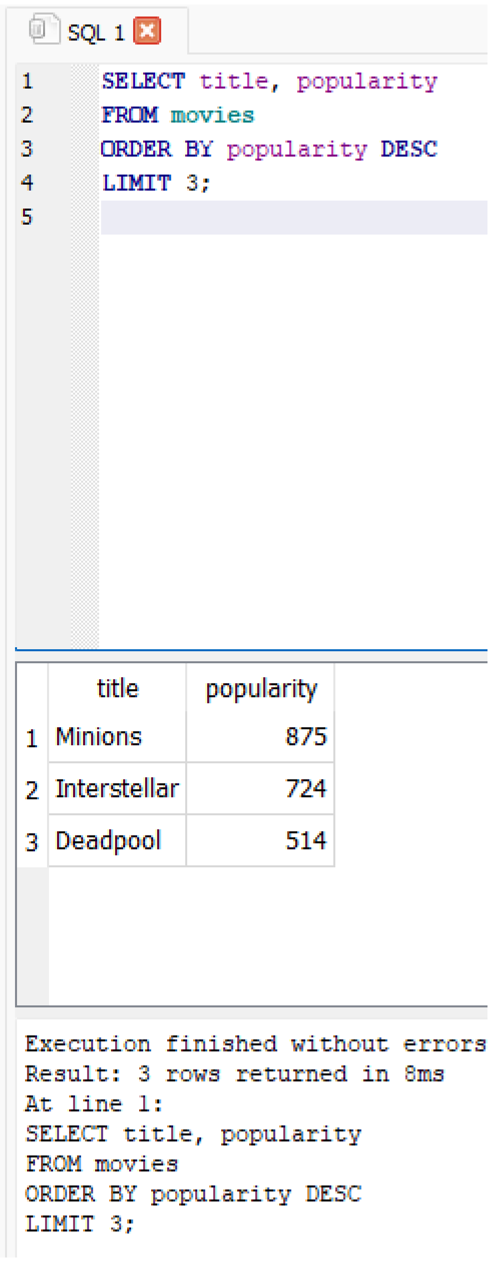
W13-Lab11: Advanced SQL

Members: Tyler, Christine, Angelina, Humza, Nathan, Efaz

1. Get the top 3 most popular movies



SQL 1

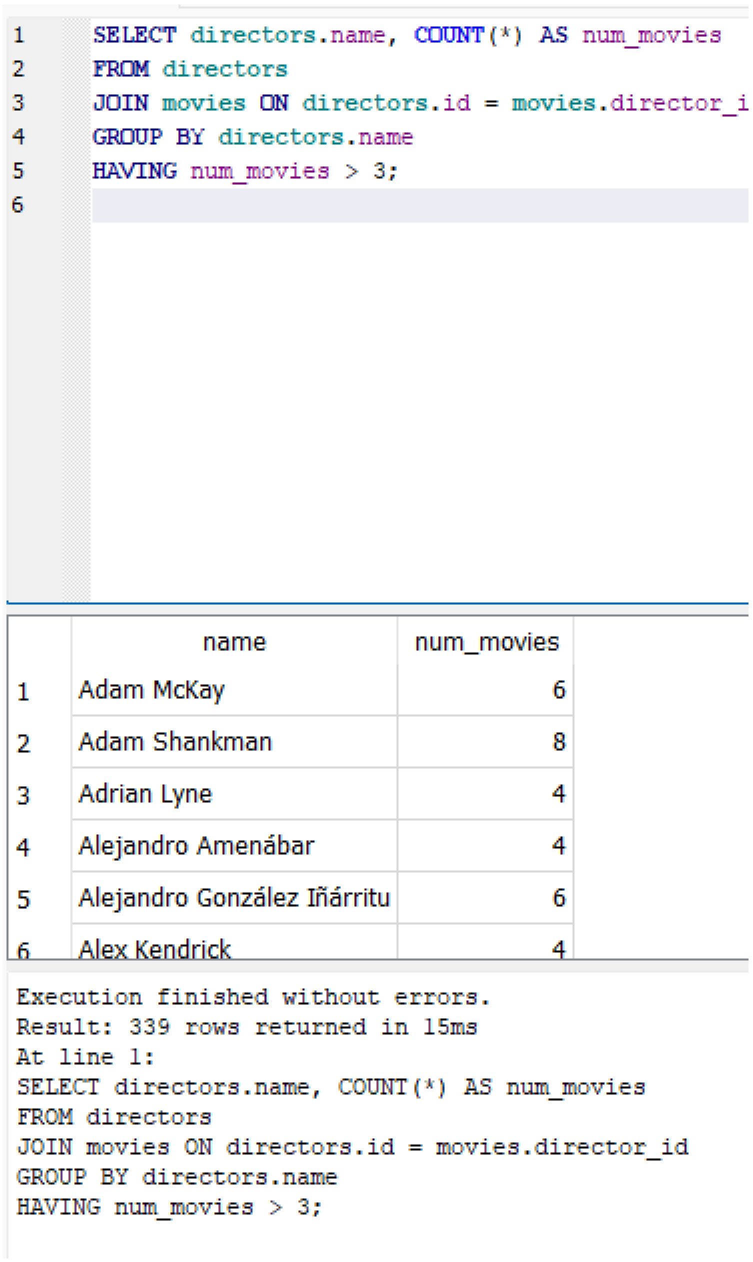
```sql
SELECT title, popularity
FROM movies
ORDER BY popularity DESC
LIMIT 3;
```

	title	popularity
1	Minions	875
2	Interstellar	724
3	Deadpool	514

```
Execution finished without errors
Result: 3 rows returned in 8ms
At line 1:
SELECT title, popularity
FROM movies
ORDER BY popularity DESC
LIMIT 3;
```

   a.

2. List the names of directors along with the count of movies they have directed, for directors who have directed more than 3 movies.



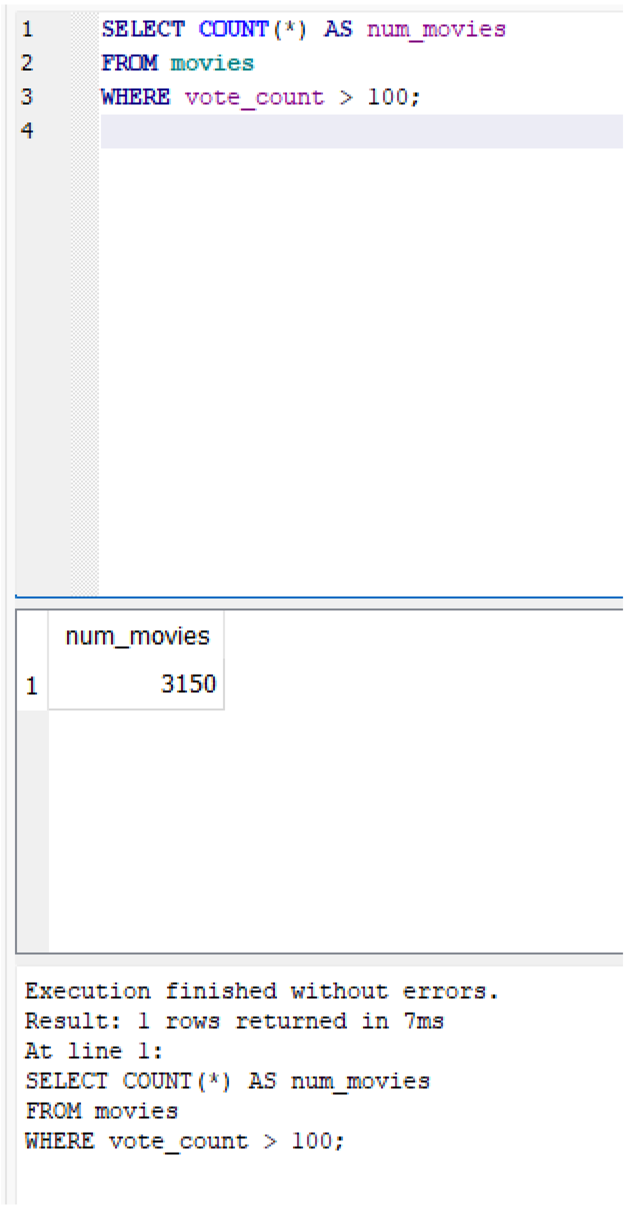
```
SELECT directors.name, COUNT(*) AS num_movies
FROM directors
JOIN movies ON directors.id = movies.director_i
GROUP BY directors.name
HAVING num_movies > 3;
```

	name	num_movies
1	Adam McKay	6
2	Adam Shankman	8
3	Adrian Lyne	4
4	Alejandro Amenábar	4
5	Alejandro González Iñárritu	6
6	Alex Kendrick	4

```
Execution finished without errors.
Result: 339 rows returned in 15ms
At line 1:
SELECT directors.name, COUNT(*) AS num_movies
FROM directors
JOIN movies ON directors.id = movies.director_id
GROUP BY directors.name
HAVING num_movies > 3;
```

a.

3. Count how many movies have a vote count greater than 100

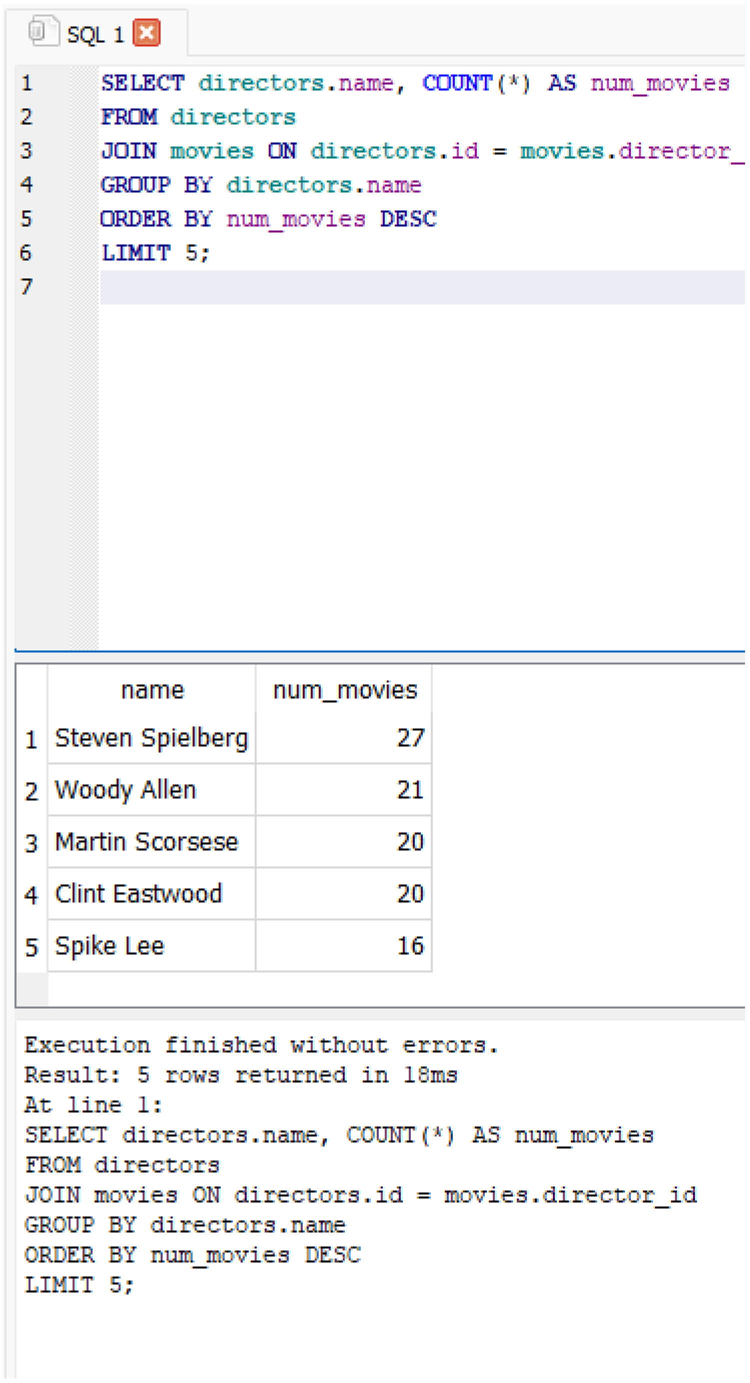

```sql
SELECT COUNT(*) AS num_movies
FROM movies
WHERE vote_count > 100;
```

	num_movies
1	3150

```
Execution finished without errors.
Result: 1 rows returned in 7ms
At line 1:
SELECT COUNT(*) AS num_movies
FROM movies
WHERE vote_count > 100;
```

a.

4. Retrieve a list of the top 5 directors who have the most movies in the database



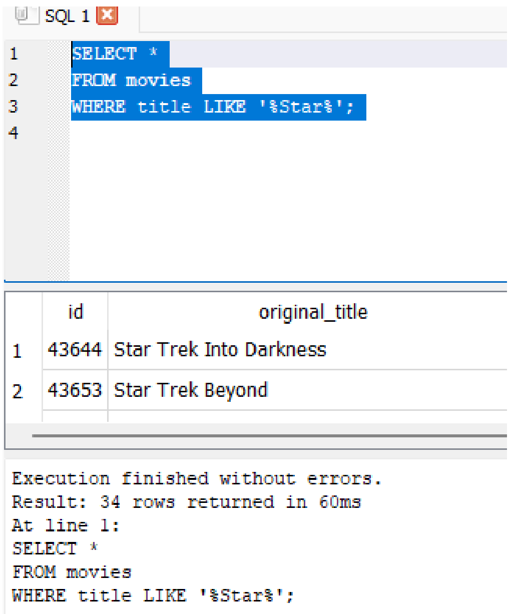
```
SELECT directors.name, COUNT(*) AS num_movies
FROM directors
JOIN movies ON directors.id = movies.director_
GROUP BY directors.name
ORDER BY num_movies DESC
LIMIT 5;
```

	name	num_movies
1	Steven Spielberg	27
2	Woody Allen	21
3	Martin Scorsese	20
4	Clint Eastwood	20
5	Spike Lee	16

```
Execution finished without errors.
Result: 5 rows returned in 18ms
At line 1:
SELECT directors.name, COUNT(*) AS num_movies
FROM directors
JOIN movies ON directors.id = movies.director_id
GROUP BY directors.name
ORDER BY num_movies DESC
LIMIT 5;
```

a.

5. List all movies that have ‘Star’ in the title


SQL 1

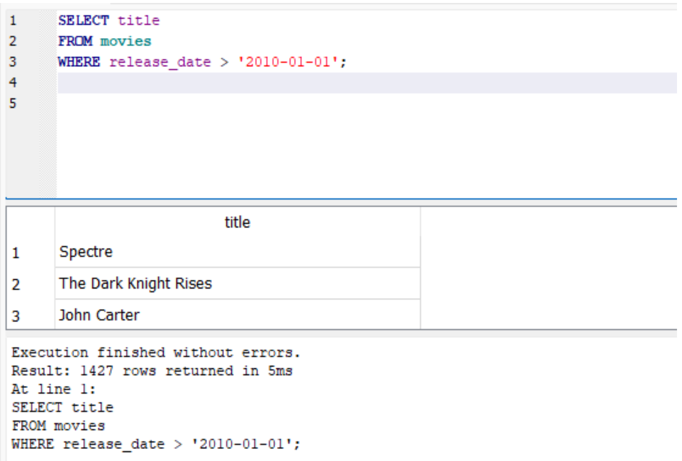
```
SELECT *
FROM movies
WHERE title LIKE '%Star%';
```

	id	original_title
1	43644	Star Trek Into Darkness
2	43653	Star Trek Beyond

```
Execution finished without errors.
Result: 34 rows returned in 60ms
At line 1:
SELECT *
FROM movies
WHERE title LIKE '%Star%';
```

a.

6. Retrieve the titles of movies released after January 1st, 2010



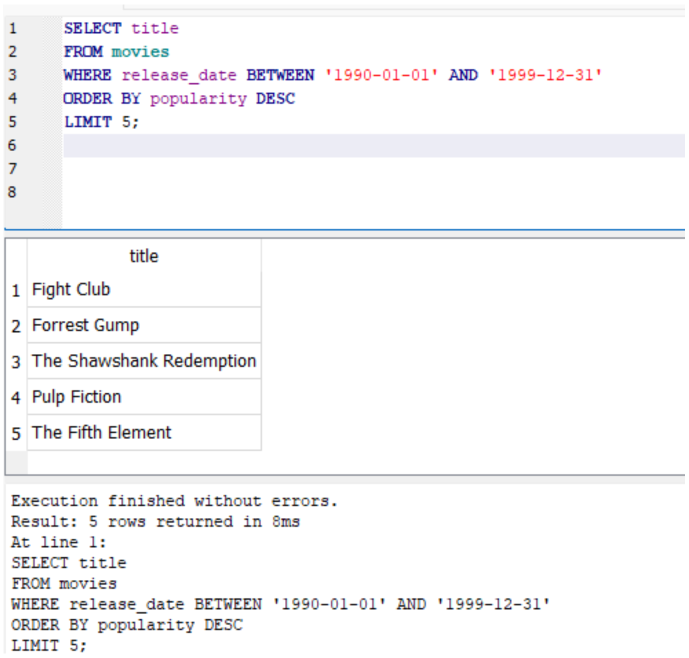
```
SELECT title
FROM movies
WHERE release_date > '2010-01-01';
```

	title
1	Spectre
2	The Dark Knight Rises
3	John Carter

```
Execution finished without errors.
Result: 1427 rows returned in 5ms
At line 1:
SELECT title
FROM movies
WHERE release_date > '2010-01-01';
```

a.

7. List the top 5 popular movies released in the 1990s, ordered by popularity



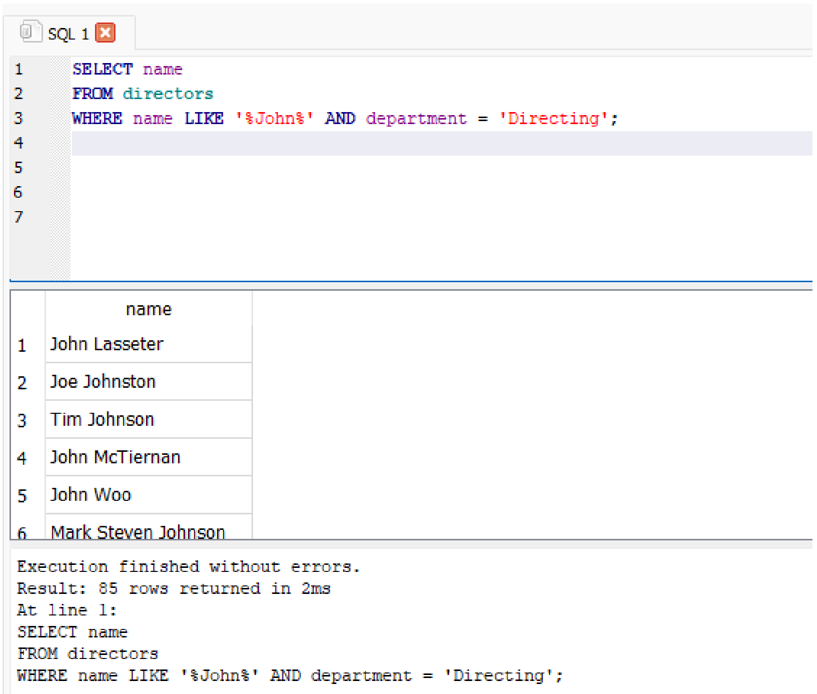
```
SELECT title
FROM movies
WHERE release_date BETWEEN '1990-01-01' AND '1999-12-31'
ORDER BY popularity DESC
LIMIT 5;
```

	title
1	Fight Club
2	Forrest Gump
3	The Shawshank Redemption
4	Pulp Fiction
5	The Fifth Element

```
Execution finished without errors.
Result: 5 rows returned in 8ms
At line 1:
SELECT title
FROM movies
WHERE release_date BETWEEN '1990-01-01' AND '1999-12-31'
ORDER BY popularity DESC
LIMIT 5;
```

a.

8. Display the names of directors with 'John' in their name who direct in the 'Directing' department



SQL 1

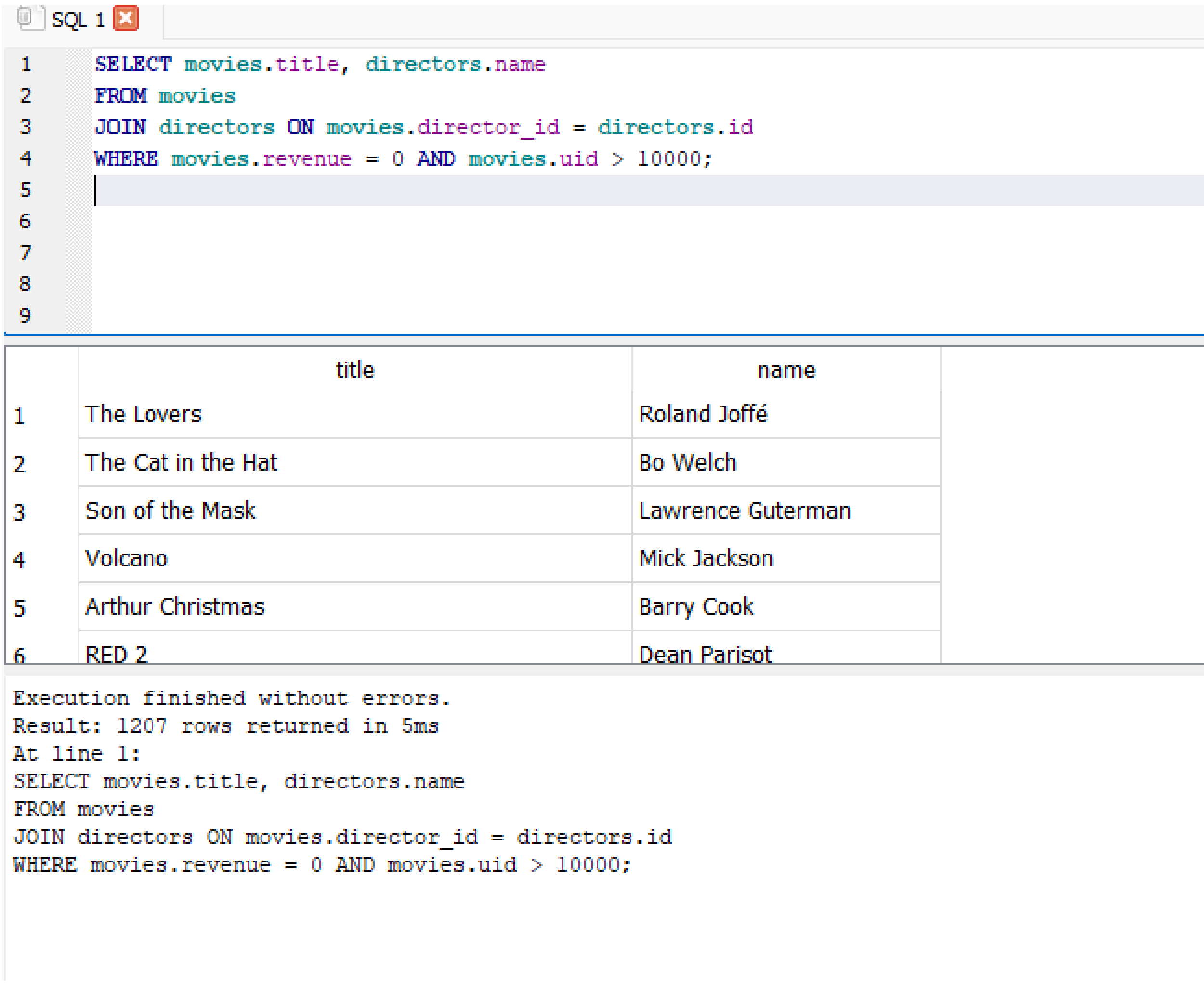
```
SELECT name
FROM directors
WHERE name LIKE '%John%' AND department = 'Directing';
```

	name
1	John Lasseter
2	Joe Johnston
3	Tim Johnson
4	John McTiernan
5	John Woo
6	Mark Steven Johnson

```
Execution finished without errors.
Result: 85 rows returned in 2ms
At line 1:
SELECT name
FROM directors
WHERE name LIKE '%John%' AND department = 'Directing';
```

a.

9. Retrieve the movie titles and directors' names for films with no revenue that still have a uid greater than 10000.


SQL 1

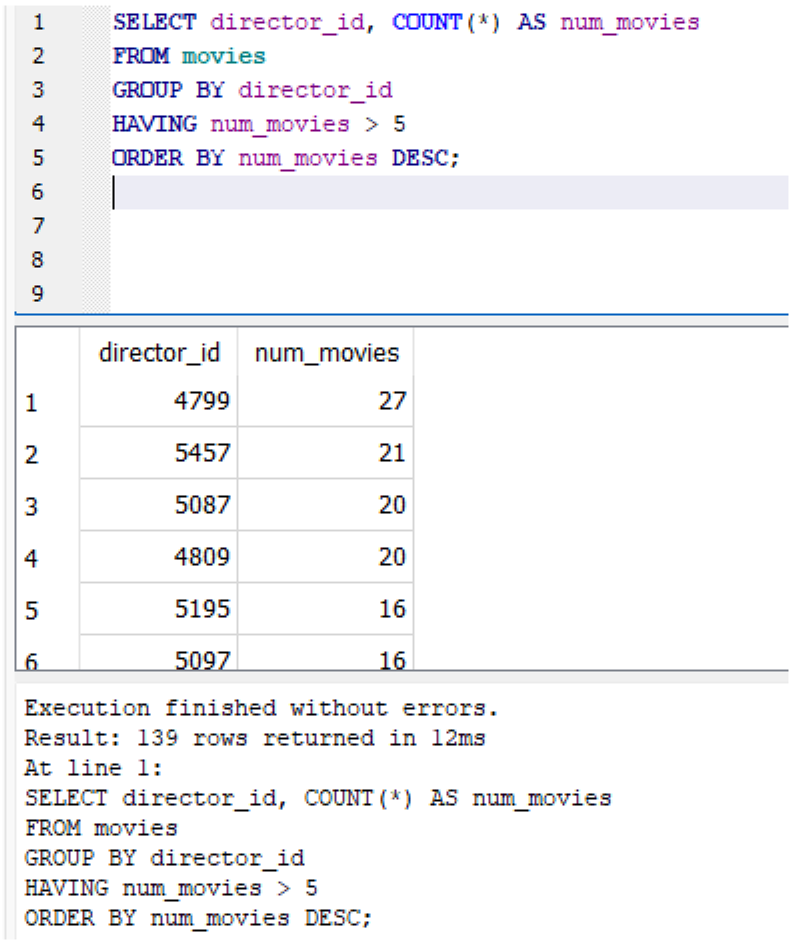
```
SELECT movies.title, directors.name
FROM movies
JOIN directors ON movies.director_id = directors.id
WHERE movies.revenue = 0 AND movies.uid > 10000;
```

	title	name
1	The Lovers	Roland Joffé
2	The Cat in the Hat	Bo Welch
3	Son of the Mask	Lawrence Guterman
4	Volcano	Mick Jackson
5	Arthur Christmas	Barry Cook
6	RED 2	Dean Parisot

```
Execution finished without errors.
Result: 1207 rows returned in 5ms
At line 1:
SELECT movies.title, directors.name
FROM movies
JOIN directors ON movies.director_id = directors.id
WHERE movies.revenue = 0 AND movies.uid > 10000;
```

a.

10. Display the count of movies each director has made, but only show directors with more than 5 movies, ordered by the count of movies in descending order



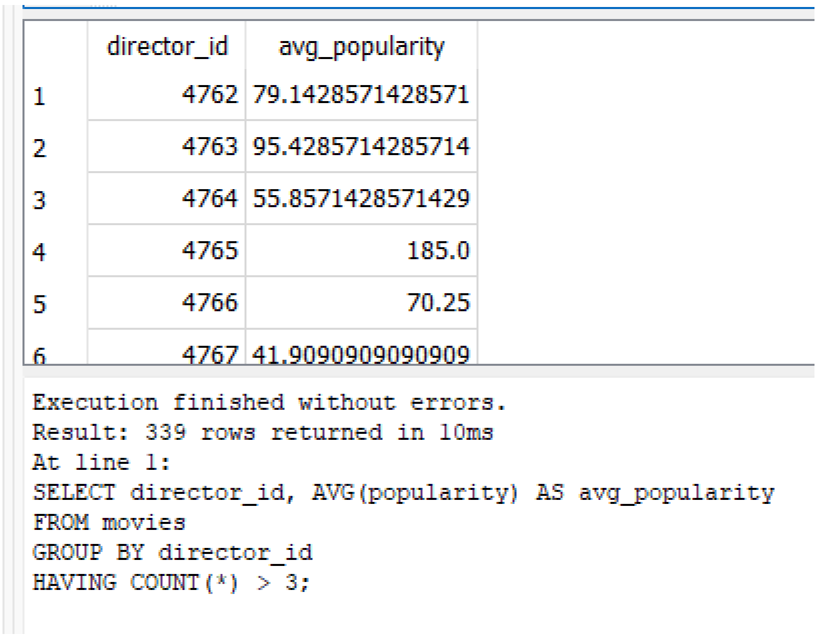
```sql
SELECT director_id, COUNT(*) AS num_movies
FROM movies
GROUP BY director_id
HAVING num_movies > 5
ORDER BY num_movies DESC;
```

	director_id	num_movies
1	4799	27
2	5457	21
3	5087	20
4	4809	20
5	5195	16
6	5097	16

```
Execution finished without errors.
Result: 139 rows returned in 12ms
At line 1:
SELECT director_id, COUNT(*) AS num_movies
FROM movies
GROUP BY director_id
HAVING num_movies > 5
ORDER BY num_movies DESC;
```

a.

11. Display the average popularity of movies for each director, but only include directors who have made more than 3 movies.

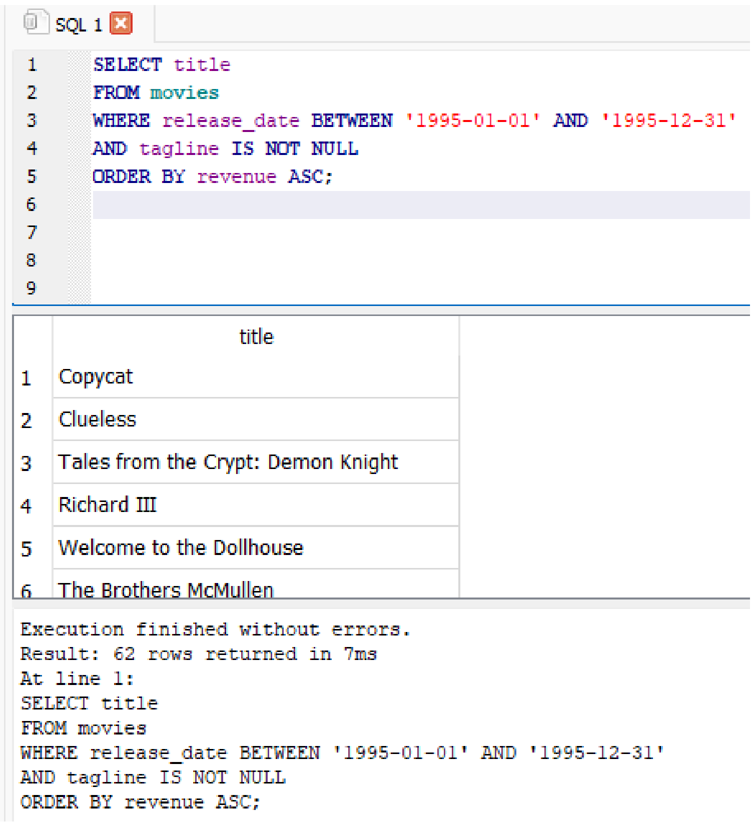


	director_id	avg_popularity
1	4762	79.1428571428571
2	4763	95.4285714285714
3	4764	55.8571428571429
4	4765	185.0
5	4766	70.25
6	4767	41.9090909090909

```
Execution finished without errors.
Result: 339 rows returned in 10ms
At line 1:
SELECT director_id, AVG(popularity) AS avg_popularity
FROM movies
GROUP BY director_id
HAVING COUNT(*) > 3;
```

a.

12. Show the title of movies that have a tagline and were released between ‘1995-01-01’ and ‘1995-12-31’, ordered by revenue in ascending order



SQL 1

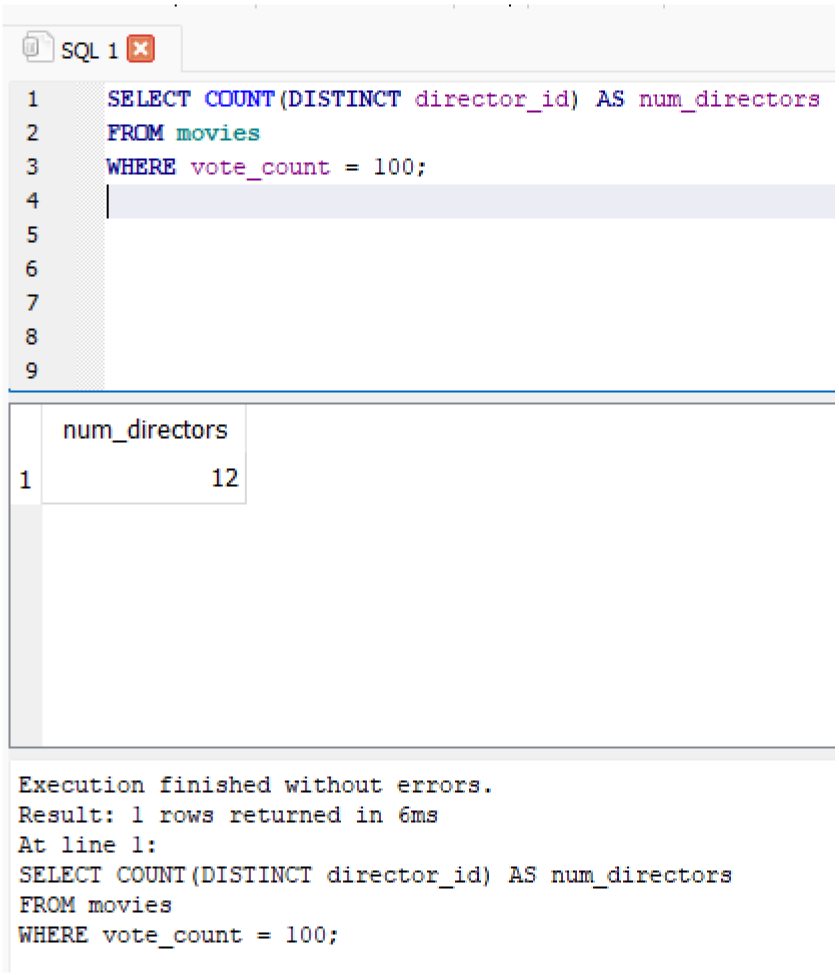
```
SELECT title
FROM movies
WHERE release_date BETWEEN '1995-01-01' AND '1995-12-31'
AND tagline IS NOT NULL
ORDER BY revenue ASC;
```

	title
1	Copycat
2	Clueless
3	Tales from the Crypt: Demon Knight
4	Richard III
5	Welcome to the Dollhouse
6	The Brothers McMullen

```
Execution finished without errors.
Result: 62 rows returned in 7ms
At line 1:
SELECT title
FROM movies
WHERE release_date BETWEEN '1995-01-01' AND '1995-12-31'
AND tagline IS NOT NULL
ORDER BY revenue ASC;
```

a.

13. Count the number of directors who have directed a movie with a vote count of exactly ‘100’


SQL 1

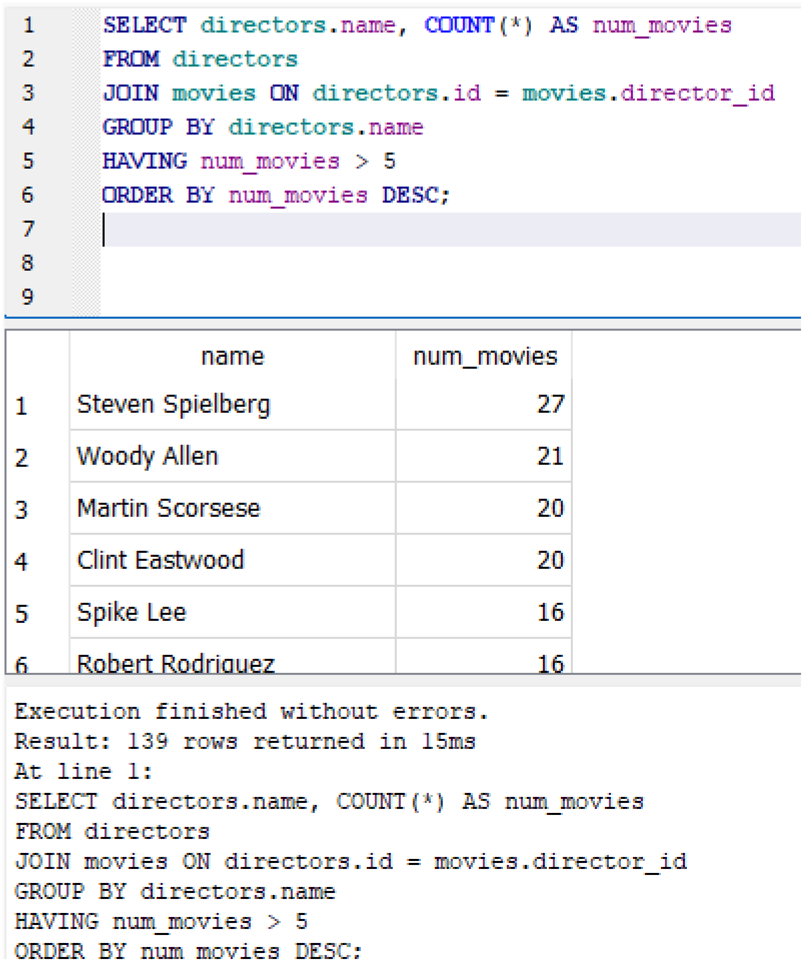
```
SELECT COUNT(DISTINCT director_id) AS num_directors
FROM movies
WHERE vote_count = 100;
```

```
   num_directors
1             12
```

```
Execution finished without errors.
Result: 1 rows returned in 6ms
At line 1:
SELECT COUNT(DISTINCT director_id) AS num_directors
FROM movies
WHERE vote_count = 100;
```

a.

14. Count the number of movies each director has directed and order the directors by this count in descending order, showing only directors who have directed more than 5 movies.



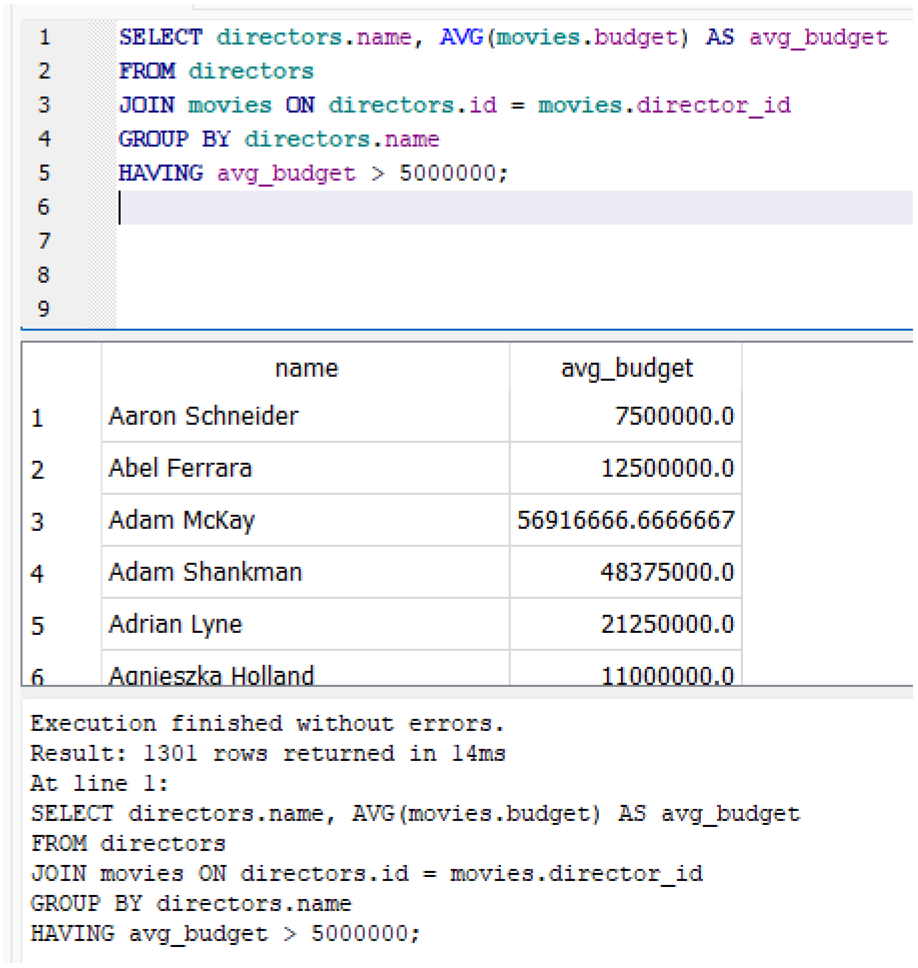
```
SELECT directors.name, COUNT(*) AS num_movies
FROM directors
JOIN movies ON directors.id = movies.director_id
GROUP BY directors.name
HAVING num_movies > 5
ORDER BY num_movies DESC;
```

	name	num_movies
1	Steven Spielberg	27
2	Woody Allen	21
3	Martin Scorsese	20
4	Clint Eastwood	20
5	Spike Lee	16
6	Robert Rodriguez	16

```
Execution finished without errors.
Result: 139 rows returned in 15ms
At line 1:
SELECT directors.name, COUNT(*) AS num_movies
FROM directors
JOIN movies ON directors.id = movies.director_id
GROUP BY directors.name
HAVING num_movies > 5
ORDER BY num_movies DESC;
```

a.

15. Find the average budget of movies for each director, showing only those with an average movie budget of over 5 million



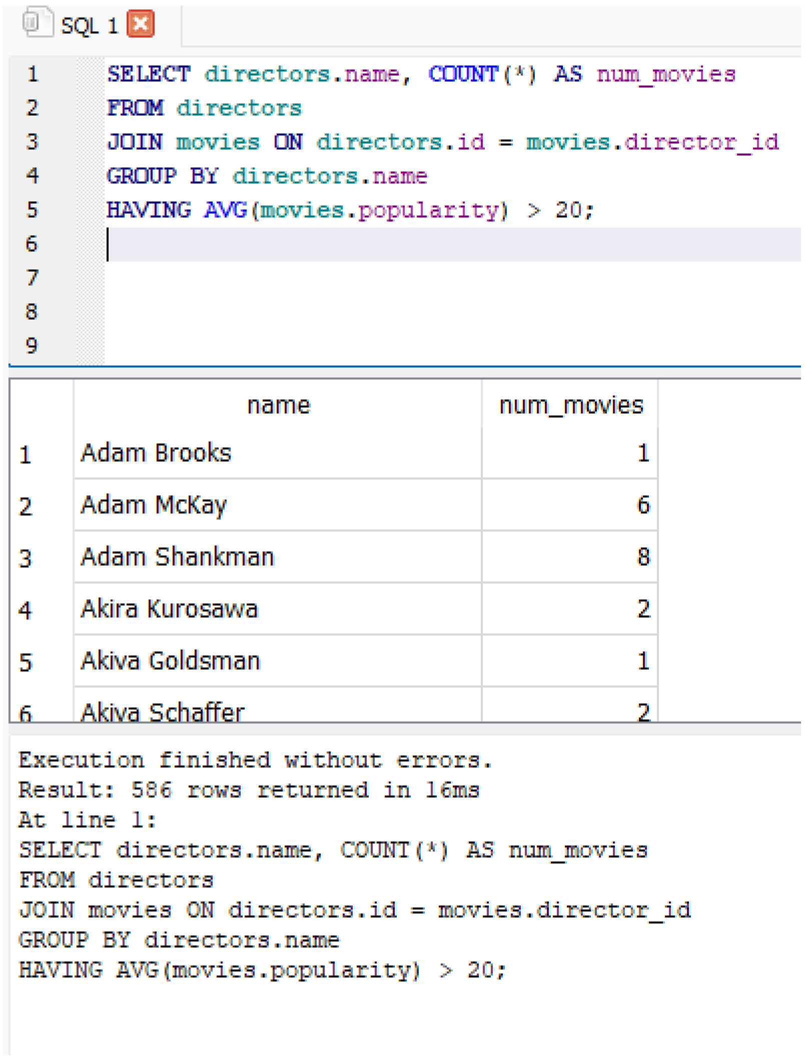
```
SELECT directors.name, AVG(movies.budget) AS avg_budget
FROM directors
JOIN movies ON directors.id = movies.director_id
GROUP BY directors.name
HAVING avg_budget > 5000000;
```

	name	avg_budget
1	Aaron Schneider	7500000.0
2	Abel Ferrara	12500000.0
3	Adam McKay	56916666.6666667
4	Adam Shankman	48375000.0
5	Adrian Lyne	21250000.0
6	Agnieszka Holland	11000000.0

```
Execution finished without errors.
Result: 1301 rows returned in 14ms
At line 1:
SELECT directors.name, AVG(movies.budget) AS avg_budget
FROM directors
JOIN movies ON directors.id = movies.director_id
GROUP BY directors.name
HAVING avg_budget > 5000000;
```

a.

16. Show the number of movies each director has in the database, but only for those directors whose movies have an average popularity of more than 20


SQL 1

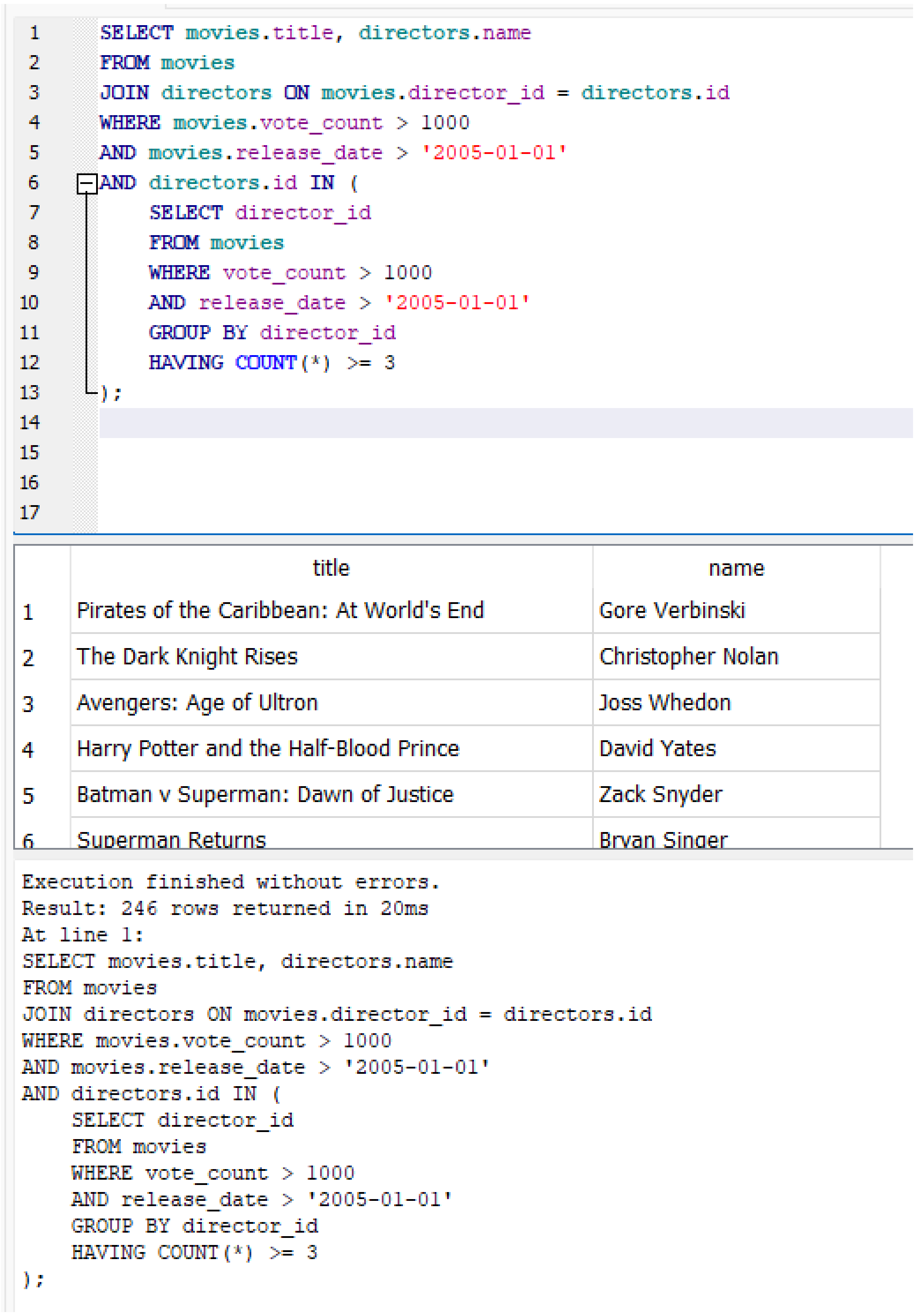
```sql
SELECT directors.name, COUNT(*) AS num_movies
FROM directors
JOIN movies ON directors.id = movies.director_id
GROUP BY directors.name
HAVING AVG(movies.popularity) > 20;
```

	name	num_movies
1	Adam Brooks	1
2	Adam McKay	6
3	Adam Shankman	8
4	Akira Kurosawa	2
5	Akiva Goldsman	1
6	Akiva Schaffer	2

```
Execution finished without errors.
Result: 586 rows returned in 16ms
At line 1:
SELECT directors.name, COUNT(*) AS num_movies
FROM directors
JOIN movies ON directors.id = movies.director_id
GROUP BY directors.name
HAVING AVG(movies.popularity) > 20;
```

a.

17. List movies and their directors for films that have received more than 1000 votes and were released after 2005. Include only directors with at least 3 movies meeting these criteria



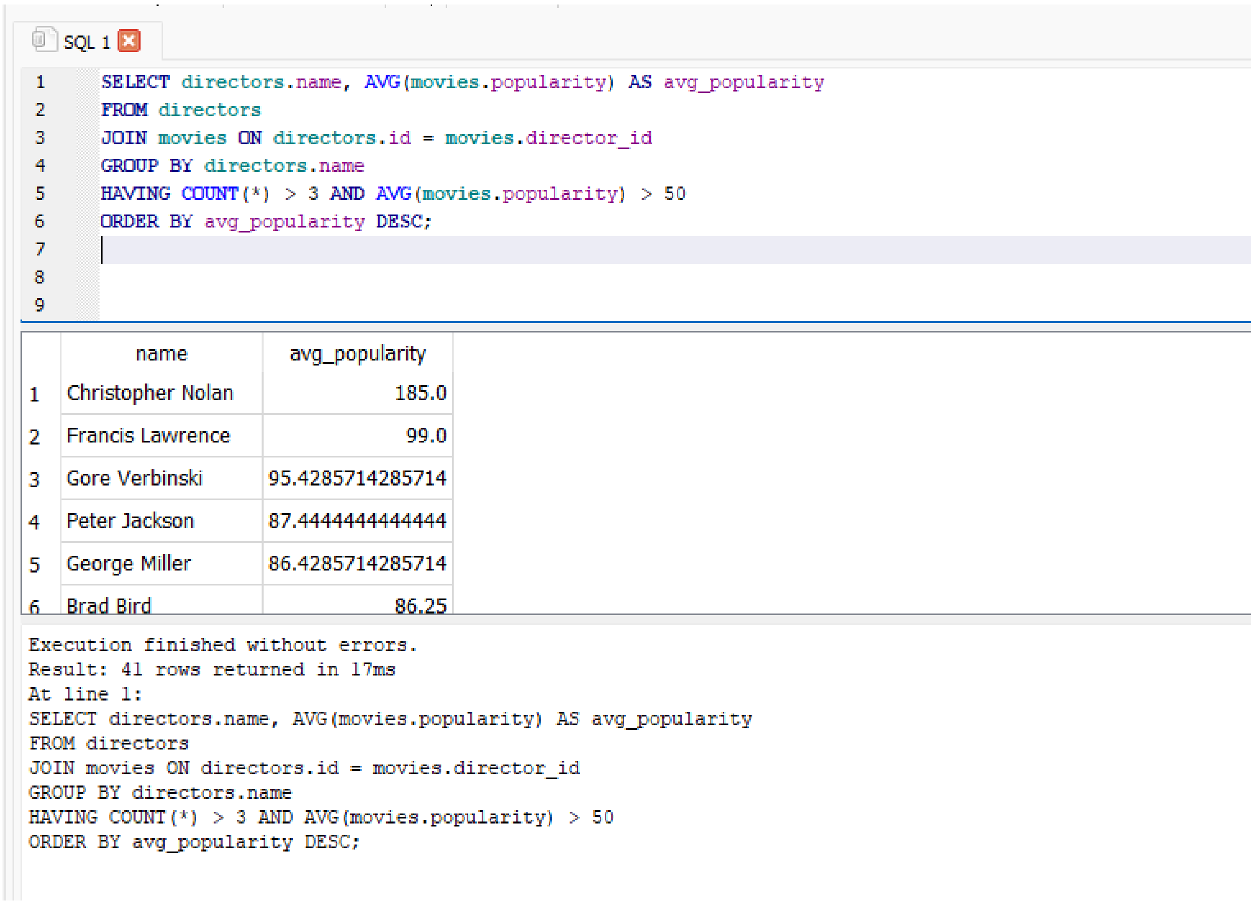
```
SELECT movies.title, directors.name
FROM movies
JOIN directors ON movies.director_id = directors.id
WHERE movies.vote_count > 1000
AND movies.release_date > '2005-01-01'
AND directors.id IN (
    SELECT director_id
    FROM movies
    WHERE vote_count > 1000
    AND release_date > '2005-01-01'
    GROUP BY director_id
    HAVING COUNT(*) >= 3
);
```

	title	name
1	Pirates of the Caribbean: At World's End	Gore Verbinski
2	The Dark Knight Rises	Christopher Nolan
3	Avengers: Age of Ultron	Joss Whedon
4	Harry Potter and the Half-Blood Prince	David Yates
5	Batman v Superman: Dawn of Justice	Zack Snyder
6	Superman Returns	Bryan Singer

```
Execution finished without errors.
Result: 246 rows returned in 20ms
At line 1:
SELECT movies.title, directors.name
FROM movies
JOIN directors ON movies.director_id = directors.id
WHERE movies.vote_count > 1000
AND movies.release_date > '2005-01-01'
AND directors.id IN (
    SELECT director_id
    FROM movies
    WHERE vote_count > 1000
    AND release_date > '2005-01-01'
    GROUP BY director_id
    HAVING COUNT(*) >= 3
);
```

a.

18. List the directors who have directed more than three movies with an average popularity of over 50, ordered by the average popularity



SQL 1

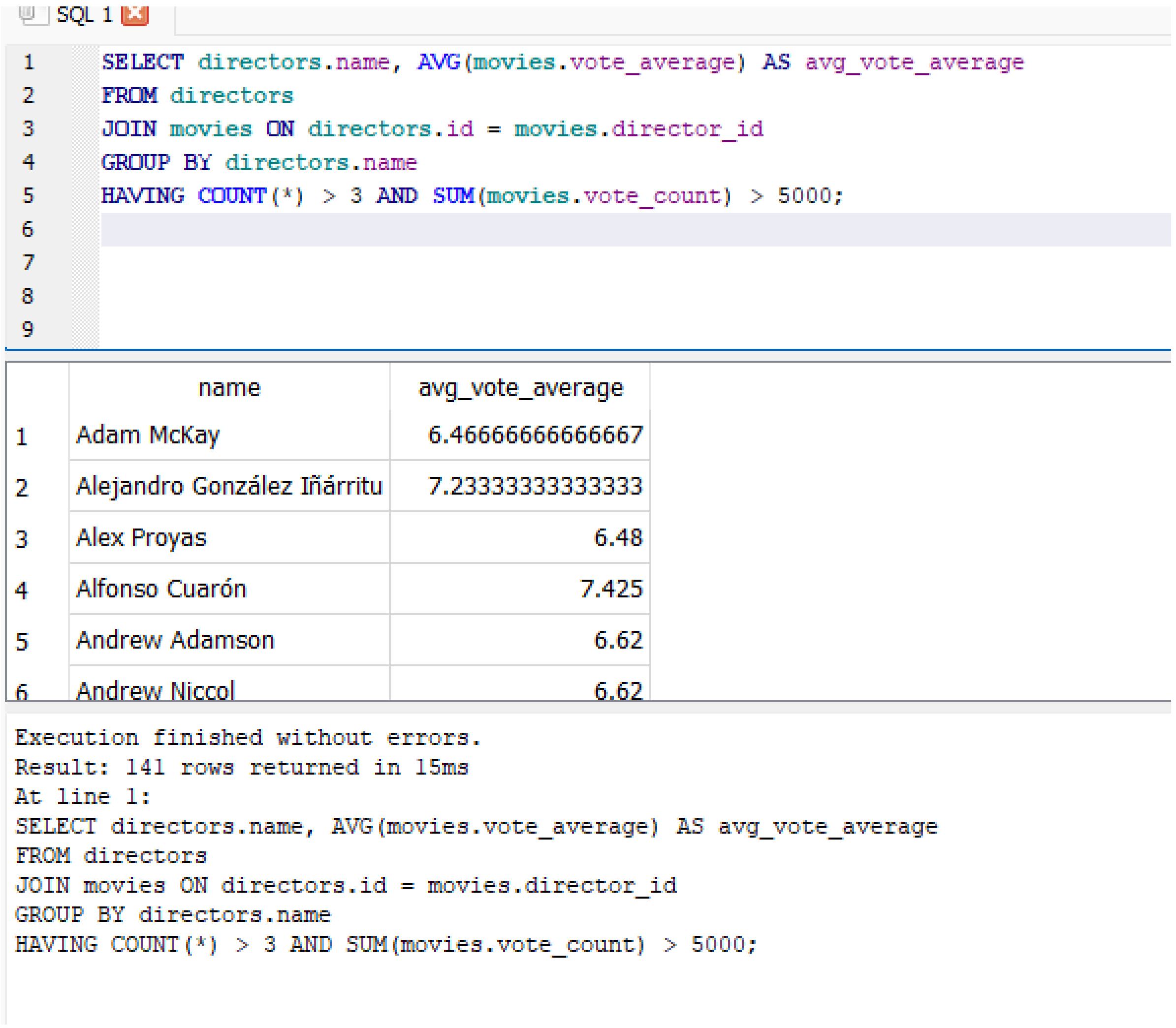
```sql
SELECT directors.name, AVG(movies.popularity) AS avg_popularity
FROM directors
JOIN movies ON directors.id = movies.director_id
GROUP BY directors.name
HAVING COUNT(*) > 3 AND AVG(movies.popularity) > 50
ORDER BY avg_popularity DESC;
```

	name	avg_popularity
1	Christopher Nolan	185.0
2	Francis Lawrence	99.0
3	Gore Verbinski	95.4285714285714
4	Peter Jackson	87.4444444444444
5	George Miller	86.4285714285714
6	Brad Bird	86.25

```
Execution finished without errors.
Result: 41 rows returned in 17ms
At line 1:
SELECT directors.name, AVG(movies.popularity) AS avg_popularity
FROM directors
JOIN movies ON directors.id = movies.director_id
GROUP BY directors.name
HAVING COUNT(*) > 3 AND AVG(movies.popularity) > 50
ORDER BY avg_popularity DESC;
```

a.

19. List the directors and their average vote_average, including only those directors whose movies have garnered more than 5000 votes in total and have directed more than 3 movies


SQL 1

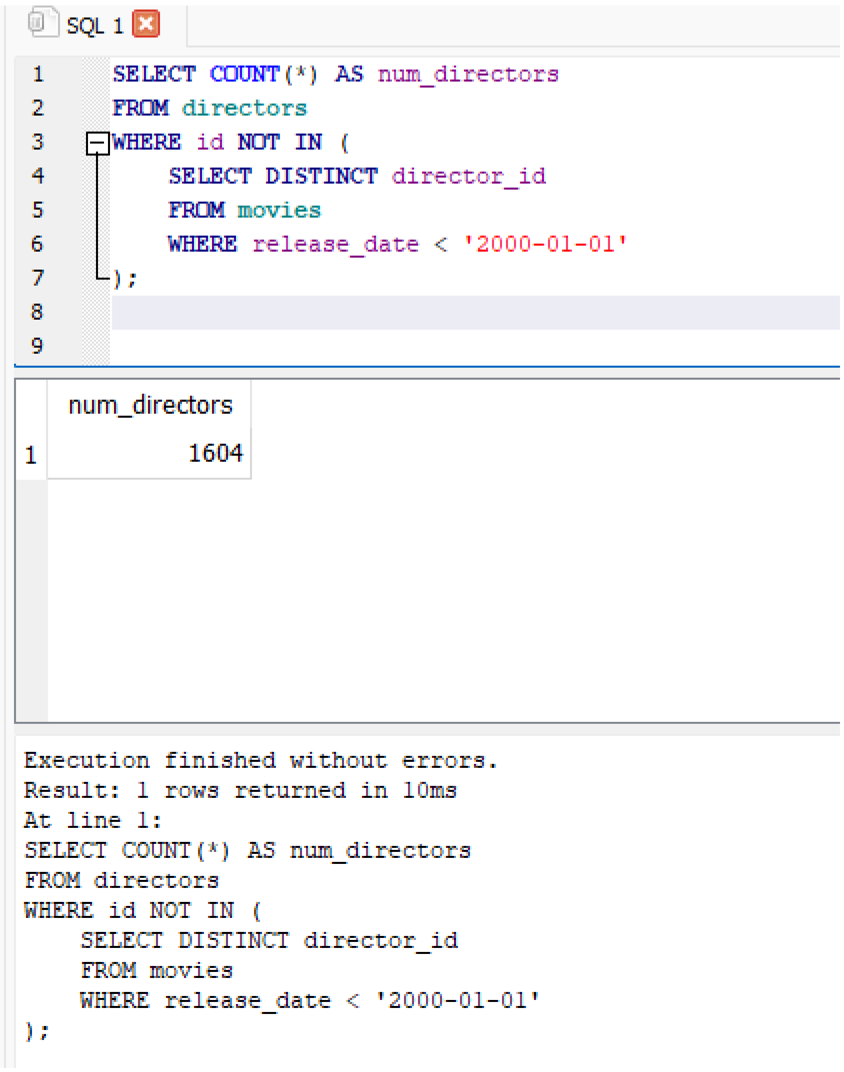
```sql
SELECT directors.name, AVG(movies.vote_average) AS avg_vote_average
FROM directors
JOIN movies ON directors.id = movies.director_id
GROUP BY directors.name
HAVING COUNT(*) > 3 AND SUM(movies.vote_count) > 5000;
```

	name	avg_vote_average
1	Adam McKay	6.46666666666667
2	Alejandro González Iñárritu	7.23333333333333
3	Alex Proyas	6.48
4	Alfonso Cuarón	7.425
5	Andrew Adamson	6.62
6	Andrew Niccol	6.62

```
Execution finished without errors.
Result: 141 rows returned in 15ms
At line 1:
SELECT directors.name, AVG(movies.vote_average) AS avg_vote_average
FROM directors
JOIN movies ON directors.id = movies.director_id
GROUP BY directors.name
HAVING COUNT(*) > 3 AND SUM(movies.vote_count) > 5000;
```

a.

20. Count the number of directors who have not directed any movie released before the year 2000



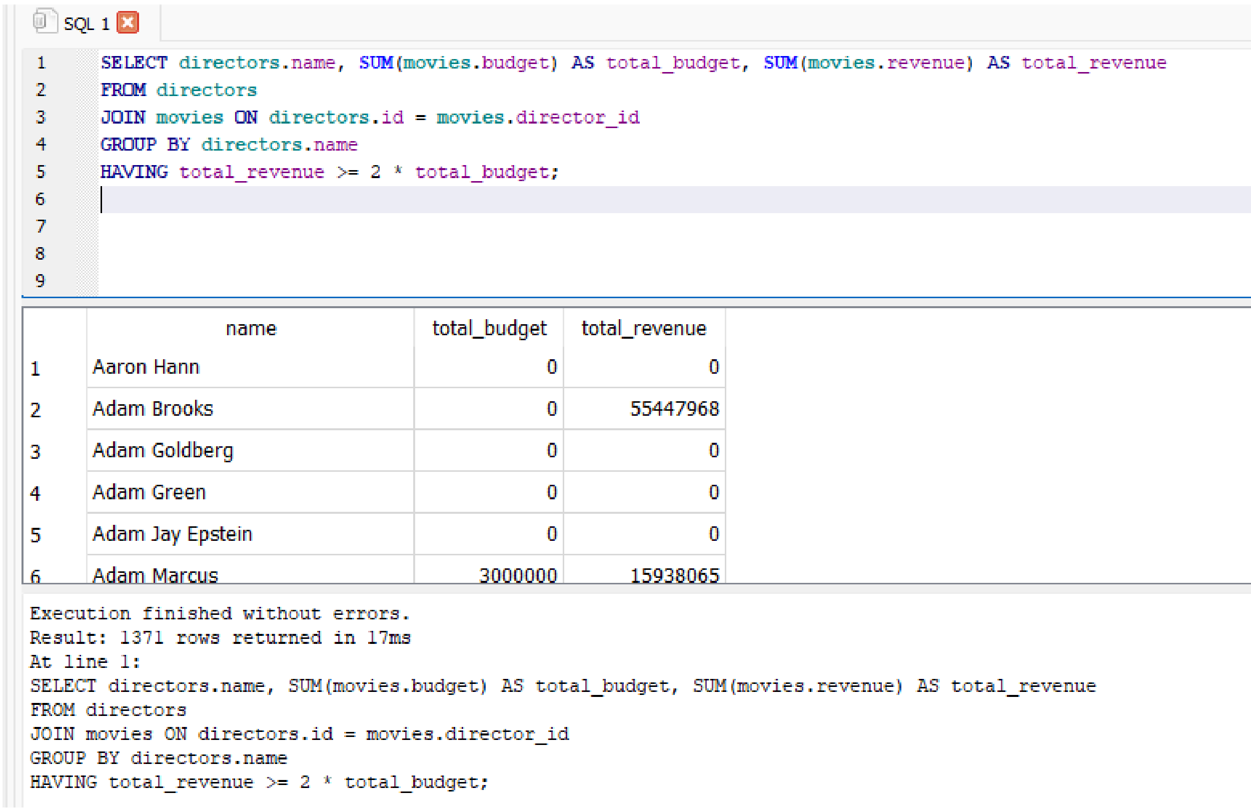
```
SELECT COUNT(*) AS num_directors
FROM directors
WHERE id NOT IN (
    SELECT DISTINCT director_id
    FROM movies
    WHERE release_date < '2000-01-01'
);
```

	num_directors
1	1604

```
Execution finished without errors.
Result: 1 rows returned in 10ms
At line 1:
SELECT COUNT(*) AS num_directors
FROM directors
WHERE id NOT IN (
    SELECT DISTINCT director_id
    FROM movies
    WHERE release_date < '2000-01-01'
);
```

a.

21. Show directors' names along with the total budget and total revenue of all their movies, for those who have earned at least twice as much revenue as the budget.


SQL 1

```
SELECT directors.name, SUM(movies.budget) AS total_budget, SUM(movies.revenue) AS total_revenue
FROM directors
JOIN movies ON directors.id = movies.director_id
GROUP BY directors.name
HAVING total_revenue >= 2 * total_budget;
```

	name	total_budget	total_revenue
1	Aaron Hann	0	0
2	Adam Brooks	0	55447968
3	Adam Goldberg	0	0
4	Adam Green	0	0
5	Adam Jay Epstein	0	0
6	Adam Marcus	3000000	15938065

```
Execution finished without errors.
Result: 1371 rows returned in 17ms
At line 1:
SELECT directors.name, SUM(movies.budget) AS total_budget, SUM(movies.revenue) AS total_revenue
FROM directors
JOIN movies ON directors.id = movies.director_id
GROUP BY directors.name
HAVING total_revenue >= 2 * total_budget;
```

a.