Creating and altering database tables

This is an ungraded lab. Sections in green will be demonstrated by the instructor but are not required activities for students. All other sections should be completed either in pairs or individually.

Note: To get the documentation on any Postgres command, the easiest way is to use a web search restricted to the postgresql.org domain. For example, “select site:postgresql.org” or “group by site:postgresql.org” – or even better, restrict to “site:postgresql.org/docs/current”. Another good way to understand a Postgres command is to use the graphical interface in DBeaver or Supabase to perform an operation. There is usually some method of viewing the corresponding SQL, and this can give you useful examples for understanding the SQL. You already did this to understand the UPDATE command in assignment CC1, question B4. Below you will see other important examples, such as how to use ALTER TABLE.

# Part 1: Using Postgres from the command line

[Demonstration only; not required for students to do this.] Start psql; demonstrate that we can use Postgres from the command line (on Windows I use Cygwin terminal for this, also need cmd.exe /c chcp 1252 to set the correct code page):

psql -U postgres

Demonstrate creating and using a schema:

\dn

create schema myschema2;

\dn

drop schema myschema2;

\dn

Demonstrate basic SQL functionality:

SET search\_path TO wine;

SELECT \* FROM product WHERE PRODTYPE='red' AND AVAILABLE\_QUANTITY>120;

Show the distinct processes running Postgres Server and the psql client. (On Windows, we can see the process tree using Process Explorer – Make sure to run as administrator, observe all the distinct process roles of the Postgres processes by looking at the command line used to launch each one.)

# Part 2: Creating and altering tables in DBeaver

[Student activity starts here]

Create schema carlisle\_menu\_items using DBeaver. Create a ‘restaurant’ table with appropriate columns for the restaurant:

Table ‘**restaurant’**

|  |  |  |
| --- | --- | --- |
| **id** | **Name** | **Address** |
| 23 | Issei Carlisle | 54 W High St, Carlisle, PA 17013 |
| 49 | Mt Fuji | 149 N Hanover St, Carlisle, PA 17013 |

What data types did you use for the columns? Experiment with changing them. So far, your table has just columns with no data in them.

[Instructor only] Use the SQL produced here (via Generate SQL | DDL) to paste into terminal for a different schema e.g. temp\_menu\_items

Add a row of data to the restaurant table using DBeaver (use the Add Row button near the bottom).

View the corresponding SQL: right-click, Export Data, SQL...

Now delete the table completely (right click, Delete – do View Script to understand the SQL).

Next we practice creating the table from a file instead.

Create a new Excel file, copy and paste all table data including the column names into the spreadsheet. Save as CSV. View the result in a text editor to make sure you understand the formatting. In DBeaver, make sure you are browsing the carlisle\_menu\_items schema then right click Tables, choose Import Data. Select the saved CSV file, check that all the options look sensible (the name of the table should be restaurant), preview the data, click through remaining options. Take a look at the resulting table. Look at its SQL too. Any differences to earlier?

Check the data types of the columns. Practice changing a data type: view the table, then Properties tab, Columns tab, click on data type to change it. Check out the equivalent SQL – It will be something like:

ALTER TABLE carlisle\_menu\_items.restaurants ALTER COLUMN id TYPE int8 USING id::int8;

Note also that we will not be able to edit the content of the table until we have created a primary key. This is because DBeaver needs a way to refer to specific tuples and there is no way to do that without a key. Before we add a primary key, try to edit and save one of the cells. Observe the error message then cancel.

Now add the restaurant id as a primary key in DBeaver: view the table, then Properties tab, Constraints tab, Add New Constraint (small icon at the bottom). Choose Primary Key type, and check the id column. When you save the changes observe the SQL which should be something like the following:

ALTER TABLE carlisle\_menu\_items.restaurants ADD CONSTRAINT restaurants\_pk PRIMARY KEY (id);

Now you should be able to edit the table values. Try to change an address or an ID for a restaurant and save the changes. Observe the equivalent SQL and observe the saved results.

We’re finished working on this table. Observe the final structure of the table by checking out the equivalent SQL. It should be similar to

CREATE TABLE carlisle\_menu\_items.restaurants (

 id int8 NOT NULL,

 "Name" text NULL,

 "Address" varchar(50) NULL,

 CONSTRAINT restaurants\_pk PRIMARY KEY (id)

);

Now add the data from the menu items into a new table called item. Use the CSV file method; don’t type all the data in manually. Here is the data to be inserted:

Table ‘**item’**:

|  |  |  |
| --- | --- | --- |
| id | name | restaurant\_id |
| 6 | miso ramen | 23 |
| 18 | ebi tempura | 49 |
| 8 | tantan ramen | 23 |
| 18 | pad thai | 23 |
| 18V | vegetable pad thai | 23 |
| D4 | mabo tofu | 23 |
| 23 | temaki dinner | 49 |

Make appropriate changes to the data types, and add a primary key. (For the primary key, if you get an error message about a duplicate value, think about what column or columns you can select to obtain a unique value for every tuple.)

Now we need to link the two tables using a foreign key. Specifically, we want the restaurant\_id column in the item table to reference the id column in the restaurant table. Try to figure out how to do this on your own. If it’s not working within a few minutes, get some help from AI assistant or the instructor. Check the SQL before saving and make sure that you are linking the **restaurant\_id** column in the item table to the id column in restaurant; the SQL should resemble

ALTER TABLE carlisle\_menu\_items.items ADD CONSTRAINT items\_restaurants\_fk FOREIGN KEY (restaurant\_id) REFERENCES carlisle\_menu\_items.restaurants(id);

Now try to change a restaurant ID in the restaurant table and/or items table. What happens?

Delete the foreign key constraint and add it again, this time specifying Cascade for updates (there is a checkbox for this). Now are you able to change a restaurant ID in the restaurant table and/or items table?

Suppose we wanted restaurant names to be unique. Find a way to do this. What is the equivalent SQL?

Finally, notice that we can provide default values for columns. As an example of this, change the name column in the item table to have the default value 'unknown' (you will probably need the single quotes). The equivalent SQL is something like:

ALTER TABLE carlisle\_menu\_items.items ALTER COLUMN "name" SET DEFAULT 'unknown';

To see the effect of this, browse the item table and create a new item.