2

Connecting to Oracle Database and Exploring It

You can connect to Oracle Database only through a client program, such as SQL*Plus or SQL Developer. When connected to the database, you can view schema objects, view the properties and data of Oracle Database tables, and use queries to retrieve data from Oracle Database tables.

After connecting to Oracle Database through a client program, you enter and run commands in that client program. For details, see the documentation for your client program.

Connecting to Oracle Database from SQL*Plus

SQL*Plus is a client program from which you can access Oracle Database. This topic shows how to start SQL*Plus and connect to Oracle Database.

> **Note:**
>
> For steps 3 and 4 of the following procedure, you need a user name and password.

To connect to Oracle Database from SQL*Plus:

1. If you are on a Windows system, display a Windows command prompt.
2. At the command prompt, type `sqlplus` and then press the key **Enter**.
3. At the user name prompt, type your user name and then press the key **Enter**.
4. At the password prompt, type your password and then press the key **Enter**.

   > **Note:**
   >
   > For security, your password is not visible on your screen.

   The system connects you to an Oracle Database instance.

   You are in the SQL*Plus environment. At the `SQL>` prompt, you can enter and run SQL*Plus commands, SQL statements, PL/SQL statements, and operating system commands.

   To exit SQL*Plus, type `exit` and press the key **Enter**.

> **Note:**
>
> Exiting SQL*Plus ends the SQL*Plus session, but does not shut down the Oracle Database instance.

Example 2-1 starts SQL*Plus, connects to Oracle Database, runs a SQL `SELECT` statement, and exits SQL*Plus. User input is **bold**.

Example 2-1 Connecting to Oracle Database from SQL*Plus

```
> sqlplus
SQL*Plus: Release 12.1.0.1.0 Production on Thu Dec 27 07:43:41 2012

Copyright (c) 1982, 2012, Oracle.  All rights reserved.

Enter user-name: your_user_name
Enter password: your_password

Connected to:
Oracle Database 12c Enterprise Edition Release - 12.1.0.1.0 64bit Production

SQL> select count(*) from employees;

  COUNT(*)
----------
       107

SQL> exit

Disconnected from Oracle Database 12c Enterprise Edition Release - 12.1.0.1.0 64bit Production
>
```

> **See Also:**
>
> - "Connecting to Oracle Database as User HR from SQL*Plus"
> - "About SQL*Plus" for a brief description of SQL*Plus
> - *SQL*Plus User's Guide and Reference* for more information about starting SQL*Plus and connecting to Oracle Database

Connecting to Oracle Database from SQL Developer

SQL Developer is a client program with which you can access Oracle Database.

You are encouraged to use the currently available release of SQL Developer, which you can download from:

http://www.oracle.com/technetwork/developer-tools/sql-developer/downloads/

This section assumes that SQL Developer is installed on your system, and shows how to start it and connect to Oracle Database. If SQL Developer is not installed on your system, then see *Oracle SQL Developer User's Guide* for installation instructions.

> **Note:**
>
> For the following procedure:
>
> - If you're using a SQL Developer kit that does not include the JDK, then the first time you start SQL Developer on your system, you must provide the path to the Java Development Kit.
> - When prompted, you need to enter a user name and password.

To connect to Oracle Database from SQL Developer:

1. Start SQL Developer.

   For instructions, see *Oracle SQL Developer User's Guide*.

   If this is the first time you have started SQL Developer on your system, you are prompted to enter the path to the Java Development Kit (JDK) installation (for example, `C:\Program Files\Java\jdk1.8.0_65`). Either type the path after the prompt or browse to it, and then press the key **Enter**.

2. In the Connections frame, click the icon **New Connection**.

3. In the New/Select Database Connection window:

   a. Type the appropriate values in the fields Connection Name, Username, and Password.

      For security, the password characters that you type appear as asterisks.

      Near the Password field is the check box Save Password. By default, it is deselected. Oracle recommends accepting the default.

   b. If the Oracle pane is not showing, click the tab **Oracle**.

   c. In the Oracle pane, accept the default values.

      (The default values are: Connection Type, Basic; Role, default, Hostname, localhost; Port, 1521; SID option, selected; SID field, xe.)

   d. Click the button **Test**.

      The connection is tested. If the connection succeeds, the Status indicator changes from blank to `Success`.

   e. If the test succeeded, click the button **Connect**.

      The New/Select Database Connection window closes. The Connections frame shows the connection whose name you entered in the Connection Name field in step 3.

You are in the SQL Developer environment.

To exit SQL Developer, select **Exit** from the File menu.

> **Note:**
>
> Exiting SQL Developer ends the SQL Developer session, but does not shut down the Oracle Database instance. The next time you start SQL Developer, the connection you created using the preceding procedure still exists. SQL Developer prompts you for the password that you supplied in step 3 (unless you selected the check box Save Password).

> **See Also:**
>
> - "Connecting to Oracle Database as User HR from SQL Developer"
> - "About SQL Developer" for a brief description of SQL Developer
> - *Oracle SQL Developer User's Guide* for more information about using SQL Developer to create connections to Oracle Database

Connecting to Oracle Database as User HR

To complete the tutorials and examples in this document, you must install the `hr` sample schema and connect to Oracle Database as the user `HR`.

The user `HR` owns the `hr` sample schema that the examples and tutorials in this document use.

> **See Also:**
>
> - Installing the Sample Schemas in *Database Sample Schemas* for information about how to install the `hr` schema

Unlocking the HR Account

You must unlock the HR account and reset its password before you can connect to Oracle Database as the user HR.

By default, when the HR schema is installed, the HR account is locked and its password is expired.

> **Note:**
>
> For the following procedure, you need the name and password of a user who has the ALTER USERsystem privilege.

To unlock the HR account and reset its password:

1. Using SQL*Plus, connect to Oracle Database as a user with the ALTER USER system privilege.
2. At the `SQL>` prompt, unlock the HR account and reset its password:

> **Caution:**
>
> Choose a secure password. For guidelines for secure passwords, see *Oracle Database Security Guide*.

```
ALTER USER HR ACCOUNT UNLOCK IDENTIFIED BY password;
```

The system responds:

```
User altered.
```

The `HR` account is unlocked and its password is `password`.

Now you can connect to Oracle Database as user HR with the password `password`.

> **See Also:**
>
> - *Oracle SQL Developer User's Guide* for information about accessing SQL*Plus within SQL Developer

Connecting to Oracle Database as User HR from SQL*Plus

You can use SQL*Plus to connect to Oracle Database as the HR user.

> **Note:**
>
> If the HR account is locked, see "Unlocking the HR Account" and then return to this section.

To connect to Oracle Database as user HR from SQL*Plus:

> **Note:**
>
> For this task, you need the password for the HR account.

1. If you are connected to Oracle Database, close your current connection.

2. Follow the directions in "Connecting to Oracle Database from SQL*Plus", entering the user name `HR` at step 3 and the password for the HR account at step 4.

   You are now connected to Oracle Database as the user HR.

> **See Also:**
>
> *SQL*Plus User's Guide and Reference* for an example of using SQL*Plus to create an `HR` connection

Connecting to Oracle Database as User HR from SQL Developer

You can use SQL Developer to connect to Oracle Database as the HR user.

> **Note:**
>
> If the HR account is locked, see "Unlocking the HR Account" and then return to this section.

To connect to Oracle Database as user HR from SQL Developer:

> **Note:**
>
> For this task, you need the password for the HR account.

Follow the directions in "Connecting to Oracle Database from SQL Developer", entering the following values at steps 3:

- For Connection Name, enter `hr_conn`.

  (You can enter a different name, but the tutorials in this document assume that you named the connection `hr_conn`.)
- For Username, enter `HR`.
- For Password, enter the password for the HR account.

You are now connected to Oracle Database as the user HR.

Exploring Oracle Database with SQL*Plus

If the `hr` sample schema is installed and you are connected to Oracle Database from SQL*Plus as the user HR, you can view HR schema objects and the properties of the EMPLOYEES table.

> **Note:**
>
> If you are not connected to Oracle Database as user HR from SQL*Plus, see "Connecting to Oracle Database as User HR from SQL*Plus" and then return to this section.

Viewing HR Schema Objects with SQL*Plus

With SQL*Plus, you can view the objects that belong to the HR schema by querying the static data dictionary view USER_OBJECTS.

Example 2-2 shows how to view the names and data types of the objects that belong to the HR schema.

Example 2-2 Viewing HR Schema Objects with SQL*Plus

```
COLUMN OBJECT_NAME FORMAT A25
COLUMN OBJECT_TYPE FORMAT A25

SELECT OBJECT_NAME, OBJECT_TYPE FROM USER_OBJECTS
ORDER BY OBJECT_TYPE, OBJECT_NAME;
```

Result is similar to:

```
OBJECT_NAME               OBJECT_TYPE
------------------------- -------------------------
COUNTRY_C_ID_PK           INDEX
DEPT_ID_PK                INDEX
DEPT_LOCATION_IX          INDEX
EMP_DEPARTMENT_IX         INDEX
EMP_EMAIL_UK              INDEX
EMP_EMP_ID_PK             INDEX
EMP_JOB_IX                INDEX
EMP_MANAGER_IX            INDEX
EMP_NAME_IX               INDEX
JHIST_DEPARTMENT_IX       INDEX
JHIST_EMPLOYEE_IX         INDEX
JHIST_EMP_ID_ST_DATE_PK   INDEX
JHIST_JOB_IX              INDEX
JOB_ID_PK                 INDEX
LOC_CITY_IX               INDEX
LOC_COUNTRY_IX            INDEX
LOC_ID_PK                 INDEX
LOC_STATE_PROVINCE_IX     INDEX
REG_ID_PK                 INDEX
ADD_JOB_HISTORY           PROCEDURE
SECURE_DML                PROCEDURE
DEPARTMENTS_SEQ           SEQUENCE
EMPLOYEES_SEQ             SEQUENCE
LOCATIONS_SEQ             SEQUENCE
COUNTRIES                 TABLE
DEPARTMENTS               TABLE
EMPLOYEES                 TABLE
JOBS                      TABLE
JOB_HISTORY               TABLE
LOCATIONS                 TABLE
REGIONS                   TABLE
```

```
SECURE_EMPLOYEES          TRIGGER
UPDATE_JOB_HISTORY        TRIGGER
EMP_DETAILS_VIEW          VIEW

34 rows selected.
```

> **See Also:**
>
> - *Oracle Database Reference* for information about USER_OBJECTS
> - "Selecting Table Data" for information about using queries to view table data
> - "About Sample Schema HR" for general information about the schema HR

Viewing EMPLOYEES Table Properties and Data with SQL*Plus

You can a SQL*Plus command, the SQL SELECTstatement, and static data dictionary views to view the properties and data of the HR.EMPLOYEES table.

You can use the SQL*Plus command `DESCRIBE` to view the properties of the columns of the EMPLOYEES table in the HR schema and the SQL statement SELECT to view the data. To view other properties of the table, use static data dictionary views (for example, USER_CONSTRAINTS, USER_INDEXES, and USER_TRIGGERS).

Example 2-3 shows how to view the properties of the EMPLOYEES table in the HR schema.

Example 2-3 Viewing EMPLOYEES Table Properties with SQL*Plus

```
DESCRIBE EMPLOYEES
```

Result:

```
 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)
```

Example 2-4 shows how to view some data in the EMPLOYEES table in the HR schema.

Example 2-4 Viewing EMPLOYEES Table Data with SQL*Plus

```
COLUMN FIRST_NAME FORMAT A20
COLUMN LAST_NAME FORMAT A25
```

```
COLUMN PHONE_NUMBER FORMAT A20

SELECT LAST_NAME, FIRST_NAME, PHONE_NUMBER FROM EMPLOYEES
ORDER BY LAST_NAME;
```

Result is similar to:

```
LAST_NAME                 FIRST_NAME           PHONE_NUMBER
------------------------- -------------------- --------------------
Abel                      Ellen                011.44.1644.429267
Ande                      Sundar               011.44.1346.629268
Atkinson                  Mozhe                650.124.6234
Austin                    David                590.423.4569
Baer                      Hermann              515.123.8888
Baida                     Shelli               515.127.4563
Banda                     Amit                 011.44.1346.729268
Bates                     Elizabeth            011.44.1343.529268
...
Urman                     Jose Manuel          515.124.4469
Vargas                    Peter                650.121.2004
Vishney                   Clara                011.44.1346.129268
Vollman                   Shanta               650.123.4234
Walsh                     Alana                650.507.9811
Weiss                     Matthew              650.123.1234
Whalen                    Jennifer             515.123.4444
Zlotkey                   Eleni                011.44.1344.429018

107 rows selected.
```

See Also:

- *SQL*Plus User's Guide and Reference* for information about `DESCRIBE`
- "Selecting Table Data" for information about using queries to view table data
- *Oracle Database Reference* for information about static data dictionary views

Exploring Oracle Database with SQL Developer

If the `hr` sample schema is installed and you are connected to Oracle Database from SQL Developer as the user HR, you can view HR schema objects and the properties of the EMPLOYEES table.

Tutorial: Viewing HR Schema Objects with SQL Developer

This tutorial shows how to use SQL Developer to view the objects that belong to the HR schema—that is, how to **browse** the HR schema.

> **Note:**
>
> If you are not connected to Oracle Database as user HR from SQL Developer, see "Connecting to Oracle Database as User HR from SQL Developer" and then return to this tutorial.

To browse the HR schema:

1. In the Connections frame, to the left of the hr_conn icon, click the **plus sign (+)**.

   If you are not connected to the database, the Connection Information window opens. If you are connected to the database, the hr_conn information expands (see the information that follows "Click OK" in step 2).

2. If the Connection Information window opens:

   a. In the User Name field, enter `hr`.

   b. In the Password field, enter the password for the user HR.

   c. Click **OK**.

   The hr_conn information expands: The plus sign becomes a minus sign (-), and under the hr_conn icon, a list of schema object types appears—Tables, Views, Indexes, and so on. (If you click the minus sign, the hr_conn information collapses: The minus sign becomes a plus sign, and the list disappears.)

> **See Also:**
>
> - *Oracle SQL Developer User's Guide* for more information about the SQL Developer user interface
> - "About Sample Schema HR" for general information about schema HR

Tutorial: Viewing EMPLOYEES Table Properties and Data with SQL Developer

This tutorial shows how to use SQL Developer to view the properties and data of the EMPLOYEES table in the HR schema.

> **Note:**
>
> If you are not browsing the HR schema, see "Tutorial: Viewing HR Schema Objects with SQL Developer" and then return to this tutorial.

To view the properties and data of the EMPLOYEES table:

1. In the Connections frame, expand **Tables**.

   Under Tables, a list of the tables in the HR schema appears.

2. Select the table **EMPLOYEES**.

   In the right frame of the Oracle SQL Developer window, in the Columns pane, a list of all columns of this table appears. To the right of each column are its properties—name, data type, and so on. (To see all column properties, move the horizontal scroll bar to the right.)

3. In the right frame, click the tab **Data**.

   The Data pane appears, showing a numbered list of all records in this table. (To see more records, move the vertical scroll bar down. To see more columns of the records, move the horizontal scroll bar to the right.)

4. In the right frame, click the tab **Constraints**.

   The Constraints pane appears, showing a list of all constraints on this table. To the right of each constraint are its properties—name, type, search condition, and so on. (To see all constraint properties, move the horizontal scroll bar to the right.)

5. Explore the other properties by clicking on the appropriate tabs.

   To see the SQL statement for creating the EMPLOYEES table, click the **SQL** tab. The SQL statement appears in a pane named EMPLOYEES. To close this pane, click the **x** to the right of the name EMPLOYEES.

> **See Also:**
>
> *Oracle SQL Developer User's Guide* for more information about the SQL Developer user interface

Selecting Table Data

> **Note:**
>
> To do the tutorials and examples in this section, the `hr` sample schema must be installed and you must be connected to Oracle Database as the user HR from SQL Developer. For instructions, see "Connecting to Oracle Database as User HR from SQL Developer".

About Queries

A **query**, or SQL SELECT statement, selects data from one or more tables or views.

The simplest form of query has this syntax:

```
SELECT select_list FROM source_list
```

The select_list specifies the columns from which the data is to be selected, and the source_list specifies the tables or views that have these columns.

A query nested within another SQL statement is called a **subquery**.

In the SQL*Plus environment, you can enter a query (or any other SQL statement) after the `SQL>` prompt.

In the SQL Developer environment, you can enter a query (or any other SQL statement) in the Worksheet.

> **Note:**
>
> When the result of a query is displayed, records can be in any order, unless you specify their order with the ORDER BY clause. For more information, see "Sorting Selected Data".

> **See Also:**
>
> - *Oracle Database SQL Language Reference* for more information about queries and subqueries
> - *Oracle Database SQL Language Reference* for more information about the SELECT statement
> - *SQL*Plus User's Guide and Reference* for more information about the SQL*Plus command line interface
> - *Oracle SQL Developer User's Guide* for information about using the Worksheet in SQL Developer

Running Queries in SQL Developer

This section explains how to run queries in SQL Developer, using the Worksheet.

> **Note:**
>
> The Worksheet is not limited to queries; you can use it to run any SQL statement.

To run queries in SQL Developer:

1. If the right frame of SQL Developer shows the hr_conn pane:
   a. If the Worksheet subpane does not show, click the tab **Worksheet**.
   b. Go to step 4.
2. Click the icon **SQL Worksheet**.
3. If the Select Connection window opens:
   a. If the Connection field does not have the value `hr_conn`, select that value from the menu.
   b. Click **OK**.

   A pane appears with a tab labeled hr_conn and two subpanes, Worksheet and Query Builder. In the Worksheet, you can enter a SQL statement.
4. In the Worksheet, type a query (a SELECT statement).
5. Click the icon **Run Statement**.

   The query runs. Under the Worksheet, the Query Result pane appears, showing the query result.
6. Under the hr_conn tab, click the icon **Clear**.

   The query disappears, and you can enter another SQL statement in the Worksheet. When you run another SQL statement, its result appears in the Query Result pane, replacing the result of the previously run SQL statement.

> **See Also:**
>
> *Oracle SQL Developer User's Guide* for information about using the Worksheet in SQL Developer

Tutorial: Selecting All Columns of a Table

This tutorial shows how to select all columns of the EMPLOYEES table.

To select all columns of the EMPLOYEES Table:

1. If a pane with the tab hr_conn is there, select it. Otherwise, click the icon **SQL Worksheet**, as in "Running Queries in SQL Developer".

2. In the Worksheet, enter this query:

```
SELECT * FROM EMPLOYEES;
```

3. Click the icon **Run Statement**.

   The query runs. Under the Worksheet, the Query Result pane appears, showing all columns of the EMPLOYEES table.

> **Caution:**
>
> Be very careful about using SELECT * on tables with columns that store sensitive data, such as passwords or credit card information.

> **See Also:**
>
> "Tutorial: Viewing EMPLOYEES Table Properties and Data with SQL Developer" for information about another way to view table data with SQL Developer

Tutorial: Selecting Specific Columns of a Table

This tutorial shows how to select only the columns FIRST_NAME, LAST_NAME, and DEPARTMENT_ID of the EMPLOYEES table.

To select only FIRST_NAME, LAST_NAME, and DEPARTMENT_ID:

1. If a pane with the tab hr_conn is there, select it. Otherwise, click the icon SQL Worksheet, as in "Running Queries in SQL Developer".
2. If the Worksheet pane contains a query, clear the query by clicking the icon **Clear**.
3. In the Worksheet, enter this query:

```
SELECT FIRST_NAME, LAST_NAME, DEPARTMENT_ID FROM EMPLOYEES;
```

4. Click the icon **Run Statement**.

   The query runs. Under the Worksheet, the Query Result pane appears, showing the results of the query, which are similar to:

```
FIRST_NAME           LAST_NAME                DEPARTMENT_ID
-------------------- ------------------------ -------------
Donald               OConnell                            50
Douglas              Grant                               50
Jennifer             Whalen                              10
Michael              Hartstein                           20
Pat                  Fay                                 20
Susan                Mavris                              40
Hermann              Baer                                70
Shelley              Higgins                            110
William              Gietz                              110
Steven               King                                90
Neena                Kochhar                             90

FIRST_NAME           LAST_NAME                DEPARTMENT_ID
```

```
------------------- ------------------------ -------------
Lex                 De Haan                            90
...
Kevin               Feeney                             50

107 rows selected.
```

Displaying Selected Columns Under New Headings

In displayed query results, default column headings are column names. To display a column under a new heading, specify the new heading (**alias**) immediately after the column name. The alias renames the column for the duration of the query, but does not change its name in the database.

The query in Example 2-5 selects the same columns as the query in "Tutorial: Selecting Specific Columns of a Table", but it also specifies aliases for them. Because the aliases are not enclosed in double quotation marks, they are displayed in uppercase letters.

If you enclose column aliases in double quotation marks, case is preserved, and the aliases can include spaces, as in Example 2-6.

> **See Also:**
>
> *Oracle Database SQL Language Reference* for more information about the SELECT statement, including the column alias (c_alias)

Example 2-5 Displaying Selected Columns Under New Headings

```
SELECT FIRST_NAME First, LAST_NAME last, DEPARTMENT_ID DepT
FROM EMPLOYEES;
```

Result is similar to:

```
FIRST               LAST                       DEPT
------------------- ------------------------ ----------
Donald              OConnell                         50
Douglas             Grant                            50
Jennifer            Whalen                           10
Michael             Hartstein                        20
Pat                 Fay                              20
Susan               Mavris                           40
Hermann             Baer                             70
Shelley             Higgins                         110
William             Gietz                           110
Steven              King                             90
Neena               Kochhar                          90

FIRST               LAST                       DEPT
------------------- ------------------------ ----------
Lex                 De Haan                          90
...
Kevin               Feeney                           50

107 rows selected.
```

Example 2-6 Preserving Case and Including Spaces in Column Aliases

```
SELECT FIRST_NAME "Given Name", LAST_NAME "Family Name"
FROM EMPLOYEES;
```

Result is similar to:

```
Given Name           Family Name
-------------------- -------------------------
Donald               OConnell
Douglas              Grant
Jennifer             Whalen
Michael              Hartstein
Pat                  Fay
Susan                Mavris
Hermann              Baer
Shelley              Higgins
William              Gietz
Steven               King
Neena                Kochhar

Given Name           Family Name
-------------------- -------------------------
Lex                  De Haan
...
Kevin                Feeney

107 rows selected.
```

Selecting Data that Satisfies Specified Conditions

To select only data that matches a specified condition, include the WHERE clause in the SELECT statement.

The condition in the WHERE clause can be any SQL condition (for information about SQL conditions, see *Oracle Database SQL Language Reference*).

The query in Example 2-7 selects data only for employees in department 90.

To select data only for employees in departments 100, 110, and 120, use this WHERE clause:

```
WHERE DEPARTMENT_ID IN (100, 110, 120);
```

The query in Example 2-8 selects data only for employees whose last names start with "Ma".

To select data only for employees whose last names include "ma", use this WHERE clause:

```
WHERE LAST_NAME LIKE '%ma%';
```

The query in Example 2-9 tests for two conditions—whether the salary is at least 11000, and whether the commission percentage is not null.

See Also:

- *Oracle Database SQL Language Reference* for more information about the SELECT statement, including the WHERE clause
- *Oracle Database SQL Language Reference* for more information about SQL conditions

Example 2-7 Selecting Data from One Department

```
SELECT FIRST_NAME, LAST_NAME, DEPARTMENT_ID
FROM EMPLOYEES
WHERE DEPARTMENT_ID = 90;
```

Result is similar to:

```
FIRST_NAME           LAST_NAME                 DEPARTMENT_ID
-------------------- ------------------------- -------------
Steven               King                                 90
Neena                Kochhar                              90
Lex                  De Haan                              90


3 rows selected.
```

Example 2-8 Selecting Data for Last Names that Start with the Same Substring

```
SELECT FIRST_NAME, LAST_NAME
FROM EMPLOYEES
WHERE LAST_NAME LIKE 'Ma%';
```

Result is similar to:

```
FIRST_NAME           LAST_NAME
-------------------- -------------------------
Jason                Mallin
Steven               Markle
James                Marlow
Mattea               Marvins
Randall              Matos
Susan                Mavris

6 rows selected.
```

Example 2-9 Selecting Data that Satisfies Two Conditions

```
SELECT FIRST_NAME, LAST_NAME, SALARY, COMMISSION_PCT "%"
FROM EMPLOYEES
WHERE (SALARY >= 11000) AND (COMMISSION_PCT IS NOT NULL);
```

Result is similar to:

```
FIRST_NAME           LAST_NAME                   SALARY          %
-------------------- ------------------------- ---------- ----------
John                 Russell                      14000         .4
Karen                Partners                     13500         .3
Alberto              Errazuriz                    12000         .3
Gerald               Cambrault                    11000         .3
Lisa                 Ozer                         11500        .25
```

```
Ellen               Abel                          11000         .3

6 rows selected.
```

Sorting Selected Data

When query results are displayed, records can be in any order, unless you specify their order with the ORDER BY clause.

The query results in Example 2-10 are sorted by `LAST_NAME`, in ascending order (the default).

Alternatively, in SQL Developer, you can omit the ORDER BY clause and double-click the name of the column to sort.

The sort criterion need not be included in the select list, as Example 2-11 shows.

> **See Also:**
>
> *Oracle Database SQL Language Reference* for more information about the SELECT statement, including the ORDER BY clause

Example 2-10 Sorting Selected Data by LAST_NAME

```
SELECT FIRST_NAME, LAST_NAME, HIRE_DATE
FROM EMPLOYEES
ORDER BY LAST_NAME;
```

Result:

```
FIRST_NAME          LAST_NAME                 HIRE_DATE
------------------- ------------------------- ---------
Ellen               Abel                      11-MAY-04
Sundar              Ande                      24-MAR-08
Mozhe               Atkinson                  30-OCT-05
David               Austin                    25-JUN-05
Hermann             Baer                      07-JUN-02
Shelli              Baida                     24-DEC-05
Amit                Banda                     21-APR-08
Elizabeth           Bates                     24-MAR-07
...
FIRST_NAME          LAST_NAME                 HIRE_DATE
------------------- ------------------------- ---------
Jose Manuel         Urman                     07-MAR-06
Peter               Vargas                    09-JUL-06
Clara               Vishney                   11-NOV-05
Shanta              Vollman                   10-OCT-05
Alana               Walsh                     24-APR-06
Matthew             Weiss                     18-JUL-04
Jennifer            Whalen                    17-SEP-03
Eleni               Zlotkey                   29-JAN-08

107 rows selected
```

Example 2-11 Sorting Selected Data by an Unselected Column

```
SELECT FIRST_NAME, HIRE_DATE
FROM EMPLOYEES
ORDER BY LAST_NAME;
```

Result:

```
FIRST_NAME           HIRE_DATE
-------------------- ---------
Ellen                11-MAY-04
Sundar               24-MAR-08
Mozhe                30-OCT-05
David                25-JUN-05
Hermann              07-JUN-02
Shelli               24-DEC-05
Amit                 21-APR-08
Elizabeth            24-MAR-07
...
FIRST_NAME           HIRE_DATE
-------------------- ---------
Jose Manuel          07-MAR-06
Peter                09-JUL-06
Clara                11-NOV-05
Shanta               10-OCT-05
Alana                24-APR-06
Matthew              18-JUL-04
Jennifer             17-SEP-03
Eleni                29-JAN-08

107 rows selected.
```

Selecting Data from Multiple Tables

To select data from multiple tables, you use a query that is called a **join**. The tables in a join must share at least one column name.

Suppose that you want to select the FIRST_NAME, LAST_NAME, and DEPARTMENT_NAME of every employee. FIRST_NAME and LAST_NAME are in the EMPLOYEES table, and DEPARTMENT_NAME is in the DEPARTMENTS table. Both tables have DEPARTMENT_ID. You can use the query in Example 2-12.

Table-name qualifiers are optional for column names that appear in only one table of a join, but are required for column names that appear in both tables. The following query is equivalent to the query in Example 2-12:

```
SELECT FIRST_NAME "First",
LAST_NAME "Last",
DEPARTMENT_NAME "Dept. Name"
FROM EMPLOYEES, DEPARTMENTS
WHERE EMPLOYEES.DEPARTMENT_ID = DEPARTMENTS.DEPARTMENT_ID
ORDER BY DEPARTMENT_NAME, LAST_NAME;
```

To make queries that use qualified column names more readable, use table aliases, as in the following example:

```
SELECT FIRST_NAME "First",
LAST_NAME "Last",
DEPARTMENT_NAME "Dept. Name"
FROM EMPLOYEES e, DEPARTMENTS d
```

```
WHERE e.DEPARTMENT_ID = d.DEPARTMENT_ID
ORDER BY d.DEPARTMENT_NAME, e.LAST_NAME;
```

Although you create the aliases in the FROM clause, you can use them earlier in the query, as in the following example:

```
SELECT e.FIRST_NAME "First",
e.LAST_NAME "Last",
d.DEPARTMENT_NAME "Dept. Name"
FROM EMPLOYEES e, DEPARTMENTS d
WHERE e.DEPARTMENT_ID = d.DEPARTMENT_ID
ORDER BY d.DEPARTMENT_NAME, e.LAST_NAME;
```

> **See Also:**
>
> *Oracle Database SQL Language Reference* for more information about joins

Example 2-12 Selecting Data from Two Tables (Joining Two Tables)

```
SELECT EMPLOYEES.FIRST_NAME "First",
EMPLOYEES.LAST_NAME "Last",
DEPARTMENTS.DEPARTMENT_NAME "Dept. Name"
FROM EMPLOYEES, DEPARTMENTS
WHERE EMPLOYEES.DEPARTMENT_ID = DEPARTMENTS.DEPARTMENT_ID
ORDER BY DEPARTMENTS.DEPARTMENT_NAME, EMPLOYEES.LAST_NAME;
```

Result:

```
First                Last                      Dept. Name
-------------------- ------------------------- ------------------------------
William              Gietz                     Accounting
Shelley              Higgins                   Accounting
Jennifer             Whalen                    Administration
Lex                  De Haan                   Executive
Steven               King                      Executive
Neena                Kochhar                   Executive
John                 Chen                      Finance
...
Jose Manuel          Urman                     Finance
Susan                Mavris                    Human Resources
David                Austin                    IT
...
Valli                Pataballa                 IT
Pat                  Fay                       Marketing
Michael              Hartstein                 Marketing
Hermann              Baer                      Public Relations
Shelli               Baida                     Purchasing
...
Sigal                Tobias                    Purchasing
Ellen                Abel                      Sales
...
Eleni                Zlotkey                   Sales
Mozhe                Atkinson                  Shipping
...
Matthew              Weiss                     Shipping

106 rows selected.
```

Using Operators and Functions in Queries

The select_list of a query can include SQL expressions, which can include SQL operators and SQL functions. These operators and functions can have table data as operands and arguments. The SQL expressions are evaluated, and their values appear in the results of the query.

> **See Also:**
>
> - *Oracle Database SQL Language Reference* for more information about SQL operators
> - *Oracle Database SQL Language Reference* for more information about SQL functions

Using Arithmetic Operators in Queries

The basic arithmetic operators—+ (addition), - (subtraction), * (multiplication), and / (division)—operate on column values.

The query in Example 2-13 displays LAST_NAME, SALARY (monthly pay), and annual pay for each employee in department 90, in descending order of SALARY.

Example 2-13 Using an Arithmetic Expression in a Query

```
SELECT LAST_NAME,
SALARY "Monthly Pay",
SALARY * 12 "Annual Pay"
FROM EMPLOYEES
WHERE DEPARTMENT_ID = 90
ORDER BY SALARY DESC;
```

Result:

```
LAST_NAME                 Monthly Pay Annual Pay
------------------------- ----------- ----------
King                            24000     288000
De Haan                         17000     204000
Kochhar                         17000     204000
```

Using Numeric Functions in Queries

Numeric functions accept numeric input and return numeric values. Each numeric function returns a single value for each row that is evaluated.

The numeric functions that SQL supports are listed and described in *Oracle Database SQL Language Reference*.

The query in Example 2-14 uses the numeric function `ROUND` to display the daily pay of each employee in department 100, rounded to the nearest cent.

The query in Example 2-15 uses the numeric function `TRUNC` to display the daily pay of each employee in department 100, truncated to the nearest dollar.

> **See Also:**
>
> *Oracle Database SQL Language Reference* for more information about SQL numeric functions

Example 2-14 Rounding Numeric Data

```
SELECT LAST_NAME,
ROUND (((SALARY * 12)/365), 2) "Daily Pay"
FROM EMPLOYEES
WHERE DEPARTMENT_ID = 100
ORDER BY LAST_NAME;
```

Result:

```
LAST_NAME                 Daily Pay
------------------------- ----------
Chen                          269.59
Faviet                        295.89
Greenberg                     394.52
Popp                          226.85
Sciarra                       253.15
Urman                         256.44

6 rows selected.
```

Example 2-15 Truncating Numeric Data

```
SELECT LAST_NAME,
TRUNC ((SALARY * 12)/365) "Daily Pay"
FROM EMPLOYEES
WHERE DEPARTMENT_ID = 100
ORDER BY LAST_NAME;
```

Result:

```
LAST_NAME                 Daily Pay
------------------------- ----------
Chen                             269
Faviet                           295
Greenberg                        394
Popp                             226
Sciarra                          253
Urman                            256

6 rows selected.
```

Using the Concatenation Operator in Queries

The concatenation operator (`||`) combines two strings into one string, by appending the second string to the first. For example, `'a'||'b'='ab'`. You can use this operator to combine information from two columns or expressions in the same column of a query result.

The query in Example 2-16 concatenates the first name, a space, and the last name of each selected employee.

> **See Also:**
>
> *Oracle Database SQL Language Reference* for more information about the concatenation operator

Example 2-16 Concatenating Character Data

```
SELECT FIRST_NAME || ' ' || LAST_NAME "Name"
FROM EMPLOYEES
WHERE DEPARTMENT_ID = 100
ORDER BY LAST_NAME;
```

Result:

```
Name
---------------------------------------------
John Chen
Daniel Faviet
Nancy Greenberg
Luis Popp
Ismael Sciarra
Jose Manuel Urman

6 rows selected.
```

Using Character Functions in Queries

Character functions accept character input. Most return character values, but some return numeric values. Each character function returns a single value for each row that is evaluated.

The character functions that SQL supports are listed and described in *Oracle Database SQL Language Reference*.

The functions UPPER, INITCAP, and LOWER display their character arguments in uppercase, initial capital, and lowercase, respectively.

The query in Example 2-17 displays LAST_NAME in uppercase, FIRST_NAME with the first character in uppercase and all others in lowercase, and EMAIL in lowercase.

> **See Also:**
>
> *Oracle Database SQL Language Reference* for more information about SQL character functions

Example 2-17 Changing the Case of Character Data

```
SELECT UPPER(LAST_NAME) "Last",
INITCAP(FIRST_NAME) "First",
LOWER(EMAIL) "E-Mail"
FROM EMPLOYEES
WHERE DEPARTMENT_ID = 100
ORDER BY EMAIL;
```

Result:

```
Last                    First               E-Mail
----------------------- ------------------- -------------------------
FAVIET                  Daniel              dfaviet
SCIARRA                 Ismael              isciarra
CHEN                    John                jchen
URMAN                   Jose Manuel         jmurman
POPP                    Luis                lpopp
GREENBERG               Nancy               ngreenbe

6 rows selected.
```

Using Datetime Functions in Queries

Datetime functions operate on DATE, time stamp, and interval values. Each datetime function returns a single value for each row that is evaluated.

The datetime functions that SQL supports are listed and described in *Oracle Database SQL Language Reference*.

For each DATE and time stamp value, Oracle Database stores this information:

- Year
- Month
- Date
- Hour
- Minute
- Second

  For each time stamp value, Oracle Database also stores the fractional part of the second, whose precision you can specify. To store the time zone also, use the data type TIMESTAMP WITH TIME ZONE or TIMESTAMP WITH LOCAL TIME ZONE.

For more information about the DATE data type, see *Oracle Database SQL Language Reference*.

For more information about the TIMESTAMP data type, see *Oracle Database SQL Language Reference*.

For information about the other time stamp data types and the interval data types, see *Oracle Database SQL Language Reference*.

The query in Example 2-18 uses the EXTRACT and SYSDATE functions to show how many years each employee in department 100 has been employed. The SYSDATE function returns the current date of the system clock as a DATE value. For more information about the SYSDATE function, see *Oracle Database SQL Language Reference*. For information about the EXTRACT function, see *Oracle Database SQL Language Reference*.

The query in Example 2-19 uses the SYSTIMESTAMP function to display the current system date and time. The SYSTIMESTAMP function returns a TIMESTAMP value. For information about the SYSTIMESTAMP function, see *Oracle Database SQL Language Reference*.

The table in the FROM clause of the query, DUAL, is a one-row table that Oracle Database creates automatically along with the data dictionary. Select from DUAL when you want to compute a constant expression with the SELECT statement. Because

DUAL has only one row, the constant is returned only once. For more information about selecting from DUAL, see *Oracle Database SQL Language Reference*.

> **See Also:**
>
> *Oracle Database SQL Language Reference* for more information about SQL datetime functions

Example 2-18 Displaying the Number of Years Between Dates

```
SELECT LAST_NAME,
(EXTRACT(YEAR FROM SYSDATE) - EXTRACT(YEAR FROM HIRE_DATE)) "Years Employed"
FROM EMPLOYEES
WHERE DEPARTMENT_ID = 100
ORDER BY "Years Employed";
```

Result:

```
LAST_NAME                 Years Employed
------------------------- --------------
Popp                                   5
Urman                                  6
Chen                                   7
Sciarra                                7
Greenberg                             10
Faviet                                10

6 rows selected.
```

Example 2-19 Displaying System Date and Time

```
SELECT EXTRACT(HOUR FROM SYSTIMESTAMP) || ':' ||
EXTRACT(MINUTE FROM SYSTIMESTAMP) || ':' ||
ROUND(EXTRACT(SECOND FROM SYSTIMESTAMP), 0) || ', ' ||
EXTRACT(MONTH FROM SYSTIMESTAMP) || '/' ||
EXTRACT(DAY FROM SYSTIMESTAMP) || '/' ||
EXTRACT(YEAR FROM SYSTIMESTAMP) "System Time and Date"
FROM DUAL;
```

Results depend on current SYSTIMESTAMP value, but have this format:

```
System Time and Date
---------------------------------------------------------------------
18:17:53, 12/27/2012
```

Using Conversion Functions in Queries

Conversion functions convert one data type to another.

The conversion functions that SQL supports are listed and described in *Oracle Database SQL Language Reference*.

The query in Example 2-20 uses the TO_CHAR function to convert HIRE_DATE values (which are of type DATE) to character values that have the format `FMMonth DD YYYY`. `FM` removes leading and trailing blanks from the month name. `FMMonth DD YYYY` is an example of a **datetime format model**. For information about datetime format models, see *Oracle Database SQL Language Reference*.

The query in Example 2-21 uses the TO_NUMBER function to convert POSTAL_CODE values (which are of type VARCHAR2) to values of type NUMBER, which it uses in calculations.

> **See Also:**
>
> - *Oracle Database SQL Language Reference* for more information about SQL conversion functions
> - "About the NLS_DATE_FORMAT Parameter"

Example 2-20 Converting Dates to Characters Using a Format Template

```
SELECT LAST_NAME,
HIRE_DATE,
TO_CHAR(HIRE_DATE, 'FMMonth DD YYYY') "Date Started"
FROM EMPLOYEES
WHERE DEPARTMENT_ID = 100
ORDER BY LAST_NAME;
```

Result:

```
LAST_NAME                 HIRE_DATE Date Started
------------------------- --------- -----------------
Chen                      28-SEP-05 September 28 2005
Faviet                    16-AUG-02 August 16 2002
Greenberg                 17-AUG-02 August 17 2002
Popp                      07-DEC-07 December 7 2007
Sciarra                   30-SEP-05 September 30 2005
Urman                     07-MAR-06 March 7 2006

6 rows selected.
```

Example 2-21 Converting Characters to Numbers

```
SELECT CITY,
POSTAL_CODE "Old Code",
TO_NUMBER(POSTAL_CODE) + 1 "New Code"
FROM LOCATIONS
WHERE COUNTRY_ID = 'US'
ORDER BY POSTAL_CODE;
```

Result:

```
CITY                          Old Code       New Code
----------------------------- ------------ ----------
Southlake                     26192             26193
South Brunswick               50090             50091
Seattle                       98199             98200
South San Francisco           99236             99237

4 rows selected.
```

Using Aggregate Functions in Queries

An aggregate function takes a group of rows and returns a single result row. The group of rows can be an entire table or view.

The aggregate functions that SQL supports are listed and described in *Oracle Database SQL Language Reference*.

Aggregate functions are especially powerful when used with the GROUP BY clause, which groups query results by one or more columns, with a result for each group.

The query in Example 2-22 uses the COUNT function and the GROUP BY clause to show how many people report to each manager. The wildcard character, *, represents an entire record.

Example 2-22 shows that one employee does not report to a manager. The following query selects the first name, last name, and job title of that employee:

```
COLUMN FIRST_NAME FORMAT A10;
COLUMN LAST_NAME FORMAT A10;
COLUMN JOB_TITLE FORMAT A10;

SELECT e.FIRST_NAME,
e.LAST_NAME,
j.JOB_TITLE
FROM EMPLOYEES e, JOBS j
WHERE e.JOB_ID = j.JOB_ID
AND MANAGER_ID IS NULL;
```

Result:

```
FIRST_NAME LAST_NAME  JOB_TITLE
---------- ---------- ----------
Steven     King       President
```

To have the query return only rows where aggregate values meet specified conditions, use an aggregate function in the HAVING clause of the query.

The query in Example 2-23 shows how much each department spends annually on salaries, but only for departments for which that amount exceeds $1,000,000.

The query in Example 2-24 uses several aggregate functions to show statistics for the salaries of each JOB_ID.

> **See Also:**
>
> *Oracle Database SQL Language Reference* for more information about SQL aggregate functions

Example 2-22 Counting the Number of Rows in Each Group

```
SELECT MANAGER_ID "Manager",
COUNT(*) "Number of Reports"
FROM EMPLOYEES
GROUP BY MANAGER_ID
ORDER BY MANAGER_ID;
```

Result:

```
   Manager Number of Reports
---------- -----------------
       100                14
       101                 5
       102                 1
       103                 4
       108                 5
       114                 5
       120                 8
       121                 8
       122                 8
       123                 8
       124                 8
       145                 6
       146                 6
       147                 6
       148                 6
       149                 6
       201                 1
       205                 1
                           1

19 rows selected.
```

Example 2-23 Limiting Aggregate Functions to Rows that Satisfy a Condition

```
SELECT DEPARTMENT_ID "Department",
SUM(SALARY*12) "All Salaries"
FROM EMPLOYEES
HAVING SUM(SALARY * 12) >= 1000000
GROUP BY DEPARTMENT_ID;
```

Result:

```
Department All Salaries
---------- ------------
        50      1876800
        80      3654000
```

Example 2-24 Using Aggregate Functions for Statistical Information

```
SELECT JOB_ID,
COUNT(*) "#",
MIN(SALARY) "Minimum",
ROUND(AVG(SALARY), 0) "Average",
MEDIAN(SALARY) "Median",
MAX(SALARY) "Maximum",
ROUND(STDDEV(SALARY)) "Std Dev"
FROM EMPLOYEES
GROUP BY JOB_ID
ORDER BY JOB_ID;
```

Result:

```
JOB_ID              #    Minimum    Average     Median    Maximum    Std Dev
---------- ---------- ---------- ---------- ---------- ---------- ----------
AC_ACCOUNT          1       8300       8300       8300       8300          0
AC_MGR              1      12008      12008      12008      12008          0
AD_ASST             1       4400       4400       4400       4400          0
AD_PRES             1      24000      24000      24000      24000          0
```

```
AD_VP              2      17000      17000      17000      17000          0
FI_ACCOUNT         5       6900       7920       7800       9000        766
FI_MGR             1      12008      12008      12008      12008          0
HR_REP             1       6500       6500       6500       6500          0
IT_PROG            5       4200       5760       4800       9000       1926
MK_MAN             1      13000      13000      13000      13000          0
MK_REP             1       6000       6000       6000       6000          0
PR_REP             1      10000      10000      10000      10000          0
PU_CLERK           5       2500       2780       2800       3100        239
PU_MAN             1      11000      11000      11000      11000          0
SA_MAN             5      10500      12200      12000      14000       1525
SA_REP            30       6100       8350       8200      11500       1524
SH_CLERK          20       2500       3215       3100       4200        548
ST_CLERK          20       2100       2785       2700       3600        453
ST_MAN             5       5800       7280       7900       8200       1066

19 rows selected.
```

Using NULL-Related Functions in Queries

The NULL-related functions facilitate the handling of NULL values.

The NULL-related functions that SQL supports are listed and described in *Oracle Database SQL Language Reference*.

The query in Example 2-25 returns the last name and commission of the employees whose last names begin with 'B'. If an employee receives no commission (that is, if COMMISSION_PCT is NULL), the NVL function substitutes "Not Applicable" for NULL.

The query in Example 2-26 returns the last name, salary, and income of the employees whose last names begin with 'B', using the NVL2 function: If COMMISSION_PCT is not NULL, the income is the salary plus the commission; if COMMISSION_PCT is NULL, income is only the salary.

> **See Also:**
>
> - *Oracle Database SQL Language Reference* for more information about the NVL function
> - *Oracle Database SQL Language Reference* for more information about the NVL2 function

Example 2-25 Substituting a String for a NULL Value

```
SELECT LAST_NAME,
NVL(TO_CHAR(COMMISSION_PCT), 'Not Applicable') "COMMISSION"
FROM EMPLOYEES
WHERE LAST_NAME LIKE 'B%'
ORDER BY LAST_NAME;
```

Result:

```
LAST_NAME                 COMMISSION
------------------------- ----------------------------------------
Baer                      Not Applicable
Baida                     Not Applicable
Banda                     .1
```

```
Bates                     .15
Bell                      Not Applicable
Bernstein                 .25
Bissot                    Not Applicable
Bloom                     .2
Bull                      Not Applicable

9 rows selected.
```

Example 2-26 Specifying Different Expressions for NULL and Not NULL Values

```
SELECT LAST_NAME, SALARY,
NVL2(COMMISSION_PCT, SALARY + (SALARY * COMMISSION_PCT), SALARY) INCOME
FROM EMPLOYEES WHERE LAST_NAME LIKE 'B%'
ORDER BY LAST_NAME;
```

Result:

```
LAST_NAME                    SALARY     INCOME
------------------------- ---------- ----------
Baer                          10000      10000
Baida                          2900       2900
Banda                          6200       6820
Bates                          7300       8395
Bell                           4000       4000
Bernstein                      9500      11875
Bissot                         3300       3300
Bloom                         10000      12000
Bull                           4100       4100

9 rows selected.
```

Using CASE Expressions in Queries

A CASE expression lets you use IF ... THEN ... ELSE logic in SQL statements without invoking subprograms. There are two kinds of CASE expressions, simple and searched.

The query in Example 2-27 uses a simple CASE expression to show the country name for each country code.

The query in Example 2-28 uses a searched CASE expression to show proposed salary increases (15%, 10%, 5%, or 0%), based on date ranges associated with length of service.

See Also:

- *Oracle Database SQL Language Reference* for more information about CASE expressions
- *Oracle Database PL/SQL Language Reference* for more information about CASE expressions
- "Using the DECODE Function in Queries"
- "Using the CASE Statement"

Example 2-27 Using a Simple CASE Expression in a Query

```
SELECT UNIQUE COUNTRY_ID ID,
       CASE COUNTRY_ID
         WHEN 'AU' THEN 'Australia'
         WHEN 'BR' THEN 'Brazil'
         WHEN 'CA' THEN 'Canada'
         WHEN 'CH' THEN 'Switzerland'
         WHEN 'CN' THEN 'China'
         WHEN 'DE' THEN 'Germany'
         WHEN 'IN' THEN 'India'
         WHEN 'IT' THEN 'Italy'
         WHEN 'JP' THEN 'Japan'
         WHEN 'MX' THEN 'Mexico'
         WHEN 'NL' THEN 'Netherlands'
         WHEN 'SG' THEN 'Singapore'
         WHEN 'UK' THEN 'United Kingdom'
         WHEN 'US' THEN 'United States'
       ELSE 'Unknown'
       END COUNTRY
FROM LOCATIONS
ORDER BY COUNTRY_ID;
```

Result:

```
ID COUNTRY
-- --------------
AU Australia
BR Brazil
CA Canada
CH Switzerland
CN China
DE Germany
IN India
IT Italy
JP Japan
MX Mexico
NL Netherlands
SG Singapore
UK United Kingdom
US United States

14 rows selected.
```

Example 2-28 Using a Searched CASE Expression in a Query

```
SELECT LAST_NAME "Name",
HIRE_DATE "Started",
SALARY "Salary",
CASE
  WHEN HIRE_DATE < TO_DATE('01-Jan-03', 'dd-mon-yy')
    THEN TRUNC(SALARY*1.15, 0)
  WHEN HIRE_DATE >= TO_DATE('01-Jan-03', 'dd-mon-yy') AND
       HIRE_DATE < TO_DATE('01-Jan-06', 'dd-mon-yy')
    THEN TRUNC(SALARY*1.10, 0)
  WHEN HIRE_DATE >= TO_DATE('01-Jan-06', 'dd-mon-yy') AND
       HIRE_DATE < TO_DATE('01-Jan-07', 'dd-mon-yy')
    THEN TRUNC(SALARY*1.05, 0)
  ELSE SALARY
END "Proposed Salary"
FROM EMPLOYEES
```

```
WHERE DEPARTMENT_ID = 100
ORDER BY HIRE_DATE;
```

Result:

```
Name                      Started       Salary Proposed Salary
------------------------- --------- ---------- ---------------
Faviet                    16-AUG-02       9000           10350
Greenberg                 17-AUG-02      12008           13809
Chen                      28-SEP-05       8200            9020
Sciarra                   30-SEP-05       7700            8470
Urman                     07-MAR-06       7800            8190
Popp                      07-DEC-07       6900            6900

6 rows selected.
```

Using the DECODE Function in Queries

The DECODE function compares an expression to several search values. Whenever the value of the expression matches a search value, DECODE returns the result associated with that search value. If DECODE finds no match, then it returns the default value (if specified) or NULL (if no default value is specified).

The query in Example 2-29 uses the DECODE function to show proposed salary increases for three different jobs. The expression is JOB_ID; the search values are 'PU_CLERK', 'SH_CLERK', and 'ST_CLERK'; and the default is SALARY.

> **Note:**
>
> The arguments of the DECODE function can be any of the SQL numeric or character types. Oracle automatically converts the expression and each search value to the data type of the first search value before comparing. Oracle automatically converts the return value to the same data type as the first result. If the first result has the data type CHAR or if the first result is NULL, then Oracle converts the return value to the data type VARCHAR2.

> **See Also:**
>
> - *Oracle Database SQL Language Reference* for information about the DECODE function
> - "Using CASE Expressions in Queries"

Example 2-29 Using the DECODE Function in a Query

```
SELECT LAST_NAME, JOB_ID, SALARY,
DECODE(JOB_ID,
  'PU_CLERK', SALARY * 1.10,
  'SH_CLERK', SALARY * 1.15,
  'ST_CLERK', SALARY * 1.20,
  SALARY) "Proposed Salary"
FROM EMPLOYEES
WHERE JOB_ID LIKE '%_CLERK'
```

```
AND LAST_NAME < 'E'
ORDER BY LAST_NAME;
```

Result:

```
LAST_NAME                 JOB_ID         SALARY Proposed Salary
------------------------- ---------- ---------- ---------------
Atkinson                  ST_CLERK         2800            3360
Baida                     PU_CLERK         2900            3190
Bell                      SH_CLERK         4000            4600
Bissot                    ST_CLERK         3300            3960
Bull                      SH_CLERK         4100            4715
Cabrio                    SH_CLERK         3000            3450
Chung                     SH_CLERK         3800            4370
Colmenares                PU_CLERK         2500            2750
Davies                    ST_CLERK         3100            3720
Dellinger                 SH_CLERK         3400            3910
Dilly                     SH_CLERK         3600            4140

11 rows selected.
```