

My partner told me they would leave me if I don't stop making Microsoft puns, and I need some advice

I immediately left my Office and tried explaining myself.

Sure, on the Surface I do it often, but I think it Works.

It's not just about Word play, either; my Outlook on life helps me Excel.

We have such a great Team Foundation, I Azure you.

I wanted to Exchange my thoughts with them, so we could work with OneDrive.

I looked at my partner right in the Windows of their soul, to Access the deepest parts of their heart, and told them I loved them.

Completely on Edge, I awaited their answer...

PowerPoint of the story is: does anyone know of a good divorce lawyer?

Module 2 Overview

SQL

Week 1: SQL Language

Week 2: Using SQL in Java

API

Week 3: Consuming & Creating APIs in Java

Week 4: APIs continued

Module 2 API mini-capstone

Week 5 Overview

Monday

Intro to
Databases

Tuesday

Ordering,
limiting, and
grouping

Wednesday

SQL Joins

Thursday

HOLIDAY

Friday

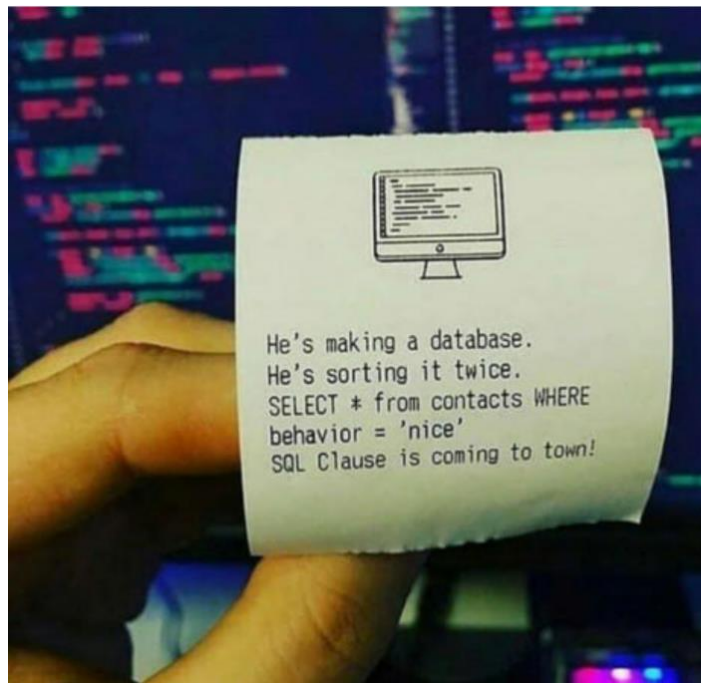
HOLIDAY

Module 2-1

Introduction to Databases and SQL

Objectives

- Understand what a database is
- Understand what SQL is
- Basic proficiency with tools specific to PostgreSQL
- Be able to write simple queries that retrieve data



Databases

- A database is an electronically stored organized collection of data.
- A **relational database** is one in which the data is organized around columns and tables:
 - A table is designed to store an **entity**, a data representation of a real world object.
 - Each row of a table represents one instance of the entity.
 - The columns represent attributes the entity might have.

Databases

- Relational (SQL Server, Oracle, MySQL, PostgreSQL)
 - NoSQL (MongoDB, CouchDB)
 - Cloud (Microsoft Azure, Amazon Relational DB Service)
 - Columnar (Google BigQuery, MariaDB, Azure SQL Data Warehouse)
 - Wide Column (BigTable, Apache Cassandra)
 - Object-oriented (Wakanda, ObjectStore)
 - Key-value (Amazon DynamoDB, Redis)
 - Hierarchical (IMS, Windows Registry)
- and more!

<https://www.matillion.com/resources/blog/the-types-of-databases-with-examples>

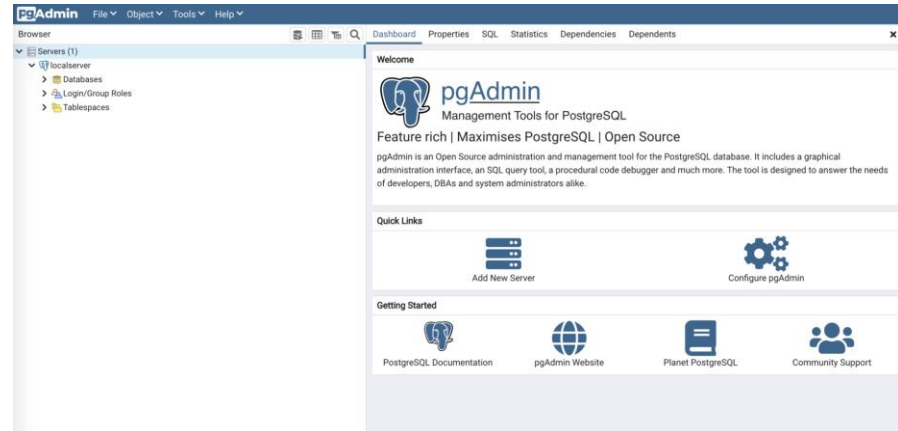
Database Management System (DBMS)



(R)DBMS

A Relational Database Management System ((R)DBMS) is a software application designed to manage a database. It has four basic functions

1. Data Definition
2. Data Storage
3. Data Retrieval
4. Administration



How to create a database in command line (specifically Bash)

```
winpty createdb -U postgres UnitedStates
```

winpty – so we can
use a bash window
instead of cmd

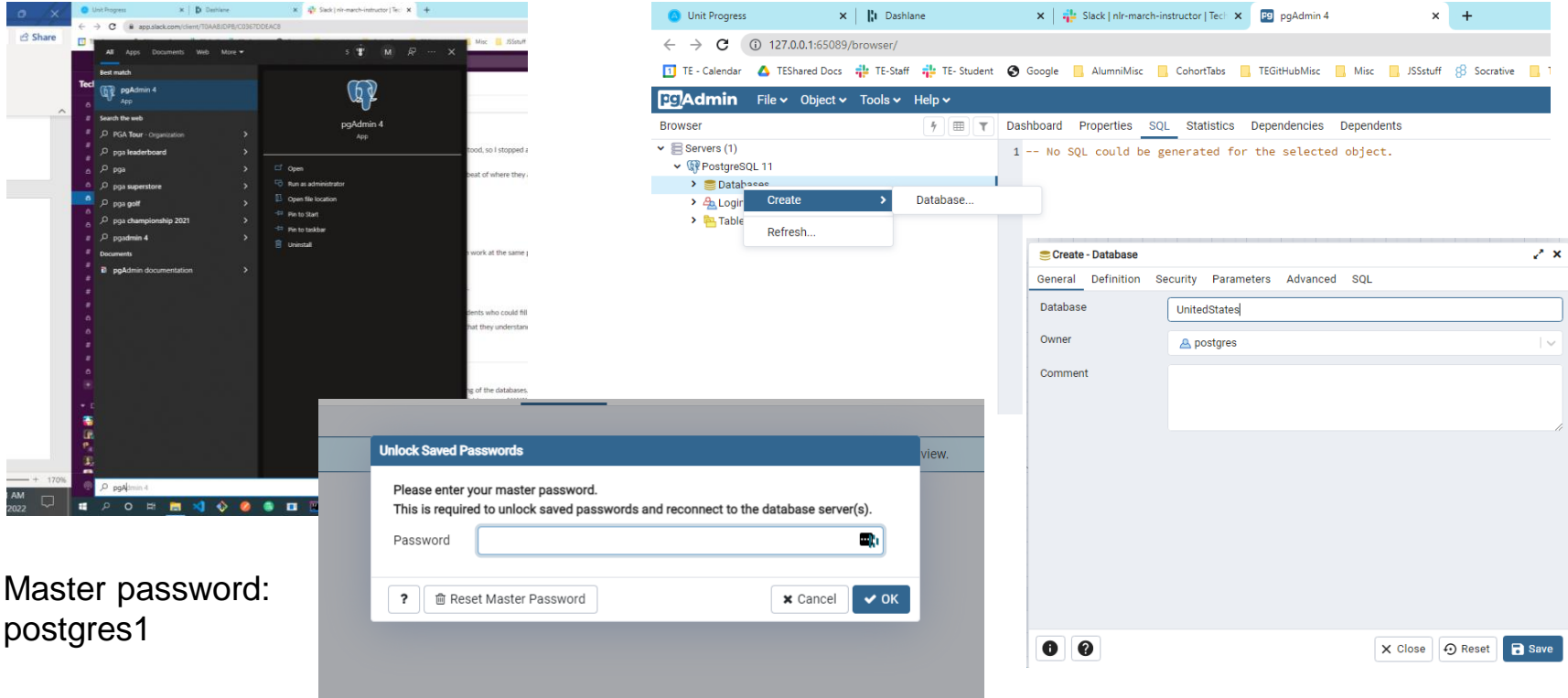
createdb – create
the db in psql
(PostgreSQL)

-U – for user

postgres

Database named
UnitedStates

Create a database in PgAdmin



Master password:
postgres1

Let's get setup!

Tables, Columns, Rows

An **entity** is a set of data being stored a *table*.

A **Table** defines a set of data elements and the structure to store them. Tables are structured into *columns* and *rows*.

Columns - attributes of a table and define the name and data type. A table has a set number of defined columns.

Rows - the data being stored. A table has an unlimited number or rows.

A **Cell** is the location where a *column* and *row* intersect, and is used to refer to a specific value or row of data (*entity*).

Column

id	name	countrycode	district	population
3793	New York	USA	New York	8008278
3794	Los Angeles	USA	California	3694820
3795	Chicago	USA	Illinois	2896016
3796	Houston	USA	Texas	1953631
3797	Philadelphia	USA	Pennsylvania	1517550
3798	Phoenix	USA	Arizona	1321045
3799	San Diego	USA	California	1223400
3800	Dallas	USA	Texas	1188580
3801	San Antonio	USA	Texas	1144646
3802	Detroit	USA	Michigan	951270

Column

- Tables have a set number.
- Define the data the table will hold
- Provides a label for each part of the data being stored

Columns on this Table

id, name, countrycode, district, population

Row

id	name	countrycode	district	population
3793	New York	USA	New York	8008278
3794	Los Angeles	USA	California	3694820
3795	Chicago	USA	Illinois	2896016
3796	Houston	USA	Texas	1953631
3797	Philadelphia	USA	Pennsylvania	1517550
3798	Phoenix	USA	Arizona	1321045
3799	San Diego	USA	California	1223400
3800	Dallas	USA	Texas	1188580
3801	San Antonio	USA	Texas	1144646
3802	Detroit	USA	Michigan	951270

Row

- Tables have a unlimited number (0...n)
- Contain the data
- Has a value for each column

Cell

id	name	countrycode	district	population
3793	New York	USA	New York	8008278
3794	Los Angeles	USA	California	3694820
3795	Chicago	USA	Illinois	2896016
3796	Houston	USA	Texas	1953631
3797	Philadelphia	USA	Pennsylvania	1517550
3798	Phoenix	USA	Arizona	1321045
3799	San Diego	USA	California	1223400
3800	Dallas	USA	Texas	1188580
3801	San Antonio	USA	Texas	1144646
3802	Detroit	USA	Michigan	951270

Cell

- The intersection of a column and row
- Used to identify a specific row of data

In this example we would identify the row we want to access by saying the ROW where the CELL in the COLUMN labelled 'name' has the value 'Chicago'

ANSI SQL

SQL - Structured Query Language : a language that lets you access and manipulate databases

ANSI-SQL - A standard that databases must follow to be considered a SQL Database.

All SQL databases support the ANSI-SQL language, however, most databases extend it with their own proprietary additions.

Character Data Types

1. **char(#)** - character. # defined the length of the data.
2. **varchar(#)** - varying character. # defined the length of the data.
3. **text** - text based data that is not limited by a predefined size

String literals in SQL use single quotes: 'Hello'

Three Strings are : **char(12)**

- PYTHON
- HTML
- SCHEMA.ORG

P	Y	T	H	O	N	blank	blank	blank	blank	blank	blank
H	T	M	L	blank	blank	blank	blank	blank	blank	blank	blank
S	C	H	E	M	A	.	O	R	G	blank	blank

Three Strings are : **varchar(12)**

- PYTHON
- HTML
- LINUX SERVER

P	Y	T	H	O	N						
H	T	M	L								
L	I	N	U	X	blank	S	E	R	V	E	R

Numeric Data Types

1. **int** or integer - similar to Java's int
2. **bigint** - Big Integer, similar to Java's long
3. **decimal(p, s)** - floating point numbers, similar to Java's double or BigDecimal
 - a. **p** - **precision** - the total number of digits being stored
 - b. **s** - **scale** - number of digits to the right of the decimal point

1234.567 has a *precision* of 7

1234.567 has a *scale* of 3

There are other numeric data types that specific to each RDMS. This is just a basic list of the ANSI types and not representative of every numeric data type.

Other Data Types

1. boolean - true/false

- a. Not the same in all SQL databases
 - i. Mysql - tinyint(1)
 - ii. PostgreSQL / Oracle - boolean
 - iii. MS Sql - bit

2. DATE

- a. yyyy-mm-dd

3. TIME

- a. hh:mm:ss

4. TIMESTAMP / DATETIME

- a. yyyy-mm-dd hh:mm:ss

Date and Time datatypes also support the usage of 24-hour vs 12-hour clocks and Time Zones.

<https://www.postgresql.org/docs/9.3/datatype.html>

Relational Database: SQL

- SQL is an acronym for Structured Query Language
- SQL is a declarative programming language (what actions should be performed rather than how to perform those actions)
- SQL is the language used to interact with relational database management systems.
- The exact implementation of SQL varies slightly depending on the database system involved, i.e. there will be minor differences in the language between PostgreSQL and MS SQL Server.
- This class will be using PostgreSQL.

3 types of commands

- DML

- Database Manipulation Language
 - INSERT, SELECT, DELETE, etc.

- DDL

- Data Definition Language
 - Commands for creating tables, defining relationships, etc.

- DCL

- Data Control Language
 - Commands that control permissions on the data and access rights

Postgres!

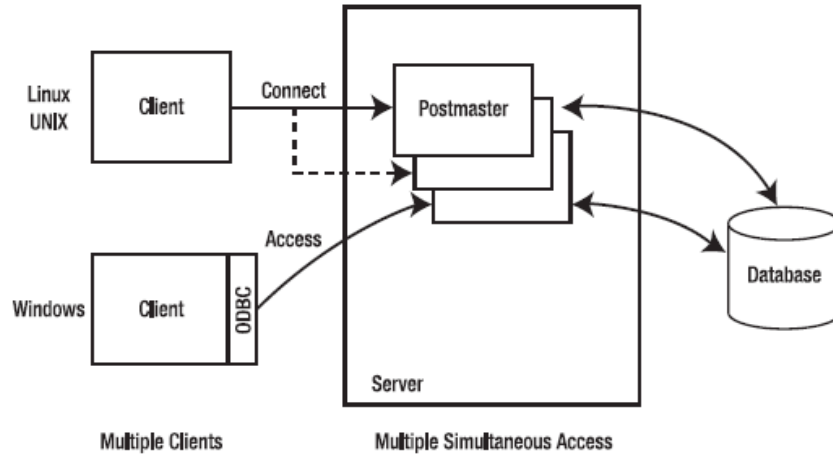


Figure 1-3. PostgreSQL architecture

SQL: SELECT

- The most basic SQL statement is a SELECT query, and it follows the following format:

SELECT **[column]**, **[column-n]** FROM **[table]**;

- **[column]** and **[column-n]** are stand ins for the attributes or columns that you want returned from your query.
- **[table]** refers to the name of the table you are querying.
- You can create column Aliases using the “**AS**” keyword followed by the alias.

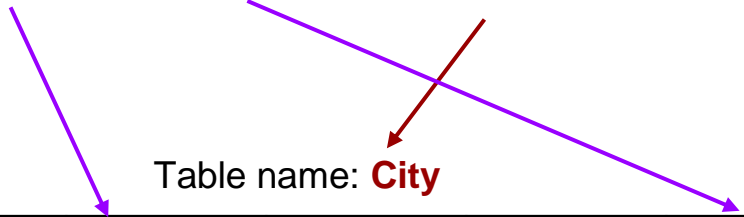
SELECT

The **SELECT** clause *indicates what columns* to return in the results of the query

The **FROM** clause *indicates which table(s)* to retrieve the data from.

```
SELECT name, population FROM city;
```

Table name: **City**



id	name	countrycode	district	population
3793	New York	USA	New York	8008278
3794	Los Angeles	USA	California	3694820
3795	Chicago	USA	Illinois	2896016
3796	Houston	USA	Texas	1953631
3797	Philadelphia	USA	Pennsylvania	1517550
3801	San Antonio	USA	Texas	1144646
3802	Detroit	USA	Michigan	951270

SELECT Modifiers

***** can be used to select all columns from a table.

```
SELECT * FROM country;
```

AS can be used with a column name to give it an **Alias** (new name)

```
SELECT name AS 'CityName' FROM city;
```

Or to give a name to a combined result:

```
SELECT ( col1 + col2 ) AS 'Sum'
```

DISTINCT can be used with a column name to return only unique values from that column.

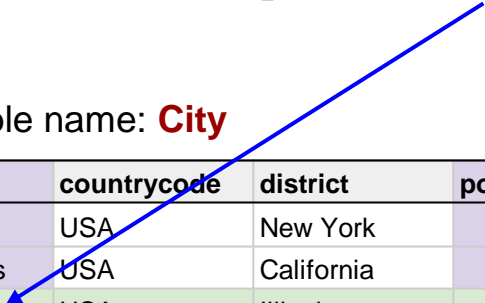
```
SELECT DISTINCT name FROM city;
```

WHERE

The **WHERE** clause is used to filter the rows returned in the results using a boolean condition. Rows that match that the expressions evaluates true for are returned by the query.

```
SELECT name, population FROM city WHERE name = 'Chicago' ;
```

Table name: **City**



id	name	countrycode	district	population
3793	New York	USA	New York	8008278
3794	Los Angeles	USA	California	3694820
3795	Chicago	USA	Illinois	2896016
3796	Houston	USA	Texas	1953631
3797	Philadelphia	USA	Pennsylvania	1517550
3801	San Antonio	USA	Texas	1144646
3802	Detroit	USA	Michigan	951270

WHERE Clause Conditionals

=	equal to
!=, <>	Not equal to
>, <, >=, <=	Greater/Less Than
IS NULL	value is null
IS NOT NULL	value is not null
IN (val1, val2, ...)	Value is IN the list
NOT IN (val1, val2, ...)	Value is NOT IN the list
BETWEEN va1 AND val2	The value is between the 2 values Example: number BETWEEN 2 AND 10;
LIKE (%)	The value matches a pattern created with % Example: a% - the value starts with a %a - the value ends with a %a% - the value contains an a

Conditionals can be chained together using **AND** and **OR**.

Precedent can be set using **()**

Example:

WHERE num > 5 AND
(name LIKE 'A%' OR name LIKE 'B%')

Strings in SQL use *single quotes*.

name = 'John'

SQL: SELECT with WHERE clause

- We can include a WHERE clause in our select statements to limit the data returned by specifying a condition.
- The WHERE statement relies on comparison operators.
 - Greater Than: >
 - Greater Than or Equal To: >=
 - Less Than: <
 - Less Than or Equal To: <=
 - Equal: =
 - Not Equal To: <> !=
- There is a special comparison operator called **LIKE** which is often used in conjunction with a wildcard (%) operator.

SQL: SELECT with WHERE clause Example 1

Let's take the Vehicle table we just saw as an example:

- We could write the following SELECT statement:

SELECT * FROM Vehicle WHERE Manufacturer = 'Ford';

- Only 1 row matches this criteria, and thus the results of the query will be:

CarName	Manufacturer	NumberOfDoors	FuelEconomy
Explorer	Ford	4	23

SQL: SELECT with WHERE clause Example 2

Here is an example of the WHERE clause using the LIKE / Wildcard.

- We could write the following SELECT statement:

SELECT * FROM Vehicle WHERE CarName LIKE 'Ex%';

- Only 1 row matches this criteria, and thus the results of the query will be:

CarName	Manufacturer	NumberOfDoors	FuelEconomy
Explorer	Ford	4	23

Derived Columns with Math Operations

- A custom field containing math operations can be included in the SELECT.
- The basic math operators are present: **+**, **-**, *****, **/**, **%**

```
SELECT employee_id, employee_name, salary, salary + 100  
       AS "salary + 100" FROM addition;
```


Derived Columns Example

- Consider the following example:

SELECT CarName, FuelEconomy * 0.425144 AS kpl FROM Vehicle;

CarName	kpl
Explorer	9.778312
C-Class	9.778312
Jeep Wrangler	8.50288

SQL: AND / OR on WHERE statements

- Within the WHERE statement, various filter conditions can be combined using the AND / OR statement.
- Consider the following example:

SELECT * FROM Vehicle WHERE Manufacturer = 'Ford' OR NumberOfDoors = 4;

- Two rows are returned:

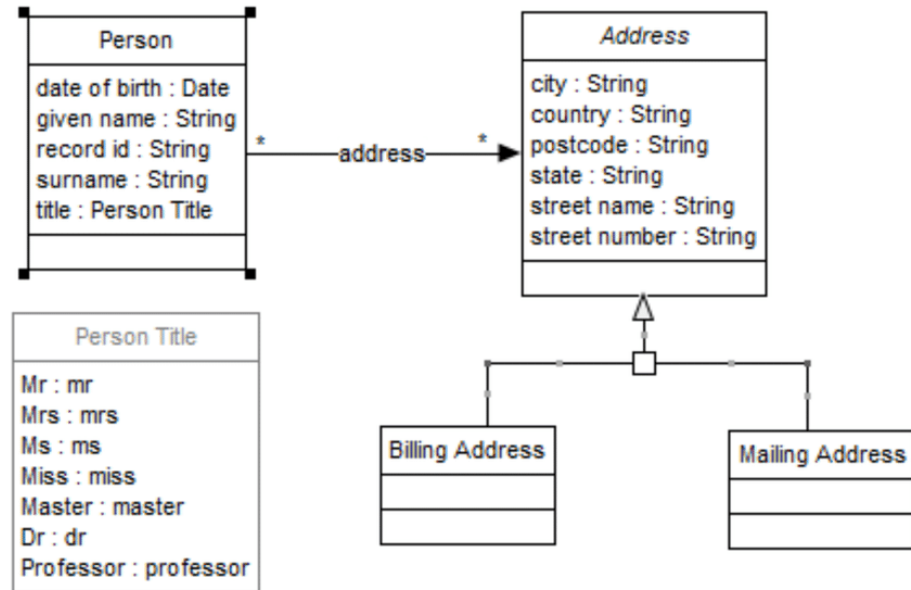
CarName	Manufacturer	NumberOfDoors	FuelEconomy
Explorer	Ford	4	23
C-Class	Mercedes Benz	4	28

Let's write some `SELECT` statements!

But first...

Objectives

- Understand what a database is



Objectives

- Understand what a database is
- Understand what SQL is

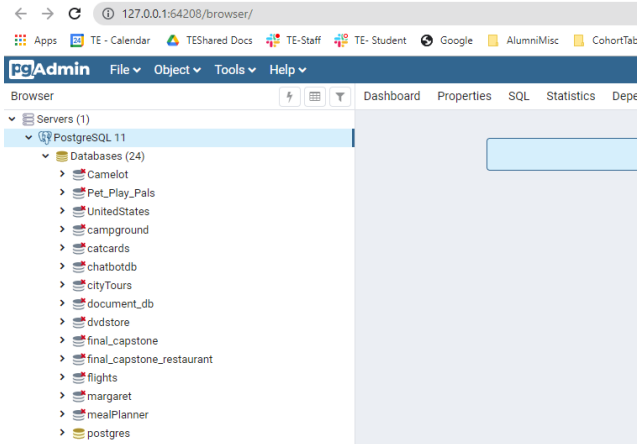
```
postgres=# Select * from company;
 company_id | name   | address | phone           | country | website_url
-----+-----+-----+-----+-----+-----
          1 | Samsung | 123.... | +337277888      | Korea   | www.samsung.com
          2 | Symphony | 67/A ... | +42343567       | China   | www.symphony.com
          3 | LG      | 45/B ... |                  | Japan   | www.lg.com
(3 rows)

postgres=# select * from items;
 item_id | name       | quantity | company_id
-----+-----+-----+-----
        4 | LG 122     |      4000 |          3
        5 | Samsung 460 |      7000 |          1
        6 | Symphony E80 |      2200 |          2
(3 rows)

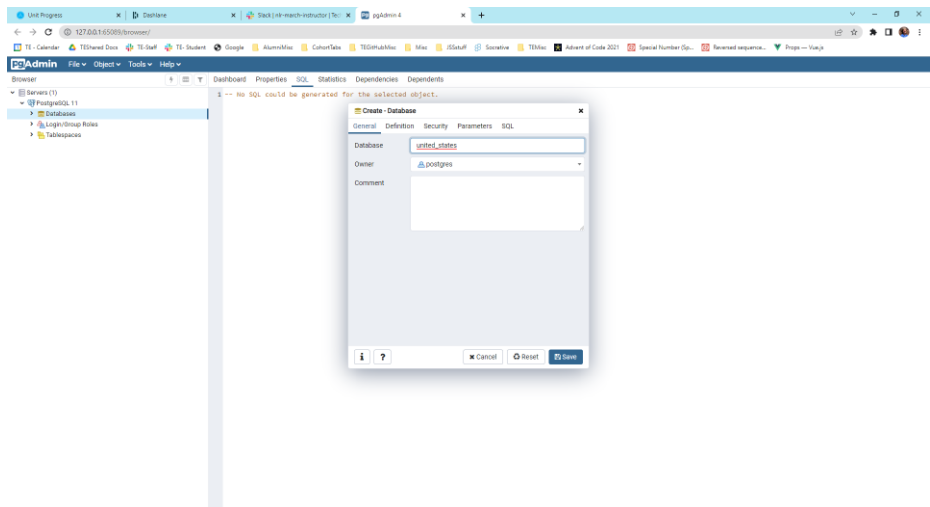
postgres=# select * from customers;
 customer_id | name   | address | phone           | company_id
-----+-----+-----+-----+-----
          4 | Micheal | 23/C... | +9343422343     |          1
          5 | Watson  | 88...   | +23434345       |          1
          6 | Gilmore | 123/C... | +63423233       |          2
(3 rows)
```

Objectives

- Understand what a database is
- Understand what SQL is
- Basic proficiency with tools specific to PostgreSQL



```
/usr/bin/bash --login -i
~/workspace/NLR-3/java-orange-main/module-2/01_Introduction_to_Databases/lecture-
15:25 $ winpty createdb -U postgres UnitedStates
Password:
~/workspace/NLR-3/java-orange-main/module-2/01_Introduction_to_Databases/lecture-
15:26 $ ls
Census_Regions_of_US.png  UnitedStates_ERD.png  UnitedStates-data.psql
~/workspace/NLR-3/java-orange-main/module-2/01_Introduction_to_Databases/lecture-
15:27 $ psql -U postgres -d UnitedStates -f UnitedStates-data.psql |
```



Objectives

- Understand what a database is
- Understand what SQL is
- Basic proficiency with tools specific to PostgreSQL
- Be able to write simple queries that retrieve data

