**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH**

🙞∙∙∙☼∙∙∙🙜



BÀI TẬP LỚN SỐ 2

**RESTAURANT DATABASE SYSTEM**

MÔN: HỆ CƠ SỞ DỮ LIỆU (CO2013)

LỚP L02 – NHÓM: L02.E

GVHD: Trương Huỳnh Chi, Lê Thị Bảo Thu

SVTH: Nguyễn Minh Bảo – 1912676

Phạm Đại Hoàng An – 1912539

Phạm Ngọc Tân – 1915071

Nguyễn Phước Toàn – 1915551

Lê Hồ Hoài Bão - 1912698

**Thành phố Hồ Chí Minh – Tháng 12/2021**

MỤC LỤC

[I. PHẦN CHUNG 3](#_Toc89108804)

[1. Các câu lệnh tạo bảng và ràng buộc: 3](#_Toc89108805)

[2. Các câu lệnh tạo chỉ mục: 3](#_Toc89108806)

[3. Các câu lệnh insert dữ liệu: 3](#_Toc89108807)

[4. Màn hình hiển thị 3](#_Toc89108808)

[II. PHẦN RIÊNG 3](#_Toc89108809)

[1. Thành viên: Nguyễn Minh Bảo – MSSV: 1912676 3](#_Toc89108810)

[2. Thành viên: Phạm Đại Hoàng An – MSSV: 1912539 3](#_Toc89108811)

[3. Thành viên: Phạm Ngọc Tân – MSSV: 1915071 8](#_Toc89108812)

[4. Thành viên: Nguyễn Phước Toàn – MSSV: 1915551 8](#_Toc89108813)

[5. Thành viên: Lê Hồ Hoài Bão – MSSV: 1912698 8](#_Toc89108814)

[III. PHỤ LỤC 8](#_Toc89108815)

# PHẦN CHUNG

## Các câu lệnh tạo bảng và ràng buộc:

CREATE TABLE EMPLOYEE(

    ID CHAR(9) NOT NULL PRIMARY KEY,

    Name VARCHAR(255) NOT NULL,

    Email VARCHAR(255) NOT NULL,

    Phone CHAR(10) NOT NULL,

    Bdate DATE NOT NULL,

    Sex VARCHAR(255) NOT NULL,

    Address VARCHAR(255) NOT NULL,

    Salary DECIMAL(10,2) NOT NULL,

    Eaccount VARCHAR(255),

    Bnumber INT NOT NULL

  --  FOREIGN KEY(Eaccount) REFERENCES ACCOUNT(Username)

);

CREATE TABLE MANAGER(

    ID CHAR(9) NOT NULL PRIMARY KEY

  --  FOREIGN KEY(ID) REFERENCES EMPLOYEE(ID)

);

CREATE TABLE SHIFT(

    EID CHAR(9) NOT NULL,

    Day\_of\_week VARCHAR(32) NOT NULL,

    Session VARCHAR(32) NOT NULL,

    Day VARCHAR(32) NOT NULL,

    PRIMARY KEY(EID,Day\_of\_week,Session,Day)

   -- FOREIGN KEY(EID) REFERENCES EMPLOYEE(ID)

);

CREATE TABLE SERVERS(

    ID CHAR(9) NOT NULL PRIMARY KEY,

    Work\_experience VARCHAR(255) NULL

  --  FOREIGN KEY(ID) REFERENCES EMPLOYEE(ID)

);

-- Bảo

CREATE TABLE ACCOUNT(

    Username VARCHAR(255) NOT NULL PRIMARY KEY,

    Password INT NOT NULL,

    Email VARCHAR(255) NOT NULL,

    Access\_right VARCHAR(255) NOT NULL

);

CREATE TABLE COOK(

    ID CHAR(9) NOT NULL PRIMARY KEY,

    Years\_of\_experience INT  NULL,

    Chef\_ID CHAR(9)

);

CREATE TABLE COOKS(

    ID CHAR(9) NOT NULL,

    Onumber INT NOT NULL,

    PRIMARY KEY (ID,Onumber)

);

CREATE TABLE CASHIER(

    ID CHAR(9) NOT NULL PRIMARY KEY

);

-- Tân

CREATE TABLE DEPENDENT

(

    Emp\_ID CHAR(9) NOT NULL,

    Dependent\_name VARCHAR(15) NOT NULL,

    Bdate DATE,

    Address VARCHAR(255),

    Sex VARCHAR(255),

    Relationship VARCHAR(255) NOT NULL,

    Phone CHAR(10) NOT NULL,

    PRIMARY KEY (Emp\_ID, Dependent\_name)

);

CREATE TABLE PROCESSES

(

    Onumber INT NOT NULL,

    CashierID CHAR(9) NOT NULL,

    PRIMARY KEY (Onumber)

);

CREATE TABLE FOOD\_ORDER

(

    Number INT NOT NULL PRIMARY KEY,

    Status VARCHAR(255) NOT NULL

);

CREATE TABLE INCLUDES

(

    Onum INT NOT NULL,

    Dname VARCHAR(255) NOT NULL,

    Ord\_quantity INT NOT NULL,

    Listed\_price DECIMAL(10,2) NOT NULL,

    PRIMARY KEY (Onum, Dname)

);

-- Toàn

CREATE TABLE CUSTOMER(

    ID CHAR(9) NOT NULL,

    Name VARCHAR(255) NOT NULL,

    Phone CHAR(10) NOT NULL,

    Accumulated\_point INT NULL,

    PRIMARY KEY (ID)

);

CREATE TABLE HAS\_ACC(

    Customer\_ID CHAR(9) NOT NULL,

    Username VARCHAR(255) NOT NULL,

    PRIMARY KEY (Customer\_ID)

   -- FOREIGN KEY (Customer\_ID) REFERENCES Customer(ID),

   -- FOREIGN KEY (Username) REFERENCES Account(Username)

);

CREATE TABLE BRANCH(

    Number INT NOT NULL,

    Name VARCHAR(255) NOT NULL,

    Location VARCHAR(255),

    Phone CHAR(10) NOT NULL,

    Manager\_ID CHAR(9) NOT NULL,

    Quantity\_Employee INT NOT NULL DEFAULT 0,

    PRIMARY KEY (Number)

   -- FOREIGN KEY (Manager\_ID) REFERENCES Manager(ID)

);

CREATE TABLE HAS\_FOOD(

    Dish\_name VARCHAR(255) NOT NULL,

    Number INT NOT NULL,

    Available\_quantity INT NOT NULL,

    PRIMARY KEY (Dish\_name, Number)

-- FOREIGN KEY (Dish\_name) REFERENCES Dish(Name),

 --   FOREIGN KEY (Number) REFERENCES Branch(Number)

);

-- Bão

CREATE TABLE BILL

(

    Number INT NOT NULL,

    Order\_Num INT NOT NULL,

    Customer\_name VARCHAR(255) NOT NULL,

    Tax DECIMAL(3,2) NOT NULL,

    PRIMARY KEY (Number)

    -- FOREIGN KEY (Order\_Num) REFERENCES ORDER (Number)

);

CREATE TABLE BUY\_ONLINE

(

    ONumber INT NOT NULL,

    Username VARCHAR(255) NOT NULL,

    PRIMARY KEY (ONumber)

    -- FOREIGN KEY (ONumber) REFERENCES ORDER (Number),

    -- FOREIGN KEY (Username) REFERENCES ACCOUNT (Username)

);

CREATE TABLE BUY\_DIRECTLY

(

    ONumber INT NOT NULL,

    CustomerID CHAR(9) NOT NULL,

    PRIMARY KEY (ONumber)

    -- FOREIGN KEY (ONumber) REFERENCES ORDER (Number),

    -- FOREIGN KEY (CustomerID) REFERENCES CUSTOMER (ID)

);

CREATE TABLE DISH

(

    Name VARCHAR(255) NOT NULL,

    Price DECIMAL(10,2) NOT NULL,

    PRIMARY KEY (Name)

);

-- Foreign key Bảo

ALTER TABLE COOK

ADD FOREIGN KEY (Chef\_ID) REFERENCES COOK(ID) ;

ALTER TABLE COOK

ADD FOREIGN KEY (ID) REFERENCES EMPLOYEE(ID) ;

ALTER TABLE COOKS

ADD FOREIGN KEY (ID) REFERENCES COOK(ID) ;

ALTER TABLE COOKS

ADD FOREIGN KEY (Onumber) REFERENCES FOOD\_ORDER(Number) ;

-- Foreign key An

ALTER TABLE EMPLOYEE

ADD FOREIGN KEY(Eaccount) REFERENCES ACCOUNT(Username) ,

ADD FOREIGN KEY(Bnumber) REFERENCES BRANCH(Number) ;

ALTER TABLE MANAGER

ADD FOREIGN KEY(ID) REFERENCES EMPLOYEE(ID) ;

ALTER TABLE SHIFT

ADD FOREIGN KEY(EID) REFERENCES EMPLOYEE(ID) ;

ALTER TABLE SERVERS

ADD FOREIGN KEY(ID) REFERENCES EMPLOYEE(ID) ;

-- Foreign key Tan

ALTER TABLE DEPENDENT

ADD FOREIGN KEY (Emp\_ID) REFERENCES EMPLOYEE(ID);

ALTER TABLE PROCESSES

ADD FOREIGN KEY (CashierID) REFERENCES CASHIER(ID),

ADD FOREIGN KEY (Onumber) REFERENCES FOOD\_ORDER(Number) ;

ALTER TABLE INCLUDES

ADD FOREIGN KEY (Dname) REFERENCES DISH(Name),

ADD FOREIGN KEY (Onum) REFERENCES FOOD\_ORDER(Number) ;

-- Foreign key Toan

ALTER TABLE HAS\_ACC

ADD  FOREIGN KEY (Customer\_ID) REFERENCES CUSTOMER(ID),

ADD  FOREIGN KEY (Username) REFERENCES ACCOUNT(Username) ;

ALTER TABLE HAS\_FOOD

ADD FOREIGN KEY (Dish\_name) REFERENCES DISH(Name) ,

ADD FOREIGN KEY (Number) REFERENCES BRANCH(Number) ;

ALTER TABLE BRANCH

ADD FOREIGN KEY (Manager\_ID) REFERENCES MANAGER(ID) ;

-- Foreign key Bão

ALTER TABLE BILL

ADD FOREIGN KEY (Order\_Num) REFERENCES FOOD\_ORDER(Number) ;

ALTER TABLE BUY\_ONLINE

ADD FOREIGN KEY (ONumber) REFERENCES FOOD\_ORDER (Number) ,

ADD FOREIGN KEY (Username) REFERENCES ACCOUNT (Username) ;

ALTER TABLE BUY\_DIRECTLY

ADD FOREIGN KEY (ONumber) REFERENCES FOOD\_ORDER (Number) ,

ADD FOREIGN KEY (CustomerID) REFERENCES CUSTOMER (ID) ;

-- Phần chỉnh sửa Tân

ALTER TABLE DISH

ADD Image\_Link Text;

## Các câu lệnh tạo chỉ mục:

CREATE INDEX indexDishNam

ON dish (name);

CREATE INDEX indexBranchName

ON branch (Name);

CREATE INDEX indexEmployeeID

ON employee (ID);

CREATE INDEX indexBranchName

ON includes (Onum, Dname);

## Các câu lệnh insert dữ liệu:

INSERT INTO MANAGER VALUES ('200000005');

INSERT INTO MANAGER VALUES ('100000005');

INSERT INTO MANAGER VALUES ('100000001');

INSERT INTO MANAGER VALUES ('100000002');

INSERT INTO MANAGER VALUES ('100000003');

INSERT INTO BRANCH VALUES (1, 'HBT', '35 Hai Ba Trung', '090789001', '100000005',0);

INSERT INTO BRANCH VALUES (2, 'PCT', '411 Phan Chu Trinh', '090789002', '200000005',0);

INSERT INTO BRANCH VALUES (3, 'PCT', '41 Hoang Dieu', '090711111', '100000001',0);

INSERT INTO BRANCH VALUES (4, 'PCT', '11 Vo Thi Sau', '090722222', '100000002',0);

INSERT INTO EMPLOYEE VALUE('100000001','Tran Van B','abc1@gmail.com','0903334441','2001-01-02','Nam','104 Thu Duc,TP Ho Chi Minh',6000000,'hoaibaoTV4',1);

INSERT INTO EMPLOYEE VALUE('100000002','Tran Thi C','abc2@gmail.com','0903334442','2001-01-02','Nu','105 Thu Duc,TP Ho Chi Minh',6000000,'hoaibaoTV4-1',1);

INSERT INTO EMPLOYEE VALUE('100000003','Tran Van D','abc3@gmail.com','0903334443','2001-01-02','Nam','106 Thu Duc,TP Ho Chi Minh',6000000,'hoaibaoTV4-2',1);

INSERT INTO EMPLOYEE VALUE('100000004','Tran Van E','abc4@gmail.com','0903334444','2001-01-02','Nam','107 Thu Duc,TP Ho Chi Minh',6000000,'hoaibaoTV4-3',1);

INSERT INTO EMPLOYEE VALUE('100000005','Tran Van F','abc5@gmail.com','0903334445','2001-01-02','Nam','108 Thu Duc,TP Ho Chi Minh',8000000,'hoanganTV3',1);

INSERT INTO EMPLOYEE VALUE('200000001','Tran Thi A','abc6@gmail.com','0903334441','2001-01-22','Nu','202 Quan6,TP Ho Chi Minh',6000000,'hoaibaoTV4',2);

INSERT INTO EMPLOYEE VALUE('200000002','Tran Nam B','abc7@gmail.com','0903334411','2001-01-22','Nam','203 Quan6,TP Ho Chi Minh',6000000,'hoaibaoTV4-10',2);

INSERT INTO EMPLOYEE VALUE('200000003','Tran Nam C','abc8@gmail.com','0903334421','2001-01-22','Nam','204 Quan6,TP Ho Chi Minh',6000000,'hoaibaoTV4-20',2);

INSERT INTO EMPLOYEE VALUE('200000004','Tran Nam D','abc9@gmail.com','0903334431','2001-01-22','Nam','205 Quan6,TP Ho Chi Minh',6000000,'hoaibaoTV4-30',2);

INSERT INTO EMPLOYEE VALUE('200000005','Tran Thi E','abc0@gmail.com','0903334451','2001-01-22','Nu','206 Quan6,TP Ho Chi Minh',6000000,'hoanganTV30',2);

INSERT INTO SHIFT VALUES ('100000005','T2','Sang','07');

INSERT INTO SHIFT VALUES ('100000005','T3','Chieu','08');

INSERT INTO SHIFT VALUES ('100000005','T4','Sang','09');

INSERT INTO SHIFT VALUES ('200000005','T2','Sang','07');

INSERT INTO SHIFT VALUES ('200000005','T3','Sang','08');

INSERT INTO SHIFT VALUES ('200000005','T4','Chieu','09');

INSERT INTO SERVERS VALUES ('100000002',null);

INSERT INTO SERVERS VALUES ('200000002',null);

INSERT INTO DISH VALUES ("Cháo Ếch Singapore", 45000, "https://images.foody.vn/res/g108/1072266/prof/s640x400/foody-upload-api-foody-mobile-bia-210406100724.jpg");

INSERT INTO DISH VALUES ("Chân Gà Ngâm Sả Tắc", 70000, "https://images.foody.vn/res/g106/1052099/prof/s640x400/foody-upload-api-foody-mobile-chan-ga-sa-tac-500-201021160112.jpg");

INSERT INTO DISH VALUES ("Bánh Tráng Mỡ Hành Tóp Mỡ", 25000, "https://images.foody.vn/res/g97/960254/prof/s640x400/file\_restaurant\_photo\_0pr0\_16348-af99b606-211022145721.jpg");

INSERT INTO DISH VALUES ("Hủ Tiếu Nam Vang", 35000, "https://images.foody.vn/res/g70/699162/prof/s640x400/foody-upload-api-foody-mobile-foody-upload-api-foo-200728152955.jpg");

INSERT INTO DISH VALUES ("Bún Chả Lá Lốt", 30000, "https://images.foody.vn/res/g106/1050668/prof/s640x400/foody-upload-api-foody-mobile-36-201012145559.jpg");

INSERT INTO DISH VALUES ("Bánh Mì Hà Nội", 25000, "https://images.foody.vn/res/g109/1082517/prof/s640x400/foody-upload-api-foody-mobile-cv-2bbb053f-210614103407.jpeg");

INSERT INTO DISH VALUES ("Hamburger Bò Teriyaki", 40000, "https://images.foody.vn/res/g109/1081665/prof/s640x400/foody-upload-api-foody-mobile-ha-e5acab07-210607120946.jpeg");

INSERT INTO DISH VALUES ("Cơm Chiên Dương Châu", 32000, "https://images.foody.vn/res/g97/966840/prof/s640x400/image-be32b319-200910114138.jpeg");

INSERT INTO DISH VALUES ("Mì Trộn Muối Ớt", 37000, "https://images.foody.vn/res/g107/1060971/prof/s640x400/file\_restaurant\_photo\_zc3t\_16099-7f8de683-210106212421.jpg");

INSERT INTO DISH VALUES ("Gà Sốt Sệt", 180000, "https://images.foody.vn/res/g106/1053558/prof/s640x400/file\_restaurant\_photo\_ioed\_16213-6877b5bd-210518123042.jpeg");

INSERT INTO HAS\_FOOD VALUES ("Cháo Ếch Singapore", 1, 30);

INSERT INTO HAS\_FOOD VALUES ("Bánh Tráng Mỡ Hành Tóp Mỡ", 1, 50);

INSERT INTO HAS\_FOOD VALUES ("Hủ Tiếu Nam Vang", 1, 35);

INSERT INTO HAS\_FOOD VALUES ("Bún Chả Lá Lốt", 1, 37);

INSERT INTO HAS\_FOOD VALUES ("Bánh Mì Hà Nội", 1, 60);

INSERT INTO HAS\_FOOD VALUES ("Hamburger Bò Teriyaki", 1, 25);

INSERT INTO HAS\_FOOD VALUES ("Gà Sốt Sệt", 1, 22);

INSERT INTO HAS\_FOOD VALUES ("Chân Gà Ngâm Sả Tắc", 2, 45);

INSERT INTO HAS\_FOOD VALUES ("Bánh Tráng Mỡ Hành Tóp Mỡ", 2, 30);

INSERT INTO HAS\_FOOD VALUES ("Hủ Tiếu Nam Vang", 2, 20);

INSERT INTO HAS\_FOOD VALUES ("Bún Chả Lá Lốt", 2, 28);

INSERT INTO HAS\_FOOD VALUES ("Cơm Chiên Dương Châu", 2, 17);

INSERT INTO HAS\_FOOD VALUES ("Mì Trộn Muối Ớt", 2, 33);

INSERT INTO HAS\_FOOD VALUES ("Cháo Ếch Singapore", 3, 18);

INSERT INTO HAS\_FOOD VALUES ("Chân Gà Ngâm Sả Tắc", 3, 27);

INSERT INTO HAS\_FOOD VALUES ("Hủ Tiếu Nam Vang", 3, 31);

INSERT INTO HAS\_FOOD VALUES ("Bún Chả Lá Lốt", 3, 25);

INSERT INTO HAS\_FOOD VALUES ("Bánh Mì Hà Nội", 3, 39);

INSERT INTO HAS\_FOOD VALUES ("Mì Trộn Muối Ớt", 3, 42);

INSERT INTO HAS\_FOOD VALUES ("Gà Sốt Sệt", 3, 7);

INSERT INTO HAS\_FOOD VALUES ("Cháo Ếch Singapore", 4, 23);

INSERT INTO HAS\_FOOD VALUES ("Chân Gà Ngâm Sả Tắc", 4, 35);

INSERT INTO HAS\_FOOD VALUES ("Bánh Tráng Mỡ Hành Tóp Mỡ", 4, 46);

INSERT INTO HAS\_FOOD VALUES ("Bánh Mì Hà Nội", 4, 36);

INSERT INTO HAS\_FOOD VALUES ("Cơm Chiên Dương Châu", 4, 21);

INSERT INTO HAS\_FOOD VALUES ("Mì Trộn Muối Ớt", 4, 29);

INSERT INTO `food\_order` (`Number`, `Status`) VALUES

(1, 'Chưa xử lý'),

(2, 'Chưa xử lý'),

(3, 'Đang xử lý'),

(4, 'Đã hoàn thành'),

(5, 'Đã hoàn thành');

INSERT INTO includes (Onum, Dname, Ord\_quantity, Listed\_price) VALUES

(1, 'Bánh Mì Hà Nội', 2, 50000),

(1, 'Gà Sốt Sệt', 1, 180000),

(1, 'Mì Trộn Muối Ớt', 3, 111000),

(2, 'Mì Trộn Muối Ớt', 1, 37000),

(2, 'Cơm Chiên Dương Châu', 1, 32000),

(3, 'Bánh Tráng Mỡ Hành Tóp Mỡ', 2, 50000),

(3, 'Bún Chả Lá Lốt', 1, 30000),

(3, 'Cháo Ếch Singapore', 2, 90000),

(3, 'Chân Gà Ngâm Sả Tắc', 3, 210000),

(3, 'Hamburger Bò Teriyaki', 2, 80000),

(4, 'Gà Sốt Sệt', 2, 360000),

(4, 'Hủ Tiếu Nam Vang', 2, 70000),

(4, 'Mì Trộn Muối Ớt', 4, 148000),

(5, 'Bánh Mì Hà Nội', 2, 50000),

(5, 'Hủ Tiếu Nam Vang', 1, 35000);

INSERT INTO BILL VALUES (1, 4, 'Nguyen Van B', 0.05);

INSERT INTO BILL VALUES (2, 5, 'Pham Thi C', 0.05);

INSERT INTO BUY\_ONLINE VALUES (1, 'minhbaoTV1');

INSERT INTO BUY\_ONLINE VALUES (3, 'ngoctanTV2');

INSERT INTO BUY\_DIRECTLY VALUES (2, '000000001');

INSERT INTO BUY\_DIRECTLY VALUES (4, '000000002');

INSERT INTO BUY\_DIRECTLY VALUES (5, '000000003');

INSERT INTO `dependent` (`Emp\_ID`, `Dependent\_name`, `Bdate`, `Address`, `Sex`, `Relationship`, `Phone`) VALUES

('100000001', 'Phạm Hồng Hải', '1972-07-25', 'TP. Dĩ An, T. Bình Dương', 'Nam', 'Cha', '0549284506'),

('100000002', 'Phạm Hông Phước', '2001-05-10', 'TP. Dĩ An, T. Bình Dương', 'Nam', 'Em trai', '0457283905'),

('100000003', 'Trần Ánh Tuyết', '1977-11-30', 'TP. Dĩ An, T. Bình Dương', 'Nu', 'Mẹ', '0457239058'),

('100000004', 'Nguyễn Minh Tú', '1964-07-02', 'Quận 4, TP. HCM', 'Nam', 'Cha', '0456378377'),

('100000005', 'Trần Ngọc Trúc', '1995-05-30', 'TP. Dĩ An, T. Bình Dương', 'Nu', 'Vợ', '0567354671'),

('200000001', 'Trần Văn Bình', '1970-05-06', 'TP. Thủ Đức, TP.HCM', 'Nam', 'Cha', '0956234568'),

('200000002', 'Võ Thị Hoa', '1975-08-15', 'TP. Thủ Đức, TP.HCM', 'Nu', 'Mẹ', '0765224506'),

('200000003', 'Đỗ Thị Trang', '1998-02-15', 'TP. Thủ Đức, TP. HCM', 'Nu', 'Em gái', '0674890376'),

('200000004', 'Đỗ Văn Tân', '1965-03-15', 'TP. Thủ Đức, TP. HCM', 'Nam', 'Cha', '0768936471'),

('200000005', 'Hồ Minh Tâm', '1990-06-12', 'Quận 2, TP.HCM', 'Nam', 'Chồng', '0436287407');

INSERT INTO ACCOUNT

VALUE ('minhbaoTV1', 20052001, 'minhbao@gmail.com','customer');

INSERT INTO ACCOUNT

VALUE ('ngoctanTV2', 11022001, 'ngoctan@gmail.com','customer');

INSERT INTO ACCOUNT

VALUE ('hoanganTV3', 23092001, 'hoangan@gmail.com','admin');

INSERT INTO ACCOUNT

VALUE ('hoaibaoTV4', 01012001, 'hoaibao@gmail.com','employee');

INSERT INTO ACCOUNT

VALUE ('hoaibaoTV4-1', 01012001, 'hoaibao@gmail.com','employee');

INSERT INTO ACCOUNT

VALUE ('hoaibaoTV4-2', 01012001, 'hoaibao@gmail.com','employee');

INSERT INTO ACCOUNT

VALUE ('hoaibaoTV4-3', 01012001, 'hoaibao@gmail.com','employee');

INSERT INTO ACCOUNT

VALUE ('phuoctoanTV5', 15122001, 'phuoctoan@gmail.com','customer');

INSERT INTO ACCOUNT

VALUE ('hoanganTV30', 23092001, 'hoangan@gmail.com','admin');

INSERT INTO ACCOUNT

VALUE ('hoaibaoTV40', 01012001, 'hoaibao@gmail.com','employee');

INSERT INTO ACCOUNT

VALUE ('hoaibaoTV4-10', 01012001, 'hoaibao@gmail.com','employee');

INSERT INTO ACCOUNT

VALUE ('hoaibaoTV4-20', 01012001, 'hoaibao@gmail.com','employee');

INSERT INTO ACCOUNT

VALUE ('hoaibaoTV4-30', 01012001, 'hoaibao@gmail.com','employee');

INSERT INTO HAS\_ACC VALUES ('000000003', 'minhbaoTV1');

INSERT INTO HAS\_ACC VALUES ('000000004', 'ngoctanTV2');

INSERT INTO HAS\_ACC VALUES ('000000005', 'phuoctoanTV5');

INSERT INTO CASHIER

VALUE ('100000003');

INSERT INTO CASHIER

VALUE ('200000003');

INSERT INTO COOKS

VALUE ('100000001',1);

INSERT INTO COOKS

VALUE ('100000004', 2);

INSERT INTO COOKS

VALUE ('200000001', 3);

INSERT INTO COOKS

VALUE ('200000004', 3);

INSERT INTO COOK

VALUE ('100000001', 3, NULL);

INSERT INTO COOK

VALUE ('100000004', 1, '100000001');

INSERT INTO COOK

VALUE ('200000001', 2, null);

INSERT INTO COOK

VALUE ('200000004', 1, '200000001');

INSERT INTO CUSTOMER VALUES ('000000001', 'Le Van A', '0907894562', 0);

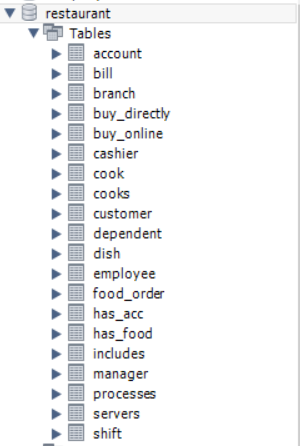
INSERT INTO CUSTOMER VALUES ('000000002', 'Nguyen Van B', '0907894563', 5);

INSERT INTO CUSTOMER VALUES ('000000003', 'Pham Thi C', '0907894564', 0);

INSERT INTO CUSTOMER VALUES ('000000004', 'Tran Duc D', '0907894565', 0);

INSERT INTO CUSTOMER VALUES ('000000005', 'To Hien E', '0907894566', 15);

## Màn hình hiển thị



# PHẦN RIÊNG

## THÀNH VIÊN: NGUYỄN MINH BẢO – MSSV: 1912676

**Câu 1. Thủ tục insert dữ liệu:**

- Mô tả chức năng: Thủ tục insert dữ liệu vào bảng BRANCH, đồng thời kiểm tra

điều kiện, nếu dữ liệu insert vào có “Number” trùng với bất kì record nào

đang có trong bảng hoặc “manage\_ID” không tồn tại thì sẽ thông báo lỗi và insert dữ liệu không thành công.

- Câu lệnh tạo thủ tục:

DROP PROCEDURE IF EXISTS Add\_Branch;

DELIMITER $$

CREATE PROCEDURE Add\_Branch(

    IN NumberIN INT,

    IN NameIN VARCHAR(255),

    IN LocationIN VARCHAR(255),

    IN PhoneIN CHAR(10),

    IN Manager\_IDIN CHAR(9))

proc\_label:BEGIN

    IF (NOT EXISTS (SELECT \* FROM manager WHERE ID = Manager\_IDIN)) THEN

            BEGIN

                    signal sqlstate '45000' set message\_text = 'Không tìm thấy manager\_ID';

            END;

    ELSEIF(EXISTS (SELECT \* FROM branch WHERE Number = NumberIN)) THEN

            BEGIN

                    signal sqlstate '45000' set message\_text = 'Number đã tồn tại';

            END;

    ELSE

            INSERT INTO branch (Number, Name, Location, Phone, Manager\_ID) VALUES (NumberIN, NameIN, LocationIN, PhoneIN, Manager\_IDIN);

    END IF;

END$$

DELIMITER ;

- Câu lệnh thực thi thủ tục mẫu:

CALL Add\_Branch(4, 'THB', '79 Ly Thuong Kiet', '038789729', '100000005');

- Kết quả màn hình hiển thị từ DBMS:



**Câu 2. Trigger:**

***Trigger 1:***

- Mô tả chức năng: Mỗi BRANHCH chỉ có 1 manager. Khi thêm một BRANCH thì trigger dưới đây sẽ kiểm soát vấn đề đó.

- Câu lệnh tạo trigger:

DROP TRIGGER IF EXISTS Check\_Manager;

DELIMITER $$

CREATE TRIGGER Check\_Manager

BEFORE INSERT ON branch

FOR EACH ROW

proc\_label: BEGIN

    IF (NOT EXISTS (SELECT \* FROM manager WHERE ID = new.Manager\_ID)) THEN

            BEGIN

                    signal sqlstate '45000' set message\_text = 'Không tìm thấy manager\_ID';

                    LEAVE proc\_label;

            END;

    ELSEIF(EXISTS (SELECT \* FROM branch WHERE Manager\_ID = new.Manager\_ID)) THEN

        BEGIN

            signal sqlstate '45000' set message\_text = 'Manage đã quản lý nhà hàng. Xin hãy chọn người quản lý khác';

            LEAVE proc\_label;

        END;

    END IF;

END$$

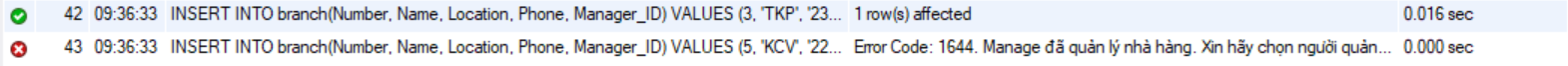
DELIMITER ;

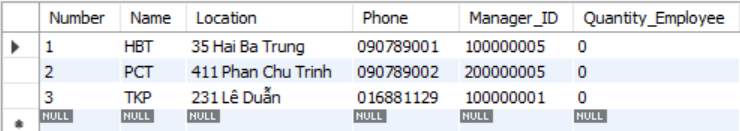
- Câu lệnh kiểm tra trigger hoạt động:

INSERT INTO branch(Number, Name, Location, Phone, Manager\_ID) VALUES (3, 'TKP', '231 Lê Duẫn', '016881129', '100000001');

INSERT INTO branch(Number, Name, Location, Phone, Manager\_ID) VALUES (5, 'KCV', '22 Trường Chinh', '017880293', '100000001');

- Kết quả màn hình hiển thị từ DBMS:





***Trigger 2:***

- Mô tả chức năng: Tự động Update lại Quantity\_Employee(Số lượng nhân viên) vào bảng BRANCH khi thêm một nhân viên ở bảng EMPLOYEE.

- Câu lệnh tạo trigger:

DROP TRIGGER IF EXISTS Quantity\_Emloyee;

DELIMITER $$

CREATE TRIGGER Quantity\_Emloyee

AFTER INSERT ON employee

FOR EACH ROW

proc\_label: BEGIN

    UPDATE branch

    SET Quantity\_Employee = Quantity\_Employee + 1

    WHERE Number = new.Bnumber;

END$$

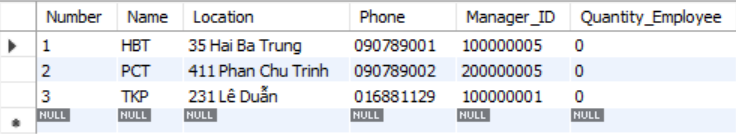
DELIMITER ;

- Câu lệnh kiểm tra trigger hoạt động:

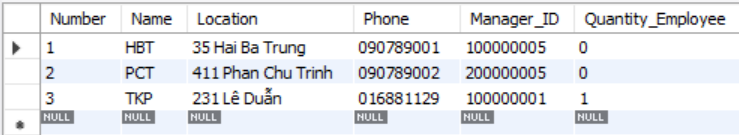
INSERT INTO EMPLOYEE VALUE('200000005','Tran Thi E','abc0@gmail.com','0903334451','2001-01-22','Nu','206 Quan6,TP Ho Chi Minh',6000000,'hoanganTV30',2);

- Kết quả màn hình hiển thị từ DBMS:

*Trước khi insert Employ thông tin bảng Branch như sau:*



*Sau khi insert Employ thông tin bảng Branch như sau:*



**Câu 3. Thủ tục chứa câu SQL:**

***Thủ tục 1:***

- Mô tả chức năng: Hiển thị danh sách các các nhân viên của một nhà hàng với tham số đầu vào là ‘Number’ của nhà hàng đó và sắp xếp tên các nhân viên theo chiều tăng dần

- Câu lệnh tạo thủ tục:

DROP PROCEDURE IF EXISTS Employee\_Of\_Branch;

DELIMITER $$

CREATE PROCEDURE Employee\_Of\_Branch(

    IN Num INT

)

proc\_label:BEGIN

    SELECT ID, employee.Name, Email, employee.Phone, Bdate, Sex, Address, Salary

    FROM employee

    JOIN branch ON employee.Bnumber = branch.Number

    WHERE branch.Number = Num

    ORDER BY Name;

END$$

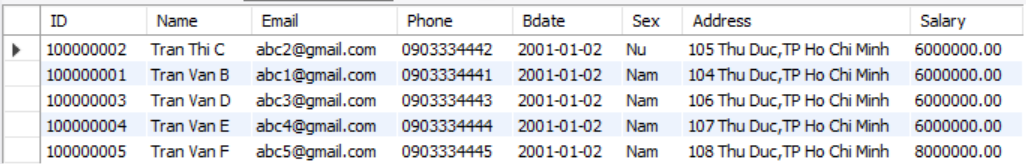
DELIMITER ;

CALL Employee\_Of\_Branch(1);

- Câu lệnh thực thi thủ tục mẫu:

CALL Employee\_Of\_Branch(1);

- Kết quả màn hình hiển thị từ DBMS:



***Thủ tục 2:***

- Mô tả chức năng: Với mỗi Branch cho biết Branch nào có số lượng nhân viên ‘nam’ lớn hơn một số với số đó làm tham số đầu vào và sắp xếp theo tên của các Branch

- Câu lệnh tạo thủ tục:

DROP PROCEDURE IF EXISTS Check\_Quantity\_Employee;

DELIMITER $$

CREATE PROCEDURE Check\_Quantity\_Employee(

    IN Num INT

)

proc\_label:BEGIN

    SELECT branch.Number, branch.Name, branch.Location, branch.Phone, branch.Manager\_ID, branch.Quantity\_Employee

    FROM branch

    JOIN (SELECT Number, COUNT(\*) AS count

        FROM branch

        LEFT JOIN employee ON employee.Bnumber = branch.Number

WHERE employee.Sex = 'Nam'

        GROUP BY Number

        HAVING COUNT(\*) >= Num) AS table\_count

    ON branch.Number = table\_count.Number

    ORDER BY branch.Number;

END$$

DELIMITER ;

- Câu lệnh thực thi thủ tục mẫu:

CALL Check\_Quantity\_Employee(3);

- Kết quả màn hình hiển thị từ DBMS:



**Câu 4. Hàm**

***Function 1:***

- Mô tả chức năng: Hàm nhập vào “Number” của Branch và trả về số lượng món ăn có Branch đó có được

- Câu lệnh tạo function:

DROP FUNCTION IF EXISTS Count\_Dish;

DELIMITER $$

CREATE FUNCTION Count\_Dish(

    Num INT

)

RETURNS INT

DETERMINISTIC

proc\_label:BEGIN

    IF (NOT EXISTS (SELECT \* FROM Branch WHERE Number = Num)) THEN

            BEGIN

                    signal sqlstate '45000' set message\_text = 'Không tìm thấy Number Branch';

            END;

    ELSE

        return(

            SELECT COUNT(\*)

            FROM has\_food

            JOIN dish ON has\_food.Dish\_name = dish.Name WHERE has\_food.Number = Num

        );

    END IF;

END $$

DELIMITER ;

- Câu lệnh thực thi function:

SELECT Count\_Dish(2);

- Kết quả màn hình hiển thị từ DBMS:

****

***Function 2:***

- Mô tả chức năng: Hàm nhập vào “Number” của Branch, tính toán tỉ lệ phần trăm nữ trong Brand đó.

- Câu lệnh tạo function:

DROP FUNCTION IF EXISTS stat;

DELIMITER $$

CREATE FUNCTION stat(

    Num INT

)

RETURNS decimal(10,2)

DETERMINISTIC

proc\_label:BEGIN

    IF (NOT EXISTS (SELECT \* FROM Branch WHERE Number = Num)) THEN

            BEGIN

                    signal sqlstate '45000' set message\_text = 'Không tìm thấy Number Branch';

            END;

    ELSE

        BEGIN

            DECLARE male  decimal(10,2);

            DECLARE total decimal(10,2);

            SET male = (SELECT COUNT(\*)

                        FROM branch

                        JOIN employee ON branch.Number = employee.Bnumber WHERE employee.Sex = 'Nu' AND branch.Number = Num);

            SET total = (SELECT COUNT(\*)

                        FROM branch

                        JOIN employee ON branch.Number = employee.Bnumber WHERE branch.Number = Num);

            return (male / total) \* 100;

        END;

    END IF;

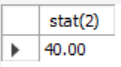
END $$

DELIMITER ;

- Câu lệnh thực thi function:

SELECT stat(2);

- Kết quả màn hình hiển thị từ DBMS:



## Thành viên: Phạm Đại Hoàng An – MSSV: 1912539

**Câu 1: Thủ tục Insert dữ liệu**

Mô tả chức năng: Thêm dữ liệu của Employee vào Database.

Câu lệnh tạo thủ tục:

Delimiter //

CREATE PROCEDURE Add\_Employee(   ID\_new char(9),

                                 Name\_new VARCHAR(255),

                                 Email\_new varchar(255),

                                 Phone\_new char(10),

                                 Bdate\_new date,

                                 sex\_new varchar(255),

                                 address\_new varchar(255),

                                 Salary\_new decimal(10,2),

                                 Eaccount\_new varchar(255),

                                 Bnumber\_new int)

BEGIN

    DECLARE m\_id char(9);

    DECLARE m\_bnumber int;

    SET m\_id = ( SELECT ID

                 FROM EMPLOYEE

                 WHERE ID = ID\_new);

    SET m\_bnumber = (SELECT Number

                    FROM BRANCH

                    WHERE Bnumber\_new = Number);

    if (m\_id is not null) then

        begin

            SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = "ALREADY HAVE ID EMPLOYEE";

        end;

    elseif (m\_bnumber is null) then

        begin

            SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = "DON'T HAVE NUMBER BRANCH";

        end;

    else

        INSERT INTO ACCOUNT Value(Eaccount\_new,"11111111", Email\_new,"employee");

        INSERT INTO Employee Value( ID\_new, Name\_new,Email\_new,Phone\_new, Bdate\_new, sex\_new, address\_new, Salary\_new, Eaccount\_new, Bnumber\_new);

    end if;

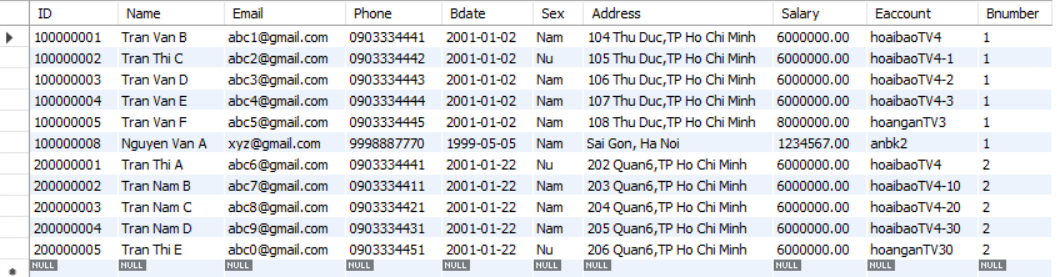
END;

// Delimiter ;

Câu lệnh thực thi thủ tục:

CALL Add\_Employee('100000008','Nguyen Van A','xyz@gmail.com','9998887770','2010-05-05','Nam','Sai Gon, Ha Noi',1234567,'anbk1',1);

Kết quả màn hình hiển thị DBMS:



**Câu 2: Trigger**

Mô tả chức năng: 1 trigger dùng để kiểm tra xem nhân viên có lớn hơn 18 tuổi chưa và 1 trigger dùng để tăng lương cho nhân viên thêm 300.000 mỗi ca làm việc khi nhân viên có nhiều hơn 3 ca làm việc.

Câu lệnh tạo trigger:

DELIMITER $$

CREATE TRIGGER Check\_age\_employee

BEFORE INSERT

ON employee FOR EACH ROW

BEGIN

    IF(YEAR(NOW()) - YEAR(new.Bdate) < 18) THEN

        BEGIN

            signal sqlstate '45000' set message\_text = 'Nhan vien phai lon hon 18 tuoi';

        END;

    END IF;

END$$

DELIMITER ;

Delimiter $$

CREATE TRIGGER Increase\_salary

AFTER INSERT

ON SHIFT FOR EACH ROW

BEGIN

    DECLARE TOTAL INT default 0;

    SET TOTAL =( SELECT COUNT(\*)

                 FROM SHIFT

                 WHERE new.EID = EID );

    IF TOTAL > 3 THEN

        BEGIN

            UPDATE Employee

            SET Salary = 6000000 + (TOTAL-3)\*300000

            WHERE ID = new.EID;

        END;

    END IF;

END$$

Delimiter ;

Câu lệnh kiểm tra trigger hoạt động:

* Kiểm tra nhân viên đã 18 tuổi hay chưa

CALL Add\_Employee('100000009','Nguyen Van A','xyz@gmail.com','9998887770','2010-05-05','Nam','Sai Gon, Ha Noi',1234567,'anbk3',1);

* Tăng lương cho nhân viên khi có trên 3 ca làm việc:

INSERT INTO SHIFT value('200000005','T7','Chieu','12');

Kết quả màn hình hiển thị từ DBMS:

* Báo lỗi khi nhân viên chưa đủ 18 tuổi:



* Nhân viên với mã ID 200000005 tăng lương từ 6 000 000 => 6 300 000 khi có 4 ca làm việc:



**Câu 3: Thủ tục chứa câu SQL**

Mô tả chức năng: Hiển thị tên các nhân viên làm việc trong một chi nhánh với tham số đầu vào là tên chi nhánh và hiển thị các nhân viên có số ca làm việc nhiều hơn một con số nhất định với tham số đầu vào là một con số.

Câu lệnh tạo thủ tục:

*-- a)*

Delimiter //

CREATE PROCEDURE EoB(Bname VARCHAR(255))

BEGIN

    DECLARE b\_name VARCHAR(255);

    SET b\_name = (  SELECT Name

                    FROM Branch

                    WHERE Bname = Name );

    if (b\_name is null) then

        signal sqlstate '45000' set message\_text = 'Khong ton tai chi nhanh';

    else

        SELECT  employee.ID, employee.Name , employee.Email, employee.Phone,employee.Bdate,employee.Sex, employee.Address, employee.Salary, employee.Eaccount, employee.Bnumber

        FROM branch, employee

        WHERE branch.Number = employee.Bnumber AND branch.Name = Bname

        ORDER BY employee.ID;

    end if;

END;

// Delimiter ;

*-- b)*

Delimiter //

CREATE PROCEDURE Number\_of\_shift ( Num INT )

BEGIN

    SELECT employee.ID, employee.Name , employee.Email, employee.Phone,employee.Bdate,employee.Sex, employee.Address, employee.Salary, employee.Eaccount, employee.Bnumber

    FROM branch,employee, ( SELECT EID , COUNT(\*) As NumShift

                            FROM SHIFT

                            GROUP BY EID

                            HAVING NumShift >= Num) As shiftv2

    WHERE branch.Number= employee.Bnumber AND employee.ID = shiftv2.EID

    ORDER BY branch.Number;

END;

// Delimiter ;

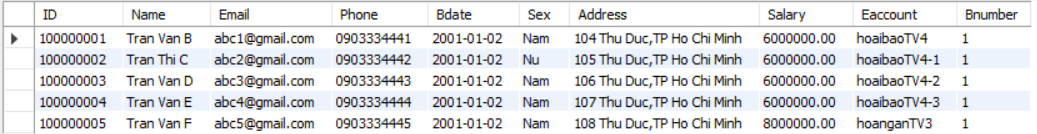
Câu lệnh thực thi thủ tục:

CALL EoB("HBT");

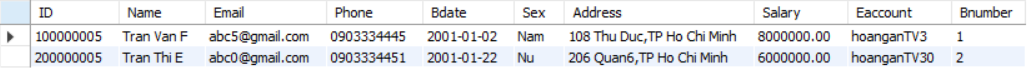
CALL Number\_of\_shift(3);

Kết quả màn hình hiển thị từ DBMS:

* Nhân viên làm việc tại chi nhánh HBT:



* Nhân viên có số ca làm việc không dưới 3:



**Câu 4: Hàm**

Mô tả chức năng: Hàm tính quỹ lương trả cho nhân viên của 1 chi nhánh với tham số đầu vào là mã chi nhánh và hàm trả về việc làm của mỗi nhân viên.

Câu lệnh tạo hàm:

*-- Tinh quy luong cua moi Chi Nhanh*

Delimiter //

CREATE FUNCTION Branch\_salary( Bnum INT)

RETURNS DECIMAL(10,2)

BEGIN

    DECLARE bid INT;

    DECLARE total DECIMAL(10,2);

    SET bid = ( SELECT Number

                FROM branch

                WHERE Bnum = Number);

    if bid is null then

        begin

            SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = "Chi Nhanh khong ton tai";

        end;

    end if;

    SET total = (SELECT SUM(Salary) As Quy\_Luong

                 FROM Employee

                 WHERE Bnumber = Bnum);

    RETURN total;

END;

// Delimiter ;

*-- Cong viec cua moi thanh vien*

Delimiter //

CREATE FUNCTION employee\_job (emp\_ID CHAR(9))

RETURNS VARCHAR(32)

BEGIN

    DECLARE job\_name VARCHAR(32) default "";

    DECLARE f\_manager CHAR(9);

    DECLARE f\_servers CHAR(9);

    DECLARE f\_cook CHAR(9);

    DECLARE f\_cashier CHAR(9);

    SET f\_manager = ( SELECT ID

                      FROM MANAGER

                      WHERE ID = emp\_ID);

    SET f\_servers = ( SELECT ID

                      FROM SERVERS

                      WHERE ID = emp\_ID);

    SET f\_cook    = ( SELECT ID

                      FROM COOK

                      WHERE ID = emp\_ID);

    SET f\_cashier = ( SELECT ID

                      FROM CASHIER

                      WHERE ID = emp\_ID);

    if f\_manager is not null then

        SET job\_name = "MANAGER";

    elseif f\_servers is not null then

        SET job\_name = "SERVERS";

    elseif f\_cook is not null then

        SET job\_name = "COOK";

    elseif f\_cashier is not null then

        SET job\_name = "CASHIER";

    end if;

    RETURN job\_name;

END;

// Delimiter ;

Câu lệnh SELECT minh họa hàm:

SELECT Number, Branch\_salary(`number`) as Quy\_Luong, Name

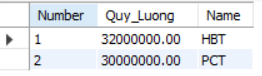
FROM BRANCH;

SELECT ID,Name, Salary,Sex, employee\_job(`ID`) As Job

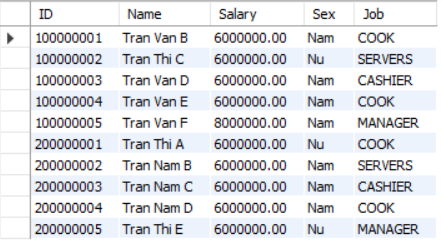
FROM EMPLOYEE;

Kết quả màn hình hiển thị từ DBMS:

* Quỹ lương của 2 chi nhánh:



* Việc làm của mỗi nhân viên:



**Câu 5: Giao diện ứng dụng và các hình ảnh minh họa**

## Thành viên: Phạm Ngọc Tân – MSSV: 1915071

***Câu 1: Procedure dùng để insert dữ liệu***

- Mô tả chức năng: Procedure với 3 tham số đầu vào, có chức năng thêm 1 record mới vào bảng DISH (có kiểm tra giá trị các tham số đầu vào) với giá trị các thuộc tính tương ứng là new\_Name, new\_Price và new\_Image\_Link.

- Câu lệnh tạo Procedure:

DELIMITER $$

CREATE PROCEDURE insert\_DISH(

    new\_Name VARCHAR(255),

    new\_Price DECIMAL(10,2),

    new\_Image\_Link TEXT)

BEGIN

    DECLARE C INT;

    SELECT COUNT(Name) INTO C

    FROM DISH

    WHERE Name = new\_Name;

    IF C != 0 THEN

        SIGNAL SQLSTATE '45000'

        SET MESSAGE\_TEXT = "This Name of DISH already exists. Please try again!";

    ELSEIF new\_Price < 20000 THEN

        SIGNAL SQLSTATE '45000'

        SET MESSAGE\_TEXT = "Price must not be less then 20000. Please try again!";

    ELSEIF MOD(new\_Price, 1000) != 0 THEN

        SIGNAL SQLSTATE '45000'

        SET MESSAGE\_TEXT = "Price must be divisible by 1000. Please try again!";

    ELSE

        INSERT INTO DISH VALUES (new\_Name, new\_Price, new\_Image\_Link);

    END IF;

END; $$

DELIMITER ;

- Câu lệnh thực thi mẫu:

CALL insert\_DISH("Hủ Tiếu Nam Vang", 30000, "");

CALL insert\_DISH("Bánh Bao Trứng Cút", 15000, "");

CALL insert\_DISH("Bánh Bao Trứng Cút", 20001, "");

CALL insert\_DISH("Bánh Bao Trứng Cút", 25000, "");

- Kết quả màn hình hiển thị từ DBMS:

https://lh4.googleusercontent.com/sZ_s941GoxV2CbbMwd-SGWiRuiBaK5zwMiWBzELlZYsIQSnRjYAH6Z4vAxyYI-N9vrmrnSMupsL_-qwRGEW-7CnGaPcNAwe9DmZwPQ4EcM3s4OnaTYKhVSvKYujscZBS9IzTwtm8

***Câu 2: Trigger***

* *Trigger a:*

- Mô tả chức năng: Trigger dùng để kiểm tra ràng buộc số đơn vị món ăn có sẵn tại 1 chi nhánh trong 1 ngày không vượt quá 100 khi thêm món đó vào chi nhánh.

- Câu lệnh tạo Trigger:

DELIMITER $$

CREATE TRIGGER before\_insert\_HAS\_FOOD

BEFORE INSERT

ON HAS\_FOOD

FOR EACH ROW

BEGIN

    IF NEW.Available\_Quantity > 100 THEN

        BEGIN

            SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = "Available quantity can not exceed 100!";

        END;

    END IF;

END; $$

DELIMITER ;

- Câu lệnh kiểm tra Trigger hoạt động:

INSERT INTO HAS\_FOOD VALUES ("Cơm Chiên Dương Châu", 1, 120);

INSERT INTO HAS\_FOOD VALUES ("Cơm Chiên Dương Châu", 1, 60);

- Kết quả màn hình hiển thị từ DBMS:

https://lh4.googleusercontent.com/7SEY11iw4uwo4LY2V8gzBvSXV5jmNnXHuzCu55x-dxBG4PHmoRvQqCa91Y2HRpE1jsQOjPZ4q4RtquqdDKhGw6aJxbsgW9_AnszxF7utsbkCh-CVsqWiKhSjn-NJQJxctxnNMBnh

* *Trigger b:*

- Mô tả chức năng: Trigger dùng để giảm giá 1 món ăn xuống 3000đ nếu món ăn đó có ở nhiều hơn 3 chi nhánh, được kích hoạt sau khi thêm món ăn đó vào 1 chi nhánh.

- Câu lệnh tạo Trigger:

DELIMITER $$

CREATE TRIGGER after\_insert\_HAS\_FOOD

AFTER INSERT

ON HAS\_FOOD

FOR EACH ROW

BEGIN

    IF (SELECT COUNT(Number) FROM HAS\_FOOD WHERE Dish\_Name = NEW.Dish\_Name) > 3 THEN

        BEGIN

            UPDATE DISH

            SET Price = Price - 3000

            WHERE Name = NEW.Dish\_Name;

        END;

    END IF;

END; $$

DELIMITER ;

- Câu lệnh kiểm tra trigger hoạt động:

* Thực thi câu lệnh SELECT sau ta được *Hình 1* như bên dưới.

SELECT D.Name, D.Price, H.Number

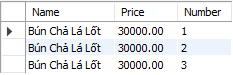
FROM DISH D, HAS\_FOOD H

WHERE D.Name = H.Dish\_Name AND Dish\_Name = "Bún Chả Lá Lốt";

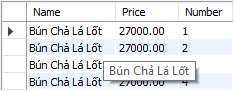
* Thực thi câu lệnh INSERT sau rồi tiếp tục với câu lệnh SELECT trên ta được *Hình 2* như bên dưới.

INSERT INTO HAS\_FOOD VALUES ("Bún Chả Lá Lốt", 4, 30);

- Kết quả màn hình hiển thị từ DBMS:

**

*Hình 1*

**

*Hình 2*

***Câu 3: Procedure chứa câu truy vấn SQL***

* *Procedure a:*

- Mô tả chức năng: Procedure với 2 tham số đầu vào là Compared\_Price và Branch\_Num, có chức năng liệt kê danh sách tất cả các món ăn có giá không vượt quá Compared\_Price tại chi nhánh có mã số Branch\_Num.

- Câu lệnh tạo Procedure:

DELIMITER $$

CREATE PROCEDURE Dish\_Price\_Filter(

    Compared\_Price DECIMAL(10,2),

    Branch\_Num INT)

BEGIN

    IF Branch\_num NOT IN (SELECT Number FROM BRANCH) THEN

        SIGNAL SQLSTATE '45000'

        SET MESSAGE\_TEXT = "This Number of BRANCH does not exist. Please try again!";

    ELSE

        SELECT H.Number AS Branch\_Num, D.Image\_Link AS Image\_Link, H.Dish\_name AS Dish\_Name, D.Price AS Price

        FROM DISH AS D, HAS\_FOOD AS H

        WHERE H.Dish\_Name = D.Name AND D.Price <= Compared\_Price AND H.number = Branch\_Num

        ORDER BY Price;

    END IF;

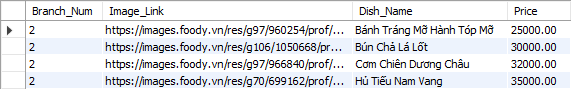
END; $$

DELIMITER ;

- Câu lệnh thực thi Procedure mẫu:

CALL Dish\_Price\_Filter(35000, 2);

- Kết quả màn hình hiển thị từ DBMS:



* *Procedure b:*

- Mô tả chức năng: Procedure với 1 tham số đầu vào là Total\_Qty, có chức năng liệt kê danh sách tất cả các món ăn có tại nhiều hơn 1 chi nhánh, có tổng số lượng đơn vị món có sẵn tại tất cả chi nhánh đó tối thiểu bằng Total\_Qty.

- Câu lệnh tạo Procedure:

DELIMITER $$

CREATE PROCEDURE Available\_Dish(

    Total\_Qty INT)

BEGIN

        SELECT D.Image\_Link AS Image\_Link, H.Dish\_Name AS Dish\_Name, D.Price AS Price,

               COUNT(H.Number) AS Num\_of\_Branches, SUM(H.Available\_Quantity) AS Total\_Quantity

        FROM DISH AS D, HAS\_FOOD AS H

        WHERE H.Dish\_Name = D.Name

        GROUP BY H.Dish\_Name

        HAVING COUNT(H.Number) > 1 AND SUM(H.Available\_Quantity) >= Total\_Qty

        ORDER BY H.Dish\_Name;

END; $$

DELIMITER ;

- Câu lệnh thực thi Procedure mẫu:

CALL Available\_Dish(20);

- Kết quả màn hình hiển thị từ DBMS:



***Câu 4: Function***

* *Function a:*

- Mô tả chức năng: Function với 1 tham số đầu vào là Price có chức năng phân loại giá cho món ăn, cụ thể là xác định món ăn đó có giá thấp (dưới 30000đ) hay giá trung bình (từ 30000đ đến 50000đ) hay giá cao (trên 50000đ).

- Câu lệnh tạo Function:

DELIMITER $$

CREATE FUNCTION Price\_Level(Price DECIMAL(10,2))

RETURNS VARCHAR(12)

DETERMINISTIC

BEGIN

    DECLARE Price\_Level VARCHAR(12);

    IF Price < 30000 THEN

        SET Price\_Level = 'Thấp';

    ELSEIF (Price >= 30000 AND Price <= 50000) THEN

        SET Price\_Level = 'Trung bình';

    ELSEIF Price > 50000 THEN

        SET Price\_Level = 'Cao';

    END IF;

    RETURN (Price\_Level);

END $$

DELIMITER ;

- Câu lệnh thực thi Function:

 SELECT Name, Price, Price\_Level(Price) AS Price\_Level

FROM DISH

ORDER BY Price;

- Kết quả màn hình hiển thị từ DBMS:



* *Function b:*

- Mô tả chức năng: Function với 1 tham số đầu vào là Dish, có chức năng xác định xem 1 món ăn có phải là món “đặc biệt” hay không với điều kiện món ăn “ đặc biệt” là món chỉ có ở 1 chi nhánh.

- Câu lệnh tạo Function:

DELIMITER $$

CREATE FUNCTION Unique\_Dish(Dish VARCHAR(255))

RETURNS VARCHAR(8)

DETERMINISTIC

BEGIN

    DECLARE is\_Unique VARCHAR(8) DEFAULT '';

    DECLARE Num\_of\_Branches INT DEFAULT 0;

    SELECT COUNT(Dish\_Name) INTO Num\_of\_Branches

    FROM HAS\_FOOD

    WHERE Dish\_Name = Dish;

    IF Num\_of\_Branches = 1 THEN

        SET is\_Unique = 'Đặc biệt';

    END IF;

    RETURN (is\_Unique);

END $$

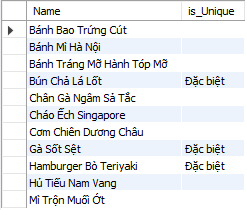
DELIMITER ;

- Câu lệnh thực thi Function:

SELECT Name, Unique\_Dish(Name) AS is\_Unique

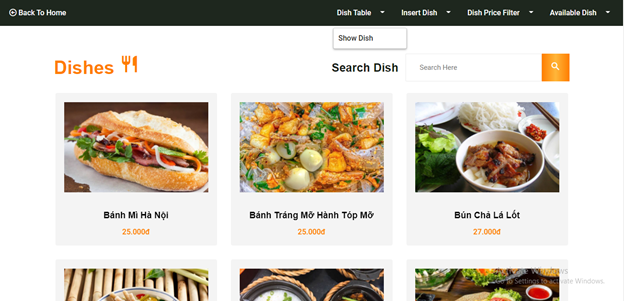
FROM DISH;

- Kết quả màn hình hiển thị từ DBMS:

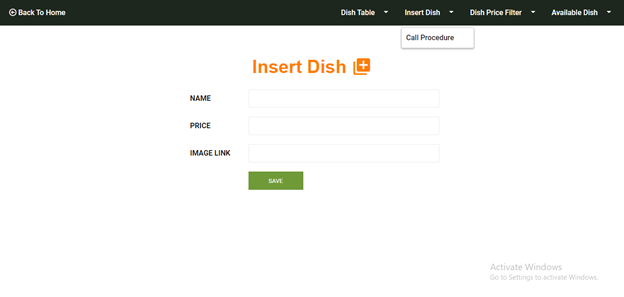


***Câu 5: Giao diện ứng dụng***

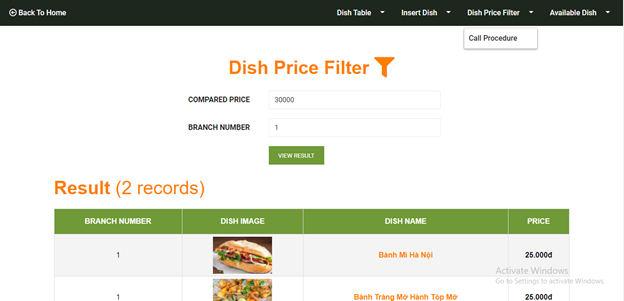
- Giao diện bao gồm 4 tùy chọn chính: Dish Table, Insert Dish, Dish Price Filter và Available Dish.

- Chọn Show Dish trong tùy chọn Dish Table để hiển thị toàn bộ các món ăn của nhà hàng dưới dạng các card. Ở trang này chúng ta có thể tìm kiếm các món ăn, chỉnh sửa hoặc xóa các món ăn.

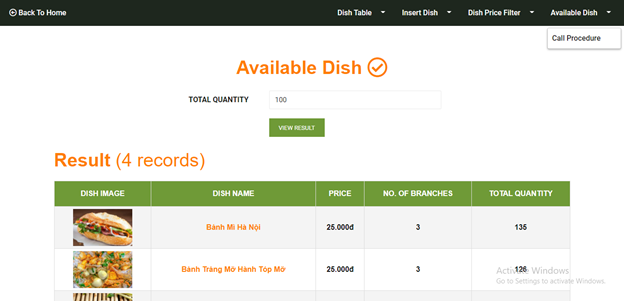
- Chọn Call Procedure trong tùy chọn Insert Dish để gọi Procedure insert\_Dish() hiển thị form thực hiện chức năng insert 1 record vào bảng DISH.



- Chọn Call Procedure trong tùy chọn Dish Price Filter để gọi Procedure Dish\_Price\_Filter() trong câu 3a:



- Chọn Call Procedure trong tùy chọn Available Dish để gọi Procedure Available\_Dish() trong câu 3b:



## Thành viên: Nguyễn Phước Toàn – MSSV: 1915551

## Thành viên: Lê Hồ Hoài Bão – MSSV: 1912698

**Câu 1: Thủ tục Insert dữ liệu**

Mô tả chức năng: Thêm dữ liệu của Includes (Các món ăn trong đơn hàng) vào Database

Câu lệnh tạo thủ tục:

DELIMITER $$

CREATE PROCEDURE cau1 (IN num INT, IN dish\_name VARCHAR(255), IN quantity INT)

BEGIN

DECLARE p DECIMAL(10,2);

DECLARE d VARCHAR(255);

SET p = (SELECT Price FROM dish D WHERE dish\_name = D.Name);

SET d = (SELECT Dname FROM includes I WHERE I.Onum = num AND I.Dname = dish\_name);

IF (num < 1) THEN

BEGIN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = "INVALID NUMBER";

END;

ELSEIF (quantity < 1) THEN

BEGIN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = "INVALID QUANTITY";

END;

ELSEIF (p is null) THEN

BEGIN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = "DISH IS NOT EXIST";

END;

ELSEIF (d is not null) THEN

UPDATE includes SET Ord\_quantity = Ord\_quantity + quantity, Listed\_price = Listed\_price + quantity \* p WHERE Onum = num AND Dname = dish\_name;

ELSE

INSERT INTO includes VALUES (num, dish\_name, quantity, p \* quantity);

END IF;

END $$

DELIMITER ;

Câu lệnh thực thi thủ tục:

CALL cau1(6,"Bánh Mì Hà Nội",3)

Kết quả màn hình hiển thị DBMS:



**Câu 2: Trigger**

Mô tả chức năng:

- Trigger đầu dùng để kiểm tra xem món ăn mới thêm vào đã nằm trong order có sẵn chưa hay là từ một order mới.

- Trigger sau dùng để thêm hóa đơn (Bill) khi đơn hàng được cập nhật trạng thái đã hoàn thành.

Câu lệnh tạo trigger:

DELIMITER $$

CREATE TRIGGER cau2\_1

BEFORE INSERT ON includes

FOR EACH ROW

BEGIN

DECLARE x INT;

SET x = (SELECT Number FROM food\_order F WHERE F.Number = NEW.Onum);

IF (x is null) THEN

INSERT INTO food\_order VALUES (NEW.Onum, "Chưa xử lý");

END IF;

END $$

DELIMITER ;

DELIMITER $$

CREATE TRIGGER cau2\_2

AFTER UPDATE ON food\_order

FOR EACH ROW

BEGIN

DECLARE x INT;

DECLARE cus VARCHAR(255);

SET cus = (SELECT C.Name FROM customer C, buy\_directly D WHERE D.ONumber = NEW.Number AND D.CustomerID = C.ID);

IF (cus is null) THEN

SET cus = (SELECT C.Name

FROM customer C, has\_acc HA, buy\_online O

WHERE O.ONumber = NEW.Number AND O.Username = HA.Username AND HA.Customer\_ID = C.ID);

END IF;

SET x = (SELECT MAX(B.Number) FROM bill B);

IF (NEW.Status = "Đã hoàn thành") THEN

INSERT INTO bill VALUES (x+1,NEW.Number,cus,0);

End IF;

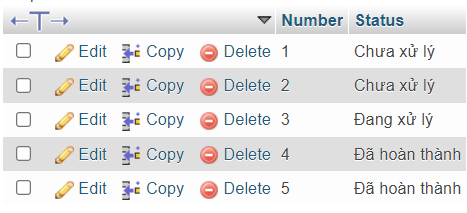
END $$

DELIMITER ;

Câu lệnh kiểm tra trigger hoạt động:

* Thêm 1 món ăn vào bảng include nhưng chưa có trong bảng food\_order

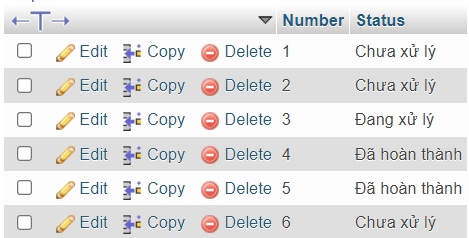
+ Bảng food\_order trước khi nhập lệnh:



+ Câu lệnh nhập:

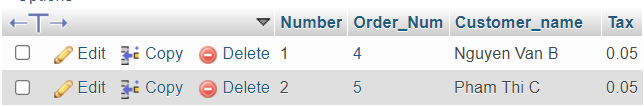
INSERT INTO includes VALUES (6,"Bánh Mì Hà Nội",1,25000)

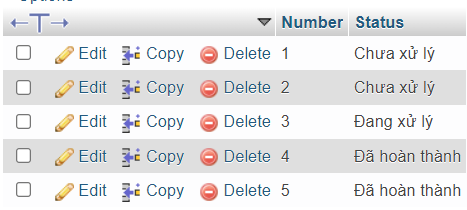
+ Bảng food\_order sau khi nhập:



* Cập nhật 1 đơn hàng thành trạng thái “Đã hoàn thành”:

+ Bảng bill và food\_order trước khi nhập lệnh:

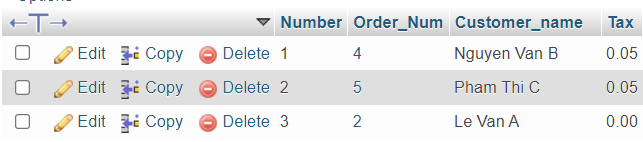


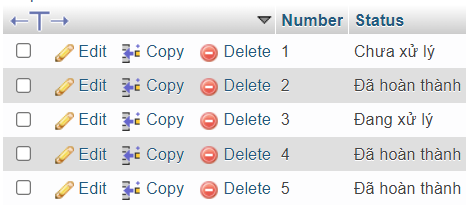


+ Câu lệnh nhập:

UPDATE food\_order F SET F.Status = "Đã hoàn thành" WHERE F.Number = 2;

+ Bảng bill và food\_order sau khi nhập:





**Câu 3: Thủ tục chứa câu SQL**

Mô tả chức năng: Thủ tục 1 hiển thị những đơn hàng và tình trạng của chúng với mỗi món ăn nhập vào, thủ tục 2 xuất ra những món ăn có số lượng đặt hàng online có số lượng lớn hơn số nhập vào.

Câu lệnh tạo thủ tục:

DELIMITER $$

CREATE PROCEDURE cau3\_1 (IN dish\_name VARCHAR(255))

BEGIN

SELECT F.Number, F.Status

FROM food\_order F, includes I

WHERE I.Dname = dish\_name AND F.Number = I.Onum

ORDER BY F.Status;

END $$

DELIMITER ;

DELIMITER $$

CREATE PROCEDURE cau3\_2 (IN numb INT)

BEGIN

SELECT I.Dname, COUNT(\*)

FROM includes I, buy\_online O

WHERE I.Onum = O.ONumber

GROUP BY I.Dname

HAVING COUNT(\*) > numb;

END $$

DELIMITER ;

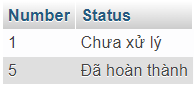
Câu lệnh thực thi thủ tục:

CALL cau3\_1("Bánh Mì Hà Nội")

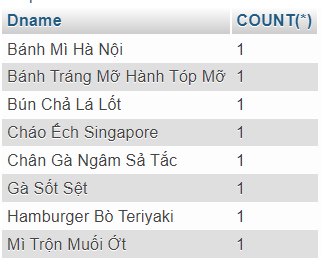
CALL cau3\_2(0)

Kết quả màn hình hiển thị từ DBMS:

* Những đơn hàng có món “Bánh Mì Hà Nội”:



* Những món ăn có số lượng đặt online lớn hơn 0:



**Câu 4: Hàm**

Mô tả chức năng: Hàm đầu tiên kiểm tra tổng tiền thu được thông qua hình thức bán online hoặc trực tiếp, nếu đầu vào không đúng sẽ báo lỗi. Hàm sau xuất ra tổng tiền thu của món ăn nhập vào nếu tổng tiền đó lớn hơn 500000

Câu lệnh tạo hàm:

DELIMITER $$

CREATE FUNCTION cau4\_1 (Onl\_or\_Direct VARCHAR(10)) RETURNS DECIMAL(10,2)

DETERMINISTIC

BEGIN

DECLARE summ DECIMAL(10,2);

IF (Onl\_or\_Direct = "Online") THEN

BEGIN

SET summ = (SELECT SUM(I.Listed\_price) FROM includes I, buy\_online O WHERE O.ONumber = I.Onum);

RETURN summ;

END;

ELSEIF (Onl\_or\_Direct = "Direct") THEN

BEGIN

SET summ = (SELECT SUM(I.Listed\_price) FROM includes I, buy\_directly D WHERE D.ONumber = I.Onum);

RETURN summ;

END;

ELSE

BEGIN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = "WRONG METHOD";

END;

END IF;

END $$

DELIMITER ;

DELIMITER $$

CREATE FUNCTION cau4\_2 (dishname VARCHAR(255)) RETURNS DECIMAL(10,2)

DETERMINISTIC

BEGIN

DECLARE nnum INT;

DECLARE pri DECIMAL(10,2);

SET nnum = (SELECT COUNT(\*) FROM includes I WHERE I.Dname = dishname);

IF (nnum = 0) THEN

BEGIN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = "NO DISH BOOKING";

END;

ELSE

BEGIN

SET pri = (SELECT SUM(I.Listed\_price) FROM includes I WHERE I.Dname = dishname);

IF (pri > 500000) THEN RETURN pri;

ELSE

BEGIN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = "NOT PASS KPI";

END;

END IF;

END;

END IF;

END $$

DELIMITER ;

Câu lệnh SELECT minh họa hàm:

SELECT cau4\_1("Online")

SELECT cau4\_2("Bánh mì hà nội")

Kết quả màn hình hiển thị từ DBMS tổng tiền thu về từ hình thức bán hàng online:



Kết quả màn hình hiển thị từ DBMS khi kiểm tra doanh thu món “Bánh mì hà nội”



Tức là món ăn này không đủ doanh thu trên 500000.

**Câu 5: Giao diện ứng dụng và các hình ảnh minh họa**

# PHỤ LỤC