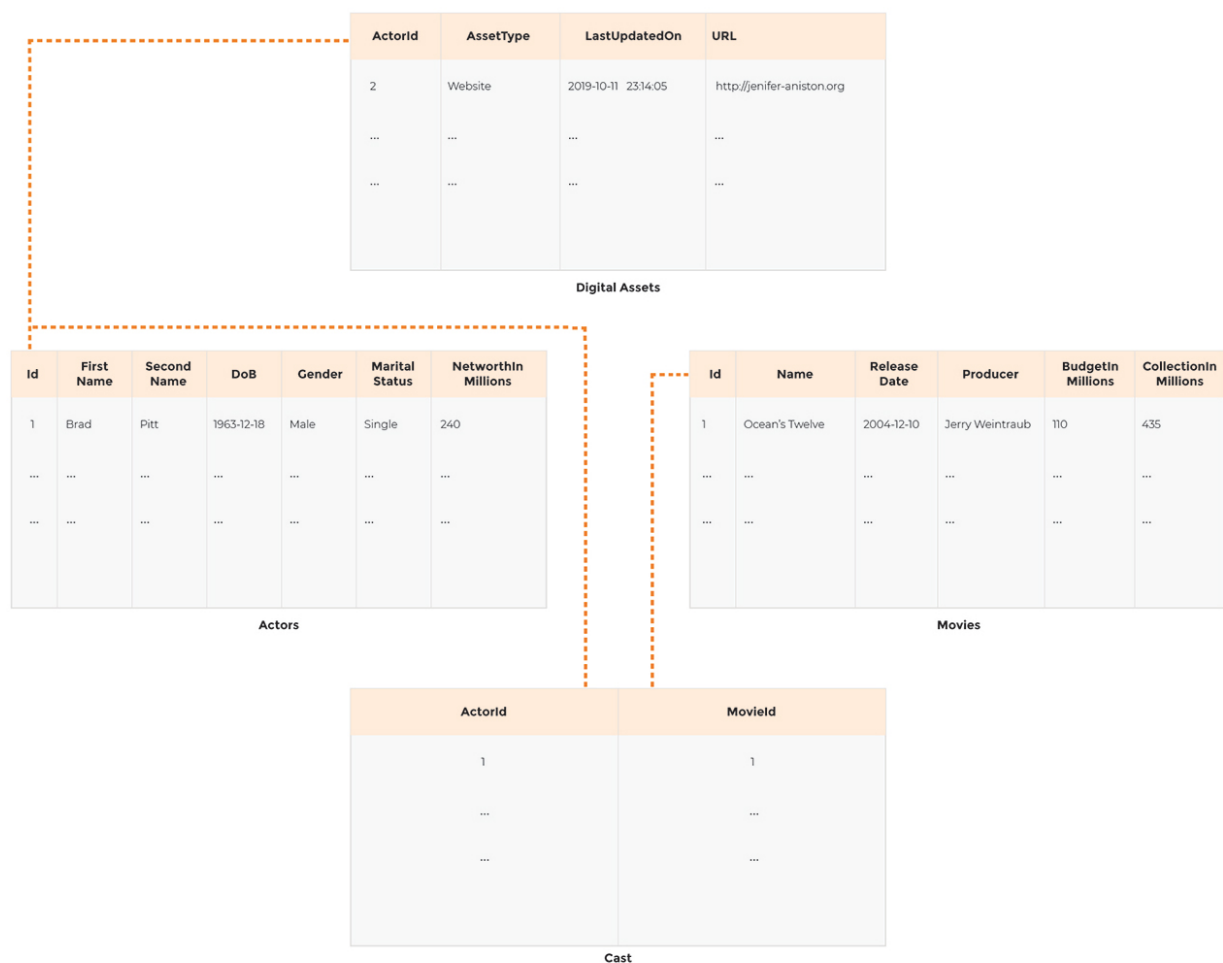


Solution Practice Set 1

Solution Practice Set 1

The database relationship model is reprinted below for reference.



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/quiz.sh** and wait for the MySQL prompt to

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



-- Question # 1, Query 1

```
SELECT Name
FROM Movies
ORDER BY CollectionInMillions DESC
LIMIT 3;
```

-- Question # 2, Query 1

```
SELECT * FROM Movies a
INNER JOIN Movies b;
```

-- Question # 2, Query 2

```
SELECT concat(a.FirstName, " ", b.SecondName)
FROM Actors a
INNER JOIN Actors b
ON a.SecondName = b.SecondName
WHERE a.ID != b.ID;
```

-- Question # 2, Query 3

```
SELECT DISTINCT concat(a.FirstName, " ", b.SecondName)
AS Actors_With_Shared_SecondNames
FROM Actors a
INNER JOIN Actors b
ON a.SecondName = b.SecondName
WHERE a.Id != b.Id;
```

-- Question # 3, Query 1

```
SELECT b.SecondName
FROM Actors a
INNER JOIN Actors b
ON a.SecondName = b.SecondName
WHERE a.Id != b.Id
GROUP BY b.SecondName;
```

-- Question # 3, Query 2

```
SELECT a.SecondName,
COUNT(DISTINCT a.FirstName)
FROM Actors a
INNER JOIN Actors b
ON a.SecondName = b.SecondName
WHERE a.Id != b.Id
group by a.SecondName;
```

-- Question # 3, Query 3

```
SELECT a.SecondName AS Actors_With_Shared_SecondNames,
COUNT(DISTINCT a.Id) AS Count
FROM Actors a
INNER JOIN Actors b
ON a.SecondName = b.SecondName
WHERE a.Id != b.Id
group by a.SecondName;
```

-- Question # 4, Query 1

```
SELECT DISTINCT CONCAT(FirstName, " ", SecondName) AS Actors_Acted_In_Atleast_1_Movies
FROM Actors
INNER JOIN Cast
ON Id = ActorId;
```

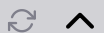
```
-- Question # 5, Query 1
SELECT Id, CONCAT(FirstName, " ", SecondName) AS Actors_With_No_Movies

FROM Actors
WHERE Id NOT IN (SELECT Id
                  FROM Actors
                  INNER JOIN Cast
                  ON Id = ActorId);

-- Question # 5, Query 2
SELECT *
FROM Actors
LEFT JOIN Cast
ON Id = ActorId;

-- Question # 5, Query 3
SELECT CONCAT(FirstName, " ", SecondName) AS Actors_With_No_Movies
FROM Actors
LEFT JOIN Cast
ON Id = ActorId
WHERE MovieId IS NULL;
```

Terminal



Question # 1

Write a query that prints the top three movies by box office collection?

This asks for us to print the top three movies and should hint towards sorting. The sort key should be the column **CollectionInMillions**. However, remember that by default, **ORDER BY** sorts in ascending order so we'll need to sort in descending order. The last piece to the puzzle is to apply the **LIMIT** clause so that we only retrieve the top three rows.

```
SELECT Name
FROM Movies
ORDER BY CollectionInMillions DESC
LIMIT 3;
```

```
mysql> SELECT Name
-> FROM Movies
-> ORDER BY CollectionInMillions DESC
-> LIMIT 3;
```

```
+-----+
| Name                                |
+-----+
| Avengers: Endgame                  |
| Mission: Impossible - Fallout     |
| Mr & Mrs. Smith                    |
+-----+
3 rows in set (0.00 sec)
```

Question # 2

Can you write a query to determine if any two actors share the same second name?

The information we want to extract is contained within the **Actors** table. However, we need a way to compare the second name of the first actor with all the other actors in the table except with itself. How can you make the comparison? The answer is using a self join.

Whenever you hear yourself thinking in terms of picking up a row from a table and comparing it to another row from the same table or another table, you are looking for a join. Without further ado, we'll perform an inner join of the Actors table.

```
SELECT * FROM Movies a
INNER JOIN Movies b;
```

Only part of the join is shown below because of limited space.

```
mysql> SELECT * FROM Movies a
--> INNER JOIN Movies b;
```

Name	ReleaseDate	Producer	Id	BudgetInMillions	CollectionInMillions	Name	ReleaseDate	Producer	Id	BudgetInMillions	CollectionInMillions
Ocean's Twelve	2004-12-10	Jerry Weintraub	1	110	435.00000	Ocean's Twelve	2004-12-10	Jerry Weintraub	1	110	435.00000
Mr & Mrs. Smith	2005-06-07	Arnon Milchan	2	110	478.20000	Ocean's Twelve	2004-12-10	Jerry Weintraub	1	110	435.00000
Mission: Impossible - Fallout	2018-07-12	J. J. Abrams	3	178	791.10000	Ocean's Twelve	2004-12-10	Jerry Weintraub	1	110	435.00000
London Fields	2018-09-20	Chris Hanley	4	8	0.48700	Ocean's Twelve	2004-12-10	Jerry Weintraub	1	110	435.00000
Ocean's 11	2001-12-07	Jerry Weintraub	5	85	450.70000	Ocean's Twelve	2004-12-10	Jerry Weintraub	1	110	435.00000
Ocean's 11	1960-08-03	Lewis Milestone	6	3	5.00000	Ocean's Twelve	2004-12-10	Jerry Weintraub	1	110	435.00000
Regarding Henry	1991-07-12	J. J. Abrams	7	25	43.00000	Ocean's Twelve	2004-12-10	Jerry Weintraub	1	110	435.00000
Mohabbatein	2000-10-27	Yash Chopra	8	3	12.55000	Ocean's Twelve	2004-12-10	Jerry Weintraub	1	110	435.00000
Keeping Up with the Kardashians	2007-10-14	Ryan Seacrest	9	100	300.00000	Ocean's Twelve	2004-12-10	Jerry Weintraub	1	110	435.00000
Avengers: Endgame	2019-04-22	Kevin Feige	10	356	2798.00000	Ocean's Twelve	2004-12-10	Jerry Weintraub	1	110	435.00000
Ocean's Twelve	2004-12-10	Jerry Weintraub	1	110	435.00000	Mr & Mrs. Smith	2005-06-07	Arnon Milchan	2	110	478.20000
Mr & Mrs. Smith	2005-06-07	Arnon Milchan	2	110	478.20000	Mr & Mrs. Smith	2005-06-07	Arnon Milchan	2	110	478.20000
Mission: Impossible - Fallout	2018-07-12	J. J. Abrams	3	178	791.10000	Mr & Mrs. Smith	2005-06-07	Arnon Milchan	2	110	478.20000
London Fields	2018-09-20	Chris Hanley	4	8	0.48700	Mr & Mrs. Smith	2005-06-07	Arnon Milchan	2	110	478.20000
Ocean's 11	2001-12-07	Jerry Weintraub	5	85	450.70000	Mr & Mrs. Smith	2005-06-07	Arnon Milchan	2	110	478.20000
Ocean's 11	1960-08-03	Lewis Milestone	6	3	5.00000	Mr & Mrs. Smith	2005-06-07	Arnon Milchan	2	110	478.20000
Regarding Henry	1991-07-12	J. J. Abrams	7	25	43.00000	Mr & Mrs. Smith	2005-06-07	Arnon Milchan	2	110	478.20000
Mohabbatein	2000-10-27	Yash Chopra	8	3	12.55000	Mr & Mrs. Smith	2005-06-07	Arnon Milchan	2	110	478.20000
Keeping Up with the Kardashians	2007-10-14	Ryan Seacrest	9	100	300.00000	Mr & Mrs. Smith	2005-06-07	Arnon Milchan	2	110	478.20000
Avengers: Endgame	2019-04-22	Kevin Feige	10	356	2798.00000	Mr & Mrs. Smith	2005-06-07	Arnon Milchan	2	110	478.20000
Ocean's Twelve	2004-12-10	Jerry Weintraub	1	110	435.00000	Mission: Impossible - Fallout	2018-07-12	J. J. Abrams	3	178	791.10000
Mr & Mrs. Smith	2005-06-07	Arnon Milchan	2	110	478.20000	Mission: Impossible - Fallout	2018-07-12	J. J. Abrams	3	178	791.10000

Notice we didn't specify any condition for the inner join and the result is a cartesian product. Every row from the table is compared or *joined* with every other row including itself. Since we are joining on the table itself, we have to specify aliases **a** and **b** to refer to the table and its copy to avoid ambiguity. This is what we wanted. Now we'll apply our criteria or check to narrow down to rows we want.

- We don't want each row to compare to itself so our condition can be **a.Id != b.Id**.
- The other condition is to find those actors that have matching second names. We can specify that as **a.SecondName = b.SecondName**.

Let's see what we have so far:

```
SELECT concat(a.FirstName, " ", b.SecondName)
FROM Actors a
INNER JOIN Actors b
ON a.SecondName = b.SecondName
WHERE a.ID != b.ID;
```

```
mysql> SELECT concat(a.FirstName, " ", b.SecondName)
-> FROM Actors a
-> INNER JOIN Actors b
-> ON a.SecondName = b.SecondName
-> WHERE a.Id != b.Id;
```

concat(a.FirstName, " ", b.SecondName)
Khloe Kardashian
Kourtney Kardashian
Abhishek Bachchan
Kim Kardashian
Kourtney Kardashian
Kim Kardashian
Khloe Kardashian
Amitabh Bachchan

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

We are able to print the names of the actors with same second names, however, each name is printed twice. This is because when we find two actors, say **X** and **Y**, with matching second names, the matches also take place when we process the row for actor **Y**. So, one match happens when we process **X** and **Y** and the second match happens when we process **Y** and **X**. The final piece we need is to add the **DISTINCT** clause so that all names are printed only once.

```
SELECT DISTINCT concat(a.FirstName, " ", b.SecondName)
AS Actors_With_Shared_SecondNames
FROM Actors a
INNER JOIN Actors b
ON a.SecondName = b.SecondName
WHERE a.Id != b.Id;
```

```
mysql> SELECT DISTINCT concat(a.FirstName, " ", b.SecondName)
-> AS Actors_With_Shared_SecondNames
-> FROM Actors a
-> INNER JOIN Actors b
-> ON a.SecondName = b.SecondName
-> WHERE a.Id != b.Id;

+-----+
| Actors_With_Shared_SecondNames |
+-----+
| Khloe Kardashian                |
| Kourtney Kardashian             |
| Abhishek Bachchan               |
| Kim Kardashian                  |
| Amitabh Bachchan                |
+-----+
5 rows in set (0.00 sec)
```

Question # 3

Write a query to count the number of actors who share the same second name. Print the second name along with the count.

This question builds upon the previous question. We listed all the actors who shared the same last name but now we want to count how many actors share a particular second name. For instance, the second name Kardashian is shared by 3 actors. Asking for the count per shared second name is a hint to use aggregation. We have already done the heavy lifting in the previous question. Let's **GROUP BY** second name first and see what we get:

```
SELECT b.SecondName
FROM Actors a
INNER JOIN Actors b
ON a.SecondName = b.SecondName
WHERE a.Id != b.Id
GROUP BY b.SecondName
```

```
GROUP BY b.SecondName;
```

```
mysql> SELECT b.SecondName
-> FROM Actors a
-> INNER JOIN Actors b
-> ON a.SecondName = b.SecondName
-> WHERE a.Id != b.Id
-> GROUP BY b.SecondName;
```

```
+-----+
| SecondName |
+-----+
| Bachchan   |
| Kardashian |
+-----+
2 rows in set (0.00 sec)
```

Remember **GROUP BY** returns one row per group so we get two rows as there are only two shared second names. Now we need to count how many rows fall into each group. One way is to count the first names in each group as follows:

```
SELECT a.SecondName,
COUNT(DISTINCT a.FirstName)
FROM Actors a
INNER JOIN Actors b
ON a.SecondName = b.SecondName
WHERE a.Id != b.Id
group by a.SecondName;
```



```
mysql> SELECT b.SecondName
-> FROM Actors a
-> INNER JOIN Actors b
-> ON a.SecondName = b.SecondName
-> WHERE a.Id != b.Id
-> GROUP BY b.SecondName;

+-----+
| SecondName |
+-----+
| Bachchan   |
| Kardashian |
+-----+
2 rows in set (0.00 sec)
```

The astute reader would immediately realize that in our setup, **FirstName** isn't the primary key. The above query will fail when we have two actors with exactly the same first and second names (born on different dates), which is a possibility. We'll report one less than the actual number of actors who share the same second name in such a scenario. The fix is to count the IDs that fall into each group as follows:

```
SELECT a.SecondName AS Actors_With_Shared_SecondNames,
COUNT(DISTINCT a.Id) AS Count
FROM Actors a
INNER JOIN Actors b
ON a.SecondName = b.SecondName
WHERE a.Id != b.Id
group by a.SecondName;
```

```
mysql> SELECT a.SecondName AS Actors_With_Shared_SecondNames,
-> COUNT(DISTINCT a.Id) AS Count
-> FROM Actors a
-> INNER JOIN Actors b
-> ON a.SecondName = b.SecondName
-> WHERE a.Id != b.Id
-> group by a.SecondName;
```

```
+-----+-----+
| Actors_With_Shared_SecondNames | Count |
+-----+-----+
| Bachchan                       |      2 |
| Kardashian                     |      3 |
+-----+-----+
2 rows in set (0.00 sec)
```

Question # 4

Write a query to display all those actors who have acted in at least one movie.

We want to print the names of all those actors who have acted in at least one movie. Imagine our database also holds aspiring actors who have not yet landed a movie contract.

The data we are interested in resides in the two tables **Cast** and **Actors**. If we join the two tables, we'll get all the actors that have at least one movie to their name. We can then grab the names of the actors from the joined tables and print them. There's a caveat though; we may print an actor's name if the actor has multiple movies, therefore, we must use the **DISTINCT** clause. The query is shown below:

```
SELECT DISTINCT CONCAT(FirstName, " ", SecondName) AS Actors_Acted_In
_Atleast_1_Movies
FROM Actors
INNER JOIN Cast
ON Id = ActorId;
```

```
mysql> SELECT DISTINCT CONCAT(FirstName, " ", SecondName) AS Actors_Acted_In_Atleast_1_Movies
-> FROM Actors
-> INNER JOIN Cast
-> ON Id = ActorId;

+-----+
| Actors_Acted_In_Atleast_1_Movies |
+-----+
| Brad Pitt                        |
| Angelina Jolie                  |
| Johnny Depp                    |
| Natalie Portman                 |
| Tom Cruise                     |
| Kim Kardashian                 |
| Shahrukh Khan                  |
| Khloe Kardashian                |
| Kourtney Kardashian             |
| Abhishek Bachchan               |
| Frank Sinatra                   |
+-----+
11 rows in set (0.00 sec)
```

Question # 5

As a corollary to the previous question, can you find the different ways of listing those aspiring actors who haven't acted in any movie yet?

One of the easiest ways to answer this query is to take IDs of the actors from the previous query and minus those IDs from the entire set of **Actors**. The remaining IDs will be of actors who don't have a film to their name yet. The query is shown below:

```
SELECT Id, CONCAT(FirstName, " ", SecondName) AS Actors_With_No_Movie
S
FROM Actors
WHERE Id NOT IN (SELECT Id
                  FROM Actors
                  INNER JOIN Cast
                  ON Id = ActorId);
```

```
mysql> SELECT Id, CONCAT(FirstName, " ", SecondName) AS Actors_With_No_Movies
-> FROM Actors
-> WHERE Id NOT IN (SELECT Id
-> FROM Actors
-> INNER JOIN Cast
-> ON Id = ActorId);
```

Id	Actors_With_No_Movies
2	Jennifer Aniston
7	Kylie Jenner
9	Amitabh Bachchan
11	priyanka Chopra
16	Fahim Haq

5 rows in set (0.00 sec)

The subquery returns IDs of those actors who have participated in films. We simply list those actors whose ID doesn't appear in the result set of the inner subquery.

Yet another way is to use a **LEFT JOIN** the two tables, with **Actor** as the left argument and the **Cast** table as the right argument. Let's see the result of a left join between the two tables.

```
SELECT *
FROM Actors
LEFT JOIN Cast
ON Id = ActorId;
```

```
mysql> SELECT *
-> FROM Actors
-> LEFT JOIN Cast
-> ON Id = ActorId;
```

Id	FirstName	SecondName	DoB	Gender	MaritalStatus	NetWorthInMillions	ActorId	MovieId
1	Brad	Pitt	1963-12-18	Male	Single	240	1	1
1	Brad	Pitt	1963-12-18	Male	Single	240	1	2
1	Brad	Pitt	1963-12-18	Male	Single	240	1	5
2	Jennifer	Aniston	1969-11-02	Female	Single	240	NULL	NULL
3	Angelina	Jolie	1975-06-04	Female	Single	100	3	2
4	Johnny	Depp	1963-06-09	Male	Single	200	4	4
5	Natalie	Portman	1981-06-09	Male	Married	60	5	10
6	Tom	Cruise	1962-07-03	Male	Divorced	570	6	3
7	Kylie	Jenner	1997-08-10	Female	Married	1000	NULL	NULL
8	Kim	Kardashian	1980-10-21	Female	Married	370	8	9
9	Amitabh	Bachchan	1942-10-11	Male	Married	400	NULL	NULL
10	Shahrukh	Khan	1965-11-02	Male	Married	600	10	8
11	priyanka	Chopra	1982-07-18	Female	Married	28	NULL	NULL
12	Khloe	Kardashian	1984-06-27	Female	Divorced	40	12	9
13	Kourtney	Kardashian	1979-04-18	Female	Single	35	13	9
14	Abhishek	Bachchan	1976-02-05	Male	Married	35	14	8
15	Frank	Sinatra	1915-12-12	Male	Married	600	15	6
16	Fahim	Haq	1981-01-01	Male	Married	1	NULL	NULL

18 rows in set (0.00 sec)

You can observe from the joined table that the MovieID column is NULL for those actors who haven't been part of any movie. Thus, we can use the condition **MovieID = NULL** in our join query to identify aspiring actors. The query is shown below:

```
SELECT CONCAT(FirstName, " ", SecondName) AS Actors_With_No_Movies
FROM Actors
LEFT JOIN Cast
ON Id = ActorId
WHERE MovieId IS NULL;
```

```
mysql> SELECT CONCAT(FirstName, " ", SecondName) AS Actors_With_No_Movies
-> FROM Actors
-> LEFT JOIN Cast
-> ON Id = ActorId
-> WHERE MovieId IS NULL;
```

```
+-----+
| Actors_With_No_Movies |
+-----+
| Jennifer Aniston      |
| Kylie Jenner          |
| Amitabh Bachchan      |
| priyanka Chopra       |
| Fahim Haq             |
+-----+
5 rows in set (0.00 sec)
```