Final Project

Submission

Your submission will consist of two files:

- A single text-based SQL file including all the code for the various coding activities. Make sure they are clearly marked with comments as to which question is being addressed.
  – and –
- An output file demonstrating that your stored procedures work. This must include functional completeness and error handling.

Reminders

- 1 – Make sure you use proper indentations
- 2 – Make sure you fully comment your stored procedure
- 3 – Make sure you include error handling and test your error handling code

Tasks

1 – (0%) This is preparation – It is very important to ensure you have successfully ingested the data into both tables and have the exact number of rows specified below. Double check this.

- Create the EMPLOYEE table and STAFF table based on DDL in the dbs501-assignment-DDL file
- Take file dbs501-assignment-employee and load this data into an employee table
- Take file dbs501-assignment-staff and load this data into a staff table

- Make sure all records have been successfully loaded or your result sets may be incorrect
- Perform a SELECT COUNT(*) from both tables to ensure there is an exact match with rows
- You should have 35 records in STAFF and 42 records in EMPLOYEE

You can use INSERT, IMPORT, LOAD – whatever you prefer – to ingest the data

2 – (10%) Write a stored procedure called *staff_add* that will add a new row to the STAFF table as described below:

- ID – should be calculated as 10 higher than the highest ID currently in the STAFF table
- NAME – must be provided as input
- DEPT – set this automatically to 90
- JOB – must be provided as input; must be either “Sales” or “Clerk” or “Mgr” – anything else should cause an error
- YEARS – set this automatically to 1
- SALARY – must be provided as input
- COMM – must be provided as input

WHAT TO HAND IN: A copy of your stored procedure and the output of a successful insert, and error handling

3 – (10%) Create an INSERT trigger *ins_job* to enhance the error checking on the JOB column. If an incorrect job is provided an error record should be placed in the STAFFAUDTBL – which records the error. It should include the following columns:

- ID – the ID of the record being input

- INCJOB – the value of the JOB that was incorrect

WHAT TO HAND IN: A copy of your trigger code and a copy of STAFFAUDTBL after testing a few INSERTs with incorrect JOB values.

4 – (10%) Create a function called *total_cmp* which returns the total compensation (the sum of SALARY and COMM) and takes ID as input. This is a single row function. Make sure to handle the situation where an invalid ID is provided.

WHAT TO HAND IN: A copy of your function code and output showing a successful call of the function – and – a call with an invalid ID.

5 – (20%) Create a stored procedure called *set_comm* which will go through each record in the STAFF table and set the COMM columns as follows:

- 20% of the SALARY if JOB = “Mgr”
- 10% of the SALARY if JOB = “Clerk”
- 30% of the SALARY if JOB = “Sales”
- 50% of the SALARY if JOB = “Prez”

Use an UPDATE trigger called *upd_comm* which will record (insert) a record in the STAFFAUDTBL which records what change was made to the COMM of each individual in the STAFF table.

You will need to add two columns to the STAFFAUDTBL table. One called OLDCOMM and another called NEWCOMM.

When INSERTing a record into this table for Q5 you should fill in the ID, set INCJOB to NULL, place the original commission (COMM) in the OLDCOMM field and new commission (COMM) in the NEWCOMM field.

WHAT TO HAND IN: A copy of your stored procedure code and trigger code – and – output of the STAFF table after this stored procedure has been executed – and – a copy of your STAFFAUDTBL after the stored procedure has been executed.

6 – (10%) Take the 2 triggers you have previously created and combine them into a single trigger called staff_trig which continues to provide all the functionality of the previous triggers – and – also handles a DELETE by recording (INSERT) a record into the STAFFAUDTBL when a DELETE occurs. Here is the new STAFFAUDTBL schema and how to handle I/U/D:

Columns:

- ID (ID of the record being affected)
- ACTION ("I" or "U" or "D" based on what action is happening)
- INCJOB (The value of a non-valid JOB in the case of an INSERT or UPDATE, otherwise NULL if JOB value is fine – see valid values in Q3)
- OLDCOMM (The value of old commission if there is ANY change to COMM from an UPDATE, otherwise NULL)
- NEWCOMM (The value of new commission if there is ANY change to COMM from an UPDATE, otherwise NULL)

Drop the existing STAFFAUDTBL.

Create a new STAFFAUDTBL.

Make sure you DISABLE your two existing triggers so both don't fire at the same time.

Test your new trigger.

WHAT TO HAND IN: A copy of your trigger code – and – the output of your STAFFAUDTBL after you have tested with an INSERT, UPDATE and DELETE. Make sure to test both good and bad JOB values.

7 – (10%) Create a new function called *fun_name* which will take the NAME as input and provide an output of the name which alternates between upper case and lower case characters. For example:

Smith -> SmItH

Robertson -> RoBeRtSoN

This is a single row function only. Do not worry about handling multiple rows.

This function provides the output, but, DOES NOT update the values in the table.

WHAT TO HAND IN: A copy of your function code – and – output showing a couple of examples of the function working.

8 – (20%) Create a new function called *vowel_cnt* which will take multiple NAME values as input and count the number of vowels (only count “A”, “a”, “E”, “e”, “I”, “I”, “O”, “o”, “U”, “u”). This is a multi-row function. The function needs to be called on the NAME column or JOB column – for instance:

SELECT vowel_cnt(NAME) from STAFF – or –

SELECT vowel_cnt(JOB) from STAFF.

WHAT TO HAND IN: A copy of your function code – and – output showing a couple of examples of the function working on each of the

columns. Make sure to test when there are no rows in the result set – you can use the WHERE clause for this.

9 – (10%) Create a package called staff_pck which defines and externalizes the stored procedures and functions we have defined in the previous questions. Make sure to include:

- Package Specification
- Package Body

WHAT TO HAND IN: A copy of your package specification, package body definition – and – pick a couple of stored procedures and/or functions to call from your package.