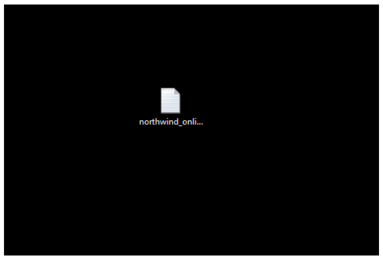
Practice SQL with PostgreSQL

Performing SQL queries for each of the 5 TRANSFORMATION actions required

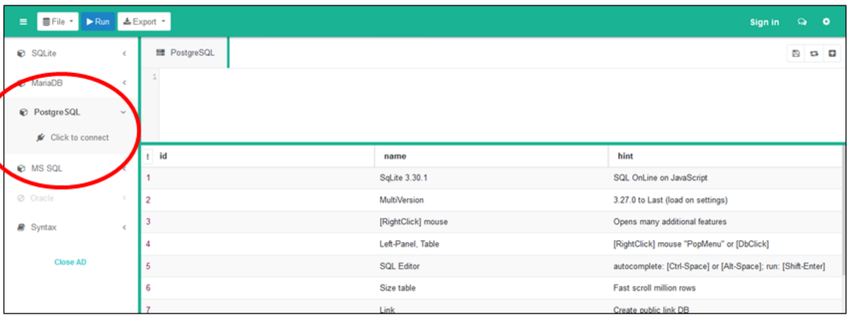
Read the instructions here after and follow the steps:

0. Requirements

On the loop page of the module, download the document called “northwind_onlinedemo.txt” on your desktop.

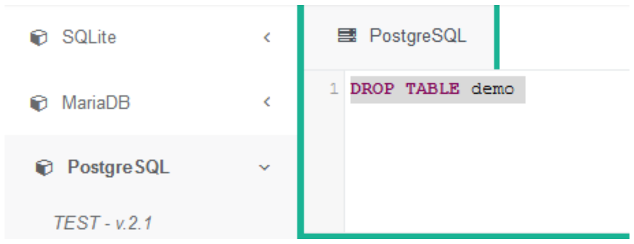


Open your web browser and go to https://sqliteonline.com/ (free emulation of SQL servers). In sqliteonline.com, click PostgreSQL (left box) and click “Click to connect”.



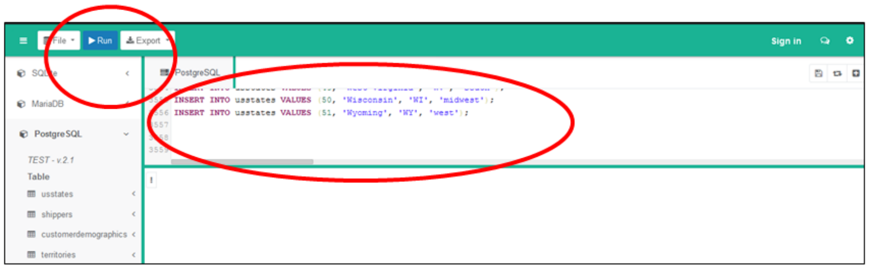
id	name	hint
1	SqLite 3.30.1	SQL OnLine on JavaScript
2	MultiVersion	3.27.0 to Last (load on settings)
3	[RightClick] mouse	Opens many additional features
4	Left-Panel, Table	[RightClick] mouse "PopMenu" or [DbClick]
5	SQL Editor	autocomplete: [Ctrl-Space] or [Alt-Space]; run: [Shift-Enter]
6	Size table	Fast scroll million rows
7	Link	Create public link DB

Optional: DROP demo Table (right click on demo, use DROP). If so, delete the text “DROP TABLE demo” that appeared in the main box.



```
DROP TABLE demo
```

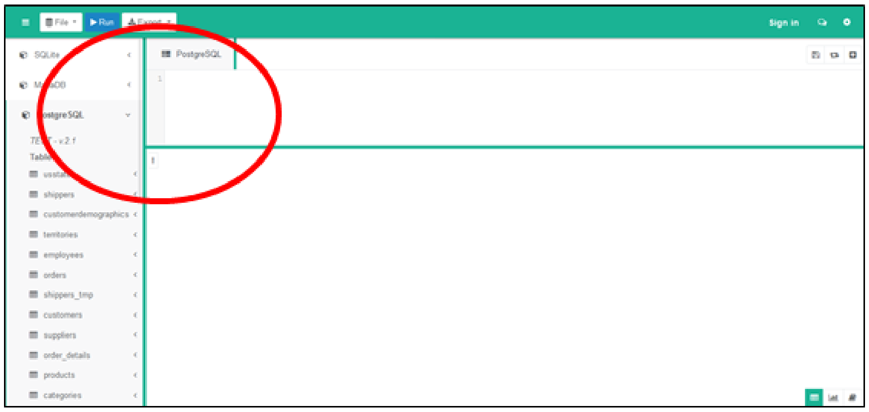
In the top box Copy-Paste the text of the file “northwind_onlinedemo.txt” (3556 lines) and press Run on the top menu bar (you should see the 15 table appears on the left box).



```
INSERT INTO usstates VALUES (50, 'Wisconsin', 'WI', 'midwest');
INSERT INTO usstates VALUES (51, 'Wyoming', 'WY', 'west');
```

If the operation was successful, 15 tables name should appear in the list of available tables.

Select all the lines in the box (`CTRL + A` or `Cmd + A`) and delete them. The first line should be empty as displayed here below.



Have a look at the first values of the Tables customers, products and orders by running the following SQL queries:

```sql
SELECT *
FROM customers;
```

```sql
SELECT *
FROM products;
```

```sql
SELECT *
FROM orders;
```

All the game of this tutorial will be to create new tables that can be downloaded for our analyses.

NOTE: In https://sqliteonline.com/, the server is allocated for a duration of 15min only. When this duration is expired, you will have to setup the database again.

1. Extension

First, let's start by just changing the name of variables. Even if the values of the variables did not change, it is always useful to know how to rename a variable.

Create 1 new column just by changing its name:

```sql
SELECT CustomerID AS customer
FROM customers;
```

Create 2 new columns just by changing their names:

```sql
SELECT CustomerID AS customer, ContactName AS customer_name
FROM customers;
```

Question 1: Why have we changed the column names in that way and how it is called?

Now, let's create a new variable by applying arithmetic operators and functions. Select the variable `ProductName` and create the variable `profit_max` which is the multiplication of `UnitsInStock` and `UnitPrice`:

```sql
SELECT ProductName, UnitsInStock * UnitPrice AS profit_max
FROM products;
```

Select the variable `ContactName` and create the variable `address_full` which is the result of the function `CONCAT()` applied to the variables `Address`, `PostalCode` and `City`:

```sql
SELECT ContactName, CONCAT(Address, ‘ ’, PostalCode, ‘ ’, City)  AS address_full
FROM customers;
```

Question 2: What is the result of the function `CONCAT()` and why are these arguments separated by ‘ ’?

2. Reduction

Now let's try to subset a small amount from a table by using a filter. Filter the values from the table `customers` by keeping only the values of the variable `Country` which equals to 'Mexico':

```sql
SELECT *
FROM customers
WHERE Country = 'Mexico';
```

Question 3: Why the character sting Mexico is between quotation marks " and not Country?

It is possible to add more filters with the `AND` and `OR` statements. Filter the values from the table `customers` by keeping only the values of the variable `Country` which equals to 'Mexico' or 'France'

```sql
SELECT *
FROM customers
WHERE Country = 'Mexico' OR Country = 'France';
```

Filters can be applied to character string and to numeric values as well. Filter the values from the table `products` to keep only the data from the product which ID is 1:

```sql
SELECT *
FROM products
WHERE ProductID = 1;
```

3. Direction

Ordering variables is not that useful in general but it can be a very efficient way to obtain specific values. First, try to order of the values from the variable `Country` of the `customers` table in alphabetical order:

```
SELECT *
FROM customers
ORDER BY Country;
```

Now, change the direction order by sorting the values from the variable `Country` of the `customers` table by descendant values:

```
SELECT *
FROM customers
ORDER BY Country DESC;
```

It is also possible to order values according two variables (ascendant and descendant):

```
SELECT *
FROM customers
ORDER BY Country ASC, ContactName DESC;
```

Question 4: for variables made of character string, to which order ASC and DESC are referring to?

Obviously ordering values can also be done for variables made of numeric values. Try to sort the variable `UnitPrice` from the `products` table in an ascending order:

```
SELECT ProductName, UnitsInStock, UnitPrice
FROM products;
ORDER BY UnitPrice ASC;
```

Try to do the same in a descending order:

```
SELECT ProductName, UnitsInStock, UnitPrice
FROM products;
ORDER BY UnitPrice DESC;
```

4. Aggregation

Instead of downloading a full table to compute a certain information/summary, it is possible to obtain it directly by SQL query. Basic summaries include `COUNT()`, `SUM()`, `AVG()`, `STDEV()`, `MIN()`, `MAX()` ...

First, use the function `COUNT()` on the variable `ContactName` from the `customers` table to obtain the total amount of customers:

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

Similarly, use the `AVG()` function on the variable `UnitPrice` from the `products` table to obtain the average price of the products sold by the company:

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

Aggregating data is very powerful when it is combined with the `GROUP BY` statement. Then, the aggregation will be performed for each distinct value of the grouping variable. Compute the amount of contacts by country with the following code:

```
SELECT COUNT(ContactName)
FROM customers
GROUP BY Country;
```

Question 4: what is the problem of the previous code?

Try again by selecting the variable Country as well:

```
SELECT COUNT(ContactName), Country
FROM customers
GROUP BY Country;
```

In the same way, compute the average price of the product sold by `CategoryID` from the `products` table:

```sql
SELECT AVG(UnitPrice), CategoryID
FROM products
GROUP BY CategoryID;
```

5. Combination

Because the information you need to access can be present in two different tables, sometimes it is necessary to join these tables together. The operation is obtained with the JOIN statement. As we have seen in the lecture, the four main join operation are LEFT, RIGHT, INNER and FULL.

Join works as follow: first define the table to join and then, identify the common variable of the two tables to perform the join operation. This column as to be introduced by the name of the table followed by a dot.

For example, combine all the variables from the table Orders with the table Customer on the variable key CustomerID:

```sql
SELECT *
FROM Orders
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```

Question 5: which values from the table ***Customers*** *are not included in the results of INNER JOIN?*

By joining all the variables, the results can be very wide. This time select only the relevant variables customerID, companyName, shipCountry and country to see if the customer are shipping their products to the country where they are coming from:

```sql
SELECT companyName, shipCountry, country
FROM Orders
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```

This works well when selecting variable names which are unique (i.e. present in one or the other table but not in both). Try to include the variable CustomerID as well:

```sql
SELECT CustomerID, companyName, shipCountry, country
FROM Orders
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```

An error is produced because CustomerID is present in the table Orders and in the table Customers. To solve this issue, it is possible to explicitly specify the provenance of the variable selected by indicating the table name before the variable name:

```
SELECT Orders.CustomerID, Customers.companyName, Orders.shipCountry, Customers.co
FROM Orders
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```

In order to simplify the query and make it shorter, it is possible to use alias to table name with unique letters. To do so, just add a letter after the name of each table and select the variables with this letter:

```
SELECT a.CustomerID, b.companyName, a.shipCountry, b.country
FROM Orders a
INNER JOIN Customers b ON a.CustomerID = b.CustomerID;
```

The result obtained is very long, some companies could be repeated multiple time, try to order the results by `companyName` to verify:

```sql
SELECT companyName, shipCountry, country
FROM Orders
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID
ORDER BY companyName;
```

In order to avoid repeated values you can use the function `DISTINCT()` on the `companyName` variable which will select only unique names:

```sql
SELECT DISTINCT(companyName), shipCountry, country
FROM Orders
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID
ORDER BY companyName;
```

6. Exercises

Now, formulate the queries by yourself to answer the following questions. Don't hesitate to Google your SQL question to find help.

1. Retrieve all columns in the Region table.
2. Select the FirstName and LastName columns from the Employees table.
3. Select the FirstName and LastName columns from the Employees table. Sort by LastName.
4. Create a report showing Northwind's orders sorted by Freight from most expensive to cheapest. Show OrderID, OrderDate, ShippedDate, CustomerID, and Freight.
5. Create a report showing the title and the first and last name of all sales representatives.
6. Create a report showing the first and last names of all employees who have a region corresponding to 'WA'.
7. Create a report showing the first and last name of all employees whose last names start with a letter in the last half of the alphabet. Sort by LastName in descending order.
8. Create a report showing the first and last name of all sales representatives who are from Seattle or Redmond.
9. Create a report that shows the company name, contact title, city and country of all customers in Mexico or in any city in Spain except Madrid.
10. If the cost of freight is greater than or equal to 500.00, it will now be taxed by 10%. Create a report that shows the order id, freight cost, freight cost with this tax for all orders of 500 or more.
11. Find the Total Number of Units Ordered of Product ID 3
12. Retrieve the number of employees in each city
13. Create a report showing employee orders.
14. Create a report showing the Order ID, the name of the company that placed the order, and the first and last name of the associated employee. Only show orders placed after January 1, 1998 that shipped after they were required. Sort by Company Name.

15. Create a report that shows the total quantity of products (from the Order_Details table) ordered. Only show records for products.

7. More Exercises

For next week, answer the following questions. They all include a specific statement that was not previously introduced. With the help of Google find which statement has to be included and how to formulate the query.

16. Create a report showing the first and last names of all employees who have a region specified.
17. Create a report showing the title of courtesy and the first and last name of all employees whose title of courtesy begins with “M”.
18. Find the number of sales representatives in each city that contains at least 2 sales representatives. Order by the number of employees.
19. Find the Companies (the CompanyName) that placed orders in 1997
20. Create a report that shows the total quantity of products (from the Order_Details table) ordered. Only show records for products for which the quantity ordered is fewer than 200. The report should return the following 5 rows.