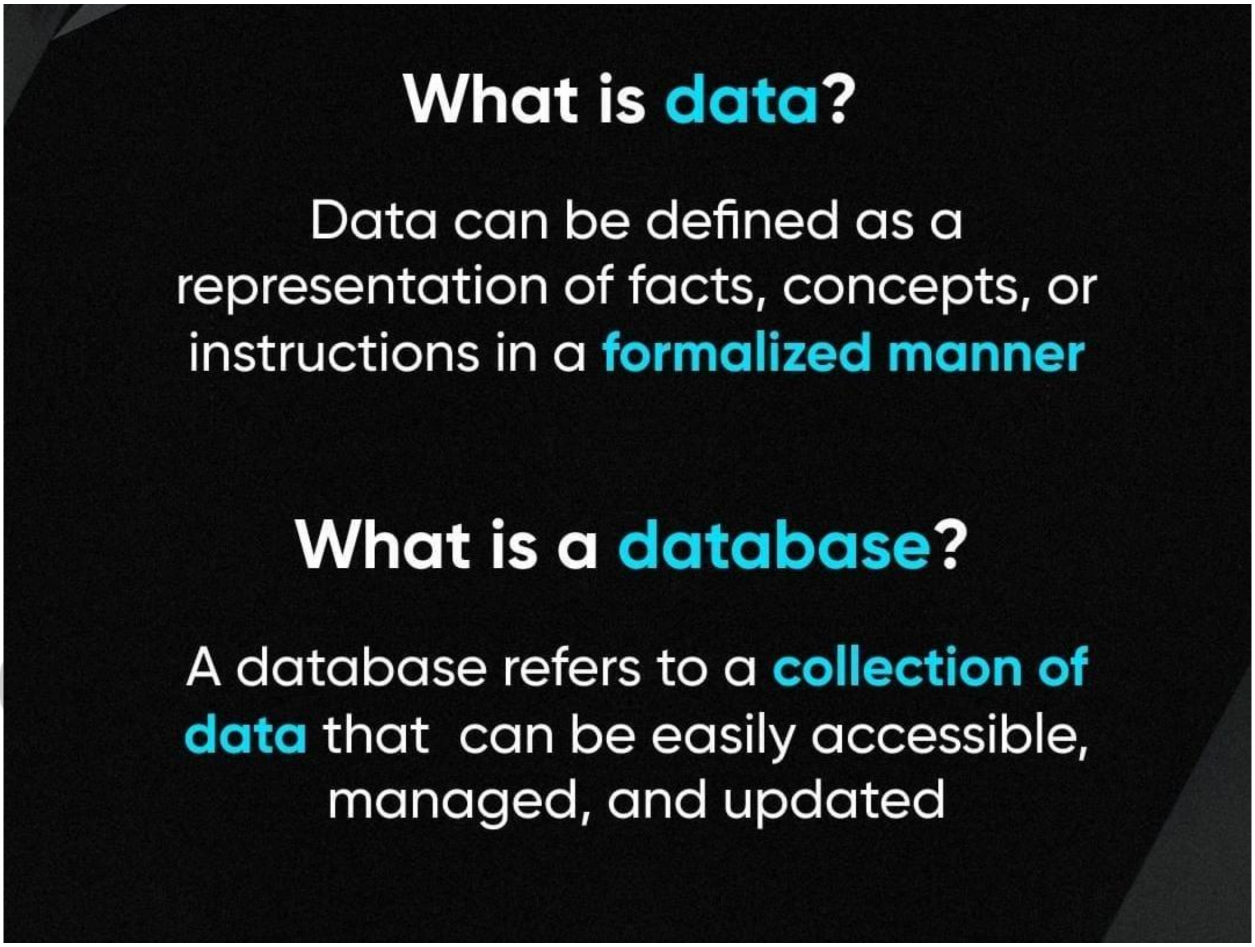
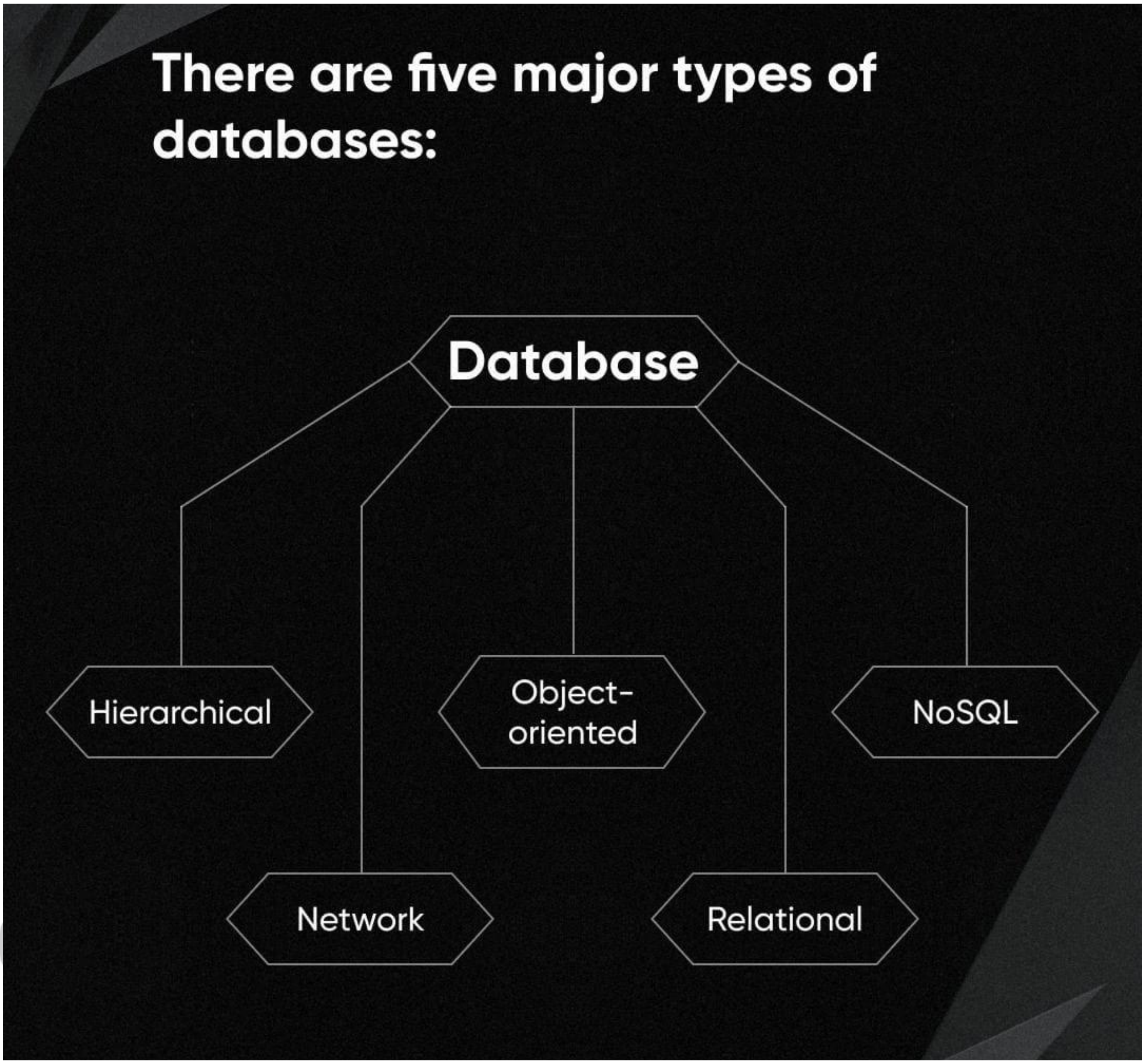


What is data?

Data can be defined as a representation of facts, concepts, or instructions in a **formalized manner**

What is a database?

A database refers to a **collection of data** that can be easily accessible, managed, and updated



There are five major types of databases:

Database

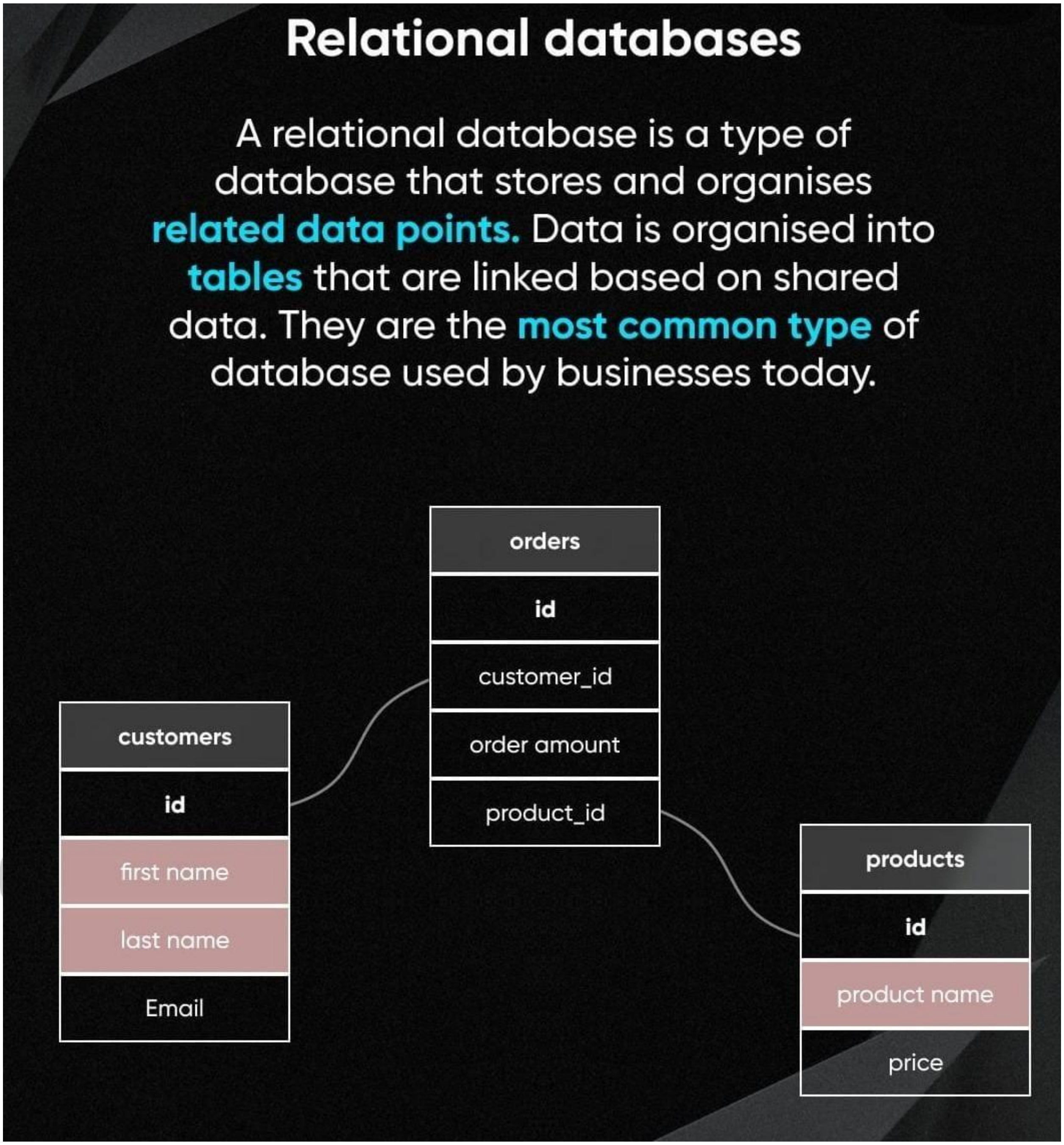
Hierarchical

Network

Object-oriented

Relational

NoSQL



Relational databases

A relational database is a type of database that stores and organises **related data points.** Data is organised into **tables** that are linked based on shared data. They are the **most common type** of database used by businesses today.

orders
id
customer_id
order amount
product_id

customers
id
first name
last name
Email

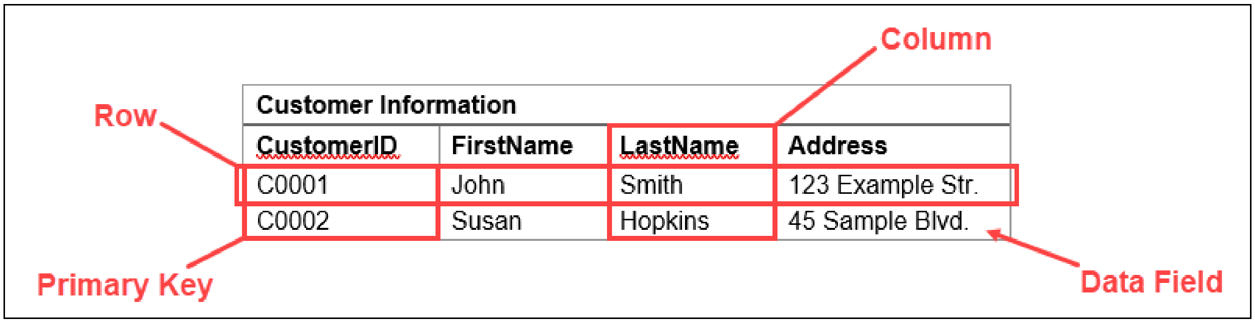
products
id
product name
price

What is a Database Management System (DBMS)?

Database Management System (DBMS) is a collection of programs that enable its users to access databases, manipulate data, report, and represent data. It also helps to control access to the database.

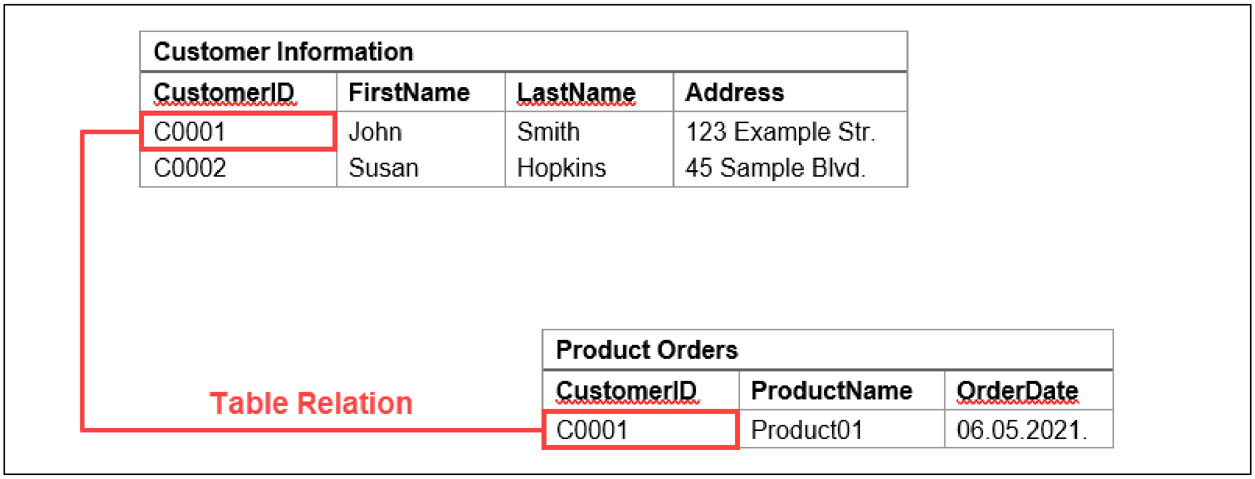
Relational Database Examples

1. **MySQL**
2. Oracle
3. **PostgreSQL**
4. **MariaDB**



Customer Information

CustomerID	FirstName	LastName	Address
C0001	John	Smith	123 Example Str.
C0002	Susan	Hopkins	45 Sample Blvd.



Customer Information

CustomerID	FirstName	LastName	Address
C0001	John	Smith	123 Example Str.
C0002	Susan	Hopkins	45 Sample Blvd.

Product Orders

CustomerID	ProductName	OrderDate
C0001	Product01	06.05.2021.

MySQL/Oracle:

1. **CREATE DATABASE employeesdb;**

2. **SHOW DATABASES**

3. **USE emplyeedb;**

4. **DROP DATABASE mytestdb_copy;**

Tables:

```
CREATE TABLE People(
   id int NOT NULL AUTO_INCREMENT,
   name varchar(45) NOT NULL,
   occupation varchar(35) NOT NULL,
   age int,
   PRIMARY KEY (id)
);
```

1. If we want to store single records for all fields, use the syntax as follows:

```
INSERT INTO People (id, name, occupation, age)
VALUES (101, 'Peter', 'Engineer', 32);
```

2. If we want to store multiple records, use the following statements where we can either specify all field names or don't specify any field.

```
INSERT INTO People VALUES
(102, 'Joseph', 'Developer', 30),
(103, 'Mike', 'Leader', 28),
(104, 'Stephen', 'Scientist', 45);
```

Exponent
IT TRAINING & SERVICES

3. If we want to store records without giving all fields, we use the following partial field statements. In such case, it is mandatory to specify field names.

```
INSERT INTO People (name, occupation)
VALUES ('Stephen', 'Scientist'), ('Bob', 'Actor');
```

```
SELECT * FROM People;
```

```
DELETE FROM People WHERE  id=101;
```

Alter Table

Exponent
IT TRAINING & SERVICES

Day-2:

The WHERE clause is used to filter records.

It is used to extract only those records that fulfill a specified condition.

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

WHERE clause is not only used in SELECT statements, it is also used in UPDATE, DELETE, etc.

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

```
UPDATE employee
SET name = 'ABC XYZ', City = 'Pune'
WHERE empID = 1;
```

```
DELETE FROM Employee WHERE name='ABC';
```

Operators in The WHERE Clause
The following operators can be used in the WHERE clause:

Operator	Description	Example
=	Equal	
>	Greater than	
<	Less than	
>=	Greater than or equal	
<=	Less than or equal	
<>	Not equal. Note: In some versions of SQL this operator may be written as !=	

Exponent
IT TRAINING & SERVICES

BETWEEN	**Between a certain range**
LIKE	**Search for a pattern**
IN	**To specify multiple possible values for a column**

How to Alter Table Details :

```
ALTER TABLE student
ADD Email varchar(255);

ALTER TABLE student
DROP COLUMN Email;

Create table EmployeeTemp (
        ID int,
        LastName varchar(255));


DROP TABLE EmployeeTemp ;
```

Exponent
IT TRAINING & SERVICES

MySQL Constraints:

- Used to specify rules for the data in a table

- **NOT NULL - Ensures that a column cannot have a NULL value**

  ALTER TABLE <Table_Name> ALTER COLUMN <columnName> <data_type> NOT NULL;

  Note- IN some version ALTER/MODIFY will work

E.g.

```
    ALTER TABLE Student MODIFY COLUMN Age int NOT NULL;
```

- **UNIQUE - Ensures that all values in a column are different**

E.g.

```
    CREATE TABLE Student ( ID int NOT NULL,LastName
varchar(255) NOT NULL,FirstName varchar(255),Age int,  UNIQUE
(ID));
```

- **PRIMARY KEY -**
- **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).**

Example :

```
CREATE TABLE Employee (

    ID int NOT NULL,

    LastName varchar(255) NOT NULL,

    FirstName varchar(255),

    Age int,

   PRIMARY KEY (ID)

);



CREATE TABLE Employee (
    ID int NOT NULL,

    LastName varchar(255) NOT NULL,

    FirstName varchar(255),

    Age int,

   CONSTRAINT PK_Employee PRIMARY KEY (ID,LastName)

);

ALTER TABLE Employee ADD PRIMARY KEY (ID);
```

Note: If you use ALTER TABLE to add a primary key, the primary key column(s) must have been declared to not contain NULL values (when the table was first created).

- **FOREIGN KEY -**
- **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.**

```
CREATE TABLE Orders (

     OrderID int NOT NULL,

     OrderNumber int NOT NULL,

     PersonID int,

    PRIMARY KEY (OrderID),

    CONSTRAINT FK_Employee FOREIGN KEY (ID)
```

```
        REFERENCES Employee(ID)
    );
```

Key Points :

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.

AUTO INCREMENT -

- **Auto-increment allows a unique number to be generated automatically when a new record is inserted into a table.**
- **Often this is the primary key field that we would like to be created automatically every time a new record is inserted.**

```
CREATE TABLE Employee (
    Empid int NOT NULL AUTO_INCREMENT,
     LastName varchar(255) NOT NULL,
     FirstName varchar(255),
     Age int,
    PRIMARY KEY (Empid)
);
```

-By default, the starting value for AUTO_INCREMENT is 1, and it will increment by 1 for each new record.

Exponent IT TRAINING & SERVICES

ALl Queries of Day -2

```
SELECT * FROM student s;

select name from student where id <> 110 AND name <> 'ABC2';

select name from student where id <> 110 OR name = 'ABC2';

select * from student where name like 'A%4';

update student set name='ANC10' where id=103 ;

delete from student where id=103;

ALTER TABLE student ADD Email varchar(255);

update student set email='abc@gmail.com' where id=102 ;

ALTER TABLE student DROP COLUMN Email;

SELECT * FROM EmployeeTemp1 e;

drop table EmployeeTemp1;


create table employee (id int  NOT NULL, name varchar(50) );

select * from employee;
```

Exponent IT TRAINING & SERVICES

```
select * from student;
insert into employee (name) values( 'ABC');

ALTER TABLE employee MODIFY COLUMN name varchar(40) NOT NULL;

insert into employee (id) values (101);

CREATE TABLE Student121 ( ID int NOT NULL,LastName varchar(255)
NOT NULL,FirstName varchar(255),Age int,  UNIQUE (ID));

select * from student121;

insert into student121(id, lastname) values(12, 'ABC1'), (13, 'ABC2');

CREATE TABLE Employee12 (
            ID int NOT NULL,
            LastName varchar(255) NOT NULL,
            FirstName varchar(255),
            Age int,
    PRIMARY KEY (ID)
);


CREATE TABLE Orders (
            OrderID int NOT NULL,
            OrderNumber int NOT NULL,
            PersonID int,
productId int
    PRIMARY KEY (OrderID),
    CONSTRAINT FK_OrdersEmployee FOREIGN KEY (PersonID)
    REFERENCES Employee12(ID),
    foreign key (productId) references Product(id)
);

select * from orders;


SELECT * FROM employee12 e;
```

Exponent
IT TRAINING & SERVICES

```
CREATE TABLE Employee123 (
   Empid int NOT NULL AUTO_INCREMENT,
         LastName varchar(255) NOT NULL,
         Age int,
   PRIMARY KEY (Empid)
);


select * from employee123;

insert into Employee123(lastName, age) values('PQR1', 22);
```

Exponent
IT TRAINING & SERVICES

Day-03

MySQL join clauses

-A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

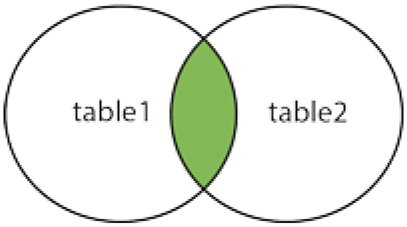
Supported Types of Joins in MySQL

- INNER JOIN: Returns records that have matching values in both tables
- LEFT JOIN: Returns all records from the left table, and the matched records from the right table
- RIGHT JOIN: Returns all records from the right table, and the matched records from the left table
- CROSS JOIN: Returns all records from both tables

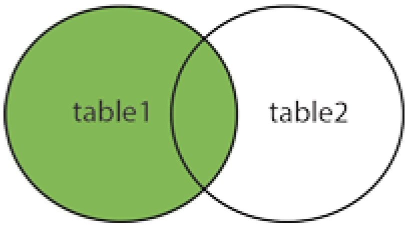
If you add a WHERE/on clause (if table1 and table2 has a relationship), the CROSS JOIN will produce the same result as the INNER JOIN clause:

IT TRAINING & SERVICES

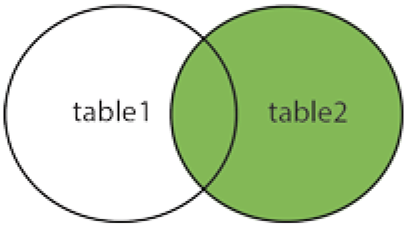
INNER JOIN



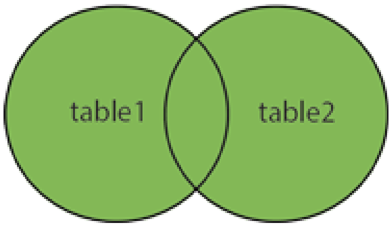
LEFT JOIN



RIGHT JOIN



CROSSJOIN



```
create database JoinsTest;

use joinsTest;

create table course (id int auto_Increment, name varchar(20), primary key
(id));

select * from course;

insert into course(name) values('java'), ('cpp');

insert into course(name) values('Python'), ('JAVASCRIPT');

drop table student;

create table student(id int auto_increment, name varchar(20), cid int, primary
key(id),constraint fk_stu foreign key(cid) references course(id) );

select * from student;

insert into student(name, cid) values ('ABC1', 1), ('ABC2', 2);

insert into student(name) values ('ABC3'), ('ABC4'), ('ABC5');

update student set cid=null where id=3;


select c.name, s.name from course c INNER join student s on c.id=s.cid;

select c.name, s.name from course c left join student s on c.id=s.cid;

select c.name, s.name from course c right join student s on c.id=s.cid;

select * from course c cross join student s;

create table studentdetails (id int auto_increment, name varchar(20), marks int,
primary key (id));
```

```
insert into studentdetails (name, marks) values('ABC1', 70),('ABC2',
80),('ABC3',

90);

select AVG(marks) from studentdetails;

select MIN(marks) from studentdetails;

select MAX(marks) from studentdetails;

select SUM(marks) from studentdetails;

select count(marks) from studentdetails;

select distinct(name) from studentdetails;
```

IT TRAINING & SERVICES

Day-04

MySQL ORDER BY:

The ORDER BY keyword is used to sort the result-set in ascending or descending order.

The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.

```
select * from studentdetails order by name desc;

select * from studentdetails order by name asc, marks ASC;
```

IT TRAINING & SERVICES

MySQL GROUP BY Statement:

-Groups rows that have the same values into a select clause.

The GROUP BY statement is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns.

```
select SUM(marks), name from studentDetails group by name;

select MIN(marks), name from studentDetails where marks>50
group by name;

select * from studentdetails where id >5 group by name order by
name ASC/DESC;
```

MySQL UNION Operator

The `UNION` operator is used to combine the result-set of two or more `SELECT` statements.

- Every `SELECT` statement within `UNION` must have the same number of columns
- The columns must also have similar data types
- The columns in every `SELECT` statement must also be in the same order

UNION Syntax

```
SELECT column_name(s) FROM table1

UNION

SELECT column_name(s) FROM table2;

select name from studentdetails union  select name from student;
```

The `UNION` operator selects only distinct values by default. To allow duplicate values, use `UNION ALL`:

```
select name from studentdetails union all select name from student;

select name from studentdetails union  select name from student order by name desc;
select * from studentdetails order by name desc;

select * from studentdetails order by name asc, marks ASC;

insert into studentdetails(name, marks) values('ABC1', 70),('ABC2',
65),('ABC3', 70);

select SUM(marks), name from studentDetails  group by name;

select * from studentdetails where id >5 group by name order by
name ;

select name from studentdetails where id>2 union all select name
from student where id >4 order by name desc;


select c.name, s.name from course c INNER join student s on
c.id=s.cid group by c.name;
```

Exponent
IT TRAINING & SERVICES

```
select * from studentDetails order by marks ;

select name, marks from studentDetails order by marks DESC;

select * from studentDetails order by name DESC, marks DESC;

select sum(marks) from studentdetails;

select * from studentdetails where name= 'ABC3';

select * from studentdetails group by marks ;

 select sum(marks), name from studentdetails group by name ;

select AVG(marks), name from studentdetails group by name ;

select * from studentdetails group by marks order by id;
select sum(marks), name from studentdetails group by name order
by  name;
```

```
select name from studentdetails

 UNION ALL

 select name from student;
```

Exponent
IT TRAINING & SERVICES

Exponent

IT TRAINING & SERVICES