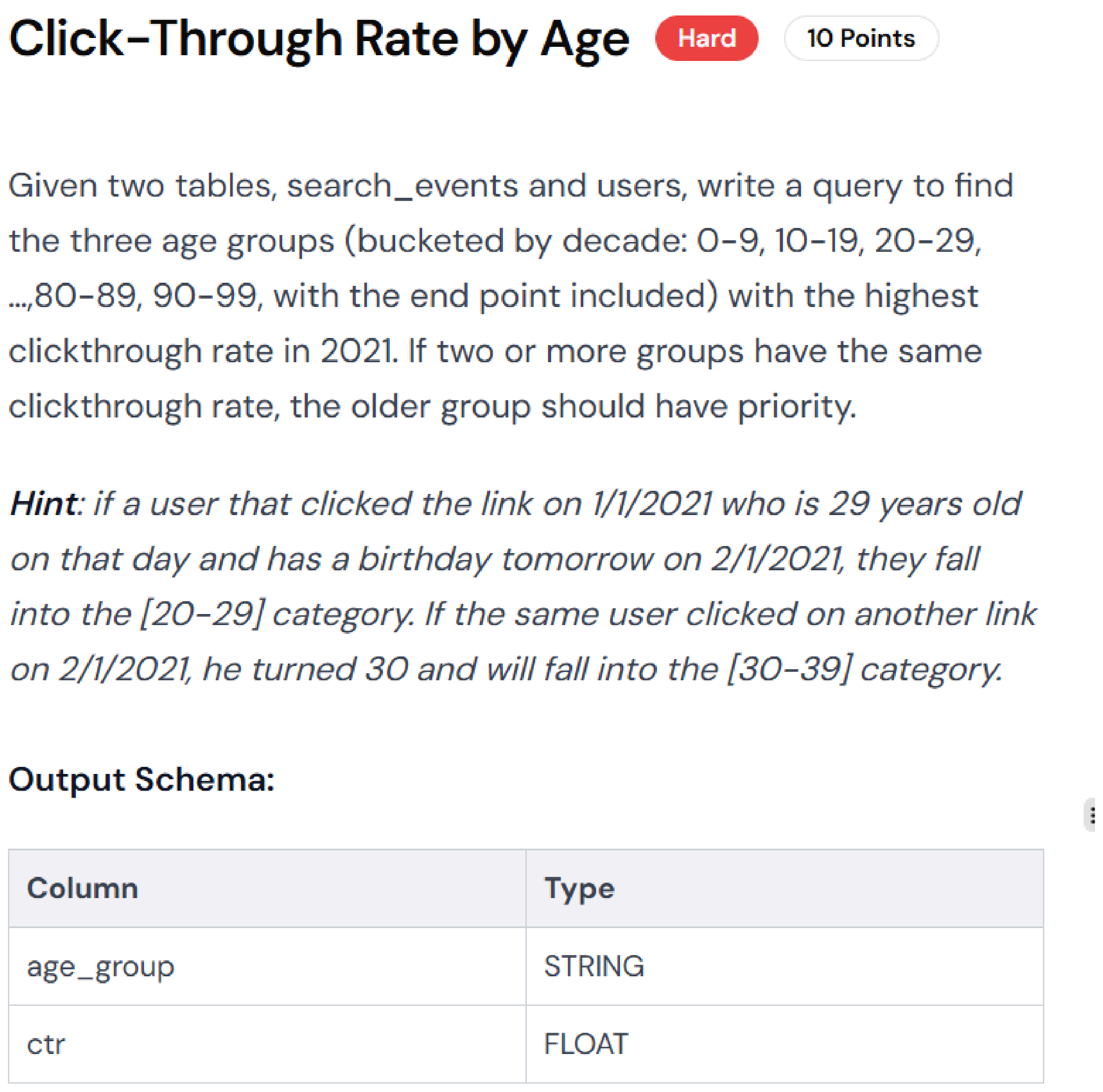
QUESTION



Click-Through Rate by Age

Hard

10 Points

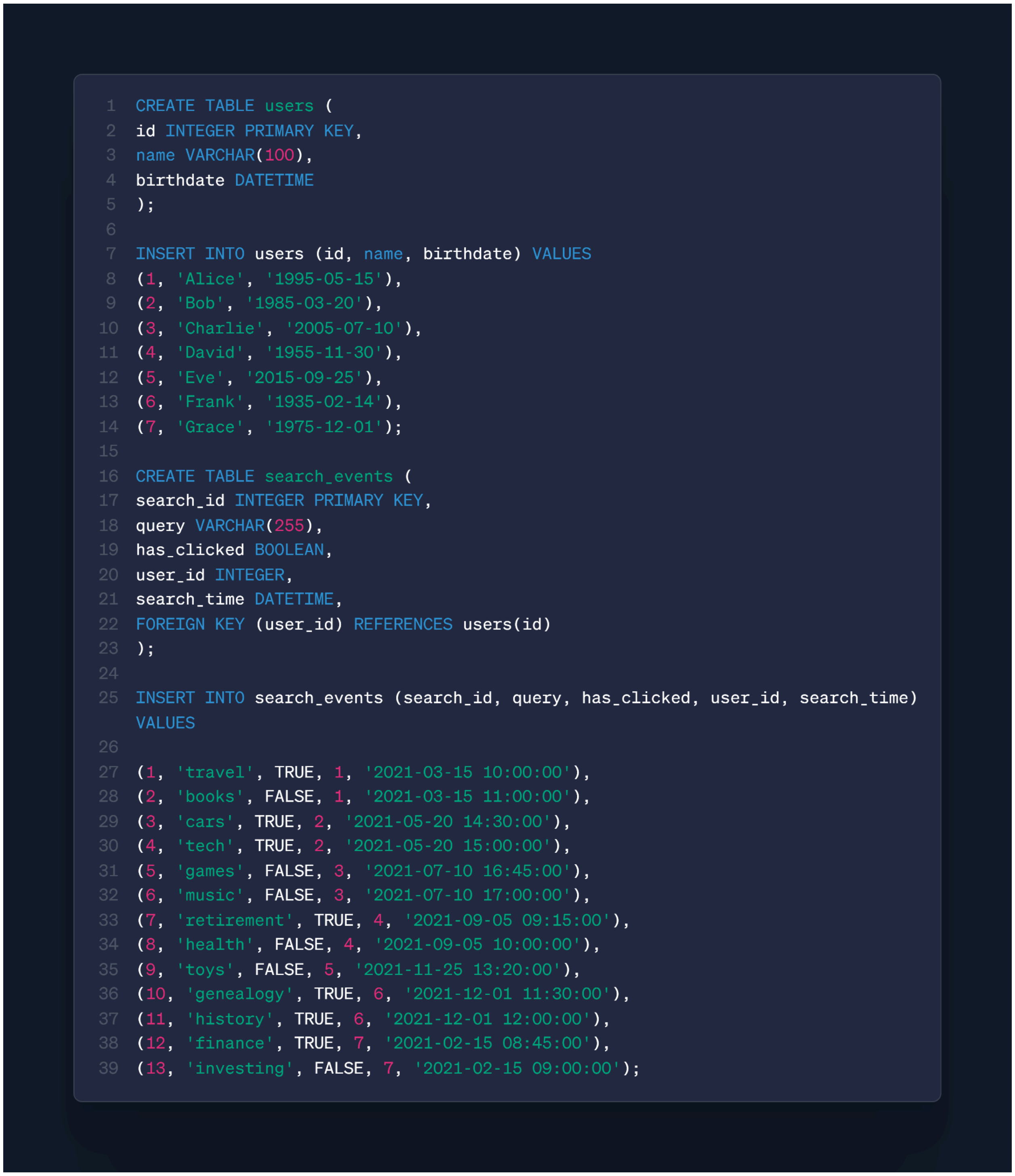
Given two tables, search_events and users, write a query to find the three age groups (bucketed by decade: 0-9, 10-19, 20-29, ...,80-89, 90-99, with the end point included) with the highest clickthrough rate in 2021. If two or more groups have the same clickthrough rate, the older group should have priority.

***Hint**: if a user that clicked the link on 1/1/2021 who is 29 years old on that day and has a birthday tomorrow on 2/1/2021, they fall into the [20-29] category. If the same user clicked on another link on 2/1/2021, he turned 30 and will fall into the [30-39] category.*

Output Schema:

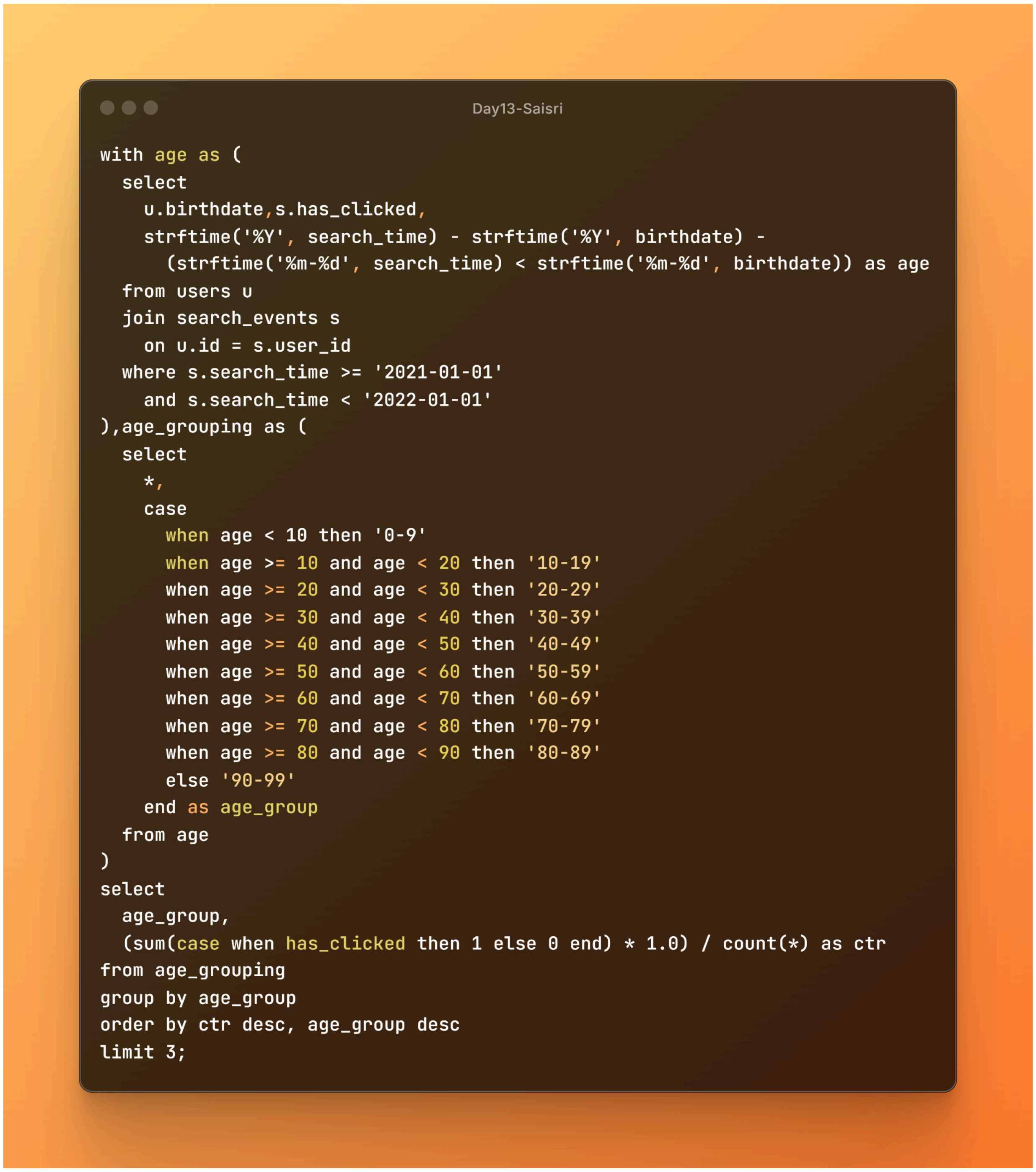
Column	Type
age_group	STRING
ctr	FLOAT

TABLE SCHEMA



```sql
CREATE TABLE users (
id INTEGER PRIMARY KEY,
name VARCHAR(100),
birthdate DATETIME
);

INSERT INTO users (id, name, birthdate) VALUES
(1, 'Alice', '1995-05-15'),
(2, 'Bob', '1985-03-20'),
(3, 'Charlie', '2005-07-10'),
(4, 'David', '1955-11-30'),
(5, 'Eve', '2015-09-25'),
(6, 'Frank', '1935-02-14'),
(7, 'Grace', '1975-12-01');

CREATE TABLE search_events (
search_id INTEGER PRIMARY KEY,
query VARCHAR(255),
has_clicked BOOLEAN,
user_id INTEGER,
search_time DATETIME,
FOREIGN KEY (user_id) REFERENCES users(id)
);

INSERT INTO search_events (search_id, query, has_clicked, user_id, search_time) VALUES

(1, 'travel', TRUE, 1, '2021-03-15 10:00:00'),
(2, 'books', FALSE, 1, '2021-03-15 11:00:00'),
(3, 'cars', TRUE, 2, '2021-05-20 14:30:00'),
(4, 'tech', TRUE, 2, '2021-05-20 15:00:00'),
(5, 'games', FALSE, 3, '2021-07-10 16:45:00'),
(6, 'music', FALSE, 3, '2021-07-10 17:00:00'),
(7, 'retirement', TRUE, 4, '2021-09-05 09:15:00'),
(8, 'health', FALSE, 4, '2021-09-05 10:00:00'),
(9, 'toys', FALSE, 5, '2021-11-25 13:20:00'),
(10, 'genealogy', TRUE, 6, '2021-12-01 11:30:00'),
(11, 'history', TRUE, 6, '2021-12-01 12:00:00'),
(12, 'finance', TRUE, 7, '2021-02-15 08:45:00'),
(13, 'investing', FALSE, 7, '2021-02-15 09:00:00');
```

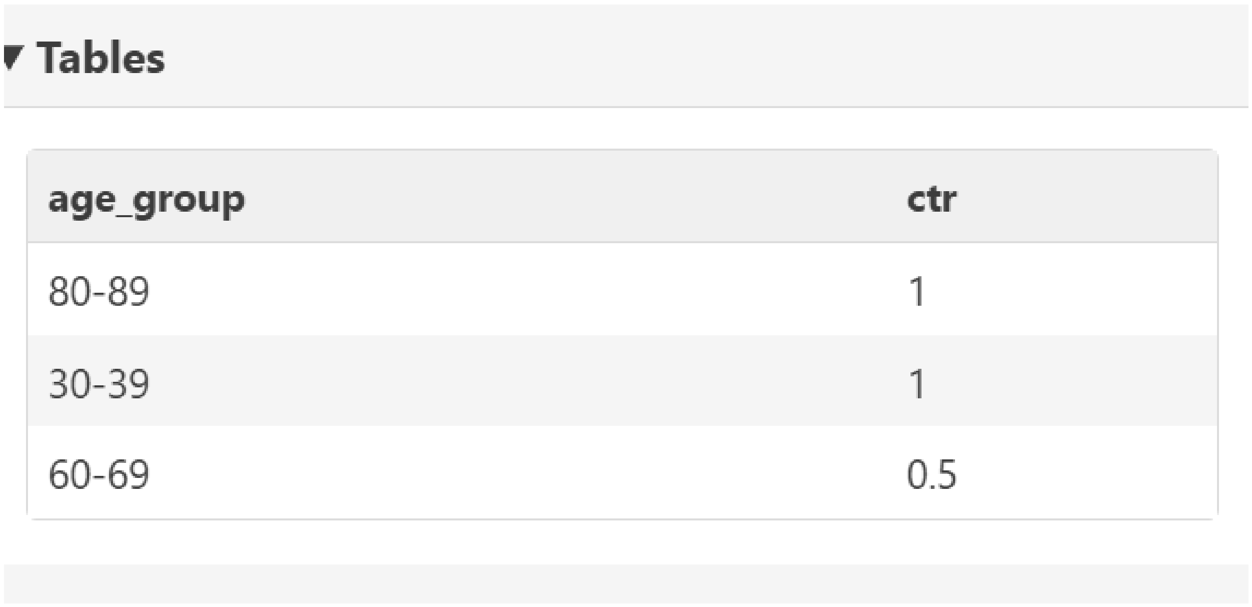
SOLUTION



Day13-Saisri

```
with age as (
  select
    u.birthdate,s.has_clicked,
    strftime('%Y', search_time) - strftime('%Y', birthdate) -
      (strftime('%m-%d', search_time) < strftime('%m-%d', birthdate)) as age
  from users u
  join search_events s
    on u.id = s.user_id
  where s.search_time >= '2021-01-01'
    and s.search_time < '2022-01-01'
),age_grouping as (
  select
    *,
    case
      when age < 10 then '0-9'
      when age >= 10 and age < 20 then '10-19'
      when age >= 20 and age < 30 then '20-29'
      when age >= 30 and age < 40 then '30-39'
      when age >= 40 and age < 50 then '40-49'
      when age >= 50 and age < 60 then '50-59'
      when age >= 60 and age < 70 then '60-69'
      when age >= 70 and age < 80 then '70-79'
      when age >= 80 and age < 90 then '80-89'
      else '90-99'
    end as age_group
  from age
)
select
  age_group,
  (sum(case when has_clicked then 1 else 0 end) * 1.0) / count(*) as ctr
from age_grouping
group by age_group
order by ctr desc, age_group desc
limit 3;
```

OUTPUT



▾ Tables

age_group	ctr
80-89	1
30-39	1
60-69	0.5

My Thought Process:

I started by joining the user and search events data, then focused on calculating each user's age at the time of the search, not just based on birth year. From there, I grouped ages into decades like '20-29', '30-39', and so on.

Next, I counted how many events had clicks vs. total searches in each group and calculated CTR.

Business Impact:

This kind of insight can be incredibly useful for UX personalization on search platforms. For example, if older users in their 60s or 70s show the highest CTRs, a company might decide to simplify the UI for that age group, highlight trending topics relevant to them, or adjust onboarding tutorials. It helps improve engagement by tailoring the experience based on who's actually interacting with the platform the most.