1

Introduction

Lesson Objectives

After completing this lesson, you should be able to:

- Discuss the goals of the course
- Describe the database schema and tables that are used in the course
- Identify the available environments that can be used in the course
- Review some of the basic concepts of SQL

Lesson Agenda

- Course objectives and course agenda
- The database schema, the appendixes and practices, and development environments used in this course
- Review of some basic SQL concepts
- Oracle Database 12*c* documentation and additional resources

Course Objectives

After completing this course, you should be able to:

- Manage objects with data dictionary views
- Create schema objects
- Manage schema objects
- Write multiple-column subqueries
- Use scalar and correlated subqueries
- Control user access to specific database objects
- Add new users with different levels of access privileges
- Manipulate large data sets in the Oracle database by using subqueries
- Manage data in different time zones

Course Prerequisites

The *Oracle Database: SQL Workshop I* course is a prerequisite for this course.

The *Oracle Database: SQL Workshop I* course offers you an introduction to Oracle Database technology. In this course, you learn the basic concepts of relational databases and the powerful SQL programming language. This course provides the essential SQL skills that enable you to write queries against single and multiple tables, manipulate data in tables, create database objects, and query metadata.

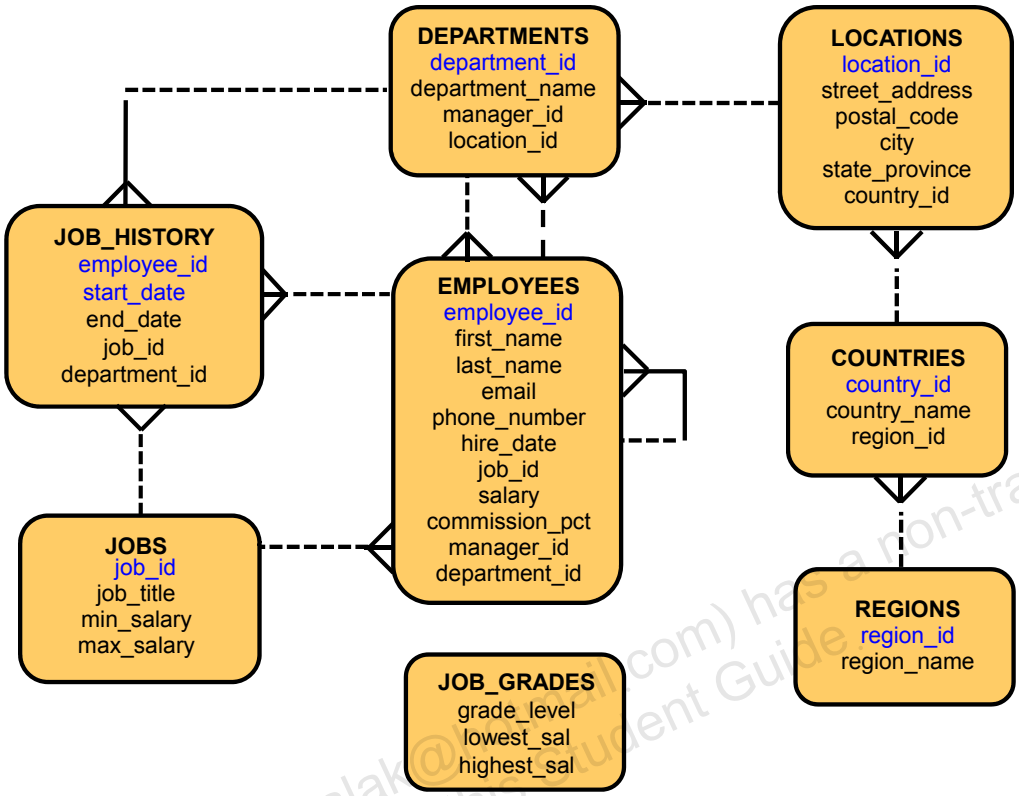
Course Agenda

- Day 1:
  - Introduction
  - Introduction to Data Dictionary Views
  - Creating Sequence, Synonyms, and Indexes
  - Creating Views
  - Managing Schema Objects
- Day 2:
  - Retrieving Data by Using Subqueries
  - Manipulating Data by Using Subqueries
  - Controlling User Access
  - Manipulating Data
  - Managing Data in Different Time Zones

Lesson Agenda

- Course objectives and course agenda
- The database schema, the appendixes and practices, and development environments used in this course
- Review of some basic SQL concepts
- Oracle Database 12*c* documentation and additional resources

Tables Used in This Course



This course uses data from the following tables:

Table Descriptions

- The `EMPLOYEES` table contains information about all the employees, such as their first and last names, job `IDs`, salaries, hire dates, department `IDs`, and manager `IDs`. This table is a child of the `DEPARTMENTS` table.
- The `DEPARTMENTS` table contains information such as the department `ID`, department name, manager `ID`, and location `ID`. This table is the primary key table to the `EMPLOYEES` table.
- The `LOCATIONS` table contains department location information. It contains location `ID`, street address, city, state province, postal code, and country `ID` information. It is the primary key table to the `DEPARTMENTS` table and is a child of the `COUNTRIES` table.
- The `COUNTRIES` table contains the country names, country `IDs`, and region `IDs`. It is a child of the `REGIONS` table. This table is the primary key table to the `LOCATIONS` table.
- The `REGIONS` table contains region `IDs` and region names of the various countries. It is a primary key table to the `COUNTRIES` table.

- The JOB_GRADES table identifies a salary range per job grade. The salary ranges do not overlap.
- The JOB_HISTORY table stores job history of the employees.
- The JOBS table contains job titles and salary ranges.

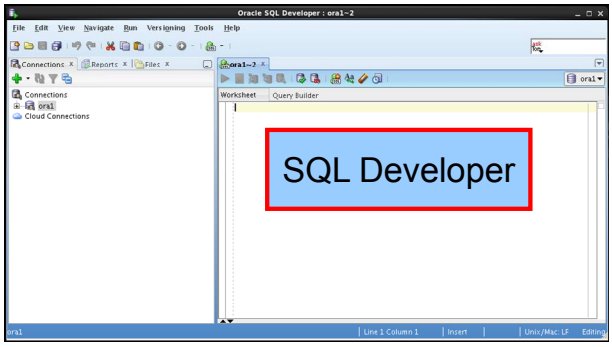
Appendixes and Practices Used in This Course

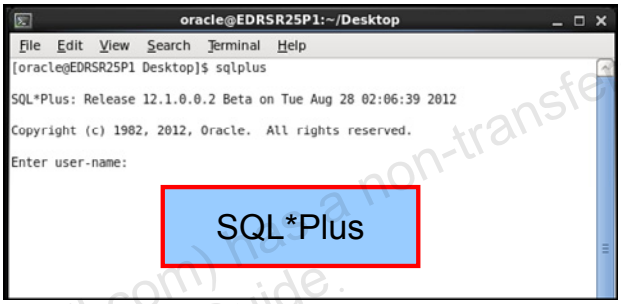
- Appendix A: Table Descriptions
- Appendix B: Using SQL Developer
- Appendix C: Using SQL* Plus
- Appendix D: Commonly Used SQL Commands
- Appendix E: Generating Reports by Grouping Related Data
- Appendix F: Hierarchical Retrieval
- Appendix G: Writing Advanced Scripts
- Appendix H: Oracle Database Architectural Components
- Appendix I : Regular Expression Support
- Practices and Solutions
- Additional Practices and Solutions

Development Environments

There are two development environments for this course:

- The primary tool is Oracle SQL Developer.
- You can also use SQL*Plus command-line interface.





SQL Developer

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.

SQL*Plus

The SQL*Plus environment may also be used to run all SQL commands covered in this course.

Note

- See Appendix B titled “Using SQL Developer” for information about using SQL Developer.
- See Appendix C titled “Using SQL*Plus” for information about using SQL*Plus.

Lesson Agenda

- Course objectives and course agenda
- The database schema, the appendixes and practices, and development environments used in this course
- Review of some basic SQL concepts
- Oracle Database 12*c* documentation and additional resources

ORACLE

The next few slides provide a brief overview of some of the basic concepts that you learned in the course titled *Oracle Database: SQL Workshop I*.

Review of Restricting Data

- Restrict the rows that are returned by using the `WHERE` clause.
- Use comparison conditions to compare one expression with another value or expression.

Operator	Meaning
`BETWEEN ...AND...`	Between two values (inclusive)
`IN(set)`	Match any of a list of values
`LIKE`	Match a character pattern

- Use logical conditions to combine the result of two component conditions and produce a single result based on those conditions.

You can restrict the rows that are returned from the query by using the `WHERE` clause. A `WHERE` clause contains a condition that must be met, and it directly follows the `FROM` clause.

The `WHERE` clause can compare values in columns, literal values, arithmetic expression, or functions. It consists of three elements:

- Column name
- Comparison condition
- Column name, constant, or list of values

You use comparison conditions in the `WHERE` clause in the following format:

```
... WHERE expr operator value
```

Apart from those mentioned in the slide, you use other comparison conditions such as `=`, `<`, `>`, `<>`, `<=`, and `>=`.

Three logical operators are available in SQL:

- `AND`
- `OR`
- `NOT`

Review of Sorting Data

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

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

	LAST_NAME	JOB_ID	DEPARTMENT_ID	HIRE_DATE
1	De Haan	AD_VP	90	13-JAN-01
2	Gietz	AC_ACCOUNT	110	07-JUN-02
3	Baer	PR_REP	70	07-JUN-02
4	Mavris	HR_REP	40	07-JUN-02
5	Higgins	AC_MGR	110	07-JUN-02
6	Faviet	FI_ACCOUNT	100	16-AUG-02
7	Greenberg	FI_MGR	100	17-AUG-02

...

ORACLE

The order of rows that are returned in a query result is undefined. The `ORDER BY` clause can be used to sort the rows. If you use the `ORDER BY` clause, it must be the last clause of the SQL statement. You can specify an expression, an alias, or a column position as the sort condition.

Syntax

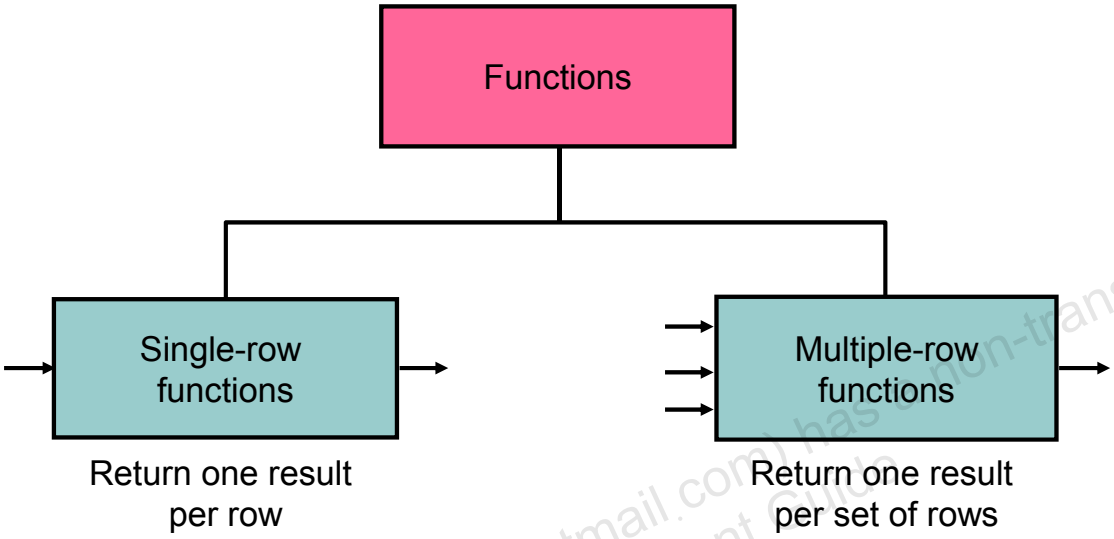
```
SELECT          expr
FROM            table
[WHERE          condition(s)]
[ORDER BY   {column, expr, numeric_position} [ASC|DESC]];
```

In the syntax:

`ORDER BY`	Specifies the order in which the retrieved rows are displayed
`ASC`	Orders the rows in ascending order (This is the default order.)
`DESC`	Orders the rows in descending order

If the `ORDER BY` clause is not used, the sort order is undefined, and the Oracle Server may not fetch rows in the same order for the same query twice. Use the `ORDER BY` clause to display the rows in a specific order.

Review of SQL Functions



There are two types of functions:

- Single-row functions
- Multiple-row functions

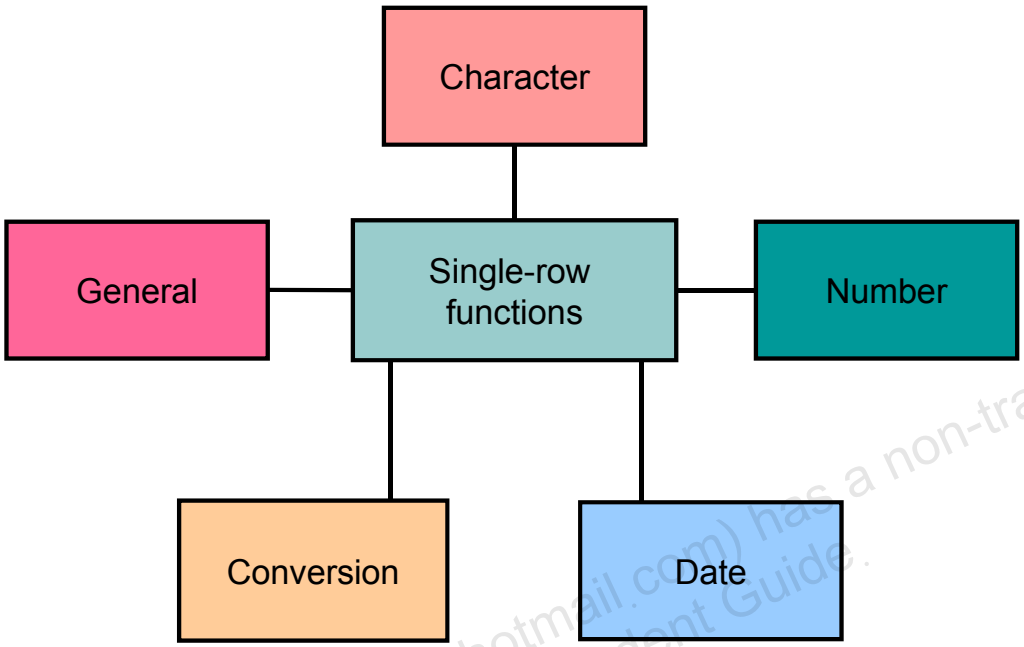
Single-Row Functions

These functions operate only on single rows and return one result per row. There are different types of single-row functions such as character, number, date, conversion, and general functions.

Multiple-Row Functions

Functions can manipulate groups of rows to give one result per group of rows. These functions are also known as *group functions*.

Review of Single-Row Functions

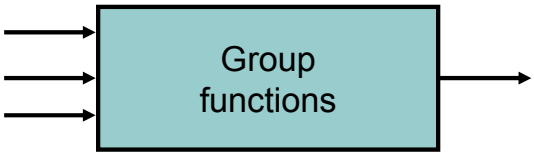


The following are different types of single-row functions:

- **Character functions:** Accept character input and can return both character and number values
- **Number functions:** Accept numeric input and return numeric values
- **Date functions:** Operate on values of the `DATE` data type (All date functions return a value of the `DATE` data type, except the `MONTHS_BETWEEN` function, which returns a number.)
- **Conversion functions:** Convert a value from one data type to another
- **General functions:**
  - `NVL`
  - `NVL2`
  - `NULLIF`
  - `COALESCE`
  - `CASE`
  - `DECODE`

Review of Types of Group Functions

- AVG
- COUNT
- MAX
- MIN
- STDDEV
- SUM
- VARIANCE



Each of the functions accepts an argument. The following table identifies the options that you can use in the syntax:

Function	Description
AVG([DISTINCT|ALL] *n*)	Average value of *n*, ignoring null values
COUNT({* | [DISTINCT|ALL] *expr*})	Number of rows, where *expr* evaluates to something other than null (count all selected rows using *, including duplicates and rows with nulls)
MAX([DISTINCT|ALL] *expr*)	Maximum value of *expr*, ignoring null values
MIN([DISTINCT|ALL] *expr*)	Minimum value of *expr*, ignoring null values
STDDEV([DISTINCT|ALL] *n*)	Standard deviation of *n*, ignoring null values
SUM([DISTINCT|ALL] *n*)	Sum values of *n*, ignoring null values
VARIANCE([DISTINCT|ALL] *n*)	Variance of *n*, ignoring null values

Review of Using Subqueries

- A subquery is a SELECT statement nested in a clause of another SELECT statement.
- Syntax:

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

- Types of subqueries:

Single-row subquery	Multiple-row subquery
Returns only one row	Returns more than one row
Uses single-row comparison operators	Uses multiple-row comparison operators

You can build powerful statements out of simple ones by using subqueries. Subqueries are useful when a query is based on a search criterion with unknown intermediate values.

You can place the subquery in a number of SQL clauses, including the following:

- WHERE clause
- HAVING clause
- FROM clause

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

A single-row subquery uses a single-row operator such as =, >, <, >=, <=, or <>. With a multiple-row subquery, you use a multiple-row operator such as IN, ANY, or ALL.

Example: Display details of employees whose salary is equal to the minimum salary.

```
SELECT last_name, salary, job_id
FROM   employees
WHERE  salary = (SELECT MIN(salary)
                 FROM   employees );
```

In the example, the `MIN` group function returns a single value to the outer query.
Note: In this course, you learn how to use multiple-column subqueries. Multiple-column subqueries return more than one column from the inner `SELECT` statement.

Review of Managing Tables Using DML Statements

A data manipulation language (DML) statement is executed when you:

- Add new rows to a table
- Modify existing rows in a table
- Remove existing rows from a table

Function	Description
INSERT	Adds a new row to the table
UPDATE	Modifies existing rows in the table
DELETE	Removes existing rows from the table
MERGE	Updates, inserts, or deletes a row conditionally into/from a table

When you want to add, update, or delete data in the database, you execute a DML statement. A collection of DML statements that form a logical unit of work is called a transaction. You can add new rows to a table by using the INSERT statement. With the following syntax, only one row is inserted at a time.

```
INSERT INTO table [(column [, column…])]
VALUES      (value[, value...]);
```

You can use the INSERT statement to add rows to a table where the values are derived from existing tables. In place of the VALUES clause, you use a subquery. The number of columns and their data types in the column list of the INSERT clause must match the number of values and their data types in the subquery.

The UPDATE statement modifies specific rows if you specify the WHERE clause.

```
UPDATE table
SET column = value [, column = value, ...]
[WHERE condition];
```

You can remove existing rows by using the `DELETE` statement. You can delete specific rows by specifying the `WHERE` clause in the `DELETE` statement.

```
DELETE [FROM] table
[WHERE condition];
```

You learn about the `MERGE` statement in the lesson titled “Manipulating Data.”

Lesson Agenda

- Course objectives and course agenda
- The database schema, the appendixes and practices, and development environments used in this course
- Review of some basic SQL concepts
- Oracle Database 12*c* documentation and additional resources

Oracle Database SQL Documentation

- *Oracle Database New Features Guide*
- *Oracle Database Reference*
- *Oracle Database SQL Language Reference*
- *Oracle Database Concepts*
- *Oracle Database SQL Developer User's Guide Release 3.2*

ORACLE

Navigate to http://www.oracle.com/pls/db121/homepage to access the Oracle Database 12*c* Release 1 documentation library.

Additional Resources

For additional information about the new Oracle 12*c* SQL, refer to the following:

- *Oracle Database 12c: New Features Self Studies*
- *Oracle by Example series (OBE): Oracle Database 12c*
- *Oracle Learning Library:*
  - http://www.oracle.com/goto/oll
- Access the online SQL Developer Home Page, which is available at:
  - http://www.oracle.com/technology/products/database/sql_developer/index.html
- Access the SQL Developer tutorial, which is available online at
  - http://download.oracle.com/oll/tutorials/SQLDeveloper/index.htm

Summary

In this lesson, you should have learned how to:

- Discuss the goals of the course
- Describe the database schema and tables that are used in the course
- Identify the available environments that can be used in the course
- Recall some of the basic concepts of SQL

Practice 1: Overview

This practice covers the following topics:

- Running the SQL Developer online tutorial
- Starting SQL Developer and creating a new database connection and browsing the tables
- Executing SQL statements using the SQL Worksheet
- Running some basic SQL commands

ORACLE

In this practice, you use SQL Developer to execute SQL statements.

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