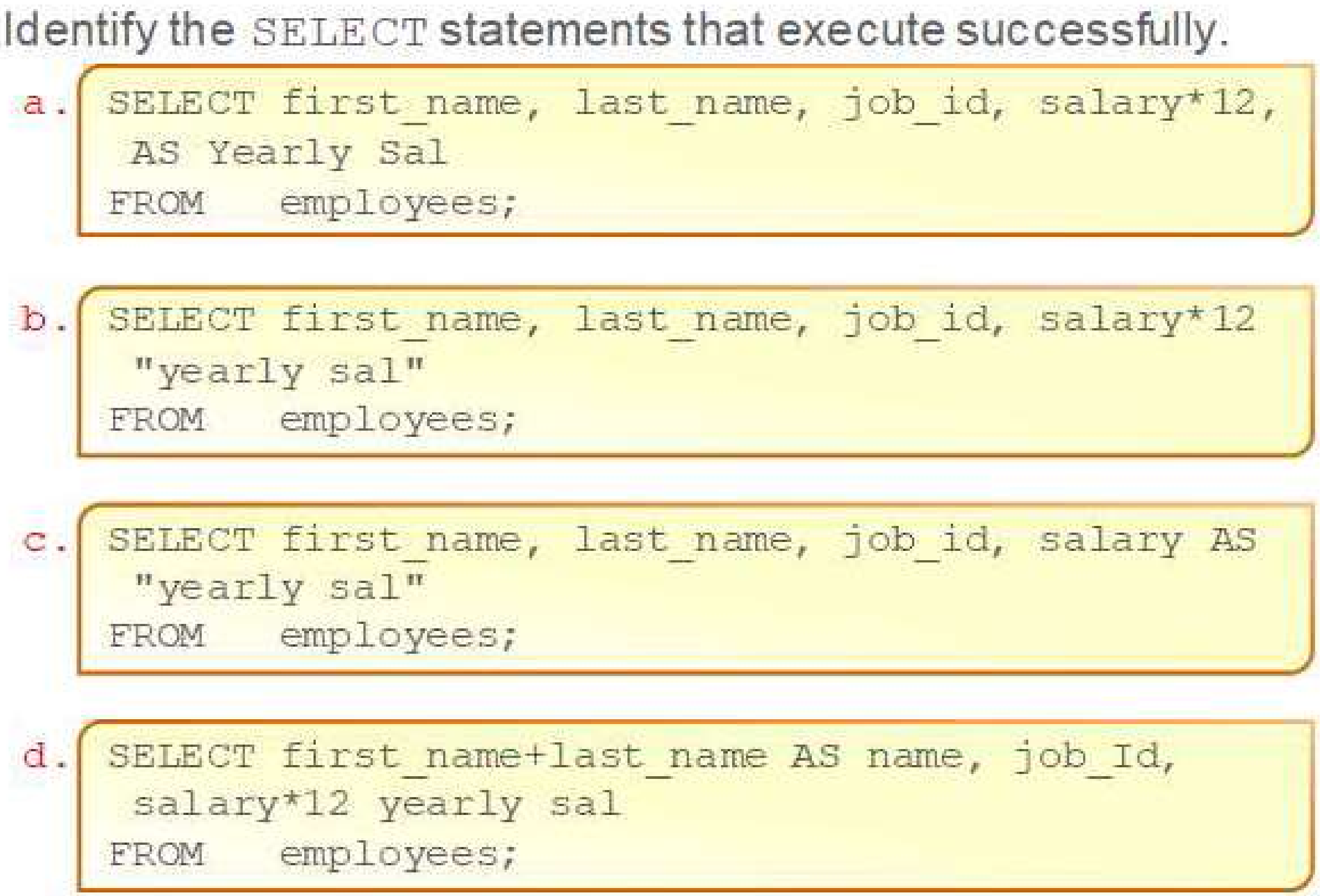
Using the DESCRIBE Command

```
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)

11 rows selected
```

Quiz



Identify the `SELECT` statements that execute successfully.

a.
```
SELECT first_name, last_name, job_id, salary*12,
 AS Yearly Sal
FROM   employees;
```

b.
```
SELECT first_name, last_name, job_id, salary*12
 "yearly sal"
FROM   employees;
```

c.
```
SELECT first_name, last_name, job_id, salary AS
 "yearly sal"
FROM   employees;
```

d.
```
SELECT first_name+last_name AS name, job_Id,
 salary*12 yearly sal
FROM   employees;
```

Summary

In this lesson, you should have learned how to:

- Write a SELECT statement that:
  - Returns all rows and columns from a table
  - Returns specified columns from a table
  - Uses column aliases to display more descriptive column headings

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

Practice 1: Overview

This practice covers the following topics:

- Using SQL Developer
- Selecting all data from different tables
- Describing the structure of tables
- Performing arithmetic calculations and specifying column names