**TRUTH TABLE - Combinators or combinations**

Birthday gift 1: Dresser Set

Birthday gift 2: Money

Dresser Set OR Money

Scenario 1: You don’t get the dresser set. You don’t even get the money. Michael Sad.

Scenario 2: You get money but not the dresser set, Michael is happy.

Scenario 3: You get the dresser set but not money, Michael is happy.

Scenario 4: You get the money as well as the dresser set, Michael is HAPPY.

**TRUE FALSE # Boolean**

| **Dresser Set** | **Money** | **Michael Happy** |
| --- | --- | --- |
| False | False | False |
| False | True | True |
| True | False | True |
| True | True | True |

**AND**

**For the class to happen, Michael AND Abdul have to be available for the class.**

Scenario 1: I am not available for a class. You are not available for a class. The class will not happen.

Scenario 2: Abdul not available for class. Michael available for class. The class will not happen.

Scenario 3: Michael not available for class. Abdul available for class. The class will not happen.

Scenario 4: Michael is available! Abdul is available! The class will happen! Let’s goooooooooooooooooooo!

| **Abdul Available** | **Michael Available** | **Class Happens (Yay!)** |
| --- | --- | --- |
| False | False | False |
| False | True | False |
| True | False | False |
| True | True | True |

**NOT True = False**

**Not False = True**

**Not basically inverts**

**Select \* from customers WHERE city = ‘New York’**

**WHERE IS LITERALLY A FILTER**

**AND WHAT COMES AFTER IS THE BASIS OF THAT FILTER/ THE CONDITION**

**Select \* from customers WHERE state = ‘Florida’**

**Select \* from customers WHERE city IN (‘New York’, ‘Sarasota’, ‘Houston’, ‘Manchester’, ‘London’, ‘Amsterdam’, ‘Dallas’)**

**Select \* from students where age < 18; #Consent form from parents**

**Select \* from students where age >= 18; #Consent form different of liability waiver**

**Select \* from employees where salary between 70000 and 90000**

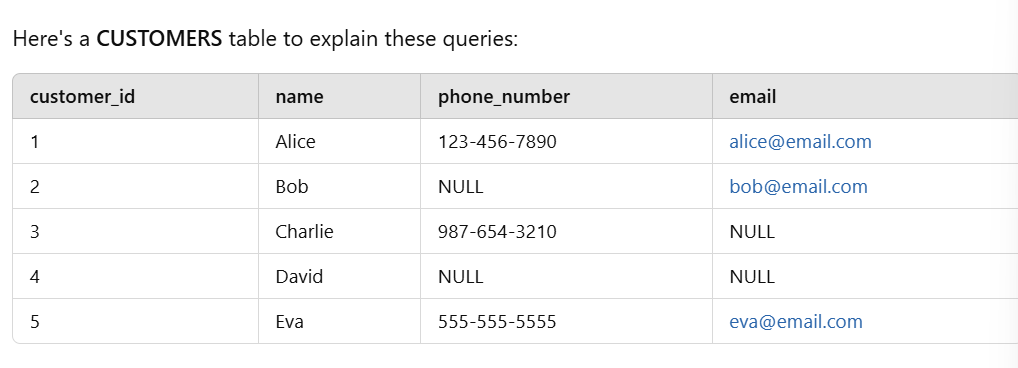
**Select \* from customers where phone\_number is NULL**

**#Find customers in the database whom number we don’t have**

**Select \* from customer where email is NOT NULL**

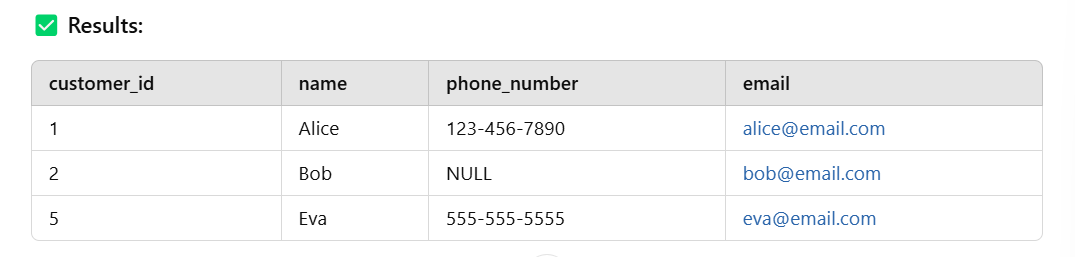
**#Simply list the users who have an email**

**Some customers have their phone number listed. Some have their email listed. Some have both listed. They can be contacted whether they have one or the other listed.**

****

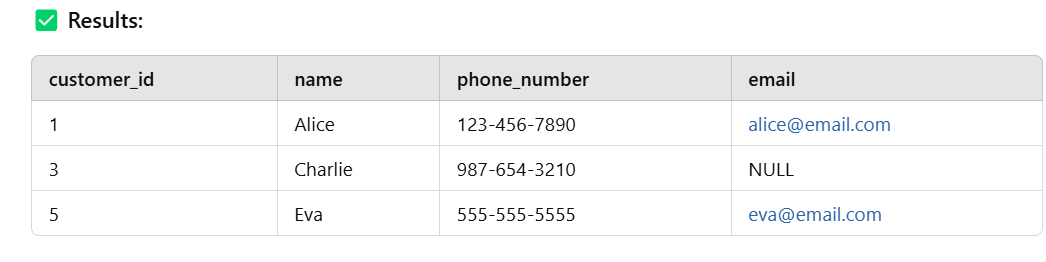
**SELECT \* FROM CUSTOMERS**

**WHERE email IS NOT NULL;**

****

**SELECT \* FROM CUSTOMERS**

**WHERE phone\_number IS NOT NULL;**

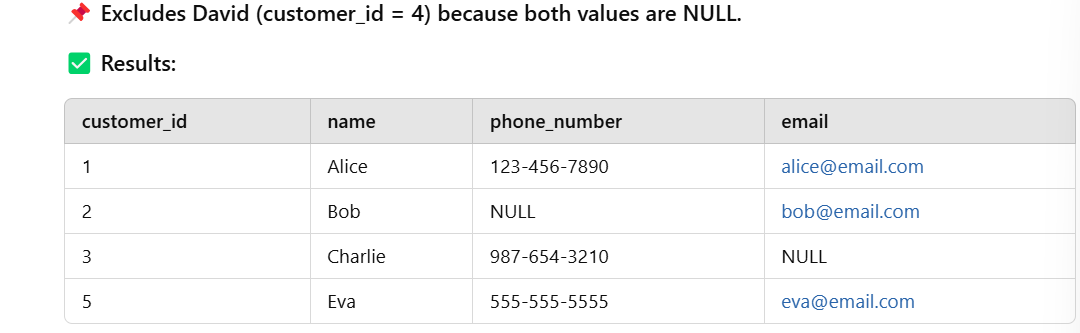
****

**Select names, addresses from citizens where age >=18 and age <=65;**

**#Query to get the names and addresses of all the citizens between age 18 and 65 who can drive.**

**SELECT \* FROM CUSTOMERS**

**WHERE phone\_number IS NOT NULL OR email IS NOT NULL;**

****

**Select \* from customers where premium is NOT NULL**

**#Showcase premium customers**

**SELECT \* FROM CUSTOMERS WHERE email is NOT NULL AND phone\_number is NOT NULL;**

**AND PLUS OR function in these queries**

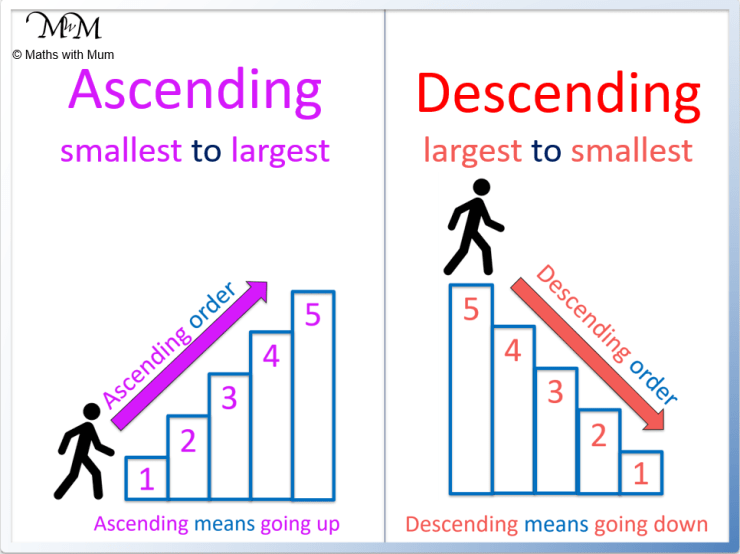
**Select \* from Orders where status = ‘Shipped’ LIMIT 5;**

**ORDER BY - A WAY TO SORT YOUR RESULTS ON THE BASE OF A COLUMN EITHER ASCENDING OR DESCENDING**

**Select \* from students ORDER BY GPA DESC;**

**Select \* from customer ORDER BY age ASC;**

**# Youngest to oldest customer**

****

**Select \* from students ORDER BY GPA DESC LIMIT 10;**

**FIRST YOU GOT THE STUDENT TABLE IN NATURAL FORM**

**THEN YOU GOT THE STUDENT TABLE ORDERED BY BASIS OF GPA DESCENDING**

**THEN FINALLY, YOU GOT THE TOP TEN ROWS OF THAT ORDERED BY TABLE.**

**ORDERED BY LITERALLY MEANING SORTED ON THE BASIS OF A COLUMN EITHER ASC OR DESC**

**—------------------------------------------------------------- Please copy**

**-- Create Database**

**CREATE DATABASE PlayDB;**

**USE PlayDB;**

**-- Create CUSTOMERS Table**

**CREATE TABLE CUSTOMERS (**

**customer\_id INT PRIMARY KEY AUTO\_INCREMENT,**

**name VARCHAR(50),**

**phone\_number VARCHAR(20),**

**email VARCHAR(50),**

**city VARCHAR(50),**

**age INT**

**);**

**-- Insert Data into CUSTOMERS Table**

**INSERT INTO CUSTOMERS (name, phone\_number, email, city, age) VALUES**

**('Alice', '123-456-7890', 'alice@email.com', 'New York', 30),**

**('Bob', NULL, 'bob@email.com', 'Chicago', 25),**

**('Charlie', '987-654-3210', NULL, 'Houston', 35),**

**('David', NULL, NULL, 'Los Angeles', 40),**

**('Eva', '555-555-5555', 'eva@email.com', 'Chicago', 28);**

**-- Create ORDERS Table**

**CREATE TABLE ORDERS (**

**order\_id INT PRIMARY KEY AUTO\_INCREMENT,**

**customer\_id INT,**

**amount DECIMAL(10,2),**

**city VARCHAR(50),**

**order\_date DATE,**

**FOREIGN KEY (customer\_id) REFERENCES CUSTOMERS(customer\_id)**

**);**

**-- Insert Data into ORDERS Table**

**INSERT INTO ORDERS (customer\_id, amount, city, order\_date) VALUES**

**(1, 100.00, 'New York', '2024-03-01'),**

**(2, 200.00, 'Chicago', '2024-03-02'),**

**(1, 150.00, 'New York', '2024-03-05'),**

**(3, 300.00, 'Houston', '2024-03-06'),**

**(2, 250.00, 'Chicago', '2024-03-07');**

**—---------------------**

**SELECT - CRUD - R FOR READ**

**Select \* from table\_name; #EVERY SINGLE ROW EVERY SINGLE COLUMN**

**Select name, age from CUSTOMERS; #EVERY SINGLE ROW BUT LIMITED OR CHOSEN COLUMNS**

**Select \* from CUSTOMERS Limit 4; #EVERY SINGLE COLUMN BUT JUST 4 ROWS -Only 4 customers!**

**SELECT NAME, AGE FROM CUSTOMERS LIMIT 3;**

**#I HAVE LIMITED BOTH - COLUMNS AND ROWS**

**#Limit rows to 3**

**#Limit columns to just the name and age**

**WHERE – ON THE BASIS OF A COLUMN AND THE VALUE IT HOLDS IN EVERY ROW, I’M PUTTING A CONDITION WHETHER OR NOT TO SHOW IT.**

**WHERE AGE >= 18**

**#FILTER SO THAT I CAN SEE FIRSTLY WHAT I WANT TO SEE**

**#SECONDLY HOW I WANT TO SEE IT.**

**WHAT YOU WANT TO SEE IS THE DISHES YOU SELECT IN A 3 COURSE MEAL, HOW YOU WANT TO SEE IT IS THE ORDER YOU GET YOUR FOOD IN APPETIZER, MAIN COURSE, DESSERT.**

**I want displayed the name, age, and the order amount of the customers of the TOP 4 CUSTOMERS WITH THE BIGGEST ORDER AMOUNT**

**Select name, age, amount from customers**

**Inner join orders on customers.customer\_id = orders.customer\_id ORDER BY amount desc LIMIT 4;**

**ORDER BY**

**REMEMBER WHAT YOU WANT TO SEE**

**AND HOW YOU WANT TO SEE IT!**

**-- Create Customers Table**

**CREATE TABLE Customers (**

**customer\_id INT AUTO\_INCREMENT PRIMARY KEY,**

**name VARCHAR(100) NOT NULL,**

**email VARCHAR(100) UNIQUE NOT NULL,**

**phone VARCHAR(20),**

**address TEXT**

**);**

**-- Create Products Table**

**CREATE TABLE Products (**

**product\_id INT AUTO\_INCREMENT PRIMARY KEY,**

**name VARCHAR(100) NOT NULL,**

**description TEXT,**

**price DECIMAL(10,2) NOT NULL,**

**stock\_quantity INT NOT NULL**

**);**

**-- Create Orders Table**

**CREATE TABLE Orders (**

**order\_id INT AUTO\_INCREMENT PRIMARY KEY,**

**customer\_id INT,**

**order\_date DATETIME DEFAULT CURRENT\_TIMESTAMP,**

**total\_amount DECIMAL(10,2),**

**FOREIGN KEY (customer\_id) REFERENCES Customers(customer\_id) ON DELETE CASCADE**

**);**

**-- Create Order Details Table**

**CREATE TABLE OrderDetails (**

**order\_detail\_id INT AUTO\_INCREMENT PRIMARY KEY,**

**order\_id INT,**

**product\_id INT,**

**quantity INT NOT NULL,**

**price DECIMAL(10,2) NOT NULL,**

**subtotal DECIMAL(10,2) GENERATED ALWAYS AS (quantity \* price) STORED,**

**FOREIGN KEY (order\_id) REFERENCES Orders(order\_id) ON DELETE CASCADE,**

**FOREIGN KEY (product\_id) REFERENCES Products(product\_id) ON DELETE CASCADE**

**);**

**-- Insert Sample Data into Customers**

**INSERT INTO Customers (name, email, phone, address) VALUES**

**('John Doe', 'johndoe@example.com', '1234567890', '123 Main St, City A'),**

**('Jane Smith', 'janesmith@example.com', '0987654321', '456 Elm St, City B');**

**-- Insert Sample Data into Products**

**INSERT INTO Products (name, description, price, stock\_quantity) VALUES**

**('Laptop', 'High-end gaming laptop', 1200.99, 10),**

**('Smartphone', 'Latest smartphone with great features', 799.50, 20),**

**('Headphones', 'Noise-canceling over-ear headphones', 150.75, 30);**

**-- Insert Sample Data into Orders**

**INSERT INTO Orders (customer\_id, total\_amount) VALUES**

**(1, 2000.49),**

**(2, 950.75);**

**-- Insert Sample Data into OrderDetails**

**INSERT INTO OrderDetails (order\_id, product\_id, quantity, price) VALUES**

**(1, 1, 1, 1200.99),**

**(1, 3, 2, 150.75),**

**(2, 2, 1, 799.50),**

**(2, 3, 1, 150.75);**

**-- Insert More Data into Customers**

**INSERT INTO Customers (name, email, phone, address) VALUES**

**('Alice Johnson', 'alicej@example.com', '1112223333', '789 Pine St, City C'),**

**('Bob Brown', 'bobb@example.com', '4445556666', '321 Oak St, City D'),**

**('Charlie White', 'charliew@example.com', '7778889999', '654 Maple St, City E'),**

**('David Black', 'davidb@example.com', '1011121314', '987 Birch St, City F');**

**-- Insert More Data into Products**

**INSERT INTO Products (name, description, price, stock\_quantity) VALUES**

**('Tablet', 'Lightweight tablet with long battery life', 450.00, 25),**

**('Smartwatch', 'Feature-packed smartwatch with health tracking', 250.25, 15),**

**('Gaming Mouse', 'Ergonomic gaming mouse with RGB lighting', 75.99, 50),**

**('Mechanical Keyboard', 'Mechanical keyboard with customizable keys', 120.50, 40),**

**('Monitor', '4K Ultra HD Monitor', 299.99, 20);**

**-- Insert More Data into Orders**

**INSERT INTO Orders (customer\_id, total\_amount) VALUES**

**(3, 950.99),**

**(4, 575.50),**

**(5, 1300.75),**

**(6, 620.25),**

**(1, 1200.99),**

**(2, 299.99);**

**-- Insert More Data into OrderDetails**

**INSERT INTO OrderDetails (order\_id, product\_id, quantity, price) VALUES**

**(3, 4, 2, 450.00), -- 2 Tablets**

**(3, 5, 1, 250.25), -- 1 Smartwatch**

**(4, 6, 3, 75.99), -- 3 Gaming Mice**

**(4, 7, 2, 120.50), -- 2 Mechanical Keyboards**

**(5, 1, 1, 1200.99), -- 1 Laptop**

**(5, 8, 2, 299.99), -- 2 Monitors**

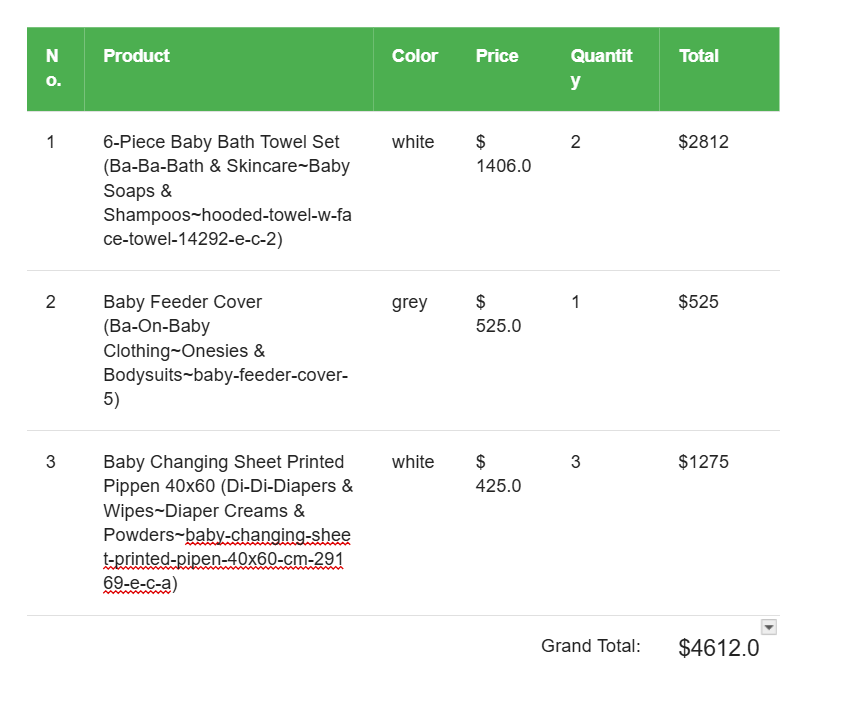
**(6, 2, 1, 799.50), -- 1 Smartphone**

**(6, 3, 1, 150.75), -- 1 Headphones**

**(6, 4, 1, 450.00), -- 1 Tablet**

**(1, 5, 2, 250.25), -- 2 Smartwatches**

**(2, 6, 3, 75.99); -- 3 Gaming Mice**

****

**Senior Data Engineer**

**job\_title LIKE ‘%Data%’ #CONTAINS THE WORD DATA**

**Senior Database Administrator #YES Includes checking against Data with:**

**Seni**

**enio**

**nior**

**ior**

**or D**

**\*until\***

**Data**

**Senior Datbase Administrator**

**#FALSE - The decision is to not include this row in the result, continues until the data ends, checking each possibility of 4 characters to match Data**

**The LIKE operator is used for pattern matching in SQL when searching for specific text values in a column.  
 It supports wildcards:**

* **% → Represents zero or more characters.**
* **\_ → Represents a single character.**

**—-----------------------------------------**

**CREATE TABLE SALES (**

**sale\_id INT PRIMARY KEY AUTO\_INCREMENT,**

**customer\_name VARCHAR(50),**

**product VARCHAR(50),**

**category VARCHAR(50),**

**amount DECIMAL(10,2),**

**sale\_date DATE**

**);**

**INSERT INTO SALES (customer\_name, product, category, amount, sale\_date) VALUES**

**('Alice', 'Laptop', 'Electronics', 1200, '2024-03-01'),**

**('Bob', 'TV', 'Electronics', 800, '2024-03-02'),**

**('Charlie', 'Phone', 'Electronics', 600, '2024-03-03'),**

**('Alice', 'Washing Machine', 'Home Appliances', 500, '2024-03-04'),**

**('David', 'Refrigerator', 'Home Appliances', 900, '2024-03-05'),**

**('Eva', 'Headphones', 'Electronics', 200, '2024-03-06'),**

**('Bob', 'Microwave', 'Home Appliances', 300, '2024-03-07'),**

**('Charlie', 'Tablet', 'Electronics', 400, '2024-03-08');**

Show Total Sales per Category, Ordered by Highest Sales

– First I see what columns I need, that is the Category, and then the second column will cater to the sum (aggregate function in this case) of the amount column named “total\_sales\_per\_category”.

Part of query: select category, sum(amount)as total\_sales\_per\_category from SALES;

– Then, I need to make sure the sum shows in the column ACCORDING or NEXT TO or GROUPED BY each category.

Part of query: select category, sum(amount) as total\_sales\_per\_category from SALES

Group By Category;

– Finally, the question is now asking to take this result and then ORDER IT by what criteria? The first row shown should be according to the new column made in this case the SUM of the AMOUNT Grouped by EACH Category! So I need to take the NAME OF THE NEW COLUMN HYPOTHETICAL I CREATED and then MAKE IT ORDER BY DESCENDING!

Final query: select category, sum(amount) as total\_sales\_per\_category from SALES

Group By Category

Order by total\_sales\_per\_category DESC;

-- Get only categories where total sales > 1500

I need the columns “category” & amount. Since I need total sales, I will select SUM(amount) AS “total\_sales”.

I need to group them, so since the priority is the category, I will GROUP BY category.

Total\_sales must exceed 1,500, so I will ensure it is HAVING total\_sales > 1500.

– SUB QUERIES:

CREATE DATABASE CompanyDB;

USE CompanyDB;

CREATE TABLE departments (

department\_id INT PRIMARY KEY AUTO\_INCREMENT,

department\_name VARCHAR(50)

);

CREATE TABLE employees (

employee\_id INT PRIMARY KEY AUTO\_INCREMENT,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

salary DECIMAL(10,2),

department\_id INT,

FOREIGN KEY (department\_id) REFERENCES departments(department\_id)

);

CREATE TABLE customers (

customer\_id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100)

);

CREATE TABLE products (

product\_id INT PRIMARY KEY AUTO\_INCREMENT,

product\_name VARCHAR(100),

category VARCHAR(50),

price DECIMAL(10,2)

);

CREATE TABLE orders (

order\_id INT PRIMARY KEY AUTO\_INCREMENT,

customer\_id INT,

total\_amount DECIMAL(10,2),

FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id)

);

-- Insert Departments

INSERT INTO departments (department\_name) VALUES

('HR'), ('IT'), ('Sales'), ('Marketing');

-- Insert Employees

INSERT INTO employees (first\_name, last\_name, salary, department\_id) VALUES

('Alice', 'Johnson', 60000, 2),

('Bob', 'Smith', 50000, 1),

('Charlie', 'Brown', 70000, 2),

('David', 'Williams', 55000, 3),

('Eve', 'Davis', 80000, 2),

('Frank', 'Miller', 45000, 3);

-- Insert Customers

INSERT INTO customers (name) VALUES

('John Doe'), ('Jane Smith'), ('Michael Johnson'), ('Emily Davis');

-- Insert Products

INSERT INTO products (product\_name, category, price) VALUES

('Laptop', 'Electronics', 1200),

('Phone', 'Electronics', 800),

('Desk Chair', 'Furniture', 150),

('Monitor', 'Electronics', 300),

('Table', 'Furniture', 250);

-- Insert Orders

INSERT INTO orders (customer\_id, total\_amount) VALUES

(1, 1500), (2, 800), (3, 2200), (4, 500), (2, 1200);

**Subquery** is cleaner if you only need to filter by condition and need the result/columns from only one table.

**Join** is better if you need department details (e.g., department name) beyond the employees, hence where the data from two tables is shown together.

### **Numeric Data Types**

### **1 0 0 0 0 0 0 0 // 2 to the power 7 is 128 possible figures**

The last number on the left demarcates the sign of the number.

128 positive combinations means from 0 to 127.

128 negative combinations -1 to -128.

| **Type** | **Description** |
| --- | --- |
| TINYINT | Very small integer (-128 to 127 or 0 to 255) 1 BYTE - 256 COMBINATIONS |
| SMALLINT | Small integer (-32,768 to 32,767)  32,768 Combinations positive 0 to 32,767  32,768 combinations negative -1 to - 32,768  TOTAL Combinations 2 BYTES 2 to the power 16 (16 bits) - 65536 |
| MEDIUMINT | Medium-sized integer 3 BYTES |
| INT / INTEGER | Standard integer (-2^31 to 2^31-1)  SPACE OF 4 BYTES RESERVED PER DATA ENTRY IN THE ROW PER COLUMN  Positive combinations 0 to 2^31-1  Negative combinations -1 to -2^31  Total combinations 2 to the power 32, 32 bits meaning 4 bytes reserved!  INT 216 - 00000000 00000000 00000000 11011000 |
| BIGINT | Large integer (-2^63 to 2^63-1) 5 BYTES |
| DECIMAL(m,d) | Fixed-point, for exact values like money |
| FLOAT(p) | Single-precision floating point |
| DOUBLE | Double-precision floating point |
| BIT(n) | Bit field (used for binary values) |

### **📅 Date and Time Data Types**

| **Type** | **Description** |
| --- | --- |
| DATE | YYYY-MM-DD |
| DATETIME | YYYY-MM-DD HH:MM:SS |
| TIMESTAMP | Same as DATETIME, auto updates |
| TIME | HH:MM:SS |
| YEAR | Year in 4-digit format (e.g., 2025) |

### **🔤 String Data Types**

| **Type** | **Description** |
| --- | --- |
| CHAR(n) | Fixed-length string (0 to 255 characters) |
| VARCHAR(n) | Variable-length string (0 to 65,535 bytes total) |
| TEXT types | Large texts (see below) |
| TINYTEXT | Max 255 bytes |
| TEXT | Max 65,535 bytes |
| MEDIUMTEXT | Max 16,777,215 bytes |
| LONGTEXT | Max 4,294,967,295 bytes |
| ENUM | One value from a predefined list |
| SET | Zero or more values from a predefined list |

### **🧱 Other Data Types**

| **Type** | **Description** |
| --- | --- |
| BLOB | Binary Large Object (for storing binary data like images/files) |
| JSON | Stores and validates JSON documents |
| GEOMETRY | Spatial data type (used in GIS apps) |

### **🔢 Numeric Data Types**

| **Data Type** | **Example** | **Description** |
| --- | --- | --- |
| TINYINT | age TINYINT  age = 23 | Good for small numbers like age |
| SMALLINT | year\_of\_joining SMALLINT  = 2020 | Slightly larger range than TINYINT |
| MEDIUMINT | views MEDIUMINT  = 500000 | Medium-range numbers like view counts |
| INT / INTEGER | id INT  = 101 | Common for primary keys or IDs |
| BIGINT | population BIGINT  = 8000000000 | For very large numbers |
| DECIMAL(10,2) | salary DECIMAL(10,2)  = 25000.75 | Accurate currency or price values |
| FLOAT(5,2) | rating FLOAT(5,2)  = 4.75 | Approximate decimals |
| DOUBLE | pi DOUBLE  = 3.1415926535 | More precision than FLOAT |
| BIT(1) | is\_active BIT(1)  = b'1' | 0 or 1 — used for boolean logic |

### **📅 Date and Time Data Types**

| **Data Type** | **Example** | **Description** |
| --- | --- | --- |
| DATE | birth\_date DATE  = '1990-05-15' | Just date |
| DATETIME | created\_at DATETIME  = '2025-04-20 14:30:00' | Full timestamp |
| TIMESTAMP | updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP | Auto-stores current time |
| TIME | duration TIME  = '01:30:00' | Time only (1 hour 30 minutes) |
| YEAR | graduation YEAR  = 2023 | Stores only the year |

### **🔤 String Data Types**

| **Data Type** | **Example** | **Description** |
| --- | --- | --- |
| CHAR(5) | country\_code CHAR(5)  = 'US' | Fixed length, pads short values |
| VARCHAR(100) | email VARCHAR(100)  = 'test@example.com' | Flexible string |
| TINYTEXT | short\_bio TINYTEXT  = 'Hi!' | Small pieces of text |
| TEXT | description TEXT  = 'This is a full product description.' | For longer text content |
| MEDIUMTEXT | article MEDIUMTEXT  = '<entire blog post>' | Medium-long text |
| LONGTEXT | book LONGTEXT  = '<entire novel>' | Very long text |
| ENUM | status ENUM('active','inactive','banned')  = 'active' | Only one from a predefined list |
| SET | roles SET('admin','editor','viewer')  = 'admin,viewer' | Multiple values from a list |

### **🧱 Other Data Types**

| **Data Type** | **Example** | **Description** |
| --- | --- | --- |
| BLOB | profile\_picture BLOB  = (image binary data) | Stores binary files like images |
| JSON | preferences JSON  = '{"theme":"dark","fontSize":14}' | Stores structured JSON data |
| GEOMETRY | location GEOMETRY  = ST\_GeomFromText('POINT(10 20)') | For geographic coordinates |

If light == green: yellow

Go ahead

Else If light == red:

Dont go ahead unless u like jail

Else If light == yellow:

Then be careful, dont be a dirty fellow.

Else you might get squashed like a marsh mellow.

Else:

Call 911, the lights are out, or rainbow dancing, or showing two at once or all 3 at once!

1 If there are salads, get salads.

If there are sandwiches, get sandwiches.

2 If there are salads, get salads.

Else if there are sandwiches, get sandwiches.

light = input('Please provide the colour of the traffic signal')

if light == 'red':

print('Wait for it to turn green, dont go forward')

elif light == 'green':

print('Go forward')

elif light == 'yellow':

print('Go when the police is not there')

else:

print('Call 911, report the signal which is not working, be a good citizen!')

#One scenario - all 3 lights are on

#Another scenario - Signal is out/all black

#Another scenario - Signal is giving rainbow colours

print('End of program')

CREATE TABLE comic\_books (

Id int PRIMARY KEY,

Author VARCHAR(50),

Title VARCHAR(100),

Publisher VARCHAR(255),

Release\_year int,

Price Decimal(5,2)

)

INSERT INTO comic\_books (Id, Author, Title, Publisher, Release\_year, Price)

VALUES

(1, ‘Stan Lee’, ‘Spider Man : Homecoming’, ‘Marvel Comics’, 2017, 14.99) ,

(2, ‘Alan Moore’ , ‘Batman: The Killing Joke’ , ‘DC Comics’ , 1988, 19.99),

(3, ‘Alan Scott’, ‘Green Lantern’, ‘DC Comics’, 1970, 7.99)

—-

A Git repository (repo) is a storage space where all the files, directories, and their version history for a project are tracked and managed. It serves as the backbone of Git's version control system, enabling developers to collaborate, track changes, and maintain a history of modifications over time.