11-1 : Ensuring Quality Query Results

QUERY_1:

- **Problem:**

–Create a list of all tables whose first two characters in the name of the table is JO

–The tables must be owned by the current Oracle User

```
SELECT table_name
FROM user_tables
WHERE table_name LIKE 'JO%'
ORDER BY  table_name;
```

QUERY-2:

Problem: – Create a list that includes the first initial of every employee's first name, a space, and the last name of the employee

```
SELECT SUBSTR( first_name,1,1)|| ' ' || last_name AS “Employee name”
FROM employees
```

QUERY-3:

Problem: – Create a list of every employee's first name concatenated to a space and the employee's last name, and the email of all employees where the email address contains the string 'IN'.

```
SELECT first_name || ' ' || last_name AS “Employee name”,email
FROM employees
WHERE email LIKE '%IN%';
```

QUERY-4:

Problem: – Create a list of 'smallest' last name and the 'highest' last name from the employees table.

```
SELECT MIN(last_name),MAX(last_name)
FROM  employees;
```

QUERY-5:

Problem:

– Create a list of weekly salaries from the employees table where the weekly salary is between 700 and 3000

– The salaries should be formatted to include a $- sign and have two decimal points like: $9999.99

```
SELECT  TO_CHAR(salary / 52, '$9999.99') AS weekly_salary
FROM employees
WHERE  (salary / 52) BETWEEN 700 AND 3000;
                              (Or)
SELECT '$'||ROUND((salary*12)/52,2) AS weekly_salary
FROM employees
WHERE  (salary*12)/52 ETWEEN 700 AND 3000;
```

QUERY-6:

Problem:

– Create a list of every employee and his related job title sorted by job_title

```
SELECT  e.first_name || ' ' || e.last_name AS full_name, j.job_title
```

FROM employees e JOIN jobs j ON e.job_id = j.job_id

ORDER BY j.job_title;

QUERY-7:

- **Problem:**

–Create a list of every employee's job, the salary ranges within the job, and the employee's salary

–List the lowest and highest salary range within each job with a dash to separate the salaries like this: 100 – 200

SELECT SUBSTR(first_name,1,1)||' '||last_name AS "Employee Name", job title AS "Job",min_salary||'-'||max_salary AS "salary range",salary AS "Employees salary

FROM employees e,jobs j

WHERE e.job_id = j.job_id

ORDER BY j.job_title, e.salary;

QUERY-8:

Problem:

– Using an ANSII join method, create a list of every employee's first initial and last name, and department name

– Make sure the tables are joined on all of the foreign keys declared between the two tables

SELECT SUBSTR (e.first_name, 1, 1) || '. ' || e.last_name AS employee_name,d.department_name

FROM employees e JOIN departments d ON e.department_id = d.department_id;

QUERY-9:

Problem:

– Change the previous listing to join only on the department_id column

SELECT SUBSTR (e.first_name, 1, 1) || '. ' || e.last_name AS employee_name, d.department_name

FROM employees e JOIN departments d ON e.department_id = d.department_id;

QUERY-10:

- **Problem:**

− Create a list of every employee's last name, and the word nobody or somebody depending on whether or not the employee has a manager

− Use the Oracle DECODE function to create the list

SELECT last_name,DECODE(manager_id, NULL, 'nobody', 'somebody') AS manager_status

FROM employees;

QUERY-11:

Problem:

− Create a list of every employee's first initial and last name, salary, and a yes or no to show whether or not an employee makes a commission

− Fix this query to produce the result

SELECT SUBSTR(first_name, 1, 1) || '. ' || last_name AS employee_name,salary,NVL2(commission_pct, 'yes', 'no') AS commission_status

FROM employees;

QUERY-12:

Problem:

− Create a list of every employee's last name, department name, city, and state_province

− Include departments without employees − An outer join is required

```
SELECT e.last_name,d.department_name,  l.city,  l.state_province

FROM  departments d LEFT JOIN employees e ON d.department_id =
e.department_id LEFT JOIN locations l ON d.location_id = l.location_id

ORDER BY d.department_name, e.last_name;
```

QUERY-13:

• Problem: −Create a list of every employee's first and last names, and the first occurrence of: commission_pct, manager_id, or -1 −If an employee gets commission, display the commission_pct column; if no commission, then display his manager_id; if he has neither commission nor manager, then the number -1.

```
SELECT  first_name, last_name,

CASE

    WHEN commission_pct IS NOT NULL THEN commission_pct

    WHEN manager_id IS NOT NULL THEN manager_id

    ELSE -1

 END AS result

FROM employees;
```

QUERY-14:

Problem: − Create a list of every employee's last name, salary, and job_grade for all employees working in departments with a department_id greater than 50.

```
SELECT   e.last_name,  e.salary,  j.job_grade

FROM employees e JOIN   jobs j ON e.job_id = j.job_id

WHERE   e.department_id > 50;
```

QUERY-15:

• Problem:

– Produce a list of every employee's last name and department name – Include both employees without departments, and departments without employees

```
SELECT e.last_name,  d.department_name

FROM   departments d LEFT JOIN  employees e ON d.department_id =
e.department_id

ORDER BY   d.department_name, e.last_name;
```

QUERY-16:

•Problem:

–Create a treewalkinglist of every employee's last name, his manager's last name, and his position in the company

–The top level manager has position 1, this manager's subordinates position 2, their subordinates position 3, and so on

–Start the listing with employee number 100.

```
WITH EmployeeHierarchy AS

   (SELECT e.employee_id,  e.last_name AS employee_last_name,
e.manager_id,NULL AS manager_last_name,   1 AS position

   FROM employees e

   WHERE   e.employee_id = 100

   UNION ALL

   SELECT  e.employee_id,e.last_name AS employee_last_name,

      e.manager_id,eh.employee_last_name AS manager_last_name,

      eh.position + 1
```

```
   FROM  employees e INNER JOIN EmployeeHierarchy eh ON e.manager_id =
eh.employee_id)
SELECT eh.employee_id,eh.employee_last_name,eh.manager_last_name,
   eh.position
FROM  EmployeeHierarchy eh
ORDER BY eh.position, eh.employee_id;
```

QUERY-17:

Problem:

– Produce a list of the earliest hire date, the latest hire date, and the number of employees from the employees table

```
SELECT  MIN(hire_date) AS earliest_hire_date,
   MAX(hire_date) AS latest_hire_date,
   COUNT(*) AS number_of_employees
FROM  employees;
```

QUERY-18:

Problem:

– Create a list of department names and the departmental costs (salaries added up)

– Include only departments whose salary costs are between 15000 and 31000, and sort the listing by the cost

```
SELECT d.department_name, SUM(e.salary) AS department_cost
FROM   employees e INNER JOIN   departments d ON e.department_id =
d.department_id
GROUP BY  d.department_name
HAVING   SUM(e.salary) BETWEEN 15000 AND 31000
ORDER BY   department_cost;
```

QUERY-19:

Problem: – Create a list of department names, the manager id, manager name (employee last name) of that department, and the average salary in each department

```
SELECT  d.department_name,  d.manager_id,m.last_name AS manager_name,
  AVG(e.salary) AS average_salary
FROM departments d INNER JOIN  employees m ON d.manager_id =
m.employee_id INNER JOIN  employees e ON d.department_id =
e.department_id
GROUP BY  d.department_name, d.manager_id, m.last_name;
```

QUERY-20:

Problem: – Show the highest average salary for the departments in the employees table – Round the result to the nearest whole number

```
SELECT ROUND(MAX(avg_salary)) AS highest_average_salary
FROM ( SELECT   AVG(salary) AS avg_salary FROM  employees
   GROUP BY department_id) subquery;
```

QUERY-21:

Problem: – Create a list of department names and their monthly costs (salaries added up)

```
SELECT d.department_name, SUM(e.salary) AS monthly_cost
FROM employees e INNER JOIN departments d ON e.department_id =
d.department_id
GROUP BY  d.department_name;
```

QUERY-22:

Problem: – Create a list of department names, and job_ids

– Calculate the monthly salary cost for each job_idwithin a department, for each department, and for all departments added together

-- Monthly salary cost for each job_id within a department

```
WITH DepartmentJobCosts AS (  SELECTd.department_name,e.job_id,
SUM(e.salary / 12) AS monthly_cost FROM  employees e INNER JOIN
     departments d ON e.department_id = d.department_id
   GROUP BY d.department_name, e.job_id)

-- Monthly salary cost for each department
DepartmentCosts AS (  SELECT  department_name, NULL AS job_id,
       SUM(monthly_cost) AS monthly_cost
   FROM  DepartmentJobCosts
   GROUP BY department_name
)

-- Monthly salary cost for all departments
, TotalCost AS (SELECT 'All Departments' AS department_name, NULL AS job_id,
SUM(monthly_cost) AS monthly_cost FROM DepartmentCosts )

SELECT department_name, job_id, monthly_cost
 FROM DepartmentJobCosts UNION ALL SELECT department_name, job_id,
monthly_cost FROM DepartmentCosts UNION ALL SELECT department_name,
job_id, monthly_cost FROM TotalCost
 ORDER BY department_name, job_id;
```

QUERY-23:

•Problem:

–Create a list of department names, and job_ids

–Calculate the monthly salary cost for each job_idwithin a department, for each department, for each group of job_idsirrespective of the department, and for all departments added together (Hint: Cube)

```
SELECT  department_name,  job_id,SUM(salary / 12) AS monthly_cost
FROM    employees e INNER JOIN departments d ON e.department_id =
d.department_id
GROUP BY  CUBE(department_name, job_id)
ORDER BY department_name, job_id;
```

QUERY-24:

Problem: – Expand the previous list to also show if the department_idor job_id was used to create the subtotals shown in the output (Hint: Cube, Grouping)

```
SELECT department_name,  job_id, SUM(salary / 12) AS monthly_cost,
   GROUPING(department_name) AS is_department_total,
   GROUPING(job_id) AS is_job_total
FROM employees e INNER JOIN  departments d ON e.department_id =
d.department_id
GROUP BY CUBE(department_name, job_id)
ORDER BY department_name, job_id;
```

QUERY-25:

Problem: – Create a list that includes the monthly salary costs for each job title within a department – In the same list, display the monthly salary cost per city. (Hint: Grouping Sets)

```
SELECT  department_name,  job_title,  city, SUM(salary / 12) AS monthly_cost
FROM   employees e INNER JOIN departments d ON e.department_id =
d.department_id
INNER JOIN  locations l ON d.location_id = l.location_id
GROUP BY
   GROUPING SETS (  (department_name, job_title),   (city))
ORDER BY
   department_name, job_title, city;
```

QUERY-26:

Problem: –Create a list of employee names as shown and department ids

–In the same report, list the department ids and department names. And finally, list the cities

–The rows should not be joined, just listed in the same report. (Hint: Union)

```
-- List of employee names and department IDs
SELECT  e.employee_name AS name, e.department_id AS id,NULL AS
department_name,NULL AS city
FROM  employees e
UNION ALL
-- List of department IDs and department names
SELECT  NULL AS name,
   d.department_id AS id, d.department_name AS department_name,
   NULL AS city
FROM departments d

UNION ALL
```

```
SELECT NULL AS name, NULL AS id, NULL AS department_name, l.city AS city
FROM  locations l
ORDER BY name, id, department_name, city;
```

QUERY-27:

Problem: – Create a list of each employee's first initial and last name, salary, and department name for each employee earning more than the average for his department

```
SELECT SUBSTR(e.first_name, 1, 1) AS first_initial,e.last_name, e.salary,
   d.department_name
FROM  Employees e JOIN    Departments d ON e.department_id =
d.department_id
WHERE e.salary > (SELECT AVG(salary) FROM EmployeesWHERE department_id
= e.department_id);
```