|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | | | **Type** | | **Return Type** |
| ADD\_CUSTOMER | | | Stored Procedure | | None |
| **Description** | Add a new customerto Customertable. | | | | |
| **Parameters** | **Name** | **Type** | | **Description** | |
| pcustid | INT | | Customer Id | |
| pcustname | nvarchar(100) | | Customer Name | |
| **Requirements** | Insert a new customer using parameter values.  Set the SALES\_YTD value to zero. Set the STATUS value to 'OK' | | | | |
| **Exceptions** | **Type** | | | **Throw Exception Details** | |
| Duplicate primary key | | | 50010. Duplicate customerID | |
| pcustid outside range: 1-499 | | | 50020. CustomerID out of range | |
| Other | | | 50000. Use value of error\_message() | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | | | **Type** | | **Return Type** |
| DELETE\_ALL\_CUSTOMERS | | | Stored Procedure | | Int |
| **Description** | Delete all customers from Customer table. | | | | |
| **Parameters** | **Name** | **Type** | | **Description** | |
| **Requirements** | Delete all customers from Customer table.  Return the int of rows deleted | | | | |
| **Exceptions** | **Type** | | | **Throw Exception Details** | |
| Other | | | 50000. Use value of error\_message() | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | | | **Type** | | **Return Type** |
| ADD\_PRODUCT | | | Stored Procedure | | None |
| **Description** | Add a new productto Producttable. | | | | |
| **Parameters** | **Name** | **Type** | | **Description** | |
| pprodid | Int | | ProductId | |
| pprodname | nvarchar | | ProductName | |
| pprice | money | | Price | |
| **Requirements** | Insert a new productusing parameter values.  Set the SALES\_YTD value to zero. | | | | |
| **Exceptions** | **Type** | | | **Throw Exception Details** | |
| Duplicate primary key | | | 50030. Duplicate productID | |
| pprodid outside range: 1000 - 2500 | | | 50040. ProductID out of range | |
| pprice outside range:  0 – 999.99 | | | 50050. Price out of range | |
| Other | | | 50000. Use value of error\_message() | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | | | **Type** | | **Return Type** |
| DELETE\_ALL\_PRODUCTS | | | Stored Procedure | | Int |
| **Description** | Delete all products from Product table. | | | | |
| **Parameters** | **Name** | **Type** | | **Description** | |
| **Requirements** | Delete all products from Product table.  Return the int of rows deleted | | | | |
| **Exceptions** | **Type** | | | **Throw Exception Details** | |
| Other | | | 50000. Use value of error\_message() | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | | | **Type** | | **Return Type** |
| GET\_CUSTOMER\_STRING | | | Stored Procedure | | None |
| **Description** | Get one customers details from customer table | | | | |
| **Parameters** | **Name** | **Type** | | **Description** | |
| pcustid | Int | | CustomerId | |
| pReturnString | NVARCHAR(1000) | | OUT Parameter | |
| **Requirements** | Assign a string to the OUT parameter using the format:  Custid: 999 Name:XXXXXXXXXXXXXXXXXXXX Status XXXXXXX SalesYTD:99999.99 | | | | |
| **Exceptions** | **Type** | | | **Throw Exception Details** | |
| No matching customer id found | | | 50060. Customer ID not found | |
| Other | | | 50000. Use value of error\_message() | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | | | **Type** | | **Return Type** |
| UPD\_CUST\_SALESYTD | | | Stored Procedure | | None |
| **Description** | Update one customer's sales\_ytd value in the customer table | | | | |
| **Parameters** | **Name** | **Type** | | **Description** | |
| pcustid | Int | | CustomerId | |
| pamt | Money | | Change Amount | |
| **Requirements** | Change one customer's SALES\_YTD value by the pamt value. | | | | |
| **Exceptions** | **Type** | | | **Throw Exception Details** | |
| No rows updated | | | 50070. Customer ID not found | |
| pamt outside range:  -999.99 to 999.99 | | | 50080. Amount out of range | |
| Other | | | 50000. Use value of error\_message() | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | | | **Type** | | **Return Type** |
| GET\_PROD\_STRING | | | Stored Procedure | | None |
| **Description** | Get one products details from product table | | | | |
| **Parameters** | **Name** | **Type** | | **Description** | |
| pprodid | Int | | ProductId | |
| pReturnString | NVARCHAR(1000) | | OUT Parameter | |
| **Requirements** | Assign a string to the OUT parameter using the format:  Prodid: 999 Name:XXXXXXXXXXXXXXXXXXXX Price 999.99 SalesYTD:99999.99 | | | | |
| **Exceptions** | **Type** | | | **Throw Exception Details** | |
| No matching product id found | | | 50090. Product ID not found | |
| Other | | | 50000. Use value of error\_message() | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | | | **Type** | | **Return Type** |
| UPD\_PROD\_SALESYTD | | | Stored Procedure | | None |
| **Description** | Update one product's sales\_ytd value in the product table | | | | |
| **Parameters** | **Name** | **Type** | | **Description** | |
| pprodid | Int | | ProductId | |
| pamt | Money | | Change Amount | |
| **Requirements** | Change one product's SALES\_YTD value by the pamt value. | | | | |
| **Exceptions** | **Type** | | | **Throw Exception Details** | |
| No rows updated | | | 50100. Product ID not found | |
| pamt outside range:  -999.99 to 999.99 | | | 50110. Amount out of range | |
| Other | | | 50000. Use value of error\_message() | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | | | **Type** | | **Return Type** |
| UPD\_CUSTOMER\_STATUS | | | Stored Procedure | | None |
| **Description** | Update one customer's status value in the customer table | | | | |
| **Parameters** | **Name** | **Type** | | **Description** | |
| pcustid | Int | | CustomerId | |
| pstatus | Nvarchar | | New status | |
| **Requirements** | Change one customer's status value. | | | | |
| **Exceptions** | **Type** | | | **Throw Exception Details** | |
| No rows updated | | | 50120. Customer ID not found | |
| Invalid status  (not either OK or SUSPEND) | | | 50130. Invalid Status value | |
| Other | | | 50000. Use value of error\_message() | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | | | **Type** | | **Return Type** |
| ADD\_SIMPLE\_SALE | | | Stored Procedure | | None |
| **Description** | Update one customer's status value in the customer table | | | | |
| **Parameters** | **Name** | **Type** | | **Description** | |
| pcustid | Int | | CustomerId | |
| pprodid | Int | | ProductId | |
| pqty | Int | | Sale Qty | |
| **Requirements** | Check if customer status is 'OK'. If not raise an exception.  Check if quantity value is valid. If not raise an exception.  Update both the Customer and Product SalesYTD values  Note: The YTD values must be increased by pqty \* the product price  Calls UPD\_CUST\_SALES\_YTD and UPD\_PROD\_SALES\_YTD | | | | |
| **Exceptions** | **Type** | | | **Throw Exception Details** | |
| Sale Quantity range  1 - 999 | | | 50140. Sale Quantity outside valid range | |
| Invalid customer status  (status is not 'OK') | | | 50150. Customer status is not OK | |
| No matching customer id found | | | 50160. Customer ID not found | |
| No matching product id found | | | 50170. Product ID not found | |
| Other | | | 50000. Use value of error\_message() | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | | | **Type** | | **Return Type** |
| SUM\_CUSTOMER\_SALESYTD | | | Stored Procedure | | Int |
| **Description** | Sum and return the SalesYTD value of all rows in the Customer table | | | | |
| **Parameters** | **Name** | **Type** | | **Description** | |
| **Requirements** | Sum and return the SalesYTD value of all rows in the Customer table | | | | |
| **Exceptions** | **Type** | | | **Throw Exception Details** | |
| Other | | | 50000. Use value of error\_message() | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | | | **Type** | | **Return Type** |
| SUM\_PRODUCT\_SALESYTD | | | Stored Procedure | | Int |
| **Description** | Sum and return the SalesYTD value of all rows in the Product table | | | | |
| **Parameters** | **Name** | **Type** | | **Description** | |
| **Requirements** | Sum and return the SalesYTD value of all rows in the Product table | | | | |
| **Exceptions** | **Type** | | | **Throw Exception Details** | |
| Other | | | 50000. Use value of error\_message() | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | | | **Type** | | **Return Type** |
| GET\_ALL\_CUSTOMERS | | | Stored Procedure | | None |
| **Description** | Get all customer details and return as a SYS\_REFCURSOR | | | | |
| **Parameters** | **Name** | **Type** | | **Description** | |
| POUTCUR | Cursor | | Output parameter Cursor | |
| **Requirements** | Get all customer details and assign to pOutCur | | | | |
| **Exceptions** | **Type** | | | **Throw Exception Details** | |
| Other | | | 50000. Use value of error\_message() | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | | | **Type** | | **Return Type** |
| GET\_ALL\_PRODUCTS | | | Stored Procedure | | None |
| **Description** | Get all product details and assign to pOutCur | | | | |
| **Parameters** | **Name** | **Type** | | **Description** | |
| POUTCUR | Cursor | | Output parameter Cursor | |
| **Requirements** | Get all product details and return as a SYS\_REFCURSOR | | | | |
| **Exceptions** | **Type** | | | **Throw Exception Details** | |
| Other | | | 50000. Use value of error\_message() | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | | | **Type** | | **Return Type** |
| ADD\_LOCATION | | | Stored Procedure | |  |
| **Description** | Adds a new row to the location table | | | | |
| **Parameters** | **Name** | **Type** | | **Description** | |
| ploccode | nvarchar | | Location Code- format ‘locnn’ ‘nn’= int | |
| pminqty | Int | | Min qty | |
| pmaxqty | Int | | Max qty | |
| **Requirements** | Add a new row to the location table | | | | |
| **Exceptions** | **Type** | | | **Throw Exception Details** | |
| Duplicate primary key | | | 50180. Duplicate location ID | |
|  | CHECK\_LOCID\_LENGTH check failed | | | 50190. Location Code length invalid | |
|  | CHECK\_MINQTY\_RANGE check failed | | | 50200. Minimum Qty out of range | |
|  | CHECK\_MAXQTY\_RANGE check failed | | | 50210. Maximum Qty out of range | |
|  | CHECK\_MAXQTY\_GREATER\_MIXQTY check failed | | | 50220. Minimum Qty larger than   Maximum Qty | |
|  | Other | | | 50000. Use value of error\_message() | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | | | **Type** | | **Return Type** |
| ADD\_COMPLEX\_SALE | | | Stored Procedure | | None |
| **Description** | Adds a complex sale to the database | | | | |
| **Parameters** | **Name** | **Type** | | **Description** | |
| pcustid | Int | | CustomerId | |
| pprodid | Int | | ProductId | |
| pqty | Int | | Sale Qty | |
| pdate | Nvarchar | | Sale Date format yyyymmdd | |
| **Requirements** | Check if customer status is 'OK'. If not raise an exception.  Check if quantity value is valid. If not raise an exception.  Check if date value is valid. If not raise an exception.  Insert a new row into the Sale table.  The saleid value must be obtained from the SALE\_SEQ  Update both the Customer and Product SalesYTD values  Note: The YTD values must be increased by pqty \* the unit price  Calls UPD\_CUST\_SALES\_YTD and UPD\_PROD\_SALES\_YTD | | | | |
| **Exceptions** | **Type** | | | **Throw Exception Details** | |
| Sale Quantity range  1 - 999 | | | 50230. Sale Quantity outside valid range | |
| Invalid customer status  (status is not 'OK') | | | 50240. Customer status is not OK | |
| Invalid sale date | | | 50250. Date not valid | |
| No matching customer id found | | | 50260. Customer ID not found | |
| No matching product id found | | | 50270. Product ID not found | |
| Other | | | 50000. Use value of error\_message() | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | | | **Type** | | **Return Type** |
| GET\_ALLSALES | | | Stored Procedure | | None |
| **Description** | Get all customer details and return as a SYS\_REFCURSOR | | | | |
| **Parameters** | **Name** | **Type** | | **Description** | |
| POUTCUR | Cursor | | Output parameter Cursor | |
| **Requirements** | Get all complex sale details and assign to pOutCur | | | | |
| **Exceptions** | **Type** | | | **Throw Exception Details** | |
| Other | | | 50000. Use value of error\_message() | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | | | **Type** | | **Return Type** |
| COUNT\_PRODUCT\_SALES | | | Stored Procedure | | Int |
| **Description** | Count and return the int of sales with nn days of current date | | | | |
| **Parameters** | **Name** | **Type** | | **Description** | |
| pdays | int | | Count sales made within pdays of today's date | |
| **Requirements** | Count and return the int of sales in the SALES table with ***nn days*** of current date | | | | |
| **Exceptions** | **Type** | | | **Throw Exception Details** | |
| Other | | | 50000. Use value of error\_message() | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | | | **Type** | | **Return Type** |
| DELETE\_SALE | | | Stored Procedure | | Int |
| **Description** | Delete a row from the SALE table | | | | |
| **Parameters** | **Name** | **Type** | | **Description** | |
| **Requirements** | Determine the smallest saleid value in the SALE table. (use Select MIN()…)  If the value is NULL raise a No Sale Rows Found exception.  Otherwise delete a row from the SALE table with the matching sale id  Calls UPD\_CUST\_SALES\_YTD and UPD\_PROD\_SALES\_YTD so that the correct amount is subtracted from SALES\_YTD. You must calculate the amount using the PRICE in the SALE table multiplied by the QTY  This function must return the SaleID value of the Sale row that was deleted.  *(It is a bit unrealistic to delete a row with the smallest saleid. Normally you would ask a user to enter a sale id value. However this is difficult to do when testing with an anonymous block. So we will settle for smallest saleid in this assignment).* | | | | |
| **Exceptions** | **Type** | | | **Throw Exception Details** | |
| No Sale Rows Found | | | 50280. No Sale Rows Found | |
| Other | | | 50000. Use value of error\_message() | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | | | **Type** | | **Return Type** |
| DELETE\_ALL\_SALES | | | Stored Procedure | | None |
| **Description** | Delete a row from the SALE table | | | | |
| **Parameters** | **Name** | **Type** | | **Description** | |
| **Requirements** | Delete all rows in the SALE table  Set the Sales\_YTD value to zero for all rows in the Customer and Product tables | | | | |
| **Exceptions** | **Type** | | | **Throw Exception Details** | |
| Other | | | 50000. Use value of error\_message() | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | | | **Type** | | **Return Type** |
| DELETE\_CUSTOMER | | | Stored Procedure | |  |
| **Description** | Delete a row from the Customer table | | | | |
| **Parameters** | **Name** | **Type** | | **Description** | |
| pCustid | int | | Customer Id | |
| **Requirements** | Delete a customer with a matching customer id  If ComplexSales exist for the customer, replace the default error code with a custom made exception to handle this error & raise the exception below | | | | |
| **Exceptions** | **Type** | | | **Throw Exception Details** | |
| No matching customer id found | | | 50290. Customer ID not found | |
| Customer has child complexsales rows | | | 50300. Customer cannot be deleted as sales exist | |
| Other | | | 50000. Use value of error\_message() | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | | | **Type** | | **Return Type** |
| DELETE\_PRODUCT | | | Stored Procedure | |  |
| **Description** | Delete a row from the Product table | | | | |
| **Parameters** | **Name** | **Type** | | **Description** | |
| pProdid | int | | Product Id | |
| **Requirements** | Delete a product with a matching Product id  If ComplexSales exist for the customer, Oracle would normally generate a 'Child Record Found' error (error code -2292). Instead,  Create a custom made exception to handle this error & raise the exception below | | | | |
| **Exceptions** | **Type** | | | **Throw Exception Details** | |
| No matching Product id found | | | 50310. Product ID not found | |
| Product has child complexsales rows | | | 50320. Product cannot be deleted as sales exist | |
| Other | | | 50000. Use value of error\_message() | |