**Documentation (pl-sql and jdbc codes with inline comments)**

The honesty statement: “We have done this assignment completely on our own. We have not copied it, nor have we given our solution to anyone else. We understand that if we are involved in plagiarism or cheating, we will have to sign an official form that we have cheated and that this form will be stored in our official university records. We also understand that we will receive a grade of 0 for the involved assignment and our grades will be reduced by one level (e.g., from A to A- or from B+ to B) for our first offense, and that we will receive a grade of “F” for the course for any additional offense of any kind.”

Signed by: Neha Joshi, Ramya Ravikanti, Renee H Choong

Team members:

1)Neha M Joshi

njoshi3@binghamton.edu

2)Ramya Ravikanti

[rravika1@binghamton.edu](mailto:rravika1@binghamton.edu)

3)Renee H Choong

[rchoong1@binghamton.edu](mailto:rchoong1@binghamton.edu)

Tables used:

Employees(eid, name, telephone#, email)

Customers(cid, name, telephone#, visits\_made, last\_visit\_date)

Products(pid, name, qoh, qoh\_threshold, regular\_price, discnt\_rate)

Purchases(pur#, eid, pid, cid, pur\_date, qty, unit\_price, total, saving)

Logs(log#, user\_name, operation, op\_time, table\_name, tuple\_pkey

Code with inline comments:

Create two sequences – for pur# and the other will be used to automatically generate unique values for log#. The sequence for pur# needs to generate numbers that start with 100001 and the sequence for log# needs to generate numbers that start with 1001[task 1]

drop sequence pur#\_seq ## drop the sequence if there is a sequence prior with same name

create sequence pur#\_seq ## creating sequence

start with 10001 ##specifying the start point

increment by 1## incrementing the values of sequence by1

maxvalue 999999## the maximum number of values in sequence

nocache## values of the **sequence** are not preallocated

nocycle;##the sequence cannot generate more values after reaching its maximum or minimum value

drop sequence log#\_seq## drop the sequence if there is a sequence prior with same name

create sequence log#\_seq## creating sequence

start with 1001##specifying the start point

increment by 1## incrementing the values of sequence by1

maxvalue 99999## the maximum number of values in sequence

nocache## values of the **sequence** are not preallocated

nocycle;##the sequence cannot generate more values after reaching its maximum or minimum value

Create procedure to display the tuples of the table[task 2]

/to display the tuples of purchases

create or replace procedure show\_purchases is ## creating procedure show\_procedures

2 pur\_id purchases.pur%type; ## parameter takes the same data type as that of the column in the table

3 pur\_date purchases.pur\_date%type;

4 pur\_qty purchases.qty%type;

5 pur\_unit\_price purchases.unit\_price%type;

6 pur\_total purchases.total%type;

7.pur\_savings purchases.saving%type;

8.p\_eid purchases.eid%type;

9.p\_pid purchases.pid%type;

10.p\_cid purchases.cid%type;

11 cursor c is## defining what cursor is

12 select \* from customers;# selecting all tuples from table

13 begin## begin the SQL statements

14 open c;##start to evaluate the command in the cursor c

15 loop#declaration of the loop

16 fetch c into pur\_id,pur\_date,pur\_qty;pur\_unit\_price,pur\_total,pur\_savings,p\_eid,p\_pid,p\_cid## fetching the tuples

17 dbms\_output.put\_line(pur\_id||' '||pur\_date||' '||pur\_qty||' '||pur\_unit\_price||' '||pur\_total||' ' ||pur\_savings|| ' ' ||p\_eid||' ' ||p\_pid||' ' ||p\_cid||); ## the format of the output required

18 exit when c%notfound;##exit from the loop when the cursor c is not found

19 end loop;## end the loop

20 close c;## close the cursor

21 end;## end the procedure

/to show the tuples of employees

create or replace procedure show\_employeesis ## creating procedure show\_employees

2 e\_id employees.eid%type; ## parameter

takes the same data type as that of the column in the table

3 e\_name employees.name%type;

4 e\_tel employees.telephone#%type;

5 e\_email employees.email%type;

6 pur\_total purchases.total%type;

7 cursor c is ## defining what cursor is

8 select \* from employees;# selecting all tuples from table

9 begin## begin the SQL statements

10 open c;##start to evaluate the command in the

cursor c

11 loop#declaration of the loop

12 fetch c into e\_id,e\_name,e\_tel,e\_email## fetching the

tuples

13 dbms\_output.put\_line(e\_id||' '||e\_name||' '||e\_tel||'

'||e\_email||); ## the format of the output required

14 exit when c%notfound;##exit from the loop when the

cursor c is not found

15 end loop;## end the loop

16 close c;## close the cursor

17 end;## end the procedure

/ to show the tuples of customers

create or replace procedure show\_customers is ## creating the procedure show\_customers

2 c\_id customers.cid%type; ; ## parameter takes the same data type as that of

the column in the table

3 c\_name [customers.name](http://customers.name/)%type;

4 c\_tel customers.telephone#%type;

5 c\_visits customers.visits\_made%type;

6 c\_last customers.last\_visit\_date%type;

7 cursor c is## defining what cursor is

8 select \* from customers; ;# selecting all tuples from table

9 begin##

begin the SQL statements

10 open c; ;##start

to evaluate the command in the cursor c

11 loop##declaration of the loop

12 fetch c into c\_id,c\_name,c\_tel,c\_visits,c\_last; ## fetching the tuples

13 dbms\_output.put\_line(c\_id||'

'||c\_name||' '||c\_tel||' '||c\_visits||' '||c\_last||); ## the format of the output required

14 exit when c%notfound; ;##exit from the loop when the cursor c is not found

15 end loop;## end the loop

16 close c;## close the cursor

17 end;## end the procedure

18 /

/to show the tuples of logs

create or replace procedure show\_logs is ## creating the procedure show\_customers

2 l\_log logs.log#%type; ; ## parameter takes the same data type as that of

the column in the table

3 l\_name [logs.user\_name](http://customers.name/)%type;

4 l\_opr logs.operation%type;

5 l\_op\_time logs.op\_time%type;

6 l\_table\_name logs.table\_name%type;

7 l\_tuple\_key logs.tuple\_key%type

8 cursor c is##

defining what cursor is

9 select \* from logs; ;# selecting all tuples from table

10 begin##

begin the SQL statements

11 open c; ;##start to

evaluate the command in the cursor c

12 loop##declaration of the loop

13 fetch c into l\_log,l\_name,l\_opr,c\_l\_op\_time,l\_table\_name,l\_tuple\_key; ## fetching the tuples

14 dbms\_output.put\_line(l\_log||' '||l\_name||'

'||l\_opr||' '||l\_op\_time||' '||l\_table\_name||' '||l\_tuple\_key ||); ## the format of the output required

15 exit when c%notfound; ;##exit from the loop when the cursor c is not found

16 end loop;## end the loop

17 close c;## close the cursor

18 end;## end the procedure

19 /

/ to show the tuples of products

create or replace procedure show\_products is ## creating the procedure show\_customers

2 p\_id products.pid%type; ; ## parameter takes the same data type as that of

the column in the table

3 p\_name [products.name](http://customers.name/)%type;

4 p\_qoh products.qoh%type;

5 p\_qoh\_threshold products.qoh\_threshold%type;

6 p\_regular\_price products.regular\_price%type;

7 p\_discnt\_rate products.discnt\_rate%type;

8 cursor c is##

defining what cursor is

9 select \* from products; ;# selecting all tuples from table

10 begin##

begin the SQL statements

11 open c; ;##start to

evaluate the command in the cursor c

12 loop##declaration of the loop

13 fetch c into p\_id,p\_name,p\_qoh,p\_qoh\_threshold,p\_regular\_price,p\_discnt\_rate; ## fetching the tuples

14 dbms\_output.put\_line(p\_id||' '||p\_name||'

'||p\_qoh||' '||p\_qoh\_threshold||' '||p\_regular\_price||' '||p\_discnt\_rate||); ## the format of the output required

15 exit when c%notfound; ;##exit from the loop when the cursor c is not found

16 end loop;## end the loop

17 close c;## close the cursor

18 end;## end the procedure

19 /

/ Create a procedure, say **purchases\_made(cid)**, that, for a given cid of a customer as an input parameter, will return the name of the customer as well as every purchase (output pid, pur\_date, qty, unit\_price, and total) the customer has made[task 3]

SQL> create or replace procedure purchases\_made(customer\_id in purchases.cid%type)is ## creating a procedure purchases\_made

2 begin##begin the SQL statements

3 dbms\_output.put\_line('cid' || ' ,' || 'name' || ', ' || 'pid' ||', ' || 'pur\_date' || ',' || 'qty' || ',' || 'unit\_price' || ',' || 'total');##required output format

4 for c in(select \* from (select cid,name from customers c where cid=customer\_id)a,(select pid,pur\_date,qty,unit\_price,total from purchases p where cid=customer\_id)b)##selecting tuples from purchases and customers that will be returned as required output

5 loop##declaration of loop

6 dbms\_output.put\_line(c.cid||','||c.name|| ',' ||c.pid ||',' || c.pur\_date|| ',' ||c.qty || ',' ||c.unit\_price || ',' ||c.total);##required output

7 exception

when NO\_DATA\_FOUND then

dbms\_output.put\_line('Invalid pur#.');

8 end loop;##loop ends

8 end;## procedure ends

9 /

/Create a function, say **number\_customers(pid)**, to report the number of customers who have purchased the product identified by the pid (which is an in parameter of the function). [task4]

SQL> create or replace function number\_customers(cust\_pid purchases.pid%type) return number is cust\_num number;## create function number\_customers

2 begin##loop begins

3 select count(cid) into cust\_num from purchases where pid=cust\_pid;## gives us the count of customers who purchased the products that have pid

4 exception

when NO\_DATA\_FOUND then## declaration of exceptions

dbms\_output.put\_line('Invalid pur#.');##required output/message to be displayed

number\_customers:= -1;

return number\_customers;#3 returns the numbers of customers

6 end;## function ends

/Create a procedure for adding tuples to the Customers table[task 5]

SQL> create or replace procedure add\_customer(c\_id in customers.cid%type, c\_name in customers.name%type, c\_telephone# in customers.telephone#%type) is## creating procedure procedure

2 begin## sql query begin declaration

3 insert into customers values (c\_id, c\_name, c\_telephone#, 1, sysdate);##inserting tuples c\_id,c\_name,c\_telephone#,1. visits\_made is given an initial value of 1 and the initial value for last\_visit\_date is generated by sysdate.

4 end;##end of the procedure

5 /

/Create a procedure for adding tuples to the Purchases table[task 6]

 create or replace  procedure add\_purchases(e\_id in purchases.eid%type, p\_id in purchases.pid%type, c\_id in purchases.cid%type, pur\_qty in purchases.qty%type,pur\_unit\_price in purchases.unit\_price%type)is ## creating procedure add\_purchase  
  2  pur\_total purchases.total%type;## ## parameter takes the same data type as that of

the column in the table

3   pur\_saving purchases.saving%type;  
  4  product\_qoh products.qoh%type;  
  5  product\_qoh\_thresh products.qoh%type;  
  6  temp\_eid purchases.eid%type;# declaring the temp\_eid variable which have same data as purcahes.eid for further exception purposes  
  7   temp\_cid purchases.cid%type; ;# declaring the temp\_cid variable which have same data as purcahes.cid for further exception purposes  
  8  temp\_pid purchases.pid%type; ;# declaring the temp\_pid variable which have same data as purcahes.pid for further exception purposes  
  9  error\_msg varchar2(1000);## error message data type  
 10  begin  
 11  begin##declaration of begin for the SQL statements  
 12  select distinct eid into temp\_eid from purchases where eid=e\_id;#will select the eid into temp\_eid which are in eid of purchases table  
 13  if(temp\_eid!=e\_id)then ##if the given input e\_id is not in temp\_eid then following error msg will be displayed  
 14  error\_msg:='INVALID EID IS GIVEN';## error msg to be displayed for above condition  
 15  end if;#the if loop ends  
 16   exception#declaration of exceptions  
 17  when no\_data\_found then #type of exception  
 18  RAISE\_APPLICATION\_ERROR(-20001,'INVALID EID IS GIVEN');#following error will be raised showing above as msg  
 19  return;  
 20  end;  
 21  begin  
 22  select distinct  pid into temp\_pid from purchases where pid=p\_id;# will select the pid into temp\_pid which are in pid of purchases table  
  
 23  if(temp\_pid!=p\_id)then# if the given input p\_id is not in temp\_pid then following error msg will be displayed  
  
 24   error\_msg:='INVALID PID IS GIVEN';# error msg to be displayed for above condition  
 25  end if;##end the if case  
 26  exception#declaration of exception  
 27  when no\_data\_found then##type of exception  
 28  RAISE\_APPLICATION\_ERROR(-20002,'INVALID PID IS GIVEN');## following error will be raised showing above as msg  
  
 29  return;  
 30  end;  
 31  begin  
 32  select distinct cid into temp\_cid from purchases where cid=c\_id; ;# will select the cid into temp\_cid which are in cid of purchases table  
  
  
 33  if(temp\_cid!=c\_id)then### if the given input c\_id is not in temp\_cid then following error msg will be displayed  
  
 34  error\_msg:='INVALID CID IS GIVEN';##the msg to be displayed  
 35  end if;##end the if case  
 36  exception#declaration of exception

 37   when no\_data\_found then## when exception occurs   
 38  RAISE\_APPLICATION\_ERROR(-20003,'INVALID CID IS GIVEN');##the error msg to be displayed as  
 39  return;  
 40  end;  
 41  select qoh into product\_qoh from products where pid=p\_id;#selecting qoh into product\_qoh  
 42  select p.qty\*(pr.regular\_price-p.unit\_price) into pur\_saving from purchases p,products pr where p.pid=p\_id and  p.pid=pr.pid;#calculating savings into pur\_savings  
 43  select qoh\_threshold into product\_qoh\_thresh from products where pid=p\_id;#selecting qoh\_threshold into product\_qoh\_threshold  
 44  if(pur\_qty>product\_qoh)then#if purchase quantity is more than product quantity on hand  
 45  dbms\_output.put\_line('Insufficient quantity in stock');#if above case is true then following msg will be displayed  
 46  else#else case if if condition is not true  
 47  insert into purchases values(pur#\_seq.nextval,e\_id,p\_id,c\_id,sysdate,pur\_qty,pur\_unit\_price,pur\_qty\*pur\_unit\_price,pur\_saving);#it will insert into purchase if if condition is false  
 48  end if;#end the if case  
 49  if(product\_qoh<product\_qoh\_thresh)then#if product of quantity on hand is less than threshold qoh then   
 50  dbms\_output.put\_line('new supply required');#if above if condition is true it will display this as msg  
 51  product\_qoh:=product\_qoh\_thresh+10##if above if condition is true it will u[date the new qoh  
 52  update products set qoh=product\_qoh where pid=p\_id;## update the new qoh is our qoh now  
  
 53  dbms\_output.put\_line('the new qoh is'||product\_qoh);##display the updated qoh  
 54  end if;##end the if case

 55  end;##end the procedure  
 56  /

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

create or replace trigger qoh\_update##creating a trigger  
  2  after insert on purchases  
  3  for each row  
  4  declare  
  5  pro\_qty products.qoh%type; ;##parameter takes the same data type as that of

the column in the table

  6  pro\_thresh products.qoh\_threshold%type;  
  7  begin  
  8  update products set qoh = qoh - :new.qty where pid = :new.pid;#updating the qoh  
  9  select qoh, qoh\_threshold into pro\_qty, pro\_thresh from products where pid = :new.pid;#selecting the qoh,qoh\_threshold into pro\_qty and pro\_thresh respectively  
 10  if(pro\_qty < pro\_thresh) then #if qoh is greater than qoh\_threshold then  
 11  dbms\_output.put\_line('The current qoh of the product is below the required threshold and new supply is required');##if above if condition is true it will show display this msg  
 12  pro\_qty:=pro\_thresh+10;#update the qoh as threshold+10  
 13  update products set qoh = pro\_qty where pid = :new.pid;# update qoh as pro\_qty  
 14  dbms\_output.put\_line('After supply, QOH is: ' ||pro\_qty );#print the new qoh after supply  
 15  end if;#end the if case  
 16  end;#end the trigger  
 17  /

/Create triggers that can add tuples to the logs table [task 7]

1

create or replace trigger customers\_insert## creating trigger customers\_insert trigger that can add tuples to the logs table when certain events happen

2 after insert on customers## trigger will work after there is insertion in

3 for each row

4 begin##declaration for SQL query begin

5 insert into logs values(seq\_log#.nextval,user,'insert',sysdate,'customers',:new.cid);##add tuples to logs table

6 end;## end of the SQL query

7 /

7.2 SQL> create or replace trigger customers\_update\_last\_visit## create a trigger customers\_update\_last\_visit when ther is update in last\_visit \_date attribute

2 after update of last\_visit\_date on customers## trigger will work after last\_visit\_date is updated

3 for each row

4 begin##declaration for begin for SQL query

5 insert into logs values(seq\_log#.nextval,user,'update',sysdate,'customers',:old.cid);##add tuples to logs table

6 end;## end of the SQL query

7 /

7.3 SQL> create or replace trigger customers\_update\_vists\_made##creating trigger customers\_update

2 after update of visits\_made on customers##trigger will work after visits\_made is updated

3 for each row

4 begin## declaration for begin of SQL query

5 insert into logs values(seq\_log#.nextval,user,'update',sysdate,'customers',:old.cid);##add tuples into logs table

6 end;

7 /

7.4 create or replace trigger purchases\_insert## create trigger purchases\_insert

2 after insert on purchases##trigger will work after there is insertion of tuples in purchases table

3 for each row

4 begin##SQL query begins

5 insert into logs values(seq\_log#.nextval,user,'insert',sysdate,'purchases',:new.pur#);##add tuples to logs table

6 end;

7 /

7.5 create or replace trigger products\_update## create trigger products\_update

2 after update of qoh on products##trigger will work after there is update of qoh in products

3 for each row

4 begin##SQL query begins

5 insert into logs values(seq\_log#.nextval,user,'update',sysdate,'products',:old.pid);

##inserttuples into logs table

6 end;

7 /

JDBC code:

import java.sql.\*; # importing the packages

import oracle.jdbc.\*;

import java.math.\*;

import java.io.\*;

import java.awt.\*;

import oracle.jdbc.pool.OracleDataSource;

public class cust { #class declaration

public static void main (String args []) throws SQLException {

try

{

//Connection to Oracle server. Need to replace username and

//password by your username and your password. For security

//consideration, it's better to read them in from keyboard.

OracleDataSource ds = new oracle.jdbc.pool.OracleDataSource();

ds.setURL("jdbc:oracle:thin:@castor.cc.binghamton.edu:1521:acad111");

Connection conn = ds.getConnection("rravika1", "Ramfam21121997");

// Query

Statement stmt = conn.createStatement ();

// Save result

ResultSet rset;

rset = stmt.executeQuery ("SELECT \* FROM customers");

// Print

while (rset.next ()) {

System.out.print (rset.getString (1)+" ");

System.out.print (rset.getString (2)+" ");

System.out.print (rset.getString (3)+" ");

System.out.println (rset.getString (4)+" ");

}

//Input sid from keyboard

BufferedReader readKeyBoard;

String icid;

readKeyBoard = new BufferedReader(new InputStreamReader(System.in));

System.out.print("Please enter CID:");

icid = readKeyBoard.readLine();

ResultSet rset2;

rset2=stmt.executeQuery ("Select \* from customers where cid='"+icid +"'");

// Print

while (rset2.next ()) {

System.out.print (rset2.getString (1)+" ");

System.out.print (rset2.getString (2)+" ");

System.out.print (rset2.getString (3)+" ");

System.out.println (rset2.getString (4)+" ");

}

//close the result set, statement, and the connection

rset.close();

rset2.close();

stmt.close();

conn.close();

}

catch (SQLException ex) { System.out.println ("\n\*\*\* SQLException caught \*\*\*\n");}

catch (Exception e) {System.out.println ("\n\*\*\* other Exception caught \*\*\*\n");}

}

}

Cccc