IS5413 Final Report: Maxim’s Cake

YANG Yuxin 58671770
DONG Xiangjun 58869955
QI Yili 58893365
ZHENG Yiyang 58838131
DU Qinshu 58819313

Research Subject

Maxim's Cake: https://www.maximscakes.com.hk/en

Introduction

In recent years, Maxim’s Cake has actively employed various digitalization tools such as Cloud Computing, Artificial Intelligence, and Database Management Systems to facilitate its business operations in coping with the intensively competitive environment. The company uses these new technologies and successfully manages supply chain management, sales forecasting, and marketing strategies. To accommodate the growth of eCommerce, they keep optimizing their online ordering systems, storing user data in a well-structured database and enabling users to place orders and conveniently alter other related information.

Project Objective

The purpose of this project is to design and document a robust and scalable database system for Maxim's Cake's ordering website. This system will enable efficient management of customer orders, product details, order fulfillment, and inventory across both online and offline sales channels. The specific goals of the project proposal are:

Define Entities and Relationships: Clearly define and document the entities relevant to Maxim's Cake's business operations, including Users, Products, Orders, Order Details, Pick-ups, Deliveries, and Offline Shops, along with their attributes and relationships. Attributes serving as primary keys (PK) or foreign keys (FK) will be specified.

Conceptual Database Design: Develop a detailed conceptual schema using both an Enterprise Entity-Relationship (ER) model and an Extended Entity-Relationship (EER) model to capture the essence of Maxim's Cake's business processes and data requirements for product ordering.

Logical Database Design: Translate the conceptual schema into a logical schema that adheres to the principles of normalization to ensure data integrity, minimize redundancy, and maintain the database design in the Third Normal Form (3NF).

Implementation: Denormalize the final design and implement the logical design in MySQL Server. This involves creating tables, inserting values, designing 10 SQL queries, and showcasing the execution results.

Define Entities and Relationships

1. Entity Definitions

1.1 Users: Individuals who place orders on the Maxim's Cake ordering website.

Attribute	Description	Data Type
User_ID (PK)	Unique id for each user	INT
Name	User name	VARCHAR(50)
Birth	Date of birth	DATE
Address	Split into: Address_Line_One, Address_Line_Two, Address_District	VARCHAR(100)
Phone	8-digit HK phone number	VARCHAR(8)
Email	Email address	VARCHAR(50)
Octopus_ID	Octopus ID number	VARCHAR(20)
Registration_Date	The day the user successfully registered	DATE

1.2 Product: Bakery products offered by Maxim's Cake.

Attributes	Description	Data Type
Product_ID (PK)	Unique alphanumeric code for each product (length of 6)	INT
Product_Name	Product name	VARCHAR(100)
Main_Category	Main category of the product (e.g., “Cake” or “Bread”)	VARCHAR(50)
Subcategory	Subcategory of the product (e.g., “Fresh Fruit” or “Chocolate Lover” under the main category “Cake”)	VARCHAR(50)
Product_Description (optional)	Detailed description of the product including its ingredients (optional; only applicable to products in main category “Cake”)	TEXT
Weight (optional)	Size/weight of the product in Lb. (optional; only applicable to products in main category “Cake”)	DECIMAL(10, 2)
Fastest_Pickup_Time	The earliest time when the product is ready for pickup	TIME
Price	Product price	DECIMAL(15, 2)
StockQuantity	The total number of products in stock, which equals the	INT

	sum of the product quantities in storage across all stores.	

1.3 Orders: Record of a user's purchase from the website.

Attribute	Description	Data Type
Order_ID (PK)	Unique id for each order	INT
Total_Price	Total price before adding any discounts	DECIMAL(10, 2),
Discount	Discounted amount	DECIMAL(10, 2)
Order_Date_Time	Exact order date and time	DATETIME
Order_Type	Discriminator for subtypes: “P” for Pick-up and “D” for Delivery	CHAR(1)
Payment_Method	Payment method includes: Credit Card, Octopus, Cash	VARCHAR(50)
User_ID (FK)	Unique id for each user	INT
Shop_ID (FK)	Unique 4-digit id for an offline shop	INT

1.4 Order_Detail: One specific product and its quantity purchased within an order.

Attribute	Description	Data Type
OrderID (PK; FK)	Unique id for each order	INT
ProductID (PK; FK)	Unique alphanumeric code of each product (length of 6)	INT
Quantity	The quantity of the product purchased	INT

1.5 Pick_up: Details regarding the user's decision to pick up their order from a physical store.

Attribute	Description	Data Type
OrderID (PK; FK)	Unique id for each order	INT
Pick_up_Date	The exact day for pickup	DATE
Pick_up_Time	The exact time slot for pickup	TIME

1.6 Delivery: Information about the delivery of an order to the user's specified address.

Attribute	Description	Data Type
OrderID (PK; FK)	Unique id for each order	INT
Delivery_Date	Exact date for delivery	DATE
Delivery_Time	Exact time slot for delivery	TIME
Delivery_Address	Split into: Delivery_Address_Line_ One,	VARCHAR(100)

	Delivery_Address_Line_Two, Delivery_Address_District	
Recipient_Name	Recipient's name	VARCHAR(50)
Recipient_Phone	Recipient's phone number	VARCHAR(8)
Delivery_Fee	Delivery fee	DECIMAL(10, 2)

1.7 Offline_Shop: Physical stores where customers can pick up their orders.

Attribute	Description	Data Type
Shop_ID (PK)	Unique 4-digit id for each offline shop	INT
Shop_Area	The area of the shop	VARCHAR(50)
Shop_Address	The detailed address of the shop. Split into: Shop_Address_Line_One, Shop_Address_Line_Two, Shop_Address_District	VARCHAR(100)
Business_Hours	Business hours of the shop	VARCHAR(50)
Contact_Number	Contact number of the shop	VARCHAR(8)

2. Relationship Definitions

2.1 User and Order (One-to-Many): A User can place multiple Orders, but each Order is placed by one User.

2.2 Order and Order detail (One-to-Many): An Order can contain multiple Order details, but each Order detail is contained in one Order. Cardinality Constraints - Mandatory Many: An order must contain for at least one order detail and can contain many. Mandatory One: An order detail is contained in one and only order.

2.3 Product and Order detail (One-to-Many): A Product can be included in multiple OrderDetails, but an OrderDetail can contain only one Product.

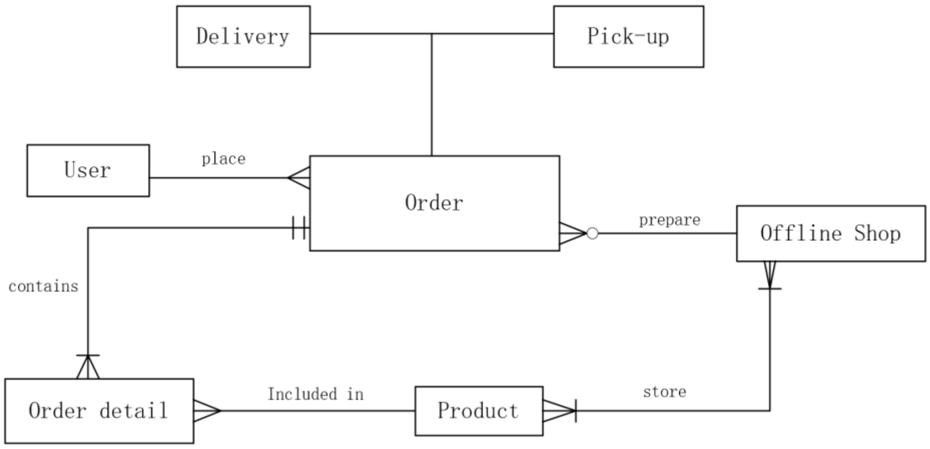
2.4 Order, Pick-up and Delivery (Order & Pick-up: One-to-One; Order & Delivery: One-to-One): The Order entity has two mutually exclusive subtypes: Pick-up and Delivery. Each order will be either a pick-up or a delivery, but not both. For example, an Order can have one Delivery/Pick-up record, and each Delivery/Pick-up record is associated with one Order. Thus, the relationship between this supertype and these subtypes fully adheres to disjoint specialization and total specialization.

2.5 Order and Offline shop (Many-to-One): A Order is prepared by one Offline Shop, but an Offline Shop can prepare for multiple Orders. Cardinality Constraints - Optional Many: An offline shop may prepare for any number of orders, or may not prepare any at all.

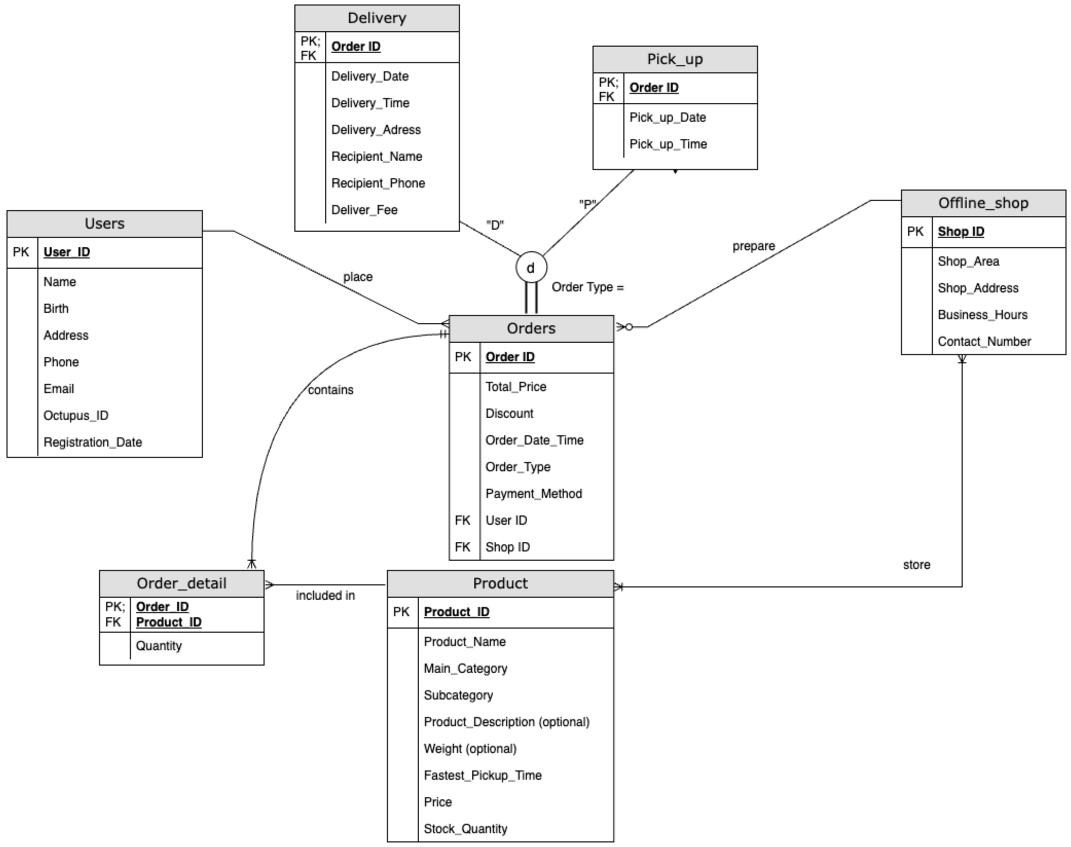
2.6 Product and Offline shop (Many-to-Many): A Product can be stored at many Offline Shops, and an Offline Shop can store many Products. Cardinality Constraints - Mandatory Many: An offline shop must prepare for at least one product and can prepare many. A product must be stored in at least one offline shop and can be stored in many.

Conceptual Database Design

1. Enterprise ER Model

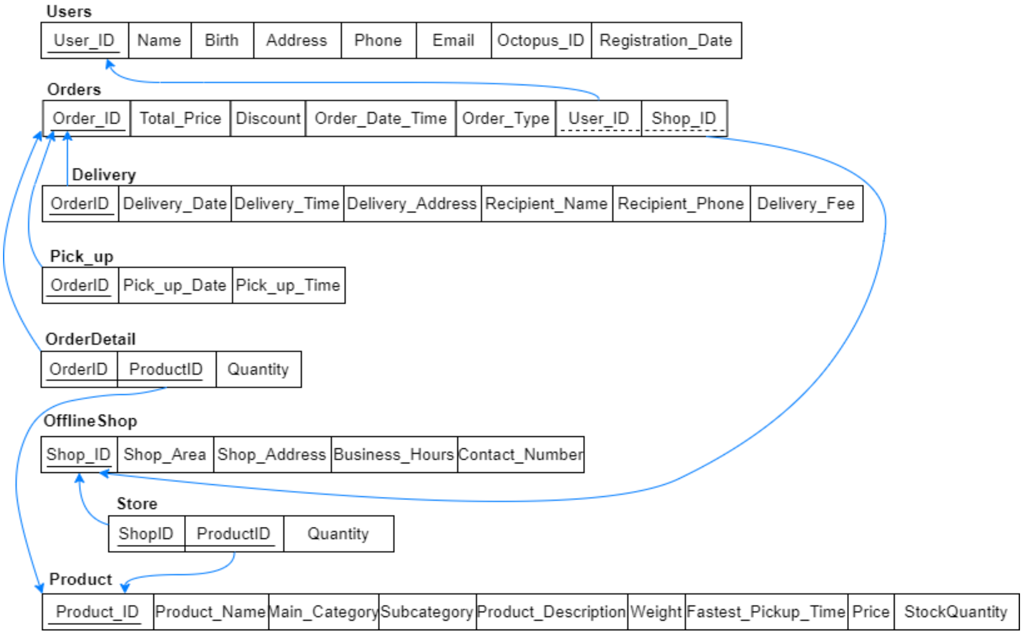


2. EER Diagram



Logical Database Design

1. Relations derived from EER



Users

User_ID	Name	Birth	Address	Phone	Email	Octopus_ID	Registration_Date

Orders

Order_ID	Total_Price	Discount	Order_Date_Time	Order_Type	User_ID	Shop_ID

Delivery

OrderID	Delivery_Date	Delivery_Time	Delivery_Address	Recipient_Name	Recipient_Phone	Delivery_Fee

Pick_up

OrderID	Pick_up_Date	Pick_up_Time

OrderDetail

OrderID	ProductID	Quantity

OfflineShop

Shop_ID	Shop_Area	Shop_Address	Business_Hours	Contact_Number

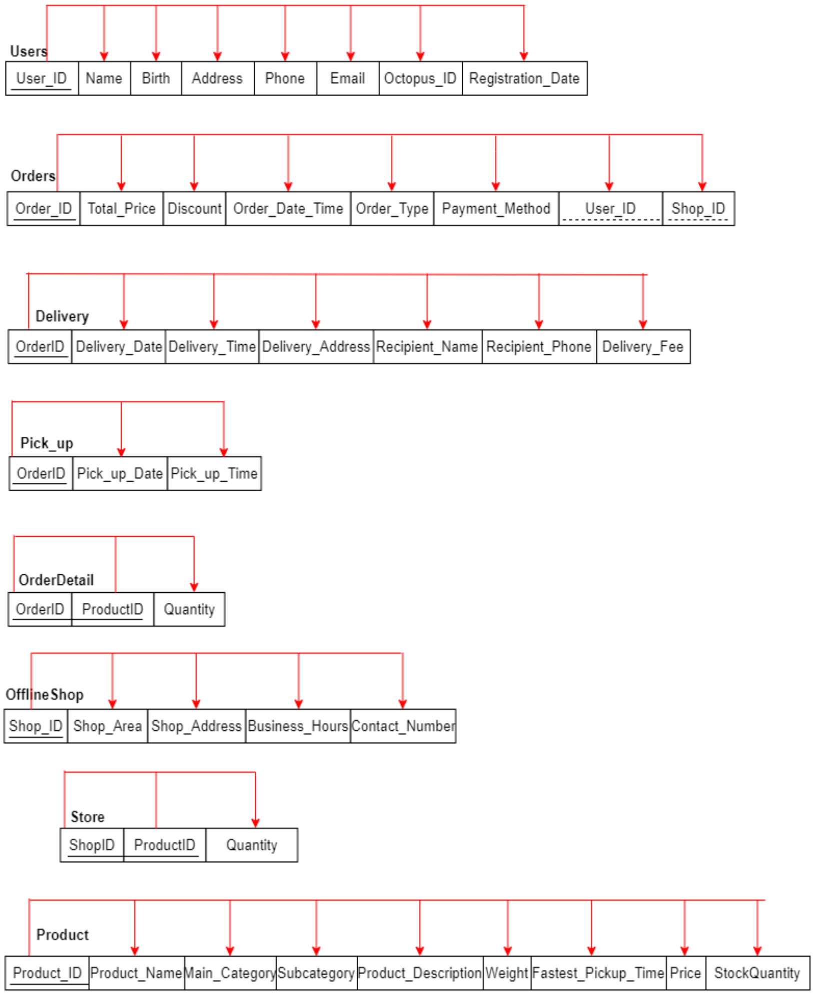
Store

ShopID	ProductID	Quantity

Product

Product_ID	Product_Name	Main_Category	Subcategory	Product_Description	Weight	Fastest_Pickup_Time	Price	StockQuantity

2. Well-structured Relations (3NF)



Users

User_ID	Name	Birth	Address	Phone	Email	Octopus_ID	Registration_Date

Orders

Order_ID	Total_Price	Discount	Order_Date_Time	Order_Type	Payment_Method	User_ID	Shop_ID

Delivery

OrderID	Delivery_Date	Delivery_Time	Delivery_Address	Recipient_Name	Recipient_Phone	Delivery_Fee

Pick_up

OrderID	Pick_up_Date	Pick_up_Time

OrderDetail

OrderID	ProductID	Quantity

OfflineShop

Shop_ID	Shop_Area	Shop_Address	Business_Hours	Contact_Number

Store

ShopID	ProductID	Quantity

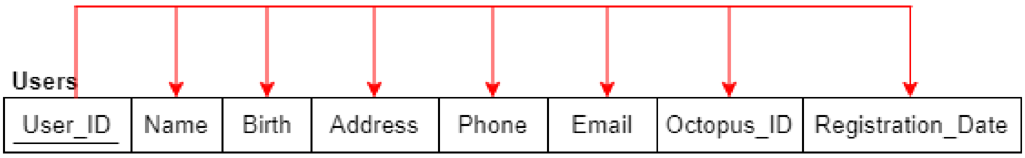
Product

Product_ID	Product_Name	Main_Category	Subcategory	Product_Description	Weight	Fastest_Pickup_Time	Price	StockQuantity

1NF Analysis:

Because all of our attributes are atomized and we have confirmed no multivalued attributes, our database design satisfies 1NF.

For example, the attributes contained in USER entity User_ID (PK), Name, Birth, Address, Email, Phone, Octopus_ID all only have one single value, as a customer can only enter one value when creating their account, and no extra value will be added after they successfully create the account.



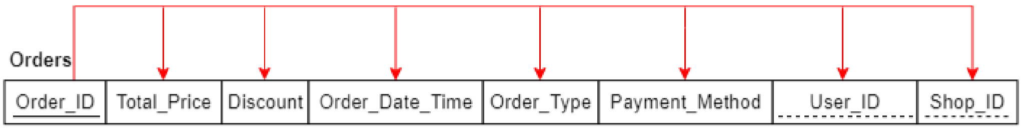
1NF-Example: User

2NF Analysis:

No partial dependency exists, as seen in our relations. All the non-key fields depend on the ENTIRE primary key only, so our design meets the requirement of 2NF.

For example, in the Order relationship, Order ID is the primary key and all other attributes such as Total Price, Discount, etc. are fully dependent on Order ID.

We did not find any partial dependencies, i.e., all the non-key fields do not depend on a part of the primary key.

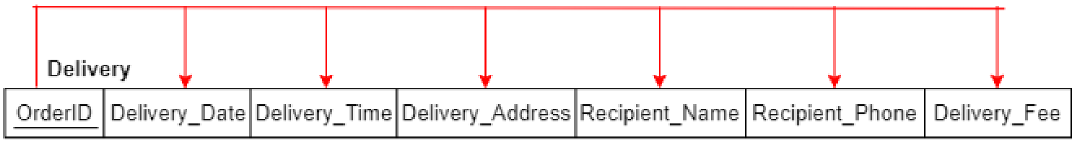


2NF-Example: Order

3NF Analysis:

Our model already satisfies the requirements of First Normal Form (1NF) and Second Normal Form (2NF). To meet the requirement of 3NF, it must not have any transitive dependencies, which means that no non-prime attribute should depend on another non-prime attribute. All non-prime attributes must depend directly on the candidate key(s) of the relation.

For example, Order ID is the primary key, and the other attributes such as Delivery_Date, Delivery_Time are all non-prime attributes in the Delivery table. These non-primary attributes don’t rely on the other non-primary attributes to depend on Order ID , all of them depend directly on Order ID.



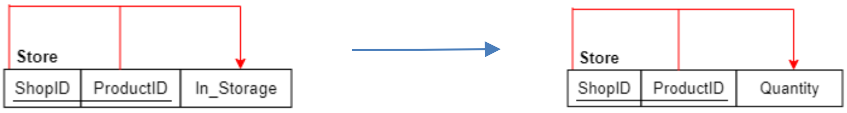
3NF-Example: Delivery

In conclusion, our database design meets the requirements of well-structured relations with minimal data redundancy and data integrity.

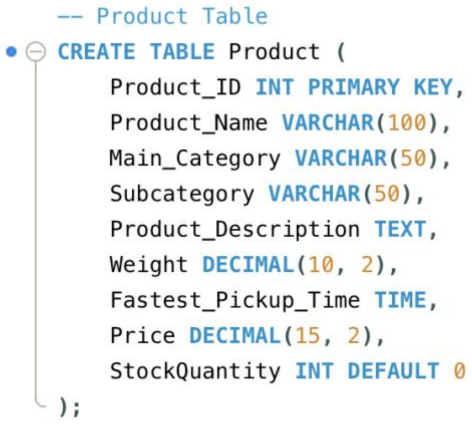
Implementation

1. Denormalization

In the initial version of our report, we included an attribute called "In_Storage" in the "Store" table. In the final implementation phase, we replaced it with a “Quantity” attribute which not only carry information on “whether the product is in storage of the offline shop or not”, but also specifies the exact quantity of products stored.



In order to simplify queries, we applied denormalization to our design and added a “StockQuantity” attribute to the Product table which shows the total quantity of a product stored at all offline shops. However, this may cause potential data redundancy since the “StockQuantity” is a derived attribute (i.e., it equals the sum of “Quantity” recorded in the “Store” table where the “Product_ID” is the same) which should not be included in an efficient database design. This may also cause data inconsistency when updating either the “Store” table or the “Product” table alone.



```
-- Product Table
CREATE TABLE Product (
    Product_ID INT PRIMARY KEY,
    Product_Name VARCHAR(100),
    Main_Category VARCHAR(50),
    Subcategory VARCHAR(50),
    Product_Description TEXT,
    Weight DECIMAL(10, 2),
    Fastest_Pickup_Time TIME,
    Price DECIMAL(15, 2),
    StockQuantity INT DEFAULT 0
);
```

2. Normalized Relational Database Design

```
SHOW VARIABLES LIKE 'read_only';
SET GLOBAL read_only = OFF;
CREATE DATABASE MaximsCakeDB;
USE MaximsCakeDB;

-- Users Table
CREATE TABLE Users (
    User_ID INT PRIMARY KEY,
    Name VARCHAR(50),
    Birth DATE,
    Address VARCHAR(100),
    Phone VARCHAR(8),
    Email VARCHAR(50),
    Octopus_ID VARCHAR(20),
    Registration_Date DATE
);

-- Product Table
CREATE TABLE Product (
    Product_ID INT PRIMARY KEY,
    Product_Name VARCHAR(100),
    Main_Category VARCHAR(50),
    Subcategory VARCHAR(50),
    Product_Description TEXT NULL,
    Weight DECIMAL(10, 2) NULL,
    Fastest_Pickup_Time TIME,
    Price DECIMAL(15, 2),
    StockQuantity INT DEFAULT 0
);

-- Offline Shop Table
CREATE TABLE Offline_Shop (
    Shop_ID INT PRIMARY KEY,
    Shop_Area VARCHAR(50),
    Shop_Address VARCHAR(100),
    Business_Hours VARCHAR(50),
    Contact_Number VARCHAR(8)
);

-- Store Table
CREATE TABLE Store (
    Shop_ID INT,
    Product_ID INT,
    Quantity INT DEFAULT 0,
    PRIMARY KEY (Shop_ID, Product_ID),
    FOREIGN KEY (Product_ID) REFERENCES Product(Product_ID),
    FOREIGN KEY (Shop_ID) REFERENCES Offline_Shop(Shop_ID)
);

-- Orders Table
CREATE TABLE Orders (
    Order_ID INT PRIMARY KEY,
    Total_Price DECIMAL(10, 2),
    Discount DECIMAL(10, 2),
    Order_Date_Time DATETIME,
    Order_Type CHAR(1), -- 'D' for Delivery, 'P' for Pick-up
    Payment_Method VARCHAR(50),
    User_ID INT,
    Shop_ID INT,
    FOREIGN KEY (User_ID) REFERENCES Users(User_ID),
    FOREIGN KEY (Shop_ID) REFERENCES Offline_Shop(Shop_ID)
);

-- Order Detail Table
CREATE TABLE Order_Detail (
    Order_ID INT,
    Product_ID INT,
    Quantity INT,
    PRIMARY KEY (Order_ID, Product_ID),
    FOREIGN KEY (Order_ID) REFERENCES Orders(Order_ID),
    FOREIGN KEY (Product_ID) REFERENCES Product(Product_ID)
);

-- Delivery Table
CREATE TABLE Delivery (
    Order_ID INT PRIMARY KEY,
    Delivery_Date DATE,
    Delivery_Time TIME,
    Delivery_Address VARCHAR(100),
    Recipient_Name VARCHAR(50),
    Recipient_Phone VARCHAR(8),
    Delivery_Fee DECIMAL(10, 2),
    FOREIGN KEY (Order_ID) REFERENCES Orders(Order_ID)
);

-- Pick-up Table
CREATE TABLE Pick_up (
    Order_ID INT PRIMARY KEY,
    Pick_up_Date DATE,
    Pick_up_Time TIME,
    FOREIGN KEY (Order_ID) REFERENCES Orders(Order_ID)
);
```

3. Inserting value to the database

```
INSERT INTO Users (User_ID, Name, Birth, Address, Phone, Registration_Date) VALUES
(1, 'John Wong', '1990-01-01', 'Sai Ying Poon, Hongkong Island', '91234567', '2024-09-05'),
(2, 'Alice Chan', '1992-05-15', 'Port Centre, Hongkong Island', '92345678', '2024-09-12'),
(3, 'Mike Lee', '1988-08-22', 'Shek Tong Tsui, Hongkong Island', '93456789', '2024-09-22'),
(4, 'Emma Tam', '1993-11-05', 'New Jade, Hongkong Island', '94567890', '2024-10-01'),
(5, 'Daniel Cheung', '1985-07-22', 'Wah Fu Estate, Hongkong Island', '95678901', '2024-10-12'),
……

INSERT INTO Product (Product_ID, Product_Name, Main_Category, Subcategory, Price) VALUES
(1, 'My Melody & Kuromi Strawberry Cream Cake', 'Order Cakes', 'WINTER WONDERLAND CHRISTMAS COLLECTION', 248.00),
(2, 'Hangyodon Black Forest Cake', 'Order Cakes', 'WINTER WONDERLAND CHRISTMAS COLLECTION', 248.00),
(3, 'Sanrio characters Mixed Fruit Cake', 'Order Cakes', 'WINTER WONDERLAND CHRISTMAS COLLECTION', 288.00),
(4, 'Hazelnut Black Forest Cake', 'Order Cakes', 'NEW CAKES', 238.00),
......

UPDATE Product SET StockQuantity = 45 WHERE Product_ID IN (35, 36, 37); -- Bread & Packaged Product (35-37)
UPDATE Product SET StockQuantity = 45 WHERE Product_ID IN (38, 39, 40); -- Bread & Packaged Product (38-40)
UPDATE Product SET StockQuantity = 45 WHERE Product_ID IN (41, 42, 43); -- Bread & Packaged Product (41-43)
UPDATE Product SET StockQuantity = 45 WHERE Product_ID IN (44, 45, 46, 47); -- Assorted Cake and Dessert (44-47)
UPDATE Product SET StockQuantity = 45 WHERE Product_ID IN (48, 49, 50, 51); -- Assorted Cake and Dessert (48-51)
UPDATE Product SET StockQuantity = 45 WHERE Product_ID IN (52, 53, 54, 55, 56, 57, 58, 59);
UPDATE Product SET StockQuantity = 30 WHERE Product_ID IN (55); -- Assorted Cake and Dessert (52-59)
UPDATE Product SET StockQuantity = 45 WHERE Product_ID IN (60, 61); -- Assorted Cake and Dessert (60-61)
UPDATE Product SET StockQuantity = 45 WHERE Product_ID IN (62, 63); -- Assorted Cake and Dessert (62-63)
UPDATE Product SET StockQuantity = 15 WHERE Product_ID IN (1, 2, 3, 4, 5, 6);-- Order Cakes
UPDATE Product SET StockQuantity = 15 WHERE Product_ID IN (7, 8, 9, 10, 11, 12);-- Order Cakes
UPDATE Product SET StockQuantity = 15 WHERE Product_ID IN (13, 14, 15, 16, 17, 18);-- Order Cakes
UPDATE Product SET StockQuantity = 15 WHERE Product_ID IN (19, 20, 21, 22, 23, 24);-- Order Cakes
UPDATE Product SET StockQuantity = 15 WHERE Product_ID IN (25, 26, 27, 28, 29, 30);-- Order
```

```
Cakes
UPDATE Product SET StockQuantity = 15 WHERE Product_ID IN (31, 32, 33, 34); -- Order Cakes

INSERT INTO Store (Shop_ID, Product_ID, Quantity)
SELECT s.Shop_ID, p.Product_ID, 0
FROM Offline_Shop s
CROSS JOIN Product p
WHERE p.Product_ID BETWEEN 1 AND 63;

UPDATE Store SET Quantity = 3 WHERE Product_ID IN (35, 36, 37); -- Bread & Packaged Product (35-37)
UPDATE Store SET Quantity = 3 WHERE Product_ID IN (38, 39, 40); -- Bread & Packaged Product (38-40)
UPDATE Store SET Quantity = 3 WHERE Product_ID IN (41, 42, 43); -- Bread & Packaged Product (41-43)
UPDATE Store SET Quantity = 3 WHERE Product_ID IN (44, 45, 46, 47, 48, 49, 50, 51); -- Assorted Cake and Dessert (44-51)
UPDATE Store SET Quantity = 3 WHERE Product_ID IN (52, 53, 54, 56, 57, 58, 59);
UPDATE Store SET Quantity = 2 WHERE Product_ID IN (55);-- Assorted Cake and Dessert (52-59)
UPDATE Store SET Quantity = 3 WHERE Product_ID IN (60, 61); -- Assorted Cake and Dessert (60-61)
UPDATE Store SET Quantity = 3 WHERE Product_ID IN (62, 63); -- Assorted Cake and Dessert (62-63)
UPDATE Store SET Quantity = 1 WHERE Product_ID IN (1, 2, 3, 4, 5, 6);
UPDATE Store SET Quantity = 1 WHERE Product_ID IN (7, 8, 9, 10, 11, 12);
UPDATE Store SET Quantity = 1 WHERE Product_ID IN (13, 14, 15, 16, 17, 18);
UPDATE Store SET Quantity = 1 WHERE Product_ID IN (19, 20, 21, 22, 23, 24);
UPDATE Store SET Quantity = 1 WHERE Product_ID IN (25, 26, 27, 28, 29, 30);
UPDATE Store SET Quantity = 1 WHERE Product_ID IN (31, 32, 33, 34); -- Order Cakes (1-34)

INSERT INTO Orders (Order_ID, Total_Price, Discount, Order_Date_Time, Order_Type, User_ID, Shop_ID, Payment_Method) VALUES
(1, 277.00, 0.00, '2024-11-01 08:30:00', 'P', 21, 3112, 'Credit Card'),
(2, 545.00, 0.00, '2024-11-01 09:45:00', 'D', 7, 3116, 'Octopus'),
(3, 483.50, 0.00, '2024-11-02 08:00:00', 'D', 12, 3130, 'Credit Card'),
(4, 51.50, 0.00, '2024-11-02 09:15:00', 'D', 22, 3138, 'Octopus'),
(5, 440.00, 0.00, '2024-11-03 08:30:00', 'D', 3, 3154, 'Cash'),
......

INSERT INTO Order_Detail (Order_ID, Product_ID, Quantity) VALUES
(1, 35, 2),    -- 3-mixed Cheese Ring Bread
(1, 36, 1),    -- Ovaltine Creamy Stick
(1, 1, 1),     -- My Melody & Kuromi Strawberry Cream Cake
```

```
(2, 37, 3),    -- Low Sugar Brown Rice and Walnut Bun
......

INSERT INTO Delivery (Order_ID, Delivery_Date, Delivery_Time, Delivery_Address, Recipient_Name, Recipient_Phone, Delivery_Fee) VALUES
(2, '2024-11-02', '09:45:00', 'Port Centre, Hongkong Island', 'Alice Chan', '92345678', 5.00),
(3, '2024-11-03', '08:00:00', 'Shek Tong Tsui, Hongkong Island', 'Mike Lee', '93456789', 10.00),
(4, '2024-11-03', '09:15:00', 'New Jade, Hongkong Island', 'Emma Tam', '94567890', 5.00),
......

INSERT INTO Pick_up (Order_ID, Pick_up_Date, Pick_up_Time) VALUES
(1, '2024-11-01', '09:00:00'),
(7, '2024-11-04', '09:30:00'),
(8, '2024-11-04', '10:00:00'),
......
```

4. Non-Unique Indexes for Frequently Accessed or Sorted Columns

Order Detail Index: Created an index on Product_ID in the Order_Detail table to optimize product-related queries.

Composite Shop Order Index: Created a composite index on Shop_ID and Total_Price in the Orders table to optimize queries and analyses by store and total price.

User Information Indexes: Separate indexes on Address and Birth fields in the Users table. Index on Registration_Date for user queries by registration date. These indexes help improve user information retrieval efficiency

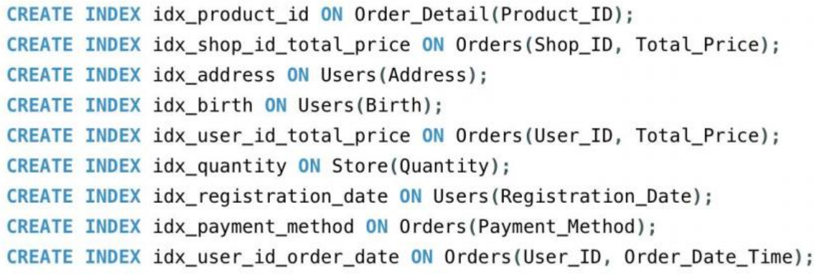
Order-Related Indexes: Composite index on User_ID and Total_Price. Index on Payment_Method for payment method analysis. Composite index on User_ID and Order_Date_Time to optimize order time queries

Inventory Index: Index on Quantity field in the Store table for quick inventory level queries.

In this setup:

1. Appropriate use of composite indexes to optimize multi-field queries;
2. Focus on time-related fields needed for user behavior analysis;
3. Consideration of core query requirements for the order system;
4. Basic needs for inventory and product management are covered.

Note: The design of these indexes should be adjusted based on actual query frequency and performance requirements.

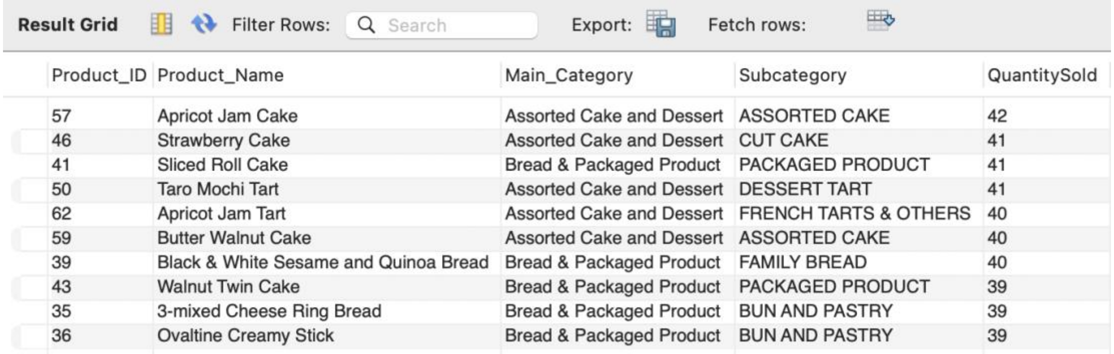


```
CREATE INDEX idx_product_id ON Order_Detail(Product_ID);
CREATE INDEX idx_shop_id_total_price ON Orders(Shop_ID, Total_Price);
CREATE INDEX idx_address ON Users(Address);
CREATE INDEX idx_birth ON Users(Birth);
CREATE INDEX idx_user_id_total_price ON Orders(User_ID, Total_Price);
CREATE INDEX idx_quantity ON Store(Quantity);
CREATE INDEX idx_registration_date ON Users(Registration_Date);
CREATE INDEX idx_payment_method ON Orders(Payment_Method);
CREATE INDEX idx_user_id_order_date ON Orders(User_ID, Order_Date_Time);
```

5. Ten Queries using SQL statements

Query 1: Find the most popular products (top 10)

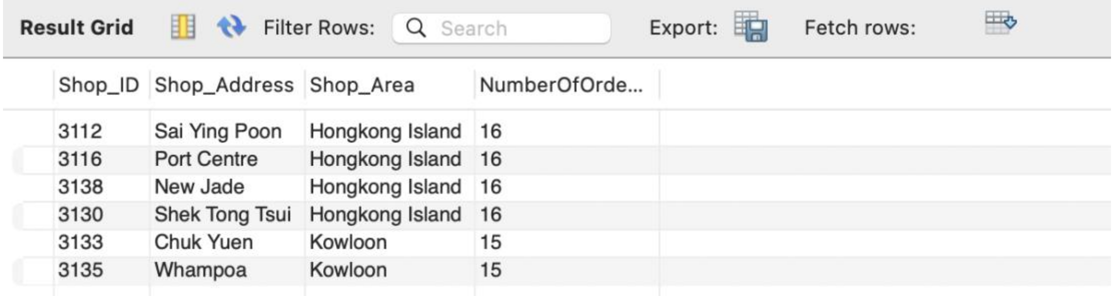
```
SELECT OD.Product_ID, P.Product_Name, P.Main_Category, P.Subcategory, SUM(OD.Quantity)
AS QuantitySold
FROM Product AS P
JOIN Order_Detail AS OD ON P.Product_ID = OD.Product_ID
GROUP BY OD.Product_ID, P.Product_Name, P.Main_Category, P.Subcategory
ORDER BY QuantitySold DESC
LIMIT 10;
```



Product_ID	Product_Name	Main_Category	Subcategory	QuantitySold
57	Apricot Jam Cake	Assorted Cake and Dessert	ASSORTED CAKE	42
46	Strawberry Cake	Assorted Cake and Dessert	CUT CAKE	41
41	Sliced Roll Cake	Bread & Packaged Product	PACKAGED PRODUCT	41
50	Taro Mochi Tart	Assorted Cake and Dessert	DESSERT TART	41
62	Apricot Jam Tart	Assorted Cake and Dessert	FRENCH TARTS & OTHERS	40
59	Butter Walnut Cake	Assorted Cake and Dessert	ASSORTED CAKE	40
39	Black & White Sesame and Quinoa Bread	Bread & Packaged Product	FAMILY BREAD	40
43	Walnut Twin Cake	Bread & Packaged Product	PACKAGED PRODUCT	39
35	3-mixed Cheese Ring Bread	Bread & Packaged Product	BUN AND PASTRY	39
36	Ovaltine Creamy Stick	Bread & Packaged Product	BUN AND PASTRY	39

Query 2: Find the top 6 offline stores with the highest number of orders.

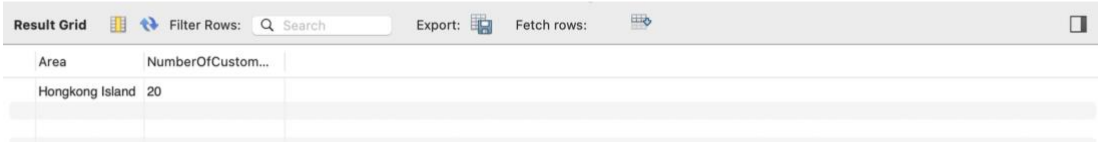
```
SELECT S.Shop_ID, S.Shop_Address, S.Shop_Area, COUNT(O.Shop_ID) AS NumberOfOrders
FROM Offline_Shop AS S
LEFT JOIN Orders AS O ON O.Shop_ID = S.Shop_ID
GROUP BY S.Shop_ID, S.Shop_Address, S.Shop_Area
ORDER BY NumberOfOrders DESC
LIMIT 6;
```



Shop_ID	Shop_Address	Shop_Area	NumberOfOrde...
3112	Sai Ying Poon	Hongkong Island	16
3116	Port Centre	Hongkong Island	16
3138	New Jade	Hongkong Island	16
3130	Shek Tong Tsui	Hongkong Island	16
3133	Chuk Yuen	Kowloon	15
3135	Whampoa	Kowloon	15

Query 3: Find the shop area with the highest concentration of customers (/Users).

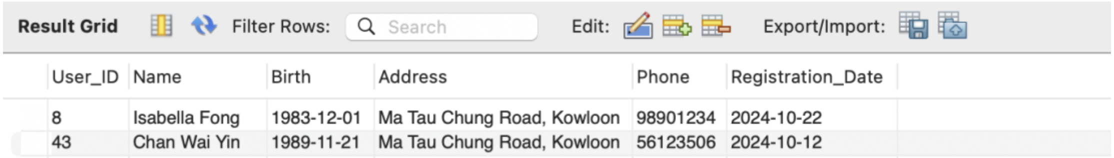
```
select dsa.Shop_Area as Area, count(u.User_ID) as NumberOfCustomers
from Users as u
join (
        select distinct Shop_Area
        from Offline_Shop
        ) as dsa
on u.Address like concat('%', dsa.Shop_Area)
group by dsa.Shop_Area
order by 2 desc
LIMIT 1;
```



Area	NumberOfCustom...
Hongkong Island	20

Query 4: Find the customers whose birthdays are coming within a month (i.e., 30 days following the current calendar date)

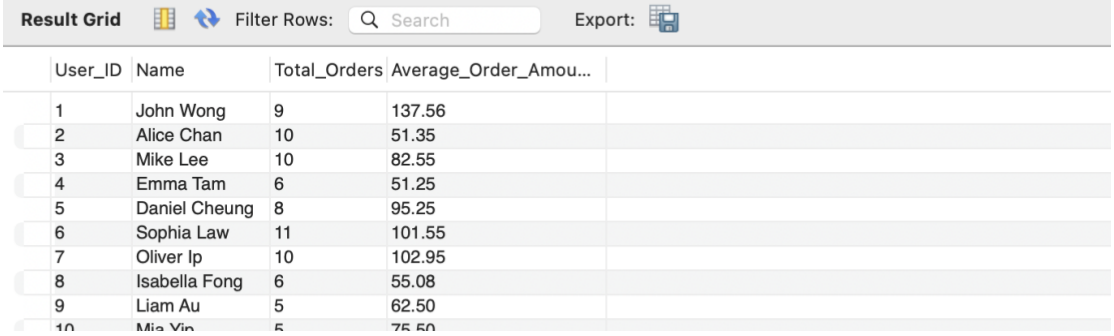
```
SELECT User_ID, Name, Birth, Address, Phone, Registration_Date
FROM users
WHERE DATE_FORMAT(Birth, '%m-%d') BETWEEN DATE_FORMAT(CURDATE(), '%m-%d') AND DATE_FORMAT(DATE_ADD(CURDATE(), INTERVAL 30 DAY), '%m-%d');
```



User_ID	Name	Birth	Address	Phone	Registration_Date
8	Isabella Fong	1983-12-01	Ma Tau Chung Road, Kowloon	98901234	2024-10-22
43	Chan Wai Yin	1989-11-21	Ma Tau Chung Road, Kowloon	56123506	2024-10-12

Query 5: Query the average order amount for each user

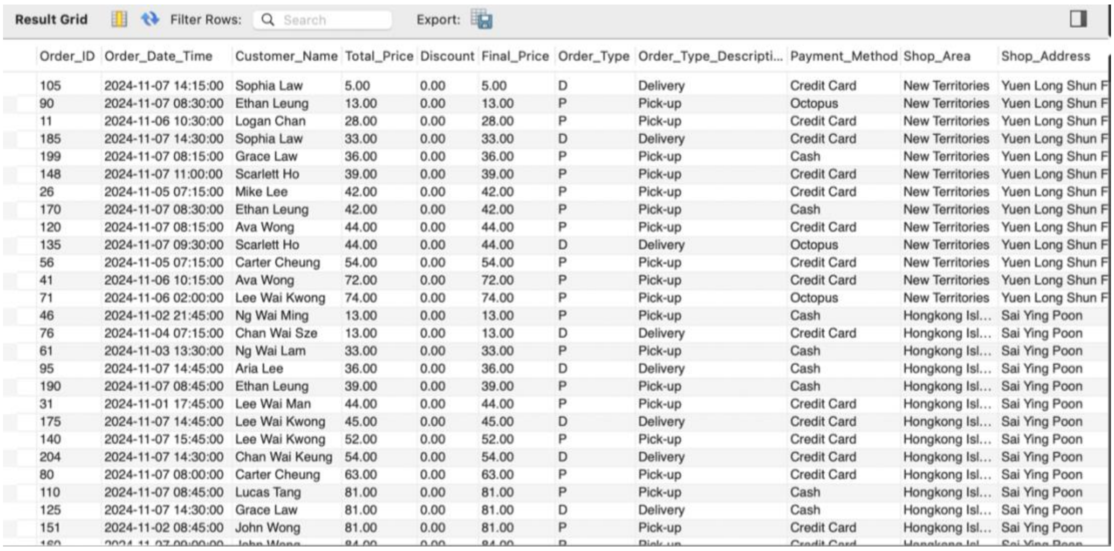
```
SELECT
    u.User_ID,
    u.Name,
    COUNT(o.Order_ID) as Total_Orders,
    ROUND(AVG(o.Total_Price), 2) as Average_Order_Amount
FROM
    Users u
    LEFT JOIN Orders o ON u.User_ID = o.User_ID
GROUP BY
    u.User_ID, u.Name
ORDER BY
    u.User_ID;
```



User_ID	Name	Total_Orders	Average_Order_Amou...
1	John Wong	9	137.56
2	Alice Chan	10	51.35
3	Mike Lee	10	82.55
4	Emma Tam	6	51.25
5	Daniel Cheung	8	95.25
6	Sophia Law	11	101.55
7	Oliver Ip	10	102.95
8	Isabella Fong	6	55.08
9	Liam Au	5	62.50
10	Mia Yin	5	75.50

Query 6: Create a view to display order information

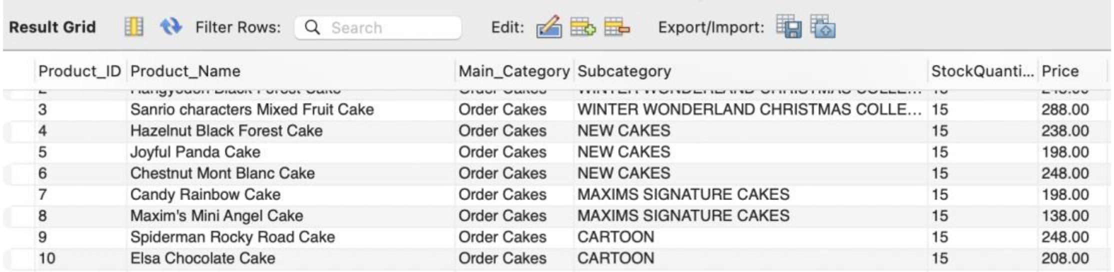
```
CREATE VIEW Order_Information AS
SELECT
    o.Order_ID,
    o.Order_Date_Time,
    u.Name as Customer_Name,
    o.Total_Price,
    o.Discount,
    (o.Total_Price - COALESCE(o.Discount, 0)) as Final_Price,
    o.Order_Type,
    CASE
        WHEN o.Order_Type = 'D' THEN 'Delivery'
        WHEN o.Order_Type = 'P' THEN 'Pick-up'
    END as Order_Type_Description,
    o.Payment_Method,
    os.Shop_Area,
    os.Shop_Address,
    CASE
        WHEN o.Order_Type = 'D' THEN d.Delivery_Address
        ELSE os.Shop_Address
    END as Delivery_Pick_up_Location,
    CASE
        WHEN o.Order_Type = 'D' THEN d.Delivery_Date
        ELSE p.Pick_up_Date
    END as Fulfillment_Date,
    CASE
        WHEN o.Order_Type = 'D' THEN d.Delivery_Time
        ELSE p.Pick_up_Time
    END as Fulfillment_Time
FROM
    Orders o
    JOIN Users u ON o.User_ID = u.User_ID
    JOIN Offline_Shop os ON o.Shop_ID = os.Shop_ID
    LEFT JOIN Delivery d ON o.Order_ID = d.Order_ID
    LEFT JOIN Pick_up p ON o.Order_ID = p.Order_ID;
```



Order_ID	Order_Date_Time	Customer_Name	Total_Price	Discount	Final_Price	Order_Type	Order_Type_Descripti...	Payment_Method	Shop_Area	Shop_Address
105	2024-11-07 14:15:00	Sophia Law	5.00	0.00	5.00	D	Delivery	Credit Card	New Territories	Yuen Long Shun F
90	2024-11-07 08:30:00	Ethan Leung	13.00	0.00	13.00	P	Pick-up	Octopus	New Territories	Yuen Long Shun F
11	2024-11-06 10:30:00	Logan Chan	28.00	0.00	28.00	P	Pick-up	Credit Card	New Territories	Yuen Long Shun F
185	2024-11-07 14:30:00	Sophia Law	33.00	0.00	33.00	D	Delivery	Credit Card	New Territories	Yuen Long Shun F
199	2024-11-07 08:15:00	Grace Law	36.00	0.00	36.00	P	Pick-up	Cash	New Territories	Yuen Long Shun F
148	2024-11-07 11:00:00	Scarlett Ho	39.00	0.00	39.00	P	Pick-up	Credit Card	New Territories	Yuen Long Shun F
26	2024-11-05 07:15:00	Mike Lee	42.00	0.00	42.00	P	Pick-up	Credit Card	New Territories	Yuen Long Shun F
170	2024-11-07 08:30:00	Ethan Leung	42.00	0.00	42.00	P	Pick-up	Cash	New Territories	Yuen Long Shun F
120	2024-11-07 08:15:00	Ava Wong	44.00	0.00	44.00	P	Pick-up	Credit Card	New Territories	Yuen Long Shun F
135	2024-11-07 09:30:00	Scarlett Ho	44.00	0.00	44.00	D	Delivery	Octopus	New Territories	Yuen Long Shun F
56	2024-11-05 07:15:00	Carter Cheung	54.00	0.00	54.00	P	Pick-up	Credit Card	New Territories	Yuen Long Shun F
41	2024-11-06 10:15:00	Ava Wong	72.00	0.00	72.00	P	Pick-up	Credit Card	New Territories	Yuen Long Shun F
71	2024-11-06 02:00:00	Lee Wai Kwong	74.00	0.00	74.00	P	Pick-up	Octopus	New Territories	Yuen Long Shun F
46	2024-11-02 21:45:00	Ng Wai Ming	13.00	0.00	13.00	P	Pick-up	Cash	Hongkong Isl...	Sai Ying Poon
76	2024-11-04 07:15:00	Chan Wai Sze	13.00	0.00	13.00	D	Delivery	Credit Card	Hongkong Isl...	Sai Ying Poon
61	2024-11-03 13:30:00	Ng Wai Lam	33.00	0.00	33.00	P	Pick-up	Cash	Hongkong Isl...	Sai Ying Poon
95	2024-11-07 14:45:00	Aria Lee	36.00	0.00	36.00	D	Delivery	Cash	Hongkong Isl...	Sai Ying Poon
190	2024-11-07 08:45:00	Ethan Leung	39.00	0.00	39.00	P	Pick-up	Cash	Hongkong Isl...	Sai Ying Poon
31	2024-11-01 17:45:00	Lee Wai Man	44.00	0.00	44.00	P	Pick-up	Credit Card	Hongkong Isl...	Sai Ying Poon
175	2024-11-07 14:45:00	Lee Wai Kwong	45.00	0.00	45.00	D	Delivery	Credit Card	Hongkong Isl...	Sai Ying Poon
140	2024-11-07 15:45:00	Lee Wai Kwong	52.00	0.00	52.00	P	Pick-up	Credit Card	Hongkong Isl...	Sai Ying Poon
204	2024-11-07 14:30:00	Chan Wai Keung	54.00	0.00	54.00	D	Delivery	Credit Card	Hongkong Isl...	Sai Ying Poon
80	2024-11-07 08:00:00	Carter Cheung	63.00	0.00	63.00	P	Pick-up	Credit Card	Hongkong Isl...	Sai Ying Poon
110	2024-11-07 08:45:00	Lucas Tang	81.00	0.00	81.00	P	Pick-up	Cash	Hongkong Isl...	Sai Ying Poon
125	2024-11-07 14:30:00	Grace Law	81.00	0.00	81.00	D	Delivery	Cash	Hongkong Isl...	Sai Ying Poon
151	2024-11-02 08:45:00	John Wong	81.00	0.00	81.00	P	Pick-up	Credit Card	Hongkong Isl...	Sai Ying Poon
160	2024-11-07 09:00:00	John Wong	84.00	0.00	84.00	P	Pick-up	Credit Card	Hongkong Isl...	Sai Ying Poon

Query 7: Query products with low stock (StockQuantity < 20)

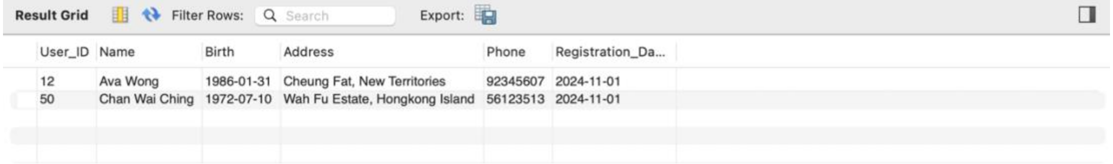
```
SELECT
        Product_ID,
        Product_Name,
        Main_Category,
        Subcategory,
        StockQuantity,
        Price
FROM
        Product
WHERE
        StockQuantity < 20
ORDER BY
        StockQuantity ASC;
```



Product_ID	Product_Name	Main_Category	Subcategory	StockQuanti...	Price
2	Hangyodon Black Forest Cake	Order Cakes	WINTER WONDERLAND CHRISTMAS COLLE...	15	[illegible]
3	Sanrio characters Mixed Fruit Cake	Order Cakes	WINTER WONDERLAND CHRISTMAS COLLE...	15	288.00
4	Hazelnut Black Forest Cake	Order Cakes	NEW CAKES	15	238.00
5	Joyful Panda Cake	Order Cakes	NEW CAKES	15	198.00
6	Chestnut Mont Blanc Cake	Order Cakes	NEW CAKES	15	248.00
7	Candy Rainbow Cake	Order Cakes	MAXIMS SIGNATURE CAKES	15	198.00
8	Maxim's Mini Angel Cake	Order Cakes	MAXIMS SIGNATURE CAKES	15	138.00
9	Spiderman Rocky Road Cake	Order Cakes	CARTOON	15	248.00
10	Elsa Chocolate Cake	Order Cakes	CARTOON	15	208.00

Query 8: Query the number of new users registered in the past week up to 11/7 and create a view.

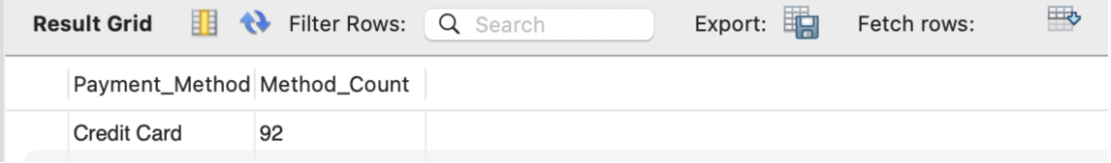
```
CREATE VIEW Recent_Registered_Users AS
SELECT User_ID, Name, Birth, Address, Phone, Registration_Date
FROM Users
WHERE Registration_Date BETWEEN DATE('2024-10-31') AND DATE('2024-11-07');
```



User_ID	Name	Birth	Address	Phone	Registration_Da...
12	Ava Wong	1986-01-31	Cheung Fat, New Territories	92345607	2024-11-01
50	Chan Wai Ching	1972-07-10	Wah Fu Estate, Hongkong Island	56123513	2024-11-01

Query 9: Query the most frequently used payment method by users.

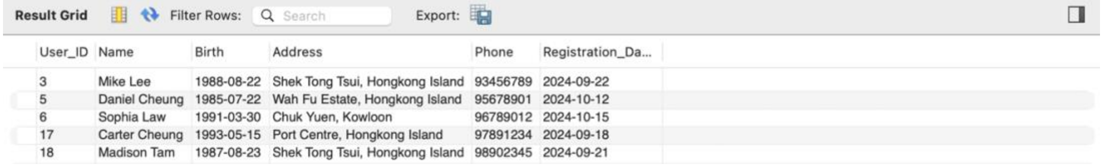
```
SELECT Payment_Method, COUNT(*) AS Method_Count
FROM Orders
GROUP BY Payment_Method
ORDER BY Method_Count DESC
LIMIT 1;
```



Payment_Method	Method_Count
Credit Card	92

Query 10: active users and create a view (active users are those who have placed three orders within the five days leading up to 11/7).

```
CREATE VIEW Active_Users AS
SELECT u.User_ID, u.Name, u.Birth, u.Address, u.Phone, u.Registration_Date
FROM Users u
JOIN (
    SELECT User_ID
    FROM Orders
    WHERE Order_Date_Time BETWEEN DATE('2024-11-03') AND DATE('2024-11-07')
    GROUP BY User_ID
    HAVING COUNT(Order_ID) >= 3
) AS active_users ON u.User_ID = active_users.User_ID;
```



User_ID	Name	Birth	Address	Phone	Registration_Da...
3	Mike Lee	1988-08-22	Shek Tong Tsui, Hongkong Island	93456789	2024-09-22
5	Daniel Cheung	1985-07-22	Wah Fu Estate, Hongkong Island	95678901	2024-10-12
6	Sophia Law	1991-03-30	Chuk Yuen, Kowloon	96789012	2024-10-15
17	Carter Cheung	1993-05-15	Port Centre, Hongkong Island	97891234	2024-09-18
18	Madison Tam	1987-08-23	Shek Tong Tsui, Hongkong Island	98902345	2024-09-21