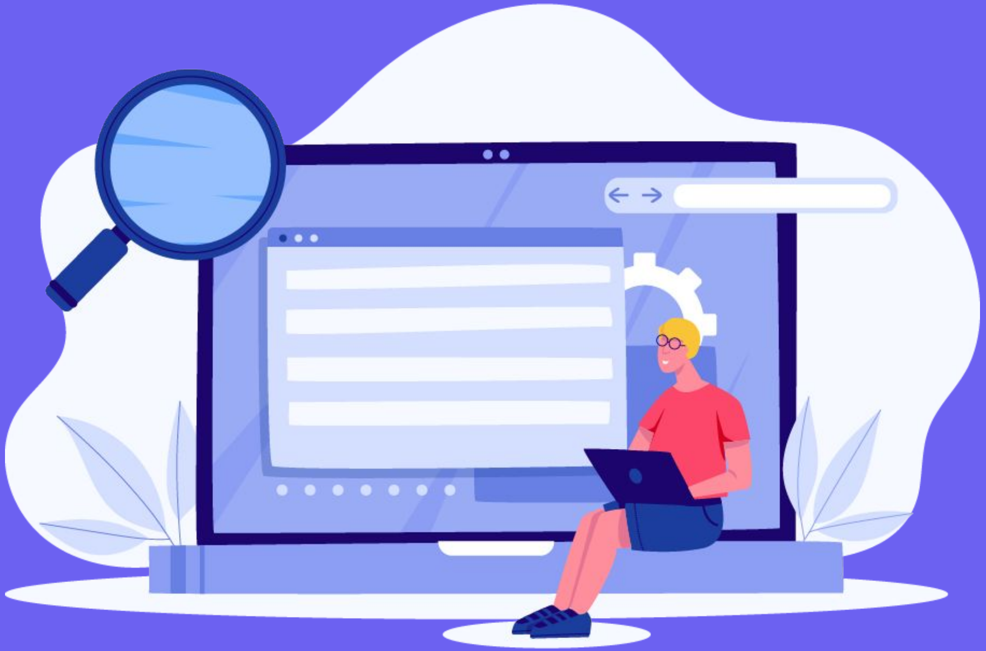
Creating Databases using DDL and DML commands



Relevel
by Unacademy

What are DDL commands?

The DDL commands in SQL are used to create database schema and to define the type and structure of the data that will be stored in a database.

SQL DDL commands are further divided into the following major categories:

- CREATE
- ALTER
- DROP
- TRUNCATE

Create Commands

The CREATE query is used to create a:

- Database
- Objects such as tables, views, stored procedures, etc.

Creating a Database

Syntax for creating a database

CREATE DATABASE “database name”

The following example demonstrates how the CREATE query can be used to create a database:

```
CREATE DATABASE LibraryDB
```

The script above creates a database named “LibraryDB”.

Creating a Table

The create statement could also be used to create a table(an object in database):

Syntax for creating a table

```
CREATE TABLE “table name” (
Column1 datatype,
Column2 datatype,
Column3 datatype,
…….
),
```

Creating a Table

A copy/manipulation on an existing table can also be used to create a table.

Syntax for creating a table:

```
CREATE TABLE new_table_name AS
   SELECT column1, column2,...
   FROM existing_table_name
   WHERE ....;
```

Creating a Table – An Example

Question - Write a sql query to generate a new table named as Books having attributes(or column_names) as ID, Name and Price

```
CREATE TABLE Books
(
Id INT PRIMARY KEY,
Name VARCHAR (50) NOT NULL,
Price INT
)
```

The script above creates a " Books " table in the previously created "LibraryDB" database.

There are three columns in the "Books" table: Id, Name, and Price.

The primary key column is Id, and it cannot be NULL. A PRIMARY KEY constraint requires that a column contain unique values. We must also specify the values for the Name column, which cannot be NULL. Finally, NULL values are permitted in the Price column.

SQL Views

In SQL Server, a VIEW is similar to a virtual table that contains data from one or more tables. It has no data and does not exist in the database physically.

Like a SQL table, the view name should be unique in a database. It includes a set of predefined SQL queries for retrieving data from the database.

It can also contain database tables from single or multiple databases.

A VIEW does not require database storage because it does not exist physically. We can also control user security for accessing data from database tables in a VIEW. We can allow users to retrieve data from the VIEW, and the user does not need permission to retrieve data from each table or column.

Creating View

Syntax for creating a view

```
CREATE VIEW “vName” AS
Select column1, Column2...Column N From tables
Where conditions;
```

The view “vName” can later be used to query data.

Creating View – An example

We will use the table “Books” created in the previous example for view creation.

```
CREATE VIEW  vbooks AS
SELECT
        id,
        name
FROM
        Books
```

Alter Command

The ALTER command in SQL DDL is used to modify the structure of an already existing table.

The modification in the structure could be:

- Adding a new column
- Modifying a column
- Deleting a column

Alter Command

Question- Write a query to add new column ISBN in existing table

Alter Command – Adding a column

Syntax

```
ALTER TABLE table_name
ADD column_name datatype;
```

Example

```
ALTER TABLE Books
ADD ISBN INT NOT NULL;
```

The above example added a new column ‘ISBN’ to the existing Books table.

Alter Command –Modifying an existing column

Syntax

```
ALTER TABLE table_name
ALTER COLUMN column_name datatype;
```

Example

```
ALTER TABLE Books
ALTER COLUMN ISBN VARCHAR(50);
```

In the above example, we change the data type from integer to varchar.

Alter Command –Deleting a column

Question-Write a query to delete a column from table.

Syntax

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

Example

```
ALTER TABLE Books
DROP COLUMN ISBN;
```

In the above example, we deleted the column ‘ISBN’.

DROP Command

The DROP command is a type of SQL DDL command that is used to:

- Delete an existing database
- An object within a database

DROP Command – Dropping a Database

Syntax

DROP DATABASE ‘database name’.

Example

DROP DATABASE LibraryDB

In the above example, we deleted the database LibraryDB which we created earlier.

DROP Command – Dropping a Table

Syntax

DROP Table ‘table name’.

Example

DROP Table Book

In the above example, we deleted the table ‘Book’ which we created earlier.

Truncate Command

The TRUNCATE command in SQL DDL is used to remove all the records from a table.

Syntax

```
TRUNCATE TABLE ‘table name’
```

Example

```
TRUNCATE TABLE Books
```

In the above example, we deleted all the records from table ‘Books’.

DELETE VS DROP VS TRUNCATE

- Delete is a DML command. It is used to delete some or all records from a table.
- **DELETE FROM “table” WHERE condition**
- TRUNCATE is a DDL command. It is used to delete all the records from a table. But it retains the table schema.
- DROP is a DDL command. It drops all the records as well as the schema of the table.

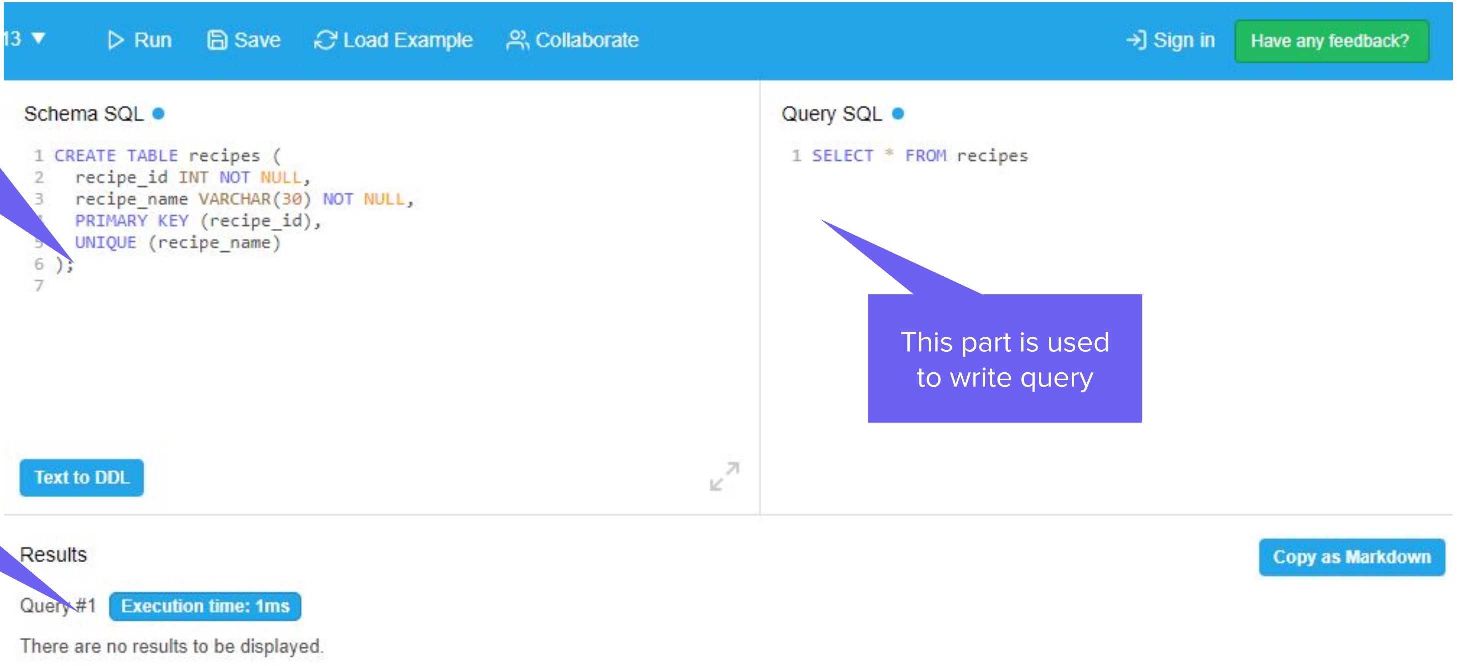
DDL commands – an End to End case study

We are going to do a real-time example where we will:

- create a table,insert data,
- add a column,
- modify a column,
- delete some rows,
- truncate the data, and
- eventually drop the table

We will use db-fiddle.com to understand this case study.

DB-Fiddle - Overview



Schema SQL

```
CREATE TABLE recipes (
  recipe_id INT NOT NULL,
  recipe_name VARCHAR(30) NOT NULL,
  PRIMARY KEY (recipe_id),
  UNIQUE (recipe_name)
);
```

Query SQL

```
SELECT * FROM recipes
```

Results

Query #1 Execution time: 1ms

There are no results to be displayed.

This part of the Db-fiddle is use to define the schema

This part is used to write query

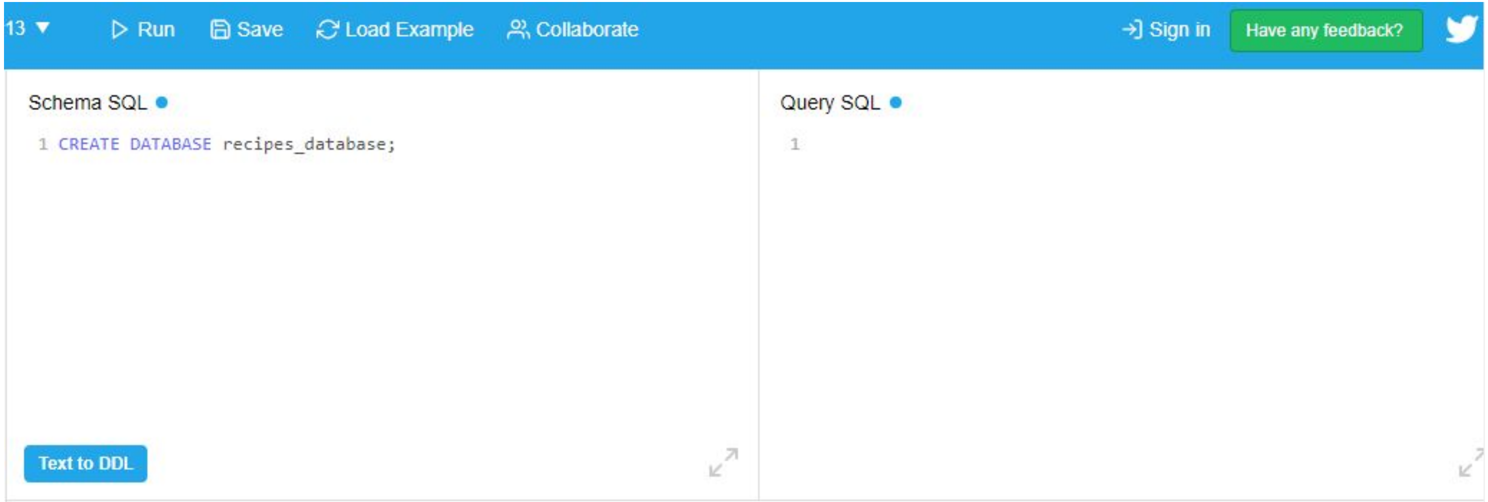
This shows the result

End to End case study – Creating Database

In this example, we will first create a database called recipes_database:

Syntax:

CREATE DATABASE recipes_database;



```
CREATE DATABASE recipes_database;
```

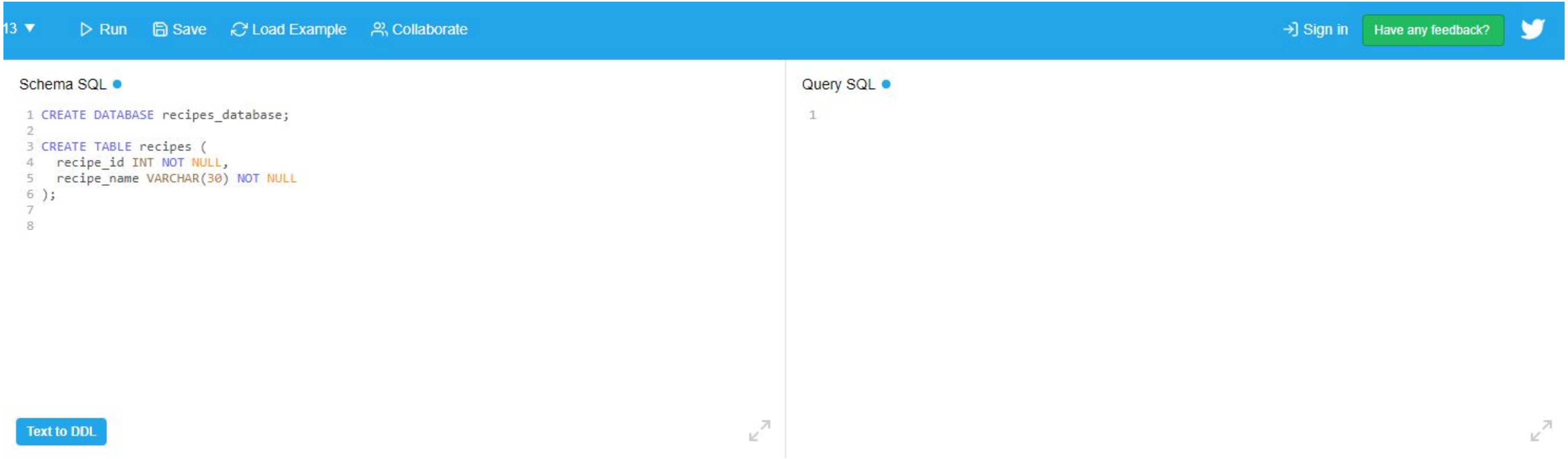
End to End case study – Creating Table

In this example, we will use database ‘recipes_database’ and create a table called ‘recipes’.

Syntax:

```
CREATE TABLE recipes (
  recipe_id INT NOT NULL,
  recipe_name VARCHAR(30) NOT NULL,
  PRIMARY KEY (recipe_id),
  UNIQUE (recipe_name)
);
```

End to End case study – Creating Table



```
CREATE DATABASE recipes_database;

CREATE TABLE recipes (
  recipe_id INT NOT NULL,
  recipe_name VARCHAR(30) NOT NULL
);


```

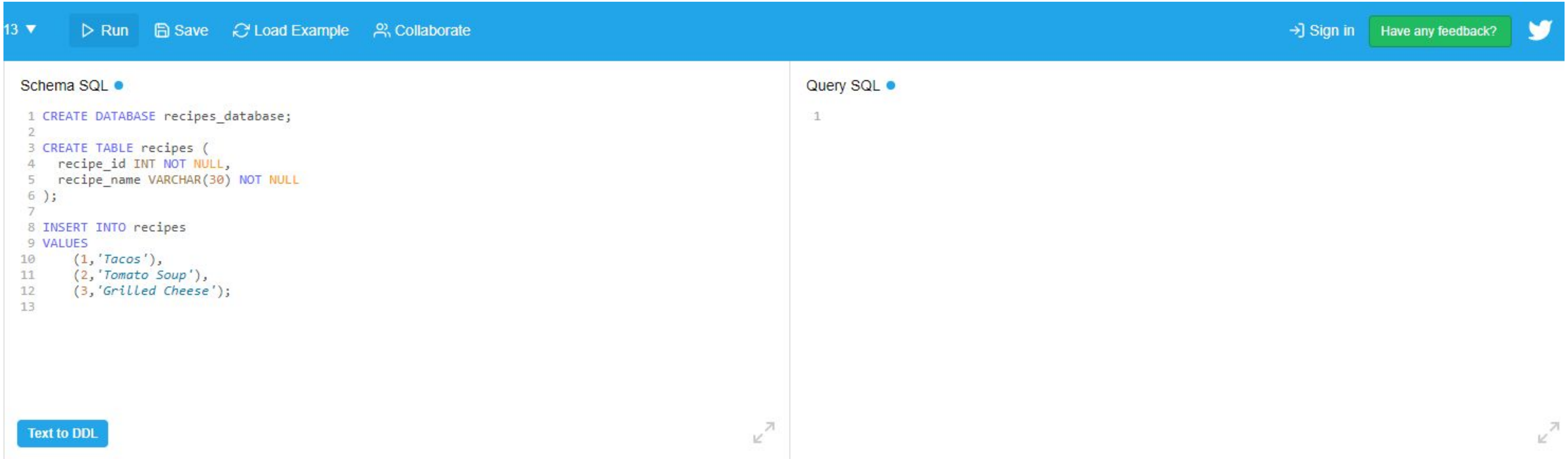
End to End case study – Inserting Data

In this example, we will use database recipes_database and create a table called ‘recipes’.

Syntax:

```
INSERT INTO recipes
    (recipe_id, recipe_name)
VALUES
    (1,’Tacos’),
    (2,’Tomato Soup’),
    (3,’Grilled Cheese’);
```

End to End case study – Inserting Data

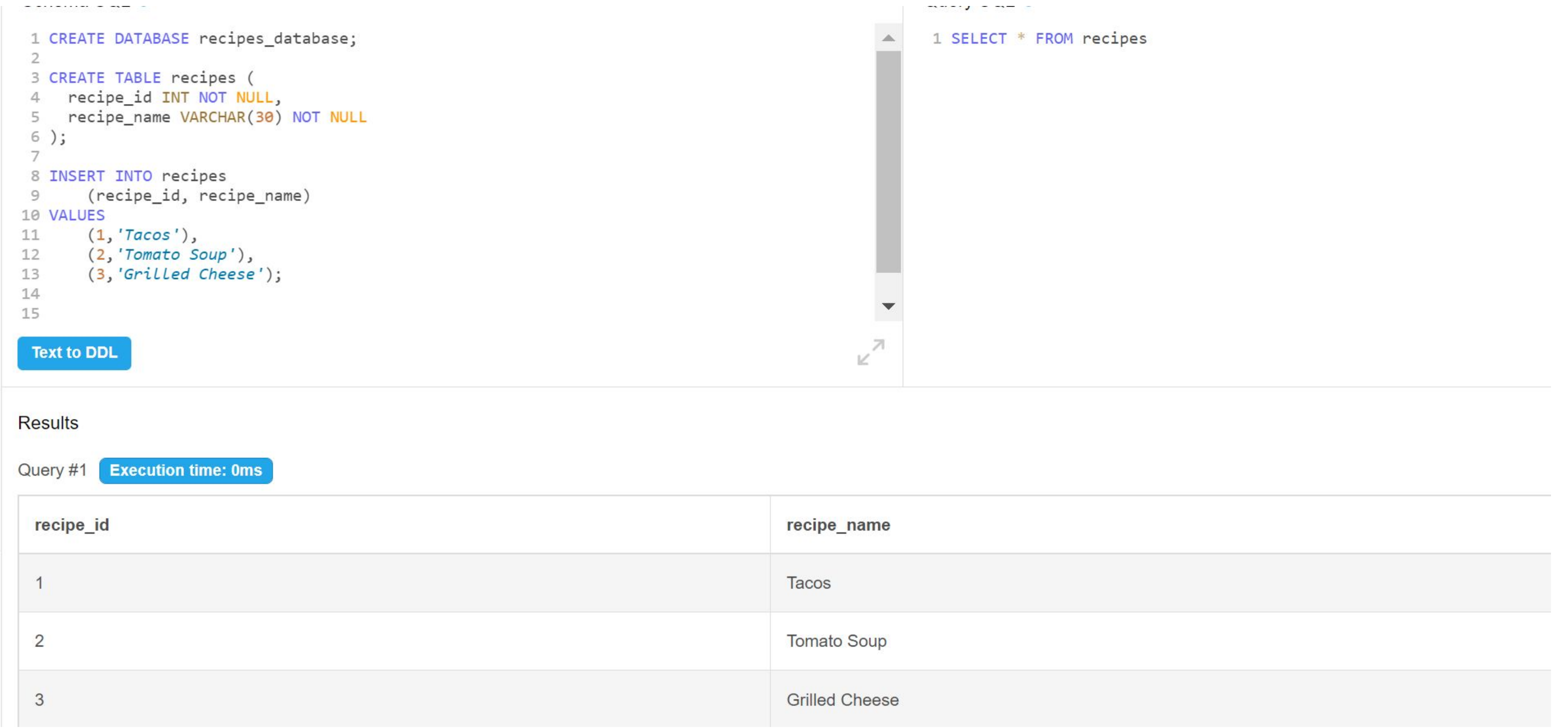


```sql
CREATE DATABASE recipes_database;

CREATE TABLE recipes (
  recipe_id INT NOT NULL,
  recipe_name VARCHAR(30) NOT NULL
);

INSERT INTO recipes
VALUES
    (1,'Tacos'),
    (2,'Tomato Soup'),
    (3,'Grilled Cheese');
```

End to End case study – Running the query



```sql
CREATE DATABASE recipes_database;

CREATE TABLE recipes (
  recipe_id INT NOT NULL,
  recipe_name VARCHAR(30) NOT NULL
);

INSERT INTO recipes
    (recipe_id, recipe_name)
VALUES
    (1,'Tacos'),
    (2,'Tomato Soup'),
    (3,'Grilled Cheese');
```

```sql
SELECT * FROM recipes
```

Results

Query #1 Execution time: 0ms

recipe_id	recipe_name
1	Tacos
2	Tomato Soup
3	Grilled Cheese

End to End case study – Altering Table

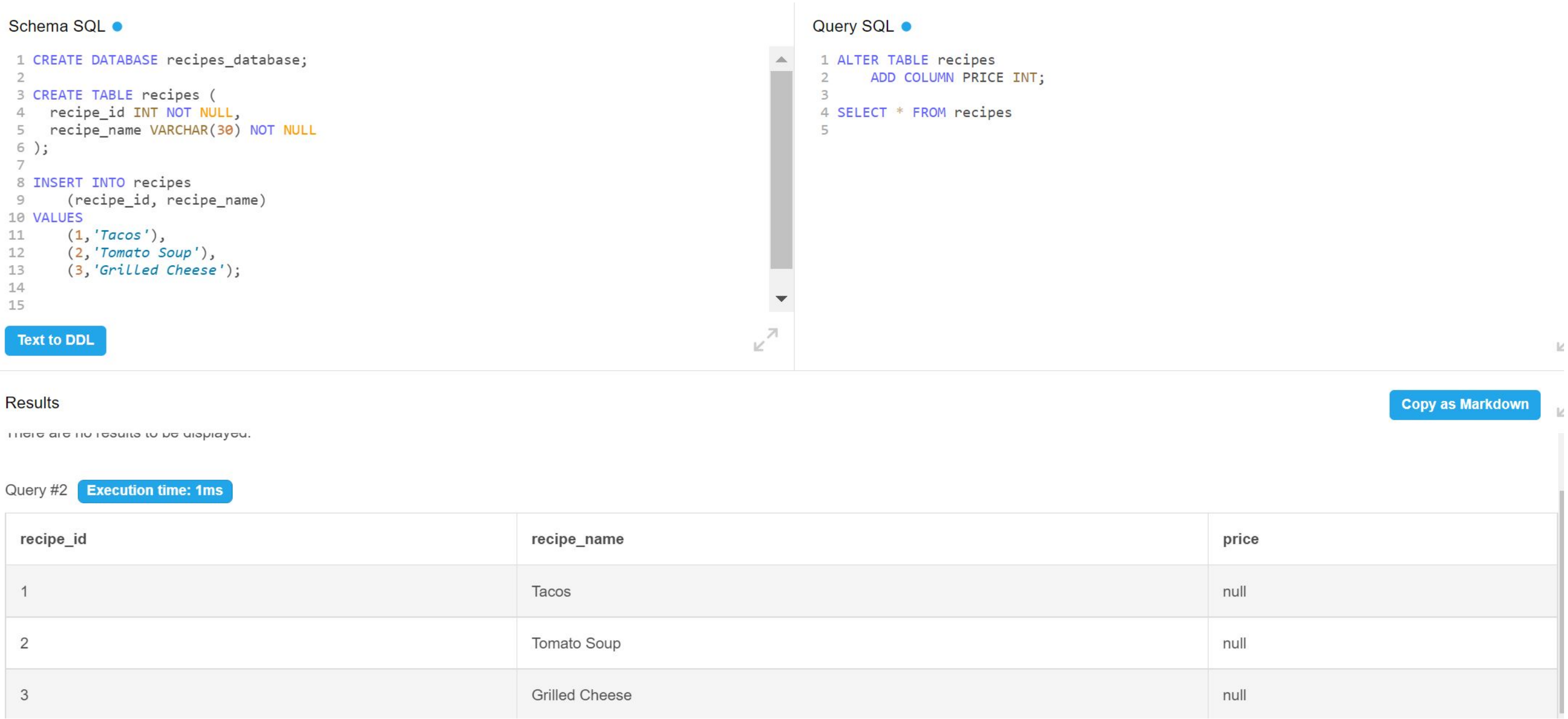
In this step, we will add a new column price.

Syntax:

```
ALTER TABLE recipes
        ADD COLUMN PRICE INT;


SELECT * FROM recipes
```

End to End case study – Altering Table



Schema SQL

```
CREATE DATABASE recipes_database;

CREATE TABLE recipes (
  recipe_id INT NOT NULL,
  recipe_name VARCHAR(30) NOT NULL
);

INSERT INTO recipes
    (recipe_id, recipe_name)
VALUES
    (1,'Tacos'),
    (2,'Tomato Soup'),
    (3,'Grilled Cheese');
```

Query SQL

```
ALTER TABLE recipes
    ADD COLUMN PRICE INT;

SELECT * FROM recipes
```

Results

There are no results to be displayed.

Query #2 Execution time: 1ms

recipe_id	recipe_name	price
1	Tacos	null
2	Tomato Soup	null
3	Grilled Cheese	null

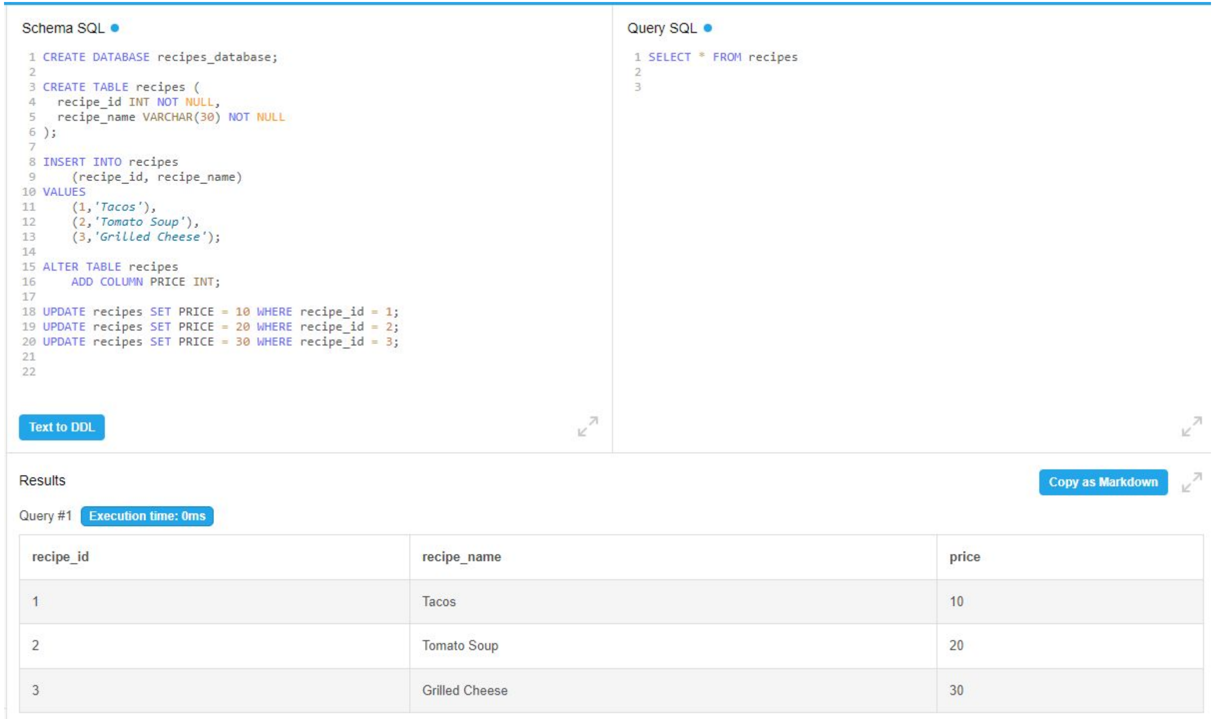
End to End case study – Adding data in new column

In this step, we will add a new column price.

Syntax:

```
UPDATE recipes SET PRICE = 10 WHERE recipe_id = 1;
UPDATE recipes SET PRICE = 20 WHERE recipe_id = 2;
UPDATE recipes SET PRICE = 30 WHERE recipe_id = 3;
```

End to End case study – Adding data in new column



```
CREATE DATABASE recipes_database;

CREATE TABLE recipes (
  recipe_id INT NOT NULL,
  recipe_name VARCHAR(30) NOT NULL
);

INSERT INTO recipes
    (recipe_id, recipe_name)
VALUES
    (1,'Tacos'),
    (2,'Tomato Soup'),
    (3,'Grilled Cheese');

ALTER TABLE recipes
    ADD COLUMN PRICE INT;

UPDATE recipes SET PRICE = 10 WHERE recipe_id = 1;
UPDATE recipes SET PRICE = 20 WHERE recipe_id = 2;
UPDATE recipes SET PRICE = 30 WHERE recipe_id = 3;
```

```
SELECT * FROM recipes
```

recipe_id	recipe_name	price
1	Tacos	10
2	Tomato Soup	20
3	Grilled Cheese	30

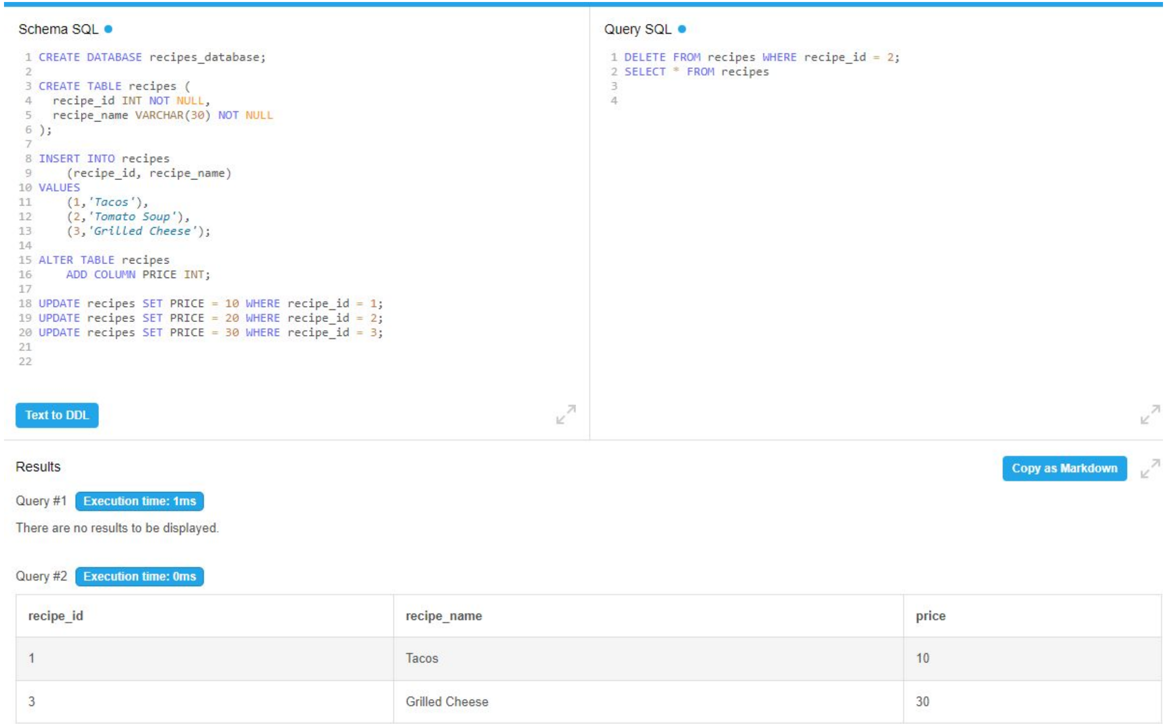
End to End case study – Deleting a row

In this step, we will delete one row.

Syntax:

```
DELETE FROM recipes WHERE recipe_id = 2;
```

End to End case study – Deleting a row



```
CREATE DATABASE recipes_database;

CREATE TABLE recipes (
  recipe_id INT NOT NULL,
  recipe_name VARCHAR(30) NOT NULL
);

INSERT INTO recipes
    (recipe_id, recipe_name)
VALUES
    (1,'Tacos'),
    (2,'Tomato Soup'),
    (3,'Grilled Cheese');

ALTER TABLE recipes
    ADD COLUMN PRICE INT;

UPDATE recipes SET PRICE = 10 WHERE recipe_id = 1;
UPDATE recipes SET PRICE = 20 WHERE recipe_id = 2;
UPDATE recipes SET PRICE = 30 WHERE recipe_id = 3;
```

```
DELETE FROM recipes WHERE recipe_id = 2;
SELECT * FROM recipes
```

Results

Query #1 Execution time: 1ms

There are no results to be displayed.

Query #2 Execution time: 0ms

recipe_id	recipe_name	price
1	Tacos	10
3	Grilled Cheese	30

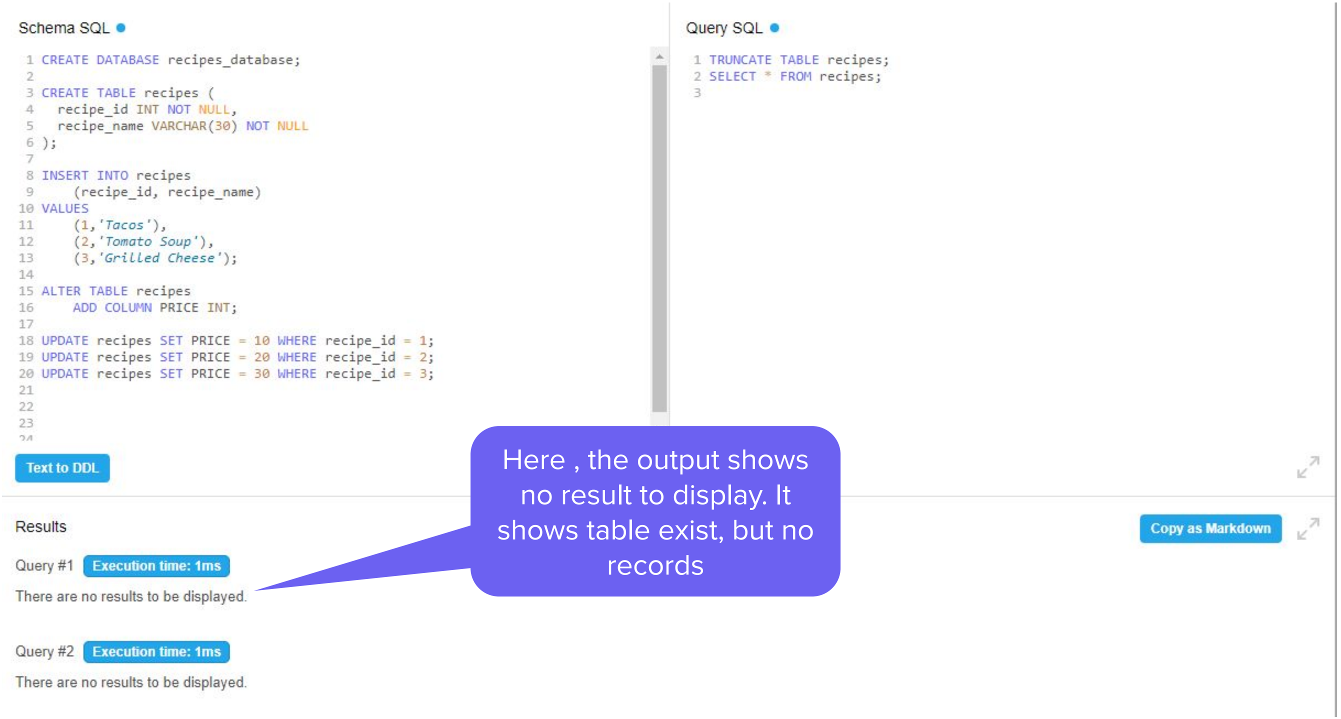
End to End case study – Truncating the TABLE

In this step, we will drop all the rows. However, the table schema will be retained.

Syntax:

```
TRUNCATE TABLE recipes;
```

End to End case study – Truncating the TABLE



Schema SQL

```sql
CREATE DATABASE recipes_database;

CREATE TABLE recipes (
  recipe_id INT NOT NULL,
  recipe_name VARCHAR(30) NOT NULL
);

INSERT INTO recipes
    (recipe_id, recipe_name)
VALUES
    (1,'Tacos'),
    (2,'Tomato Soup'),
    (3,'Grilled Cheese');

ALTER TABLE recipes
    ADD COLUMN PRICE INT;

UPDATE recipes SET PRICE = 10 WHERE recipe_id = 1;
UPDATE recipes SET PRICE = 20 WHERE recipe_id = 2;
UPDATE recipes SET PRICE = 30 WHERE recipe_id = 3;
```

Query SQL

```sql
TRUNCATE TABLE recipes;
SELECT * FROM recipes;
```

Results

Query #1 Execution time: 1ms

There are no results to be displayed.

Query #2 Execution time: 1ms

There are no results to be displayed.

Here , the output shows no result to display. It shows table exist, but no records

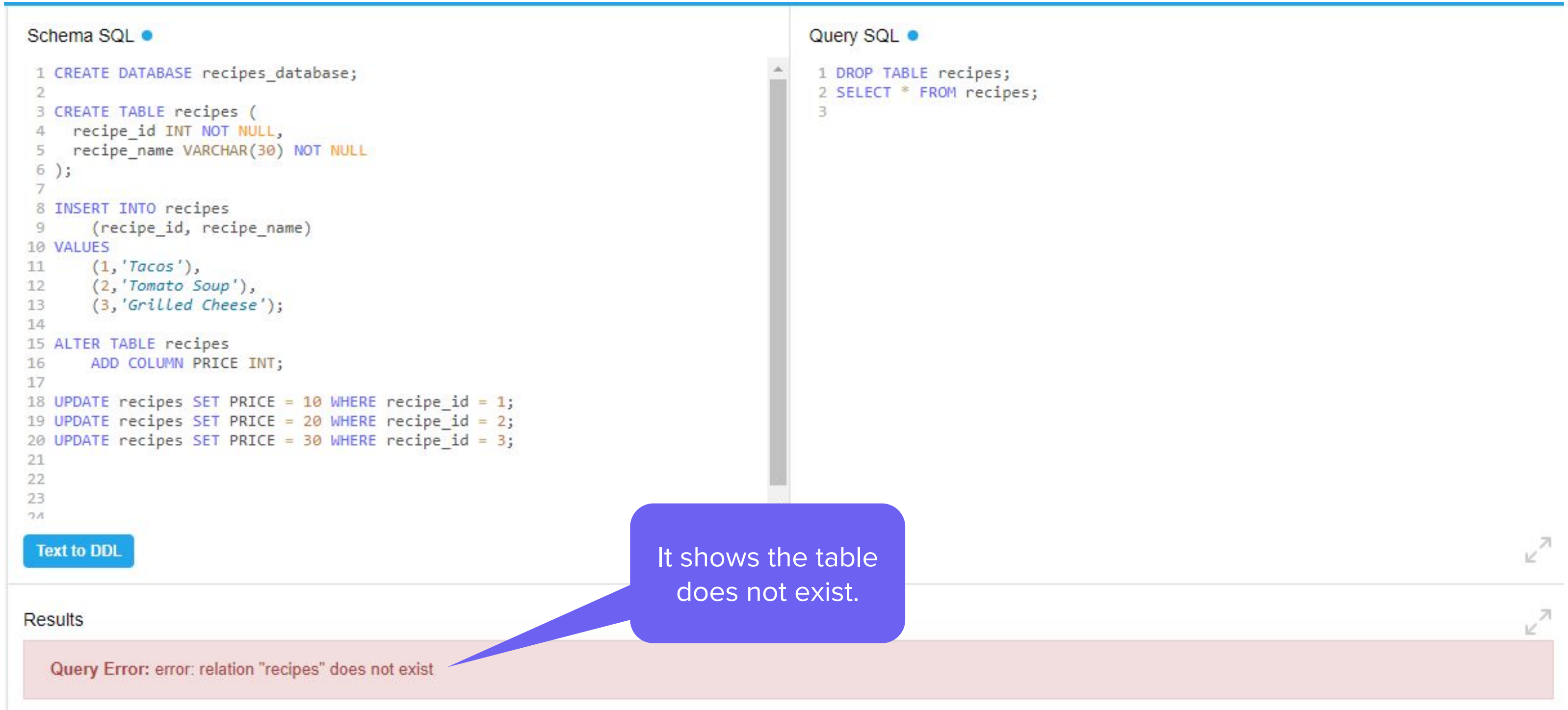
End to End case study – Dropping the TABLE

In this step, we will drop the table.

Syntax:

```
DROPTABLE recipes;
```

End to End case study – Dropping the TABLE



Schema SQL

```
CREATE DATABASE recipes_database;

CREATE TABLE recipes (
  recipe_id INT NOT NULL,
  recipe_name VARCHAR(30) NOT NULL
);

INSERT INTO recipes
    (recipe_id, recipe_name)
VALUES
    (1,'Tacos'),
    (2,'Tomato Soup'),
    (3,'Grilled Cheese');

ALTER TABLE recipes
    ADD COLUMN PRICE INT;

UPDATE recipes SET PRICE = 10 WHERE recipe_id = 1;
UPDATE recipes SET PRICE = 20 WHERE recipe_id = 2;
UPDATE recipes SET PRICE = 30 WHERE recipe_id = 3;
```

Query SQL

```
DROP TABLE recipes;
SELECT * FROM recipes;
```

Results

```
Query Error: error: relation "recipes" does not exist
```

It shows the table does not exist.

Practice Question

Instructions for practice questions

- We will use db-fiddle.com
- Use PostgreSQLv13 as the database type

Practice Question - 1

Create a table – “Course”. Insert two columns Course_ID(int), COURSE_NAME(VARCHAR)

Solution - 1

```
CREATE TABLE COURSE ( Course_ID Int, Course_Name Varchar(10) )
```

Practice Question - 2

Insert 4 rows into the table:

- 1, SQL
- 2, ‘Python’
- 3, ‘JAVA’
- 4, ‘C’

Solution - 2

```
INSERT INTO COURSE
VALUES
        (1,'SQL'),
        (2,’Python'),
        (3,'JAVA'),
        (4,'C');
```

Practice Question - 3

Add another column – difficulty_level.

Solution - 3

```
ALTER TABLE COURSE
        ADD COLUMN difficulty_level VARCHAR;
```

Practice Question - 4

Insert the following value in difficulty_level columns for each language:

- SQL - Easy
- Python - Medium
- JAVA - Hard
- C – Very Hard

Solution - 4

```
UPDATE COURSE SET difficulty_level = 'Easy' WHERE COURSE_NAME =  'SQL';
UPDATE COURSE SET difficulty_level = 'Medium' WHERE COURSE_NAME =  'Python';
UPDATE COURSE SET difficulty_level = 'Hard' WHERE COURSE_NAME =  'JAVA';
UPDATE COURSE SET difficulty_level = 'Very Hard' WHERE COURSE_NAME =  'C’;

SELECT * FROM COURSE
```

Practice Question - 5

Delete the row from course_name = ‘Python’.

Solution - 5

```
DELETE FROM COURSE WHERE course_name  = 'Python';


SELECT * FROM COURSE
```

Practice Question - 6

DROP the course Table.

Solution - 6

```
DROP TABLE COURSE;
```

In the next class, we will study

- Data Manipulation in SQL