**Database Design: Using MySQL and PHP to**

**Implement the Retail Business Management System**

This project is to use MySQL and PHP to implement the RBMS application.

**1. Preparation (5 points)**

We will use the following tables:

**Employees**(eid, ename, city)

**Customers**(cid, cname, city, visits\_made, last\_visit\_time)

**Products**(pid, pname, qoh, qoh\_threshold, original\_price, discnt\_rate, sid)

**Suppliers(**sid, sname, city, telephone)

**Purchases**(pur, cid, eid, pid, qty, pdate, total\_price)

In addition, the following table is also needed for this project:

**Logs**(logid, who, time, table\_name, operation, key\_value)

Each tuple in the logs table describes who (the login name of a database user) has performed what operation (insert, delete, update) on which table (give the table name) and which tuple (as indicated by the value of the primary key of the tuple) at what time. Attribute logid is the primary key of table.

Use the following SQL DDL statements to create the seven tables required for this project. Note that you need to use the exact statements as shown below to ensure that the instructor can test your programs using the instructor’s data later. Please also note that the tables are created in certain order such that by the time when a foreign key needs to be created, the corresponding primary key has already been created.

create table employees

(eid varchar(3) not null,   
//change all eid, cid, pid, pur, key\_value to int types in the future

ename varchar(15),

city varchar(15),

primary key(eid));

create table customers

(cid varchar(4) not null,

cname varchar(15),

city varchar(15),

visits\_made int(5),

last\_visit\_time datetime,

primary key(cid));

create table suppliers

(sid varchar(2) not null,

sname varchar(15) not null,

city varchar(15),

telephone\_no char(10),

primary key(sid),

unique(sname));

create table products

(pid varchar(4) not null,

pname varchar(15) not null,

qoh int(5) not null,

qoh\_threshold int(5),

original\_price decimal(6,2),

discnt\_rate decimal(3,2),

sid varchar(2),

primary key(pid),

foreign key (sid) references suppliers (sid));

create table purchases

(pur varchar(4)not null,

cid varchar(4) not null,

eid varchar(3) not null,

pid varchar(4) not null,

qty int(5),

ptime datetime,

total\_price decimal(7,2),

primary key (pur),

foreign key (cid) references customers(cid),

foreign key (eid) references employees(eid),

foreign key (pid) references products(pid));

create table logs

(logid int(5) not null auto\_incrment,

who varchar(10) not null,

time datetime not null,

table\_name varchar(20) not null,

operation varchar(6) not null,

key\_value varchar(4),

primary key (logid));

The meanings of most of the tables and their attributes are clear. If they are not clear to you, please let the instructor know. For a given customer, visits\_made indicates how many times the customer has purchased products from the business. For a given customer, last\_visit\_time is the time of the most recent visit made by the customer. Attribute qoh in the products table indicates quantity on hand and for each product, qoh\_threshold is an integer such that when qoh becomes less than qoh\_threshold, it is time to get new supplies of this product from the suppliers. Each tuple in the purchases table tells which customer (cid) has purchased what product (pid) with what quantity (qty) and the total price (total\_price) from which employee (eid) at what time (ptime). The total price is computed by multiplying the discount price of the product and the quantity purchased. The discount price of a product is computed based on its original price and its discount rate.

You should populate the first five tables with appropriate tuples to test your program.

**2. MySQL Implementation (50 points)**

You need to write SQL queries, stored procedures/functions, and triggers to implement this project. The following requirements and functionalities need to be implemented.

1. (6 points) Write a stored procedure to show the tuples in each table. For example, you can implement a procedure, say **show\_products()**, to display all products in the products table.

利用PREPARE,EXECUTE语句进行编写存储过程，使其能够打印不同的表单，源代码如下：

DELIMITER //

DROP PROCEDURE IF EXISTS SHOW\_TABLE;

CREATE PROCEDURE SHOW\_TABLE(IN TABLE\_NAME VARCHAR(64))

BEGIN

SET @sentence = concat('SELECT \* FROM ', TABLE\_NAME, ';');

PREPARE stmt FROM @sentence;

EXECUTE stmt;

SET @sentence = NULL;

END //

DELIMITER ;

结果如下图1所示：



图1(a). SHOW\_TABLE存储过程运行结果图

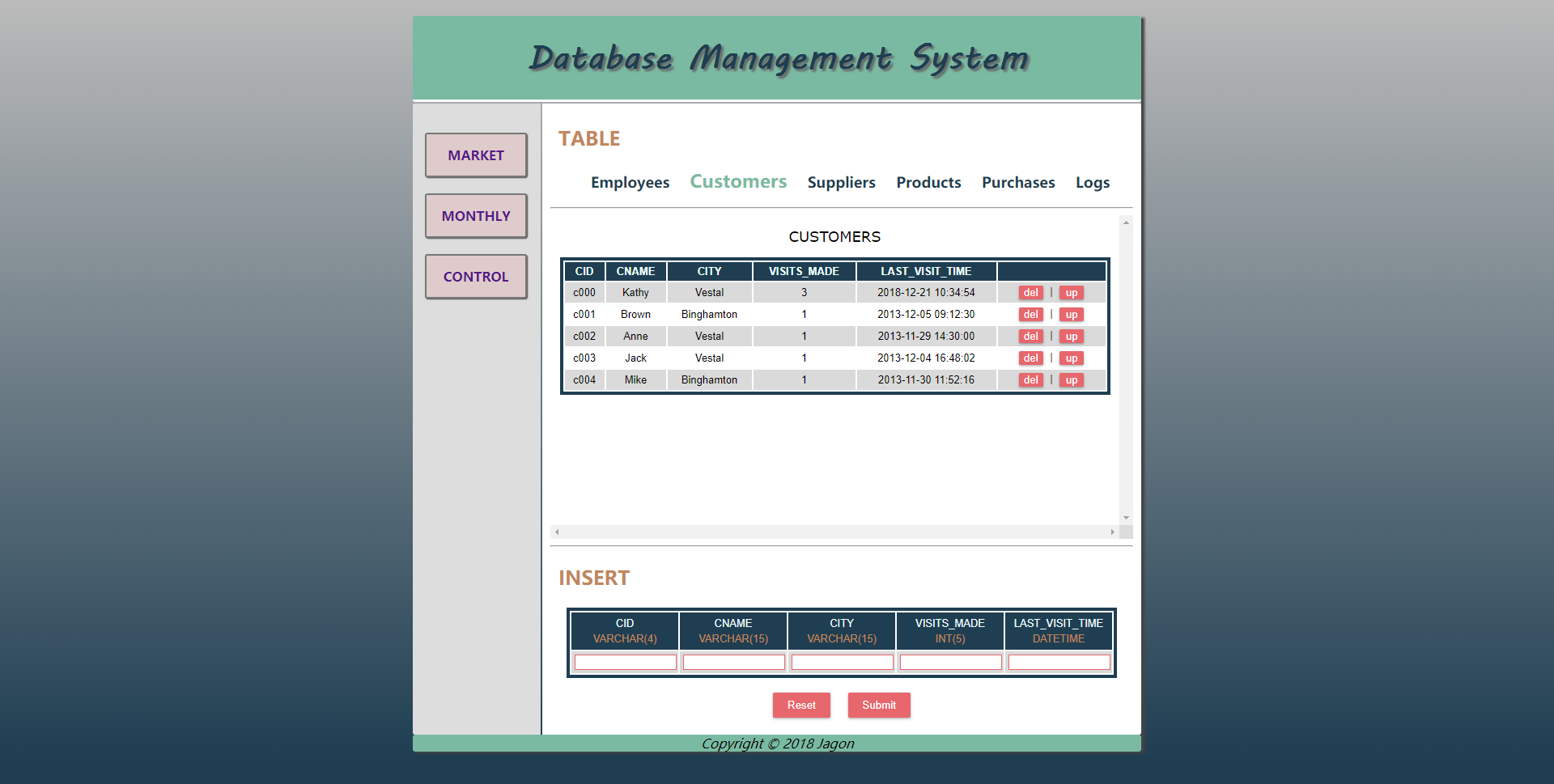


图1(b). 网页利用该存储过程显示表内容结果图

1. (4 points) Write a procedure to report the monthly sale information for any given product. For example, you can use a procedure, say report\_monthly\_sale(prod\_id), for this operation. For the given product id, you need to report the product name, the month (the first three letters of the month, e.g., FEB for February), year, the total quantity sold each month, the total dollar amount sold each month, and the average sale price (the total dollar amount divided by the total quantity) of each month. You need to list the information for only those months during which the given product has been purchased by some customers.

首先，写出一个转换月份为数字的函数：

DELIMITER //

DROP FUNCTION IF EXISTS MONTH\_NUM2ALPHA//

CREATE FUNCTION MONTH\_NUM2ALPHA(MON INT) RETURNS VARCHAR(3)

BEGIN

DECLARE result VARCHAR(3) DEFAULT '';

CASE MON

WHEN 1 THEN

SET result = 'JAN';

WHEN 2 THEN

SET result = 'FEB';

WHEN 3 THEN

SET result = 'MAR';

WHEN 4 THEN

SET result = 'APR';

WHEN 5 THEN

SET result = 'MAY';

WHEN 6 THEN

SET result = 'JUN';

WHEN 7 THEN

SET result = 'JUL';

WHEN 8 THEN

SET result = 'AUG';

WHEN 9 THEN

SET result = 'SEP';

WHEN 10 THEN

SET result = 'OCT';

WHEN 11 THEN

SET result = 'NOV';

WHEN 12 THEN

SET result = 'DEC';

ELSE

SET result = 'ERROR';

END CASE;

RETURN result;

END //

DELIMITER;

其次，利用自然连接写出打印表的存储过程：

DELIMITER //

DROP PROCEDURE IF EXISTS report\_monthly\_sale //

CREATE PROCEDURE report\_monthly\_sale(IN prod\_id VARCHAR(4))

BEGIN

SELECT PNAME, MONTH\_NUM2ALPHA(MONTH) AS MONTH, YEAR, SUM(QTY) AS TOTAL\_QUANTITY, SUM(TOTAL\_PRICE) AS TOTAL\_PRICE, SUM(TOTAL\_PRICE)/SUM(QTY) AS AVERAGE\_PRICE FROM  
(SELECT PNAME, MONTH(PTIME) AS MONTH, YEAR(PTIME) AS YEAR, QTY, TOTAL\_PRICE FROM  
PURCHASES NATURAL JOIN PRODUCTS WHERE PID= prod\_id)   
AS T GROUP BY YEAR, MONTH;

END //

DELIMITER ;

结果如下图2所示：

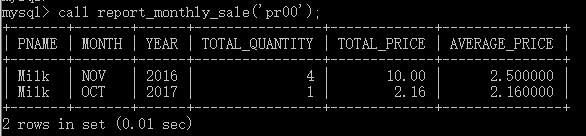


图2(a). report\_monthly\_sale运行结果图

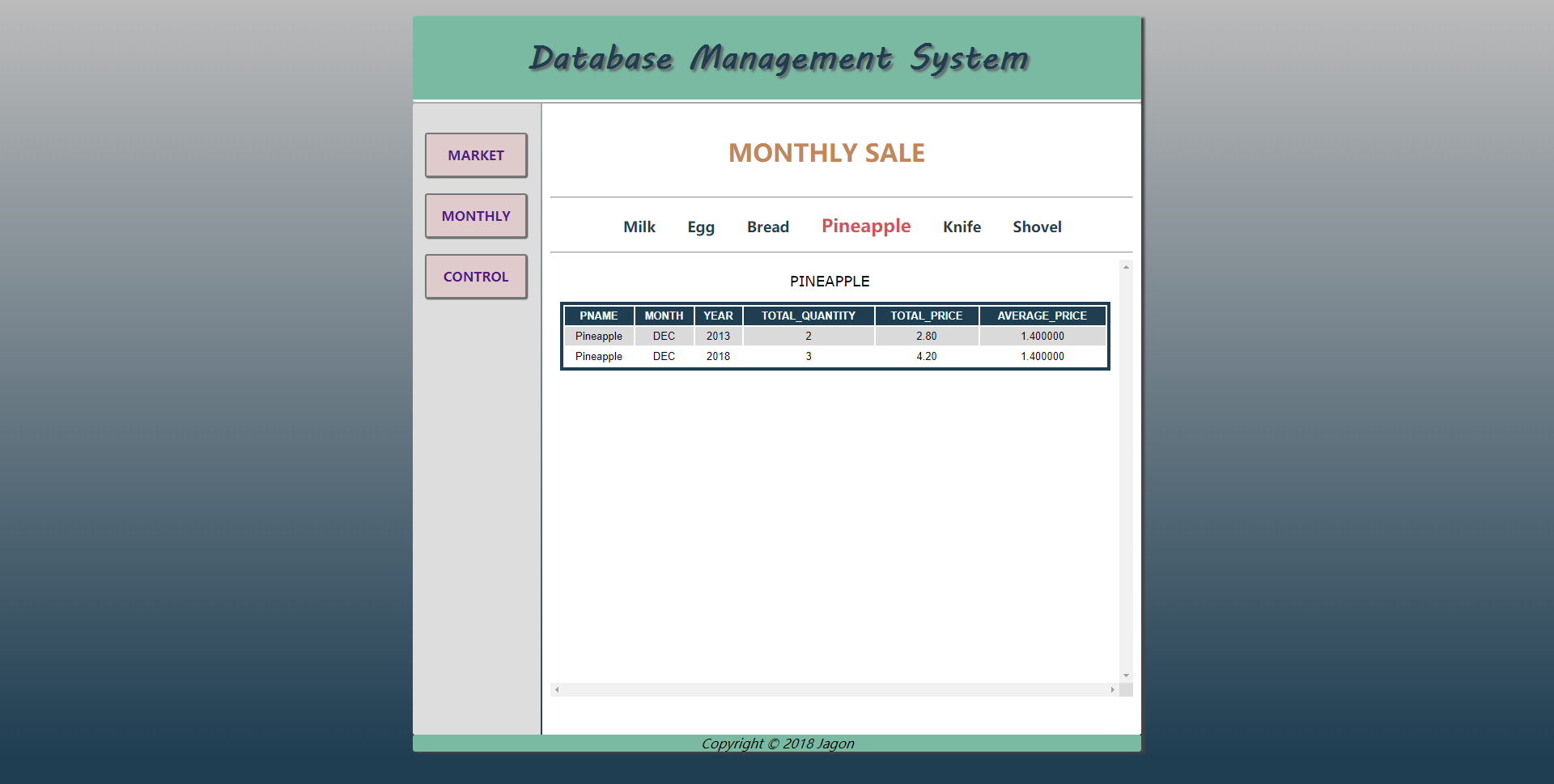


图2(b). 调用该存储过程显示月销量结果图

1. (7 points) Write procedures to add tuples into the purchases table and the products table. As an example, you can use a procedure, say add\_purchase(pur\_no, c\_id, e\_id, p\_id, pur\_qty), to add a tuple in the purchases table, where pur\_no, c\_id, e\_id, p\_id and pur\_qty are parameters of the procedure. Note that total\_price should be computed based on the data in the database automatically and ptime should be the current time (use current\_timestamp).

本存储过程主要是根据pid得到价格和折扣，其它只需使用insert就可以完成，代码如下：

DELIMITER //

DROP PROCEDURE IF EXISTS add\_purchase //

CREATE PROCEDURE add\_purchase(pur\_no VARCHAR(4), c\_id VARCHAR(4), e\_id VARCHAR(3), p\_id VARCHAR(4), pur\_qty INT)

BEGIN

DECLARE price DECIMAL(6,2);

DECLARE discount DECIMAL(3,2);

SELECT ORIGINAL\_PRICE FROM PRODUCTS WHERE PID=p\_id INTO price;

SELECT DISCNT\_RATE FROM PRODUCTS WHERE PID=p\_id INTO discount;

INSERT INTO PURCHASES VALUES(pur\_no, c\_id, e\_id, p\_id, pur\_qty, CURRENT\_TIMESTAMP(), pur\_qty\*price\*(1-discount));

END //

DELIMITER ;

运行结果如下所示：

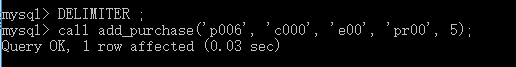


图3(a). add\_purchase运行结果图

插入后，结果正确：

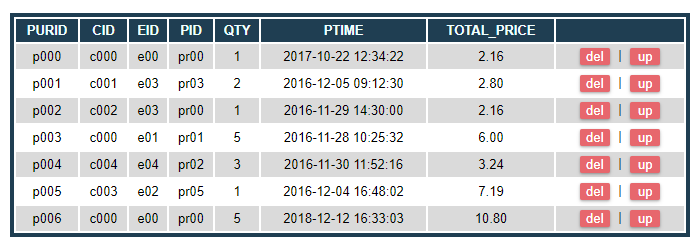


图3(b). add\_purchase运行结果图

1. (9 points) Add a tuple to the logs table automatically whenever any table is modified. To simplify, you are only required to consider the following modifications (events): (1) insert a tuple into the purchases table; (2) update the qoh attribute of the products table; and (3) update the visits\_made attribute of the customers table. When a tuple is added to the logs table due to the first event, the table\_name should be “purchases”, the operation should be “insert” and the key\_value should be the pur of the newly inserted purchase. When a tuple is added to the logs table due to the second event, the table\_name should be “products”, the operation should be “update” and the key\_value should be the pid of the affected product. When a tuple is added to the logs table due to the third event, the table\_name should be “customers”, the operation should be “update” and the key\_value should be the cid of the affected customer. Adding tuples to the logs table should be implemented using triggers. You need to implement three triggers for this task, one for each event.

按要求依次建立三个触发器，使用connection\_id()作为logid，user()的前半段作为who的值，其余按照实验要求添加即可，源代码如下所示：

CREATE TRIGGER m\_purchases AFTER INSERT ON PURCHASES

FOR EACH ROW

INSERT INTO LOGS VALUES(connection\_id(), left(user(), instr(user(), '@')-1), CURRENT\_TIMESTAMP(), 'Purchases', 'Insert', NEW.PURID);

CREATE TRIGGER m\_products AFTER UPDATE ON PRODUCTS

FOR EACH ROW

INSERT INTO LOGS VALUES(connection\_id(), left(user(), instr(user(), '@')-1), CURRENT\_TIMESTAMP(), 'Products', 'Update', NEW.PID);

CREATE TRIGGER m\_customers AFTER UPDATE ON CUSTOMERS

FOR EACH ROW

INSERT INTO LOGS VALUES(connection\_id(), left(user(), instr(user(), '@')-1), CURRENT\_TIMESTAMP(), 'Customers', 'Update', NEW.CID);

添加了这三条trigger和后面购买时对客户购买时间和库存更新的trigger后，每当购买时，logs都会增加三条记录，因为购买时同时触发了插入purchases表，更新customer表和更新products表事件，如下图所示，该图是输入  
“call add\_purchase('p008', 'c000', 'e00', 'pr00', 15);”命令后的结果：

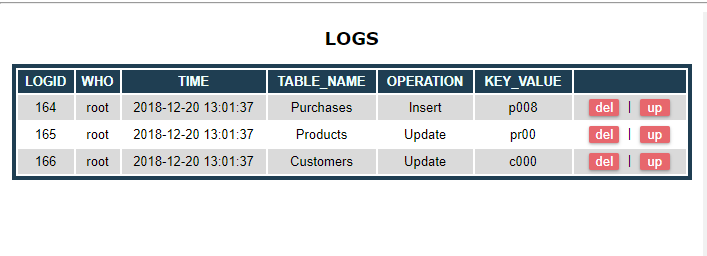


图4. 增加trigger后购买结果图

1. (4 points) Before a purchase is actually made (i.e., before a tuple is added into the purchases table), your program 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.

我们只需要在add\_purchase函数中修改，增加一个判断条件即可，即用户选择的pur\_qty如果大于库存qoh，则不插入，并打印提示信息，更改后代码如下：

DELIMITER //

DROP PROCEDURE IF EXISTS add\_purchase //

CREATE PROCEDURE add\_purchase(pur\_no VARCHAR(4), c\_id VARCHAR(4), e\_id VARCHAR(3), p\_id VARCHAR(4), pur\_qty INT)

BEGIN

DECLARE price DECIMAL(6,2);

DECLARE discount DECIMAL(3,2);

DECLARE theqoh INT;

SELECT qoh from PRODUCTS WHERE PID=p\_id INTO theqoh;

SELECT ORIGINAL\_PRICE FROM PRODUCTS WHERE PID=p\_id INTO price;

SELECT DISCNT\_RATE FROM PRODUCTS WHERE PID=p\_id INTO discount;

IF pur\_qty <= theqoh THEN

INSERT INTO PURCHASES VALUES(pur\_no, c\_id, e\_id, p\_id, pur\_qty, CURRENT\_TIMESTAMP(), pur\_qty\*price\*(1-discount));

ELSE

CALL show\_message('Insufficient quantity in stock.');

END IF;

END //

DELIMITER ;

运行超过库存时运行截图如下：

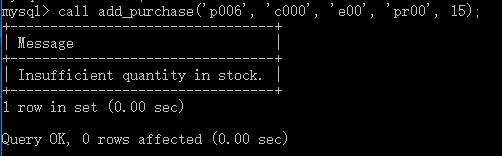


图5(a). 购买数量超过库存时结果示意图

若没超过，则正常购买：



图5(b). 成功购买结果示意图

以下截图是在网页购买时，不同情况下的提示信息：

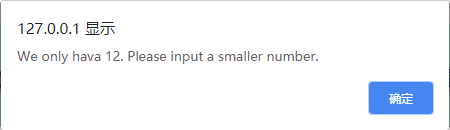


图5(c). 购买数量超过库存时网页结果示意图



图5(d). 网页成功购买结果示意图

1. (16 points) 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 program should perform the following tasks: (1) print a message indicating the current qoh of the product, (b) increase qoh by making it 2 \* old\_qoh, where old\_qoh represents the value of qoh before the corresponding purchase was made (other attribute values of the product will not be changed), and (c) print another message indicating that the quantity on hand of the product has been increased by old\_qoh + qty\_sold, where qty\_sold is the number of the product sold in the involved purchase. In addition, the insertion of the new tuple in the purchases table will cause the visits\_made of the customer to be increased by one. Use triggers to implement the update of qoh, printing of the messages and the update of visits\_made and last\_visit\_time.

写出触发器，当qoh低于qoh\_thredshold的时候，qoh变为原来的两倍，同时更改customers的时间和次数，代码如下：

DELIMITER //

DROP TRIGGER IF EXISTS qoh\_purchases //

CREATE TRIGGER qoh\_purchases AFTER INSERT ON PURCHASES

FOR EACH ROW

BEGIN

DECLARE old\_qoh INT;

DECLARE thredshold INT;

DECLARE new\_qoh INT;

DECLARE vm INT;

DECLARE lvt DATETIME;

SELECT QOH, QOH\_THRESHOLD FROM PRODUCTS WHERE PID = NEW.PID INTO old\_qoh, thredshold;

SET new\_qoh = old\_qoh - NEW.QTY;

IF new\_qoh < thredshold THEN

SET new\_qoh = old\_qoh\*2;

END IF;

UPDATE PRODUCTS SET QOH = new\_qoh WHERE PID = NEW.PID;

SELECT VISITS\_MADE FROM CUSTOMERS WHERE CID = NEW.CID INTO vm;

UPDATE CUSTOMERS SET VISITS\_MADE = vm + 1, LAST\_VISIT\_TIME = CURRENT\_TIMESTAMP() WHERE CID = NEW.CID;

END //

DELIMITER ;

如图所示，增加了该trigger后，当我们购买正常的物品时，customer是last\_visit\_time和product的qoh都发生了改变：

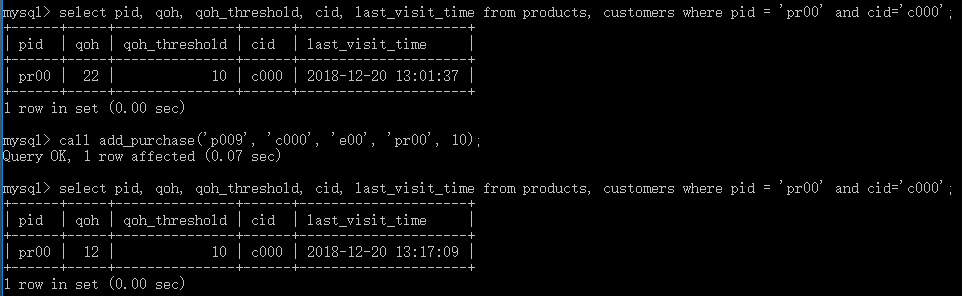


图6(a). 正常购买时结果示意图

当购买后库存小于qoh\_threshold时，qoh变为原来的两倍：

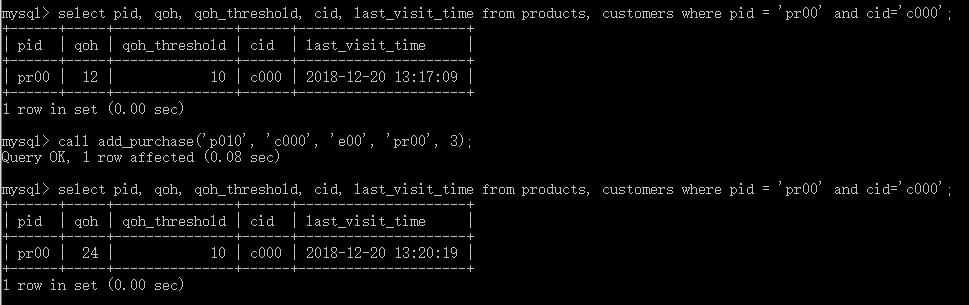


图6(b). 购买后qoh小于qoh\_threshold时结果示意图

在网页时，若购买后数量小于阈值，则会有以下信息提示。该信息是牛奶原数量为24，购买数量为22情况下的提示信息：



图6(c). 购买后qoh小于qoh\_threshold时网页提示示意图



图6(d). 购买后qoh小于qoh\_threshold时网页提示示意图

1. (4 points) You need to make your code user friendly by designing and displaying appropriate messages for all exceptions. For example, if someone wants to find the purchases of a customer but entered a non-existent customer id, your program should report the problem clearly.

如下图所示，当用户输入错误的cid，eid或者过大的购买数量时，都会对其进行相对应的提醒：

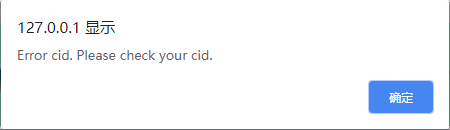


图7(a). 输入错误cid时提醒示意图

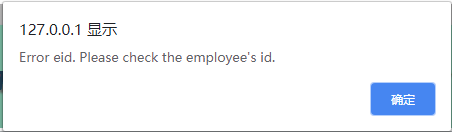


图7(b). 输入错误eid时提醒示意图

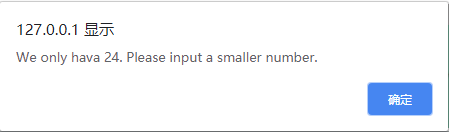


图7(b). 输入过多购买数量时提醒示意图

**3. Interface (35 points)**

Implement a Web interactive interface using PHP. Your interface program should utilize as many of your MySQL stored procedures/functions as possible.

首先是对各个物品的购买功能：

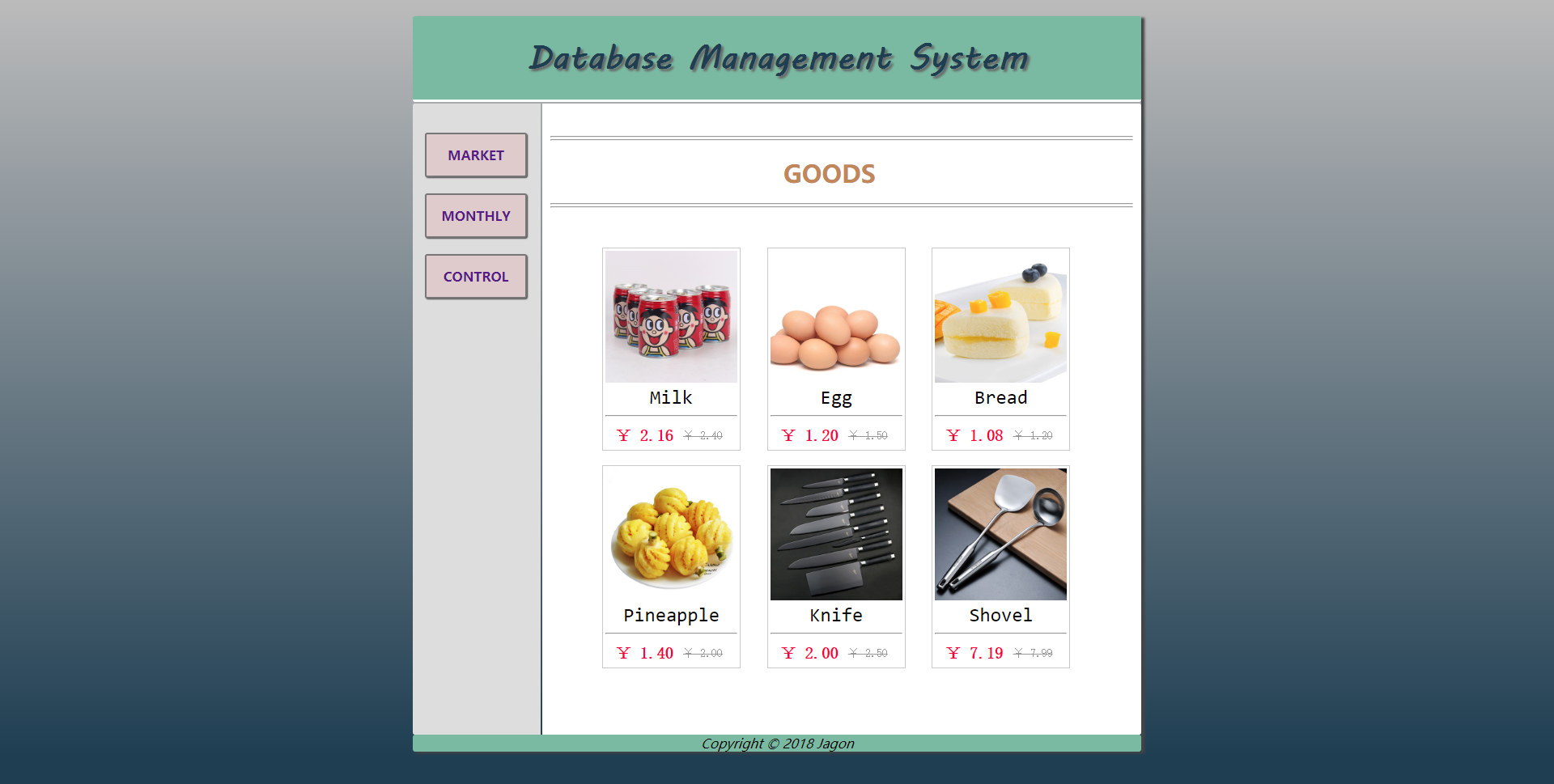


图8(a). 购买接口示意图

其次是每个商品的月销量情况：

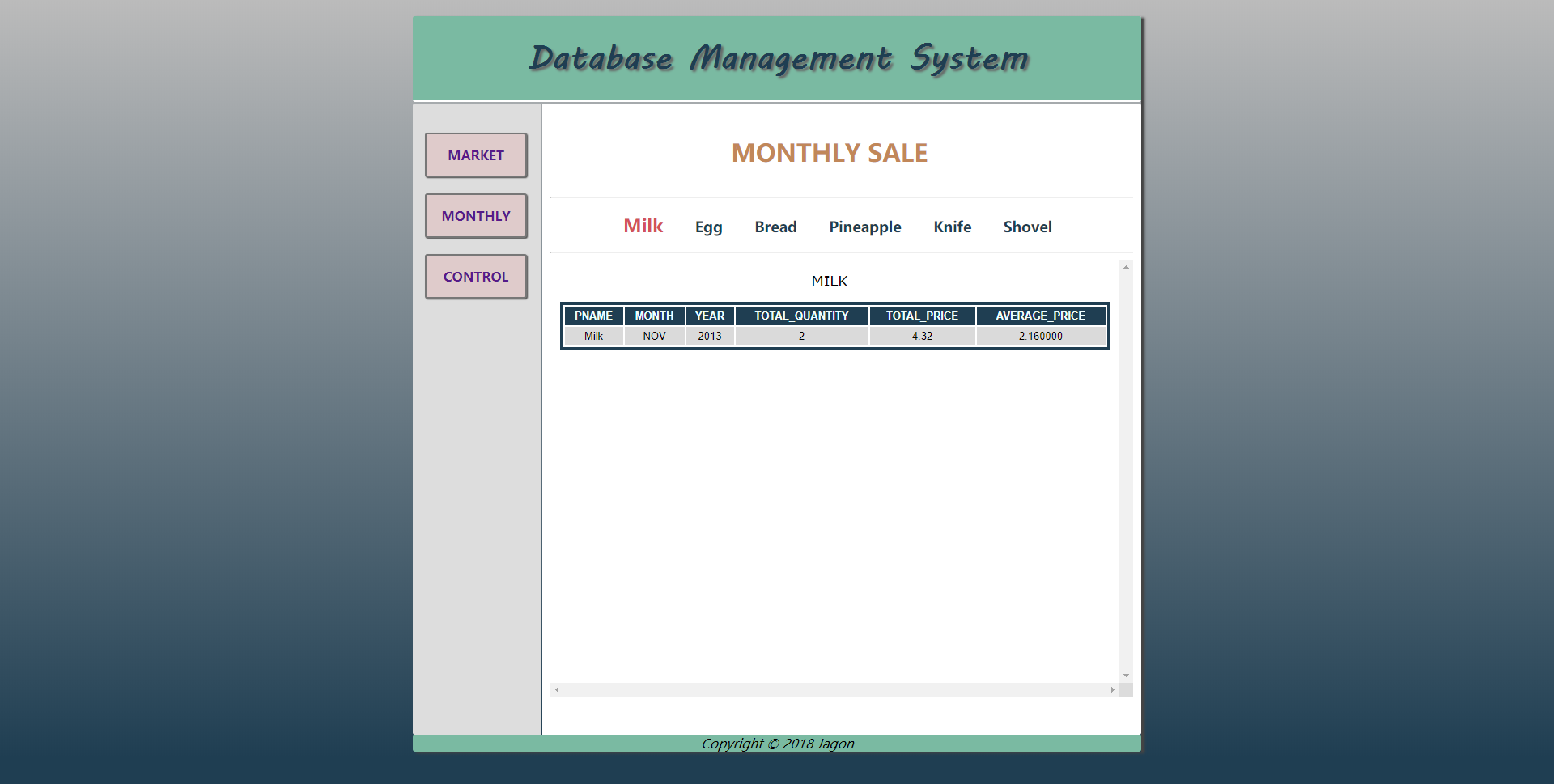


图8(b). 商品月销量示意图

最后是对各个表提供查看、修改、插入以及删除功能：

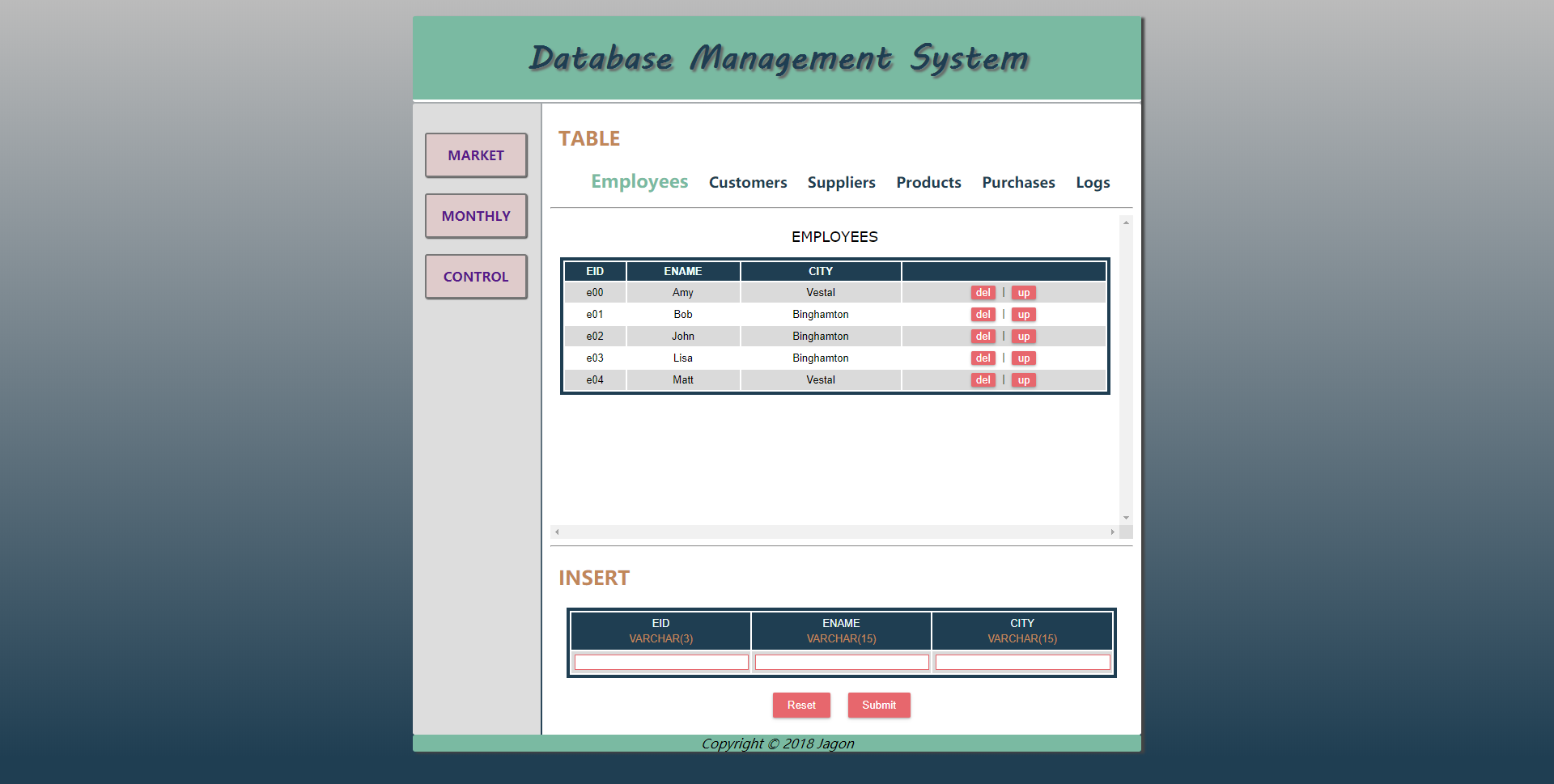


图8(c). 各个表展示示意图

**4. Documentation (10 points)**

Documentation consists of the following aspects:

1. Each procedure and function and every other object you create for your project needs to be explained clearly regarding its objective and usage.
2. Your code needs to be well documented with in-line comments.

**5. Hand-ins, Demo and Grading**

1. You will also need to submit your source code along with your documentation to the Blackboard.
2. It is required to demonstrate your project to the instructor using tuples created by the instructor. More instructions on demo will be given before the demo.
3. The grading will be based on the quality of your code, the documentation and on how successful of your demo is.