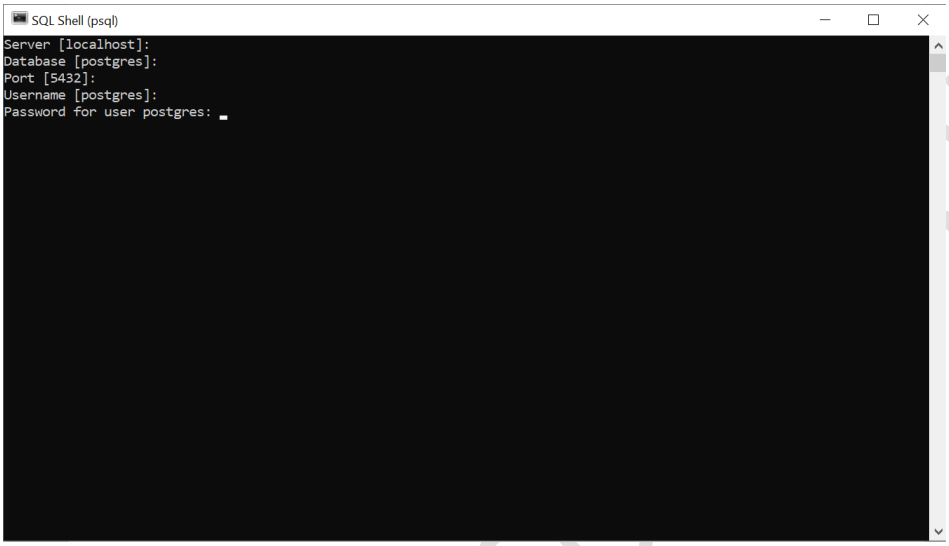
Table of Contents

PostgreSQL Practice Using SQL Shell

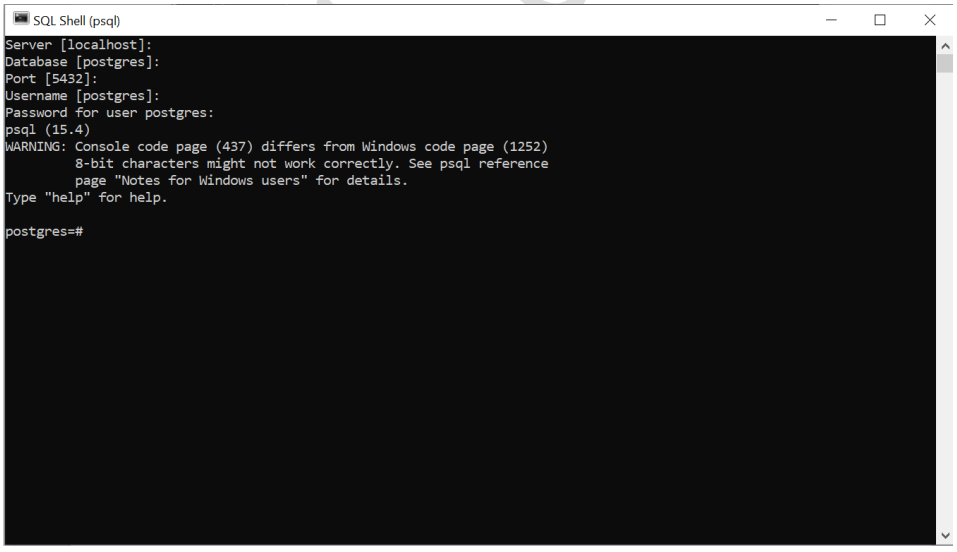
Basic of Installation

Staring

To start SQL Shell, at first install PostgreSQL from official website. Then find it in start menu or search in search bar. Next, you will find it like this terminal. Hit enter for all, but enter your password that you have set for installation.



Then you will find it like this.

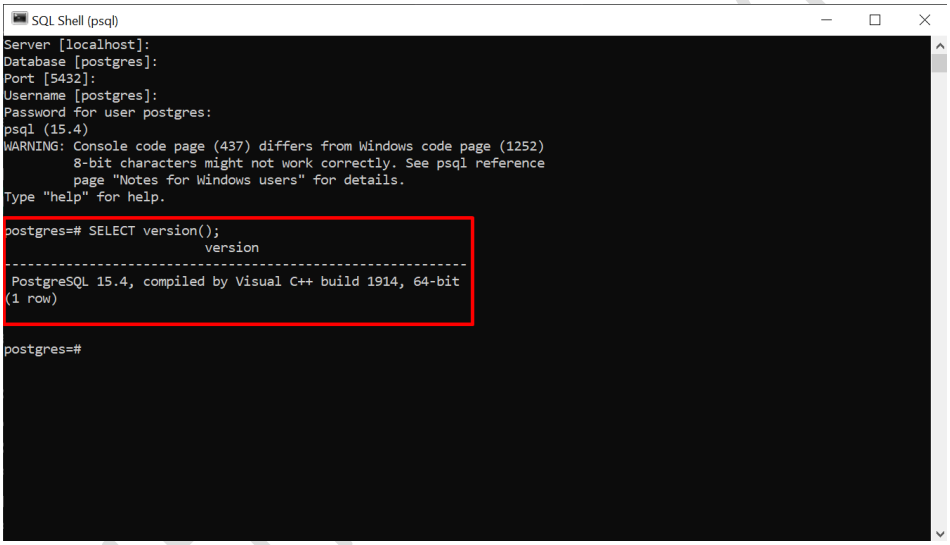


Check Version

After entering password, you will see the current version of PostgreSQL.

Or you can execute this command to see current version.

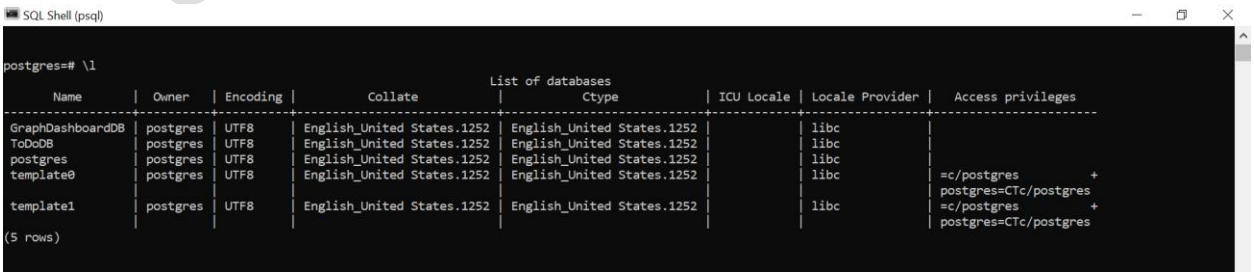
```
SELECT version();
```



List of Databases

To see list of database, you can run this command.

```
\l
```



Create Database

There are many ways to create database. We use simple method.

To create DB

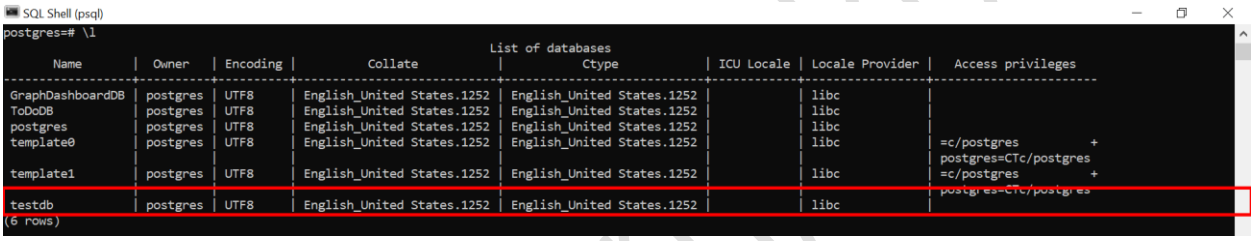
The basic syntax of CREATE DATABASE statement is as follows –

```
CREATE DATABASE dbname;
```

where *dbname* is the name of a database to create. Our database name is TestDB.

To see list of database, you can run this command.

```
\l
```



Create Table

To create table

To create table `CREATE TABLE` command is needed, then table name and its field name, data type.

```
 CREATE TABLE cars (
  brand VARCHAR(255),
  model VARCHAR(255),
  year INT
);
```

When you execute the above statement, an empty table named cars will be created, and the SQL Shell application will return the following `CREATE TABLE`:

To see all data of table

You can "display" the empty table you just created with another SQL statement:

```
SELECT * FROM cars;
```

Insert Data

To insert data in table

To insert data into a table in PostgreSQL, we use the `INSERT INTO` statement.

The following SQL statement will insert one row of data into the `cars` table.

```
INSERT INTO cars (brand, model, year)
VALUES ('Ford', 'Mustang', 1964);
```

The SQL Shell application will return the following:

INSERT 0 1

Which means that `1` row was inserted.

Note:
As you can see in the SQL statement above, string values must be written with apostrophes.
Numeric values can be written without apostrophes, but you can include them if you want.

To check the result we can display the table with this SQL statement:

```
SELECT * FROM cars;
```

Insert Multiple Rows

To insert multiple rows of data, we use the same `INSERT INTO` statement, but with multiple values:

```
INSERT INTO cars (brand, model, year)
VALUES
  ('Volvo', 'p1800', 1968),
  ('BMW', 'M1', 1978),
  ('Toyota', 'Celica', 1975);
```

Select/Fetch Data

To retrieve data from a database, we use the `SELECT` statement.

Specify Columns

By specifying the column names, we can choose which columns to select:

```
SELECT brand, year FROM cars;
```

Return ALL Columns

Specify a `*` instead of the column names to select *all* columns:

```
SELECT * FROM cars;
```

Add Column

The ALTER TABLE Statement

To add a column to an existing table, we have to use the `ALTER TABLE` statement.

The `ALTER TABLE` statement is used to add, delete, or modify columns in an existing table.

The `ALTER TABLE` statement is also used to add and drop various constraints on an existing table.

Add Column

When adding columns we must also specify the data type of the column. Our `color` column will be a string, and we specify string types with the `VARCHAR` keyword. we also want to restrict the number of characters to 255:

```
ALTER TABLE cars
ADD color VARCHAR(255);
```

The SQL Shell application will return the following:

```
ALTER TABLE
```

To check the result we can display the table with this SQL statement:

```
SELECT * FROM cars;
```

As you can see, the `cars` table now has a `color` column. The new column is empty.

UPDATE

The `UPDATE` statement is used to modify the value(s) in existing records in a table.

Set the color of the Volvo to 'red':

```
UPDATE cars
SET color = 'red'
WHERE brand = 'Volvo';
```

WE see that SQL Shell return UPDATE 1, which means that `1` row was affected by the `UPDATE` statement.

Note:

1. Be careful with the `WHERE` clause, in the example above ALL rows where brand = 'Volvo' gets updated.
2. Be careful when updating records. If you omit the `WHERE` clause, ALL records will be updated!

To check the result we can display the table with this SQL statement:

```
SELECT * FROM cars;
```

Update Multiple Columns

To update more than one column, separate the name/value pairs with a comma `,`:

Update color and year for the Toyota

```
UPDATE cars
SET color = 'white', year = 1970
WHERE brand = 'Toyota';
```

To check the result we can display the table with this SQL statement:

```
SELECT * FROM cars;
```

Alter Column

The ALTER TABLE Statement

To change the data type, or the size of a table column we have to use the `ALTER TABLE` statement.

The `ALTER TABLE` statement is used to add, delete, or modify columns in an existing table.

The `ALTER TABLE` statement is also used to add and drop various constraints on an existing table.

Alter Column

We want to change the data type of the `year` column of the `cars` table from `INT` to `VARCAHR(4)`.

To modify a column, use the `ALTER COLUMN` statement and the `TYPE` keyword followed by the new data type:

```
ALTER TABLE cars
ALTER COLUMN year TYPE VARCHAR(4);
```

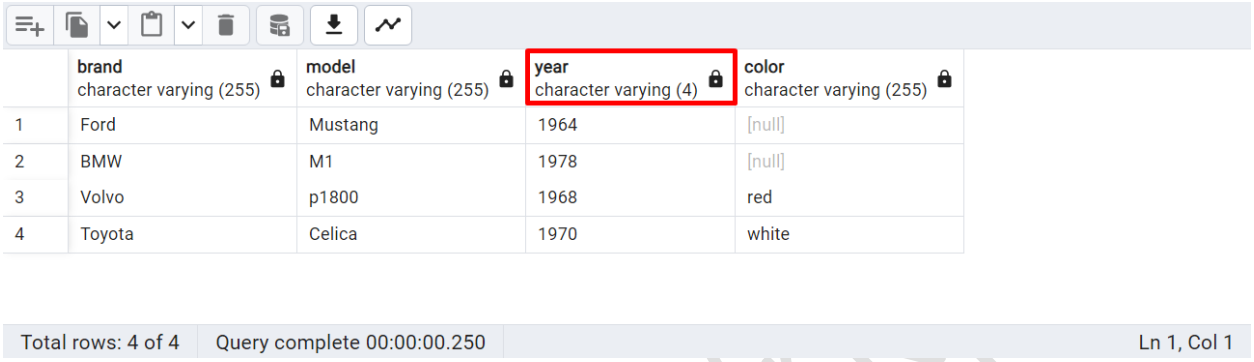
SQL Shell will return `ALTER TABLE`.

Note: Some data types cannot be converted if the column has value. E.g. numbers can always be converted to text, but text cannot always be converted to numbers.

You can see how many table are available in your databases. Just run below command in your SQL Shell.

```
\dt
```

You will find it in pgAdmin 4 that shows year data type is changed.



	brand character varying (255)	model character varying (255)	year character varying (4)	color character varying (255)
1	Ford	Mustang	1964	[null]
2	BMW	M1	1978	[null]
3	Volvo	p1800	1968	red
4	Toyota	Celica	1970	white

Total rows: 4 of 4 | Query complete 00:00:00.250 | Ln 1, Col 1

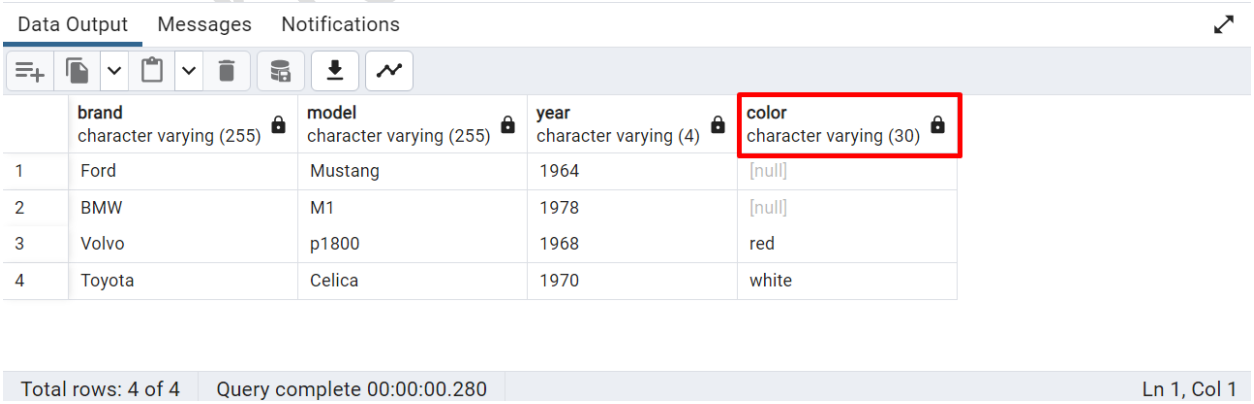
Change Maximum Allowed Characters

We also want to change the maximum number of characters allowed in the `color` column of the `cars` table.

Change the `color` column from `VARCHAR(255)` to `VARCHAR(30)`:

```
ALTER TABLE cars
ALTER COLUMN color TYPE VARCHAR(30);
```

SQL Shell also return `ALTER TABLE`. Your changed is available in pgAdmin 4.



Data Output | Messages | Notifications

	brand character varying (255)	model character varying (255)	year character varying (4)	color character varying (30)
1	Ford	Mustang	1964	[null]
2	BMW	M1	1978	[null]
3	Volvo	p1800	1968	red
4	Toyota	Celica	1970	white

Total rows: 4 of 4 | Query complete 00:00:00.280 | Ln 1, Col 1

Drop Column

The ALTER TABLE Statement

To remove a column from a table, we have to use the `ALTER TABLE` statement.

The `ALTER TABLE` statement is used to add, delete, or modify columns in an existing table.

The `ALTER TABLE` statement is also used to add and drop various constraints on an existing table.

Drop Column

We want to remove the column named `color` from the `cars` table.

To remove a column, use the `DROP COLUMN` statement:

```
ALTER TABLE cars
DROP COLUMN color;
```

We will not see color column in our table anymore. Just run this command.

```
SELECT * FROM cars;
```

Delete

The DELETE Statement

The `DELETE` statement is used to delete existing records in a table.

Note: Be careful when deleting records in a table! Notice the `WHERE` clause in the `DELETE` statement. The `WHERE` clause specifies which record(s) should be deleted.

If you omit the `WHERE` clause, **all records in the table will be deleted**!.

To to delete the record(s) where brand is 'Volvo', use this statement:

```
DELETE FROM cars
WHERE brand = 'Volvo';
```

DELETE 1, which means that `1` row was deleted.

If you run below command, you will see that deleted row is not shown.

```
SELECT * FROM cars;
```

Delete All Records

It is possible to delete all rows in a table without deleting the table. This means that the table structure, attributes, and indexes will be intact.

The following SQL statement deletes all rows in the `cars` table, without deleting the table:

```
DELETE FROM cars;
```

We will see DELETE 3 SQL shell, which means that `3` row was deleted.

Truncate Table

Because we omit the `WHERE` clause in the `DELETE` statement above, all records will be deleted from the cars table.

The same would have been achieved by using the `TRUNCATE TABLE` statement:

Delete all records in the `cars` table:

```
TRUNCATE TABLE cars;
```

It also returns TRUNCATE TABLE.

If we run this statement, we will see

```
SELECT * FROM cars;
```

Drop Table

The DROP TABLE Statement

The `DROP TABLE` statement is used to drop an existing table in a database.

Note: Be careful before dropping a table. Deleting a table will result in loss of all information stored in the table!

The following SQL statement drops the existing table `cars`:

```
DROP TABLE cars;
```

SQL Shell return `DROP TABLE`, that means cars table is deleted.

If we try to run `SELECT * FROM cars;` statement in SQL Shell, we will get an error.

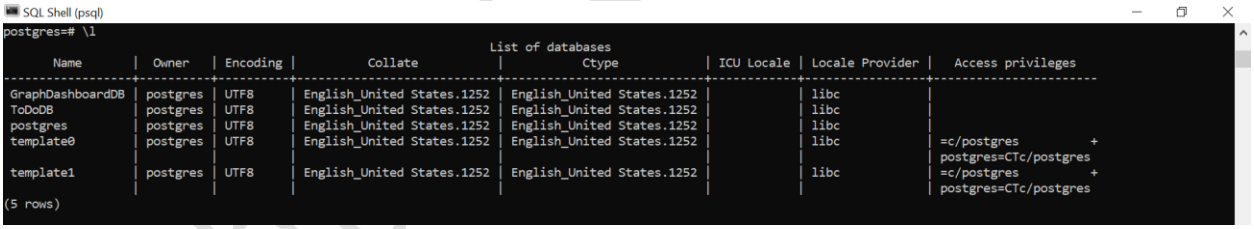
Drop Database

To drop database, run the following command DROP DATABASE.

```
DROP DATABASE testdb;
```

SQL Shell returns DROP DATABASE, which means that we are successfully deleted testdb database. To see list of database just run the command

```
\l
```



Here, we do not see the deleted database.

DDL (Data Definition Language) discussion is over.

Thanks.

Create Demo Database

Demo Database

Now we want to create more tables with more content to be able to demonstrate more database features. We will create these 6 tables in our PostgreSQL database:

categories
customers
products
orders
order_details
testproducts

At first, we need a database. So we will create a database called ecommercedb.

Create Database

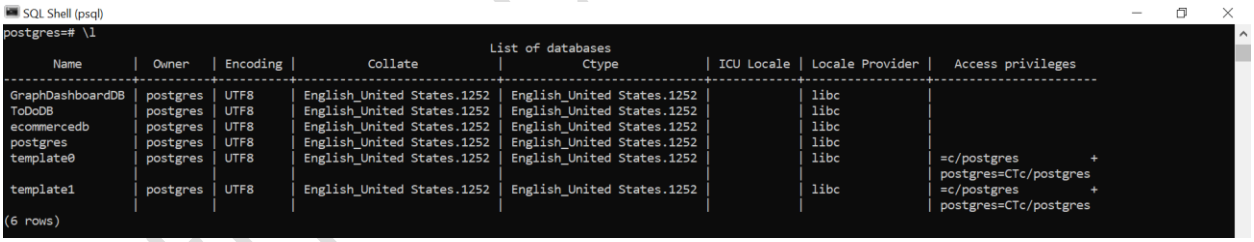
To create database, simply run this statement.

```
CREATE DATABASE ecommercedb;
```

See All Database

Now see all database list by running this statement. You will see our newly created database here. You can also see in pgAdmin 4.

```
\l
```



Select Database

Now we will create all table in our created database. So we need to select the database.

To select database, execute this statement.

```
\c ecommercedb
```

Now we are connected with our created database.

Categories

The following SQL statement will create a table named `categories`:

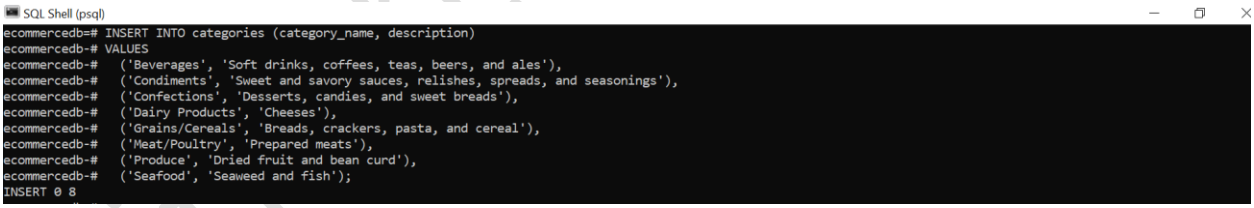
CREATE TABLE categories

```
CREATE TABLE categories (
  category_id SERIAL NOT NULL PRIMARY KEY,
  category_name VARCHAR(255),
  description VARCHAR(255)
);
```

INSERT INTO categories

The following SQL statement will fill the `categories` table with content:

```
INSERT INTO categories (category_name, description)
VALUES
  ('Beverages', 'Soft drinks, coffees, teas, beers, and ales'),
  ('Condiments', 'Sweet and savory sauces, relishes, spreads, and
seasonings'),
  ('Confections', 'Desserts, candies, and sweet breads'),
  ('Dairy Products', 'Cheeses'),
  ('Grains/Cereals', 'Breads, crackers, pasta, and cereal'),
  ('Meat/Poultry', 'Prepared meats'),
  ('Produce', 'Dried fruit and bean curd'),
  ('Seafood', 'Seaweed and fish');
```



SQL Shell returns `INSERT 0 8` that means 8 rows are inserted with data.

Customers

The following SQL statement will create a table named `customers`:

CREATE TABLE customers

```
CREATE TABLE customers (
  customer_id SERIAL NOT NULL PRIMARY KEY,
  customer_name VARCHAR(255),
  contact_name VARCHAR(255),
  address VARCHAR(255),
  city VARCHAR(255),
```

```
  postal_code VARCHAR(255),
  country VARCHAR(255)
);
```

INSERT INTO customers

The following SQL statement will fill the `customers` table with content:

```
INSERT INTO customers (customer_name, contact_name, address, city, postal_code, country)
VALUES
  ('Alfreds Futterkiste', 'Maria Anders', 'Obere Str. 57', 'Berlin', '12209', 'Germany'),
  ('Ana Trujillo Emparedados y helados', 'Ana Trujillo', 'Avda. de la Constitucion 2222', 'Mexico D.F.', '05021', 'Mexico'),
  ('Antonio Moreno Taquera', 'Antonio Moreno', 'Mataderos 2312', 'Mexico D.F.', '05023', 'Mexico'),
  ('Around the Horn', 'Thomas Hardy', '120 Hanover Sq.', 'London', 'WA1 1DP', 'UK'),
  ('Berglunds snabbkoep', 'Christina Berglund', 'Berguvsvegen 8', 'Lulea', 'S-958 22', 'Sweden'),
  ('Blauer See Delikatessen', 'Hanna Moos', 'Forsterstr. 57', 'Mannheim', '68306', 'Germany'),
  ('Blondel pere et fils', 'Frederique Citeaux', '24, place Kleber', 'Strasbourg', '67000', 'France'),
  ('Bolido Comidas preparadas', 'Martin Sommer', 'C/ Araquil, 67', 'Madrid', '28023', 'Spain'),
  ('Bon app', 'Laurence Lebihans', '12, rue des Bouchers', 'Marseille', '13008', 'France'),
  ('Bottom-Dollar Marketse', 'Elizabeth Lincoln', '23 Tsawassen Blvd.', 'Tsawassen', 'T2F 8M4', 'Canada'),
  ('Bs Beverages', 'Victoria Ashworth', 'Fauntleroy Circus', 'London', 'EC2 5NT', 'UK'),
  ('Cactus Comidas para llevar', 'Patricio Simpson', 'Cerrito 333', 'Buenos Aires', '1010', 'Argentina'),
  ('Centro comercial Moctezuma', 'Francisco Chang', 'Sierras de Granada 9993', 'Mexico D.F.', '05022', 'Mexico'),
  ('Chop-suey Chinese', 'Yang Wang', 'Hauptstr. 29', 'Bern', '3012', 'Switzerland'),
  ('Comercio Mineiro', 'Pedro Afonso', 'Av. dos Lusiadas, 23', 'Sao Paulo', '05432-043', 'Brazil'),
  ('Consolidated Holdings', 'Elizabeth Brown', 'Berkeley Gardens 12 Brewery ', 'London', 'WX1 6LT', 'UK'),
  ('Drachenblut Delikatessend', 'Sven Ottlieb', 'Walserweg 21', 'Aachen', '52066', 'Germany'),
  ('Du monde entier', 'Janine Labrune', '67, rue des Cinquante Otages', 'Nantes', '44000', 'France'),
  ('Eastern Connection', 'Ann Devon', '35 King George', 'London', 'WX3 6FW', 'UK'),
  ('Ernst Handel', 'Roland Mendel', 'Kirchgasse 6', 'Graz', '8010', 'Austria'),
  ('Familia Arquibaldo', 'Aria Cruz', 'Rua Oros, 92', 'Sao Paulo', '05442-030', 'Brazil'),
  ('FISSA Fabrica Inter. Salchichas S.A.', 'Diego Roel', 'C/ Moralzarzal, 86', 'Madrid', '28034', 'Spain'),
  ('Folies gourmandes', 'Martine Rance', '184, chaussee de Tournai', 'Lille', '59000', 'France'),
  ('Folk och fe HB', 'Maria Larsson', 'Akergatan 24', 'Brecke', 'S-844 67', 'Sweden'),
  ('Frankenversand', 'Peter Franken', 'Berliner Platz 43', 'Munchen', '80805', 'Germany'),
  ('France restauration', 'Carine Schmitt', '54, rue Royale', 'Nantes', '44000', 'France'),
  ('Franchi S.p.A.', 'Paolo Accorti', 'Via Monte Bianco 34', 'Torino', '10100', 'Italy'),
  ('Furia Bacalhau e Frutos do Mar','Lino Rodriguez ', 'Jardim das rosas n. 32', 'Lisboa', '1675', 'Portugal'),
  ('Galeria del gastronomo', 'Eduardo Saavedra', 'Rambla de Cataluna, 23', 'Barcelona', '08022', 'Spain'),
  ('Godos Cocina Tipica', 'Jose Pedro Freyre', 'C/ Romero, 33', 'Sevilla', '41101', 'Spain'),
  ('Gourmet Lanchonetes', 'Andre Fonseca', 'Av. Brasil, 442', 'Campinas', '04876-786', 'Brazil'),
  ('Great Lakes Food Market', 'Howard Snyder', '2732 Baker Blvd.', 'Eugene', '97403', 'USA'),
  ('GROSELLA-Restaurante', 'Manuel Pereira', '5th Ave. Los Palos Grandes', 'Caracas', '1081', 'Venezuela'),
  ('Hanari Carnes', 'Mario Pontes', 'Rua do Paco, 67', 'Rio de Janeiro', '05454-876', 'Brazil'),
  ('HILARION-Abastos', 'Carlos Hernandez', 'Carrera 22 con Ave. Carlos Soublette #8-35', 'San Cristobal', '5022', 'Venezuela'),
  ('Hungry Coyote Import Store', 'Yoshi Latimer', 'City Center Plaza 516 Main St.', 'Elgin', '97827', 'USA'),
  ('Hungry Owl All-Night Grocers', 'Patricia McKenna', '8 Johnstown Road', 'Cork', '', 'Ireland'),
  ('Island Trading', 'Helen Bennett', 'Garden House Crowther Way', 'Cowes', 'PO31 7PJ', 'UK'),
  ('Koniglich Essen', 'Philip Cramer', 'Maubelstr. 90', 'Brandenburg', '14776', 'Germany'),
  ('La corne d abondance', 'Daniel Tonini', '67, avenue de l Europe', 'Versailles', '78000', 'France'),
  ('La maison d Asie', 'Annette Roulet', '1 rue Alsace-Lorraine', 'Toulouse', '31000', 'France'),
  ('Laughing Bacchus Wine Cellars', 'Yoshi Tannamuri', '1900 Oak St.', 'Vancouver', 'V3F 2K1', 'Canada'),
  ('Lazy K Kountry Store', 'John Steel', '12 Orchestra Terrace', 'Walla Walla', '99362', 'USA'),
  ('Lehmanns Marktstand', 'Renate Messner', 'Magazinweg 7', 'Frankfurt a.M. ', '60528', 'Germany'),
  ('Lets Stop N Shop', 'Jaime Yorres', '87 Polk St. Suite 5', 'San Francisco', '94117', 'USA'),
  ('LILA-Supermercado', 'Carlos Gonzalez', 'Carrera 52 con Ave. Bolivar #65-98 Llano Largo', 'Barquisimeto', '3508',
'Venezuela'),
  ('LINO-Delicateses', 'Felipe Izquierdo', 'Ave. 5 de Mayo Porlamar', 'I. de Margarita', '4980', 'Venezuela'),
  ('Lonesome Pine Restaurant', 'Fran Wilson', '89 Chiaroscuro Rd.', 'Portland', '97219', 'USA'),
  ('Magazzini Alimentari Riuniti', 'Giovanni Rovelli', 'Via Ludovico il Moro 22', 'Bergamo', '24100', 'Italy'),
  ('Maison Dewey', 'Catherine Dewey', 'Rue Joseph-Bens 532', 'Bruxelles', 'B-1180', 'Belgium'),
  ('Mere Paillarde', 'Jean Fresniere', '43 rue St. Laurent', 'Montreal', 'H1J 1C3', 'Canada'),
  ('Morgenstern Gesundkost', 'Alexander Feuer', 'Heerstr. 22', 'Leipzig', '04179', 'Germany'),
  ('North/South', 'Simon Crowther', 'South House 300 Queensbridge', 'London', 'SW7 1RZ', 'UK'),
  ('Oceano Atlantico Ltda.', 'Yvonne Moncada', 'Ing. Gustavo Moncada 8585 Piso 20-A', 'Buenos Aires', '1010', 'Argentina'),
  ('Old World Delicatessen', 'Rene Phillips', '2743 Bering St.', 'Anchorage', '99508', 'USA'),
  ('Ottilies Keseladen', 'Henriette Pfalzheim', 'Mehrheimerstr. 369', 'Koln', '50739', 'Germany'),
  ('Paris specialites', 'Marie Bertrand', '265, boulevard Charonne', 'Paris', '75012', 'France'),
  ('Pericles Comidas clasicas', 'Guillermo Fernandez', 'Calle Dr. Jorge Cash 321', 'Mexico D.F.', '05033', 'Mexico'),
  ('Piccolo und mehr', 'Georg Pipps', 'Geislweg 14', 'Salzburg', '5020', 'Austria'),
```

```
  ('Princesa Isabel Vinhoss', 'Isabel de Castro', 'Estrada da saude n. 58', 'Lisboa', '1756', 'Portugal'),
  ('Que Delicia', 'Bernardo Batista', 'Rua da Panificadora, 12', 'Rio de Janeiro', '02389-673', 'Brazil'),
  ('Queen Cozinha', 'Lucia Carvalho', 'Alameda dos Canarios, 891', 'Sao Paulo', '05487-020', 'Brazil'),
  ('QUICK-Stop', 'Horst Kloss', 'Taucherstrasse 10', 'Cunewalde', '01307', 'Germany'),
  ('Rancho grande', 'Sergio Gutiarrez', 'Av. del Libertador 900', 'Buenos Aires', '1010', 'Argentina'),
  ('Rattlesnake Canyon Grocery', 'Paula Wilson', '2817 Milton Dr.', 'Albuquerque', '87110', 'USA'),
  ('Reggiani Caseifici', 'Maurizio Moroni', 'Strada Provinciale 124', 'Reggio Emilia', '42100', 'Italy'),
  ('Ricardo Adocicados', 'Janete Limeira', 'Av. Copacabana, 267', 'Rio de Janeiro', '02389-890', 'Brazil'),
  ('Richter Supermarkt', 'Michael Holz', 'Grenzacherweg 237', 'Genève', '1203', 'Switzerland'),
  ('Romero y tomillo', 'Alejandra Camino', 'Gran Via, 1', 'Madrid', '28001', 'Spain'),
  ('Santa Gourmet', 'Jonas Bergulfsen', 'Erling Skakkes gate 78', 'Stavern', '4110', 'Norway'),
  ('Save-a-lot Markets', 'Jose Pavarotti', '187 Suffolk Ln.', 'Boise', '83720', 'USA'),
  ('Seven Seas Imports', 'Hari Kumar', '90 Wadhurst Rd.', 'London', 'OX15 4NB', 'UK'),
  ('Simons bistro', 'Jytte Petersen', 'Vinbeltet 34', 'Kobenhavn', '1734', 'Denmark'),
  ('Specialites du monde', 'Dominique Perrier', '25, rue Lauriston', 'Paris', '75016', 'France'),
  ('Split Rail Beer & Ale', 'Art Braunschweiger', 'P.O. Box 555', 'Lander', '82520', 'USA'),
  ('Supremes delices', 'Pascale Cartrain', 'Boulevard Tirou, 255', 'Charleroi', 'B-6000', 'Belgium'),
  ('The Big Cheese', 'Liz Nixon', '89 Jefferson Way Suite 2', 'Portland', '97201', 'USA'),
  ('The Cracker Box', 'Liu Wong', '55 Grizzly Peak Rd.', 'Butte', '59801', 'USA'),
  ('Toms Spezialiteten', 'Karin Josephs', 'Luisenstr. 48', 'Manster', '44087', 'Germany'),
  ('Tortuga Restaurante', 'Miguel Angel Paolino', 'Avda. Azteca 123', 'Mexico D.F.', '05033', 'Mexico'),
  ('Tradicao Hipermercados', 'Anabela Domingues', 'Av. Ines de Castro, 414', 'Sao Paulo', '05634-030', 'Brazil'),
  ('Trails Head Gourmet Provisioners', 'Helvetius Nagy', '722 DaVinci Blvd.', 'Kirkland', '98034', 'USA'),
  ('Vaffeljernet', 'Palle Ibsen', 'Smagsloget 45', 'Arhus', '8200', 'Denmark'),
  ('Victuailles en stock', 'Mary Saveley', '2, rue du Commerce', 'Lyon', '69004', 'France'),
  ('Vins et alcools Chevalier', 'Paul Henriot', '59 rue de l Abbaye', 'Reims', '51100', 'France'),
  ('Die Wandernde Kuh', 'Rita Moller', 'Adenauerallee 900', 'Stuttgart', '70563', 'Germany'),
  ('Wartian Herkku', 'Pirkko Koskitalo', 'Torikatu 38', 'Oulu', '90110', 'Finland'),
  ('Wellington Importadora', 'Paula Parente', 'Rua do Mercado, 12', 'Resende', '08737-363', 'Brazil'),
  ('White Clover Markets', 'Karl Jablonski', '305 - 14th Ave. S. Suite 3B', 'Seattle', '98128', 'USA'),
  ('Wilman Kala', 'Matti Karttunen', 'Keskuskatu 45', 'Helsinki', '21240', 'Finland'),
  ('Wolski', 'Zbyszek', 'ul. Filtrowa 68', 'Walla', '01-012', 'Poland');
```

If you write above statement in your SQL Shell, you will see that it returns `INSERT 0 91`.

```
INSERT 0 91
```

Which means we have inserted 91 row in our customers table.

Products

The following SQL statement will create a table named `products`:

CREATE TABLE products

```
CREATE TABLE products (
  product_id SERIAL NOT NULL PRIMARY KEY,
  product_name VARCHAR(255),
  category_id INT,
  unit VARCHAR(255),
  price DECIMAL(10, 2)
);
```

INSERT INTO products

The following SQL statement will fill the `products` table with content:

```
INSERT INTO products (product_id, product_name, category_id, unit, price)
VALUES
  (1, 'Chais', 1, '10 boxes x 20 bags', 18),
  (2, 'Chang', 1, '24 - 12 oz bottles', 19),
```

```
(3, 'Aniseed Syrup', 2, '12 - 550 ml bottles', 10),
(4, 'Chef Antons Cajun Seasoning', 2, '48 - 6 oz jars', 22),
(5, 'Chef Antons Gumbo Mix', 2, '36 boxes', 21.35),
(6, 'Grandmas Boysenberry Spread', 2, '12 - 8 oz jars', 25),
(7, 'Uncle Bobs Organic Dried Pears', 7, '12 - 1 lb pkgs.', 30),
(8, 'Northwoods Cranberry Sauce', 2, '12 - 12 oz jars', 40),
(9, 'Mishi Kobe Niku', 6, '18 - 500 g pkgs.', 97),
(10, 'Ikura', 8, '12 - 200 ml jars', 31),
(11, 'Queso Cabrales', 4, '1 kg pkg.', 21),
(12, 'Queso Manchego La Pastora', 4, '10 - 500 g pkgs.', 38),
(13, 'Konbu', 8, '2 kg box', 6),
(14, 'Tofu', 7, '40 - 100 g pkgs.', 23.25),
(15, 'Genen Shouyu', 2, '24 - 250 ml bottles', 15.5),
(16, 'Pavlova', 3, '32 - 500 g boxes', 17.45),
(17, 'Alice Mutton', 6, '20 - 1 kg tins', 39),
(18, 'Carnarvon Tigers', 8, '16 kg pkg.', 62.5),
(19, 'Teatime Chocolate Biscuits', 3, '10 boxes x 12 pieces', 9.2),
(20, 'Sir Rodneys Marmalade', 3, '30 gift boxes', 81),
(21, 'Sir Rodneys Scones', 3, '24 pkgs. x 4 pieces', 10),
(22, 'Gustafs Kneckebrod', 5, '24 - 500 g pkgs.', 21),
(23, 'Tunnbrod', 5, '12 - 250 g pkgs.', 9),
(24, 'Guarani Fantastica', 1, '12 - 355 ml cans', 4.5),
(25, 'NuNuCa Nui-Nougat-Creme', 3, '20 - 450 g glasses', 14),
(26, 'Gumber Gummiberchen', 3, '100 - 250 g bags', 31.23),
(27, 'Schoggi Schokolade', 3, '100 - 100 g pieces', 43.9),
(28, 'Rassle Sauerkraut', 7, '25 - 825 g cans', 45.6),
(29, 'Thoringer Rostbratwurst', 6, '50 bags x 30 sausgs.', 123.79),
(30, 'Nord-Ost Matjeshering', 8, '10 - 200 g glasses', 25.89),
(31, 'Gorgonzola Telino', 4, '12 - 100 g pkgs', 12.5),
(32, 'Mascarpone Fabioli', 4, '24 - 200 g pkgs.', 32),
(33, 'Geitost', 4, '500 g', 2.5),
(34, 'Sasquatch Ale', 1, '24 - 12 oz bottles', 14),
(35, 'Steeleye Stout', 1, '24 - 12 oz bottles', 18),
(36, 'Inlagd Sill', 8, '24 - 250 g jars', 19),
(37, 'Gravad lax', 8, '12 - 500 g pkgs.', 26),
(38, 'Cote de Blaye', 1, '12 - 75 cl bottles', 263.5),
(39, 'Chartreuse verte', 1, '750 cc per bottle', 18),
(40, 'Boston Crab Meat', 8, '24 - 4 oz tins', 18.4),
(41, 'Jacks New England Clam Chowder', 8, '12 - 12 oz cans', 9.65),
(42, 'Singaporean Hokkien Fried Mee', 5, '32 - 1 kg pkgs.', 14),
(43, 'Ipoh Coffee', 1, '16 - 500 g tins', 46),
(44, 'Gula Malacca', 2, '20 - 2 kg bags', 19.45),
(45, 'Rogede sild', 8, '1k pkg.', 9.5),
(46, 'Spegesild', 8, '4 - 450 g glasses', 12),
(47, 'Zaanse koeken', 3, '10 - 4 oz boxes', 9.5),
(48, 'Chocolade', 3, '10 pkgs.', 12.75),
(49, 'Maxilaku', 3, '24 - 50 g pkgs.', 20),
(50, 'Valkoinen suklaa', 3, '12 - 100 g bars', 16.25),
(51, 'Manjimup Dried Apples', 7, '50 - 300 g pkgs.', 53),
(52, 'Filo Mix', 5, '16 - 2 kg boxes', 7),
(53, 'Perth Pasties', 6, '48 pieces', 32.8),
(54, 'Tourtiare', 6, '16 pies', 7.45),
(55, 'Pate chinois', 6, '24 boxes x 2 pies', 24),
(56, 'Gnocchi di nonna Alice', 5, '24 - 250 g pkgs.', 38),
(57, 'Ravioli Angelo', 5, '24 - 250 g pkgs.', 19.5),
(58, 'Escargots de Bourgogne', 8, '24 pieces', 13.25),
(59, 'Raclette Courdavault', 4, '5 kg pkg.', 55),
(60, 'Camembert Pierrot', 4, '15 - 300 g rounds', 34),
(61, 'Sirop d arable', 2, '24 - 500 ml bottles', 28.5),
(62, 'Tarte au sucre', 3, '48 pies', 49.3),
(63, 'Vegie-spread', 2, '15 - 625 g jars', 43.9),
(64, 'Wimmers gute Semmelknadel', 5, '20 bags x 4 pieces', 33.25),
(65, 'Louisiana Fiery Hot Pepper Sauce', 2, '32 - 8 oz bottles', 21.05),
(66, 'Louisiana Hot Spiced Okra', 2, '24 - 8 oz jars', 17),
(67, 'Laughing Lumberjack Lager', 1, '24 - 12 oz bottles', 14),
(68, 'Scottish Longbreads', 3, '10 boxes x 8 pieces', 12.5),
(69, 'Gudbrandsdalsost', 4, '10 kg pkg.', 36),
(70, 'Outback Lager', 1, '24 - 355 ml bottles', 15),
(71, 'Flotemysost', 4, '10 - 500 g pkgs.', 21.5),
(72, 'Mozzarella di Giovanni', 4, '24 - 200 g pkgs.', 34.8),
(73, 'Red Kaviar', 8, '24 - 150 g jars', 15),
(74, 'Longlife Tofu', 7, '5 kg pkg.', 10),
(75, 'Rhenbreu Klosterbier', 1, '24 - 0.5 l bottles', 7.75),
(76, 'Lakkalikeeri', 1, '500 ml ', 18),
(77, 'Original Frankfurter gr□ne Soae', 2, '12 boxes', 13);
```

If you write above statement in your SQL Shell, you will see that it returns `INSERT 0 77`. Which means that we have inserted 77 rows in our products table.

Orders

The following SQL statement will create a table named `orders`:

CREATE TABLE orders

```
CREATE TABLE orders (
  order_id SERIAL NOT NULL PRIMARY KEY,
  customer_id INT,
  order_date DATE
);
```

INSERT INTO orders

The following SQL statement will fill the `orders` table with content:

```
INSERT INTO orders (order_id, customer_id, order_date)
VALUES
  (10248, 90, '2021-07-04'),
  (10249, 81, '2021-07-05'),
  (10250, 34, '2021-07-08'),
  (10251, 84, '2021-07-08'),
  (10252, 76, '2021-07-09'),
  (10253, 34, '2021-07-10'),
  (10254, 14, '2021-07-11'),
  (10255, 68, '2021-07-12'),
  (10256, 88, '2021-07-15'),
  (10257, 35, '2021-07-16'),
  (10258, 20, '2021-07-17'),
  (10259, 13, '2021-07-18'),
  (10260, 55, '2021-07-19'),
  (10261, 61, '2021-07-19'),
  (10262, 65, '2021-07-22'),
  (10263, 20, '2021-07-23'),
  (10264, 24, '2021-07-24'),
  (10265, 7, '2021-07-25'),
  (10266, 87, '2021-07-26'),
  (10267, 25, '2021-07-29'),
  (10268, 33, '2021-07-30'),
  (10269, 89, '2021-07-31'),
  (10270, 87, '2021-08-01'),
  (10271, 75, '2021-08-01'),
  (10272, 65, '2021-08-02'),
  (10273, 63, '2021-08-05'),
  (10274, 85, '2021-08-06'),
  (10275, 49, '2021-08-07'),
  (10276, 80, '2021-08-08'),
  (10277, 52, '2021-08-09'),
  (10278, 5, '2021-08-12'),
  (10279, 44, '2021-08-13'),
  (10280, 5, '2021-08-14'),
  (10281, 69, '2021-08-14'),
  (10282, 69, '2021-08-15'),
  (10283, 46, '2021-08-16'),
  (10284, 44, '2021-08-19'),
  (10285, 63, '2021-08-20'),
  (10286, 63, '2021-08-21'),
  (10287, 67, '2021-08-22'),
  (10288, 66, '2021-08-23'),
  (10289, 11, '2021-08-26'),
  (10290, 15, '2021-08-27'),
  (10291, 61, '2021-08-27'),
  (10292, 81, '2021-08-28'),
  (10293, 80, '2021-08-29'),
  (10294, 65, '2021-08-30'),
  (10295, 85, '2021-09-02'),
  (10296, 46, '2021-09-03'),
  (10297, 7, '2021-09-04'),
  (10298, 37, '2021-09-05'),
  (10299, 67, '2021-09-06'),
  (10300, 49, '2021-09-09'),
  (10301, 86, '2021-09-09'),
  (10302, 76, '2021-09-10'),
  (10303, 30, '2021-09-11'),
  (10304, 80, '2021-09-12'),
  (10305, 55, '2021-09-13'),
  (10306, 69, '2021-09-16'),
  (10307, 48, '2021-09-17'),
  (10308, 2, '2021-09-18'),
  (10309, 37, '2021-09-19'),
  (10310, 77, '2021-09-20'),
  (10311, 18, '2021-09-20'),
  (10312, 86, '2021-09-23'),
  (10313, 63, '2021-09-24'),
  (10314, 65, '2021-09-25'),
  (10315, 38, '2021-09-26'),
  (10316, 65, '2021-09-27'),
  (10317, 48, '2021-09-30'),
  (10318, 38, '2021-10-01'),
  (10319, 80, '2021-10-02'),
  (10320, 87, '2021-10-03'),
  (10321, 38, '2021-10-03'),
  (10322, 58, '2021-10-04'),
  (10323, 39, '2021-10-07'),
  (10324, 71, '2021-10-08'),
  (10325, 39, '2021-10-09'),
  (10326, 8, '2021-10-10'),
  (10327, 24, '2021-10-11'),
  (10328, 28, '2021-10-14'),
  (10329, 75, '2021-10-15'),
  (10330, 46, '2021-10-16'),
```

```
(10331, 9, '2021-10-16'),
(10332, 51, '2021-10-17'),
(10333, 87, '2021-10-18'),
(10334, 84, '2021-10-21'),
(10335, 37, '2021-10-22'),
(10336, 60, '2021-10-23'),
(10337, 25, '2021-10-24'),
(10338, 55, '2021-10-25'),
(10339, 51, '2021-10-28'),
(10340, 9, '2021-10-29'),
(10341, 73, '2021-10-29'),
(10342, 25, '2021-10-30'),
(10343, 44, '2021-10-31'),
(10344, 89, '2021-11-01'),
(10345, 63, '2021-11-04'),
(10346, 65, '2021-11-05'),
(10347, 21, '2021-11-06'),
(10348, 86, '2021-11-07'),
(10349, 75, '2021-11-08'),
(10350, 41, '2021-11-11'),
(10351, 20, '2021-11-11'),
(10352, 28, '2021-11-12'),
(10353, 59, '2021-11-13'),
(10354, 58, '2021-11-14'),
(10355, 4, '2021-11-15'),
(10356, 86, '2021-11-18'),
(10357, 46, '2021-11-19'),
(10358, 41, '2021-11-20'),
(10359, 72, '2021-11-21'),
(10360, 7, '2021-11-22'),
(10361, 63, '2021-11-22'),
(10362, 9, '2021-11-25'),
(10363, 17, '2021-11-26'),
(10364, 19, '2021-11-26'),
(10365, 3, '2021-11-27'),
(10366, 29, '2021-11-28'),
(10367, 83, '2021-11-28'),
(10368, 20, '2021-11-29'),
(10369, 75, '2021-12-02'),
(10370, 14, '2021-12-03'),
(10371, 41, '2021-12-03'),
(10372, 62, '2021-12-04'),
(10373, 37, '2021-12-05'),
(10374, 91, '2021-12-05'),
(10375, 36, '2021-12-06'),
(10376, 51, '2021-12-09'),
(10377, 72, '2021-12-09'),
(10378, 24, '2021-12-10'),
(10379, 61, '2021-12-11'),
(10380, 37, '2021-12-12'),
(10381, 46, '2021-12-12'),
(10382, 20, '2021-12-13'),
(10383, 4, '2021-12-16'),
(10384, 5, '2021-12-16'),
(10385, 75, '2021-12-17'),
(10386, 21, '2021-12-18'),
(10387, 70, '2021-12-18'),
(10388, 72, '2021-12-19'),
(10389, 10, '2021-12-20'),
(10390, 20, '2021-12-23'),
(10391, 17, '2021-12-23'),
(10392, 59, '2021-12-24'),
(10393, 71, '2021-12-25'),
(10394, 36, '2021-12-25'),
(10395, 35, '2021-12-26'),
(10396, 25, '2021-12-27'),
(10397, 60, '2021-12-27'),
(10398, 71, '2021-12-30'),
(10399, 83, '2021-12-31'),
(10400, 19, '2022-01-01'),
(10401, 65, '2022-01-01'),
(10402, 20, '2022-01-02'),
(10403, 20, '2022-01-03'),
(10404, 49, '2022-01-03'),
(10405, 47, '2022-01-06'),
(10406, 62, '2022-01-07'),
(10407, 56, '2022-01-07'),
(10408, 23, '2022-01-08'),
(10409, 54, '2022-01-09'),
(10410, 10, '2022-01-10'),
(10411, 10, '2022-01-10'),
(10412, 87, '2022-01-13'),
(10413, 41, '2022-01-14'),
(10414, 21, '2022-01-14'),
(10415, 36, '2022-01-15'),
(10416, 87, '2022-01-16'),
(10417, 73, '2022-01-16'),
(10418, 63, '2022-01-17'),
(10419, 68, '2022-01-20'),
(10420, 88, '2022-01-21'),
(10421, 61, '2022-01-21'),
(10422, 27, '2022-01-22'),
(10423, 31, '2022-01-23'),
(10424, 51, '2022-01-23'),
(10425, 41, '2022-01-24'),
(10426, 29, '2022-01-27'),
(10427, 59, '2022-01-27'),
(10428, 66, '2022-01-28'),
(10429, 37, '2022-01-29'),
(10430, 20, '2022-01-30'),
(10431, 10, '2022-01-30'),
(10432, 75, '2022-01-31'),
(10433, 60, '2022-02-03'),
(10434, 24, '2022-02-03'),
(10435, 16, '2022-02-04'),
(10436, 7, '2022-02-05'),
(10437, 87, '2022-02-05'),
(10438, 79, '2022-02-06'),
(10439, 51, '2022-02-07'),
(10440, 71, '2022-02-10'),
(10441, 55, '2022-02-10'),
(10442, 20, '2022-02-11'),
(10443, 66, '2022-02-12'),
(10444, 5, '2022-02-12'),
(10445, 5, '2022-02-13'),
(10446, 79, '2022-02-14'),
(10447, 67, '2022-02-14'),
(10448, 64, '2022-02-17'),
(10449, 7, '2022-02-18'),
(10450, 84, '2022-02-19'),
(10451, 63, '2022-02-19'),
(10452, 71, '2022-02-20'),
(10453, 4, '2022-02-21'),
(10454, 41, '2022-02-21'),
(10455, 87, '2022-02-24'),
(10456, 39, '2022-02-25'),
(10457, 39, '2022-02-25'),
(10458, 76, '2022-02-26'),
(10459, 84, '2022-02-27'),
(10460, 24, '2022-02-28'),
(10461, 46, '2022-02-28'),
(10462, 16, '2022-03-03'),
```

```
(10463, 76, '2022-03-04'),
(10464, 28, '2022-03-04'),
(10465, 83, '2022-03-05'),
(10466, 15, '2022-03-06'),
(10467, 49, '2022-03-06'),
(10468, 39, '2022-03-07'),
(10469, 89, '2022-03-10'),
(10470, 9, '2022-03-11'),
(10471, 11, '2022-03-11'),
(10472, 72, '2022-03-12'),
(10473, 38, '2022-03-13'),
(10474, 58, '2022-03-13'),
(10475, 76, '2022-03-14'),
(10476, 35, '2022-03-17'),
(10477, 60, '2022-03-17'),
(10478, 84, '2022-03-18'),
(10479, 65, '2022-03-19'),
(10480, 23, '2022-03-20'),
(10481, 67, '2022-03-20'),
(10482, 43, '2022-03-21'),
(10483, 89, '2022-03-24'),
(10484, 11, '2022-03-24'),
(10485, 47, '2022-03-25'),
(10486, 35, '2022-03-26'),
(10487, 62, '2022-03-26'),
(10488, 25, '2022-03-27'),
(10489, 59, '2022-03-28'),
(10490, 35, '2022-03-31'),
(10491, 28, '2022-03-31'),
(10492, 10, '2022-04-01'),
(10493, 41, '2022-04-02'),
(10494, 15, '2022-04-02'),
(10495, 42, '2022-04-03'),
(10496, 81, '2022-04-04'),
(10497, 44, '2022-04-04'),
(10498, 35, '2022-04-07'),
(10499, 46, '2022-04-08'),
(10500, 41, '2022-04-09'),
(10501, 6, '2022-04-09'),
(10502, 58, '2022-04-10'),
(10503, 37, '2022-04-11'),
(10504, 89, '2022-04-11'),
(10505, 51, '2022-04-14'),
(10506, 39, '2022-04-15'),
(10507, 3, '2022-04-15'),
(10508, 56, '2022-04-16'),
(10509, 6, '2022-04-17'),
(10510, 71, '2022-04-18'),
(10511, 9, '2022-04-18'),
(10512, 21, '2022-04-21'),
(10513, 86, '2022-04-22'),
(10514, 20, '2022-04-22'),
(10515, 63, '2022-04-23'),
(10516, 37, '2022-04-24'),
(10517, 53, '2022-04-24'),
(10518, 80, '2022-04-25'),
(10519, 14, '2022-04-28'),
(10520, 70, '2022-04-29'),
(10521, 12, '2022-04-29'),
(10522, 44, '2022-04-30'),
(10523, 72, '2022-05-01'),
(10524, 5, '2022-05-01'),
(10525, 9, '2022-05-02'),
(10526, 87, '2022-05-05'),
(10527, 63, '2022-05-05'),
(10528, 32, '2022-05-06'),
(10529, 50, '2022-05-07'),
(10530, 59, '2022-05-08'),
(10531, 54, '2022-05-08'),
(10532, 19, '2022-05-09'),
(10533, 24, '2022-05-12'),
(10534, 44, '2022-05-12'),
(10535, 3, '2022-05-13'),
(10536, 44, '2022-05-14'),
(10537, 68, '2022-05-14'),
(10538, 11, '2022-05-15'),
(10539, 11, '2022-05-16'),
(10540, 63, '2022-05-19'),
(10541, 34, '2022-05-19'),
(10542, 39, '2022-05-20'),
(10543, 46, '2022-05-21'),
(10544, 48, '2022-05-21'),
(10545, 43, '2022-05-22'),
(10546, 84, '2022-05-23'),
(10547, 72, '2022-05-23'),
(10548, 79, '2022-05-26'),
(10549, 63, '2022-05-27'),
(10550, 30, '2022-05-28'),
(10551, 28, '2022-05-28'),
(10552, 35, '2022-05-29'),
(10553, 87, '2022-05-30'),
(10554, 56, '2022-05-30'),
(10555, 71, '2022-06-02'),
(10556, 73, '2022-06-03'),
(10557, 44, '2022-06-03'),
(10558, 4, '2022-06-04'),
(10559, 7, '2022-06-05'),
(10560, 25, '2022-06-06'),
(10561, 24, '2022-06-06'),
(10562, 66, '2022-06-09'),
(10563, 67, '2022-06-10'),
(10564, 65, '2022-06-10'),
(10565, 51, '2022-06-11'),
(10566, 7, '2022-06-12'),
(10567, 37, '2022-06-12'),
(10568, 29, '2022-06-13'),
(10569, 65, '2022-06-16'),
(10570, 51, '2022-06-17'),
(10571, 20, '2022-06-17'),
(10572, 5, '2022-06-18'),
(10573, 3, '2022-06-19'),
(10574, 82, '2022-06-19'),
(10575, 52, '2022-06-20'),
(10576, 80, '2022-06-23'),
(10577, 82, '2022-06-23'),
(10578, 11, '2022-06-24'),
(10579, 45, '2022-06-25'),
(10580, 56, '2022-06-26'),
(10581, 21, '2022-06-26'),
(10582, 6, '2022-06-27'),
(10583, 87, '2022-06-30'),
(10584, 7, '2022-06-30'),
(10585, 88, '2022-07-01'),
(10586, 66, '2022-07-02'),
(10587, 61, '2022-07-02'),
(10588, 63, '2022-07-03'),
(10589, 32, '2022-07-04'),
(10590, 51, '2022-07-07'),
(10591, 83, '2022-07-07'),
(10592, 44, '2022-07-08'),
(10593, 44, '2022-07-09'),
(10594, 55, '2022-07-09'),
```

```
(10595, 20, '2022-07-10'),
(10596, 89, '2022-07-11'),
(10597, 59, '2022-07-11'),
(10598, 65, '2022-07-14'),
(10599, 11, '2022-07-15'),
(10600, 36, '2022-07-16'),
(10601, 35, '2022-07-16'),
(10602, 83, '2022-07-17'),
(10603, 71, '2022-07-18'),
(10604, 28, '2022-07-18'),
(10605, 51, '2022-07-21'),
(10606, 81, '2022-07-22'),
(10607, 71, '2022-07-22'),
(10608, 79, '2022-07-23'),
(10609, 18, '2022-07-24'),
(10610, 41, '2022-07-25'),
(10611, 91, '2022-07-25'),
(10612, 71, '2022-07-28'),
(10613, 35, '2022-07-29'),
(10614, 6, '2022-07-29'),
(10615, 90, '2022-07-30'),
(10616, 32, '2022-07-31'),
(10617, 32, '2022-07-31'),
(10618, 51, '2022-08-01'),
(10619, 51, '2022-08-04'),
(10620, 42, '2022-08-05'),
(10621, 38, '2022-08-05'),
(10622, 67, '2022-08-06'),
(10623, 25, '2022-08-07'),
(10624, 78, '2022-08-07'),
(10625, 2, '2022-08-08'),
(10626, 5, '2022-08-11'),
(10627, 71, '2022-08-11'),
(10628, 7, '2022-08-12'),
(10629, 30, '2022-08-12'),
(10630, 39, '2022-08-13'),
(10631, 41, '2022-08-14'),
(10632, 86, '2022-08-14'),
(10633, 20, '2022-08-15'),
(10634, 23, '2022-08-15'),
(10635, 49, '2022-08-18'),
(10636, 87, '2022-08-19'),
(10637, 62, '2022-08-19'),
(10638, 47, '2022-08-20'),
(10639, 70, '2022-08-20'),
(10640, 86, '2022-08-21'),
(10641, 35, '2022-08-22'),
(10642, 73, '2022-08-22'),
(10643, 1, '2022-08-25'),
(10644, 88, '2022-08-25'),
(10645, 34, '2022-08-26'),
(10646, 37, '2022-08-27'),
(10647, 61, '2022-08-27'),
(10648, 67, '2022-08-28'),
(10649, 50, '2022-08-28'),
(10650, 21, '2022-08-29'),
(10651, 86, '2022-09-01'),
(10652, 31, '2022-09-01'),
(10653, 25, '2022-09-02'),
(10654, 5, '2022-09-02'),
(10655, 66, '2022-09-03'),
(10656, 32, '2022-09-04'),
(10657, 71, '2022-09-04'),
(10658, 63, '2022-09-05'),
(10659, 62, '2022-09-05'),
(10660, 36, '2022-09-08'),
(10661, 37, '2022-09-09'),
(10662, 48, '2022-09-09'),
(10663, 9, '2022-09-10'),
(10664, 28, '2022-09-10'),
(10665, 48, '2022-09-11'),
(10666, 68, '2022-09-12'),
(10667, 20, '2022-09-12'),
(10668, 86, '2022-09-15'),
(10669, 73, '2022-09-15'),
(10670, 25, '2022-09-16'),
(10671, 26, '2022-09-17'),
(10672, 5, '2022-09-17'),
(10673, 90, '2022-09-18'),
(10674, 38, '2022-09-18'),
(10675, 25, '2022-09-19'),
(10676, 80, '2022-09-22'),
(10677, 3, '2022-09-22'),
(10678, 71, '2022-09-23'),
(10679, 7, '2022-09-23'),
(10680, 55, '2022-09-24'),
(10681, 32, '2022-09-25'),
(10682, 3, '2022-09-25'),
(10683, 18, '2022-09-26'),
(10684, 56, '2022-09-26'),
(10685, 31, '2022-09-29'),
(10686, 59, '2022-09-30'),
(10687, 37, '2022-09-30'),
(10688, 83, '2022-10-01'),
(10689, 5, '2022-10-01'),
(10690, 34, '2022-10-02'),
(10691, 63, '2022-10-03'),
(10692, 1, '2022-10-03'),
(10693, 89, '2022-10-06'),
(10694, 63, '2022-10-06'),
(10695, 90, '2022-10-07'),
(10696, 89, '2022-10-08'),
(10697, 47, '2022-10-08'),
(10698, 20, '2022-10-09'),
(10699, 52, '2022-10-09'),
(10700, 71, '2022-10-10'),
(10701, 37, '2022-10-13'),
(10702, 1, '2022-10-13'),
(10703, 24, '2022-10-14'),
(10704, 62, '2022-10-14'),
(10705, 35, '2022-10-15'),
(10706, 55, '2022-10-16'),
(10707, 4, '2022-10-16'),
(10708, 77, '2022-10-17'),
(10709, 31, '2022-10-17'),
(10710, 27, '2022-10-20'),
(10711, 71, '2022-10-21'),
(10712, 37, '2022-10-21'),
(10713, 71, '2022-10-22'),
(10714, 71, '2022-10-22'),
(10715, 9, '2022-10-23'),
(10716, 64, '2022-10-24'),
(10717, 25, '2022-10-24'),
(10718, 39, '2022-10-27'),
(10719, 45, '2022-10-27'),
(10720, 61, '2022-10-28'),
(10721, 63, '2022-10-29'),
(10722, 71, '2022-10-29'),
(10723, 89, '2022-10-30'),
(10724, 51, '2022-10-30'),
(10725, 21, '2022-10-31'),
(10726, 19, '2022-11-03'),
```

```
(10727, 66, '2022-11-03'),
(10728, 62, '2022-11-04'),
(10729, 47, '2022-11-04'),
(10730, 9, '2022-11-05'),
(10731, 14, '2022-11-06'),
(10732, 9, '2022-11-06'),
(10733, 5, '2022-11-07'),
(10734, 31, '2022-11-07'),
(10735, 45, '2022-11-10'),
(10736, 37, '2022-11-11'),
(10737, 85, '2022-11-11'),
(10738, 74, '2022-11-12'),
(10739, 85, '2022-11-12'),
(10740, 89, '2022-11-13'),
(10741, 4, '2022-11-14'),
(10742, 10, '2022-11-14'),
(10743, 4, '2022-11-17'),
(10744, 83, '2022-11-17'),
(10745, 63, '2022-11-18'),
(10746, 14, '2022-11-19'),
(10747, 59, '2022-11-19'),
(10748, 71, '2022-11-20'),
(10749, 38, '2022-11-20'),
(10750, 87, '2022-11-21'),
(10751, 68, '2022-11-24'),
(10752, 53, '2022-11-24'),
(10753, 27, '2022-11-25'),
(10754, 49, '2022-11-25'),
(10755, 9, '2022-11-26'),
(10756, 75, '2022-11-27'),
(10757, 71, '2022-11-27'),
(10758, 68, '2022-11-28'),
(10759, 2, '2022-11-28'),
(10760, 50, '2022-12-01'),
(10761, 65, '2022-12-02'),
(10762, 24, '2022-12-02'),
(10763, 23, '2022-12-03'),
(10764, 20, '2022-12-03'),
(10765, 63, '2022-12-04'),
(10766, 56, '2022-12-05'),
(10767, 76, '2022-12-05'),
(10768, 4, '2022-12-08'),
(10769, 83, '2022-12-08'),
(10770, 34, '2022-12-09'),
(10771, 20, '2022-12-10'),
(10772, 44, '2022-12-10'),
(10773, 20, '2022-12-11'),
(10774, 24, '2022-12-11'),
(10775, 78, '2022-12-12'),
(10776, 20, '2022-12-15'),
(10777, 31, '2022-12-15'),
(10778, 5, '2022-12-16'),
(10779, 52, '2022-12-16'),
(10780, 46, '2022-12-16'),
(10781, 87, '2022-12-17'),
(10782, 12, '2022-12-17'),
(10783, 34, '2022-12-18'),
(10784, 49, '2022-12-18'),
(10785, 33, '2022-12-18'),
(10786, 62, '2022-12-19'),
(10787, 41, '2022-12-19'),
(10788, 63, '2022-12-22'),
(10789, 23, '2022-12-22'),
(10790, 31, '2022-12-22'),
(10791, 25, '2022-12-23'),
(10792, 91, '2022-12-23'),
(10793, 4, '2022-12-24'),
(10794, 61, '2022-12-24'),
(10795, 20, '2022-12-24'),
(10796, 35, '2022-12-25'),
(10797, 17, '2022-12-25'),
(10798, 38, '2022-12-26'),
(10799, 39, '2022-12-26'),
(10800, 72, '2022-12-26'),
(10801, 8, '2022-12-29'),
(10802, 73, '2022-12-29'),
(10803, 88, '2022-12-30'),
(10804, 72, '2022-12-30'),
(10805, 77, '2022-12-30'),
(10806, 84, '2022-12-31'),
(10807, 27, '2022-12-31'),
(10808, 60, '2023-01-01'),
(10809, 88, '2023-01-01'),
(10810, 42, '2023-01-01'),
(10811, 47, '2023-01-02'),
(10812, 66, '2023-01-02'),
(10813, 67, '2023-01-05'),
(10814, 84, '2023-01-05'),
(10815, 71, '2023-01-05'),
(10816, 32, '2023-01-06'),
(10817, 39, '2023-01-06'),
(10818, 49, '2023-01-07'),
(10819, 12, '2023-01-07'),
(10820, 65, '2023-01-07'),
(10821, 75, '2023-01-08'),
(10822, 82, '2023-01-08'),
(10823, 46, '2023-01-09'),
(10824, 24, '2023-01-09'),
(10825, 17, '2023-01-09'),
(10826, 7, '2023-01-12'),
(10827, 9, '2023-01-12'),
(10828, 64, '2023-01-13'),
(10829, 38, '2023-01-13'),
(10830, 81, '2023-01-13'),
(10831, 70, '2023-01-14'),
(10832, 41, '2023-01-14'),
(10833, 56, '2023-01-15'),
(10834, 81, '2023-01-15'),
(10835, 1, '2023-01-15'),
(10836, 20, '2023-01-16'),
(10837, 5, '2023-01-16'),
(10838, 47, '2023-01-19'),
(10839, 81, '2023-01-19'),
(10840, 47, '2023-01-19'),
(10841, 76, '2023-01-20'),
(10842, 80, '2023-01-20'),
(10843, 84, '2023-01-21'),
(10844, 59, '2023-01-21'),
(10845, 63, '2023-01-21'),
(10846, 76, '2023-01-22'),
(10847, 71, '2023-01-22'),
(10848, 16, '2023-01-23'),
(10849, 39, '2023-01-23'),
(10850, 84, '2023-01-23'),
(10851, 67, '2023-01-26'),
(10852, 65, '2023-01-26'),
(10853, 6, '2023-01-27'),
(10854, 20, '2023-01-27'),
(10855, 55, '2023-01-27'),
(10856, 3, '2023-01-28'),
(10857, 5, '2023-01-28'),
(10858, 40, '2023-01-29'),
```

```
(10859, 25, '2023-01-29'),
(10860, 26, '2023-01-29'),
(10861, 89, '2023-01-30'),
(10862, 44, '2023-01-30'),
(10863, 35, '2023-02-02'),
(10864, 4, '2023-02-02'),
(10865, 63, '2023-02-02'),
(10866, 5, '2023-02-03'),
(10867, 48, '2023-02-03'),
(10868, 62, '2023-02-04'),
(10869, 72, '2023-02-04'),
(10870, 91, '2023-02-04'),
(10871, 9, '2023-02-05'),
(10872, 30, '2023-02-05'),
(10873, 90, '2023-02-06'),
(10874, 30, '2023-02-06'),
(10875, 5, '2023-02-06'),
(10876, 9, '2023-02-09'),
(10877, 67, '2023-02-09'),
(10878, 63, '2023-02-10'),
(10879, 90, '2023-02-10'),
(10880, 24, '2023-02-10'),
(10881, 12, '2023-02-11'),
(10882, 71, '2023-02-11'),
(10883, 48, '2023-02-12'),
(10884, 45, '2023-02-12'),
(10885, 76, '2023-02-12'),
(10886, 34, '2023-02-13'),
(10887, 29, '2023-02-13'),
(10888, 30, '2023-02-16'),
(10889, 65, '2023-02-16'),
(10890, 18, '2023-02-16'),
(10891, 44, '2023-02-17'),
(10892, 50, '2023-02-17'),
(10893, 39, '2023-02-18'),
(10894, 71, '2023-02-18'),
(10895, 20, '2023-02-18'),
(10896, 50, '2023-02-19'),
(10897, 37, '2023-02-19'),
(10898, 54, '2023-02-20'),
(10899, 46, '2023-02-20'),
(10900, 88, '2023-02-20'),
(10901, 35, '2023-02-23'),
(10902, 24, '2023-02-23'),
(10903, 34, '2023-02-24'),
(10904, 89, '2023-02-24'),
(10905, 88, '2023-02-24'),
(10906, 91, '2023-02-25'),
(10907, 74, '2023-02-25'),
(10908, 66, '2023-02-26'),
(10909, 70, '2023-02-26'),
(10910, 90, '2023-02-26'),
(10911, 30, '2023-02-26'),
(10912, 37, '2023-02-26'),
(10913, 62, '2023-02-26'),
(10914, 62, '2023-02-27'),
(10915, 80, '2023-02-27'),
(10916, 64, '2023-02-27'),
(10917, 69, '2023-03-02'),
(10918, 10, '2023-03-02'),
(10919, 47, '2023-03-02'),
(10920, 4, '2023-03-03'),
(10921, 83, '2023-03-03'),
(10922, 34, '2023-03-03'),
(10923, 41, '2023-03-03'),
(10924, 5, '2023-03-04'),
(10925, 34, '2023-03-04'),
(10926, 2, '2023-03-04'),
(10927, 40, '2023-03-05'),
(10928, 29, '2023-03-05'),
(10929, 25, '2023-03-05'),
(10930, 76, '2023-03-06'),
(10931, 68, '2023-03-06'),
(10932, 9, '2023-03-06'),
(10933, 38, '2023-03-06'),
(10934, 44, '2023-03-09'),
(10935, 88, '2023-03-09'),
(10936, 32, '2023-03-09'),
(10937, 12, '2023-03-10'),
(10938, 63, '2023-03-10'),
(10939, 49, '2023-03-10'),
(10940, 9, '2023-03-11'),
(10941, 71, '2023-03-11'),
(10942, 66, '2023-03-11'),
(10943, 11, '2023-03-11'),
(10944, 10, '2023-03-12'),
(10945, 52, '2023-03-12'),
(10946, 83, '2023-03-12'),
(10947, 11, '2023-03-13'),
(10948, 30, '2023-03-13'),
(10949, 10, '2023-03-13'),
(10950, 49, '2023-03-16'),
(10951, 68, '2023-03-16'),
(10952, 1, '2023-03-16'),
(10953, 4, '2023-03-16'),
(10954, 47, '2023-03-17'),
(10955, 24, '2023-03-17'),
(10956, 6, '2023-03-17'),
(10957, 35, '2023-03-18'),
(10958, 54, '2023-03-18'),
(10959, 31, '2023-03-18'),
(10960, 35, '2023-03-19'),
(10961, 62, '2023-03-19'),
(10962, 63, '2023-03-19'),
(10963, 28, '2023-03-19'),
(10964, 74, '2023-03-20'),
(10965, 55, '2023-03-20'),
(10966, 14, '2023-03-20'),
(10967, 79, '2023-03-23'),
(10968, 20, '2023-03-23'),
(10969, 15, '2023-03-23'),
(10970, 8, '2023-03-24'),
(10971, 26, '2023-03-24'),
(10972, 40, '2023-03-24'),
(10973, 40, '2023-03-24'),
(10974, 75, '2023-03-25'),
(10975, 10, '2023-03-25'),
(10976, 35, '2023-03-25'),
(10977, 24, '2023-03-26'),
(10978, 50, '2023-03-26'),
(10979, 20, '2023-03-26'),
(10980, 24, '2023-03-27'),
(10981, 34, '2023-03-27'),
(10982, 10, '2023-03-27'),
(10983, 71, '2023-03-27'),
(10984, 71, '2023-03-30'),
(10985, 37, '2023-03-30'),
(10986, 54, '2023-03-30'),
(10987, 19, '2023-03-31'),
(10988, 65, '2023-03-31'),
(10989, 61, '2023-03-31'),
(10990, 20, '2023-04-01'),
```

```
(10991, 63, '2023-04-01'),
(10992, 77, '2023-04-01'),
(10993, 24, '2023-04-01'),
(10994, 83, '2023-04-02'),
(10995, 58, '2023-04-02'),
(10996, 63, '2023-04-02'),
(10997, 46, '2023-04-03'),
(10998, 91, '2023-04-03'),
(10999, 56, '2023-04-03'),
(11000, 65, '2023-04-06'),
(11001, 24, '2023-04-06'),
(11002, 71, '2023-04-06'),
(11003, 78, '2023-04-06'),
(11004, 50, '2023-04-07'),
(11005, 90, '2023-04-07'),
(11006, 32, '2023-04-07'),
(11007, 60, '2023-04-08'),
(11008, 20, '2023-04-08'),
(11009, 30, '2023-04-08'),
(11010, 66, '2023-04-09'),
(11011, 1, '2023-04-09'),
(11012, 25, '2023-04-09'),
(11013, 69, '2023-04-09'),
(11014, 47, '2023-04-10'),
(11015, 70, '2023-04-10'),
(11016, 4, '2023-04-10'),
(11017, 20, '2023-04-13'),
(11018, 48, '2023-04-13'),
(11019, 64, '2023-04-13'),
(11020, 56, '2023-04-14'),
(11021, 63, '2023-04-14'),
(11022, 34, '2023-04-14'),
(11023, 11, '2023-04-14'),
(11024, 19, '2023-04-15'),
(11025, 87, '2023-04-15'),
(11026, 27, '2023-04-15'),
(11027, 10, '2023-04-16'),
(11028, 39, '2023-04-16'),
(11029, 14, '2023-04-16'),
(11030, 71, '2023-04-17'),
(11031, 71, '2023-04-17'),
(11032, 89, '2023-04-17'),
(11033, 68, '2023-04-17'),
(11034, 55, '2023-04-20'),
(11035, 76, '2023-04-20'),
(11036, 17, '2023-04-20'),
(11037, 30, '2023-04-21'),
(11038, 76, '2023-04-21'),
(11039, 47, '2023-04-21'),
(11040, 32, '2023-04-22'),
(11041, 14, '2023-04-22'),
(11042, 15, '2023-04-22'),
(11043, 74, '2023-04-22'),
(11044, 91, '2023-04-23'),
(11045, 10, '2023-04-23'),
(11046, 86, '2023-04-23'),
(11047, 19, '2023-04-24'),
(11048, 10, '2023-04-24'),
(11049, 31, '2023-04-24'),
(11050, 24, '2023-04-27'),
(11051, 41, '2023-04-27'),
(11052, 34, '2023-04-27'),
(11053, 59, '2023-04-27'),
(11054, 12, '2023-04-28'),
(11055, 35, '2023-04-28'),
(11056, 19, '2023-04-28'),
(11057, 53, '2023-04-29'),
(11058, 6, '2023-04-29'),
(11059, 67, '2023-04-29'),
(11061, 32, '2023-04-30'),
(11062, 66, '2023-04-30'),
(11063, 37, '2023-04-30'),
(11064, 71, '2023-05-01'),
(11065, 46, '2023-05-01'),
(11066, 89, '2023-05-01'),
(11067, 17, '2023-05-04'),
(11068, 62, '2023-05-04'),
(11069, 80, '2023-05-04'),
(11070, 44, '2023-05-05'),
(11071, 46, '2023-05-05'),
(11072, 20, '2023-05-05'),
(11073, 58, '2023-05-05'),
(11074, 73, '2023-05-06'),
(11075, 68, '2023-05-06'),
(11076, 9, '2023-05-06'),
(11077, 65, '2023-05-06');
```

If you write above statement in your SQL Shell, you will see that it returns `INSERT 0 830`. Which means that we have inserted 830 rows in our `orders` table.

Order_Details

The following SQL statement will create a table named `order_details`:

CREATE TABLE order_details

```
CREATE TABLE order_details (
  order_detail_id SERIAL NOT NULL PRIMARY KEY,
  order_id INT,
```

```
  product_id INT,
  quantity INT
);
```

INSERT INTO order_details

The following SQL statement will fill the `order_details` table with content:

```
INSERT INTO order_details (order_id, product_id, quantity)
VALUES
  (10248, 11, 12),
  (10248, 42, 10),
  (10248, 72, 5),
  (10249, 14, 9),
  (10249, 51, 40),
  (10250, 41, 10),
  (10250, 51, 35),
  (10250, 65, 15),
  (10251, 22, 6),
  (10251, 57, 15),
  (10251, 65, 20),
  (10252, 20, 40),
  (10252, 33, 25),
  (10252, 60, 40),
  (10253, 31, 20),
  (10253, 39, 42),
  (10253, 49, 40),
  (10254, 24, 15),
  (10254, 55, 21),
  (10254, 74, 21),
  (10255, 2, 20),
  (10255, 16, 35),
  (10255, 36, 25),
  (10255, 59, 30),
  (10256, 53, 15),
  (10256, 77, 12),
  (10257, 27, 25),
  (10257, 39, 6),
  (10257, 77, 15),
  (10258, 2, 50),
  (10258, 5, 65),
  (10258, 32, 6),
  (10259, 21, 10),
  (10259, 37, 1),
  (10260, 41, 16),
  (10260, 57, 50),
  (10260, 62, 15),
  (10260, 70, 21),
  (10261, 21, 20),
  (10261, 35, 20),
  (10262, 5, 12),
  (10262, 7, 15),
  (10262, 56, 2),
  (10263, 16, 60),
  (10263, 24, 28),
  (10263, 30, 60),
  (10263, 74, 36),
  (10264, 2, 35),
  (10264, 41, 25),
  (10265, 17, 30),
  (10265, 70, 20),
  (10266, 12, 12),
  (10267, 40, 50),
  (10267, 59, 70),
  (10267, 76, 15),
  (10268, 29, 10),
  (10268, 72, 4),
  (10269, 33, 60),
  (10269, 72, 20),
  (10270, 36, 30),
  (10270, 43, 25),
  (10271, 33, 24),
  (10272, 20, 6),
  (10272, 31, 40),
  (10272, 72, 24),
  (10273, 10, 24),
  (10273, 31, 15),
  (10273, 33, 20),
  (10273, 40, 60),
  (10273, 76, 33),
  (10274, 71, 20),
  (10274, 72, 7),
  (10275, 24, 12),
  (10275, 59, 6),
  (10276, 10, 15),
  (10276, 13, 10),
  (10277, 28, 20),
  (10277, 62, 12),
  (10278, 44, 16),
  (10278, 59, 15),
  (10278, 63, 8),
  (10278, 73, 25),
  (10279, 17, 15),
  (10280, 24, 12),
  (10280, 55, 20),
  (10280, 75, 30),
  (10281, 19, 1),
  (10281, 24, 6),
  (10281, 35, 4),
  (10282, 30, 6),
  (10282, 57, 2),
  (10283, 15, 20),
  (10283, 19, 18),
  (10283, 60, 35),
  (10283, 72, 3),
  (10284, 27, 15),
  (10284, 44, 21),
  (10284, 60, 20),
  (10284, 67, 5),
  (10285, 1, 45),
```

```
(10285, 40, 40),
(10285, 53, 36),
(10286, 35, 100),
(10286, 62, 40),
(10287, 16, 40),
(10287, 34, 20),
(10287, 46, 15),
(10288, 54, 10),
(10288, 68, 3),
(10289, 3, 30),
(10289, 64, 9),
(10290, 5, 20),
(10290, 29, 15),
(10290, 49, 15),
(10290, 77, 10),
(10291, 13, 20),
(10291, 44, 24),
(10291, 51, 2),
(10292, 20, 20),
(10293, 18, 12),
(10293, 24, 10),
(10293, 63, 5),
(10293, 75, 6),
(10294, 1, 18),
(10294, 17, 15),
(10294, 43, 15),
(10294, 60, 21),
(10294, 75, 6),
(10295, 56, 4),
(10296, 11, 12),
(10296, 16, 30),
(10296, 69, 15),
(10297, 39, 60),
(10297, 72, 20),
(10298, 2, 40),
(10298, 36, 40),
(10298, 59, 30),
(10298, 62, 15),
(10299, 19, 15),
(10299, 70, 20),
(10300, 66, 30),
(10300, 68, 20),
(10301, 40, 10),
(10301, 56, 20),
(10302, 17, 40),
(10302, 28, 28),
(10302, 43, 12),
(10303, 40, 40),
(10303, 65, 30),
(10303, 68, 15),
(10304, 49, 30),
(10304, 59, 10),
(10304, 71, 2),
(10305, 18, 25),
(10305, 29, 25),
(10305, 39, 30),
(10306, 30, 10),
(10306, 53, 10),
(10306, 54, 5),
(10307, 62, 10),
(10307, 68, 3),
(10308, 69, 1),
(10308, 70, 5),
(10309, 4, 20),
(10309, 6, 30),
(10309, 42, 2),
(10309, 43, 20),
(10309, 71, 3),
(10310, 16, 10),
(10310, 62, 5),
(10311, 42, 6),
(10311, 69, 7),
(10312, 28, 4),
(10312, 43, 24),
(10312, 53, 20),
(10312, 75, 10),
(10313, 36, 12),
(10314, 32, 40),
(10314, 58, 30),
(10314, 62, 25),
(10315, 34, 14),
(10315, 70, 30),
(10316, 41, 10),
(10316, 62, 70),
(10317, 1, 20),
(10318, 41, 20),
(10318, 76, 6),
(10319, 17, 8),
(10319, 28, 14),
(10319, 76, 30),
(10320, 71, 30),
(10321, 35, 10),
(10322, 52, 20),
(10323, 15, 5),
(10323, 25, 4),
(10323, 39, 4),
(10324, 16, 21),
(10324, 35, 70),
(10324, 46, 30),
(10324, 59, 40),
(10324, 63, 80),
(10325, 6, 6),
(10325, 13, 12),
(10325, 14, 9),
(10325, 31, 4),
(10325, 72, 40),
(10326, 4, 24),
(10326, 57, 16),
(10326, 75, 50),
(10327, 2, 25),
(10327, 11, 50),
(10327, 30, 35),
(10327, 58, 30),
(10328, 59, 9),
(10328, 65, 40),
(10328, 68, 10),
(10329, 19, 10),
(10329, 30, 8),
(10329, 38, 20),
(10329, 56, 12),
(10330, 26, 50),
(10330, 72, 25),
(10331, 54, 15),
(10332, 18, 40),
(10332, 42, 10),
(10332, 47, 16),
(10333, 14, 10),
(10333, 21, 10),
(10333, 71, 40),
(10334, 52, 8),
(10334, 68, 10),
(10335, 2, 7),
```

```
(10335, 31, 25),
(10335, 32, 6),
(10335, 51, 48),
(10336, 4, 18),
(10337, 23, 40),
(10337, 26, 24),
(10337, 36, 20),
(10337, 37, 28),
(10337, 72, 25),
(10338, 17, 20),
(10338, 30, 15),
(10339, 4, 10),
(10339, 17, 70),
(10339, 62, 28),
(10340, 18, 20),
(10340, 41, 12),
(10340, 43, 40),
(10341, 33, 8),
(10341, 59, 9),
(10342, 2, 24),
(10342, 31, 56),
(10342, 36, 40),
(10342, 55, 40),
(10343, 64, 50),
(10343, 68, 4),
(10343, 76, 15),
(10344, 4, 35),
(10344, 8, 70),
(10345, 8, 70),
(10345, 19, 80),
(10345, 42, 9),
(10346, 17, 36),
(10346, 56, 20),
(10347, 25, 10),
(10347, 39, 50),
(10347, 40, 4),
(10347, 75, 6),
(10348, 1, 15),
(10348, 23, 25),
(10349, 54, 24),
(10350, 50, 15),
(10350, 69, 18),
(10351, 38, 20),
(10351, 41, 13),
(10351, 44, 77),
(10351, 65, 10),
(10352, 24, 10),
(10352, 54, 20),
(10353, 11, 12),
(10353, 38, 50),
(10354, 1, 12),
(10354, 29, 4),
(10355, 24, 25),
(10355, 57, 25),
(10356, 31, 30),
(10356, 55, 12),
(10356, 69, 20),
(10357, 10, 30),
(10357, 26, 16),
(10357, 60, 8),
(10358, 24, 10),
(10358, 34, 10),
(10358, 36, 20),
(10359, 16, 56),
(10359, 31, 70),
(10359, 60, 80),
(10360, 28, 30),
(10360, 29, 35),
(10360, 38, 10),
(10360, 49, 35),
(10360, 54, 28),
(10361, 39, 54),
(10361, 60, 55),
(10362, 25, 50),
(10362, 51, 20),
(10362, 54, 24),
(10363, 31, 20),
(10363, 75, 12),
(10363, 76, 12),
(10364, 69, 30),
(10364, 71, 5),
(10365, 11, 24),
(10366, 65, 5),
(10366, 77, 5),
(10367, 34, 36),
(10367, 54, 18),
(10367, 65, 15),
(10367, 77, 7),
(10368, 21, 5),
(10368, 28, 13),
(10368, 57, 25),
(10368, 64, 35),
(10369, 29, 20),
(10369, 56, 18),
(10370, 1, 15),
(10370, 64, 30),
(10370, 74, 20),
(10371, 36, 6),
(10372, 20, 12),
(10372, 38, 40),
(10372, 60, 70),
(10372, 72, 42),
(10373, 58, 80),
(10373, 71, 50),
(10374, 31, 30),
(10374, 58, 15),
(10375, 14, 15),
(10375, 54, 10),
(10376, 31, 42),
(10377, 28, 20),
(10377, 39, 20),
(10378, 71, 6),
(10379, 41, 8),
(10379, 63, 16),
(10379, 65, 20),
(10380, 30, 18),
(10380, 53, 20),
(10380, 60, 6),
(10380, 70, 30),
(10381, 74, 14),
(10382, 5, 32),
(10382, 18, 9),
(10382, 29, 14),
(10382, 33, 60),
(10382, 74, 50),
(10383, 13, 20),
(10383, 50, 15),
(10383, 56, 20),
(10384, 20, 28),
(10384, 60, 15),
(10385, 7, 10),
(10385, 60, 20),
```

```
(10385, 68, 8),
(10386, 24, 15),
(10386, 34, 10),
(10387, 24, 15),
(10387, 28, 6),
(10387, 59, 12),
(10387, 71, 15),
(10388, 45, 15),
(10388, 52, 20),
(10388, 53, 40),
(10389, 10, 16),
(10389, 55, 15),
(10389, 62, 20),
(10389, 70, 30),
(10390, 31, 60),
(10390, 35, 40),
(10390, 46, 45),
(10390, 72, 24),
(10391, 13, 18),
(10392, 69, 50),
(10393, 2, 25),
(10393, 14, 42),
(10393, 25, 7),
(10393, 26, 70),
(10393, 31, 32),
(10394, 13, 10),
(10394, 62, 10),
(10395, 46, 28),
(10395, 53, 70),
(10395, 69, 8),
(10396, 23, 40),
(10396, 71, 60),
(10396, 72, 21),
(10397, 21, 10),
(10397, 51, 18),
(10398, 35, 30),
(10398, 55, 120),
(10399, 68, 60),
(10399, 71, 30),
(10399, 76, 35),
(10399, 77, 14),
(10400, 29, 21),
(10400, 35, 35),
(10400, 49, 30),
(10401, 30, 18),
(10401, 56, 70),
(10401, 65, 20),
(10401, 71, 60),
(10402, 23, 60),
(10402, 63, 65),
(10403, 16, 21),
(10403, 48, 70),
(10404, 26, 30),
(10404, 42, 40),
(10404, 49, 30),
(10405, 3, 50),
(10406, 1, 10),
(10406, 21, 30),
(10406, 28, 42),
(10406, 36, 5),
(10406, 40, 2),
(10407, 11, 30),
(10407, 69, 15),
(10407, 71, 15),
(10408, 37, 10),
(10408, 54, 6),
(10408, 62, 35),
(10409, 14, 12),
(10409, 21, 12),
(10410, 33, 49),
(10410, 59, 16),
(10411, 41, 25),
(10411, 44, 40),
(10411, 59, 9),
(10412, 14, 20),
(10413, 1, 24),
(10413, 62, 40),
(10413, 76, 14),
(10414, 19, 18),
(10414, 33, 50),
(10415, 17, 2),
(10415, 33, 20),
(10416, 19, 20),
(10416, 53, 10),
(10416, 57, 20),
(10417, 38, 50),
(10417, 46, 2),
(10417, 68, 36),
(10417, 77, 35),
(10418, 2, 60),
(10418, 47, 55),
(10418, 61, 16),
(10418, 74, 15),
(10419, 60, 60),
(10419, 69, 20),
(10420, 9, 20),
(10420, 13, 2),
(10420, 70, 8),
(10420, 73, 20),
(10421, 19, 4),
(10421, 26, 30),
(10421, 53, 15),
(10421, 77, 10),
(10422, 26, 2),
(10423, 31, 14),
(10423, 59, 20),
(10424, 35, 60),
(10424, 38, 49),
(10424, 68, 30),
(10425, 55, 10),
(10425, 76, 20),
(10426, 56, 5),
(10426, 64, 7),
(10427, 14, 35),
(10428, 46, 20),
(10429, 50, 40),
(10429, 63, 35),
(10430, 17, 45),
(10430, 21, 50),
(10430, 56, 30),
(10430, 59, 70),
(10431, 17, 50),
(10431, 40, 50),
(10431, 47, 30),
(10432, 26, 10),
(10432, 54, 40),
(10433, 56, 28),
(10434, 11, 6),
(10434, 76, 18),
(10435, 2, 10),
(10435, 22, 12),
(10435, 72, 10),
```

```
(10436, 46, 5),
(10436, 56, 40),
(10436, 64, 30),
(10436, 75, 24),
(10437, 53, 15),
(10438, 19, 15),
(10438, 34, 20),
(10438, 57, 15),
(10439, 12, 15),
(10439, 16, 16),
(10439, 64, 6),
(10439, 74, 30),
(10440, 2, 45),
(10440, 16, 49),
(10440, 29, 24),
(10440, 61, 90),
(10441, 27, 50),
(10442, 11, 30),
(10442, 54, 80),
(10442, 66, 60),
(10443, 11, 6),
(10443, 28, 12),
(10444, 17, 10),
(10444, 26, 15),
(10444, 35, 8),
(10444, 41, 30),
(10445, 39, 6),
(10445, 54, 15),
(10446, 19, 12),
(10446, 24, 20),
(10446, 31, 3),
(10446, 52, 15),
(10447, 19, 40),
(10447, 65, 35),
(10447, 71, 2),
(10448, 26, 6),
(10448, 40, 20),
(10449, 10, 14),
(10449, 52, 20),
(10449, 62, 35),
(10450, 10, 20),
(10450, 54, 6),
(10451, 55, 120),
(10451, 64, 35),
(10451, 65, 28),
(10451, 77, 55),
(10452, 28, 15),
(10452, 44, 100),
(10453, 48, 15),
(10453, 70, 25),
(10454, 16, 20),
(10454, 33, 20),
(10454, 46, 10),
(10455, 39, 20),
(10455, 53, 50),
(10455, 61, 25),
(10455, 71, 30),
(10456, 21, 40),
(10456, 49, 21),
(10457, 59, 36),
(10458, 26, 30),
(10458, 28, 30),
(10458, 43, 20),
(10458, 56, 15),
(10458, 71, 50),
(10459, 7, 16),
(10459, 46, 20),
(10459, 72, 40),
(10460, 68, 21),
(10460, 75, 4),
(10461, 21, 40),
(10461, 30, 28),
(10461, 55, 60),
(10462, 13, 1),
(10462, 23, 21),
(10463, 19, 21),
(10463, 42, 50),
(10464, 4, 16),
(10464, 43, 3),
(10464, 56, 30),
(10464, 60, 20),
(10465, 24, 25),
(10465, 29, 18),
(10465, 40, 20),
(10465, 45, 30),
(10465, 50, 25),
(10466, 11, 10),
(10466, 46, 5),
(10467, 24, 28),
(10467, 25, 12),
(10468, 30, 8),
(10468, 43, 15),
(10469, 2, 40),
(10469, 16, 35),
(10469, 44, 2),
(10470, 18, 30),
(10470, 23, 15),
(10470, 64, 8),
(10471, 7, 30),
(10471, 56, 20),
(10472, 24, 80),
(10472, 51, 18),
(10473, 33, 12),
(10473, 71, 12),
(10474, 14, 12),
(10474, 28, 18),
(10474, 40, 21),
(10474, 75, 10),
(10475, 31, 35),
(10475, 66, 60),
(10475, 76, 42),
(10476, 55, 2),
(10476, 70, 12),
(10477, 1, 15),
(10477, 21, 21),
(10477, 39, 20),
(10478, 10, 20),
(10479, 38, 30),
(10479, 53, 28),
(10479, 59, 60),
(10479, 64, 30),
(10480, 47, 30),
(10480, 59, 12),
(10481, 49, 24),
(10481, 60, 40),
(10482, 40, 10),
(10483, 34, 35),
(10483, 77, 30),
(10484, 21, 14),
(10484, 40, 10),
(10484, 51, 3),
(10485, 2, 20),
```

```
(10485, 3, 20),
(10485, 55, 30),
(10485, 70, 60),
(10486, 11, 5),
(10486, 51, 25),
(10486, 74, 16),
(10487, 19, 5),
(10487, 26, 30),
(10487, 54, 24),
(10488, 59, 30),
(10488, 73, 20),
(10489, 11, 15),
(10489, 16, 18),
(10490, 59, 60),
(10490, 68, 30),
(10490, 75, 36),
(10491, 44, 15),
(10491, 77, 7),
(10492, 25, 60),
(10492, 42, 20),
(10493, 65, 15),
(10493, 66, 10),
(10493, 69, 10),
(10494, 56, 30),
(10495, 23, 10),
(10495, 41, 20),
(10495, 77, 5),
(10496, 31, 20),
(10497, 56, 14),
(10497, 72, 25),
(10497, 77, 25),
(10498, 24, 14),
(10498, 40, 5),
(10498, 42, 30),
(10499, 28, 20),
(10499, 49, 25),
(10500, 15, 12),
(10500, 28, 8),
(10501, 54, 20),
(10502, 45, 21),
(10502, 53, 6),
(10502, 67, 30),
(10503, 14, 70),
(10503, 65, 20),
(10504, 2, 12),
(10504, 21, 12),
(10504, 53, 10),
(10504, 61, 25),
(10505, 62, 3),
(10506, 25, 18),
(10506, 70, 14),
(10507, 43, 15),
(10507, 48, 15),
(10508, 13, 10),
(10508, 39, 10),
(10509, 28, 3),
(10510, 29, 36),
(10510, 75, 36),
(10511, 4, 50),
(10511, 7, 50),
(10511, 8, 10),
(10512, 24, 10),
(10512, 46, 9),
(10512, 47, 6),
(10512, 60, 12),
(10513, 21, 40),
(10513, 32, 50),
(10513, 61, 15),
(10514, 20, 39),
(10514, 28, 35),
(10514, 56, 70),
(10514, 65, 39),
(10514, 75, 50),
(10515, 9, 16),
(10515, 16, 50),
(10515, 27, 120),
(10515, 33, 16),
(10515, 60, 84),
(10516, 18, 25),
(10516, 41, 80),
(10516, 42, 20),
(10517, 52, 6),
(10517, 59, 4),
(10517, 70, 6),
(10518, 24, 5),
(10518, 38, 15),
(10518, 44, 9),
(10519, 10, 16),
(10519, 56, 40),
(10519, 60, 10),
(10520, 24, 8),
(10520, 53, 5),
(10521, 35, 3),
(10521, 41, 10),
(10521, 68, 6),
(10522, 1, 40),
(10522, 8, 24),
(10522, 30, 20),
(10522, 40, 25),
(10523, 17, 25),
(10523, 20, 15),
(10523, 37, 18),
(10523, 41, 6),
(10524, 10, 2),
(10524, 30, 10),
(10524, 43, 60),
(10524, 54, 15),
(10525, 36, 30),
(10525, 40, 15),
(10526, 1, 8),
(10526, 13, 10),
(10526, 56, 30),
(10527, 4, 50),
(10527, 36, 30),
(10528, 11, 3),
(10528, 33, 8),
(10528, 72, 9),
(10529, 55, 14),
(10529, 68, 20),
(10529, 69, 10),
(10530, 17, 40),
(10530, 43, 25),
(10530, 61, 20),
(10530, 76, 50),
(10531, 59, 2),
(10532, 30, 15),
(10532, 66, 24),
(10533, 4, 50),
(10533, 72, 24),
(10533, 73, 24),
(10534, 30, 10),
(10534, 40, 10),
```

```
(10534, 54, 10),
(10535, 11, 50),
(10535, 40, 10),
(10535, 57, 5),
(10535, 59, 15),
(10536, 12, 15),
(10536, 31, 20),
(10536, 33, 30),
(10536, 60, 35),
(10537, 31, 30),
(10537, 51, 6),
(10537, 58, 20),
(10537, 72, 21),
(10537, 73, 9),
(10538, 70, 7),
(10538, 72, 1),
(10539, 13, 8),
(10539, 21, 15),
(10539, 33, 15),
(10539, 49, 6),
(10540, 3, 60),
(10540, 26, 40),
(10540, 38, 30),
(10540, 68, 35),
(10541, 24, 35),
(10541, 38, 4),
(10541, 65, 36),
(10541, 71, 9),
(10542, 11, 15),
(10542, 54, 24),
(10543, 12, 30),
(10543, 23, 70),
(10544, 28, 7),
(10544, 67, 7),
(10545, 11, 10),
(10546, 7, 10),
(10546, 35, 30),
(10546, 62, 40),
(10547, 32, 24),
(10547, 36, 60),
(10548, 34, 10),
(10548, 41, 14),
(10549, 31, 55),
(10549, 45, 100),
(10549, 51, 48),
(10550, 17, 8),
(10550, 19, 10),
(10550, 21, 6),
(10550, 61, 10),
(10551, 16, 40),
(10551, 35, 20),
(10551, 44, 40),
(10552, 69, 18),
(10552, 75, 30),
(10553, 11, 15),
(10553, 16, 14),
(10553, 22, 24),
(10553, 31, 30),
(10553, 35, 6),
(10554, 16, 30),
(10554, 23, 20),
(10554, 62, 20),
(10554, 77, 10),
(10555, 14, 30),
(10555, 19, 35),
(10555, 24, 18),
(10555, 51, 20),
(10555, 56, 40),
(10556, 72, 24),
(10557, 64, 30),
(10557, 75, 20),
(10558, 47, 25),
(10558, 51, 20),
(10558, 52, 30),
(10558, 53, 18),
(10558, 73, 3),
(10559, 41, 12),
(10559, 55, 18),
(10560, 30, 20),
(10560, 62, 15),
(10561, 44, 10),
(10561, 51, 50),
(10562, 33, 20),
(10562, 62, 10),
(10563, 36, 25),
(10563, 52, 70),
(10564, 17, 16),
(10564, 31, 6),
(10564, 55, 25),
(10565, 24, 25),
(10565, 64, 18),
(10566, 11, 35),
(10566, 18, 18),
(10566, 76, 10),
(10567, 31, 60),
(10567, 51, 3),
(10567, 59, 40),
(10568, 10, 5),
(10569, 31, 35),
(10569, 76, 30),
(10570, 11, 15),
(10570, 56, 60),
(10571, 14, 11),
(10571, 42, 28),
(10572, 16, 12),
(10572, 32, 10),
(10572, 40, 50),
(10572, 75, 15),
(10573, 17, 18),
(10573, 34, 40),
(10573, 53, 25),
(10574, 33, 14),
(10574, 40, 2),
(10574, 62, 10),
(10574, 64, 6),
(10575, 59, 12),
(10575, 63, 6),
(10575, 72, 30),
(10575, 76, 10),
(10576, 1, 10),
(10576, 31, 20),
(10576, 44, 21),
(10577, 39, 10),
(10577, 75, 20),
(10577, 77, 18),
(10578, 35, 20),
(10578, 57, 6),
(10579, 15, 10),
(10579, 75, 21),
(10580, 14, 15),
(10580, 41, 9),
(10580, 65, 30),
```

```
(10581, 75, 50),
(10582, 57, 4),
(10582, 76, 14),
(10583, 29, 10),
(10583, 60, 24),
(10583, 69, 10),
(10584, 31, 50),
(10585, 47, 15),
(10586, 52, 4),
(10587, 26, 6),
(10587, 35, 20),
(10587, 77, 20),
(10588, 18, 40),
(10588, 42, 100),
(10589, 35, 4),
(10590, 1, 20),
(10590, 77, 60),
(10591, 3, 14),
(10591, 7, 10),
(10591, 54, 50),
(10592, 15, 25),
(10592, 26, 5),
(10593, 20, 21),
(10593, 69, 20),
(10593, 76, 4),
(10594, 52, 24),
(10594, 58, 30),
(10595, 35, 30),
(10595, 61, 120),
(10595, 69, 65),
(10596, 56, 5),
(10596, 63, 24),
(10596, 75, 30),
(10597, 24, 35),
(10597, 57, 20),
(10597, 65, 12),
(10598, 27, 50),
(10598, 71, 9),
(10599, 62, 10),
(10600, 54, 4),
(10600, 73, 30),
(10601, 13, 60),
(10601, 59, 35),
(10602, 77, 5),
(10603, 22, 48),
(10603, 49, 25),
(10604, 48, 6),
(10604, 76, 10),
(10605, 16, 30),
(10605, 59, 20),
(10605, 60, 70),
(10605, 71, 15),
(10606, 4, 20),
(10606, 55, 20),
(10606, 62, 10),
(10607, 7, 45),
(10607, 17, 100),
(10607, 33, 14),
(10607, 40, 42),
(10607, 72, 12),
(10608, 56, 28),
(10609, 1, 3),
(10609, 10, 10),
(10609, 21, 6),
(10610, 36, 21),
(10611, 1, 6),
(10611, 2, 10),
(10611, 60, 15),
(10612, 10, 70),
(10612, 36, 55),
(10612, 49, 18),
(10612, 60, 40),
(10612, 76, 80),
(10613, 13, 8),
(10613, 75, 40),
(10614, 11, 14),
(10614, 21, 8),
(10614, 39, 5),
(10615, 55, 5),
(10616, 38, 15),
(10616, 56, 14),
(10616, 70, 15),
(10616, 71, 15),
(10617, 59, 30),
(10618, 6, 70),
(10618, 56, 20),
(10618, 68, 15),
(10619, 21, 42),
(10619, 22, 40),
(10620, 24, 5),
(10620, 52, 5),
(10621, 19, 5),
(10621, 23, 10),
(10621, 70, 20),
(10621, 71, 15),
(10622, 2, 20),
(10622, 68, 18),
(10623, 14, 21),
(10623, 19, 15),
(10623, 21, 25),
(10623, 24, 3),
(10623, 35, 30),
(10624, 28, 10),
(10624, 29, 6),
(10624, 44, 10),
(10625, 14, 3),
(10625, 42, 5),
(10625, 60, 10),
(10626, 53, 12),
(10626, 60, 20),
(10626, 71, 20),
(10627, 62, 15),
(10627, 73, 35),
(10628, 1, 25),
(10629, 29, 20),
(10629, 64, 9),
(10630, 55, 12),
(10630, 76, 35),
(10631, 75, 8),
(10632, 2, 30),
(10632, 33, 20),
(10633, 12, 36),
(10633, 13, 13),
(10633, 26, 35),
(10633, 62, 80),
(10634, 7, 35),
(10634, 18, 50),
(10634, 51, 15),
(10634, 75, 2),
(10635, 4, 10),
(10635, 5, 15),
(10635, 22, 40),
```

```
(10636, 4, 25),
(10636, 58, 6),
(10637, 11, 10),
(10637, 50, 25),
(10637, 56, 60),
(10638, 45, 20),
(10638, 65, 21),
(10638, 72, 60),
(10639, 18, 8),
(10640, 69, 20),
(10640, 70, 15),
(10641, 2, 50),
(10641, 40, 60),
(10642, 21, 30),
(10642, 61, 20),
(10643, 28, 15),
(10643, 39, 21),
(10643, 46, 2),
(10644, 18, 4),
(10644, 43, 20),
(10644, 46, 21),
(10645, 18, 20),
(10645, 36, 15),
(10646, 1, 15),
(10646, 10, 18),
(10646, 71, 30),
(10646, 77, 35),
(10647, 19, 30),
(10647, 39, 20),
(10648, 22, 15),
(10648, 24, 15),
(10649, 28, 20),
(10649, 72, 15),
(10650, 30, 30),
(10650, 53, 25),
(10650, 54, 30),
(10651, 19, 12),
(10651, 22, 20),
(10652, 30, 2),
(10652, 42, 20),
(10653, 16, 30),
(10653, 60, 20),
(10654, 4, 12),
(10654, 39, 20),
(10654, 54, 6),
(10655, 41, 20),
(10656, 14, 3),
(10656, 44, 28),
(10656, 47, 6),
(10657, 15, 50),
(10657, 41, 24),
(10657, 46, 45),
(10657, 47, 10),
(10657, 56, 45),
(10657, 60, 30),
(10658, 21, 60),
(10658, 40, 70),
(10658, 60, 55),
(10658, 77, 70),
(10659, 31, 20),
(10659, 40, 24),
(10659, 70, 40),
(10660, 20, 21),
(10661, 39, 3),
(10661, 58, 49),
(10662, 68, 10),
(10663, 40, 30),
(10663, 42, 30),
(10663, 51, 20),
(10664, 10, 24),
(10664, 56, 12),
(10664, 65, 15),
(10665, 51, 20),
(10665, 59, 1),
(10665, 76, 10),
(10666, 29, 36),
(10666, 65, 10),
(10667, 69, 45),
(10667, 71, 14),
(10668, 31, 8),
(10668, 55, 4),
(10668, 64, 15),
(10669, 36, 30),
(10670, 23, 32),
(10670, 46, 60),
(10670, 67, 25),
(10670, 73, 50),
(10670, 75, 25),
(10671, 16, 10),
(10671, 62, 10),
(10671, 65, 12),
(10672, 38, 15),
(10672, 71, 12),
(10673, 16, 3),
(10673, 42, 6),
(10673, 43, 6),
(10674, 23, 5),
(10675, 14, 30),
(10675, 53, 10),
(10675, 58, 30),
(10676, 10, 2),
(10676, 19, 7),
(10676, 44, 21),
(10677, 26, 30),
(10677, 33, 8),
(10678, 12, 100),
(10678, 33, 30),
(10678, 41, 120),
(10678, 54, 30),
(10679, 59, 12),
(10680, 16, 50),
(10680, 31, 20),
(10680, 42, 40),
(10681, 19, 30),
(10681, 21, 12),
(10681, 64, 28),
(10682, 33, 30),
(10682, 66, 4),
(10682, 75, 30),
(10683, 52, 9),
(10684, 40, 20),
(10684, 47, 40),
(10684, 60, 30),
(10685, 10, 20),
(10685, 41, 4),
(10685, 47, 15),
(10686, 17, 30),
(10686, 26, 15),
(10687, 9, 50),
(10687, 29, 10),
(10687, 36, 6),
(10688, 10, 18),
```

```
(10688, 28, 60),
(10688, 34, 14),
(10689, 1, 35),
(10690, 56, 20),
(10690, 77, 30),
(10691, 1, 30),
(10691, 29, 40),
(10691, 43, 40),
(10691, 44, 24),
(10691, 62, 48),
(10692, 63, 20),
(10693, 9, 6),
(10693, 54, 60),
(10693, 69, 30),
(10693, 73, 15),
(10694, 7, 90),
(10694, 59, 25),
(10694, 70, 50),
(10695, 8, 10),
(10695, 12, 4),
(10695, 24, 20),
(10696, 17, 20),
(10696, 46, 18),
(10697, 19, 7),
(10697, 35, 9),
(10697, 58, 30),
(10697, 70, 30),
(10698, 11, 15),
(10698, 17, 8),
(10698, 29, 12),
(10698, 65, 65),
(10698, 70, 8),
(10699, 47, 12),
(10700, 1, 5),
(10700, 34, 12),
(10700, 68, 40),
(10700, 71, 60),
(10701, 59, 42),
(10701, 71, 20),
(10701, 76, 35),
(10702, 3, 6),
(10702, 76, 15),
(10703, 2, 5),
(10703, 59, 35),
(10703, 73, 35),
(10704, 4, 6),
(10704, 24, 35),
(10704, 48, 24),
(10705, 31, 20),
(10705, 32, 4),
(10706, 16, 20),
(10706, 43, 24),
(10706, 59, 8),
(10707, 55, 21),
(10707, 57, 40),
(10707, 70, 28),
(10708, 5, 4),
(10708, 36, 5),
(10709, 8, 40),
(10709, 51, 28),
(10709, 60, 10),
(10710, 19, 5),
(10710, 47, 5),
(10711, 19, 12),
(10711, 41, 42),
(10711, 53, 120),
(10712, 53, 3),
(10712, 56, 30),
(10713, 10, 18),
(10713, 26, 30),
(10713, 45, 110),
(10713, 46, 24),
(10714, 2, 30),
(10714, 17, 27),
(10714, 47, 50),
(10714, 56, 18),
(10714, 58, 12),
(10715, 10, 21),
(10715, 71, 30),
(10716, 21, 5),
(10716, 51, 7),
(10716, 61, 10),
(10717, 21, 32),
(10717, 54, 15),
(10717, 69, 25),
(10718, 12, 36),
(10718, 16, 20),
(10718, 36, 40),
(10718, 62, 20),
(10719, 18, 12),
(10719, 30, 3),
(10719, 54, 40),
(10720, 35, 21),
(10720, 71, 8),
(10721, 44, 50),
(10722, 2, 3),
(10722, 31, 50),
(10722, 68, 45),
(10722, 75, 42),
(10723, 26, 15),
(10724, 10, 16),
(10724, 61, 5),
(10725, 41, 12),
(10725, 52, 4),
(10725, 55, 6),
(10726, 4, 25),
(10726, 11, 5),
(10727, 17, 20),
(10727, 56, 10),
(10727, 59, 10),
(10728, 30, 15),
(10728, 40, 6),
(10728, 55, 12),
(10728, 60, 15),
(10729, 1, 50),
(10729, 21, 30),
(10729, 50, 40),
(10730, 16, 15),
(10730, 31, 3),
(10730, 65, 10),
(10731, 21, 40),
(10731, 51, 30),
(10732, 76, 20),
(10733, 14, 16),
(10733, 28, 20),
(10733, 52, 25),
(10734, 6, 30),
(10734, 30, 15),
(10734, 76, 20),
(10735, 61, 20),
(10735, 77, 2),
(10736, 65, 40),
```

```
(10736, 75, 20),
(10737, 13, 4),
(10737, 41, 12),
(10738, 16, 3),
(10739, 36, 6),
(10739, 52, 18),
(10740, 28, 5),
(10740, 35, 35),
(10740, 45, 40),
(10740, 56, 14),
(10741, 2, 15),
(10742, 3, 20),
(10742, 60, 50),
(10742, 72, 35),
(10743, 46, 28),
(10744, 40, 50),
(10745, 18, 24),
(10745, 44, 16),
(10745, 59, 45),
(10745, 72, 7),
(10746, 13, 6),
(10746, 42, 28),
(10746, 62, 9),
(10746, 69, 40),
(10747, 31, 8),
(10747, 41, 35),
(10747, 63, 9),
(10747, 69, 30),
(10748, 23, 44),
(10748, 40, 40),
(10748, 56, 28),
(10749, 56, 15),
(10749, 59, 6),
(10749, 76, 10),
(10750, 14, 5),
(10750, 45, 40),
(10750, 59, 25),
(10751, 26, 12),
(10751, 30, 30),
(10751, 50, 20),
(10751, 73, 15),
(10752, 1, 8),
(10752, 69, 3),
(10753, 45, 4),
(10753, 74, 5),
(10754, 40, 3),
(10755, 47, 30),
(10755, 56, 30),
(10755, 57, 14),
(10755, 69, 25),
(10756, 18, 21),
(10756, 36, 20),
(10756, 68, 6),
(10756, 69, 20),
(10757, 34, 30),
(10757, 59, 7),
(10757, 62, 30),
(10757, 64, 24),
(10758, 26, 20),
(10758, 52, 60),
(10758, 70, 40),
(10759, 32, 10),
(10760, 25, 12),
(10760, 27, 40),
(10760, 43, 30),
(10761, 25, 35),
(10761, 75, 18),
(10762, 39, 16),
(10762, 47, 30),
(10762, 51, 28),
(10762, 56, 60),
(10763, 21, 40),
(10763, 22, 6),
(10763, 24, 20),
(10764, 3, 20),
(10764, 39, 130),
(10765, 65, 80),
(10766, 2, 40),
(10766, 7, 35),
(10766, 68, 40),
(10767, 42, 2),
(10768, 22, 4),
(10768, 31, 50),
(10768, 60, 15),
(10768, 71, 12),
(10769, 41, 30),
(10769, 52, 15),
(10769, 61, 20),
(10769, 62, 15),
(10770, 11, 15),
(10771, 71, 16),
(10772, 29, 18),
(10772, 59, 25),
(10773, 17, 33),
(10773, 31, 70),
(10773, 75, 7),
(10774, 31, 2),
(10774, 66, 50),
(10775, 10, 6),
(10775, 67, 3),
(10776, 31, 16),
(10776, 42, 12),
(10776, 45, 27),
(10776, 51, 120),
(10777, 42, 20),
(10778, 41, 10),
(10779, 16, 20),
(10779, 62, 20),
(10780, 70, 35),
(10780, 77, 15),
(10781, 54, 3),
(10781, 56, 20),
(10781, 74, 35),
(10782, 31, 1),
(10783, 31, 10),
(10783, 38, 5),
(10784, 36, 30),
(10784, 39, 2),
(10784, 72, 30),
(10785, 10, 10),
(10785, 75, 10),
(10786, 8, 30),
(10786, 30, 15),
(10786, 75, 42),
(10787, 2, 15),
(10787, 29, 20),
(10788, 19, 50),
(10788, 75, 40),
(10789, 18, 30),
(10789, 35, 15),
(10789, 63, 30),
(10789, 68, 18),
```

```
(10790, 7, 3),
(10790, 56, 20),
(10791, 29, 14),
(10791, 41, 20),
(10792, 2, 10),
(10792, 54, 3),
(10792, 68, 15),
(10793, 41, 14),
(10793, 52, 8),
(10794, 14, 15),
(10794, 54, 6),
(10795, 16, 65),
(10795, 17, 35),
(10796, 26, 21),
(10796, 44, 10),
(10796, 64, 35),
(10796, 69, 24),
(10797, 11, 20),
(10798, 62, 2),
(10798, 72, 10),
(10799, 13, 20),
(10799, 24, 20),
(10799, 59, 25),
(10800, 11, 50),
(10800, 51, 10),
(10800, 54, 7),
(10801, 17, 40),
(10801, 29, 20),
(10802, 30, 25),
(10802, 51, 30),
(10802, 55, 60),
(10802, 62, 5),
(10803, 19, 24),
(10803, 25, 15),
(10803, 59, 15),
(10804, 10, 36),
(10804, 28, 24),
(10804, 49, 4),
(10805, 34, 10),
(10805, 38, 10),
(10806, 2, 20),
(10806, 65, 2),
(10806, 74, 15),
(10807, 40, 1),
(10808, 56, 20),
(10808, 76, 50),
(10809, 52, 20),
(10810, 13, 7),
(10810, 25, 5),
(10810, 70, 5),
(10811, 19, 15),
(10811, 23, 18),
(10811, 40, 30),
(10812, 31, 16),
(10812, 72, 40),
(10812, 77, 20),
(10813, 2, 12),
(10813, 46, 35),
(10814, 41, 20),
(10814, 43, 20),
(10814, 48, 8),
(10814, 61, 30),
(10815, 33, 16),
(10816, 38, 30),
(10816, 62, 20),
(10817, 26, 40),
(10817, 38, 30),
(10817, 40, 60),
(10817, 62, 25),
(10818, 32, 20),
(10818, 41, 20),
(10819, 43, 7),
(10819, 75, 20),
(10820, 56, 30),
(10821, 35, 20),
(10821, 51, 6),
(10822, 62, 3),
(10822, 70, 6),
(10823, 11, 20),
(10823, 57, 15),
(10823, 59, 40),
(10823, 77, 15),
(10824, 41, 12),
(10824, 70, 9),
(10825, 26, 12),
(10825, 53, 20),
(10826, 31, 35),
(10826, 57, 15),
(10827, 10, 15),
(10827, 39, 21),
(10828, 20, 5),
(10828, 38, 2),
(10829, 2, 10),
(10829, 8, 20),
(10829, 13, 10),
(10829, 60, 21),
(10830, 6, 6),
(10830, 39, 28),
(10830, 60, 30),
(10830, 68, 24),
(10831, 19, 2),
(10831, 35, 8),
(10831, 38, 8),
(10831, 43, 9),
(10832, 13, 3),
(10832, 25, 10),
(10832, 44, 16),
(10832, 64, 3),
(10833, 7, 20),
(10833, 31, 9),
(10833, 53, 9),
(10834, 29, 8),
(10834, 30, 20),
(10835, 59, 15),
(10835, 77, 2),
(10836, 22, 52),
(10836, 35, 6),
(10836, 57, 24),
(10836, 60, 60),
(10836, 64, 30),
(10837, 13, 6),
(10837, 40, 25),
(10837, 47, 40),
(10837, 76, 21),
(10838, 1, 4),
(10838, 18, 25),
(10838, 36, 50),
(10839, 58, 30),
(10839, 72, 15),
(10840, 25, 6),
(10840, 39, 10),
(10841, 10, 16),
```

```
(10841, 56, 30),
(10841, 59, 50),
(10841, 77, 15),
(10842, 11, 15),
(10842, 43, 5),
(10842, 68, 20),
(10842, 70, 12),
(10843, 51, 4),
(10844, 22, 35),
(10845, 23, 70),
(10845, 35, 25),
(10845, 42, 42),
(10845, 58, 60),
(10845, 64, 48),
(10846, 4, 21),
(10846, 70, 30),
(10846, 74, 20),
(10847, 1, 80),
(10847, 19, 12),
(10847, 37, 60),
(10847, 45, 36),
(10847, 60, 45),
(10847, 71, 55),
(10848, 5, 30),
(10848, 9, 3),
(10849, 3, 49),
(10849, 26, 18),
(10850, 25, 20),
(10850, 33, 4),
(10850, 70, 30),
(10851, 2, 5),
(10851, 25, 10),
(10851, 57, 10),
(10851, 59, 42),
(10852, 2, 15),
(10852, 17, 6),
(10852, 62, 50),
(10853, 18, 10),
(10854, 10, 100),
(10854, 13, 65),
(10855, 16, 50),
(10855, 31, 14),
(10855, 56, 24),
(10855, 65, 15),
(10856, 2, 20),
(10856, 42, 20),
(10857, 3, 30),
(10857, 26, 35),
(10857, 29, 10),
(10858, 7, 5),
(10858, 27, 10),
(10858, 70, 4),
(10859, 24, 40),
(10859, 54, 35),
(10859, 64, 30),
(10860, 51, 3),
(10860, 76, 20),
(10861, 17, 42),
(10861, 18, 20),
(10861, 21, 40),
(10861, 33, 35),
(10861, 62, 3),
(10862, 11, 25),
(10862, 52, 8),
(10863, 1, 20),
(10863, 58, 12),
(10864, 35, 4),
(10864, 67, 15),
(10865, 38, 60),
(10865, 39, 80),
(10866, 2, 21),
(10866, 24, 6),
(10866, 30, 40),
(10867, 53, 3),
(10868, 26, 20),
(10868, 35, 30),
(10868, 49, 42),
(10869, 1, 40),
(10869, 11, 10),
(10869, 23, 50),
(10869, 68, 20),
(10870, 35, 3),
(10870, 51, 2),
(10871, 6, 50),
(10871, 16, 12),
(10871, 17, 16),
(10872, 55, 10),
(10872, 62, 20),
(10872, 64, 15),
(10872, 65, 21),
(10873, 21, 20),
(10873, 28, 3),
(10874, 10, 10),
(10875, 19, 25),
(10875, 47, 21),
(10875, 49, 15),
(10876, 46, 21),
(10876, 64, 20),
(10877, 16, 30),
(10877, 18, 25),
(10878, 20, 20),
(10879, 40, 12),
(10879, 65, 10),
(10879, 76, 10),
(10880, 23, 30),
(10880, 61, 30),
(10880, 70, 50),
(10881, 73, 10),
(10882, 42, 25),
(10882, 49, 20),
(10882, 54, 32),
(10883, 24, 8),
(10884, 21, 40),
(10884, 56, 21),
(10884, 65, 12),
(10885, 2, 20),
(10885, 24, 12),
(10885, 70, 30),
(10885, 77, 25),
(10886, 10, 70),
(10886, 31, 35),
(10886, 77, 40),
(10887, 25, 5),
(10888, 2, 20),
(10888, 68, 18),
(10889, 11, 40),
(10889, 38, 40),
(10890, 17, 15),
(10890, 34, 10),
(10890, 41, 14),
(10891, 30, 15),
(10892, 59, 40),
```

```
(10893, 8, 30),
(10893, 24, 10),
(10893, 29, 24),
(10893, 30, 35),
(10893, 36, 20),
(10894, 13, 28),
(10894, 69, 50),
(10894, 75, 120),
(10895, 24, 110),
(10895, 39, 45),
(10895, 40, 91),
(10895, 60, 100),
(10896, 45, 15),
(10896, 56, 16),
(10897, 29, 80),
(10897, 30, 36),
(10898, 13, 5),
(10899, 39, 8),
(10900, 70, 3),
(10901, 41, 30),
(10901, 71, 30),
(10902, 55, 30),
(10902, 62, 6),
(10903, 13, 40),
(10903, 65, 21),
(10903, 68, 20),
(10904, 58, 15),
(10904, 62, 35),
(10905, 1, 20),
(10906, 61, 15),
(10907, 75, 14),
(10908, 7, 20),
(10908, 52, 14),
(10909, 7, 12),
(10909, 16, 15),
(10909, 41, 5),
(10910, 19, 12),
(10910, 49, 10),
(10910, 61, 5),
(10911, 1, 10),
(10911, 17, 12),
(10911, 67, 15),
(10912, 11, 40),
(10912, 29, 60),
(10913, 4, 30),
(10913, 33, 40),
(10913, 58, 15),
(10914, 71, 25),
(10915, 17, 10),
(10915, 33, 30),
(10915, 54, 10),
(10916, 16, 6),
(10916, 32, 6),
(10916, 57, 20),
(10917, 30, 1),
(10917, 60, 10),
(10918, 1, 60),
(10918, 60, 25),
(10919, 16, 24),
(10919, 25, 24),
(10919, 40, 20),
(10920, 50, 24),
(10921, 35, 10),
(10921, 63, 40),
(10922, 17, 15),
(10922, 24, 35),
(10923, 42, 10),
(10923, 43, 10),
(10923, 67, 24),
(10924, 10, 20),
(10924, 28, 30),
(10924, 75, 6),
(10925, 36, 25),
(10925, 52, 12),
(10926, 11, 2),
(10926, 13, 10),
(10926, 19, 7),
(10926, 72, 10),
(10927, 20, 5),
(10927, 52, 5),
(10927, 76, 20),
(10928, 47, 5),
(10928, 76, 5),
(10929, 21, 60),
(10929, 75, 49),
(10929, 77, 15),
(10930, 21, 36),
(10930, 27, 25),
(10930, 55, 25),
(10930, 58, 30),
(10931, 13, 42),
(10931, 57, 30),
(10932, 16, 30),
(10932, 62, 14),
(10932, 72, 16),
(10932, 75, 20),
(10933, 53, 2),
(10933, 61, 30),
(10934, 6, 20),
(10935, 1, 21),
(10935, 18, 4),
(10935, 23, 8),
(10936, 36, 30),
(10937, 28, 8),
(10937, 34, 20),
(10938, 13, 20),
(10938, 43, 24),
(10938, 60, 49),
(10938, 71, 35),
(10939, 2, 10),
(10939, 67, 40),
(10940, 7, 8),
(10940, 13, 20),
(10941, 31, 44),
(10941, 62, 30),
(10941, 68, 80),
(10941, 72, 50),
(10942, 49, 28),
(10943, 13, 15),
(10943, 22, 21),
(10943, 46, 15),
(10944, 11, 5),
(10944, 44, 18),
(10944, 56, 18),
(10945, 13, 20),
(10945, 31, 10),
(10946, 10, 25),
(10946, 24, 25),
(10946, 77, 40),
(10947, 59, 4),
(10948, 50, 9),
(10948, 51, 40),
```

```
(10948, 55, 4),
(10949, 6, 12),
(10949, 10, 30),
(10949, 17, 6),
(10949, 62, 60),
(10950, 4, 5),
(10951, 33, 15),
(10951, 41, 6),
(10951, 75, 50),
(10952, 6, 16),
(10952, 28, 2),
(10953, 20, 50),
(10953, 31, 50),
(10954, 16, 28),
(10954, 31, 25),
(10954, 45, 30),
(10954, 60, 24),
(10955, 75, 12),
(10956, 21, 12),
(10956, 47, 14),
(10956, 51, 8),
(10957, 30, 30),
(10957, 35, 40),
(10957, 64, 8),
(10958, 5, 20),
(10958, 7, 6),
(10958, 72, 5),
(10959, 75, 20),
(10960, 24, 10),
(10960, 41, 24),
(10961, 52, 6),
(10961, 76, 60),
(10962, 7, 45),
(10962, 13, 77),
(10962, 53, 20),
(10962, 69, 9),
(10962, 76, 44),
(10963, 60, 2),
(10964, 18, 6),
(10964, 38, 5),
(10964, 69, 10),
(10965, 51, 16),
(10966, 37, 8),
(10966, 56, 12),
(10966, 62, 12),
(10967, 19, 12),
(10967, 49, 40),
(10968, 12, 30),
(10968, 24, 30),
(10968, 64, 4),
(10969, 46, 9),
(10970, 52, 40),
(10971, 29, 14),
(10972, 17, 6),
(10972, 33, 7),
(10973, 26, 5),
(10973, 41, 6),
(10973, 75, 10),
(10974, 63, 10),
(10975, 8, 16),
(10975, 75, 10),
(10976, 28, 20),
(10977, 39, 30),
(10977, 47, 30),
(10977, 51, 10),
(10977, 63, 20),
(10978, 8, 20),
(10978, 21, 40),
(10978, 40, 10),
(10978, 44, 6),
(10979, 7, 18),
(10979, 12, 20),
(10979, 24, 80),
(10979, 27, 30),
(10979, 31, 24),
(10979, 63, 35),
(10980, 75, 40),
(10981, 38, 60),
(10982, 7, 20),
(10982, 43, 9),
(10983, 13, 84),
(10983, 57, 15),
(10984, 16, 55),
(10984, 24, 20),
(10984, 36, 40),
(10985, 16, 36),
(10985, 18, 8),
(10985, 32, 35),
(10986, 11, 30),
(10986, 20, 15),
(10986, 76, 10),
(10986, 77, 15),
(10987, 7, 60),
(10987, 43, 6),
(10987, 72, 20),
(10988, 7, 60),
(10988, 62, 40),
(10989, 6, 40),
(10989, 11, 15),
(10989, 41, 4),
(10990, 21, 65),
(10990, 34, 60),
(10990, 55, 65),
(10990, 61, 66),
(10991, 2, 50),
(10991, 70, 20),
(10991, 76, 90),
(10992, 72, 2),
(10993, 29, 50),
(10993, 41, 35),
(10994, 59, 18),
(10995, 51, 20),
(10995, 60, 4),
(10996, 42, 40),
(10997, 32, 50),
(10997, 46, 20),
(10997, 52, 20),
(10998, 24, 12),
(10998, 61, 7),
(10998, 74, 20),
(10998, 75, 30),
(10999, 41, 20),
(10999, 51, 15),
(10999, 77, 21),
(11000, 4, 25),
(11000, 24, 30),
(11000, 77, 30),
(11001, 7, 60),
(11001, 22, 25),
(11001, 46, 25),
(11001, 55, 6),
(11002, 13, 56),
```

```
(11002, 35, 15),
(11002, 42, 24),
(11002, 55, 40),
(11003, 1, 4),
(11003, 40, 10),
(11003, 52, 10),
(11004, 26, 6),
(11004, 76, 6),
(11005, 1, 2),
(11005, 59, 10),
(11006, 1, 8),
(11006, 29, 2),
(11007, 8, 30),
(11007, 29, 10),
(11007, 42, 14),
(11008, 28, 70),
(11008, 34, 90),
(11008, 71, 21),
(11009, 24, 12),
(11009, 36, 18),
(11009, 60, 9),
(11010, 7, 20),
(11010, 24, 10),
(11011, 58, 40),
(11011, 71, 20),
(11012, 19, 50),
(11012, 60, 36),
(11012, 71, 60),
(11013, 23, 10),
(11013, 42, 4),
(11013, 45, 20),
(11013, 68, 2),
(11014, 41, 28),
(11015, 30, 15),
(11015, 77, 18),
(11016, 31, 15),
(11016, 36, 16),
(11017, 3, 25),
(11017, 59, 110),
(11017, 70, 30),
(11018, 12, 20),
(11018, 18, 10),
(11018, 56, 5),
(11019, 46, 3),
(11019, 49, 2),
(11020, 10, 24),
(11021, 2, 11),
(11021, 20, 15),
(11021, 26, 63),
(11021, 51, 44),
(11021, 72, 35),
(11022, 19, 35),
(11022, 69, 30),
(11023, 7, 4),
(11023, 43, 30),
(11024, 26, 12),
(11024, 33, 30),
(11024, 65, 21),
(11024, 71, 50),
(11025, 1, 10),
(11025, 13, 20),
(11026, 18, 8),
(11026, 51, 10),
(11027, 24, 30),
(11027, 62, 21),
(11028, 55, 35),
(11028, 59, 24),
(11029, 56, 20),
(11029, 63, 12),
(11030, 2, 100),
(11030, 5, 70),
(11030, 29, 60),
(11030, 59, 100),
(11031, 1, 45),
(11031, 13, 80),
(11031, 24, 21),
(11031, 64, 20),
(11031, 71, 16),
(11032, 36, 35),
(11032, 38, 25),
(11032, 59, 30),
(11033, 53, 70),
(11033, 69, 36),
(11034, 21, 15),
(11034, 44, 12),
(11034, 61, 6),
(11035, 1, 10),
(11035, 35, 60),
(11035, 42, 30),
(11035, 54, 10),
(11036, 13, 7),
(11036, 59, 30),
(11037, 70, 4),
(11038, 40, 5),
(11038, 52, 2),
(11038, 71, 30),
(11039, 28, 20),
(11039, 35, 24),
(11039, 49, 60),
(11039, 57, 28),
(11040, 21, 20),
(11041, 2, 30),
(11041, 63, 30),
(11042, 44, 15),
(11042, 61, 4),
(11043, 11, 10),
(11044, 62, 12),
(11045, 33, 15),
(11045, 51, 24),
(11046, 12, 20),
(11046, 32, 15),
(11046, 35, 18),
(11047, 1, 25),
(11047, 5, 30),
(11048, 68, 42),
(11049, 2, 10),
(11049, 12, 4),
(11050, 76, 50),
(11051, 24, 10),
(11052, 43, 30),
(11052, 61, 10),
(11053, 18, 35),
(11053, 32, 20),
(11053, 64, 25),
(11054, 33, 10),
(11054, 67, 20),
(11055, 24, 15),
(11055, 25, 15),
(11055, 51, 20),
(11055, 57, 20),
(11056, 7, 40),
(11056, 55, 35),
```

```
(11056, 60, 50),
(11057, 70, 3),
(11058, 21, 3),
(11058, 60, 21),
(11058, 61, 4),
(11059, 13, 30),
(11059, 17, 12),
(11059, 60, 35),
(11060, 60, 4),
(11060, 77, 10),
(11061, 60, 15),
(11062, 53, 10),
(11062, 70, 12),
(11063, 34, 30),
(11063, 40, 40),
(11063, 41, 30),
(11064, 17, 77),
(11064, 41, 12),
(11064, 53, 25),
(11064, 55, 4),
(11064, 68, 55),
(11065, 30, 4),
(11065, 54, 20),
(11066, 16, 3),
(11066, 19, 42),
(11066, 34, 35),
(11067, 41, 9),
(11068, 28, 8),
(11068, 43, 36),
(11068, 77, 28),
(11069, 39, 20),
(11070, 1, 40),
(11070, 2, 20),
(11070, 16, 30),
(11070, 31, 20),
(11071, 7, 15),
(11071, 13, 10),
(11072, 2, 8),
(11072, 41, 40),
(11072, 50, 22),
(11072, 64, 130),
(11073, 11, 10),
(11073, 24, 20),
(11074, 16, 14),
(11075, 2, 10),
(11075, 46, 30),
(11075, 76, 2),
(11076, 6, 20),
(11076, 14, 20),
(11076, 19, 10),
(11077, 2, 24),
(11077, 3, 4),
(11077, 4, 1),
(11077, 6, 1),
(11077, 7, 1),
(11077, 8, 2),
(11077, 10, 1),
(11077, 12, 2),
(11077, 13, 4),
(11077, 14, 1),
(11077, 16, 2),
(11077, 20, 1),
(11077, 23, 2),
(11077, 32, 1),
(11077, 39, 2),
(11077, 41, 3),
(11077, 46, 3),
(11077, 52, 2),
(11077, 55, 2),
(11077, 60, 2),
(11077, 64, 2),
(11077, 66, 1),
(11077, 73, 2),
(11077, 75, 4),
(11077, 77, 2);
```

If you write above statement in your SQL Shell, you will see that it returns `INSERT 0 2155`. Which means that we have inserted 2155 rows in our `order_details` table.

Testproducts

We will also add a table called `testproducts` for demonstration purposes later in the tutorial.

The following SQL statement will create a table named `testproducts`:

CREATE TABLE testproducts

```
CREATE TABLE testproducts (
  testproduct_id SERIAL NOT NULL PRIMARY KEY,
  product_name VARCHAR(255),
  category_id INT
);
```

INSERT INTO testproducts

We will fill the table with 10 dummy products:

```
INSERT INTO testproducts (product_name, category_id)
VALUES
  ('Johns Fruit Cake', 3),
  ('Marys Healthy Mix', 9),
  ('Peters Scary Stuff', 10),
  ('Jims Secret Recipe', 11),
  ('Elisabeths Best Apples', 12),
  ('Janes Favorite Cheese', 4),
  ('Billys Home Made Pizza', 13),
  ('Ellas Special Salmon', 8),
  ('Roberts Rich Spaghetti', 5),
  ('Mias Popular Ice', 14);
```

If you write above statement in your SQL Shell, you will see that it returns `INSERT 0 10`. Which means that we have inserted 10 rows in our `testproducts` table.

Cars

We will create a table called `cars`, that is not associated with this tutorial. But we will use it in another example.

CREATE TABLE cars

The following SQL statement will create a table named `cars`:

```
CREATE TABLE cars (
  brand VARCHAR(255),
  model VARCHAR(255),
  year INT
);
```

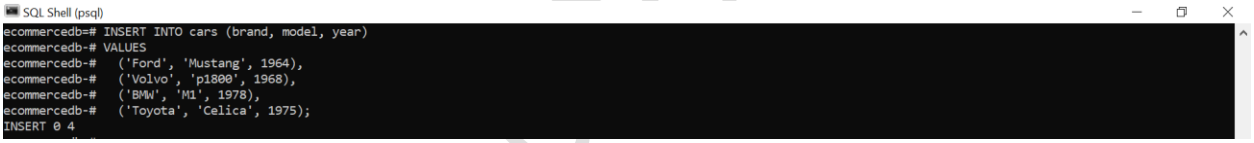
In the SQL Shell application on your computer the operation above might look like this:

INSERT INTO cars

The following SQL statement will insert data in a table named `cars`:

```
INSERT INTO cars (brand, model, year)
VALUES
  ('Ford', 'Mustang', 1964),
  ('Volvo', 'p1800', 1968),
  ('BMW', 'M1', 1978),
  ('Toyota', 'Celica', 1975);
```

If you write above statement in your SQL Shell, you will see that it returns `INSERT 0 10`. Which means that we have inserted 10 rows in our `testproducts` table.



See All Table List

To see all table which we are created. Execute the following command.

```
\dt
```

We can see that we have created total 7 tables in SQL Shell.

PostgreSQL Syntax (DML)

Operators

Operators in the WHERE clause

We can operate with different operators in the WHERE clause:

=	Equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
<>	Not equal to
!=	Not equal to
LIKE	Check if a value matches a pattern (case sensitive)
ILIKE	Check if a value matches a pattern (case insensitive)
AND	Logical AND
OR	Logical OR
IN	Check if a value is between a range of values
BETWEEN	Check if a value is between a range of values
IS NULL	Check if a value is NULL
NOT	Makes a negative result e.g. NOT LIKE, NOT IN, NOT BETWEEN

Equal To

The = operator is used when you want to return all records where a column is equal to a specified value:

Return all records where the brand is 'Volvo':

```
SELECT * FROM cars
WHERE brand = 'Volvo';
```

Less Than

The < operator is used when you want to return all records where a column is less than a specified value.

Return all records where the year is less than 1975:

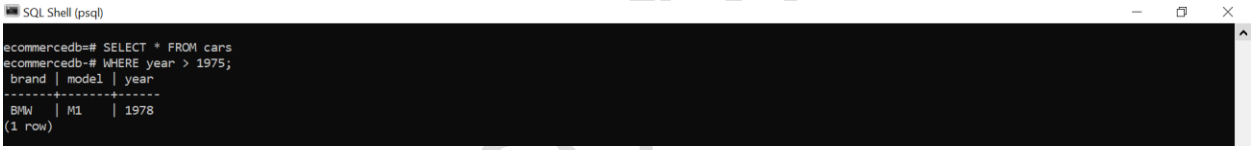
```
SELECT * FROM cars
WHERE year < 1975;
```

Greater Than

The > operator is used when you want to return all records where a column is greater than a specified value.

Return all records where the year is greater than 1975:

```
SELECT * FROM cars
WHERE year > 1975;
```



Less Than or Equal To

The <= operator is used when you want to return all records where a column is less than, or equal to, a specified value.

Return all records where the year is less than or equal to 1975:

```
SELECT * FROM cars
WHERE year <= 1975;
```

Greater Than or Equal to

The >= operator is used when you want to return all records where a columns is greater than, or equal to, a specified value.

Return all records where the `year` is greater than or equal 1975:

```
SELECT * FROM cars
WHERE year >= 1975;
```

Not Equal To

The <> operator is used when you want to return all records where a column is NOT equal to a specified value:

Return all records where the brand is `NOT` 'Volvo':

```
SELECT * FROM cars
WHERE brand <> 'Volvo';
```

Or you can follow this statement. It returns the same output, but its syntax is slightly different.

```
SELECT * FROM cars
WHERE brand != 'Volvo';
```

LIKE

The `LIKE` operator is used when you want to return all records where a column is equal to a specified pattern.

The pattern can be an absolute value like 'Volvo', or with a wildcard that has a special meaning.

There are two wildcards often used in conjunction with the LIKE operator:

- The percent sign `%`, represents zero, one, or multiple characters.
- The underscore sign _, represents one single character.

Return all records where the model STARTS with a capital 'M':

```
SELECT * FROM cars
WHERE model LIKE 'M%';
```

Note: The `LIKE` operator is case sensitive.

ILIKE

Same as the `LIKE` operator, but `ILIKE` is case insensitive.

Return all records where the model starts with a 'm':

```
SELECT * FROM cars
WHERE model ILIKE 'm%';
```

AND

The logical `AND` operator is used when you want to check more that one condition:

Return all records where the brand is 'Volvo' and the year is 1968:

```
SELECT * FROM cars
WHERE brand = 'Volvo' AND year = 1968;
```

OR

The logical `OR` operator is used when you can accept that only one of many conditions is true:

Return all records where the brand is 'Volvo' OR the year is 1975:

```
SELECT * FROM cars
WHERE brand = 'Volvo' OR year = 1975;
```

IN

Return all records where the brand is present in this list: ('Volvo', 'Mercedes', 'Ford'):

```
SELECT * FROM cars
WHERE brand IN ('Volvo', 'Mercedes', 'Ford');
```

BETWEEN

The `BETWEEN` operator is used to check if a column's value is between a specified range of values:

Return all records where the `year` is between 1970 and 1980:

```
SELECT * FROM cars
WHERE year BETWEEN 1970 AND 1980;
```

The `BETWEEN` operator includes the *`from`* and *`to`* values, meaning that in the above example, the result would include cars made in 1970 and 1980 as well.

IS NULL

The `IS NULL` operator is used to check if a column's value is NULL:

Return all records where the model is NULL:

```
SELECT * FROM cars
WHERE model IS NULL;
```

NOT

The `NOT` operator can be used together with `LIKE`, `ILIKE`, `IN`, `BETWEEN`, and `NULL` operators to reverse the truth of the operator.

NOT LIKE

Return all records where the `brand` does `NOT` start with a capital 'B' (case sensitive):

```
SELECT * FROM cars
WHERE brand NOT LIKE 'B%';
```

NOT ILIKE

Return all records where the `brand` does `NOT` start with a 'b' (case insensitive):

```
SELECT * FROM cars
WHERE brand NOT ILIKE 'b%';
```

NOT IN

Return all records where the brand is NOT present in this list: ('Volvo', 'Mercedes', 'Ford'):

```
SELECT * FROM cars
WHERE brand NOT IN ('Volvo', 'Mercedes', 'Ford');
```

NOT BETWEEN

Return all records where the year is NOT between 1970 and 1980:

```
SELECT * FROM cars
WHERE year NOT BETWEEN 1970 AND 1980;
```

Note: The `NOT BETWEEN` operator excludes the *`from`* and *`to`* values, meaning that in the above example, the result would not include cars made in 1970 and 1980.

IS NOT NULL

Return all records where the model is NOT null:

```
SELECT * FROM cars
WHERE model IS NOT NULL;
```

The `cars` table has no columns with NULL values, so the example above will return all 4 rows.

SELECT

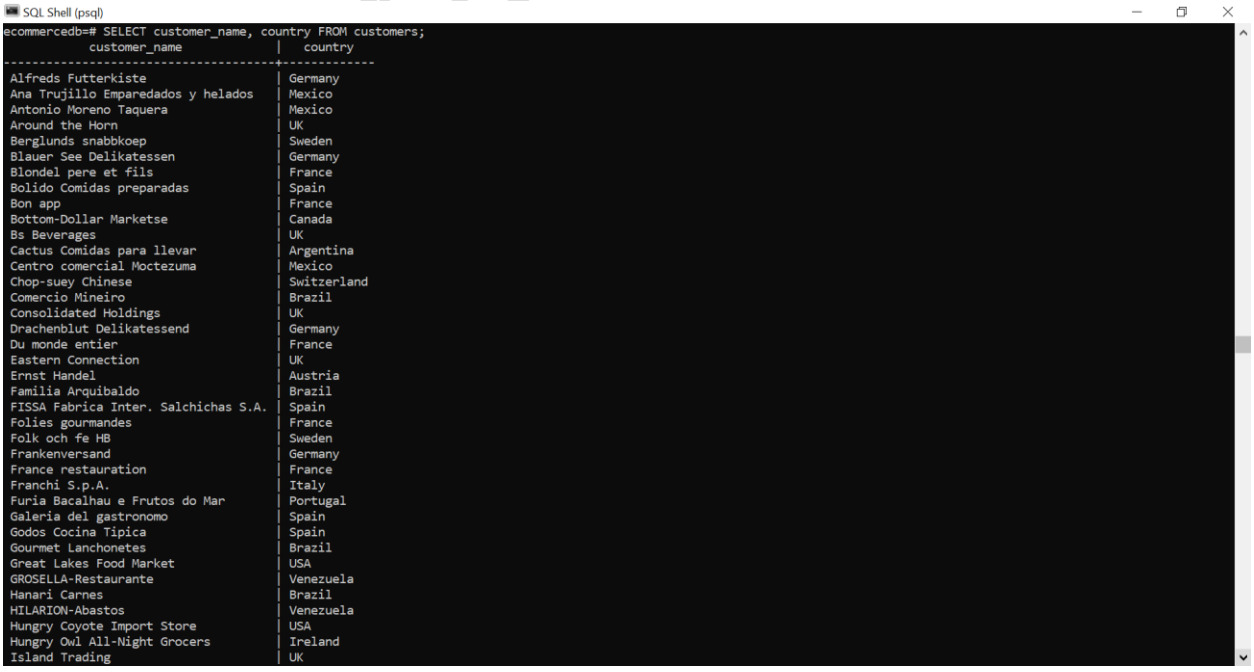
Select Data

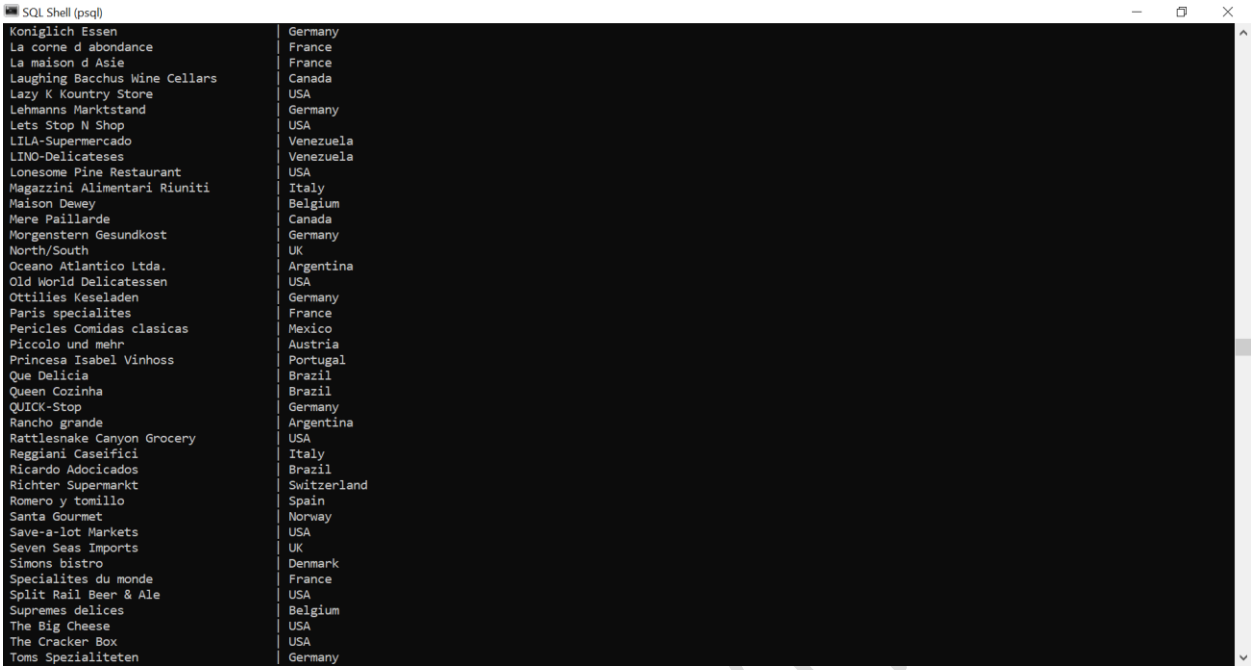
To retrieve data from a data base, we use the `SELECT` statement.

Specify Columns

By specifying the column names, we can choose which columns to select:

```
SELECT customer_name, country FROM customers;
```





This query returns 91 rows of specific columns.

Return ALL Columns

Specify a * instead of the column names to select *all* columns:

```
SELECT * FROM testproducts;
```

SELECT DISTINCT

The SELECT DISTINCT Statement

The `SELECT DISTINCT` statement is used to return only distinct (different) values.

Inside a table, a column often contains many duplicate values and sometimes you only want to list the different (distinct) values.

```
SELECT DISTINCT country FROM customers;
```

Even though the customers table has 91 records, it only has 21 different countries, and that is what you get as a result when executing the `SELECT DISTINCT` statement above.

SELECT COUNT(DISTINCT)

We can also use the `DISTINCT` keyword in combination with the `COUNT` statement, which in the example below will return the number of different countries there are in the `customers` table.

Return the number of different countries there are in the `customers` table:

```
SELECT COUNT(DISTINCT country) FROM customers;
```

WHERE - Filter Data

Filter Records

The `WHERE` clause is used to filter records.

It is used to extract only those records that fulfill a specified condition.

If we want to return only the records where `city` is `London`, we can specify that in the `WHERE` clause:

```
SELECT * FROM customers
WHERE city = 'London';
```

Text Fields vs. Numeric Fields

PostgreSQL requires quotes around text values.

However, numeric fields should not be enclosed in quotes:

```
SELECT * FROM customers
WHERE customer_id = 19;
```

Note: Quotes around numeric fields will not fail, but it is good practice to always write numeric values without quotes.

Greater than

Use the > operator to return all records where `customer_id` is greater than 80:

```
SELECT * FROM customers
WHERE customer_id > 80;
```

Note: Quotes around numeric fields will not fail, but it is good practice to always write numeric values without quotes.

ORDER BY

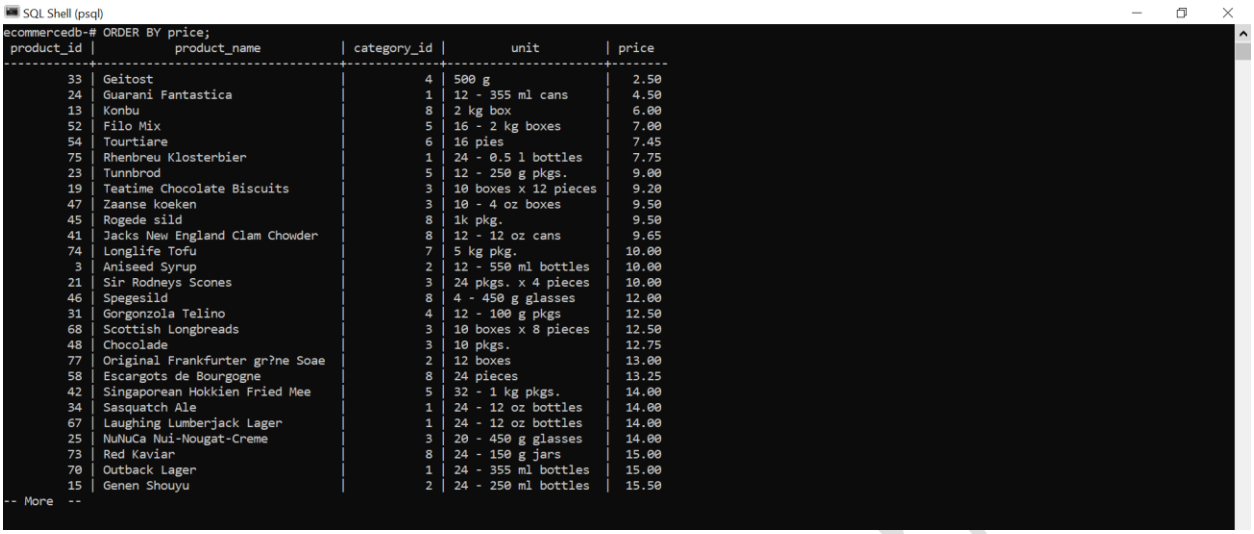
Sort Data

The `ORDER BY` keyword is used to sort the result in ascending or descending order. The `ORDER BY` keyword sorts the records in ascending order by default. To sort the records in descending order, use the `DESC` keyword.

Sort the table by `price`:

```
SELECT * FROM products
ORDER BY price;
```

When we execute this statement, then we will see it like this. We know that products table has 77 rows, all rows will be returned.

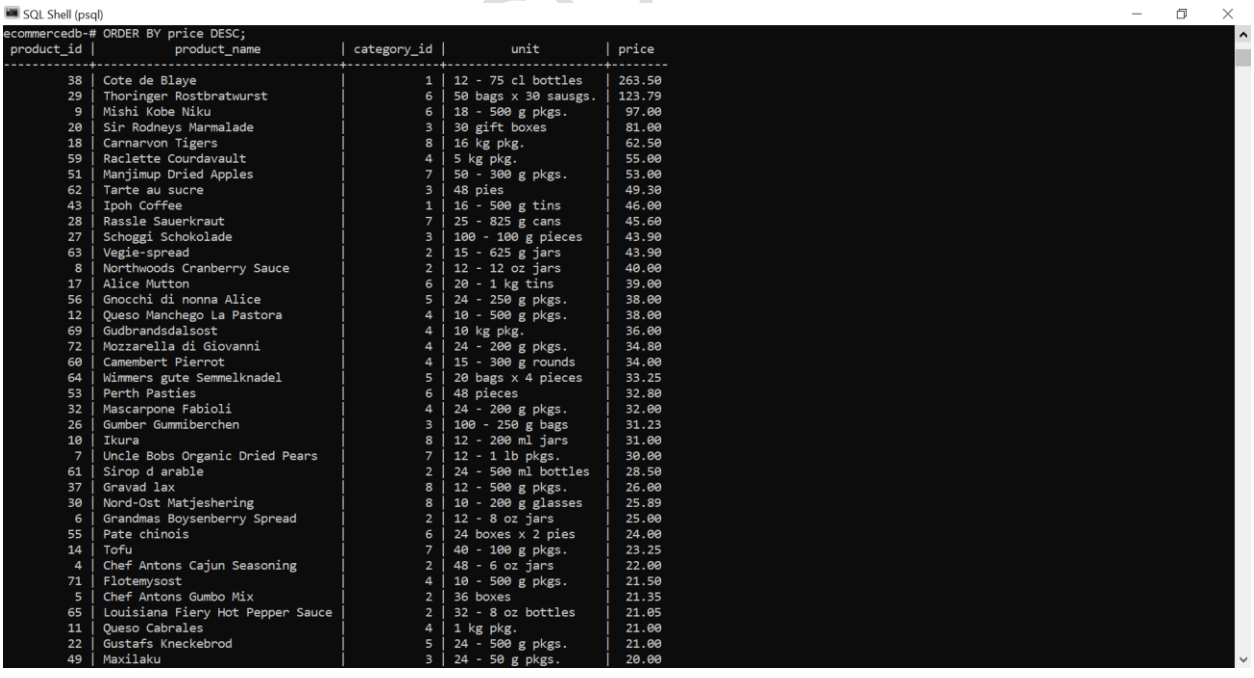


DESC

The `ORDER BY` keyword sorts the records in ascending order by default. To sort the records in descending order, use the `DESC` keyword

Sort the table by `price`, in descending order:

```
SELECT * FROM products
ORDER BY price DESC;
```



Sort Alphabetically

For string value the `ORDER BY` keyword will order alphabetically:

Sort the table by `product_name`:

```
SELECT * FROM products
ORDER BY product_name;
```

SQL Shell (psql)

```
ecommercedb-# ORDER BY product_name;
 product_id |          product_name          | category_id |        unit         | price
------------+--------------------------------+-------------+---------------------+--------
         17 | Alice Mutton                   |           6 | 20 - 1 kg tins      |  39.00
          3 | Aniseed Syrup                  |           2 | 12 - 550 ml bottles |  10.00
         40 | Boston Crab Meat               |           8 | 24 - 4 oz tins      |  18.40
         60 | Camembert Pierrot              |           4 | 15 - 300 g rounds   |  34.00
         18 | Carnarvon Tigers               |           8 | 16 kg pkg.          |  62.50
          1 | Chais                          |           1 | 10 boxes x 20 bags  |  18.00
          2 | Chang                          |           1 | 24 - 12 oz bottles  |  19.00
         39 | Chartreuse verte               |           1 | 750 cc per bottle   |  18.00
          4 | Chef Antons Cajun Seasoning    |           2 | 48 - 6 oz jars      |  22.00
          5 | Chef Antons Gumbo Mix          |           2 | 36 boxes            |  21.35
         48 | Chocolade                      |           3 | 10 pkgs.            |  12.75
         38 | Cote de Blaye                  |           1 | 12 - 75 cl bottles  | 263.50
         58 | Escargots de Bourgogne         |           8 | 24 pieces           |  13.25
         52 | Filo Mix                       |           5 | 16 - 2 kg boxes     |   7.00
         71 | Flotemysost                    |           4 | 10 - 500 g pkgs.    |  21.50
         33 | Geitost                        |           4 | 500 g               |   2.50
         15 | Genen Shouyu                   |           2 | 24 - 250 ml bottles |  15.50
         56 | Gnocchi di nonna Alice         |           5 | 24 - 250 g pkgs.    |  38.00
         31 | Gorgonzola Telino              |           4 | 12 - 100 g pkgs     |  12.50
          6 | Grandmas Boysenberry Spread    |           2 | 12 - 8 oz jars      |  25.00
         37 | Gravad lax                     |           8 | 12 - 500 g pkgs.    |  26.00
         24 | Guarani Fantastica             |           1 | 12 - 355 ml cans    |   4.50
         69 | Gudbrandsdalsost               |           4 | 10 kg pkg.          |  36.00
         44 | Gula Malacca                   |           2 | 20 - 2 kg bags      |  19.45
         26 | Gumber Gummiberchen            |           3 | 100 - 250 g bags    |  31.23
         22 | Gustafs Kneckebrod             |           5 | 24 - 500 g pkgs.    |  21.00
         10 | Ikura                          |           8 | 12 - 200 ml jars    |  31.00
         36 | Inlagd Sill                    |           8 | 24 - 250 g jars     |  19.00
         43 | Ipoh Coffee                    |           1 | 16 - 500 g tins     |  46.00
         41 | Jacks New England Clam Chowder |           8 | 12 - 12 oz cans     |   9.65
         13 | Konbu                          |           8 | 2 kg box            |   6.00
         76 | Lakkalikeeri                   |           1 | 500 ml              |  18.00
         67 | Laughing Lumberjack Lager      |           1 | 24 - 12 oz bottles  |  14.00
         74 | Longlife Tofu                  |           7 | 5 kg pkg.           |  10.00
         65 | Louisiana Fiery Hot Pepper Sauce |         2 | 32 - 8 oz bottles   |  21.05
         66 | Louisiana Hot Spiced Okra      |           2 | 24 - 8 oz jars      |  17.00
         51 | Manjimup Dried Apples          |           7 | 50 - 300 g pkgs.    |  53.00
         32 | Mascarpone Fabioli             |           4 | 24 - 200 g pkgs.    |  32.00
```

Alphabetically DESC

To sort the table reverse alphabetically, use the `DESC` keyword:

```
SELECT * FROM products
ORDER BY product_name DESC;
```

SQL Shell (psql)

```
ecommercedb=# SELECT * FROM products
ecommercedb-# ORDER BY product_name DESC;
 product_id |          product_name          | category_id |        unit         | price
------------+--------------------------------+-------------+---------------------+--------
         47 | Zaanse koeken                  |           3 | 10 - 4 oz boxes     |   9.50
         64 | Wimmers gute Semmelknadel      |           5 | 20 bags x 4 pieces  |  33.25
         63 | Vegie-spread                   |           2 | 15 - 625 g jars     |  43.90
         50 | Valkoinen suklaa               |           3 | 12 - 100 g bars     |  16.25
          7 | Uncle Bobs Organic Dried Pears |           7 | 12 - 1 lb pkgs.     |  30.00
         23 | Tunnbrod                       |           5 | 12 - 250 g pkgs.    |   9.00
         54 | Tourtiare                      |           6 | 16 pies             |   7.45
         14 | Tofu                           |           7 | 40 - 100 g pkgs.    |  23.25
         29 | Thoringer Rostbratwurst        |           6 | 50 bags x 30 sausgs. | 123.79
         19 | Teatime Chocolate Biscuits     |           3 | 10 boxes x 12 pieces |   9.20
         62 | Tarte au sucre                 |           3 | 48 pies             |  49.30
         35 | Steeleye Stout                 |           1 | 24 - 12 oz bottles  |  18.00
         46 | Spegesild                      |           8 | 4 - 450 g glasses   |  12.00
         61 | Sirop d arable                 |           2 | 24 - 500 ml bottles |  28.50
         21 | Sir Rodneys Scones             |           3 | 24 pkgs. x 4 pieces |  10.00
         20 | Sir Rodneys Marmalade          |           3 | 30 gift boxes       |  81.00
         42 | Singaporean Hokkien Fried Mee  |           5 | 32 - 1 kg pkgs.     |  14.00
         68 | Scottish Longbreads            |           3 | 10 boxes x 8 pieces |  12.50
         27 | Schoggi Schokolade             |           3 | 100 - 100 g pieces  |  43.90
         34 | Sasquatch Ale                  |           1 | 24 - 12 oz bottles  |  14.00
         45 | Rogede sild                    |           8 | 1k pkg.             |   9.50
         75 | Rhenbreu Klosterbier           |           1 | 24 - 0.5 l bottles  |   7.75
         73 | Red Kaviar                     |           8 | 24 - 150 g jars     |  15.00
         57 | Ravioli Angelo                 |           5 | 24 - 250 g pkgs.    |  19.50
         28 | Rassle Sauerkraut              |           7 | 25 - 825 g cans     |  45.60
         59 | Raclette Courdavault           |           4 | 5 kg pkg.           |  55.00
         12 | Queso Manchego La Pastora      |           4 | 10 - 500 g pkgs.    |  38.00
         11 | Queso Cabrales                 |           4 | 1 kg pkg.           |  21.00
         53 | Perth Pasties                  |           6 | 48 pieces           |  32.80
         16 | Pavlova                        |           3 | 32 - 500 g boxes    |  17.45
         55 | Pate chinois                   |           6 | 24 boxes x 2 pies   |  24.00
         70 | Outback Lager                  |           1 | 24 - 355 ml bottles |  15.00
         77 | Original Frankfurter gr?ne Soae |          2 | 12 boxes            |  13.00
         25 | NuNuCa Nui-Nougat-Creme        |           3 | 20 - 450 g glasses  |  14.00
          8 | Northwoods Cranberry Sauce     |           2 | 12 - 12 oz jars     |  40.00
         30 | Nord-Ost Matjeshering          |           8 | 10 - 200 g glasses  |  25.89
         72 | Mozzarella di Giovanni         |           4 | 24 - 200 g pkgs.    |  34.80
```

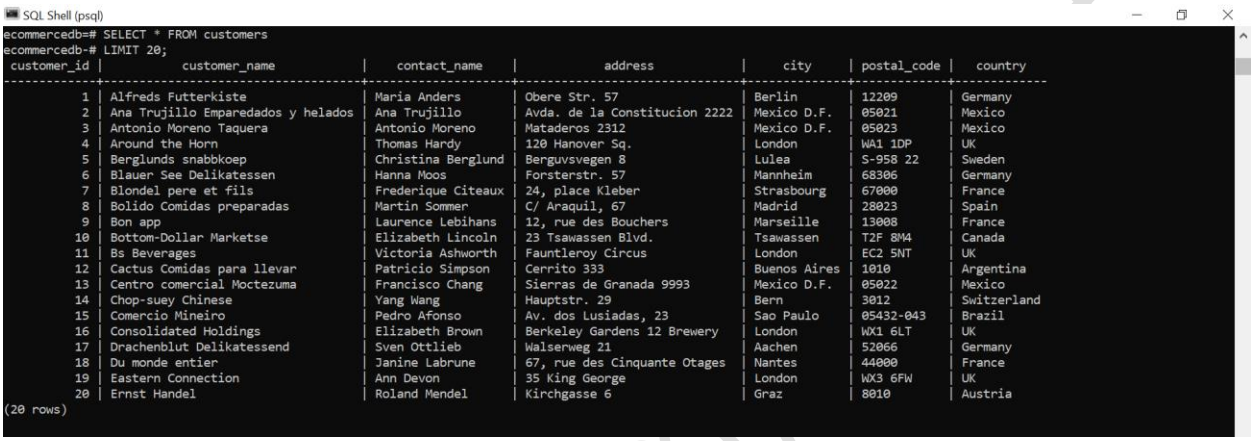
LIMIT

The LIMIT Clause

The `LIMIT` clause is used to limit the maximum number of records to return.

Return only the 20 first records from the `customers` table:

```
SELECT * FROM customers
LIMIT 20;
```



```
SQL Shell (psql)
ecommercedb=# SELECT * FROM customers
ecommercedb-# LIMIT 20;
 customer_id |          customer_name           |    contact_name    |          address           |     city      | postal_code |   country
-------------+----------------------------------+--------------------+----------------------------+---------------+-------------+-------------
           1 | Alfreds Futterkiste              | Maria Anders       | Obere Str. 57              | Berlin        | 12209       | Germany
           2 | Ana Trujillo Emparedados y helados | Ana Trujillo     | Avda. de la Constitucion 2222 | Mexico D.F. | 05021       | Mexico
           3 | Antonio Moreno Taquera           | Antonio Moreno     | Mataderos 2312             | Mexico D.F.   | 05023       | Mexico
           4 | Around the Horn                  | Thomas Hardy       | 120 Hanover Sq.            | London        | WA1 1DP     | UK
           5 | Berglunds snabbkoep              | Christina Berglund | Berguvsvegen 8             | Lulea         | S-958 22    | Sweden
           6 | Blauer See Delikatessen          | Hanna Moos         | Forsterstr. 57             | Mannheim      | 68306       | Germany
           7 | Blondel pere et fils             | Frederique Citeaux | 24, place Kleber           | Strasbourg    | 67000       | France
           8 | Bolido Comidas preparadas        | Martin Sommer      | C/ Araquil, 67             | Madrid        | 28023       | Spain
           9 | Bon app                          | Laurence Lebihans  | 12, rue des Bouchers       | Marseille     | 13008       | France
          10 | Bottom-Dollar Marketse           | Elizabeth Lincoln  | 23 Tsawassen Blvd.         | Tsawassen     | T2F 8M4     | Canada
          11 | Bs Beverages                     | Victoria Ashworth  | Fauntleroy Circus          | London        | EC2 5NT     | UK
          12 | Cactus Comidas para llevar       | Patricio Simpson   | Cerrito 333                | Buenos Aires  | 1010        | Argentina
          13 | Centro comercial Moctezuma       | Francisco Chang    | Sierras de Granada 9993    | Mexico D.F.   | 05022       | Mexico
          14 | Chop-suey Chinese                | Yang Wang          | Hauptstr. 29               | Bern          | 3012        | Switzerland
          15 | Comercio Mineiro                 | Pedro Afonso       | Av. dos Lusiadas, 23       | Sao Paulo     | 05432-043   | Brazil
          16 | Consolidated Holdings            | Elizabeth Brown    | Berkeley Gardens 12 Brewery | London       | WX1 6LT     | UK
          17 | Drachenblut Delikatessend        | Sven Ottlieb       | Walserweg 21               | Aachen        | 52066       | Germany
          18 | Du monde entier                  | Janine Labrune     | 67, rue des Cinquante Otages | Nantes      | 44000       | France
          19 | Eastern Connection               | Ann Devon          | 35 King George             | London        | WX3 6FW     | UK
          20 | Ernst Handel                     | Roland Mendel      | Kirchgasse 6               | Graz          | 8010        | Austria
(20 rows)
```

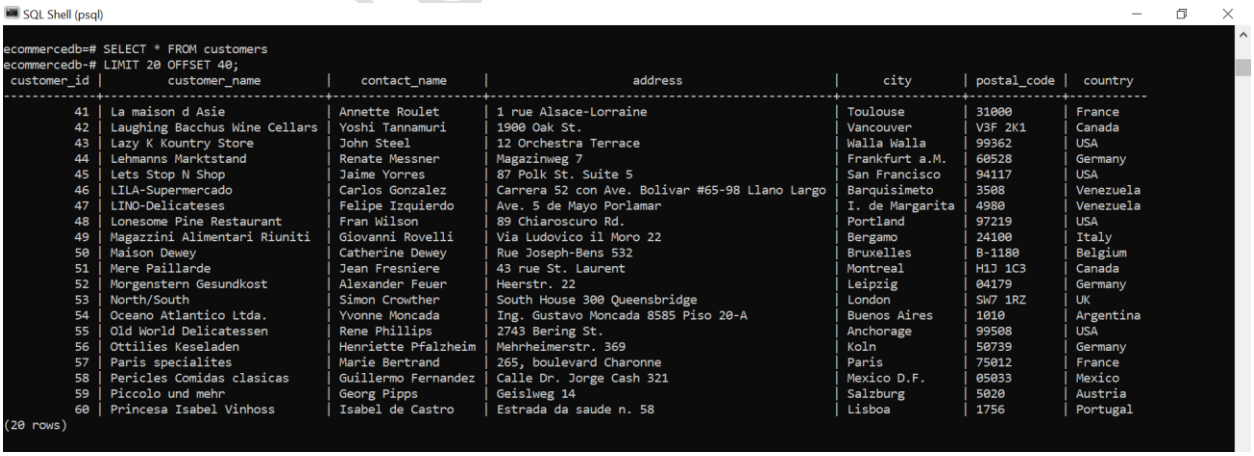
The OFFSET Clause

The `OFFSET` clause is used to specify where to start selecting the records to return.

If you want to return 20 records, but start at number 40, you can use both `LIMIT` and `OFFSET`.

Return 20 records, starting from the 41th record:

```
SELECT * FROM customers
LIMIT 20 OFFSET 40;
```



```
SQL Shell (psql)
ecommercedb=# SELECT * FROM customers
ecommercedb-# LIMIT 20 OFFSET 40;
 customer_id |        customer_name        |    contact_name    |                address                |      city      | postal_code |  country
-------------+-----------------------------+--------------------+---------------------------------------+----------------+-------------+-----------
          41 | La maison d Asie            | Annette Roulet     | 1 rue Alsace-Lorraine                 | Toulouse       | 31000       | France
          42 | Laughing Bacchus Wine Cellars | Yoshi Tannamuri  | 1900 Oak St.                          | Vancouver      | V3F 2K1     | Canada
          43 | Lazy K Kountry Store        | John Steel         | 12 Orchestra Terrace                  | Walla Walla    | 99362       | USA
          44 | Lehmanns Marktstand         | Renate Messner     | Magazinweg 7                          | Frankfurt a.M. | 60528       | Germany
          45 | Lets Stop N Shop            | Jaime Yorres       | 87 Polk St. Suite 5                   | San Francisco  | 94117       | USA
          46 | LILA-Supermercado           | Carlos Gonzalez    | Carrera 52 con Ave. Bolivar #65-98 Llano Largo | Barquisimeto | 3508 | Venezuela
          47 | LINO-Delicateses            | Felipe Izquierdo   | Ave. 5 de Mayo Porlamar               | I. de Margarita | 4980       | Venezuela
          48 | Lonesome Pine Restaurant    | Fran Wilson        | 89 Chiaroscuro Rd.                    | Portland       | 97219       | USA
          49 | Magazzini Alimentari Riuniti | Giovanni Rovelli  | Via Ludovico il Moro 22               | Bergamo        | 24100       | Italy
          50 | Maison Dewey                | Catherine Dewey    | Rue Joseph-Bens 532                   | Bruxelles      | B-1180      | Belgium
          51 | Mere Paillarde              | Jean Fresniere     | 43 rue St. Laurent                    | Montreal       | H1J 1C3     | Canada
          52 | Morgenstern Gesundkost      | Alexander Feuer    | Heerstr. 22                           | Leipzig        | 04179       | Germany
          53 | North/South                 | Simon Crowther     | South House 300 Queensbridge          | London         | SW7 1RZ     | UK
          54 | Oceano Atlantico Ltda.      | Yvonne Moncada     | Ing. Gustavo Moncada 8585 Piso 20-A   | Buenos Aires   | 1010        | Argentina
          55 | Old World Delicatessen      | Rene Phillips      | 2743 Bering St.                       | Anchorage      | 99508       | USA
          56 | Ottilies Keseladen          | Henriette Pfalzheim | Mehrheimerstr. 369                   | Koln           | 50739       | Germany
          57 | Paris specialites           | Marie Bertrand     | 265, boulevard Charonne               | Paris          | 75012       | France
          58 | Pericles Comidas clasicas   | Guillermo Fernandez | Calle Dr. Jorge Cash 321             | Mexico D.F.    | 05033       | Mexico
          59 | Piccolo und mehr            | Georg Pipps        | Geislweg 14                           | Salzburg       | 5020        | Austria
          60 | Princesa Isabel Vinhoss     | Isabel de Castro   | Estrada da saude n. 58                | Lisboa         | 1756        | Portugal
(20 rows)
```

Note: The first record is number `0`, so when you specify `OFFSET 40` it means starting at record number 41.

MIN and MAX Functions

MIN

The `MIN()` function returns the smallest value of the selected column.

Return the lowest price in the `products` table:

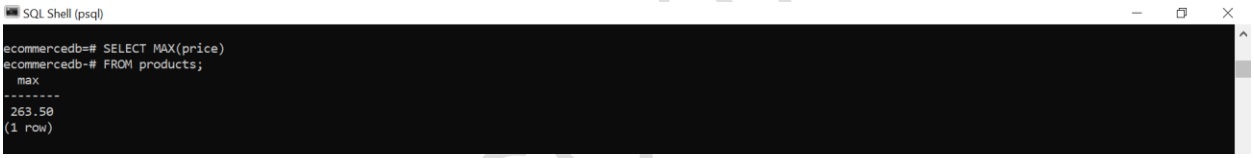
```
SELECT MIN(price)
FROM products;
```

MAX

The `MAX()` function returns the largest value of the selected column.

Return the highest price in the `products` table:

```
SELECT MAX(price)
FROM products;
```



Set Column Name

When you use `MIN()` or `MAX()`, the returned column will be named `min` or `max` by default. To give the column a new name, use the `AS` keyword.

Return the lowest price, and name the column `lowest_price`:

```
SELECT MIN(price) AS lowest_price
FROM products;
```

COUNT Function

COUNT

The `COUNT()` function returns the number of rows that matches a specified criterion.

If the specified criterion is a column name, the `COUNT()` function returns the number of columns with that name.

Return the number of customers from the `customers` table:

```
SELECT COUNT(customer_id)
FROM customers;
```

Note: NULL values are not counted.

COUNT With WHERE Clause

By specifying a `WHERE` clause, you can e.g. return the number of customers that comes from London:

Return the number of customers from London:

```
SELECT COUNT(customer_id)
FROM customers
WHERE city = 'London';
```

SUM Function

SUM

The `SUM()` function returns the total sum of a numeric column.

The following SQL statement finds the sum of the `quantity` fields in the `order_details` table:

Return the total amount of ordered items:

```
SELECT SUM(quantity)
FROM order_details;
```

Note: NULL values are ignored.

AVG Function

AVG

The `AVG()` function returns the average value of a numeric column.

Return the average price of all the products in the `products` table:

```
SELECT AVG(price)
FROM products;
```

Note: NULL values are ignored.

With 2 Decimals

The above example returned the average price of all products, the result was `28.8663636363636364`.

We can use the `::NUMERIC` operator to round the average price to a number with 2 decimals:

Return the average price of all the products, rounded to 2 decimals:

```
SELECT AVG(price)::NUMERIC(10,2)
FROM products;
```

Note: If we want to show more decimal numbers, then replace them with a number where 2 is present.

LIKE Operator

LIKE

The `LIKE` operator is used in a `WHERE` clause to search for a specified pattern in a column.

There are two wildcards often used in conjunction with the `LIKE` operator:

- `%` The percent sign represents zero, one, or multiple characters
- `_` The underscore sign represents one, single character

Starts with

To return records that starts with a specific letter or phrase, add the `%` at the end of the letter or phrase.

Return all customers with a name that starts with the letter 'A':

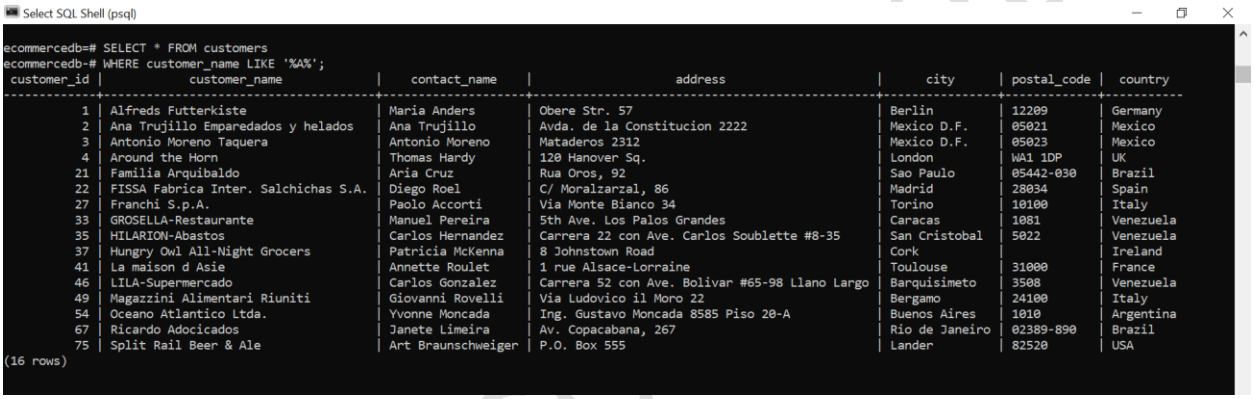
```
SELECT * FROM customers
WHERE customer_name LIKE 'A%';
```

Contains

To return records that contains a specific letter or phrase, add the `%` both before and after the letter or phrase.

Return all customers with a name that contains the letter 'A':

```
SELECT * FROM customers
WHERE customer_name LIKE '%A%';
```



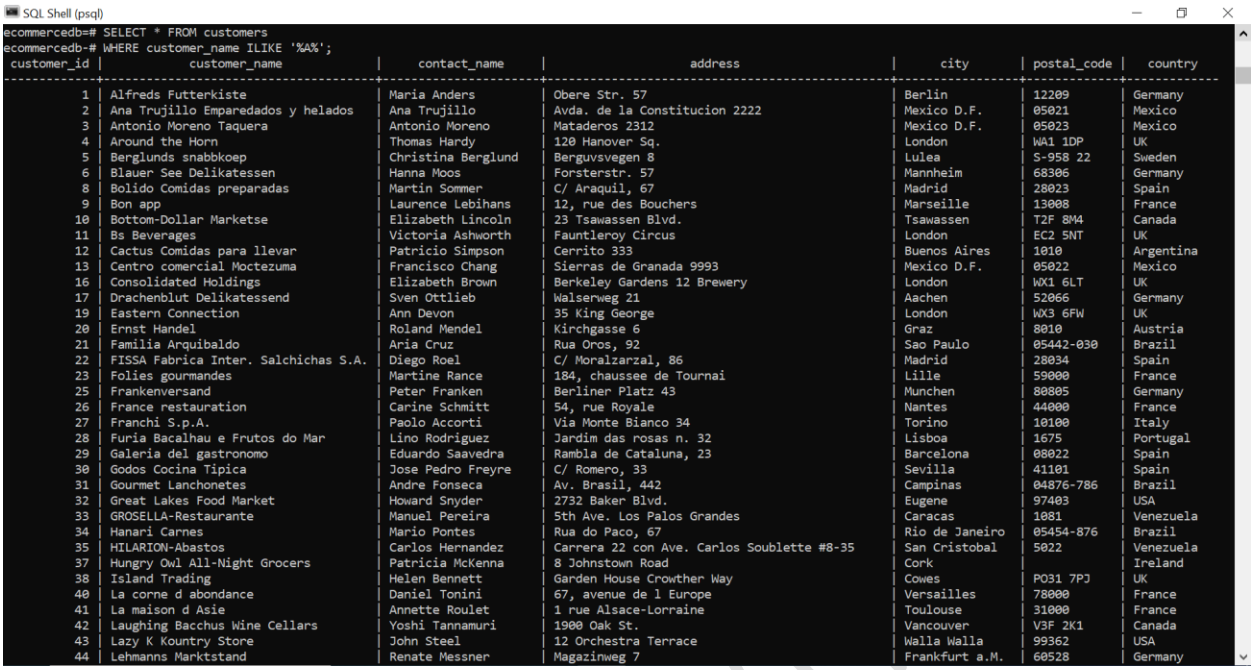
Note: The `LIKE` operator is case sensitive, if you want to do a case insensitive search, use the `ILIKE` operator instead.

ILIKE

Return all customers with a name that contains the letter 'A' or 'a':

```
SELECT * FROM customers
WHERE customer_name ILIKE '%A%';
```

This statement returns many rows, because of case insensitive as well as containing only one ‘a’ or ‘A’.



customer_id	customer_name	contact_name	address	city	postal_code	country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitucion 2222	Mexico D.F.	05021	Mexico
3	Antonio Moreno Taquera	Antonio Moreno	Mataderos 2312	Mexico D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbkoep	Christina Berglund	Berguvsvegen 8	Lulea	S-958 22	Sweden
6	Blauer See Delikatessen	Hanna Moos	Forsterstr. 57	Mannheim	68306	Germany
8	Bolido Comidas preparadas	Martin Sommer	C/ Araquil, 67	Madrid	28023	Spain
9	Bon app	Laurence Lebihans	12, rue des Bouchers	Marseille	13008	France
10	Bottom-Dollar Marketse	Elizabeth Lincoln	23 Tsawassen Blvd.	Tsawassen	T2F 8M4	Canada
11	Bs Beverages	Victoria Ashworth	Fauntleroy Circus	London	EC2 5NT	UK
12	Cactus Comidas para llevar	Patricio Simpson	Cerrito 333	Buenos Aires	1010	Argentina
13	Centro comercial Moctezuma	Francisco Chang	Sierras de Granada 9993	Mexico D.F.	05022	Mexico
16	Consolidated Holdings	Elizabeth Brown	Berkeley Gardens 12 Brewery	London	WX1 6LT	UK
17	Drachenblut Delikatessend	Sven Ottlieb	Walserweg 21	Aachen	52066	Germany
19	Eastern Connection	Ann Devon	35 King George	London	WX3 6FW	UK
20	Ernst Handel	Roland Mendel	Kirchgasse 6	Graz	8010	Austria
21	Familia Arquibaldo	Aria Cruz	Rua Oros, 92	Sao Paulo	05442-030	Brazil
22	FISSA Fabrica Inter. Salchichas S.A.	Diego Roel	C/ Moralzarzal, 86	Madrid	28034	Spain
23	Folies gourmandes	Martine Rance	184, chaussee de Tournai	Lille	59000	France
25	Frankenversand	Peter Franken	Berliner Platz 43	Munchen	80805	Germany
26	France restauration	Carine Schmitt	54, rue Royale	Nantes	44000	France
27	Franchi S.p.A.	Paolo Accorti	Via Monte Bianco 34	Torino	10100	Italy
28	Furia Bacalhau e Frutos do Mar	Lino Rodriguez	Jardim das rosas n. 32	Lisboa	1675	Portugal
29	Galeria del gastronomo	Eduardo Saavedra	Rambla de Cataluna, 23	Barcelona	08022	Spain
30	Godos Cocina Tipica	Jose Pedro Freyre	C/ Romero, 33	Sevilla	41101	Spain
31	Gourmet Lanchonetes	Andre Fonseca	Av. Brasil, 442	Campinas	04876-786	Brazil
32	Great Lakes Food Market	Howard Snyder	2732 Baker Blvd.	Eugene	97403	USA
33	GROSELLA-Restaurante	Manuel Pereira	5th Ave. Los Palos Grandes	Caracas	1081	Venezuela
34	Hanari Carnes	Mario Pontes	Rua do Paco, 67	Rio de Janeiro	05454-876	Brazil
35	HILARION-Abastos	Carlos Hernandez	Carrera 22 con Ave. Carlos Soublette #8-35	San Cristobal	5022	Venezuela
37	Hungry Owl All-Night Grocers	Patricia McKenna	8 Johnstown Road	Cork		Ireland
38	Island Trading	Helen Bennett	Garden House Crowther Way	Cowes	PO31 7PJ	UK
40	La corne d abondance	Daniel Tonini	67, avenue de l Europe	Versailles	78000	France
41	La maison d Asie	Annette Roulet	1 rue Alsace-Lorraine	Toulouse	31000	France
42	Laughing Bacchus Wine Cellars	Yoshi Tannamuri	1900 Oak St.	Vancouver	V3F 2K1	Canada
43	Lazy K Kountry Store	John Steel	12 Orchestra Terrace	Walla Walla	99362	USA
44	Lehmanns Marktstand	Renate Messner	Magazinweg 7	Frankfurt a.M.	60528	Germany

Ends with

To return records that ends with a specific letter or phrase, add the `%` before the letter or phrase.

Return all customers with a name that ends with the phrase 'en':

```
SELECT * FROM customers
WHERE customer_name LIKE '%en';
```

customer_id	customer_name	contact_name	address	city	postal_code	country
6	Blauer See Delikatessen	Hanna Moos	Forsterstr. 57	Mannheim	68306	Germany
39	Koniglich Essen	Philip Cramer	Maubelstr. 90	Brandenburg	14776	Germany
55	Old World Delicatessen	Rene Phillips	2743 Bering St.	Anchorage	99508	USA
56	Ottilies Keseladen	Henriette Pfalzheim	Mehrheimerstr. 369	Koln	50739	Germany
79	Toms Spezialiteten	Karin Josephs	Luisenstr. 48	Manster	44087	Germany

(5 rows)

The Undescore _ Wildcard

The _ wildcard represents a single character.

It can be any character or number, but each _ represents one, and only one, character.

Return all customers from a `city` that starts with 'L' followed by one wildcard character, then 'nd' and then two wildcard characters:

```
SELECT * FROM customers
WHERE city LIKE 'L_nd__';
```

IN Operator

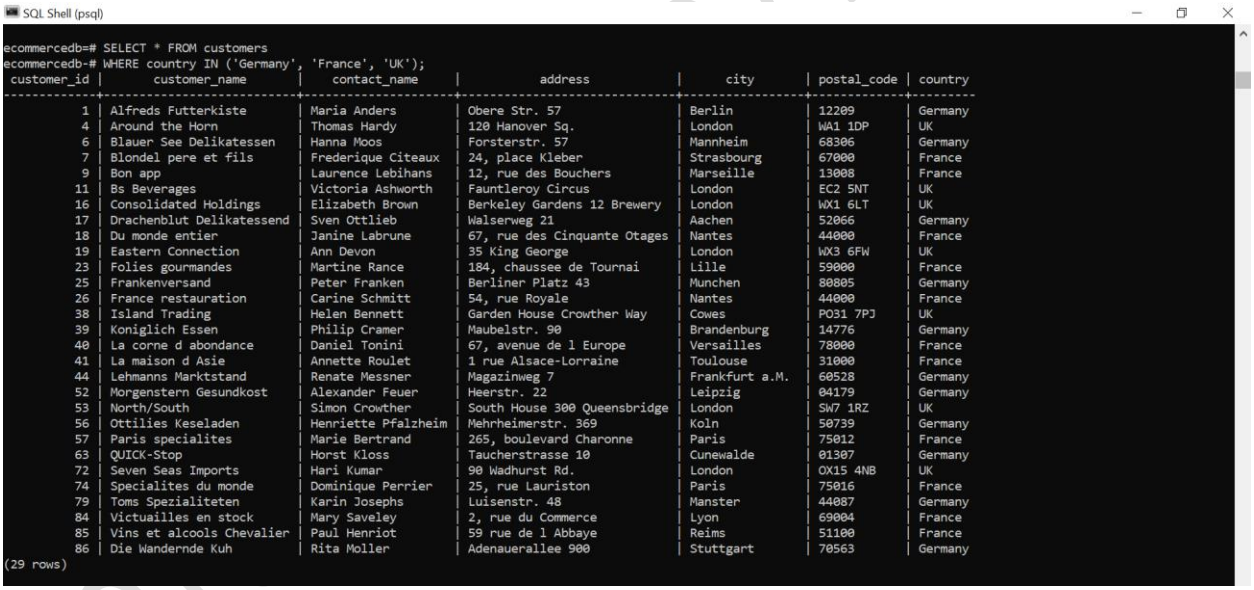
IN

The `IN` operator allows you to specify a list of possible values in the WHERE clause.

The `IN` operator is a shorthand for multiple `OR` conditions.

Return all customers from 'Germany', France' or 'UK':

```
SELECT * FROM customers
WHERE country IN ('Germany', 'France', 'UK');
```

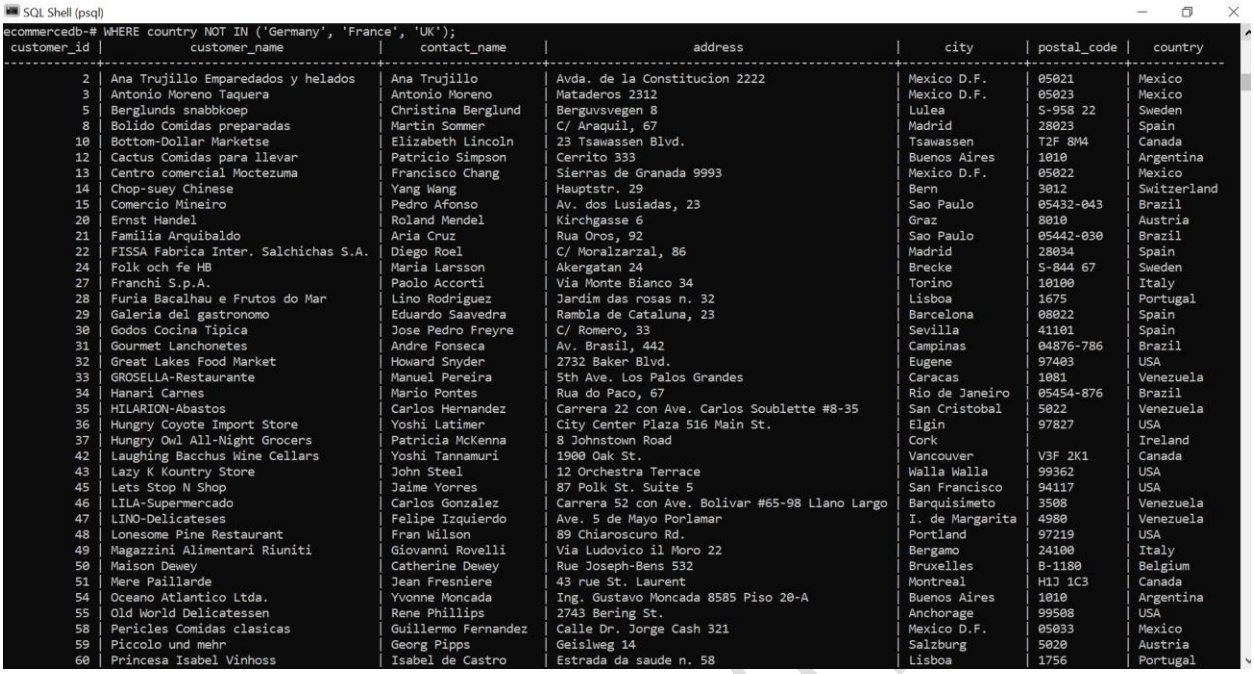


NOT IN

By using the `NOT` keyword in front of the `IN` operator, you return all records that are NOT any of the values in the list.

Return all customers that are NOT from 'Germany', France' or 'UK':

```
SELECT * FROM customers
WHERE country NOT IN ('Germany', 'France', 'UK');
```

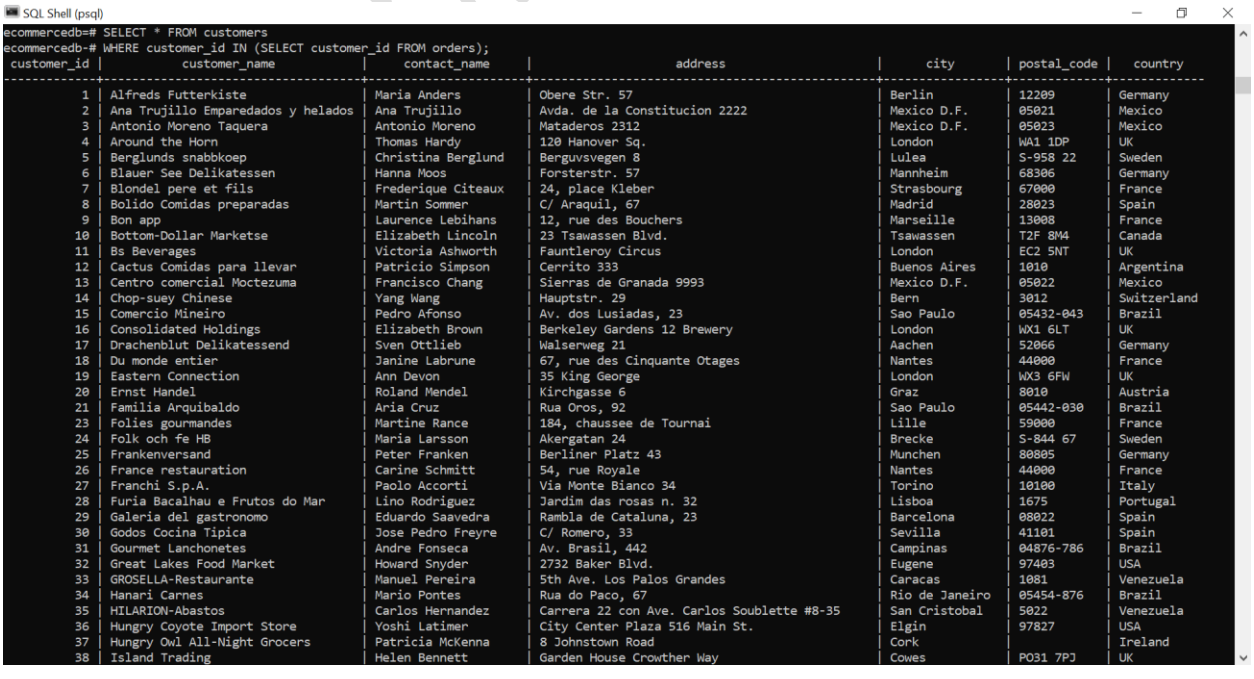


IN (SELECT)

You can also use a `SELECT` statement inside the parenthesis to return all records that are in the result of the `SELECT` statement.

Return all customers that have an order in the `orders` table:

```
SELECT * FROM customers
WHERE customer_id IN (SELECT customer_id FROM orders);
```



NOT IN (SELECT)

The result in the example above returned 89 records, that means that there are 2 customers that haven't placed any orders.

Let us check if that is correct, by using the `NOT IN` operator.

Return all customers that have NOT placed any orders in the `orders` table:

```
SELECT * FROM customers
WHERE customer_id NOT IN (SELECT customer_id FROM orders);
```

BETWEEN Operator

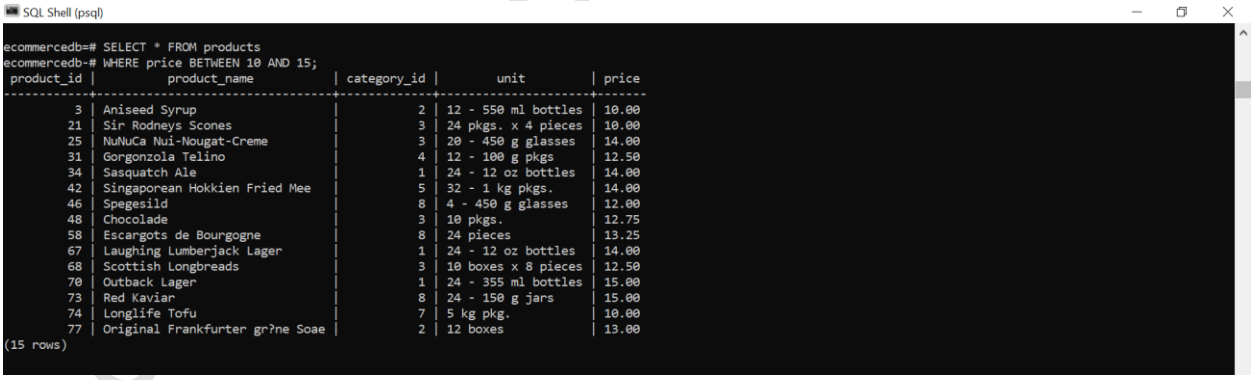
BETWEEN

The `BETWEEN` operator selects values within a given range. The values can be numbers, text, or dates.

The `BETWEEN` operator is inclusive: begin and end values are included.

Select all products with a `price` between 10 and 15:

```
SELECT * FROM products
WHERE price BETWEEN 10 AND 15;
```



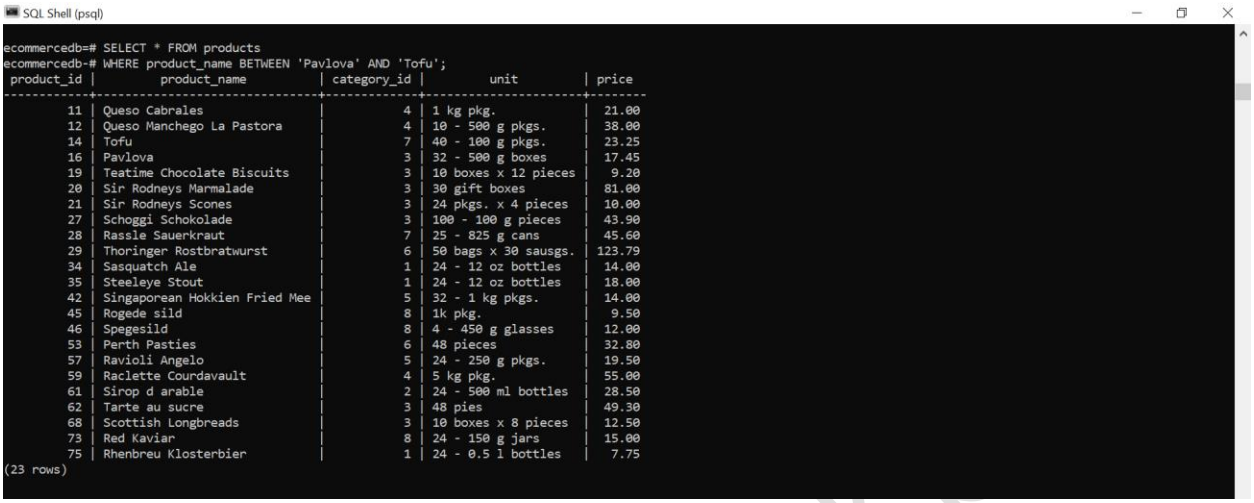
BETWEEN Text Values

The `BETWEEN` operator can also be used on text values.

The result returns all records that are *alphabetically* between the specified values.

Select all products between 'Pavlova' and 'Tofu':

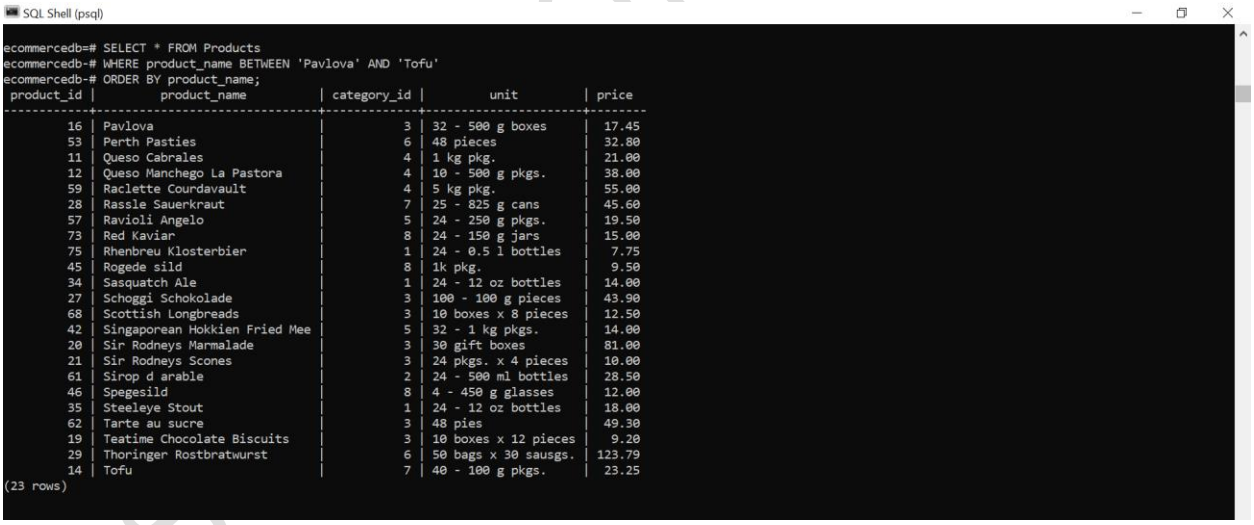
```
SELECT * FROM products
WHERE product_name BETWEEN 'Pavlova' AND 'Tofu';
```



If we add an `ORDER BY` clause to the example above, it will be a bit easier to read:

Same example as above, but we sort it by `product_name`:

```
SELECT * FROM Products
WHERE product_name BETWEEN 'Pavlova' AND 'Tofu'
ORDER BY product_name;
```



BETWEEN Date Values

The `BETWEEN` operator can also be used on date values.

Select all `orders` between 27. of April 2023 and 5. of May 2023:

```
SELECT * FROM orders
WHERE order_date BETWEEN '2023-04-27’ AND '2023-05-05';
```

AS For Aliases

Aliases

SQL aliases are used to give a table, or a column in a table, a temporary name.

Aliases are often used to make column names more readable.

An alias only exists for the duration of that query.

An alias is created with the `AS` keyword.

Using aliases for columns:

```
SELECT customer_id AS id
FROM customers;
```

This statement returns all customer id, which means 91 rows will be found.

AS is Optional

Actually, you can skip the `AS` keyword and get the same result:

Same result without `AS`:

```
SELECT customer_id id
FROM customers;
```

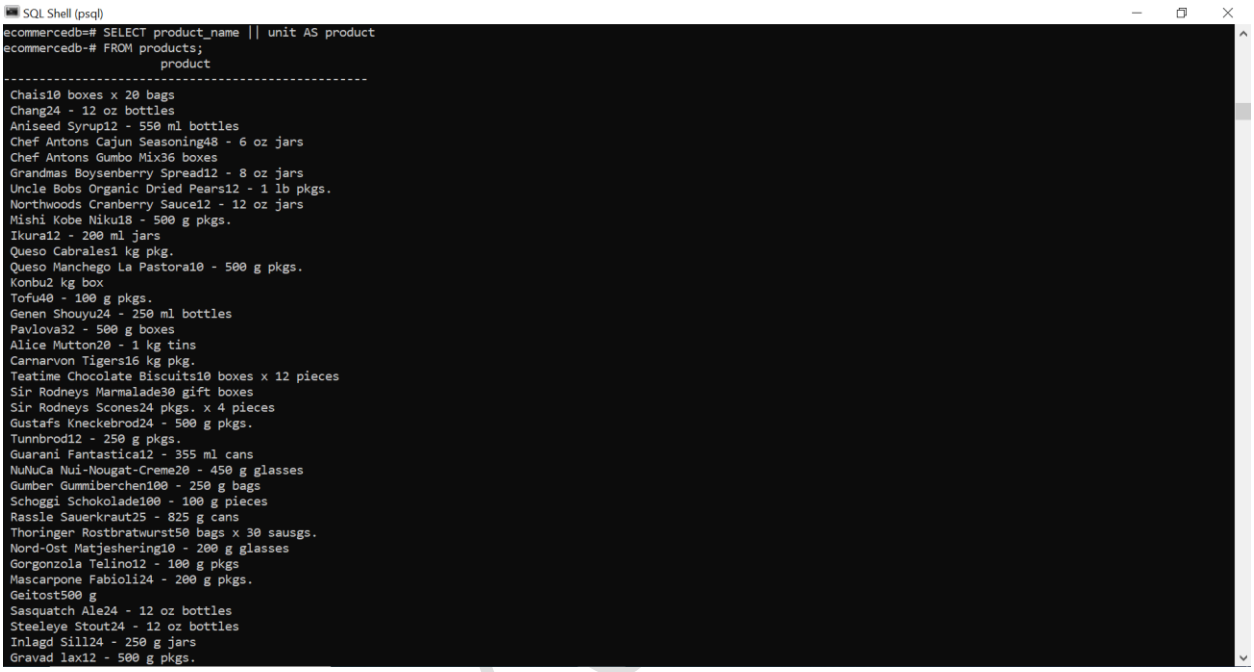
Concatenate Columns

The `AS` keyword is often used when two or more fields are concatenated into one.

To concatenate two fields use `||`.

Concatenate two fields and call them `product`:

```
SELECT product_name || unit AS product
FROM products;
```

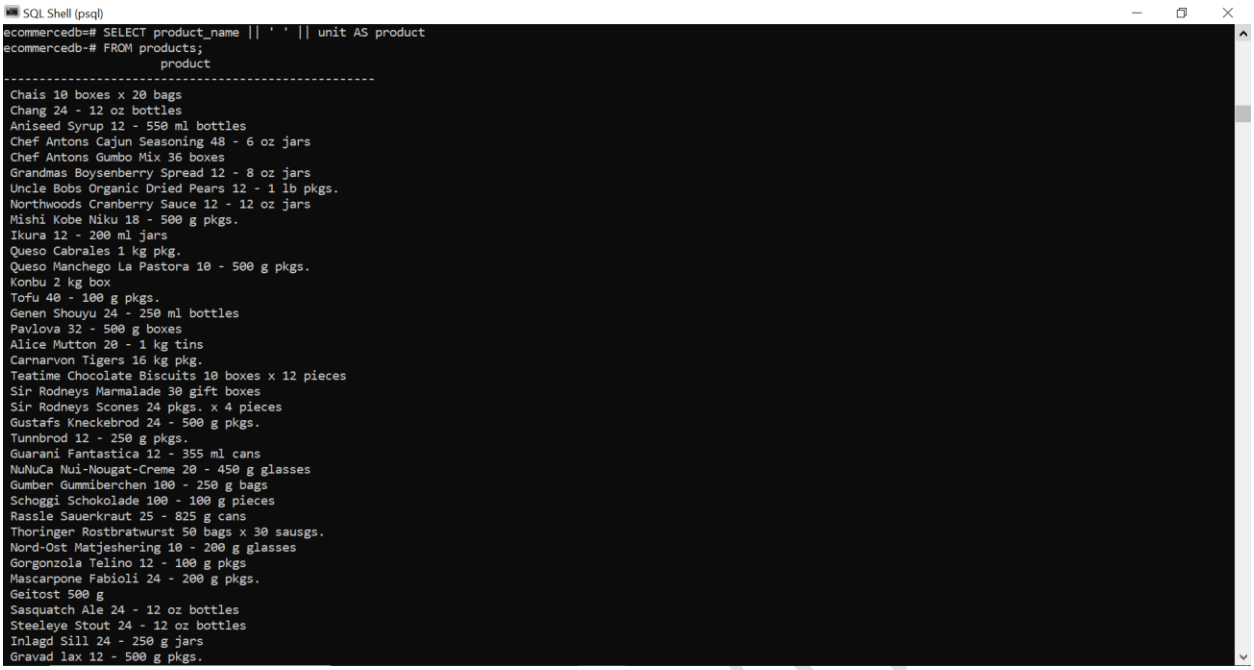


Note: In the result of the example above we are missing a space between product_name and unit. To add a space when concatenating, use `|| ' ' ||`.

Concatenate, with space:

```
SELECT product_name || ' ' || unit AS product
FROM products;
```

For this statement, we will get readable output.

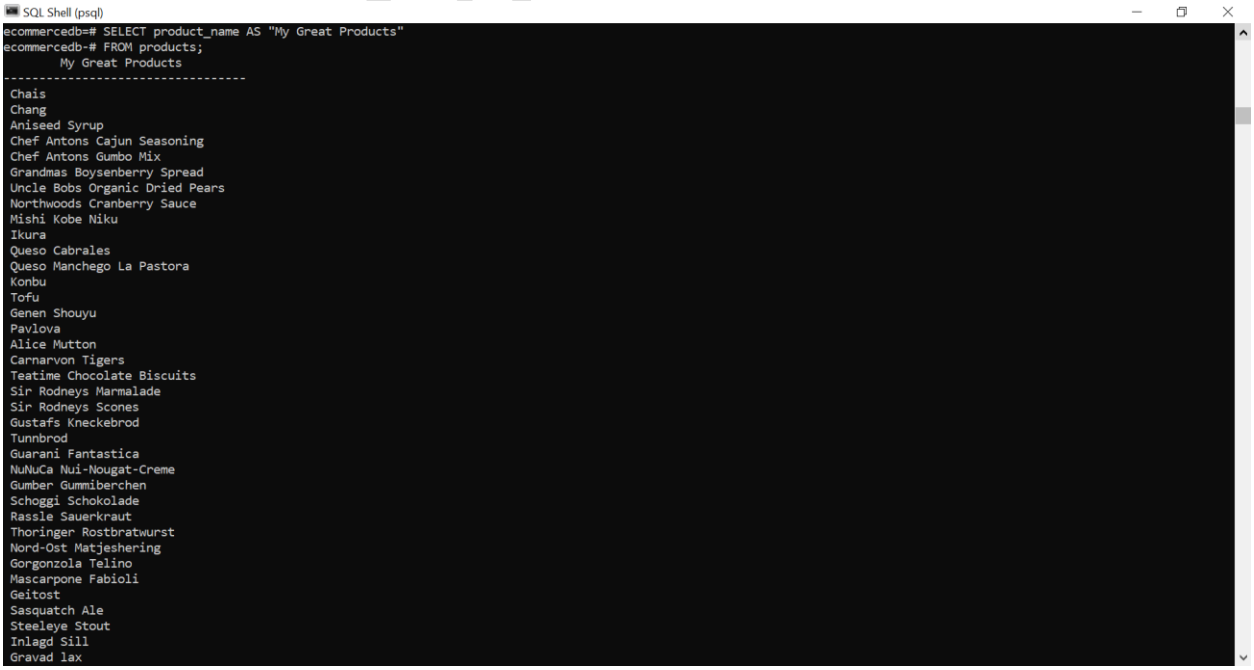


Using Aliases With a Space Character

If you want your alias to contain one or more spaces, like "`My Great Products`", surround your alias with double quotes.

Surround your alias with double quotes:

```
SELECT product_name AS "My Great Products"
FROM products;
```



JOINS

JOIN

A `JOIN` clause is used to combine rows from two or more tables, based on a related column between them.

Let's look at a selection from the `products` table:

```
product_id |  product_name  | category_id
  ------------+----------------------+-------------
         33 | Geitost          |        4
         34 | Sasquatch Ale    |        1
         35 | Steeleye Stout   |        1
         36 | Inlagd Sill      |        8
```

Then, look at a selection from the `categories` table:

```
category_id | category_name
  -------------+----------------
          1 | Beverages
          2 | Condiments
          3 | Confections
          4 | Dairy Products
```

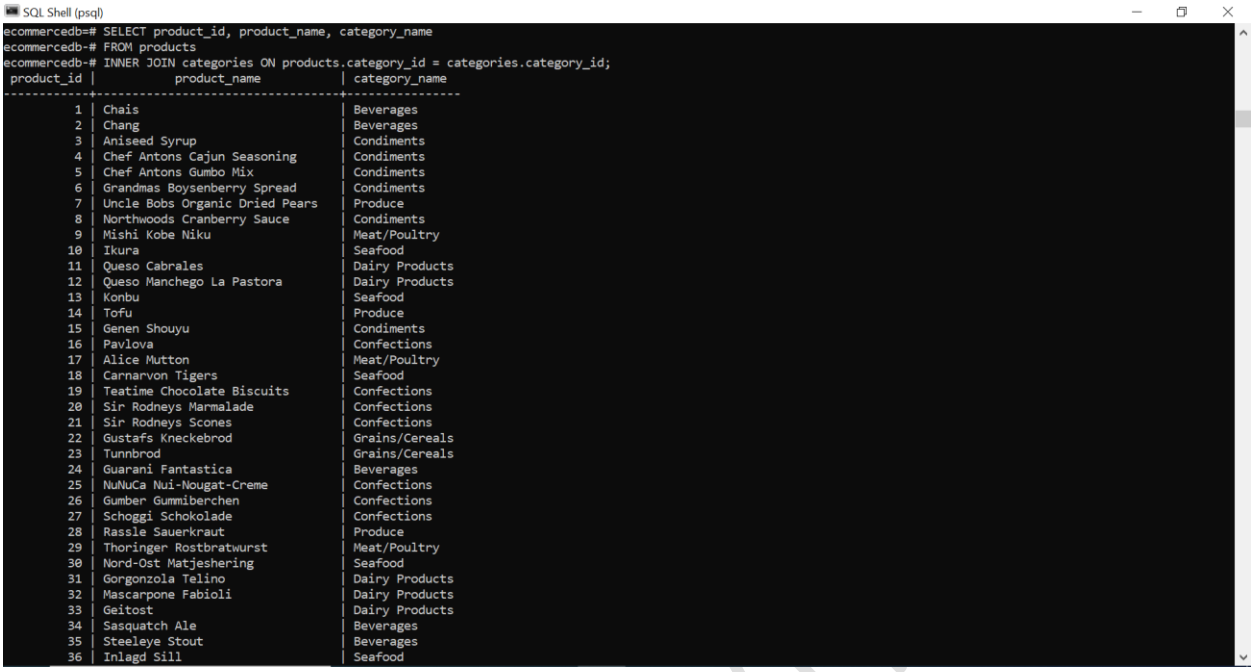
Notice that the `category_id` column in the `products` table refers to the `category_id` in the `categories` table. The relationship between the two tables above is the `category_id` column.

Then, we can create the following SQL statement (with a JOIN), that selects records that have matching values in both tables:

Join `products` to `categories` using the `category_id` column:

```
SELECT product_id, product_name, category_name
FROM products
INNER JOIN categories ON products.category_id = categories.category_id;
```

If we pull out the same selection from products table above, we get this result:



Different Types of Joins

Here are the different types of the Joins in PostgreSQL:

- `INNER JOIN`: Returns records that have matching values in both tables
- `LEFT JOIN`: Returns all records from the left table, and the matched records from the right table
- `RIGHT JOIN`: Returns all records from the right table, and the matched records from the left table
- `FULL JOIN`: Returns all records when there is a match in either left or right table

INNER JOIN

INNER JOIN Keyword

The `INNER JOIN` keyword selects records that have matching values in both tables.

Let's look at an example using our dummy `testproducts` table:

	testproduct_id [PK] integer	product_name character varying (255)	category_id integer
1	1	Johns Fruit Cake	3
2	2	Marys Healthy Mix	9
3	3	Peters Scary Stuff	10
4	4	Jims Secret Recipe	11
5	5	Elisabeths Best Apples	12
6	6	Janes Favorite Cheese	4
7	7	Billys Home Made Pizza	13
8	8	Ellas Special Salmon	8
9	9	Roberts Rich Spaghetti	5
10	10	Mias Popular Ice	14

Total rows: 10 of 10 | Query complete 00:00:00.093

We will try to join the `testproducts` table with the `categories` table:

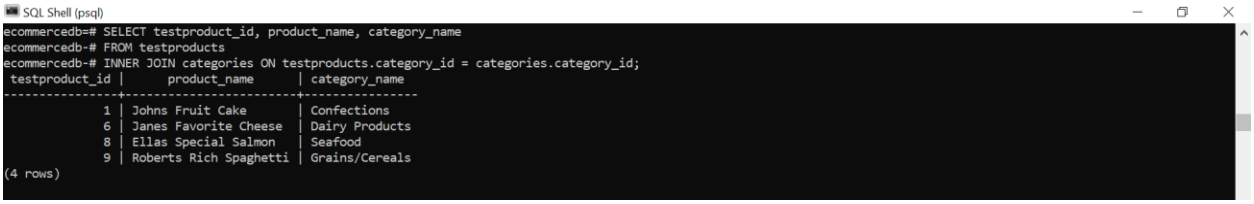
	category_id [PK] integer	category_name character varying (255)	description character varying (255)
1	1	Beverages	Soft drinks, coffees, teas, beers, and ales
2	2	Condiments	Sweet and savory sauces, relishes, spreads, and seasonin…
3	3	Confections	Desserts, candies, and sweet breads
4	4	Dairy Products	Cheeses
5	5	Grains/Cereals	Breads, crackers, pasta, and cereal
6	6	Meat/Poultry	Prepared meats
7	7	Produce	Dried fruit and bean curd
8	8	Seafood	Seaweed and fish

Notice that many of the products in `testproducts` have a `category_id` that does not match any of the categories in the `categories` table.

By using `INNER JOIN` we will not get the records where there is not a match, we will only get the records that matches *both* tables:

Join `testproducts` to `categories` using the `category_id` column:

```
SELECT testproduct_id, product_name, category_name
FROM testproducts
INNER JOIN categories ON testproducts.category_id =
categories.category_id;
```



Note: `JOIN` and `INNER JOIN` will give the same result. `INNER` is the default join type for `JOIN`, so when you write `JOIN` the parser actually writes `INNER JOIN`.

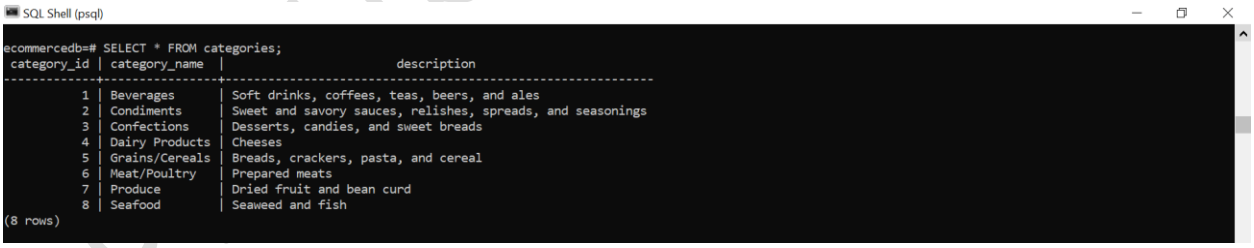
LEFT JOIN

LEFT JOIN Keyword

The `LEFT JOIN` keyword selects ALL records from the "left" table, and the matching records from the "right" table. The result is 0 records from the right side if there is no match.

Let's look at an example using our dummy `testproducts` table:

We will try to join the `testproducts` table with the `categories` table:



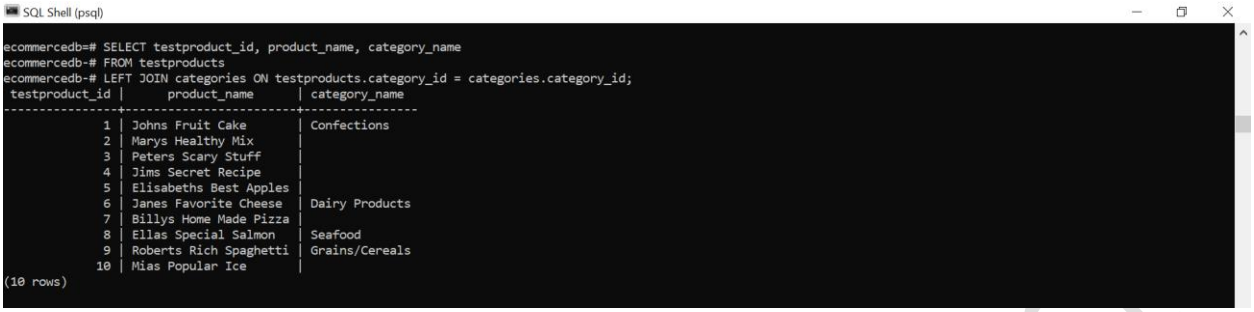
Note: Many of the products in `testproducts` have a `category_id` that does not match any of the categories in the `categories` table.

By using `LEFT JOIN` we will get all records from `testpoducts`, even the ones with no match in the `categories` table:

`Join testproducts to categories using the category_id column:`

```
SELECT testproduct_id, product_name, category_name
FROM testproducts
LEFT JOIN categories ON testproducts.category_id = categories.category_id;
```

All records from `testproducts`, and only the matched records from `categories`:



Note: `LEFT JOIN` and `LEFT OUTER JOIN` will give the same result. `OUTER` is the default join type for `LEFT JOIN`, so when you write `LEFT JOIN` the parser actually writes `LEFT OUTER JOIN`.

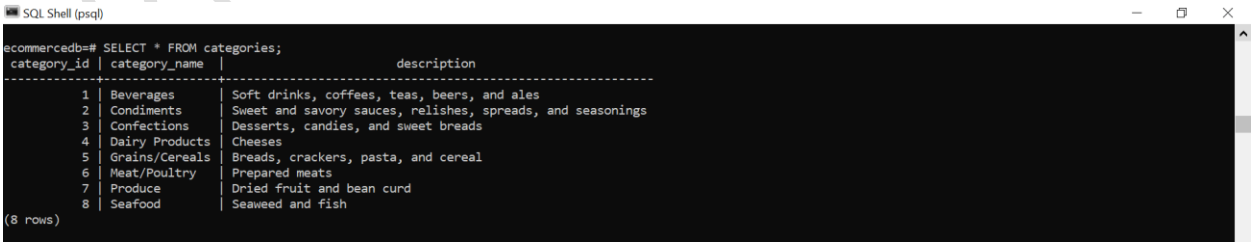
RIGHT JOIN

RIGHT JOIN Keyword

The `RIGHT JOIN` keyword selects ALL records from the "right" table, and the matching records from the "left" table. The result is 0 records from the left side if there is no match.

Let's look at an example using our dummy `testproducts` table:

We will try to join the `testproducts` table with the `categories` table:



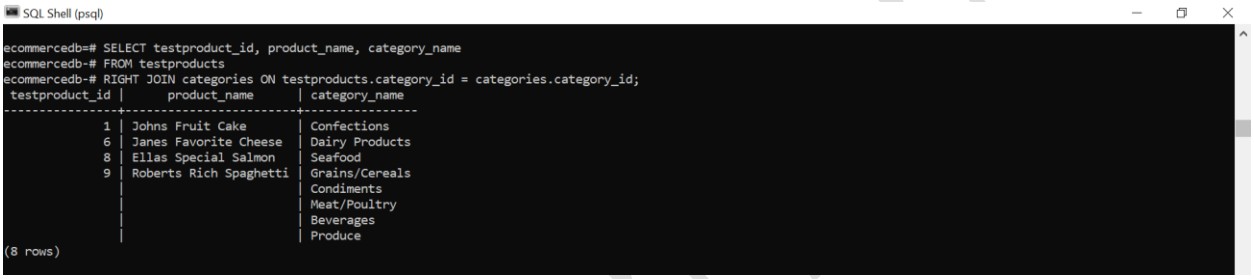
Note: Many of the products in `testproducts` have a `category_id` that does not match any of the categories in the `categories` table.

By using `RIGHT JOIN` we will get all records from `categories`, even the ones with no match in the `testproducts` table:

Join `testproducts` to `categories` using the `category_id` column:

```
SELECT testproduct_id, product_name, category_name
FROM testproducts
RIGHT JOIN categories ON testproducts.category_id =
categories.category_id;
```

All records from `categories`, and only the matched records from `testproducts`:



Note: `RIGHT JOIN` and `RIGHT OUTER JOIN` will give the same result. `OUTER` is the default join type for `RIGHT JOIN`, so when you write `RIGHT JOIN` the parser actually writes `RIGHT OUTER JOIN`.

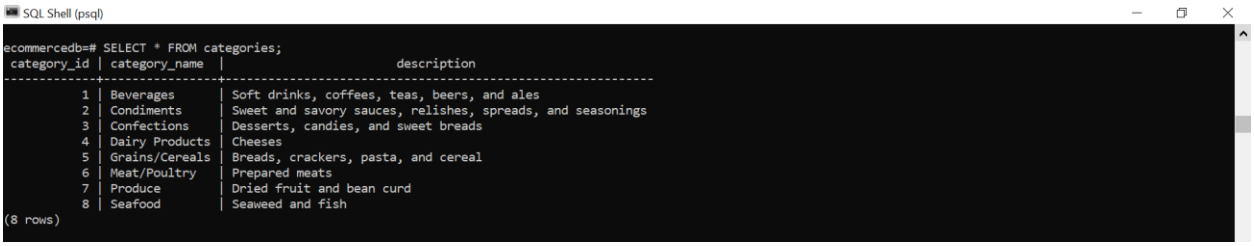
FULL JOIN

FULL JOIN Keyword

The `FULL JOIN` keyword selects ALL records from both tables, even if there is not a match. For rows with a match the values from both tables are available, if there is not a match the empty fields will get the value `NULL`.

Let's look at an example using our dummy `testproducts` table:

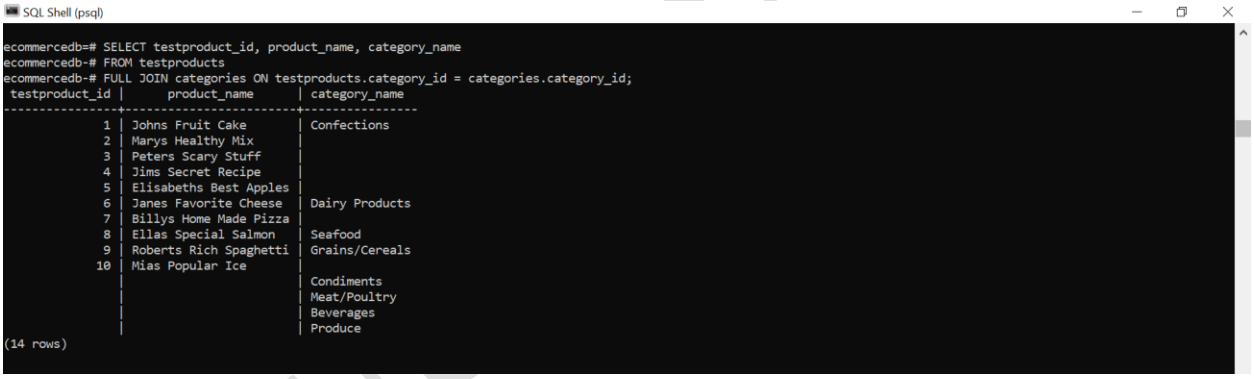
We will try to join the `testproducts` table with the `categories` table:



Join `testproducts` to `categories` using the `category_id` column:

```
SELECT testproduct_id, product_name, category_name
FROM testproducts
FULL JOIN categories ON testproducts.category_id =
categories.category_id;
```

All records from both tables are returned. Rows with no match will get a NULL value in fields from the opposite table:



Here, testproducts has 10 rows and categories has 8 rows. So, total rows are 18. But matched row is 4. They are matched with each other according to categories_id. So, this statement returns 14 rows.

Returned Row = (Table1's Row + Table2's Row) – (Matched Row Between Two Tables)

CROSS JOIN

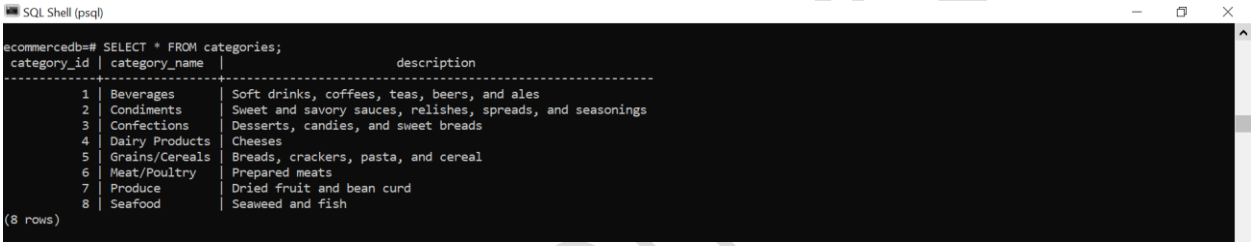
CROSS JOIN Keyword

The `CROSS JOIN` keyword matches ALL records from the "left" table with EACH record from the "right" table. That means that all records from the "right" table will be returned for each record in the "left" table.

This way of joining can potentially return very large table, and you should not use it if you do not have to.

Let's look at an example using our dummy `testproducts` table:

We will try to join the `testproducts` table with the `categories` table:

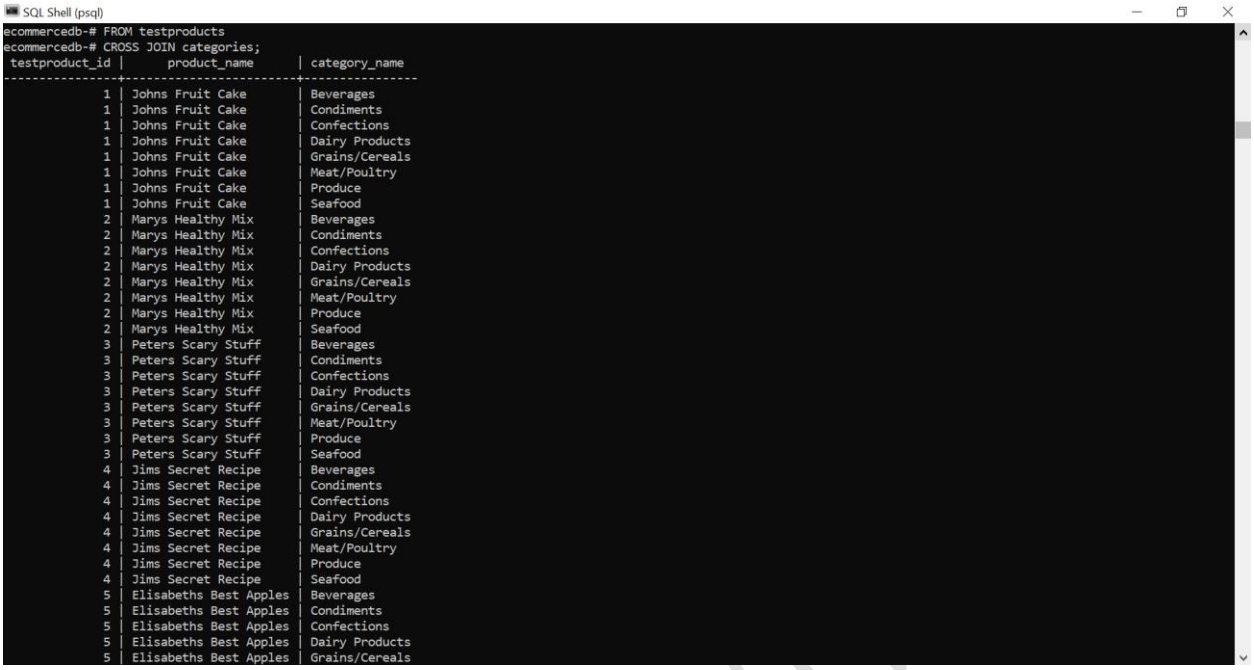


Note: The `CROSS JOIN` method will return ALL `categories` for EACH `testproduct`, meaning that it will return 80 rows (10 * 8).

Join `testproducts` to `categories` using the `CROSS JOIN` keyword:

```
SELECT testproduct_id, product_name, category_name
FROM testproducts
CROSS JOIN categories;
```

All `categories` for each `testproduct` will be returned:



UNION Operator

UNION

The `UNION` operator is used to combine the result-set of two or more queries.

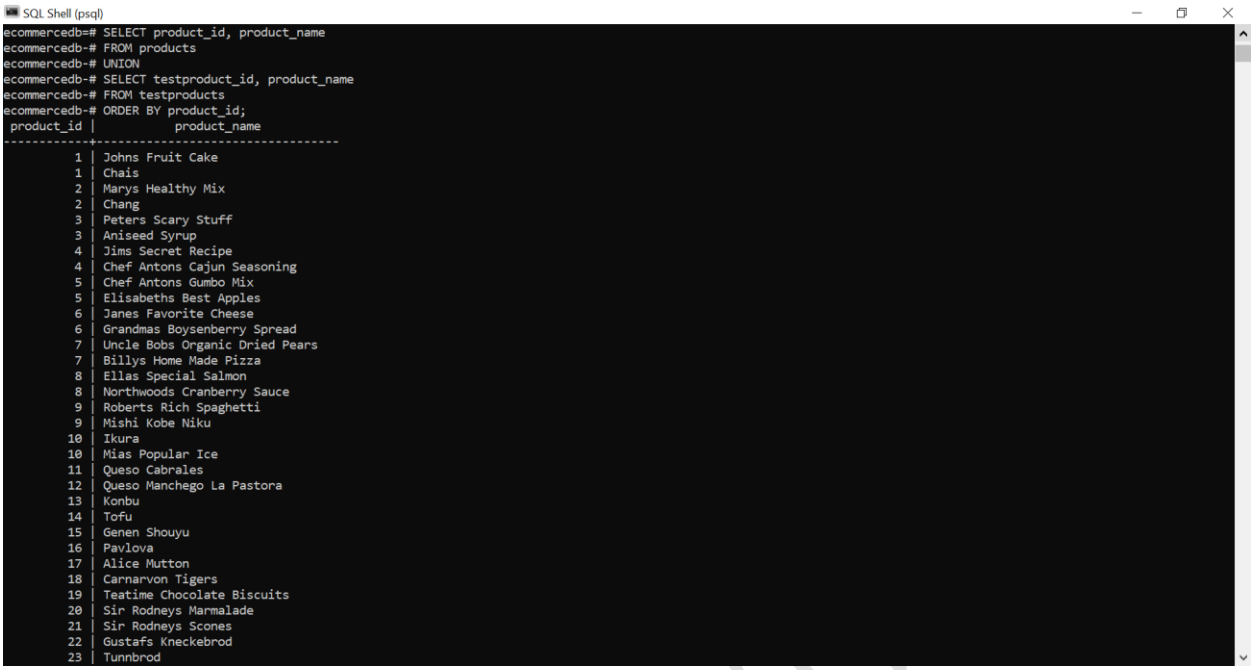
The queries in the union must follow these rules:

- They must have the same number of columns
- The columns must have the same data types
- The columns must be in the same order

Combine `products` and `testproducts` using the `UNION` operator:

```
SELECT product_id, product_name
FROM products
UNION
SELECT testproduct_id, product_name
FROM testproducts
ORDER BY product_id;
```

This statement returns the output as shown in the figure below. Where `testproducts` table row shows first, then `products` table. Because we have used `ORDER BY` keyword. We will see 87 rows in our SQL Shell.



UNION vs UNION ALL

With the `UNION` operator, if some rows in the two queries returns the exact same result, only one row will be listed, because `UNION` selects only distinct values.

Use `UNION ALL` to return duplicate values.

UNION

Let's make some changes to the queries, so that we have duplicate values in the result:

```
SELECT product_id
FROM products
UNION
SELECT testproduct_id
FROM testproducts
ORDER BY product_id;
```

This statement returns 77 rows that have no duplicates in it. The `UNION` operator merges two tables without duplicate values.

UNION ALL

```
SELECT product_id
FROM products
UNION ALL
SELECT testproduct_id
FROM testproducts
ORDER BY product_id;
```

This statement returns 87 rows that have duplicates in it. The `UNION ALL` operator merges two tables with duplicate values.

GROUP BY

GROUP BY Clause

The `GROUP BY` clause groups rows that have the same values into summary rows, like "find the number of customers in each country".

The `GROUP BY` clause is often used with aggregate functions like `COUNT()`, `MAX()`, `MIN()`, `SUM()`, `AVG()` to group the result-set by one or more columns.

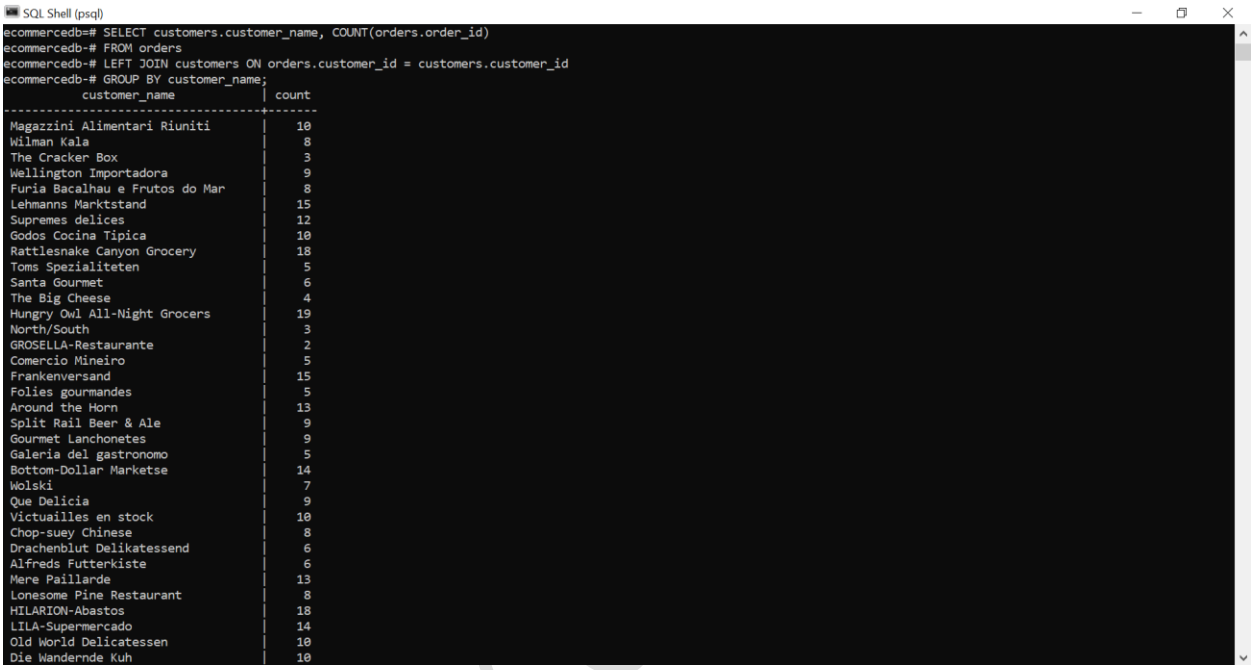
Lists the number of customers in each country:

```
SELECT COUNT(customer_id), country
FROM customers
GROUP BY country;
```

GROUP BY With JOIN

The following SQL statement lists the number of orders made by each customer:

```
SELECT customers.customer_name, COUNT(orders.order_id)
FROM orders
LEFT JOIN customers ON orders.customer_id = customers.customer_id
GROUP BY customer_name;
```



This statement returns 89 rows. Some of them are given above.

HAVING Clause

HAVING

The `HAVING` clause was added to SQL because the `WHERE` clause cannot be used with aggregate functions.

Aggregate functions are often used with `GROUP BY` clauses, and by adding `HAVING` we can write condition like we do with `WHERE` clauses.

List only countries that are represented more than 5 times:

```
SELECT COUNT(customer_id), country
FROM customers
GROUP BY country
HAVING COUNT(customer_id) > 5;
```

```
SQL Shell (psql)
ecommercedb=# SELECT COUNT(customer_id), country
ecommercedb-# FROM customers
ecommercedb-# GROUP BY country
ecommercedb-# HAVING COUNT(customer_id) > 5;
 count | country
-------+---------
    13 | USA
    11 | France
     9 | Brazil
     7 | UK
    11 | Germany
(5 rows)
```

More HAVING Examples

The following SQL statement lists only orders with a total price of 400$ or more:

```
SELECT order_details.order_id, SUM(products.price)
FROM order_details
LEFT JOIN products ON order_details.product_id = products.product_id
GROUP BY order_id
HAVING SUM(products.price) > 400.00;
```

```
SQL Shell (psql)
ecommercedb=# SELECT order_details.order_id, SUM(products.price)
ecommercedb-# FROM order_details
ecommercedb-# LEFT JOIN products ON order_details.product_id = products.product_id
ecommercedb-# GROUP BY order_id
ecommercedb-# HAVING SUM(products.price) > 400.00;
 order_id |  sum
----------+--------
    10360 | 460.34
    11077 | 574.35
    10372 | 413.30
(3 rows)
```

Lists customers that have ordered for 1000$ or more:

```
SELECT customers.customer_name, SUM(products.price)
FROM order_details
LEFT JOIN products ON order_details.product_id = products.product_id
LEFT JOIN orders ON order_details.order_id = orders.order_id
LEFT JOIN customers ON orders.customer_id = customers.customer_id
GROUP BY customer_name
HAVING SUM(products.price) > 1000.00;
```

```
SQL Shell (psql)
ecommercedb=# SELECT customers.customer_name, SUM(products.price)
ecommercedb-# FROM order_details
ecommercedb-# LEFT JOIN products ON order_details.product_id = products.product_id
ecommercedb-# LEFT JOIN orders ON order_details.order_id = orders.order_id
ecommercedb-# LEFT JOIN customers ON orders.customer_id = customers.customer_id
ecommercedb-# GROUP BY customer_name
ecommercedb-# HAVING SUM(products.price) > 1000.00;
       customer_name        |   sum
----------------------------+---------
 Lehmanns Marktstand        | 1088.29
 Rattlesnake Canyon Grocery | 2399.99
 Hungry Owl All-Night Grocers | 1837.95
 Frankenversand             | 1067.41
 Mere Paillarde             | 1072.50
 HILARION-Abastos           | 1001.47
 Ernst Handel               | 2891.35
 Bon app                    | 1122.55
 Great Lakes Food Market    | 1091.54
 Save-a-lot Markets         | 2788.48
 Queen Cozinha              | 1166.59
 Koniglich Essen            | 1290.48
 QUICK-Stop                 | 2842.45
 Hanari Carnes              | 1423.00
 Folk och fe HB             | 1098.58
 White Clover Markets       | 1314.93
 Berglunds snabbkoep        | 1507.48
(17 rows)
```

EXISTS Operator

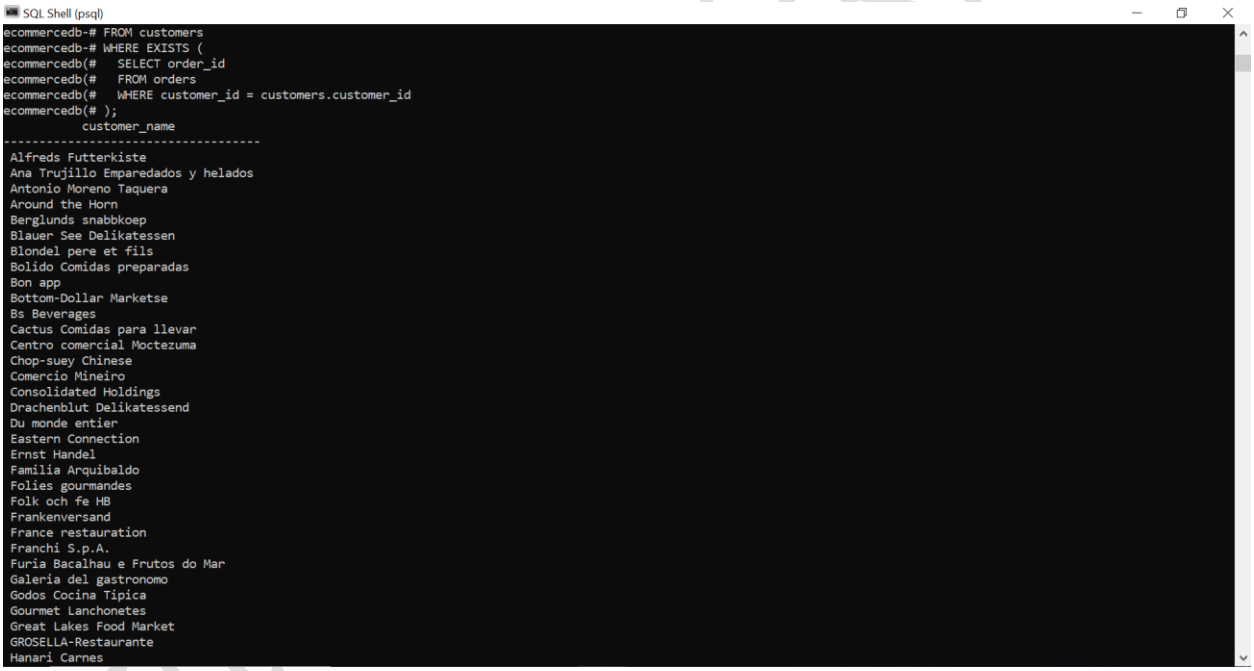
EXISTS

The `EXISTS` operator is used to test for the existence of any record in a sub query.

The `EXISTS` operator returns TRUE if the sub query returns one or more records.

Return all customers that is represented in the `orders` table:

```
SELECT customers.customer_name
FROM customers
WHERE EXISTS (
  SELECT order_id
  FROM orders
  WHERE customer_id = customers.customer_id
);
```



The result in example above showed that 89 customers had at least one order in the `orders` table.

NOT EXISTS

To check which customers that do not have any orders, we can use the `NOT` operator together with the `EXISTS` operator:

Return all customers that is NOT represented in the `orders` table:

```
SELECT customers.customer_name
FROM customers
```

```
WHERE NOT EXISTS (
  SELECT order_id
  FROM orders
  WHERE customer_id = customers.customer_id
);
```

This statement will be returned 2 rows, because the `customers` table has 91 rows. Since the above example returned 89 rows. This query is alternative of above example.

ANY Operator

ANY

The `ANY` operator allows you to perform a comparison between a single column value and a range of other values.

The `ANY` operator:

- returns a Boolean value as a result
- returns TRUE if ANY of the sub query values meet the condition

`ANY` means that the condition will be true if the operation is true for any of the values in the range.

List products that have ANY records in the `order_details` table with a quantity larger than 120:

```
SELECT product_name
FROM products
WHERE product_id = ANY (
  SELECT product_id
  FROM order_details
  WHERE quantity > 120
);
```

ALL Operator

ALL

The `ALL` operator:

- returns a Boolean value as a result
- returns TRUE if ALL of the sub query values meet the condition
- is used with `SELECT`, `WHERE` and `HAVING` statements

`ALL` means that the condition will be true only if the operation is true for all values in the range.

List the products if ALL the records in the `order_details` with quantity larger than 10.

```
SELECT product_name
FROM products
WHERE product_id = ALL (
  SELECT product_id
  FROM order_details
  WHERE quantity > 10
);
```

Note: This will of course return FALSE because the quantity column has many different values (not only the value of 10):

CASE Expression

CASE

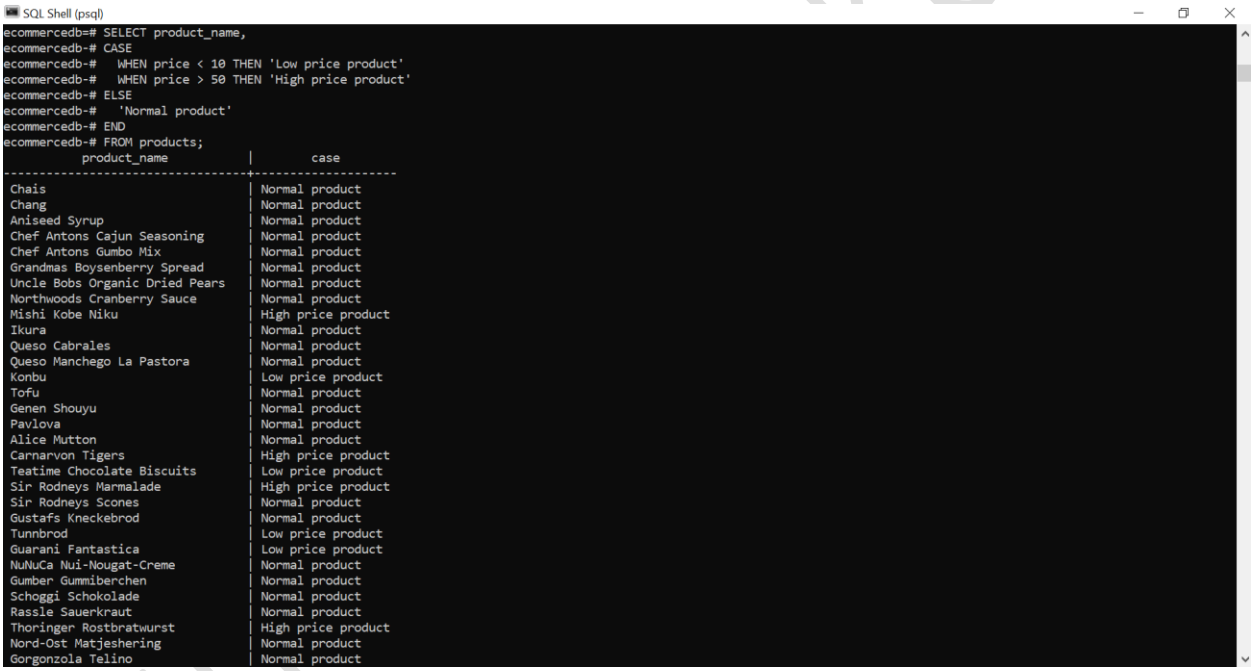
The `CASE` expression goes through conditions and returns a value when the first condition is met (like an if-then-else statement).

Once a condition is true, it will stop reading and return the result. If no conditions are true, it returns the value in the `ELSE` clause.

If there is no `ELSE` part and no conditions are true, it returns NULL.

Return specific values if the price meets a specific condition:

```
SELECT product_name,
CASE
  WHEN price < 10 THEN 'Low price product'
  WHEN price > 50 THEN 'High price product'
ELSE
  'Normal product'
END
FROM products;
```



This statement returns 77 rows with `Case` results.

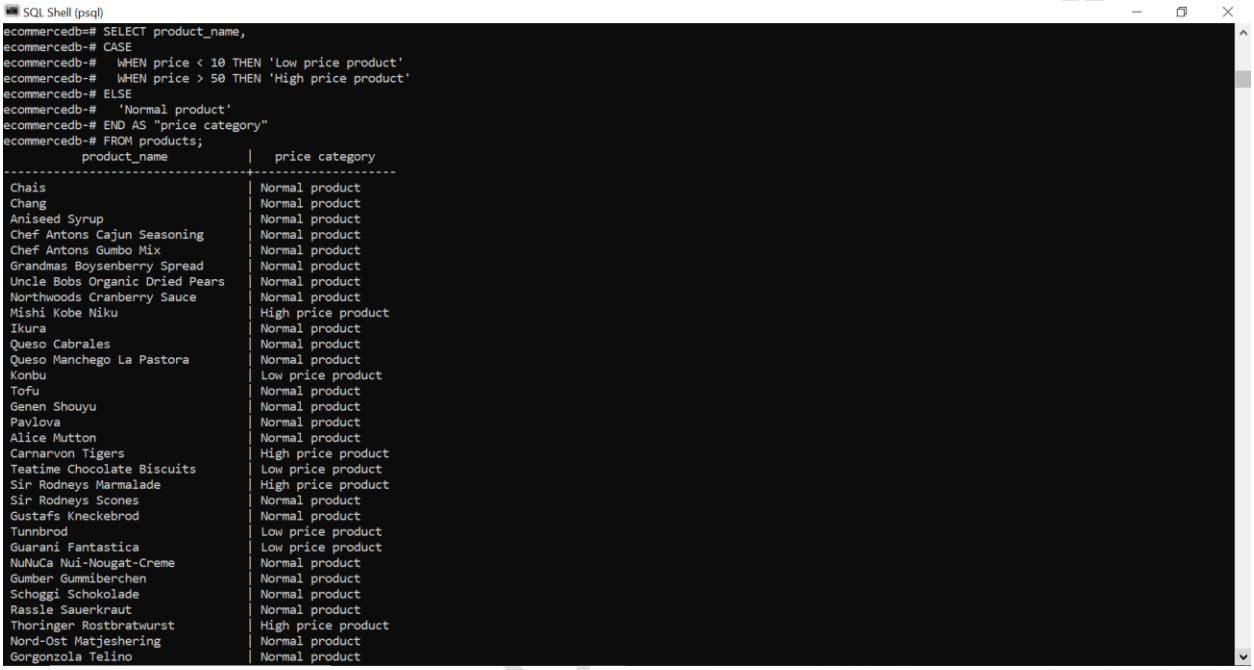
With an Alias

When a column name is not specified for the "case" field, the parser uses `case` as the column name. To specify a column name, add an alias after the `END` keyword.

Same example, but with an alias for the case column:

```
SELECT product_name,
CASE
```

```
  WHEN price < 10 THEN 'Low price product'
  WHEN price > 50 THEN 'High price product'
ELSE
  'Normal product'
END AS "price category"
FROM products;
```



Follow Me:

https://github.com/rkpust

https://www.linkedin.com/in/rkpust

https://twitter.com/IamRezaul_Karim

**********The End**********