**Trigger Code**

*Preface:*

We have 25 triggers total, however, 15 of them are disjoint triggers. Disjoint triggers are triggers that ensure that two or more children of a specialization subclass do not overlap.

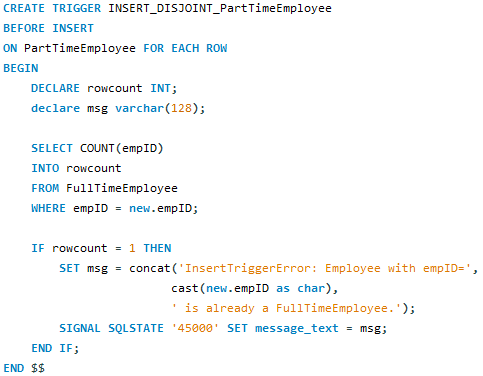
Example:

There cannot exist an **Employee** that is *both* **FullTimeEmployee** *and* **PartTimeEmployee.**

We solved this by putting a trigger on both **FullTimeEmployee** and **PartTimeEmployee** to check if the Employee in question is already in one of the two tables. If so, then the INSERT will fail.

We have 15 of these triggers which are prefaced with ‘INSERT\_DISJOINT\_<tablename>’ but they all work the same way so we will only show one example. Then we will show the other 10 which are core business logic triggers and are more interesting.

Our trigger code is located in the file trigger\_code.sql, everything in this document is just screenshots.

****

a. See preface about INSERT\_DISJOINT triggers for explanation.

b. **5** is the empID of an Employee that is already a FullTimeEmployee so we should not be able to insert that same empID into the PartTimeEmployee table but when we try:



Our trigger activates and prevents that insertion:

****



a. The above trigger ensures that an Employee is not given both the ‘evening’ and ‘morning’ shifts of the same date. This trigger enforces our business rule #5.

b. We are going to try to associate an empID that already has a shift on a certain date:



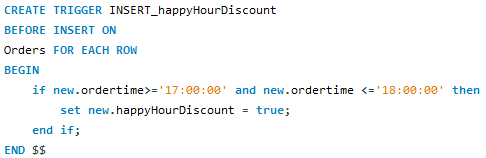
\*note:

shiftID=1 {dateOfShift= 2017-08-01 / shiftType = ‘Morning’}(this shift was already assigned to empID=12)

shiftID=2 {dateOfShift= 2017-08-01 / shiftType = ‘Evening’}

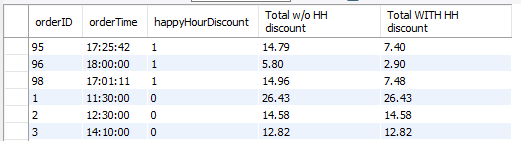
RESULT:



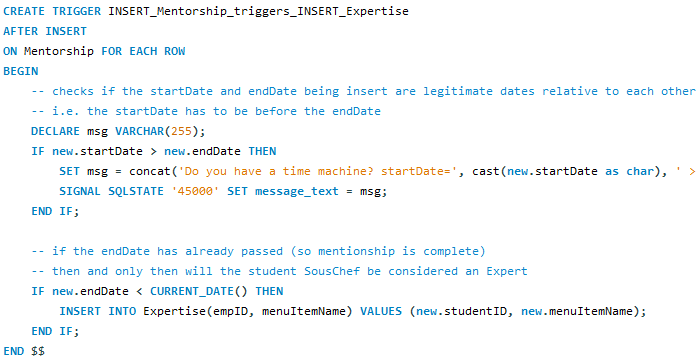


a. The above trigger occurs with every INSERT into Orders. Depending on whether the orderTime is between 5pm and 6pm, we determine if the order is eligible for a happy hour discount.

b. Not possible to show results of this trigger as it does not prevent an insert. However, one way to get a feeling for the trigger is with a query:



You can see that if the orderTime is between 5pm-6pm, the happyHourDiscount Boolean is set to true.



a. This trigger activates whenever a user/DBA inserts into the Mentorship table. It ensures that if the endDate of the Mentorship is passed then the student SousChef will be considered an Expert at that item too, therefore an insert into the Expertise table will occur. It also checks if the startDate of the Mentorship is after the endDate, otherwise the insertion since those are invalid data.

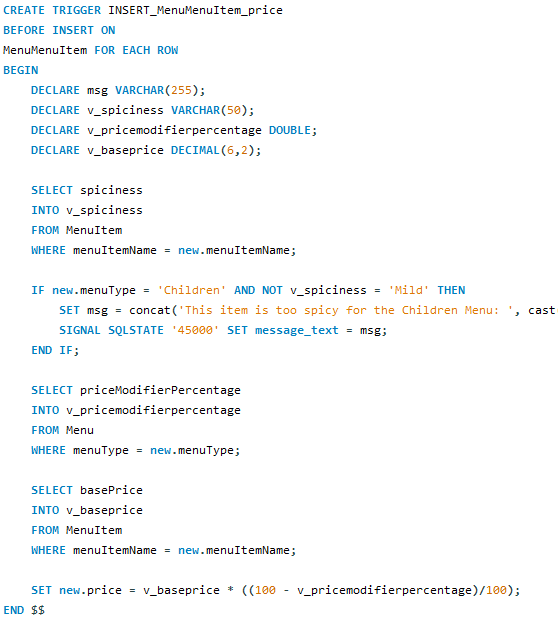
b. Here is an example of the insertion failing:



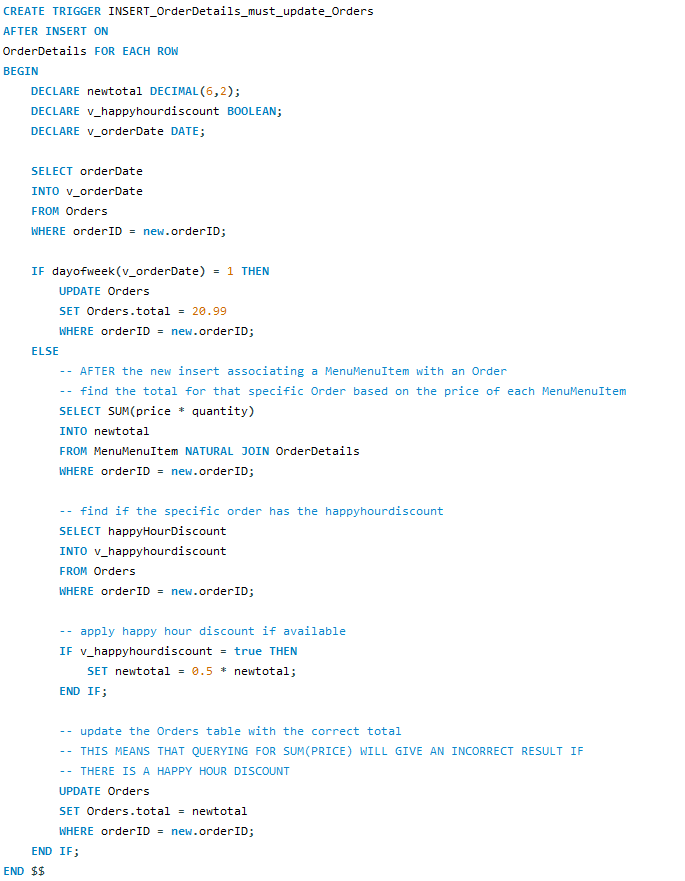
Custom error message:

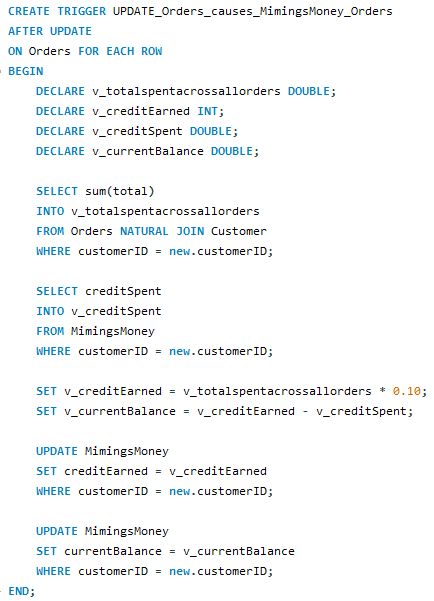


a. This trigger ensures that a Customer can only order from a valid menu. For example, our Lunch Menu is known to be from 10am to 4:59pm; if a Customer places an Order with an orderTime that is not within those ranges than they cannot order off the Lunch Menu. A similar event happens if the orderDate is not a Sunday if trying to order off the Sunday Menu.

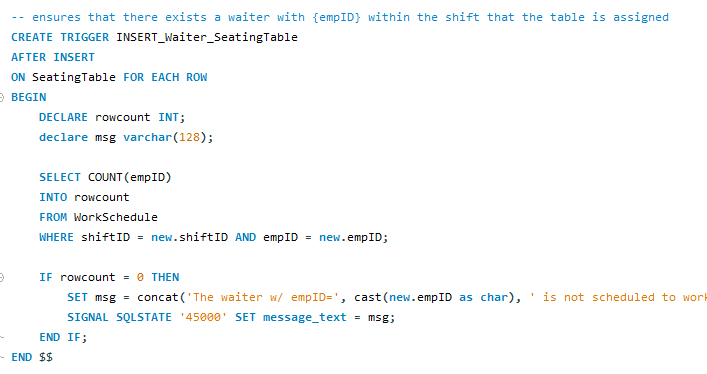


a. When a MenuItem is added to a Menu, we have to calculate the price based of the base price of the MenuItem. This trigger also checks if a MenuItem is too hot for the Children’s Menu.

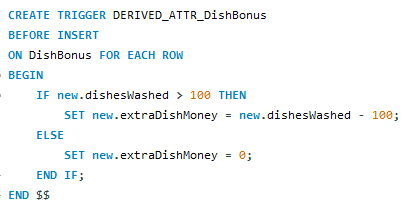
a. Whenever you insert into OrderDetails, you are essentially adding an item to the Order (ordering another item). Therefore, we must update the Orders’ total value based on the new MenuMenuItem that was added to the Order. This must happen because the Orders table contains the info that a happy hour discount is available.



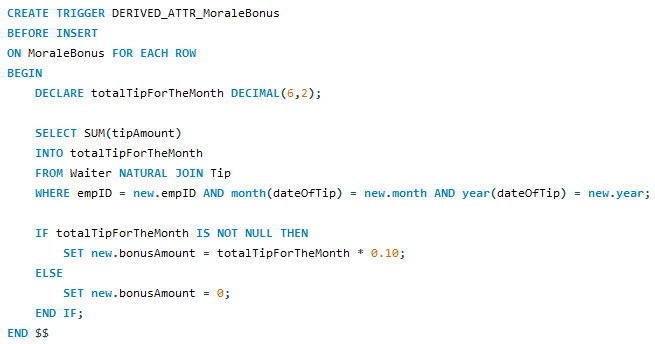
a. Everytime an Order is updated (when a Customer wants to buy more food), the Customer earned more MimingsMoney (store credit). This trigger calculates the Mimin’s credit that the Customer has earned based off the total for ALL their Orders. This value gets recalculated from scratch every update to Orders; it is not a running count/update.



a. This trigger ensures that there exists a Waiter within the WorkShift that the SeatingTable is assigned.



a. This trigger calculates the DishBonus which is our business rule #1. It ensures that the Waiter gets their dish bonus based on the amount of dishes they have washed.



a. This trigger calculates the MoraleBonus which is our business rule #3. It ensures that when we pick a Waiter at random to receive the MoraleBonus for each month, they get 10% of their tips for that entire month.