1. (3 points) Create a sequence to automatically generate the values for each of the three attributes pur#, sup# and log# when new tuples are inserted into the corresponding table. For each of these primary key attributes, if its data type is number(n), its values should have n digits (this can be achieved by starting a sequence with a value of n digits). Implement a different sequence for each of three attributes.

Usage - To generate purchase id

Objective - Automatic generation of primary key of Purchases table

create sequence pur# start with 1000 increment by 1;

Usage - To generate sup id

Objective - Automatic generation of primary key of Supplies table

create sequence sup# start with 100 increment by 1;

Usage - To generate log id

Objective - Automatic generation of primary key of Log table

create sequence log# start with 1000 increment by 1;

2. (4 points) Create a procedure or function to show the tuples in each table. If you just want to run your package in the SQL\*Plus environment, it is sufficient to create a procedure for each table. As an example, you can implement a procedure, say show\_employees, in your package to show all employees in the employees table. You need to implement 8 procedures or functions, one for each table. Alternatively, if you want to get the results to your JDBC interface program, you should create a function and use ref cursor (see sample program 3 for the use of ref cursor).

Usage – Procedure get details of all employees

Objective – To get employees report

PROCEDURE show\_employees(out\_prc OUT sys\_refcursor);

Usage – Procedure get details of all customers

Objective – To get customers report

PROCEDURE show\_customers(out\_prc OUT sys\_refcursor);

Usage – Procedure get details of all products

Objective – To get products report

PROCEDURE show\_products(out\_prc OUT sys\_refcursor);

Usage – Procedure get details of all discounts

Objective – To get discount list

PROCEDURE show\_discounts(out\_prc OUT sys\_refcursor);

Usage – Procedure get details of all suppliers

Objective – To get suppliers report

PROCEDURE show\_suppliers(out\_prc OUT sys\_refcursor);

Usage – Procedure get details of all supplied products by particular supplier

Objective – To get check all products supplied by supplier.

PROCEDURE show\_supplies(s\_id IN CHAR, out\_prc OUT sys\_refcursor);

Usage – Procedure get details of all purchases

Objective – To get purchase report

PROCEDURE show\_purchases(out\_prc OUT sys\_refcursor);

3. (3 points) Create a function, say purchase\_saving(pur#), to report the total saving of any purchase for any given pur# (as an in parameter).

Usage – Function get saving details of particular purchase

Objective – To check the total amount saved in selected purchase.

FUNCTION purchase\_saving(pur# IN CHAR) RETURN NUMBER;

4. (4 points) Create a procedure to report the monthly sale activity information for any given employee. For example, you can use a procedure, say monthly\_sale\_activities(employee\_id), for this operation. For the given employee id (an in parameter), you need to report the employee id, employee name, the month (the first three letters of the month, e.g., FEB for February), the year (4 digits), the total number of times the employee made sales (i.e., the number of purchases that involve the employee) each month, the total quantity sold by the employee each month, and the total dollar amount sold by the employee each month. Only need to list the information for those months during which the given employee has actual made sales.

Usage – Procedure get monthly report for given employee (i.e number of sales, total amount of products sold)

Objective – To get montly sale report for particular employee

PROCEDURE monthly\_sale\_activities(employee\_id IN CHAR, out\_prc OUT sys\_refcursor);

5. (3 points) Create a procedure for adding tuples to the Customers table. You may use a procedure, say add\_customer(c\_id, c\_name, c\_telephone#), in your package to add a tuple into the Customer table, where c\_id, c\_name, and c\_telephone# are all in parameters of the procedure. Note that the initial values of visits\_made and last\_visit\_date of any newly added customer should be generated by your procedure. Specifically, visits\_made should be given an initial value of 1 and the initial value for last\_visit\_date can be generated by sysdate.

Usage – Procedure to add customer in customer table

Objective – To add customer details

PROCEDURE add\_customer(c\_id IN CHAR, c\_name IN VARCHAR, c\_telephone# IN CHAR);

6. (10 points) Create triggers that can add tuples to the logs table automatically whenever certain events happen. In this project, the following events need to be tracked: (1) insert a tuple into the Customers table; (2) update the last\_visit\_date attribute of the Customers table; (3) insert a tuple into the Purchases table; (4) update the qoh attribute of the Products table; and (5) insert a tuple into the Supplies table. When a tuple is added to the logs table due to the first event, the table\_name should be “customers”, the operation should be “insert” and the tuple\_pkey should be the cid of the newly inserted customer. When a tuple is added to the logs table due to the second event, the table\_name should be “customers”, the operation should be “update” and the tuple\_pkey should be the cid of the affected customer. When a tuple is added to the logs table due to the third event, the table\_name should be “purchases”, the operation should be “insert” and the tuple\_pkey should be the pur# of the newly inserted purchase. When a tuple is added to the logs table due to the fourth event, the table\_name should be “products”, the operation should be “update” and the tuple\_pkey should be the pid of the affected product. When a tuple is added to the logs table due to the fifth event, the table\_name should be “supplies”, the operation should be “insert” and the tuple\_key should be the sup# of the newly inserted supply. You need to implement five triggers for this task, one for each event.

Usage – Triggers are created wherever applicable.

Objective – To log events in database;

Following Triggers are created:

TRIGGER insertcustomer

To create log entry when new customer is added;

TRIGGER updatecustomervisits

To create log entry when customer visits retail business management system.

TRIGGER insertpurchase

To create log entry when new purchase is made by customer

TRIGGER updateqoh

To create log entry when quantity of product is modified.

TRIGGER insertsupply

To create log entry when product is supplied by supplier.

7. (15 points) Create a procedure for adding tuples to the Purchases table. You may use a procedure, say add\_purchase(e\_id, p\_id, c\_id, pur\_qty), in your package for this purpose, where e\_id, p\_id, c\_id and pur\_qty are all in parameters of the procedure. Note that the pur# of any newly added purchase should be automatically generated by your sequence. In addition, total\_price should be computed based on the data in the database automatically and ptime should be automatically generated by sysdate.

Before a tuple is added into the table, your procedure needs to make sure that, for the involved product, the quantity to be purchased is equal to or smaller than the quantity on hand (qoh). Otherwise, an appropriate message should be displayed (e.g., “Insufficient quantity in stock.”) and the purchase request should be rejected.

After adding a tuple to the Purchases table, the qoh column of the Products table should be modified accordingly, that is, the qoh of the product involved in the purchase should be reduced by the quantity purchased. If the purchase causes the qoh of the product to be below qoh\_threshold, your procedure should perform the following tasks:

(a) Print a message saying that the current qoh of the product is below the required threshold and new supply is required

(b) Automatically order supply for the product (i.e., add a new tuple to the Supplies table): the sup# is automatically generated by a sequence, the pid of the new supply is the pid of the product involved in the purchase, the sid of the new supply is the sid of a supplier who has supplied this product before (there should be such information in the current Supplies table; if multiple suppliers have supplied this product before, use the supplier with the smallest sid), the quantity of the new supply should be computed using 10 + M + qoh, where M is the minimum value for quantity such that M + qoh > qoh\_threshold, and use sysdate for sdate,

(c) Increase qoh of the product by the quantity ordered.

(d) Print another message showing the new value of the qoh of the product.

In addition, the insertion of the new tuple into the Purchases table may cause the visits\_made of the involved customer to be increased by one if the purchase is made on a new date and the last\_visit\_date may also have updated accordingly.

Use triggers to implement the update of qoh, the printing (displaying) of the messages, the insertion of new supplies to the Supplies table, and the updates of visits\_made and last\_visit\_date.

Usage – Procedure to add purchase details

Objective – To check if enough quantity is available, if new purchase makes quantity below threshold order new supply, change last visit of customer, show error if any.

PROCEDURE add\_purchase(e\_id IN CHAR, p\_id IN CHAR, c\_id IN CHAR, pur\_qty IN

NUMBER);

8. (8 points) Create a procedure for deleting tuples from the Purchases table to simulate returning purchases by customers. You may use a procedure, say delete\_purchase(pur#), in your package for this purpose, where pur# is an in parameter of the procedure. After the deletion is completed, all data in the database should be made consistent by triggers. Specifically, quantity on hand (qoh) of the returned product in the Products table, and visits\_made and last\_visit\_date should be adjusted accordingly. The visit to return a purchase is also considered a visit to the retail business.

Usage – Procedure to delete purchase.

Objective – To delete purchase from purchases, return all values to stock, update customers visit.

PROCEDURE delete\_purchase(pur# IN CHAR);

Few local functions created in package are as follows:

FUNCTION get\_total\_employees RETURN NUMBER;

FUNCTION get\_total\_customers RETURN NUMBER;

FUNCTION get\_total\_products RETURN NUMBER;

FUNCTION get\_total\_suppliers RETURN NUMBER;

FUNCTION get\_total\_supplies(s\_id IN CHAR) RETURN NUMBER;

FUNCTION get\_total\_purchases RETURN NUMBER;