MIS 381N

HW5– DDL Script Assignment

Team Members

Name	**UT EID**
Bhavana Reddy	bc35833
Safiuddin Mohammed	sm76833
Brandt Green	bwg537
Ramya Madhuri Desineedi	rd32895
Suchit Das	sd38448
Lucas Fernandez	lf23234

Executive Summary

1. A PL/SQL block was created for the purpose of discovering the number of reservations that have been placed by a customer. The block will notify the user via server output whether or not the customer has placed more or less than 15 reservations. This block is hardcoded to only look at customer with the customer_id = 100002.
2. Another PL/SQL block was written to accomplish nearly the same goal as in '1', with the only difference being that the user will now be asked to input a specific customer_id to examine. This was accomplished by using a substitution variable.
3. An anonymous block of PL/SQL was used to insert a new customer into the customer table with NULL values for columns that did not need a value. The message "1 row has been inserted into the customer table" was reported if the code ran successfully; otherwise, an exception was raised indicating that the row was not added. A commit was run after the anonymous block.
4. An anonymous block of PL/SQL was used to bulk collect all the features that begin with the letter P from the features table, sorted alphabetically by feature name. A new table called feat_name was created to store the bulk collect of the features into and this new table was iterated over to produce an output message of "Hotel feature #?: the name of the feature" for each feature.
5. We define a cursor ' city_feature_cursor' to iterate over the joined table resulting from merging location, features, and location_feature_linking table. **As part of the bonus question, we use the substitution variable to take 'city' as input from the user**. We use an IF block within the cursor to display records only belonging to the respective city.
6. We define a stored procedure 'insert_customer' that takes 8 parameters as Input, these are the mandatory parameters on the table. Note, since we use the sequence to generate the customer_Id column we do not need to ask the user to input the same. The Procedure is tested with the given two test statements successfully.
7. We define a function 'hold_count' that takes as input parameter the customer_id. The function looks up the complete history of rooms booked by the customer by referencing the join table resulting from reservation_details and reservation tables. This is to account for cases where multiple rooms are booked on one reservation id. The function is tested by executing it for each customer in the reservation table.