Hands-On Assignments Part I

Assignment 5-1: Creating a Procedure

Use these steps to create a procedure that allows a company employee to make corrections to a product's assigned name. Review the BB_PRODUCT table and identify the PRODUCT NAME and PRIMARY KEY columns. The procedure needs two `IN` parameters to identify the product ID and supply the new description. This procedure needs to perform only a DML action, so no `OUT` parameters are necessary.

1. In SQL Developer, create the following procedure:

```
CREATE OR REPLACE PROCEDURE prod_name_sp
  (p_prodid IN bb_product.idproduct%TYPE,
   p_descrip IN bb_product.description%TYPE)


  IS
BEGIN
  UPDATE bb_product
    SET description = p_descrip
    WHERE idproduct = p_prodid;
  COMMIT;
END;
```

2. Before testing the procedure, verify the current description value for product ID 1 with `SELECT * FROM bb_product;`.
3. Call the procedure with parameter values of `1` for the product ID and `CapressoBar Model #388` for the description.
4. Verify that the update was successful by querying the table with `SELECT * FROM bb_product;`.

Assignment 5-2: Using a Procedure with `IN` Parameters

Follow these steps to create a procedure that allows a company employee to add a new product to the database. This procedure needs only `IN` parameters.

1. In SQL Developer, create a procedure named `PROD_ADD_SP` that adds a row for a new product in the BB_PRODUCT table. Keep in mind that the user provides values for the product name, description, image filename, price, and active status. Address the input values or parameters in the same order as in the preceding sentence.
2. Call the procedure with these parameter values: `('Roasted Blend', 'Well-balanced mix of roasted beans, a medium body', 'roasted.jpg',9.50,1)`.
3. Check whether the update was successful by querying the BB_PRODUCT table.

Assignment 5-3: Calculating the Tax on an Order

Follow these steps to create a procedure for calculating the tax on an order. The BB_TAX table contains states that require submitting taxes for Internet sales. If the state isn't listed in the table, no tax should be assessed on the order. The shopper's state and basket subtotal are the inputs to the procedure, and the tax amount should be returned.

1. In SQL Developer, create a procedure named `TAX_COST_SP`. Remember that the state and subtotal values are inputs to the procedure, which should return the tax amount. Review the BB_TAX table, which contains the tax rate for each applicable state.
2. Call the procedure with the values `VA` for the state and `$100` for the subtotal. Display the tax amount the procedure returns. (It should be $4.50.)

Assignment 5-4: Updating Columns in a Table

After a shopper completes an order, a procedure is called to update the following columns in the BASKET table: ORDERPLACED, SUBTOTAL, SHIPPING, TAX, and TOTAL. The value 1 entered in the ORDERPLACED column indicates that the shopper has completed an order. Inputs to the procedure are the basket ID and amounts for the subtotal, shipping, tax, and total.

1. In SQL Developer, create a procedure named `BASKET_CONFIRM_SP` that accepts the input values specified in the preceding description. Keep in mind that you're modifying an existing row of the BB_BASKET table in this procedure.

2. Enter the following statements to create a new basket containing two items:

```
INSERT INTO BB_BASKET (IDBASKET, QUANTITY, IDSHOPPER,
                       ORDERPLACED, SUBTOTAL, TOTAL,
                       SHIPPING, TAX, DTCREATED, PROMO)
    VALUES (17, 2, 22, 0, 0, 0, 0, 0, '28-FEB-12', 0);
INSERT INTO BB_BASKETITEM (IDBASKETITEM, IDPRODUCT, PRICE,
                           QUANTITY, IDBASKET, OPTION1, OPTION2)
    VALUES (44, 7, 10.8, 3, 17, 2, 3);
INSERT INTO BB_BASKETITEM (IDBASKETITEM, IDPRODUCT, PRICE,
                           QUANTITY, IDBASKET, OPTION1, OPTION2)
    VALUES (45, 8, 10.8, 3, 17, 2, 3);
```

3. Type and run `COMMIT;` to save the data from these statements.
4. Call the procedure with the following parameter values: `17, 64.80, 8.00, 1.94, 74.74`. As mentioned, these values represent the basket ID and the amounts for the subtotal, shipping, tax, and total.
5. Query the BB_BASKET table to confirm that the procedure was successful:

```
SELECT subtotal, shipping, tax, total, orderplaced
  FROM bb_basket
  WHERE idbasket = 17;
```

Assignment 5-5: Updating Order Status

Create a procedure named `STATUS_SHIP_SP` that allows an employee in the Shipping Department to update an order status to add shipping information. The BB_BASKETSTATUS table lists events for each order so that a shopper can see the current status, date, and comments as each stage of the order process is finished. The IDSTAGE column of the BB_BASKETSTATUS table identifies each stage; the value 3 in this column indicates that an order has been shipped.

The procedure should allow adding a row with an IDSTAGE of 3, date shipped, tracking number, and shipper. The `BB_STATUS_SEQ` sequence is used to provide a value for the primary key column. Test the procedure with the following information:

```
Basket # = 3
Date shipped = 20-FEB-12
Shipper = UPS
Tracking # = ZW2384YXK4957
```

Assignment 5-6: Returning Order Status Information

Create a procedure that returns the most recent order status information for a specified basket. This procedure should determine the most recent ordering-stage entry in the BB_BASKETSTATUS table and return the data. Use an `IF` or `CASE` clause to return a stage description instead of an IDSTAGE number, which means little to shoppers. The IDSTAGE column of the BB_BASKETSTATUS table identifies each stage as follows:

- 1—Submitted and received
- 2—Confirmed, processed, sent to shipping
- 3—Shipped
- 4—Cancelled
- 5—Back-ordered

The procedure should accept a basket ID number and return the most recent status description and date the status was recorded. If no status is available for the specified basket ID, return a message stating that no status is available. Name the procedure `STATUS_SP`. Test the procedure twice with the basket ID **4** and then **6.**

Assignment 5-7: Identifying Customers

Brewbean's wants to offer an incentive of free shipping to customers who haven't returned to the site since a specified date. Create a procedure named **PROMO_SHIP_SP** that determines who these customers are and then updates the BB_PROMOLIST table accordingly. The procedure uses the following information:

- *Date cutoff*—Any customers who haven't shopped on the site since this date should be included as incentive participants. Use the basket creation date to reflect shopper activity dates.
- *Month*—A three-character month (such as APR) should be added to the promotion table to indicate which month free shipping is effective.
- *Year*—A four-digit year indicates the year the promotion is effective.
- promo_flag—1 represents free shipping.

The BB_PROMOLIST table also has a USED column, which contains the default value N and is updated to Y when the shopper uses the promotion. Test the procedure with the cutoff date 15-FEB-12. Assign free shipping for the month APR and the year 2012.

Assignment 5-8: Adding Items to a Basket

As a shopper selects products on the Brewbean's site, a procedure is needed to add a newly selected item to the current shopper's basket. Create a procedure named **BASKET_ADD_SP** that accepts a product ID, basket ID, price, quantity, size code option (1 or 2), and form code option (3 or 4) and uses this information to add a new item to the BB_BASKETITEM table. The table's PRIMARY KEY column is generated by BB_IDBASKETITEM_SEQ. Run the procedure with the following values:

- Basket ID—14
- Product ID—8
- Price—10.80
- Quantity—1
- Size code—2
- Form code—4

Assignment 5-9: Creating a Logon Procedure

The home page of the Brewbean's Web site has an option for members to log on with their IDs and passwords. Develop a procedure named **MEMBER_CK_SP** that accepts the ID and password as inputs, checks whether they make up a valid logon, and returns the member name and cookie value. The name should be returned as a single text string containing the first and last name.

The head developer wants the number of parameters minimized so that the same parameter is used to accept the password and return the name value. Also, if the user doesn't enter a valid username and password, return the value **INVALID** in a parameter named **p_check**. Test the procedure using a valid logon first, with the username **rat55** and password **kile**. Then try it with an invalid logon by changing the username to **rat**.