DATABASES

CODINGHUB

MK

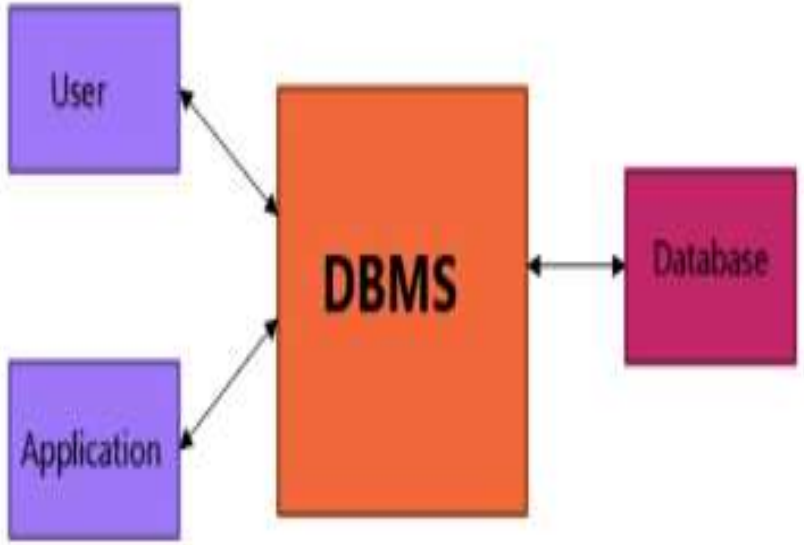
WHAT IS A DATABASE?

- Database is structured set of data for quick access, manage and update.
- Database is a **collection of one or more tables** and the tables contain data rows or records.
- Refrigerator example: refrigerator is database, refrigerator compartments are tables and vegetables, foods and beverages is data.

DATA BASE MANAGEMENT SYSTEMS(SOFTWARE)

- Program to interact with the databases. Create and manage databases. Store, modify and extract data.
- Interface between the databases and database applications or users.
- Manages 3 important things: 1) The Data 2) Data engines 3) Database schema
- **Types of DBMS**:

1. RDMS: Relational database management system.
2. NoSQL DBMS
3. In memory database management systems. (IMDBMS)
4. Columnar database management system (CDBMS)
5. Cloud based database management system



SQL(LANGUAGE)

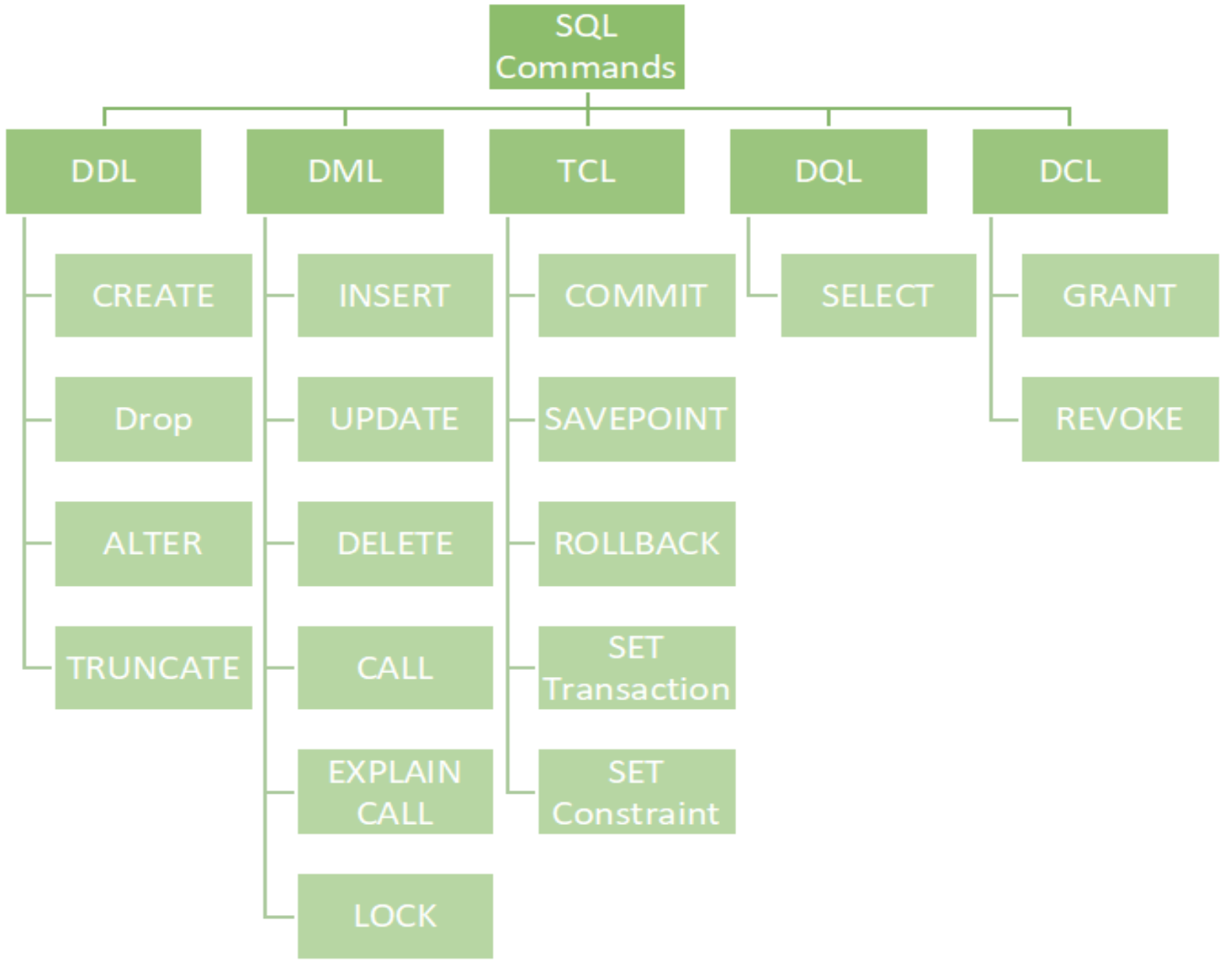
- **SQL IS USED TO COMMUNICATE WITH DATABASE .** THE STANDARD LANGUAGE FOR RELATIONAL DATABASE SYSTEM
- **SQL** IS A DATABASE COMPUTER LANGUAGE DESIGNED FOR THE RETRIEVAL AND MANAGEMENT OF DATA IN A RELATIONAL DATABASEYOU CAN USE SQL TO GET THE DBMS TO DO THINGS FOR YOU
- SQL IS STRUCTURED QUERY LANGUAGE, WHICH IS A COMPUTER LANGUAGE FOR STORING, MANIPULATING AND RETRIEVING DATA STORED IN A RELATIONAL DATABASE.

CONTD…

- SQL IS A COMBINATION OF 4 LANGUAGES
- DQL : DATA QUERY LANGUAGE
  - USED TO QUERY DATABASE FOR INFORMATION
  - GET INFORMATION THAT IS ALREADY STORED THERE
- DML : DATA MANIPULATION LANGUAGE
  - USED FOR INSERTING, DELETING AND UPDATING DATA FROM THE DATABASE
- DDL : DATA DEFINITION LANGUAGE
  - USED TO DEFINE THE TABLES(DATABASE SCHEMAS)
- DCL : DATA CONTROL LANGUAGE
  - USED FOR CONTROLLING ACCESS TO THE DATA IN THE DATA BASE
  - USER AND PERMISSION MANAGEMENT

RDBMS(SOFTWARE)

- Relational Database Management System (RDBMS) is a more advanced version of a DBMS system that allows access to data in a more efficient way. It is used to store or manage only the data that are in the form of tables.
- The software used to store, manage, query, and retrieve data stored in a relational database is called a relational database management system (RDBMS)
- The RDBMS provides an interface between users and applications and the database
- Data in the form of a table are linked together
- EXAMPLES:
  - SQL Server, Oracle, MySQL, MariaDB, and SQLite.


SQL Commands
DDL
DML
TCL
DQL
DCL
CREATE
Drop
ALTER
TRUNCATE
INSERT
UPDATE
DELETE
CALL
EXPLAIN CALL
LOCK
COMMIT
SAVEPOINT
ROLLBACK
SET Transaction
SET Constraint
SELECT
GRANT
REVOKE

Features of RDBMS

- Offers information to be saved in the tables
- In order to exclusively find out the rows, the primary key is used
- The data are always saved in rows and columns
- To retrieve the information the indexes are used
- Columns are being shared between tables using the keys

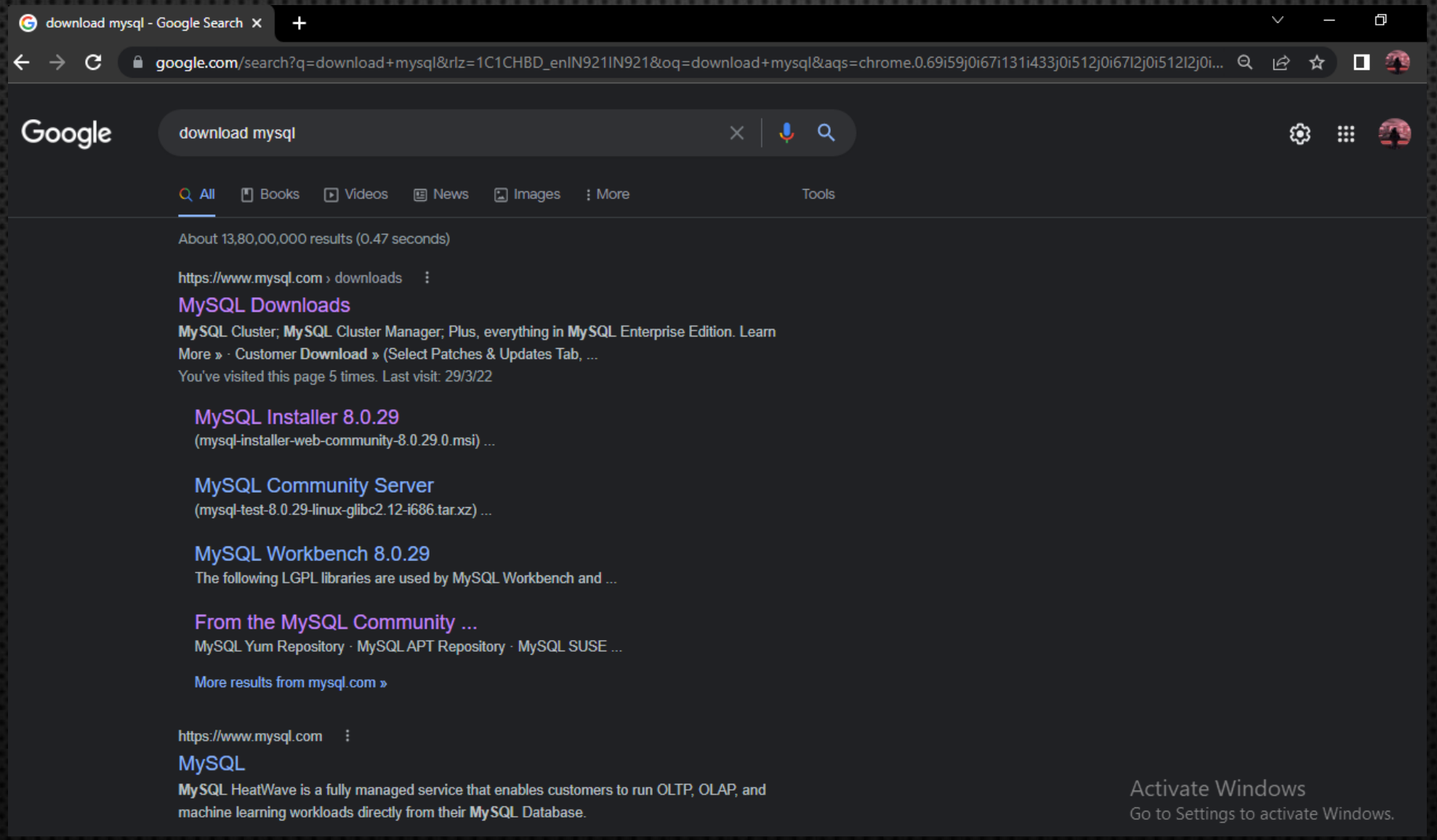
MYSQL

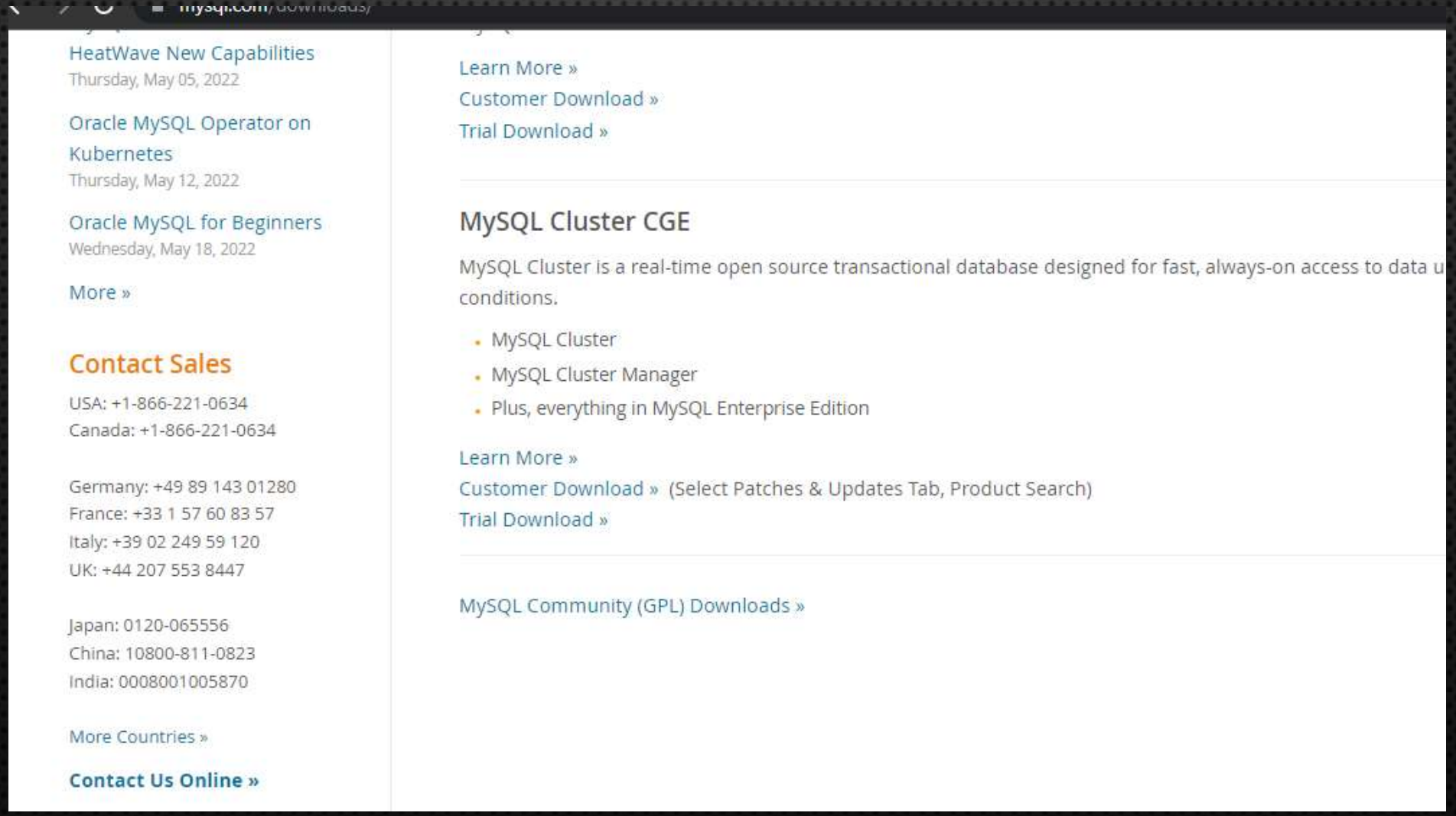
- MySQL is the most popular open source database OFFERED BY ORACLE
- **MySQL is a relational database management system (RDBMS) developed by Oracle that is based on structured query language (SQL).**
- MySQL is a tool used to manage databases and servers, so while it's not a database, it's widely used in relation to managing and organising data in databases

SQL (LANGUAGE)VS MYSQL(SOFTWARE)

- Structured query language (SQL) is a standard language for database creation and manipulation.
- MySQL is a relational database program

SQL	MySQL
SQL is a language to manage databases.	MySQL is a database software.
SQL is used to query databases.	MySQL stores the data.
SQL is structured query language.	MySQL is RDBMS (Relational Database Management System)
SQL does not provide connectors.	MySQL provide an integrated tool called "MySQL workbench"
SQL codes or commands are used in Oracle, SQL server, PostgreSQL, DB2, MariaDB, MySQL etc.	MySQL uses SQL.


download mysql - Google Search
google.com/search?q=download+mysql&rlz=1C1CHBD_enIN921IN921&oq=download+mysql&aqs=chrome.0.69i59j0i67i131i433j0i512j0i67l2j0i512l2j0i...
Google
download mysql
All
Books
Videos
News
Images
More
Tools
About 13,80,00,000 results (0.47 seconds)
https://www.mysql.com › downloads
MySQL Downloads
MySQL Cluster; MySQL Cluster Manager; Plus, everything in MySQL Enterprise Edition. Learn More » · Customer Download » (Select Patches & Updates Tab, ...
You've visited this page 5 times. Last visit: 29/3/22
MySQL Installer 8.0.29
(mysql-installer-web-community-8.0.29.0.msi) ...
MySQL Community Server
(mysql-test-8.0.29-linux-glibc2.12-i686.tar.xz) ...
MySQL Workbench 8.0.29
The following LGPL libraries are used by MySQL Workbench and ...
From the MySQL Community ...
MySQL Yum Repository · MySQL APT Repository · MySQL SUSE ...
More results from mysql.com »
https://www.mysql.com
MySQL
MySQL HeatWave is a fully managed service that enables customers to run OLTP, OLAP, and machine learning workloads directly from their MySQL Database.
Activate Windows
Go to Settings to activate Windows.


HeatWave New Capabilities
Thursday, May 05, 2022
Oracle MySQL Operator on Kubernetes
Thursday, May 12, 2022
Oracle MySQL for Beginners
Wednesday, May 18, 2022
More »
Contact Sales
USA: +1-866-221-0634
Canada: +1-866-221-0634
Germany: +49 89 143 01280
France: +33 1 57 60 83 57
Italy: +39 02 249 59 120
UK: +44 207 553 8447
Japan: 0120-065556
China: 10800-811-0823
India: 0008001005870
More Countries »
Contact Us Online »
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 u
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 »

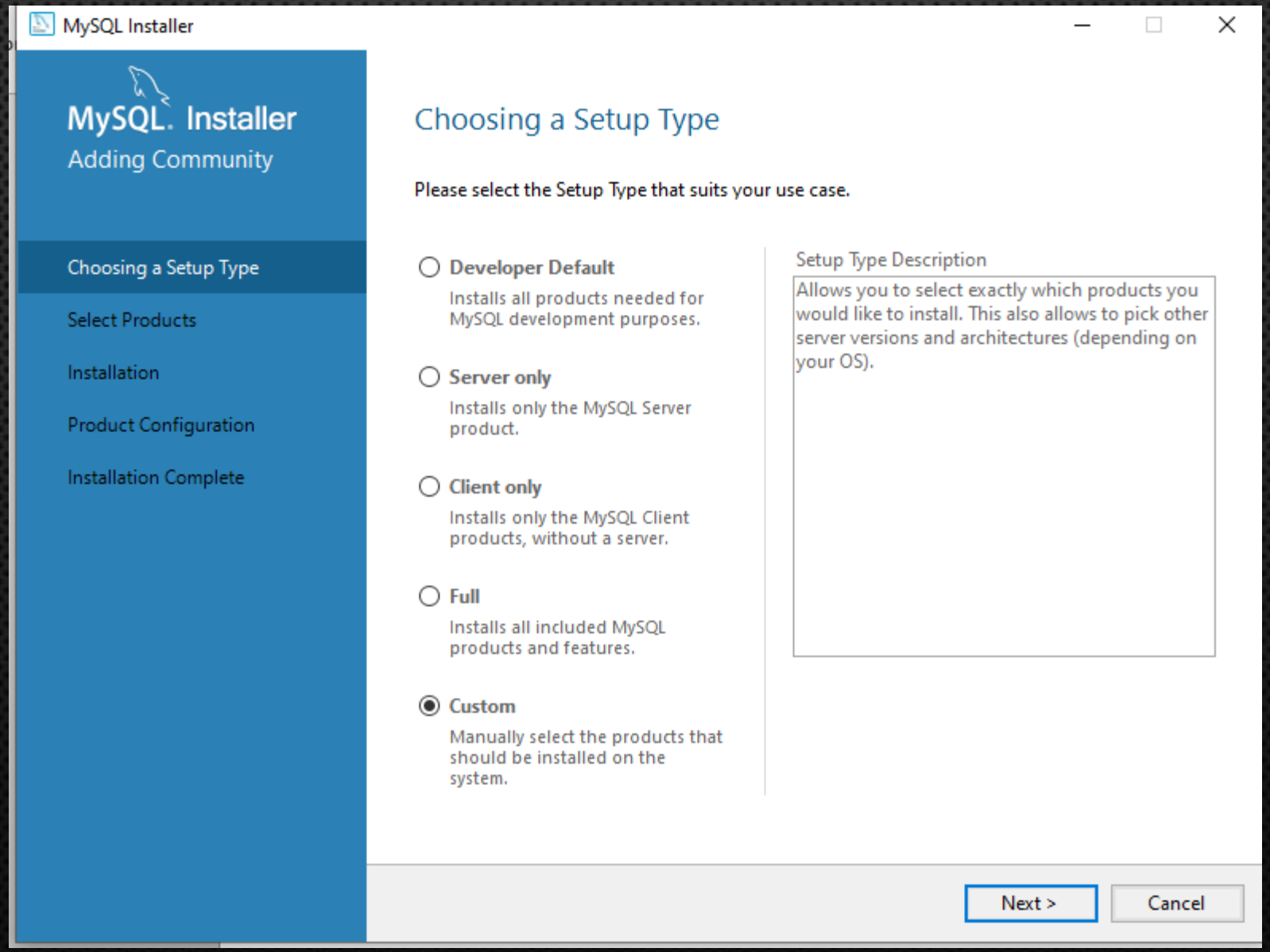
- MySQL Yum Repository
- MySQL APT Repository
- MySQL SUSE Repository

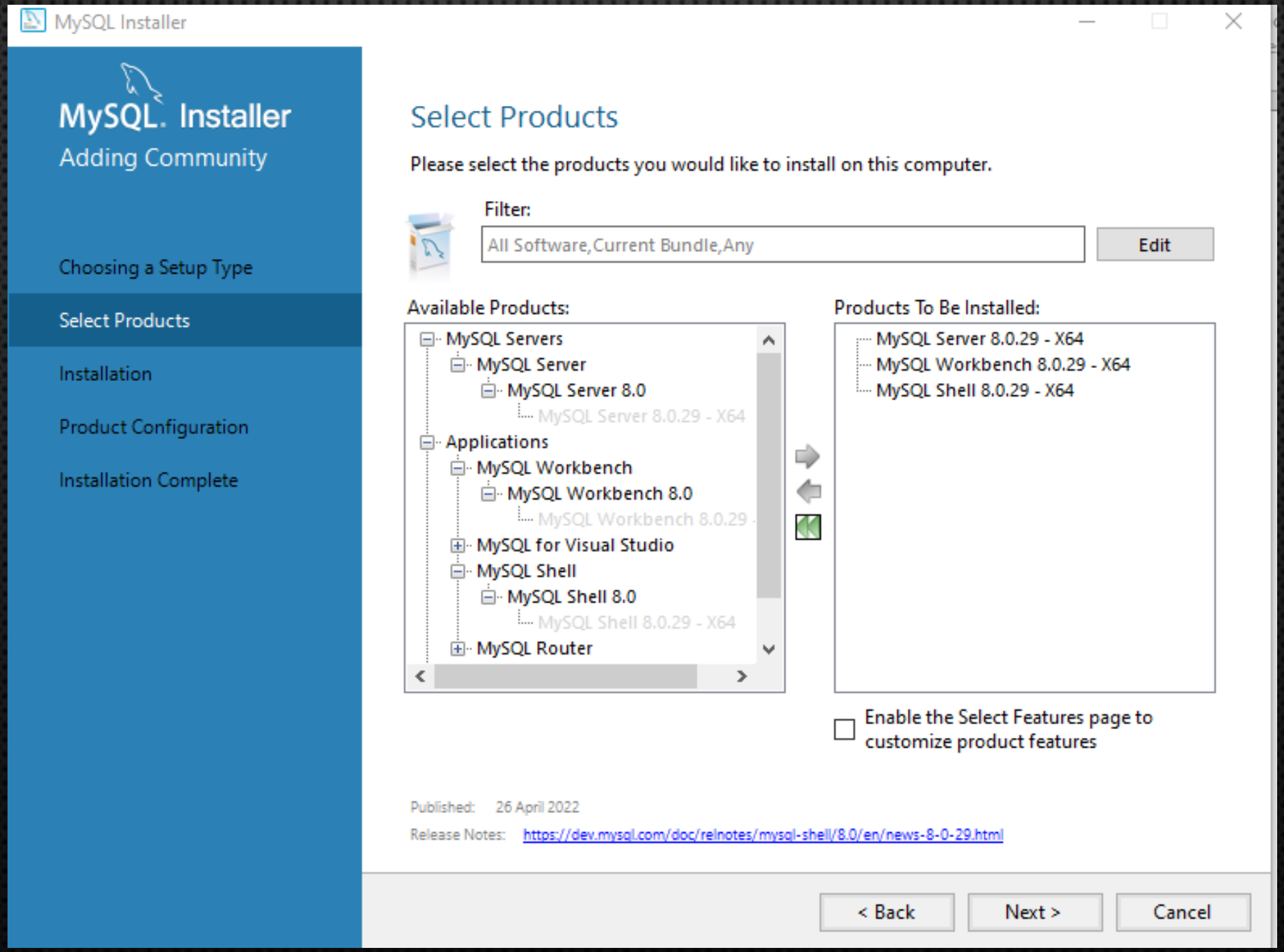
- MySQL Community Server
- MySQL Cluster
- MySQL Router
- MySQL Shell
- MySQL Workbench

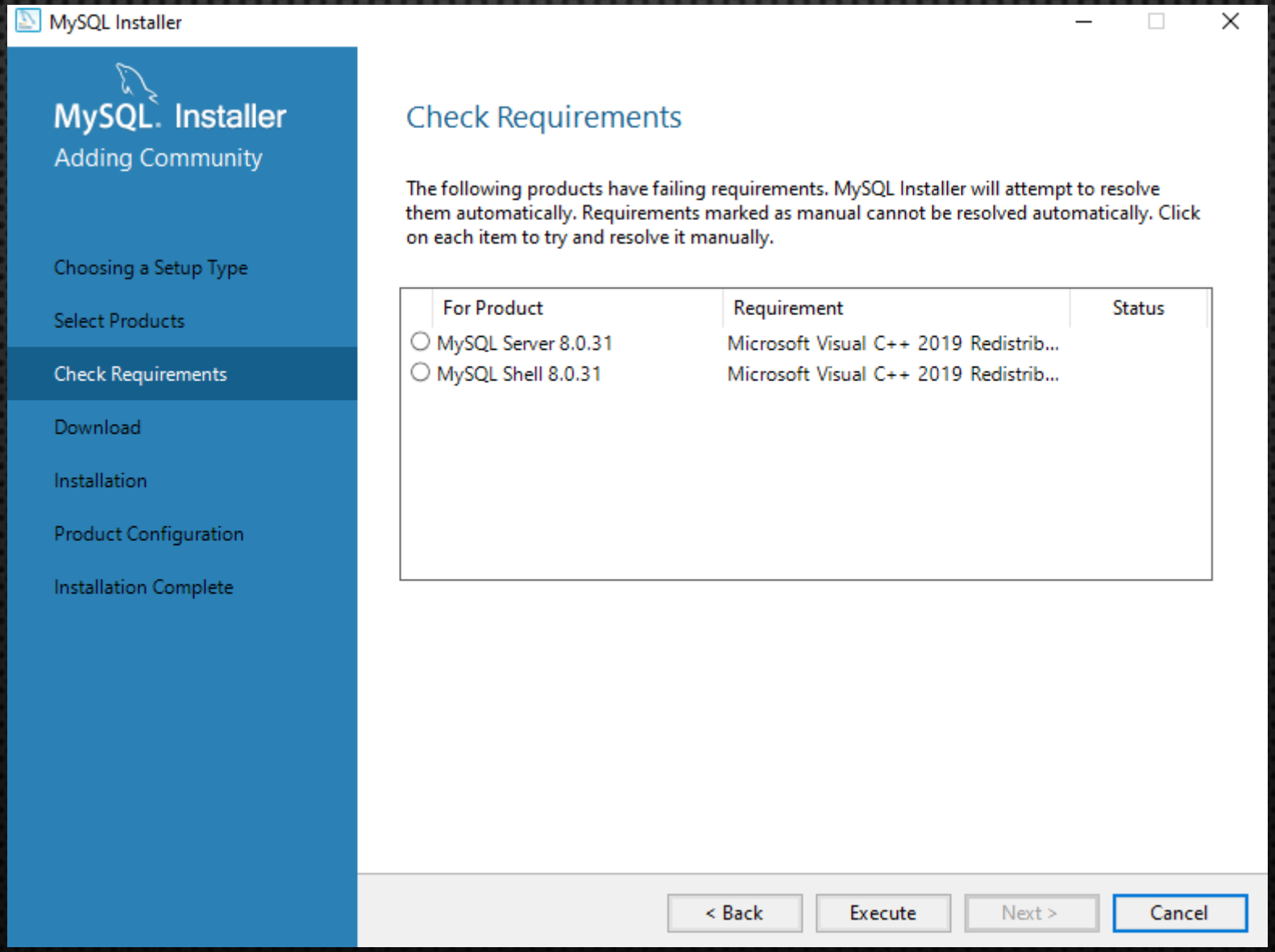
- MySQL Installer for Windows
- MySQL for Visual Studio

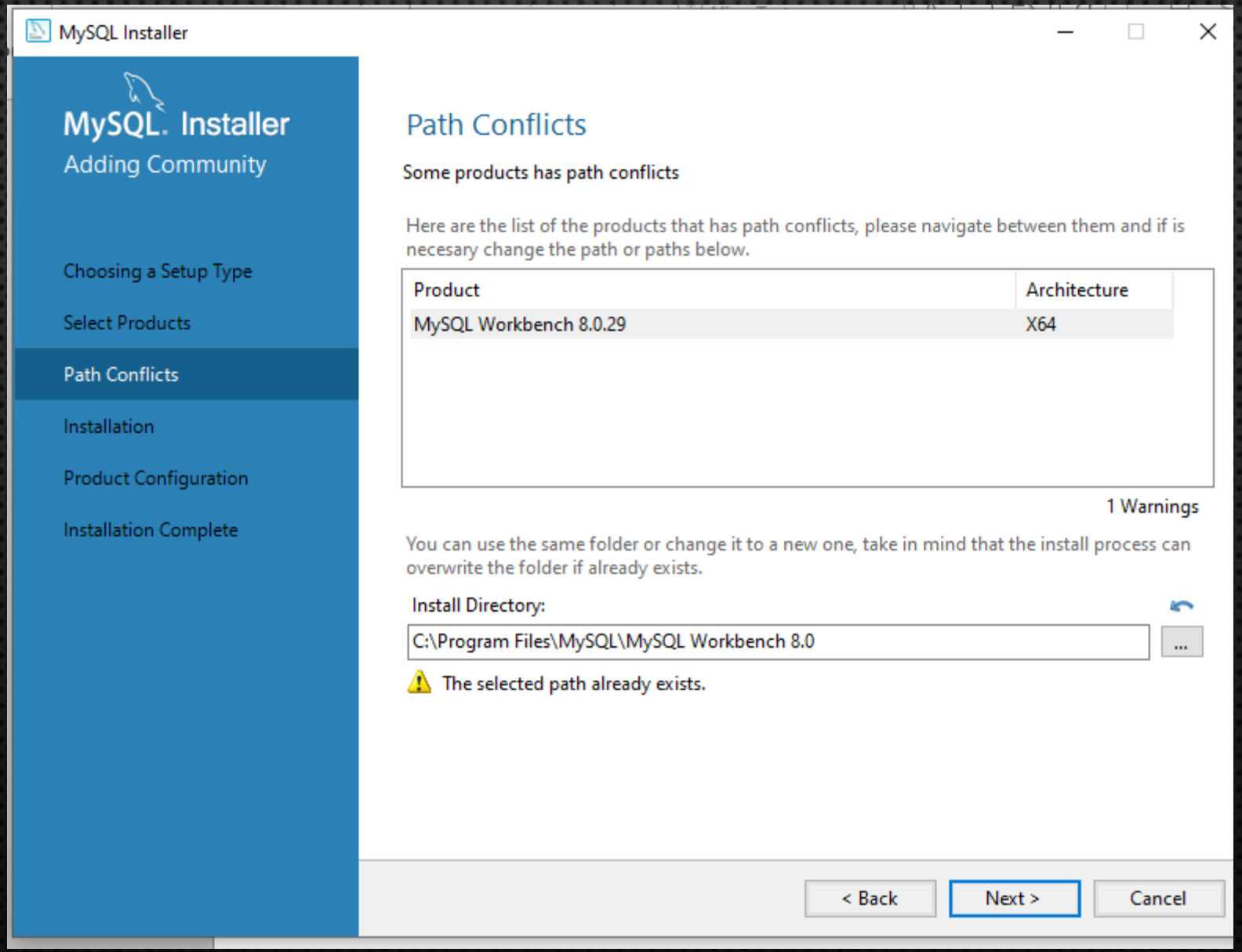
- C API (libmysqlclient)
- Connector/C++
- Connector/J
- Connector/NET
- Connector/Node.js
- Connector/ODBC
- Connector/Python
- MySQL Native Driver for PHP

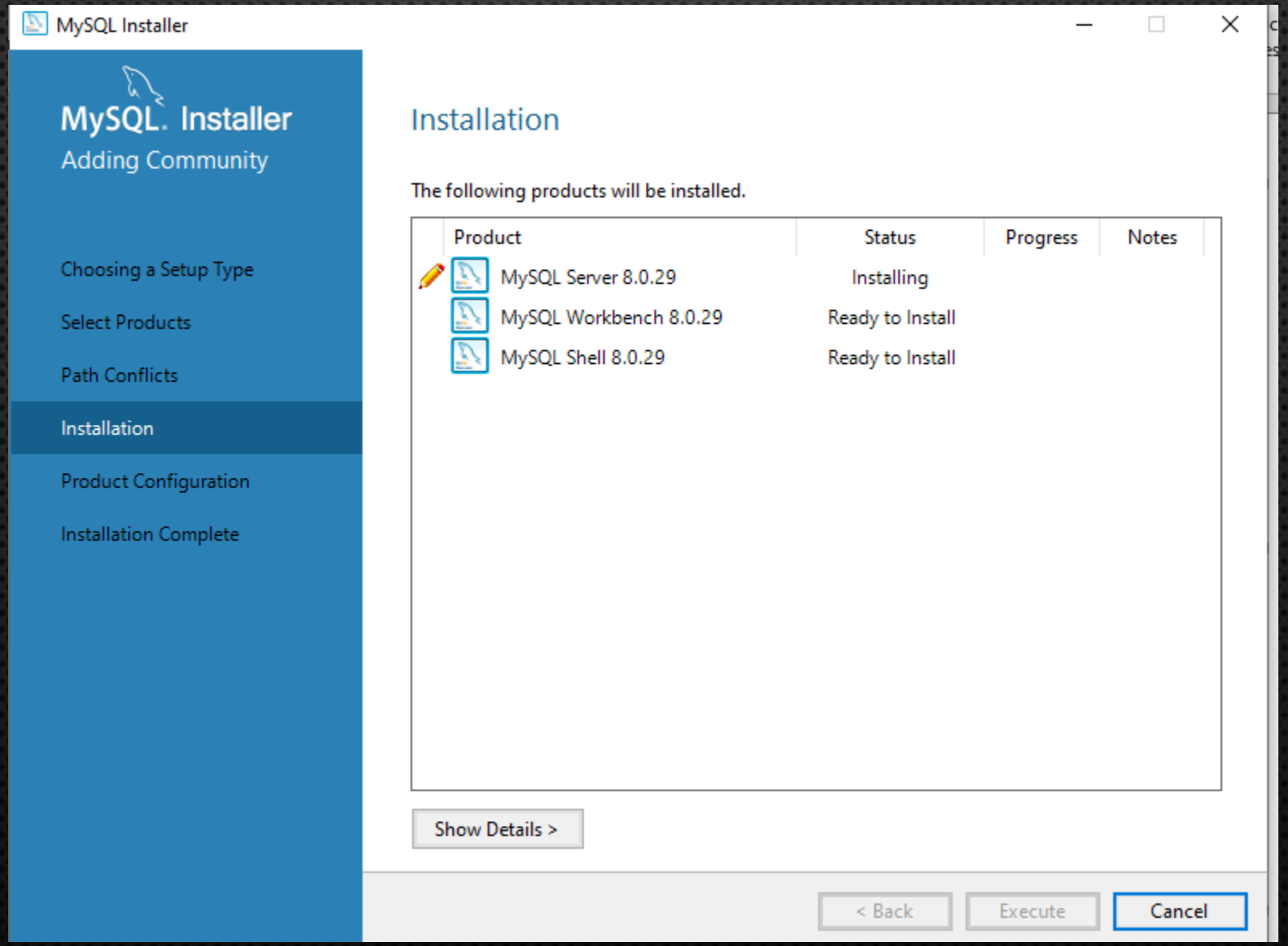
- MySQL Benchmark Tool
- Time zone description tables
- Download Archives

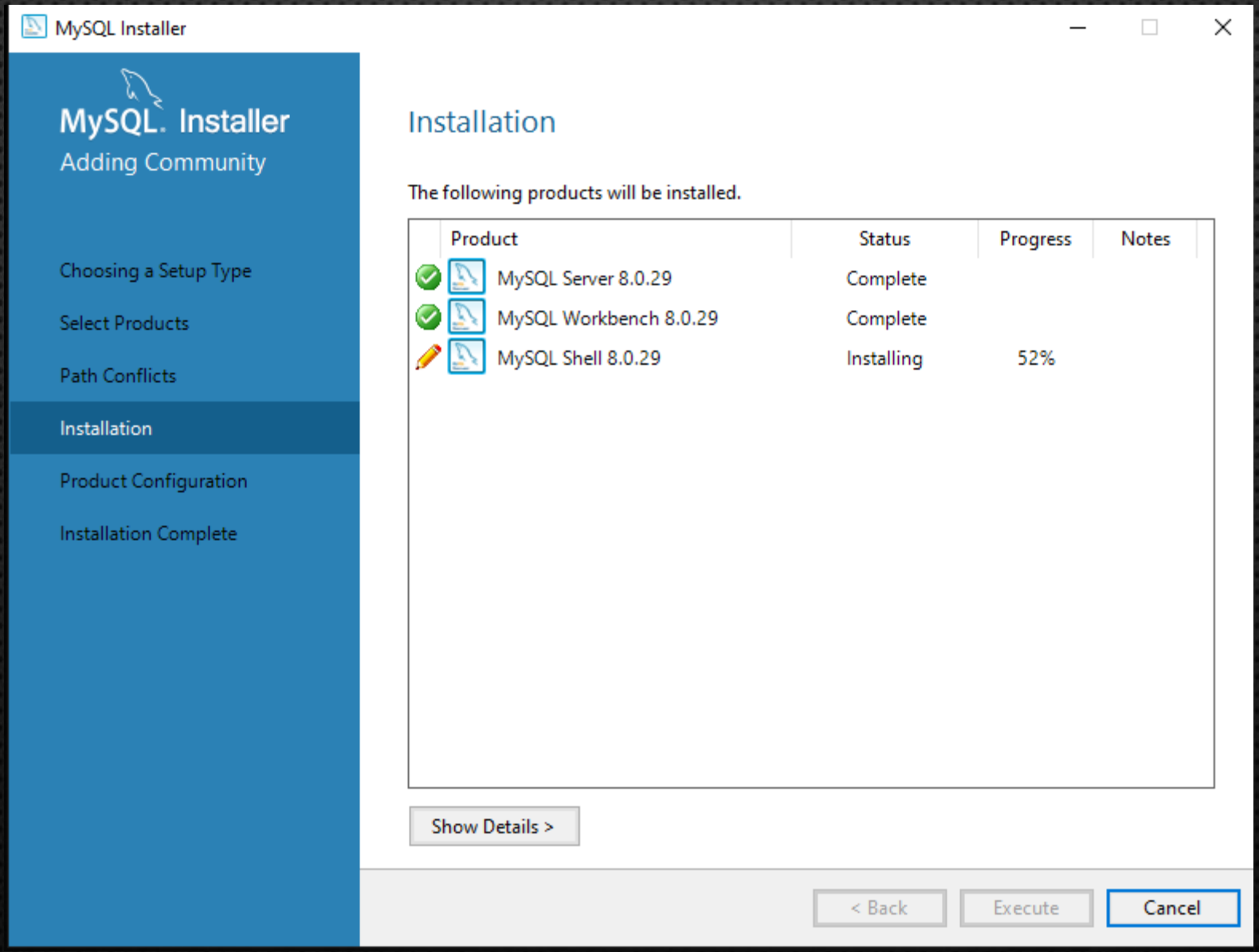

MySQL Installer
MySQL. Installer
Adding Community
Choosing a Setup Type
Select Products
Installation
Product Configuration
Installation Complete
Choosing a Setup Type
Please select the Setup Type that suits your use case.
Developer Default
Installs all products needed for MySQL development purposes.
Server only
Installs only the MySQL Server product.
Client only
Installs only the MySQL Client products, without a server.
Full
Installs all included MySQL products and features.
Custom
Manually select the products that should be installed on the system.
Setup Type Description
Allows you to select exactly which products you would like to install. This also allows to pick other server versions and architectures (depending on your OS).
Next >
Cancel

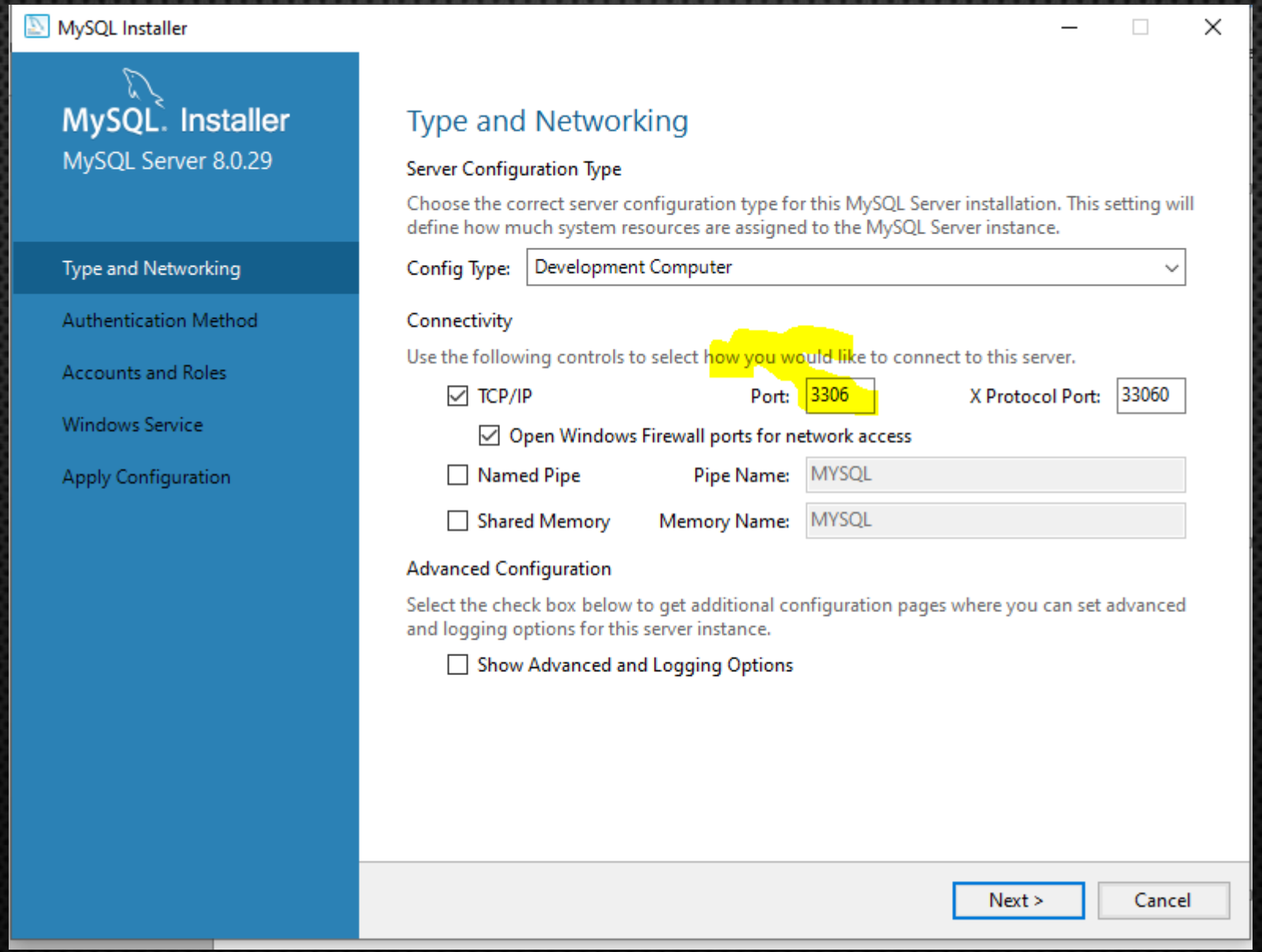

MySQL Installer
MySQL. Installer
Adding Community
Choosing a Setup Type
Select Products
Installation
Product Configuration
Installation Complete
Select Products
Please select the products you would like to install on this computer.
Filter:
All Software,Current Bundle,Any
Edit
Available Products:
MySQL Servers
MySQL Server
MySQL Server 8.0
MySQL Server 8.0.29 - X64
Applications
MySQL Workbench
MySQL Workbench 8.0
MySQL Workbench 8.0.29 -
MySQL for Visual Studio
MySQL Shell
MySQL Shell 8.0
MySQL Shell 8.0.29 - X64
MySQL Router
Products To Be Installed:
MySQL Server 8.0.29 - X64
MySQL Workbench 8.0.29 - X64
MySQL Shell 8.0.29 - X64
Enable the Select Features page to customize product features
Published: 26 April 2022
Release Notes: https://dev.mysql.com/doc/relnotes/mysql-shell/8.0/en/news-8-0-29.html
< Back
Next >
Cancel

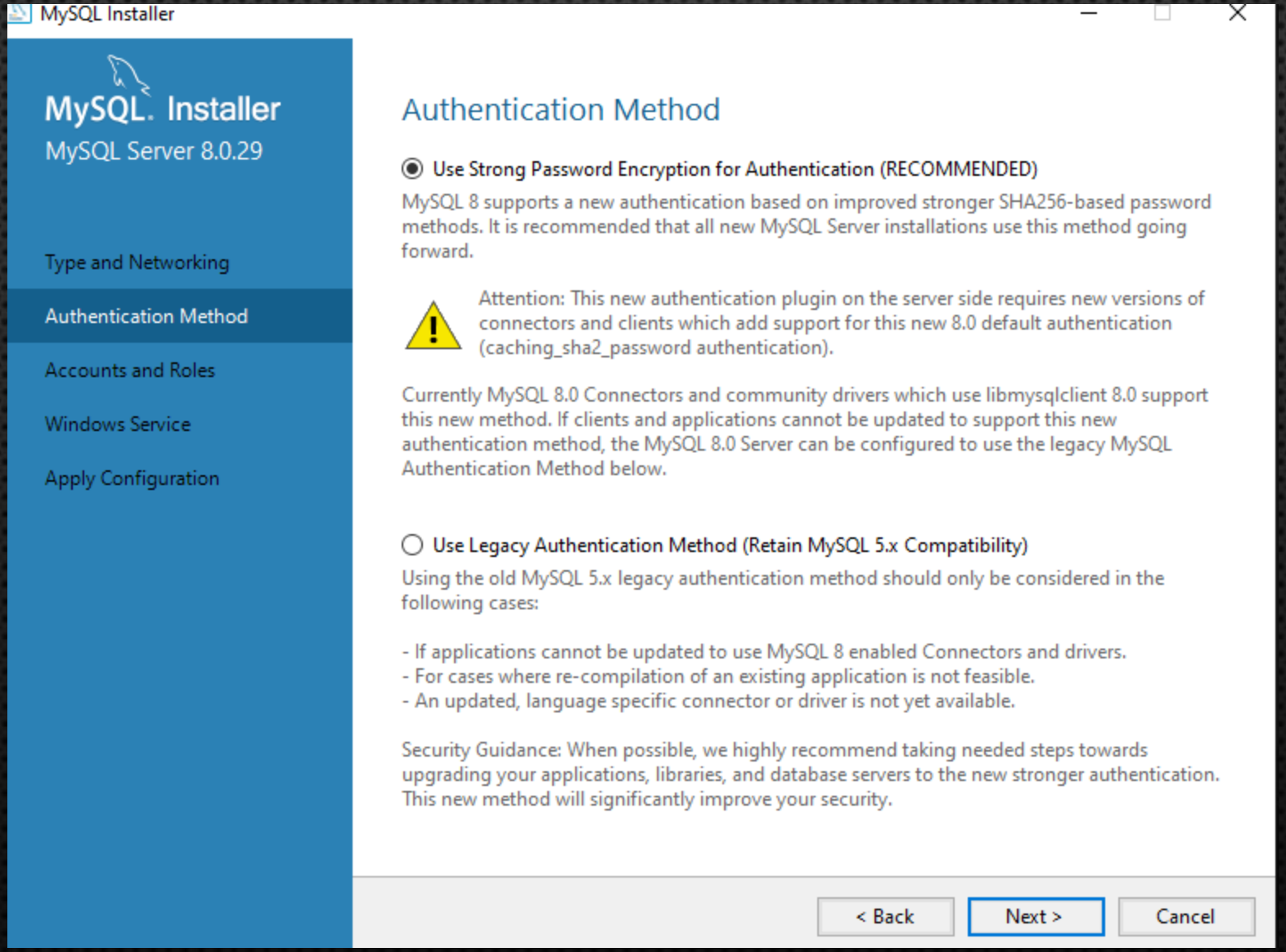

MySQL Installer
MySQL. Installer
Adding Community
Choosing a Setup Type
Select Products
Check Requirements
Download
Installation
Product Configuration
Installation Complete
Check Requirements
The following products have failing requirements. MySQL Installer will attempt to resolve them automatically. Requirements marked as manual cannot be resolved automatically. Click on each item to try and resolve it manually.
For Product
Requirement
Status
MySQL Server 8.0.31
Microsoft Visual C++ 2019 Redistrib...
MySQL Shell 8.0.31
Microsoft Visual C++ 2019 Redistrib...
< Back
Execute
Next >
Cancel

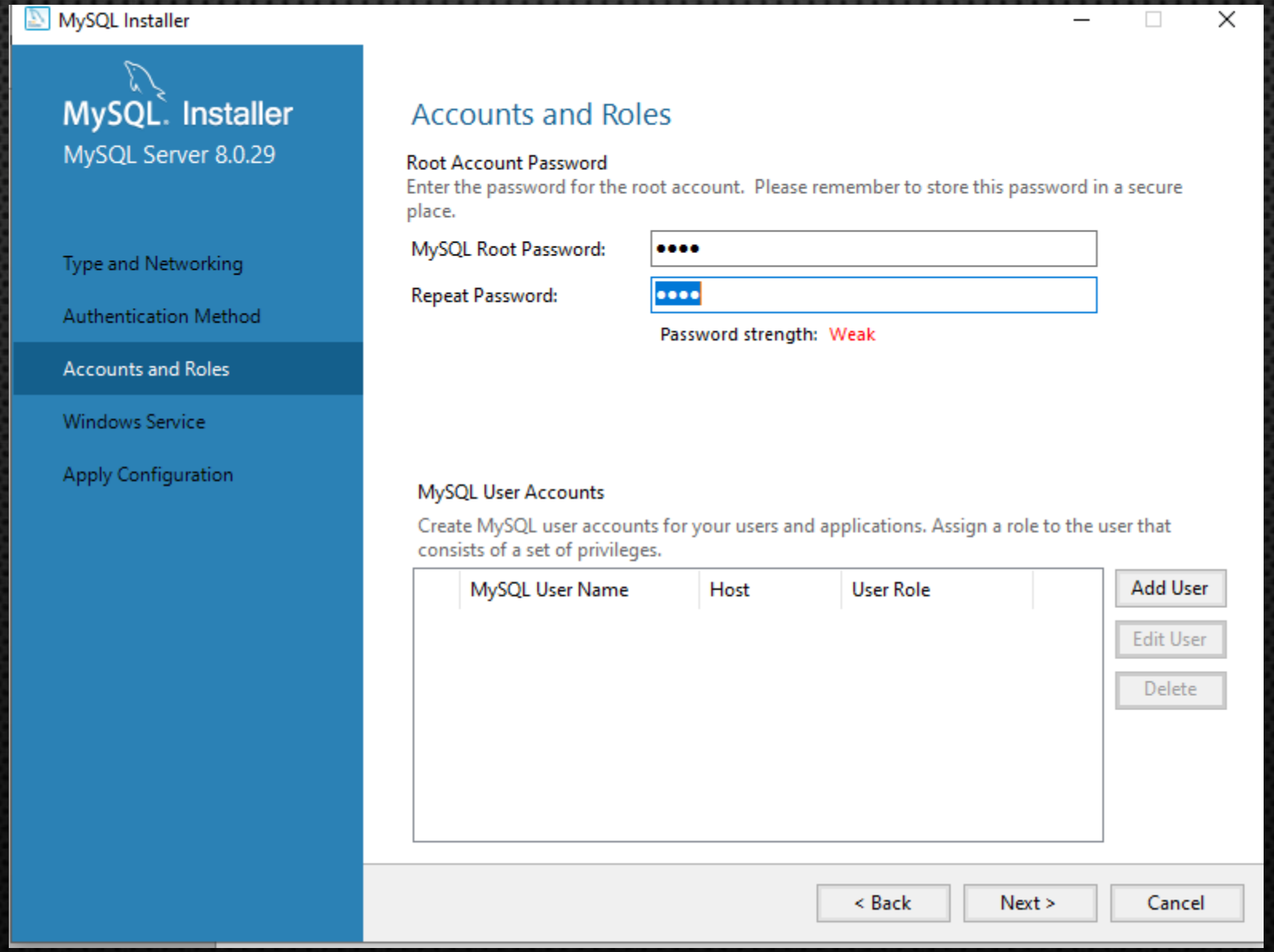

MySQL Installer
MySQL. Installer
Adding Community
Choosing a Setup Type
Select Products
Path Conflicts
Installation
Product Configuration
Installation Complete
Path Conflicts
Some products has path conflicts
Here are the list of the products that has path conflicts, please navigate between them and if is necesary change the path or paths below.
Product
Architecture
MySQL Workbench 8.0.29
X64
1 Warnings
You can use the same folder or change it to a new one, take in mind that the install process can overwrite the folder if already exists.
Install Directory:
C:\Program Files\MySQL\MySQL Workbench 8.0
...
The selected path already exists.
< Back
Next >
Cancel

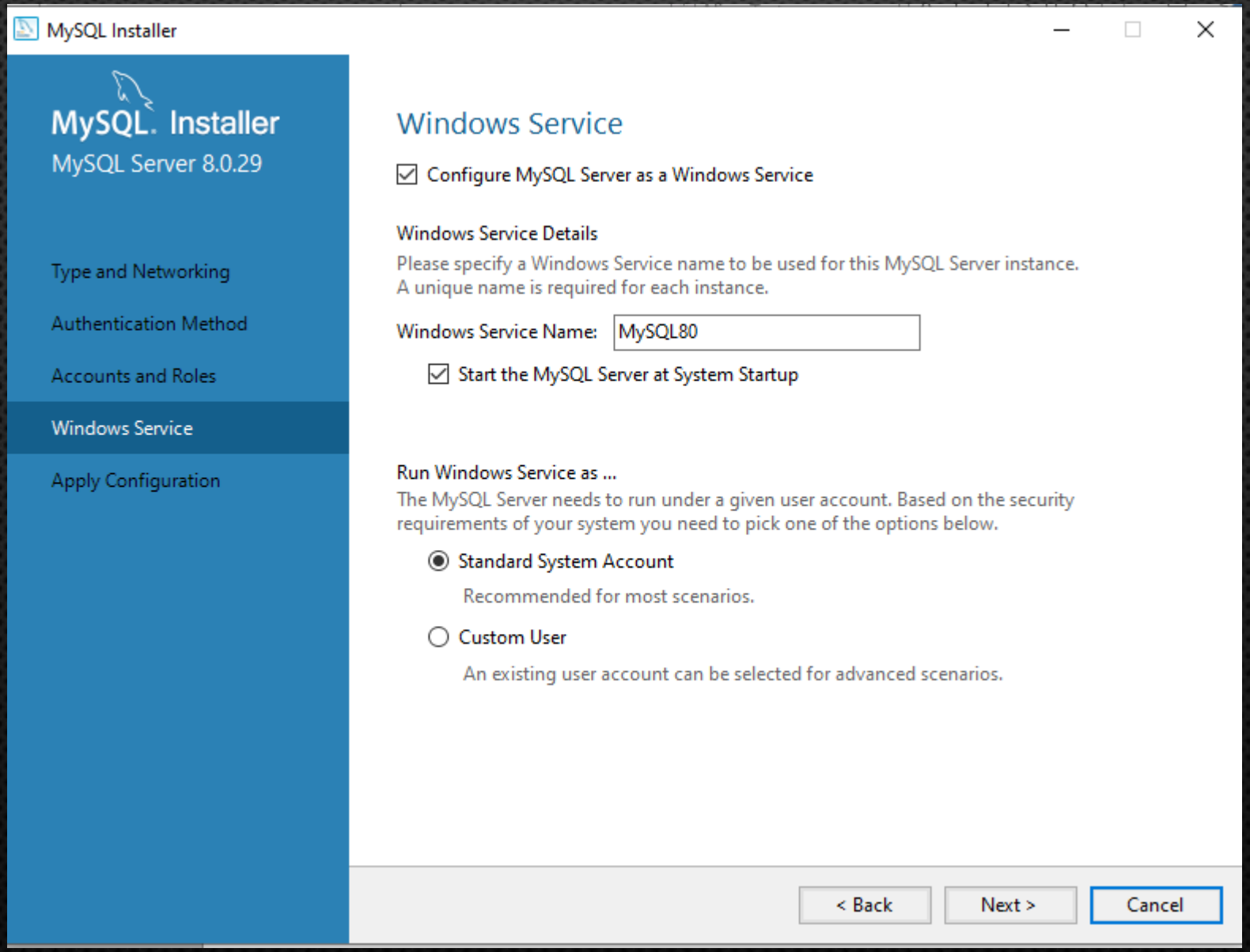

MySQL Installer
MySQL. Installer
Adding Community
Choosing a Setup Type
Select Products
Path Conflicts
Installation
Product Configuration
Installation Complete
Installation
The following products will be installed.
Product
Status
Progress
Notes
MySQL Server 8.0.29
Installing
MySQL Workbench 8.0.29
Ready to Install
MySQL Shell 8.0.29
Ready to Install
Show Details >
< Back
Execute
Cancel

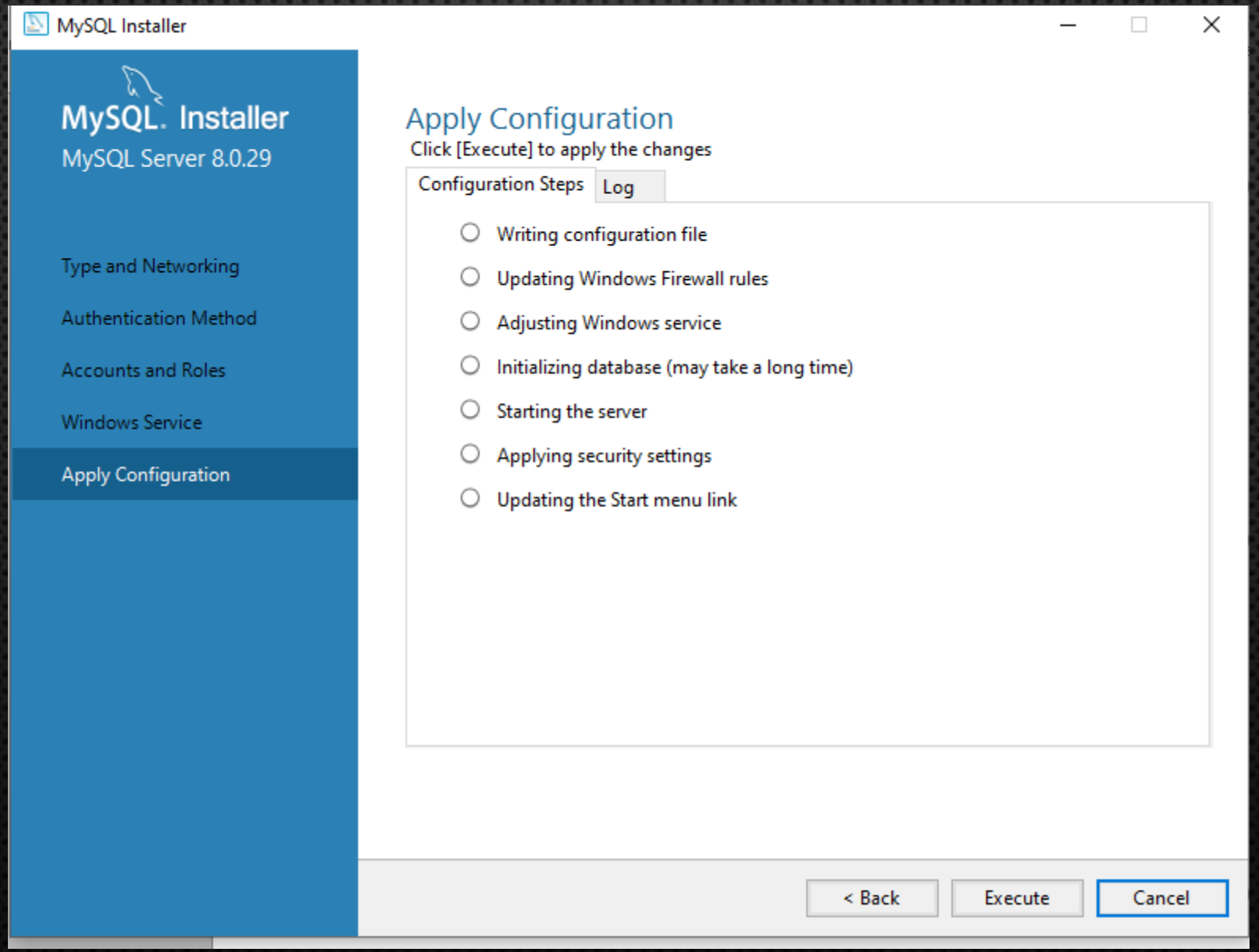

MySQL Installer
MySQL. Installer
Adding Community
Choosing a Setup Type
Select Products
Path Conflicts
Installation
Product Configuration
Installation Complete
Installation
The following products will be installed.
Product
Status
Progress
Notes
MySQL Server 8.0.29
Complete
MySQL Workbench 8.0.29
Complete
MySQL Shell 8.0.29
Installing
52%
Show Details >
< Back
Execute
Cancel

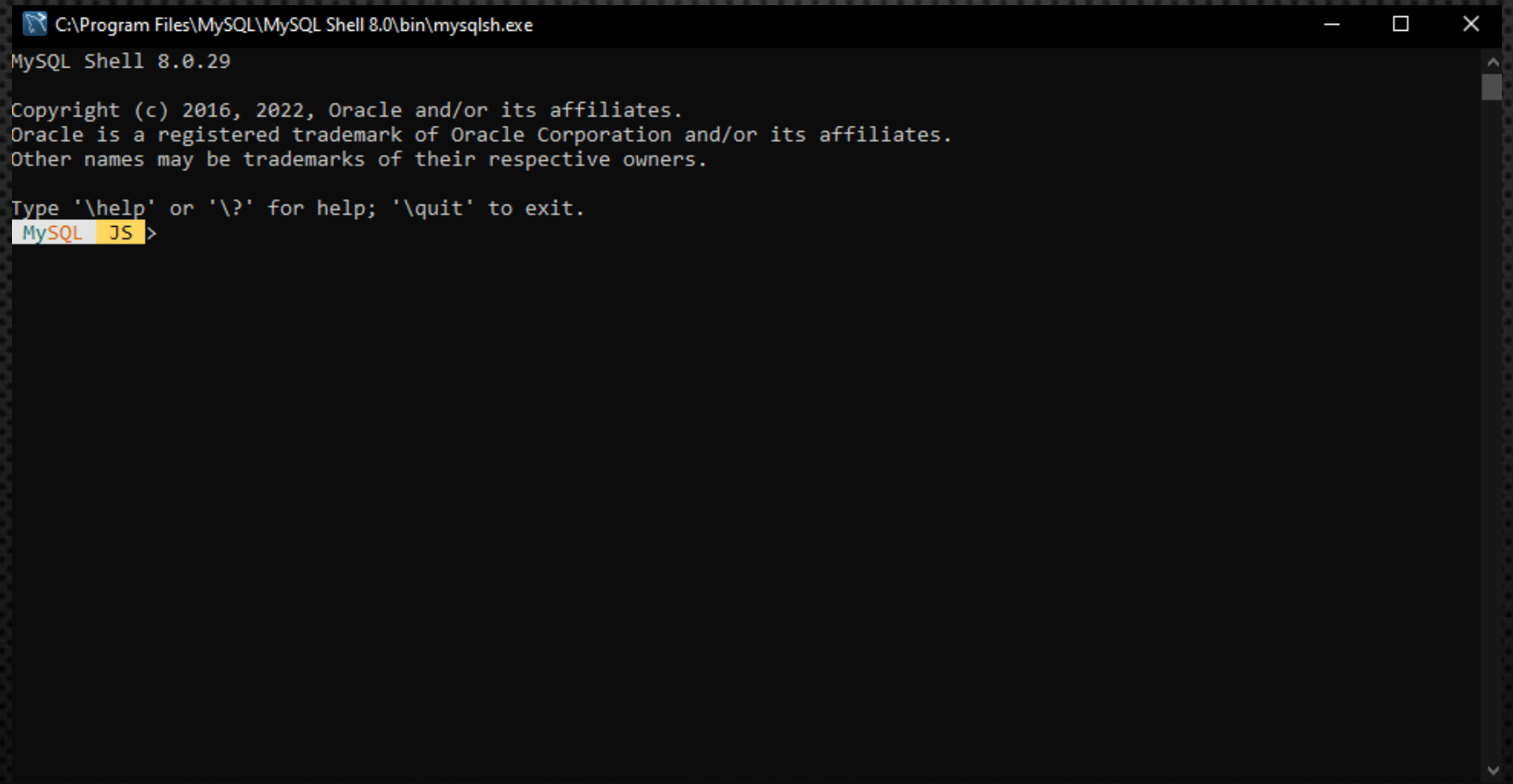

MySQL Installer
MySQL. Installer
MySQL Server 8.0.29
Type and Networking
Authentication Method
Accounts and Roles
Windows Service
Apply Configuration
Type and Networking
Server Configuration Type
Choose the correct server configuration type for this MySQL Server installation. This setting will define 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

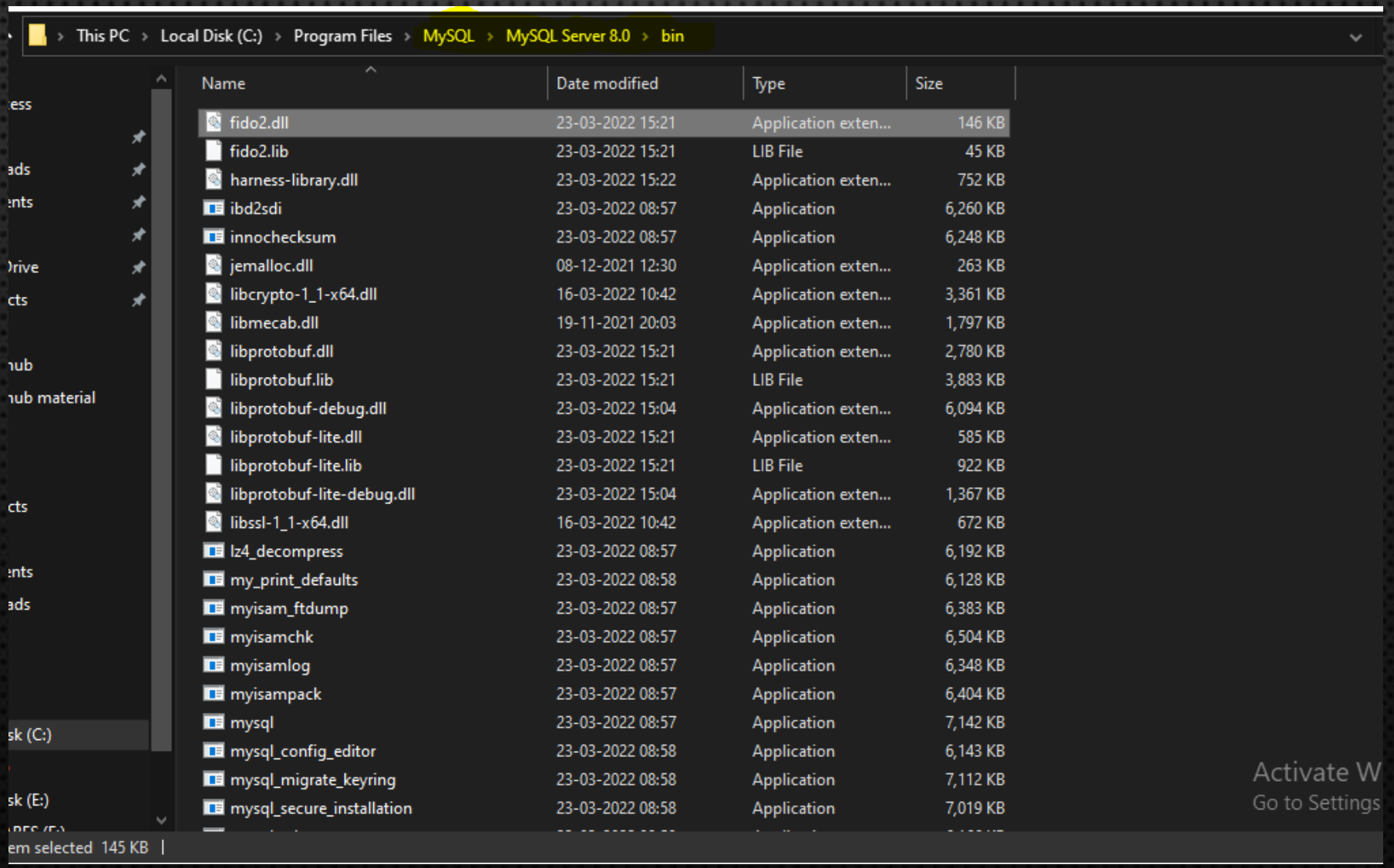

MySQL Installer
MySQL. Installer
MySQL Server 8.0.29
Type and Networking
Authentication Method
Accounts and Roles
Windows Service
Apply Configuration
Authentication Method
Use Strong Password Encryption for Authentication (RECOMMENDED)
MySQL 8 supports a new authentication based on improved stronger SHA256-based password methods. It is recommended that all new MySQL Server installations use this method going forward.
Attention: This new authentication plugin on the server side requires new versions of connectors and clients which add support for this new 8.0 default authentication (caching_sha2_password authentication).
Currently MySQL 8.0 Connectors and community drivers which use libmysqlclient 8.0 support this new method. If clients and applications cannot be updated to support this new authentication method, the MySQL 8.0 Server can be configured to use the legacy MySQL Authentication Method below.
Use Legacy Authentication Method (Retain MySQL 5.x Compatibility)
Using the old MySQL 5.x legacy authentication method should only be considered in the following cases:
- If applications cannot be updated to use MySQL 8 enabled Connectors and drivers.
- For cases where re-compilation of an existing application is not feasible.
- An updated, language specific connector or driver is not yet available.
Security Guidance: When possible, we highly recommend taking needed steps towards upgrading your applications, libraries, and database servers to the new stronger authentication. This new method will significantly improve your security.
< Back
Next >
Cancel

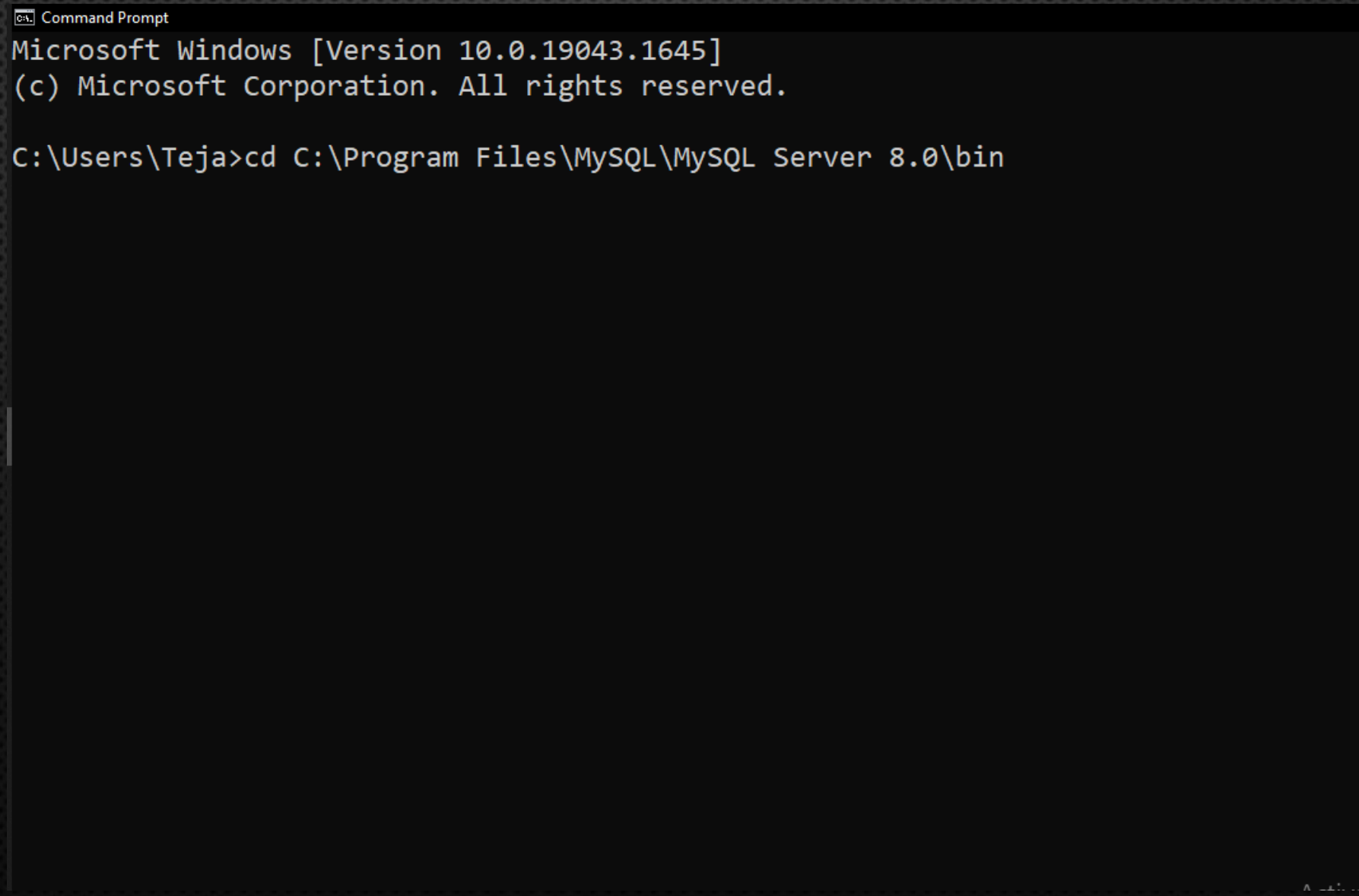

MySQL Installer
MySQL. Installer
MySQL Server 8.0.29
Type and Networking
Authentication Method
Accounts and Roles
Windows Service
Apply Configuration
Accounts and Roles
Root Account Password
Enter the password for the root account. Please remember to store this password in a secure place.
MySQL Root 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

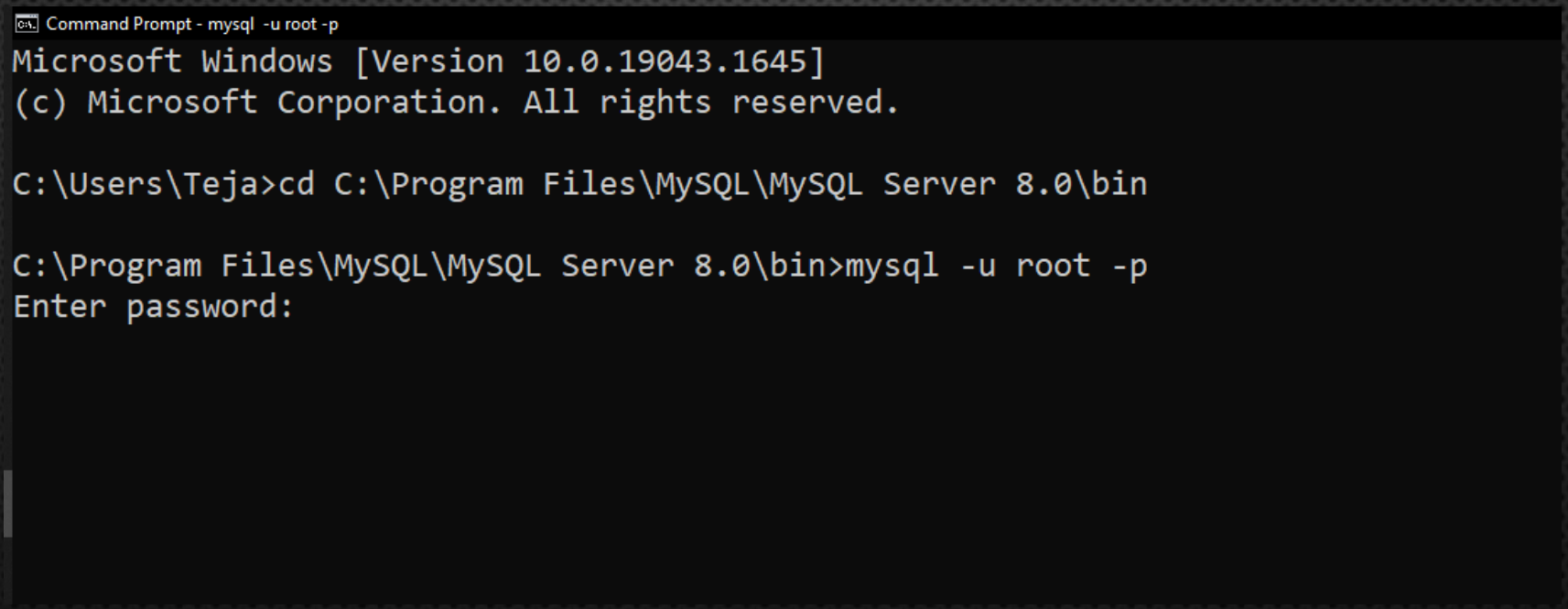

MySQL Installer
MySQL. Installer
MySQL Server 8.0.29
Type and Networking
Authentication Method
Accounts and Roles
Windows Service
Apply Configuration
Windows Service
Configure MySQL Server as a Windows Service
Windows Service Details
Please specify a Windows Service name to be used for this MySQL Server instance.
A unique name is required for each instance.
Windows Service Name:
MySQL80
Start the MySQL Server at System Startup
Run Windows Service as ...
The MySQL Server needs to run under a given user account. Based on the security requirements of your system you need to pick one of the options below.
Standard System Account
Recommended for most scenarios.
Custom User
An existing user account can be selected for advanced scenarios.
< Back
Next >
Cancel

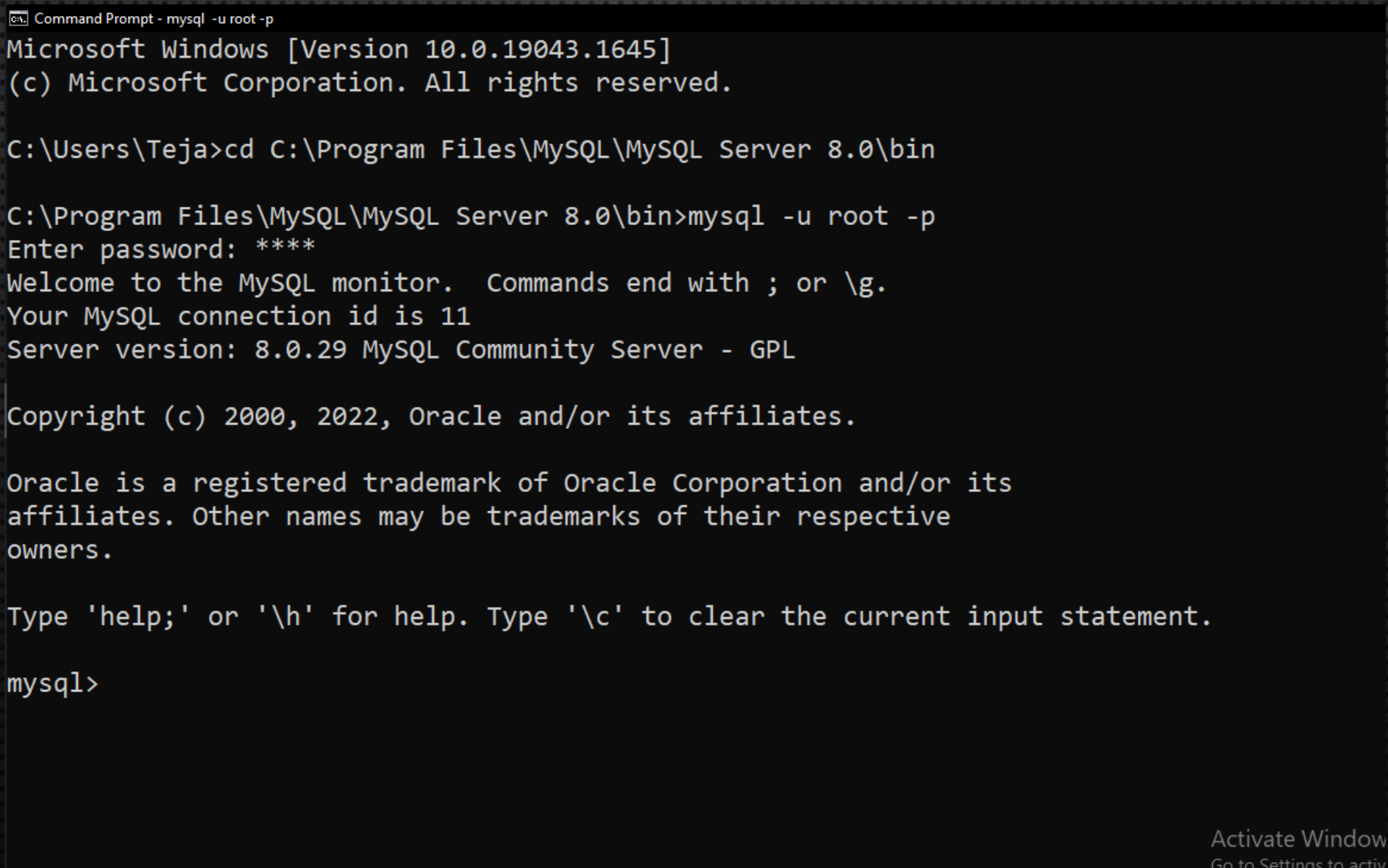

MySQL Installer
MySQL. Installer
MySQL Server 8.0.29
Type and Networking
Authentication Method
Accounts and Roles
Windows Service
Apply Configuration
Apply Configuration
Click [Execute] to apply the changes
Configuration Steps
Log
Writing configuration file
Updating Windows Firewall rules
Adjusting Windows service
Initializing database (may take a long time)
Starting the server
Applying security settings
Updating the Start menu link
< Back
Execute
Cancel


C:\Program Files\MySQL\MySQL Shell 8.0\bin\mysqlsh.exe
MySQL Shell 8.0.29
Copyright (c) 2016, 2022, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.
Type '\help' or '\?' for help; '\quit' to exit.
MySQL JS >


This PC › Local Disk (C:) › Program Files › MySQL › MySQL Server 8.0 › bin
Name
Date modified
Type
Size
fido2.dll 23-03-2022 15:21 Application exten... 146 KB
fido2.lib 23-03-2022 15:21 LIB File 45 KB
harness-library.dll 23-03-2022 15:22 Application exten... 752 KB
ibd2sdi 23-03-2022 08:57 Application 6,260 KB
innochecksum 23-03-2022 08:57 Application 6,248 KB
jemalloc.dll 08-12-2021 12:30 Application exten... 263 KB
libcrypto-1_1-x64.dll 16-03-2022 10:42 Application exten... 3,361 KB
libmecab.dll 19-11-2021 20:03 Application exten... 1,797 KB
libprotobuf.dll 23-03-2022 15:21 Application exten... 2,780 KB
libprotobuf.lib 23-03-2022 15:21 LIB File 3,883 KB
libprotobuf-debug.dll 23-03-2022 15:04 Application exten... 6,094 KB
libprotobuf-lite.dll 23-03-2022 15:21 Application exten... 585 KB
libprotobuf-lite.lib 23-03-2022 15:21 LIB File 922 KB
libprotobuf-lite-debug.dll 23-03-2022 15:04 Application exten... 1,367 KB
libssl-1_1-x64.dll 16-03-2022 10:42 Application exten... 672 KB
lz4_decompress 23-03-2022 08:57 Application 6,192 KB
my_print_defaults 23-03-2022 08:58 Application 6,128 KB
myisam_ftdump 23-03-2022 08:57 Application 6,383 KB
myisamchk 23-03-2022 08:57 Application 6,504 KB
myisamlog 23-03-2022 08:57 Application 6,348 KB
myisampack 23-03-2022 08:57 Application 6,404 KB
mysql 23-03-2022 08:57 Application 7,142 KB
mysql_config_editor 23-03-2022 08:58 Application 6,143 KB
mysql_migrate_keyring 23-03-2022 08:58 Application 7,112 KB
mysql_secure_installation 23-03-2022 08:58 Application 7,019 KB
em selected 145 KB
Activate W
Go to Settings


Command Prompt
Microsoft Windows [Version 10.0.19043.1645]
(c) Microsoft Corporation. All rights reserved.
C:\Users\Teja>cd C:\Program Files\MySQL\MySQL Server 8.0\bin


Command Prompt - mysql -u root -p
Microsoft Windows [Version 10.0.19043.1645]
(c) Microsoft Corporation. All rights reserved.
C:\Users\Teja>cd C:\Program Files\MySQL\MySQL Server 8.0\bin
C:\Program Files\MySQL\MySQL Server 8.0\bin>mysql -u root -p
Enter password:



USING COMMAND PROMPT

```
C:\Program Files\MySQL\MySQL Server 8.0\bin>mysql -u root -p
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 25
Server version: 8.0.25 MySQL Community Server - GPL

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

How to create and use database

- Display existing databases
  - SHOW DATABASES;
- Create a new database
  - CREATE DATABASE SCHOOL;
- Use a database
  - USE SCHOOL
- Delete the database
  - DROP DATABASE SCHOOL

DATA TYPES

Numeric Types

- INT
- SMALLINT
- TINYINT
- MEDIUMINT
- BIGINT
- DECIMAL
- NUMERIC
- FLOAT
- DOUBLE
- BIT

String Types

- CHAR
- VARCHAR
- BINARY
- VARBINARY
- BLOB
- TINYBLOB
- MEDIUMBLOB
- LONGBLOB
- TEXT
- TINYTEXT
- MEDIUMTEXT
- LONGTEXT
- ENUM

Date Types

- DATE
- DATETIME
- TIMESTAMP
- TIME

CHAR	VARCHAR
Used to store strings of fixed size	Used to store strings of variable length
Can range in size from 1 to 8000 bytes	Can range in size from 1 to 8000 bytes
Uses a fixed amount of storage, based on the size of the column	Use varying amounts of storage space based on the size of the string stored.
Takes up 1 to 4 byte for each character, based on collation setting	Takes up 1 to 4 byte for each character based on collation and requires one or more bytes to store the length of the data
Better performance	Slightly poorer performance because length has to be accounted for.
Pads spaces to the right when storing strings less than the fixed size length	No padding necessary because it is variable in size

ONLY FOR PRACTICE NOT FOR REMEMBERING

Type	Length in Bytes	Minimum Value (Signed)	Maximum Value (Signed)	Minimum Value (Unsigned)	Maximum Value (Unsigned)
TINYINT	1	-128	127	0	255
SMALLINT	2	-32768	32767	0	65535
MEDIUMINT	3	-8388608	8388607 to	0	16777215
INT	4	-2147483648	2147483647	0	4294967295
BIGINT	8	-9223372036854775808	9223372036854775807	0	18446744073709551615

CONSTRAINTS IN SQL

- **NOT NULL**: This constraint tells that we cannot store a null value in a column. That is, if a column is specified as NOT NULL then we will not be able to store null in this particular column any more.
- **UNIQUE**: This constraint when specified with a column, tells that all the values in the column must be unique. That is, the values in any row of a column must not be repeated.
- **PRIMARY KEY**: A primary key is a field which can uniquely identify each row in a table. And this constraint is used to specify a field in a table as primary key.
- **FOREIGN KEY**: A Foreign key is a field which can uniquely identify each row in a another table. And this constraint is used to specify a field as Foreign key.
- **CHECK**: This constraint helps to validate the values of a column to meet a particular condition. That is, it helps to ensure that the value stored in a column meets a specific condition.
- **DEFAULT**: This constraint specifies a default value for the column when no value is specified by the user.

UNIQUE CONSTRAINT

- The UNIQUE Constraint is useful to restrict storing of duplicate data row value in a given column.
- Multiple columns in a single table can have the UNIQUE Constraint.
- The PRIMARY KEY automatically sets UNIQUE Constraint for that column.
- Examples:

```
ALTER TABLE tablename ADD UNIQUE (column1);
ALTER TABLE tablename DROP INDEX column1;
```

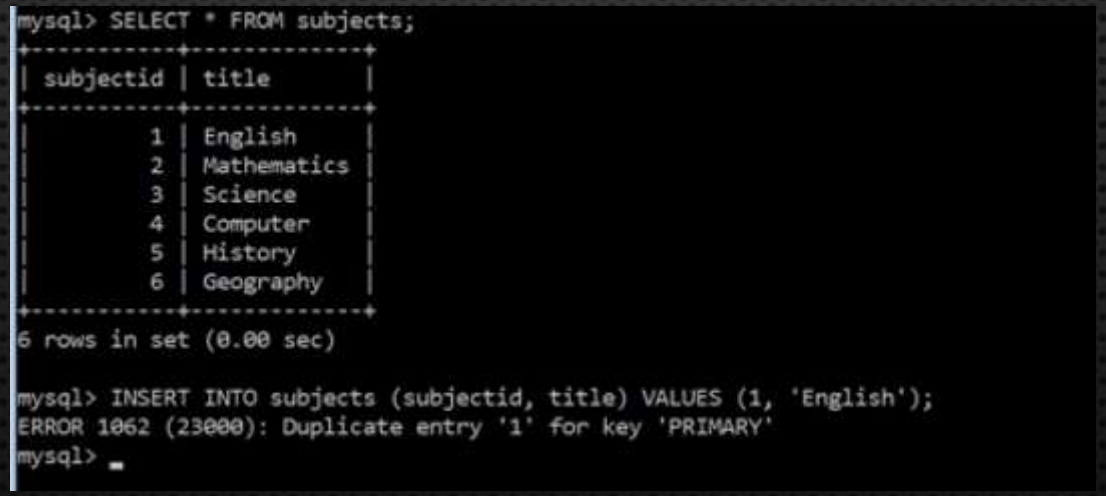
```
mysql> ALTER TABLE subjects MODIFY COLUMN title VARCHAR(50) UNIQUE;
Query OK, 0 rows affected (0.65 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> DESCRIBE subjects;
+-----------+-------------+------+-----+---------+----------------+
| Field     | Type        | Null | Key | Default | Extra          |
+-----------+-------------+------+-----+---------+----------------+
| subjectid | int(11)     | NO   | PRI | NULL    | auto_increment |
| title     | varchar(50) | YES  | UNI | NULL    |                |
+-----------+-------------+------+-----+---------+----------------+
2 rows in set (0.00 sec)

mysql> INSERT INTO subjects (title) VALUES ('English');
ERROR 1062 (23000): Duplicate entry 'English' for key 'title'
```

```
mysql> INSERT INTO subjects (title) VALUES ('English');
ERROR 1062 (23000): Duplicate entry 'English' for key 'title'
mysql> ALTER TABLE subjects DROP INDEX title;
Query OK, 0 rows affected (0.12 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> DESCRIBE subjects;
+-----------+-------------+------+-----+---------+----------------+
| Field     | Type        | Null | Key | Default | Extra          |
+-----------+-------------+------+-----+---------+----------------+
| subjectid | int(11)     | NO   | PRI | NULL    | auto_increment |
| title     | varchar(50) | YES  |     | NULL    |                |
+-----------+-------------+------+-----+---------+----------------+
2 rows in set (0.00 sec)
```

PRIMARY KEY

- The PRIMARY KEY Constraint is useful to restrict storing of duplicate data row value in a given column.
- The PRIMARY KEY is similar to UNIQUE Constraint but unlike UNIQUE Constraint there can be only one PRIMARY KEY for one table.
- The PRIMARY KEY automatically sets UNIQUE Constraint for that table column.
- The PRIMARY KEY column can not contain NULL values
- The primary key can be defined while creating a new database table or can be added by using ALTER TABLE statement.
- Example:

```
ALTER TABLE tablename ADD PRIMARY KEY (column1);
ALTER TABLE tablename DROP PRIMARY KEY column1;
```


mysql> SELECT * FROM subjects;
+-----------+-------------+
| subjectid | title |
+-----------+-------------+
1	English
2	Mathematics
3	Science
4	Computer
5	History
6	Geography
+-----------+-------------+
6 rows in set (0.00 sec)
mysql> INSERT INTO subjects (subjectid, title) VALUES (1, 'English');
ERROR 1062 (23000): Duplicate entry '1' for key 'PRIMARY'
mysql>

FOREIGN KEY

- A FOREIGN KEY in a table always points to a PRIMARY KEY in another table.
- The FOREIGN KEY constraint terminates the actions that can destroy the links between relational tables.
- The FOREIGN KEY constraint also restrict adding invalid data to the foreign key column because the foreign key column values must be similar to the linked primary key values.
- The FOREIGN KEY can be defined while creating a new database table or can be added by using ALTER TABLE statement.
- Example:

```
ALTER TABLE tablename1 ADD CONSTRAINT fk_column1 FOREIGN KEY (column1) REFERENCES tablename2(column1);

ALTER TABLE tablename1 DROP FOREIGN KEY fk_column1;
```

PRIMARY KEY

subjectid	title
1	English
2	Mathematics
3	Science

Table: subjects

FOREIGN KEY

teacherid	name	phone	subjectid
1	John Doe	1234567890	3
2	Caroline Smith	0987654321	1
3	Jason King	1234512345	1

Table: teachers

- ALTER TABLE teachers ADD CONSTRAINT fk_subjectid FOREIGN KEY (subjectid) REFERENCES subjects(subjectid);
- ALTER TABLE teachers DROP FOREIGN KEY fk_subjectid;

- The column 'subjectid' in teachers table points to the 'subjectid' in 'subjects' table.

- The column 'subjectid' in 'teachers' table is the PRIMARY KEY in 'teachers' table While The column 'subjectid' in 'subjects' table is a FOREIGN KEY in the 'subjects' table.

TABLE

```
mysql> SELECT * FROM subjects;
+-----------+-------------+
| subjectid | title       |
+-----------+-------------+
|         1 | English     |
|         2 | Mathematics |
|         3 | Science     |
|         4 | Computer    |
|         5 | History     |
|         6 | Geography   |
+-----------+-------------+
6 rows in set (0.00 sec)

mysql> SELECT * FROM teachers;
+-----------+----------------+------------+
| teacherid | name           | phone      |
+-----------+----------------+------------+
|         1 | John Doe       | 1234567890 |
|         2 | Caroline Smith | 0987654321 |
|         3 | Jason King     | 1234512345 |
|         4 | Stella Brown   | 0987612345 |
|         5 | Eric Hall      | 0987609876 |
|         6 | Peter Brown    | 1234509876 |
+-----------+----------------+------------+
6 rows in set (0.05 sec)
```

```
mysql> ALTER TABLE teachers ADD COLUMN subjectid INT(11) NOT NULL DEFAULT 1;
Query OK, 0 rows affected (0.55 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> DESCRIBE teachers;
+-----------+------------+------+-----+---------+----------------+
| Field     | Type       | Null | Key | Default | Extra          |
+-----------+------------+------+-----+---------+----------------+
| teacherid | int(11)    | NO   | PRI | NULL    | auto_increment |
| name      | varchar(80)| NO   |     | NULL    |                |
| phone     | varchar(10)| YES  |     | NULL    |                |
| subjectid | int(11)    | NO   |     | 1       |                |
+-----------+------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

AUTO INCREMENT

- The AUTO_INCREMENT attribute **generates and save a unique number** every time a new data row is inserted into a table.
- For PRIMARY KEY column we always require a unique number to be stored. Instead of creating a unique number manually, AUTO_INCREMENT or SEQUENCE attribute are useful.
- The AUTO INCREMENT or SEQUENCE are only used with numeric data columns.
- In MySQL by default AUTO_INCREMENT generates new value by adding 1, for each new record.

CONTD..

- MySQL

```
CREATE TABLE tablename ( column1 datatype NOT NULL PRIMARY KEY AUTO_INCREMENT, column2 datatype );
```

- SQL Server

```
CREATE TABLE tablename ( column1 datatype IDENTITY(1,1) PRIMARY KEY, column2 datatype );
```

- MS Access

```
CREATE TABLE tablename ( column1 datatype PRIMARY KEY AUTOINCREMENT, column2 datatype );
```

- Oracle

```
CREATE SEQUENCE column1
MINVALUE 1
START WITH 1
INCREMENT BY 1
CACHE 50
```

NULL

- Null is nothing but the value ‘is not known’.
- doesn’t mean it’s ‘0’

NOT NULL , PRIMARY KEY & UNIQUE

```
mysql> CREATE TABLE CONSTRAINTS_EXAMPLE (ID TINYINT NOT NULL UNIQUE,PRIMARY KEY(ID));
Query OK, 0 rows affected (1.89 sec)
```

```
mysql> CREATE TABLE CUSTOMERS
    -> (
    -> CID TINYINT NOT NULL UNIQUE AUTO_INCREMENT,
    -> NAME VARCHAR(30) NOT NULL,
    -> ADDRESS VARCHAR(30) NOT NULL,
    -> PRIMARY KEY (CID)
    -> );
Query OK, 0 rows affected (1.61 sec)

mysql> INSERT INTO CUSTOMERS(NAME,ADDRESS) VALUES
    -> ('RAJU','HYD'),
    -> ('RAVI','VIZAG'),
    -> ('RAMESH','VIJAYAWADA');
Query OK, 3 rows affected (0.27 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

```
mysql> CREATE TABLE ORDERS
    -> (
    -> OID INT NOT NULL UNIQUE AUTO_INCREMENT,
    -> O_NO INT NOT NULL,
    -> CID TINYINT,
    -> PRIMARY KEY(OID),
    -> FOREIGN KEY(CID) REFERENCES CUSTOMERS(CID)
    -> );
Query OK, 0 rows affected (2.21 sec)
```

```
mysql> INSERT INTO  ORDERS(O_NO,CID) VALUES
    -> (3424, 2),
    -> (3478,3),
    -> (9384,1);
Query OK, 3 rows affected (0.16 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

CHECK

```
CREATE TABLE Student
(
ID int(6) NOT NULL,
NAME varchar(10) NOT NULL,
AGE int NOT NULL CHECK (AGE >= 18)
);
```

DEFAULT

```
CREATE TABLE Student
(
ID int(6) NOT NULL,
NAME varchar(10) NOT NULL,
AGE int DEFAULT 18
);
```

To Set Default Values

```
CREATE TABLE cats3
  (
     name VARCHAR(100) DEFAULT 'unnamed',
     age  INT DEFAULT 99
  );
```

```
mysql>
mysql> CREATE TABLE cats3
    -> (
    ->   name VARCHAR(20) DEFAULT 'no name provided',
    ->   age INT DEFAULT 99
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql> DESC cats3;
+-------+-------------+------+-----+-------------------+-------+
| Field | Type        | Null | Key | Default           | Extra |
+-------+-------------+------+-----+-------------------+-------+
| name  | varchar(20) | YES  |     | no name provided  |       |
| age   | int(11)     | YES  |     | 99                |       |
+-------+-------------+------+-----+-------------------+-------+
2 rows in set (0.00 sec)
```

DATE TIME

```
mysql> create table datetime_example(transaction datetime);
Query OK, 0 rows affected (0.70 sec)

mysql> insert into datetime_example values(now());
Query OK, 1 row affected (0.19 sec)

mysql> select * from datetime_example;
+---------------------+
| transaction         |
+---------------------+
| 2022-06-23 09:07:00 |
+---------------------+
1 row in set (0.00 sec)
```

```
mysql> create table manual_datetime_example(transaction datetime);
Query OK, 0 rows affected (0.83 sec)

mysql> insert into manual_datetime_example values('2022-06-23 09-10-30');
Query OK, 1 row affected, 1 warning (0.20 sec)

mysql> select * from manual_datetime_example;
+---------------------+
| transaction         |
+---------------------+
| 2022-06-23 09:10:30 |
+---------------------+
1 row in set (0.00 sec)
```

Ac

```
mysql> CREATE TABLE DEPARTMENTS
    -> (
    -> ID INT NOT NULL UNIQUE AUTO_INCREMENT,
    -> NAME VARCHAR(30) NOT NULL,
    -> PRIMARY KEY(ID)
    -> );
Query OK, 0 rows affected (1.32 sec)

mysql> CREATE TABLE EMPLOYEES
    -> (
    -> ID INT NOT NULL UNIQUE AUTO_INCREMENT,
    -> FNAME VARCHAR(30) NOT NULL,
    -> LNAME VARCHAR(30) NOT NULL,
    -> MOBILE BIGINT NOT NULL ,
    -> EMAIL VARCHAR(30) NOT NULL,
    -> ADDRESS VARCHAR(30) NOT NULL,
    -> DEPARTMEN_ID INT NOT NULL,
    -> PRIMARY KEY(ID),
    -> FOREIGN KEY (DEPARTMEN_ID) REFERENCES DEPARTMENTS(ID)
    -> );
Query OK, 0 rows affected (1.45 sec)
```

DEPARTMENTS

Departments

Id	Name
1	HR
2	Sales
3	Tech

```
mysql> CREATE TABLE DEPARTMENTS
    -> (
    ->   ID INT NOT NULL AUTO_INCREMENT,
    ->   NAME VARCHAR(25) NOT NULL,
    ->   PRIMARY KEY(ID)
    -> );
Query OK, 0 rows affected (0.50 sec)

mysql> INSERT INTO DEPARTMENTS (ID,NAME) VALUES
    -> (1,'HR'),
    -> (2,'SALES'),
    -> (3,'TECH')
    -> ;
Query OK, 3 rows affected (0.09 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM DEPARTMENTS;
+----+--------+
| ID | NAME   |
+----+--------+
|  1 | HR     |
|  2 | SALES  |
|  3 | TECH   |
+----+--------+
```

EMPLOYEES

Employees

Id	FName	LName	PhoneNumber	ManagerId	DepartmentId	Salary	HireDate
1	James	Smith	1234567890	NULL	1	1000	01-01-2002
2	John	Johnson	2468101214	1	1	400	23-03-2005
3	Michael	Williams	1357911131	1	2	600	12-05-2009
4	Johnathon	Smith	1212121212	2	1	500	24-07-2016

```
mysql> CREATE TABLE Employees (
    -> Id INT NOT NULL AUTO_INCREMENT,
    -> FName VARCHAR(35) NOT NULL,
    -> LName VARCHAR(35) NOT NULL,
    -> PhoneNumber VARCHAR(11),
    -> ManagerId INT,
    -> DepartmentId INT NOT NULL,
    -> Salary INT NOT NULL,
    -> HireDate DATETIME NOT NULL,
    -> PRIMARY KEY(Id),
    -> FOREIGN KEY (ManagerId) REFERENCES Employees(Id),
    -> FOREIGN KEY (DepartmentId) REFERENCES Departments(Id)
    -> );
Query OK, 0 rows affected (1.05 sec)
```

```
mysql>  INSERT INTO EMPLOYEES
    ->  (ID,FNAME,LNAME,PHONENUMBER,MANAGERID,DEPARTMENTID,SALARY,HIREDATE)
    ->  VALUES
    -> (2, 'John', 'Johnson', 2468101214, '1', 1, 400,'2021-03-10'),
    -> (3, 'Michael', 'Williams', 1357911131, '1', 2, 600, '2019-12-05'),
    -> (4, 'Johnathon', 'Smith', 1212121212, '2', 1, 500, '2020-07-21')
    -> ;
Query OK, 3 rows affected (0.18 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM EMPLOYEES;
```

CUSTOMERS

Customers

Id	FName	LName	Email	PhoneNumber	PreferredContact
1	William	Jones	william.jones@example.com	3347927472	PHONE
2	David	Miller	dmiller@example.net	2137921892	EMAIL
3	Richard	Davis	richard0123@example.com	NULL	EMAIL

SQL statements to create the table:

```
mysql> CREATE TABLE Customers (
    -> Id INT NOT NULL AUTO_INCREMENT,
    -> FName VARCHAR(35) NOT NULL,
    -> LName VARCHAR(35) NOT NULL,
    -> Email varchar(100) NOT NULL,
    -> PhoneNumber VARCHAR(11),
    -> PreferredContact VARCHAR(5) NOT NULL,
    -> PRIMARY KEY(Id)
    -> );
Query OK, 0 rows affected (1.33 sec)

mysql> INSERT INTO CUSTOMERS
    -> (ID,FNAME,LNAME,EMAIL,PHONENUMBER,PREFERREDCONTACT)
    -> VALUES
    -> (1, 'William', 'Jones', 'william.jones@example.com', '3347927472', 'PHONE'),
    -> (2, 'David', 'Miller', 'dmiller@example.net', '2137921892', 'EMAIL'),
    -> (3, 'Richard', 'Davis', 'richard0123@example.com', NULL, 'EMAIL')
    -> ;
Query OK, 3 rows affected (0.06 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

CARS

Cars

Id	CustomerId	EmployeeId	Model	Status	Total Cost
1	1	2	Ford F-150	READY	230
2	1	2	Ford F-150	READY	200
3	2	1	Ford Mustang	WAITING	100
4	3	3	Toyota Prius	WORKING	1254

```
mysql> CREATE TABLE Cars (
    -> Id INT NOT NULL AUTO_INCREMENT,
    -> CustomerId INT NOT NULL,
    -> EmployeeId INT NOT NULL,
    -> Model varchar(50) NOT NULL,
    -> Status varchar(25) NOT NULL,
    -> TotalCost INT NOT NULL,
    -> PRIMARY KEY(Id),
    -> FOREIGN KEY (CustomerId) REFERENCES Customers(Id),
    -> FOREIGN KEY (EmployeeId) REFERENCES Employees(Id)
    -> );
Query OK, 0 rows affected (0.94 sec)
```

```
mysql> INSERT INTO CARS
    -> (ID,CUSTOMERID,EMPLOYEEID,MODEL,STATUS,TOTALCOST)
    -> VALUES
    -> ('1', '1', '2', 'Ford F-150', 'READY', '230'),
    -> ('2', '1', '2', 'Ford F-150', 'READY', '200'),
    -> ('3', '2', '1', 'Ford Mustang', 'WAITING', '100'),
    -> ('4', '3', '3', 'Toyota Prius', 'WORKING', '1254')
    -> ;
Query OK, 4 rows affected (0.09 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

AUTHORS

Id	Name	Country
1	J.D. Salinger	USA
2	F. Scott. Fitzgerald	USA
3	Jane Austen	UK
4	Scott Hanselman	USA
5	Jason N. Gaylord	USA
6	Pranav Rastogi	India
7	Todd Miranda	USA
8	Christian Wenz	USA

```
CREATE TABLE Authors (
    Id INT NOT NULL AUTO_INCREMENT,
    Name VARCHAR(70) NOT NULL,
    Country VARCHAR(100) NOT NULL,
    PRIMARY KEY(Id)
);
```

BOOKS

Id	Title
1	The Catcher in the Rye
2	Nine Stories
3	Franny and Zooey
4	The Great Gatsby
5	Tender id the Night
6	Pride and Prejudice
7	Professional ASP.NET 4.5 in C# and VB

SQL to create the table:

```
CREATE TABLE Books (
    Id INT NOT NULL AUTO_INCREMENT,
    Title VARCHAR(50) NOT NULL,
    PRIMARY KEY(Id)
);
```

BOOKSAUTHORS

```sql
CREATE TABLE BooksAuthors (
    AuthorId INT NOT NULL,
    BookId  INT NOT NULL,
    FOREIGN KEY (AuthorId) REFERENCES Authors(Id),
    FOREIGN KEY (BookId) REFERENCES Books(Id)
);

INSERT INTO BooksAuthors
    (BookId, AuthorId)
VALUES
    (1, 1),
    (2, 1),
    (3, 1),
    (4, 2),
    (5, 2),
    (6, 3),
    (7, 4),
    (7, 5),
    (7, 6),
    (7, 7),
    (7, 8)
;
```

```sql
SELECT
  ba.AuthorId,
  a.Name AuthorName,
  ba.BookId,
  b.Title BookTitle
FROM BooksAuthors ba
  INNER JOIN Authors a ON a.id = ba.authorid
  INNER JOIN Books b ON b.id = ba.bookid
;
```

DDL & DML COMMANDS

DATA DEFINITION LANGUAGE

- Manage database objects like tables and columns.
- Changes the database structure.
- Tables and columns, are created, modified or removed.
- CREATE, ALTER, DROP

DATA MANIPULATION LANGUAGE

- Manage the data that resides in our tables and columns.
- Changes only the data.
- Data inside a table is inserted, updated or deleted.
- INSERT, UPDATE, and DELETE

UPDATE ROWS IN A TABLE

- UPDATE STUDENT SET NAME= ‘ LAKSHMAN’, MAJOR=‘CE’ WHERE ROLL= 4;

UPDATE

- UPDATE statement is **used to update the data rows** in a table.
- The UPDATE statement can update one or multiple column values in single SQL statement.
- WHERE clause is used to specify the data row to be updated.
- Any existing data rows or records in target table remain unaffected.
- Example:

```
UPDATE tablename SET column1 = newvalue;
UPDATE tablename SET column1 = newvalue1, column2 = newvalue2 WHERE columnid = 3;
```

EXAMPLE

```
mysql> UPDATE students SET class = 'Second' WHERE studentid = 1;
Query OK, 1 row affected (0.10 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
mysql> UPDATE employees SET jobtitle = 'Application Developer', salary = 3800 WHERE employeeid = 6;
Query OK, 1 row affected (0.05 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

ADD NEW COLUMN

```
mysql> alter table student add gender char ;
Query OK, 0 rows affected (0.38 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> explain student;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| STUDENTNAME   | varchar(20) | YES  |     | NULL    |       |
| STUDENTSECTION | char(1)    | YES  |     | NULL    |       |
| STUDENTROLL   | int         | YES  |     | NULL    |       |
| STUDENTMARKS  | float       | YES  |     | NULL    |       |
| STUDENTAVG    | double      | YES  |     | NULL    |       |
| gender        | char(1)     | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

DELETE

- DELETE statement is **used to delete the data rows** in a table.
- The DELETE statement can delete one or multiple data rows from a table.
- WHERE clause is used to specify the data row to be deleted.
- Example:

```
DELETE FROM tablename;
DELETE FROM tablename WHERE column1 = value;
```

EXAMPLE

```
mysql> DELETE FROM students2 WHERE studentid = 60;
Query OK, 1 row affected (0.06 sec)

mysql> SELECT * FROM students2;
```

```
mysql> DELETE FROM students2;
Query OK, 58 rows affected (0.09 sec)

mysql> SELECT * FROM students2;
Empty set (0.00 sec)
```

```
mysql> update employees set lname='graham' where salary=500;
Query OK, 1 row affected (0.40 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from employees;
+----+----------+---------+-------------+-----------+--------------+--------+------(
----------+
| Id | FName    | LName   | PhoneNumber | ManagerId | DepartmentId | Salary | HireD(
          |
+----+----------+---------+-------------+-----------+--------------+--------+------(
----------+
|  1 | James    | Smith   | 1234567890  |      NULL |            1 |   1000 | 2002-(
 00:00:00 |
|  2 | James    | Smith   | 1234567890  |      NULL |            1 |   1000 | 2002--
 00:00:00 |
|  3 | Michael  | Williams | 1357911131 |         1 |            2 |    600 | 2009-a
 00:00:00 |
|  4 | Johnathon | graham | 1212121212  |         2 |            1 |    500 | 2016-(
 00:00:00 |
+----+----------+---------+-------------+-----------+--------------+--------+------
----------+
4 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM STUDENT;
+--------+--------+-----------+------+---------------------+-----------+
| ROLLNO | NAME   | MOBILE    | CGPA | ADMISSION           | activites |
+--------+--------+-----------+------+---------------------+-----------+
|      1 | RAJU   | 438934985 |  7.4 | 2022-06-23 10:23:44 | swimming  |
|      2 | RAMU   | 348934985 |  5.4 | 2012-06-23 09:23:44 | NULL      |
|      3 | PAVAN  | 438778548 |    7 | 2012-07-07 00:00:00 | NULL      |
|      4 | SAGAR  |  49589554 |    9 | 2013-10-07 00:00:00 | NULL      |
|      5 | VENKAT | 894598489 |  9.6 | 2015-07-05 00:00:00 | NULL      |
|   NULL | NULL   |      NULL | NULL | NULL                | NULL      |
+--------+--------+-----------+------+---------------------+-----------+
6 rows in set (0.00 sec)

mysql> UPDATE STUDENT SET ACTIVITES='GYMNASTICS' WHERE NAME='RAMU'OR NAME='PAVAN';
Query OK, 2 rows affected (0.08 sec)
Rows matched: 2  Changed: 2  Warnings: 0

mysql> SELECT * FROM STUDENT;
+--------+--------+-----------+------+---------------------+------------+
| ROLLNO | NAME   | MOBILE    | CGPA | ADMISSION           | activites  |
+--------+--------+-----------+------+---------------------+------------+
|      1 | RAJU   | 438934985 |  7.4 | 2022-06-23 10:23:44 | swimming   |
|      2 | RAMU   | 348934985 |  5.4 | 2012-06-23 09:23:44 | GYMNASTICS |
|      3 | PAVAN  | 438778548 |    7 | 2012-07-07 00:00:00 | GYMNASTICS |
|      4 | SAGAR  |  49589554 |    9 | 2013-10-07 00:00:00 | NULL       |
|      5 | VENKAT | 894598489 |  9.6 | 2015-07-05 00:00:00 | NULL       |
|   NULL | NULL   |      NULL | NULL | NULL                | NULL       |
+--------+--------+-----------+------+---------------------+------------+
6 rows in set (0.00 sec)
```

```
mysql> select * from student;
+--------+-------+-----------+------+---------------------+-----------+
| ROLLNO | NAME  | MOBILE    | CGPA | ADMISSION           | activites |
+--------+-------+-----------+------+---------------------+-----------+
|      1 | RAJU  | 438934985 |  7.4 | 2022-06-23 10:23:44 | swimming  |
|      4 | SAGAR |  49589554 |    9 | 2013-10-07 00:00:00 | ATHLETS   |
|   NULL | NULL  |      NULL | NULL | NULL                | NULL      |
+--------+-------+-----------+------+---------------------+-----------+
3 rows in set (0.00 sec)

mysql> delete from student where rollno is null;
Query OK, 1 row affected (0.10 sec)

mysql> select * from student;
+--------+-------+-----------+------+---------------------+-----------+
| ROLLNO | NAME  | MOBILE    | CGPA | ADMISSION           | activites |
+--------+-------+-----------+------+---------------------+-----------+
|      1 | RAJU  | 438934985 |  7.4 | 2022-06-23 10:23:44 | swimming  |
|      4 | SAGAR |  49589554 |    9 | 2013-10-07 00:00:00 | ATHLETS   |
+--------+-------+-----------+------+---------------------+-----------+
2 rows in set (0.00 sec)
```

```
+--------+-------+-----------+------+---------------------+-----------+
| ROLLNO | NAME  | MOBILE    | CGPA | ADMISSION           | activites |
+--------+-------+-----------+------+---------------------+-----------+
|      1 | RAJU  | 438934985 |  7.4 | 2022-06-23 10:23:44 | swimming  |
|      4 | SAGAR |  49589554 |    9 | 2013-10-07 00:00:00 | ATHLETS   |
+--------+-------+-----------+------+---------------------+-----------+
2 rows in set (0.00 sec)

mysql> delete from student;
Query OK, 2 rows affected (0.17 sec)

mysql> select * from student;
Empty set (0.00 sec)

mysql> explain student;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| ROLLNO    | int         | YES  |     | NULL    |       |
| NAME      | varchar(30) | YES  |     | NULL    |       |
| MOBILE    | bigint      | YES  |     | NULL    |       |
| CGPA      | double      | YES  |     | NULL    |       |
| ADMISSION | datetime    | YES  |     | NULL    |       |
| activites | varchar(30) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
6 rows in set (0.05 sec)
```

DELETE A RECORD IN TABLE

- DELETE FROM STUDENT WHERE ROLL=1;

ALTER ADD

- Alter table student add dob date;
- Update student set dob='1999-04-05' where roll=1;

```
mysql> alter table student add activites varchar(30);
Query OK, 0 rows affected (1.75 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> explain student;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| ROLLNO    | int         | YES  |     | NULL    |       |
| NAME      | varchar(30) | YES  |     | NULL    |       |
| MOBILE    | bigint      | YES  |     | NULL    |       |
| CGPA      | double      | YES  |     | NULL    |       |
| ADMISSION | datetime    | YES  |     | NULL    |       |
| activites | varchar(30) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
6 rows in set (0.29 sec)
```

```
mysql> ALTER TABLE EXAMPLE MODIFY MOBILENO BIGINT;
Query OK, 4 rows affected (2.27 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> DESCRIBE EXAMPLE;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| slno     | int         | YES  |     | NULL    |       |
| sname    | varchar(30) | YES  |     | NULL    |       |
| address  | varchar(30) | YES  |     | NULL    |       |
| MOBILENO | bigint      | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql> ALTER TABLE EXAMPLE DROP COLUMN ADDRESS;
Query OK, 0 rows affected (0.28 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
mysql> create table x
    -> (
    -> xname varchar(30),
    -> xno int not null
    -> );
Query OK, 0 rows affected (1.37 sec)

mysql> alter table x add primary key(xno);
Query OK, 0 rows affected (2.80 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> describe x;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| xname | varchar(30) | YES  |     | NULL    |       |
| xno   | int         | NO   | PRI | NULL    |       |
+-------+-------------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

```
mysql> ALTER TABLE EXAMPLE DROP COLUMN ADDRESS;
Query OK, 0 rows affected (0.28 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM EXAMPLE;
+------+---------+-----------+
| slno | sname   | MOBILENO  |
+------+---------+-----------+
|    1 | krishna |  98463522 |
|    2 | rama    | 893897894 |
|    3 | raju    | 487849023 |
|    4 | ravi    |   9347843 |
+------+---------+-----------+
4 rows in set (0.00 sec)
```

ALTER TABLE NAME OR COLUMN NAME

```
mysql> alter table table1 rename tablex ;
Query OK, 0 rows affected (1.44 sec)
```

```
mysql> alter table tablex rename column sno to serial;
Query OK, 0 rows affected (0.88 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> select * from tablex;
+--------+
| serial |
+--------+
|     15 |
|     20 |
|     25 |
|     30 |
|     35 |
+--------+
5 rows in set (0.00 sec)
```

DROP A COLUMN

- ALTER TABLE PRODUCT DROP COLUMN DOB;

ALTER MODIFY

- To modify the data type of a column we use drop modify
- Alter table student modify column roll varchar(10);

WHAT IS A SCHEMA?

- A SCHEMA IS A LIST OF LOGICAL STRUCTURES OF DATA
- IT CONTAINS TABLES THAT STORE INFORMATION REQUIRED BY THE MYSQL SERVER AS IT RUNS. A BROAD CATEGORIZATION IS THAT THE MYSQL SCHEMA CONTAINS DATA DICTIONARY TABLES THAT STORE DATABASE OBJECT METADATA, AND SYSTEM TABLES USED FOR OTHER OPERATIONAL PURPOSES

```
mysql> TRUNCATE TABLE ORDERS;
Query OK, 0 rows affected (1.61 sec)

mysql> DESCRIBE ORDERS;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| OID    | int         | NO   | PRI | NULL    |       |
| ONAME  | varchar(34) | NO   |     | NULL    |       |
| OPRICE | int         | NO   |     | NULL    |       |
| OAGE   | int         | YES  |     | 27      |       |
+--------+-------------+------+-----+---------+-------+
4 rows in set (0.05 sec)
```

TCL(TRANSACTION CONTROL LANGUAGE) COMMANDS

- COMMIT : SAVES THE TRANSACTION
- ROLLBACK : TRANSACTION WILL BE ROLLED BACK
- SAVEPOINT : A SAVEPOINT IS **A POINT IN A TRANSACTION IN WHICH YOU CAN ROLL THE TRANSACTION BACK TO A CERTAIN POINT WITHOUT ROLLING BACK THE ENTIRE TRANSACTION**

NOTE

- Mysql uses autocommit.whatever the dml queries are executed they are default set to autocommit =1 . So whatever dml command we use those changes are permanent.
- But in oracle dml commands are not autocommit.That is in oracle autocommit=0. so whatever the commands we use those changes are not permanent
- So use autocommit=0 then execute the dml commands

```
mysql> create table codinghub
    -> (
    -> sid int not null primary key,
    -> sname varchar(20) not null
    -> );
Query OK, 0 rows affected (3.77 sec)

mysql> insert into codinghub (sid,sname) values
    -> (1,'murali'),
    -> (2,'krishna'),
    -> (3,'sudhakar'),
    -> (4,'pavan');
Query OK, 4 rows affected (0.11 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> set autocommit=0;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> delete from codinghub where sname='murali';
Query OK, 1 row affected (0.00 sec)

mysql> select * from codinghub;
+-----+----------+
| sid | sname    |
+-----+----------+
|   2 | sudhakar |
|   3 | krishna  |
|   4 | pavan    |
|   5 | venkat   |
+-----+----------+
4 rows in set (0.00 sec)

mysql> rollback;
Query OK, 0 rows affected (0.08 sec)

mysql> select * from codinghub;
+------+------------+
| sid  | sname      |
+------+------------+
|    1 | murali     |
|    2 | sudhakar   |
|    3 | krishna    |
|    4 | pavan      |
|    5 | venkat     |
+------+------------+
```

```
mysql> select * from codinghub;
+-----+----------+
| sid | sname    |
+-----+----------+
|   1 | murali   |
|   2 | sudhakar |
|   3 | krishna  |
|   4 | pavan    |
|   5 | venkat   |
|   6 | ram      |
+-----+----------+
6 rows in set (0.00 sec)

mysql> commit;
Query OK, 0 rows affected (0.07 sec)

mysql> insert into codinghub values(7,'raj');
Query OK, 1 row affected (0.04 sec)

mysql> commit;
Query OK, 0 rows affected (0.05 sec)
```

```
mysql> savepoint x;
Query OK, 0 rows affected (0.00 sec)

mysql> delete from codinghub where sid=7;
Query OK, 1 row affected (0.06 sec)

mysql> select * from codinghub;
+-----+----------+
| sid | sname    |
+-----+----------+
|   1 | murali   |
|   2 | sudhakar |
|   3 | krishna  |
|   4 | pavan    |
|   5 | venkat   |
|   6 | ram      |
+-----+----------+
6 rows in set (0.00 sec)

mysql> rollback to x;
Query OK, 0 rows affected (0.07 sec)
```

```
mysql> select * from codinghub;
+-----+----------+
| sid | sname    |
+-----+----------+
|   1 | murali   |
|   2 | sudhakar |
|   3 | krishna  |
|   4 | pavan    |
|   5 | venkat   |
|   6 | ram      |
|   7 | raj      |
+-----+----------+
7 rows in set (0.00 sec)
```

Operators

- Arithmetic operators
- Comparison operators
- Logical operators

ARITHMETIC OPERATORS

- Numeric operators include addition (+), subtraction (-), multiplication (*) and division (/).
- It can be directly used in the SQL statements or in combination with column values.
- Examples:

```
SELECT 10 + 15;
SELECT firstname, age + 10 FROM students;
```

CONTD..

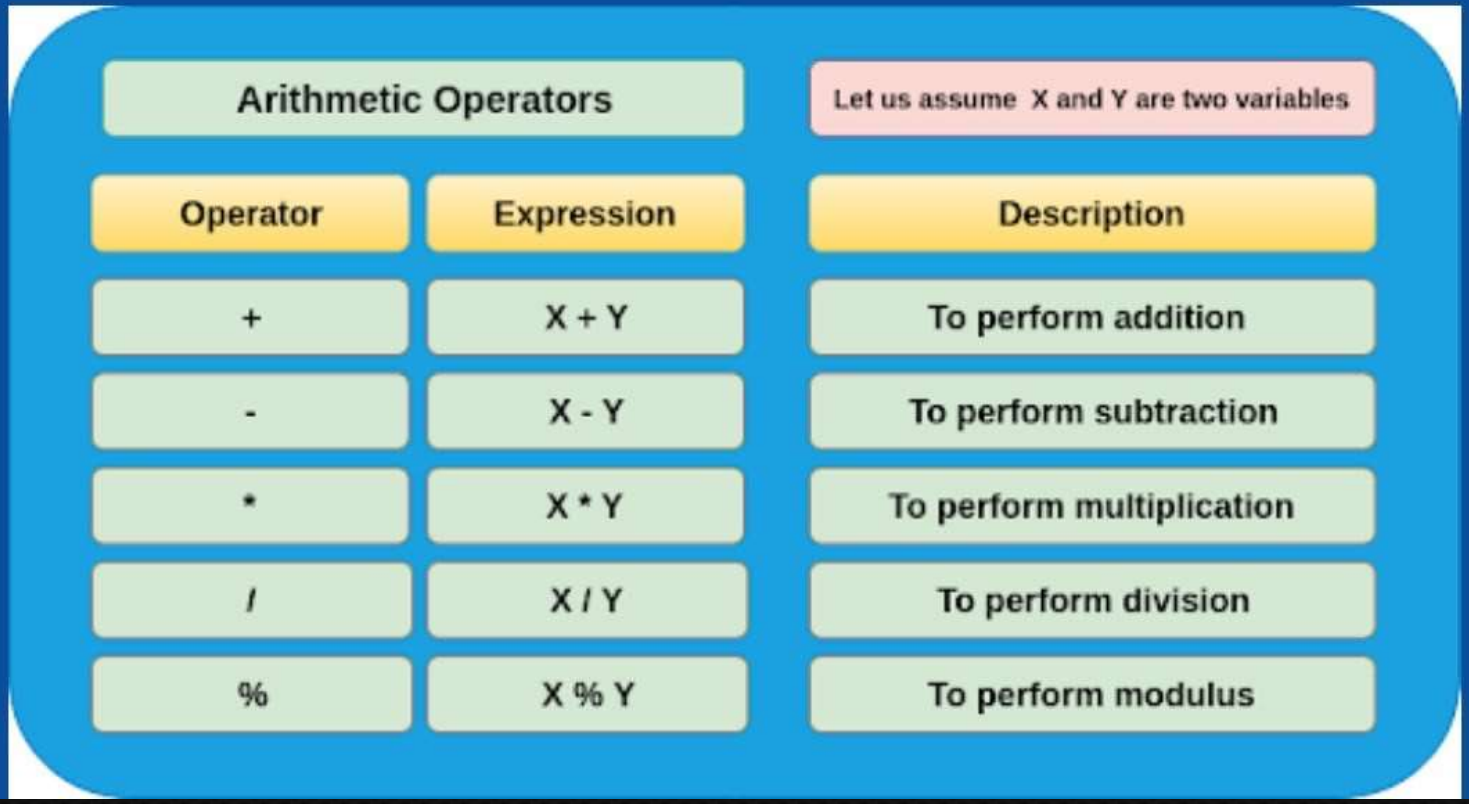
```
mysql> SELECT 15 * 10;
+---------+
| 15 * 10 |
+---------+
|     150 |
+---------+
1 row in set (0.00 sec)

mysql> SELECT 15 / 10;
+---------+
| 15 / 10 |
+---------+
|  1.5000 |
+---------+
1 row in set (0.00 sec)
```

```
mysql> SELECT firstname, age + 10 FROM students;
+-----------+----------+
| firstname | age + 10 |
+-----------+----------+
| John      |       15 |
| Mary      |       17 |
| James     |       15 |
| Mike      |       16 |
| Linda     |       17 |
| John      |       17 |
| James     |       16 |
| John      |       16 |
| Daniel    |       17 |
| Paul      |       15 |
| Mark      |       16 |
| Steven    |       17 |
| Brian     |       15 |
| Kevin     |       16 |
| Jason     |       15 |
| Jose      |       16 |
```

CONTD..

```
mysql> SELECT *, salary + 50 FROM employees;
+------------+---------------+-----------------------+--------+-------------+
| employeeid | name          | jobtitle              | salary | salary + 50 |
+------------+---------------+-----------------------+--------+-------------+
|          1 | John Doe      | Web Developer         |   3500 |        3550 |
|          2 | Mary Smith    | Web Developer         |   3500 |        3550 |
|          3 | James Brown   | Web Developer         |   3500 |        3550 |
|          4 | Mike Walker   | Web Developer         |   3500 |        3550 |
|          5 | Linda Jones   | Web Developer         |   3500 |        3550 |
|          6 | John Doe      | Systems Administrator |   3400 |        3450 |
|          7 | James Smith   | Systems Administrator |   3400 |        3450 |
|          8 | John Brown    | Systems Administrator |   3400 |        3450 |
|          9 | Daniel Jones  | Systems Administrator |   3400 |        3450 |
|         10 | Paul Anderson | Systems Administrator |   3400 |        3450 |
|         11 | Mark Davis    | IT Support Manager    |   3200 |        3250 |
|         12 | Steven Thomas | IT Support Manager    |   3200 |        3250 |
|         13 | Brian King    | IT Support Manager    |   3200 |        3250 |
|         14 | Kevin Hall    | IT Support Manager    |   3200 |        3250 |
|         15 | Jason Lee     | IT Support Manager    |   3200 |        3250 |
|         16 | Jose Young    | Database Administrator|   3700 |        3750 |
|         17 | Frank Smith   | Database Administrator|   3700 |        3750 |
|         18 | Eric Jones    | Database Administrator|   3700 |        3750 |
|         19 | Jerry Martin  | Database Administrator|   3700 |        3750 |
```

Arithmetic Operators in SQL



Arithmetic Operators		Let us assume X and Y are two variables
Operator	Expression	Description
+	X + Y	To perform addition
-	X - Y	To perform subtraction
*	X * Y	To perform multiplication
/	X / Y	To perform division
%	X % Y	To perform modulus

COMPARISON OPERATORS

- The comparison operators **compares two values** and determines whether the two values are equal or a value is greater than the other, or less than the other.
- The comparison operators can be applied to numbers, strings and dates.
- Examples:

```
SELECT * FROM students WHERE age < 6;
SELECT * FROM employees WHERE salary > 3500;
SELECT * FROM items WHERE price <= 30;
```

Symbols	Meaning
=	Equal to
<> Or !=	Not equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

LOGICAL OPERATORS

- AND = TWO CONDITIONS MUST BE TRUE
- OR = ANY ONE OF THE CONDITION HAS TO BE TRUE
- NOT =
- BETWEEN = PRINTS THE RANGE
- LIKE
- IN = ONLY SELECTED ROWS WILL BE DISPLAYED
- UNIQUE
- IS NULL

BETWEEN

- The BETWEEN Operator is used to **compare the value in a column**.
- The BETWEEN operator **tests a range of values** with a column value.
- The BETWEEN operator is written as a word "**BETWEEN**" followed by a range of the values.
- Examples:

```
SELECT * FROM students WHERE age BETWEEN 6 AND 10;
```

AND OPERATOR

- The AND operator is useful to **add multiple conditions** in a single SQL statement.
- It only **displays the data rows if all conditions are TRUE**.
- Written as a word '**AND**' or '**&&**' (2 and or ampersand symbols).
- SELECT * FROM students WHERE (class = 'first' AND age = 5);
- SELECT * FROM students WHERE class = 'first' && age = 5;

OR OPERATOR

- The OR operator is useful to **add multiple conditions** in a single SQL statement.
- It displays the data rows if **any one of the multiple conditions is TRUE**.
- Written as a word '**OR**' or '**||**' (2 pipe symbols).
- SELECT * FROM students WHERE (age = 5 OR age = 7);
- SELECT * FROM students WHERE age = 5 || age = 7;

IN CONDITION

- The "IN" operator **evaluates multiple values on a single data column**.
- It displays the data row if **any one of the given value is matched** with the data column value.
- Written as a word '**IN**' followed by multiple values separated by comma inside brackets.
- SELECT * FROM students WHERE age IN (5, 6);

CONTD..

```
mysql> SELECT * FROM employees WHERE salary NOT IN (3200, 3400, 3500, 3700);
+------------+-------------+----------------------+--------+
| employeeid | name        | jobtitle             | salary |
+------------+-------------+----------------------+--------+
|         21 | Henry Clark | Application Developer |   3800 |
|         22 | Carl White  | Application Developer |   3800 |
|         23 | Joe Thomas  | Application Developer |   3800 |
|         24 | Jack Smith  | Application Developer |   3800 |
|         25 | Roy King    | Application Developer |   3800 |
+------------+-------------+----------------------+--------+
```

LESS THAN

```
mysql> SELECT * FROM students WHERE age < 6;
+-----------+-----------+----------+--------+------+
| studentid | firstname | lastname | class  | age  |
+-----------+-----------+----------+--------+------+
|         1 | John      | Doe      | First  |    5 |
|         3 | James     | Brown    | First  |    5 |
|        10 | Paul      | Anderson | Third  |    5 |
|        13 | Brian     | King     | Second |    5 |
|        15 | Jason     | Lee      | Third  |    5 |
|        19 | Jerry     | Martin   | Third  |    5 |
|        22 | Carl      | White    | Second |    5 |
|        25 | Roy       | King     | Second |    5 |
|        29 | Jimmy     | Jones    | Third  |    5 |
|        30 | Emma      | Walker   | First  |    5 |
|        34 | Emily     | Walker   | Second |    5 |
|        35 | Grace     | King     | First  |    5 |
|        38 | Layla     | Young    | First  |    5 |
|        40 | Anna      | Jones    | Second |    5 |
|        42 | Camila    | Hall     | First  |    5 |
|        46 | Ellie     | King     | Third  |    5 |
|        48 | Stella    | White    | First  |    5 |
|        51 | Maya      | Brown    | First  |    5 |
|        52 | Faith     | Thomas   | Third  |    5 |
|        55 | Ashley    | Davis    | First  |    5 |
+-----------+-----------+----------+--------+------+
20 rows in set (0.00 sec)
```

LIKE (SQL WILD CARD CHARACTERS)

- A wildcard character is used as a **substitute for any other characters** in a string.
- The wildcards are **used to search for data** in a given column.
- The wildcard characters are used with LIKE operator.
- The percent (%) means zero or more characters and the underscore (_) means exactly one character.
- Example:

```
SELECT * FROM students WHERE firstname LIKE 'Jo%';
SELECT * FROM students WHERE firstname LIKE 'Jo_';
```

```
mysql> select * from employees where fname like 'j%';
+----+----------+--------+-------------+-----------+--------------+--------+------------
--------+
| id | fname    | lname  | phonenumber | managerid | departmentid | salary | hiredate
       |
+----+----------+--------+-------------+-----------+--------------+--------+------------
--------+
|  1 | james    | smith  | 1234567890  |      NULL |            1 |   1000 | 2002-01-01 (
0:00:00 |
|  2 | James    | Smith  | 1234567890  |      NULL |            1 |   1000 | 2002-01-01 (
0:00:00 |
|  3 | John     | Johnson | 2468101214 |         1 |            1 |    400 | 2005-03-23 (
0:00:00 |
|  5 | Johnathon | Smith | 1212121212  |         2 |            1 |    500 | 2016-07-24 (
0:00:00 |
+----+----------+--------+-------------+-----------+--------------+--------+------------
--------+
4 rows in set (0.00 sec)
```

CONCAT

```
mysql> select concat(fname,lname)as fullname from employees;
+------------------+
| fullname         |
+------------------+
| jamessmith       |
| JamesSmith       |
| JohnJohnson      |
| MichaelWilliams  |
| JohnathonSmith   |
+------------------+
5 rows in set (0.05 sec)
```

TRUE / FALSE CONDITIONS

TRUE

- WHERE clause depends up on a **condition** which evaluates as either be **TRUE**, **FALSE** or **UNKNOWN**.

- A condition is made up of keywords, identifiers & constants to compare values with data rows values. If a condition is matched, its called as a TRUE condition else a FALSE condition.

- SELECT * FROM students WHERE studentid = 6;

The FROM clause will create an "intermediate result set" containing all the data rows. Then filter the records using the condition "studentid = 6". It compares each row's studentid column value to the constant value 6, and returns the data rows where condition is TRUE. It happens all at once.

- The WHERE condition can be TRUE for one or more than one data rows.

FALSE CONDITION OF WHERE

- If a conditions is not true it means the condition if FALSE.
- We can use FALSE condition to filter out the data that we don't need.
- SELECT * FROM students WHERE NOT studentid = 6;

The NOT keyword inverts the truthfulness of the condition.

- SELECT * FROM students WHERE studentid <> 6;

"less than greater than" symbol represents "not equal to".

SELECT DISTINCT STATEMENT

- SELECT DISTINCT SQL statement is used to **return only distinct or different values** from a table column.

studentid	firstname	lastname	class	age
1	John	Doe	First	5
2	Mary	Smith	Third	7
3	John	Brown	Second	6
4	James	Smith	First	6
5	John	Doe	Third	7

- If we want to get only the distinct values from the columns, the **SELECT DISTINCT** Statement is very useful.

EXAMPLE

- SELECT DISTINCT CLASS FROM STUDENT;
- SELECT DISTINCT CLASS,AGE FROM STUDENTS

SQL AGGREGATE FUNCTIONS

1. COUNT()
2. AVG()
3. SUM()
4. MAX()
5. MIN()
6. LIMIT

ALIASES

- Aliases are the temporary names given to table or column for the purpose of a particular SQL query. It is used when name of column or table is used other than their original names, but the modified name is only temporary.
- We can use
- Column aliases and
- Table aliases

ALIAS

- Alias means a **temporary name** given to a table or a column.
- Aliases makes table or column names more readable.
- Aliases are useful using multiple columns or tables in a single query. OR table names or column names are big or not readable.
- The **AS keyword** is used to define the Alias in SQL statement.
- Example:

```
SELECT s.firstname, s.lastname FROM students AS s;
SELECT COUNT(*) AS 'Total Students' FROM students;
```

TABLE ALIASING

```
mysql> select e.fname,e.lname from employees as e;
+-----------+----------+
| fname     | lname    |
+-----------+----------+
| james     | smith    |
| James     | Smith    |
| John      | Johnson  |
| Michael   | Williams |
| Johnathon | Smith    |
+-----------+----------+
5 rows in set (0.05 sec)
```

```
mysql> select fname as 'firstname',lname as 'lastname'from employees;
+-----------+----------+
| firstname | lastname |
+-----------+----------+
| james     | smith    |
| James     | Smith    |
| John      | Johnson  |
| Michael   | Williams |
| Johnathon | Smith    |
+-----------+----------+
5 rows in set (0.01 sec)
```

COUNT

- The COUNT function is **used to count the data rows returned** in the result set.
- The COUNT function **counts distinct or all values** in data rows returned in a result set.
- The COUNT function **does not count NULL** values.
- Examples:

```
SELECT COUNT(*) AS total_rows FROM tablename;
SELECT COUNT(DISTINCT columnmname) AS total_rows FROM tablename;
```

EXAMPLE

```
mysql> SELECT COUNT(*) FROM students;
+----------+
| COUNT(*) |
+----------+
|       59 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT COUNT(*) AS 'Total Rows' FROM students;
+------------+
| Total Rows |
+------------+
|         59 |
+------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT COUNT(DISTINCT class) FROM students;
+----------------------+
| COUNT(DISTINCT class) |
+----------------------+
|                    3 |
+----------------------+
1 row in set (0.00 sec)
```

FIRST()

- The FIRST function **returns the first value** of the given column.
- The FIRST function is **only supported in Microsoft Access or MS Access**.
- Examples:

SELECT FIRST(columnname) FROM tablename; (in MS access)

SELECT columnname FROM tablename LIMIT 1; (in MySQL)

EXAMPLE

```
mysql> SELECT firstname FROM students LIMIT 1;
+-----------+
| firstname |
+-----------+
| John      |
+-----------+
1 row in set (0.00 sec)
```

LAST()

- The LAST function **returns the last value** of the given column.
- The LAST function is **only supported in Microsoft Access or MS Access**.
- Examples:

SELECT LAST(columnname) FROM tablename; (in MS access)

SELECT columnname FROM tablename ORDER BY columnname DESC LIMIT 1; (in MySQL)

```
mysql> select count(fname) from employees;
+--------------+
| count(fname) |
+--------------+
|            5 |
+--------------+
1 row in set (0.42 sec)

mysql> select sum(salary)from employees;
+-------------+
| sum(salary) |
+-------------+
|        3500 |
+-------------+
1 row in set (0.04 sec)

mysql> select max(salary)from employees;
+-------------+
| max(salary) |
+-------------+
|        1000 |
+-------------+
1 row in set (0.02 sec)
```

MAX

```
mysql> select fname,max(salary) from employees group by fname;
+-----------+--------------+
| fname     | max(salary)  |
+-----------+--------------+
| james     |         1000 |
| John      |          400 |
| Michael   |          600 |
| Johnathon |          500 |
+-----------+--------------+
4 rows in set (0.10 sec)
```

LIMIT OR TOP CLAUSE

- The LIMIT or TOP clause is useful to **specify the number of data rows to return**.
- In tables with millions of data rows, it takes time to return all the records which causes database performance issue. To fix this problem, use LIMIT/TOP to return only specified number of data rows.
- The LIMIT or TOP Clause SQL statement varies for different database systems.
- Example:

SELECT TOP 5 FROM students; (SQL Server or MS Access)

SELECT * FROM students WHERE ROWNUM <=5; (Oracle)

SELECT * FROM students LIMIT 5; (MySQL)

LIMIT

```
mysql> SELECT * FROM students LIMIT 5;
+-----------+-----------+----------+--------+-------+
| studentid | firstname | lastname | class  | age   |
+-----------+-----------+----------+--------+-------+
|         1 | John      | Doe      | First  |     5 |
|         2 | Mary      | Smith    | Third  |     7 |
|         3 | James     | Brown    | First  |     5 |
|         4 | Mike      | Walker   | Second |     6 |
|         5 | Linda     | Jones    | Third  |     7 |
+-----------+-----------+----------+--------+-------+
5 rows in set (0.00 sec)
```

SQL SUB-QUERIES

```
mysql> select * from employees where salary > (select avg(salary) from employees);
+----+-------+-------+-------------+-----------+--------------+--------+-----------------
--+
| id | fname | lname | phonenumber | managerid | departmentid | salary | hiredate
  |
+----+-------+-------+-------------+-----------+--------------+--------+-----------------
--+
|  1 | james | smith | 1234567890  |      NULL |            1 |   1000 | 2002-01-01 00:00:0
0 |
|  2 | James | Smith | 1234567890  |      NULL |            1 |   1000 | 2002-01-01 00:00:0
0 |
+----+-------+-------+-------------+-----------+--------------+--------+-----------------
--+
2 rows in set (0.00 sec)
```

SQL SUB QUERIES

- A Subquery or Inner query or a Nested query is a query within another SQL query and embedded within the WHERE clause.
- A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved
- Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN, etc.

```
SELECT column_name
FROM table_name
WHERE column_name expression operator
( SELECT column_name  from table_name WHERE ... );
```

```
mysql> select eid from employee where esalary > 15000;
+------+
| eid  |
+------+
|    2 |
|    4 |
|    5 |
|    7 |
|    9 |
+------+
5 rows in set (0.00 sec)
```

```
mysql> select * from employee where eid in(select  eid from employee where esalary > 15000);
+------+-------+---------+---------+-------+
| eid  | ename | esalary | egender | edept |
+------+-------+---------+---------+-------+
|    2 | bbb   |   20000 | male    | cse   |
|    4 | ddd   |   25000 | female  | ece   |
|    5 | eee   |   17000 | female  | mech  |
|    7 | fff   |   19000 | male    | mech  |
|    9 | hhh   |   50000 | female  | civil |
+------+-------+---------+---------+-------+
5 rows in set (0.00 sec)
```

SQL SUB-QUERIES

```
mysql> select * from employees where salary > (select avg(salary) from employees);
+----+-------+-------+-------------+-----------+--------------+--------+-----------------
--+
| id | fname | lname | phonenumber | managerid | departmentid | salary | hiredate
  |
+----+-------+-------+-------------+-----------+--------------+--------+-----------------
--+
|  1 | james | smith | 1234567890  |      NULL |            1 |   1000 | 2002-01-01 00:00:0
0 |
|  2 | James | Smith | 1234567890  |      NULL |            1 |   1000 | 2002-01-01 00:00:0
0 |
+----+-------+-------+-------------+-----------+--------------+--------+-----------------
--+
2 rows in set (0.00 sec)
```

```
mysql> select avg(esalary) from employee;
+--------------+
| avg(esalary) |
+--------------+
|   20125.0000 |
+--------------+
1 row in set (0.00 sec)

mysql> select * from employee where esalary >(select avg(esalary) from employee);
+------+-------+---------+---------+-------+
| eid  | ename | esalary | egender | edept |
+------+-------+---------+---------+-------+
|    4 | ddd   |   25000 | female  | ece   |
|    9 | hhh   |   50000 | female  | civil |
+------+-------+---------+---------+-------+
2 rows in set (0.00 sec)
```

```
mysql> select * from employee where eid=(select min(eid) from employee);
+------+-------+---------+---------+-------+
| eid  | ename | esalary | egender | edept |
+------+-------+---------+---------+-------+
|    1 | aaa   |   10000 | male    | it    |
+------+-------+---------+---------+-------+
1 row in set (0.00 sec)

mysql> select * from employee where esalary =(select min(esalary) from employee);
+------+-------+---------+---------+-------+
| eid  | ename | esalary | egender | edept |
+------+-------+---------+---------+-------+
|    8 | ggg   |    5000 | female  | civil |
+------+-------+---------+---------+-------+
1 row in set (0.00 sec)
```

```
mysql> select * from employee where esalary in(select esalary from employee where edept='mech');
+------+-------+---------+---------+-------+
| eid  | ename | esalary | egender | edept |
+------+-------+---------+---------+-------+
|    5 | eee   |   17000 | female  | mech  |
|    7 | fff   |   19000 | male    | mech  |
+------+-------+---------+---------+-------+
2 rows in set (0.00 sec)

mysql> select * from employee where esalary in(select esalary from employee where esalary<19000);
+------+-------+---------+---------+-------+
| eid  | ename | esalary | egender | edept |
+------+-------+---------+---------+-------+
|    1 | aaa   |   10000 | male    | it    |
|    3 | ccc   |   15000 | female  | eee   |
|    5 | eee   |   17000 | female  | mech  |
|    8 | ggg   |    5000 | female  | civil |
+------+-------+---------+---------+-------+
4 rows in set (0.00 sec)

mysql> select * from employee where esalary in(select esalary from employee where esalary=19000);
+------+-------+---------+---------+-------+
| eid  | ename | esalary | egender | edept |
+------+-------+---------+---------+-------+
|    7 | fff   |   19000 | male    | mech  |
+------+-------+---------+---------+-------+
1 row in set (0.00 sec)
```

```
mysql> select * from employee;
+------+-------+---------+---------+-------+
| eid  | ename | esalary | egender | edept |
+------+-------+---------+---------+-------+
|    1 | aaa   |   10000 | male    | it    |
|    2 | bbb   |   20000 | male    | cse   |
|    3 | ccc   |   15000 | female  | eee   |
|    4 | ddd   |   25000 | female  | ece   |
|    5 | eee   |   17000 | female  | mech  |
|    7 | fff   |   19000 | male    | mech  |
|    8 | ggg   |    5000 | female  | civil |
|    9 | hhh   |   50000 | female  | civil |
+------+-------+---------+---------+-------+
8 rows in set (0.00 sec)

mysql> select * from employee group by egender having count(*)>1;
+------+-------+---------+---------+-------+
| eid  | ename | esalary | egender | edept |
+------+-------+---------+---------+-------+
|    1 | aaa   |   10000 | male    | it    |
|    3 | ccc   |   15000 | female  | eee   |
+------+-------+---------+---------+-------+
2 rows in set (0.00 sec)
```

```
mysql> select egender,count(*) from employee group by egender having count(*)>1;
+---------+----------+
| egender | count(*) |
+---------+----------+
| male    |        3 |
| female  |        5 |
+---------+----------+
2 rows in set (0.00 sec)
```

MORE FUNCTIONS (STRING FUNCTIONS)

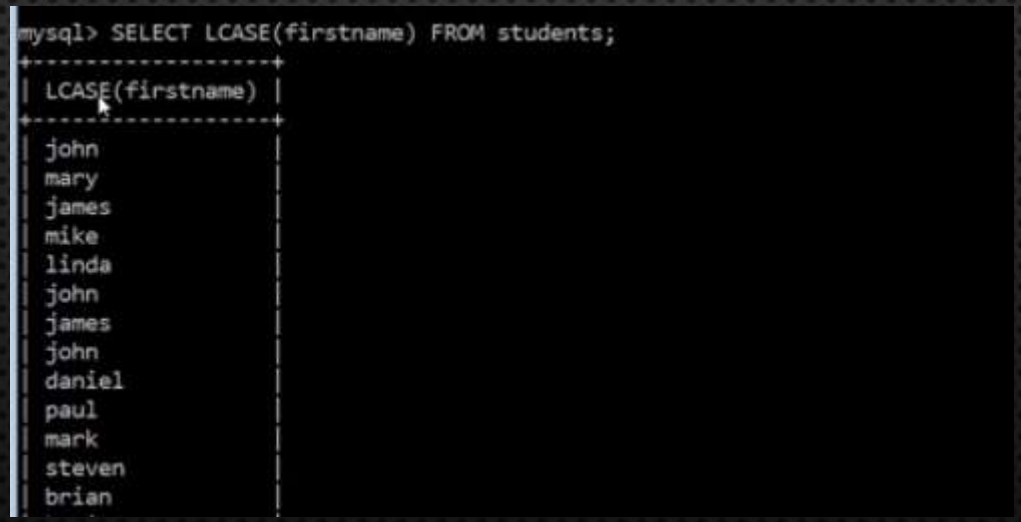
- UCASE
- LCASE
- MID
- LENGTH
- …

LCASE()

- The LCASE function **converts string data row values to lowercase**.
- The LCASE function **only work on string columns**.
- Examples:

```
SELECT LCASE(columnname) FROM tablename;
```

EXAMPLE


mysql> SELECT LCASE(firstname) FROM students;
LCASE(firstname)
john
mary
james
mike
linda
john
james
john
daniel
paul
mark
steven
brian

MID()

- The MID function **extracts characters from the given string data**.
- The MID function **only work on string columns**.
- Examples:

SELECT MID(columnname,start,length) FROM tablename;

start: starting position

length: is optional and returns the number of characters to specified. If not defined returns rest of the characters from start.

EXAMPLE

```
mysql> SELECT MID(firstname, 1, 3) FROM students;
+----------------------+
| MID(firstname, 1, 3) |
+----------------------+
| Joh                  |
| Mar                  |
| Jam                  |
| Mik                  |
| Lin                  |
| Joh                  |
| Jam                  |
| Joh                  |
| Dan                  |
| Pau                  |
| Mar                  |
| Ste                  |
| Bri                  |
| Kev                  |
| Jas                  |
| Jos                  |
| Fra                  |
| Eri                  |
| Jer                  |
```

```
mysql> SELECT MID(firstname, 2) FROM students;
+--------------------+
| MID(firstname, 2) |
+--------------------+
| ohn                |
| ary                |
| ames               |
| ike                |
| inda               |
| ohn                |
| ames               |
| ohn                |
| aniel              |
| aul                |
| ark                |
| teven              |
| rian               |
| evin               |
| ason               |
| ose                |
| rank               |
| ric                |
| erry               |
| eter               |
```

LEN()

- The LEN function **returns the length of the given string value**.
- The LEN function **only work on string columns**.
- MySQL does not support LEN function. In MySQL LENGTH is an alternate function for LEN function.
- Examples:

SELECT LEN(columnname) FROM tablename; (MS Access)

SELECT LENGTH(columnname) FROM tablename; (MySQL)

CLAUSES

- FROM
- WHERE
- ORDER BY : SORTS THE COLUMN IN ASCENDING / DESCENDING ORDER
- GROUP BY
- HAVING
- DISTINCT

FROM CLAUSE

- The FROM clause can be simple, and it can also be quite complex.
- The FROM clause **produces a tabular structure also called as the "result set" or an "intermediate result set"** or an "intermediate table" of the FROM clause.
- If we get the FROM clause wrong, the SQL statement will always return the wrong results.
- The FROM clause is the **first clause** that the database system looks at when it parses the SQL statement.

```
SELECT firstname FROM students;
```

- The FROM statements can return the result sets from one table, more than one table using joins, views and subqueries.

WHERE

- The WHERE clause is an optional clause used in SQL statements.
- Acts as **a filter on the rows of the result set** produced by the FROM clause.
- The WHERE clause mainly **depend up on a condition** which evaluates as either be true, false or unknown.
- SELECT * FROM students WHERE studentid = 6;

ORDER BY

- The order by keyword is used to sort the result-set in ascending or descending order (by default uses ascending order)
- To sort the record in descending order use desc keyword

```
mysql> SELECT * FROM EMPLOYEES ORDER BY ID DESC;
+----+----------+---------+-------------+-----------+--------------+--------+-----------
---------+
| id | fname    | lname   | phonenumber | managerid | departmentid | salary | hiredate
    |
+----+----------+---------+-------------+-----------+--------------+--------+-----------
---------+
|  5 | Johnathon | Smith   | 1212121212  |         2 |            1 |    500 | 2016-07-24
00:00:00 |
|  4 | Michael  | Williams | 1357911131  |         1 |            2 |    600 | 2009-05-12
00:00:00 |
|  3 | John     | Johnson | 2468101214  |         1 |            1 |    400 | 2005-03-23
00:00:00 |
|  2 | James    | Smith   | 1234567890  |      NULL |            1 |   1000 | 2002-01-01
00:00:00 |
|  1 | james    | smith   | 1234567890  |      NULL |            1 |   1000 | 2002-01-01
00:00:00 |
+----+----------+---------+-------------+-----------+--------------+--------+-----------
---------+
5 rows in set (0.00 sec)
```

```
mysql> SELECT Count(*) FROM Employees where ManagerId IS NOT NULL;
+----------+
| Count(*) |
+----------+
|        3 |
+----------+
1 row in set (0.00 sec)
```

```
mysql> SELECT COUNT(DISTINCT FNAME)FROM EMPLOYEES;
+-------------------------+
| COUNT(DISTINCT FNAME) |
+-------------------------+
|                       4 |
+-------------------------+
1 row in set (0.12 sec)
```

ORDER BY

- The ORDER BY clause is **used to order or sort the data rows** in a result set.
- The ORDER BY clause can be used to sort the data rows by one or more columns.
- By default the ORDER BY clause sorts the records in ascending order.
- ASC keyword for ascending while DESC keyword for descending order.
- Example:

```
SELECT * FROM students ORDER BY firstname, lastname;
SELECT * FROM students ORDER BY age DESC;
```

GROUP BY

- GROUP BY STATEMENT GROUPS ROWS THAT HAVE THE SAME VALUES INTO SUMMARY ROWS
- GROUP BY CLAUSE IS OFTEN USED WITH AGGREGATE FUNCTIONS

GROUP BY

- The GROUP BY clause **groups the data together**.
- The FROM and WHERE clause creates intermediate tabular result set and GROUP BY clause systematically groups the data.
- The GROUP BY clause can group the result set by one or more columns.
- Example:

```
SELECT class, COUNT(*) FROM students GROUP BY class;
```

```
mysql> SELECT * FROM EMPLOYEES GROUP BY DEPARTMENTID ORDER BY DEPARTMENTID DESC;
+----+-----------+----------+-------------+-----------+--------------+--------+-----------
---------+
| id | fname     | lname    | phonenumber | managerid | departmentid | salary | hiredate
        |
+----+-----------+----------+-------------+-----------+--------------+--------+-----------
---------+
|  4 | Michael   | Williams | 1357911131  |         1 |            2 |    600 | 2009-05-12
00:00:00 |
|  5 | Johnathon | Smith    | 1212121212  |         2 |            1 |    500 | 2016-07-24
00:00:00 |
+----+-----------+----------+-------------+-----------+--------------+--------+-----------
---------+
2 rows in set (0.05 sec)
```

```
mysql> select avg(salary) from employees where departmentid=1;
+-------------+
| avg(salary) |
+-------------+
|    725.0000 |
+-------------+
1 row in set (0.00 sec)

mysql> select avg(salary) from employees group by departmentid=1;
+-------------+
| avg(salary) |
+-------------+
|    725.0000 |
|    600.0000 |
+-------------+
2 rows in set (0.05 sec)
```

HAVING CLAUSE

- The HAVING clause was added to SQL as WHERE could not be used with aggregated data rows.
- The HAVING clause **filters the group rows produced by the GROUP BY clause**.
- WHERE clause filters the intermediate data result rows, while HAVING clause operates on group rows.
- The HAVING clause uses conditions and operators to build complex SQL statements.
- Since The HAVING clause acts as a filter on group rows, the only possible column in group rows are columns specified in the GROUP BY clause.
- Example:

```
SELECT class, COUNT(*) FROM students GROUP BY class HAVING COUNT(*) < 20;
```

```
mysql> SELECT * FROM EMPLOYEES GROUP BY DEPARTMENTID HAVING SALARY=600;
+----+---------+---------+-------------+-----------+--------------+--------+-------------
-------+
| id | fname   | lname   | phonenumber | managerid | departmentid | salary | hiredate
       |
+----+---------+---------+-------------+-----------+--------------+--------+-------------
-------+
|  4 | Michael | Williams | 1357911131  |         1 |            2 |    600 | 2009-05-12 00
:00:00 |
+----+---------+---------+-------------+-----------+--------------+--------+-------------
```

```sql
SELECT EmpID, SUM (MonthlySalary)
FROM Employee
GROUP BY EmpID
```

DEFINITION

- A JOIN **combines two or more tables** to produce a single tabular structure.
- The rows from two more multiple tables are combined together to return intermediate result set.
- In SQL, the joins are created using the **JOIN clause**.
- There are mainly 3 types of joins, 1. INNER JOIN, 2. OUTER JOIN, 3. CROSS JOIN
- INNER JOIN: Created with INNER JOIN keyword. It returns all data rows when there is at least one match in both tables.
- OUTER JOIN: Created with OUTER JOIN keyword. It can return matched as well as unmatched data rows. The OUTER JOIN have 3 types, Left Outer Join, Right Outer Join, Full Outer Join.

CONTD..

- A JOIN **combines two or more tables** to produce a single tabular structure.
- The rows from two more multiple tables are combined together to return intermediate result set.
- In SQL, the joins are created using the **JOIN clause**.
- There are mainly 3 types of joins, 1. INNER JOIN, 2. OUTER JOIN, 3. CROSS JOIN
- INNER JOIN: Created with INNER JOIN keyword. It returns all data rows when there is at least one match in both tables.
- OUTER JOIN: Created with OUTER JOIN keyword. It can return matched as well as unmatched data rows. The OUTER JOIN have 3 types, Left Outer Join, Right Outer Join, Full Outer Join.
- CROSS JOIN: Created with CROSS JOIN keyword. It returns all rows when there is a match in one of the tables.

```
mysql> CREATE DATABASE joins;
Query OK, 1 row affected (0.00 sec)

mysql> USE joins;
Database changed
mysql> CREATE TABLE table1 (column1 INT);
Query OK, 0 rows affected (0.25 sec)

mysql> CREATE TABLE table2 (column2 INT);
Query OK, 0 rows affected (0.19 sec)

mysql> SHOW TABLES;
+-----------------+
| Tables_in_joins |
+-----------------+
| table1          |
| table2          |
+-----------------+
2 rows in set (0.00 sec)

mysql> _
```

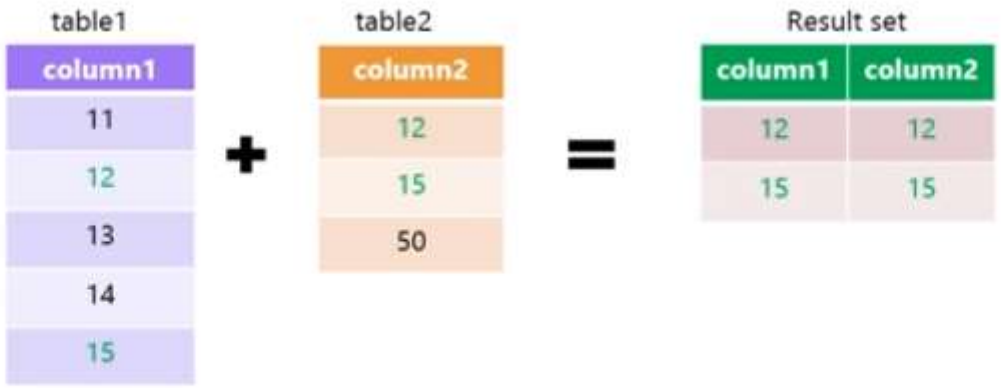
CONTD...

```
mysql> INSERT INTO table1 VALUES (11), (12), (13), (14), (15);
Query OK, 5 rows affected (0.05 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> INSERT INTO table2 VALUES (12), (15), (50);
Query OK, 3 rows affected (0.04 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM table1;
+----------+
| column1 |
+----------+
|       11 |
|       12 |
|       13 |
|       14 |
|       15 |
+----------+
5 rows in set (0.00 sec)
```

INNER JOIN

- An INNER JOIN is created by using **INNER JOIN keyword**.
- An INNER JOIN is the **most common** type of join.
- INNER JOIN returns the row in result set where the column value in a row of table1 is equal to the column value in a row of table2.
- In INNER JOIN the ON clause defines the columns and condition to be evaluated.

EXAMPLE



- SELECT * FROM table1 INNER JOIN table2 ON table1.column1 = table2.column2;

```
mysql> SELECT * FROM table2;
+---------+
| column2 |
+---------+
|      12 |
|      15 |
|      50 |
+---------+
3 rows in set (0.00 sec)

mysql> SELECT * FROM table1 INNER JOIN table2 ON table1.column1 = table2.column2;
+---------+---------+
| column1 | column2 |
+---------+---------+
|      12 |      12 |
|      15 |      15 |
+---------+---------+
2 rows in set (0.00 sec)
```

```
mysql> SELECT name, jobtitle, title FROM employees INNER JOIN projects ON employees.employeeid = projects.employeeid;
+--------------+------------------------+-----------------------------------------+
| name         | jobtitle               | title                                   |
+--------------+------------------------+-----------------------------------------+
| James Brown  | Web Developer          | Develop ecommerse website from scratch  |
| Mary Smith   | Web Developer          | WordPress website for our company       |
| James Smith  | Systems Administrator  | Manage our company servers              |
| Daniel Jones | Systems Administrator  | Hosting account is not working          |
| Frank Smith  | Database Administrator | MySQL database from my desktop application |
| Joe Thomas   | Application Developer  | Android Application development          |
+--------------+------------------------+-----------------------------------------+
6 rows in set (0.02 sec)
```

```
mysql> SELECT userid, name, phone, items, total FROM customers INNER JOIN orders ON customers.userid = orders.userid;
ERROR 1052 (23000): Column 'userid' in field list is ambiguous
mysql> SELECT customers.userid, name, phone, items, total FROM customers INNER JOIN orders ON customers.userid = orders.useri
d;
+--------+-------------+------------+-----------------------------+-------+
| userid | name        | phone      | items                       | total |
+--------+-------------+------------+-----------------------------+-------+
|      2 | Henry Clark | 0987654321 | i7 processor, 8GB RAM Laptop |  1000 |
|      1 | Jimmy Jones | 1234567890 | Healthy dog food recipes    |    19 |
|      1 | Jimmy Jones | 1234567890 | How to train your cat       |    25 |
|      3 | Ruby Young  | 1234512345 | Blue Colored Jeans          |   150 |
|      4 | Julia King  | 0987612345 | Beautiful Black T-Shirts    |    99 |
+--------+-------------+------------+-----------------------------+-------+
5 rows in set (0.00 sec)
```

```
mysql> SELECT c.userid, c.name, c.phone, o.items, o.total FROM customers AS c INNER JOIN orders AS o ON c.userid = o.userid;
+--------+-------------+------------+-----------------------------+--------+
| userid | name        | phone      | items                       | total  |
+--------+-------------+------------+-----------------------------+--------+
|      2 | Henry Clark | 0987654321 | i7 processor, 8GB RAM Laptop |   1000 |
|      1 | Jimmy Jones | 1234567890 | Healthy dog food recipes    |     19 |
|      1 | Jimmy Jones | 1234567890 | How to train your cat       |     25 |
|      3 | Ruby Young  | 1234512345 | Blue Colored Jeans          |    150 |
|      4 | Julia King  | 0987612345 | Beautiful Black T-Shirts    |     99 |
+--------+-------------+------------+-----------------------------+--------+
5 rows in set (0.00 sec)
```

LEFT OUTER JOIN

- The OUTER JOIN is created by using **OUTER JOIN keyword**.
- The OUTER JOIN **returns matched data rows as well as unmatched data rows** from the table.
- In LEFT OUTER JOIN **all the data rows from left table** are returned.
- The table mentioned to the left of the OUTER JOIN keyword is left table and the table mentioned to the right of the OUTER JOIN keyword is right table.

table1

column1
11
12
13
14
15

+

table2

column2
12
15
50

=

Result set

column1	column2
12	12
15	15
11	NULL
13	NULL
14	NULL

- SELECT * FROM table1 LEFT OUTER JOIN table2 ON table1.column1 = table2.column2;

```
mysql> SELECT * FROM table1;
+---------+
| column1 |
+---------+
|      11 |
|      12 |
|      13 |
|      14 |
|      15 |
+---------+
5 rows in set (0.00 sec)

mysql> SELECT * FROM table2;
+---------+
| column2 |
+---------+
|      12 |
|      15 |
|      50 |
+---------+
3 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM table1 LEFT OUTER JOIN table2 ON table1.column1 = table2.column2;
+---------+---------+
| column1 | column2 |
+---------+---------+
|      12 |      12 |
|      15 |      15 |
|      11 |    NULL |
|      13 |    NULL |
|      14 |    NULL |
+---------+---------+
5 rows in set (0.00 sec)
```

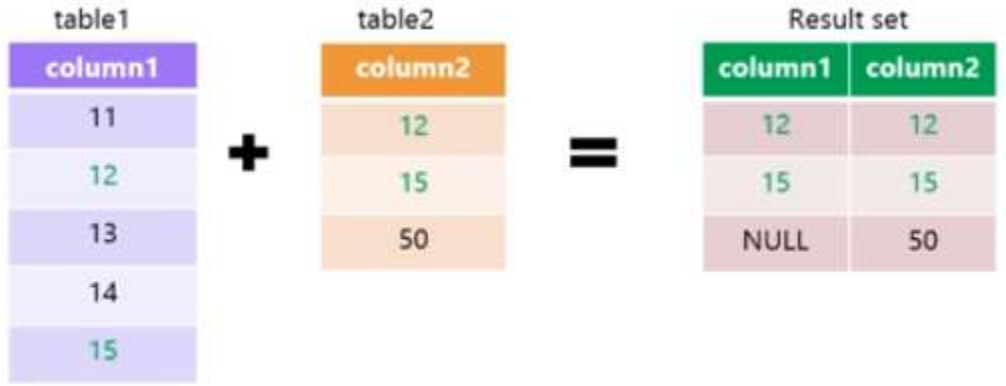
```
mysql> SELECT name, jobtitle, title FROM employees LEFT OUTER JOIN projects ON employees.employeeid = projects.employeeid;
+---------------+------------------------+-----------------------------------------+
| name          | jobtitle               | title                                   |
+---------------+------------------------+-----------------------------------------+
| James Brown   | Web Developer          | Develop ecommerse website from scratch  |
| Mary Smith    | Web Developer          | WordPress website for our company       |
| James Smith   | Systems Administrator  | Manage our company servers              |
| Daniel Jones  | Systems Administrator  | Hosting account is not working          |
| Frank Smith   | Database Administrator | MySQL database from my desktop application |
| Joe Thomas    | Application Developer  | Android Application development         |
| John Doe      | Web Developer          | NULL                                    |
| Mike Walker   | Web Developer          | NULL                                    |
| Linda Jones   | Web Developer          | NULL                                    |
| John Doe      | Application Developer  | NULL                                    |
| John Brown    | Systems Administrator  | NULL                                    |
| Paul Anderson | Systems Administrator  | NULL                                    |
| Mark Davis    | IT Support Manager     | NULL                                    |
| Steven Thomas | IT Support Manager     | NULL                                    |
| Brian King    | IT Support Manager     | NULL                                    |
| Kevin Hall    | IT Support Manager     | NULL                                    |
| Jason Lee     | IT Support Manager     | NULL                                    |
| Jose Young    | Database Administrator | NULL                                    |
| Eric Jones    | Database Administrator | NULL                                    |
```

```
mysql> USE onlineshop;
Database changed
mysql> SELECT * FROM customers;
+--------+-------------+------------+-----------------------------------+
| userid | name        | phone      | address                           |
+--------+-------------+------------+-----------------------------------+
|      1 | Jimmy Jones | 1234567890 | Victoria Pavilion, Las Vegas, NV  |
|      2 | Henry Clark | 0987654321 | 4125 Sydney Place, Miami, FL      |
|      3 | Ruby Young  | 1234512345 | 6170 Peshwar Place, Washington, DC |
|      4 | Julia King  | 0987612345 | MountainView Hospital, Victoria, TX |
|      5 | Anna Jones  | 0987609876 | 1234 Obere Street, Miami, FL      |
+--------+-------------+------------+-----------------------------------+
5 rows in set (0.00 sec)

mysql> SELECT * FROM orders;
+---------+--------+-----------------------------+-------+---------------------+
| orderid | userid | items                       | total | orderdate           |
+---------+--------+-----------------------------+-------+---------------------+
|       1 |      2 | i7 processor, 8GB RAM Laptop |  1000 | 2016-02-09 10:34:33 |
|       2 |      1 | Healthy dog food recipes    |    19 | 2016-02-09 10:34:33 |
|       3 |     25 | Healthy dog food recipes    |    19 | 2016-02-09 10:34:33 |
|       4 |      1 | How to train your cat       |    25 | 2016-02-09 10:34:33 |
|       5 |      3 | Blue Colored Jeans          |   150 | 2016-02-09 10:34:33 |
|       6 |     15 | Beautiful Black T-Shirts    |    99 | 2016-02-09 10:34:33 |
|       7 |      4 | Beautiful Black T-Shirts    |    99 | 2016-02-09 10:34:33 |
+---------+--------+-----------------------------+-------+---------------------+
7 rows in set (0.00 sec)
```

```
mysql> SELECT c.userid, c.name, c.phone, o.items, o.total FROM customers AS c LEFT OUTER JOIN orders AS o ON c.userid = o.use
rid;
+--------+-------------+------------+-----------------------------+-------+
| userid | name        | phone      | items                       | total |
+--------+-------------+------------+-----------------------------+-------+
|      2 | Henry Clark | 0987654321 | i7 processor, 8GB RAM Laptop |  1000 |
|      1 | Jimmy Jones | 1234567890 | Healthy dog food recipes    |    19 |
|      1 | Jimmy Jones | 1234567890 | How to train your cat       |    25 |
|      3 | Ruby Young  | 1234512345 | Blue Colored Jeans          |   150 |
|      4 | Julia King  | 0987612345 | Beautiful Black T-Shirts    |    99 |
|      5 | Anna Jones  | 0987609876 | NULL                        |  NULL |
+--------+-------------+------------+-----------------------------+-------+
6 rows in set (0.00 sec)
```

RIGHT OUTER JOIN

- The OUTER JOIN is created by using **OUTER JOIN keyword**.
- The OUTER JOIN **returns matched data rows as well as unmatched data rows** from the table.
- In RIGHT OUTER JOIN all the **data rows from right table are returned**.
- The table mentioned to the left of the OUTER JOIN keyword is left table and the table mentioned to the right of the OUTER JOIN keyword is right table.



- SELECT * FROM table1 RIGHT OUTER JOIN table2 ON table1.column1 = table2.column2;

```
+---------+
|      11 |
|      12 |
|      13 |
|      14 |
|      15 |
+---------+
5 rows in set (0.00 sec)

mysql> SELECT * FROM table2;
+---------+
| column2 |
+---------+
|      12 |
|      15 |
|      50 |
+---------+
3 rows in set (0.00 sec)

mysql> SELECT * FROM table1 RIGHT OUTER JOIN table2 ON table1.column1 = table2.column2;
+---------+---------+
| column1 | column2 |
+---------+---------+
|      12 |      12 |
|      15 |      15 |
|    NULL |      50 |
+---------+---------+
3 rows in set (0.00 sec)
```

```
mysql> SELECT name, jobtitle, title FROM employees RIGHT OUTER JOIN projects ON employees.employeeid = projects.employeeid;
+--------------+-------------------------+-----------------------------------------------------+
| name         | jobtitle                | title                                               |
+--------------+-------------------------+-----------------------------------------------------+
| James Brown  | Web Developer           | Develop ecommerse website from scratch              |
| Mary Smith   | Web Developer           | WordPress website for our company                   |
| James Smith  | Systems Administrator   | Manage our company servers                          |
| Daniel Jones | Systems Administrator   | Hosting account is not working                      |
| Frank Smith  | Database Administrator  | MySQL database from my desktop application          |
| NULL         | NULL                    | Develop new WordPress plugin for my business website |
| NULL         | NULL                    | Migrate web application and database to new server  |
| Joe Thomas   | Application Developer   | Android Application development                     |
+--------------+-------------------------+-----------------------------------------------------+
8 rows in set (0.00 sec)
```

```
mysql> SELECT c.userid, c.name, c.phone, o.items, o.total FROM customers AS c RIGHT OUTER JOIN orders AS o ON c.userid = o.us
erid;
+--------+-------------+------------+-----------------------------+--------+
| userid | name        | phone      | items                       | total  |
+--------+-------------+------------+-----------------------------+--------+
|      1 | Jimmy Jones | 1234567890 | Healthy dog food recipes    |     19 |
|      1 | Jimmy Jones | 1234567890 | How to train your cat       |     25 |
|      2 | Henry Clark | 0987654321 | i7 processor, 8GB RAM Laptop|   1000 |
|      3 | Ruby Young  | 1234512345 | Blue Colored Jeans          |    150 |
|      4 | Julia King  | 0987612345 | Beautiful Black T-Shirts    |     99 |
|   NULL | NULL        | NULL       | Healthy dog food recipes    |     19 |
|   NULL | NULL        | NULL       | Beautiful Black T-Shirts    |     99 |
+--------+-------------+------------+-----------------------------+--------+
7 rows in set (0.00 sec)
```

FULL OUTER JOIN

- The OUTER JOIN is created by using **OUTER JOIN keyword**.
- The OUTER JOIN **returns matched data rows as well as unmatched data rows** from the table.
- In FULL OUTER JOIN **all the data rows from left and right tables are returned**.
- **MySQL doesn't support FULL OUTER JOIN**, we will emulate the FULL OUTER JOIN using UNION Operator.

table1

column1
11
12
13
14
15

+

table2

column2
12
15
50

=

Result set

column1	column2
12	12
15	15
11	NULL
13	NULL
14	NULL
NULL	50

- SELECT * FROM table1 FULL OUTER JOIN table2 ON table1.column1 = table2.column2;
- SELECT * FROM table1 LEFT OUTER JOIN table2 ON table1.column1 = table2.column2 UNION SELECT * FROM table1 RIGHT OUTER JOIN table2 ON table1.column1 = table2.column2;

```
mysql> SELECT * FROM table1;
+---------+
| column1 |
+---------+
|      11 |
|      12 |
|      13 |
|      14 |
|      15 |
+---------+
5 rows in set (0.00 sec)

mysql> SELECT * FROM table2;
+---------+
| column2 |
+---------+
|      12 |
|      15 |
|      50 |
+---------+
3 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM table1 LEFT OUTER JOIN table2 ON table1.column1 = table2.column2 UNION SELECT * FROM table1 RIGHT OUTER
JOIN table2 ON table1.column1 = table2.column2;
+---------+---------+
| column1 | column2 |
+---------+---------+
|      12 |      12 |
|      15 |      15 |
|      11 |    NULL |
|      13 |    NULL |
|      14 |    NULL |
|    NULL |      50 |
+---------+---------+
6 rows in set (0.00 sec)
```

- The CROSS JOIN is created by using **CROSS JOIN keyword**.
- The CROSS JOIN **does not contain ON clause**.
- The CROSS JOIN returns matched data rows as well as unmatched data rows from the table.
- In CROSS all rows from all tables are returned, joined to every row of the other table regardless of whether they match.

table1

column1
11
12
13
14
15

+

table2

column2
12
15
50

=

Result set

column1	column2
11	12
11	15
11	50
12	12
12	15
12	50
13	12
13	15
13	50

column1	column2
14	12
14	15
14	50
15	12
15	15
15	50

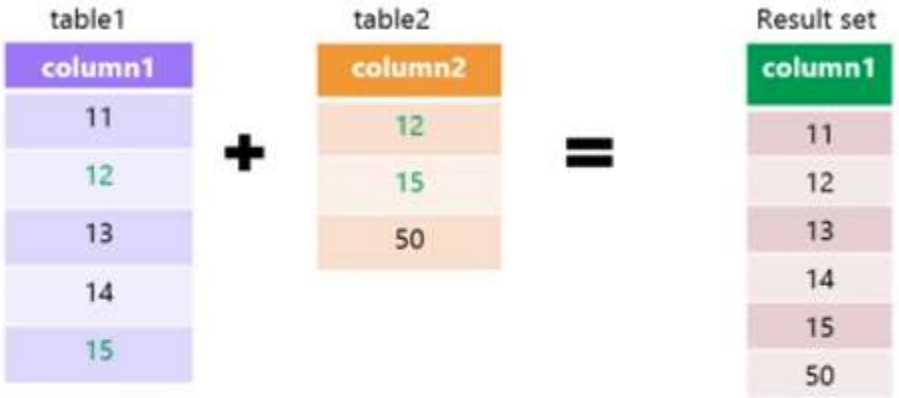
- SELECT * FROM table1 CROSS JOIN table2;

```
mysql> SELECT * FROM table1 CROSS JOIN table2;
+---------+---------+
| column1 | column2 |
+---------+---------+
|      11 |      12 |
|      11 |      15 |
|      11 |      50 |
|      12 |      12 |
|      12 |      15 |
|      12 |      50 |
|      13 |      12 |
|      13 |      15 |
|      13 |      50 |
|      14 |      12 |
|      14 |      15 |
|      14 |      50 |
|      15 |      12 |
|      15 |      15 |
|      15 |      50 |
+---------+---------+
15 rows in set (0.00 sec)
```

```
mysql> SELECT employees.name, employees.jobtitle, projects.title FROM employees CROSS JOIN projects;
```

UNION

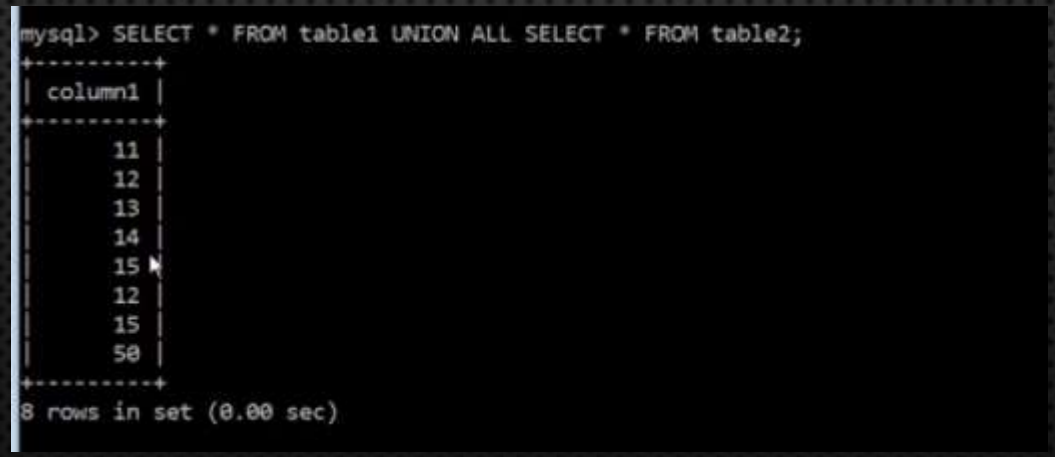
- The UNION Operator is **used to combine the result set of two or more SELECT statements**.
- Every SELECT statement must have similar number of columns.
- The columns in SELECT statement must be in the same order and have similar data types.
- The UNION operator selects only distinct values. The UNION ALL operator is used to select duplicate values.



- SELECT * FROM table1 UNION SELECT * FROM table2;

```
mysql> SELECT * FROM table2;
+---------+
| column2 |
+---------+
|      12 |
|      15 |
|      50 |
+---------+
3 rows in set (0.00 sec)

mysql> SELECT * FROM table1 UNION SELECT * FROM table2;
+---------+
| column1 |
+---------+
|      11 |
|      12 |
|      13 |
|      14 |
|      15 |
|      50 |
+---------+
6 rows in set (0.00 sec)
```

CONTD..


mysql> SELECT * FROM table1 UNION ALL SELECT * FROM table2;
+---------+
| column1 |
+---------+
| 11 |
| 12 |
| 13 |
| 14 |
| 15 |
| 12 |
| 15 |
| 50 |
+---------+
8 rows in set (0.00 sec)

```
SELECT COLUMN_NAME(S)
FROM TABLE_NAME
WHERE CONDITION
GROUP BY COLUMN_NAME(S)
HAVING CONDITION
ORDER BY COLUMN_NAME(S);
```

SQL SUBQUERIES

SQL SUBQUERIES

- Subquery simply an SQL query which is placed inside another SQL query.

```
mysql> select * from employee;
+-----+-----------+-------+---------+
| sno | sname     | dname | salary  |
+-----+-----------+-------+---------+
|   1 | raju      | A     |   12500 |
|   2 | ramu      | BB    |   11000 |
|   3 | rahul     | cc    |   15000 |
|   4 | anil      | dd    |   13500 |
|   5 | arshdeep  | ee    |   19000 |
|   6 | harsha    | ff    |   21000 |
|   7 | arshad    | gg    |   11200 |
+-----+-----------+-------+---------+
7 rows in set (0.00 sec)
```

FIND THE EMPLOYEES WHOSE SALARY IS MORE THAN THE AVERAGE SALARY EARNED BY ALL EMPLOYEES

```
mysql> select avg(salary) from employee;
+-------------+
| avg(salary) |
+-------------+
|  14742.8571 |
+-------------+
1 row in set (0.07 sec)
```

SELECT * FROM EMPLOYEE WHERE SALARY > (SELECT AVG(SALARY) FROM EMPLOYEE);

AVERAGE SALARY

```
mysql> select * from employee where salary > (select avg(salar
y) from employee);
+-----+----------+-------+--------+
| sno | sname    | dname | salary |
+-----+----------+-------+--------+
|   3 | rahul    | cc    |  15000 |
|   5 | arshdeep | ee    |  19000 |
|   6 | harsha   | ff    |  21000 |
+-----+----------+-------+--------+
3 rows in set (0.08 sec)
```

TYPES OF SUB QUERIES

- THERE ARE 3 TYPES OF SUB QUERIES ARE THERE
  - SCALAR SUBQUERY
  - MULTIPLE ROW SUBQUERY
  - CORRELATED SUBQUERY

SCALAR SUBQUERY

- RETURNS ONE ROW AND ONE COLOUM
- RETURNS AVERAGE SALARY ..THE OUTPUT IS ONE ROW AND ONE COLUMN

SECOND HIGHEST EMPLOYEE SALARY

```
mysql> select max(salary) from employee where salary < (select
 max(salary) from employee);
+-------------+
| max(salary) |
+-------------+
|       19200 |
+-------------+
1 row in set (0.00 sec)
```

SCALAR SUB QUERY

```
mysql> select avg(salary) from employee;
+-------------+
| avg(salary) |
+-------------+
|  14742.8571 |
+-------------+
1 row in set (0.07 sec)
```

- FIND THE EMPLOYEES WHOSE SALARY IS MORE THAN THE AVERAGE SALARY EARNED BY ALL EMPLOYEES

USING JOIN

```
mysql> select * from employee e join customers c on c.sno=e.sn
o;
+-----+-------+-------+--------+-----+-------+-------+--------
+
| sno | sname | dname | salary | SNO | SNMAE | CITY  | gender
|
+-----+-------+-------+--------+-----+-------+-------+--------
+
|   1 | raju  | A     |  12500 |   1 | HELLO | HYD   |
|
|   2 | ramu  | BB    |  11000 |   2 | HI    | DELHI | f
|
|   3 | rahul | cc    |  15000 |   3 | H     | AP    |
|
|   4 | anil  | dd    |  13500 |   4 | HERKE | TEL   |
```

EXAMPLE

```
mysql> select * from employee as e join (select avg(salary) as
 sal from employee)as avg_sal on e.salary> avg_sal.sal;
+-----+----------+-------+--------+------------+
| sno | sname    | dname | salary | sal        |
+-----+----------+-------+--------+------------+
|   3 | rahul    | cc    |  15000 | 14742.8571 |
|   5 | arshdeep | ee    |  19000 | 14742.8571 |
|   6 | harsha   | ff    |  21000 | 14742.8571 |
+-----+----------+-------+--------+------------+
3 rows in set (0.07 sec)
```

MULTIPLE ROW SUBQUERY

- SUBQUERY WHICH RETURNS MULTIPLE COLUMN AND MULTIPLE ROW

FIND THE EMPLOYEES WHO EARN THE HIGHEST SALARY IN EACH DEPARTMENT

```
mysql> select * from employee;
+-----+----------+-------+--------+
| sno | sname    | dname | salary |
+-----+----------+-------+--------+
|   1 | raju     | A     |  12500 |
|   2 | ramu     | BB    |  11000 |
|   3 | rahul    | cc    |  15000 |
|   4 | anil     | dd    |  13500 |
|   5 | arshdeep | ee    |  19000 |
|   6 | harsha   | ff    |  21000 |
|   7 | arshad   | gg    |  11200 |
|   8 | rao      | bb    |  12345 |
|   9 | rak      | bb    |   1345 |
|  10 | rak      | dd    |   1200 |
|  11 | syam     | dd    |  19200 |
+-----+----------+-------+--------+
11 rows in set (0.00 sec)
```

```
mysql> select dname,max(salary) from employee group by dname;
+-------+-------------+
| dname | max(salary) |
+-------+-------------+
| A     |       12500 |
| BB    |       12345 |
| cc    |       15000 |
| dd    |       19200 |
| ee    |       19000 |
| ff    |       21000 |
| gg    |       11200 |
+-------+-------------+
7 rows in set (0.05 sec)
```

MULTIPLE ROW MULTIPLE COLUMN SUBQUERY

```
mysql> select * from employee where(dname,salary) in (select d
name,max(salary)from employee group by dname);
+-----+----------+-------+--------+
| sno | sname    | dname | salary |
+-----+----------+-------+--------+
|   1 | raju     | A     |  12500 |
|   3 | rahul    | cc    |  15000 |
|   5 | arshdeep | ee    |  19000 |
|   6 | harsha   | ff    |  21000 |
|   7 | arshad   | gg    |  11200 |
|   8 | rao      | bb    |  12345 |
|  11 | syam     | dd    |  19200 |
+-----+----------+-------+--------+
7 rows in set (0.07 sec)
```

```
mysql> select * from employee where(dname,salary) = (select dn
ame,max(salary)from employee group by dname) order by dname de
sc;
ERROR 1242 (21000): Subquery returns more than 1 row
mysql>
```

CORELATED SUBQUERY(SUBQUERY DEPENDENT ON THE OUTER QUERY VALUES)

FIND THE EMPLOYEES IN EACH DEPAETMENT IN EACH DEPARTMENT WHE EARN MORE THAN THE AVERAGE SALARY IN THAT DEPARTMENT

```
mysql> select * from employee e1 where salary > (select avg(sa
lary) from employee e2 where e2.dname=e1.dname);
+-----+-------+-------+--------+
| sno | sname | dname | salary |
+-----+-------+-------+--------+
|   2 | ramu  | BB    |  11000 |
|   4 | anil  | dd    |  13500 |
|   8 | rao   | bb    |  12345 |
|  11 | syam  | dd    |  19200 |
+-----+-------+-------+--------+
4 rows in set (0.72 sec)
```

VIEWS

- A View in SQL is simply a virtual table created based on a result set of another SQL statement.
- Views are virtual tables, i.e., they don't exist in reality in the database, hence don't require any storage in a database. Virtual Tables also have rows and columns similar to a real table in a database. Such views are simply made by selecting data(fields) from one or more tables, present in the database, with some conditions for selecting rows of the table.

ADVANTAGES :

- Reduce complexity of multiple tables
- Deliver data in a simple way
- Data integrity
- Provides data security

```
mysql> select * from employees;
+-----+-------+-------+---------+---------+----------------+---------------+
| EID | FNAME | LNAME | MOBILE  | ADDRESS | EMAIL          | DEPARTMENT_ID |
+-----+-------+-------+---------+---------+----------------+---------------+
|   1 | AAA   | Z     | 3748378 | HYD     | EIURHOIEUIORO  |             1 |
|   2 | DJF   | EYR   |  347834 | WRL     | DIFFFDFJDF     |             5 |
|   5 | DF    | DUF   | 3784849 | HYD     | KFPOERJ        |             2 |
|   6 | DF    | DUF   |   37849 | HYD     | KFPO           |             3 |
|   7 | DEF   | DSUF  | 3457849 | HYD     | KDFPO          |             4 |
+-----+-------+-------+---------+---------+----------------+---------------+
5 rows in set (0.08 sec)

mysql> create view hyd_people as
    -> select * from employees where address="hyd";
Query OK, 0 rows affected (1.72 sec)

mysql> show tables;
+----------------+
| Tables_in_demo |
+----------------+
| check_demo     |
| default_demo   |
| department     |
| employees      |
| hyd_people     |
| school         |
| unique_demo    |
| voter_reg      |
+----------------+
8 rows in set (0.01 sec)

mysql> select * from hyd_people;
+-----+-------+-------+---------+---------+----------------+---------------+
| EID | FNAME | LNAME | MOBILE  | ADDRESS | EMAIL          | DEPARTMENT_ID |
+-----+-------+-------+---------+---------+----------------+---------------+
|   1 | AAA   | Z     | 3748378 | HYD     | EIURHOIEUIORO  |             1 |
|   5 | DF    | DUF   | 3784849 | HYD     | KFPOERJ        |             2 |
|   6 | DF    | DUF   |   37849 | HYD     | KFPO           |             3 |
|   7 | DEF   | DSUF  | 3457849 | HYD     | KDFPO          |             4 |
+-----+-------+-------+---------+---------+----------------+---------------+
4 rows in set (0.14 sec)
```

```
mysql> update hyd_people set email = "admin@codinghub.org.in"  where eid=7;
Query OK, 1 row affected (0.45 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from hyd_people;
+-----+-------+-------+---------+---------+-------------------------+---------------+
| EID | FNAME | LNAME | MOBILE  | ADDRESS | EMAIL                   | DEPARTMENT_ID |
+-----+-------+-------+---------+---------+-------------------------+---------------+
|   1 | AAA   | Z     | 3748378 | HYD     | EIURHOIEUIORO           |             1 |
|   5 | DF    | DUF   | 3784849 | HYD     | KFPOERJ                 |             2 |
|   6 | DF    | DUF   |   37849 | HYD     | KFPO                    |             3 |
|   7 | DEF   | DSUF  | 3457849 | HYD     | admin@codinghub.org.in  |             4 |
+-----+-------+-------+---------+---------+-------------------------+---------------+
4 rows in set (0.00 sec)

mysql> select * from employees;
+-----+-------+-------+---------+---------+-------------------------+---------------+
| EID | FNAME | LNAME | MOBILE  | ADDRESS | EMAIL                   | DEPARTMENT_ID |
+-----+-------+-------+---------+---------+-------------------------+---------------+
|   1 | AAA   | Z     | 3748378 | HYD     | EIURHOIEUIORO           |             1 |
|   2 | DJF   | EYR   |  347834 | WRL     | DIFFFDFJDF              |             5 |
|   5 | DF    | DUF   | 3784849 | HYD     | KFPOERJ                 |             2 |
|   6 | DF    | DUF   |   37849 | HYD     | KFPO                    |             3 |
|   7 | DEF   | DSUF  | 3457849 | HYD     | admin@codinghub.org.in  |             4 |
+-----+-------+-------+---------+---------+-------------------------+---------------+
5 rows in set (0.00 sec)
```

```
mysql> update employees set fname = "MK" WHERE department_id=4;
Query OK, 1 row affected (0.06 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from employees;
+-----+-------+-------+---------+---------+------------------------+---------------+
| EID | FNAME | LNAME | MOBILE  | ADDRESS | EMAIL                  | DEPARTMENT_ID |
+-----+-------+-------+---------+---------+------------------------+---------------+
|   1 | AAA   | Z     | 3748378 | HYD     | EIURHOIEUIORO          |             1 |
|   2 | DJF   | EYR   |  347834 | WRL     | DIFFFDFJDF             |             5 |
|   5 | DF    | DUF   | 3784849 | HYD     | KFPOERJ                |             2 |
|   6 | DF    | DUF   |   37849 | HYD     | KFPO                   |             3 |
|   7 | MK    | DSUF  | 3457849 | HYD     | admin@codinghub.org.in |             4 |
+-----+-------+-------+---------+---------+------------------------+---------------+
5 rows in set (0.00 sec)

mysql> select * from hyd_people;
+-----+-------+-------+---------+---------+------------------------+---------------+
| EID | FNAME | LNAME | MOBILE  | ADDRESS | EMAIL                  | DEPARTMENT_ID |
+-----+-------+-------+---------+---------+------------------------+---------------+
|   1 | AAA   | Z     | 3748378 | HYD     | EIURHOIEUIORO          |             1 |
|   5 | DF    | DUF   | 3784849 | HYD     | KFPOERJ                |             2 |
|   6 | DF    | DUF   |   37849 | HYD     | KFPO                   |             3 |
|   7 | MK    | DSUF  | 3457849 | HYD     | admin@codinghub.org.in |             4 |
+-----+-------+-------+---------+---------+------------------------+---------------+
4 rows in set (0.00 sec)
```

```
mysql> select * from department;
+---------+-----------+-----------+-------------+
| DEPT_ID | DEPT_NAME | DEPT_ADDR | DEPT_MOBILE |
+---------+-----------+-----------+-------------+
|       1 | CSE       | HYD       |     3243273 |
|       2 | ECE       | WRL       |   348443873 |
|       3 | EEE       | VJWD      |   783489734 |
|       4 | MECH      | NLG       |        3743 |
|       5 | CIVIL     | SRPT      |       39874 |
|       6 | IT        | KODAD     |      378489 |
+---------+-----------+-----------+-------------+
6 rows in set (0.05 sec)

mysql> select * from employees;
+-----+-------+-------+---------+---------+------------------------+---------------+
| EID | FNAME | LNAME | MOBILE  | ADDRESS | EMAIL                  | DEPARTMENT_ID |
+-----+-------+-------+---------+---------+------------------------+---------------+
|   1 | AAA   | Z     | 3748378 | HYD     | EIURHOIEUIORO          |             1 |
|   2 | DJF   | EYR   |  347834 | WRL     | DIFFFDFJDF             |             5 |
|   5 | DF    | DUF   | 3784849 | HYD     | KFPOERJ                |             2 |
|   6 | DF    | DUF   |   37849 | HYD     | KFPO                   |             3 |
|   7 | MK    | DSUF  | 3457849 | HYD     | admin@codinghub.org.in |             4 |
+-----+-------+-------+---------+---------+------------------------+---------------+
5 rows in set (0.00 sec)
```

```
mysql> create view employees_deparment as
    -> select e.eid ,e.mobile,e.email,d.dept_id,d.dept_name,d.dept_mobile from employees as e,department as d where e.
eid=d.dept_id;
Query OK, 0 rows affected (0.22 sec)

mysql> show tables;
+----------------------+
| Tables_in_demo       |
+----------------------+
| check_demo           |
| default_demo         |
| department           |
| employees            |
| employees_deparment  |
| hyd_people           |
| school               |
| unique_demo          |
| voter_reg            |
+----------------------+
9 rows in set (0.00 sec)

mysql> select * from employees_department;
ERROR 1146 (42S02): Table 'demo.employees_department' doesn't exist
mysql> select * from employees_deparment;
+-----+---------+---------------+---------+-----------+-------------+
| eid | mobile  | email         | dept_id | dept_name | dept_mobile |
+-----+---------+---------------+---------+-----------+-------------+
|   1 | 3748378 | EIURHOIEUIORO |       1 | CSE       |     3243273 |
|   2 |  347834 | DIFFFDFJDF    |       2 | ECE       |   348443873 |
|   5 | 3784849 | KFPOERJ       |       5 | CIVIL     |       39874 |
|   6 |   37849 | KFPO          |       6 | IT        |      378489 |
+-----+---------+---------------+---------+-----------+-------------+
4 rows in set (0.05 sec)
```

ER DIRAGRAM

ENTITY RELATIONSHIP MODEL

- PICTORIAL REPRESENTATION OF DATABASE SHOWING ENTITY TYPES & THEIR RELATIONSHIPS

ENTITY

- Student entity
- Attributes of student
  - Name
  - Gender
  - section

COMPONENTS

- ENTITY
  - STRONG
  - WEEK
- ATTRIBUTES
  - SIMPLE / ATOMIC ATTRIBUTES: CANT BE FURTHER DIVIDED EX : MOBILE, GENDER, EMAIL
  - COMPOSITE ATTRIBUTES : IF AN ATTRIBUTE WHICH CAN BE FURTHER DIVIDED TO FORM A INDEPENDENT ATTRIBUTE EX:NAME,FIRST NAME,…
  - KEY ATTRIBUTES : USED TO IDENTIFY AN ENTITY UNIQUELY EX: PRIMARY KEY
  - SINGLE VALUED ATTRIBUTES : WHICH CAN HOLD ONLY ONE VALUE EX: GENDER
  - MULTI VALUED ATTRIBUTES: WHICH CAN HOLD MULTIPLE VALUES EX: CONTACT NUMBERS
  - DERIVED ATTRIBUTES : WHICH CAN BE DERIVED FROM ANOTHER ATTRIBUTE
    - EX: AGE AND DOB
- RELATIONSHIPS
  - ONE OT ONE
  - ONE TO MANY
  - MANY TO MANY

ATTRIBUTE IS PROPERTIES OR CHARACTERISTICS THAT DESCRIBES AN ENTITY

- SIMPLE ATTRIBUTES
- COMPOSITE ATTRIBUTES
- KEY ATTRIBUTES
- SINGLE VALUED ATTRIBUTES
- MULTI VALUED ATTRIBUTES

WHAT IS AN ENTITY

- Real world object having characteristics & properties
- Two types are there
  - Tangible entites which exists physically ex: car, person
  - Intangible entities exists logically ex : account number , Login details
- Entity set : group of entities having same attributes
- Entity type : enity set is represented by entity type
  - Strong entity type : have a key attribute
  - Week entity type

NORMALIZATION

ANAMOLY

- A database anamoly is a fault in a database.
- A database anamoly is an inconsistency in data resulting from an operation like
  - Insertion
  - Updation
  - deletion
- Anomalies occur when the data present in the database has too much redundancy and if the tables making up the database are poorly constructed

ENTITY Integrity CONSTRAINTS

Every relation must have primary key

Every primary key attribute is non- null and unique

Unique means, value of the primary key must be different not be same.

Null is a value, suppose no other value not assign or apply.

Null is not zero (0) or blank (" ")

ENTITY Integrity CONSTRAINTS

Not Unique

Department

Dept Name	Location	Fax
Marketting	Sohar	24535332
Accounting	Sohar	24522222
Info System	Muscat	26565666
Finance	Muscat	26222228

Entity integrity not enforced

Not Unique

Same value

Dept Name	Location	Fax
Marketting	Sohar	24535332
Accounting	Sohar	24522222
Info System	Muscat	26565666
Marketting	Muscat	26222228

ENTITY Integrity CONSTRAINTS

Null is a value, suppose no other value not assign or apply.

Null is not zero (0) or blank (" ")

Entity integrity not enforced

Not Null

Dept Name	Location	Fax
Marketting	Sohar	24535332
Accounting	Sohar	24522222
Info System	Muscat	26565666
NULL	Muscat	26222228

No null value allowed in the primary key

INSERTION ANAMOLY

```
mysql> create table student
    -> (
    -> sid int not null ,
    -> sname varchar(20) not null,
    -> saddr varchar(20) not null,
    -> course varchar(20) not null,
    -> time int not null
    -> );
Query O;
ERROR:
No query specified student add primary key(

mysql> alter table student add primary key(sid,course);
Query OK, 0 rows affected (1.25 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> insert into student values(100,"raju","hyd","java",5);
Query OK, 1 row affected (0.13 sec)

mysql> insert into student values(100,"raju","hyd",null,5);
ERROR 1048 (23000): Column 'course' cannot be null
```

INSERT ANOMALY

The primary key for this table is the combination of Emp_ID and Course_Title

EmpId	Ename	DeptName	Salary	Course_Title	Date_Completed
100	Ali	Marketting	4800	SPSS	6/19/2011
100	Ali	Marketting	4800	Surveys	10/7/2011
140	Said	Accounting	5200	Tax Acc	12/8/2011
110	ahmed	Infosystem	4300	Visual Basic	1/12/2011
110	ahmed	Infosystem	4300	C++	4/22/2011
150	Khalid	Marketting	4200	SPSS	6/19/2011
150	khalid	Marketting	4200	Java	8/12/2011

Because, in the new record empid is available but course_title is Null. As per entity integrity rule primary key must have not null value.

EmpId	Ename	DeptName	Salary	Course_Title	Date_Completed
100	Ali	Marketting	4800	Null	6/19/2011

This is insert anomaly

UPDATE ANOMALY

EmpId 150 has two records, the salary as 4200

EmpId	Ename	DeptName	Salary	Course_Title	Date_Completed
100	Ali	Marketting	4800	SPSS	6/19/2011
100	Ali	Marketting	4800	Surveys	10/7/2011
140	Said	Accounting	5200	Tax Acc	12/8/2011
110	ahmed	Infosystem	4300	Visual Basic	1/12/2011
110	ahmed	Infosystem	4300	C++	4/22/2011
150	Khalid	Marketting	4200	SPSS	6/19/2011
150	khalid	Marketting	4200	Java	8/12/2011

Once record change the salary as 4500 and another record missed for updating the changes is called update anomaly

EmpId	Ename	DeptName	Salary	Course_Title	Date_Completed
150	Khalid	Marketting	4200	SPSS	6/19/2011
150	khalid	Marketting	4500	Java	8/12/2011

This is update anomaly

DELETE ANOMALY

EmpId	Ename	DeptName	Salary	Course_Title	Date_Completed
100	Ali	Marketting	4800	SPSS	6/19/2011
100	Ali	Marketting	4800	Surveys	10/7/2011
140	Said	Accounting	5200	Tax Acc	12/8/2011
110	ahmed	Infosystem	4300	Visual Basic	1/12/2011
110	ahmed	Infosystem	4300	C++	4/22/2011
150	Khalid	Marketting	4200	SPSS	6/19/2011
150	khalid	Marketting	4200	Java	8/12/2011

Suppose delete empid 140, it also delete the course_title and date_completed also.

EmpId	Ename	DeptName	Salary	Course_Title	Date_Completed
140	Said	Accounting	5200	Tax Acc	12/8/2011

CONCLUSION

- A database anomaly usually occurs as a result of poor planning and the use of flat databases.
- In this article we learned about insert, delete and update anomalies as well as the circumstances that can lead to them.
- Anomalies are usually removed by normalizing the tables by splitting or joining them
- Normalization provides structured data

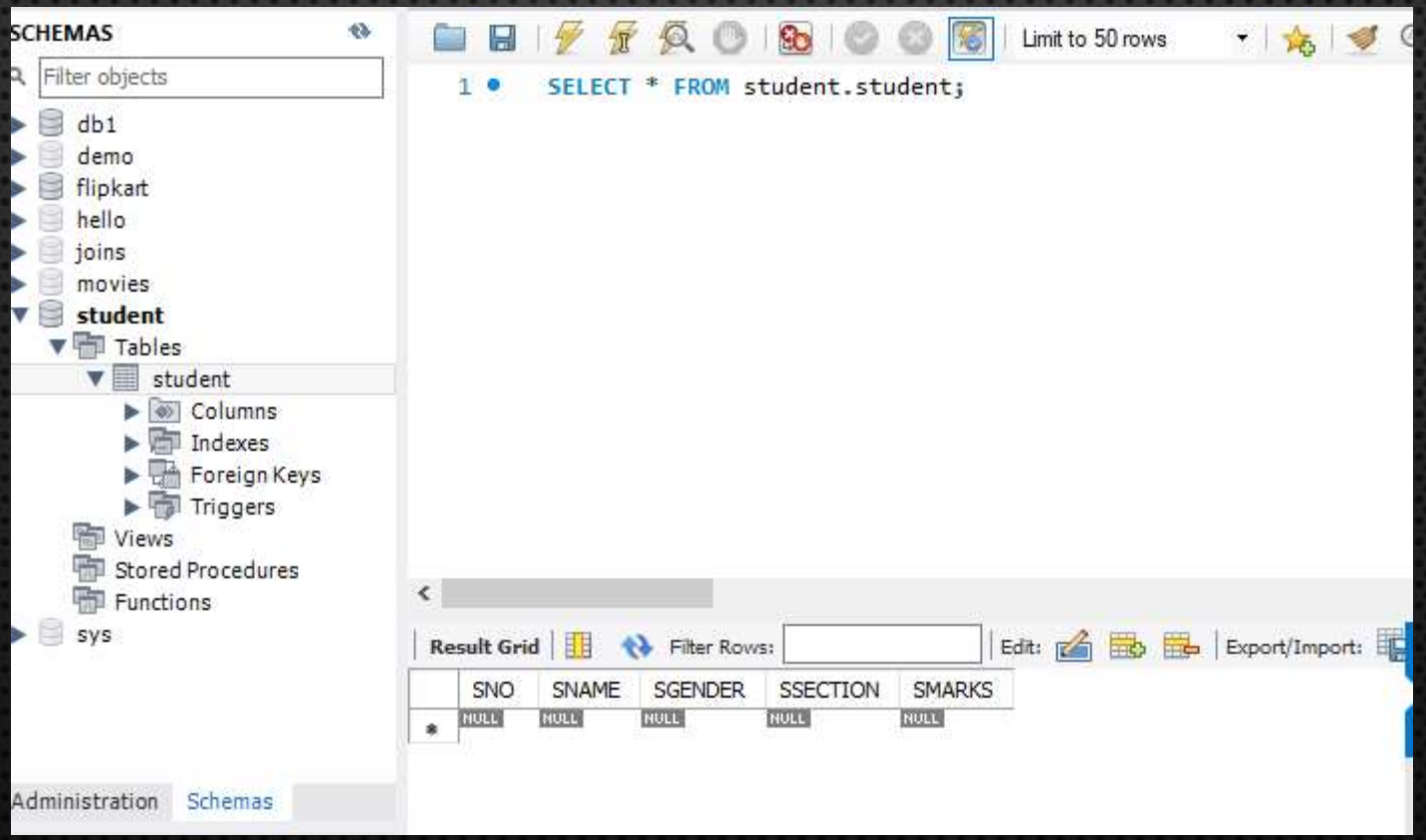
Stored Procedures

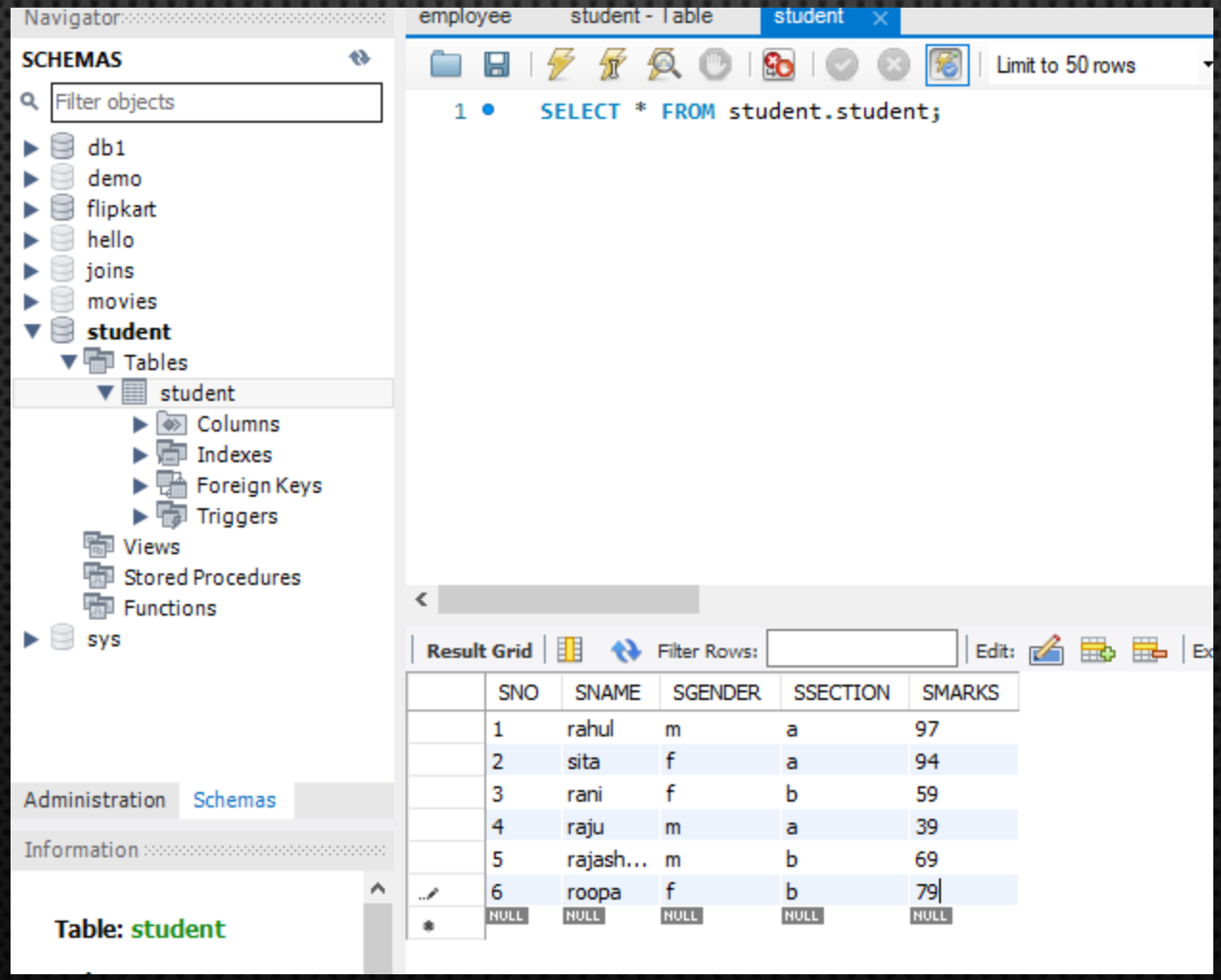
- Database Procedures (sometimes referred to as Stored Procedures or Procs) are subroutines that can contain one or more SQL statements that perform a specific task
- **Stored Procedures** are created to perform one or more DML & DDL operations on Database. It is nothing but the group of SQL statements that accepts some input in the form of parameters and performs some task and may or may not returns a value.

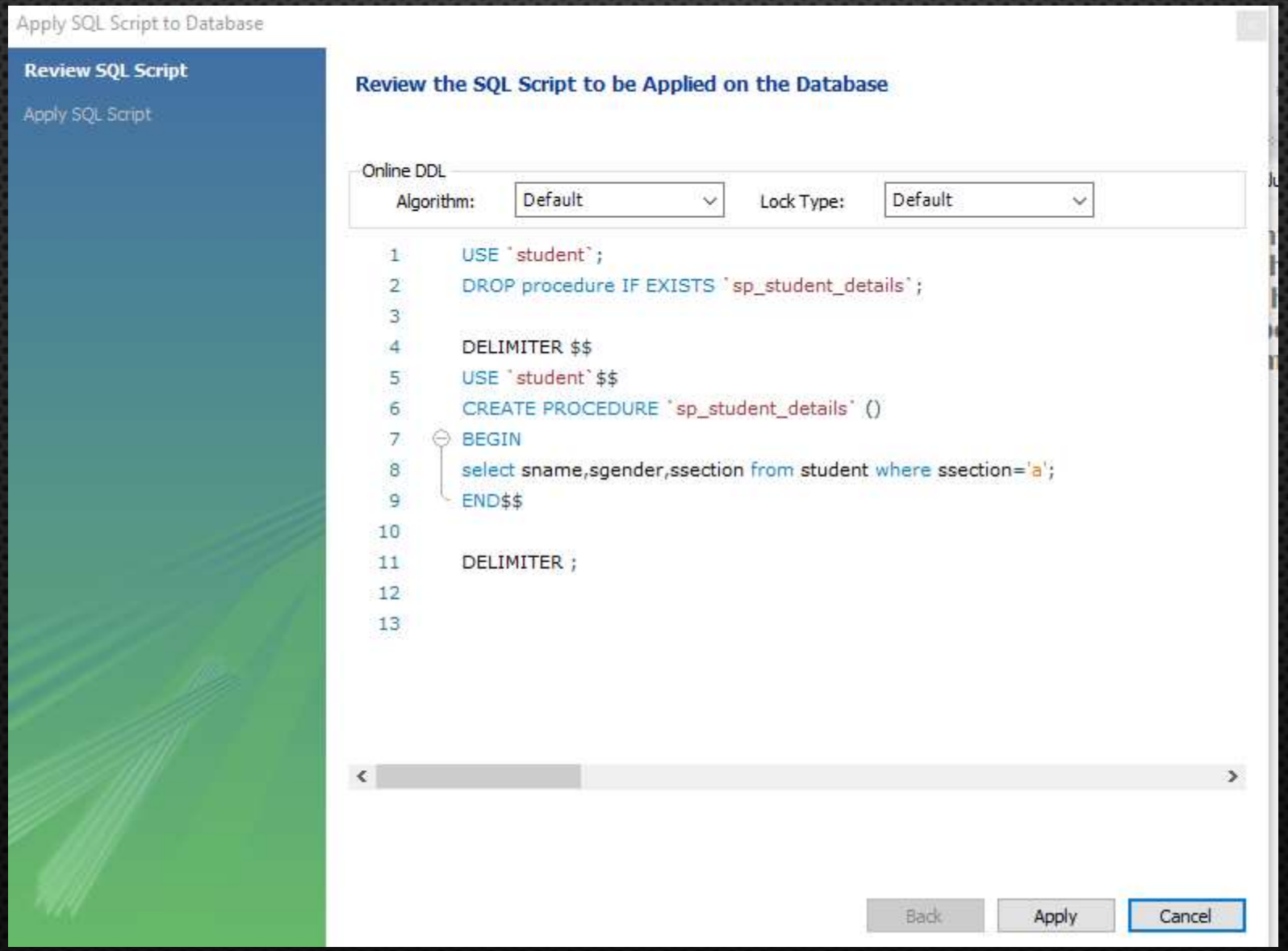
```
mysql> DELIMITER $$
mysql> CREATE PROCEDURE FIRST()
    -> BEGIN
    -> SELECT BRANCH,SNAME FROM COLLEGEDB;
    -> END $$
Query OK, 0 rows affected (0.63 sec)

mysql> DELIMITER ;
mysql> CALL FIRST();
+--------+-------+
| BRANCH | SNAME |
+--------+-------+
| cse    | aa    |
| cse    | bb    |
| cse    | zz    |
| ece    | dd    |
| ece    | ff    |
+--------+-------+
5 rows in set (0.02 sec)
```

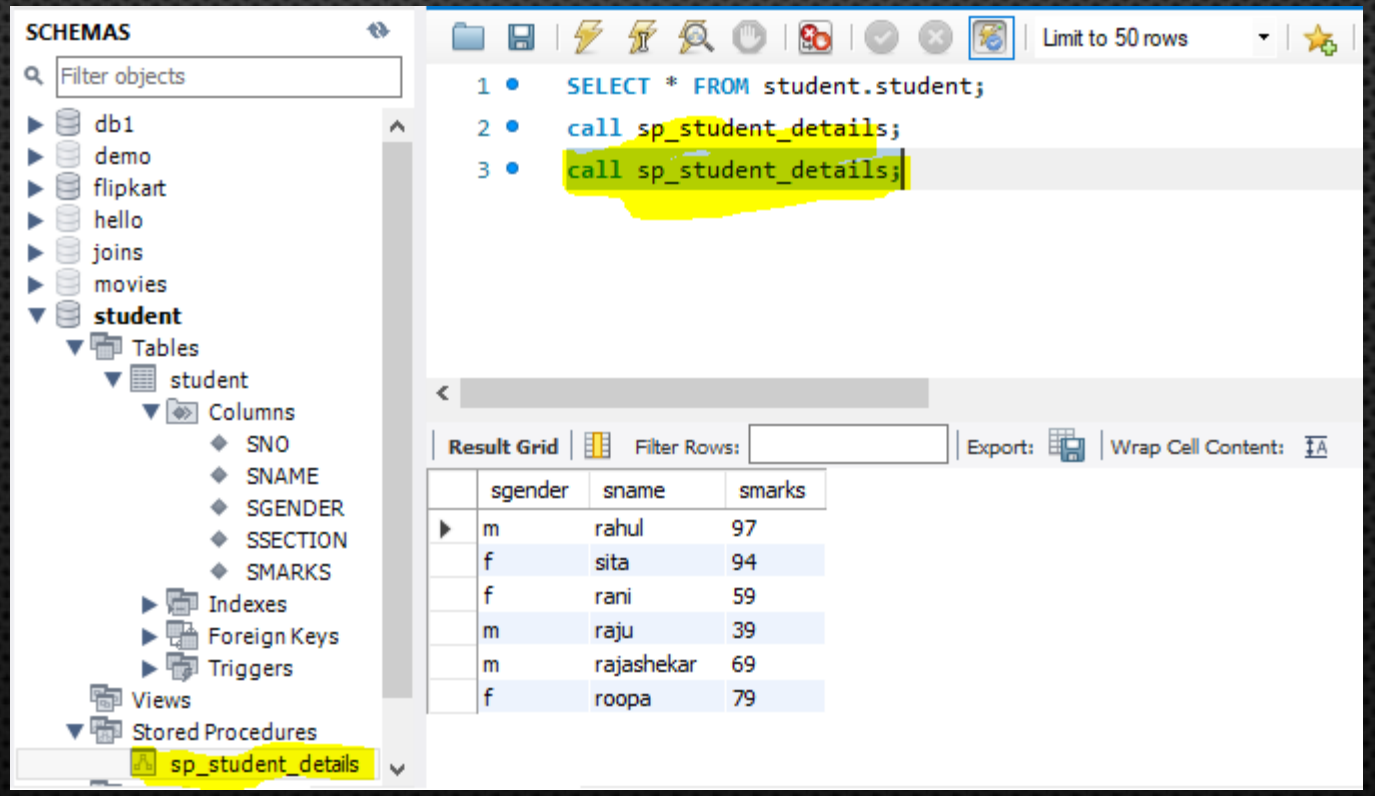
```
mysql> DELIMITER %%
mysql> CREATE PROCEDURE SECOND(A INT)
    -> BEGIN
    -> SELECT * FROM COLLEGEDB WHERE SID=A;
    -> END %%
Query OK, 0 rows affected (0.23 sec)

mysql> DELIMTER ;
    -> ^C
mysql> DELIMITER ;
mysql> CALL SECOND(4);
+--------+------+-------+------+
| branch | sid  | sname | sage |
+--------+------+-------+------+
| ece    |    4 | dd    |  100 |
+--------+------+-------+------+
1 row in set (0.02 sec)

Query OK, 0 rows affected (0.02 sec)
```

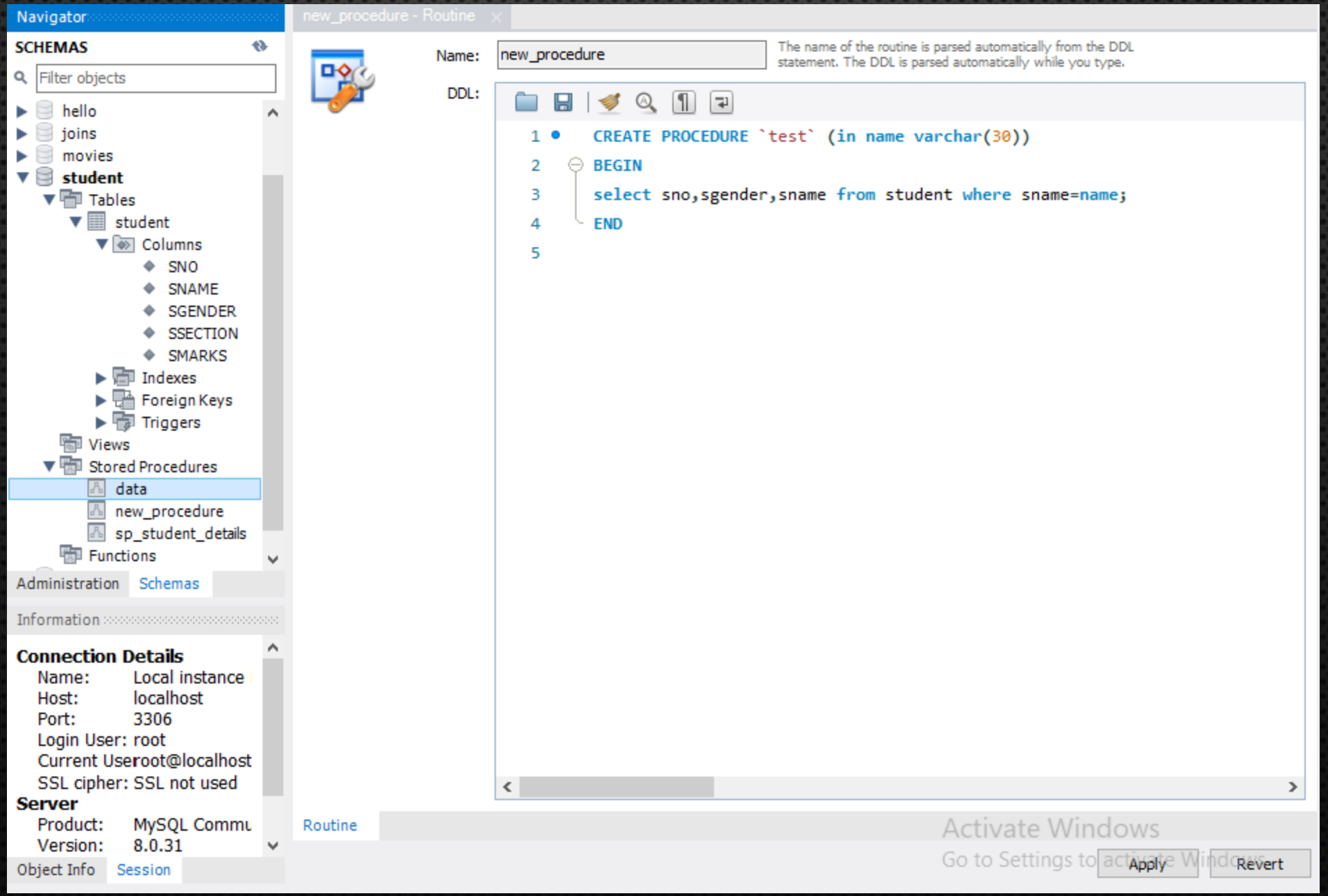
```
mysql> DELIMITER ;
mysql> DROP PROCEDURE IF EXISTS TWO;
Query OK, 0 rows affected (0.45 sec)
```

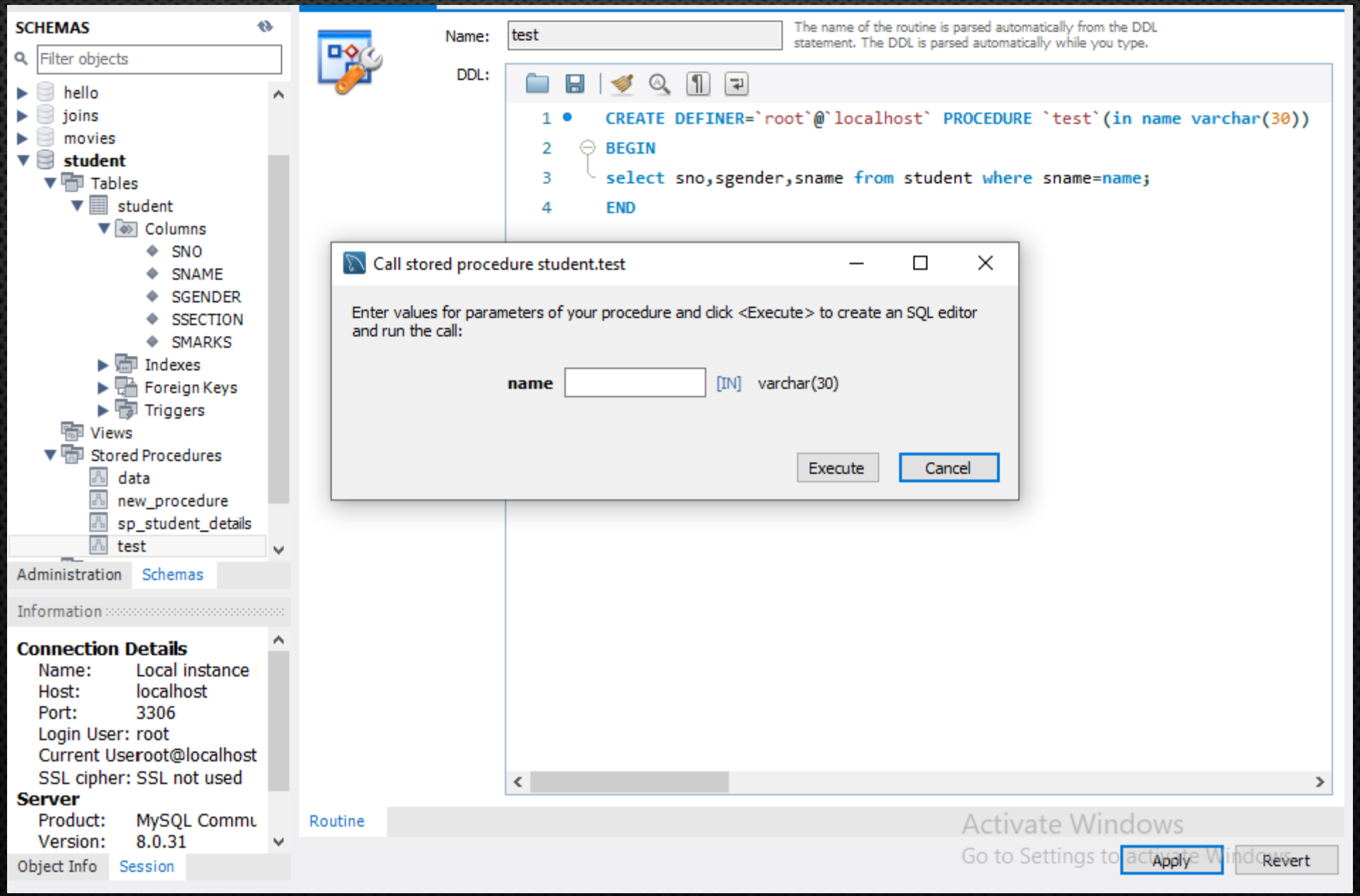

SCHEMAS
Filter objects
db1
demo
flipkart
hello
joins
movies
student
Tables
student
Columns
Indexes
Foreign Keys
Triggers
Views
Stored Procedures
Functions
sys
Administration
Schemas
Limit to 50 rows
1 SELECT * FROM student.student;
Result Grid
Filter Rows:
Edit:
Export/Import:
SNO
SNAME
SGENDER
SSECTION
SMARKS
NULL

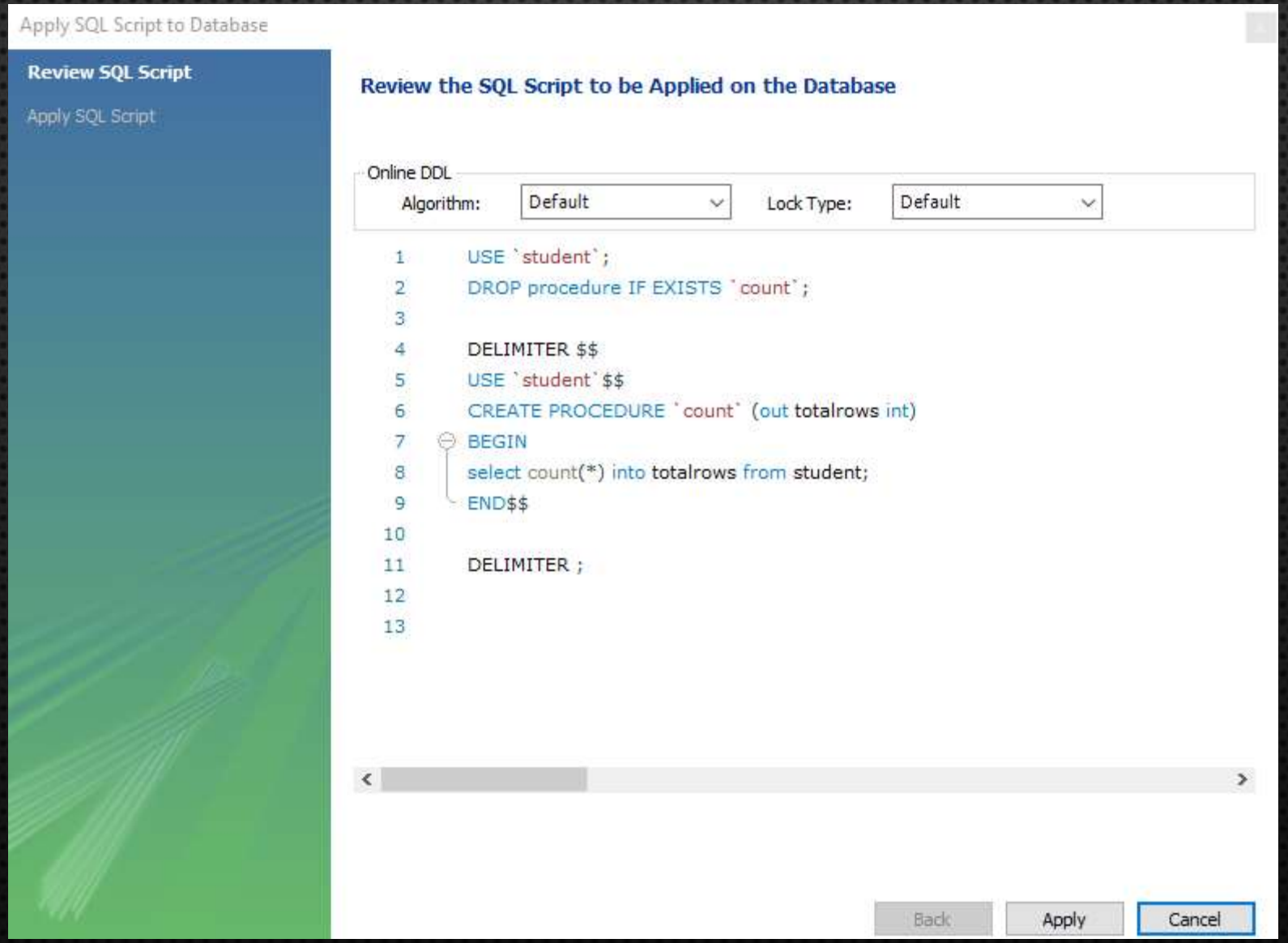

Navigator
SCHEMAS
Filter objects
db1
demo
flipkart
hello
joins
movies
student
Tables
student
Columns
Indexes
Foreign Keys
Triggers
Views
Stored Procedures
Functions
sys
Administration
Schemas
Information
Table: student
employee
student - Table
student
Limit to 50 rows
1 SELECT * FROM student.student;
Result Grid
Filter Rows:
Edit:
SNO SNAME SGENDER SSECTION SMARKS
1 rahul m a 97
2 sita f a 94
3 rani f b 59
4 raju m a 39
5 rajash... m b 69
6 roopa f b 79
NULL NULL NULL NULL NULL

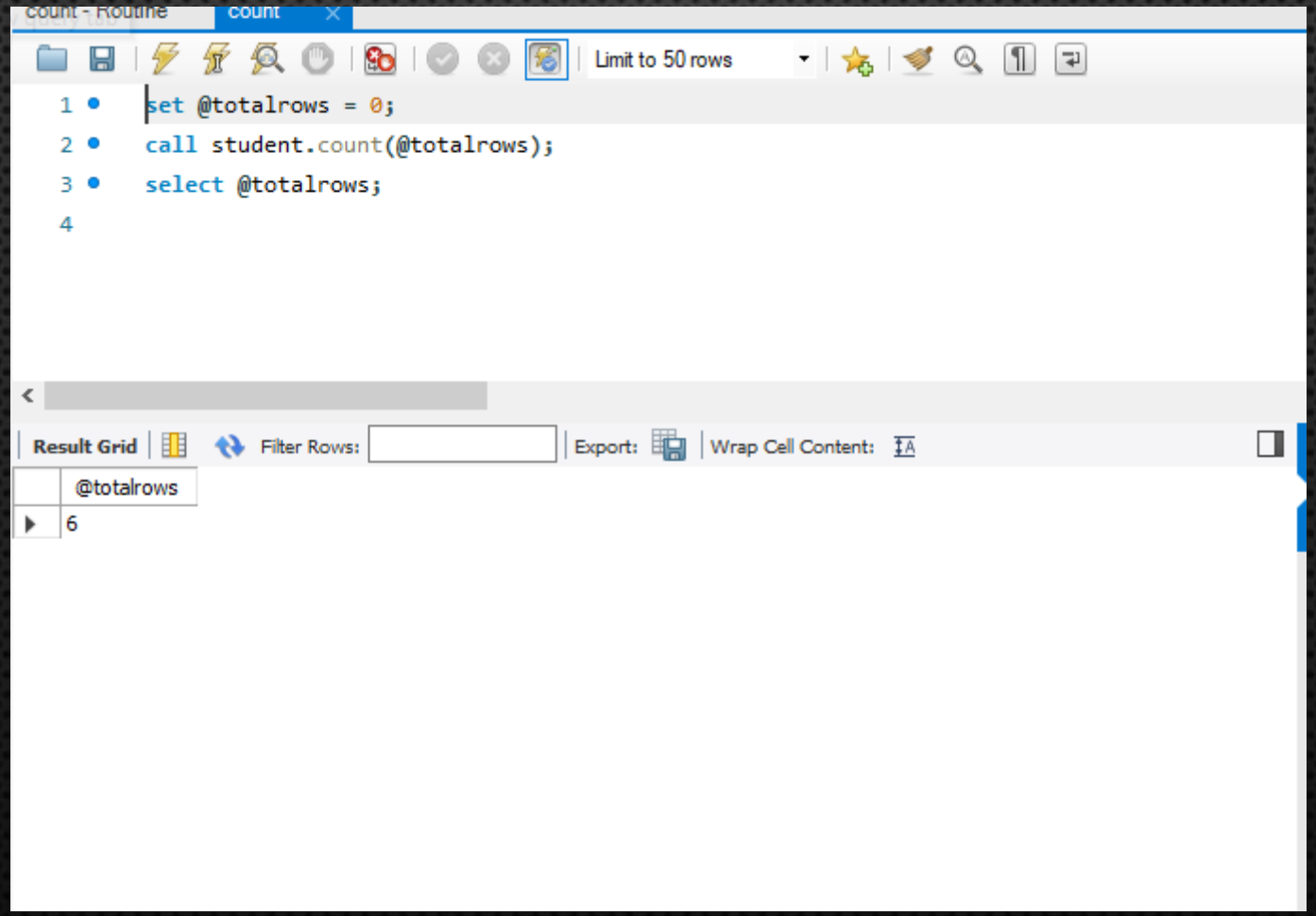

Apply SQL Script to Database
Review SQL Script
Apply SQL Script
Review the SQL Script to be Applied on the Database
Online DDL
Algorithm:
Default
Lock Type:
Default
1 USE `student`;
2 DROP procedure IF EXISTS `sp_student_details`;
3
4 DELIMITER $$
5 USE `student`$$
6 CREATE PROCEDURE `sp_student_details` ()
7 BEGIN
8 select sname,sgender,ssection from student where ssection='a';
9 END$$
10
11 DELIMITER ;
12
13
Back
Apply
Cancel

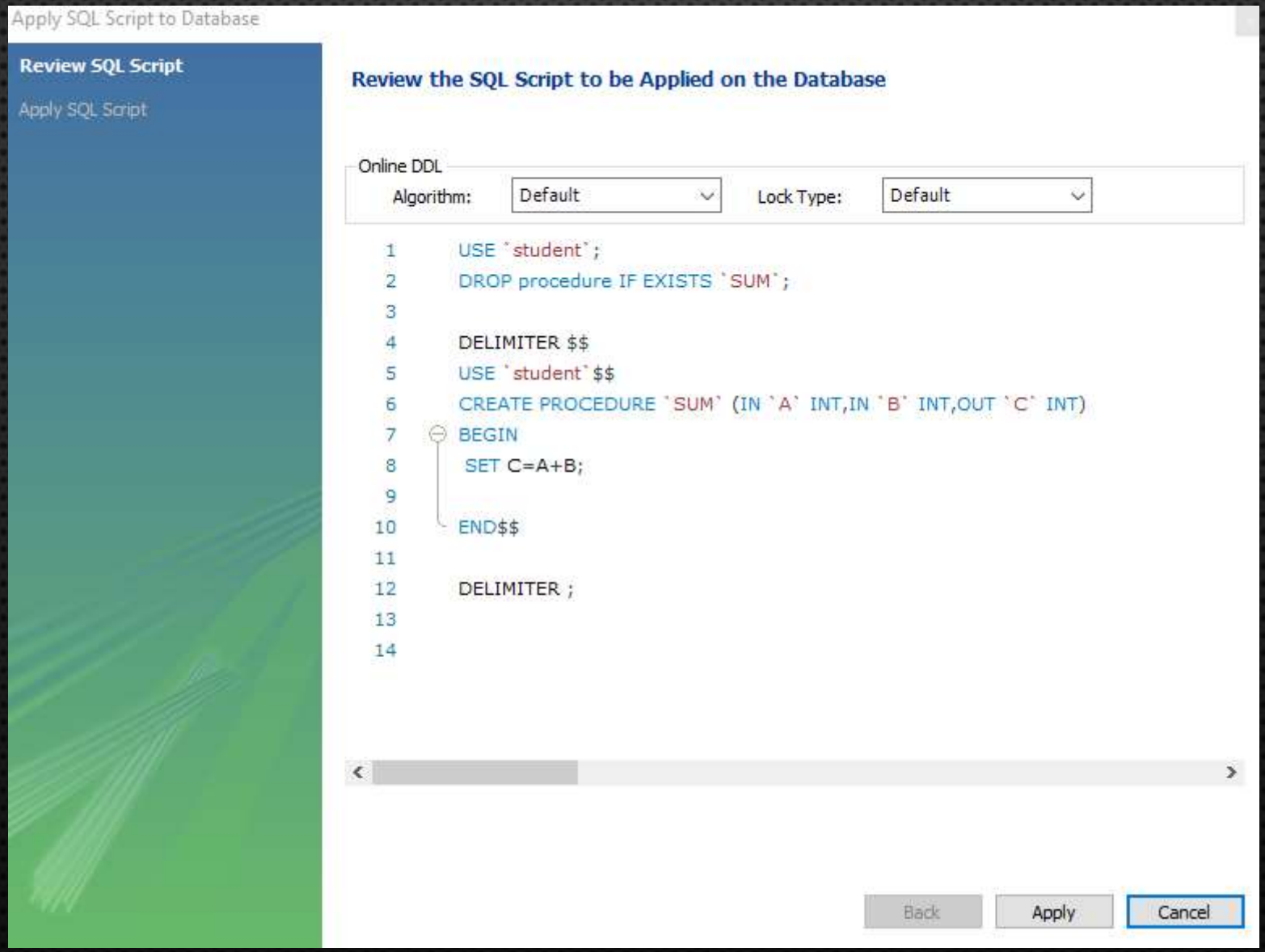

SCHEMAS
Filter objects
db1
demo
flipkart
hello
joins
movies
student
Tables
student
Columns
SNO
SNAME
SGENDER
SSECTION
SMARKS
Indexes
Foreign Keys
Triggers
Views
Stored Procedures
sp_student_details
Limit to 50 rows
1 SELECT * FROM student.student;
2 call sp_student_details;
3 call sp_student_details;
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
sgender sname smarks
m rahul 97
f sita 94
f rani 59
m raju 39
m rajashekar 69
f roopa 79

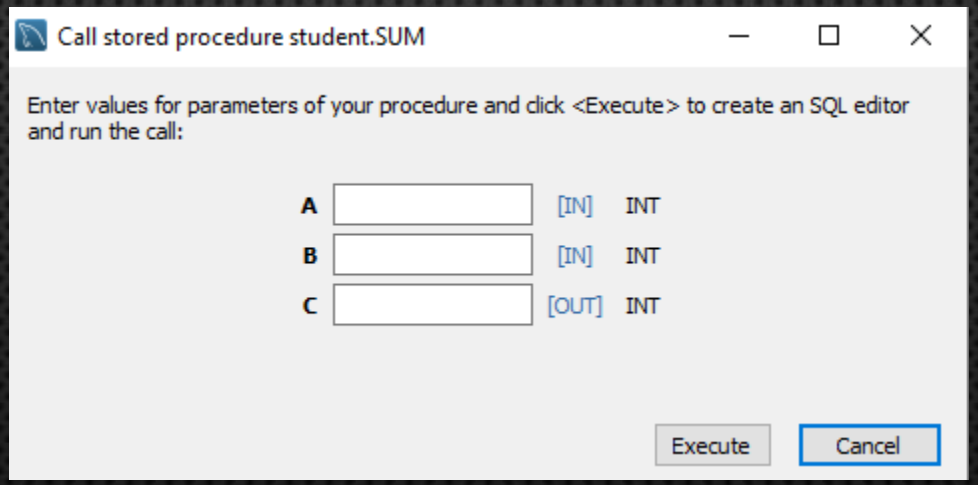

Navigator
SCHEMAS
Filter objects
hello
joins
movies
student
Tables
student
Columns
SNO
SNAME
SGENDER
SSECTION
SMARKS
Indexes
Foreign Keys
Triggers
Views
Stored Procedures
data
new_procedure
sp_student_details
Functions
Administration
Schemas
Information
Connection Details
Name: Local instance
Host: localhost
Port: 3306
Login User: root
Current User root@localhost
SSL cipher: SSL not used
Server
Product: MySQL Commu
Version: 8.0.31
Object Info
Session
new_procedure - Routine
Name: new_procedure
The name of the routine is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.
DDL:
1 CREATE PROCEDURE `test` (in name varchar(30))
2 BEGIN
3 select sno,sgender,sname from student where sname=name;
4 END
5
Routine
Activate Windows
Go to Settings to activate Windows.
Apply
Revert

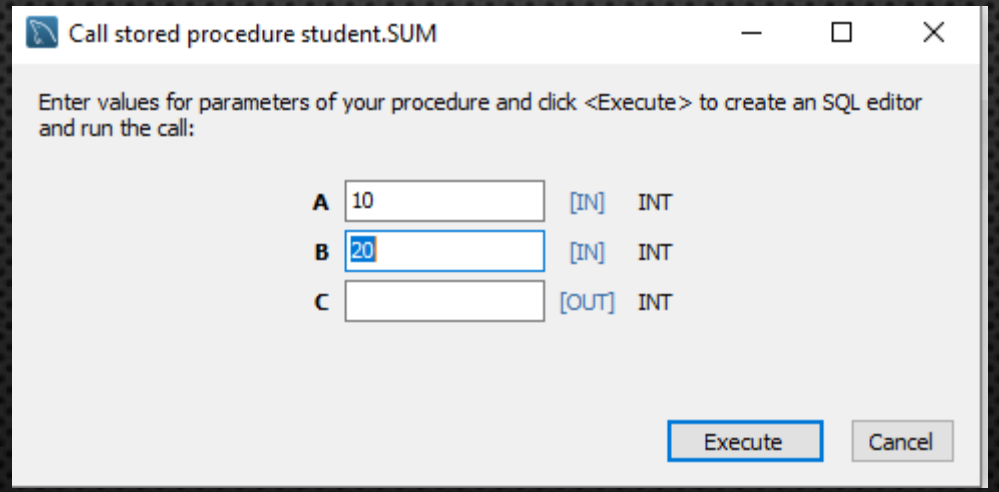

SCHEMAS
Filter objects
hello
joins
movies
student
Tables
student
Columns
SNO
SNAME
SGENDER
SSECTION
SMARKS
Indexes
Foreign Keys
Triggers
Views
Stored Procedures
data
new_procedure
sp_student_details
test
Administration
Schemas
Information
Connection Details
Name: Local instance
Host: localhost
Port: 3306
Login User: root
Current User root@localhost
SSL cipher: SSL not used
Server
Product: MySQL Commu
Version: 8.0.31
Object Info
Session
Name: test
The name of the routine is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.
DDL:
1 CREATE DEFINER=`root`@`localhost` PROCEDURE `test`(in name varchar(30))
2 BEGIN
3 select sno,sgender,sname from student where sname=name;
4 END
Call stored procedure student.test
Enter values for parameters of your procedure and click <Execute> to create an SQL editor and run the call:
name
[IN] varchar(30)
Execute
Cancel
Routine
Activate Windows
Go to Settings to activate Windows.
Apply
Revert

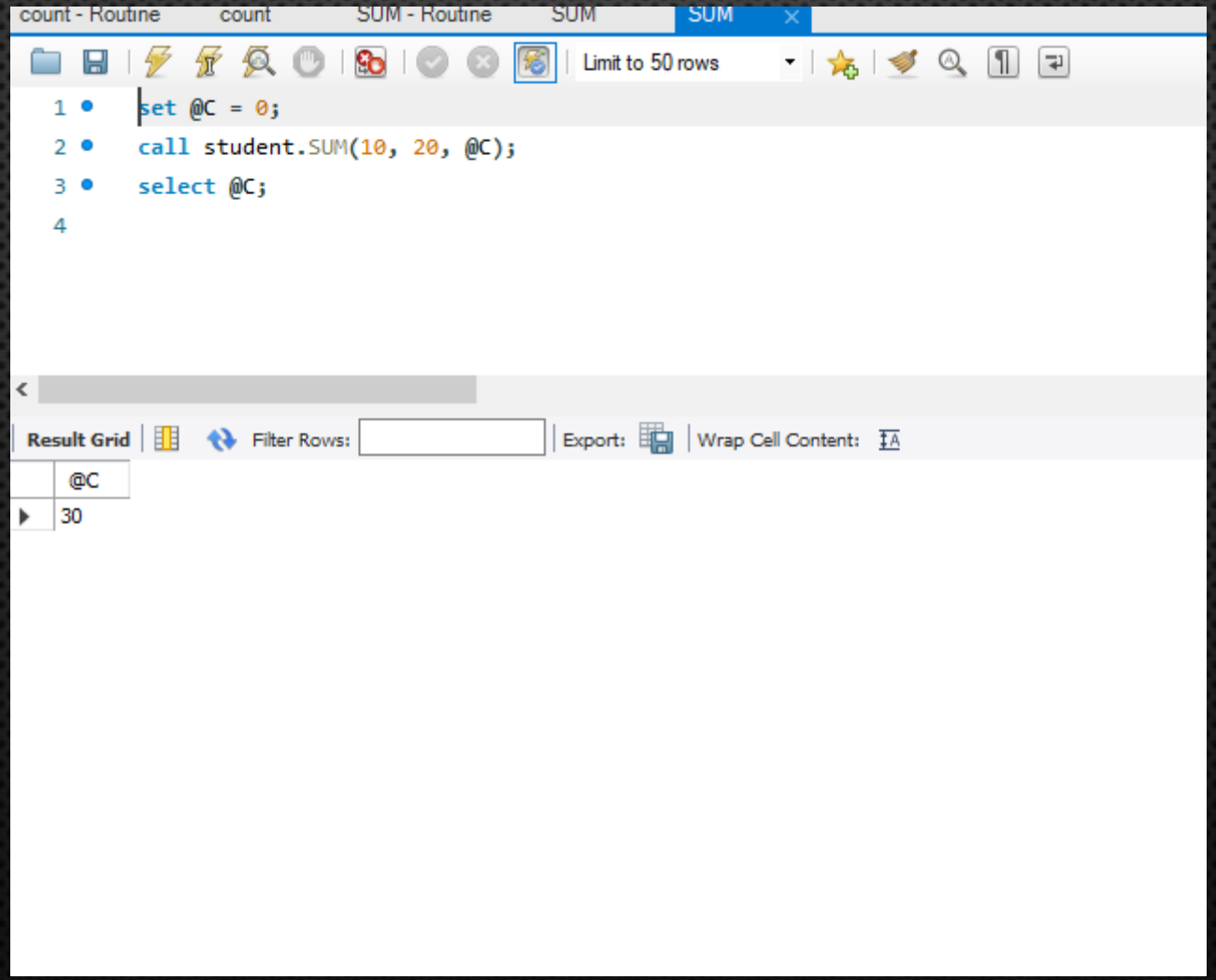

Apply SQL Script to Database
Review SQL Script
Apply SQL Script
Review the SQL Script to be Applied on the Database
Online DDL
Algorithm:
Default
Lock Type:
Default
1 USE `student`;
2 DROP procedure IF EXISTS `count`;
3
4 DELIMITER $$
5 USE `student`$$
6 CREATE PROCEDURE `count` (out totalrows int)
7 BEGIN
8 select count(*) into totalrows from student;
9 END$$
10
11 DELIMITER ;
12
13
Back
Apply
Cancel


count - Routine
count
Limit to 50 rows
1 set @totalrows = 0;
2 call student.count(@totalrows);
3 select @totalrows;
4
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
@totalrows
6


Apply SQL Script to Database
Review SQL Script
Apply SQL Script
Review the SQL Script to be Applied on the Database
Online DDL
Algorithm:
Default
Lock Type:
Default
1 USE `student`;
2 DROP procedure IF EXISTS `SUM`;
3
4 DELIMITER $$
5 USE `student`$$
6 CREATE PROCEDURE `SUM` (IN `A` INT,IN `B` INT,OUT `C` INT)
7 BEGIN
8 SET C=A+B;
9
10 END$$
11
12 DELIMITER ;
13
14
Back
Apply
Cancel


Call stored procedure student.SUM
Enter values for parameters of your procedure and click <Execute> to create an SQL editor and run the call:
A
[IN]
INT
B
[IN]
INT
C
[OUT]
INT
Execute
Cancel


Call stored procedure student.SUM
Enter values for parameters of your procedure and click <Execute> to create an SQL editor and run the call:
A
10
[IN]
INT
B
20
[IN]
INT
C
[OUT]
INT
Execute
Cancel


count - Routine
count
SUM - Routine
SUM
SUM
Limit to 50 rows
1 set @C = 0;
2 call student.SUM(10, 20, @C);
3 select @C;
4
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
@C
30