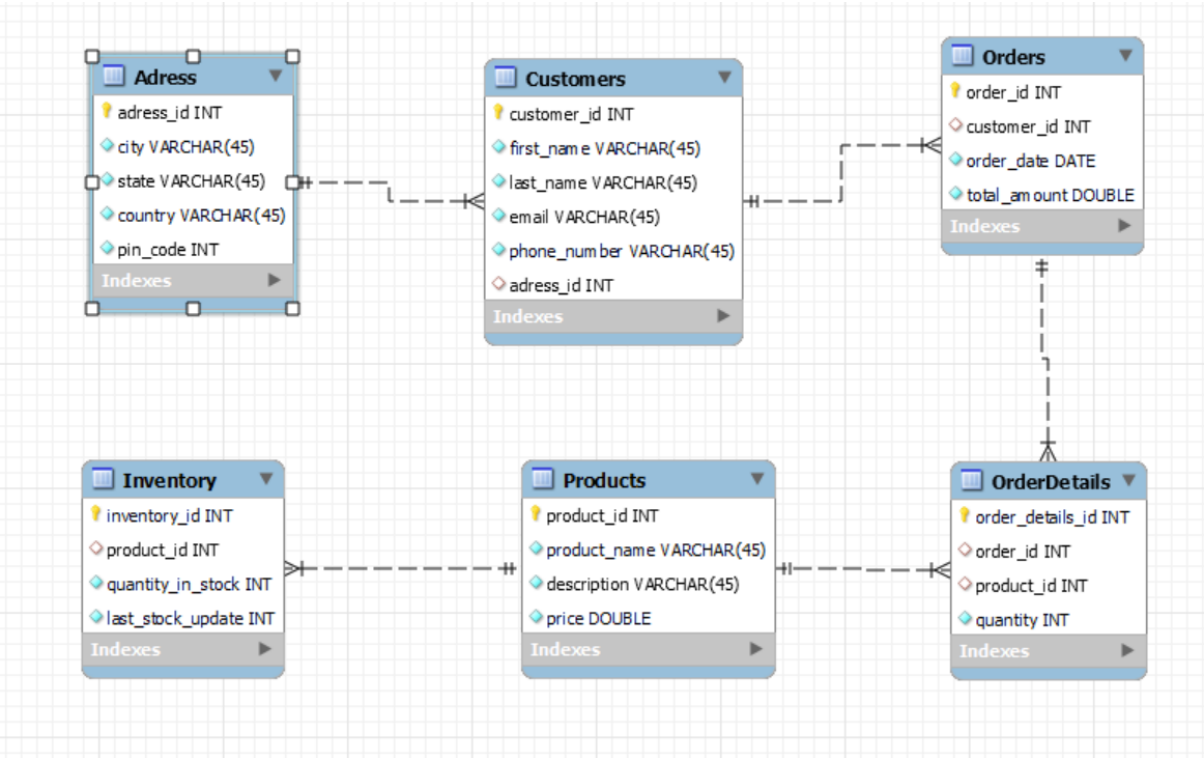

-- MySQL Script generated by MySQL Workbench

-- Tue Mar 5 20:06:53 2024

-- Model: New Model   Version: 1.0

-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;

SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;

SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ER
ROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- ---

-- Schema TechShop

-- ---

-- ---

-- Schema TechShop

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

-- -----------------------------------------------------
-- Table `TechShop`.`Adress`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `TechShop`.`Adress` (
  `adress_id` INT NOT NULL AUTO_INCREMENT,
  `city` VARCHAR(45) NOT NULL,
  `state` VARCHAR(45) NOT NULL,
  `country` VARCHAR(45) NOT NULL,
  `pin_code` INT NOT NULL,
  PRIMARY KEY (`adress_id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `TechShop`.`Customers`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `TechShop`.`Customers` (
  `customer_id` INT NOT NULL AUTO_INCREMENT,
  `first_name` VARCHAR(45) NOT NULL,
  `last_name` VARCHAR(45) NOT NULL,
  `email` VARCHAR(45) NOT NULL,
  `phone_number` VARCHAR(45) NOT NULL,
  `adress_id` INT NULL,
  PRIMARY KEY (`customer_id`),
  INDEX `customer_adress_id_idx` (`adress_id` ASC),
  CONSTRAINT `customer_adress_id`
    FOREIGN KEY (`adress_id`)
```

```
    REFERENCES `TechShop`.`Adress` (`adress_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `TechShop`.`Products`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `TechShop`.`Products` (
  `product_id` INT NOT NULL AUTO_INCREMENT,
  `product_name` VARCHAR(45) NOT NULL,
  `description` VARCHAR(45) NOT NULL,
  `price` DOUBLE NOT NULL,
  PRIMARY KEY (`product_id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `TechShop`.`Orders`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `TechShop`.`Orders` (
  `order_id` INT NOT NULL AUTO_INCREMENT,
  `customer_id` INT NULL,
  `order_date` DATE NOT NULL,
  `total_amount` DOUBLE NOT NULL,
  PRIMARY KEY (`order_id`),
  INDEX `customer_id_idx` (`customer_id` ASC),
  CONSTRAINT `customer_id`
    FOREIGN KEY (`customer_id`)
    REFERENCES `TechShop`.`Customers` (`customer_id`)
```

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


-- -----------------------------------------------------
-- Table `TechShop`.`OrderDetails`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `TechShop`.`OrderDetails` (
  `order_details_id` INT NOT NULL AUTO_INCREMENT,
  `order_id` INT NULL,
  `product_id` INT NULL,
  `quantity` INT NOT NULL,
  PRIMARY KEY (`order_details_id`),
  INDEX `order_id_idx` (`order_id` ASC) ,
  INDEX `ordered_product_id_idx` (`product_id` ASC) ,
  CONSTRAINT `order_id`
    FOREIGN KEY (`order_id`)
    REFERENCES `TechShop`.`Orders` (`order_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `ordered_product_id`
    FOREIGN KEY (`product_id`)
    REFERENCES `TechShop`.`Products` (`product_id`)
    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,
  `product_id` INT NULL,
  `quantity_in_stock` INT NOT NULL,
  `last_stock_update` INT NOT NULL,
  PRIMARY KEY (`inventory_id`),
  INDEX `inventory_product_id_idx` (`product_id` ASC),
  CONSTRAINT `inventory_product_id`
    FOREIGN KEY (`product_id`)
    REFERENCES `TechShop`.`Products` (`product_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;



SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;

insert into Adress(city,state,country,pin_code)
values('vzm','ap','india',535280),
('vizag','ap','india',535280),
('chennai','TN','india',535280),
('mumbai','MR','india',535280),
('hyderabad','TS','india',535280);

insert into customers(first_name,last_name,email,phone_number,adress_id)
values('venky','ramana','venky@gmail.com',9988776655,1),
('ram','rock','ram@gmail.com',9988776655,3),
('hare','krishna','hare@gmail.com',9988776655,2),
```

```
('lucky','viju','lucky@gmail.com',9988776655,5),
('james','gossling','james@gmail.com',9988776655,1),
('prabhu','ramana','prabhu@gmail.com',9988776655,5);

insert into orders(customer_id,order_date,total_amount)
values(1,'2024-01-01',1000),
(2,'2024-02-01',3000),
(3,'2024-02-10',500),
(1,'2024-03-05',1000);

insert into products(product_name,description,price)
values('smart watch','samsung galaxy smart watch',800),
('t-shirts','red colour team spirit',450),
('jeans','cotton jeans',1200),
('smart watch','apple smart watch',3000),
('power bank','30000 MAH power bank',800);

insert into orderDetails(order_id,product_id,quantity)
values(1,1,1),
(4,5,1),
(2,3,2),
(3,2,1);

insert into inventory(product_id,quantity_in_stock,last_stock_update)
values(1,12,5200),
(2,50,5000),
(3,40,6200),
(4,52,6400),
(5,55,6000);
```

-- Tasks 2: Select, Where, Between, AND, LIKE:

-- 1. Write an SQL query to retrieve the names and emails of all customers.

```
select concat(first_name,' ',last_name) from customers;
```

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

```
select c.customer_id,concate(c.first_name,c.last_name) as customer_name,o.order_date
from customers c join orders o
on 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_number,adress_id)
values('mahesh','babu','mahesh@gmail.com',9988776655,1);
```

-- 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 description='electronics' where product_id in (1,4,5);
update products set description='cloths' where product_id in (2,3);

update products set price=price+price*(10/100) where description='electronics';
```

-- 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 where order_id=2; -- onaction=cascade
```

-- 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(customer_id,order_date,total_amount)
values(5,'2024-03-02',5000);
```

-- 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='james_gossling@gmail.com' ,adress_id=4 where customer_id=5;
```

-- 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.

-- since order and orderdetails tables are different so unable to 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 where order_id=2; -- onaction=cascade
```

-- 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('iphone 14 max pro','electronics',140000);
```

-- 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.

```
alter table orders add column status varchar(45);
update orders set status='pending';

update orders set status='shipped' where order_id=3;


-- 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.

select c.customer_id,c.first_name,count(o.order_id) as number_of_orders
from customers c join orders o
on c.customer_id=o.customer_id
group by c.customer_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 c.customer_id,c.first_name,o.order_id,order_date,total_amount,status
from customers c join orders o
on c.customer_id=o.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_id,p.product_name,(p.price*os.quantity) as total_revenue
from products p join orderdetails os
on p.product_id=os.product_id
where p.description='electronics';
```

-- 3. Write an SQL query to list all customers who have made at least one purchase. Include their

-- names and contact information.

```
select c.customer_id,c.first_name,c.email,c.phone_number,count(o.order_id) as
number_of_purchases
from customers c join orders o
on c.customer_id=o.customer_id
group by c.customer_id
having number_of_purchases>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_id,p.product_name,sum(os.quantity) as total_quantity
from products p join orderdetails os
on p.product_id=os.product_id
where p.description='electronics'
group by  p.product_id;
```

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

-- there are no colums like categories in the schema

-- 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.customer_id,c.first_name,avg(o.total_amount) as average_order_value
from customers c join orders o
```

```
on c.customer_id=o.customer_id
group by c.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.customer_id,c.first_name,o.total_amount
from customers c join orders o
on c.customer_id=o.customer_id
order by o.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_id,p.product_name,p.description,count(p.product_id) as
number_of_times_ordered
from products p join orderdetails os
on p.product_id=os.product_id
join orders o
on os.order_id=o.order_id
 where p.description='electronics'
 group by p.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.customer_id,c.first_name,p.product_id,p.product_name,p.description
from products p join orderdetails os
on p.product_id=os.product_id
join orders o
on os.order_id=o.order_id
join customers c
```

```
on c.customer_id=o.customer_id
 where p.description='electronics' and p.product_name='power bank';

-- 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 sum(total_amount) as total_revenue
from orders
where order_date between '2024-01-01' and '2024-03-01';

-- Task 4. Subquery and its type:

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

select customer_id,first_name
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 count(product_id) as total_number_of_products_available_for_sale
from products where product_id in
(select product_id from inventory where  quantity_in_stock>0);

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

select sum(total_amount) as total_revenue
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 product_id,avg(quantity)
from orderdetails where product_id in (select product_id from products where
description='electronics');

-- 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) from orders where customer_id=1;

-- 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 c.customer_id,c.first_name,c.last_name,count(c.customer_id) as num_orders
from customers c join orders o
on c.customer_id=o.customer_id
group by c.customer_id
order by num_orders desc limit 0,1;

-- 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.

select description
from products
where product_id in
(select product_id from orderdetails group by product_id  order by sum(quantity) desc );

-- 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 customer_id,sum(total_amount) as total_revenue
```

```
from orders
where order_id in (select order_id from orderdetails
        where product_id in (select product_id from products where description='electronics'))
   group by 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 avg(total_amount) 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 customer_id,count(customer_id) as num_of_orders
from orders
where customer_id in (select customer_id from customers)
group by customer_id;
```