

Question 2)

SELECT DISTINCT

t.premiered AS premiere_year,

t.primary_title || '(' || t.original_title || ')' AS title_format

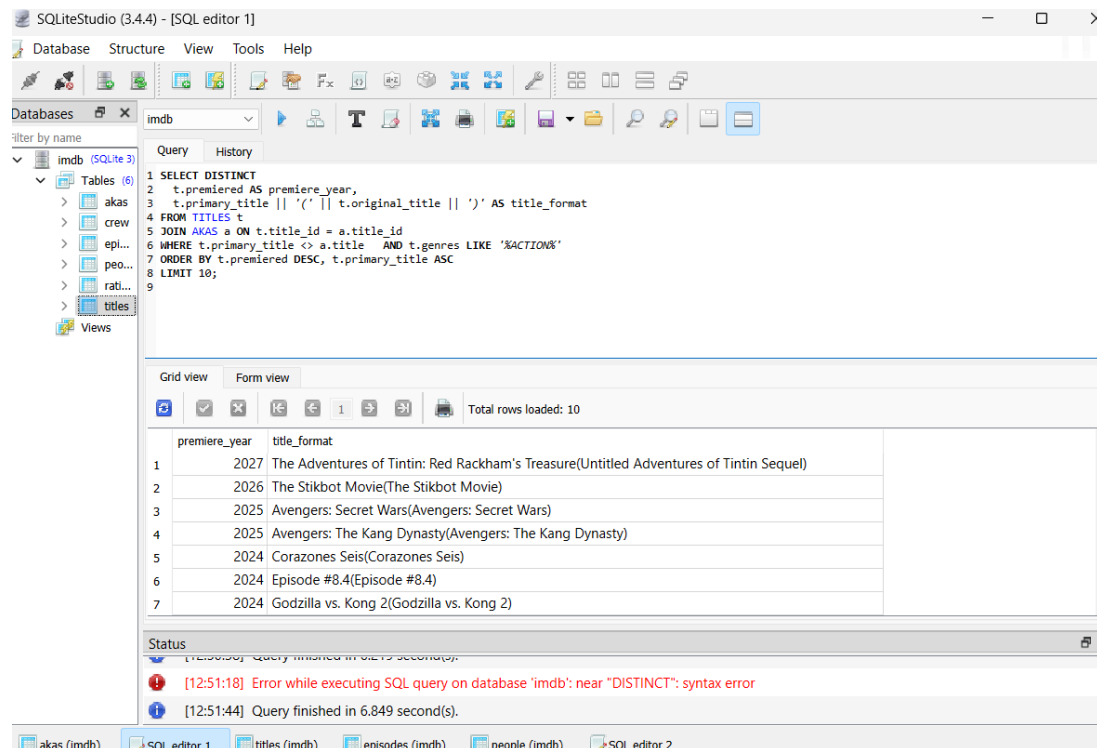
FROM TITLES t

JOIN AKAS a ON t.title_id = a.title_id

WHERE t.primary_title <> a.title AND t.genres LIKE '%ACTION%'

ORDER BY t.premiered DESC, t.primary_title ASC

LIMIT 10;



Question 3)

SELECT

t.primary_title,

CASE

WHEN t.ended IS NULL THEN 2024 - t.premiered

ELSE t.ended - t.premiered

END AS running_years

```

FROM TITLES t

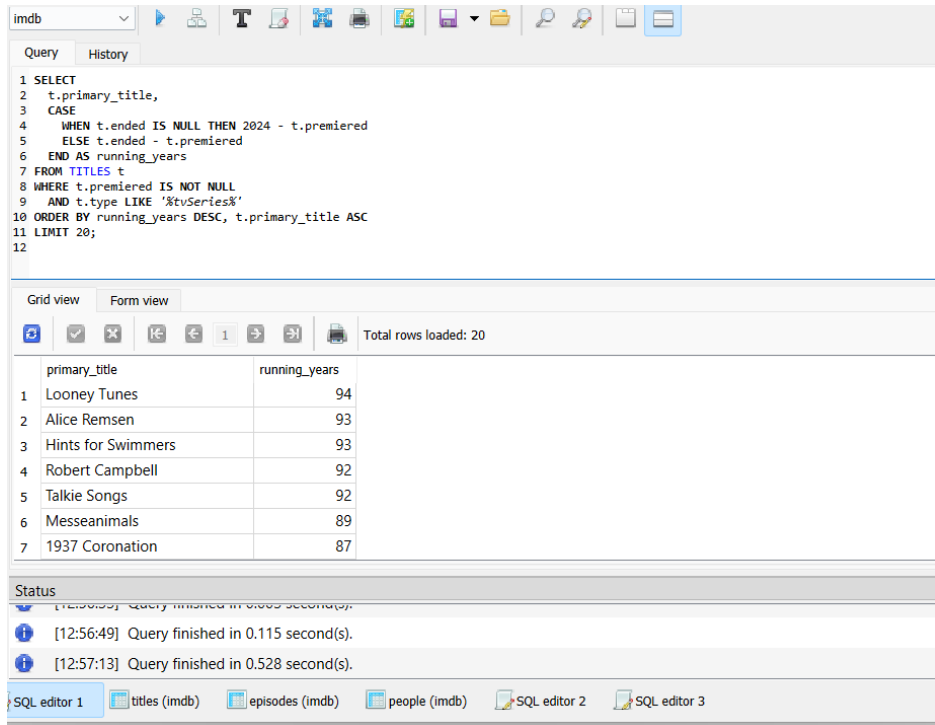
WHERE t.premiered IS NOT NULL

    AND t.type LIKE '%tvSeries%'

ORDER BY running_years DESC, t.primary_title ASC

LIMIT 20;

```



The screenshot shows a SQL query editor with the following query:

```

1 SELECT
2   t.primary_title,
3   CASE
4     WHEN t.ended IS NULL THEN 2024 - t.premiered
5     ELSE t.ended - t.premiered
6   END AS running_years
7 FROM TITLES t
8 WHERE t.premiered IS NOT NULL
9    AND t.type LIKE '%tvSeries%'
10 ORDER BY running_years DESC, t.primary_title ASC
11 LIMIT 20;
12

```

The results are displayed in a table with 2 columns: primary_title and running_years. The table shows the top 7 results.

	primary_title	running_years
1	Looney Tunes	94
2	Alice Remsen	93
3	Hints for Swimmers	93
4	Robert Campbell	92
5	Talkie Songs	92
6	Messeanimals	89
7	1937 Coronation	87

The status bar at the bottom shows the query finished in 0.115 second(s) and 0.528 second(s).

Question 4)

```

SELECT DISTINCT

    (FLOOR(born / 10) * 10) || 's' AS decade,

    COUNT(DISTINCT person_id) AS num_directors

FROM people

JOIN crew USING (person_id)

WHERE crew.category = 'director'

    AND born >= 1900

GROUP BY FLOOR(born / 10) * 10

ORDER BY FLOOR(born / 10) * 10;

```

imdb

Query

History

```

1 SELECT DISTINCT
2   (FLOOR(born / 10) * 10) || 's' AS decade,
3   COUNT(DISTINCT person_id) AS num_directors
4 FROM people
5 JOIN crew USING (person_id)
6 WHERE crew.category = 'director'
7       AND born >= 1900
8 GROUP BY FLOOR(born / 10) * 10
9 ORDER BY FLOOR(born / 10) * 10;
10

```

Grid view

Form view

Total rows loaded: 13

	decade	num_directors
1	1900s	376
2	1910s	389
3	1920s	721
4	1930s	810
5	1940s	999
6	1950s	1084
7	1960s	1625

Question 5)

SELECT

```
t.type AS TITLE_TYPE,  
ROUND(AVG(r.rating), 2) AS AVG_RATING,  
MIN(r.rating) AS MIN_RATING,  
MAX(r.rating) AS MAX_RATING
```

FROM

titles t

JOIN

akas a ON t.title_id = a.title_id

JOIN

```
ratings r ON t.title_id = r.title_id
```

WHERE

```

a.language = 'de' -- Language column for German titles
AND a.types IN ('imdbDisplay', 'original') -- akas types are either 'imdbDisplay' or 'original'

```

GROUP BY

t.type

ORDER BY

AVG_RATING ASC;

Grid view		Form view						
					1			
								Total rows loaded: 7
	TITLE TYPE	AVG RATING	MIN RATING	MAX RATING				
1	movie	6.65	3.4	8.2				
2	tvMovie	6.77	5.5	7.3				
3	tvEpisode	6.9	6.9	6.9				
4	short	7.12	5	8.1				
5	tvMiniSeries	7.2	7.2	7.2				
6	videoGame	7.2	7.2	7.2				
7	tvSeries	7.63	7	8.5				

Question 6)

WITH BatmanActors AS (

SELECT DISTINCT c.person_id

FROM crew c

WHERE c.characters LIKE '%Batman%'

AND c.category = 'actor'

),

ActorWorks AS (

SELECT

p.person_id,

p.name AS actor_name,

r.rating

FROM people p

JOIN crew c ON p.person_id = c.person_id

JOIN ratings r ON c.title_id = r.title_id

WHERE p.person_id IN (SELECT person_id FROM BatmanActors)

),

ActorRatings AS (

SELECT

actor_name,

ROUND(AVG(rating), 2) AS avg_rating

FROM ActorWorks

GROUP BY actor_name

)

```
SELECT actor_name, avg_rating
```

```
FROM ActorRatings
```

```
ORDER BY avg_rating DESC
```

```
LIMIT 10;
```

The screenshot shows the SQLStudio interface with a query editor and a results grid. The query is as follows:

```
1 WITH BatmanActors AS (  
2     SELECT DISTINCT c.person_id  
3     FROM crew c  
4     WHERE c.characters LIKE '%Batman%'  
5         AND c.category = 'actor'  
6 ),  
7 ActorWorks AS (  
8     SELECT  
9         p.person_id,  
10        p.name AS actor_name,  
11        r.rating  
12    FROM people p  
13    JOIN crew c ON p.person_id = c.person_id  
14    JOIN titles t ON c.title_id = t.title_id
```

The results grid shows the following data:

	actor_name	avg_rating
1	Rupert Raineri	8.6
2	William M. Lemke	8.2
3	David Mazouz	8.15
4	Adam Marcinowski	8.1
5	Kevin Porter	8.1
6	Kayd Currier	8.05
7	Kellen Goff	8.01

The status bar at the bottom indicates: [13:45:37] Query finished in 50.066 second(s).

Question 7)

```
WITH PrestigeYear AS (
```

```
-- Find the premiere year of "The Prestige"
```

```
SELECT premiered
```

```
FROM titles
```

```
WHERE primary_title = 'The Prestige'
```

```
),
```

```
ActorsBornInPrestigeYear AS (
```

```
-- Find the number of actors or actresses born in that year
```

```
SELECT COUNT(DISTINCT p.person_id) AS num_actors
```

```
FROM people p
```

```
JOIN crew c ON p.person_id = c.person_id
```

```
WHERE c.category IN ('actor', 'actress')
```

```
AND p.born = (SELECT premiered FROM PrestigeYear)
```

```
)
```

```
SELECT num_actors
FROM ActorsBornInPrestigeYear;
```

The screenshot shows a SQL query editor with a query window and a results window. The query is as follows:

```
1 WITH PrestigeYear AS (
2   -- Find the premiere year of "The Prestige"
3   SELECT premiered
4   FROM titles
5   WHERE primary_title = 'The Prestige'
6 ),
7 ActorsBornInPrestigeYear AS (
8   -- Find the number of actors or actresses born in that year
9   SELECT COUNT(DISTINCT p.person_id) AS num_actors
10  FROM people p
11  JOIN crew c ON p.person_id = c.person_id
12  WHERE c.category IN ('actor', 'actress')
13  AND p.born = (SELECT premiered FROM PrestigeYear)
14 )
```

The results window shows a table with one column, `num_actors`, and one row with the value 30. The status bar indicates "Total rows loaded: 1".

	num_actors
1	30

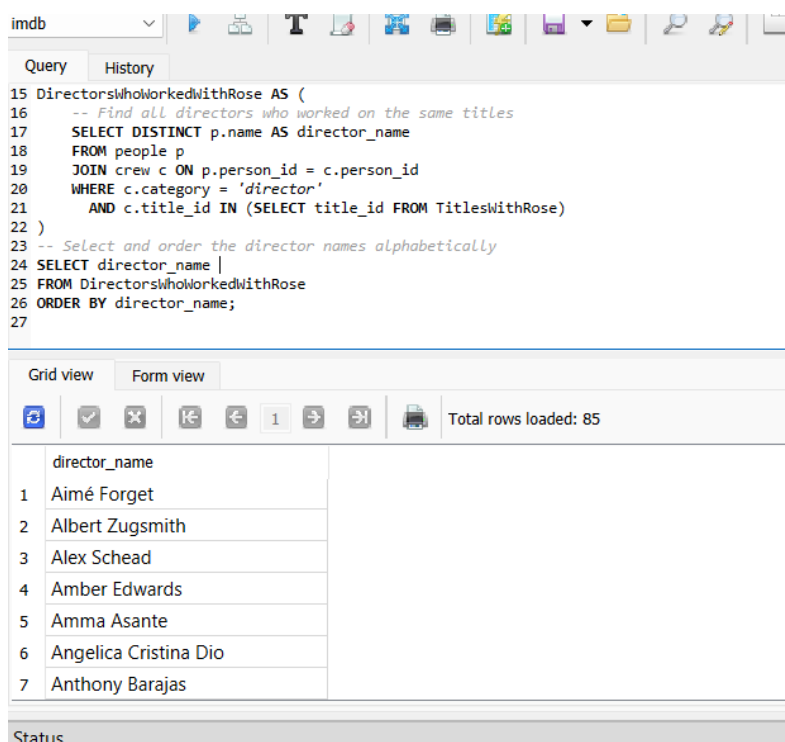
Question 8)

```
WITH ActressesNamedRose AS (
  -- Find all actresses with the first name "Rose"
  SELECT DISTINCT p.person_id
  FROM people p
  JOIN crew c ON p.person_id = c.person_id
  WHERE c.category = 'actress'
  AND p.name LIKE 'Rose%'
),
TitlesWithRose AS (
  -- Find all titles that these actresses have worked on
  SELECT DISTINCT c.title_id
  FROM crew c
  WHERE c.person_id IN (SELECT person_id FROM ActressesNamedRose)
),
DirectorsWhoWorkedWithRose AS (
  -- Find all directors who worked on the same titles
  SELECT DISTINCT p.name AS director_name
```

```

FROM people p
JOIN crew c ON p.person_id = c.person_id
WHERE c.category = 'director'
      AND c.title_id IN (SELECT title_id FROM TitlesWithRose)
)
-- Select and order the director names alphabetically
SELECT director_name
FROM DirectorsWhoWorkedWithRose
ORDER BY director_name;

```



The screenshot shows a database query tool interface. The top toolbar includes icons for various database operations. Below the toolbar, the 'Query' tab is active, displaying the following SQL code:

```

15 DirectorsWhoWorkedWithRose AS (
16     -- Find all directors who worked on the same titles
17     SELECT DISTINCT p.name AS director_name
18     FROM people p
19     JOIN crew c ON p.person_id = c.person_id
20     WHERE c.category = 'director'
21           AND c.title_id IN (SELECT title_id FROM TitlesWithRose)
22 )
23 -- Select and order the director names alphabetically
24 SELECT director_name |
25 FROM DirectorsWhoWorkedWithRose
26 ORDER BY director_name;
27

```

Below the query editor, the 'Grid view' tab is active, showing the results of the query. The results are displayed in a table with two columns: 'director_name' and an empty column. The first seven rows are visible, showing the following director names:

	director_name
1	Aimé Forget
2	Albert Zugsmith
3	Alex Schead
4	Amber Edwards
5	Amma Asante
6	Angelica Cristina Dio
7	Anthony Barajas

The status bar at the bottom indicates 'Total rows loaded: 85'.

Question 9)

```

WITH RankedActors AS (
SELECT
p.name,
c.category,
p.died,
t.runtime_minutes,
t.primary_title AS longest_work_title, -- Use primary_title instead of title
ROW_NUMBER() OVER (
PARTITION BY c.category, p.name

```

```

ORDER BY t.runtime_minutes DESC, t.title_id ASC
) AS rank
FROM
people p
JOIN
crew c ON p.person_id = c.person_id
JOIN
titles t ON c.title_id = t.title_id
WHERE
p.died IS NOT NULL
AND t.runtime_minutes IS NOT NULL
),
Top5ActorsPerCategory AS (
SELECT
ra.name,
ra.category,
ra.died,
ra.runtime_minutes,
ra.longest_work_title, -- Include the longest work title
ROW_NUMBER() OVER (
PARTITION BY ra.category
ORDER BY ra.died ASC, ra.name ASC
) AS category_rank
FROM
RankedActors ra
WHERE
ra.rank = 1
)
SELECT
category,
name,
died,

```


longest_work_title, -- Include the longest work title in the final output

runtime_minutes,

category_rank

FROM

Top5ActorsPerCategory

WHERE

category_rank <= 5

ORDER BY

category, -- Sort alphabetically by category

died, -- Then by year of death

name

```
1 /
2 SELECT
3   category,
4   name,
5   died,
6   longest_work_title, -- Include the longest work title in the final output
7   runtime_minutes,
8   category_rank
9 FROM
10  Top5ActorsPerCategory
11 WHERE
12  category_rank <= 5
13 ORDER BY
14  category, -- Sort alphabetically by category
15  died, -- Then by year of death
16  name; -- Finally by alphabetical order of name
17
```

Grid viewForm view

<

Question 10)