UNIVERSITY OF CHITTAGONG

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SQL Exercises

Database Systems Exercises

Course Information

Course Code: 413

Course Title: Database Systems

Semester: 4th

Submitted By:

Md. Abdullah Al Mamun Emon
ID: 23701028
GitHub: emon4075

Submitted To:

Dr. Rudra Pratap Deb Nath
Associate Professor
Department of Computer Science & Engineering
University of Chittagong
Chittagong-4331

June 13, 2025

Table of Contents

Contents

1 Chapter - 8 - Manipulating Data

Question 1

Create a table named MY_EMPLOYEE with the following structure:

Column Name	Data Type	Constraint
ID	NUMBER(4)	NOT NULL
LAST_NAME	VARCHAR2(25)	
FIRST_NAME	VARCHAR2(25)	
USERID	VARCHAR2(8)	
SALARY	NUMBER(9,2)	

Answer 1

Command:

```
CREATE TABLE MY_EMPLOYEE(
    ID NUMBER(4) NOT NULL,
    LAST_NAME VARCHAR2(25),
    FIRST_NAME VARCHAR2(25),
    USERID VARCHAR2(8),
    SALARY NUMBER(9,2)
);
```

Question 2

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

Answer 2

Command:

```
DESCRIBE MY_EMPLOYEE;
```

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

Answer 3

Command:

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

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 SELECT clause.

Answer 4

Command:

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

Question 5

Confirm your addition to the table.

Answer 5

Command:

```
SELECT *
FROM MY_EMPLOYEE;
```

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 populate the user ID.

Answer 6

Command:

```
INSERT INTO MY_EMPLOYEE (SALARY, LAST_NAME, FIRST_NAME, USERID, ID)
VALUES (1100, 'Biri', 'Ben', 'bbiri', 3);

INSERT INTO MY_EMPLOYEE (ID, LAST_NAME, FIRST_NAME, USERID, SALARY)
VALUES (4, 'Newman', 'Chad', 'cnewman', 750);
```

Question 7

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

Answer 7

Execute the above script in VS Code Oracle SQL Developer to run the INSERT statements.

Question 8

Confirm your additions to the table.

Answer 8

Command:

```
SELECT *
FROM MY_EMPLOYEE;
```

Question 9

Make the data additions permanent.

Answer 9

Command:

```
COMMIT;
```

Question 10

Update and delete data in the MY_EMPLOYEE table. Change the last name of employee 3 to Drexler.

Answer 10

Command:

```
UPDATE MY_EMPLOYEE
SET LAST_NAME = 'Drexler'
WHERE ID = 3;
```

Question 11

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

Answer 11

Command:

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

Question 12

Verify your changes to the table.

Answer 12

Command:

```
SELECT *
FROM MY_EMPLOYEE;
```

Question 13

Delete Betty Dancs from the MY_EMPLOYEE table.

Answer 13

Command:

```
DELETE FROM MY_EMPLOYEE
WHERE ID = 3;

-- Alternative approach:
DELETE FROM MY_EMPLOYEE
WHERE FIRST_NAME = 'Betty' AND LAST_NAME = 'Dancs';
```

Question 14

Confirm your changes to the table.

Answer 14

Command:

```
SELECT *
FROM MY_EMPLOYEE;
```

Question 15

Commit all pending changes.

Answer 15

Command:

```
COMMIT;
```

Question 16

Control data transaction to the MY_EMPLOYEE table: 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.

Answer 16

Command:

```
INSERT ALL
    INTO MY_EMPLOYEE VALUES (4, 'Newman', 'Chad', 'cnewman', 750)
    INTO MY_EMPLOYEE VALUES (5, 'Ropeburn', 'Audrey', 'aropebur', 1550)
SELECT * FROM dual;
```

Question 17

Confirm your addition to the table.

Answer 17

Command:

```
SELECT *
FROM MY_EMPLOYEE
ORDER BY ID;
```

Question 18

Mark an intermediate point in the processing of the transaction.

Answer 18

Command:

```
SAVEPOINT all_inserted;
```

Question 19

Empty the entire table.

Answer 19

Command:

```
DELETE FROM MY_EMPLOYEE;
```

Question 20

Confirm that the table is empty.

Answer 20

Command:

```
SELECT *
FROM MY_EMPLOYEE
ORDER BY ID;
```

Question 21

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

Answer 21

Command:

```
ROLLBACK TO SAVEPOINT all_inserted;
```

Question 22

Confirm that the new row is still intact.

Answer 22

Command:

```
SELECT *
FROM MY_EMPLOYEE
ORDER BY ID;
```

Question 23

Make the data addition permanent.

Answer 23

Command:

```
COMMIT;
```

2 Chapter - 9 - Creating and Managing Tables

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		
Null/Unique		
FK Table		
FK Column		
Data type	NUMBER	VARCHAR2
Length	7	25

Expected structure:

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

Answer 1

Command:

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

DESCRIBE DEPT;
```

Question 2

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

Answer 2

Command:

```
INSERT ALL
    INTO DEPT VALUES (10, 'Administration')
    INTO DEPT VALUES (20, 'Marketing')
    INTO DEPT VALUES (70, 'Public Relations')
    INTO DEPT VALUES (30, 'Purchasing')
    INTO DEPT VALUES (50, 'Shipping')
    INTO DEPT VALUES (60, 'IT')
    INTO DEPT VALUES (100, 'Finance')
    INTO DEPT VALUES (80, 'Sales')
SELECT * FROM dual;
```

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				
Null/Unique				
FK Table				
FK Column				
Data type	NUMBER	VARCHAR2	VARCHAR2	NUMBER
Length	7	25	25	7

Expected structure:

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

Answer 3

Command:

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

DESCRIBE EMP;
```

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)

Answer 4

Command:

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

DESCRIBE EMP;
```

Question 5

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

Answer 5

Command:

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

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.

Answer 6

Command:

```
CREATE TABLE EMPLOYEES2 (
    ID NUMBER,
    FIRST_NAME VARCHAR2(20),
    LAST_NAME VARCHAR2(25),
    SALARY NUMBER(8,2),
    DEPT_ID NUMBER(4)
);

-- Alternative approach using CTAS (Create Table As Select):
CREATE TABLE EMPLOYEES2 AS
SELECT EMPLOYEE_ID AS ID,
       FIRST_NAME,
       LAST_NAME,
       SALARY,
       DEPARTMENT_ID AS DEPT_ID
FROM EMPLOYEES;
```

Question 7

Drop the EMP table.

Answer 7

Command:

```
DROP TABLE EMP;
```

Question 8

Rename the EMPLOYEES2 table as EMP.

Answer 8

Command:

```
RENAME EMPLOYEES2 TO EMP;
```

Question 9

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

Answer 9

Command:

```
COMMENT ON TABLE DEPT IS 'Contains Id, Name';
COMMENT ON TABLE EMP IS 'Contains Id, Names, Salary';

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

Question 10

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

Answer 10

Command:

```
ALTER TABLE EMP
DROP COLUMN FIRST_NAME;

DESCRIBE EMP;
```

Question 11

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

Answer 11

Command:

```
ALTER TABLE EMP
SET UNUSED COLUMN DEPT_ID;

DESCRIBE EMP;
```

Question 12

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

Answer 12

Command:

```
ALTER TABLE EMP
DROP UNUSED COLUMNS;

DESCRIBE EMP;
```

3 Chapter - 10 - Including Constraints

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.

Answer 1

Command:

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

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

Answer 2

Command:

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

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.

Answer 3

Command:

```
ALTER TABLE EMP
ADD DEPT_ID NUMBER(4);

ALTER TABLE EMP
ADD CONSTRAINT my_emp_dept_id_fk
FOREIGN KEY (DEPT_ID) REFERENCES DEPT(ID);
```

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.
Expected result:

CONSTRAINT_NAME	C
MY_DEPT_ID_PK	P
SYS_C00541	C
MY_EMP_ID_PK	P
MY_EMP_DEPT_ID_FK	R

Answer 4

Command:

```
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE C
FROM USER_CONSTRAINTS;
```

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.

Answer 5

Command:

```
SELECT OBJECT_NAME, OBJECT_TYPE
FROM USER_OBJECTS
WHERE OBJECT_NAME NOT IN ('MY_EMPLOYEE');
```

Question 6

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.

Answer 6

Command:

```
ALTER TABLE EMP
ADD COMMISSION NUMBER(2,2) CHECK (COMMISSION > 0);

-- Alternative with named constraint:
ALTER TABLE EMP
ADD COMMISSION NUMBER(2,2) CONSTRAINT emp_commission_ck CHECK (COMMISSION > 0);
```

4 Chapter - 18 - Advanced Subqueries

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.
Expected Result:

LAST_NAME	DEPARTMENT_ID	SALARY
Taylor	80	8600
Zlotkey	80	10500
Abel	80	11000

Answer 1

Command:

```
SELECT LAST_NAME, DEPARTMENT_ID, SALARY
FROM HR.EMPLOYEES
WHERE (DEPARTMENT_ID, SALARY) IN (
    SELECT DEPARTMENT_ID, SALARY
    FROM HR.EMPLOYEES
    WHERE COMMISSION_PCT IS NOT null
);
```

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.
Expected Result:

LAST_NAME	DEPARTMENT_NAME	SALARY
Whalen	Administration	4400
Gietz	Accounting	8300
Higgins	Accounting	12000
Kochhar	Executive	17000
De Haan	Executive	17000
King	Executive	24000

Answer 2

Command:

```
SELECT e1.LAST_NAME, d1.DEPARTMENT_NAME, e1.SALARY
FROM HR.EMPLOYEES e1, HR.DEPARTMENTS d1
WHERE e1.DEPARTMENT_ID = d1.DEPARTMENT_ID
AND (e1.SALARY, e1.COMMISSION_PCT) IN (
    SELECT e2.SALARY, e2.COMMISSION_PCT
    FROM HR.EMPLOYEES e2, HR.DEPARTMENTS d2
    WHERE e2.DEPARTMENT_ID = d2.DEPARTMENT_ID
    AND d2.LOCATION_ID = 2400
);
```

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.
Note: Do not display Kochhar in the result set.
Expected Result:

LAST_NAME	HIRE_DATE	SALARY
De Haan	13-JAN-93	17000

Answer 3

Command:

```
SELECT LAST_NAME, HIRE_DATE, SALARY
FROM HR.EMPLOYEES
WHERE (SALARY, COMMISSION_PCT) IN (
    SELECT e.SALARY, e.COMMISSION_PCT
    FROM HR.EMPLOYEES e
    WHERE e.LAST_NAME = 'Kochhar'
)
AND LAST_NAME <> 'Kochhar';
```

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.
Expected Result:

LAST_NAME	JOB_ID	SALARY
King	AD_PRES	24000
Kochhar	AD_VP	17000
De Haan	AD_VP	17000
Hartstein	MK_MAN	13000
Higgins	AC_MGR	12000
Abel	SA_REP	11000

Answer 4

Command:

```
SELECT LAST_NAME, JOB_ID, SALARY
FROM HR.EMPLOYEES
WHERE SALARY > (
    SELECT MAX(SALARY)
    FROM HR.EMPLOYEES
    WHERE JOB_ID = 'SA_MAN'
)
ORDER BY SALARY DESC;

-- Alternative using ALL operator:
SELECT LAST_NAME, JOB_ID, SALARY
FROM HR.EMPLOYEES
WHERE SALARY > ALL (
    SELECT SALARY
    FROM HR.EMPLOYEES
    WHERE JOB_ID = 'SA_MAN'
)
ORDER BY SALARY DESC;
```

Question 5

Display the details of the employee ID, last name, and department ID of those employees who live in cities whose name begins with T.
Expected Result:

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
201	Hartstein	20
202	Fay	20

Answer 5

Command:

```
SELECT EMPLOYEE_ID, LAST_NAME, DEPARTMENT_ID
FROM EMPLOYEES
WHERE DEPARTMENT_ID IN (
    SELECT DEPARTMENT_ID
    FROM DEPARTMENTS
    WHERE LOCATION_ID IN (
        SELECT LOCATION_ID
        FROM LOCATIONS
        WHERE CITY LIKE 'T%'
    )
);
```

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 aliases for the columns retrieved by the query as shown in the sample output.
Expected Result:

ENAME	SALARY	DEPTNO	DEPT_AVG
Mavingos	6500	50	3500
Hunold	9000	60	6400
Hartstein	13000	20	9500
Abel	11000	80	10333.3333
Zlotkey	10500	80	10333.3333
Higgins	12000	110	10150
King	24000	90	19333.3333

Answer 6

Command:

```
WITH dept_averages AS (
    SELECT DEPARTMENT_ID, AVG(SALARY) AS dept_avg
    FROM HR.EMPLOYEES
    WHERE DEPARTMENT_ID IS NOT NULL
    GROUP BY DEPARTMENT_ID
)
SELECT e.LAST_NAME AS ENAME, e.SALARY, e.DEPARTMENT_ID AS DEPTNO, da.dept_avg AS DEPT_AVG
FROM HR.EMPLOYEES e
JOIN dept_averages da ON e.DEPARTMENT_ID = da.DEPARTMENT_ID
WHERE e.SALARY > da.dept_avg
ORDER BY da.dept_avg;
```

Question 7

Find all employees who are not supervisors.
a. First do this using the NOT EXISTS operator.
Expected Result:

LAST_NAME
Ernst
Lorentz
Rajs
Davies
Matos
Vargas
Abel
Taylor
Grant
Whalen
Fay
Gietz

b. Can this be done by using the NOT IN operator? How, or why not?

Answer 7

Command:

```sql
-- a. Using NOT EXISTS operator:
SELECT LAST_NAME
FROM EMPLOYEES e1
WHERE NOT EXISTS (
    SELECT 1
    FROM EMPLOYEES e2
    WHERE e2.MANAGER_ID = e1.EMPLOYEE_ID
);

-- b. Using NOT IN operator:
SELECT LAST_NAME
FROM EMPLOYEES
WHERE EMPLOYEE_ID NOT IN (
    SELECT MANAGER_ID
    FROM EMPLOYEES
    WHERE MANAGER_ID IS NOT NULL
);
```

Answer 7b - Explanation

Yes, this can be done using the NOT IN operator, but we must be careful with NULL values. The WHERE MANAGER_ID IS NOT NULL clause is essential because NOT IN will return no results if any NULL values exist in the subquery result set. The NOT EXISTS approach is generally safer as it handles NULLs correctly without additional conditions.

Question 8

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

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

Answer 8

Command:

```
SELECT LAST_NAME
FROM EMPLOYEES e1
WHERE SALARY < (
    SELECT AVG(SALARY)
    FROM EMPLOYEES e2
    WHERE e2.DEPARTMENT_ID = e1.DEPARTMENT_ID
    AND e2.DEPARTMENT_ID IS NOT NULL
);
```

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.
Expected Result:

LAST_NAME
Rajs
Davies
Matos
Vargas
Taylor

Answer 9

Command:

```
SELECT DISTINCT e1.LAST_NAME
FROM EMPLOYEES e1
WHERE EXISTS (
    SELECT 1
    FROM EMPLOYEES e2
    WHERE e2.DEPARTMENT_ID = e1.DEPARTMENT_ID
    AND e2.HIRE_DATE > e1.HIRE_DATE
    AND e2.SALARY > e1.SALARY
    AND e1.DEPARTMENT_ID IS NOT NULL
);
```

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.
Expected Result (partial):

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

Answer 10

Command:

```
SELECT EMPLOYEE_ID, LAST_NAME,
    (SELECT DEPARTMENT_NAME
     FROM DEPARTMENTS d
     WHERE d.DEPARTMENT_ID = e.DEPARTMENT_ID) AS DEPARTMENT
FROM EMPLOYEES e
ORDER BY EMPLOYEE_ID;
```

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.
Expected Result:

DEPARTMENT_NAME	DEPT_TOTAL
Executive	58000
Sales	37100

Answer 11

Command:

```sql
WITH SUMMARY AS (
    SELECT
        d.DEPARTMENT_NAME,
        SUM(e.SALARY) AS dept_total,
        (SELECT SUM(SALARY) FROM EMPLOYEES) / 8 AS company_threshold
    FROM DEPARTMENTS d
    JOIN EMPLOYEES e ON d.DEPARTMENT_ID = e.DEPARTMENT_ID
    GROUP BY d.DEPARTMENT_NAME
)
SELECT DEPARTMENT_NAME, dept_total
FROM SUMMARY
WHERE dept_total > company_threshold;
```