Additional Practices: Solutions

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

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

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

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

```
SELECT last_name, job_id, salary, commission_pct
FROM   employees
WHERE  commission_pct IS NOT NULL
ORDER BY salary DESC;
```

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

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

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

```
SELECT last_name,
       TRUNC(MONTHS_BETWEEN(SYSDATE, hire_date) / 12) YEARS,
       TRUNC(MOD(MONTHS_BETWEEN(SYSDATE, hire_date), 12))   MONTHS
FROM employees
ORDER BY years DESC, MONTHS desc;
```

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

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

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

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

Additional Practices: Solutions (continued)

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

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

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

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

```
SELECT COUNT(*)
FROM   employees
WHERE  last_name LIKE '%n';
--or
SELECT COUNT(*)
FROM   employees
WHERE  SUBSTR(last_name, -1) = 'n';
```

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

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

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

```
SELECT DISTINCT job_id
FROM   employees
WHERE  department_id IN (10, 20);
```

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

```
SELECT e.job_id, count(e.job_id) FREQUENCY
FROM    employees e JOIN departments d
ON  e.department_id = d.department_id
WHERE    d.department_name IN ('Administration', 'Executive')
GROUP BY e.job_id
ORDER BY FREQUENCY DESC;
```

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

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

Additional Practices: Solutions (continued)

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

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

```
SELECT last_name, salary, TRUNC(salary, -3)/1000  Thousands
FROM   employees;
```

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

```
SELECT e.last_name, m.last_name manager, m.salary,  j.grade_level
FROM   employees e JOIN employees m
ON     e.manager_id = m.employee_id
JOIN   job_grades j
ON     m.salary BETWEEN j.lowest_sal AND j.highest_sal
AND    m.salary > 15000;
```

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

```
SELECT  d.department_id, d.department_name,
        count(e1.employee_id) employees,
        NVL(TO_CHAR(AVG(e1.salary), '99999.99'), 'No average' ) avg_sal,
        e2.last_name, e2.salary, e2.job_id
FROM    departments d RIGHT OUTER JOIN employees e1
ON      d.department_id = e1.department_id
RIGHT OUTER JOIN  employees e2
ON    d.department_id = e2.department_id
GROUP BY d.department_id, d.department_name, e2.last_name, e2.salary,
         e2.job_id
ORDER BY d.department_id, employees;
```

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

```
SELECT department_id, MIN(salary)
FROM   employees
GROUP BY department_id
HAVING AVG(salary) = (SELECT MAX(AVG(salary))
                      FROM   employees
                      GROUP BY department_id);
```

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

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

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

18. Create the following statistical reports for the HR department: Include the department number, department name, and the number of employees working in each department that:

    a. Employs fewer than three employees:

```
SELECT d.department_id, d.department_name, COUNT(*)
FROM   departments d JOIN employees e
ON     d.department_id = e.department_id
GROUP BY d.department_id, d.department_name
HAVING COUNT(*) < 3;
```

    b. Has the highest number of employees:

```
SELECT d.department_id, d.department_name, COUNT(*)
FROM   departments d JOIN employees e
ON     d.department_id = e.department_id
GROUP BY d.department_id, d.department_name
HAVING COUNT(*) = (SELECT MAX(COUNT(*))
                   FROM   employees
                   GROUP BY department_id);
```

    c. Has the lowest number of employees:

```
SELECT d.department_id, d.department_name, COUNT(*)
FROM   departments d JOIN employees e
ON     d.department_id = e.department_id
GROUP BY d.department_id, d.department_name
HAVING COUNT(*) = (SELECT MIN(COUNT(*))
                   FROM   employees
                   GROUP BY department_id);
```

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

```
SELECT e.employee_id, e.last_name, e.department_id, e.salary,
AVG(s.salary)
FROM   employees e JOIN employees s
ON     e.department_id = s.department_id
GROUP BY e.employee_id, e.last_name, e.department_id, e.salary;
```

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

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

Additional Practices: Solutions (continued)

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

```
SELECT last_name, TO_CHAR(hire_date, 'Month DD') BIRTHDAY
FROM   employees
ORDER BY TO_CHAR(hire_date, 'DDD');
```

Additional Practices: Case Study Solutions

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

   a. Table name: MEMBER

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

   b. Table name: TITLE

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

   c. Table name: TITLE_COPY

```
CREATE TABLE title_copy
       (copy_id       NUMBER(10),
       title_id      NUMBER(10)
         CONSTRAINT title_copy_title_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

Additional Practices: Case Study Solutions (continued)

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

e. Table name: RESERVATION

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

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

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

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

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

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

```
CREATE SEQUENCE member_id_seq
START WITH 101
NOCACHE;
```

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

```
CREATE SEQUENCE title_id_seq
START WITH 92
NOCACHE;
```

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

```
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 the commands in the script. Be sure to use the sequence to add the member numbers.

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

Additional Practices: Case Study Solutions (continued)

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

c. Add the following movie copies in the `TITLE_COPY` table:
Note: Have the `TITLE_ID` numbers available for this exercise.

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

Additional Practices: Case Study Solutions (continued)

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

d. Add the following rentals to the RENTAL table:
Note: The title number may be different depending on the sequence number.

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

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

Note: Your results may be different.

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

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

6. Make changes to the data in the tables.

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

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

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

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

7. Make a modification to one of the tables.

   a. Run the script in `lab_apcs_7a.sql` to add a `PRICE` column to the `TITLE` table to record the purchase price of the video. Verify your modifications.

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

DESCRIBE title
```

   b. Create a script named `lab_apcs_7b.sql` that contains `UPDATE` statements that update each video with a price according to the list provided. Run the commands in the script.
   Note: Have the `TITLE_ID` numbers available for this exercise.

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

Additional Practices: Case Study Solutions (continued)

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

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

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

SET VERIFY ON
SET ECHO ON
```