Chapter 8 Solutions

Question

Question 1

Run the statement in the lab8_1.sql script to build the MY_EMPLOYEE table to be used for the lab.

Solution

```
-- To run the script in Oracle SQL*Plus:
@lab8_1.sql

-- Or in other SQL environments:
-- SOURCE lab8_1.sql
```

Question

Question 2

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

Name	**Null?**	**Type**
ID	NOT NULL	NUMBER(4)
LAST_NAME		VARCHAR2(25)
FIRST_NAME		VARCHAR2(25)
USERID		VARCHAR2(8)
SALARY		NUMBER(9,2)

Solution

```
-- In Oracle:
DESCRIBE MY_EMPLOYEE;

-- In other SQL environments:
-- SHOW COLUMNS FROM MY_EMPLOYEE;
```

Question

Question 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	895
2	Dancs	Betty	bdancs	860
3	Biri	Ben	bbiri	1100
4	Newman	Chad	cnewman	750
5	Ropeburn	Audrey	aropebur	1550

Solution

```sql
INSERT INTO MY_EMPLOYEE
VALUES (1, ’Patel’, ’Ralph’, ’rpatel’, 895);
```

Question

Question 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.

Solution

```sql
INSERT INTO MY_EMPLOYEE (ID, LAST_NAME, FIRST_NAME, USERID, SALARY)
VALUES (2, ’Dancs’, ’Betty’, ’bdancs’, 860);
```

Question

Question 5

Confirm your addition to the table.

Solution

```sql
SELECT * FROM MY_EMPLOYEE;
```

Question

Question 6

Write an INSERT statement in a text file named loademp.sql to load rows into the MY_EMPLOYEE table. Concatenate the first letter of the first name and the first seven characters of the last name to produce the user ID.

Solution

```sql
-- File: loademp.sql
-- This script loads rows 3 and 4 into the MY_EMPLOYEE table

INSERT INTO MY_EMPLOYEE (ID, LAST_NAME, FIRST_NAME, USERID, SALARY)
VALUES (3, 'Biri', 'Ben',
        SUBSTR(FIRST_NAME,1,1) || SUBSTR(LAST_NAME,1,7),
        1100);

INSERT INTO MY_EMPLOYEE (ID, LAST_NAME, FIRST_NAME, USERID, SALARY)
VALUES (4, 'Newman', 'Chad',
        SUBSTR(FIRST_NAME,1,1) || SUBSTR(LAST_NAME,1,7),
        750);
```

Question

Question 7

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

Solution

```
-- To run the script in Oracle SQL*Plus:
@loademp.sql

-- Or in other SQL environments:
-- SOURCE loademp.sql
```

Question

Question 8

Confirm your additions to the table.

Solution

```
SELECT * FROM MY_EMPLOYEE;
```

This will display the table with 4 rows, matching the table shown in the question.

Question

Question 9

Make the data additions permanent.

Solution

```
COMMIT;
```

Question

Question 10

Change the last name of employee 3 to Drexler.

Solution

```sql
UPDATE MY_EMPLOYEE
SET LAST_NAME = ’Drexler’
WHERE ID = 3;
```

Question

Question 11

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

Solution

```sql
UPDATE MY_EMPLOYEE
SET SALARY = 1000
WHERE SALARY < 900;
```

Question

Question 12

Verify your changes to the table.

Solution

```sql
SELECT * FROM MY_EMPLOYEE;
```

Question

Question 13

Delete Betty Dancs from the MY_EMPLOYEE table.

Solution

```
DELETE FROM MY_EMPLOYEE
WHERE LAST_NAME = 'Dancs' AND FIRST_NAME = 'Betty';

-- Alternative way using ID:
-- DELETE FROM MY_EMPLOYEE WHERE ID = 2;
```

Question

Question 14

Confirm your changes to the table.

Solution

```
SELECT * FROM MY_EMPLOYEE;
```

This will display the final table with Betty Dancs removed, matching the final table shown in the question.

Question

Question 15

Commit all pending changes.

Solution

```
COMMIT;
```

Question

Question 16

Populate the table with the last row of sample data by modifying the statements in the script that you created in step 6. Run the statements in the script.

Solution

First, we need to modify the loademp.sql script to insert the last row of sample data:

```
-- Modified loademp.sql file
INSERT INTO MY_EMPLOYEE (ID, LAST_NAME, FIRST_NAME, USERID, SALARY)
VALUES (5, 'Ropeburn', 'Audrey',
        SUBSTR(FIRST_NAME,1,1) || SUBSTR(LAST_NAME,1,7),
        1550);
```

Then run the script:

```
-- To run the script in Oracle SQL*Plus:
@loademp.sql

-- Or in other SQL environments:
-- SOURCE loademp.sql
```

Question

Question 17

Confirm your addition to the table.

Solution

```
SELECT * FROM MY_EMPLOYEE;
```

Question

Question 18

Mark an intermediate point in the processing of the transaction.

Solution

```
SAVEPOINT before_empty;
```

Question

Question 19

Empty the entire table.

Solution

```
DELETE FROM MY_EMPLOYEE;
```

Question

Question 20

Confirm that the table is empty.

Solution

```
SELECT * FROM MY_EMPLOYEE;
```

Question

Question 21

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

Solution

```
ROLLBACK TO SAVEPOINT before_empty;
```

This rolls back the transaction to the savepoint we created before emptying the table, which undoes the DELETE operation but keeps the INSERT operation that added row 5.

Question

Question 22

Confirm that the new row is still intact.

Solution

```
SELECT * FROM MY_EMPLOYEE;
```

Question

Question 23

Make the data addition permanent.

Solution

```
COMMIT;
```

Chapter 9 Solutions

Question

Question 1

Create the DEPT table based on the following table instance chart. Place the syntax in a script called lab9_1.sql, then execute the statement in the script to create the table. Confirm that the table is created.

Column Name	ID	NAME
Key Type		
Nulls/Unique		
FK Table		
FK Column		
Data type	NUMBER	VARCHAR2
Length	7	25

Here's the description of the created table:

Name	**Null?**	**Type**
ID		NUMBER(7)
NAME		VARCHAR2(25)

Solution

```sql
-- File: lab9_1.sql
-- Create DEPT table

CREATE TABLE DEPT (
    ID NUMBER(7),
    NAME VARCHAR2(25)
);

-- To verify the table was created:
DESCRIBE DEPT;
```

Listing 1: lab9_1.sql

To execute the script:

```sql
-- In Oracle SQL*Plus:
@lab9_1.sql

-- Or in other SQL environments:
-- SOURCE lab9_1.sql
```

Question

Question 2

Populate the DEPT table with data from the DEPARTMENTS table. Include only columns that you need.

Solution

```sql
-- Assuming DEPARTMENTS table has columns DEPARTMENT_ID and DEPARTMENT_NAME
-- that correspond to the ID and NAME columns in our DEPT table

INSERT INTO DEPT (ID, NAME)
SELECT DEPARTMENT_ID, DEPARTMENT_NAME
FROM DEPARTMENTS;

-- Verify the data was inserted:
SELECT * FROM DEPT;
```

Question

Question 3

Create the EMP table based on the following table instance chart. Place the syntax in a script called lab9_3.sql, and then execute the statement in the script to create the table. Confirm that the table is created.

Column Name	ID	LAST_NAME	FIRST_NAME	DEPT_ID
Key Type				
Nulls/Unique				
FK Table				
FK Column				
Data type	NUMBER	VARCHAR2	VARCHAR2	NUMBER
Length	7	25	25	7

Here's the description of the created table:

Name	Null?	Type
ID		NUMBER(7)
LAST_NAME		VARCHAR2(25)
FIRST_NAME		VARCHAR2(25)
DEPT_ID		NUMBER(7)

Solution

```sql
-- File: lab9_3.sql
-- Create EMP table

CREATE TABLE EMP (
    ID NUMBER(7),
    LAST_NAME VARCHAR2(25),
    FIRST_NAME VARCHAR2(25),
    DEPT_ID NUMBER(7)
);

-- To verify the table was created:
DESCRIBE EMP;
```

Listing 2: lab9_3.sql

To execute the script:

```sql
-- In Oracle SQL*Plus:
@lab9_3.sql

-- Or in other SQL environments:
-- SOURCE lab9_3.sql
```

Question

Question 4

Modify the EMP table to allow for longer employee last names. Confirm your modification.

Name	Null?	Type
ID		NUMBER(7)
LAST_NAME		VARCHAR2(50)
FIRST_NAME		VARCHAR2(25)
DEPT_ID		NUMBER(7)

Solution

```sql
ALTER TABLE EMP
MODIFY (LAST_NAME VARCHAR2(50));

-- To verify the modification:
DESCRIBE EMP;
```

Question

Question 5

Confirm that both the DEPT and EMP tables are stored in the data dictionary. (Hint: USER_TABLES)

TABLE_NAME
DEPT
EMP

Solution

```sql
SELECT TABLE_NAME
FROM USER_TABLES
WHERE TABLE_NAME IN ('DEPT', 'EMP')
ORDER BY TABLE_NAME;
```

Question

Question 6

Create the EMPLOYEES2 table based on the structure of the EMPLOYEES table. Include only the EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY, and DEPARTMENT_ID columns. Name the columns in your new table ID, FIRST_NAME, LAST_NAME, SALARY, and DEPT_ID, respectively.

Solution

```sql
CREATE TABLE EMPLOYEES2 AS
SELECT EMPLOYEE_ID AS ID,
       FIRST_NAME,
       LAST_NAME,
       SALARY,
       DEPARTMENT_ID AS DEPT_ID
FROM EMPLOYEES;

-- To verify the table creation:
DESCRIBE EMPLOYEES2;
```

Question

Question 7

Drop the EMP table.

Solution

```
DROP TABLE EMP;

-- Verify the table is dropped:
SELECT TABLE_NAME
FROM USER_TABLES
WHERE TABLE_NAME = ’EMP’;
-- Should return no rows
```

Question

Question 8

Rename the EMPLOYEES2 table as EMP.

Solution

```
RENAME EMPLOYEES2 TO EMP;

-- Verify the rename:
SELECT TABLE_NAME
FROM USER_TABLES
WHERE TABLE_NAME = ’EMP’;
```

Question

Question 9

Add a comment to the DEPT and EMP table definitions describing the tables. Confirm your additions in the data dictionary.

Solution

```sql
COMMENT ON TABLE DEPT IS 'Department table storing department information';
COMMENT ON TABLE EMP IS 'Employee table storing employee information';

-- Verify the comments:
SELECT TABLE_NAME, COMMENTS
FROM USER_TAB_COMMENTS
WHERE TABLE_NAME IN ('DEPT', 'EMP');
```

Question

Question 10

Drop the FIRST_NAME column from the EMP table. Confirm your modification by checking the description of the table.

Solution

```sql
ALTER TABLE EMP
DROP COLUMN FIRST_NAME;

-- Verify the modification:
DESCRIBE EMP;
```

Question

Question 11

In the EMP table, mark the DEPT_ID column in the EMP table as UNUSED. Confirm your modification by checking the description of the table.

Solution

```sql
ALTER TABLE EMP
SET UNUSED COLUMN DEPT_ID;

-- Verify the modification:
DESCRIBE EMP;
```

Question

Question 12

Drop all the UNUSED columns from the EMP table. Confirm your modification by checking the description of the table.

Solution

```sql
ALTER TABLE EMP
DROP UNUSED COLUMNS;

-- Verify the modification:
DESCRIBE EMP;
```

Chapter 10 Solutions

Question

Question 1

Add a table-level PRIMARY KEY constraint to the EMP table on the ID column. The constraint should be named at creation. Name the constraint my_emp_id_pk.

Hint: The constraint is enabled as soon as the ALTER TABLE command executes successfully.

Solution

```
ALTER TABLE EMP
ADD CONSTRAINT my_emp_id_pk PRIMARY KEY (ID);

-- To verify the constraint was added:
SELECT constraint_name, constraint_type
FROM user_constraints
WHERE table_name = 'EMP';
```

Question

Question 2

Create a PRIMARY KEY constraint to the DEPT table using the ID column. The constraint should be named at creation. Name the constraint my_deptid_pk.

Hint: The constraint is enabled as soon as the ALTER TABLE command executes successfully.

Solution

```
ALTER TABLE DEPT
ADD CONSTRAINT my_deptid_pk PRIMARY KEY (ID);

-- To verify the constraint was added:
SELECT constraint_name, constraint_type
FROM user_constraints
WHERE table_name = 'DEPT';
```

Question

Question 3

Add a column DEPT_ID to the EMP table. Add a foreign key reference on the EMP table that ensures that the employee is not assigned to a nonexistent department. Name the constraint my_emp_dept_id_fk.

Solution

```
-- First, add the DEPT_ID column to the EMP table
ALTER TABLE EMP
ADD DEPT_ID NUMBER(7);

-- Then, add the foreign key constraint
ALTER TABLE EMP
ADD CONSTRAINT my_emp_dept_id_fk
FOREIGN KEY (DEPT_ID) REFERENCES DEPT(ID);

-- To verify the column and constraint were added:
DESCRIBE EMP;
SELECT constraint_name, constraint_type, r_constraint_name
FROM user_constraints
WHERE table_name = 'EMP';
```

Question

Question 4

Confirm that the constraints were added by querying the USER_CONSTRAINTS view. Note the types and names of the constraints. Save your statement text in a file called lab10_4.sql.

CONSTRAINT_NAME	C
MY_DEPTID_PK	P
SYS_C002541	C
MY_EMP_ID_PK	P
MY_EMP_DEPT_ID_FK	R

Solution

```
-- File: lab10_4.sql
-- Query to display constraint information

SELECT constraint_name,
       DECODE(constraint_type,
              'P', 'P',
              'R', 'R',
              'C', 'C',
              'U', 'U',
              constraint_type) AS C
FROM user_constraints
WHERE table_name IN ('EMP', 'DEPT')
ORDER BY constraint_name;
```

Listing 3: lab10_4.sql

Question

Question 5

Display the object names and types from the USER_OBJECTS data dictionary view for the EMP and DEPT tables. Notice that the new tables and a new index were created.

Solution

```
SELECT object_name, object_type
FROM user_objects
WHERE object_name IN ('EMP', 'DEPT')
   OR object_name LIKE '%EMP%'
   OR object_name LIKE '%DEPT%'
ORDER BY object_type, object_name;
```

Question

Question 6 (Optional Exercise)

Modify the EMP table. Add a COMMISSION column of NUMBER data type, precision 2, scale 2. Add a constraint to the commission column that ensures that a commission value is greater than zero.

Solution

```
-- Add the COMMISSION column
ALTER TABLE EMP
ADD COMMISSION NUMBER(2,2);

-- Add a check constraint to ensure commission is greater than zero
ALTER TABLE EMP
ADD CONSTRAINT emp_comm_chk CHECK (COMMISSION > 0);

-- Verify the column and constraint
DESCRIBE EMP;
SELECT constraint_name, constraint_type, search_condition
FROM user_constraints
WHERE table_name = 'EMP' AND constraint_name = 'EMP_COMM_CHK';
```

Chapter 18 Solutions

Question

Question 1

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

Solution

```sql
SELECT last_name, department_id, salary
FROM employees
WHERE (department_id, salary) IN
    (SELECT department_id, salary
     FROM employees
     WHERE commission_pct IS NOT NULL);
```

Question

Question 2

Display the last name, department name, and salary of any employee whose salary and commission match the salary and commission of any employee located in location ID 1700.

Solution

```sql
SELECT e.last_name, d.department_name, e.salary
FROM employees e
JOIN departments d ON e.department_id = d.department_id
WHERE (e.salary, NVL(e.commission_pct, 0)) IN
    (SELECT salary, NVL(commission_pct, 0)
     FROM employees e
     JOIN departments d ON e.department_id = d.department_id
     WHERE d.location_id = 1700);
```

Question

Question 3

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

Solution

```sql
SELECT last_name, hire_date, salary
FROM employees
WHERE (salary, NVL(commission_pct, 0)) =
    (SELECT salary, NVL(commission_pct, 0)
     FROM employees
     WHERE last_name = 'Kochhar')
AND last_name != 'Kochhar';
```

Question

Question 4

Create a query to display the employees who earn a salary that is higher than the salary of all of the sales managers (job_id = 'SA_MAN'). Sort the results on salary from highest to lowest.

Solution

```sql
SELECT last_name, job_id, salary
FROM employees
WHERE salary > ALL
    (SELECT salary
     FROM employees
     WHERE job_id = 'SA_MAN')
ORDER BY salary DESC;
```

Question

Question 5

Display the details of the employee IDs, last names, and department IDs of those employees who live in cities whose name begins with T.

Solution

```
SELECT employee_id, last_name, department_id
FROM employees e
JOIN departments d ON e.department_id = d.department_id
JOIN locations l ON d.location_id = l.location_id
WHERE UPPER(l.city) LIKE ’T%’;
```

Question

Question 6

Write a query to find all employees who earn more than the average salary in their departments. Display last name, salary, department ID, and the average salary for the department. Sort by average salary. Use the WITH clause to create the complex query.

Solution

```
WITH dept_avg AS (
    SELECT department_id, AVG(salary) as avg_salary
    FROM employees
    GROUP BY department_id
)
SELECT e.last_name, e.salary, e.department_id, d.avg_salary
FROM employees e
JOIN dept_avg d ON e.department_id = d.department_id
WHERE e.salary > d.avg_salary
ORDER BY d.avg_salary;
```

Question

Question 7

Find all employees who are not supervisors.

Solution

```sql
SELECT e.last_name
FROM employees e
WHERE e.employee_id NOT IN (
    SELECT DISTINCT manager_id
    FROM employees
    WHERE manager_id IS NOT NULL
);
```

Question

Question 7.a

Find all this by using the NOT EXISTS operator.

Solution

```sql
SELECT e.last_name
FROM employees e
WHERE NOT EXISTS (
    SELECT 1
    FROM employees s
    WHERE s.manager_id = e.employee_id
);
```

Question

Question 7.b

Can this be done by using the ANY operator? How, or why not?

Solution

Yes, this can be done using the ANY operator, but with a double negative logic:

```sql
SELECT e.last_name
FROM employees e
WHERE e.employee_id != ANY (
    SELECT DISTINCT manager_id
    FROM employees
    WHERE manager_id IS NOT NULL
);
```

A more appropriate way would be:

```sql
SELECT e.last_name
FROM employees e
WHERE NOT (e.employee_id = ANY (
    SELECT DISTINCT manager_id
    FROM employees
    WHERE manager_id IS NOT NULL
));
```

Question

Question 8

Write a query to display the last names of the employees who earn less than the average salary in their departments.

LAST_NAME
Kochhar
De Haan
Ernst
Lorentz
Davies
Matos
Vargas
Taylor
Fay
Gietz

Solution

```sql
SELECT LAST_NAME
FROM EMPLOYEES E
WHERE SALARY < (
SELECT AVG(SALARY)
FROM EMPLOYEES
WHERE DEPARTMENT_ID = E.DEPARTMENT_ID
);
```

Question

Question 9

Write a query to display the last names of the employees who have one or more coworkers in their departments with later hire dates but higher salaries.

LAST_NAME
Rajs
Davies
Matos
Vargas
Taylor

Solution

```sql
SELECT E1.LAST_NAME
FROM EMPLOYEES E1
WHERE EXISTS (
SELECT 1
FROM EMPLOYEES E2
WHERE E1.DEPARTMENT_ID = E2.DEPARTMENT_ID
AND E2.HIRE_DATE > E1.HIRE_DATE
AND E2.SALARY > E1.SALARY
);
```

Question

Question 10

Write a query to display the employee ID, last names, and department names of all employees. **Note:** Use a scalar subquery to retrieve the department name in the `SELECT` statement.

EMPLOYEE_ID	LAST_NAME	DEPARTMENT
205	Higgins	Accounting
206	Gietz	Accounting
200	Whalen	Administration
100	King	Executive
101	Kochhar	Executive
102	De Haan	Executive
103	Hunold	IT
104	Ernst	IT
107	Lorentz	IT
201	Hartstein	Marketing
202	Fay	Marketing
149	Zlotkey	Sales
176	Taylor	Sales
174	Abel	Sales
124	Mourgos	Shipping
141	Rajs	Shipping
142	Davies	Shipping
143	Matos	Shipping
144	Vargas	Shipping
178	Grant	Shipping

Solution

```
SELECT EMPLOYEE_ID, LAST_NAME,
  (SELECT DEPARTMENT_NAME
   FROM DEPARTMENTS D
   WHERE D.DEPARTMENT_ID = E.DEPARTMENT_ID) AS DEPARTMENT
FROM EMPLOYEES E;
```

Question

Question 11

Write a query to display the department names of those departments whose total salary cost is above one eighth (1/8) of the total salary cost of the whole company. Use the `WITH` clause to write this query. Name the query `SUMMARY`.

DEPARTMENT_NAME	DEPT_TOTAL
Executive	58000
Sales	37100

Solution

```sql
WITH SUMMARY AS (
  SELECT D.DEPARTMENT_NAME, SUM(E.SALARY) AS DEPT_TOTAL
  FROM EMPLOYEES E
  JOIN DEPARTMENTS D ON E.DEPARTMENT_ID = D.DEPARTMENT_ID
  GROUP BY D.DEPARTMENT_NAME
)
SELECT DEPARTMENT_NAME, DEPT_TOTAL
FROM SUMMARY
WHERE DEPT_TOTAL > (
  SELECT SUM(SALARY)/8 FROM EMPLOYEES
);
```