SQL Exercise 4 – Jason Chan

1. List the first and last name of the employee having a last name beginning with **Thomas**. Use local variables for the first and last name and the **@@ROWCOUNT** command. Display the first and last name if the name is found, and the message ‘Employee not found’ if the last name does not exist in the EMPLOYEE table. The query should produce the result set listed below.

```
Employee Name is Gary Thomas
```

```
DECLARE       @lname varchar(30)
DECLARE       @fname varchar(30)
SET           @lname = ' '
SET           @fname = ' '
SELECT        @lname = lname,
              @fname = fname
FROM          employee
WHERE         lname LIKE 'Thomas%'
IF      @@ROWCOUNT > 0
        PRINT 'Employee Name is ' + @fname + ' ' + @lname
ELSE
        PRINT 'Employee not found'
```

2. List employees with a hire date between **January 1 1989** and **December 31 1990**. Display the employee ID, first name, last name, hire date, and job ID from the EMPLOYEE table, and the job description from the JOB table. Use local variables for the two dates. Display the name of the employee as the last name, followed by a comma and a space, followed by the first name. Display the hire date in the format of **MMM DD YYYY**. Order the result set by the employee name. The query should produce the result set listed below.

```
EmployeeID      Name                    HireDate        JobID   Description
-----------------  ------------------------  -----------------  --------  ----------------------------
PSA89086M       Afonso, Pedro           Dec 24 1990     14      Designer
VPA30890F       Ashworth, Victoria      Sep 13 1990     6       Managing Editor
H-B39728F       Bennett, Helen          Sep 21 1989     12      Editor
.....
A-R89858F       Roulet, Annette         Feb 21 1990     6       Managing Editor
MFS52347M       Sommer, Martin          Apr 13 1990     10      Productions Manager
DBT39435M       Tonini, Daniel          Jan 1 1990      11      Operations Manager
(15 row(s) affected)
```

```
DECLARE       @hire_date1 datetime
DECLARE       @hire_date2 datetime
SET           @hire_date1 = 'Jan 1 1989'
SET           @hire_date2 = 'Dec 31 1990'
SELECT        e.emp_id AS EmployeeID,
              (e.lname + ', ' + e.fname) AS Name,
              CONVERT(char(12),e.hire_date,109) AS HireDate,
              e.job_id AS JobID,
              j.job_desc AS Description
FROM          employee e
INNER JOIN    jobs j ON e.job_id = j.job_id
WHERE         e.hire_date BETWEEN @hire_date1 AND @hire_date2
ORDER BY Name
```

3. Create a stored procedure called **author_information** which takes **2 input parameters** consisting of an author ID and a title ID, and returns **3 output parameters** consisting of the last name, first name, and royalty percentage. If the author ID and title ID matches the input parameters, the stored procedure should return the last and first names from the AUTHORS table, and the royalty percentage from the TITLEAUTHOR table.

```
CREATE PROCEDURE author_information
(      @au_id varchar(11),
       @title_id varchar(6),
       @last_name varchar(40) OUTPUT,
       @first_name varchar(20) OUTPUT,
       @royaltyper int OUTPUT )
AS
SELECT @last_name = a.au_lname,
       @first_name = a.au_fname,
       @royaltyper = ta.royaltyper
FROM          authors a
INNER JOIN    titleauthor ta ON a.au_id = ta.au_id
WHERE         a.au_id = @au_id
AND           ta.title_id = @title_id
GO
```

4. Run the stored procedure **author_information** for author ID **‘672-71-3249’** and the title ID ‘**TC7777’**. Display the output values from the stored procedure for the first name, last name, and royalty percentage. The stored procedure should produce the result below.

```
Author: Akiko Yokomoto
Royalty percentage = 40
```

```
DECLARE @last_name varchar(40)
DECLARE @first_name varchar(40)
DECLARE @royaltyper int
EXEC author_information    '672-71-3249','TC7777',
                           @last_name OUTPUT,
                           @first_name OUTPUT,
                           @royaltyper OUTPUT
PRINT 'Author: ' + @first_name + ' ' + @last_name
PRINT 'Royalty percentage = ' + CONVERT(char(20),@royaltyper)
```

5. Create a stored procedure called **store_information** which takes an **input variable** for the price of a book. The stored procedure should list the store ID and order date from the SALES table, the store name from the STORES table, and the title id, price, and advance from the TITLES table, where the price is greater than or equal to the input variable. Display the order date in the format of **YYYY.MM.DD**. Order the result set by the store ID.

```
CREATE PROCEDURE store_information
(      @price money )
AS
SELECT s.stor_id                        AS StoreID,
       st.stor_name                     AS Name,
       CONVERT(CHAR(12),s.ord_date,102) AS OrderDate,
       t.title_id                       AS TitleID,
       t.price                          AS Price,
       t.advance                        AS Advance
FROM sales s
INNER JOIN stores st       ON s.stor_id = st.stor_id
INNER JOIN titles t        ON s.title_id = t.title_id
WHERE t.price >= @price
ORDER BY s.stor_id
GO
```

6. Run the stored procedure **store_information** using a value of **$15.00** for the price. The stored procedure should produce the result set listed below.

```
Store_ID     Name                                        OrderDate      TitleID   Price    Advance
-----------  ------------------------------------------  -------------  --------- -------  -----------
6380         Eric the Read Books                         1994.09.14     BU1032    19.99    5000.00
7066         Barnum's                                    1993.05.24     PC8888    20.00    8000.00
7067         News & Brews                                1992.06.15     TC3218    20.95    7000.00
7131         Doc-U-Mat: Quality Laundry and Books        1993.05.29     PS1372    21.59    7000.00
7131         Doc-U-Mat: Quality Laundry and Books        1993.05.29     PS3333    19.99    2000.00
7896         Fricative Bookshop                          1993.10.28     BU7832    19.99    5000.00
7896         Fricative Bookshop                          1993.12.12     MC2222    19.99    .00
8042         Bookbeat                                    1994.09.14     BU1032    19.99    5000.00
8042         Bookbeat                                    1993.05.22     PC1035    22.95    7000.00
(9 row(s) affected)
```

```
EXEC store_information 15.00
```

7. Create an **INSERT** trigger attached to the SALES table called **tr_insert_ytd**. The trigger should add the quantity inserted into the SALES table to the ytd sales column in the TITLES table (Hint: use UPDATE). Use the following code to test your trigger and query the TITLES table before and after to ensure that the ytd sales has, in fact, been increased by 5 for title ID ‘PS7777’.

```
INSERT sales
VALUES ('7131', 'Q789', 'Mar 1 2007', 5, 'Net 30', 'PS7777' )
```

```
CREATE TRIGGER tr_insert_ytd ON sales
FOR INSERT
AS
DECLARE       @qty smallint
DECLARE       @title_id varchar(6)
SELECT        @qty = qty,
              @title_id = title_id
FROM          inserted
UPDATE        titles
SET           ytd_sales = (ytd_sales + @qty)
WHERE         title_id = @title_id
GO
```

8. Create a stored procedure called **pr_author_states** which displays the first name, last name, address, and city from the AUTHORS table. The name should be in the format of first name followed by a space followed by the last name. The stored procedure will have **one input parameter** to indicate the state to be selected. If the state is not entered, display a message indicating that a value is required. Use the following code to test your stored procedure to produce the result set listed below.

```
EXECUTE pr_author_states 'KS'
```

```
AuthorID        Name                Address               City
--------------- ------------------- --------------------- --------------
341-22-1782     Meander Smith       10 Mississippi Dr     Lawrence
(1 row(s) affected) 4
```

SQL Exercise 4 – Jason Chan

```
CREATE PROCEDURE pr_author_states
( @state char(2) = NULL )
AS
IF @state IS NULL
       BEGIN
              PRINT 'Enter valid state'
       END
ELSE
       BEGIN
              SELECT au_id                           AS AuthorID,
                     (au_fname + ' ' + au_lname)       AS Name,
                     address                         AS Address,
                     city                            AS City
              FROM authors
              WHERE state = @state
END
GO
```

9. Change the **pr_author_states** stored procedure by using the ALTER command to add the state and zip code from the AUTHORS table. Rerun your stored procedure to produce the result set listed below.

```
EXECUTE pr_author_states 'KS'

AuthorID         Name               Address              City            State    Zip
---------------  -------------------  ---------------------  --------------  ------   ------
341-22-1782      Meander Smith      10 Mississippi Dr    Lawrence        KS       66044
(1 row(s) affected)
```

```
ALTER PROCEDURE pr_author_states
( @state char(2) = NULL )
AS
IF @state IS NULL
       BEGIN
              PRINT 'Enter valid state'
       END
ELSE
       BEGIN
              SELECT au_id                           AS AuthorID,
                     (au_fname + ' ' + au_lname)       AS Name,
                     address                         AS Address,
                     city                            AS City,
                     state                           AS State,
                     zip                             AS Zip
              FROM authors
              WHERE state = @state
       END
GO
```

10. Delete the **pr_author_states** stored procedure.

```
DROP PROCEDURE pr_author_states
```