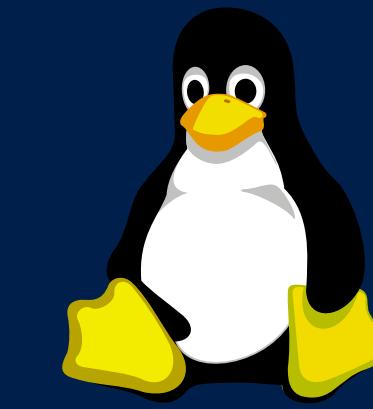




ALRIGHT!

# MYSQL TUTORIAL



**SETUP AND START WORKING ON LINUX**

# Topics

- MySql Installation
- Securing MySQL
- Testing connectivity with MySQL
- Basic necessary queries
- Creating Database
- Creating Table
- Inserting data into the table
- Reading data from Table
- Update data into a table

# Step 1: Installation

```
#yum install mysql-server.x86_64
```

# Start/Enable MySQL

```
#sudo systemctl status mysqld.service
```

```
#sudo systemctl start mysqld.service
```

```
#sudo systemctl enable/disable mysqld.service
```

## Step 2: Changing default options

- Secure the authentication process by setting password for root

```
#sudo mysql_secure_installation
```

# Step 3: Testing Connection

You can verify your installation with the  
mysqladmin tool

```
#mysqladmin -u root -p version
```

To connect and start using SQL Queries

```
#mysql -u root -p
```

# **Some Important Queries in starting**

**SHOW DATABASES;**

**SHOW TABLES;**

**USE <database\_name>**

# Creating a new Database

```
CREATE DATABASE <db_name>;
```

# CRUD

- CREATE
- READ
- UPDATE
- DELETE

<b>id</b>	<b>name</b>	<b>city</b>
101	Raju	Delhi
102	Sham	Mumbai
103	Paul	Chennai
104	Alex	Pune

# Creating a new Table

```
CREATE TABLE students (  
    id INT,  
    name VARCHAR(100)  
)
```

```
CREATE TABLE students (name VARCHAR(100), age INT)
```

# Checking your table

DESC TABLE

# Adding data into a Table

```
INSERT INTO students(id, name)  
VALUES (101, "Rahul")
```

```
INSERT INTO students VALUES (101, "Rahul")
```

# Reading data from a Table

```
SELECT * FROM <table_name>
```

```
SELECT <column_name> from students
```

# Modify/Update data from a Table

```
UPDATE students  
SET contact=12345  
WHERE name='Raju';
```

# **DELETE data from a Table**

```
DELETE FROM students  
WHERE name='Raju' ;
```

# **Deleting a Database or Table**

**DROP DATABASE <db\_name>;**

**DROP TABLE <table\_name>;**

**THANKS FOR WATCHING**

# Databases

# **List All Existing Database**

**SHOW DATABASES;**

# **Creating a new Database**

**CREATE DATABASE <db\_name>;**

# Working with a Database

**USE <db\_name>;**

# **Deleting a Database**

**DROP DATABASE <db\_name>;**

# Tables

# Table

A table is a collection of related data held in a table format within a database.

<b>id</b>	<b>name</b>	<b>city</b>
101	Raju	Delhi
102	Sham	Mumbai
103	Paul	Chennai
104	Alex	Pune

# Creating a new Table

```
CREATE TABLE students (
    id INT,
    name VARCHAR(100)
)
```

```
CREATE TABLE students (name VARCHAR(100), age INT)
```

# Checking your table

**DESC TABLE**

# Adding data into a Table

```
INSERT INTO students(id, name)  
VALUES (101, "Rahul")
```

```
INSERT INTO students VALUES (101, "Rahul")
```

# Reading data from a Table

```
SELECT * FROM <table_name>
```

```
SELECT <column_name> from students
```

# Modify/Update data from a Table

```
UPDATE students  
SET contact=12345  
WHERE name='Raju';
```

# **DELETE data from a Table**

```
DELETE FROM students  
WHERE name='Raju' ;
```

# Data Types

# DataTypes

An attribute that specifies the type of data in a column of our database - table.

<b>id</b>	<b>name</b>	<b>city</b>
101	Raju	Delhi
102	Sham	Mumbai
103	Paul	Chennai
104	Alex	Pune

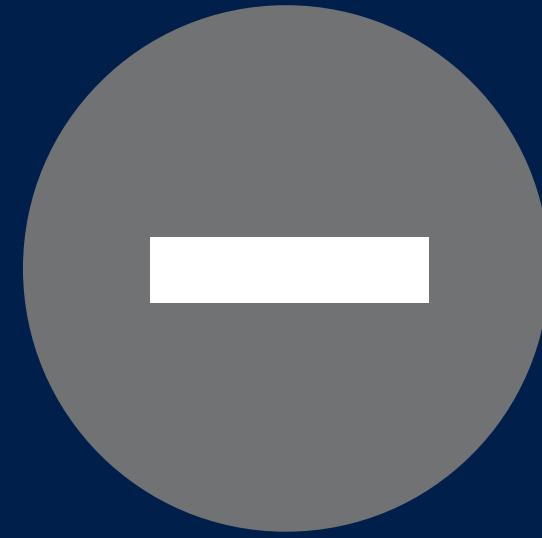
```
CREATE TABLE students (
    id INT,
    name VARCHAR(100)
)
```

**Most widely used are**

**Numeric - INT DOUBLE FLOAT DECIMAL**

**String - VARCHAR**

**Date**



**NOT NULL**

```
CREATE TABLE customers
(
    id INT NOT NULL,
    name VARCHAR(100) NOT NULL
);
```

# DEFAULT value

```
CREATE TABLE employee(  
    name VARCHAR(100),  
    acc_type VARCHAR(50) DEFAULT 'savings'  
);
```

# Primary Key



# 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.

acc_no	name	acc_type
101	Raju	Savings
102	Sham	Savings
103	Paul	Current
104	Alex	Savings

```
CREATE TABLE customers
(
    acc_no INT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    acc_type VARCHAR(50) NOT NULL DEFAULT 'Savings'
);
```

# AUTO\_INCREMENT

```
CREATE TABLE customers
(
    acc_no INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100) NOT NULL,
    acc_type VARCHAR(50) NOT NULL DEFAULT 'Savings'
);
```

# ALIAS



**WHERE**

```
SELECT * FROM students  
WHERE  
name='Raju';
```

```
SELECT * FROM students WHERE name='Raju';
```

```
UPDATE students SET contact=12345 WHERE id=104;
```

```
DELETE FROM students WHERE id=104;
```



# Exercise - 1

# Task

- Create a database - **bank\_db**
- Create a table - **employees**
  - **emp\_id**
  - **name**
  - **desig**
  - **dept**
- **emp\_id** column should not allow duplicate and null values
  - Value of **emp\_id** column should auto increment
- **name** column should not contain null value
- **desig** column should have default value as '**Probation**'

# Task

- Insert the following data

emp_id	name	desig	dept
101	Raju	Manager	Loan
102	Sham	Cashier	Cash
103	Paul	Associate	Loan
104	Alex	Accountant	Account
105	Victor	Associate	Deposit

# Task

- Display the data

emp_id	name	desig	dept
101	Raju	Manager	Loan
102	Sham	Cashier	Cash
103	Paul	Associate	Loan
104	Alex	Accountant	Account
105	Victor	Associate	Deposit

emp_id	name
101	Raju
102	Sham
103	Paul
104	Alex
105	Victor



# Exercise - 2

emp_id	name	desig	dept
101	Raju	Manager	Loan
102	Sham	Cashier	Cash
103	Paul	Associate	Loan
104	Alex	Accountant	Account
105	Victor	Associate	Deposit

# Task

- Display the data

emp_id	name	desig	dept
101	Raju	Manager	Loan
103	Paul	Associate	Loan

emp_id	name	desig	dept
101	Raju	Manager	Loan

emp_id	name
101	Raju

# Task

- Update the following data

emp_id	name	desig	dept
101	Raju	Manager	Loan
102	Sham	Cashier	Cash
103	Paul	Associate	IT
104	Alex	Accountant	Account
105	Victor	Associate	Deposit

# Task

- DELETE the following data

emp_id	name	desig	dept
101	Raju	Manager	Loan
102	Sham	Cashier	Cash
103	Paul	Associate	IT
104	Alex	Accountant	Account
105	Victor	Associate	Deposit



# String Functions



# CONCAT

`CONCAT(first_col, sec_col)`

`CONCAT(first_word, sec_word, ...)`



# CONCAT\_WS

`CONCAT_WS( '-' , fname , lname )`



# SUBSTRING

```
SELECT SUBSTRING('Hey Buddy', 1, 4);
```



# SUBSTRING

```
SELECT SUBSTRING('Hey Buddy', 1, 4);  
          1   4
```

Result: Hey



# REPLACE



# REPLACE

Hey Buddy

Hey → Hello

Hello Buddy



**REPLACE(str, from\_str, to\_str)**

**REPLACE('Hey Buddy', 'Hey', 'Hello')**



# REVERSE

```
SELECT REVERSE('Hello World');
```



# CHAR\_LENGTH

Select CHAR\_LENGTH('Hello World');



# UPPER & LOWER

```
SELECT UPPER('Hello World');
```

```
SELECT LOWER('Hello World');
```



# Other Functions

```
SELECT INSERT( 'Hello Wassup' , 6, 0, 'Raju' );
```

```
SELECT LEFT( 'Abcdefghij' , 3);
```

```
SELECT RIGHT( 'Abcdefghij' , 4);
```

```
SELECT REPEAT( 'o' , 5);
```

```
SELECT TRIM( ' Alright! ' );
```



# Exercise - 3

emp_id	fname	lname	desig	dept
101	Raju	Rastogi	Manager	Loan
102	Sham	Mohan	Cashier	Cash
103	Baburao	Apte	Associate	Loan
104	Paul	Philip	Accountant	Account
105	Alex	Watt	Associate	Deposit

**Task 1:**

**101:Raju:Manager:Loan**

**Task2:**

**101:Raju Rastogi:Manager:Loan**

**Task3**

**101:Raju:MANAGER:Loan**

**Task4**

**L101 Raju**

**C102 Sham**



# DISTINCT

```
SELECT DISTINCT fname FROM employees;
```



# ORDER BY

```
SELECT * FROM employees ORDER BY fname;
```



# LIKE

```
Select * FROM employees  
WHERE dept LIKE "%Acc%";
```



# LIMIT

```
SELECT * FROM employees LIMIT 3;
```



```
ALTER TABLE employees  
ADD COLUMN  
salary INT NOT NULL  
DEFAULT 25000;
```



# GROUP BY

emp_id	fname	lname	desig	dept
101	Raju	Rastogi	Manager	Loan
102	Sham	Mohan	Cashier	Cash
103	Baburao	Apte	Associate	Loan
104	Paul	Philip	Accountant	Account
105	Alex	Watt	Associate	Deposit
106	Rick	Watt	Manager	Account
107	Leena	Jhonson	Lead	Cash
108	John	Paul	Manager	IT
109	Alex	Watt	Probation	Loan



Loan



Cash



Account



IT



Deposit

```
SELECT dept FROM employees GROUP BY dept;
```

```
SELECT dept, COUNT(fname) FROM employees GROUP  
BY dept;
```



# COUNT

```
SELECT COUNT(*) FROM employees;
```



# MAX & MIN

```
SELECT MAX(age) FROM employees;  
SELECT MIN(age) FROM employees;
```

```
SELECT emp_id, fname, salary FROM employees
WHERE
salary = (SELECT MAX(salary) FROM employees);
```



# SUM & AVG

```
SELECT SUM(salary) FROM employees;  
SELECT AVG(salary) FROM employees;
```



# Exercise - 4

DISTINCT, ORDER BY, LIKE and LIMIT

emp_id	fname	lname	desig	dept	salary
101	Raju	Rastogi	Manager	Loan	37000
102	Sham	Mohan	Cashier	Cash	32000
103	Baburao	Apte	Associate	Loan	25000
104	Paul	Philip	Accountant	Account	45000
105	Alex	Watt	Associate	Deposit	35000
106	Rick	Watt	Manager	Account	65000
107	Leena	Jhonson	Lead	Cash	25000
108	John	Paul	Manager	IT	75000
109	Alex	Watt	Probation	Loan	40000

- 1: Find Different type of departments in database?
- 2: Display records with High-low salary
- 3: How to see only top 3 records from a table?
- 4: Show records where first name start with letter 'A'
- 5: Show records where length of the lname is 4 characters

**1**

dept
Loan
Cash
Account
Deposit
IT

emp_id	fname	lname	desig	dept	salary
108	John	Paul	Manager	IT	75000
106	Rick	Watt	Manager	Account	65000
104	Paul	Philip	Accountant	Account	45000
109	Alex	Watt	Probation	Loan	40000
101	Raju	Rastogi	Manager	Loan	37000
105	Alex	Watt	Associate	Deposit	35000
102	Sham	Mohan	Cashier	Cash	32000
103	Baburao	Apte	Associate	Loan	25000
107	Leena	Jhonson	Lead	Cash	25000

**2****3**

emp_id	fname	lname	desig	dept	salary
101	Raju	Rastogi	Manager	Loan	37000
102	Sham	Mohan	Cashier	Cash	32000
103	Baburao	Apte	Associate	Loan	25000

**4**

emp_id	fname	lname	desig	dept	salary
105	Alex	Watt	Associate	Deposit	35000
109	Alex	Watt	Probation	Loan	40000

**5**

emp_id	fname	lname	desig	dept	salary
103	Baburao	Apte	Associate	Loan	25000
105	Alex	Watt	Associate	Deposit	35000
106	Rick	Watt	Manager	Account	65000
108	John	Paul	Manager	IT	75000
109	Alex	Watt	Probation	Loan	40000



# Exercise - 5

COUNT, GROUP BY, MIN, MAX and SUM and AVG

emp_id	fname	lname	desig	dept	salary
101	Raju	Rastogi	Manager	Loan	37000
102	Sham	Mohan	Cashier	Cash	32000
103	Baburao	Apte	Associate	Loan	25000
104	Paul	Philip	Accountant	Account	45000
105	Alex	Watt	Associate	Deposit	35000
106	Rick	Watt	Manager	Account	65000
107	Leena	Jhonson	Lead	Cash	25000
108	John	Paul	Manager	IT	75000
109	Alex	Watt	Probation	Loan	40000

**1: Find Total no. of employees in database?**

**2: Find no. of employees in each department.**

**3: Find lowest salary paying**

**4: Find highest salary paying**

**5: Find total salary paying in Loan department?**

**6: Average salary paying in each department**

Dept	Count
Loan	3
Cash	2
Account	2
Deposit	1
IT	1

Dept	Avg_Salary
Loan	34000.0000
Cash	28500.0000
Account	55000.0000
Deposit	35000.0000
IT	75000.0000

emp_id	fname	lname	desig	dept	salary
108	John	Paul	Manager	IT	75000



# DATATYPES

**CHAR vs VARCHAR**



## DATA TYPES

The **CHAR** and **VARCHAR** types are similar,  
but differ in the way they are stored  
and retrieved.



MySQL®

## DATA TYPES

### VARCHAR(50)

The length can be specified as a value from 0 to 65,535.



**MySQL®**

# **DATA TYPES**

**CHAR is fixed length**

**CHAR(5) --> Only 5 character allowed**

**The length can be any value from 0 to 255.**



## DATA TYPES

When CHAR values are stored, they are right-padded with spaces to the specified length.

When CHAR values are retrieved, trailing spaces are removed unless the PAD\_CHAR\_TO\_FULL\_LENGTH SQL mode is enabled.

**CHAR(4)**

If you try to insert value 'AB'  
MySQL will store the value as 'AB ' '

<b>Value</b>	<b>CHAR (4)</b>	<b>Storage Required</b>	<b>VARCHAR (4)</b>	<b>Storage Required</b>
''	' '	4 bytes	''	1 byte
'ab'	'ab '	4 bytes	'ab'	3 bytes
'abcd'	'abcd'	4 bytes	'abcd'	5 bytes
'abcdefg'h'	'abcd'	4 bytes	'abcd'	5 bytes

**CHAR** can be used in case of fixed length values  
like

**Country code:** **US, IN, AU**



# DATAYPES

## DECIMAL



## **DATAYPES**

**So far we have only used and seen  
INT Datatype...**



# DATA TYPES

What will happen when we store  
values like 15.35?



# DATA TYPES

**DECIMAL(6,3)**

The maximum number of digits for DECIMAL is 65



# MySQL® DATATYPES

Digits after decimal

**DECIMAL(5,2)**

    
Total digit



# MySQL® DATATYPES

Example: 155.38

- |         |   |
|---------|---|
| 119.12  | ✓ |
| 28.15   | ✓ |
| 1150.15 | ✗ |



# DATAYPES

FLOAT, DOUBLE



# FLOAT, DOUBLE

**MySQL uses four bytes for single-precision values and eight bytes for double-precision**



# FLOAT, DOUBLE

**Float** - upto ~7 digits, takes 4 bytes of memory

**Double** - upto ~15 digits, takes 8 bytes of memory

If accuracy is not too important or if speed is the highest priority, the DOUBLE type may be good enough.



# DATAYPES

**DATE, TIME, DATETIME**



# **DATAYPES**

**DATE, TIME, DATETIME**

**DATE**

**yyyy-mm-dd format**



# **DATAYPES**

**DATE, TIME, DATETIME**

**TIME**

**HH:MM:SS format**



# **DATA TYPES**

**DATE, TIME, DATETIME**

**DATETIME**

**'yyyy-mm-dd HH:MM:SS' format**



# **DATE TIME Functions**

**CURDATE, CURTIME, NOW**



## DATE TIME Functions

**CURDATE()** - yyyy-mm-dd

**CURTIME()** - hh:mm:ss

**NOW()** - yyyy-mm-dd hh:mm:ss



# DATE TIME Functions

**DAYNAME ,  
DAYOFMONTH , DAYOFWEEK**

```
mysql> SELECT DAYNAME('2007-02-03');  
-> 'Saturday'
```

```
mysql> SELECT DAYOFMONTH('2007-02-03');  
-> 3
```

```
mysql> SELECT DAYOFWEEK('2007-02-03');  
-> 7
```

```
mysql> SELECT MONTHNAME( '2008-02-03' );
      -> 'February'
```

```
mysql> SELECT HOUR( '10:05:03' );
      -> 10
mysql> SELECT HOUR( '272:59:59' );
      -> 272
```



# **DATE Formatting**



## DATE Formatting

Suppose we need to get date in a given format like

- Tue Mar 28th
- 21st Tue at 21:20:28
- 2023/04/18



## DATE Formatting

### DATE\_FORMAT()

`DATE_FORMAT(now(), '%D %a at %T')`

Result: 21st Tue at 21:20:28

`DATE_FORMAT(now(), '%m/%d/%y')`

Result: 04/16/23



## DATE Formatting

`DATE_FORMAT()`

`DATE_FORMAT(now(), '%H:%i')`

Result: 20:34

`DATE_FORMAT(dob, '%r')`

Result: 08:35:48 PM



# DATE MATH



MATH



+



÷

×

**DATEDIFF(expr1, expr2)**

**DATE\_ADD(date, INTERVAL expr unit)**  
**DATE\_SUB(date, INTERVAL expr unit)**

**DATE\_ADD('2023-05-01',INTERVAL 1 DAY)**

**DATE\_ADD('2023-05-01',INTERVAL 1 YEAR)**

**DATE\_SUB('2023-05-01',INTERVAL 1 MONTH)**

**TIMEDIFF(expr1, expr2)**

**TIMEDIFF('20:00:00', '18:00:00')**



# **DEFAULT & ON UPDATE TIMESTAMP**



• 00:00:00:28

```
CREATE TABLE blogs (
    text VARCHAR(150),
    created_at DATETIME default CURRENT_TIMESTAMP,
    updated_at DATETIME ON UPDATE CURRENT_TIMESTAMP
);
```



# Exercise - 6



# Exercise - 6

- Print the current time (HH:MM:SS)
- Print the current date (yyyy-mm-dd)
- Print current day of the week (Monday, Tuesday...)
- What is the use case of CHAR datatype?
- Which datatype can be used to store value 123.456?
- Display date in format
  - dd:mm:yyyy
  - April 22nd at 22:00:00



# Exercise - 6

# Solution



# Exercise - 6

- Print the current time (HH:MM:SS)
- Print the current date (yyyy-mm-dd)
- Print current day of the week (Monday, Tuesday...)
- What is the use case of CHAR datatype?
- Which datatype can be used to store value 123.456?
- Display date in format
  - dd:mm:yyyy
  - April 22nd at 22:00:00



# Exercise - 6

- `SELECT CURTIME();`
- `SELECT CURDATE();`
- `SELECT DAYNAME(NOW());`
- What is the use case of CHAR datatype?
- Which datatype can be used to store value 123.456?
- Display date in format
  - dd:mm:yyyy - `date_format(now(), '%d:%m:%Y')`
  - April 22nd at 22:00:00 - `date_format(now(), '%M %D at %T')`



# Operators





# Relational Operators



# Find employees whose salary is more than 65000



# We have relational operators

Operators	Meaning
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Equal to
!=	Not equal to

```
SELECT * FROM employees  
WHERE salary > 65000;
```



# Logical Operators



**AND**

**OR**

Condition 1

AND

Condition 2

When both the conditions are true

salary=25000 AND dept=Loan

103   Baburao   Apte	Associate	Loan	25000
----------------------	-----------	------	-------

Condition 1

OR

Condition 2

When either of the condition is true

salary=65000

OR

desig='Lead'



**IN, NOT IN**

# Find employees From following department

- 
- 
- 

Account  
Cash  
Loan



```
SELECT * FROM employees  
WHERE dept = 'Account'  
      OR dept = 'Cash'  
      OR dept = 'Loan';
```

```
SELECT * FROM employees  
WHERE dept IN ('Account', 'Cash', 'Loan');
```



# BETWEEN

# Find employees whose salary is more than 40000 and Less than 65000

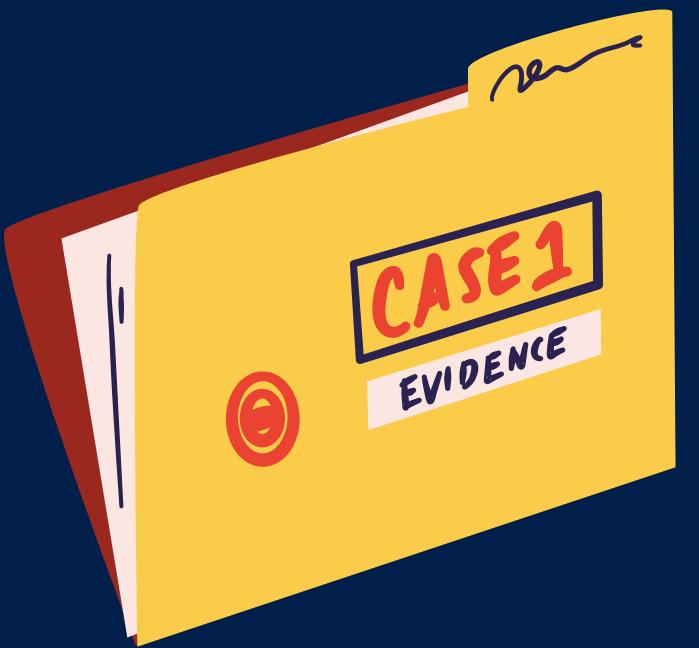
>40000

<65000



```
SELECT * FROM employees  
WHERE  
salary >=40000 AND salary <=65000;
```

```
SELECT * FROM employees  
WHERE  
salary BETWEEN 40000 AND 65000;
```



# CASE

emp_id	fname	lname	desig	dept	salary
101	Raju	Rastogi	Manager	Loan	37000
102	Sham	Mohan	Cashier	Cash	32000
103	Baburao	Apte	Associate	Loan	25000
104	Paul	Philip	Accountant	Account	45000
105	Alex	Watt	Associate	Deposit	35000
106	Rick	Watt	Manager	Account	65000
107	Leena	Jhonson	Lead	Cash	25000
108	John	Paul	Manager	IT	75000
109	Alex	Watt	Probation	Loan	40000



fname	salary	Salary Category
Raju	37000	Low Salary
Sham	32000	Low Salary
Baburao	25000	Low Salary
Paul	45000	Low Salary
Alex	35000	Low Salary
Rick	65000	Higher Salary
Leena	25000	Low Salary
John	75000	Higher Salary
Alex	40000	Low Salary

```
SELECT
    fname,
    salary,
CASE
    WHEN salary >= 50000 THEN 'Higher Salary'
    ELSE 'Low Salary'
END AS 'Salary Category'
FROM
    employees;
```

```
SELECT
    fname,
    salary,
CASE
    WHEN salary >= 50000 THEN 'Higher Salary'
    WHEN salary >= 40000
    AND salary < 50000 THEN 'Mid Salary'
    ELSE 'Low Salary'
END AS 'Salary Category'
FROM
    employees;
```



**IS NULL**



# IS NULL

```
SELECT * FROM employees  
WHERE fname IS NULL;
```



NOT LIKE



# IS NULL

```
SELECT * FROM employees  
WHERE fname NOT LIKE 'A%';
```



# Exercise - 7

emp_id	fname	lname	desig	dept	salary
101	Raju	Rastogi	Manager	Loan	37000
102	Sham	Mohan	Cashier	Cash	32000
103	Baburao	Apte	Associate	Loan	25000
104	Paul	Philip	Accountant	Account	45000
105	Alex	Watt	Associate	Deposit	35000
106	Rick	Watt	Manager	Account	65000
107	Leena	Jhonson	Lead	Cash	25000
108	John	Paul	Manager	IT	75000
109	Alex	Watt	Probation	Loan	40000

- Find employees whose salary are between 30000 to 40000
- Find employees whose first name start with 'R' or 'S'
- Find employee whose salary=25000 and department should be 'Cash'
- Find employees from following designation
  - Manager, Lead and Associate

fname	salary	sal in dollars
Raju	37000	462.5000
Sham	32000	400.0000
Baburao	25000	312.5000
Paul	45000	562.5000
Alex	35000	437.5000
Rick	65000	812.5000
Leena	25000	312.5000
John	75000	937.5000
Alex	40000	500.0000



# Exercise - 7

# Solution

```
mysql> select * from employees  
-> where desig in ('Manager', 'Lead', 'Associate');
```

emp_id	fname	lname	desig	dept	salary
101	Raju	Rastogi	Manager	Loan	37000
103	Baburao	Apte	Associate	Loan	25000
105	Alex	Watt	Associate	Deposit	35000
106	Rick	Watt	Manager	Account	65000
107	Leena	Jhonson	Lead	Cash	25000
108	John	Paul	Manager	IT	75000

```
SELECT
    fname,
    salary,
    CASE
        WHEN salary != 0 THEN salary/80
    END AS 'Salary In Dollars'
FROM
    employees;
```



# UNIQUE

What if we need to store the phone numbers in a column.



```
CREATE TABLE contacts(  
    name VARCHAR(50),  
    mob VARCHAR(15) UNIQUE  
);
```



# CHECK

We want to make sure phone no. is  
atleast 10 digits...



```
CREATE TABLE contacts(  
    name VARCHAR(50),  
    mob VARCHAR(15) UNIQUE CHECK (LENGTH(mob) >= 10)  
);
```

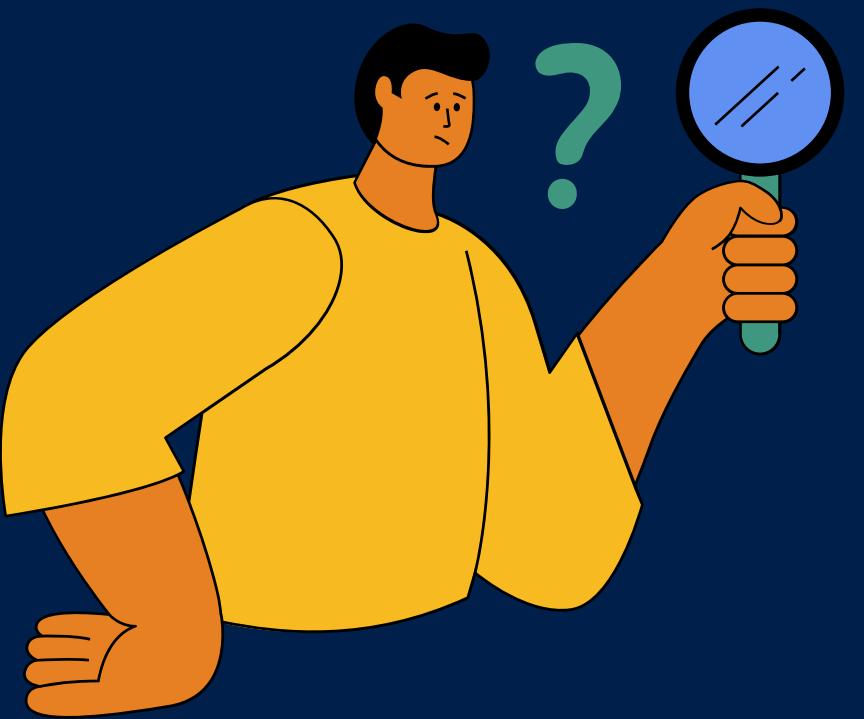
# NAMED CONSTRAINT

```
CREATE TABLE contacts(  
    name VARCHAR(50),  
    mob VARCHAR(15) UNIQUE,  
    CONSTRAINT mob_no_less_than_10digits CHECK (LENGTH(mob) >= 10)  
);
```



# ALTERING Tables

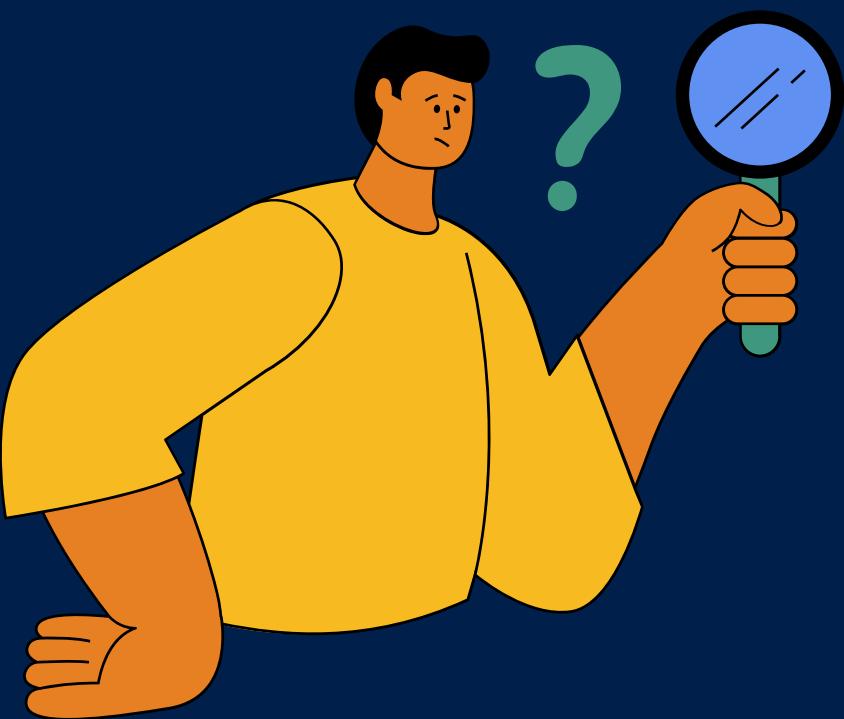
# How to add or remove a column?



```
ALTER TABLE contacts  
ADD COLUMN city VARCHAR(50);|
```

```
ALTER TABLE contacts  
DROP COLUMN city;
```

# How to rename a column or table name?



# How to rename a column?

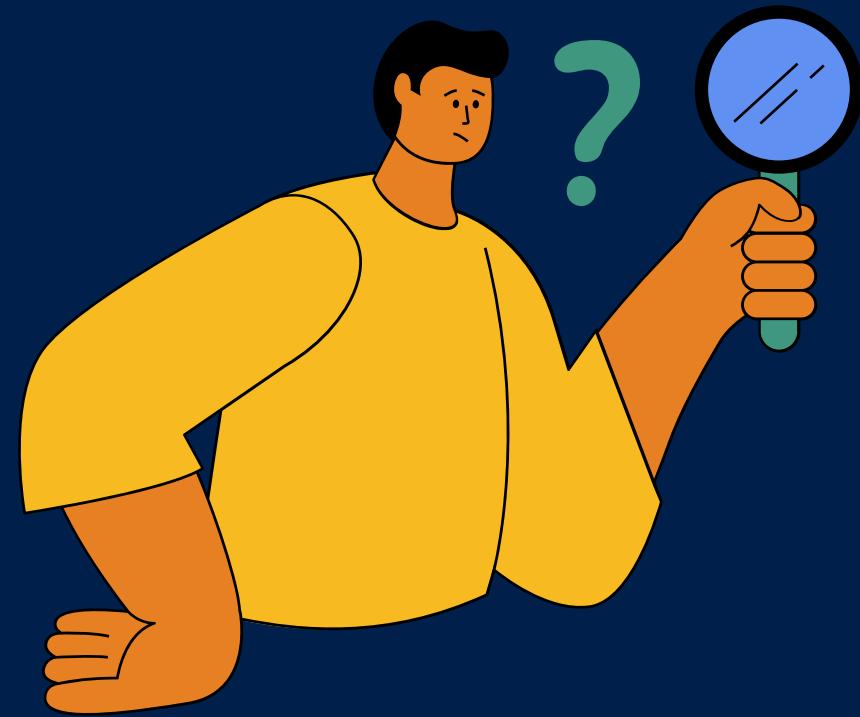
```
ALTER TABLE contacts  
RENAME COLUMN name TO full_name;
```

# How to rename a table name?

```
ALTER TABLE contacts  
RENAME TO mycontacts;
```

```
RENAME TABLE contacts TO mycontacts;
```

# How to modify a column?



**Ex: Changing datatype  
or adding Default values etc**

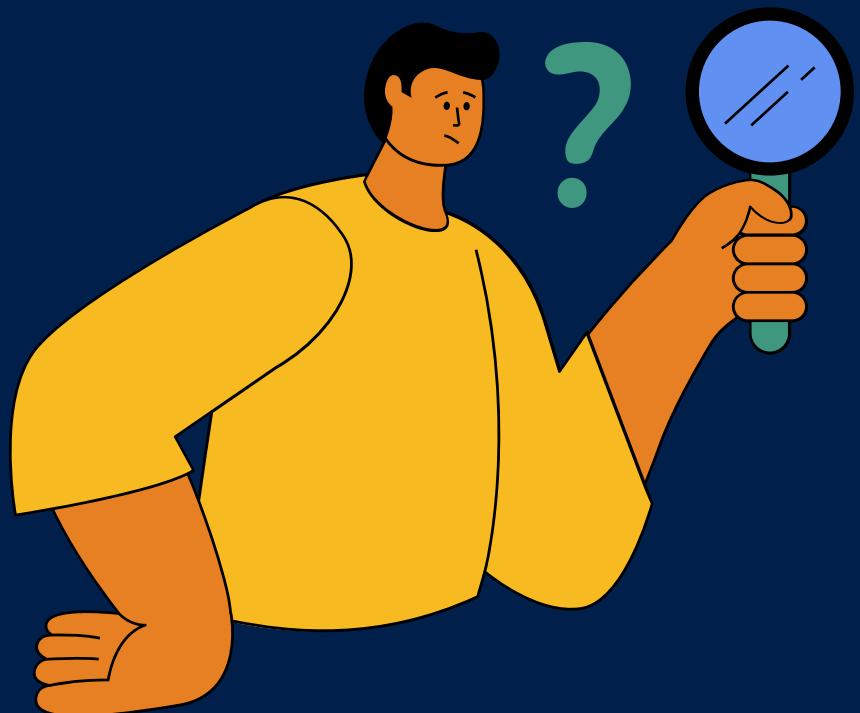
# How to add DEFAULT value to a column?

```
ALTER TABLE contacts  
MODIFY mob VARCHAR(15) DEFAULT 'unknown';
```

# How to change column name and set NOT NULL?

```
ALTER TABLE contacts  
CHANGE name emp_name VARCHAR(50) NOT NULL;
```

# How to Add or Drop Constraints?



# How to Check Existing Constraints?

```
SELECT  
    CONSTRAINT_TYPE,CONSTRAINT_NAME  
FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS  
WHERE TABLE_NAME='contacts';
```

```
ALTER TABLE contacts  
DROP CONSTRAINT mob_no_less_than_10digits;
```

```
ALTER TABLE contacts  
ADD CONSTRAINT mob_not_null CHECK (mob != null);
```

# Relationship





# RELATIONSHIP

# Simple data and table...

emp_id	fname	lname	desig	dept	salary
101	Raju	Rastogi	Manager	Loan	37000
102	Sham	Mohan	Cashier	Cash	32000
103	Baburao	Apte	Associate	Loan	25000
104	Paul	Philip	Accountant	Account	45000
105	Alex	Watt	Associate	Deposit	35000
106	Rick	Watt	Manager	Account	65000
107	Leena	Jhonson	Lead	Cash	25000
108	John	Paul	Manager	IT	75000
109	Alex	Watt	Probation	Loan	40000

# Real World



# Data is not that simple...



**Employees**

**Salary**

**Attendance**

**Employees**

**requests**

**offices**

**task**

# Types of Relationship

- One to One
- One to Many
- Many to Many

1 : 1

## Employees

emp_id	name	dept
101	Raju	IT
102	Sham	Finance

## Employee Details

emp_id	addr	City	phone	title
101	CP	Delhi	909090909	Manager
102	Bhandup	Mumbai	902020200	Accountant

**1 : MANY**

## Employees

emp_id	name	dept
101	Raju	IT
102	Sham	Finance

## Employee Task

emp_id	task_no	task_detail
101	TS-1	Opening account for Ram
102	TS-2	Closing account for Neru
101	TS-3	Loan sanction

**Many : Many**



**Books**



**Authors**



**Book A**



**Author A**



**Book B**



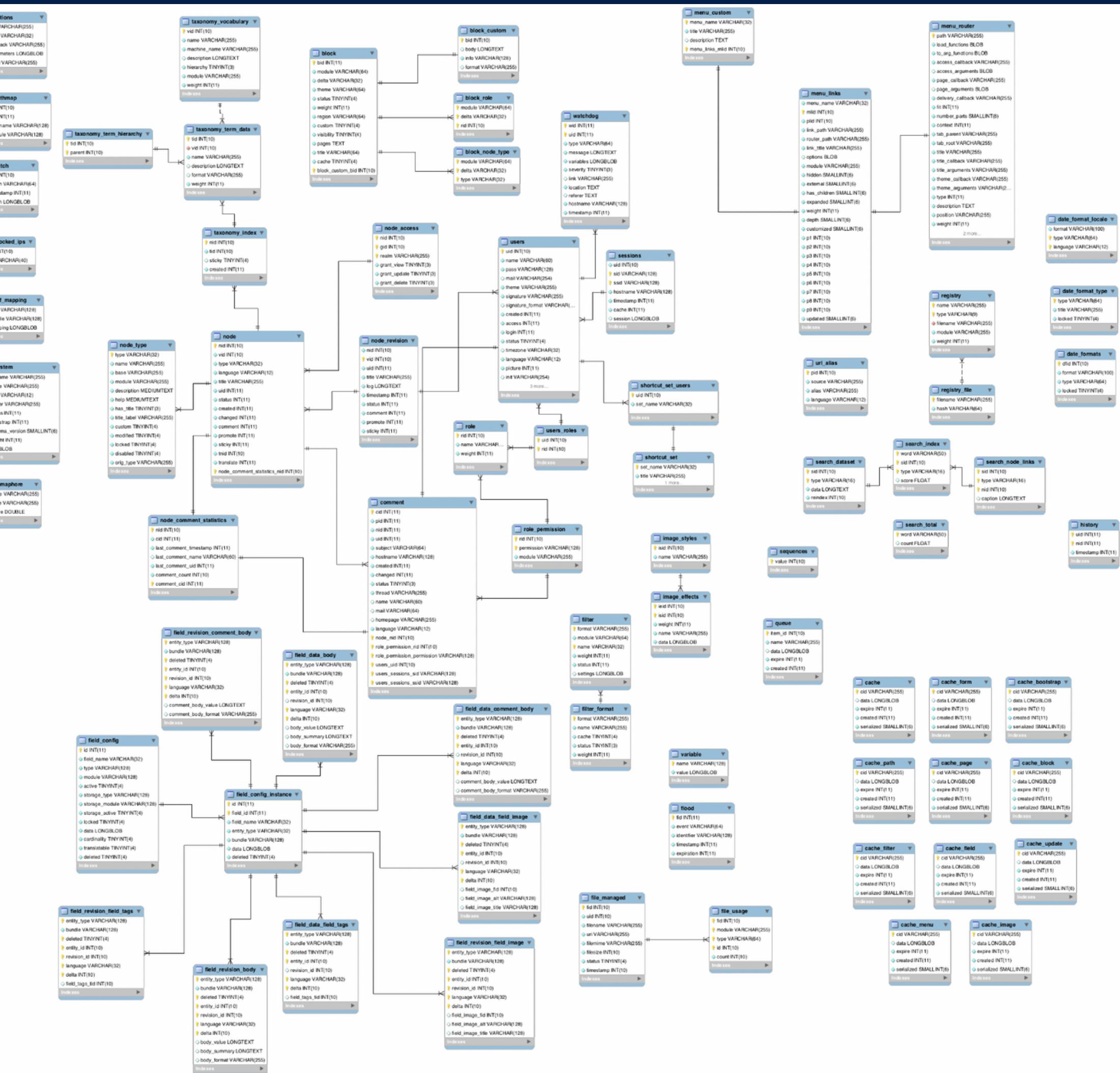
**Book C**



**Book D**



**Author B**



**Let's Understand a Use-Case of  
1:Many**





Suppose we need to store the following data

- customer name
- customer email
- order date
- order price

cust_name	email	order_daate	amount
Raju	raju@email.com	2023-05-15	200
Sham	sham@email.com	2023-04-28	500
Raju	raju@email.com	2023-05-14	1000
Baburao	babu@email.com	NULL	NULL
Sham	sham@email.com	2023-03-15	800

## **Customers**

**cust\_id**  
**cust\_name**  
**cust\_email**

## **Orders**

**order\_id**  
**order\_date**  
**order\_amount**

## Customers

**cust\_id**  
**cust\_name**  
**cust\_email**

## Orders

**order\_id**  
**order\_date**  
**order\_amount**  
**cust\_id**

## Customers

cust_id	name	email
101	Raju	raju@email.com
102	Sham	sham@email.com
103	Baburao	babu@email.com

## Orders

order_id	date	amount	cust_id
ord-1	2023-05-15	200	101
ord-2	2023-04-28	500	102
ord-3	2023-05-14	1000	101



# Foreign Key

## Customers

cust_id	name	email
101	Raju	raju@email.com
102	Sham	sham@email.com
103	Baburao	babu@email.com

## Orders

order_id	date	amount	cust_id
ord-1	2023-05-15	200	101
ord-2	2023-04-28	500	102
ord-3	2023-05-14	1000	101

## Primary Key

Customers

cust_id	name	email
101	Raju	raju@email.com
102	Sham	sham@email.com
103	Baburao	babu@email.com

Orders

## Primary Key

## Foreign Key

order_id	date	amount	cust_id
ord-1	2023-05-15	200	101
ord-2	2023-04-28	500	102
ord-3	2023-05-14	1000	101

**Let's work practically with  
Foreign Key..**

```
CREATE table customers(  
    cust_id INT AUTO_INCREMENT PRIMARY KEY,  
    name VARCHAR(50),  
    email VARCHAR(50)  
);
```

```
CREATE TABLE orders(
    ord_id INT AUTO_INCREMENT PRIMARY KEY,
    date DATE,
    amount DECIMAL(10, 2),
    cust_id INT,
    FOREIGN KEY (cust_id) REFERENCES customers(cust_id)
);
```



# JOINS

**JOIN** operation is used to combine rows from two or more tables based on a related column between them.

## Primary Key

Customers

cust_id	name	email
101	Raju	raju@email.com
102	Sham	sham@email.com
103	Baburao	babu@email.com

Orders

## Primary Key

## Foreign Key

order_id	date	amount	cust_id
ord-1	2023-05-15	200	101
ord-2	2023-04-28	500	102
ord-3	2023-05-14	1000	101

# Types of Join

- Cross Join
- Inner Join
- Left Join
- Right Join

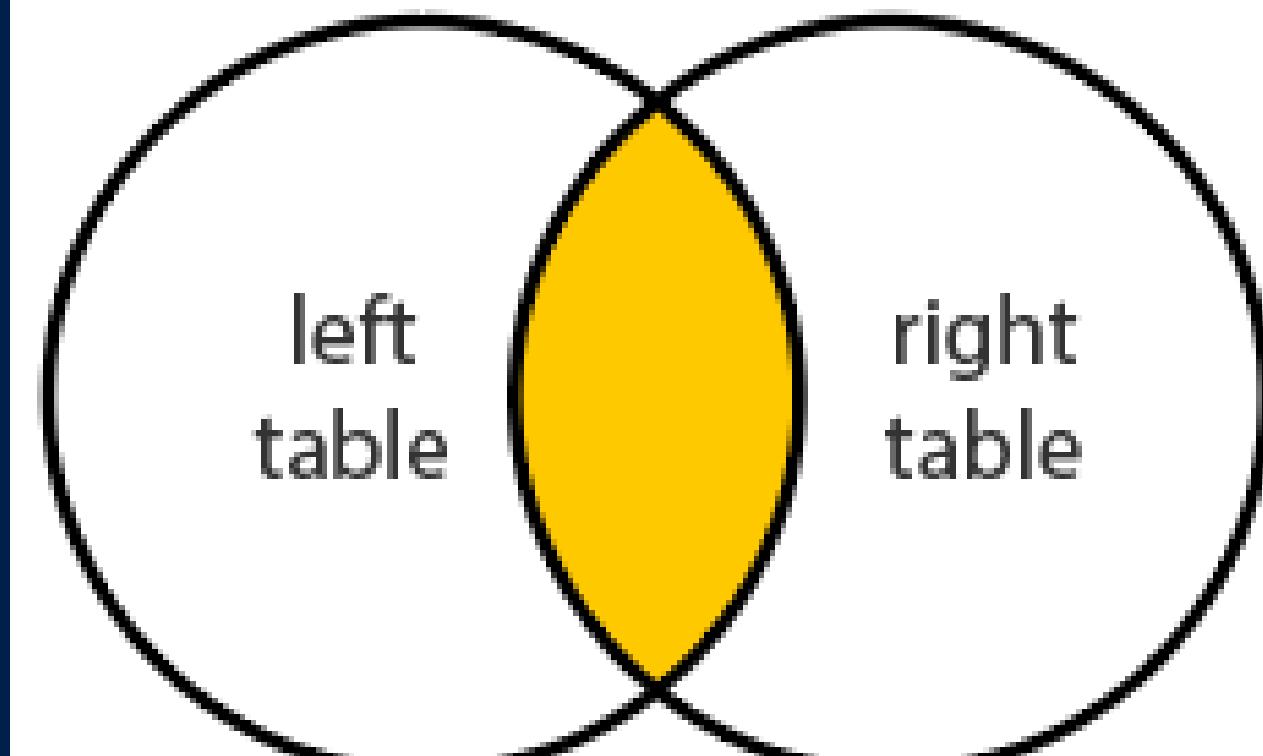
# Cross Join

**Every row from one table is combined with  
every row from another table.**

# Inner Join

Returns only the rows where there is a match between the specified columns in both the left (or first) and right (or second) tables.

## INNER JOIN



```
SELECT * FROM customers
INNER JOIN orders
ON orders.cust_id=customers.cust_id;
```

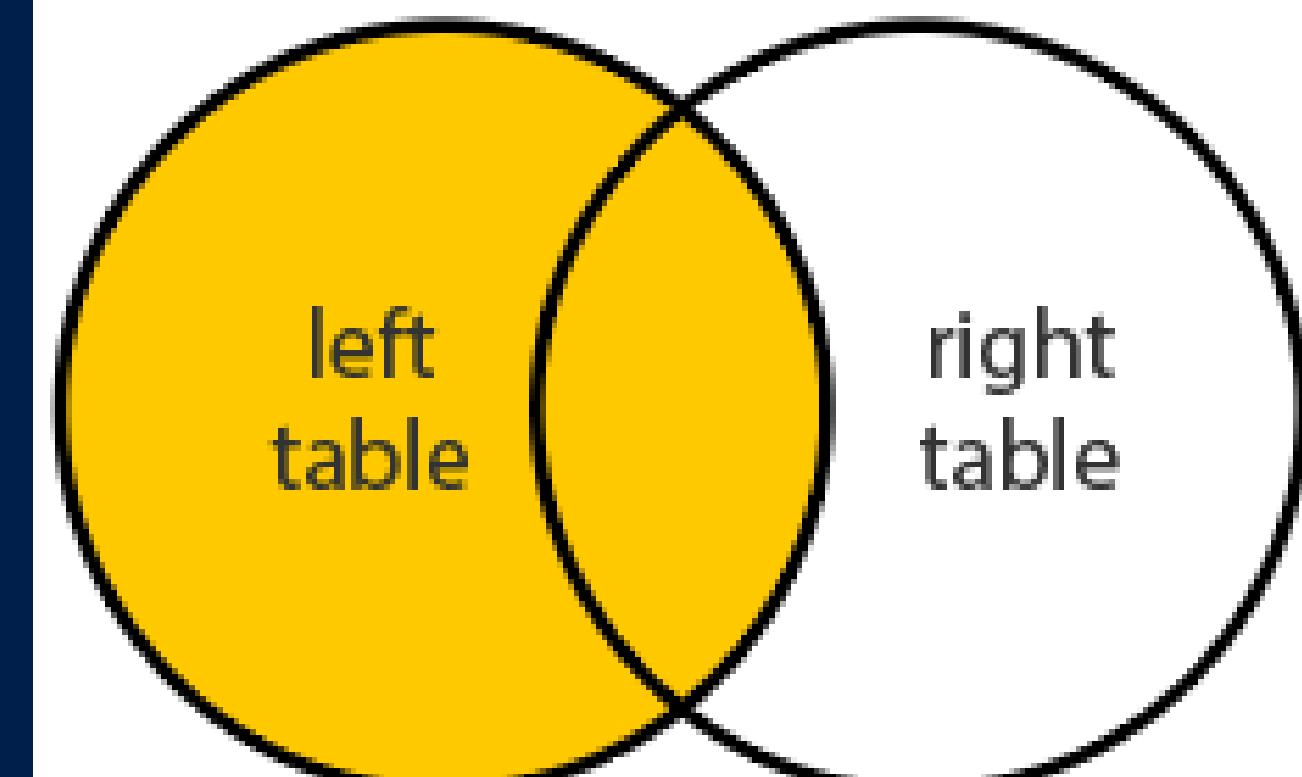
# Inner Join with Group By

```
SELECT name FROM customers
  INNER JOIN orders
    ON orders.cust_id=customers.cust_id
GROUP BY name;
```

# Left Join

Returns all rows from the left (or first) table  
and the matching rows from the right (or  
second) table.

## LEFT JOIN

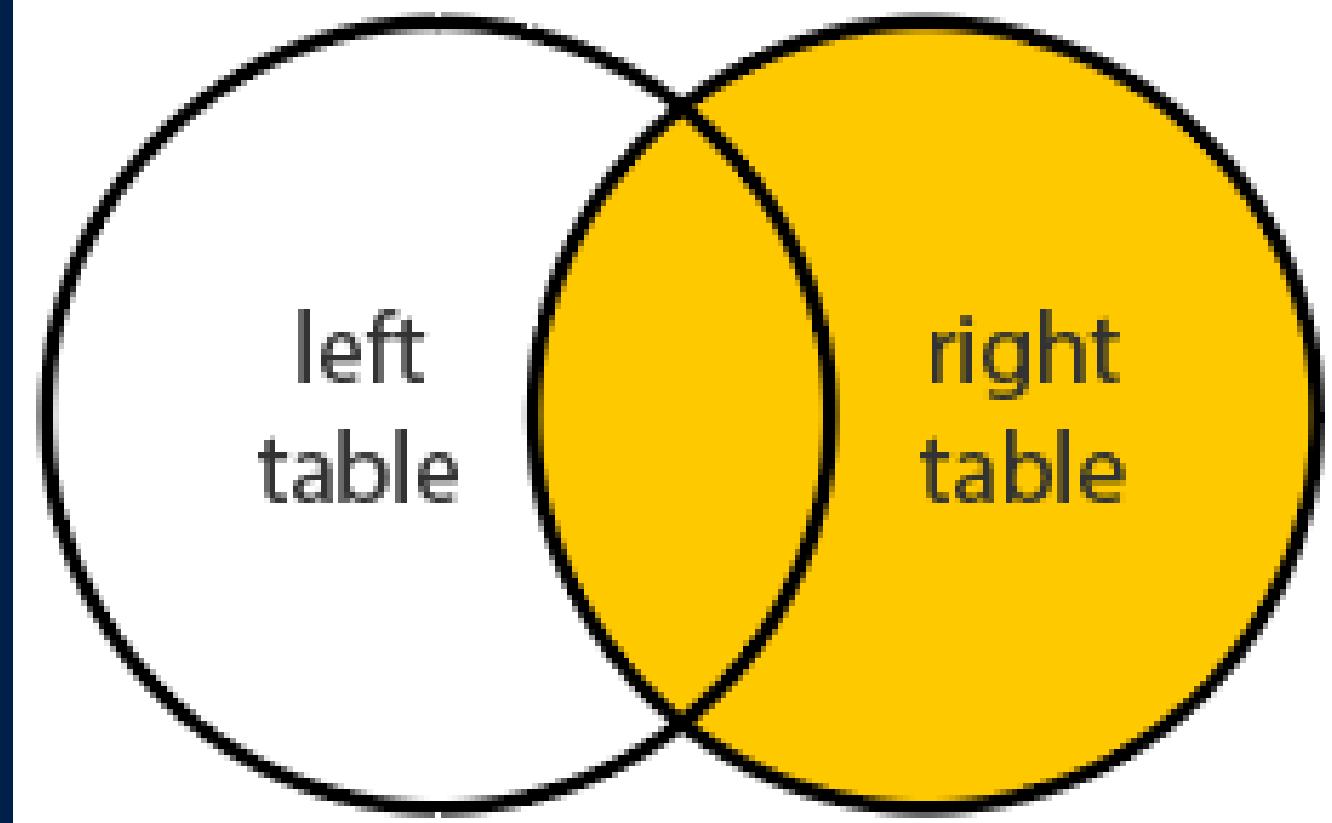


```
SELECT * FROM customers
LEFT JOIN orders
ON orders.cust_id=customers.cust_id;
```

# Right Join

Returns all rows from the right (or second) table and the matching rows from the left (or first) table.

## RIGHT JOIN



```
SELECT * FROM customers  
RIGHT JOIN orders  
ON orders.cust_id=customers.cust_id;
```



# CASCADE ON DELETE

## Primary Key

Customers

cust_id	name	email
101	Raju	raju@email.com
102	Sham	sham@email.com
103	Baburao	babu@email.com

Orders

## Primary Key

## Foreign Key

order_id	date	amount	cust_id
ord-1	2023-05-15	200	101
ord-2	2023-04-28	500	102
ord-3	2023-05-14	1000	101

```
CREATE TABLE orders(
    ord_id INT AUTO_INCREMENT PRIMARY KEY,
    date DATE,
    amount DECIMAL(10, 2),
    cust_id INT,
    FOREIGN KEY (cust_id) REFERENCES customers(cust_id) ON DELETE CASCADE
)
```



# Exercise - 8

# Authors

author\_id  
author\_name

# Books

book\_id  
title  
ratings  
au\_id



```
INSERT INTO authors (author_name)
VALUES ('Raju'), ('Sham'), ('Baburao'), ('Paul')
```

```
INSERT INTO books (title, ratings, au_id)
VALUES
  ('Story of Raju', 5, 1),
  ('Story of Baburao', 4, 3),
  ('Raju - The Great Man', 2, 1),
  ('Love story by Sham', 1, 2)
```

Raju	Story of Raju		5
Raju	Raju - The Great Man		2
Sham	Love story by Sham		1
Baburao	Story of Baburao		4

Raju	Story of Raju	5	
Raju	Raju - The Great Man	2	
Sham	Love story by Sham	1	
Baburao	Story of Baburao	4	
Paul	NULL	NULL	

Raju	Story of Raju	5
Raju	Raju - The Great Man	2
Sham	Love story by Sham	1
Baburao	Story of Baburao	4
Paul	Not Found	0

author_name	ratings	remark
Raju	5	Good
Raju	2	Average
Sham	1	Average
Baburao	4	Good



# Exercise - 8

# Solution



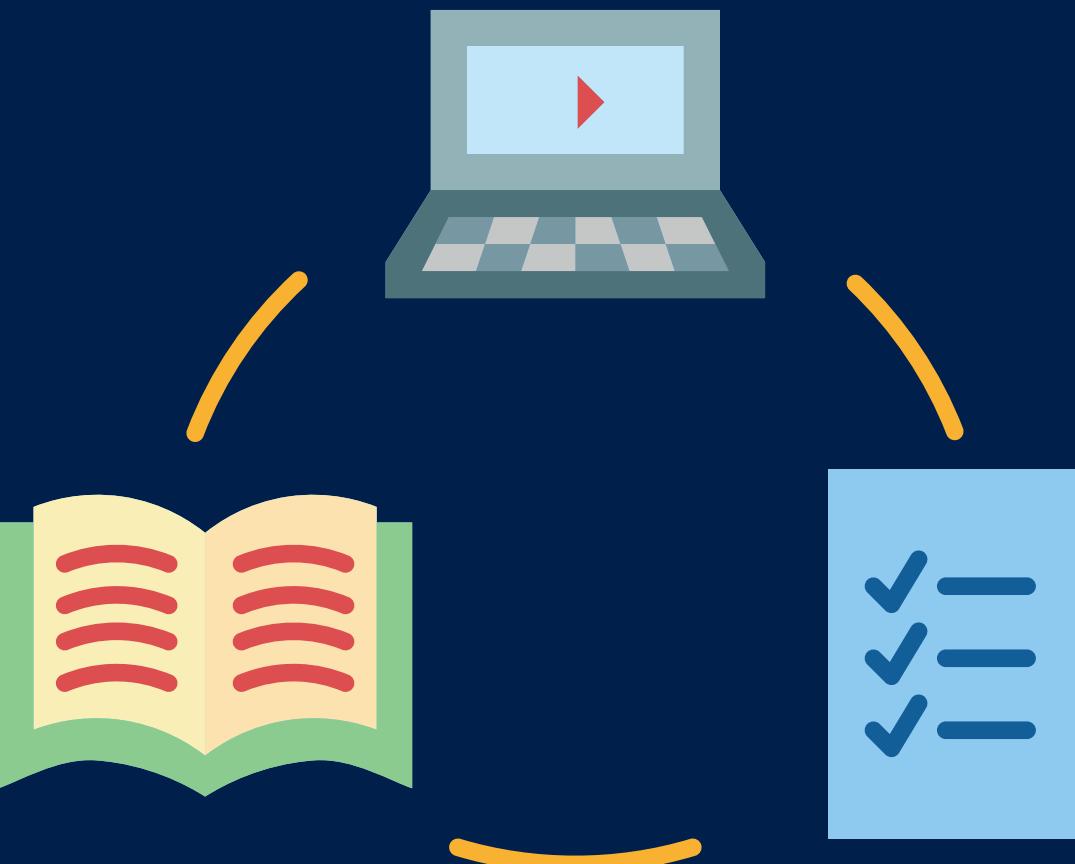
# Many : Many

Let's Understand a Use-Case of  
Many : Many

# Students



# Courses



**Student A**



**Course A**



**Course B**



**Course C**

# Course A



**Student A**



**Student B**



**Student C**

# students

- id
- student\_name

# courses

- id
- course\_name
- fees

# **students**

- **id**
- **student\_name**

# **courses**

- **id**
- **course\_name**
- **fees**

# **student\_course**

- **student\_id**
- **course\_id**

## **students**

- id
- student\_name

## **courses**

- id
- course\_name
- fees

## **student\_course**

- student\_id
- course\_id



```
# Joins with Many to Many relationship
SELECT student_name, course_name FROM students
JOIN
    student_course ON student_course.student_id=students.id
JOIN
    courses ON student_course.course_id=courses.id;
```



# Exercise - 9

# Solution

# Print No. of students for each course.

PD	1
Java	2
Linux	2
SQL	1
Python	1

# Print No. of courses taken by each students

Raju	3
Sham	2
Paul	1
Alex	1

# Total FEES Paid by Each Student

+-----+		18000		+-----+
Raju		18000		+-----+
Sham		15000		+-----+
Paul		4000		+-----+
Alex		6000		+-----+



# VIEWS

```
CREATE VIEW inst_info AS
SELECT student_name, course_name, fees FROM students
JOIN
    student_course ON student_course.student_id=students.id
JOIN
    courses ON student_course.course_id=courses.id;|
```



# HAVING Clause

```
SELECT student_name, SUM(fees) FROM inst_info  
GROUP BY student_name HAVING SUM(fees)>10000;
```



# GROUP BY ROLLUP

```
SELECT student_name, SUM(fees) FROM inst_info  
GROUP BY student_name WITH ROLLUP;
```



# STORED Routine

# **STORED Routine**

An SQL statement or a set of SQL Statement that can be stored on database server which can be call no. of times.

# Types of STORED Routine

- STORED Procedure
- User defined Functions



# STORED Procedure

# **STORED PROCEDURE**

These are routines that contain a series of SQL statements and procedural logic.

Often used for performing actions like data modification, transaction control, and executing sequences of statements.

```
| #We can drop if exist already  
DROP PROCEDURE IF EXISTS p_name;
```

```
| #Temp changing delimiter  
DELIMITER $$
```

```
CREATE PROCEDURE p_name()  
BEGIN  
    SELECT * FROM employees  
    LIMIT 10;  
END$$
```

```
| #Again changing Delimiter  
DELIMITER ;
```

```
DELIMITER $$
```

```
CREATE procedure get_empid(IN p_fname VARCHAR(50))  
BEGIN
```

```
    Select emp_id from employees  
    where fname=p_fname;  
END$$
```

```
DELIMITER ;
```

```
|
```

```
DELIMITER $$

CREATE procedure get_sum(IN p_fname VARCHAR(50), OUT p_sum DECIMAL(10,2))
BEGIN
    Select SUM(salary) INTO p_sum from employees;
END$$

DELIMITER ;
```



# USER DEFINED FUNCTIONS

```
DELIMITER $$

CREATE FUNCTION get_sum(p_fname VARCHAR(50)) RETURNS VARCHAR(50)
DETERMINISTIC NO SQL READS SQL DATA

BEGIN
    DECLARE v_max INT;
    DECLARE v_name VARCHAR(50);

    Select MAX(salary) INTO v_max from employees;
    Select fname into v_name from employees
        where salary=v_max;

    return v_name;
END$$

DELIMITER ;
```



# WINDOW FUNCTIONS

Introduced in version 8.0.

Window functions, also known as analytic functions allow you to perform calculations across a set of rows related to the current row.

Defined by an **OVER()** clause.

- **ROW\_NUMBER()**
- **RANK()**
- **DENSE\_RANK()**
- **LAG()**
- **LEAD()**