SQL GUIDE - IDA COACHING CENTER

SQL - Structured Query Language

DATABASE

Tạo Database
```
CREATE DATABASE CUSTOMER
```
Mỗi Database chứa nhiều bảng và cấu trúc dữ liệu của từng bảng (Object Explorer)

Xóa Database
```
DROP DATABASE CUSTOMER
```

Thiết lập môi trường làm việc trực tiếp
```
USE database_name
```
VD:
```
USE Customer
```

TABLE

 # Tạo bảng mới
```
CREATE TABLE Table_name (
   column_name1 data_type,
   column_name2 data_type...)
```

 # Insert dữ liệu vào SQL server

1. Insert bằng câu lệnh Insert into

VD:
```
INSERT INTO RETAIL_2019
VALUES('2019-01-01', 'burgers', '5', '10')
```
2. Insert bằng cách dùng Import data from Excel (link)

 # Xóa bảng

-- Xóa hoàn toàn:
```
DROP TABLE Table_name
```
-- Chỉ xóa bản ghi trong bảng:
```
DELETE FROM Table_name WHERE Condition1,2,3…
```
 # Thay đổi dữ liệu bảng:

-- Đổi tên bảng
```
EXEC SP_RENAME 'retail_2019', 'retail_2020'
EXEC SP_RENAME 'retail_2020', 'retail_2019'
```
-- Đổi tên cột:
```
EXEC SP_RENAME 'retail_2019.PRODUCT_NAME', 'PRODUCT_NAME2', 'COLUMN'
EXEC SP_RENAME 'retail_2019.PRODUCT_NAME2', 'PRODUCT_NAME', 'COLUMN'
```

-- Thay đổi định dạng dữ liệu của cột trong bảng:
```
ALTER TABLE RETAIL_2019 ALTER COLUMN PRODUCT_NAME VARCHAR(40)
```

-- Xóa cột trong bảng:
```
ALTER TABLE Table_name DROP COLUMN ten_cot_can_xoa
```

-- Add thêm cột mới:
```
ALTER TABLE RETAIL_2019 ADD Payment_amount float
```

Update dữ liệu trong bảng:
```
UPDATE Table_name
SET C1 = …
WHERE (Condition)
```
VD:
```
UPDATE RETAIL_2019
SET PAYMENT_AMOUNT = QUANTITY*PRICE
```

QUERYING DATA

Trước tiên, hãy cùng tạo 1 Database, thiết lập môi trường làm việc, tạo 1 bảng biểu và cùng thực hành viết code truy vấn dữ liệu trên các bảng đã tạo nhé:

```
CREATE DATABASE RETAIL --> database retail đã được tạo
USE RETAIL  --> môi trường làm việc là DB retail đã được thiết lập

CREATE TABLE RETAIL_2019
(Sale_date DATE, Product_name VARCHAR(30), Price FLOAT, Quantity FLOAT)
--> bảng Retail_2019 đã được tạo

INSERT INTO RETAIL_2019 VALUES('2019-01-01', 'burgers', '5', '10')
INSERT INTO RETAIL_2019 VALUES('2019-01-02', 'soup', '9', '2')
INSERT INTO RETAIL_2019 VALUES('2019-02-01', 'cookies', '10', '8')
INSERT INTO RETAIL_2019 VALUES('2019-02-02', 'spaghetti', '1', '6')
INSERT INTO RETAIL_2019 VALUES('2019-02-03', 'red wine', '4', '8')
INSERT INTO RETAIL_2019 VALUES('2019-03-01', 'cheese', '1', '5')
INSERT INTO RETAIL_2019 VALUES('2019-03-05', 'burgers', '10', '5')
INSERT INTO RETAIL_2019 VALUES('2019-03-06', 'soup', '7', '5')
INSERT INTO RETAIL_2019 VALUES('2019-04-01', 'cookies', '4', '5')
INSERT INTO RETAIL_2019 VALUES('2019-04-02', 'spaghetti', '7', '7')
```

--> mỗi câu code trên đây đã đưa 1 dòng dữ liệu vào bảng Retail_2019

-- Xem kết quả:

```
SELECT *
FROM RETAIL_2019
```

Sale_date	Product_name	Price	Quantity
2019-01-01	burgers	5	10
2019-01-02	soup	9	2
2019-02-01	cookies	10	8
2019-02-02	spaghetti	1	6
2019-02-03	red wine	4	8
2019-03-01	cheese	1	5
2019-03-05	burgers	10	5
2019-03-06	soup	7	5
2019-04-01	cookies	4	5
2019-04-02	spaghetti	7	7

Querying from a tables

-- truy vấn một số cột từ bảng:

```
SELECT SALE_DATE, PRODUCT_NAME
FROM RETAIL_2019
```

--> Kết quả:

Sale_date	Pr_name
1/1/2019	burgers
1/2/2019	soup
2/1/2019	cookies
2/2/2019	spaghetti
2/3/2019	red wine
3/1/2019	cheese
3/5/2019	burgers
3/6/2019	soup
4/1/2019	cookies
4/2/2019	spaghetti

-- truy vấn toàn bộ cột từ bảng:

```
SELECT *
FROM RETAIL_2019
```

Sale_date	Product_name	Price	Quantity
2019-01-01	burgers	5	10
2019-01-02	soup	9	2
2019-02-01	cookies	10	8
2019-02-02	spaghetti	1	6
2019-02-03	red wine	4	8
2019-03-01	cheese	1	5
2019-03-05	burgers	10	5
2019-03-06	soup	7	5
2019-04-01	cookies	4	5
2019-04-02	spaghetti	7	7

-- truy vấn top N dòng ngẫu nhiên:

```
SELECT top 5 *
FROM RETAIL_2019
```

Sale_date	Product_name	Price	Quantity
1/1/2019	burgers	5	10
1/2/2019	soup	9	2
2/1/2019	cookies	10	8
2/2/2019	spaghetti	1	6
2/3/2019	red wine	4	8

-- sắp xếp kết quả truy vấn:

```
SELECT top 10 *
FROM RETAIL_2019
ORDER BY Quantity --> sắp xếp tăng dần theo giá trị trường Quantity
```

Sale_date	Product_name	Price	Quantity
1/2/2019	soup	9	2
3/1/2019	cheese	1	5
3/5/2019	burgers	10	5
3/6/2019	soup	7	5
4/1/2019	cookies	4	5

```
SELECT top 5 *
FROM RETAIL_2019
ORDER BY Quantity DESC --> sắp xếp giảm dần theo giá trị trường Quantity
```

Sale_date	Product_name	Price	Quantity
1/1/2019	burgers	5	10
2/1/2019	cookies	10	8
2/3/2019	red wine	4	8
4/2/2019	spaghetti	7	7
2/2/2019	spaghetti	1	6

-- truy vấn có điều kiện:

```
SELECT *
FROM RETAIL_2019
WHERE Quantity > 5
```

Sale_date	Product_name	Price	Quantity
1/1/2019	burgers	5	10
2/1/2019	cookies	10	8
2/2/2019	spaghetti	1	6
2/3/2019	red wine	4	8
4/2/2019	spaghetti	7	7

---- một số condition thường dùng:

-- like: tìm ký tự giống:

```
SELECT *
FROM RETAIL_2019
WHERE Product_name LIKE '%BUR%'
```

-- lấy các đơn hàng mà sản phẩm có cụm từ BUR

-- mỗi ký tự % đại diện cho 1 cụm ký tự nào đó hoặc không có cụm ký tự nào

Sale_date	Product_name	Price	Quantity
1/1/2019	burgers	5	10
3/5/2019	burgers	10	5

-- between a and b: tương tự WHERE C1 >= a AND C1 <= b

```
SELECT *
FROM RETAIL_2019
WHERE Quantity BETWEEN 5 AND 10
```

--> lấy các đơn hàng có Quantity giao dịch từ 1 đến 5

Sale_date	Product_name	Price	Quantity
1/2/2019	soup	9	2
3/1/2019	cheese	1	5
3/5/2019	burgers	10	5
3/6/2019	soup	7	5
4/1/2019	cookies	4	5

-- 1 trường nằm trong 1 danh sách nào đó:
-- lấy các đơn hàng của burgers và cheese:

```
SELECT *
FROM RETAIL_2019
WHERE PRODUCT_NAME IN ('burgers', 'cheese')
```

Sale_date	Product_name	Price	Quantity
1/1/2019	burgers	5	10
3/1/2019	cheese	1	5
3/5/2019	burgers	10	5

-- lấy các giá trị không trùng:
-- có bao nhiêu sản phẩm không trùng trong bảng retail:

```
SELECT DISTINCT Product_name
FROM RETAIL_2019
```

Pr_name
burgers
cheese
cookies
red wine
soup
spaghetti

-- tính toán theo nhóm với group by:
-- group by giúp nhóm các bản ghi lại và tính toán một giá trị nào đó cho các bản ghi này:
-- vd tính số đơn hàng, giá trung bình, tổng số lượng bán theo các sản phẩm:

```
SELECT Product_name, COUNT(1) so_don_hang, AVG(PRICE) gia_TB, SUM(Quantity) tong_sl
FROM RETAIL_2019
GROUP BY Product_name
```

Product_name	so_don_hang	gia_TB	tong_sl
burgers	2	7.5	15
cheese	1	1	5
cookies	2	7	13
red wine	1	4	8
soup	2	8	7
spaghetti	2	4	13

-- điều kiện khi tính toán với group by:
-- vd tính số đơn hàng, giá TB, tổng SL bán theo các sản phẩm của các sản phẩm có giá TB trên 5:

```
SELECT Product_name, COUNT(1) so_don_hang, AVG(PRICE) gia_TB, SUM(Quantity) tong_sl
FROM RETAIL_2019
GROUP BY Product_name
HAVING AVG(PRICE) > 5
```

Product_name	so_don_hang	gia_TB	tong_sl
burgers	2	7.5	15
cookies	2	7	13
soup	2	8	7

Querying from multiple tables

-- tạo thêm bảng sale_infor:

```
CREATE TABLE SALE_INFOR (SALE_NAME VARCHAR(30), PRODUCT_NAME VARCHAR(20))
INSERT INTO SALE_INFOR VALUES ('Lena', 'burgers')
INSERT INTO SALE_INFOR VALUES ('Jonh', 'soup')
INSERT INTO SALE_INFOR VALUES ('Jenifer', 'red wine')
```

--- JOIN:
-- ? lấy toàn bộ thông tin bán hàng kèm thông tin sale:

```
SELECT A.*, B.SALE_NAME
FROM RETAIL_2019 A LEFT JOIN SALE_INFOR B
ON A.Product_name = B.PRODUCT_NAME
```

Sale_date	Product_name	Price	Quantity	SALE_NAME
1/1/2019	burgers	5	10	Lena
1/2/2019	soup	9	2	Jonh
2/1/2019	cookies	10	8	NULL
2/2/2019	spaghetti	1	6	NULL
2/3/2019	red wine	4	8	Jenifer
3/1/2019	cheese	1	5	NULL
3/5/2019	burgers	10	5	Lena
3/6/2019	soup	7	5	Jonh
4/1/2019	cookies	4	5	NULL
4/2/2019	spaghetti	7	7	NULL

-- ? lấy thông tin bán hàng kèm thông tin sale của chỉ những đơn được bán bởi các bạn sale có trong bảng SALE_INFOR:

```
SELECT A.*, B.SALE_NAME
FROM RETAIL_2019 A INNER JOIN SALE_INFOR B
ON A.Product_name = B.PRODUCT_NAME
```

Sale_date	Product_name	Price	Quantity	SALE_NAME
1/1/2019	burgers	5	10	Lena
1/2/2019	soup	9	2	Jonh
2/3/2019	red wine	4	8	Jenifer
3/5/2019	burgers	10	5	Lena
3/6/2019	soup	7	5	Jonh

-- cú pháp câu lệnh Right join ngược lại câu lệnh left join
-- thay vì dùng inner join cũng có thể dùng truy vấn:

```
SELECT A.*, B.SALE_NAME
FROM RETAIL_2019 A, SALE_INFOR B
WHERE A.Product_name = B.PRODUCT_NAME
```

Sale_date	Product_name	Price	Quantity	SALE_NAME
1/1/2019	burgers	5	10	Lena
1/2/2019	soup	9	2	Jonh
2/3/2019	red wine	4	8	Jenifer
3/5/2019	burgers	10	5	Lena
3/6/2019	soup	7	5	Jonh

-- lấy toàn bộ 2 bảng, bao gồm cả các phần dữ liệu không giao nhau:

```
SELECT A.*, B.SALE_NAME
FROM RETAIL_2019 A FULL JOIN SALE_INFOR B
ON A.Product_name = B.PRODUCT_NAME
```

Sale_date	Product_name	Price	Quantity	SALE_NAME
1/1/2019	burgers	5	10	Lena
1/2/2019	soup	9	2	Jonh
2/1/2019	cookies	10	8	NULL
2/2/2019	spaghetti	1	6	NULL
2/3/2019	red wine	4	8	Jenifer
3/1/2019	cheese	1	5	NULL
3/5/2019	burgers	10	5	Lena
3/6/2019	soup	7	5	Jonh
4/1/2019	cookies	4	5	NULL
4/2/2019	spaghetti	7	7	NULL

-- nối dữ liệu từ 2 bảng:

```
SELECT *
FROM RETAIL_2019
WHERE Quantity > 5
UNION --ALL
SELECT *
FROM RETAIL_2019
WHERE Price < 10
```

--> khác biệt giữa Union và Union all
--> khi dùng Union/Union all, các trường lấy ra ở các mệnh đề phải giống nhau về thứ tự

Sale_date	Product_name	Price	Quantity
1/1/2019	burgers	5	10
1/2/2019	soup	9	2
2/1/2019	cookies	10	8
2/2/2019	spaghetti	1	6
2/3/2019	red wine	4	8
3/1/2019	cheese	1	5
3/6/2019	soup	7	5
4/1/2019	cookies	4	5
4/2/2019	spaghetti	7	7

```
-- tạo thêm cột Payment_amount:
ALTER TABLE RETAIL_2019 ADD Payment_amount float

-- Tính Payment_amount = Quantity x Price
UPDATE RETAIL_2019
SET PAYMENT_AMOUNT = QUANTITY*PRICE

-- Xem kết quả
SELECT * FROM RETAIL_2019
```

Sale_date	Product_name	Price	Quantity	Payment
1/1/2019	burgers	5	10	50
1/2/2019	soup	9	2	18
2/1/2019	cookies	10	8	80
2/2/2019	spaghetti	1	6	6
2/3/2019	red wine	4	8	32
3/1/2019	cheese	1	5	5
3/5/2019	burgers	10	5	50
3/6/2019	soup	7	5	35
4/1/2019	cookies	4	5	20
4/2/2019	spaghetti	7	7	49