Week 5, Homework 3 – SQL III Solutions

Mariah N Cornelio, 1002053287

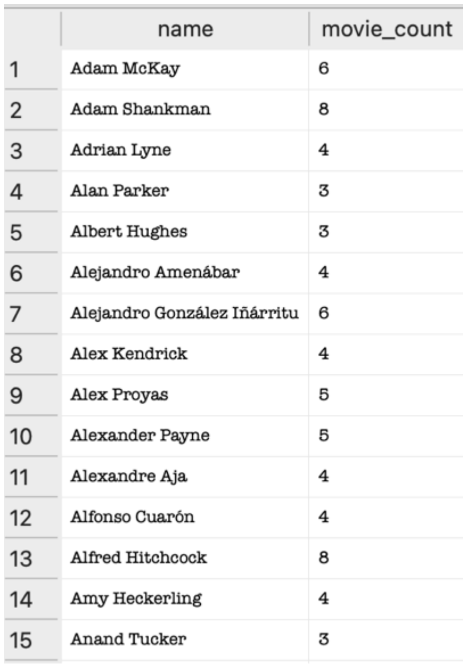
#1. Directors with more movies than the average

- Code

```
SELECT d.name, COUNT(m.id) AS movie_count --select the director name and count how many movies there are
FROM directors as d -- from directors database as d
JOIN movies as m -- from movies database as m
ON d.id=m.director_id -- join both databases by the id
GROUP BY d.id, d.name -- group by director name to see how many movies each one has
HAVING COUNT(m.id) > ( -- where the count of the movies
  SELECT AVG(movie_count) --subquery to create average movie count
  FROM (SELECT COUNT(m2.id) AS movie_count -- selecting the ones that has movie count
    FROM directors as d2
    JOIN movies as m2
    ON d2.id=m2.director_id
    GROUP BY d2.id
  ) as moremoviesthantheaverage) -- that is more than the average
ORDER BY d.name; -- and then order alphabetically
```

  ○

- Output

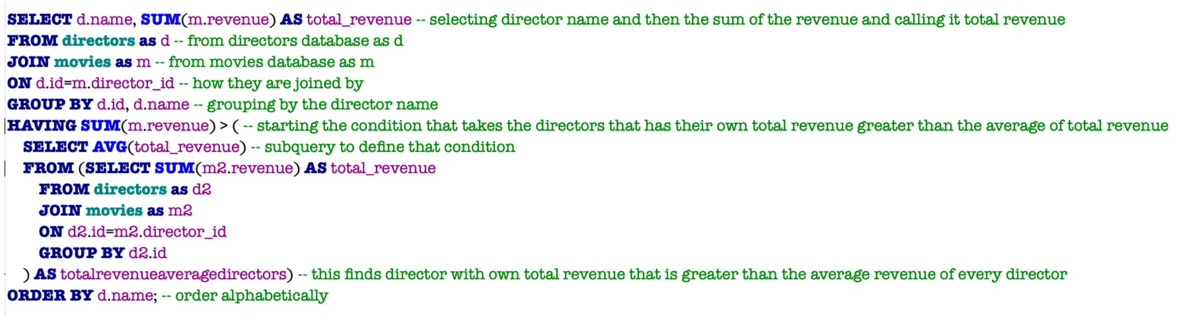


	name	movie_count
1	Adam McKay	6
2	Adam Shankman	8
3	Adrian Lyne	4
4	Alan Parker	3
5	Albert Hughes	3
6	Alejandro Amenábar	4
7	Alejandro González Iñárritu	6
8	Alex Kendrick	4
9	Alex Proyas	5
10	Alexander Payne	5
11	Alexandre Aja	4
12	Alfonso Cuarón	4
13	Alfred Hitchcock	8
14	Amy Heckerling	4
15	Anand Tucker	3

  ○

#2. Find directors with total revenue above the average revenue of all directors

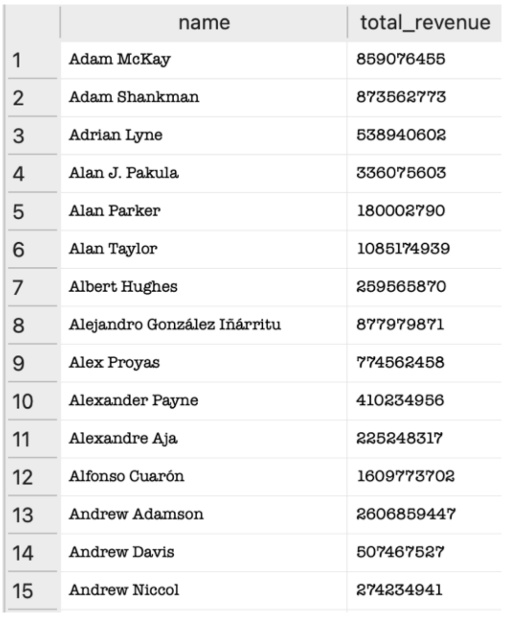
- Code



```sql
SELECT d.name, SUM(m.revenue) AS total_revenue -- selecting director name and then the sum of the revenue and calling it total revenue
FROM directors as d -- from directors database as d
JOIN movies as m -- from movies database as m
ON d.id=m.director_id -- how they are joined by
GROUP BY d.id, d.name -- grouping by the director name
HAVING SUM(m.revenue) > ( -- starting the condition that takes the directors that has their own total revenue greater than the average of total revenue
  SELECT AVG(total_revenue) -- subquery to define that condition
  FROM (SELECT SUM(m2.revenue) AS total_revenue
    FROM directors as d2
    JOIN movies as m2
    ON d2.id=m2.director_id
    GROUP BY d2.id
  ) AS totalrevenueaveragedirectors) -- this finds director with own total revenue that is greater than the average revenue of every director
ORDER BY d.name; -- order alphabetically
```

- This code is similar to #1's code!

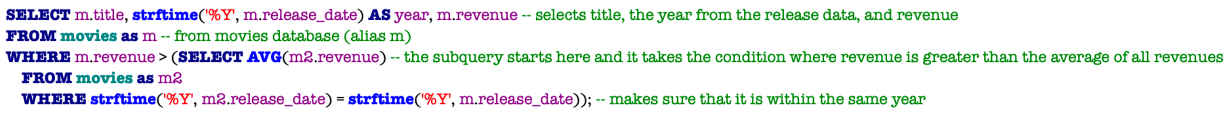
- Output



	name	total_revenue
1	Adam McKay	859076455
2	Adam Shankman	873562773
3	Adrian Lyne	538940602
4	Alan J. Pakula	336075603
5	Alan Parker	180002790
6	Alan Taylor	1085174939
7	Albert Hughes	259565870
8	Alejandro González Iñárritu	877979871
9	Alex Proyas	774562458
10	Alexander Payne	410234956
11	Alexandre Aja	225248317
12	Alfonso Cuarón	1609773702
13	Andrew Adamson	2606859447
14	Andrew Davis	507467527
15	Andrew Niccol	274234941

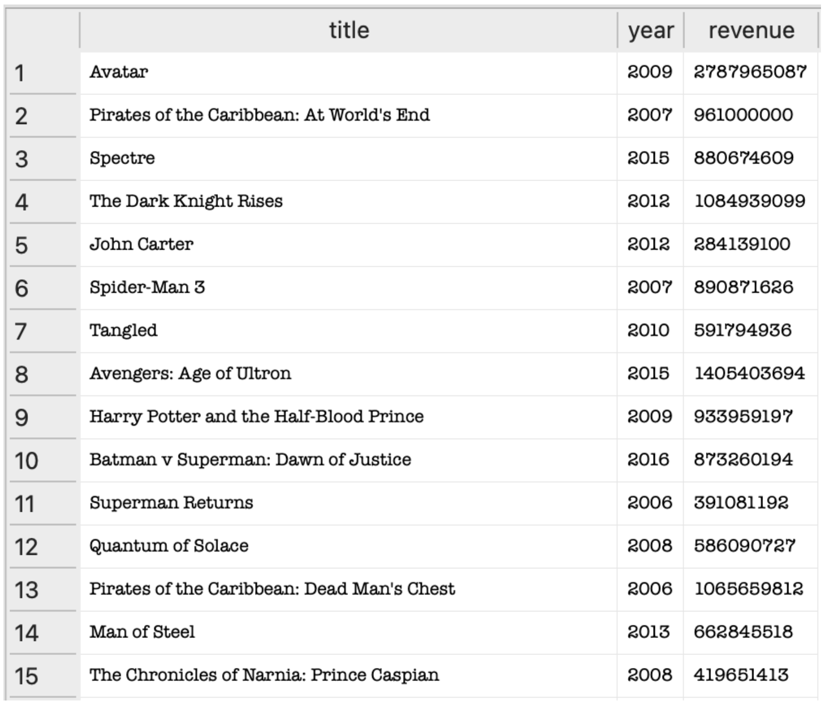
#3. Movies with revenue higher than the average revenue of all movies released in the same year

- Code



```sql
SELECT m.title, strftime('%Y', m.release_date) AS year, m.revenue -- selects title, the year from the release data, and revenue
FROM movies as m -- from movies database (alias m)
WHERE m.revenue > (SELECT AVG(m2.revenue) -- the subquery starts here and it takes the condition where revenue is greater than the average of all revenues
  FROM movies as m2
  WHERE strftime('%Y', m2.release_date) = strftime('%Y', m.release_date)); -- makes sure that it is within the same year
```

- Output



	title	year	revenue
1	Avatar	2009	2787965087
2	Pirates of the Caribbean: At World's End	2007	961000000
3	Spectre	2015	880674609
4	The Dark Knight Rises	2012	1084939099
5	John Carter	2012	284139100
6	Spider-Man 3	2007	890871626
7	Tangled	2010	591794936
8	Avengers: Age of Ultron	2015	1405403694
9	Harry Potter and the Half-Blood Prince	2009	933959197
10	Batman v Superman: Dawn of Justice	2016	873260194
11	Superman Returns	2006	391081192
12	Quantum of Solace	2008	586090727
13	Pirates of the Caribbean: Dead Man's Chest	2006	1065659812
14	Man of Steel	2013	662845518
15	The Chronicles of Narnia: Prince Caspian	2008	419651413

#4. Display the first 10 characters of each movie's title

- Code

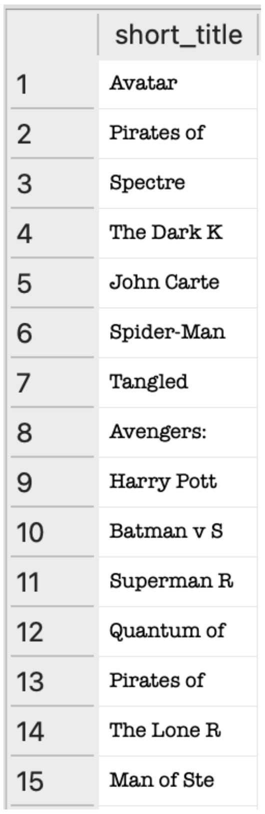
```
SELECT LEFT(title, 10) AS short_title
FROM movies;
```

  - LEFT() and RIGHT() are not supported directly in SQLite so I'll just use the SUBSTR() function

```
SELECT substr(title, 1, 10) AS short_title
FROM movies;
```

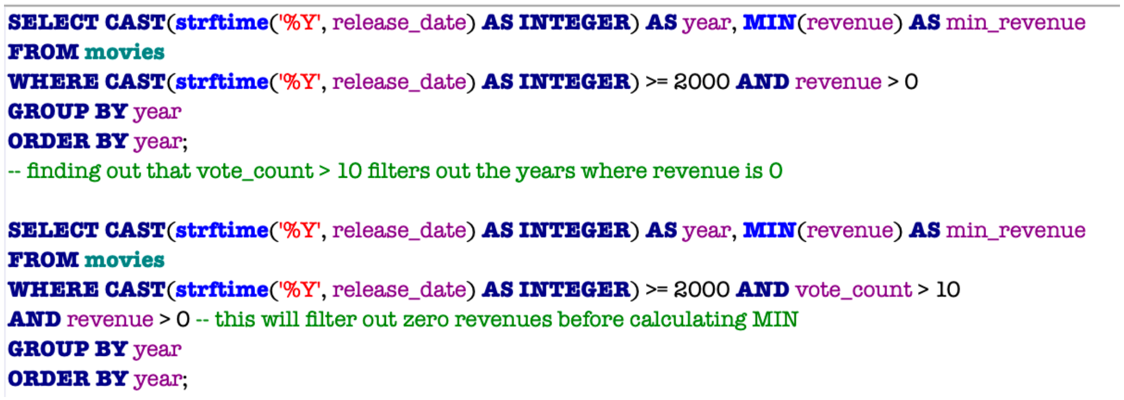
  - SUBSTR() grabs part of the string starting at index 1 and taking the length 10

- Output

- 

	short_title
1	Avatar
2	Pirates of
3	Spectre
4	The Dark K
5	John Carte
6	Spider-Man
7	Tangled
8	Avengers:
9	Harry Pott
10	Batman v S
11	Superman R
12	Quantum of
13	Pirates of
14	The Lone R
15	Man of Ste

#5. For movies released after the year 2000, display the year and the minimum revenue generated in that year for movies with more than 10 votes. Exclude years where the minimum revenue was 0

- Code
  - 

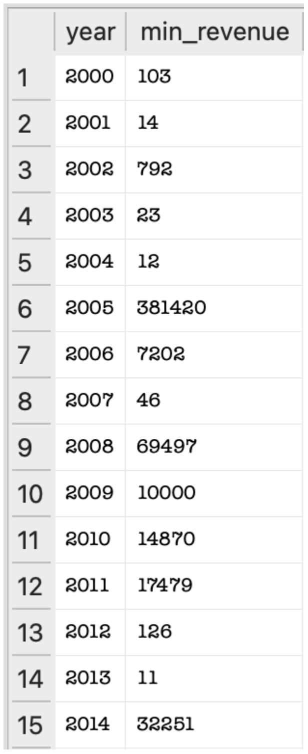
```
SELECT CAST(strftime('%Y', release_date) AS INTEGER) AS year, MIN(revenue) AS min_revenue
FROM movies
WHERE CAST(strftime('%Y', release_date) AS INTEGER) >= 2000 AND revenue > 0
GROUP BY year
ORDER BY year;
-- finding out that vote_count > 10 filters out the years where revenue is 0

SELECT CAST(strftime('%Y', release_date) AS INTEGER) AS year, MIN(revenue) AS min_revenue
FROM movies
WHERE CAST(strftime('%Y', release_date) AS INTEGER) >= 2000 AND vote_count > 10
AND revenue > 0 -- this will filter out zero revenues before calculating MIN
GROUP BY year
ORDER BY year;
```

    - Had to do some debugging first because vote_count > 10 was filtering out the rows where the budget was 0; the second block of code is correct
    - More explanation below

```sql
SELECT CAST(strftime('%Y', release_date) AS INTEGER) AS year,  -- extract the year from release_date (e.g. '2003-07-15' → '2003'),
-- then CAST it to an integer so we can do numeric comparisons and rename the column as "year"
MIN(revenue) AS min_revenue -- select and find the smallest revenue per year and call it "min_revenue"
FROM movies
WHERE CAST(strftime('%Y', release_date) AS INTEGER) >= 2000 -- only consider movies released in 2000 or later (cast as integer like we did before)
AND vote_count > 10 -- takees movies with more than 10 votes
 AND revenue > 0 -- ignore movies with 0 revenue
GROUP BY year -- group results by year
ORDER BY year; -- order the results by year
```

- Output



	year	min_revenue
1	2000	103
2	2001	14
3	2002	792
4	2003	23
5	2004	12
6	2005	381420
7	2006	7202
8	2007	46
9	2008	69497
10	2009	10000
11	2010	14870
12	2011	17479
13	2012	126
14	2013	11
15	2014	32251

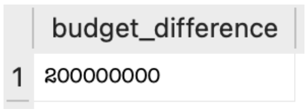
#6. Show the difference between the highest and lowest budget for movies released in ‘1997’

- Code

```sql
SELECT MAX(budget) - MIN(budget) AS budget_difference
FROM movies
WHERE strftime('%Y', release_date) = '1997';
```

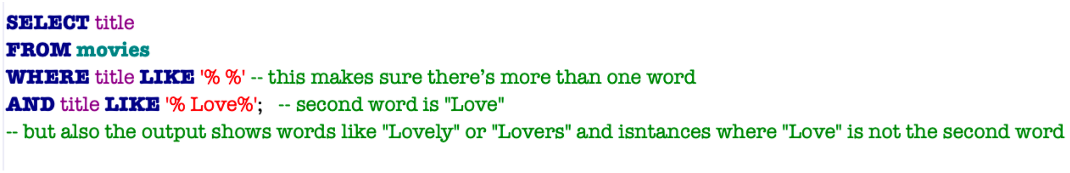
  - MAX of budget – MIN of budget is the budget difference and it takes the budget difference of the year 1997

- Output

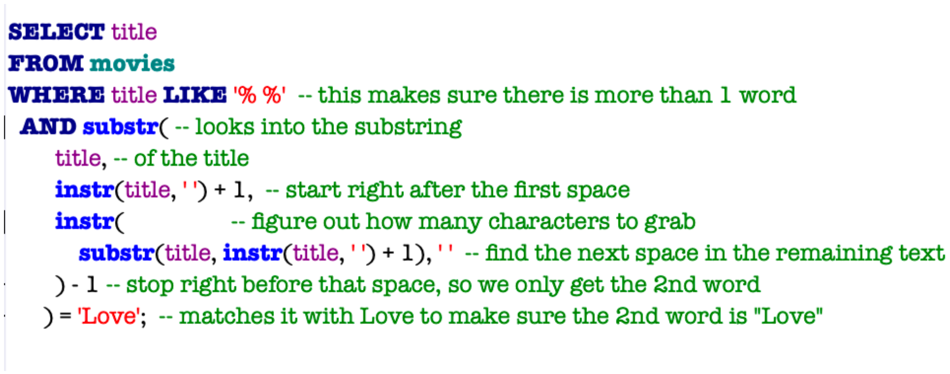


	budget_difference
1	200000000

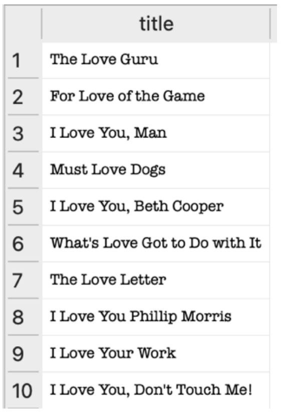
#7. List all movies where the second word in the title is ‘Love,’ considering titles with more than one word

- Code
  - 

```
SELECT title
FROM movies
WHERE title LIKE '% %' -- this makes sure there’s more than one word
AND title LIKE '% Love%';  -- second word is "Love"
-- but also the output shows words like "Lovely" or "Lovers" and isntances where "Love" is not the second word
```

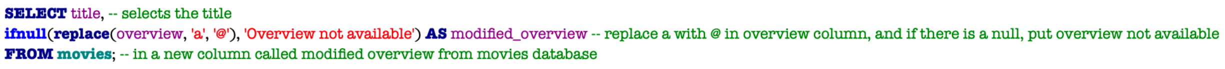
    - This code block is not so precise explained above
  - 

```
SELECT title
FROM movies
WHERE title LIKE '% %'  -- this makes sure there is more than 1 word
 AND substr( -- looks into the substring
    title, -- of the title
    instr(title, ' ') + 1,  -- start right after the first space
    instr(              -- figure out how many characters to grab
      substr(title, instr(title, ' ') + 1), ' '  -- find the next space in the remaining text
    ) - 1 -- stop right before that space, so we only get the 2nd word
   ) = 'Love';  -- matches it with Love to make sure the 2nd word is "Love"
```

- Output
  - 

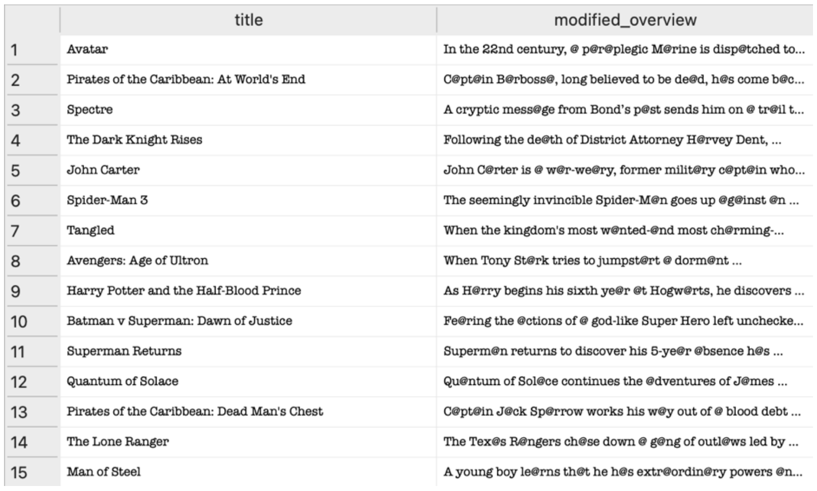
	title
1	The Love Guru
2	For Love of the Game
3	I Love You, Man
4	Must Love Dogs
5	I Love You, Beth Cooper
6	What's Love Got to Do with It
7	The Love Letter
8	I Love You Phillip Morris
9	I Love Your Work
10	I Love You, Don't Touch Me!

#8. Show the movie titles along with a modified overview where ‘a’ is replaced by ‘@’ and if the overview is ‘NULL’, display ‘Overview not available’

- Code
  - 

```
SELECT title, -- selects the title
ifnull(replace(overview, 'a', '@'), 'Overview not available') AS modified_overview -- replace a with @ in overview column, and if there is a null, put overview not available
FROM movies; -- in a new column called modified overview from movies database
```

- Output

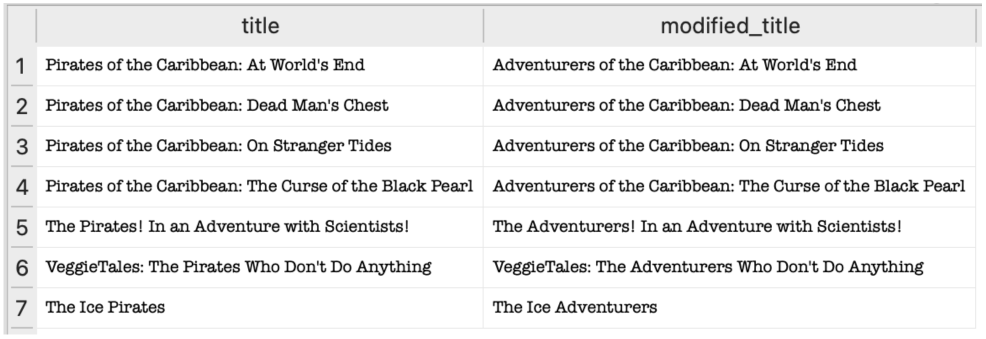


	title	modified_overview
1	Avatar	In the 22nd century, @ p@r@plegic M@rine is disp@tched to...
2	Pirates of the Caribbean: At World's End	C@pt@in B@rboss@, long believed to be de@d, h@s come b@c...
3	Spectre	A cryptic mess@ge from Bond's p@st sends him on @ tr@il t...
4	The Dark Knight Rises	Following the de@th of District Attorney H@rvey Dent, ...
5	John Carter	John C@rter is @ w@r-we@ry, former milit@ry c@pt@in who...
6	Spider-Man 3	The seemingly invincible Spider-M@n goes up @g@inst @n ...
7	Tangled	When the kingdom's most w@nted-@nd most ch@rming-...
8	Avengers: Age of Ultron	When Tony St@rk tries to jumpst@rt @ dorm@nt ...
9	Harry Potter and the Half-Blood Prince	As H@rry begins his sixth ye@r @t Hogw@rts, he discovers ...
10	Batman v Superman: Dawn of Justice	Fe@ring the @ctions of @ god-like Super Hero left unchecke...
11	Superman Returns	Superm@n returns to discover his 5-ye@r @bsence h@s ...
12	Quantum of Solace	Qu@ntum of Sol@ce continues the @dventures of J@mes ...
13	Pirates of the Caribbean: Dead Man's Chest	C@pt@in J@ck Sp@rrow works his w@y out of @ blood debt ...
14	The Lone Ranger	The Tex@s R@ngers ch@se down @ g@ng of outl@ws led by ...
15	Man of Steel	A young boy le@rns th@t he h@s extr@ordin@ry powers @n...

#9. Replace a word in movie titles

- Code

  -
    ```sql
    SELECT title, REPLACE(title, 'Pirates', 'Adventurers') AS modified_title
    FROM movies
    WHERE title LIKE '%Pirates%';
    ```
    - Replaces Pirates with Adventurers in the title column of movies database and outputs it into new column called modified title
    - And then shows instances where the title has Pirates in it to show a comparison of before and after
- Output



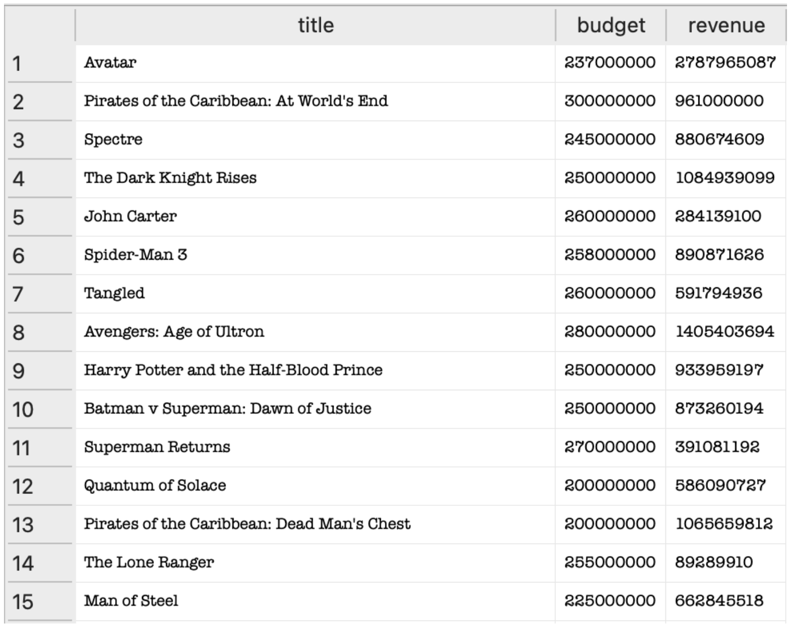
	title	modified_title
1	Pirates of the Caribbean: At World's End	Adventurers of the Caribbean: At World's End
2	Pirates of the Caribbean: Dead Man's Chest	Adventurers of the Caribbean: Dead Man's Chest
3	Pirates of the Caribbean: On Stranger Tides	Adventurers of the Caribbean: On Stranger Tides
4	Pirates of the Caribbean: The Curse of the Black Pearl	Adventurers of the Caribbean: The Curse of the Black Pearl
5	The Pirates! In an Adventure with Scientists!	The Adventurers! In an Adventure with Scientists!
6	VeggieTales: The Pirates Who Don't Do Anything	VeggieTales: The Adventurers Who Don't Do Anything
7	The Ice Pirates	The Ice Adventurers

#10. Replace null budgets and revenues with default values

- Code

  -
    ```sql
    SELECT title, ifnull(budget, 0) AS budget, ifnull(revenue, 0) AS revenue
    FROM movies;
    ```

        - This just replaces any null values in the budget and revenue column of the movies database with a 0, if there are any null values

- Output
    - 

	title	budget	revenue
1	Avatar	237000000	2787965087
2	Pirates of the Caribbean: At World's End	300000000	961000000
3	Spectre	245000000	880674609
4	The Dark Knight Rises	250000000	1084939099
5	John Carter	260000000	284139100
6	Spider-Man 3	258000000	890871626
7	Tangled	260000000	591794936
8	Avengers: Age of Ultron	280000000	1405403694
9	Harry Potter and the Half-Blood Prince	250000000	933959197
10	Batman v Superman: Dawn of Justice	250000000	873260194
11	Superman Returns	270000000	391081192
12	Quantum of Solace	200000000	586090727
13	Pirates of the Caribbean: Dead Man's Chest	200000000	1065659812
14	The Lone Ranger	255000000	89289910
15	Man of Steel	225000000	662845518