Question 1)

```
SELECT DISTINCT(category) FROM crew
ORDER BY category
LIMIT 10;
```

Grid view | Form view

Total rows loaded: 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_titleand type == 'movie' and genres like '%Action%'
order by premiered desc, primary_title asc
limit 10;
```

Grid view | Form view

Total rows loaded: 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 Raidâ Ôzu 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 (Kidô 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
         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 ;

Grid view | Form view

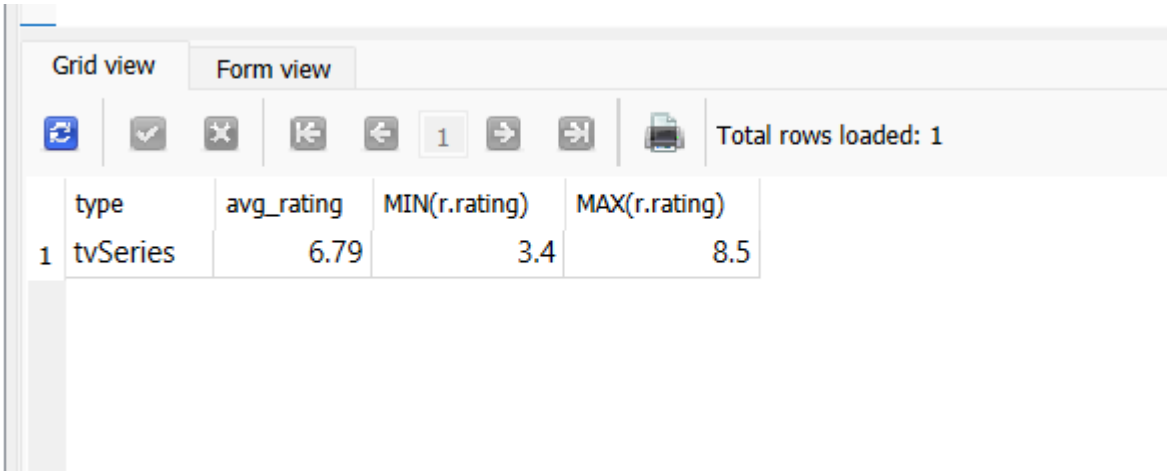
Total rows loaded: 13

	decade	num_directors
1	1900s	376
2	1910s	389
3	1920s	721
4	1930s	810
5	1940s	999
6	1950s	1084
7	1960s	1625
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
         JOIN ratings as r ON r.title_id = a.title_id
         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;



	type	avg_rating	MIN(r.rating)	MAX(r.rating)
1	tvSeries	6.79	3.4	8.5

QUESTION 6

```
WITH batman_actors AS(
SELECT DISTINCT(people.person_id) AS id, name
FROM
people
 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
 JOIN crew ON ratings.title_id == crew.title_id
 JOIN batman_actors ON crew.person_id == batman_actors.id
GROUP BY batman_actors.id
```

ORDER BY avg_rating DESC

LIMIT 10;

Grid view | Form view

Total rows loaded: 10

	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

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;

Grid view | Form view

Total rows loaded: 1

	COUNT(DISTINCT people.person_id)
1	30

Question 8

WITH rose_titles AS (SELECT DISTINCT(c.title_id)

FROM people AS P

```
 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
 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
 JOIN rose_directors AS rd ON rd.person_id = p.person_id
ORDER BY name ASC;
```

Grid view | Form view

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 10 (used chatgpt now have a idea about cte storing different datatypes

```
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...