การสร้างรายงานด้วย Report Designer ขั้นประยุกต์

ระหว่างวันที่ ๑๓ – ๑๗ มิถุนายน ๒๕๕๔

จัดโดย

บ.บางกอก เมดิคอล ซอฟแวร์ จำกัด ๒ ชั้น ๒ ม.๘ ซ.สุขสวัสดิ์ ๓๓
แขวง/เขต ราษฎร์บูรณะ กรุงเทพฯ

เรียบเรียงโดย
นายสุชาติ จันต๊ะวงศ์ นักวิชาการสาธารณสุขชำนาญการ
สำนักงานสาธารณสุขอำเภอเชียงกลาง

ตารางการอบรม

วันที่ ๑

- คำสั่ง SQL สำหรับการจัดการข้อมูลและเลือกข้อมูลแบบมีเงื่อนไข
- การประยุกต์ใช้คำสั่ง SQL เพื่อการทำรายงาน
- แบบฝึกหัด

วันที่ ๒

- ภาษา Pascal ขั้นพื้นฐาน
- ภาษา Pascal เพื่อการประยุกต์ใช้ในการทำรายงาน
- แบบฝึกหัด

วันที่ ๓

- การใช้ Variable ในการสร้างรายงาน
- การใช้ variable ช่วยในการสร้างรายงานที่ซับซ้อมขึ้น
- แบบฝึกหัด

วันที่ ๔

- การใช้ Function ในการสร้างรายงาน
- การใช้ Function ช่วยในการสร้างรายงานแบบมีเงื่อนไข
- แบบฝึกหัด

วันที่ ๕

- การสร้างรายงานในรูปแบบต่างๆ
- แบบฝึกหัด

SQL

โครงการอบรมผู้ดูแลระบบ Report Designer

---

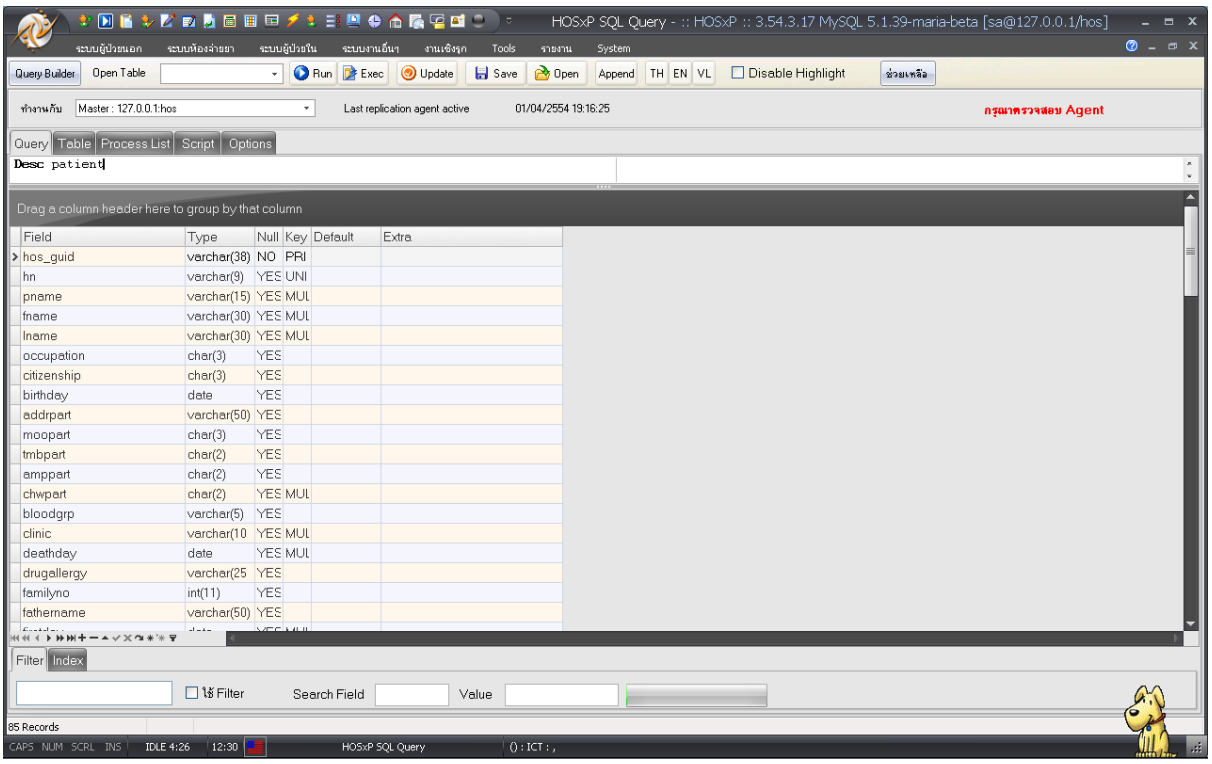
๑. การดูโครงสร้างตารางด้วยคำสั่ง DESC หรือ DESCRIBE

รูปแบบคำสั่ง

DESC [table name] หรือ DESCRIBE [table name]

ตัวอย่าง

Desc patient



๒. การสอบถามข้อมูลพื้นฐาน (SELECT)

รูปแบบคำสั่ง

SELECT Column๑,Column๒,Column๓,... FROM [Table Name]

ตัวอย่าง

SELECT * FROM patient

SELECT hn,pname,fname,lname,cid FROM patient

๓. คำสั่งเลือกข้อมูลมาแสดงตาม record ที่ต้องการ LIMIT

รูปแบบคำสั่ง

SELECT Column๑,Column๒,Column๓,... FROM [Table Name]

LIMIT [int-start],[int-end]

ตัวอย่าง

SELECT hn,pname,fname,lname,cid FROM patient

LIMIT ๑๐๐

๔. คำสั่งรวมข้อความด้วย SELECT ... CONCAT

รูปแบบคำสั่ง

SELECT CONCAT (Column๑, Column๒,...)

ตัวอย่างคำสั่ง

SELECT hn,CONCAT(pname,fname," ",lname),cid

FROM patient

๕. คำสั่งเปลี่ยนหัวคอลัมน์เป็นข้อความอื่นด้วย SELECT ... AS

รูปแบบคำสั่ง

SELECT <Column> AS "New Heading"

ตัวอย่าง

SELECT hn,CONCAT(pname,fname," ",lname) AS name ,cid

FROM patient

๖. คำสั่งเลือกเฉพาะข้อมูลที่ต้องการด้วย WHERE

รูปแบบคำสั่ง

SELECT <Column-name> FROM <Table-name>

WHERE <condition>

ตัวอย่าง

SELECT * FROM patient

WHERE nationality = "๙๐"

๗. คำสั่งเลือกเฉพาะข้อมูลที่ต้องการ หลายเงื่อนไข (OR, AND)

รูปแบบคำสั่ง

SELECT <Column-name> FROM <Table-name>

WHERE <condition> AND / OR <condition>

ตัวอย่าง

SELECT * FROM patient WHERE nationality = “๙๙”

AND occupation = “๑๑๒”

SELECT * FROM patient WHERE pttype <> “๒๐”

๘. คำสั่งเลือกข้อมูลที่ตรงตามชุดข้อมูลด้วย IN,NOT IN

รูปแบบคำสั่ง

SELECT <Column-name> FROM <Table-name>

WHERE <Column-name> [NOT] IN (data set)

ตัวอย่าง

SELECT * FROM patient

WHERE pttype in (“๒๐”,”๒๑”,”๕๙”)

SELECT * FROM patient

WHERE pttype not in (“๒๐”,”๒๑”,”๕๙”)

๙. คำสั่งเลือกข้อมูลตามเงื่อนไข like

รูปแบบคำสั่ง

SELECT <Column-name> FROM <Table-name>

WHERE <Column-name> like “%<name>%”

ตัวอย่าง

SELECT fname FROME patient WHERE like “%พร”

SELECT fname FROME patient WHERE like “พร%”

SELECT fname FROME patient WHERE like “%พร%”

๑๐. คำสั่งหาค่าว่างหรือไม่ว่าง (Is null, Is not null, “ “)

รูปแบบคำสั่ง

SELECT <Column-name> FROM <Table-name>

WHERE <Column-name> = “ “

SELECT <Column-name> FROM <Table-name>

WHERE <Column-name> IS NULL, IS NOT NULL

ตัวอย่าง

SELECT * FROM vn_stat

WHERE pdx IS NOT NULL or pdx <> or pdx =””

๑๑. คำสั่งเงื่อนไขเพื่อเลือกช่วงข้อมูล BETWEEN .. AND

รูปแบบคำสั่ง

SELECT <Column-name> FROM <table-name>

WHERE <Column-name> BETWEEN <value๑> AND <value๒>

ตัวอย่าง

SELECT * FROM vn_stat

WHERE vstdate BETWEEN “๒๐๑๑-๐๓-๐๑” AND “๒๐๑๑-๐๓-๓๑”

๑๒. คำสั่งฟังก์ชันที่ใช้ในการตัดค่าที่ซ้ำ distinct

รูปแบบคำสั่ง

SELECT distinct (Column-name) FROM <Table-name>

ตัวอย่าง

SELECT count (distinct (hn)) AS con FROM va_stat

WHERE vstdate BETWEEN “๒๐๑๑-๐๓-๐๑” AND “๒๐๑๑-๐๓-๓๑”

๑๓. คำสั่งตัดข้อความที่ต้องการด้วย SUBSRING

รูปแบบคำสั่ง

SELECT SUBSTR (Column-name,pos,len) AS “New-Field”

ตัวอย่าง

```
SELECT SUBSTR (vstdate, ๑,๔) AS “Year”
FROM vn_stat
```

๑๔. คำสั่งการจัดกลุ่มข้อมูลด้วยคำสั่ง GROUP BY

รูปแบบคำสั่ง

```
SELECT <Column-name> FROM <Table-name>
GROUP BY <Column-name>
```

ตัวอย่าง

```
SELECT vn.hn,pdx,vstdate FROM vn_stat
WHERE vstdate BETWEEN “๒๐๑๑-๐๓-๐๑” AND “๒๐๑๑-๐๓-๓๑”
GROUP BY pdx
```

๑๕. คำสั่งจัดเรียงข้อมูลด้วยคำสั่ง ORDER BY

รูปแบบคำสั่ง

```
SELECT <Column-name> FROM <Table-name>
ORDER BY <Column-name> [DESC][ASC]
```

ตัวอย่าง

```
SELECT vn,hn,pdx,vstdate FROM vn_stat
WHERE vstdate BETWEEN “๒๐๑๑-๐๓-๐๑” AND “๒๐๑๑-๐๓-๓๑”
GROUP BY pdx
ORDER BY pdx DESC
```

๑๖. คำสั่งเงื่อนไขของกลุ่มด้วย HAVING

รูปแบบคำสั่ง

```
SELECT <Column-name> FROM <Table-name>
WHERE <condition>
GROUP BY <Column-name>
HAVING <condition>
```

ตัวอย่าง

```
SELECT pttype,sum(income) AS income FROM vn_stat
WHERE pttype IN("๗๐","๗๑","๗๗")
GROUP BY pttype
HAVING income < ๕๐๐๐๐
ORDER BY pttype
```

๑๗. คำสั่ง SUB QUERY

รูปแบบคำสั่ง

```
SELECT Column๑,Column๒,Column๓, ...
FROM [Table-name]
WHERE <condition> (SELECT Column FROM [Table-name]
```

ตัวอย่าง

```
SELECT * FROM opitemrece
WHERE icode IN(DELECT icode FROM drugitems)
LIMIT ๑๐๐
```

๑๘. คำสั่งฟังก์ชันที่ใช้หาค่าข้อมูลทั้งกลุ่ม

รูปแบบคำสั่ง

```
SELECT Function <Column-name>
FROM <Table-name>
```

AVG : ให้ผลลัพธ์เป็นค่าเฉลี่ยของข้อมูลทั้งกลุ่ม
SUM : ให้ผลลัพธ์เป็นผลบวกของข้อมูลทั้งกลุ่ม
MIN : ให้ผลลัพธ์เป็นค่าน้อยที่สุดของข้อมูลทั้งกลุ่ม
MAX : ให้ผลลัพธ์เป็นค่ามากที่สุดของข้อมูลทั้งกลุ่ม
COUNT : ให้ผลลัพธ์เป็นจำนวนรายการของข้อมูลทั้งกลุ่ม

๑๙. เชื่อมTable แบบ EQUI JION

รูปแบบคำสั่ง

SELECT <Column-name> FROM <Table-name><Nickname>,<Table-join><Nickname>

WHERE <Nickname><key> = <Nickname-Table-join><key>

ตัวอย่าง

SELECT o.icode,CONCAT(d>name," ",d.strength),d.unitprice,o.qty

FROM poitemrece o, drugitems d

WHERE o.icode = d.icode

AND o.rxdate BETWEEN "๒๐๑๑-๐๓-๐๑" AND "๒๐๑๑-๐๓-๓๑"

๒๐. เชื่อมTable แบบ LEFT OUTER JOIN

รูปแบบคำสั่ง

SELECT <Column-name>

FROM <Table-name><Nickname> *** เฉพาะตารางหลัก

LEFT OUTER JOIN <Table-join><Nickname>

ON <Nickname><key> = <Nickname-join><key>

ตัวอย่าง

SELECT o.icode,CONCAT(d>name," ",d.strength), d.unitprice,o.qty

FROM poitemrece o

LEFT OUTER JOIN drugitems d ON o.icode = d.icode

WHERE o.rxdate BETWEEN "๒๐๑๑-๐๓-๐๑" AND "๒๐๑๑-๐๓-๓๑"

๒๑. เชื่อม Table แบบ Self Join

รูปแบบคำสั่ง

SELECT <Column-name>

FROM <Table-name><Nickname๑>, <Table-name><Nickname๒>

WHERE <Nickname๑><key> = <Nickname๒><key>

ตัวอย่าง

SELECT p๑.hn,p๑.fname,p๑.lname,p๑.birthday,p๒.hn,p๒.fname,p๒.lname

FROM p๑.lname = p๒.lname AND p๑.hn <> p๒.hn

รูปแบบของโปรแกรม Pascal

Program

Var

Begin

End.

๑. ส่วนหัวของโปรแกรม : Program Heading

ลักษณะที่ใช้

ใช้ในการตั้งชื่อของโปรแกรม

รูปแบบ

Program ชื่อโปรแกรม;

ตัวอย่าง

MyFirstProgram;

๒. ส่วนหัวการประกาศ : Declaration Part

ลักษณะที่ใช้

๑. ประกาศตัวแปรที่ใช้ในโปรแกรม (Variable Declaration)
รูปแบบ Var ชื่อตัวแปร : ชนิดตัวแปร;
ตัวอย่าง Var date๑,date๒ : datetme; ds๑,ds๒ : string;

๒. ประกาศค่าคงที่ (Constant Declaration)
รูปแบบ Const ชื่อค่าคงที่ = ค่าที่กำหนด;
ตัวอย่าง Const Addon = ๘๐;

๓. ประกาศชนิดของข้อมูลที่สร้างขึ้นใหม่ (Type Declaration)
ตัวอย่าง Type Color = (red,green,blue);

๔. ประกาศตัวแปรย่อยในรูปของ Procedure และ Function

๓. ส่วนคำสั่ง : Statement

ลักษณะที่ใช้

ใช้ในการเขียนคำสั่งประมวลผล

รูปแบบ

```
Begin
    Statement_๑
    Statement_๒
    ..........
    Statement_n
End.
```

ตัวอย่าง

```
Program     MyFirstProgram
Begin
            ShowMessage('This is a Pascal Program');
End.
```

```
Program     MyFirstProgram
Var         IntegerVar : Integer;
Begin
            IntegerVar := ๓;
            ShowMessage(IntToStr(IntegerVar));
End.
```

หลักการตั้งชื่อตัวแปรในภาษาปาสคาล

๑. ต้องขึ้นต้นด้วยอักษร A-Z หรือ a-z หรือ เครื่องหมาย _ (underscore)

๒. ภายในชื่อสามารถใช้ได้แต่ตัวอักษร A-Z หรือ a-z หรือตัวเลข ๐-๙ หรือเครื่องหมาย _ เท่านั้น

๓. การใช้ตัวอักษรพิมพ์ใหญ่ หรือพิมพ์เล็ก ไม่มีผล คือมีค่าไม่ต่างกัน เช่น Name ก็มีความหมายเหมือนกับ NAME หรือ name เป็นต้น

๔. ห้ามนำคำสงวนมาตั้งชื่อ

๕. ห้ามเว้นช่องว่างในชื่อ

ตัวอย่างการตั้งชื่อตัวแปรที่ถูกต้อง

- Average;
- Lab๘;
- _sum;
- SUM_income;

ตัวอย่างการตั้งชื่อตัวแปรที่ไม่ถูกต้อง

- ๙SUM;
- Begin;
- Sum@A
- _Data-๒

ตัวอย่างคำสงวน

And	downto	in	or	then
Asm	else	inline	packed	to
Array	end	interface	procedure	type
Begin	exports	label	program	unit
Case	file	library	record	until
Const	for	mod	repeat	uses
Constructor	function	nil	set	var
Destructor	goto	not	shl	while
Div	if	object	shr	with
Do	implementation	of	string	xor

รูปแบบของการเขียนโปรแกรม

๑. ชื่อ คำ หรือตัวเลข ต้องแยกกันโดยใช้ช่องว่าง อย่างน้อย ๑ ช่อง หรือแยกกันโดยการขึ้นบรรทัดใหม่ หรือด้วยสัญลักษณ์ : (colon) ; (semi-colon) หรือ , (comma)

๒. จะแบ่งข้อความที่อยู่ในเครื่องหมาย ‘’ ให้อยู่คนละบรรทัดไม่ได้

๓. การตั้งชื่อตัวแปรควรตั้งให้สื่อความหมาย

๔. การย่อหน้าให้ตรงกัน จะช่วยทำให้โปรแกรมอ่านง่าย

๕. การใส่ comment จะใช้เครื่องหมาย // ด้านหน้า หรือให้อยู่ในเครื่องหมาย { } หรือ (* *)

ชนิดของข้อมูล

๑. Integer เก็บข้อมูลแบบตัวเลขจำนวนเต็ม มีค่าระหว่าง –๓๒๗๖๘ ถึง ๓๒๗๙๗
๒. Real เก็บข้อมูลแบบตัวเลขจำนวนจริง ที่ประกอบด้วยตัวเลขจำนวนเต็มและทศนิยม
๓. Char เก็บข้อมูลเป็นตัวอักขระ หรือตัวอักษร เพียง ๑ ตัว อาจจะเป็นตัวเลข, ตัวอักษร, สัญลักษณ์พิเศษ
๔. String เก็บข้อมูลที่เป็นข้อความ
๕. Boolean เก็บข้อมูลที่เป็นแบบตรรกศาสตร์ ที่แสดงถึงการตัดสินใจว่าข้อมูลนั้นจริง(True) หรือเท็จ(False)

การกำหนดค่าให้ตัวแปร

รูปแบบ

Variable := expression

ตัวอย่าง

```
Discount    := ๑๕;
Pay         := Salary + ๑๕๐๐;
Sum         := Sum + Num๑;
Str๑        := '๑๒๓๔๕ABCDE';
```

การประกาศค่าคงที่

```
Program         DisplayConst;
Const           Addon           = ๘๐;
                InterestRate    = ๐.๑๐;
                AccountCode             = 'CD';
Begin
        Statement;
End;
```

เครื่องหมายทางคณิตศาสตร์ (Arithmetic Operator)

๑. เครื่องหมายบวก สัญลักษณ์ +
๒. เครื่องหมายลบ สัญลักษณ์ -
๓. เครื่องหมายคูณ สัญลักษณ์ *
๔. เครื่องหมายหาร สัญลักษณ์ /
๕. เครื่องหมายหารตัดเศษ สัญลักษณ์ div
๖. เครื่องหมายหารเอาแต่เศษ สัญลักษณ์ mod
๗. ฟังก์ชันเพื่อคำนวณการยกกำลังสองของX สัญลักษณ์ sqr(x)
๘. ฟังก์ชันการหารากที่สองของx สัญลักษณ์ sqrt(x)

ลำดับการคำนวณของเครื่องหมายทางคณิตศาสตร์

๑. คำนวณนิพจน์ที่อยู่ในวงเล็บก่อน โดยเริ่มจากวงเล็บในสุด
๒. ทำการ คูณ, หาร, div, mod, and
๓. ทำการบวก, ลบ, or
๔. ถ้าลำดับเท่ากัน ทำจากซ้ายไปขวา

เครื่องหมายเปรียบเทียบ

๑. เครื่องหมายเท่ากับ สัญลักษณ์ =
๒. เครื่องหมายน้อยกว่า สัญลักษณ์ <
๓. เครื่องหมายน้อยกว่าหรือเท่ากับ สัญลักษณ์ <=
๔. เครื่องหมายมากว่า สัญลักษณ์ >
๕. เครื่องหมายมากกว่าหรือเท่ากับ สัญลักษณ์ >=
๖. เครื่องหมายไม่เท่ากับ สัญลักษณ์ <>

เครื่องหมายเปรียบเทียบทางตรรกะ

๑. Not ให้ค่าตรงข้ามกับเงื่อนไขที่ตามหลัง
๒. And ให้ค่าเป็นจริง เมื่อเงื่อนไขทั้งคู่เป็นจริง และให้ค่าเป็นเท็จเมื่อเงื่อนไขใดเงื่อนไขหนึ่งมีค่าเป็นเท็จ หรือเป็นเท็จทั้งคู่
๓. Or ให้ค่าเป็นเท็จเงื่อนไขทั้งคู่เป็นเท็จ และให้ค่าเป็นจริงเมื่อเงื่อนไขใดเงื่อนไขหนึ่งมีค่าเป็นจริง หรือเป็นจริงทั้งคู่

๔. Xor ให้ค่าเป็นจริง เมื่อเงื่อนไขใดเงื่อนไขหนึ่งเท่านั้นเป็นจริง และให้ค่าเป็นเท็จ เมื่อเงื่อนไขทั้งคู่เป็นจริง หรือเงื่อนไขทั้งคู่เป็นเท็จ

ข้อควรระวังในการลำดับการเปรียบเทียบ

๑. ให้ใส่วงเล็บคั่นระหว่างสองเงื่อนไข กับเครื่องหมายตรรกะ เช่น x > ๔๐ or x < ๐ ให้เขียนใหม่เป็น (x >๔๐) or (x < ๐)
๒. เงื่อนไข not A and not B เขียนได้เป็น not(A or B)
๓. เงื่อนไข not A or not B เขียนได้เป็น not(A and B)

โปรแกรมแบบมีเงื่อนไข

รูปแบบ

```
If condition then
        Statement๑
Else
        Statement๒;
```

ตัวอย่างการเปรียบเทียบ

```
Program  CheckNumber;
Var  A : Integer;
Begin
        A := ๑๐;
        If A >= ๐ Then
        ShowMessage('A is positive number')
Else
        ShowMessage('A is nagative number');
End.
```

คำสั่ง IF ไม่จำเป็นต้องมี ELSE ตามหลังก็ได้

ตัวอย่าง

```
If Num > ๑๐ Then Showmessage('Num>๑๐');
```

Compound Statement

รูปแบบ

```
IF Comdition๑ Then
        Begin
                .....
        End     // end ตัวแรกไม่ต้องมีเครื่องหมาย;
Else
        Begin
                .....
End;
```

Nested if

รูปแบบ

```
If Condition๑ Then
        Begin
                ......
        End   // end ตัวแรกไม่ต้องมีเครื่องหมาย;
Else If Condition๒ Then
        Begin
                ......
        End
Else
        Begin
        End;
```

คำสั่ง Case

รูปแบบ

```
Case selector Of
        Constant๑ : Statement๑;
        Constant๒ : Statement๒;
        Constant๓ : Statement๓;
        .....
        .....
        Constant_n : Statement_n;
Else
        Default_Statement
End;
```

ตัวอย่าง Case

```
Program  MyCase;
Var     DayInteger : Integer;
        DayName : String;
Begin
        DayInteger := ๑;
        Case DayInteger Of
                ๑ : DayName := ‘Monday’;
                ๒ : DayName := ‘Tuesday’;
        End;
End;
```

การทำงานแบบวนรอบ หรือทำซ้ำ (LOOP)

เป็นการเขียนคำสั่งเพื่อให้โปรแกรมวนทำงานซ้ำคำสั่งเดิม โดยการกำหนดเงื่อนไขในการวนรอบทำงานจนกว่าเงื่อนไขไม่เป็นจริง จึงจะหยุด

คำสั่งที่ใช้ในการทำงานแบบวนรอบ

๑. FOR
๒. WHILE
๓. REPEAT

โปรแกรมแบบทำซ้ำด้วยคำสั่ง FOR

รูปแบบ

```
For Control_Value := Initial_value To Final_value Do Statement;
```

ตัวอย่าง โปรแกรมแบบทำซ้ำ ด้วยคำสั่ง FOR

```
Program For_Loop;
Var Number : Integer;
Begin
    For Number := ๑ To ๑๐ Do
    Begin
    Showmessage('Number = '+IntToStr(Number));
    End;
End;
```

ตัวอย่าง คำสั่ง FOR ซ้อนกัน

```
Program For_Nested;
Var Number๑, Number๒ : Integer;
Begin
    For Number๑ := ๑ To ๑๐ Do
    Begin
        Showmessage(‘Number๑ = ‘+IntToStr(Number๑));
        For Number๒ := ๑ To ๓ Do
        Begin
        Showmessage(‘Number๒ = ‘+IntToStr(Number๒));
    End;
End;
End;
```

คำสั่ง WHILE

รูปแบบ

```
While Condition do
    Statement;
หรือ
While Condition do
Begin
    Statement๑;
    Statement๒;
    .....
    Increment/Decrement;
End;
```

ตัวอย่างคำสั่ง WHILE

```
Program Test_While;
Var I : Integer;
Begin
    I := ๑๐;
    While I > ๐ Do
    Begin
    ShowMessage(IntToStr(I));
    I := I – ๑;
    End;
End;
```

คำสั่ง REPEAT

รูปแบบคำสั่ง

```
Repeat
    Statement๑;
    Statement๒;
    .....
    Increment/Decrement;
Until Condition;
```

ตัวอย่างการใช้ คำสั่ง REPEAT

```
Program Test_Prpeat;
Var I : Integer;
Begin
    I := ๑๐;
    Repeat
        ShowMessage(IntToStr(I));
        I := I – ๑;
    Until I < ๐;
End;
```

สรุปการใช้คำสั่ง

๑. คำสั่ง FOR เหมาะกับการทำงานที่รู้จำนวนครั้งที่ต้องการให้ทำงานซ้ำ และข้อมูลที่ใช้เพื่อตรวจสอบจำนวนครั้งในการทำซ้ำ ต้องเป็นเลขจำนวนเต็ม

๒. คำสั่ง WHILE เหมะกับการทำงานที่ไม่รู้จำนวนครั้งที่แน่นอนของการทำซ้ำ โยจะตรวจสอบเงื่อนไขก่อนเข้าทำงานใน LOOP และต้องกำหนดค่าเริ่มต้นให้เงื่อนไขก่อนเข้า LOOP เสมอ

๓. คำสั่ง REPRAT เหมะกับการทำงานที่ไม่รู้จำนวนครั้งที่แน่นอนของการทำงานซ้ำ โดยจะเข้าทำงานใน LOOP ก่อนหนึ่งครั้ง แล้วจึงตรวจสอบเงื่อนไข

โปรแกรมย่อยชนิดโพรซีเยอร์ (Procedure)

Procedure เปรียบเหมือนโปรแกรมย่อยที่เขียนคำสั่งต่างๆรวมไว้ด้วยกัน เพื่อทำงานออกมาให้ตรงตามที่เราต้องการ

รูปแบบการประกาศ

```
Procedure  Procedure_Name(parameter๑,parameter๒,...,parameter_n);
Var  name : type;
Begin
        Statement_๑
        Statement_๒
        ..................
        Statement_n
End;
```

ข้อดีของการใช้โพรซีเยอร์

คือ ถ้าภายในโปรแกรมต้องทำงานที่ซ้ำซ้อนหรือซ้ำกันหลายครั้ง ยกตัวอย่างเช่น การแสดงเส้นคั่นบรรทัดด้วยเครื่องหมาย * ยาว ๕๐ คอลัมน์ คำสั่งที่ต้องเขียนให้แสดง ดังนี้

```
Program line;
Uses wincrt;
Var Count : byte;
Begin
    For Count := ๑ to ๕๐ do
        Write('*');
        writeln;
    end.
```

โปรแกรมย่อยชนิดโพซีเยอร์(Procedure)

การเรียกใช้โพรซีเยอร์

```
Procedure_Name(actual_parameter_list);
```

หรือ

```
Procedure_Name;
```

การส่งค่าพารามิเตอร์

```
Procedure_Name(actual_parameter_list);
```

หรือ

```
Procedure_Name;
```

โปรแกรมย่อยชนิดฟังก์ชัน (FUNCTION)

มีหน้าที่การทำงานเหมือนโพรซีเยอร์ โดยเปรียบเหมือนโปรแกรมย่อยทำงานตามที่กำหนด ดังนั้นโครงสร้างและหลักการที่ใช้กับโพรซีเยอร์ ก็สามารถนำมาใช้กับฟังก์ชันได้เช่นกัน ข้อแตกต่างระหว่างโพรซีเยอร์และฟังก์ชันมีเพียงข้อเดียว คือการทำงานของฟังก์ชันจะมีการส่งค่ากลับออกมาจากฟังก์ชัน ๑ ค่า

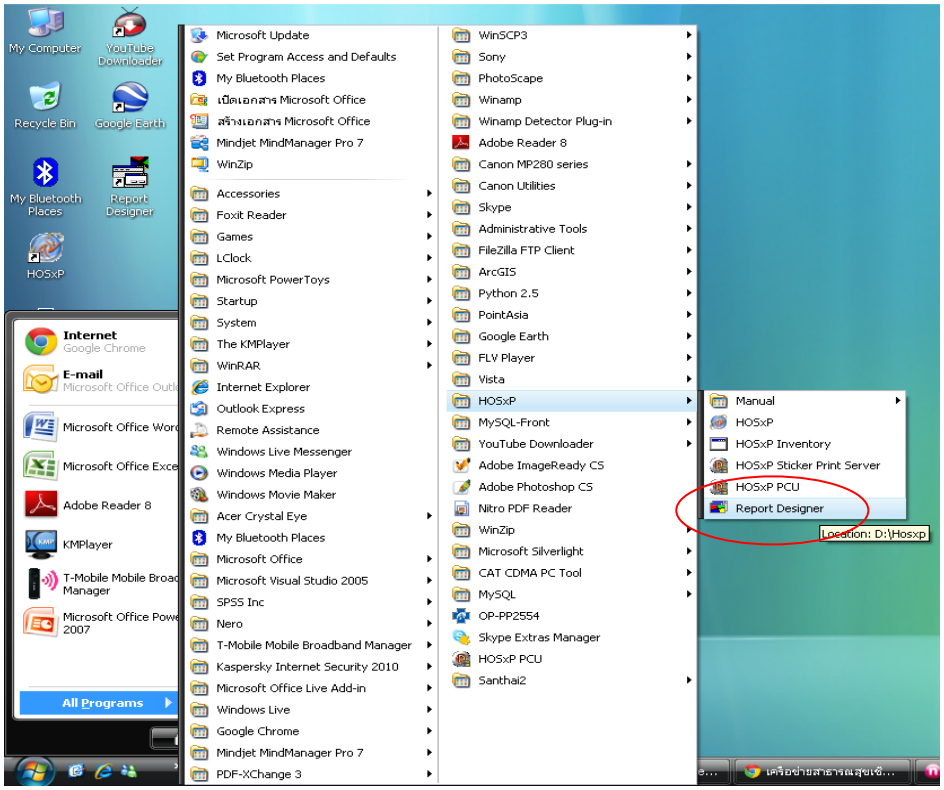
รูปแบบการประกาศ

```
Function
        Function_name(parameter_๑,parameter_๒,....,parameter_n);
        Return;
Var name : type;
Begin
        Statement_๑
        Statement_๒
        ..................
        Statement_n;
End;
```

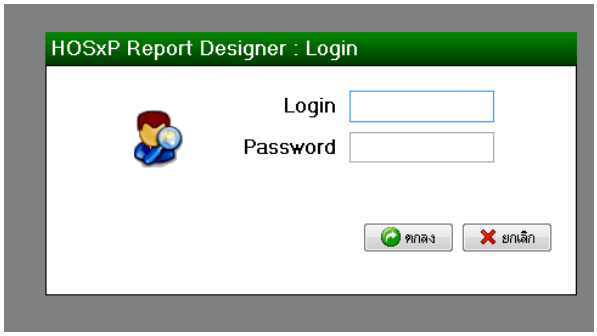
การใช้โปรแกรม HOSxP Report Designer

การใช้งาน

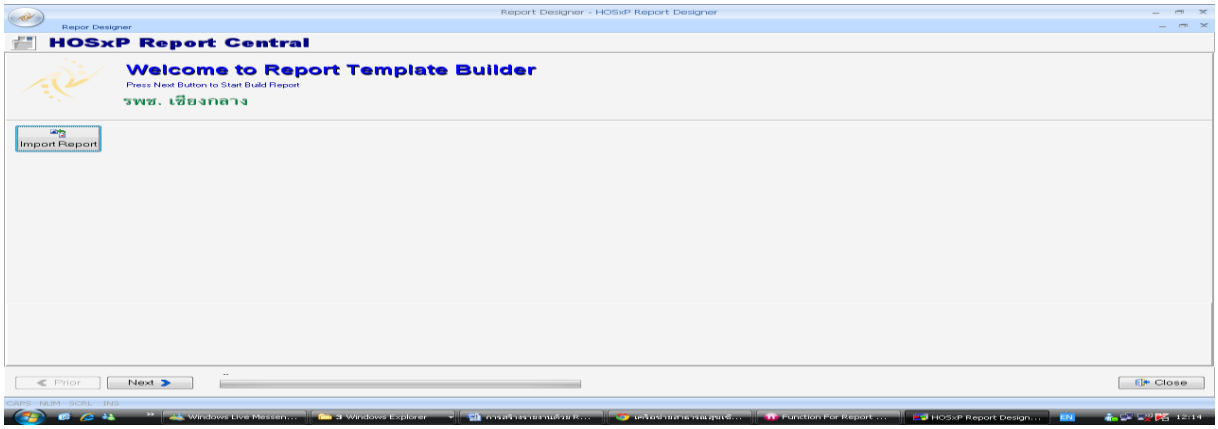
๑. เข้าโปรแกรม เลือก All Programs เลือก HOSxP แล้วเลือก Reports Designer



๒. Login และ Password เดียวกับของผู้ดูแลระบบที่เข้าโปรแกรม HOSxP

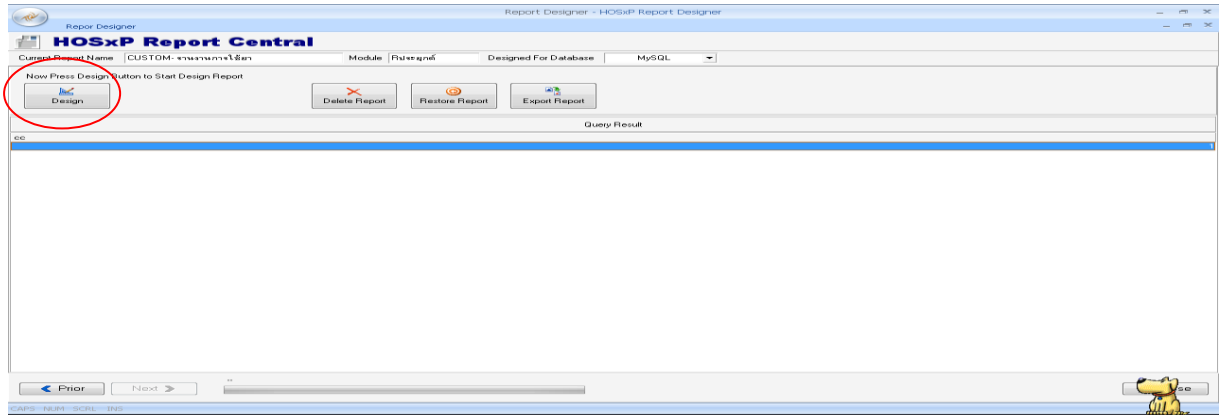


๓. หน้าตาของโปรแกรม Import Report ใช้ในการนำคำสั่งของ Report ใหม่จากที่อื่นเข้าสู่โปรแกรม

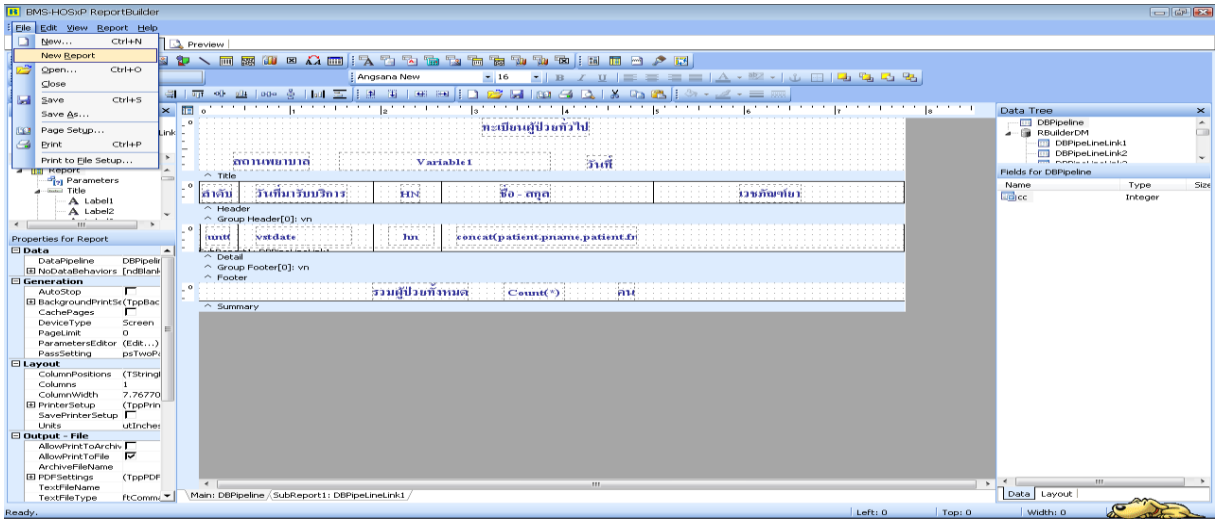


การสร้าง Report Designer ใหม่

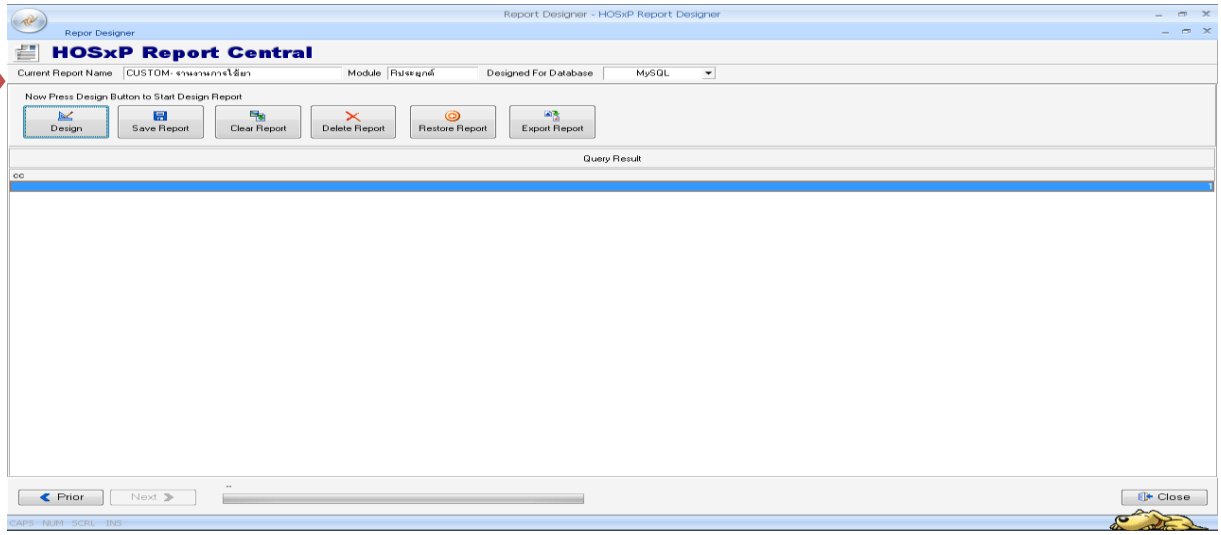
๑. ในหน้าจอของ HOSxP Report Central คลิก Next เลือก Custom Report เลือกคำสั่ง Report หนึ่งคำสั่งเป็นชื่ออะไรก็ได้ คลิกเลือก Next คลิก Next อีกครั้ง



๒. คลิกเลือก Design

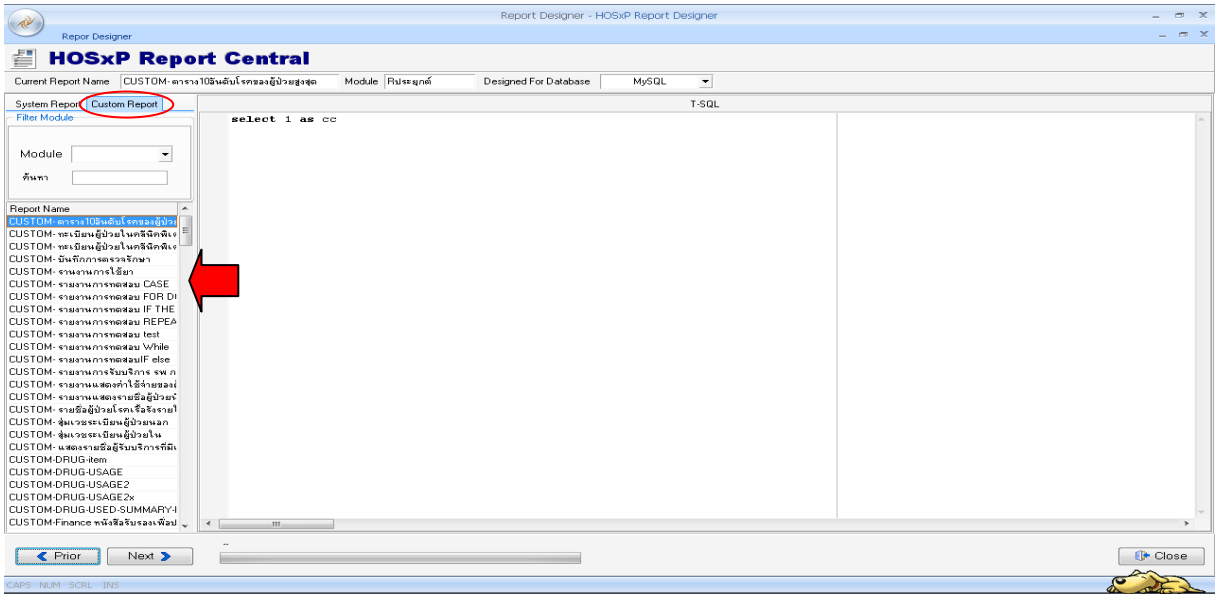


๓. ไปที่ File เลือก New Report และลบข้อมูลเดิมออกให้หมด คลิกปิดหน้า Report



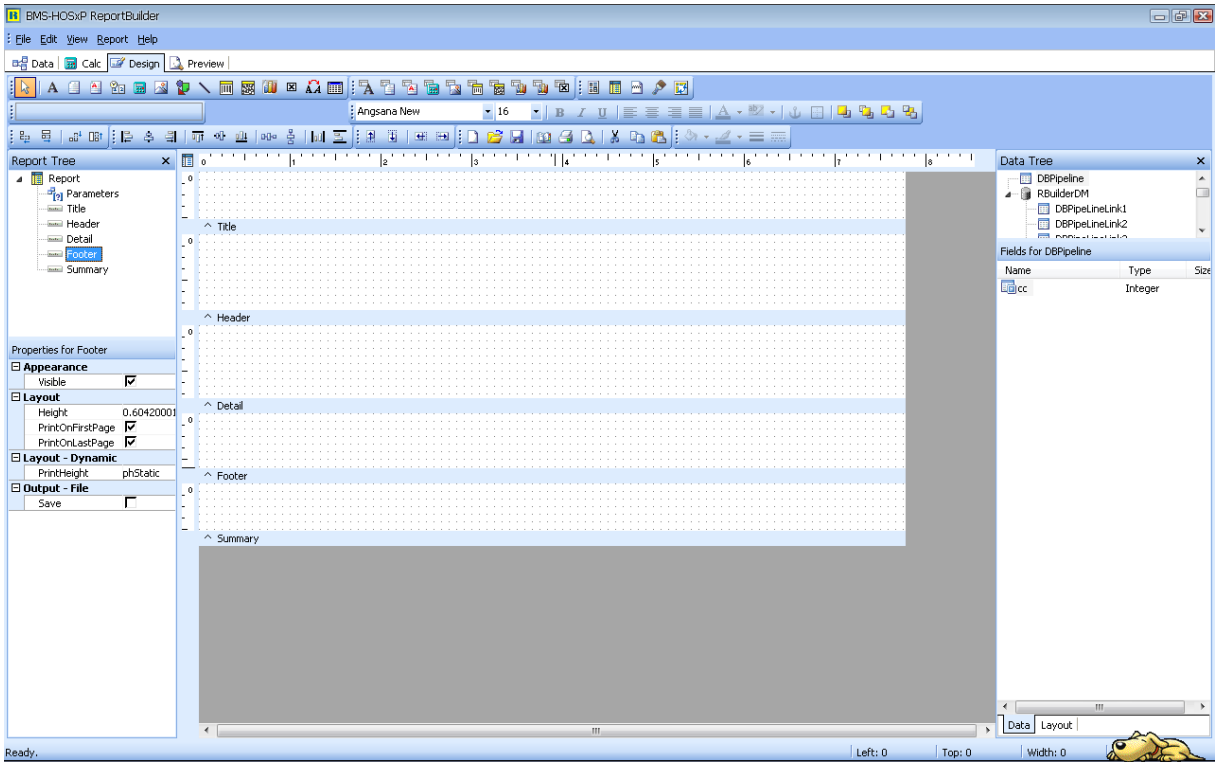
๔. ที่ Current Report Name ตั้งชื่อ Report ใหม่ และที่ Module ชื่อโมดูลที่จะเก็บรายงานใหม่ คลิกเลือก Save Report

๕. เข้ารายงานที่สร้างใหม่ คลิก Next



๖. เลือก Custom Report เลือก Report ที่สร้างใหม่ คลิก Next คลิก Next อีกครั้ง

๗. คลิกเลือก Design



การแสดงผลรายงานจากการออกแบบ

พื้นที่ Design มี ๕ แถบ ได้แก่

๑. แถบ Title

สิ่งที่อยู่ในแถบนี้ จะแสดงเฉพาะหน้าแรกเท่านั้น นิยมใช้เขียนชื่อรายงาน

๒. แถบ Header

จะแสดงที่ส่วนหัวของรายงานที่ทุกหน้าของรายงาน

๓. แถบ Detail

รายละเอียดของรายงาน ที่จะแสดงออกมาจากคำสั่งหลัก

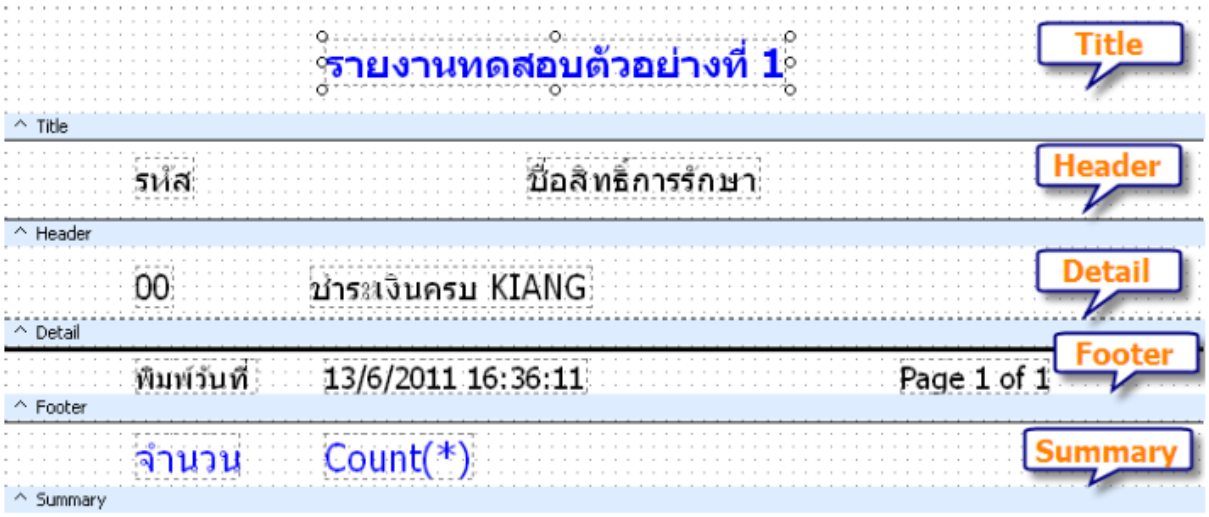
๔. แถบ Footer

จะแสดงส่วนท้ายของรายงานที่ทุกหน้าของรายงาน

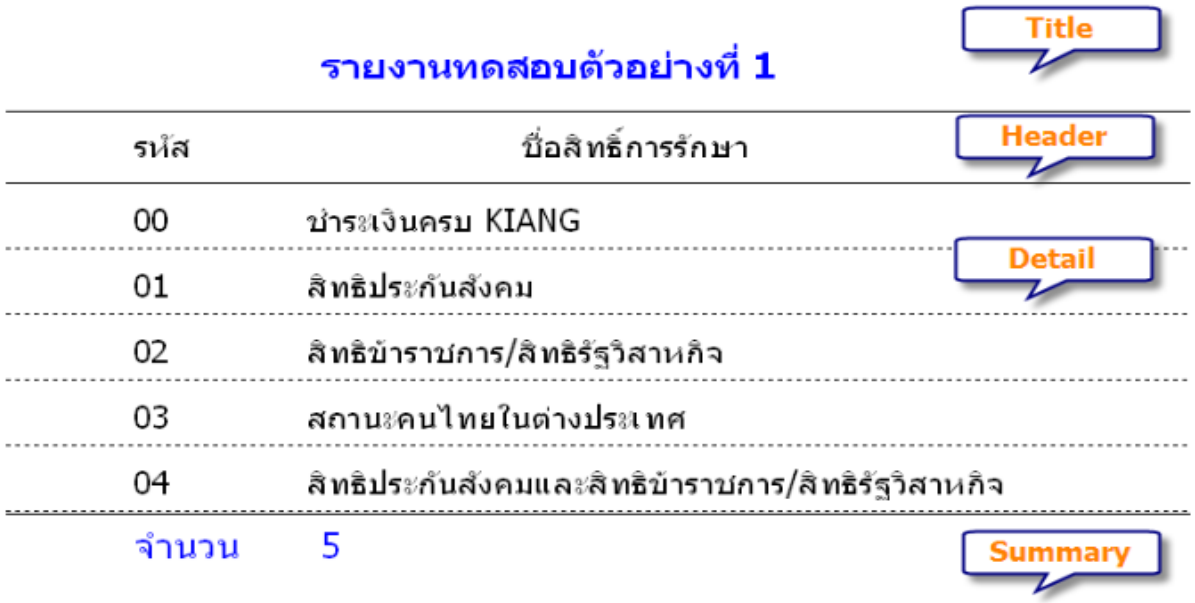
๕. แถบ Summary

จะแสดงต่อจาก Record สุดท้ายของ Detail โดยจะแสดงหน้าสุดท้ายเพียงหน้าเดียว

ตัวอย่างการออกแบบรายงาน



การแสดงผลรายงานที่ได้จากการออกแบบ



การเชื่อมคำ หรือคำสั่งด้วย ‘+’

ในการเขียนคำสั่งใน โปรแกรม Report Design จะต้องมีการเชื่อมคำ เพื่อให้โปรแกรมได้รู้จัก

ตัวอย่างการเชื่อคำหรือประโยคของคำสั่ง

Wonder girl

Wonder’+’girl	ผลที่ได้	wondergirl
Wonder ‘+’girl	ผลที่ได้	wonder girl
Wonder’+’ girl	ผลที่ได้	wonder girl

Select * from patient

Select * from’+’ patient	ผลที่ได้	Select * frompatient
Select * from ‘+’patient	ผลที่ได้	Select * from patient
Select * from’+’ patient	ผลที่ได้	Select * from patient

ดังนั้น คำสั่งใน Pascal Code จะมีลักษณะดังต่อไปนี้

```
Value := GetSQLStringData(‘select name from pttype ‘+
        ‘ order by pttype ‘);
```

ผลที่ได้ คือ

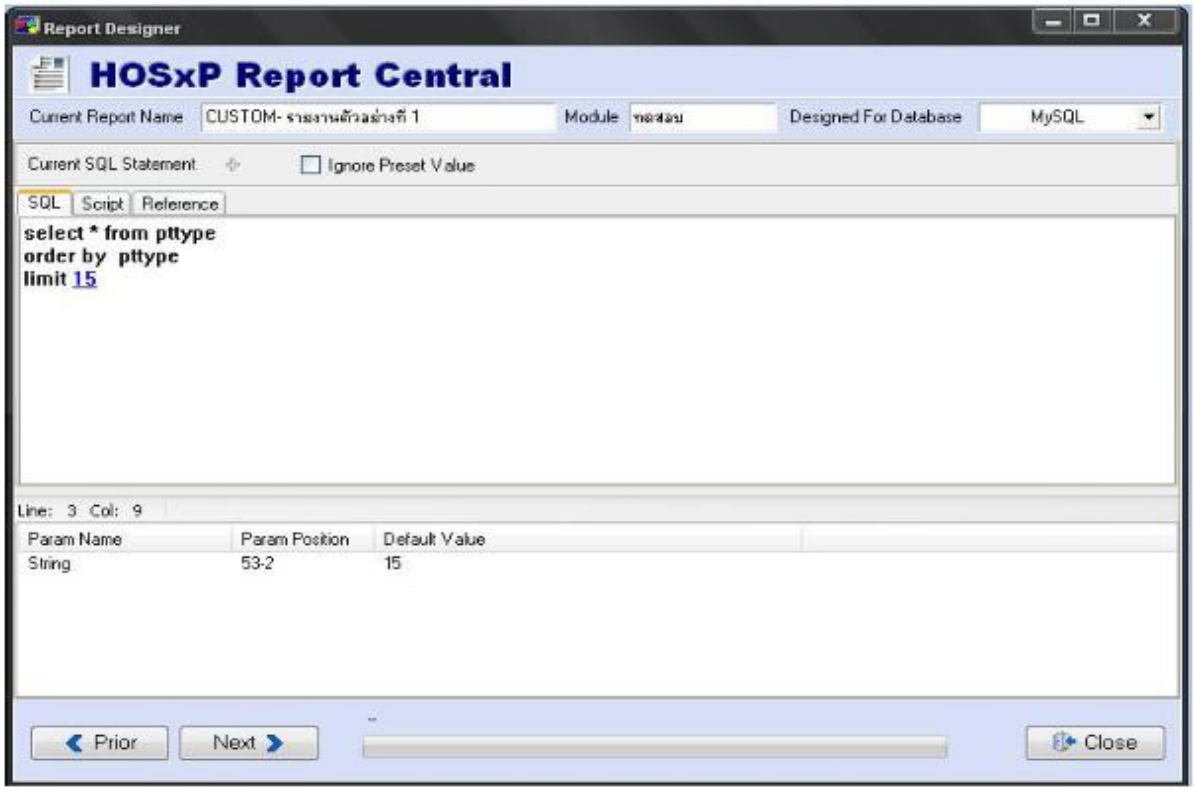
```
select name from pttype order by pttype
```

ความแตกต่างระหว่างหน้าจอที่ใส่คำสั่งหลัก

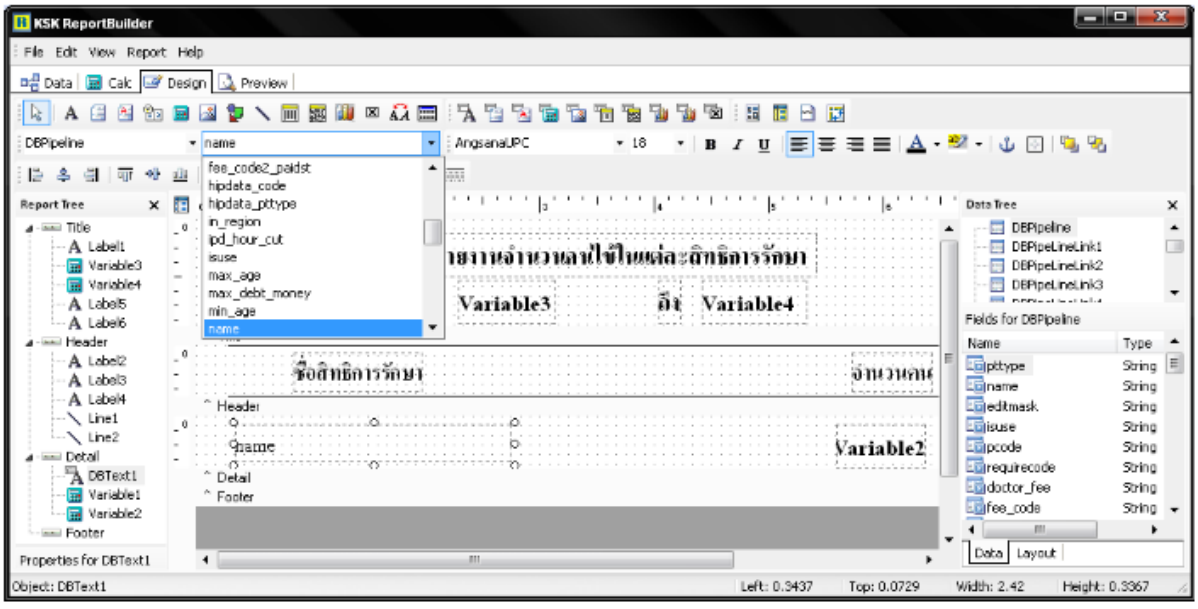
คำสั่งหลักใน Tab SQL

คำสั่งหลักใน Tab Calc

คำสั่ง SQL ใน Tab SQL

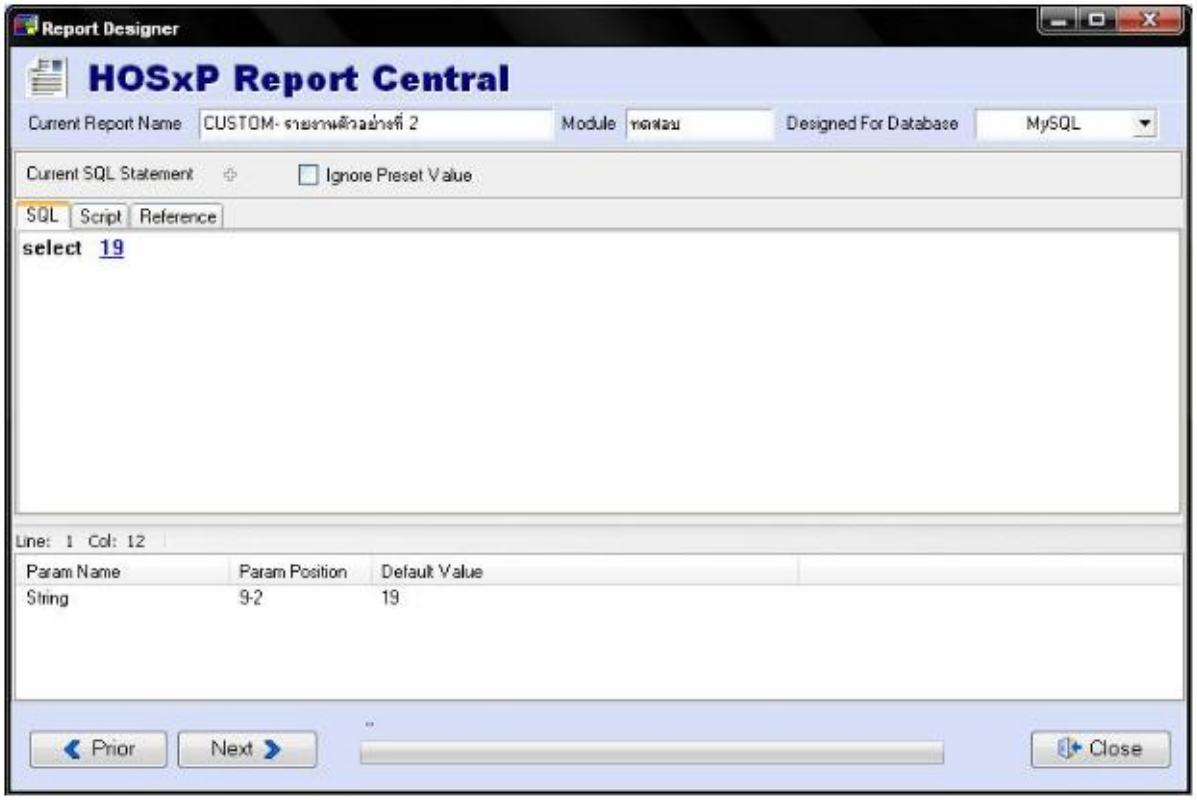


โดยผลลัพธ์ได้จากคำสั่งหลักใน Tab SQL สังเกตที่ Data Control จะปรากฏชื่อ DBPipeline

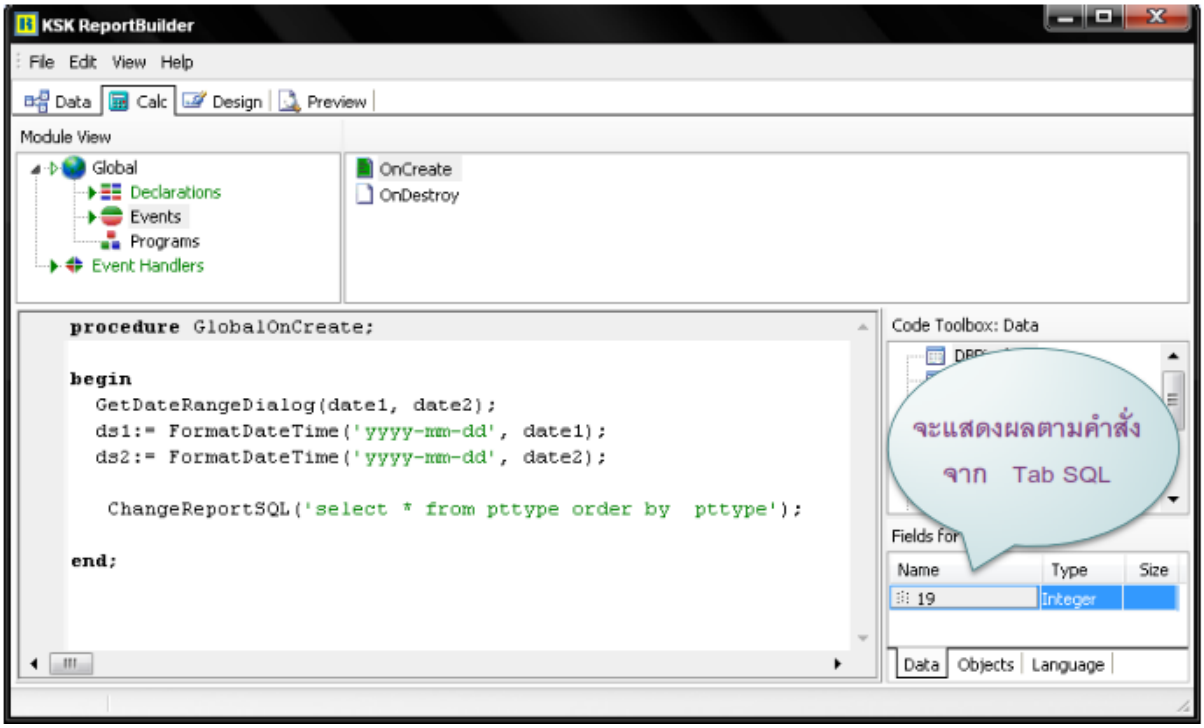


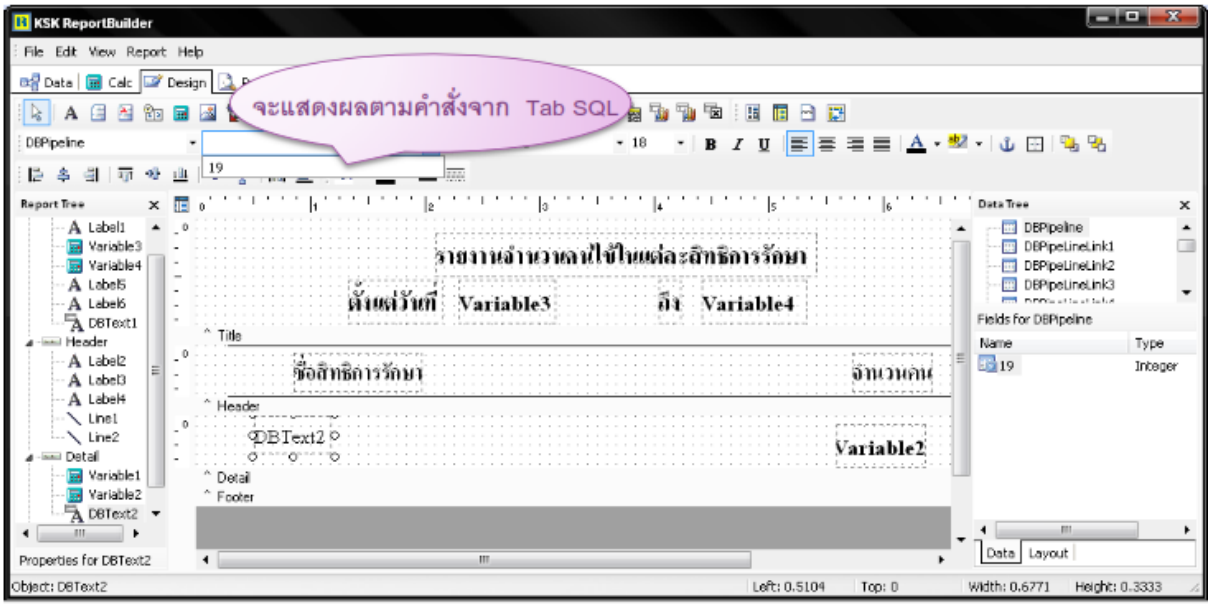
คำสั่ง SQL ใน Tab Calc

คำสั่งหลักใน Tab SQL จะ select อะไรก็ได้

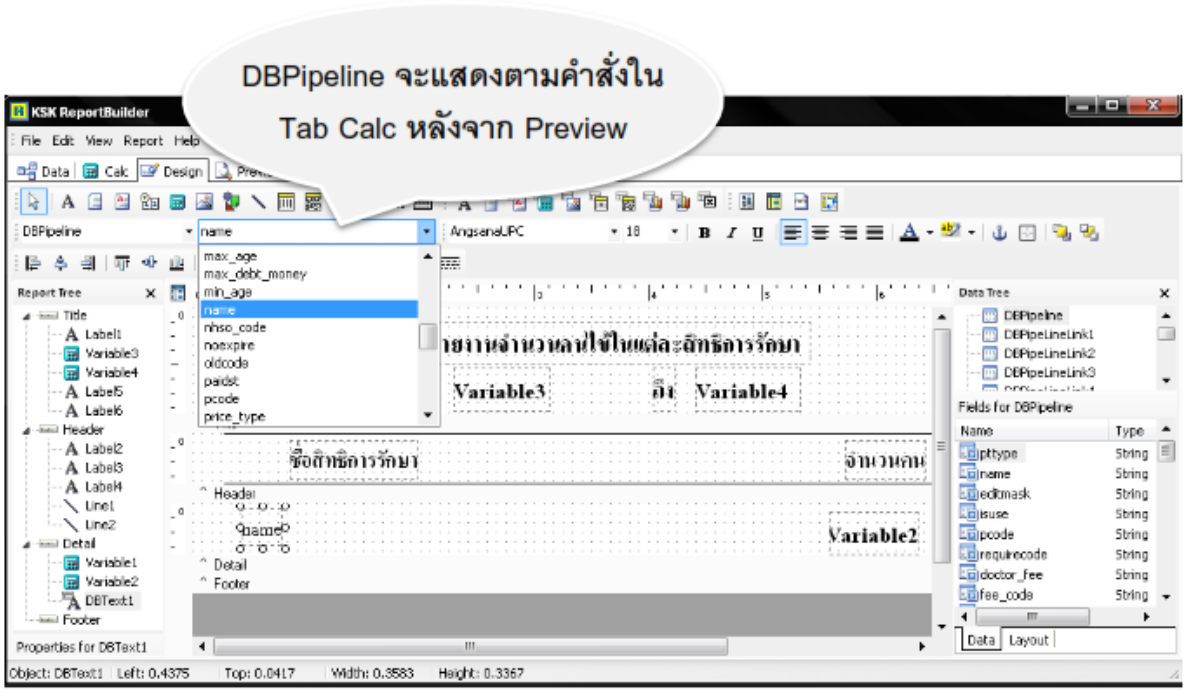


คำสั่งหลักใน Tab Calc





ให้ Preview แล้วให้กลับมาที่หน้าจอ Design สังเกตที่ Data Control



Function For Report

Toolbox: Function

Function Type

๑. String

เป็นฟังก์ชันกลุ่มของตัวอักษร ที่นำมาเรียงกันอยู่ภายในเครื่องหมายคำพูด (‘’) และไม่สามารถนำไปคำนวณได้

๒. Conversion

เป็นฟังก์ชันการแปลงค่า หรือแปลงชนิดของข้อมูลนั้นให้เป็นอีกค่าหนึ่ง

๓. Format

เป็นฟังก์ชันที่ใช้จัดรูปแบบของข้อมูล

๔. Date Time

เป็นฟังก์ชันในการจัดการข้อมูลวันที่ และเวลา

๕. Math

เป็นฟังก์ชันที่ช่วยคำนวณทางคณิตศาสตร์ที่ซับซ้อน

๖. Utility

เป็นฟังก์ชันที่ใช้ประโยชน์ในด้านต่างๆของโปรแกรม จะเป็นการแสดงข้อความ Massage

ฟังก์ชันที่มักใช้บ่อยในโปรแกรม Report Designer (Function frequently used)

๑. ฟังก์ชันแบบ String ได้แก่

๑.๑ ChangeReportSQL(sql)

เป็นฟังก์ชันไว้สำหรับ run คำสั่ง SQL ที่เป็นคำสั่งหลักโดยผลที่ได้จะเป็นไปตามค่าที่เราเขียนคำสั่ง SQL ซึ่งเราจะเรียกค่านั้นว่า DBPipline

๑.๒ GetDateRangeDialog(date๑, date๒);

เป็นฟังก์ชันที่สร้างช่วงวันที่ให้เลือก

๑.๓ GetDateTimeRangeDialog(date๑, date๒);

เป็นฟังก์ชันที่สร้างช่วงวันที่และเวลาให้เลือก

๑.๔ FormatThaiDate(fFormat, fDate);

เป็นฟังก์ชันเปลี่ยนรูปแบบวันที่จาก ค.ศ.เป็น พ.ศ.

๑.๕ GetPatientAddress(hn);

เป็นฟังก์ชันหาที่อยู่ของคนไข้

๑.๖ GetThaiAge(BirthDay, VisitDay);

เป็นฟังก์ชันหาอายุของ

๑.๗ HospitalName;

เป็นฟังก์ชันแสดงชื่อสถานพยาบาล

๑.๘ ThaiMoney(m);

เป็นฟังก์ชันที่แปลงจากค่าเงินที่เป็นตัวเลขเป็นตัวอักษร

๑.๙ GetCurrentUser;

เป็นฟังก์ชันที่แสดง Login name ของผู้ที่กำลังใช้งานอยู่

๑.๑๐ GetSQLStringData(sql);

เป็นฟังก์ชันที่ไว้สำหรับรับค่าที่เป็น String ซึ่งอาจเป็นได้ทั้งข้อความและตัวเลข ที่ไม่สามารถนำค่าที่ได้ไปคำนวณได้

๑.๑๑ GetPickupList(sql);

เป็นฟังก์ชันที่สร้างตัวเลือก ซึ่งสามารถเลือกได้ครั้งละ ๑ ตัวเลือก

๑.๑๒ GetMultipleList(sql);

เป็นฟังก์ชันที่สร้างตัวเลือก ซึ่งสามารถเลือกได้มากกว่า ๑ ตัวเลือก

๑.๑๓ InputQuery(title, label);

เป็นฟังก์ชันที่สร้างกล่องไว้สำหรับ รับค่าที่เป็น String หรือข้อความ

๒. Conversion

๒.๑ IntToStr(value);

เป็นฟังก์ชันเปลี่ยนประเภทข้อมูลจาก Integer เป็น string

๒.๒ StrToInt(S);

เป็นฟังก์ชันเปลี่ยนประเภทข้อมูลจาก String เป็น integer

๓. Format

๓.๑ FormatDateTime(Format, aDateTime);

เป็นฟังก์ชันที่เปลี่ยนรูปแบบของวันที่

๔. Date Time

๔.๑ GetSQLDateData(sql);

เป็นฟังก์ชันที่ไว้สำหรับ รับค่าที่เป็น Date วันที่

๕. Math

๕.๑ GetSQLDoubleData(sql);

เป็นฟังก์ชันที่ไว้สำหรับ รับค่าที่เป็นตัวเลขซึ่งอาจเป็นเลขทศนิยมก็ได้ และสามารถนำค่าที่ได้ไปคำนวณต่อได้

๕.๒ GetSQLIntegerData(sql);

เป็นฟังก์ชันที่ไว้สำหรับ รับค่าที่เป็นตัวเลขซึ่งไม่เป็นเลขทศนิยม และสามารถนำค่าที่ได้ไปคำนวณต่อได้

๖. Utility

๖.๑ ShowMessage(Msg);

เป็นฟังก์ชันที่ไว้สำหรับแสดงข้อความเตือนต่างๆ

ตัวอย่างการเขียนรายงานด้วยโปรแกรม Report Designer

๑. รายงานแสดงค่าใช้จ่ายของผู้ป่วยนอกแยกตามหมวดค่าใช้จ่าย

<u>รูปแบบ</u>

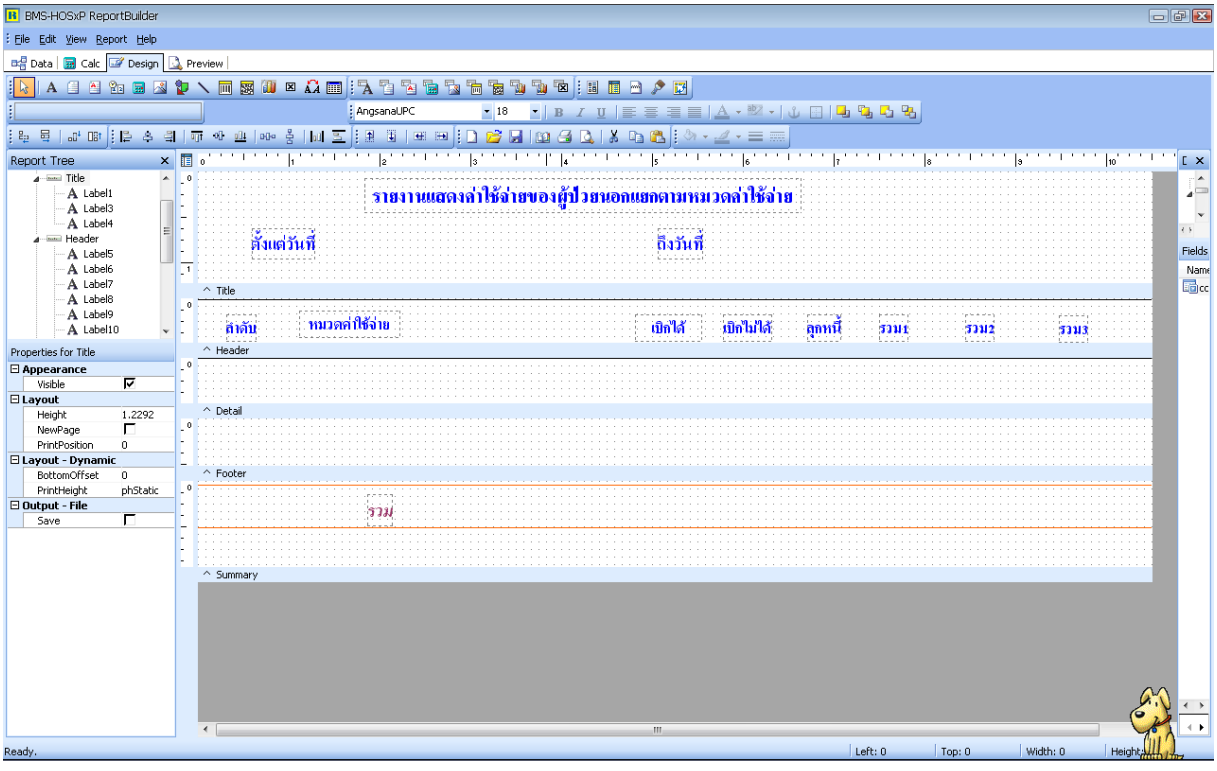
รายงานแสดงค่าใช้จ่ายของผู้ป่วยนอกแยกตามหมวดค่าใช้จ่าย

ตั้งแต่วันที่ ถึงวันที่

หมวดค่าใช้จ่าย	เบิกได้	เบิกไม่ได้	ลูกหนี้	รวม๑	รวม๒	รวม๓
ค่าห้องและค่าอาหาร	๐	๐	๐	๐	๐	๐
ค่าอวัยวะเทียมและอุปกรณ์ในการบำบัดรักษาโรค	๐	๐	๐	๐	๐	๐
ค่ายาในบัญชียาหลักแห่งชาติ	๐	๐	๐	๐	๐	๐
ค่ายากลับบ้าน	๐	๐	๐	๐	๐	๐
.....	๐	๐	๐	๐	๐	๐
ฯลฯ	๐	๐	๐	๐	๐	๐
รวม	๐	๐	๐	๐	๐	๐

<u>ขั้นตอนการเขียนรายงาน</u>

๑. เปิดโปรแกรม **Report Designer** และสร้าง Report ใหม่ เลือก Design คลิกเลือกแถบ Design ดำเนินการออกแบบรายงานที่ต้องการ โดยใช้เครื่องมือ label A ในการสร้างกล่องชื่อ



๒. การสร้างช่วงวันที่ที่ต้องการระบุ

๒.๑ การประกาศตัวแปรวันที่

ที่แถบ Calc ประกาศตัวแปรวันที่โดยกำหนด และคลิก Declarations

```
var
    date๑,date๒ : datetime;
    ds๑,ds๒ : string;
```

เสร็จแล้วลอง compile ตัวแปรดู

๒.๒ การสร้างกล่องวันที่ Dialog

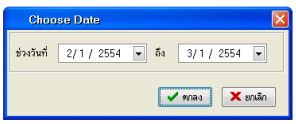
ที่แถบ Calc เลือก Events เลือก On Create เขียนคำสั่งโดยดำเนินการดังนี้

- ที่ Code Toolbox เลือก String
- เลือกแถบ Language เลือกฟังก์ชัน GetDateRangeDialog(date๑, date๒); ลากมาวางใต้ begin และตัวแปร date๑ และdate๒ จะต้องตรงกับชื่อตัวแปร datetime ที่ประกาศใน var ในที่นี้เขียนได้ดังนี้ GetDateRangeDialog(date๑, date๒);

๒.๓ เปลี่ยนรูปแบบวันที่ให้กลายเป็นรูปแบบ พ.ศ. โดยใช้ฟังก์ชัน FormatDateTime(Format, aDateTime); ที่ Code Toolbox เลือก Format เลือกแถบ Language เลือกฟังก์ชัน FormatDateTime(Format, aDateTime); ลากไปวางใต้คำสั่ง ฟังก์ชัน GetDateRangeDialog(date๑, date๒); และสร้างตัวแปรมารับค่า โดยใช้ตัวแปรที่ได้ประกาศตัวแปรไว้ที่ Declarations ได้แก่ ds๑,ds๒ : string; ไปวางหน้าฟังก์ชัน ในที่นี้เขียนคำสั่งฟังก์ชันได้ดังนี้

```
ds๑ := FormatDateTime('yyyy-mm-dd', date๑);
ds๒ := FormatDateTime('yyyy-mm-dd', date๒);
```

- เสร็จแล้วลอง compile และ preview



๒.๔ สร้างกล่องวันที่ในตัวรายงานที่หน้า Design

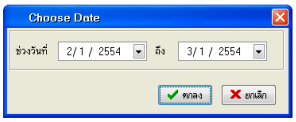
ใช้กล่องเครื่องมือ Variable มาวางตรงที่ต้องการช่วงวันที่ แล้วคลิกขวาที่ variable เลือก calculations... แล้วกำหนดค่าวันที่ให้ตรงตัวแปรที่เราต้องการ

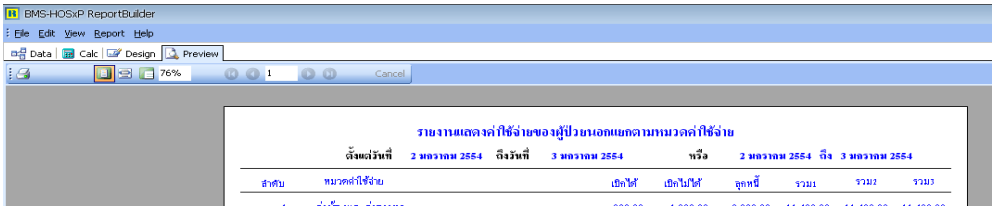
เช่น ค่าแรกกำหนดเป็น Value := date๑; ค่าที่สองกำหนดเป็น Value := date๒;

เสร็จแล้ว กำหนดชนิดข้อมูลเป็น date time ที่บริเวณมุมขวาบน

หรืออีกวิธีการสร้างวันที่โดยใช้รูปแบบการเชื่อมคำโดยใช้ + ในการสร้างกล่อง variable เพียงกล่องเดียว แล้วกำหนดเป็น

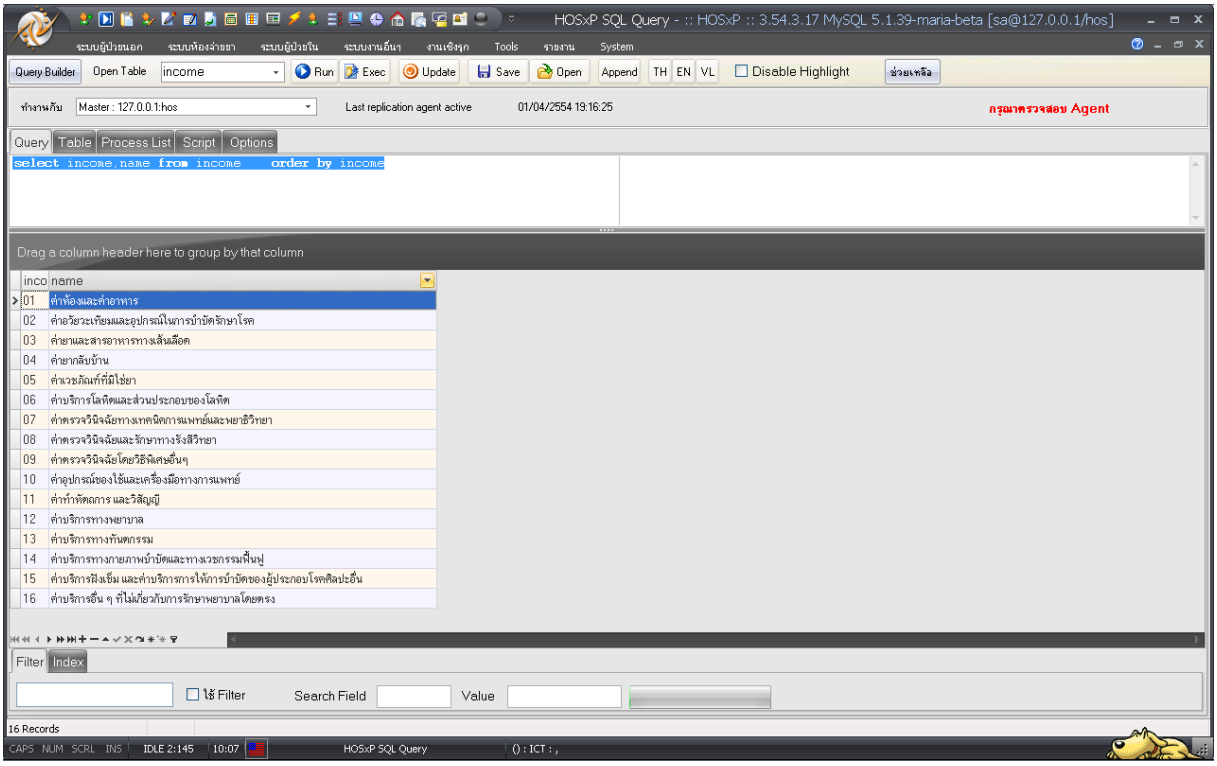
Value := FormatThaiDate('D MMMM EEEE', date๑)+' ' +'ถึง'+' '+ FormatThaiDate('D MMMM EEEE', date๒);





๓. การสร้างตัวแปรของข้อมูลที่ต้องการ

๓.๑ การเขียนคำสั่ง sql เลือกข้อมูลที่ต้องการใน HOSxP SQL Query ในตาราง income โดยใช้คำสั่ง SQL Query เป็น select income,name from income order by income ผลที่ได้จากการ RUN คำสั่ง



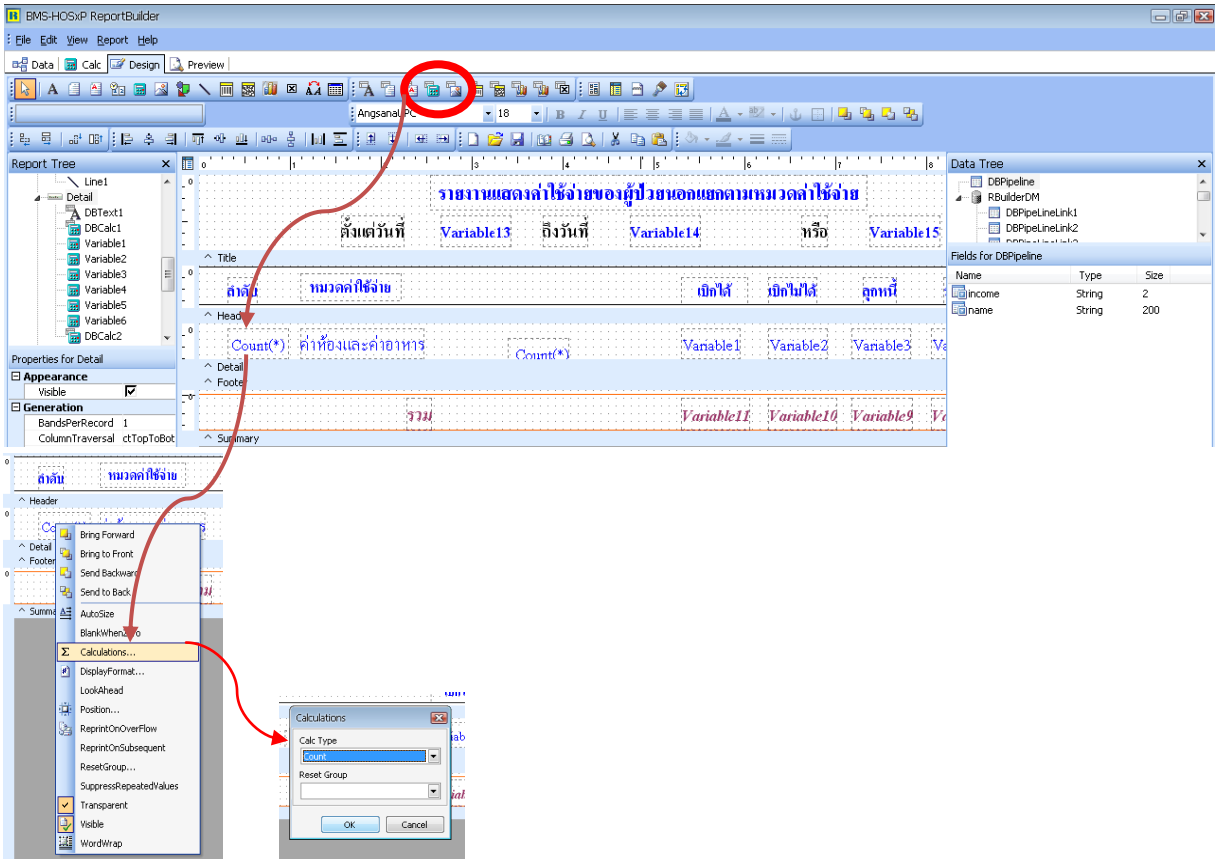
๓.๒ คัดลอกคำสั่งไปวางใน Calc ตามหลังฟังก์ชัน ChangeReportSQL(sql); โดยวางแทน sql โดยใส่เครื่องหมาย ‘’ก่อน ผลที่ได้

ChangeReportSQL('select income,name from income order by income');

โดยวางถัดจากฟังก์ชันของวันที่ เสร็จแล้ว ลอง compile และ preview ดู จะได้ ที่แถบ data จะมี จะปรากฏชื่อ DBPipeline

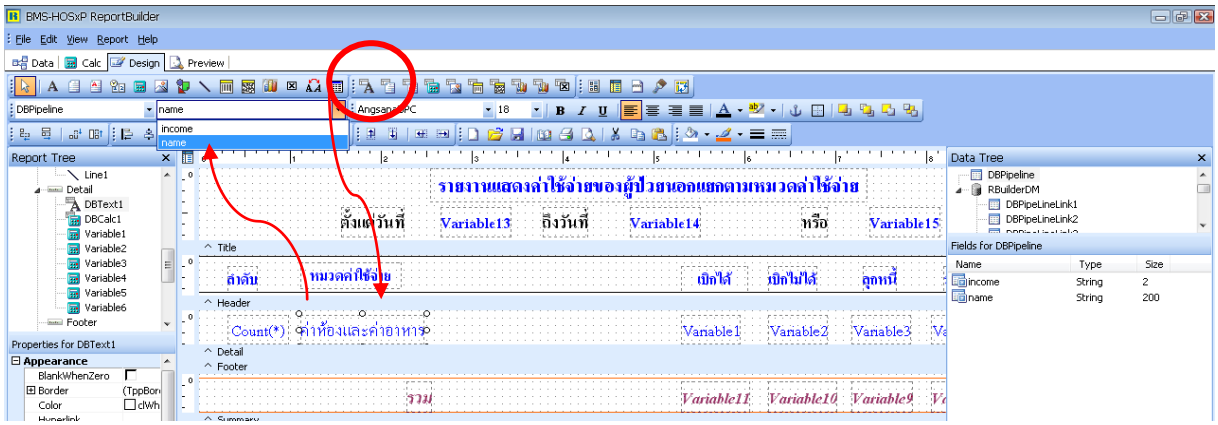
๓.๓ ออกแบบรายงานในหน้า Design โดยดำเนินการดังนี้

- การสร้างลำดับที่ให้ RUN อัตโนมัติ โดยใช้เครื่องชื่อ DBcalc คลิกใต้ชื่อลำดับในแถบ Detail แล้วคลิกขวาที่ variable เลือก calculations... Calc Type เลือก Count คลิก OK



- การสร้างข้อมูลตัวแปรในหมวดค่าใช้จ่าย

  โดยใช้เครื่องมือที่ชื่อว่า DBtext คลิกวางใต้หมวดค่าใช้จ่ายในแถบ Detail ที่แถบ DBPipeline ข้างบนเลือก name

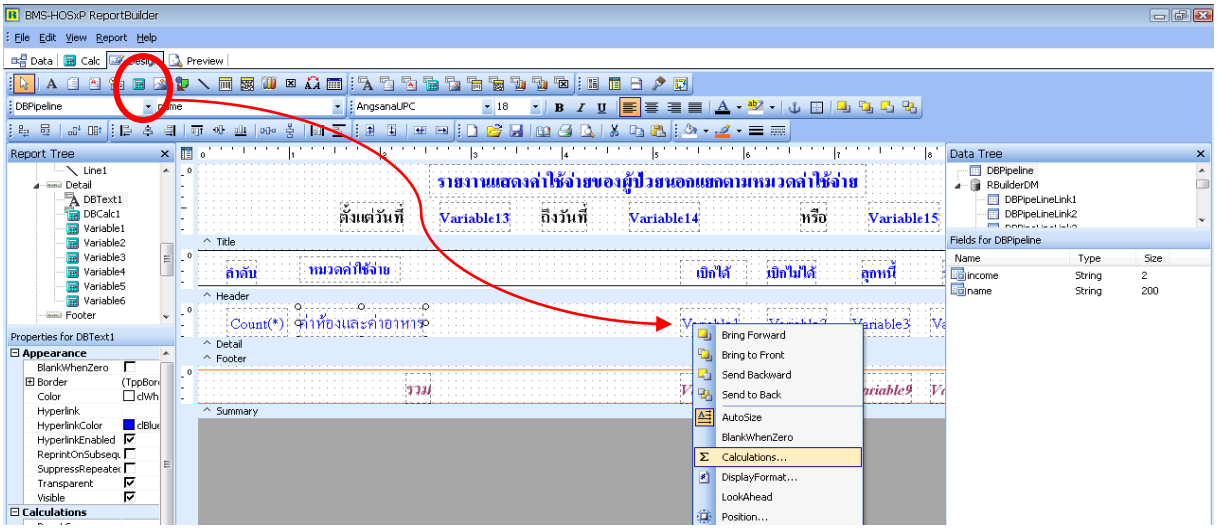


- การสร้างตัวแปรที่แสดงรายละเอียดค่าใช้จ่าย โดยตารางในโปรแกรม HOSxP จะเก็บค่าใช้จ่ายในตาราง opitemrece จะใช้แยกค่าใช้จ่ายตามช่วงวันที่ที่เราต้องการ และตามตาราง paidst จะระบุค่าประเภทของค่าใช้จ่าย ได้แก่ paidst = ๐๐ ค้างชำระ, ๐๑ ชำระเงินเบิกได้, ๐๒ ลูกหนี้สิทธิ, ๐๓ ชำระเองเบิกไม่ได้, ๐๔ ส่วนลดเงินสด

  โดยดำเนินการดังนี้

  ๑. ค่าใช้จ่ายประเภทเบิกได้

  โดยใช้เครื่องมือ Variable วางใต้ เบิกได้แล้วคลิกขวา variable เลือก calculations... กำหนดค่าตัวแปร variable โดยที่ Code Toolbox เลือก Math เลือกฟังก์ชัน GetSQLDoubleData(sql); มาวางต่อจาก Value := และเขียนคำสั่ง sql เป็น Value := GetSQLDoubleData('select sum(sum_price) from opitemrece where paidst = "๐๑" '+ 'and vstdate between "'+ds๑+'" and "'+ds๒+'" and income="'+DBPipeline['income']+'"'); อย่าลืมการเชื่อมคำด้วย + และการกำหนดค่า income = ให้ใช้ DBPipeline['income'] จากแถบ data ลากมาวางแทนการพิมพ์ใน "'++'"



๒. ค่าใช้จ่ายประเภทเบิกไม่ได้

โดยกำหนดค่า paidst = "๐๓"

Value := GetSQLDoubleData('select sum(sum_price) '+

'from opitemrece where '+'paidst = "๐๓" '+'and vstdate between

"'+ds๑+'" and "'+ds๒+'" '+'and income = "'+DBPipeline['income']+'"');

๓. ค่าใช้จ่ายประเภทลูกหนี้

โดยกำหนดค่า paidst = "๐๒"

Value := GetSQLDoubleData('select sum(sum_price) '+

'from opitemrece where '+'paidst = "๐๒" '+'and vstdate between

"'+ds๑+'" and "'+ds๒+'" '+'and income = "'+DBPipeline['income']+'"');

๔. ค่าใช้จ่ายรวม๑ (เป็นการรวมด้วย variable บวกไปเรื่อยๆ)

โดยกำหนดค่า

Value := variable๑.value + variable๒.value + variable๓.value;

๕. ค่าใช้จ่ายรวม๒

โดยกำหนดค่า

Value := GetSQLDoubleData('select sum(sum_price) '+'from opitemrece

where '+'paidst in ("๐๑","๐๒","๐๓") '+'and vstdate between "'+ds๑+'" and

"'+ds๒+'" '+'and income = "'+DBPipeline['income']+'"');

๖. ค่าใช้จ่ายรวม๓ (รวมด้วยฟังก์ชัน อย่าลืมประกาศตัวแปร sum๑)

โดยกำหนดค่า

```
if Report.DataPipeline.bof then sum๑ :=๐;
Value := GetSQLDoubleData('select sum(sum_price) '+
'from opitemrece where '+'paidst in ("๐๑","๐๒","๐๓") '+
'and  vstdate between "'+ds๑+'" and "'+ds๒+'" '+'and income =
"'+DBPipeline['income']+'");sum๑ := sum๑+value;
```

โดยรูปแบบการใช้ฟังก์ชันทั้ง ๓ แบบจะส่งค่าออกมาเท่ากัน

๗. ค่ารวมทั้งหมดแต่ละรายการ ทำได้ดังนี้

- ค่ารวมเบิกได้ ใช้เครื่องมือ variable คลิกขวา เลือก calculations... กำหนดค่าตัวแปร

```
Value := GetSQLDoubleData('select sum(sum_price) '+
'from opitemrece where '+
'paidst in ("๐๑") '+
'and  vstdate between "'+ds๑+'" and "'+ds๒+' ");
```

- ค่ารวมเบิกไม่ได้ กำหนดค่าตัวแปร

```
Value := GetSQLDoubleData('select sum(sum_price) '+
'from opitemrece where '+
'paidst = "๐๓" '+
'and  vstdate between "'+ds๑+'" and "'+ds๒+'" '+'and income =
"'+DBPipeline['income']+'");
```

- ค่ารวมลูกหนี้

```
Value := GetSQLDoubleData('select sum(sum_price) '+
'from opitemrece where '+
'paidst = "๐๒" '+
'and  vstdate between "'+ds๑+'" and "'+ds๒+'" '+'and income =
"'+DBPipeline['income']+'");
```

- ค่าผลรวมของรวม๑

เป็นการรวมด้วย variable บวกไปเรื่อยๆ การรวมแบบนี้จะเป็นการรวมในแนวนอน คือ เอา variable ของรวมเบิกได้ รวมเบิกไม่ได้ และรวมลูกหนี้มาบวกกัน ได้แก่

```
Value := variable๑๑.value+variable๑๐.value+variable๙.value
```

- ค่าผลรวมของรวม๒ (รวมด้วยคำสั่ง ตัดเงื่อนไขบางส่วนออกไป)

  Value := GetSQLDoubleData('select sum(sum_price) '+
  'from opitemrece where '+
  'paidst in ("๐๑","๐๒","๐๓") '+
  'and vstdate between "'+ds๑+'" and "'+ds๒+' "');

- ค่าผลรวมของรวม๓ (รวมด้วยฟังก์ชัน)

  Value := sum๑;

  ๘. ปิดรายงาน แล้วบันทึกรายงาน เข้ารายงานใหม่ ลอง preview ดู

ตัวอย่างของรายงานแบบนี้ เป็นการคำนวณค่าของจำนวนเงิน สามารถใช้ฟังก์ชัน GetSQLDoubleData(sql); หรือ GetSQLIntegerData(sql); ก็ได้ เพราะฟังก์ชันทั้ง ๒ แบบ สามารถส่งค่าออกมาแล้วสามารถนำไปคำนวณได้ แต่โดยปกติจะใช้แบบ GetSQLDoubleData(sql); และ จุดประสงค์ของตัวอย่างรายงานนี้เพื่อให้เห็นตัวอย่างคำสั่งของการรวมแต่ละแบบ

๒. ตัวอย่างแบบรายงานที่ ๒ รายงานแสดงรายชื่อผู้ป่วยนัด

เป็นการใช้ฟังก์ชัน GetPickupList

<u>รูปแบบ</u>

รายงานแสดงรายชื่อผู้ป่วยนัด : (นัดทั้งหมด, มาตามนัด, ขาดนัด)

ตรวจสอบวันที่นัด ถึง วันที่

ลำดับที่	วันที่นัด	HN	ชื่อ – สกุล	รายการนัด	แพทย์ผู้นัด

การออกแบบรายงาน

๑. ออกแบบรายงานตามแบบฟอร์มในแถบ Design

๒. สร้างกล่องวันที่เพื่อรับตัวแปรการระบุช่วงวันที่

๒.๑ การประกาศตัวแปรวันที่

ที่แถบ Calc ประกาศตัวแปรวันที่โดยกำหนด และคลิก Declarations

```
var
   date๑,date๒ : datetime;
   ds๑,ds๒ : string;
```

เสร็จแล้วลอง compile ตัวแปรดู

๒.๒ การสร้างกล่องวันที่ Dialog

ที่แถบ Calc เลือก Events เลือก On Create เขียนคำสั่งโดยดำเนินการดังนี้

- ที่ Code Toolbox เลือก String
- เลือกแถบ Language เลือกฟังก์ชัน GetDateRangeDialog(date๑, date๒);

ลากมาวางใต้ begin และตัวแปร date๑ และdate๒ จะต้องตรงกับชื่อตัวแปร datetime ที่ประกาศใน var ในที่นี้เขียนได้ดังนี้ GetDateRangeDialog(date๑, date๒);

๒.๓ เปลี่ยนรูปแบบวันที่ให้กลายเป็นรูปแบบ พ.ศ. โดยใช้ฟังก์ชัน FormatDateTime(Format, aDateTime); ที่ Code Toolbox เลือก Format เลือกแถบ Language เลือกฟังก์ชัน FormatDateTime(Format, aDateTime); ลากไปวางใต้คำสั่ง ฟังก์ชัน GetDateRangeDialog(date๑, date๒); และสร้างตัวแปรมารับค่า โดยใช้ตัวแปรที่ได้ประกาศตัวแปรไว้ที่ Declarations ได้แก่ ds๑,ds๒ : string; ไปวางหน้าฟังก์ชัน ในที่นี้เขียนคำสั่งฟังก์ชันได้ดังนี้

ds๑ := FormatDateTime('yyyy-mm-dd', date๑);

ds๒ := FormatDateTime('yyyy-mm-dd', date๒);

- เสร็จแล้วลอง compile และ preview

๒.๔ สร้างกล่องวันที่ในตัวรายงานที่หน้า Design

สร้างวันที่โดยใช้เครื่องมือ Variable ในการสร้างกล่อง คลิกขวาที่ variable เลือก calculations... กำหนดค่าตัวแปร โดยใช้ฟังก์ชัน ISO๒Date(d, f); ที่ Code Toolbox เลือก Format เลือกแถบ Language เลือกฟังก์ชัน ISO๒Date(d, f); ไปวาง แก้ไขคำสั่งเป็น

Value := ISO๒Date(ds๑, 'D MMMM EEEE')+' ถึงวันที่ '+ ISO๒Date(ds๒, 'D MMMM EEEE');

๓. การสร้าง Pickup List เพื่อแสดงประเภทคนไข้นัด(นัดทั้งหมด, มาตามนัด, ขาดนัด)

๓.๑ ประกาศตัวแปร

App : string (app จะรับค่าตัวแปรประเภทคนไข้นัด)

๓.๒ ใช้ฟังก์ชัน GetPickupList(sql); ที่ Code Toolbox เลือก Format เลือกแถบ Language เลือกฟังก์ชัน GetPickupList(sql); ไปวางใต้ begin โดยเขียนคำสั่ง SQL ดังนี้

```
app:= GetPickupList('select "คนไข้นัดทั้งหมด"
        Union select "มาตามนัดจริง"
        Union select "ไม่ได้มาตามนัด"');
 If app = 'คนไข้นัดทั้งหมด' then
         ChangeReportSQL('select * from oapp where nextdate between
        "'+ds๑+'" and "'+ds๒+'"')
   Else
    If app = 'มาตามนัดจริง' then
        ChangeReportSQL('select * from oapp where nextdate between
        "'+ds๑+'" and "'+ds๒+'"'+' and patient_visit = "y"')
     Else
      If app = 'ไม่ได้มาตามนัด' then
```

ChangeReportSQL('select * from oapp where nextdate between "'+ds๑+'" and "'+ds๒+'""+' and (patient_visit = "N" or patient_visit is null or patient_visit = "")');

End;

๔. การสร้างตัวแปร เพื่อรับค่าประเภทคนไข้นัดในแถบ Design
ใช้เครื่องมือ variable วางต่อจากชื่อ ตาราง คลิกขวาเลือก calculations... กำหนดค่าตัวแปร Value := app;

๕. ลอง compile และ Preview ดู

๖. การสร้างลำดับที่ให้ RUN อัตโนมัติ โดยใช้เครื่องชื่อ DBcalc คลิกใต้ชื่อลำดับในแถบ Detail แล้วคลิกขวาที่ variable เลือก calculations... Calc Type เลือก Count คลิก OK

๗. การสร้างตัวแปร วันที่นัด และ HN
โดยใช้เครื่องมือที่ชื่อว่า DBtext คลิกวางใต้ตัวแปรในแถบ Detail ที่แถบ DBPipeline ข้างบนเลือก ค่าวันที่นัด และ เลข HN

๘. การสร้างตัวแปร ชื่อ – สกุลคนไข้นัด โดยใช้เครื่องมือ variable คลิกขวา กำหนดค่าเป็น
Value := GetSQLStringData('select concat(pname,fname," ",lname) from patient where hn = "'+DBPipeline['hn']+'"');

๙. การสร้างตัวแปรแพทย์ผู้นัด โดยใช้เครื่องมือ variable คลิกขวา กำหนดค่าเป็น
Value := GetSQLStringData('select name from doctor
where code = "'+DBPipeline['doctor']+'"');

๓. ตัวอย่างแบบรายงานที่ ๓ แบบบันทึกการตรวจรักษา

<u>รูปแบบ</u>

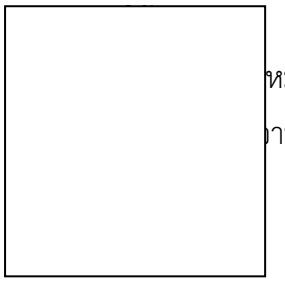
บันทึกการตรวจรักษา หน่วยบริการ รพช.เชียงกลาง

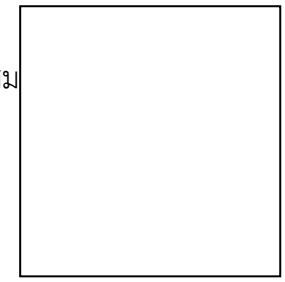
ชื่อ – สกุล ที่อยู่ HN

วันที่มารับบริการ แพทย์ผู้รักษา วันที่นัด อายุ

BW H BMI รอบเอว BP RR HR

ER Image ER Image การให้คำแนะนำ





☐ หมาะสม

☐ าหาร

☐

☐

☐

อาการ (CC.)

HPI

PE

การรักษา

วิธีการออกแบบรายงาน

๑. ออกแบบรายงานในแถบ Design

๑.๑ ชื่อหน่วยบริการ ใช้ฟังก์ชัน Value := HospitalName;

๒. ประกาศตัวแปร

```
var
        ad๑,ad๒,ad๓,ad๔,ad๕                : string;
        ad๑๑,ad๒๒,ad๓๓,ad๔๔,ad๕๕      : string;
        hn                                 : string;
        date๑,date๒                        : datetime;
        ds๑,ds๒,vstdate                    : string;
```

๓. แถบ Calc เลือก Events เลือก On Create เขียนคำสั่ง ดังนี้

```
begin
        hn:= InputQuery('plese insert HN', 'ใส่หมายเลข HN ที่ต้องการ');
        vstdate := GetListFromQuery('select vstdate from opdscreen where
       hn = "'+hn+'" order by vstdate desc ');
        ShowMessage(vstdate);
       GetDateRangeDialog(date๑, date๒);
       ds๑ := FormatDateTime('yyyy-mm-dd', date๑);
       ds๒ := FormatDateTime('yyyy-mm-dd', date๒);
       ChangeReportSQL('select o.vn,o.hn,o.vstdate,bpd,bps,bw,cc,
       pe,pulse,temperature ,rr ,height, advice๑, advice๒,advice๓,advice๔,
       advice๕,advice๖,advice๗,bmi ,hpi,waist , v.dx_doctor ,
       concat(v.age_y," ปี " ,v.age_m," เดือน") as age,pe_image.image๑ '+
       ' from opdscreen o '+
              ' left outer join vn_stat v on o.vn=v.vn '+
               ' left outer join pe_image on o.vn = pe_image.vn  '+
               ' left outer join er_image on o.vn = er_image.vn '+
       ' where o.hn="'+hn+'" and o.vstdate between "'+ds๑+'" and "'+ds๒+'"
       '+'order by o.vstdate desc');
End;
```

เสร็จแล้ว ลอง compile และลอง preview ดู

๔. การสร้างตัวแปร ชื่อ – สกุลผู้รับบริการ

ใช้เครื่องมือ variable คลิกขวาเลือก calculations... ใส่ค่าคำสั่งเป็น

Value :=GetSQLStringData('select concat(pname,fname," ",lname) from patient where hn="'+DBPipeline['hn']+'" ');

๕. การสร้างตัวแปรที่อยู่ผู้รับบริการ

ใช้ฟังก์ชัน GetPatientAddress(hn); ใช้เครื่องมือ variable คลิกขวาเลือก calculations... ใส่ค่าคำสั่งเป็น

Value := GetPatientAddress(DBPipeline['hn']);

๖. การสร้างข้อมูล HN ของผู้รับบริการ

ใช้เครื่องมือ DBtext แล้วใส่ค่า DBPipeline เป็น HN

๗. การสร้างข้อมูล วันที่มารับบริการ

ใช้เครื่องมือ DBtext แล้วใส่ค่า DBPipeline เป็น vstdate

๘. การสร้างตัวแปร แพทย์ผู้รักษา

ใช้ฟังก์ชัน GetSQLStringData(sql); ใช้เครื่องมือ variable คลิกขวาเลือก calculations... ใส่ค่าคำสั่งเป็น

Value := GetSQLStringData('select name from doctor where code = "'+DBPipeline['dx_doctor']+'"');

๙. การสร้างตัวแปร วันที่นัด

ใช้ฟังก์ชัน GetSQLDateData(sql); ใช้เครื่องมือ variable คลิกขวาเลือก calculations... ใส่ค่าคำสั่งเป็น

Value :=GetSQLDateData('select nextdate from oapp where vn ="'+DBPipeline['vn']+'");

๑๐. การสร้างข้อมูล อายุของผู้รับบริการ

ใช้เครื่องมือ DBtext แล้วใส่ค่า DBPipeline เป็น age

๑๑. การสร้างข้อมูล BW.ของผู้รับบริการ

ใช้เครื่องมือ DBtext แล้วใส่ค่า DBPipeline เป็น bw

๑๒. การสร้างข้อมูล H.ของผู้รับบริการ

ใช้เครื่องมือ DBtext แล้วใส่ค่า DBPipeline เป็น height

๑๓. การสร้างข้อมูล BMI.ของผู้รับบริการ

ใช้เครื่องมือ DBtext แล้วใส่ค่า DBPipeline เป็น bmi

๑๔. การสร้างข้อมูล รอบเอวของผู้รับบริการ

ใช้เครื่องมือ DBtext แล้วใส่ค่า DBPipeline เป็น waist

๑๕. การสร้างข้อมูล BPI.ของผู้รับบริการ

ใช้เครื่องมือ DBtext แล้วใส่ค่า DBPipeline เป็น bps “/” bpd

๑๖. การสร้างข้อมูล RR.ของผู้รับบริการ

ใช้เครื่องมือ DBtext แล้วใส่ค่า DBPipeline เป็น rr

๑๗. การสร้างข้อมูล HR.ของผู้รับบริการ

ใช้เครื่องมือ DBtext แล้วใส่ค่า DBPipeline เป็น pulse

๑๘. การสร้างบล็อกคำแนะนำ

โดยนำคำแนะนำในตาราง opdscreen ในข้อมูล advice๑-๗ มาใช้โดยใช้หลักการ ถ้ามีการให้คำแนะนำในเรื่องดังกล่าว จะโชว์เครื่องหมาย / ในหัวข้อดังกลาง โดยดำเนินการ

- ออกแบบการให้คำแนะนำในแถบ Design
- ใช้เครื่องมือ Checkbox ☒ ในแถบเมนูบาร์ นำเครื่องหมายถูกที่ Mark ออกในแถบ properties for checkbox โดยจะใช้คำสั่งให้กระทำแทน
- ใช้เครื่องมือ variable คลิกขวา เขียนคำสั่ง

```
ad๑ := GetSQLStringData('select advice๑ from opdscreen where vn =
        "'+DBPipeline['vn']+'");
 if
        ad๑='Y' then
        checkbox๑.checked := true
 else
        checkbox๑.checked := false;
```

กระทำอย่างนี้ทุกข้อคำแนะนำ

๑๙. การทำรายการ อาการสำคัญ (CC.)

ดำเนินการดังนี้

- ออกแบบรายงานในแถบ design
- ใช้เครื่องมือ DBmono บนแถบเมนู ในการสร้าง
- ที่ DBPipeline เลือก CC.

๒๐.