EXERCISE 10

PL/SQL Loops and Controls

1. Create a temporary table “OddEven” with three attributes – index, num, type. Datatype

of the attributes are number, number and varchar respectively. Create a PL/SQL block

that takes one number at a time (from 100 to 125), deduce whether it is odd or even

number and update the same in the “type” attribute of the temporary table.

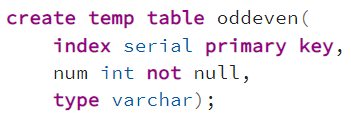
Index Num Type

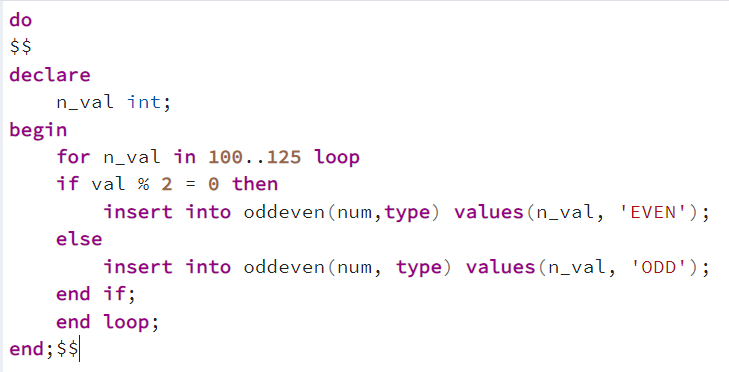
1 100 Even

2 101 Odd

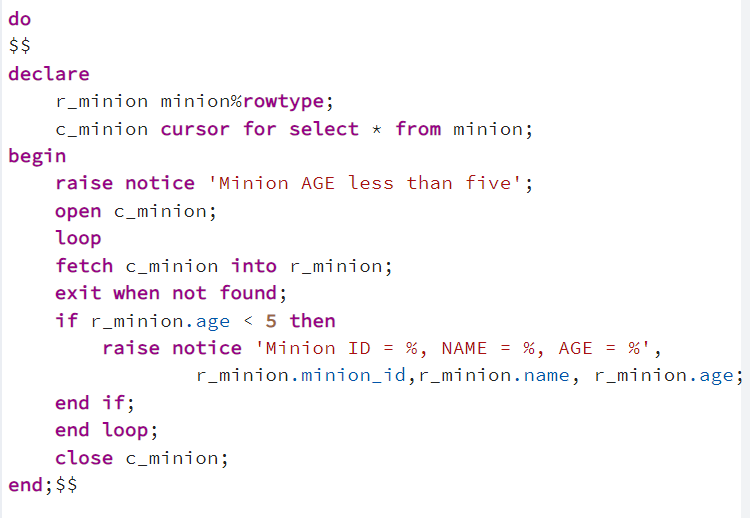
.. .. ..

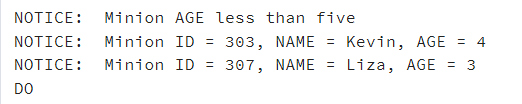
26 125 Odd



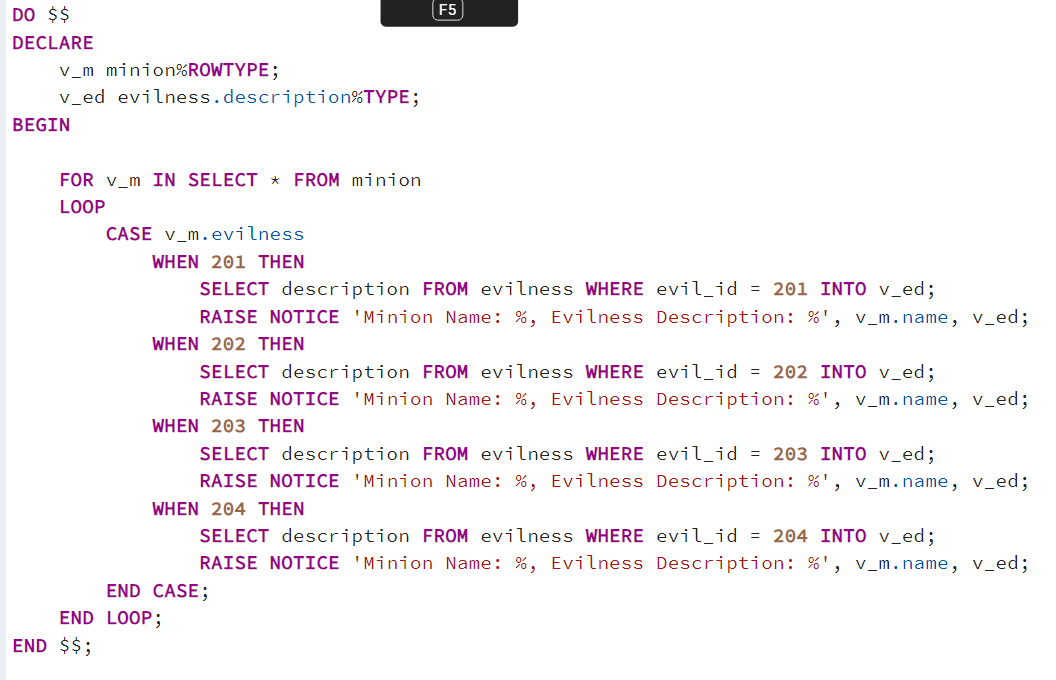


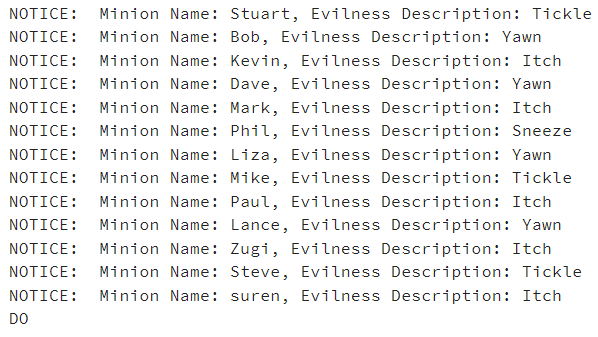
2. Create a PL/SQL block to display all the minions of age < 5.



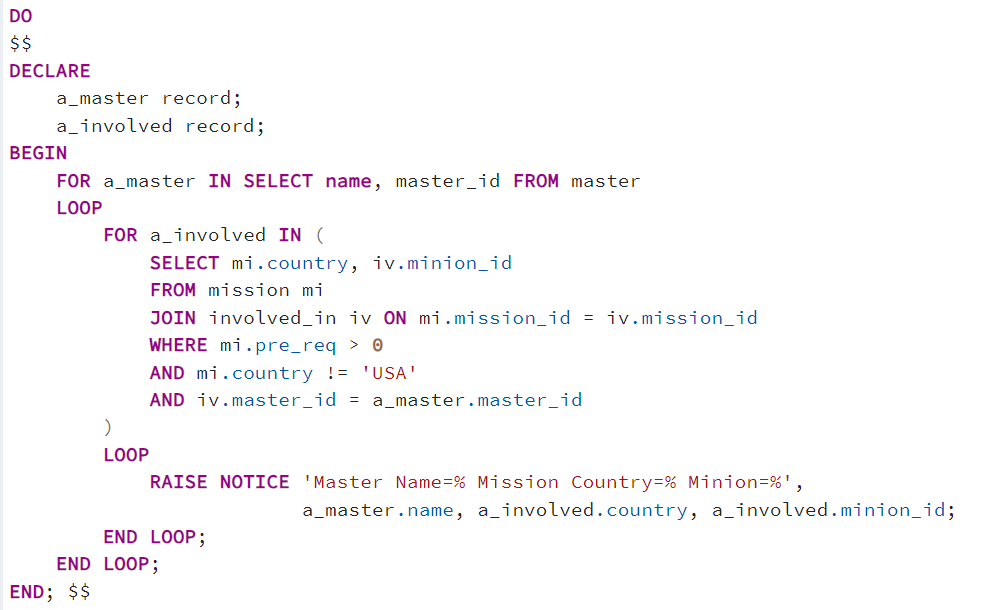


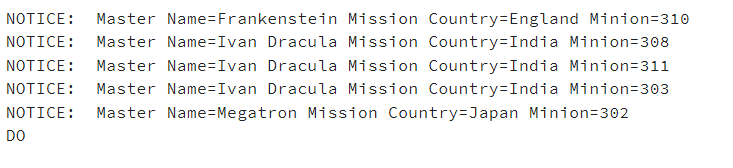
3. Display the evilness (description) of each minion using case statement.



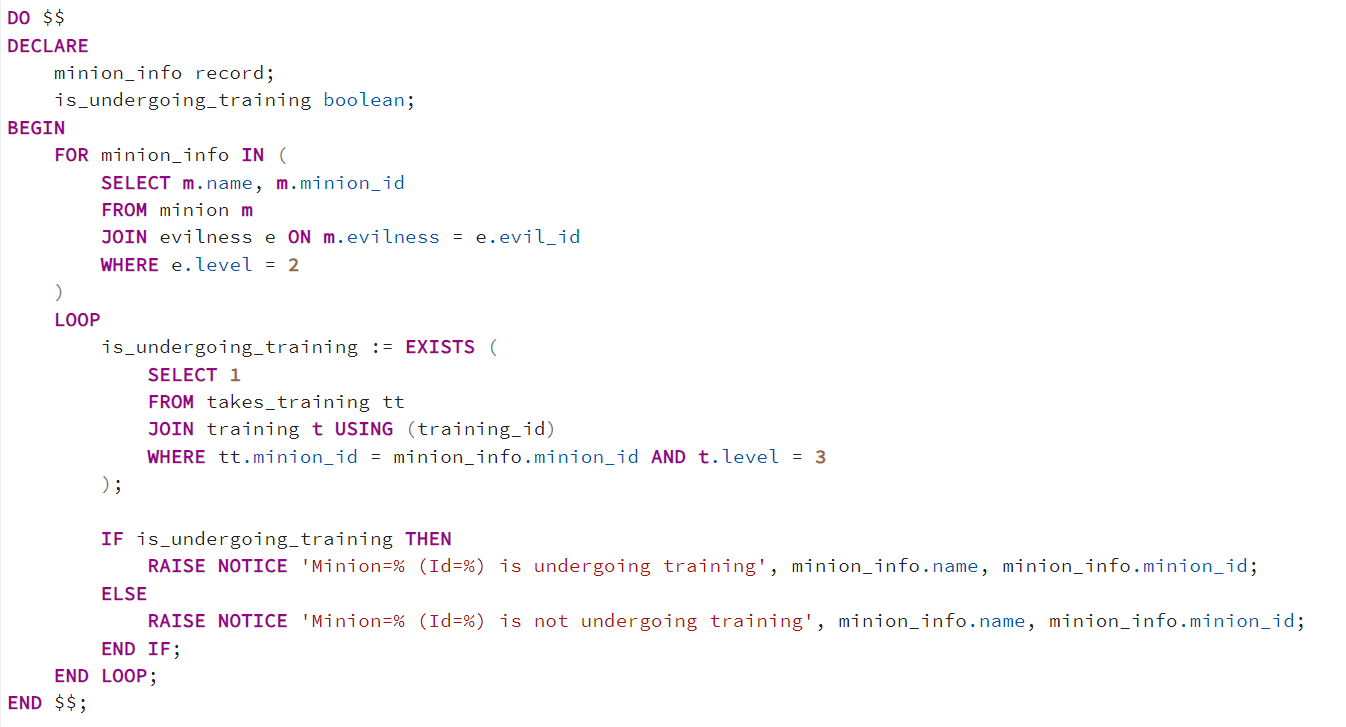


4. Display the master and the country of all missions that require pre\_req greater than three and missions not undergoing in USA.



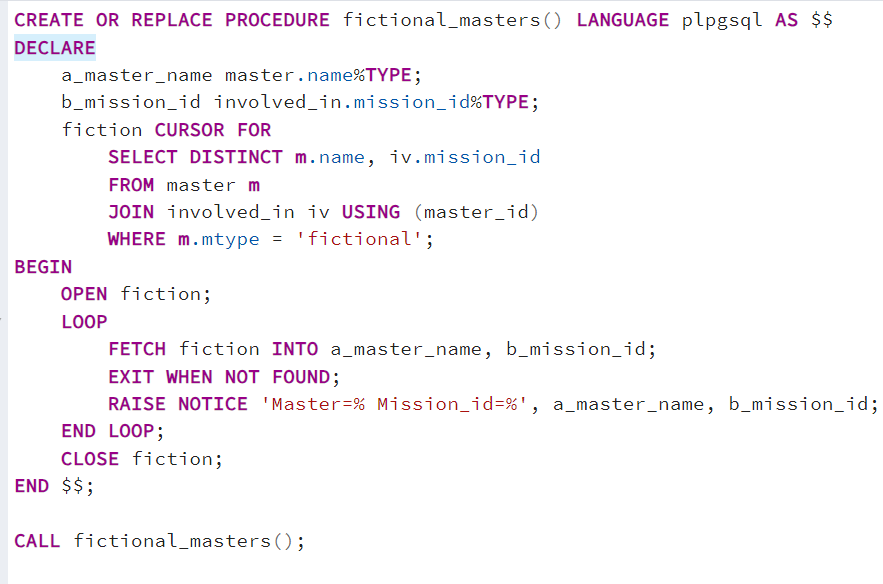


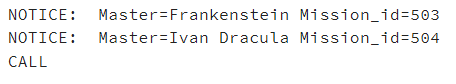
5. Find all the minions with level 2 evilness. Using control structures display whether they are undergoing training for level 3 or not.



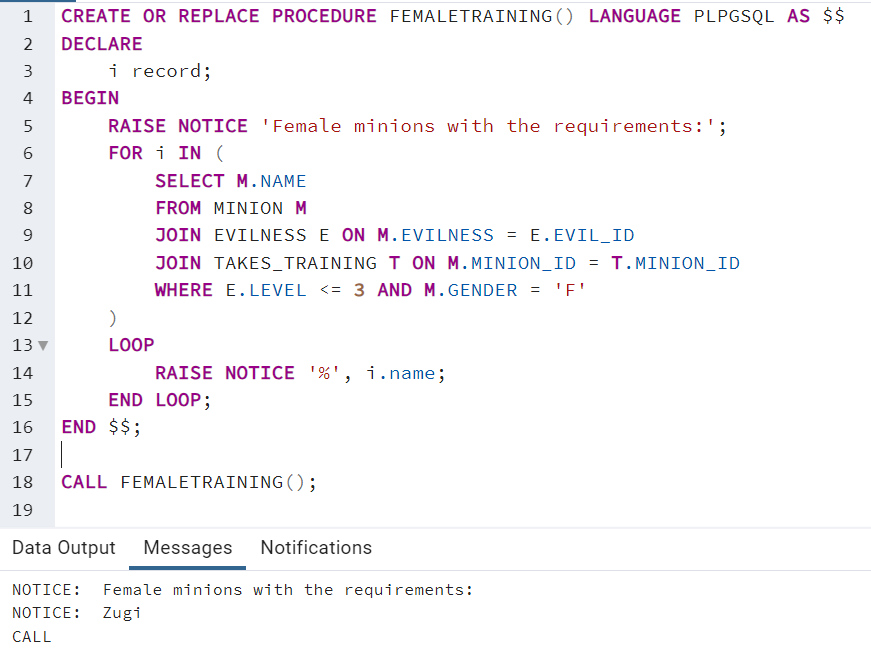
**PL/SQL Procedure**

1. Create a procedure to display all fictional masters assigned with missions.

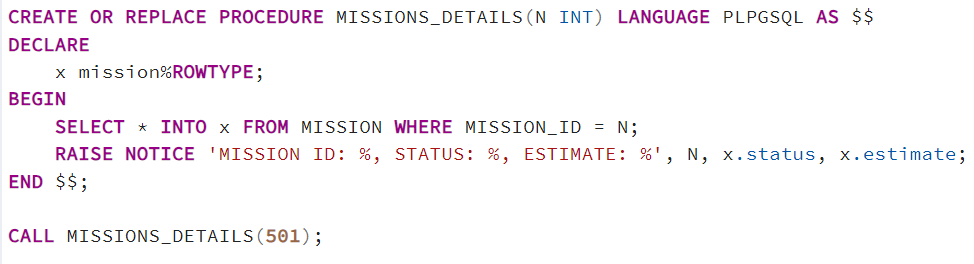




1. Create a procedure to display all female minions with evilness<=3 and undergoes training.

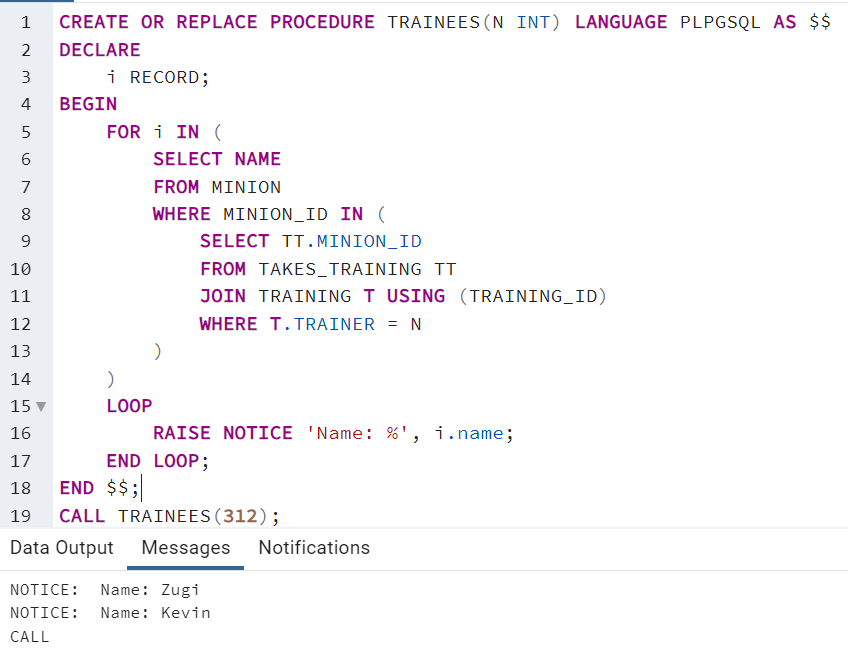


1. Create a procedure that gets the mission\_id and find the corresponding mission status and the estimate.

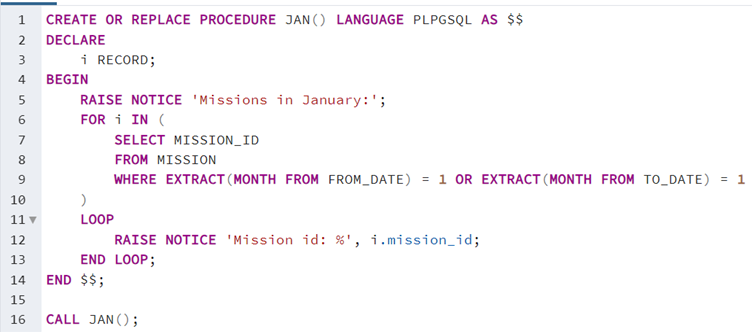




1. Create a procedure that takes trainer as the input and displays the trainees.

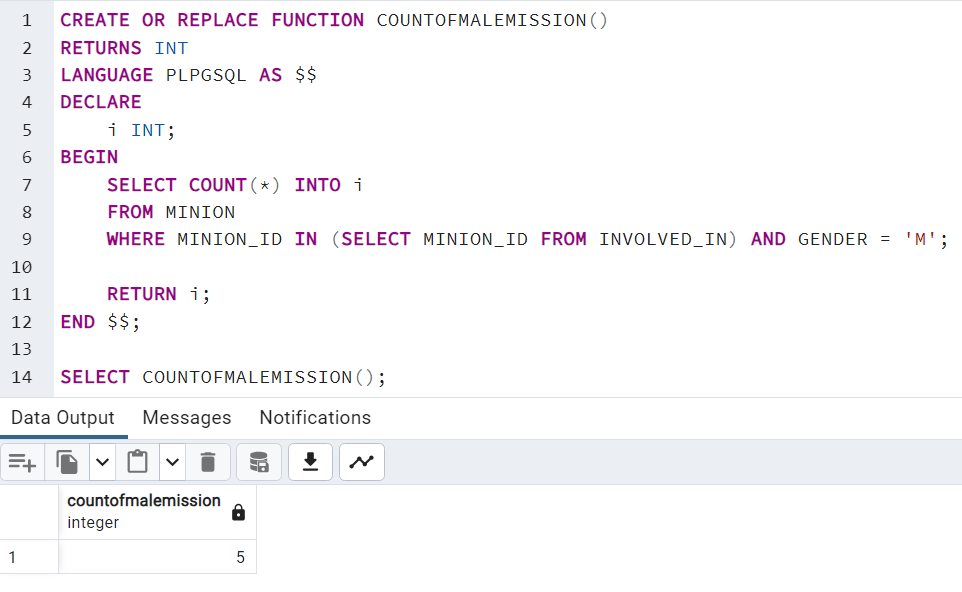


1. Write a procedure to display all missions that happen in January.

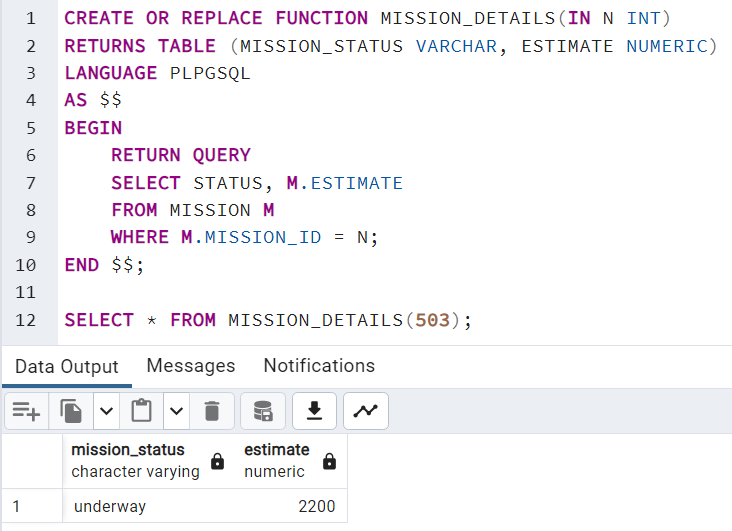


**PL/SQL Functions**

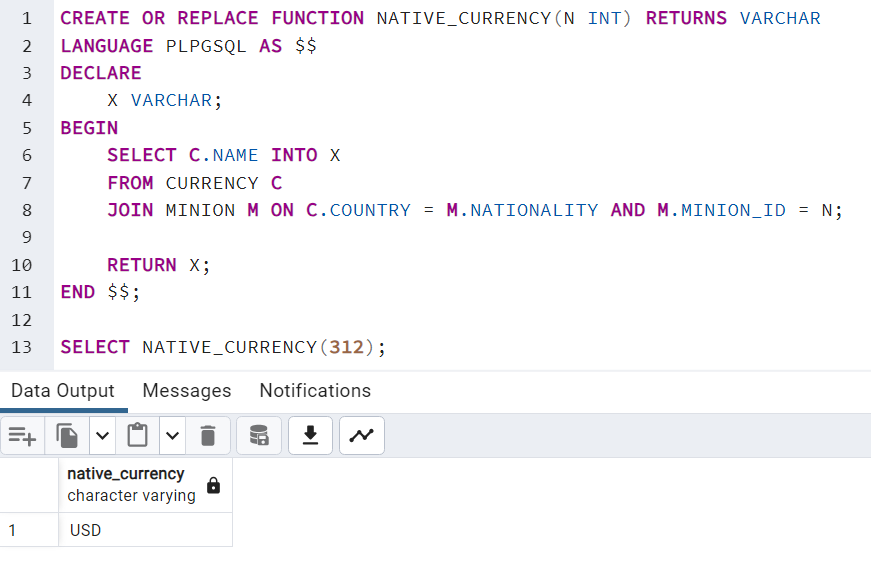
1. Create a function to display the count of male minions assigned in various missions.



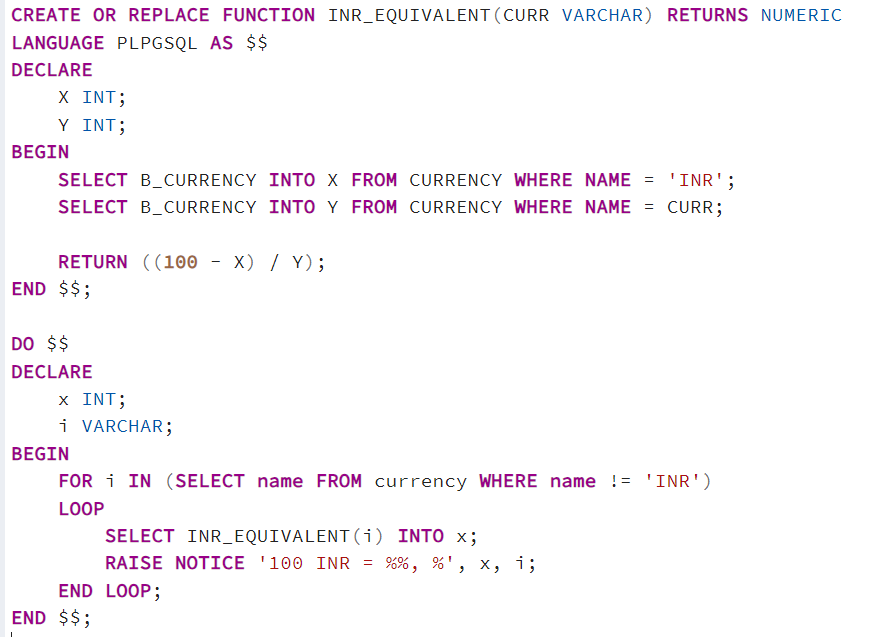
1. Create a function that gets the mission\_id and find the corresponding mission status and the estimate.



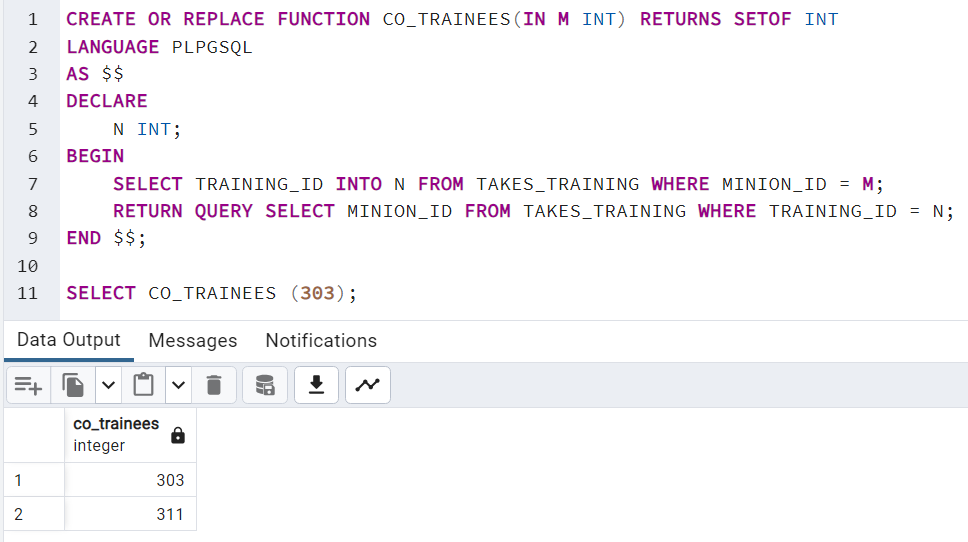
1. Write a function to receive a minion\_id and display its native currency.



1. Find the currency equivalents of INR 100.

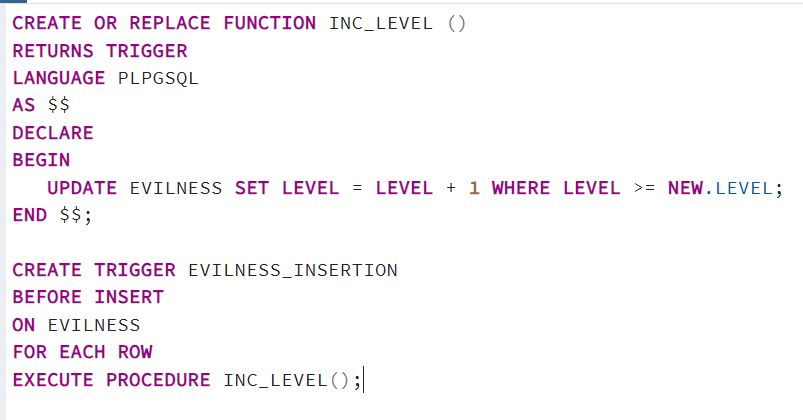


1. Find the minions who have done training together

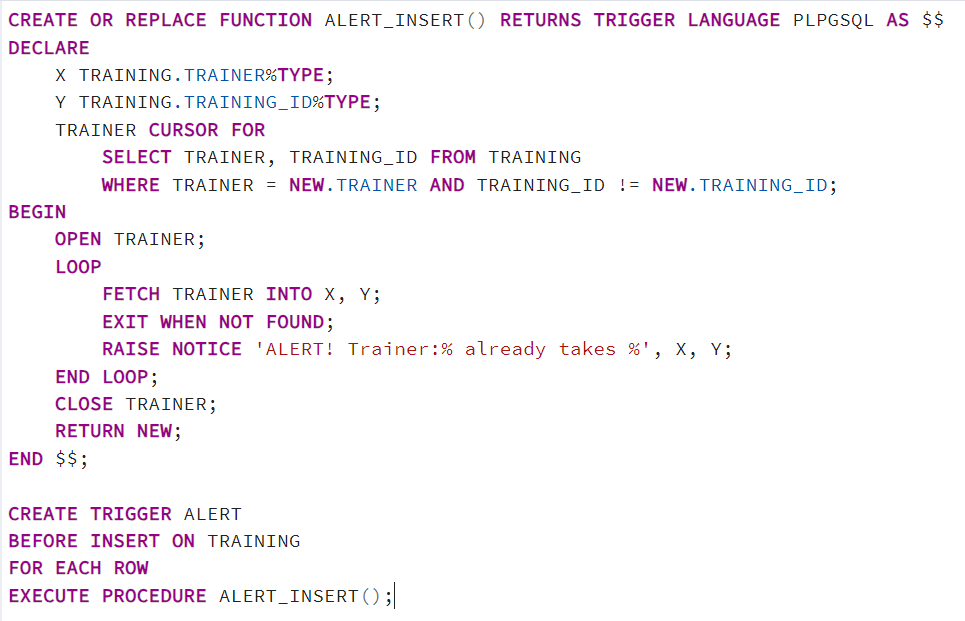


Triggers

1. Create a trigger to increment the level to all the tuples in evilness before any insert on the relation Evilness. Say, if you are inserting an evilness “Burp” for level 2, the existing tuples with level 2, 3 and 4 should be updated as 3, 4 and 5.

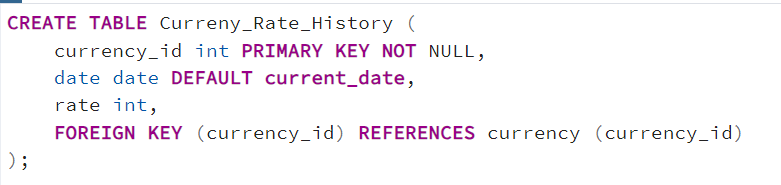


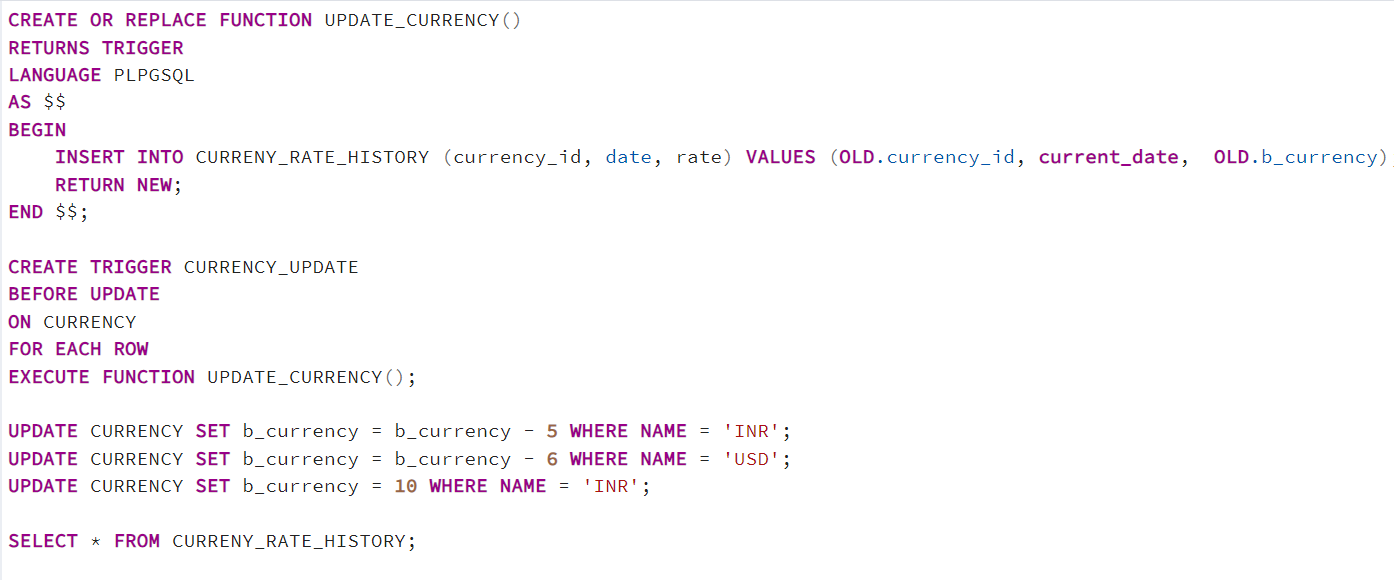
1. Create a trigger to alert the user on other training courses offered by the same trainer, when tuples are inserted to relation training.



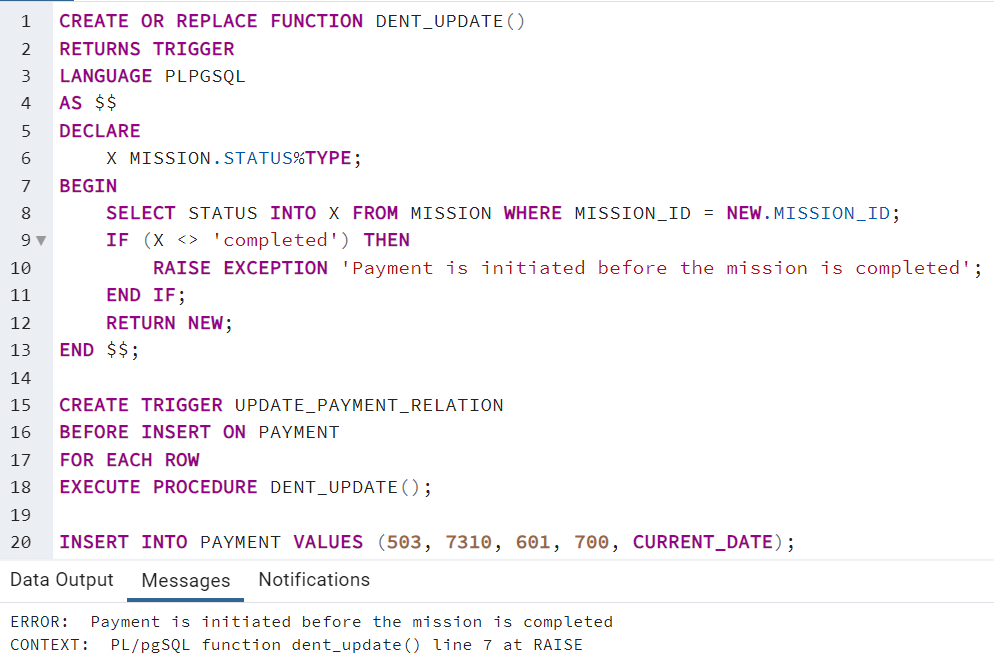


1. Create a table ‘Currency\_Rate\_History’ with attributes Currency\_id (number), rate (number) – referring to Currency(b\_currency) and date (date). Create a trigger that updates the table Currency\_Rate\_History’ before any update to the values of b\_currency in Currency table.

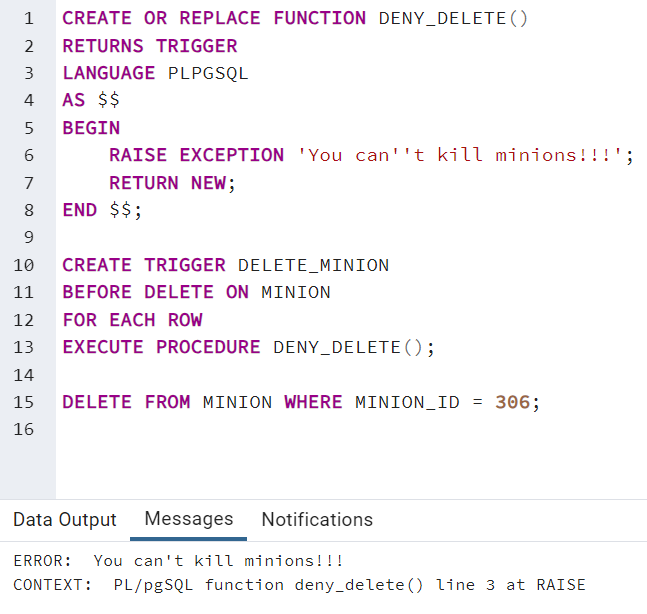




1. Make a trigger to deny update to Payment relation if payment is initiated before the completion of mission.



1. Trigger an Exception as ‘You can’t kill Minions!!!’ on trying to delete tuples from Minion.



1. Create a table identical to takes\_training as Takes\_Training\_History Create a trigger that updates a row in ‘Takes\_Training\_History’ when any tuples in takes\_training gets deleted.

