

Nested Column Queries

This lesson discusses nested queries that return a column of values.

Nested Column Queries

In the previous lesson we examined nested queries that returned a single value. In this lesson we'll see nested queries that return values belonging to the same column.

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/31lesson.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 * FROM Actors
INNER JOIN DigitalAssets ON ActorId=Id
WHERE AssetType = ANY (SELECT DISTINCT AssetType
                       FROM DigitalAssets
                       WHERE AssetType != 'Website');

-- Query 2
SELECT * FROM Actors
INNER JOIN DigitalAssets ON ActorId=Id
WHERE AssetType != 'Website';

-- Query 3
SELECT FirstName, SecondName
FROM Actors
WHERE Id = ANY (SELECT ActorId
                FROM DigitalAssets
                WHERE AssetType = 'Facebook');

-- Query 4
SELECT FirstName, SecondName
```

```

FROM Actors
WHERE Id IN (SELECT ActorId
             FROM DigitalAssets
             WHERE AssetType = 'Facebook');

-- Query 5
SELECT FirstName, SecondName
FROM Actors
WHERE NetworthInMillions > ALL (SELECT NetworthInMillions
                                FROM Actors
                                WHERE FirstName LIKE "j%");

```

Terminal

1. We'll use a slightly contrived example this time. Imagine we want to list all the social media accounts for all the actors, except for their personal websites. From our database schema we know that the table **DigitalAssets** has a column **AssetType**, which is essentially an enum and has a value **website** to denote an actor's personal website. The DigitalAssets table by itself can't give us the names of the actors since it only contains actor IDs. We'll require an inner join with the **Actors** table to get the actor names. The complete query is shown below:

```

SELECT * FROM Actors

INNER JOIN DigitalAssets ON ActorId=Id

WHERE AssetType = ANY (SELECT DISTINCT AssetType
                       FROM DigitalAssets
                       WHERE AssetType != 'Website');

```

mysql> SELECT * FROM Actors

mysql> SELECT Id FROM DigitalAssets WHERE ActorId=Id

mysql> SELECT * FROM DigitalAssets WHERE AssetType != 'Website'

Id	FirstName	SecondName	Id	AssetType	NetworthInMillions
1	John	Doe	1	Facebook	100
2	Jane	Doe	2	Facebook	100
3	John	Doe	3	Facebook	100
4	Jane	Doe	4	Facebook	100
5	John	Doe	5	Facebook	100
6	Jane	Doe	6	Facebook	100
7	John	Doe	7	Facebook	100
8	Jane	Doe	8	Facebook	100
9	John	Doe	9	Facebook	100
10	Jane	Doe	10	Facebook	100
11	John	Doe	11	Facebook	100
12	Jane	Doe	12	Facebook	100
13	John	Doe	13	Facebook	100
14	Jane	Doe	14	Facebook	100
15	John	Doe	15	Facebook	100
16	Jane	Doe	16	Facebook	100
17	John	Doe	17	Facebook	100
18	Jane	Doe	18	Facebook	100
19	John	Doe	19	Facebook	100
20	Jane	Doe	20	Facebook	100
21	John	Doe	21	Facebook	100
22	Jane	Doe	22	Facebook	100
23	John	Doe	23	Facebook	100
24	Jane	Doe	24	Facebook	100
25	John	Doe	25	Facebook	100
26	Jane	Doe	26	Facebook	100
27	John	Doe	27	Facebook	100
28	Jane	Doe	28	Facebook	100
29	John	Doe	29	Facebook	100
30	Jane	Doe	30	Facebook	100
31	John	Doe	31	Facebook	100
32	Jane	Doe	32	Facebook	100
33	John	Doe	33	Facebook	100
34	Jane	Doe	34	Facebook	100
35	John	Doe	35	Facebook	100
36	Jane	Doe	36	Facebook	100
37	John	Doe	37	Facebook	100
38	Jane	Doe	38	Facebook	100
39	John	Doe	39	Facebook	100
40	Jane	Doe	40	Facebook	100
41	John	Doe	41	Facebook	100
42	Jane	Doe	42	Facebook	100
43	John	Doe	43	Facebook	100
44	Jane	Doe	44	Facebook	100
45	John	Doe	45	Facebook	100
46	Jane	Doe	46	Facebook	100
47	John	Doe	47	Facebook	100
48	Jane	Doe	48	Facebook	100
49	John	Doe	49	Facebook	100
50	Jane	Doe	50	Facebook	100
51	John	Doe	51	Facebook	100
52	Jane	Doe	52	Facebook	100
53	John	Doe	53	Facebook	100
54	Jane	Doe	54	Facebook	100
55	John	Doe	55	Facebook	100
56	Jane	Doe	56	Facebook	100
57	John	Doe	57	Facebook	100
58	Jane	Doe	58	Facebook	100
59	John	Doe	59	Facebook	100
60	Jane	Doe	60	Facebook	100
61	John	Doe	61	Facebook	100
62	Jane	Doe	62	Facebook	100
63	John	Doe	63	Facebook	100
64	Jane	Doe	64	Facebook	100
65	John	Doe	65	Facebook	100
66	Jane	Doe	66	Facebook	100
67	John	Doe	67	Facebook	100
68	Jane	Doe	68	Facebook	100
69	John	Doe	69	Facebook	100
70	Jane	Doe	70	Facebook	100
71	John	Doe	71	Facebook	100
72	Jane	Doe	72	Facebook	100
73	John	Doe	73	Facebook	100
74	Jane	Doe	74	Facebook	100
75	John	Doe	75	Facebook	100
76	Jane	Doe	76	Facebook	100
77	John	Doe	77	Facebook	100
78	Jane	Doe	78	Facebook	100
79	John	Doe	79	Facebook	100
80	Jane	Doe	80	Facebook	100
81	John	Doe	81	Facebook	100
82	Jane	Doe	82	Facebook	100
83	John	Doe	83	Facebook	100
84	Jane	Doe	84	Facebook	100
85	John	Doe	85	Facebook	100
86	Jane	Doe	86	Facebook	100
87	John	Doe	87	Facebook	100
88	Jane	Doe	88	Facebook	100
89	John	Doe	89	Facebook	100
90	Jane	Doe	90	Facebook	100
91	John	Doe	91	Facebook	100
92	Jane	Doe	92	Facebook	100
93	John	Doe	93	Facebook	100
94	Jane	Doe	94	Facebook	100
95	John	Doe	95	Facebook	100
96	Jane	Doe	96	Facebook	100
97	John	Doe	97	Facebook	100
98	Jane	Doe	98	Facebook	100
99	John	Doe	99	Facebook	100
100	Jane	Doe	100	Facebook	100

The subquery returns all the enum values for the column **AssetType**

except the value "Website". The **WHERE** clause of the outer query sets up a condition which evaluates to true whenever the column **AssetType** of the resulting inner join equals *any* of the values returned by the inner query. The **ANY** operator allows us to match the column **AssetType** with *any one of* the values returned for the column **AssetType**.

Granted, the same query can be written much simpler as follows without the need for an inner query, but the intention was to demonstrate a column subquery.

```
-- A much simpler approach to get the same result

SELECT * FROM Actors

INNER JOIN DigitalAssets ON ActorId=Id

WHERE AssetType != 'Website';
```

mysql> SELECT * FROM Actors

mysql> SELECT * FROM DigitalAssets

ActorId	FirstName	SecondName	Id	AssetType	AssetValue	AssetType	AssetValue
1	John	Doe	1	Facebook	John Doe	Facebook	John Doe
2	Jane	Smith	2	Twitter	Jane Smith	Twitter	Jane Smith
3	Mike	Johnson	3	LinkedIn	Mike Johnson	LinkedIn	Mike Johnson
4	Sarah	Williams	4	Instagram	Sarah Williams	Instagram	Sarah Williams
5	David	Brown	5	YouTube	David Brown	YouTube	David Brown
6	Emily	Green	6	Facebook	Emily Green	Facebook	Emily Green
7	Robert	White	7	Twitter	Robert White	Twitter	Robert White
8	Laura	Black	8	LinkedIn	Laura Black	LinkedIn	Laura Black
9	James	Grey	9	Instagram	James Grey	Instagram	James Grey
10	Maria	Gold	10	YouTube	Maria Gold	YouTube	Maria Gold
11	Chris	Silver	11	Facebook	Chris Silver	Facebook	Chris Silver
12	Alex	Copper	12	Twitter	Alex Copper	Twitter	Alex Copper
13	Patricia	Iron	13	LinkedIn	Patricia Iron	LinkedIn	Patricia Iron
14	Thomas	Steel	14	Instagram	Thomas Steel	Instagram	Thomas Steel
15	Michelle	Aluminum	15	YouTube	Michelle Aluminum	YouTube	Michelle Aluminum
16	Kevin	Brass	16	Facebook	Kevin Brass	Facebook	Kevin Brass
17	Nancy	Gold	17	Twitter	Nancy Gold	Twitter	Nancy Gold
18	Steven	Silver	18	LinkedIn	Steven Silver	LinkedIn	Steven Silver
19	Angela	Copper	19	Instagram	Angela Copper	Instagram	Angela Copper
20	Isabella	Iron	20	YouTube	Isabella Iron	YouTube	Isabella Iron

- Let's work another example. Say we now want to find the names of all the actors that have a Facebook presence. One way we can answer this query is to first collect all the actor IDs from the **DigitalAssets** table that have Facebook asset types. Next, we select all those rows from the **Actors** table whose ID matches any of the IDs from the first query:

```
SELECT FirstName, SecondName

FROM Actors

WHERE Id = ANY (SELECT ActorId
                FROM DigitalAssets
                WHERE AssetType = 'Facebook');
```

```
mysql> SELECT FirstName, SecondName
->
-> FROM Actors
->
-> WHERE Id = ANY (SELECT ActorId
->                  FROM DigitalAssets
->                  WHERE AssetType = 'Facebook');
```

FirstName	SecondName
Jennifer	Aniston
Natalie	Portman
Tom	Cruise
Kim	Kardashian
Shahrukh	Khan

```
5 rows in set (0.00 sec)
```

The **ANY** clause has an alias **IN** that can be used interchangeably. We can rewrite the above query as follows:

```
SELECT FirstName, SecondName

FROM Actors

WHERE Id IN (SELECT ActorId
             FROM DigitalAssets
             WHERE AssetType = 'Facebook');
```



```
mysql> SELECT FirstName, SecondName
-> FROM Actors
-> WHERE NetworthInMillions > ALL (SELECT NetworthInMillions
-> FROM Actors
-> WHERE FirstName LIKE "j%");
```

FirstName	SecondName
Tom	Cruise
Kylie	Jenner
Kim	Kardashian
Amitabh	Bachchan
Shahrukh	Khan

```
5 rows in set (0.00 sec)
```