

목록

데이터분석 기초 SQL 부트캠프 1일차_v08	1
데이터분석 기초 SQL 부트캠프 2일차_v05	80
데이터분석 기초 SQL 부트캠프 3일차_v06	137
데이터분석 기초 SQL 부트캠프 4일차_v03	206

데이터분석 기초 SQL 부트캠프

목차

1. 데이터베이스와 RDBMS 그리고 SQL

2. MYSQL & HEIDISQL 설치

3. DataType

문자형, 숫자형, 날짜형

4. 데이터베이스

5. Table

create, alter, drop, truncate, insert, update

csv file import to table

06. 데이터 불러오기

select, from, alias, limit, offset

1.

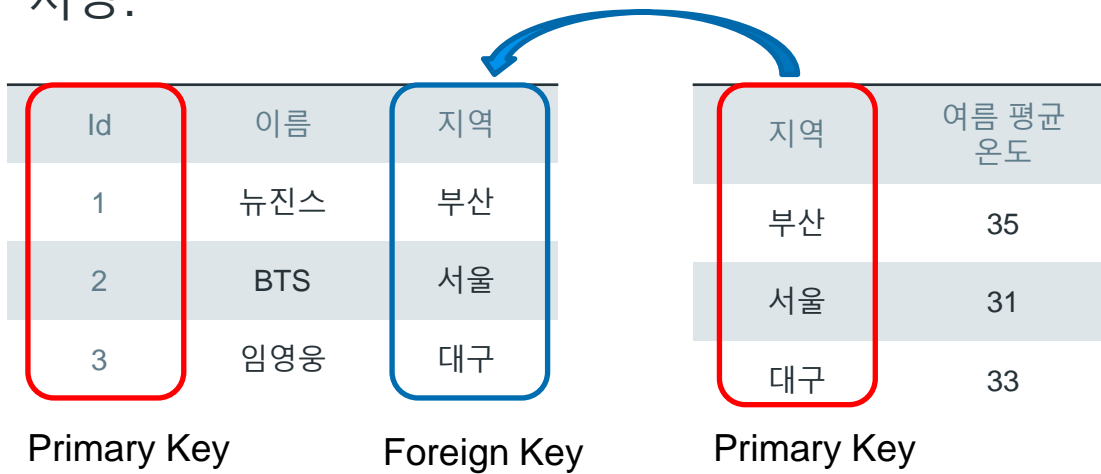
데이터베이스와 RDBMS 그리고 SQL

DATABASE(DB)

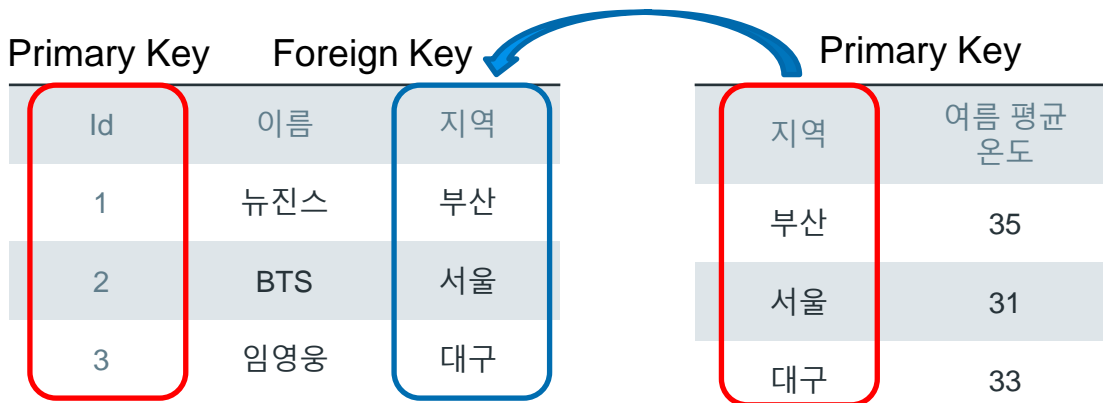
- Database(DB) : 데이터베이스란 컴퓨터에 저장되는 데이터들의 집합, 저장소.
- DBMS(DataBase Management System) : DB를 관리하는 시스템
ex) oracle, mysql, mariadb, ms server

RDBMS(Relational database management system)

- RDB(Relational DataBase) : 관계형 데이터베이스는 데이터들 사이의 관계를 가지는 데이터들을 다루며, 테이블 형태로 자료들을 저장.



RDBMS(Relational database management system)



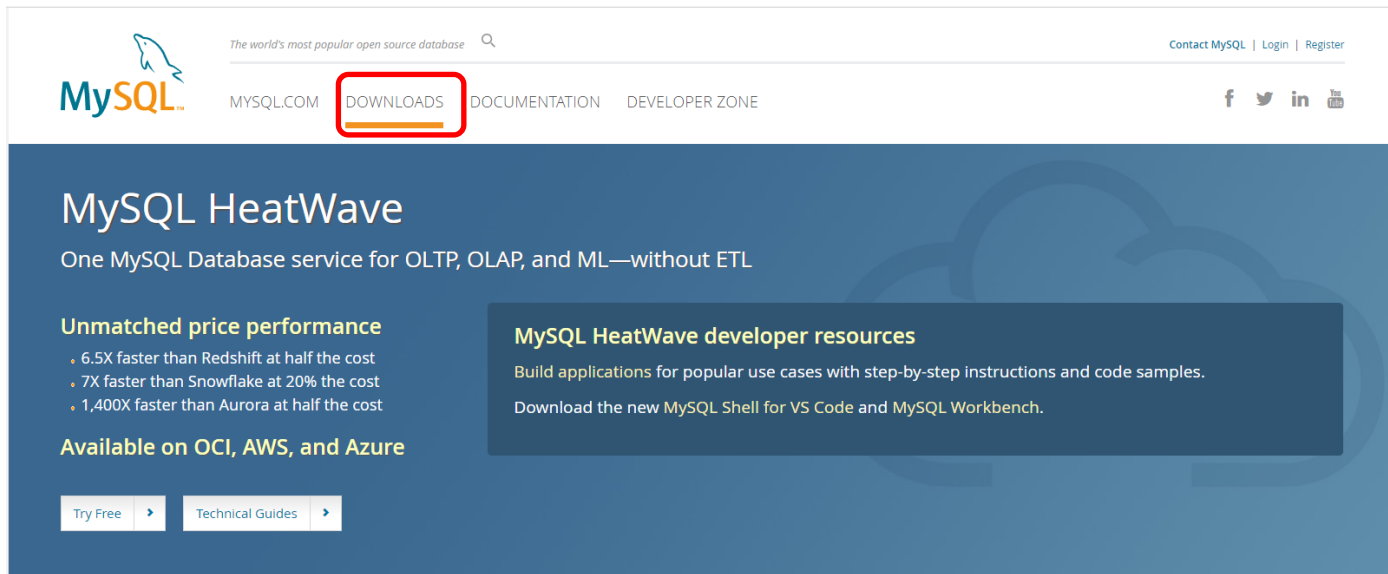
- Primary Key : Table에서 row를 식별하게 해주며, 유일한 값이고 공백일 수 없으며 중복될 수 없음
- Foreign Key : 다른 Table과 연결시켜주는 column. 다른 Table의 primary key를 참조. 공백, 중복 가능.

2.

MYSQL 설치

HEIDISQL 설치

MySQL 설치



The screenshot shows the MySQL website. The MySQL logo is on the left, followed by the tagline "The world's most popular open source database" and a search icon. To the right of the logo is the text "MYSQL.COM". The "DOWNLOADS" link is highlighted with a red rectangular box. To its right are the links "DOCUMENTATION" and "DEVELOPER ZONE". Further right are social media icons for Facebook, Twitter, LinkedIn, and YouTube. In the top right corner, there are links for "Contact MySQL", "Login", and "Register".

MySQL HeatWave
One MySQL Database service for OLTP, OLAP, and ML—without ETL

Unmatched price performance

- 6.5X faster than Redshift at half the cost
- 7X faster than Snowflake at 20% the cost
- 1,400X faster than Aurora at half the cost

Available on OCI, AWS, and Azure

MySQL HeatWave developer resources
Build applications for popular use cases with step-by-step instructions and code samples.
Download the new MySQL Shell for VS Code and MySQL Workbench.

[Try Free](#) [Technical Guides](#)

MySQL 설치

Unmatched price performance

- 6.5X faster than Redshift at half the cost
- 7X faster than Snowflake at 20% the cost
- 1,400X faster than Aurora at half the cost

Available on OCI, AWS, and Azure

Try Free

Technical Guides

MySQL HeatWave developer resources

Build applications for popular use cases with step-by-step instructions and code samples.
Download the new MySQL Shell for VS Code and MySQL Workbench.

MySQL Newsletter

[Subscribe »](#)

[Archive »](#)

Free Webinars

Introducing MySQL HeatWave
Lakehouse - processing data in
object store with record
performance
Thursday, July 27, 2023

Deep Dive into MySQL HeatWave
Lakehouse
Tuesday, August 08, 2023

Migrating from On-Premises
MySQL to MySQL HeatWave
On-Demand

[More »](#)

MySQL Enterprise Edition

MySQL Enterprise Edition includes the most comprehensive set of advanced features, management tools and technical support for MySQL.

[Learn More »](#)

[Customer Download »](#)

[Trial Download »](#)

MySQL Cluster CGE

MySQL Cluster is a real-time open source transactional database designed for fast, always-on access to data under high throughput conditions.

- MySQL Cluster
- MySQL Cluster Manager
- Plus, everything in MySQL Enterprise Edition

[Learn More »](#)

[Customer Download »](#) (Select Patches & Updates Tab, Product Search)

[Trial Download »](#)

[MySQL Community \(GPL\) Downloads »](#)

MySQL 설치

MySQL Community Downloads

- MySQL Yum Repository
- MySQL APT Repository
- MySQL SUSE Repository
- MySQL Community Server
- MySQL Cluster
- MySQL Router
- MySQL Shell
- MySQL Operator
- MySQL NDB Operator
- MySQL Workbench
- MySQL Installer for Windows
- C API (libmysqlclient)
- Connector/C++
- Connector/J
- Connector/NET
- Connector/Node.js
- Connector/ODBC
- Connector/Python
- MySQL Native Driver for PHP
- MySQL Benchmark Tool
- Time zone description tables
- Download Archives

MySQL 설치

MySQL Community Downloads

MySQL Community Server

The screenshot shows the MySQL Community Downloads page. At the top, there are tabs for "General Availability (GA) Releases", "Archives", and a help icon. The main heading is "MySQL Community Server 8.0.34". Below this, there are two dropdown menus: "Select Version:" with "8.0.34" selected, and "Select Operating System:" with "Microsoft Windows" selected. Below these is a section titled "Recommended Download:" which features a large graphic for "MySQL Installer for Windows". The graphic includes the text "All MySQL Products. For All Windows Platforms. In One Package." and a Windows logo. At the bottom of the graphic, it says "Starting with MySQL 5.6 the MySQL Installer package replaces the standalone MSI packages." Below the graphic, there is a link that says "Windows (x86, 32 & 64-bit), MySQL Installer MSI" and a button that says "Go to Download Page >".

General Availability (GA) Releases Archives

MySQL Community Server 8.0.34

Select Version:
8.0.34

Select Operating System:
Microsoft Windows

Recommended Download:

MySQL Installer
for Windows

All MySQL Products. For All Windows Platforms.
In One Package.

Starting with MySQL 5.6 the MySQL Installer package replaces the standalone MSI packages.

Windows (x86, 32 & 64-bit), MySQL Installer MSI

[Go to Download Page >](#)

MySQL 설치

General Availability (GA) ReleasesArchives

MySQL Installer 8.0.34

! Note: MySQL 8.0 is the final series with MySQL Installer. As of MySQL 8.1, use a MySQL product's MSI or Zip archive for installation. MySQL Server 8.1 and higher also bundle MySQL Configurator, a tool that helps configure MySQL Server.

Select Version:

8.0.34

Select Operating System:

Microsoft Windows

Windows (x86, 32-bit), MSI Installer (mysql-installer-web-community-8.0.34.0.msi)	8.0.34	2.4M	Download
Windows (x86, 32-bit), MSI Installer (mysql-installer-community-8.0.34.0.msi)	8.0.34	331.3M	Download

! We suggest that you use the MD5 checksums and GnuPG signatures to verify the integrity of the packages you download.

12

MySQL 설치

Login Now or Sign Up for a free account.

An Oracle Web Account provides you with the following advantages:

- Fast access to MySQL software downloads
- Download technical White Papers and Presentations
- Post messages in the MySQL Discussion Forums
- Report and track bugs in the MySQL bug system

Login »

using my Oracle Web account

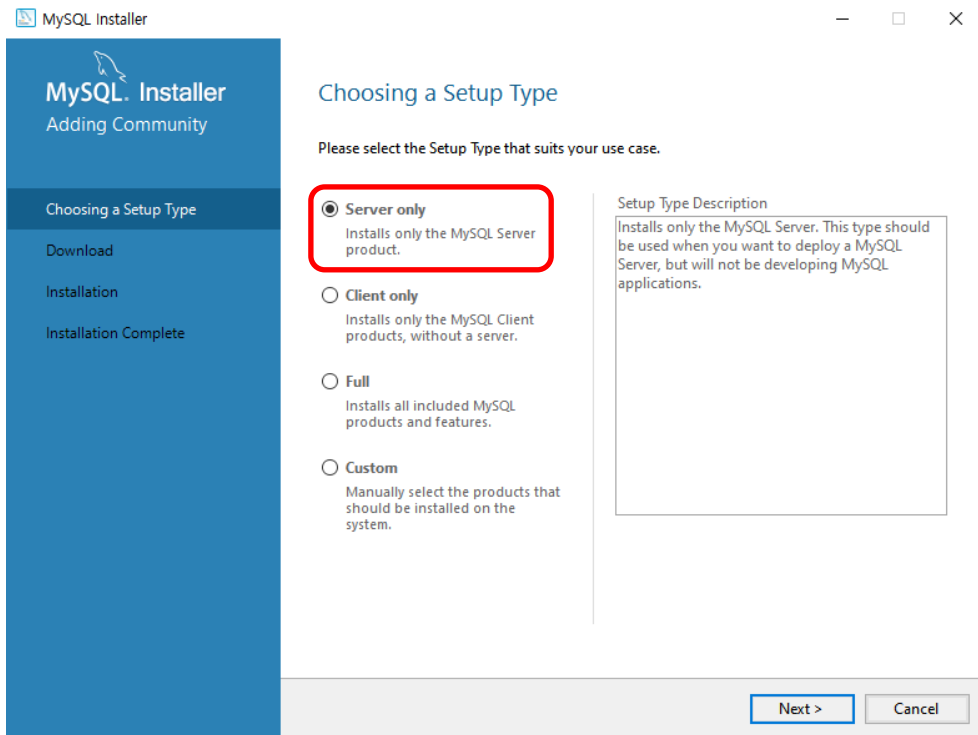
Sign Up »

for an Oracle Web account

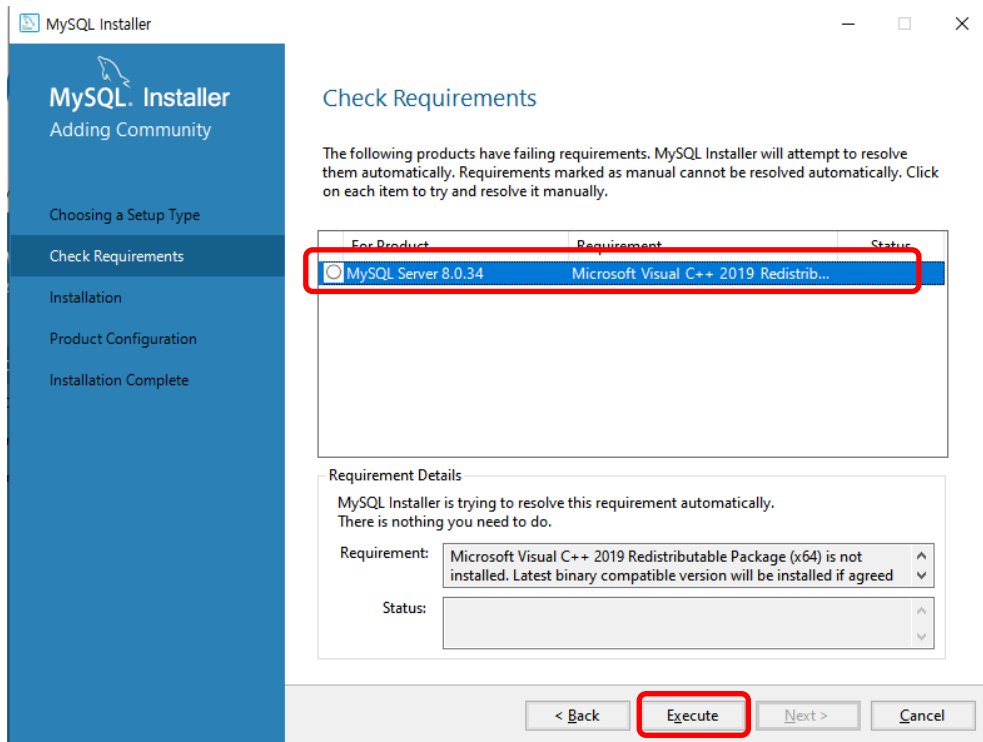
MySQL.com is using Oracle SSO for authentication. If you already have an Oracle Web account, click the Login link. Otherwise, you can signup for a free account by clicking the Sign Up link and following the instructions.

No thanks, just start my download.

MySQL 설치

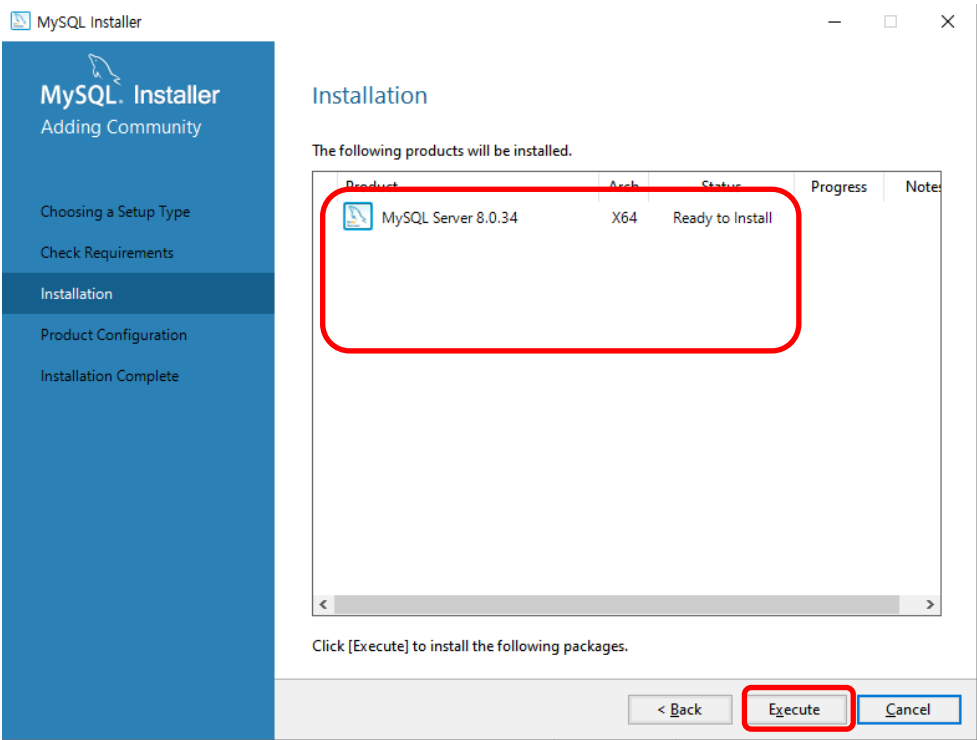


MySQL 설치



- 요구되는 프로그램들 모두 Execute. 동의 및 설치.

MySQL 설치



MySQL 설치

MySQL Installer

MySQL Server 8.0.34

Type and Networking

Authentication Method

Accounts and Roles

Windows Service

Server File Permissions

Apply Configuration

Type and Networking

Server Configuration Type

Choose the correct server configuration type for this MySQL Server installation. This setting will define how much system resources are assigned to the MySQL Server instance.

Config Type: Development Computer

Connectivity

Use the following controls to select how you would like to connect to this server.

☒ TCP/IP Port: 3306 X Protocol Port: 33060

☒ Open Windows Firewall ports for network access

☐ Named Pipe Pipe Name: MYSQL

☐ Shared Memory Memory Name: MYSQL

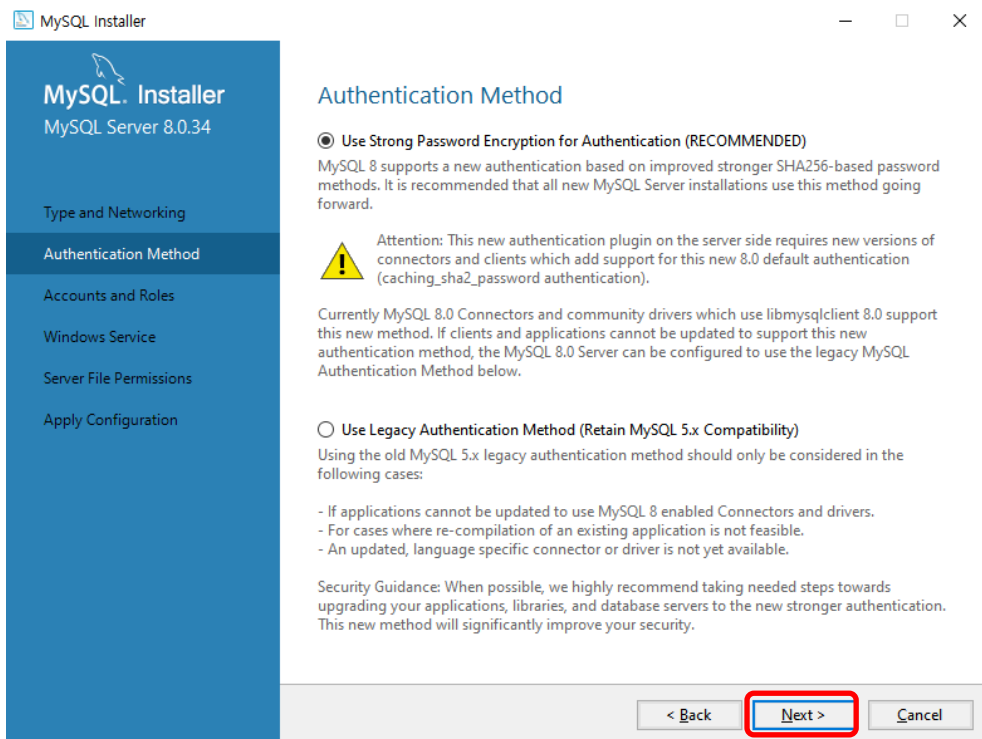
Advanced Configuration

Select the check box below to get additional configuration pages where you can set advanced and logging options for this server instance.

☐ Show Advanced and Logging Options

Next > Cancel

MySQL 설치



MySQL 설치

The screenshot shows the 'MySQL Installer' window for 'MySQL Server 8.0.34'. The left sidebar lists the installation steps: 'Type and Networking', 'Authentication Method', 'Accounts and Roles' (which is the current step), 'Windows Service', 'Server File Permissions', and 'Apply Configuration'. The main area is titled 'Accounts and Roles' and contains two sections. The first section, 'Root Account Password', prompts the user to enter a password for the root account. It includes two input fields: 'MySQL Root Password:' and 'Repeat Password:'. The password strength is indicated as 'Weak'. The second section, 'MySQL User Accounts', provides instructions to create user accounts and assign roles. It features a table with columns for 'MySQL User Name', 'Host', and 'User Role'. To the right of the table are buttons for 'Add User', 'Edit User', and 'Delete'. At the bottom of the window, there are three buttons: '< Back', 'Next >' (which is highlighted with a red rectangle), and 'Cancel'.

MySQL Installer

MySQL Server 8.0.34

Type and Networking

Authentication Method

Accounts and Roles

Windows Service

Server File Permissions

Apply Configuration

Accounts and Roles

Root Account Password
Enter the password for the root account. Please remember to store this password in a secure place.

MySQL Root Password:

Repeat Password:

Password strength: **Weak**

MySQL User Accounts
Create MySQL user accounts for your users and applications. Assign a role to the user that consists of a set of privileges.

MySQL User Name	Host	User Role
-----------------	------	-----------

Add User

Edit User

Delete

< Back **Next >** Cancel

- Root Password는 가장 중요한 비밀번호이기 때문에 까먹지 않을 수 있는 비밀번호를 사용.
- 이후 next -> excute->설치완료

HeidiSQL 설치

 **HeidiSQL**

Home **Downloads** Images Forum Donate Bugtracker Help

Ads were blocked - no problem. But keep in mind that developing HeidiSQL, user support and hosting takes time and money. You may want to send a donation instead.

Ads were blocked - no problem. But keep in mind that developing HeidiSQL, user support and hosting takes time and money. You may want to send a donation instead.

Download HeidiSQL 12.5, released on 08 May 2023

 **Installer, 32/64 bit combined** 

 Portable version (zipped): 32 bit \approx 64 bit \approx

 Sourcecode

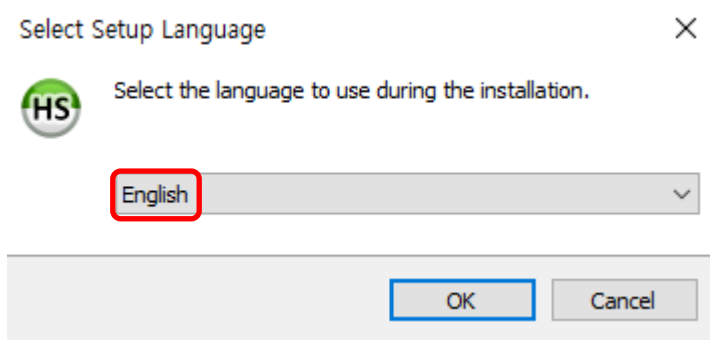
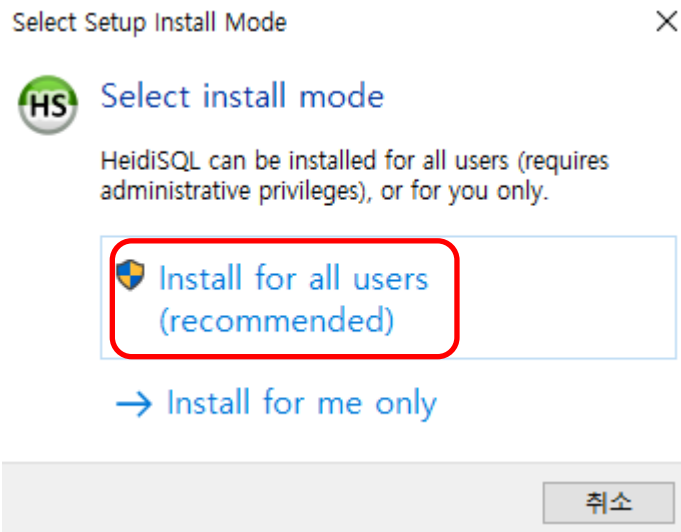
 Previous releases

Donate
Support making HeidiSQL better and better

Compatibility notes

- HeidiSQL runs fine on Windows 10 and 11 (and on Windows 7 + 8 with some minor issues)
- Running HeidiSQL on Wine can have various issues, depending on the version of Wine and HeidiSQL.

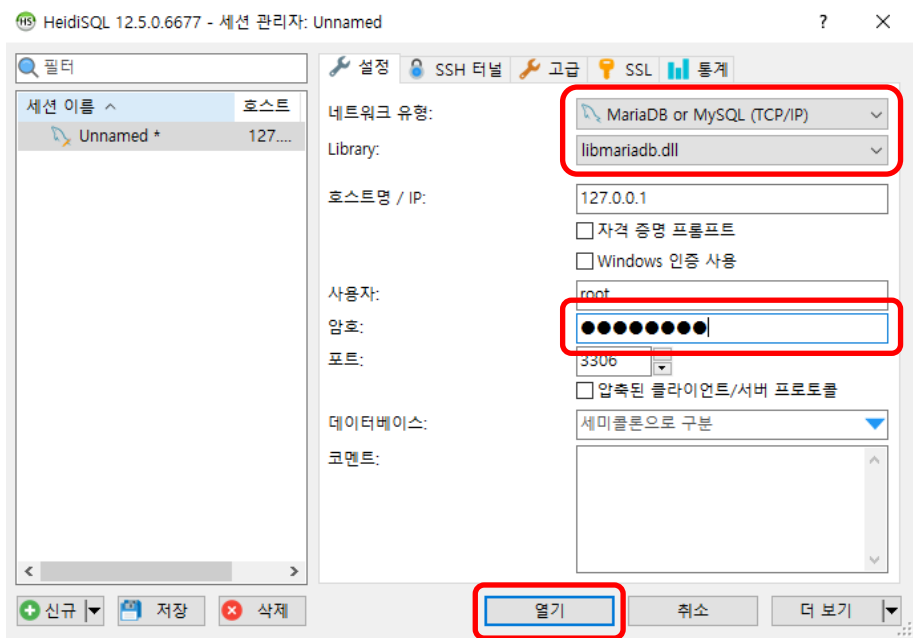
HeidiSQL 설치



HeidiSQL 설치

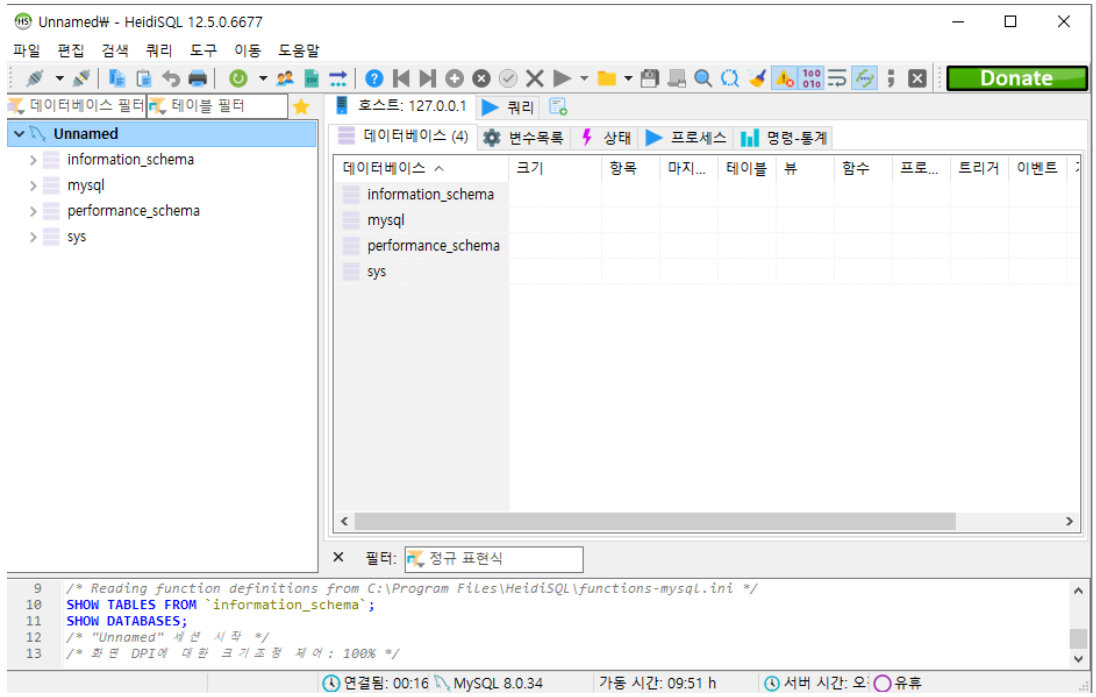


HeidiSQL 설치



- MySQL 설치시
설정했던 root의 암호

HeidiSQL 설치



3.

DataType

문자형, 숫자형, 날짜형

문자형

데이터 타입	Byte	설명
CHAR(n)	1 ~ 255	고정길이 문자형 n : 1 ~ 255
VARCHAR(n)	1 ~ 65535	가변길이 문자형 n : 1 ~ 65535
TEXT	1 ~ 65535	N 크기의 TEXT 데이터 값

- 강의에서는 encoding type utf8을 사용할 예정.
- Encoding : 문자를 컴퓨터가 이해할 수 있는 신호로 만드는 것
- Utf8에서 영어, 공백은 1byte, 한글은 한글자당 3byte.

숫자형(정수형)

데이터 타입	설명
TINYINT	(1byte) / -128 ~ +127 0 ~ 255수 표현 가능
SMALLINT	(2byte) / -32,768 ~ 32,767 0 ~ 65,536수 표현 가능
MEDIUMINT	(3byte) / -8,388,608 ~ +8,388,607 0 ~ 16,777,215수 표현 가능
INT	(4byte) / -2,147,483,648 ~ +2,147,483,647 0 ~ 4,294,967,295수 표현 가능
BIGINT	(8byte) / 무제한 수 표현 가능

- TINYINT SIGNED(생략가능)
-128 ~ 127
- TINYINT UNSIGNED
0 ~ 255

숫자형(실수형)

데이터 타입	설명
DECIMAL(n,d)	n : 전체 자릿수(Precision : 정밀도) d : 소수점 뒷자리수(Scale : 배율) DECIMAL(4) : -9,999 ~ 9,999 DECIMAL(4,1) : -999.9 ~ 999.9 DECIMAL(4,2) : -99.99 ~ 99.99

- DECIMAL은 최대 65자리까지 지원

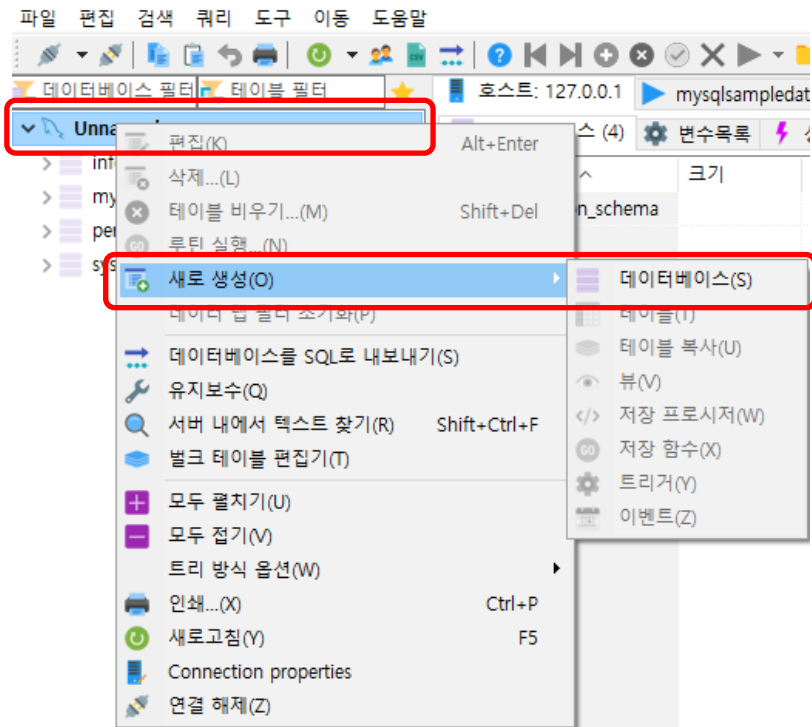
날짜형

데이터 타입	설명
DATE	'YYYY-MM-DD'
DATETIME	'YYYY-MM-DD HH:MM:SS'
TIME	'HH:MM:SS'
YEAR	YYYY

4.

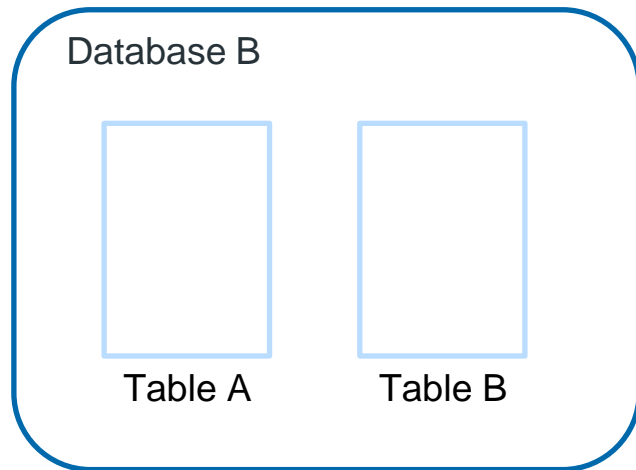
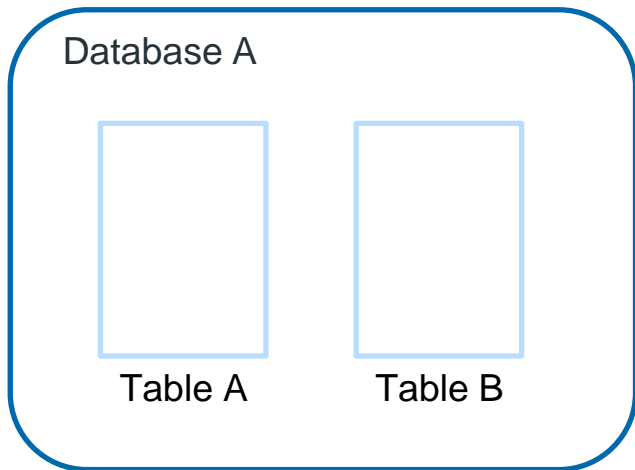
Database

Database 생성



- 계정명 우클릭 ->
새로생성 ->
데이터베이스

Database



- Database는 데이터를 저장하는 저장소.
- 여러 Database를 만드는 이유는 접근 권한을 쉽게 나누기 위함.

Database 생성

데이터베이스 생성...

이름(N)

조합(O)

서버 기본값: utf8mb4_0900_ai_ci

CREATE 코드:

```
CREATE DATABASE `test` /*!40100 COLLATE 'u
```



Database List:

- information_schema
- mysql
- performance_schema
- sys
- test** 0 B

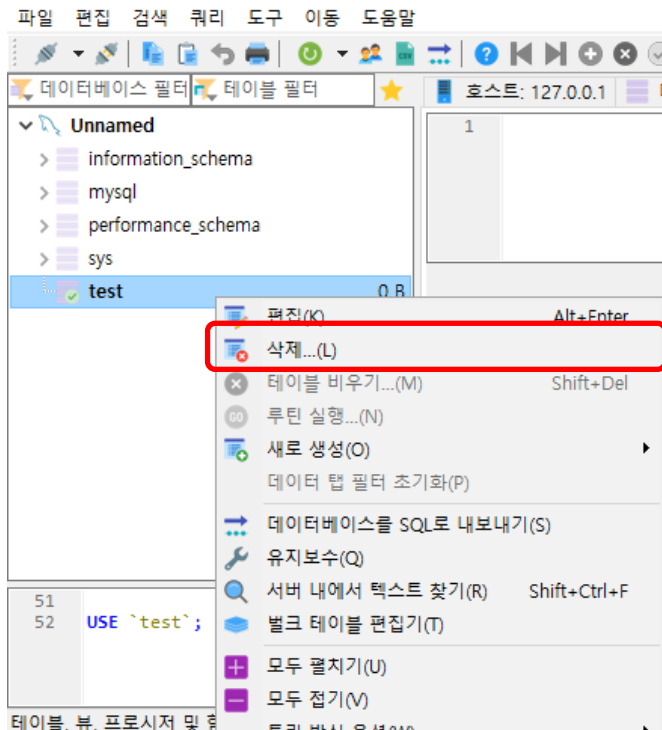
SQL Editor:

```
51
52 USE `test`;
```

- 조합
Utf8bm4_general_ci 로 변경

- Test라는 DB가 생성됨.
- 마우스로 클릭시 USE `test`;
쿼리가 자동 실행

Database 삭제



- Test라는 DB 마우스 우클릭 후 삭제.
- DROP DATABASE 'TEST';와 같은 효과

5.

Table

CREATE, ALTER, DROP, TRUNCATE

CREATE TABLE

- CREATE TABLE 테이블이름 (
 컬럼명 데이터타입,
 컬럼명 데이터타입,
 ...
 컬럼명 데이터타입
);

CREATE TABLE

- SQL Code

```
CREATE TABLE customers(
customer_number INT,
customer_name VARCHAR(50),
phone VARCHAR(50)
);
```

- 실행결과

#	이름	데이터 유형	길이/설정	부호 없음	NULL 허용	0으로 채움	기본값
1	customer_nu...	INT	10	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
2	customer_name	VARCHAR	50	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
3	phone	VARCHAR	50	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL

CREATE TABLE

- SQL Code

```
CREATE TABLE customers2(
customer_number INT NOT NULL,
customer_name VARCHAR(50) NOT NULL,
phone VARCHAR(50) NOT NULL
);
```

- 실행결과

#	이름	데이터 유형	길이/설정	부호 없음	NULL 허용	0으로 채움	기본값
1	customer_nu...	INT	10	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	기본값 없음
2	customer_name	VARCHAR	50	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	기본값 없음
3	phone	VARCHAR	50	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	기본값 없음

INSERT

- INSERT INTO 테이블이름
 (컬럼명1, 컬럼명2, 컬럼명3)
VALUES
 (DATA1, DATA2, DATA3)
;

INSERT

- SQL Code

```
insert into customers
  (customer_number, customer_name, phone)
values
  (1, '이상훈', '010-1234-5678'),
  (2, '김상훈', '010-1234-5679'),
  (3, '박상훈', '010-1234-5679')
;
```

- 실행결과

test.customers: 3 행 (총) (대략적)

customer_number	customer_name	phone
1	이상훈	010-1234-5678
2	김상훈	010-1234-5679
3	박상훈	010-1234-5679

DELETE / TRUNCATE

- 둘다 테이블을 지우는 쿼리.
- DELETE : 데이터 삭제
- TRUNCATE : 테이블 초기화
- DELETE FROM `테이블이름` **where ~**
- DELETE 구문에서 where 문을 생략하면 모든 데이터 삭제

UPDATE

- 이미 존재하는 TABLE 의 내용을 수정하는 구문
- UPDATE 테이블이름

SET 컬럼명1 = DATA1, 컬럼명2 = DATA2

WHERE 조건~~;

ALTER

- UPDATE가 TABLE의 내용을 수정한다면, ALTER는 TABLE의 내용 외적인 부분을 변경.
- TABLE의 이름변경, 컬럼추가, 컬럼이름 변경, 데이터타입변경, 컬럼삭제

ALTER TABLE 이름 변경

- SQL Code
- ALTER TABLE `테이블명` RENAME `새이름`

```
ALTER TABLE customers rename newcustomers;
```

- 실행결과

test	48.0 KiB
customers	16.0 KiB
customers2	16.0 KiB
customers3	16.0 KiB

test	48.0 KiB
customers2	16.0 KiB
customers3	16.0 KiB
newcustomers	16.0 KiB

ALTER TABLE 컬럼추가

- SQL Code
- ALTER TABLE `테이블명` ADD `새이름` 데이터타입

```
ALTER TABLE newcustomers ADD 지역 VARCHAR(50);
```

- 실행결과

#	이름	데이터 유형	길이/설정	부호 없음	NULL 허용	0으로 채움	기본값
1	customer_nu...	INT	10	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
2	customer_name	VARCHAR	50	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
3	phone	VARCHAR	50	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
4	지역	VARCHAR	50	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL

ALTER TABLE 컬럼 데이터 타입 변경

- SQL Code
- ALTER TABLE `테이블명` MODIFY `컬럼명` 데이터타입

```
ALTER TABLE newcustomers MODIFY 지역 INT;
```

- 실행결과

#	이름	데이터 유형	길이/설정	부호 없음	NULL 허용	0으로 채움	기본값
1	customer_nu...	INT	10	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
2	customer_name	VARCHAR	50	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
3	phone	VARCHAR	50	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
4	지역	INT	10	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL

ALTER TABLE 컬럼 이름 변경

- SQL Code
- ALTER TABLE `테이블명` CHANGE `기존컬럼명` `새컬럼명` 데이터타입

```
ALTER TABLE newcustomers CHANGE 지역 REGION VARCHAR(10);
```

- 실행결과

#	이름	데이터 유형	길이/설정	부호 없음	NULL 허용	0으로 채움	기본값
1	customer_nu...	INT	10	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
2	customer_name	VARCHAR	50	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
3	phone	VARCHAR	50	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
4	REGION	VARCHAR	10	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL

ALTER TABLE 컬럼 삭제

- SQL Code
- ALTER TABLE `테이블명` DROP `컬럼명`

```
ALTER TABLE newcustomers DROP REGION ;
```

- 실행결과

#	이름	데이터 유형	길이/설정	부호 없음	NULL 허용	0으로 채움	기본값
1	customer_nu...	INT	10	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
2	customer_name	VARCHAR	50	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
3	phone	VARCHAR	50	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL

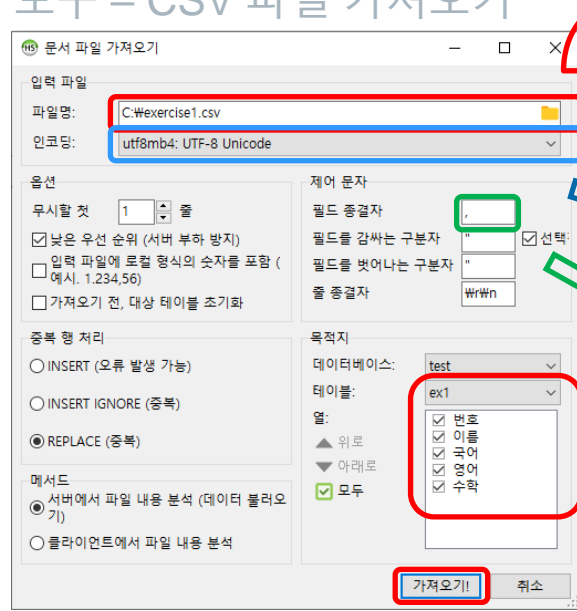
6.

데이터를 테이블에 넣는 방법

Import csv file to table, sql query

IMPORT CSV FILE TO TABLE

- 확장자 .CSV 의 데이터를 TABLE에 넣는 기능을 HEIDISQL에서 편리하게 제공.
- 도구 – CSV 파일 가져오기



입력 파일

파일명: C:\exercise1.csv

인코딩: utf8mb4: UTF-8 Unicode

옵션

무시할 첫 1 줄

☒ 낮은 우선 순위 (서버 부하 방지)

☐ 입력 파일에 로컬 형식의 숫자를 포함 (예시. 1.234,56)

☐ 가져오기 전, 대상 테이블 초기화

중복 행 처리

☐ INSERT (오류 발생 가능)

☐ INSERT IGNORE (중복)

☒ REPLACE (중복)

메서드

☒ 서버에서 파일 내용 분석 (데이터 볼러오기)

☐ 클라이언트에서 파일 내용 분석

제어 문자

필드 종결자: ;

필드를 감싸는 구분자: "

필드를 벗어나는 구분자: "

줄 종결자: \r\n

목적지

데이터베이스: test

테이블: ex1

열:

- ☒ 번호
- ☒ 이름
- ☒ 국어
- ☒ 영어
- ☒ 수학

☒ 모두

가져오기!! 취소

- 파일 경로에 한글이 없도록 설정

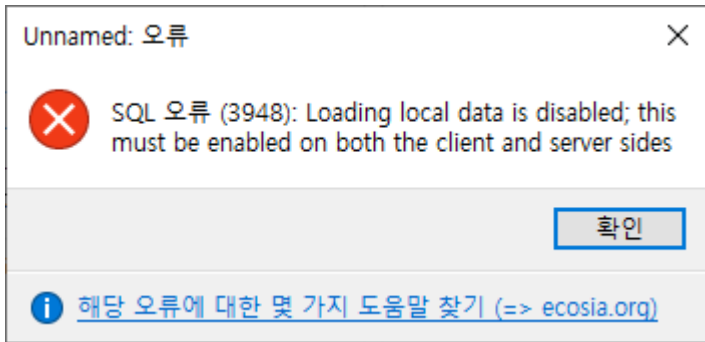
- 처음 데이터베이스 설정시 인코딩 Utf8bm4_general_ci 과 일치

- CSV 파일은 , 로 구분이 된다

- Csv 파일 업로드 전, 컬럼과 타입을 파악하여 테이블을 생성해준다.

IMPORT CSV FILE TO TABLE

- 가져오기 클릭시 발생할 수 있는 오류



- LOCAL 에서 데이터를 가져오는 것이 허용되어있지 않기 때문에 허용해줘야 함.

IMPORT CSV FILE TO TABLE

- SQL CODE

```
SHOW GLOBAL VARIABLES LIKE 'LOCAL_INFILE';
```

- 결과

Variable_name	Value
local_infile	OFF

- SQL CODE

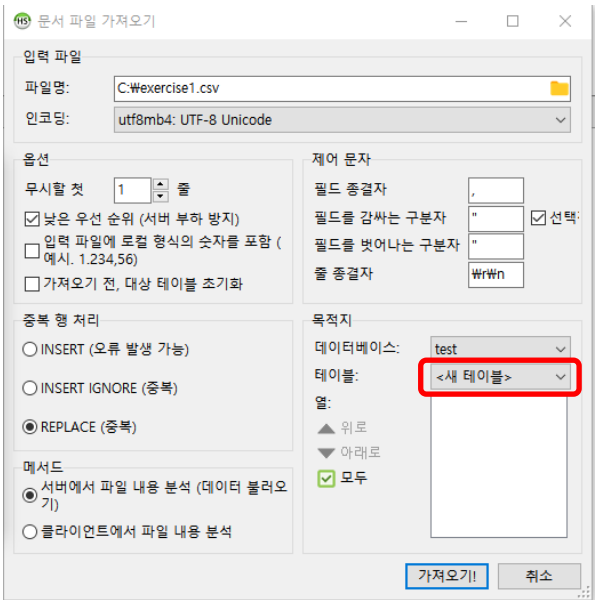
```
SET GLOBAL LOCAL_INFILE=TRUE  
SHOW GLOBAL VARIABLES LIKE 'LOCAL_INFILE';
```

- 결과

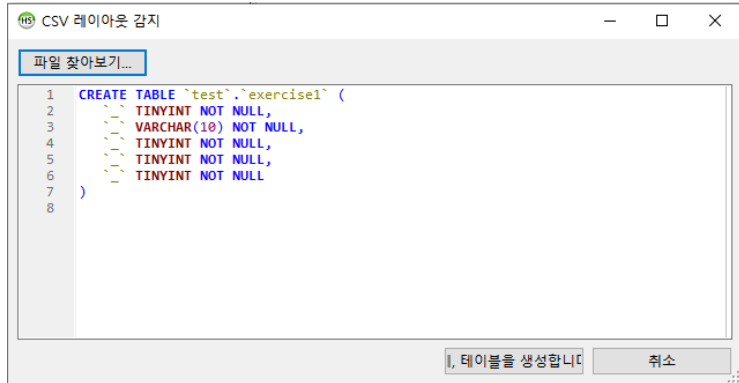
Variable_name	Value
local_infile	ON

IMPORT CSV FILE TO TABLE

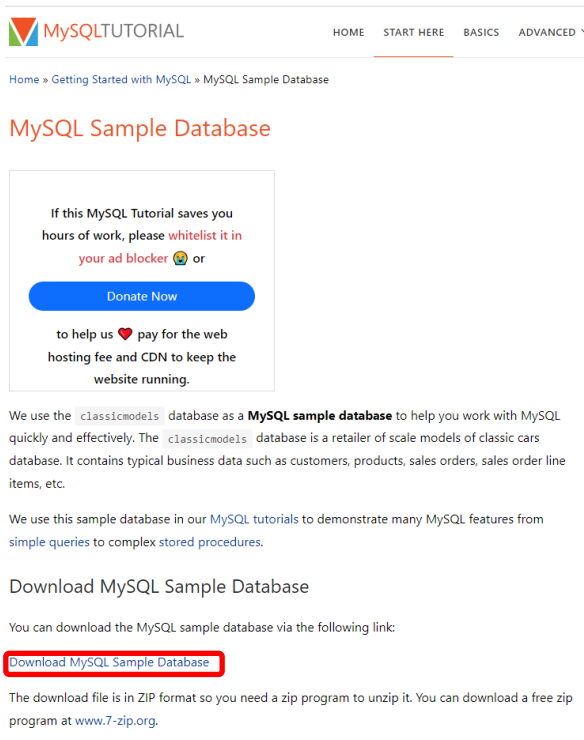
- 확장자 .CSV 의 데이터를 TABLE에 넣는 기능을 HEIDISQL에서 편리하게 제공.



- 새 테이블을 선택하면 자동으로 데이터의 타입을 추정해서 제시를 해준다.



MYSQL SAMPLE DATABASE



The screenshot shows the MySQL Tutorial website. The header includes the MySQL Tutorial logo and navigation links: HOME, START HERE, BASICS, and ADVANCED. The breadcrumb trail reads: Home » Getting Started with MySQL » MySQL Sample Database. The main heading is "MySQL Sample Database". Below this is a donation request box with the text: "If this MySQL Tutorial saves you hours of work, please whitelist it in your ad blocker or Donate Now to help us pay for the web hosting fee and CDN to keep the website running." The main content area explains that the website uses the "classicmodels" database as a "MySQL sample database" to help users work with MySQL. It describes the "classicmodels" database as a retailer of scale models of classic cars, containing typical business data. It also mentions that the sample database is used in MySQL tutorials to demonstrate various features. A section titled "Download MySQL Sample Database" provides a link to download the database. The link "Download MySQL Sample Database" is highlighted with a red box. Below the link, it states that the download file is in ZIP format and provides a link to a free zip program at www.7-zip.org.

MySQL Tutorial

HOME START HERE BASICS ADVANCED

Home » Getting Started with MySQL » MySQL Sample Database

MySQL Sample Database

If this MySQL Tutorial saves you hours of work, please whitelist it in your ad blocker or [Donate Now](#) to help us pay for the web hosting fee and CDN to keep the website running.

We use the `classicmodels` database as a **MySQL sample database** to help you work with MySQL quickly and effectively. The `classicmodels` database is a retailer of scale models of classic cars database. It contains typical business data such as customers, products, sales orders, sales order line items, etc.

We use this sample database in our [MySQL tutorials](#) to demonstrate many MySQL features from simple queries to complex stored procedures.

Download MySQL Sample Database

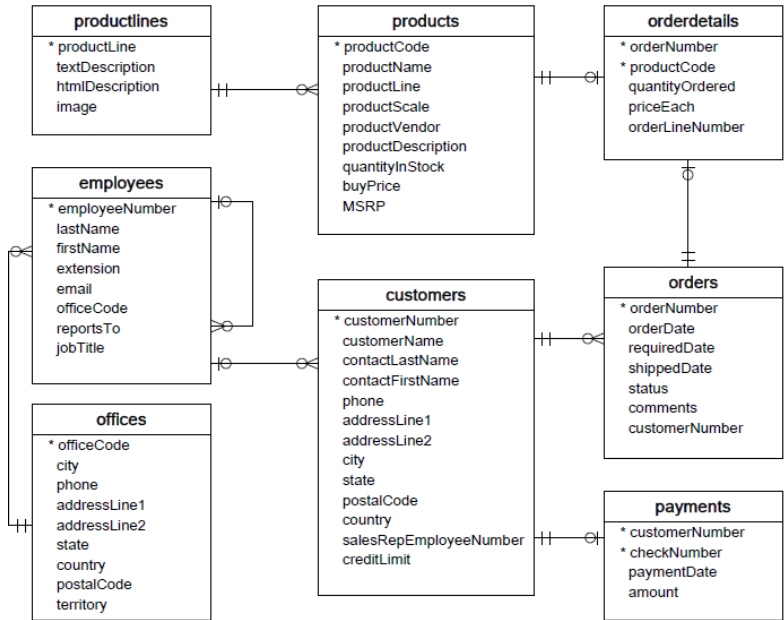
You can download the MySQL sample database via the following link:

[Download MySQL Sample Database](#)

The download file is in ZIP format so you need a zip program to unzip it. You can download a free zip program at www.7-zip.org.

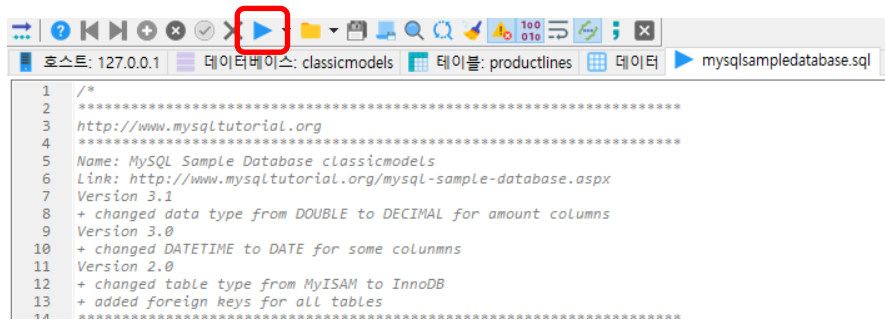
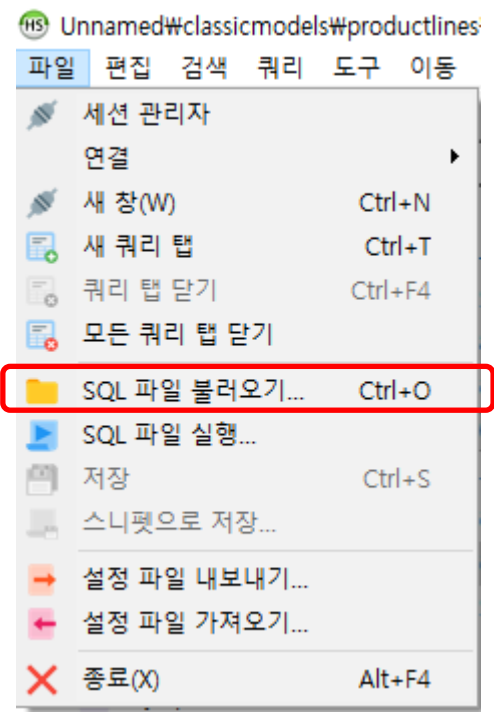
- <https://www.mysqltutorial.org/mysql-sample-database.aspx>

MYSQL SAMPLE DATABASE



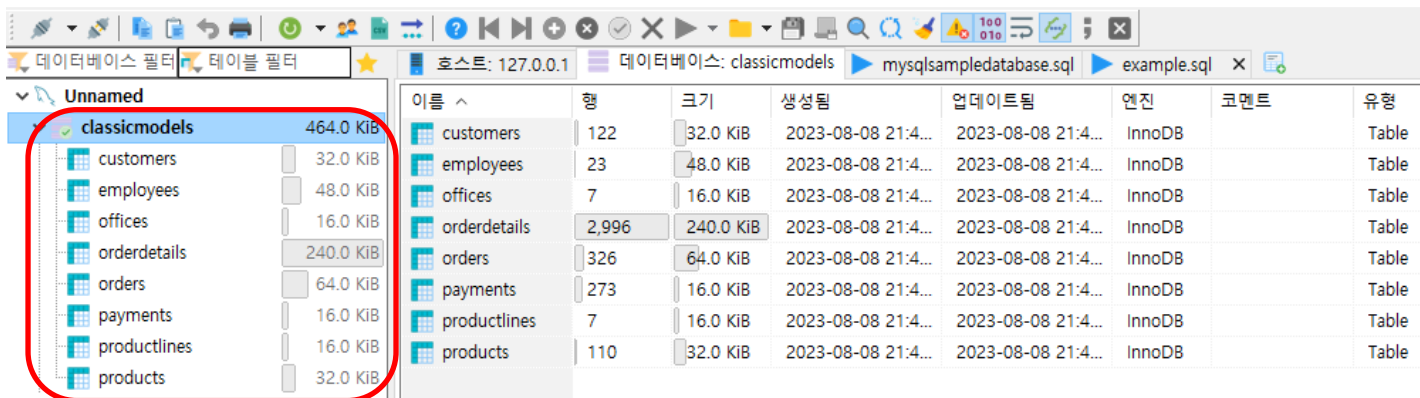
- MYSQLTUTORIAL
홈페이지에서 제공하는
SAMPLE DATA BASE의 구성.

DATABASE 불러오기



- 스크립트의 구성을 보면 CREATE TABLE & INSERT 함수로 테이블 생성과 데이터 입력을 동시에 진행한다.

DATABASE 불러오기



The screenshot shows a database management interface. On the left, a tree view under 'Unnamed' shows the 'classicmodels' database (464.0 KiB) expanded, listing tables: customers (32.0 KiB), employees (48.0 KiB), offices (16.0 KiB), orderdetails (240.0 KiB), orders (64.0 KiB), payments (16.0 KiB), productlines (16.0 KiB), and products (32.0 KiB). The 'classicmodels' database name is circled in red. On the right, a table lists the details of these tables.

이름 ^	행	크기	생성됨	업데이트됨	엔진	코멘트	유형
customers	122	32.0 KiB	2023-08-08 21:4...	2023-08-08 21:4...	InnoDB		Table
employees	23	48.0 KiB	2023-08-08 21:4...	2023-08-08 21:4...	InnoDB		Table
offices	7	16.0 KiB	2023-08-08 21:4...	2023-08-08 21:4...	InnoDB		Table
orderdetails	2,996	240.0 KiB	2023-08-08 21:4...	2023-08-08 21:4...	InnoDB		Table
orders	326	64.0 KiB	2023-08-08 21:4...	2023-08-08 21:4...	InnoDB		Table
payments	273	16.0 KiB	2023-08-08 21:4...	2023-08-08 21:4...	InnoDB		Table
productlines	7	16.0 KiB	2023-08-08 21:4...	2023-08-08 21:4...	InnoDB		Table
products	110	32.0 KiB	2023-08-08 21:4...	2023-08-08 21:4...	InnoDB		Table

- 스크립트의 실행 결과로 데이터베이스 classicmodels 가 생성.
- 그 데이터베이스 속에 8개의 table이 생성.

DATABASE 불러오기

classicmodels.orderdetails: 2,996 행 (총) (대략적), 제한 수: 1,000

orderNumber	productCode	quantityOrdered	priceEach	orderLineNumber
10,100	S18_1749	30	136.0	3
10,100	S18_2248	50	55.09	2
10,100	S18_4409	22	75.46	4
10,100	S24_3969	49	35.29	1
10,101	S18_2325	25	108.06	4
10,101	S18_2795	26	167.06	1
10,101	S24_1937	45	32.53	3
10,101	S24_2022	46	44.35	2
10,102	S18_1342	39	95.55	2
10,102	S18_1367	41	43.13	1
10,103	S10_1949	26	214.3	11
10,103	S10_4962	42	119.67	4
10,103	S12_1666	27	121.64	8
10,103	S18_1097	35	94.5	10
10,103	S18_2432	22	58.34	2
10,103	S18_2949	27	92.19	12
10,103	S18_2957	35	61.84	14

- Orderdetails Table
- Ordernumber : 주문번호
- productCode : 제품번호
- quantityOrdered : 수량
- priceEach : 개당가격
- orderLineNumber : 주문라인번호

6.

데이터 불러오기

SELECT FROM

SELECT FROM

- SQL Code
- SELECT * FROM 테이블명;
- 여기서 * 는 모든 컬럼

```
SELECT * FROM `customers`  
;
```

- 실행결과

customers (3r × 3c)		
customer_number	customer_name	phone
1	이상훈	010-1234-5678
2	김상훈	010-1234-5679
3	박상훈	010-1234-5679

SELECT FROM

- SQL Code
- SELECT 컬럼명1, 컬럼명2 FROM 테이블명;

```
SELECT customer_name, phone FROM customers  
;
```

- 실행결과

customers (3r × 2c)	
customer_name	phone
이상호	010-1234-5678
김상호	010-1234-5679
박상호	010-1234-5679

ALIAS(별칭)

- SQL Code
- SELECT 컬럼명 AS 별칭 FROM 테이블명;
- 사용하는 경우
- SELECT 구문에서 별칭을 사용하는 경우, 컬럼명을 바꾸기 위해서 사용.
- 뒤에서 배우는 서브쿼리에서는 테이블 등에 별칭을 부여할 수 있음.

ALIAS(별칭)

- 원본테이블

수학	영어	이름
75	100	이상훈
99	50	김상훈
40	30	조상훈

- 컬럼의 합

```
SELECT 이름, 영어, 수학, 수학+영어
FROM ex1
;
```

이름	영어	수학	수학+영어
이상훈	100	75	175
김상훈	50	99	149
조상훈	30	40	70

컬럼의 합을 별칭으로 표현

```
SELECT 이름, 영어, 수학, 수학+영어 AS 영수
FROM ex1
;
```

이름	영어	수학	영수
이상훈	100	75	175
김상훈	50	99	149
조상훈	30	40	70

LIMIT / OFFSET

- **LIMIT**과 **OFFSET**은 주로 결과 집합의 특정 부분을 제한하거나 건너뛸 때 사용
- **LIMIT** 은 반환되는 레코드의 수를 제한하는 데 사용
- 예를 들어, 결과의 처음 5개 레코드만을 선택하려면 **LIMIT 5**을 사용

LIMIT / OFFSET

- **OFFSET** 은 처음 몇 개의 레코드를 건너뛸 것인지를 지정하는 데 사용
- 예를 들어, 처음 5개 레코드를 건너뛰고 그 다음 10개 레코드를 선택하려면 **LIMIT 10 OFFSET 5**를 사용합니다

LIMIT n

- 원본테이블

번호	이름	국어	영어	수학
1	a	64	2	97
2	b	86	40	33
3	c	92	20	86
4	d	58	27	13
5	e	42	39	2
6	f	75	85	3
7	g	40	71	65
8	h	36	65	57
9	i	6	29	91
10	g	30	28	44
11	k	10	45	88
12	l	51	88	71
13	m	38	50	71
14	j	86	62	5
15	o	16	88	64
16	p	35	80	4
17	q	73	56	34
18	r	72	26	97

- SQL Code

```
SELECT * FROM exercise1 LIMIT 5
;
```

- Limit n : 0번째 행부터
5개의 행 출력

- 결과

번호	이름	국어	영어	수학
1	a	64	2	97
2	b	86	40	33
3	c	92	20	86
4	d	58	27	13
5	e	42	39	2

LIMIT n, m

- 원본테이블

번호	이름	국어	영어	수학
1	a	64	2	97
2	b	86	40	33
3	c	92	20	86
4	d	58	27	13
5	e	42	39	2
6	f	75	85	3
7	g	40	71	65
8	h	36	65	57
9	i	6	29	91
10	g	30	28	44
11	k	10	45	88
12	l	51	88	71
13	m	38	50	71
14	j	86	62	5
15	o	16	88	64
16	p	35	80	4
17	q	73	56	34
18	r	72	26	97

- SQL Code

```
SELECT * FROM exercise1 LIMIT 5, 3
;
```

- Limit n : 5번째 행부터
3개의 행을 출력

- 결과

번호	이름	국어	영어	수학
6	f	75	85	3
7	g	40	71	65
8	h	36	65	57

LIMIT n OFFSET m

- 원본테이블

번호	이름	국어	영어	수학
1	a	64	2	97
2	b	86	40	33
3	c	92	20	86
4	d	58	27	13
5	e	42	39	2
6	f	75	85	3
7	g	40	71	65
8	h	36	65	57
9	i	6	29	91
10	g	30	28	44
11	k	10	45	88
12	l	51	88	71
13	m	38	50	71
14	j	86	62	5
15	o	16	88	64
16	p	35	80	4
17	q	73	56	34
18	r	72	26	97

- SQL Code

```
SELECT * FROM exercise1 LIMIT 3 OFFSET 5;
```

- OFFSET 으로 5개의 데이터 다음 3개의 데이터 출력.
- LIMIT 5,3 과 같은 결과

- 결과

번호	이름	국어	영어	수학
6	f	75	85	3
7	g	40	71	65
8	h	36	65	57

DISTINCT

- 원본테이블

번호	이름	국어	영어	수학
1	a	64	2	97
2	b	86	40	33
3	c	92	20	86
4	d	58	27	13
5	e	42	39	2
6	f	75	85	3
7	g	40	71	65
8	h	36	65	57
9	i	6	29	91
10	g	30	28	44
11	k	10	45	88
12	l	51	88	71
13	m	38	50	71
14	j	86	62	5
15	o	16	88	64
16	p	35	80	4
17	q	73	56	34
18	r	72	26	97

- SQL Code

```
SELECT DISTINCT 수학 FROM exercise1
;
```

- 수학 컬럼에서 중복된 항목 하나만 남기고 선택
- 단점은 그 행만 보여지는것

수학
97
33
86
13
2
3
65
57
91
44
88
71
5
64
4
34

실습

- 문제 1: customers 테이블에서 customerName을 가져오되, 중복 없이 가져오세요.

정답

```
SELECT DISTINCT customerName  
FROM customers;
```

실습

- 문제 2: products 테이블에서 제품 이름(productName)만 5개만 가져오세요.

정답

```
SELECT productName  
FROM products  
LIMIT 5;
```


실습

- 문제 3: orders 테이블에서 주문 상태(status)의 종류를 모두 나열하세요. 중복은 제거하세요.

정답

```
SELECT DISTINCT status  
FROM orders;
```

실습

- 문제 4: employees 테이블에서 직원의 성(lastName)을 10개만 가져오되, 5번째부터 시작하세요.

정답

```
SELECT lastName  
FROM employees  
LIMIT 4, 10;
```

실습

- 문제 5: products 테이블의 productVendor를 별칭(alias) Vendor로 조회하세요.

정답

```
SELECT productVendor AS Vendor  
FROM products;
```

실습

- 문제 6: orders 테이블에서 고객 번호(customerNumber)의 중복 없는 값을 7개만 가져오세요.

정답

```
SELECT DISTINCT customerNumber
FROM orders
LIMIT 7;
```

실습

- 문제 7: employees 테이블에서 employeeNumber를 별칭 Employee_ID로 조회하되, 3번째부터 6개만 가져오세요.

정답

```
SELECT employeeNumber AS Employee_ID  
FROM employees  
LIMIT 2, 6;
```

실습

- 문제 8: offices 테이블에서 국가(country)의 종류를 중복 없이 나열하세요.

정답

```
SELECT DISTINCT country  
FROM offices;
```

실습

- 문제 9: orderdetails 테이블에서 제품 코드(productCode)를 Code라는 별칭으로, 주문 수량(quantityOrdered)를 Quantity라는 별칭으로 조회하세요.

정답

```
SELECT productCode AS Code, quantityOrdered AS Quantity  
FROM orderdetails;
```

실습

- 문제 10: payments 테이블에서 체크 번호(checkNumber)의 중복 없는 값을 10개만 가져오세요.

정답

```
SELECT DISTINCT checkNumber  
FROM payments  
LIMIT 10;
```


데이터분석 기초 SQL 부트캠프

목차

1. WHERE 구문

비교연산자(<,>=)

논리연산자(AND, OR, NOT)

LIKE, IN ,BETWEEN, ISNULL

2. ORDER BY구문

3. GROUP BY 구문

6.

WHERE구문

비교연산자(<,>=), 논리연산자(AND, OR, NOT)
LIKE, IN ,BETWEEN, ISNULL

비교연산자(<, >, =)

- SQL Code

```
select  
    컬럼  
from  
    테이블  
where  
    조건
```

비교연산자(<, >, =)

- SQL Code

```
SELECT *
FROM orderdetails
WHERE orderlinenumber = 1
;
```

- 결과

orderNumber	productCode	quantityOrdered	priceEach	orderLineNumber
10,100	S24_3969	49	35.29	1
10,101	S18_2795	26	167.06	1
10,102	S18_1367	41	43.13	1
10,103	S24_2300	36	107.34	1
10,104	S12_3148	34	131.44	1
10,105	S24_3816	50	75.47	1
10,106	S700_2834	32	113.9	1
10,107	S12_2823	21	122.0	1
10,108	S24_3856	40	132.0	1
10,109	S18_2870	26	126.72	1
10,110	S18_2795	31	163.69	1
10,111	S18_3136	43	94.25	1
10,112	S10_1949	29	197.16	1
10,113	S32_3522	23	58.82	1

비교연산자(<, >, =)

- SQL Code

```
SELECT *
FROM orderdetails
WHERE productcode =
's24_3969'
;
```

- 결과

orderNumber	productCode	quantityOrdered	priceEach	orderLineNumber
10,100	S24_3969	49	35.29	1
10,110	S24_3969	48	35.29	5
10,124	S24_3969	46	36.11	4
10,138	S24_3969	29	32.82	4
10,149	S24_3969	26	38.57	9
10,162	S24_3969	37	32.82	7
10,173	S24_3969	35	35.7	11
10,182	S24_3969	23	34.88	8
10,193	S24_3969	22	38.16	12
10,204	S24_3969	39	34.88	2
10,214	S24_3969	44	38.57	5
10,227	S24_3969	27	34.88	8
10,242	S24_3969	46	36.52	1

비교연산자(<, >, =)

- SQL Code

```
SELECT *
FROM orderdetails
WHERE quantityordered > 70
;
```

- 결과

orderdetails (8r x 5c)					
orderNumber	productCode	quantityOrdered	priceEach	orderLineNumber	
10,401	S700_2466	85	98.72	10	
10,401	S700_3167	77	73.6	9	
10,404	S12_3990	77	67.03	4	
10,404	S18_3278	90	67.54	6	
10,405	S12_4675	97	115.16	5	
10,405	S24_3856	76	127.79	3	
10,407	S18_1749	76	141.1	2	
10,407	S24_2766	76	81.78	6	

비교연산자(<, >, =)

- SQL Code

```
SELECT *  
FROM orderdetails  
WHERE quantityordered >=  
70  
;
```

- 결과

orderdetails (10r x 5c)					
orderNumber	productCode	quantityOrdered	priceEach	orderLineNumber	
10,401	S700_2466	85	98.72	10	
10,401	S700_3167	77	73.6	9	
10,404	S12_3990	77	67.03	4	
10,404	S18_3278	90	67.54	6	
10,405	S12_4675	97	115.16	5	
10,405	S24_3856	76	127.79	3	
10,407	S18_1749	76	141.1	2	
10,407	S24_2766	76	81.78	6	
10,412	S24_2300	70	109.9	10	
10,419	S24_3856	70	112.34	8	

비교연산자(<, >, =) 실습

- 문제 1. customers 테이블에서 creditLimit이 10000보다 큰 고객들의 이름(customerName)을 조회하세요..

- 정답

customers (98r x 1c)	
customerName	
Atelier graphique	
Signal Gift Stores	
Australian Collectors, Co.	
La Rochelle Gifts	
Baane Mini Imports	
Mini Gifts Distributors Ltd.	
Blauer See Auto, Co.	
Mini Wheels Co.	
Land of Toys Inc.	

논리연산자(AND, OR, NOT)

- SQL Code

```
SELECT *  
FROM orderdetails  
WHERE quantityordered = 46  
AND  
productcode = 's24_3969'  
;
```

- 결과

orderdetails (3r x 5c)					
orderNumber	productCode	quantityOrdered	priceEach	orderLineNumber	
10,124	s24_3969	46	36.11	4	
10,242	s24_3969	46	36.52	1	
10,368	s24_3969	46	36.52	3	

논리연산자(AND, OR, NOT)

- SQL Code

```
SELECT *
FROM orderdetails
WHERE quantityordered = 46
OR
productcode = 's24_3969'
;
```

- 결과

orderNumber	productCode	quantityOrdered	priceEach	orderLineNumber
10,100	S24_3969	49	35.29	1
10,101	S24_2022	46	44.35	2
10,103	S18_3320	46	86.31	16
10,109	S18_3232	46	160.87	5
10,110	S24_2887	46	112.74	10
10,110	S24_3969	48	35.29	5
10,115	S12_4473	46	111.39	5
10,115	S18_2238	46	140.81	4
10,119	S10_4757	46	112.88	11
10,120	S10_4698	46	158.8	2
10,120	S18_2625	46	57.54	4
10,123	S18_2870	46	114.84	3
10,124	S24_3969	46	36.11	4
10,126	S18_2957	46	61.84	14
10,127	S12_1108	46	193.25	2

논리연산자(AND, OR, NOT)

- AND 와 OR을 같이 사용하는 경우 우선순위는 AND에 있음
- OR 가 우선적으로 실행되기 위해서는 소괄호 활용()

논리연산자(AND, OR, NOT)

- SQL Code

```
SELECT *
FROM orderdetails
WHERE priceEach = 35.29
AND
productcode = 's24_3969'
OR
quantityordered = 46
;
```

- 결과

orderNumber	productCode	quantityOrdered	priceEach	orderLineNumber
10,100	s24_3969	49	35.29	1
10,110	s24_3969	48	35.29	5
10,280	s24_3969	33	35.29	14
10,420	s24_3969	15	35.29	3
10,394	s24_2840	46	35.36	6
10,124	s24_3969	46	36.11	4
10,242	s24_3969	46	36.52	1
10,368	s24_3969	46	36.52	3
10,190	s32_2206	46	38.62	1
10,215	s18_4668	46	42.76	1

논리연산자(AND, OR, NOT)

- SQL Code

```
SELECT *
FROM orderdetails
WHERE (priceEach = 35.29
AND
productcode = 's24_3969'
)OR
quantityordered = 46
;
```

- 결과

orderNumber	productCode	quantityOrdered	priceEach	orderLineNumber
10,100	524_3969	49	35.29	1
10,110	524_3969	48	35.29	5
10,280	524_3969	33	35.29	14
10,420	524_3969	15	35.29	3
10,394	524_2840	46	35.36	6
10,124	524_3969	46	36.11	4
10,242	524_3969	46	36.52	1
10,368	524_3969	46	36.52	3
10,190	532_2206	46	38.62	1
10,215	518_4668	46	42.76	1

- AND의 우선순위가 앞서기 때문에 같은 결과 출력

논리연산자(AND, OR, NOT)

- SQL Code

```
SELECT *
FROM orderdetails
WHERE priceEach = 35.29
AND
(productcode = 's24_3969'
OR
quantityordered = 46 )
;
```

- 결과

orderdetails (4r x 5c)					
orderNumber	productCode	quantityOrdered	priceEach	orderLineNumber	
10,100	s24_3969	49	35.29	1	
10,110	s24_3969	48	35.29	5	
10,280	s24_3969	33	35.29	14	
10,420	s24_3969	15	35.29	3	

- 괄호를 통해서 **OR** 조건을 우선순위로 하면 다른 결과가 출력

논리연산자(AND, OR, NOT) 실습

- 문제 2: orders 테이블에서 status가 'Shipped' 이거나 'In Process'인 주문의 orderNumber를 조회하세요.

- 정답

orders (309r x 1c)	
orderNumber	
10,100	
10,101	
10,102	
10,103	
10,104	
10,105	
10,106	
10,107	

논리연산자(AND, OR, NOT) 실습

- 문제 3: products 테이블에서 재고(quantityInStock)가 100개 미만이거나 500개 초과인 제품의 productName을 조회하세요.

- 정답

products (107r × 1c)	
productName	
1969 Harley Davidson Ultimate Chopper	
1952 Alpine Renault 1300	
1996 Moto Guzzi 1100i	
2003 Harley-Davidson Eagle Drag Bike	
1972 Alfa Romeo GTA	
1962 LanciaA Delta 16V	
1968 Ford Mustang	
2001 Ferrari Enzo	
1958 Setra Bus	
2002 Suzuki XREO	

논리연산자(AND, OR, NOT) 실습

- 문제 4: products 테이블에서 제품의 가격(buyPrice)이 50보다 크고 재고(quantityInStock)가 100보다 작은 제품의 productName을 조회하세요.

- 정답

products (1r × 1c)	
productName	
1968 Ford Mustang	

LIKE

- **LIKE** 함수는 문자열에서 원하는 문자가 포함되어 있는지를 검색
- **SQL Code**
- **SELECT * FROM table**
- **WHERE LIKE '%CARS'** : CARS 로 끝나는 데이터 검색
- **WHERE LIKE 'CARS%'** : CARS 로 시작하는 데이터 검색
- **WHERE LIKE '%CARS'%** : CARS를 포함하는 데이터 검색

LIKE

- Product table

classicmodels.products: 110 행 (총) (대략적)

» 더

productCode	productName	productLine	productScale	productVendor	productDescription	quantityInStock	buyPrice	MSRP
S10_1678	1969 Harley Davidson Ultimate Chopper	Motorcycles	1:10	Min Lin Diecast	This replica features working kickstand, front suspension...	7,933	48.81	95.7
S10_1949	1952 Alpine Renault 1300	Classic Cars	1:10	Classic Metal Creations	Turnable front wheels; steering function; detailed interio...	7,305	98.58	214.3
S10_2016	1996 Moto Guzzi 1100i	Motorcycles	1:10	Highway 66 Mini Classics	Official Moto Guzzi logos and insignias, saddle bags locat...	6,625	68.99	118.94
S10_4698	2003 Harley-Davidson Eagle Drag Bike	Motorcycles	1:10	Red Start Diecast	Model features, official Harley Davidson logos and insigni...	5,582	91.02	193.66
S10_4757	1972 Alfa Romeo GTA	Classic Cars	1:10	Motor City Art Classics	Features include: Turnable front wheels; steering functio...	3,252	85.68	136.0
S10_4962	1962 LanciaA Delta 16V	Classic Cars	1:10	Second Gear Diecast	Features include: Turnable front wheels; steering functio...	6,791	103.42	147.74
S12_1099	1968 Ford Mustang	Classic Cars	1:12	Autoart Studio Design	Hood, doors and trunk all open to reveal highly detailed i...	68	95.34	194.57
S12_1108	2001 Ferrari Enzo	Classic Cars	1:12	Second Gear Diecast	Turnable front wheels; steering function; detailed interio...	3,619	95.59	207.8
S12_1666	1958 Setra Bus	Trucks and Buses	1:12	Welly Diecast Productions	Model features 30 windows, skylights & glare resistant gl...	1,579	77.9	136.67
S12_2823	2002 Suzuki XREO	Motorcycles	1:12	Unimax Art Galleries	Official logos and insignias, saddle bags located on side o...	9,997	66.27	150.62
S12_3148	1969 Corvair Monza	Classic Cars	1:18	Welly Diecast Productions	1:18 scale die-cast about 10" long doors open, hood ope...	6,906	89.14	151.08

- Productline column을 보면 다양한 상품이 있는 것 같다.
- 어떤 상품이 있는지 확인하기 위해 distinct 를 활용

LIKE

- SQL Code

```
SELECT DISTINCT  
productline  
FROM products  
;
```

- 결과

products (7r x 1c)	
productline	
Classic Cars	
Motorcycles	
Planes	
Ships	
Trains	
Trucks and Buses	
Vintage Cars	

- 결과는 총 7개의 제품이며 자동차 종류는 Classic Cars와 Vintage Cars로 두 종류

LIKE

- SQL Code

```
SELECT *  
FROM products  
WHERE productline = 'vintage cars' OR productline = 'classic cars'  
;
```

```
SELECT *  
FROM products  
WHERE productline LIKE "%cars"  
;
```

- 위 두 코드는 서로 같은 결과를 출력.
- Like '%cars%' 도 가능

LIKE 실습

- 문제 5: employees 테이블에서 jobTitle에 'Sales'라는 단어가 포함된 직원의 firstName과 lastName을 조회하세요.

- 정답

employees (20r x 2c)	
firstName	lastName
Mary	Patterson
William	Patterson
Anthony	Bow
Leslie	Jennings
Leslie	Thompson
Julie	Firrelli
Steve	Patterson
Foon Yue	Tseng

BETWEEN


- **BETWEEN** 함수는 A 이상 B 이하의 데이터를 출력
- **SQL Code**
- `SELECT * FROM table`
`WHERE column1 BETWEEN 10 AND 30`
-> column1 이 10 이상 30 이하인 모든 데이터 출력

BETWEEN

- SQL Code

```
SELECT *  
FROM orders  
WHERE orderdate  
BETWEEN '2003-01-01' AND '2003-01-31'  
;
```

- 결과

orders (5r × 7c)		
orderNumber		orderDate
10,100		2003-01-06
10,101		2003-01-09
10,102		2003-01-10
10,103		2003-01-29
10,104		2003-01-31

- BETWEEN A AND B** : A 와 B
포함의 데이터 출력

BETWEEN

- SQL Code

```
SELECT *  
FROM orders  
WHERE orderdate BETWEEN '2003-01-01' AND '2003-01-31'  
;
```

```
SELECT *  
FROM orders  
WHERE orderdate >= '2003-01-01' AND orderdate <= '2003-01-31'  
;
```

- 위 두 코드는 서로 같은 결과를 출력.

BETWEEN 실습

- 문제 6: orderdetails 테이블에서 주문 수량(quantityOrdered)이 10개 이상 50개 이하인 주문의 orderNumber를 조회하세요.

- 정답

orderdetails (2,928r × 1c)	
orderNumber	
10,100	
10,100	
10,100	
10,100	
10,101	
10,101	
10,101	
10,101	

NOT BETWEEN

- NOT BETWEEN 함수는 A 이상 B 이하를 제외한 데이터를 출력
- SQL Code
- ```
SELECT * FROM table
WHERE column1 NOT BETWEEN 10 AND 30
```

-> column1 이 10 이상 30 이하 제외 모든 데이터 출력

## NOT BETWEEN

- SQL Code

```
SELECT *
FROM orders
WHERE orderdate NOT BETWEEN '2003-01-01' AND '2003-01-31'
;
```

```
SELECT *
FROM orders
WHERE orderdate < '2003-01-01' OR orderdate > '2003-01-31'
;
```

- 위 두 코드는 서로 같은 결과를 출력.

## IN

- IN 은 특정 값이 있을 때 조회
- SQL Code
- ```
SELECT * FROM table  
WHERE column1 IN (10, 20, 30)
```

-> column1 이 10, 20, 30 인 데이터 조회


IN

- SQL Code

```
SELECT *  
FROM orders  
WHERE orderdate IN ('2003-  
02-11', '2003-02-17')  
;
```

- WHERE orderdate = '2003-02-11' OR orderdate = '2003-02-17' 과 같은 결과

- 결과

orders (2r x 7c)				
orderNumber		orderDate	requiredDate	shippedDate
10,105		2003-02-11	2003-02-21	2003-02-12
10,106		2003-02-17	2003-02-24	2003-02-21

IN 실습

- 문제 7: customers 테이블에서 국가(country)가 'USA', 'Canada', 'France' 중 하나인 고객의 customerName을 조회하세요.

- 정답

customers (51r × 1c)	
customerName	
Atelier graphique	
Signal Gift Stores	
La Rochelle Gifts	
Mini Gifts Distributors Ltd.	
Mini Wheels Co.	
Land of Toys Inc.	
Saveley & Henriot, Co.	
Muscle Machine Inc	
Diecast Classics Inc.	

NOT IN

- NOT IN 은 특정 값이 포함되지 않은 데이터 조회
- SQL Code
- ```
SELECT * FROM table
WHERE column1 NOT IN (10, 20, 30)
```

-> column1 이 10, 20, 30 이 아닌 데이터 조회

## NOT IN

- SQL Code

```
SELECT *
FROM employees
WHERE officecode not IN (1,2,3)
;
```

```
SELECT *
FROM employees
WHERE officecode <> 1 and officecode <> 2 and officecode <>3
;
```

- 위 두 코드는 서로 같은 결과를 출력.

## IS NULL / IS NOT NULL

- **IS NULL** : COLUMN 의 값이 NULL 인 데이터 조회
- **IS NOT NULL** : COLUMN 의 값이 NULL 이 아닌 값 조회

```
SELECT *
FROM orders
WHERE comments IS NULL
;
```

| orders (246r × 7c) |            |              |             |         |          |  |
|--------------------|------------|--------------|-------------|---------|----------|--|
| orderNumber        | orderDate  | requiredDate | shippedDate | status  | comments |  |
| 10,100             | 2003-01-06 | 2003-01-13   | 2003-01-10  | Shipped | (NULL)   |  |
| 10,102             | 2003-01-10 | 2003-01-18   | 2003-01-14  | Shipped | (NULL)   |  |
| 10,103             | 2003-01-29 | 2003-02-07   | 2003-02-02  | Shipped | (NULL)   |  |

```
SELECT *
FROM orders
WHERE comments IS not NULL
;
```

| orders (80r × 7c) |            |              |             |         |                                                            |  |
|-------------------|------------|--------------|-------------|---------|------------------------------------------------------------|--|
| orderNumber       | orderDate  | requiredDate | shippedDate | status  | comments                                                   |  |
| 10,101            | 2003-01-09 | 2003-01-18   | 2003-01-11  | Shipped | Check on availability.                                     |  |
| 10,107            | 2003-02-24 | 2003-03-03   | 2003-02-26  | Shipped | Difficult to negotiate with customer. We need more mark... |  |
| 10,109            | 2003-03-10 | 2003-03-19   | 2003-03-11  | Shipped | Customer requested that FedEx Ground is used for this s... |  |

## IS NULL 실습

- 문제 8: employees 테이블에서 상사(reportsTo)가 지정되지 않은 직원의 firstName과 lastName을 조회하세요.

- 정답

| employees (1r x 2c) |          |
|---------------------|----------|
| firstName           | lastName |
| Diane               | Murphy   |

7.

**ORDER BY구문**

## ORDER BY

- **ORDER BY** : SQL 쿼리의 결과를 특정 컬럼 또는 여러 컬럼을 기준으로 정렬. 기본적으로 오름차순(ASC)으로 정렬되며, 내림차순(DISC)으로 정렬하려면 DESC 키워드를 사용

```
SELECT *
FROM A
WHERE ~~
ORDER BY 컬럼명1 desc, 컬럼명2 asc, ...
```

- ORDER BY 는 WHERE 절(생략 가능) 다음에 사용.

## ORDER BY ASC

- SQL Code

```
SELECT LASTNAME,
FIRSTNAME, OFFICECODE
FROM employees
ORDER BY OFFICECODE
LIMIT 8;
```

- 결과

| employees (8r x 3c) |           |            |
|---------------------|-----------|------------|
| LASTNAME            | FIRSTNAME | OFFICECODE |
| Murphy              | Diane     | 1          |
| Patterson           | Mary      | 1          |
| Firrelli            | Jeff      | 1          |
| Bow                 | Anthony   | 1          |
| Jennings            | Leslie    | 1          |
| Thompson            | Leslie    | 1          |
| Firrelli            | Julie     | 2          |
| Patterson           | Steve     | 2          |

- 오름차순 ACS는 생략 가능

## ORDER BY DESC

- SQL Code

```
SELECT LASTNAME,
FIRSTNAME, OFFICECODE
FROM employees
ORDER BY OFFICECODE DESC
LIMIT 8;
```

- 결과

| LASTNAME  | FIRSTNAME | OFFICECODE |  |
|-----------|-----------|------------|--|
| Bott      | Larry     | 7          |  |
| Jones     | Barry     | 7          |  |
| Patterson | William   | 6          |  |
| Fixter    | Andy      | 6          |  |
| Marsh     | Peter     | 6          |  |
| King      | Tom       | 6          |  |
| Nishi     | Mami      | 5          |  |
| Kato      | Yoshimi   | 5          |  |

- OFFICECODE 로 내림차순 정렬



## ORDER BY 다중정렬

- SQL Code

```
SELECT *
FROM orderdetails
ORDER BY
quantityordered DESC ,
priceEach asc
;
```

- 결과

| orderNumber | productCode | quantityOrdered | priceEach | orderLineNumber |
|-------------|-------------|-----------------|-----------|-----------------|
| 10,405      | S12_4675    | 97              | 115.16    | 5               |
| 10,404      | S18_3278    | 90              | 67.54     | 6               |
| 10,401      | S700_2466   | 85              | 98.72     | 10              |
| 10,404      | S12_3990    | 77              | 67.03     | 4               |
| 10,401      | S700_3167   | 77              | 73.6      | 9               |
| 10,407      | S24_2766    | 76              | 81.78     | 6               |
| 10,405      | S24_3856    | 76              | 127.79    | 3               |
| 10,407      | S18_1749    | 76              | 141.1     | 2               |

- quantityordered 로 먼저 내림차순 정렬,  
이후 priceEach로 오름차순 정렬

## ORDER BY 다중정렬

- SQL Code

```
SELECT *
FROM orderdetails
ORDER BY
quantityordered DESC ,
priceEach desc
;
```

- 결과

| orderNumber | productCode | quantityOrdered | priceEach | orderLineNumber |
|-------------|-------------|-----------------|-----------|-----------------|
| 10,405      | 512_4675    | 97              | 115.16    | 5               |
| 10,404      | 518_3278    | 90              | 67.54     | 6               |
| 10,401      | 5700_2466   | 85              | 98.72     | 10              |
| 10,401      | 5700_3167   | 77              | 73.6      | 9               |
| 10,404      | 512_3990    | 77              | 67.03     | 4               |
| 10,407      | 518_1749    | 76              | 141.1     | 2               |
| 10,405      | 524_3856    | 76              | 127.79    | 3               |
| 10,407      | 524_2766    | 76              | 81.78     | 6               |

- quantityordered 로 먼저 내림차순 정렬,  
이후 priceEach로 내림차순 정렬

## ORDER BY 실습

- 문제 9: products 테이블에서 제품 이름(productName)을 조회하되, 가격(buyPrice)을 기준으로 내림차순으로 정렬하세요.

- 정답

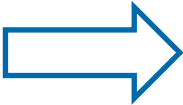
| products (110r × 1c)                 |  |
|--------------------------------------|--|
| productName                          |  |
| 1962 LanciaA Delta 16V               |  |
| 1998 Chrysler Plymouth Prowler       |  |
| 1952 Alpine Renault 1300             |  |
| 1956 Porsche 356A Coupe              |  |
| 2001 Ferrari Enzo                    |  |
| 1968 Ford Mustang                    |  |
| 1995 Honda Civic                     |  |
| 1970 Triumph Spitfire                |  |
| 2003 Harley-Davidson Eagle Drag Bike |  |
| 1969 Corvair Monza                   |  |

8.

**GROUP BY구문**

# GROUP BY

| 종목명      | 섹터  | 상장주식수 |
|----------|-----|-------|
| 삼성전자     | 반도체 | 100   |
| 셀트리온     | 바이오 | 150   |
| SK하이닉스   | 반도체 | 200   |
| 삼성바이오로직스 | 바이오 | 300   |
| NAVER    | IT  | 400   |
| 카카오      | IT  | 500   |



| 지수  | 섹터별<br>상장주식수 |
|-----|--------------|
| 반도체 | 300          |
| 바이오 | 450          |
| 반도체 | 900          |

- GROUP BY 절은 한 개 이상의 컬럼을 기준으로 결과를 그룹화
- 주로 SUM(), AVG(), COUNT(), MAX(), MIN() 등의 집계 함수와 결합되어 사용

## GROUP BY count

- SQL Code

```
SELECT productline,
count(productline)
FROM products
GROUP BY productline
;
```

- 결과

| products (7r × 2c) |                    |
|--------------------|--------------------|
| productline        | count(productline) |
| Classic Cars       | 38                 |
| Motorcycles        | 13                 |
| Planes             | 12                 |
| Ships              | 9                  |
| Trains             | 3                  |
| Trucks and Buses   | 11                 |
| Vintage Cars       | 24                 |

- Products 테이블에서 productLine 별 정보의 수

## GROUP BY avg

- SQL Code

```
SELECT productline,
avg(buyprice)
FROM products
GROUP BY productline
;
```

- 결과

| productline      | avg(buyprice) |
|------------------|---------------|
| Classic Cars     | 64.446316     |
| Motorcycles      | 50.685385     |
| Planes           | 49.629167     |
| Ships            | 47.007778     |
| Trains           | 43.923333     |
| Trucks and Buses | 56.329091     |
| Vintage Cars     | 46.06625      |

- Products 테이블에서 productLine 평균 가격

## GROUP BY sum

- SQL Code

```
SELECT productcode,
SUM(quantityordered)
FROM orderdetails
WHERE orderlinenumber = 1
group BY productcode
;
```

- 결과

| orderdetails (101r x 2c) |                      |
|--------------------------|----------------------|
| productcode              | SUM(quantityordered) |
| S10_1678                 | 254                  |
| S10_1949                 | 120                  |
| S10_2016                 | 106                  |
| S10_4698                 | 76                   |
| S10_4757                 | 51                   |
| S10_4962                 | 48                   |
| S12_1108                 | 33                   |
| S12_1666                 | 106                  |

- orderdetails 테이블에서 orderlinenumber 가 1인 제품들 중에서 productcode별 주문 수량의 합



## GROUP BY max

- SQL Code

```
SELECT productline,
MAX(msrp)
FROM products
GROUP BY productline
;
```

- 결과

| products (7r × 2c) |           |
|--------------------|-----------|
| productline        | MAX(msrp) |
| Classic Cars       | 214.3     |
| Motorcycles        | 193.66    |
| Planes             | 157.69    |
| Ships              | 122.89    |
| Trains             | 100.84    |
| Trucks and Buses   | 136.67    |
| Vintage Cars       | 170.0     |

- Products 테이블에서 productLine 별 최고 msrp(권장 판매 가격)

## GROUP BY 실습

- 문제 10: orders 테이블에서 각 상태(status)별로 주문 개수를 구하세요.

- 정답

| orders (6r × 2c) |            |
|------------------|------------|
| status           | OrderCount |
| Shipped          | 303        |
| Resolved         | 4          |
| Cancelled        | 6          |
| On Hold          | 4          |
| Disputed         | 3          |
| In Process       | 6          |

## GROUP BY 실습

- 문제 11: orderdetails 테이블에서 각 제품 코드(productCode)별로 주문된 총 수량(quantityOrdered)을 구하세요.

- 정답

| orderdetails (109r × 2c) |              |
|--------------------------|--------------|
| productCode              | TotalOrdered |
| S10_1678                 | 1,057        |
| S10_1949                 | 961          |
| S10_2016                 | 999          |
| S10_4698                 | 985          |
| S10_4757                 | 1,030        |
| S10_4962                 | 932          |
| S12_1099                 | 933          |
| S12_1108                 | 1,019        |

## GROUP BY 실습

- 문제 12: products 테이블에서 제품 라인(productLine)별 제품 개수를 조회하세요.

- 정답

| products (7r × 2c) |              |
|--------------------|--------------|
| productLine        | ProductCount |
| Classic Cars       | 38           |
| Motorcycles        | 13           |
| Planes             | 12           |
| Ships              | 9            |
| Trains             | 3            |
| Trucks and Buses   | 11           |
| Vintage Cars       | 24           |

## GROUP BY 실습

- 문제 13. "products" 테이블에서 각 제품 라인("productLine")별로 제품의 최대 가격("buyPrice")과 최소 가격("buyPrice")을 계산하세요.

- 정답

| products (7r x 3c) |          |          |
|--------------------|----------|----------|
| productLine        | maxPrice | minPrice |
| Classic Cars       | 103.42   | 15.91    |
| Motorcycles        | 91.02    | 24.14    |
| Planes             | 77.27    | 29.34    |
| Ships              | 82.34    | 33.3     |
| Trains             | 67.56    | 26.72    |
| Trucks and Buses   | 84.76    | 24.92    |
| Vintage Cars       | 86.7     | 20.61    |

## GROUP BY 실습

- 문제 14. "customers" 테이블에서 각 고객 도시("city")별로 평균 크레딧 한도("creditLimit") 상위 5개를 조회하세요.

- 정답

| customers (5r × 2c) |                    |
|---------------------|--------------------|
| city                | averageCreditLimit |
| San Rafael          | 210,500.0          |
| Genève              | 141,300.0          |
| Manchester          | 136,800.0          |
| Lyon                | 123,900.0          |
| Reggio Emilia       | 121,400.0          |

## GROUP BY 실습

- 문제 15.  
"orderdetails"  
테이블에서 주문  
번호("orderNumber")  
별로 총 주문 총액  
("priceEach" \*  
"quantityOrdered")  
상위 5개를  
계산하세요.

- 정답

| orderdetails (5r × 2c) |                 |
|------------------------|-----------------|
| orderNumber            | totalOrderPrice |
| 10,165                 | 67,392.85       |
| 10,287                 | 61,402.0        |
| 10,310                 | 61,234.67       |
| 10,212                 | 59,830.55       |
| 10,207                 | 59,265.14       |

## GROUP BY 실습

- 문제 16. "customers" 테이블에서 각 국가("country") 별로 고객 수가 많은 상위 5개를 조회하세요.

- 정답

| customers (5r × 2c) |                   |
|---------------------|-------------------|
| country             | numberOfCustomers |
| USA                 | 36                |
| Germany             | 13                |
| France              | 12                |
| Spain               | 7                 |
| Australia           | 5                 |



## GROUP BY 실습

- 문제 17. "products" 테이블에서 productScale이 '1:10'인 제품 라인("productLine")별로 제품의 평균 가격("buyPrice")을 계산하세요.

- 정답

| products (2r × 2c) |              |
|--------------------|--------------|
| productLine        | averagePrice |
| Classic Cars       | 95.893333    |
| Motorcycles        | 69.606667    |

# 데이터분석 기초 SQL 부트캠프

# 목차

8. GROUP BY 구문

HAVING

9. IF / CASE

10. JOIN

8.

# GROUP BY구문

HAVING

## HAVING 절

- **HAVING** 절은 SQL의 GROUP BY 절과 함께 사용되며 그룹화된 결과에 조건을 적용하는 데 사용
- **WHERE** 절은 개별 테이블에 대한 조건을 적용하는 반면, **HAVING** 절은 그룹화된 결과의 집계 값에 대한 조건을 적용

## HAVING

- SQL Code 처리 순서

```
SELECT
 컬럼
FROM
 테이블
WHERE
 조건
GROUP BY
 조건
HAVING
 조건
ORDER BY
 조건
```

- HAVING : 그룹화 조건 확인.
- 항상 GROUP BY 뒤에 위치하며 GROUP BY 이후 그룹화 된 테이블에 조건 적용

## HAVING

- SQL Code

```
SELECT productline,
count(productline)
FROM products
GROUP BY productline
HAVING COUNT(productline)
>20
;
```

- 결과

| productline  | count(productline) |
|--------------|--------------------|
| Classic Cars | 38                 |
| Vintage Cars | 24                 |

- Products 테이블에서  
productLine 별 정보의 수가 20  
이상

## HAVING

- SQL Code

```
SELECT productline,
avg(buyprice)
FROM products
GROUP BY productline
HAVING AVG(BUYPRICE) < 50
;
```

- 결과

| productline  | avg(buyprice) |
|--------------|---------------|
| Planes       | 49.629167     |
| Ships        | 47.007778     |
| Trains       | 43.923333     |
| Vintage Cars | 46.06625      |

- Products 테이블에서  
productLine 평균 가격이 \$50  
미만



## HAVING

- SQL Code

```
SELECT productcode,
SUM(quantityordered)
FROM orderdetails
WHERE orderlinenumber = 1
group BY productcode
HAVING PRODUCTCODE LIKE
'S10%'
;
```

- 결과

| orderdetails (6r x 2c) |                      |
|------------------------|----------------------|
| productcode            | SUM(quantityordered) |
| S10_1678               | 254                  |
| S10_1949               | 120                  |
| S10_2016               | 106                  |
| S10_4698               | 76                   |
| S10_4757               | 51                   |
| S10_4962               | 48                   |

- orderdetails 테이블에서 orderlinenumber  
가 1인 제품들 중에서 productcode가  
S10으로 시작하는 주문 수량의 합

## HAVING

- SQL Code

```
SELECT productline,
MAX(msrp)
FROM products
GROUP BY productline
HAVING productline =
'planes'
;
```

- 결과

| products (1r × 2c) |           |
|--------------------|-----------|
| productline        | MAX(msrp) |
| Planes             | 157.69    |

- Products 테이블에서  
productLine가 plane인 최고  
msrp(권장 판매 가격)

## HAVING 실습

- 문제 1: 'orders' 테이블에서 연도별 주문 건수가 100건을 초과하는 연도를 조회하세요.  
(orderDate와 orderNumber 컬럼 사용)

- 정답

| orders (2r × 2c) |             |
|------------------|-------------|
| OrderYear        | TotalOrders |
| 2,003            | 111         |
| 2,004            | 151         |

## HAVING 실습

- 문제 2: 'orderdetails' 테이블에서 상품별 연도별 총 주문량이 500개 이상인 상품 코드를 조회하세요. (productCode와 quantityOrdered 컬럼 사용)


- 정답

| productCode | TotalQuantity |
|-------------|---------------|
| S10_1678    | 1,057         |
| S10_1949    | 961           |
| S10_2016    | 999           |
| S10_4698    | 985           |
| S10_4757    | 1,030         |
| S10_4962    | 932           |
| S12_1099    | 933           |
| S12_1108    | 1,019         |
| S12_1666    | 972           |
| S12_2823    | 1,028         |

## HAVING 실습

- 문제 3: 'payments'  
테이블에서 고객별 총  
결제 금액이  
\$150,000을 초과하는  
고객 번호를  
조회하세요.  
(customerNumber와  
amount 컬럼 사용)

- 정답

| payments (6r × 2c) |                                                                                     |             |
|--------------------|-------------------------------------------------------------------------------------|-------------|
| customerNumber     |  | TotalAmount |
| 114                |                                                                                     | 180,585.07  |
| 124                |                                                                                     | 584,188.24  |
| 141                |                                                                                     | 715,738.98  |
| 148                |                                                                                     | 156,251.03  |
| 151                |                                                                                     | 177,913.95  |
| 323                |                                                                                     | 154,622.08  |

## HAVING 실습

- 문제 4: 'customers' 테이블에서 국가별 고객 수가 10명 이상인 국가를 조회하세요.  
(country와 customerNumber 컬럼 사용)

- 정답

| customers (3r × 2c) |                |
|---------------------|----------------|
| country             | TotalCustomers |
| France              | 12             |
| USA                 | 36             |
| Germany             | 13             |

9.

## IF / CASE구문

HAVING

## IF

- IF(condition, value\_if\_true, value\_if\_false)
- IF 문은 SQL의 GROUP BY 절과 함께 사용되며 그룹화된 결과에 조건을 적용하는 데 사용
- WHERE 절은 개별 테이블에 대한 조건을 적용하는 반면, HAVING 절은 그룹화된 결과의 집계 값에 대한 조건을 적용



## IF

- SQL Code

```
SELECT checkNumber,
amount, IF(amount > 50000,
'Large', 'Small') AS
orderSize
FROM payments
;
```

- 결과

| payments (273r × 3c) |           |           |
|----------------------|-----------|-----------|
| checkNumber          | amount    | orderSize |
| HQ336336             | 6,066.78  | Small     |
| JM555205             | 14,571.44 | Small     |
| OM314933             | 1,676.14  | Small     |
| B0864823             | 14,191.12 | Small     |
| HQ55022              | 32,641.98 | Small     |
| ND748579             | 33,347.88 | Small     |
| GG31455              | 45,864.03 | Small     |
| MA765515             | 82,261.22 | Large     |

- payments 테이블에서 amount가 50000 초과인 경우 large, 이하인 경우 small 로 출력

## IF 실습

- 문제 5. products' 테이블을 사용하여, 상품별로 가격이 \$100을 초과하면 'Expensive'로, 그렇지 않으면 'Cheap'으로 표시하는 쿼리를 작성하세요.
- (products.productName, products.buyPrice 컬럼 사용)

- 정답

| products (110r × 2c)                  |               |
|---------------------------------------|---------------|
| productName                           | PriceCategory |
| 1969 Harley Davidson Ultimate Chopper | Cheap         |
| 1952 Alpine Renault 1300              | Cheap         |
| 1996 Moto Guzzi 1100i                 | Cheap         |
| 2003 Harley-Davidson Eagle Drag Bike  | Cheap         |
| 1972 Alfa Romeo GTA                   | Cheap         |
| 1962 LanciaA Delta 16V                | Expensive     |
| 1968 Ford Mustang                     | Cheap         |
| 2001 Ferrari Enzo                     | Cheap         |

## CASE

- CASE 문은 여러 조건을 테스트하고 여러 결과 중 하나를 반환

```
CASE expression
WHEN value1 THEN result1
WHEN value2 THEN result2
...
ELSE result
END as ~
```

## CASE

- SQL Code

```
SELECT productName, buyPrice,
CASE
WHEN buyPrice < 20 THEN 'Cheap'
WHEN buyPrice BETWEEN 20 AND 50
THEN 'Moderate'
ELSE 'Expensive'
END AS priceCategory
FROM products;
```

- 결과

| productName                           | buyPrice | priceCategory |
|---------------------------------------|----------|---------------|
| 1969 Harley Davidson Ultimate Chopper | 48.81    | Moderate      |
| 1952 Alpine Renault 1300              | 98.58    | Expensive     |
| 1996 Moto Guzzi 1100i                 | 68.99    | Expensive     |
| 2003 Harley-Davidson Eagle Drag Bike  | 91.02    | Expensive     |
| 1972 Alfa Romeo GTA                   | 85.68    | Expensive     |
| 1962 LanciaA Delta 16V                | 103.42   | Expensive     |
| 1968 Ford Mustang                     | 95.34    | Expensive     |
| 2001 Ferrari Enzo                     | 95.59    | Expensive     |

- Buyprice 의 분류에 따라 cheap, moderate, expensive

## CASE 실습

- 문제 6. employees' 테이블을 사용하여, 각 직원의 직책(jobTitle)에 따라 다음과 같이 분류하세요:
- 'Sales Rep': 'Sales Team'
- 'VP Sales': 'Management'
- 'VP Marketing': 'Management'
- 그 외: 'Other Positions'

## 정답

```
SELECT firstName, lastName, jobTitle,
CASE jobTitle
WHEN 'Sales Rep' THEN 'Sales Team'
WHEN 'VP Sales' THEN 'Management'
WHEN 'VP Marketing' THEN 'Management'
ELSE 'Other Positions'
END AS PositionCategory
FROM employees;
```

## CASE 실습

- 문제 6. employees' 테이블을 사용하여, 각 직원의 직책(jobTitle)에 따라 다음과 같이 분류하세요:
- 'Sales Rep': 'Sales Team'
- 'VP Sales': 'Management'
- 'VP Marketing': 'Management'
- 'Management': 'Management'
- 그 외: 'Other Positions'

### 정답

| employees (23r x 4c) |           |                      |                  |
|----------------------|-----------|----------------------|------------------|
| firstName            | lastName  | jobTitle             | PositionCategory |
| Diane                | Murphy    | President            | Other Positions  |
| Mary                 | Patterson | VP Sales             | Management       |
| Jeff                 | Firrelli  | VP Marketing         | Management       |
| William              | Patterson | Sales Manager (APAC) | Other Positions  |
| Gerard               | Bondur    | Sale Manager (EMEA)  | Other Positions  |
| Anthony              | Bow       | Sales Manager (NA)   | Other Positions  |
| Leslie               | Jennings  | Sales Rep            | Sales Team       |
| Leslie               | Thompson  | Sales Rep            | Sales Team       |
| Julie                | Firrelli  | Sales Rep            | Sales Team       |

10.

## JOIN구문

INNER JOIN, LEFT JOIN, RIGHT JOIN, UNION

# JOIN

- JOIN 연산자는 두 테이블 간의 관계를 나타내기 위해 사용.
- 문법 :
- `SELECT * FROM TABLE A JOIN B ON 조건 ~`
- 예제) EX3 테이블과 EX4 테이블을 활용

test.ex3: 3 행 (총) (대략적)

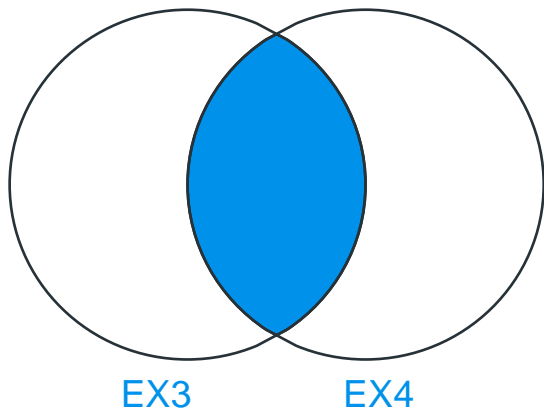
| id | NAME | age |
|----|------|-----|
| 1  | 이상훈  | 34  |
| 2  | 박상훈  | 30  |
| 3  | 최상훈  | 20  |

test.ex4: 3 행 (총) (

| id | region |
|----|--------|
| 1  | 서울     |
| 4  | 대구     |
| 5  | 부산     |



## INNER JOIN



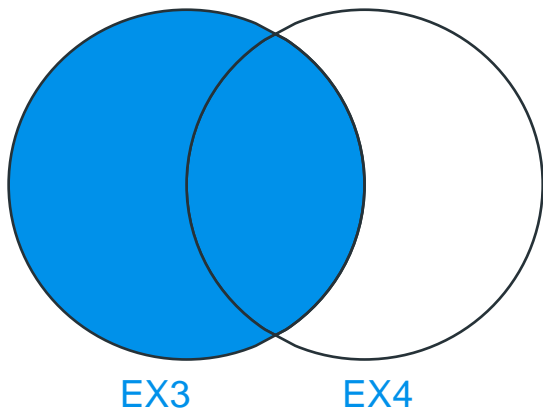
- SQL CODE

```
SELECT *
FROM ex3
JOIN ex4 ON ex3.id = ex4.id;
```

- 결과

| 결과 #1 (1r × 5c) |      |     |    |        |
|-----------------|------|-----|----|--------|
| id              | NAME | age | id | region |
| 1               | 이상훈  | 34  | 1  | 서울     |

## LEFT JOIN(1/2)



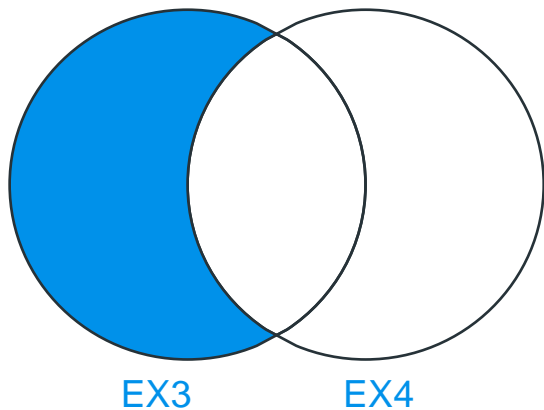
- SQL CODE

```
SELECT *
FROM ex3
LEFT JOIN ex4 ON ex3.id = ex4.id;
```

- 결과

| 결과 #1 (3r × 5c) |      |     |        |        |
|-----------------|------|-----|--------|--------|
| id              | NAME | age | id     | region |
| 1               | 이상훈  | 34  | 1      | 서울     |
| 2               | 박상훈  | 30  | (NULL) | (NULL) |
| 3               | 최상훈  | 20  | (NULL) | (NULL) |

## LEFT JOIN(2/2)



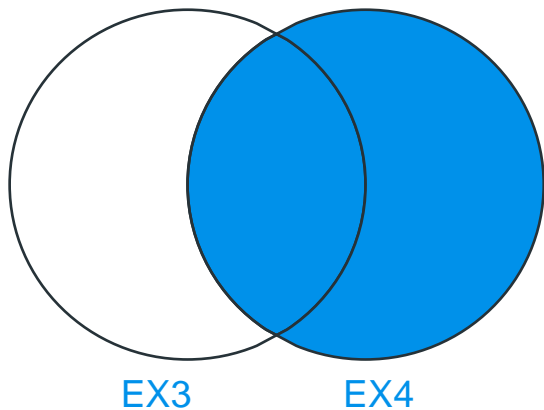
- SQL CODE

```
SELECT *
FROM ex3
left JOIN ex4 ON ex3.id = ex4.id
WHERE ex4.id IS null;
```

- 결과

| 결과 #1 (2r × 5c) |      |     |        |        |
|-----------------|------|-----|--------|--------|
| id              | NAME | age | id     | region |
| 2               | 박상훈  | 30  | (NULL) | (NULL) |
| 3               | 최상훈  | 20  | (NULL) | (NULL) |

## RIGHT JOIN(1/2)



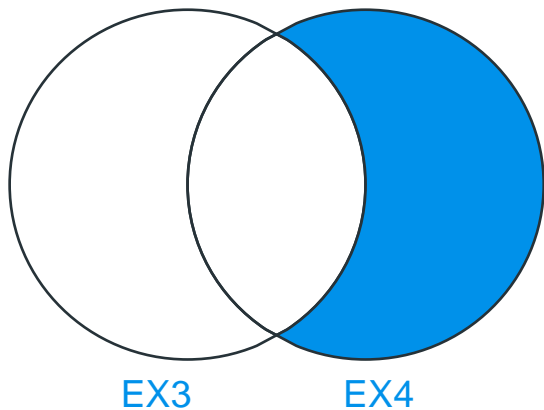
- SQL CODE

```
SELECT *
FROM ex3
RIGHT JOIN ex4 ON ex3.id = ex4.id;
```

- 결과

| 결과 #1 (3r × 5c) |        |        |    |        |
|-----------------|--------|--------|----|--------|
| id              | NAME   | age    | id | region |
| 1               | 이상훈    | 34     | 1  | 서울     |
| (NULL)          | (NULL) | (NULL) | 4  | 대구     |
| (NULL)          | (NULL) | (NULL) | 5  | 부산     |

## RIGHT JOIN(2/2)



- SQL CODE

```
SELECT *
FROM ex3
RIGHT JOIN ex4 ON ex3.id = ex4.id
WHERE ex3.id IS NULL;
```

- 결과

/ 결과 #1 (2r x 5c) \

| id     | NAME   | age    | id | region |
|--------|--------|--------|----|--------|
| (NULL) | (NULL) | (NULL) | 4  | 대구     |
| (NULL) | (NULL) | (NULL) | 5  | 부산     |

## UNION

- UNION 은 두 테이블의 데이터를 세로로 쭉 나열하는 역할.
- COLUMN의 수가 같아야 하며 중복은 제거.

- SQL CODE

```
SELECT id FROM ex3
UNION
SELECT id FROM ex4;
```

- 결과

| ex3 (5r × 1c) |  |
|---------------|--|
| id            |  |
| 1             |  |
| 2             |  |
| 3             |  |
| 4             |  |
| 5             |  |

## UNION ALL

- UNION 은 중복을 그대로 표시

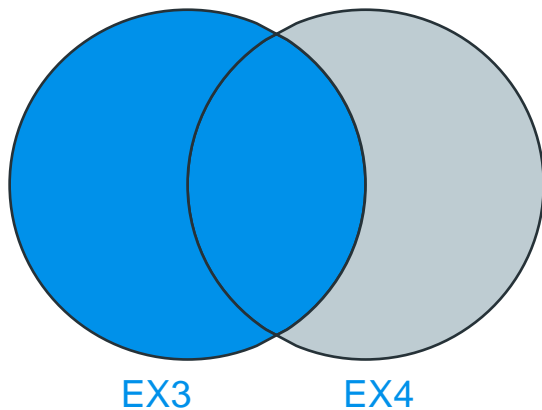
- SQL CODE

```
SELECT id FROM ex3
UNION ALL
SELECT id FROM ex4;
```

- 결과

| id |
|----|
| 1  |
| 2  |
| 3  |
| 1  |
| 4  |
| 5  |

## FULL OUTER JOIN



- SQL CODE

```
SELECT ex3.id, ex3.name, ex3.age, ex4.id, ex4.region
FROM ex3
left JOIN ex4 ON ex3.id = ex4.id
union
SELECT ex3.id, ex3.name, ex3.age, ex4.id, ex4.region
FROM ex3
RIGHT JOIN ex4 ON ex3.id = ex4.id
WHERE ex3.id IS NULL;
```

- 결과

| ex3 (5r × 5c) |        |        |        |        |
|---------------|--------|--------|--------|--------|
| id            | name   | age    | id     | region |
| 1             | 이상훈    | 34     | 1      | 서울     |
| 2             | 박상훈    | 30     | (NULL) | (NULL) |
| 3             | 최상훈    | 20     | (NULL) | (NULL) |
| (NULL)        | (NULL) | (NULL) | 4      | 대구     |
| (NULL)        | (NULL) | (NULL) | 5      | 부산     |



## JOIN 실습

- 문제 7. 'customers' 테이블과 'orders' 테이블을 사용하여, 모든 고객의 이름과 주문 번호를 조회하세요.

- 정답

| 결과 #1 (326r × 2c)          |               |
|----------------------------|---------------|
| customerName               | orderNumber 🔑 |
| Atelier graphique          | 10,123        |
| Atelier graphique          | 10,298        |
| Atelier graphique          | 10,345        |
| Signal Gift Stores         | 10,124        |
| Signal Gift Stores         | 10,278        |
| Signal Gift Stores         | 10,346        |
| Australian Collectors, Co. | 10,120        |
| Australian Collectors, Co. | 10,125        |
| Australian Collectors, Co. | 10,223        |
| Australian Collectors, Co. | 10,342        |
| Australian Collectors, Co. | 10,347        |

## JOIN 실습

- 문제 8. 'products' 테이블과 'orderdetails' 테이블을 사용하여, 상품 이름과 주문된 수량을 조회하세요.

- 정답

| 결과 #1 (2,996r × 2c)                   |                 |
|---------------------------------------|-----------------|
| productName                           | quantityOrdered |
| 1969 Harley Davidson Ultimate Chopper | 30              |
| 1969 Harley Davidson Ultimate Chopper | 34              |
| 1969 Harley Davidson Ultimate Chopper | 41              |
| 1969 Harley Davidson Ultimate Chopper | 45              |
| 1969 Harley Davidson Ultimate Chopper | 49              |
| 1969 Harley Davidson Ultimate Chopper | 36              |
| 1969 Harley Davidson Ultimate Chopper | 29              |
| 1969 Harley Davidson Ultimate Chopper | 48              |

## JOIN 실습

- 문제 9. 'Leslie'이라는 이름을 가진 직원이 담당하는 모든 고객의 이름을 조회하세요.
- 힌트. 'employees' 테이블과 'customers' 테이블을 사용

- 정답

| customers (12r × 1c)         |  |
|------------------------------|--|
| customerName                 |  |
| Mini Gifts Distributors Ltd. |  |
| Mini Wheels Co.              |  |
| Technics Stores Inc.         |  |
| Corporate Gift Ideas Co.     |  |
| The Sharp Gifts Warehouse    |  |
| Signal Collectibles Ltd.     |  |
| Signal Gift Stores           |  |
| Toys4GrownUps.com            |  |
| Boards & Toys Co.            |  |
| Collectable Mini Designs Co. |  |
| Men 'R' US Retailers, Ltd.   |  |
| West Coast Collectables Co.  |  |

## JOIN 실습

- 문제 10.
- San Francisco 사무실에서 근무하는 모든 직원의 이름을 조회하세요.
- 힌트. 'employees' 테이블과 'offices' 테이블을 사용

- 정답

| employees (6r × 2c) |           |
|---------------------|-----------|
| firstName           | lastName  |
| Diane               | Murphy    |
| Mary                | Patterson |
| Jeff                | Firrelli  |
| Anthony             | Bow       |
| Leslie              | Jennings  |
| Leslie              | Thompson  |

## JOIN 실습

- 문제 11. 주문 가격이 상품의 구매 가격보다 2.5배 높은 상품의 이름, 코드, 판매가격, 주문가격, 주문 수량을 조회하세요.
- 힌트. 'orderdetails' 테이블과 'products' 테이블을 사용


### 정답

| 결과 #1 (3r × 5c) |                       |           |          |                 |
|-----------------|-----------------------|-----------|----------|-----------------|
| productcode     | productName           | priceEach | buyPrice | quantityOrdered |
| 524_4620        | 1961 Chevrolet Impala | 80.84     | 32.33    | 23              |
| 524_4620        | 1961 Chevrolet Impala | 80.84     | 32.33    | 22              |
| 524_4620        | 1961 Chevrolet Impala | 80.84     | 32.33    | 41              |

## JOIN 실습

- 문제 12. 2003년에 주문한 고객의 이름과 주문 번호를 조회하세요.
- 힌트. 'customers' 테이블과 'orders' 테이블을 JOIN

- 정답

| 결과 #1 (111r x 2c)            |                                                                                                 |
|------------------------------|-------------------------------------------------------------------------------------------------|
| customerName                 | orderNumber  |
| Atelier graphique            | 10,123                                                                                          |
| Signal Gift Stores           | 10,124                                                                                          |
| Australian Collectors, Co.   | 10,120                                                                                          |
| Australian Collectors, Co.   | 10,125                                                                                          |
| Baane Mini Imports           | 10,103                                                                                          |
| Baane Mini Imports           | 10,158                                                                                          |
| Mini Gifts Distributors Ltd. | 10,113                                                                                          |
| Mini Gifts Distributors Ltd. | 10,135                                                                                          |
| Mini Gifts Distributors Ltd. | 10,142                                                                                          |

## JOIN 실습

- 문제 13. 2004년에  
결제한 고객의 이름과  
결제 금액을  
조회하세요.
- 힌트. 'customers'  
테이블과 'payments'  
테이블을 JOIN

### 정답

| 결과 #1 (136r x 2c)          |           |
|----------------------------|-----------|
| customerName               | amount    |
| Atelier graphique          | 6,066.78  |
| Atelier graphique          | 1,676.14  |
| Signal Gift Stores         | 14,191.12 |
| Signal Gift Stores         | 33,347.88 |
| Australian Collectors, Co. | 82,261.22 |
| Australian Collectors, Co. | 44,894.74 |
| La Rochelle Gifts          | 19,501.82 |
| La Rochelle Gifts          | 47,924.19 |
| Baane Mini Imports         | 17,876.32 |

## JOIN 실습

- 문제 14. 각 직원별로 담당한 고객의 수를 조회하세요.
- 힌트. 'employees' 테이블과 'customers' 테이블을 JOIN

### 정답

| EmployeeName     | NumberOfCustomers |
|------------------|-------------------|
| Leslie Jennings  | 6                 |
| Leslie Thompson  | 6                 |
| Julie Firrelli   | 6                 |
| Steve Patterson  | 6                 |
| Foon Yue Tseng   | 7                 |
| George Vanauf    | 8                 |
| Loui Bondur      | 6                 |
| Gerard Hernandez | 7                 |
| Pamela Castillo  | 10                |
| Larry Bott       | 8                 |
| Barry Jones      | 9                 |
| Andy Fixter      | 5                 |
| Peter Marsh      | 5                 |
| Mami Nishi       | 5                 |
| Martin Gerard    | 6                 |



## JOIN 실습

- 문제 15. 2003년에 주문된 상품 이름과 해당 주문의 수량을 조회하세요.
- 힌트. 'orders' 테이블, 'orderdetails' 테이블, 'products' 테이블을 JOIN

- 정답

| 결과 #1 (1,052r x 2c)                       |                 |
|-------------------------------------------|-----------------|
| productName                               | quantityOrdered |
| 1917 Grand Touring Sedan                  | 30              |
| 1911 Ford Town Car                        | 50              |
| 1932 Alfa Romeo 8C2300 Spider Sport       | 22              |
| 1936 Mercedes Benz 500k Roadster          | 49              |
| 1932 Model A Ford J-Coupe                 | 25              |
| 1928 Mercedes-Benz SSK                    | 26              |
| 1939 Chevrolet Deluxe Coupe               | 45              |
| 1938 Cadillac V-16 Presidential Limousine | 46              |
| 1937 Lincoln Berline                      | 39              |

## JOIN 실습

- 문제 16. 각 고객별 총 주문 금액을 조회하세요.

- 정답

| customerName                 | TotalOrderValue |
|------------------------------|-----------------|
| Atelier graphique            | 22,314.36       |
| Signal Gift Stores           | 80,180.98       |
| Australian Collectors, Co.   | 180,585.07      |
| La Rochelle Gifts            | 158,573.12      |
| Baane Mini Imports           | 104,224.79      |
| Mini Gifts Distributors Ltd. | 591,827.34      |
| Blauer See Auto, Co.         | 75,937.76       |
| Mini Wheels Co.              | 66,710.56       |
| Land of Toys Inc.            | 149,085.15      |
| Euro+ Shopping Channel       | 820,689.54      |

- 힌트. 'customers' 테이블, 'orders' 테이블, 'orderdetails' 테이블을 JOIN

## JOIN 실습

- 문제 17. 각 직원별로  
담당한 고객들의 총  
결제 금액을  
조회하세요.
- 힌트. 'employees'  
테이블, 'customers'  
테이블, 'payments'  
테이블을 JOIN

- 정답

| employees (15r x 2c) |               |
|----------------------|---------------|
| EmployeeName         | TotalPayments |
| Leslie Jennings      | 989,906.55    |
| Leslie Thompson      | 347,533.03    |
| Julie Firrelli       | 386,663.2     |
| Steve Patterson      | 449,219.13    |
| Foon Yue Tseng       | 488,212.67    |
| George Vanauf        | 584,406.8     |
| Loui Bondur          | 569,485.75    |
| Gerard Hernandez     | 1,112,003.81  |
| Pamela Castillo      | 750,201.87    |
| Larry Bott           | 686,653.25    |

## JOIN 실습

- 문제 18. 각 상품 라인별로 주문된 상품의 총 수량을 조회하세요.
- 힌트. 'products' 테이블, 'orderdetails' 테이블, 'productlines' 테이블을 JOIN

- 정답

| productlines (7r × 2c) |               |
|------------------------|---------------|
| productLine            | TotalQuantity |
| Classic Cars           | 35,582        |
| Motorcycles            | 12,778        |
| Planes                 | 11,872        |
| Ships                  | 8,532         |
| Trains                 | 2,818         |
| Trucks and Buses       | 11,001        |
| Vintage Cars           | 22,933        |

## JOIN 실습

- 문제 19. 2004년에 가장 많이 판매된 상위 5개 상품 이름과 해당 상품의 총 판매 수량을 조회하세요.
- 힌트. 'orders' 테이블, 'orderdetails' 테이블, 'products' 테이블을 JOIN

- 정답

| products (5r × 2c)                      |               |
|-----------------------------------------|---------------|
| productName                             | TotalQuantity |
| 1992 Ferrari 360 Spider red             | 789           |
| 1980s Black Hawk Helicopter             | 567           |
| 2001 Ferrari Enzo                       | 566           |
| The USS Constitution Ship               | 541           |
| 1941 Chevrolet Special Deluxe Cabriolet | 536           |

# 11.

## WINDOW 함수

SUM, AVG, MIN, MAX, COUNT  
ROW\_NUMBER, RANK, DENSE\_RANK, LAG, LEAD,  
FIRST\_VALUE, LAST\_VALUE  
OVER (PARTITION BY ~ ORDER BY ~)

## WINDOW 함수

- **SELECT** 구문에서 사용되며 분석 구간을 변동시키는 역할
- EX)누적합
- **SUM(COLUMN1) OVER(PARTITION BY COLUMN2  
ORDER BY COLUMN3) AS NEW\_COLUMN**

## 누적합

- `SUM(COLUMN1) OVER(PARTITION BY COLUMN2 ORDER BY COLUMN3) AS NEW_COLUMN`
- `PARTITION BY` : `GROUP BY` 와 비슷한 역할로, 그룹별로 누적 합계를 구할 수 있다. `GROUP BY` 는 집계 결과로 조회가 되는 반면 `PARTITION BY` 는 본래의 TABLE 그대로 출력.
- 생략 가능



## 누적합

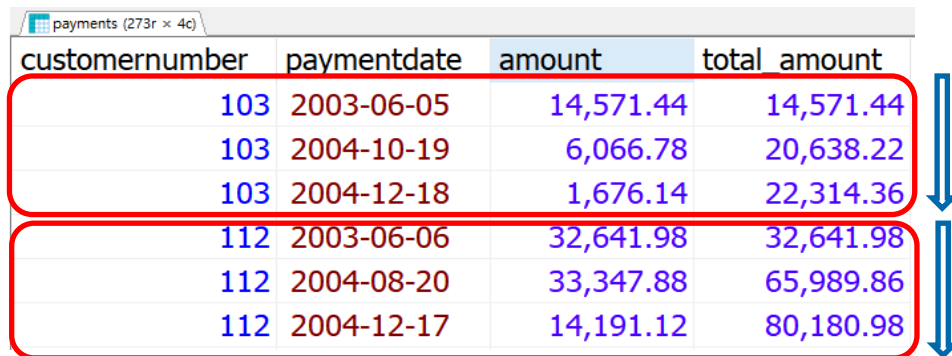
- `SUM(COLUMN1) OVER(PARTITION BY COLUMN2  
ORDER BY COLUMN3) AS NEW_COLUMN`
- `ORDER BY` : 계산을 하는 순서를 정해준다.
- 생략 하면 누적 합이 계산되지 않는다.

## 누적합(over partition by order by 사용)

- SQL CODE

```
SELECT customernumber, paymentdate, amount,
sum(amount) OVER(PARTITION BY CUSTOMERNUMBER
ORDER BY PAYMENTDATE) AS total_amount
FROM payments;
```

- 결과



| customernumber | paymentdate | amount    | total amount |
|----------------|-------------|-----------|--------------|
| 103            | 2003-06-05  | 14,571.44 | 14,571.44    |
| 103            | 2004-10-19  | 6,066.78  | 20,638.22    |
| 103            | 2004-12-18  | 1,676.14  | 22,314.36    |
| 112            | 2003-06-06  | 32,641.98 | 32,641.98    |
| 112            | 2004-08-20  | 33,347.88 | 65,989.86    |
| 112            | 2004-12-17  | 14,191.12 | 80,180.98    |

## 누적합(over partition by 사용)

- SQL CODE

```
SELECT customernumber, paymentdate, amount,
sum(amount) OVER(PARTITION BY
CUSTOMERNUMBER) AS total_amount
FROM payments;
```

- 결과(order by 생략으로 누적 합계 순서 x)

payments (273r x 4c)

| customernumber | paymentdate | amount    | total amount |
|----------------|-------------|-----------|--------------|
| 103            | 2004-10-19  | 6,066.78  | 22,314.36    |
| 103            | 2003-06-05  | 14,571.44 | 22,314.36    |
| 103            | 2004-12-18  | 1,676.14  | 22,314.36    |
| 112            | 2004-12-17  | 14,191.12 | 80,180.98    |
| 112            | 2003-06-06  | 32,641.98 | 80,180.98    |
| 112            | 2004-08-20  | 33,347.88 | 80,180.98    |

## 누적합(over order by 사용)

- SQL CODE

```
SELECT customernumber, paymentdate, amount,
sum(amount) OVER(ORDER BY PAYMENTDATE) AS
total_amount
FROM payments;
```

- 결과(partition by 생략으로 그룹화 사라짐)

payments (273r x 4c)

| customernumber | paymentdate | amount    | total_amount |
|----------------|-------------|-----------|--------------|
| 363            | 2003-01-16  | 10,223.83 | 10,223.83    |
| 128            | 2003-01-28  | 10,549.01 | 20,772.84    |
| 181            | 2003-01-30  | 5,494.78  | 26,267.62    |
| 121            | 2003-02-16  | 50,218.95 | 76,486.57    |
| 145            | 2003-02-20  | 53,959.21 | 130,445.78   |
| 141            | 2003-02-25  | 40,206.2  | 170,651.98   |
| 278            | 2003-03-02  | 52,151.81 | 222,803.79   |

## 누적합(over partition by 생략 order by 생략)

- SQL CODE

```
SELECT customernumber, paymentdate, amount,
sum(amount) OVER() AS total_amount
FROM payments;
```

- 결과(partition by, order by 생략으로 전체 sum 출력)

| payments (273r x 4c) |             |           |              |
|----------------------|-------------|-----------|--------------|
| customernumber       | paymentdate | amount    | total_amount |
| 103                  | 2004-10-19  | 6,066.78  | 8,853,839.23 |
| 103                  | 2003-06-05  | 14,571.44 | 8,853,839.23 |
| 103                  | 2004-12-18  | 1,676.14  | 8,853,839.23 |
| 112                  | 2004-12-17  | 14,191.12 | 8,853,839.23 |
| 112                  | 2003-06-06  | 32,641.98 | 8,853,839.23 |
| 112                  | 2004-08-20  | 33,347.88 | 8,853,839.23 |

## over(partition by ~ order by ~ ) 실습

- 문제 20. 문제:  
"orderdetails"  
테이블에서 각  
주문별로 주문된  
제품의 평균  
수량("quantityOrdered")을 계산하세요..

- 정답

| orderNumber | productCode | avg_quantity_per_order |
|-------------|-------------|------------------------|
| 10,100      | S18_1749    | 37.75                  |
| 10,100      | S18_2248    | 37.75                  |
| 10,100      | S18_4409    | 37.75                  |
| 10,100      | S24_3969    | 37.75                  |
| 10,101      | S18_2325    | 35.5                   |
| 10,101      | S18_2795    | 35.5                   |
| 10,101      | S24_1937    | 35.5                   |
| 10,101      | S24_2022    | 35.5                   |

## over(partition by ~ order by ~ ) 실습

- 문제21: "orders"  
테이블에서 각  
고객별로 주문 날짜에  
따라서 지금까지의  
주문 횟수를  
계산하세요.

### 정답

| customerNumber | orderNumber | orderDate  | order_count_so_far |
|----------------|-------------|------------|--------------------|
| 103            | 10,123      | 2003-05-20 | 1                  |
| 103            | 10,298      | 2004-09-27 | 2                  |
| 103            | 10,345      | 2004-11-25 | 3                  |
| 112            | 10,124      | 2003-05-21 | 1                  |
| 112            | 10,278      | 2004-08-06 | 2                  |
| 112            | 10,346      | 2004-11-29 | 3                  |
| 114            | 10,120      | 2003-04-29 | 1                  |
| 114            | 10,125      | 2003-05-21 | 2                  |
| 114            | 10,223      | 2004-02-20 | 3                  |
| 114            | 10,342      | 2004-11-24 | 4                  |
| 114            | 10,347      | 2004-11-29 | 5                  |

## WINDOW 함수

- LEAD / LAG 함수
- LEAD : 다음 행 데이터를 가져온다.
- LAG : 이전 행 데이터를 가져온다.



## LEAD/LAG(over partition by order by 사용)

- SQL CODE

```
SELECT orderNumber, customerNumber, orderDate,
LAG(orderDate) OVER (PARTITION BY customerNumber ORDER BY orderDate)
AS prev_order_date,
LEAD(orderDate) OVER (PARTITION BY customerNumber ORDER BY
orderDate) AS next_order_date
FROM orders;
```

- 결과

orders (326r × 5c)

| orderNumber | customerNumber | orderDate  | prev_order_date | next_order_date |
|-------------|----------------|------------|-----------------|-----------------|
| 10,123      | 103            | 2003-05-20 | (NULL)          | 2004-09-27      |
| 10,298      | 103            | 2004-09-27 | 2003-05-20      | 2004-11-25      |
| 10,345      | 103            | 2004-11-25 | 2004-09-27      | (NULL)          |
| 10,124      | 112            | 2003-05-21 | (NULL)          | 2004-08-06      |
| 10,278      | 112            | 2004-08-06 | 2003-05-21      | 2004-11-29      |
| 10,346      | 112            | 2004-11-29 | 2004-08-06      | (NULL)          |

## LEAD/LAG(over order by 사용)

- SQL CODE

```
SELECT orderNumber, customerNumber, orderDate,
LAG(orderDate) OVER (ORDER BY orderDate) AS prev_order_date,
LEAD(orderDate) OVER (ORDER BY orderDate) AS next_order_date
FROM orders;
```

- 결과

| orders (326r × 5c) |                |            |                 |                 |
|--------------------|----------------|------------|-----------------|-----------------|
| orderNumber        | customerNumber | orderDate  | prev_order_date | next_order_date |
| 10,100             | 363            | 2003-01-06 | (NULL)          | 2003-01-09      |
| 10,101             | 128            | 2003-01-09 | 2003-01-06      | 2003-01-10      |
| 10,102             | 181            | 2003-01-10 | 2003-01-09      | 2003-01-29      |
| 10,103             | 121            | 2003-01-29 | 2003-01-10      | 2003-01-31      |
| 10,104             | 141            | 2003-01-31 | 2003-01-29      | 2003-02-11      |
| 10,105             | 145            | 2003-02-11 | 2003-01-31      | 2003-02-17      |
| 10,106             | 278            | 2003-02-17 | 2003-02-11      | 2003-02-24      |

## LEAD/LAG(over partition by 사용)

- SQL CODE

```
SELECT orderNumber, customerNumber, orderDate,
LAG(orderDate) OVER (PARTITION BY customerNumber) AS
prev_order_date,
LEAD(orderDate) OVER (PARTITION BY customerNumber) AS
next_order_date
FROM orders;
```

- 결과

orders (326r × 5c)

| orderNumber | customerNumber | orderDate  | prev_order_date | next_order_date |
|-------------|----------------|------------|-----------------|-----------------|
| 10,123      | 103            | 2003-05-20 | (NULL)          | 2004-09-27      |
| 10,298      | 103            | 2004-09-27 | 2003-05-20      | 2004-11-25      |
| 10,345      | 103            | 2004-11-25 | 2004-09-27      | (NULL)          |
| 10,124      | 112            | 2003-05-21 | (NULL)          | 2004-08-06      |
| 10,278      | 112            | 2004-08-06 | 2003-05-21      | 2004-11-29      |
| 10,346      | 112            | 2004-11-29 | 2004-08-06      | (NULL)          |

## LEAD/LAG(over order by 사용)

- SQL CODE

```
SELECT orderNumber, customerNumber, orderDate,
LAG(orderDate) OVER () AS prev_order_date,
LEAD(orderDate) OVER () AS next_order_date
FROM orders;
```

- 결과

| orders (326r × 5c) |                |            |                 |                 |
|--------------------|----------------|------------|-----------------|-----------------|
| orderNumber        | customerNumber | orderDate  | prev_order_date | next_order_date |
| 10,100             | 363            | 2003-01-06 | (NULL)          | 2003-01-09      |
| 10,101             | 128            | 2003-01-09 | 2003-01-06      | 2003-01-10      |
| 10,102             | 181            | 2003-01-10 | 2003-01-09      | 2003-01-29      |
| 10,103             | 121            | 2003-01-29 | 2003-01-10      | 2003-01-31      |
| 10,104             | 141            | 2003-01-31 | 2003-01-29      | 2003-02-11      |
| 10,105             | 145            | 2003-02-11 | 2003-01-31      | 2003-02-17      |
| 10,106             | 278            | 2003-02-17 | 2003-02-11      | 2003-02-24      |

## over(partition by ~ order by ~ ) 실습

- 문제 22 :  
"orderdetails"  
테이블에서 각 제품  
코드별로 주문된  
수량(ORDERNUMBE  
R)을 기준으로  
정렬했을 때, 주문  
수량의 증분을  
계산하시오.

### 정답

| orderdetails (2,996r × 4c) |             |                 |                     |
|----------------------------|-------------|-----------------|---------------------|
| orderNumber                | productCode | quantityOrdered | quantity_difference |
| 10,107                     | S10_1678    | 30              | (NULL)              |
| 10,121                     | S10_1678    | 34              | 4                   |
| 10,134                     | S10_1678    | 41              | 7                   |
| 10,145                     | S10_1678    | 45              | 4                   |
| 10,159                     | S10_1678    | 49              | 4                   |
| 10,168                     | S10_1678    | 36              | -13                 |
| 10,180                     | S10_1678    | 29              | -7                  |
| 10,188                     | S10_1678    | 48              | 19                  |

## 순위 함수

- **ROW\_NUMBER** : 중복 없이 고유한 순위 부여
- **RANK** : 중복값에 같은 순위 부여, 중복된 숫자만큼 건너뛰기(1,1,1,4,5,6)
- **DENSE\_RANK** : RANK와 유사하지만 중복된 숫자를 건너뛰지 않음(1,1,1,2,3,4)

## 순위 함수

- SQL CODE

```
SELECT customername, creditlimit,
ROW_NUMBER() OVER (ORDER BY creditlimit ASC) AS row_number_,
RANK() OVER (ORDER BY creditlimit ASC) AS rank_,
DENSE_RANK() OVER (ORDER BY creditlimit ASC) AS dense_rank_
FROM customers
ORDER BY creditlimit ASC;
```

## 순위 함수

### 결과

| customername                 | creditlimit | row_number_ | rank_ | dense_rank_ |
|------------------------------|-------------|-------------|-------|-------------|
| Euro+ Shopping Channel       | 227,600.0   | 1           | 1     | 1           |
| Mini Gifts Distributors Ltd. | 210,500.0   | 2           | 2     | 2           |
| Vida Sport, Ltd              | 141,300.0   | 3           | 3     | 3           |
| Muscle Machine Inc           | 138,500.0   | 4           | 4     | 4           |
| AV Stores, Co.               | 136,800.0   | 5           | 5     | 5           |
| Saveley & Henriot, Co.       | 123,900.0   | 6           | 6     | 6           |
| Marta's Replicas Co.         | 123,700.0   | 7           | 7     | 7           |
| L'ordine Souvenirs           | 121,400.0   | 8           | 8     | 8           |
| Heintze Collectables         | 120,800.0   | 9           | 9     | 9           |
| Toms Spezialitäten, Ltd      | 120,400.0   | 10          | 10    | 10          |
| Rovelli Gifts                | 119,600.0   | 11          | 11    | 11          |
| La Rochelle Gifts            | 118,200.0   | 12          | 12    | 12          |
| Australian Collectors, Co.   | 117,300.0   | 13          | 13    | 13          |
| Scandinavian Gift Ideas      | 116,400.0   | 14          | 14    | 14          |
| Land of Toys Inc.            | 114,900.0   | 15          | 15    | 15          |
| Online Diecast Creations Co. | 114,200.0   | 16          | 16    | 16          |
| Amica Models & Co.           | 113,000.0   | 17          | 17    | 17          |
| Kelly's Gift Shop            | 110,000.0   | 18          | 18    | 18          |
| Anna's Decorations, Ltd      | 107,800.0   | 19          | 19    | 19          |
| Collectable Mini Designs Co. | 105,000.0   | 20          | 20    | 20          |
| Corporate Gift Ideas Co.     | 105,000.0   | 21          | 20    | 20          |
| Corrida Auto Replicas, Ltd   | 104,600.0   | 22          | 22    | 21          |



## 순위 함수

- **First\_value()**: 가장 첫번째 오는 row 조회
- **Last\_value()**: 가장 마지막에 오는 row 조회
- 위 두 함수는 order by의 활용에 따라 결과가 달라짐
- Ex) first\_value(column) over ( partition by ~ order by ~)

## over(partition by ~ order by ~) 실습

- 문제 23 : "products"  
테이블에서 각 제품  
라인별로 가장 비싼  
제품의 이름과 가장  
싼 제품의 이름을  
조회하세요.

### 정답

| productLine  | productName                   | buyprice | cheapest_product               | most_expensive_product |
|--------------|-------------------------------|----------|--------------------------------|------------------------|
| Classic Cars | 1962 LanciaA Delta 16V        | 103.42   | 1958 Chevy Corvette Limited... | 1962 LanciaA Delta 16V |
| Classic Cars | 1998 Chrysler Plymouth Pro... | 101.51   | 1958 Chevy Corvette Limited... | 1962 LanciaA Delta 16V |
| Classic Cars | 1952 Alpine Renault 1300      | 98.58    | 1958 Chevy Corvette Limited... | 1962 LanciaA Delta 16V |

## WINDOW FRAME(윈도우 프레임)

- **ROW:** 행의 개수로 윈도우 프레임을 정의
- **RANGE:** 정렬의 기준이 되는 행의 값을 기준으로 정의
- **PRECEDING:** 현재 행보다 전에 있는 행들을 의미
- **FOLLOWING:** 현재 행보다 다음에 있는 행들을 의미
- **UNBOUNDED PRECEDING:** 현재 파티션의 첫 번째 행부터 현재 행까지의 범위를 의미
- **UNBOUNDED FOLLOWING:** 현재 행부터 현재 파티션의 마지막 행까지의 범위를 의미
- **CURRENT ROW:** 현재 행

## WINDOW FRAME(윈도우 프레임)

- SQL CODE

```
SELECT orderNumber, productCode, quantityOrdered,
AVG(quantityOrdered) OVER (ORDER BY orderNumber ROWS BETWEEN 1
PRECEDING AND 1 FOLLOWING) AS moving_avg_quantity_1,
AVG(quantityOrdered) OVER (ORDER BY orderNumber ROWS BETWEEN
CURRENT ROW AND 1 FOLLOWING) AS moving_avg_quantity_2,
AVG(quantityOrdered) OVER (ORDER BY orderNumber ROWS BETWEEN 1
PRECEDING AND CURRENT ROW) AS moving_avg_quantity_3,
AVG(quantityOrdered) OVER (ORDER BY orderNumber RANGE BETWEEN 1
PRECEDING AND 1 FOLLOWING) AS moving_avg_quantity_4
FROM orderdetails
;
```

## WINDOW FRAME(윈도우 프레임)

- SQL CODE

| orderNumber | productCode | quantityOrdered | moving_avg_quantity_1 | moving_avg_quantity_2 | moving_avg_quantity_3 | moving_avg_quantity_4 |
|-------------|-------------|-----------------|-----------------------|-----------------------|-----------------------|-----------------------|
| 10,100      | S18_1749    | 30              | 40.0                  | 40.0                  | 30.0                  | 36.625                |
| 10,100      | S18_2248    | 50              | 34.0                  | 36.0                  | 40.0                  | 36.625                |
| 10,100      | S18_4409    | 22              | 40.3333               | 35.5                  | 36.0                  | 36.625                |
| 10,100      | S24_3969    | 49              | 32.0                  | 37.0                  | 35.5                  | 36.625                |
| 10,101      | S18_2325    | 25              | 33.3333               | 25.5                  | 37.0                  | 37.3                  |
| 10,101      | S18_2795    | 26              | 32.0                  | 35.5                  | 25.5                  | 37.3                  |
| 10,101      | S24_1937    | 45              | 39.0                  | 45.5                  | 35.5                  | 37.3                  |
| 10,101      | S24_2022    | 46              | 43.3333               | 42.5                  | 45.5                  | 37.3                  |
| 10,102      | S18_1342    | 39              | 42.0                  | 40.0                  | 42.5                  | 34.6818               |
| 10,102      | S18_1367    | 41              | 35.3333               | 33.5                  | 40.0                  | 34.6818               |
| 10,103      | S10_1949    | 26              | 36.3333               | 34.0                  | 33.5                  | 34.3226               |

## over(partition by ~ order by ~ ) 실습

- 문제 24 : 직원별로 담당하는 고객 수를 계산하고, 각 직원별 담당 고객 수의 누적 합계를 계산하세요.

### 정답

| employeeNumber | firstName | lastName  | customerCount | cumulativeCustomerCount |
|----------------|-----------|-----------|---------------|-------------------------|
| 1,165          | Leslie    | Jennings  | 6             | 6                       |
| 1,166          | Leslie    | Thompson  | 6             | 12                      |
| 1,188          | Julie     | Firrelli  | 6             | 18                      |
| 1,216          | Steve     | Patterson | 6             | 24                      |
| 1,286          | Foon Yue  | Tseng     | 7             | 31                      |
| 1,323          | George    | Vanauf    | 8             | 39                      |
| 1,337          | Loui      | Bondur    | 6             | 45                      |
| 1,370          | Gerard    | Hernandez | 7             | 52                      |
| 1,401          | Pamela    | Castillo  | 10            | 62                      |
| 1,501          | Larry     | Bott      | 8             | 70                      |
| 1,504          | Barry     | Jones     | 9             | 79                      |
| 1,611          | Andy      | Fixter    | 5             | 84                      |
| 1,612          | Peter     | Marsh     | 5             | 89                      |
| 1,621          | Mami      | Nishi     | 5             | 94                      |
| 1,702          | Martin    | Gerard    | 6             | 100                     |

# 데이터분석 기초 SQL 부트캠프

# 목차

## 12. 서브쿼리

**scalar,row,column,table,exists,**

**correlated**

selec절, from절, where절

## 13. With 구문



# 12.

## 서브쿼리(Sub-query)

Select절, from절, where절

## Scalar Subquery

- 스칼라 서브쿼리는 단일값을 반환
- 일반적으로 select, where, having 절에서 사용
- 모든 서브쿼리는 반드시 괄호 '()' 안에 포함.

## Scalar Subquery

- SQL Code

```
SELECT productName,
buyPrice
FROM products
WHERE buyPrice > (SELECT
AVG(buyPrice) FROM
products);
```

- 결과

| products (54r x 2c)            |          |
|--------------------------------|----------|
| productName                    | buyPrice |
| 1952 Alpine Renault 1300       | 98.58    |
| 1996 Moto Guzzi 1100i          | 68.99    |
| 2003 Harley-Davidson Eagle ... | 91.02    |
| 1972 Alfa Romeo GTA            | 85.68    |
| 1962 LanciaA Delta 16V         | 103.42   |
| 1968 Ford Mustang              | 95.34    |
| 2001 Ferrari Enzo              | 95.59    |

- 평균 가격(buyprice)보다 비싼 상품을 조회
- 서브 쿼리의 결과가 단일 값

## Table Subquery

- 테이블 서브쿼리는 테이블처럼 사용할 수 있는 행과 열을 반환
- 일반적으로 from 절에서 사용.
- From 절에 사용되는 서브쿼리는 별칭 필수

## Table Subquery

- SQL Code

```
SELECT customerNumber,
order_count
FROM (SELECT
customerNumber,
COUNT(orderNumber) AS
order_count
FROM orders GROUP BY
customerNumber) AS
subquery
WHERE order_count >= 5;
```

- 주문 개수가 5개 이상인 고객만 조회
- 서브 쿼리의 결과가 테이블


- 결과

| 결과 #1 (7r x 2c) |             |
|-----------------|-------------|
| customerNumber  | order_count |
| 114             | 5           |
| 124             | 17          |
| 141             | 26          |
| 145             | 5           |
| 148             | 5           |
| 323             | 5           |
| 353             | 5           |

## Scalar Subquery 실습

- 문제 1: 최대 주문 개수를 가진 고객의 ID를 조회

- 정답

| orders (1r × 1c) |                                                                                     |
|------------------|-------------------------------------------------------------------------------------|
| customerNumber   |  |
|                  | 141                                                                                 |

## Table Subquery 실습

- 문제 2: products와 orderdetails 테이블을 이용하여, 각 제품별로 총 주문 금액(quantityOrdered \* priceEach)을 계산하고, 그 결과를 기반으로 상위 5개의 제품만 조회하세요.

- 정답

| 결과 #1 (5r x 3c) |                                |             |
|-----------------|--------------------------------|-------------|
| productCode     | productName                    | totalAmount |
| S18_3232        | 1992 Ferrari 360 Spider red    | 276,839.98  |
| S12_1108        | 2001 Ferrari Enzo              | 190,755.86  |
| S10_1949        | 1952 Alpine Renault 1300       | 190,017.96  |
| S10_4698        | 2003 Harley-Davidson Eagle ... | 170,686.0   |
| S12_1099        | 1968 Ford Mustang              | 161,531.48  |

## 상관 Subquery

- 상관 서브쿼리(Correlated Subquery)는 일반 서브쿼리와는 달리 메인 쿼리의 각 행을 참조하여 수행
- 메인 쿼리의 각 행마다 한 번씩 실행되며, 메인 쿼리의 현재 행과 연관된 결과를 반환



## 상관 Subquery

- **특징:**
- **반복 실행:** 상관 서브쿼리는 메인 쿼리의 각 행에 대해 별도로 실행. 따라서 메인 쿼리에 100개의 행이 있다면 상관 서브쿼리도 100번 실행
- **참조:** 상관 서브쿼리는 메인 쿼리의 열을 참조할 수 있음. 이 참조를 통해 서브쿼리는 메인 쿼리의 현재 행에 따라 다른 값을 반환할 수 있음.

## 상관 Subquery

- SQL Code & “각 고객별로 가장 최근의 주문일을 조회하세요”

```
SELECT c.customerName, o.orderDate
FROM customers c, orders o
WHERE c.customerNumber = o.customerNumber
AND o.orderDate = (SELECT MAX(orderDate) FROM orders WHERE
customerNumber = c.customerNumber);
```

- 결과

| 결과 #1 (98r x 2c)        |            |
|-------------------------|------------|
| customerName            | orderDate  |
| King Kong Collectab...  | 2003-12-01 |
| Men 'R' US Retailers... | 2004-01-09 |
| Double Decker Gift ...  | 2004-01-22 |

## (상관서브쿼리)Subquery 실습

- 문제 3: 2003년에 주문한 모든 고객의 이름을 조회하세요.

- 정답

| customers (74r × 1c)         |  |
|------------------------------|--|
| customerName                 |  |
| Atelier graphique            |  |
| Signal Gift Stores           |  |
| Australian Collectors, Co.   |  |
| Baane Mini Imports           |  |
| Mini Gifts Distributors Ltd. |  |

# 13.

## WITH 구문

Common Table Expression(CTE)

## Common Table Expression(CTE)

- WITH문 (Common Table Expression, CTE)과 서브쿼리는 둘 다 SQL 쿼리의 구조를 단순화하고 복잡한 쿼리를 분해하는 데 유용

## With문 vs subquery

- 정의 및 가독성:
- **CTE (WITH문)**: CTE는 쿼리의 시작 부분에 정의되며, 이름을 가진 임시 결과 집합을 생성-> 코드의 가독성 향상, CTE 내에서 정의된 쿼리를 메인 쿼리에서 여러 번 재사용할 수 있음.
- **서브쿼리**: 서브쿼리는 메인 쿼리 내에서 일반적으로 한 번만 사용됩니다.

## Common Table Expression(CTE)

- WITH문 (Common Table Expression, CTE)를 스크립드 맨 먼저 선언을 해준 다음, 메인쿼리에서 사용

```
WITH cte_name AS (
 -- CTE 정의 쿼리
)

-- 메인 쿼리
SELECT * FROM cte_name;
```

## Common Table Expression(CTE)

- WITH문 (Common Table Expression, CTE)를 스크립드 맨 먼저 선언을 해준 다음, 메인쿼리에서 사용



## Common Table Expression(CTE)

- orders 테이블에서 가장 최근 주문을 찾는 쿼리를 작성

```
WITH LatestOrders AS (
 SELECT customerNumber, MAX(orderDate) AS MaxOrderDate
 FROM orders
 GROUP BY customerNumber
)
SELECT o.orderNumber, o.orderDate, o.customerNumber
FROM orders o
JOIN LatestOrders lo ON o.customerNumber = lo.customerNumber AND
o.orderDate = lo.MaxOrderDate;
```

## Common Table Expression(CTE)실습

- 문제 5: products 테이블에서 각 제품 라인별로 평균 제품 가격을 계산하세요. 그리고 이 평균 가격보다 높은 가격을 가진 제품들만을 해당 제품 라인별로 조회하세요.

### 정답

| productName                    | productLine  | buyPrice | avgPrice  |
|--------------------------------|--------------|----------|-----------|
| 1952 Alpine Renault 1300       | Classic Cars | 98.58    | 64.446316 |
| 1996 Moto Guzzi 1100i          | Motorcycles  | 68.99    | 50.685385 |
| 2003 Harley-Davidson Eagle ... | Motorcycles  | 91.02    | 50.685385 |
| 1972 Alfa Romeo GTA            | Classic Cars | 85.68    | 64.446316 |
| 1962 LanciaA Delta 16V         | Classic Cars | 103.42   | 64.446316 |
| 1968 Ford Mustang              | Classic Cars | 95.34    | 64.446316 |
| 2001 Ferrari Enzo              | Classic Cars | 95.59    | 64.446316 |

## Common Table Expression(CTE)실습

- 문제 6: 각 제품 라인별로 제품의 평균 가격과 전체 제품의 평균 가격을 비교하여 전체 평균 가격보다 높은 제품 라인만 조회하세요.

- 정답

| products (2r × 3c) |           |               |
|--------------------|-----------|---------------|
| productLine        | avgPrice  | totalAvgPrice |
| Classic Cars       | 64.446316 | 54.395182     |
| Trucks and Buses   | 56.329091 | 54.395182     |

## Common Table Expression(CTE)실습

- 문제 7: 각 직원별로  
맡은 고객들의 평균  
크레딧 한도를  
계산하고, 크레딧  
한도의 평균이  
\$100,000 이상인  
직원만 조회하세요.

- 정답

| employees (1r x 4c) |           |          |                |
|---------------------|-----------|----------|----------------|
| employeeNumber      | firstName | lastName | avgCredit      |
| 1,165               | Leslie    | Jennings | 100,433.333333 |