ASSIGNMENT 2: PL/SQL LO 3

This assignment is worth 25% of the overall module mark.

NGĀ AROMATAWAI | ASSESSMENT DETAILS

INTRODUCTION

XTREME Quality Cars Inc. (XTREME QC) and the Hamilton outlet (XTREME QC Hamilton) managed by Micheal Knapp are very happy with the database solution you created for them. Micheal has asked for a series of reports for analysis purposes and has asked you to provide a reporting system which would make it easy for him to retrieve data from the database. Micheal has also informed you that EXTREME QC in Auckland would like to use the reporting system as well.

Knowing that Micheal is not a database expert nor does he understand SQL and that EXTREME QC in Auckland would need access to the reporting system you have decided to implement a series of procedures, functions and packages as the solution. This means that Micheal or EXTREME QC need only execute procedure or package calls making it easier for them and more productive.

RESOURCE FILES DOWNLOAD

Use the following resource files for Assignment 2 which is the complete database for XTREME QC

1. Download the [Sample ER Model for XTREME QC Hamilton]
2. Download the [EXTREMEQCHamiltonDatabase.sql] file and run the script on your database

ASSUMPTION, RESTRICTIONS AND LIMITATIONS

To complete the assessment the following assumptions are made

- Vehicles that appear in the database are either is/has or are/were on the lot. It makes sense that vehicles sold do not appear on the lot, however, it does not mean they are removed from the database – a database keeps track of all records. Therefore, questions which request queries using the phrase 'is/has' or 'are/were' refer to all vehicles on the lot – aka in the database regardless of being sold.
- All data in the database is accurate, such as
    - All vehicles that are/were on the lot have been ordered and have been purchased after the order date
    - All orders were made by supervisors
    - Sales people who are supervisors currently supervise three people or less
    - All sales purchases involving sales people have occurred after the sales persons start date
    - All customers in the database have purchased one or more vehicles
    - All purchase sales dates appear after the date of the vehicle is removed from the lot
- To implement such business rules go beyond the scope of the assessment, however, the assessment does implement a few rules

REPORTING WITH PL-SQL

1. Create the following reports using CSV formatting, Cursors and Exceptions [20 Marks]
   1.1. Customers: Displays all customers that have purchased a vehicle. Include the *Customer ID, First and Last Name, City, Vehicle Registration, Make, Model, Colour* and *Sales Purchase Total*
   1.2. Purchase Sales: Displays all purchase sales of vehicles. Include the *Invoice Number, Sale Purchase, Customer details (ID, Name and Phone), Vehicle details (Rego, Year, Make and Model), Sales Person details (ID, Name and Supervisor ID)*

2. Create Procedures with CSV output for efficient flexible reporting. Requests input from the user for parameter input into procedures [40 Marks]
   2.1. Vehicles are available for 'Sale' and Micheal would like to find vehicles by **make** or **model**. The procedure names are *SaleByMake()* and *SaleByModel()* - create procedures for both. Test your procedures with 'Mazda' and 'Bluebird'
   2.2. Performs **purchase sales reports for a given year** or **purchase sales reports for a time period** – a time period has a start date and an end date. Create a procedure that encompasses the report, the procedure name has *SalesReport()*. Test your procedure with time periods for the year 2015 and from the period of January of 2016 to June 2016.
   2.3. Create **TWO** procedures for searching customers details by a ***specific id*** and ***last name***. Include the details of the customer including the total sale purchase total, the vehicle details, and the salesperson's id who sold the vehicle. Test your procedures with customer id 14 and customer name 'Avern'
   2.4. Performs payments for a **specific time period** – a time period has a start date and an end date. Create the required procedures. Test your procedure with period of the year 2015.

3. Create a Function with a Procedure CSV Report for the following
   3.1. Create a **function** *NumberOfDays()* that calculates the number of days between two dates. The function accepts the start date and end date, then returns how many days it covers over that period. [5 Marks]
   3.2. Create a CSV procedure report that determines *how many days since an order was made?*. Include the *order id, order date, total qty, total* and the *number of days* since it was ordered. Use the NumberOfDays() function to calculate the number of days since the order date and the current date. [5 Marks]
   HINTS: Perform the function call in the FOR loop when creating the CSV output

MAINTENANCE WITH PL-SQL

UPDATE AUTOMATION AND TRIGGERS

Maintenance

4. The design of the database requires fields to be updated for formula calculations or cross-table field updates upon data insertion. These occur on the Sales Purchase, Order Line and Orders

Tables. For maintenance and efficiency the following procedures are created to automate the process. [20 Marks Each]

4.1. Create a procedure called *AddPurchaseSale()* which inserts data into the Sales Purchases table. The procedure performs the following:

- 4.1.1. Accepts parameters to insert data into the Sales Purchase table. Data that is pulled from another table should not be listed as a parameter – refer to the *XTREMEQCHamiltonDatabase.SQL* assignment script for the insert statement into the table and determine which fields are appropriate as parameters
- 4.1.2. Substitute the parameters for the data values in the INSERT. . . SELECT statement
- 4.1.3. Add the UPDATE formulas for the Additional Cost and Total fields from the assignment script
- 4.1.4. If the purchase is successful, then it should COMMIT the transaction - any errors should display an appropriate error message and ROLLBACK the transaction
- 4.1.5. Test the procedure with test data to show it works. Test the procedure as such:
  - 4.1.5.1. Perform a SELECT query on Sales_Purhases
  - 4.1.5.2. Call the AddPurchaseSale() procedure with suitable test data
  - 4.1.5.3. Perform a SELECT query on Sales_Purhases again
  - 4.1.5.4. Perform a ROLLBACK so that the database returns to its original state

4.2. Create a procedure called *AddPurchaseOrderItem()* which performs similar data inserts to Order Lines. The procedure performs the following:

- 4.2.1. Accepts parameters to insert data into the Order Lines table – make sure to choose appropriate parameters again
- 4.2.2. Substitute the parameters for the data values in the INSERT. . . SELECT statement
- 4.2.3. Add the UPDATE formulas for the Subtotal and UPDATE formulas for the Orders table from the assignment script
- 4.2.4. Test the procedure with test data to show it works - create an order first, and then test the procedure as such
  - 4.2.4.1. Perform a SELECT query on orders
  - 4.2.4.2. Create an Order so that you can get an order number
  - 4.2.4.3. Perform a SELECT query on your order to get the order number
  - 4.2.4.4. Call the AddPurchaseOrderItem() with your order number and other required parameters
  - 4.2.4.5. Perform a select query on Order_Lines and Orders based on your order number
  - 4.2.4.6. Perform a ROLLBACK so that the database returns to its original state

Triggers

5. Create the following business rules [10 Marks]

5.1. A supervisor can supervise no more than 2 people. This trigger should occur on INSERT and UPDATE from the sales persons table. Test that the trigger is working by inserting rows into Sales_Person with the same supervisor. Use ROLLBACK to return the database to its original state when testing is completed

MARKING SCHEDULE

MARKING CRITERIA

	DESCRIPTION	MARKS
	REPORTING WITH PL-SQL **(50 MARKS)**	
	CURSORS AND PL/SQL	20
	PROCEDURES	40
	FUNCTIONS	10
	MAINTENANCE WITH PL-SQL **(25 MARKS)**	
	UPDATE AUTOMATION	20
	TRIGGERS	10
	Total	**100**