

My participant table has 815 rows, and the participant_id is sequential from 1 to 815 with no gaps (this is a set of fictitious demo data). This query gives me a thousand fictitious names and addresses:

```
; WITH People AS (
    SELECT TOP (1000)
        CAST (1000 AS INT) AS Participant_ID ,
        [fileAs] ,
        [givenname] ,
        [surname] ,
        '20250101' AS [birthdate] ,
        [gender] ,
        pa.*
    FROM [Anon].[dbo].[Contact] c
    JOIN PostalAddress pa On pa.contactID = c.contactid AND pa.province='SK'
)

Select * from People
```

what i would like to do, however, is replace the static "1000" with a sequential number from 1 to 1000. I think using a windowing function and a row_number might do that. then, i'd like to save the resulting output as a new table in a different database. It will be [YouthConnectDemo].[dbo].[participants_demographics] I can run the query from either database, they are on the same server. Can you write or adjust this query to do that?

ChatGPT said:

Perfect — yes, using `ROW_NUMBER()` is exactly the right approach here. Here's how you can do what you described, step-by-step and cleanly.

✓ Full query — sequential IDs and save to another database

```
;WITH People AS (
    SELECT
        ROW_NUMBER() OVER (ORDER BY c.contactid) AS participant_id,
        c.[fileAs],
        c.[givenname],
        c.[surname],
        CAST('20250101' AS date) AS [birthdate],
        c.[gender],
        pa.*
    FROM [Anon].[dbo].[Contact] AS c
    INNER JOIN [Anon].[dbo].[PostalAddress] AS pa
        ON pa.contactID = c.contactid
        AND pa.province = 'SK'
)
SELECT *
INTO [YouthConnectDemo].[dbo].[participants_demographics]
FROM People;
```

Explanation

1. `ROW_NUMBER()`

- Replaces your hardcoded `1000` with a sequential number starting at 1.
- Ordered by `contactid` (you can change this if you prefer a different sort order).

2. `SELECT ... INTO`

- Creates a new table `[YouthConnectDemo].[dbo].[participants_demographics]` and copies all columns + data into it in one step.
- If the table already exists, you can instead use:

```
INSERT INTO [YouthConnectDemo].[dbo].[participants_demographics] (...)  
SELECT ...
```

3. Types and casting

- `'20250101'` is explicitly cast to a `date` to avoid string-type inference.
- If you want that column to remain as a string, just remove the `CAST()`.