Table of Contents

Subqueries are a technique that we will develop over the semester. In this discussion we look at some simple queries that use an equality test or an In filter to test against the data is brought back by a subquery.

1. Subquery using an equality test

Suppose we want to find all of the people who work in the same department as employee 162. We could write a query to find that department id.

```
Select dept_id
From a_emp.employees
Where emp_id = 162;
```

And then write a query to find employees in that department.

```
Select emp_id
,        name_last as "Employee"
,        dept_id
From a_emp.employees
Where dept_id =    --  put the department number here
```

But we can also build these two queries into one query by using a subquery.

Demo 01: A simple subquery- we want to find employees who work in the same department as employee with ID 162

```
Select emp_id, name_last as "Employee"
From a_emp.employees
Where dept_id =
          (
           Select dept_id
           From a_emp.employees
           Where emp_id = 162
);
+--------+----------+
| emp_id | Employee |
+--------+----------+
|    162 | Holme    |
|    200 | Whale    |
|    207 | Russ     |
+--------+----------+
```

The inner query, the subquery, is a complete query that could stand by itself. It returns a single value- the department id where this employee works.

The subquery is enclosed in parentheses and the result of the subquery is used by the outer query to return the other employees from that department.

Demo 02: If we really want to return the **other** people from that department, we can eliminate employee 162 from the final result set. Note that the filter emp_id <> 162 is written outside the subquery.

```
Select emp_id, name_last as "Employee"
```

```
From a_emp.employees
Where dept_id =
         (
          Select dept_id
          From a_emp.employees
          Where emp_id = 162
          )
and emp_id <> 162;
+--------+----------+
| emp_id | Employee |
+--------+----------+
|    200 | Whale    |
|    207 | Russ     |
+--------+----------+
```

With the filter we have to take care that the subquery returns a single value- one row and one column, because we are using it in an = test.

We could use this technique to find employees who earn more than employee 162.

The subquery to find the salary for employee 162 would be

```
          Select salary
          From a_emp.employees
          Where emp_id = 162
```

Each employee has exactly one salary value and we have only one employee 162 since emp_id is the primary key. This subquery returns one value.

Demo 03: Using the subquery

```
Select emp_id, name_last as "Employee", salary
From a_emp.employees
Where salary >
         (
          Select salary
          From a_emp.employees
          Where emp_id = 162
          )
;
+--------+----------+----------+
| emp_id | Employee | salary   |
+--------+----------+----------+
|    101 | Koch     | 98005.00 |
+--------+----------+----------+
1 row in set (0.02 sec)
```

Try this query with some different employee id value- who earns more than employee 104? More than employee 101?

Demo 04: Now try this query; you will get an error message

```
Select emp_id, name_last as "Employee", salary
From a_emp.employees
Where salary >
         (
          Select salary
          From a_emp.employees
          Where dept_id = 210
          );
ERROR 1242 (21000): Subquery returns more than 1 row
```

This error message is correct. The subquery is being used in a test where a single-row subquery is expected. But the subquery returns more than one row,

The subquery is

```
Select salary
From a_emp.employees
Where dept_id = 210
```

We have two employees in dept 210, so the result of that subquery is

```
+----------+
| salary   |
+----------+
|  9000.00 |
| 50000.00 |
+----------+
2 rows in set (0.00 sec)
```

So are we asking who earns more than 9000 or more than 50000? This is an invalid query; you cannot use an = or a > test against a subquery that returns more than one row.

2. Subquery using an in list test instead of joins

Suppose we want to find all of the orders for sporting goods items. We could do this with an inner join.

Demo 05: a query to find the prod_id of all of the sporting goods items we have in the products table.

```
Select distinct ord_id
From a_oe.order_details od
Join a_prd.products     pr on od.prod_id = pr.prod_id
Where catg_id = 'spg'
Order by ord_id
;
+--------+
| ord_id |
+--------+
|    105 |
|    106 |
|    117 |
|    120 |
|    121 |
|    128 |
|    302 |
. . . rows omitted
```

Demo 06: We can also do this with a subquery that finds a list of product ids that are sporting goods from the products table and then delivers that list to the outer query which finds those product IDs in the order_details table. Since there is more than one product id for sporting goods, we need to use the IN list syntax.

```
Select distinct ord_id
From a_oe.order_details
Where prod_id in (
         Select prod_id
         From a_prd.products
         Where catg_id = 'SPG')
Order by ord_id;
```

This gives the same output as the previous demo.

This query would not work with an equality test in the outer query since the subquery returns multiple rows.

If we leave off the word Distinct then an order id will appear multiple times if a single order includes more than one sporting goods item line.

With the subquery approach the only attributes that can be displayed are those found in the tables in the From clause of the main query. We could display attributes from the order detail tables but not the attributes from the products table.