

# SQL

Please open

[https://www.w3schools.com/sql/trysql.asp?filename=trysql\\_create\\_table](https://www.w3schools.com/sql/trysql.asp?filename=trysql_create_table)

IN CHROME OR SAFARI (Otherwise, some examples won't work)

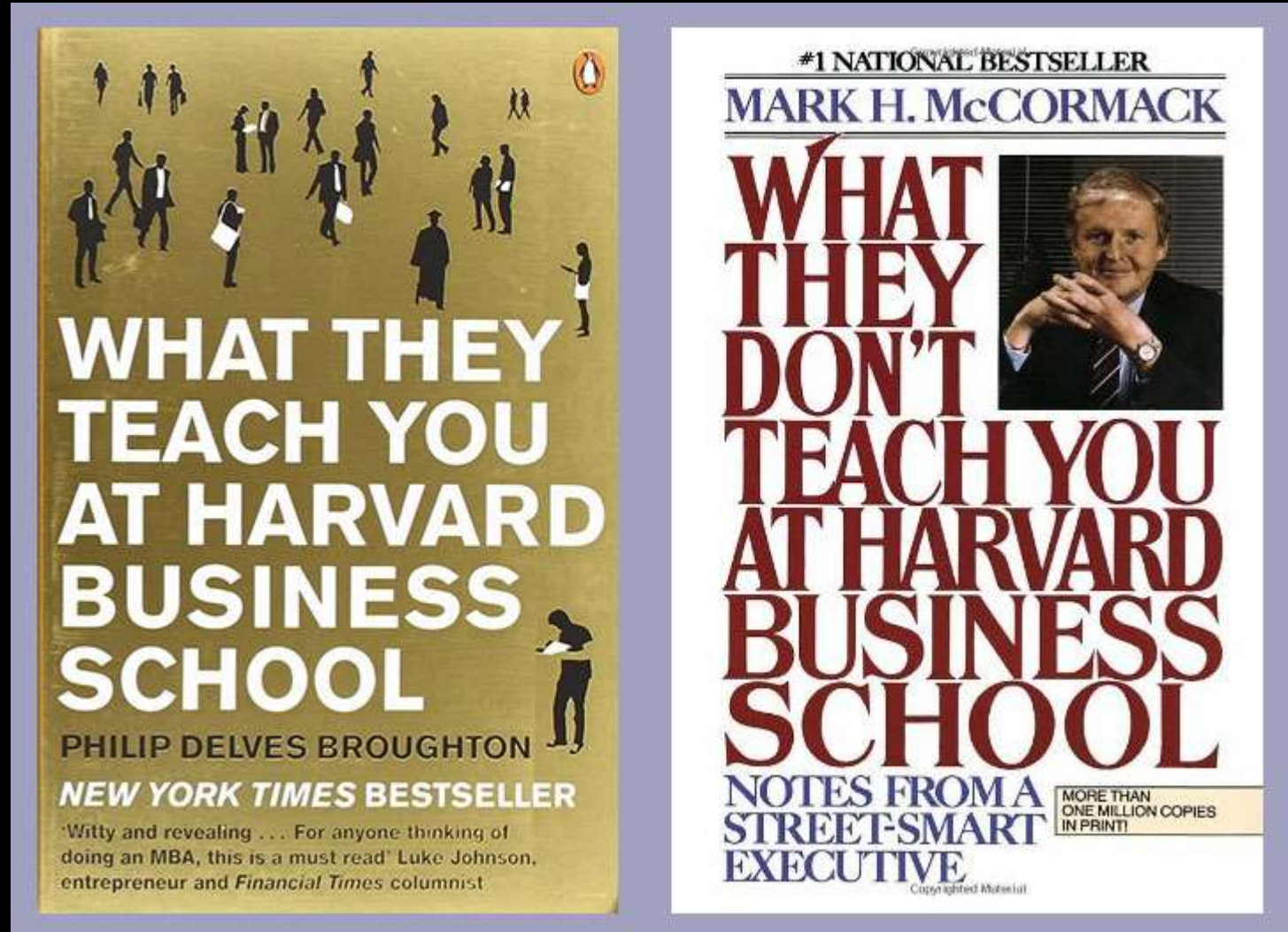
# What Should Stick

- Relational vs Non-Relational:
  - The basic difference
  - There are advantages to each
  - Scenarios where each is preferred
- Basic SQL Syntax
  - INSERT
  - UPDATE
  - WHERE
  - GROUP/ORDER BY
  - COUNT/SUM
  - JOIN ON



# Two books to rule them all

- Two books cover all of human knowledge...
- Today, we will also cover all of knowledge



Which makes me wonder why we're here.  
Just buy the book, you guys : )



# Everything you need to know

- We also cover all of knowledge:
- SQL and NoSQL
- However, remember that SQL is just a language
- The real distinction is relational vs non-relational



But first: Why have databases?

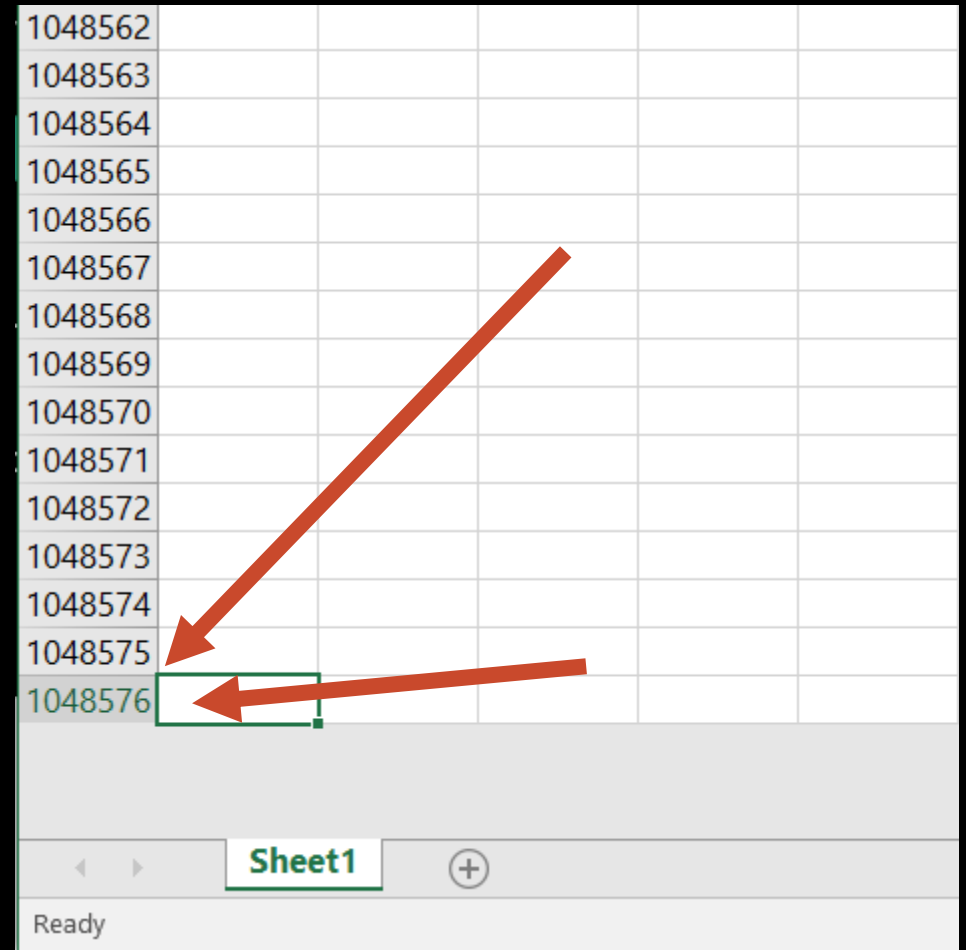
# Why Databases?

- Why have a database?
- Excel...
- Put simply: hit ctrl+down arrow in excel
- It tops out at 1 million entries



# WHHHHY?

- It's not even a power of 2...



The image shows a screenshot of an Excel spreadsheet. The left column contains row numbers from 1048562 to 1048576. The cell for row 1048576 is highlighted with a green border. Three red arrows point to this row: one from the left, one from the top-right, and one from the right. The spreadsheet interface includes a 'Sheet1' tab and a 'Ready' status bar.

1048562					
1048563					
1048564					
1048565					
1048566					
1048567					
1048568					
1048569					
1048570					
1048571					
1048572					
1048573					
1048574					
1048575					
1048576					

So, what is a Database?



# What is a Database?

- Well, it's a big 'ol file on the computer's hard disk
- But aren't databases *things*? Boxes I can kick?
  - Nope. Any computer can have a database on it, or not
  - Some happen to ONLY have a database on them, hence the naming

## Immediate questions:

- How is the file structured?
- And what happens if the file is too big for memory? Or for the whole computer?

# CAP Theorem [You can't have it all]

- Think of a particular service (Facebook, Netflix, Banking)
- Pick one to lose:
  - Consistency- Users always get the most recent data
  - Availability- Always get a response (even if out of date)
  - Partition Tolerance- Keeps working if the network goes to heck
- Well, the network *will* go to heck, so which other one do you give up?
- That tells you a lot about your application and whether you want SQL vs NoSQL

Note: If the network stays up you don't have to choose, but other tradeoffs still exist: see the **PACELC theorem**

# SQL vs NoSQL

Depends on the job

## SQL

- Consistency- You'll always see up-to-date data
- ACID- The database will properly handle [+10, -10], even if those commands are interrupted
- Requires cross-referencing many tables to understand an entry
- Hard to scale to multiple machines

## No SQL

- Availability- You'll get something reasonable
- ACID is negotiable- Maybe it's fine that edits to a Facebook post get lost
- Duplicates data, but once you find an entry you have it all
- Designed to scale to multiple machines [easy sharding]

## SQL

- Always uses the row/column model
- All entries have same properties, settled at database creation
- Built for never losing data, and always being right
- **Think: banking, shipping records**

## No SQL

- Lots of different flavors, each with pros and cons
- New features can be added to single records, on the fly
- Can be very reliable, or can be optimized for size/speed
- **Think: Facebook posts, YouTube videos**

# Relational vs Non-Relational

- The basic difference
  - Row/Column, multiple tables on a single machine
  - Many formats, designed to be split to multiple machines
- There are advantages to each
  - Maturity vs Scalability
- Scenarios where each is preferred
  - Banking is different than Twitter





# SQL Syntax

Please open

[https://www.w3schools.com/sql/trysql.asp?filename=trysql\\_create\\_table](https://www.w3schools.com/sql/trysql.asp?filename=trysql_create_table)

IN CHROME OR SAFARI (Otherwise, some examples won't work)

# Overview

- This database tracks orders fulfilled by a European grocer during a few months of 1996/1997
- It's heavily normalized- rather than having customer data in the Order table there's a customerId. Customers are recorded in their own table
- Our path: Customers, Categories, Orders, OrderDetails

Customers Table

# Customers

- Click on the Customers table
  - See how it wrote `SELECT * FROM` for us? The `*` gives us all columns
- Notice that any table can be read as “Each [customer] has a [Name], [Contact], [Address]...”
- What data types should these columns have?

Number of Records: 91

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obera Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden
6	Blauer See Delikatessen	Hanna Moos	Forsterstr. 57	Mannheim	68306	Germany
7	Blondel père et fils	Frédérique Citeaux	24, place Kléber	Strasbourg	67000	France
8	Bólido Comidas preparadas	Martin Sommer	C/ Araquil, 67	Madrid	28023	Spain
9	Bon app'	Laurence Labihans	12, rue des Bouchers	Marseille	13008	France
10	Bottom-Dollar Marketse	Elizabeth Lincoln	23 Tsawassen Blvd.	Tsawassen	T2F 8M4	Canada
11	B's Beverages	Victoria Ashworth	Fauntleroy Circus	London	EC2 5NT	UK
12	Cactus Comidas para llevar	Patricio Simpson	Cerrito 333	Buenos Aires	1010	Argentina

# Categories Table

# Categories

- Let's insert a new category

```
INSERT INTO Categories (colname,  
colname) VALUES (v1,v2)
```

- Specifically, add a category named Utensils and its description

```
INSERT INTO Categories  
(CategoryName, Description) VALUES  
( 'Utensils', 'Forks, glasses' )
```

- These INSERT statements are broadly how the database came to exist in the first place

Number of Records: 8		
CategoryID	CategoryName	Description
1	Beverages	Soft drinks, coffees, teas, beers, and ales
2	Condiments	Sweet and savory sauces, relishes, spreads, and seasonings
3	Confections	Desserts, candies, and sweet breads
4	Dairy Products	Cheeses
5	Grains/Cereals	Breads, crackers, pasta, and cereal
6	Meat/Poultry	Prepared meats
7	Produce	Dried fruit and bean curd
8	Seafood	Seaweed and fish



Orders Table

# Orders

- There's a lot of normalization going on...
  - What was normalization?
- Also, there are a lot of entries. Let's poke around...

```
SELECT * FROM [Orders] WHERE  
EmployeeID == 4
```

```
SELECT * FROM [Orders] WHERE EmployeeID == 4 AND CustomerID=10
```

```
SELECT * FROM [Orders] WHERE EmployeeID == 4 OR CustomerID=10
```

Number of Records: 196

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10248	90	5	1996-07-04	3
10249	81	6	1996-07-05	1
10250	34	4	1996-07-08	2
10251	84	3	1996-07-08	1
10252	76	4	1996-07-09	2
10253	34	3	1996-07-10	2
10254	14	5	1996-07-11	2
10255	68	9	1996-07-12	3
10256	88	3	1996-07-15	2
10257	35	4	1996-07-16	3
10258	20	1	1996-07-17	1

# Orders

```
SELECT * FROM [Orders] WHERE  
CustomerID=34
```

- What's this? Customer 34 is cheating on Employee 4! Let's fix that

- Mistake:

```
UPDATE Orders SET EmployeeID=4
```

- Correct:

```
UPDATE Orders SET EmployeeID=4  
WHERE CustomerID=34
```

Number of Records: 2

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10250	34	4	1996-07-08	2
10253	34	3	1996-07-10	2

Number of Records: 2

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10250	34	4	1996-07-08	2
10253	34	4	1996-07-10	2

# Orders

- Hmm, does a given customer always have the same shipper?

```
SELECT ShipperID, CustomerID FROM  
Orders ORDER BY CustomerID
```

- Notice that we can select only the columns we care about
- What does ORDER BY do?

Number of Records: 2

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10250	34	4	1996-07-08	2
10253	34	4	1996-07-10	2

Number of Records: 196

ShipperID	CustomerID
3	2
2	3
1	4
3	4
2	5
1	5

OrderDetails Table

# RECAP

- SELECT – Workhorse. Get specific columns
- INSERT -- Useful. Add new rows
- UPDATE – Useful. Change an existing row
- WHERE -- Workhorse. Specify which particular rows
- ORDER BY – Useful. How to sort the results





# OrderDetails

- Neat, there are multiple order details (products) per order!
- Up until now we've only seen one (customer, shipper, etc) per order

Number of Records: 518

OrderDetailID	OrderID	ProductID	Quantity
1	10248	11	12
2	10248	42	10
3	10248	72	5
4	10249	14	9
5	10249	51	40
6	10250	41	10
7	10250	51	35
8	10250	65	15
9	10251	22	6
10	10251	57	15
11	10251	65	20
12	10252	20	40
13	10252	33	25
14	10252	60	40
15	10253	31	20
16	10253	39	42
17	10253	49	40

# OrderDetails

- Which product is most popular?

## Mistake:

```
SELECT Quantity,ProductID FROM  
[OrderDetails] ORDER BY ProductID
```

## New Trick:

```
SELECT ProductID,SUM(Quantity) FROM  
[OrderDetails] GROUP BY ProductID
```

- Cool. But it orders by product.  
How can we make it prettier?

```
SELECT ProductID,SUM(Quantity) FROM  
[OrderDetails] GROUP BY ProductID  
ORDER BY SUM(Quantity)
```

Number of Records: 518

OrderDetailID	OrderID	ProductID	Quantity
1	10248	11	12
2	10248	42	10
3	10248	72	5
4	10249	14	9
5	10249	51	40
6	10250	41	10
7	10250	51	35

Number of Records: 77

ProductID	SUM(Quantity)
1	159
2	341
3	80
4	107
5	129
6	36
7	25
8	140
9	20
10	85
11	182

# Extension

- In addition to SUM you can COUNT, MIN, MAX, or AVG
  - Always wise to use these with GROUP BY
- DESC (Descending) can be added to ORDER BY to flip the order and put the biggest values on top

# What the heck is product 31, anyway?

- We can go look in the products table ourselves, like savages
- Or...

```
SELECT * FROM OrderDetails JOIN  
Products ON  
OrderDetails.ProductID=Products.  
ProductID
```

- Note that we do  
    TableName.ColumnName to  
    clarify
- Result is messy. I only want  
    Order, Product, Quantity,  
    Name... How do we do that?

Number of Records: 518

OrderDetailID	OrderID	ProductID	Quantity	ProductName	SupplierID	CategoryID	Unit	Price
1	10248	11	12	Queso Cabrales	5	4	1 kg pkg.	21
2	10248	42	10	Singaporean Hokkien Fried Mee	20	5	32 - 1 kg pkgs.	14
3	10248	72	5	Mozzarella di Giovanni	14	4	24 - 200 g pkgs.	34.8
4	10249	14	9	Tofu	6	7	40 - 100 g pkgs.	23.25
5	10249	51	40	Manjimup Dried Apples	24	7	50 - 300 g pkgs.	53
6	10250	41	10	Jack's New England Clam Chowder	19	8	12 - 12 oz cans	9.65
7	10250	51	35	Manjimup Dried Apples	24	7	50 - 300 g pkgs.	53
8	10250	65	15	Louisiana Fiery Hot Pepper Sauce	2	2	32 - 8 oz bottles	21.05
9	10251	22	6	GustaF's Knäckebröd	9	5	24 - 500 g pkgs.	21
10	10251	57	15	Ravioli Angelo	26	5	24 - 250 g pkgs.	19.5
11	10251	65	20	Louisiana Fiery Hot Pepper Sauce	2	2	32 - 8 oz bottles	21.05
12	10252	20	40	Sir Rodney's Marmalade	8	3	30 gift boxes	81
13	10252	33	25	Geitost	15	4	500 g	2.5

# RECAP

- SELECT – Workhorse. Get specific columns
- INSERT -- Useful. Add new rows
- UPDATE – Useful. Change an existing row
- WHERE -- Workhorse. Specify which particular rows
- ORDER BY – Useful. How to sort the results
- COUNT – Workhorse. The most fundamental kind of math
- SUM/MAX/MIN/AVG – Workhorse.
- GROUP BY – Workhorse. Combines to one response per unique entry
- JOIN – Workhorse. Ties two normalized tables back together



# Syntax

```
SELECT a FROM b WHERE c GROUP BY d;
```

A- List the columns you want. Can be real columns or derived ones like SUM(Quantity) or COUNT(ID)

B- Describe the table, often including one or more JOIN \_\_\_\_ ON \_\_\_\_

C- Describe the rows you care about, e.g. EmployeeID=5 AND ...

D- Set which column(s) to group/order by. Very important for counts/sums

# Practice

Please open

[https://www.w3schools.com/sql/trysql.asp?filename=trysql\\_create\\_table](https://www.w3schools.com/sql/trysql.asp?filename=trysql_create_table)

IN CHROME OR SAFARI (Otherwise, some examples won't work)

# Practice Questions

- Q1: How many customers in Germany?
  - A: 11
- Q2: How many units of product 29 did we ship?
  - A: 168
- Q3: How many orders in October?
  - A: 26. Research the BETWEEN clause
- Q4: Which country has the most customers?
  - USA. But you should be able to tell me who is tied for 2<sup>nd</sup>
- Q5: Which country has the most orders?
  - USA with 29. But should be able to tell me and verify the number in Belgium
- Q6: Which customer placed the most orders?
  - Ernst Handel with 10



# Practice Answers

- **Q1: How many customers in Germany?**
  - `SELECT * FROM Customers WHERE Country='Germany'`
  - `SELECT Count(*) FROM Customers WHERE Country='Germany'`
- **Q2: How many units of product 29 did we ship?**
  - `SELECT Sum(Quantity) FROM [OrderDetails] where ProductID=29`
- **Q3: How many orders in October?**
  - `SELECT Count(*) FROM [Orders] WHERE OrderDate BETWEEN '1996-10-01' AND '1996-10-31'`
- **Q4: Which country has the most customers?**
  - `SELECT Count(*),Country FROM [Customers] GROUP BY Country`
- **Q5: Which country has the most orders?**
  - `SELECT count(*),Customers.Country FROM [Orders] JOIN Customers on Customers.CustomerID=Orders.CustomerID GROUP BY Customers.Country`
- **Q6: Which customer placed the most orders**
  - `SELECT Count(*),CustomerName FROM Orders JOIN Customers ON Customers.CustomerID=Orders.CustomerID GROUP BY Orders.CustomerID ORDER BY Count(*) DESC`

# Really tough question

- Q: Which employee shipped the most products from New England Seafood Cannery?
  - A: Employee 2 with 98 (More than everyone else combined)

```
SELECT SUM(Quantity), EmployeeID
FROM OrderDetails
JOIN Products on Products.ProductID=OrderDetails.ProductID
JOIN Suppliers on Suppliers.SupplierID=Products.ProductID
JOIN Orders on Orders.OrderID=OrderDetails.OrderID
WHERE Suppliers.SupplierName='New England Seafood Cannery'
GROUP BY EmployeeID
```