1

Retrieving Data Using the SQL SELECT Statement

ORACLE

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

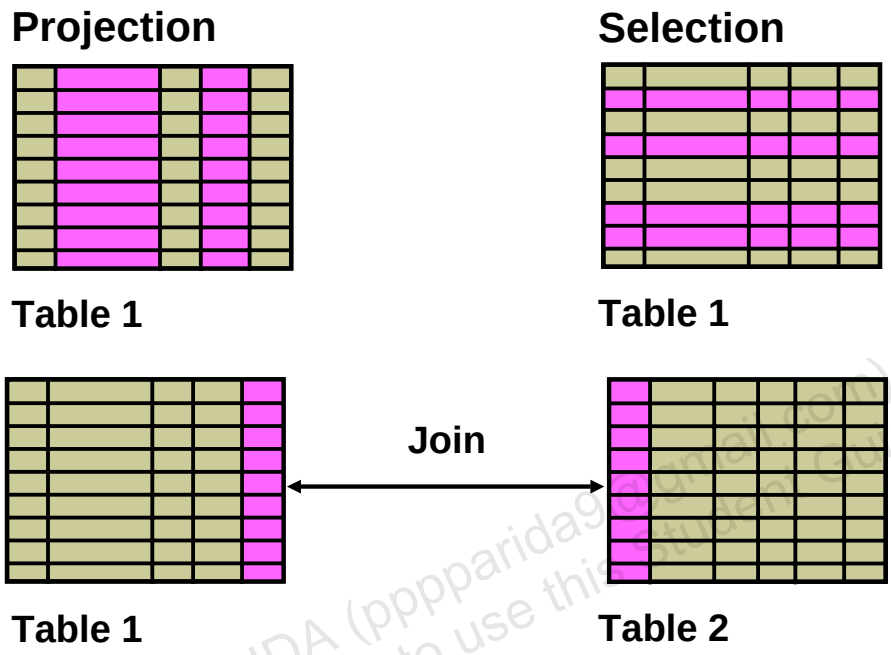
ORACLE

Objectives

To extract data from the database, you need to use the structured query language (SQL) SELECT statement. You may need to restrict the columns that are displayed. This lesson describes all the SQL statements that are needed to perform these actions. You may want to create SELECT statements that can be used more than once.

This lesson also covers the SQL Developer environment in which you execute SQL statements.

Capabilities of SQL SELECT Statements



Capabilities of SQL SELECT Statements

A SELECT statement retrieves information from the database. With a SELECT statement, you can use the following capabilities:

- **Projection:** Choose the columns in a table that are returned by a query. Choose as few or as many of the columns as needed.
- **Selection:** Choose the rows in a table that are returned by a query. Various criteria can be used to restrict the rows that are retrieved.
- **Joining:** Bring together data that is stored in different tables by specifying the link between them. SQL joins are covered in more detail in the lesson titled “Displaying Data from Multiple Tables.”

Basic SELECT Statement

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

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

ORACLE

Basic SELECT Statement

In its simplest form, a SELECT statement must include the following:

- A SELECT clause, which specifies the columns to be displayed
- A FROM clause, which identifies the table containing the columns that are listed in the SELECT clause

In the syntax:

SELECT	is a list of one or more columns
*	selects all columns
DISTINCT	suppresses duplicates
column|expression	selects the named column or the expression
alias	gives selected columns different headings
FROM *table*	specifies the table containing the columns

Note: Throughout this course, the words *keyword*, *clause*, and *statement* are used as follows:

- A *keyword* refers to an individual SQL element.
  For example, SELECT and FROM are keywords.
- A *clause* is a part of a SQL statement.
  For example, SELECT employee_id, last_name, ... is a clause.
- A *statement* is a combination of two or more clauses.
  For example, SELECT * FROM employees is a SQL statement.

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

ORACLE

Selecting All Columns of All Rows

You can display all columns of data in a table by following the SELECT keyword with an asterisk (*). In the example in the slide, the department table contains four columns: DEPARTMENT_ID, DEPARTMENT_NAME, MANAGER_ID, and LOCATION_ID. The table contains eight rows, one for each department.

You can also display all columns in the table by listing all the columns after the SELECT keyword. For example, the following SQL statement (like the example in the slide) displays all columns and all rows of the DEPARTMENTS table:

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

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

ORACLE

Selecting Specific Columns of All Rows

You can use the SELECT statement to display specific columns of the table by specifying the column names, separated by commas. The example in the slide displays all the department numbers and location numbers from the DEPARTMENTS table.

In the SELECT clause, specify the columns that you want, in the order in which you want them to appear in the output. For example, to display location before department number going from left to right, you use the following statement:

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

	LOCATION_ID	DEPARTMENT_ID
1	1700	10
2	1800	20
3	1500	50
4	1400	60

▪ ▪ ▪

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 (;).

ORACLE

Writing SQL Statements

Using the following simple rules and guidelines, you can construct valid statements that are both easy to read and easy to edit:

- SQL statements are not case-sensitive (unless indicated).
- SQL statements can be entered on one or many lines.
- Keywords cannot be split across lines or abbreviated.
- Clauses are usually placed on separate lines for readability and ease of editing.
- Indents should be used to make code more readable.
- Keywords typically are entered in uppercase; all other words, such as table names and columns, are entered in lowercase.

Executing SQL Statements

Using SQL Developer, click the Execute button to run the command or commands in the editing window.

Using SQL*Plus, terminate the SQL statement with a semicolon and then press the Enter key to run the command.

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

ORACLE

Column Heading Defaults

In SQL Developer, column headings are displayed in uppercase and centered.

```
SELECT last_name, hire_date, salary
FROM   employees;
```

	LAST_NAME	HIRE_DATE	SALARY
1	Whalen	17-SEP-87	4400
2	Hartstein	17-FEB-96	13000
3	Fay	17-AUG-97	6000
4	Higgins	07-JUN-94	12000
5	Gietz	07-JUN-94	8300
6	King	17-JUN-87	24000
7	Kochhar	21-SEP-89	17000
8	De Haan	13-JAN-93	17000

...

You can override the column heading display with an alias. Column aliases are covered later in this lesson.

Arithmetic Expressions

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

Operator	Description
+	Add
-	Subtract
*	Multiply
/	Divide

ORACLE

Arithmetic Expressions

You may need to modify the way in which data is displayed, or you may want to perform calculations or look at what-if scenarios. These are all possible using arithmetic expressions. An arithmetic expression can contain column names, constant numeric values, and the arithmetic operators.

Arithmetic Operators

The slide lists the arithmetic operators that are available in SQL. You can use arithmetic operators in any clause of a SQL statement (except the `FROM` clause).

Note: With the `DATE` and `TIMESTAMP` data types, you can use the addition and subtraction operators only.

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

...

Using Arithmetic Operators

The example in the slide uses the addition operator to calculate a salary increase of $300 for all employees. The slide also displays a `SALARY+300` column in the output.

Note that the resultant calculated column `SALARY+300` is not a new column in the `EMPLOYEES` table; it is for display only. By default, the name of a new column comes from the calculation that generated it—in this case, `salary+300`.

Note: The Oracle server ignores blank spaces before and after the arithmetic operator.

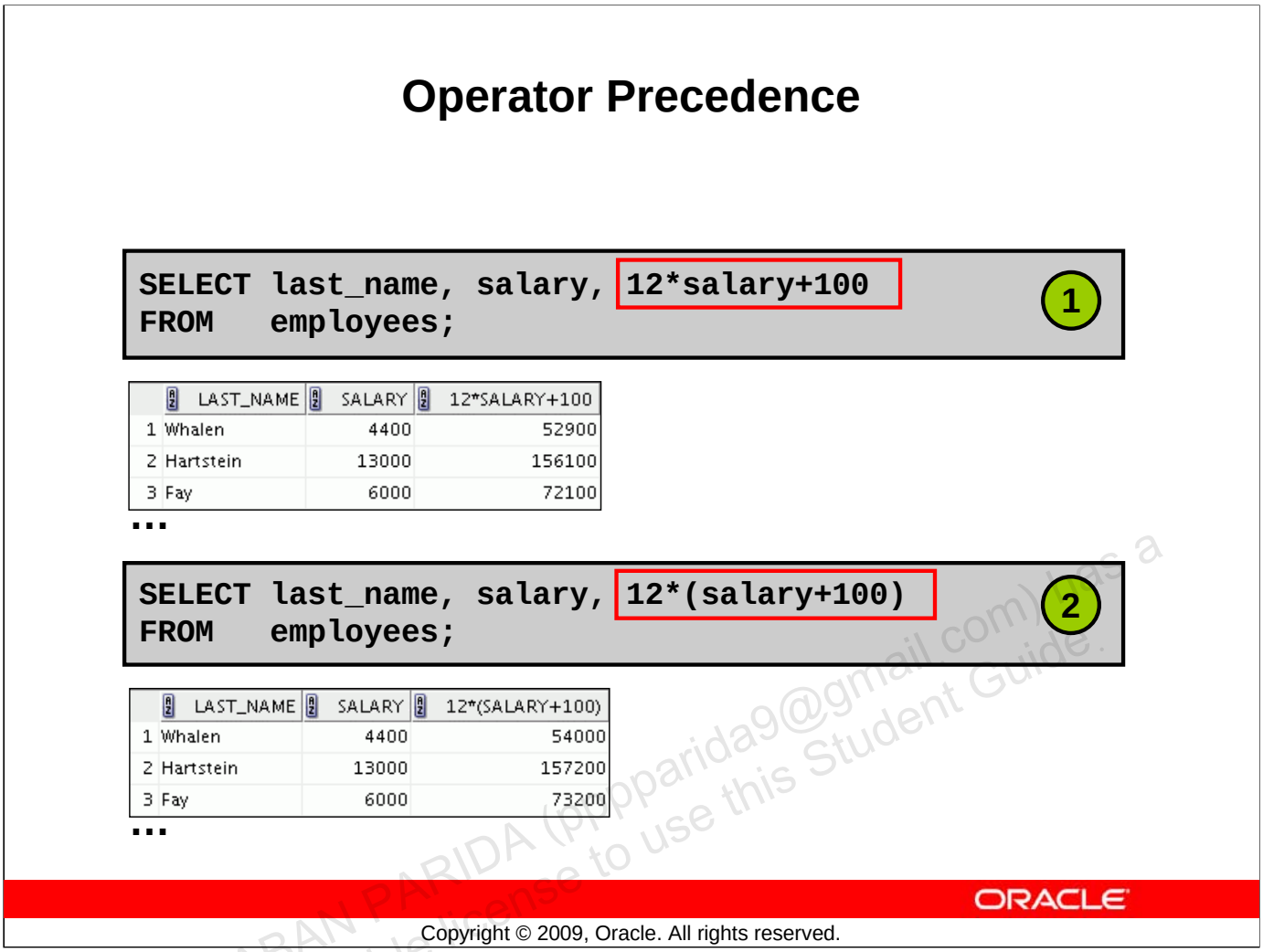
Operator Precedence

If an arithmetic expression contains more than one operator, multiplication and division are evaluated first. If operators in an expression are of the same priority, then evaluation is done from left to right.

You can use parentheses to force the expression that is enclosed by parentheses to be evaluated first.

Rules of Precedence:

- Multiplication and division occur before addition and subtraction.
- Operators of the same priority are evaluated from left to right.
- Parentheses are used to override the default precedence or to clarify the statement.



Operator Precedence (continued)

The first example in the slide displays the last name, salary, and annual compensation of employees. It calculates the annual compensation by multiplying the monthly salary by 12, plus a one-time bonus of $100. Note that multiplication is performed before addition.

Note: Use parentheses to reinforce the standard order of precedence and to improve clarity. For example, the expression in the slide can be written as `(12*salary)+100` with no change in the result.

Using Parentheses

You can override the rules of precedence by using parentheses to specify the desired order in which operators are to be executed.

The second example in the slide displays the last name, salary, and annual compensation of employees. It calculates the annual compensation as follows: adding a monthly bonus of $100 to the monthly salary, and then multiplying that subtotal by 12. Because of the parentheses, addition takes priority over multiplication.

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

ORACLE

Null Values

If a row lacks a data value for a particular column, that value is said to be *null* or to contain a null.

A null is a value that is unavailable, unassigned, unknown, or inapplicable. A null is not the same as a zero or a space. Zero is a number, and a space is a character.

Columns of any data type can contain nulls. However, some constraints (`NOT NULL` and `PRIMARY KEY`) prevent nulls from being used in the column.

In the `COMMISSION_PCT` column in the `EMPLOYEES` table, notice that only a sales manager or sales representative can earn a commission. Other employees are not entitled to earn commissions. A null represents that fact.

Null Values in Arithmetic Expressions

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

ORACLE

Null Values in Arithmetic Expressions

If any column value in an arithmetic expression is null, the result is null. For example, if you attempt to perform division by zero, you get an error. However, if you divide a number by null, the result is a null or unknown.

In the example in the slide, employee King does not get any commission. Because the `COMMISSION_PCT` column in the arithmetic expression is null, the result is null.

For more information, see “Basic Elements of SQL” in *SQL Reference*.

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

ORACLE

Column Aliases

When displaying the result of a query, SQL Developer normally uses the name of the selected column as the column heading. This heading may not be descriptive and, therefore, maybe difficult to understand. You can change a column heading by using a column alias.

Specify the alias after the column in the SELECT list using a space as a separator. By default, alias headings appear in uppercase. If the alias contains spaces or special characters (such as # or $), or if it is case-sensitive, enclose the alias in double quotation marks (" ").

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

...

ORACLE

Column Aliases (continued)

The first example displays the names and the commission percentages of all the employees. Notice that the optional AS keyword has been used before the column alias name. The result of the query is the same whether the AS keyword is used or not. Also notice that the SQL statement has the column aliases, `name` and `comm`, in lowercase, whereas the result of the query displays the column headings in uppercase. As mentioned in a previous slide, column headings appear in uppercase by default.

The second example displays the last names and annual salaries of all the employees. Because `Annual Salary` contains a space, it has been enclosed in double quotation marks. Notice that the column heading in the output is exactly the same as the column alias.

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

...

Concatenation Operator

You can link columns to other columns, arithmetic expressions, or constant values to create a character expression by using the *concatenation operator* (||). Columns on either side of the operator are combined to make a single output column.

In the example, `LAST_NAME` and `JOB_ID` are concatenated, and they are given the alias `Employees`. Notice that the employee last name and job code are combined to make a single output column.

The `AS` keyword before the alias name makes the `SELECT` clause easier to read.

Null Values with the Concatenation Operator

If you concatenate a null value with a character string, the result is a character string. `LAST_NAME || NULL` results in `LAST_NAME`.

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.

ORACLE

Literal Character Strings

A literal is a character, a number, or a date that is included in the SELECT list and that is not a column name or a column alias. It is printed for each row returned. Literal strings of free-format text can be included in the query result and are treated the same as a column in the SELECT list.

Date and character literals *must* be enclosed by single quotation marks (`' '`); number literals need not be so enclosed.

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

...

ORACLE

Literal Character Strings (continued)

The example in the slide displays last names and job codes of all employees. The column has the heading Employee Details. Notice the spaces between the single quotation marks in the SELECT statement. The spaces improve the readability of the output.

In the following example, the last name and salary for each employee are concatenated with a literal to give the returned rows more meaning:

```
SELECT last_name ||': 1 Month salary = '||salary Monthly
FROM   employees;
```

	MONTHLY
1	Whalen: 1 Month salary = 4400
2	Hartstein: 1 Month salary = 13000
3	Fay: 1 Month salary = 6000
4	Higgins: 1 Month salary = 12000
5	Gietz: 1 Month salary = 8300
6	King: 1 Month salary = 24000
7	Kochhar: 1 Month salary = 17000

...

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

...

ORACLE

Alternative Quote (q) Operator

Many SQL statements use character literals in expressions or conditions. If the literal itself contains a single quotation mark, you can use the quote (q) operator and choose your own quotation mark delimiter.

You can choose any convenient delimiter, single-byte or multibyte, or any of the following character pairs: [ ], { }, ( ), or < >.

In the example shown, the string contains a single quotation mark, which is normally interpreted as a delimiter of a character string. By using the q operator, however, the brackets [] are used as the quotation mark delimiter. The string between the brackets delimiters is interpreted as a literal character string.

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

...

ORACLE

Duplicate Rows

Unless you indicate otherwise, SQL Developer displays the results of a query without eliminating duplicate rows. The first example in the slide displays all the department numbers from the EMPLOYEES table. Notice that the department numbers are repeated.

To eliminate duplicate rows in the result, include the DISTINCT keyword in the SELECT clause immediately after the SELECT keyword. In the second example in the slide, the EMPLOYEES table actually contains 20 rows, but there are only seven unique department numbers in the table.

You can specify multiple columns after the DISTINCT qualifier. The DISTINCT qualifier affects all the selected columns, and the result is every distinct combination of the columns.

```
SELECT  DISTINCT department_id, job_id
FROM    employees;
```

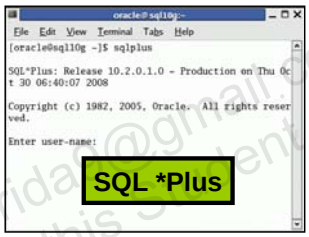
	DEPARTMENT_ID	JOB_ID
1	110	AC_ACCOUNT
2	90	AD_VP
3	50	ST_CLERK

...

Development Environments for SQL

In this course:

- Primarily use Oracle SQL Developer 1.5.1
- 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



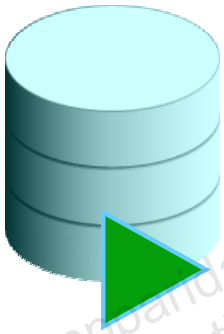
ORACLE

Development Environments for SQL

This course has been developed using Oracle SQL Developer as the tool for running the SQL statements discussed in the examples in the slide and the practices. For commands that are not supported by Oracle SQL Developer, use the SQL*Plus environment.

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

What Is Oracle SQL Developer?

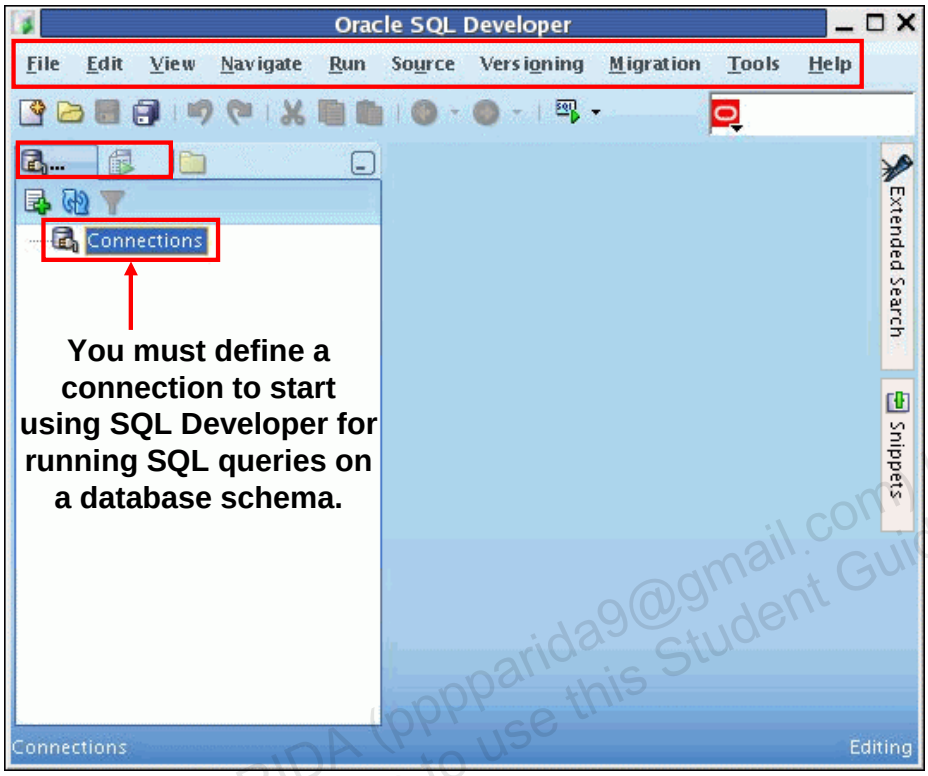
Oracle SQL Developer is a free graphical tool designed to improve your productivity and simplify the development of everyday database tasks. With just a few clicks, you can easily create and debug stored procedures, test SQL statements, and view optimizer plans.

Oracle SQL Developer, the visual tool for database development, simplifies the following tasks:

- Browsing and managing database objects
- Executing SQL statements and scripts
- Editing and debugging PL/SQL statements
- Creating reports

You can connect to any target Oracle Database schema by using the standard Oracle Database authentication. When connected, you can perform operations on objects in the database.

Oracle SQL Developer Interface



Oracle SQL Developer Interface

Oracle SQL Developer has two main navigation tabs:

- **Connections tab:** By using this tab, you can browse database objects and users to which you have access.
- **Reports tab:** By using this tab, you can run predefined reports, or create and add your own reports.

Oracle SQL Developer uses the left pane for navigation to find and select objects, and the right pane to display information about selected objects. You can customize many aspects of the appearance and behavior of Oracle SQL Developer by setting preferences. The menus at the top contain standard entries, plus entries for features specific to Oracle SQL Developer.

Note: You must define at least one connection to be able to connect to a database schema and issue SQL queries or run procedures/functions.

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.

ORACLE

Creating a Database Connection

A connection is an Oracle SQL Developer object that specifies the necessary information for connecting to a specific database as a specific user of that database. To use Oracle SQL Developer, you must have at least one database connection, which may be existing, created, or imported.

You can create and test connections for multiple databases and for multiple schemas.

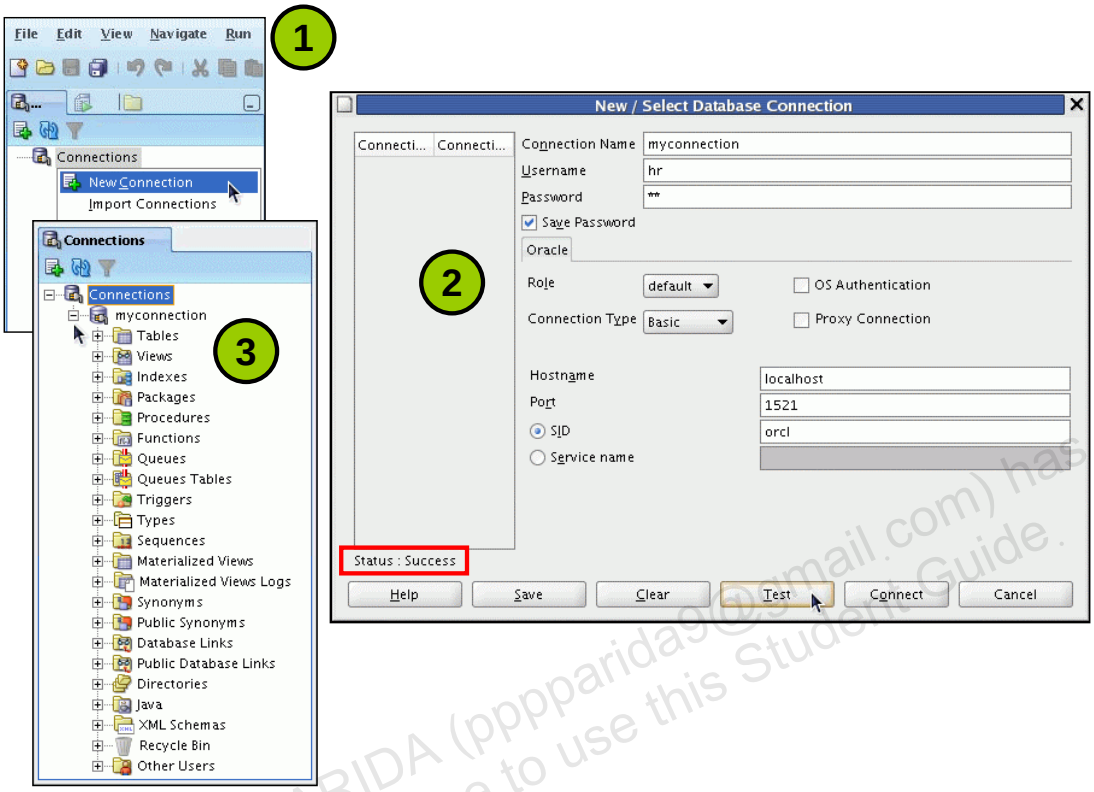
By default, the `tnsnames.ora` file is located in the `$ORACLE_HOME/network/admin` directory. But, it can also be in the directory specified by the `TNS_ADMIN` environment variable or the registry value. When you start Oracle SQL Developer and display the Database Connections dialog box, Oracle SQL Developer automatically imports any connections defined in the `tnsnames.ora` file on your system.

Note: On Windows systems, if the `tnsnames.ora` file exists but Oracle SQL Developer is not using its connections, define `TNS_ADMIN` as a system environment variable.

You can export connections to an XML file so that you can reuse it later.

You can create additional connections as different users to the same database or to connect to the different databases.

Creating a Database Connection



Creating a Database Connection (continued)

To create a database connection, perform the following steps:

1. On the Connections tabbed page, right-click Connections and select New Connection.
2. In the New/Select Database Connection window, enter the connection name. Enter the username and password of the schema that you want to connect to.
   1. From the Role drop-down list, you can select either *default* or SYSDBA (you will select SYSDBA for the `sys` user or any user with DBA privileges).
   2. You can select the connection type as:
      - **Basic:** In this type, you enter the host name and system identifier (SID) for the database that you want to connect to. The Port is already set to 1521. Or, you can also enter the Service name directly if you are using a remote database connection.
      - **TNS:** You select any one of the database aliases imported from the `tnsnames.ora` file
      - **Advanced:** You define a custom JDBC URL to connect to the database.
   3. Click Test to make sure that the connection has been set correctly.
   4. Click Connect.

If you select the Save Password check box, the password is saved to an XML file. So, after you close the Oracle SQL Developer connection and open it again, you will not be prompted for the password.

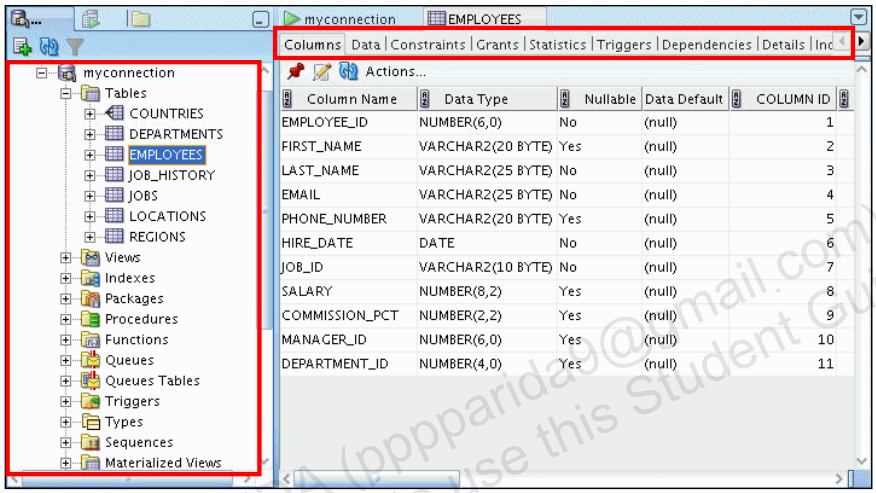
Creating a Database Connection (continued)

3. The connection gets added in the Connections Navigator. You can expand the connection to view the database objects and view object definitions, for example, dependencies, details, statistics, and so on.

Browsing Database Objects

Use the Connections Navigator to:

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



ORACLE

Browsing Database Objects

After you have created a database connection, you can use the Connections Navigator to browse through many objects in a database schema including Tables, Views, Indexes, Packages, Procedures, Triggers, Types, and so on.

Oracle SQL Developer uses the left pane for navigation to find and select objects and the right pane to display information about the selected objects. You can customize many aspects of the appearance of Oracle SQL Developer by setting preferences.

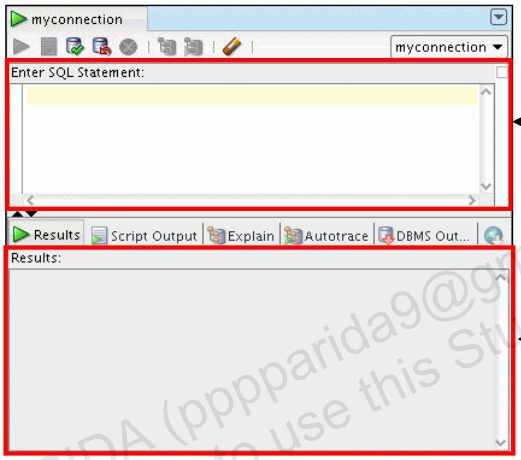
You can see the definition of the objects broken into tabs of information that is pulled out of the data dictionary. For example, if you select a table in the Navigator, the details about columns, constraints, grants, statistics, triggers, and so on are displayed in an easy-to-read tabbed page.

If you want to see the definition of the `EMPLOYEES` table as shown in the slide, perform the following steps:

1. Expand the Connections node in the Connections Navigator.
2. Expand Tables.
3. Click EMPLOYEES. By default, the Columns tab is selected. It shows the column description of the table. By using the Data tab, you can view the data in the table and also enter new rows, update data, and commit these changes to the database.

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.



Using the SQL Worksheet

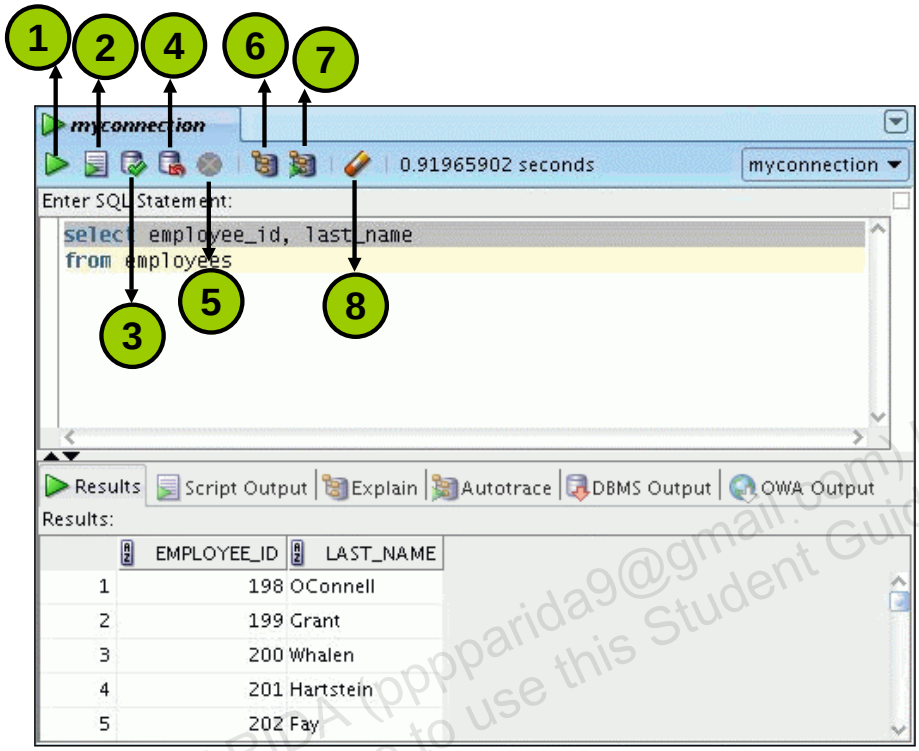
When you connect to a database, a SQL Worksheet window for that connection is automatically opened. You can use the SQL Worksheet to enter and execute SQL, PL/SQL, and SQL*Plus statements. All SQL and PL/SQL commands are supported as they are passed directly from the SQL Worksheet to the Oracle Database. However, the SQL*Plus commands used in Oracle SQL Developer have to be interpreted by the SQL Worksheet before being passed to the database.

The SQL Worksheet currently supports a number of SQL*Plus commands. Those commands that are not supported by the SQL Worksheet are ignored and not sent to the Oracle Database. Through the SQL Worksheet, you can execute SQL statements and some of the SQL*Plus commands.

You can display a SQL Worksheet by using any of the following two options:

- Select Tools > SQL Worksheet.
- Click the Open SQL Worksheet icon available on the main toolbar.

Using the SQL Worksheet



ORACLE

Using the SQL Worksheet (continued)

You may want to use shortcut keys or icons to perform certain tasks, such as executing a SQL statement, running a script, or viewing the history of the SQL statements that you have executed.

You can use the SQL Worksheet toolbar that contains icons to perform the following tasks:

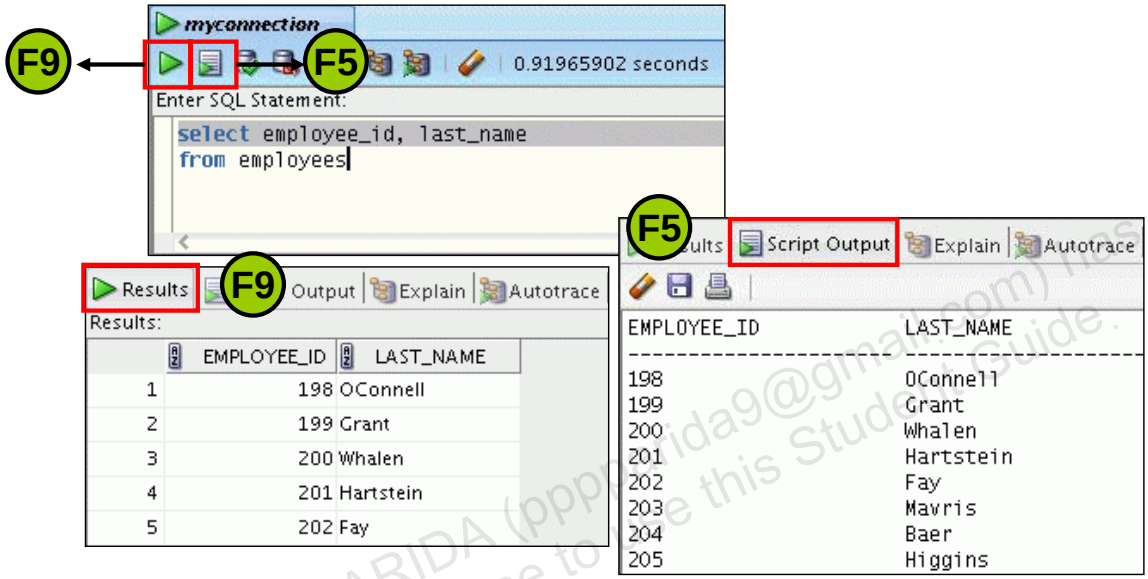
1. **Execute Statement:** This executes the statement at the cursor in the Enter SQL Statement box. Alternatively, you can press [F9]. The output is generally shown in a formatted manner in the Results tab page.
2. **Run Script:** This executes all statements in the Enter SQL Statement box using the Script Runner. The output is generally shown in the conventional script format in the Scripts tab page.
3. **Commit:** This writes any changes to the database and ends the transaction.
4. **Rollback:** This discards any changes to the database, without writing them to the database, and ends the transaction.
5. **Cancel:** This stops the execution of any statements currently being executed.
6. **Execute Explain Plan:** This generates the execution plan, which you can see by clicking the Explain tab.

Using the SQL Worksheet (continued)

7. **Autotrace:** This displays trace-related information when you execute the SQL statement by clicking the Autotrace icon. This information can help you to identify the SQL statements that will benefit from tuning.
8. **Clear:** This erases the statement or statements in the Enter SQL Statement box. Alternatively, press and hold [Ctrl] + [D] to erase the statements.

Executing SQL Statements

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



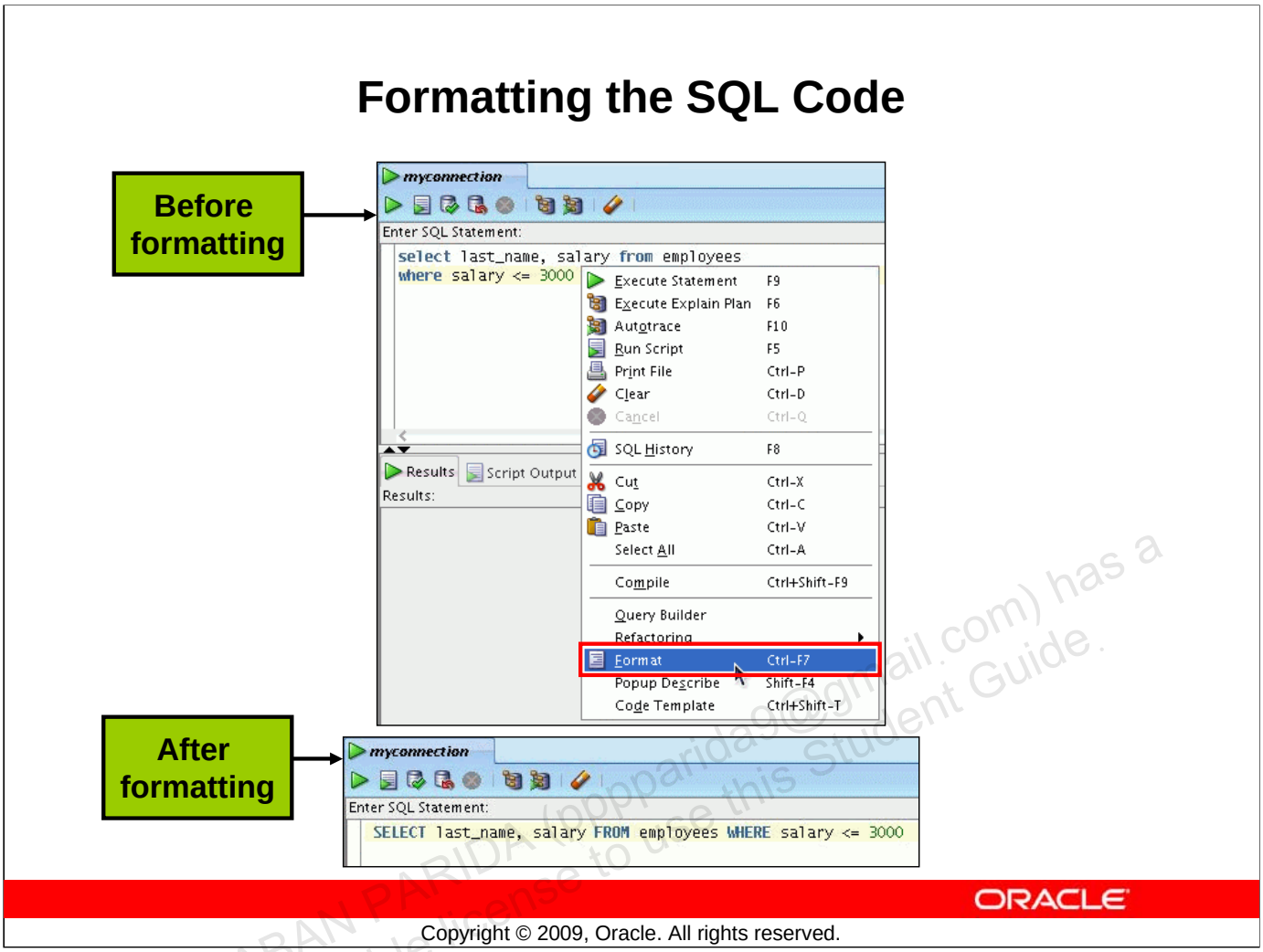
Executing SQL Statements

In the SQL Worksheet, you can use the Enter SQL Statement box to enter a single or multiple SQL statements. For a single statement, the semicolon at the end is optional.

When you enter the statement, the SQL keywords are automatically highlighted. To execute a SQL statement, ensure that your cursor is within the statement and click the Execute Statement icon. Alternatively, you can press [F9].

To execute multiple SQL statements and see the results, click the Run Script icon. Alternatively, you can press [F5].

The example in the slide shows the difference in output for the same query when F9 key or Execute Statement is used versus the output when F5 or Run Script is used.



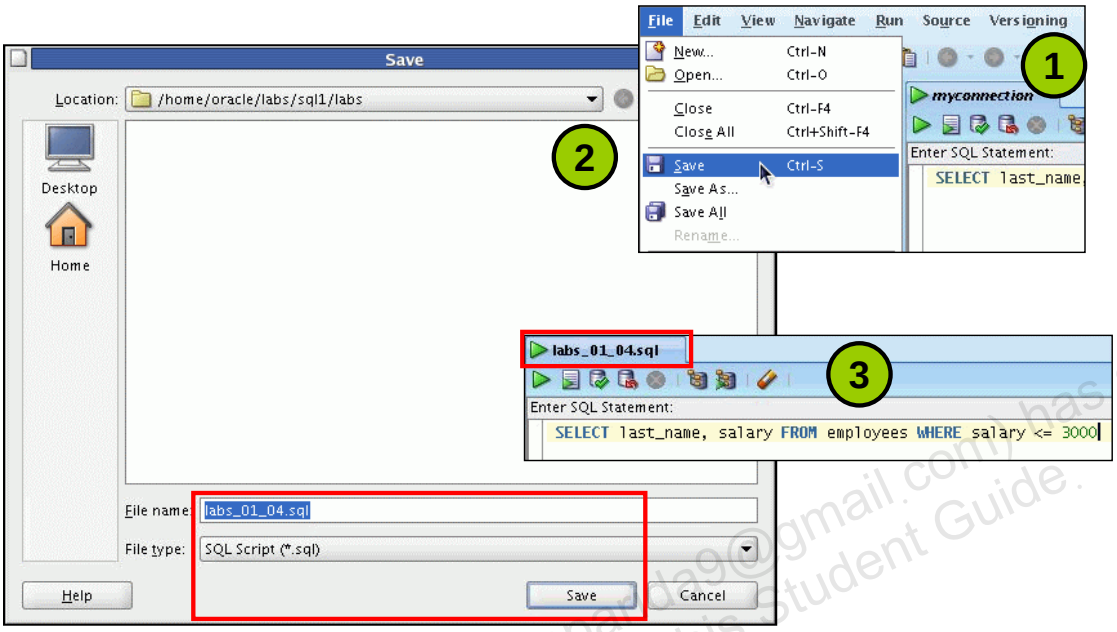
Formatting the SQL Code

You may want to beautify the indentation, spacing, capitalization, and the line separation of the SQL code. Oracle SQL Developer has the feature for formatting the SQL code.

To format the SQL code, right-click in the statement area, and select Format SQL.

In the example in the slide, before formatting, the keywords are not capitalized and the statement is not properly indented in the SQL code. After formatting, the SQL code is beautified with the keywords capitalized and the statement properly indented, if needed.

Saving SQL Statements

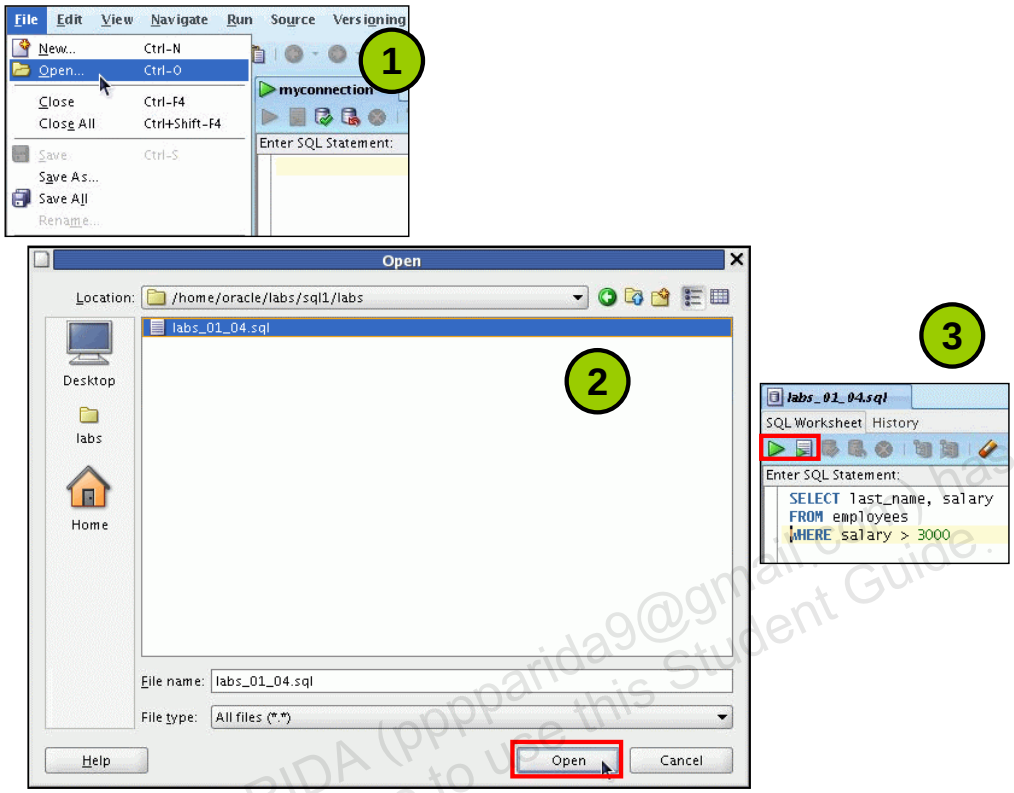


Saving SQL Statements

In most of the practices that you will perform, you will need to save a particular query in the SQL Worksheet as a `.sql` file. To do so, perform the following:

1. From the File menu, select Save or Save As (if you are renaming a current `.sql` script). Alternatively, you can press and hold [CTRL] + [S].
2. In the Save dialog box, enter the appropriate filename. Make sure the extension is `.sql` or the File type is selected as SQL Script (*.sql). Click Save.
   Note: For this course, you need to save your `sql` scripts in the `\home\oracle\labs\sql1\labs` folder.
3. The SQL Worksheet is renamed to the filename that you saved the script as. Make sure you do not enter any other SQL statements in the same worksheet. To continue with other SQL queries, open a new worksheet.

Running Script Files



ORACLE

Running Script Files

To run the saved `.sql` script files, perform the following:

1. From the File menu, select Open. Alternatively, you can press and hold [CTRL] + [O].
2. In the Open dialog box, move to the `\home\oracle\labs\sql1\labs` folder, or to the location in which you saved the script file, select the file and click Open.
3. The script file opens in a new worksheet. Now, you can run the script by either clicking the Execute Statement icon or the Run Script icon. Again, make sure you do not enter any other SQL statements in the same worksheet. To continue with other SQL queries, open a new worksheet.
   Note: You may want to set the default directory to `\home\oracle\labs\sql1` folder, so that every time you try to open or save a script, SQL Developer chooses the same path to look for scripts. From Tools menu, select Preferences. In the Preferences dialog box, expand Database and select Worksheet Parameters. In the right pane, click Browse to set the default path to look for scripts and click OK.

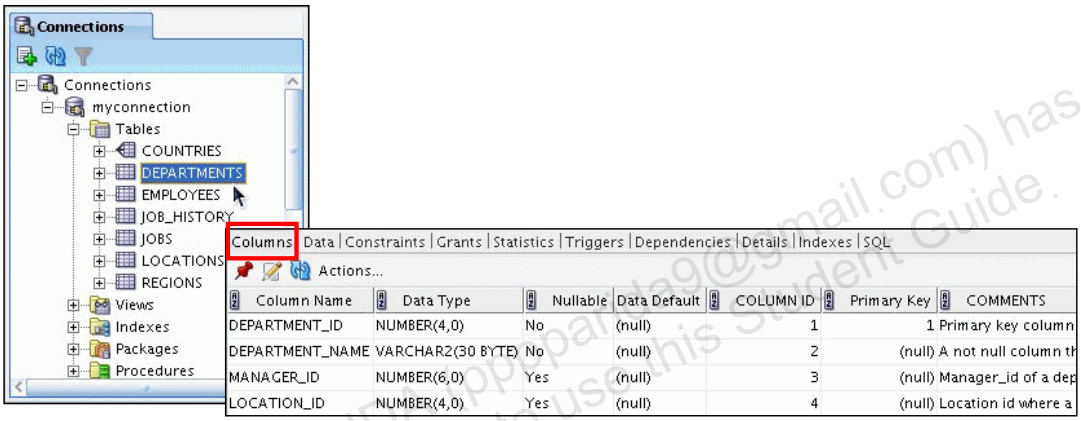
Note: For more details on how to use the Oracle SQL Developer GUI interface for other data objects creation and data retrieval tasks, refer to Appendix E “Using SQL Developer.”

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.



ORACLE

Displaying the Table Structure

In SQL Developer, you can display the structure of a table by using the DESCRIBE command. The command displays the column names and the data types, and it shows you whether a column *must* contain data (that is, whether the column has a NOT NULL constraint).

In the syntax, *table name* is the name of any existing table, view, or synonym that is accessible to the user.

Using the SQL Developer GUI interface, you can select the table in the Connections tree and use the Columns tab to view the table structure.

Note: The DESCRIBE command is supported by both SQL*Plus and SQL Developer.

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
```


ORACLE

Using the DESCRIBE Command

The example in the slide displays information about the structure of the EMPLOYEES table using the DESCRIBE command.

In the resulting display, *Null* indicates that the values for this column may be unknown. NOT NULL indicates that a column must contain data. *Type* displays the data type for a column.

The data types are described in the following table:

Data Type	Description
NUMBER(*p*,*s*)	Number value having a maximum number of digits *p*, with *s* digits to the right of the decimal point
VARCHAR2(*s*)	Variable-length character value of maximum size *s*
DATE	Date and time value between January 1, 4712 B.C. and December 31, A.D. 9999.
CHAR(*s*)	Fixed-length character value of size *s*

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
- Use the SQL Developer environment to write, save, and execute SQL statements

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


ORACLE

Summary

In this lesson, you should have learned how to retrieve data from a database table with the SELECT statement.

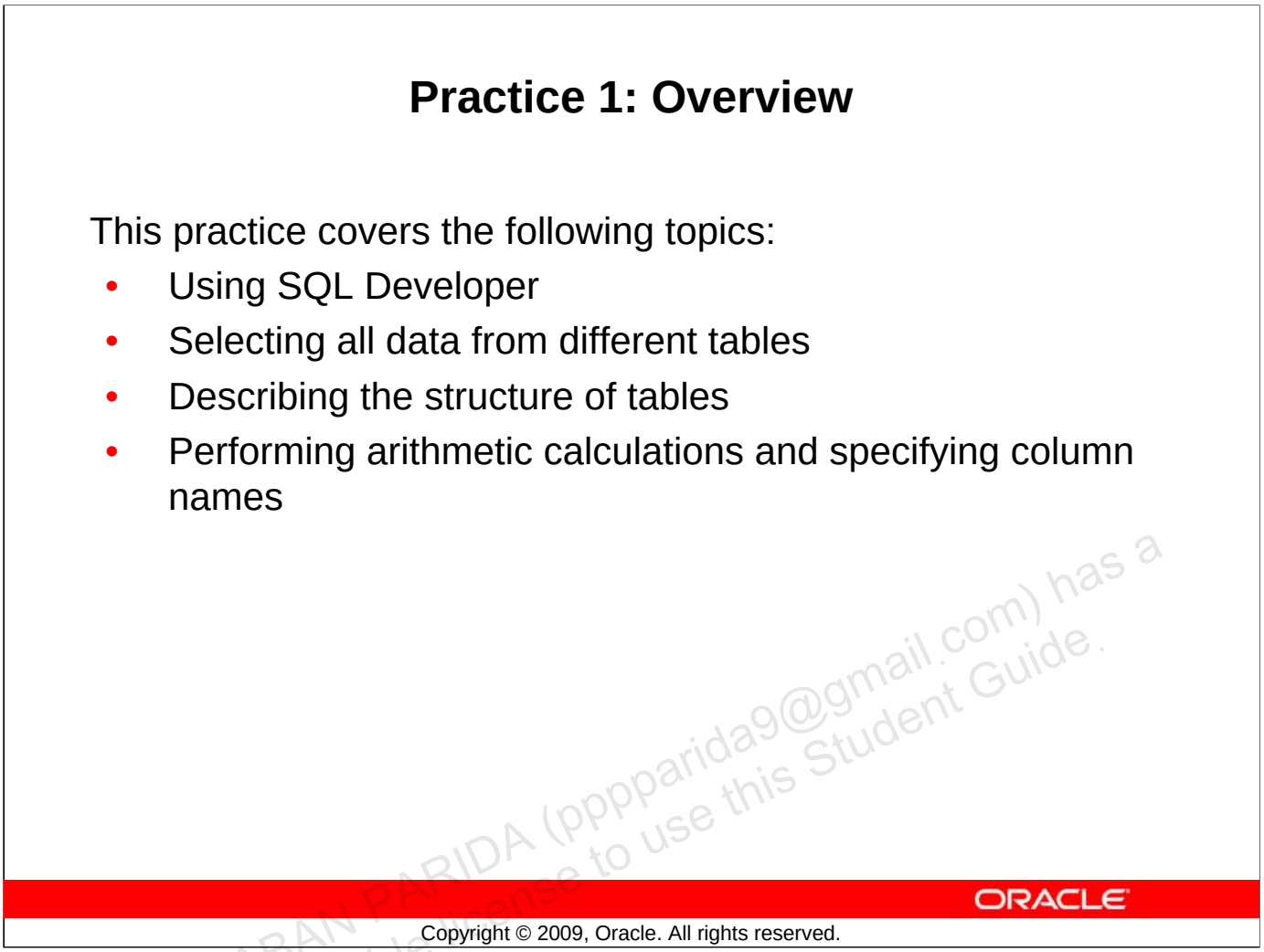
```
SELECT  *|{[DISTINCT] column [alias],...}
FROM    table;
```

In the syntax:

SELECT	is a list of one or more columns
*	selects all columns
DISTINCT	suppresses duplicates
column|*expression*	selects the named column or the expression
alias	gives selected columns different headings
FROM *table*	specifies the table containing the columns

SQL Developer

SQL Developer is an execution environment that you can use to send SQL statements to the database server and to edit and save SQL statements. Statements can be executed from the SQL prompt or from a script file.



Practice 1: Overview

This is the first of many practices in this course. The solutions (if you require them) can be found in Appendix A. The practices are intended to cover all the topics that are presented in the corresponding lesson.

Note the following location for the lab files:

`\home\oracle\labs\SQL1\labs`

If you are asked to save any lab files, save them at this location.

In any practice, there may be exercises that are prefaced with the phrases “If you have time” or “If you want an extra challenge.” Work on these exercises only if you have completed all other exercises in the allocated time and would like a further challenge to your skills.

Perform the practices slowly and precisely. You can experiment with saving and running command files. If you have any questions at any time, ask your instructor.

Note: All written practices use Oracle SQL Developer as the development environment. Although it is recommended that you use Oracle SQL Developer, you can also use SQL*Plus that is available in this course.

Practice 1

Part 1

Test your knowledge:

1. The following `SELECT` statement executes successfully:

```
SELECT last_name, job_id, salary AS Sal
FROM   employees;
```

   True/False

2. The following `SELECT` statement executes successfully:

```
SELECT *
FROM   job_grades;
```

   True/False

3. There are four coding errors in the following statement. Can you identify them?

```
SELECT    employee_id, last_name
sal x 12  ANNUAL SALARY
FROM      employees;
```

Part 2

Note the following location for the lab files:

`\home\oracle\labs\SQL1\labs`

If you are asked to save any lab files, save them at this location.

To start Oracle SQL Developer, double-click the SQL Developer desktop icon.

Before you begin with the practices, you need a database connection to be able to connect to the database and issue SQL queries.

4. To create a new database connection in the Connections Navigator, right-click Connections. Select New Connection from the shortcut menu. The New/Select Database Connection dialog box appears.
5. Create a database connection using the following information:
   a. Connection Name: `myconnection`
   b. Username: `ora1`
   c. Password: `ora1`
   d. Hostname: localhost
   e. Port: 1521
   f. SID: ORCL
   g. Ensure that you select the Save Password check box.

Practice 1 (continued)

You have been hired as a SQL programmer for Acme Corporation. Your first task is to create some reports based on data from the Human Resources tables.

6. Your first task is to determine the structure of the DEPARTMENTS table and its contents.

```
DESCRIBE departments
Name                           Null     Type
------------------------------ -------- -------------
DEPARTMENT_ID                  NOT NULL NUMBER(4)
DEPARTMENT_NAME                NOT NULL VARCHAR2(30)
MANAGER_ID                              NUMBER(6)
LOCATION_ID                             NUMBER(4)

4 rows selected
```

	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

7. You need to determine the structure of the EMPLOYEES table.

```
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
```

The HR department wants a query to display the last name, job code, hire date, and employee number for each employee, with the employee number appearing first. Provide an alias STARTDATE for the HIRE_DATE column. Save your SQL statement to a file named lab_01_07.sql so that you can dispatch this file to the HR department.

Practice 1 (continued)

8. Test your query in the `lab_01_07.sql` file to ensure that it runs correctly.

	EMPLOYEE_ID	LAST_NAME	JOB_ID	STARTDATE
1	200	Whalen	AD_ASST	17-SEP-87
2	201	Hartstein	MK_MAN	17-FEB-96
3	202	Fay	MK_REP	17-AUG-97
4	205	Higgins	AC_MGR	07-JUN-94
5	206	Gietz	AC_ACCOUNT	07-JUN-94
6	100	King	AD_PRES	17-JUN-87
7	101	Kochhar	AD_VP	21-SEP-89
8	102	De Haan	AD_VP	13-JAN-93
9	103	Hunold	IT_PROG	03-JAN-90
10	104	Ernst	IT_PROG	21-MAY-91
11	107	Lorentz	IT_PROG	07-FEB-99
12	124	Mourgos	ST_MAN	16-NOV-99
13	141	Rajs	ST_CLERK	17-OCT-95
14	142	Davies	ST_CLERK	29-JAN-97
15	143	Matos	ST_CLERK	15-MAR-98
16	144	Vargas	ST_CLERK	09-JUL-98
17	149	Zlotkey	SA_MAN	29-JAN-00
18	174	Abel	SA_REP	11-MAY-96
19	176	Taylor	SA_REP	24-MAR-98
20	178	Grant	SA_REP	24-MAY-99

9. The HR department needs a query to display all unique job codes from the `EMPLOYEES` table.

	JOB_ID
1	AC_ACCOUNT
2	AC_MGR
3	AD_ASST
4	AD_PRES
5	AD_VP
6	IT_PROG
7	MK_MAN
8	MK_REP
9	SA_MAN
10	SA_REP
11	ST_CLERK
12	ST_MAN

Practice 1 (continued)

Part 3

If you have time, complete the following exercises:

10. The HR department wants more descriptive column headings for its report on employees. Copy the statement from `lab_01_07.sql` to the SQL Developer text box. Name the column headings `Emp #`, `Employee`, `Job`, and `Hire Date`, respectively. Then run your query again.

	Emp #	Employee	Job	Hire Date
1	200	Whalen	AD_ASST	17-SEP-87
2	201	Hartstein	MK_MAN	17-FEB-96
3	202	Fay	MK_REP	17-AUG-97
4	205	Higgins	AC_MGR	07-JUN-94
5	206	Gietz	AC_ACCOUNT	07-JUN-94
6	100	King	AD_PRES	17-JUN-87
7	101	Kochhar	AD_VP	21-SEP-89
8	102	De Haan	AD_VP	13-JAN-93
9	103	Hunold	IT_PROG	03-JAN-90
10	104	Ernst	IT_PROG	21-MAY-91

▪ ▪ ▪

20	178	Grant	SA_REP	24-MAY-99

11. The HR department has requested a report of all employees and their job IDs. Display the last name concatenated with the job ID (separated by a comma and space) and name the column `Employee and Title`.

	Employee and Title
1	Abel, SA_REP
2	Davies, ST_CLERK
3	De Haan, AD_VP
4	Ernst, IT_PROG
5	Fay, MK_REP
6	Gietz, AC_ACCOUNT
7	Grant, SA_REP
8	Hartstein, MK_MAN
9	Higgins, AC_MGR
10	Hunold, IT_PROG

▪ ▪ ▪

20	Zlotkey, SA_MAN

Practice 1 (continued)

If you want an extra challenge, complete the following exercise:

12. To familiarize yourself with the data in the EMPLOYEES table, create a query to display all the data from that table. Separate each column output with a comma. Name the column THE_OUTPUT.

	THE_OUTPUT
1	200,Jennifer,Whalen,JWHALEN,515.123.4444,AD_ASST,101,17-SEP-87,4400,,10
2	201,Michael,Hartstein,MHARTSTE,515.123.5555,MK_MAN,100,17-FEB-96,13000,,20
3	202,Pat,Fay,PFAY,603.123.6666,MK_REP,201,17-AUG-97,6000,,20
4	205,Shelley,Higgins,SHIGGINS,515.123.8080,AC_MGR,101,07-JUN-94,12000,,110
5	206,William,Gietz,WGIETZ,515.123.8181,AC_ACCOUNT,205,07-JUN-94,8300,,110
6	100,Steven,King,SKING,515.123.4567,AD_PRES,,17-JUN-87,24000,,90
7	101,Neena,Kochhar,NKOCHHAR,515.123.4568,AD_VP,100,21-SEP-89,17000,,90
8	102,Lex,De Haan,LDEHAAN,515.123.4569,AD_VP,100,13-JAN-93,17000,,90
9	103,Alexander,Hunold,AHUNOLD,590.423.4567,IT_PROG,102,03-JAN-90,9000,,60
10	104,Bruce,Ernst,BERNST,590.423.4568,IT_PROG,103,21-MAY-91,6000,,60

▪ ▪ ▪

20	178,Kimberely,Grant,KGRANT,011.44.1644.429263,SA_REP,149,24-MAY-99,7000,.15,