9.1 Lesson Summary - Introduction to SQL

We have seen a number of different ways that data is stored. CSV and Excel files are easy to read and edit but they can't handle large amounts of data nor are they very well suited to describing relationships between tables of data. Relational databases are one of the preferred methods for dealing with large amounts of data that may have complex relationships between tables. Relational databases operate by having a program manage how your data is written onto a hard drive. Structured Query Language (SQL) is used for creating and interacting with these databases.

Concept: **PostgreSQL** is one of the most popular applications for managing relational databases. It allows you to setup and access databases on your laptop or on larger production servers. **pgAdmin** is an application with an intuitive GUI that allows you to create, view, and edit your PostgreSQL databases. To create a database in pdAdmin select the *Object* dropdown menu and then choose *Create* -> *Database*. Then specify the name and owner of your database. Or you can right click on your PostgreSQL server in the browser toolbar and choose the *Create* -> *Database* option.

* Activity: 01-Evr\_Creating\_a\_Database
* Suppl link: <https://www.postgresql.org/>

Concept: Data is stored in **tables** in relational databases. Tables in relational databases are similar to worksheets in Excel and DataFrames in Pandas. Before you can store data in a relational database table you must specify which **fields** you will have in that table. Fields are analogous to columns in an Excel worksheet or column Series in a Pandas DataFrame. In order to create a field in a relational database's table you must specify the kind of data that field will contain. Different kinds of data include BOOLEAN, VARCHAR(strings), INT (integers), and more. You can also specify default parameters and behaviors for table fields. To create a table describing people, including their names, age, and if they like ice cream you could use the following code:

*CREATE TABLE people (*

*name VARCHAR(30) NOT NULL,*

*age INT,*

*likes\_ice\_cream BOOLEAN DEFAULT true*

*);*

You can also delete a table using the following SQL query:

*DROP TABLE people;*

* Activity: 02-Ins\_Creating\_Tables, 03-Stu\_Creating\_Tables
* Suppl link: <https://www.w3schools.com/sql/sql_datatypes.asp>

Concept: To **retrieve** data from your relational database table you can use the following code:

*SELECT name, age*

*FROM people;*

You can also retrieve specific kinds of data using a *WHERE* clause with Boolean operators. For example, if we wanted to retrieve only the name and age of people that were over 35 and didn't like ice cream from our people table, we could use the following SQL query:

*SELECT name, age*

*FROM people*

*WHERE likes\_ice\_cream = false*

*AND age > 35;*

* Activity: 02-Ins\_Creating\_Tables, 03-Stu\_Creating\_Tables
* Suppl link: <https://www.w3schools.com/sql/sql_where.asp>, <https://www.w3schools.com/sql/sql_and_or.asp>

Concept: To **add** data to a relational database table you can use the following code:

*INSERT INTO people (name, age, likes\_ice\_cream)*

*VALUES ('Jacob', 42, true),*

*('Susan', 12, true),*

*('Dave', 54, true);*

* Activity: 04-Ins\_Values\_of\_Uniques

Concept: It is often useful to have **unique identifiers** for the rows in your table. In our ice cream example if we had two unique instances of a person named 'Susan' who was 48 and liked ice cream it would be important to be able to distinguish between these 2 women. This can be accomplished by giving each row a unique ID field. You can manually add data to this field, just like any other, but it's often simpler to have PostgreSQL automatically add this ID for you by indicating that the field is SERIAL and a PRIMARY KEY. For example:

*CREATE TABLE people (*

*id SERIAL PRIMARY KEY,*

*name VARCHAR(30) NOT NULL,*

*age INT,*

*likes\_ice\_cream BOOLEAN DEFAULT true*

*);*

* Activity: 04-Ins\_Values\_of\_Uniques, 05-Stu\_Making\_IDs

Concept: Data is often exchanged using the **csv** format and it can be **imported** directly into a relational database table. To accomplish this in PostgreSQL open pdAdmin. Right click your table in the Browser pane and select the *Import* option. Choose your CSV file and select the necessary options.

* Activity: 06-Ins\_Importing\_Data, 07-Stu\_Hide\_and\_Seek

Concept: The functionality provided by data storage tools, including PostgreSQL is often described using the **CRUD** acronym for **Create**, **Read**, **Update**, and **Delete**. Most data storage tools offer some form of these functionalities. In SQL Create is usually accomplished using an INSERT statement. Read is often accomplished using a SELECT statement. Updating existing records is accomplished using an UPDATE statement and Deleting data is accomplished using a DELETE statement. If we wanted to update the age of the person with ID 19 in our *people* table, we could use the following query:

*UPDATE people*

*SET age = 49*

*WHERE ID = 19;*

* Activity: 08-Stu\_CRUD

Concept: It is often necessary to combine data from multiple different SQL tables together to analyze the data that's most important. SQL provides a number of different techniques to **JOIN** different tables together. An INNER JOIN will only return records that match the join criteria. A LEFT JOIN will include all of the records in the left table and only the matching records in the right table. A RIGHT JOIN will include all of the records in the right table and only the matching records in the left table. A FULL OUTER JOIN will return all records from both tables, combining the rows that match the specified criteria and filling in unmatched rows with null values. An INNER JOIN of a *players* and a *matches* table could look like the following query:

*SELECT players.first\_name, players.last\_name, players.hand, matches.loser\_rank*

*FROM matches*

*INNER JOIN players ON*

*players.player\_id=matches.loser\_id;*

* Activity: 09-Ins\_Joins, 10-Stu\_Joins
* Suppl link: <https://www.w3schools.com/sql/sql_join.asp>