**CHAPTER 1**

**Creating Database and Table**

SQL allows to define the structures that hold data, to organize relationships in the data.

A table is a grid of rows and columns that store data. Each row holds a collection of columns, and each column contains data of a specified type.

**Creating a Database**

When installing PostgreSQL, it created a database server, an instance of the application running on the computer.

A database is a collection of objects that includes tables, functions, user roles and much more.

It is generally good practice to create a new database for each project.

**Creating a Table**

When creating a table, you assign a name to each column and assign a data type.

The datatype **bigserial** is a special integer type that auto increments everytime you add a row to the table. The first row receives the value of 1 and the second row 2 and so on.

(<https://www.convert-in.com/mysql-to-postgres-types-mapping.htm>)

**INSERT data into the table**

First row, we specify the “INSERT INTO table\_name (column\_names to insert to)”.

The next row, we give the VALUES (‘val\_1’,’val\_2’,… 124).

The order of your values must also match the order of the columns specified after the table name.

Text and date entries require quotes, but numbers including integers and decimals don’t require quotes.

**CHAPTER 2**

In SELECT \*, the asterisk is a wildcard. A wildcard is like a stand-in for a value: it doesn’t represent anything in particular and instead represents everything that value could possibly be. It is short hand for “select all columns”.

Instead of the asterisk, we can also give a column name and obtain that column from the table. This allows us to query a subset of the table.

**Using DISTINCT to Find Unique Values**

To understand the range of values in a column we use the DISTINCT keyword

We can also use DISTINCT with 2 columns of a table, however the return will be the unique combinations. “For each x in table 1, what are the corresponding y values in table 2”

**Using ORDER BY to sort data**

Data can make more sense and may reveal patterns more readily when its arranged in order rather than jumbled randomly.

SELECT col\_name1,col\_name2,col\_name3

FROM teachers

ORDER BY salary DESC

**Note that you don’t always need the column you are ordering by, in your SELECT statement.**

**Also we can use ORDER BY with multiple columns**

SELECT first\_name,last\_name,school,hire\_date

FROM teachers

ORDER BY school ASC, hire\_date DESC;

In the above, we first order by school in ascending order, and then within each school, we order by the hire\_date, showing the hire\_date in descending order.

**Filtering rows with WHERE**

Sometimes we want to limit the rows a query returns to only those in which one or more columns meet certain criteria.

Syntax:

SELECT col\_name1,col\_name2

FROM Table\_1

WHERE col\_name1>3000 ( can use different operators)

**Using LIKE and ILIKE with WHERE**

In SQL there are two special characters (Percent sign (%)) and Underscore(\_). Percent sign will match one or more characters, \_ will match only one character.

LIKE is case sensitive, ILIKE is not.

**Using AND , OR operators**

Syntax:

SELECT col\_name1, col\_name2

FROM Table\_1

WHERE col\_name3 > x AND/OR col\_name4 = ‘abc’

Can also use as follows:

SELECT col\_name1, col\_name2

FROM Table\_1

WHERE col\_name3 > x AND/OR (col\_name4 = ‘abc’ AND col\_name5 <4)

**Usual Syntax:**

SELECT column\_names

FROM table\_name

WHERE criteria

ORDER BY column\_names