Database Management Systems Training

June 12 – 16, 2023

One Central Hotel
Cebu City, Philippines

SQL Syntax

SQL Syntax

- SQL syntax refers to the rules and conventions that dictate the structure and format of SQL statements.
- It encompasses keywords, functions, and operators used for querying and managing data in relational databases.
- It is essential for constructing valid and accurate statements that can be executed by the database management system.
- Following the correct syntax ensures that statements are well-formed and conform to the SQL standard or the specific database system being used.

Case Sensitivity

- SQL is not case sensitive, but, most developers write the SQL keywords all uppercase, because it makes these statements easier to read.

White Space

- The line breaks also don’t matter.
- SQL is not sensitive to white space.
- You can split the SQL statement into multiple lines for readability.

Semicolon

- Officially, a complete SQL statement should be ended with a semicolon, though, many DBMS just don’t care.

Comments

- You can write comments in two ways; either using the c-style comment, or using two hyphen characters (*with no space between them*) followed by a space and the comment text.

```
/*
  this is a
  multi-line comment
*/
SELECT * FROM Employee; -- this is a single line comment
```

Operators

- An operator is a reserved word used to perform an operation like combining between expressions, comparisons, or arithmetic operations.

Commonly Used Operators

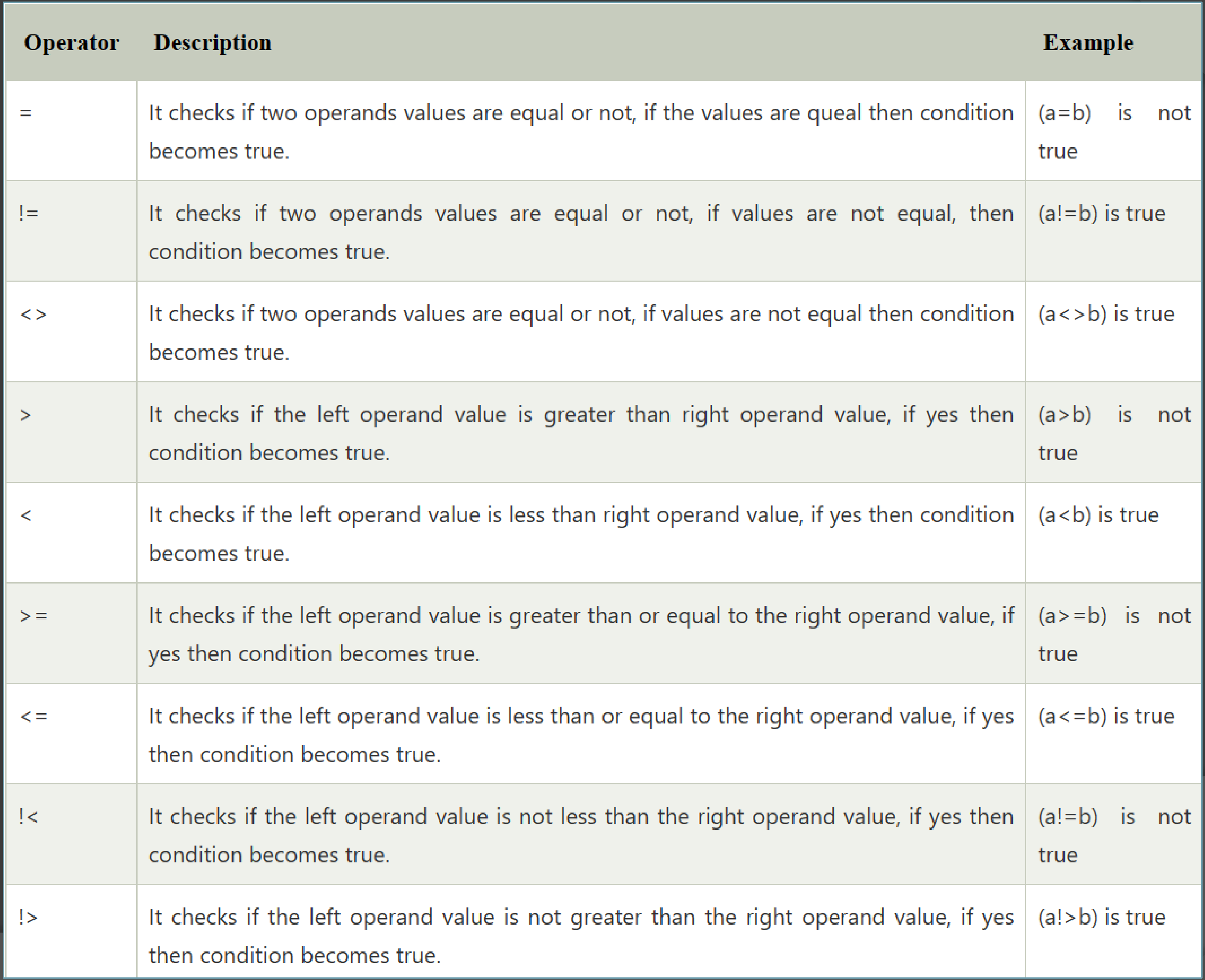
- Comparison Operators
- Logical Operators
- Arithmetic Operators

These operators can be combined with other SQL clauses like **SELECT, WHERE, JOIN**, and **HAVING** to perform various operations and conditions on data.

The specific availability and behavior of operators may vary slightly depending on the database management system being used.

Comparison Operators

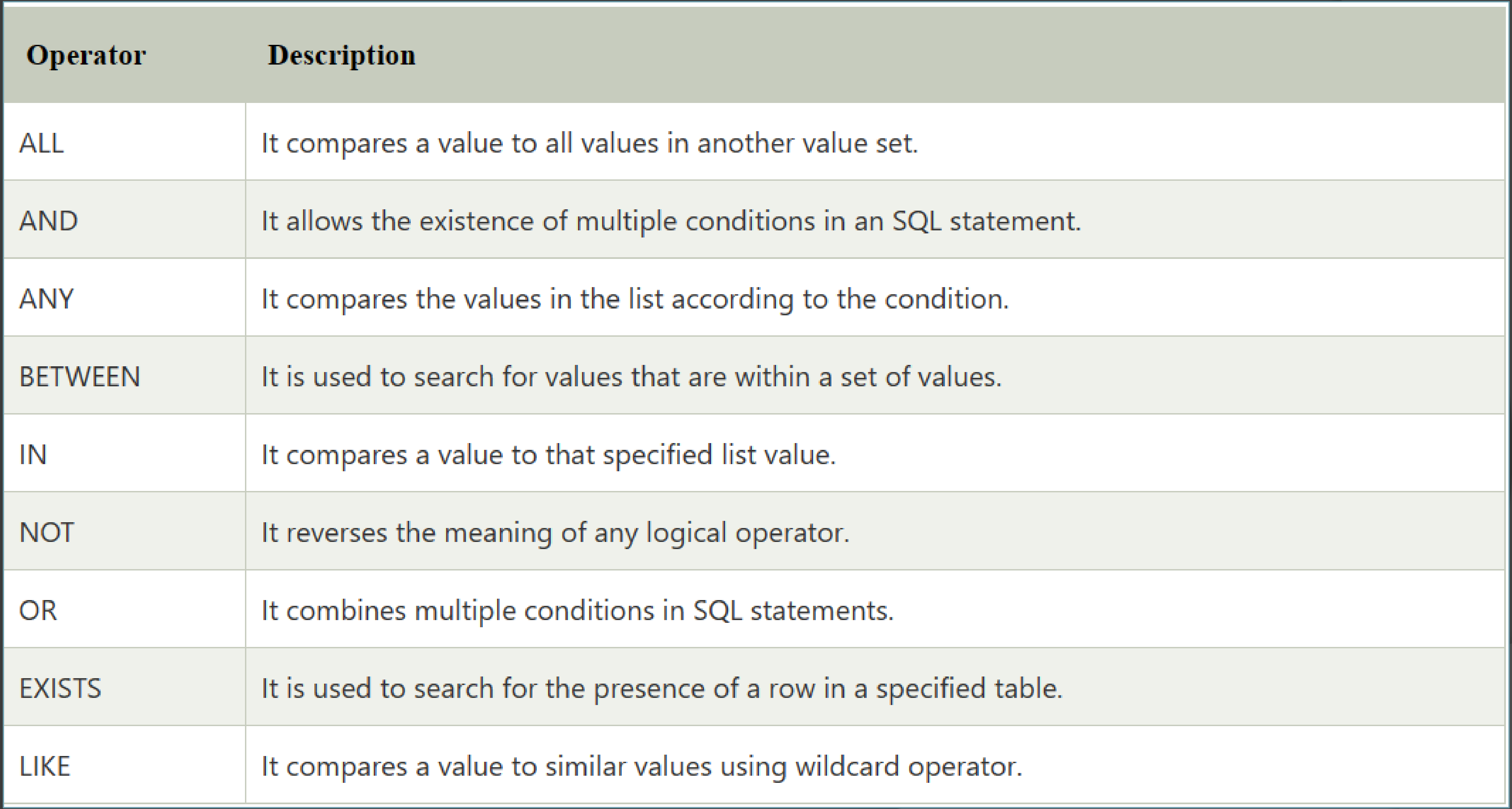
- Used to compare values and return a boolean result.



Operator	Description	Example
=	It checks if two operands values are equal or not, if the values are queal then condition becomes true.	(a=b) is not true
!=	It checks if two operands values are equal or not, if values are not equal, then condition becomes true.	(a!=b) is true
<>	It checks if two operands values are equal or not, if values are not equal then condition becomes true.	(a<>b) is true
>	It checks if the left operand value is greater than right operand value, if yes then condition becomes true.	(a>b) is not true
<	It checks if the left operand value is less than right operand value, if yes then condition becomes true.	(a<b) is true
>=	It checks if the left operand value is greater than or equal to the right operand value, if yes then condition becomes true.	(a>=b) is not true
<=	It checks if the left operand value is less than or equal to the right operand value, if yes then condition becomes true.	(a<=b) is true
!<	It checks if the left operand value is not less than the right operand value, if yes then condition becomes true.	(a!=b) is not true
!>	It checks if the left operand value is not greater than the right operand value, if yes then condition becomes true.	(a!>b) is true

Logical Operators

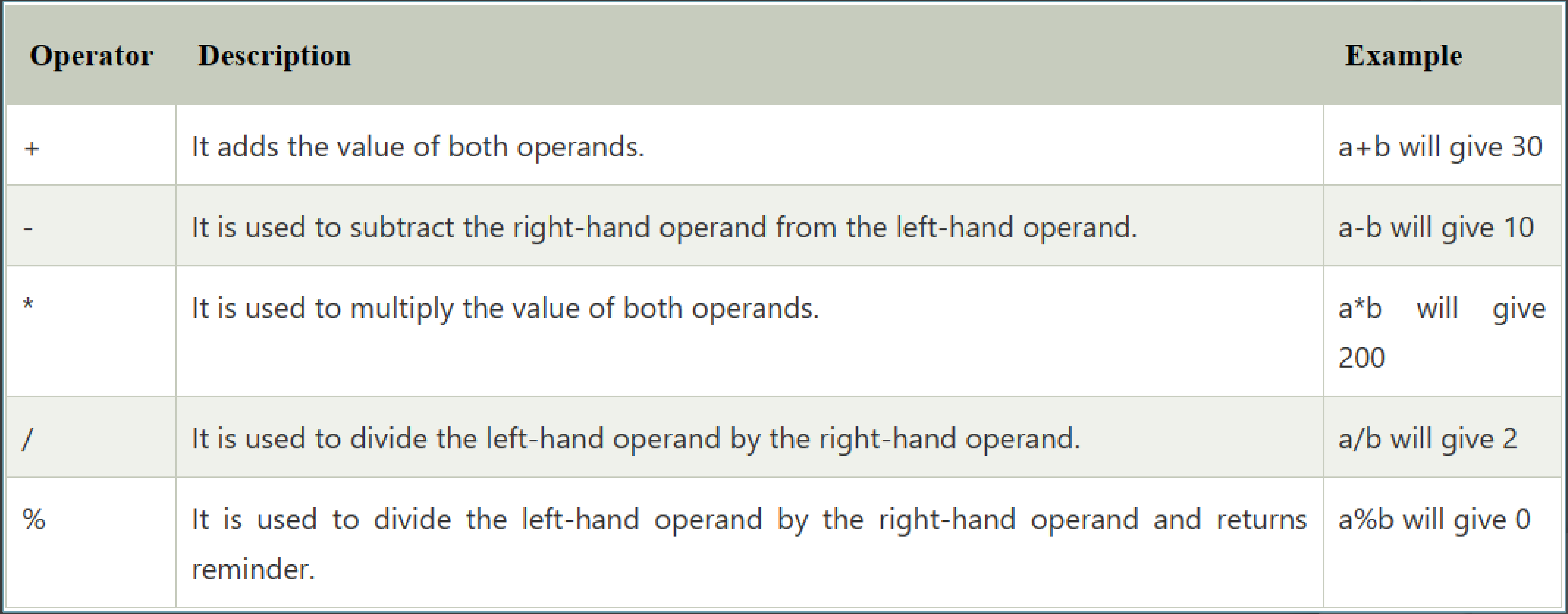
- Used to combine multiple conditions or evaluate logical expressions.



Operator	Description
ALL	It compares a value to all values in another value set.
AND	It allows the existence of multiple conditions in an SQL statement.
ANY	It compares the values in the list according to the condition.
BETWEEN	It is used to search for values that are within a set of values.
IN	It compares a value to that specified list value.
NOT	It reverses the meaning of any logical operator.
OR	It combines multiple conditions in SQL statements.
EXISTS	It is used to search for the presence of a row in a specified table.
LIKE	It compares a value to similar values using wildcard operator.

Arithmetic Operators

- Used for mathematical operations.



Operator	Description	Example
+	It adds the value of both operands.	a+b will give 30
-	It is used to subtract the right-hand operand from the left-hand operand.	a-b will give 10
*	It is used to multiply the value of both operands.	a*b will give 200
/	It is used to divide the left-hand operand by the right-hand operand.	a/b will give 2
%	It is used to divide the left-hand operand by the right-hand operand and returns reminder.	a%b will give 0

NULL Comparison Operators

- Used to check for **NULL** values.
  - IS NULL (checks if a value is NULL)
  - IS NOT NULL (checks if a value is not NULL)

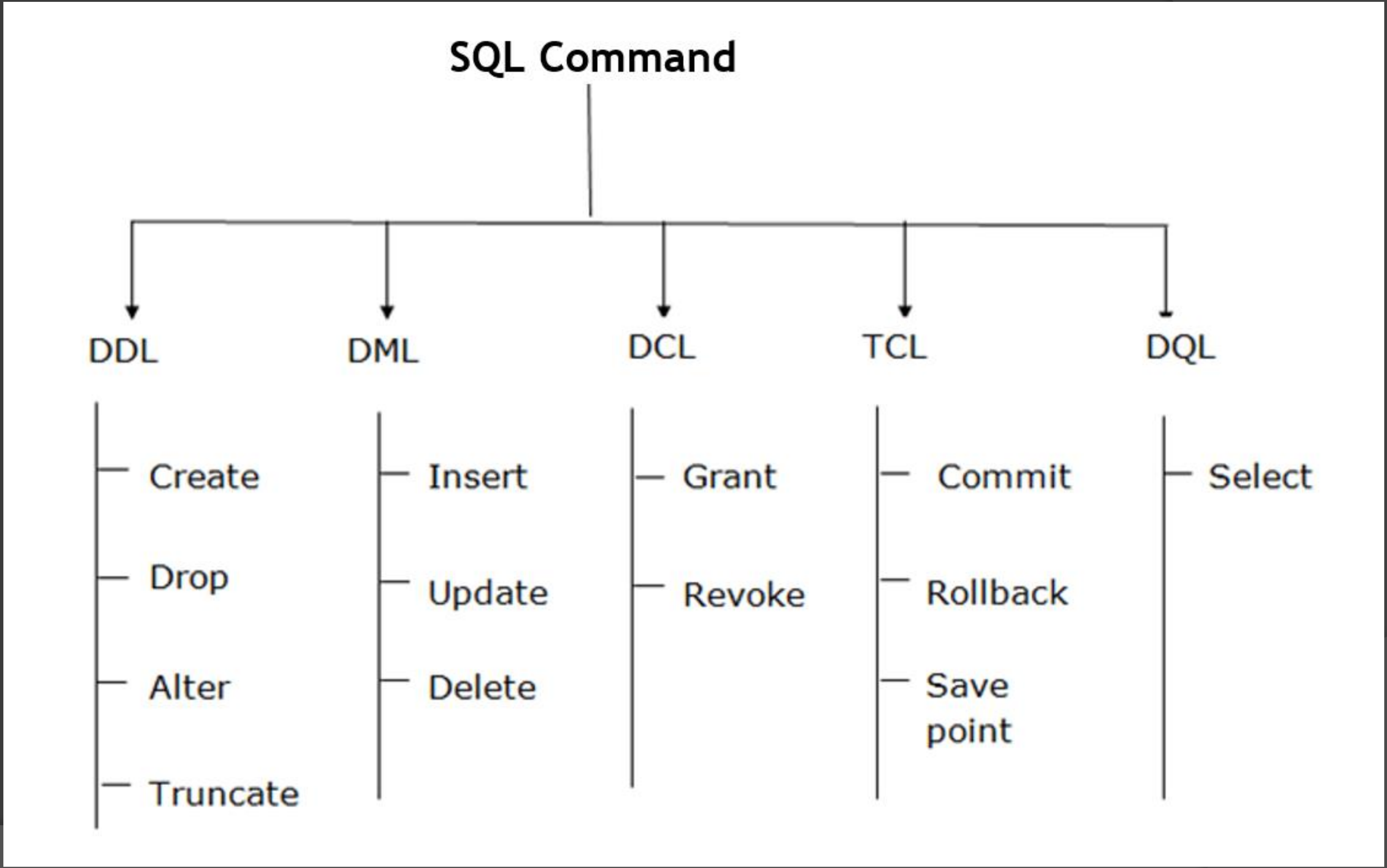
SQL Command

SQL Command

- SQL commands are instructions.
- It is used to communicate with the database.
- It is also used to perform specific tasks, functions, and queries of data.
- SQL can perform various tasks like create a table, add data to tables, drop the table, modify the table, set permission for users.

Types of SQL Commands

- DDL - Data Definition Language
- DML - Data Manipulation Language
- DCL - Data Control Language
- TCL - Transaction Control Language
- DQL - Data Query Language


SQL Command
DDL
DML
DCL
TCL
DQL
Create
Drop
Alter
Truncate
Insert
Update
Delete
Grant
Revoke
Commit
Rollback
Save point
Select

Some of The Most Important SQL Commands

- SELECT - extracts data from a database
- UPDATE - updates data in a database
- DELETE - deletes data from a database
- INSERT INTO - inserts new data into a database
- CREATE DATABASE - creates a new database
- ALTER DATABASE - modifies a database
- CREATE TABLE - creates a new table
- ALTER TABLE - modifies a table
- DROP TABLE - deletes a table
- CREATE INDEX - creates an index (search key)
- DROP INDEX - deletes an index

DDL Commands

CREATE DATABASE

- It is used to create a new SQL database.
- Syntax:

```
CREATE DATABASE database_name;
```

CREATE DATABASE

- Examples

```
[MySQL]

CREATE DATABASE testdb;

CREATE DATABASE test_db CHARACTER SET utf8;



[MSSQL]

CREATE DATABASE testdb
GO
```

DROP DATABASE

- It is used to drop an existing database

- Syntax:

```
DROP DATABASE database_name;
```

DROP DATABASE

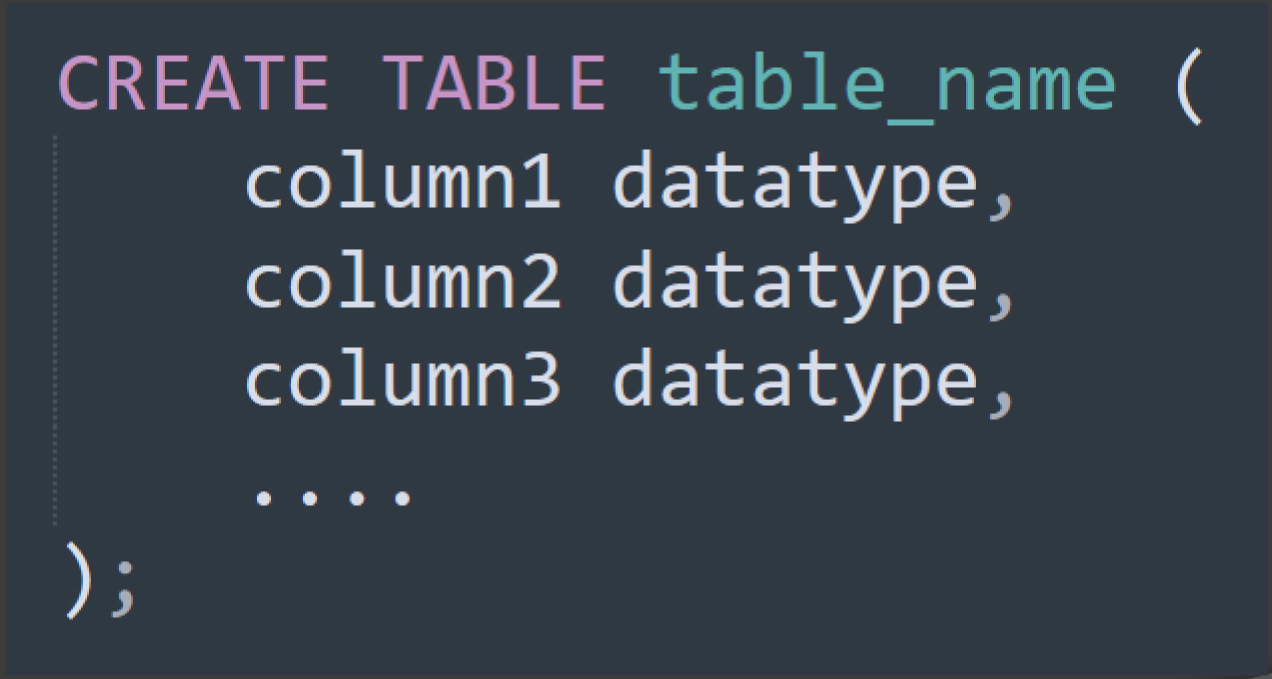
- Example

```
[MySQL]

DROP DATABASE testdb;
```

```
[MSSQL]

DROP DATABASE testdb
GO
```

CREATE TABLE

- It is used to create a new table in a database
- Syntax:



```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
    ....
);
```

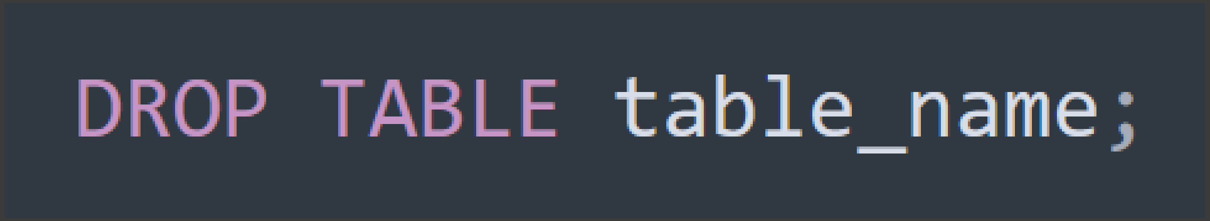
CREATE TABLE

- Example

```sql
CREATE TABLE employee (
    objid varchar(50) NOT NULL,
    idno varchar(10) NOT NULL,
    name varchar(100) NOT NULL,
    address varchar(255),
    contactno varchar(11),
    salary decimal(10,2) NOT NULL DEFAULT '0'
);
```

DROP TABLE

- It is used to drop an existing table in a database
- Syntax:



```
DROP TABLE table_name;
```

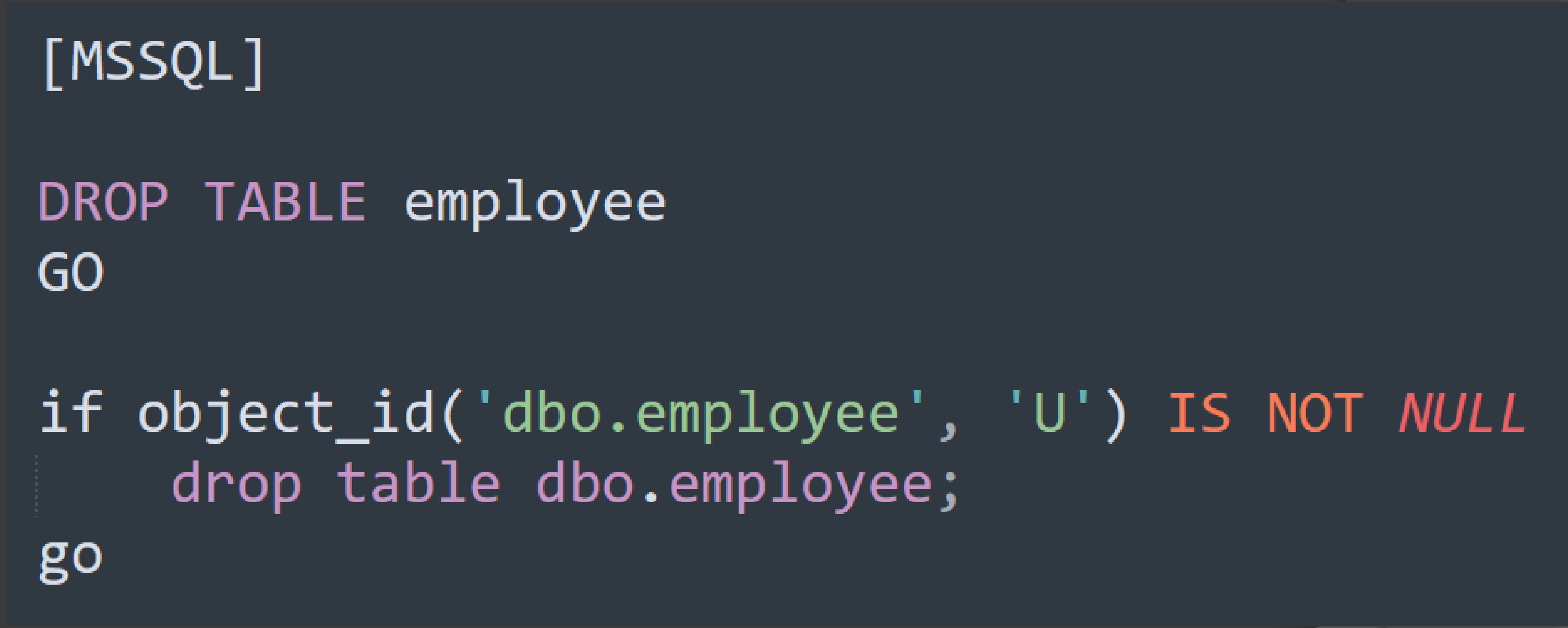
DROP TABLE

- Example

```
[MySQL]

DROP TABLE employee;

DROP TABLE IF EXISTS employee;
```

DROP TABLE

- Example



```
[MSSQL]

DROP TABLE employee
GO

if object_id('dbo.employee', 'U') IS NOT NULL
    drop table dbo.employee;
go
```

ALTER TABLE

- It is used to add, delete, or modify columns in an existing table
- It is also used to add and drop various constraints on an existing table.

ALTER TABLE - Add Column

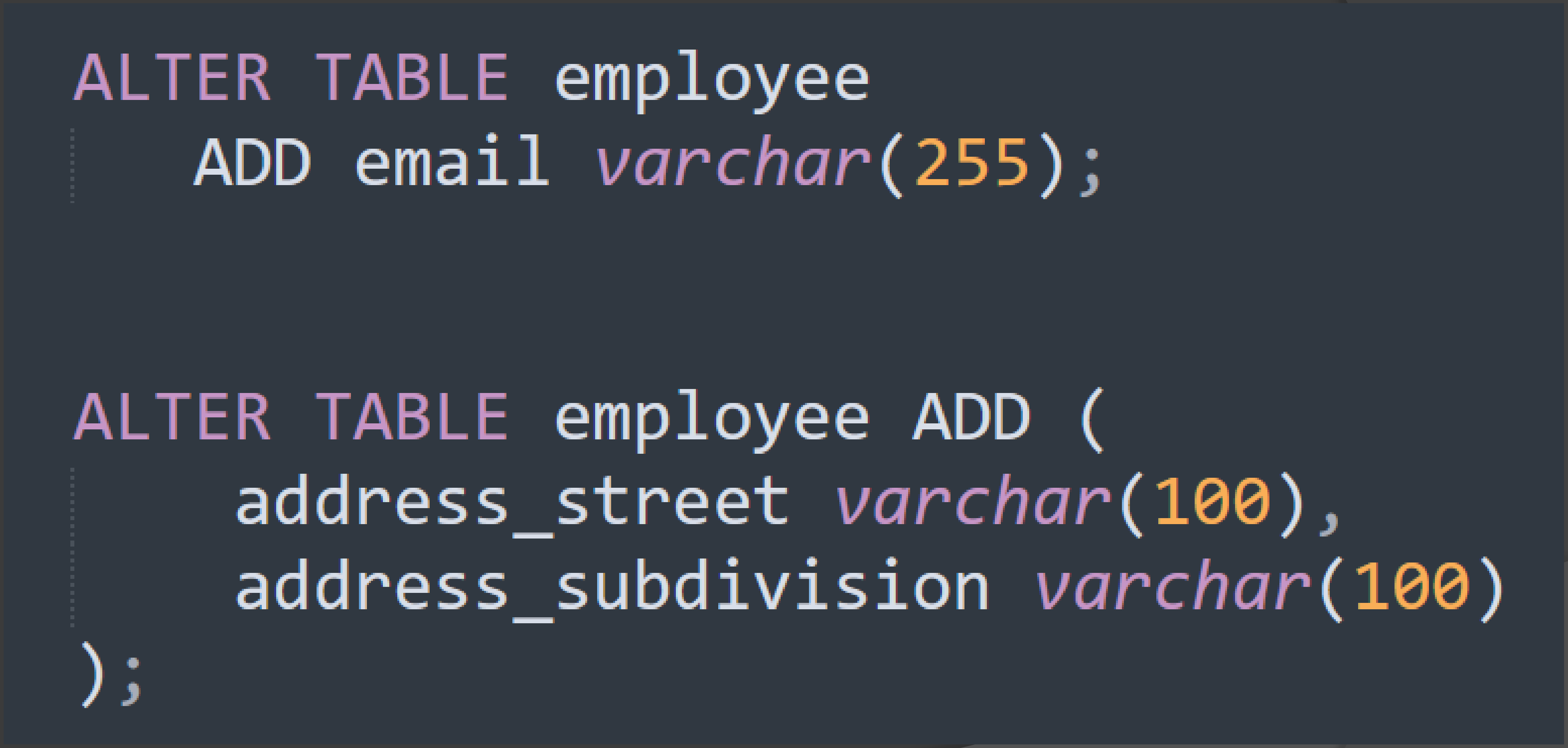
- To add a column in a table

- Syntax:

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

ALTER TABLE - Add Column

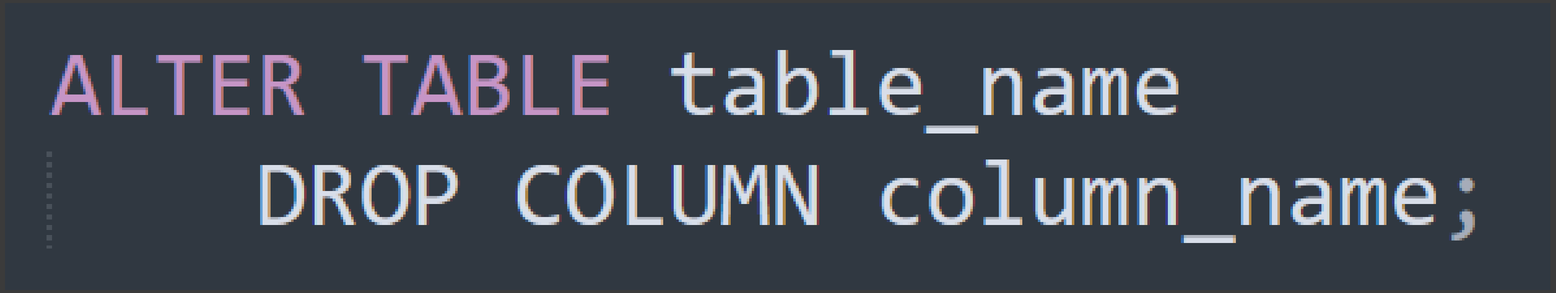
- Example



```
ALTER TABLE employee
    ADD email varchar(255);


ALTER TABLE employee ADD (
      address_street varchar(100),
      address_subdivision varchar(100)
);
```

ALTER TABLE - Drop Column

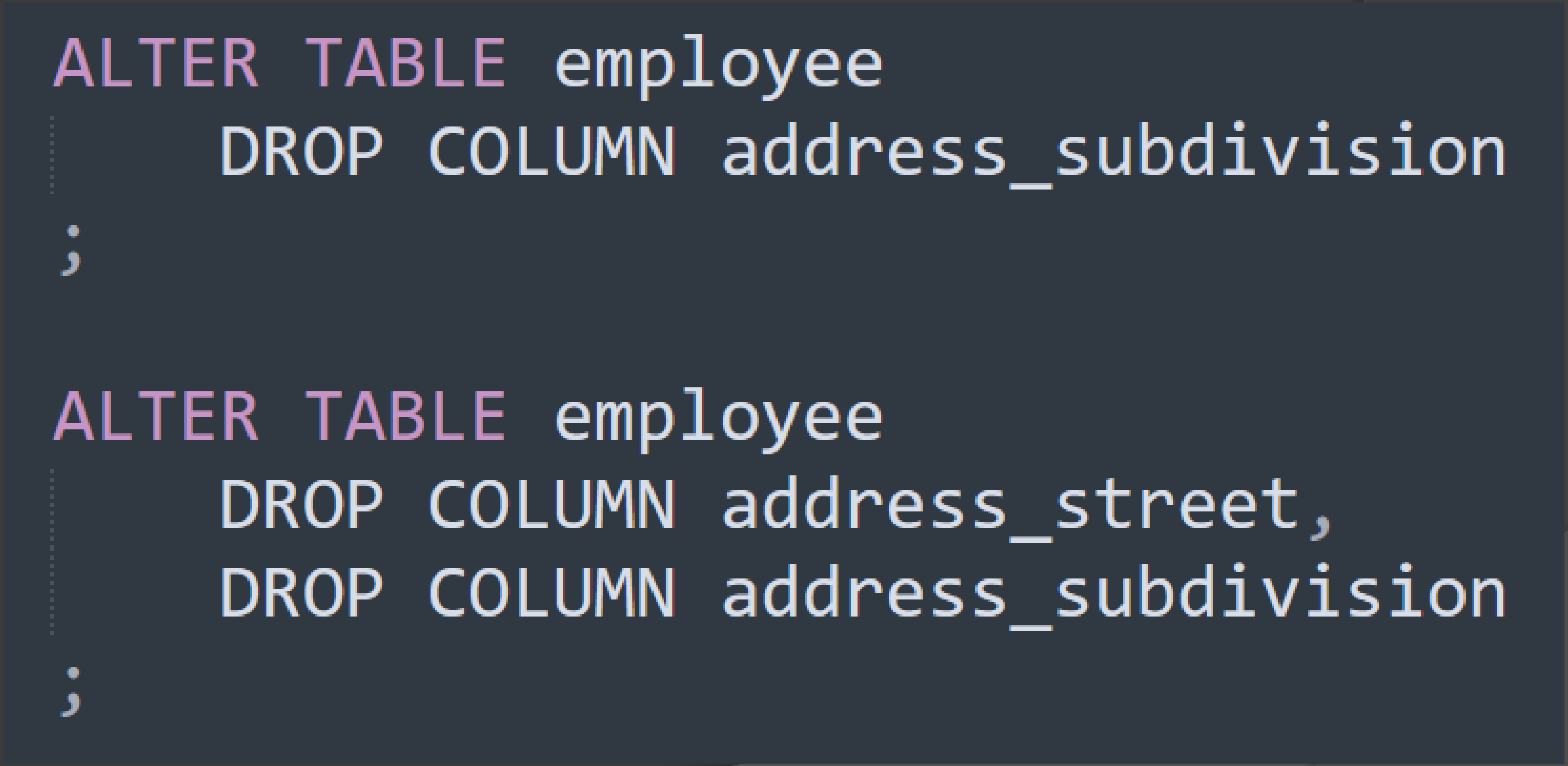
- To delete a column in a table
- Syntax:



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

ALTER TABLE - Drop Column

- Example



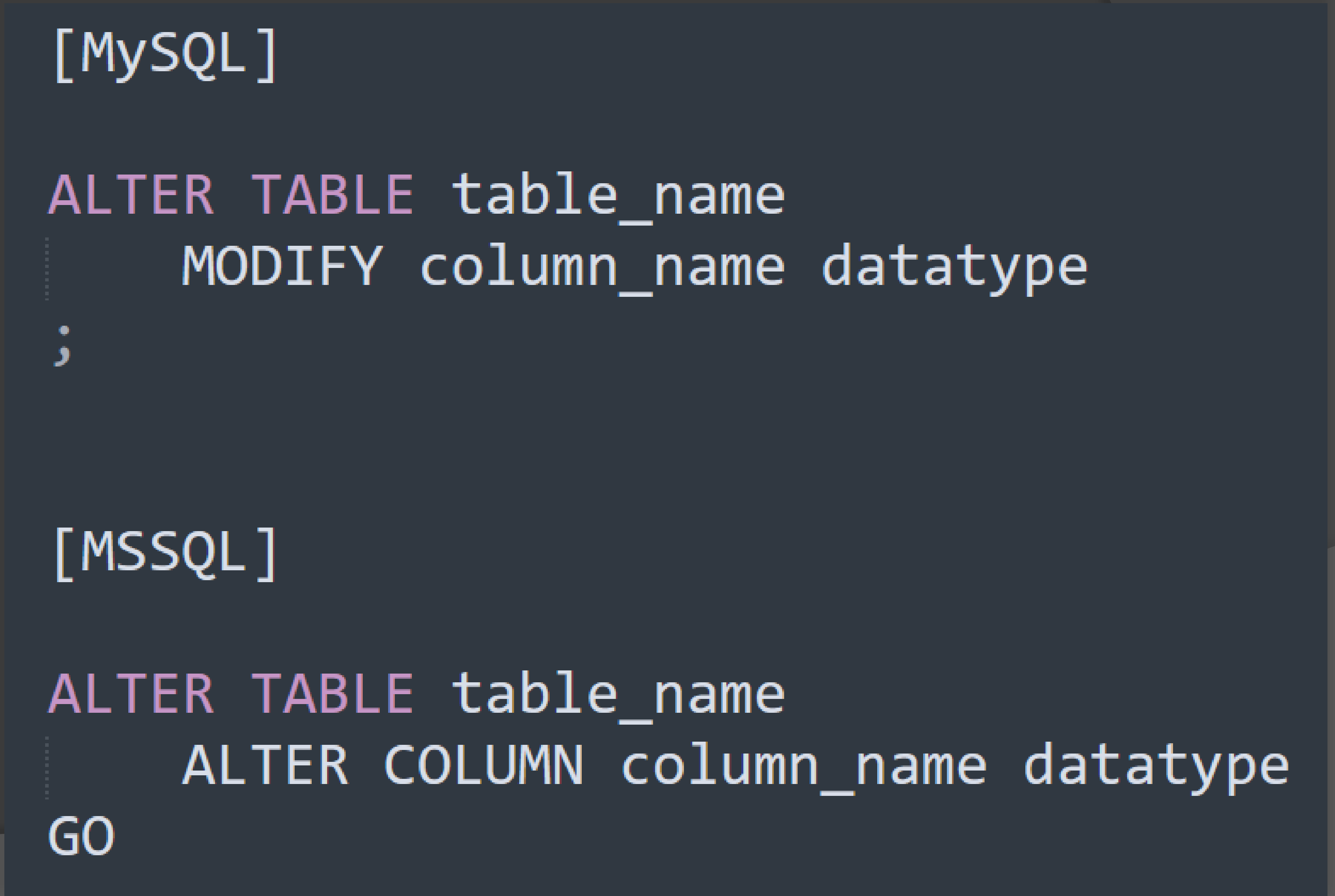
```
ALTER TABLE employee
    DROP COLUMN address_subdivision
;

ALTER TABLE employee
    DROP COLUMN address_street,
    DROP COLUMN address_subdivision
;
```

ALTER TABLE - Modify Column

- To change the data type of a column in a table

ALTER TABLE - Modify Column

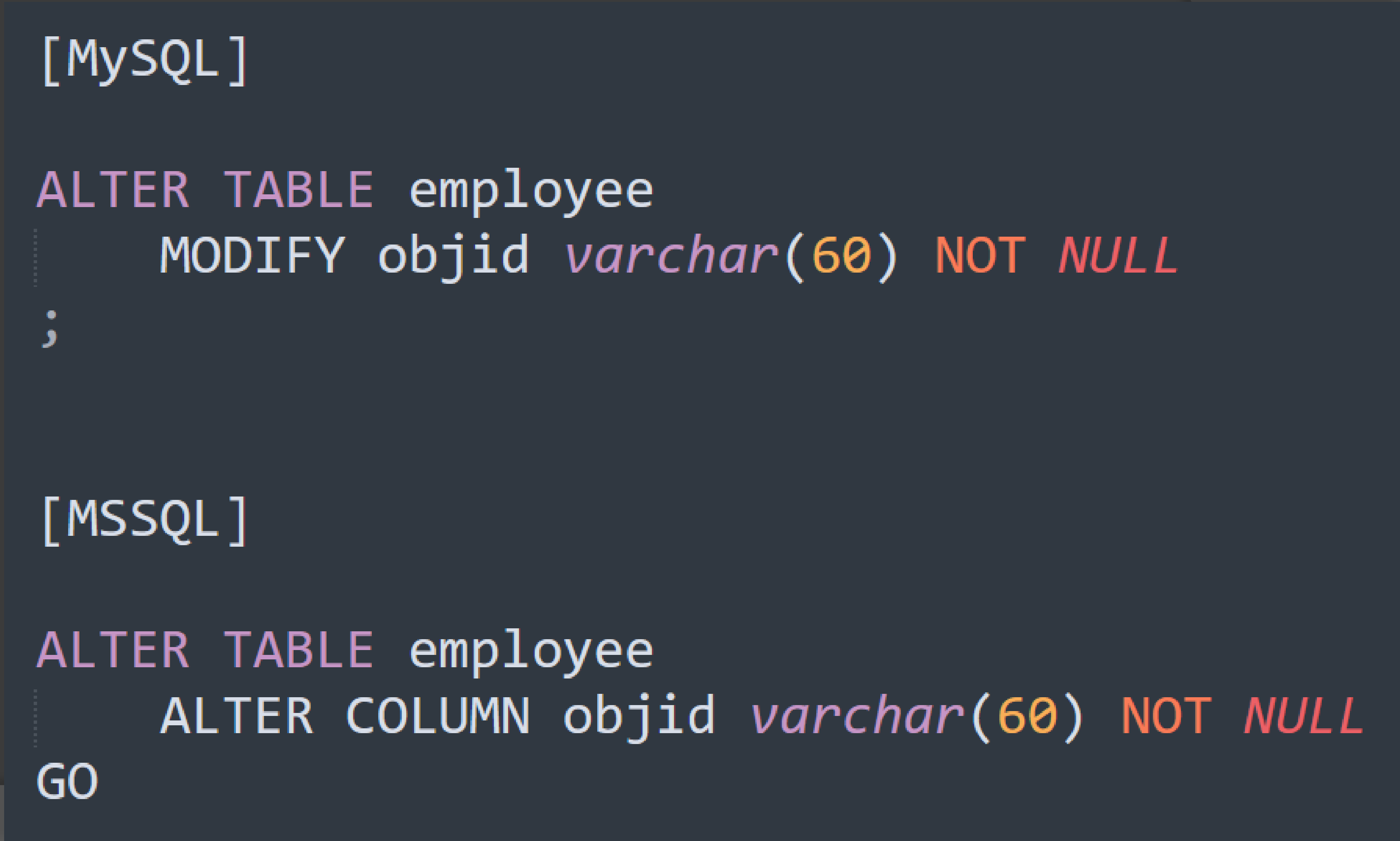
- Syntax:



```
[MySQL]

ALTER TABLE table_name
    MODIFY column_name datatype
;


[MSSQL]

ALTER TABLE table_name
    ALTER COLUMN column_name datatype
GO
```

ALTER TABLE - Modify Column

- Example:



```
[MySQL]

ALTER TABLE employee
    MODIFY objid varchar(60) NOT NULL
;


[MSSQL]

ALTER TABLE employee
    ALTER COLUMN objid varchar(60) NOT NULL
GO
```

ALTER TABLE - Rename Column

- To rename column in a table

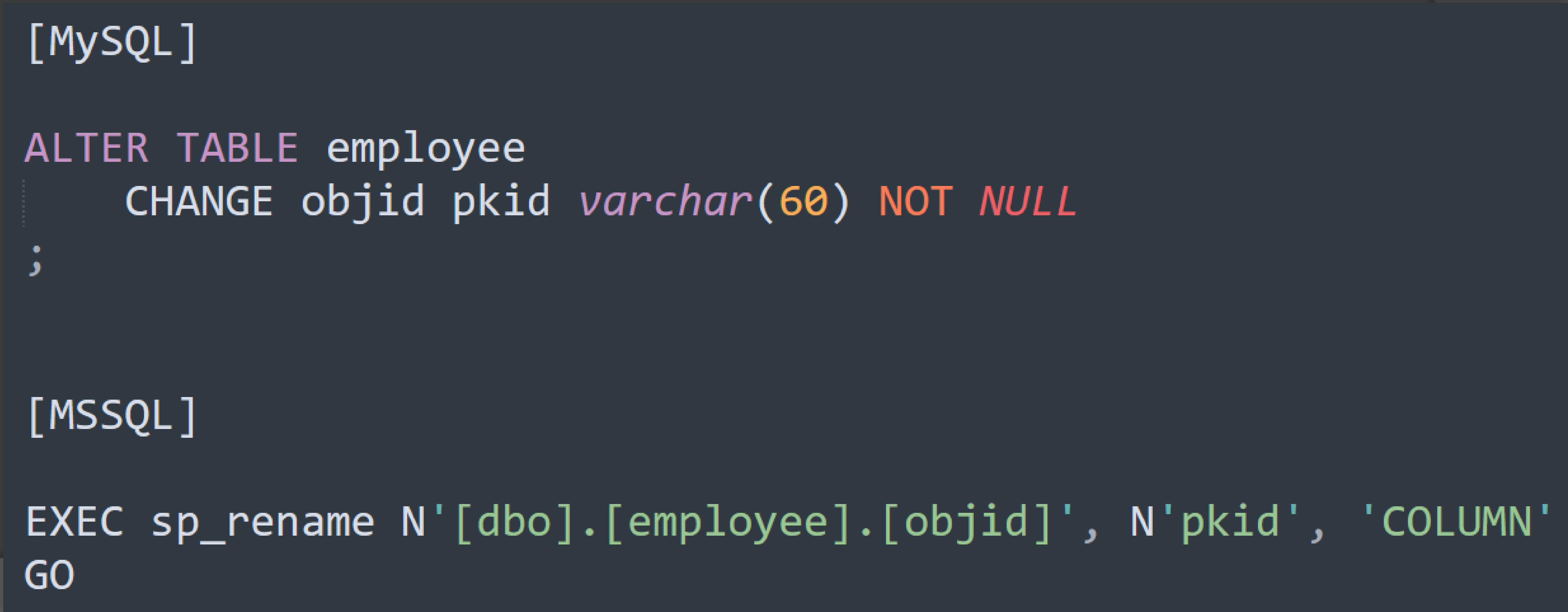
ALTER TABLE - Rename Column

- Syntax

```
[MySQL]

ALTER TABLE table_name
    CHANGE col_name newcol_name datatype
;



[MSSQL]

EXEC sp_rename N'[dbo].[table_name].[col_name]', N'newcol_name', 'COLUMN'
GO
```

ALTER TABLE - Rename Column

- Example

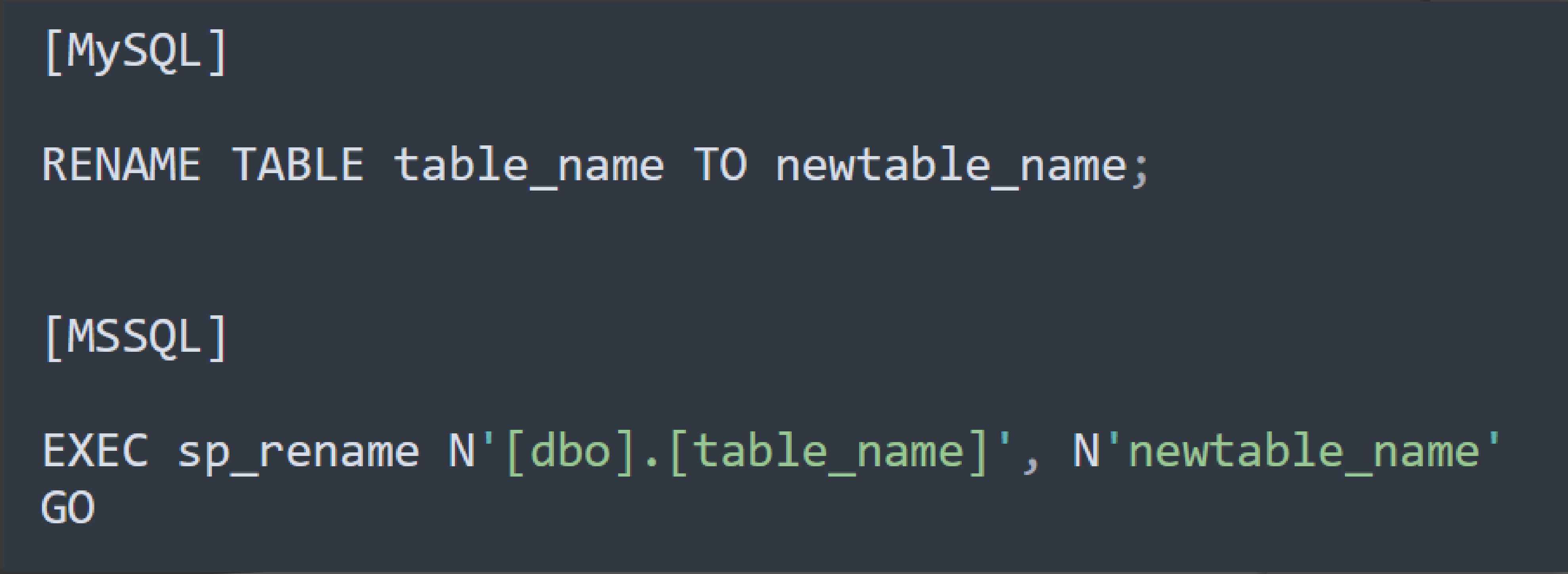


```
[MySQL]

ALTER TABLE employee
    CHANGE objid pkid varchar(60) NOT NULL
;



[MSSQL]

EXEC sp_rename N'[dbo].[employee].[objid]', N'pkid', 'COLUMN'
GO
```

RENAME TABLE

- It is used to rename an existing table in a database

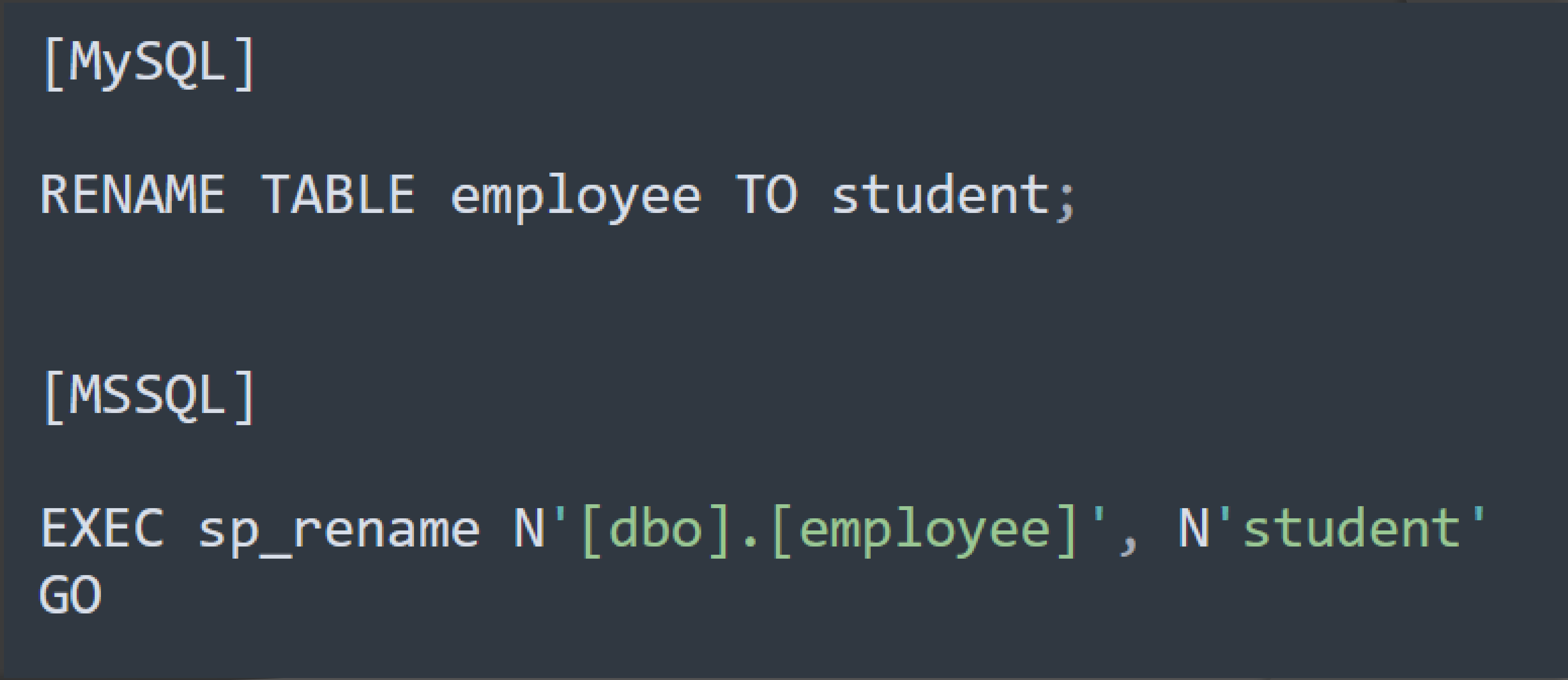
- Syntax



```
[MySQL]

RENAME TABLE table_name TO newtable_name;


[MSSQL]

EXEC sp_rename N'[dbo].[table_name]', N'newtable_name'
GO
```

RENAME TABLE

- Example



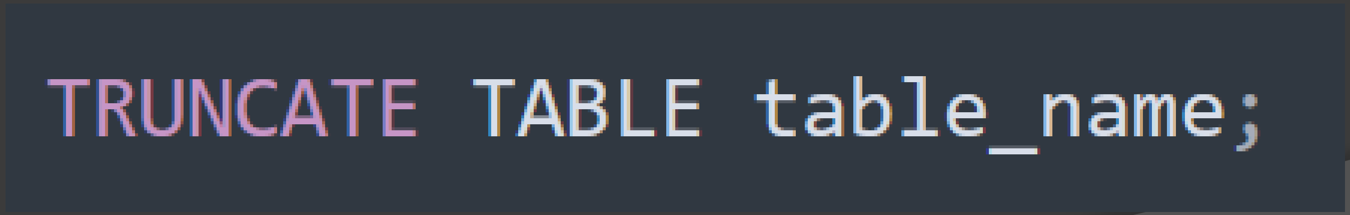
```
[MySQL]

RENAME TABLE employee TO student;


[MSSQL]

EXEC sp_rename N'[dbo].[employee]', N'student'
GO
```

TRUNCATE TABLE

- It is used to delete all rows from a table, effectively removing all data stored in the table.

- It is a fast and efficient way to delete all records, as it bypasses the transaction log, freeing up storage space without logging individual row deletions.

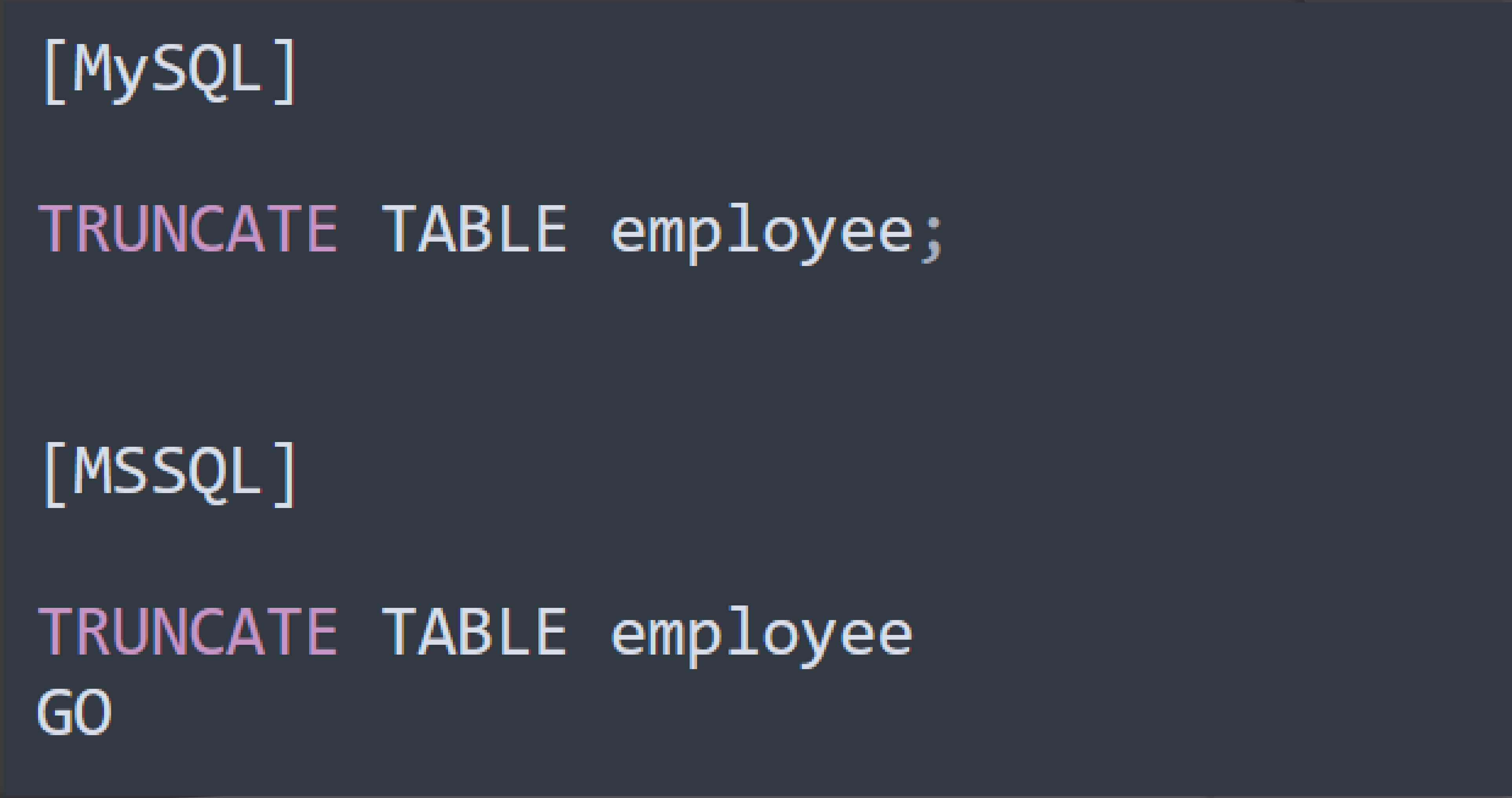
- Syntax:



```sql
TRUNCATE TABLE table_name;
```

TRUNCATE TABLE

- Example



```
[MySQL]

TRUNCATE TABLE employee;



[MSSQL]

TRUNCATE TABLE employee
GO
```

SQL Constraints

SQL Constraints

- Constraints are used to specify rules for the data in a table.
- Constraints are used to limit the type of data that can go into a table.
- It ensures the accuracy and reliability of the data in the table.
- Constraints can be a column level or table level.
- Column level constraints apply to a column, and table level constraints apply to the whole table.

Commonly Used SQL Constraints

- **NOT NULL** - Ensures that a column cannot have a NULL value
- **UNIQUE** - Ensures that all values in a column are different
- **PRIMARY KEY** - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
- **FOREIGN KEY** - Prevents actions that would destroy links between tables

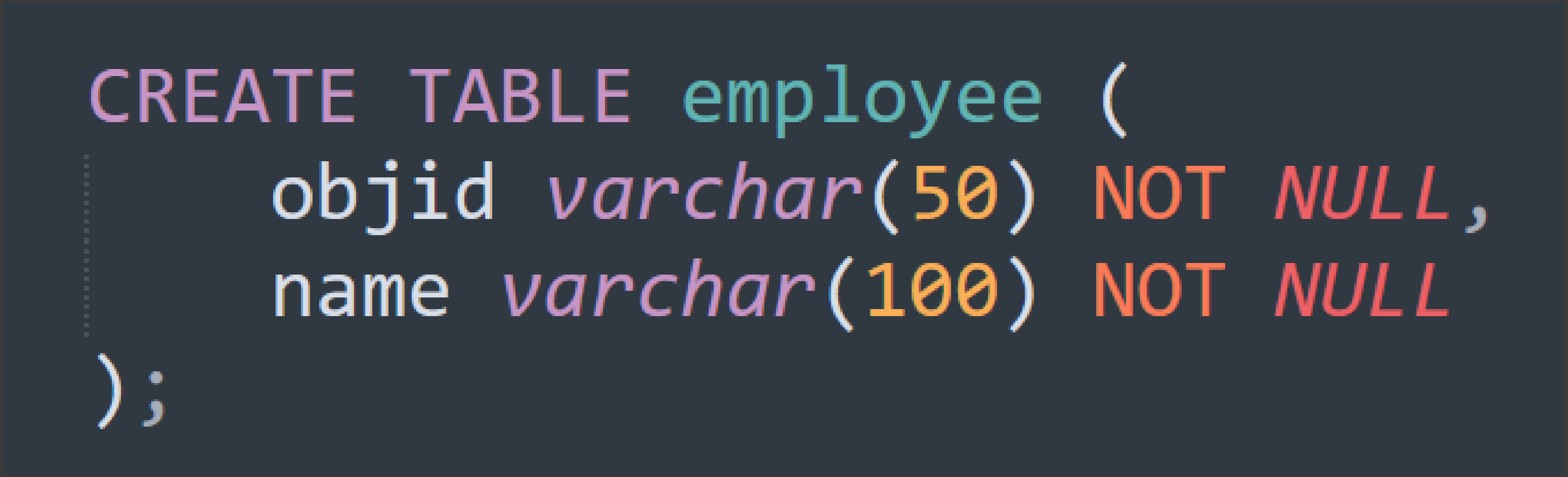
Commonly Used SQL Constraints

- **CHECK** - Ensures that the values in a column satisfies a specific condition
- **DEFAULT** - Sets a default value for a column if no value is specified
- **CREATE INDEX** - Used to create and retrieve data from the database very quickly

NOT NULL Constraint

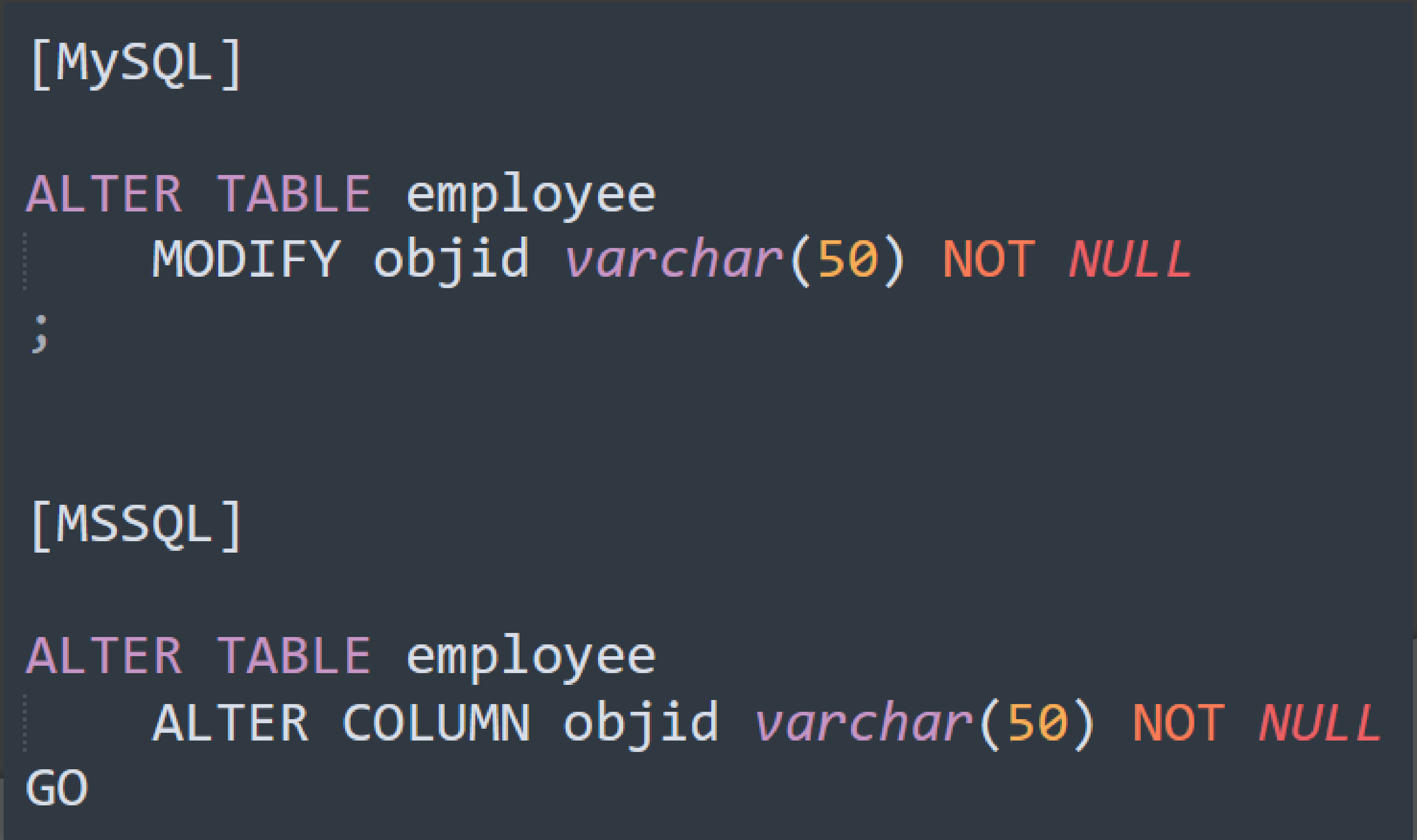
- Enforces a column NOT to accept NULL values, and that you cannot insert a new record, or update a record without adding a value to this column.

NOT NULL on CREATE TABLE



```sql
CREATE TABLE employee (
    objid varchar(50) NOT NULL,
    name varchar(100) NOT NULL
);
```

NOT NULL on ALTER TABLE



```
[MySQL]

ALTER TABLE employee
    MODIFY objid varchar(50) NOT NULL
;


[MSSQL]

ALTER TABLE employee
    ALTER COLUMN objid varchar(50) NOT NULL
GO
```

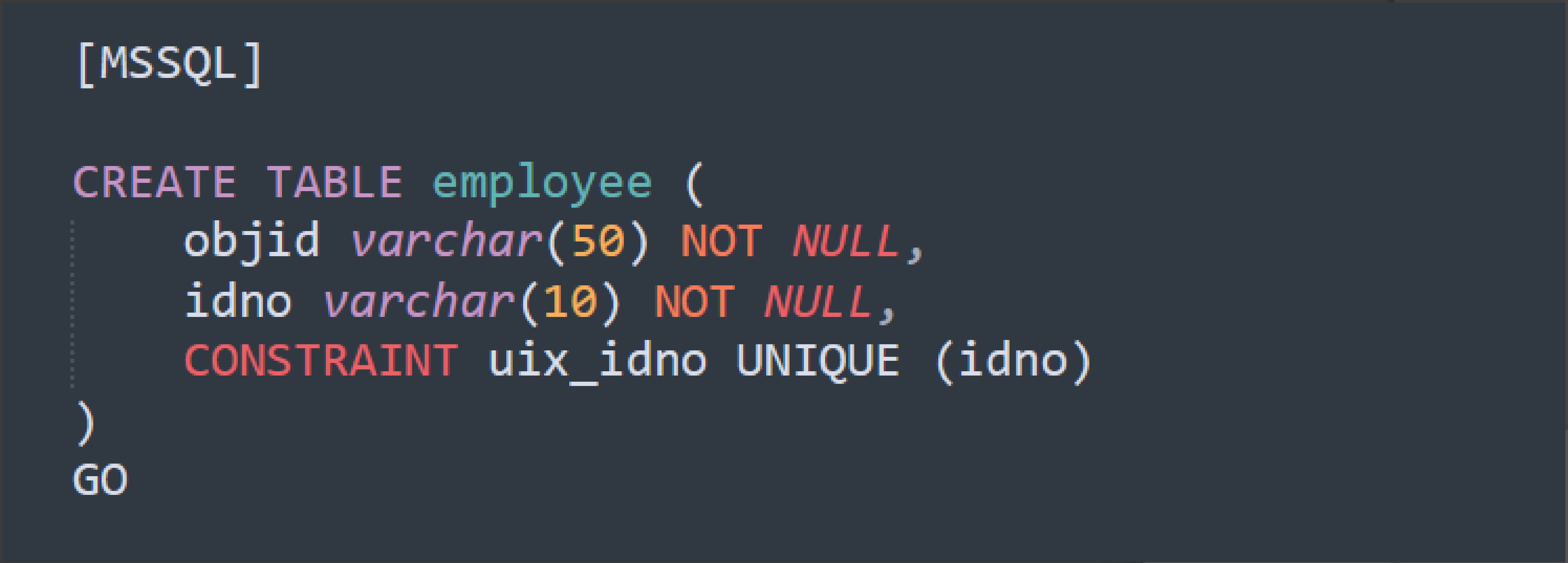
UNIQUE Constraint

- The UNIQUE constraint ensures that all values in a column are different.
- Both the UNIQUE and PRIMARY KEY constraints provide a guarantee for uniqueness for a column or set of columns.
- A PRIMARY KEY constraint automatically has a UNIQUE constraint.
- However, you can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table

UNIQUE Constraint on CREATE TABLE

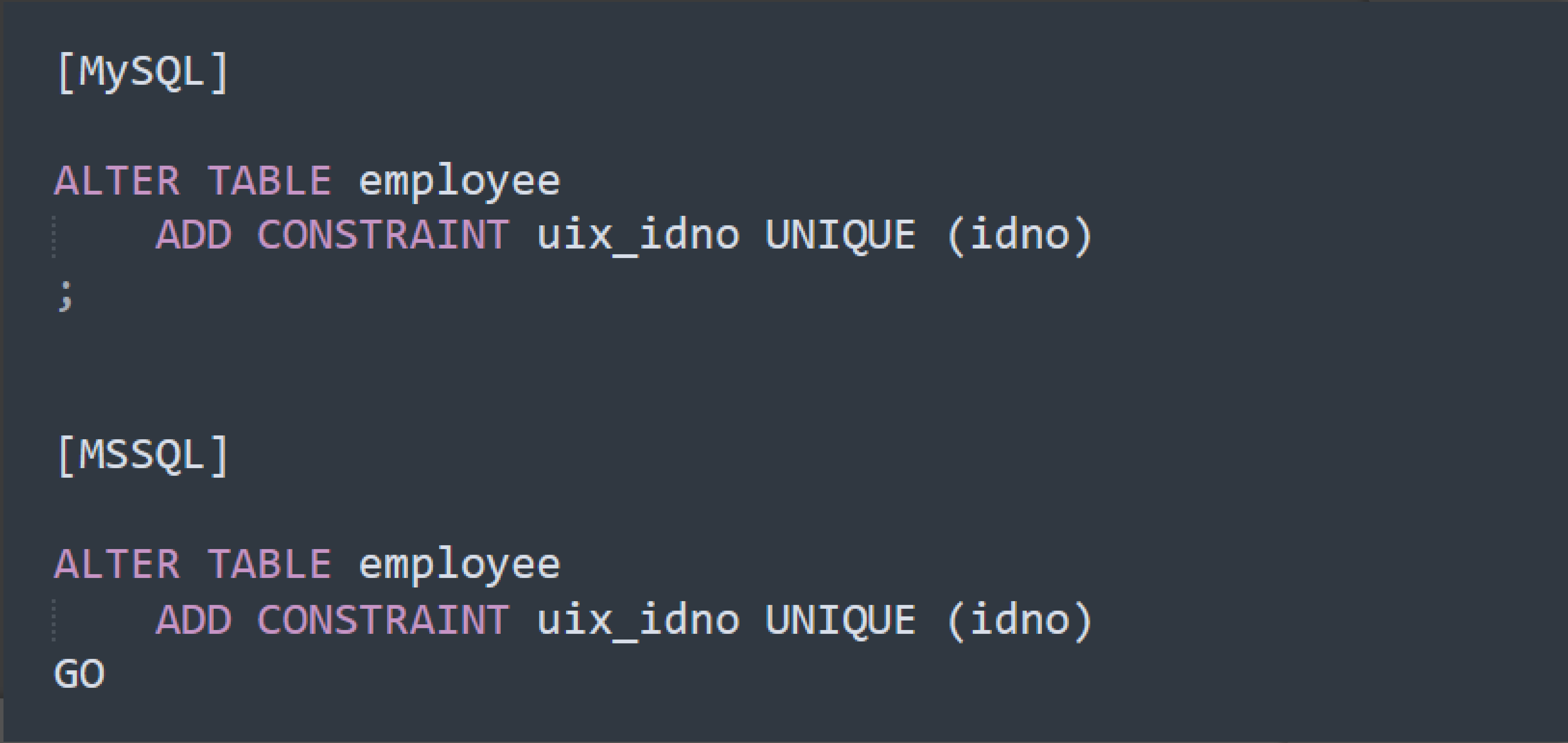
```
[MySQL]

CREATE TABLE employee (
    objid varchar(50) NOT NULL,
    idno varchar(10) NOT NULL,
    UNIQUE KEY uix_idno (idno)
);

CREATE TABLE employee (
    objid varchar(50) NOT NULL,
    idno varchar(10) NOT NULL,
    CONSTRAINT uix_idno UNIQUE (idno)
);
```

UNIQUE Constraint on CREATE TABLE



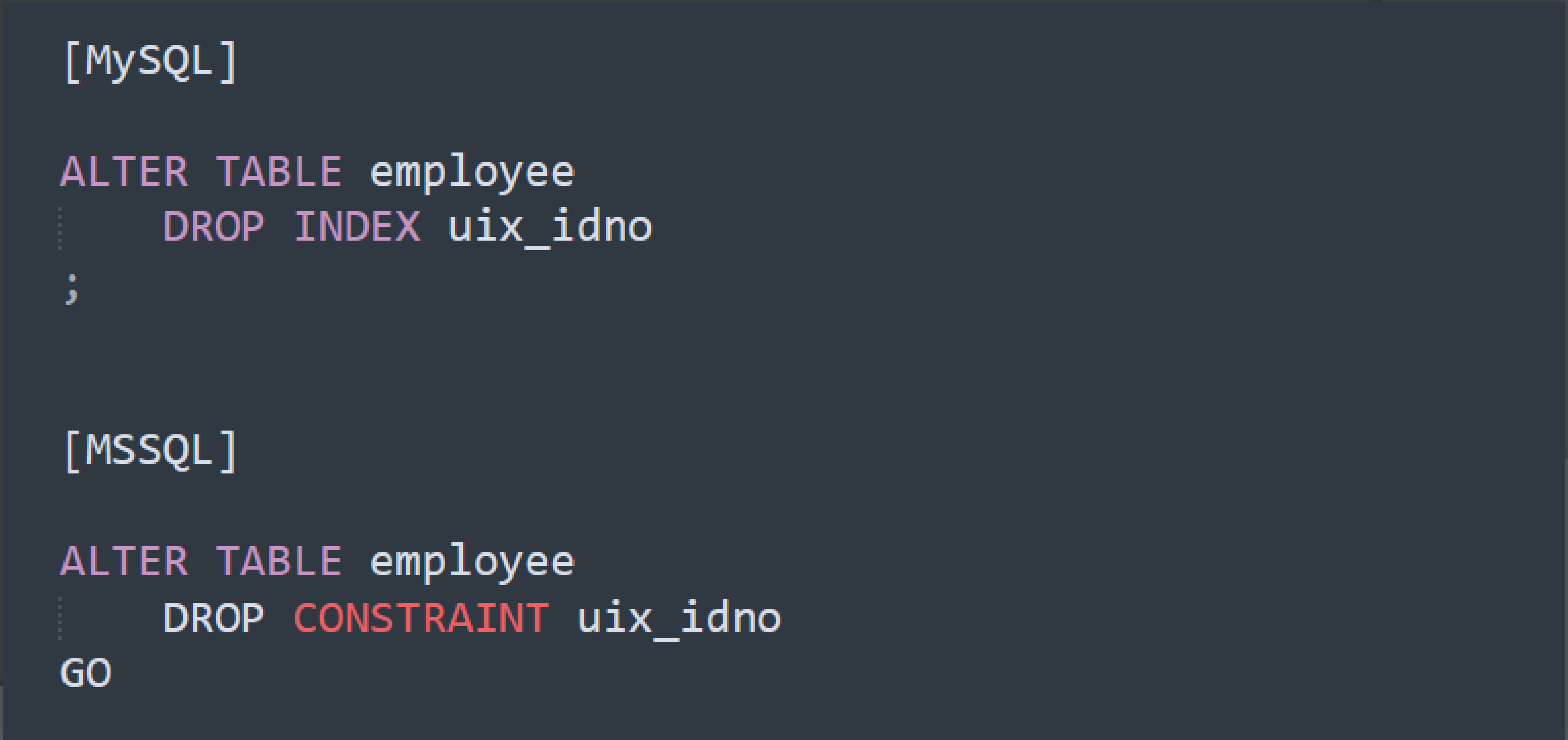
```
[MSSQL]

CREATE TABLE employee (
    objid varchar(50) NOT NULL,
    idno varchar(10) NOT NULL,
    CONSTRAINT uix_idno UNIQUE (idno)
)
GO
```

UNIQUE Constraint on ALTER TABLE



```
[MySQL]

ALTER TABLE employee
    ADD CONSTRAINT uix_idno UNIQUE (idno)
;


[MSSQL]

ALTER TABLE employee
    ADD CONSTRAINT uix_idno UNIQUE (idno)
GO
```

DROP UNIQUE Constraint



```
[MySQL]

ALTER TABLE employee
    DROP INDEX uix_idno
;


[MSSQL]

ALTER TABLE employee
    DROP CONSTRAINT uix_idno
GO
```

PRIMARY KEY Constraint

- The PRIMARY KEY constraint uniquely identifies each record in a table.
- Primary keys must contain UNIQUE values, and cannot contain NULL values.
- A table can have only ONE primary key; and in the table, this primary key can consist of single or multiple columns (fields).

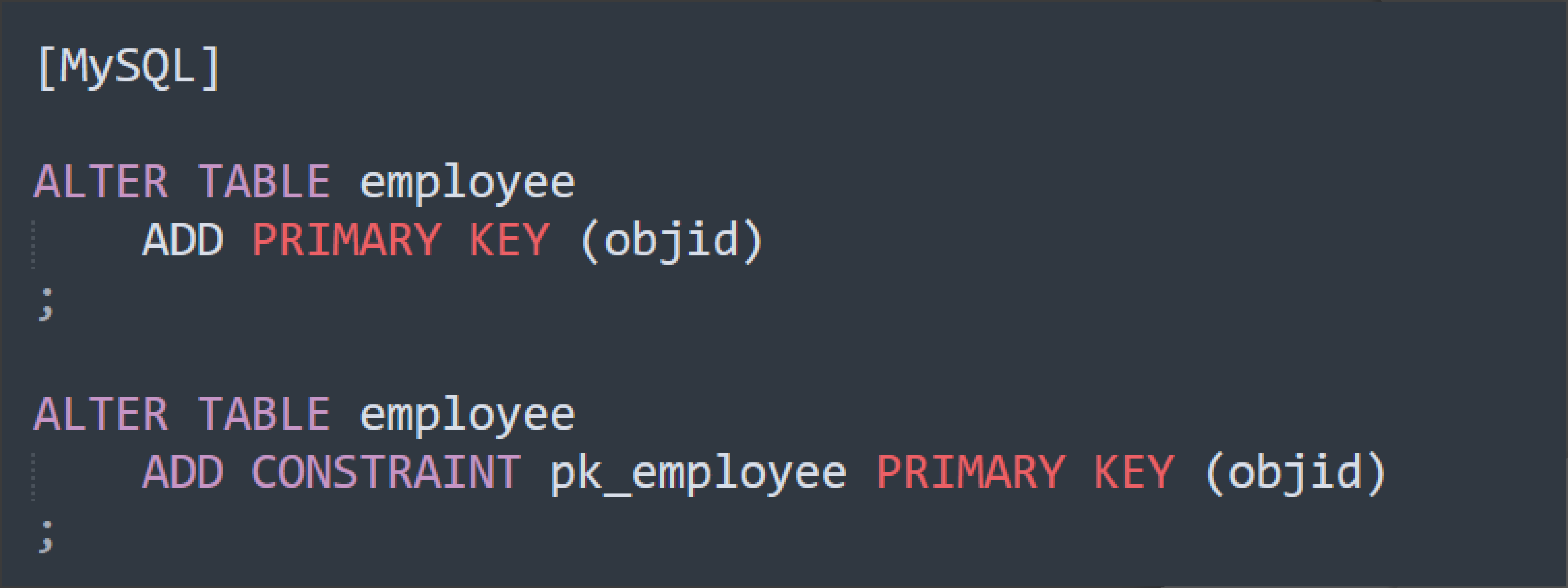
PRIMARY KEY Constraint

On CREATE TABLE

```
[MySQL]

CREATE TABLE employee (
    objid varchar(50) NOT NULL,
    CONSTRAINT pk_employee PRIMARY KEY (objid)
);


[MSSQL]

CREATE TABLE employee (
    objid varchar(50) NOT NULL,
    CONSTRAINT pk_employee PRIMARY KEY (objid)
)
GO
```

PRIMARY KEY Constraint

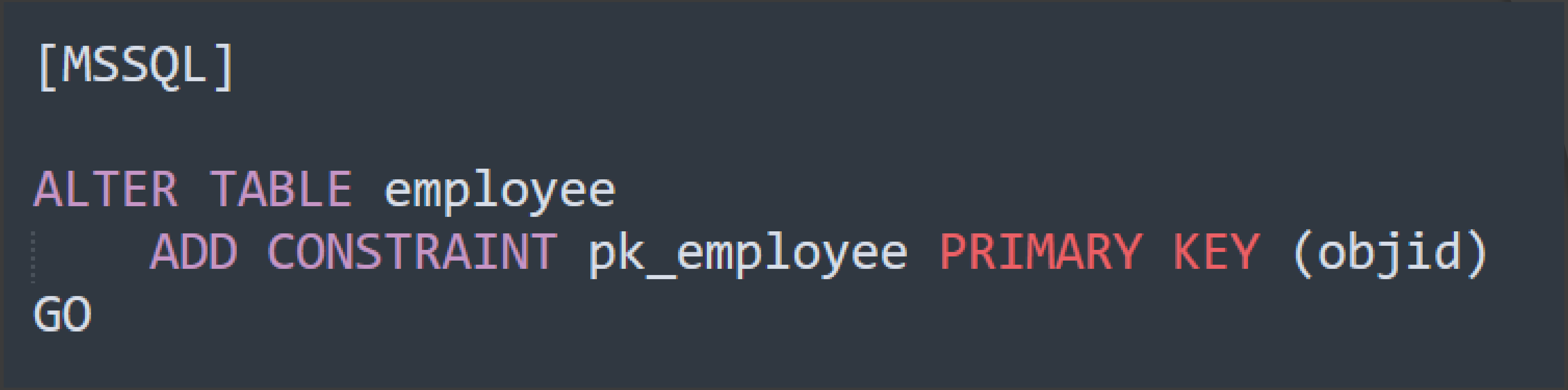
On ALTER TABLE



```
[MySQL]

ALTER TABLE employee
    ADD PRIMARY KEY (objid)
;

ALTER TABLE employee
    ADD CONSTRAINT pk_employee PRIMARY KEY (objid)
;
```

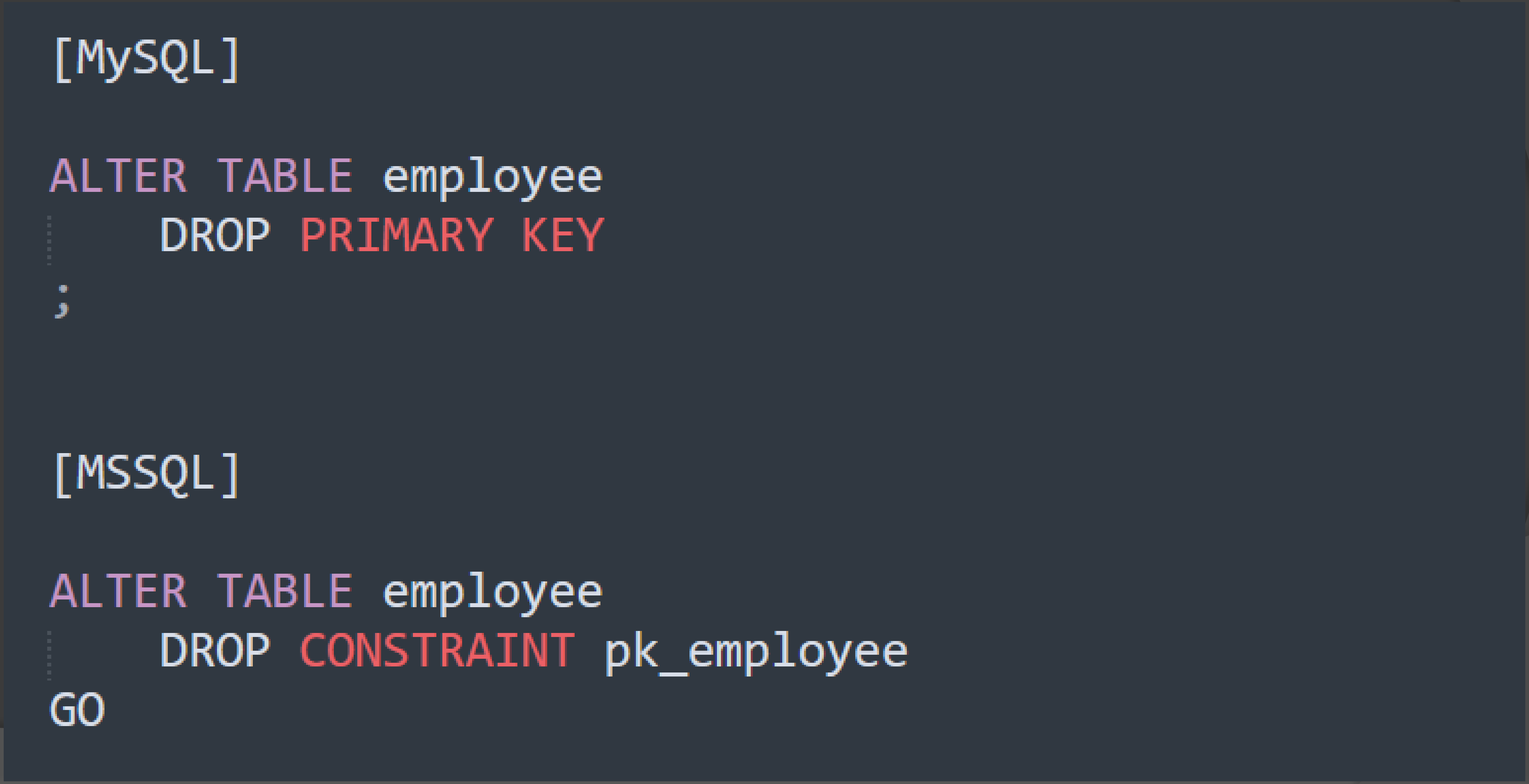
PRIMARY KEY Constraint

On ALTER TABLE



```
[MSSQL]

ALTER TABLE employee
    ADD CONSTRAINT pk_employee PRIMARY KEY (objid)
GO
```

DROP PRIMARY KEY Constraint



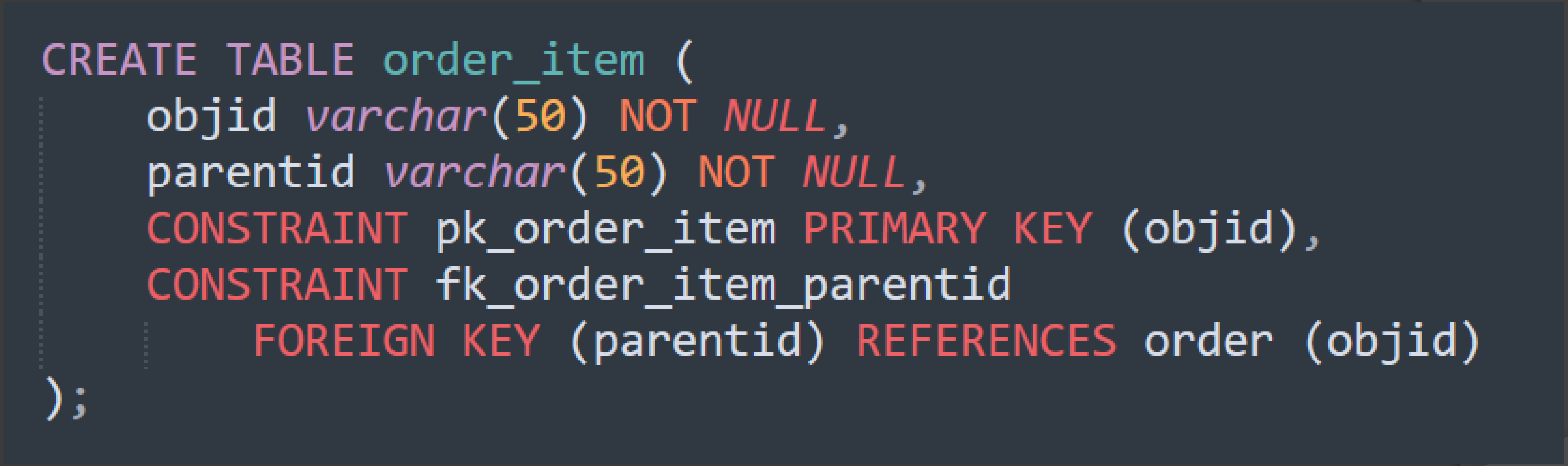
```
[MySQL]

ALTER TABLE employee
    DROP PRIMARY KEY
;



[MSSQL]

ALTER TABLE employee
    DROP CONSTRAINT pk_employee
GO
```

FOREIGN KEY Constraint

- The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables.
- A FOREIGN KEY is a field (or collection of fields) in one table, that refers to the PRIMARY KEY in another table.
- The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table

FOREIGN KEY Constraint

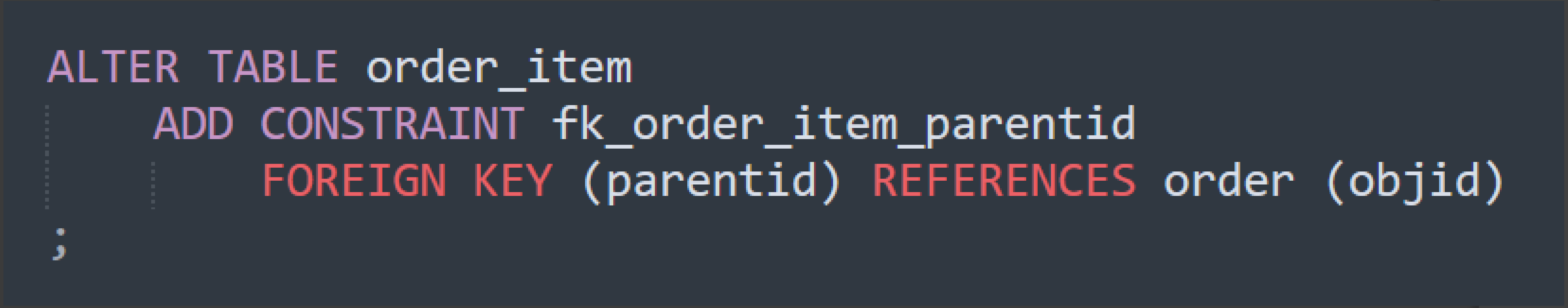
On CREATE TABLE



```sql
CREATE TABLE order_item (
    objid varchar(50) NOT NULL,
    parentid varchar(50) NOT NULL,
    CONSTRAINT pk_order_item PRIMARY KEY (objid),
    CONSTRAINT fk_order_item_parentid
        FOREIGN KEY (parentid) REFERENCES order (objid)
);
```

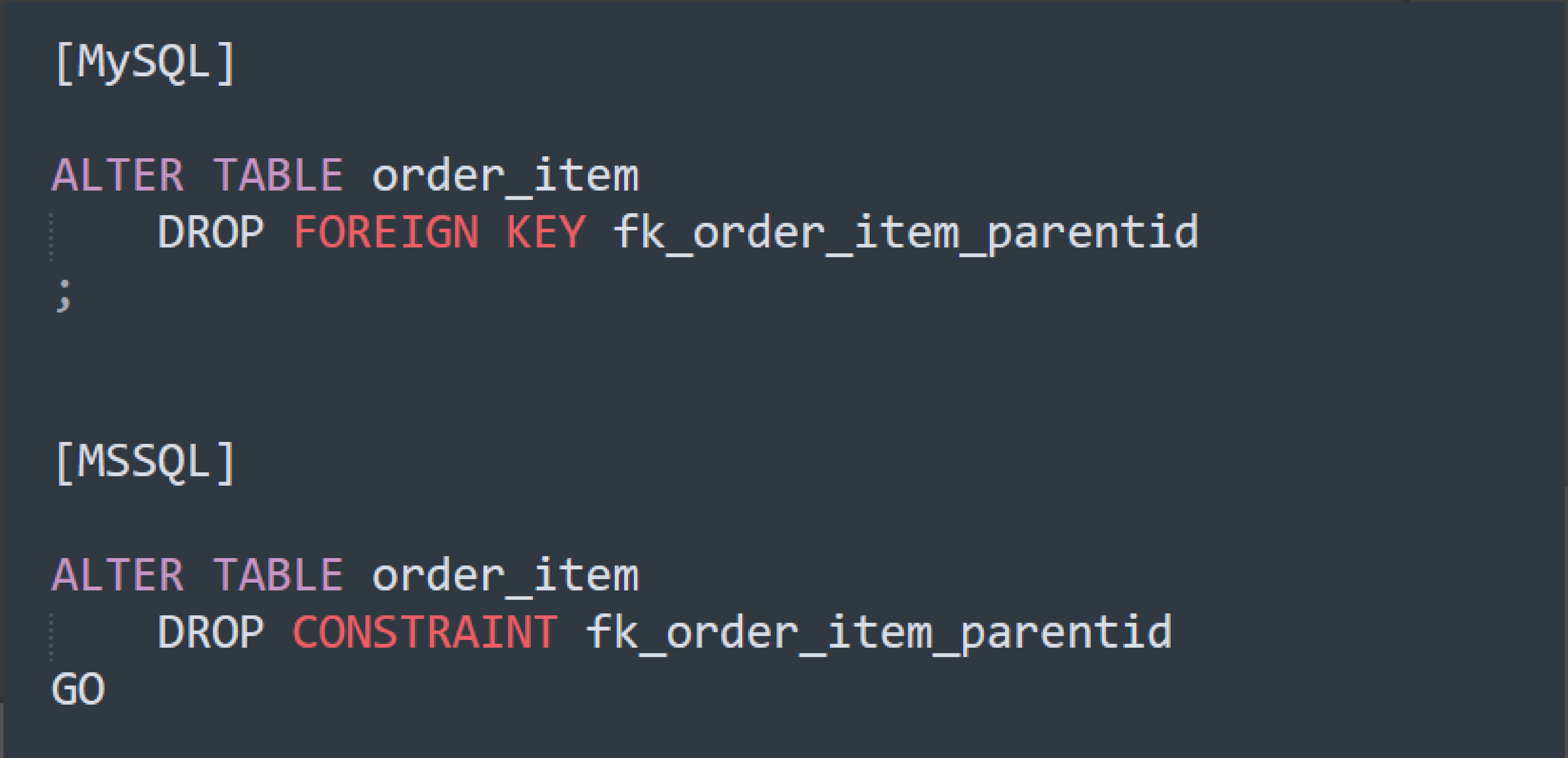
FOREIGN KEY Constraint

On ALTER TABLE



```
ALTER TABLE order_item
    ADD CONSTRAINT fk_order_item_parentid
        FOREIGN KEY (parentid) REFERENCES order (objid)
;
```

DROP FOREIGN KEY Constraint



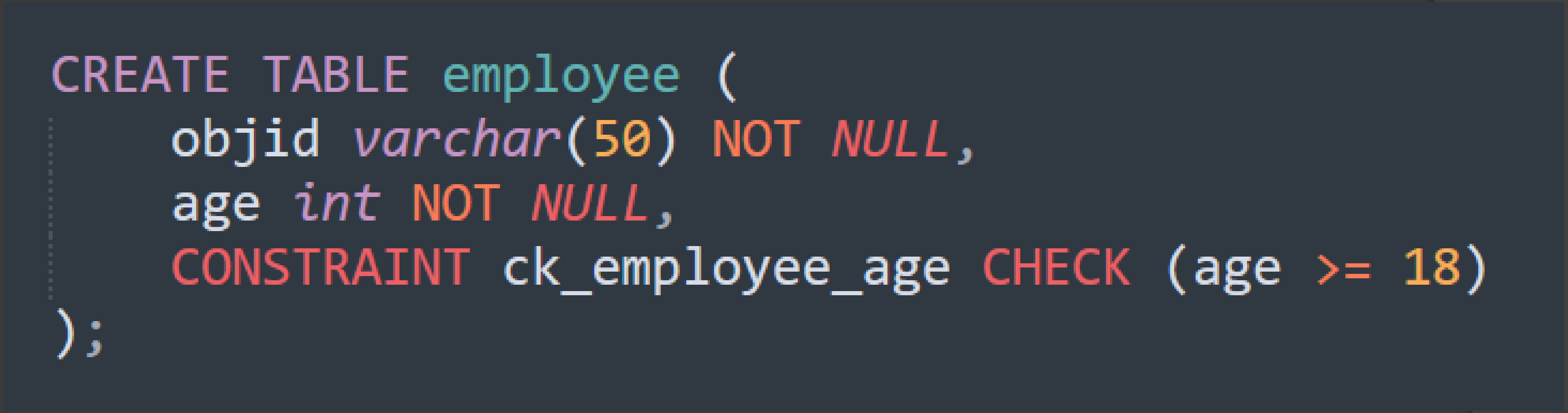
```
[MySQL]

ALTER TABLE order_item
    DROP FOREIGN KEY fk_order_item_parentid
;


[MSSQL]

ALTER TABLE order_item
    DROP CONSTRAINT fk_order_item_parentid
GO
```

CHECK Constraint

- The CHECK constraint is used to limit the value range that can be placed in a column.
- If you define a CHECK constraint on a column it will allow only certain values for this column.
- If you define a CHECK constraint on a table it can limit the values in certain columns based on values in other columns in the row.

CHECK Constraint

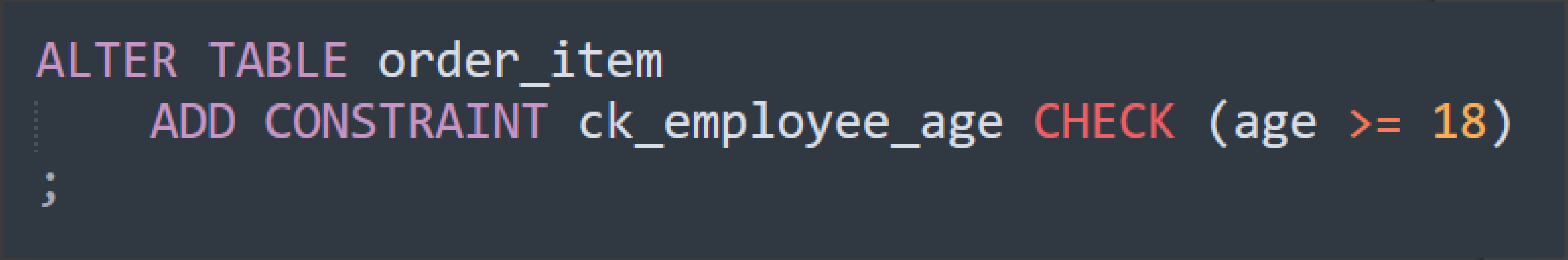
On CREATE TABLE



```sql
CREATE TABLE employee (
    objid varchar(50) NOT NULL,
    age int NOT NULL,
    CONSTRAINT ck_employee_age CHECK (age >= 18)
);
```

CHECK Constraint

On ALTER TABLE



```
ALTER TABLE order_item
    ADD CONSTRAINT ck_employee_age CHECK (age >= 18)
;
```

DROP CHECK Constraint

```
[MySQL]

ALTER TABLE order_item
    DROP CHECK ck_employee_age
;



[MSSQL]

ALTER TABLE order_item
    DROP CONSTRAINT ck_employee_age
;
```

DEFAULT Constraint

- The DEFAULT constraint is used to set a default value for a column.
- The default value will be added to all new records, if no other value is specified.

DEFAULT Constraint

On CREATE TABLE

```
[MySQL]

CREATE TABLE employee (
    objid varchar(50) NOT NULL,
    salary decimal(10,2) NOT NULL DEFAULT '0'
);


[MSSQL]

CREATE TABLE employee (
    objid varchar(50) NOT NULL,
    salary decimal(10,2) NOT NULL DEFAULT ('0')
)
GO
```

DEFAULT Constraint

On ALTER TABLE

```
[MySQL]

ALTER TABLE employee
    ALTER salary SET DEFAULT '0'
;



[MSSQL]

ALTER TABLE employee
    ADD CONSTRAINT df_salary
        DEFAULT '0' FOR salary
GO
```

DROP DEFAULT Constraint

```
[MySQL]

ALTER TABLE employee
    ALTER salary DROP DEFAULT
;



[MSSQL]

ALTER TABLE employee
    DROP CONSTRAINT df_salary
GO
```