****

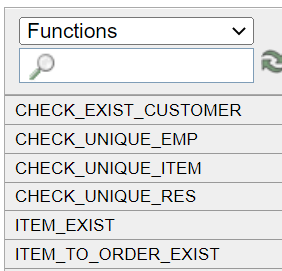
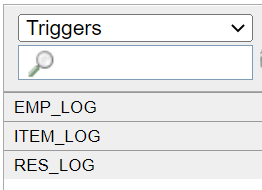
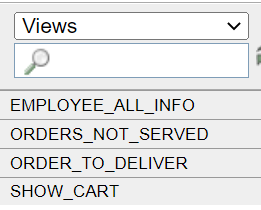
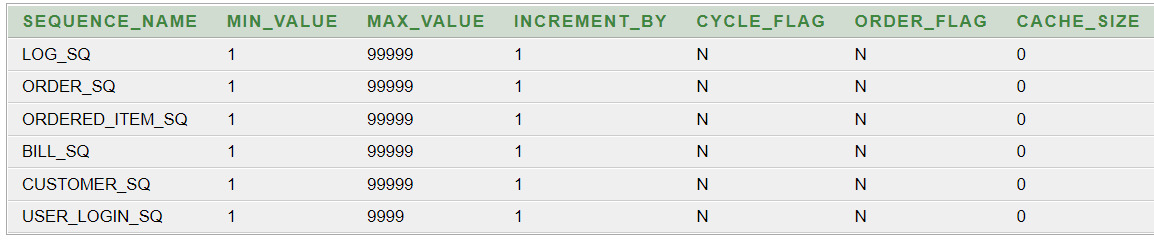
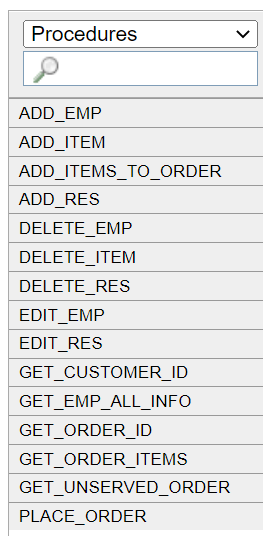
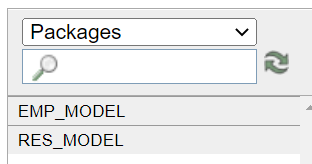
**AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH**

*Course:* Advanced Database Management System (Fall 2022-23)  
*Section:* C

*Project title:* Restaurant Management System

*Group:*

|  |  |  |
| --- | --- | --- |
| Name | ID | Section |
| Saikat Mahmud | 19-41556-3 | C |
| Arafat Al Khair Fuad | 20-43607-1 | C |
| Nusrat Jahan Ekra | 19-40156-1 | A |



2 Package, 15 Procedure

Sequences

4 View

3 Trigger

6 Function

create table Restaurants (

Tables Creation

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)

Functions

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

CREATE PACKAGE RES\_MODEL AS

Package & Procedures

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 TRIGGER EMP\_LOG

Triggers

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;