welcome back

SQL Essentials

Roadmap

- RDBMS
  ER Model
- SQL Server
  DDL
- DML
  Operator
- Function
  SQL Clause
- Join data
- Sub Query
- Best
  Practice

Previous lecture

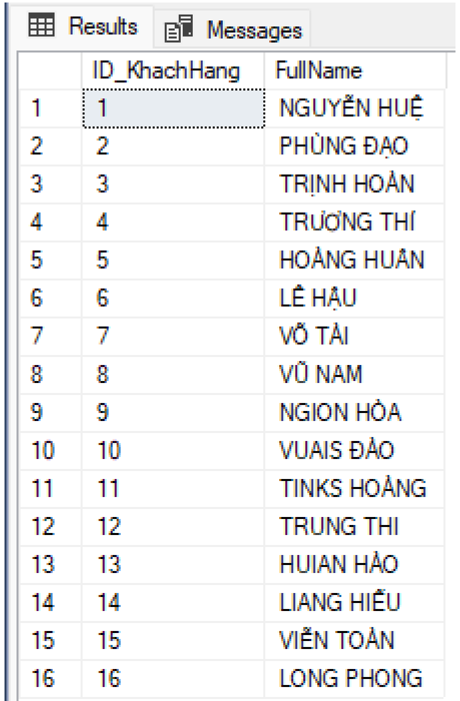
- Sub queries
- Advance operator
- Rules of sub query
- Practice

What we will explore today?

- Paging data
- Backup the database
- Stored Procedures
- Trigger
- Index Demo
- SQL Injection
- Best pratice

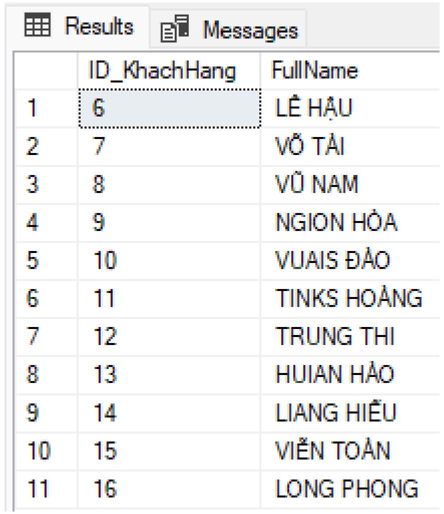
```
USE LECTURE6_PRATICE_JOIN
```

```
SELECT ID_KhachHang, FullName
FROM Customer
ORDER BY ID_KhachHang
```



	ID_KhachHang	FullName
1	1	NGUYỄN HUỆ
2	2	PHÙNG ĐẠO
3	3	TRỊNH HOÀN
4	4	TRƯƠNG THÍ
5	5	HOÀNG HUẤN
6	6	LÊ HẬU
7	7	VÕ TÀI
8	8	VŨ NAM
9	9	NGION HÒA
10	10	VUAIS ĐẢO
11	11	TINKS HOÀNG
12	12	TRUNG THI
13	13	HUIAN HẢO
14	14	LIANG HIẾU
15	15	VIẾN TOÀN
16	16	LONG PHONG

```
SELECT ID_KhachHang, FullName
FROM Customer
ORDER BY ID_KhachHang
OFFSET 5 ROWS
```

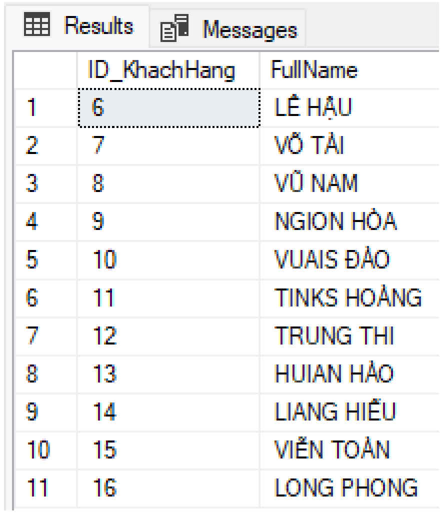


	ID_KhachHang	FullName
1	6	LÊ HẬU
2	7	VÕ TÀI
3	8	VŨ NAM
4	9	NGION HÒA
5	10	VUAIS ĐẢO
6	11	TINKS HOÀNG
7	12	TRUNG THI
8	13	HUIAN HẢO
9	14	LIANG HIẾU
10	15	VIẾN TOÀN
11	16	LONG PHONG

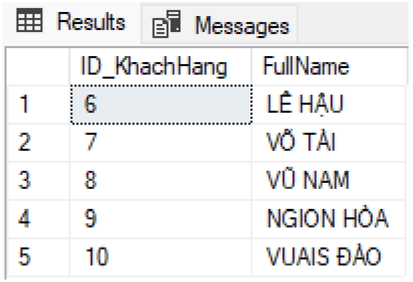
```sql
USE LECTURE6_PRATICE_JOIN
```

```sql
SELECT ID_KhachHang, FullName
FROM Customer
ORDER BY ID_KhachHang
OFFSET 5 ROWS
```

```sql
SELECT ID_KhachHang, FullName
FROM Customer
ORDER BY ID_KhachHang
OFFSET 5 ROWS
FETCH NEXT 5 ROWS ONLY
```



	ID_KhachHang	FullName
1	6	LÊ HẬU
2	7	VÕ TÀI
3	8	VŨ NAM
4	9	NGION HÒA
5	10	VUAIS ĐẢO
6	11	TINKS HOẢNG
7	12	TRUNG THI
8	13	HUIAN HẢO
9	14	LIANG HIẾU
10	15	VIỄN TOẢN
11	16	LONG PHONG

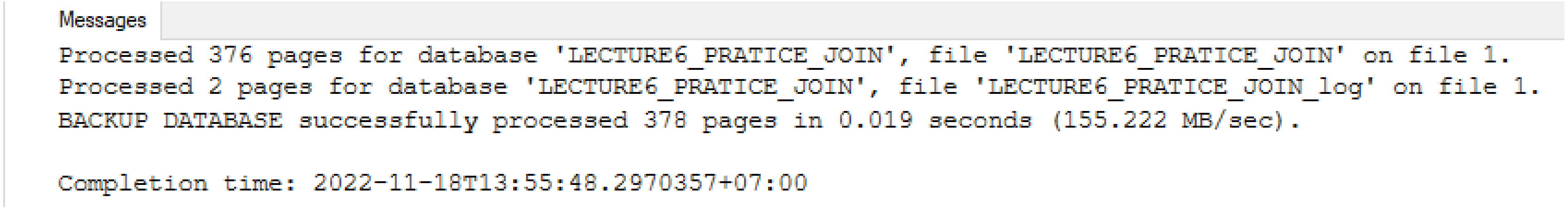


	ID_KhachHang	FullName
1	6	LÊ HẬU
2	7	VÕ TÀI
3	8	VŨ NAM
4	9	NGION HÒA
5	10	VUAIS ĐẢO

Backup the database

```
BACKUP DATABASE databasename
TO DISK = 'filepath';
```

```
BACKUP DATABASE LECTURE6_PRATICE_JOIN
TO DISK = 'D:\LECTURE6_PRATICE_JOIN_BackUp.bak';
```



```
Processed 376 pages for database 'LECTURE6_PRATICE_JOIN', file 'LECTURE6_PRATICE_JOIN' on file 1.
Processed 2 pages for database 'LECTURE6_PRATICE_JOIN', file 'LECTURE6_PRATICE_JOIN_log' on file 1.
BACKUP DATABASE successfully processed 378 pages in 0.019 seconds (155.222 MB/sec).

Completion time: 2022-11-18T13:55:48.2970357+07:00
```

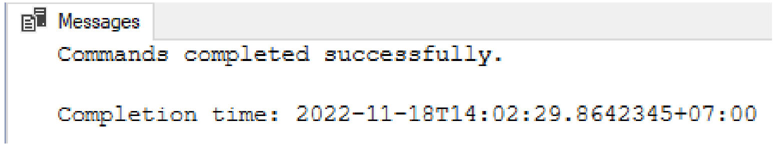
Stored Procedures

- A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.
- You can also pass parameters to a stored procedure

Stored Procedures syntax

```
CREATE PROCEDURE procedure_name
AS
sql_statement
GO;
```

```
USE LECTURE6_PRATICE_JOIN;
CREATE PROCEDURE MyFirstStoreProcedure
AS
SELECT ID_KHACHHANG,  FullName
FROM Customer
GO;
```



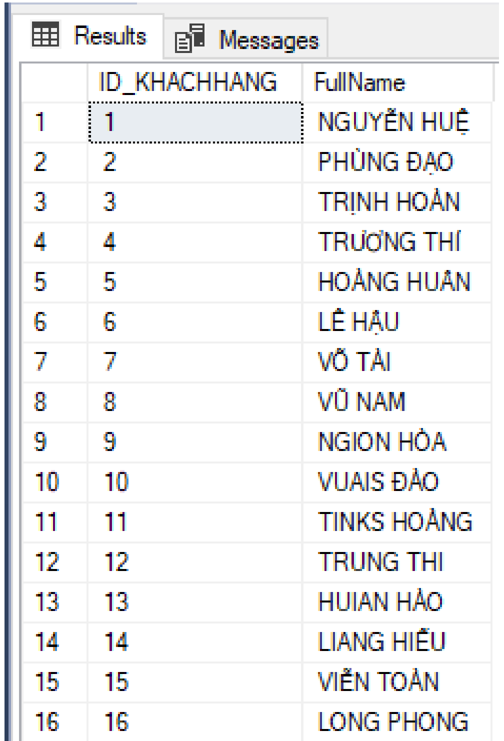
```
Commands completed successfully.

Completion time: 2022-11-18T14:02:29.8642345+07:00
```

“Run” Stored Procedures

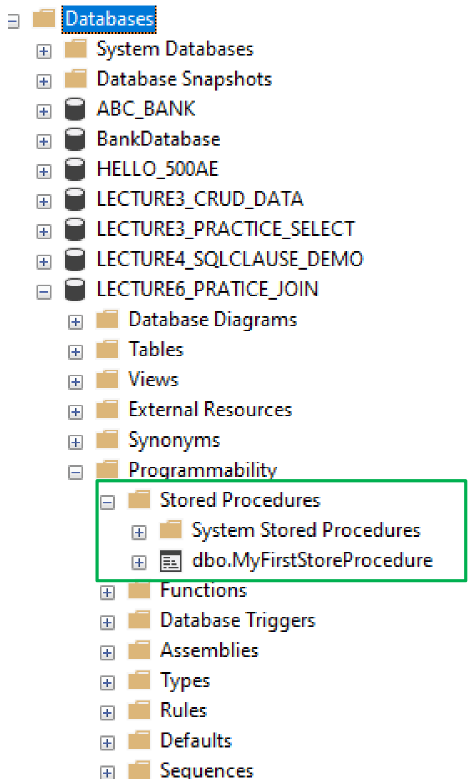
```
EXEC procedure_name;
```

```
EXEC MyFirstStoreProcedure
```



	ID_KHACHHANG	FullName
1	1	NGUYỄN HUỆ
2	2	PHÙNG ĐẠO
3	3	TRỊNH HOÀN
4	4	TRƯƠNG THÍ
5	5	HOÀNG HUÂN
6	6	LÊ HẬU
7	7	VÕ TÀI
8	8	VŨ NAM
9	9	NGION HÒA
10	10	VUAIS ĐẢO
11	11	TINKS HOÀNG
12	12	TRUNG THI
13	13	HUIAN HẢO
14	14	LIANG HIẾU
15	15	VIỄN TOẢN
16	16	LONG PHONG

Where is it?


Databases
System Databases
Database Snapshots
ABC_BANK
BankDatabase
HELLO_500AE
LECTURE3_CRUD_DATA
LECTURE3_PRACTICE_SELECT
LECTURE4_SQLCLAUSE_DEMO
LECTURE6_PRATICE_JOIN
Database Diagrams
Tables
Views
External Resources
Synonyms
Programmability
Stored Procedures
System Stored Procedures
dbo.MyFirstStoreProcedure
Functions
Database Triggers
Assemblies
Types
Rules
Defaults
Sequences

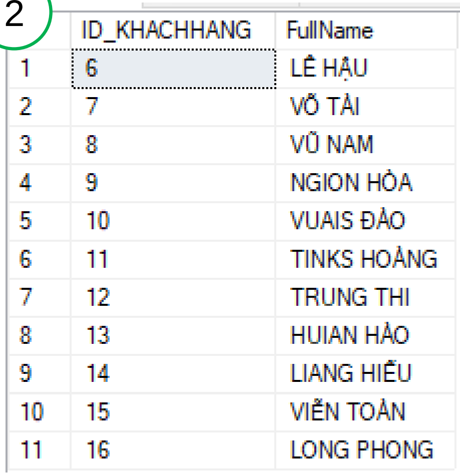
With parameters

```
USE LECTURE6_PRATICE_JOIN
GO
CREATE PROCEDURE MyFirstStoreProcedureWithParam @IDKhachGreater int
AS
SELECT ID_KHACHHANG,  FullName
FROM Customer
WHERE ID_KhachHang >  @IDKhachGreater
GO
```

1

```
EXEC MyFirstStoreProcedureWithParam @IDKhachGreater = 5
```

2



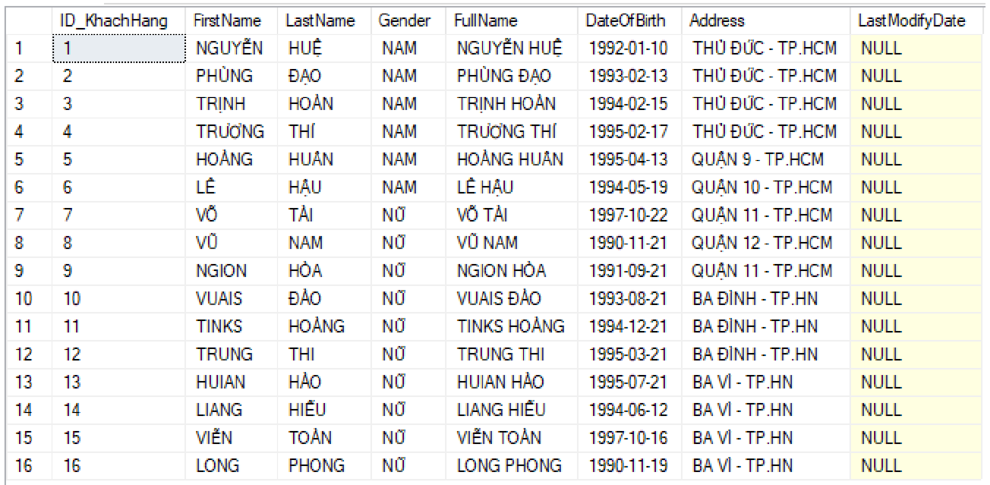
	ID_KHACHHANG	FullName
1	6	LÊ HẬU
2	7	VÕ TÀI
3	8	VŨ NAM
4	9	NGION HÒA
5	10	VUAIS ĐÀO
6	11	TINKS HOÀNG
7	12	TRUNG THI
8	13	HUIAN HÀO
9	14	LIANG HIẾU
10	15	VIỄN TOÀN
11	16	LONG PHONG

TRIGGER

- A special kind of stored procedure, which “reacts” to certain actions we make in the database. The main idea behind triggers is that they always perform an action in case some event happens.

Add LastModifyDate column

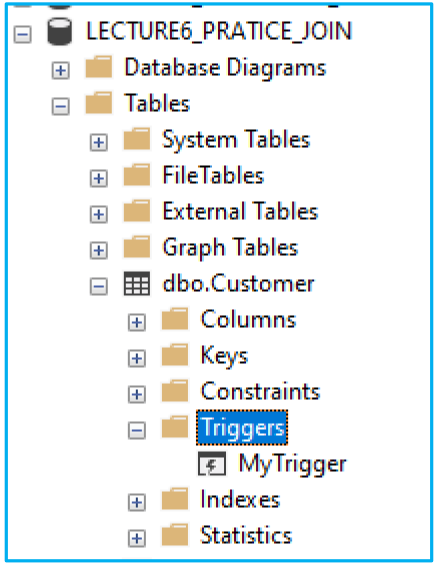
```sql
ALTER TABLE Customer
ADD LastModifyDate datetime
```



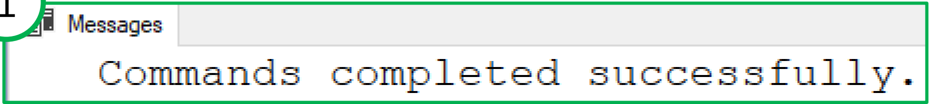
	ID_KhachHang	FirstName	LastName	Gender	FullName	DateOfBirth	Address	LastModifyDate
1	1	NGUYỄN	HUỆ	NAM	NGUYỄN HUỆ	1992-01-10	THỦ ĐỨC - TP.HCM	NULL
2	2	PHÙNG	ĐẠO	NAM	PHÙNG ĐẠO	1993-02-13	THỦ ĐỨC - TP.HCM	NULL
3	3	TRỊNH	HOÀN	NAM	TRỊNH HOÀN	1994-02-15	THỦ ĐỨC - TP.HCM	NULL
4	4	TRƯƠNG	THÍ	NAM	TRƯƠNG THÍ	1995-02-17	THỦ ĐỨC - TP.HCM	NULL
5	5	HOẢNG	HUÂN	NAM	HOẢNG HUÂN	1995-04-13	QUẬN 9 - TP.HCM	NULL
6	6	LÊ	HẬU	NAM	LÊ HẬU	1994-05-19	QUẬN 10 - TP.HCM	NULL
7	7	VÕ	TÀI	NỮ	VÕ TÀI	1997-10-22	QUẬN 11 - TP.HCM	NULL
8	8	VŨ	NAM	NỮ	VŨ NAM	1990-11-21	QUẬN 12 - TP.HCM	NULL
9	9	NGION	HÒA	NỮ	NGION HÒA	1991-09-21	QUẬN 11 - TP.HCM	NULL
10	10	VUAIS	ĐÀO	NỮ	VUAIS ĐÀO	1993-08-21	BA ĐÌNH - TP.HN	NULL
11	11	TINKS	HOẢNG	NỮ	TINKS HOẢNG	1994-12-21	BA ĐÌNH - TP.HN	NULL
12	12	TRUNG	THI	NỮ	TRUNG THI	1995-03-21	BA ĐÌNH - TP.HN	NULL
13	13	HUIAN	HẢO	NỮ	HUIAN HẢO	1995-07-21	BA VÌ - TP.HN	NULL
14	14	LIANG	HIẾU	NỮ	LIANG HIẾU	1994-06-12	BA VÌ - TP.HN	NULL
15	15	VIẾN	TOÀN	NỮ	VIẾN TOÀN	1997-10-16	BA VÌ - TP.HN	NULL
16	16	LONG	PHONG	NỮ	LONG PHONG	1990-11-19	BA VÌ - TP.HN	NULL

Create Trigger

```
IF OBJECT_ID (N'MyTrigger') IS NOT NULL
    DROP TRIGGER MyTrigger;
GO
CREATE TRIGGER MyTrigger ON Customer
AFTER UPDATE
AS
BEGIN
    SET NOCOUNT ON;
    UPDATE Customer
    SET LastModifyDate = GETDATE()
    WHERE ID_KhachHang IN (SELECT i.ID_KhachHang
                            FROM inserted AS i);
END
```



1



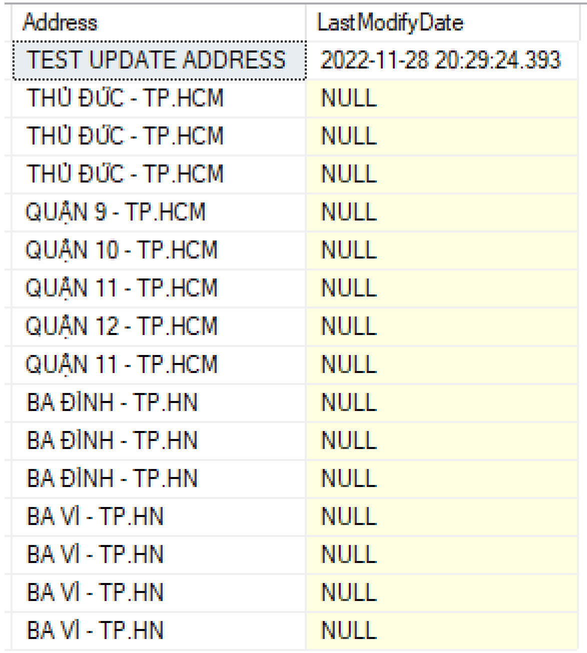
```
Commands completed successfully.
```

Update 1 record

```
UPDATE Customer
SET Address = N'TEST UPDATE ADDRESS'
WHERE ID_KhachHang = 1
```

1

```
SELECT Address, LastModifyDate
FROM Customer
```

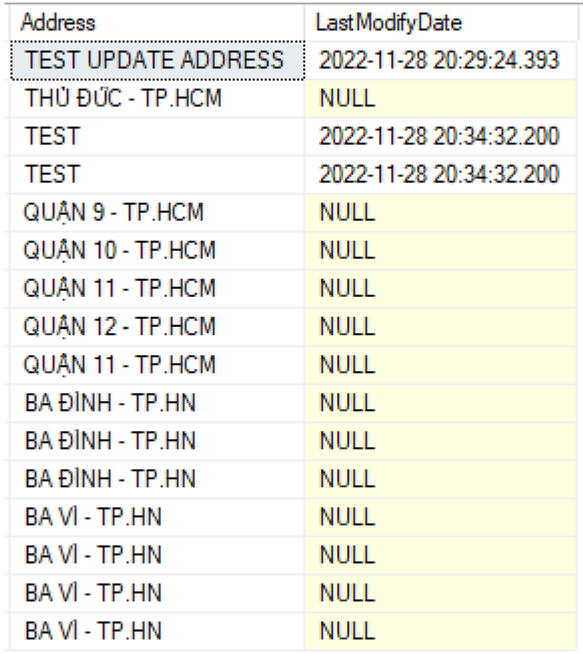


Address	LastModifyDate
TEST UPDATE ADDRESS	2022-11-28 20:29:24.393
THỦ ĐỨC - TP.HCM	NULL
THỦ ĐỨC - TP.HCM	NULL
THỦ ĐỨC - TP.HCM	NULL
QUẬN 9 - TP.HCM	NULL
QUẬN 10 - TP.HCM	NULL
QUẬN 11 - TP.HCM	NULL
QUẬN 12 - TP.HCM	NULL
QUẬN 11 - TP.HCM	NULL
BA ĐÌNH - TP.HN	NULL
BA ĐÌNH - TP.HN	NULL
BA ĐÌNH - TP.HN	NULL
BA VÌ - TP.HN	NULL
BA VÌ - TP.HN	NULL
BA VÌ - TP.HN	NULL
BA VÌ - TP.HN	NULL

Update 2 records

```sql
UPDATE Customer
SET Address = N'TEST'
WHERE ID_KhachHang = 3 OR ID_KhachHang = 4
```

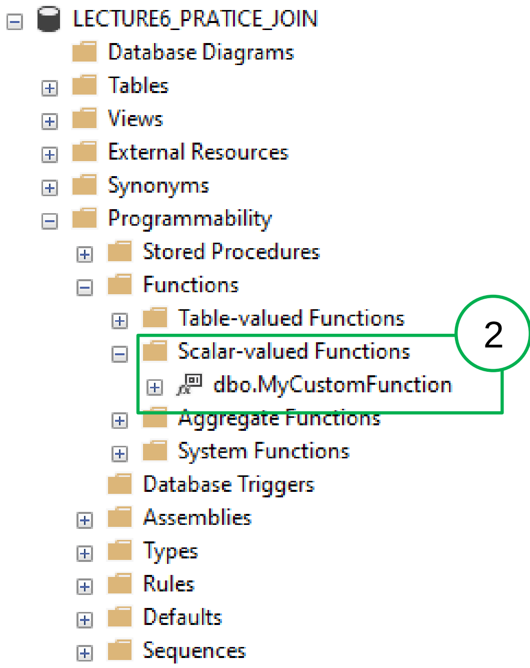
```sql
SELECT Address, LastModifyDate
FROM Customer
```

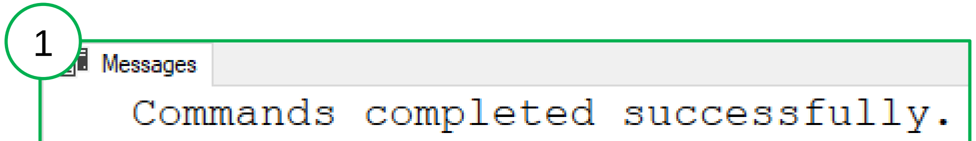


Address	LastModifyDate
TEST UPDATE ADDRESS	2022-11-28 20:29:24.393
THỦ ĐỨC - TP.HCM	NULL
TEST	2022-11-28 20:34:32.200
TEST	2022-11-28 20:34:32.200
QUẬN 9 - TP.HCM	NULL
QUẬN 10 - TP.HCM	NULL
QUẬN 11 - TP.HCM	NULL
QUẬN 12 - TP.HCM	NULL
QUẬN 11 - TP.HCM	NULL
BA ĐÌNH - TP.HN	NULL
BA ĐÌNH - TP.HN	NULL
BA ĐÌNH - TP.HN	NULL
BA VÌ - TP.HN	NULL
BA VÌ - TP.HN	NULL
BA VÌ - TP.HN	NULL
BA VÌ - TP.HN	NULL

user-defined functions

```
IF OBJECT_ID (N'dbo.MyCustomFunction', N'FN') IS NOT NULL
    DROP FUNCTION MyCustomFunction;
GO
CREATE FUNCTION MyCustomFunction(@YourMoney int)
RETURNS nvarchar(50)
AS
BEGIN
    DECLARE @Result nvarchar(50)
        IF (@YourMoney >= 1 * 1000 * 1000  * 1000) -- 1 tỷ
            SET @Result = N'bạn xứng đáng có 10 người yêu';
        ELSE
            SET @Result = N'nỗ lực thì sẽ có ngày thành công';
    RETURN @Result
END;
```

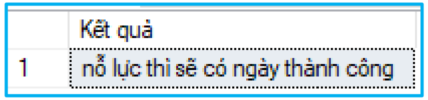




```
Commands completed successfully.
```

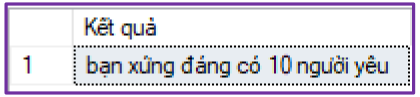
Call it

```sql
SELECT dbo.MyCustomFunction(200) AS 'Kết quả'
```



	Kết quả
1	nỗ lực thì sẽ có ngày thành công

```sql
SELECT dbo.MyCustomFunction(1000000000) AS 'Kết quả'
```



	Kết quả
1	bạn xứng đáng có 10 người yêu

Index

- Indexes are used to retrieve data from the database more quickly than otherwise.
- The users cannot see the indexes, they are just used to speed up searches/queries.

Index demo

- open file “LECTURE7_INDEX_DEMO.sql”

SQL Injection

- open file "LECTURE7_SQL_injection.sql"

Benifit of Coding Standards

- Enhanced Efficiency
- Risk of project failure is reduced
- Minimal Complexity
- Easy to Maintain
- Bug Rectification
- A Comprehensive Look
- Cost-Efficient

SQL Comment

- Always use comment to explain your code.
- Use natural/human language in comment to easy understand.
- All comments should be same format.
- Break comment line to avoid horizontal scroll bar.

Naming conventions

- Must be simple, meaningful & do not conflix with system name.
- Names must begin with a letter and may not end with an underscore.

Format code

- Always use **UPPERCASE** for the reserved keywords like SELECT and WHERE.
- Break line to avoid horizontal scroll bar. It recommended that start line with **KEYWORD**

Avoid SELECT *

```sql
-- Bad query
SELECT *
FROM table_name;
```

```sql
-- Better query
SELECT col1, col2, col3
FROM table_name;
```

DISTINCT

```sql
-- Bad query
SELECT DISTINCT ID, FirstName, LastName
FROM Customers;
```

```sql
-- Better query
SELECT ID, FirstName, LastName
FROM Customers;
```

Careful with HAVING

- The HAVING clause is used to filter the rows after all the rows are selected and it is used like a filter.
- It works by going through the final result table of the query parsing out the rows that don’t meet the HAVING condition.

Careful with HAVING

```
USE LECTURE5_JOIN;
-- Bad query
SELECT CustomerID, COUNT(CustomerID) AS
OrderCount
FROM CustomerOrder
GROUP BY CustomerID
HAVING CustomerID = 1 OR CustomerID = 3;

USE LECTURE5_JOIN;
-- Better query
SELECT CustomerID, COUNT(CustomerID) AS
OrderCount
FROM CustomerOrder
WHERE CustomerID = 1 OR CustomerID = 3
GROUP BY CustomerID
```

COUNT, AVG, SUM

- COUNT(1) & COUNT (*) are the same
- Ignore NULL value

Avoid using UNION

- Avoid using UNION clause whenever possible
- UNION clause causes sorting data in the table and that slows down SQL execution.
- use UNION ALL and remove duplicates

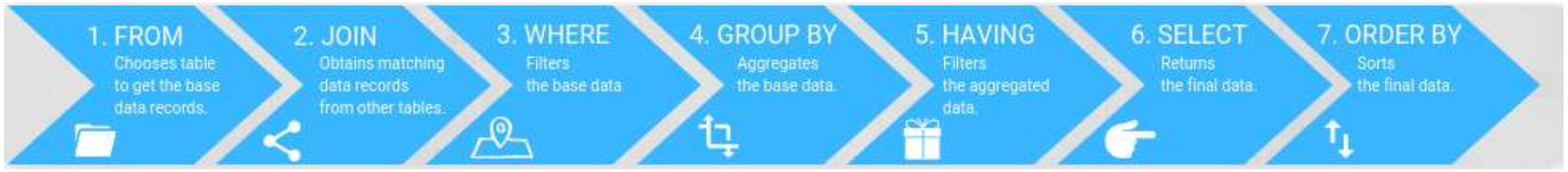
Simplicity

```sql
-- Bad query
SELECT OrderID, FoodName, DeliveryAddressID
FROM CustomerOrder
WHERE DeliveryAddressID = 1 + 1;


-- Better query
SELECT OrderID, FoodName, DeliveryAddressID
FROM CustomerOrder
WHERE DeliveryAddressID = 2;
```

Big picture

```
SELECT column_data
FROM source
    JOIN source2
WHERE condition
GROUP BY
HAVING condition
ORDER BY sort [ASC|DESC]
```


1. FROM
Chooses table to get the base data records.
2. JOIN
Obtains matching data records from other tables.
3. WHERE
Filters the base data
4. GROUP BY
Aggregates the base data.
5. HAVING
Filters the aggregated data.
6. SELECT
Returns the final data.
7. ORDER BY
Sorts the final data.

Practice on class

- OPEN FILE “LECTURE7-CLASS_PRACTICE.sql”

Some other topic

- SQL Wildcards, Trigger
- IF ELSE, SQL CASE Expression
- SOME Operators
- SQL AUTO INCREASE ON/OFF
- SQL INJECTION
- DELETE, UPDATE CASCADE
- SQL Concurrency

And more

- SQL Transaction
- Database clusters(high avaiability)
- Scaling database(scale ability)
- Distribution database
- Database cluster
- No-SQL

Extra Resources

Name	Link
became SQL god?	https://www.w3schools.com/sql/default.asp