Database Programming with SQL

20-1: Ensuring Quality Query Results– Advanced Technique

Practice Activities

Objectives

- Create a query to produce specific data
- Modify a query to produce specified data

Try It / Solve It

1. Create additional tables used in this section by executing the following statements:

   CREATE TABLE emp AS SELECT * FROM employees;
   CREATE TABLE dept AS SELECT * FROM departments;

2. Produce a report that lists the constraint name, type, column name, and column position of all the constraints on the JOB_HISTORY table, apart from the not null constraints.

3. Create a primary key constraint on the emp table's employee_id column

4. Create a primary key on the dept table's department_id column

5. Add a foreign constraint between DEPT and EMP so that only valid departments can be entered in the EMP table. Make sure you can delete any row from the DEPT table, and that referenced rows in the EMP table are deleted.

6. Test the foreign key constraint you just created:

   Count the number of rows in the EMP table.
   Remove department 10 from the dept table.
   Now count emps again. There should be fewer employees.

7. Produce a report that returns the last name, salary, department number, and average salary of all the departments where salary is greater than the average salary.

8. Create a view named V2 that returns the highest salary, lowest salary, average salary and department name.

9. Create a view named Dept_Managers_view that returns a listing of department names long with the manager initial and surname for that department. Test the view by returning all the rows from it. Make sure no rows can be updated through the view. Try to run an UPDATE statement against the view.

10. Create a sequence named ct_seq using all the default values.

11. Examine the following insert statement and fix the errors.

```
INSERT INTO emp
(employee_id, first_name, last_name, email, phone_number, hire_date,
   job_id, salary, commission_pct, manager_id, department_id)
VALUES
(ct_seq.nextvalue, ''Kaare'', 'Hansen', 'KHANSEN', '44965 832123',
   sysdate, 'SA_REP', $6500, null, 100, 20);
```

12. Write the SQL statement to list all the user tables which contains the name PRIV.

13. Give select access to public on the EMP table, and verify the grant by running this query.

```
SELECT *
FROM   user_tab_privs
WHERE table_name = 'EMP';
```

14. Replace the ?? in the following query using regular expressions to return only the numbers from the following string: 'Oracle Academy9547d6905%&^ db apex'.

```
SELECT REGEXP_REPLACE('Oracle Academy9547d6905%&^ db apex',??,'') regexpreplace
FROM DUAL;
```

15. Amend the previous query using regular expressions to return the number of digits from the following string: 'Oracle Academy9547d6905%&^ db'

```
SELECT LENGTH(REGEXP_REPLACE('Oracle Academy9547d6905%&^ db
apex','??','')) regexpreplace
FROM DUAL;
```

16. Amend the query again to return only the non-numeric characters.

```
SELECT REGEXP_REPLACE('Oracle Academy9547d6905%&^ db apex','??','') regex-
preplace
FROM DUAL;
```

17. Using Oracle proprietary joins, construct a statement that returns all the employee_ids joined to all the department_names.

18. Still using Oracle Joins, correct the previous statement so that it returns only the name of the department that the employee actually works in.

19. Still using Oracle Joins, construct a query that lists the employees last name, the department name, the salary, and the country name of all employees.

20. Still using Oracle join syntax, alter the previous query so that it also includes the employee record of the employee with no department_id, ‘Grant’.