

# TECHNOLOGY



**Caltech** | **Center for Technology & Management Education**

**Full Stack Java Developer**

# TECHNOLOGY



## MySQL



## Working with Tables and SQL Commands



# Learning Objectives

By the end of this lesson, you will be able to:

- 👁 Learn to modify and delete a SQL table
- 👁 Comprehend the table components in MySQL
- 👁 Learn all the SQL statements and clauses with their syntaxes
- 👁 Recognize the SQL keys and classify them
- 👁 Understand how to write and use the SQL keys



# Learning Objectives

By the end of this lesson, you will be able to:

- 🕒 Learn how to update and delete databases in SQL
- 🕒 Describe SQL commands and table manipulation
- 🕒 Create full functional tables in the database
- 🕒 Learn Join and its types with syntaxes
- 🕒 Understand the significance of DELETE and UPDATE keywords





# A Day in the Life of a Full Stack Developer

You are hired as a full-stack developer in an organization and have been assigned to an application development project that collects data from the candidate along with their resumes and suggests an appropriate job opening for them.

You being a developer of the application, have been storing all the incoming data in the database and whenever the user searches for the details of the applied jobs, fetch the data from the database and display it to the user.

However, the users also want to add, remove, and modify their requirements whenever needed. Since the application collects multiple data and stores it in different tables, fetching data from multiple tables can be challenging for you. However, you figure out that this can be done using the Join statement.

To accomplish the task, you explore more about inserting, updating, and deleting the data from tables and using the Join statements.



## Modifying and Deleting a SQL Table

# UPDATE

The UPDATE statement is used to modify the existing records in the table.



Syntax:

```
UPDATE name_of_table  
SET column1 = value1, column2 =  
value2, ...  
WHERE condition;
```





# DELETE

The DELETE statement is used to delete a SQL table.



Syntax:

```
DELETE FROM name_of_table WHERE condition;
```

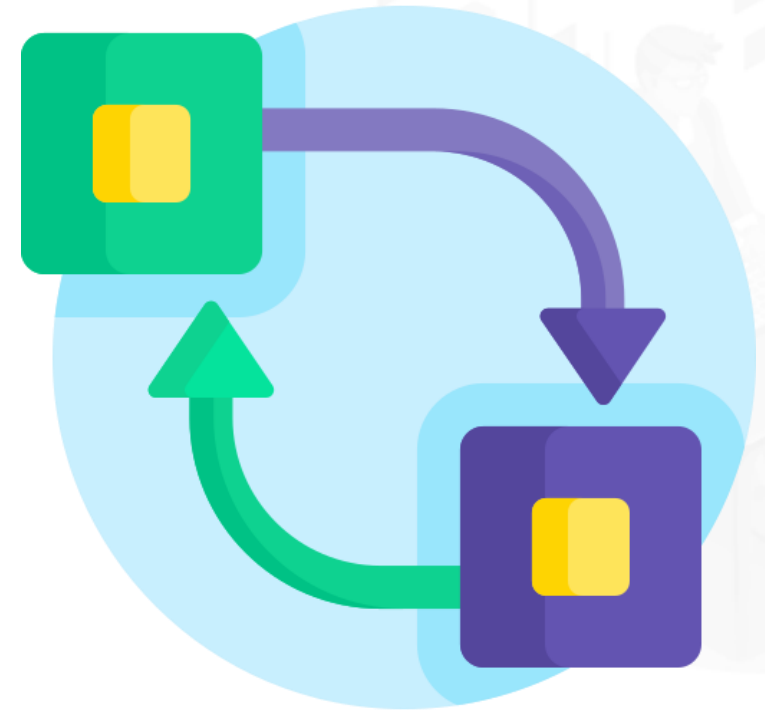


# ALTER

The ALTER statement is used for adding, deleting, or modifying columns in an existing table.

Syntax:

```
ALTER TABLE  name_of_table ADD column_name  
datatype;
```



# DROP TABLE

The DROP TABLE is used to delete an existing table in a database.



Syntax:

```
DROP TABLE  name_of_table
```





## Modifying and Deleting a SQL Database

# Modifying a SQL Database

ALTER is used to change the features of a database.

ALTER privilege is required by the ALTER statement on the database.

Syntax:

```
ALTER {DATABASE | SCHEMA} database_name
    alter_option ...
alter_option: {
    [DEFAULT] CHARACTER SET [=] charset_name
  | [DEFAULT] COLLATE [=] collation_name
  | [DEFAULT] ENCRYPTION [=] {'Y' | 'N'}
  | READ ONLY [=] {DEFAULT | 0 | 1}
}
```



# Deleting a Database

DROP DATABASE statement is used to delete an existing database.



Syntax:

```
DROP DATABASE database_name;
```





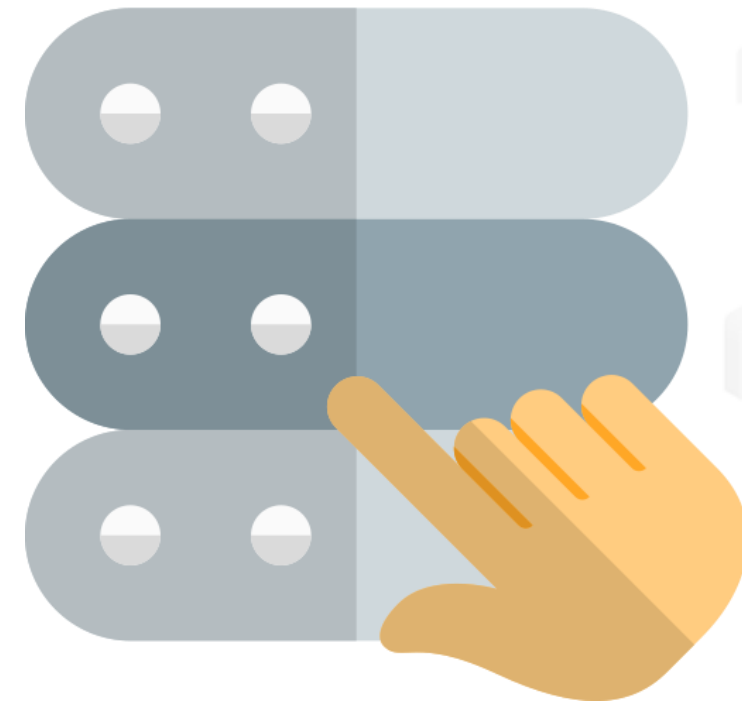
## SQL Commands

# MySQL Select

It is used for selecting data from a database.

Syntax:

```
SELECT column1, column2,  
...  
FROM name_of_table;
```



The returned data is stored in a result table.

# MySQL Where Clause

This is used to filter out the results.

Syntax:

```
SELECT field1, field2,...fieldN name_of_table1,  
name_of_table2...  
[WHERE condition1 [AND [OR]] condition2.....
```

It allows specifying selection criteria to select the required records.





# MySQL AND/OR Operator

It is used for selecting data from a database.

The returned data is stored in a result table.

Syntax:

```
A AND B
```

	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

# MySQL AND/OR Operator

It is used for selecting data from a database.

Syntax:

```
A or B
```

	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

# MySQL OrderBy Keyword

It is used to sort the outcome in ascending or descending order.

Syntax:

```
SELECT column1, column2, ...  
FROM name_of_table  
ORDER BY column1, column2, ...  
ASC|DESC;
```





# MySQL GroupBy

It is used for collecting data from multiple records and grouping the output.

This is mostly used in SELECT statements along with SUM, COUNT, MAX, etc.

Syntax:

```
SELECT expression_a, expression_b, ...  
expression_n,  
aggregate_function (expression)  
FROM tables  
[WHERE conditions]  
GROUP BY expression_a, expression_b, ...  
expression_n;
```



# MySQL Having

It is used by the GROUP clause. It returns the rows where the condition is TRUE.

Syntax:

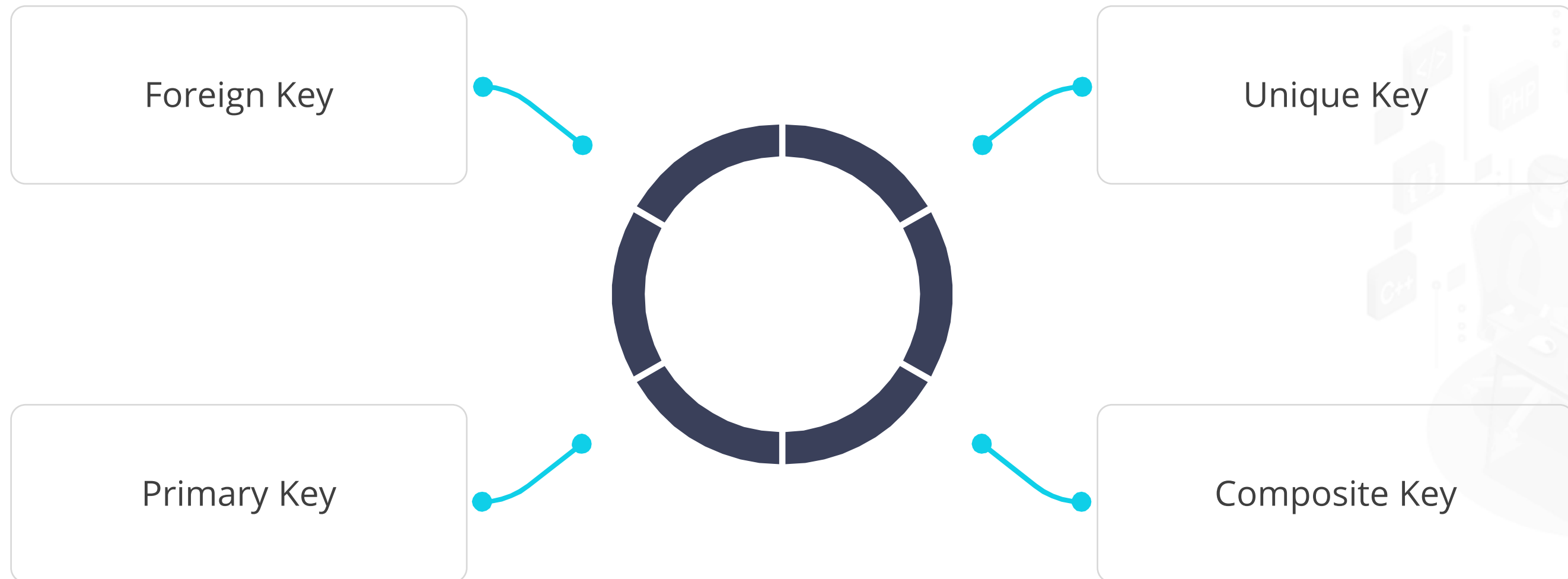
```
SELECT expression_a, expression_b, ...  
expression_n,  
aggregate_function (expression)  
FROM tables  
[WHERE conditions]  
GROUP BY expression_a, expression_b, ...  
expression_n  
HAVING condition;
```



## MySQL Keys

# MySQL Keys

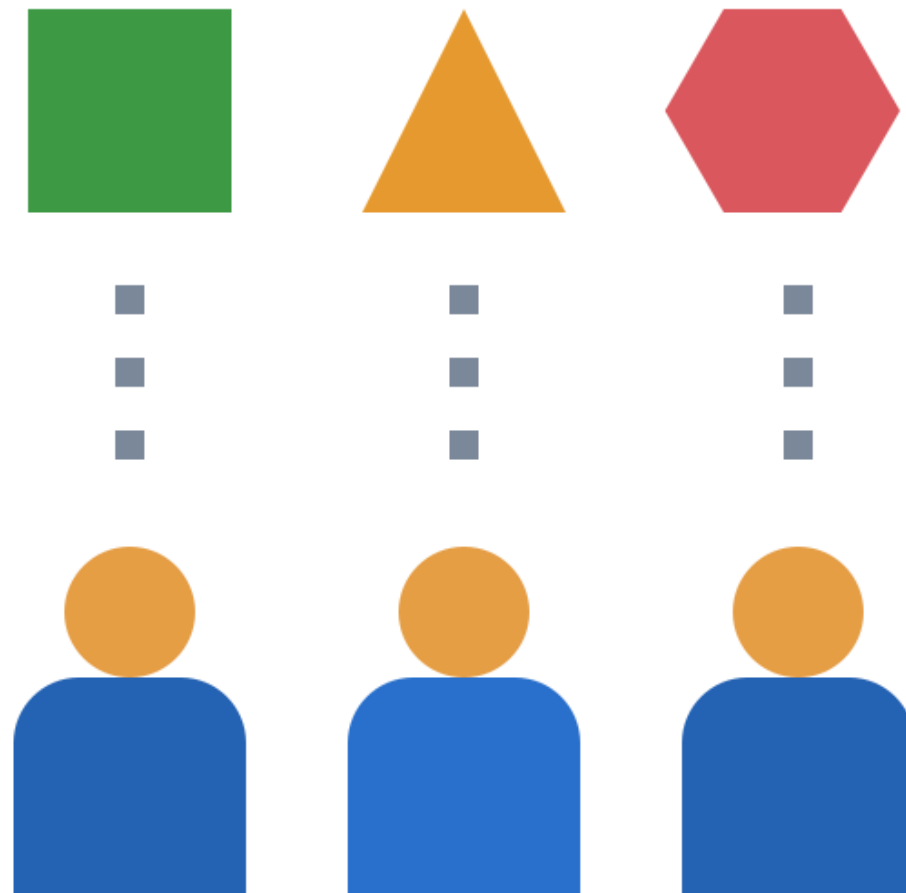
MySQL offers four keys:



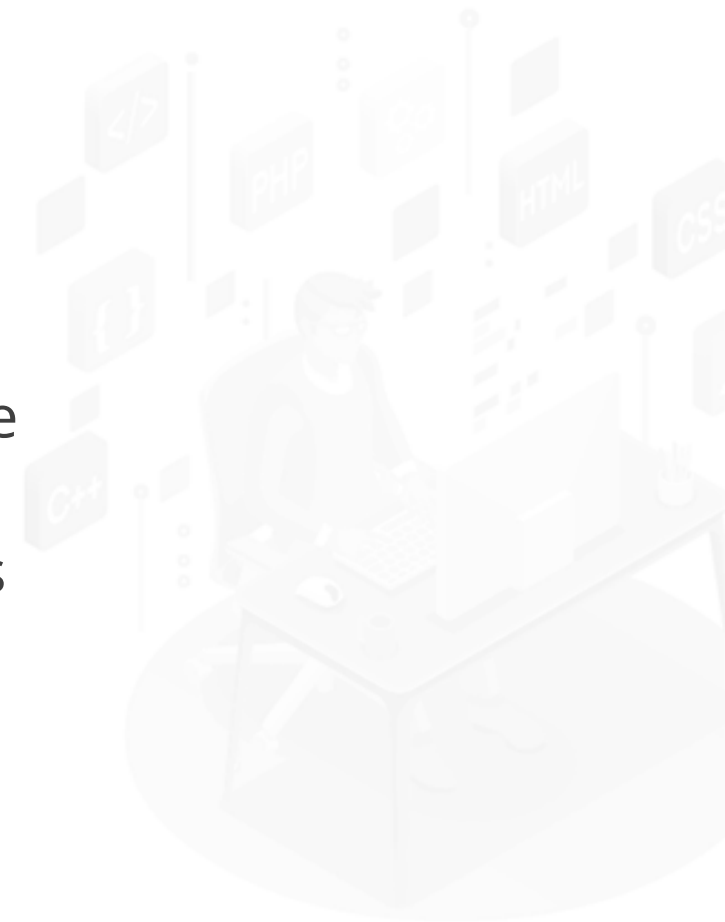
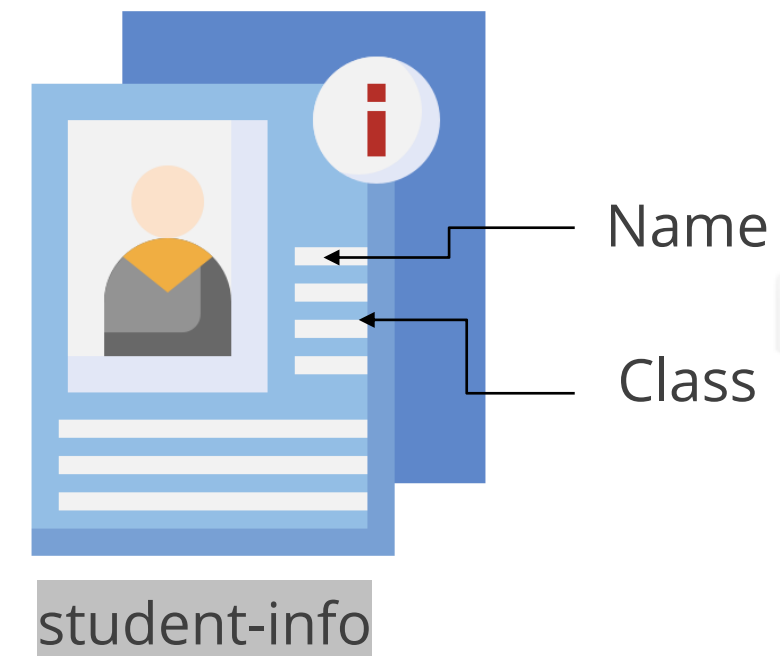


# Unique Key

In MySQL, there is a single field or combination of fields that make sure all values that are going to be stored in the column are unique, which means the column cannot store the same values.

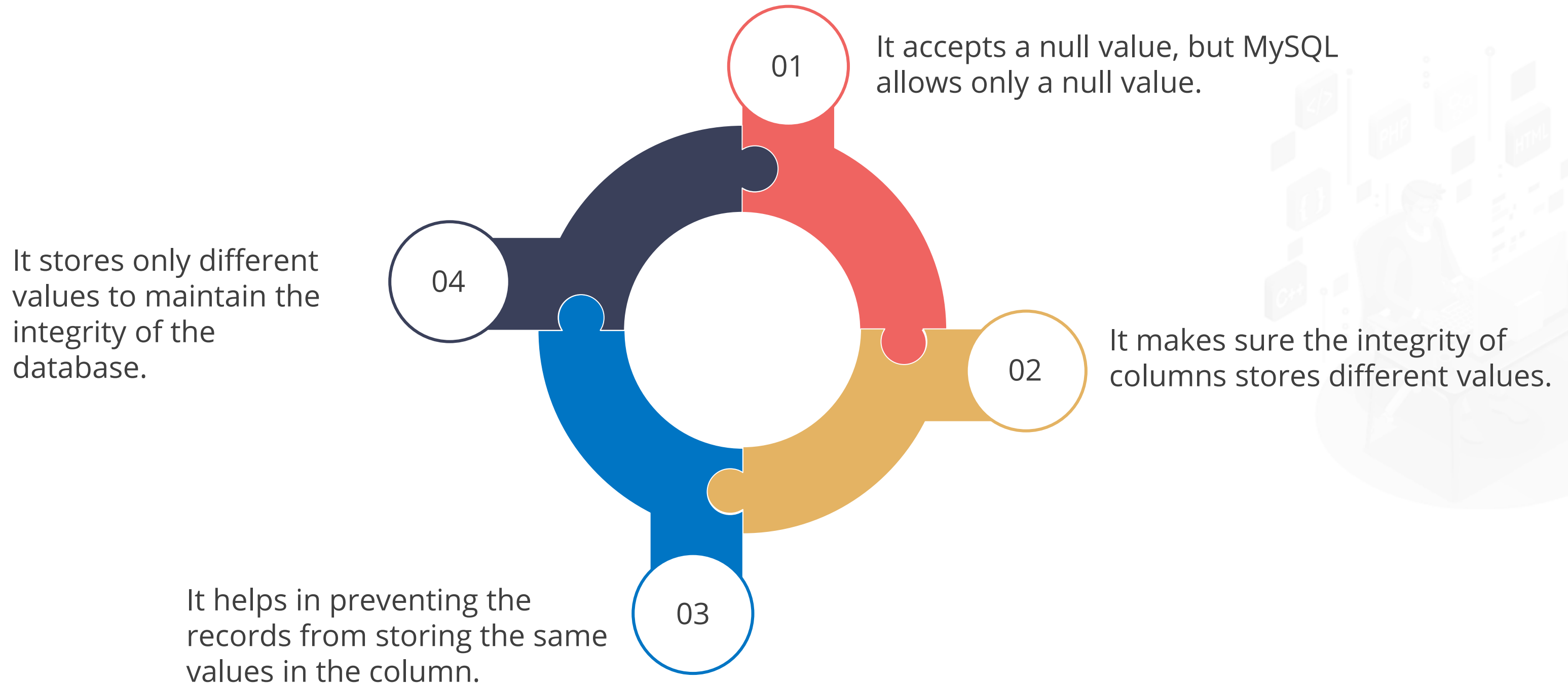


Example:



# Unique Key

MySQL allows to use more than one column with UNIQUE values in the table.



# Unique Key

The syntax to create a unique key in the table is:

```
CREATE TABLE table_name(  
    col1 data_type,  
    col2 data_type UNIQUE,  
    ...  
);
```

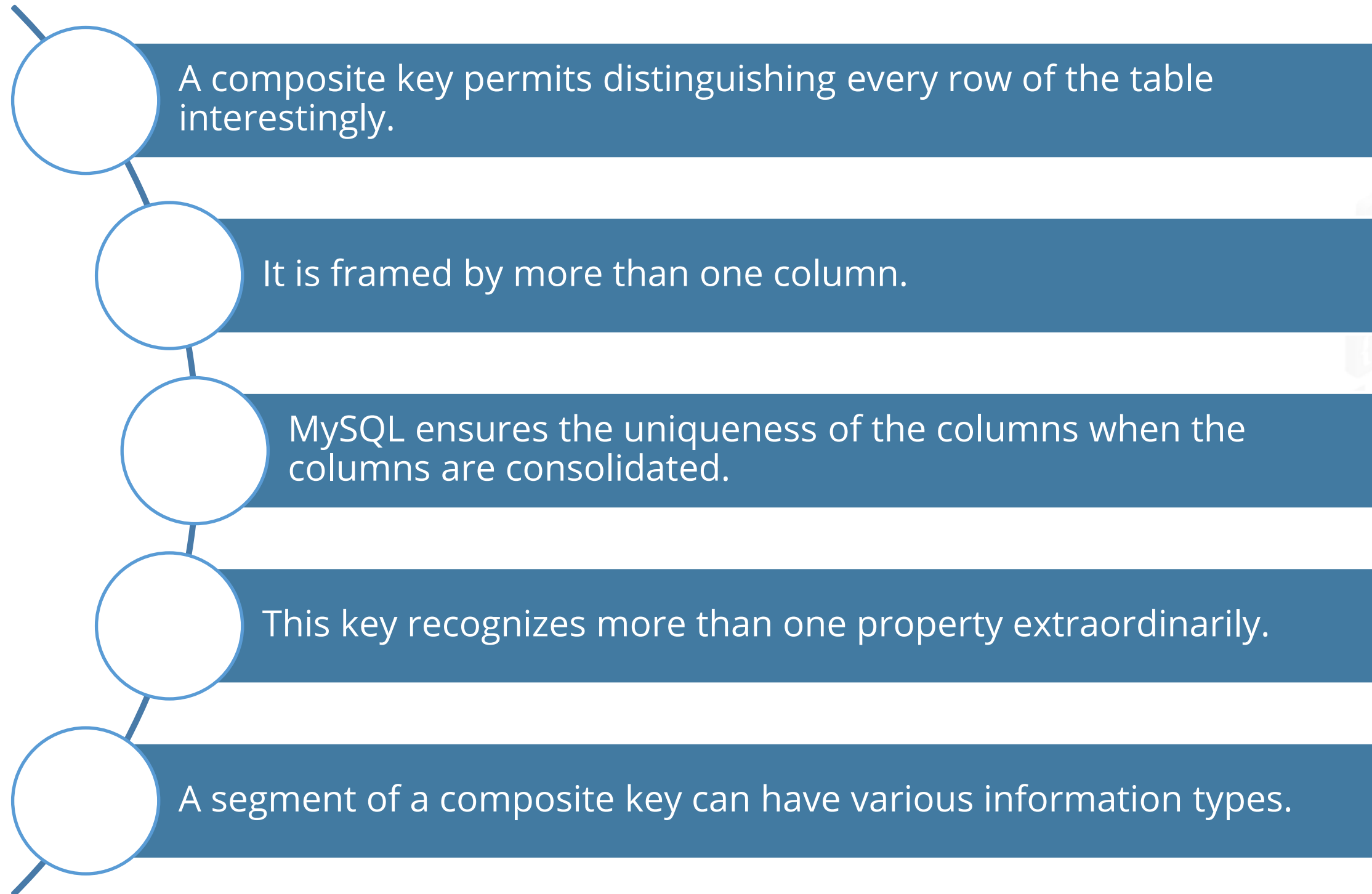
The syntax to insert multiple unique keys into a table is:

```
CREATE TABLE table_name(  
    col1 column_definition,  
    col2 column_definition,  
    ...  
    [CONSTRAINT constraint_name]  
    UNIQUE(col_name(s))  
);
```

If a unique constraint is not specified, SQL gives a name for that column.



# Composite Keys



# Composite Keys

The composite key can be added in two different ways:



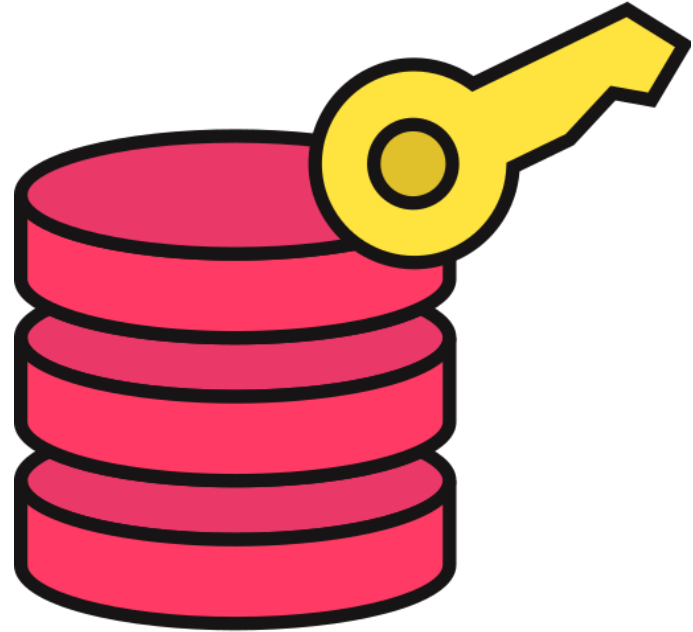


# Primary Keys

It helps to uniquely recognize each record in a table.

It contains unique values and not NULL values.

Syntax:



```
CREATE TABLE Persons (  
  ID int NOT NULL,  
  LastName varchar(255) NOT NULL,  
  FirstName varchar(255),  
  Age int,  
  CONSTRAINT PK_Person PRIMARY KEY  
  (ID, LastName)  
);
```

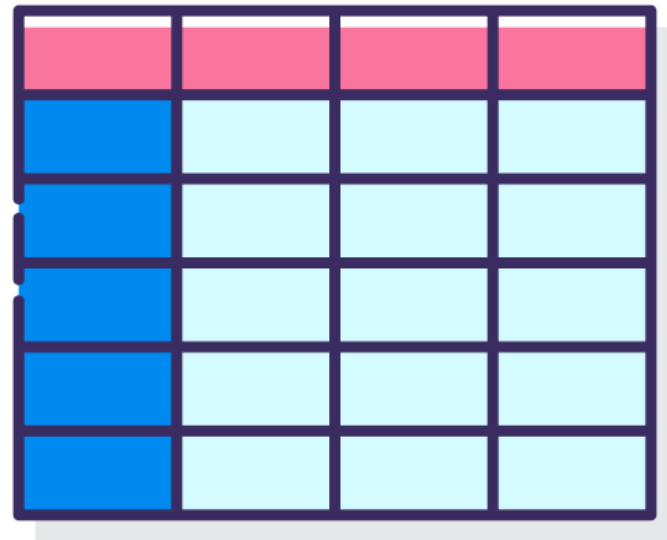


# Foreign Keys

It is used to prevent actions that would impact connections between tables.

**Foreign Key**

Child Table







**Parent Key**

Referenced or  
Parent Table



# Foreign Keys

The syntax to create a Foreign Key on the "ProductId" column when the "Orders" table is created is:

```
CREATE TABLE Orders (  
  OrderID int NOT NULL,  
  OrderNumber int NOT NULL,  
  ProductID int,  
    PRIMARY KEY (OrderID),  
    FOREIGN KEY (ProductId) REFERENCES  
  Persons (ProductId)  
);
```



# MySQL Distinct Keyword

The SELECT DISTINCT statement is used to return different values.



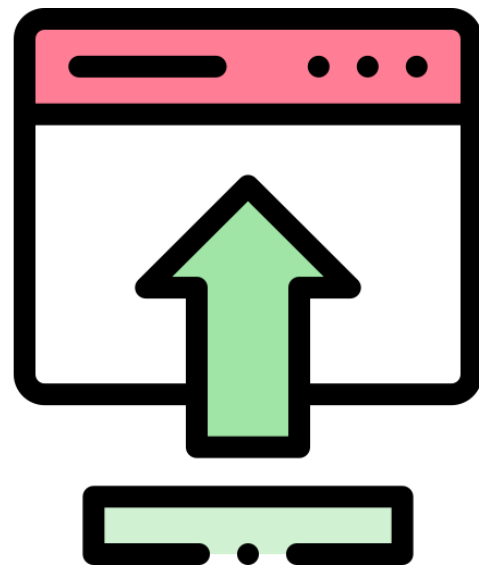
Syntax:

```
SELECT DISTINCT column1, column2, ...  
FROM name_of_table;
```



# MySQL Insert Into Keyword

The MySQL Insert statement is used to insert single or multiple records into a table.



Syntax:

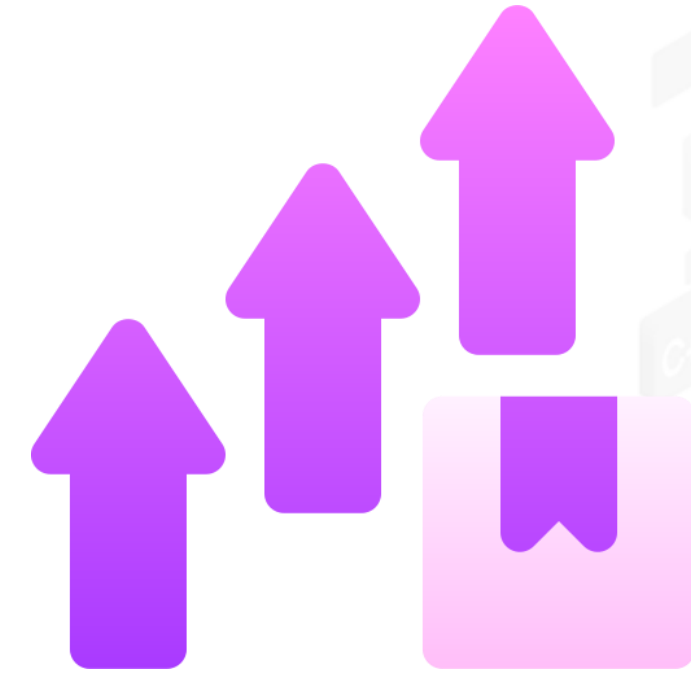
```
INSERT INTO table  
(column1, column2, ... )  
VALUES  
(expression1, expression2, ... ),  
(expression1, expression2, ... ),  
...;
```

# MySQL Get Last Inserted ID

The LAST\_INSERT\_ID() function returns the AUTO\_INCREMENT ID of the last row added or updated in a table.

Syntax:

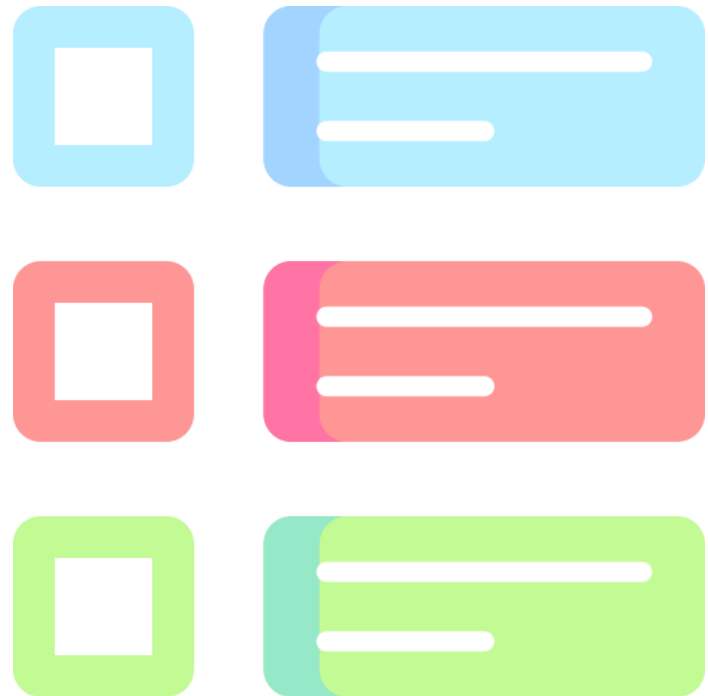
```
LAST_INSERT_ID(expression)
```





# MySQL Insert Multiple Records Command

It is used to insert multiple rows into a table.



Syntax:

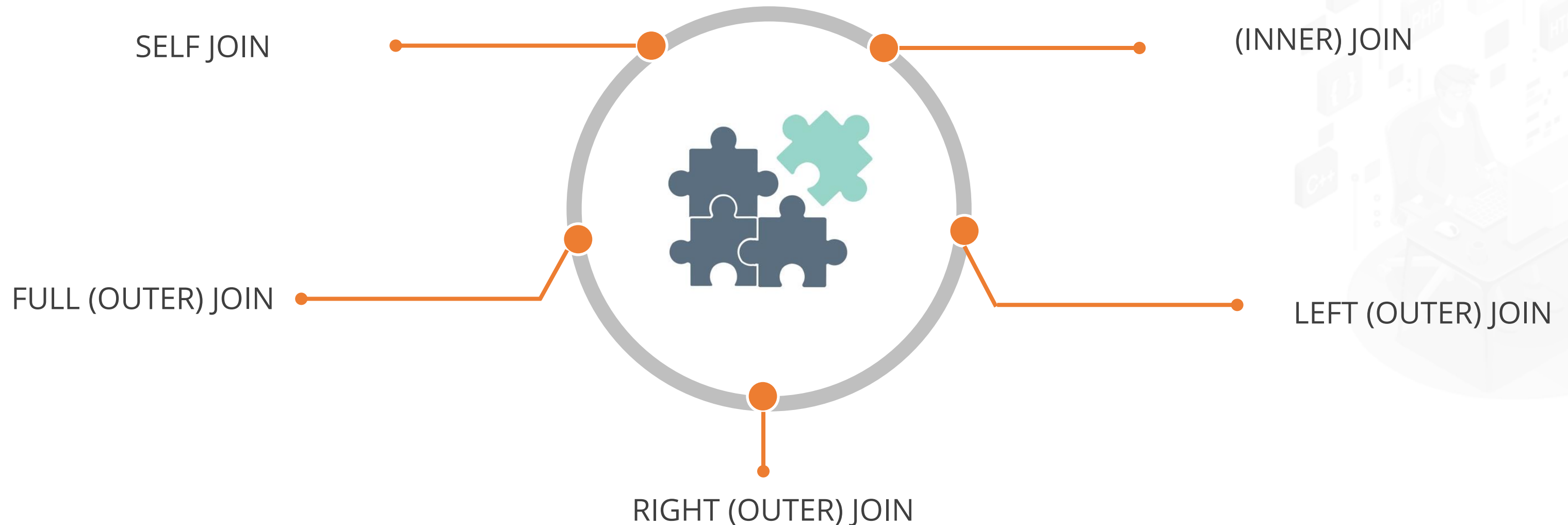
```
LAST_INSERT_ID(exINSERT INTO name_of_table  
  (column_list)  
VALUES  
  (value_list_1),  
  (value_list_2),  
  ...  
  (value_list_n);  
pression)
```

## MySQL Joins

# MySQL Joins

A Join statement is used to merge the rows from two or more tables.

Different types of JOINS are:



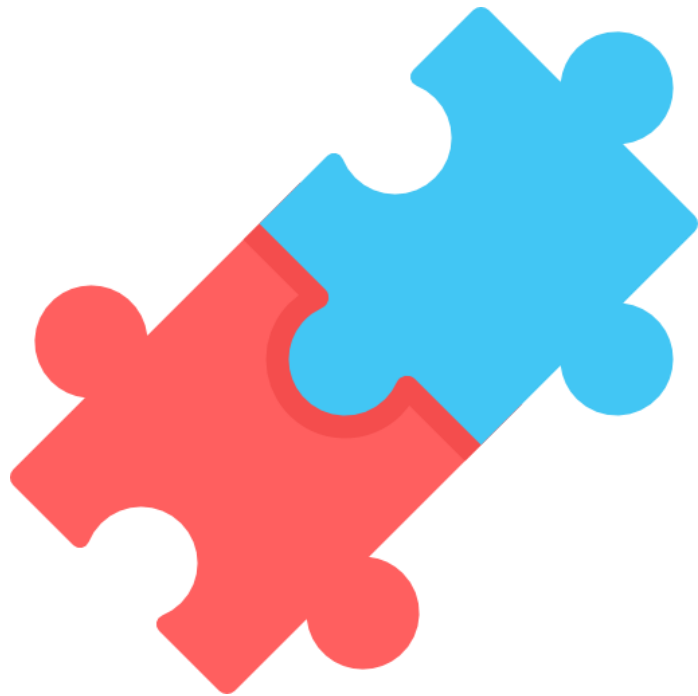
# Inner Join

It returns records that have the same values in both tables.

The keyword takes the records with matching values in both tables.

Syntax:

```
SELECT column_name_(s)
FROM table1
INNER JOIN table2
ON table1.column_name =
table2.column_name;
```



# Left Join

It returns the same records from the right table and all records from the left table.

It shows 0 records from the right side if no match is found.

Syntax:



```
SELECT column_name_(s)
FROM table1
LEFT JOIN table2
ON table1.column_name =
table2.column_name;
```

# Right Join

It returns the same records from the left table and all records from the right table.

It shows 0 records from the left side if no match is found.

Syntax:

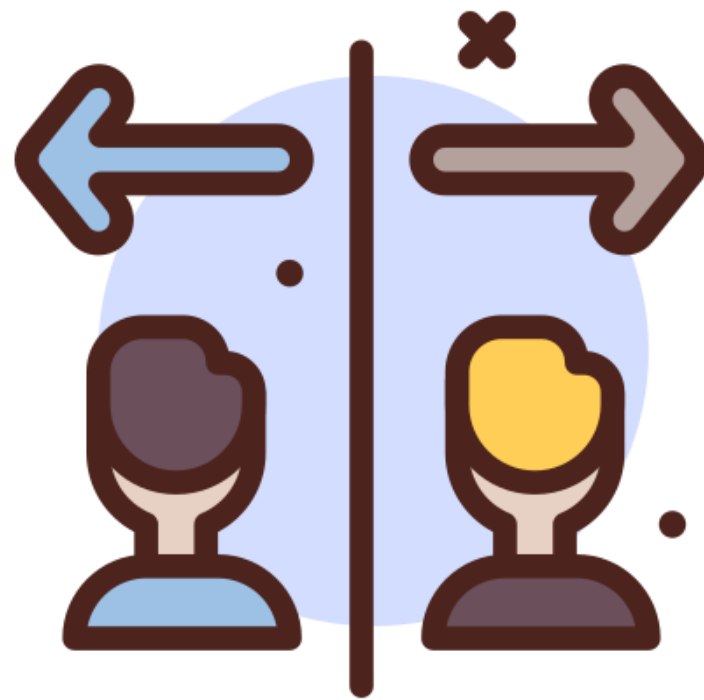
```
SELECT column_name_(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name =
table2.column_name;
```





# Full Join

It returns all records when the value is the same, either on the right or left side.



Syntax:

```
SELECT column_name_(s)
FROM table1
FULL OUTER JOIN table2
ON table1.column_name = table2.column_name
WHERE condition;
```

# Self Join

It is a regular join where the table is joined with itself.

Syntax:

```
SELECT column_name_(s)
FROM table1 T1, table1 T2
WHERE condition;
```



## MySQL Delete and Update Records

# DELETE Keyword

It is used to delete the current records from a table.

Syntax:

```
DELETE FROM table_name WHERE  
[condition];
```



# UPDATE Keyword

It is used to update the current records from a table.

It helps to modify the values of one or multiple columns or rows.



Syntax:

```
UPDATE [LOW_PRIORITY] [IGNORE]
table_name
SET
    column_name1 = expr1,
    column_name2 = expr2,
    ...
[WHERE condition];
```

## Key Takeaways

- The UPDATE statement is used to modify the existing records in the table.
- ALTER is used to change the features of a database.
- A composite key permits you to distinguish every row of the table interestingly.
- The MySQL Insert statement is used to insert single or multiple records into a table.
- A Join statement is used to merge the rows from two or more tables.
- The DELETE keyword is used to delete the current records from a table.





# TECHNOLOGY

**Thank You**