PROJECT REPORT
Retail Business Management System

Created By

Anurag Singh

Debashish Majumdar

Sean Annunciation

CS 532 – Database System Project

PL/SQL and JDBC Connectivity

Department Of Computer Science

State University Of New York At Binghamton

30th November 2017

INTRODUCTION

This project enables the user to access various details regarding the Retail Business. The user can easily interact with the system using a GUI which is connected to the database. The user can also add customers, purchases as well as check the monthly sale activities of a particular employee. Also, the user can check the total savings for a particular purchase. A purchase can also be deleted which will update the required tables after the deletion.

All operations on the database like insert, update, view tables for this project have been implemented using PL/SQL along with functions, procedures, sequences, triggers which have then been implemented in the Java application. The Logs table also helps to keep track of the various updates or inserts made into the various tables using triggers and sequences.

IMPLEMENTATION

Implementation consists of the following steps

1) Creation of Database tables in Oracle 11g
2) Package creation which will contain the procedures and functions which will perform various insert , delete and select operations in PL/SQL.
3) Sequence generation to autoincrement the PUR#, LOG# and SUP#.
4) Creation of triggers to update the respective tables as well as to keep a check on various conditions as required by this project.

SEQUENCES CREATED FOR THE PROJECT

These are the sequences that have been created for this project.

log_seq - This sequence is used to generate log# values for the Logs table when a new entry is made into the table. It will begin from 00001 and increment by 1. Below is the code.

```
create sequence log_seq
      increment by 1
      start with 00001
      maxvalue 99999
      cycle
      order;
```

pur_seq - This sequence is used to generate pur# values for the purchases table when a new entry is made into the table. It will begin from 100015 and increment by 1. Below is the code.

```
create sequence pur_seq
  increment by 1
  start with 100015
  maxvalue 999999
  cycle
  order;
```

sup_seq - This sequence is used to generate sup# values for the supplies table when a new entry is made into the table. It will begin from 1010 and increment by 1. Below is the code.

```
create sequence sup_seq
      increment by 1
      start with 1010
      maxvalue 9999
```

```
    nocache
    cycle;
```

PACKAGE CREATED FOR THE PROJECT

The Package named rbms_pack has been created for this project which contains all the procedures and functions used by the project. This package is created as follows.

1) Creating the package with the declaration of procedures and functions. Below is the code for the same.

```
create or replace package rbms_package as
type ref_cursor is ref cursor;

/*This procedure is used to display the products table*/
procedure display_products(prod_cur out ref_cursor);

/*This procedure is used to display the employees table*/
procedure display_employees(emp_cur out ref_cursor);

/*This procedure is used to display the customers table*/
procedure display_customers(cust_cur out ref_cursor);

/*This procedure is used to display the discounts table*/
procedure display_discounts(dis_cur out ref_cursor);

/*This procedure is used to display the suppliers table*/
procedure display_suppliers(supplier_cur out ref_cursor);

/*This procedure is used to display the supplies table*/
procedure display_supplies(supplies_cur out ref_cursor);

/*This procedure is used to display the purchases table*/
procedure display_purchases(pur_cur out ref_cursor);

/*This procedure is used to display the logs table*/
procedure display_logs(log_cur out ref_cursor);

/*This function is used to report the total savings for a pur#*/
FUNCTION purchase_saving(pur#_in in NUMBER)
return number;

/*This procedure is used to check the monthly sales activities*/ /*for an employee*/
procedure monthly_sale_activities(employee_id in
employees.eid%type,Invaliderror out varchar2, c1 OUT ref_cursor);

/*This procedure is used to add customers*/
procedure add_customers(c_id in customers.cid%type,c_name in
customers.name%type,c_telephone in customers.telephone#%type,
Invaliderror1 out varchar2);
```

```
/*This procedure is used to add purchases*/
procedure add_purchase(e_id in purchases.eid%type,
p_id in purchases.pid%type,
c_id in purchases.cid%type,
pur_qty in purchases.qty%type, poutput out varchar, error out varchar2);

/*This procedure is used to delete a purchase*/
PROCEDURE DELETE_PURCHASE(PUR_IN in purchases.pur#%type, error out varchar2,
poutput out varchar);


end rbms_package;
/
```

2) Creating the package body with the functions and procedures defined in it.

```
set serveroutput on
create or replace package body rbms_package as

procedure display_products(prod_cur out ref_cursor) is
begin
        open prod_cur for select * from products order by pid;
end display_products;

/*procedure display_customers(cust_cur out ref_cursor)*/

procedure display_customers(cust_cur out ref_cursor) is
begin
      open cust_cur for select * from customers order by cid;
end display_customers;

/*procedure display_discounts(dis_cur out ref_cursor);*/

procedure display_discounts(dis_cur out ref_cursor) is
begin
      open dis_cur for select * from discounts;
end display_discounts;


/*procedure display_suppliers(supplier_cur out ref_cursor);*/

procedure display_suppliers(supplier_cur out ref_cursor) is
begin
      open supplier_cur for select * from suppliers order by sid;
end display_suppliers;
```

```
/*procedure display_supplies(supplies_cur out ref_cursor);*/

procedure display_supplies(supplies_cur out ref_cursor) is
begin
        open supplies_cur for select * from supplies order by sup#;
end display_supplies;


/* procedure display_purchases(pur_cur out ref_cursor);*/

procedure display_purchases(pur_cur out ref_cursor) is
begin
        open pur_cur for select * from purchases order by pur#;
end display_purchases;

/*procedure display_logs(log_cur out ref_cursor);*/

procedure display_logs(log_cur out ref_cursor) is
begin
        open log_cur for select * from logs order by log#;
end display_logs;

/*procedure to display employees*/

procedure display_employees(emp_cur out ref_cursor) is
begin
            open emp_cur for select * from employees;
end display_employees;


/* function to report the total saving of any purchase */

FUNCTION purchase_saving(pur#_in IN NUMBER)
RETURN NUMBER
IS
SAVING NUMBER;
pur#_count number;

BEGIN
select count(*) into pur#_count from purchases where pur#=pur#_in;

if(pur#_count=0) then
        RETURN 0;
else
SELECT ((p.original_price*pr.qty)-pr.total_price) INTO SAVING from
purchases pr join products p on p.pid=pr.pid where
pur#=pur#_in;

RETURN SAVING;
end if;
```

```
END purchase_saving;


/* procedure to report the monthly sales activity of any
given employee */
procedure monthly_sale_activities(employee_id in
employees.eid%type, Invaliderror out varchar2 , c1 OUT ref_cursor)
is

Invalideid exception;
count_val number;

begin
select count(*) into count_val from employees where eid = employee_id;

if(count_val = 0) then
        raise Invalideid;
else
     open c1 for
select e.eid, e.name, to_char(pu.ptime,
'MON-YYYY') "month", count(pu.ptime)total_sales, sum(pu.qty)total_quantity,
sum(pu.total_price)total_amount from employees e, purchases pu where
e.eid=pu.eid and e.eid=employee_id group by e.eid, e.name,
to_char(pu.ptime, 'MON-YYYY');

end if;
exception
        when Invalideid then
        Invaliderror:='Employee id does not exist';

end monthly_sale_activities;



/* procedure to add customers to the customer table */

procedure add_customers(
c_id in customers.cid%type,
c_name in customers.name%type,
c_telephone in customers.telephone#%type,
Invaliderror1 out varchar2) is

Invalidcid exception;
count_val1 number;

begin
select count(*) into count_val1 from customers where cid = c_id;
if(count_val1=0) then
Invaliderror1:='';
     insert into customers (cid, name, telephone#, visits_made,last_visit_date)
```

```
values (c_id, c_name, c_telephone, 1, sysdate);

else
raise Invalidcid;
end if;

exception
        when Invalidcid then
        Invaliderror1:='customer already exists';

end add_customers;

/* procedure to add tuples in purchases table */

procedure add_purchase(e_id in purchases.eid%type,
p_id in purchases.pid%type,
c_id in purchases.cid%type,
pur_qty in purchases.qty%type,
poutput out varchar,
error out varchar2) is

pid_error exception;
eid_error exception;
cid_error exception;
pur_date date;
pur_total_price number(7,2);
next_pur# number(6);
remain_qoh number(5);
o_price number(6,2);
d_rate  number(3,2);
pid_count number;
eid_count number;
cid_count number;



BEGIN

pur_date:=SYSDATE;
select count(*) into pid_count from products where pid = p_id;
select count(*) into eid_count from employees where eid = e_id;
select count(*) into cid_count from customers where cid = c_id;


if(eid_count=0) then
raise eid_error;

elsif(pid_count=0) then
raise pid_error;
```

```
elsif(cid_count=0) then
raise cid_error;

else
error:='';
SELECT pr.original_price , d.discnt_rate into o_price,d_rate from
products pr, discounts d where d.discnt_category=pr.discnt_category
and pr.pid = p_id;

pur_total_price:=(o_price*(1-d_rate))* pur_qty;
select qoh into remain_qoh from products pr where pr.pid = p_id;

if (remain_qoh-pur_qty)<0 then
------dbms_output.put_line('Insufficient quantity in stock, the purchase request is rejected');
poutput:= 'Insufficient quantity in stock';

else
        next_pur#:=pur_seq.nextval;
        insert into purchases values (next_pur#,e_id,p_id,c_id,
pur_qty, pur_date, pur_total_price);
--------dbms_output.put_line('Purchase Successful');
poutput:= 'Purchase Successful';
end if;

end if;

exception
when eid_error then
error:='Employee does not exists';

when pid_error then
error:='Product does not exists';

when cid_error then
error:='Customer does not exists';

END add_purchase;


/*procedure to delete tuple from purchase*/

PROCEDURE DELETE_PURCHASE(PUR_IN in purchases.pur#%type, error out varchar2,
poutput out varchar)
is
pur_error exception;
pur_count number;

BEGIN
select count(*) into pur_count from purchases where pur# = PUR_IN;
```

```
if(pur_count=0) then
raise pur_error;

else
error:='';

DELETE FROM PURCHASES
WHERE PURCHASES.PUR#=PUR_IN;
poutput:= 'Delete Successful';

end if;

exception
when pur_error then
error:='Purchases number does not exists';

END DELETE_PURCHASE;

end rbms_package;
/
```

PROCEDURES AND FUNCTIONS USED IN THE PACKAGE

FUNCTIONS DEFINED IN THIS PACKAGE

1. **purchase_saving** – This Function takes PUR# as a parameter and returns the total purchase saving for that pur# if the pur# is valid and returns a meaningful message if it does not exist.

PROCEDURES DEFINED IN THIS PACKAGE

1. **display_products** – This procedure is used to display the contents of the products table.
2. **display_customers** - This procedure is used to display the contents of the customers table.
3. **display_discounts** - This procedure is used to display the contents of the discounts table.
4. **display_suppliers** - This procedure is used to display the contents of the suppliers table.
5. **display_supplies** – This procedure is used to display the contents of the supplies table.
6. **display_purchases** - This procedure is used to display the contents of the purchases table.
7. **display_logs** - This procedure is used to display the contents of the logs table.
8. **display_employees** - This procedure is used to display the contents of the employees table.
9. **monthly_sale_activities** – This procedure is to report the monthly sales activity for a given employee.
   Parameters
   IN parameter – Employee ID
   OUT parameter – Error and cursor
10. **add_customers** – This procedure is used to add a customer to the customers table.
    Parameters
    IN parameter – CID, CNAME, TELEPHONE#
    OUT parameter – Error
11. **add_purchase** – This procedure is used to add purchases to the purchases table.
    Parameters
    IN parameter – PID, CID, QTY
    OUT parameter – output, error
12. **DELETE_PURCHASE** – This procedure is used to delete a tuple from the purchases tables. It also uses triggers to update the products table and the customers table.
    Parameters
    IN parameter – PUR#
    OUT parameter – error, output

TRIGGERS CREATED FOR THE PROJECT

1. **LOG_TRIGGER** – This trigger will be fired when the user performs an insert on the customers table. This trigger will insert the username, tablename which is customers as well as the newly inserted customer ID into the LOGS table. Below is the code.

```
CREATE OR REPLACE TRIGGER LOG_TRIGGER
AFTER INSERT ON CUSTOMERS
FOR EACH ROW
BEGIN
INSERT INTO LOGS
(LOG#,USER_NAME,OPERATION,OP_TIME,TABLE_NAME,TUPLE_PKEY)
VALUES(LOG_SEQ.NEXTVAL,user,'INSERT',sysdate,'CUSTOMERS',:NEW.cid);
END;
/
```

2. **LOG_UPDATE_CUST_TRIGGER** – This trigger will be fired when the last_visit_date of the customers table has been updated for a particular customer. It will enter the respective customer ID, username, tablename which is customers as well as the operation which is UPDATE into the LOGS table.Below is the code.

```
CREATE OR REPLACE TRIGGER LOG_UPDATE_CUST_TRIGGER
AFTER UPDATE OF LAST_VISIT_DATE ON CUSTOMERS
FOR EACH ROW
BEGIN
INSERT INTO LOGS
(LOG#,USER_NAME,OPERATION,OP_TIME,TABLE_NAME,TUPLE_PKEY)
VALUES(LOG_SEQ.NEXTVAL,user,'UPDATE',sysdate,'CUSTOMERS',:NEW.cid);
END;
/
```

3. **LOG_PURCHASES_TRIGGER** – This trigger will be fired when the user performs an insert on the purchases table. This trigger will insert the username, tablename which is purchases, operation which is INSERT as well as the newly inserted purchase number into the LOGS table. Below is the code.

```
CREATE OR REPLACE TRIGGER LOG_PURCHASES_TRIGGER
AFTER INSERT ON PURCHASES
FOR EACH ROW
BEGIN
INSERT INTO LOGS
(LOG#,USER_NAME,OPERATION,OP_TIME,TABLE_NAME,TUPLE_PKEY)
VALUES(LOG_SEQ.NEXTVAL,user,'INSERT',sysdate,'PURCHASES',:NEW.pur#);
END;
/
```

4. **LOG_UPDATE_PROD_TRIGGER** – This trigger will be fired when the qoh of the products table has been updated for a particular product. It will enter the respective product ID,

username, tablename which is products as well as the operation which is UPDATE into the LOGS table. Below is the code.

```
CREATE OR REPLACE TRIGGER LOG_UPDATE_PROD_TRIGGER
AFTER UPDATE OF QOH ON PRODUCTS
FOR EACH ROW
BEGIN
INSERT INTO LOGS
(LOG#,USER_NAME,OPERATION,OP_TIME,TABLE_NAME,TUPLE_PKEY)
VALUES(LOG_SEQ.NEXTVAL,user,'UPDATE',sysdate,'PRODUCTS',:NEW.pid);
END;
/
```

5. **LOG_SUPPLIES_TRIGGER** – This trigger will be fired when the user performs an insert on the supplies table. This trigger will insert the username, tablename which is supplies, operation which is INSERT as well as the newly inserted sup number into the LOGS table. Below is the code.

```
CREATE OR REPLACE TRIGGER LOG_SUPPLIES_TRIGGER
AFTER INSERT ON SUPPLIES
FOR EACH ROW
BEGIN
INSERT INTO LOGS
(LOG#,USER_NAME,OPERATION,OP_TIME,TABLE_NAME,TUPLE_PKEY)
VALUES(LOG_SEQ.NEXTVAL,user,'INSERT',sysdate,'SUPPLIES',:NEW.sup#);
END;
/
```

6. **update_q_vm_lvd** – This trigger will be fired when an insert is made into the purchases table. It updates the qoh value in the products table by the difference in the current qoh value and the newly entered qoh value. It also updates the visits_made by 1 and the last_visit_date by the current date. If the new qoh value is less than the qoh_threshold value it will display a message saying that the qoh is less than the threshold and new supply is required. It then inserts a new sup_qty which is 10+qoh_threshold+1 into the supplies table for that particular sid and pid. It will also increase the qoh value in the products table by the new qoh value. Below is the code.

```
create or replace trigger update_q_vm_lvd
after insert on purchases
declare
pur#_id purchases.pur#%type;
p_id purchases.pid%type;
c_id purchases.cid%type;
pur_qty purchases.qty%type;
sup#_id supplies.sup#%type;
sup_date date;
sup_qty supplies.quantity%type;
temp_qoh_threshold products.qoh_threshold%type;
new_qoh products.qoh%type;
last_visit date;
```

```
temp_visits_made customers.visits_made%type;
s_sid supplies.sid%type;


BEGIN

Select sysdate into sup_date from dual;
select pur#,pid,cid,qty,ptime into pur#_id,p_id,c_id,pur_qty,last_visit from purchases group
by pur#,pid,cid,qty,ptime having pur#=(select max(pur#) from purchases);
update products set qoh=qoh-pur_qty where pid=p_id;
select qoh, qoh_threshold into new_qoh, temp_qoh_threshold from products pr where pr.pid
= p_id;
select visits_made into temp_visits_made from customers where cid=c_id;
update customers set visits_made = temp_visits_made+1 , last_visit_date = last_visit where
cid=c_id;

if (new_qoh < temp_qoh_threshold) then
        dbms_output.put_line('Quantity on hand(qoh) is below the required threshold and
new supply is required');
  sup_qty:=10+temp_qoh_threshold+1;
        select sid into s_sid from (select sid from supplies where pid=p_id order by sid asc)
where rownum = 1;
        insert into supplies values (sup_seq.nextval, p_id, s_sid, sup_date, sup_qty);
        update products set qoh=(qoh+sup_qty) where pid=p_id;
        dbms_output.put_line('New QOH: ' || (new_qoh+sup_qty));
end if;
end;
/
```

7. **PRODUCT_TRIGGER** – This trigger will be fired when a tuple in the purchases table has been deleted. It will increment the qoh value of that product in the products table by the qoh value recently deleted. It will also increment the visits_made in the customers table for that customer by 1. Also, the last_visit_date will be the current date. Below is the code.

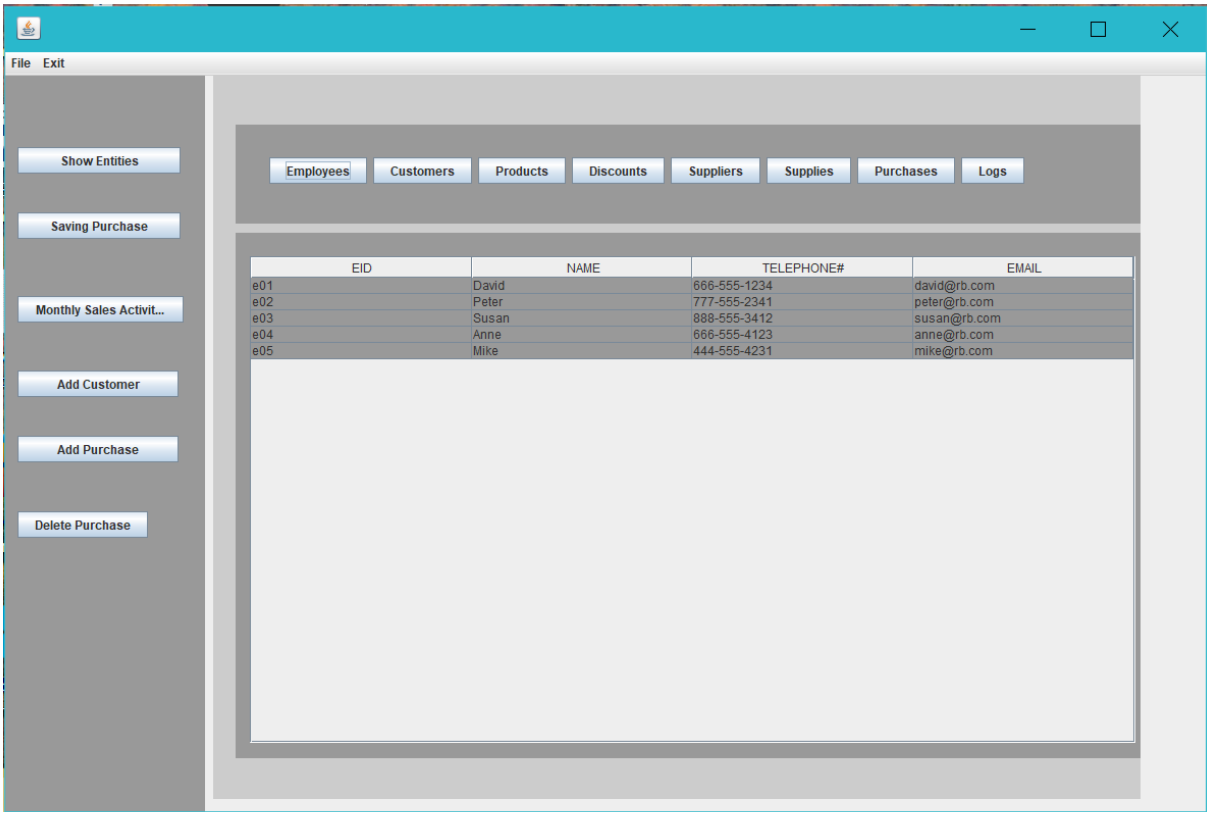
```
CREATE OR REPLACE TRIGGER PRODUCT_TRIGGER
AFTER DELETE ON PURCHASES
FOR EACH ROW
DECLARE
PROD_ID PURCHASES.PID%TYPE;
LAST_DATE PURCHASES.PTIME%TYPE;
BEGIN
UPDATE PRODUCTS SET PRODUCTS.QOH=PRODUCTS.QOH+:old.qty
WHERE PRODUCTS.PID=:old.pid;
UPDATE CUSTOMERS SET VISITS_MADE=VISITS_MADE+1,
LAST_VISIT_DATE=sysdate
WHERE CID=:old.cid;
END;
/
```

INTERFACE IMPLEMENTATION

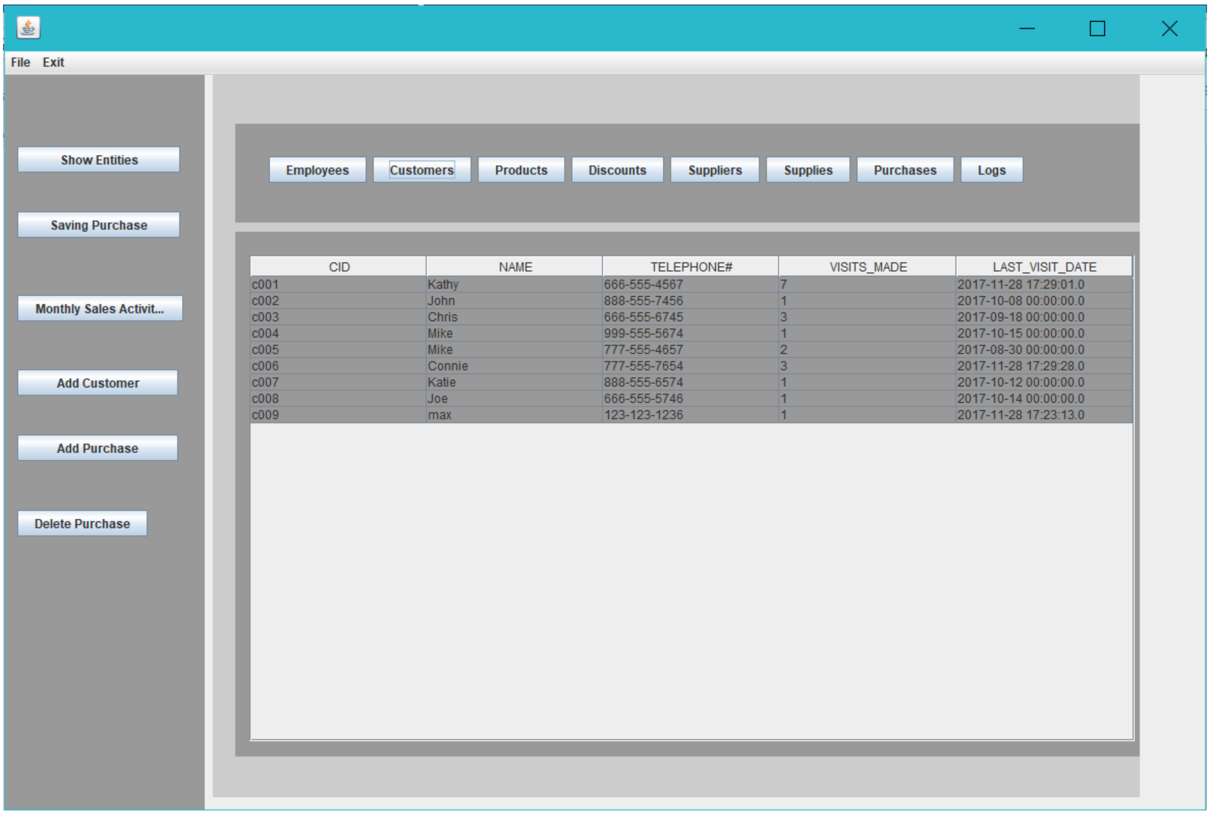
We have used Eclipse IDE to design the GUI as well as perform various operations on the database objects using JDBC connectivity for which ojdbc8 jar file has been used.
The Java application consists of the following Java Main files:

1. ConvertToJTable.java- This is used to display the tables in the GUI when a select operation is performed.
2. HomePage.java- This consists of the all the operations to be performed on the database objects which include viewing tables, inserting and deleting tuples in required tables.
3. projectDB.java- Java file to test the database data in the console.
4. RetailBusinessManagementSystem.java- It consists of the main method which is used to execute the home page and display the GUI.
5. ShowTables.java- It consists of the code to show the tables from the database in the GUI.

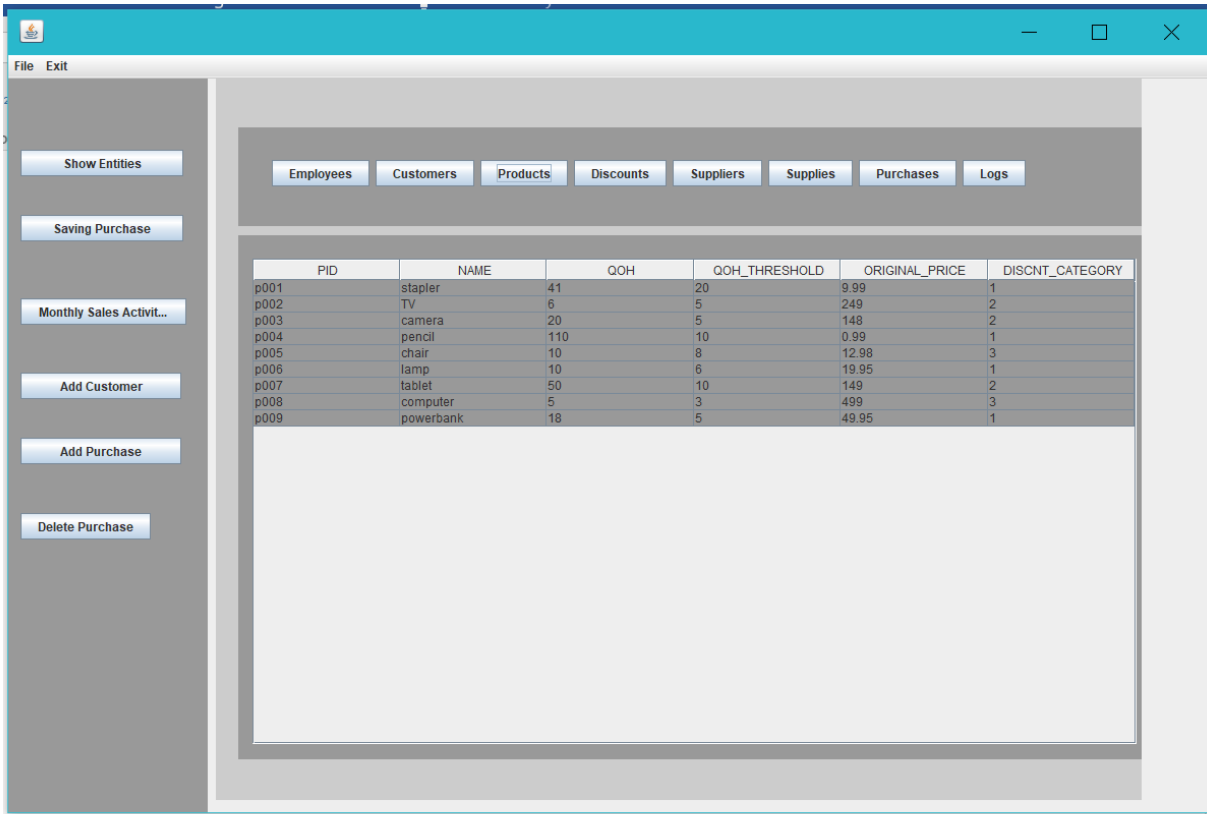
Result



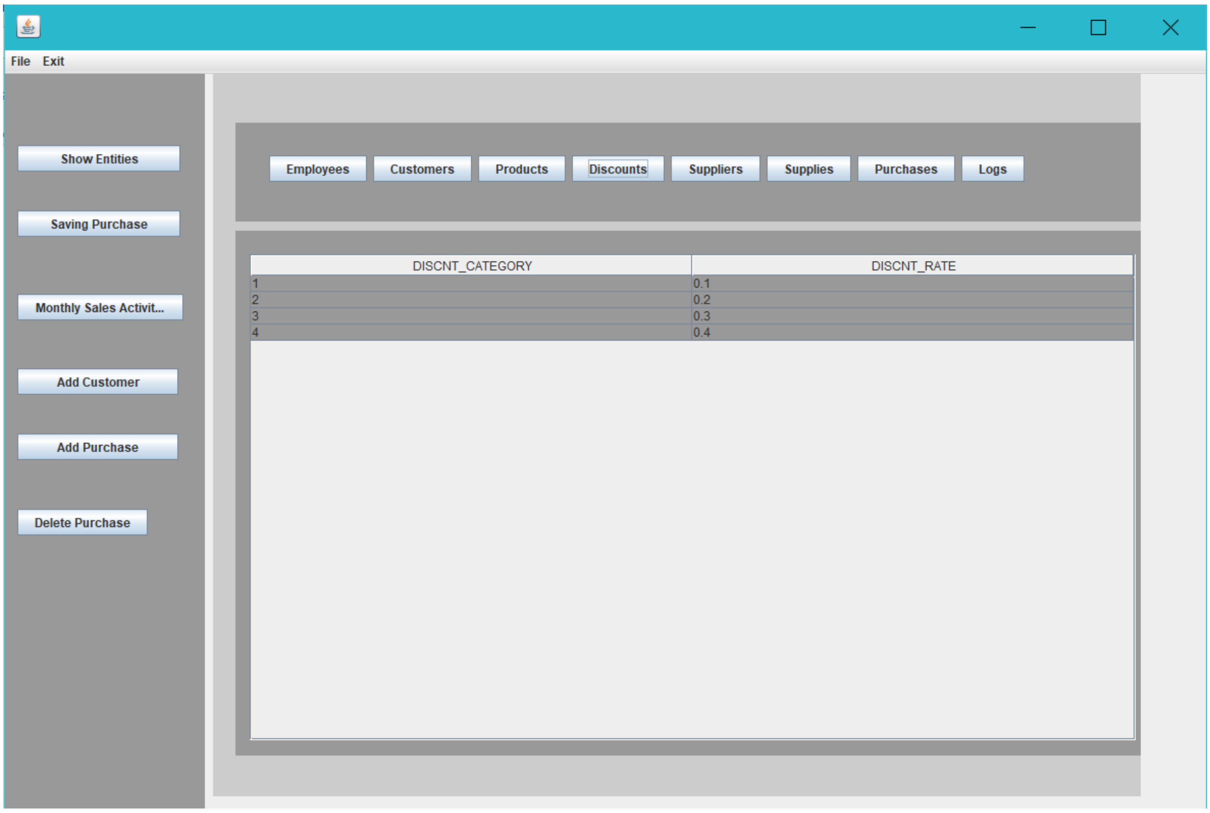
EID	NAME	TELEPHONE#	EMAIL
e01	David	666-555-1234	david@rb.com
e02	Peter	777-555-2341	peter@rb.com
e03	Susan	888-555-3412	susan@rb.com
e04	Anne	666-555-4123	anne@rb.com
e05	Mike	444-555-4231	mike@rb.com



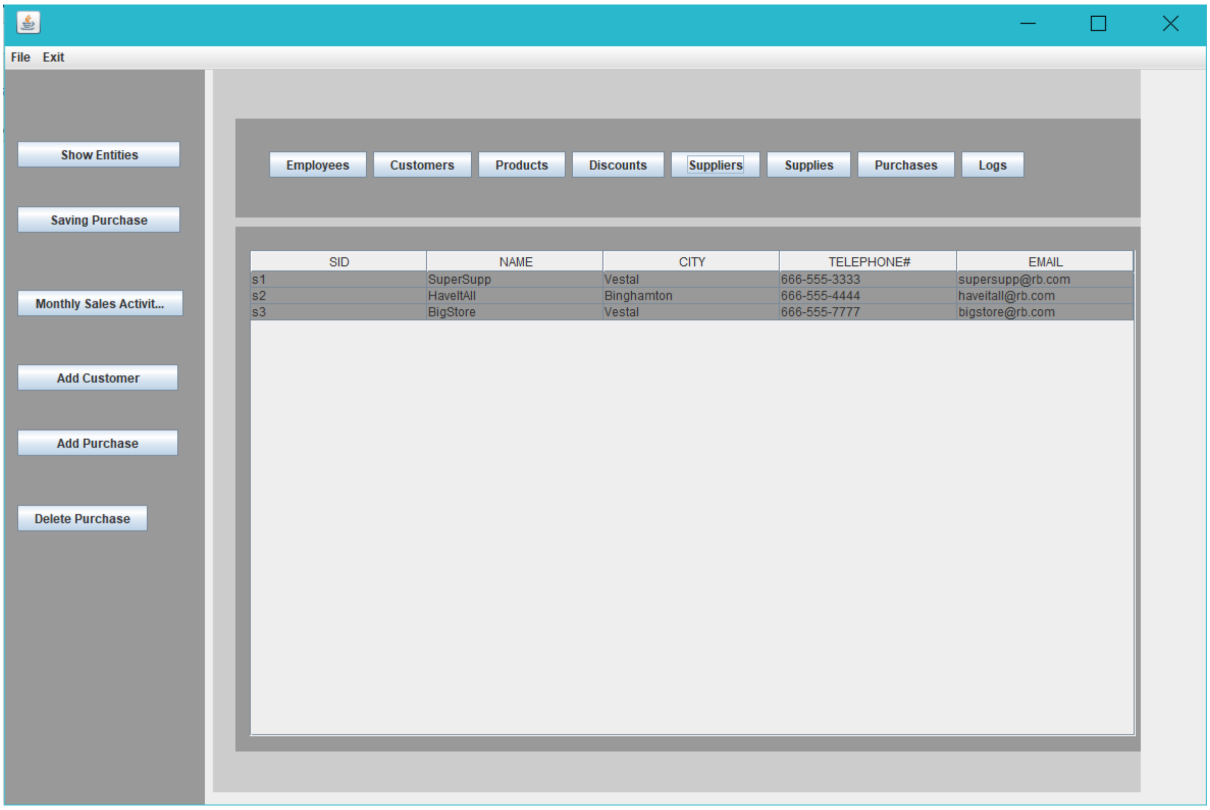
CID	NAME	TELEPHONE#	VISITS_MADE	LAST_VISIT_DATE
c001	Kathy	666-555-4567	7	2017-11-28 17:29:01.0
c002	John	888-555-7456	1	2017-10-08 00:00:00.0
c003	Chris	666-555-6745	3	2017-09-18 00:00:00.0
c004	Mike	999-555-5674	1	2017-10-15 00:00:00.0
c005	Mike	777-555-4657	2	2017-08-30 00:00:00.0
c006	Connie	777-555-7654	3	2017-11-28 17:29:28.0
c007	Katie	888-555-6574	1	2017-10-12 00:00:00.0
c008	Joe	666-555-5746	1	2017-10-14 00:00:00.0
c009	max	123-123-1236	1	2017-11-28 17:23:13.0



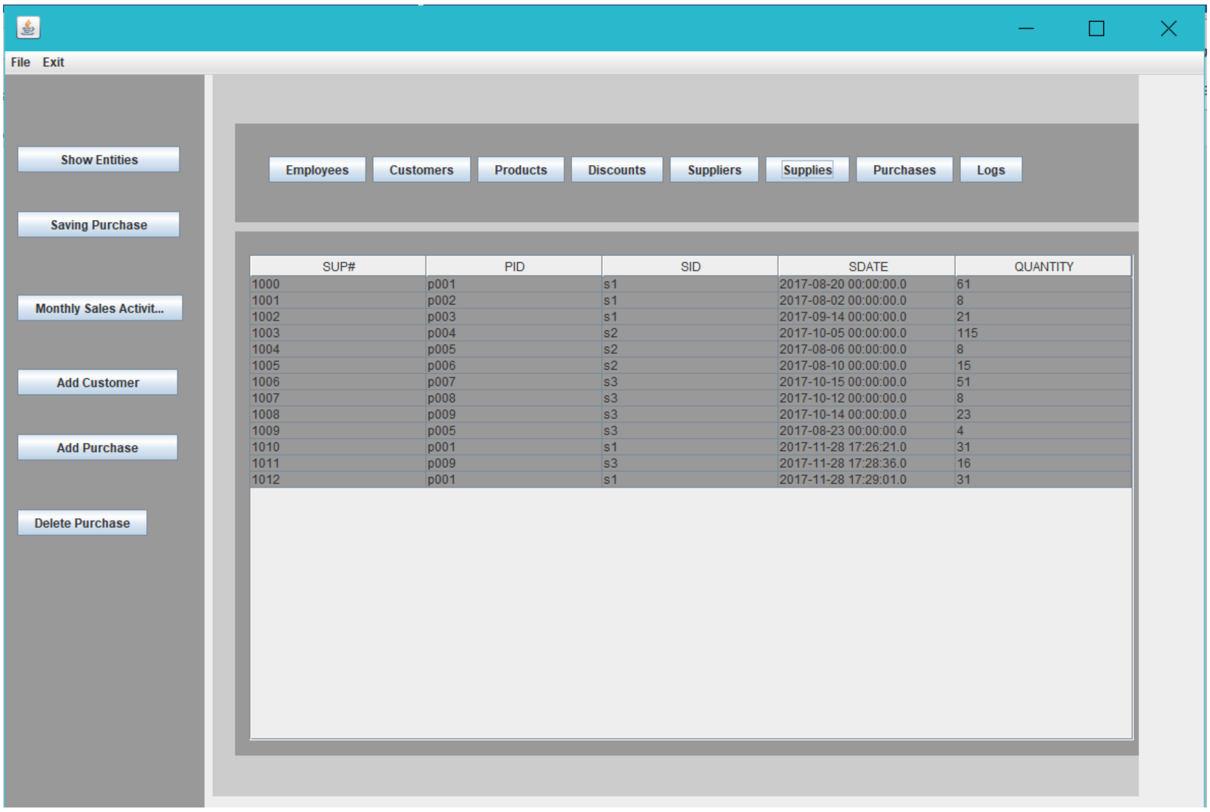
PID	NAME	QOH	QOH_THRESHOLD	ORIGINAL_PRICE	DISCNT_CATEGORY
p001	stapler	41	20	9.99	1
p002	TV	6	5	249	2
p003	camera	20	5	148	2
p004	pencil	110	10	0.99	1
p005	chair	10	8	12.98	3
p006	lamp	10	6	19.95	1
p007	tablet	50	10	149	2
p008	computer	5	3	499	3
p009	powerbank	18	5	49.95	1



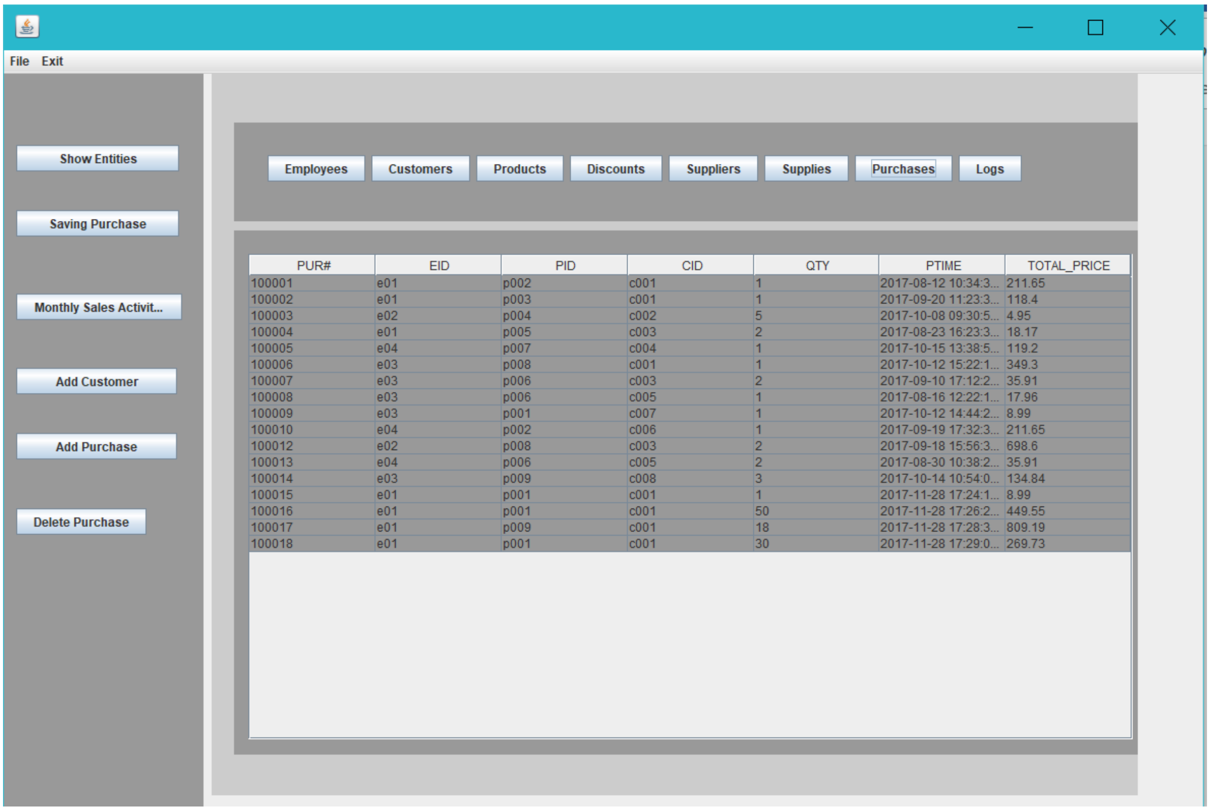
DISCNT_CATEGORY	DISCNT_RATE
1	0.1
2	0.2
3	0.3
4	0.4



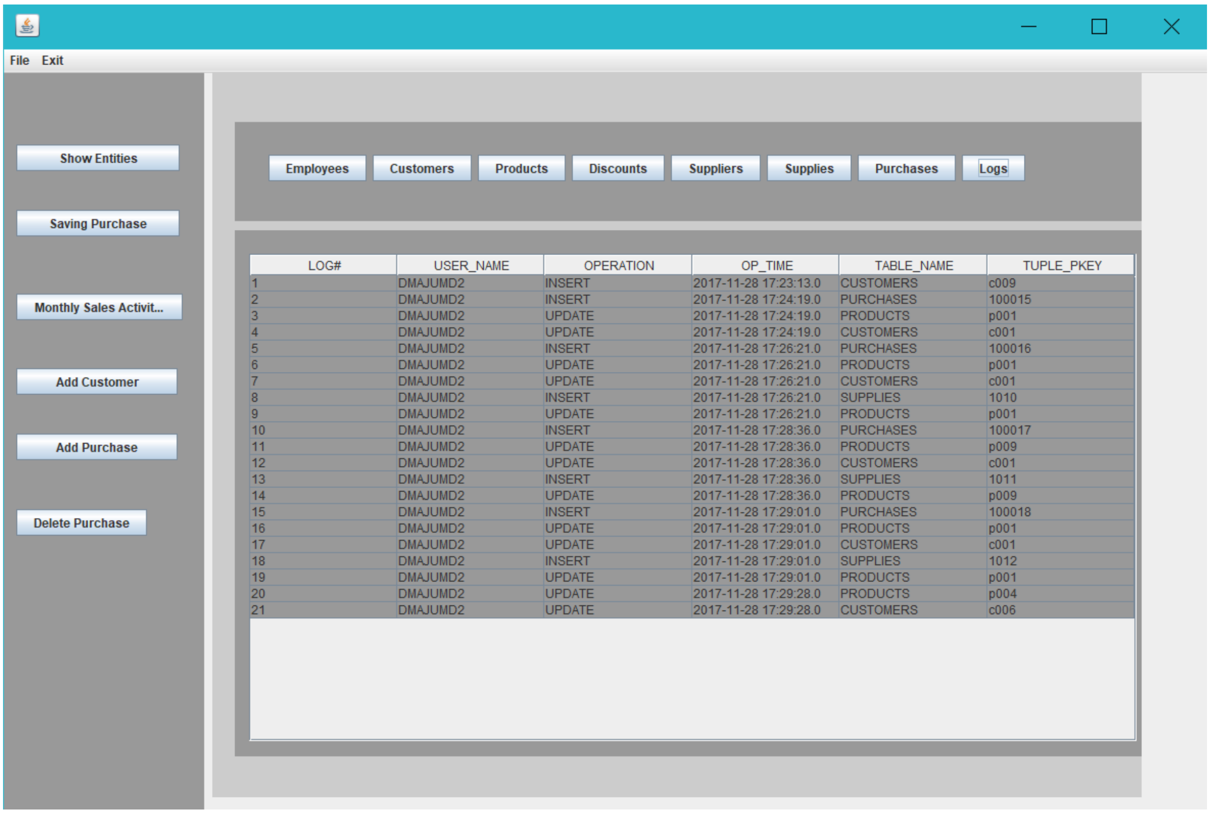
SID	NAME	CITY	TELEPHONE#	EMAIL
s1	SuperSupp	Vestal	666-555-3333	supersupp@rb.com
s2	HaveItAll	Binghamton	666-555-4444	haveitall@rb.com
s3	BigStore	Vestal	666-555-7777	bigstore@rb.com



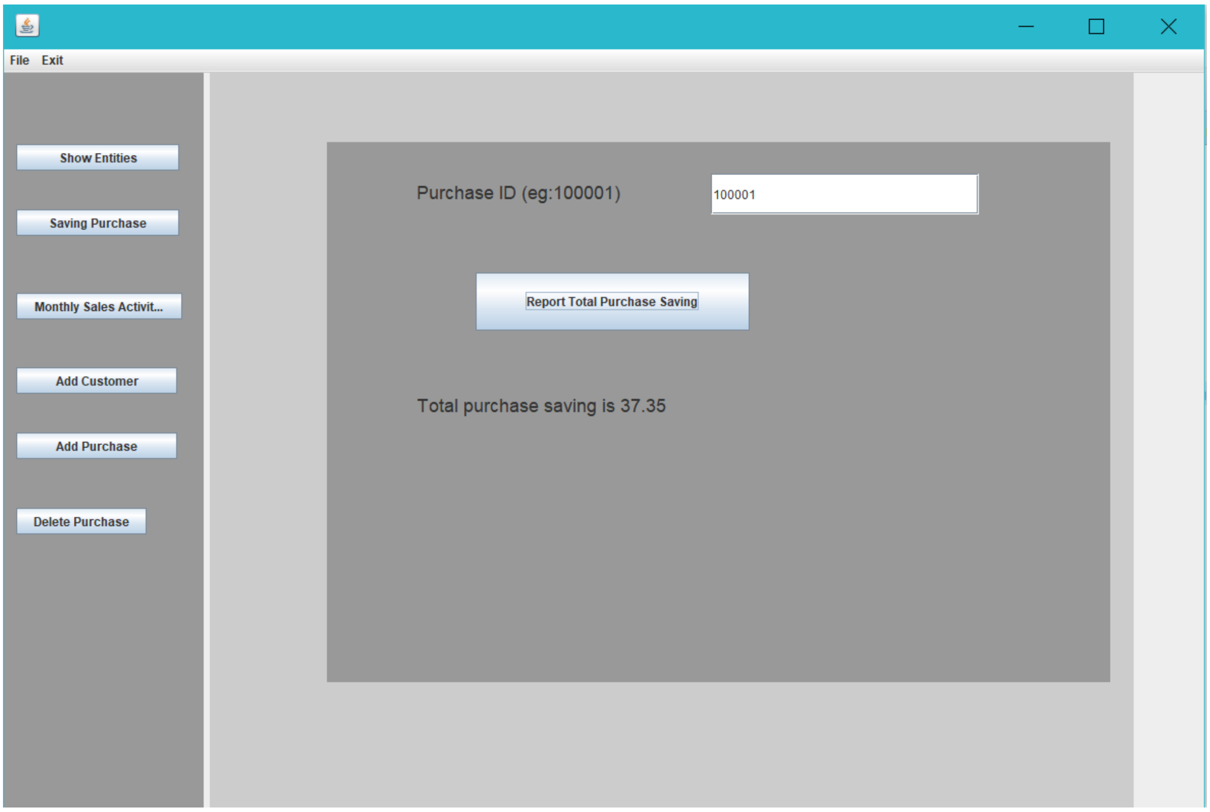
SUP#	PID	SID	SDATE	QUANTITY
1000	p001	s1	2017-08-20 00:00:00.0	61
1001	p002	s1	2017-08-02 00:00:00.0	8
1002	p003	s1	2017-09-14 00:00:00.0	21
1003	p004	s2	2017-10-05 00:00:00.0	115
1004	p005	s2	2017-08-06 00:00:00.0	8
1005	p006	s2	2017-08-10 00:00:00.0	15
1006	p007	s3	2017-10-15 00:00:00.0	51
1007	p008	s3	2017-10-12 00:00:00.0	8
1008	p009	s3	2017-10-14 00:00:00.0	23
1009	p005	s3	2017-08-23 00:00:00.0	4
1010	p001	s1	2017-11-28 17:26:21.0	31
1011	p009	s3	2017-11-28 17:28:36.0	16
1012	p001	s1	2017-11-28 17:29:01.0	31



PUR#	EID	PID	CID	QTY	PTIME	TOTAL_PRICE
100001	e01	p002	c001	1	2017-08-12 10:34:3...	211.65
100002	e01	p003	c001	1	2017-09-20 11:23:3...	118.4
100003	e02	p004	c002	5	2017-10-08 09:30:5...	4.95
100004	e01	p005	c003	2	2017-08-23 16:23:3...	18.17
100005	e04	p007	c004	1	2017-10-15 13:38:5...	119.2
100006	e03	p008	c001	1	2017-10-12 15:22:1...	349.3
100007	e03	p006	c003	2	2017-09-10 17:12:2...	35.91
100008	e03	p006	c005	1	2017-08-16 12:22:1...	17.96
100009	e03	p001	c007	1	2017-10-12 14:44:2...	8.99
100010	e04	p002	c006	1	2017-09-19 17:32:3...	211.65
100012	e02	p008	c003	2	2017-09-18 15:56:3...	698.6
100013	e04	p006	c005	2	2017-08-30 10:38:2...	35.91
100014	e03	p009	c008	3	2017-10-14 10:54:0...	134.84
100015	e01	p001	c001	1	2017-11-28 17:24:1...	8.99
100016	e01	p001	c001	50	2017-11-28 17:26:2...	449.55
100017	e01	p009	c001	18	2017-11-28 17:28:3...	809.19
100018	e01	p001	c001	30	2017-11-28 17:29:0...	269.73

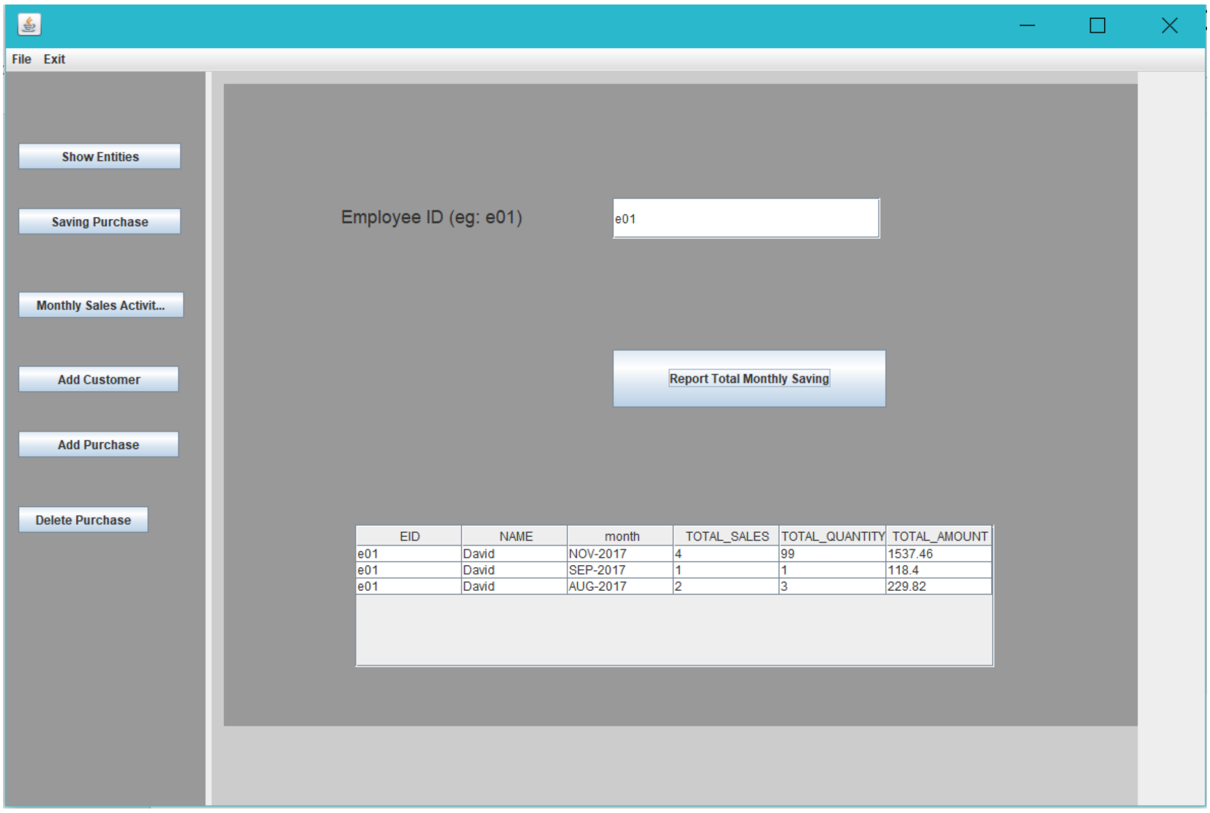


LOG#	USER_NAME	OPERATION	OP_TIME	TABLE_NAME	TUPLE_PKEY
1	DMAJUMD2	INSERT	2017-11-28 17:23:13.0	CUSTOMERS	c009
2	DMAJUMD2	INSERT	2017-11-28 17:24:19.0	PURCHASES	100015
3	DMAJUMD2	UPDATE	2017-11-28 17:24:19.0	PRODUCTS	p001
4	DMAJUMD2	UPDATE	2017-11-28 17:24:19.0	CUSTOMERS	c001
5	DMAJUMD2	INSERT	2017-11-28 17:26:21.0	PURCHASES	100016
6	DMAJUMD2	UPDATE	2017-11-28 17:26:21.0	PRODUCTS	p001
7	DMAJUMD2	UPDATE	2017-11-28 17:26:21.0	CUSTOMERS	c001
8	DMAJUMD2	INSERT	2017-11-28 17:26:21.0	SUPPLIES	1010
9	DMAJUMD2	UPDATE	2017-11-28 17:26:21.0	PRODUCTS	p001
10	DMAJUMD2	INSERT	2017-11-28 17:28:36.0	PURCHASES	100017
11	DMAJUMD2	UPDATE	2017-11-28 17:28:36.0	PRODUCTS	p009
12	DMAJUMD2	UPDATE	2017-11-28 17:28:36.0	CUSTOMERS	c001
13	DMAJUMD2	INSERT	2017-11-28 17:28:36.0	SUPPLIES	1011
14	DMAJUMD2	UPDATE	2017-11-28 17:28:36.0	PRODUCTS	p009
15	DMAJUMD2	INSERT	2017-11-28 17:29:01.0	PURCHASES	100018
16	DMAJUMD2	UPDATE	2017-11-28 17:29:01.0	PRODUCTS	p001
17	DMAJUMD2	UPDATE	2017-11-28 17:29:01.0	CUSTOMERS	c001
18	DMAJUMD2	INSERT	2017-11-28 17:29:01.0	SUPPLIES	1012
19	DMAJUMD2	UPDATE	2017-11-28 17:29:01.0	PRODUCTS	p001
20	DMAJUMD2	UPDATE	2017-11-28 17:29:28.0	PRODUCTS	p004
21	DMAJUMD2	UPDATE	2017-11-28 17:29:28.0	CUSTOMERS	c006



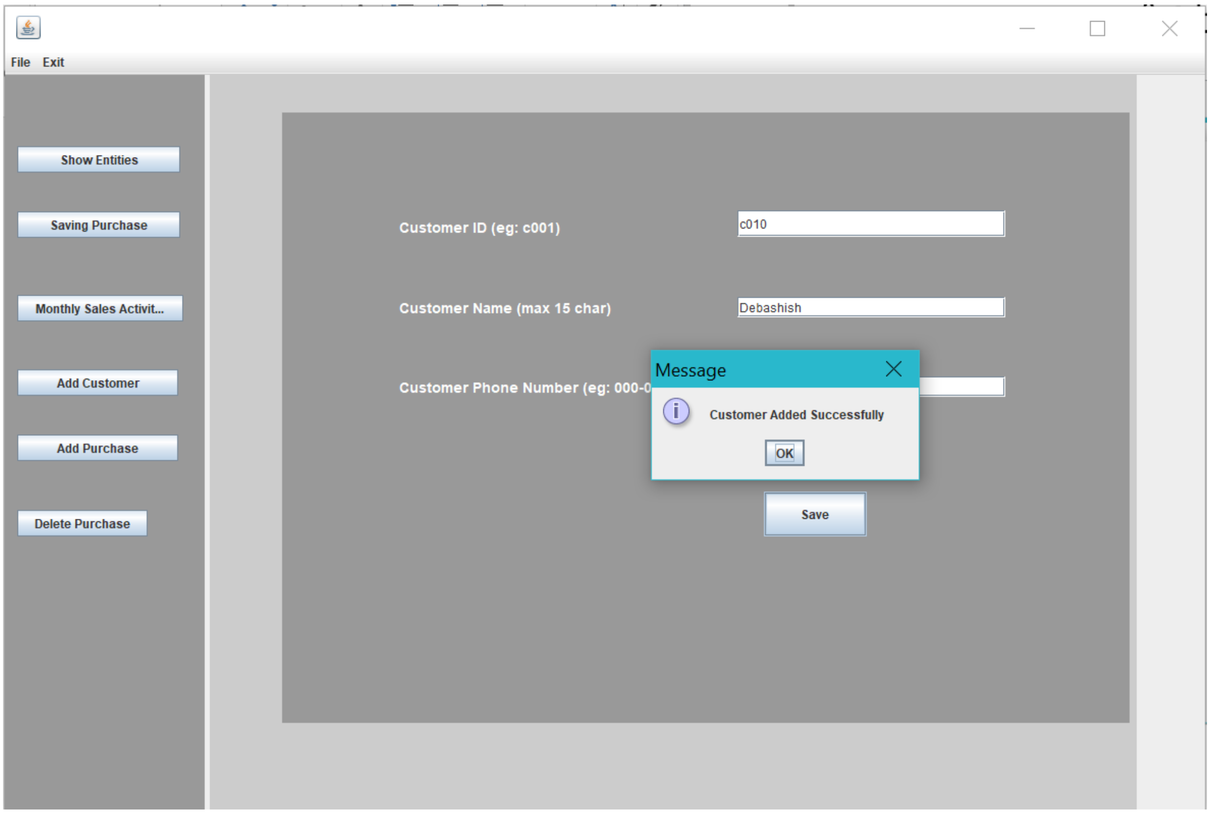
Purchase ID (eg:100001) 100001

Total purchase saving is 37.35



Employee ID (eg: e01) e01

EID	NAME	month	TOTAL_SALES	TOTAL_QUANTITY	TOTAL_AMOUNT
e01	David	NOV-2017	4	99	1537.46
e01	David	SEP-2017	1	1	118.4
e01	David	AUG-2017	2	3	229.82

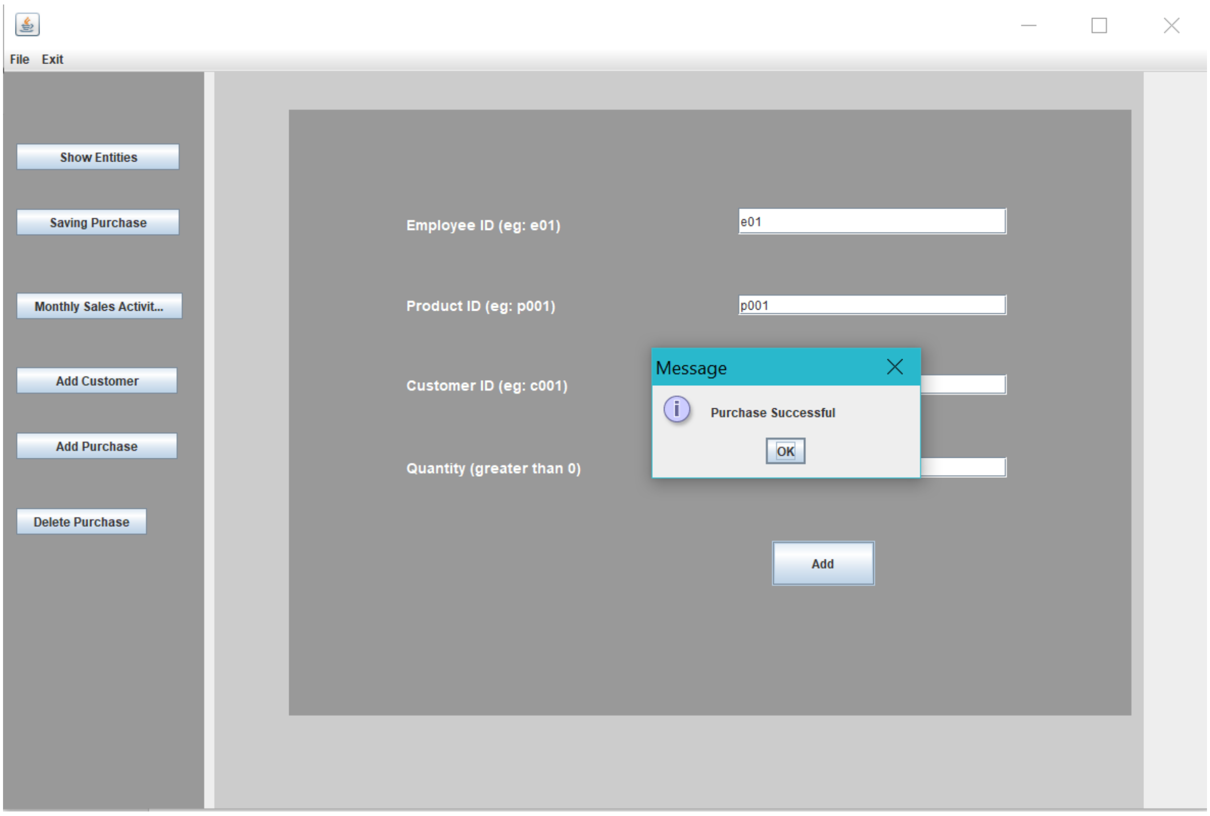


Customer ID (eg: c001) c010

Customer Name (max 15 char) Debashish

Customer Phone Number (eg: 000-0

Customer Added Successfully



Employee ID (eg: e01) e01

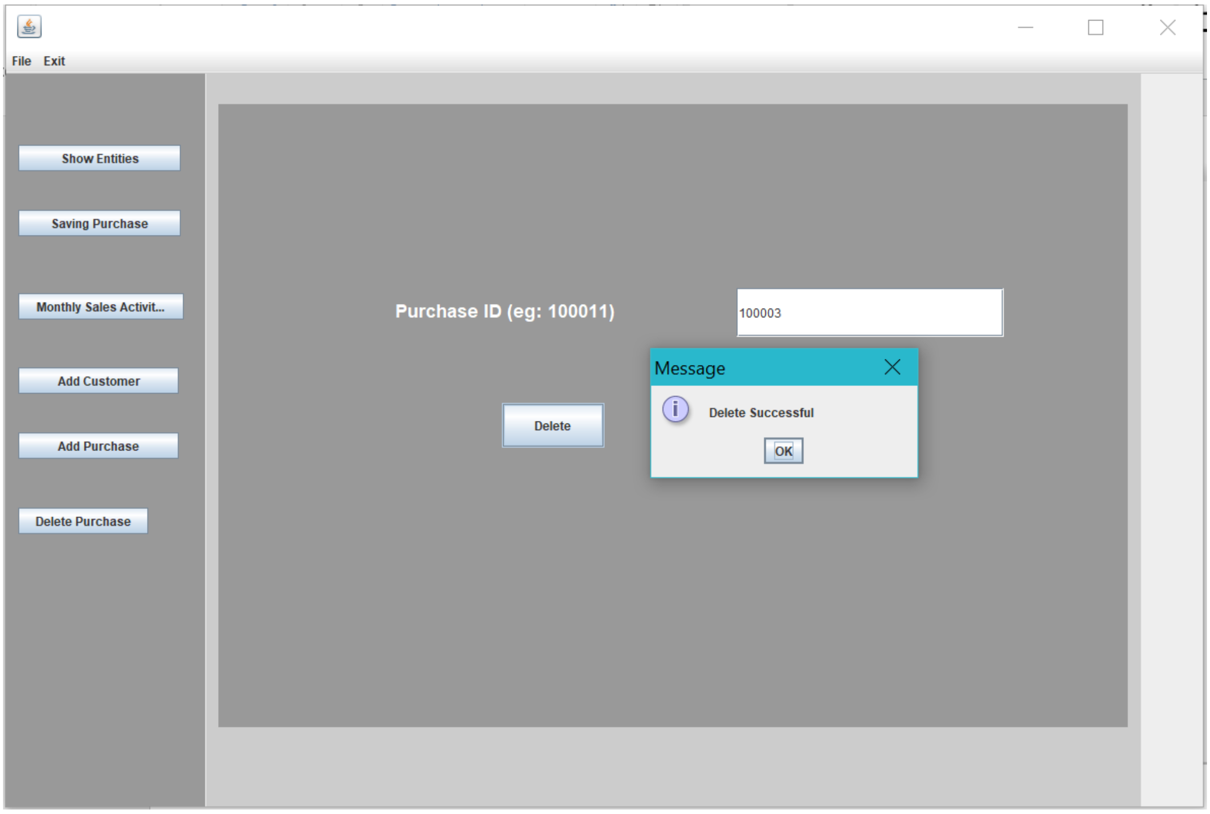
Product ID (eg: p001) p001

Customer ID (eg: c001)

Quantity (greater than 0)

Message

Purchase Successful



Purchase ID (eg: 100011) 100003

Message

Delete Successful

RESULT

All queries were successfully executed and implemented using PL/SQL and JDBC connectivity. Also, we have created a GUI in java which reflects the database data.

COLLABORATION REPORT

TEAM MEMBERS
ANURAG SINGH
DEBASHISH MAJUMDAR
SEAN ANNUNCIATION

13th November
During our first meeting we had a brief discussion about the project. We also discussed how to design the GUI and what should be displayed on it.

15th November
The 9 questions were divided amongst us.
Debashish (1,2,5,9) Anurag (2,3,4,7) and Sean (3,6,8).

16th November to 20th November
We finished the PL/SQL.

21st November
We met to discuss various errors that were encountered and resolved them. Also, we tried different test cases. On successful implementation of the procedures and functions we created a package with the procedures and functions while the triggers and sequences were outside the package.

22nd November to 24th November
The GUI was then developed by Debashish, Anurag and Sean. The JDBC connections were done by Debashish. Anurag and Sean worked on designing the UI. It was then combined and the final GUI was made functional.

25th November
Once the GUI was developed we began connecting the PL/SQL code with the database.

27th November
Sean created the report for the project. Anurag and Debashish executed various test cases.

It was a good experience working on this project as we got to learn new concepts as well as it improved our knowledge base.