زينة شحادة – أسماء المغربي

1)

CREATE TABLE restaurants\_offers (

r\_id NUMBER PRIMARY KEY NOT NULL,

r\_details VARCHAR2(200),

r\_cost NUMBER,

r\_name VARCHAR2(50),

r\_address VARCHAR2(100)

);

CREATE TABLE customers (

c\_id NUMBER PRIMARY KEY NOT NULL,

c\_mob\_num NUMBER,

c\_limit NUMBER

);

CREATE TABLE orders (

o\_id NUMBER PRIMARY KEY NOT NULL,

o\_date DATE,

o\_price NUMBER,

o\_status VARCHAR2(50),

o\_c\_id NUMBER,

o\_r\_id NUMBER,

FOREIGN KEY (o\_c\_id) REFERENCES customers(c\_id),

FOREIGN KEY (o\_r\_id) REFERENCES restaurants\_offers(r\_id),

CONSTRAINT chk\_o\_status CHECK (o\_status IN ('new', 'delivered'))

);

2)

INSERT INTO restaurants\_offers VALUES (1, 'nice offer 1', 2400, 'italian', 'Damascus 24');

INSERT INTO restaurants\_offers VALUES (2, 'nice offer 2', 3800, 'italian', 'Damascus 24');

INSERT INTO restaurants\_offers VALUES (3, 'nice offer 3', 1600, 'arab', 'Damascus 50');

INSERT INTO restaurants\_offers VALUES (4, 'nice offer 4', 4000, 'arab', 'Damascus 50');

INSERT INTO restaurants\_offers VALUES (5, 'nice offer 5', 1200, 'indian', 'Damascus 12');

INSERT INTO restaurants\_offers VALUES (6, 'nice offer 6', 7000, 'indian', 'Damascus 12');

INSERT INTO customers VALUES (1, 0946352484, 200);

INSERT INTO customers VALUES (3, 0936475845, 200);

INSERT INTO customers VALUES (4, 0964776453, 200);

INSERT INTO customers VALUES (5, 0935475864, 200);

INSERT INTO customers VALUES (6, 0947353934, 200);

INSERT INTO orders VALUES (1, SYSDATE, 9000, 'new', 1, 2);

INSERT INTO orders VALUES (2, SYSDATE, 12000, 'new', 3, 5);

INSERT INTO orders VALUES (3, SYSDATE, 7000, 'new', 4, 3);

INSERT INTO orders VALUES (4, SYSDATE, 8100, 'delivered', 5, 6);

INSERT INTO orders VALUES (5, SYSDATE, 20000, 'delivered', 6, 1);

3)

CREATE OR REPLACE PROCEDURE print\_offers

IS

CURSOR offer\_cur IS SELECT \* FROM restaurants\_offers;

offer restaurants\_offers%ROWTYPE;

BEGIN

OPEN offer\_cur;

LOOP

FETCH offer\_cur INTO offer;

EXIT WHEN offer\_cur%NOTFOUND;

dbms\_output.put\_line(offer.r\_id || ' ' || offer.r\_details || ' ' || offer.r\_cost || ' ' || offer.r\_name || ' ' || offer.r\_address);

END LOOP;

CLOSE offer\_cur;

END;

4)

CREATE OR REPLACE FUNCTION costu (cu\_id customers.c\_id%TYPE, of\_id restaurants\_offers.r\_id%TYPE)

RETURN boolean

IS

temp customers.c\_id%TYPE;

limi customers.c\_limit%TYPE;

cont NUMBER;

BEGIN

SELECT c\_id, c\_limit INTO temp, limi FROM customers WHERE c\_id= cu\_id;

SELECT COUNT(o\_id) INTO cont FROM orders WHERE o\_c\_id= cu\_id AND o\_status='new';

IF cont<limi THEN

RETURN true;

ELSE

RETURN false;

END IF;

EXCEPTION WHEN no\_data\_found THEN

RETURN null;

END;

5)

CREATE OR REPLACE PROCEDURE order\_offer (or\_id orders.o\_id%TYPE, cu\_id customers.c\_id%TYPE, of\_id restaurants\_offers.r\_id%TYPE)

IS

ok boolean;

deli NUMBER := 5000;

price NUMBER;

BEGIN

ok := costu(cu\_id, of\_id);

IF ok=true THEN

SELECT r\_cost INTO price FROM restaurants\_offers WHERE r\_id=of\_id;

price := deli + price;

INSERT INTO orders VALUES (or\_id, SYSDATE, price, 'new', cu\_id, of\_id);

ELSIF ok=false THEN

dbms\_output.put\_line('Sorry, you cant make any more orders now');

ELSE

dbms\_output.put\_line('Sorry, customer not found');

END IF;

END;

6)

CREATE OR REPLACE PROCEDURE update\_state (or\_id orders.o\_id%TYPE)

IS

BEGIN

UPDATE orders SET o\_status = 'delivered' WHERE o\_id=or\_id;

END;

7)

CREATE TABLE order\_log (

or\_id NUMBER,

or\_status\_o VARCHAR2(50),

or\_status\_n VARCHAR2(50),

or\_price\_o NUMBER,

or\_price\_n NUMBER,

user\_id VARCHAR2(50),

l\_date DATE,

FOREIGN KEY (or\_id) REFERENCES orders(o\_id),

CONSTRAINT chk\_or\_status\_o CHECK (or\_status\_o IN ('new', 'delivered')),

CONSTRAINT chk\_or\_status\_n CHECK (or\_status\_n IN ('new', 'delivered'))

);

CREATE OR REPLACE TRIGGER create\_log

AFTER UPDATE OF o\_status, o\_price ON orders

FOR EACH ROW

BEGIN

IF :OLD.o\_status=:NEW.o\_status AND :OLD.o\_price=:NEW.o\_price THEN

RETURN;

ELSIF :OLD.o\_status=:NEW.o\_status THEN

INSERT INTO order\_log (or\_id, or\_price\_o, or\_price\_n, user\_id, l\_date) VALUES (:NEW.o\_id, :OLD.o\_price, :NEW.o\_price, USER, SYSDATE);

ELSIF :OLD.o\_price=:NEW.o\_price THEN

INSERT INTO order\_log (or\_id, or\_status\_o, or\_status\_n, user\_id, l\_date) VALUES (:NEW.o\_id, :OLD.o\_status, :NEW.o\_status, USER, SYSDATE);

ELSE

INSERT INTO order\_log VALUES (:NEW.o\_id, :OLD.o\_status, :NEW.o\_status, :OLD.o\_price, :NEW.o\_price, USER, SYSDATE);

END IF;

END;

8)

CREATE OR REPLACE PROCEDURE print\_log

IS

CURSOR log\_cur IS SELECT \* FROM order\_log;

log order\_log %ROWTYPE;

BEGIN

dbms\_output.put\_line('Order ID' || ' ' || 'Old Status' || ' ' || 'New Status' || ' ' || 'Old Price' || ' ' || 'New Price' || ' ' || 'User' || ' ' || 'Date');

OPEN log\_cur;

LOOP

FETCH log\_cur INTO log;

EXIT WHEN log\_cur%NOTFOUND;

dbms\_output.put\_line(log.or\_id || ' ' || log.or\_status\_o || ' ' || log.or\_status\_n || ' ' || log.or\_price\_o || ' ' || log. or\_price\_n || ' ' || log.user\_id || ' ' || log.l\_date);

END LOOP;

CLOSE log\_cur;

END;

BEGIN

print\_offers;

order\_offer(8, 2,6);

order\_offer(15, 5, 3);

update\_state(15);

print\_log;

END;