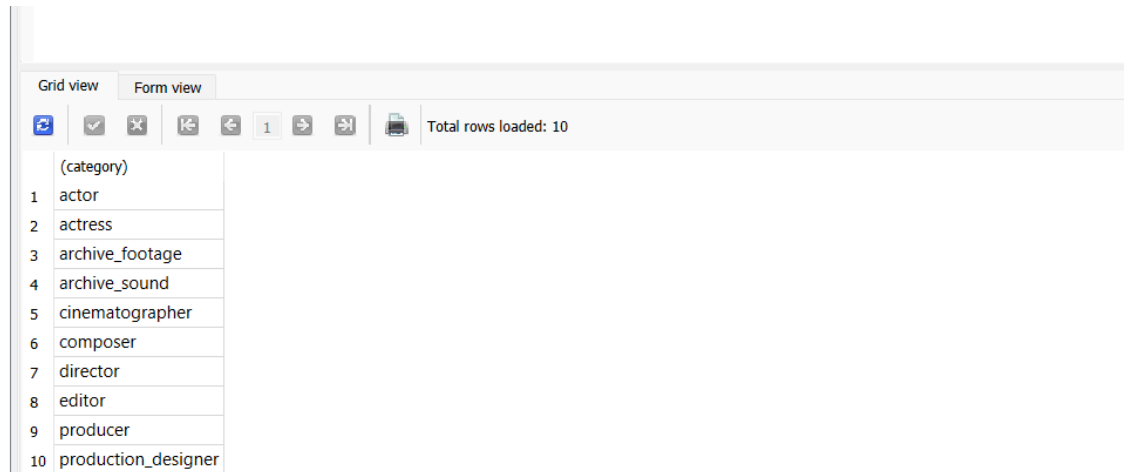


Question 1)

```
SELECT DISTINCT(category) FROM crew
```

```
ORDER BY category
```

```
LIMIT 10;
```



	(category)
1	actor
2	actress
3	archive_footage
4	archive_sound
5	cinematographer
6	composer
7	director
8	editor
9	producer
10	production_designer

Question 2

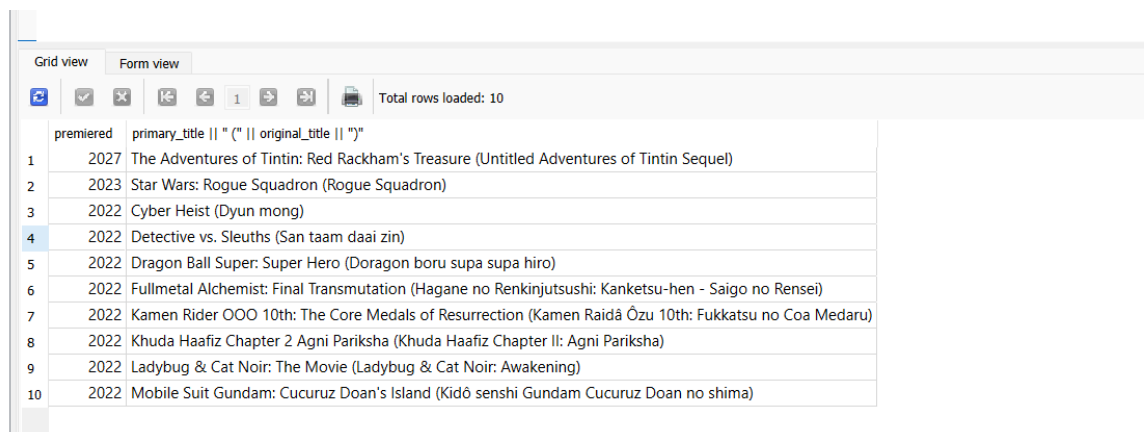
```
select premiered, primary_title || " (" || original_title || ")"
```

```
from titles
```

```
where primary_title <> original_title and type == 'movie' and genres like '%Action%'
```

```
order by premiered desc, primary_title asc
```

```
limit 10;
```



	premiered	primary_title " (" original_title ")"
1	2027	The Adventures of Tintin: Red Rackham's Treasure (Untitled Adventures of Tintin Sequel)
2	2023	Star Wars: Rogue Squadron (Rogue Squadron)
3	2022	Cyber Heist (Dyun mong)
4	2022	Detective vs. Sleuths (San taam daai zin)
5	2022	Dragon Ball Super: Super Hero (Doragon boru supa supa hiro)
6	2022	Fullmetal Alchemist: Final Transmutation (Hagane no Renkinjutsushi: Kanketsu-hen - Saigo no Rensei)
7	2022	Kamen Rider OOO 10th: The Core Medals of Resurrection (Kamen Raidā Ōzu 10th: Fukkatsu no Coa Medaru)
8	2022	Khuda Haafiz Chapter 2 Agni Pariksha (Khuda Haafiz Chapter II: Agni Pariksha)
9	2022	Ladybug & Cat Noir: The Movie (Ladybug & Cat Noir: Awakening)
10	2022	Mobile Suit Gundam: Cucuruz Doan's Island (Kidō senshi Gundam Cucuruz Doan no shima)

Question 3

```
SELECT primary_title,
```

```
       CASE WHEN ended IS NOT NULL THEN ended - premiered
```

```
       ELSE 2023 - premiered
```

```

END AS runtime

FROM titles

WHERE premiered NOT NULL AND type = 'tvSeries'

ORDER BY runtime DESC,

primary_title ASC

LIMIT 20;

```

Grid view		Form view
		Total rows loaded: 20
	primary_title	runtime
1	Looney Tunes	93
2	Alice Remsen	92
3	Hints for Swimmers	92
4	Robert Campbell	91
5	Talkie Songs	91
6	Messeanimals	88
7	1937 Coronation	86
8	Sports Review	86
9	The Disorderly Room	86
10	Craftsmen at Work	85
11	Looney Tunes	84
12	WNBC-TV News	82

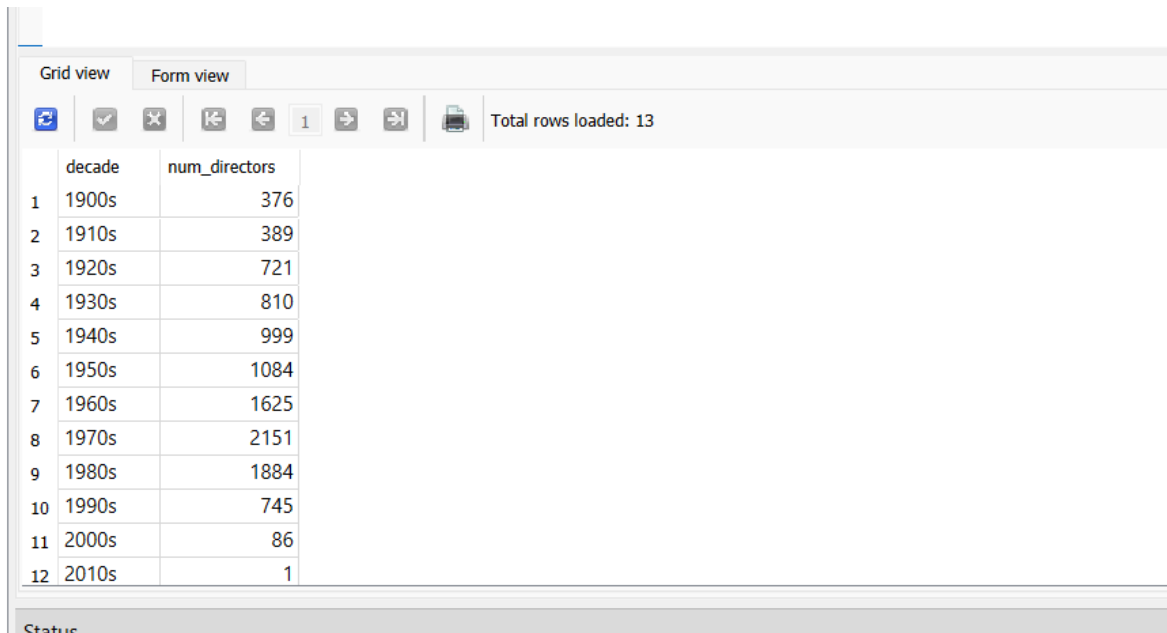
Question 4

```

SELECT CAST(born / 10 * 10 AS TEXT) || 's' AS decade,
        COUNT(DISTINCT(people.person_id)) AS num_directors
FROM people
        INNER JOIN crew ON people.person_id = crew.person_id
WHERE
        born IS NOT NULL
        AND category == 'director'
        AND born >= 1900
GROUP BY decade ;

```

ORDER BY decade ;



The screenshot shows a data grid interface with a toolbar at the top containing icons for various actions like refresh, save, delete, and navigation. The grid displays two columns: 'decade' and 'num_directors'. The data is sorted by decade in ascending order. The total number of rows loaded is 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
8	1970s	2151
9	1980s	1884
10	1990s	745
11	2000s	86
12	2010s	1

QUESTION 5

SELECT

t.type,
ROUND(AVG(r.rating), 2) AS avg_rating,
MIN(r.rating),
MAX(r.rating)

FROM

akas as a
INNER JOIN ratings as r ON r.title_id = a.title_id
INNER JOIN titles as t ON t.title_id = a.title_id

WHERE

a.language = 'de'
AND (
a.types = 'imdbDisplay'
OR a.types = 'original'
)

GROUP BY t.type

ORDER BY avg_rating;

19 avg_rating;

Grid view Form view

Total rows loaded: 7

	type	avg_rating	MIN(r.rating)	MAX(r.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 batman_actors AS(

SELECT DISTINCT(people.person_id) AS id, name

FROM

people

INNER JOIN crew ON people.person_id = crew.person_id

WHERE

characters LIKE '%"Batman"%'

AND category == "actor"

)

SELECT

name,

ROUND(AVG(rating), 2) as avg_rating

FROM

ratings

INNER JOIN crew ON ratings.title_id == crew.title_id

INNER JOIN batman_actors ON crew.person_id == batman_actors.id

GROUP BY batman_actors.id

ORDER BY avg_rating DESC

LIMIT 10;(tried a lots of time the output doesn't matc



	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
8	Jason Steele	7.71
9	Jeremy Kewley	7.67
10	Jason O'Mara	7.62

Question 7

SELECT COUNT(DISTINCT people.person_id)

FROM people

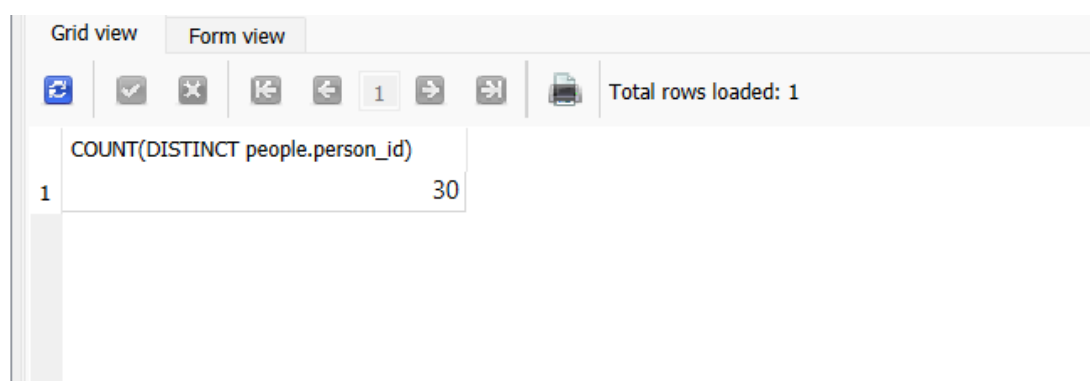
INNER JOIN crew ON people.person_id == crew.person_id

WHERE

born IN (SELECT premiered FROM titles WHERE primary_title == "The Prestige")

AND (category == "actor" OR category == "actress")

ORDER BY name;



	COUNT(DISTINCT people.person_id)
1	30

Question 8









WITH rose_titles AS (SELECT DISTINCT(c.title_id)

FROM people AS P

```

INNER JOIN crew AS c ON c.person_id = p.person_id
WHERE category = 'actress' AND name LIKE 'Rose%'
), rose_directors AS (SELECT DISTINCT(p.person_id)
FROM people AS p
INNER JOIN crew AS c ON c.person_id = p.person_id
WHERE
c.category = 'director'AND c.title_id IN rose_titles
)
SELECT name
FROM people AS p
INNER JOIN rose_directors AS rd ON rd.person_id = p.person_id
ORDER BY name ASC;

```

Grid view Form view	
     1    Total rows loaded: 68	
name	
1	Aimé Forget
2	Albert Zugsmith
3	Alex Schead
4	Anthony Barajas
5	Anthony Parker
6	Asad Panjwani
7	Bill Hitchcock
8	Bruce Kessler
9	Charles Beeson
10	Colin Barr
11	Cullen McGraw
12	Daniel Leahey

Question 9

WITH longest_per_person AS (SELECT

category,

name,

died,

```

        primary_title,
        runtime_minutes,
        DENSE_RANK() OVER (
            PARTITION BY c.category
            ORDER BY
                p.died ASC,
                p.name ASC
        ) AS rank_num,
        DENSE_RANK() OVER (
            PARTITION BY c.category,
                c.person_id
            ORDER BY
                runtime_minutes DESC,
                t.title_id ASC
        ) AS rank_num_runtime
FROM
    crew AS c
    INNER JOIN people as p ON p.person_id =
c.person_id
    INNER JOIN titles as t ON c.title_id = t.title_id
WHERE
    p.died NOT NULL
    AND runtime_minutes NOT NULL
),
top_titles AS (
    SELECT
        category,
        name,

```

```

        died,
        primary_title,
        runtime_minutes,
        rank_num
    FROM
        longest_per_person as lpp
    WHERE
        rank_num <= 5
        AND rank_num_runtime = 1
)
SELECT
    *
FROM
    top_titles
ORDER BY
    category ASC,
    rank_num ASC;

```

	category	name	died	primary_title	runtime_min	rank_num
1	actor	Verner Clarges	1911	A Summer Idyll	17	1
2	actor	George Ober	1912	The Curio Hunters	9	2
3	actor	Antonio Grisanti	1914	Parsifal	50	3
4	actor	Robert Drouet	1914	Beverly of Graustark	57	4
5	actor	John Bunny	1915	Hearts and Diamonds	33	5
6	actress	Mrs. E.M. Kimball	1919	The Delayed Letter	11	1
7	actress	Clarine Seymour	1920	The Idol Dancer	104	2
8	actress	Katherine Griffith	1921	Huckleberry Finn	70	3
9	actress	Virginia Rappe	1921	His Musical Sneeze	19	4
10	actress	Lillian Russell	1922	La Tosca	30	5
11	archive_footage	Grand Duchess Alice	1878	Constraint	59	1
12	archive_footage	King Edward VII	1910	The House of Windsor	90	2

Question 10

```

WITH json_table(json_data) AS ( SELECT c.characters as json_data
    FROM
        people AS p,
        crew AS c

```




```


WHERE
    p.name = "Leonardo DiCaprio"
    AND p.born = 1974
    AND p.person_id = c.person_id
ORDER BY
    c.characters
),
characters(character) AS (
    SELECT
        DISTINCT value as character
    FROM
        json_table,
        json_each(json_table.json_data)
    WHERE
        character <> ""
        AND character NOT LIKE "%SELF%"
    ORDER BY
        character
)
SELECT
    GROUP_CONCAT(character)
FROM
    characters;

```


Grid view


Form view














1







Total rows loaded: 1

GROUP_CONCAT(character)

1 Alex,Amsterdam Vallon,Arnie Grape,Arthur Rimbaud,Billy,Brandon Darrow,Bud,Calvin Candie,Child,Cobb,Danny...

PS: