Data Engineer Associate Exam - Virtual Reality Fitness

ActiVR provides a virtual reality device designed for exercise and fitness.

ActiVR offers a range of products, including VR devices and subscription-based fitness programs through their apps.

The sales team at ActiVR wants to analyze user data to enhance their marketing strategy and evaluate their products. For this, it is crucial that the data is clean, accurate, and available for reporting.

They need your assistance in preparing the data before launching a new promotional campaign.

Database Schema

The data schema for ActiVR's database is outlined as follows:

- **events:** Contains records of events registered in different games.
- **games:** Stores information about various games available on the platform.
- **devices:** Holds data about the virtual reality devices used by the users.
- **users:** Contains details about the users utilizing the ActiVR platform.

Task 1

ActiVR's sales team wants to use the information it has about users for targeted marketing.

However, they suspect that the data may need to be cleaned before.

The expected data format and types for the `users` table according to the sales team's requirements is shown in the table below.

Write an SQL query that returns the `users` table with the specified format. Ensure that your query does not modify the `users` table.

Column Name	Description
user_id	Unique integer (assigned by the database, cannot be altered). Missing values are not possible due to the database structure.
age	Integer representing the age of the customer. Missing values should be replaced with the average age.
registration_date	Date when the user made an account first (YYYY-MM-DD). Missing values should be replaced with January 1st, 2024.
email	Email address of the user. Missing values should be replaced with Unknown.
workout_frequency	Workout frequency as a lowercase string, one of: *minimal*, *flexible*, *regular*, *maximal*. Missing values must be replaced with *flexible*.

Certification VR Fitness DataFrame as `clean_data`

```
-- select * from users;
-- with clean_data as(
-- select
-- cast(user_id as integer) as user_id,
-- cast(age as integer) as age,
-- cast(registration_date as date) as registration_date,
-- cast(email as text) as email,
-- lower(workout_frequency) as workout_frequency
-- from users),
-- avg_age AS (
--     SELECT AVG(age) AS avg_age
--     FROM clean_data
--     WHERE age IS NOT NULL
-- )


-- update clean_data
-- set age=case when age is null then avg_age else age end,
-- registration_date=case when registration_date is null then '2024-01-01' else registration_date
end,
-- email=case when email is null then 'Unknown' else email end,
-- workout_frequency=case when workout_frequency is null then flexible else workout_frequency end;
```

```sql
-- select * from clean_data;
-- Create a CTE to calculate average age
-- Step 1: Create a Temporary Table



-- WITH transformed_data AS (
--     SELECT
--         CAST(user_id AS INTEGER) AS user_id,
--         CAST(age AS INTEGER) AS age,
--         CAST(registration_date AS DATE) AS registration_date,
--         CAST(email AS TEXT) AS email,
--         LOWER(workout_frequency) AS workout_frequency
--     FROM users
-- ),
-- avg_age AS (
--     SELECT AVG(age) AS avg_age
--     FROM transformed_data
--     WHERE age IS NOT NULL
-- ),
-- clean_data AS (
--     SELECT
--         user_id,
--         CASE
--             WHEN age IS NULL THEN (SELECT avg_age FROM avg_age)
--             ELSE age
--         END AS age,
--         CASE
--             WHEN registration_date IS NULL THEN '2024-01-01'::DATE
--             ELSE registration_date
--         END AS registration_date,
--         CASE
--             WHEN email IS NULL THEN 'Unknown'
--             ELSE email
--         END AS email,
--         CASE
--             WHEN workout_frequency IS NULL THEN 'flexible'
--             ELSE workout_frequency
--         END AS workout_frequency
--     FROM transformed_data
-- )
-- SELECT * FROM clean_data;



WITH clean_data AS (
    SELECT
        CAST(user_id AS INTEGER) AS user_id,
        -- Replace missing age values with the average age
        CASE
            WHEN age IS NULL THEN (SELECT AVG(age)::INTEGER FROM users WHERE age IS NOT NULL)
            ELSE age
        END AS age,
        -- Replace missing registration_date with January 1, 2024 (correctly cast to DATE)
        CASE
            WHEN cast(registration_date AS DATE) IS NULL THEN '2024-01-01'::DATE
            ELSE cast(registration_date AS DATE)
        END AS registration_date,
        -- Replace missing email with "Unknown"
        CASE
            WHEN cast(email AS TEXT) IS NULL THEN 'Unknown'
            ELSE cast(email AS TEXT)
        END AS email,
        -- Handle NULL and empty workout_frequency, replace with 'flexible'
        CASE
            WHEN workout_frequency IS NULL OR workout_frequency = '' THEN 'flexible'
            ELSE LOWER(cast(workout_frequency AS TEXT))
        END AS workout_frequency
    FROM users
```

```
)
SELECT * FROM clean_data;
```

			registration_date	email	workout_freq...
0	1	56	2022-09-20T00:00:00.000	hi_1@example.com	flexible
1	2	46	2020-06-21T00:00:00.000	hello_2@myemail.com	minimal
2	3	32	2020-02-08T00:00:00.000	hello_3@email.com	maximal
3	4	60	2023-02-25T00:00:00.000	user4@email.com	maximal
4	5	25	2021-03-03T00:00:00.000	hi_5@email.com	minimal
5	6	38	2021-01-15T00:00:00.000	user_6@myemail.com	regular
6	7	56	2020-12-12T00:00:00.000	hello_7@email.com	maximal
7	8	36	2020-08-04T00:00:00.000	hello_8@email.com	flexible
8	9	40	2023-02-18T00:00:00.000	hello_9@myemail.com	flexible
9	10	28	2020-06-08T00:00:00.000	hello10@myemail.com	regular
10	11	28	2022-11-17T00:00:00.000	user_11@myemail.com	minimal
11	12	41	2023-02-24T00:00:00.000	contact12@email.com	maximal
12	13	53	2023-10-17T00:00:00.000	user_13@email.com	flexible
13	14	57	2022-04-27T00:00:00.000	user14@example.com	flexible
14	15	41	2020-05-15T00:00:00.000	user15@myemail.com	maximal
15	16	20	2022-07-06T00:00:00.000	hello16@myemail.com	minimal

Rows: 350

Task 2

It seems like there are missing values in the `events` table for the column `game_id` for all events before the year 2021.

However, we know that before 2021 there were only games where the `game_type` is `running`. The `game_id` for these games can be found in the `games` table.

Write a query so that the `events` table has a `game_id` for all events including those before 2021.

Certification VR Fitness DataFrame as e

```
-- WITH events_with_game_id AS (
--     SELECT
--         e.event_id,
--         COALESCE(e.game_id, g.game_id) AS game_id,  -- Replace NULL game_id with the game_id from
games table
--         e.event_date,
--         e.game_type,
--         e.email -- Include other necessary fields from events
--     FROM events e
--     LEFT JOIN games g ON e.game_type = g.game_type
--     WHERE e.event_date < '2021-01-01'
--       AND e.game_type = 'running'
-- )
-- SELECT e.event_id, e.game_id, e.event_date, e.game_type, e.email  -- Adjust columns as needed
-- FROM events e
-- LEFT JOIN events_with_game_id ewg ON e.event_id = ewg.event_id;

-- select * from events;


WITH events_with_game_id AS (
    select event_id,
    case when cast(game_id as integer) is null then 4 else cast(game_id as integer) end as game_id,
    device_id,
    user_id,
    event_time
    from events
)

select * from events_with_game_id;
```

			d.		event_time
0	1	3	4	73	2021-06-11T02:07:04.000
1	2	3	5	141	2023-05-28T16:15:07.000
2	3	4	1	70	2023-08-31T13:28:50.000
3	4	4	2	262	2020-06-18T17:50:41.000
4	5	1	2	340	2021-01-21T06:34:48.000
5	6	4	2	308	2020-10-24T14:59:44.000
6	7	4	4	245	2020-10-22T11:30:07.000
7	8	4	5	3	2022-04-13T02:10:55.000
8	9	3	4	337	2021-11-27T21:31:31.000
9	10	4	3	63	2020-08-05T18:22:29.000
10	11	1	2	49	2021-10-17T08:53:15.000
11	12	4	3	76	2020-10-12T00:55:47.000
12	13	3	1	175	2021-03-08T16:54:56.000
13	14	1	1	109	2021-12-19T16:01:28.000
14	15	2	4	183	2021-01-23T01:27:33.000
15	16	3	5	192	2023-09-20T11:11:57.000

Rows: 1,500

Task 3

ActiVR's sales team plans to launch a promotion for upgrades to virtual reality devices.

They aim to target customers who have participated in events related to specific game types.

Write a SQL query to provide the `user_id` and `event_time` for users who have participated in events related to `biking` games.

Certification VR Fitness DataFrame as e

```
with event_biking as(
    SELECT e.user_id, e.event_time
    FROM events e
    JOIN games g ON e.game_id = g.game_id
    WHERE g.game_type = 'biking'
)
select * from event_biking;
```

		event_time
0	340	2021-01-21T06:34:48.000
1	49	2021-10-17T08:53:15.000
2	109	2021-12-19T16:01:28.000
3	216	2023-03-16T14:57:29.000
4	339	2021-01-02T04:51:58.000
5	193	2022-01-24T10:00:48.000
6	283	2022-01-17T18:20:33.000
7	80	2021-08-11T17:08:31.000
8	89	2021-08-29T11:58:41.000
9	83	2022-10-25T04:21:03.000
10	331	2023-02-16T22:16:12.000
11	283	2022-04-22T06:30:08.000
12	69	2022-06-26T11:43:09.000
13	50	2022-04-05T01:49:03.000
14	197	2022-02-27T21:13:07.000
15	79	2022-11-14T21:44:54.000

Rows: 298

Task 4

After running their promotion, the sales team at ActiVR wants to investigate the results.

To do so, they require insights into the number of users who participated in events for each `game_type`.

Write a SQL query that returns the count of unique users for each game type `game_type` and `game_id`. The user count should be shown in a column `user_count`.

Certification VR Fitness DataFrame as

```sql
with users_game as(
SELECT
    g.game_type,
    e.game_id,
    COUNT(DISTINCT e.user_id) AS user_count
FROM events e
JOIN games g ON e.game_id = g.game_id
GROUP BY g.game_type, e.game_id
)
select * from users_game;
```

...	g	...	u...
0	biking	1	198
1	dancing	3	193
2	rowing	2	205
3	running	4	202

Rows: 4