Implementing Subqueries

Jason P. Browning
EXECUTIVE DIRECTOR, DIXIE STATE UNIVERSITY
@jason_from_ky

Overview

Defining subqueries

Choosing between subqueries and joins

Aggregating values with subqueries

Correlated subqueries

Subquery

A nested query where the result of one query can be used in another query

```
SELECT a.store_name,
       a.store_location
  FROM stores a
 WHERE a.store_name IN
       (SELECT store_name
          FROM orders
         WHERE order_value > 500);
```

store_name	store_location
Target	Minneapolis, MN
Wal-Mart	Fayetteville, AR

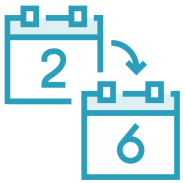
Stores

store_id	store_name	store_location
121	Wal-Mart	Fayetteville, AR
122	Target	Minneapolis, MN
123	Valu-Mart	Detroit, MI
124	Local Grocer	St. George, UT

Orders

order_id	store_name	order_value
901	Local Grocer	120
902	Target	550
903	Target	420
904	Wal-Mart	590
904	Valu-Mart	280

Subquery Guidelines



A subquery occurs in the SELECT, FROM, or WHERE clause

Can use comparison operators or multiple-row operators (such as IN)

Inner query (subquery) executes before outer query

Subquery or Join

The primary purpose of a join is to combine rows from tables based on a match condition

A subquery returns a single value (scalar) or row set that is immediately available for use in the outer query

```
SELECT a.store_name,
       a.store_location
  FROM stores a
 WHERE a.store_name IN
       (SELECT store_name
          FROM orders
         WHERE order_value > 500);

SELECT a.store_name,
       a.store_location
  FROM stores a
 INNER JOIN orders b
    ON a.store_name = b.store_name
 WHERE b.order_value > 500;
```

Subquery or Join

- A subquery and join can often be used to achieve the same result
- Consider performance implications

```
SELECT a.store_name,
       a.store_location
  FROM stores a
 WHERE a.store_name IN
       (SELECT store_name
          FROM orders
         WHERE order_value > 500);

SELECT a.store_name,
       a.store_location
  FROM stores a
 INNER JOIN orders b
    ON a.store_name = b.store_name
 WHERE b.order_value > 500;
```

Stores

store_id	store_name	store_location
121	Wal-Mart	Fayetteville, AR
122	Target	Minneapolis, MN
123	Valu-Mart	Detroit, MI
124	Local Grocer	St. George, UT

Orders

order_id	store_name	order_value
901	Local Grocer	$ 120
902	Target	$ 550
903	Target	$ 420
904	Wal-Mart	$ 590
904	Valu-Mart	$ 280

store_name	store_location
Target	Minneapolis, MN
Wal-Mart	Fayetteville, AR

Demo

Using subqueries

Subqueries are advantageous when you need to calculate aggregate values on-the-fly or for membership questions.

Calculating Aggregate Values

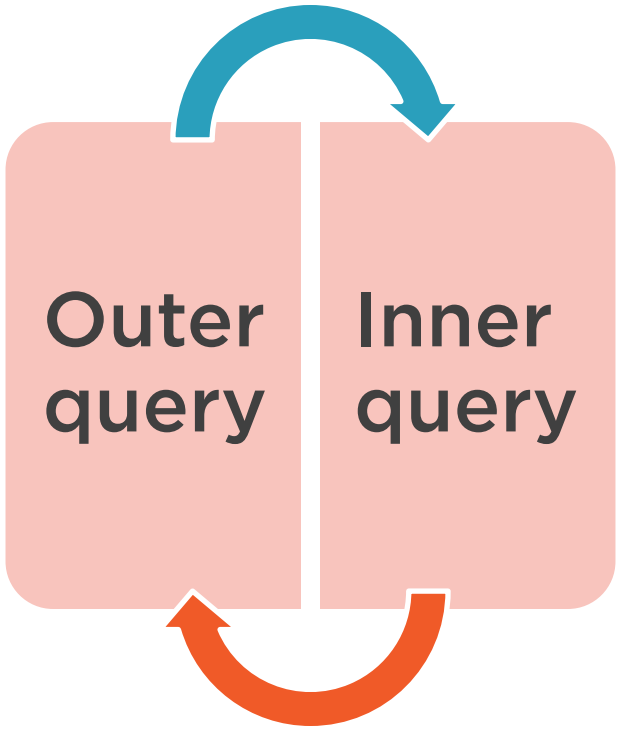
```
SELECT a.store_name,
       a.order_id
  FROM orders a
 WHERE a.order_value >
       (SELECT AVG(order_value)
          FROM orders);
```

Correlated Subqueries

A subquery that uses values from the primary query

Subquery is evaluated for each row processed by the primary query

Subqueries



In a regular subquery, outer query relies on values from the inner query

In a correlated subquery, inner query relies on values from the outer query

```
SELECT a.trans_id,
       b.first_name,
       b.last_name,
       a.amount
  FROM transactions a
INNER JOIN customers b
    ON a.cust_id = b.cust_id
 WHERE a.amount >
       (SELECT AVG(amount)
          FROM transactions aa
         WHERE a.cust_id = aa.cust_id)
 ORDER BY a.trans_id;
```

Correlated Subquery

- Inner query (subquery) calculates the average transaction amount for each customer
- Outer query retrieves customer name and specific transactions
- The WHERE clause limits results returned to transactions that exceed the average transaction amount for each customer

```
SELECT a.trans_id,
       b.first_name,
       b.last_name,
       a.amount
  FROM transactions a
INNER JOIN customers b
    ON a.cust_id = b.cust_id
 WHERE a.amount >
       (SELECT AVG(amount)
          FROM transactions aa
         WHERE a.cust_id = aa.cust_id)
 ORDER BY a.trans_id;
```

Transactions

trans_id	cust_id	trans_dt	amount
9100	101	01/03/00	$ 763.50
9101	101	01/07/00	$ 248.26
9102	103	01/02/00	$ 250.00
9103	104	01/09/00	$ 133.04
9104	104	01/16/00	$ 102.01
9103	104	01/28/00	$ 618.09

cust_id	AVG(amount)
101	$ 505.88
103	$ 250.00
104	$ 284.38

```
SELECT a.trans_id,
       b.first_name,
       b.last_name,
       a.amount
  FROM transactions a
INNER JOIN customers b
    ON a.cust_id = b.cust_id
 WHERE a.amount >
       (SELECT AVG(amount)
          FROM transactions aa
         WHERE a.cust_id = aa.cust_id)
 ORDER BY a.trans_id;
```

Transactions

trans_id	cust_id	trans_dt	amount
9100	101	01/03/00	$ 763.50
9101	101	01/07/00	$ 248.26
9102	103	01/02/00	$ 250.00
9103	104	01/09/00	$ 133.04
9104	104	01/16/00	$ 102.01
9103	104	01/28/00	$ 618.09

cust_id	AVG(amount)
101	$ 505.88
103	$ 250.00
104	$ 284.38

```
SELECT a.trans_id,
       b.first_name,
       b.last_name,
       a.amount
  FROM transactions a
INNER JOIN customers b
   ON a.cust_id = b.cust_id
WHERE a.amount >
      (SELECT AVG(amount)
         FROM transactions aa
        WHERE a.cust_id = aa.cust_id)
ORDER BY a.trans_id;
```

Transactions

trans_id	cust_id	trans_dt	amount
9100	101	01/03/00	$ 763.50
9103	104	01/28/00	$ 618.09

Customers

cust_id	first_name	last_name
101	John	Smith
104	Jacob	Marley

trans_id	first_name	last_name	amount
9100	John	Smith	$ 763.50
9103	Jacob	Marley	$ 618.09

Demo

Using correlated subqueries

Summary

Use subqueries when you need to use the result of one query in another query

Joins and subqueries are similar

Subqueries are best used for aggregation and membership questions

Correlated subqueries use values from the primary query