SQL Exercises

Chapter 7

1. making changes to a PRODUCT table

1. making changes to a PRODUCT table

```
UPDATE  PRODUCT
SET P_INDATE = '18-JAN-2004'
WHERE P_CODE = '13-Q2/P2';
```

2. delete the table row where the P_CODE is ‘BRT-345’.

2. delete the table row where the P_CODE is ‘BRT-345’.

```
DELETE FROM PRODUCT
WHERE P_CODE = ‘BRT-345’;
```

3. output the table contents when the value of V_CODE is equal to 21344?

3. output the table contents when the value of V_CODE is equal to 21344?

```
SELECT P_DESCRIPT, P_INDATE, P_PRICE,
V_CODE
FROM PRODUCT
WHERE V_CODE = 21344;
```

4. output the table contents when
the value of V_CODE is NOT equal to 21344?

4. output the table contents when
the value of V_CODE is NOT equal to 21344?

```
SELECT P_DESCRIPT, P_INDATE, P_PRICE,
V_CODE
FROM PRODUCT
WHERE V_CODE <> 21344;
```

5. output the table contents when the value of the character field P_CODE
is alphabetically less than 1558-QW1?

5. output the table contents when the value of the character field P_CODE
is alphabetically less than 1558-QW1?

```
SELECT P_CODE, P_DESCRIPT, P_QOH, P_MIN, P_PRICE
FROM PRODUCT
WHERE P_CODE <‘1558-QW1’;
```

6. list all the rows in which the inventory stock dates occur on or after January 20, 2016?

6. list all the rows in which the inventory stock dates occur on or after January 20, 2016?

```
SELECT P_DESCRIPT, P_QOH, P_MIN, P_PRICE,
P_INDATE
FROM PRODUCT
WHERE P_INDATE >= ‘20-JAN-2016’;
```

7. determine the total value of inventory held on hand

7. determine the total value of inventory held on hand

```
SELECT SUM(P_QOH*P_PRICE)
FROM PRODUCT;
```

8. List the table contents for either V_CODE = 21344 or V_CODE = 24288?

8. List the table contents for either V_CODE = 21344 or V_CODE = 24288?

```
SELECT P_DESCRIPT, P_INDATE, P_PRICE, V_COD
FROM PRODUCT
WHERE V_CODE = 21344 OR V_CODE = 24288;
```

9. The query to join the P_DESCRIPT and P_PRIC fields from the PRODUCT table and the V_NAME, V_AREACODE, V_PHONE, and V_CONTACT fields from the VENDOR table where the values of V_CODE.

9. The query to join the P_DESCRIPT and P_PRIC fields from the PRODUCT table and the V_NAME, V_AREACODE, V_PHONE, and V_CONTACT fields from the VENDOR table where the values of V_CODE.

```
    SELECT P_DESCRIPT, P_PRICE, V_NAME,
    V_CONTACT, V_AREACODE, V_PHONE
    FROM PRODUCT, VENDOR
WHERE PRODUCT.V_CODE = VENDOR.V_CODE;
```

10. The query to join the P_DESCRIPT and P_PRI
fields from the PRODUCT table and the V_NAM
V_AREACODE, V_PHONE and V_CONTACT fields
the VENDOR table, where the values of V_CODE
match and the output is ordered by the price.

10. The query to join the P_DESCRIPT and P_PRI fields from the PRODUCT table and the V_NAM V_AREACODE, V_PHONE and V_CONTACT fields the VENDOR table, where the values of V_CODE match and the output is ordered by the price.

```
    SELECT  PRODUCT.P_DESCRIPT,  PRODUCT.P_PRICE,
    VENDOR.V_NAME, VENDOR.V_CONTACT,
VENDOR.V_AREACODE, VENDOR.V_PHONE
FROM PRODUCT, VENDOR
WHERE PRODUCT.V_CODE = VENDOR.V_CODE
ORDER BY PRODUCT.P_PRICE;
```

Q1. Here is a table for recording guests' stays at a hotel (arrDate denotes arrival date, depDate is departure date):

```
CREATE TABLE HotelStays
(roomNum INTEGER NOT NULL,
arrDate DATE NOT NULL,
depDate DATE NOT NULL,
guestName CHAR(30) NOT NULL,
PRIMARY KEY (roomNum, arrDate));
```

There are two problems (issues) with the above. First, the arrival date could be incorrectly entered to be later than the departure date. Second, a new entry (for a new guest) could be accidentally put in for a room number, even before the existing guest in that room has checked out:

How would you redesign the table to fix both these issues? For your answer, you can either provide a textual explanation, and/or provide SQL statements. Hint - "do not be concerned with efficiency" - ANY working solution is acceptable :)

RoomNumber Date GuestName

```
CREATE TABLE HotelStays
(roomNum INTEGER NOT NULL,
date DATE NOT NULL,
guestName CHAR(30) NOT NULL,
PRIMARY KEY (roomNum, date));
```

In other words, create a distinct row for each day a guest stays in a room! That will avoid duplicates (overlaps), and get rid of the problem of end<start.

```
CREATE OR REPLACE FUNCTION date_validator() RETURNS TRIGGER AS $exe$
  BEGIN
    IF NEW.arrDate > NEW.depDate Then
      delete from HotelStays where (roomNum = NEW.roomNum AND arrDate =
NEW.arrDate);
    ELSEIF EXISTS(select arrDate, depDate from HotelStays where ((NEW.arrDate >arrDate
AND NEW.arrDate<depDate)OR (NEW.depDate >arrDate AND NEW.depDate<depDate)OR
(NEW.depDate >depDate AND NEW.arrDate<arrDate))) Then
      delete from HotelStays where (roomNum = NEW.roomNum AND arrDate =
NEW.arrDate);
    END IF;
    RETURN NEW;
  END;
$exe$
Language plpgsql;

create trigger date_valid
AFTER INSERT ON HotelStays
FOR EACH ROW EXECUTE PROCEDURE date_validator();

INSERT INTO HotelStays VALUES (123, to_date('20160202', 'YYYYMMDD'),
to_date('20160206','YYYYMMDD'), 'A');
INSERT INTO HotelStays VALUES (123, to_date('20160204', 'YYYYMMDD'),
```

http://www.w3resource.com/sql-exercises/sql-retrieve-from-table.php

Schema for Exercises 6-11