University of Chittagong

Department of Computer Science & Engineering

Database Systems Lab

Assignment 1

Chapters 1-4 Practice Problems

CSE 414

Assignment 01

Submitted By: Debashish Chakraborty
2022 - 2023
23701034

Submitted To: Dr. Rudra Pratap Deb Nath
Associate Professor

February 27, 2025

Contents

1 Chapter 1 Problems & Solutions

1.1 Part 1

Problem 1. The following SELECT statement executes successfully:

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

Solution: TRUE

Problem 2. The following SELECT statement executes successfully:

```sql
SELECT *
FROM job_grades;
```

Solution: TRUE

Problem 3. There are four coding errors in the following statement. Can you identify them?

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

Solution: The detected errors are:

1. Column names with spaces must be written correctly - employee_id and last_name are invalid unless quoted.
2. Missing commas between selected columns - There's no comma between last_name and sal x 12 ANNUAL SALARY.
3. Invalid expression syntax - sal x 12 is invalid. It should be salary * 12.
4. Missing AS for alias (optional but recommended) - ANNUAL SALARY should be written as AS "ANNUAL SALARY" if you want a readable header.

1.2 Part 2

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

Solution:

```sql
DESC HR.EMPLOYEES;
```

Table structure:

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

1.3 Part 3

Problem 5. 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.

Solution:

```sql
SELECT employee_id, last_name, hire_date
FROM HR.EMPLOYEES;
```

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

Solution:

```sql
SELECT employee_id, last_name, hire_date
FROM HR.EMPLOYEES;
```

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

Solution:

```sql
SELECT UNIQUE(job_id)
FROM hr.employees;
```

Problem 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.

Solution:

```sql
SELECT employee_id AS "Emp #",
       CONCAT(FIRST_NAME, ' ', LAST_NAME) AS "Employee",
       job_id AS "Job",
       hire_date AS "Hire Date"
FROM hr.employees;
```

Problem 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.

Solution:

```sql
SELECT LAST_NAME || ', ' || JOB_ID AS "Employee and Title"
FROM HR.EMPLOYEES;
```

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

Solution:

```sql
SELECT
    EMPLOYEE_ID || ', ' ||
    FIRST_NAME || ', ' ||
    LAST_NAME || ', ' ||
    EMAIL || ', ' ||
    PHONE_NUMBER || ', ' ||
    TO_CHAR(HIRE_DATE, 'YYYY-MM-DD') || ', ' ||
    JOB_ID || ', ' ||
    SALARY || ', ' ||
    COMMISSION_PCT || ', ' ||
    MANAGER_ID || ', ' ||
    DEPARTMENT_ID AS "THE_OUTPUT"
FROM HR.EMPLOYEES;
```

2 Chapter 2 Problems & Solutions

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

Solution:

```
SELECT LAST_NAME, SALARY
FROM HR.EMPLOYEES
WHERE SALARY > 12000;
```

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

Solution:

```
SELECT LAST_NAME, DEPARTMENT_ID
FROM HR.EMPLOYEES
WHERE EMPLOYEE_ID = 176;
```

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

Solution:

```
SELECT LAST_NAME, SALARY
FROM HR.EMPLOYEES
WHERE SALARY NOT BETWEEN 5000 AND 12000;
```

Problem 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 the hire date.

Solution:

```
SELECT LAST_NAME, JOB_ID, HIRE_DATE
FROM HR.EMPLOYEES
WHERE LAST_NAME IN ('Matos', 'Taylor')
ORDER BY HIRE_DATE ASC;
```

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

Solution:

```
SELECT LAST_NAME, DEPARTMENT_ID
FROM HR.EMPLOYEES
```

```sql
WHERE DEPARTMENT_ID IN (20, 50)
ORDER BY LAST_NAME ASC;
```

Problem 6. Modify lab_02_03.sql to display 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.
Solution:

```sql
SELECT LAST_NAME AS "Employee",
       SALARY AS "Monthly Salary"
FROM HR.EMPLOYEES
WHERE SALARY BETWEEN 5000 AND 12000
  AND DEPARTMENT_ID IN (20, 50);
```

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

```sql
SELECT LAST_NAME, HIRE_DATE
FROM HR.EMPLOYEES
WHERE TO_CHAR(HIRE_DATE, 'YYYY') = '1994';
```

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

```sql
SELECT LAST_NAME, JOB_ID
FROM HR.EMPLOYEES
WHERE MANAGER_ID IS NULL;
```

Problem 9. Create a report to display the last name, salary, and commission of 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.
Solution:

```sql
SELECT LAST_NAME, SALARY, COMMISSION_PCT
FROM HR.EMPLOYEES
WHERE COMMISSION_PCT IS NOT NULL
ORDER BY 2 DESC, 3 DESC;
```

Problem 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. Save this query to a file named lab_02_10.sql. If you enter 12000 when prompted, the report displays the following results:
Solution:

```sql
-- lab_02_10.sql
SELECT last_name, salary
FROM HR.EMPLOYEES
WHERE salary > &amount;
```

Problem 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.

Solution: We can create a flexible query that prompts for both the manager ID and the sorting column. Here's how you'd do it in Oracle SQL*Plus (which supports substitution variables):

```sql
SELECT employee_id , last_name , salary , department_id
FROM HR.EMPLOYEES
WHERE manager_id = &manager_id
ORDER BY &sort_column;
```

Examples:

- manager_id = 103, sorted by last_name:

```
Enter value for manager_id: 103
Enter value for sort_column: last_name
```

- manager_id = 201, sorted by salary:

```
Enter value for manager_id: 201
Enter value for sort_column: salary
```

- manager_id = 124, sorted by employee_id:

```
Enter value for manager_id: 124
Enter value for sort_column: employee_id
```

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

Solution:

```sql
SELECT last_name
FROM HR.EMPLOYEES
WHERE last_name LIKE '__a%';
```

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

Solution:

```sql
SELECT last_name
FROM HR.EMPLOYEES
WHERE last_name LIKE '%a%' AND last_name LIKE '%e%';
```

Problem 14. Display the last name, job, and salary for all employees whose jobs are either those of a sales representative or of a stock clerk, and whose salaries are not equal to $2,500, $3,500, or $7,000.

Solution:

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

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

Solution:

```sql
SELECT last_name, salary, commission_pct
FROM HR.EMPLOYEES
WHERE commission_pct = 0.2;
```

3 Chapter 3 Problems & Solutions

Problem 1. Write a query to display the system date. Label the column as Date.

Note: If your database is remotely located in a different time zone, the output will be the date for the operating system on which the database resides.

Solution:

```sql
SELECT SYSDATE AS "Date"
FROM dual;
```

Problem 2. The HR department needs a report to display the employee number, last name, salary, and salary increased by 15.5% (expressed as a whole number) for each employee. Label the column New Salary. Save your SQL statement in a file named lab_03_02.sql.

Solution:

```sql
-- lab_03_02.sql
SELECT employee_id, last_name, salary,
       FLOOR(salary * 1.155) AS "New Salary"
FROM HR.EMPLOYEES;
```

Problem 3. Run your query in the lab_03_02.sql file.

Solution:

```sql
-- lab_03_02.sql
SELECT employee_id, last_name, salary,
       FLOOR(salary * 1.155) AS "New Salary"
FROM HR.EMPLOYEES;
```

Sample output:

```
EMPLOYEE_ID LAST_NAME  SALARY  NEW SALARY
100         King       24000   27720
101         Kochhar    17000   19635
102         De Haan    17000   19635
103         Hunold      9000   10395
104         Ernst       6000    6930
...         ...         ...     ...
```

Problem 4. Modify your query lab_03_02.sql to add a column that subtracts the old salary from the new salary. Label the column Increase. Save the contents of the file as lab_03_04.sql. Run the revised query.

Solution:

```sql
-- lab_03_04.sql
SELECT employee_id, last_name, salary,
       FLOOR(salary * 1.155) AS "New Salary",
       FLOOR(salary * 1.155) - salary AS "Increase"
FROM HR.EMPLOYEES;
```

Problem 5. Write a query that displays the last name (with the first letter in uppercase and all the other letters in lowercase) and the length of the last name for all employees whose name starts with the letters "J," "A," or "M." Give each column an appropriate label. Sort the results by the employees' last names.

Solution:

```sql
SELECT INITCAP(last_name) AS "Formatted Last Name",
       LENGTH(last_name) AS "Name Length"
FROM HR.EMPLOYEES
WHERE UPPER(SUBSTR(last_name, 1, 1)) IN ('J', 'A', 'M')
ORDER BY last_name;
```

Rewrite the query so that the user is prompted to enter a letter that the last name starts with. For example, if the user enters "H" (capitalized) when prompted for a letter, then the output should show all employees whose last name starts with the letter "H."

```sql
SELECT INITCAP(last_name) AS "Formatted Last Name",
       LENGTH(last_name) AS "Name Length"
FROM HR.EMPLOYEES
WHERE UPPER(SUBSTR(last_name, 1, 1)) = UPPER('&start_letter')
ORDER BY last_name;
```

Modify the query such that the case of the entered letter does not affect the output. The entered letter must be capitalized before being processed by the SELECT query.

```sql
SELECT INITCAP(last_name) AS "Formatted Last Name",
       LENGTH(last_name) AS "Name Length"
FROM HR.EMPLOYEES
WHERE UPPER(SUBSTR(last_name, 1, 1)) = UPPER('&letter')
ORDER BY last_name;
```

Problem 6. The HR department wants to find the duration of employment for each employee. For each employee, display the last name and calculate the number of months between today and the date on which the employee was hired. Label the column as MONTHS_WORKED. Order your results by the number of months employed. Round the number of months up to the closest whole number.

Solution:

```sql
SELECT last_name,
       CEIL(MONTHS_BETWEEN(SYSDATE, hire_date)) AS
          "MONTHS_WORKED"
FROM HR.EMPLOYEES
ORDER BY "MONTHS_WORKED";
```

Problem 7. Create a query to display the last name and salary for all employees. Format the salary to be 15 characters long, left-padded with the $ symbol. Label the column as SALARY.

Solution:

```
SELECT last_name,
       LPAD(salary, 15, '$') AS "SALARY"
FROM HR.EMPLOYEES;
```

Problem 8. Create a query that displays the first eight characters of the employees' last names and indicates the amounts of their salaries with asterisks. Each asterisk signifies a thousand dollars. Sort the data in descending order of salary. Label the column as EMPLOYEES_AND_THEIR_SALARIES.

Solution:

```
SELECT RPAD(SUBSTR(last_name, 1, 8), 8, ' ') || ' ' ||
       RPAD('*', FLOOR(salary / 1000), '*') AS
          "EMPLOYEES_AND_THEIR_SALARIES"
FROM HR.EMPLOYEES
ORDER BY salary DESC;
```

Problem 9. Create a query to display the last name and the number of weeks employed for all employees in department 90. Label the number of weeks column as TENURE. Truncate the number of weeks value to 0 decimal places. Show the records in descending order of the employee's tenure. Note: The TENURE value will differ as it depends on the date on which you run the query.

Solution:

```
SELECT last_name,
       TRUNC((SYSDATE - hire_date) / 7) AS TENURE
FROM HR.EMPLOYEES
WHERE department_id = 90
ORDER BY TENURE DESC;
```

4 Chapter 4 Problems & Solutions

Problem 1. Create a report that produces the following for each employee: <employee last name> earns <salary> monthly but wants <3 times salary.> Label the column Dream Salaries.

Solution:

```
SELECT last_name || ' earns ' || salary || ' monthly but wants '
    ||
       (salary * 3) || '.' AS "Dream Salaries"
FROM HR.EMPLOYEES;
```

Problem 2. Display each employee's last name, hire date, and salary review date, which is the first Monday after six months of service. Label the column REVIEW. Format the dates to appear in the format similar to "Monday, the Thirty-First of July, 2000."

Solution:

```sql
SELECT last_name,
       TO_CHAR(hire_date, 'Day, "the" Ddspth "of" Month, YYYY')
          AS "Hire Date",
       TO_CHAR(
           NEXT_DAY(ADD_MONTHS(hire_date, 6) - 1, 'MONDAY'),
           'Day, "the" Ddspth "of" Month, YYYY'
       ) AS "Review"
FROM HR.EMPLOYEES;
```

Problem 3. Display the last name, hire date, and day of the week on which the employee started. Label the column DAY. Order the results by the day of the week, starting with Monday.

Solution:

```sql
SELECT last_name,
       hire_date,
       TO_CHAR(hire_date, 'Day') AS DAY
FROM HR.EMPLOYEES
ORDER BY TO_CHAR(hire_date, 'D');
```

Problem 4. Create a query that displays the employees' last names and commission amounts. If an employee does not earn commission, show "No Commission." Label the column COMM.

Solution:

```sql
SELECT last_name,
       NVL(TO_CHAR(commission_pct), 'No Commission') AS COMM
FROM HR.EMPLOYEES;
```

Problem 5. Using the DECODE function, write a query that displays the grade of all employees based on the value of the column JOB_ID, using the following data:

Solution:

```sql
SELECT last_name,
       job_id,
       DECODE(job_id,
           'AD_PRES', 'A',
           'ST_MAN', 'B',
           'IT_PROG', 'C',
           'SA_REP', 'D',
           'ST_CLERK','E',
           '0') AS GRADE
FROM HR.EMPLOYEES;
```

Problem 6. Rewrite the statement in the preceding exercise using the CASE syntax.

Solution:

```sql
SELECT last_name,
       job_id,
       CASE job_id
           WHEN 'AD_PRES' THEN 'A'
           WHEN 'ST_MAN' THEN 'B'
           WHEN 'IT_PROG' THEN 'C'
```

```
            WHEN 'SA_REP' THEN 'D'
            WHEN 'ST_CLERK' THEN 'E'
            ELSE 'O'
        END AS GRADE
FROM HR.EMPLOYEES;
```