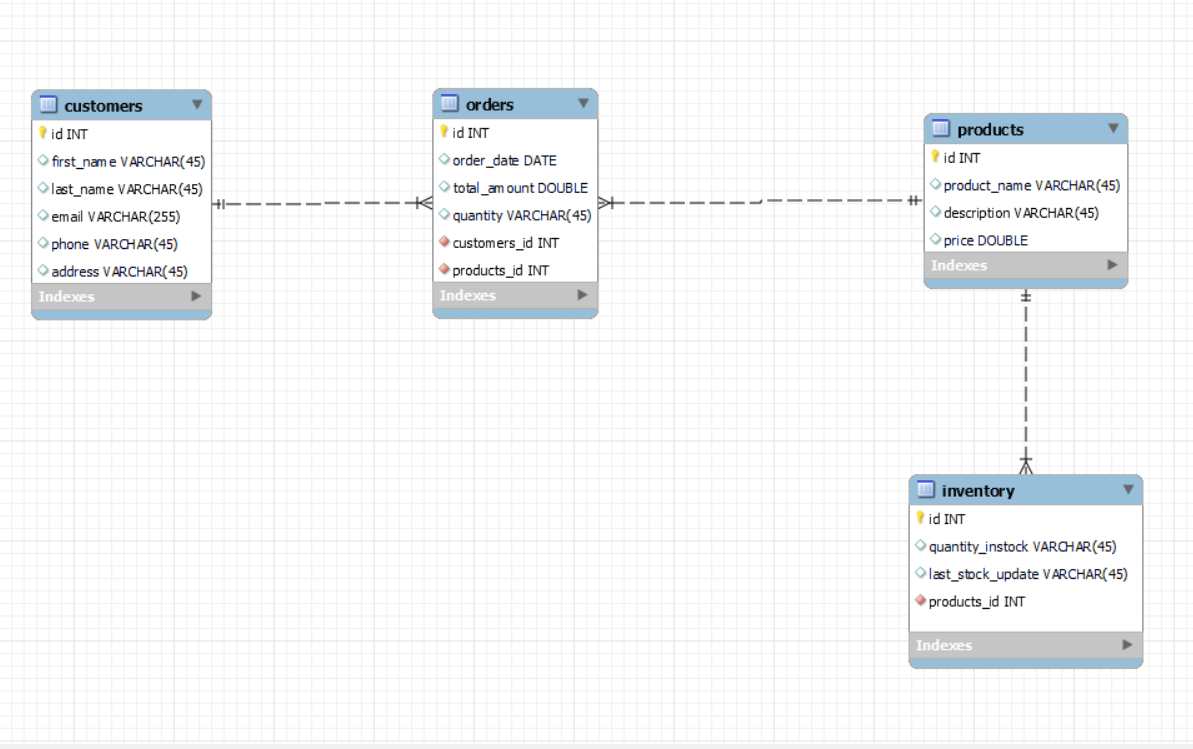
TechShop, an electronic gadgets shop



-- MySQL Workbench Forward Engineering

-- -----------------------------------------------------

-- Schema tech\_shop

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema tech\_shop

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `tech\_shop` DEFAULT CHARACTER SET utf8 ;

USE `tech\_shop` ;

-- -----------------------------------------------------

-- Table `tech\_shop`.`customers`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `tech\_shop`.`customers` (

`id` INT NOT NULL AUTO\_INCREMENT,

`first\_name` VARCHAR(45) NULL,

`last\_name` VARCHAR(45) NULL,

`email` VARCHAR(255) NULL,

`phone` VARCHAR(45) NULL,

`address` VARCHAR(45) NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `tech\_shop`.`products`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `tech\_shop`.`products` (

`id` INT NOT NULL AUTO\_INCREMENT,

`product\_name` VARCHAR(45) NULL,

`description` VARCHAR(45) NULL,

`price` DOUBLE NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `tech\_shop`.`orders`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `tech\_shop`.`orders` (

`id` INT NOT NULL AUTO\_INCREMENT,

`order\_date` DATE NULL,

`total\_amount` DOUBLE NULL,

`quantity` VARCHAR(45) NULL,

`customers\_id` INT NOT NULL,

`products\_id` INT NOT NULL,

PRIMARY KEY (`id`),

INDEX `fk\_orders\_customers\_idx` (`customers\_id` ASC) ,

INDEX `fk\_orders\_products1\_idx` (`products\_id` ASC) ,

CONSTRAINT `fk\_orders\_customers`

FOREIGN KEY (`customers\_id`)

REFERENCES `tech\_shop`.`customers` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_orders\_products1`

FOREIGN KEY (`products\_id`)

REFERENCES `tech\_shop`.`products` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `tech\_shop`.`inventory`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `tech\_shop`.`inventory` (

`id` INT NOT NULL AUTO\_INCREMENT,

`quantity\_instock` VARCHAR(45) NULL,

`last\_stock\_update` VARCHAR(45) NULL,

`products\_id` INT NOT NULL,

PRIMARY KEY (`id`),

INDEX `fk\_inventory\_products1\_idx` (`products\_id` ASC) ,

CONSTRAINT `fk\_inventory\_products1`

FOREIGN KEY (`products\_id`)

REFERENCES `tech\_shop`.`products` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

#insertions

insert into customers (first\_name, last\_name, email, phone, address)

values

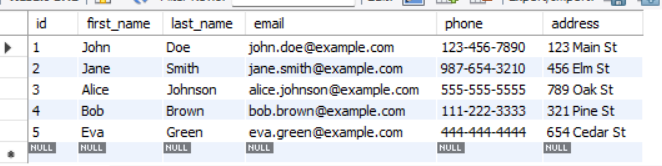
('john', 'doe', 'john.doe@example.com', '123-456-7890', '123 main st'),

('jane', 'smith', 'jane.smith@example.com', '987-654-3210', '456 elm st'),

('alice', 'johnson', 'alice.johnson@example.com', '555-555-5555', '789 oak st'),

('bob', 'brown', 'bob.brown@example.com', '111-222-3333', '321 pine st'),

('eva', 'green', 'eva.green@example.com', '444-444-4444', '654 cedar st');



insert into products (product\_name, description, price)

values

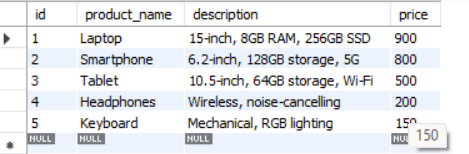
('laptop', '15-inch, 8gb ram, 256gb ssd', 900),

('smartphone', '6.2-inch, 128gb storage, 5g', 800),

('tablet', '10.5-inch, 64gb storage, wi-fi', 500),

('headphones', 'wireless, noise-cancelling', 200),

('keyboard', 'mechanical, rgb lighting', 150);



insert into orders (order\_date, total\_amount, quantity, customers\_id, products\_id)

values

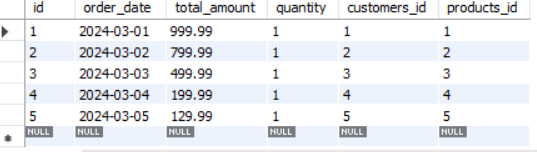
('2024-03-01', 999.99, '1', 1, 1),

('2024-03-02', 799.99, '1', 2, 2),

('2024-03-03', 499.99, '1', 3, 3),

('2024-03-04', 199.99, '1', 4, 4),

('2024-03-05', 129.99, '1', 5, 5);



insert into inventory (quantity\_instock, last\_stock\_update, products\_id)

values

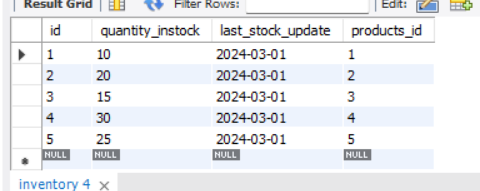
('10', '2024-03-01', 1),

('20', '2024-03-01', 2),

('15', '2024-03-01', 3),

('30', '2024-03-01', 4),

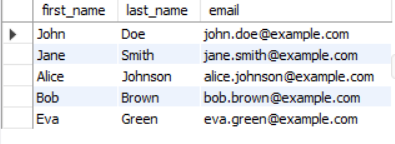
('25', '2024-03-01', 5);



tasks 2: select, where, between, and, like:

1. write an sql query to retrieve the names and emails of all customers.

select first\_name, last\_name, email from customers;

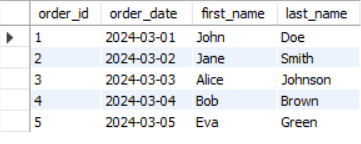


1. write an sql query to list all orders with their order dates and corresponding customer names.

select o.id as order\_id, o.order\_date, c.first\_name, c.last\_name

from orders o

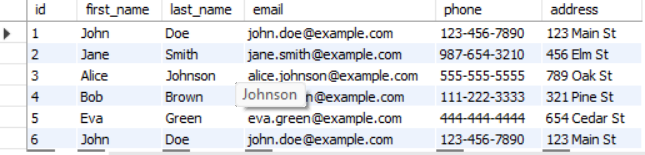
join customers c on o.customers\_id = c.id;



1. write an sql query to insert a new customer record into the "customers" table. include customer information such as name, email, and address.

insert into customers (first\_name, last\_name, email, phone, address)

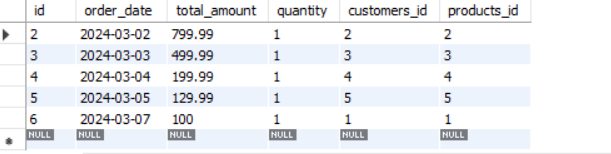
values ('john', 'doe', 'john.doe@example.com', '123-456-7890', '123 main st');



1. write an sql query to update the prices of all electronic gadgets in the "products" table by increasing them by 10%.
2. write an sql query to delete a specific order and its associated order details from the "orders" and "orderdetails" tables. allow users to input the order id as a parameter.
3. write an sql query to insert a new order into the "orders" table. include the customer id, order date, and any other necessary information.

insert into orders (order\_date, total\_amount, quantity, customers\_id, products\_id)

values ('2024-03-07', 100.00, '1', 1, 1);

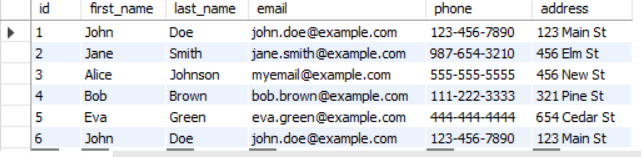


1. write an sql query to update the contact information (e.g., email and address) of a specific customer in the "customers" table. allow users to input the customer id and new contact information.

update customers

set email = 'myemail@example.com', address = '456 new st'

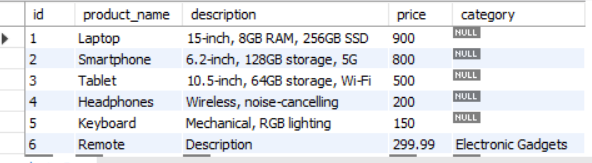
where id = 3;



10. write an sql query to insert a new electronic gadget product into the "products" table, including product name, category, price, and any other relevant details.

insert into products (product\_name, description, price, category)

values ('new electronic gadget', 'description', 299.99, 'electronic gadgets');

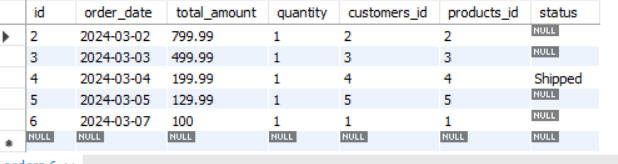


11. write an sql query to update the status of a specific order in the "orders" table (e.g., from "pending" to "shipped"). allow users to input the order id and the new status.

update orders

set status = 'shipped'

where id = <order\_id>;



12. write an sql query to calculate and update the number of orders placed by each customer in the "customers" table based on the data in the "orders" table.

update customers c

set num\_orders = (select count(\*) from orders o where o.customers\_id = c.id );

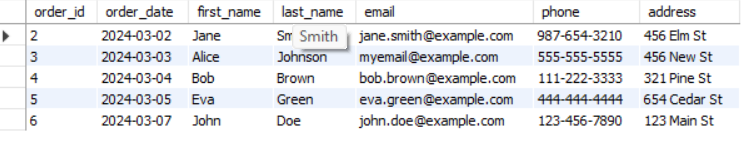
task 3. aggregate functions, having, order by, groupby and joins:

1. write an sql query to retrieve a list of all orders along with customer information (e.g., customer name) for each order.

select o.id as order\_id, o.order\_date, c.first\_name, c.last\_name, c.email, c.phone, c.address

from orders o

join customers c on o.customers\_id = c.id;



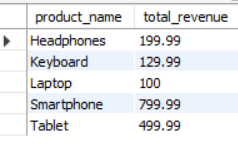
1. write an sql query to find the total revenue generated by each electronic gadget product. include the product name and the total revenue

select p.product\_name, sum(o.total\_amount) as total\_revenue

from orders o

join products p on o.products\_id = p.id

group by p.product\_name;



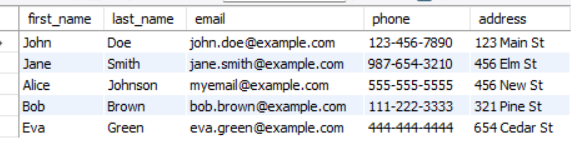
1. write an sql query to list all customers who have made at least one purchase. include their names and contact information.

select c.first\_name, c.last\_name, c.email, c.phone, c.address

from customers c

join orders o on c.id = o.customers\_id

group by c.id;



1. write an sql query to find the most popular electronic gadget, which is the one with the highest total quantity ordered. include the product name and the total quantity ordered.

select p.product\_name, sum(o.quantity) as total\_quantity\_ordered

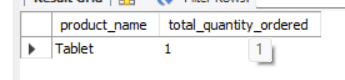
from orders o

join products p on o.products\_id = p.id

group by p.product\_name

order by total\_quantity\_ordered desc

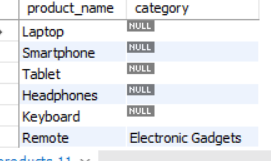
limit 1;



1. write an sql query to retrieve a list of electronic gadgets along with their corresponding categories.

select p.product\_name, p.category

from products p;



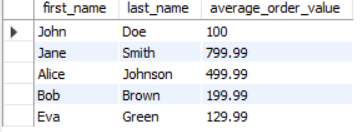
1. write an sql query to calculate the average order value for each customer. include the customer's name and their average order value.

select c.first\_name, c.last\_name, avg(o.total\_amount) as average\_order\_value

from orders o

join customers c on o.customers\_id = c.id

group by c.id;



1. write an sql query to find the order with the highest total revenue. include the order id, customer information, and the total revenue.

select o.id as order\_id, c.first\_name, c.last\_name, c.email, c.phone, c.address, sum(o.total\_amount) as total\_revenue

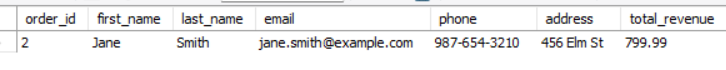
from orders o

join customers c on o.customers\_id = c.id

group by o.id

order by total\_revenue desc

limit 1;



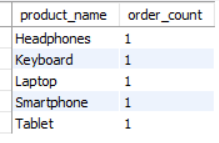
1. write an sql query to list electronic gadgets and the number of times each product has been ordered.

select p.product\_name, count(o.id) as order\_count

from orders o

join products p on o.products\_id = p.id

group by p.product\_name;



1. write an sql query to find customers who have purchased a specific electronic gadget product. allow users to input the product name as a parameter.

select distinct c.first\_name, c.last\_name, c.email, c.phone, c.address

from customers c

join orders o on c.id = o.customers\_id

join products p on o.products\_id = p.id

where p.product\_name = 'tablet';

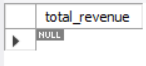


1. write an sql query to calculate the total revenue generated by all orders placed within a specific time period. allow users to input the start and end dates as parameters.

select sum(total\_amount) as total\_revenue

from orders

where order\_date between 'start\_date' and 'end\_date';



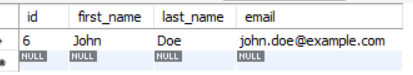
task 4. subquery and its type:

1. write an sql query to find out which customers have not placed any orders.

select id, first\_name, last\_name, email

from customers

where id not in (select distinct customers\_id from orders);



1. write an sql query to find the total number of products available for sale.

select count(\*) as total\_products

from products;



1. write an sql query to calculate the total revenue generated by techshop.

select sum(total\_amount) as total\_revenue

from orders;



1. write an sql query to calculate the average quantity ordered for products in a specific category. allow users to input the category name as a parameter.
2. write an sql query to calculate the total revenue generated by a specific customer. allow users to input the customer id as a parameter.

select sum(total\_amount) as total\_revenue

from orders

where customers\_id = 1;



1. write an sql query to find the customers who have placed the most orders. list their names and the number of orders they've placed.

select c.first\_name, c.last\_name, count(\*) as order\_count

from customers c

join orders o on c.id = o.customers\_id

group by c.id

order by order\_count desc

limit 1;



1. write an sql query to find the most popular product category, which is the one with the highest total quantity ordered across all orders.

select p.category, sum(od.quantity) as total\_quantity\_ordered

from orderdetails od

join products p on od.products\_id = p.id

group by p.category

order by total\_quantity\_ordered desc

limit 1;

1. write an sql query to find the customer who has spent the most money (highest total revenue) on electronic gadgets. list their name and total spending.

select c.first\_name, c.last\_name, sum(od.quantity \* p.price) as total\_spending

from customers c

join orders o on c.id = o.customers\_id

join orderdetails od on o.id = od.order\_id

join products p on od.products\_id = p.id

where p.category = 'electronic gadgets'

group by c.id

order by total\_spending desc

limit 1;

1. write an sql query to calculate the average order value (total revenue divided by the number of orders) for all customers.

select avg(total\_amount) as avg\_order\_value

from orders;



10. write an sql query to find the total number of orders placed by each customer and list their names along with the order count

select c.first\_name, c.last\_name, count(\*) as order\_count

from customers c

join orders o on c.id = o.customers\_id

group by c.id;

