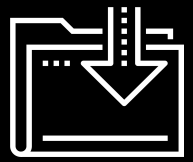
# Introduction to SQL

Data Boot Camp

Lesson 9.1



# Learning Outcomes

By the end of this unit, you will be able to:

01

Create a data model to represent the objects and relationships in a dataset.

02

Create schemas, tables, and databases for relational data.

03

Retrieve data by using advanced database queries.

# Class Objectives

By the end of today's class, you will be able to:

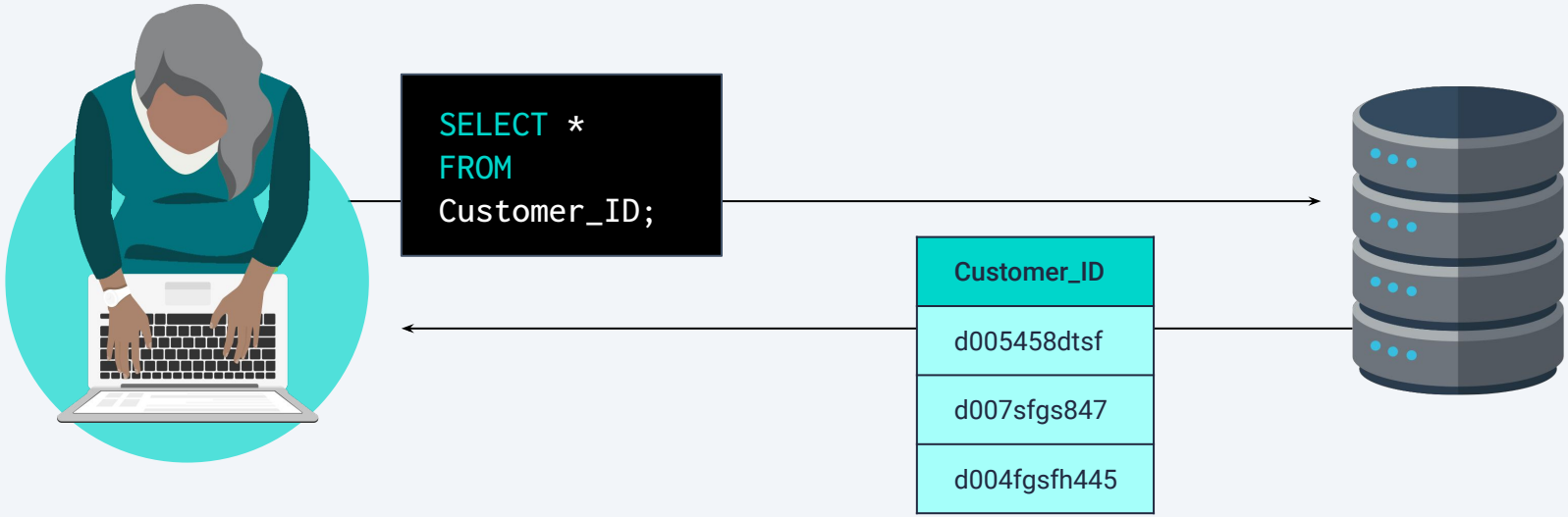
- Install and run Postgres and pgAdmin on your computer.
- Create a database and tables using pgAdmin.
- Define SQL data types, primary keys, and unique values.
- Load CSV files into a database and query the data.
- Articulate the four basic functions of persistent storage (CRUD) and apply this set of functions to a database.
- Combine data from multiple tables using JOINs.

# Introduction to SQL

# Introduction to SQL

SQL (often pronounced "sequel") stands for Structured Query Language.

It is a powerful tool that enables programmers to create, populate, manipulate, and access databases. It also provides an easy method for dealing with server-side storage.



# Introduction to SQL

Data using SQL is stored in tables on the server, much like spreadsheets you would create in Microsoft Excel.

This makes the data easy to visualize and search.

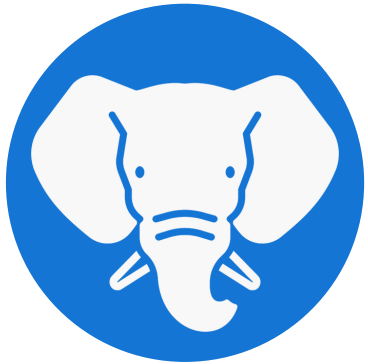
| Customer_ID | Date_ID |
|---|---|
| d005458dtsf | 6/26/2019 |
| d007sfgs847 | 8/3/2018 |
| d004fgsfh445 | 12/3/2018 |

| Order_ID | Customer_ID | Date_ID |
|---|---|---|
| 10001 | d005458dtsf | 6/26/2019 |
| 10002 | d007sfgs847 | 8/3/2018 |
| 10003 | d004fgsfh445 | 12/3/2018 |

# Introduction to SQL

## PostgreSQL

PostgreSQL (usually referred to as "Postgres") is an object-relational database system that uses the SQL language.



## pgAdmin

pgAdmin is the management tool used for working with Postgres.
It simplifies creation, maintenance, and use of database objects.

# Time to Code

## Create a Database

Suggested Time:

5 minutes

# Create a Database

| Instructions | |
|---|---|
| | In the pgAdmin editor, right-click the newly established server to create a new database. |
| | From the menu, select **Create**, and then select **Database** to create a new database. |
| | Enter `animals_db` as the database name. |
| | Make sure the owner is set as **postgres** (default setting), and then click **Save**. |

# Questions?

# Instructor Demonstration

## Create a Table

# Create a Table

| | |
|---|---|
| `CREATE TABLE people (<COLUMNS>);c` | Creates a table called people with the columns listed within the parentheses. |
| `name VARCHAR(30) NOT NULL` | creates a `name` column that holds character strings of up to 30 characters and will not allow null fields. |
| `NOT NULL` | Requires the name field to have a value specified. |
| `pet_type VARCHAR(10) NOT NULL` | Creates a `pet_type` in the same manner as the name column is created. The only difference is the number of characters allowed in the column. |
| `has_pet BOOLEAN DEFAULT false` | Creates a `has_pet` column that holds either true or false values. Here the default value is set as false. |
| `pet_name VARCHAR(30)` | creates a `pet_name` column that holds character strings of up to 30 characters and will allow null fields. |
| `pet_age INT` | Creates a `pet_age` column that hold whole numbers. |

# Create a Table

The `SELECT` clause can specify more than one column.

```sql
SELECT pet_type, pet_name
FROM people
WHERE pet_type = 'dog'
AND pet_age < 5;
```

# Create a Table

Data is filtered by using additional clauses such as `WHERE` and `AND`.

```sql
SELECT pet_type, pet_name
FROM people
WHERE pet_type = 'dog'
 AND pet_age < 5;
```

# Create a Table

The `WHERE` clause will extract only the data that meets the condition specified. `AND` adds a second condition to the original clause, further refining the query.

```sql
SELECT pet_type, pet_name
FROM people
WHERE pet_type = 'dog'
AND pet_age < 5;
```

# Create a Table

Note that unlike in Python where comparisons are done with a double equals (`==`) sign, in SQL only a single equal sign is used.

```sql
SELECT pet_type, pet_name
FROM people
WHERE pet_type = 'dog'
AND pet_age < 5;
```

# Activity: Creating Tables

In this activity, you will use pgAdmin to recreate and query a table from an image provided.

Suggested Time:

15 minutes

# Time’s Up! Let’s Review.

# Instructor Demonstration

## The Value of Unique Values

# Questions?

# Activity: Making and Using an ID

In this activity, you will recreate a table and then query, insert, and update data.

Suggested Time:

10 minutes

# Time’s Up! Let’s Review.


Break

# Instructor Demonstration

## Import Data

# Activity: Hide and Seek

In this activity, you will create a new table and import data from a CSV file.

Suggested Time:

10 minutes

# Activity: Hide and Seek

| | |
|---|---|
| **Instructions** | Create a new table in the `Miscellaneous_DB` database called `movie_words_comparison`. |
| | Import the data from the `soft-attributes.csv` file in the Resources folder. |
| | Create a query in which the data in the `reference_title` column is `Home Alone (1990)`. |
| | Create a query that collects all rows in which the rater is within the range 10–15. |
| | Create a query that searches for any rows that have `artsy` or `heartfelt` in the `soft-attribute` column. |
| **Bonus** | Create a query that will collect all rows where the reference title is `Batman (1989)` and the soft attribute is `scary`. |
| | Create a query that will collect all rows where the reference title is `Home Alone (1990)` and the soft attribute is `artsy` with rater range between 30 and 40. |

Note: Data provided by Google Research Datasets.

# Time’s Up! Let’s Review.

# Using CRUD

# Using CRUD

CRUD represents a set of tools that are used throughout programming:

| | |
|---|---|
| **Create** | Create data in a table with the `INSERT` statement. |
| **Read** | Read data by using `SELECT`. |
| **Update** | Updated a table's data by using `UPDATE`. |
| **Delete** | Deleted data via `DELETE`. |

These tools are fundamental to all programming languages, not just SQL.

# Wildcard: % and _

Use wildcards to substitute zero, one, or multiple characters in a string.
The keyword `LIKE` indicates the use of a wildcard.

```
SELECT *
FROM actor
WHERE last_name LIKE 'Will%';
```

# Wildcard: % and _

The % will substitute zero, one, or multiple characters in a query.
In this example, all of the following are matches: Will, Willa, and Willows.

```
SELECT *
FROM actor
WHERE last_name LIKE 'Will%';
```

# Wildcard: % and _

The `_` will substitute only **one** character in a query.

`_an` returns all actors whose first name contains three letters,
the second and third of which are `an`.

```
SELECT *
FROM actor
WHERE first_name LIKE '_an';
```

# Activity: Using CRUD

In this activity, you will use CRUD operations (Create, Read, Update, Delete) on the provided data.

Suggested Time:

20 minutes

# Activity: Using CRUD

| | |
|---|---|
| **Instructions** | Create a new database named `Malaysia` in pgAdmin. |
| | Create two new tables called road_accidents and `accidents_by_state` in the `Malaysia` database by copying the code provided in `schema.sql` into a new query window in pgAdmin. Import the data from `mys_road_accidents.csv` and `mys_accidents_by_state.csv` by using the Import/Export tool. |
| | In the road_accidents table, find the row with missing data, and make notes on what needs to be updated. |
| | In the `accidents_by_state` table, delete all the rows for years that do not have missing data from `road_accidents`. |
| | In the `accidents_by_state` table, find the **Sum** for the columns with missing data in `road_accidents`, and rename the columns with their column names. |
| | Update the `road_accidents` table with the new information. |
| **Bonus** | Delete all rows from `accidents_by_state` and re-import `mys_accidents_by_state.csv`. |
| | Without deleting any rows, calculate the sum of `road_crashes`, `road_deaths`, `serious_injury`, and `slight_injury` for a subsequent year, and add those values plus the year to the `road_accidents` table. |

Note: Data provided by Google Research Datasets.

# Time’s Up! Let’s Review.

# Five Primary Types of Joins used with PostgreSQL

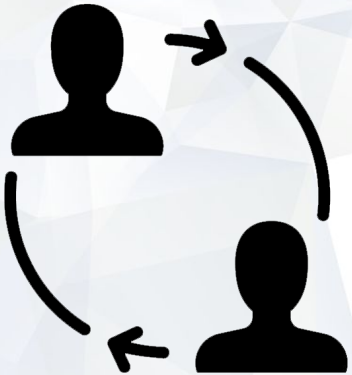
# Five Primary Types of Joins used with PostgreSQL

| | |
|---|---|
| `INNER JOIN` | Returns records that have matching values in both tables. |
| `LEFT JOIN` | Returns all records from the left table and the matched records from the right table. |
| `RIGHT JOIN` | Returns all records from the right table and the matched records from the left table. |
| `CROSS JOIN` | Returns records that match every row of the left table with every row of the right table. This type of join has the potential to make very large tables. |
| `FULL OUTER JOIN` | Places null values within the columns that do not match between the two tables, after an inner join is performed. |

# Instructor Demonstration

## Joins

# Questions?



# Partner Activity: Joining Bird Bands

In this activity, the you will use joins to learn more about North American bird banding.

Suggested Time:

20 minutes

# Activity: Using CRUD

**Instructions**

Create a new database named `bird_banding_DB` and create eight new tables with pgAdmin named `bird_bands`, age, `band_type`, `bird_status`, `country_state`, `event_type`, `extra_info`, and `sex`.

Copy the code from `schema.sql` to create the tables, and then import the corresponding data from `Players.csv` and `Seasons_Stats.csv`.

**Hint**

Remember to refresh the database because newly created tables will not immediately appear.

Note: Data provided by Google Research Datasets.

# Time’s Up! Let’s Review.