MySQL - AUTO_INCREMENT - Generate IDs (Identity, Sequence)

AUTO_INCREMENT option allows you to automatically generate unique integer numbers (IDs, identity, sequence) for a column.

Quick Example:

```
-- Define a table with an auto-increment column (id starts at 100)
CREATE TABLE airlines
(
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(90)
)
AUTO_INCREMENT = 100;

-- Insert a row, ID will be automatically generated
INSERT INTO airlines (name) VALUES ('United Airlines');

-- Get generated ID
SELECT LAST_INSERT_ID();
-- Returns: 100
```

Overview:

Start Value	✓	Default is 1
Increment	✗	Always 1
How to Generate IDs	Omit the AUTO_INCREMENT column in INSERT, or specify NULL or 0	
Explicit ID Insert	✓	
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 *table_name* AUTO_INCREMENT = *new_start_value*;	

MySQL AUTO_INCREMENT Details

To generate a ID value, you can omit the auto-increment column in INSERT statement, or specify NULL or 0 value explicitly:

```
-- Omit auto-increment column
INSERT INTO airlines (name) VALUES ('Delta');

-- Specify NULL or 0
INSERT INTO airlines VALUES (NULL, 'Southwest');
INSERT INTO airlines VALUES (0, 'American Airlines');
```

Make a Gap

You can insert an ID value explicitly, then MySQL will generate new IDs starting from it adding 1:

```
INSERT INTO airlines VALUES (200, 'Lufthansa');

INSERT INTO airlines (name) VALUES ('British Airways');   -- id 201 is assigned
```

You can still insert *inside* the gap using ID less than the current maximum ID, but this does not affect ID that will be used for other rows:

```
INSERT INTO airlines VALUES (150, 'Air France');   -- id 150 inserted

INSERT INTO airlines (name) VALUES ('KLM');   -- id 202 is assigned
```

Table content:

id	**name**
100	United Airlines
101	Delta
102	Southwest
103	American Airlines
150	Air France
200	Lufthansa
201	British Airways
202	KLM

Getting Generated ID

LAST_INSERT_ID function returns ID of the *first successfully* inserted row. For example, in a multi-row INSERT:

```
INSERT IGNORE INTO airlines VALUES
```

```
    (150, 'North Air'),    -- this row will be skipped as ID 150 already exists, and IGNORE option
used
    (0, 'Emirates'),       -- id 203 is assigned
    (0, 'Qantas');         -- id 204

  SELECT LAST_INSERT_ID();
  -- Returns: 203
```

Restart ID

You cannot reset the auto-increment counter to the start value less or equal than the current maximum ID:

```
  ALTER TABLE airlines AUTO_INCREMENT = 1;

  INSERT INTO airlines (name) VALUES ('US Airways');   -- id 205 is assigned
```

After you have deleted all rows, the counter is **not** automatically reset to the start value:

```
  DELETE FROM airlines;

  INSERT INTO airlines (name) VALUES ('United');   -- id 206 is assigned
```

You can restart the auto-increment to 1 if there are no rows in a table:

```
  DELETE FROM airlines;

  ALTER TABLE airlines AUTO_INCREMENT = 1;

  INSERT INTO airlines (name) VALUES ('United');   -- id 1 is assigned
```

MySQL AUTO_INCREMENT in Other Databases

Auto-increment columns in other databases:

Oracle:

Auto-increment or Identity	✗	Can be emulated using sequence and trigger

SQL Server:

IDENTITY(start, increment)	✓	Increment can be specified

PostgreSQL:

SERIAL Data Type			
Start Value	✗	Always 1	ALTER SEQUENCE to change
Increment	✗	Always 1	

Generate ID	NULL and 0 do *not* force ID generation
Last ID	LASTVAL(), CURRVAL('seq_name') and INSERT RETURNING

MySQL AUTO_INCREMENT Conversion to Other Databases

Converting MySQL AUTO_INCREMENT:

Oracle:

Oracle does not support AUTO_INCREMENT (IDENTITY) property on a column, but this functionality can be implemented using a sequence and a trigger:

```
CREATE TABLE airlines
(
  id NUMBER(10,0) PRIMARY KEY,
  name VARCHAR2(90)
);

CREATE SEQUENCE airlines_seq START WITH 100 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER airlines_seq_tr
 BEFORE INSERT ON airlines FOR EACH ROW
 WHEN (NEW.id IS NULL OR NEW.id = 0)
BEGIN
 SELECT airlines_seq.NEXTVAL INTO :NEW.id FROM dual;
END;
/
```

Note that a trigger is required as Oracle does not allow using NEXTVAL in DEFAULT clause for a column.

SQL Server:

SQL Server supports IDENTITY property and allows you to specify the increment:

```
CREATE TABLE airlines
(
  id INT IDENTITY(100, 1) PRIMARY KEY,
  name VARCHAR(90)
);
```

PostgreSQL:

PostgreSQL supports SERIAL data type that allows you to automatically generate IDs. Although SERIAL does not provide options to set the initial and increment values, you can modify the underlying sequence object:

```
CREATE TABLE airlines
(
  id SERIAL PRIMARY KEY,
  name VARCHAR(90)
);
```

```
ALTER SEQUENCE airlines_id_seq RESTART WITH 100;
```