**Online Food Ordering Database User Guide**

Steven On

CPT 310: Database Systems & Management

Professor Lisa Sims

March 1, 2021

**User Guide**

The intent of this user guide is to create an online food ordering database from table creation to populating data. Along with the “how to” guide of implementing the MySQL database, this paper describes the SQL functionalities and overall tasks of why specific actions are performed for the online food ordering database.

**How to run the CreateDB.sql script in PHPMYADMIN**

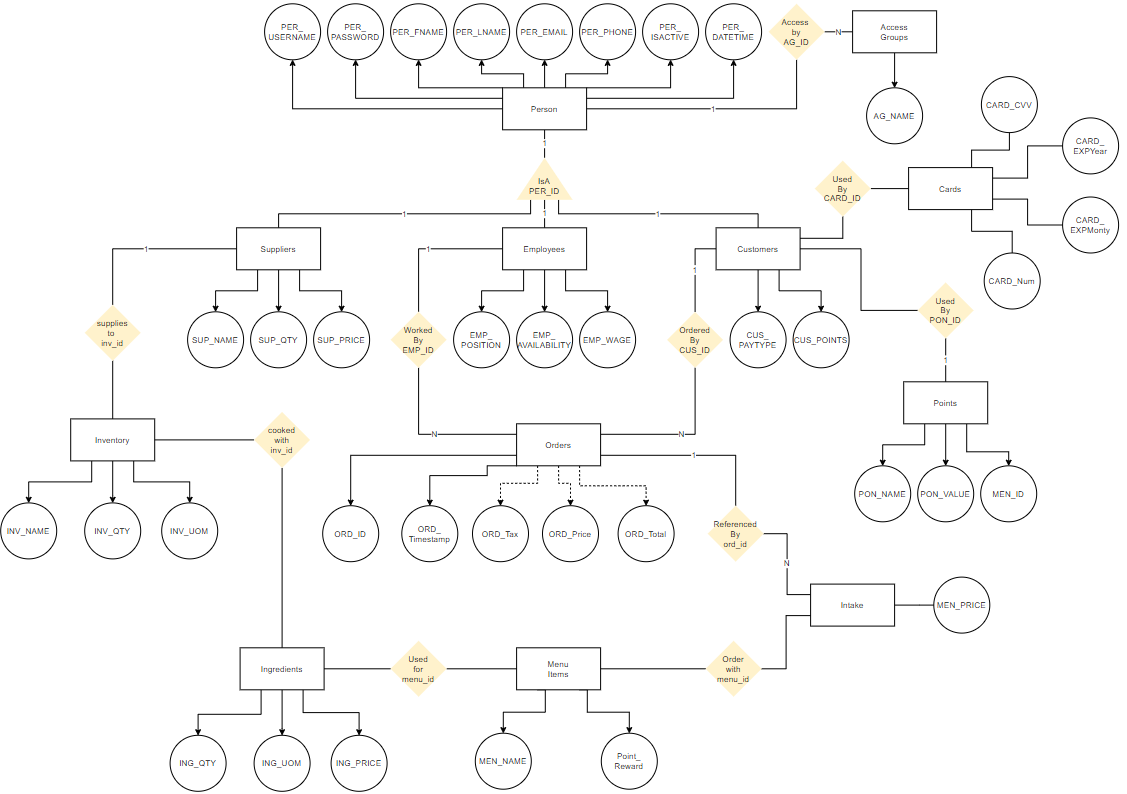
1. Select the **New** database icon.
2. Select the **IMPORT** tab.
3. Select **Choose File.**
   1. In the file explorer menu, navigate and select the .sql from the folder directory.
4. Select **Go**.

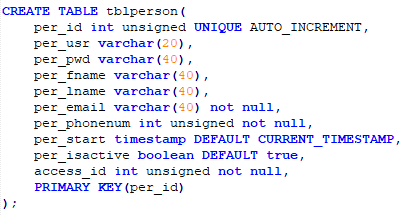
**How to run the PopulateData.sql script in PHPMYADMIN**

1. Select **ninjapizzadb** text located in the database listing on the left pane.
2. Follow steps 2-4 to import the PopulateData.sql script.

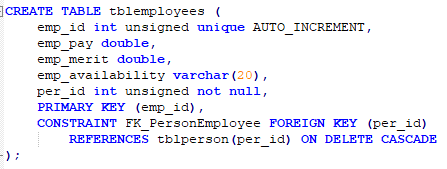
**Creating Tables**

The command “Create Database ninjapizzadb;” must be used first in order to create tables, triggers, or views in the database. These types of commands are called Data Definition Language (DDL) which are commands that help build the structure of the database. The structure design of the database is conceptually presented in the following Chen entity diagram.



The syntax to create a table is best displayed in the tblperson example below: 

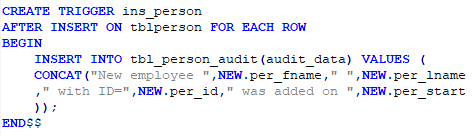
This superclass table has a primary key of per\_id that is referenced all throughout the database for any entity that represents a person. The types of people in the database are customers, employees, and suppliers. Each person plays a unique role in assisting the online food ordering database while still possessing common attributes such as username, password, first name, last name, email, and start date. The employee table references the person table through a foreign key that is shown in the example below:



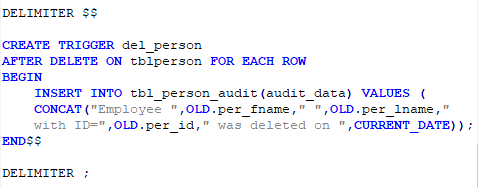
The foreign key allows for faster querying of the data and also contains an ON DELETE CASCADE condition where if the record in tblperson is deleted, the record that is related to the foreign key is also deleted from the database.

**SQL Triggers**

Events such as triggers are also included in DDL. The syntax to create a trigger event for auditing new records to tblperson is displayed below:



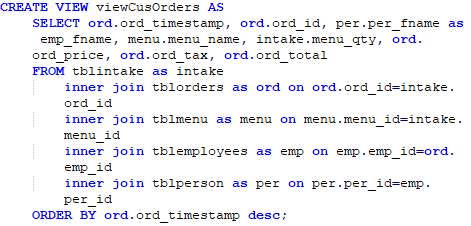
The criteria of the trigger are activated after an insert has occurred on tblperson. MySQL looks at each row in the table and executes SQL Data Manipulation Language (DML) statements between BEGIN and END. Because you can issue many DMLs within a trigger a delimiter is required for the SQL complier to recognize the ending of a statement. The delimiter use for creating triggers is listed below for the del\_person trigger:



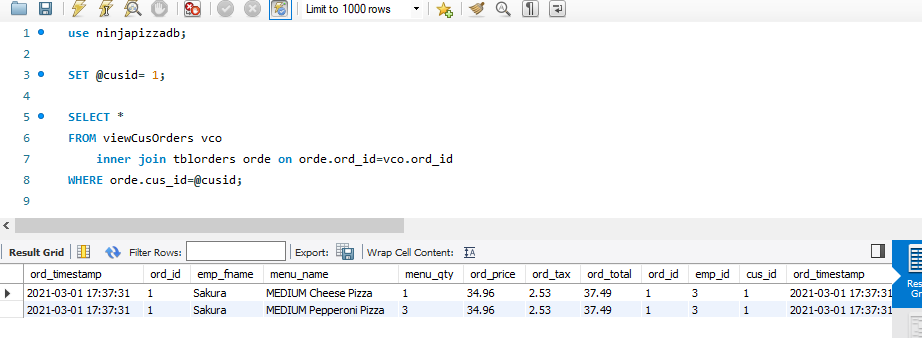
A table represent entities within a database and triggers are events that can issue DML or DDL commands based on a BEFORE or AFTER, INSERT, UPDATE, or DELETE event.

**SQL Views**

Views are structures that allow users a report to view a denormalized dataset. Technically, views are a saved query structure that can be called and saved for complex queries. A hypothetical use case would be that customers would like access to the online food ordering database to view all of their past orders. An example of this use case is displayed in the viewCusOrders view below:



The viewCusOrders view contains one DML SELECT query joining five tables to display all customer orders. A system report can be generated to filter the view by passing the customer’s id as a parameter as show in the SELECT query example below:

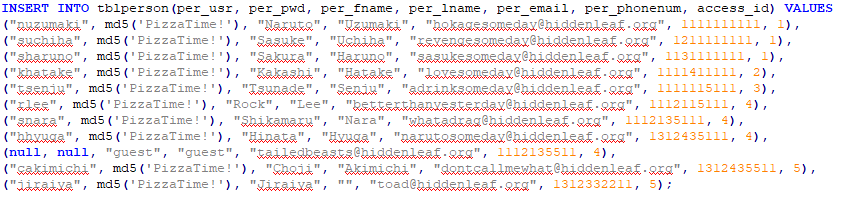


**Populating Data**

The PopulateData.sql file is required to use the command USE ninjapizzadb; in order to point the .sql file to the correct database for populating the tables that were previously imported via the CreateDB.sql file. Populating data is also considered as manipulating the data. The DML in SQL, allows the user to SELECT, INSERT, DELETE, or UPDATE data. In contrast, to CREATE, ALTER, or DROP is used by DDL to create the database framework.

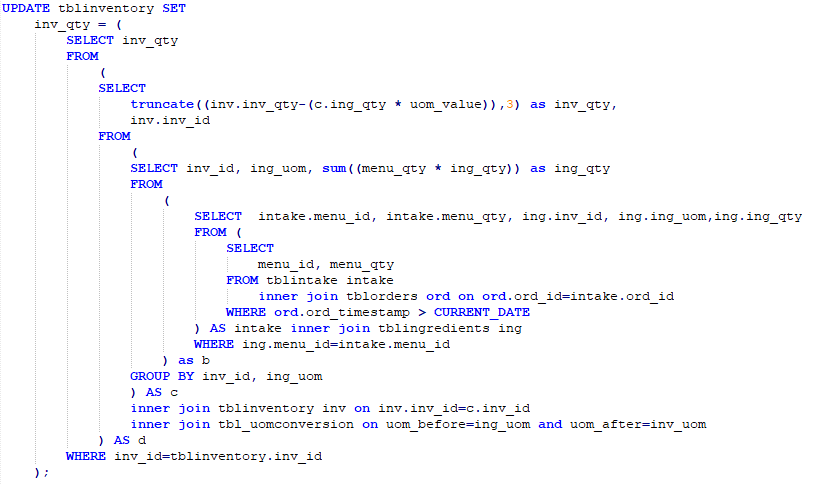
**INSERT INTO Statement**

An example of the INSERT INTO DML is displayed below:



The syntax for an insert statement uses the command INSERT INTO followed by the table and columns are wrapped in the parentheses. The command VALUES allows the user to input the values for each column. The user must follow the column order to input the correct value in the right column making a record.

**UPDATE Statement**

The basic syntax of an UPDATE statement requires the table followed by the command SET= to a value. A best practice to avoid data discrepancy is to include a WHERE clause to specifically identify what records are going to be updated by the UPDATE statement. The screenshot below demonstrates the tblinventory update statement that subtracts the menu items that were purchased based on the tblingredients table which represents each ingredient required to make a menu item: 

This UPDATE statement first looks at orders that are greater than the current date, joins tblintake with tblingredients and uses the SUM function to aggregate the ingredient quantity together. Once the first select has been completed, the query continues by joining tblinventory by inventory ID and the tbl\_uomconversion table that contains unit of measure (uom) conversions, for example: the ratio of a gram to pounds is 0.00220462, therefore if the food ordering database used three grams of sugar, the update statement must search the tblinventory table to locating the inventory ID for sugar, grabbing the uom of pounds and converting grams to pounds and subtracting the result from the inventory storage for sugar.

**End to End Process Video**

<https://drive.google.com/file/d/1wSug-advamOmqSkzxTm41zkZmT6sASUX/view?usp=sharing>

**References**

Coronel, C., & Morris, S. (2019). Database systems: Design, implementation, and management (13th ed.). Boston, MA: Cengage.