

DR. ALVIN'S PUBLICATIONS

# LEARNING DB BROWSER

---

DR. ALVIN ANG



---

1 | PAGE

COPYRIGHTED BY DR ALVIN ANG  
WWW.ALVINANG.SG

---

CONTENTS

---

<b>Contents.....</b>	<b>2</b>
<b>I. Using DB Browser .....</b>	<b>4</b>
<b>A. Installation .....</b>	<b>4</b>
<b>B. Preparation .....</b>	<b>5</b>
<b>C. SELECT .....</b>	<b>6</b>
1. Code Used .....	6
2. Original 'Orders' Table .....	6
3. Explanation.....	7
<b>D. WHERE .....</b>	<b>8</b>
1. Orders with freight charge more than \$10.....	8
a) Code Used .....	8
2. Orders on and after the 1st Jan 2013.....	9
a) Code Used .....	9
<b>E. AND.....</b>	<b>10</b>
1. Orders with freight charge more than \$10 AND on and after the 1st Jan 2013 .....	10
a) Code Used .....	10
<b>F. OR .....</b>	<b>11</b>
1. Order Details table.....	11
a) Code Used .....	11
2. Order details with unit price more than 10 OR quantity more than 50 .....	12
a) Code Used .....	12
<b>G. ORDER BY .....</b>	<b>13</b>
1. Order by freight charge .....	13
a) Code Used .....	13
2. Order by freight charge with highest cost first .....	14
a) Code Used .....	14
<b>H. TOP/LIMIT .....</b>	<b>15</b>
1. Select the top 5 highest freight charge orders .....	15
a) Code Used .....	15
<b>I. AVG.....</b>	<b>16</b>
1. Average freight charge .....	16
a) Code Used .....	16
<b>J. SUM .....</b>	<b>17</b>
1. Total freight charge.....	17
a) Code Used .....	17

<b>K. AS (Rename)</b> .....	<b>18</b>
a) Code Used .....	18
<b>L. JOINS</b> .....	<b>19</b>
1. LEFT JOIN .....	20
a) Code Used .....	20
b) Original Orders Table (Table A) .....	21
c) Original Customers Table (Table B).....	21
2. INNER JOIN .....	23
a) Code Used .....	23
3. FULL JOIN.....	25
a) Code Used .....	25
4. CROSS JOIN .....	26
a) Lists all Employees .....	27
b) Lists all Customers .....	28
c) Lists all combinations of Employees and Customers .....	29
<b>M. ARITHMETIC OPERATORS</b> .....	<b>30</b>
1. Total price of each item in the “OrderDetails” table.....	30
a) Code Used .....	30
b) Original Table .....	30
c) Output.....	31
<b>N. GROUP BY + ORDER BY</b> .....	<b>32</b>
1. Count the number of orders each customer has made .....	32
a) Code Used .....	32
b) Output.....	34
<b>About Dr Alvin Ang</b> .....	<b>35</b>

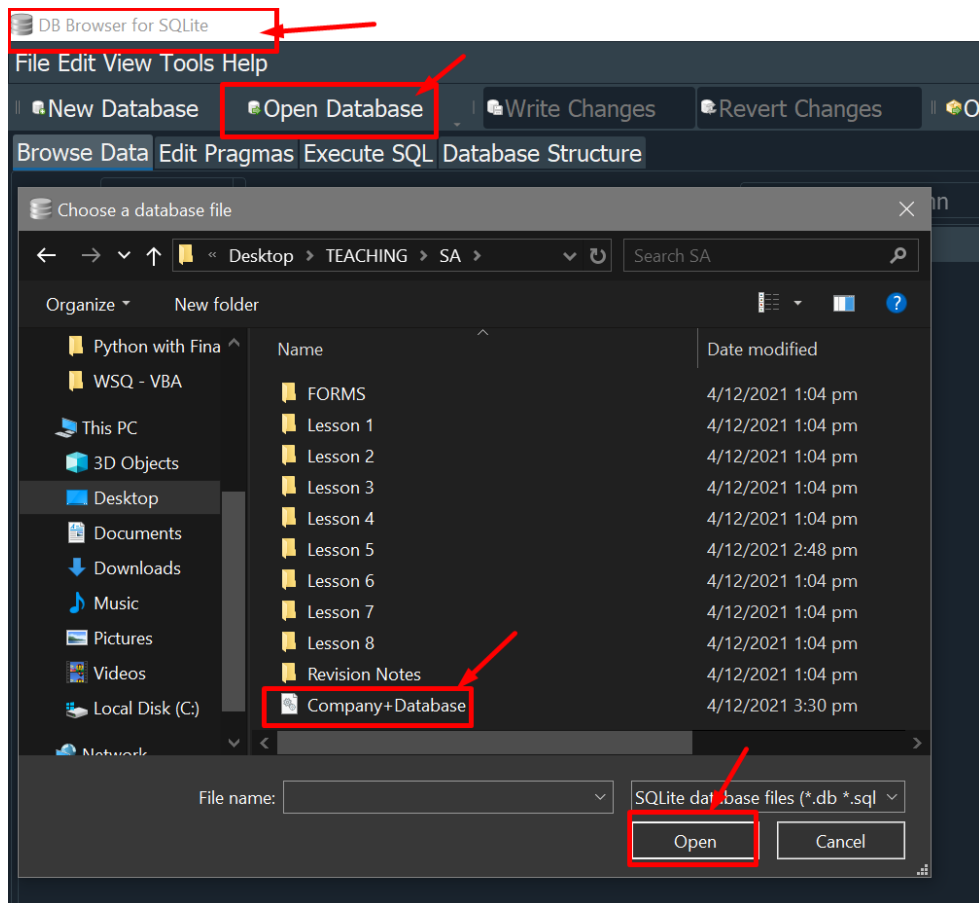
---

## I. USING DB BROWSER

---

### A. INSTALLATION

- <https://sqlitebrowser.org/>
- <https://www.alvinang.sg/s/Company-Database.db>
- Use the Sqlite browser software to open the .db file



## B. PREPARATION

There are lots of extra windows at the side panel which we don't need  
Close all of them

CustomerID	CompanyName	ContactName	ContactTitle	Address	City	State
1	Deerfield Tile	Dick Terrcotta	Owner	450 Village Street	Deerfield	IL
2	Sagebrush Carpet	Barbara Berber	Director of Installations	10 Industrial Drive	El Paso	TX
3	Floor Co.	Jim Wood	Installer	34218 Private Lane	Monclair	NJ
4	Main Tile and Bath	Toni Faucet	Owner	Suite 23, Henry Building	Orlando	FL
5	Slots Carpet	Jack Diamond III	Purchaser	3024 Jackpot Drive	Las Vegas	NV

you may click on "browse data" to glance at all the different tables stored in the "Company Database.db"

Name	Type	Schema
Customers	Table	CREATE TABLE Customers ( C
Employees	Table	CREATE TABLE Employees ( E
OrderDetails	Table	CREATE TABLE OrderDetails (
Orders	Table	CREATE TABLE Orders ( Order
Shippers	Table	CREATE TABLE Shippers ( Shij
Indices (0)	Index	
Views (0)	View	
Triggers (0)	Trigger	

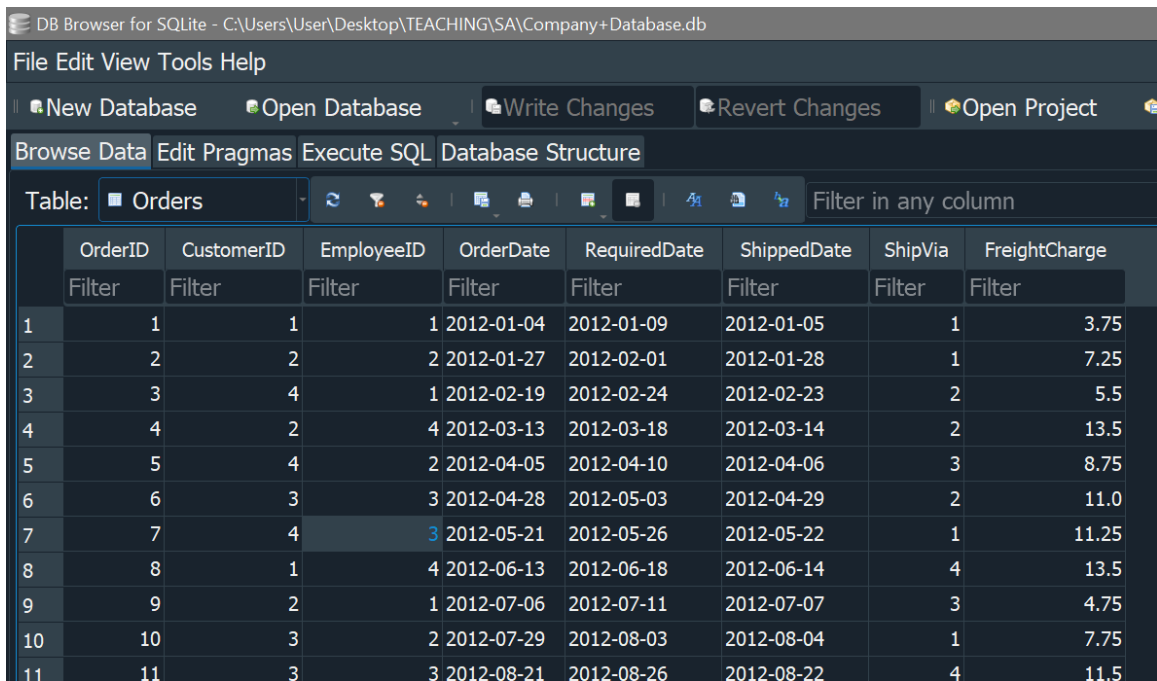
## C. SELECT

### 1. CODE USED

```
SELECT *
```

```
FROM Orders
```

### 2. ORIGINAL 'ORDERS' TABLE



The screenshot shows the DB Browser for SQLite interface. The title bar indicates the database path: C:\Users\User\Desktop\TEACHING\SA\Company+Database.db. The menu bar includes File, Edit, View, Tools, and Help. The toolbar contains buttons for New Database, Open Database, Write Changes, Revert Changes, and Open Project. The main area shows the 'Orders' table selected, with a search filter 'Filter in any column'. The table data is as follows:

	OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate	ShippedDate	ShipVia	FreightCharge
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	1	1	1 2012-01-04	2012-01-09	2012-01-05	1	3.75
2	2	2	2	2 2012-01-27	2012-02-01	2012-01-28	1	7.25
3	3	4	4	1 2012-02-19	2012-02-24	2012-02-23	2	5.5
4	4	2	2	4 2012-03-13	2012-03-18	2012-03-14	2	13.5
5	5	4	4	2 2012-04-05	2012-04-10	2012-04-06	3	8.75
6	6	3	3	3 2012-04-28	2012-05-03	2012-04-29	2	11.0
7	7	4	4	3 2012-05-21	2012-05-26	2012-05-22	1	11.25
8	8	1	1	4 2012-06-13	2012-06-18	2012-06-14	4	13.5
9	9	2	2	1 2012-07-06	2012-07-11	2012-07-07	3	4.75
10	10	3	3	2 2012-07-29	2012-08-03	2012-08-04	1	7.75
11	11	3	3	3 2012-08-21	2012-08-26	2012-08-22	4	11.5

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project

Browse Data Edit Pragma Execute SQL Database Structure

SQL 1

```
1 SELECT *
2 FROM Orders
3
```

	OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate	ShippedDate	ShipVia	FreightCharge
1	1	1	1	2012-01-04	2012-01-09	2012-01-05	1	3.75
2	2	2	2	2012-01-27	2012-02-01	2012-01-28	1	7.25
3	3	4	4	2012-02-19	2012-02-24	2012-02-23	2	5.5
4	4	2	4	2012-03-13	2012-03-18	2012-03-14	2	13.5
5	5	4	2	2012-04-05	2012-04-10	2012-04-06	3	8.75
6	6	3	3	2012-04-28	2012-05-03	2012-04-29	2	11.0

Execution finished without errors.  
Result: 20 rows returned in 22ms  
At line 1:  
SELECT \*  
FROM Orders

The entire original 'Orders' table is displayed

### 3. EXPLANATION

- Select \* means Select all columns
- From the Orders Table
- All the columns in Orders Table is taken out for display only.
- The original database is left untouched – nothing has been changed within the tables.

## D. WHERE

### 1. ORDERS WITH FREIGHT CHARGE MORE THAN \$10

#### a) Code Used

SELECT \*

FROM Orders

WHERE FreightCharge>10

DB browser for SQLite - C:\Users\user\Desktop\EXERCISES\Company Database.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project

Browse Data Edit Pragma **Execute SQL** Database Structure

Click here

SQL 1

```
1 SELECT *
2 FROM Orders
3 WHERE FreightCharge>10
4
```

copy paste the code here and run it

all freight charges > 10 are displayed

	OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate	ShippedDate	ShipVia	FreightCharge
1	4	2	4	2012-03-13	2012-03-18	2012-03-14	2	13.5
2	6	3	3	2012-04-28	2012-05-03	2012-04-29	2	11.0
3	7	4	3	2012-05-21	2012-05-26	2012-05-22	1	11.25
4	8	1	4	2012-06-13	2012-06-18	2012-06-14	4	13.5
5	11	3	3	2012-08-21	2012-08-26	2012-08-22	4	11.5
6	12	1	4	2012-09-13	2012-09-18	2012-09-14	2	13.0

Execution finished without errors.  
Result: 10 rows returned in 10ms  
At line 1:  
SELECT \*  
FROM Orders  
WHERE FreightCharge>10



2. ORDERS ON AND AFTER THE 1ST JAN 2013

a) Code Used

```
SELECT *
```

```
FROM Orders
```

```
WHERE OrderDate>='2013-01-01'
```

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project

Browse Data Edit Pragmas Execute SQL Database Structure

SQL 1

```
1 SELECT *
2 FROM Orders
3 WHERE OrderDate>='2013-01-01'
4
```

	OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate	ShippedDate	ShipVia	FreightCharge
1	17	5	1	2013-01-06	2013-01-11	2013-01-07	3	6.25
2	18	3	3	2013-01-29	2013-02-03	2013-01-30	1	10.75
3	19	2	4	2013-02-21	2013-02-26	2013-03-01	4	14.0
4	20	3	1	2013-03-16	2013-03-21	2013-03-17	4	5.5

all order dates >= 2013 are displayed

Execution finished without errors.  
Result: 4 rows returned in 10ms  
At line 1:  
SELECT \*  
FROM Orders  
WHERE OrderDate>='2013-01-01'

## E. AND

1. ORDERS WITH FREIGHT CHARGE MORE THAN \$10 AND ON AND AFTER THE 1ST JAN 2013

a) *Code Used*

```
SELECT *
```

```
FROM Orders
```

```
WHERE FreightCharge>10 AND OrderDate>='2013-01-01'
```

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project

Browse Data Edit Pragmas Execute SQL Database Structure

SQL 1

```
1 SELECT *
2 FROM Orders
3 WHERE FreightCharge>10 AND OrderDate>='2013-01-01'
4
```

	OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate	ShippedDate	ShipVia	FreightCharge
1	18	3	3	2013-01-29	2013-02-03	2013-01-30	1	10.75
2	19	2	4	2013-02-21	2013-02-26	2013-03-01	4	14.0

Execution finished without errors.  
Result: 2 rows returned in 10ms  
At line 1:  
SELECT \*  
FROM Orders  
WHERE FreightCharge>10 AND OrderDate>='2013-01-01'

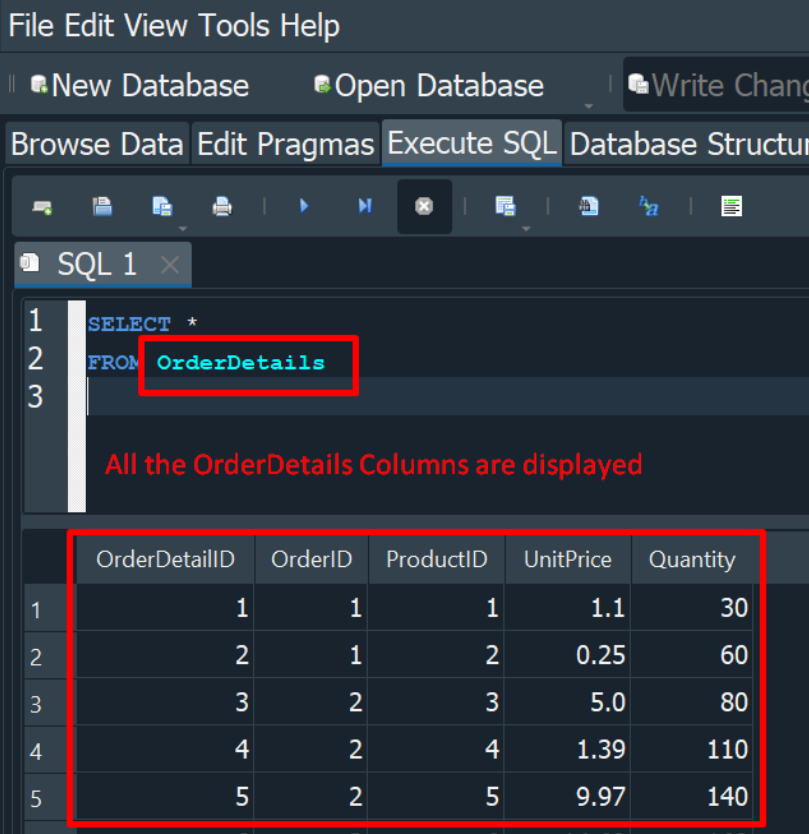
F. OR

1. ORDER DETAILS TABLE

a) Code Used

SELECT \*

FROM OrderDetails



The screenshot shows the SQL Server Enterprise Manager interface. The 'Execute SQL' tab is active, displaying a query window with the following SQL code:

```
1 SELECT *
2 FROM OrderDetails
3
```

The text 'OrderDetails' in the FROM clause is highlighted with a red box. Below the query, a red text annotation reads: 'All the OrderDetails Columns are displayed'. The results grid below shows the following data:

	OrderDetailID	OrderID	ProductID	UnitPrice	Quantity
1	1	1	1	1.1	30
2	2	1	2	0.25	60
3	3	2	3	5.0	80
4	4	2	4	1.39	110
5	5	2	5	9.97	140

2. ORDER DETAILS WITH UNIT PRICE MORE THAN 10 OR QUANTITY MORE THAN 50

a) Code Used

SELECT \*

FROM OrderDetails

WHERE UnitPrice>10 OR Quantity>50

File Edit View Tools Help

New Database Open Database Write Changes

Browse Data Edit Pragmas Execute SQL Database Structure

SQL 1

```
1 SELECT *
2 FROM OrderDetails
3 WHERE UnitPrice>10 OR Quantity>50
4
```

	OrderDetailID	OrderID	ProductID	UnitPrice	Quantity	
1		2	1	2	0.25	60
2		3	2	3	5.0	80
3		4	2	4	1.39	110
4		5	2	5	9.97	140
5		6	3	6	14.69	160
6		9	4	3	5.0	80

Execution finished without errors.  
Result: 24 rows returned in 9ms  
At line 1:  
SELECT \*  
FROM OrderDetails  
WHERE UnitPrice>10 OR Quantity>50

- Notice that because its OR, even if UnitPrice < 10 will be displayed as long as Qty > 50

## G. ORDER BY

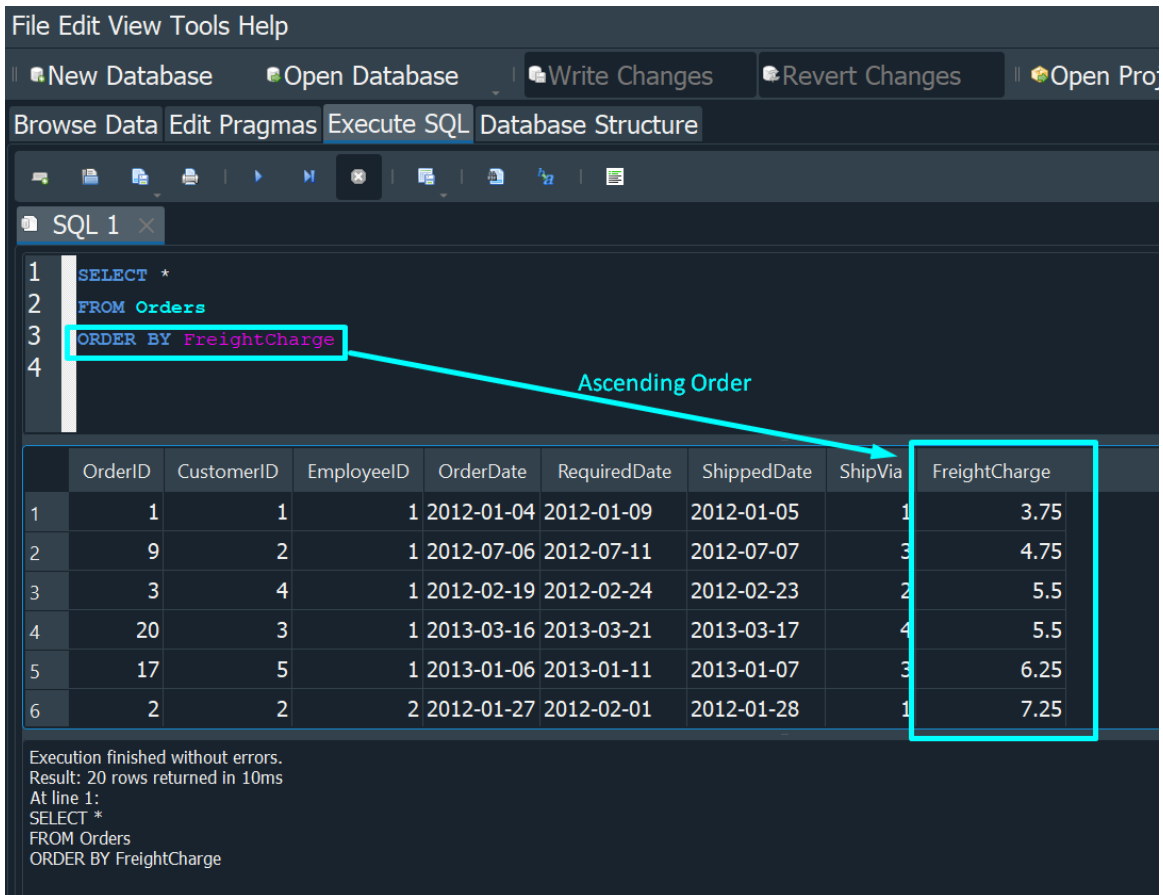
### 1. ORDER BY FREIGHT CHARGE

#### a) Code Used

```
SELECT *
```

```
FROM Orders
```

```
ORDER BY FreightCharge
```



File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Proj

Browse Data Edit Pragmas Execute SQL Database Structure

SQL 1

```
1 SELECT *
2 FROM Orders
3 ORDER BY FreightCharge
4
```

Ascending Order

	OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate	ShippedDate	ShipVia	FreightCharge
1	1	1	1	2012-01-04	2012-01-09	2012-01-05	1	3.75
2	9	2	2	2012-07-06	2012-07-11	2012-07-07	3	4.75
3	3	4	4	2012-02-19	2012-02-24	2012-02-23	2	5.5
4	20	3	3	2013-03-16	2013-03-21	2013-03-17	4	5.5
5	17	5	5	2013-01-06	2013-01-11	2013-01-07	3	6.25
6	2	2	2	2012-01-27	2012-02-01	2012-01-28	1	7.25

Execution finished without errors.  
Result: 20 rows returned in 10ms  
At line 1:  
SELECT \*  
FROM Orders  
ORDER BY FreightCharge

2. ORDER BY FREIGHT CHARGE WITH HIGHEST COST FIRST

a) Code Used

```
SELECT *
```

```
FROM Orders
```

```
ORDER BY FreightCharge DESC
```

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Pr

Browse Data Edit Pragmas Execute SQL Database Structure

SQL 1

```
1 SELECT *
2 FROM Orders
3 ORDER BY FreightCharge DESC
4
```

Descending Order

	OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate	ShippedDate	ShipVia	FreightCharge
1	16	3	4	2012-12-14	2012-12-19	2012-12-15	2	14.0
2	19	2	4	2013-02-21	2013-02-26	2013-03-01	4	14.0
3	4	2	4	2012-03-13	2012-03-18	2012-03-14	2	13.5
4	8	1	4	2012-06-13	2012-06-18	2012-06-14	4	13.5
5	12	1	4	2012-09-13	2012-09-18	2012-09-14	2	13.0
6	13	5	3	2012-10-06	2012-10-11	2012-10-07	3	12.25

Execution finished without errors.  
Result: 20 rows returned in 10ms  
At line 1:  
SELECT \*  
FROM Orders  
ORDER BY FreightCharge DESC

## H. TOP/LIMIT

### 1. SELECT THE TOP 5 HIGHEST FREIGHT CHARGE ORDERS

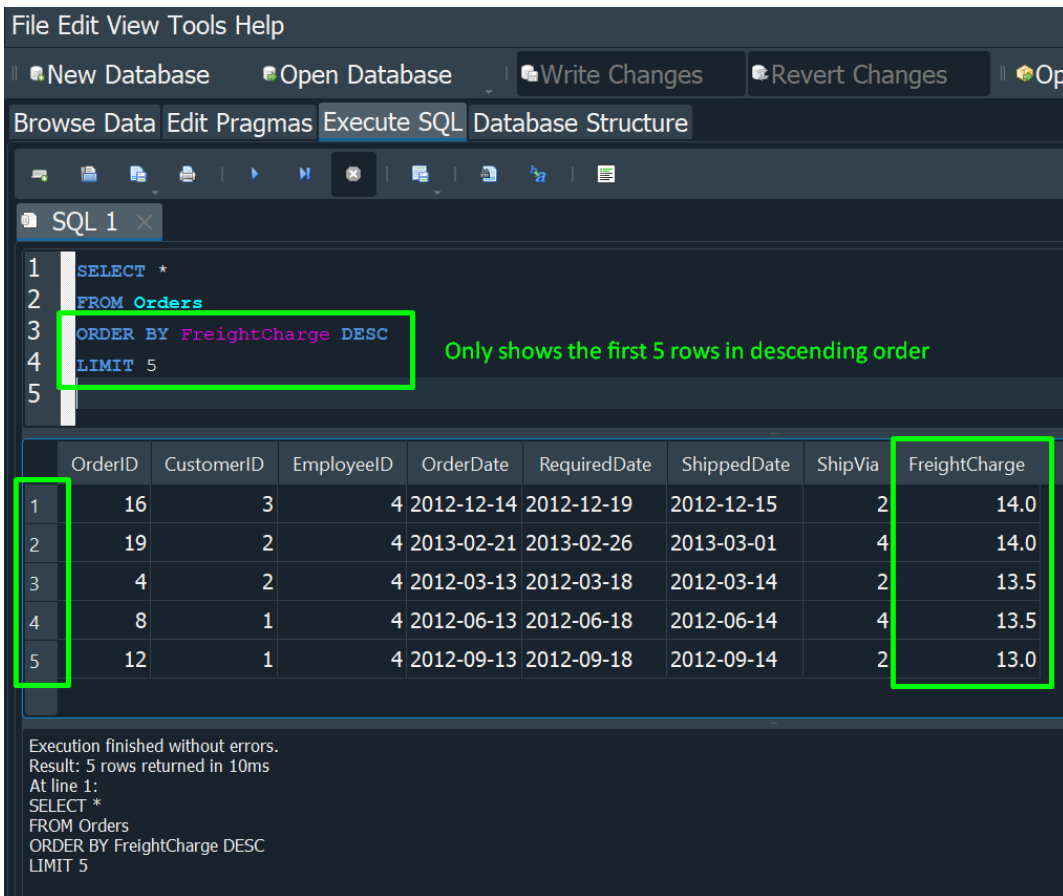
#### a) Code Used

SELECT \*

FROM Orders

ORDER BY FreightCharge DESC

LIMIT 5



The screenshot shows the SQL Developer interface. The SQL editor contains the following query:

```
1 SELECT *
2 FROM Orders
3 ORDER BY FreightCharge DESC
4 LIMIT 5
5
```

A green box highlights the query text. A green annotation reads: "Only shows the first 5 rows in descending order".

The results pane displays a table with the following data:

	OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate	ShippedDate	ShipVia	FreightCharge
1	16	3	4	2012-12-14	2012-12-19	2012-12-15	2	14.0
2	19	2	4	2013-02-21	2013-02-26	2013-03-01	4	14.0
3	4	2	4	2012-03-13	2012-03-18	2012-03-14	2	13.5
4	8	1	4	2012-06-13	2012-06-18	2012-06-14	4	13.5
5	12	1	4	2012-09-13	2012-09-18	2012-09-14	2	13.0

The status bar at the bottom indicates: "Execution finished without errors. Result: 5 rows returned in 10ms. At line 1: SELECT \* FROM Orders ORDER BY FreightCharge DESC LIMIT 5".

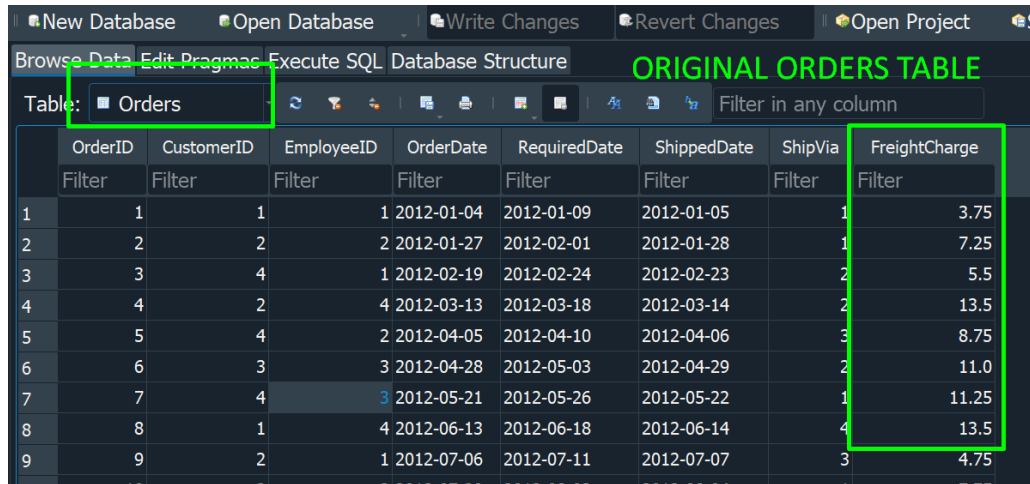
## I. AVG

### 1. AVERAGE FREIGHT CHARGE

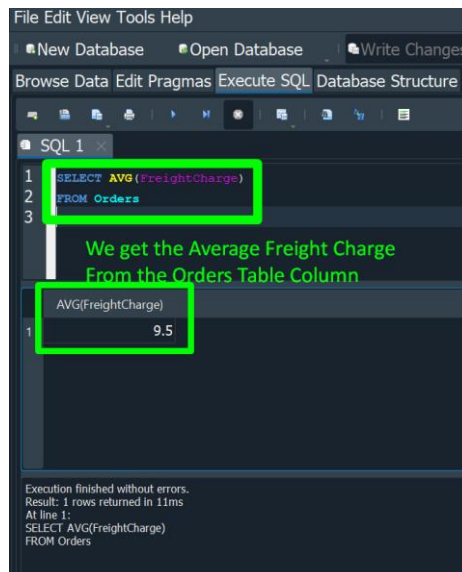
#### a) Code Used

```
SELECT AVG(FreightCharge)
```

```
FROM Orders
```



	OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate	ShippedDate	ShipVia	FreightCharge
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	1	1	2012-01-04	2012-01-09	2012-01-05	1	3.75
2	2	2	2	2012-01-27	2012-02-01	2012-01-28	1	7.25
3	3	4	4	2012-02-19	2012-02-24	2012-02-23	2	5.5
4	4	2	2	2012-03-13	2012-03-18	2012-03-14	2	13.5
5	5	4	4	2012-04-05	2012-04-10	2012-04-06	3	8.75
6	6	3	3	2012-04-28	2012-05-03	2012-04-29	2	11.0
7	7	4	3	2012-05-21	2012-05-26	2012-05-22	1	11.25
8	8	1	4	2012-06-13	2012-06-18	2012-06-14	4	13.5
9	9	2	1	2012-07-06	2012-07-11	2012-07-07	3	4.75
10	10	3	3	2012-07-20	2012-08-03	2012-08-04	1	7.75



```
File Edit View Tools Help
New Database Open Database Write Changes
Browse Data Edit Pragmas Execute SQL Database Structure
SQL 1
1 SELECT AVG(FreightCharge)
2 FROM Orders
3
We get the Average Freight Charge
From the Orders Table Column
AVG(FreightCharge)
1 9.5
Execution finished without errors.
Result: 1 rows returned in 11ms
At line 1:
SELECT AVG(FreightCharge)
FROM Orders
```



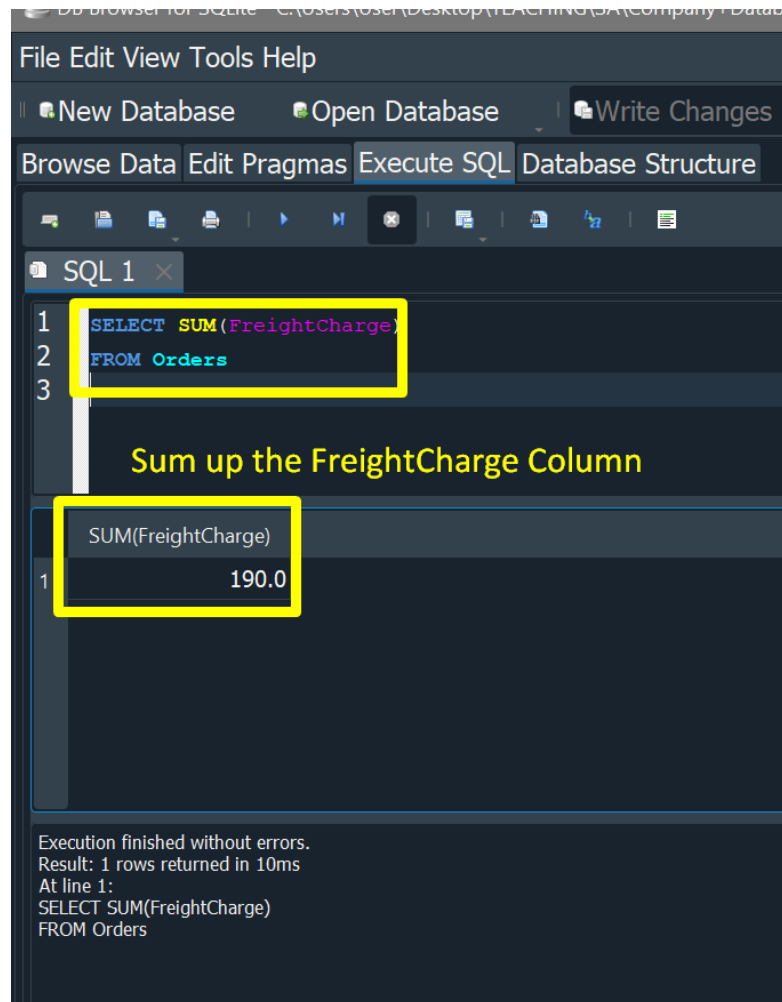
## J. SUM

### 1. TOTAL FREIGHT CHARGE

#### a) Code Used

```
SELECT SUM(FreightCharge)
```

```
FROM Orders
```



## K. AS (RENAME)

### a) Code Used

```
SELECT AVG(FreightCharge)
```

```
AS AverageFreightCharge,
```

```
SUM(FreightCharge)
```

```
AS TotalFreightCharge
```

```
FROM Orders
```

The screenshot shows a SQL IDE interface with a query editor and a results pane. The query is as follows:

```
1 SELECT AVG(FreightCharge)
2
3 AS AverageFreightCharge,
4
5 SUM(FreightCharge)
6
7 AS TotalFreightCharge
8
9 FROM Orders
```

Annotations in the image explain the query parts:

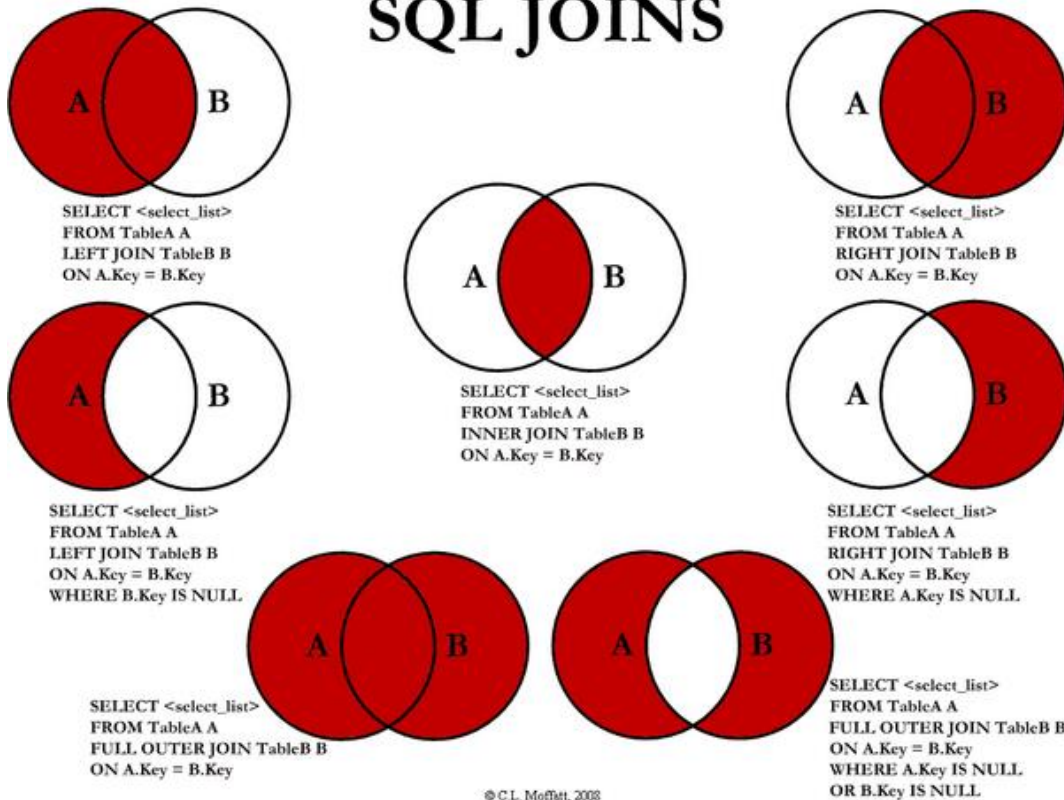
- Line 1: `AVG(FreightCharge)` - we get the average freight charge then
- Line 3: `AS AverageFreightCharge,` - we rename it to AverageFreightCharge
- Line 5: `SUM(FreightCharge)` - we get the sum of all the freight charges then
- Line 7: `AS TotalFreightCharge` - we rename it to TotalFreightCharge

The results pane shows the following data:

AverageFreightCharge	TotalFreightCharge
9.5	190.0

## L. JOINS

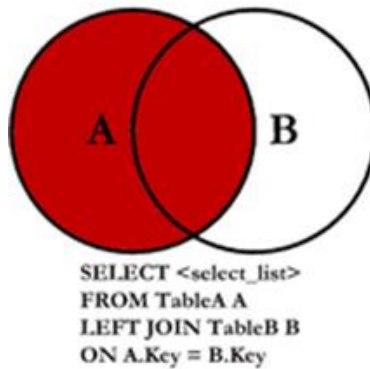
# SQL JOINS



Source: <https://stackoverflow.com/questions/5706437/whats-the-difference-between-inner-join-left-join-right-join-and-full-join>

- INNER JOIN: returns rows when there is a match in both tables.
- LEFT JOIN: returns all rows from the left table, even if there are no matches in the right table.
- RIGHT JOIN: returns all rows from the right table, even if there are no matches in the left table.
- FULL JOIN: combines the results of both left and right outer joins. The joined table will contain all records from both the tables and fill in NULLs for missing matches on either side.
- SELF JOIN: joins a table to itself as if the table were two tables, temporarily renaming at least one table in the SQL statement.

## 1. LEFT JOIN



### a) Code Used

```
SELECT Orders.OrderID, Customers.ContactName
```

```
FROM Orders
```

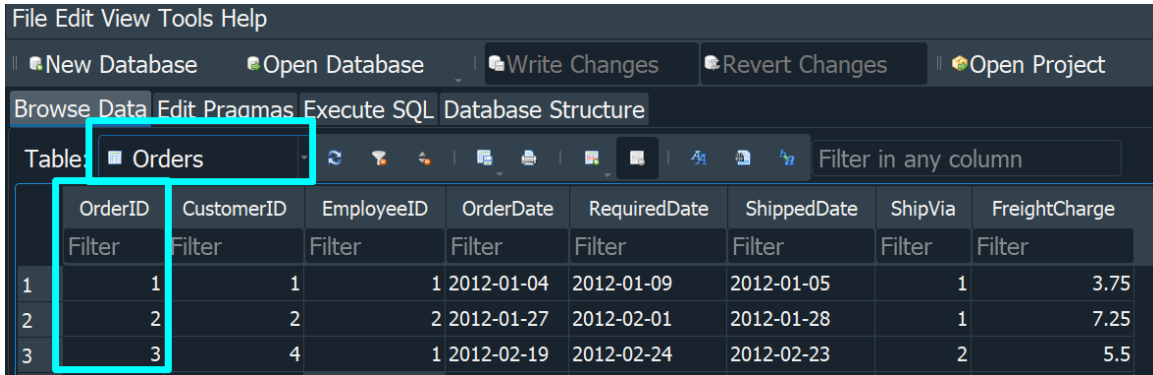
```
LEFT JOIN Customers
```

```
ON Orders.CustomerID=Customers.CustomerID
```

- Table A: Orders Table
  - We want the *OrderID column* from the 'Orders Table'
- Table B: Customers Table
  - We want the *ContactName column* from the 'Customers Table'
- Left Join:
  - Entire *OrderID column* from Table A (the 'Orders Table') will be displayed
- Key: CustomerID
  - Both Table A and Table B share the same Key (CustomerID column) → we need to match this column to join both tables.

b) Original Orders Table (Table A)

- We select the Orders → OrderID column.

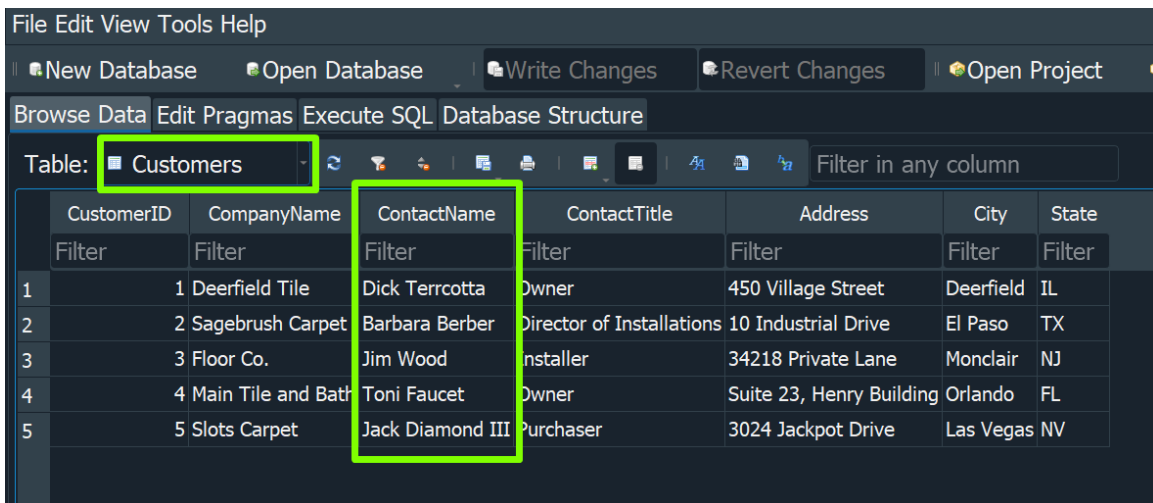


The screenshot shows the SQL Server Enterprise Manager interface. The 'Table: Orders' dropdown is highlighted with a red box. The table data is as follows:

	OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate	ShippedDate	ShipVia	FreightCharge
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	1	1	2012-01-04	2012-01-09	2012-01-05	1	3.75
2	2	2	2	2012-01-27	2012-02-01	2012-01-28	1	7.25
3	3	4	1	2012-02-19	2012-02-24	2012-02-23	2	5.5

c) Original Customers Table (Table B)

- We select the Customers → Contact Name column



The screenshot shows the SQL Server Enterprise Manager interface. The 'Table: Customers' dropdown is highlighted with a red box. The table data is as follows:

	CustomerID	CompanyName	ContactName	ContactTitle	Address	City	State
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	Deerfield Tile	Dick Terrcotta	Owner	450 Village Street	Deerfield	IL
2	2	Sagebrush Carpet	Barbara Berber	Director of Installations	10 Industrial Drive	El Paso	TX
3	3	Floor Co.	Jim Wood	Installer	34218 Private Lane	Monclair	NJ
4	4	Main Tile and Bath	Toni Faucet	Owner	Suite 23, Henry Building	Orlando	FL
5	5	Slots Carpet	Jack Diamond III	Purchaser	3024 Jackpot Drive	Las Vegas	NV

**Table A**

OrderID	CustomerID	EmployeeID
1	1	1
2	2	2
3	3	4

**Table B**

CustomerID	CompanyName	ContactName
1	Deerfield Tile	Dick Terrcotta
2	Sagebrush Carpet	Barbara Berber
3	Floor Co.	Jim Wood
4	Main Tile and Bath	Toni Faucet
5	Slots Carpet	Jack Diamond III

Match Based on This column

**Output**

OrderID	ContactName
1	Dick Terrcotta
2	Barbara Berber
3	Toni Faucet
4	Barbara Berber
5	Toni Faucet
6	Jim Wood

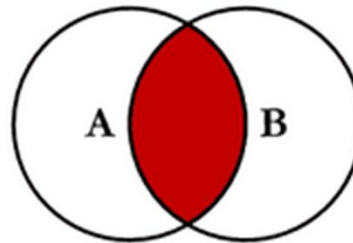
- Note that the CustomerID column will not be displayed.
- Note that the ENTIRE OrderID column will be displayed even though there might not be any ContactNames linked to it.
- For example....

OrderID	ContactName
19	Barbara Berber
20	Jim Wood
21	NaN
22	NaN

Will Display NaN if there are no matches to OrderID

But OrderID is present

## 2. INNER JOIN



```
SELECT <select_list>  
FROM TableA A  
INNER JOIN TableB B  
ON A.Key = B.Key
```

### a) Code Used

```
SELECT Orders.OrderID , Customers.ContactName
```

```
FROM Orders
```

```
INNER JOIN Customers
```

```
ON Orders.CustomerID =Customers.CustomerID
```

- Table A: Orders Table
  - We want the *OrderID column* from the 'Orders Table'
- Table B: Customers Table
  - We want the *ContactName column* from the 'Customers Table'
- Inner Join:
  - Only Matching Values from CustomerID column will be displayed
- Key: CustomerID
  - Both Table A and Table B share the same Key (CustomerID column) → we need to match this column to join both tables.

**Table A**

OrderID	CustomerID	EmployeeID
1	1	1
2	2	2
3	3	4

**Table B**

CustomerID	CompanyName	ContactName
1	Deerfield Tile	Dick Terrcotta
2	Sagebrush Carpet	Barbara Berber
3	Floor Co.	Jim Wood
4	Main Tile and Bath	Toni Faucet
5	Slots Carpet	Jack Diamond III

Match Based on This column

**Output**

OrderID	ContactName
1	Dick Terrcotta
2	Barbara Berber
3	Toni Faucet
4	Barbara Berber
5	Toni Faucet
6	Jim Wood

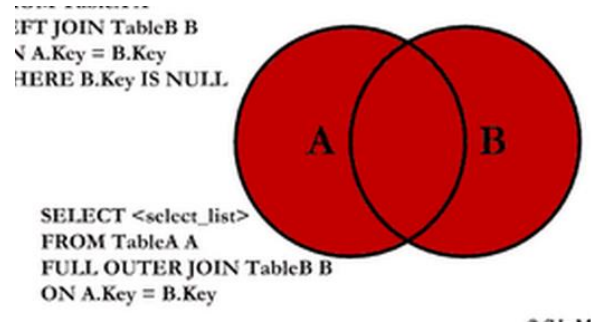
- Note that the CustomerID column will not be displayed.
- Note that only MATCHING values of the CustomerID column will be displayed.
- For example....

OrderID	ContactName
19	Barbara Berber
20	Jim Wood
21	NaN
22	NaN

It will stop at row 20  
Because NaN values show no match, and won't be displayed

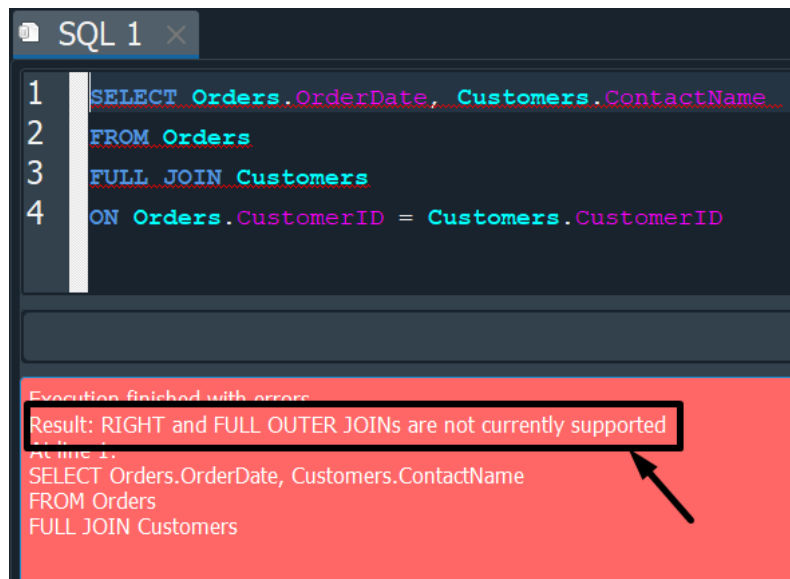


### 3. FULL JOIN



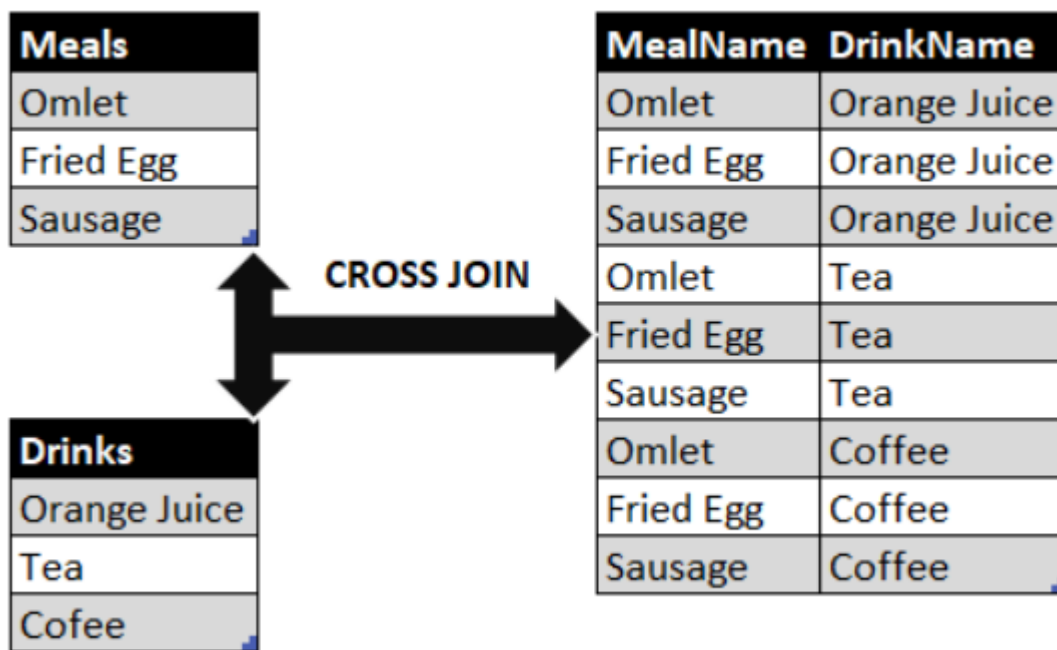
#### a) Code Used

```
SELECT Orders.OrderDate, Customers.ContactName  
FROM Orders  
FULL JOIN Customers  
ON Orders.CustomerID = Customers.CustomerID
```



Note that currently, DB for SQLite does not support RIGHT and FULL JOINS.

#### 4. CROSS JOIN



This is how Cross Join Works

- It creates a combination of every kind from both tables
- You see that Orange Juice is combined with every Meal
- You see that Tea is combined with every Meal.
- And so forth...

a) Lists all Employees

```
SELECT FirstName AS EmployeeName
```

```
FROM Employees
```

	EmployeeID	LastName	FirstName	Title	Address	HireDate
	Filter	Filter	Filter	Filter	Filter	Filter
1	1	White	James	Account Manager		2011-04-03
2	2	Lee	Patty	Account Manager	NULL	2011-09-15
3	3	Smith	Robert	Account Manager	NULL	2004-06-28
4	4	Baker	Lisa	Account Manager	NULL	2010-11-20

```
SQL 1 x
1 SELECT FirstName AS EmployeeName
2 FROM Employees
3
```

	EmployeeName
1	James
2	Patty
3	Robert
4	Lisa

Execution finished without errors.  
Result: 4 rows returned in 11ms  
At line 1:  
SELECT FirstName AS EmployeeName  
FROM Employees

b) Lists all Customers

```
SELECT ContactName AS CustomerName
```

```
FROM Customers
```

	CustomerID	CompanyName	ContactName	ContactTitle	Address	City	State
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	Deerfield Tile	Dick Terrcotta	Owner	450 Village Street	Deerfield	IL
2	2	Sagebrush Carpet	Barbara Berber	Director of Installations	10 Industrial Drive	El Paso	TX
3	3	Floor Co.	Jim Wood	Installer	34218 Private Lane	Monclair	NJ
4	4	Main Tile and Bath	Toni Faucet	Owner	Suite 23, Henry Building	Orlando	FL
5	5	Slots Carpet	Jack Diamond III	Purchaser	3024 Jackpot Drive	Las Vegas	NV

```
1 SELECT ContactName AS CustomerName
2 FROM Customers
3
```

	CustomerName
1	Dick Terrcotta
2	Barbara Berber
3	Jim Wood
4	Toni Faucet
5	Jack Diamond III

c) Lists all combinations of Employees and Customers

```
SELECT Employees.FirstName AS EmployeeName,
```

```
Customers.ContactName AS CustomerName
```

```
FROM Employees
```

```
CROSS JOIN Customers
```

```
1 SELECT Employees.FirstName AS EmployeeName,
2 Customers.ContactName AS CustomerName
3 FROM Employees
4 CROSS JOIN Customers
```

	EmployeeName	CustomerName	
1	James	Dick Terrcotta	All Combinations Appear Meaning,
2	James	Barbara Berber	
3	James	Jim Wood	
4	James	Toni Faucet	
5	James	Jack Diamond III	James to
6	Patty	Dick Terrcotta	- Dick
7	Patty	Barbara Berber	- Barbara
8	Patty	Jim Wood	- Jim
9	Patty	Toni Faucet	- Toni
10	Patty	Jack Diamond III	- Jack
11	Robert	Dick Terrcotta	Patty to
12	Robert	Barbara Berber	~.....

And this repeats for ALL names

## M. ARITHMETIC OPERATORS

### 1. TOTAL PRICE OF EACH ITEM IN THE "ORDERDETAILS" TABLE

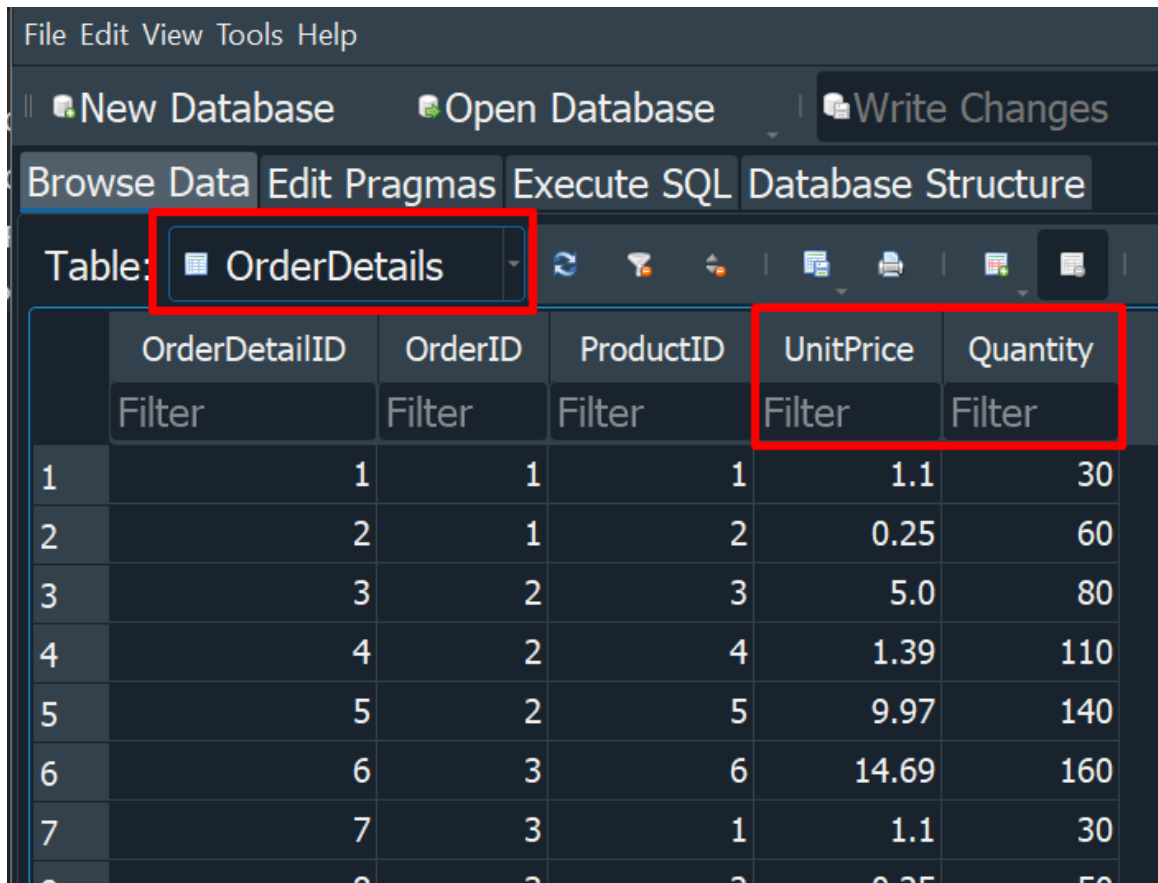
#### a) Code Used

```
SELECT *, UnitPrice*Quantity AS Price
```

```
FROM OrderDetails
```

#### b) Original Table

- This is the original OrderDetails Table.



The screenshot shows a database management tool interface. The 'Table:' dropdown menu is set to 'OrderDetails'. The table data is as follows:

	OrderDetailID	OrderID	ProductID	UnitPrice	Quantity
	Filter	Filter	Filter	Filter	Filter
1	1	1	1	1.1	30
2	2	1	2	0.25	60
3	3	2	3	5.0	80
4	4	2	4	1.39	110
5	5	2	5	9.97	140
6	6	3	6	14.69	160
7	7	3	1	1.1	30
8	8	2	2	0.25	50

c) Output

```
1 SELECT *, UnitPrice*Quantity AS Price
2 FROM OrderDetails
```

	OrderDetailID	OrderID	ProductID	UnitPrice	Quantity	Price	
1	1	1	1	1.1	30	33.0	1.1 x 30 = 33
2	2	1	2	0.25	60	15.0	0.25 x 60 = 15
3	3	2	3	5.0	80	400.0	5 x 80 = 400
4	4	2	4	1.39	110	152.9	....

## N. GROUP BY + ORDER BY

1. COUNT THE NUMBER OF ORDERS EACH CUSTOMER HAS MADE

a) *Code Used*

SELECT

COUNT(Orders.OrderID) AS OrderCount ,

Customers.ContactName AS CustomerName,

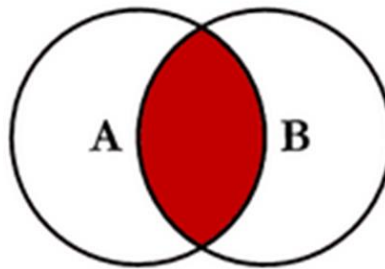
FROM Orders

INNER JOIN Customers ON Customers.CustomerID=Orders.CustomerID

GROUP BY Customers.ContactName

ORDER BY COUNT(Orders.OrderID) DESC





```
SELECT <select_list>  
FROM TableA A  
INNER JOIN TableB B  
ON A.Key = B.Key
```

- Table A: Orders Table
  - We want the *OrderID* column from the 'Orders Table'
- Table B: Customers Table
  - We want the *ContactName* column from the 'Customers Table'
- Inner Join:
  - Only Matching Values from CustomerID column will be displayed
- Key: CustomerID
  - Both Table A and Table B share the same Key (CustomerID column) → we need to match this column to join both tables.
- Groupby: ContactName
  - We group all the same 'ContactNames' together
- Orderby: Count of OrderID
  - Groupby cannot be used by itself, it needs to be followed by an operation.
  - We count the number of times each 'ContactName' appeared (based on tallying its CustomerID) in the 'Orders' Table; and put it in Descending Order.

b) Output

Table A

OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate	ShippedDate	ShipVia	FreightCharge
1	1	1	2012-01-04	2012-01-09	2012-01-05	1	3.75
2	2	2	2012-01-27	2012-02-01	2012-01-28	1	2.29
3	3	4	2012-02-19	2012-02-24	2012-02-23	2	5.5
4	4	2	2012-03-13	2012-03-18	2012-03-14	2	13.5

Table B

CustomerID	CompanyName	ContactName	ContactTitle	Address	City	State
1	Deerfield Tile	Dick Terrcotta	Owner	450 Village Street	Deerfield	IL
2	Sagebrush Carpet	Barbara Berber	Director of Installations	10 Industrial Drive	El Paso	TX
3	Floor Co.	Jim Wood	Installer	34218 Private Lane	Monclair	NJ
4	Main Tile and Bath	Toni Faucet	Owner	Suite 23, Henry Building	Orlando	FL
5	Slots Carpet	Jack Diamond III	Purchaser	3024 Jackpot Drive	Las Vegas	NV

Output

```
8 GROUP BY Customers.ContactName
9 ORDER BY COUNT(Orders.OrderID) DESC
10
```

OrderCount	CustomerName
6	Jim Wood
5	Barbara Berber
4	Toni Faucet
3	Dick Terrcotta
2	Jack Diamond III

Groupby ContactName means grouping all the ContactNames together.  
Groupby cannot be by itself.  
It needs to be followed by an operation.  
In this case, its Ordered by Count  
(No. of times the ContactName appeared)

---

## ABOUT DR ALVIN ANG

---



Dr. Alvin Ang earned his Ph.D., Masters and Bachelor degrees from NTU, Singapore. He was previously a Professor, Scientist and Financial Consultant. Currently, he owns multiple startups and is a Personal/Business Advisor.

More about him at [www.AlvinAng.sg](http://www.AlvinAng.sg)