University of Chittagong

Department of Computer Science & Engineering

Database Systems

Anik Kirtania

ID: 23701002

Submitted to:

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

CSE 413

Database Systems Assignment

Chapter 1 Solution:

1. **The following SELECT statement executes successfully:**

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

Answer: True
Explanation: The syntax is correct. The statement retrieves the last_name, job_id, and salary (renamed as Sal) from the employees table.

2. **The following SELECT statement executes successfully:**

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

Answer: True
Explanation: Assuming the table job_grades exists, this is a correct SELECT statement to retrieve all columns.

3. **There are four coding errors in the following statement:**

```sql
-- Original incorrect statement:
SELECT   employee_id, last_name
sal x 12 ANNUAL SALARY
FROM     employees;

-- Corrected statement:
SELECT employee_id, last_name,
       salary * 12 AS "ANNUAL SALARY"
FROM   employees;
```

Errors:

a) Missing a comma after last_name
b) Incorrect column name sal; should be salary
c) Incorrect multiplication operator; should be * instead of x
d) Alias ANNUAL SALARY should be in double quotes or AS keyword should be used

4. **Determine the structure of the DEPARTMENTS table:**

```sql
DESCRIBE departments;

SELECT *
FROM departments;
```

5. **Display selected columns from EMPLOYEES table:**

```sql
DESCRIBE employees;

SELECT employee_id, last_name, job_id,
       hire_date AS STARTDATE
FROM employees;
```

6. **Test lab_01_05.sql query:**

```
-- Open lab_01_05.sql in SQL Developer and run:
SELECT employee_id, last_name, job_id,
       hire_date AS STARTDATE
FROM employees;
```

7. **Display unique job IDs:**

```
SELECT DISTINCT job_id
FROM employees
ORDER BY job_id;
```

8. **Display employees with descriptive column headings:**

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

9. **Concatenate employee and job title:**

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

10. **Display all employee data in a single column:**

```
SELECT
    employee_id || ’,’ ||
    first_name || ’,’ ||
    last_name || ’,’ ||
    email || ’,’ ||
    phone_number || ’,’ ||
    job_id || ’,’ ||
    manager_id || ’,’ ||
    hire_date || ’,’ ||
    salary || ’,’ ||
    NVL(commission_pct, 0) AS THE_OUTPUT
FROM employees
ORDER BY employee_id;
```

Chapter 2 Solution:

Query 1: High Salary Employee Report

The following SQL query displays the last name and salary of employees who earn more than $12,000:

```
SELECT last_name, salary
FROM employees
WHERE salary > 12000
ORDER BY salary DESC;
```

Query 2: Employee Department Lookup

The following SQL query displays the last name and department number for employee number 176:

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

Query 3: Salary Range Exclusion Report

The following SQL query displays the last name and salary for employees whose salary is outside the range of $5,000 to $12,000:

```sql
SELECT last_name, salary
FROM employees
WHERE salary NOT BETWEEN 5000 AND 12000
ORDER BY salary DESC;
```

Query 4: Employee Hire Date Report

The following SQL query displays the last name, job ID, and hire date for employees named Matos and Taylor:

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

Query 5: Department Employee Report

The following SQL query displays the last name and department ID for employees in departments 20 or 50:

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

Query 6: Salary Range Department Report

The following SQL query displays employees in departments 20 or 50 with salaries between $5,000 and $12,000:

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

Query 7: 1994 Hire Date Report

The following SQL query displays the last name and hire date for employees hired in 1994:

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

Query 8: Employees Without Managers

The following SQL query displays the last name and job title of employees who don't have a manager:

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

Query 9: Commission Report

The following SQL query displays the last name, salary, and commission of employees who earn commissions, sorted by salary and commission in descending order:

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

Query 10: Flexible Salary Threshold Report

The following SQL query creates an interactive report that prompts for a salary threshold and displays employees earning above that amount:

```sql
SELECT last_name, salary
FROM employees
WHERE salary > &salary_threshold
ORDER BY salary DESC;
```

Query 11: Manager Employee Report

The following SQL query creates a flexible report based on manager ID, showing employee details:

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

Example usage with different parameters:

- For manager_id = 103 (sorted by last_name):

```sql
SELECT employee_id, last_name, salary, department_id
FROM employees
WHERE manager_id = 103
ORDER BY last_name;
```

- For manager_id = 201 (sorted by salary):

```sql
SELECT employee_id, last_name, salary, department_id
FROM employees
WHERE manager_id = 201
ORDER BY salary;
```

- For manager_id = 124 (sorted by employee_id):

```sql
SELECT employee_id, last_name, salary, department_id
FROM employees
WHERE manager_id = 124
ORDER BY employee_id;
```

Query 12: Third Letter 'a' Report

The following SQL query displays all employee last names where the third letter is 'a':

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

Query 13: Names with 'a' and 'e'

The following SQL query displays last names of employees who have both 'a' and 'e' in their last name:

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

Query 14: Sales and Stock Clerk Salary Report

The following SQL query displays information for sales representatives and stock clerks with specific salaries:

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

Query 15: 20% Commission Report

The following SQL query displays employees with 20% commission:

```sql
SELECT last_name AS Employee,
       salary AS "Monthly Salary",
       commission_pct AS COMMISSION_PCT
FROM employees
WHERE commission_pct = 0.2
ORDER BY salary DESC;
```

Chapter 3 Solution:

1. Display System Date

Task: Write a query to display the system date with column alias 'Date'.

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

2. Calculate Salary Increase

Task: Display employee details with 15.5

```sql
SELECT employee_id, last_name, salary,
       ROUND(salary * 1.155) AS "New Salary"
FROM employees
ORDER BY employee_id;
```

3. Execute Saved Query

Task: Run the query in lab_03_02.sql file.

```sql
-- Contents of lab_03_02.sql
SELECT employee_id, last_name, salary,
       ROUND(salary * 1.155) AS "New Salary"
FROM employees
ORDER BY employee_id;
```

4. Calculate Salary Increase Difference

Task: Modify the previous query to show the difference between old and new salaries.

```sql
SELECT employee_id, last_name, salary,
       ROUND(salary * 1.155) AS "New Salary",
       ROUND(salary * 1.155) - salary AS "Increase"
FROM employees
ORDER BY employee_id;
```

5. Name Case Formatting and Length

Task: Write a query to display formatted last names and their lengths for employees whose names start with specific letters.

a) Initial Query for employees with last names starting with ’J’, ’A’, or ’M’:

```sql
SELECT INITCAP(last_name) AS "Name",
       LENGTH(last_name) AS "Length"
FROM employees
WHERE last_name LIKE ’J%’
   OR last_name LIKE ’A%’
   OR last_name LIKE ’M%’
ORDER BY last_name;
```

b) Interactive Query using substitution variable:

```sql
-- First, set the verification of substitution variables ON
SET VERIFY ON

-- Define the format for substitution variable prompts
SET VERIFY OFF
SET FEEDBACK ON

-- Main query with substitution variable
SELECT INITCAP(last_name) AS "Name",
       LENGTH(last_name) AS "Length"
```

```
FROM employees
WHERE UPPER(last_name) LIKE UPPER(’&START_LETTER%’)
ORDER BY last_name;
```

c) Modified Query with case-insensitive search:

```
SELECT INITCAP(last_name) AS "Name",
       LENGTH(last_name) AS "Length"
FROM employees
WHERE UPPER(last_name) LIKE UPPER(’&START_LETTER%’)
ORDER BY last_name;
```

6. Calculate Employment Duration

Task: Calculate the duration of employment for each employee in months.

```
SELECT last_name,
       ROUND(MONTHS_BETWEEN(SYSDATE, hire_date)) AS "MONTHS_WORKED"
FROM employees
ORDER BY MONTHS_WORKED;
```

7. Salary Formatting with *Symbol*

Task: Create a query to display last names and salaries with left-padded *symbols*.

```
SELECT last_name,
       LPAD(’$’ || salary, 15, ’$’) AS "SALARY"
FROM employees
ORDER BY salary DESC;
```

8. Salary Display with Asterisks

Task: Display first 8 characters of last names with salary represented by asterisks.

```
SELECT RPAD(SUBSTR(last_name, 1, 8), 8, ’ ’) || ’ ’ ||
       RPAD(’*’, ROUND(salary/1000), ’*’)
       AS "EMPLOYEES_AND_THEIR_SALARIES"
FROM employees
ORDER BY salary DESC;
```

9. Employment Tenure in Department 90

Task: Display last names and tenure (in weeks) for department 90 employees.

```
SELECT last_name,
       TRUNC((SYSDATE - hire_date)/7, 0) AS "TENURE"
FROM employees
WHERE department_id = 90
ORDER BY "TENURE" DESC;
```

Chapter 4 Solution:

1. Employee Dream Salaries Report

Task: Create a report showing current salary and desired salary (3 times current) for each employee.

```
SELECT last_name || ’ earns $’ ||
       TO_CHAR(salary, ’99,999.00’) ||
       ’ monthly but wants $’ ||
       TO_CHAR(salary * 3, ’99,999.00’) || ’.’
       AS "Dream Salaries"
FROM employees
ORDER BY salary DESC;
```

2. Salary Review Dates

Task: Display employee names, hire dates, and review dates (first Monday after six months).

```
SELECT last_name, hire_date,
       TO_CHAR(
           NEXT_DAY(
               ADD_MONTHS(hire_date, 6),
               ’MONDAY’
           ),
           ’fmDay, "the" Ddspth "of" Month, YYYY’
       ) AS "REVIEW"
FROM employees
ORDER BY hire_date;
```

3. Employee Start Day of Week

Task: Display last name, hire date, and the day of week when hired, ordered by weekday starting Monday.

```
SELECT last_name, hire_date,
       TO_CHAR(hire_date, ’DAY’) AS "DAY"
FROM employees
ORDER BY TO_CHAR(hire_date, ’D’);
```

4. Commission Report

Task: Display employee last names and commission amounts, showing ”No Commission” for null values.

```
SELECT last_name,
       NVL2(commission_pct,
            TO_CHAR(commission_pct),
            ’No Commission’) AS "COMM"
FROM employees
ORDER BY last_name;
```

5. Grade Assignment using DECODE

Task: Display employee job IDs and grades using the DECODE function.

```
SELECT job_id,
       DECODE(job_id,
              ’AD_PRES’, ’A’,
              ’ST_MAN’,  ’B’,
              ’IT_PROG’, ’C’,
              ’SA_REP’,  ’D’,
              ’ST_CLERK’,’E’,
              ’0’) AS "GRADE"
FROM employees
ORDER BY job_id;
```

6. Grade Assignment using CASE

Task: Rewrite the previous query using CASE syntax.

```
SELECT 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 ’0’
       END AS "GRADE"
FROM employees
ORDER BY job_id;
```