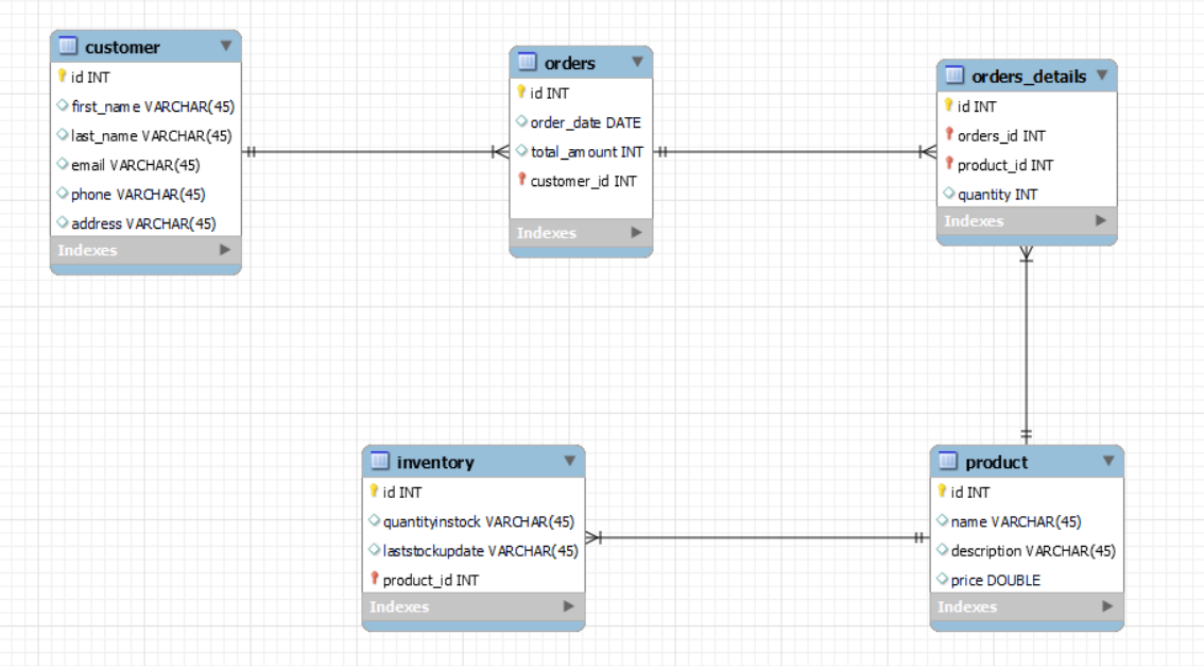
ASSIGNMENT NO 1

TECHSHOP

ER DIAGRAM:



Task 1: Database Design

-- MySQL Workbench Forward Engineering

-- ---

-- Schema assignment_electronic

-- ---

-- ---

-- Schema assignment_electronic

-- ---

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


-- -----------------------------------------------------
-- Table `assignment_electronic`.`customer`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `assignment_electronic`.`customer` (
  `id` INT NOT NULL,
  `first_name` VARCHAR(45) NULL,
  `last_name` VARCHAR(45) NULL,
  `email` VARCHAR(45) NULL,
  `phone` VARCHAR(45) NULL,
  `address` VARCHAR(45) NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `assignment_electronic`.`product`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `assignment_electronic`.`product` (
  `id` INT NOT NULL,
  `name` VARCHAR(45) NULL,
  `description` VARCHAR(45) NULL,
  `price` DOUBLE NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;



-- -----------------------------------------------------
```

```
-- Table `assignment_electronic`.`inventory`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `assignment_electronic`.`inventory` (
  `id` INT NOT NULL,
  `quantityinstock` VARCHAR(45) NULL,
  `laststockupdate` VARCHAR(45) NULL,
  `product_id` INT NOT NULL,
  PRIMARY KEY (`id`, `product_id`),
  INDEX `fk_inventory_product1_idx` (`product_id` ASC) ,
  CONSTRAINT `fk_inventory_product1`
    FOREIGN KEY (`product_id`)
    REFERENCES `assignment_electronic`.`product` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `assignment_electronic`.`orders`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `assignment_electronic`.`orders` (
  `id` INT NOT NULL,
  `order_date` DATE NULL,
  `total_amount` INT NULL,
  `customer_id` INT NOT NULL,
  PRIMARY KEY (`id`, `customer_id`),
  INDEX `fk_orders_customer1_idx` (`customer_id` ASC) ,
  CONSTRAINT `fk_orders_customer1`
    FOREIGN KEY (`customer_id`)
    REFERENCES `assignment_electronic`.`customer` (`id`)
```

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



-- -----------------------------------------------------
-- Table `assignment_electronic`.`orders_details`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `assignment_electronic`.`orders_details` (
  `id` INT NOT NULL,
  `orders_id` INT NOT NULL,
  `product_id` INT NOT NULL,
  `quantity` INT NULL,
  PRIMARY KEY (`id`, `orders_id`, `product_id`),
  INDEX `fk_orders_details_orders1_idx` (`orders_id` ASC) ,
  INDEX `fk_orders_details_product1_idx` (`product_id` ASC) ,
  CONSTRAINT `fk_orders_details_orders1`
    FOREIGN KEY (`orders_id`)
    REFERENCES `assignment_electronic`.`orders` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_orders_details_product1`
    FOREIGN KEY (`product_id`)
    REFERENCES `assignment_electronic`.`product` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

INSERTION:

-- customer insertion

```
INSERT INTO customer (id,first_name, last_name, phone, address)
VALUES
  (1,'MS', 'Dhoni',  '1234567890','a st'),
   (2,'Rishab', 'Pant', '9876543210','b st'),
   (3,'Rohit', 'Sharma', '5678901234','c st'),
   (4,'Virat', 'Kohli',  '3456789012','d st'),
   (5,'Jasprit', 'Bumrah',  '7890123456','e st'),
   (6,'Kuldeep', 'Yadav',  '2345678901','f st'),
   (7,'Ravichandran', 'Ashwin', '8901234567','g st'),
   (8,'Rinku', 'Singh',  '4567890123','i st'),
   (9,'Ravindra', 'Jadeja',  '6789012345','j st'),
   (10,'Shubman', 'gill',  '9012345678','k st');
```

```
mysql> select * from customer;
+----+--------------+-----------+------------------+------------+---------+
| id | first_name   | last_name | email            | phone      | address |
+----+--------------+-----------+------------------+------------+---------+
|  1 | MS           | Dhoni     | ms@gmail.com     | 1234567890 | a st    |
|  2 | Rishab       | Pant      | rishab@gmail.com | 9876543210 | b st    |
|  3 | Rohit        | Sharma    | rs@gmail.com     | 5678901234 | c st    |
|  4 | Virat        | Kohli     | vk@gmail.com     | 3456789012 | d st    |
|  5 | Jasprit      | Bumrah    | boom@gmail.com   | 7890123456 | e st    |
|  6 | Kuldeep      | Yadav     | yad@gmail.com    | 2345678901 | f st    |
|  7 | Ravichandran | Ashwin    | ash100@gmail.com | 8901234567 | g st    |
|  8 | Rinku        | Singh     | rk@gmail.com     | 4567890123 | i st    |
|  9 | Ravindra     | Jadeja    | jaddu@gmail.com  | 6789012345 | j st    |
| 10 | Shubman      | gill      | shub@gmail.com   | 9012345678 | k st    |
+----+--------------+-----------+------------------+------------+---------+
10 rows in set (0.02 sec)
```

--product insertion

```
INSERT INTO product (id, name, description, price)
VALUES
   (1,'Laptop', '15-inch, 8GB RAM, 512GB SSD', 999.99),
   (2,'Smartphone', '6.5-inch, 128GB storage, Android', 699.99),
   (3,'Tablet', '10.2-inch, 64GB storage, Wi-Fi', 399.99),
```

(4,'Headphones', 'Over-ear, noise-canceling', 199.99),

(5,'Smartwatch', 'Fitness tracker, heart rate monitor', 149.99),

(6,'Bluetooth Speaker', 'Waterproof, 20-hour battery life', 79.99),

(7,'Camera', 'DSLR, 24MP, 4K video', 899.99),

(8,'Printer', 'Color laser, wireless', 349.99),

(9,'Gaming Console', '4K gaming, 1TB storage', 499.99),

(10,'Drone', '4K camera, GPS', 799.99);

```
mysql> select * from product;
+----+------------------+-------------------------------------+--------+
| id | name             | description                         | price  |
+----+------------------+-------------------------------------+--------+
|  1 | Laptop           | 15-inch, 8GB RAM, 512GB SSD         | 999.99 |
|  2 | Smartphone       | 6.5-inch, 128GB storage, Android    | 699.99 |
|  3 | Tablet           | 10.2-inch, 64GB storage, Wi-Fi      | 399.99 |
|  4 | Headphones       | Over-ear, noise-canceling           | 199.99 |
|  5 | Smartwatch       | Fitness tracker, heart rate monitor | 149.99 |
|  6 | Bluetooth Speaker | Waterproof, 20-hour battery life    |  79.99 |
|  7 | Camera           | DSLR, 24MP, 4K video                | 899.99 |
|  8 | Printer          | Color laser, wireless               | 349.99 |
|  9 | Gaming Console   | 4K gaming, 1TB storage              | 499.99 |
| 10 | Drone            | 4K camera, GPS                      | 799.99 |
+----+------------------+-------------------------------------+--------+
10 rows in set (0.03 sec)
```

--- orders insertion

INSERT INTO orders (id, order_date, total_amount,customer_id)

VALUES

(1, '2024-03-01', 1299.98,'1'),

(2, '2024-03-02', 1499.97,'2'),

(3, '2024-03-03', 899.99,'3'),

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

(5, '2024-03-05', 699.98,'5'),

(6, '2024-03-06', 279.97,'6'),

(7, '2024-03-07', 599.99,'7'),

(8, '2024-03-08', 399.98,'8'),

(9, '2024-03-09', 999.99,'9'),

(10,'2024-03-10',1599.96,'10');

```
mysql> select * from orders;
+----+------------+--------------+-------------+
| id | order_date | total_amount | customer_id |
+----+------------+--------------+-------------+
|  1 | 2024-03-01 |         1300 |           1 |
|  2 | 2024-03-02 |         1500 |           2 |
|  3 | 2024-03-03 |          900 |           3 |
|  4 | 2024-03-04 |          200 |           4 |
|  5 | 2024-03-05 |          700 |           5 |
|  6 | 2024-03-06 |          280 |           6 |
|  7 | 2024-03-07 |          600 |           7 |
|  8 | 2024-03-08 |          400 |           8 |
|  9 | 2024-03-09 |         1000 |           9 |
| 10 | 2024-03-10 |         1600 |          10 |
+----+------------+--------------+-------------+
10 rows in set (0.01 sec)
```

--- order detail insertion

insert into orders_details(id,orders_id,product_id,quantity) values

(1,1,2,2),

(2,2,1,1),

(3,3,5,2),

(4,4,6,3),

(5,5,7,2),

(6,6,3,5),

(7,7,10,1),

(8,8,4,10),

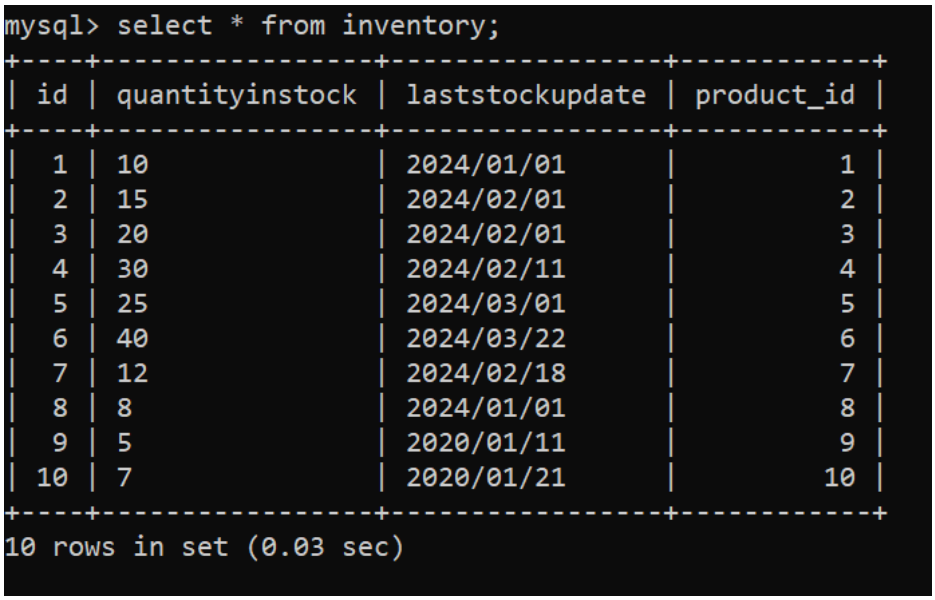
(9,9,9,10),

(10,10,1,1);

```
mysql> select * from orders_details;
+----+-----------+------------+----------+
| id | orders_id | product_id | quantity |
+----+-----------+------------+----------+
|  1 |         1 |          2 |        2 |
|  2 |         2 |          1 |        1 |
|  3 |         3 |          5 |        2 |
|  4 |         4 |          6 |        3 |
|  5 |         5 |          7 |        2 |
|  6 |         6 |          3 |        5 |
|  7 |         7 |         10 |        1 |
|  8 |         8 |          4 |       10 |
|  9 |         9 |          9 |       10 |
| 10 |        10 |          1 |        1 |
+----+-----------+------------+----------+
10 rows in set (0.00 sec)
```

--- inventory insertion

```
INSERT INTO Inventory (id, product_id, quantityinstock, laststockupdate)
VALUES
    (1,1, 10, '2024/01/01'),
    (2,2, 15, '2024/02/01'),
    (3,3, 20, '2024/02/01'),
    (4,4, 30, '2024/02/11'),
    (5,5, 25, '2024/03/01'),
    (6,6, 40, '2024/03/22'),
    (7,7, 12, '2024/02/18'),
    (8,8, 8, '2024/01/01'),
    (9,9, 5, '2020/01/11'),
    (10,10, 7, '2020/01/21');
```



```
mysql> select * from inventory;
+----+-----------------+-----------------+------------+
| id | quantityinstock | laststockupdate | product_id |
+----+-----------------+-----------------+------------+
|  1 | 10              | 2024/01/01      |          1 |
|  2 | 15              | 2024/02/01      |          2 |
|  3 | 20              | 2024/02/01      |          3 |
|  4 | 30              | 2024/02/11      |          4 |
|  5 | 25              | 2024/03/01      |          5 |
|  6 | 40              | 2024/03/22      |          6 |
|  7 | 12              | 2024/02/18      |          7 |
|  8 | 8               | 2024/01/01      |          8 |
|  9 | 5               | 2020/01/11      |          9 |
| 10 | 7               | 2020/01/21      |         10 |
+----+-----------------+-----------------+------------+
10 rows in set (0.03 sec)
```

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

-- 1. Retrieve names and emails of all customers

```
select concat(first_name," ",last_name) as name,phone from customer;
```

-- 2. List all orders with their order dates and corresponding customer names

```
SELECT  o.order_date, CONCAT(c.first_name, ' ', c.last_name) AS CustomerName
FROM Orders o
JOIN Customer c ON o.customer_id = c.id;
```

-- 3. Insert a new customer record into the "Customers" table

```
INSERT INTO customer (id,first_name, last_name, phone, address)
VALUES  (11,'Mohd', 'Shami',  '1239567890','l st');
```

-- 4. Update the prices of all electronic gadgets in the "Products" table by increasing them by 10%

```
UPDATE product
SET price = price * 1.0
WHERE id =10;
```

-- 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 o,s from order_details o,orders s where s.id=o.orders_id and s.id=7;
```

---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 (id,order_date,total_amount,customer_id) values
(11'2024/02/29',7000,11);
```

--- 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='msd7@gmail.com',address='aa st' where id=1;
```

---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.

```
update orders o
set total_amount=(select p.price*od.quantity from
product p join orders_details od on p.id=od.product_id
where o.id=od.orders_id);
```

---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 id=9;
```

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 (name, description, price) VALUES
('mac book', 'High-performance laptop with mac os', 150000);
```

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=1;
```

---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 customer c
set number_of_orders=(select count(*)
from orders o
 where c.id=o.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 o.orderid, o.order_date, CONCAT(c.first_name, ' ', c.last_name) AS
CustomerName, c.email, c.phone FROM orders o

JOIN customer c ON o.CustomerID = c.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.name, sum(o.total_amount) as revenue

from product p join orders_details od on p.id=od.product_id

join orders o on o.id=od.orders_id

group by p.id;
```

3. Write an SQL query to list all customers who have made at least one purchase. Include their names and contact information.

```
select concat(c.first_name," ",c.last_name) as Customer_Name,c.phone,c.email

from customer c join orders o on c.id=o.customer_id

group by c.id

having count(c.id)>=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.name,sum(od.quantity) as popular_gadget

from product p join orders_details od

on p.id=od.product_id

group by p.id

order by popular_gadget desc limit 0,1;
```

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

```
select name as devices 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.total_amount)

from customer c join orders o on c.id=o.customer_id

group by c.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 o.id,c.*,o.total_amount

from customer c join orders o on c.id=o.customer_id

having max(o.total_amount);


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

select p.name,count(p.id) as number_of_times_ordered

from product p join orders_details od on p.id=od.product_id

group by p.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 p.name,group_concat(concat(c.first_name," ",c.last_name)) as

customers

from customer c join orders o on c.id=o.customer_id

join orders_details od on o.id=od.orders_id

join product p on p.id=od.product_id

group by p.id;


-- 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 '2024-12-31';
```

Task 4. Subquery and its type

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

```
select concat(first_name," ",last_name) as customer
from customer where id not in(select customer_id from orders);
```

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

```
select i.product_id,(i.quantityinstock- (select sum(od.quantity)
from orders_details od
where od.product_id=i.id)) as number_of_products_available_for_sale
from inventory i ;
```

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

```
select sum(total_amount) as toatal_revenue
from (select total_amount from orders) as revenue_by_techshop;
```

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.

No field category

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 concat(c.first_name," ",c.last_name) as name ,
(select sum(o.total_amount) from orders o
where o.customer_id=c.id
group by o.customer_id)as total_revenue
from customer c;
```

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 name ,
(select count(o.customer_id) from orders o
```

```
where o.customer_id=c.id
group by o.customer_id)as order_count
from customer c
order by order_count desc;
```

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 p.name , (select sum(od.quantity)
from orders_details od
where p.id=od.product_id
group by od.product_id) as popular_product
from product p
order by popular_product desc limit 0,1;
```

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 concat(c.first_name," ",c.last_name) as most_money_spender,
(select sum(o.total_amount) from orders o
where c.id=o.customer_id
group by o.customer_id) as money_spent
from customer c
order by money_spent 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 concat(c.first_name," ",c.last_name) as name,
(select avg(o.total_amount) from orders o
where o.customer_id=c.id
group by o.customer_id)
as average_order_value from customer c
order by average_order_value desc;
```

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 name,
(select count(o.customer_id) from orders o
where o.customer_id=c.id
group by o.customer_id)
as total_number_of_orders from customer c
order by total_number_of_orders desc;
```