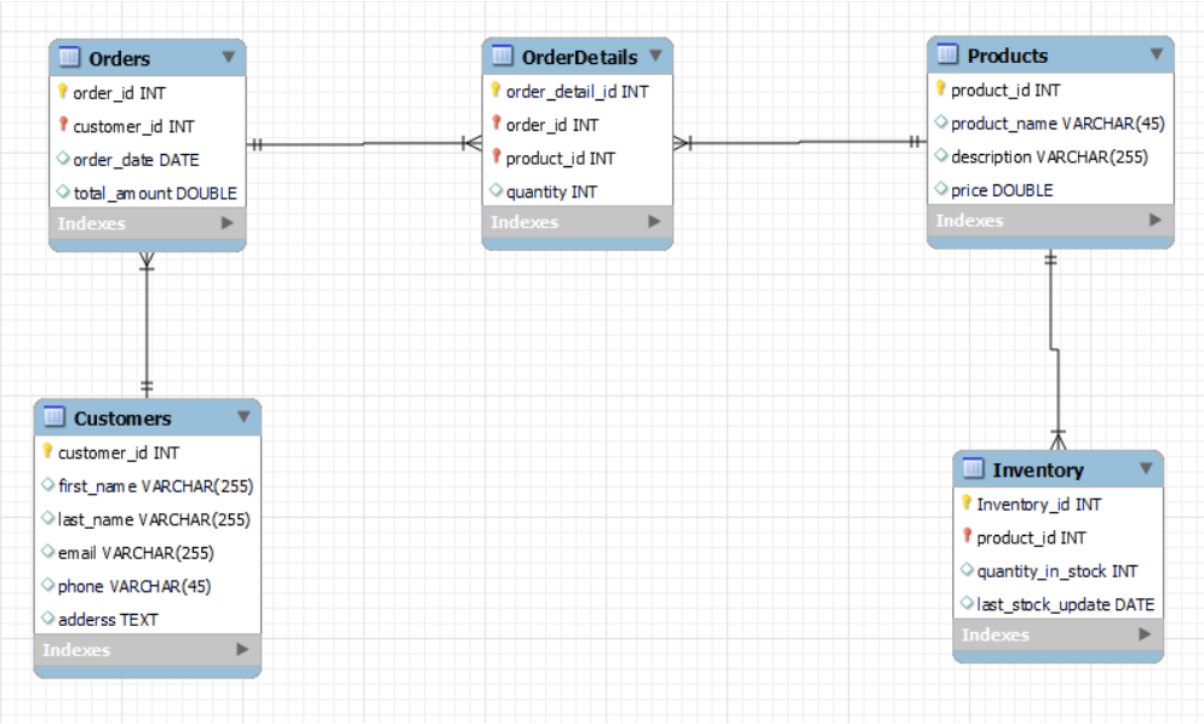
TechShop, an electronic gadgets shop

ER Diagram



TASK-1: Database Design:

-- 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` (
  `customer_id` INT NOT NULL AUTO_INCREMENT,
  `first_name` VARCHAR(255) NULL,
```

```
  `last_name` VARCHAR(255) NULL,
  `email` VARCHAR(255) NULL,
  `phone` VARCHAR(45) NULL,
  `adderss` TEXT NULL,
  PRIMARY KEY (`customer_id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `tech_shop`.`Products`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `tech_shop`.`Products` (
  `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 `tech_shop`.`Orders`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `tech_shop`.`Orders` (
  `order_id` INT NOT NULL AUTO_INCREMENT,
  `customer_id` INT NOT NULL,
  `order_date` DATE NULL,
  `total_amount` DOUBLE NULL,
  PRIMARY KEY (`order_id`, `customer_id`),
  INDEX `fk_Orders_Customers_idx` (`customer_id` ASC),
  CONSTRAINT `fk_Orders_Customers`
    FOREIGN KEY (`customer_id`)
    REFERENCES `tech_shop`.`Customers` (`customer_id`)
    ON DELETE NO ACTION
```

```
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `tech_shop`.`OrderDetails`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `tech_shop`.`OrderDetails` (
  `order_detail_id` INT NOT NULL AUTO_INCREMENT,
  `order_id` INT NOT NULL,
  `product_id` INT NOT NULL,
  `quantity` INT NULL,
  PRIMARY KEY (`order_detail_id`, `order_id`, `product_id`),
  INDEX `fk_OrderDetails_Orders1_idx` (`order_id` ASC),
  INDEX `fk_OrderDetails_Products1_idx` (`product_id` ASC),
  CONSTRAINT `fk_OrderDetails_Orders1`
    FOREIGN KEY (`order_id`)
    REFERENCES `tech_shop`.`Orders` (`order_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_OrderDetails_Products1`
    FOREIGN KEY (`product_id`)
    REFERENCES `tech_shop`.`Products` (`product_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


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

```
  `quantity_in_stock` INT NULL,
  `last_stock_update` DATE NULL,
  PRIMARY KEY (`Inventory_id`, `product_id`),
  INDEX `fk_Inventory_Products1_idx` (`product_id` ASC),
  CONSTRAINT `fk_Inventory_Products1`
    FOREIGN KEY (`product_id`)
    REFERENCES `tech_shop`.`Products` (`product_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

TASK-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;
```

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

```
select c.first_name,c.last_name,o.order_id,o.order_date
from customers c,orders o
where c.customer_id=o.customer_id;
```

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 customers (first_name, last_name, email, phone, adderss)
values
('Abishek', 'Roy', 'abishek.roy@gmail.com', '1234567890', '123 Main Street, City');
```

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

```
update products
set price= price * 1.10
where product_id is not null;
```

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 orderDetails
```

```
where order_id = @OrderID;

-- then delete from

delete from Orders

where order_id = @OrderID;
```

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 orderDetails (order_id, product_id, quantity)

values (1, 1, 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 customers

set email=@NewEmail , adderss= @NewAddress

where customer_id=@CusId;
```

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. //doubt

```
update orders

set total_amount = (

   select orderDetails.quantity * products.price

   from orderDetails,products

   where orderDetails.product_id = products.product_id

)

where order_id in (select distinct order_id FROM OrderDetails);
```

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 orderDetails

where order_id in(select order_id from orders where customer_id=@cusId);


delete from orders

where customer_id=@cusId;
```

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)
values
('Laptop', 'High-performance laptop', 1200);
```

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.*,c.first_name,c.last_name from
orders o,customers c where o.customer_id=c.customer_id;
```

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 p.product_name,sum(od.quantity*p.price) as Revenue_generated from
orderDetails od, products p where p.product_id=od.product_id
group by od.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.last_name,c.phone from
orders o,customers c
where o.customer_id=c.customer_id
group by o.customer_id
having count(o.order_id)>0;
```

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,sum(quantity) as total_quantity_ordered
from orderDetails od, products p
where od.product_id=p.product_id
group by od.product_id
order by total_quantity_ordered desc
limit 0,1;
```

5. Write an SQL query to retrieve a list of electronic gadgets along with their corresponding categories.

```
select product_name,description
```

from products;

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 customer_id, avg(total_amount) as Average_order_value from orders
group by customer_id;
```

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 c.first_name,c.last_name,o.order_id,total_amount
from customers c,orders o
where c.customer_id=o.customer_id
order by total_amount desc
limit 0,1;
```

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

```
select p.product_name,count(od.order_id) as Number_Of_Orders from
orderDetails od,products p
where od.product_id=p.product_id
group by od.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.

```
set @prodName := 'Laptop';
select c.first_name,p.product_name
from customers c,orders o,orderDetails od,products p
where c.customer_id=o.customer_id and o.order_id=od.order_id and od.product_id=p.product_id and
p.product_name = 'Laptop';
```

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 * from orders
where order_date between '2024-03-01' and '2024-03-31';
```

TASK-4. Subquery and its type:

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

```
select * from customers
where customer_id not in (select customer_id from orders);
```

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

```
select * from products
where product_id in (select product_id from inventory where quantity_in_stock is not null);
```

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

```
select sum(total_amount) as total_revenue_generated from orders;
```

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.

```
select Round(avg(quantity),2) as Average_quantity from orderDetails;
```

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 customer_id,sum(total_amount) as revenue_generated from orders
where customer_id=1
group by customer_id;
```

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
   concat(c.first_name, ' ', c.last_name) as full_name,c.email,c.phone, Number_Of_Orders
from customers c
join (
   select
      customer_id,
      count(order_id) as Number_Of_Orders
   from orders o
   group by customer_id
   order by Number_of_orders desc
   limit 0,1
```

```
) as customer_orders on c.customer_id = customer_orders.customer_id;
```

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.customer_id,concat(c.first_name,' ',c.last_name) as full_name, revenue_generated from
customers c
join (select customer_id,sum(total_amount) as revenue_generated
        from orders group by customer_id
    order by revenue_generated desc
    limit 1) as  subQuery on subQuery.customer_id=c.customer_id;
```

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

```
select c.customer_id,concat(c.first_name,' ',c.last_name) as full_name, average_order_value from
customers c
join (select customer_id,avg(total_amount) as average_order_value
        from orders group by customer_id) as  subQuery on subQuery.customer_id=c.customer_id;
```

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 concat(c.first_name,'  ',c.last_name) as customer_name , Total_orders_placed from customers c
join (select customer_id, count(order_id) as Total_orders_placed
        from orders
    group by customer_id) as subQuery on subQuery.customer_id=c.customer_id;
```