Online Shopping Database

Author: Yan Peng

CSC675 2023.5.18

Data Base Description

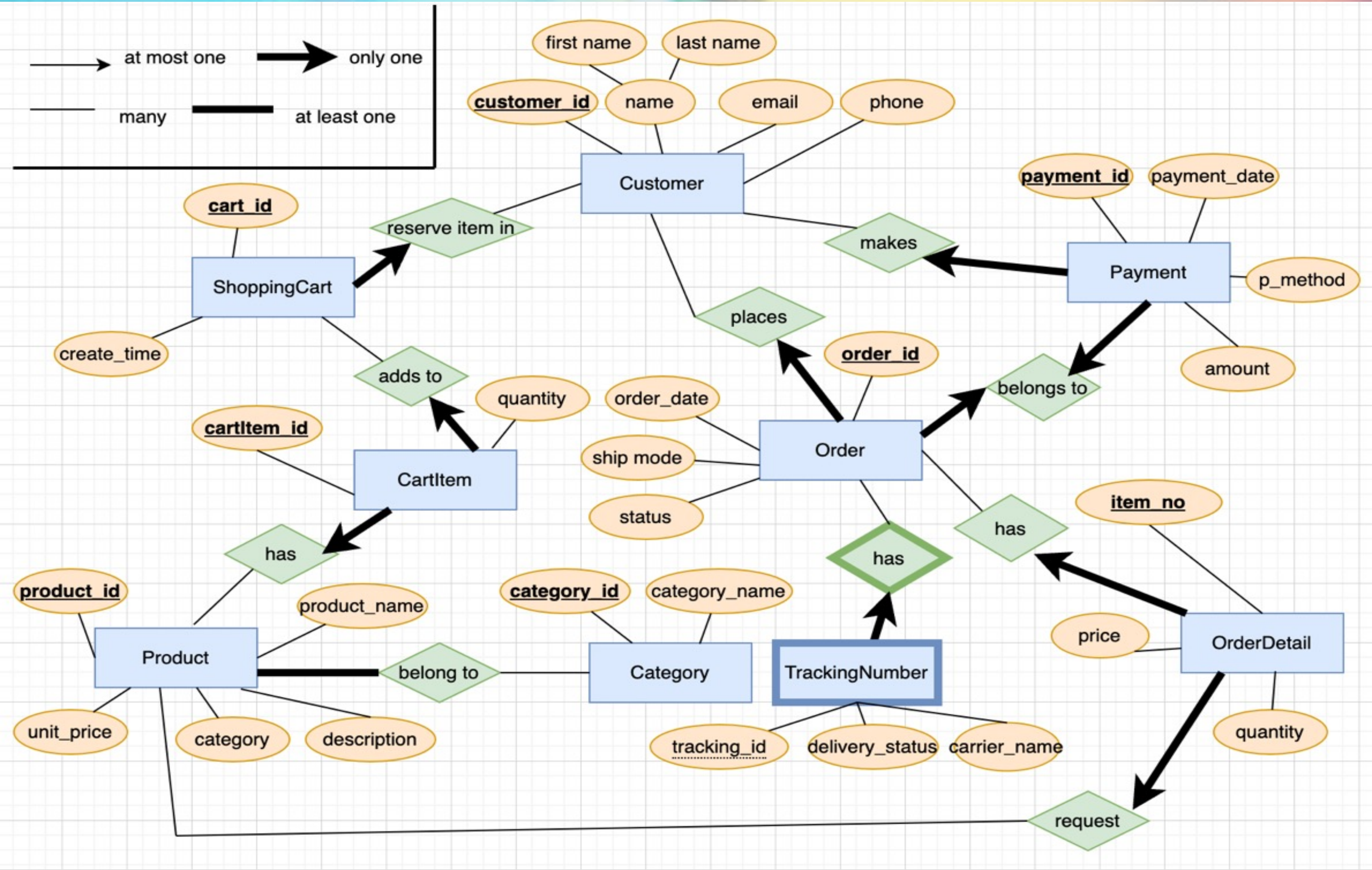
This online shopping database is a collection of organized data that stores information about products, categories, shopping carts, cart items, customers, orders, order details, tracking and payments for an online shopping store.

Data Requirement

This online shopping database contains:

- Product Information
- Customer Information
- Shopping Cart Information
- Order Information
- Tracking Information
- Payment Information

ER DIAGRAM


at most one
only one
many
at least one
first name
last name
customer_id
name
email
phone
Customer
payment_id
payment_date
cart_id
reserve item in
makes
Payment
p_method
ShoppingCart
places
create_time
order_id
amount
adds to
belongs to
quantity
order_date
cartItem_id
ship mode
Order
CartItem
status
item_no
has
has
has
product_id
category_id
category_name
product_name
price
OrderDetail
Product
belong to
Category
TrackingNumber
unit_price
category
description
tracking_id
delivery_status
carrier_name
quantity
request

Relational Schemas:

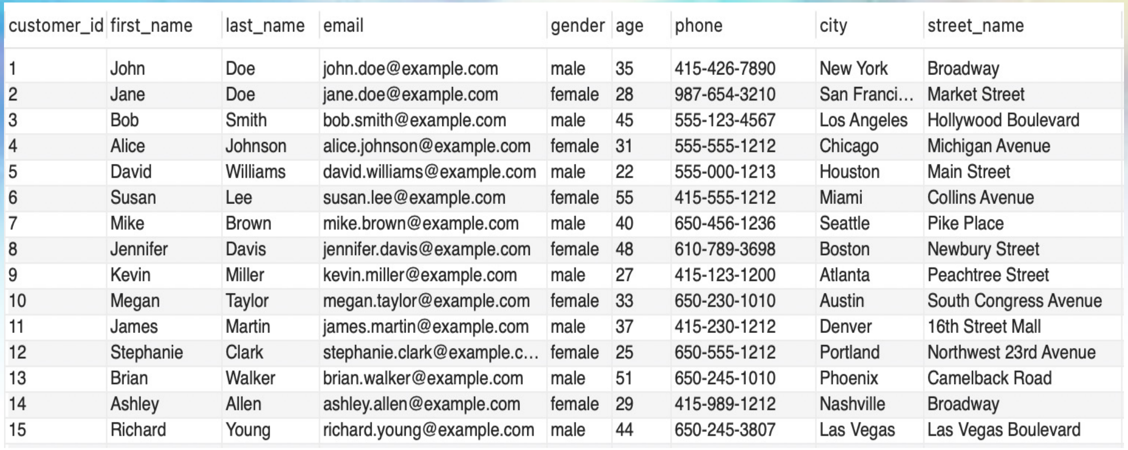
1) Customer {**customer_id (PK**), first name, last name, email, phone}
2) ShoppingCart {**cart_id (PK),** creat_time, customer_id (**FK**)}
3) CartItem {**cartItem_id (PK),** quantity, cart_id (**FK**), product_id (**FK**)}
4) Product {**product_id (PK),** unit_price, category, description, product_name, category_id(**FK**)}
5) Category {**category_id (PK),** category_name}
6) Payment {**payment_id (PK),** date, amount, payment_method, customer_id (**FK**)}
7) Orders {**order_id (PK),** date, status, shipmethod, customer_id (**FK**), payment_id (**FK**)}
8) OrderDetail {**item_no (PK),** price, quantity, order_id (**FK**), product_id (**FK**)}
9) TrackingNumber {(**order_id, tracking_id) (PK),** delivery_status, carrier_name, order_id(**FK**)**}**

Implementation

1. Created all tables:
   - 1.1 Customer Table
   - 1.2 ShoppingCart Table
   - 1.3 Product Table
   - 1.4 Category Table
   - 1.5 CartItem Table
   - 1.6 Payment Table
   - 1.7 Orders Table
   - 1.8 OrderDetail Table
   - 1.9 TrackingNumber Table

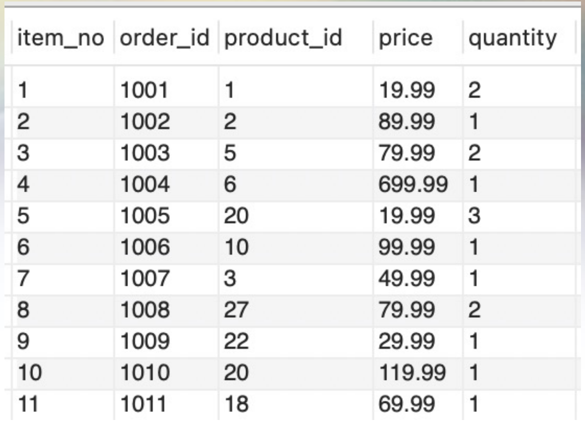
Table Content I

Customer



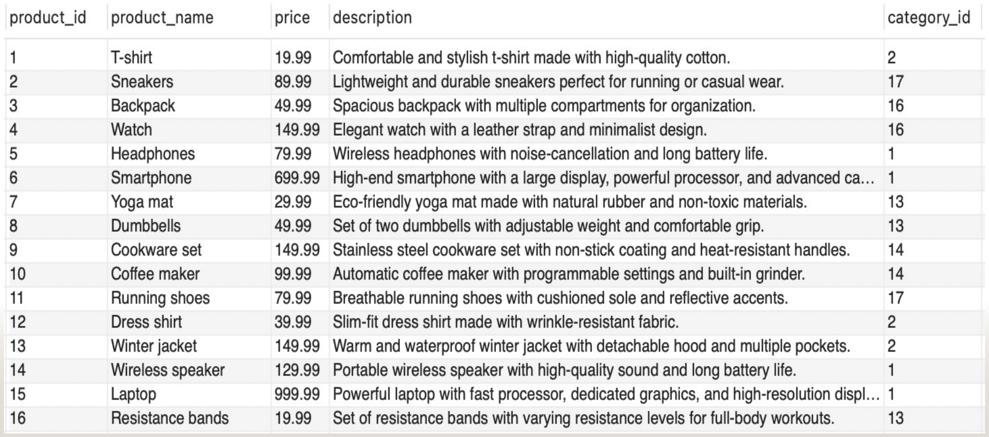
customer_id	first_name	last_name	email	gender	age	phone	city	street_name
1	John	Doe	john.doe@example.com	male	35	415-426-7890	New York	Broadway
2	Jane	Doe	jane.doe@example.com	female	28	987-654-3210	San Franci…	Market Street
3	Bob	Smith	bob.smith@example.com	male	45	555-123-4567	Los Angeles	Hollywood Boulevard
4	Alice	Johnson	alice.johnson@example.com	female	31	555-555-1212	Chicago	Michigan Avenue
5	David	Williams	david.williams@example.com	male	22	555-000-1213	Houston	Main Street
6	Susan	Lee	susan.lee@example.com	female	55	415-555-1212	Miami	Collins Avenue
7	Mike	Brown	mike.brown@example.com	male	40	650-456-1236	Seattle	Pike Place
8	Jennifer	Davis	jennifer.davis@example.com	female	48	610-789-3698	Boston	Newbury Street
9	Kevin	Miller	kevin.miller@example.com	male	27	415-123-1200	Atlanta	Peachtree Street
10	Megan	Taylor	megan.taylor@example.com	female	33	650-230-1010	Austin	South Congress Avenue
11	James	Martin	james.martin@example.com	male	37	415-230-1212	Denver	16th Street Mall
12	Stephanie	Clark	stephanie.clark@example.c…	female	25	650-555-1212	Portland	Northwest 23rd Avenue
13	Brian	Walker	brian.walker@example.com	male	51	650-245-1010	Phoenix	Camelback Road
14	Ashley	Allen	ashley.allen@example.com	female	29	415-989-1212	Nashville	Broadway
15	Richard	Young	richard.young@example.com	male	44	650-245-3807	Las Vegas	Las Vegas Boulevard

OrderDetail



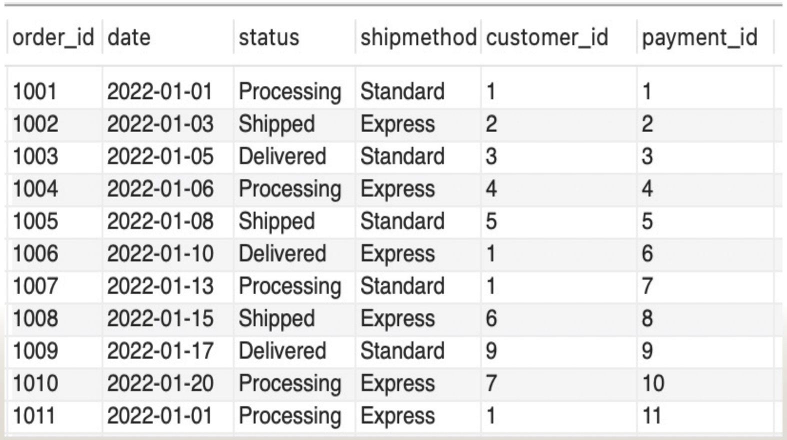
item_no	order_id	product_id	price	quantity
1	1001	1	19.99	2
2	1002	2	89.99	1
3	1003	5	79.99	2
4	1004	6	699.99	1
5	1005	20	19.99	3
6	1006	10	99.99	1
7	1007	3	49.99	1
8	1008	27	79.99	2
9	1009	22	29.99	1
10	1010	20	119.99	1
11	1011	18	69.99	1

Product



product_id	product_name	price	description	category_id
1	T-shirt	19.99	Comfortable and stylish t-shirt made with high-quality cotton.	2
2	Sneakers	89.99	Lightweight and durable sneakers perfect for running or casual wear.	17
3	Backpack	49.99	Spacious backpack with multiple compartments for organization.	16
4	Watch	149.99	Elegant watch with a leather strap and minimalist design.	16
5	Headphones	79.99	Wireless headphones with noise-cancellation and long battery life.	1
6	Smartphone	699.99	High-end smartphone with a large display, powerful processor, and advanced ca…	1
7	Yoga mat	29.99	Eco-friendly yoga mat made with natural rubber and non-toxic materials.	13
8	Dumbbells	49.99	Set of two dumbbells with adjustable weight and comfortable grip.	13
9	Cookware set	149.99	Stainless steel cookware set with non-stick coating and heat-resistant handles.	14
10	Coffee maker	99.99	Automatic coffee maker with programmable settings and built-in grinder.	14
11	Running shoes	79.99	Breathable running shoes with cushioned sole and reflective accents.	17
12	Dress shirt	39.99	Slim-fit dress shirt made with wrinkle-resistant fabric.	2
13	Winter jacket	149.99	Warm and waterproof winter jacket with detachable hood and multiple pockets.	2
14	Wireless speaker	129.99	Portable wireless speaker with high-quality sound and long battery life.	1
15	Laptop	999.99	Powerful laptop with fast processor, dedicated graphics, and high-resolution displ…	1
16	Resistance bands	19.99	Set of resistance bands with varying resistance levels for full-body workouts.	13

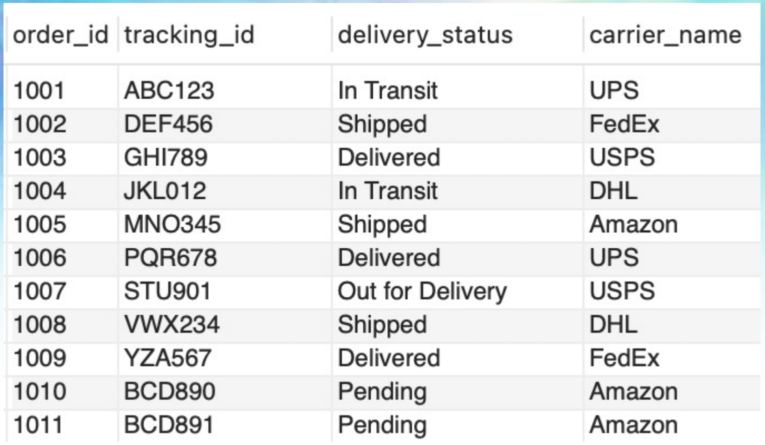
Orders



order_id	date	status	shipmethod	customer_id	payment_id
1001	2022-01-01	Processing	Standard	1	1
1002	2022-01-03	Shipped	Express	2	2
1003	2022-01-05	Delivered	Standard	3	3
1004	2022-01-06	Processing	Express	4	4
1005	2022-01-08	Shipped	Standard	5	5
1006	2022-01-10	Delivered	Express	1	6
1007	2022-01-13	Processing	Standard	1	7
1008	2022-01-15	Shipped	Express	6	8
1009	2022-01-17	Delivered	Standard	9	9
1010	2022-01-20	Processing	Express	7	10
1011	2022-01-01	Processing	Express	1	11

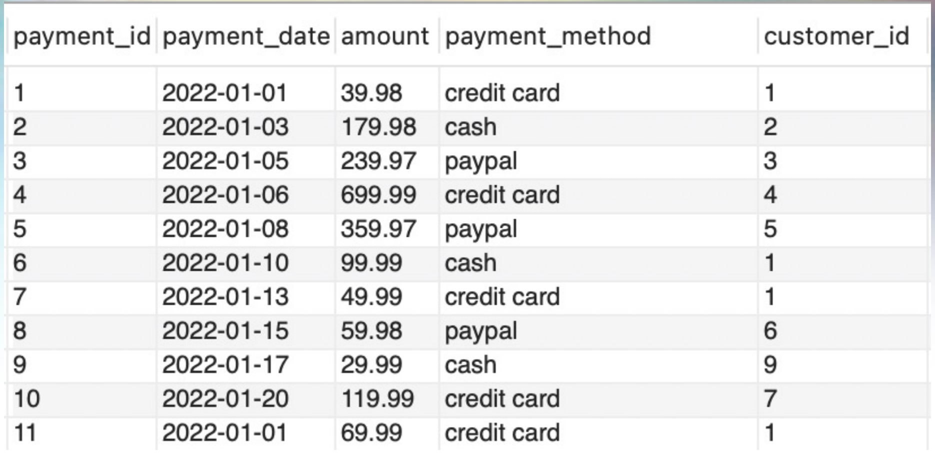
Table Content II

TrackingNumber



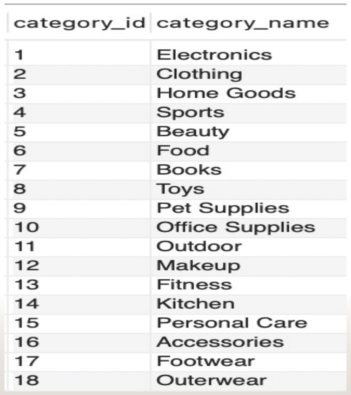
order_id	tracking_id	delivery_status	carrier_name
1001	ABC123	In Transit	UPS
1002	DEF456	Shipped	FedEx
1003	GHI789	Delivered	USPS
1004	JKL012	In Transit	DHL
1005	MNO345	Shipped	Amazon
1006	PQR678	Delivered	UPS
1007	STU901	Out for Delivery	USPS
1008	VWX234	Shipped	DHL
1009	YZA567	Delivered	FedEx
1010	BCD890	Pending	Amazon
1011	BCD891	Pending	Amazon

Payment



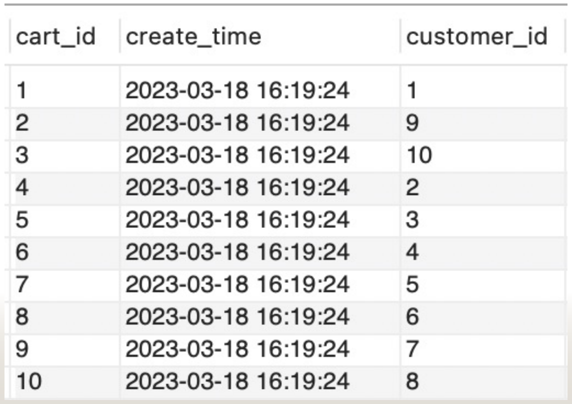
payment_id	payment_date	amount	payment_method	customer_id
1	2022-01-01	39.98	credit card	1
2	2022-01-03	179.98	cash	2
3	2022-01-05	239.97	paypal	3
4	2022-01-06	699.99	credit card	4
5	2022-01-08	359.97	paypal	5
6	2022-01-10	99.99	cash	1
7	2022-01-13	49.99	credit card	1
8	2022-01-15	59.98	paypal	6
9	2022-01-17	29.99	cash	9
10	2022-01-20	119.99	credit card	7
11	2022-01-01	69.99	credit card	1

Category



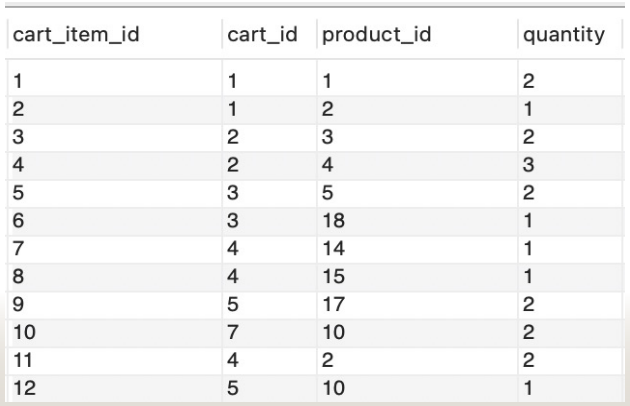
category_id	category_name
1	Electronics
2	Clothing
3	Home Goods
4	Sports
5	Beauty
6	Food
7	Books
8	Toys
9	Pet Supplies
10	Office Supplies
11	Outdoor
12	Makeup
13	Fitness
14	Kitchen
15	Personal Care
16	Accessories
17	Footwear
18	Outerwear

ShoppingCart



cart_id	create_time	customer_id
1	2023-03-18 16:19:24	1
2	2023-03-18 16:19:24	9
3	2023-03-18 16:19:24	10
4	2023-03-18 16:19:24	2
5	2023-03-18 16:19:24	3
6	2023-03-18 16:19:24	4
7	2023-03-18 16:19:24	5
8	2023-03-18 16:19:24	6
9	2023-03-18 16:19:24	7
10	2023-03-18 16:19:24	8

CartItem



cart_item_id	cart_id	product_id	quantity
1	1	1	2
2	1	2	1
3	2	3	2
4	2	4	3
5	3	5	2
6	3	18	1
7	4	14	1
8	4	15	1
9	5	17	2
10	7	10	2
11	4	2	2
12	5	10	1

Implementation (MySQL)

2. SQL QUERIES

2.1 Creating View

This query will create view that display the total amount of sales by customer.

```
CREATE VIEW customer_sale AS
SELECT c.customer_id, c.first_name, c.last_name, SUM(od.price * od.quantity) AS total_sales
FROM Customer c, OrderDetail od, Orders o
WHERE c.customer_id = o.customer_id
AND o.order_id = od.order_id
GROUP BY c.customer_id, c.first_name, c.last_name;
```

How to use this view? Use the view in this query just like any other table.
If want to retrieve the top 5 customers by sales.

```
SELECT customer_id, total_sale
FROM customer_sale
ORDER BY total_sales DESC
LIMIT 5;
```

Implementation (MySQL)

2.2 Using Group By, Having with aggregate operators

Ex #1 The query groups the customers by gender and calculates the average age for each gender.

```
SELECT gender, Round(AVG(age),0) AS
avg_age
FROM Customer
GROUP BY gender
HAVING avg_age > 20;
```

Ex #2 The query summarizes orders with a total amount greater than $100.

```
SELECT o.order_id, o.date, c.last_name,
c.first_name,
SUM(od.quantity) AS total_quantity,
SUM(od.price * od.quantity) AS total_amount
FROM Orders o, Customer c, OrderDetail od
WHERE o.customer_id = c.customer_id AND
o.order_id = od.order_id
GROUP BY o.order_id
HAVING total_amount > 100;
```

Implementation (MySQL)

2.3 Using “IN” & “EXIST” With Nested Queries

EX #1 This query retrieves the product names and prices of all products that have been ordered at least once:

```
SELECT product_name, price
FROM Product
WHERE product_id IN
        (SELECT product_id FROM
OrderDetail)
```

“In” operator used to check if a value matches any value in a list of specified values.

“EXISTS” check whether a record exists in a table or not.

EX #2 This query retrieves the names and emails of all customers who have made at least one order and paid for it using a credit card:

Solution 1

```
SELECT last_name,first_name, email
FROM Customer
WHERE EXISTS (
 SELECT * FROM Orders
 JOIN Payment ON Orders.payment_id = Payment.payment_id
 WHERE Orders.customer_id = Customer.customer_id
  AND Payment.payment_method = 'credit card)
```

Solution 2

```
SELECT last_name, first_name, email
FROM Customer
WHERE EXISTS (
    SELECT * FROM Orders, Payment
    WHERE Orders.payment_id = Payment.payment_id
    AND Orders.customer_id = Customer.customer_id
    AND Payment.payment_method = 'credit card');
```

Implementation (MySQL)

2.4 Using ALL in Nested Queries

This query that uses the ALL operator to retrieve all products that have been ordered by every customer

```
SELECT * FROM Product p
WHERE p.product_id = ALL (
                    SELECT DISTINCT od.product_id FROM OrderDetail od
                    JOIN Orders o ON od.order_id = o.order_id
                    JOIN Customer c ON o.customer_id = c.customer_id
                    WHERE p.product_id = od.product_id);
```

2.5 Using ANY in nested Queries

This query retrieves the order IDs, tracking numbers and delivery status for all orders with a delivery status of 'Delivered' or 'In transit', using a nested query with the ANY operator:

```
SELECT order_id, tracking_id, delivery_status
FROM TrackingNumber
WHERE delivery_status = ANY (
                    SELECT delivery_status
                    FROM TrackingNumber
                    WHERE delivery_status = 'Delivered' OR delivery_status = 'In transit');
```

Implementation (MySQL)

2.6 Using INDEX

2.6.1 B+ tree Index

B+ tree index on the price column of the Product table:

```
CREATE INDEX idx_product_price ON Product (price) USING BTREE
```

This index will help optimize queries that involve sorting, filtering, and searching by price.
As an example, retrieves all products with a price less than or equal to 50.00:

```
SELECT * FROM Product WHERE price <= 50.00;
```

This query should be much faster, as the database can use the index to quickly find all relevant rows, rather than scanning the entire table.

Implementation (MySQL)

2.6 Using INDEX

2.6.2 Composite index

The composite **index on the city and age** columns of the **Customer** table.

```
CREATE INDEX index_customer_city_age ON Customer(age, city);
```

To utilize this index in a query,

```
SELECT *
FROM Customer
WHERE city = 'New York' AND age > 30;
```

- This query will use the composite index to efficiently retrieve all rows from the Customer table that have a city value of 'New York' and an age value greater than 30.
- Order of fields in composite index key can be important.

Implementation (MySQL)

3 Some Simple Queries

Search the most popular product (i.e., the product that has been ordered the most):

```
SELECT p.product_name, SUM(od.quantity) as total_quantity
FROM OrderDetail od
JOIN Product p ON od.product_id = p.product_id
GROUP BY p.product_name
ORDER BY total_quantity DESC
LIMIT 1;
```

Thank You