Nested Row Queries

This lesson discusses nested queries that return a set of rows.

Nested Row Queries

In this lesson we'll study nested queries that return rows, allowing the outer query to match on multiple different column values. Furthermore, so far, we have used nested queries only with the **WHERE** clause, but now we'll also use them with the **FROM** clause.

> 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/32lesson.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 FirstName
FROM Actors
INNER JOIN DigitalAssets
ON Id=ActorId
AND MONTH(DoB) = MONTH(LastUpdatedOn)
AND DAY(DoB) = DAY(LastUpdatedOn);

-- Query 2
SELECT FirstName
FROM Actors
WHERE (Id, MONTH(DoB), DAY(DoB))
IN ( SELECT ActorId, MONTH(LastUpdatedOn), DAY(LastUpdatedOn)
     FROM DigitalAssets);

--Query 3
SELECT ActorId, AssetType, LastUpdatedOn FROM DigitalAssets;

-- Query 4
SELECT FirstName, AssetType, LastUpdatedOn
```

```
SELECT FirstName, AssetType, LastUpdatedOn
FROM Actors
INNER JOIN (SELECT ActorId, AssetType, LastUpdatedOn
            FROM DigitalAssets) AS tbl
ON ActorId = Id;

-- Query 5
SELECT FirstName, AssetType, LastUpdatedOn
FROM Actors
INNER JOIN (SELECT ActorId, AssetType, LastUpdatedOn
            FROM DigitalAssets) AS tbl
ON ActorId = Id
WHERE FirstName = "Kim";

-- Query 6
SELECT FirstName, AssetType, LastUpdatedOn
FROM Actors
INNER JOIN (SELECT ActorId, AssetType, LastUpdatedOn
            FROM DigitalAssets) AS tbl
ON ActorId = Id
WHERE FirstName = "Kim"
ORDER BY LastUpdatedOn DESC LIMIT 1;
```

Terminal

1. Let's say we want to find the list of all the actors whose latest update to any of their online accounts was on the day of their birthday. The date of birth for each actor is in the **Actors** table and the **LastUpdatedOn** column is in the **DigitalAssets** table. We can extract the birthday month and day using the **MONTH()** and **DAY()** functions on the **DoB** column and match them with the corresponding extracted values from the **LastUpdatedOn** column. Finally, we'll also need to match the actor IDs in the two tables. The inner join query to get the results is as follows:

   ```
   SELECT FirstName
   FROM Actors
   INNER JOIN DigitalAssets
   ON Id=ActorId
   AND MONTH(DoB) = MONTH(LastUpdatedOn)
   AND DAY(DoB) = DAY(LastUpdatedOn);
   ```

```
mysql> SELECT FirstName
    -> FROM Actors
    -> INNER JOIN DigitalAssets
    -> ON Id=ActorId
    -> AND MONTH(DoB) = MONTH(LastUpdatedOn)
    -> AND DAY(DoB) = DAY(LastUpdatedOn);
+-----------+
| FirstName |
+-----------+
| Natalie   |
| Shahrukh  |
| Natalie   |
| Angelina  |
| Natalie   |
+-----------+
5 rows in set (0.00 sec)
```

Instead of the inner join we can also use a nested query. We'll return three columns from the inner query, the day and month of the last update and the actor ID. The outer query will match on these three columns using the **IN** clause.

```
SELECT FirstName

FROM Actors

WHERE (Id, MONTH(DoB), DAY(DoB))

IN ( SELECT ActorId, MONTH(LastUpdatedOn), DAY(LastUpdatedOn)
     FROM DigitalAssets);
```

```
mysql> SELECT FirstName
    ->
    -> FROM Actors
    ->
    -> WHERE (Id, MONTH(DoB), DAY(DoB))
    ->
    -> IN ( SELECT ActorId, MONTH(LastUpdatedOn), DAY(LastUpdatedOn)
    ->      FROM DigitalAssets);
+-----------+
| FirstName |
+-----------+
| Angelina  |
| Natalie   |
| Shahrukh  |
+-----------+
3 rows in set (0.00 sec)
```

The inner query returns a temporary result set of several rows with three columns. The outer query lists columns from the Actors table that should be matched against the columns from the result set of the inner query and the match takes place in the order of the listing of columns. The syntax allows us to match multiple columns per row for several rows. The first name column of the matching rows of the inner query's result set are then returned as the result of the overall query.

2. To demonstrate using a nested query with the **FROM** clause, we'll move onto a slightly harder query to answer. Say you are asked to find out which of her online accounts Kim Kardashian most recently updated. Let's think about it for a minute: the two pieces of information we need are present in the two tables: Actors (name) and DigitalAssets (last update timestamp). First, let's understand how we can find the latest updated account for an actor. If we know the ActorID, we can use the following query to list all the online accounts belonging to that actor along with their latest update times.

```
SELECT ActorId, AssetType, LastUpdatedOn FROM DigitalAssets;
```

```
mysql> SELECT ActorId, AssetType, LastUpdatedOn FROM DigitalAssets;
+---------+-----------+---------------------+
| ActorId | AssetType | LastUpdatedOn       |
+---------+-----------+---------------------+
|       2 | Website   | 2019-10-11 23:14:05 |
|       3 | Website   | 2019-05-01 12:54:02 |
|       6 | Website   | 2019-10-23 09:56:33 |
|      10 | Twitter   | 2019-08-18 18:39:08 |
|       2 | Twitter   | 2019-02-13 03:04:25 |
|       3 | Twitter   | 2019-02-03 00:04:25 |
|       8 | Twitter   | 2019-08-28 22:19:33 |
|       5 | Twitter   | 2019-06-09 10:12:21 |
|       6 | Twitter   | 2018-07-05 06:16:12 |
|       1 | Website   | 2020-01-01 23:12:13 |
|      10 | Facebook  | 2019-11-02 01:00:54 |
|       2 | Facebook  | 2019-11-11 15:00:00 |
|       4 | Website   | 2018-07-11 17:17:18 |
|       8 | Facebook  | 2019-09-04 18:07:38 |
|       5 | Facebook  | 2019-06-09 09:14:20 |
|       6 | Facebook  | 2019-10-28 19:39:40 |
|       1 | Instagram | 2019-05-15 16:25:02 |
|       8 | Website   | 2018-03-15 13:25:00 |
|       5 | Website   | 2018-11-24 15:06:59 |
|       3 | Pinterest | 2019-06-04 03:44:36 |
|       5 | Pinterest | 2019-06-09 09:14:20 |
+---------+-----------+---------------------+
21 rows in set (0.00 sec)
```

The above query also retrieves us the actor IDs. If we could determine Kardashian's actor ID from the output of the above query, we could use that in a WHERE clause and answer the original question, but we don't. We'll need to join the result of the above query with the actor table based on actor IDs to know which rows from the **DigitalAssets** table belong to Kardashian. So far, we have:

```
SELECT FirstName, AssetType, LastUpdatedOn

FROM Actors

INNER JOIN (SELECT ActorId, AssetType, LastUpdatedOn
            FROM DigitalAssets) AS tbl

ON ActorId = Id;
```

```
mysql> SELECT FirstName, AssetType, LastUpdatedOn
    ->
    -> FROM Actors
    ->
    -> INNER JOIN (SELECT ActorId, AssetType, LastUpdatedOn
    ->             FROM DigitalAssets) AS tbl
    ->
    -> ON ActorId = Id;
+-----------+-----------+---------------------+
| FirstName | AssetType | LastUpdatedOn       |
+-----------+-----------+---------------------+
| Jennifer  | Website   | 2019-10-11 23:14:05 |
| Angelina  | Website   | 2019-05-01 12:54:02 |
| Tom       | Website   | 2019-10-23 09:56:33 |
| Shahrukh  | Twitter   | 2019-08-18 18:39:08 |
| Jennifer  | Twitter   | 2019-02-13 03:04:25 |
| Angelina  | Twitter   | 2019-02-03 00:04:25 |
| Kim       | Twitter   | 2019-08-28 22:19:33 |
| Natalie   | Twitter   | 2019-06-09 10:12:21 |
| Tom       | Twitter   | 2018-07-05 06:16:12 |
| Brad      | Website   | 2020-01-01 23:12:13 |
| Shahrukh  | Facebook  | 2019-11-02 01:00:54 |
| Jennifer  | Facebook  | 2019-11-11 15:00:00 |
| Johnny    | Website   | 2018-07-11 17:17:18 |
| Kim       | Facebook  | 2019-09-04 18:07:38 |
| Natalie   | Facebook  | 2019-06-09 09:14:20 |
| Tom       | Facebook  | 2019-10-28 19:39:40 |
| Brad      | Instagram | 2019-05-15 16:25:02 |
| Kim       | Website   | 2018-03-15 13:25:00 |
| Natalie   | Website   | 2018-11-24 15:06:59 |
| Angelina  | Pinterest | 2019-06-04 03:44:36 |
| Natalie   | Pinterest | 2019-06-09 09:14:20 |
+-----------+-----------+---------------------+
21 rows in set (0.00 sec)
```

Note that we give an alias of tbl to the result set of the inner query. When the result of an inner query is used as a derived table, MySQL requires us to provide an alias for the table. This is a syntax requirement. If we skip aliasing the result set of the inner query, we'll not be able to use it in the join clause. In order to narrow down the rows for Kardashian we'll need to add a WHERE clause with the condition **FirstName="Kim"**. Now we'll have all the digital accounts belonging to Kardashian as follows:

belonging to Kardashian as follows.

```
SELECT FirstName, AssetType, LastUpdatedOn

FROM Actors

INNER JOIN (SELECT ActorId, AssetType, LastUpdatedOn
            FROM DigitalAssets) AS tbl

ON ActorId = Id

WHERE FirstName = "Kim";
```

```
mysql> SELECT FirstName, AssetType, LastUpdatedOn
    ->
    -> FROM Actors
    ->
    -> INNER JOIN (SELECT ActorId, AssetType, LastUpdatedOn
    ->             FROM DigitalAssets) AS tbl
    ->
    -> ON ActorId = Id
    ->
    -> WHERE FirstName = "Kim";
+-----------+-----------+---------------------+
| FirstName | AssetType | LastUpdatedOn       |
+-----------+-----------+---------------------+
| Kim       | Twitter   | 2019-08-28 22:19:33 |
| Kim       | Facebook  | 2019-09-04 18:07:38 |
| Kim       | Website   | 2018-03-15 13:25:00 |
+-----------+-----------+---------------------+
3 rows in set (0.00 sec)
```

The last piece is to order the rows by **LastUpdatedOn** to get the latest updated online account for Kardashian.

```
SELECT FirstName, AssetType, LastUpdatedOn

FROM Actors

INNER JOIN (SELECT ActorId, AssetType, LastUpdatedOn
            FROM DigitalAssets) AS tbl

ON ActorId = Id

WHERE FirstName = "Kim"
```

```
ORDER BY LastUpdatedOn DESC LIMIT 1;
```

```
mysql> SELECT FirstName, AssetType, LastUpdatedOn
    ->
    -> FROM Actors
    ->
    -> INNER JOIN (SELECT ActorId, AssetType, LastUpdatedOn
    ->              FROM DigitalAssets) AS tbl
    ->
    -> ON ActorId = Id
    ->
    -> WHERE FirstName = "Kim"
    ->
    -> ORDER BY LastUpdatedOn DESC LIMIT 1;
+-----------+-----------+---------------------+
| FirstName | AssetType | LastUpdatedOn       |
+-----------+-----------+---------------------+
| Kim       | Facebook  | 2019-09-04 18:07:38 |
+-----------+-----------+---------------------+
1 row in set (0.00 sec)
```

The astute reader would realize that in the **FROM** clause we could have just as well used the **DigitalAssets** table instead of plugging in a nested query. Sure, we could, but the intent here is to demonstrate how nested queries can be used with the **FROM** clause, so in that sense, it is a slightly contrived example.