Function

ฟังก์ชั่นสำหรับเปลี่ยนชนิดข้อมูล

- CAST(นิพจน์เริ่มต้น AS ชนิดข้อมูลที่ต้องการ)
  เปลี่ยนได้หลายชนิดข้อมูล **Ex**. CAST(1234 AS varchar(10))
- STR(นิพจน์ตัวเลข,ความยาวที่ต้องการ,จำนวนหลังทศนิยม)
  เปลี่ยนตัวตัวเลข,ทศนิยมเป็นตัวอักษร **Ex**. STR(234.23,5,2)

ฟังก์ชั่นที่ทำงานเกี่ยวกับวันที่

- GETDATE() แสดงวันที่และเวลาปัจจุบัน
- DAY(DateTime), MONTH(DateTime), Year(DateTime)
  แสดงค่าวัน (วันที่) เดือน (ลำดับเดือน) และปี (ค.ศ.) ของวันที่
- DATENAME(Datepart, DateTime)
  แสดงชื่อของส่วนหนึ่งของวันที่กำหนด **Ex**. DATENAME(Month, Birthday)
- DATEDIFF(Datepart, DateTime1, DateTime2)
  แสดงช่วงเวลาที่ห่างกันจากวันที่ทั้งสอง
  Ex. DATEDIFF(Year, Birthday, GETDATE())

Function

ฟังก์ชั่นที่ใช้คำนวณทางคณิตศาสตร์

- ABS(numeric_expr)
  แสดงค่าสัมบูรณ์ของค่าที่เราส่งไป **Ex**. ABS(-24)
- CEILING (numeric_expr)
  แสดงค่าตัวเลขจำนวนเต็ม (ปัดทศนิยมขึ้น)
- FLOOR(numeric_expr)
  แสดงค่าตัวเลขจำนวนเต็ม (ปัดทศนิยมลง)

ฟังก์ชั่นที่เกี่ยวกับสตริง

- CHARINDEX(expression1, expression2)
  แสดงค่าตำแหน่งเริ่มต้นของข้อความย่อยจากข้อความที่กำหนด
  Ex. CHARINDEX('pa','Lampang')
- LEFT(characer_expression, Integer_expression)
  แสดงค่ากลุ่มตัวอักษรทางซ้ายของนิพจน์ตามจำนวนที่กำหนด

Function

ฟังก์ชั่นที่เกี่ยวกับสตริง (ต่อ)

- LEN(character_expression)
  แสดงขนาดของตัวอักษรนิพจน์
- LOWER(character_expression)
  แสดงตัวอักษรพิมพ์เล็ก
- LTRIM(character_expression)
  แสดงตัวอักษรโดยตัดช่องว่างด้านซ้าย
- RIGHT(character_expression)
  แสดงค่ากลุ่มข้อมูลทางด้านขวาตามจำนวนที่กำหนด
- RTRIM(character_expression)
  แสดงตัวอักษรโดยตัดช่องว่างด้านขวา
- SPACE(integer_expression)
  แสดงอักษรเคาะวรรคตามที่กำหนด

Function

ฟังก์ชั่นในกลุ่มคำนวณชุดข้อมูล (ถูกใช้ใน SELECT)

- AVG(expression)
  แสดงค่าเฉลี่ยของนิพจน์
- COUNT(expression)
  แสดงจำนวนของนิพจน์
- MAX(expression)
  แสดงค่ามากสุดของนิพจน์
- MIN(expression)
  แสดงค่าน้อยสุดของนิพจน์
- SUM(expression)
  แสดงผลรวมของนิพจน์
- STDEV(expression)
  แสดงค่าส่วนเบี่ยงเบนมาตรฐานของนิพจน์

พื้นฐานของคำสั่ง

- การประกาศตัวแปร
  DECLARE @ชื่อตัวแปร ชนิดตัวแปร [,...]
- คำสั่งกำหนดค่าให้กับตัวแปร
  SET ตัวแปร = expression
  SELECT ตัวแปร = column FROM ...
- คำสั่งกำหนดบล๊อกชุดคำสั่ง
  BEGIN...END
- คำสั่งแสดงผลทางจอภาพ
  PRINT 'ข้อความ'
- คำสั่งตรวจสอบค่า Null และเปลี่ยนค่าใหม่
  ISNULL(expr1, expr2)

พื้นฐานของคำสั่ง

- **คำสั่งตรวจสอบเงื่อนไข IF**

  IF (เงื่อนไข)

  คำสั่งเมื่อเงื่อนไขเป็นจริง

  ELSE

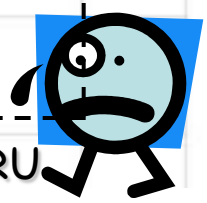
  คำสั่งเมื่อเงื่อนไขเป็นเท็จ

```
Ex. IF Score > 60
        PRINT 'คุณสอบผ่าน'
    ELSE
        PRINT 'คุณสอบไม่ผ่าน'
```

IF EXISTS()

คำสั่งเมื่อเงื่อนไขการตรวจสอบการมีอยู่เป็นจริง

```
Ex. IF EXISTS(SELECT * FROM TStudent
            WHERE Std_id  = '50122660101')
            PRINT 'พบข้อมูลคุณไม่สามารถบันทึกข้อมูลได้'
        ELSE
            PRINT 'คุณสามารถบันทึกข้อมูลได้'
```

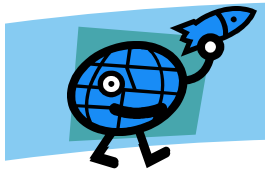


พื้นฐานของคำสั่ง

- **คำสั่งตรวจสอบเงื่อนไข CASE**

  CASE WHEN เงื่อนไข THEN คำสั่งเมื่อเงื่อนไขเป็นจริง
  WHEN เงื่อนไข THEN คำสั่งเมื่อเงื่อนไขเป็นจริง
  ELSE คำสั่งเมื่อไม่มีเงื่อนไขใดเป็นจริง
  END

```
Ex.   SELECT Std_id, Std_Fname_Th, Std_Lname_Th,
      CASE   WHEN Gender = '1' THEN PRINT 'ผู้ชาย'
             WHEN Gender = '2' THEN PRINT 'ผู้หญิง' END
      FROM TStudent WHERE Admit_year = '2550'
```

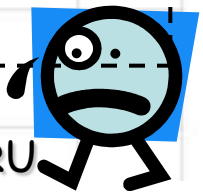


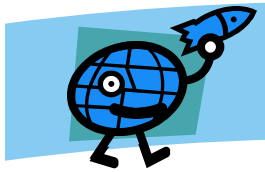
พื้นฐานของคำสั่ง

❑ คำสั่งตรวจสอบเงื่อนไข CASE

```
SELECT Prog_id, CASE WHEN Gender = '1' THEN  COUNT(Gender) END
            AS Male, CASE WHEN Gender = '2' THEN COUNT(Gender) END
            AS Female
FROM TStudent WHERE Admit_year = '2550' and Gender IS NOT NULL
GROUP BY Prog_id, Gender
ORDER BY Prog_id
```

```
SELECT *
FROM (  SELECT Prog_id, Gender FROM TStudent
            WHERE Admit_year = '2550') A
PIVOT
(
            COUNT(Gender) FOR Gender IN ( [1] , [2] )) P
ORDER BY Prog_id
```





พื้นฐานของคำสั่ง

- **คำสั่งตรวจสอบเงื่อนไข COALESCE / NULLIF**

คำสั่งใช้แทน CASE กรณีที่ตรวจสอบค่า NULL

```
SELECT CASE WHEN Color IS NOT NULL THEN color
            WHEN Color IS NULL THEN 'not specified'
       END
```

สามารถเขียนได้ด้วยคำสั่ง

```
SELECT COALESCE(Color, 'not specified')
```

หรือ

```
SELECT CASE WHEN color = 'Black' THEN Null
            ELSE color END
```

สามารถเขียนได้ด้วยคำสั่ง

```
SELECT NULLIF(color,'Black')
```

พื้นฐานของคำสั่ง

- คำสั่งทำงานวนซ้ำ

  WHILE เงื่อนไข

  คำสั่งเมื่อเงื่อนไขเป็นจริง

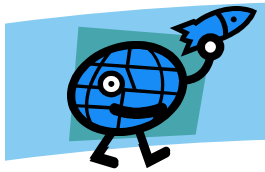
```
Ex.     DECLARE @x AS int =0;
        WHILE @x <3
        Begin
                PRINT 'value of x = '+Str(@x);
                SET @x = @x + 1;
        End;
```

  CONTINUE

  คำสั่งที่ใช้ร่วมกับ WHILE โดยจะดำเนินการวนซ้ำทันที

  BREAK

  คำสั่งที่ใช้ร่วมกับ WHILE โดยจะยุดการวนซ้ำทันที



พื้นฐานของคำสั่ง

- คำสั่งทำงานวนซ้ำ

```
1  DECLARE  @i int = 1;
2  WHILE  @i <= 10
3  BEGIN
4        PRINT @i :
5        SET @i =  @i+1;
6        CONTINUE;
7        PRINT 'ABC';
8  END
9  PRINT 'DEF';
10 GO
```

1 ➡ ผลลัพธ์ ?

```
1  DECLARE  @i int = 1;
2  WHILE  @i <= 10
3  BEGIN
4        PRINT @i ;
5        SET @i = @i / 1;
6        BREAK;
7        PRINT 'ABC';
8  END
9  PRINT 'DEF';
10 GO
```

ผลลัพธ์ ? ⬅ 2

พื้นฐานของคำสั่ง

- คำสั่ง TRY/CATCH

  คำสั่งที่ควบคุมการเกิด run time error ที่อาจไม่ได้เกิดจากบั๊ก แต่เกิดจากสถานการณ์หรือข้อบังคับ

```
BEGIN TRY
        < คำสั่งบรรทัด 1 >
        < คำสั่งบรรทัด n >
END TRY
BEGIN CATCH
        < คำสั่งที่รับมือกับ error 1 >
        < คำสั่งที่รับมือกับ error 2 >
END CATCH
```

- แสดงหมายเลข error
- แสดงสถานะ error
- แสดงบรรทัดที่เกิด error
- แสดงชื่อ procedure หรือ trigger ที่เกิด error
- แสดงข้อความ error

ค่า Error ที่สามารถเรียกแสดงได้คือ
ERROR_NUMBER(), ERROR _STATE(), ERROR_LINE(), ERROR_PROCEDURE(), ERROR_MESSAGE()

พื้นฐานของคำสั่ง

- TRY..CATCH

Ex.

```
BEGIN TRY
        SELECT 1/0;
END TRY
BEGIN CATCH
        SELECT ERROR_NUMBER() AS Error1,
               ERROR_MESSAGE() AS Error2
END CATCH
```

	Error1	Error2
1	8134	Divide by zero error encountered.

การทำคิวรี เพื่อรายงาน (ROLLUP/CUBE/PIVOT)
คิวรีระดับสูง

การจัดทำรายงานสรุปต้องอาศัยจัดกลุ่มแถวข้อมูลด้วยคำสั่ง GROUP BY ซึ่งสามารถใช้คำสั่งเพิ่มขยายด้วยคำสั่ง ROLLUP และ CUBE ได้ เช่น

ตัวอย่างเช่น

- ต้องการทราบว่านักศึกษารหัส 50122660199 แต่ละภาคเรียนได้เกรดอะไรบ้าง

```
SELECT      id_no, term, tgrade, COUNT(Tgrade)
FROM        Tgrade
WHERE       id_no like '50122660199'
GROUP BY id_no, term, tgrade
ORDER BY id_no, term
```

❑ GROUP BY ROLLUP()

```
...
GROUP BY ROLLUP (id_no, term, tgrade)
```

กลุ่มข้อมูลที่เกิดจะเท่ากับใช้คำสั่ง GROUP BY ปกติ แต่จำนวนแถวจะปรากฏค่าที่เป็นไปได้ทั้งหมด

การทำคิวรี เพื่อรายงาน (ROLLUP/CUBE/PIVOT)
คิวรีระดับสูง

❑ GROUP BY CUBE ()

...

GROUP BY CUBE (id_no, term, tgrade)

กลุ่มข้อมูลที่เกิดจะเท่ากับใช้คำสั่ง GROUP BY ปกติ แต่จำนวนแถวจะแบ่งการจัดกลุ่มเป็นหลายระดับ ซึ่งต่างจาก GROUP BY จะทำให้จำนวนแถวน้อยลง แต่คำสั่ง CUBE กลับตรงกันข้าม เพราะจะสรุปเป็นจำนวนกลุ่มต่าง ๆ จำนวนมาก คือครบทุกกลุ่มและทุกค่าที่เป็นไปได้

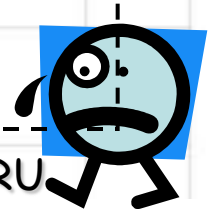
โดยสรุป...ในการทำรายงานหากต้องการจับกลุ่มข้อมูลอย่างพลิกแพลง

คำสั่ง GROUP BY จะทำไม่ได้ ต้องใช้คำสั่ง ROLLUP() และ CUBE()

หมายเหตุ...

หากรันคิวรีแล้วเกิดข้อผิดพลาด ให้กำหนด Allow mode ดังนี้

EXEC sp_dbcmptlevel ชื่อฐานข้อมูล, 100;



การทำคิวรี เพื่อรายงาน (ROLLUP/CUBE/PIVOT)
คิวรีระดับสูง

❑ PIVOT

คือคำสั่งทำหน้าที่นำข้อมูลมาทำเป็นคอลัมน์ผลลัพธ์ โดยเปลี่ยนข้อมูลสองมิติให้เป็นสามมิติ

```
SELECT *
FROM
        (SELECT id_no, term, tgrade
        FROM TGrade
        WHERE id_no = '50122660106') A
PIVOT
(       COUNT(tgrade)
        FOR Tgrade IN ([A],[B+],[B],[C+],[C],[D+],[D],[E])) P
ORDER BY id_no, Term
```