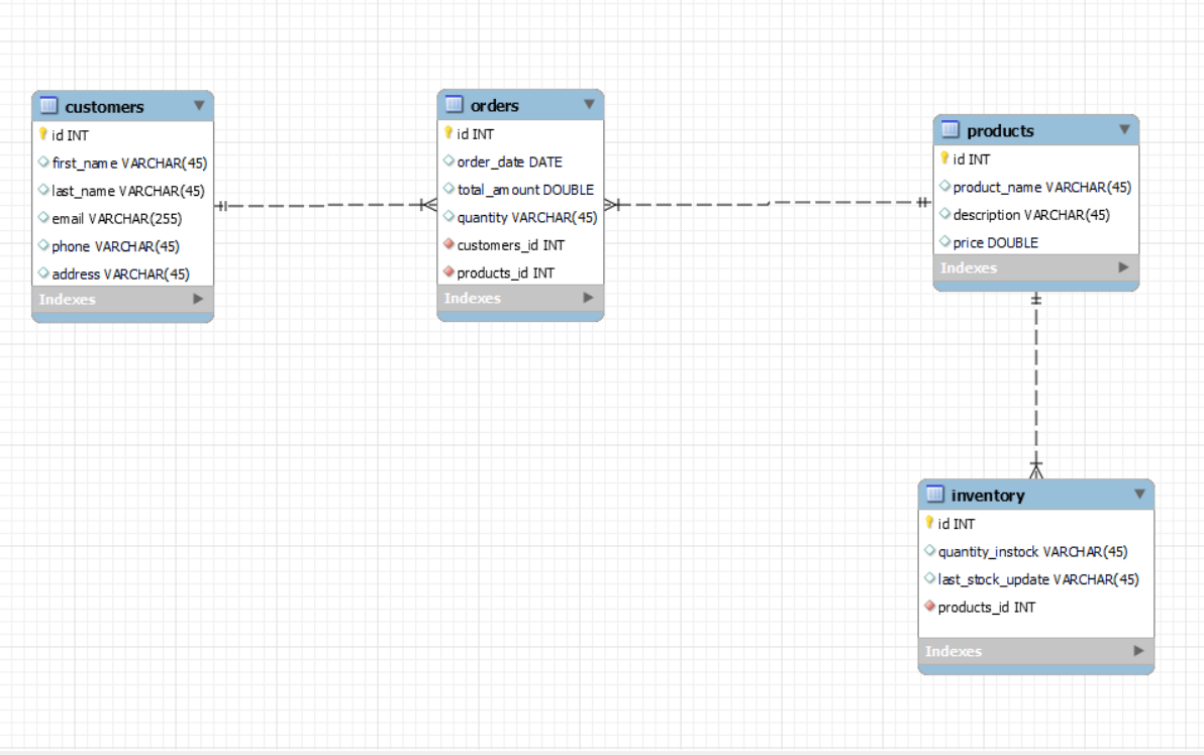
TechShop, an electronic gadgets shop



-- 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');
```

id	first_name	last_name	email	phone	address
1	John	Doe	john.doe@example.com	123-456-7890	123 Main St
2	Jane	Smith	jane.smith@example.com	987-654-3210	456 Elm St
3	Alice	Johnson	alice.johnson@example.com	555-555-5555	789 Oak St
4	Bob	Brown	bob.brown@example.com	111-222-3333	321 Pine St
5	Eva	Green	eva.green@example.com	444-444-4444	654 Cedar St
NULL	NULL	NULL	NULL	NULL	NULL

```
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);
```

id	product_name	description	price
1	Laptop	15-inch, 8GB RAM, 256GB SSD	900
2	Smartphone	6.2-inch, 128GB storage, 5G	800
3	Tablet	10.5-inch, 64GB storage, Wi-Fi	500
4	Headphones	Wireless, noise-cancelling	200
5	Keyboard	Mechanical, RGB lighting	150
NULL	NULL	NULL	NULL

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);

id	order_date	total_amount	quantity	customers_id	products_id
1	2024-03-01	999.99	1	1	1
2	2024-03-02	799.99	1	2	2
3	2024-03-03	499.99	1	3	3
4	2024-03-04	199.99	1	4	4
5	2024-03-05	129.99	1	5	5
NULL	NULL	NULL	NULL	NULL	NULL

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);

Result Grid | Filter Rows: | Edit:

id	quantity_instock	last_stock_update	products_id
1	10	2024-03-01	1
2	20	2024-03-01	2
3	15	2024-03-01	3
4	30	2024-03-01	4
5	25	2024-03-01	5
NULL	NULL	NULL	NULL

inventory 4 ×

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;

first_name	last_name	email
John	Doe	john.doe@example.com
Jane	Smith	jane.smith@example.com
Alice	Johnson	alice.johnson@example.com
Bob	Brown	bob.brown@example.com
Eva	Green	eva.green@example.com

2. 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;
```

order_id	order_date	first_name	last_name
1	2024-03-01	John	Doe
2	2024-03-02	Jane	Smith
3	2024-03-03	Alice	Johnson
4	2024-03-04	Bob	Brown
5	2024-03-05	Eva	Green

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

id	first_name	last_name	email	phone	address
1	John	Doe	john.doe@example.com	123-456-7890	123 Main St
2	Jane	Smith	jane.smith@example.com	987-654-3210	456 Elm St
3	Alice	Johnson	alice.johnson@example.com	555-555-5555	789 Oak St
4	Bob	Brown	Johnson n@example.com	111-222-3333	321 Pine St
5	Eva	Green	eva.green@example.com	444-444-4444	654 Cedar St
6	John	Doe	john.doe@example.com	123-456-7890	123 Main St

4. write an sql query to update the prices of all electronic gadgets in the "products" table by increasing them by 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.

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 (order_date, total_amount, quantity, customers_id, products_id)

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

id	order_date	total_amount	quantity	customers_id	products_id
2	2024-03-02	799.99	1	2	2
3	2024-03-03	499.99	1	3	3
4	2024-03-04	199.99	1	4	4
5	2024-03-05	129.99	1	5	5
6	2024-03-07	100	1	1	1
NULL	NULL	NULL	NULL	NULL	NULL

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 = 'myemail@example.com', address = '456 new st'
where id = 3;

id	first_name	last_name	email	phone	address
1	John	Doe	john.doe@example.com	123-456-7890	123 Main St
2	Jane	Smith	jane.smith@example.com	987-654-3210	456 Elm St
3	Alice	Johnson	myemail@example.com	555-555-5555	456 New St
4	Bob	Brown	bob.brown@example.com	111-222-3333	321 Pine St
5	Eva	Green	eva.green@example.com	444-444-4444	654 Cedar St
6	John	Doe	john.doe@example.com	123-456-7890	123 Main St

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');

id	product_name	description	price	category
1	Laptop	15-inch, 8GB RAM, 256GB SSD	900	NULL
2	Smartphone	6.2-inch, 128GB storage, 5G	800	NULL
3	Tablet	10.5-inch, 64GB storage, Wi-Fi	500	NULL
4	Headphones	Wireless, noise-cancelling	200	NULL
5	Keyboard	Mechanical, RGB lighting	150	NULL
6	Remote	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>;

id	order_date	total_amount	quantity	customers_id	products_id	status
2	2024-03-02	799.99	1	2	2	NULL
3	2024-03-03	499.99	1	3	3	NULL
4	2024-03-04	199.99	1	4	4	Shipped
5	2024-03-05	129.99	1	5	5	NULL
6	2024-03-07	100	1	1	1	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL

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;
```

order_id	order_date	first_name	last_name	email	phone	address
2	2024-03-02	Jane	Smith	jane.smith@example.com	987-654-3210	456 Elm St
3	2024-03-03	Alice	Johnson	myemail@example.com	555-555-5555	456 New St
4	2024-03-04	Bob	Brown	bob.brown@example.com	111-222-3333	321 Pine St
5	2024-03-05	Eva	Green	eva.green@example.com	444-444-4444	654 Cedar St
6	2024-03-07	John	Doe	john.doe@example.com	123-456-7890	123 Main St

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(o.total_amount) as total_revenue
from orders o
join products p on o.products_id = p.id
group by p.product_name;
```

product_name	total_revenue
Headphones	199.99
Keyboard	129.99
Laptop	100
Smartphone	799.99
Tablet	499.99

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.email, c.phone, c.address
```

```
from customers c
join orders o on c.id = o.customers_id
group by c.id;
```

first_name	last_name	email	phone	address
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	myemail@example.com	555-555-5555	456 New 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

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(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;
```

product_name	total_quantity_ordered
Tablet	1

5. 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;
```

product_name	category
Laptop	NULL
Smartphone	NULL
Tablet	NULL
Headphones	NULL
Keyboard	NULL
Remote	Electronic Gadgets

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, 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;
```

first_name	last_name	average_order_value
John	Doe	100
Jane	Smith	799.99
Alice	Johnson	499.99
Bob	Brown	199.99
Eva	Green	129.99

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 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;
```

order_id	first_name	last_name	email	phone	address	total_revenue
2	Jane	Smith	jane.smith@example.com	987-654-3210	456 Elm St	799.99

8. 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;
```

product_name	order_count
Headphones	1
Keyboard	1
Laptop	1
Smartphone	1
Tablet	1

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 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';
```

first_name	last_name	email	phone	address
Alice	Johnson	myemail@example.com	555-555-5555	456 New St

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 'start_date' and 'end_date';
```

total_revenue
NULL

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);
```

id	first_name	last_name	email
6	John	Doe	john.doe@example.com
NULL	NULL	NULL	NULL

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

```
select count(*) as total_products
from products;
```

total_products
6

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

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

total_revenue
1729.96

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.

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(total_amount) as total_revenue
from orders
where customers_id = 1;
```

total_revenue
100

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.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;
```

first_name	last_name	order_count
John	Doe	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 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;
```

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, 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;
```

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) as avg_order_value
from orders;
```

avg_order_value
345.992

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;
```

first_name	last_name	order_count
John	Doe	1
Jane	Smith	1
Alice	Johnson	1
Bob	Brown	1
Eva	Green	1