

# **Oracle Database 10g: SQL Fundamentals I**

**Student Guide • Volume 3**

D17108GC30

Edition 3.0

January 2009

D57872

**ORACLE®**

**Authors**

Salome Clement  
Chaitanya Koratamaddi  
Nancy Greenberg

**Technical Contributors  
and Reviewers**

Wayne Abbott  
Christian Bauwens  
Claire Bennett  
Perry Benson  
Brian Boxx  
Zarko Cesljas  
Dairy Chan  
Laszlo Czinkoczki  
Joel Goodman  
Matthew Gregory  
Sushma Jagannath  
Yash Jain  
Angelika Krupp  
Isabelle Marchand  
Malika Marghadi  
Valli Pataballa  
Narayanan Radhakrishnan  
Bryan Roberts  
Helen Robertson  
Lata Shivaprasad  
John Soltani  
James Spiller  
Priya Vennapusa

**Editors**

Arijit Ghosh  
Raj Kumar

**Graphic Designer**

Rajiv Chandrabhanu

**Publisher**

Giri Venugopal

Copyright © 2009, Oracle. All rights reserved.

**Disclaimer**

This document contains proprietary information and is protected by copyright and other intellectual property laws. You may copy and print this document solely for your own use in an Oracle training course. The document may not be modified or altered in any way. Except where your use constitutes "fair use" under copyright law, you may not use, share, download, upload, copy, print, display, perform, reproduce, publish, license, post, transmit, or distribute this document in whole or in part without the express authorization of Oracle.

The information contained in this document is subject to change without notice. If you find any problems in the document, please report them in writing to: Oracle University, 500 Oracle Parkway, Redwood Shores, California 94065 USA. This document is not warranted to be error-free.

**Restricted Rights Notice**

If this documentation is delivered to the United States Government or anyone using the documentation on behalf of the United States Government, the following notice is applicable:

**U.S. GOVERNMENT RIGHTS**

The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose these training materials are restricted by the terms of the applicable Oracle license agreement and/or the applicable U.S. Government contract.

**Trademark Notice**

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

# Contents

## Preface

### I Introduction

Lesson Objectives	I-2
Goals of the Course	I-3
Oracle10g	I-4
Oracle Database 10g	I-6
Oracle Application Server 10g	I-7
Oracle Enterprise Manager 10g Grid Control	I-8
Relational and Object Relational Database Management Systems	I-9
Oracle Internet Platform	I-10
System Development Life Cycle	I-11
Data Storage on Different Media	I-13
Relational Database Concept	I-14
Definition of a Relational Database	I-15
Data Models	I-16
Entity Relationship Model	I-17
Entity Relationship Modeling Conventions	I-19
Relating Multiple Tables	I-21
Relational Database Terminology	I-23
Relational Database Properties	I-25
Communicating with an RDBMS Using SQL	I-26
Oracle's Relational Database Management System	I-27
SQL Statements	I-28
Tables Used in the Course	I-29
Summary	I-30

### 1 Retrieving Data Using the SQL `SELECT` Statement

Objectives	1-2
Capabilities of SQL <code>SELECT</code> Statements	1-3
Basic <code>SELECT</code> Statement	1-4
Selecting All Columns	1-5
Selecting Specific Columns	1-6
Writing SQL Statements	1-7
Column Heading Defaults	1-8

Arithmetic Expressions	1-9
Using Arithmetic Operators	1-10
Operator Precedence	1-11
Defining a Null Value	1-12
Null Values in Arithmetic Expressions	1-13
Defining a Column Alias	1-14
Using Column Aliases	1-15
Concatenation Operator	1-16
Literal Character Strings	1-17
Using Literal Character Strings	1-18
Alternative Quote (q) Operator	1-19
Duplicate Rows	1-20
Development Environments for SQL	1-21
What Is Oracle SQL Developer?	1-22
Oracle SQL Developer Interface	1-23
Creating a Database Connection	1-24
Browsing Database Objects	1-27
Using the SQL Worksheet	1-28
Executing SQL Statements	1-31
Formatting the SQL Code	1-32
Saving SQL Statements	1-33
Running Script Files	1-34
Displaying the Table Structure	1-35
Using the DESCRIBE Command	1-36
Summary	1-37
Practice 1: Overview	1-38

## **2 Restricting and Sorting Data**

Objectives	2-2
Limiting Rows Using a Selection	2-3
Limiting the Rows That Are Selected	2-4
Using the WHERE Clause	2-5
Character Strings and Dates	2-6
Comparison Conditions	2-7
Using Comparison Conditions	2-8
Using the BETWEEN Condition	2-9
Using the IN Condition	2-10
Using the LIKE Condition	2-11
Using the NULL Conditions	2-13
Logical Conditions	2-14

Using the <code>AND</code> Operator	2-15
Using the <code>OR</code> Operator	2-16
Using the <code>NOT</code> Operator	2-17
Rules of Precedence	2-18
Using the <code>ORDER BY</code> Clause	2-20
Sorting	2-21
Substitution Variables	2-22
Using the <code>&amp;</code> Substitution Variable	2-24
Character and Date Values with Substitution Variables	2-26
Specifying Column Names, Expressions, and Text	2-27
Using the <code>&amp;&amp;</code> Substitution Variable	2-28
Using the <code>DEFINE</code> Command	2-29
Using the <code>VERIFY</code> Command	2-30
Summary	2-31
Practice 2: Overview	2-32

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

Objectives	3-2
SQL Functions	3-3
Two Types of SQL Functions	3-4
Single-Row Functions	3-5
Character Functions	3-7
Case-Manipulation Functions	3-9
Using Case-Manipulation Functions	3-10
Character-Manipulation Functions	3-11
Using the Character-Manipulation Functions	3-12
Number Functions	3-13
Using the <code>ROUND</code> Function	3-14
Using the <code>TRUNC</code> Function	3-15
Using the <code>MOD</code> Function	3-16
Working with Dates	3-17
Arithmetic with Dates	3-20
Using Arithmetic Operators with Dates	3-21
Date Functions	3-22
Using Date Functions	3-23
Practice 3: Overview of Part 1	3-25
Conversion Functions	3-26
Implicit Data Type Conversion	3-27
Explicit Data Type Conversion	3-29
Using the <code>TO_CHAR</code> Function with Dates	3-32

Elements of the Date Format Model	3-33
Using the TO_CHAR Function with Dates	3-37
Using the TO_CHAR Function with Numbers	3-38
Using the TO_NUMBER and TO_DATE Functions	3-41
RR Date Format	3-43
RR Date Format: Example	3-44
Nesting Functions	3-45
General Functions	3-47
NVL Function	3-48
Using the NVL Function	3-49
Using the NVL2 Function	3-50
Using the NULLIF Function	3-51
Using the COALESCE Function	3-52
Conditional Expressions	3-54
CASE Expression	3-55
Using the CASE Expression	3-56
DECODE Function	3-57
Using the DECODE Function	3-58
Summary	3-60
Practice 3: Overview of Part 2	3-61

#### **4 Reporting Aggregated Data Using the Group Functions**

Objectives	4-2
What Are Group Functions?	4-3
Types of Group Functions	4-4
Group Functions: Syntax	4-5
Using the AVG and SUM Functions	4-6
Using the MIN and MAX Functions	4-7
Using the COUNT Function	4-8
Using the DISTINCT Keyword	4-9
Group Functions and Null Values	4-10
Creating Groups of Data	4-11
Creating Groups of Data: GROUP BY Clause Syntax	4-12
Using the GROUP BY Clause	4-13
Grouping by More Than One Column	4-15
Using the GROUP BY Clause on Multiple Columns	4-16
Illegal Queries Using Group Functions	4-17
Restricting Group Results	4-19
Restricting Group Results with the HAVING Clause	4-20

Using the `HAVING` Clause 4-21

Nesting Group Functions 4-23

Summary 4-24

Practice 4: Overview 4-25

## **5 Displaying Data from Multiple Tables**

Objectives 5-2

Obtaining Data from Multiple Tables 5-3

Types of Joins 5-4

Joining Tables Using SQL:1999 Syntax 5-5

Creating Natural Joins 5-6

Retrieving Records with Natural Joins 5-7

Creating Joins with the `USING` Clause 5-8

Joining Column Names 5-9

Retrieving Records with the `USING` Clause 5-10

Qualifying Ambiguous Column Names 5-11

Using Table Aliases 5-12

Creating Joins with the `ON` Clause 5-13

Retrieving Records with the `ON` Clause 5-14

Self-Joins Using the `ON` Clause 5-15

Applying Additional Conditions to a Join 5-17

Creating Three-Way Joins with the `ON` Clause 5-18

Nonequijoins 5-19

Retrieving Records with Nonequijoins 5-20

Outer Joins 5-21

`INNER` Versus `OUTER` Joins 5-22

`LEFT OUTER JOIN` 5-23

`RIGHT OUTER JOIN` 5-24

`FULL OUTER JOIN` 5-25

Cartesian Products 5-26

Generating a Cartesian Product 5-27

Creating `Cross Joins` 5-28

Summary 5-29

Practice 5: Overview 5-30

## **6 Using Subqueries to Solve Queries**

Objectives 6-2

Using a Subquery to Solve a Problem 6-3

Subquery Syntax 6-4

Using a Subquery 6-5

Guidelines for Using Subqueries	6-6
Types of Subqueries	6-7
Single-Row Subqueries	6-8
Executing Single-Row Subqueries	6-9
Using Group Functions in a Subquery	6-10
The HAVING Clause with Subqueries	6-11
What Is Wrong with This Statement?	6-12
Will This Statement Return Rows?	6-13
Multiple-Row Subqueries	6-14
Using the ANY Operator in Multiple-Row Subqueries	6-15
Using the ALL Operator in Multiple-Row Subqueries	6-16
Null Values in a Subquery	6-17
Summary	6-19
Practice 6: Overview	6-20

## 7 Using the Set Operators

Objectives	7-2
Set Operators	7-3
Tables Used in This Lesson	7-4
UNION Operator	7-8
Using the UNION Operator	7-9
UNION ALL Operator	7-11
Using the UNION ALL Operator	7-12
INTERSECT Operator	7-13
Using the INTERSECT Operator	7-14
MINUS Operator	7-15
Set Operator Guidelines	7-17
The Oracle Server and Set Operators	7-18
Matching the SELECT Statements	7-19
Matching the SELECT Statement: Example	7-20
Controlling the Order of Rows	7-21
Summary	7-22
Practice 7: Overview	7-23

## 8 Manipulating Data

Objectives	8-2
Data Manipulation Language	8-3
Adding a New Row to a Table	8-4
INSERT Statement Syntax	8-5
Inserting New Rows	8-6



Inserting Rows with Null Values	8-7
Inserting Special Values	8-8
Inserting Specific Date Values	8-9
Creating a Script	8-10
Copying Rows from Another Table	8-11
Changing Data in a Table	8-12
UPDATE Statement Syntax	8-13
Updating Rows in a Table	8-14
Updating Two Columns with a Subquery	8-15
Updating Rows Based on Another Table	8-16
Removing a Row from a Table	8-17
DELETE Statement	8-18
Deleting Rows from a Table	8-19
Deleting Rows Based on Another Table	8-20
TRUNCATE Statement	8-21
Using a Subquery in an INSERT Statement	8-22
Database Transactions	8-24
Advantages of COMMIT and ROLLBACK Statements	8-26
Controlling Transactions	8-27
Rolling Back Changes to a Marker	8-28
Implicit Transaction Processing	8-29
State of the Data Before COMMIT or ROLLBACK	8-31
State of the Data After COMMIT	8-32
Committing Data	8-33
State of the Data After ROLLBACK	8-34
Statement-Level Rollback	8-36
Read Consistency	8-37
Implementation of Read Consistency	8-38
Summary	8-39
Practice 8: Overview	8-40

## **9 Using DDL Statements to Create and Manage Tables**

Objectives	9-2
Database Objects	9-3
Naming Rules	9-4
CREATE TABLE Statement	9-5
Referencing Another User's Tables	9-6
DEFAULT Option	9-7
Creating Tables	9-8
Data Types	9-9

Datetime Data Types	9-11
Including Constraints	9-17
Constraint Guidelines	9-18
Defining Constraints	9-19
NOT NULL Constraint	9-21
UNIQUE Constraint	9-22
PRIMARY KEY Constraint	9-24
FOREIGN KEY Constraint	9-25
FOREIGN KEY Constraint: Keywords	9-27
CHECK Constraint	9-28
CREATE TABLE: Example	9-29
Violating Constraints	9-30
Creating a Table by Using a Subquery	9-32
ALTER TABLE Statement	9-34
Dropping a Table	9-35
Summary	9-36
Practice 9: Overview	9-37

## **10 Creating Other Schema Objects**

Objectives	10-2
Database Objects	10-3
What Is a View?	10-4
Advantages of Views	10-5
Simple Views and Complex Views	10-6
Creating a View	10-7
Retrieving Data from a View	10-10
Modifying a View	10-11
Creating a Complex View	10-12
Rules for Performing DML Operations on a View	10-13
Using the WITH CHECK OPTION Clause	10-16
Denying DML Operations	10-18
Removing a View	10-20
Practice 10: Overview of Part 1	10-21
Sequences	10-22
CREATE SEQUENCE Statement: Syntax	10-24
Creating a Sequence	10-25
NEXTVAL and CURRVAL Pseudocolumns	10-26
Using a Sequence	10-28
Caching Sequence Values	10-29
Modifying a Sequence	10-30

Guidelines for Modifying a Sequence	10-31
Indexes	10-33
How Are Indexes Created?	10-35
Creating an Index	10-36
Index Creation Guidelines	10-37
Removing an Index	10-38
Synonyms	10-39
Creating and Removing Synonyms	10-41
Summary	10-42
Practice 10: Overview of Part 2	10-43

## **11 Managing Objects with Data Dictionary Views**

Objectives	11-2
The Data Dictionary	11-3
Data Dictionary Structure	11-4
How to Use the Dictionary Views	11-6
USER_OBJECTS and ALL_OBJECTS Views	11-7
USER_OBJECTS View	11-8
Table Information	11-9
Column Information	11-10
Constraint Information	11-12
View Information	11-15
Sequence Information	11-16
Synonym Information	11-18
Adding Comments to a Table	11-19
Summary	11-20
Practice 11: Overview	11-21

### **A Practice Solutions**

### **B Table Descriptions and Data**

### **C Oracle Join Syntax**

### **D Using SQL\*Plus**

### **E Using SQL Developer**

### **Index**

**Additional Practices**

**Additional Practices: Table Descriptions and Data**

**Additional Practices: Solutions**

---

## Additional Practices

---

## Additional Practices

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

1. The HR department needs to find data for all the clerks who were hired after 1997.

	EMP...	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COM...	MAN...	DEPART...
1	143	Randall	Matos	RMATOS	650.121.2874	15-MAR-98	ST_CLERK	2600	(null)	124	50
2	144	Peter	Vargas	PVARGAS	650.121.2004	09-JUL-98	ST_CLERK	2500	(null)	124	50

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

	LAST_NAME	JOB_ID	SALARY	COMMISSION_PCT
1	Abel	SA_REP	11000	0.3
2	Zlotkey	SA_MAN	10500	0.2
3	Taylor	SA_REP	8600	0.2
4	Grant	SA_REP	7000	0.15

3. 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).

	New salary
1	The salary of Whalen after a 10% raise is 4840
2	The salary of Hartstein after a 10% raise is 14300
3	The salary of Fay after a 10% raise is 6600
4	The salary of Higgins after a 10% raise is 13200
5	The salary of Gietz after a 10% raise is 9130
6	The salary of King after a 10% raise is 26400
7	The salary of Kochhar after a 10% raise is 18700
8	The salary of De Haan after a 10% raise is 18700
9	The salary of Hunold after a 10% raise is 9900
10	The salary of Ernst after a 10% raise is 6600
11	The salary of Lorentz after a 10% raise is 4620
12	The salary of Mourgos after a 10% raise is 6380
13	The salary of Rajs after a 10% raise is 3850
14	The salary of Davies after a 10% raise is 3410
15	The salary of Matos after a 10% raise is 2860
16	The salary of Vargas after a 10% raise is 2750

**Additional Practices (continued)**

4. Create a report of employees and their duration of 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.

	LAST_NAME	YEARS	MONTHS
1	King	21	5
2	Whalen	21	2
3	Kochhar	19	2
4	Hunold	18	10
5	Ernst	17	6

...

19	Mourgos	9	0
20	Zlotkey	8	9

5. Show those employees who have a last name starting with the letters *J*, *K*, *L*, or *M*.

	LAST_NAME
1	King
2	Kochhar
3	Lorentz
4	Matos
5	Mourgos

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

	LAST_NAME	SALARY	COMMISSION
1	Whalen	4400	No
2	Hartstein	13000	No
3	Fay	6000	No

...

19	Taylor	8600	Yes
20	Grant	7000	Yes

**Additional Practices (continued)**

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

7. Create a report that displays the department name, location, name, job title, and salary of those employees who work in a specific location. Prompt the user for the location. For example, if the user enters 1800, the following are the results:

	DEPARTMENT_NAME	LOCATION_ID	LAST_NAME	JOB_ID	SALARY
1	Marketing	1800	Hartstein	MK_MAN	13000
2	Marketing	1800	Fay	MK_REP	6000

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

	COUNT(*)
1	3

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

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID	COUNT(E.EMPLOYEE_ID)
1	80	Sales	2500	3
2	110	Accounting	1700	2
3	10	Administration	1700	1
4	60	IT	1400	3
5	20	Marketing	1800	2
6	90	Executive	1700	3
7	50	Shipping	1500	5
8	190	Contracting	1700	0

10. 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.

	JOB_ID
1	AD_ASST
2	MK_MAN
3	MK_REP

11. 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.

	JOB_ID	FREQUENCY
1	AD_VP	2
2	AD_PREP	1
3	AD_ASST	1



**Additional Practices (continued)**

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

12. Show all employees who were hired in the first half of the month (before the 16th of the month).

	LAST_NAME	HIRE_DATE
1	Higgins	07-JUN-94
2	Gietz	07-JUN-94
3	De Haan	13-JAN-93
4	Hunold	03-JAN-90
5	Lorentz	07-FEB-99
6	Matos	15-MAR-98
7	Vargas	09-JUL-98
8	Abel	11-MAY-96

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

	LAST_NAME	SALARY	THOUSANDS
1	Whalen	4400	4
2	Hartstein	13000	13
3	Fay	6000	6
4	Higgins	12000	12

...

16	Vargas	2500	2
17	Zlotkey	10500	10
18	Abel	11000	11
19	Taylor	8600	8
20	Grant	7000	7

14. 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.

	LAST_NAME	MANAGER	SALARY	GRADE_LEVEL
1	Whalen	Kochhar	17000	E
2	Higgins	Kochhar	17000	E
3	Hunold	De Haan	17000	E
4	Hartstein	King	24000	E
5	Kochhar	King	24000	E
6	De Haan	King	24000	E
7	Mourgos	King	24000	E
8	Zlotkey	King	24000	E

**Additional Practices (continued)**

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

	DEP...	DEPARTMENT_NAME	EMPLOYEES	AVG_SAL	LAST_NAME	SALARY	JOB_ID
1	10	Administration	1	4400.00	Whalen	4400	AD_ASST
2	20	Marketing	2	9500.00	Hartstein	13000	MK_MAN
3	20	Marketing	2	9500.00	Fay	6000	MK_REP
4	50	Shipping	5	3500.00	Rajs	3500	ST_CLERK
5	50	Shipping	5	3500.00	Mourgos	5800	ST_MAN
6	50	Shipping	5	3500.00	Vargas	2500	ST_CLERK
7	50	Shipping	5	3500.00	Davies	3100	ST_CLERK
8	50	Shipping	5	3500.00	Matos	2600	ST_CLERK
9	60	IT	3	6400.00	Hunold	9000	IT_PROG
10	60	IT	3	6400.00	Lorentz	4200	IT_PROG
11	60	IT	3	6400.00	Ernst	6000	IT_PROG
12	80	Sales	3	10033.33	Taylor	8600	SA_REP
13	80	Sales	3	10033.33	Zlotkey	10500	SA_MAN
14	80	Sales	3	10033.33	Abel	11000	SA_REP
15	90	Executive	3	19333.33	De Haan	17000	AD_VP
16	90	Executive	3	19333.33	Kochhar	17000	AD_VP
17	90	Executive	3	19333.33	King	24000	AD_PRES
18	110	Accounting	2	10150.00	Higgins	12000	AC_MGR
19	110	Accounting	2	10150.00	Gietz	8300	AC_ACCOUNT
20	(null)	(null)	0	No average	Grant	7000	SA_REP

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

	DEPARTMENT_ID	MIN(SALARY)
1	90	17000

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

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	90	Executive	100	1700
6	110	Accounting	205	1700
7	190	Contracting	(null)	1700

**Additional Practices (continued)**

18. 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:

a. Employs fewer than three employees:

	DEPARTMENT_ID	DEPARTMENT_NAME	COUNT(*)
1	10	Administration	1
2	110	Accounting	2
3	20	Marketing	2

b. Has the highest number of employees:

	DEPARTMENT_ID	DEPARTMENT_NAME	COUNT(*)
1	50	Shipping	5

c. Has the lowest number of employees:

	DEPARTMENT_ID	DEPARTMENT_NAME	COUNT(*)
1	10	Administration	1

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

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	SALARY	AVG(S.SALARY)
1	149	Zlotkey	80	10500	10033.33333333333333333333...
2	174	Abel	80	11000	10033.33333333333333333333...
3	144	Vargas	50	2500	3500
4	205	Higgins	110	12000	10150
5	100	King	90	24000	19333.33333333333333333333...
6	101	Kochhar	90	17000	19333.33333333333333333333...
7	103	Hunold	60	9000	6400
8	142	Davies	50	3100	3500
9	104	Ernst	60	6000	6400
10	143	Matos	50	2600	3500
11	200	Whalen	10	4400	4400
12	202	Fay	20	6000	9500
13	102	De Haan	90	17000	19333.33333333333333333333...
14	107	Lorentz	60	4200	6400
15	141	Rajs	50	3500	3500
16	201	Hartstein	20	13000	9500
17	206	Gietz	110	8300	10150
18	176	Taylor	80	8600	10033.33333333333333333333...
19	124	Mourgos	50	5800	3500

**Additional Practices (continued)**

20. Show all employees who were hired on the day of the week on which the highest number of employees were hired.

	LAST_NAME	DAY
1	Higgins	TUESDAY
2	Gietz	TUESDAY
3	Ernst	TUESDAY
4	Mourgos	TUESDAY
5	Rajs	TUESDAY
6	Taylor	TUESDAY

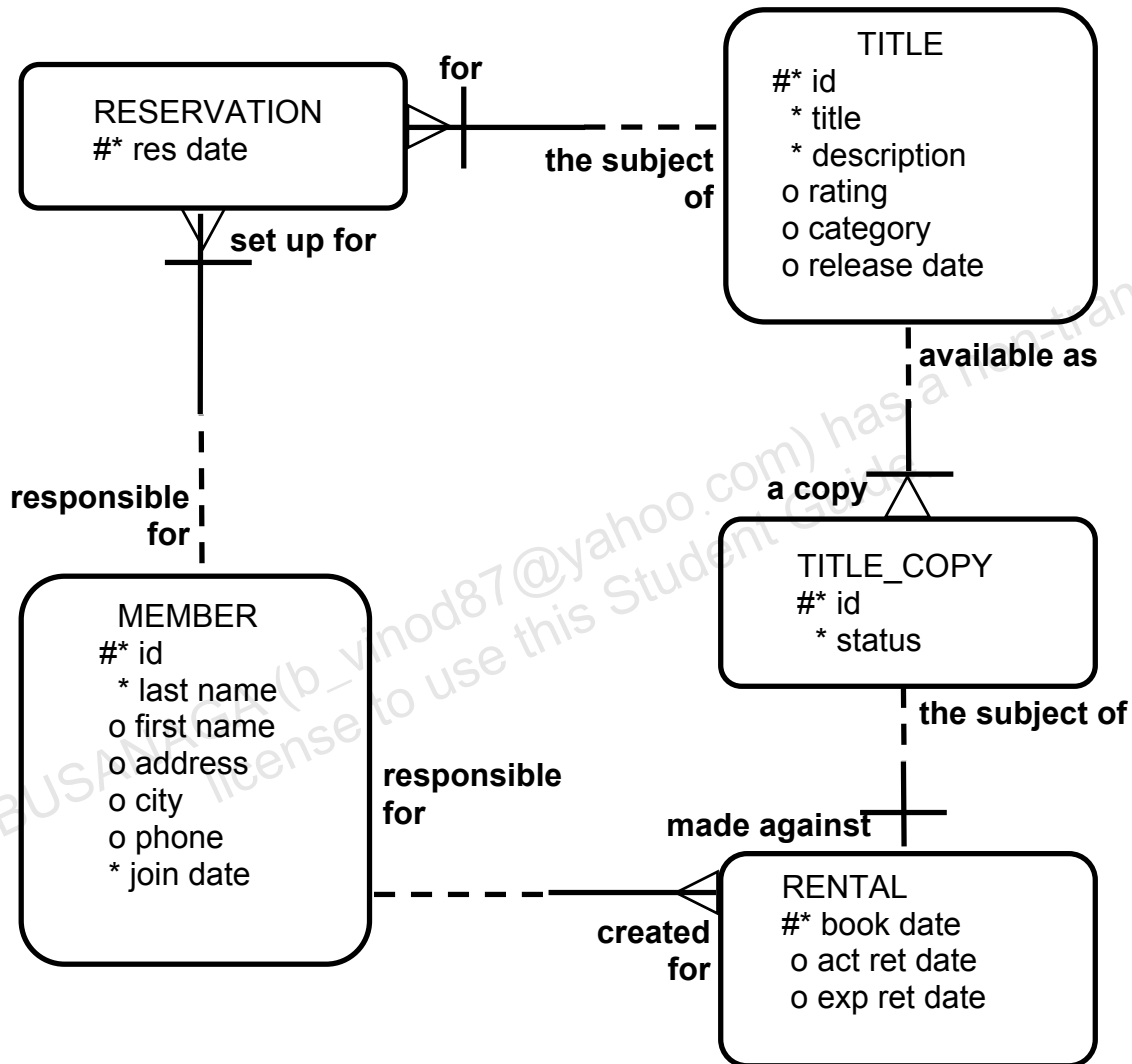
21. Create an anniversary overview based on the hire date of the employees. Sort the anniversaries in ascending order.

	LAST_NAME	BIRTHDAY
1	Hunold	January 03
2	De Haan	January 13
3	Zlotkey	January 29
4	Davies	January 29
5	Lorentz	February 07
6	Hartstein	February 17
7	Matos	March 15
8	Taylor	March 24
9	Abel	May 11
10	Ernst	May 21
11	Grant	May 24
12	Gietz	June 07
13	Higgins	June 07
14	King	June 17
15	Vargas	July 09
16	Fay	August 17
17	Whalen	September 17
18	Kochhar	September 21
19	Rajs	October 17
20	Mourgos	November 16

## Additional Practices: Case Study

In this case study, you build a set of database tables for a video application. After you create the tables, you insert, update, and delete records in a video store database and generate a report. The database contains only the essential tables.

The following is a diagram of the entities and attributes for the video application:



**Note:** If you want to build the tables, you can execute the commands in the `buildtab.sql` script in SQL Developer. If you want to drop the tables, you can execute the commands in the `dropvid.sql` script in SQL Developer. Then you can execute the commands in the `buildvid.sql` script in SQL Developer to create and populate the tables.

- If you use the `buildtab.sql` script to build the tables, start with step 4.
- If you use the `dropvid.sql` script to remove the video tables, start with step 1.
- If you use the `buildvid.sql` script to build and populate the tables, start with step 6(b).

**Additional Practices: Case Study (continued)**

1. Create the tables based on the following table instance charts. Choose the appropriate data types and ensure that you add integrity constraints.

a. Table name: MEMBER

Column_ Name	MEMBER_ ID	LAST_ NAME	FIRST_NAME	ADDRESS	CITY	PHONE	JOIN _DATE
Key Type	PK						
Null/ Unique	NN,U	NN					NN
Default Value							System Date
Data Type	NUMBER	VARCHAR2	VARCHAR2	VARCHAR2	VARCHAR2	VARCHAR2	DATE
Length	10	25	25	100	30	15	

b. Table name: TITLE

Column_ Name	TITLE_ID	TITLE	DESCRIPTION	RATING	CATEGORY	RELEASE_ DATE
Key Type	PK					
Null/ Unique	NN,U	NN	NN			
Check				G, PG, R, NC17, NR	DRAMA, COMEDY, ACTION, CHILD, SCIFI, DOCUMENTARY	
Data Type	NUMBER	VARCHAR2	VARCHAR2	VARCHAR2	VARCHAR2	DATE
Length	10	60	400	4	20	

**Additional Practices: Case Study (continued)**

c. Table name: TITLE\_COPY

<b>Column Name</b>	COPY_ID	TITLE_ID	STATUS
<b>Key Type</b>	PK	PK,FK	
<b>Null/Unique</b>	NN,U	NN,U	NN
<b>Check</b>			AVAILABLE, DESTROYED, RENTED, RESERVED
<b>FK Ref Table</b>		TITLE	
<b>FK Ref Col</b>		TITLE_ID	
<b>Data Type</b>	NUMBER	NUMBER	VARCHAR2
<b>Length</b>	10	10	15

d. Table name: RENTAL

<b>Column Name</b>	BOOK_DATE	MEMBER_ID	COPY_ID	ACT_RET_DATE	EXP_RET_DATE	TITLE_ID
<b>Key Type</b>	PK	PK,FK1	PK,FK2			PK,FK2
<b>Default Value</b>	System Date				System Date + 2 days	
<b>FK Ref Table</b>		MEMBER	TITLE_COPY			TITLE_COPY
<b>FK Ref Col</b>		MEMBER_ID	COPY_ID			TITLE_ID
<b>Data Type</b>	DATE	NUMBER	NUMBER	DATE	DATE	NUMBER
<b>Length</b>		10	10			10

**Additional Practices: Case Study (continued)**

e. Table name: RESERVATION

Column Name	RES_DATE	MEMBER_ID	TITLE_ID
Key Type	PK	PK,FK1	PK,FK2
Null/Unique	NN,U	NN,U	NN
FK Ref Table		MEMBER	TITLE
FK Ref Column		MEMBER_ID	TITLE_ID
Data Type	DATE	NUMBER	NUMBER
Length		10	10

2. Verify that the tables and constraints were created properly by checking the data dictionary.

TABLE_NAME
1 MEMBER
2 RENTAL
3 RESERVATION
4 TITLE
5 TITLE_COPY

CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME
1 MEMBER_LAST_NAME_NN	C	MEMBER
2 MEMBER_JOIN_DATE_NN	C	MEMBER
3 MEMBER_MEMBER_ID_PK	P	MEMBER
4 RENTAL_BOOK_DATE_COPY_TITLE_PK	P	RENTAL
5 RENTAL_MEMBER_ID_FK	R	RENTAL
6 RENTAL_COPY_ID_TITLE_ID_FK	R	RENTAL
7 RESERVATION_RESDATE_MEM_TIT_PK	P	RESERVATION
8 RESERVATION_MEMBER_ID	R	RESERVATION
9 RESERVATION_TITLE_ID	R	RESERVATION
10 TITLE_TITLE_NN	C	TITLE
11 TITLE_DESCRIPTION_NN	C	TITLE
12 TITLE_RATING_CK	C	TITLE
13 TITLE_CATEGORY_CK	C	TITLE
14 TITLE_TITLE_ID_PK	P	TITLE
15 TITLE_COPY_STATUS_NN	C	TITLE_COPY
16 TITLE_COPY_STATUS_CK	C	TITLE_COPY
17 TITLE_COPY_COPY_ID_TITLE_ID_PK	P	TITLE_COPY
18 TITLE_COPY_TITLE_IF_FK	R	TITLE_COPY



**Additional Practices: Case Study (continued)**

3. Create sequences to uniquely identify each row in the MEMBER table and the TITLE table.

- a. Member number for the MEMBER table: Start with 101; do not allow caching of values. Name the sequence MEMBER\_ID\_SEQ.
- b. Title number for the TITLE table: Start with 92; do not allow caching of values. Name the sequence TITLE\_ID\_SEQ.
- c. Verify the existence of the sequences in the data dictionary.

	SEQUENCE_NAME	INCREMENT_BY	LAST_NUMBER
1	MEMBER_ID_SEQ	1	101
2	TITLE_ID_SEQ	1	92

4. Add data to the tables. Create a script for each set of data to be added.

- a. Add movie titles to the TITLE table. Write a script to enter the movie information. Save the statements in a script named lab\_apcs\_4a.sql. Use the sequences to uniquely identify each title. Enter the release dates in the DD-MON-YYYY format. Remember that single quotation marks in a character field must be specially handled. Verify your additions.

	TITLE
1	Willie and Christmas Too
2	Alien Again
3	The Glob
4	My Day Off
5	Miracles on Ice
6	Soda Gang

**Additional Practices: Case Study (continued)**

<b>Title</b>	<b>Description</b>	<b>Rating</b>	<b>Category</b>	<b>Release_date</b>
Willie and Christmas Too	All of Willie's friends make a Christmas list for Santa, but Willie is yet to add his own wish list.	G	CHILD	05-OCT-1995
Alien Again	Yet another installation of science fiction history. Can the heroine save the planet from the alien life form?	R	SCIFI	19-MAY-1995
The Glob	A meteor crashes near a small American town and unleashes carnivorous goo in this classic.	NR	SCIFI	12-AUG-1995
My Day Off	With a little luck and a lot of ingenuity, a teenager skips school for a day in New York.	PG	COMEDY	12-JUL-1995
Miracles on Ice	A six-year-old has doubts about Santa Claus, but she discovers that miracles really do exist.	PG	DRAMA	12-SEP-1995
Soda Gang	After discovering a cache of drugs, a young couple find themselves pitted against a vicious gang.	NR	ACTION	01-JUN-1995

- b. Add data to the MEMBER table. Place the INSERT statements in a script named lab\_apcs\_4b.sql. Execute the commands in the script. Be sure to use the sequence to add the member numbers.

<b>First_Name</b>	<b>Last_Name</b>	<b>Address</b>	<b>City</b>	<b>Phone</b>	<b>Join_Date</b>
Carmen	Velasquez	283 King Street	Seattle	206-899-6666	08-MAR-1990
LaDoris	Ngao	5 Modrany	Bratislava	586-355-8882	08-MAR-1990
Midori	Nagayama	68 Via Centrale	Sao Paolo	254-852-5764	17-JUN-1991
Mark	Quick-to-See	6921 King Way	Lagos	63-559-7777	07-APR-1990
Audry	Ropeburn	86 Chu Street	Hong Kong	41-559-87	18-JAN-1991
Molly	Urguhart	3035 Laurier	Quebec	418-542-9988	18-JAN-1991

**Additional Practices: Case Study (continued)**

c. Add the following movie copies in the `TITLE_COPY` table:

**Note:** Have the `TITLE_ID` numbers available for this exercise.

Title	Copy_Id	Status	Title	Copy_Id
Willie and Christmas Too	1	AVAILABLE	Willie and Christmas Too	1
Alien Again	1	AVAILABLE	Alien Again	1
	2	RENTED		2
The Glob	1	AVAILABLE	The Glob	1
My Day Off	1	AVAILABLE	My Day Off	1
	2	AVAILABLE		2
	3	RENTED		3
Miracles on Ice	1	AVAILABLE	Miracles on Ice	1
Soda Gang	1	AVAILABLE	Soda Gang	1

d. Add the following rentals to the `RENTAL` table:

**Note:** The title number may be different depending on the sequence number.

Title_Id	Copy_Id	Member_Id	Book_date	Exp_Ret_Date
92	1	101	3 days ago	1 day ago
93	2	101	1 day ago	1 day from now
95	3	102	2 days ago	Today
97	1	106	4 days ago	2 days ago

**Additional Practices: Case Study (continued)**

5. Create a view named `TITLE_AVAIL` to show the movie titles, the availability of each copy, and its expected return date if rented. Query all rows from the view. Order the results by title.

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

	TITLE	COPY_ID	STATUS	EXP_RET_DATE
1	Alien Again	1	AVAILABLE	(null)
2	Alien Again	2	RENTED	26-NOV-08
3	Miracles on Ice	1	AVAILABLE	(null)
4	My Day Off	1	AVAILABLE	(null)
5	My Day Off	2	AVAILABLE	(null)
6	My Day Off	3	RENTED	27-NOV-08
7	Soda Gang	1	AVAILABLE	25-NOV-08
8	The Glob	1	AVAILABLE	(null)
9	Willie and Christmas Too	1	AVAILABLE	26-NOV-08

6. Make changes to the data in the tables.
- Add a new title. The movie is “Interstellar Wars,” which is rated PG and classified as a science fiction movie. The release date is 07-JUL-77. The description is “Futuristic interstellar action movie. Can the rebels save the humans from the evil empire?” Be sure to add a title copy record for two copies.
  - Enter two reservations. One reservation is for Carmen Velasquez, who wants to rent “Interstellar Wars.” The other is for Mark Quick-to-See, who wants to rent “Soda Gang.”
7. Make a modification to one of the tables.
- Run the script in `lab_apcs_7a.sql` to add a `PRICE` column to the `TITLE` table to record the purchase price of the video. Verify your modifications.

Name	Null	Type
TITLE_ID	NOT NULL	NUMBER(10)
TITLE	NOT NULL	VARCHAR2(60)
DESCRIPTION	NOT NULL	VARCHAR2(400)
RATING		VARCHAR2(4)
CATEGORY		VARCHAR2(20)
RELEASE_DATE		DATE
PRICE		NUMBER(8,2)

Title	Price
Willie and Christmas Too	25
Alien Again	35
The Glob	35
My Day Off	35
Miracles on Ice	30
Soda Gang	35
Interstellar Wars	29

- b. Create a script named `lab_apcs_7b.sql` that contains UPDATE statements that update each video with a price according to the preceding list. Run the commands in the script.

**Note:** Have the `TITLE_ID` numbers available for this exercise.

8. Create a report that contains each customer's history of renting videos. Be sure to include the customer name, movie rented, dates of the rental, and duration of rentals. Total the number of rentals for all customers for the reporting period. Save the commands that generate the report in a script file named `lab_apcs_8.sql`.

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

	MEMBER	TITLE	BOOK_DATE	DURATION
1	Carmen Velasquez	Willie and Christmas Too	24-NOV-08	1
2	Carmen Velasquez	Alien Again	26-NOV-08	(null)
3	LaDoris Ngao	My Day Off	25-NOV-08	(null)
4	Molly Urguhart	Soda Gang	23-NOV-08	2

VINOD BUSANAGA (b\_vinod87@yahoo.com) has a non-transferable  
license to use this Student Guide.

---

# Additional Practices

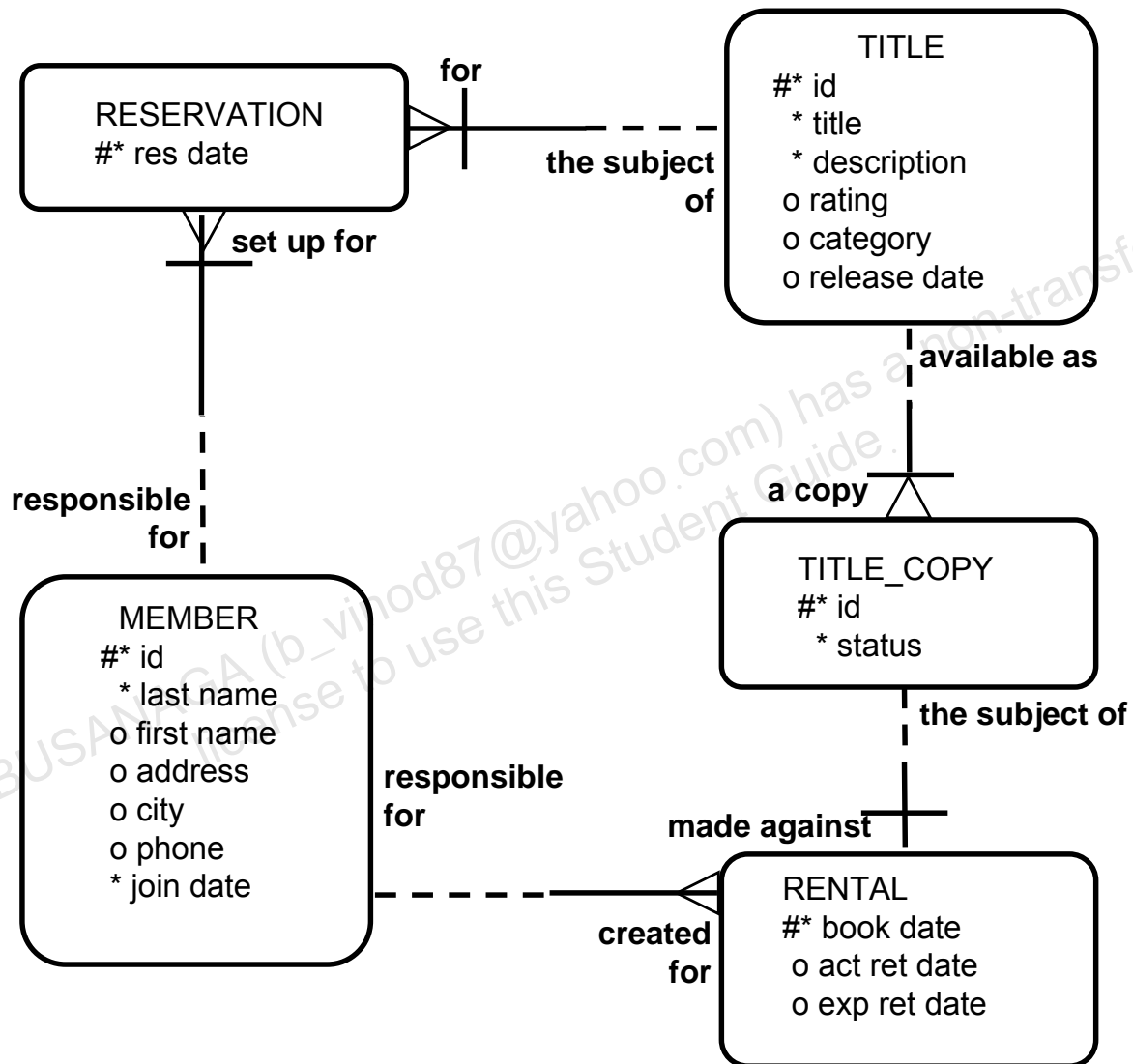
---

## Table Descriptions and Data

## Tables Used in Additional Practices

Additional practice questions 1-21 use the HR schema. Refer to Appendix B to look at the HR schema tables. The tables used in the additional practices: case study are described below.

Note: These table do not exist by default. You will be creating them in the case study practice questions.





**RESERVATION Table**

DESCRIBE reservation

Name	Null	Type
RES_DATE	NOT NULL	DATE
MEMBER_ID	NOT NULL	NUMBER(10)
TITLE_ID	NOT NULL	NUMBER(10)

**MEMBER Table**

DESCRIBE member

Name	Null	Type
MEMBER_ID	NOT NULL	NUMBER(10)
LAST_NAME	NOT NULL	VARCHAR2(25)
FIRST_NAME		VARCHAR2(25)
ADDRESS		VARCHAR2(100)
CITY		VARCHAR2(30)
PHONE		VARCHAR2(15)
JOIN_DATE	NOT NULL	DATE

**TITLE Table**

DESCRIBE title

Name	Null	Type
TITLE_ID	NOT NULL	NUMBER(10)
TITLE	NOT NULL	VARCHAR2(60)
DESCRIPTION	NOT NULL	VARCHAR2(400)
RATING		VARCHAR2(4)
CATEGORY		VARCHAR2(20)
RELEASE_DATE		DATE

**TITLE\_COPY Table**

DESCRIBE title\_copy

Name	Null	Type
COPY_ID	NOT NULL	NUMBER(10)
TITLE_ID	NOT NULL	NUMBER(10)
STATUS	NOT NULL	VARCHAR2(15)

**RENTAL Table**

DESCRIBE rental

Name	Null	Type
BOOK_DATE	NOT NULL	DATE
MEMBER_ID	NOT NULL	NUMBER(10)
COPY_ID	NOT NULL	NUMBER(10)
ACT_RET_DATE		DATE
EXP_RET_DATE		DATE
TITLE_ID	NOT NULL	NUMBER(10)

VINOD BUSANAGA (b\_vinod87@yahoo.com) has a non-transferable  
license to use this Student Guide.

---

## **Additional Practices: Solutions**

---

## Additional Practices: Solutions

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

1. The HR department needs to find data for all the clerks who were hired after 1997.

```
SELECT *
FROM   employees
WHERE  job_id = 'ST_CLERK'
AND    hire_date > '31-DEC-1997';
```

2. The HR department needs a report of employees who earn 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;
```

3. 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).

```
SELECT 'The salary of ' || last_name || ' after a 10% raise is '
      || ROUND(salary*1.10) "New salary"
FROM   employees
WHERE  commission_pct IS NULL;
```

4. Create a report of employees and their duration of employment. Show the last names of all the employees along 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;
```

5. Show those employees who have a last name starting with the letters *J*, *K*, *L*, or *M*.

```
SELECT last_name
FROM   employees
WHERE  SUBSTR(last_name, 1,1) IN ('J', 'K', 'L', 'M');
```

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

```
SELECT last_name, salary,
       decode(commission_pct, NULL, 'No', 'Yes') commission
FROM   employees;
```



**Additional Practices: Solutions (continued)**

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

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

```
SELECT d.department_name, d.location_id, e.last_name, e.job_id, e.salary
FROM   employees e, departments d
WHERE  e.department_id = d.department_id
AND    d.location_id = &dept_no;
```

8. 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';
```

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

```
SELECT d.department_id, d.department_name,
       d.location_id, COUNT(e.employee_id)
FROM   employees e RIGHT OUTER JOIN departments d
ON     e.department_id = d.department_id
GROUP BY d.department_id, d.department_name, d.location_id;
```

10. 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);
```

11. 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, SQL functions, joins, group functions, and subqueries.

12. Show all employees who were hired in the first half of the month (before the 16th of the month).

**Additional Practices: Solutions (continued)**

```
SELECT last_name, hire_date
FROM   employees
WHERE  TO_CHAR(hire_date, 'DD') < 16;
```

13. 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;
```

14. 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;
```

15. Show the department number, name, number of employees, and average salary of all departments together with the 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;
```

16. 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);
```

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

```
SELECT *
FROM   departments
WHERE  department_id NOT IN(SELECT department_id
                           FROM employees)
```

**Additional Practices: Solutions (continued)**

```
WHERE job_id = 'SA REP'
AND department_id IS NOT NULL);
```

18. 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:

a. 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;
```

b. 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);
```

c. 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);
```

19. 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;
```

20. Show all employees who were hired on the day of the week on which the highest number of employees were hired.

```
SELECT last_name, TO_CHAR(hire_date, 'DAY') day
FROM   employees
WHERE  TO_CHAR(hire_date, 'Day') =
      (SELECT TO_CHAR(hire_date, 'Day')
       FROM   employees
       GROUP BY TO_CHAR(hire_date, 'Day')
       HAVING COUNT(*) = (SELECT MAX(COUNT(*))
                          FROM   employees
                          GROUP BY TO_CHAR(hire_date, 'Day')));
```

**Additional Practices: Solutions (continued)**

21. Create an anniversary overview based on the hire date of the 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: Case Study Solutions**

1. Create tables based on the following table instance charts. Choose the appropriate data types and be sure to add integrity constraints.

- a. Table name: MEMBER

```
CREATE TABLE member
  (member_id      NUMBER(10)
   CONSTRAINT member_member_id_pk PRIMARY KEY,
   last_name      VARCHAR2(25)
   CONSTRAINT member_last_name_nn NOT NULL,
   first_name     VARCHAR2(25),
   address        VARCHAR2(100),
   city           VARCHAR2(30),
   phone          VARCHAR2(15),
   join_date      DATE DEFAULT SYSDATE
   CONSTRAINT member_join_date_nn NOT NULL);
```

- b. Table name: TITLE

```
CREATE TABLE title
  (title_id       NUMBER(10)
   CONSTRAINT title_title_id_pk PRIMARY KEY,
   title          VARCHAR2(60)
   CONSTRAINT title_title_nn NOT NULL,
   description    VARCHAR2(400)
   CONSTRAINT title_description_nn NOT NULL,
   rating         VARCHAR2(4)
   CONSTRAINT title_rating_ck CHECK
     (rating IN ('G', 'PG', 'R', 'NC17', 'NR')),
   category       VARCHAR2(20)
   CONSTRAINT title_category_ck CHECK
     (category IN ('DRAMA', 'COMEDY', 'ACTION',
                  'CHILD', 'SCIFI', 'DOCUMENTARY')),
   release_date   DATE);
```

- c. Table name: TITLE\_COPY

```
CREATE TABLE title_copy
  (copy_id        NUMBER(10),
   title_id       NUMBER(10)
   CONSTRAINT title_copy_title_id_fk REFERENCES title(title_id),
   status         VARCHAR2(15)
   CONSTRAINT title_copy_status_nn NOT NULL
   CONSTRAINT title_copy_status_ck CHECK (status IN
     ('AVAILABLE', 'DESTROYED', 'RENTED', 'RESERVED')),
   CONSTRAINT title_copy_copy_id_title_id_pk
     PRIMARY KEY (copy_id, title_id));
```

- d. Table name: RENTAL

**Additional Practices: Case Study Solutions (continued)**

```

CREATE TABLE rental
  (book_date      DATE DEFAULT SYSDATE,
   member_id      NUMBER(10)
     CONSTRAINT rental_member_id_fk REFERENCES member(member_id),
   copy_id        NUMBER(10),
   act_ret_date   DATE,
   exp_ret_date   DATE DEFAULT SYSDATE + 2,
   title_id       NUMBER(10),
   CONSTRAINT rental_book_date_copy_title_pk
     PRIMARY KEY (book_date, member_id, copy_id, title_id),
   CONSTRAINT rental_copy_id_title_id_fk
     FOREIGN KEY (copy_id, title_id)
       REFERENCES title_copy(copy_id, title_id));

```

e. Table name: RESERVATION

```

CREATE TABLE reservation
  (res_date       DATE,
   member_id      NUMBER(10)
     CONSTRAINT reservation_member_id REFERENCES member(member_id),
   title_id       NUMBER(10)
     CONSTRAINT reservation_title_id REFERENCES title(title_id),
   CONSTRAINT reservation_resdate_mem_tit_pk PRIMARY KEY
     (res_date, member_id, title_id));

```

2. Verify that the tables and constraints were created properly by checking the data dictionary.

```

SELECT  table_name
FROM    user_tables
WHERE   table_name IN ('MEMBER', 'TITLE', 'TITLE_COPY',
                      'RENTAL', 'RESERVATION');

SELECT  constraint_name, constraint_type, table_name
FROM    user_constraints
WHERE   table_name IN ('MEMBER', 'TITLE', 'TITLE_COPY',
                      'RENTAL', 'RESERVATION');

```

3. Create sequences to uniquely identify each row in the MEMBER table and the TITLE table.

a. Member number for the MEMBER table: Start with 101; do not allow caching of values.  
Name the sequence MEMBER\_ID\_SEQ.

```

CREATE SEQUENCE member_id_seq
START WITH 101
NOCACHE;

```

b. Title number for the TITLE table: Start with 92; do not allow caching of values. Name the sequence TITLE\_ID\_SEQ.

```

CREATE SEQUENCE title_id_seq
START WITH 92
NOCACHE;

```

c. Verify the existence of the sequences in the data dictionary.

**Additional Practices: Case Study Solutions (continued)**

```

SELECT  sequence_name, increment_by, last_number
FROM    user_sequences
WHERE   sequence_name IN ('MEMBER_ID_SEQ', 'TITLE_ID_SEQ');

```

4. Add data to the tables. Create a script for each set of data to be added.
- Add movie titles to the TITLE table. Write a script to enter the movie information. Save the statements in a script named `lab_apcs_4a.sql`. Use the sequences to uniquely identify each title. Enter the release dates in the DD-MON-YYYY format. Remember that single quotation marks in a character field must be specially handled. Verify your additions.

```

INSERT INTO title(title_id, title, description, rating,
                  category, release_date)
VALUES (title_id_seq.NEXTVAL, 'Willie and Christmas Too',
        'All of Willie''s friends make a Christmas list for
        Santa, but Willie has yet to add his own wish list.',
        'G', 'CHILD', TO_DATE('05-OCT-1995','DD-MON-YYYY'))
/
INSERT INTO title(title_id, title, description, rating,
                  category, release_date)
VALUES (title_id_seq.NEXTVAL, 'Alien Again', 'Yet another
        installment of science fiction history. Can the
        heroine save the planet from the alien life form?',
        'R', 'SCIFI', TO_DATE('19-MAY-1995','DD-MON-YYYY'))
/
INSERT INTO title(title_id, title, description, rating,
                  category, release_date)
VALUES (title_id_seq.NEXTVAL, 'The Glob', 'A meteor crashes
        near a small American town and unleashes carnivorous
        goo in this classic.', 'NR', 'SCIFI',
        TO_DATE('12-AUG-1995','DD-MON-YYYY'))
/
INSERT INTO title(title_id, title, description, rating,
                  category, release_date)
VALUES (title_id_seq.NEXTVAL, 'My Day Off', 'With a little
        luck and a lot ingenuity, a teenager skips school for
        a day in New York.', 'PG', 'COMEDY',
        TO_DATE('12-JUL-1995','DD-MON-YYYY'))
/
...
COMMIT
/
SELECT  title
FROM    title;

```

- Add data to the MEMBER table. Place the INSERT statements in a script named `lab_apcs_4b.sql`. Execute the commands in the script. Be sure to use the sequence to add the member numbers.

```

SET VERIFY OFF
INSERT INTO member(member_id, first_name, last_name,
                  address, city, phone, join_date)

```

**Additional Practices: Case Study Solutions (continued)**

```

VALUES (member_id_seq.NEXTVAL, 'Carmen', 'Velasquez',
        '283 King Street', 'Seattle', '206-899-6666', TO_DATE('08-MAR-
1990',
        'DD-MM-YYYY'))
/
INSERT INTO member(member_id, first_name, last_name,
                    address, city, phone, join_date)
VALUES (member_id_seq.NEXTVAL, 'LaDoris', 'Ngao',
        '5 Modrany', 'Bratislava', '586-355-8882', TO_DATE('08-MAR-1990',
        'DD-MM-YYYY'))
/
INSERT INTO member(member_id, first_name, last_name,
                    address, city, phone, join_date)
VALUES (member_id_seq.NEXTVAL, 'Midori', 'Nagayama',
        '68 Via Centrale', 'Sao Paolo', '254-852-5764', TO_DATE('17-JUN-
1991',
        'DD-MM-YYYY'))
/
INSERT INTO member(member_id, first_name, last_name,
                    address, city, phone, join_date)
VALUES (member_id_seq.NEXTVAL, 'Mark', 'Quick-to-See',
        '6921 King Way', 'Lagos', '63-559-7777', TO_DATE('07-APR-1990',
        'DD-MM-YYYY'))
/
INSERT INTO member(member_id, first_name, last_name,
                    address, city, phone, join_date)
VALUES (member_id_seq.NEXTVAL, 'Audry', 'Ropeburn',
        '86 Chu Street', 'Hong Kong', '41-559-87', TO_DATE('18-JAN-1991',
        'DD-MM-YYYY'))
/
INSERT INTO member(member_id, first_name, last_name,
                    address, city, phone, join_date)
VALUES (member_id_seq.NEXTVAL, 'Molly', 'Urguhart',
        '3035 Laurier', 'Quebec', '418-542-9988', TO_DATE('18-JAN-1991',
        'DD-MM-YYYY'));
/
COMMIT
SET VERIFY ON

```

c. Add the following movie copies in the TITLE\_COPY table:

**Note:** Have the TITLE\_ID numbers available for this exercise.

```

INSERT INTO title_copy(copy_id, title_id, status)
VALUES (1, 92, 'AVAILABLE')
/
INSERT INTO title_copy(copy_id, title_id, status)
VALUES (1, 93, 'AVAILABLE')
/
INSERT INTO title_copy(copy_id, title_id, status)
VALUES (2, 93, 'RENTED')
/
INSERT INTO title_copy(copy_id, title_id, status)
VALUES (1, 94, 'AVAILABLE')
/

```



**Additional Practices: Case Study Solutions (continued)**

```

INSERT INTO title_copy(copy_id, title_id, status)
VALUES (1, 95, 'AVAILABLE')
/
INSERT INTO title_copy(copy_id, title_id,status)
VALUES (2, 95, 'AVAILABLE')
/
INSERT INTO title_copy(copy_id, title_id,status)
VALUES (3, 95, 'RENTED')
/
INSERT INTO title_copy(copy_id, title_id,status)
VALUES (1, 96, 'AVAILABLE')
/
INSERT INTO title_copy(copy_id, title_id,status)
VALUES (1, 97, 'AVAILABLE')
/

```

- d. Add the following rentals to the RENTAL table:

**Note:** The title number may be different depending on the sequence number.

```

INSERT INTO rental(title_id, copy_id, member_id,
                  book_date, exp_ret_date, act_ret_date)
VALUES (92, 1, 101, sysdate-3, sysdate-1, sysdate-2)
/
INSERT INTO rental(title_id, copy_id, member_id,
                  book_date, exp_ret_date, act_ret_date)
VALUES (93, 2, 101, sysdate-1, sysdate-1, NULL)
/
INSERT INTO rental(title_id, copy_id, member_id,
                  book_date, exp_ret_date, act_ret_date)
VALUES (95, 3, 102, sysdate-2, sysdate, NULL)
/
INSERT INTO rental(title_id, copy_id, member_id,
                  book_date, exp_ret_date, act_ret_date)
VALUES (97, 1, 106, sysdate-4, sysdate-2, sysdate-2)
/
COMMIT
/

```

5. Create a view named TITLE\_AVAIL to show the movie titles, the availability of each copy, and its expected return date if rented. Query all rows from the view. Order the results by title.

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

```

CREATE VIEW title_avail AS
SELECT  t.title, c.copy_id, c.status, r.exp_ret_date
FROM    title t JOIN title_copy c
ON      t.title_id = c.title_id
FULL OUTER JOIN rental r
ON      c.copy_id = r.copy_id
AND     c.title_id = r.title_id;

```

**Additional Practices: Case Study Solutions (continued)**

```
SELECT      *
FROM        title_avail
ORDER BY title, copy_id;
```

6. Make changes to the data in the tables.

- a. Add a new title. The movie is “Interstellar Wars,” which is rated PG and classified as a science fiction movie. The release date is 07-JUL-77. The description is “Futuristic interstellar action movie. Can the rebels save the humans from the evil empire?” Be sure to add a title copy record for two copies.

```
INSERT INTO title(title_id, title, description, rating,
                  category, release_date)
VALUES (title_id_seq.NEXTVAL, 'Interstellar Wars',
       'Futuristic interstellar action movie. Can the
       rebels save the humans from the evil empire?',
       'PG', 'SCIFI', '07-JUL-77')
/
INSERT INTO title_copy (copy_id, title_id, status)
VALUES (1, 98, 'AVAILABLE')
/
INSERT INTO title_copy (copy_id, title_id, status)
VALUES (2, 98, 'AVAILABLE')
/
```

- b. Enter two reservations. One reservation is for Carmen Velasquez, who wants to rent “Interstellar Wars.” The other is for Mark Quick-to-See, who wants to rent “Soda Gang.”

```
INSERT INTO reservation (res_date, member_id, title_id)
VALUES (SYSDATE, 101, 98)
/
INSERT INTO reservation (res_date, member_id, title_id)
VALUES (SYSDATE, 104, 97)
/
```

7. Make a modification to one of the tables.

- a. Run the script in lab\_apcs\_7a.sql to add a PRICE column to the TITLE table to record the purchase price of the video. Verify your modifications.

```
ALTER TABLE title
ADD (price NUMBER(8,2));

DESCRIBE title
```

- b. Create a script named lab\_apcs\_7b.sql that contains UPDATE statements that update each video with a price according to the list provided. Run the commands in the script.

**Note:** Have the TITLE\_ID numbers available for this exercise.

```
SET ECHO OFF
SET VERIFY OFF
UPDATE title
SET      price = &price
```

**Additional Practices: Case Study Solutions (continued)**

```
WHERE title_id = &title_id;
SET VERIFY OFF
SET ECHO OFF
```

8. Create a report that contains each customer's history of renting videos. Be sure to include the customer name, movie rented, dates of the rental, and duration of rentals. Total the number of rentals for all customers for the reporting period. Save the commands that generate the report in a script file named lab\_apcs\_8.sql.

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

```
SET ECHO OFF
SET VERIFY OFF
SELECT  m.first_name||' '||m.last_name MEMBER, t.title,
        r.book_date, r.act_ret_date - r.book_date DURATION
FROM    member m, title t, rental r
WHERE   r.member_id = m.member_id
AND     r.title_id = t.title_id
ORDER BY member;

SET VERIFY ON
SET ECHO ON
```

VINOD BUSANAGA (b\_vinod87@yahoo.com) has a non-transferable  
license to use this Student Guide.