Lab Cycle II - PLSQL

NAME: Kamran Ansari

REG NO: 22MCA0223

Consider the following relational database schema for billing and product tracking system of a departmental store. The name of the tables and column headers are self-explanatory.

```
CUSTOMER(Cus_code, Cus_fname, Cus_lname, Cus_balance)
INVOICE(Inv_no, Cus_code, Inv_date, Inv_amount)
LINE(Inv_no, Line_no, P_code, Line_units, Line_price)
PRODUCT(P_code, P_desc, P_qoh, P_min, P_price, V_code)
VENDOR(V_code, V_name, V_Contact)
The primary keys are underlined and foreign keys are self-explanatory.
```

TABLE CREATION

```
CREATE TABLE CUSTOMER(

CUS_CODE VARCHAR2(10),

CUS_FNAME VARCHAR2(30),

CUS_LNAME VARCHAR(30),

CUS_BALANCE DOUBLE PRECISION,

CONSTRAINT CUSTOMER_CUS_CODE_PK PRIMARY KEY(CUS_CODE)

);

CREATE TABLE INVOICE(

INV_NO VARCHAR2(10),

CUS_CODE VARCHAR2(10),

INV_DATE DATE,
```

```
INV_AMOUNT DOUBLE PRECISION,
    CONSTRAINT INVOICE_INV_NO_PK PRIMARY KEY(INV_NO),
    CONSTRAINT INVOICE_CUS_CODE_FK FOREIGN KEY(CUS_CODE) REFERENCES
CUSTOMER(CUS_CODE)
);
CREATE TABLE VENDOR(
   V_CODE VARCHAR2(10),
   V_NAME VARCHAR2(30),
   V_CONTACT INT,
    CONSTRAINT VENDOR V CODE PK PRIMARY KEY(V CODE)
);
CREATE TABLE PRODUCT(
    P_CODE VARCHAR2(10),
    P_DESC VARCHAR2(100),
    P_QOH INT,
    P MIN DOUBLE PRECISION,
    P_PRICE DOUBLE PRECISION,
   V_CODE VARCHAR2(10),
    CONSTRAINT PRODUCT P CODE PK PRIMARY KEY(P CODE)
);
CREATE TABLE LINE(
    INV_NO VARCHAR2(10),
    LINE_NO VARCHAR2(10),
    P CODE VARCHAR2(10),
    LINE_UNITS INTEGER,
    LINE_PRICE DOUBLE PRECISION,
    CONSTRAINT LINE INV NO LINE NO PK PRIMARY KEY(INV NO, LINE NO),
```

```
CONSTRAINT LINE_INV_NO_FK FOREIGN KEY(INV_NO) REFERENCES
INVOICE(INV_NO),

CONSTRAINT P_CODE FOREIGN KEY(P_CODE) REFERENCES PRODUCT(P_CODE)
);
```

TABLE CREATION OUTPUT

SQL> desc customer Name	Null?	Туре
CUS_CODE CUS_FNAME CUS_LNAME CUS_BALANCE	NOT NULL	VARCHAR2(10) VARCHAR2(30) VARCHAR2(30) FLOAT(126)
SQL> desc invoice Name	Null?	Туре
Name		
INV_NO CUS_CODE INV_DATE INV_AMOUNT	NOT NULL	VARCHAR2(10) VARCHAR2(10) DATE FLOAT(126)
SQL> desc vendor		
Name	Null?	Туре
V_CODE V_NAME V_CONTACT	NOT NULL	VARCHAR2(10) VARCHAR2(30) NUMBER(38)

SQL> desc product Name	Null?	Туре
P_CODE P_DESC P_QOH P_MIN P_PRICE V_CODE	NOT NULL	VARCHAR2(10) VARCHAR2(100) NUMBER(38) FLOAT(126) FLOAT(126) VARCHAR2(10)
SQL> desc line		
Name	Null?	Type
INV_NO LINE_NO P_CODE LINE_UNITS LINE_PRICE		VARCHAR2(10) VARCHAR2(10) VARCHAR2(10) NUMBER(38) FLOAT(126)

TABLE INSERTION

```
INSERT INTO CUSTOMER VALUES(
   1,
   'C1',
   'L1',
   45000
);
INSERT INTO CUSTOMER VALUES(
   2,
   'C2',
   'L2',
   44500
);
INSERT INTO CUSTOMER VALUES(
    3,
   'C3',
   'L3',
    50000
);
INSERT INTO CUSTOMER VALUES(
   4,
   'C4',
    'L4',
    50000
);
INSERT INTO CUSTOMER VALUES(
```

```
5,
    'C5',
    'L5',
    50000
);
INSERT INTO INVOICE VALUES(
    1,
    1,
   '22-FEB-22',
   1200
);
INSERT INTO INVOICE VALUES(
   2,
   2,
   '23-JUN-21',
    200
);
INSERT INTO INVOICE VALUES(
   3,
   3,
   '04-AUG-22',
    20000
);
INSERT INTO INVOICE VALUES(
    4,
   4,
```

```
'04-JUL-22',
    5000
);
INSERT INTO INVOICE VALUES(
    5,
    5,
    '10-SEP-22',
    5000
);
INSERT INTO INVOICE VALUES(
    8005,
    1,
   '27-AUG-20',
    250.50
);
INSERT INTO INVOICE VALUES(
    8006,
    2,
   '04-AUG-22',
    1500.5
);
INSERT INTO VENDOR VALUES(
    1,
    'V1',
    1232323125
);
```

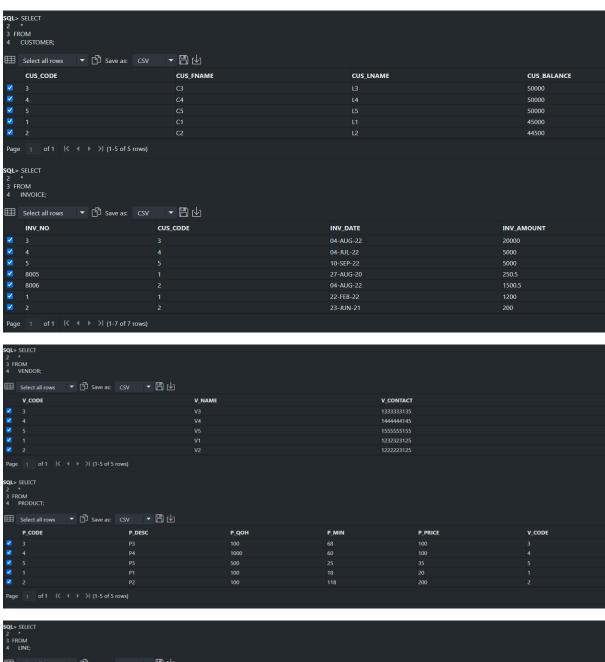
```
INSERT INTO VENDOR VALUES(
   2,
    'V2',
    1222223125
);
INSERT INTO VENDOR VALUES(
    3,
   'V3',
   1333333135
);
INSERT INTO VENDOR VALUES(
   4,
   'V4',
   1444444145
);
INSERT INTO VENDOR VALUES(
    5,
   'V5',
    1555555155
);
INSERT INTO PRODUCT VALUES(
   1,
   'P1',
    100,
    18,
```

```
20,
    1
);
INSERT INTO PRODUCT VALUES(
    2,
    'P2',
    100,
    118,
    200,
    2
);
INSERT INTO PRODUCT VALUES(
    3,
    'P3',
    100,
    68,
    100,
    3
);
INSERT INTO PRODUCT VALUES(
    4,
    'P4',
    1000,
    60,
    100,
    4
);
```

```
INSERT INTO PRODUCT VALUES(
    5,
    'P5',
    500,
    25,
    35,
    5
);
INSERT INTO LINE VALUES(
    1,
    1,
    1,
    20,
    19
);
INSERT INTO LINE VALUES(
    2,
    2,
    2,
    45,
    119
);
INSERT INTO LINE VALUES(
    3,
    3,
    3,
```

```
78,
    69
);
INSERT INTO LINE VALUES(
    4,
    4,
   4,
    55,
    100
);
INSERT INTO LINE VALUES(
    5,
    5,
    5,
    50,
    30
);
```

TABLE INSERTION OUTPUT



SQLS- SELECT 2 3 FROM 4 LINE;						
⊞ Selectall rows ▼ 🖒 Save as: CSV ▼ 🖺 🖟						
	INV_NO	LINE_NO	P_CODE	LINE_UNITS	LINE_PRICE	
₩					69	
M					100	
					30	
					19	
					119	
Page	Page 1 of 1 K ← ► > (1-5 of 5 rows)					

1. Write a procedure to add a new customer to the CUSTOMER table. Use the following values in the new record: <1002, 'Rauthor', 'Peter', 0.00>.

```
CREATE OR REPLACE PROCEDURE INSERTCUSTOMER (
    CUS_CODE IN CUSTOMER.CUS_CODE%TYPE,
    CUS_FNAME IN CUSTOMER.CUS_FNAME%TYPE,
    CUS_LNAME IN CUSTOMER.CUS_LNAME%TYPE,
    CUS_BALANCE IN CUSTOMER.CUS_BALANCE%TYPE
) IS
BEGIN
    INSERT INTO CUSTOMER VALUES (
        CUS_CODE,
        CUS_FNAME,
        CUS_LNAME,
        CUS BALANCE
    );
END;
/
EXECUTE INSERTCUSTOMER(1002, 'RAUTHOR', 'PETER', 0.0);
SELECT
FROM
    CUSTOMER;
```



2. Write a procedure to add a new invoice record to the INVOICE table. Use the following values in the new record: <8006, 1000, '30-APR-16', 301.72>.
Run a query to see if the record has been added.

```
CREATE OR REPLACE PROCEDURE PROD ADD INV (
    I NO IN INVOICE. INV NO%TYPE,
    C_CODE IN INVOICE.CUS_CODE%TYPE,
    I_DATE IN INVOICE.INV_DATE%TYPE,
    I_AMT IN INVOICE.INV_AMOUNT%TYPE
) IS
BEGIN
    INSERT INTO CUSTOMER(
        CUS CODE
    ) VALUES(
        C CODE
    );
    INSERT INTO INVOICE VALUES(
        I_NO,
        C_CODE,
        I_DATE,
        I AMT
    );
    COMMIT;
END;
/
EXECUTE PROD_ADD_INV(8006, 1000, '30-APR-2016', 301.72);
SELECT
```

*

FROM

INVOICE;



3. Write a PL/SQL function to compute purchase made by a given customer for a particular invoice. Test the function in another function to compute the total purchase made by a customer.

```
CREATE OR REPLACE FUNCTION GETINVOICE(
    CUSCODE NUMBER
) RETURN NUMBER IS
    AMT NUMBER;
BEGIN
    SELECT
        SUM(INV_AMOUNT) INTO AMT
    FROM
        INVOICE
    WHERE
        INVOICE.CUS CODE = CUSCODE;
    RETURN AMT;
    COMMIT;
END;
/
DECLARE
    A NUMBER;
BEGIN
    A := GETINVOICE(1);
    DBMS_OUTPUT.PUT_LINE(A);
END;
/
```

```
SQL> CREATE OR REPLACE FUNCTION GETINVOICE(
 2 CUSCODE NUMBER
 3 ) RETURN NUMBER IS
 4 AMT NUMBER;
 5 BEGIN
 6 SELECT
       SUM(INV_AMOUNT) INTO AMT
 8 FROM
 9
      INVOICE
 10
     WHERE
       INVOICE.CUS_CODE = CUSCODE;
 11
 12 RETURN AMT;
 13
     COMMIT;
 14 END;
 15 /
FUNCTION created.
Commit complete.
SQL> DECLARE
2 A NUMBER;
 3 BEGIN
4 A := GETINVOICE(1);
 5 DBMS_OUTPUT.PUT_LINE(A);
 6 END;
 7 /
1450.5
PL/SQL procedure successfully completed.
Commit complete.
```

4. Write a procedure to delete an invoice, giving the invoice number as a parameter. Test the procedure by deleting invoices 8005 and 8006.

```
SQL> CREATE OR REPLACE PROCEDURE PROD_DEL_INV (
     IN_NO IN NUMBER
 3 ) AS
 4 BEGIN
    IF IN_NO IS NOT NULL THEN
       DELETE FROM INVOICE
 7
       WHERE
         INV_NO = IN_NO;
 9 END IF;
 10 END;
 11 /
PROCEDURE created.
Commit complete.
SQL> EXEC PROD_DEL_INV(8005);
PL/SQL procedure successfully completed.
Commit complete.
SQL> EXEC PROD_DEL_INV(8006);
PL/SQL procedure successfully completed.
Commit complete.
```

SQL>	SQL> SELECT *FROM INVOICE;					
	Select all rows ▼ 🖺 Save as: CSV	□ - 13 4				
	INV_NO	CUS_CODE	INV_DATE	INV_AMOUNT		
~			04-AUG-22	20000		
~			04-JUL-22	5000		
~			10-SEP-22	5000		
~			22-FEB-22	1200		
~			23-JUN-21	200		
Page	1 of 1 <					

5. Write a procedure to display the INV_SUBTOTAL, INV_TAX, and INV_TOTAL. The procedure takes the invoice number as a parameter. The INV_SUBTOTAL is the sum of the LINE_TOTAL amounts for the invoice, the INV_TAX is the product of the INV_SUBTOTAL and the tax rate (8 percent), and the INV_TOTAL is the sum of the INV_SUBTOTAL and the INV_TAX.

```
CREATE OR REPLACE PROCEDURE PROD_TOTALS_INV(
    INVNUM IN NUMBER
) AS
    INV SUBTOTAL NUMBER;
    INV TAX
                 NUMBER;
    INV TOTAL
                 NUMBER;
    CNT
                 NUMBER;
BEGIN
    SELECT
        COUNT(*) INTO CNT
    FROM
        INVOICE
    WHERE
        INV NO = INVNUM;
    IF CNT = 1 THEN
        SELECT
            SUM(LINE_UNITS * LINE_PRICE) INTO INV_SUBTOTAL
        FROM
            LINE
        WHERE
            LINE.INV NO = INVNUM;
        INV_TAX := 0.08 * INV_SUBTOTAL;
        INV_TOTAL := INV_SUBTOTAL+ INV_TAX;
```

SQL> EXECUTE PROD_TOTALS_INV(1);

SUBTOTAL: 380

TAX : 30.4 TOTAL : 410.4

PL/SQL procedure successfully completed.

6. Write suitable PL/SQL code to display the list of vendors who must be contacted whenever a product reaches reorder level.

```
CREATE OR REPLACE TRIGGER TRIG_MIN_VENDOR AFTER
    INSERT OR DELETE OR UPDATE ON PRODUCT FOR EACH ROW
WHEN(NEW.P_QOH < OLD.P_MIN)
DECLARE
    V_NO VENDOR.V_CODE%TYPE:= :NEW.V_CODE;
    VNAME VENDOR.V_NAME%TYPE;
    VCONT VENDOR.V_CONTACT%TYPE;
    CURSOR C1 IS
        SELECT
            V NAME,
            V CONTACT
        FROM
            VENDOR
        WHERE
            V_CODE=V_NO;
BEGIN
    OPEN C1;
    LO<sub>O</sub>P
        FETCH C1 INTO VNAME, VCONT;
        EXIT WHEN C1%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE( CHR(13)
            ||CHR(10)
            ||'V_Name: '
            | | VNAME
            || Contact Number: '
            ||VCONT);
    END LOOP;
```

```
CLOSE C1;
END;
/

UPDATE PRODUCT
SET
    P_QOH = 15
WHERE
    P_CODE = 1;
```

```
SQL> CREATE OR REPLACE TRIGGER TRIG_MIN_VENDOR AFTER
2 INSERT OR DELETE OR UPDATE ON PRODUCT FOR EACH ROW WHEN(NEW.P_QOH < OLD.P_MIN)
3 DECLARE
4
    V_NO VENDOR.V_CODE%TYPE:= :NEW.V_CODE;
    VNAME VENDOR.V_NAME%TYPE;
    VCONT VENDOR.V_CONTACT%TYPE;
    CURSOR C1 IS
8
      SELECT
9
        V_NAME,
10
         V_CONTACT
11
       FROM
         VENDOR
 12
13
       WHERE
         V_CODE=V_NO;
14
15 BEGIN
16
    OPEN C1;
17
    LOOP
18
       FETCH C1 INTO VNAME, VCONT;
 19
       EXIT WHEN C1%NOTFOUND;
20
       DBMS_OUTPUT.PUT_LINE( CHR(13)
21
         ||CHR(10)
22
         ||'V_Name: '
23
         ||VNAME
         || Contact Number: '
24
25
         [VCONT);
    END LOOP;
26
27 CLOSE C1;
28 END;
29 /
TRIGGER created.
Commit complete.
SQL> UPDATE PRODUCT
2 SET
   P_QOH = 15
4 WHERE
5 P_CODE = 1;
1 row updated.
Commit complete.
```

7. Write the trigger to update the CUST_BALANCE in the CUSTOMER table when a new invoice record is entered. (Assume that the sale is a credit sale.) Test the trigger using the following new INVOICE record: <8005, 1001, '27-APR-16', 225.40>.

```
INSERT INTO CUSTOMER VALUES(
    1001,
    'R1',
    'C1',
    20000
);
CREATE OR REPLACE TRIGGER TRIG NEW INV AFTER
    INSERT ON INVOICE FOR EACH ROW
BEGIN
    UPDATE CUSTOMER
    SET
        CUS_BALANCE = CUS_BALANCE + :NEW.INV_AMOUNT
    WHERE
        CUS_CODE = :NEW.CUS_CODE;
END;
INSERT INTO INVOICE VALUES(
    8005,
    1001,
    '27-APR-16',
    225.40
);
SELECT
```

FROM

CUSTOMER;



8. Write a trigger to update the customer balance when an invoice is deleted.

```
CREATE OR REPLACE TRIGGER TRIG_INV_DEL AFTER
    DELETE ON INVOICE FOR EACH ROW
BEGIN
    UPDATE CUSTOMER
    SET
       CUS_BALANCE = CUS_BALANCE - :OLD.INV_AMOUNT
    WHERE
       CUS_CODE = :OLD.CUS_CODE;
END;
/
SELECT
FROM
    INVOICE;
SELECT
FROM
    CUSTOMER;
DELETE FROM INVOICE
WHERE
    INV_NO = 8005;
SELECT
FROM
```

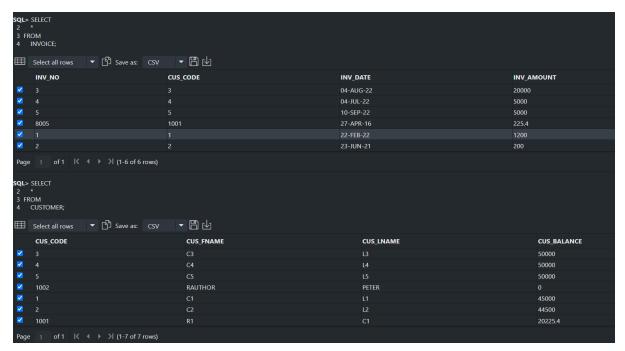
INVOICE;

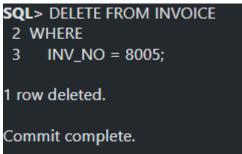
SELECT

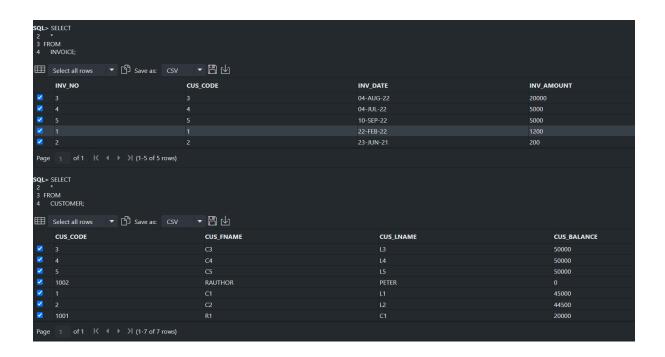
*

FROM

CUSTOMER;



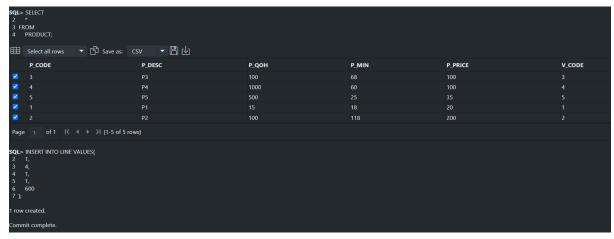


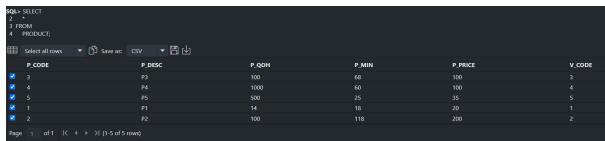


9. Write a trigger that automatically updates the quantity on hand for each product sold after a new LINE row is added.

```
CREATE OR REPLACE TRIGGER TRIG_NEW_LINE AFTER
    INSERT ON LINE FOR EACH ROW
BEGIN
    UPDATE PRODUCT
    SET
        P_QOH = P_QOH - :NEW.LINE_UNITS
    WHERE
        PRODUCT.P_CODE = :NEW.P_CODE;
END;
/
SELECT
FROM
   PRODUCT;
INSERT INTO LINE VALUES(
    1,
    4,
    1,
    1,
    600
);
SELECT
FROM
```

PRODUCT;





10. Write a trigger to throw exception whenever the invoice amount exceeds customer balance.

```
CREATE OR REPLACE TRIGGER CHECK_CUSTOMER_BALANCE BEFORE
    INSERT ON INVOICE FOR EACH ROW
DECLARE
    ERR_MSG VARCHAR2(255) := 'Invoice amount exceeds customer
balance';
    CUS_BALANCE NUMBER;
BEGIN
    SELECT
        CUS_BALANCE INTO CUS_BALANCE
    FROM
        CUSTOMER
    WHERE
        CUS_CODE = :NEW.CUS_CODE;
    IF :NEW.INV AMOUNT>CUS BALANCE THEN
        RAISE_APPLICATION_ERROR(-20001, ERR_MSG);
    END IF;
END;
INSERT INTO INVOICE (
    INV_NO,
    CUS_CODE,
    INV_DATE,
    INV AMOUNT
) VALUES (
    8006,
    1001,
    '30-APR-16',
    301.72
```

```
SQL> CREATE OR REPLACE TRIGGER check_customer_balance
 2 BEFORE INSERT ON INVOICE
 3 FOR EACH ROW
 4 DECLARE
        err_msg VARCHAR2(255) := 'Invoice amount exceeds customer balance';
        cus_balance NUMBER;
 6
 7
    BEGIN
 8
        SELECT cus_balance INTO cus_balance FROM CUSTOMER WHERE cus_code = :new.cus_code;
        IF :new.inv_amount > cus_balance THEN
 9
          RAISE_APPLICATION_ERROR(-20001, err_msg);
 10
 11
        END IF;
 12 END;
 13 /
Trigger created.
SQL> INSERT INTO INVOICE (inv_no, cus_code, inv_date, inv_amount)
 2 VALUES (8006, 1001, '30-APR-16', 301.72);
INSERT INTO INVOICE (inv_no, cus_code, inv_date, inv_amount)
ERROR at line 1:
ORA-20001: Invoice amount exceeds customer balance
ORA-06512: at "UNIV_ADMIN.CHECK_CUSTOMER_BALANCE", line 7
ORA-04088: error during execution of trigger
'UNIV_ADMIN.CHECK_CUSTOMER_BALANCE'
SQL> SELECT cus_code, cus_balance FROM CUSTOMER WHERE cus_code = 1001;
  CUS_CODE CUS_BALANCE
      1001
                   100
```