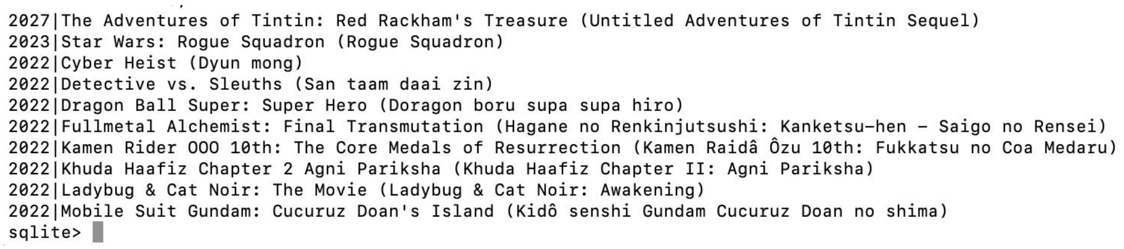
Q1 (q1_sample):

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

Q2 (q2_not_the_same_title):

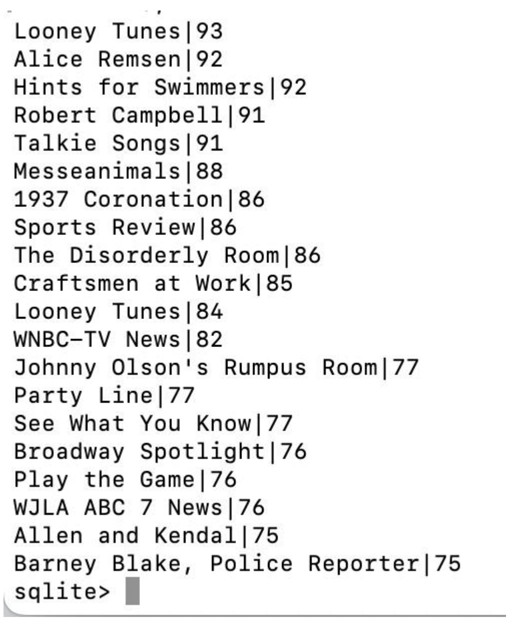
```
select premiered, primary_title || " (" || original_title || ")"
from titles
where primary_title <> original_title and type == 'movie' and genres like
'%Action%'
order by premiered desc, primary_title asc
limit 10;
```



```
2027|The Adventures of Tintin: Red Rackham's Treasure (Untitled Adventures of Tintin Sequel)
2023|Star Wars: Rogue Squadron (Rogue Squadron)
2022|Cyber Heist (Dyun mong)
2022|Detective vs. Sleuths (San taam daai zin)
2022|Dragon Ball Super: Super Hero (Doragon boru supa supa hiro)
2022|Fullmetal Alchemist: Final Transmutation (Hagane no Renkinjutsushi: Kanketsu-hen - Saigo no Rensei)
2022|Kamen Rider OOO 10th: The Core Medals of Resurrection (Kamen Raidâ Ôzu 10th: Fukkatsu no Coa Medaru)
2022|Khuda Haafiz Chapter 2 Agni Pariksha (Khuda Haafiz Chapter II: Agni Pariksha)
2022|Ladybug & Cat Noir: The Movie (Ladybug & Cat Noir: Awakening)
2022|Mobile Suit Gundam: Cucuruz Doan's Island (Kidô senshi Gundam Cucuruz Doan no shima)
sqlite>
```

Q3 (q3_longest_running_tv):

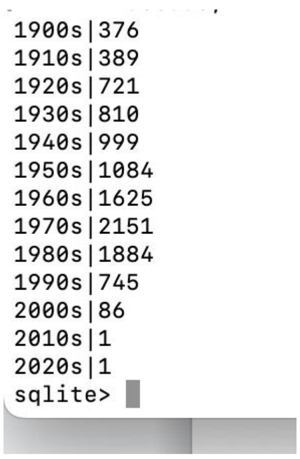
```
SELECT PRIMARY_TITLE, CASE
WHEN ENDED IS NOT NULL THEN ENDED - PREMIERED
ELSE 2023 - PREMIERED
END AS RUNTIME
FROM TITLES
WHERE TYPE = 'TVSERIES' AND TYPE = PREMIERED NOT NULL
ORDER BY RUNTIME DESC, PRIMARY_TITLE ASC
LIMIT 20;
```



```
Looney Tunes|93
Alice Remsen|92
Hints for Swimmers|92
Robert Campbell|91
Talkie Songs|91
Messeanimals|88
1937 Coronation|86
Sports Review|86
The Disorderly Room|86
Craftsmen at Work|85
Looney Tunes|84
WNBC-TV News|82
Johnny Olson's Rumpus Room|77
Party Line|77
See What You Know|77
Broadway Spotlight|76
Play the Game|76
WJLA ABC 7 News|76
Allen and Kendal|75
Barney Blake, Police Reporter|75
sqlite>
```

Q4 (q4_directors_in_each_decade):

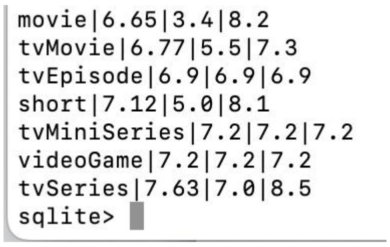
```
select cast(born / 10 * 10 AS TEXT) || 's' as decade,
count (distinct(people.person_id)) as num_directors
from people
inner join crew on people.person_id = crew.person_id
where born is not null and category == 'director' and born >= 1900
group by decade
order by decade;
```



```
1900s|376
1910s|389
1920s|721
1930s|810
1940s|999
1950s|1084
1960s|1625
1970s|2151
1980s|1884
1990s|745
2000s|86
2010s|1
2020s|1
sqlite>
```

Q5 (q5_german_type_ratings):

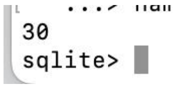
```
select t.type, round(avg(r.rating), 2) as avg_rating, min(r.rating), max(r.rating)
from akas as a
inner join ratings as r on r.title_id = a.title_id
inner join titles as t on t.title_id = a.title_id
where a.language = 'de' and (a.types = 'imdbDisplay' or a.types = 'original')
group by t.type
order by avg_rating;
```



```
movie|6.65|3.4|8.2
tvMovie|6.77|5.5|7.3
tvEpisode|6.9|6.9|6.9
short|7.12|5.0|8.1
tvMiniSeries|7.2|7.2|7.2
videoGame|7.2|7.2|7.2
tvSeries|7.63|7.0|8.5
sqlite>
```

Q7 (q7_born_with_prestige):

```
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;
```



```
   ...> nam
30
sqlite>
```