Joins, Procedures and Triggers

1) print the store name, title name, quantity, sale amount, pulisher name, author name for all the sales. Also print those books which have not been sold and authors who have not written.

```
select stor_name 'Store Name',title 'Book',qty 'Quanity',(ytd_sales*price) as 'Sale Amount',
pub_name 'Publisher name',au_fname+' '+au_lname 'Author Name'
from stores s join sales sa on s.stor_id=sa.stor_id right outer join titles t
on t.title_id=sa.title_id
join publishers p on t.pub_id=p.pub_id join titleauthor ta on ta.title_id=t.title_id
right outer join authors a on a.au_id=ta.au_id
```

2) Create a stored procedure that will take the author's name and print the total sales amount for all the books authored by him/her

Note: - If there are no books sold then print "Sale yet to gear up"

```
create PROCEDURE total_sales_author (@author_name VARCHAR(60))
AS
BEGIN
DECLARE @total_sales DECIMAL(10,2);
SET @total_sales = (SELECT sum(s.qty )* sum(t.price)  FROM sales s
INNER JOIN titles t ON s.title_id = t.title_id
INNER JOIN titleauthor ta ON t.title_id = ta.title_id
INNER JOIN authors a ON ta.au_id = a.au_id
WHERE a.au_fname + ' ' + a.au_lname = @author_name)
IF @total_sales IS NULL OR @total_sales = 0
BEGIN
PRINT 'Sale yet to gear up';
END
ELSE
BEGIN
PRINT 'The total sales amount for ' + @author_name + ' is ' + CAST(@total_sales AS VARCHAR);
END;
END;

EXEC total_sales_author @author_name = 'Green Marjorie';
```

3) print the details of the sale when the sale quantity is greater than the sale quantity of all the same titles sold in the same store

```
SELECT
    s.stor_id AS 'Store ID',
    s.title_id AS 'Title ID',
    t.title AS 'Title',
    s.qty AS 'Sale Quantity',
    s.ord_date AS 'Sale Date'
FROM sales s
left JOIN (
    SELECT
        s.stor_id,
        s.title_id,
        MAX(s.qty) AS max_qty
    FROM sales s
    GROUP BY s.stor_id, s.title_id
) max_sale_qty ON s.stor_id = max_sale_qty.stor_id
    AND s.title_id = max_sale_qty.title_id
    AND s.qty > max_sale_qty.max_qty
JOIN titles t ON s.title_id = t.title_id
```

4) Print the average price of every author's book with the author's full name

```
select concat(au_fname,' ',au_lname) as AuthorName, title 'Book' ,avg(price) 'Average Price' from
authors a join titleauthor ta on a.au_id=ta.au_id join titles t on t.title_id=ta.title_id
group by concat(au_fname,' ',au_lname),title
```

5) Print the schema of the titles table and locate all the constraints

```
sp_help titles
```

6) Create a procedure that will take the price and prints the count of book that are priced less than that

```
create procedure proc_CountBooks (@Price decimal(10, 2))
as
begin
    select count(*) 'No. of Books' from titles where price < @Price
END

EXEC proc_CountBooks 20
```

7) Find a way to ensure that the price of books is not updated if the price is below 7

```
CREATE TRIGGER trgPreventPriceUpdate
ON titles
FOR UPDATE
AS
BEGIN
    IF UPDATE(price) AND EXISTS (SELECT 1 FROM inserted WHERE price < 7)
    BEGIN
        ROLLBACK;
        PRINT 'Price update is not allowed for prices below 7.';
    END
END;
```

8) print the books that have 'e' and 'a' in their name

```
select title 'Titles' from titles where title like '%e%' and title like '%a%'
```