
SQL1

Student Guide • Volume 2



SQL1

Student Guide • Volume 2

41021GC13
Production 1.3
July 1999
M08914

ORACLE®

Authors

Neena Kochhar
Ellen Gravina
Priya Nathan

Technical Contributors and Reviewers

Claire Bennet
Christa Miethaner
Tony Hickman
Sherin Nassa
Nancy Greenberg
Hazel Russell
Kenneth Goetz
Piet van Zon
Ulrike Dietrich
Helen Robertson
Thomas Nguyen
Lisa Jansson
Kuljit Jassar

Publisher

Jerry Brosnan

Copyright © Oracle Corporation, 1998, 1999. All rights reserved.

This documentation contains proprietary information of Oracle Corporation. It is provided under a license agreement containing restrictions on use and disclosure and is also protected by copyright law. Reverse engineering of the software is prohibited. If this documentation is delivered to a U.S. Government Agency of the Department of Defense, then it is delivered with Restricted Rights and the following legend is applicable:

Restricted Rights Legend

Use, duplication or disclosure by the Government is subject to restrictions for commercial computer software and shall be deemed to be Restricted Rights software under Federal law, as set forth in subparagraph (c)(1)(ii) of DFARS 252.227-7013, Rights in Technical Data and Computer Software (October 1988).

This material or any portion of it may not be copied in any form or by any means without the express prior written permission of Oracle Corporation. Any other copying is a violation of copyright law and may result in civil and/or criminal penalties.

If this documentation is delivered to a U.S. Government Agency not within the Department of Defense, then it is delivered with "Restricted Rights," as defined in FAR 52.227-14, Rights in Data-General, including Alternate III (June 1987).

The information in this document is subject to change without notice. If you find any problems in the documentation, please report them in writing to Education Products, Oracle Corporation, 500 Oracle Parkway, Box SB-6, Redwood Shores, CA 94065. Oracle Corporation does not warrant that this document is error-free.

Oracle and all references to Oracle products are trademarks or registered trademarks of Oracle Corporation.

All other products or company names are used for identification purposes only, and may be trademarks of their respective owners.

Contents

Preface

Curriculum Map

Introduction

Objectives	I-2
System Development Life Cycle	I-3
Data Storage on Different Media	I-5
Relational Database Concept	I-6
Definition of a Relational Database	I-7
Data Models	I-8
Entity Relationship Model	I-9
Entity Relationship Modeling Conventions	I-10
Relational Database Terminology	I-12
Relating Multiple Tables	I-13
Relational Database Properties	I-14
Communicating with a RDBMS Using SQL	I-15
Relational Database Management System	I-16
Oracle8: Object Relational Database Management System	I-17
Oracle8i: Internet Platform Database for Internet Computing Features	I-18
Oracle Internet Platform	I-19
SQL Statements	I-20
Overview of Course Material	I-21
Tables Used in the Course	I-22
Summary	I-23

1 Writing Basic SQL Statements

Objectives	1-2
Capabilities of SQL SELECT Statements	1-3
Basic SELECT Statement	1-4
Writing SQL Statements	1-5
Selecting All Columns	1-6
Selecting Specific Columns	1-7
Column Heading Defaults	1-8
Arithmetic Expressions	1-9
Using Arithmetic Operators	1-10
Operator Precedence	1-11
Using Parentheses	1-13
Defining a Null Value	1-14

Null Values in Arithmetic Expressions	1-15
Defining a Column Alias	1-16
Using Column Aliases	1-17
Concatenation Operator	1-18
Using the Concatenation Operator	1-19
Literal Character Strings	1-20
Using Literal Character Strings	1-21
Duplicate Rows	1-22
Eliminating Duplicate Rows	1-23
SQL and SQL*Plus Interaction	1-24
SQL Statements Versus SQL*Plus Commands	1-25
Overview of SQL*Plus	1-26
Logging In to SQL*Plus	1-27
Displaying Table Structure	1-28
SQL*Plus Editing Commands	1-30
SQL*Plus File Commands	1-32
Summary	1-33
Practice Overview	1-34
2 Restricting and Sorting Data	
Objectives	2-2
Limiting Rows Using a Selection	2-3
Limiting Rows Selected	2-4
Using the WHERE Clause	2-5
Character Strings and Dates	2-6
Comparison Operators	2-7
Using the Comparison Operators	2-8
Other Comparison Operators	2-9
Using the BETWEEN Operator	2-10
Using the IN Operator	2-11
Using the LIKE Operator	2-12
Using the IS NULL Operator	2-14
Logical Operators	2-15
Using the AND Operator	2-16
Using the OR Operator	2-17
Using the NOT Operator	2-18
Rules of Precedence	2-19
ORDER BY Clause	2-22
Sorting in Descending Order	2-23
Sorting by Column Alias	2-24

Sorting by Multiple Columns	2-25
Summary	2-26
Practice Overview	2-27
3 Single-Row Functions	
Objectives	3-2
SQL Functions	3-3
Two Types of SQL Functions	3-4
Single-Row Functions	3-5
Character Functions	3-7
Case Conversion Functions	3-9
Using Case Conversion Functions	3-10
Character Manipulation Functions	3-11
Using the Character Manipulation Functions	3-12
Number Functions	3-13
Using the ROUND Function	3-14
Using the TRUNC Function	3-15
Using the MOD Function	3-16
Working with Dates	3-17
Arithmetic with Dates	3-18
Using Arithmetic Operators with Dates	3-19
Date Functions	3-20
Using Date Functions	3-21
Conversion Functions	3-23
Implicit Datatype Conversion	3-24
Explicit Datatype Conversion	3-26
TO_CHAR Function with Dates	3-29
Elements of Date Format Model	3-30
Using TO_CHAR Function with Dates	3-32
TO_CHAR Function with Numbers	3-33
Using TO_CHAR Function with Numbers	3-34
TO_NUMBER and TO_DATE Functions	3-35
RR Date Format	3-36
NVL Function	3-37
Using the NVL Function	3-38
DECODE Function	3-39
Using the DECODE Function	3-40
Nesting Functions	3-42
Summary	3-44
Practice Overview	3-45

4 Displaying Data from Multiple Tables

- Objectives 4-2
- Obtaining Data from Multiple Tables 4-3
- What Is a Join? 4-4
- Cartesian Product 4-5
- Generating a Cartesian Product 4-6
- Types of Joins 4-7
- What Is an Equijoin? 4-8
- Retrieving Records with Equijoins 4-9
- Qualifying Ambiguous Column Names 4-10
- Additional Search Conditions Using the AND Operator 4-11
- Using Table Aliases 4-12
- Joining More Than Two Tables 4-13
- Non-Equijoins 4-14
- Retrieving Records with Non-Equijoins 4-15
- Outer Joins 4-16
- Using Outer Joins 4-18
- Self Joins 4-19
- Joining a Table to Itself 4-20
- Summary 4-21
- Practice Overview 4-22

5 Aggregating Data Using Group Functions

- Objectives 5-2
- What Are Group Functions? 5-3
- Types of Group Functions 5-4
- Using Group Functions 5-5
- Using AVG and SUM Functions 5-6
- Using MIN and MAX Functions 5-7
- Using the COUNT Function 5-8
- Group Functions and Null Values 5-10
- Using the NVL Function with Group Functions 5-11
- Creating Groups of Data 5-12
- Creating Groups of Data: GROUP BY Clause 5-13
- Using the GROUP BY Clause 5-14
- Grouping by More Than One Column 5-16
- Using the GROUP BY Clause on Multiple Columns 5-17
- Illegal Queries Using Group Functions 5-18
- Excluding Group Results 5-20
- Excluding Group Results: HAVING Clause 5-21
- Using the HAVING Clause 5-22

Nesting Group Functions 5-24

Summary 5-25

Practice Overview 5-26

6 Subqueries

Objectives 6-2

Using a Subquery to Solve a Problem 6-3

Subqueries 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

HAVING Clause with Subqueries 6-11

What Is Wrong with This Statement? 6-12

Will This Statement Work? 6-13

Multiple-Row Subqueries 6-14

Using ANY Operator in Multiple-Row Subqueries 6-15

Using ALL Operator in Multiple-Row Subqueries 6-16

Summary 6-17

Practice Overview 6-18

7 Multiple-Column Subqueries

Objectives 7-2

Multiple-Column Subqueries 7-3

Using Multiple-Column Subqueries 7-4

Column Comparisons 7-6

Nonpairwise Comparison Subquery 7-7

Nonpairwise Subquery 7-8

Null Values in a Subquery 7-9

Using a Subquery in the FROM Clause 7-10

Summary 7-11

Practice Overview 7-12

8 Producing Readable Output with SQL*Plus

Objectives 8-2

Interactive Reports 8-3

Substitution Variables 8-4

Using the & Substitution Variable 8-5

Using the SET VERIFY Command 8-6

Character and Date Values with Substitution Variables 8-7

Specifying Column Names, Expressions, and Text at Runtime	8-8
Using the && Substitution Variable	8-10
Defining User Variables	8-11
The ACCEPT Command	8-12
Using the ACCEPT Command	8-13
DEFINE and UNDEFINE Commands	8-14
Using the DEFINE Command	8-15
Customizing the SQL*Plus Environment	8-16
SET Command Variables	8-17
Saving Customizations in the <code>login.sql</code> File	8-18
SQL*Plus Format Commands	8-19
The COLUMN Command	8-20
Using the COLUMN Command	8-21
COLUMN Format Models	8-22
Using the BREAK Command	8-23
Using the TTITLE and BTITLE Commands	8-24
Creating a Script File to Run a Report	8-25
Sample Report	8-27
Summary	8-28
Practice Overview	8-29

9 Manipulating Data

Objectives	9-2
Data Manipulation Language	9-3
Adding a New Row to a Table	9-4
The INSERT Statement	9-5
Inserting New Rows	9-6
Inserting Rows with Null Values	9-7
Inserting Special Values	9-8
Inserting Specific Date Values	9-9
Inserting Values by Using Substitution Variables	9-10
Creating a Script with Customized Prompts	9-11
Copying Rows from Another Table	9-12
Changing Data in a Table	9-13
The UPDATE Statement	9-14
Updating Rows in a Table	9-15
Updating with Multiple-Column Subquery	9-16
Updating Rows Based on Another Table	9-17
Updating Rows: Integrity Constraint Error	9-18
Removing a Row from a Table	9-19
The DELETE Statement	9-20

Deleting Rows from a Table	9-21
Deleting Rows Based on Another Table	9-22
Deleting Rows: Integrity Constraint Error	9-23
Database Transactions	9-24
Advantages of COMMIT and ROLLBACK Statements	9-26
Controlling Transactions	9-27
Implicit Transaction Processing	9-28
State of the Data Before COMMIT or ROLLBACK	9-29
State of the Data After COMMIT	9-30
Committing Data	9-31
State of the Data After ROLLBACK	9-32
Rolling Back Changes to a Marker	9-33
Statement-Level Rollback	9-34
Read Consistency	9-35
Implementation of Read Consistency	9-36
Locking	9-37
Summary	9-38
Practice Overview	9-39

10 Creating and Managing Tables

Objectives	10-2
Database Objects	10-3
Naming Conventions	10-4
The CREATE TABLE Statement	10-5
Referencing Another User's Tables	10-6
The DEFAULT Option	10-7
Creating Tables	10-8
Tables in the Oracle Database	10-9
Querying the Data Dictionary	10-10
Datatypes	10-11
Creating a Table by Using a Subquery	10-13
The ALTER TABLE Statement	10-15
Adding a Column	10-16
Modifying a Column	10-18
Dropping a Column	10-19
SET UNUSED Option	10-20
Dropping a Table	10-21
Changing the Name of an Object	10-22
Truncating a Table	10-23

Adding Comments to a Table	10-24)
Summary	10-25)
Practice Overview	10-26)
11 Including Constraints)
Objectives	11-2)
What Are Constraints?	11-3)
Constraint Guidelines	11-4)
Defining Constraints	11-5)
The NOT NULL Constraint	11-7)
The UNIQUE KEY Constraint	11-9)
The PRIMARY KEY Constraint	11-11)
The FOREIGN KEY Constraint	11-13)
FOREIGN KEY Constraint Keywords	11-15)
The CHECK Constraint	11-16)
Adding a Constraint	11-17)
Dropping a Constraint	11-19)
Disabling Constraints	11-20)
Enabling Constraints	11-21)
Cascading Constraints	11-22)
Viewing Constraints	11-24)
Viewing the Columns Associated with Constraints	11-25)
Summary	11-26)
Practice Overview	11-27)
12 Creating Views)
Objectives	12-2)
Database Objects	12-4)
What Is a View?	12-5)
Why Use Views?	12-6)
Simple Views and Complex Views	12-7)
Creating a View	12-8)
Retrieving Data from a View	12-11)
Querying a View	12-12)
Modifying a View	12-13)
Creating a Complex View	12-14)
Rules for Performing DML Operations on a View	12-15)
Using the WITH CHECK OPTION Clause	12-17)
Denying DML Operations	12-18)
Removing a View	12-19)
Inline Views	12-20)

"Top-N" Analysis	12-21
Performing "Top-N" Analysis	12-22
Example of "Top-N" Analysis	12-23
Summary	12-24
Practice Overview	12-26
13 Other Database Objects	
Objectives	13-2
Database Objects	13-3
What Is a Sequence?	13-4
The CREATE SEQUENCE Statement	13-5
Creating a Sequence	13-6
Confirming Sequences	13-7
NEXTVAL and CURRVAL Pseudocolumns	13-8
Using a Sequence	13-10
Modifying a Sequence	13-12
Guidelines for Modifying a Sequence	13-13
Removing a Sequence	13-14
What Is an Index?	13-15
How Are Indexes Created?	13-16
Creating an Index	13-17
When to Create an Index	13-18
When Not to Create an Index	13-19
Confirming Indexes	13-20
Function-Based Indexes	13-21
Removing an Index	13-22
Synonyms	13-23
Creating and Removing Synonyms	13-24
Summary	13-25
Practice Overview	13-26
14 Controlling User Access	
Objectives	14-2
Controlling User Access	14-3
Privileges	14-4
System Privileges	14-5
Creating Users	14-6
User System Privileges	14-7
Granting System Privileges	14-8
What Is a Role?	14-9
Creating and Granting Privileges to a Role	14-10

- Changing Your Password 14-11
- Object Privileges 14-12
- Granting Object Privileges 14-14
- Using WITH GRANT OPTION and PUBLIC Keywords 14-15
- Confirming Privileges Granted 14-16
- How to Revoke Object Privileges 14-17
- Revoking Object Privileges 14-18
- Summary 14-19
- Practice Overview 14-20

15 SQL Workshop

- Workshop Overview 15-2
- Practice 15 15-3

A Practice Solutions

B Table Descriptions and Data

Index

15

SQL Workshop

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

Workshop Overview

- **Creating tables and sequences**
- **Modifying data in the tables**
- **Modifying a table definition**
- **Creating a view**
- **Writing scripts containing SQL and SQL*Plus commands**
- **Generating a simple report**

15-2

Copyright © Oracle Corporation, 1999. All rights reserved.

ORACLE™

Workshop Overview

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

Note: If you want to build the tables, you can execute the `buildtab.sql` script in SQL*Plus. If you want to drop the tables, you can execute the `dropvid.sql` script in SQL*Plus. Then you can execute the `buildvid.sql` script in SQL*Plus to create and populate the tables. If you use the `buildvid.sql` script to build and populate the tables, start with Practice #6b.

Practice 15

1. Create the tables based on the following table instance charts. Choose the appropriate datatypes and be sure to 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
Datatype	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	
Datatype	Number	VARCHAR2	VARCHAR2	VARCHAR2	VARCHAR2	Date
Length	10	60	400	4	20	

Practice 15 (continued)

c. Table name: TITLE_COPY

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

d. Table name: RENTAL

Column Name	BOOK_DATE	MEMBER_ID	COPY_ID	ACT_RET_DATE	EXP_RET_DATE	TITLE_ID
Key Type	PK	PK,FK1	PK,FK2			PK,FK2
Default Value	System Date				System Date + 2 days	
FK Ref Table		MEMBER	TITLE_COPY			TITLE_COPY
FK Ref Col		MEMBER_ID	COPY_ID			TITLE_ID
Datatype	Date	Number	Number	Date	Date	Number
Length		10	10			10

Practice 15 (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
Datatype	Date	Number	Number
Length		10	10

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

```
TABLE_NAME
-----
MEMBER
RENTAL
RESERVATION
TITLE
TITLE_COPY
```

```
CONSTRAINT_NAME          C TABLE_NAME
-----
MEMBER_LAST_NAME_NN      C MEMBER
MEMBER_JOIN_DATE_NN      C MEMBER
MEMBER_MEMBER_ID_PK      P MEMBER
RENTAL_BOOK_DATE_COPY_TITLE_PK P RENTAL
RENTAL_MEMBER_ID_FK      R RENTAL
RENTAL_COPY_ID_TITLE_ID_FK R RENTAL
RESERVATION_RESDATE_MEM_TIT_PK P RESERVATION
RESERVATION_MEMBER_ID    R RESERVATION
RESERVATION_TITLE_ID     R RESERVATION
...
17 rows selected.
```

Practice 15 (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 the values. Name the sequence member_id_seq.
 - b. Title number for the TITLE table: start with 92; no caching. Name the sequence title_id_seq.
 - c. Verify the existence of the sequences in the data dictionary.

```
SEQUENCE_NAME  INCREMENT_BY  LAST_NUMBER
-----
TITLE_ID_SEQ           1              92
MEMBER_ID_SEQ         1             101
```

4. Add data to the tables. Create a script for each set of data to add.
 - a. Add movie titles to the TITLE table. Write a script to enter the movie information. Save the script as p15q4a.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
-----
Willie and Christmas Too
Alien Again
The Glob
My Day Off
Miracles on Ice
Soda Gang
6 rows selected.
```

Practice 15 (continued)

Title	Description	Rating	Category	Release_date
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	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

Practice 15 (continued)

- b. Add data to the MEMBER table. Write a script named p15q4b.sql to prompt users for the information. Execute the script. Be sure to use the sequence to add the member numbers.

First_Name	Last_Name	Address	City	Phone	Join_Date
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

Practice 15 (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
Willie and Christmas Too	1	AVAILABLE
Alien Again	1	AVAILABLE
	2	RENTED
The Glob	1	AVAILABLE
My Day Off	1	AVAILABLE
	2	AVAILABLE
	3	RENTED
Miracles on Ice	1	AVAILABLE
Soda Gang	1	AVAILABLE

d. Add the following rentals to the RENTAL table:

Note: Title number may be different depending on sequence number.

Title_Id	Copy_Id	Member_Id	Book_date	Exp_Ret_Date	Act_Ret_Date
92	1	101	3 days ago	1 day ago	2 days 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	2 days ago

Practice 15 (continued)

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

TITLE	COPY_ID	STATUS	EXP_RET_D
-----	-----	-----	-----
Alien Again	1	AVAILABLE	
Alien Again	2	RENTED	05-NOV-97
Miracles on Ice	1	AVAILABLE	
My Day Off	1	AVAILABLE	
My Day Off	2	AVAILABLE	
My Day Off	3	RENTED	06-NOV-97
Soda Gang	1	AVAILABLE	04-NOV-97
The Glob	1	AVAILABLE	
Willie and Christmas Too	1	AVAILABLE	05-NOV-97

9 rows selected.

6. Make changes to data in the tables.
- Add a new title. The movie is "Interstellar Wars," which is rated PG and classified as a sci-fi 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."

Practice 15 (continued)

- c. Customer Carmen Velasquez rents the movie "Interstellar Wars," copy 1. Remove her reservation for the movie. Record the information about the rental. Allow the default value for the expected return date to be used. Verify that the rental was recorded by using the view you created.

TITLE	COPY_ID	STATUS	EXP_RET_D
-----	-----	-----	-----
Alien Again	1	AVAILABLE	
Alien Again	2	RENTED	05-NOV-97
Interstellar Wars	1	RENTED	08-NOV-97
Interstellar Wars	2	AVAILABLE	
Miracles on Ice	1	AVAILABLE	
My Day Off	1	AVAILABLE	
My Day Off	2	AVAILABLE	
My Day Off	3	RENTED	06-NOV-97
Soda Gang	1	AVAILABLE	04-NOV-97
The Glob	1	AVAILABLE	
Willie and Christmas Too	1	AVAILABLE	05-NOV-97

11 rows selected.

7. Make a modification to one of the tables.

- a. Add a PRICE column to the TITLE table to record the purchase price of the video. The column should have a total length of eight digits and two decimal places. 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)

Practice 15 (continued)

- b. Create a script named p15q7b.sql to update each video with a price according to the following list.

Note: Have the title_id numbers available for this exercise.

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

- c. Ensure that in the future all titles will contain a price value. Verify the constraint.

```

CONSTRAINT_NAME      C  SEARCH_CONDITION
-----
TITLE_PRICE_NN       C  PRICE IS NOT NULL
    
```

8. Create a report titled Customer History Report. This report will contain 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 script in a file named p15q8.sql.

```

MEMBER          TITLE                                BOOK_DATE  DURATION
-----
Carmen Velasquez Willie and Christmas Too 03-NOV-97    1
                  Alien Again          09-AUG-98
                  Interstellar Wars    10-AUG-98

LaDoris Ngao    My Day Off                    08-AUG-98

Molly Urguhart  Soda Gang                      06-AUG-98    2
    
```

A

.....

Practice Solutions

Practice 1 Solutions

1. Initiate a SQL*Plus session using the user ID and password provided by the instructor.
2. SQL*Plus commands access the database.

False

3. Will the SELECT statement executes successfully?

True

```
SQL> SELECT   ename, job, sal Salary
  2 FROM      emp;
```

4. Will the SELECT statement executes successfully?

True

```
SQL> SELECT *
  2 FROM   salgrade;
```

5. There are four coding errors in this statement. Can you identify them?

```
SQL> SELECT      empno, ename
  2              salary x 12 ANNUAL SALARY
  3 FROM         emp;
```

- The EMP table does not contain a column called salary. The column is called sal.
- 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 be enclosed in double quotation marks.
- A comma is missing after the column, ENAME.

6. Show the structure of the DEPT table. Select all data from the DEPT table.

```
SQL> DESCRIBE dept
SQL> SELECT *
  2 FROM   dept;
```

7. Show the structure of the EMP table. Create a query to display the name, job, hire date, and employee number for each employee, with employee number appearing first. Save your SQL statement to a file named plq7.sql.

```
SQL> DESCRIBE emp
SQL> SELECT empno, ename, job, hiredate
  2 FROM   emp;
SQL> SAVE plq7.sql
Created file plq7.sql
```

Practice 1 Solutions (continued)

8. Run your query in the p1q7.sql file.

```
SQL> START p1q7.sql
```

9. Create a query to display unique jobs from the EMP table.

```
SQL> SELECT DISTINCT job
2 FROM emp;
```

If you have time, complete the following exercises:

10. Load p1q7.sql into the SQL buffer. Name the column headings Emp #, Employee, Job, and Hire Date, respectively. Rerun your query.

```
SQL> GET p1q7.sql
1 SELECT empno, ename, job, hiredate
2* FROM emp
SQL> 1 SELECT empno "Emp #", ename "Employee",
SQL> i
2i job "Job", hiredate "Hire Date"
3i
SQL> SAVE p1q7.sql REPLACE
Created file p1q7.sql
SQL> START p1q7.sql
```

11. Display the name concatenated with the job, separated by a comma and space, and name the column Employee and Title.

```
SQL> SELECT ename||', '||job "Employee and Title"
2 FROM emp;
```

Practice 1 Solutions (continued)

If you want extra challenge, complete the following exercise:

12. Create a query to display all the data from the EMP table. Separate each column by a comma. Name the column THE_OUTPUT.

```
SQL> SELECT empno || ',' || ename || ',' || job || ',' ||  
2          mgr || ',' || hiredate || ',' || sal || ',' ||  
3          comm || ',' || deptno THE_OUTPUT  
4 FROM    emp;
```

Practice 2 Solutions

1. Create a query to display the name and salary of employees earning more than \$2850. Save your SQL statement to a file named p2q1.sql. Run your query.

```
SQL> SELECT   ename, sal
  2 FROM      emp
  3 WHERE     sal > 2850;
```

```
SQL> SAVE p2q1.sql
Created file p2q1.sql
```

2. Create a query to display the employee name and department number for employee number 7566.

```
SQL> SELECT   ename, deptno
  2 FROM      emp
  3 WHERE     empno = 7566;
```

3. Modify p2q1.sql to display the name and salary for all employees whose salary is not in the range of \$1500 and \$2850. Resave your SQL statement to a file named p2q3.sql. Rerun your query.

```
SQL> EDIT p2q1.sql

SELECT   ename, sal
FROM     emp
WHERE    sal NOT BETWEEN 1500 AND 2850
/

SQL> START p2q3.sql
```

4. Display the employee name, job, and start date of employees hired between February 20, 1981, and May 1, 1981. Order the query in ascending order by start date.

```
SQL> SELECT   ename, job, hiredate
  2 FROM      emp
  3 WHERE     hiredate BETWEEN
  4 TO_DATE('20-Feb-1981', 'DD-MON-YYYY') AND
  5 TO_DATE('01-May-1981', 'DD-MON-YYYY')
  6 ORDER BY  hiredate;
```

Practice 2 Solutions (continued)

5. Display the employee name and department number of all employees in departments 10 and 30 in alphabetical order by name.

```
SQL> SELECT      ename, deptno
 2 FROM          emp
 3 WHERE         deptno IN (10, 30)
 4 ORDER BY     ename;
```

6. Modify p2q3.sql to list the name and salary of employees who earn more than \$1500 and are in department 10 or 30. Label the column Employee and Monthly Salary, respectively. Resave your SQL statement to a file named p2q6.sql. Rerun your query.

```
SQL> EDIT p2q3.sql
      SELECT      ename "Employee", sal "Monthly Salary"
      FROM        emp
      WHERE       sal > 1500
      AND         deptno IN (10, 30)
      /
SQL> START p2q6.sql
```

7. Display the name and hire date of every employee who was hired in 1982.

```
SQL> SELECT      ename, hiredate
 2 FROM          emp
 3 WHERE         hiredate LIKE '%82';
```

8. Display the name and title of all employees who do not have a manager.

```
SQL> SELECT      ename, job
 2 FROM          emp
 3 WHERE         mgr IS NULL;
```

9. Display the name, salary, and commission for all employees who earn commissions. Sort data in descending order of salary and commissions.

```
SQL> SELECT      ename, sal, comm
 2 FROM          emp
 3 WHERE         comm IS NOT NULL
 4 ORDER BY     sal DESC, comm DESC;
```

Practice 2 Solutions (continued)

If you have time, complete the following exercises.

10. Display the names of all employees where the third letter of their name is an *A*.
Note: There are two underscores () before the *A* in the WHERE clause.

```
SQL> SELECT   ename
2 FROM       emp
3 WHERE      ename LIKE ' __A% ';
```

11. Display the names of all employees that have two *Ls* in their name and are in department 30 or their manager is 7782.

```
SQL> SELECT   ename
2 FROM       emp
3 WHERE      ename LIKE '%L%L%'
4 AND       deptno = 30
5 OR        mgr = 7782;
```

If you want extra challenge, complete the following exercises.

12. Display the name, job, and salary for all employees whose job is Clerk or Analyst and their salary is not equal to \$1000, \$3000, or \$5000.

```
SQL> SELECT   ename, job, sal
2 FROM       emp
3 WHERE      job IN ('CLERK', 'ANALYST')
4 AND       sal NOT IN (1000, 3000, 5000);
```

13. Modify p2q6.sql to display the name, salary, and commission for all employees whose commission amount is greater than their salary increased by 10%. Rerun your query.
Resave your query as p2q13.sql.

```
SQL> EDIT p2q6.sql
      SELECT   ename "Employee", sal "Monthly Salary", comm
      FROM     emp
      WHERE    comm > sal * 1.1
      /
SQL> START p2q13.sql
```

Practice 3 Solutions

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

```
SQL> SELECT sysdate "Date"  
2 FROM dual;
```

2. Display the employee number, name, salary, and salary increase by 15% expressed as a whole number. Label the column New Salary. Save your SQL statement to a file named p3q2.sql.

```
SQL> SELECT empno, ename, sal,  
2 ROUND(sal * 1.15, 0) "New Salary"  
3 FROM emp;
```

```
SQL> SAVE p3q2.sql  
Created file p3q2.sql
```

3. Run your query in the file p3q2.sql.

```
SQL> START p3q2.sql
```

4. Modify your query p3q2.sql to add a column that will subtract the old salary from the new salary. Label the column Increase. Rerun your query.

```
SQL> EDIT p3q2.sql  
SELECT empno, ename, sal,  
        ROUND(sal * 1.15, 0) "New Salary",  
        ROUND(sal * 1.15, 0) - sal "Increase"  
FROM emp  
/  
SQL> START p3q2.sql
```

5. Display the employee's 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 the format similar to "Sunday, the Seventh of September, 1981".

```
SQL> SELECT ename, hiredate,  
2 TO_CHAR(NEXT_DAY(ADD_MONTHS(hiredate, 6),  
3 'MONDAY'),  
4 'fmDay, "the" Ddspth "of" Month, YYYY') REVIEW  
5 FROM emp;
```

Practice 3 Solutions (continued)

- For each employee display the employee name and calculate the number of months between today and the date the employee was hired. Label the column MONTHS_WORKED. Order your results by the number of months employed. Round the number of months up to the closest whole number.

```
SQL> SELECT      ename, ROUND(MONTHS_BETWEEN
2              (SYSDATE, hiredate)) MONTHS_WORKED
3 FROM          emp
4 ORDER BY     MONTHS_BETWEEN(SYSDATE, hiredate);
```

- Write a query that produces the following for each employee:
<employee name> earns <salary> monthly but wants <3 times salary>. Label the column Dream Salaries.

```
SQL> SELECT  ename || ' earns '
2           || TO_CHAR(sal, 'fm$99,999.00')
3           || ' monthly but wants '
4           || TO_CHAR(sal * 3, 'fm$99,999.00')
5           || '.' "Dream Salaries"
6 FROM      emp;
```

If you have time, complete the following exercises:

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

```
SQL> SELECT  ename,
2           LPAD(sal, 15, '$') SALARY
3 FROM      emp;
```

- Write a query that will display the employee's name with the first letter capitalized and all other letters lowercase and the length of their name, for all employees whose name starts with J, A, or M. Give each column an appropriate label.

```
SQL> SELECT  INITCAP(ename) "Name",
2           LENGTH(ename) "Length"
3 FROM      emp
4 WHERE     ename LIKE 'J%'
5 OR       ename LIKE 'M%'
6 OR       ename LIKE 'A%';
```

Practice 3 Solutions (continued)

10. Display the name, hire date, and day of the week on which the employee started. Label the column DAY. Order the results by the day of the week starting with Monday.

```
SQL> SELECT   ename, hiredate,
2            TO_CHAR(hiredate, 'DAY') DAY
3 FROM       emp
4 ORDER BY  TO_CHAR(hiredate - 1, 'd');
```

If you want extra challenge, complete the following exercises:

11. Create a query that will display the employee name and commission amount. If the employee does not earn commission, put "No Commission". Label the column COMM.

```
SQL> SELECT   ename,
2            NVL(TO_CHAR(comm), 'No Commission') COMM
3 FROM       emp;
```

12. Create a query that displays the employees' names and indicates the amount of their salaries through asterisks. Each asterisk signifies a hundred dollars. Sort the data in descending order of salary. Label the column EMPLOYEE_AND_THEIR_SALARIES.

```
SQL> SELECT   rpad(ename, 8) || ' ' || rpad(' ', sal/100+1, '*')
2            EMPLOYEE_AND_THEIR_SALARIES
3 FROM       emp
4 ORDER BY  sal DESC;
```

13. Write a query that displays the grade of all employees based on the value of the column JOB, as per the table shown below

<i>JOB</i>	<i>GRADE</i>
PRESIDENT	A
MANAGER	B
ANALYST	C
SALESMAN	D
CLERK	E
None of the above	O

```
SQL> SELECT job, decode (job, 'CLERK',      'E',
2                        'SALESMAN',      'D',
3                        'ANALYST',      'C',
4                        'MANAGER',      'B',
5                        'PRESIDENT',    'A',
6                        '0') GRADE
7 FROM emp;
```

Practice 4 Solutions

1. Write a query to display the name, department number, and department name for all employees.

```
SQL> SELECT   e.ename, e.deptno, d.dname
  2 FROM      emp e, dept d
  3 WHERE     e.deptno = d.deptno;
```

2. Create a unique listing of all jobs that are in department 30. Include the location of department 30 in the output.

```
SQL> SELECT   DISTINCT e.job, d.loc
  2 FROM      emp e, dept d
  3 WHERE     e.deptno = d.deptno
  4 AND      e.deptno = 30;
```

3. Write a query to display the employee name, department name, and location of all employees who earn a commission.

```
SQL> SELECT   e.ename, d.dname, d.loc
  2 FROM      emp e, dept d
  3 WHERE     e.deptno = d.deptno
  4 AND      e.comm IS NOT NULL;
```

4. Display the employee name and department name for all employees who have an *A* in their name. Save your SQL statement in a file called p4q4.sql.

```
SQL> SELECT   e.ename, d.dname
  2 FROM      emp e, dept d
  3 WHERE     e.deptno = d.deptno
  4 AND      e.ename LIKE '%A%';
```

5. Write a query to display the name, job, department number, and department name for all employees who work in DALLAS.

```
SQL> SELECT   e.ename, e.job, e.deptno, d.dname
  2 FROM      emp e, dept d
  3 WHERE     e.deptno = d.deptno
  4 AND      d.loc = 'DALLAS';
```

Practice 4 Solutions (continued)

6. Display the employee name and employee number along with their manager's name and manager number. Label the columns Employee, Emp#, Manager, and Mgr#, respectively. Save your SQL statement in a file called p4q6.sql.

```
SQL> SELECT   e.ename "Employee", e.empno "Emp#",
2            m.ename "Manager", m.empno "Mgr#"
3            FROM     emp e, emp m
4            WHERE    e.mgr = m.empno;
SQL> SAVE p4q6.sql
Created file p4q6.sql
```

7. Modify p4q6.sql to display all employees including King, who has no manager. Resave as p4q7.sql. Run p4q7.sql.

```
SQL> EDIT p4q6.sql
          SELECT e.ename "Employee", e.empno "Emp#",
          m.ename "Manager", m.empno "Mgr#"
          FROM   emp e, emp m
          WHERE  e.mgr = m.empno(+)
          /
SQL> START p4q7.sql
```

If you have time, complete the following exercises.

8. Create a query that will display the employee name, department number, and all the employees that work in the same department as a given employee. Give each column an appropriate label.

```
SQL> SELECT   e.deptno department, e.ename employee,
2            c.ename colleague
3            FROM   emp e, emp c
4            WHERE  e.deptno = c.deptno
5            AND    e.empno <> c.empno
6            ORDER BY e.deptno, e.ename, c.ename;
```

Practice 4 Solutions (continued)

9. Show the structure of the SALGRADE table. Create a query that will display the name, job, department name, salary, and grade for all employees.

```
SQL> DESCRIBE salgrade
SQL> SELECT e.ename, e.job, d.dname, e.sal, s.grade
2 FROM emp e, dept d, salgrade s
3 WHERE e.deptno = d.deptno
4 AND e.sal BETWEEN s.losal AND s.hisal;
```

If you want extra challenge, complete the following exercises:

10. Create a query to display the name and hire date of any employee hired after employee Blake.

```
SQL> SELECT emp.ename, emp.hiredate
2 FROM emp, emp blake
3 WHERE blake.ename = 'BLAKE'
4 AND blake.hiredate < emp.hiredate;
```

11. Display all employees' names and hire dates along with their manager's name and hire date for all employees who were hired before their managers. Label the columns Employee, Emp Hiredate, Manager, and Mgr Hiredate, respectively.

```
SQL> SELECT e.ename "Employee", e.hiredate "Emp Hiredate",
2 m.ename "Manager", m.hiredate "Mgr Hiredate"
3 FROM emp e, emp m
4 WHERE e.mgr = m.empno
5 AND e.hiredate < m.hiredate;
```

Practice 5 Solutions

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

1. Group functions work across many rows to produce one result.
True
2. Group functions include nulls in calculations.
False. Group functions ignore null values. If you want to include null values, use the NVL function.
3. The WHERE clause restricts rows prior to inclusion in a group calculation.
True
4. Display 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 in a file called p5q4.sql.

```
SQL> SELECT    ROUND(MAX(sal),0) "Maximum" ,
2             ROUND(MIN(sal),0) "Minimum" ,
3             ROUND(SUM(sal),0) "Sum" ,
4             ROUND(AVG(sal),0) "Average"
5 FROM        emp;
```

```
SQL> SAVE p5q4.sql
Created file p5q4.sql
```

5. Modify p5q4.sql to display the minimum, maximum, sum, and average salary for each job type. Resave to a file called p5q5.sql. Rerun your query.

```
SQL> EDIT p5q6.sql
SELECT    job, ROUND(MAX(sal),0) "Maximum" ,
          ROUND(MIN(sal),0) "Minimum" ,
          ROUND(SUM(sal),0) "Sum" ,
          ROUND(AVG(sal),0) "Average"
FROM      emp
GROUP BY job
/
SQL> START p5q5.sql
```

Practice 5 Solutions (continued)

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

```
SQL> SELECT   job, COUNT(*)
2  FROM      emp
3  GROUP BY  job;
```

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

```
SQL> SELECT   COUNT(DISTINCT mgr) "Number of Managers"
2  FROM      emp;
```

8. Write a query that will display the difference between the highest and lowest salaries. Label the column DIFFERENCE.

```
SQL> SELECT   MAX(sal) - MIN(sal) DIFFERENCE
2  FROM      emp;
```

If you have time, complete the following exercises.

9. 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 less than \$1000. Sort the output in descending order of salary.

```
SQL> SELECT   mgr, MIN(sal)
2  FROM      emp
3  WHERE     mgr IS NOT NULL
4  GROUP BY  mgr
5  HAVING    MIN(sal) > 1000
6  ORDER BY  MIN(sal) DESC;
```

10. Write a query to display the department name, location name, number of employees, and the average salary for all employees in that department. Label the columns dname, loc, Number of People, and Salary, respectively. Round the average salary to two decimal places.

```
SQL> SELECT   d.dname, d.loc, COUNT(*) "Number of People",
2           ROUND(AVG(sal),2) "Salary"
3  FROM      emp e, dept d
4  WHERE     e.deptno = d.deptno
5  GROUP BY  d.dname, d.loc;
```

Practice 5 Solutions (continued)

If you want extra challenge, complete the following exercises:

11. Create a query that will display the total number of employees and of that total the number who were hired in 1980, 1981, 1982, and 1983. Give appropriate column headings.

```
SQL> SELECT COUNT(*) total,  
2          SUM(DECODE(TO_CHAR(hiredate, 'YYYY'),  
3                    1980,1,0)) "1980",  
4          SUM(DECODE(TO_CHAR(hiredate, 'YYYY'),  
5                    1981,1,0)) "1981",  
6          SUM(DECODE(TO_CHAR(hiredate, 'YYYY'),  
7                    1982,1,0)) "1982",  
8          SUM(DECODE(TO_CHAR(hiredate, 'YYYY'),  
9                    1983,1,0)) "1983"  
10 FROM    emp;
```

12. Create a matrix query to display the job, the salary for that job based on department number, and the total salary for that job for all departments, giving each column an appropriate heading.

```
SQL> SELECT      job "Job",  
2              SUM(DECODE(deptno, 10, sal)) "Dept 10",  
3              SUM(DECODE(deptno, 20, sal)) "Dept 20",  
4              SUM(DECODE(deptno, 30, sal)) "Dept 30",  
5              SUM(sal) "Total"  
6 FROM          emp  
7 GROUP BY     job;
```

Practice 6 Solutions

1. Write a query to display the employee name and hire date for all employees in the same department as Blake. Exclude Blake.

```
SQL> SELECT   ename, hiredate
  2 FROM      emp
  3 WHERE     deptno = (SELECT   deptno
  4                               FROM      emp
  5                               WHERE     ename = 'BLAKE')
  6 AND       ename != 'BLAKE';
```

2. Create a query to display the employee number and name for all employees who earn more than the average salary. Sort the results in descending order of salary.

```
SQL> SELECT   empno, ename
  2 FROM      emp
  3 WHERE     sal > (SELECT AVG(sal)
  4                               FROM emp)
  5 ORDER BY  sal DESC;
```

3. Write a query that will display the employee number and name for all employees who work in a department with any employee whose name contains a *T*. Save your SQL statement in a file called p6q3.sql.

```
SQL> SELECT   empno, ename
  2 FROM      emp
  3 WHERE     deptno IN (SELECT   deptno
  4                               FROM      emp
  5                               WHERE     ename LIKE '%T%');
SQL> SAVE p6q3.sql
Created file p6q3.sql
```

4. Display the employee name, department number, and job title for all employees whose department location is Dallas.

```
SQL> SELECT   ename, deptno, job
  2 FROM      emp
  3 WHERE     deptno IN (SELECT   deptno
  4                               FROM      dept
  5                               WHERE     loc = 'DALLAS');
```

Practice 6 Solutions (continued)

5. Display the employee name and salary of all employees who report to King.

```
SQL> SELECT ename, sal
 2 FROM emp
 3 WHERE mgr = (SELECT empno
 4             FROM emp
 5             WHERE ename = 'KING');
```

6. Display the department number, name, and job for all employees in the Sales department.

```
SQL> SELECT deptno, ename, job
 2 FROM emp
 3 WHERE deptno IN (SELECT deptno
 4                 FROM dept
 5                 WHERE dname = 'SALES');
```

If you have time, complete the following exercise:

7. Modify p6q3.sql to display the employee number, name, and salary for all employees who earn more than the average salary and who work in a department with any employee with a *T* in their name. Resave as p6q7.sql. Rerun your query.

```
SQL> EDIT p6q3.sql
SELECT empno, ename, sal
  FROM emp
  WHERE sal > (SELECT AVG(sal)
              FROM emp)
            AND deptno IN (SELECT deptno
                          FROM emp
                          WHERE ename LIKE '%T%')
/
SQL> START p6q7.sql
```

Practice 7 Solutions

1. Write a query to display the name, department number, and salary of any employee whose department number and salary match the department number and salary of any employee who earns a commission.

```
SQL> SELECT  ename, deptno, sal
2 FROM      emp
3 WHERE      (sal, deptno) IN
4             (SELECT  sal, deptno
5                  FROM    emp
6                  WHERE   comm IS NOT NULL);
```

2. Display the name, department name, and salary of any employee whose salary and commission match the salary and commission of any employee located in Dallas.

```
SQL> SELECT  ename, dname, sal
2 FROM      emp e, dept d
3 WHERE      e.deptno = d.deptno
4 AND        (sal, NVL(comm,0)) IN
5             (SELECT  sal, NVL(comm,0)
6                  FROM    emp e, dept d
7                  WHERE   e.deptno = d.deptno
8                  AND     d.loc = 'DALLAS');
```

3. Create a query to display the name, hire date, and salary for all employees who have the same salary and commission as Scott.

```
SQL> SELECT  ename, hiredate, sal
2 FROM      emp
3 WHERE      (sal, NVL(comm,0)) IN
4             (SELECT  sal, NVL(comm,0)
5                  FROM    emp
6                  WHERE   ename = 'SCOTT')
7 AND        ename != 'SCOTT';
```

4. Create a query to display the employees that earn a salary that is higher than the salary of all of the clerks. Sort the results on salary from highest to lowest.

```
SQL> SELECT  ename, job, sal
2 FROM      emp
3 WHERE      sal > ALL (SELECT sal
4                  FROM    emp
5                  WHERE   job = 'CLERK')
6 ORDER BY  sal DESC;
```

Practice 8 Solutions

Determine whether the following statements are true or false:

1. A single ampersand substitution variable prompts at most once.

False

However, if the variable is defined, then the single ampersand substitution variable will not prompt at all. In fact, it will pick up the value in the predefined variable.

2. The ACCEPT command is a SQL command.

False

The ACCEPT command is a SQL*Plus command. It is issued at the SQL prompt.

3. Write a script file to display the employee name, job, and hire date for all employees who started between a given range. Concatenate the name and job together, separated by a space and comma, and label the column Employees. Prompt the user for the two ranges using the ACCEPT command. Use the format MM/DD/YYYY. Save the script file as p8q3.sql.

```
SET ECHO OFF
SET VERIFY OFF
ACCEPT low_date DATE FORMAT 'MM/DD/YYYY' -
PROMPT 'Please enter the low date range ('MM/DD/YYYY'):'
ACCEPT high_date DATE FORMAT 'MM/DD/YYYY' -
PROMPT 'Please enter the high date range ('MM/DD/YYYY'):'
COLUMN EMPLOYEES FORMAT A25
SELECT      ename ||', '|| job EMPLOYEES, hiredate
FROM        emp
WHERE       hiredate BETWEEN
            TO_DATE('&low_date', 'MM/DD/YYYY')
            AND TO_DATE('&high_date', 'MM/DD/YYYY')
/
UNDEFINE low_date
UNDEFINE high_date
COLUMN EMPLOYEES CLEAR
SET VERIFY ON
SET ECHO ON
SQL> START p8q3.sql;
```

Practice 8 Solutions (continued)

4. Write a script to display the employee name, job, and department name for a given location. The search condition should allow for case-insensitive searches of the department location. Save the script file as p8q4.sql.

```
SET ECHO OFF
SET VERIFY OFF
ACCEPT p_location PROMPT 'Please enter the location name: '
COLUMN ename HEADING "EMPLOYEE NAME" FORMAT A15
COLUMN dname HEADING "DEPARTMENT NAME" FORMAT A15
SELECT e.ename, e.job, d.dname
FROM emp e, dept d
WHERE e.deptno = d.deptno
AND LOWER(d.loc) LIKE LOWER('%&p_location%')
/
UNDEFINE p_location
COLUMN ename CLEAR
COLUMN dname CLEAR
SET VERIFY ON
SET ECHO ON
SQL> START p8q4.sql
```

Practice 8 Solutions (continued)

5. Modify p8q4.sql to create a report containing the department name, employee name, hire date, salary, and each employees' annual salary for all employees in a given location. Prompt the user for the location. Label the columns DEPARTMENT NAME, EMPLOYEE NAME, START DATE, SALARY and ANNUAL SALARY, placing the labels on multiple lines. Resave the script as p8q5.sql.

```
SET ECHO OFF
SET FEEDBACK OFF
SET VERIFY OFF
BREAK ON  dname
ACCEPT p_location PROMPT 'Please enter the location name:
COLUMN dname HEADING "DEPARTMENT|NAME" FORMAT A15
COLUMN ename HEADING "EMPLOYEE|NAME" FORMAT A15
COLUMN hiredate HEADING "START|DATE" FORMAT A15
COLUMN sal HEADING "SALARY" FORMAT $99,990.00
COLUMN asal HEADING "ANNUAL|SALARY" FORMAT $99,990.00
SELECT      d.dname, e.ename, e.hiredate,
            e.sal,   e.sal * 12 asal
FROM        emp e,  dept d
WHERE       e.deptno = d.deptno
AND         LOWER(d.loc) LIKE LOWER('%&p_location%')
ORDER BY   dname
/
UNDEFINE p_location
COLUMN dname CLEAR
COLUMN ename CLEAR
COLUMN hiredate CLEAR
COLUMN sal CLEAR
COLUMN asal CLEAR
CLEAR BREAK
SET VERIFY ON
SET FEEDBACK ON
SET ECHO ON
SQL> START p8q5.sql
```

Practice 9 Solutions

Insert data into the MY_EMPLOYEE table.

1. Run the lab9_1.sql script to build the MY_EMPLOYEE table that will be used for the lab.

```
SQL> START lab9_1.sql
```

2. Describe the structure of the MY_EMPLOYEE table to identify the column names.

```
SQL> DESCRIBE my_employee
```

3. 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.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	795
2	Dancs	Betty	bdancs	860
3	Biri	Ben	bbiri	1100
4	Newman	Chad	cnewman	750
5	Ropeburn	Audry	aropebur	1550

```
SQL> INSERT INTO my_employee  
2 VALUES (1, 'Patel', 'Ralph', 'rpatel', 795);
```

4. Populate the MY_EMPLOYEE table with the second row of sample data from the preceding list. This time, list the columns explicitly in the INSERT clause.

```
SQL> INSERT INTO my_employee (id, last_name, first_name,  
2                          userid, salary)  
3 VALUES (2, 'Dancs', 'Betty', 'bdancs', 860);
```

5. Confirm your addition to the table.

```
SQL> SELECT *  
2 FROM my_employee;
```

Practice 9 Solutions (continued)

6. Create a script named `loademp.sql` to load rows into the `MY_EMPLOYEE` table interactively. Prompt the user for the employee's id, first name, last name, and salary. Concatenate the first letter of the first name and the first seven characters of the last name to produce the `userid`.

```
SET ECHO OFF
SET VERIFY OFF
ACCEPT p_id -
PROMPT 'Please enter the employee number: '
ACCEPT p_first_name -
PROMPT 'Please enter the employee's first name: '
ACCEPT p_last_name -
PROMPT 'Please enter the employee's last name: '
ACCEPT p_salary PROMPT 'Please enter the employee's salary: '
INSERT INTO my_employee
VALUES      (&p_id, '&p_last_name', '&p_first_name',
            lower(substr('&p_first_name', 1, 1) ||
            substr('&p_last_name', 1, 7)), &p_salary)

/
SET VERIFY ON
SET ECHO ON
```

7. Populate the table with the next two rows of sample data by running the script that you created.

```
SQL> START loademp.sql
SQL> START loademp.sql
```

8. Confirm your additions to the table.

```
SQL> SELECT *
      2 FROM my_employee;
```

9. Make the data additions permanent.

```
SQL> COMMIT;
```

Practice 9 Solutions (continued)

Update and delete data in the MY_EMPLOYEE table.

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

```
SQL> UPDATE my_employee
  2 SET    last_name = 'Drexler'
  3 WHERE id = 3;
```

11. Change the salary to 1000 for all employees with a salary less than 900.

```
SQL> UPDATE my_employee
  2 SET    salary = 1000
  3 WHERE salary < 900;
```

12. Verify your changes to the table.

```
SQL> SELECT last_name, salary
  2 FROM    my_employee;
```

13. Delete Betty Dancs from the MY_EMPLOYEE table.

```
SQL> DELETE
  2 FROM    my_employee
  3 WHERE  last_name = 'Dancs'
  4 AND    first_name = 'Betty';
```

14. Confirm your changes to the table.

```
SQL> SELECT *
  2 FROM    my_employee;
```

15. Commit all pending changes.

```
SQL> COMMIT;
```

Control data transaction to the MY_EMPLOYEE table.

16. Populate the table with the last row of sample data by running the script that you created in step 6.

```
SQL> START loademp.sql
```

Practice 9 Solutions (continued)

17. Confirm your addition to the table.

```
SQL> SELECT *  
      2 FROM my_employee;
```

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

```
SQL> SAVEPOINT a;
```

19. Empty the entire table.

```
SQL> DELETE  
      2 FROM my_employee;
```

20. Confirm that the table is empty.

```
SQL> SELECT *  
      2 FROM my_employee;
```

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

```
SQL> ROLLBACK TO SAVEPOINT a;
```

22. Confirm that the new row is still intact.

```
SQL> SELECT *  
      2 FROM my_employee;
```

23. Make the data addition permanent.

```
SQL> COMMIT;
```

Practice 12 Solutions

1. Create a view called EMP_VU based on the employee number, employee name, and department number from the EMP table. Change the heading for the employee name to EMPLOYEE.

```
SQL> CREATE VIEW emp_vu AS
  2 SELECT      empno, ename employee, deptno
  3 FROM        emp;
```

2. Display the contents of the EMP_VU view.

```
SQL> SELECT    *
  2 FROM        emp_vu;
```

3. Select the view name and text from the data dictionary USER_VIEWS.

```
SQL> COLUMN view_name FORMAT A30
SQL> COLUMN text FORMAT A50
SQL> SELECT    view_name, text
  2 FROM        user_views;
```

4. Using your view EMP_VU, enter a query to display all employee names and department numbers.

```
SQL> SELECT    employee, deptno
  2 FROM        emp_vu;
```

5. Create a view named DEPT20 that contains the employee number, employee name, and department number for all employees in department 20. Label the view column EMPLOYEE_ID, EMPLOYEE, and DEPARTMENT_ID. Do not allow an employee to be reassigned to another department through the view.

```
SQL> CREATE VIEW dept20 AS
  2 SELECT      empno employee_id, ename employee,
  3             deptno department_id
  4 FROM        emp
  5 WHERE       deptno = 20
  6 WITH CHECK OPTION CONSTRAINT emp_dept_20;
```

Practice 12 Solutions (continued)

6. Display the structure and contents of the DEPT20 view.

```
SQL> DESCRIBE dept20
SQL> SELECT *
  2 FROM dept20;
```

7. Attempt to reassign Smith to department 30.

```
SQL> UPDATE dept20
  2 SET department_id = 30
  3 WHERE employee = 'SMITH';
```

If you have time, complete the following exercise:

8. Create a view called SALARY_VU based on the employee name, department name, salary and salary grade for all employees. Label the columns Employee, Department, Salary, and Grade, respectively.

```
SQL> CREATE VIEW salary_vu AS
  2 SELECT ename employee, dname department,
  3        sal salary, grade
  4 FROM emp e, dept d, salgrade s
  5 WHERE e.deptno = d.deptno
  6 AND e.sal between s.losal and s.hisal;
```

Practice 13 Solutions

1. Create a sequence to be used with the primary key column of the DEPARTMENT table. The sequence should start at 60 and have a maximum value of 200. Have your sequence increment by ten numbers. Name the sequence DEPT_ID_SEQ.

```
SQL> CREATE SEQUENCE dept_id_seq
  2  START WITH 60
  3  INCREMENT BY 10
  4  MAXVALUE 200;
```

2. Write a script to display the following information about your sequences: sequence name, maximum value, increment size, and last number. Name the script p13q2.sql. Execute your script.

```
SQL> EDIT p13q2.sql
      SELECT      sequence_name, max_value,
                 increment_by, last_number
      FROM        user_sequences
      /
SQL> START p13q2.sql
```

3. Write an interactive script to insert a row into the DEPARTMENT table. Name your script p13q3.sql. Be sure to use the sequence that you created for the ID column. Create a customized prompt to enter the department name. Execute your script. Add two departments named Education and Administration. Confirm your additions.

```
SQL> EDIT p13q3.sql
      SET ECHO OFF
      SET VERIFY OFF
      ACCEPT name PROMPT 'Please enter the department name: '
      INSERT INTO      department (id, name)
      VALUES          (dept_id_seq.NEXTVAL, '&name')
      /
      SET VERIFY ON
      SET ECHO ON
SQL> START p13q3.sql
SQL> SELECT      *
  2  FROM        department;
```

4. Create a non-unique index on the foreign key column (dept_id) in the EMPLOYEE table.

```
SQL> CREATE INDEX employee_dept_id_idx ON employee (dept_id);
```

Practice 13 Solutions (continued)

5. Display the indexes and uniqueness that exist in the data dictionary for the EMPLOYEE table. Save the statement into a script named p13q5.sql.

```
SQL> SELECT index_name, table_name, uniqueness
       2 FROM user_indexes
       3 WHERE table_name = 'EMPLOYEE';
SQL> SAVE p13q5.sql
```

Practice 14 Solutions

1. What privilege should a user be given to log in to the Oracle Server? Is this a system or an object privilege?

The CREATE SESSION system privilege

2. What privilege should a user be given to create tables?

The CREATE TABLE privilege

3. If you create a table, who can pass along privileges to other users on your table?

You can, or anyone you have given those privileges to by using the WITH GRANT OPTION.

4. You are the DBA. You are creating many users who require the same system privileges. What would you use to make your job easier?

Create a role containing the system privileges and grant the role to the users

5. What command do you use to change your password?

The ALTER USER statement

6. Grant another user access to your DEPT table. Have the user grant you query access to his or her DEPT table.

Team 2 executes the GRANT statement.

```
SQL> GRANT    select
      2 ON     dept
      3 TO     <user1>;
```

Team 1 executes the GRANT statement.

```
SQL> GRANT    select
      2 ON     dept
      3 TO     <user2>;
```

WHERE *user1* is the name of team 1 and *user2* is the name of team 2.

7. Query all the rows in your DEPT table.

```
SQL> SELECT  *
      2 FROM   dept;
```

Practice 14 Solutions (continued)

8. Add a new row to your DEPT table. Team 1 should add Education as department number 50. Team 2 should add Administration as department number 50. Make the changes permanent.

Team 1 executes this INSERT statement.

```
SQL> INSERT INTO dept(deptno, dname)
  2 VALUES (50, 'Education');
SQL> COMMIT;
```

Team 2 executes this INSERT statement.

```
SQL> INSERT INTO dept(deptno, dname)
  2 VALUES (50, 'Administration');
SQL> COMMIT;
```

9. Create a synonym for the other team's DEPT table.

Team 1 creates a synonym named team2.

```
SQL> CREATE SYNONYM team2
  2 FOR <user2>.DEPT;
```

Team 2 creates a synonym named team1.

```
SQL> CREATE SYNONYM team1
  2 FOR <user1>.DEPT;
```

10. Query all the rows in the other team's DEPT table by using your synonym.

Team 1 executes this SELECT statement.

```
SQL> SELECT *
  2 FROM team2;
```

Team 2 executes this SELECT statement.

```
SQL> SELECT *
  2 FROM team1;
```

Practice 14 Solutions (continued)

11. Query the USER_TABLES data dictionary to see information about the tables that you own.

```
SQL> SELECT table_name
       2 FROM user_tables;
```

12. Query the ALL_TABLES data dictionary view to see information about all the tables that you can access. Exclude tables that you own.

```
SQL> SELECT table_name, owner
       2 FROM all_tables
       3 WHERE owner != <your account>;
```

13. Revoke the SELECT privilege from the other team.

Team 1 revokes the privilege.

```
SQL> REVOKE select
       2 ON dept
       3 FROM user2;
```

Team 2 revokes the privilege.

```
SQL> REVOKE select
       2 ON dept
       3 FROM user1;
```

Practice 15 Solutions

1. Create the tables based on the following table instance charts. Choose the appropriate datatypes and be sure to 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
Datatype	Number	VARCHAR2	VARCHAR2	VARCHAR2	VARCHAR2	VARCHAR2	Date
Length	10	25	25	100	30	15	

```
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);
```

Practice 15 Solutions (continued)

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	
Datatype	Number	VARCHAR2	VARCHAR2	VARCHAR2	VARCHAR2	Date
Length	10	60	400	4	20	

```

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);

```

Practice 15 Solutions (continued)

c. Table name: TITLE_COPY

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

```
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));
```

Practice 15 Solutions (continued)

d. Table name: RENTAL

Column Name	BOOK_DATE	MEMBER_ID	COPY_ID	ACT_RET_DATE	EXP_RET_DATE	TITLE_ID
Key Type	PK	PK,FK1	PK,FK2			PK,FK2
Default Value	System Date				2 days after book date	
FK Ref Table		MEMBER	TITLE_COPY			TITLE_COPY
FK Ref Col		MEMBER_ID	COPY_ID			TITLE_ID
Datatype	Date	Number	Number	Date	Date	Number
Length		10	10			10

```
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));
```

Practice 15 Solutions (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
Datatype	Date	Number	Number
Length		10	10

```
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));
```

Practice 15 Solutions (continued)

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

```
SQL> SELECT   table_name
 2 FROM     user_tables
 3 WHERE    table_name IN ('MEMBER', 'TITLE', 'TITLE_COPY',
 4                          'RENTAL', 'RESERVATION');
```

```
SQL> COLUMN   constraint_name FORMAT A30
SQL> COLUMN   table_name FORMAT A15
SQL> SELECT   constraint_name, constraint_type,
 2           table_name
 3 FROM     user_constraints
 4 WHERE    table_name IN ('MEMBER', 'TITLE', 'TITLE_COPY',
 5                          '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 the values. Name the sequence member_id_seq.

```
SQL> CREATE SEQUENCE member_id_seq
 2 START WITH 101
 3 NOCACHE;
```

- b. Title number for the TITLE table: start with 92; no caching. Name the sequence title_id_seq.

```
SQL> CREATE SEQUENCE title_id_seq
 2 START WITH 92
 3 NOCACHE;
```

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

```
SQL> SELECT   sequence_name, increment_by, last_number
 2 FROM     user_sequences
 3 WHERE    sequence_name IN ('MEMBER_ID_SEQ',
 4                          'TITLE_ID_SEQ');
```

Practice 15 Solutions (continued)

4. Add data to the tables. Create a script for each set of data to add.
 - a. Add movie titles to the TITLE table. Write a script to enter the movie information. Save the script as p15q4a.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.

```
SQL> EDIT p15q4a.sql
SET ECHO OFF
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
/
SET ECHO ON

SQL> SELECT title
       2 FROM title;
```

Practice 15 Solutions (continued)

Title	Description	Rating	Category	Release_date
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	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

Practice 15 Solutions (continued)

- b. Add data to the MEMBER table. Write a script named p15q4b.sql to prompt users for the information. Execute the script. Be sure to use the sequence to add the member numbers.

First_Name	Last_Name	Address	City	Phone	Join_Date
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 Centralc	Sao Paolo	254-852-5764	17-JUN-1991
Mark	Quick-to- Scott	6921 King Way	Lagos	63-559-7777	07-APR-1990
Audry	Ropburn	86 Chu Street	Hong Kong	41-559-87	18-JAN-1991
Molly	Urguhart	3035 Laurier	Quebec	418-542-9988	18-JAN-1991

```
SQL> EDIT p15q4b.sql
SET ECHO OFF
SET VERIFY OFF
INSERT INTO member(member_id, first_name, last_name, address,
                    city, phone, join_date)
VALUES (member_id_seq.NEXTVAL, '&first_name', '&last_name',
        '&address', '&city', '&phone', TO_DATE('&join_date',
        'DD-MM-YYYY'))
/
COMMIT
/
SET VERIFY ON
SET ECHO ON
SQL> START p15q4b.sql
```

Practice 15 Solutions (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
Willie and Christmas Too	1	AVAILABLE
Alien Again	1	AVAILABLE
	2	RENTED
The Glob	1	AVAILABLE
My Day Off	1	AVAILABLE
	2	AVAILABLE
	3	RENTED
Miracles on Ice	1	AVAILABLE
Soda Gang	1	AVAILABLE

```
SQL> INSERT INTO title_copy(copy_id, title_id, status)
      2 VALUES (1, 92, 'AVAILABLE');
SQL> INSERT INTO title_copy(copy_id, title_id, status)
      2 VALUES (1, 93, 'AVAILABLE');
SQL> INSERT INTO title_copy(copy_id, title_id, status)
      2 VALUES (2, 93, 'RENTED');
SQL> INSERT INTO title_copy(copy_id, title_id, status)
      2 VALUES (1, 94, 'AVAILABLE');
SQL> INSERT INTO title_copy(copy_id, title_id, status)
      2 VALUES (1, 95, 'AVAILABLE');
SQL> INSERT INTO title_copy(copy_id, title_id, status)
      2 VALUES (2, 95, 'AVAILABLE');
SQL> INSERT INTO title_copy(copy_id, title_id, status)
      2 VALUES (3, 95, 'RENTED');
SQL> INSERT INTO title_copy(copy_id, title_id, status)
      2 VALUES (1, 96, 'AVAILABLE');
SQL> INSERT INTO title_copy(copy_id, title_id, status)
      2 VALUES (1, 97, 'AVAILABLE');
```

Practice 15 Solutions (continued)

d. Add the following rentals to the RENTAL table:

Note: Title number may be different depending on sequence number.

Title_ Id	Copy_ Id	Member_ Id	Book_date	Exp_Ret_Date	Act_Ret_Date
92	1	101	3 days ago	1 day ago	2 days 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	2 days ago

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

Practice 15 Solutions (continued)

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

```
SQL> CREATE VIEW title_avail AS
  2   SELECT   t.title, c.copy_id, c.status, r.exp_ret_date
  3   FROM     title t, title_copy c, rental r
  4   WHERE    t.title_id = c.title_id
  5   AND      c.copy_id = r.copy_id(+)
  6   AND      c.title_id = r.title_id(+);
```

```
SQL> COLUMN title FORMAT A30
```

```
SQL> SELECT   *
  2   FROM     title_avail
  3   ORDER BY title, copy_id;
```

6. Make changes to data in the tables.
- a. Add a new title. The movie is "Interstellar Wars," which is rated PG and classified as a sci-fi 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."

```
SQL> INSERT INTO reservation (res_date, member_id, title_id)
  2   VALUES (SYSDATE, 101, 98);
SQL> INSERT INTO reservation (res_date, member_id, title_id)
  2   VALUES (SYSDATE, 104, 97);
```

Practice 15 Solutions (continued)

- c. Customer Carmen Velasquez rents the movie “Interstellar Wars,” copy 1. Remove her reservation for the movie. Record the information about the rental. Allow the default value for the expected return date to be used. Verify that the rental was recorded by using the view you created.

```
SQL> INSERT INTO rental(title_id, copy_id, member_id)
  2 VALUES (98, 1,101);
SQL> UPDATE title_copy
  2 SET status= 'RENTED'
  3 WHERE title_id = 98
  4 AND copy_id = 1;
SQL> DELETE
  2 FROM reservation
  3 WHERE member_id = 101;
SQL> SELECT *
  2 FROM title_avail
  3 ORDER BY title, copy_id;
```

7. Make a modification to one of the tables.
 - a. Add a PRICE column to the TITLE table to record the purchase price of the video. The column should have a total length of eight digits and two decimal places. Verify your modifications.

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

Practice 15 Solutions (continued)

- b. Create a script named p15q7b.sql to update each video with a price according to the following list.

Note: Have the title_id numbers available for this exercise.

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

```
SET ECHO OFF
SET VERIFY OFF
UPDATE      title
SET          price = &price
WHERE       title_id = &title_id
/
SET VERIFY OFF
SET ECHO OFF
SQL> START p15q7b.sql
```

- c. Ensure that in the future all titles will contain a price value. Verify the constraint.

```
SQL> ALTER TABLE title
  2  MODIFY (price CONSTRAINT title_price_nn NOT NULL);
SQL> SELECT  constraint_name, constraint_type,
  2          search_condition
  3  FROM      user_constraints
  4  WHERE     table_name = 'TITLE';
```

Practice 15 Solutions (continued)

8. Create a report titled Customer History Report. This report will contain 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 script in a file named p15q8.sql.

```
SQL> EDIT p15q8.sql
SET ECHO OFF
SET VERIFY OFF
SET PAGESIZE 30
COLUMN member FORMAT A17
COLUMN title FORMAT A25
COLUMN book_date FORMAT A9
COLUMN duration FORMAT 9999999
TTITLE 'Customer History Report'
BREAK ON member SKIP 1 ON REPORT
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
/
CLEAR BREAK
COLUMN member CLEAR
COLUMN title CLEAR
COLUMN book_date CLEAR
COLUMN duration CLEAR
TTITLE OFF
SET VERIFY ON
SET PAGESIZE 24
SET ECHO ON
```

B

.....

Table Descriptions and Data

EMP Table

```
SQL> DESCRIBE emp
```

Name	Null?	Type
-----	-----	-----
EMPNO	NOT NULL	NUMBER(4)
ENAME		VARCHAR2(10)
JOB		VARCHAR2(9)
MGR		NUMBER(4)
HIREDATE		DATE
SAL		NUMBER(7,2)
COMM		NUMBER(7,2)
DEPTNO	NOT NULL	NUMBER(2)

```
SQL> SELECT * FROM emp;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
-----	-----	-----	-----	-----	-----	-----	-----
7839	KING	PRESIDENT		17-NOV-81	5000		10
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7788	SCOTT	ANALYST	7566	09-DEC-82	3000		20
7876	ADAMS	CLERK	7788	12-JAN-83	1100		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

DEPT Table

SQL> DESCRIBE dept

Name	Null?	Type
DEPTNO	NOT NULL	NUMBER(2)
DNAME		VARCHAR2(14)
LOC		VARCHAR2(13)

SQL> SELECT * FROM dept;

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

SALGRADE Table

```
SQL> DESCRIBE salgrade
```

Name	Null?	Type
-----	-----	-----
GRADE		NUMBER
LOSAL		NUMBER
HISAL		NUMBER

```
SQL> SELECT * FROM salgrade;
```

GRADE	LOSAL	HISAL
-----	-----	-----
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

ORD Table

SQL> DESCRIBE ord

Name	Null?	Type
ORDID	NOT NULL	NUMBER(4)
ORDERDATE		DATE
COMMPLAN		VARCHAR2(1)
CUSTID	NOT NULL	NUMBER(6)
SHIPDATE		DATE
TOTAL		NUMBER(8,2)

SQL> SELECT * FROM ord;

ORDID	ORDERDATE	C	CUSTID	SHIPDATE	TOTAL
610	07-JAN-87	A	101	08-JAN-87	101.4
611	11-JAN-87	B	102	11-JAN-87	45
612	15-JAN-87	C	104	20-JAN-87	5860
601	01-MAY-86	A	106	30-MAY-86	2.4
602	05-JUN-86	B	102	20-JUN-86	56
604	15-JUN-86	A	106	30-JUN-86	698
605	14-JUL-86	A	106	30-JUL-86	8324
606	14-JUL-86	A	100	30-JUL-86	3.4
609	01-AUG-86	B	100	15-AUG-86	97.5
607	18-JUL-86	C	104	18-JUL-86	5.6
608	25-JUL-86	C	104	25-JUL-86	35.2
603	05-JUN-86		102	05-JUN-86	224
620	12-MAR-87		100	12-MAR-87	4450
613	01-FEB-87		108	01-FEB-87	6400
614	01-FEB-87		102	05-FEB-87	23940
616	03-FEB-87		103	10-FEB-87	764
619	22-FEB-87		104	04-FEB-87	1260
617	05-FEB-87		105	03-MAR-87	46370
615	01-FEB-87		107	06-FEB-87	710
618	15-FEB-87	A	102	06-MAR-87	3510.5
621	15-MAR-87	A	100	01-JAN-87	730

PRODUCT Table

SQL> DESCRIBE product

Name	Null?	Type
PRODID	NOT NULL	NUMBER(6)
DESCRIP		VARCHAR2(30)

SQL> SELECT * FROM product;

PRODID DESCRIP

100860	ACE TENNIS RACKET I
100861	ACE TENNIS RACKET II
100870	ACE TENNIS BALLS-3 PACK
100871	ACE TENNIS BALLS-6 PACK
100890	ACE TENNIS NET
101860	SP TENNIS RACKET
101863	SP JUNIOR RACKET
102130	RH: "GUIDE TO TENNIS"
200376	SB ENERGY BAR-6 PACK
200380	SB VITA SNACK-6 PACK

ITEM Table

SQL> DESCRIBE item

Name	Null?	Type
ORDID	NOT NULL	NUMBER(4)
ITEMID	NOT NULL	NUMBER(4)
PRODID		NUMBER(6)
ACTUALPRICE		NUMBER(8,2)
QTY		NUMBER(8)
ITEMTOT		NUMBER(8,2)

SQL> SELECT * FROM item;

ORDID	ITEMID	PRODID	ACTUALPRICE	QTY	ITEMTOT
610	3	100890	58	1	58
611	1	100861	45	1	45
612	1	100860	30	100	3000
601	1	200376	2.4	1	2.4
602	1	100870	2.8	20	56
604	1	100890	58	3	174
604	2	100861	42	2	84
604	3	100860	44	10	440
603	2	100860	56	4	224
610	1	100860	35	1	35
610	2	100870	2.8	3	8.4
613	4	200376	2.2	200	440
614	1	100860	35	444	15540
614	2	100870	2.8	1000	2800
612	2	100861	40.5	20	810
612	3	101863	10	150	1500
620	1	100860	35	10	350
620	2	200376	2.4	1000	2400
620	3	102130	3.4	500	1700
613	1	100871	5.6	100	560
613	2	101860	24	200	4800
613	3	200380	4	150	600
619	3	102130	3.4	100	340
617	1	100860	35	50	1750
617	2	100861	45	100	4500
614	3	100871	5.6	1000	5600

Continued on next page

ITEM Table (continued)

ORDID	ITEMID	PRODID	ACTUALPRICE	QTY	ITEMTOT
616	1	100861	45	10	450
616	2	100870	2.8	50	140
616	3	100890	58	2	116
616	4	102130	3.4	10	34
616	5	200376	2.4	10	24
619	1	200380	4	100	400
619	2	200376	2.4	100	240
615	1	100861	45	4	180
607	1	100871	5.6	1	5.6
615	2	100870	2.8	100	280
617	3	100870	2.8	500	1400
617	4	100871	5.6	500	2800
617	5	100890	58	500	29000
617	6	101860	24	100	2400
617	7	101863	12.5	200	2500
617	8	102130	3.4	100	340
617	9	200376	2.4	200	480
617	10	200380	4	300	1200
609	2	100870	2.5	5	12.5
609	3	100890	50	1	50
618	1	100860	35	23	805
618	2	100861	45.11	50	2255.5
618	3	100870	45	10	450
621	1	100861	45	10	450
621	2	100870	2.8	100	280
615	3	100871	5	50	250
608	1	101860	24	1	24
608	2	100871	5.6	2	11.2
609	1	100861	35	1	35
606	1	102130	3.4	1	3.4
605	1	100861	45	100	4500
605	2	100870	2.8	500	1400
605	3	100890	58	5	290
605	4	101860	24	50	1200
605	5	101863	9	100	900
605	6	102130	3.4	10	34
612	4	100871	5.5	100	550
619	4	100871	5.6	50	280

CUSTOMER Table

SQL> DESCRIBE customer

Name	Null?	Type
CUSTID	NOT NULL	NUMBER(6)
NAME		VARCHAR2(45)
ADDRESS		VARCHAR2(40)
CITY		VARCHAR2(30)
STATE		VARCHAR2(2)
ZIP		VARCHAR2(9)
AREA		NUMBER(3)
PHONE		VARCHAR2(9)
REPID	NOT NULL	NUMBER(4)
CREDITLIMIT		NUMBER(9,2)
COMMENTS		LONG

CUSTOMER Table (continued)

SQL> SELECT * FROM customer;

CUSTID	NAME	ADDRESS
100	JOCKSPORTS	345 VIEWRIDGE
101	TKB SPORT SHOP	490 BOLI RD.
102	VOLLYRITE	9722 HAMILTON
103	JUST TENNIS	HILLVIEW MALL
104	EVERY MOUNTAIN	574 SURRY RD.
105	K + T SPORTS	3476 EL PASEO
106	SHAPE UP	908 SEQUOIA
107	WOMENS SPORTS	VALCO VILLAGE
108	NORTH WOODS HEALTH AND FITNESS SUPPLY CENTER	98 LONE PINE WAY

CITY	ST	ZIP	AREA	PHONE	REPID	CREDITLIMIT
BELMONT	CA	96711	415	598-6609	7844	5000
REDWOOD CITY	CA	94061	415	368-1223	7521	10000
BURLINGAME	CA	95133	415	644-3341	7654	7000
BURLINGAME	CA	97544	415	677-9312	7521	3000
CUPERTINO	CA	93301	408	996-2323	7499	10000
SANTA CLARA	CA	91003	408	376-9966	7844	5000
PALO ALTO	CA	94301	415	364-9777	7521	6000
SUNNYVALE	CA	93301	408	967-4398	7499	10000
HIBBING	MN	55649	612	566-9123	7844	8000

COMMENTS

Very friendly people to work with -- sales rep likes to be called Mike.
 Rep called 5/8 about change in order - contact shipping.
 Company doing heavy promotion beginning 10/89. Prepare for large orders during winter
 Contact rep about new line of tennis rackets.
 Customer with high market share (23%) due to aggressive advertising.
 Tends to order large amounts of merchandise at once. Accounting is considering raising their credit limit
 Support intensive. Orders small amounts (< 800) of merchandise at a time.
 First sporting goods store geared exclusively towards women. Unusual promotional style

PRICE Table

SQL> DESCRIBE price

Name	Null?	Type
PRODID	NOT NULL	NUMBER(6)
STDPRICE		NUMBER(8,2)
MINPRICE		NUMBER(8,2)
STARTDATE		DATE
ENDDATE		DATE

SQL> SELECT * FROM price;

PRODID	STDPRICE	MINPRICE	STARTDATE	ENDDATE
100871	4.8	3.2	01-JAN-85	01-DEC-85
100890	58	46.4	01-JAN-85	
100890	54	40.5	01-JUN-84	31-MAY-84
100860	35	28	01-JUN-86	
100860	32	25.6	01-JAN-86	31-MAY-86
100860	30	24	01-JAN-85	31-DEC-85
100861	45	36	01-JUN-86	
100861	42	33.6	01-JAN-86	31-MAY-86
100861	39	31.2	01-JAN-85	31-DEC-85
100870	2.8	2.4	01-JAN-86	
100870	2.4	1.9	01-JAN-85	01-DEC-85
100871	5.6	4.8	01-JAN-86	
101860	24	18	15-FEB-85	
101863	12.5	9.4	15-FEB-85	
102130	3.4	2.8	18-AUG-85	
200376	2.4	1.75	15-NOV-86	
200380	4	3.2	15-NOV-86	

Index

Symbol

3-34

% 2-13

&& 8-10

& 8-4

(+) 4-17

* 1-6

; 1-5

A

ACCEPT 8-11

ADD clause. 11-17

ADD_MONTHS 3-20

alias 1-16

ALL 6-16

ALTER SEQUENCE 13-12

 TABLE 10-15

 USER 14-11

AND 2-16

ANY 6-15

arithmetic operators 1-9

AS 1-17

ASC 2-22

Attribute 1-9

AVG 5-6

B

base table 12-5

BETWEEN 2-10

BREAK 8-23

BTITLE 8-24

C

- CACHE 13-5
- Cartesian product 4-5
- CASCADE CONSTRAINTS 11-25
- CHECK constraint 11-16
- COLUMN 8-21
- COLUMN format models 8-22
- column heading 1-8
- COMMENT 10-24
- COMMIT 9-27
- Comparison operators 2-7
- complex view 12-7
- concatenation operator 1-18
- constraints 11-3
- COUNT 5-8
- CREATE INDEX 13-17
 - SEQUENCE 13-6
 - SYNONYM 13-23
 - TABLE 10-5
 - USER 14-6
 - VIEW 12-8
- CURRVAL 13-8
- CYCLE 13-5

D

- data definition language 10-5
- data dictionary 10-9
- Data manipulation language 9-3
- database 1-5
- Database security 14-3
- datatype 1-29

D

datatype conversion 3-23

dates 3-17

DD-MON-YY 3-17

DDL 10-5

DECODE 3-39

DEFAULT 10-7

DEFINE 8-11

DELETE 9-20

DESC 2-22

DESCRIBE 1-28

DISABLE clause 11-20

DISTINCT 1-23

DML 9-3

double-ampersand 8-10

DROP clause 11-19

 COLUMN 10-19

 INDEX 13-22

 SEQUENCE 13-14

 TABLE 10-21

 VIEW 12-19

E

ENABLE clause 11-21

Entity 1-9

entity relationship (ER) 1-9

Equijoins 4-8

ESCAPE 2-13

F

- field 1-12
- FOREIGN KEY 11-13
- format model 3-29
- functions 3-3

G

- GRANT 14-8
- GROUP BY 5-12
- group functions 5-3

H

- HAVING 5-21
- IN 2-11

I

- INCREMENT BY 13-5
- index 13-15
- INITCAP 3-9
- Inline View 12-20
- INSERT 9-5
- IS NULL 2-14

J

- join 4-4

L

- LAST_DAY 3-20
- LENGTH 3-11
- LIKE 2-12
- literal 1-20
- Locks 9-37
- logical operators 2-15

login.sql 8-18

LOWER 3-9

LPAD 3-11

M

MAX 5-7

MAXVALUE 13-5

MIN 5-7

MINVALUE 13-5

MOD 3-16

MODIFY 10-18

MONTHS_BETWEEN 3-20

Multiple-column subqueries 6-7

Multiple-row subqueries 6-7

N

Nested functions 3-42

NEXT_DAY 3-20

NEXTVAL 13-8

non-equijoin 4-14

nonpairwise comparisons 7-6

NOT 2-18

NOT NULL constraint 11-7

null value 1-12

Number functions 3-13

NVL 3-37

O

object privilege 14-4
ON DELETE CASCADE 11-15
OR 2-17
ORDER BY 2-22
order of precedence 1-12
outer join 4-17

P

pairwise comparisons 7-6
PRIMARY KEY 11-11
Privileges 14-4
PUBLIC 14-16

R

read consistency 9-35
REFERENCES 11-15
referential integrity 11-13
relational database 1-7
Relational database management systems 1-6
Relationship 1-9
RENAME 10-22
reports 8-3
REVOKE 14-18
role 14-9
ROLLBACK 9-27
ROUND 3-14
ROWNUM 12-22
RPAD 3-11
RR date format 3-36

S

SAVEPOINT 9-27
Schema 10-6
SELECT 1-3
self join 4-19
sequence 13-4
SET UNUSED 10-20
SET VERIFY 8-6
simple view 12-7
Multiple-row functions 3-4
Single-row functions 3-5
single-row subquery 6-8
Single-row subqueries 6-7
SQL 1-15
SQL buffer 1-5
SQL*Plus 1-24
START WITH 13-5
statement-level rollback 9-34
subquery 6-4
subquery in the FROM clause 7-10
SUBSTR 3-11
SUM 5-6
synonym 13-23
SYSDATE 3-17
system development life cycle 1-3
system privilege 14-4

T

table alias 4-12
TO_CHAR 3-33
TO_DATE 3-35
TO_DATE function 9-9
TO_NUMBER 3-35
transaction 9-3
TRIM 3-11
TRUNC 3-15
TRUNCATE TABLE 10-23
TTITLE 8-24
"Top-N" analysis 12-21
tuple 1-12

U

UNDEFINE 8-14
UNIQUE key constraint 11-9
UPDATE 9-14
UPPER 3-9
USER_CONS_COLUMNS 1-28
USER_CONSTRAINTS 11-4
USER_IND_COLUMNS 13-20
USER_INDEXES 13-20
USER_SEQUENCES 13-7

V

view 12-5

W

WHERE 2-4
WITH CHECK OPTION 12-17
 GRANT OPTION 14-13
 READ ONLY 12-18

ORACLE®

Oracle is a registered trademark of Oracle Corporation. All Oracle product names are trademarks or registered trademarks of Oracle Corporation. All other companies and product names mentioned are used for identification purposes only, and may be trademarks of their respective owner.

**Copyright © Oracle Corporation 1998
All Rights Reserved**