IDENTITY Columns - Generate IDs - PostgreSQL to MySQL Migration

IDENTITY column property allows you to automatically generate sequential integer numbers (IDs) for a column. IDENTITY columns are available since PostgreSQL 10.

In MySQL you can use AUTO_INCREMENT column property. Note that you can use the AUTO_INCREMENT table property to define the start value, but you cannot specify the increment step, it is always 1. See AUTO_INCREMENT for more details.

PostgreSQL - Quick Example:

```
-- Define a table with an IDENTITY column (id starts at 10)
CREATE TABLE cities
(
  id   INTEGER GENERATED ALWAYS AS IDENTITY (START WITH 10),
  name VARCHAR(30)
);
```

MySQL - Quick Example:

```
-- Define a table with an IDENTITY column (id starts at 10)
CREATE TABLE cities
(
  id   INTEGER AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(30)
)
AUTO_INCREMENT = 10;
```

PostgreSQL - Syntax:

```
-- Simplified syntax
GENERATED {ALWAYS | BY DEFAULT} AS IDENTITY [(START WITH start INCREMENT BY inc ...)]
```

Conversion Overview:

	PostgreSQL	**MySQL**
Start Value	Specified using START WITH option, default is 1	Specified as a table property, default is 1
Increment	Specified using INCREMENT BY option, default is 1	Always 1
How to Generate IDs	Omit the IDENTITY column in INSERT	Omit the AUTO_INCREMENT column in INSERT, or specify NULL or 0
Explicit ID Insert	No allowed if GENERATED ALWAYS is specified. Either ALWAYS or BY DEFAULT must be specified	Allowed

Restrictions		Only one AUTO_INCREMENT column per table
		Primary key or unique must be specified
		DEFAULT is not allowed
		Data type of column must be an integer. DECIMAL and NUMERIC are not allowed. DOUBLE and FLOAT are allowed but deprecated.
Last ID		LAST_INSERT_ID returns the last value inserted in the current session
		LAST_INSERT_ID returns ID for the first successfully inserted row in multi-row INSERT
Gaps		If a value larger than the current max ID value is explicitly inserted, then new IDs with start from this value + 1
Restart (Reset)		ALTER TABLE *name* AUTO_INCREMENT = *new_start_value*;

Consider the following sample data:

PostgreSQL:

```
CREATE TABLE cities
(
  id   INTEGER GENERATED ALWAYS AS IDENTITY (START WITH 10),
  name VARCHAR(30)
);

-- Some test data
INSERT INTO cities(name) VALUES ('Malaga');
INSERT INTO cities(name) VALUES ('Seville');
INSERT INTO cities(name) VALUES ('Madrid');

-- See the inserted data
SELECT * FROM cities;
```

Result:

id	name
10	Malaga
11	Seville
12	Madrid