Oracle Database 10*g*: SQL Fundamentals I

Student Guide • Volume III

D17108GC21
Edition 2.1
December 2006
D48185

ORACLE

Authors

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
Angelika Krupp
Isabelle Marchand
Malika Marghadi
Valli Pataballa
Bryan Roberts
Helen Robertson
Lata Shivaprasad
John Soltani
Priya Vennapusa
Narayanan Radhakrishnan

Editors

Arijit Ghosh
Raj Kumar

Graphic Designer

Rajiv Chandrabhanu

Publisher

Giri Venugopal

Contents

Preface

Profile

Before You Begin This Course

Before you begin this course, you should be able to use a graphical user interface (GUI). The prerequisite is a familiarity with data processing concepts and techniques.

How This Course Is Organized

Oracle Database 10g: SQL Fundamentals I is an instructor-led course featuring lectures and hands-on exercises. Online demonstrations and written practice sessions reinforce the concepts and skills that are introduced.

Related Publications

Oracle Publications

Title	Part Number
Oracle® Database Reference 10g Release 2 (10.2)	B14237-02
Oracle® Database SQL Reference 10g Release 2 (10.2)	B14200-02
Oracle® Database Concepts 10*g* Release *2 (10.2)*	B14220-02
Oracle® Database Application Developer's Guide - Fundamentals 10g Release 2 (10.2)	B14251-01
*SQL*Plus® User's Guide and Reference*	B14357-01

Additional Publications

- System release bulletins
- Installation and user's guides
- *read.me* files
- International Oracle User's Group (IOUG) articles
- *Oracle Magazine*

Typographic Conventions

What follows are two lists of typographical conventions that are used specifically within text or within code.

Typographic Conventions Within Text

Convention	Object or Term	Example
Uppercase	Commands, functions, column names, table names, PL/SQL objects, schemas	Use the `SELECT` command to view information stored in the `LAST_NAME` column of the `EMPLOYEES` table.
Lowercase, italic	Filenames, syntax variables, usernames, passwords	**where:** *role* is the name of the role to be created.
Initial cap	Trigger and button names	Assign a When-Validate-Item trigger to the ORD block. Choose Cancel.
Italic	Books, names of courses and manuals, and emphasized words or phrases	For more information on the subject see *Oracle SQL Reference Manual* Do *not* save changes to the database.
Quotation marks	Lesson module titles referenced within a course	This subject is covered in Lesson 3, "Working with Objects."

Typographic Conventions (continued)

Typographic Conventions Within Code

Convention	Object or Term	Example
Uppercase	Commands, functions	`SELECT employee_id` `FROM  employees;`
Lowercase, italic	Syntax variables	`CREATE ROLE role;`

Initial cap	Forms triggers	`Form module: ORD`
		`Trigger level: S_ITEM.QUANTITY item`
		`Trigger name: When-Validate-Item`
		`. . .`

Lowercase	Column names, table names, filenames, PL/SQL objects	`. . .`
		`OG_ACTIVATE_LAYER`
		`(OG_GET_LAYER ('prod_pie_layer'))`
		`. . .`
		`SELECT last_name`
		`FROM   employees;`

Bold	Text that must be entered by a user	`CREATE USER scott` `IDENTIFIED BY tiger;`

Additional Practices

Additional Practices

These exercises can be used for extra practice after you have discussed the following topics: basic SQL `SELECT` statement, basic *i*SQL*Plus commands, and SQL functions.

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

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	
143	Randall	Matos	RMATOS	650.121.2874	15-MAR-98	ST
144	Peter	Vargas	PVARGAS	650.121.2004	09-JUL-98	ST

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

LAST_NAME	JOB_ID	SALARY	COMMISSION_PCT
Abel	SA_REP	11000	.3
Zlotkey	SA_MAN	10500	.2
Taylor	SA_REP	8600	.2
Grant	SA_REP	7000	.15

Additional Practices (continued)

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
The salary of King after a 10% raise is 26400
The salary of Kochhar after a 10% raise is 18700
The salary of De Haan after a 10% raise is 18700
The salary of Hunold after a 10% raise is 9900
The salary of Ernst after a 10% raise is 6600
The salary of Lorentz after a 10% raise is 4620
The salary of Mourgos after a 10% raise is 6380
The salary of Rajs after a 10% raise is 3850
The salary of Davies after a 10% raise is 3410
The salary of Matos after a 10% raise is 2860
The salary of Vargas after a 10% raise is 2750
The salary of Whalen after a 10% raise is 4840
The salary of Hartstein after a 10% raise is 14300
The salary of Fay after a 10% raise is 6600
The salary of Higgins after a 10% raise is 13200
The salary of Gietz after a 10% raise is 9130

16 rows selected.

Additional Practices (continued)

4. Create a report of employees and their length 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 length of their employment. The employee who has been employed the longest should appear at the top of the list.

LAST_NAME	YEARS	MONTHS
King	16	7
Whalen	16	4
Kochhar	14	4
Hunold	14	0
Ernst	12	8

• • •

Mourgos	4	2
Zlotkey	3	11

20 rows selected.

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

LAST_NAME
King
Kochhar
Lorentz
Matos
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.
Note: Results are continued on next page.

LAST_NAME	SALARY	COMMISSIO
King	24000	No
Kochhar	17000	No
De Haan	17000	No
Hunold	9000	No
Ernst	6000	No
Lorentz	4200	No
Mourgos	5800	No
Rajs	3500	No

Additional Practices (continued)

6. (continued)

Davies	3100	No
Matos	2600	No
Vargas	2500	No
Zlotkey	10500	Yes
Abel	11000	Yes
Taylor	8600	Yes
Grant	7000	Yes
Whalen	4400	No
Hartstein	13000	No
Fay	6000	No
Higgins	12000	No
Gietz	8300	No

20 rows selected.

Additional Practices (continued)

These exercises can be used for extra practice after you have discussed the following topics: basic SQL `SELECT` statement, basic *i*SQL*Plus commands, 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, these are the results:

DEPARTMENT_NAME	LOCATION_ID	LAST_NAME	JOB_ID	SALARY
Marketing	1800	Hartstein	MK_MAN	13000
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(*)
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)
10	Administration	1700	1
20	Marketing	1800	2
50	Shipping	1500	5
60	IT	1400	3
80	Sales	2500	3
90	Executive	1700	3
110	Accounting	1700	2
190	Contracting	1700	0

8 rows selected.

Additional Practices (continued)

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
AD_ASST
MK_MAN
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
AD_VP	2
AD_ASST	1
AD_PRES	1

Additional Practices (continued)

These exercises can be used for extra practice after you have discussed the following topics: basic SQL `SELECT` statements, basic *i*SQL*Plus commands, 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
De Haan	13-JAN-93
Hunold	03-JAN-90
Lorentz	07-FEB-99
Matos	15-MAR-98
Vargas	09-JUL-98
Abel	11-MAY-96
Higgins	07-JUN-94
Gietz	07-JUN-94

8 rows selected.

13. Create a report that displays the following for all employees: last name, salary, and salary expressed in terms of thousands of dollars.
Note: Results are continued on next page.

LAST_NAME	SALARY	THOUSANDS
King	24000	24
Kochhar	17000	17
De Haan	17000	17
Hunold	9000	9
Ernst	6000	6
Lorentz	4200	4
Mourgos	5800	5
Rajs	3500	3
Davies	3100	3
Matos	2600	2

• • •

Additional Practices (continued)

13. (continued)

Vargas	2500	2
Zlotkey	10500	10
Abel	11000	11
Taylor	8600	8
Grant	7000	7
Whalen	4400	4
Hartstein	13000	13
Fay	6000	6
Higgins	12000	12
Gietz	8300	8

20 rows selected.

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_LEV
Whalen	Kochhar	17000	E
Higgins	Kochhar	17000	E
Hunold	De Haan	17000	E
Kochhar	King	24000	E
De Haan	King	24000	E
Mourgos	King	24000	E
Zlotkey	King	24000	E
Hartstein	King	24000	E

8 rows selected.

Additional Practices (continued)

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.

DEPARTMENT_ID	DEPARTMENT_NAME	EMPLOYEES	AVG_SAL	LAST_NAME	SALA
10	Administration	1	4400.00	Whalen	4
20	Marketing	2	9500.00	Fay	6
20	Marketing	2	9500.00	Hartstein	13
50	Shipping	5	3500.00	Rajs	3
50	Shipping	5	3500.00	Matos	2
50	Shipping	5	3500.00	Davies	3
50	Shipping	5	3500.00	Vargas	2
50	Shipping	5	3500.00	Mourgos	5
60	IT	3	6400.00	Ernst	6
60	IT	3	6400.00	Hunold	9
60	IT	3	6400.00	Lorentz	4
80	Sales	3	10033.33	Abel	11
80	Sales	3	10033.33	Taylor	8
80	Sales	3	10033.33	Zlotkey	10
90	Executive	3	19333.33	King	24
90	Executive	3	19333.33	De Haan	17
90	Executive	3	19333.33	Kochhar	17
110	Accounting	2	10150.00	Gietz	8
110	Accounting	2	10150.00	Higgins	12
190	Contracting	0	No average		

20 rows selected.

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

DEPARTMENT_ID	MIN(SALARY)
90	17000

Additional Practices (continued)

17. Create a report that displays 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
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700

7 rows selected.

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(*)
10	Administration	1
20	Marketing	2
110	Accounting	2

    b. Has the highest number of employees:

DEPARTMENT_ID	DEPARTMENT_NAME	COUNT(*)
50	Shipping	5

    c. Has the lowest number of employees:

DEPARTMENT_ID	DEPARTMENT_NAME	COUNT(*)
10	Administration	1

Additional Practices (continued)

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

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	SALARY	AVG (S.SALARY)
206	Gietz	80	8300	9600
149	Zlotkey	80	10500	9600
174	Abel	80	11000	9600
144	Vargas	50	2500	3500
101	Kochhar	90	17000	19333.3333
100	King	90	24000	19333.3333
103	Hunold	60	9000	6400
142	Davies	50	3100	3500
205	Higgins	110	12000	12000
104	Ernst	60	6000	6400
143	Matos	50	2600	3500
102	De Haan	90	17000	19333.3333
107	Lorentz	60	4200	6400
141	Rajs	50	3500	3500
200	Whalen	10	4400	4400
202	Fay	20	6000	9500
176	Taylor	80	8600	9600
201	Hartstein	20	13000	9500
124	Mourgos	50	5800	3500

19 rows selected.

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
Ernst	TUESDAY
Mourgos	TUESDAY
Rajs	TUESDAY
Taylor	TUESDAY
Higgins	TUESDAY
Gietz	TUESDAY

6 rows selected.

Additional Practices (continued)

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

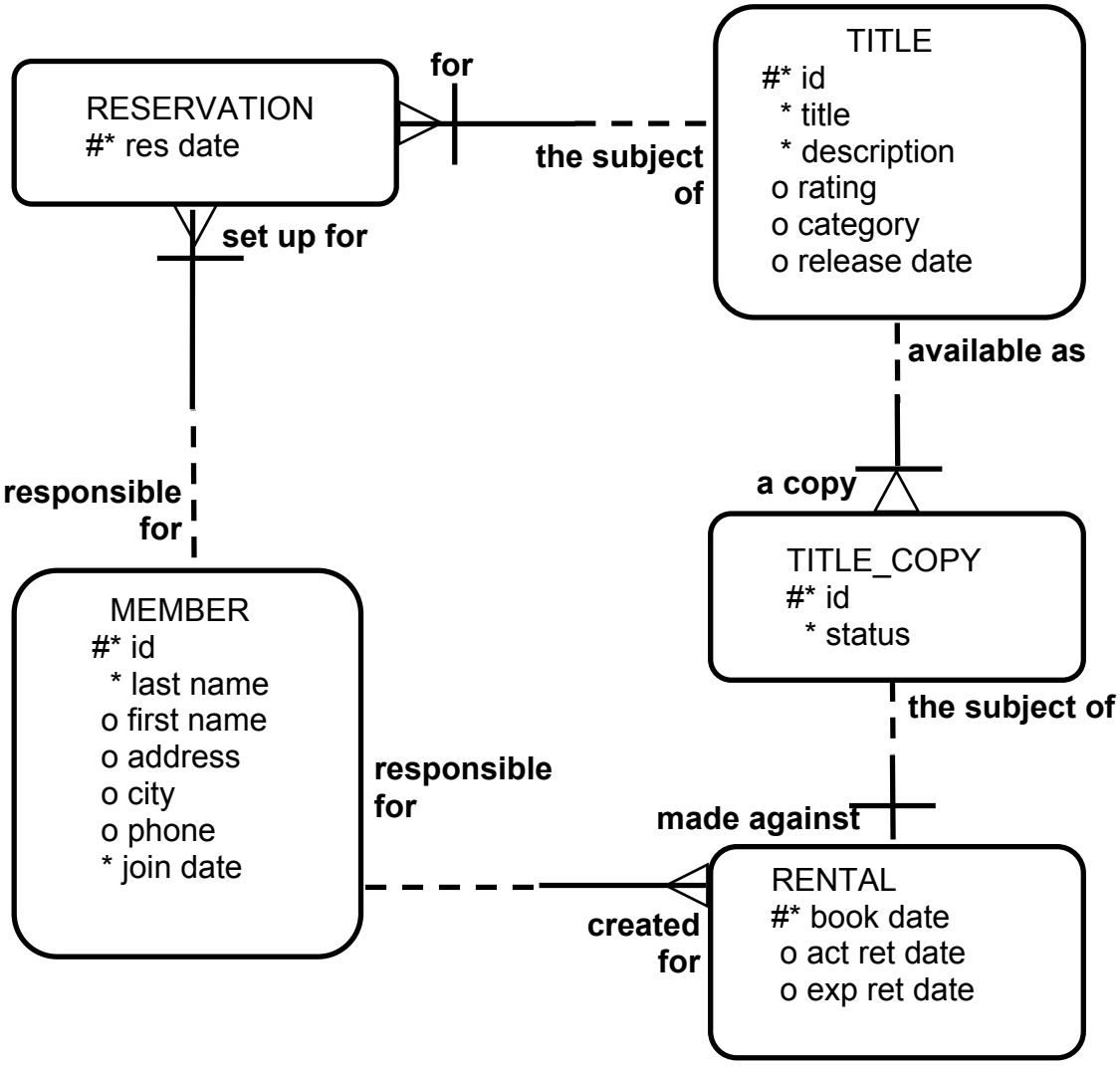
LAST_NAME	BIRTHDAY
Hunold	January 03
De Haan	January 13
Davies	January 29
Zlotkey	January 29
Lorentz	February 07
Hartstein	February 17
Matos	March 15
Taylor	March 24
Abel	May 11
Ernst	May 21
Grant	May 24
Higgins	June 07
Gietz	June 07
King	June 17
Vargas	July 09
Fay	August 17
Whalen	September 17
Kochhar	September 21
Rajs	October 17
Mourgos	November 16

20 rows selected.

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 *i*SQL*Plus. If you want to drop the tables, you can execute the commands in the dropvid.sql script in *i*SQL*Plus. Then you can execute the commands in the `buildvid.sql` script in *i*SQL*Plus 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 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
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, DOCUMEN TARY	
Data Type	NUMBER	VARCHAR2	VARCHAR2	VARCHAR2	VARCHAR2	DATE
Length	10	60	400	4	20	

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	
Data Type	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_I D	COPY_ ID			TITLE_ID
Data Type	DATE	NUMBER	NUMBER	DATE	DATE	NUMBER
Length		10	10			10

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
MEMBER
RENTAL
RESERVATION
TITLE
TITLE_COPY

CONSTRAINT_NAME	CON	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

• • •

COPY_TITLE_ID_PK	P	TITLE_COPY
COPY_TITLE_ID_FK	R	TITLE_COPY

18 rows selected.

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 the values. Name the sequence `MEMBER_ID_SEQ`.

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

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

SEQUENCE_NAME	INCREMENT_BY	LAST_NUMBER
MEMBER_ID_SEQ	1	101
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
Willie and Christmas Too
Alien Again
The Glob
My Day Off
Miracles on Ice
Soda Gang

6 rows selected.

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

b. Add data to the `MEMBER` table. Place the insert statements in a script named `lab_apcs_4b.sql`. Execute commands in 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

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

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
Alien Again	1	AVAILABLE	
Alien Again	2	RENTED	12-FEB-04
Miracles on Ice	1	AVAILABLE	
My Day Off	1	AVAILABLE	
My Day Off	2	AVAILABLE	
My Day Off	3	RENTED	11-FEB-04
Soda Gang	1	AVAILABLE	09-FEB-04
The Glob	1	AVAILABLE	
Willie and Christmas Too	1	AVAILABLE	10-FEB-04

9 rows selected.

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 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.

   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."

TITLE	COPY_ID	STATUS	EXP_RET_DATE
Alien Again	1	AVAILABLE	
Alien Again	2	RENTED	12-FEB-04
Interstellar Wars	1	RENTED	10-FEB-04
Interstellar Wars	2	AVAILABLE	
Miracles on Ice	1	AVAILABLE	
My Day Off	1	AVAILABLE	
My Day Off	2	AVAILABLE	
My Day Off	3	RENTED	11-FEB-04
Soda Gang	1	AVAILABLE	09-FEB-04
The Glob	1	AVAILABLE	
Willie and Christmas Too	1	AVAILABLE	10-FEB-04

11 rows selected.

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.

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
Carmen Velasquez	Alien Again	10-FEB-04	
Carmen Velasquez	Willie and Christmas Too	08-FEB-04	1
LaDoris Ngao	My Day Off	09-FEB-04	
Molly Urguhart	Soda Gang	07-FEB-04	2

Additional Practices

Table Descriptions and Data

COUNTRIES Table

```
DESCRIBE countries
```

Name	Null?	Type
COUNTRY_ID	NOT NULL	CHAR(2)
COUNTRY_NAME		VARCHAR2(40)
REGION_ID		NUMBER

```
SELECT * FROM countries;
```

CO	COUNTRY_NAME	REGION_ID
CA	Canada	2
DE	Germany	1
UK	United Kingdom	1
US	United States of America	2

DEPARTMENTS Table

```
DESCRIBE departments
```

Name	Null?	Type
DEPARTMENT_ID	NOT NULL	NUMBER(4)
DEPARTMENT_NAME	NOT NULL	VARCHAR2(30)
MANAGER_ID		NUMBER(6)
LOCATION_ID		NUMBER(4)

```
SELECT * FROM departments;
```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700

8 rows selected.

EMPLOYEES Table

```
DESCRIBE employees
```

Name	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

```
SELECT * FROM employees;
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE
100	Steven	King	SKING	515.123.4567	17-JUN-87
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-93
103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-90
104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-91
107	Diana	Lorentz	DLORENTZ	590.423.5567	07-FEB-99
124	Kevin	Mourgos	KMOURGOS	650.123.5234	16-NOV-99
141	Trenna	Rajs	TRAJS	650.121.8009	17-OCT-95
142	Curtis	Davies	CDAVIES	650.121.2994	29-JAN-97
143	Randall	Matos	RMATOS	650.121.2874	15-MAR-98
144	Peter	Vargas	PVARGAS	650.121.2004	09-JUL-98
149	Eleni	Zlotkey	EZLOTKEY	011.44.1344.429018	29-JAN-00
174	Ellen	Abel	EABEL	011.44.1644.429267	11-MAY-96
176	Jonathon	Taylor	JTAYLOR	011.44.1644.429265	24-MAR-98
178	Kimberely	Grant	KGRANT	011.44.1644.429263	24-MAY-99
200	Jennifer	Whalen	JWHALEN	515.123.4444	17-SEP-87
201	Michael	Hartstein	MHARTSTE	515.123.5555	17-FEB-96
202	Pat	Fay	PFAY	603.123.6666	17-AUG-97
205	Shelley	Higgins	SHIGGINS	515.123.8080	07-JUN-94
206	William	Gietz	WGIETZ	515.123.8181	07-JUN-94

20 rows selected.

EMPLOYEES Table (continued)

JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
AD_PRES	24000			90
AD_VP	17000		100	90
AD_VP	17000		100	90
IT_PROG	9000		102	60
IT_PROG	6000		103	60
IT_PROG	4200		103	60
ST_MAN	5800		100	50
ST_CLERK	3500		124	50
ST_CLERK	3100		124	50
ST_CLERK	2600		124	50
ST_CLERK	2500		124	50
SA_MAN	10500	.2	100	80
SA_REP	11000	.3	149	80
SA_REP	8600	.2	149	80
SA_REP	7000	.15	149	
AD_ASST	4400		101	10
MK_MAN	13000		100	20
MK_REP	6000		201	20
AC_MGR	12000		101	110
AC_ACCOUNT	8300		205	110

20 rows selected.

JOBS Table

```
DESCRIBE jobs
```

Name	Null?	Type
JOB_ID	NOT NULL	VARCHAR2(10)
JOB_TITLE	NOT NULL	VARCHAR2(35)
MIN_SALARY		NUMBER(6)
MAX_SALARY		NUMBER(6)

```
SELECT * FROM jobs;
```

JOB_ID	JOB_TITLE	MIN_SALARY	MAX_SALARY
AD_PRES	President	20000	40000
AD_VP	Administration Vice President	15000	30000
AD_ASST	Administration Assistant	3000	6000
AC_MGR	Accounting Manager	8200	16000
AC_ACCOUNT	Public Accountant	4200	9000
SA_MAN	Sales Manager	10000	20000
SA_REP	Sales Representative	6000	12000
ST_MAN	Stock Manager	5500	8500
ST_CLERK	Stock Clerk	2000	5000
IT_PROG	Programmer	4000	10000
MK_MAN	Marketing Manager	9000	15000
MK_REP	Marketing Representative	4000	9000

12 rows selected.

JOB_GRADES Table

```
DESCRIBE job_grades
```

Name	Null?	Type
GRADE_LEVEL		VARCHAR2(3)
LOWEST_SAL		NUMBER
HIGHEST_SAL		NUMBER

```
SELECT * FROM job_grades;
```

GRA	LOWEST_SAL	HIGHEST_SAL
A	1000	2999
B	3000	5999
C	6000	9999
D	10000	14999
E	15000	24999
F	25000	40000

6 rows selected.

JOB_HISTORY Table

```
DESCRIBE job_history
```

Name	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
START_DATE	NOT NULL	DATE
END_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
DEPARTMENT_ID		NUMBER(4)

```
SELECT * FROM job_history;
```

EMPLOYEE_ID	START_DAT	END_DATE	JOB_ID	DEPARTMENT_ID
102	13-JAN-93	24-JUL-98	IT_PROG	60
101	21-SEP-89	27-OCT-93	AC_ACCOUNT	110
101	28-OCT-93	15-MAR-97	AC_MGR	110
201	17-FEB-96	19-DEC-99	MK_REP	20
114	24-MAR-98	31-DEC-99	ST_CLERK	50
122	01-JAN-99	31-DEC-99	ST_CLERK	50
200	17-SEP-87	17-JUN-93	AD_ASST	90
176	24-MAR-98	31-DEC-98	SA_REP	80
176	01-JAN-99	31-DEC-99	SA_MAN	80
200	01-JUL-94	31-DEC-98	AC_ACCOUNT	90

10 rows selected.

LOCATIONS Table

```
DESCRIBE locations
```

Name	Null?	Type
LOCATION_ID	NOT NULL	NUMBER(4)
STREET_ADDRESS		VARCHAR2(40)
POSTAL_CODE		VARCHAR2(12)
CITY	NOT NULL	VARCHAR2(30)
STATE_PROVINCE		VARCHAR2(25)
COUNTRY_ID		CHAR(2)

```
SELECT * FROM locations;
```

LOCATION_ID	STREET_ADDRESS	POSTAL_CODE	CITY	STATE_PROVINCE	CO
1400	2014 Jabberwocky Rd	26192	Southlake	Texas	US
1500	2011 Interiors Blvd	99236	South San Francisco	California	US
1700	2004 Charade Rd	98199	Seattle	Washington	US
1800	460 Bloor St. W.	ON M5S 1X8	Toronto	Ontario	CA
2500	Magdalen Centre, The Oxford Science Park	OX9 9ZB	Oxford	Oxford	UK

`REGIONS` Table

```
DESCRIBE regions
```

Name	Null?	Type
REGION_ID	NOT NULL	NUMBER
REGION_NAME		VARCHAR2(25)

```
SELECT * FROM regions;
```

REGION_ID	REGION_NAME
1	Europe
2	Americas
3	Asia
4	Middle East and Africa

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, basic *i*SQL*Plus commands, and SQL functions.

1. The HR department needs to find data for all of the clerks who were hired after the year 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 those 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 length 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 length of their employment. The employee who has been employeed 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 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, basic *i*SQL*Plus commands, 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 those 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, basic *i*SQL*Plus commands, 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).

```
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 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
                            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')));
```

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 the 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_if_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

```
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 the 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 the 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.

```
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.

   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.

```
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;
```

b. Add data to the MEMBER table. Place the insert statements in a script named lab_apcs_4b.sql. Execute 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)
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
```

Additional Practices: Case Study Solutions (continued)

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')
/
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;

SELECT   *
FROM     title_avail
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 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
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
```