Nested Scalar Queries

This lesson discusses nested queries that result in a single value.

Nested Queries

In this lesson we'll examine nested queries, a query within another query. With the ability to *nest* a query, we can combine queries to get the desired result in a single uber query rather than executing each constituent query individually. Nested queries are generally slower but more readable and expressive than equivalent join queries. Also, in some situations nested queries are the only way to retrieve desired information from a database.

> Connect to the terminal below by clicking in the widget. Once connected, the command line prompt will show up. Enter or copy and paste the command **./DataJek/Lessons/30lesson.sh** and wait for the MySQL prompt to start-up.

```
-- The lesson queries are reproduced below for convenient copy/paste into the terminal.

-- Query 1
SELECT URL AS "Brad's Insta Page"
FROM Actors
INNER JOIN DigitalAssets
WHERE AssetType = "Instagram" AND FirstName  ="Brad";

-- Query 2
SELECT URL
FROM DigitalAssets
WHERE AssetType = "Instagram" AND
ActorId = (SELECT Id
           FROM Actors
           WHERE FirstName = "Brad");

-- Query 3
SELECT FirstName
```

```
FROM Actors
INNER JOIN DigitalAssets
ON ActorId = Id

WHERE LastUpdatedOn = (SELECT MAX(LastUpdatedOn)
                       FROM DigitalAssets);
```

Terminal

1. Many times, inner join queries can be rewritten as nested queries. For instance, let's say we want to find the Instagram page for Brad Pitt. One way to glean this information is to use the following inner join query:

```
SELECT URL AS "Brad's Insta Page"

FROM Actors

INNER JOIN DigitalAssets

WHERE AssetType = "Instagram" AND FirstName  ="Brad";
```

```
mysql> SELECT URL AS "Brad's Insta Page"
    ->
    -> FROM Actors
    ->
    -> INNER JOIN DigitalAssets
    ->
    -> WHERE AssetType = "Instagram" AND FirstName  ="Brad";
+-----------------------------------------+
| Brad's Insta Page                       |
+-----------------------------------------+
| https://www.instagram.com/bradpittoficial |
+-----------------------------------------+
1 row in set (0.00 sec)
```

The above join query can be rewritten as a nested query as follows:

```
SELECT URL

FROM DigitalAssets

WHERE AssetType = "Instagram" AND
```

```
ActorId = (SELECT Id

           FROM Actors
           WHERE FirstName = "Brad");
```

```
mysql> SELECT URL
    ->
    -> FROM DigitalAssets
    ->
    -> WHERE AssetType = "Instagram" AND
    ->
    -> ActorId = (SELECT Id
    ->             FROM Actors
    ->             WHERE FirstName = "Brad");
+-----------------------------------------+
| URL                                     |
+-----------------------------------------+
| https://www.instagram.com/bradpittoficial |
+-----------------------------------------+
1 row in set (0.00 sec)
```

The nested query is enclosed within parentheses. The result of the nested query is a single integer value that feeds into the WHERE condition of the outer query. Another alternative is to first execute the inner query and remember the output it produces, then execute the outer query and plug in the output of the inner query in the WHERE clause. When a subquery returns a single value, it is said to return a scalar operand. Nesting allows us to get the result in one shot. In the example above, if we change the sub-query to return multiple values or return the entire row, the overall query will fail as shown below:

```
mysql> SELECT URL FROM DigitalAssets WHERE ActorId = (SELECT * from Actors WHERE FirstName='Brad');
ERROR 1241 (21000): Operand should contain 1 column(s)
mysql>
mysql>
mysql>
mysql> SELECT URL FROM DigitalAssets WHERE ActorId = (SELECT Id from Actors WHERE NetWorth>Billions=240);
ERROR 1242 (21000): Subquery returns more than 1 row
```

2. Let's work with another example, say we are asked to return the actor who has most recently updated any of his or her online social accounts. Given the schema of our database we know the column

accounts. Given the schema of our database we know the column **LastUpdatedOn** stores the timestamp of when an actor last performed an activity on their online account. We could use the **MAX** function to find the row with the most recent activity. Next, we'll need to match the actor ID with the maximum value for the **LastUpdatedOn** column with a row with the same ID in the **Actors** table. This query can be written as an inner join query as follows:

```
SELECT FirstName

FROM Actors

INNER JOIN DigitalAssets

ON ActorId = Id

WHERE LastUpdatedOn = (SELECT MAX(LastUpdatedOn)
                       FROM DigitalAssets);
```

```
mysql> SELECT FirstName
    ->
    -> FROM Actors
    ->
    -> INNER JOIN DigitalAssets
    ->
    -> ON ActorId = Id
    ->
    -> WHERE LastUpdatedOn = (SELECT MAX(LastUpdatedOn)
    ->                        FROM DigitalAssets);
+-----------+
| FirstName |
+-----------+
| Brad      |
+-----------+
1 row in set (0.00 sec)
```

Note that if there is more than one actor with the most recent activity, the above query will return more than one row. This query demonstrates a scenario where we don't have an alternative other than a nested query to get our desired information in a single query execution.