A:
Practice Solutions

Solutions for Practice 1: Retrieving Data Using the SQL SELECT Statement

Part 1

Test your knowledge:

1. The following SELECT statement executes successfully:

```
SELECT last_name, job_id, salary AS Sal
FROM   employees;
```

True/False

2. The following SELECT statement executes successfully:

```
SELECT *
FROM   job_grades;
```

True/False

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

```
SELECT     employee_id, last_name
sal x 12   ANNUAL SALARY
FROM       employees;
```

- **The EMPLOYEES table does not contain a column called sal. The column is called SALARY.**
- **The multiplication operator is *, not x, as shown in line 2.**
- **The ANNUAL SALARY alias cannot include spaces. The alias should read ANNUAL_SALARY or should be enclosed within double quotation marks.**
- **A comma is missing after the LAST_NAME column.**

Part 2

Note the following points before you begin with the practices:

- Save all your lab files at the following location: *D:\labs\SQL1\labs*.
- Enter your SQL statements in a SQL Worksheet. To save a script in SQL Developer, from the File menu, select Save As or right-click in the SQL Worksheet and select Save file to save your SQL statement as a `lab_<lessonno>_<stepno>.sql` script. To modify an existing script, use File > Open to open the script file, make changes and then make sure you use Save As to save it with a different filename.
- To run the query, click the Execute Statement icon (or press [F9]) in the SQL Worksheet. For DML and DDL statements, click the Run Script icon (or press [F5]).
- After you have executed a saved script, make sure that you do not enter your next query in the same worksheet. Open a new worksheet.

Solutions for Practice 1: Retrieving Data Using the SQL SELECT Statement (continued)

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

4. Your first task is to determine the structure of the DEPARTMENTS table and its contents.

   a. To determine the DEPARTMENTS table structure:

```
DESCRIBE departments
```

   b. To view the data contained by the DEPARTMENTS table:

```
SELECT *
FROM   departments;
```

5. You need to determine the structure of the EMPLOYEES table.

```
DESCRIBE employees
```

   The HR department wants a query to display the last name, job ID, hire date, and employee ID for each employee, with the employee ID appearing first. Provide an alias STARTDATE for the HIRE_DATE column. Save your SQL statement to a file named lab_01_05.sql so that you can dispatch this file to the HR department.

```
SELECT employee_id, last_name, job_id, hire_date StartDate
FROM   employees;
```

6. Test your query in the file lab_01_05.sql to ensure that it runs correctly.

```
SELECT employee_id, last_name, job_id, hire_date StartDate
FROM   employees;
```

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

```
SELECT DISTINCT job_id
FROM   employees;
```

Solutions for Practice 1: Retrieving Data Using the SQL SELECT Statement (continued)

Part 3

If you have the time, complete the following exercises:

8. The HR department wants more descriptive column headings for its report on employees. Copy the statement from `lab_01_05.sql` to a new SQL Worksheet. Name the column headings `Emp #`, `Employee`, `Job`, and `Hire Date`, respectively. Then run your query again.

```
SELECT employee_id "Emp #", last_name "Employee",
       job_id "Job", hire_date "Hire Date"
FROM   employees;
```

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

```
SELECT last_name||', '||job_id "Employee and Title"
FROM   employees;
```

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

10. To familiarize yourself with the data in the EMPLOYEES table, create a query to display all the data from the EMPLOYEES table. Separate each column output with a comma. Name the column as THE_OUTPUT.

```
SELECT employee_id || ',' || first_name || ',' || last_name
       || ',' || email || ',' || phone_number || ','|| job_id
       || ',' || manager_id || ',' || hire_date || ','
       || salary || ',' || commission_pct || ',' || department_id
       THE_OUTPUT
FROM   employees;
```

Solutions for Practice 2: Restricting and Sorting Data

The HR department needs your assistance in creating some queries.

1. Because of budget issues, the HR department needs a report that displays the last name and salary of employees earning more than $12,000. Save your SQL statement as a file named `lab_02_01.sql`. Run your query.

```
SELECT  last_name, salary
FROM    employees
WHERE   salary > 12000;
```

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

```
SELECT  last_name, department_id
FROM    employees
WHERE   employee_id = 176;
```

3. The HR departments needs to find employees with high salary and low salary. Modify `lab_02_01.sql` to display the last name and salary for all employees whose salary is not in the range of $5,000 to $12,000. Save your SQL statement as `lab_02_03.sql`.

```
SELECT  last_name, salary
FROM    employees
WHERE   salary NOT BETWEEN 5000 AND 12000;
```

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

```
SELECT   last_name, job_id, hire_date
FROM     employees
WHERE    last_name IN ('Matos', 'Taylor')
ORDER BY hire_date;
```

5. Display the last name and department ID of all employees in departments 20 or 50 in ascending alphabetical order by name.

```
SELECT   last_name, department_id
FROM     employees
WHERE    department_id IN (20, 50)
ORDER BY last_name ASC;
```

Solutions for Practice 2: Restricting and Sorting Data (continued)

6. Modify `lab_02_03.sql` to list the last name and salary of employees who earn between $5,000 and $12,000, and are in department 20 or 50. Label the columns `Employee` and `Monthly Salary`, respectively. Resave `lab_02_03.sql` as `lab_02_06.sql`. Run the statement in `lab_02_06.sql`.

```
SELECT    last_name "Employee", salary "Monthly Salary"
FROM      employees
WHERE     salary  BETWEEN 5000 AND 12000
AND       department_id IN (20, 50);
```

7. The HR department needs a report that displays the last name and hire date for all employees who were hired in 1994.

```
SELECT    last_name, hire_date
FROM      employees
WHERE     hire_date LIKE '%94';
```

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

```
SELECT    last_name, job_id
FROM      employees
WHERE     manager_id IS NULL;
```

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

```
SELECT    last_name, salary, commission_pct
FROM      employees
WHERE     commission_pct IS NOT NULL
ORDER BY 2 DESC, 3 DESC;
```

10. Members of the HR department want to have more flexibility with the queries that you are writing. They would like a report that displays the last name and salary of employees who earn more than an amount that the user specifies after a prompt. (You can use the query created in practice exercise 1 and modify it.) Save this query to a file named `lab_02_10.sql`.

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

```
SELECT   last_name, salary
FROM     employees
WHERE    salary > &sal_amt;
```

Solutions for Practice 2: Restricting and Sorting Data (continued)

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

    manager _id = 103, sorted by last_name

    manager_id = 201, sorted by salary

    manager_id = 124, sorted by employee_id

```
SELECT employee_id, last_name, salary, department_id
FROM employees
WHERE manager_id = &mgr_num
ORDER BY &order_col;
```

If you have the time, complete the following exercises:

12. Display all employee last names in which the third letter of the name is "a."

```
SELECT    last_name
FROM      employees
WHERE     last_name LIKE '__a%';
```

13. Display the last names of all employees who have both an "a" and an "e" in their last name.

```
SELECT    last_name
FROM      employees
WHERE     last_name LIKE '%a%'
AND       last_name LIKE '%e%';
```

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

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

```
SELECT    last_name, job_id, salary
FROM      employees
WHERE     job_id IN ('SA_REP', 'ST_CLERK')
AND       salary NOT IN (2500, 3500, 7000);
```

15. Modify `lab_02_06.sql` to display the last name, salary, and commission for all employees whose commission amount is 20%. Resave `lab_02_06.sql` as `lab_02_15.sql`. Rerun the statement in `lab_02_15.sql`.

```
SELECT    last_name "Employee", salary "Monthly Salary",
          commission_pct
FROM      employees
WHERE     commission_pct = .20;
```

Solutions for Practice 8: Using the Set Operators

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

```
SELECT department_id
FROM   departments
MINUS
SELECT department_id
FROM   employees
WHERE  job_id = 'ST_CLERK';
```

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

```
SELECT country_id,country_name
FROM countries
MINUS
SELECT l.country_id,c.country_name
FROM locations l JOIN countries c
ON (l.country_id = c.country_id)
JOIN departments d
ON d.location_id=l.location_id;
```

3. Produce a list of jobs for departments 10, 50, and 20, in that order. Display job ID and department ID using the set operators.

```
SELECT distinct job_id, department_id
FROM employees
WHERE department_id = 10
UNION ALL
SELECT DISTINCT job_id, department_id
FROM employees
WHERE department_id = 50
UNION ALL
SELECT DISTINCT job_id, department_id
FROM employees
WHERE department_id = 20
```

Solutions for Practice 8: Using the Set Operators (continued)

4. Create a report that lists the employee IDs and job IDs of those employees who currently have a job title that is the same as their job title when they were initially hired by the company (that is, they changed jobs but have now gone back to doing their original job).

```
SELECT     employee_id,job_id
FROM       employees
INTERSECT
SELECT    employee_id,job_id
FROM      job_history;
```

5. The HR department needs a report with the following specifications:

   - Last name and department ID of all the employees from the EMPLOYEES table, regardless of whether or not they belong to a department
   - Department ID and department name of all the departments from the DEPARTMENTS table, regardless of whether or not they have employees working in them

   Write a compound query to accomplish this.

```
SELECT last_name,department_id,TO_CHAR(null)
FROM   employees
UNION
SELECT TO_CHAR(null),department_id,department_name
FROM  departments;
```

Solutions for Practice 9: Manipulating Data

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

Note: For all the DML statements, click the Run Script icon (or press [F5]) to execute the query. This way you get to see the feedback messages on the Script Output tabbed page. For SELECT queries, continue to click the Execute Statement icon or press [F9] to get the formatted output on the Results tabbed page.

Insert data into the MY_EMPLOYEE table.

1. Run the statement in the lab_09_01.sql script to build the MY_EMPLOYEE table used in this practice.

    a. From File menu, select Open. In the Open dialog box, navigate to D:\labs\sql1\labs folder, double-click lab_09_01.sql.

    b. After the statement is opened in a SQL Worksheet, click the Run Script icon to run the script. You get a Create Table succeeeded message on the Script Output tabbed page.

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

```
DESCRIBE my_employee
```

3. Create an INSERT statement to add *the first row* of data to the MY_EMPLOYEE table from the following sample data. Do not list the columns in the INSERT clause.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860
3	Biri	Ben	bbiri	1100
4	Newman	Chad	cnewman	750
5	Ropeburn	Audrey	aropebur	1550

```
INSERT INTO my_employee
  VALUES (1, 'Patel', 'Ralph', 'rpatel', 895);
```

Solutions for Practice 9: Manipulating Data (continued)

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

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

5. Confirm your additions to the table.

```
SELECT    *
FROM      my_employee;
```

6. Write an insert statement in a dynamic reusable script file to load the remaining rows into the MY_EMPLOYEE table. The script should prompt for all the columns (ID, LAST_NAME, FIRST_NAME, USERID and SALARY). Save this script to a file named lab_09_06.sql.

```
INSERT INTO my_employee
VALUES (&p_id, '&p_last_name', '&p_first_name',
        '&p_userid', &p_salary);
```

7. Populate the table with the next two rows of sample data listed in step 3 by running the INSERT statement in the script that you created.

```
INSERT INTO my_employee
VALUES (&p_id, '&p_last_name', '&p_first_name',
        '&p_userid', &p_salary);
```

8. Confirm your additions to the table.

```
SELECT    *
FROM my_employee;
```

9. Make the data additions permanent.

```
COMMIT;
```

Solutions for Practice 11: Creating Other Schema Objects (continued)
Part 2

7. You need a sequence that can be used with the primary key column of the DEPT table. The sequence should start at 200 and have a maximum value of 1,000. Have your sequence increment by 10. Name the sequence DEPT_ID_SEQ.

```
CREATE SEQUENCE dept_id_seq
  START WITH 200
  INCREMENT BY 10
  MAXVALUE 1000;
```

8. To test your sequence, write a script to insert two rows in the DEPT table. Name your script lab_11_08.sql. Be sure to use the sequence that you created for the ID column. Add two departments: Education and Administration. Confirm your additions. Run the commands in your script.

```
INSERT INTO dept
VALUES (dept_id_seq.nextval, 'Education');

INSERT INTO dept
VALUES (dept_id_seq.nextval, 'Administration');
```

9. Create a nonunique index on the NAME column in the DEPT table.

```
CREATE INDEX dept_name_idx ON dept (name);
```

10. Create a synonym for your EMPLOYEES table. Call it EMP.

```
CREATE SYNONYM emp FOR EMPLOYEES;
```

Solutions for Practice C: Oracle Join Syntax

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

```
SELECT location_id, street_address, city, state_province, country_name
FROM   locations, countries
WHERE  locations.country_id = countries.country_id;
```

2. The HR department needs a report of all employees. Write a query to display the last name, department number, and department name for all employees. Run the query.

```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e, departments d
WHERE  e.department_id = d.department_id;
```

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

```
SELECT e.last_name, e.job_id, e.department_id, d.department_name
FROM   employees e, departments d , locations l
WHERE  e.department_id = d.department_id
AND    d.location_id = l.location_id
AND    LOWER(l.city) = 'toronto';
```

4. Create a report to display the employee last name and the employee number along with the last name of the employee's manager and manager number. Label the columns Employee, Emp#, Manager, and Mgr#, respectively. Save your SQL statement as lab_c_04.sql.

```
SELECT w.last_name "Employee", w.employee_id "EMP#",
       m.last_name "Manager", m.employee_id  "Mgr#"
FROM   employees w, employees m
WHERE  w.manager_id = m.employee_id;
```

5. Modify lab_c_04.sql to display all employees including King, who has no manager. Order the results by the employee number. Save the SQL statement as lab_c_05.sql. Run the query in lab_c_05.sql.

```
SELECT w.last_name "Employee", w.employee_id "EMP#",
       m.last_name "Manager", m.employee_id  "Mgr#"
FROM   employees w, employees m
WHERE  w.manager_id  = m.employee_id (+);
```

Solutions for Practice C: Oracle Join Syntax (continued)

6. Create a report for the HR department that displays employee last names, department numbers, and all the employees who work in the same department as a given employee. Give each column an appropriate label. Save the script to a file named `lab_c_06.sql`.

```
SELECT e1.department_id department, e1.last_name employee,
       e2.last_name colleague
FROM   employees e1, employees e2
WHERE  e1.department_id = e2.department_id
AND    e1.employee_id <> e2.employee_id
ORDER BY e1.department_id, e1.last_name, e2.last_name;
```

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

```
DESC JOB_GRADES

SELECT e.last_name, e.job_id, d.department_name,
       e.salary, j.grade_level
FROM   employees e, departments d, job_grades j
WHERE  e.department_id = d.department_id
AND    e.salary BETWEEN j.lowest_sal AND j.highest_sal;
```

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

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

```
SELECT e.last_name, e.hire_date
FROM   employees e , employees davies
WHERE  davies.last_name = 'Davies'
AND    davies.hire_date < e.hire_date;
```

9. The HR department needs to find the names and hire dates for all employees who were hired before their managers, along with their managers' names and hire dates. Label the columns `Employee`, `Emp Hired`, `Manager`, and `Mgr Hired`, respectively. Save the script to a file named `lab_c_09.sql`.

```
SELECT w.last_name, w.hire_date, m.last_name, m.hire_date
FROM   employees w , employees m
WHERE  w.manager_id = m.employee_id
AND    w.hire_date <  m.hire_date;
```