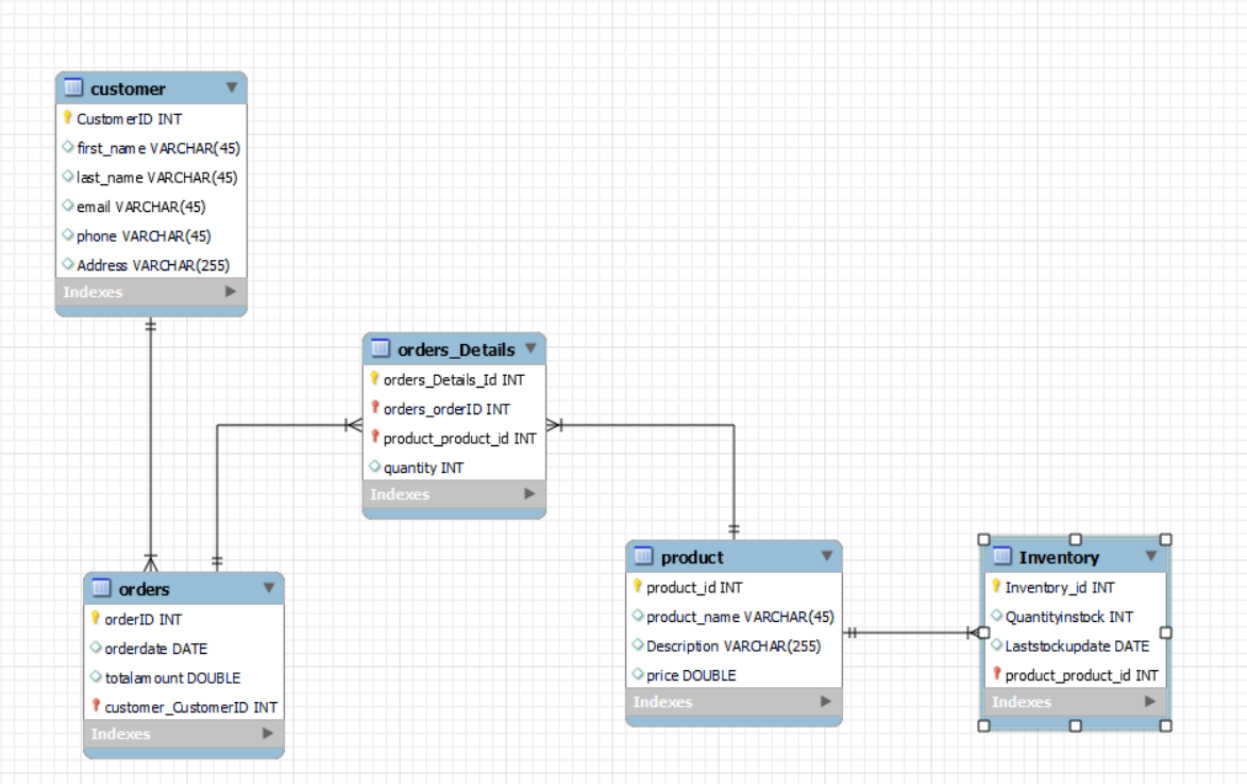
Electronic gadgets Tech shop case study

ER DIAGRAM :



#db scripts

-- MySQL Workbench Forward Engineering

-- ---
-- Schema Techshop
-- ---

-- ---
-- Schema Techshop
-- ---
CREATE SCHEMA IF NOT EXISTS `Techshop` DEFAULT CHARACTER SET utf8 ;
USE `Techshop` ;

-- ---
-- Table `Techshop`.`customer`
-- ---

```
CREATE TABLE IF NOT EXISTS `Techshop`.`customer` (
  `CustomerID` INT NOT NULL AUTO_INCREMENT,
  `first_name` VARCHAR(45) NULL,
  `last_name` VARCHAR(45) NULL,
  `email` VARCHAR(45) NULL,
  `phone` VARCHAR(45) NULL,
  `Address` VARCHAR(255) NULL,
  PRIMARY KEY (`CustomerID`))
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `Techshop`.`product`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Techshop`.`product` (
  `product_id` INT NOT NULL AUTO_INCREMENT,
  `product_name` VARCHAR(45) NULL,
  `Description` VARCHAR(255) NULL,
  `price` DOUBLE NULL,
  PRIMARY KEY (`product_id`))
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `Techshop`.`orders`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Techshop`.`orders` (
  `orderID` INT NOT NULL AUTO_INCREMENT,
  `orderdate` DATE NULL,
  `totalamount` DOUBLE NULL,
  `customer_CustomerID` INT NOT NULL,
  PRIMARY KEY (`orderID`, `customer_CustomerID`),
  INDEX `fk_order_customer1_idx` (`customer_CustomerID` ASC) ,
  CONSTRAINT `fk_order_customer1`
    FOREIGN KEY (`customer_CustomerID`)
    REFERENCES `Techshop`.`customer` (`CustomerID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `Techshop`.`Inventory`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Techshop`.`Inventory` (
  `Inventory_id` INT NOT NULL AUTO_INCREMENT,
  `Quantityinstock` INT NULL,
```

```
  `Laststockupdate` DATE NULL,
  `product_product_id` INT NOT NULL,
  PRIMARY KEY (`Inventory_id`, `product_product_id`),
  INDEX `fk_Inventory_product1_idx` (`product_product_id` ASC) ,
  CONSTRAINT `fk_Inventory_product1`
    FOREIGN KEY (`product_product_id`)
    REFERENCES `Techshop`.`product` (`product_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `Techshop`.`orderDetails`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Techshop`.`orderDetails` (
  `orderDetail_ID` INT NOT NULL AUTO_INCREMENT,
  `product_product_id` INT NOT NULL,
  `order_orderID` INT NOT NULL,
  `Quantity` INT NULL,
  PRIMARY KEY (`orderDetail_ID`, `product_product_id`, `order_orderID`),
  INDEX `fk_product_has_order_order1_idx` (`order_orderID` ASC) ,
  INDEX `fk_product_has_order_product_idx` (`product_product_id` ASC) ,
  CONSTRAINT `fk_product_has_order_product`
    FOREIGN KEY (`product_product_id`)
    REFERENCES `Techshop`.`product` (`product_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_product_has_order_order1`
    FOREIGN KEY (`order_orderID`)
    REFERENCES `Techshop`.`orders` (`orderID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Query Insertions

```
use techshop;

insert into customer( first_name ,last_name , email , phone , address )
values ('Nivetha ' , 'Thiyagarajan' , 'nive@gmail.com', 9876543, 'chennai'),
('Nithi', 'Dev', 'nithu@gmail.com ', 9867543, 'Mumbai'),
('ram', 'kumar',  'ram@gmail.com' , 98754567,'kerla'),
('visha', 'sam', 'visha@gmail.com' , 78889567, 'bangalore');
```

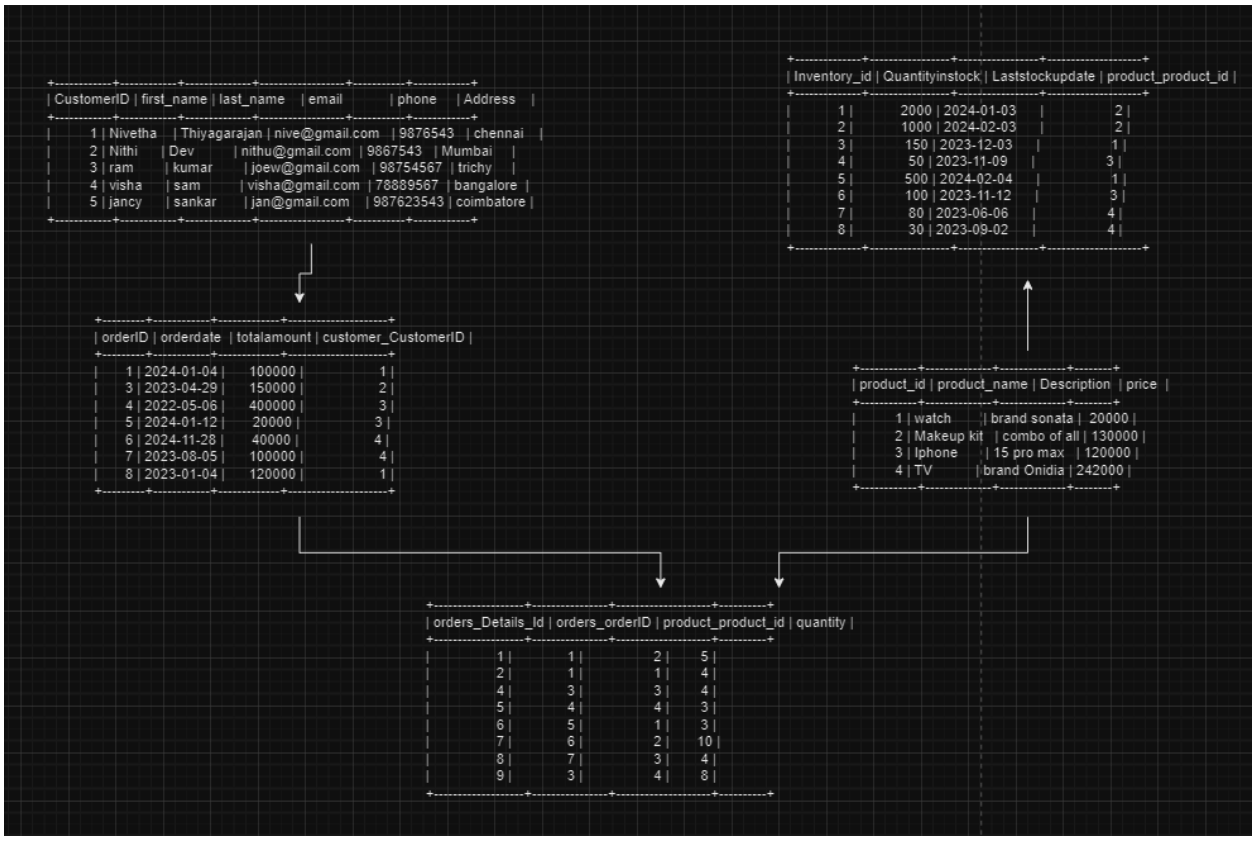
```
insert into orders (orderdate , totalamount , customer_Customerid)
values ( '2024-01-04',100000 ,1),
('2024-01-04',500000  ,2),
( '2023-04-29',150000 ,2),
( '2022-05-06',400000 ,3),
 ('2024-01-12',20000 ,3),
( '2024-11-28',40000 ,4),
( '2023-08-05',100000 ,4);

insert into product (product_name , Description , price)
values ('watch','brand sonata',20000),
('Makeup kit','combo of all',130000),
('Iphone','15 pro max ',120000),
('TV','brand Onidia',220000);

insert into inventory (Quantityinstock , Laststockupdate , product_product_id)
values ( 2000 , '2024-01-03' , 2),
(1000 , '2024-02-03' , 2),
( 150 , '2023-12-03' , 1),
( 50, '2023-11-09' , 3),
( 500 , '2024-02-04' , 1),
( 100 , '2023-11-12' , 3),
( 80 , '2023-06-06' , 4),
( 30 , '2023-09-02' , 4);

insert into orders_details (orders_orderID , product_product_id, quantity)
values (1, 2 , 5),
(1, 1 , 4),
(2 , 1 , 5),
(3 , 3, 4),
(4, 4, 3),
(5,1, 3),
(6,2, 10),
(7,3, 4),
(3,4, 8),
(2,3,9);
```

Reference Image:



TASK 2:

-- Tasks 2: Select, Where, Between, AND, LIKE:
-- 1. Write an SQL query to retrieve the names and emails of all customers.

```
select first_name , email from customer;
```

-- 2. Write an SQL query to list all orders with their order dates and corresponding customer names.

```
select orderid , orderdate , customer_customerid from orders;
```

-- 3. 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 customer( first_name ,last_name , email , phone , address )
values ('jancy' , 'sankar' , 'jan@gmail.com', 987623543, 'coimbatore');
```

-- 4. Write an SQL query to update the prices of all electronic gadgets in the "Products" table by increasing them by 10%.

```
update product set price = (price * 1.1) where product_id = 4;
```

-- 5. 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.

```
delete from Orders_Details where orders_orderid =2;
delete from Orders
where orderID = 2;
```

-- 6. 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 (orderdate , totalamount , customer_Customerid)
values ( '2023-01-04',120000 ,1);
```

-- 7. 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 customer set email = 'joew@gmail.com', address='trichy'  where customerid=3;
```

-- 8. Write an SQL query to recalculate and update the total cost of each
-- order in the "Orders" table based on the prices and quantities in the "OrderDetails" table
-- price is not present in the database so could not update

-- 9. Write an SQL query to delete all orders and their associated order details for a specific customer from the "Orders" and "OrderDetails" tables. Allow users to input the customer ID as a parameter.

```
delete from Orders_Details where orders_orderid =2;
delete from Orders
where customer_customer_id = 2;
```

-- 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 product (product_name , Description , price)
values ('watchclock','brand titan',500000);
```

-- 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.
-- status column is not present in the db

-- 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

– in customer table there is no column called number_of_ordes

TASK 3:

-- 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.orderid , c.first_name
from orders o join customer c on c.customerid = o.customer_customerid;
```

-- 2. Write an SQL query to find the total revenue generated by each electronic gadget product.
-- Include the product name and the total revenue.

```
select product_name , sum(price) as total_revenue
from product
group by product_id;
```

-- 3. 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.phone, count(o.orderid) as total_count
from customer c join orders o on c.customerid = o.customer_customerid
group by customerid
having total_count>1;
```

-- 4. 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 , od.quantity
from product p join orders_details od on p.product_id = od.product_product_id
where od.quantity = ( select max(quantity) from orders_details);
```

-- 5. Write an SQL query to retrieve a list of electronic gadgets along with their corresponding categories.
-- category column is not present here i am using description

```
select product_name ,description
from product;
```

-- 6. 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, avg(o.totalamount) as Average_amount
from customer c join orders o on o.customer_customerid =c.customerid
group by c.customerid;
```

-- 7. 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.orderid ,c.customerid , c.first_name , c.email , c.phone , max(o.totalamount) as
total_revenue
from orders o join customer c on o.customer_customerid =c.customerid
where o.totalamount=(select max(totalamount) from orders);
```

-- 8. Write an SQL query to list electronic gadgets and the number of times each product has been ordered.

```
select p.product_name , count(p.product_id) as No_of_times_ordered
from product p join orders_details od on p.product_id =od.product_product_id
group by product_id;
```

-- 9. 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 c.customerid , c.first_name , c.email , c.phone , p.product_name
from customer c join orders o on o.customer_customerid =c.customerid
join orders_details od on o.orderid=od.orders_orderid join
product p on p.product_id = od.product_product_id
where product_name ='watch';
```

-- 10. 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 orderid ,totalamount , orderdate
from orders where orderdate between  '2023-01-06' and '2024-09-01';
```

TASK 4:

-- task 4 subquery and its types

-- 1. Write an SQL query to find out which customers have not placed any orders.

```
select customerid , first_name
from customer where customerid not in ( select distinct customer_customerid from orders);
```

-- 2. Write an SQL query to find the total number of products available for sale.

```
select count(distinct product_id) as total_number_of_products
from product;
```

-- 3. Write an SQL query to calculate the total revenue generated by TechShop.

```
select sum( total_price) as tech_shop_total_revenue
from((select p.price*od.quantity as total_price from product p join orders_details od on
p.product_id = od.product_product_id)) as sub;
```

-- 4. 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.
-- here i used description instead of category . as category is not present

```
select avg(quantity) as Average_quantity
from orders_Details where product_product_id in (select product_id from product
where description='brand sonata');
```

-- 5. 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(totalamount ) as total_revenue
from orders where customer_customerid in ( select customerid from customer where
customerid=3);
```

-- 6. 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 first_name , max(total_count) as most_placed_count
from  ( select first_name ,customer_customerid  , count(orderid)  as total_count from orders o
join
customer c on c.customerid = o.customer_customerid
```

```
group by customer_customerid
) as subquery;
```

-- 7. Write an SQL query to find the most popular product category, which is the one with the highest total quantity ordered across all orders.
-- category is not present so i took product_name

```
 select product_product_id
from( select product_product_id , sum(quantity)  as highest_quantity
from orders_details group by product_product_id order by highest_quantity desc limit 0,1 ) as
subquery;
```

-- 8. 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, sum(p.price) as total_revenue
from customer c join orders o on o.customer_customerid =c.customerid
join orders_details od on o.orderid=od.orders_orderid join
product p on p.product_id = od.product_product_id
group by c.customerid
order by p.price desc
limit 0, 1;
```

-- 9. Write an SQL query to calculate the average order value (total revenue divided by the number of orders) for all customers.

```
select customer_customerid,(sum(totalamount) / count(orderid)) as average_order_value
from orders group by customer_customerid;
```

-- 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  first_name ,(select count(orderid) from orders o WHERE o.customer_customerid =
c.customerid) AS order_count
from customer c
group by c.customerid;
```