DATA SOCIETY®

Introduction to SQL - day 1

"One should look for what is and not what he thinks should be."
-Albert Einstein.

Module completion checklist

Objective	Complete
Discuss what SQL is and what it is used for	
Summarize what databases are and give examples	
Understanding the entity relationship diagram (ERD)	
Establish a database connection & creating a database	
Summarize DDL: data types and constraints	
Summarize DDL: create, update & delete table and data	

Reactivating questions

- What do you like/dislike about working with spreadsheets?
- What do you know about SQL?
- What would you like to learn to do with SQL?

Introduction to SQL

- **SQL** (Structured Query Language) is a query language used to query a relational database
- **SQL** is used to:
  - define the structure of the data
  - modify the data
  - define security constraints
- **SQL** has several parts:
  - **Data Definition Language (DDL)** is used to define the data structures stored in the database
  - **Data Manipulation Language (DML)** is used to manipulate the data structures previously defined
  - **Data Control Language (DCL)** is used to begin, end, and roll back transactions, access & privileges

Ultimately, SQL is the language you use to interact with a database

Why should you learn SQL?

- SQL is the most commonly used database language
- It powers database engines like **MySQL, SQL Server, SQLite, and PostgreSQL**
- Data stored in a relational database can be can be queried, modified, and manipulated easily with basic commands in SQL
- SQL is not a programming language, **it's a query language**
- SQL was created to give the people a way to get data from a database, without needing programming skills
- Its queries are basic English commands, which makes it easier for English speakers to write a SQL query

What is MySQL?

- **MySQL** is a popular open source database server
- Available for free
- Extremely simple to download and install on various platforms
- The MySQL website: ***https://www.mysql.com/***

What is a database?

- A **database** is a collection of interrelated data
- The primary goal is to store and retrieve information in a convenient and efficient way
- Databases are widely used
- For example, a telephone book is a database of
  - Names
  - Addresses
  - Phone numbers
- A university database has data on
  - Student details
  - Faculty information
  - Department data
  - Courses offered

Overview on how databases operate

Module completion checklist

Objective	Complete
Discuss what SQL is and what it is used for	✔
Summarize what databases are and give examples	✔
Understanding the entity relationship diagram (ERD)	
Establish a database connection & creating a database	
Summarize DDL: data types and constraints	
Summarize DDL: create, update & delete table and data	

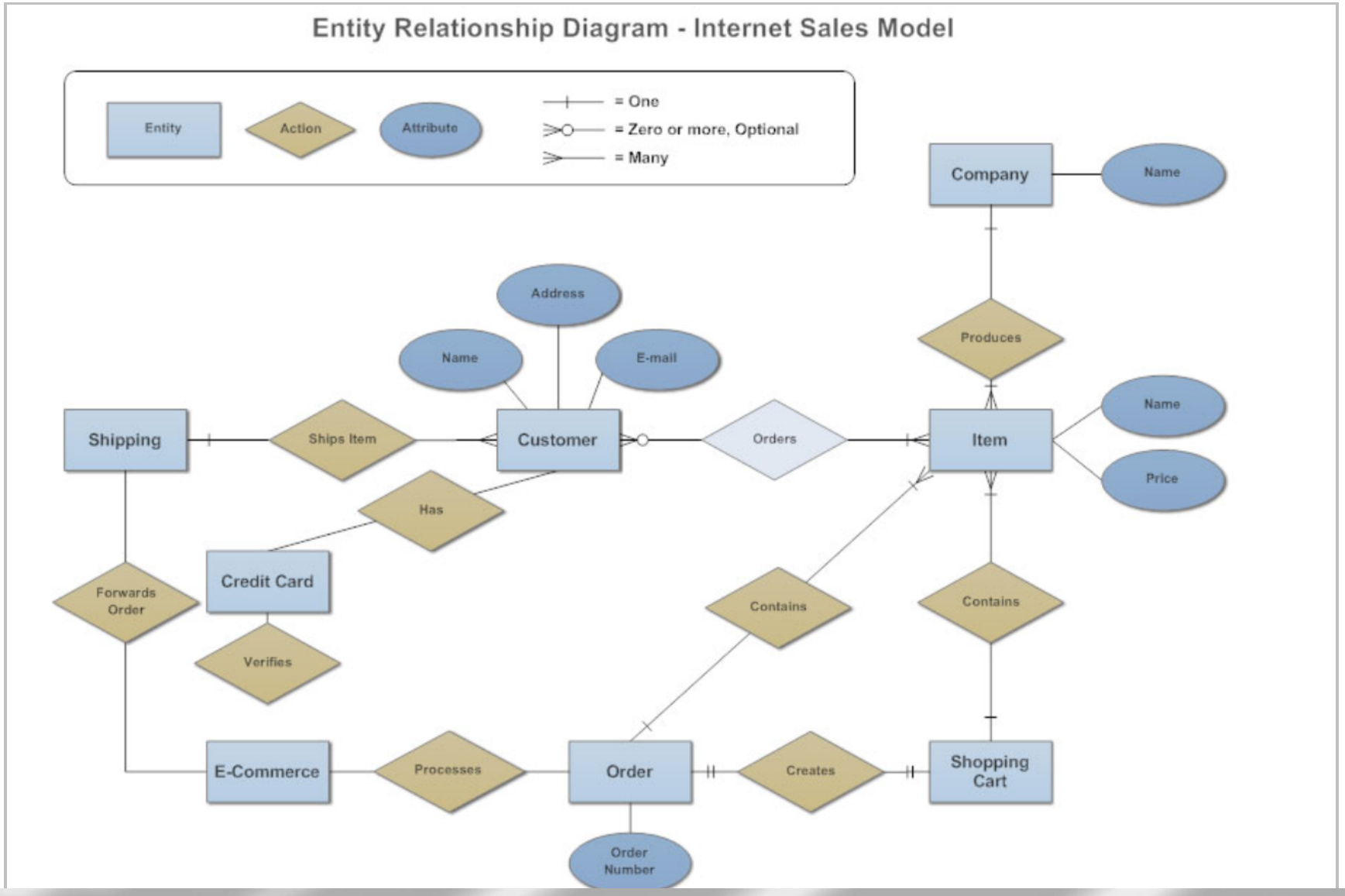
Entity relationship model

- An entity relationship model is also called an **ER model**
- It represents the conceptual view of the database
- It is the first step of designing the database before the actual implementation
- ER model is represented using **ER diagram**, a.k.a **ERD**
- Tools such as **MS Visio**, **Rational Rose**, **LucidCharts** are used for drawing the ERD

Components of ERD

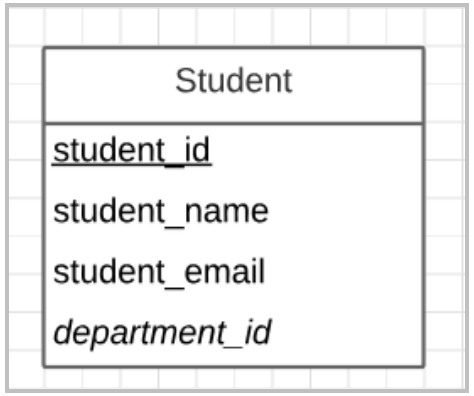
Term	Definition
Entity	Any real world object like person, place, event
Relationship	Link between the entities
Attributes	Properties or characteristics of the entity or relationship

A sample ER diagram



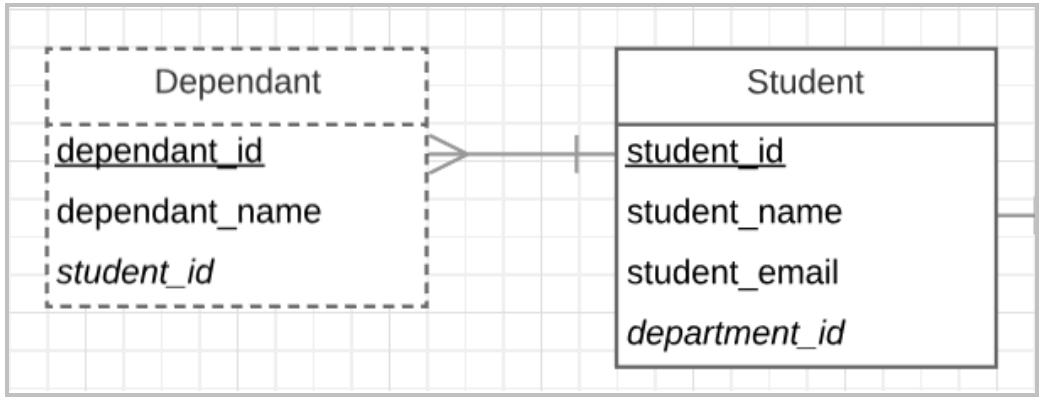
Entity

- Any identifiable real world object about which the organization wants to maintain data
- **Entity instance**
  - Corresponds to a single row in a table
  - Single occurrence of an entity
- **Entity type or entity set**
  - Collection of entities
  - Corresponds to a table



Strong vs weak entity

- **Strong entity**
  - Has its own identifier
  - Exists independent of other entities
  - Represented by a solid line
- **Weak entity**
  - Has a partial identifier
  - Depends on another entity
  - Represented by a double or dashed line



Required vs optional attributes

- **Required attribute**
  - Has a value for every single entity instance it is associated with
  - Example: `student_id` is required in the university database
- **Optional attribute**
  - May not have a value for every single entity instance
  - Example: `middle_name` may / may not be present for a student

Simple vs composite attributes

- **Simple attribute**
  - Atomic, cannot be divided further
  - Example: `student_phone` is atomic 10 digit number which cannot be broken into components
- **Composite attribute**
  - Can be broken into meaningful components
  - Example: `address` can be broken into `street_name`, `city`, `state`

Single-valued vs multi-valued attributes

- **Single-valued attribute**
  - Represented by a single value only
  - Example: `ssn_number` is unique, a student cannot have more than 1
- **Multi-valued attribute**
  - May be represented by multiple values
  - Example: a student may have multiple instances of `mobile_number`

Stored vs derived attributes

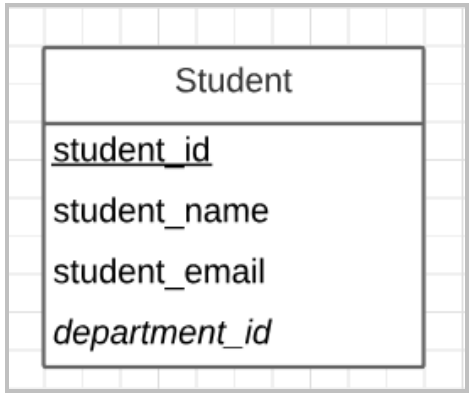
- **Stored attribute**
  - Attributes that are defined and stored
  - Example: `dateofBirth` is stored
- **Derived attribute**
  - Obtained from another attribute
  - Example: `age` is derived from `dateofBirth`

Identifier attributes

- An attribute (or attributes) that uniquely identifies an entity instance is also called a **key**
- **Simple key**
  - Single attribute that identifies an entity instance uniquely
- **Candidate key**
  - When there are multiple columns acting as the primary key, the minimal subset of columns needed is called a *candidate key*

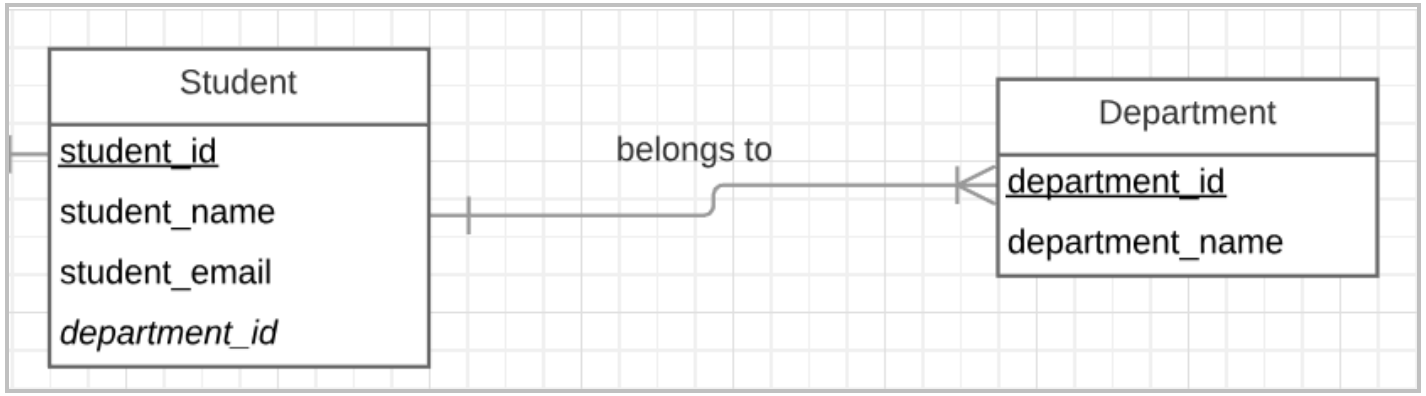
An identifier:

- Will not **change** its value
- Will not be **null**
  - In this student entity, **student_id** is a simple key



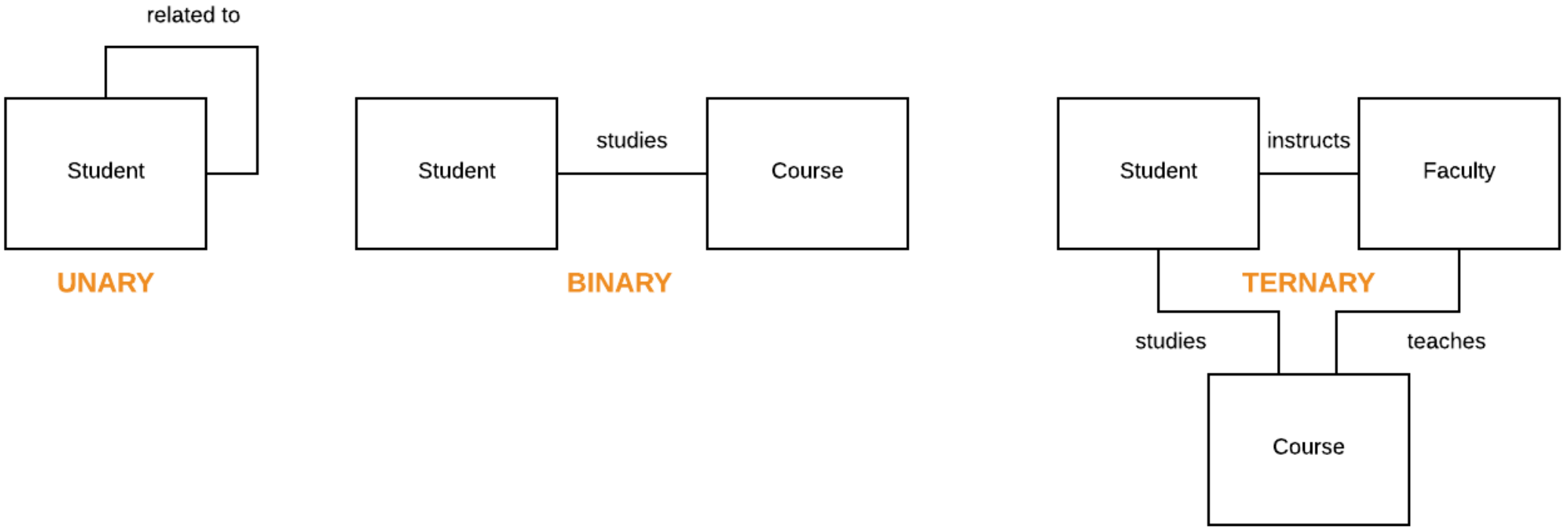
Relationship

- It is a link between entities that explains the connection between them
- Two entities can have **more than one** relationship between them
- **Relationship instance**
  - Association of entities where there is exactly one entity in each participating entity type
- **Relationship set**
  - Set of relationships of similar type between entity types
  - Set of associations



Degree of relationships

- **Unary** - one entity related to another of the **same** entity type
- **Binary** - entities of **two** different types related to each other
- **Ternary** - entities of **three** different types related to each other



Cardinality of relationships

- Cardinality determines how many instances of one entity relates to the instance of another entity
- **One to one**
  - Each entity in the relationship will have exactly one related entity
- **One to many**
  - An entity on one side can have many related entities on other side, but entity on other side can have exactly one entity
- **Many to many**
  - Entities of both sides can have many related entities on either side

Cardinality constraints

- A constraint is the number of instances of one entity that *can* or *must* be associated with each instance of other entity
- **Minimum cardinality**
  - If zero, then **optional**
  - If one or more, then **mandatory**
- **Maximum cardinality**
  - The maximum number of instances that one entity may associate to another entity

Overview on how databases operate

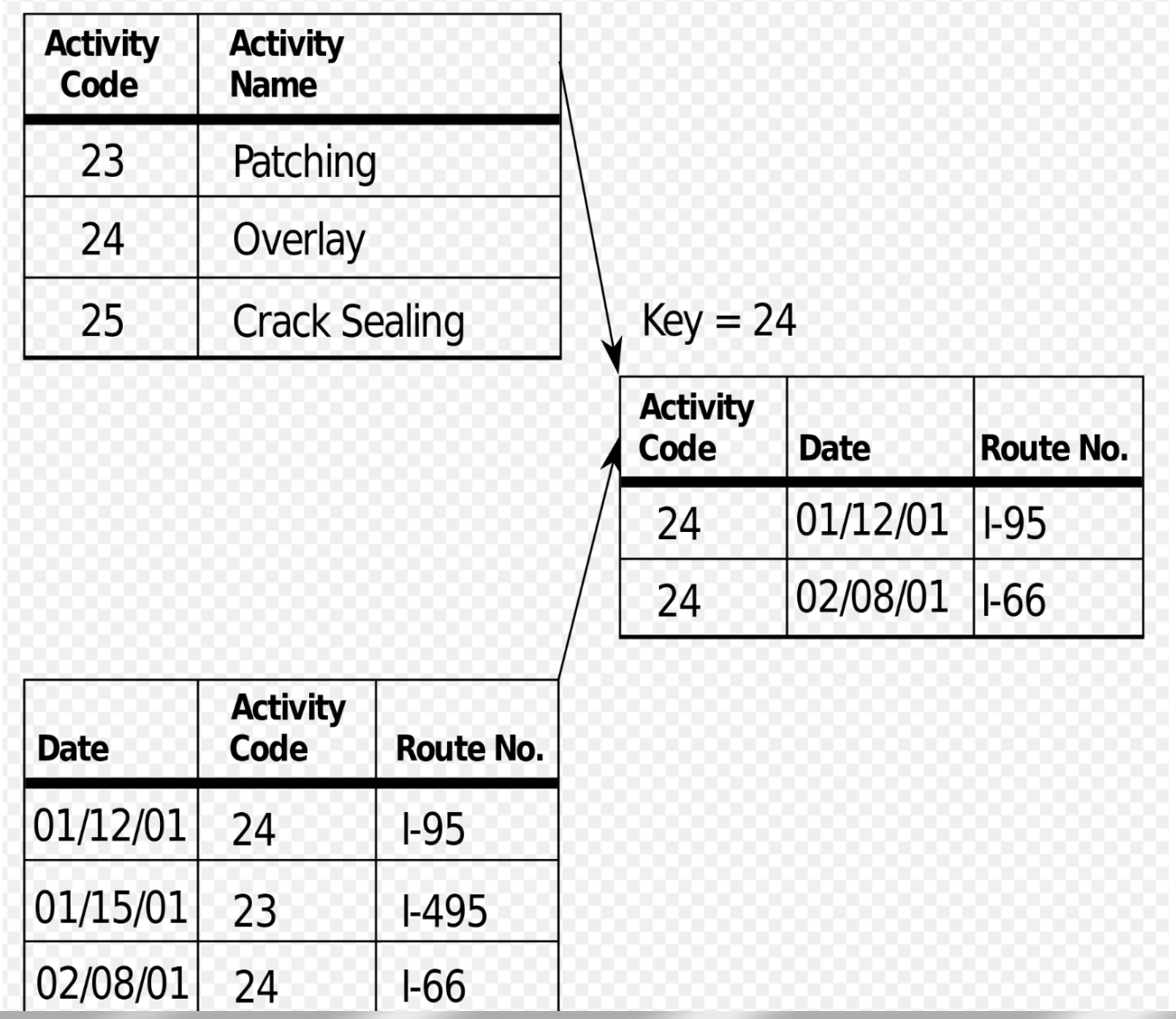
- Until now we saw how to design the logic behind the database
- Let's move on to the functional design of the database using relational database model

Relational database model

- A **relational database model** is a widely used database model in today's world
- It consists of a collection of tables having unique names to store information
- A structure of a database model with its tables, constraints, and relationships is called a **schema**

Term	Definition	Example
Table	collection of rows & columns	faculty (in university database)
Column a.k.a attributes	an individual piece of data stored in table	faculty_id, faculty_name, department_name, course_id
Rows a.k.a tuples	represent a single implicit structured data item in a table	dep_001, “Andrew”, “Economics”, “EG0932”
Primary key	one or more columns that uniquely identify each row in a table	faculty_id
Foreign key	one or more columns that is used to identify a row in another table	course_id

A sample relational database



Activity Code	Activity Name
23	Patching
24	Overlay
25	Crack Sealing

Key = 24

Activity Code	Date	Route No.
24	01/12/01	I-95
24	02/08/01	I-66

Date	Activity Code	Route No.
01/12/01	24	I-95
01/15/01	23	I-495
02/08/01	24	I-66

Knowledge check 1

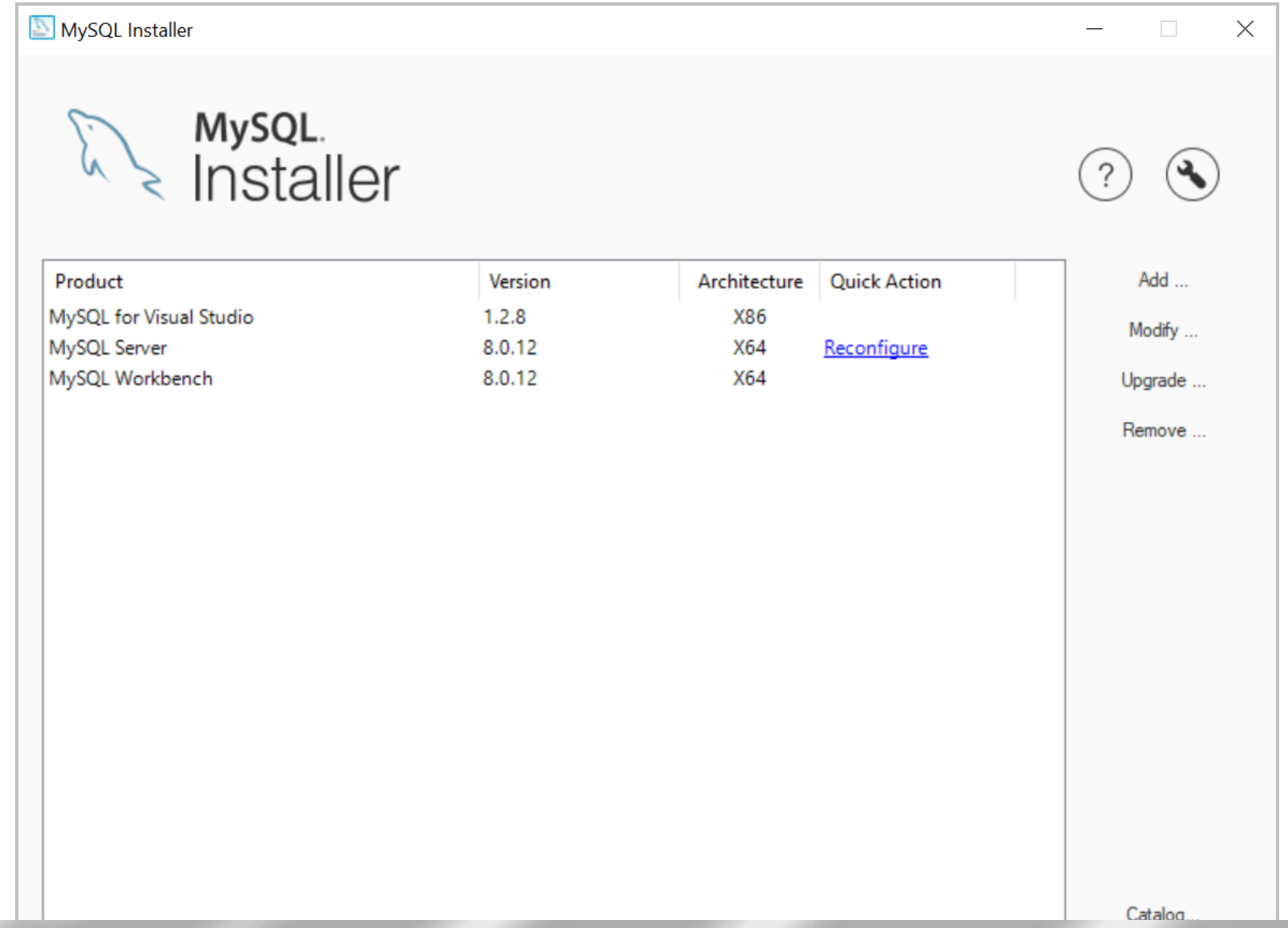
Module completion checklist

Objective	Complete
Discuss what SQL is and what it is used for	✔
Summarize what databases are and give examples	✔
Understanding the entity relationship diagram (ERD)	✔
Establish a database connection & creating a database	
Summarize DDL: data types and constraints	
Summarize DDL: create, update & delete table and data	

Creating our first database

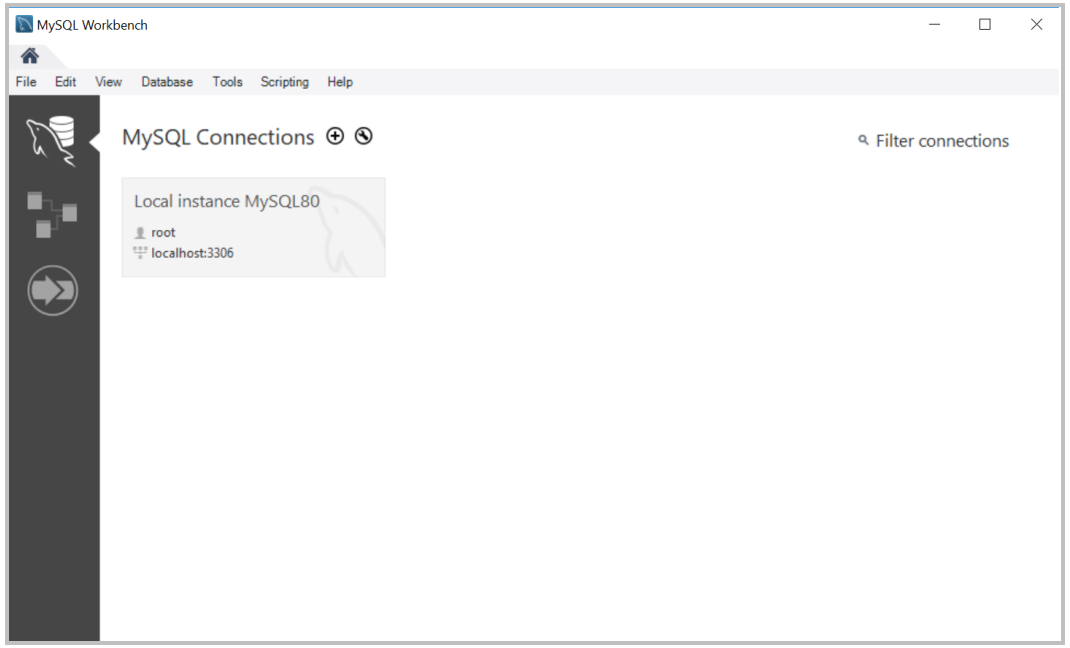
- In this module, we will be creating a `university` database
- We will use this database to create tables to suit the needs of the academic administrator working at your university
- The academic administrator needs:
  - Courses to be structured accordingly in tables and columns for easy access
  - The database in a way that someone with no SQL background could easily pull course data

MySQL workbench & server installed



Establish a database connection

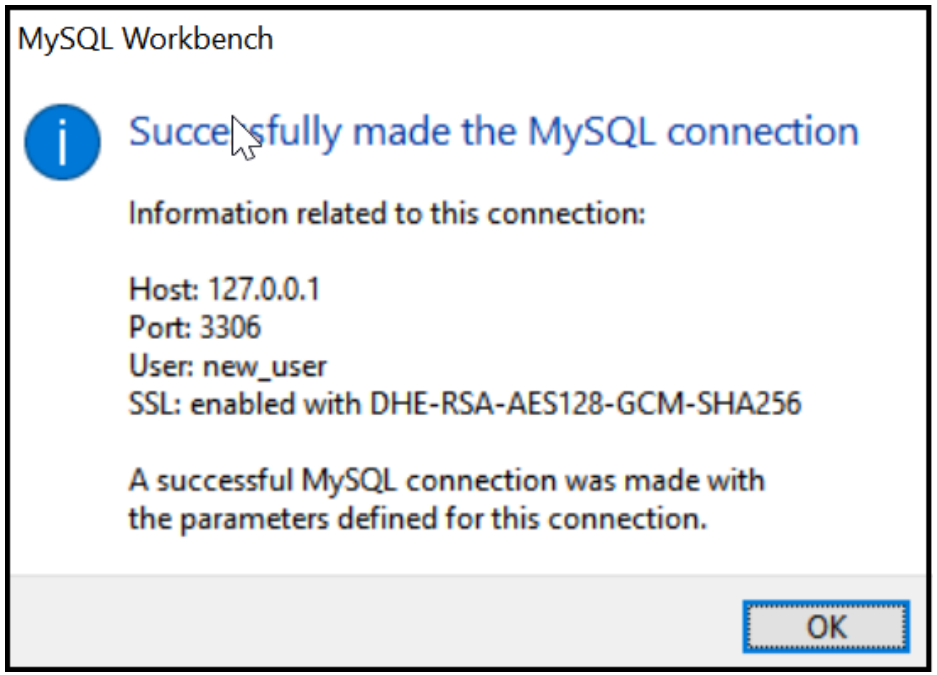
- An instance of MySQL Server must be installed, started and accessible by now
- To create a new database, there should be a connection to the server
- We are going to establish a connection to a new user
- Launch the MySQL Workbench from home
- There will not be any connection initially, except for `root`
- Click on the **'+'** symbol next to the **MySQL Connections**



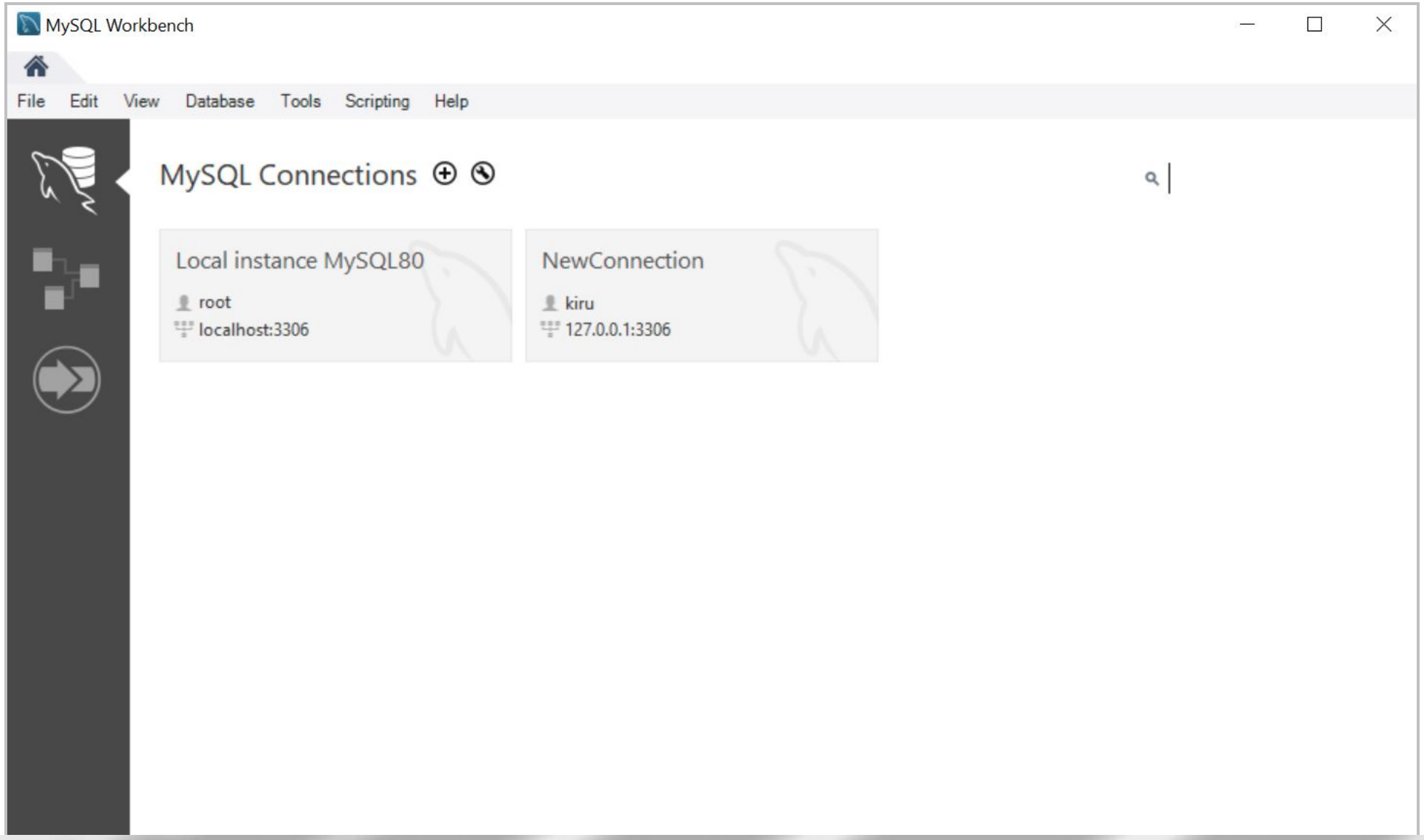
Establish a database connection

Test the new connection

- Test the established connection

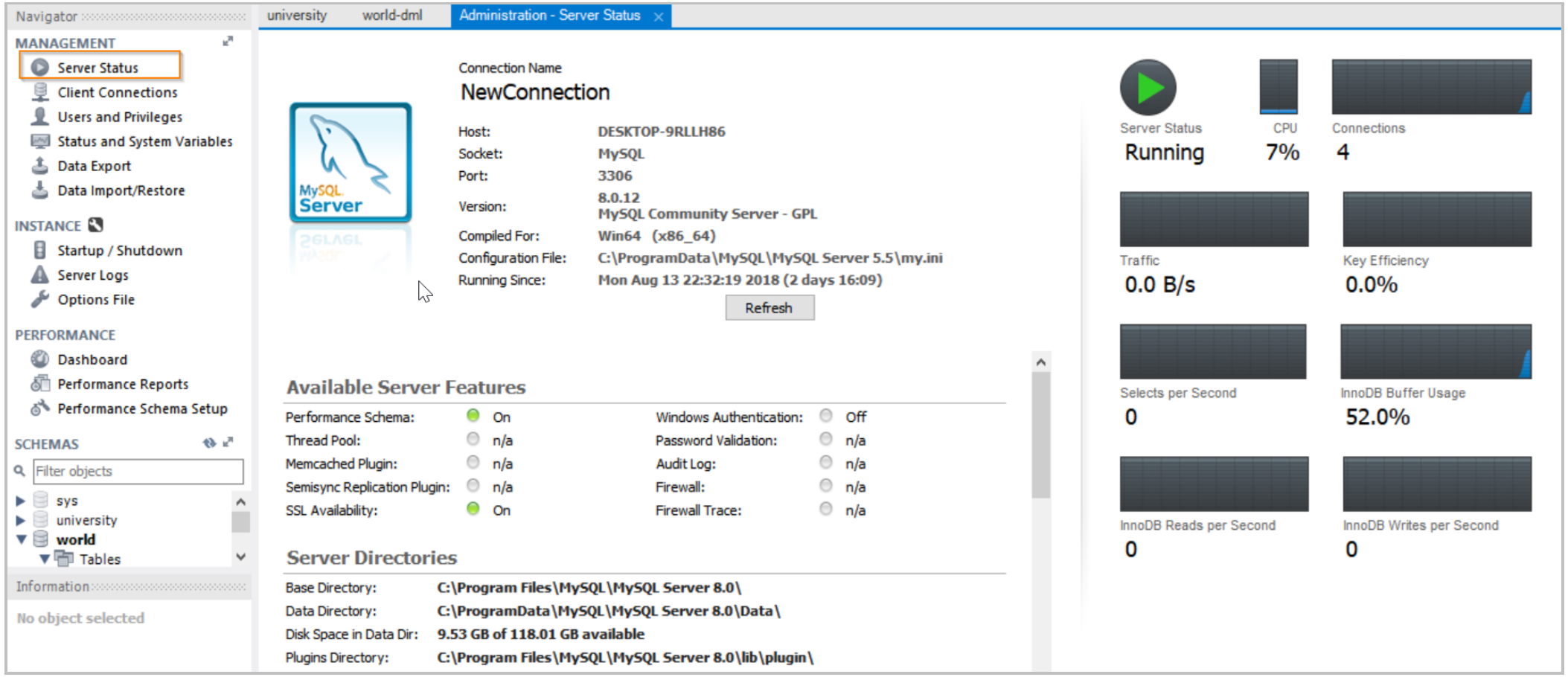


New connection is successfully created

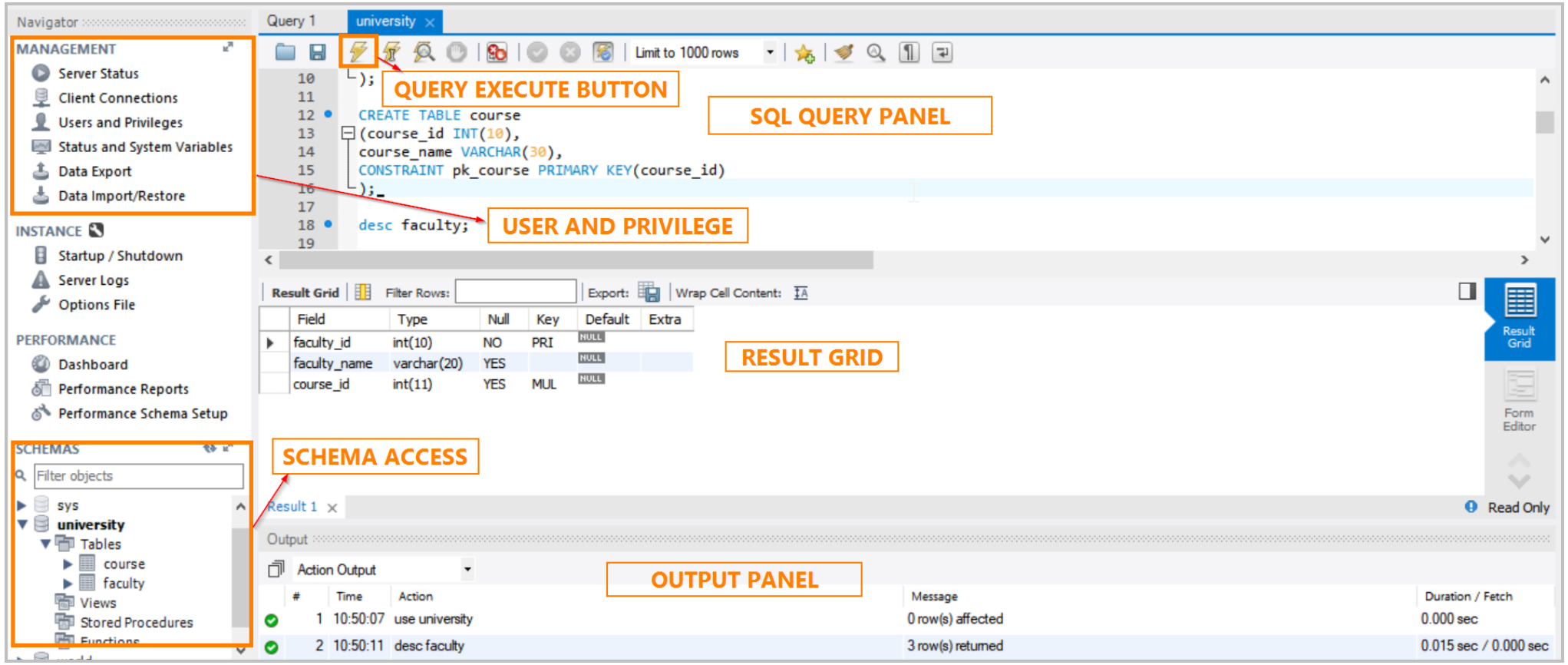


New connection is successfully created

- Check the **Server Status** to see that the server is running and connection is successful

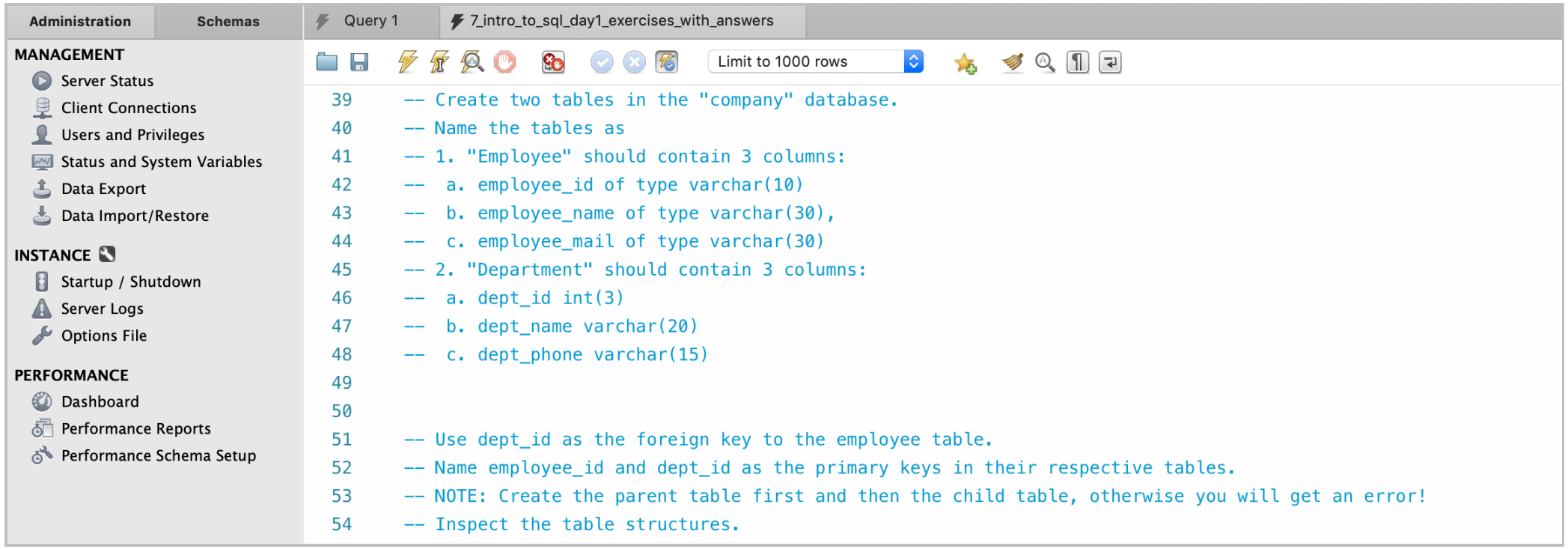


MySQL workbench overview



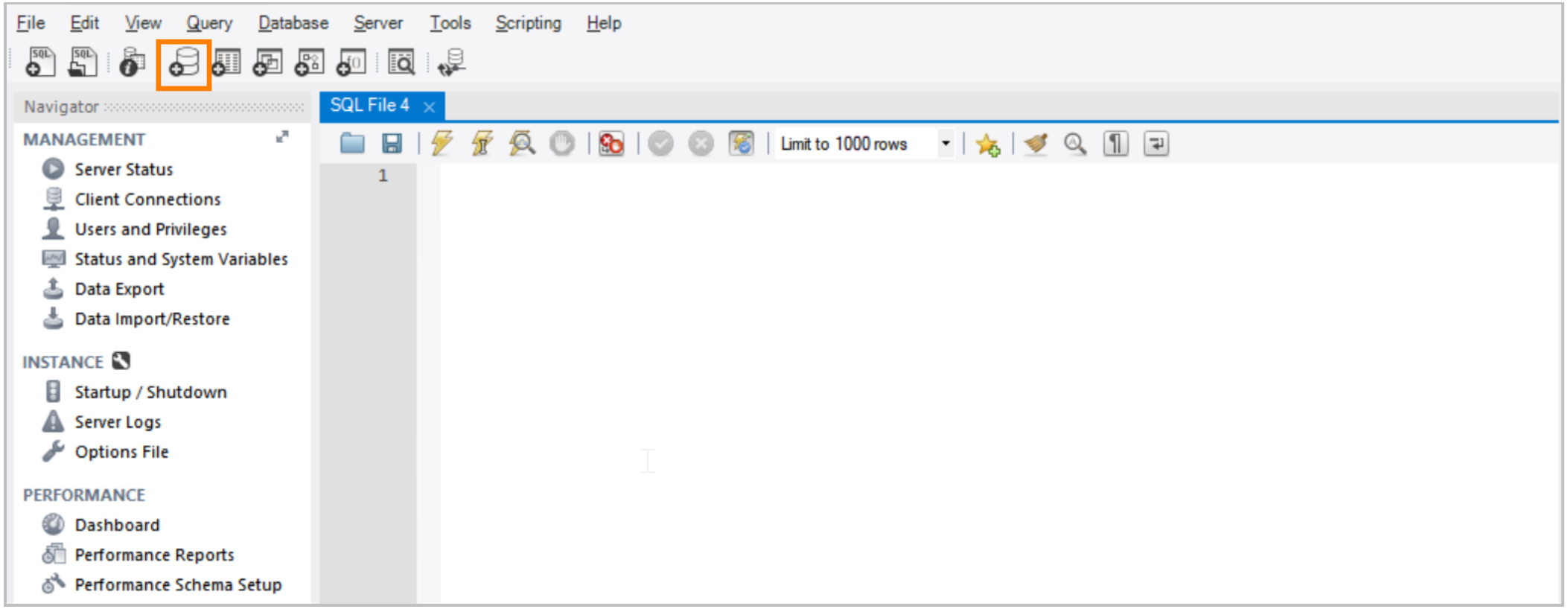
MySQL workbench overview

- Sometimes the schema tab can be next to admin tab rather than below the admin section



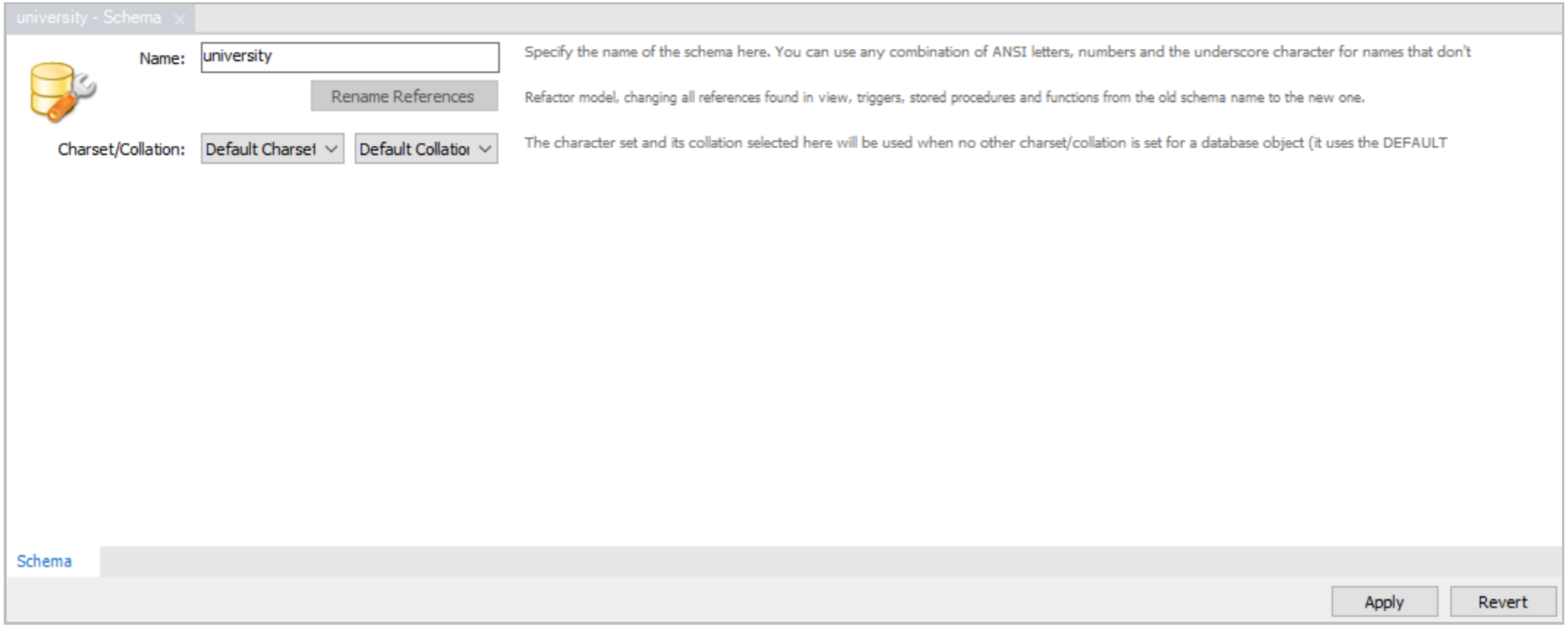
Create a new database

- Click on the **cylinder** symbol to create a new database



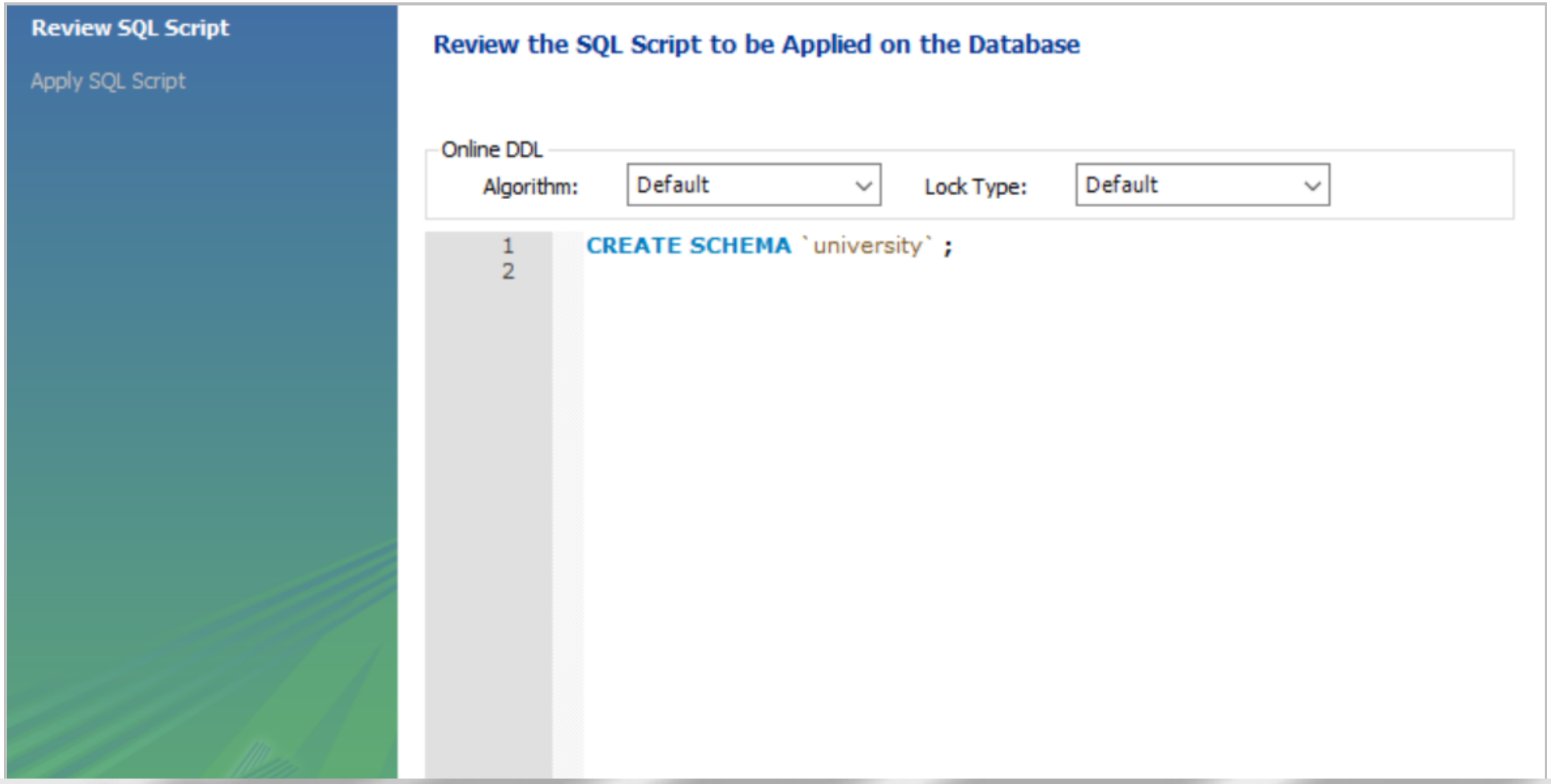
Create a new database

- Type in the database name and click **apply**

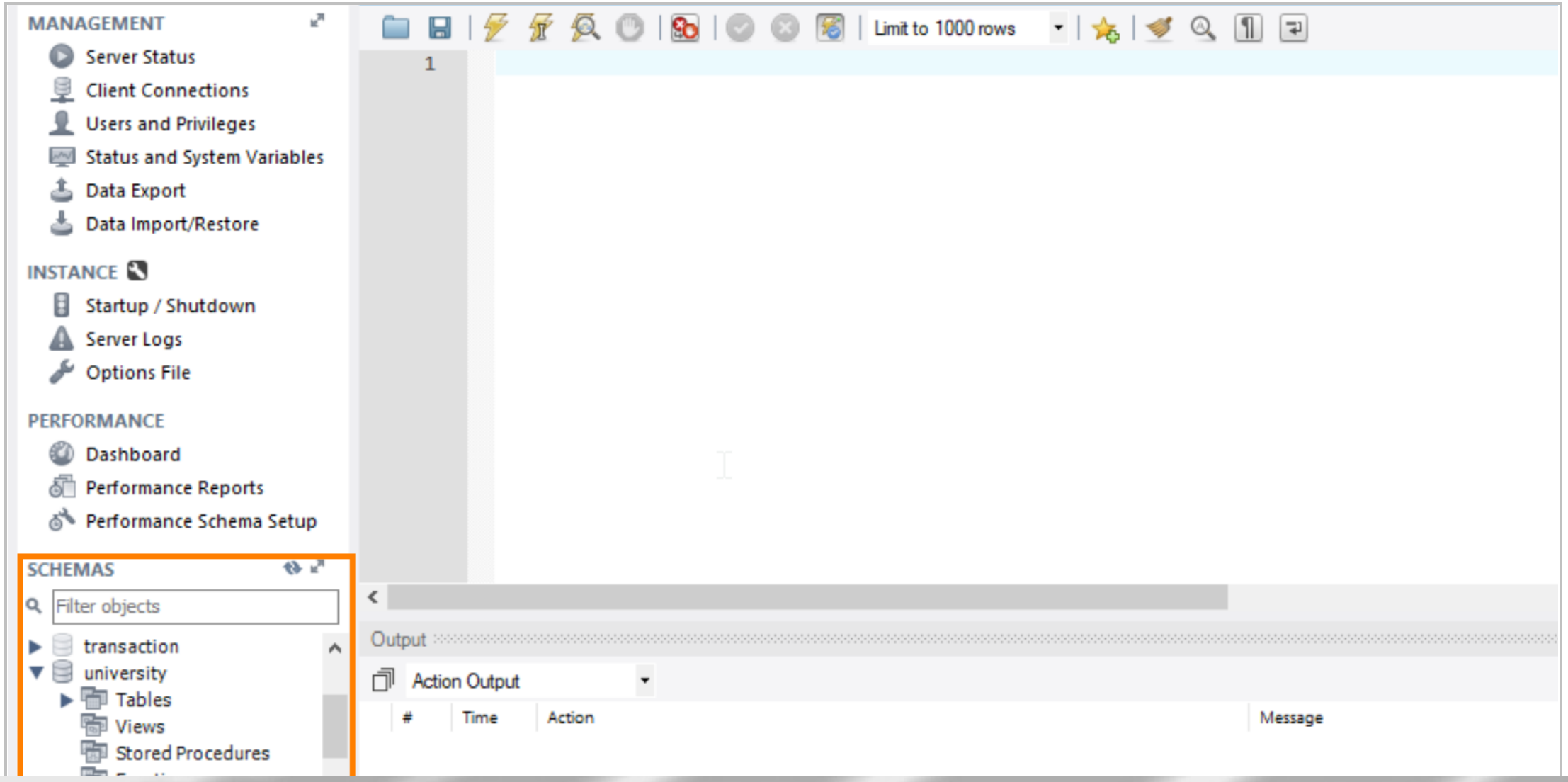


Create a new database

- Once you click **apply**, the code is auto generated by the workbench to create the database



Inspect new database



Knowledge check 2

Exercise 1

Module completion checklist

Objective	Complete
Discuss what SQL is and what it is used for	✔
Summarize what databases are and give examples	✔
Understanding the entity relationship diagram (ERD)	✔
Establish a database connection & creating a database	✔
Summarize DDL: data types and constraints	
Summarize DDL: create, update & delete table and data	

Data types

- Each column in the table should be of a particular data type
- MySQL has 3 data types

Data Type	Definition
Character	Any text data
Numeric	Any numeric data
Temporal	Any date or time data

Character data

- The character data can be fixed or variable length
  - **char(length)** - fixed length data
  - **varchar(length)** - variable length data
- **Character sets**
  - Represents multi-byte character sets
  - A character set is for languages that have a large number of characters
  - Type `SHOW CHARACTER SET;` to see the detailed description of the character set
- **Text data**
  - If we want to store **data that exceeds 64KB limit** of `varchar`, we need to use the `text` data type instead

Character data

Text type	Maximum number of bytes
Tinytext	255
Text	65,535
Mediumtext	16,777,215
Longtext	4,294,967,295

- **Note:** `1 byte = 1 character`

Numeric data

- Numeric data could be of **integer** type or **floating point** type
- Integer could be **signed** or **unsigned**
- Each type has a specific range to store
- Detailed numeric data standards can be found in MySQL official documentation: ***https://dev.mysql.com/doc/refman/8.0/en/numeric-type-overview.html***

Numeric data

Integer data type

Type	Signed range	Unsigned range
Tinyint	-128 to 127	0 to 255
Smallint	-32768 to 32767	0 to 65535
Mediumint	-8388608 to 8388607	0 to 16777215
Int	-2147483648 to 2147483647	0 to 4294967295
Bigint	-9223372036854775808 to 9223372036854775807	0 to 18446744073709551615

Floating point type

Type	Precision
Float(p,s)	24
Double(p,s)	53

Temporal data

- Temporal data type is used for storing date/time information

Type	Format
Date	YYYY-MM-DD
Datetime	YYYY-MM-DD HH:MI:SS
Timestamp	YYYY-MM-DD HH:MI:SS
Year	YYYY
Time	HHH:MI:SS

Primary Key

Keys in SQL are the **identifiers** we saw in the ER diagram

- **Primary key**
  - Uniquely identifies the rows in a table
  - Cannot be **NULL** and should be **UNIQUE** for every row in a table
  - Can consist of a single column or multiple columns
  - When there is minimal number of columns that uniquely identify the rows in table, it is called a **candidate key**
  - Example: **emp_id** uniquely identifies each employee in the employee table

Foreign Key

Foreign key

- Usually specified to link the tables in the database
- Serves as a primary key for the linked table
- **Child table**: the table that has the foreign key
- **Referenced or parent table**: the table that has the candidate or primary key

Constraints

- Constraints in SQL are the rules for the data in the table
- They exist to maintain accuracy, consistency, and integrity of the database
- If there is a violation in any constraint, the data action is aborted

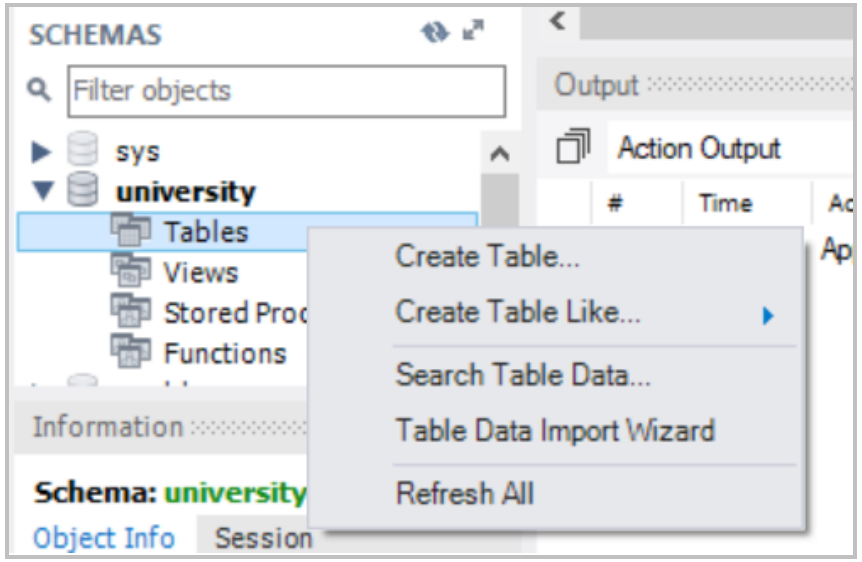
Constraint Name	Definition
NOT NULL	The column cannot store any null values in its rows
UNIQUE	All values in the column must be unique
PRIMARY KEY	Specifies that a given column is the primary key in the table
FOREIGN KEY	Specifies that a given column is the foreign key in the table
CHECK	Helps validating values in the column to meet a specific condition
DEFAULT	Specifies a default value to the column when a user does not provide any other value

Knowledge check 3

Module completion checklist

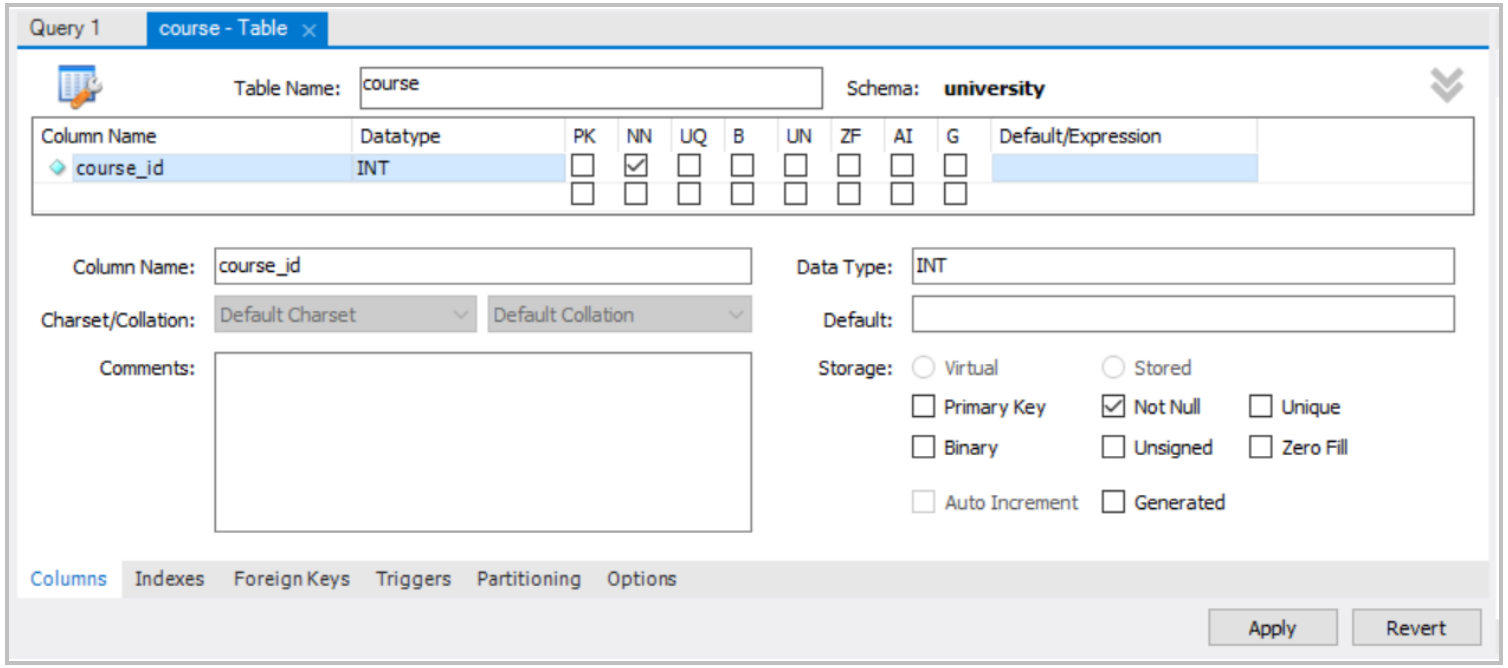
Objective	Complete
Discuss what SQL is and what it is used for	✔
Summarize what databases are and give examples	✔
Understanding the entity relationship diagram (ERD)	✔
Establish a database connection & creating a database	✔
Summarize DDL: data types and constraints	✔
Summarize DDL: create, update & delete table and data	

Add table to database



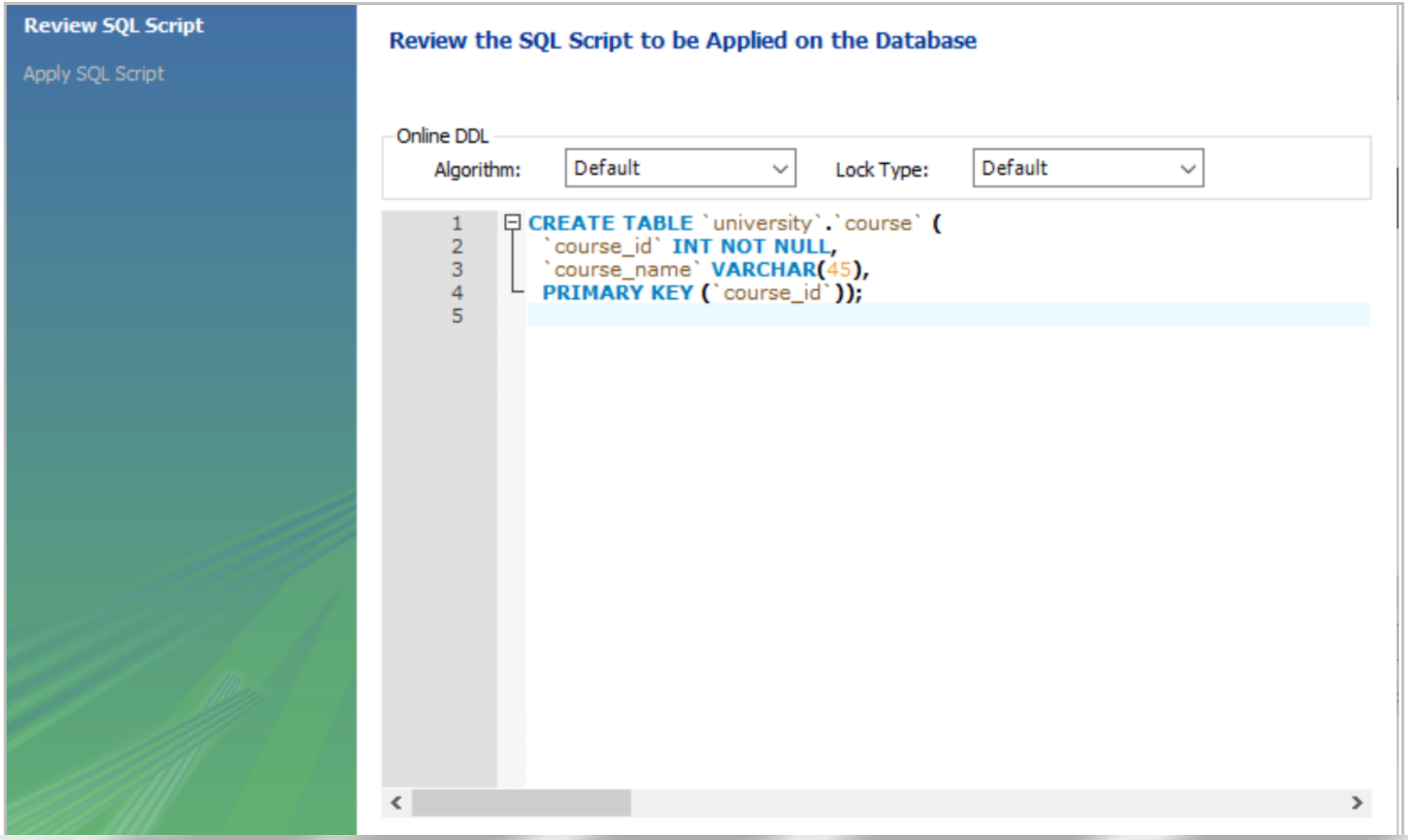
Add table to database

- In the open window, type in `course` for Table Name
- Create `course_id` column of `INT` type and click in `NN` box to set constraint to `NOT NULL` (as in picture below)



- Then, click `PK` for primary key
- After that, create another column `course_name` of type `varchar(45)` and click the **Apply** button

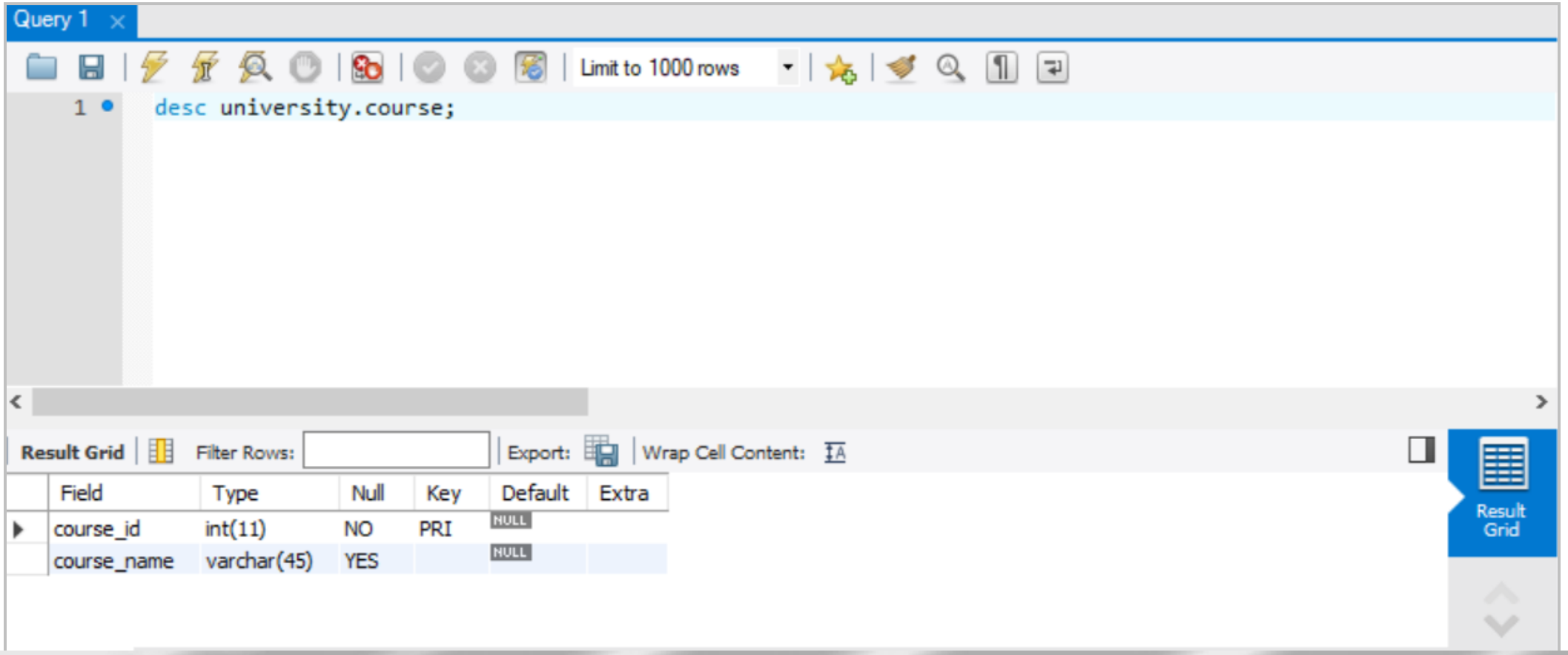
Add table to database



DESC

- To inspect the structure of the table, use the **DESC** clause

```
DESC university.course;
```



CREATE TABLE

- We can type the code to create a table in the database using the **CREATE TABLE** clause

Note: The foreign key is defined *only when there exists a foreign key in a table*, but primary key is defined for every table created

```
USE university;                                          -- define database to avoid referencing everytime

CREATE TABLE faculty                                     -- table name
(faculty_id INT(10),                                     -- column 1 - faculty id
faculty_name VARCHAR(20),                                -- column 2 - faculty_name
course_id INT(11),                                       -- column 3 - course_id
CONSTRAINT pk_faculty PRIMARY KEY(faculty_id),           -- define primary key
CONSTRAINT fk_faculty_course_id FOREIGN KEY(course_id)   -- define foreign key
REFERENCES course(course_id)                             -- referencing table
);

DESC faculty;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
faculty_id	int(10)	NO	PRI	NULL	
faculty_name	varchar(20)	YES		NULL	
course_id	int(11)	YES	MUL	NULL	

INSERT

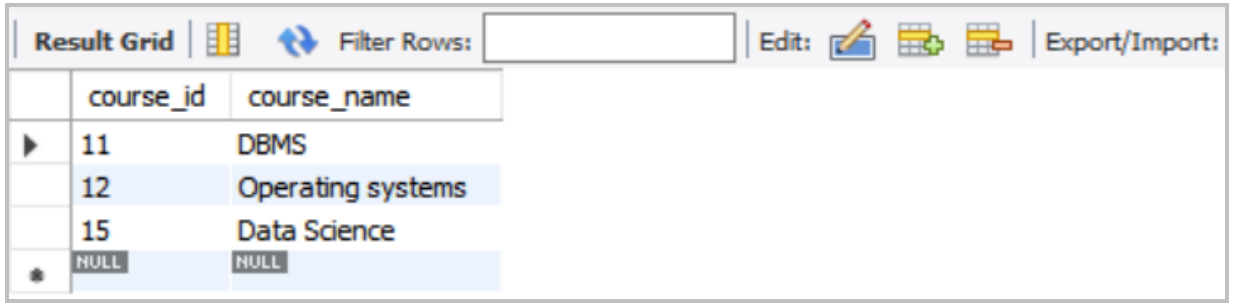
- To insert values into the table, use the **INSERT INTO** clause

```
INSERT INTO course        -- table name
(course_id, course_name) -- column/attributes name
VALUES (011, "DBMS");     -- values to the attributes
```

SELECT

- To view all the data available in the table, use the **SELECT** clause

```
SELECT * from course; -- * to view all the data in table
```



course_id	course_name
11	DBMS
12	Operating systems
15	Data Science
NULL	NULL

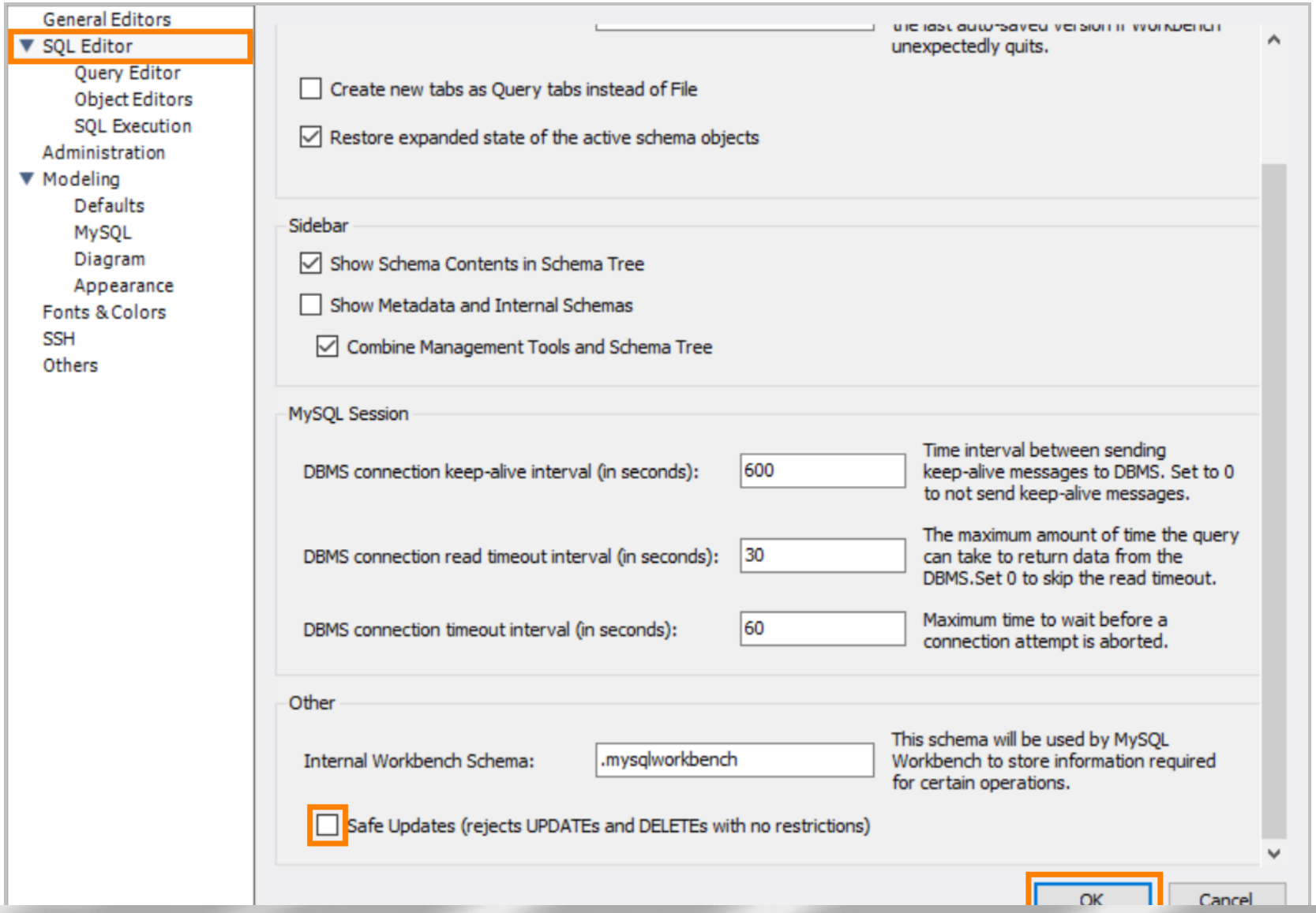
UPDATE

- We can update the table structure and data inserted using the **UPDATE** clause
- Condition for updating can be given using the **WHERE** clause
- Single row can be updated by `'='` operator
- Multiple rows can be updated by `'<', '>', '<=', '>=', '!='` operators
- New values are updated using the **SET** clause

Note: Before we run the update queries in MySQL, let us disable the `safe update` option.

- Go to `preferences` -> uncheck the `safe update` option and reconnect your MySQL
- If you are:
  - `windows/linux` user `preferences` can be found in `edit` tab
  - `mac` user `preferences` can be found in `MySQLWorkbench` tab on the top

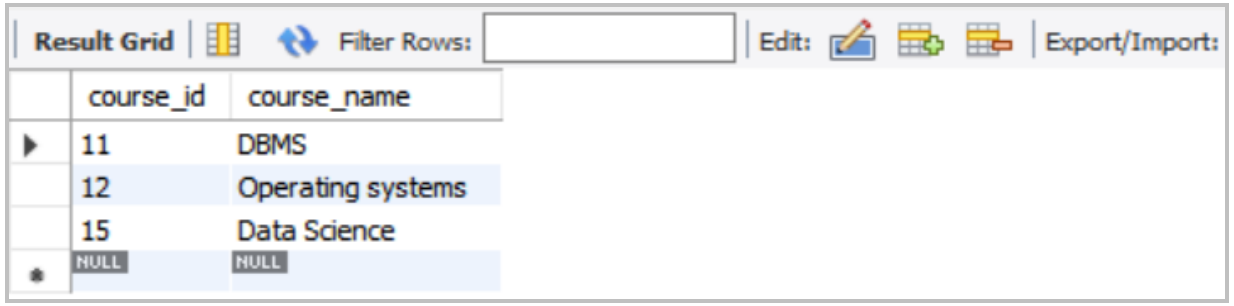
UPDATE


General Editors
SQL Editor
Query Editor
Object Editors
SQL Execution
Administration
Modeling
Defaults
MySQL
Diagram
Appearance
Fonts & Colors
SSH
Others
unexpectedly quits.
Create new tabs as Query tabs instead of File
Restore expanded state of the active schema objects
Sidebar
Show Schema Contents in Schema Tree
Show Metadata and Internal Schemas
Combine Management Tools and Schema Tree
MySQL Session
DBMS connection keep-alive interval (in seconds):
600
Time interval between sending keep-alive messages to DBMS. Set to 0 to not send keep-alive messages.
DBMS connection read timeout interval (in seconds):
30
The maximum amount of time the query can take to return data from the DBMS.Set 0 to skip the read timeout.
DBMS connection timeout interval (in seconds):
60
Maximum time to wait before a connection attempt is aborted.
Other
Internal Workbench Schema:
.mysqlworkbench
This schema will be used by MySQL Workbench to store information required for certain operations.
Safe Updates (rejects UPDATEs and DELETEs with no restrictions)
OK
Cancel

UPDATE

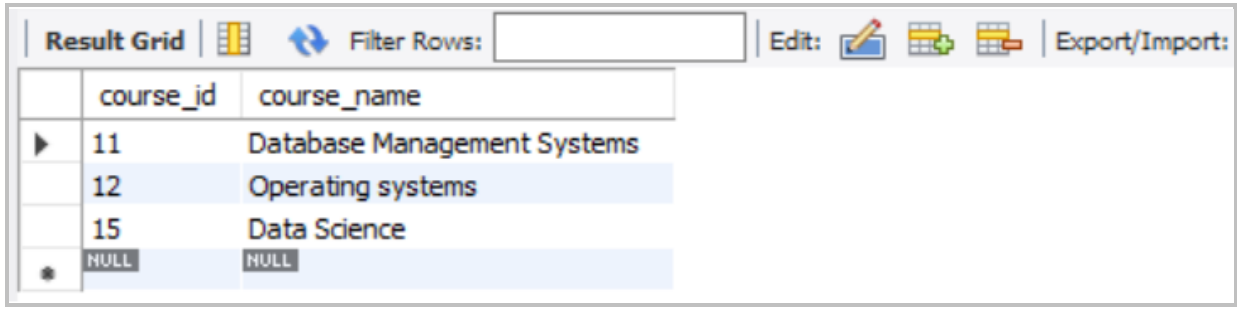
```
UPDATE course                                        -- update the table
SET course_name = "Database Management Systems"      -- set new value
WHERE course_id = 011;                               -- condition to set the data
```

Before updating



course_id	course_name
11	DBMS
12	Operating systems
15	Data Science
NULL	NULL

After updating



course_id	course_name
11	Database Management Systems
12	Operating systems
15	Data Science
NULL	NULL

ALTER TABLE

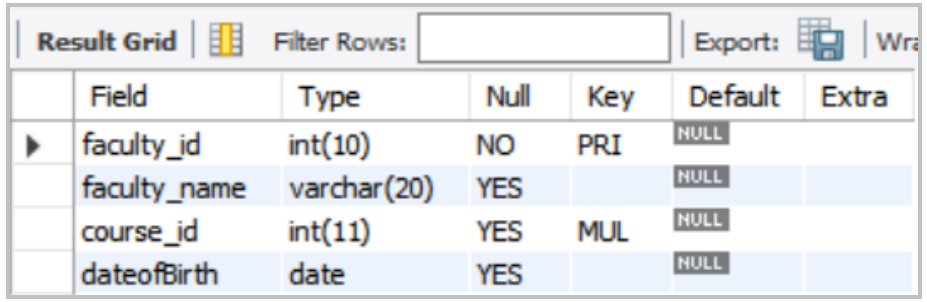
- We can change the table structure using the **ALTER TABLE** clause
- **ALTER TABLE** can be used to
  - **Add** a new column
  - **Modify** the existing column
  - **Drop** the existing column

ALTER TABLE - ADD

- To add a new column, use the **ADD** clause and define the column name & type

```
ALTER TABLE faculty   -- table name
ADD dateofBirth DATE; -- add new column 'dateofBirth'
```

```
DESC faculty;
```



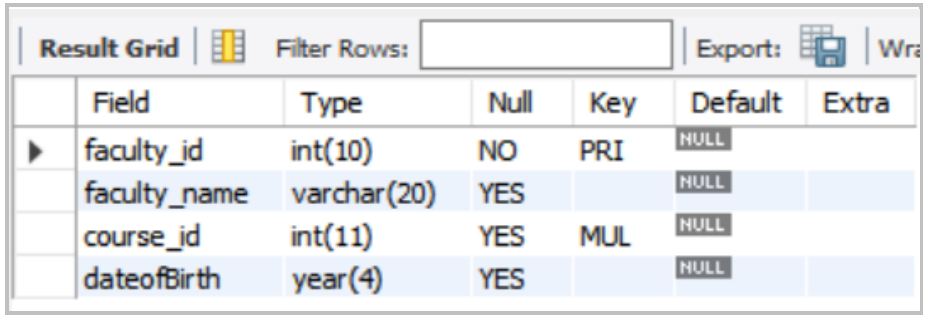
Field	Type	Null	Key	Default	Extra
faculty_id	int(10)	NO	PRI	NULL	
faculty_name	varchar(20)	YES		NULL	
course_id	int(11)	YES	MUL	NULL	
dateofBirth	date	YES		NULL	

ALTER TABLE - MODIFY & DROP

- To modify the column structure, use the **MODIFY** clause

```
ALTER TABLE faculty       -- table name
MODIFY dateofBirth YEAR; -- alter 'dateofBirth'
                          --  date to year type
```

```
DESC faculty;
```



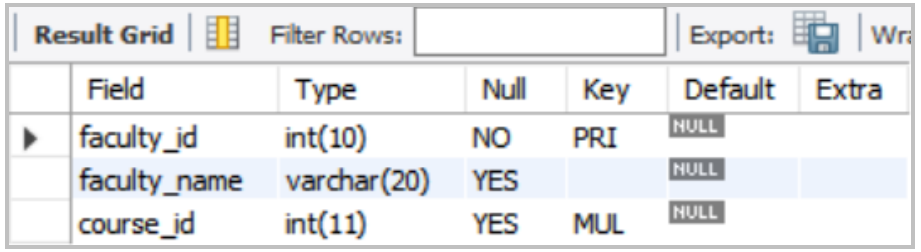
Result Grid | Filter Rows: | Export: | Wra

Field	Type	Null	Key	Default	Extra
faculty_id	int(10)	NO	PRI	NULL	
faculty_name	varchar(20)	YES		NULL	
course_id	int(11)	YES	MUL	NULL	
dateofBirth	year(4)	YES		NULL	

- To delete a column, use the **DROP** clause

```
ALTER TABLE faculty       -- table name
DROP COLUMN dateofBirth; -- delete column
                          --   'dateofBirth'
```

```
DESC faculty;
```



Result Grid | Filter Rows: | Export: | Wra

Field	Type	Null	Key	Default	Extra
faculty_id	int(10)	NO	PRI	NULL	
faculty_name	varchar(20)	YES		NULL	
course_id	int(11)	YES	MUL	NULL	

Practice UPDATE and INSERT

```
-- Update course table.
UPDATE course
SET course_name = "DBMS"            -- set new value
WHERE course_id = 011;              -- condition to set the data

-- Insert a new row into course table.
INSERT INTO course                  -- table name
(course_id, course_name)            -- column/attributes name
VALUES (012, "Operating systems"); -- values to the attributes

-- Insert a new row into course table.
INSERT INTO course                  -- table name
(course_id, course_name)            -- column/attributes name
VALUES (015, "Data Science");       -- values to the attributes
```

DELETE - data

- We can delete the data from the table by using the **DELETE FROM** clause
- Use the **WHERE** clause to set the condition to delete the data
- Single row can be deleted by `'='` operator
- Multiple rows can be deleted by `'<', '>', '<=', '>=', '!='` operators

- **Before deleting**

```
SELECT * from course;
```

Result Grid | Filter Rows:

course_id	course_name
11	DBMS
12	Operating systems
15	Data Science
NULL	NULL

- **After deleting**

```
DELETE FROM course WHERE course_id = 12;
```

Result Grid | Filter Rows:

course_id	course_name
11	DBMS
15	Data Science
NULL	NULL

DELETE - table & column

- To delete the entire data from the table

```
DELETE FROM course;
```

- To remove the table itself from the database, use the **DROP TABLE**

```
DROP TABLE faculty;  -- child table having course_id as foreign key
DROP TABLE course;   -- parent table
```

- **Note: always delete the child table with the foreign key first**
- Table `faculty` has a foreign key from `course` table, so delete `faculty` table first
- `course` table is the referenced table - if we delete course table first, it will throw an error
- Delete faculty table first and then course table

✖	43	01:12:16	drop table course	Error Code: 3730. Cannot drop table 'course' referenced by a foreign key constraint '...	0.016 sec
✔	44	01:12:30	drop table faculty	0 row(s) affected	0.031 sec
✔	45	01:12:37	drop table course	0 row(s) affected	0.031 sec

When good SQL statements go bad

- Non unique **primary key**
- Non existent **foreign key**
- Column value violation
- Invalid **date** conversion

Knowledge check 4

Exercise 2

Module completion checklist

Objective	Complete
Discuss what SQL is and what it is used for	✔
Summarize what databases are and give examples	✔
Understanding the entity relationship diagram (ERD)	✔
Establish a database connection & creating a database	✔
Summarize DDL: data types and constraints	✔
Summarize DDL: create, update & delete table and data	✔

Workshop!

- Workshops are to be completed in the afternoon either with a dataset for a capstone project or with another dataset of your choosing
- Make sure to annotate and comment your code so that it is easy for others to understand what you are doing
- This is an exploratory exercise to get you comfortable with the content we discussed today
- Today, you will:
  - Create a database of your choice
  - Create tables along with primary key and foreign key relation
  - Insert records into the database
  - Practice working with alter and update of the table and data

This completes our module

Congratulations!