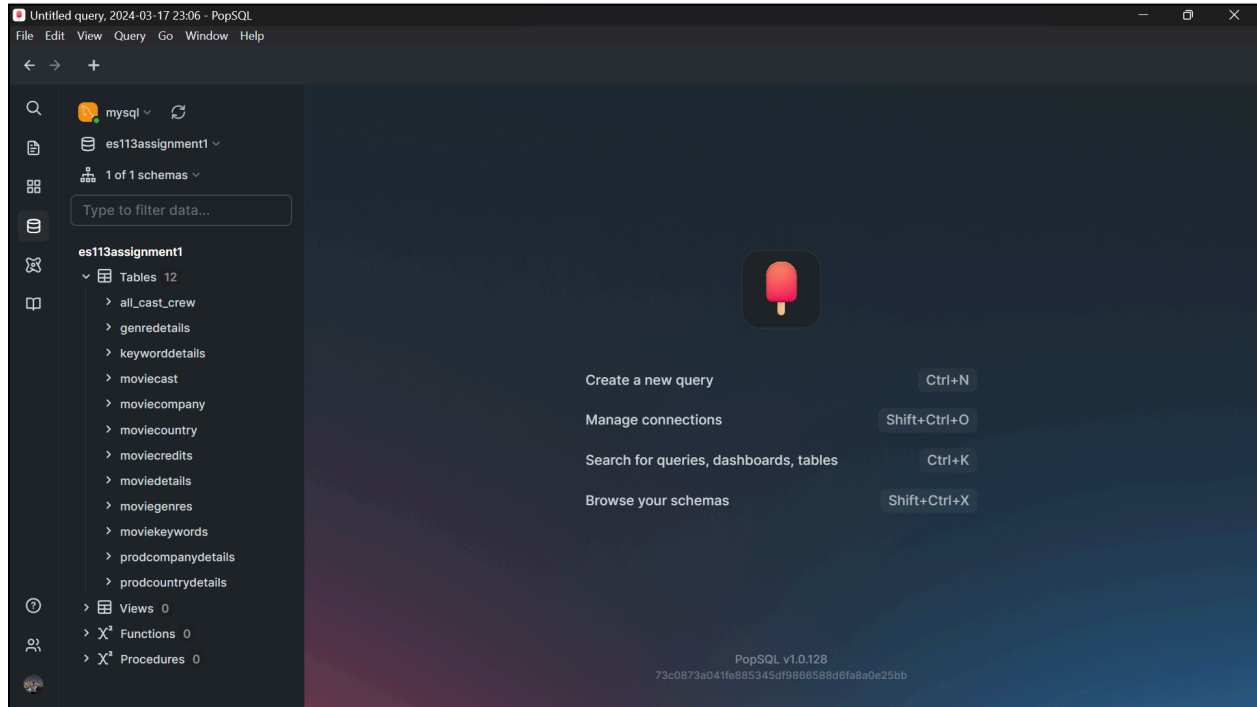




A Foreword



I am using POPSQL to query my database. As far as importing the movie database is concerned, I imported it using the workbench that gets installed while installing MYSQL. Above, It can be seen in the POPSQL window that the database has indeed been imported successfully, and the schema has been named es113assignment1.

Solutions

Question 1

The following screenshot shows the solution to the Question as well as the associated query used

The screenshot displays the PopSQL web application interface. On the left, a sidebar contains navigation icons. The main editor area shows a SQL query for 'Question1' with line numbers 1 through 7. The query is as follows:

```
--Question1
--How many number of movies are available in the moviedetails table?
1
2
3
4 SELECT * FROM moviedetails;
5
6 SELECT COUNT(movieid)
7 FROM moviedetails;
```

On the right side of the interface, there are two execution results for 'Question1'. The top result shows 'Success (1 row)' with a duration of 0.1 s, displaying a single row with the value 4803 for the COUNT(movieid) query. The bottom result shows 'Success (100 rows)' with a duration of 0 s, displaying a table of movie details.

	movieid	budget	original_language	title
1	5	4000000	en	Four Rooms
2	11	11000000	en	Star Wars
3	12	94000000	en	Finding Nemo
4	13	55000000	en	Forrest Gump
5	14	15000000	en	American Beauty
6	16	12800000	en	Dancer in the Dark
7	18	90000000	en	The Fifth Element
8	19	92620000	de	Metropolis
9	20	0	en	My Life Without Me
10	22	140000000	en	Pirates of the Caribbean: The Curse of

The Code:

```
SELECT * FROM moviedetails;

SELECT COUNT(movieid)
FROM moviedetails;
```

Question 2

The following screenshot shows the solution to the Question as well as the associated query used

The screenshot displays the PopSQL web application interface. The top navigation bar includes 'File', 'Edit', 'View', 'Query', 'Go', 'Window', and 'Help'. Below this, a tab bar shows multiple open files, with 'Question2' currently selected. The main editor area contains a SQL query with line numbers 1 through 10. The query is as follows:

```
1 -- --Question 2
2 -- --Which is the least popular movie?
3
4
5
6 SELECT title
7 FROM moviedetails
8 ORDER BY POPULARITY
9 LIMIT 1;
10
```

On the right side of the interface, there are buttons for 'Run (limit 100)' and 'Share'. Below these, a status bar indicates 'mysql' and 'es113assignment1'. The results panel on the right shows a 'Success' message with '1 row' and '0 s' execution time. It includes tabs for 'Explore', 'SQL', 'Data', 'Chart', and 'Export'. The 'Data' tab is active, displaying a table with one row:

	title
1	America Is Still the Place

The Code:

```
SELECT title
FROM moviedetails
ORDER BY POPULARITY
LIMIT 1;
```

Question 3

The following screenshot shows the solution to Question 3 as well as the associated query used

The screenshot shows a SQL IDE interface with a dark theme. The main editor displays a SQL query for 'Question 3'. The query is a subquery that finds the least popular movie by joining 'all_cast_crew', 'moviecredits', and 'moviedetails' tables, filtering by 'movie_id' from 'moviedetails', and ordering by 'popularity' in descending order, limited to 1 row. The query is executed successfully, as indicated by the 'Success (2 rows)' status and the 'Run history' panel on the right. The results table shows two rows: one for 'Pierre Coffin' (pid 124747) and one for 'Jeannine Berger' (pid 1447310).

```
1 -- Question 3
2 -- Which is the least popular movie?
3
4
5
6 select pid,name
7 from all_cast_crew
8 where pid IN
9 (SELECT crewid
10  FROM moviecredits
11  WHERE movie_id=
12  (SELECT movieid
13   FROM moviedetails
14   ORDER BY popularity DESC
15   LIMIT 1));
16
```

	pid	name
1	124747	Pierre Coffin
2	1447310	Jeannine Berger

The Code:

```
select pid,name
FROM all_cast_crew
WHERE pid IN
(SELECT crewid
FROM moviecredits
WHERE movie_id=
(SELECT movieid
FROM moviedetails
ORDER BY popularity DESC
LIMIT 1));
```

Question 4

The following screenshot shows the solution to Question 4 as well as the associated query used

The screenshot shows the PopSQL interface with the following SQL query in the editor:

```
1 --Question 3
2 --List the count of jobs (unique) in each movie department from the moviecredits table.
3 --E.g.: For the 'Directing' department, there are nine unique jobs (Director, Assistant Director, etc.)
4
5
6 SELECT department,COUNT(DISTINCT job)
7 FROM moviecredits
8 WHERE department IN
9 (
10  SELECT DISTINCT department AS A
11  FROM moviecredits
12 )
13 GROUP BY department;
14
15
```

The query is executed successfully, returning 11 rows of data. The results are displayed in a table with the following columns: department and COUNT(DISTINCT job).

	department	COUNT(DISTINCT job)
1	Art	28
2	Camera	19
3	Costume & Make-Up	22
4	Crew	92
5	Directing	9
6	Editing	13
7	Lighting	12
8	Production	27
9	Sound	39
10	Visual Effects	32

The screenshot shows the PopSQL interface with the following SQL query in the editor:

```
1 --Question 3
2 --List the count of jobs (unique) in each movie department from the moviecredits table.
3 --E.g.: For the 'Directing' department, there are nine unique jobs (Director, Assistant Director, etc.)
4
5
6 SELECT department,COUNT(DISTINCT job)
7 FROM moviecredits
8 WHERE department IN
9 (
10  SELECT DISTINCT department AS A
11  FROM moviecredits
12 )
13 GROUP BY department;
14
15
```

The query is executed successfully, returning 11 rows of data. The results are displayed in a table with the following columns: department and COUNT(DISTINCT job).

	department	COUNT(DISTINCT job)
11	Writing	18

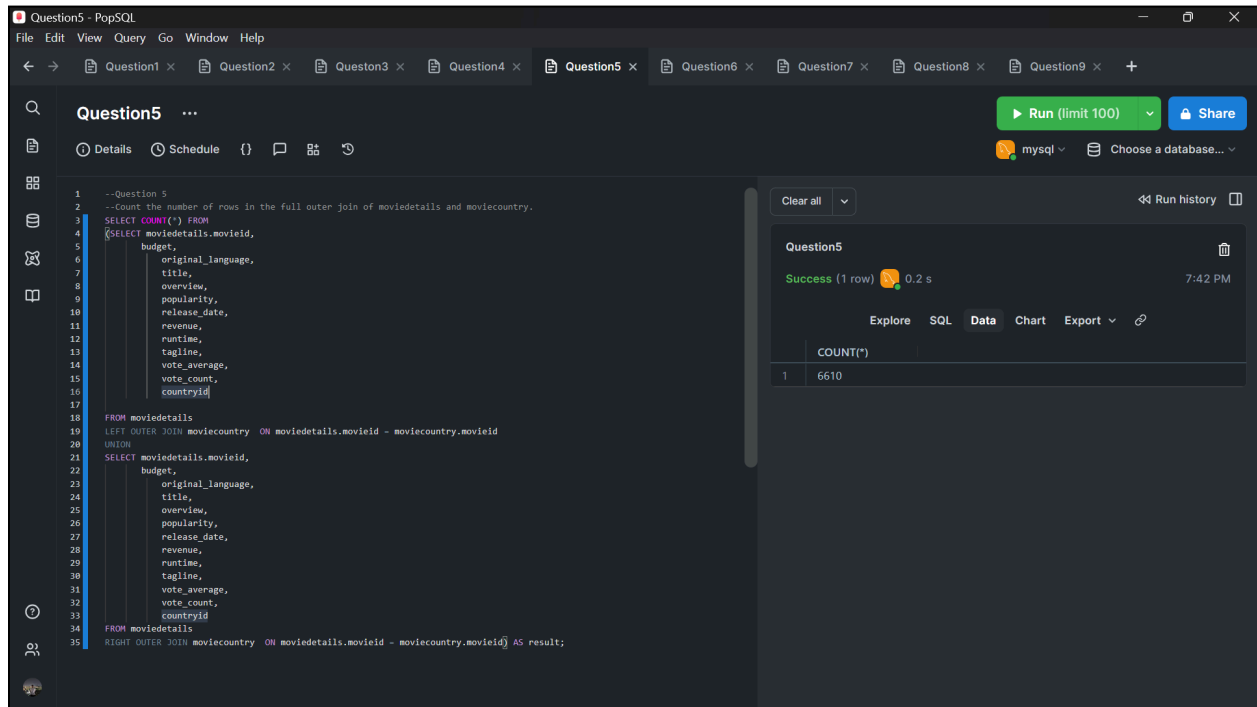
The Code:

```
SELECT department,COUNT(DISTINCT job)
FROM moviecredits
WHERE department IN
(
SELECT DISTINCT department AS A
```

```
FROM moviecredits
)
GROUP BY department;
```

Question 5

The following screenshot shows the solution to Question as well as the associated query used



The screenshot displays the PopSQL interface with the following components:

- Query Editor:** Contains the SQL query for Question 5, which counts the number of rows in a full outer join of `moviedetails` and `moviecountry`, unioned with the `moviedetails` table.
- Run Button:** A green button labeled "Run (limit 100)" with a "Share" button next to it.
- Results Panel:** Shows the execution status as "Success (1 row)" with a duration of "0.2 s". It includes tabs for "Explore", "SQL", "Data", "Chart", and "Export". The "Data" tab is active, showing a table with one row and one column, `COUNT(*)`, with the value 6610.

```
1 --Question 5
2 --Count the number of rows in the full outer join of moviedetails and moviecountry.
3 SELECT COUNT(*) FROM
4 (SELECT moviedetails.movieid,
5  budget,
6  original_language,
7  title,
8  overview,
9  popularity,
10 release_date,
11 revenue,
12 runtime,
13 tagline,
14 vote_average,
15 vote_count,
16 countryid
17 FROM moviedetails
18 LEFT OUTER JOIN moviecountry ON moviedetails.movieid = moviecountry.movieid
19 UNION
20 SELECT moviedetails.movieid,
21  budget,
22  original_language,
23  title,
24  overview,
25  popularity,
26  release_date,
27  revenue,
28  runtime,
29  tagline,
30  vote_average,
31  vote_count,
32  countryid
33 FROM moviedetails
34 RIGHT OUTER JOIN moviecountry ON moviedetails.movieid = moviecountry.movieid) AS result;
```

The Code:

```
SELECT COUNT(*) FROM
(SELECT moviedetails.movieid,
    budget,
    original_language,
    title,
    overview,
    popularity,
    release_date,
    revenue,
    runtime,
    tagline,
    vote_average,
    vote_count,
    countryid

FROM moviedetails
LEFT OUTER JOIN moviecountry ON moviedetails.movieid = moviecountry.movieid
UNION
SELECT moviedetails.movieid,
    budget,
    original_language,
    title,
    overview,
    popularity,
    release_date,
    revenue,
```

```
runtime,  
tagline,  
vote_average,  
vote_count,  
countryid  
FROM moviedetails  
RIGHT OUTER JOIN moviecountry ON moviedetails.movieid = moviecountry.movieid) AS result;
```


Question 6

The following screenshot shows the solution to Question as well as the associated query used

The screenshot displays the PopSQL web application interface. The top navigation bar includes 'File', 'Edit', 'View', 'Query', 'Go', 'Window', and 'Help'. Below this, a series of tabs represent different queries, with 'Question6' currently selected. The main editor area on the left contains the following SQL query:

```
--Question6
--How many movies are there whose popularity is more than 'Toy Story 2'?

SELECT COUNT(title)
FROM moviedetails
WHERE popularity>
(
  SELECT popularity
  FROM moviedetails
  WHERE title = 'Toy Story 2'
);
```

On the right side of the interface, there is a 'Run (limit 100)' button and a 'Share' button. Below these, a status bar indicates 'Success (1 row)' and '0.1 s'. A 'Run history' button is also present. The results are displayed in a table with the following structure:

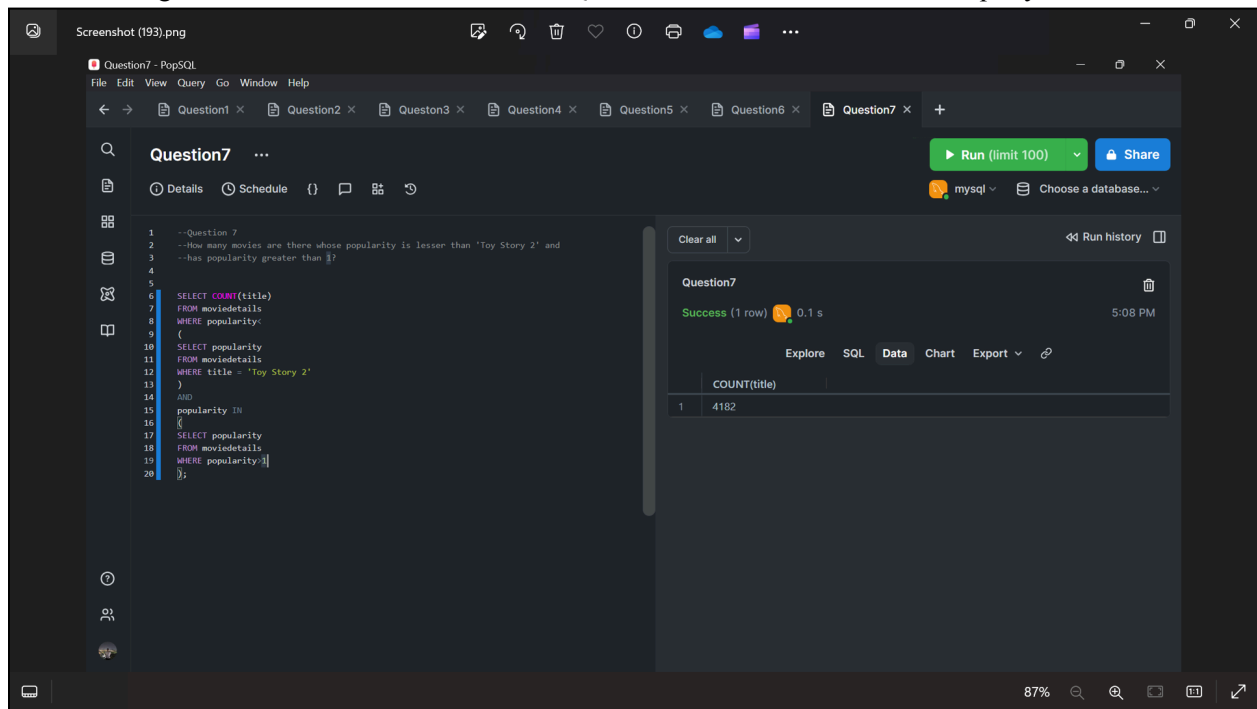
	COUNT(title)
1	201

The Code:

```
SELECT COUNT(title)
FROM moviedetails
WHERE popularity>
(
  SELECT popularity
  FROM moviedetails
  WHERE title = 'Toy Story 2'
);
```

Question 7

The following screenshot shows the solution to Question as well as the associated query used



The screenshot displays a SQL IDE interface with a dark theme. The main editor shows a SQL query for 'Question 7'. The query is a complex join between two subqueries. The first subquery selects the count of titles from the 'moviedetails' table where the popularity is less than the popularity of 'Toy Story 2'. The second subquery selects the popularity of 'Toy Story 2' from the 'moviedetails' table. The main query then counts the number of titles from the 'moviedetails' table where the popularity is greater than the result of the first subquery and the title is 'Toy Story 2'.

```
1 --Question 7
2 --How many movies are there whose popularity is lesser than 'Toy Story 2' and
3 --has popularity greater than ?
4
5
6 SELECT COUNT(title)
7 FROM moviedetails
8 WHERE popularity<
9 (
10 SELECT popularity
11 FROM moviedetails
12 WHERE title = 'Toy Story 2'
13 )
14 AND
15 popularity IN
16 (
17 SELECT popularity
18 FROM moviedetails
19 WHERE popularity>1
20 );
```

The results pane on the right shows a single row with the count of titles, which is 4182. The status bar at the bottom indicates 87% zoom.

The Code:

```
SELECT COUNT(title)
FROM moviedetails
WHERE popularity<
(
SELECT popularity
FROM moviedetails
WHERE title = 'Toy Story 2'
)
AND
popularity IN
(
SELECT popularity
FROM moviedetails
WHERE popularity>1
);
```

Question 8

The following screenshot shows the solution to Question as well as the associated query used

Question8 - PopSQL

File Edit View Query Go Window Help

← → Question1 × Question2 × Question3 × Question4 × Question5 × Question6 × Question7 × Question8 × Question9 × +

Question8 ...

Run (limit 100) Share

mysql Choose a database...

Details Schedule {} [] @

2 --Save the full outer join of moviedetails and moviecountry in table moviedetailscountry

3 --and find the movies whose countryid is 'US', along with their count.

4

5

6 CREATE TABLE moviedetailscountry AS

7 (SELECT moviedetails.movieid,

8 budget,

9 original_language,

10 title,

11 overview,

12 popularity,

13 release_date,

14 revenue,

15 runtime,

16 tagline,

17 vote_average,

18 vote_count,

19 countryid

20

21 FROM moviedetails

22 LEFT OUTER JOIN moviecountry ON moviedetails.movieid = moviecountry.movieid

23 UNION

24 SELECT moviedetails.movieid,

25 budget,

26 original_language,

27 title,

28 overview,

29 popularity,

30 release_date,

31 revenue,

32 runtime,

33 tagline,

34 vote_average,

35 vote_count,

36 countryid

37 FROM moviedetails

38 RIGHT OUTER JOIN moviecountry ON moviedetails.movieid = moviecountry.movieid);

39

40

Clear all

Run history

Question8

Success (1 row) 0.1 s 5:12 PM

Explore SQL Data Chart Export

	COUNT(title)
1	3956

Question8 - PopSQL

File Edit View Query Go Window Help

← → Question1 × Question2 × Question3 × Question4 × Question5 × Question6 × Question7 × Question8 × Question9 × +

Question8 ...

Run (limit 100) Share

mysql Choose a database...

Details Schedule {} ☰ ↺

21 FROM moviedetails

22 LEFT OUTER JOIN moviecountry ON moviedetails.movieid = moviecountry.movieid

23 UNION

24 SELECT moviedetails.movieid,

25 budget,

26 original_language,

27 title,

28 overview,

29 popularity,

30 release_date,

31 revenue,

32 runtime,

33 tagline,

34 vote_average,

35 vote_count,

36 countryid

37 FROM moviedetails

38 RIGHT OUTER JOIN moviecountry ON moviedetails.movieid = moviecountry.movieid;

39

40

41

42 CREATE TABLE A AS

43 (SELECT title

44 FROM moviedetailscountry

45 WHERE countryid = 'US')

46

47 SELECT COUNT(title)

48 FROM A

Clear all

Run history

Question8

Success (1 row) 0.1 s 5:12 PM

Explore SQL Data Chart Export

	COUNT(title)
1	3956

The Code:

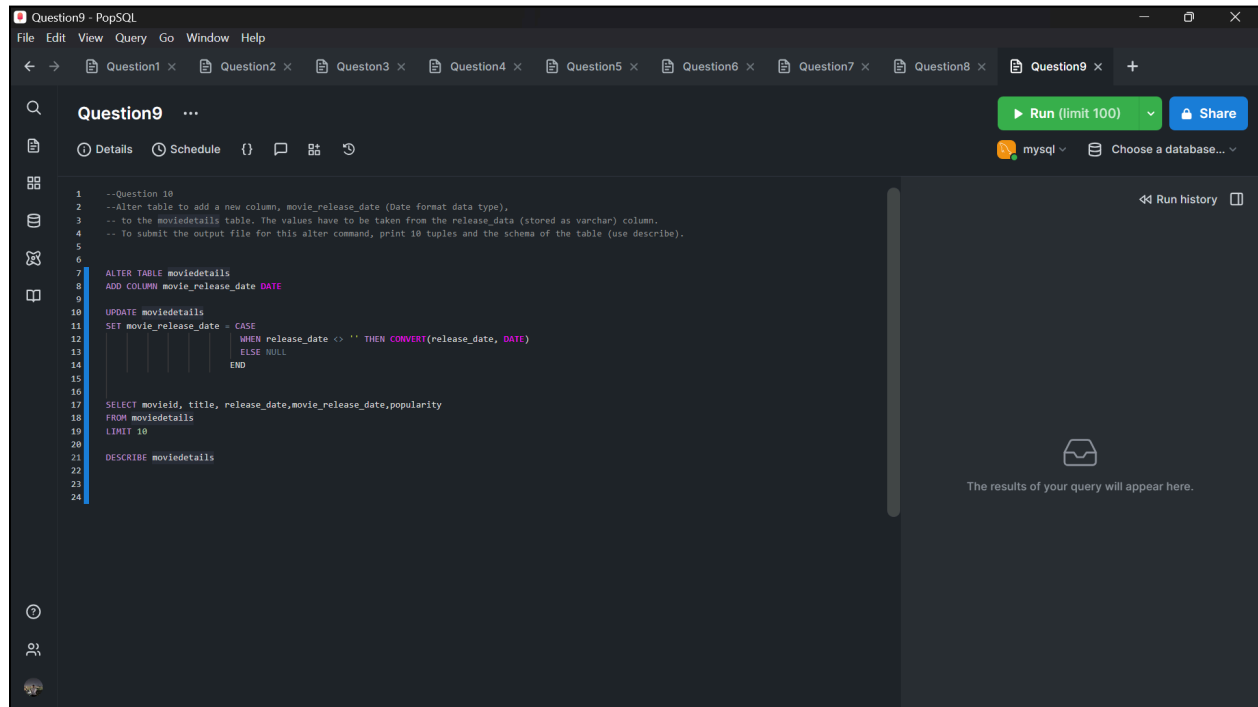
```
CREATE TABLE moviedetailscountry AS
(SELECT moviedetails.movieid,
    budget,
    original_language,
    title,
    overview,
    popularity,
    release_date,
    revenue,
    runtime,
    tagline,
    vote_average,
    vote_count,
    countryid
FROM moviedetails
LEFT OUTER JOIN moviecountry ON moviedetails.movieid = moviecountry.movieid
UNION
SELECT moviedetails.movieid,
    budget,
    original_language,
    title,
    overview,
    popularity,
    release_date,
    revenue,
    runtime,
    tagline,
    vote_average,
    vote_count,
    countryid
FROM moviedetails
RIGHT OUTER JOIN moviecountry ON moviedetails.movieid = moviecountry.movieid);

CREATE TABLE A AS
(SELECT title
FROM moviedetailscountry
WHERE countryid = 'US')

SELECT COUNT(title)
FROM A
```

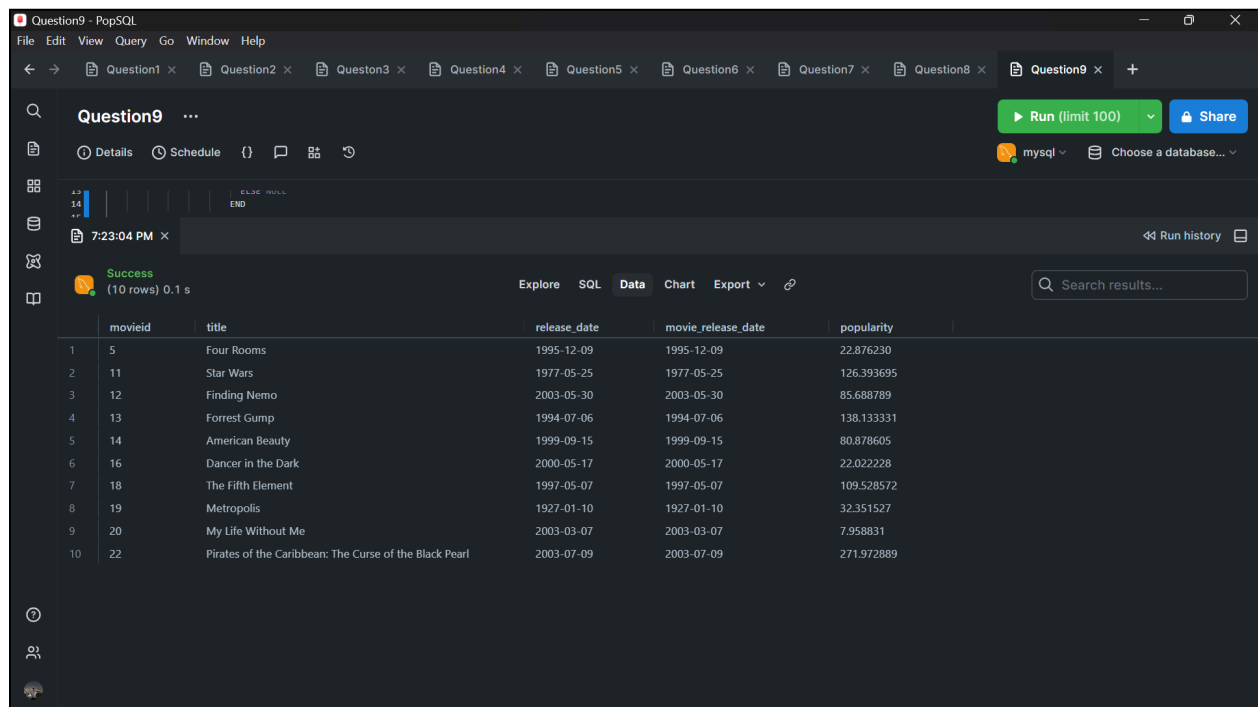
Question 9

The following screenshot shows the query used to solve the Question



```
1 --Question 10
2 --Alter table to add a new column, movie_release_date (Date format data type),
3 -- to the moviedetails table. The values have to be taken from the release_data (stored as varchar) column.
4 -- To submit the output file for this alter command, print 10 tuples and the schema of the table (use describe).
5
6
7 ALTER TABLE moviedetails
8 ADD COLUMN movie_release_date DATE
9
10 UPDATE moviedetails
11 SET movie_release_date = CASE
12     WHEN release_date <> '' THEN CONVERT(release_date, DATE)
13     ELSE NULL
14 END
15
16
17 SELECT movieid, title, release_date, movie_release_date, popularity
18 FROM moviedetails
19 LIMIT 10
20
21 DESCRIBE moviedetails
22
23
24
```

An example shown below of first 10 instances from the table shows clearly that the data from the column release_date did indeed get mapped to movie_release_data, along with data type conversion from VARCHAR to DATE as shown in the trailing screenshot



	movieid	title	release_date	movie_release_date	popularity
1	5	Four Rooms	1995-12-09	1995-12-09	22.876230
2	11	Star Wars	1977-05-25	1977-05-25	126.393695
3	12	Finding Nemo	2003-05-30	2003-05-30	85.688789
4	13	Forrest Gump	1994-07-06	1994-07-06	138.133331
5	14	American Beauty	1999-09-15	1999-09-15	80.878605
6	16	Dancer in the Dark	2000-05-17	2000-05-17	22.022228
7	18	The Fifth Element	1997-05-07	1997-05-07	109.528572
8	19	Metropolis	1927-01-10	1927-01-10	32.351527
9	20	My Life Without Me	2003-03-07	2003-03-07	7.958831
10	22	Pirates of the Caribbean: The Curse of the Black Pearl	2003-07-09	2003-07-09	271.972889

Question9 - PopSQL

File Edit View Query Go Window Help

Question1 x Question2 x Question3 x Question4 x Question5 x Question6 x Question7 x Question8 x Question9 x +

Question9 ...

Run (limit 100) Share

mysql Choose a database...

Details Schedule SQL Data Chart Export

Success (13 rows) 0 s

Search results...

	Field	Type	Null	Key	Default	Extra
1	movieid	int	NO	PR	null	
2	budget	int	YES		null	
3	original_language	varchar(2)	YES		null	
4	title	varchar(86)	YES		null	
5	overview	varchar(1000)	YES		null	
6	popularity	decimal(9,6)	YES		null	
7	release_date	varchar(10)	YES		null	
8	revenue	bigint	YES		null	
9	runtime	decimal(10,0)	YES		null	
10	tagline	varchar(252)	YES		null	
11	vote_average	decimal(3,1)	YES		null	
12	vote_count	int	YES		null	
13	movie_release_date	date	YES		null	

The Code:

```
ALTER TABLE moviedetails
ADD COLUMN movie_release_date DATE

UPDATE moviedetails
SET movie_release_date = CASE
    WHEN release_date <> '' THEN CONVERT(release_date, DATE)
    ELSE NULL
END

SELECT movieid, title, release_date, movie_release_date, popularity
FROM moviedetails
LIMIT 10

DESCRIBE moviedetails
```