**Introduction to SQL Queries**

# Table of Contents

**Course Practice Environment: Security Credentials I-1**

Course Practice Environment: Security Credentials.................................................................................. I-2

Practices for Lesson 1: Introduction 1-1

[Practices for Lesson 1: Overview 1-2](#_TOC_250026)

[Practice 1-1: Introduction 1-3](#_TOC_250025)

[Solution 1-1: Introduction 1-5](#_TOC_250024)

Practices for Lesson 2: Retrieving Data Using the SQL SELECT Statement 2-1

[Practices for Lesson 2: Overview 2-2](#_TOC_250023)

[Practice 2-1: Retrieving Data Using the SQL SELECT Statement 2-3](#_TOC_250022)

[Solution 2-1: Retrieving Data Using the SQL SELECT Statement 2-8](#_TOC_250021)

Practices for Lesson 3: Restricting and Sorting Data 3-1

[Practices for Lesson 3: Overview 3-2](#_TOC_250020)

[Practice 3-1: Restricting and Sorting Data 3-3](#_TOC_250019)

[Solution 3-1: Restricting and Sorting Data 3-7](#_TOC_250018)

Practices for Lesson 4: Using Single-Row Functions to Customize Output 4-1

[Practices for Lesson 4: Overview 4-2](#_TOC_250017)

[Practice 4-1: Using Single-Row Functions to Customize Output 4-3](#_TOC_250016)

[Solution 4-1: Using Single-Row Functions to Customize Output 4-8](#_TOC_250015)

Practices for Lesson 5: Using Conversion Functions and Conditional Expressions 5-1

[Practices for Lesson 5: Overview 5-2](#_TOC_250014)

[Practice 5-1: Using Conversion Functions and Conditional Expressions 5-3](#_TOC_250013)

[Solution 5-1: Using Conversion Functions and Conditional Expressions 5-8](#_TOC_250012)

Practices for Lesson 6: Reporting Aggregated Data Using the Group Functions 6-1

[Practices for Lesson 6: Overview 6-2](#_TOC_250011)

[Practice 6-1: Reporting Aggregated Data by Using Group Functions 6-3](#_TOC_250010)

[Solution 6-1: Reporting Aggregated Data by Using Group Functions 6-6](#_TOC_250009)

Practices for Lesson 7: Displaying Data from Multiple Tables Using Joins 7-1

Practices for Lesson 7: Overview 7-2

[Practice 7-1: Displaying Data from Multiple Tables by Using Joins 7-3](#_TOC_250008)

[Solution 7-1: Displaying Data from Multiple Tables by Using Joins 7-8](#_TOC_250007)

Practices for Lesson 8: Using Subqueries to Solve Queries 8-1

Practices for Lesson 8: Overview 8-2

[Practice 8-1: Using Subqueries to Solve Queries 8-3](#_TOC_250006)

[Solution 8-1: Using Subqueries to Solve Queries 8-6](#_TOC_250005)

Practices for Lesson 9: Using the Set Operators 9-1

Practices for Lesson 9: Overview 9-2

[Practice 9-1: Using Set Operators 9-3](#_TOC_250004)

[Solution 9-1: Using Set Operators 9-5](#_TOC_250003)

Practices for Lesson 10: Managing Tables Using DML Statements 10-1

[Practices for Lesson 10: Overview 10-2](#_TOC_250002)

[Practice 10-1: Managing Tables by Using DML Statements 10-3](#_TOC_250001)

[Solution 10-1: Managing Tables by Using DML Statements 10-7](#_TOC_250000)

Table of Contents

iii

**Practices for Lesson 11: Introduction to Data Definition Language 11-1**

Practices for Lesson 11: Overview 11-2

Practice 11-1: Introduction to Data Definition Language 11-3

Solution 11-1: Introduction to Data Definition Language 11-7

**Additional Practices and Solutions 12-1**

Practices for Lesson 1: Overview 12-2

Practice 1-1: Additional Practice 12-3

Solution 1-1: Additional Practice 12-11

Case Study: Online Book Store 12-16

Practice 1-2 12-17

Solution 1-2 12-22

Table of Contents

iv

**Course Practice Environment: Security Credentials**

**Chapter I**

# Course Practice Environment: Security Credentials

For connection-specific credentials used in this course, see the following table:

|  |  |  |
| --- | --- | --- |
| **Connection-Specific Credentials** | | |
| **Connection\_Name** | **Username** | **Password** |
| system | system | fenago |
| myconnection | hr | hr |

Course Practice Environment: Security Credentials

Chapter I - Page 2

**Practices for Lesson 1: Introduction**

**Chapter 1**

# Practices for Lesson 1: Overview

### Practice Overview

In this practice, you start Oracle SQL Developer, create a new database connection, and browse your HR tables. You also set some SQL Developer preferences.

In some of the practices, 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 within the allocated time, and would like an additional 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.
* For any query, the sequence of rows retrieved from the database may differ from the screenshots shown.

Practices for Lesson 1: Introduction

Chapter 1 - Page 2

# Practice 1-1: Introduction

### Overview

This is the first of many practices in this course. The solutions (if you require them) can be found at the end of the practices for each lesson. The practices are intended to cover most of the topics that are presented in the corresponding lesson.

In this practice, you perform the following:

* Start Oracle SQL Developer and create a new connection to the hr account.
* Use Oracle SQL Developer to examine the data objects in the hr account. The

hr account contains the HR schema tables.

Note the following location for the practice files:

Desktop/oracle19c-labs/sql-intro/labs/sql1/labs

If you are asked to save any practice files, save them in the above location.

### Tasks

1. Start Oracle SQL Developer by Using the SQL Developer Desktop Icon
2. Create a New Oracle SQL Developer Database Connection
   1. To create a new database connection, in the Connections Navigator, right-click Connections and select New Connection from the context menu. The New / Select Database Connection dialog box appears.
   2. Create a database connection by using the following information: Connection Name: myconnection

Username: hr

Password: Enter the password from the Course Practice Environment: Security Credentials document

Hostname: localhost

Port: 1521

Service Name: FENAGODB1

Ensure that you select the Save Password check box.

1. Test the Oracle SQL Developer Database Connection and Connect to the Database
   1. Test the new connection.
   2. If the status is Success, connect to the database by using this new connection.
2. Browse the Tables in the Connections Navigator
   1. In the Connections Navigator, view the objects that are available to you in the Tables node. Verify that the following tables are present:

COUNTRIES DEPARTMENTS EMPLOYEES JOB\_GRADES JOB\_HISTORY JOBS

LOCATIONS REGIONS

Practices for Lesson 1: Introduction

Chapter 1 - Page 3

b. Browse the structure of the EMPLOYEES table.

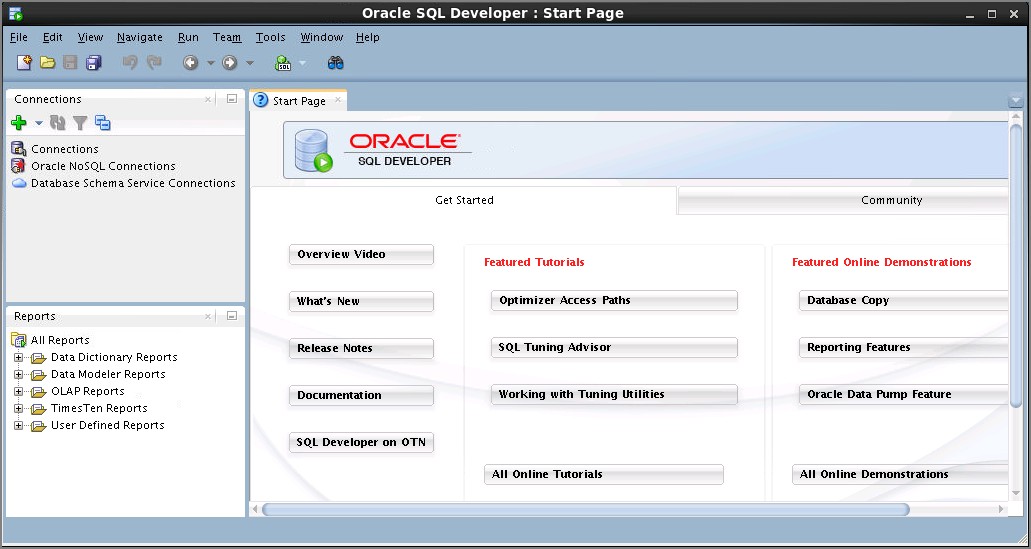
c. View the data of the DEPARTMENTS table.

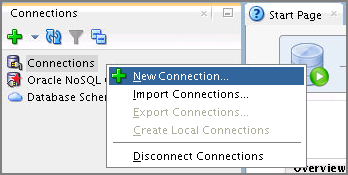
# Solution 1-1: Introduction

1. Start Oracle SQL Developer by Using the SQL Developer Desktop Icon Double-click the SQL Developer desktop icon.



The SQL Developer interface appears.

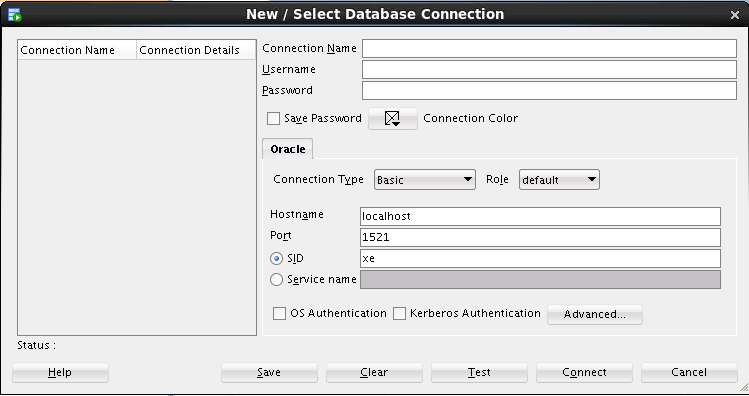


1. Create a New Oracle SQL Developer Database Connection
   1. To create a new database connection, in the Connections Navigator, right-click Connections and select New Connection from the context menu.

Practices for Lesson 1: Introduction

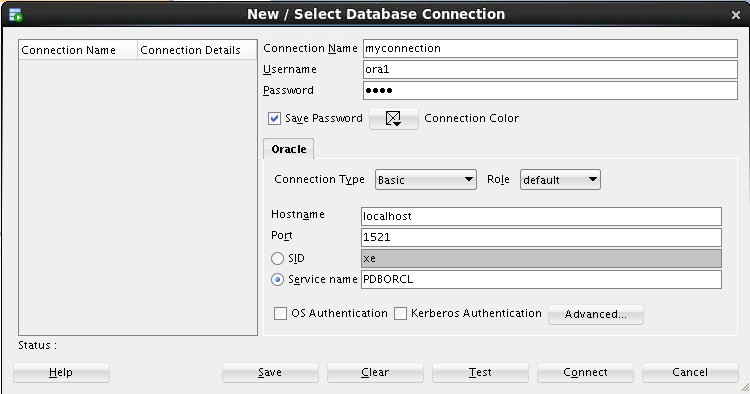
Chapter 1 - Page 5

The New / Select Database Connection dialog box appears.



* 1. Create a database connection by using the following information:
     1. Connection Name: myconnection
     2. Username: hr
     3. Password: Enter the password from the Course Practice Environment: Security Credentials document
     4. Hostname: localhost
     5. Port: 1521
     6. Service Name: FENAGODB1

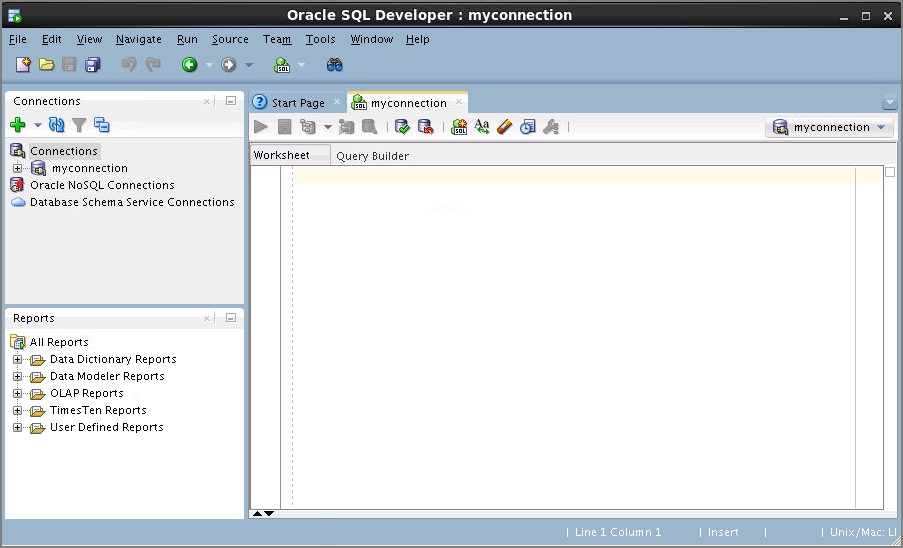
Ensure that you select the Save Password check box.



Practices for Lesson 1: Introduction

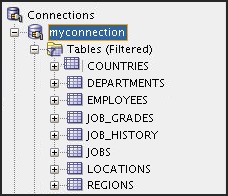
Chapter 1 - Page 6

## 

When you create a connection, a SQL Worksheet for that connection opens automatically.

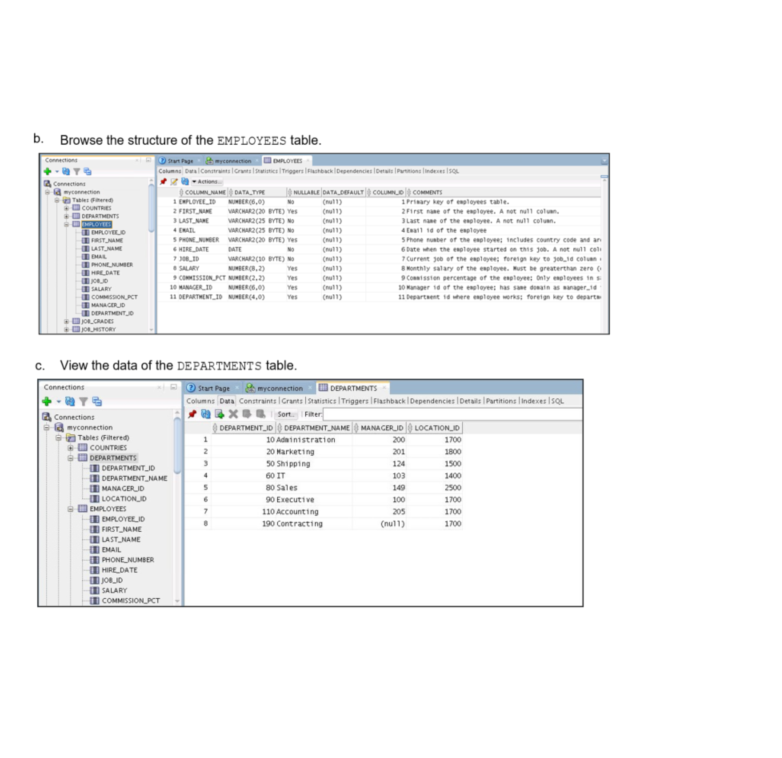
1. Browse the Tables in the Connections Navigator
   1. In the Connections Navigator, view the objects that are available to you in the Tables node. Verify that the following tables are present:

COUNTRIES DEPARTMENTS EMPLOYEES JOB\_GRADES JOB\_HISTORY JOBS

LOCATIONS REGIONS

Practices for Lesson 1: Introduction

Chapter 1 - Page 8



**Practices for Lesson 2:**

**Retrieving Data Using the SQL SELECT Statement**

**Chapter 2**

# Practices for Lesson 2: Overview

### Practice Overview

This practice covers the following topics:

* Selecting all data from different tables
* Describing the structure of tables
* Performing arithmetic calculations and specifying column names

Practices for Lesson 2: Retrieving Data Using the SQL SELECT Statement

Chapter 2 - Page 2

# Practice 2-1: Retrieving Data Using the SQL SELECT Statement

### Overview

In this practice, you write simple SELECT queries. The queries cover most of the SELECT

clauses and operations that you learned in this lesson.

### Task 1

Test your knowledge:

1. The following SELECT statement executes successfully:

SELECT last\_name, job\_id, salary AS Sal FROM employees;

True/False

1. The following SELECT statement executes successfully:

SELECT \*

FROM job\_grades;

True/False

1. 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;

### Task 2

Note the following points before you begin with the practices:

* + Save all your practice files at the following location:

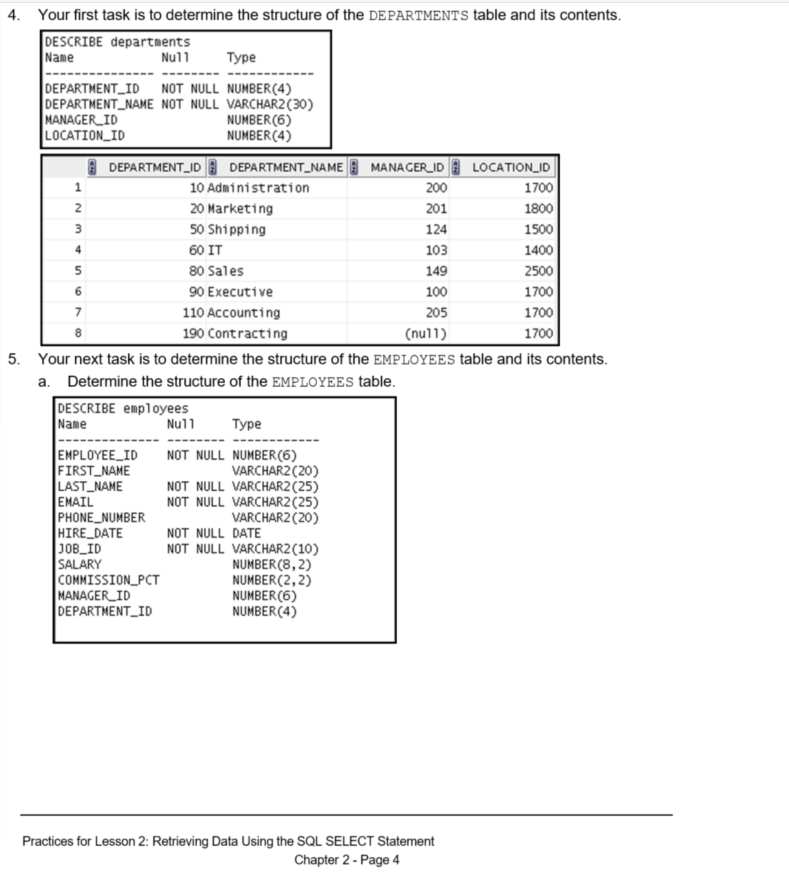
Desktop/oracle19c-labs/sql-intro/labs/sql1/labs

* + Enter your SQL statements in a SQL Worksheet. To open a new worksheet, click File menu, select New. A New Gallery dialog window appears. Click Database Files under Database Tier on the left pane. Select SQL File on the right pane and click OK.
  + To save a script in SQL Developer, make sure that the required SQL Worksheet is active, and then from the File menu, select Save As to save your SQL statement as a lab\_<lessonno>\_<stepno>.sql script. When you modify an existing script, make sure that you use Save As to save it with a different file name.
  + To run the query, click the Run Statement icon in the SQL Worksheet. Alternatively, you can press F9. For DML and DDL statements, use the Run Script icon or press F5.
  + After you have executed the query, make sure that you do not enter your next query in the same worksheet. Open a new worksheet.

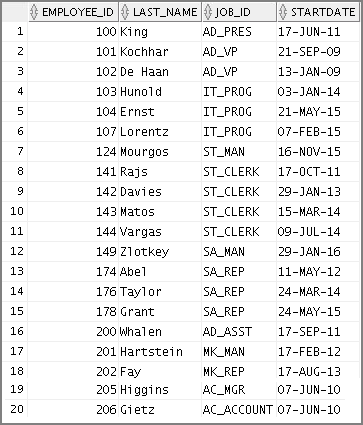
You have been hired as a SQL programmer for Acme Corporation. Your first task is to create some reports based on data from the Human Resources tables.

Practices for Lesson 2: Retrieving Data Using the SQL SELECT Statement

Chapter 2 - Page 3

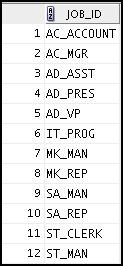


* 1. The HR department wants a query to display the last name, job ID, hire date, and employee ID for each employee, with the employee ID appearing first. Provide an alias STARTDATE for the HIRE\_DATE column. Save your SQL statement to a file named lab\_02\_5b.sql so that you can dispatch this file to the HR department. Test your query in the lab\_02\_5b.sql file to ensure that it runs correctly.

**Note:** After you have executed the query, make sure that you do not enter your next query in the same worksheet. Open a new worksheet.

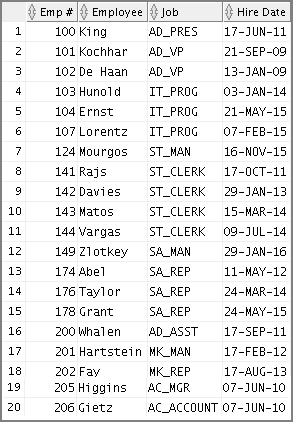
Practices for Lesson 2: Retrieving Data Using the SQL SELECT Statement

Chapter 2 - Page 5

1. The HR department wants a query to display all unique job IDs from the EMPLOYEES table.

### Task 3

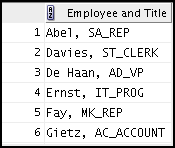
If you have time, complete the following exercises:

1. The HR department wants more descriptive column headings for its report on employees. Copy the statement from lab\_02\_5b.sql to a new SQL Worksheet. Name the columns Emp #, Employee, Job, and Hire Date, respectively. Then run the query again.

Practices for Lesson 2: Retrieving Data Using the SQL SELECT Statement

Chapter 2 - Page 6

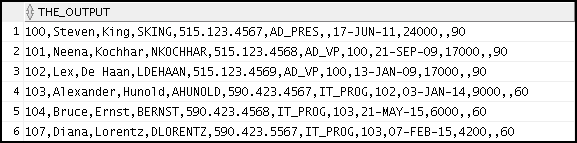
1. The HR department has requested a report of all employees and their job IDs. Display the last name concatenated with the job ID (separated by a comma and space) and name the column Employee and Title.



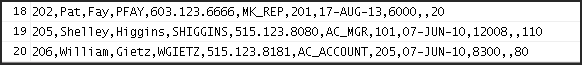
#### …



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

1. 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 by a comma. Name the column THE\_OUTPUT.

#### …



Practices for Lesson 2: Retrieving Data Using the SQL SELECT Statement

Chapter 2 - Page 7

# Solution 2-1: Retrieving Data Using the SQL SELECT Statement

### Task 1

Test your knowledge:

1. The following SELECT statement executes successfully:

SELECT last\_name, job\_id, salary AS Sal FROM employees;

**True**/False

1. The following SELECT statement executes successfully:

SELECT \*

FROM job\_grades;

**True**/False

1. 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;

#### The EMPLOYEES table does not contain a column called sal. The column is called

**SALARY.**

#### The multiplication operator is \*, not x as shown in line 2.

* + **The ANNUAL SALARY alias cannot include spaces. The alias should read**

#### ANNUAL\_SALARY or should be enclosed within double quotation marks.

* + **A comma is missing after the LAST\_NAME column.**

### Task 2

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.

1. Your first task is to determine the structure of the DEPARTMENTS table and its contents.
2. To determine the DEPARTMENTS table structure:

DESCRIBE departments

1. To view the data contained in the DEPARTMENTS table:

SELECT \*

FROM departments;

Practices for Lesson 2: Retrieving Data Using the SQL SELECT Statement

Chapter 2 - Page 8

1. Your next task is to determine the structure of the EMPLOYEES table and its contents.
2. Determine the structure of the EMPLOYEES table.

DESCRIBE employees

1. The HR department wants a query to display the last name, job ID, hire date, and employee ID for each employee, with the employee ID appearing first. Provide an alias STARTDATE for the HIRE\_DATE column. Save your SQL statement to a file named lab\_02\_5b.sql so that you can dispatch this file to the HR department. Test your query in the lab\_02\_5b.sql file to ensure that it runs correctly.

SELECT employee\_id, last\_name, job\_id, hire\_date StartDate

FROM employees;

1. The HR department wants a query to display all unique job IDs from the EMPLOYEES table.

SELECT DISTINCT job\_id FROM employees;

### Task 3

If you have time, complete the following exercises:

1. The HR department wants more descriptive column headings for its report on employees. Copy the statement from lab\_02\_5b.sql to a new SQL Worksheet. Name the columns Emp #, Employee, Job, and Hire Date, respectively. Then run the query again.

SELECT employee\_id "Emp #", last\_name "Employee", job\_id "Job", hire\_date "Hire Date"

FROM employees;

1. 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.

SELECT last\_name||', '||job\_id "Employee and Title"

FROM employees;

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

1. 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 by a comma. Name the column THE\_OUTPUT.

SELECT employee\_id || ',' || first\_name || ',' || last\_name

|| ',' || email || ',' || phone\_number || ','|| job\_id

|| ',' || manager\_id || ',' || hire\_date || ','

|| salary || ',' || commission\_pct || ',' || department\_id

THE\_OUTPUT

FROM employees;

Practices for Lesson 2: Retrieving Data Using the SQL SELECT Statement

Chapter 2 - Page 9

T

**Practices for Lesson 3: Restricting and Sorting Data**

**Chapter 3**

# Practices for Lesson 3: Overview

### Practices Overview

This practice covers the following topics:

* Selecting data and changing the order of the rows that are displayed
* Restricting rows by using the WHERE clause
* Sorting rows by using the ORDER BY clause
* Using substitution variables to add flexibility to your SQL SELECT statements

Practices for Lesson 3: Restricting and Sorting Data

Chapter 3 - Page 2

# Practice 3-1: Restricting and Sorting Data

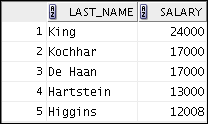
### Overview

In this practice, you build reports by using statements with the WHERE clause and the ORDER BY clause. You make the SQL statements more reusable and generic by including the ampersand substitution.

### Task

The HR department needs your assistance in creating some queries.

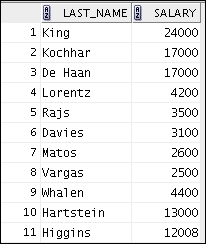
1. Because of budget issues, the HR department needs a report that displays the last name and salary of employees who earn more than $12,000. Save your SQL statement as a file named lab\_03\_01.sql. Run your query.



1. Open a new SQL Worksheet. Create a report that displays the last name and department number for employee number 176.

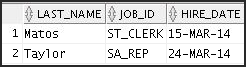


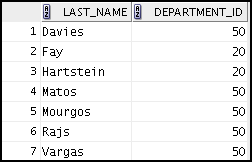
1. The HR department needs to find high-salaried and low-salaried employees. Modify lab\_03\_01.sql to display the last name and salary for any employee whose salary is not in the range $5,000 through $12,000. Save your SQL statement as lab\_03\_03.sql.



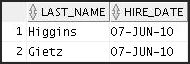
Practices for Lesson 3: Restricting and Sorting Data

Chapter 3 - Page 3

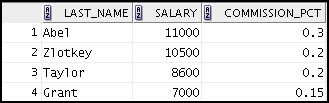
1. Create a report to display the last name, job ID, and hire date for employees with the last names of Matos and Taylor. Order the query in ascending order by hire date.
2. Display the last name and department ID of all employees in department 20 or department 50 in ascending alphabetical order by last\_name.



1. Modify lab\_03\_03.sql to display the last name and salary of employees who earn between $5,000 and $12,000, and are in department 20 or department 50. Label the columns Employee and Monthly Salary, respectively. Save lab\_03\_03.sql as lab\_03\_06.sql. Run the statement in lab\_03\_06.sql.
2. The HR department needs a report that displays the last name and hire date of all employees who were hired in 2010.



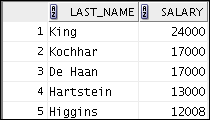
1. Create a report to display the last name and job title of all employees who do not have a manager.
2. Create a report to display the last name, salary, and commission of all employees who earn commissions. Sort the data in descending order of salary and commissions.

Use the column’s numeric position in the ORDER BY clause.

Practices for Lesson 3: Restricting and Sorting Data

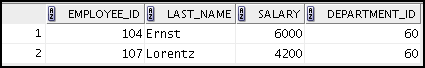
Chapter 3 - Page 4

1. Members of the HR department want to have more flexibility with the queries that you are writing. They would like a report that displays the last name and salary of employees who earn more than an amount that the user specifies after a prompt. Save this query to a file named lab\_03\_10.sql. (You can use the query created in Task 1 and modify it.) If you enter 12000 when prompted, the report displays the following results:



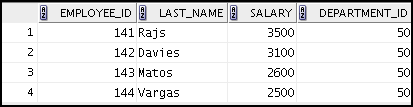
1. The HR department wants to run reports based on a manager. Create a query that prompts the user for a manager ID, and generates the employee ID, last name, salary, and department for that manager’s employees. The HR department wants the ability to sort the report on a selected column. You can test the data with the following values:

manager\_id = 103, sorted by last\_name:

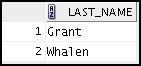


manager\_id = 201, sorted by salary:



manager\_id = 124, sorted by employee\_id:

If you have time, complete the following exercises:

1. Display the last names of all employees where the third letter of the name is “a.”
2. Display the last names of all employees who have both an “a” and an “e” in their last name.

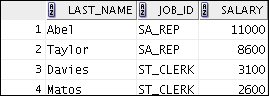


Practices for Lesson 3: Restricting and Sorting Data

Chapter 3 - Page 5

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

1. Display the last name, job, and salary for all employees whose jobs are either that of a sales representative or a stock clerk, and whose salaries are not equal to $2,500, $3,500, or $7,000.



1. Modify lab\_03\_06.sql to display the last name, salary, and commission for all employees whose commission is 20%. Save lab\_03\_06.sql as lab\_03\_15.sql. Rerun the statement in lab\_03\_15.sql.



Practices for Lesson 3: Restricting and Sorting Data

Chapter 3 - Page 6

# Solution 3-1: Restricting and Sorting Data

The HR department needs your assistance in creating some queries.

1. Because of budget issues, the HR department needs a report that displays the last name and salary of employees earning more than $12,000. Save your SQL statement as a file named lab\_03\_01.sql. Run your query.

SELECT last\_name, salary FROM employees

WHERE salary > 12000;

1. Open a new SQL Worksheet. Create a report that displays the last name and department number for employee number 176.

SELECT last\_name, department\_id FROM employees

WHERE employee\_id = 176;

1. The HR department needs to find high-salaried and low-salaried employees. Modify lab\_03\_01.sql to display the last name and salary for all employees whose salary is not in the range $5,000 through $12,000. Save your SQL statement as lab\_03\_03.sql.

SELECT last\_name, salary FROM employees

WHERE salary NOT BETWEEN 5000 AND 12000;

1. Create a report to display the last name, job ID, and hire date for employees with the last names of Matos and Taylor. Order the query in ascending order by hire date.

SELECT last\_name, job\_id, hire\_date FROM employees

WHERE last\_name IN ('Matos', 'Taylor')

ORDER BY hire\_date;

1. Display the last name and department ID of all employees in department 20 or department 50 in ascending alphabetical order by last\_name.

SELECT last\_name, department\_id FROM employees

WHERE department\_id IN (20, 50)

ORDER BY last\_name ASC;

1. Modify lab\_03\_03.sql to list the last name and salary of employees who earn between

$5,000 and $12,000, and are in department 20 or department 50. Label the columns Employee and Monthly Salary, respectively. Save lab\_03\_03.sql as lab\_03\_06.sql. Run the statement in lab\_03\_06.sql.

SELECT FROM WHERE

AND

last\_name "Employee", salary "Monthly Salary"

employees

salary

BETWEEN 5000 AND 12000

department\_id IN (20, 50);

Practices for Lesson 3: Restricting and Sorting Data

Chapter 3 - Page 7

1. The HR department needs a report that displays the last name and hire date of all employees who were hired in 2010.

SELECT last\_name, hire\_date FROM employees

WHERE hire\_date >= '01-JAN-10' AND hire\_date < '01-JAN-11';

1. Create a report to display the last name and job title of all employees who do not have a manager.

SELECT last\_name, job\_id FROM employees

WHERE manager\_id IS NULL;

1. Create a report to display the last name, salary, and commission for all employees who earn commissions. Sort data in descending order of salary and commissions. Use the column’s numeric position in the ORDER BY clause.

SELECT last\_name, salary, commission\_pct FROM employees

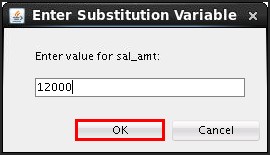
WHERE commission\_pct IS NOT NULL

ORDER BY 2 DESC, 3 DESC;

1. Members of the HR department want to have more flexibility with the queries that you are writing. They would like a report that displays the last name and salary of employees who earn more than an amount that the user specifies after a prompt. (You can use the query created in Task 1 and modify it.) Save this query to a file named lab\_03\_10.sql.

SELECT last\_name, salary FROM employees

WHERE salary > &sal\_amt;

Enter 12000 when prompted for a value in a dialog box. Click OK.

1. The HR department wants to run reports based on a manager. Create a query that prompts the user for a manager ID, and generates the employee ID, last name, salary, and department for that manager’s employees. The HR department wants the ability to sort the report on a selected column. You can test the data with the following values:

manager \_id = 103, sorted by last\_name manager\_id = 201, sorted by salary manager\_id = 124, sorted by employee\_id

SELECT employee\_id, last\_name, salary, department\_id FROM employees

WHERE manager\_id = &mgr\_num

ORDER BY &order\_col;

If you have the time, complete the following exercises:

1. Display the last names of all employees where the third letter of the name is “a.”

SELECT last\_name FROM employees

WHERE last\_name LIKE ' a%';

1. Display the last names of all employees who have both an “a” and an “e” in their last name.

SELECT last\_name FROM employees

WHERE last\_name LIKE '%a%'

AND last\_name LIKE '%e%';

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

1. Display the last name, job, and salary for all employees whose job is that of a sales representative or a stock clerk, and whose salary is not equal to $2,500, $3,500, or $7,000.

SELECT last\_name, job\_id, salary FROM employees

WHERE job\_id IN ('SA\_REP', 'ST\_CLERK')

AND salary NOT IN (2500, 3500, 7000);

1. Modify lab\_03\_06.sql to display the last name, salary, and commission for all employees whose commission amount is 20%. Save lab\_03\_06.sql as lab\_03\_15.sql. Rerun the statement in lab\_03\_15.sql.

SELECT last\_name "Employee", salary "Monthly Salary",

commission\_pct

FROM employees

WHERE commission\_pct = .20;

Practices for Lesson 3: Restricting and Sorting Data

Chapter 3 - Page 9

**Practices for Lesson 4:**

**Using Single-Row Functions to Customize Output**

**Chapter 4**

# Practices for Lesson 4: Overview

### Practice Overview

This practice covers the following topics:

* Writing a query that displays the system date
* Creating queries that require the use of numeric, character, and date functions
* Performing calculations of years and months of service for an employee

Practices for Lesson 4: Using Single-Row Functions to Customize Output

Chapter 4 - Page 2

# Practice 4-1: Using Single-Row Functions to Customize Output

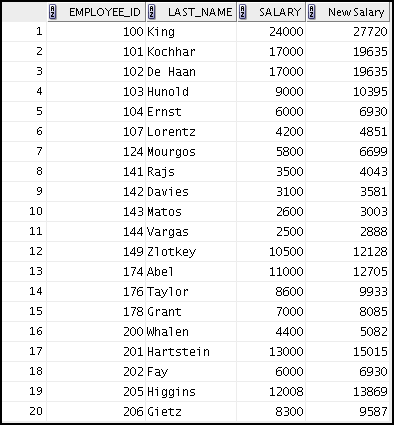
### Overview

In this practice, you use the different functions that are available for character, number, and date data types. Remember that for nested functions, the results are evaluated from the innermost function to the outermost function.

### Tasks

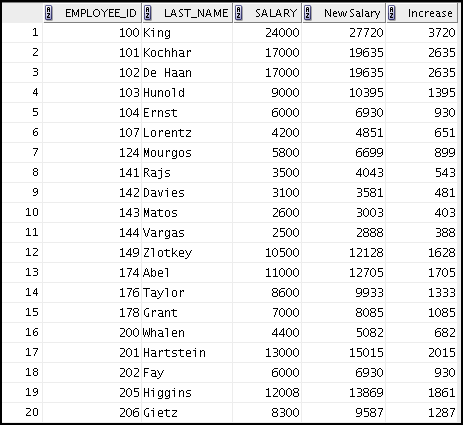
1. Write a query to display the system date. Label the column Date.

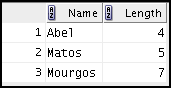
**Note:** If your database is remotely located in a different time zone, the output will be the date for the operating system on which the database resides.

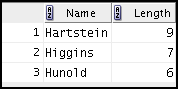
1. The HR department needs a report to display the employee number, last name, salary, and salary increased by 15.5% (expressed as a whole number) for each employee. Label the column New Salary. Save your SQL statement in a file named lab\_04\_02.sql.
2. Run your query in the lab\_04\_02.sql file.

Practices for Lesson 4: Using Single-Row Functions to Customize Output

Chapter 4 - Page 3

1. Modify your query in lab\_04\_02.sql to add a column that subtracts the old salary from the new salary. Label the column Increase. Save the contents of the file as lab\_04\_04.sql. Run the revised query.
2. Perform the following tasks:
   1. Write a query that displays the last name (with the first letter in uppercase and all the other letters in lowercase) and the length of the last name for all employees whose name starts with the letters “A,” or “M.” Give each column an appropriate label. Sort the results by the employees’ last names.

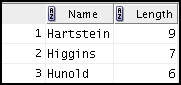
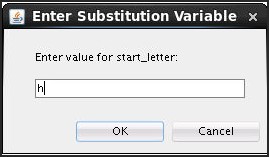


* 1. Rewrite the query so that the user is prompted to enter the letter that the last name starts with. For example, if the user enters “H” (capitalized) when prompted for a letter, the output should show all employees whose last name starts with the letter “H.”

Practices for Lesson 4: Using Single-Row Functions to Customize Output

Chapter 4 - Page 4

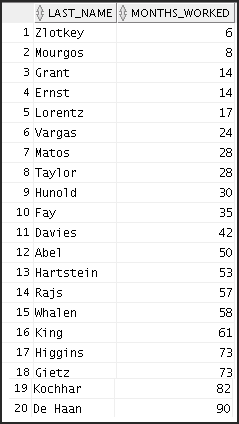
* 1. Modify the query such that the case of the letter that is entered does not affect the output. The entered letter must be capitalized before being processed by the SELECT query.



If you have time, complete the following exercises:

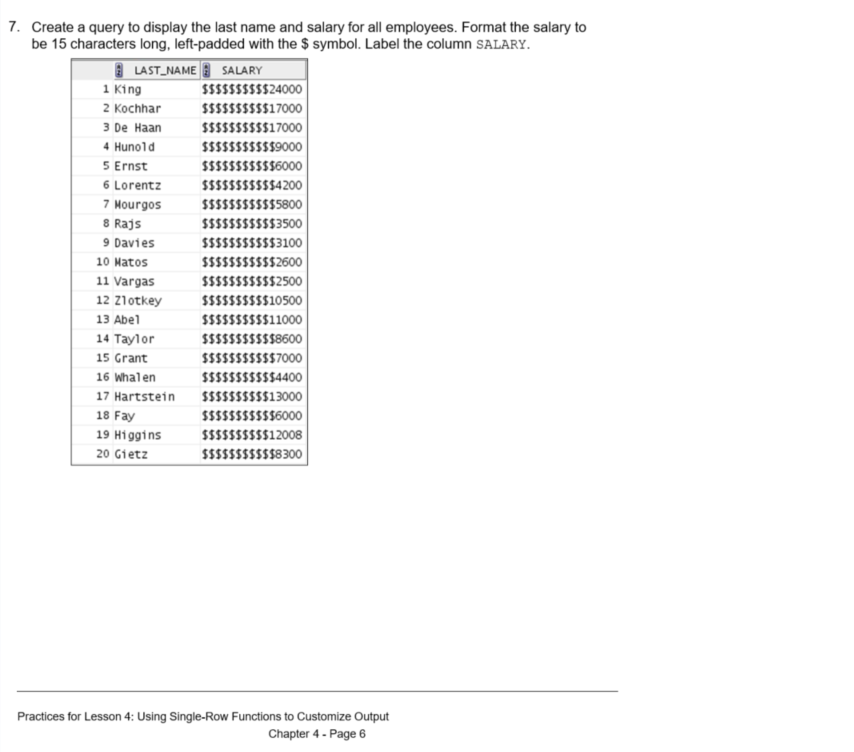
1. The HR department wants to find the duration of employment for each employee. For each employee, display the last name and calculate the number of months between today and the date on which the employee was hired. Label the column as MONTHS\_WORKED. Order your results by the number of months employed. The number of months must be rounded to the closest whole number.

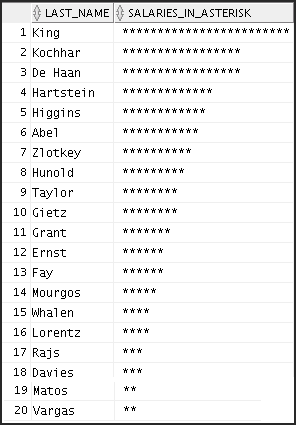
**Note:** Because this query depends on the date when it was executed, the values in the

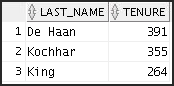
MONTHS\_WORKED column will differ for you.

Practices for Lesson 4: Using Single-Row Functions to Customize Output

Chapter 4 - Page 5



1. Create a query that displays the employees’ last names, and indicates the amounts of their salaries with asterisks. Each asterisk signifies a thousand dollars. Sort the data in descending order of salary. Label the column SALARIES\_IN\_ASTERISK.
2. Create a query to display the last name and the number of weeks employed for all employees in department 90. Label the number of weeks column as TENURE. Truncate the number of weeks value to 0 decimal places. Show the records in descending order of the employee’s tenure.

**Note:** The TENURE value will differ because it depends on the date on which you run the query.

Practices for Lesson 4: Using Single-Row Functions to Customize Output

Chapter 4 - Page 7

# Solution 4-1: Using Single-Row Functions to Customize Output

1. Write a query to display the system date. Label the column Date.

**Note:** If your database is remotely located in a different time zone, the output will be the date for the operating system on which the database resides.

SELECT sysdate "Date"

FROM dual;

1. The HR department needs a report to display the employee number, last name, salary, and salary increased by 15.5% (expressed as a whole number) for each employee. Label the column New Salary. Save your SQL statement in a file named lab\_04\_02.sql.

SELECT employee\_id, last\_name, salary, ROUND(salary \* 1.155, 0) "New Salary"

FROM employees;

1. Run your query in the file lab\_04\_02.sql.

SELECT employee\_id, last\_name, salary, ROUND(salary \* 1.155, 0) "New Salary"

FROM employees;

1. Modify your query in lab\_04\_02.sql to add a column that subtracts the old salary from the new salary. Label the column Increase. Save the contents of the file as lab\_04\_04.sql. Run the revised query.

SELECT employee\_id, last\_name, salary, ROUND(salary \* 1.155, 0) "New Salary",

ROUND(salary \* 1.155, 0) - salary "Increase"

FROM employees;

1. Perform the following tasks:
   1. Write a query that displays the last name (with the first letter in uppercase and all the other letters in lowercase) and the length of the last name for all employees whose name starts with the letters “A,” or “M.” Give each column an appropriate label. Sort the results by the employees’ last names.

SELECT INITCAP(last\_name) "Name", LENGTH(last\_name) "Length"

FROM employees

WHERE last\_name LIKE 'M%' OR last\_name LIKE 'A%'

ORDER BY last\_name;

Practices for Lesson 4: Using Single-Row Functions to Customize Output

Chapter 4 - Page 8

* 1. Rewrite the query so that the user is prompted to enter the letter that starts the last name. For example, if the user enters H (capitalized) when prompted for a letter, the output should show all employees whose last names start with the letter “H.”

SELECT INITCAP(last\_name) "Name", LENGTH(last\_name) "Length"

FROM employees

WHERE last\_name LIKE '&start\_letter%'

ORDER BY last\_name;

* 1. Modify the query such that the case of the letter that is entered does not affect the output. The entered letter must be capitalized before being processed by the SELECT query.

SELECT INITCAP(last\_name) "Name", LENGTH(last\_name) "Length"

FROM employees

WHERE last\_name LIKE UPPER('&start\_letter%' ) ORDER BY last\_name;

If you have time, complete the following exercises:

1. The HR department wants to find the duration of employment for each employee. For each employee, display the last name and calculate the number of months between today and the date on which the employee was hired. Label the column MONTHS\_WORKED. Order your results by the number of months employed. The number of months must be rounded to the closest whole number.

**Note:** Because this query depends on the date when it was executed, the values in the

MONTHS\_WORKED column will differ for you.

SELECT last\_name, ROUND(MONTHS\_BETWEEN(

SYSDATE, hire\_date)) MONTHS\_WORKED FROM employees

ORDER BY months\_worked;

1. Create a query to display the last name and salary for all employees. Format the salary to be 15 characters long, left-padded with the $ symbol. Label the column SALARY.

SELECT last\_name,

LPAD(salary, 15, '$') SALARY

FROM employees;

1. Create a query that displays employees’ last names, and indicates the amounts of their salaries with asterisks. Each asterisk signifies a thousand dollars. Sort the data in descending order of salary. Label the column SALARIES\_IN\_ASTERISK.

SELECT last\_name,

rpad(' ', (salary/1000)+1, '\*') SALARIES\_IN\_ASTERISK

FROM employees ORDER BY salary DESC;

Practices for Lesson 4: Using Single-Row Functions to Customize Output

Chapter 4 - Page 9

1. Create a query to display the last name and the number of weeks employed for all employees in department 90. Label the number of weeks column as TENURE. Truncate the number of weeks value to 0 decimal places. Show the records in descending order of the employee’s tenure.

**Note:** The TENURE value will differ because it depends on the date when you run the query.

SELECT last\_name, trunc((SYSDATE-hire\_date)/7) AS TENURE FROM employees

WHERE department\_id = 90

ORDER BY TENURE DESC;

Practices for Lesson 4: Using Single-Row Functions to Customize Output

Chapter 4 - Page 10

T

**Practices for Lesson 5:**

**Using Conversion Functions and Conditional Expressions**

**Chapter 5**

# Practices for Lesson 5: Overview

### Practice Overview

This practice covers the following topics:

* Creating queries that use the TO\_CHAR and TO\_DATE functions
* Creating queries that use conditional expressions such as CASE , searched CASE, and

DECODE

Practices for Lesson 5: Using Conversion Functions and Conditional Expressions

Chapter 5 - Page 2

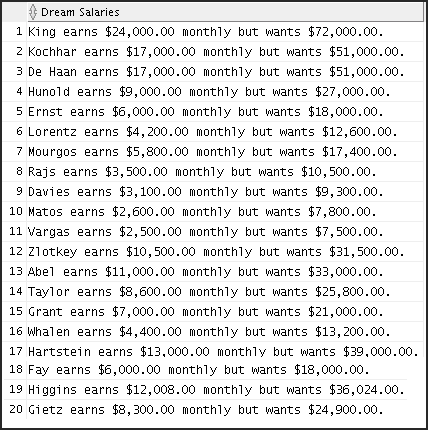
# Practice 5-1: Using Conversion Functions and Conditional Expressions

### Overview

In this practice, you use the TO\_CHAR and TO\_DATE functions, and conditional expressions such as CASE, searched CASE, and DECODE.

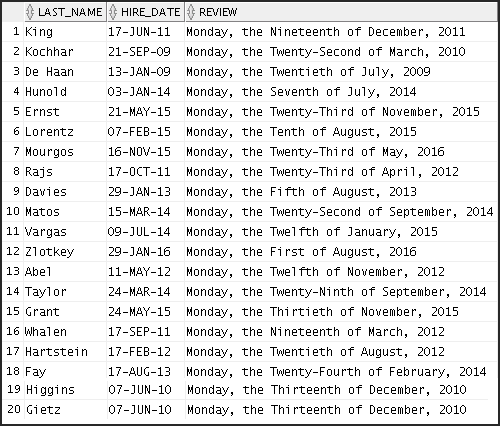
### Tasks

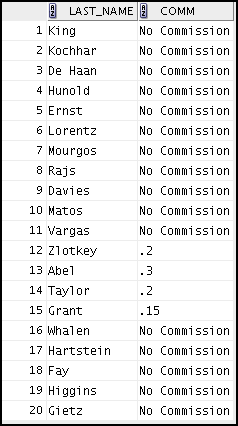
1. Create a report that produces the following for each employee:

<employee last name> earns <salary> monthly but wants <3 times salary.>. Label the column Dream Salaries.

Practices for Lesson 5: Using Conversion Functions and Conditional Expressions

Chapter 5 - Page 3

1. Display each employee’s last name, hire date, and salary review date, which is the first Monday after six months of service. Label the column REVIEW. Format the dates to appear in a format that is similar to “Monday, the Thirty-First of July, 2000.”
2. Create a query that displays employees’ last names and commission amounts. If an employee does not earn commission, show “No Commission.” Label the column COMM.



Practices for Lesson 5: Using Conversion Functions and Conditional Expressions

Chapter 5 - Page 4

1. Using the CASE function, write a query that displays the grade of all employees based on the value of the JOB\_ID column, using the following data:

##### Job Grade

AD\_PRES A

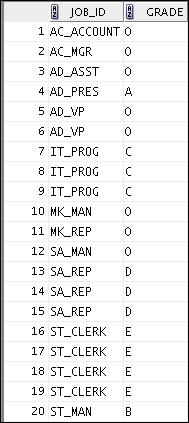
ST\_MAN B

IT\_PROG C

SA\_REP D

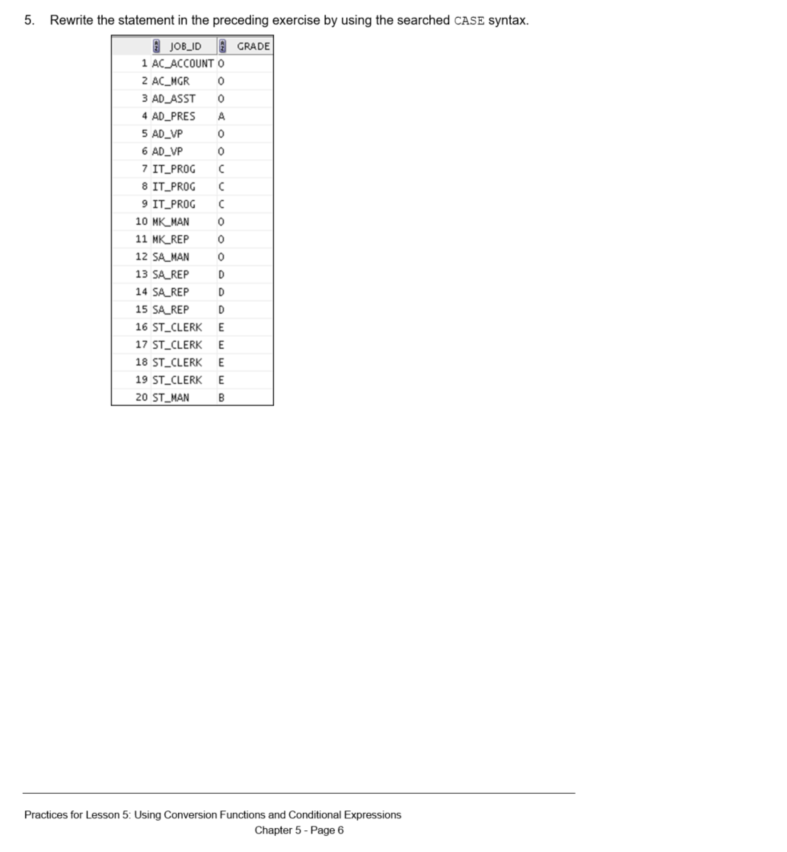
ST\_CLERK E

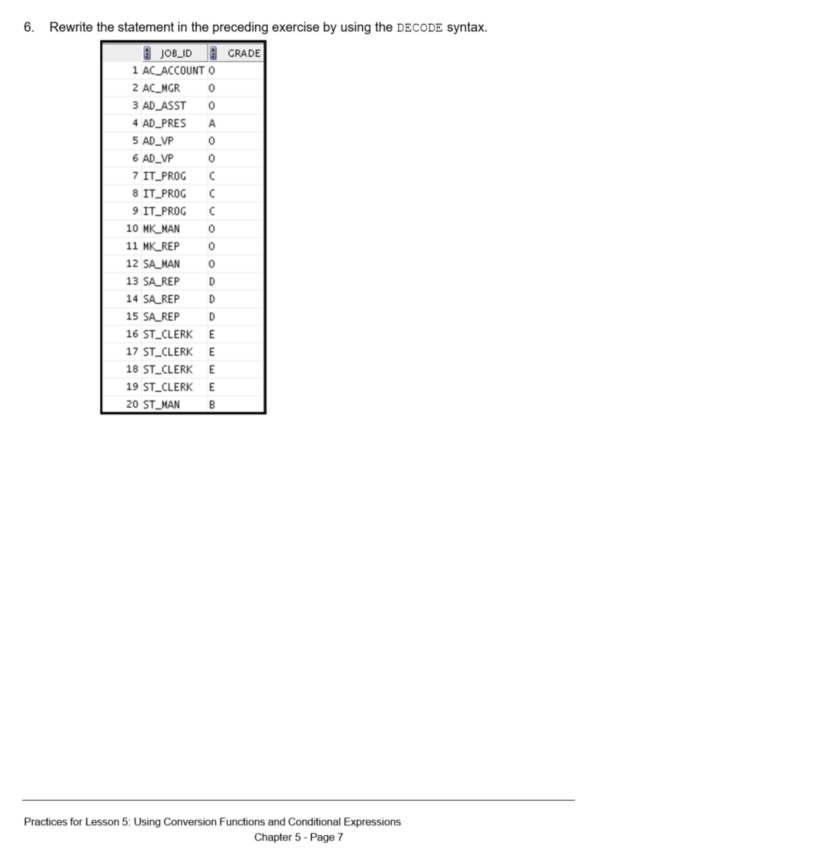
None of the above 0



Practices for Lesson 5: Using Conversion Functions and Conditional Expressions

Chapter 5 - Page 5





# Solution 5-1: Using Conversion Functions and Conditional Expressions

1. Create a report that produces the following for each employee:

<employee last name> earns <salary> monthly but wants <3 times salary.>. Label the column Dream Salaries.

SELECT last\_name || ' earns '

|| TO\_CHAR(salary, 'fm$99,999.00')

|| ' monthly but wants '

|| TO\_CHAR(salary \* 3, 'fm$99,999.00')

|| '.' "Dream Salaries"

FROM employees;

1. Display each employee’s last name, hire date, and salary review date, which is the first Monday after six months of service. Label the column REVIEW. Format the dates to appear in a format that is similar to “Monday, the Thirty-First of July, 2000.”

SELECT last\_name, hire\_date, TO\_CHAR(NEXT\_DAY(ADD\_MONTHS(hire\_date, 6),'MONDAY'),

'fmDay, "the" Ddspth "of" Month, YYYY') REVIEW FROM employees;

1. Create a query that displays employees’ last names and commission amounts. If an employee does not earn commission, show “No Commission.” Label the column COMM.

SELECT last\_name,

NVL(TO\_CHAR(commission\_pct), 'No Commission') COMM FROM employees;

1. Using the CASE function, write a query that displays the grade of all employees based on the value of the JOB\_ID column, using the following data:

##### Job Grade

AD\_PRES A

ST\_MAN B

IT\_PROG C

SA\_REP D

ST\_CLERK E

None of the above 0

Practices for Lesson 5: Using Conversion Functions and Conditional Expressions

Chapter 5 - Page 8

SELECT job\_id, CASE job\_id

WHEN 'ST\_CLERK' THEN 'E' WHEN 'SA\_REP'

WHEN 'IT\_PROG'

WHEN 'ST\_MAN' WHEN 'AD\_PRES'

ELSE '0'

FROM employees;

END

THEN 'D' THEN 'C' THEN 'B' THEN 'A'

GRADE

1. Rewrite the statement in the preceding exercise by using the searched CASE syntax.

SELECT job\_id, CASE

WHEN job\_id = 'ST\_CLERK' THEN 'E' WHEN job\_id = 'SA\_REP' THEN 'D' WHEN job\_id = 'IT\_PROG' THEN 'C' WHEN job\_id = 'ST\_MAN' THEN 'B'

WHEN job\_id = 'AD\_PRES' THEN 'A'

ELSE '0'

FROM employees;

END

GRADE

1. Rewrite the statement in the preceding exercise by using the DECODE syntax.

SELECT job\_id, decode (job\_id,

'ST\_CLERK', 'E',

'SA\_REP', 'D',

'IT\_PROG', 'C',

'ST\_MAN', 'B',

'AD\_PRES', 'A', '0')GRADE

FROM employees;

Practices for Lesson 5: Using Conversion Functions and Conditional Expressions

Chapter 5 - Page 9

**Practices for Lesson 6:**

**Reporting Aggregated Data Using the Group Functions**

**Chapter 6**

# Practices for Lesson 6: Overview

### Practice Overview

This practice covers the following topics:

* Writing queries that use group functions
* Grouping by rows to achieve multiple results
* Restricting groups by using the HAVING clause

Practices for Lesson 6: Reporting Aggregated Data Using the Group Functions

Chapter 6 - Page 2

# Practice 6-1: Reporting Aggregated Data by Using Group Functions

### Overview

In this practice, you use group functions and select groups of data.

### Tasks

Determine the validity of the following statements. Circle either True or False.

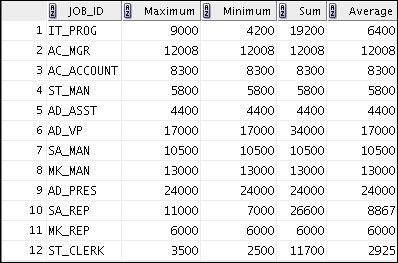
1. Group functions work across many rows to produce one result per group. True/False
2. Group functions include nulls in calculations. True/False
3. The WHERE clause restricts rows before inclusion in a group calculation. True/False

The HR department needs the following reports:

1. Find the highest, lowest, sum, and average salary of all employees. Label the columns Maximum, Minimum, Sum, and Average, respectively. Round your results to the nearest whole number. Save your SQL statement as lab\_06\_04.sql. Run the query.



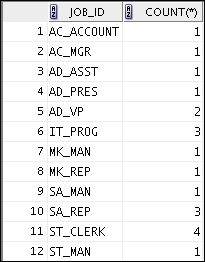
1. Modify the query in lab\_06\_04.sql to display the minimum, maximum, sum, and average salary for each job type. Save lab\_06\_04.sql as lab\_06\_05.sql. Run the statement in lab\_06\_05.sql.



Practices for Lesson 6: Reporting Aggregated Data Using the Group Functions

Chapter 6 - Page 3

1. Write a query to display the number of people with the same job.



Generalize the query so that a user in the HR department is prompted for a job title. Save the script to a file named lab\_06\_06.sql. Run the query. Enter IT\_PROG when prompted.



1. Determine the number of managers without listing them. Label the column Number of Managers.

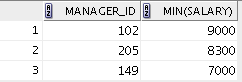
**Hint:** Use the MANAGER\_ID column to determine the number of managers.

1. Find the difference between the highest and lowest salaries. Label the column

DIFFERENCE.

If you have time, complete the following exercises:

1. Create a report to display the manager number and the salary of the lowest-paid employee for that manager. Exclude anyone whose manager is not known. Exclude any groups where the minimum salary is $6,000 or less. Sort the output in descending order of salary.

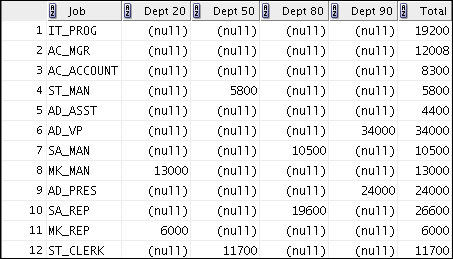


Practices for Lesson 6: Reporting Aggregated Data Using the Group Functions

Chapter 6 - Page 4

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

1. Create a query to display the total number of employees and, of that total, the number of employees hired in 2009, 2010, 2011, and 2012. Create appropriate column headings.
2. Create a matrix query to display the job, the salary for that job based on the department numbers 20, 50, 80, and 90, and the total salary for that job. Ensure to give each column an appropriate heading.



Practices for Lesson 6: Reporting Aggregated Data Using the Group Functions

Chapter 6 - Page 5

# Solution 6-1: Reporting Aggregated Data by Using Group Functions

Determine the validity of the following statements. Circle either True or False.

1. Group functions work across many rows to produce one result per group.

**True/**False

1. Group functions include nulls in calculations. True/**False**
2. The WHERE clause restricts rows before inclusion in a group calculation.

**True**/False

The HR department needs the following reports:

1. Find the highest, lowest, sum, and average salary of all employees. Label the columns Maximum, Minimum, Sum, and Average, respectively. Round your results to the nearest whole number. Save your SQL statement as lab\_06\_04.sql. Run the query.

SELECT ROUND(MAX(salary),0) "Maximum", ROUND(MIN(salary),0) "Minimum", ROUND(SUM(salary),0) "Sum",

ROUND(AVG(salary),0) "Average" FROM employees;

1. Modify the query in lab\_06\_04.sql to display the minimum, maximum, sum, and average salary for each job type. Save lab\_06\_04.sql as lab\_06\_05.sql. Run the statement in lab\_06\_05.sql.

SELECT job\_id, ROUND(MAX(salary),0) "Maximum",

ROUND(MIN(salary),0) "Minimum", ROUND(SUM(salary),0) "Sum", ROUND(AVG(salary),0) "Average"

FROM employees

GROUP BY job\_id;

1. Write a query to display the number of people with the same job.

SELECT job\_id, COUNT(\*) FROM employees

GROUP BY job\_id;

Generalize the query so that a user in the HR department is prompted for a job title. Save the script to a file named lab\_06\_06.sql. Run the query. Enter IT\_PROG when prompted and click OK.

SELECT job\_id, COUNT(\*) FROM employees

WHERE job\_id = '&job\_title' GROUP BY job\_id;

1. Determine the number of managers without listing them. Label the column Number of Managers.

**Hint:** Use the MANAGER\_ID column to determine the number of managers.

SELECT COUNT(DISTINCT manager\_id) "Number of Managers" FROM employees;

Practices for Lesson 6: Reporting Aggregated Data Using the Group Functions

Chapter 6 - Page 6

1. Find the difference between the highest and lowest salaries. Label the column

DIFFERENCE.

SELECT MAX(salary) - MIN(salary) DIFFERENCE FROM employees;

If you have time, complete the following exercises:

1. Create a report to display the manager number and the salary of the lowest-paid employee for that manager. Exclude anyone whose manager is not known. Exclude any groups where the minimum salary is $6,000 or less. Sort the output in descending order of salary.

SELECT manager\_id, MIN(salary) FROM employees

WHERE manager\_id IS NOT NULL GROUP BY manager\_id

HAVING MIN(salary) > 6000

ORDER BY MIN(salary) DESC;

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

1. Create a query that displays the total number of employees and, of that total, the number of employees hired in 2009, 2010, 2011, and 2012. Create appropriate column headings.

SELECT COUNT(\*) total,

SUM(DECODE(TO\_CHAR(hire\_date, 'YYYY'),2009,1,0))"2009", SUM(DECODE(TO\_CHAR(hire\_date, 'YYYY'),2010,1,0))"2010", SUM(DECODE(TO\_CHAR(hire\_date, 'YYYY'),2011,1,0))"2011", SUM(DECODE(TO\_CHAR(hire\_date, 'YYYY'),2012,1,0))"2012"

FROM employees;

1. Create a matrix query to display the job, the salary for that job based on the department numbers 20, 50, 80, and 90, and the total salary for that job. Ensure to give each column an appropriate heading.

SELECT job\_id "Job",

SUM(DECODE(department\_id , 20, salary)) "Dept 20",

SUM(DECODE(department\_id , 50, salary)) "Dept 50",

SUM(DECODE(department\_id , 80, salary)) "Dept 80",

SUM(DECODE(department\_id , 90, salary)) "Dept 90", SUM(salary) "Total"

FROM employees GROUP BY job\_id;

Practices for Lesson 6: Reporting Aggregated Data Using the Group Functions

Chapter 6 - Page 7

T

**Practices for Lesson 7:**

**Displaying Data from Multiple Tables Using Joins**

**Chapter 7**

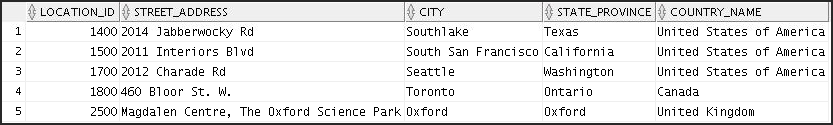


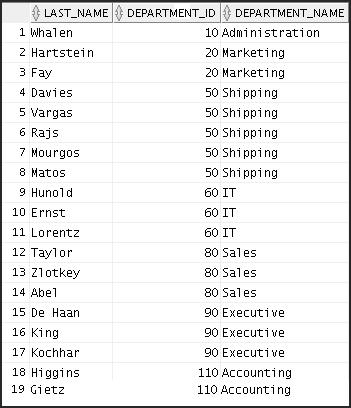
# Practice 7-1: Displaying Data from Multiple Tables by Using Joins

### Overview

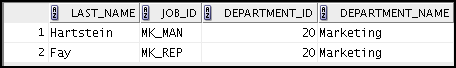
In this practice, you extract data from multiple tables using SQL:1999–compliant joins.

### Tasks

1. Write a query for the HR department to produce the addresses of all the departments. Use the LOCATIONS and COUNTRIES tables. Show the location ID, street address, city, state or province, and country in the output. Use a NATURAL JOIN to produce the results.
2. The HR department needs a report of all employees with corresponding departments. Write a query to display the last name, department number, and department name for these employees.

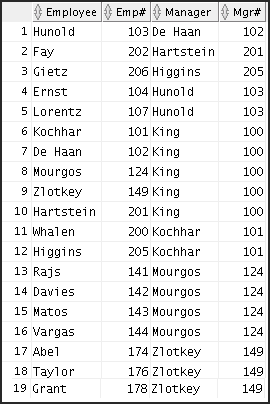


1. The HR department needs a report of employees in Toronto. Display the last name, job, department number, and the department name for all employees who work in Toronto.



Practices for Lesson 7: Displaying Data from Multiple Tables Using Joins

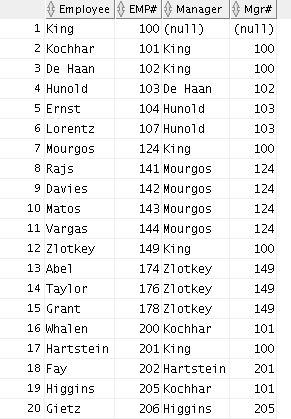
Chapter 7 - Page 3

1. Create a report to display employees’ last names and employee numbers along with their managers’ last names and manager numbers. Label the columns Employee, Emp#, Manager, and Mgr#, respectively. Save your SQL statement as lab\_07\_04.sql. Run the query.

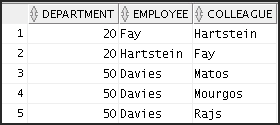
Practices for Lesson 7: Displaying Data from Multiple Tables Using Joins

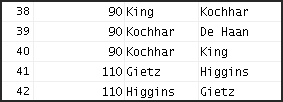
Chapter 7 - Page 4

1. Modify lab\_07\_04.sql to display all employees, including King, who has no manager. Order the results by employee number. Save your SQL statement as lab\_07\_05.sql. Run the query in lab\_07\_05.sql.

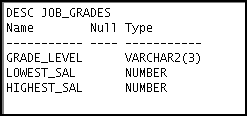


1. Create a report for the HR department that displays employee last names, department numbers, and all the employees who work in the same department as a given employee. Give each column an appropriate label. Save the script to a file named lab\_07\_06.sql.



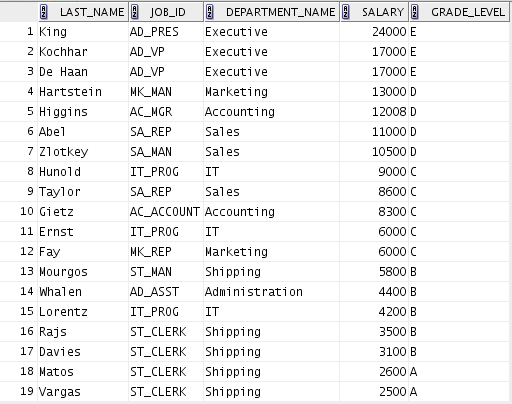
…

1. The HR department needs a report on job grades and salaries. To familiarize yourself with the JOB\_GRADES table, first show the structure of the JOB\_GRADES table. Then create a query that displays the name, job, department name, salary, and grade for all employees.

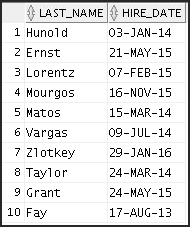


Practices for Lesson 7: Displaying Data from Multiple Tables Using Joins

Chapter 7 - Page 5

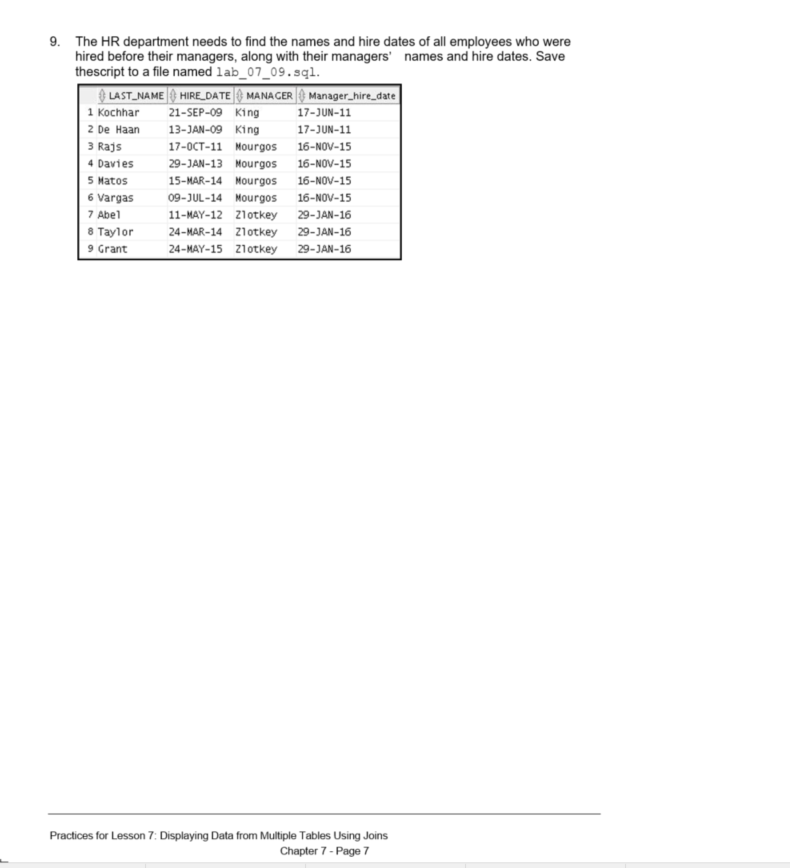


If you want an extra challenge, complete the following exercises:

1. The HR department wants to determine the names of all employees who were hired after Davies. Create a query to display the name and hire date of any employee hired after employee Davies.

Practices for Lesson 7: Displaying Data from Multiple Tables Using Joins

Chapter 7 - Page 6



# Solution 7-1: Displaying Data from Multiple Tables by Using Joins

1. Write a query for the HR department to produce the addresses of all the departments. Use the LOCATIONS and COUNTRIES tables. Show the location ID, street address, city, state or province, and country in the output. Use a NATURAL JOIN to produce the results.

SELECT location\_id, street\_address, city, state\_province, country\_name

FROM locations NATURAL JOIN countries;

1. The HR department needs a report of all employees with corresponding departments. Write a query to display the last name, department number, and department name for these employees.

SELECT last\_name, department\_id, department\_name FROM employees

JOIN departments

USING (department\_id);

1. The HR department needs a report of employees in Toronto. Display the last name, job, department number, and department name for all employees who work in Toronto.

SELECT e.last\_name, e.job\_id, e.department\_id, d.department\_name FROM employees e JOIN departments d

ON (e.department\_id = d.department\_id) JOIN locations l

USING (location\_id)

WHERE LOWER(l.city) = 'toronto';

1. Create a report to display employees’ last names and employee numbers along with their managers’ last names and manager numbers. Label the columns Employee, Emp#, Manager, and Mgr#, respectively. Save your SQL statement as lab\_07\_04.sql. Run the query.

SELECT w.last\_name "Employee", w.employee\_id "EMP#", m.last\_name "Manager", m.employee\_id "Mgr#"

FROM employees w JOIN employees m

ON (w.manager\_id = m.employee\_id);

1. Modify lab\_07\_04.sql to display all employees, including King, who has no manager. Order the results by employee number. Save your SQL statement as lab\_07\_05.sql. Run the query in lab\_07\_05.sql.

SELECT w.last\_name "Employee", w.employee\_id "EMP#", m.last\_name "Manager", m.employee\_id "Mgr#"

FROM employees w

LEFT OUTER JOIN employees m

ON (w.manager\_id = m.employee\_id) ORDER BY 2;

Practices for Lesson 7: Displaying Data from Multiple Tables Using Joins

Chapter 7 - Page 8

1. Create a report for the HR department that displays employee last names, department numbers, and all employees who work in the same department as a given employee. Give each column an appropriate label. Save the script to a file named lab\_07\_06.sql. Run the query.

SELECT e.department\_id department, e.last\_name employee, c.last\_name colleague

FROM employees e JOIN employees c

ON (e.department\_id = c.department\_id) WHERE e.employee\_id <> c.employee\_id

ORDER BY e.department\_id, e.last\_name, c.last\_name;

1. The HR department needs a report on job grades and salaries. To familiarize yourself with the JOB\_GRADES table, first show the structure of the JOB\_GRADES table. Then create a query that displays the name, job, department name, salary, and grade for all employees.

DESC JOB\_GRADES

/

SELECT e.last\_name, e.job\_id, d.department\_name, e.salary, j.grade\_level

FROM employees e JOIN departments d

ON (e.department\_id = d.department\_id) JOIN job\_grades j

ON (e.salary BETWEEN j.lowest\_sal AND j.highest\_sal);

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

1. The HR department wants to determine the names of all employees who were hired after Davies. Create a query to display the name and hire date of any employee hired after employee Davies.

SELECT e.last\_name, e.hire\_date

FROM employees e JOIN employees davies

ON (davies.last\_name = 'Davies') WHERE davies.hire\_date < e.hire\_date;

1. The HR department needs to find the names and hire dates of all employees who were hired before their managers, along with their managers’ names and hire dates. Save the script to a file named lab\_07\_09.sql.

SELECT w.last\_name, w.hire\_date, m.last\_name MANAGER, m.hire\_date "Manager\_hire\_date"

FROM

ON WHERE

employees w JOIN employees m

(w.manager\_id = m.employee\_id)

w.hire\_date <

m.hire\_date;

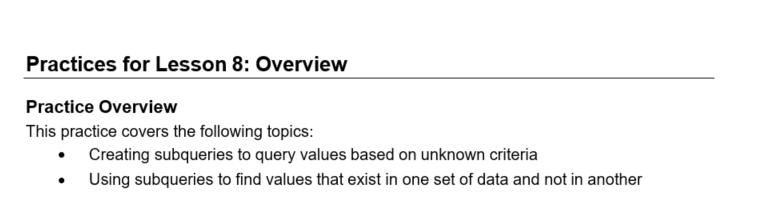
Practices for Lesson 7: Displaying Data from Multiple Tables Using Joins

Chapter 7 - Page 9

**Practices for Lesson 8:**

**Using Subqueries to Solve Queries**

**Chapter 8**



# Practice 8-1: Using Subqueries to Solve Queries

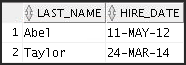
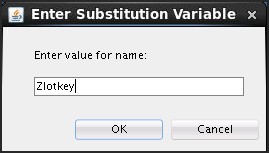
### Overview

In this practice, you write complex queries using nested SELECT statements.

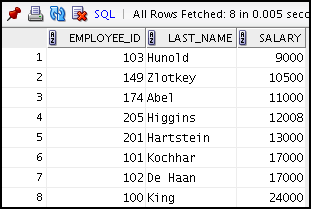
For practice questions, you may want to create the inner query first. Make sure that it runs and produces the data that you anticipate before you code the outer query.

### Tasks

1. The HR department needs a query that prompts the user for an employee’s last name. The query then displays the last name and hire date of any employee in the same department as the employee whose name the user supplies (excluding that employee). For example, if the user enters Zlotkey, find all employees who work with Zlotkey (excluding Zlotkey).



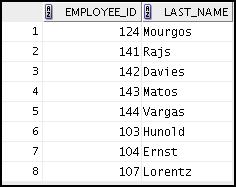
1. Create a report that displays the employee number, last name, and salary of all employees who earn more than the average salary. Sort the results in ascending order by salary.



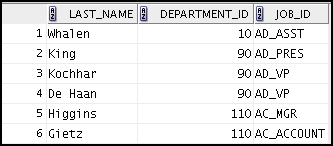
Practices for Lesson 8: Using Subqueries to Solve Queries

Chapter 8 - Page 3

1. Write a query that displays the employee number and last name of all employees who work in a department with any employee whose last name contains the letter “u.” Save your SQL statement as lab\_08\_03.sql. Run your query.

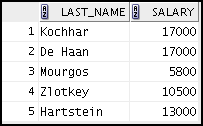


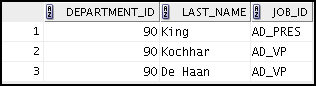
1. The HR department needs a report that displays the last name, department number, and job ID of all employees whose department location ID is 1700.



Modify the query so that the user is prompted for a location ID. Save this to a file named

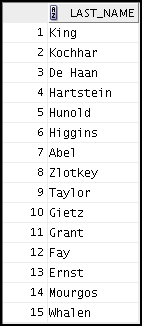
lab\_08\_04.sql.

1. Create a report for HR that displays the last name and salary of every employee who reports to King.
2. Create a report for HR that displays the department number, last name, and job ID for every employee in the Executive department.



Practices for Lesson 8: Using Subqueries to Solve Queries

Chapter 8 - Page 4

1. Create a report that displays a list of all employees whose salary is more than the salary of any employee from department 60.

If you have time, complete the following exercise:

1. Modify the query in lab\_08\_03.sql to display the employee number, last name, and salary of all employees who earn more than the average salary, and who work in a department with any employee whose last name contains the letter “u.” Save lab\_08\_03.sql as lab\_08\_08.sql again. Run the statement in lab\_08\_08.sql.

Practices for Lesson 8: Using Subqueries to Solve Queries

Chapter 8 - Page 5

# Solution 8-1: Using Subqueries to Solve Queries

1. The HR department needs a query that prompts the user for an employee’s last name. The query then displays the last name and hire date of any employee in the same department as the employee whose name the user supplies (excluding that employee). For example, if the user enters Zlotkey, find all employees who work with Zlotkey (excluding Zlotkey).

**--Execute the UNDEFINE command to remove a variable**

UNDEFINE name

**-- Execute the below SELECT statements to retrieve the values from employees table**

SELECT last\_name, hire\_date FROM employees

WHERE department\_id = (SELECT department\_id

FROM employees

WHERE last\_name = '&&name')

AND last\_name <> '&name';

**Note:** UNDEFINE and SELECT are individual queries; execute them one after the other or press Ctrl + A + F9 to run them together.

1. Create a report that displays the employee number, last name, and salary of all employees who earn more than the average salary. Sort the results in ascending order by salary.

SELECT employee\_id, last\_name, salary FROM employees

WHERE salary > (SELECT AVG(salary)

FROM employees)

ORDER BY salary;

1. Write a query that displays the employee number and last name of all employees who work in a department with any employee whose last name contains the letter “u.” Save your SQL statement as lab\_08\_03.sql. Run your query.

SELECT employee\_id, last\_name FROM employees

WHERE department\_id IN (SELECT department\_id

FROM employees

WHERE last\_name like '%u%');

1. The HR department needs a report that displays the last name, department number, and job ID of all employees whose department location ID is 1700.

SELECT last\_name, department\_id, job\_id FROM employees

WHERE department\_id IN (SELECT department\_id

FROM departments

WHERE location\_id = 1700);

Practices for Lesson 8: Using Subqueries to Solve Queries

Chapter 8 - Page 6

Modify the query so that the user is prompted for a location ID. Save this to a file named

lab\_08\_04.sql.

SELECT last\_name, department\_id, job\_id FROM employees

WHERE department\_id IN (SELECT department\_id

FROM departments WHERE location\_id =

&location);

1. Create a report for HR that displays the last name and salary of every employee who reports to King.

SELECT last\_name, salary FROM employees

WHERE manager\_id = (SELECT employee\_id

FROM employees

WHERE last\_name = 'King');

1. Create a report for HR that displays the department number, last name, and job ID for every employee in the Executive department.

SELECT department\_id, last\_name, job\_id FROM employees

WHERE department\_id IN (SELECT department\_id

FROM departments

WHERE department\_name = 'Executive');

1. Create a report that displays a list of all employees whose salary is more than the salary of any employee from department 60.

SELECT last\_name FROM employees WHERE salary > ANY (SELECT salary

FROM employees

WHERE department\_id=60);

Practices for Lesson 8: Using Subqueries to Solve Queries

Chapter 8 - Page 7

If you have time, complete the following exercise:

1. Modify the query in lab\_08\_03.sql to display the employee number, last name, and salary of all employees who earn more than the average salary and who work in a department with any employee whose last name contains the letter “u.” Save lab\_08\_03.sql to lab\_08\_08.sql again. Run the statement in lab\_08\_08.sql.

SELECT employee\_id, last\_name, salary FROM employees

WHERE department\_id IN (SELECT department\_id

FROM employees

WHERE last\_name like '%u%') AND salary > (SELECT AVG(salary)

FROM employees);

Practices for Lesson 8: Using Subqueries to Solve Queries

Chapter 8 - Page 8

T

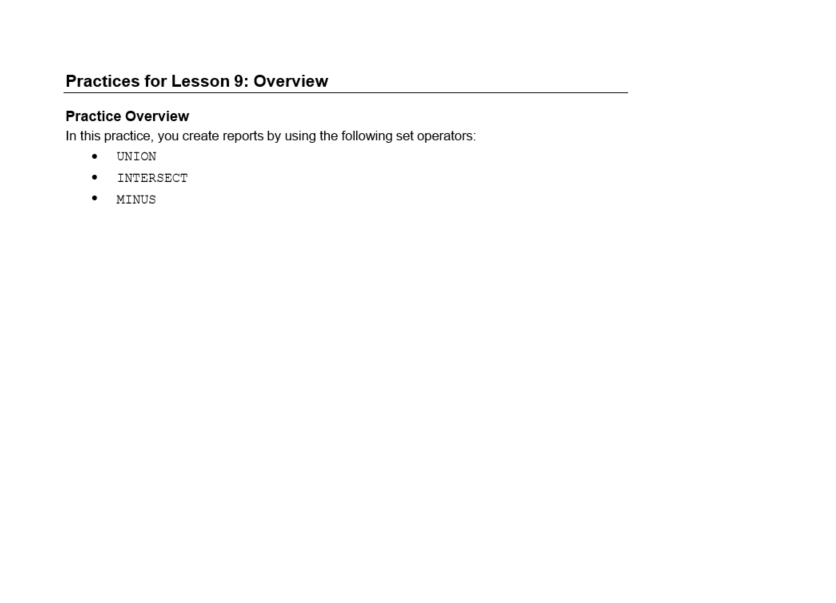
**Practices for Lesson 9:**

**Using the Set Operators**

**Chapter 9**

Practices for Lesson 9: Using the Set Operators

Chapter 9 - Page 2

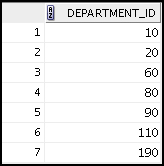


# Practice 9-1: Using Set Operators

### Overview

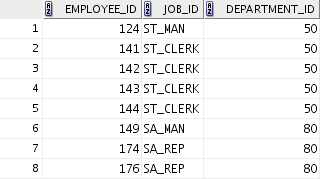
In this practice, you write queries by using the set operators UNION, INTERSECT, and MINUS.

### Tasks

1. The HR department needs a list of department IDs for departments that do not contain the job ID ST\_CLERK. Use the set operators to create this report.
2. The HR department needs a list of countries that have no departments located in them. Display the country IDs and the names of the countries. Use the set operators to create this report.



1. Produce a list of all the employees who work in departments 50 and 80. Display the employee ID, job ID, and department ID by using the set operators.



1. Create a report that lists the details of all employees who are sales representatives and are currently working in the sales department.

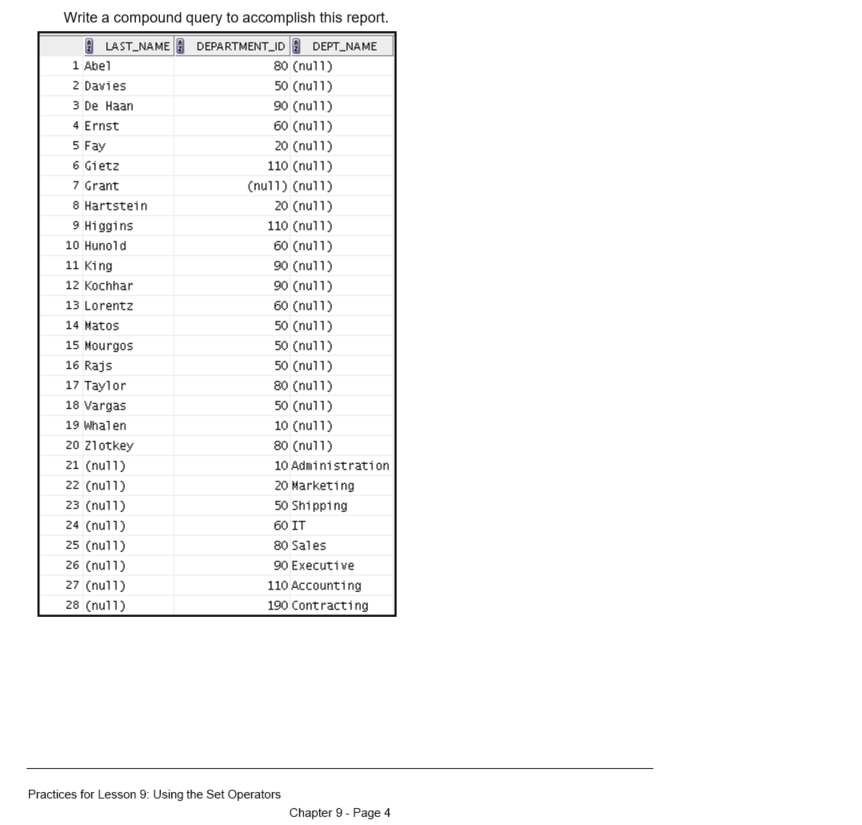


1. The HR department needs a report with the following specifications:
   * Last names and department IDs of all employees from the EMPLOYEES table, regardless of whether or not they belong to a department
   * Department IDs and department names of all departments from the DEPARTMENTS

table, regardless of whether or not they have employees working in them

Practices for Lesson 9: Using the Set Operators

Chapter 9 - Page 3



# Solution 9-1: Using Set Operators

1. The HR department needs a list of department IDs for departments that do not contain the job ID ST\_CLERK. Use the set operators to create this report.

SELECT department\_id FROM departments MINUS

SELECT department\_id FROM employees

WHERE job\_id = 'ST\_CLERK';

1. The HR department needs a list of countries that have no departments located in them. Display the country IDs and the names of the countries. Use the set operators to create this report.

SELECT country\_id,country\_name FROM countries

MINUS

SELECT l.country\_id,c.country\_name FROM locations l JOIN countries c ON (l.country\_id = c.country\_id) JOIN departments d

ON d.location\_id=l.location\_id;

1. Produce a list of all the employees who work in departments 50 and 80. Display the employee ID, job ID, and department ID by using the set operators.

SELECT employee\_id, job\_id, department\_id FROM EMPLOYEES

WHERE department\_id=50 UNION ALL

SELECT employee\_id, job\_id, department\_id FROM EMPLOYEES

WHERE department\_id=80;

1. Create a report that lists the detail of all employees who are sales representatives and are currently working in the sales department.

SELECT EMPLOYEE\_ID FROM EMPLOYEES

WHERE JOB\_ID='SA\_REP' INTERSECT

SELECT EMPLOYEE\_ID FROM EMPLOYEES

WHERE DEPARTMENT\_ID=80;

Practices for Lesson 9: Using the Set Operators

Chapter 9 - Page 5

1. The HR department needs a report with the following specifications:
   * Last names and department IDs of all employees from the EMPLOYEES table, regardless of whether or not they belong to a department
   * Department IDs and department names of all departments from the DEPARTMENTS

table, regardless of whether or not they have employees working in them Write a compound query to accomplish this report.

SELECT last\_name,department\_id,TO\_CHAR(null)dept\_name FROM employees

UNION

SELECT TO\_CHAR(null),department\_id,department\_name FROM departments;

Practices for Lesson 9: Using the Set Operators

Chapter 9 - Page 6

T

**Practices for Lesson 10:**

**Managing Tables Using DML Statements**

**Chapter 10**

# Practices for Lesson 10: Overview

### Lesson Overview

This practice covers the following topics:

* + Inserting rows into tables
  + Updating and deleting rows in a table
  + Controlling transactions

**Note:** Before starting this practice, execute the

Desktop/oracle19c-labs/sql-intro/labs/sql1/code\_ex /cleanup\_scripts/cleanup\_10.sql script.

Practices for Lesson 10: Managing Tables Using DML Statements

Chapter 10 - Page 2

# Practice 10-1: Managing Tables by Using DML Statements

### Overview

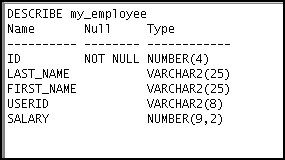
The HR department wants you to create SQL statements to insert, update, and delete employee data. As a prototype, you use the MY\_EMPLOYEE table before giving the statements to the HR department.

#### Note

* + For all DML statements, use the Run Script icon (or press F5) to execute the query. You can see the feedback messages on the Script Output tabbed page. For SELECT queries, continue to use the Execute Statement icon or press F9 to get the formatted output on the Results tabbed page.
  + Execute the cleanup\_10.sql script from Desktop/oracle19c-labs/sql-intro/labs/sql1/code\_ex

/cleanup\_scripts/ before performing the following tasks.

### Tasks

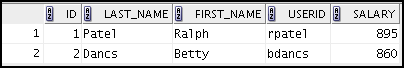
1. Run the lab\_10\_01.sql script from Desktop/oracle19c-labs/sql-intro/labs/sql1/labs/ to create the MY\_EMPLOYEE table. Describe the structure of the MY\_EMPLOYEE table to identify the column names.

Practices for Lesson 10: Managing Tables Using DML Statements

Chapter 10 - Page 3

1. Create an INSERT statement to add the *first row* of data to the MY\_EMPLOYEE table from the following sample data. Do not list the columns in the INSERT clause. *Do not enter all rows yet.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **LAST\_NAME** | **FIRST\_NAME** | **USERID** | **SALARY** |
| 1 | Patel | Ralph | rpatel | 895 |
| 2 | Dancs | Betty | bdancs | 860 |
| 3 | Biri | Ben | bbiri | 1100 |
| 4 | Newman | Chad | cnewman | 750 |
| 5 | Ropeburn | Audrey | aropebur | 1550 |

1. Populate the MY\_EMPLOYEE table with the second row of the sample data from the preceding list. This time, list the columns explicitly in the INSERT clause.
2. Confirm your addition to the table.
3. Write an INSERT statement in a dynamic reusable script file to load the next two rows of sample data into the MY\_EMPLOYEE table. The script should prompt for all the columns (ID, LAST\_NAME, FIRST\_NAME, USERID, and SALARY). Save this script to a lab\_10\_06.sql file.
4. Populate the table with the next two rows of the sample data listed in step 3 by running the

INSERT statement in the script that you created.

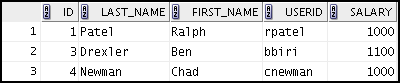
1. Confirm your additions to the table.
2. Make the data additions permanent.

Update and delete data in the MY\_EMPLOYEE table.

1. Change the last name of employee 3 to Drexler.
2. Change the salary to $1,000 for all employees who have a salary less than $900.

Practices for Lesson 10: Managing Tables Using DML Statements

Chapter 10 - Page 4

1. Verify your changes to the table.
2. Delete Betty Dancs from the MY\_EMPLOYEE table.
3. Confirm your changes to the table.
4. Commit all pending changes.

Control the data transaction to the MY\_EMPLOYEE table.

1. Populate the table with the last row of the sample data listed in step 3 by using the statements in the script that you created in step 6. Run the statements in the script.

**Note:** Perform the steps (17-23) in one session only.

1. Confirm your addition to the table.



1. Mark an intermediate point in the processing of the transaction.
2. Delete all the rows from the MY\_EMPLOYEE table.
3. Confirm that the table is empty.
4. Discard the most recent DELETE operation without discarding the earlier INSERT operation.
5. Confirm that the new row is still intact.
6. Make the data addition permanent.

If you have time, complete the following exercise:

1. Modify the lab\_10\_06.sql script such that the USERID is generated automatically by concatenating the first letter of the first name and the first seven characters of the last name. The generated USERID must be in lowercase. Therefore, the script should not prompt for the USERID. Save this script to a file named lab\_10\_24.sql.

Practices for Lesson 10: Managing Tables Using DML Statements

Chapter 10 - Page 5

1. Run the lab\_10\_24.sql script to insert the following record:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **LAST\_NAME** | **FIRST\_NAME** | **USERID** | **SALARY** |
| 6 | Anthony | Mark | manthony | 1230 |

1. Confirm that the new row was added with the correct USERID.

Practices for Lesson 10: Managing Tables Using DML Statements

Chapter 10 - Page 6

# Solution 10-1: Managing Tables by Using DML Statements

Insert data into the MY\_EMPLOYEE table.

1. Run the lab\_10\_01.sql script from Desktop/oracle19c-labs/sql-intro/labs/sql1/labs/ to create the MY\_EMPLOYEE table. Describe the structure of the MY\_EMPLOYEE table to identify the column names.

DESCRIBE my\_employee

1. Create an INSERT statement to add the first row of data to the MY\_EMPLOYEE table from the following sample data. Do not list the columns in the INSERT clause. *Do not enter all rows yet.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **LAST\_NAME** | **FIRST\_NAME** | **USERID** | **SALARY** |
| 1 | Patel | Ralph | rpatel | 895 |
| 2 | Dancs | Betty | bdancs | 860 |
| 3 | Biri | Ben | bbiri | 1100 |
| 4 | Newman | Chad | cnewman | 750 |
| 5 | Ropeburn | Audrey | aropebur | 1550 |

INSERT INTO my\_employee

VALUES (1, 'Patel', 'Ralph', 'rpatel', 895);

1. Populate the MY\_EMPLOYEE table with the second row of the sample data from the preceding list. This time, list the columns explicitly in the INSERT clause.

INSERT INTO my\_employee (id, last\_name, first\_name,

userid, salary) VALUES (2, 'Dancs', 'Betty', 'bdancs', 860);

1. Confirm your additions to the table.

SELECT \*

FROM my\_employee;

1. Write an INSERT statement in a dynamic reusable script file to load the next two rows of sample data into the MY\_EMPLOYEE table. The script should prompt for all the columns (ID, LAST\_NAME, FIRST\_NAME, USERID, and SALARY). Save this script to a file named lab\_10\_06.sql.

INSERT INTO my\_employee

VALUES (&id, '&last\_name', '&first\_name', '&userid', &salary);

Practices for Lesson 10: Managing Tables Using DML Statements

Chapter 10 - Page 7

1. Populate the table with the next two rows of the sample data listed in step 3 by running the

INSERT statement in the script that you created.

INSERT INTO my\_employee

VALUES (&id, '&last\_name', '&first\_name',

'&userid', &salary);

1. Confirm your additions to the table.

SELECT \*

FROM my\_employee;

1. Make the data additions permanent.

COMMIT;

Update and delete data in the MY\_EMPLOYEE table.

1. Change the last name of employee 3 to Drexler.

UPDATE my\_employee

SET last\_name = 'Drexler' WHERE id = 3;

1. Change the salary to $1,000 for all employees with a salary less than $900.

UPDATE my\_employee SET salary = 1000

WHERE salary < 900;

1. Verify your changes to the table.

SELECT \*

FROM my\_employee;

1. Delete Betty Dancs from the MY\_EMPLOYEE table.

DELETE

FROM my\_employee

WHERE last\_name = 'Dancs';

1. Confirm your changes to the table.

SELECT \*

FROM my\_employee;

1. Commit all pending changes.

COMMIT;

Control the data transaction to the MY\_EMPLOYEE table.

1. Populate the table with the last row of the sample data listed in step 3 by using the statements in the script that you created in step 6. Run the statements in the script.

INSERT INTO my\_employee

VALUES (&id, '&last\_name', '&first\_name', '&userid', &salary);

**Note:** Perform the steps (17-23) in one session only.

Practices for Lesson 10: Managing Tables Using DML Statements

Chapter 10 - Page 8

1. Confirm your addition to the table.

SELECT \*

FROM my\_employee;

1. Mark an intermediate point in the processing of the transaction.

SAVEPOINT step\_17;

1. Delete all the rows from the MY\_EMPLOYEE table.

DELETE

FROM my\_employee;

1. Confirm that the table is empty.

SELECT \*

FROM my\_employee;

1. Discard the most recent DELETE operation without discarding the earlier INSERT operation.

ROLLBACK TO step\_17;

1. Confirm that the new row is still intact.

SELECT \*

FROM my\_employee;

1. Make the data addition permanent.

COMMIT;

If you have time, complete the following exercise:

1. Modify the lab\_10\_06.sql script such that the USERID is generated automatically by concatenating the first letter of the first name and the first seven characters of the last name. The generated USERID must be in lowercase. The script should, therefore, not prompt for the USERID. Save this script to a file named lab\_10\_24.sql.

INSERT INTO my\_employee

VALUES (&id, '&&last\_name', '&&first\_name', lower(substr('&first\_name', 1, 1) ||

substr('&last\_name', 1, 7)), &salary);

UNDEFINE first\_name UNDEFINE last\_name

1. Run the lab\_10\_24.sql script to insert the following record:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **LAST\_NAME** | **FIRST\_NAME** | **USERID** | **SALARY** |
| 6 | Anthony | Mark | manthony | 1230 |

1. Confirm that the new row was added with the correct USERID.

SELECT \*

FROM my\_employee WHERE ID='6';

Practices for Lesson 10: Managing Tables Using DML Statements

Chapter 10 - Page 9

**Practices for Lesson 11:**

**Introduction to Data Definition Language**

**Chapter 11**

# Practices for Lesson 11: Overview

### Lesson Overview

This practice covers the following topics:

* Creating new tables
* Creating a new table by using the CREATE TABLE AS syntax
* Verifying that tables exist
* Altering tables
* Adding columns
* Dropping columns
* Setting a table to READ ONLY status
* Dropping tables

**Note:** Before starting this practice, execute the

Desktop/oracle19c-labs/sql-intro/labs/sql1/code\_ex/cleanup\_scripts/cleanup\_11.sql script.

Practices for Lesson 11: Introduction to Data Definition Language

Chapter 11 - Page 2

# Practice 11-1: Introduction to Data Definition Language

### Overview

In this practice, you create new tables by using the CREATE TABLE statement. Confirm that the new table was added to the database. You also learn to set the status of a table as READ ONLY, and then revert to READ WRITE. You use the ALTER TABLE command to modify table columns.

#### Notes

* For all DDL and DML statements, click the Run Script icon (or press F5) to execute the query in SQL Developer. Thus, you get to see the feedback messages on the Script Output tabbed page. For SELECT queries, continue to click the Execute Statement icon or press F9 to get the formatted output on the Results tabbed page.
* Execute the cleanup\_11.sql script from

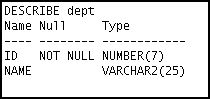
Desktop/oracle19c-labs/sql-intro/labs/sql1/code\_ex/cleanup\_scripts/cleanup\_11.sql

before performing the following tasks.

### Tasks

1. Create the DEPT table based on the following table instance chart. Save the statement in the lab\_11\_01.sql script, and then execute the statement in the script to create the table. Confirm that the table is created.

|  |  |  |
| --- | --- | --- |
| **Column Name** | ID | NAME |
| **Key Type** | Primary key |  |
| **Nulls/Unique** |  |  |
| **FK Table** |  |  |
| **FK Column** |  |  |
| **Data type** | NUMBER | VARCHAR2 |
| **Length** | 7 | 25 |

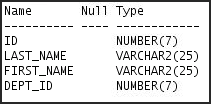


Practices for Lesson 11: Introduction to Data Definition Language

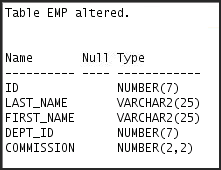
Chapter 11 - Page 3

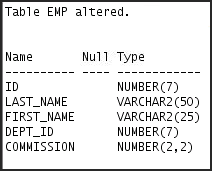
1. Create the EMP table based on the following table instance chart. Save the statement in the lab\_11\_02.sql script, and then execute the statement in the script to create the table. Confirm that the table is created.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Name** | ID | LAST\_NAME | FIRST\_NAME | DEPT\_ID |
| **Key Type** |  |  |  |  |
| **Nulls/Unique** |  |  |  |  |
| **FK Table** |  |  |  | DEPT |
| **FK Column** |  |  |  | ID |
| **Data type** | NUMBER | VARCHAR2 | VARCHAR2 | NUMBER |
| **Length** | 7 | 25 | 25 | 7 |



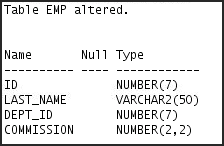
1. Modify the EMP table. Add a COMMISSION column of the NUMBER data type, with precision 2 and scale 2. Confirm your modification.

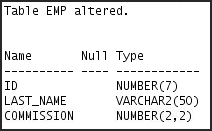


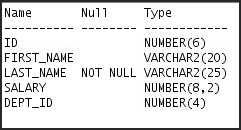
1. Modify the EMP table to allow for longer employee last names. Confirm your modification.

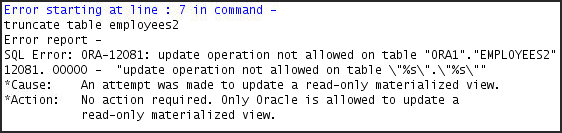
Practices for Lesson 11: Introduction to Data Definition Language

Chapter 11 - Page 4

1. Drop the FIRST\_NAME column from the EMP table. Confirm your modification by checking the description of the table.
2. In the EMP table, mark the DEPT\_ID column as UNUSED. Confirm your modification by checking the description of the table.



1. Drop all of the UNUSED columns from the EMP table.
2. Create the EMPLOYEES2 table based on the structure of the EMPLOYEES table. Include only the EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, SALARY, and DEPARTMENT\_ID columns. Name the columns in your new table ID, FIRST\_NAME, LAST\_NAME, SALARY, and DEPT\_ID, respectively. Confirm that the table is created.
3. Alter the status of the EMPLOYEES2 table to READ ONLY.
4. Try to delete all rows in the EMPLOYEES2 table.

**Note:** You will get the “Update operation not allowed on table” error message. You will not be allowed to delete data from the table because it is assigned a READ ONLY status.

Practices for Lesson 11: Introduction to Data Definition Language

Chapter 11 - Page 5

1. Revert the EMPLOYEES2 table to the READ WRITE status. Now try to truncate the rows again.

Now, because the table is assigned a READ WRITE status, you will be allowed to truncate rows in the table.

You should get the following messages:



1. Drop the EMP, DEPT, and EMPLOYEES2 table.

Practices for Lesson 11: Introduction to Data Definition Language

Chapter 11 - Page 6

# Solution 11-1: Introduction to Data Definition Language

1. Create the DEPT table based on the following table instance chart. Save the statement in a script called lab\_11\_01.sql, and then execute the statement in the script to create the table. Confirm that the table is created.

|  |  |  |
| --- | --- | --- |
| **Column Name** | ID | NAME |
| **Key Type** | Primary key |  |
| **Nulls/Unique** |  |  |
| **FK Table** |  |  |
| **FK Column** |  |  |
| **Data type** | NUMBER | VARCHAR2 |
| **Length** | 7 | 25 |

CREATE TABLE dept

(id NUMBER(7)CONSTRAINT department\_id\_pk PRIMARY KEY, name VARCHAR2(25));

To confirm that the table was created and to view its structure, issue the following command:

DESCRIBE dept;

1. Create the EMP table based on the following table instance chart. Save the statement in a script called lab\_11\_02.sql, and then execute the statement in the script to create the table. Confirm that the table is created.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Name** | ID | LAST\_NAME | FIRST\_NAME | DEPT\_ID |
| **Key Type** |  |  |  |  |
| **Nulls/Unique** |  |  |  |  |
| **FK Table** |  |  |  | DEPT |
| **FK Column** |  |  |  | ID |
| **Data type** | NUMBER | VARCHAR2 | VARCHAR2 | NUMBER |
| **Length** | 7 | 25 | 25 | 7 |

CREATE TABLE emp

(id NUMBER(7),

last\_name VARCHAR2(25), first\_name VARCHAR2(25), dept\_id NUMBER(7)

CONSTRAINT emp\_dept\_id\_FK REFERENCES dept (id)

);

Practices for Lesson 11: Introduction to Data Definition Language

Chapter 11 - Page 7

To confirm that the table was created and to view its structure:

DESCRIBE emp

1. Modify the EMP table. Add a COMMISSION column of the NUMBER data type, with precision 2 and scale 2. Confirm your modification.

ALTER TABLE emp

ADD commission NUMBER(2,2);

DESCRIBE emp

1. Modify the EMP table to allow for longer employee last names. Confirm your modification.

ALTER TABLE emp

MODIFY last\_name VARCHAR2(50);

DESCRIBE emp

1. Drop the FIRST\_NAME column from the EMP table. Confirm your modification by checking the description of the table.

ALTER TABLE emp

DROP COLUMN first\_name;

DESCRIBE emp

1. In the EMP table, mark the DEPT\_ID column as UNUSED. Confirm your modification by checking the description of the table.

ALTER TABLE emp

SET UNUSED (dept\_id);

DESCRIBE emp

1. Drop all of the UNUSED columns from the EMP table.

ALTER TABLE emp

DROP UNUSED COLUMNS;

1. Create the EMPLOYEES2 table based on the structure of the EMPLOYEES table. Include only the EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, SALARY, and DEPARTMENT\_ID columns. Name the columns in your new table ID, FIRST\_NAME, LAST\_NAME, SALARY, and DEPT\_ID, respectively. Confirm that the table is created.

CREATE TABLE employees2 AS

SELECT employee\_id id, first\_name, last\_name, salary, department\_id dept\_id

FROM employees;

DESCRIBE employees2

1. Alter the EMPLOYEES2 table status to READ ONLY.

ALTER TABLE employees2 READ ONLY;

Practices for Lesson 11: Introduction to Data Definition Language

Chapter 11 - Page 8

1. Try to delete all rows in the EMPLOYEES2 table.

**Note:** You will get the “Update operation not allowed on table” error message. You will not be allowed to delete data from the table because it is assigned a READ ONLY status.

TRUNCATE TABLE employees2;

1. Revert the EMPLOYEES2 table to the READ WRITE status. Now try to truncate the rows again.

Now, because the table is assigned a READ WRITE status, you will be allowed to truncate rows in the table.

ALTER TABLE employees2 READ WRITE;

TRUNCATE TABLE employees2; SELECT \* from employees2;

1. Drop the EMP, DEPT, and EMPLOYEES2 table.

**Note:** You can even drop a table that is in the READ ONLY status. To test this, alter the table again to READ ONLY status, and then issue the DROP TABLE command. The tables will be dropped.

DROP TABLE emp;

DROP TABLE dept;

DROP TABLE employees2;

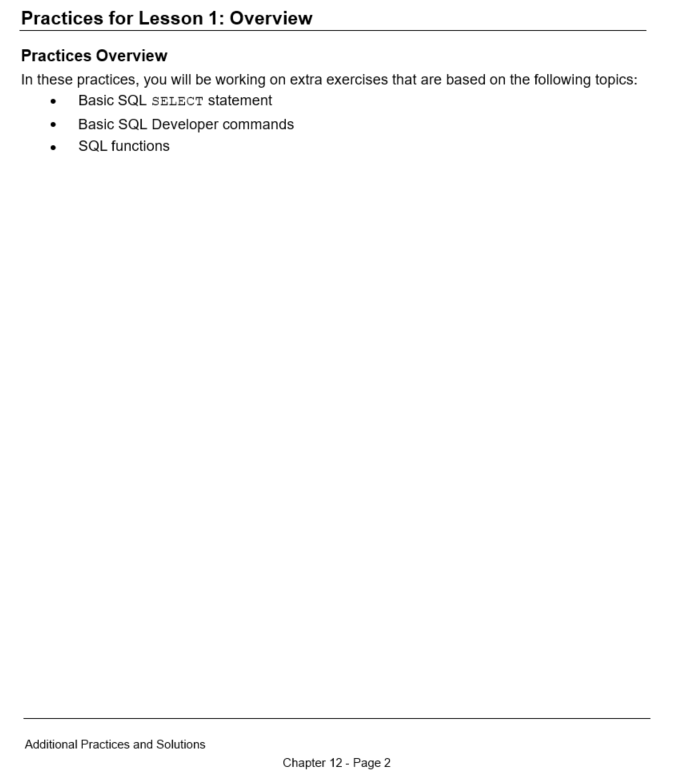
Practices for Lesson 11: Introduction to Data Definition Language

Chapter 11 - Page 9

T

**Additional Practices and Solutions**

**Chapter 12**

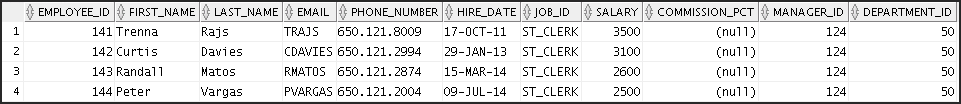


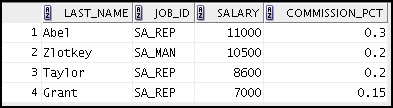
# Practice 1-1: Additional Practice

### Overview

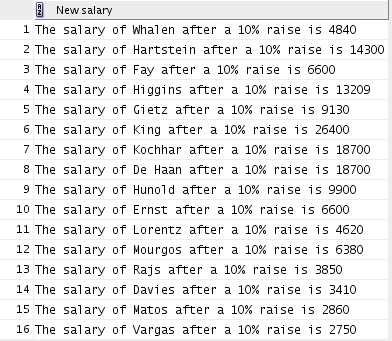
In this practice, exercises have been designed to be worked on after you have discussed the following topics: basic SQL SELECT statement, basic SQL Developer commands, and SQL functions.

### Tasks

1. The HR department needs to find data for all the clerks who were hired after 2010.
2. The HR department needs a report of employees who earn a commission. Show the last name, job, salary, and commission of these employees. Sort the data by salary in descending order.

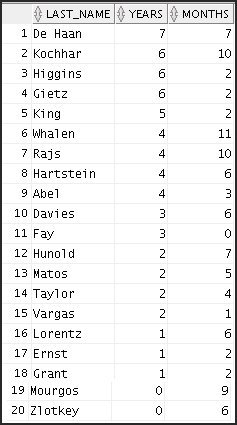


1. For budgeting purposes, the HR department needs a report on projected raises. The report should display those employees who have no commission, but who have a 10% raise in salary (round off the salaries).



Additional Practices and Solutions

Chapter 12 - Page 3

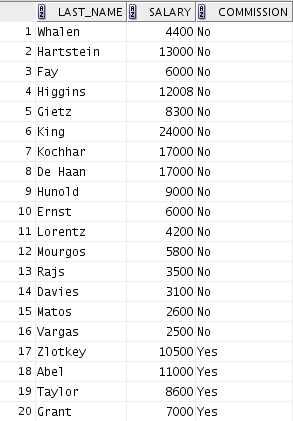
1. Create a report of employees and their duration of employment. Show the last names of all the employees together with the number of years and the number of completed months that they have been employed. Order the report by the duration of their employment. The employee who has been employed the longest should appear at the top of the list.
2. Show those employees who have a last name starting with the letters “K,” “L,” or “M.”



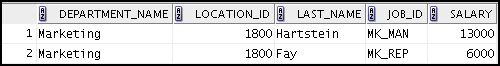
Additional Practices and Solutions

Chapter 12 - Page 4

1. Create a report that displays all employees, and indicate whether they receive a commission by using the words *Yes* or *No*. Use the DECODE expression in your query.



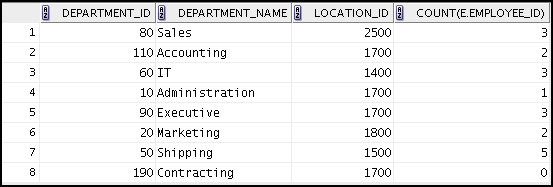
These exercises can be used for extra practice after you have discussed the following topics: basic SQL SELECT statements, basic SQL Developer commands, SQL functions, joins, and group functions.

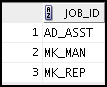
1. Create a report that displays the department name, location ID, last name, job title, and salary of those employees who work in a specific location. Prompt the user for a location. For example, if the user enters 1800, results are as follows:
2. Find the number of employees who have a last name that ends with the letter “n.” Create two possible solutions.

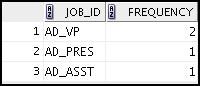


Additional Practices and Solutions

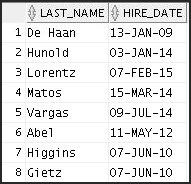
Chapter 12 - Page 5

1. Create a report that shows the name, location, and number of employees for each department. Make sure that the report also includes department\_IDs without employees.
2. The HR department needs to find the job titles in departments 10 and 20. Create a report to display the job IDs for those departments.



1. Create a report that displays the jobs that are found in the Administration and Executive departments. Also display the number of employees for these jobs. Show the job with the highest number of employees first.

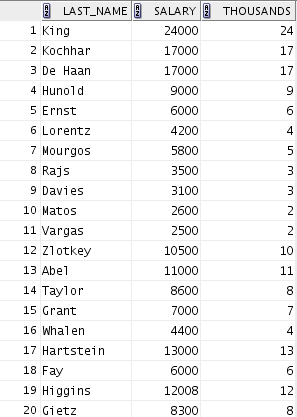
These exercises can be used for extra practice after you have discussed the following topics: basic SQL SELECT statements, basic SQL Developer commands, SQL functions, joins, group functions, and subqueries.

1. Show all the employees who were hired in the first half of the month (before the 16th of the month, irrespective of the year).

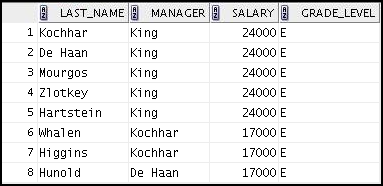
Additional Practices and Solutions

Chapter 12 - Page 6

1. Create a report that displays the following for all employees: last name, salary, and salary expressed in terms of thousands of dollars.



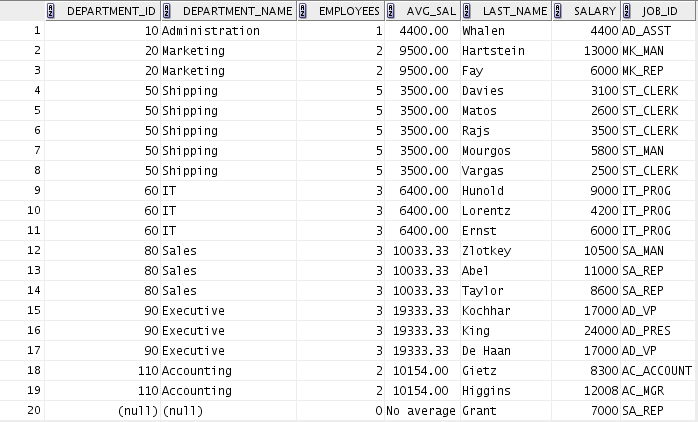
1. Show all the employees who have managers with a salary higher than $15,000. Show the following data: employee name, manager name, manager salary, and salary grade of the manager.



Additional Practices and Solutions

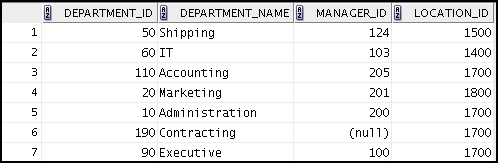
Chapter 12 - Page 7

1. Show the department number, department name, number of employees, and average salary of all the departments, together with the employee names, salaries, and jobs of the employees working in each department.



1. Create a report to display the department number and lowest salary of the department with the highest average salary.

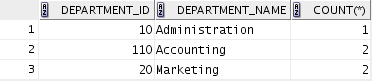


1. Create a report that displays departments where no sales representatives work. Include the department number, department name, manager ID, and location in the output.

Additional Practices and Solutions

Chapter 12 - Page 8

1. Create the following statistical reports for the HR department. Include the department number, department name, and the number of employees working in each department that:
   1. Employs fewer than three employees:



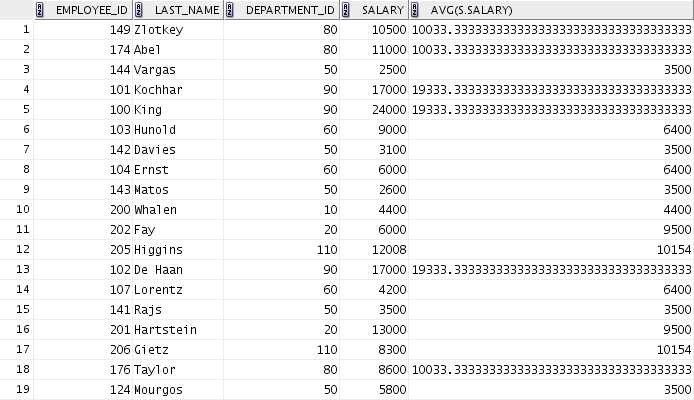
* 1. Has the highest number of employees:



* 1. Has the lowest number of employees:

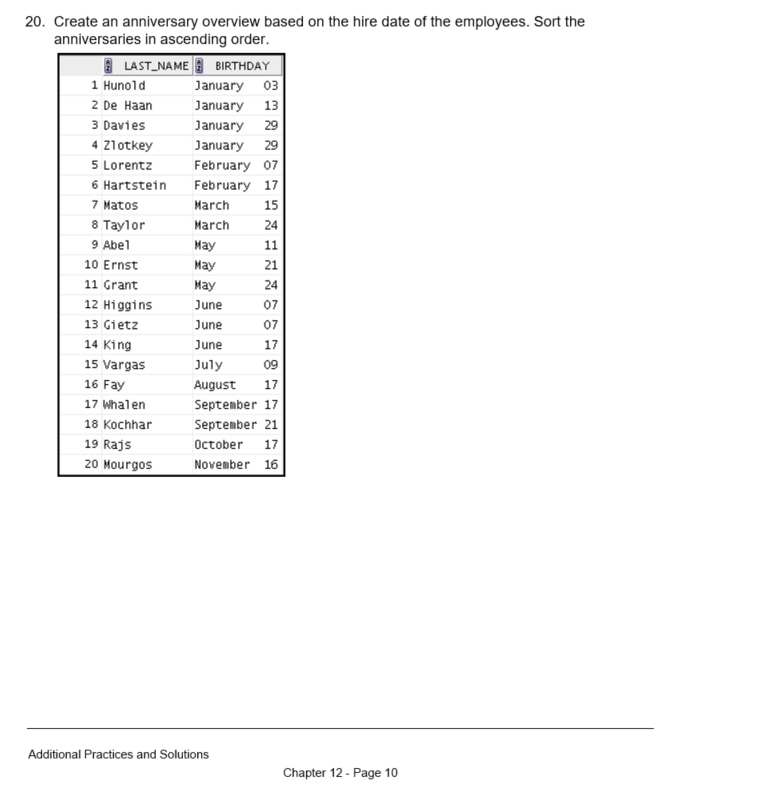


1. Create a report that displays the employee number, last name, salary, department number, and the average salary in their department for all employees.



Additional Practices and Solutions

Chapter 12 - Page 9



# Solution 1-1: Additional Practice

### Overview

Solutions to Additional Practice 1-1 are given as follows.

### Tasks

1. The HR department needs to find data for all the clerks who were hired after 2010.

SELECT \*

FROM employees

WHERE job\_id = 'ST\_CLERK' AND hire\_date > '31-DEC-2010';

1. The HR department needs a report of employees who earn a commission. Show the last name, job, salary, and commission of these employees. Sort the data by salary in descending order.

SELECT last\_name, job\_id, salary, commission\_pct FROM employees

WHERE commission\_pct IS NOT NULL

ORDER BY salary DESC;

1. For budgeting purposes, the HR department needs a report on projected raises. The report should display those employees who do not get a commission but who have a 10% raise in salary (round off the salaries).

SELECT 'The salary of '||last\_name||' after a 10% raise is '

|| ROUND(salary\*1.10) "New salary" FROM employees

WHERE commission\_pct IS NULL;

1. Create a report of employees and the duration of their employment. Show the last names of all employees, together with the number of years and the number of completed months that they have been employed. Order the report by the duration of their employment. The employee who has been employed the longest should appear at the top of the list.

SELECT last\_name,

TRUNC(MONTHS\_BETWEEN(SYSDATE, hire\_date) / 12) YEARS,

TRUNC(MOD(MONTHS\_BETWEEN(SYSDATE, hire\_date), 12)) MONTHS

FROM employees

ORDER BY years DESC, MONTHS desc;

1. Show those employees who have a last name that starts with the letters “K,” “L,” or “M.”

SELECT last\_name FROM employees

WHERE SUBSTR(last\_name, 1,1) IN ('K', 'L', 'M');

Additional Practices and Solutions

Chapter 12 - Page 11

1. Create a report that displays all employees, and indicate whether they receive a commission by using the words *Yes* or *No*. Use the DECODE expression in your query.

SELECT last\_name, salary,

decode(commission\_pct, NULL, 'No', 'Yes') commission FROM employees;

These exercises can be used for extra practice after you have discussed the following topics: basic SQL SELECT statement, basic SQL Developer commands, SQL functions, joins, and group functions.

1. Create a report that displays the department name, location ID, last name, job title, and salary of those employees who work in a specific location. Prompt the user for a location.

Enter 1800 for location\_id when prompted.

SELECT d.department\_name, d.location\_id, e.last\_name, e.job\_id, e.salary

FROM employees e JOIN departments d

ON e.department\_id = d.department\_id AND d.location\_id = &location\_id;

1. Find the number of employees who have a last name that ends with the letter “n.” Create two possible solutions.

SELECT COUNT(\*)

FROM employees

WHERE last\_name LIKE '%n';

--or

SELECT COUNT(\*)

FROM employees

WHERE SUBSTR(last\_name, -1) = 'n';

1. Create a report that shows the name, location, and number of employees for each department. Make sure that the report also includes department\_IDs without employees.

SELECT d.department\_id, d.department\_name,

d.location\_id, COUNT(e.employee\_id)

FROM

ON

employees e RIGHT OUTER JOIN

departments d

e.department\_id = d.department\_id

GROUP BY d.department\_id, d.department\_name, d.location\_id;

1. The HR department needs to find the job titles in departments 10 and 20. Create a report to display the job IDs for these departments.

SELECT DISTINCT job\_id FROM employees

WHERE department\_id IN (10, 20);

Additional Practices and Solutions

Chapter 12 - Page 12

1. Create a report that displays the jobs that are found in the Administration and Executive departments. Also display the number of employees for these jobs. Show the job with the highest number of employees first.

SELECT e.job\_id, count(e.job\_id) FREQUENCY FROM employees e JOIN departments d

ON e.department\_id = d.department\_id

WHERE d.department\_name IN ('Administration', 'Executive') GROUP BY e.job\_id

ORDER BY FREQUENCY DESC;

These exercises can be used for extra practice after you have discussed the following topics: basic SQL SELECT statements, basic SQL Developer commands, SQL functions, joins, group functions, and subqueries.

1. Show all employees who were hired in the first half of the month (before the 16th of the month, irrespective of the year).

SELECT last\_name, hire\_date FROM employees

WHERE TO\_CHAR(hire\_date, 'DD') < 16;

1. Create a report that displays the following for all employees: last name, salary, and salary expressed in terms of thousands of dollars.

SELECT last\_name, salary, TRUNC(salary, -3)/1000 Thousands

FROM employees;

1. Show all employees who have managers with a salary higher than $15,000. Show the following data: employee name, manager name, manager salary, and salary grade of the manager.

SELECT e.last\_name, m.last\_name manager, m.salary, j.grade\_level

FROM employees e JOIN employees m ON e.manager\_id = m.employee\_id JOIN job\_grades j

ON m.salary BETWEEN j.lowest\_sal AND j.highest\_sal

AND m.salary > 15000;

Additional Practices and Solutions

Chapter 12 - Page 13

1. Show the department number, department name, number of employees, and average salary of all departments, together with the employee names, salaries, and jobs of the employees working in each department.

SELECT d.department\_id, d.department\_name, count(e1.employee\_id) employees, NVL(TO\_CHAR(AVG(e1.salary), '99999.99'), 'No average' )

avg\_sal,

e2.last\_name, e2.salary, e2.job\_id

FROM departments d RIGHT OUTER JOIN employees e1 ON d.department\_id = e1.department\_id

RIGHT OUTER JOIN employees e2

ON d.department\_id = e2.department\_id

GROUP BY d.department\_id, d.department\_name, e2.last\_name, e2.salary,

e2.job\_id

ORDER BY d.department\_id, employees;

1. Create a report to display the department number and lowest salary of the department with the highest average salary.

SELECT department\_id, MIN(salary) FROM employees

GROUP BY department\_id

HAVING AVG(salary) = (SELECT MAX(AVG(salary))

FROM employees

GROUP BY department\_id);

1. Create a report that displays the departments where no sales representatives work. Include the department number, department name, manager ID, and location in the output.

SELECT \*

FROM departments

WHERE department\_id NOT IN(SELECT department\_id

FROM employees

WHERE job\_id = 'SA\_REP'

AND department\_id IS NOT NULL);

1. Create the following statistical reports for the HR department. Include the department number, department name, and the number of employees working in each department that:
   1. Employs fewer than three employees:

SELECT d.department\_id, d.department\_name, COUNT(\*) FROM departments d JOIN employees e

ON d.department\_id = e.department\_id GROUP BY d.department\_id, d.department\_name

HAVING COUNT(\*) < 3;

Additional Practices and Solutions

Chapter 12 - Page 14

* 1. Has the highest number of employees:

SELECT d.department\_id, d.department\_name, COUNT(\*) FROM departments d JOIN employees e

ON d.department\_id = e.department\_id GROUP BY d.department\_id, d.department\_name HAVING COUNT(\*) = (SELECT MAX(COUNT(\*))

FROM employees

GROUP BY department\_id);

* 1. Has the lowest number of employees:

SELECT d.department\_id, d.department\_name, COUNT(\*) FROM departments d JOIN employees e

ON d.department\_id = e.department\_id GROUP BY d.department\_id, d.department\_name HAVING COUNT(\*) = (SELECT MIN(COUNT(\*))

FROM employees

GROUP BY department\_id);

1. Create a report that displays the employee number, last name, salary, department number, and the average salary in their department for all employees.

SELECT e.employee\_id, e.last\_name, e.department\_id, e.salary, AVG(s.salary)

FROM employees e JOIN employees s

ON e.department\_id = s.department\_id

GROUP BY e.employee\_id, e.last\_name, e.department\_id, e.salary;

1. Create an anniversary overview based on the hire date of employees. Sort the anniversaries in ascending order.

SELECT last\_name, TO\_CHAR(hire\_date, 'Month DD') BIRTHDAY FROM employees

ORDER BY TO\_CHAR(hire\_date, 'DDD');

Additional Practices and Solutions

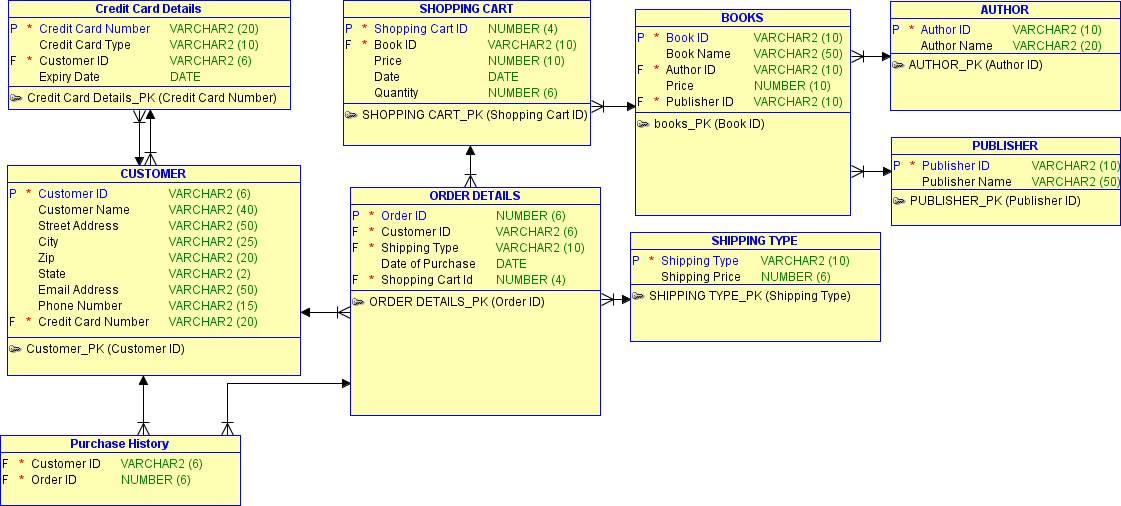
Chapter 12 - Page 15

# Case Study: Online Book Store

### Overview

In this case study, you build a set of database tables for an online book store (E-Commerce Shopping Cart). After you create the tables, you insert, update, and delete records in the book store database and generate a report. The database contains only the essential tables.

The following is a diagram of the table and columns for the online book store application:



**Note:** If you want to build the tables, you can execute the commands in the Online\_Book\_Store\_Create\_Table.sql script in SQL Developer. If you want to drop the tables, you can execute the commands in the Online\_Book\_Store\_Drop\_Tables.sql script in SQL Developer. Then you can execute the commands in the

<<Online\_Book\_Store\_Populate.sql>> script in SQL Developer to create and populate the tables.

All the three SQL scripts are present in the Desktop/oracle19c-labs/sql-intro/labs/sql1/labs folder.

* If you use the Online\_Book\_Store\_Create\_Table.sql script to build the tables, start with step 2.
* If you use the Online\_Book\_Store\_Drop\_Tables.sql script to remove the tables, start with step 1.
* If you use the Online\_Book\_Store\_Populate.sql script to build and populate the tables, start with step 6.

Additional Practices and Solutions

Chapter 12 - Page 16

# Practice 1-2

### Overview

In this practice, you create the tables based on the following table instance charts. Select the appropriate data types and be sure to add integrity constraints.

#### Tasks

1. **Table Details**
   1. Table Name: AUTHOR

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Data type** | **Key** | **Table Dependent Type** |
| Author\_ID | VARCHAR2 | PK |  |
| Author\_Name | VARCHAR2 |  |  |

* 1. Table Name: BOOKS

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Datatype** | **Key** | **Table Dependent On** |
| Book\_ID | VARCHAR2 | PK |  |
| Book\_Name | VARCHAR2 |  |  |
| Author\_ID | VARCHAR2 | FK | AUTHORS |
| Price | NUMBER |  |  |
| Publisher\_ID | VARCHAR2 | FK | PUBLISHER |

* 1. Table Name: CUSTOMER

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data type** | **Key** | **Table Dependent On** |
| Customer\_ID | VARCHAR2 | PK |  |
| Customer\_Name | VARCHAR2 |  |  |
| Street\_Address | VARCHAR2 |  |  |
| City | VARCHAR2 |  |  |
| Phone\_Number | VARCHAR2 |  |  |
| Credit\_Card\_Number | VARCHAR2 | FK | Credit\_Card\_Details |

* 1. CREDIT\_CARD\_DETAILS

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data type** | **Key** | **Table Dependent On** |
| Credit\_Card\_Number | VARCHAR2 | PK |  |
| Credit\_Card\_Type | VARCHAR2 |  |  |
| Expiry\_Date | DATE |  |  |

* 1. Table Name: ORDER\_DETAILS

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Data type** | **Key** | **Table Dependent On** |
| Order\_ID | NUMBER | PK |  |
| Customer\_ID | VARCHAR2 | FK | CUSTOMER |
| Shipping\_Type | VARCHAR2 | FK | SHIPPING\_TYPE |
| Date\_of\_Purchase | DATE |  |  |
| Shopping\_Cart\_ID | NUMBER | FK | SHOPPING\_CART |
|  |  |  |  |

Additional Practices and Solutions

Chapter 12 - Page 17

* 1. Table Name: PUBLISHER

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Data type** | **Key** | **Table Dependent Type** |
| Publisher\_ID | VARCHAR2 | PK |  |
| Publisher\_Name | VARCHAR2 |  |  |

* 1. Table Name: PURCHASE\_HISTORY

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Data type** | **Key** | **Table Dependent Type** |
| Customer\_ID | VARCHAR2 | FK | CUSTOMER |
| Order\_ID | NUMBER | FK | ORDER\_DETAILS |

* 1. Table Name: SHIPPING\_TYPE

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Data type** | **Key** | **Table Dependent Type** |
| Shipping\_Type | VARCHAR2 | PK |  |
| Shipping\_Price | NUMBER |  |  |

* 1. Table Name: SHOPPING\_CART

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Data type** | **Key** | **Table Dependent On** |
| Shopping\_Cart\_ID | NUMBER | PK |  |
| Book\_ID | VARCHAR2 | FK | BOOKS |
| Price | NUMBER |  |  |
| Shopping\_Cart\_Date | DATE |  |  |
| Quantity | NUMBER |  |  |

1. Add additional Referential Integrity constraints to the tables created.
2. Verify that the tables were created properly by checking in the Connections Navigator in SQL Developer.

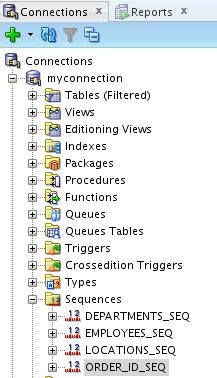
Additional Practices and Solutions

Chapter 12 - Page 18

1. Create a sequence to uniquely identify each row in the ORDER\_DETAILS table.
   1. Start with 100; do not allow caching of the values. Name the sequence

ORDER\_ID\_SEQ.

* 1. Verify the existence of the sequences in the Connections Navigator in SQL Developer.

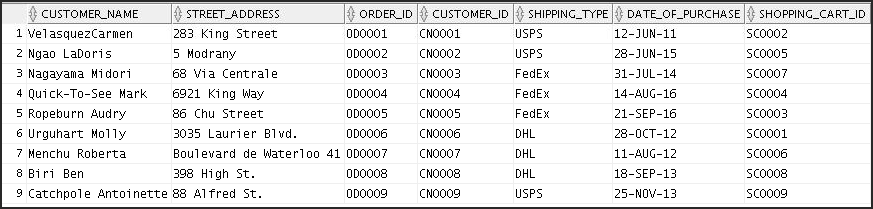


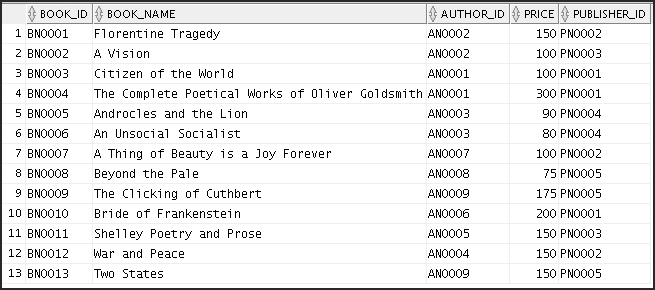
1. Add data to the tables. Create a script for each set of data to be added. Add data to the following tables:
   1. AUTHOR
   2. PUBLISHER
   3. SHIPPING\_TYPE
   4. CUSTOMER
   5. CREDIT\_CARD\_DETAILS
   6. BOOKS
   7. SHOPPING\_CART
   8. ORDER\_DETAILS
   9. PURCHASE\_HISTORY

**Note:** Save the scripts by using the task number. For example, to save the script created for the AUTHOR table, you can save it as labs\_apcs\_5a\_1.sql. Ensure that you save the scripts in /home/oracle/labs/sql1/labs folder.

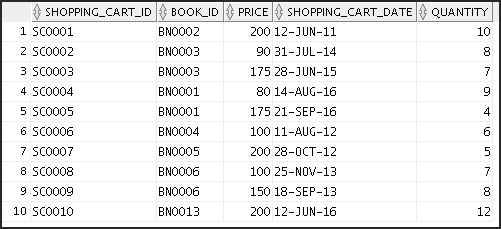
Additional Practices and Solutions

Chapter 12 - Page 19

1. Create a view named CUSTOMER\_DETAILS to show the Customer Name, Customer Address, and the details of the order placed by the customer. Order the results by Customer ID.
2. Make changes to the data in the tables.
   1. Add a new book detail. Verify if the author detail for the book is available in the AUTHOR

table. If not, make an entry in the AUTHOR table.

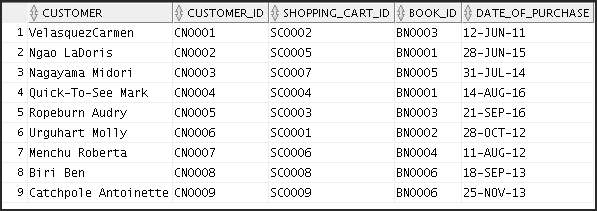
* 1. Enter a shopping cart detail for the book details that you just entered in 7(a).



Additional Practices and Solutions

Chapter 12 - Page 20

1. Create a report that contains each customer’s history of purchasing books. Be sure to include the customer name, customer ID, book ID, date of purchase, and shopping cart ID. Save the commands that generate the report in a script file named lab\_apcs\_8.sql.

**Note:** Your results may be different.

Additional Practices and Solutions

Chapter 12 - Page 21

# Solution 1-2

### Overview

The solution to Practice 1-2 is given as follows.

### Tasks

#### Table Details

* 1. AUTHOR

CREATE TABLE AUTHOR (

Author\_ID VARCHAR2 (10) NOT NULL ,

Author\_Name VARCHAR2 (20)

)

;

COMMENT ON TABLE AUTHOR IS 'Author'

;

ALTER TABLE AUTHOR

ADD CONSTRAINT AUTHOR\_PK PRIMARY KEY (Author\_ID);

* 1. BOOKS

CREATE TABLE BOOKS (

Book\_ID VARCHAR2 (10) NOT NULL ,

Book\_Name VARCHAR2 (50) ,

Author\_ID VARCHAR2 (10) NOT NULL , Price NUMBER (10) ,

Publisher\_ID VARCHAR2 (10) NOT NULL

)

;

COMMENT ON TABLE BOOKS IS 'Books'

;

ALTER TABLE BOOKS

ADD CONSTRAINT books\_PK PRIMARY KEY ( Book\_ID );

Additional Practices and Solutions

Chapter 12 - Page 22

* 1. CUSTOMER

CREATE TABLE CUSTOMER (

Customer\_ID VARCHAR2 (6) NOT NULL ,

Customer\_Name VARCHAR2 (40) , Street\_Address VARCHAR2 (50) , City VARCHAR2 (25) ,

Phone\_Number VARCHAR2 (15) , Credit\_Card\_Number VARCHAR2 (20) NOT NULL

)

;

COMMENT ON TABLE CUSTOMER IS 'Customer'

;

ALTER TABLE CUSTOMER

ADD CONSTRAINT Customer\_PK PRIMARY KEY ( Customer\_ID ) ;

* 1. CREDIT\_CARD\_DETAILS

CREATE TABLE CREDIT\_CARD\_DETAILS (

Credit\_Card\_Number VARCHAR2 (20) NOT NULL , Credit\_Card\_Type VARCHAR2 (15) , Expiry\_Date DATE

)

;

COMMENT ON TABLE CREDIT\_CARD\_DETAILS IS 'Credit Card Details'

;

ALTER TABLE CREDIT\_CARD\_DETAILS

ADD CONSTRAINT Credit\_Card\_Details\_PK PRIMARY KEY ( Credit\_Card\_Number) ;

Additional Practices and Solutions

Chapter 12 - Page 23

* 1. ORDER\_DETAILS

CREATE TABLE ORDER\_DETAILS (

Order\_ID VARCHAR2 (6) NOT NULL , Customer\_ID VARCHAR2 (6) NOT NULL ,

Shipping\_Type VARCHAR2 (10) NOT NULL , Date\_of\_Purchase DATE ,

Shopping\_Cart\_ID varchar2(6) NOT NULL

)

;

COMMENT ON TABLE ORDER\_DETAILS IS 'Order Details'

;

ALTER TABLE ORDER\_DETAILS

ADD CONSTRAINT ORDER\_DETAILS\_PK PRIMARY KEY (Order\_ID ) ;

* 1. PUBLISHER

CREATE TABLE PUBLISHER (

Publisher\_ID VARCHAR2 (10) NOT NULL ,

Publisher\_Name VARCHAR2 (50)

)

;

COMMENT ON TABLE PUBLISHER IS 'Publisher'

;

ALTER TABLE PUBLISHER

ADD CONSTRAINT PUBLISHER\_PK PRIMARY KEY ( Publisher\_ID) ;

* 1. PURCHASE\_HISTORY

CREATE TABLE PURCHASE\_HISTORY (

Customer\_ID VARCHAR2 (6) NOT NULL , Order\_ID VARCHAR2 (6) NOT NULL

)

;

Additional Practices and Solutions

Chapter 12 - Page 24

COMMENT ON TABLE PURCHASE\_HISTORY IS 'Purchase History'

;

* 1. SHIPPING\_TYPE

CREATE TABLE SHIPPING\_TYPE (

Shipping\_Type VARCHAR2 (10) NOT NULL , Shipping\_Price NUMBER (6)

)

;

COMMENT ON TABLE SHIPPING\_TYPE IS 'Shipping Type'

;

ALTER TABLE SHIPPING\_TYPE

ADD CONSTRAINT SHIPPING\_TYPE\_PK PRIMARY KEY ( Shipping\_Type

) ;

* 1. SHOPPING \_CART

CREATE TABLE SHOPPING\_CART (

Shopping\_Cart\_ID VARCHAR2 (6) NOT NULL , Book\_ID VARCHAR2 (10) NOT NULL ,

Price NUMBER (10) ,

Shopping\_Cart\_Date DATE , Quantity NUMBER (6)

)

;

COMMENT ON TABLE SHOPPING\_CART IS 'Shopping Cart'

;

ALTER TABLE SHOPPING\_CART

ADD CONSTRAINT SHOPPING\_CART\_PK PRIMARY KEY (SHOPPING\_CART\_ID)

;

Additional Practices and Solutions

Chapter 12 - Page 25

#### Adding Additional Referential Integrity Constraints to the Table Created

* 1. Include a Foreign Key constraint in the CUSTOMER table.

ALTER TABLE CUSTOMER

ADD CONSTRAINT CREDIT\_CARD\_FK FOREIGN KEY (

Credit\_Card\_Number

)

REFERENCES CREDIT\_CARD\_DETAILS (

Credit\_Card\_Number

)

;

* 1. Include a Foreign Key constraint in the BOOKS table.

ALTER TABLE BOOKS

ADD CONSTRAINT BOOKS\_AUTHOR\_FK FOREIGN KEY (

Author\_ID

)

REFERENCES AUTHOR (

Author\_ID

)

;

ALTER TABLE BOOKS

ADD CONSTRAINT BOOKS\_PUBLISHER\_FK FOREIGN KEY (

Publisher\_ID

)

REFERENCES PUBLISHER (

Publisher\_ID

);

Additional Practices and Solutions

Chapter 12 - Page 26

* 1. Include a Foreign Key constraint in the ORDER\_DETAILS table.

ALTER TABLE ORDER\_DETAILS

ADD CONSTRAINT Order\_ID\_FK FOREIGN KEY (

Customer\_ID

)

REFERENCES CUSTOMER (

Customer\_ID

)

;

ALTER TABLE ORDER\_DETAILS

ADD CONSTRAINT FK\_Order\_details FOREIGN KEY (

Shipping\_Type

)

REFERENCES SHIPPING\_TYPE (

Shipping\_Type

)

;

ALTER TABLE ORDER\_DETAILS

ADD CONSTRAINT Order\_Details\_fk FOREIGN KEY (

Shopping\_Cart\_ID

)

REFERENCES SHOPPING\_CART (

Shopping\_Cart\_ID

)

;

* 1. Include a Foreign Key constraint in the PURCHASE\_HISTORY table.

ALTER TABLE PURCHASE\_HISTORY

ADD CONSTRAINT Pur\_Hist\_ORDER\_DETAILS\_FK FOREIGN KEY (

Order\_ID

Additional Practices and Solutions

Chapter 12 - Page 27

)

REFERENCES ORDER\_DETAILS (

Order\_ID

)

;

ALTER TABLE PURCHASE\_HISTORY

ADD CONSTRAINT Purchase\_History\_CUSTOMER\_FK FOREIGN KEY (

Customer\_ID

)

REFERENCES CUSTOMER (

Customer\_ID

) ;

* 1. Include a Foreign Key constraint in the SHOPPING\_CART table.

ALTER TABLE SHOPPING\_CART

ADD CONSTRAINT SHOPPING\_CART\_BOOKS\_FK FOREIGN KEY (

Book\_ID

)

REFERENCES BOOKS (

Book\_ID

)

;

1. Verify that the tables were created properly by checking in the Connections Navigator in SQL Developer. In the Connections Navigator, expand Connections > myconnection > Tables.
2. Create a sequence to uniquely identify each row in the ORDER\_DETAILS table.
   1. Start with 100; do not allow caching of the values. Name the sequence ORDER\_ID\_SEQ.

CREATE SEQUENCE order\_id\_seq START WITH 100

NOCACHE;

Additional Practices and Solutions

Chapter 12 - Page 28

* 1. Verify the existence of the sequences in the Connections Navigator in SQL Developer.

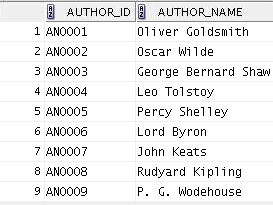
In the Connections Navigator, assuming that the myconnection node is expanded, expand Sequences.

Alternatively, you can also query the user\_sequences data dictionary view:

SELECT \* FROM user\_sequences;

1. Add data to the tables.
   1. AUTHOR Table

|  |  |
| --- | --- |
| Author\_ID | Author\_Name |
| AN0001 | Oliver Goldsmith |
| AN0002 | Oscar Wilde |
| AN0003 | George Bernard Shaw |
| AN0004 | Leo Tolstoy |
| AN0005 | Percy Shelley |
| AN0006 | Lord Byron |
| AN0007 | John Keats |
| AN0008 | Rudyard Kipling |
| AN0009 | P. G. Wodehouse |

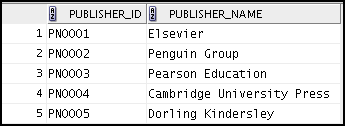


* 1. PUBLISHER Table

|  |  |
| --- | --- |
| Publisher\_ID | Publisher\_Name |
| PN0001 | Elsevier |
| PN0002 | Penguin Group |
| PN0003 | Pearson Education |
| PN0004 | Cambridge University Press |
| PN0005 | Dorling Kindersley |

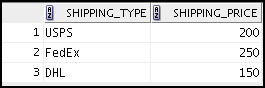
Additional Practices and Solutions

Chapter 12 - Page 29



* 1. SHIPPING \_TYPE

|  |  |
| --- | --- |
| Shipping\_Type | Shipping\_Price |
| USPS | 200 |
| FedEx | 250 |
| DHL | 150 |

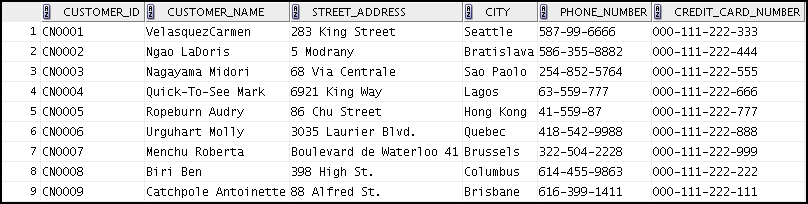


* 1. CUSTOMER

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Customer  \_ ID | Customer \_Name | Street  \_Address | City | Phone  \_number | Credit \_Card  \_Number |
| CN0001 | VelasquezCarmen | 283 King Street | Seattle | 587-99-6666 | 000-111-222-333 |
| CN0002 | Ngao LaDoris | 5 Modrany | Bratislav a | 586-355-8882 | 000-111-222-444 |
| CN0003 | Nagayama Midori | 68 Via Centrale | Sao Paolo | 254-852-5764 | 000-111-222-555 |
| CN0004 | Quick-To-See Mark | 6921 King Way | Lagos | 63-559-777 | 000-111-222-666 |
| CN0005 | Ropeburn Audry | 86 Chu Street | Hong Kong | 41-559-87 | 000-111-222-777 |
| CN0006 | Urguhart Molly | 3035 Laurier Blvd. | Quebec | 418-542-9988 | 000-111-222-888 |
| CN0007 | Menchu Roberta | Boulevard de Waterloo 41 | Brussels | 322-504-2228 | 000-111-222-999 |
| CN0008 | Biri Ben | 398 High St. | Columbu s | 614-455-9863 | 000-111-222-222 |
| CN0009 | Catchpole Antoinette | 88 Alfred St. | Brisbane | 616-399-1411 | 000-111-222-111 |

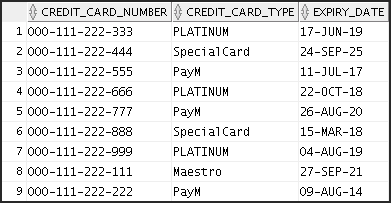
Additional Practices and Solutions

Chapter 12 - Page 30



* 1. CREDIT\_CARD\_DETAILS

|  |  |  |
| --- | --- | --- |
| Credit \_Card\_ Number | Credit \_Card \_Type | Expiry \_Date |
| 000-111-222-333 | Platinum | 17-JUN-2019 |
| 000-111-222-444 | SpecialCard | 24-SEP-2025 |
| 000-111-222-555 | PayM | 11-JUL-2017 |
| 000-111-222-666 | Platinum | 22-OCT-2018 |
| 000-111-222-777 | PayM | 26-AUG-2020 |
| 000-111-222-888 | SpecialCard | 15-MAR-2018 |
| 000-111-222-999 | Platinum | 4-AUG-2019 |
| 000-111-222-111 | Maestro | 27-SEP-2021 |
| 000-111-222-222 | PayM | 9-AUG-2024 |



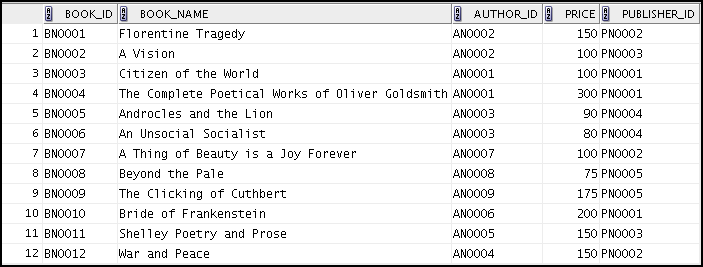
* 1. BOOKS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Book \_ID | Book \_Name | Author \_ID | Price | Publisher \_ID |
| BN0001 | Florentine Tragedy | AN0002 | 150 | PN0002 |
| BN0002 | A Vision | AN0002 | 100 | PN0003 |
| BN0003 | Citizen of the World | AN0001 | 100 | PN0001 |

Additional Practices and Solutions

Chapter 12 - Page 31

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| BN0004 | The Complete Poetical  Works of Oliver Goldsmith | AN0001 | 300 | PN0001 |
| BN0005 | Androcles and the Lion | AN0003 | 90 | PN0004 |
| BN0006 | An Unsocial Socialist | AN0003 | 80 | PN0004 |
| BN0007 | A Thing of Beauty is a Joy Forever | AN0007 | 100 | PN0002 |
| BN0008 | Beyond the Pale | AN0008 | 75 | PN0005 |
| BN0009 | The Clicking of Cuthbert | AN0009 | 175 | PN0005 |
| BN00010 | Bride of Frankenstein | AN0006 | 200 | PN0001 |
| BN00011 | Shelley's Poetry and Prose | AN0005 | 150 | PN0003 |
| BN00012 | War and Peace | AN0004 | 150 | PN0002 |

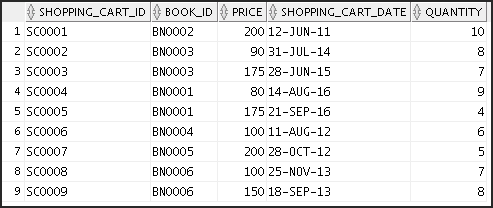
* 1. SHOPPING\_CART

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Shopping \_Cart  \_ID | Book \_ID | Price | Shopping \_Cart  \_Date | Quantity |
| SC0001 | BN0002 | 200 | 12-JUN-2011 | 10 |

Additional Practices and Solutions

Chapter 12 - Page 32

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SC0002 | BN0003 | 90 | 31-JUL-2014 | 8 |
| SC0003 | BN0003 | 175 | 28-JUN-2015 | 7 |
| SC0004 | BN0001 | 80 | 14-AUG-2016 | 9 |
| SC0005 | BN0001 | 175 | 21-SEP-2016 | 4 |
| SC0006 | BN0004 | 100 | 11-AUG-2012 | 6 |
| SC0007 | BN0005 | 200 | 28-OCT-2012 | 5 |
| SC0008 | BN0006 | 100 | 25-NOV-2013 | 7 |
| SC0009 | BN0006 | 150 | 18-SPET-2013 | 8 |

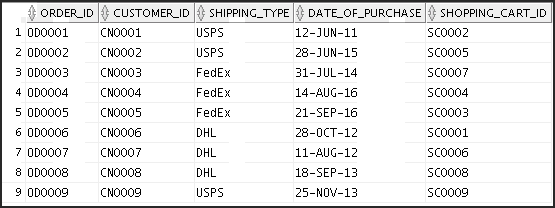


* 1. ORDER \_DETAILS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Order \_ID | Customer  \_ID | Shipping\_ Type | Date \_of \_Purchase | Shopping \_Cart \_ID |
| OD0001 | CN0001 | USPS | 12-JUN-2011 | SC0002 |
| OD0002 | CN0002 | USPS | 28-JUN-2015 | SC0005 |
| OD0003 | CN0003 | FedEx | 31-JUL-2014 | SC0007 |
| OD0004 | CN0004 | FedEx | 14-AUG-2016 | SC0004 |
| OD0005 | CN0005 | FedEx | 21-SEP-2016 | SC0003 |
| OD0006 | CN0006 | DHL | 28-OCT-2012 | SC0001 |
| OD0007 | CN0007 | DHL | 11-AUG-2012 | SC0006 |
| OD0008 | CN0008 | DHL | 18-SEP-2013 | SC0008 |
| OD0009 | CN0009 | USPS | 25-NOV-2013 | SC0009 |

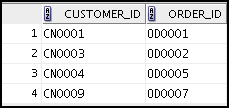
Additional Practices and Solutions

Chapter 12 - Page 33



* 1. PURCHASE\_HISTORY

|  |  |
| --- | --- |
| Customer \_ID | Order \_ID |
| CN0001 | OD0001 |
| CN0003 | OD0002 |
| CN0004 | OD0005 |
| CN0009 | OD0007 |



1. Create a view named CUSTOMER\_DETAILS to show the Customer Name, Customer Address, and details of the order placed by the customer. Order the results by Customer ID.

CREATE VIEW customer\_details AS

SELECT c.customer\_name, c.street\_address, o.order\_id, o.customer\_id, o.shipping\_type, o.date\_of\_purchase, o.shopping\_cart\_id

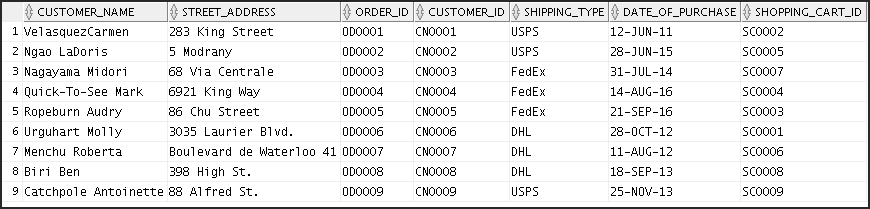
FROM customer c JOIN order\_details o ON c.customer\_id = o.customer\_id;

SELECT \*

FROM customer\_details ORDER BY customer\_id;

Additional Practices and Solutions

Chapter 12 - Page 34



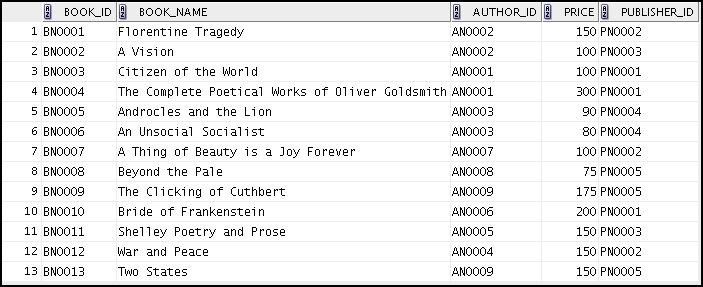
1. Make changes to the data in the tables.
   1. Add a new book detail. Verify if the author detail for the book is available in the AUTHOR

table. If not, make an entry in the AUTHOR table.

INSERT INTO books(book\_id, book\_name, author\_id, price, publisher\_id)

VALUES ('BN0013','Two States','AN0009','150','PN0005');

SELECT \* FROM books;



* 1. Enter a shopping cart detail for the book details that you just entered in 7(a).

INSERT INTO shopping\_cart(shopping\_cart\_id, book\_id, price, Shopping\_cart\_date,quantity)

VALUES ('SC0010','BN0013','200',TO\_DATE('12-JUN-2016','DD-MON- YYYY'),'12');

SELECT \* FROM shopping\_cart;

Additional Practices and Solutions

Chapter 12 - Page 35

1. Create a report that contains each customer’s history of purchasing books. Be sure to include the customer name, customer ID, book ID, date of purchase, and shopping cart ID. Save the commands that generate the report in a script file named lab\_apcs\_8.sql.

**Note:** Your results may be different.

SELECT c.customer\_name CUSTOMER, c.customer\_id, s.shopping\_cart\_id, s.book\_id,o.date\_of\_purchase

FROM customer c JOIN order\_details o

ON o.customer\_id=c.customer\_id JOIN shopping\_cart s

ON o.shopping\_cart\_id=s.shopping\_cart\_id;

Additional Practices and Solutions

Chapter 12 - Page 36