Royal University of Phnom Penh

Faculty of Science

Departement of Computer Science

Lecturer : Kheam Hong

Subject : Database Systems Implementation

មេរៀនទី១ អំពី ៖

Introduction to Structured Query Language (SQL)

សមាជិកក្រុម ៖

កាន ចន្ធា

ខ្ជប់ សាម៉ែ

សុវណ្ណ បណ្ឌិត

មាតិកា

មាតិកា 2

សេចក្តីផ្តើម 4

I. គោលបំណងមេរៀន (Objective) 5

II. ការរៀបចំដំណើរការកម្មវិធី DBMS 6

ការ Set up និង Install 6

ការតម្លើង Microsoft SQL Server 6

III. ការអនុវត្ត 19

៣.១.១.SELECT & FROM 19

៣.១.២. INSERT INTO (ការបញ្ចូលទិន្នន័យថ្មី) 51

៣.១.៣.UPDATE 55

៣.១.៤.DELETE 58

៣.១.៥. Comparation operators 59

៣.១.៦. Logical operators 62

៣.១.៧ Special operators 65

៣.១.៨ Aggregate Fucntions 71

Aggregate Functions (អនុគមន៍គណនាសរុប)៖ គឺជាបណ្តុំមុខងារនៅក្នុង SQL ដែលប្រើសម្រាប់សង្ខេប ឬគណនាតម្លៃលេខចេញពីជួរដេក (Rows) ជាច្រើននៅក្នុង Query ឱ្យមកនៅសល់តម្លៃសរុបតែមួយ។ ក្នុងចំណោមអនុគមន៍ទាំង ៥ ខាងក្រោមនេះ COUNT គឺជាមុខងារដែលត្រូវបានគេប្រើប្រាស់ញឹកញាប់ជាងគេ ដោយវាផ្តល់នូវការរាប់ចំនួនសរុបនៃជួរដេក

នៅក្នុង Query។ ចំណែកឯអនុគមន៍គណនាសរុបផ្សេងៗទៀត ត្រូវបានប្រើប្រាស់សម្រាប់សង្ខេបតម្លៃប្រភេទលេខ (Numeric Values)។ 71

ទម្រង់ទូទៅ (Syntax)៖ 71

SELECT aggregate_function(column_name) AS alias_name 71

FROM table_name; 71

៣.២.១ ការអនុវត្ត Data Definition Language (DDL) 75

IV. សេចក្តីសន្និដ្ឋាន (Conclusion) 98

សេចក្តីផ្តើម

នៅក្នុងយុគសម័យឌីជីថលនាពេលបច្ចុប្បន្ន ការគ្រប់គ្រងទិន្នន័យគឺជាផ្នែកមួយដ៏សំខាន់បំផុតសម្រាប់រាល់ស្ថាប័ន និងអាជីវកម្មគ្រប់ប្រភេទ។ ដើម្បីអាចទាក់ទង និងបញ្ជូនទៅកាន់ប្រព័ន្ធមូលដ្ឋានទិន្នន័យ (Database) *ឬាន យើងចាំបាច់ត្រូវមានភាសាមួយដែលជាស្តង់ដារ នោះគឺ* Structured Query Language (SQL)*។*

រឿយការណ៍នេះត្រូវបានរៀបចំឡើងដើម្បីបង្ហាញពីលទ្ធផលនៃការសិក្សានិងការអនុវត្តជាក់ស្តែងអំពីមូលដ្ឋានគ្រឹះនៃភាសា SQL*។ ខ្លឹមសារសំខាន់ៗដែលនឹងត្រូវលើកយកមកបង្ហាញ រួមមានការដំឡើងកម្មវិធីគ្រប់គ្រងទិន្នន័យ* (DBMS), *ការប្រើប្រាស់ពាក្យបញ្ជាសម្រាប់គ្រប់គ្រងរចនាសម្ព័ន្ធ* (DDL), *និងការប្រើប្រាស់ពាក្យបញ្ជាសម្រាប់ចាត់ចែងទិន្នន័យ* (DML) *តាមរយៈករណីសិក្សាជាក់ស្តែងលើប្រព័ន្ធព័ត៌មាននិស្សិត។ តាមរយៈការអនុវត្តនេះ យើងនឹងយល់កាន់តែច្បាស់ពីរបៀបដែលទិន្នន័យត្រូវបានរក្សាទុក កែប្រែ និងទាញយកមកប្រើប្រាស់ប្រកបដោយប្រសិទ្ធភាព។*

I. គោលបំណងមេរៀន (Objective)

បន្ទាប់ពីសិក្សាមេរៀននេះរួច និស្សិតនឹងអាច៖

- *យល់អំពីមូលដ្ឋានគ្រឹះនៃ* SQL *និងតួនាទីរបស់វាក្នុងការគ្រប់គ្រង* Database*។*
- *ការប្រើប្រាស់* SQL *ឲ្យមានប្រសិទ្ធិភាពខ្ពស់*
- *ស្គាល់ប្រភេទពាក្យបញ្ជា* SQL *សំខាន់ៗ ដូចជា* DML, DDL, TCL *និង* DCL*។*
- *អនុវត្តការបង្កើត និងកែប្រែ* Database Objects *ដូចជា* Database, Table, Index *និង* View*។*
- *ប្រើ* SQL *ដើម្បីគ្រប់គ្រងទិន្នន័យ រួមមានការបន្ថែម កែប្រែ លុប និងទាញយកទិន្នន័យ។*
- *យល់ពីការប្រើប្រាស់* SQL *ក្នុង* Data Retrieval *ដើម្បីបំពេញតម្រូវការអាជីវកម្ម និងស្វែងរកព័ត៌មានមានប្រយោជន៍។*
- *យល់អំពីស្តង់ដារ* SQL *និងសមត្ថភាពក្នុងការប្រើវាជាមួយ* DBMS *ផ្សេងៗ ដូចជា* MySQL, SQL Server *និង* Oracle *។*

II. ការរៀបចំដំណើរការកម្មវិធី DBMS

ការ Set up និង Install

សមាជិក	Edition	Version	Status
កាន ចន្ថា	Express	22.5.0	✓
ខ្ពប់ សាម៉ែ	Express	22.5.0	✓
សុវណ្ណ បណ្ឌិត	Express	22.5.0	✓

ការតម្លើង Microsoft SQL Server

- **Hardware Requirements**

Component	Requirement
Storage	Minimum of 6GB of available hard drive space.
Monitor	Super-VGA (800x600) or higher resolution monitor.

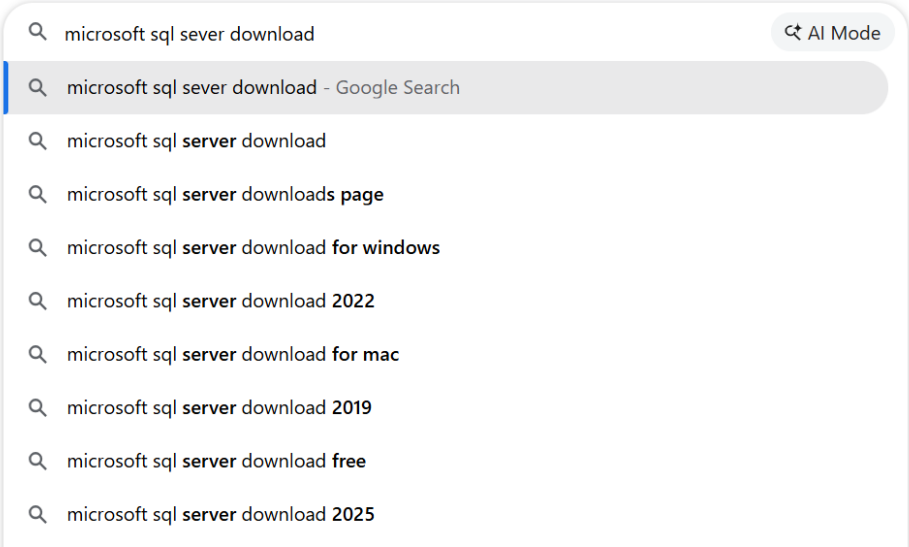
Memory	Requirement
Minimum memory	Express editions: 512 MB All other editions: 1 GB

Recommended memory	Express editions: 1 GB All other editions: At least 4 GB

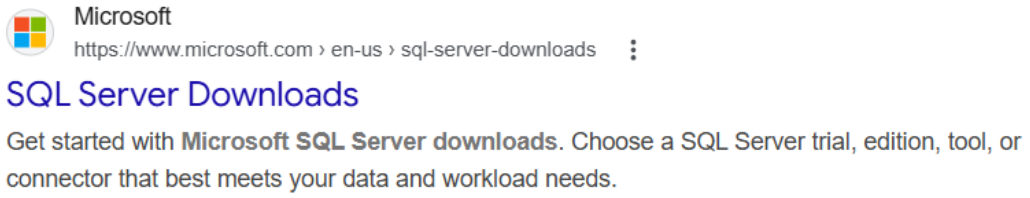
Processor	Requirement
Processor type	X64 processor. All Intel and AMD x86-64 CPUs with up to 64 cores per NUMA node.
Minimum speed	1.4 GHz
Recommended speed	1.0 GHz or faster

✚ ដំណើរការតម្លើង

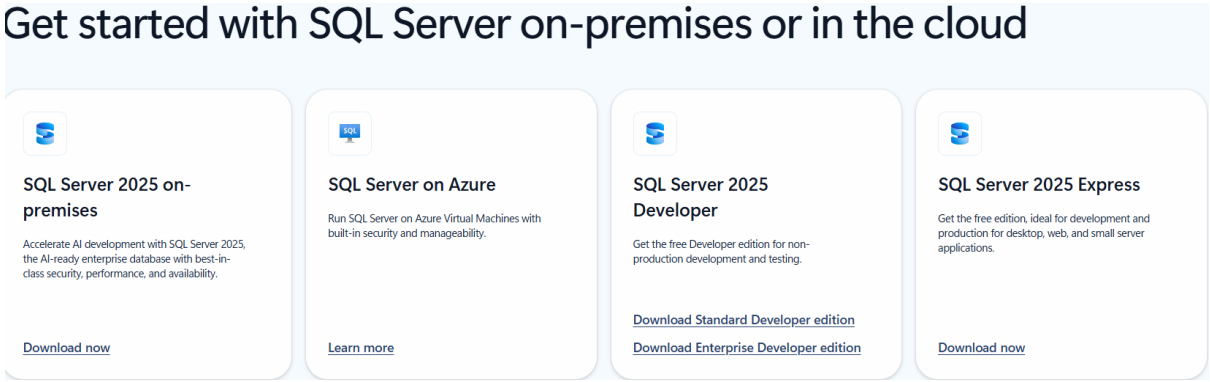
- ជំហានទី១ : វាយពាក្យ "Microsoft SQL Server Download" នៅលើ Google Chrome



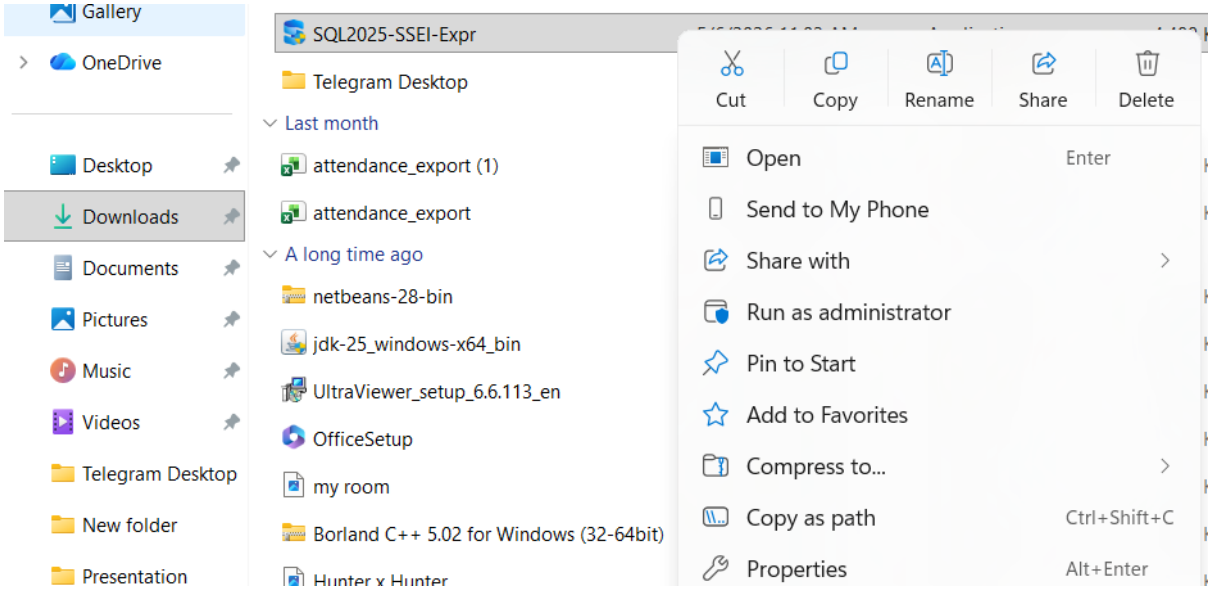
- ជំហានទី២ : ចុចលើ "SQL Server Download" ដែលជាគេហទំព័រផ្លូវការ របស់ Microsoft



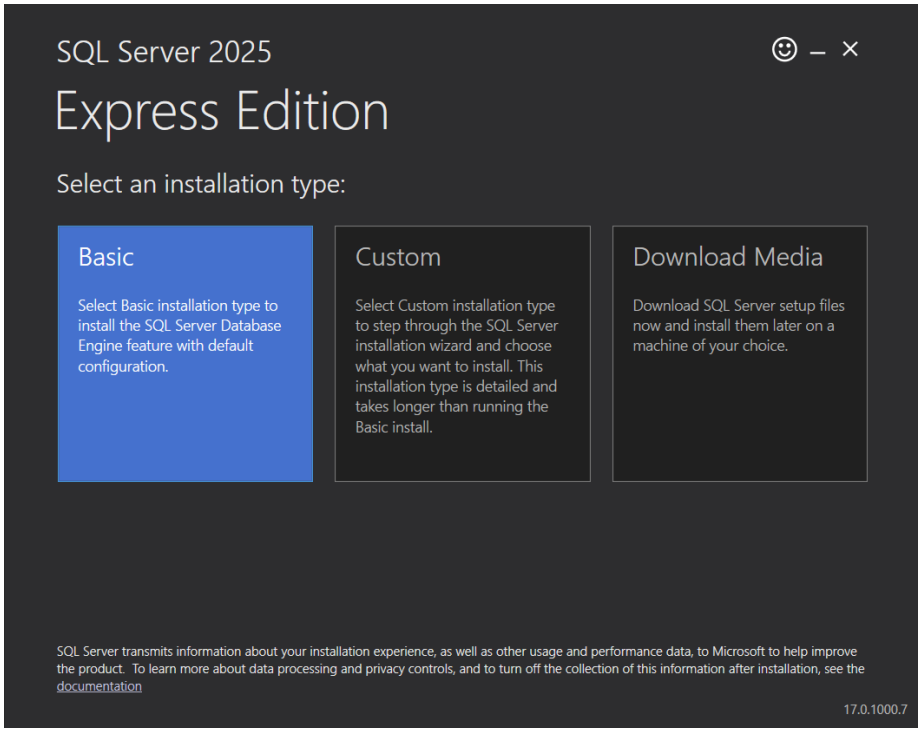
- ជំហានទី៣ : ចុច "Download now" នៅលើ Express សម្រាប់កម្មវិធីកម្រិតមធ្យមនៅលើ Computer



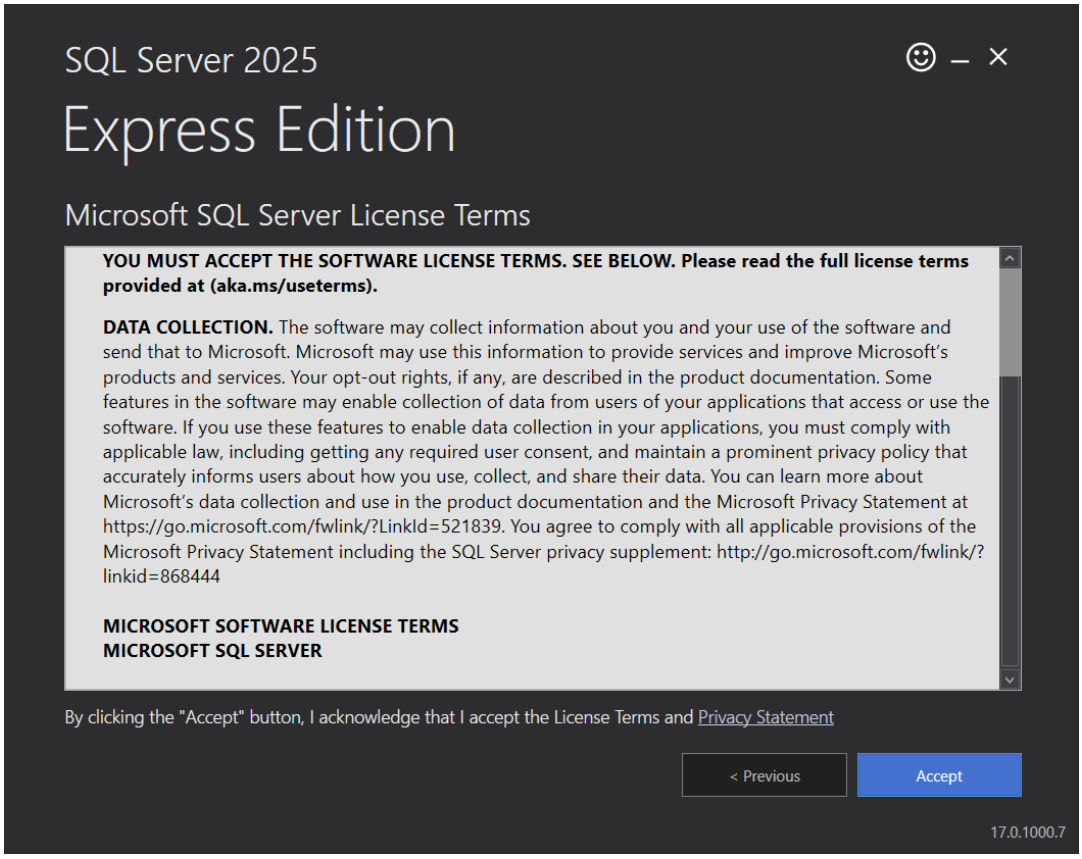
- ជំហានទី៤ : ចុច Open កម្មវិធី "**SQL2025-SSEI_Expr.exe**" នៅក្នុង Downloads folder



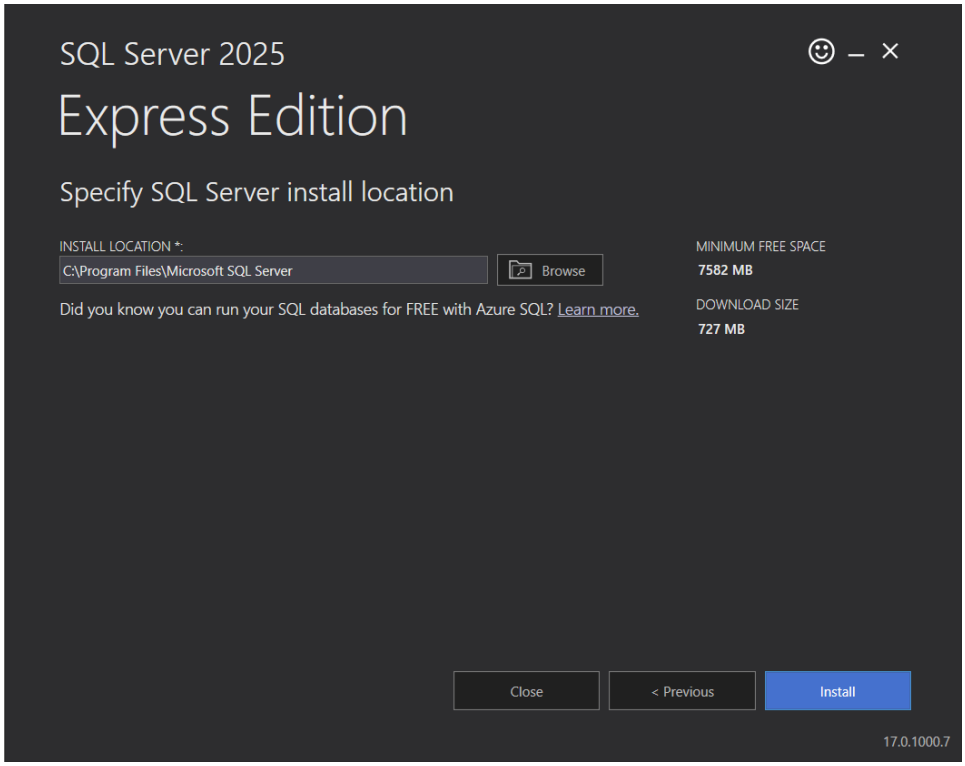
- ជំហានទី៥ : ជ្រើសរើសយក Basic



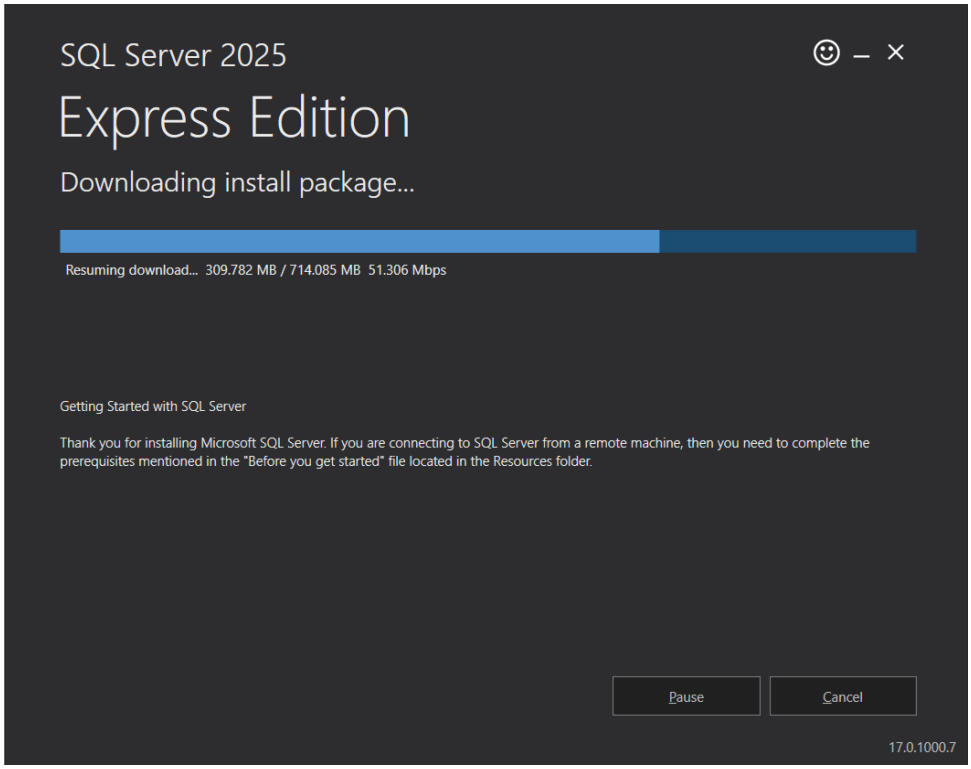
- ជំហានទី៦ : ចុច Accept ដើម្បីទទួលយកលក្ខខណ្ឌក្នុងការប្រើប្រាស់



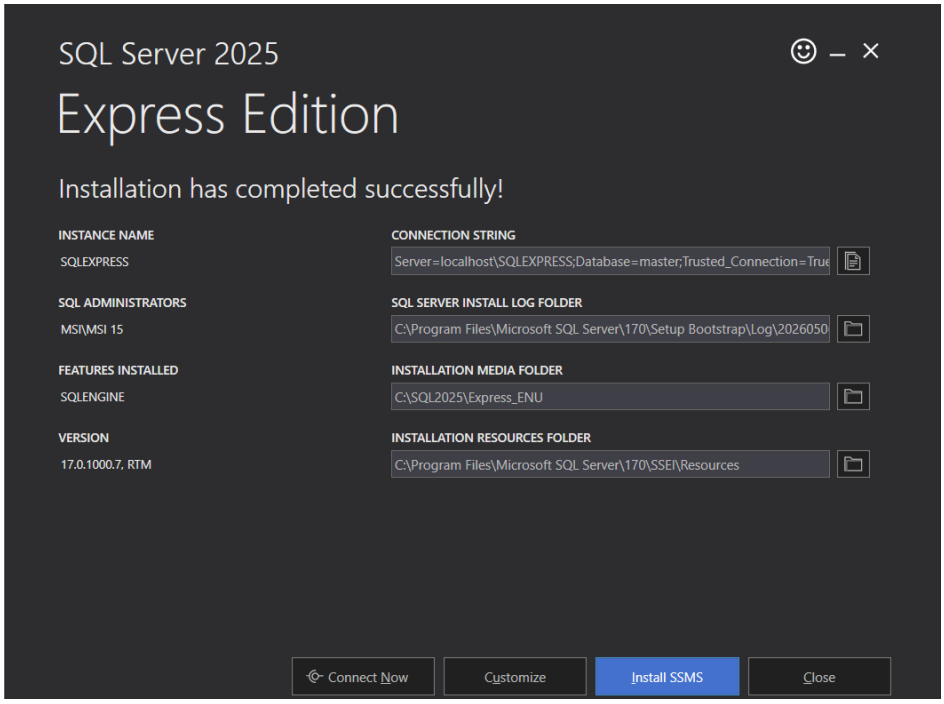
- ជំហានទី៧ : ជ្រើសរើសទីតាំង រួចចុច Installដើម្បីដំឡើង



- ជំហានទី៨ : Installer ធ្វើការទាញយក Install package



- ជំហានទី៩ : ឃើញពាក្យ "Installation has completed successfully" បានន័យថាដំឡើងបានជោគជ័យ ។ ចុច Install SSMS ដើម្បីដំឡើង SQL Server Management Studio



- ជំហានទី១០ : ចុច "Download SQL Server Management Studio 22 Installer"

Install SQL Server Management Studio

Summarize this article for me

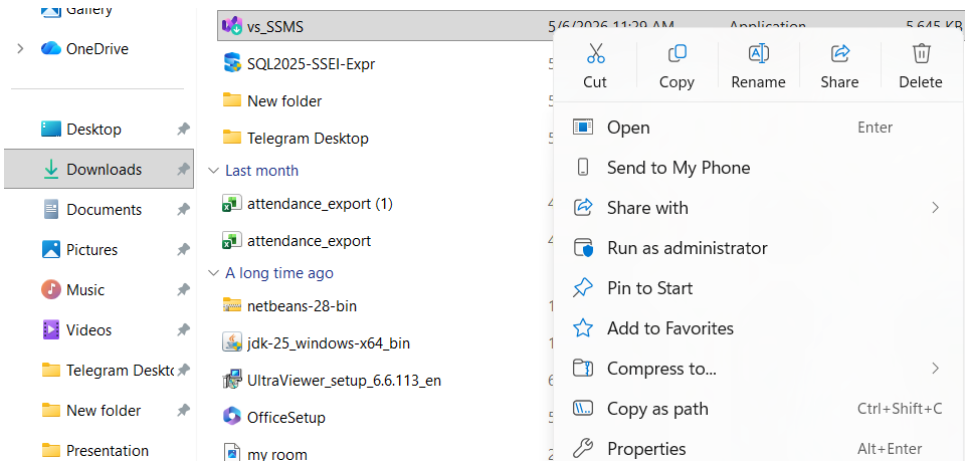
Applies to: SQL Server Azure SQL Database Azure SQL Managed Instance Azure Synapse Analytics Analytics Platform System (PDW) SQL analytics endpoint in Microsoft Fabric Warehouse in Microsoft Fabric SQL database in Microsoft Fabric

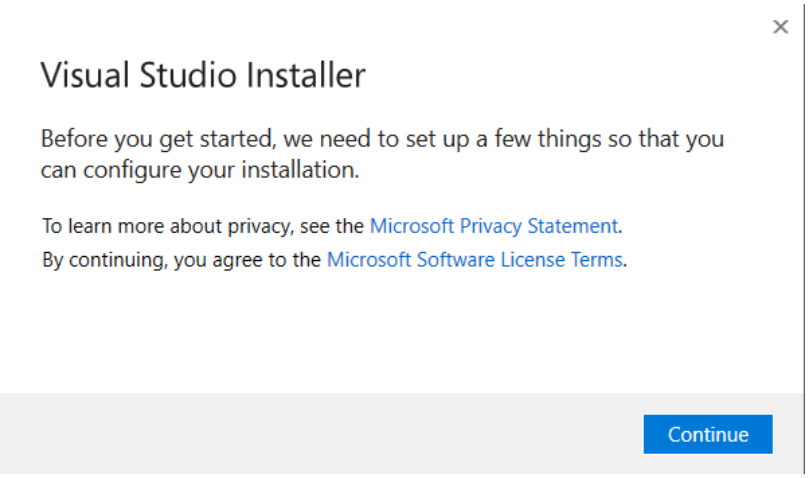
SQL Server Management Studio (SSMS) is an integrated environment for managing any SQL infrastructure. Use SSMS to access, configure, manage, administer, and develop all components of your Database Engine.

This article describes how to install the latest version of SSMS.

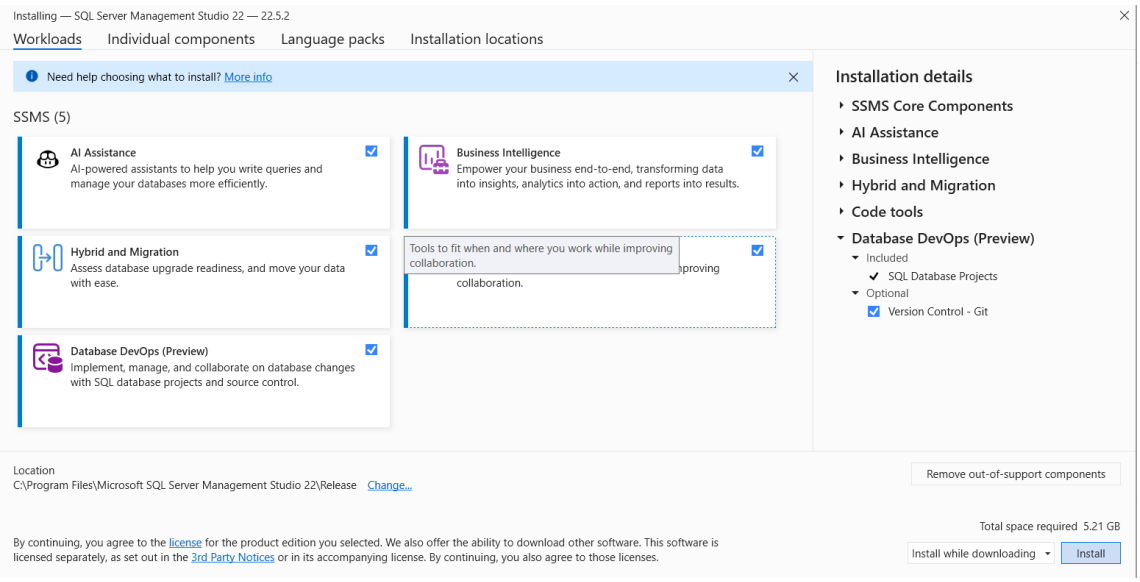
Download SQL Server Management Studio 22 installer

- ធ្វើការទាញយក (vs_SSMS.exe)

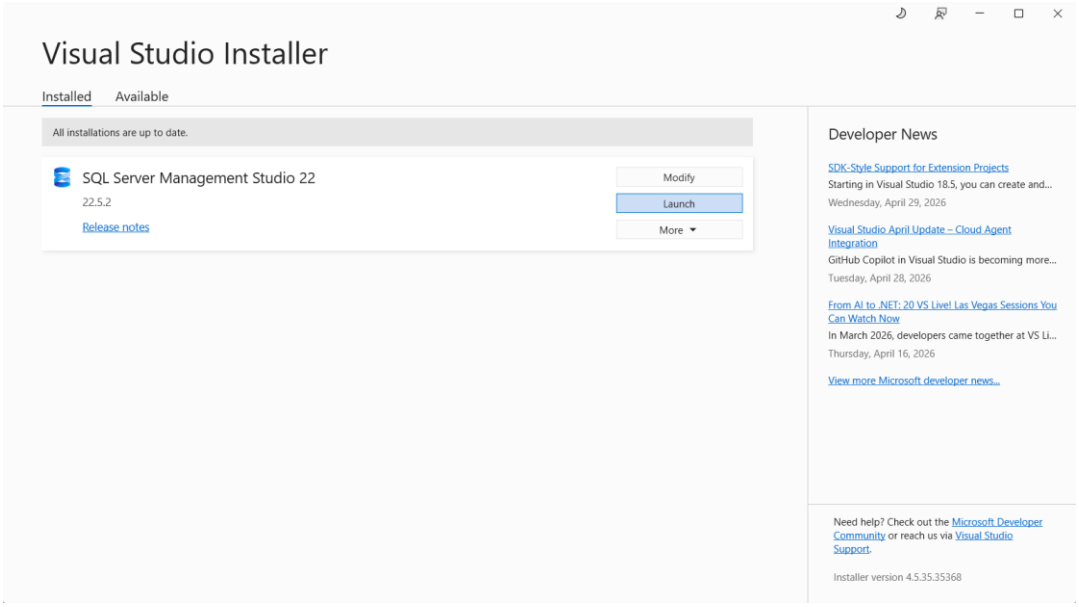




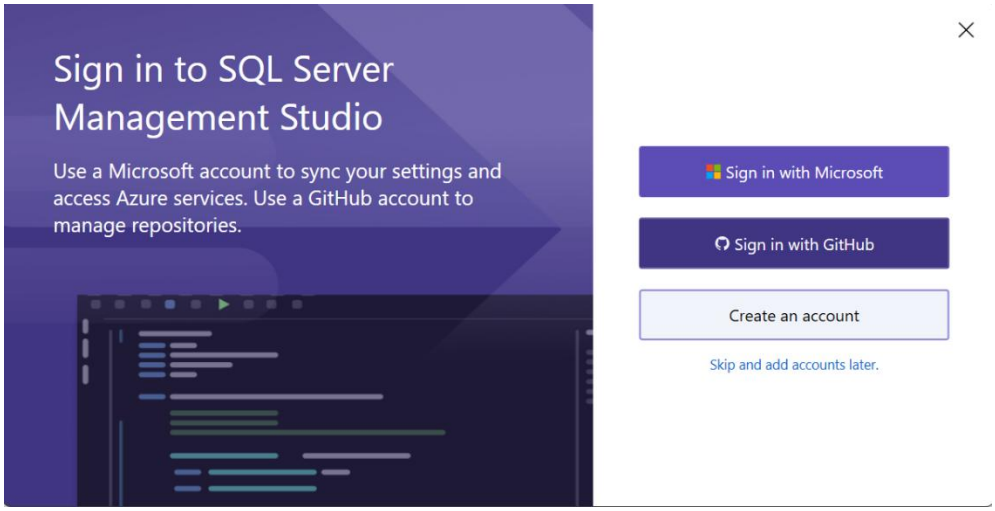
- ជ្រើសរើសមុខងារទាំងអស់ រួចចុច Install



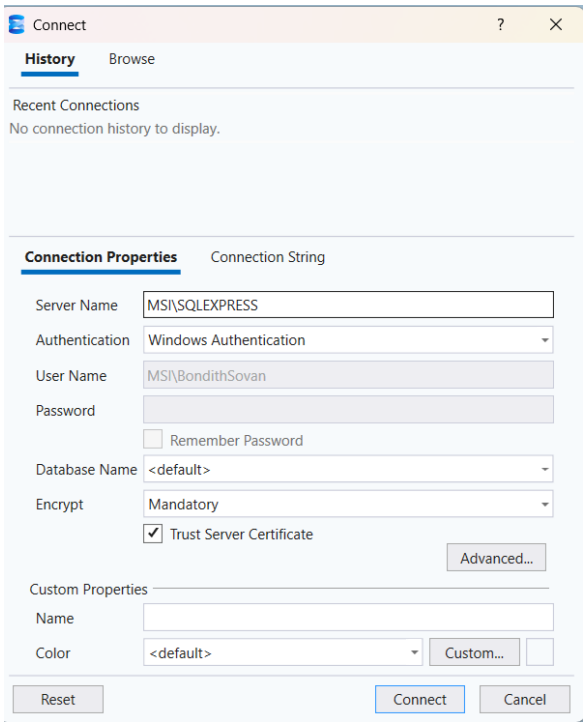
- Launch SSMS ក្រោយដំឡើងរួចដើម្បីភ្ជាប់ទៅកាន់ Instance SQL Server

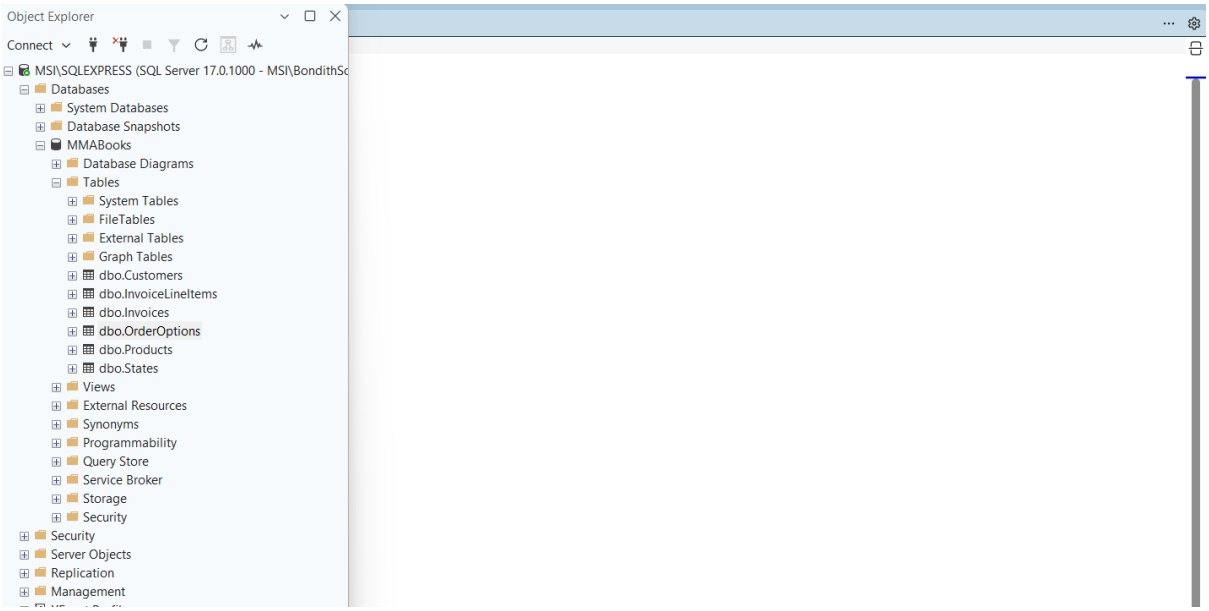


- ចុងក្រោយធ្វើការ Login ជាការស្រេច

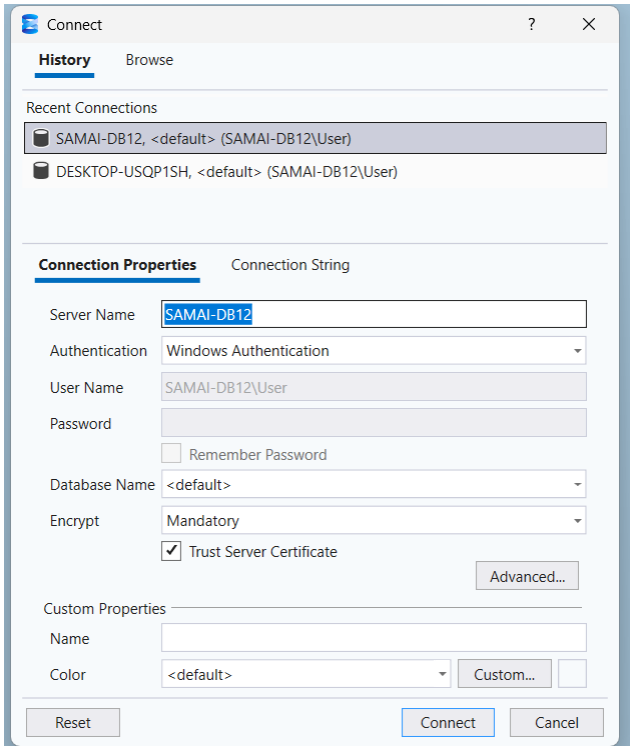


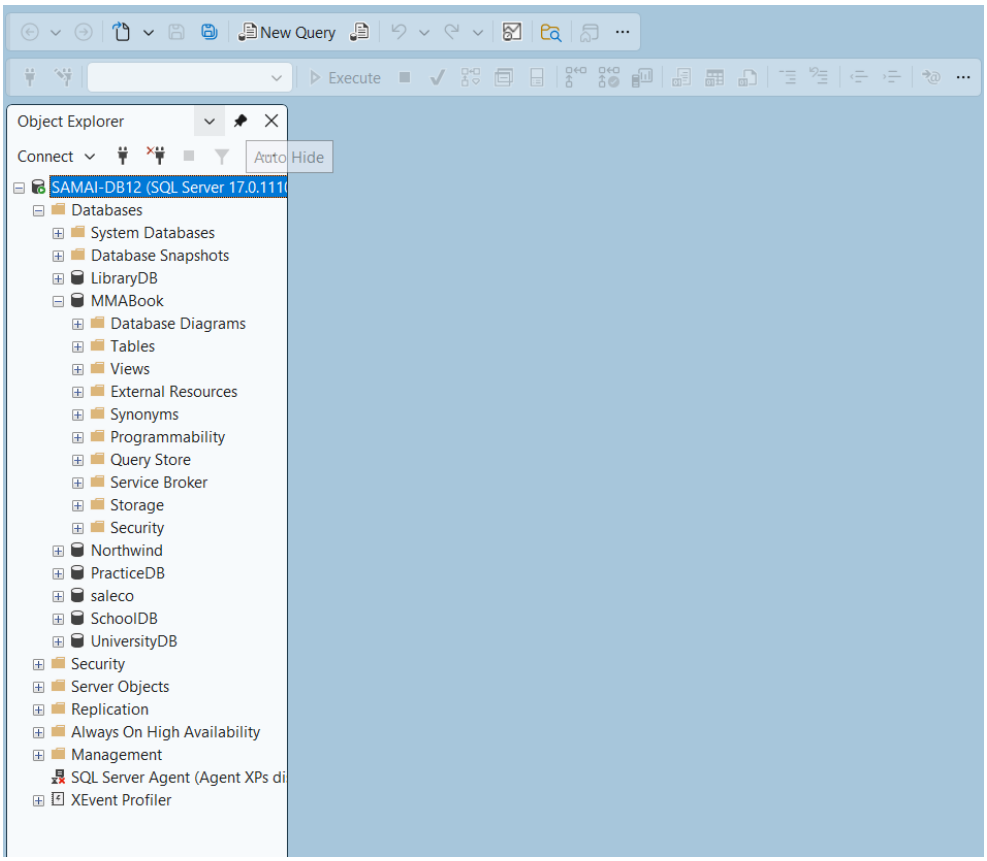
- **ការ** Login DBMS **របស់** *សុវណ្ណ បណ្ឌិត*



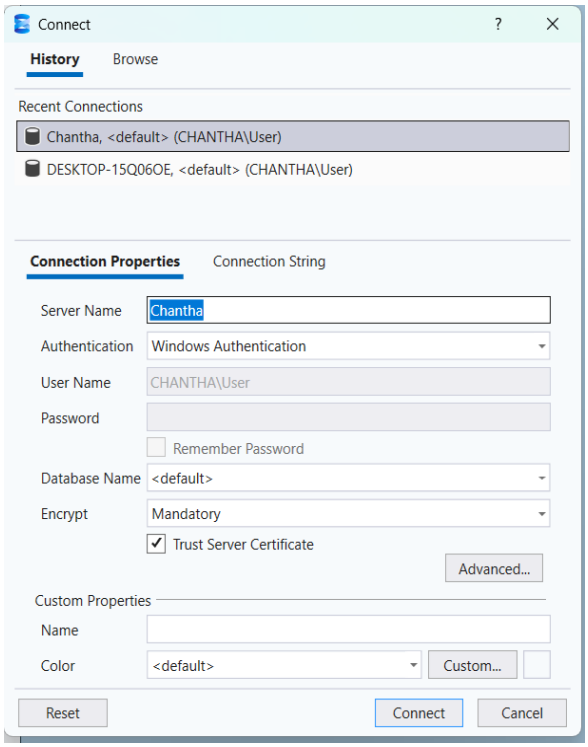


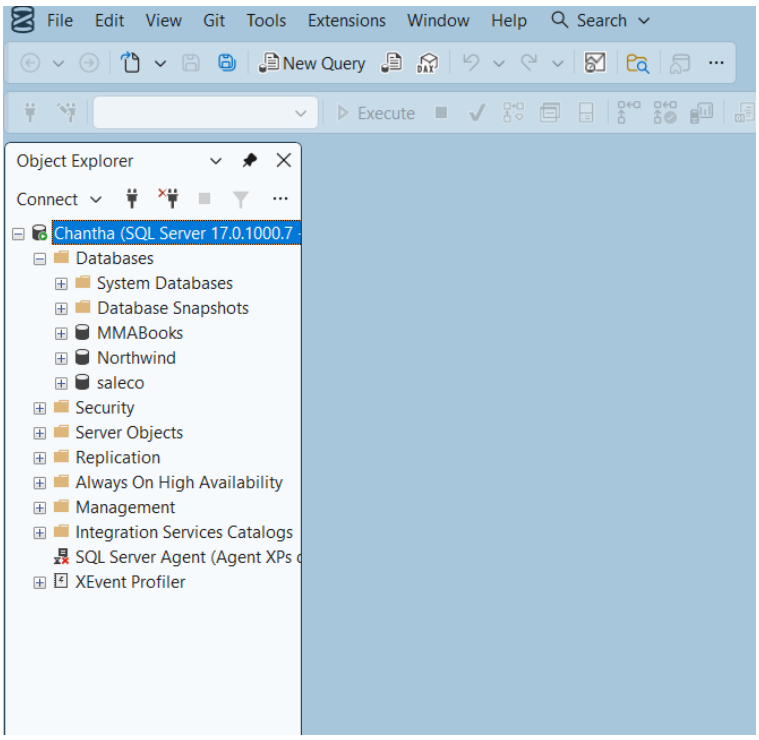
➕ ការ Login DBMS របស់ ខ្ញុំ សាម៉ែ





- **ការ** Login DBMS **របស់** ** កាន ចន្ធា**





III. ការអនុវត្ត

៣.១.Data Manipulation Language (DML)

DML គឺជាផ្នែកមួយនៃភាសា SQL ដែលត្រូវបានប្រើសម្រាប់ គ្រប់គ្រង និងកែប្រែទិន្នន័យ នៅក្នុងតារាង (Table) នៃមូលដ្ឋានទិន្នន័យ។ DML មិនមានតួនាទីបង្កើត ឬលុបរចនាសម្ព័ន្ធ Database ទេ ប៉ុន្តែវាផ្តោតលើការប្រើប្រាស់ទិន្នន័យដែលមានស្រាប់។

ប្រភេទនៃពាក្យបញ្ជា **DML** មានពាក្យបញ្ជាសំខាន់ៗ **4** ប្រភេទដូចខាងក្រោម៖

៣.១.១.SELECT & FROM

SELECT គឺជាពាក្យបញ្ជាជាមូលដ្ឋាន និងសំខាន់បំផុតនៅក្នុងភាសា SQL (ស្ថិតក្នុងក្រុម DML/DQL) ដែលប្រើប្រាស់សម្រាប់ទាញយកទិន្នន័យ (Data Retrieval) ពីជួរឈរ (Columns) ណាមួយ ឬច្រើននៃតារាងទិន្នន័យមកបង្ហាញនៅលើអេក្រង់ ទៅតាមលក្ខខណ្ឌដែលអ្នកប្រើប្រាស់ចង់បាន។

- **ការបង្ហាញទិន្នន័យដោយប្រើDatabase MMABooks** សម្រាប់ទាញយកទិន្នន័យដែលបានរក្សាទុកមកបង្ហាញ៖

```
Syntax: SELECT  *
FROM Customers;
```

Results | Messages

	CustomerID	Name	Address	City	State	ZipCode
1	1	Molunguri, A	1108 Johanna Bay Drive	Birmingham	AL	35216-6909
2	2	Muhinyi, Mauda	1420 North Charles Street	New York	NY	10044
3	3	Antony, Abdul	1109 Powderhorn Drive	Fayetteville	NC	28314
4	4	Smith, Ahmad	2509 South Crescent	Savanna	IL	61074
5	5	Chen, Rabi	2273 N. Essex Ln	Murfreesboro	TN	37130
6	6	Nuckols, Heather	150 Hayes St	Elgin	IL	60120
7	7	Lutonsky, Christopher	293 Old Holcomb Bridge Way	Woodland Hills	CA	91365
8	8	Reddy, Aj	P.O. Box 9802	Plano	TX	75025-0587
9	9	Rascano, Darrell	16 Remington Dr. E	Rancho Cordova	CA	95760
10	10	Johnson, Ajith	2200 Old Germantown Road	McGregor	CA	55555
11	11	Patwardhan, John	2512 Shades Crest Rd	Bethel Park	PA	15102
12	12	Swenson, Vin	102 Forest Drive	Albany	OR	97321
13	13	Nichols, Nadine	7403 Twin Brooks Blvd	Rogers	AR	72756
14	14	Guthrie, Rawle	3011 Belden Street	Phoenix	AZ	85023
15	15	Nguyen, Allison	4100 Coca-Cola Plaza	Memphis	TN	38118

HengRotana\SQLEXPRESS (17.0... | HENGROTANA\ASUS (56) | MMABooks | 00:00:00 | Row: 1, Col: 1 | 696 rows

❖ *ករណីបើចង់* Select *តែ* Column *ជាក់លាក់៖*

SELECT Name, Address, City

FROM [dbo].[Customers];

	Name	Address	City
1	Molunguri, A	1108 Johanna Bay Drive	Birmingham
2	Muhinyi, Mauda	1420 North Charles Street	New York
3	Antony, Abdul	1109 Powderhorn Drive	Fayetteville
4	Smith, Ahmad	2509 South Crescent	Savanna
5	Chen, Rabi	2273 N. Essex Ln	Murfreesboro
6	Nuckols, Heather	150 Hayes St	Elgin
7	Lutonsky, Christopher	293 Old Holcomb Bridge Way	Woodland Hills
8	Reddy, Aj	P.O. Box 9802	Plano
9	Rascano, Darrell	16 Remington Dr. E	Rancho Cordova
10	Johnson, Ajith	2200 Old Germantown Road	McGregor
11	Patwardhan, John	2512 Shades Crest Rd	Bethel Park
12	Swenson, Vin	102 Forest Drive	Albany
13	Nichols, Nadine	7403 Twin Brooks Blvd	Rogers
14	Guthrie, Rawle	3011 Belden Street	Phoenix
15	Nguyen, Allison	4100 Coca-Cola Plaza	Memphis
16	Kandregula, Howard	534 Radar Road	Oklahoma City
17	Yudkevich, Ed	1701 E. 12Th Street	Olathe

លទ្ធផលបង្ហាញ៖ វានឹងបង្ហាញតែជួរឈរ Name, Address *និង* City *ពីតារាង* Customers *ប៉ុណ្ណោះ*

❖ *ករណីបើចង់ភ្ជាប់* Column *ចូលគ្នាឱ្យទៅជា* Column *រួមមួយ* (Concatenation)៖

```
SELECT Address + ',' + City + ',' + ZipCode AS FullAddress

FROM [dbo].[Customers];
```

Results | Messages

	FullAddress
1	1108 Johanna Bay Drive,Birmingham,35216-6909
2	1420 North Charles Street,New York,10044
3	1109 Powderhorn Drive,Fayetteville,28314
4	2509 South Crescent,Savanna,61074
5	2273 N. Essex Ln,Murfreesboro,37130
6	150 Hayes St,Elgin,60120
7	293 Old Holcomb Bridge Way,Woodland Hills,91365
8	P.O. Box 9802,Plano,75025-0587
9	16 Remington Dr. E,Rancho Cordova,95760
10	2200 Old Germantown Road,McGregor,55555
11	2512 Shades Crest Rd,Bethel Park,15102
12	102 Forest Drive,Albany,97321
13	7403 Twin Brooks Blvd,Rogers,72756
14	3011 Belden Street,Phoenix,85023
15	4100 Coca-Cola Plaza,Memphis,38118
16	534 Radar Road,Oklahoma City,73132
17	1701 E. 12Th Street,Olathe,66062

ក. FROM Clause

FROM Clause គឺជាផ្នែកមួយដ៏សំខាន់នៅក្នុង SQL Query ដែលប្រើប្រាស់សម្រាប់បញ្ជាក់ពីឈ្មោះតារាង (Table) ឬប្រភពទិន្នន័យដែលយើងចង់ទាញយកទិន្នន័យមកធ្វើការសួរ (Query)។ បន្ថែមពីលើនេះ នៅក្នុង FROM Clause យើងក៏អាចធ្វើការភ្ជាប់តារាងពីរ ឬច្រើនបញ្ចូលគ្នាបានផងដែរ។

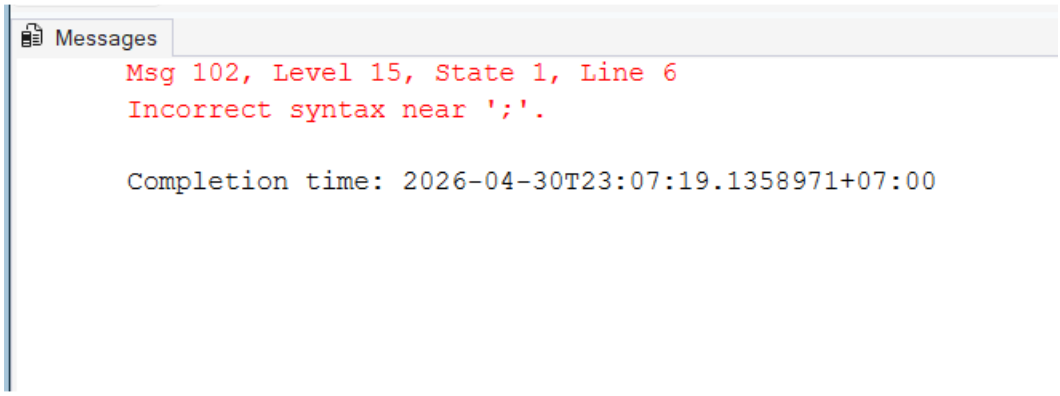
- NATURAL JOIN: ជាប្រភេទនៃការភ្ជាប់តារាង (Join) នៅក្នុង SQL ដែលធ្វើការភ្ជាប់តារាងពីរកំពូទ័រដោយស្វ័យប្រវត្តិ ផ្អែកទៅលើជួរឈរ (Column) ណាដែលមានឈ្មោះដូចគ្នា និងមានប្រភេទទិន្នន័យ (Data Type) ដូចគ្នា។ វានឹងទាញយកតែទិន្នន័យណាដែលត្រូវគ្នា (Matching values) ហើយវានឹងលុបជួរឈរដែលស្ទួនគ្នាចោលនៅពេលបង្ហាញលទ្ធផល។

Syntax: SELECT column-list

FROM table1

NATURAL JOIN table2;

ឧទហរណ៍៖ ការសរសេរ SQL Query មានការប្រើ NATURAL JOIN ដើម្បីភ្ជាប់tables Customers និង Invoices។



- លទ្ធផលកូដ NATURAL JOIN ហើយប្រព័ន្ធលោត Error ពណ៌ក្រហមចេញ ថា៖ "Incorrect syntax near ';'."
- <u>ចំណាំ</u> នៅក្នុងការអនុវត្តជាក់ស្តែងនៅលើកម្មវិធី Microsoft SQL Server (RDBMS) ដែលអ្នកកំពុងប្រើធ្វើ Lab នេះ ពាក្យបញ្ជា NATURAL JOIN មិនត្រូវបានគាំទ្រឡើយ (Not Supported)។

ដើម្បីឱ្យអ្នកមានលទ្ធផល (Result) ផ្ទៀងផ្ទាត់ពិតប្រាកដយកទៅដាក់ក្នុងរបាយការណ៍ថ្នាក់ គេត្រូវកែមកប្រើ INNER JOIN វិញ។

```
SELECT Customers.CustomerID, Name, Address, InvoiceID, InvoiceDate,
ProductTotal
FROM [dbo].Customers
INNER JOIN [dbo].Invoices
ON [dbo].Customers.CustomerID = [dbo].Invoices.CustomerID;
```

Results | Messages

	CustomerID	Name	Address	InvoiceID	InvoiceDate	ProductTotal
1	10	Johnson, Ajith	2200 Old Germantown Road	47	2013-01-13 00:00:00.000	236.50
2	10	Johnson, Ajith	2200 Old Germantown Road	58	2013-01-19 00:00:00.000	56.50
3	10	Johnson, Ajith	2200 Old Germantown Road	66	2013-01-24 00:00:00.000	56.50
4	10	Johnson, Ajith	2200 Old Germantown Road	68	2013-01-26 00:00:00.000	56.50
5	10	Johnson, Ajith	2200 Old Germantown Road	101	2013-02-09 00:00:00.000	56.50
6	11	Patwardhan, John	2512 Shades Crest Rd	69	2013-01-26 00:00:00.000	42.00
7	11	Patwardhan, John	2512 Shades Crest Rd	27	2013-01-13 00:00:00.000	56.50
8	11	Patwardhan, John	2512 Shades Crest Rd	31	2013-01-13 00:00:00.000	56.50
9	12	Swenson, Vin	102 Forest Drive	28	2013-01-13 00:00:00.000	56.50
10	13	Nichols, Nadine	7403 Twin Brooks Blvd	29	2013-01-13 00:00:00.000	56.50
11	15	Nguyen, Allison	4100 Coca-Cola Plaza	30	2013-01-13 00:00:00.000	56.50
12	15	Nguyen, Allison	4100 Coca-Cola Plaza	60	2013-01-19 00:00:00.000	56.50
13	15	Nguyen, Allison	4100 Coca-Cola Plaza	62	2013-01-19 00:00:00.000	56.50
14	20	Chamberland, Sarah	1942 S. Gaydon Avenue	50	2013-01-14 00:00:00.000	56.50
15	20	Chamberland, Sarah	1942 S. Gaydon Avenue	46	2013-01-13 00:00:00.000	215.50
16	20	Chamberland, Sarah	1942 S. Gaydon Avenue	18	2013-01-13 00:00:00.000	155.00
17	25	De la fuente, Cathy	2213 Abbotsford Dr	48	2013-01-14 00:00:00.000	56.50
18	25	De la fuente, Cathy	2213 Abbotsford Dr	104	2013-02-22 11:04:35.060	56.50
19	33	Lair, Andrew	28 Marblestone Lane	33	2013-01-13 00:00:00.000	163.00
20	35	Morgan, Robert	48289 Fremont Blvd	26	2013-01-13 00:00:00.000	56.50
21	40	Roby, Chris	182 Greylock Parkway	32	2013-01-13 00:00:00.000	113.00
22	50	Hutcheson, Larry	2902 Cannons Lane	59	2013-01-19 00:00:00.000	565.00
23	88	Monse, Charles	5933 6Th Street	57	2013-01-14 00:00:00.000	56.50
24	125	Chism, Leslie	123 Main St	106	2013-02-23 10:26:25.687	169.50

Query executed successfully. SAN-PISETH\SQLEXPRESS (1

- OUTER JOIN គឺជាពាក្យបញ្ជាសម្រាប់តភ្ជាប់តារាងដែលពង្រីកសមត្ថភាពបន្ថែមពី INNER JOIN ដោយវាមានតួនាទីទាញយកទិន្នន័យដែលមានលក្ខខណ្ឌផ្គូរផ្គងដូចគ្នាផង និងទាញយកទិន្នន័យដែលមិនមានគូផ្គូរផ្គង (Unmatched rows) ពីតារាងម្ខាងទៀតមកបង្ហាញផងដែរ ដោយបំពេញតម្លៃដែលខ្វះនោះដោយ NULL។

OUTER JOIN ត្រូវបានបែងចែកជា ៣ ប្រភេទចម្បងៗ៖

- LEFT OUTER JOIN: ទាញយកទិន្នន័យទាំងអស់ពីតារាងខាងឆ្វេង ទោះបីជាមិនមានគូផ្គូរផ្គងនៅក្នុងតារាងខាងស្តាំក៏ដោយ។
- RIGHT OUTER JOIN: ទាញយកទិន្នន័យទាំងអស់ពីតារាងខាងស្តាំ ទោះបីជាមិនមានគូផ្គូរផ្គងនៅក្នុងតារាងខាងឆ្វេងក៏ដោយ។
- FULL OUTER JOIN: ទាញយកទិន្នន័យទាំងអស់ពីតារាងទាំងសងខាង មិនចាត់មានគូផ្គូរផ្គង ឬអត់នោះឡើយ។
- ទម្រង់ទូទៅ

  SELECT Column-list

  FROM table1

  LEFT [OUTER] JOIN table2 ON join-condition

Ex ដើម្បីបង្ហាញទិន្នន័យcolumn: CustomerID, InvoiceID, InvoiceDate សម្រាប់អតិជនទាំងអស់រួមបញ្ចូលគ្នា

```
SELECT [dbo].Customers.CustomerID, InvoiceID, InvoiceDate
FROM [dbo].Customers
LEFT JOIN [dbo].Invoices ON [dbo].Invoices.CustomerID=
[dbo].Customers.CustomerID;
```

	CustomerID	InvoiceID	InvoiceDate
1	1	NULL	NULL
2	2	NULL	NULL
3	3	NULL	NULL
4	4	NULL	NULL
5	5	NULL	NULL
6	6	NULL	NULL
7	7	NULL	NULL
8	8	NULL	NULL
9	9	NULL	NULL
10	10	47	2013-01-13 00:00:00.000
11	10	58	2013-01-19 00:00:00.000
12	10	66	2013-01-24 00:00:00.000
13	10	68	2013-01-26 00:00:00.000

Query executed successfully.

- JOIN USING គឺជាទម្រង់នៃការភ្ជាប់តារាងមួយប្រភេទ ដែលត្រូវបានប្រើប្រាស់នៅពេលដែលតារាងទាំងពីរមានឈ្មោះជួរឈរដូចគ្នា (Common Column Name) ។ វាមានលក្ខណៈស្រដៀងទៅនឹង NATURAL JOIN ដែរ ប៉ុន្តែវាមានភាព

ច្បាស់លាស់ជាង (More Explicit) ដោយសារតែយើងជាអ្នកកំណត់ឈ្មោះជួរឈរ នៅក្នុងវង់ក្រចកដោយផ្ទាល់។

Syntax: SELECT column_list

FROM table1

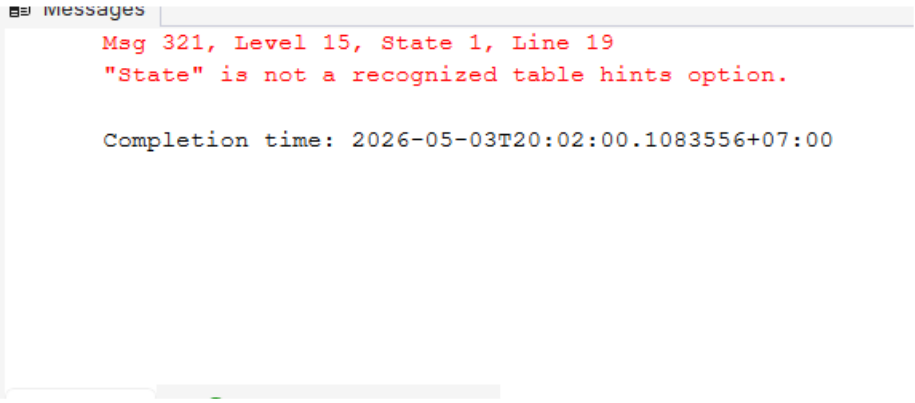
JOIN table2 USING (column_name);

Ex: ឧទហរណ៍៖ នេះគឺជាការសរសេរ query ដើម្បីទាញយកទិន្នន័យ table ពីCustomers ដោយប្រើStates JOIN តាមរយៈ JOIN USING

Syntax: SELECT CustomerID, Name, Address, StateCode, StateName

FROM [dbo].Customers

JOIN [dbo].States USING (State);



រូបភាពបង្ហាញពីការប្រើប្រាស់USING ដែលធ្វើឲមានការError

ចំណាំ ដូចគ្នានឹង NATURAL JOIN ដែរ ពាក្យបញ្ជា USING មិនត្រូវបានគាំទ្រនៅក្នុងកម្មវិធី Microsoft SQL Server ឡើយ។ នៅក្នុងការអនុវត្តជាក់ស្តែង គេត្រូវតែប្តូរមកប្រើប្រាស់ពាក្យបញ្ជា INNER JOIN ... ON ជំនួសវិញ។

- JOIN ON: គឺជាពាក្យបញ្ជានៅក្នុង SQL ដែលប្រើសម្រាប់ភ្ជាប់ទិន្នន័យពីតារាងពីរ ឬច្រើន ផ្អែកលើលក្ខខណ្ឌជាក់លាក់មួយ។ ពាក្យ JOIN ប្រើសម្រាប់ភ្ជាប់តារាង ខណៈពេលដែល ON ប្រើសម្រាប់កំណត់លក្ខខណ្ឌក្នុងការផ្គូផ្គង Column រវាងតារាងទាំងនោះ។

❖ ទម្រង់ទូទៅ (Syntax)៖SELECT column_list

FROM table1

JOIN table2 ON join_condition;

ឧទាហរណ៍៖ ទាញយកព័ត៌មានផលិតផល និងព័ត៌មានលម្អិតនៃវិក្កយបត្រ (Invoice) ដោយភ្ជាប់តារាង Products និង InvoiceLineItems តាមរយៈ ProductCode ៖

SELECT InvoiceID, Products.ProductCode, Description, Quantity

FROM Products

JOIN InvoiceLineItems ON Products.ProductCode =

InvoiceLineItems.ProductCode;

	InvoiceID	ProductCode	Description	Quantity
1	18	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	1
2	18	DB1R	DB2 for the COBOL Programmer, Part 1 (2nd Edition)	1
3	18	VB10	Murach's Visual Basic 2010	1
4	23	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	1
5	26	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	1
6	27	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	1
7	28	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	1
8	29	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	1
9	30	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	1
10	31	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	1
11	32	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	1
12	32	VB10	Murach's Visual Basic 2010	1
13	33	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	1
14	33	CRFC	Murach's CICS Desk Reference	1
15	33	VB10	Murach's Visual Basic 2010	1
16	41	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	1
17	42	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	1

Query executed successfully.

- LEFT OUTER JOIN (ឬ LEFT JOIN)៖ ជាការភ្ជាប់តារាងដែលទាញយកទិន្នន័យទាំងអស់ពីតារាងខាងឆ្វេង (Left Table) ទោះបីជាទិន្នន័យនោះមិនមានតម្លៃផ្គូផ្គងជាមួយតារាងខាងស្តាំ (Right Table) ក៏ដោយ។ ចំពោះទិន្នន័យតារាងខាងស្តាំដែលមិនត្រូវគ្នា វានឹងបង្ហាញតម្លៃជា NULL។

❖ ទម្រង់ទូទៅ (Syntax)៖

```
SELECT column_list

FROM table1

LEFT [OUTER] JOIN table2 ON join_condition;
```

ឧទាហរណ៍៖ ខាងក្រោមនេះជាការ Query ដើម្បីបង្ហាញទិន្នន័យជួរឈរ CustomerID, InvoiceID, InvoiceDate សម្រាប់អតិថិជនទាំងអស់ (ទោះបីជាអតិថិជនខ្លះមិនទាន់មានវិក្កយបត្រក៏ដោយ)៖

```
SELECT Customers.CustomerID, InvoiceID, InvoiceDate

FROM Customers

LEFT JOIN Invoices ON Invoices.CustomerID = Customers.CustomerID;
```

Results | Messages

	CustomerID	InvoiceID	InvoiceDate
1	1	NULL	NULL
2	2	NULL	NULL
3	3	NULL	NULL
4	4	NULL	NULL
5	5	NULL	NULL
6	6	NULL	NULL
7	7	NULL	NULL
8	8	NULL	NULL
9	9	NULL	NULL
10	10	47	2013-01-13 00:00:00.000
11	10	58	2013-01-19 00:00:00.000
12	10	66	2013-01-24 00:00:00.000
13	10	68	2013-01-26 00:00:00.000
14	10	101	2013-02-09 00:00:00.000
15	11	69	2013-01-26 00:00:00.000
16	11	27	2013-01-13 00:00:00.000
17	11	31	2013-01-13 00:00:00.000

Query executed successfully.

- RIGHT OUTER JOIN (ឬ RIGHT JOIN)៖ ជាការភ្ជាប់តារាងដែលទាញយកទិន្នន័យទាំងអស់ពីតារាងខាងស្តាំ (Right Table) ទោះបីជាទិន្នន័យនោះមិនមានតម្លៃធ្វើផ្គងជាមួយតារាងខាងឆ្វេង (Left Table) ក៏ដោយ។ ចំពោះទិន្នន័យតារាងខាងឆ្វេងដែលមិនត្រូវគ្នា វានឹងបង្ហាញតម្លៃជា NULL។

❖ ទម្រង់ទូទៅ (Syntax)៖

```
SELECT column_list

FROM table1

RIGHT [OUTER] JOIN table2 ON join_condition;
```

ឧទាហរណ៍៖ ការប្រើប្រាស់ RIGHT JOIN ផ្នែកលើ Database MMABooks (ប្រសិនបើទិន្នន័យទាំងអស់ក្នុងតារាង Invoices មានការធ្វើផ្គងជាមួយ Customers ទាំងអស់ នោះលទ្ធផលនឹងមិនបង្ហាញតម្លៃ NULL ឡើយ)៖

```
SELECT Customers.CustomerID, InvoiceID, InvoiceDate

FROM Customers
```

RIGHT JOIN Invoices ON Invoices.CustomerID = Customers.CustomerID;

Results | Messages

	CustomerID	InvoiceID	InvoiceDate
1	10	47	2013-01-13 00:00:00.000
2	10	58	2013-01-19 00:00:00.000
3	10	66	2013-01-24 00:00:00.000
4	10	68	2013-01-26 00:00:00.000
5	10	101	2013-02-09 00:00:00.000
6	11	69	2013-01-26 00:00:00.000
7	11	27	2013-01-13 00:00:00.000
8	11	31	2013-01-13 00:00:00.000
9	12	28	2013-01-13 00:00:00.000
10	13	29	2013-01-13 00:00:00.000
11	15	30	2013-01-13 00:00:00.000
12	15	60	2013-01-19 00:00:00.000
13	15	62	2013-01-19 00:00:00.000
14	20	50	2013-01-14 00:00:00.000
15	20	46	2013-01-13 00:00:00.000
16	20	18	2013-01-13 00:00:00.000
17	25	48	2013-01-14 00:00:00.000

Query executed successfully.

- FULL OUTER JOIN (ឬ FULL JOIN)៖ *គឺជាការទាញយកទិន្នន័យទាំងអស់ពីតារាងទាំងពីរ (ទាំងខាងឆ្វេង និងខាងស្តាំ) ទោះបីជាមានទិន្នន័យមួយចំនួនមិនមានតម្លៃគ្ជូគ្ជងគ្នាក៏ដោយ។ ប្រសិនបើ* Record *មួយមិនមានការគ្ជូគ្ជងនៅក្នុងតារាងម្ខាងទៀតទេ នោះជួរឈរបស់តារាងម្ខាងនោះនឹងបង្ហាញតម្លៃជា* NULL *។*

❖ *ទម្រង់ទូទៅ* (Syntax)៖

SELECT column_list

FROM table1

FULL [OUTER] JOIN table2 ON join_condition;

ឧទាហរណ៍៖ ប្រើ FULL OUTER JOIN *ដើម្បីបង្ហាញទិន្នន័យទាំងអស់ពីតារាង* Products *និង* InvoiceLineItems *(រួមទាំងផលិតផលដែលមិនទាន់មានការលក់ ឬមិនទាន់មានវិក្កយបត្រ)៖*

SELECT Products.ProductCode, Description, InvoiceID, Quantity

FROM Products

FULL OUTER JOIN InvoiceLineItems ON Products.ProductCode =

InvoiceLineItems.ProductCode;

Results | Messages

	ProductCode	Description	InvoiceID	Quantity
1	A4CS	Murach's ASP.NET 4 Web Programming with C# 2010	104	1
2	A4CS	Murach's ASP.NET 4 Web Programming with C# 2010	105	1
3	A4CS	Murach's ASP.NET 4 Web Programming with C# 2010	111	1
4	A4CS	Murach's ASP.NET 4 Web Programming with C# 2010	115	1
5	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	114	1
6	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	106	1
7	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	23	1
8	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	26	1
9	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	27	1
10	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	28	1
11	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	29	1
12	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	30	1
13	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	31	1
14	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	32	1
15	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	18	1
16	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	33	1
17	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	44	1

Query executed successfully.

ចំណាំ៖ កម្មវិធី Oracle និង MS SQL Server គឺគាំទ្រ (Support) Syntax FULL JOIN ប៉ុន្តែ MySQL មិនគាំទ្រឡើយ។

- **CROSS JOIN៖** គឺជាការភ្ជាប់តារាងពីរដោយផ្ទួនផ្គងរាល់ជួរដេក (Rows) ទាំងអស់នៃតារាងទី១ ជាមួយរាល់ជួរដេកទាំងអស់នៃតារាងទី២ ដោយគ្មានលក្ខខណ្ឌ (Condition) អ្វីទាំងអស់។ លទ្ធផលដែលទទួលបានហៅថា **Cartesian Product** ដែលមានចំនួនជួរដេកសរុបស្មើនឹង (ចំនួនជួរដេកនៃតារាងទី១ គុណនឹង ចំនួនជួរដេកនៃតារាងទី២)។

❖ ទម្រង់ទូទៅ (Syntax)៖SELECT column_list

FROM table1

CROSS JOIN table2;

ឧទាហរណ៍៖ ប្រើ CROSS JOIN ដើម្បីភ្ជាប់តារាង Customers និង Invoices៖

SELECT * FROM Customers

CROSS JOIN Invoices;

Results | Messages

	CustomerID	Name	Address	City	State	ZipCode	InvoiceID	CustomerID	InvoiceDate	ProductTotal	SalesTax	Shipping	InvoiceTotal
1	1	Molunguri, A	1108 Johanna Bay Drive	Birmingham	AL	35216-6909	18	20	2013-01-13 00:00:00.000	155.00	11.625	6.25	172.875
2	2	Muhinyi, Mauda	1420 North Charles Street	New York	NY	10044	18	20	2013-01-13 00:00:00.000	155.00	11.625	6.25	172.875
3	3	Antony, Abdul	1109 Powderhorn Drive	Fayetteville	NC	28314	18	20	2013-01-13 00:00:00.000	155.00	11.625	6.25	172.875
4	4	Smith, Ahmad	2509 South Crescent	Savanna	IL	61074	18	20	2013-01-13 00:00:00.000	155.00	11.625	6.25	172.875
5	5	Chen, Rabi	2273 N. Essex Ln	Murfreesboro	TN	37130	18	20	2013-01-13 00:00:00.000	155.00	11.625	6.25	172.875
6	6	Nuckols, Heather	150 Hayes St	Elgin	IL	60120	18	20	2013-01-13 00:00:00.000	155.00	11.625	6.25	172.875
7	7	Lutonsky, Christopher	293 Old Holcomb Bridge Way	Woodland Hills	CA	91365	18	20	2013-01-13 00:00:00.000	155.00	11.625	6.25	172.875
8	8	Reddy, Aj	P.O. Box 9802	Plano	TX	75025-0587	18	20	2013-01-13 00:00:00.000	155.00	11.625	6.25	172.875
9	9	Rascano, Darrell	16 Remington Dr. E	Rancho Cordova	CA	95760	18	20	2013-01-13 00:00:00.000	155.00	11.625	6.25	172.875
10	10	Johnson, Ajith	2200 Old Germantown Road	McGregor	CA	55555	18	20	2013-01-13 00:00:00.000	155.00	11.625	6.25	172.875
11	11	Patwardhan, John	2512 Shades Crest Rd	Bethel Park	PA	15102	18	20	2013-01-13 00:00:00.000	155.00	11.625	6.25	172.875
12	12	Swenson, Vin	102 Forest Drive	Albany	OR	97321	18	20	2013-01-13 00:00:00.000	155.00	11.625	6.25	172.875
13	13	Nichols, Nadine	7403 Twin Brooks Blvd	Rogers	AR	72756	18	20	2013-01-13 00:00:00.000	155.00	11.625	6.25	172.875
14	14	Guthrie, Rawle	3011 Belden Street	Phoenix	AZ	85023	18	20	2013-01-13 00:00:00.000	155.00	11.625	6.25	172.875
15	15	Nguyen, Allison	4100 Coca-Cola Plaza	Memphis	TN	38118	18	20	2013-01-13 00:00:00.000	155.00	11.625	6.25	172.875
16	16	Kandregula, Howard	534 Radar Road	Oklahoma City	OK	73132	18	20	2013-01-13 00:00:00.000	155.00	11.625	6.25	172.875
17	17	Yudkevich, Ed	1701 E. 12Th Street	Olathe	KS	66062	18	20	2013-01-13 00:00:00.000	155.00	11.625	6.25	172.875

Query executed successfully. SAMAI-DB12 (17.0 RTM) SAMAI-DB12\User (88) MMA

- Join Table with an Alias (ការប្រើឈ្មោះជំនួសតារាង)៖ Alias ត្រូវបានប្រើប្រាស់ដើម្បីកំណត់ឈ្មោះខ្លី ឬឈ្មោះជំនួសសម្រាប់តារាងមួយដែលជាប្រភពទិន្នន័យ។ ការប្រើ Alias ជួយឱ្យការសរសេរ SQL Query មានភាពងាយស្រួលក្នុងការអាន និងកាត់បន្ថយការសរសេរឈ្មោះតារាងវែងៗ ជាពិសេសនៅពេលដែលយើងត្រូវភ្ជាប់តារាងច្រើន (Multiple JOINs)។

  ឧទាហរណ៍៖ កំណត់ Alias P ជំនួសតារាង Products និង L ជំនួសតារាង InvoiceLineItems៖

SELECT P.ProductCode, P.Description, L.UnitPrice, L.Quantity

FROM Products AS P

JOIN InvoiceLineItems AS L ON P.ProductCode = L.ProductCode;

	ProductCode	Description	UnitPrice	Quantity
1	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	56.50	1
2	DB1R	DB2 for the COBOL Programmer, Part 1 (2nd Edition)	42.00	1
3	VB10	Murach's Visual Basic 2010	56.50	1
4	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	56.50	1
5	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	56.50	1
6	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	56.50	1
7	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	56.50	1
8	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	56.50	1
9	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	56.50	1
10	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	56.50	1
11	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	56.50	1
12	VB10	Murach's Visual Basic 2010	56.50	1
13	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	56.50	1
14	CRFC	Murach's CICS Desk Reference	50.00	1
15	VB10	Murach's Visual Basic 2010	56.50	1
16	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	56.50	1
17	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	56.50	1

Query executed successfully.

- RECURSIVE JOIN (ឬ SELF JOIN)៖ គឺជាការភ្ជាប់តារាងមួយជាមួយនឹងខ្លួនឯង (Table joins itself) ដោយប្រើប្រាស់ Alias ផ្សេងៗគ្នា ដើម្បីបង្ហាញទំនាក់ទំនងរវាងជួរដេក (Rows) នៅក្នុងតារាងតែមួយនោះ។

❖ ទម្រង់ទូទៅ (Syntax)៖SELECT t1.column_name, t2.column_name

FROM table_name AS t1

JOIN table_name AS t2 ON t1.common_column = t2.common_column;

ឧទាហរណ៍៖ ប្រើសម្រាប់ទាញយកព័ត៌មានអំពីអតិថិជនជាគូដែលរស់នៅក្នុងរដ្ឋ (State) ដូចគ្នា៖

SELECT C1.Name AS Customer1, C2.Name AS Customer2, C1.State

FROM Customers AS C1

JOIN Customers AS C2 ON C1.State = C2.State AND C1.CustomerID <>

C2.CustomerID;

111 % 3 0

Results Messages

	Customer1	Customer2	State
1	Wieneke, Frank	Wazny, Tina	AL
2	Molunguri, A	Wazny, Tina	AL
3	Haldorai, Brent	Wazny, Tina	AL
4	Goodwin, Mark	Wazny, Tina	AL
5	Briggs, Greiner	Wazny, Tina	AL
6	Bernard, Subramaniyan	Wazny, Tina	AL
7	Larson, Heriberto	Wazny, Tina	AL
8	Truesdale, David	Wazny, Tina	AL
9	Brown, Ashwini	Wazny, Tina	AL
10	Bingham, Robert	Wazny, Tina	AL
11	Dunn, Ole	Wazny, Tina	AL
12	Wazny, Tina	Wieneke, Frank	AL
13	Molunguri, A	Wieneke, Frank	AL
14	Haldorai, Brent	Wieneke, Frank	AL
15	Goodwin, Mark	Wieneke, Frank	AL
16	Briggs, Greiner	Wieneke, Frank	AL
17	Bernard, Subramaniyan	Wieneke, Frank	AL

Query executed successfully.

២.ការប្រើប្រាស់ WHERE Clause

WHERE ត្រូវបានប្រើដើម្បីទាញយកតែទិន្នន័យណាដែលបំពេញតាមលក្ខខណ្ឌជាក់លាក់។

Syntax: SELECT column1, column2, ...

FROM table_name

WHERE condition;

ឧទាហរណ៍ទី១ (ប្រភេទអក្សរ - Text Fields)៖ ប្រសិនបើអ្នកចង់ទាញរបាយការណ៍អតិថិជនដែលរស់នៅក្នុងទីក្រុង "Birmingham" ត្រូវសរសេរកូដដូចខាងក្រោម៖

SELECT Name, City

FROM Customers

WHERE City = 'Birmingham';

Results | Messages

	Name	City
1	Molunguri, A	Birmingham
2	Wieneke, Frank	Birmingham
3	Wazny, Tina	Birmingham
4	Bernard, Subramaniyan	Birmingham
5	Bingham, Robert	Birmingham
6	Dunn, Ole	Birmingham

(ចំណាំ៖ ចំពោះទិន្នន័យជាប្រភេទអក្សរ ឬអត្ថបទ ត្រូវតែប្រើសញ្ញាសម្រង់ទោល '' ព័ទ្ធជុំវិញជានិច្ច)

ឧទាហរណ៍ទី២ (ប្រភេទលេខ - Numeric Fields)៖ ប្រសិនបើចង់ស្វែងរកអតិថិជនដែលមានលេខសម្គាល់ CustomerID = 6 ត្រូវសរសេរកូដដូចខាងក្រោម៖

Results | Messages

	CustomerID	Name	Address	City	State	ZipCode
1	6	Nuckols, Heather	150 Hayes St	Elgin	IL	60120

- **Comparison Operators(ប្រមាណវិធីប្រៀបធៀប)**

យើងអាចប្រើសញ្ញាប្រៀបធៀបជាមួយ WHERE ដូចជា៖

=: ស្មើ

<> ឬ !=: មិនស្មើ

> : ធំជាង

< : តូចជាង

>=: ធំជាង ឬស្មើ

<=: តូចជាង ឬស្មើ

ក. សញ្ញាស្មើ (=)៖ ប្រើសម្រាប់ការស្វែងរកទិន្នន័យចំៗ (ដូចបង្ហាញក្នុងឧទាហរណ៍ខាងលើ)។

ខ. សញ្ញាធំជាង (>)៖ ប្រសិនបើអ្នកចង់ទាញរបាយការណ៍ទំនិញទាំងអស់ពីតារាង Products ដែលមានតម្លៃ (UnitPrice) ធំជាង ៥០ ដុល្លារ៖

Results | Messages

	ProductCode	Description	UnitPrice
1	A4CS	Murach's ASP.NET 4 Web Programming with C# 2010	56.50
2	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	56.50
3	ADC4	Murach's ADO.NET 4 with C# 2010	56.50
4	ADV4	Murach's ADO.NET 4 with VB 2010	56.50
5	CS10	Murach's C# 2010	56.50
6	JAVP	Murach's Java Programming	56.50
7	JSP2	Murach's JAVA Servlets and JSP (2nd Edition)	52.50
8	MCBL	Murach's Structured COBOL	62.50
9	MCCP	Murach's CICS for the COBOL Programmer	54.00
10	MDOM	Murach's JavaScript and DOM Scripting	54.50
11	SQ12	Murach's SQL Server 2012	57.50
12	VB10	Murach's Visual Basic 2010	56.50
13	ZJLR	Murach's OS/390 and z/os JCL	62.50

Query executed successfully.

គ. សញ្ញាតូចជាង (<)៖ ទាញយកទំនិញដែលមានតម្លៃថោកជាង ៥០.00 ដុល្លារ ចុះក្រោម៖

```
SELECT ProductCode, Description, UnitPrice

FROM Products

WHERE UnitPrice < 50.00;
```

Results | Messages

	ProductCode	Description	UnitPrice
1	DB1R	DB2 for the COBOL Programmer, Part 1 (2nd Edition)	42.00
2	DB2R	DB2 for the COBOL Programmer, Part 2 (2nd Edition)	45.00

ឃ. សញ្ញាធំជាង ឬស្មើ (>=)៖ ទាញយកទំនិញដែលមានតម្លៃចាប់ពី ៥០.00 ដុល្លារ ឡើងទៅ (រួមបញ្ចូលទាំងទំនិញដែលមានតម្លៃចំ ៥០.00 ដុល្លារផងដែរ)៖

```
SELECT ProductCode, Description, UnitPrice

FROM Products

WHERE UnitPrice >= 50.00;
```

Results | Messages

	ProductCode	Description	UnitPrice
1	A4CS	Murach's ASP.NET 4 Web Programming with C# 2010	56.50
2	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	56.50
3	ADC4	Murach's ADO.NET 4 with C# 2010	56.50
4	ADV4	Murach's ADO.NET 4 with VB 2010	56.50
5	CRFC	Murach's CICS Desk Reference	50.00
6	CS10	Murach's C# 2010	56.50
7	JAVP	Murach's Java Programming	56.50
8	JSP2	Murach's JAVA Servlets and JSP (2nd Edition)	52.50
9	MCBL	Murach's Structured COBOL	62.50
10	MCCP	Murach's CICS for the COBOL Programmer	54.00
11	MDOM	Murach's JavaScript and DOM Scripting	54.50
12	SQ12	Murach's SQL Server 2012	57.50
13	VB10	Murach's Visual Basic 2010	56.50
14	ZJLR	Murach's OS/390 and z/os JCL	62.50

Query executed successfully.

ង. សញ្ញាតូចជាង ឬស្មើ (<=)៖ ប្រើសម្រាប់ទាញយកទិន្នន័យដែលមានតម្លៃតូចជាង ឬស្មើនឹងតម្លៃគោល។

ឧទាហរណ៍៖ ទាញយកទិន្នន័យទំនិញពីតារាង Products *ដែលមានតម្លៃ* (UnitPrice) *ត្រឹម ៥០.០០ ដុល្លារ ចុះក្រោម (រួមបញ្ចូលទាំងទំនិញដែលមានតម្លៃចំ ៥០.០០ ដុល្លារផងដែរ)៖*

```
SELECT ProductCode, Description, UnitPrice

FROM [dbo].[Products]

WHERE UnitPrice <= 50.00;
```

Results | Messages

	ProductCode	Description	UnitPrice
1	CRFC	Murach's CICS Desk Reference	50.00
2	DB1R	DB2 for the COBOL Programmer, Part 1 (2nd Edition)	42.00
3	DB2R	DB2 for the COBOL Programmer, Part 2 (2nd Edition)	45.00

ច. សញ្ញាមិនស្មើ (<> ឬ !=)៖ ប្រើសម្រាប់ទាញយកទិន្នន័យទាំងអស់ លើកលែងតែតម្លៃដែលបានកំណត់ចេញ។

ឧទាហរណ៍៖ ទាញយកទំនិញទាំងអស់នៅក្នុងតារាង Products *លើកលែងតែទំនិញណាដែលមានតម្លៃចំ ៥០.០០ ដុល្លារគត់ចេញ៖*

```
SELECT ProductCode, Description, UnitPrice

FROM [dbo].[Products]
```

WHERE UnitPrice <> 50.00;

	ProductCode	Description	UnitPrice
1	A4CS	Murach's ASP.NET 4 Web Programming with C# 2010	56.50
2	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	56.50
3	ADC4	Murach's ADO.NET 4 with C# 2010	56.50
4	ADV4	Murach's ADO.NET 4 with VB 2010	56.50
5	CS10	Murach's C# 2010	56.50
6	DB1R	DB2 for the COBOL Programmer, Part 1 (2nd Edition)	42.00
7	DB2R	DB2 for the COBOL Programmer, Part 2 (2nd Edition)	45.00
8	JAVP	Murach's Java Programming	56.50
9	JSP2	Murach's JAVA Servlets and JSP (2nd Edition)	52.50
10	MCBL	Murach's Structured COBOL	62.50
11	MCCP	Murach's CICS for the COBOL Programmer	54.00
12	MDOM	Murach's JavaScript and DOM Scripting	54.50
13	SQ12	Murach's SQL Server 2012	57.50
14	VB10	Murach's Visual Basic 2010	56.50
15	ZJLR	Murach's OS/390 and z/os JCL	62.50

Query executed successfully.

▪ Logical Operators (AND, OR, NOT)

ខណៈពេលដែល Comparison Operators ធ្វើការត្រួតពិនិត្យទៅលើលក្ខខណ្ឌតែមួយ ប៉ុន្តែការទាញយកទិន្នន័យជាក់ស្តែងជារឿយៗតែងតែទាមទារលក្ខខណ្ឌច្រើនក្នុងពេលតែមួយ។ ដើម្បីធ្វើការងារទាំងនេះបាន SQL ប្រើប្រាស់ Logical operators ដើម្បីភ្ជាប់លក្ខខណ្ឌ៖

- AND: ទិន្នន័យនឹងត្រូវទាញយក លុះត្រាតែលក្ខខណ្ឌទាំងសងខាងពិត (TRUE) ទាំងពីរ។

❖ ទម្រង់ទូទៅ (Syntax)៖

SELECT column1, column2, ...

FROM table_name

WHERE condition1 AND condition2;

ឧទហរណ៍៖ យើងចង់ស្វែងរកផលិតផលដែលមានតម្លៃសមរម្យ គឺផលិតផលដែលមានតម្លៃថ្លៃជាង ២០.00 ដុល្លារ ផង និងចោកជាង ៥០.00 ដុល្លារ ផង៖

Results | Messages

	ProductCode	Description	UnitPrice
1	DB1R	DB2 for the COBOL Programmer, Part 1 (2nd Edition)	42.00
2	DB2R	DB2 for the COBOL Programmer, Part 2 (2nd Edition)	45.00

OR: ទិន្នន័យនឹងត្រូវទាញយក ប្រសិនបើមានលក្ខខណ្ឌណាមួយ (ឬលក្ខខណ្ឌទាំងពីរ) ពិត (TRUE)។

❖ ទម្រង់ទូទៅ (Syntax)៖SELECT column1, column2, ...

FROM table_name

WHERE condition1 OR condition2;

ឧទាហរណ៍៖ យើងចង់ទាញយកតែទិន្នន័យអតិថិជន (Customers) ណាដែលរស់នៅក្នុងរដ្ឋ New York (NY) ឬ California (CA) ប៉ុណ្ណោះ៖

Results | Messages

	Name	City	State
1	Muhinyi, Mauda	New York	NY
2	Lutonsky, Christopher	Woodland Hills	CA
3	Rascano, Darrell	Rancho Cordova	CA
4	Johnson, Ajith	McGregor	CA
5	Chamberland, Sarah	Doraville	CA
6	Thornton, Sreedhar	Brooklyn	NY
7	Leach, Venkat	Rowland Hts	CA
8	Lair, Andrew	San Francisco	CA
9	Roby, Chris	Sacto	CA
10	Kumari, Gopinath	Queens Village	NY
11	Azure, Mike	Mendon	NY
12	Deguzman, Jose	Brooklyn	NY
13	Marlatt, Robert	Yorktown Heights	NY
14	Mallipeddi, Rick	Milpitas	CA
15	Buford, Bry	New York	NY
16	Wigton, Alex	Brooklyn	NY
17	Johnson, Stewart	San Diego	CA

Query executed successfully.

➢ NOT: *ប្រើសម្រាប់បដិសេធលក្ខខណ្ឌ ឬទាញយកទិន្នន័យដែលផ្ទុយពីលក្ខខណ្ឌ ដែលបានកំណត់។*

❖ *ទម្រង់ទូទៅ* (Syntax)៖SELECT column1, column2, ...

FROM table_name

WHERE NOT condition;

ឧទាហរណ៍៖ ទាញយកទិន្នន័យអតិថិជនទាំងអស់ ដែលមិនមែនរស់នៅក្នុងរដ្ឋ

New York (NY)៖

SELECT Name, City, State

FROM [dbo].[Customers]

WHERE NOT State = 'NY';

Results | Messages

	Name	City	State
1	Molunguri, A	Birmingham	AL
2	Antony, Abdul	Fayetteville	NC
3	Smith, Ahmad	Savanna	IL
4	Chen, Rabi	Murfreesboro	TN
5	Nuckols, Heather	Elgin	IL
6	Lutonsky, Christopher	Woodland Hills	CA
7	Reddy, Aj	Plano	TX
8	Rascano, Darrell	Rancho Cordova	CA
9	Johnson, Ajith	McGregor	CA
10	Patwardhan, John	Bethel Park	PA
11	Swenson, Vin	Albany	OR
12	Nichols, Nadine	Rogers	AR
13	Guthrie, Rawle	Phoenix	AZ
14	Nguyen, Allison	Memphis	TN
15	Kandregula, Howard	Oklahoma City	OK
16	Yudkevich, Ed	Olathe	KS
17	Jiji, N	Dracut	MA

Query executed successfully.

▪ Special Operators

គឺជា Operator ដែលប្រើនៅក្នុង WHERE clause ដើម្បីកំណត់លក្ខខណ្ឌពិសេស សម្រាប់ការស្វែងរកទិន្នន័យពីតារាង។ វាជួយឱ្យការស្វែងរកទិន្នន័យមានភាពបត់បែន និងមានប្រសិទ្ធភាពជាងមុន។ Special operators មានដូចជា៖ BETWEEN, IN, LIKE, IS NULL, DISTINCT, EXISTS។

- BETWEEN operator: ប្រើសម្រាប់ស្វែងរកតម្លៃដែលស្ថិតនៅចន្លោះតម្លៃពីរ (លក្ខខណ្ឌរួមបញ្ចូលទាំងតម្លៃចុងសងខាង)។ នៅពេលប្រើ BETWEEN នៅក្នុង SQL ត្រូវតែសរសេរតម្លៃតូចជាមុន (Lower-range value) រួចទើបសរសេរតម្លៃធំតាមក្រោយ។

  ឧទាហរណ៍៖ ទាញយកទិន្នន័យផលិតផលដែលមានតម្លៃចន្លោះពី ៥០.០០ ដុល្លារ ដល់ ៦០.០០ ដុល្លារ៖

```
SELECT ProductCode, Description, UnitPrice

FROM [dbo].[Products]

WHERE UnitPrice BETWEEN 50.00 AND 60.00;
```

Results | Messages

	ProductCode	Description	UnitPrice
1	A4CS	Murach's ASP.NET 4 Web Programming with C# 2010	56.50
2	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	56.50
3	ADC4	Murach's ADO.NET 4 with C# 2010	56.50
4	ADV4	Murach's ADO.NET 4 with VB 2010	56.50
5	CRFC	Murach's CICS Desk Reference	50.00
6	CS10	Murach's C# 2010	56.50
7	JAVP	Murach's Java Programming	56.50
8	JSP2	Murach's JAVA Servlets and JSP (2nd Edition)	52.50
9	MCCP	Murach's CICS for the COBOL Programmer	54.00
10	MDOM	Murach's JavaScript and DOM Scripting	54.50
11	SQ12	Murach's SQL Server 2012	57.50
12	VB10	Murach's Visual Basic 2010	56.50

Query executed successfully.

ចំណាំ៖ ប្រសិនបើសរសេរតម្លៃខុសលំដាប់ ដូចជា BETWEEN 60.00 AND 50.00 នោះ RDBMS នឹងបកស្រាយជា WHERE UnitPrice >= 60.00 AND UnitPrice <= 50.00

ដែលលក្ខខណ្ឌនេះមិនអាចកើតឡើងបានឡើយ នាំឲ្យលទ្ធផលទទេ (ទាញមិនចេញទិន្នន័យ)។

- IN operator: ប្រើសម្រាប់ស្វែងរកតម្លៃណាដែលត្រូវគ្នានឹងតម្លៃមួយចំនួនដែលមានសៅក្នុងបញ្ជី (List)។

(ឧទាហរណ៍៖ WHERE State IN ('NY', 'CA', 'TX'))

- LIKE operator: ត្រូវបានប្រើសៅក្នុង SQL ជាមួយ Wildcards ដើម្បីស្វែងរកលំនាំអក្សរ (Patterns) នៅក្នុងទិន្នន័យប្រភេទអត្ថបទ (String)។ វាត្រូវបានប្រើនៅពេលដែលយើងមិនស្គាល់អត្ថបទទាំងមូល ប៉ុន្តែចង់ស្វែងរកតែផ្នែកខ្លះនៃអត្ថបទនោះ។

នៅក្នុង Standard SQL អនុញ្ញាតឲ្យប្រើ Wildcard characters ពីរប្រភេទគឺ៖

% (Percent sign): តំណាងឲ្យតួអក្សរជាច្រើន ឬគ្មានតួអក្សរទាល់តែសោះ។

_ (Underscore): តំណាងឲ្យតួអក្សរតែមួយគត់។

❖ ទម្រង់ទូទៅ (Syntax)៖SELECT column1, column2, ...

FROM table_name

WHERE column_name LIKE 'pattern';

ឧទាហរណ៍៖ ទាញយកទិន្នន័យរដ្ឋដែលមានលេខកូដ (StateCode) ចាប់ផ្តើមដោយអក្សរ 'A' និងមានប្រវែង ២ តួអក្សរគត់ (ប្រើ LIKE 'A_')៖

SELECT * FROM [dbo].States

WHERE StateCode LIKE 'A_';

Results | Messages

	StateCode	StateName
1	AK	Alaska
2	AL	Alabama
3	AR	Arkansas
4	AZ	Arizona

ចំណាំ៖ នៅក្នុង MS SQL Server ឬ MS Access គឺមិនប្រកាន់ Case-sensitive (មិនបែងចែកអក្សរធំឬតូច) ឡើយ ដូចនេះ 'A%' គឺស្មើនឹង 'a%'។ ប៉ុន្តែនៅលើ DBMS មួយចំនួនដូចជា MS Access គេប្រើ wildcard * និង ? ជំនួសឱ្យ % និង _ វិញ។

- IS NULL operator: ត្រូវបានប្រើសម្រាប់ពិនិត្យមើលថាតើតម្លៃនៅក្នុង Column ណាមួយជាតម្លៃទទេ (NULL) ឬអត់ (មិនមែនជាលេខសូន្យ 0 ឬចន្លោះ Space ទេ គឺតម្លៃគ្មានទិន្នន័យសោះ)។

ឧទាហរណ៍៖ ទាញយកទិន្នន័យផលិតផលដែលគ្មានការពិពណ៌នា (Description ជា NULL)៖

```
SELECT ProductCode, Description, UnitPrice, OnHandQuantity

FROM [dbo].[Products]

WHERE Description IS NULL;
```

Results | Messages

ProductCode	Description	UnitPrice	OnHandQuantity

(បញ្ជាក់៖ ប្រសិនបើចង់បញ្ចូលទិន្នន័យសាកល្បងដែលគ្មាន Description គេអាចប្រើកូដ INSERT ដោយមិនដាក់ Column Description ដូចជា៖ INSERT INTO Products (ProductCode, UnitPrice, OnHandQuantity) VALUES ('PROD000005', 56.99, 67);)

- DISTINCT operator: ប្រើសម្រាប់ចម្រោះយកតែទិន្នន័យដែលខុសៗគ្នា ដោយលុបចោលរាល់ទិន្នន័យដែលស្ទួន (Duplicate values) ចេញពីលទ្ធផលបង្ហាញ។

(ឧទាហរណ៍៖ SELECT DISTINCT UnitPrice FROM Products; វានឹងបង្ហាញតម្លៃទំនិញមិនឱ្យជាន់គ្នា)

- EXISTS operator: គឺជា SQL operator ដែលប្រើសម្រាប់ពិនិត្យមើលថាតើ Subquery (សំណួររងខាងក្នុង) មាន Record ត្រឡប់មកវិញឬអត់។ ប្រសិនបើមាន record វានឹងត្រឡប់តម្លៃ TRUE ហើយបើគ្មានទេ វានឹងត្រឡប់តម្លៃ FALSE ។ EXISTS មិនបង្ហាញទិន្នន័យដោយផ្ទាល់ទេ ប៉ុន្តែប្រើសម្រាប់ត្រួតពិនិត្យអត្ថិភាព (Existence) នៃទិន្នន័យ។

❖ ទម្រង់ទូទៅ (Syntax)៖SELECT column1, column2, ...

FROM table_name

WHERE EXISTS (subquery);

គ. ORDER BY

ORDER BY *គឺជាពាក្យគន្លឹះសម្រាប់ប្រើប្រាស់ដើម្បី រៀបលំដាប់ទិន្នន័យ ឱ្យមានសណ្តាប់ធ្នាប់ ដូចជាពីតូចទៅធំ ឬពីធំមកតូច។*

Syntax: SELECT column1, column2, ...

FROM table_name

ORDER BY column_name [ASC | DESC];

Grouping Data (*ការចាត់ក្រុមទិន្នន័យ៖ គឺជាការប្រមូលផ្តុំជួរដេក* (Rows) *ជាច្រើនដែលមានតម្លៃដូចគ្នានៅក្នុង* Column *មួយ ឱ្យទៅជាក្រុមតែមួយ ដើម្បីធ្វើការគណនាជាលក្ខណៈសរុប ដោយប្រើប្រាស់* Aggregate functions (*ដូចជា៖* SUM() *សរុប*, COUNT() *រាប់ចំនួន*, AVG() *រកមធ្យមភាគ*, MAX() *រកតម្លៃធំបំផុត*, MIN() *រកតម្លៃតូចបំផុត*)។

❖ *ទម្រង់ទូទៅ* (Syntax)៖SELECT column_name, aggregate_function(column_name) AS alias

FROM table_name

GROUP BY column_name;

ឧទាហរណ៍៖ ខាងក្រោមនេះបង្ហាញពីការប្រើប្រាស់ GROUP BY *ដើម្បីចាត់ក្រុមទិន្នន័យតាម* CustomerID *រួមជាមួយអនុគមន៍* COUNT() *ដើម្បីរាប់ចំនួនវិក្កយបត្រ និង* SUM() *ដើម្បីបូកសរុបទឹកប្រាក់ រួចតម្រៀបលទ្ធផលពីច្រើនទៅតិចតាម* TotalSpent*៖*

```
SELECT CustomerID,
    COUNT(InvoiceID) AS NumberOfInvoices,
    SUM(InvoiceTotal) AS TotalSpent
FROM [dbo].[Invoices]
GROUP BY CustomerID
```

ORDER BY TotalSpent DESC;

111 % 3 0

Results Messages

	CustomerID	NumberOfInvoices	TotalSpent
1	50	1	622.375
2	10	5	520.9375
3	20	3	476.525
4	15	3	193.4625
5	495	2	190.9625
6	125	1	188.4625
7	408	1	188.4625
8	33	1	181.475
9	11	3	177.875
10	694	2	175.375
11	25	2	128.975
12	40	1	126.475
13	233	1	126.475
14	239	1	126.475
15	470	1	126.475
16	601	1	122.175
17	666	1	64.4875

Query executed successfully.

ឃ.HAVING

- Having Clause (លក្ខខណ្ឌចម្រោះលើក្រុម)៖ គឺជា Clause មួយដែលប្រើនៅក្នុង SQL ដើម្បីកំណត់លក្ខខណ្ឌចម្រោះ (Filter) ទៅលើលទ្ធផលសរុប (Aggregate result) បន្ទាប់ពីការធ្វើ GROUP BY ដំណើរការរួចរាល់ហើយ។ វាមានលក្ខណៈដូច WHERE Clause ដែរ ប៉ុន្តែវាខុសគ្នានៅចំណុចសំខាន់មួយគឺ៖ WHERE ដំណើរការមុនពេលចាត់ក្រុម (Before grouping) ចំណែក HAVING ដំណើរការបន្ទាប់ពីចាត់ក្រុម (After grouping)។

❖ ទម្រង់ទូទៅ (Syntax)៖SELECT column_name, aggregate_function(column_name) AS alias

FROM table_name

GROUP BY column_name

HAVING aggregate_function(column_name) operator value;

ឧទាហរណ៍៖ ខាងក្រោមនេះបង្ហាញពីការប្រើប្រាស់ HAVING *ដើម្បីចម្រោះក្រុមបន្ទាប់ពី* GROUP BY *តាម* CustomerID *ដោយបង្ហាញតែអតិថិជនណាដែលមានចំនួនវិក្កយបត្រ* (Invoice) *លើសពី* ១ *និងតម្រៀបលទ្ធផលពីច្រើនទៅតិចតាម* TotalSpent

```
SELECT CustomerID,
    COUNT(*) AS NumOrders,
    SUM(InvoiceTotal) AS TotalSpent
FROM [dbo].Invoices
GROUP BY CustomerID
HAVING COUNT(*) > 1
ORDER BY TotalSpent DESC;
```

Results | Messages

	CustomerID	NumOrders	TotalSpent
1	10	5	520.9375
2	20	3	476.525
3	15	3	193.4625
4	495	2	190.9625
5	11	3	177.875
6	694	2	175.375
7	25	2	128.975

៧.ORDER BY Clause

- ORDER BY Clause (*ការតម្រៀបទិន្នន័យ៖ ត្រូវបានប្រើប្រាស់ដើម្បីតម្រៀបលទ្ធផលទិន្នន័យតាមលំដាប់លំដោយ ពីតូចទៅធំ* (Ascending - ASC: A-Z, 0-9) *ឬពីធំមកតូច* (Descending - DESC: Z-A, 9-0) *ដែលជាវិធីធ្វើឱ្យទិន្នន័យងាយស្រួលអាន និងស្វែងយល់ ។*

❖ ទម្រង់ទូទៅ (Syntax)៖

SELECT column1, column2, ...

FROM table_name

ORDER BY column1 ASC|DESC;

- ASC: តម្រៀបពីតូចទៅធំ (ជាលំនាំដើម - Default) ។
- DESC: តម្រៀបពីធំមកតូច។

ឧទាហរណ៍៖ យើងចង់មើលបញ្ជីឈ្មោះ (Name) និងទីក្រុង (City) របស់អតិថិជន ដោយចង់តម្រៀបបញ្ជីនោះតាមអក្ខរក្រមនៃឈ្មោះ ៖

- តម្រៀបតាម ASC (ពីតូចទៅធំ)៖

SELECT Name, City

FROM [dbo].[Customers]

ORDER BY Name ASC; -- (ត្រង់នេះមិនចាច់សរសេរ ASC ក៏បានព្រោះវាជាលំនាំដើមស្រាប់)

Results | Messages

	Name	City
1	Abeyatunge, Derek	North Chili
2	Abueg, Don	Montvale
3	Agarwala, Troy	Wapakoneta
4	Agrawal, Weining	Chickasha
5	Aguimba, Edward	Streamwood
6	Ahmed, Uma	Polk City
7	Akkiraju, Mike	Hazlet
8	Alderman, James	Columbia
9	Alexander, Skip	West Hartford
10	Allen, Craig	Pullman
11	Allen, Deanna	Demotte
12	Allen, Joel	Hilliard
13	Alsaidi, Jim	Apopka
14	Ambati, Greg	Landover
15	Amoroso, Hazel-ann	Tallahassee
16	Anderson, Alan	Albuquerque
17	Anderson, Jeff	Houston

Query executed successfully.

- *តម្រៀបតាម* DESC (*ពីធំមកតូច*)*៖*

SELECT Name, City

FROM [dbo].[Customers]

ORDER BY Name DESC;

Results | Messages

	Name	City
1	Zmaczynski, Mike	Schaumburg
2	Zimmer, Jian	Cinnaminson
3	Zavrotny, Lisa	Glen Allen
4	Zaslavskaya, Alok	Hacienda Heights
5	Yudkevich, Ed	Olathe
6	Yu, Hemant	Lanham
7	Young, James	Buffalo
8	Young, Jacquelyn	Richmond
9	Young, Chris	Falls Church
10	Yerrapu, Michele	Falls Church
11	Yciano, Nicolette	Gettysburg
12	Ybarra, George	Detroit
13	Yarbro, John	Dayton
14	Yang, Daniel	Queens
15	Xiong, J	Topeka
16	Wu, Jeffrey	Como
17	Wright, Venkat	Jamaica

Query executed successfully.

❖ *ករណីចង់តម្រៀបតាម* Multiple Columns (*ជួរឈរច្រើនរួមគ្នា*)៖

ឧទាហរណ៍៖ ដោយប្រើប្រាស់តារាង Products *ឧបមាថាយើងចង់មើលទំនិញនៅក្នុងស្តុកដោយចាប់ផ្តើមពីទំនិញដែលមានតម្លៃថោកជាងគេមុន។ យើងបានតម្រៀបផលិតផលដោយផ្អែកលើតម្លៃ* (UnitPrice) *ជាចម្បង* (*ពីទាបទៅខ្ពស់*) *និងបន្ទាប់មកតម្រៀបតាមអក្ខរក្រមនៃឈ្មោះទំនិញ* (Description) *ពីធំទៅតូច ដើម្បីដោះស្រាយក្នុងករណីដែលមានទំនិញតម្លៃស្មើគ្នា៖*

```
SELECT ProductCode, Description, UnitPrice

FROM [dbo].[Products]

ORDER BY UnitPrice ASC, Description DESC;
```

Results | Messages

	ProductCode	Description	UnitPrice
1	DB1R	DB2 for the COBOL Programmer, Part 1 (2nd Edition)	42.00
2	DB2R	DB2 for the COBOL Programmer, Part 2 (2nd Edition)	45.00
3	CRFC	Murach's CICS Desk Reference	50.00
4	JSP2	Murach's JAVA Servlets and JSP (2nd Edition)	52.50
5	MCCP	Murach's CICS for the COBOL Programmer	54.00
6	MDOM	Murach's JavaScript and DOM Scripting	54.50
7	VB10	Murach's Visual Basic 2010	56.50
8	JAVP	Murach's Java Programming	56.50
9	CS10	Murach's C# 2010	56.50
10	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	56.50
11	A4CS	Murach's ASP.NET 4 Web Programming with C# 2010	56.50
12	ADV4	Murach's ADO.NET 4 with VB 2010	56.50
13	ADC4	Murach's ADO.NET 4 with C# 2010	56.50
14	SQ12	Murach's SQL Server 2012	57.50
15	MCBL	Murach's Structured COBOL	62.50
16	ZJLR	Murach's OS/390 and z/os JCL	62.50

Query executed successfully.

❖ *ករណីចង់តម្រៀបតាម* Expression *ដោយប្រើអនុគមន៍* LEN()៖

វាធ្វើការគណនាប្រវែងអត្ថបទភ្លាមៗដោយមិនកែប្រែទិន្នន័យដើមនៅក្នុង Database *ឡើយ។ ចំពោះអនុគមន៍* LEN() *គឺមិនរាប់បញ្ចូលការដកឃ្លា* (Spaces) *នៅខាងចុងបញ្ចប់នៃពាក្យនោះទេ។*

ឧទាហរណ៍៖ យើងចង់មើល Column Name *ថាបើតាមលំដាប់ តើអ្នកណាដែលមានឈ្មោះវែងជាងគេ៖*

```
SELECT Name

FROM [dbo].[Customers]

ORDER BY LEN(Name) DESC;
```

Results | Messages

	Name
1	Wheatley, Sathyanarayana
2	Palanukumarasamy, Dianna
3	Valentin-eggert, Michael
4	Marshburn, Sridhar reddy
5	Chandrasekhar, Yagnaraju
6	Mukhopadhyay, Kondal rao
7	Arrington, Mallikarjuna
8	Corbett, Jayagangadhara
9	Schlusselberg, Brandon
10	Annamanaidu, Christian
11	Krishnamurthy, William
12	Shanmugasundaram, John
13	Burnett, Muralikrishna
14	Grimm, jr., Ramanathan
15	Bernard, Subramaniyan
16	Johnson, Balasundaram
17	Lutonsky, Christopher

Query executed successfully.

៣.១.២. INSERT INTO (ការបញ្ចូលទិន្នន័យថ្មី)

INSERT INTO *គឺជាពាក្យបញ្ជា* DML *ដែលប្រើសម្រាប់បន្ថែមជួរទិន្នន័យថ្មីៗ* (New Rows) *ទៅក្នុងតារាងដែលបានបង្កើតរួច។*

ក. INSERT INTO

❖ ទម្រង់ទូទៅ (Syntax)៖

INSERT INTO table_name (column1, column2, column3, ...)

VALUES (value1, value2, value3, ...);

- INSERT INTO table_name: *កំណត់ឈ្មោះតារាង* (Table) *ដែលយើងចង់បញ្ចូលទិន្នន័យថ្មី ។*
- (column1, column2, ...): *បញ្ជាក់ឈ្មោះ* Column *នីមួយៗដែលត្រូវទទួលទិន្នន័យ*
- VALUES (value1, value2, ...): *តម្លៃទិន្នន័យជាក់ស្តែងដែលត្រូវបញ្ចូល ។ តម្លៃទាំងនេះត្រូវតែមានលំដាប់លំដោយស្របគ្នាយ៉ាងត្រឹមត្រូវទៅនឹងឈ្មោះ* Column *ដែលបានរៀបរាប់ខាងលើ ។*

ឧទាហរណ៍៖ យើងចង់បញ្ចូលអតិថិជនថ្មីម្នាក់ឈ្មោះ 'Sok Dara' *ដែលរស់នៅទីក្រុង* 'Phnom Penh' *ទៅក្នុងតារាង* Customers *៖*

(សម្គាល់៖ មុនពេលយើងធ្វើការ INSERT *យើងអាចពិនិត្យមើលទិន្នន័យចាស់សិន រួចទើបអនុវត្តកូដខាងក្រោម)*

➢ INSERT Multiple Rows (*ការបញ្ចូលទិន្នន័យច្រើនជួររួមគ្នា*)៖ *ជាការបញ្ចូលទិន្នន័យច្រើនជួរ* (Multiple rows/records) *ទៅក្នុងតារាងទិន្នន័យក្នុងពេលតែមួយ។ វិធីសាស្ត្រនេះជួយកាត់បន្ថយពេលវេលាដំណើរការ និងបង្កើនប្រសិទ្ធភាពរបស់ប្រព័ន្ធ* Database*។*

❖ ទម្រង់ទូទៅ (Syntax)៖

INSERT INTO table_name (column1, column2, column3)

VALUES

(value1A, value2A, value3A),

(value1B, value2B, value3B),

(value1C, value2C, value3C);

សម្គាល់៖ យើងត្រូវប្រើសញ្ញាក្បៀស , ដើម្បីបំបែករវាងជួរទិន្នន័យថ្មីនីមួយៗ)

ឧទាហរណ៍៖ ប្រសិនបើក្រុមការងារត្រូវចុះឈ្មោះអតិថិជន (Customers) ចំនួន ៣ នាក់ចូលទៅក្នុងប្រព័ន្ធក្នុងពេលតែមួយ កូដដែលល្អបំផុតគឺមានទម្រង់បែបនេះ៖

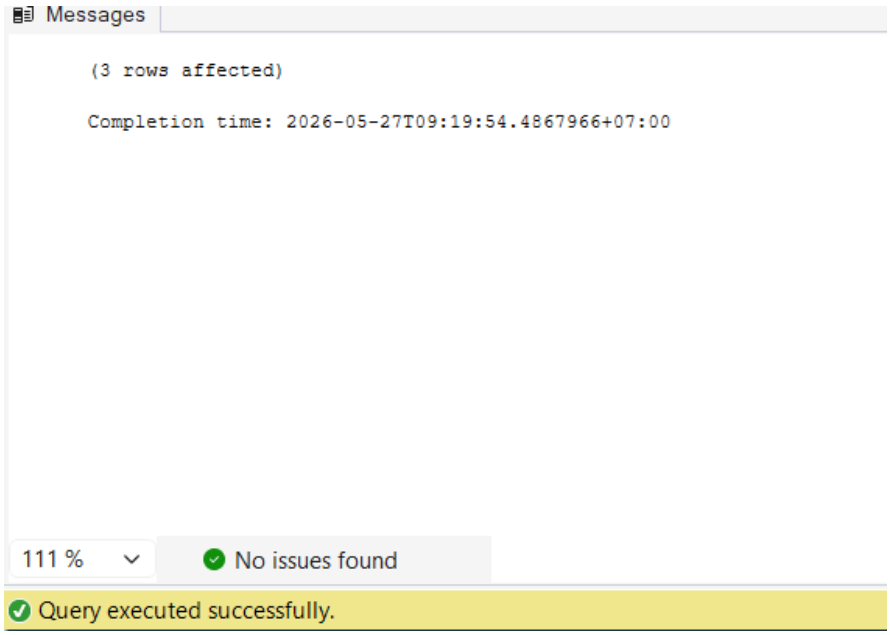
```
INSERT INTO [dbo].[Customers] (Name, Address, City, State, ZipCode)

VALUES

    ('Minea', '12 Street 200', 'Phnom Penh', 'CA', '12000'),

    ('Vibol', '99 Norodom Blvd', 'Phnom Penh', 'CA', '12000'),

    ('Sreypich', '45 Russian Fed', 'Phnom Penh', 'CA', '12000');
```



ករណីចង់ផ្ទៀងផ្ទាត់មើលទិន្នន័យដែលបានបញ្ចូលទាំង ៣ នាក់ គេអាចប្រើកូដ៖ SELECT * FROM Customers WHERE Name IN ('Minea', 'Vibol', 'Sreypich');

- BULK INSERT៖ គឺជាពាក្យបញ្ជា (Command) មួយនៅក្នុង SQL Server ដែលប្រើសម្រាប់ទាញយក (Import) ទិន្នន័យយ៉ាងច្រើនពី File ខាងក្រៅ (ដូចជាឯកសារ .txt ឬ .csv) ចូលទៅក្នុង Table នៃ Database បានដោយផ្ទាល់ យ៉ាងលឿន និងមានប្រសិទ្ធភាពខ្ពស់។

❖ ទម្រង់ទូទៅ (Syntax)៖ BULK INSERT table_name

```
FROM 'file_path'

WITH (
```

```
    FIELDTERMINATOR = 'separator_character',

    ROWTERMINATOR = 'newline_character'

);
```

- FIELDTERMINATOR: ប្រាប់ SQL Server ថាគួរប្រើសញ្ញាអ្វីដើម្បីបំបែកពី Column មួយទៅ Column មួយទៀត (ឧទាហរណ៍៖ សញ្ញាក្បៀស ,, ថេប \t, ឬបន្ទាត់បញ្ឈរ |)។

- ROWTERMINATOR: ប្រាប់ SQL Server ថាគួរប្រើសញ្ញាអ្វីដើម្បីបញ្ជាក់ពីការបញ្ចប់ជួរចាស់ និងចាប់ផ្តើមជួរថ្មី (ជាទូទៅគឺការចុះបន្ទាត់ \n)។

៣.១.៣.UPDATE

ក. ការកែប្រែជួរដេកតែមួយ (Single Row Update)

ប្រើសម្រាប់កែប្រែទិន្នន័យរបស់ Record ជាក់លាក់ណាមួយ ដោយប្រើលក្ខខណ្ឌដែលជា

Primary Key (ដូចជា ID)។

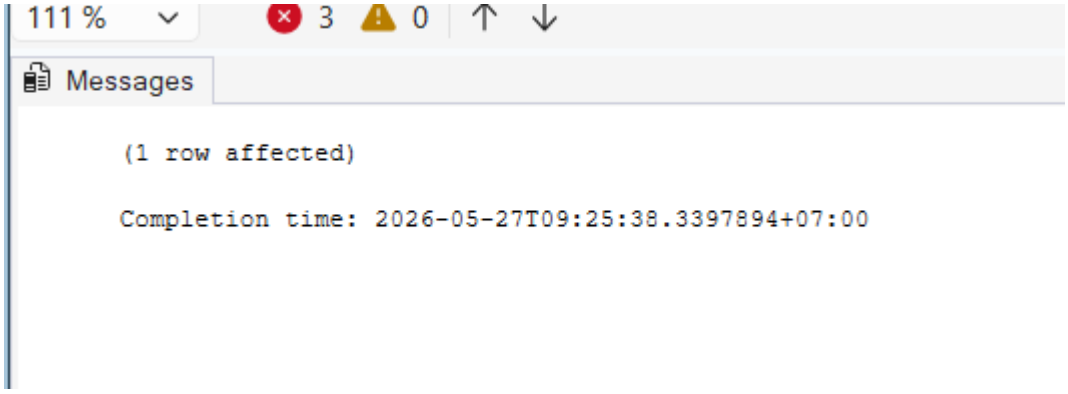
Syntax: UPDATE table_name

SET column_name = new_value

WHERE unique_id_column = specific_value;

⚠ ប្រយ័ត្ន៖ ប្រសិនបើអ្នកភ្លេចសរសេរលក្ខខណ្ឌ WHERE នោះ SQL Server នឹងធ្វើការកែប្រែទិន្នន័យថ្មីនេះទៅលើគ្រប់ជួរទិន្នន័យទាំងអស់នៅក្នុងតារាង ដែលអាចបណ្តាលឱ្យខូចខាត ឬបាត់បង់ទិន្នន័យដើមទាំងស្រុង។

ឧទាហរណ៍ (កែប្រែជួរទិន្នន័យតែមួយ)៖ យើងចង់ផ្លាស់ប្តូរតម្លៃស្យាវភៅដែលមានកូដផលិតផល 'JAVP' នៅក្នុងតារាង Products ទៅជាតម្លៃថ្មីគឺ $59.50៖



- UPDATE Multiple Rows (ការកែប្រែទិន្នន័យច្រើនជួរ)៖ កើតឡើងនៅពេលដែលលក្ខខណ្ឌនៅក្នុង WHERE ស្វែងរកឃើញទិន្នន័យលើសពីមួយ។ គេច្រើនប្រើវាសម្រាប់ធ្វើការកែសម្រួលទិន្នន័យជាសំណុំ ដូចជាការឡើងថ្លៃទំនិញ ឬការកំណត់ ZipCode ឡើងវិញជាដើម។

ឧទាហរណ៍៖ ផ្លាស់ប្តូរ ZipCode សម្រាប់អ្នកដែលរស់នៅក្នុងទីក្រុង 'Birmingham' ទាំងអស់៖

```
UPDATE [dbo].[Customers]

SET ZipCode = '35200-UPDATED'

WHERE City = 'Birmingham';
```

- UPDATE From Another Table (ការកែប្រែទិន្នន័យផ្អែកលើតារាងផ្សេង)៖ គឺជាវិធីសាស្ត្រមួយក្នុងការកែប្រែទិន្នន័យនៅក្នុងតារាងមួយ ដោយផ្អែកលើលក្ខខណ្ឌ ឬតម្លៃដែលមាននៅក្នុងតារាងមួយផ្សេងទៀតដែលមានទំនាក់ទំនង (Relationship) ជាមួយគ្នា។ វាធានាបាននូវភាពស៊ីសង្វាក់គ្នានៃទិន្នន័យ (Data consistency)។

ឧទាហរណ៍៖ យើងចង់កែប្រែតម្លៃ UnitPrice នៅក្នុងតារាង Products ឱ្យទៅជា 25.00 ផ្អែកលើទិន្នន័យដែលមានផ្គូផ្គងនៅក្នុងតារាង InvoiceLineItems៖

```
UPDATE [dbo].[Products]

SET [dbo].[Products].[UnitPrice] = 25.00

FROM [dbo].[Products]

JOIN [dbo].[InvoiceLineItems] ON [dbo].[Products].[ProductCode] =

[dbo].[InvoiceLineItems].[ProductCode];
```

- UPDATE with TOP៖ ពាក្យគន្លឹះ TOP ប្រើសម្រាប់កំណត់ចំនួនជួរទិន្នន័យអតិបរមាដែលត្រូវរងផលប៉ះពាល់ដោយបញ្ជា UPDATE ដើម្បីជៀសវាងការ Lock តារាងទាំងមូលយូរពេក ឬសម្រាប់ធ្វើការតេស្តសាកល្បង។

ឧទាហរណ៍៖ យើងចង់បង្កើនតម្លៃផលិតផល ១០% (គុណនឹង ១.១) សម្រាប់តែផលិតផល ៥ ដំបូងប៉ុណ្ណោះ៖

```
UPDATE TOP (5) [dbo].[Products]

SET UnitPrice = UnitPrice * 1.1;
```

- Updating Large Value Types with UPDATE .WRITE៖ វិធីសាស្ត្រ .WRITE ប្រើសម្រាប់កែប្រែទិន្នន័យតែមួយផ្នែកតូចនៅក្នុង Column ដែលមានប្រភេទជាទិន្នន័យទំហំធំដូចជា varchar(max)។ វាមានប្រសិទ្ធភាពខ្ពស់ព្រោះវាកែប្រែតែចំណុចដែលចាំបាច់ ដោយមិនចាំបាច់សរសេរអត្ថបទវែងៗទាំងមូលឡើងវិញឡើយ។

ទម្រង់៖ .WRITE (expression, @offset, @Length)

(Expression: ពាក្យថ្មី, Offset: ចំណុចចាប់ផ្តើមគិតជាតួអក្សរ, Length: ចំនួនតួអក្សរដែលត្រូវលុបចេញមុនជំនួសពាក្យថ្មី)។

ឧទាហរណ៍៖ ប្តូរ Column Description ឱ្យទៅជាប្រភេទ varchar(max) រួចកែប្រែបន្ថែមពាក្យ 'Advanced ' នៅខាងដើមគេ (Offset = 0, Length = 0) សម្រាប់ផលិតផលកូដ 'SQ12'៖

```
-- ជំហានដំបូង៖ ប្តូរប្រភេទ Column ជាមុនសិន

ALTER TABLE [dbo].[Products]

ALTER COLUMN [Description] varchar(max) NOT NULL;

-- ជំហានបន្ទាប់៖ ប្រើបញ្ជា .WRITE

UPDATE [dbo].[Products]

SET [Description].WRITE (N'Advanced ', 0, 0)

WHERE ProductCode = 'SQ12';
```

Results | Messages

	ProductCode	Description	UnitPrice	OnHandQuantity
1	SQ12	Advanced Murach's SQL Server 2012	25.00	2465

៣.១.៤.DELETE

DELETE គឺជាបញ្ជាមួយក្នុងចំណោមបញ្ជា DML ដែលប្រើសម្រាប់លុបទិន្នន័យ (Records) ចេញពីតារាង។

Syntax: DELETE FROM table_name

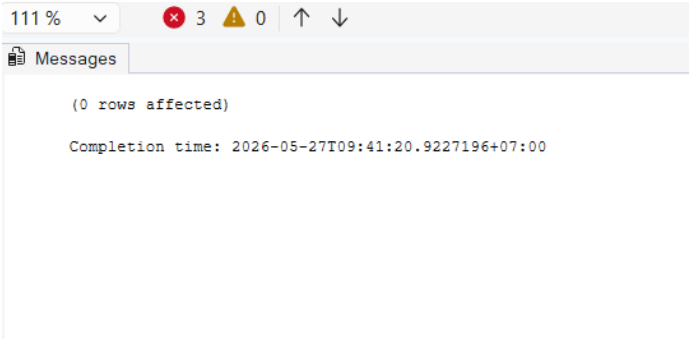
WHERE condition;

⚠ ប្រយ័ត្ន៖ ផ្នែក WHERE គឺសំខាន់បំផុត។ ប្រសិនបើអ្នកភ្លេចសរសេរលក្ខខណ្ឌ WHERE នោះរាល់ទិន្នន័យគ្រប់ជួរទាំងអស់នៅក្នុងតារាងនឹងត្រូវលុបបាត់អស់រលីងគ្មានសល់ឡើយ។

ឧទាហរណ៍ (លុបជួរទិន្នន័យតែមួយ)៖ យើងចង់លុបទិន្នន័យអតិថិជនម្នាក់ដែលមានលេខសម្គាល់ CustomerID = 702 (ដែលយើងបានបញ្ចូលក្នុងពេលធ្វើតេស្ត)៖

```
DELETE FROM [dbo].[Customers]

WHERE CustomerID = 702;
```



- Deleting Multiple Rows (ការលុបទិន្នន័យច្រើនជួរ)៖ ប្រសិនបើលក្ខខណ្ឌនៅក្នុង WHERE ត្រូវចំទិន្នន័យច្រើនជួរ នោះ SQL នឹងលុបជួរទាំងនោះចោលទាំងអស់ក្នុងពេលតែមួយ។

ឧទាហរណ៍៖ លុបទិន្នន័យអតិថិជនណាដែលមានលេខសម្គាល់ស្ថិតនៅក្នុងបញ្ជី (ប្រើ IN Clause)៖

```
DELETE FROM [dbo].[TableName]

WHERE CustomerID IN (703, 704, 705);
```

៣.១.៥. Comparation operators

Comparison operators ប្រើសម្រាប់ប្រៀបធៀបកន្សោមលក្ខខណ្ឌ (expression) ។

Syntax SELECT * FROM table_name

WHERE column_name = ,<,>,<=,=>,<> '…';

យើងមាន table_name (Drinks)

ក. សញ្ញាស្មើ (=)

ពន្យល់៖ ប្រើសម្រាប់ការស្វែងរកទិន្នន័យដែលត្រូវគ្នាចំៗ (ដូចជា

WHERE CustomerID = 6 ឬ WHERE City = 'Birmingham') ។

ខ. សញ្ញាធំជាង (>)

ឧទាហរណ៍៖ ប្រសិនបើអ្នកចង់ទាញរបាយការណ៍ទំនិញទាំងអស់ពី

តារាង Products *ដែលមានតម្លៃ* (UnitPrice) *ថ្លៃជាង* 50.00 *ដុល្លារ៖*

SELECT ProductCode, Description, UnitPrice

FROM [dbo].[Products]

WHERE UnitPrice > 50.00;

111 % 5 0

Results Messages

	ProductCode	Description	UnitPrice
1	ADC4	Murach's ADO.NET 4 with C# 2010	62.15
2	MCBL	Murach's Structured COBOL	62.50
3	MCCP	Murach's CICS for the COBOL Programmer	54.00
4	MDOM	Murach's JavaScript and DOM Scripting	54.50
5	ZJLR	Murach's OS/390 and z/os JCL	62.50

គ. សញ្ញាតូចជាង (<)

ឧទាហរណ៍៖ ចាញ់យកទិន្នន័យផលិតផលដែលមានតម្លៃថោកជាង 50.00 ដុល្លារ ចុះក្រោម៖

```
SELECT ProductCode, Description, UnitPrice

FROM [dbo].[Products]

WHERE UnitPrice < 50.00;
```

Results Messages

	ProductCode	Description	UnitPrice
1	A4CS	Murach's ASP.NET 4 Web Programming with C# 2010	27.50
2	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	27.50
3	ADV4	Murach's ADO.NET 4 with VB 2010	27.50
4	CRFC	Murach's CICS Desk Reference	27.50
5	CS10	Murach's C# 2010	25.00
6	DB1R	DB2 for the COBOL Programmer, Part 1 (2nd Edition)	25.00
7	DB2R	DB2 for the COBOL Programmer, Part 2 (2nd Edition)	25.00
8	JAVP	Murach's Java Programming	25.00
9	JSP2	Murach's JAVA Servlets and JSP (2nd Edition)	25.00
10	SQ12	Advanced Murach's SQL Server 2012	25.00
11	VB10	Murach's Visual Basic 2010	25.00

ឃ. សញ្ញាធំជាង ឬស្មើ (>=)

ឧទាហរណ៍៖ ទាញយកទិន្នន័យផលិតផលដែលមានតម្លៃចាប់ពី 50.00 ដុល្លារ ឡើងទៅ (រួមបញ្ចូលទាំងផលិតផលដែលមានតម្លៃចំ 50.00 ដុល្លារផងដែរ)៖

```
SELECT ProductCode, Description, UnitPrice

FROM [dbo].[Products]

WHERE UnitPrice >= 50.00;
```

Results | Messages

	ProductCode	Description	UnitPrice
1	ADC4	Murach's ADO.NET 4 with C# 2010	62.15
2	MCBL	Murach's Structured COBOL	62.50
3	MCCP	Murach's CICS for the COBOL Programmer	54.00
4	MDOM	Murach's JavaScript and DOM Scripting	54.50
5	ZJLR	Murach's OS/390 and z/os JCL	62.50

ង. សញ្ញាតូចជាង ឬស្មើ (<=)

ឧទាហរណ៍៖ ទាញយកទិន្នន័យផលិតផលដែលមានតម្លៃចាប់ពី 50.00 ដុល្លារ ចុះក្រោម (រួមបញ្ចូលទាំងផលិតផលដែលមានតម្លៃចំ 50.00 ដុល្លារផងដែរ)៖

Results | Messages

	ProductCode	Description	UnitPrice
1	A4CS	Murach's ASP.NET 4 Web Programming with C# 2010	27.50
2	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	27.50
3	ADV4	Murach's ADO.NET 4 with VB 2010	27.50
4	CRFC	Murach's CICS Desk Reference	27.50
5	CS10	Murach's C# 2010	25.00
6	DB1R	DB2 for the COBOL Programmer, Part 1 (2nd Edition)	25.00
7	DB2R	DB2 for the COBOL Programmer, Part 2 (2nd Edition)	25.00
8	JAVP	Murach's Java Programming	25.00
9	JSP2	Murach's JAVA Servlets and JSP (2nd Edition)	25.00
10	SQ12	Advanced Murach's SQL Server 2012	25.00
11	VB10	Murach's Visual Basic 2010	25.00

ច. សញ្ញាមិនស្មើ (<> ឬ !=)

ឧទាហរណ៍៖ ទាញយកទិន្នន័យផលិតផលទាំងអស់នៅក្នុងតារាង Products *លើកលែងតែផលិតផលណាដែលមានតម្លៃចំ* 50.00 *ដុល្លារ គត់ចេញ៖*

Results | Messages

	ProductCode	Description	UnitPrice
1	A4CS	Murach's ASP.NET 4 Web Programming with C# 2010	27.50
2	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	27.50
3	ADC4	Murach's ADO.NET 4 with C# 2010	62.15
4	ADV4	Murach's ADO.NET 4 with VB 2010	27.50
5	CRFC	Murach's CICS Desk Reference	27.50
6	CS10	Murach's C# 2010	25.00
7	DB1R	DB2 for the COBOL Programmer, Part 1 (2nd Edition)	25.00
8	DB2R	DB2 for the COBOL Programmer, Part 2 (2nd Edition)	25.00
9	JAVP	Murach's Java Programming	25.00
10	JSP2	Murach's JAVA Servlets and JSP (2nd Edition)	25.00
11	MCBL	Murach's Structured COBOL	62.50
12	MCCP	Murach's CICS for the COBOL Programmer	54.00
13	MDOM	Murach's JavaScript and DOM Scripting	54.50
14	SQ12	Advanced Murach's SQL Server 2012	25.00
15	VB10	Murach's Visual Basic 2010	25.00
16	ZJLR	Murach's OS/390 and z/os JCL	62.50

Query executed successfully.

៣.១.៦. Logical operators

Logical operators *ប្រើដើម្បីសិក្សាភាពពិតនៃលក្ខខណ្ឌមួយចំនួន ។*

Syntax SELECT * FROM table_name

WHERE column_name … AND … , … OR … , Not column name …

ក. AND

ពន្យល់៖ ប្រតិបត្តិករ AND *នឹងបង្ហាញជួរទិន្នន័យ* (Rows) *លុះត្រាតែរាល់លក្ខខណ្ឌទាំងអស់ត្រូវបានផ្ទៀងផ្ទាត់ថាត្រឹមត្រូវ* (TRUE) *ទាំងសងខាង។*

ឧទាហរណ៍៖ យើងចង់ស្វែងរកផលិតផលដែលមានតម្លៃសមរម្យ ពោលគឺផលិតផលដែលមានតម្លៃធំជាង 20.00 ដុល្លារផង និងតូចជាង 50.00 ដុល្លារផង (ចន្លោះពី ២០ ដល់ ៥០ ដុល្លារ)

```
SELECT ProductCode, Description, UnitPrice

FROM [dbo].[Products]

WHERE UnitPrice > 20.00 AND UnitPrice < 50.00;
```

Results | Messages

	ProductCode	Description	UnitPrice
1	A4CS	Murach's ASP.NET 4 Web Programming with C# 2010	27.50
2	A4VB	Murach's ASP.NET 4 Web Programming with VB 2010	27.50
3	ADV4	Murach's ADO.NET 4 with VB 2010	27.50
4	CRFC	Murach's CICS Desk Reference	27.50
5	CS10	Murach's C# 2010	25.00
6	DB1R	DB2 for the COBOL Programmer, Part 1 (2nd Edition)	25.00
7	DB2R	DB2 for the COBOL Programmer, Part 2 (2nd Edition)	25.00
8	JAVP	Murach's Java Programming	25.00
9	JSP2	Murach's JAVA Servlets and JSP (2nd Edition)	25.00
10	SQ12	Advanced Murach's SQL Server 2012	25.00
11	VB10	Murach's Visual Basic 2010	25.00

លទ្ធផលនឹងបង្ហាញតែផលិតផលណាដែលចូលក្នុងលក្ខខណ្ឌទាំងពីរនេះតែប៉ុណ្ណោះ

ខ. ប្រតិបត្តិករ OR (ឬ)

ពន្យល់៖ ប្រតិបត្តិករ OR នឹងបង្ហាញជួរទិន្នន័យ (Rows) ប្រសិនបើមានលក្ខខណ្ឌណាមួយ (យ៉ាងហោចណាស់មួយ) ត្រូវបានផ្ទៀងផ្ទាត់ថាត្រឹមត្រូវ (TRUE)។

ឧទាហរណ៍៖ យើងចង់ទាញយកតែទិន្នន័យអតិថិជន (Customers) ណាដែលរស់នៅក្នុងរដ្ឋ New York (NY) ឬរដ្ឋ California (CA) ប៉ុណ្ណោះ ៖

```
SELECT Name, City, State

FROM [dbo].[Customers]

WHERE State = 'NY' OR State = 'CA';
```

Results | Messages

	Name	City	State
1	Muhinyi, Mauda	New York	NY
2	Lutonsky, Christopher	Woodland Hills	CA
3	Rascano, Darrell	Rancho Cordova	CA
4	Johnson, Ajith	McGregor	CA
5	Chamberland, Sarah	Doraville	CA
6	Thornton, Sreedhar	Brooklyn	NY
7	Leach, Venkat	Rowland Hts	CA
8	Lair, Andrew	San Francisco	CA
9	Roby, Chris	Sacto	CA
10	Kumari, Gopinath	Queens Village	NY
11	Azure, Mike	Mendon	NY
12	Deguzman, Jose	Brooklyn	NY
13	Marlatt, Robert	Yorktown Heights	NY
14	Mallipeddi, Rick	Milpitas	CA
15	Buford, Bry	New York	NY
16	Wigton, Alex	Brooklyn	NY
17	Johnson, Stewart	San Diego	CA

គ. ប្រតិបត្តិករ NOT (មិនមែន / បដិសេធ)

ពន្យល់៖ ប្រតិបត្តិករ NOT ត្រូវបានប្រើដើម្បី**បដិសេធ ឬត្រឡប់លទ្ធផលនៃលក្ខខណ្ឌ** (បើលក្ខខណ្ឌពិត TRUE វានឹងប្តូរទៅជាមិនពិត FALSE វិញ) ពោលគឺយកទិន្នន័យណាដែលផ្ទុយពីលក្ខខណ្ឌដែលបានកំណត់។

ឧទាហរណ៍៖ យើងចង់ស្វែងរកអតិថិជនទាំងអស់ដែលរស់នៅរដ្ឋណាក៏បាន លើកលែងតែរដ្ឋ New York (NY) ចេញ។ យើងប្រើ NOT ដើម្បីដកពួកគេចេញពីលទ្ធផល ៖

```
SELECT Name, City, State

FROM [dbo].[Customers]

WHERE NOT State = 'NY';
```

Results | Messages

	Name	City	State
1	Molunguri, A	Birmingham	AL
2	Antony, Abdul	Fayetteville	NC
3	Smith, Ahmad	Savanna	IL
4	Chen, Rabi	Murfreesboro	TN
5	Nuckols, Heather	Elgin	IL
6	Lutonsky, Christopher	Woodland Hills	CA
7	Reddy, Aj	Plano	TX
8	Rascano, Darrell	Rancho Cordova	CA
9	Johnson, Ajith	McGregor	CA
10	Patwardhan, John	Bethel Park	PA
11	Swenson, Vin	Albany	OR
12	Nichols, Nadine	Rogers	AR
13	Guthrie, Rawle	Phoenix	AZ
14	Nguyen, Allison	Memphis	TN
15	Kandregula, Howard	Oklahoma City	OK
16	Yudkevich, Ed	Olathe	KS
17	Jiji, N	Dracut	MA

Query executed successfully.

លទ្ធផលនឹងបង្ហាញអតិថិជនមកពីគ្រប់រដ្ឋទាំងអស់ លើកលែងតែអ្នកមកពីរដ្ឋ NY

៣.១.៣ Special operators

ក.LIKE Operator

LIKE៖ ត្រូវបានប្រើប្រាស់នៅក្នុង WHERE Clause ជាមួយសញ្ញាជំនួស (Wildcards) ដើម្បីស្វែងរកលំនាំពាក្យ ឬអក្សរ (Patterns) នៅក្នុងជួរឈរដែលមានប្រភេទជាអត្ថបទ (String/Text Attributes)។ វាមានប្រយោជន៍ខ្លាំងនៅពេលដែលយើងមិនចាំពាក្យ ឬអត្ថបទទាំងមូល ប៉ុន្តែចង់ស្វែងរកផ្នែកណាមួយនៃពាក្យនោះ។

នៅក្នុង Standard SQL អនុញ្ញាតឱ្យប្រើប្រាស់ Wildcard Characters ចំនួនពីរប្រភេទ គឺ៖

- សញ្ញាភាគរយ (%)៖ តំណាងឱ្យតួអក្សរជាច្រើន (តួអក្សរអ្វីក៏បាន ប៉ុន្តែនតួបាន) ឬគ្មានតួអក្សរទាល់តែសោះ (Zero or more characters)។
- សញ្ញាបន្ទាត់ក្រោម (_)៖ តំណាងឱ្យតួអក្សរតែមួយគត់ (Exactly one character)។

❖ ទម្រង់ទូទៅ (Syntax)៖ SELECT column1, column2, ...

FROM table_name

WHERE column_name LIKE 'pattern';

ឧទាហរណ៍ទី១ (ប្រើ %)៖ ស្វែងរកអតិថិជនទាំងអស់ដែលមានឈ្មោះចាប់ផ្តើមដោយអក្សរ 'A' (បន្ទាប់ពីអក្សរ A អាចមានតួអក្សរអ្វីក៏បាន និងមានប្រវែងប៉ុន្តែនក៏បាន)៖

SELECT * FROM [dbo].[Customers]

WHERE Name LIKE 'A%';

Results | Messages

	CustomerID	Name	Address	City	State	ZipCode
1	3	Antony, Abdul	1109 Powderhorn Drive	Fayetteville	NC	28314
2	36	Annamanaidu, Christian	213 N. College Dr.	Omaha	NE	68127
3	48	Allen, Deanna	766 Willow Ridge Dr	Demotte	IN	46310
4	52	Azure, Mike	304 Combs	Mendon	NY	14606
5	92	Aguimba, Edward	444 Propsect St	Streamwood	IL	60107
6	153	Aneshansley, Baskaran	6720 Upper York Road	Ramona	CA	92065
7	157	Abeyatunge, Derek	1414 S. Dairy Ashford	North Chili	NY	14514
8	163	Alexander, Skip	15 Gettysburg Square 187	West Hartford	CT	06107
9	165	Ambati, Greg	1605 Candletree Dr. Suite 114	Landover	MD	20785
10	171	Ashley, Karla	108 Randolph Avenue	Sunrise	FL	33351
11	172	Arutla, Jerry I.	209 Ne 6Th	Ontario	CA	91762
12	185	Arzuza, Alioune	113 Railroad Ave.	Dallas	TX	75252
13	199	Abueg, Don	901 N. Lake Destiny	Montvale	NJ	07645
14	239	Anderson, Jeff	3625 Duval Rd #223	Houston	TX	77045
15	263	Allen, Joel	2220 W Mission Ln Apt 1127	Hilliard	OH	43026
16	303	Andrews, Ashok	949 Rockybrook Drive	Raleigh	NC	27613
17	316	Annandai, Curtis	443 Airpark Rd	Grand Prairie	TX	75052

Query executed successfully.

ឧទាហរណ៍ទី២ (ប្រើ _)៖ ស្វែងរកកូដរដ្ឋ (StateCode) ណាដែលចាប់ផ្តើមដោយអក្សរ 'A' ហើយមានប្រវែងសរុបត្រឹមតែ ២ តួអក្សរគត់៖

SELECT * FROM [dbo].[States]

WHERE StateCode LIKE 'A_';

Results | Messages

	StateCode	StateName
1	AK	Alaska
2	AL	Alabama
3	AR	Arkansas
4	AZ	Arizona

⚠️ ចំណាំ៖ * នៅក្នុង DBMS មួយចំនួនដូចជា Microsoft SQL Server វាមិនប្រកាន់តួអក្សរតូច ឬធំឡើយ (Not Case-Sensitive) ពោលគឺ 'A%' មានតម្លៃស្មើនឹង 'a%'។

ប៉ុន្តែសម្រាប់ DBMS មួយចំនួនទៀតដូចជា MS Access គេប្រើសញ្ញាផ្កាយ * ជំនួសឱ្យ % និងប្រើសញ្ញាសួរ ? ជំនួសឱ្យ ។

ខ. IS NULL Operator (ការពិនិត្យតម្លៃទទេ)

IS NULL៖ ត្រូវបានប្រើប្រាស់សម្រាប់ត្រួតពិនិត្យមើលថាតើតម្លៃនៅក្នុង Column ណាមួយជាតម្លៃទទេ (NULL) ឬអត់។ តម្លៃ NULL មានន័យថាជាតម្លៃដែលមិនទាន់ត្រូវបានបញ្ចូល ឬមិនមានទិន្នន័យទាល់តែសោះ។

⚠️ ចំណាំសំខាន់៖ នៅក្នុងភាសា SQL យើងមិនអាចប្រើសញ្ញាស្មើ (= NULL) ដើម្បីធ្វើការប្រៀបធៀបតម្លៃ NULL បានឡើយ។ ដូច្នេះ SQL ចាំបាច់ត្រូវតែប្រើប្រាស់ប្រតិបត្តិករ IS NULL នេះជំនួសវិញជានិច្ច ដើម្បីស្វែងរក Record ណាដែលគ្មានទិន្នន័យ។

❖ ទម្រង់ទូទៅ (Syntax)៖ SELECT column1, column2, ...

FROM table_name

WHERE column_name IS NULL;

ឧទាហរណ៍៖ ស្វែងរកផលិតផលទាំងអស់នៅក្នុងតារាង Products *ដែលមិនទាន់មានការពិពណ៌នា (ជួរឈរ* Description *ស្មើនឹង* NULL)*៖*

SELECT ProductCode, Description, UnitPrice, OnHandQuantity

FROM [dbo].[Products]

WHERE Description IS NULL;

Results | Messages

	ProductCode	Description	UnitPrice	OnHandQuantity
1	PROD000005	NULL	56.99	67
2	PROD000006	NULL	56.99	67
3	PROD000007	NULL	56.99	67

ផ្ទុយទៅវិញ ប្រសិនបើចង់ស្វែងរកជួរឈរណាដែលមានទិន្នន័យ មិនមែនជា NULL *គេប្រើប្រាស់* IS NOT NULL

គ. DISTINCT Operator (*ការបង្ហាញតម្លៃមិនស្ទួន*)

DISTINCT*៖ គឺជាពាក្យគន្លឹះ* (Keyword) *នៅក្នុង* SQL *ដែលត្រូវបានប្រើប្រាស់ជាមួយបញ្ជា* SELECT *ដើម្បីចម្រោះយកតែតម្លៃប្លែកៗពីគ្នា និងមិនស្ទួនគ្នានៃទិន្នន័យ* (Unique values) *យកមកបង្ហាញ។ មានន័យថា ប្រសិនបើក្នុងតារាងមានទិន្នន័យដដែលៗស្ទួនគ្នាជាច្រើនជួរ វានឹងដកភាពស្ទួននោះចេញ ហើយបង្ហាញតម្លៃនោះតែម្តងគត់។*

❖ ទម្រង់ទូទៅ (Syntax)៖

SELECT DISTINCT column_name

FROM table_name;

ឧទាហរណ៍៖ យើងចង់ដឹងថាតើតារាងផលិតផល Products មានតម្លៃឯកតា (UnitPrice) អ្វីខ្លះ ដោយមិនចង់ឱ្យបង្ហាញតម្លៃដដែលៗស្មុនគ្នាច្រើនដង៖

SELECT DISTINCT UnitPrice

FROM [dbo].[Products];

Results | Messages

	UnitPrice
1	25.00
2	27.50
3	54.00
4	54.50
5	62.15
6	62.50

ឃ.EXISTS Operator (ការពិនិត្យអត្ថិភាពទិន្នន័យ)

EXISTS៖ គឺជាប្រតិបត្តិករដែលប្រើសម្រាប់ត្រួតពិនិត្យមើលថាតើ Subquery (សំណួរយោងខាងក្នុង) អាចស្វែងរកឃើញ Record ណាខ្លះឬអត់ (ពិនិត្យអត្ថិភាពនៃទិន្នន័យ)។ ប្រសិនបើ Subquery រកឃើញទិន្នន័យយ៉ាងហោចណាស់ ១ ជួរ វានឹងប្រគល់តម្លៃពិត (TRUE) ហើយវានឹងអនុញ្ញាតឱ្យ Query ខាងក្រៅដំណើរការ ប៉ុន្តែបើមិនឃើញទាល់តែសោះ វានឹងប្រគល់តម្លៃមិនពិត (FALSE)។

❖ ទម្រង់ទូទៅ (Syntax)៖

SELECT column1, column2, ...

FROM table_name

WHERE EXISTS (subquery);

ឧទាហរណ៍៖ ស្វែងរកកូដ និងតម្លៃផលិតផលពីតារាង Products (*តំណាងដោយ* Alias p) *ដោយប្រើ* EXISTS *ដើម្បីពិនិត្យផ្ទៀងផ្ទាត់ជាមួយ* Subquery *ខាងក្នុងថាមានផលិតផលណាខ្លះដែលមានតម្លៃដូចគ្នា៖*

```
SELECT ProductCode, UnitPrice

FROM [dbo].[Products] p

WHERE EXISTS (

    SELECT * FROM [dbo].[Products]

    WHERE UnitPrice = p.UnitPrice

);
```

Results | Messages

	ProductCode	UnitPrice
1	A4CS	27.50
2	A4VB	27.50
3	ADC4	62.15
4	ADV4	27.50
5	CRFC	27.50
6	CS10	25.00
7	DB1R	25.00
8	DB2R	25.00
9	JAVP	25.00
10	JSP2	25.00
11	MCBL	62.50
12	MCCP	54.00
13	MDOM	54.50
14	SQ12	25.00
15	VB10	25.00
16	ZJLR	62.50

Query executed successfully.

៣.១.៨ Aggregate Fucntions

Aggregate Functions (អនុគមន៍គណនាសរុប)៖ គឺជាបណ្តុំមុខងារនៅក្នុង SQL ដែលប្រើសម្រាប់សង្ខេប ឬគណនាតម្លៃលេខចេញពីជួរដេក (Rows) ជាច្រើននៅក្នុង Query ឱ្យមកនៅសល់តម្លៃសរុបតែមួយ។ ក្នុងចំណោមអនុគមន៍ទាំង ៥ ខាងក្រោមនេះ COUNT គឺជាមុខងារដែលត្រូវបានគេប្រើប្រាស់ញឹកញាប់ជាងគេ ដោយវាផ្តល់នូវការរាប់ចំនួនសរុបនៃជួរដេកនៅក្នុង Query។ ចំណែកវអនុគមន៍គណនាសរុបផ្សេងៗទៀត ត្រូវបានប្រើប្រាស់សម្រាប់សង្ខេបតម្លៃប្រភេទលេខ (Numeric Values)។

ទម្រង់ទូទៅ (Syntax)៖

```
SELECT aggregate_function(column_name) AS alias_name
FROM table_name;
```

ក. COUNT

COUNT៖ គឺជាអនុគមន៍សម្រាប់រាប់ចំនួនជួរដេក (Rows) នៅក្នុងតារាង។

ឧទាហរណ៍៖ ខាងក្រោមនេះ បង្ហាញពីការប្រើប្រាស់ COUNT(*) ដើម្បីរាប់ចំនួនវិក្កយបត្រ (Invoice) សរុបទាំងអស់ពីតារាង Invoices៖

```
SELECT COUNT(*) AS TotalInvoices
```

FROM [dbo].[Invoices];

Results | Messages

	TotalInvoices
1	41

លទ្ធផលបង្ហាញថាមាន Invoice សរុបចំនួន 41 នៅក្នុងតារាង Invoice

ខ. MIN()

MIN() ប្រើសម្រាប់ស្វែងរកតម្លៃតូចបំផុត(Minimum value) ពីជួរឈរ (Column) ដែលបានបញ្ជាក់។

Syntax: SELECT MIN(column_name)
FROM table_name

ឧទាហរណ៍៖ ខាងក្រោមនេះ បង្ហាញពីការប្រើប្រាស់ MIN() ដើម្បីស្វែងរកតម្លៃទឹកប្រាក់វិក្កយបត្រដែលតូចជាងគេបំផុតពីតារាង Invoices៖

SELECT MIN(InvoiceTotal) AS SmallestOrder

FROM [dbo].[Invoices];

Results | Messages

	SmallestOrder
1	48.90

លទ្ធផលបង្ហាញថាតម្លៃ Invoice តូចបំផុតគឺ $48.90 នៅក្នុងតារាង Invoices

គ. MAX()

អនុកមន្ត MAX() ប្រើសម្រាប់ស្វែងរកតម្លៃធំបំផុត(Maximum value) ពីជួរឈរ (Column) ដែលបានជ្រើសរើស។

Syntax: SELECT MAX(column_name)
FROM table_name

ឧទាហរណ៍៖ ខាងក្រោមនេះ បង្ហាញពីការប្រើប្រាស់ MAX() ដើម្បីស្វែងរកតម្លៃទឹកប្រាក់វិក្កយបត្រដែលធំជាងគេបំផុតពីតារាង Invoices៖

```
SELECT MAX(InvoiceTotal) AS LargestOrder

FROM [dbo].[Invoices];
```

Results | Messages

	LargestOrder
1	622.375

លទ្ធផលបង្ហាញថាតម្លៃ Invoice ធំបំផុតគឺ $622,375.95 នៅក្នុងតារាង Invoices

ឃ. SUM()

អនុកមន្ត SUM() គឺជា Aggregate Function (មុខងារសរុប) ដែលប្រើសម្រាប់គណនាផលបូកសរុបនៃតម្លៃនៅក្នុងជួរឈរ (Column) ដែលជាលេខ។

Syntax: SELECT SUM(column_name)

FROM table_name

ឧទាហរណ៍៖ ខាងក្រោមនេះ បង្ហាញពីការប្រើប្រាស់ SUM() ដើម្បីគណនាផលបូកទឹកប្រាក់វិក្ក័យបត្រសរុប (InvoiceTotal) ពីតារាង Invoices

```
SELECT SUM(InvoiceTotal) AS TotalRevenue

FROM [dbo].[Invoices];
```

Results | Messages

	TotalRevenue
1	4446.8125

លទ្ធផលបង្ហាញថាផលបូក InvoiceTotal សរុបគឺ $1,290,237.95 នៅក្នុងតារាង Invoices

៧. AVG()

អនុកមន្ត AVG() ត្រូវបានប្រើគណនាមធ្យមភាគ (Average/Mean) នៃជួរឈរ (Column) ដែលមានទិន្នន័យជាលេខ

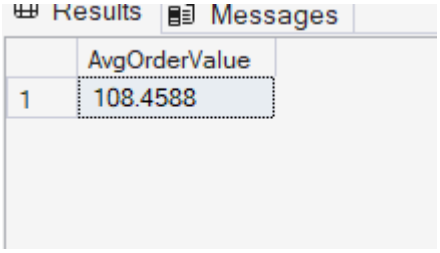
Syntax: SELECT AVG(column_name)

FROM table_name;

ឧទាហរណ៍៖ ខាងក្រោមនេះ បង្ហាញពីការប្រើប្រាស់ AVG() ដើម្បីគណនាតម្លៃមធ្យមភាគនៃវិក្កយបត្រពីតារាង Invoices៖

SELECT AVG(InvoiceTotal) AS AvgOrderValue

FROM [dbo].[Invoices];



លទ្ធផលបង្ហាញថាតម្លៃ Invoice មធ្យមភាគគឺ $108.4588នៅក្នុងតារាង Invoices

៣.២.១ ការអនុវត្ត Data Definition Language (DDL)

DDL (Data Definition Language) គឺជាផ្នែកមួយនៃភាសា SQL ដែលប្រើសម្រាប់បង្កើត កែប្រែ និងលុបរចនាសម្ព័ន្ធរបស់ Database និង Database Objects ផ្សេងៗ ដូចជា Database, Table, View, Index និង Constraint។

DDL មិនត្រូវបានប្រើសម្រាប់គ្រប់គ្រងទិន្នន័យនៅក្នុងតារាងទេ ប៉ុន្តែប្រើសម្រាប់ កំណត់រចនាសម្ព័ន្ធ និងទម្រង់នៃទិន្នន័យក្នុង Database។ ការប្រើប្រាស់ DDL ជួយឱ្យប្រព័ន្ធ Database មានសណ្តាប់ធ្នាប់ ងាយស្រួលគ្រប់គ្រង និងអាចរក្សាទុកទិន្នន័យបានត្រឹមត្រូវ។

ពាក្យបញ្ជាសំខាន់ៗរបស់ DDL រួមមាន៖

- CREATE : ប្រើសម្រាប់បង្កើត Database ឬ Table ថ្មី
- ALTER : ប្រើសម្រាប់កែប្រែរចនាសម្ព័ន្ធ Table
- DROP : ប្រើសម្រាប់លុប Database ឬ Table
- TRUNCATE : ប្រើសម្រាប់លុបទិន្នន័យទាំងអស់ក្នុង Table។

ដើម្បីប្រើប្រាស់ពាក្យបញ្ជាសំខាន់ៗរបស់DDLនេះ យើងបានលើកយកនៅCase Study "Student Information System for a Private University ដែលបានសិក្សាកាលពីឆមាសទី១មក សម្រាប់ធ្វើការអនុវត្ត ។

- ❖ សម្រាប់ការអនុវត្តដោយផ្អែកលើ Case Study "Student Information System for a Private University

ដែលបានបង្ហាញពីសាកលវិទ្យាល័យកំពុងជួបបញ្ហា ទិន្នន័យស្ទួន (Data Redundancy) និងភាពយឺតយ៉ាវក្នុងការគ្រប់គ្រង ដោយសារការប្រើប្រាស់ Spreadsheet។ ដើម្បីដោះស្រាយបញ្ហានេះ យើងត្រូវរៀបចំប្រព័ន្ធ Database ថ្មីមួយ។

មុននឹងចាប់ផ្តើមបង្កើត Database យើងត្រូវកំណត់ វិធានក្នុងប្រព័ន្ធមួយចំនួនជាមុនសិន ដើម្បីធានាថាទិន្នន័យនឹងត្រូវរក្សាទុកបានត្រឹមត្រូវ និងមិនមានការច្របូកច្របល់។

➢ Business Rules

1. Each student has a unique student ID and belongs to one department.

2. A student can enroll in many courses, and a course can have many students.

3. Each course is offered by one department.

4. A lecturer can teach multiple classes, but each class is taught by only one lecturer.

5. Each class must be scheduled in one room, but a room can host many classes at different times.

6. A student cannot enroll in the same course more than once in the same semester.

7. Grades are recorded for each student per enrolled course.

Entity	Attributes	Key Type
STUDENT	STU_ID, STU_FNAME, STU_LNAME, GENDER, DOB, SHIFT, BATCH, DEGREE_TYPE, STU_TYPE, ENROLL_STATUS, UNIVERSITY_EMAIL, DEPT_ID	STU_ID (PK), DEPT_ID (FK)
STUDENT_DETAIL	STU_ID, NATIONALITY, PHONE_NUM, POB, CURRENT_ADDRESS, PERSONAL_EMAIL, PARENT_NAME, PARENT_PHONE, ENTRY_QUALIFICATION	STU_ID (PK/FK)
DEPARTMENT	DEPT_ID, DEPT_NAME, FACULTY_NAME, OFFICE_LOC, HEAD_OF_DEPT	DEPT_ID (PK)
COURSE	COUR_ID, COUR_TITLE, CREDITS, DEPT_ID	COUR_ID (PK), DEPT_ID (FK)
LECTURER	LECT_ID, FULL_NAME, GENDER, ADDRESS, EMAIL, PHONE_NUM, SPECIALIZATION, DEPT_ID	LECT_ID (PK), DEPT_ID (FK)
CLASS	CLASS_ID, COUR_ID, LECT_ID, ROOM_ID, ACADEMIC_YEAR, SEMESTER, START_TIME, END_TIME, DAY_OF_WEEK, MAX_CAPACITY, CLASS_SECTION	CLASS_ID (PK), COUR_ID (FK), LECT_ID (FK), ROOM_ID (FK)
CLASSROOM	ROOM_ID, ROOM_NAME, ROOM_TYPE, LOCATION, FACILITIES	ROOM_ID (PK)

ENROLLMENT	STU_ID, CLASS_ID, ENROLL_DATE, ATTENDANCE_SCORE, MIDTERM_EXAM, FINAL_EXAM, STATUS, TOTAL_GRADE	STU_ID (PK/FK), CLASS_ID (PK/FK)

- អង្គភាព និងលក្ខណៈ (Entities and Attributes)
- Relationship

o STUDENT — has — STUDENT_DETAIL (Identifying relationship)

o DEPARTMENT— belongs to — STUDENT

o DEPARTMENT — offers — COURSE

o DEPARTMENT — employs — LECTURER

o LECTURER — teaches — CLASS

o COURSE — is offered as — CLASS

o CLASS — is scheduled in — CLASSROOM

o STUDENT — enrolls in — ENROLLMENT

o ENROLLMENT — is for — CLASS

- Constraints

1. លក្ខខណ្ឌនៃសោ (Key Constraints)

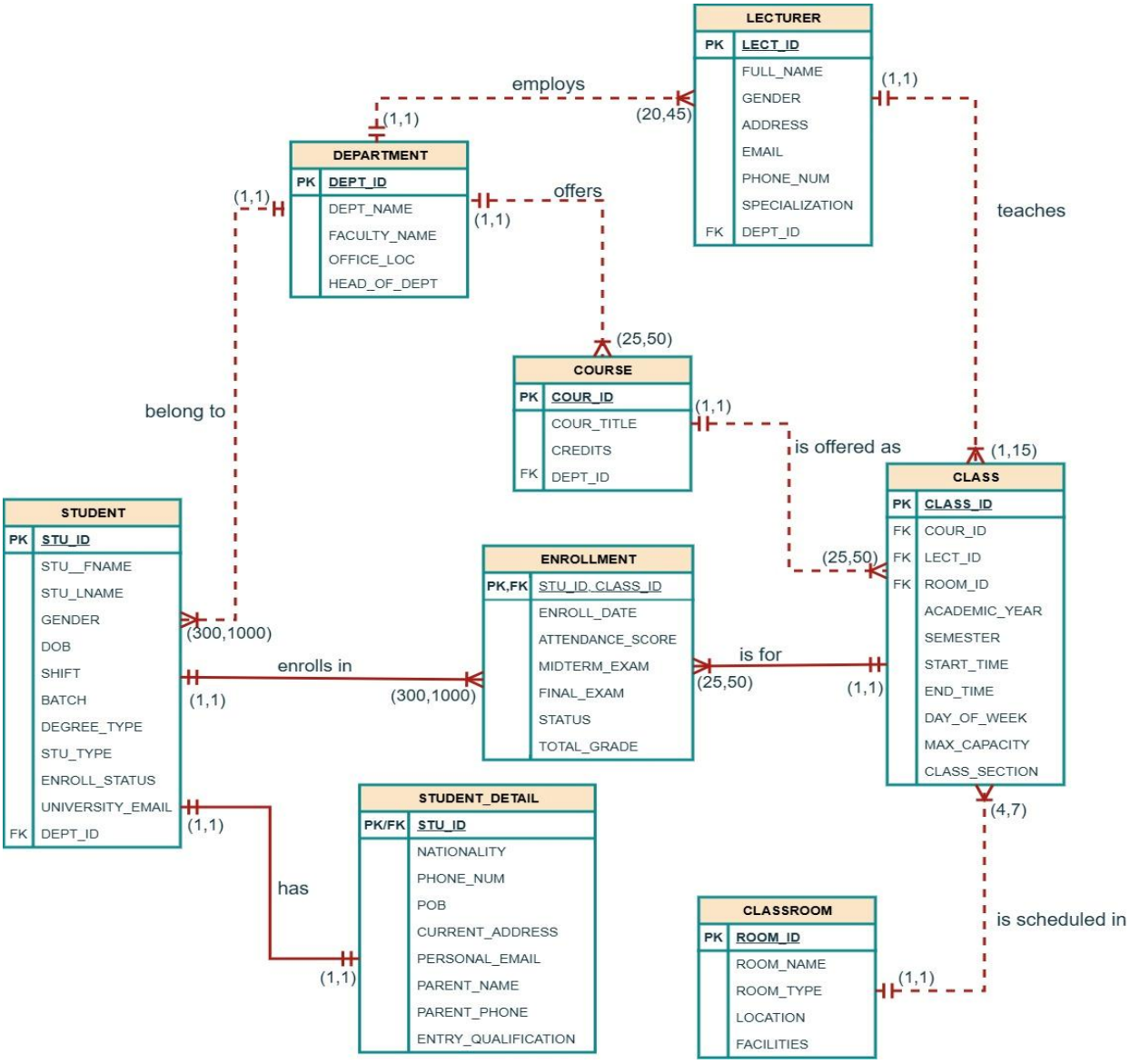
   - Primary Key: រាល់ Entity ទាំងអស់ (STU_ID, COUR_ID, LECT_ID, CLASS_ID, ROOM_ID) ត្រូវតែមានលេខសម្គាល់ដែលមិនអាចស្ទួន និងមិនអាចទទេបាន។
   - Foreign Key: ប្រើសម្រាប់ភ្ជាប់ទំនាក់ទំនងរវាងតារាង (ឧទាហរណ៍៖ DEPT_ID ក្នុងតារាង Student) ដើម្បីរក្សាសុពលភាពទំនាក់ទំនង (Referential Integrity)។
   - Unique Constraint: ក្នុងតារាង Enrollment ការរួមបញ្ចូលគ្នារវាង (STU_ID, CLASS_ID) ត្រូវតែមានតែមួយគត់ ដើម្បីការពារនិស្សិតចុះឈ្មោះរៀនថ្នាក់ដដែល ២ ដង។

2. លក្ខខណ្ឌនៃទិន្នន័យ (Data Validation)
   - Not Null: រាល់ថ្នាក់រៀន (Class) ទាំងអស់ដាច់ខាតត្រូវតែមាន សាស្ត្រាចារ្យបង្គោល (LECT_ID IS NOT NULL)។
   - Grade Range: ពិន្ទុរបស់និស្សិតត្រូវកំណត់ឱ្យបញ្ចូលបានតែចន្លោះពី ០ ដល់ ១០០ ប៉ុណ្ណោះ។
   - On Delete Cascade: នៅពេលដែលទិន្នន័យនៅក្នុងតារាង Student ត្រូវបានលុប រាល់ព័ត៌មានលម្អិតក្នុងតារាង Student_Detail នឹងត្រូវលុបដោយស្វ័យប្រវត្តិ។
3. លក្ខខណ្ឌអាជីវកម្ម (Business Logic Constraints)
   - Class Capacity: ចំនួននិស្សិតក្នុងថ្នាក់នីមួយៗមិនត្រូវឱ្យលើសពី MAX_CAPACITY នៃបន្ទប់ឡើយ។
   - Enrollment Limit: ថ្នាក់នីមួយៗត្រូវមាននិស្សិតចន្លោះពី ២៥ ទៅ ៥០ នាក់។
   - Lecturer Load: សាស្ត្រាចារ្យម្នាក់អាចបង្រៀនបានចាប់ពី ១ ដល់ ១៥ ថ្នាក់។
   - Scheduling: ការកំណត់ម៉ោង និងបន្ទប់រៀនត្រូវត្រួតពិនិត្យឱ្យបានច្បាស់លាស់ ដើម្បីជៀសវាងការជាន់ម៉ោងគ្នា។

ការដាក់លក្ខខណ្ឌទាំងនេះ នឹងជួយឱ្យ Database ដំណើរការបានយ៉ាងរលូន និងមិនមានកំហុសទិន្នន័យនៅពេលប្រើប្រាស់ជាក់ស្តែង។

➢ ER Diagram

- Draw an ER Diagram using Crow's Foot notation.



CREATE TABLE

CREATE TABLE គឺជាពាក្យបញ្ជាមួយនៅក្នុង DDL (Data Definition Language) ដែលប្រើសម្រាប់បង្កើតតារាងថ្មី (Table) នៅក្នុង Database។ តារាងនេះ(Table)ត្រូវបានប្រើសម្រាប់រក្សាទុកទិន្នន័យជាជួរឈរ (Column) និងជួរដេក (Row)។

នៅក្នុងការបង្កើត Table អ្នកអាចកំណត់៖

- ឈ្មោះ Table
- ឈ្មោះ Column
- Data Type
- Primary Key
- Foreign Key
- Constraint ផ្សេងៗ

1. **ការបង្កើត** Database (Database Creation)

ការបង្កើត Database ថ្មីមួយដែលមានឈ្មោះថា UniversityDB ដើម្បីរក្សាទុកតារាងទិន្នន័យទាំងអស់របស់សាកលវិទ្យាល័យ។

```
CREATE DATABASE UniversityDB;
GO
USE UniversityDB;
```

Syntax ទូទៅនៃការបង្កើត Table

```
CREATE TABLE table_name (
    column1_name data_type constraint,
    column2_name data_type constraint,
    column3_name data_type,
    CONSTRAINT pk_name PRIMARY KEY (column_as_primary_key),
    CONSTRAINT fk_name FOREIGN KEY (column_as_foreign_key)
        REFERENCES parent_table_name(primary_key_of_parent_table)
);
```

- CREATE TABLE tablename: ប្រើប្រាស់សម្រាប់បង្កើត Table ថ្មីមួយនៅក្នុង Database។
- column1, column2, ...: ជា Column ឬ Field ដែលជ្រើសរើសសម្រាប់ផ្ទុកទិន្នន័យ
- datatype: កំណត់ប្រភេទលក្ខណៈទិន្នន័យ ដូចជា INT (លេខគត់), VARCHAR (អក្សរ/លេខ), DATE (កាលបរិច្ឆេទ), DECIMAL (លេខទសភាគ) ជាដើម។
- onstraint: ជាការកំណត់ច្បាប់បន្ថែមទៅលើទិន្នន័យ ដូចជា NOT NULL, UNIQUE, PRIMARY KEY។
- PRIMARY KEY: ប្រើសម្រាប់កំណត់ Column ដែលមានតម្លៃមិនអាចស្ទួន និងមិនអាចទទេ (NOT NULL) និងប្រើសម្រាប់កំណត់សម្គាល់អត្តសញ្ញាណ Record នីមួយៗ។

- FOREIGN KEY: ប្រើសម្រាប់ភ្ជាប់ទំនាក់ទំនងរវាង Table មួយទៅកាន់ Table មួយទៀត។
- CONSTRAINT: ប្រើសម្រាប់កំណត់ឈ្មោះ និងគ្រប់គ្រងច្បាប់បន្ថែមនៅលើទិន្នន័យ។

2. តារាង DEPARTMENT

កូដបង្កើតរចនាសម្ព័ន្ធ និងលទ្ធផលទិន្នន័យក្នុងតារាង DEPARTMENT

- Primary Key: *គឺ* DEPT_ID *សម្រាប់សម្គាល់ដេប៉ាតឺម៉ង់នីមួយៗមិនឱ្យស្ទួន។*
- *ទិន្នន័យ៖ បង្ហាញពីឈ្មោះដេប៉ាតឺម៉ង់, មហាវិទ្យាល័យ, ទីតាំងការិយាល័យ និងឈ្មោះប្រធានដេប៉ាតឺម៉ង់។*

```sql
CREATE TABLE DEPARTMENT (
    DEPT_ID CHAR(4) PRIMARY KEY,
    DEPT_NAME VARCHAR(100),
    FACULTY_NAME VARCHAR(100),
    OFFICE_LOC VARCHAR(100),
    HEAD_OF_DEPT VARCHAR(100)
);
```

113 % 9 0

Results Messages

	DEPT_ID	DEPT_NAME	FACULTY_NAME	OFFICE_LOC	HEAD_OF_DEPT
1	D01	IT & Computer Science	Engineering	Bld A	Dr. Smith
2	D02	Psychology	Humanities	Bld B	Dr. Brown
3	D03	Economics	Business	Bld C	Dr. Wilson
4	D04	Cyber Security	ICT	Bld A	Dr. Rath
5	D05	Digital Economy	ICT	Bld A	Dr. Vichea

3. តារាង LECTURER

នៅក្នុងផ្នែកនេះ យើងបង្កើតតារាងសម្រាប់គ្រប់គ្រងព័ត៌មានលម្អិតរបស់សាស្ត្រាចារ្យ រួមមានជំនាញបង្រៀន និងព័ត៌មានទំនាក់ទំនង។

```sql
CREATE TABLE LECTURER (
    LECT_ID CHAR(5) PRIMARY KEY,
    FULL_NAME VARCHAR(100),
    GENDER CHAR(1),
    ADDRESS VARCHAR(150),
    EMAIL VARCHAR(100),
    PHONE_NUM VARCHAR(20),
    SPECIALIZATION VARCHAR(100),
    DEPT_ID CHAR(4)
);
SELECT * FROM LECTURER;

-- ========================================
```

113 % 9 0

Results Messages

	LECT_ID	FULL_NAME	GENDER	ADDRESS	EMAIL	PHONE_NUM	SPECIALIZATION	DEPT_ID
1	L201	Alice Johnson	F	123 Maple St	alice.j@uni.edu	555-0101	Data Science	D01
2	L202	Robert Chen	M	456 Oak Ave	r.chen@uni.edu	555-0102	Cybersecurity	D01
3	L203	Sarah Miller	F	789 Pine Rd	s.miller@uni.edu	555-0103	Algorithms	D02
4	L204	James Wilson	M	321 Elm Dr	j.wilson@uni.edu	555-0104	Cloud Computing	D01
5	L205	Maria Garcia	F	654 Birch Ln	m.garcia@uni.edu	555-0105	Artificial Intelligence	D03

- Primary Key: គឺ LECT_ID (ដូចជា L201, L202...) សម្រាប់សម្គាល់សាស្ត្រាចារ្យម្នាក់ៗ។
- Foreign Key: មាន DEPT_ID សម្រាប់ភ្ជាប់សាស្ត្រាចារ្យទៅកាន់ដេប៉ាតឺម៉ង់ដែលគាត់នៅក្រោមការគ្រប់គ្រង (ឧទាហរណ៍៖ D01 សម្រាប់ដេប៉ាតឺម៉ង់ IT)។
- Attributes សំខាន់ៗ: រួមមាន SPECIALIZATION (ជំនាញឯកទេស) និងព័ត៌មានទំនាក់ទំនងដូចជា EMAIL និង PHONE_NUM។

4. តារាង STUDENT

តារាងនេះប្រើសម្រាប់រក្សាទុកព័ត៌មានលម្អិតរបស់និស្សិតម្នាក់ៗ រួមទាំងព័ត៌មានផ្ទាល់ខ្លួន កម្រិតសិក្សា និងការតភ្ជាប់ទៅកាន់ដេប៉ាតឺម៉ង់តាមរយៈ Foreign Key។

```sql
CREATE TABLE STUDENT (
    STU_ID CHAR(6) PRIMARY KEY,
    STU_FNAME VARCHAR(50),
    STU_LNAME VARCHAR(50),
    GENDER CHAR(1),
    DOB DATE,
    SHIFT_TIME VARCHAR(20),
    BATCH VARCHAR(20),
    DEGREE_TYPE VARCHAR(50),
    STU_TYPE VARCHAR(50),
    ENROLL_STATUS VARCHAR(30),
    UNIVERSITY_EMAIL VARCHAR(100),
    DEPT_ID CHAR(4)
);
SELECT * FROM STUDENT;
```

113 % 9 0 Ln: 54, Ch: 1 (22 cha

Results Messages

	STU_ID	STU_FNAME	STU_LNAME	GENDER	DOB	SHIFT_TIME	BATCH	DEGREE_TYPE	STU_TYPE	ENROLL_STATUS	UNIVERSITY_EMAIL	DEPT_ID	STU_PHONE
3	S103	Michael	Brown	M	2003-02-10	Morning	2023	Associate	Part-time	Active	m.brown@uni.edu	D02	077-555-666
4	S104	Emily	Davis	F	2002-08-30	Evening	2023	Bachelors	Full-time	On Leave	e.davis@uni.edu	D01	010-444-555
5	S105	Chris	Wilson	M	2000-12-05	Morning	2021	Bachelors	Full-time	Graduated	c.wilson@uni.edu	D03	012-999-000
6	S201	Keo	Mony	M	2002-03-12	Morning	2023	Bachelors	Full-time	Active	keo.mony@uni.edu	D01	016-222-333
7	S202	Sok	Sreyneang	F	2001-05-20	Afternoon	2022	Bachelors	Full-time	Active	sok.srey@uni.edu	D01	011-777-888
8	S203	Chan	Vutha	M	2003-01-10	Morning	2023	Associate	Part-time	Active	chan.vutha@uni.edu	D02	085-555-444
9	S204	Nguon	Phally	F	2002-09-30	Evening	2023	Bachelors	Full-time	Active	nguon.phal@uni.edu	D01	093-111-222
10	S205	Heng	Rithy	M	2000-11-05	Morning	2021	Bachelors	Full-time	Active	heng.rithy@uni.edu	D03	099-888-777
11	S206	Vong	Sreypich	F	2002-07-15	Morning	2023	Bachelors	Full-time	Active	vong.pich@uni.edu	D04	081-333-555
12	S207	Chea	Sopheak	M	2001-08-25	Afternoon	2022	Bachelors	Full-time	Active	chea.pheak@uni.edu	D05	070-444-666
13	S208	Meas	Sokun	F	2003-04-18	Morning	2023	Associate	Full-time	Active	meas.sokun@uni....	D01	017-999-111
14	S209	Ros	Panha	M	2002-12-30	Evening	2023	Bachelors	Full-time	Active	ros.panha@uni.edu	D02	089-222-444
15	S210	Ouk	Dalin	F	2000-10-22	Morning	2021	Bachelors	Full-time	Active	ouk.dalin@uni.edu	D03	069-777-111

- Primary Key: គឺ STU_ID (ដូចជា S103, S104...) *សម្រាប់កំណត់អត្តសញ្ញាណនិស្សិត។*
- Foreign Key: គឺ DEPT_ID *ដែលតភ្ជាប់និស្សិតទៅកាន់ដេប៉ាតឺម៉ង់ដែលពួកគេកំពុងសិក្សា។*
- Attributes *សំខាន់ៗ៖*
  - DOB: *ថ្ងៃខែឆ្នាំកំណើត។*
  - SHIFT_TIME: *វេនសិក្សា* (Morning, Afternoon, Evening)*។*
  - ENROLL_STATUS: *ស្ថានភាពសិក្សា* (Active, Graduated, On Leave)*។*
  - DEGREE_TYPE: *កម្រិតសិក្សា* (Bachelors, Associate)*។*

5. តារាង STUDENT_DETAIL

តារាងនេះត្រូវបានបង្កើតឡើងដើម្បីរក្សាទុកព័ត៌មានលម្អិតបន្ថែមរបស់និស្សិតម្នាក់ៗ ដូចជាព័ត៌មានទំនាក់ទំនងអាណាព្យាបាល អាសយដ្ឋានបច្ចុប្បន្ន និងកម្រិតវប្បធម៌មុនចូលរៀន។

```
SQLQuery2.sq...2\User (79))*    SQLQuery1.s...\User (74))*

61
62  CREATE TABLE STUDENT_DETAIL (
63      STU_ID CHAR(6) PRIMARY KEY,
64      NATIONALITY VARCHAR(50),
65      PHONE_NUM VARCHAR(20),
66      POB VARCHAR(100),
67      CURRENT_ADDRESS VARCHAR(150),
68      PERSONAL_EMAIL VARCHAR(100),
69      PARENT_NAME VARCHAR(100),
70      PARENT_PHONE VARCHAR(20),
71      ENTRY_QUALIFICATION VARCHAR(100)
72  );
73  SELECT * FROM STUDENT_DETAIL;
74
75
76
```

113 % 9 0

Results | Messages

	STU_ID	NATIONALITY	POB	CURRENT_ADDRESS	PERSONAL_EMAIL	PARENT_NAME	PARENT_PHONE	ENTRY_QUALIFICATION	STU_PHONE
1	S101	Khmer	Phnom Penh	PP	john@mail.com	Robert	012-999-001	High School	012-111-222
2	S102	Khmer	Kandal	PP	jane@mail.com	Mary	012-999-002	High School	098-333-444
3	S103	Khmer	Siem Reap	SR	mike@mail.com	David	012-999-003	High School	077-555-666
4	S104	Khmer	Kampot	PP	emily@mail.com	Susan	012-999-004	High School	010-444-555
5	S105	Khmer	Battambang	BTB	chris@mail.com	Klaus	012-999-005	High School	012-999-000
6	S201	Khmer	Takeo	PP	mony@mail.com	Sovann	012-999-006	High School	016-222-333
7	S202	Khmer	Prey Veng	PP	srey@mail.com	Vannak	012-999-007	High School	011-777-888
8	S203	Khmer	Kampong Cham	PP	vutha@mail.com	Dara	012-999-008	High School	085-555-444
9	S204	Khmer	Kampong Speu	PP	phally@mail.com	Sokun	012-999-009	High School	093-111-222
10	S205	Khmer	Kampong Thom	PP	rithy@mail.com	Heng	012-999-010	High School	099-888-777
11	S206	Khmer	Koh Kong	PP	pich@mail.com	Vuthy	012-999-011	High School	081-333-555
12	S207	Khmer	Kep	PP	pheak@mail.com	Sambath	012-999-012	High School	070-444-666
13	S208	Khmer	Lowie	PP	sokun@mail.com	Phalla	012-999-013	High School	017-999-111
14	S209	Khmer	Kratie	PP	panha@mail.com	Piseth	012-999-014	High School	089-222-444
15	S210	Khmer	Mondulkiri	PP	dalin@mail.com	Vutha	012-999-015	High School	069-777-111

- Primary Key: *គឺ* STU_ID *ដែលប្រើសម្រាប់ភ្ជាប់ទំនាក់ទំនងបែប* 1:1 (One-to-One) *ជាមួយតារាង* STUDENT*។*
- *ព័ត៌មានអាណាព្យាបាល:* *រួមមាន* PARENT_NAME *និង* PARENT_PHONE *សម្រាប់ទំនាក់ទំនងក្នុងករណីចាំបាច់។*
- *ព័ត៌មានផ្ទាល់ខ្លួន:* *រួមមាន* NATIONALITY (*សញ្ជាតិ*), POB (*ទីកន្លែងកំណើត*), *និង* CURRENT_ADDRESS (*អាសយដ្ឋានបច្ចុប្បន្ន*)*។*

- កម្រិតវប្បធម៌: ENTRY_QUALIFICATION បង្ហាញពីប្រវត្តិសិក្សាមុនចូលសាកលវិទ្យាល័យ (ឧទាហរណ៍៖ High School)។

6. តារាង COURSE

តារាងនេះត្រូវបានបង្កើតឡើងដើម្បីគ្រប់គ្រងបញ្ជីមុខវិជ្ជាសិក្សានៅតាមដេប៉ាតឺម៉ង់នីមួយៗ រួមទាំងការកំណត់ចំនួនក្រេឌីតសម្រាប់មុខវិជ្ជានីមួយៗផងដែរ។

```
CREATE TABLE COURSE (
    COUR_ID CHAR(5) PRIMARY KEY,
    COUR_TITLE VARCHAR(100),
    CREDITS INT,
    DEPT_ID CHAR(4)
);
SELECT * FROM COURSE;

-- ========================================
```

124 % 9 0

Results Messages

	COUR_ID	COUR_TITLE	CREDITS	DEPT_ID
1	C301	Database Management Systems	3	D01
2	C302	Web Development	3	D01
3	C303	Introduction to Psychology	2	D02
4	C304	Macroeconomics	3	D03
5	C305	Data Structures & Algorithms	4	D01

- Primary Key: គឺ COUR_ID (ដូចជា C301, C302...) សម្រាប់កំណត់អត្តសញ្ញាណមុខវិជ្ជានីមួយៗឱ្យមានតែមួយគត់។
- Foreign Key: គឺ DEPT_ID ដែលភ្ជាប់មុខវិជ្ជាទៅកាន់ដេប៉ាតឺម៉ង់សាមី (ឧទាហរណ៍៖ C301 ស្ថិតក្នុងដេប៉ាតឺម៉ង់ D01 - IT)។
- Attributes សំខាន់ៗ៖
    - COUR_TITLE: ឈ្មោះពេញនៃមុខវិជ្ជា (ដូចជា Database Management Systems)។
    - CREDITS: ចំនួនក្រេឌីតនៃមុខវិជ្ជា (២, ៣ ឬ ៤ ក្រេឌីត)។

7. តារាង CLASSROOM

តារាងនេះប្រើសម្រាប់គ្រប់គ្រងទិន្នន័យទីតាំងសិក្សា និងសម្ភារឧបទេសដែលមាននៅក្នុងបន្ទប់រៀននីមួយៗ ដើម្បីងាយស្រួលក្នុងការចាត់ចែងកាលវិភាគសិក្សា។

```sql
CREATE TABLE CLASSROOM (
    ROOM_ID CHAR(4) PRIMARY KEY,
    ROOM_NAME VARCHAR(50),
    ROOM_TYPE VARCHAR(50),
    LOCATION VARCHAR(100),
    FACILITIES VARCHAR(200)
);
SELECT * FROM CLASSROOM;
```

150 % 9 0

Results | Messages

	ROOM_ID	ROOM_NAME	ROOM_TYPE	LOCATION	FACILITIES
1	R401	Lab A	Computer Lab	North Wing	30 PCs, Projector
2	R402	Lecture Hall 1	Auditorium	Main Building	Sound System, 200 Seats
3	R403	Room 202	Small Classroom	West Wing	Whiteboard, 25 Seats
4	R404	Seminar Room	Meeting Room	Library	Smart Board
5	R405	Lab B	Hardware Lab	North Wing	Electronics Kits

- Primary Key: គឺ ROOM_ID (ដូចជា R401, R402...) *សម្រាប់កំណត់សម្គាល់បន្ទប់នីមួយៗឱ្យដាច់ពីគ្នា។*
- Attributes *សំខាន់ៗ៖*
  - ROOM_NAME & ROOM_TYPE: *ប្រភេទបន្ទប់* (*ដូចជា* Computer Lab, Lecture Hall) *។*
  - LOCATION: *ទីតាំងអគារ* (*ដូចជា* North Wing, Main Building)*។*
  - FACILITIES: *បង្ហាញពីគ្រឿងបរិក្ខារដែលមានក្នុងបន្ទប់* (*ដូចជា* 30 PCs, Projector, Sound System)*។*

8. តារាង CLASS (ថ្នាក់រៀន)

តារាងនេះដើរតួជាអ្នកប្រមូលផ្តុំរាល់ធាតុផ្សំសំខាន់ៗ ដើម្បីបង្កើតជាកាលវិភាគសិក្សាជាក់ស្តែងសម្រាប់និស្សិត។

```
CREATE TABLE CLASS (
    CLASS_ID CHAR(6) PRIMARY KEY,
    COUR_ID CHAR(5),
    LECT_ID CHAR(5),
    ROOM_ID CHAR(4),
    ACADEMIC_YEAR VARCHAR(20),
    SEMESTER INT,
    START_TIME TIME,
    END_TIME TIME,
    DAY_OF_WEEK VARCHAR(20),
    MAX_CAPACITY INT,
    CLASS_SECTION VARCHAR(20)
);
SELECT * FROM CLASS;
```

113 % 9 0

Results Messages

	CLASS_ID	COUR_ID	LECT_ID	ROOM_ID	ACADEMIC_YEAR	SEMESTER	START_TIME	END_TIME	DAY_OF_WEEK	MAX_CAPACITY	CLASS_SECTION
1	CL501	C301	L201	R401	2023-2024	1	08:00:00	10:00:00	Monday	30	SEC-A
2	CL502	C302	L201	R401	2023-2024	1	10:30:00	12:30:00	Wednesday	30	SEC-B
3	CL503	C303	L203	R402	2023-2024	1	14:00:00	16:00:00	Tuesday	40	SEC-A
4	CL504	C304	L205	R403	2023-2024	2	09:00:00	11:00:00	Thursday	25	SEC-C
5	CL505	C305	L202	R405	2023-2024	2	13:00:00	15:00:00	Friday	40	SEC-A

- Primary Key: គឺ CLASS_ID (ដូចជា CL501, CL502...) *សម្រាប់សម្គាល់ថ្នាក់រៀននីមួយៗ។*
- Foreign Keys (*ការភ្ជាប់*):
  - COUR_ID: *ភ្ជាប់ទៅកាន់ មុខវិជ្ជា ដែលត្រូវរៀន។*
  - LECT_ID: *ភ្ជាប់ទៅកាន់ សាស្ត្រាចារ្យ ដែលត្រូវបង្រៀន។*
  - ROOM_ID: *ភ្ជាប់ទៅកាន់ បន្ទប់រៀន ដែលត្រូវប្រើប្រាស់។*
- *ព័ត៌មានកាលវិភាគ៖ រួមមាន* START_TIME, END_TIME *និង* DAY_OF_WEEK (*ថ្ងៃចន្ទ ដល់ សុក្រ*)*។*
- *លក្ខខណ្ឌថ្នាក់៖ រួមមាន* MAX_CAPACITY (*ចំនួននិស្សិតអតិបរមា*) *និង* CLASS_SECTION (*ដូចជា* SEC-A, SEC-B)*។*

9. តារាង ENROLLMENT

តារាងនេះត្រូវបានបង្កើតឡើងដើម្បីកត់ត្រាការចុះឈ្មោះចូលរៀនរបស់និស្សិតក្នុងថ្នាក់នីមួយៗ រួមទាំងការគ្រប់គ្រងវត្តមាន និងការរក្សាទុកពិន្ទុប្រឡងគ្រប់ដំណាក់កាល។

```sql
CREATE TABLE ENROLLMENT (
    STU_ID CHAR(6),
    CLASS_ID CHAR(6),
    ENROLL_DATE DATE,
    ATTENDANCE_SCORE DECIMAL(5,2),
    MIDTERM_EXAM DECIMAL(5,2),
    FINAL_EXAM DECIMAL(5,2),
    STATUS VARCHAR(20),
    TOTAL_GRADE DECIMAL(5,2),

    PRIMARY KEY (STU_ID, CLASS_ID)
);
SELECT * FROM ENROLLMENT;
```

124 % 9 0

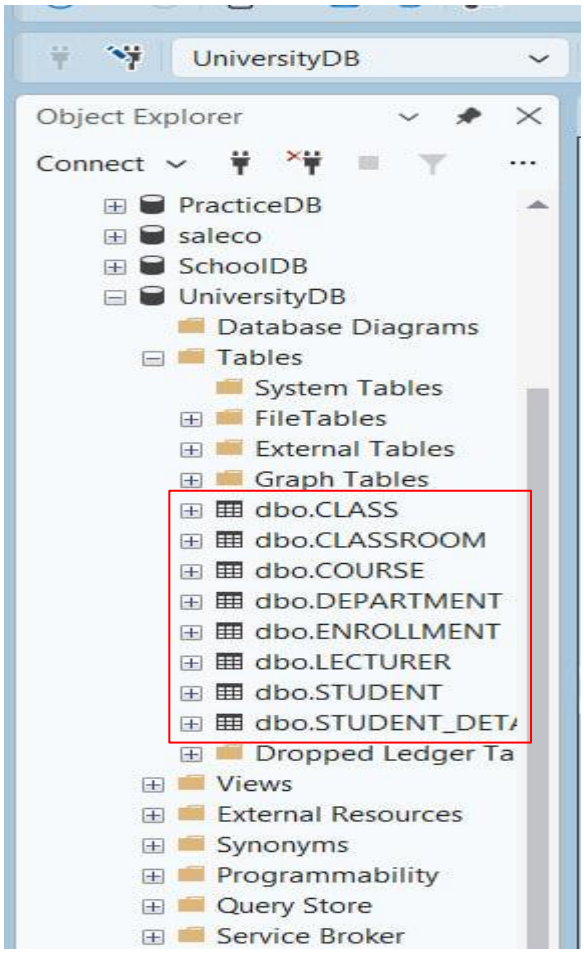
Results Messages

	STU_ID	CLASS_ID	ENROLL_DATE	ATTENDANCE_SCORE	MIDTERM_EXAM	FINAL_EXAM	STATUS	TOTAL_GRADE
1	S101	CL501	2023-09-01	95.00	85.00	90.00	Completed	88.50
2	S102	CL501	2023-09-01	88.00	78.00	82.00	Completed	81.00
3	S103	CL503	2023-09-05	100.00	92.00	95.00	Active	94.50
4	S104	CL502	2023-09-10	70.00	60.00	0.00	Withdrawn	30.00
5	S105	CL505	2024-01-15	92.00	88.00	85.00	Active	87.50
6	S201	CL501	2023-09-01	90.00	80.00	85.00	Active	83.00
7	S202	CL501	2023-09-01	85.00	75.00	80.00	Active	78.00
8	S203	CL503	2023-09-05	95.00	88.00	90.00	Active	91.00
9	S204	CL502	2023-09-10	80.00	70.00	75.00	Active	73.00
10	S205	CL504	2024-01-15	88.00	82.00	85.00	Active	84.00
11	S206	CL501	2023-09-01	92.00	85.00	88.00	Active	87.00
12	S207	CL505	2024-01-15	85.00	78.00	80.00	Active	79.00
13	S208	CL501	2023-09-01	88.00	80.00	82.00	Active	82.00
14	S209	CL503	2023-09-05	90.00	85.00	85.00	Active	86.00
15	S210	CL504	2024-01-15	82.00	75.00	78.00	Active	77.00

- Composite Primary Key: ប្រើ STU_ID និង CLASS_ID រួមគ្នាជាសោចម្បង ដើម្បីការពារកុំឱ្យនិស្សិតម្នាក់ចុះឈ្មោះរៀនថ្នាក់ដដែលលើសពីមួយដង។
- ការគ្រប់គ្រងពិន្ទុ (Grades): រួមមាន ATTENDANCE_SCORE (វត្តមាន), MIDTERM_EXAM (ពាក់កណ្តាលឆមាស), និង FINAL_EXAM (ផ្តាច់ព្រ័ត្ត)។
- លទ្ធផលចុងក្រោយ៖ TOTAL_GRADE បង្ហាញពីមធ្យមភាគពិន្ទុសរុប និង STATUS បង្ហាញពីស្ថានភាពសិក្សាក្នុងថ្នាក់នោះ (ដូចជា Completed, Active, Withdrawn)។

10.លទ្ធផលរចនាសម្ព័ន្ធក្នុងប្រព័ន្ធ (Object Explorer Results)

Object Explorer យើងឃើញថាប្រព័ន្ធបានបង្កើតតារាងចំនួន ៨ ស្របតាមការរចនា និងវិធានអាជីវកម្ម (Business Rules) រួមមាន៖ CLASS, CLASSROOM, COURSE, DEPARTMENT, ENROLLMENT, LECTURER, STUDENT, និង STUDENT_DETAIL។



ii. ALTER

Alter Table គឺជាការកែប្រែរចនាសម្ព័ន្ធតារាងបន្ទាប់ពីវាត្រូវបានបង្កើតរួច។ ក្នុងរបាយការណ៍ អ្នកអាចរៀបរាប់ថាវាត្រូវបានប្រើដើម្បីបន្ថែម Constraints (ដូចជា Foreign Key) ឬបន្ថែម Column ថ្មី។

ក្នុងករណីដែលតារាងត្រូវបានបង្កើតរួច ប៉ុន្តែយើងចង់បន្ថែមលក្ខខណ្ឌកំណត់ (Constraints) ឬកែប្រែ Column យើងប្រើប្រាស់ពាក្យបញ្ជា ALTER TABLE។

១. **ការបន្ថែម Foreign Key តាមក្រោយ**

ប្រសិនបើតារាងត្រូវបានបង្កើតដោយមិនទាន់មានទំនាក់ទំនង យើងប្រើកូដបែបនេះ៖

```
ALTER TABLE STUDENT
ADD CONSTRAINT FK_STUDENT_DEPT
FOREIGN KEY (DEPT_ID) REFERENCES DEPARTMENT(DEPT_ID);
```

Results | Messages

	STU_ID	STU_FNAME	STU_LNAME	GENDER	DOB	SHIFT_TIME	BATCH	DEGREE_TYPE	STU_TYPE	ENROLL_STATUS	UNIVERSITY_EMAIL	DEPT_ID	STU_PHONE
1	S101	John	Doe	M	2002-05-15	Morning	2023	Bachelors	Full-time	Active	john.doe@uni.edu	D01	012-111-222
2	S102	Jane	Smith	F	2001-11-20	Afternoon	2022	Bachelors	Full-time	Active	jane.smith@uni.edu	D01	098-333-444
3	S103	Michael	Brown	M	2003-02-10	Morning	2023	Associate	Part-time	Active	m.brown@uni.edu	D02	077-555-666
4	S104	Emily	Davis	F	2002-08-30	Evening	2023	Bachelors	Full-time	On Leave	e.davis@uni.edu	D01	010-444-555
5	S105	Chris	Wilson	M	2000-12-05	Morning	2021	Bachelors	Full-time	Graduated	c.wilson@uni.edu	D03	012-999-000
6	S201	Keo	Mony	M	2002-03-12	Morning	2023	Bachelors	Full-time	Active	keo.mony@uni.edu	D01	016-222-333
7	S202	Sok	Sreyneang	F	2001-05-20	Afternoon	2022	Bachelors	Full-time	Active	sok.srey@uni.edu	D01	011-777-888
8	S203	Chan	Vutha	M	2003-01-10	Morning	2023	Associate	Part-time	Active	chan.vutha@uni.edu	D02	085-555-444
9	S204	Nguon	Phally	F	2002-09-30	Evening	2023	Bachelors	Full-time	Active	nguon.phal@uni.edu	D01	093-111-222
10	S205	Heng	Rithy	M	2000-11-05	Morning	2021	Bachelors	Full-time	Active	heng.rithy@uni.edu	D03	099-888-777
11	S206	Vong	Sreypich	F	2002-07-15	Morning	2023	Bachelors	Full-time	Active	vong.pich@uni.edu	D04	081-333-555
12	S207	Chea	Sopheak	M	2001-08-25	Afternoon	2022	Bachelors	Full-time	Active	chea.pheak@uni.edu	D05	070-444-666
13	S208	Meas	Sokun	F	2003-04-18	Morning	2023	Associate	Full-time	Active	meas.sokun@uni.edu	D01	017-999-111
14	S209	Ros	Panha	M	2002-12-30	Evening	2023	Bachelors	Full-time	Active	ros.panha@uni.edu	D02	089-222-444
15	S210	Ouk	Dalin	F	2000-10-22	Morning	2021	Bachelors	Full-time	Active	ouk.dalin@uni.edu	D03	069-777-111

Query executed successfully. SAMAI-DB12 (17.0 RTM) SAMAI-DB12\User (74) UniversityD

- ALTER TABLE STUDENT: ជាការប្រាប់ប្រព័ន្ធថា យើងចង់កែប្រែរចនាសម្ព័ន្ធរបស់តារាងនិស្សិត។
- ADD CONSTRAINT: ជាការបន្ថែមលក្ខខណ្ឌកំណត់ថ្មីមួយ (ក្នុងកូដគឺឈ្មោះ FK_STUDENT_DEPT)។
- FOREIGN KEY (DEPT_ID): កំណត់ថា Column DEPT_ID ក្នុងតារាង Student គឺជាសោចម្លង។

- REFERENCES DEPARTMENT(DEPT_ID): ជាការចង្អុលបង្ហាញថាវាត្រូវភ្ជាប់ទៅកាន់ Column DEPT_ID ដែលជាសោចម្បងនៅក្នុងតារាងដេប៉ាតឺម៉ង់។

២. ការបន្ថែម Column ថ្មី

ឧទាហរណ៍៖ បន្ថែម Column សម្រាប់កត់ត្រាថ្ងៃខែឆ្នាំចូលរៀនក្នុងតារាង STUDENT។

```
ALTER TABLE STUDENT
ADD Enroll_Date DATE;
```

Results | Messages

	STU_ID	STU_FNAME	STU_LNAME	GENDER	DOB	SHIFT_TIME	BATCH	DEGREE_TYPE	STU_TYPE	ENROLL_STATUS	UNIVERSITY_EMAIL	DEPT_ID	STU_PHONE	Enroll_Date
2	S102	Jane	Smith	F	2001-11-20	Afternoon	2022	Bachelors	Full-time	Active	jane.smith@uni.edu	D01	098-333-444	NULL
3	S103	Michael	Brown	M	2003-02-10	Morning	2023	Associate	Part-time	Active	m.brown@uni.edu	D02	077-555-666	NULL
4	S104	Emily	Davis	F	2002-08-30	Evening	2023	Bachelors	Full-time	On Leave	e.davis@uni.edu	D01	010-444-555	NULL
5	S105	Chris	Wilson	M	2000-12-05	Morning	2021	Bachelors	Full-time	Graduated	c.wilson@uni.edu	D03	012-999-000	NULL
6	S201	Keo	Mony	M	2002-03-12	Morning	2023	Bachelors	Full-time	Active	keo.mony@uni.edu	D01	016-222-333	NULL
7	S202	Sok	Sreyneang	F	2001-05-20	Afternoon	2022	Bachelors	Full-time	Active	sok.srey@uni.edu	D01	011-777-888	NULL
8	S203	Chan	Vutha	M	2003-01-10	Morning	2023	Associate	Part-time	Active	chan.vutha@uni.edu	D02	085-555-444	NULL
9	S204	Nguon	Phally	F	2002-09-30	Evening	2023	Bachelors	Full-time	Active	nguon.phal@uni.edu	D01	093-111-222	NULL
10	S205	Heng	Rithy	M	2000-11-05	Morning	2021	Bachelors	Full-time	Active	heng.rithy@uni.edu	D03	099-888-777	NULL
11	S206	Vong	Sreypich	F	2002-07-15	Morning	2023	Bachelors	Full-time	Active	vong.pich@uni.edu	D04	081-333-555	NULL
12	S207	Chea	Sopheak	M	2001-08-25	Afternoon	2022	Bachelors	Full-time	Active	chea.pheak@uni.edu	D05	070-444-666	NULL
13	S208	Meas	Sokun	F	2003-04-18	Morning	2023	Associate	Full-time	Active	meas.sokun@uni.edu	D01	017-999-111	NULL
14	S209	Ros	Panha	M	2002-12-30	Evening	2023	Bachelors	Full-time	Active	ros.panha@uni.edu	D02	089-222-444	NULL
15	S210	Ouk	Dalin	F	2000-10-22	Morning	2021	Bachelors	Full-time	Active	ouk.dalin@uni.edu	D03	069-777-111	NULL

Query executed successfully. | SAMAI-DB12 (17.0 RTM) | SAMAI-DB12\User (74) | UniversityDB | 00:00:00 | Row: 1

- Structure Change: នៅក្នុងរូបភាព យើងឃើញថា Column ថ្មីដែលមានឈ្មោះថា Enroll_Date ត្រូវបានបង្កើតឡើងនៅខាងចុងគេបំផុតនៃតារាង STUDENT។
- Data State: ទិន្នន័យក្នុង Column ថ្មីនេះបង្ហាញថា NULL (ទទេ) ព្រោះវាជា Column ដែលទើបតែបន្ថែមថ្មី និងមិនទាន់មានការបញ្ចូលទិន្នន័យ (Insert/Update) ចូលទៅក្នុងនោះនៅឡើយ។

៣. ការកែប្រែប្រភេទទិន្នន័យ (Modify Data Type)

នៅក្នុងដំណើរការអភិវឌ្ឍ ប្រសិនបើយើងរកឃើញថាទំហំអក្សរដែលបានកំណត់ពីមុនមិនគ្រប់គ្រាន់សម្រាប់ផ្ទុកទិន្នន័យ (ដូចជាឈ្មោះនិស្សិតវែងពេក) យើងអាចប្រើប្រាស់ ALTER COLUMN ដើម្បីពង្រីកទំហំនោះ។

ឧទាហរណ៍៖ ប្តូរទំហំអក្សរនៃឈ្មោះនិស្សិតឱ្យវែងជាងមុន។

```
ALTER TABLE STUDENT
ALTER COLUMN STU_FNAME VARCHAR(100);
```

Results | Messages

	STU_ID	STU_FNAME	STU_LNAME	GENDER	DOB	SHIFT_TIME	BATCH	DEGREE_TYPE	STU_TYPE	ENROLL_STATUS	UNIVERSITY_EMAIL	DEPT_ID	STU_PHONE
1	S101	John	Doe	M	2002-05-15	Morning	2023	Bachelors	Full-time	Active	john.doe@uni.edu	D01	012-111-222
2	S102	Jane	Smith	F	2001-11-20	Afternoon	2022	Bachelors	Full-time	Active	jane.smith@uni.edu	D01	098-333-444
3	S103	Michael	Brown	M	2003-02-10	Morning	2023	Associate	Part-time	Active	m.brown@uni.edu	D02	077-555-666
4	S104	Emily	Davis	F	2002-08-30	Evening	2023	Bachelors	Full-time	On Leave	e.davis@uni.edu	D01	010-444-555
5	S105	Chris	Wilson	M	2000-12-05	Morning	2021	Bachelors	Full-time	Graduated	c.wilson@uni.edu	D03	012-999-000
6	S201	Keo	Mony	M	2002-03-12	Morning	2023	Bachelors	Full-time	Active	keo.mony@uni.edu	D01	016-222-333
7	S202	Sok	Sreyneang	F	2001-05-20	Afternoon	2022	Bachelors	Full-time	Active	sok.srey@uni.edu	D01	011-777-888
8	S203	Chan	Vutha	M	2003-01-10	Morning	2023	Associate	Part-time	Active	chan.vutha@uni.edu	D02	085-555-444
9	S204	Nguon	Phally	F	2002-09-30	Evening	2023	Bachelors	Full-time	Active	nguon.phal@uni.edu	D01	093-111-222
10	S205	Heng	Rithy	M	2000-11-05	Morning	2021	Bachelors	Full-time	Active	heng.rithy@uni.edu	D03	099-888-777
11	S206	Vong	Sreypich	F	2002-07-15	Morning	2023	Bachelors	Full-time	Active	vong.pich@uni.edu	D04	081-333-555
12	S207	Chea	Sopheak	M	2001-08-25	Afternoon	2022	Bachelors	Full-time	Active	chea.pheak@uni.edu	D05	070-444-666
13	S208	Meas	Sokun	F	2003-04-18	Morning	2023	Associate	Full-time	Active	meas.sokun@uni.edu	D01	017-999-111
14	S209	Ros	Panha	M	2002-12-30	Evening	2023	Bachelors	Full-time	Active	ros.panha@uni.edu	D02	089-222-444
15	S210	Ouk	Dalin	F	2000-10-22	Morning	2021	Bachelors	Full-time	Active	ouk.dalin@uni.edu	D03	069-777-111

Query executed successfully. SAMAI-DB12 (17.0 RTM) SAMAI-DB12\User (74) Univer

- ALTER COLUMN: ប្រើសម្រាប់ផ្លាស់ប្តូរលក្ខណៈ ឬប្រភេទទិន្នន័យនៃ Column ដែលមានស្រាប់។
- VARCHAR(100): កែប្រែទំហំផ្ទុកតួអក្សរឱ្យឡើងដល់ ១០០ ដើម្បីបង្ការកុំឱ្យទិន្នន័យដាច់ (Truncation Error) នៅពេលបញ្ចូលឈ្មោះដែលវែងខ្លាំង។

ការលុបតារាង (Drop Table) ក្នុងប្រព័ន្ធ Database

ពាក្យបញ្ជា DROP TABLE គឺជាផ្នែកមួយនៃភាសា DDL ដែលមានឥទ្ធិពលខ្លាំងបំផុត ព្រោះវានឹងលុបទាំងទិន្នន័យ (Data) និងរចនាសម្ព័ន្ធ (Structure/Schema) របស់តារាងចេញពីអង្គចងចាំជាស្ថាពរ។

ក. ការប្រើប្រាស់ពាក្យបញ្ជាជាមូលដ្ឋាន

Syntax សម្រាប់លុបតារាងធម្មតា៖

```
DROP TABLE ENROLLMENT;
```

ចំណាំ៖ នៅពេល Execute កូដនេះ ប្រព័ន្ធនឹងមិនសួរបញ្ជាក់ (Confirm) ឡើយ ដូច្នេះរាល់ទិន្នន័យដែលបាត់បង់មិនអាចស្រាវជ្រាវមកវិញបានទេ (លើកលែងតែមានការ Back up)។

ខ. ឧបសគ្គនៃទំនាក់ទំនង (Referential Integrity Constraints)

- *បម្រាម៖* យើងមិនអាចលុបតារាង DEPARTMENT បានទេ ប្រសិនបើតារាង STUDENT នៅមានទិន្នន័យដែលចង្អុលបង្ហាញ (Reference) មកកាន់វា។
- *ដំណោះស្រាយ៖* ត្រូវលុបតារាងកូន (Child Table) ដែលមានសោចម្លង (Foreign Key) ចេញមុនគេបង្អស់។

គ. លំដាប់លំដោយនៃការលុបដែលត្រឹមត្រូវ (Step-by-Step)

ដើម្បីលុបឱ្យបានជោគជ័យក្នុងករណីសិក្សានេះ ត្រូវអនុវត្តតាមលំដាប់នេះ៖

- *លុបមុនគេ (Child Tables):* ENROLLMENT -> STUDENT_DETAIL -> CLASS -> STUDENT
- *លុបក្រោយគេ (Parent Tables):* COURSE -> LECTURER -> CLASSROOM -> DEPARTMENT

ឃ. ការប្រើប្រាស់លក្ខខណ្ឌ IF EXISTS (បច្ចេកទេសកម្រិតខ្ពស់)

ដើម្បីការពារកុំឱ្យមានកំហុស (Error) នៅពេលប្រព័ន្ធរកមិនឃើញតារាងដែលត្រូវលុប យើងគួរប្រើ៖

```
DROP TABLE IF EXISTS ENROLLMENT;
```

អត្ថប្រយោជន៍៖ កូដនេះនឹងពិនិត្យមើលជាមុន បើមានតារាងទើបលុប បើមិនមានទេវានឹងមិនលោត Error ឡើយ។

TRUNCATE

TRUNCATE គឺជាពាក្យបញ្ជាដ៏សំខាន់មួយក្នុង SQL ដែលប្រើសម្រាប់ "លុបទិន្នន័យទាំងអស់ចេញពីតារាង ប៉ុន្តែរក្សារចនាសម្ព័ន្ធតារាងទុកដដែល"។

- *ការសម្អាតទិន្នន័យដោយប្រើ* TRUNCATE TABLE

ពាក្យបញ្ជា TRUNCATE TABLE ត្រូវបានប្រើប្រាស់នៅពេលដែលយើងចង់លុបរាល់ទិន្នន័យ (Records) ទាំងអស់ចេញពីតារាងឱ្យនៅទទេ ប៉ុន្តែយើងមិនចង់លុបតារាងនោះចោលឡើយ។

ក. ការប្រើប្រាស់ពាក្យបញ្ជា

```
TRUNCATE TABLE ENROLLMENT;
```

ខ. ភាពខុសគ្នារវាង TRUNCATE និង DROP

- TRUNCATE: លុបតែទិន្នន័យ (ដូចជាការជូតក្តារខៀនឱ្យស្អាត ដើម្បីសរសេរថ្មី)។ រចនាសម្ព័ន្ធ Column និង Constraints នៅរក្សាដដែល។
- DROP: លុបទាំងទិន្នន័យ និងរចនាសម្ព័ន្ធ (ដូចជាការដកក្តារខៀនចេញពីជញ្ជាំងតែម្តង)។

គ. លក្ខណៈពិសេស និងការប្រុងប្រយ័ត្ន

- ល្បឿនលឿន: វាដំណើរការលឿនជាង DELETE ធម្មតា ព្រោះវាមិនកត់ត្រាការលុបជួរនីមួយៗក្នុង Log ឡើយ។
- បម្រាមចំពោះ Foreign Key: យើងមិនអាច TRUNCATE តារាងមេ (Parent Table) បានឡើយ ប្រសិនបើតារាងនោះកំពុងត្រូវបានភ្ជាប់ដោយ Foreign Key ពីតារាងផ្សេង ទោះបីជាតារាងនោះមិនទាន់មានទិន្នន័យក៏ដោយ។
- Reset Identity: ប្រសិនបើតារាងមាន Column ប្រភេទ IDENTITY (លេខរៀងស្វ័យប្រវត្តិ) វានឹងកំណត់លេខរៀងឱ្យចាប់ផ្តើមពីលេខ ១ ឡើងវិញ។

ក្រៅពីមូលដ្ឋានគ្រឹះនៃ DML និង DDL ដែលបានអនុវត្តក្នុងរបាយការណ៍នេះ យើងក៏បានស្វែងយល់ជាមុននូវសមាសភាគសំខាន់ៗចំនួន ២ ទៀតគឺ Transaction Control Language (TCL)និង Data Control Language (DCL) ដែលនឹងត្រូវសិក្សាលម្អិតនៅក្នុងមេរៀនទី៨ ខាងមុខ។

IV. សេចក្តីសន្និដ្ឋាន (Conclusion)

ឆ្លងកាត់ការសិក្សា និងការអនុវត្តជាក់ស្តែងលើរបាយការណ៍ទី១ ស្តីពី «ការណែនាំអំពីភាសាសំណួររចនាសម្ព័ន្ធ (Introduction to SQL)» យើងអាចសរុបលទ្ធផលសំខាន់ៗបានដូចខាងក្រោម៖

- ការយល់ដឹងពីមូលដ្ឋានគ្រឹះ: យើងបានយល់ច្បាស់ពីតួនាទីរបស់ SQL ក្នុងនាមជាភាសាស្តង់ដារសម្រាប់គ្រប់គ្រង និងទាក់ទងជាមួយ Relational Database Management Systems (RDBMS) ដូចជា Microsoft SQL Server ជាដើម ដែលវាជាបេះដូងនៃប្រព័ន្ធព័ត៌មានវិទ្យានាពេលបច្ចុប្បន្ន។
- ការគ្រប់គ្រងទិន្នន័យ (DML): តាមរយៈការអនុវត្តផ្ទាល់លើពាក្យបញ្ជា SELECT, INSERT, UPDATE និង DELETE យើងទទួលបានជំនាញក្នុងការទាញយក និងកែប្រែទិន្នន័យតាមតម្រូវការអាជីវកម្ម។ ជាពិសេស ការប្រើប្រាស់ Aggregate Functions (SUM, AVG, COUNT, MIN, MAX) បានជួយឱ្យការវិភាគទិន្នន័យមានភាពរហ័ស និងត្រឹមត្រូវ។
- ការរៀបចំរចនាសម្ព័ន្ធ (DDL): យើងបានរៀនពីរបៀបបង្កើត និងរៀបចំ Database តាមរយៈពាក្យបញ្ជា CREATE, ALTER, DROP និង TRUNCATE។ ការយល់ដឹងពីការកំណត់ Constraints (ដូចជា Primary Key និង Foreign Key) ក្នុងករណីសិក្សា « ប្រព័ន្ធព័ត៌មាននិស្សិត» បានបង្ហាញពីសារៈសំខាន់នៃភាពត្រឹមត្រូវនៃទិន្នន័យ (Data Integrity) និងទំនាក់ទំនងរវាងតារាង។
- ប្រសិទ្ធភាពនៃការអនុវត្ត: ការដំឡើង និងការប្រើប្រាស់កម្មវិធី SQL Server Management Studio (SSMS) បានផ្តល់នូវបទពិសោធន៍ជាក់ស្តែងក្នុងការគ្រប់គ្រង Database ក្នុងបរិយាកាសការងារពិតប្រាកដ។

ជារួម របាយការណ៍នេះបានផ្តល់នូវមូលដ្ឋានគ្រឹះយ៉ាងរឹងមាំក្នុងការប្រើប្រាស់ SQL ដែលជាជំនាញចាំបាច់សម្រាប់អ្នកអភិវឌ្ឍន៍កម្មវិធី និងអ្នកគ្រប់គ្រងប្រព័ន្ធទិន្នន័យ ដើម្បីកសាងប្រព័ន្ធព័ត៌មានដែលមានប្រសិទ្ធភាព និងសុវត្ថិភាព។

ពាក្យថ្លែងអំណរគុណ

"យើងខ្ញុំជាសមាជិកក្រុមទី១៤ សូមថ្លែងអំណរគុណយ៉ាងជ្រាលជ្រៅចំពោះ លោកគ្រូ Kheam Hong ដែលបានផ្តល់ចំណេះដឹង និងការណែនាំដ៏ល្អៗក្នុងមេរៀន SQL នេះ។ ទន្ទឹមនឹងនេះ យើងខ្ញុំក៏សូមអរគុណដល់មិត្តនិស្សិតទាំងអស់ចំពោះការយកចិត្តទុកដាក់អានរបាយការណ៍មួយនេះ។"

សូមអរគុណ!