How To:
Create a sequence of 1000 integers in SQL (5 methods)

Method 1: Recursive CTE

```sql
1   WITH numbers
2   AS
3   (
4   SELECT 0 AS num
5   UNION ALL
6   SELECT num + 1
7   FROM numbers
8   WHERE num < 999
9   )
10  SELECT num
11  FROM numbers
12  ORDER BY num
13  OPTION (MAXRECURSION 1000);
```

In PostgreSQL and MySQL, we put the word "RECURSIVE" here

This syntax is SQL Server only and is needed if you want more than 100 iterations

Method 2: OPENJSON

```
1   WITH digits AS
2   (
3   SELECT "value" AS num
4   FROM OPENJSON(N'[0,1,2,3,4,5,6,7,8,9]')
5   )
6   SELECT a.num * 100 + b.num * 10 + c.num AS num
7   FROM digits a, digits b, digits c
8   ORDER BY num;
```

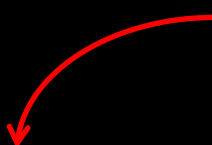We pass a list of 10 digits into the OPENJSON function as a JSON array

We use an implicit CROSS JOIN by separating three copies of the CTE with commas

flexyourdata.com | youtube.com/@flexyourdata | linkedin.com/in/owenhprice
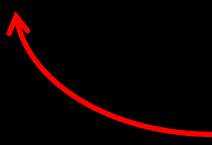
Method 3: SELECT FROM (VALUES(...

⚠️ Not supported by MySQL

```sql
1  WITH digits
2  AS
3  (
4  SELECT num
5  FROM (VALUES (0),(1),(2),(3),(4),(5),(6),(7),(8),(9)) t(num)
6  )
7  SELECT a.num * 100 + b.num * 10 + c.num AS num
8  FROM digits a, digits b, digits c
9  ORDER BY num;
```

We construct a table by passing single-column tuples of the digits into VALUES

Again, we use an implicit CROSS JOIN by separating three copies of the CTE with commas
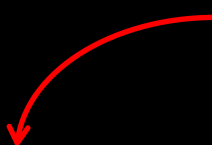
Method 4: UNNEST(ARRAY[...])

⚠️ PostgreSQL only

```sql
1  WITH digits AS
2  (
3  SELECT num
4  FROM unnest(ARRAY[0,1,2,3,4,5,6,7,8,9]) as t(num)
5  )
6  SELECT a.num * 100 + b.num * 10 + c.num AS num
7  FROM digits a, digits b, digits c;
```

We pass an array type of digits into the UNNEST function

Again, we use an implicit CROSS JOIN by separating three copies of the CTE with commas

flexyourdata.com | youtube.com/@flexyourdata | linkedin.com/in/owenhprice

Method 5: GENERATE_SERIES

⚠ PostgreSQL now, coming soon to SQL Server 2022

There's not much to say about this apart from it's incredibly easy

```
1
2    select generate_series(0,999);
3
```

We can use the function in the SELECT clause or in the FROM clause