Tables Creation

```
create table Restaurants (
reg_num number(10),
name varchar2(20),
branch varchar2(25),
contact_num number(11),
email varchar2(25),
manager_id number(10))
```

```
create table Managers (
manager_id number(10),
name varchar2(18),
address varchar2(25),
email varchar2(16),
salary number(10))
```

```
create table Manager_phone(
manager_id number(10),
phone_num number(15))
```

```
create table Employees (
employee_id number(10),
name varchar2(18),
address varchar2(25),
hire_date date,
job varchar2(10),
salary number(10),
manager_id number(10))
```

```
create table Employee_phone(
employee_id number(10),
phone_num number(15))
```

```
create table Items(
item_no number(5),
description varchar2(40),
price number(6))
```

```
create table Customers(
customer_id number(14),
name varchar2(20),
phone_num varchar2(15))
```

```
create table Orders(
order_id number(14),
amount float(9),
status varchar2(12),
customer_id number(14),
employee_id number(14))
```

```
create table Ordered_Item(
orderItem_id number(14),
order_id number(14),
item_no number(14),
quantity number(5))
```

```
create table Bills(
bill_num number(14),
total_amount float(10),
bill_date date,
order_id number(14),
customer_id number(14),
manager_id number(10))
```

```
create table user_login(
serial number(7),
username varchar2(20),
password varchar2(60),
role varchar2(12),
worker_id number(14))
```

```
create table log_info(
log_id number(10),
operation varchar2(20),
table_name varchar2(20),
date_time timestamp)
```

```
alter table Restaurants add constraint c1 primary
key(reg_num);
alter table Managers add constraint c2 primary
key(manager_id);
alter table Manager_phone add constraint c3
primary key(manager_id,phone_num);
alter table Employees add constraint c4 primary
key(employee_id);
alter table Employee_phone add constraint c5
primary key(employee_id,phone_num);
alter table Items add constraint c6 primary
key(item_no);
alter table Customers add constraint c7 primary
key(customer_id);
alter table Orders add constraint c8 primary
key(order_id);
alter table Ordered_Item add constraint c9 primary
key(orderItem_id);
alter table Bills add constraint c10 primary
key(bill_num);
alter table LOG_INFO add constraint c12 primary
key(log_id);
```

```
alter table Restaurants add constraint fk1 foreign
key(manager_id) references Managers(manager_id);
alter table Employees add constraint fk2 foreign
key(manager_id) references Managers(manager_id);
alter table Orders add constraint fk3 foreign
key(customer_id) references
Customers(customer_id);
alter table Orders add constraint fk4 foreign
key(employee_id) references
Employees(employee_id);
alter table Ordered_Item add constraint fk5 foreign
key(order_id) references Orders(order_id);
alter table Ordered_Item add constraint fk6 foreign
key(item_no) references Items(item_no);
alter table Bills add constraint fk7 foreign
key(customer_id) references
Customers(customer_id);
alter table Bills add constraint fk8 foreign
key(manager_id) references Managers(manager_id);
```

```
create sequence user_login_sq
start with 1
```

```
increment by 1
maxvalue 9999
nocycle
nocache
```

```
create sequence customer_sq
start with 11000
increment by 1
maxvalue 99999
nocycle
nocache
```

```
create sequence order_sq
start with 11000
increment by 1
maxvalue 99999
nocycle
nocache
```

```
create sequence ordered_item_sq
start with 21000
increment by 1
maxvalue 99999
nocycle
nocache
```

```
create sequence bill_sq
start with 31000
increment by 1
maxvalue 99999
nocycle
nocache
```

```
create sequence log_sq
start with 1
increment by 1
maxvalue 99999
nocycle
nocache
```

```
CREATE OR REPLACE FUNCTION CHECK_UNIQUE_ITEM(I_NO ITEMS.ITEM_NO%TYPE)
RETURN VARCHAR2
IS
C NUMBER(2);
BEGIN
SELECT COUNT(*) INTO C FROM ITEMS WHERE ITEM_NO=I_NO;
IF (C=0) THEN
RETURN 'TRUE';
ELSE
RETURN 'FALSE';
END IF;
END;
```

```
CREATE OR REPLACE FUNCTION CHECK_UNIQUE_RES(R_NO RESTAURANTS.REG_NUM%TYPE)
RETURN VARCHAR2
IS
C NUMBER(2);
BEGIN
SELECT COUNT(*) INTO C FROM RESTAURANTS WHERE REG_NUM=R_NO;
IF (C=0) THEN
RETURN 'TRUE';
ELSE
RETURN 'FALSE';
END IF;
END;
```

```
CREATE OR REPLACE FUNCTION CHECK_EXIST_CUSTOMER(MOB CUSTOMERS.PHONE_NUM%TYPE)
RETURN VARCHAR2
IS
C NUMBER(2);
BEGIN
SELECT COUNT(*) INTO C FROM CUSTOMERS WHERE PHONE_NUM=MOB;
IF (C=0) THEN
RETURN 'FALSE';
ELSE
RETURN 'TRUE';
END IF;
END;
```

```
CREATE OR REPLACE FUNCTION ITEM_EXIST(I_NO ITEMS.ITEM_NO%TYPE)
RETURN VARCHAR2
IS
C NUMBER(2);
BEGIN
SELECT COUNT(*) INTO C FROM ITEMS WHERE ITEM_NO=I_NO;
IF (C=1) THEN
RETURN 'TRUE';
ELSE
RETURN 'FALSE';
END IF;
END;
```

```
CREATE OR REPLACE FUNCTION ITEM_TO_ORDER_EXIST(I_NO ITEMS.ITEM_NO%TYPE,O_ID
ORDERS.ORDER_ID%TYPE )
RETURN VARCHAR2
IS
C NUMBER(2);
BEGIN
SELECT COUNT(*) INTO C FROM ORDERED_ITEM WHERE ORDER_ID=O_ID AND ITEM_NO=I_NO;
```

```
IF (C=1) THEN
RETURN 'TRUE';
ELSE
RETURN 'FALSE';
END IF;
END;
```

```
CREATE OR REPLACE FUNCTION CHECK_UNIQUE_EMP(E_ID EMPLOYEES.EMPLOYEE_ID%TYPE)
RETURN VARCHAR2
IS
C NUMBER(2);
BEGIN
SELECT COUNT(*) INTO C FROM EMPLOYEES WHERE EMPLOYEE_ID=E_ID;
IF (C=0) THEN
RETURN 'TRUE';
ELSE
RETURN 'FALSE';
END IF;
END;
```

Views

```
CREATE VIEW ORDER_TO_DELIVER AS SELECT ORDER_ID, NAME AS
CUSTOMER_NAME, AMOUNT, PHONE_NUM FROM ORDERS,CUSTOMERS WHERE
ORDERS.CUSTOMER_ID=CUSTOMERS.CUSTOMER_ID AND STATUS='Waiting del'
```

```
CREATE VIEW EMPLOYEE_ALL_INFO AS SELECT E.*, M.NAME AS
MGR,R.BRANCH FROM EMPLOYEES E, MANAGERS M, RESTAURANTS R WHERE
E.MANAGER_ID=M.MANAGER_ID AND M.MANAGER_ID=R.MANAGER_ID
```

```
CREATE VIEW ORDERS_NOT_SERVED AS SELECT O.ORDER_ID, OI.ITEM_NO, I.DESCRIPTION, O.STATUS FROM
ORDERS O, ITEMS I, ORDERED_ITEM OI WHERE I.ITEM_NO=OI.ITEM_NO AND O.ORDER_ID=OI.ORDER_ID AND
(STATUS='Pending' OR STATUS='Cooking')
```

```
CREATE VIEW SHOW_CART AS SELECT OI.ORDER_ID,OI.ITEM_NO, OI.QUANTITY,(I.PRICE*OI.QUANTITY) AS
ITEMTOTAL FROM ORDERED_ITEM OI,ITEMS I WHERE OI.ITEM_NO=I.ITEM_NO
```

Package & Procedures

```
CREATE PACKAGE RES_MODEL AS
PROCEDURE ADD_RES(R_NO RESTAURANTS.REG_NUM%TYPE,NAME RESTAURANTS.NAME%TYPE,BR
RESTAURANTS.BRANCH%TYPE,CONTACT RESTAURANTS.CONTACT_NUM%TYPE, EMAIL
RESTAURANTS.EMAIL%TYPE,MID RESTAURANTS.MANAGER_ID%TYPE, STAT OUT VARCHAR2);

PROCEDURE EDIT_RES(REG RESTAURANTS.REG_NUM%TYPE,NM RESTAURANTS.NAME%TYPE,BR
RESTAURANTS.BRANCH%TYPE,
CONTACT RESTAURANTS.CONTACT_NUM%TYPE,EM RESTAURANTS.EMAIL%TYPE, M_ID
RESTAURANTS.MANAGER_ID%TYPE,PREV_REG RESTAURANTS.REG_NUM%TYPE);

PROCEDURE DELETE_RES(REG RESTAURANTS.REG_NUM%TYPE, STAT OUT VARCHAR2);
END RES_MODEL;
```

```
CREATE OR REPLACE PACKAGE BODY RES_MODEL AS
PROCEDURE ADD_RES(R_NO RESTAURANTS.REG_NUM%TYPE,NAME RESTAURANTS.NAME%TYPE,BR
RESTAURANTS.BRANCH%TYPE,CONTACT RESTAURANTS.CONTACT_NUM%TYPE, EMAIL
RESTAURANTS.EMAIL%TYPE,MID RESTAURANTS.MANAGER_ID%TYPE, STAT OUT VARCHAR2)
IS
BEGIN
IF (CHECK_UNIQUE_RES(R_NO) = 'TRUE') THEN
INSERT INTO RESTAURANTS VALUES(R_NO,NAME,BR,CONTACT,EMAIL,MID);
STAT := 'Restaurant added successfully !';
ELSE
STAT := 'Reg no is not unique !';
END IF;
END ADD_RES;

PROCEDURE EDIT_RES(REG RESTAURANTS.REG_NUM%TYPE,NM RESTAURANTS.NAME%TYPE,BR
RESTAURANTS.BRANCH%TYPE,
CONTACT RESTAURANTS.CONTACT_NUM%TYPE,EM RESTAURANTS.EMAIL%TYPE, M_ID
RESTAURANTS.MANAGER_ID%TYPE,PREV_REG RESTAURANTS.REG_NUM%TYPE)
IS
BEGIN
UPDATE RESTAURANTS SET
REG_NUM=REG,NAME=NM,BRANCH=BR,CONTACT_NUM=CONTACT,EMAIL=EM,MANAGER_ID=M_ID WHERE
REG_NUM=PREV_REG;
END EDIT_RES;

PROCEDURE DELETE_RES(REG RESTAURANTS.REG_NUM%TYPE, STAT OUT VARCHAR2)
IS
BEGIN
DELETE FROM RESTAURANTS WHERE REG_NUM=REG;
STAT := 'Restaurant deleted !';
END DELETE_RES;
END RES_MODEL ;
```

```
CREATE PACKAGE EMP_MODEL AS
PROCEDURE ADD_EMP(E_ID EMPLOYEES.EMPLOYEE_ID%TYPE,NAME EMPLOYEES.NAME%TYPE,ADD
EMPLOYEES.ADDRESS%TYPE,HIRE VARCHAR2,JOB EMPLOYEES.JOB%TYPE, SAL EMPLOYEES.SALARY%TYPE,MOB
EMPLOYEE_PHONE.PHONE_NUM%TYPE, M_ID EMPLOYEES.MANAGER_ID%TYPE, STAT OUT VARCHAR2);

PROCEDURE EDIT_EMP(E_ID EMPLOYEES.EMPLOYEE_ID%TYPE,NM EMPLOYEES.NAME%TYPE,ADD
EMPLOYEES.ADDRESS%TYPE,
HIRE VARCHAR2,JB EMPLOYEES.JOB%TYPE, SAL EMPLOYEES.SALARY%TYPE,M_ID
EMPLOYEES.MANAGER_ID%TYPE);
```

```
PROCEDURE DELETE_EMP(E_ID EMPLOYEES.EMPLOYEE_ID%TYPE, STAT OUT VARCHAR2);
END EMP_MODEL;
```

```
CREATE OR REPLACE PACKAGE BODY EMP_MODEL AS
PROCEDURE ADD_EMP(E_ID EMPLOYEES.EMPLOYEE_ID%TYPE,NAME EMPLOYEES.NAME%TYPE,ADD
EMPLOYEES.ADDRESS%TYPE,HIRE VARCHAR2,JOB EMPLOYEES.JOB%TYPE, SAL EMPLOYEES.SALARY%TYPE,MOB
EMPLOYEE_PHONE.PHONE_NUM%TYPE, M_ID EMPLOYEES.MANAGER_ID%TYPE, STAT OUT VARCHAR2)
IS
BEGIN
IF (CHECK_UNIQUE_EMP(E_ID) = 'TRUE') THEN
INSERT INTO EMPLOYEES VALUES(E_ID,NAME,ADD,TO_DATE(HIRE,'yyyy-mm-dd'),JOB,SAL,M_ID);
INSERT INTO EMPLOYEE_PHONE VALUES(E_ID,MOB);
STAT := 'Employee added !';
ELSE
STAT := 'Employee ID is not unique !';
END IF;
END ADD_EMP;

PROCEDURE EDIT_EMP(E_ID EMPLOYEES.EMPLOYEE_ID%TYPE,NM EMPLOYEES.NAME%TYPE,ADD
EMPLOYEES.ADDRESS%TYPE,
HIRE VARCHAR2,JB EMPLOYEES.JOB%TYPE, SAL EMPLOYEES.SALARY%TYPE,M_ID
EMPLOYEES.MANAGER_ID%TYPE)
IS
BEGIN
UPDATE EMPLOYEES SET NAME=NM,ADDRESS=ADD,HIRE_DATE=TO_DATE(HIRE,'yyyy-mm-
dd'),JOB=JB,SALARY=SAL,MANAGER_ID=M_ID WHERE EMPLOYEE_ID=E_ID;
END EDIT_EMP;

PROCEDURE DELETE_EMP(E_ID EMPLOYEES.EMPLOYEE_ID%TYPE, STAT OUT VARCHAR2)
IS
BEGIN
DELETE FROM EMPLOYEES WHERE EMPLOYEE_ID=E_ID;
STAT := 'Employee deleted !';
END DELETE_EMP;
END EMP_MODEL;
```

```
CREATE OR REPLACE PROCEDURE ADD_ITEM(I_NO ITEMS.ITEM_NO%TYPE, DES ITEMS.DESCRIPTION%TYPE,
PRICE ITEMS.PRICE%TYPE, STAT OUT VARCHAR2)
IS
BEGIN
IF (CHECK_UNIQUE_ITEM(I_NO) = 'TRUE') THEN
INSERT INTO ITEMS VALUES (I_NO, DES, PRICE);
STAT := 'Item added successfully !';
ELSE
STAT := 'Item no is not unique !';
END IF;
END;
```

```
CREATE OR REPLACE PROCEDURE DELETE_ITEM(I_NO ITEMS.ITEM_NO%TYPE, STAT OUT VARCHAR2)
IS
BEGIN
DELETE FROM ITEMS WHERE ITEM_NO=I_NO;
STAT := 'Item deleted !';
END;
```

```
CREATE OR REPLACE PROCEDURE GET_CUSTOMER_ID(NAME CUSTOMERS.NAME%TYPE, MOB
CUSTOMERS.PHONE_NUM%TYPE, ID OUT CUSTOMERS.CUSTOMER_ID%TYPE)
IS
BEGIN
```

```
IF(CHECK_EXIST_CUSTOMER(MOB)='TRUE') THEN
SELECT CUSTOMER_ID INTO ID FROM CUSTOMERS WHERE PHONE_NUM=MOB;
ELSE
INSERT INTO CUSTOMERS VALUES (CUSTOMER_SQ.NEXTVAL, NAME, MOB);
SELECT CUSTOMER_ID INTO ID FROM CUSTOMERS WHERE PHONE_NUM=MOB;
END IF;
END;
```

```
CREATE OR REPLACE PROCEDURE GET_ORDER_ID(C_ID CUSTOMERS.CUSTOMER_ID%TYPE,E_ID
EMPLOYEES.EMPLOYEE_ID%TYPE,O_ID OUT ORDERS.ORDER_ID%TYPE)
IS
BEGIN
SELECT ORDER_SQ.NEXTVAL INTO O_ID FROM DUAL;
INSERT INTO ORDERS VALUES (O_ID, '', 'No', C_ID,E_ID);
END;
```

```
CREATE OR REPLACE PROCEDURE GET_ORDER_ITEMS(O_ID ORDERS.ORDER_ID%TYPE, CART OUT
SYS_REFCURSOR)
IS
BEGIN
OPEN CART FOR SELECT * FROM SHOW_CART WHERE ORDER_ID=O_ID;
END;
```

```
CREATE OR REPLACE PROCEDURE ADD_ITEMS_TO_ORDER(O_ID ORDERS.ORDER_ID%TYPE,I_NO
ORDERED_ITEM.ITEM_NO%TYPE, QN ORDERED_ITEM.QUANTITY%TYPE, STAT OUT VARCHAR2)
IS
BEGIN
IF(ITEM_EXIST(I_NO)='TRUE') THEN
IF (ITEM_TO_ORDER_EXIST(I_NO,O_ID)='TRUE') THEN
UPDATE ORDERED_ITEM SET QUANTITY=QN WHERE ORDER_ID=O_ID AND ITEM_NO=I_NO;
STAT := 'Item quantity updated';
ELSE
INSERT INTO ORDERED_ITEM VALUES (ORDERED_ITEM_SQ.NEXTVAL,O_ID,I_NO,QN);
STAT := 'Item added to order';
END IF;
ELSE
STAT := 'Wrong item number';
END IF;
END;
```

```
CREATE OR REPLACE PROCEDURE PLACE_ORDER(O_ID ORDERS.ORDER_ID%TYPE, TOTAL
ORDERS.AMOUNT%TYPE)
IS
BEGIN
UPDATE ORDERS SET AMOUNT=TOTAL, STATUS='Pending' WHERE ORDER_ID=O_ID;
END;
```

```
CREATE OR REPLACE PROCEDURE GET_EMP_ALL_INFO(INFO_E OUT SYS_REFCURSOR)
IS
BEGIN
OPEN INFO_E FOR SELECT * FROM EMPLOYEE_ALL_INFO;
END;
```

```
CREATE OR REPLACE PROCEDURE GET_UNSERVED_ORDER(ORD OUT SYS_REFCURSOR)
IS
BEGIN
OPEN ORD FOR SELECT * FROM ORDERS_NOT_SERVED;
END;
```

```
CREATE OR REPLACE PROCEDURE GET_PENDING_ORDER(ORD OUT SYS_REFCURSOR)
IS
BEGIN
OPEN ORD FOR SELECT ORDER_ID, AMOUNT, STATUS FROM ORDERS WHERE STATUS='Pending';
END;
```

```
CREATE OR REPLACE PROCEDURE CANCEL_ORDER(O_ID ORDERS.ORDER_ID%TYPE)
IS
BEGIN
UPDATE ORDERS SET STATUS='Canceled' WHERE ORDER_ID=O_ID;
END;
```

```
CREATE OR REPLACE PROCEDURE DELIVER_ORDER(O_ID ORDERS.ORDER_ID%TYPE)
IS
BEGIN
UPDATE ORDERS SET STATUS='Delivered' WHERE ORDER_ID=O_ID;
END;
```

Triggers

```
CREATE OR REPLACE TRIGGER EMP_LOG
AFTER INSERT OR DELETE OR UPDATE ON EMPLOYEES
DECLARE
OPNAME LOG_INFO.OPERATION%TYPE;
BEGIN
IF INSERTING THEN
OPNAME :='Insert operation';
ELSIF UPDATING THEN
OPNAME :='Update operation';
ELSE
OPNAME :='Delete operation';
END IF;
INSERT INTO LOG_INFO VALUES(LOG_SQ.NEXTVAL,OPNAME,'Employee',SYSDATE);
END;
```

```
CREATE OR REPLACE TRIGGER RES_LOG
AFTER INSERT OR DELETE OR UPDATE ON RESTAURANTS
DECLARE
OPNAME LOG_INFO.OPERATION%TYPE;
BEGIN
IF INSERTING THEN
OPNAME :='Insert operation';
ELSIF UPDATING THEN
OPNAME :='Update operation';
ELSE
OPNAME :='Delete operation';
END IF;
INSERT INTO LOG_INFO VALUES(LOG_SQ.NEXTVAL,OPNAME,'Restaurants',SYSDATE);
END;
```

```
CREATE OR REPLACE TRIGGER ITEM_LOG
AFTER INSERT OR DELETE OR UPDATE ON ITEMS
DECLARE
OPNAME LOG_INFO.OPERATION%TYPE;
BEGIN
IF INSERTING THEN
OPNAME :='Insert operation';
ELSIF UPDATING THEN
OPNAME :='Update operation';
ELSE
OPNAME :='Delete operation';
END IF;
INSERT INTO LOG_INFO VALUES(LOG_SQ.NEXTVAL,OPNAME,'Items',SYSDATE);
END;
```