Chapter 6
SUBQUERIES

- **Objectives**
  - Describe the types of problems that subqueries can solve
  - List the types of subqueries
    - Single-Row
    - Multiple-Row
    - Multiple-Column
    - Correlated Subqueries
    - Nested Subqueries
    - Scalar Subqueries

Subqueries

- Subquery technique also "joins" tables by placing an inner query (SELECT, FROM, WHERE) within a WHERE or HAVING clause of another (outer) query.
- Subqueries can be used with SELECT, INSERT, UPDATE or DELETE statements.

Which employees earn more than Russell?

    How much does Russell earn?

Subquery Format

- The inner query (subquery) executes once before the outer (main) query.
- The results of the inner query is used by the outer query.

```
SELECT  select list
FROM    table
WHERE  operator
        (SELECT  select list
         FROM      table);
```

Which employees earn more than Russell?

```
SELECT last_name, salary
FROM employees
WHERE salary >
    (SELECT salary
     FROM employees
     WHERE upper(last_name) = 'RUSSELL');
```

Guidelines

- Enclose subqueries in parentheses.
- Place subqueries on the right side of the comparison operator for readability.
- Do not add an ORDER BY clause to a subquery.

  – Use single-row operators with single-row subqueries.
  – Use multiple-row operators with multiple-row subqueries.

Types of Subqueries

- **Single-row subqueries**: Queries that return only one row from the inner SELECT statement
  - Use single-row comparison operators
    - = > < >= <= <>

List all employees who are in the same department as employee 113.

```
   SELECT employee_id, last_name, department_id
   FROM employees
   WHERE department_id =
      (SELECT department_id
       FROM employees
       WHERE employee_id = 113);
```

Want the last names, job ids and salaries of all employees who do the same job as Taylor but earn more than Taylor.

```
SELECT last_name, job_id, salary
FROM    employees
WHERE job_id =     (SELECT job_id
                    FROM employees
                    WHERE last_name = ‘Taylor’)
AND       salary >
                    (SELECT salary
                    FROM employees
                    WHERE last_name = ‘Taylor’);
```

Subqueries in a HAVING clause

- Can use a subquery in the HAVING clause of the outer query.
- Can use to filter groups of rows based on the result return by the subquery.
- Oracle server executes the subqueries first.
- Oracle server returns results into the HAVING clause of the main query

```
     SELECT department_id, MIN(salary)
     FROM employees
     GROUP BY department_id
     HAVING MIN(salary) >
             (SELECT MIN(salary)
              FROM employees
              WHERE department_id = 50);
```

What is wrong with this?

```
SELECT employee_id, last_name
FROM    employees
WHERE salary =
                        (SELECT MIN(salary)
                         FROM    employees
                         GROUP BY department_id);
```

Types of Subqueries

- **Multiple-row subqueries**: Queries that return more than one row from the inner SELECT statement
  - Use multiple row comparison operators
    - IN, ANY, ALL

“Which departments are located in Canada or the United Kingdom?"

```
SELECT department_name
FROM departments
WHERE location_id IN
  (SELECT location_id
   FROM locations
   WHERE country_id IN
         (SELECT country_id
          FROM countries
          WHERE upper(country_name) = 'CANADA'
            OR upper(country_name) = 'UNITED KINGDOM'));
```

Note: This is also a Nested Subquery.

- ANY
  - Use to compare a value with each value in a list or subquery.
  - Must place one of the following before the ANY in the query
    - =, <>, <, >, <=, >=

```
SELECT first_name || ' ' || last_name "Name", salary
FROM   employees
WHERE  salary < ANY
                (SELECT salary
                FROM employees
                WHERE job_id = 'IT_PROG')
AND    job_id <> 'IT_PROG';
```

- Note: IN is the same as =ANY

- ALL
  - Use to compare a value with every value in a list or subquery.
  - Must place one of the following before the ANY in the query
    - =, <>, <, >, <=, >=

```
SELECT first_name || ' ' || last_name "Name", salary
FROM   employees
WHERE  salary < ALL
                (SELECT salary
                FROM employees
                WHERE job_id = 'IT_PROG')
AND    job_id <> 'IT_PROG';
```

Types of Subqueries

- **Multi-Column Subqueries**:
  - Can have multiple columns returned as part of the subquery.
  - For example, you want to know who earns the maximum salary within each job category.
  - Use multiple-row comparison operators
    - IN, ANY, ALL

```
SELECT first_name || ' ' || last_name "Name", job_id, salary
FROM employees
WHERE (job_id, salary) IN
   (SELECT job_id, MAX(salary)
    FROM employees
    GROUP BY job_id);
```

List the names of all the employees who have the same job type and work for the same manager as Peter Tucker.

```
SELECT first_name, last_name
FROM employees
WHERE (job_id, manager_id) IN
   (SELECT job_id, manager_id
    FROM employees
    WHERE upper(first_name) = 'PETER' AND
               upper(last_name) = 'TUCKER');
```

Types of Subqueries

- **Correlated Subqueries**: If the subquery references a column from a table referred to in the parent statement, the subquery is considered to be correlated.
- A correlated subquery is evaluated once for each row processed by the parent statement.
- The parent can be a SELECT, UPDATE or DELETE statement.

- Example:
  - Which employee earns the least in each department?

```
SELECT department_id, last_name, salary
FROM employees emp1
WHERE salary = (SELECT MIN(salary)
      FROM employees emp2
      WHERE emp1.department_id = emp2.department_id)
ORDER BY 1, 2, 3;
```

- Which employees have worked in other departments within the company? (Assume they have a job history.)

```
SELECT first_name, last_name
FROM employees e
WHERE EXISTS
   (SELECT 'x'
    FROM job_history
    WHERE employee_id = e.employee_id);
```

Types of Subqueries

- **Nested Subqueries**
- Can nest subqueries inside other subqueries to a depth of 255.
- Joins are more efficient if can use instead.

- **Scalar Subqueries**
- A scalar subquery returns exactly one column value from each row.
- The value of the scalar subquery is the value of the select list item of the subquery.
- Can use scalar subqueries in:
  - CASE expressions
  - SELECT statement
  - WHERE clause
  - ORDER BY clause
  - Functions
  - [VALUES clause of an INSERT statement]

- Rules
  - The degree of the scalar subquery should be one. If the subquery returns more than one row, it results in an error.
  - The type of the scalar subquery is the type of the column that is being returned by the subquery.
  - The data type of the return value of the subquery should match the data type of the column to which it is being compared to in the main query.
  - If the subquery returns zero rows, the value of the scalar subquery is NULL.

- In a CASE expression:
  - Want a list of the employee ID, last name, and country of employment for all employees. Are only really interested in those working in Canada (location_id 1800), so want 'Other' for all the rest.

```
SELECT employee_id, last_name,
    (CASE WHEN department_id IN
        (SELECT department_id
          FROM departments
          WHERE location_id = 1800)
         THEN 'Canada'
         ELSE 'Other'
      END) Location
 FROM employees
 ORDER BY department_id;
```

- In a SELECT list
  - Want to list the employee ID, last name, and the name of the department in which the employee works.

```
  SELECT employee_id, last_name,
       (SELECT department_name
         FROM departments d
         WHERE e.department_id = d.department_id) Department
   FROM employees e
   ORDER BY Department;
```

- In a WHERE clause
  - Want to list the employee ID and last name of the employees who work in departments that are located in the state of California. They all may be ‘terminated”.

```
SELECT employee_id, last_name
FROM employees e
WHERE ((SELECT location_id
          FROM departments d
          WHERE e.department_id = d.department_id) IN
             (SELECT location_id
              FROM locations l
              WHERE upper(state_province) = 'CALIFORNIA'));
```

- In the ORDER BY clause
  - Want a list of the employee ID and last names of all employees in ascending order by department name.

```
SELECT employee_id, last_name, department_id
FROM employees e
ORDER BY (SELECT department_name
         FROM departments d
         WHERE e.department_id = d.department_id);
```

- In Functions
  - Want to display the last name of employees and the first ten characters of the name of the department in which they work.

```
SELECT last_name, SUBSTR(
     (SELECT department_name
      FROM departments d
      WHERE d.department_id = e.department_id),
      1, 10) Department
FROM employees e;
```

Subqueries

- Summary
  - Described the types of problems that subqueries can solve
  - Listed the types of subqueries
    - Single-Row
    - Multiple-Row
    - Multiple-Column
    - Correlated Subqueries
    - Nested Subqueries
    - Scalar Subqueries

Cautionary Note

- Subqueries are expensive in terms of processing. Only use them when you must.
- Subquery performance has always been problematic for Oracle queries, and Oracle introduced global temporary tables to allow subqueries to be executed independently of the outer query, a powerful technique where you can hypercharge Oracle performance by re-writing subqueries to use temporary tables.