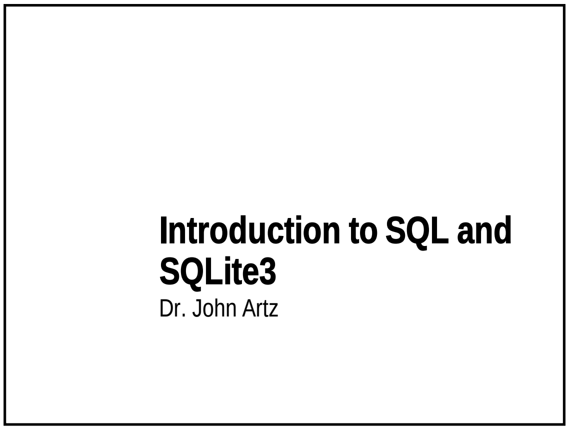
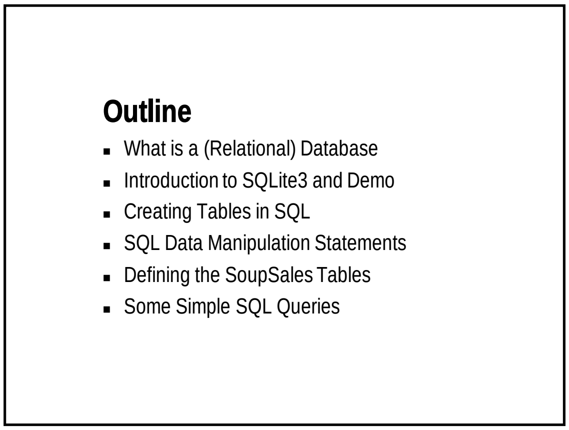


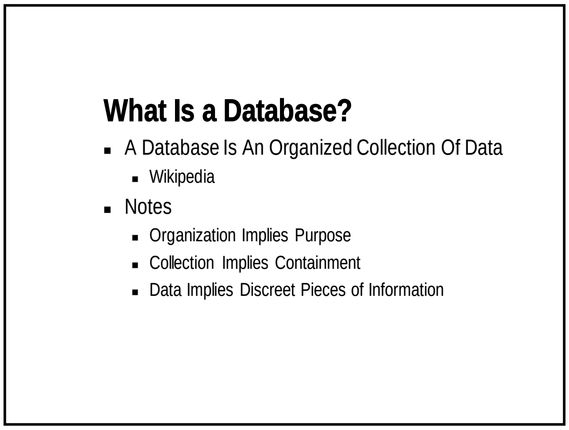
Introduction to SQL and SQLite3

Dr. John Artz



Outline

- What is a (Relational) Database
- Introduction to SQLite3 and Demo
- Creating Tables in SQL
- SQL Data Manipulation Statements
- Defining the SoupSales Tables
- Some Simple SQL Queries



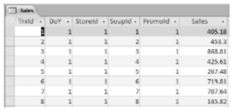
What Is a Database?

- A Database Is An Organized Collection Of Data
  - Wikipedia
- Notes
  - Organization Implies Purpose
  - Collection Implies Containment
  - Data Implies Discreet Pieces of Information

What Is a Relational Database?

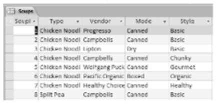
- A relational database is a digital database whose organization is based on the relational model of data, as proposed by E.F. Codd in 1970. This model organizes data into one or more tables (or "relations") of rows and columns, with a unique key for each row.
  - Wikipedia

A Relational Database is all Tables



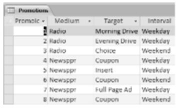
Sales

TrxId	DoY	StoreId	SoupId	PromoId	Sales
1	1	1	1	1	405.18
2	1	1	2	1	453.3
3	1	1	3	1	868.81
4	1	1	4	1	425.61
5	1	1	5	1	267.48
6	1	1	6	1	719.81
7	1	1	7	1	707.64
8	1	1	8	1	165.82



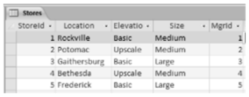
Soups

SoupI	Type	Vendor	Mode	Style
1	Chicken Noodl	Progresso	Canned	Basic
2	Chicken Noodl	Campbells	Canned	Basic
3	Chicken Noodl	Lipton	Dry	Basic
4	Chicken Noodl	Campbells	Canned	Chunky
5	Chicken Noodl	Wolfgang Puck	Canned	Gourmet
6	Chicken Noodl	Pacific Organic	Boxed	Organic
7	Chicken Noodl	Healthy Choice	Canned	Healthy
8	Split Pea	Campbells	Canned	Basic



Promotions

PromoIc	Medium	Target	Interval
1	Radio	Morning Drive	Weekday
2	Radio	Evening Drive	Weekday
3	Radio	Choice	Weekend
4	Newsppr	Coupon	Weekday
5	Newsppr	Insert	Weekday
6	Newsppr	Coupon	Weekend
7	Newsppr	Full Page Ad	Weekday
8	Newsppr	Coupon	Weekend



Stores

StoreId	Location	Elevatio	Size	MgrId
1	Rockville	Basic	Medium	1
2	Potomac	Upscale	Medium	2
3	Gaithersburg	Basic	Large	3
4	Bethesda	Upscale	Medium	4
5	Frederick	Basic	Large	5

And Not Just Any Tables

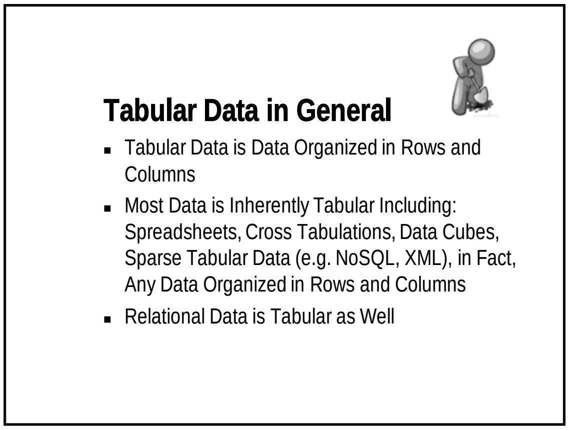
Tables Must Be ***Properly Designed***

□□□□

But, We Will Get To That Later

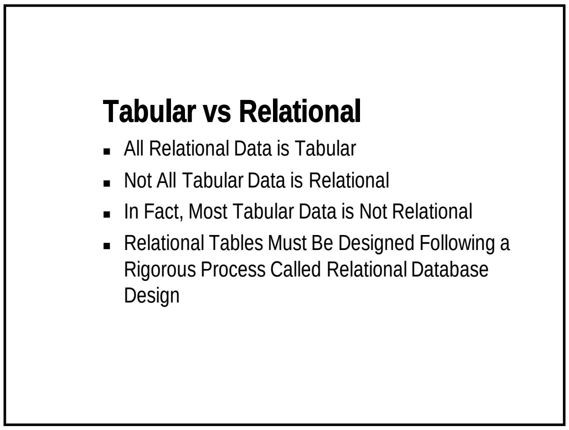
Poorly Designed Tables Are Worse Than Useless

They Provide Incorrect Information



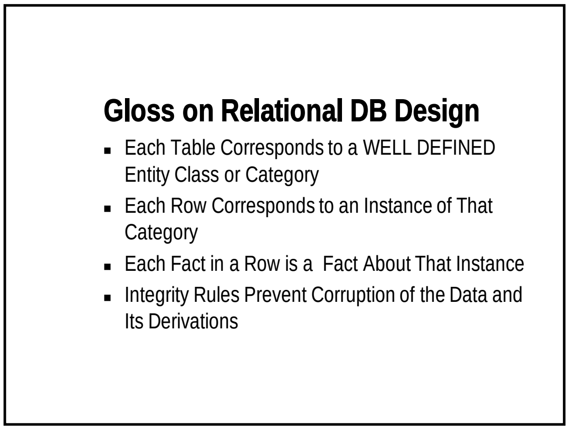
Tabular Data in General

- Tabular Data is Data Organized in Rows and Columns
- Most Data is Inherently Tabular Including: Spreadsheets, Cross Tabulations, Data Cubes, Sparse Tabular Data (e.g. NoSQL, XML), in Fact, Any Data Organized in Rows and Columns
- Relational Data is Tabular as Well



Tabular vs Relational

- All Relational Data is Tabular
- Not All Tabular Data is Relational
- In Fact, Most Tabular Data is Not Relational
- Relational Tables Must Be Designed Following a Rigorous Process Called Relational Database Design



Gloss on Relational DB Design

- Each Table Corresponds to a WELL DEFINED Entity Class or Category
- Each Row Corresponds to an Instance of That Category
- Each Fact in a Row is a Fact About That Instance
- Integrity Rules Prevent Corruption of the Data and Its Derivations

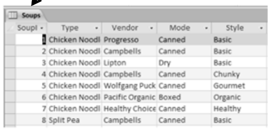
How Are Tables Designed Correctly?

- Conceptual Database Design Helps Determine if Categories are Well Formed
- Logical Database Design Helps Determine if Tables are Well Formed
- Integrity Rules Help Insure That Queries Produce the Expected Results

An Example

Categories (Relations) Must be Well Defined

Row or Table Design Has Rules Too



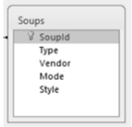
Soups

SoupI	Type	Vendor	Mode	Style
1	Chicken Noodl	Progresso	Canned	Basic
2	Chicken Noodl	Campbells	Canned	Basic
3	Chicken Noodl	Lipton	Dry	Basic
4	Chicken Noodl	Campbells	Canned	Chunky
5	Chicken Noodl	Wolfgang Puck	Canned	Gourmet
6	Chicken Noodl	Pacific Organic	Boxed	Organic
7	Chicken Noodl	Healthy Choice	Canned	Healthy
8	Split Pea	Campbells	Canned	Basic

Integrity Rules Require Rows to be the Same Kind of Thing And Address Relationships Between Categories

Relational vs. Not Relational (XML)

Relational



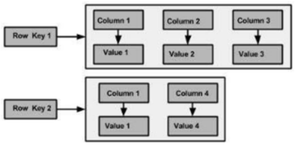
This is About a Can of Soup

It is Unclear What This is About

Not Relational (Now NoSQL)

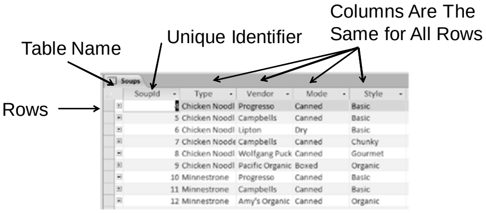
```
<Soup>
<Id>1</Id>
<Type>Chicken Noodle</Type>
<Vendor>Progresso</Vendor>
        <Mode>Canned</Mode>
<Vendor>Campbells</Vendor>
        <Mode>Canned</Mode>
<Vendor>Lipton</Vendor>
        <Mode>Dried</Mode>
</Soup>
```

Relational vs. NoSQL (Cassandra)



Note: Each Row Can Have Different Attributes
Data is Accessible via Host Language Program
How Difficult Would a Query Be?

A Relational Table



SoupId	Type	Vendor	Mode	Style
[illegible]	Chicken Noodl	Progresso	Canned	Basic
5	Chicken Noodl	Campbells	Canned	Basic
6	Chicken Noodl	Lipton	Dry	Basic
7	Chicken Noode	Campbells	Canned	Chunky
8	Chicken Noodl	Wolfgang Puck	Canned	Gourmet
9	Chicken Noodl	Pacific Organic	Boxed	Organic
10	Minnestrone	Progresso	Canned	Basic
11	Minnestrone	Campbells	Canned	Basic
12	Minnestrone	Amy's Organic	Canned	Organic

Data is Accessible Using SQL
Host Language Programs Are Not Necessary

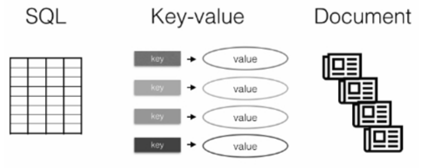
It Looks a Lot Like a Spreadsheet

- A Relational Table Does Look a Lot Like a Spreadsheet
- But, There is No Design Theory for Spreadsheets
- And Spreadsheet Technology Couldn't Handle it if There Was
- But, There is Design Theory for Relational Tables
- This Makes All the Difference in the World

Relational Database Snobbery

Only This is
a Database

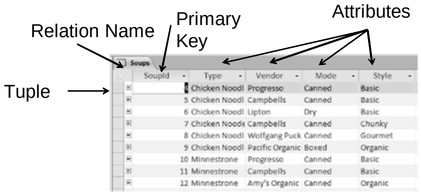
These Are Not
They Are Just for Storage



Relational Terminology

- We Call a Table a Relation
- We Call a Column an Attribute
- We Call a Row a Tuple
- We Call the Unique Identifier, a Primary Key

A Relational Table, Revisited



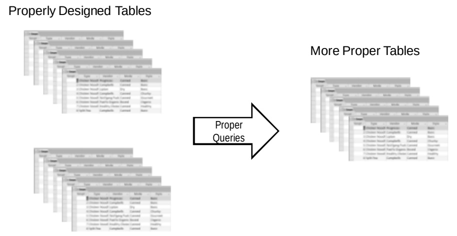
SoupId	Type	Vendor	Mode	Style
[illegible]	Chicken Noodl	Progresso	Canned	Basic
5	Chicken Noodl	Campbells	Canned	Basic
6	Chicken Noodl	Lipton	Dry	Basic
7	Chicken Noode	Campbells	Canned	Chunky
8	Chicken Noodl	Wolfgang Puck	Canned	Gourmet
9	Chicken Noodl	Pacific Organic	Boxed	Organic
10	Minnestrone	Progresso	Canned	Basic
11	Minnestrone	Campbells	Canned	Basic
12	Minnestrone	Amy's Organic	Canned	Organic

Attributes are Facts About the Thing
Identified By the Primary Key

Relational Design Restrictions

- All Rows Must Have the Same Attributes
- All Rows Must Have Unique Identifiers
- All Rows Must Be Instances of the Same Thing
- But, If You Follow All the Rules, You Can Derive New Information Using a High Level Query Language Such as Structured Query Language, or SQL

Relationally Derived Information

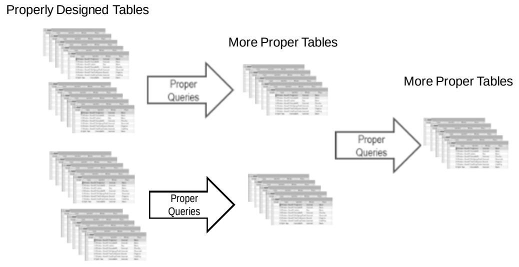


Properly Designed Tables

Proper Queries

More Proper Tables

And It Doesn't Stop There



Properly Designed Tables

Proper Queries

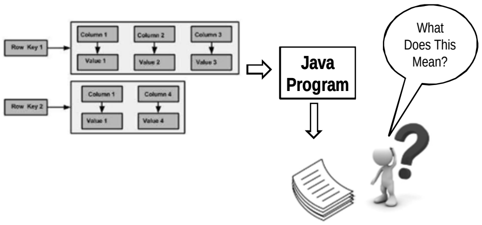
More Proper Tables

Proper Queries

More Proper Tables

Proper Queries

NoSQL Derivations



Can't Go Any
Farther With This

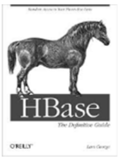
Relational Performance Problems

- Relational Databases Must Maintain Data Integrity in Order Answer Queries Correctly
  - Data Value Constraints
  - Concurrency Control
  - Transaction Models
- This Makes Relational Databases (Currently) Unacceptable for Big Data Applications

Relational DBMSs

Microsoft SQL Server
SQLite
MySQL
IBM DB2
ORACLE
SYBASE

NoSQL Databases



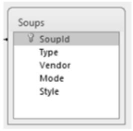
Big Data Has Given Rise To
Numerous NoSQL Databases

Relational DBs Vs. NoSQL

- Relational Databases Are For the Derivation and Delivery of Information
  - They Focus on Design and Exploitation Discipline
- NoSQL Databases Are For the Storage and Retrieval of Data
  - They Focus on Storage and Retrieval Performance

Again: Relational vs. Not Relational

Relational



Soups
SoupId
Type
Vendor
Mode
Style

Structured Data

Not Relational

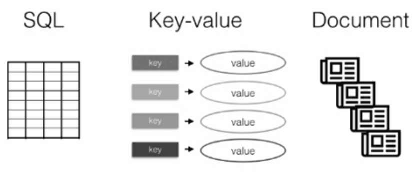
```
<Soup>
<Id>1</Id>
<Type>Chicken Noodle</Type>
<Vendor>Progresso</Vendor>
        <Mode>Canned</Mode>
<Vendor>Campbells</Vendor>
        <Mode>Canned</Mode>
<Vendor>Lipton</Vendor>
        <Mode>Dried</Mode>
</Soup>
```

Semi- Structured Data

Why Relational Snobbery?

Only This is a Database

These Are Not



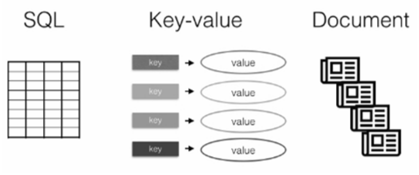
This is For Information

These Are For Storage

Why NoSQL Snobbery?

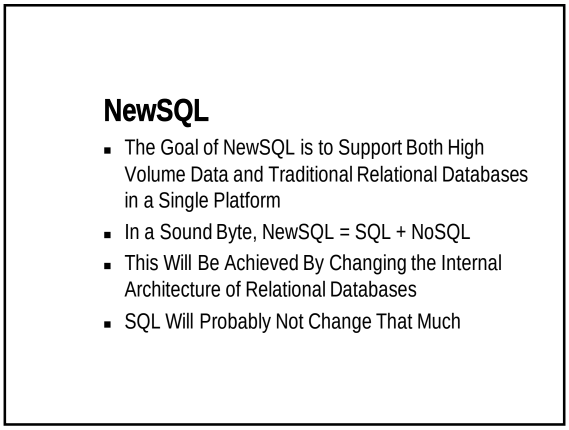
This is Slow, Restrictive And Low Volume

These Are Fast, Flexible and High Volume



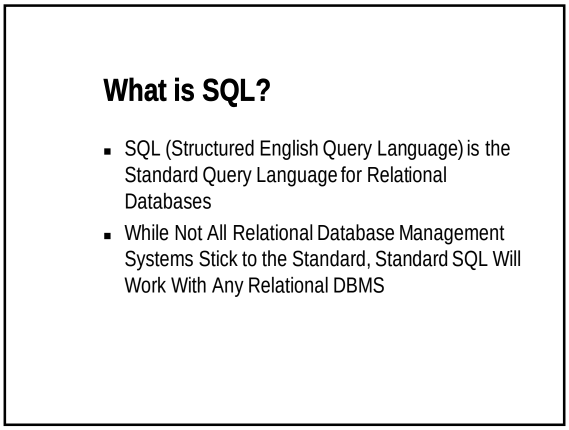
Soon, This Will All Change

SCALEBASE
ScaleArc
AKIBAN
Tokutek
dbShards
ScaleDB
SCHOONER
NewSQL
JustOne
VoltDB
NuoDB
SQLFire
TRANSLATTICE
xeround
GENIEDB
H-Store
Clustrix



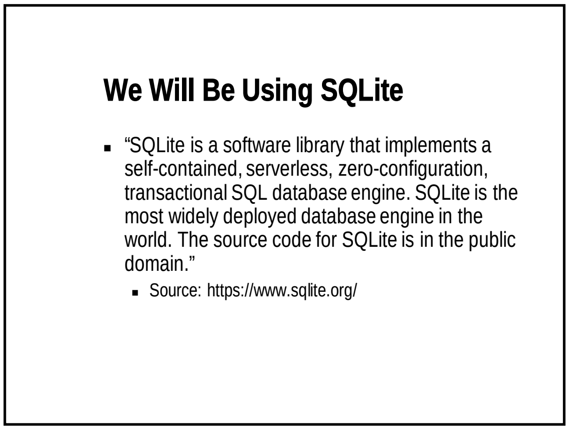
NewSQL

- The Goal of NewSQL is to Support Both High Volume Data and Traditional Relational Databases in a Single Platform
- In a Sound Byte, NewSQL = SQL + NoSQL
- This Will Be Achieved By Changing the Internal Architecture of Relational Databases
- SQL Will Probably Not Change That Much



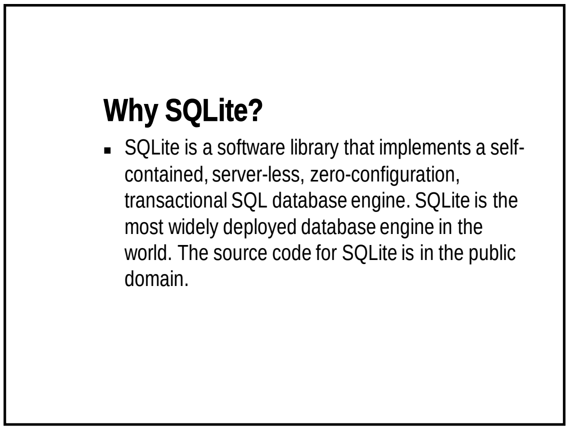
What is SQL?

- SQL (Structured English Query Language) is the Standard Query Language for Relational Databases
- While Not All Relational Database Management Systems Stick to the Standard, Standard SQL Will Work With Any Relational DBMS



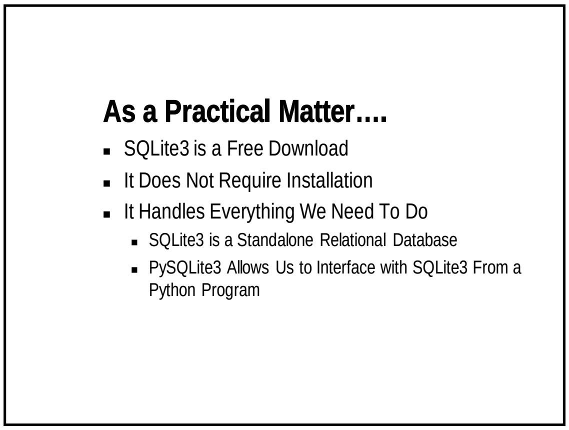
We Will Be Using SQLite

- “SQLite is a software library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. SQLite is the most widely deployed database engine in the world. The source code for SQLite is in the public domain.”
  - Source: https://www.sqlite.org/



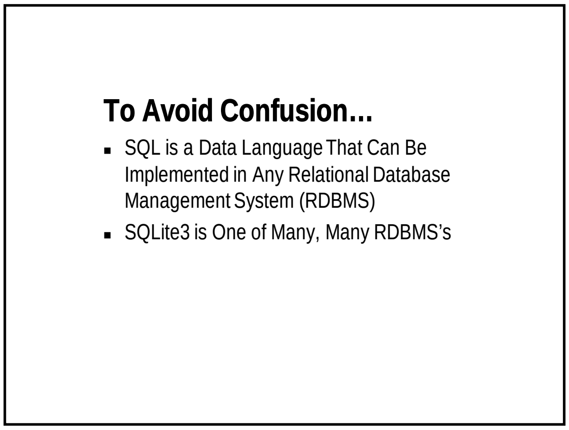
Why SQLite?

- SQLite is a software library that implements a self-contained, server-less, zero-configuration, transactional SQL database engine. SQLite is the most widely deployed database engine in the world. The source code for SQLite is in the public domain.



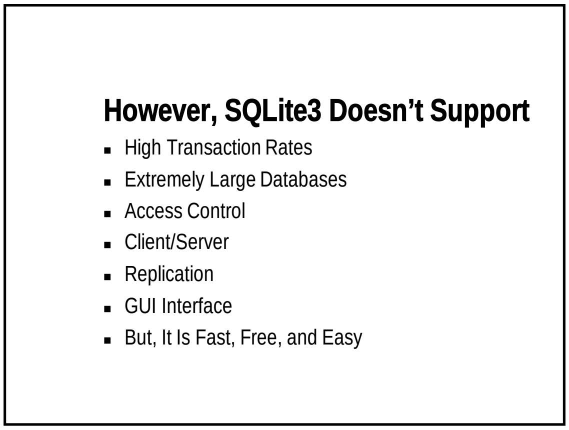
As a Practical Matter….

- SQLite3 is a Free Download
- It Does Not Require Installation
- It Handles Everything We Need To Do
  - SQLite3 is a Standalone Relational Database
  - PySQLite3 Allows Us to Interface with SQLite3 From a Python Program



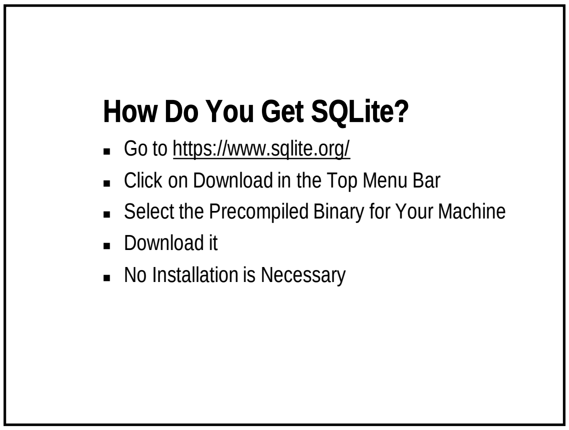
To Avoid Confusion...

- SQL is a Data Language That Can Be Implemented in Any Relational Database Management System (RDBMS)
- SQLite3 is One of Many, Many RDBMS's



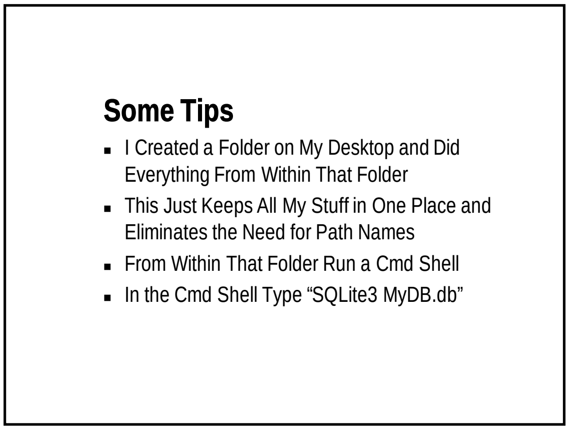
However, SQLite3 Doesn't Support

- High Transaction Rates
- Extremely Large Databases
- Access Control
- Client/Server
- Replication
- GUI Interface
- But, It Is Fast, Free, and Easy



How Do You Get SQLite?

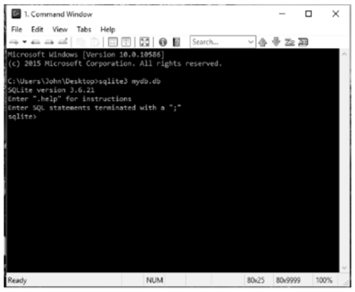
- Go to https://www.sqlite.org/
- Click on Download in the Top Menu Bar
- Select the Precompiled Binary for Your Machine
- Download it
- No Installation is Necessary



Some Tips

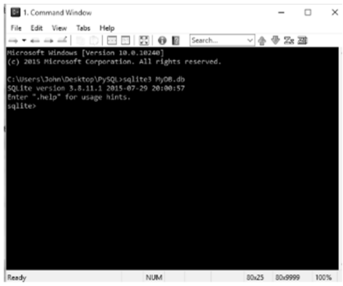
- I Created a Folder on My Desktop and Did Everything From Within That Folder
- This Just Keeps All My Stuff in One Place and Eliminates the Need for Path Names
- From Within That Folder Run a Cmd Shell
- In the Cmd Shell Type "SQLite3 MyDB.db"

From the Cmd Shell



```
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\John\Desktop>sqlite3 mydb.db
SQLite version 3.6.21
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite>
```

The SQLite Prompt



```
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\John\Desktop\PySQL>sqlite3 MyDB.db
SQLite version 3.8.11.1 2015-07-29 20:00:57
Enter ".help" for usage hints.
sqlite>
```

A Quick SQLite3 Demo

- Invoke SQLite3
- Create a Table
- Verify the Table
- Insert a Row
- Verify the Row
- Update the Row
- Verify the Update

Some Handy dot Commands

- .help – list dot commands
- .tables – list tables
- .exit or .quit – close the SQLite3 Window
- .read *filename* – read SQL or SQLite3 commands from *filename*
- .import *filename table-name* – Import data from *filename* into *table-name*

SQL Statement Types

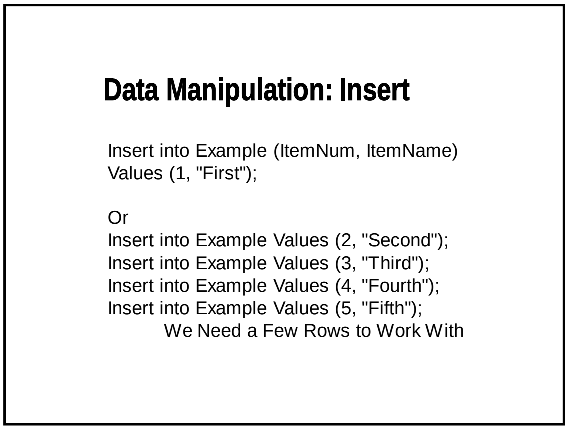
- Data Definition (Create Table)
- Data Manipulation (Insert, Update, Delete)
- Data Retrieval (Select)
- Multi-Table Selection (Joins)
- Advanced Summary (Group By)
- View Definition (Create View)

Create Table

```
Create table Example (
ItemNum int,
ItemName text);  ← Don't Forget The
                            SemiColon
```

Verify With .tables

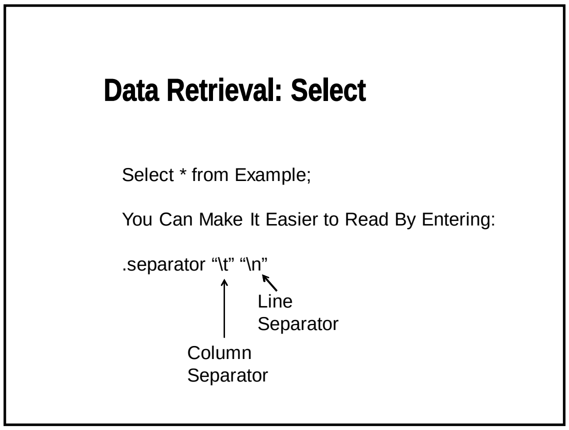
```
Drop Table Example;  ← This Will Get Rid of It
```



Data Manipulation: Insert

```
Insert into Example (ItemNum, ItemName)
Values (1, "First");

Or
Insert into Example Values (2, "Second");
Insert into Example Values (3, "Third");
Insert into Example Values (4, "Fourth");
Insert into Example Values (5, "Fifth");
```

We Need a Few Rows to Work With



Data Retrieval: Select

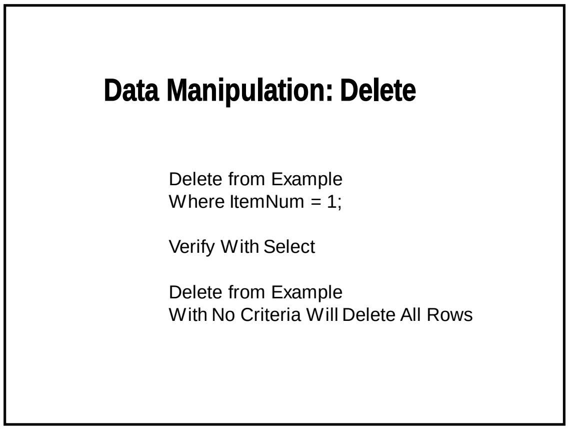
```
Select * from Example;
```

You Can Make It Easier to Read By Entering:

```
.separator “\t” “\n”
```

Line Separator

Column Separator



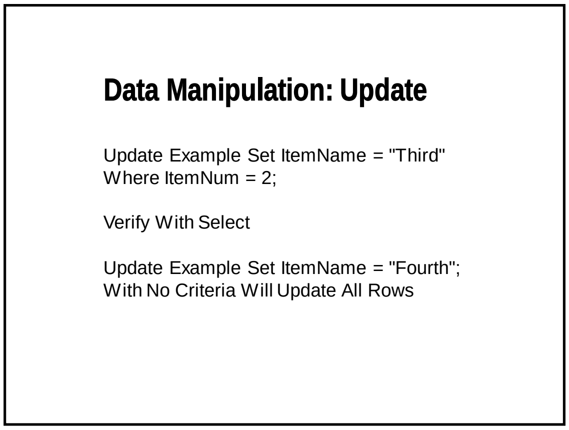
Data Manipulation: Delete

```
Delete from Example
Where ItemNum = 1;
```

Verify With Select

```
Delete from Example
```

With No Criteria Will Delete All Rows

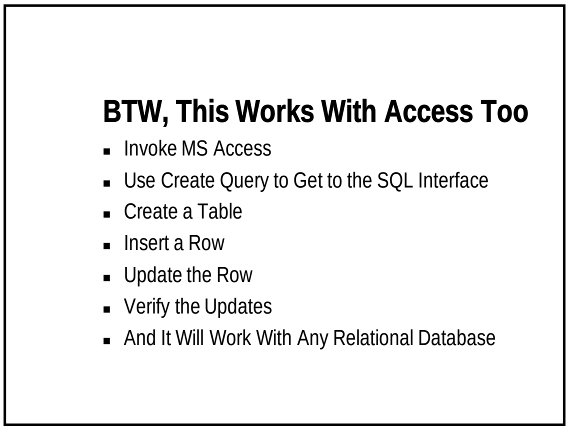


Data Manipulation: Update

Update Example Set ItemName = "Third"
Where ItemNum = 2;

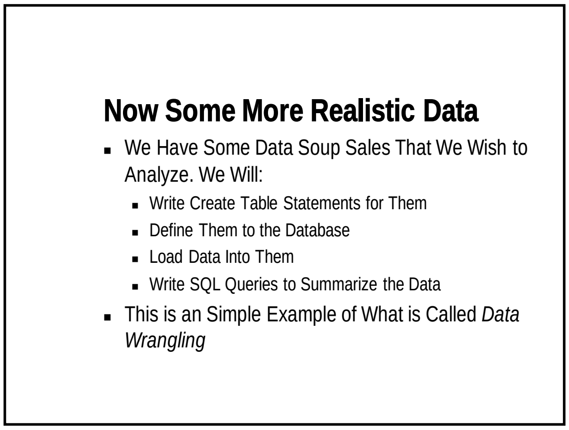
Verify With Select

Update Example Set ItemName = "Fourth";
With No Criteria Will Update All Rows



BTW, This Works With Access Too

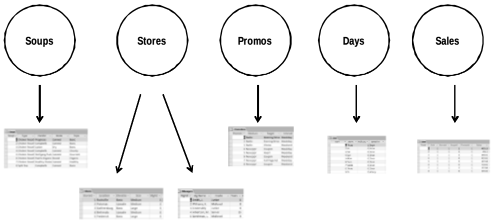
- Invoke MS Access
- Use Create Query to Get to the SQL Interface
- Create a Table
- Insert a Row
- Update the Row
- Verify the Updates
- And It Will Work With Any Relational Database



Now Some More Realistic Data

- We Have Some Data Soup Sales That We Wish to Analyze. We Will:
  - Write Create Table Statements for Them
  - Define Them to the Database
  - Load Data Into Them
  - Write SQL Queries to Summarize the Data
- This is an Simple Example of What is Called *Data Wrangling*

Soup Sales Tables



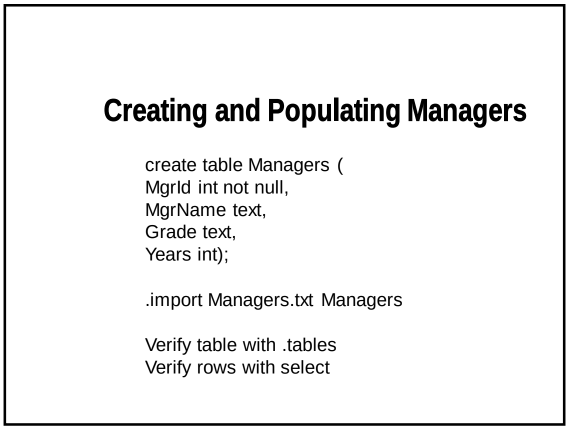
Now, We Will…

- Define the Tables Using SQL
- Use That SQL to Create the Tables in SQLite3
- Use SQL to Populate and Update the Tables
- See How to Bulk Load Data Into Tables
- And Write Some Simple SQL Queries Using Those Tables to Answer Some Questions

Creating and Populating Days

```
create table Days (
DoY int not null,
DoW text,
Holiday int,
Weather text);

.import Days.txt Days
```

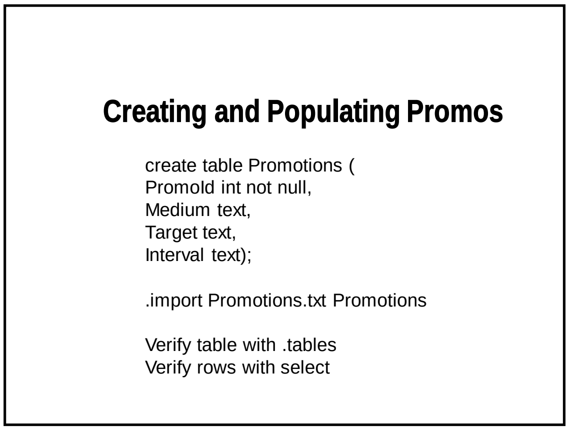
Verify table with .tables
Verify rows with select



Creating and Populating Managers

```
create table Managers (
MgrId int not null,
MgrName text,
Grade text,
Years int);

.import Managers.txt Managers
```

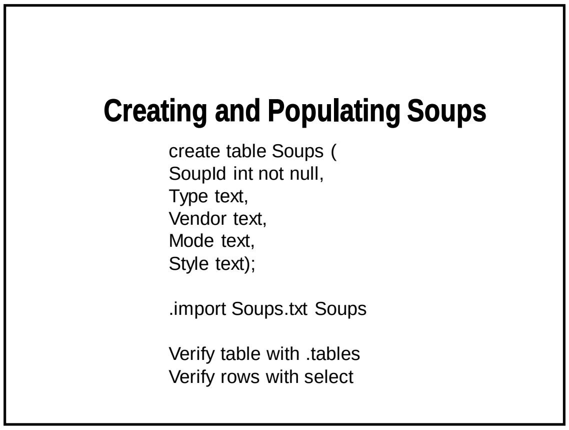
Verify table with .tables
Verify rows with select



Creating and Populating Promos

```
create table Promotions (
PromoId int not null,
Medium text,
Target text,
Interval text);

.import Promotions.txt Promotions
```

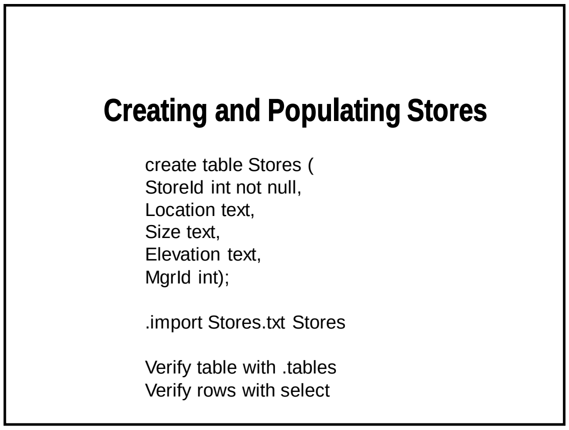
Verify table with .tables
Verify rows with select



Creating and Populating Soups

```
create table Soups (
SoupId int not null,
Type text,
Vendor text,
Mode text,
Style text);

.import Soups.txt Soups
```

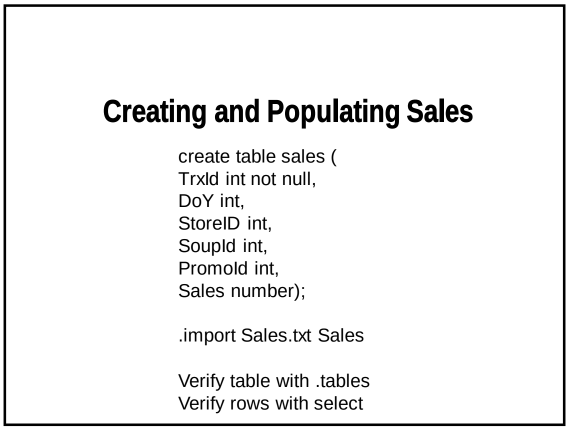
Verify table with .tables
Verify rows with select



Creating and Populating Stores

```
create table Stores (
StoreId int not null,
Location text,
Size text,
Elevation text,
MgrId int);

.import Stores.txt Stores
```

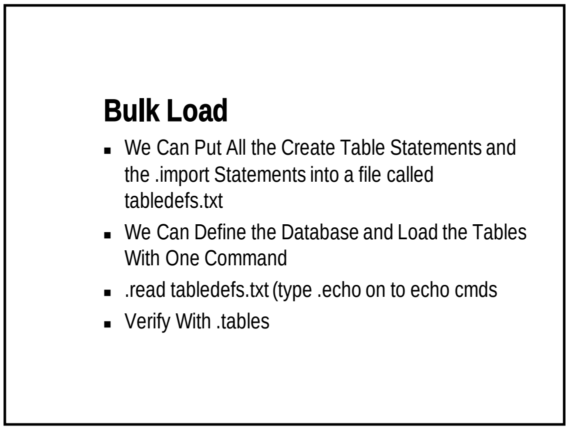
Verify table with .tables
Verify rows with select



Creating and Populating Sales

```
create table sales (
TrxId int not null,
DoY int,
StoreID int,
SoupId int,
PromoId int,
Sales number);

.import Sales.txt Sales
```

Verify table with .tables
Verify rows with select

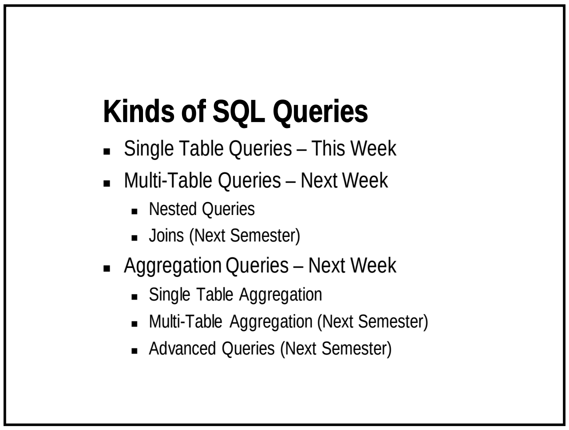


Bulk Load

- We Can Put All the Create Table Statements and the .import Statements into a file called tabledefs.txt
- We Can Define the Database and Load the Tables With One Command
- .read tabledefs.txt (type .echo on to echo cmds
- Verify With .tables

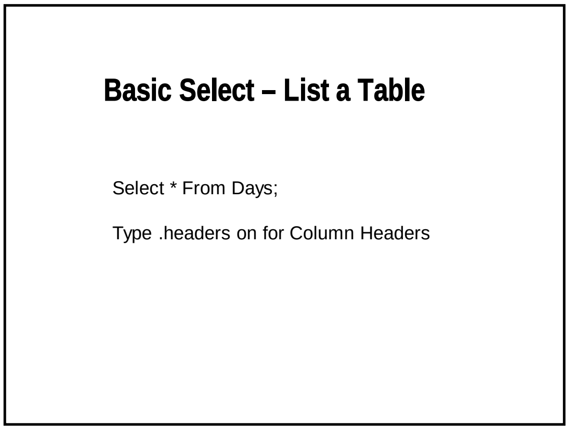
Now That We Have Data….

Let's Write
Some SQL
Queries



Kinds of SQL Queries

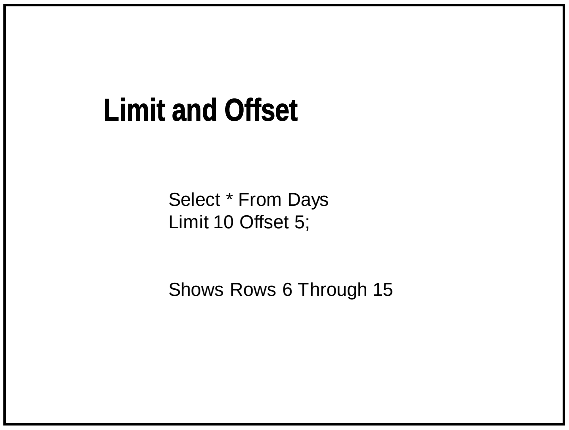
- Single Table Queries – This Week
- Multi-Table Queries – Next Week
  - Nested Queries
  - Joins (Next Semester)
- Aggregation Queries – Next Week
  - Single Table Aggregation
  - Multi-Table Aggregation (Next Semester)
  - Advanced Queries (Next Semester)



Basic Select – List a Table

Select * From Days;

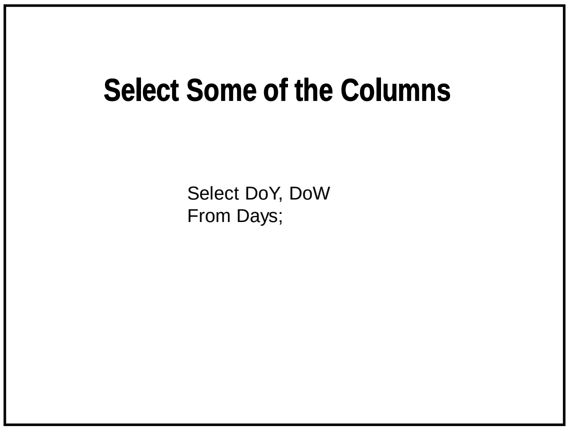
Type .headers on for Column Headers



Limit and Offset

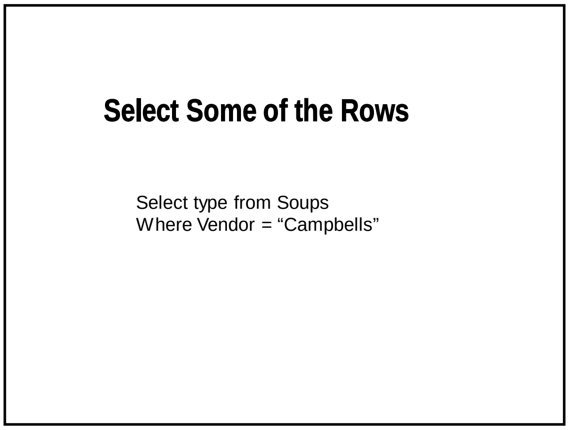
```
Select * From Days
Limit 10 Offset 5;
```

Shows Rows 6 Through 15



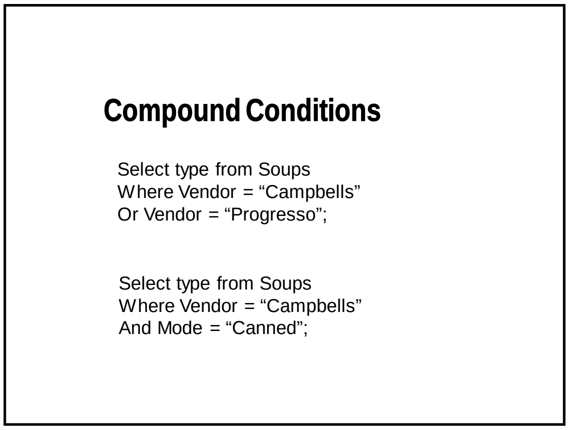
Select Some of the Columns

```
Select DoY, DoW
From Days;
```



Select Some of the Rows

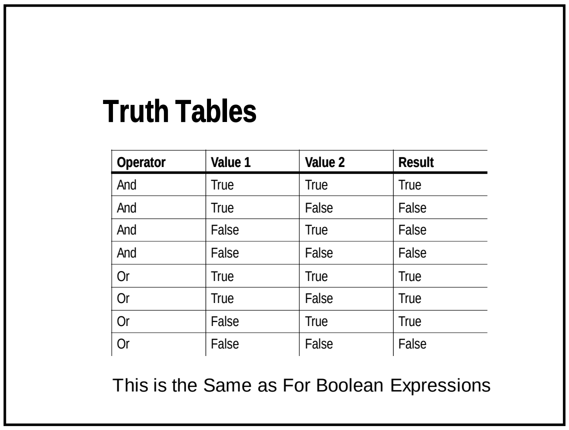
```
Select type from Soups
Where Vendor = “Campbells”
```



Compound Conditions

```
Select type from Soups
Where Vendor = “Campbells”
Or Vendor = “Progresso”;
```

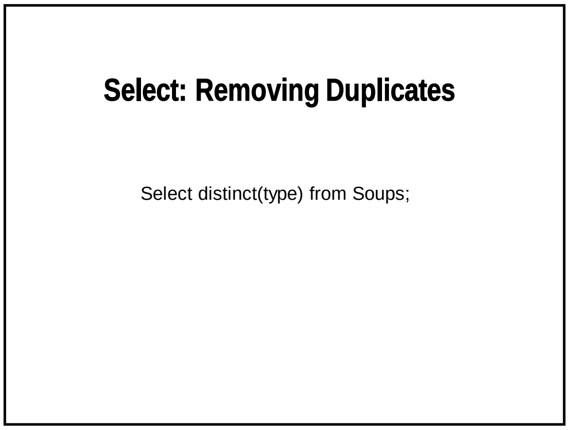
```
Select type from Soups
Where Vendor = “Campbells”
And Mode = “Canned”;
```



Truth Tables

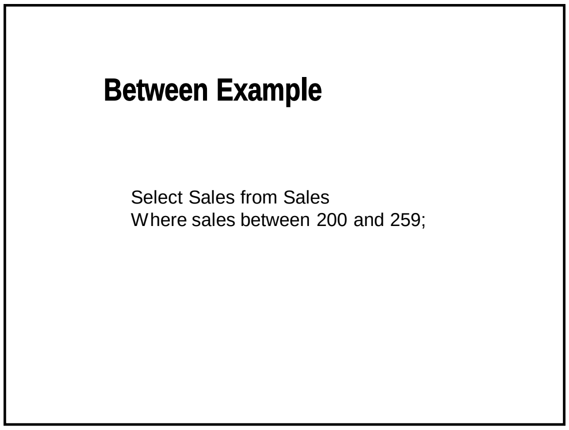
Operator	Value 1	Value 2	Result
And	True	True	True
And	True	False	False
And	False	True	False
And	False	False	False
Or	True	True	True
Or	True	False	True
Or	False	True	True
Or	False	False	False

This is the Same as For Boolean Expressions



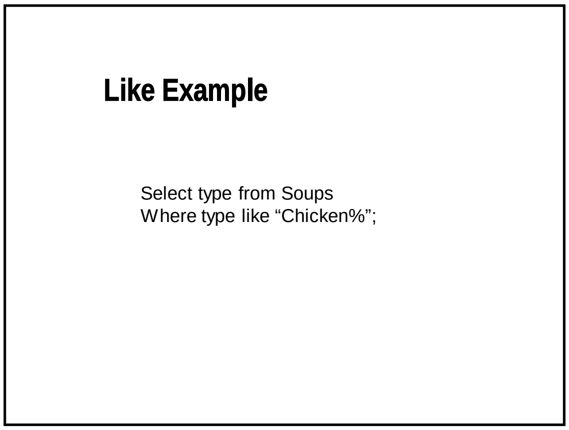
Select: Removing Duplicates

```
Select distinct(type) from Soups;
```



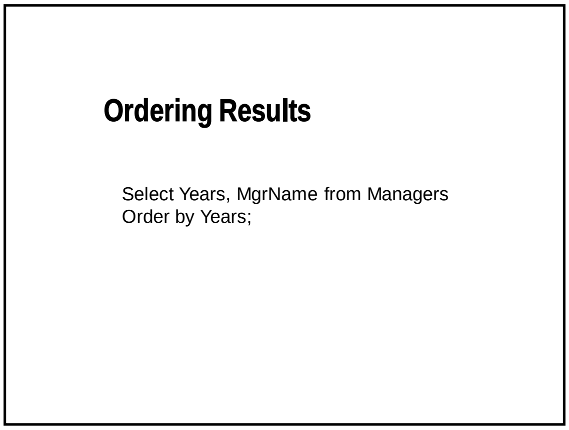
Between Example

```
Select Sales from Sales
Where sales between 200 and 259;
```



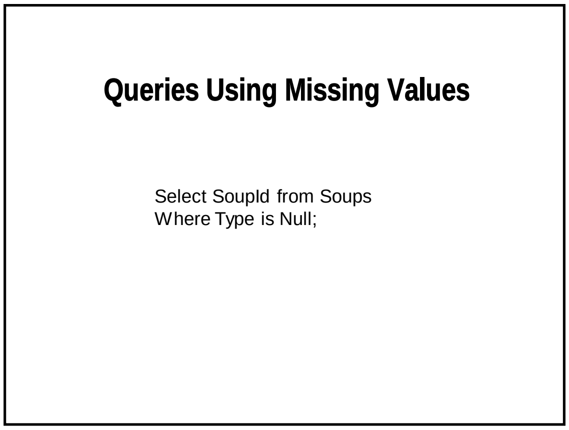
Like Example

```
Select type from Soups
Where type like “Chicken%”;
```



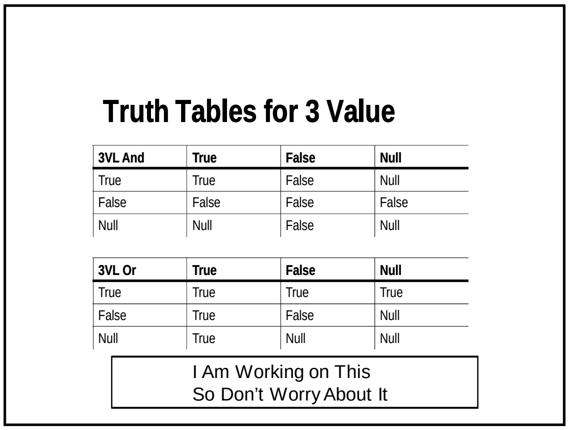
Ordering Results

```
Select Years, MgrName from Managers
Order by Years;
```



Queries Using Missing Values

```
Select SoupId from Soups
Where Type is Null;
```

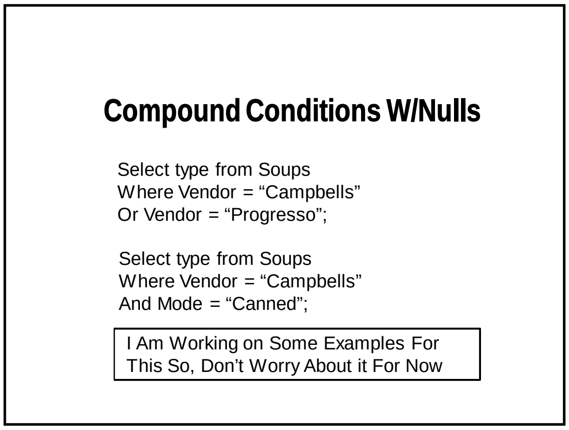


Truth Tables for 3 Value

3VL And	True	False	Null
True	True	False	Null
False	False	False	False
Null	Null	False	Null

3VL Or	True	False	Null
True	True	True	True
False	True	False	Null
Null	True	Null	Null

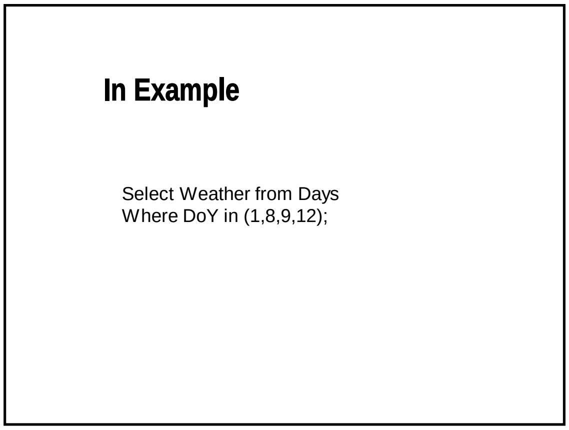
I Am Working on This
So Don’t Worry About It



Compound Conditions W/Nulls

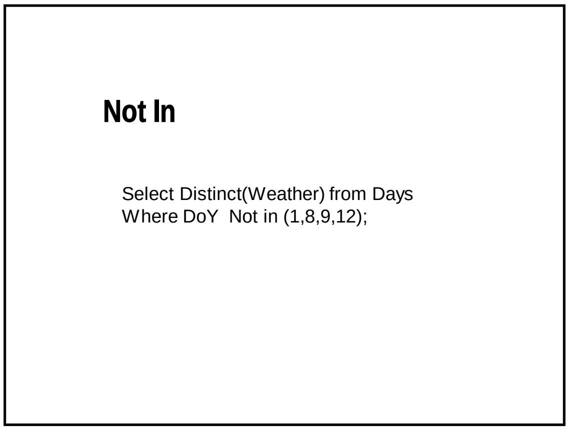
```
Select type from Soups
Where Vendor = “Campbells”
Or Vendor = “Progresso”;

Select type from Soups
Where Vendor = “Campbells”
And Mode = “Canned”;
```

I Am Working on Some Examples For
This So, Don’t Worry About it For Now



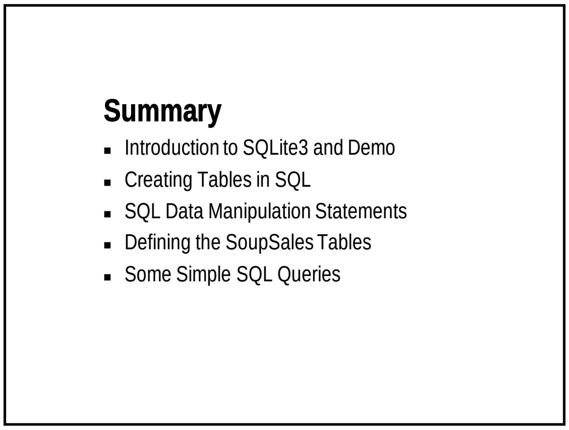
In Example

```
Select Weather from Days
Where DoY in (1,8,9,12);
```



Not In

```
Select Distinct(Weather) from Days
Where DoY  Not in (1,8,9,12);
```



Summary

- Introduction to SQLite3 and Demo
- Creating Tables in SQL
- SQL Data Manipulation Statements
- Defining the SoupSales Tables
- Some Simple SQL Queries