Business Domain Study and Database Design Project

(M4: ERD Transformation, Database Implementation, and Data Query)

Fast Food Restaurant Domain: The Pizza Company

Fast Food Restaurant Domain: The Pizza Company

Napat Tatiyakaroonwong	6522781804
Haseeb Ali	6522790136
Natchapol Lebkrut	6522781622
Napat Decha	6522770880
Vayuphak Saengthong	6522771086
Chatdanai Wongsuwan	6522781747

CSS325: Database System

Asst. Prof. Dr. Preecha Tangworakitthaworn

Sirindhorn International Institute of Technology

Thammasat University

30 November 2024

Abstract

This report deals with the design and the establishment of database processes for a fast food industry with the help of a particular fast food outlet, “The Pizza Company”. The study starts with designing a new Entity-Relationship Diagram (ERD) that has some critical entities, their attributes and relationships. After that, this ERD is systematically converted into a relational schema, which includes primary binary relationships (1:1, 1:M, M:N), multivalued attributes, specialization/generalization, etc. Finally, a schema with full normalization to serve the operational structure of the business is created.

The next step is to write the actual implementation of the database in SQL and provide the scripts of creating tables, inserting records, and making queries. The architecture of the tables, along with their attributes and constraints, is recorded in the data dictionary, offering uniformity and comprehensiveness throughout the design. For specialized functions such as the control of orders, customers and their clients, the stock, and the delivery of goods, specific queries are created.

This project illustrates how the query tools can relay the information in the data base so as to carry out the required procedures more effectively while observing the integrity of data. Using SQL, the system offers several functions including but not limited to receiving customer orders, managing sale and payment, and overseeing merchandise delivery. In connection to this, this work presents the manner in which data can be organized to enhance the operational aspects of fast food services in real time.

Table of Contents

List of Figures

1. Introduction of Business domain

The Pizza Company grows by using both franchises and owned branches. Franchisees run their own branches with support like training and marketing, while the company directly owns some branches to control quality. The company focuses on fast delivery and pick-up, promising delivery in 30 minutes. Customers can order by phone, website, or mobile app from over 400 branches. The Pizza Company has 6 main business processes including

1. **Branch Operation**
   The Pizza Company has over 400 branches. Each branch manages its employees and delivery drivers and prepares food for customers. Branches ensure orders are completed and delivered on time.

2. **User Registration and Membership**
   This process creates a user account, required to place orders and join the membership. Membership includes points and tiers: Green (default), Silver (50+ points), and Gold (300+ points).

3. **Ordering Food**

   This process collects the user's information to complete an order for delivery. An order can include different items like pizza, pasta, or fried chicken. Users can adjust the quantity of each item. After that, the order is sent to a branch for preparation.

4. **Payments**
   Customers pay for orders using credit cards or other methods. The system ensures payments are secure and confirms the order once payment is complete.

5. **Discount**

   The Pizza Company gives many discounts and coupons. You can get discounts when ordering online, during holidays, or with certain credit cards. Coupons give discounts on some items or free sides. Members earn points from purchases, which can be used for discounts or free food. Coupons only work for specific items in the order.

6. **Deliver**

   The Pizza Company makes delivery and pick-up easy.

- Delivery Now: Brings your food in 30 minutes to your saved address. If no address is saved, you can add one while ordering.
- Delivery Later: Lets you choose a date and time for delivery (between 9:30 AM and 9:00 PM, up to 1 week ahead). Same-day orders must be at least 30 minutes later.

For pick-up, you choose a branch and collect your food at the set time.

2. Revised ERD

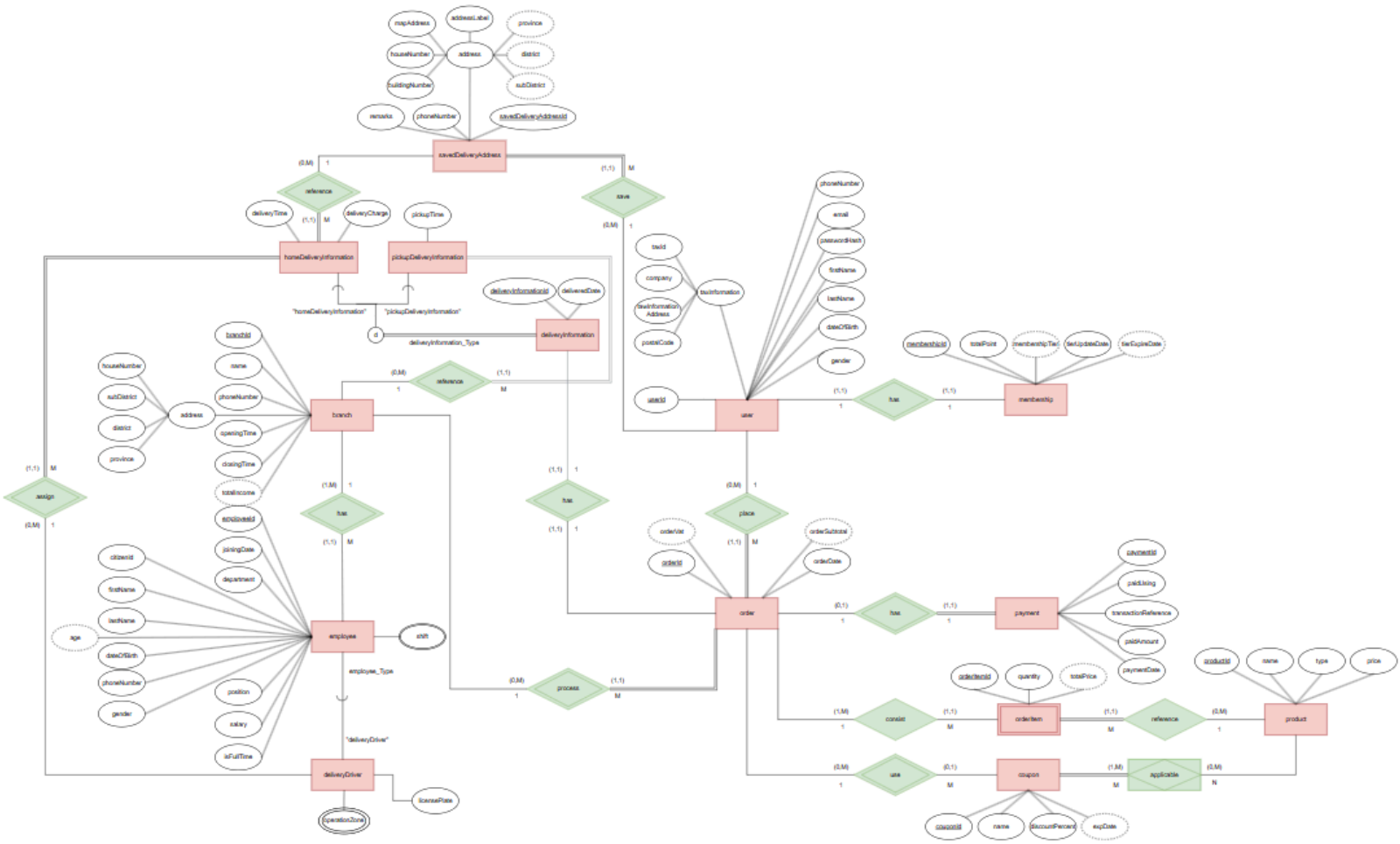


Figure 1: Revised ERD/EERD

[Full version of Revised ERD]

2. Step-by-step Relational Diagram Transformation

2.1 Step 1: Strong entity

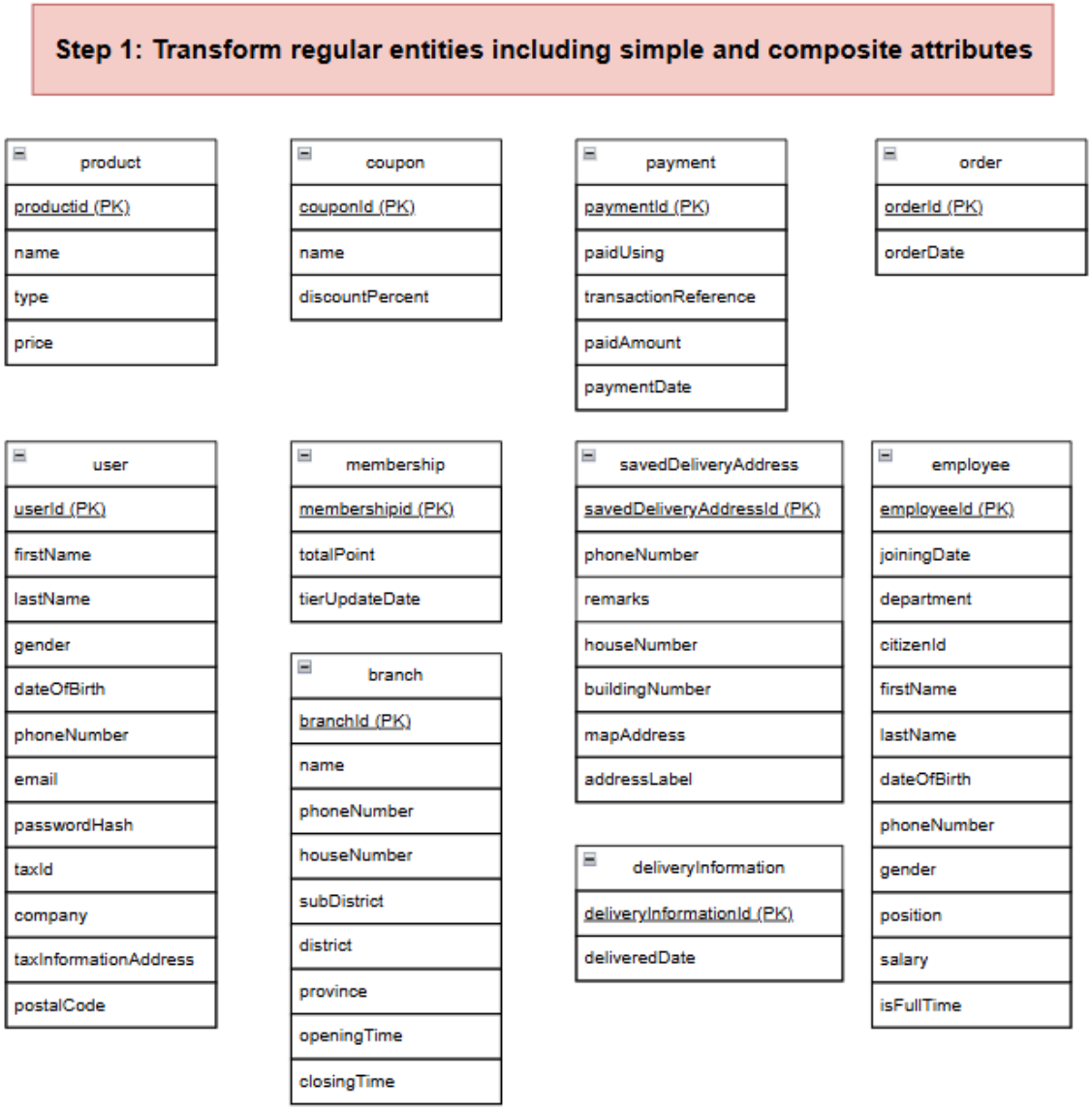


Figure 2: Transform regular entities including simple and composite attributes

2.2 Step 8(A) : Transform Specialization or Generalization

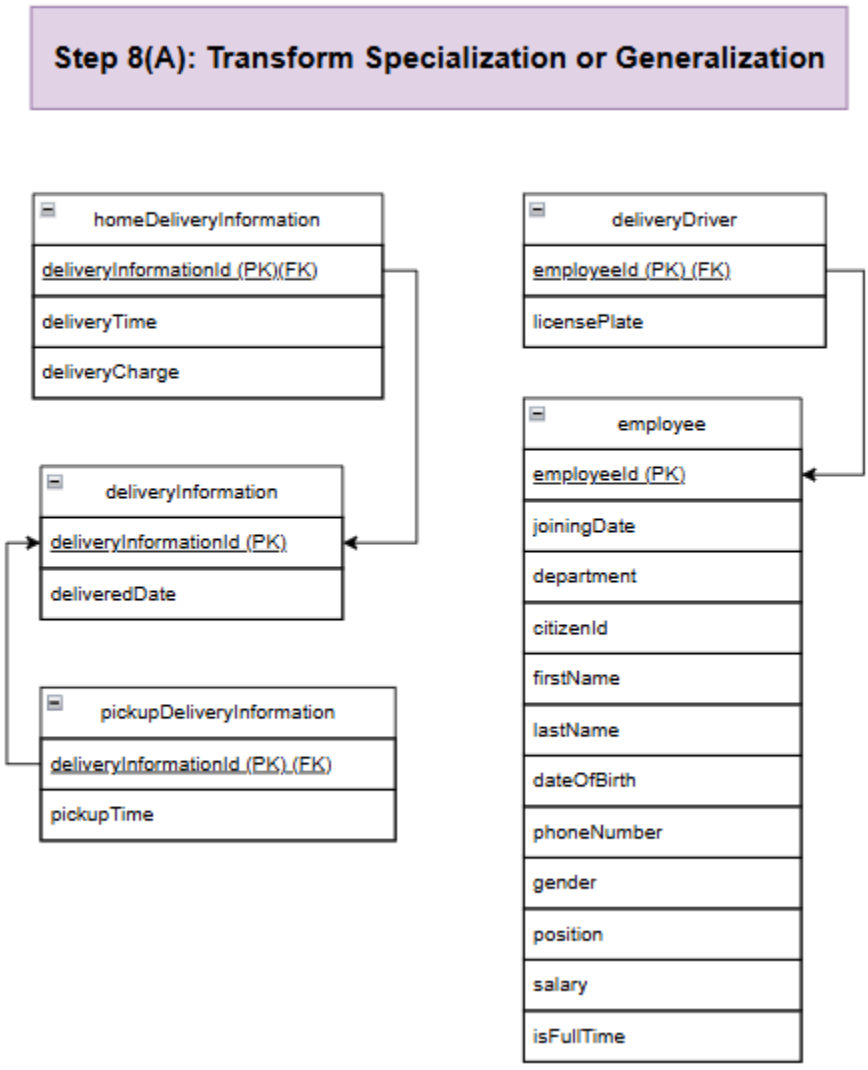


Figure 3: Transform Specialization or Generalization

2.3 Step 2: Weak entity

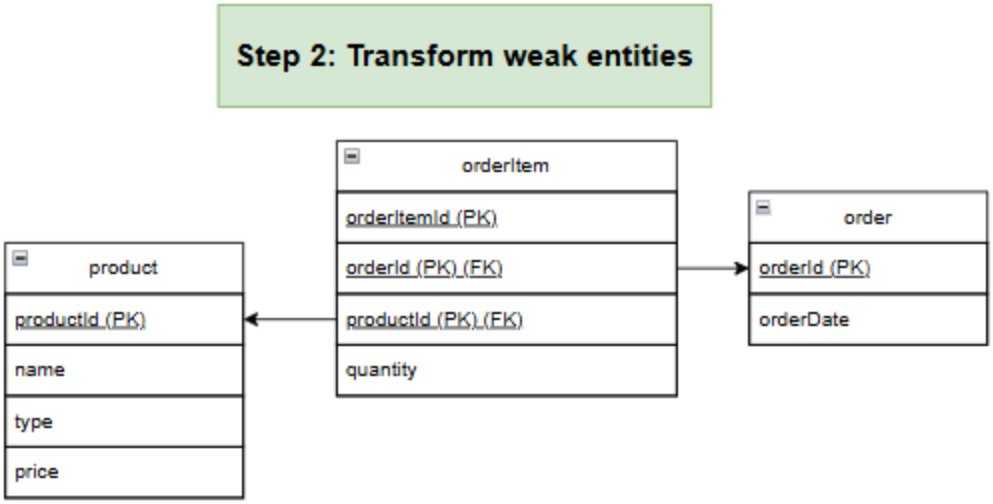


Figure 4: Transform weak entities

2.4 Step 3: Transform binary 1:1 relationship

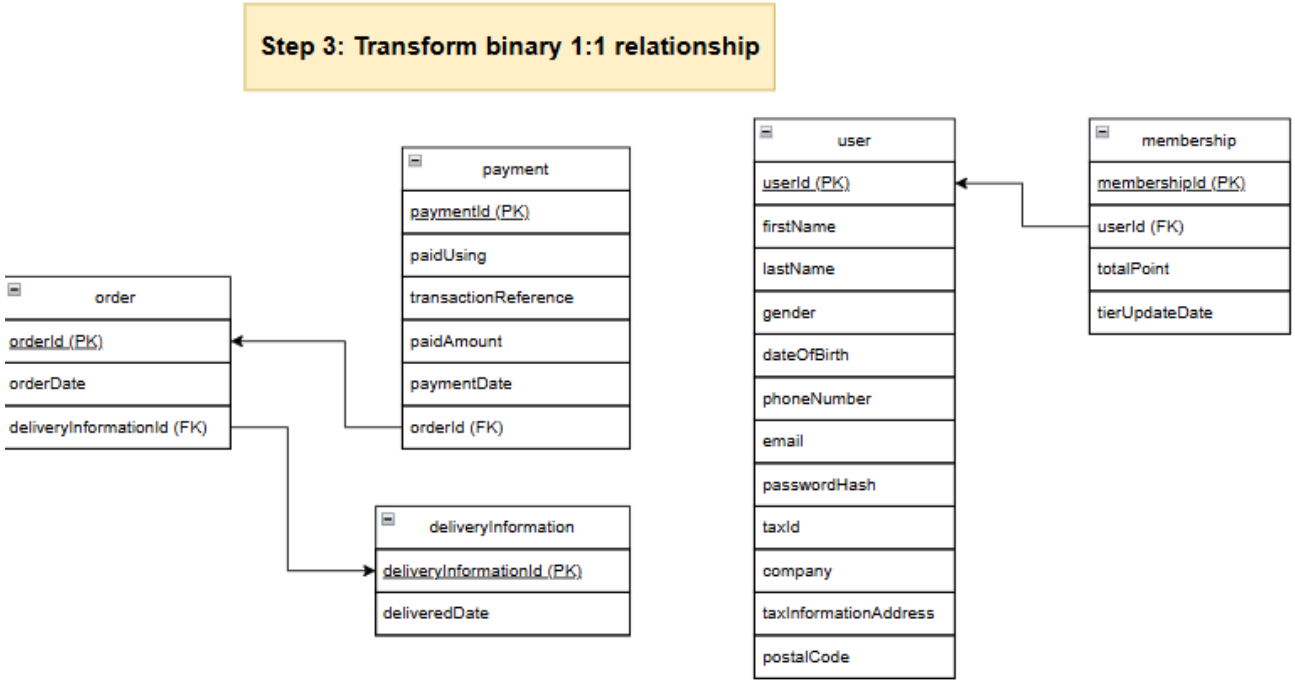


Figure 5: Transform binary 1:1 relationship

2.5 Step 4: Transform binary 1:M relationship

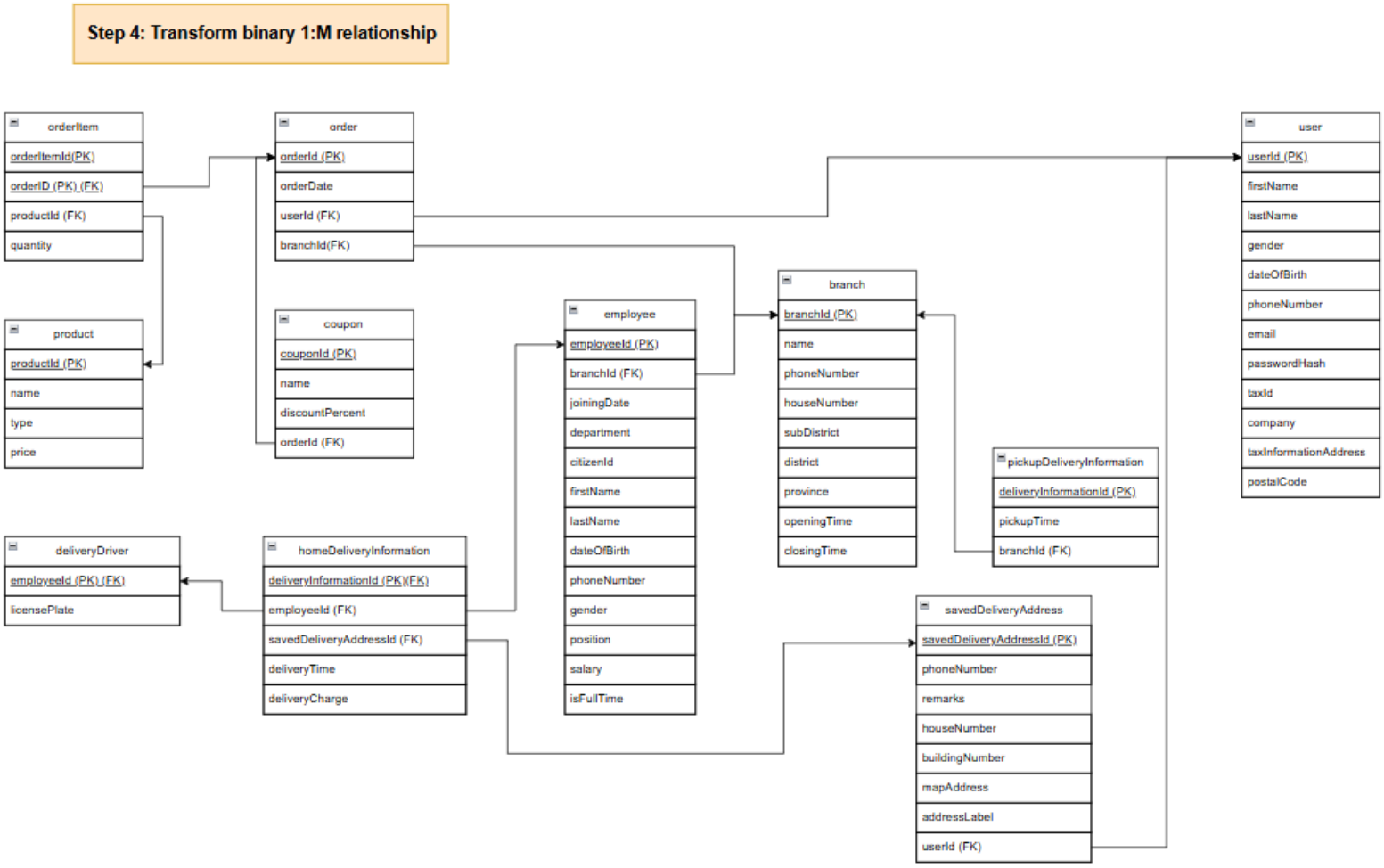


Figure 6: Transform binary 1:M relationship

2.6 Step 5: Transform binary M:N relationship

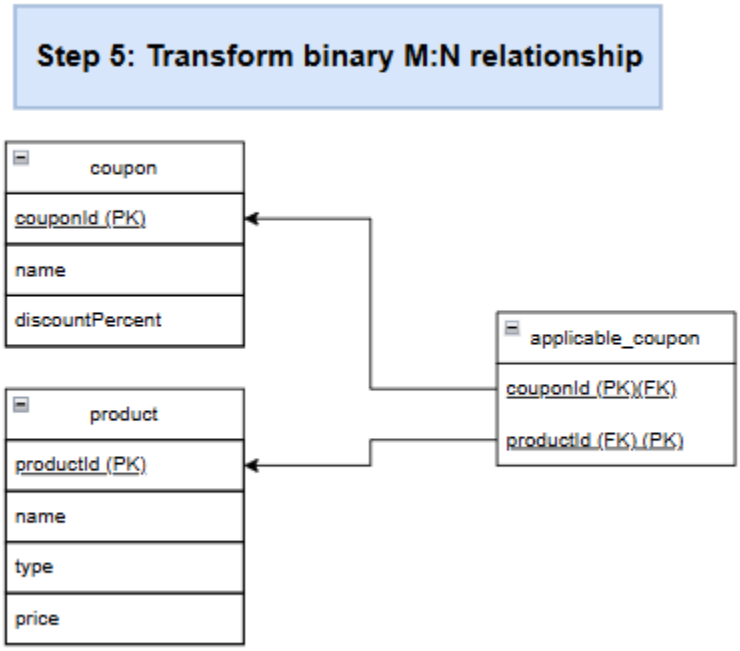


Figure 7: Transform binary M:N relationship

2.7 Step 6: Transform multivalued attributes

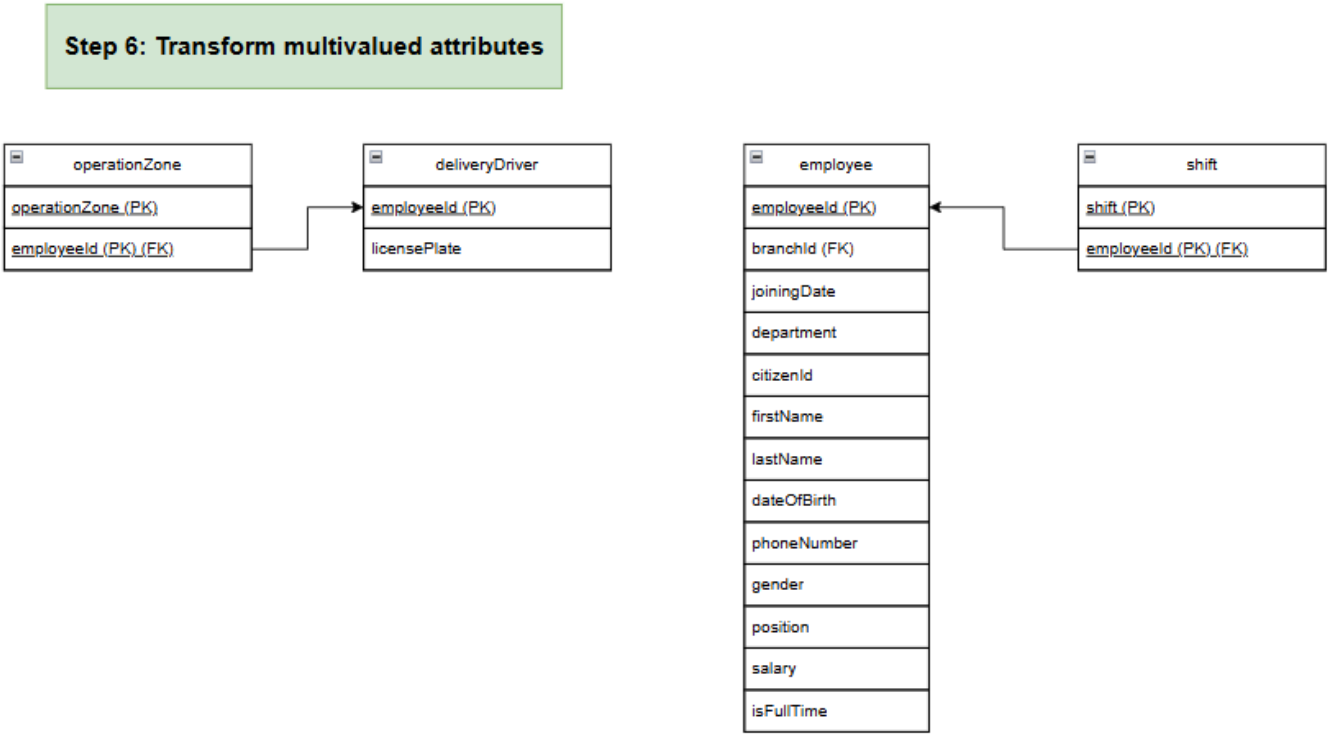


Figure 8: Transform multivalued attributes

2.8 Step 7: Transform N-ary relationships

NOT APPLICABLE

2.9 Final relational Schema

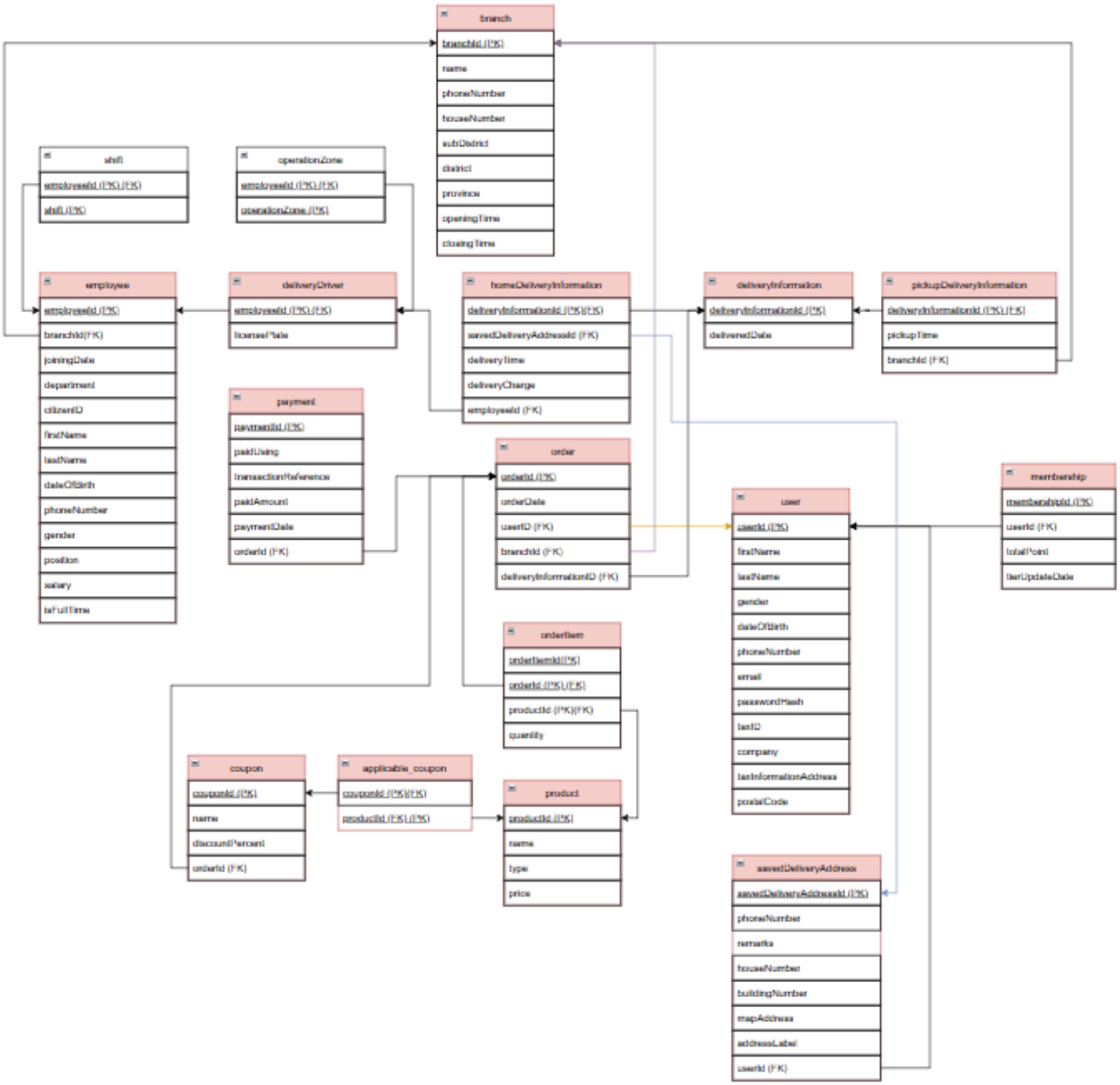


Figure 9: Final design

[Full version of Relational Schema]

3. Data Dictionary

Data Dictionary									
Table Name	Attribute Name	Contents	Type	Format	Nullable	Unique	Range	Key	FK Reference Table
user	userId	User's ID	int	x		Y		PK	
	firstName	First Name of the user	varchar(50)	Xxxxx					
	lastName	Last Name of the user	varchar(50)	Xxxxx					
	gender	Gender of the user	char(1)	X			("M", "F")		
	dateOfBirth	Date of Birth of the user	date	yyyy-mm-dd					
	phoneNumber	Phone Number of the user	varchar(20)	XXXXXX		Y			
	email	Email that user uses to register and open the account	varchar(100)	xxxxx@xxxx		Y			
	passwordHash	Hash for Password of the user account	varchar(100)	XXXXXXX					
	taxId	Tax's Id	varchar(10)	X	Y				
	company	Taxing Company	varchar(100)	XXXXXX	Y				
	taxInformationAddress	Address for Tax	varchar(200)	XXXXXX	Y				
	postalCode	Postal code for TaxInformation	varchar(10)	XXXXXX	Y				

Figure 10: Data dictionary for user table

Data Dictionary									
Table name	Attribute Name	Contents	Type	Fomat	Nullable	Unique	Range	Key	FK Reference Table
membership	membershipId	User's Membership ID	int	X		Y		PK	
	userId	the corresponding user who is the member	int	X		Y		FK	userId [user]
	totalPoint	total accumulated points for the user	int	X					
	tierUpdateDate	timestamp at which the tier will be updated again	datetime	yyyy-mm-dd HH:mm:ss					

Figure 11: Data dictionary for membership table

Data Dictionary									
Table Name	Attribute	Contents	Type	Format	Nullable	Unique	Range	Key	FK Reference Table
savedDeliveryAddress	savedDeliveryAddressId	saved delivery address Id which refer to each the delivery address that user saves in the account	int	X		Y		PK	
	phoneNumber	phone number of the delivery address	varchar(20)	xxxxxxxxxx					
	remarks	remarks about delivery	varchar(200)	xxxxxx	Y				
	houseNumber	house number of the saved delivery address	varchar(10)	xxxx					
	buildingNumber	building number of the saved delivery address	varchar(10)	xxxx	Y				
	mapAddress	google code address of the saved delivery address	varchar(50)	XXXX	Y				
	addressLabel	more detail of the address	text	X					
	userId	refer to the user who saved the address	int	X				FK	userId [user]

Figure 12: Data dictionary for savedDeliveryAddress table

Table Name	Attribute Name	Contents	Type	Format	Nullable	Unique	Range	Key	FK Reference Table
branch	branchId	Branch's Id	int	X		Y		PK	
	name	Branch's Name	varchar(100)	Xxxxx					
	phoneNumber	Branch Landline	varchar(20)	xxxxxxxxxx					
	houseNumber	Branch House Number	varchar(50)	XXXX					
	subDistrict	Branch Sub-District	varchar(50)	XXXX					
	district	Branch District	varchar(50)	XXXX					
	province	Branch Province	varchar(50)	XXXX					
	openingTime	Branch Opening Time	time	HH:mm:ss					
	closingTime	Branch Closing Time	time	HH:mm:ss					

Figure 13: Data dictionary for branch table

Data Dictionary									
Table Name	Attribute Name	Contents	Type	Format	Nullable	Unique	Range	Key	FK Reference Table
employee	employeeId	Employee's Id	int	x		Y		PK	
	branchId	Branch's Id of Employee	int	x				FK	branchId [branch]
	joiningDate	Joining Date of Employee	date	yyyy-mm-dd					
	department	Department of Employee	varchar(50)	XXXXX					
	citizenID	National ID of Employee	varchar(50)	XXXXX		Y			
	firstName	First Name of Employee	varchar(50)	Xxxxx					
	lastName	Last Name of Employee	varchar(50)	Xxxxx					
	dateOfBirth	Date of Birth of Employee	date	yyyy-mm-dd					
	phoneNumber	Phone Number of Employee	varchar(20)	XXXXX					
	gender	Gender of Employee	char(1)	X					
	position	Designation of Employee	varchar(100)	XXXXX					
	salary	Salary of Employee It is null for part time employee	decimal(12,2)	1234567890.00	Y				
	isFullTime	Full Time Status of Employee	boolean	X					

Figure 14: Data dictionary for employee table

Table Name	Attribute Name	Contents	Type	Format	Nullable	Unique	Range	Key	FK Reference Table
shift	employeeId	employee who takes the shift	int	X				PK, FK	employeeId [deliveryDriver]
	shift	shift of the employee (multivalue)	varchar(8)	xxxxxx			("morning", "evening", "night")	PK	

Figure 15: Data dictionary for shift table

Table Name	Attribute Name	Contents	Type	Format	Nullable	Unique	Range	Key	FK Reference Table
deliveryDriver	employeeId	employee's Id that is a delivery driver	int	X		Y		PK, FK	employeeId [deliveryDriver]
	licensePlate	license plate of the driver's car	nvarchar(20)	XX-XXXX XXXXX					

Figure 16: Data dictionary for deliveryDriver table

Table Name	Attribute Name	Contents	Type	Format	Nullable	Unique	Range	Key	FK Reference Table
operationZone	employeeId	An employee that is a delivery driver in the operation zone	int	X				PK, FK	employeeId [deliveryDriver]
	operationZone	operation zone of the delivery driver (multivalue)	varchar(50)	Xxxxx				PK	

Figure 17: Data dictionary for operationZone table

Table Name	Attribute Name	Contents	Type	Format	Nullable	Unique	Range	Key	FK Reference Table
deliveryInformation	deliveryInformationId	delivery information Id which refers to delivery information for each delivery	int	X		Y		PK	
	deliveredDate	date of delivery/pickup as selected by user	date	yyyy-mm-dd					

Figure 18: Data dictionary for deliveryInformation table

Table Name	Attribute Name	Contents	Type	Format	Nullable	Unique	Range	Key	FK Reference Table
product	productId	product Id	int	X				PK	
	name	product name	varchar(50)	Xxxxxx					
	type	type of product	varchar(50)	Xxxxxx					
	price	price of the product	decimal(12,2)	1234567890.00					

Figure 19: Data dictionary for product table

Table Name	Attribute Name	Contents	Type	Format	Nullable	Unique	Range	Key	FK Reference Table
order	orderId	Order Id	int	X		Y		PK	
	orderDate	Date of order	datetime	yyyy-mm-dd HH:mm:ss					
	userID	refer to the userid who places the order	int	X				FK	userId [user]
	branchID	refer to branch of the order	int	X				FK	branchId[branch]
	deliveryInformationId	Delivery Information ID of the order	int	X		Y		FK	deliveryInformationId [deliveryInformation]

Figure 20: Data dictionary for order table

Table Name	Attribute Name	Contents	Type	Format	Nullable	Unique	Range	Key	FK Reference Table
orderitem	orderItemId	order item Id	int	X		Y		PK	
	orderId	refer to the orderId which takes this orderitem	int	X				PK,FK	orderId [order]
	productId	refer to the productID	int	X				PK,FK	productId [product]
	quantity	the number of products	int	X					

Figure 21: Data dictionary for orderItem table

Table Name	Attribute	Contents	Type	Format	Nullable	Unique	Range	Key	FK Reference Table
coupon	couponId	coupon Id	int	X		Y		PK	
	name	coupon name	varchar(50)	XXXXX					
	discountPercent	discount percentage of the coupon	int	X			between (0,100)		
	orderId	refers to the order the coupon is applied on	int	X	Y			FK	orderId [order]

Figure 22: Data dictionary for coupon table

Table Name	Attribute	Contents	Type	Format	Nullable	Unique	Range	Key	FK Reference Table
applicable_coupon	couponId	coupon Id	int	X				PK, FK	couponId [coupon]
	productId	product Id	int	X				PK, FK	productId [product]

Figure 23: Data dictionary for applicable_coupon table
(This table will show which coupons can be used with specific products)

Table Name	Attribute Name	Contents	Type	Format	Nullable	Unique	Range	Key	FK Reference Table
payment	paymentId	Payment Id	int	X		Y		PK	
	paidUsing	Payment Method which 'cash' means that user paid with mobile banking application and 'credit card' means that user paid with credit card	varchar(50)	XXXXXX			("cash","credit card")		
	transactionReference	Transaction Reference	varchar(100)	XXXXXX					
	paidAmount	Total Amount Paid	decimal(12,2)	1234567890.00					
	paymentDate	Date and time of Payment	datetime	yyyy-mm-dd HH:mm:ss					
	orderId	ID of the Order Paid for	int	X		Y		FK	orderId [order]

Figure 24: Data dictionary for payment table

Table Name	Attribute Name	Contents	Type	Format	Nullable	Unique	Range	Key	FK Reference Table
pickupDeliveryInformation	deliveryInformationId	delivery information Id which for pickupDeliveryInformation the user will come and pickup an order at the branch	int	X		Y		PK, FK	deliveryInformationId [deliveryInformation]
	pickupTime	pickup time	time	HH:mm:ss					
	branchId	branch Id for the pickup order	int	X				FK	branchId[branch]

Figure 25: Data dictionary for pickupDeliveryInformation table

Table Name	Attribute Name	Contents	Type	Format	Nullable	Unique	Range	Key	FK Reference Table
homeDeliveryInfromation	deliveryInformationId	delivery information Id which for homeDeliveryInformation the driver will deliver the order to the address	int	X		Y		PK, FK	deliveryInformationId [deliveryInformation]
	savedDeliveryAddressId	saved delivery address Id	int	X				FK	savedDeliveryAddressId [savedDeliveryAddress]
	deliveryTime	delivery time	time	HH:mm:ss					
	deliveryCharge	delivery charge	decimal(12,2)	1234567890.00					
	employeeId	driver taken the delivery	int	X				Fk	employeeId [deliveryDriver]

Figure 26: Data dictionary for homeDeliveryInformation table

[Full version of Data Dictionary]

Table Description

1. **User Table**

Stores information about users, including personal details (name, gender, date of birth), contact information (email, phone number), and tax-related data.

2. **Membership Table**

Tracks user memberships, including the membership ID, associated user, total accumulated points, and the tier update date.

3. **savedDeliveryAddress Table**

Maintains user-saved delivery addresses with details like address components (house/building number, map address), phone number, and associated remarks.

4. **Branch Table**

Stores information of all the branches including its phone number, address and opening and closing times.

5. **Employee Table**

Holds the information of all the employees across all branches, with their personal information, joining dates, designation, salary and full-time status as well as the branch they belong to.

6. **Shift Table**

Holds the information regarding the shifts of a given employee i.e., morning, evening or night

7. **deliveryDriver Table**

Derives from the employee. Holds information about employees who are delivery drivers along with their license plate numbers (of their vehicle).

8. **operationZone Table**

Holds information about delivery zones as well as the delivery drivers assigned to each of these.

9. **deliveryInformation Table**

Holds delivery information for orders which were selected to be delivered. Including their date of delivery (completed).

10. **Product Table**

Holds information about all of the products offered by the company including their type, name and price.

11. **Order Table**

Holds information about all of the orders, along with the date when the order was placed, user who placed the order, branch where it was ordered from, and the delivery information of the order.

12. **Orderitem Table**

Holds the products that have been ordered, along with which order they belong to, which product it refers to and the quantity of the product ordered.

13. **Coupon Table**

Holds the information about the used coupons, including their names, percentage of discount offered, and the order which used this coupon.

14. **Applicable_coupon Table**

shows which coupons are eligible to be used with a given product.

15. **Payment Table**

Holds the payment information, including method of payment, transaction reference, total amount paid and date of payment and the order for which the payment was done.

16. **pickupDeliveryInformation Table**

Holds the information for orders that were placed with the option of using pickup as the method of delivery, along with the pickup time and the branch where the customer will pick it up from.

17. **homeDeliveryInformation Table**

Holds the information related to orders for which home delivery option was selected, along with a reference to a delivery

4. SQL Command

3.1 SQL Command for creating table

```sql
-- Create the database
DROP DATABASE IF EXISTS PizzaCompany;
CREATE DATABASE PizzaCompany;
USE PizzaCompany;


-- Table: user
CREATE TABLE `user` (
    userId                  INT PRIMARY KEY AUTO_INCREMENT,
    firstName               VARCHAR(50) NOT NULL,
    lastName                VARCHAR(50) NOT NULL,
    gender                  CHAR(1) NOT NULL,
    dateOfBirth             DATE NOT NULL,
    phoneNumber             VARCHAR(20) UNIQUE NOT NULL,
    email                   VARCHAR(100) UNIQUE NOT NULL,
    passwordHash            VARCHAR(100) NOT NULL,
    taxId                   VARCHAR(10),
    company                 VARCHAR(100),
    taxInformationAddress   VARCHAR(200),
    postalCode              VARCHAR(10),

    CHECK (gender IN ('M', 'F'))
);


-- Table: membership
CREATE TABLE membership (
      membershipId    INT PRIMARY KEY AUTO_INCREMENT,
      userId          INT UNIQUE NOT NULL,
      totalPoint      INT NOT NULL,
      tierUpdateDate  DATETIME NOT NULL,

      FOREIGN KEY (userId) REFERENCES `user`(userId)
);
```

```sql
-- Table: savedDeliveryAddress (references user table)
CREATE TABLE savedDeliveryAddress (
    savedDeliveryAddressId  INT PRIMARY KEY AUTO_INCREMENT,
    phoneNumber             VARCHAR(20) NOT NULL,
    remarks                 VARCHAR(200),
    houseNumber             VARCHAR(10) NOT NULL,
    buildingNumber          VARCHAR(10),
    mapAddress              VARCHAR(50),
    addressLabel            TEXT NOT NULL,
    userId                  INT NOT NULL,

    FOREIGN KEY (userId) REFERENCES user(userId)
);


-- Table: branch
CREATE TABLE branch (
    branchId    INT PRIMARY KEY AUTO_INCREMENT,
    name        VARCHAR(100) NOT NULL,
    phoneNumber VARCHAR(20) NOT NULL,
    houseNumber VARCHAR(50) NOT NULL,
    subDistrict VARCHAR(50) NOT NULL,
    district    VARCHAR(50) NOT NULL,
    province    VARCHAR(50) NOT NULL,
    openingTime TIME NOT NULL,
    closingTime TIME NOT NULL
);
```

```sql
-- Table: employee (references branch table)
CREATE TABLE employee (
    employeeId      INT PRIMARY KEY AUTO_INCREMENT,
    branchId        INT NOT NULL,
    joiningDate     DATE NOT NULL,
    department      VARCHAR(50) NOT NULL,
    citizenID       VARCHAR(13) UNIQUE NOT NULL,
    firstName       VARCHAR(50) NOT NULL,
    lastName        VARCHAR(50) NOT NULL,
    dateOfBirth     DATE NOT NULL,
    phoneNumber     VARCHAR(20) NOT NULL,
    gender          CHAR(1) NOT NULL,
    position        VARCHAR(100) NOT NULL,
    salary          DECIMAL(12, 2),
    isFullTime      TINYINT(1) NOT NULL,

    CHECK (gender IN ('M', 'F')),
    FOREIGN KEY (branchId) REFERENCES branch(branchId)
);


-- Table: shift (references employee table)
CREATE TABLE shift (
    employeeId  INT NOT NULL,
    shift       VARCHAR(8) NOT NULL,

    PRIMARY KEY (employeeId, shift),
    FOREIGN KEY (employeeId) REFERENCES employee(employeeId),
    CHECK (shift IN ('morning', 'evening', 'night'))
);


-- Table: deliveryDriver (references employee table)
CREATE TABLE deliveryDriver (
    employeeId      INT PRIMARY KEY,
    licensePlate    NVARCHAR(20) NOT NULL,

    FOREIGN KEY (employeeId) REFERENCES employee(employeeId)
);
```

```sql
-- Table: operationZone (references employee table)
CREATE TABLE operationZone (
    employeeId      INT NOT NULL,
    operationZone   VARCHAR(50) NOT NULL,

    PRIMARY KEY (employeeId, operationZone),
    FOREIGN KEY (employeeId) REFERENCES deliveryDriver(employeeId)
);


-- Table: deliveryInformation
CREATE TABLE deliveryInformation (
    deliveryInformationId   INT PRIMARY KEY AUTO_INCREMENT,
    deliveredDate           DATE NOT NULL
);


-- Table: product
CREATE TABLE product (
    productId   INT PRIMARY KEY AUTO_INCREMENT,
    name        VARCHAR(50) NOT NULL,
    type        VARCHAR(50) NOT NULL,
    price       DECIMAL(12, 2) NOT NULL
);


-- Table: order (references user, branch, and deliveryInformation tables)
CREATE TABLE `order` (
    orderId                 INT PRIMARY KEY AUTO_INCREMENT,
    orderDate               DATETIME NOT NULL,
    userID                  INT NOT NULL,
    branchID                INT NOT NULL,
    deliveryInformationId   INT UNIQUE NOT NULL,

    FOREIGN KEY (userID) REFERENCES user(userId),
    FOREIGN KEY (branchID) REFERENCES branch(branchId),
    FOREIGN KEY (deliveryInformationId) REFERENCES
deliveryInformation(deliveryInformationId)
);
```

```sql
-- Table: orderItem (references order and product tables)
CREATE TABLE orderItem (
    orderItemId INT NOT NULL,
    orderId     INT NOT NULL,
    productId   INT NOT NULL,
    quantity    INT NOT NULL,

    PRIMARY KEY (orderItemId, orderId, productId),
    FOREIGN KEY (orderId) REFERENCES `order`(orderId),
    FOREIGN KEY (productId) REFERENCES product(productId)
);


-- Table: coupon
CREATE TABLE coupon (
    couponId            INT PRIMARY KEY AUTO_INCREMENT,
    name                VARCHAR(50) NOT NULL,
    discountPercent     INT NOT NULL,
    orderId             INT,

    CHECK (discountPercent BETWEEN 0 AND 100),
    FOREIGN KEY (orderId) REFERENCES `order`(orderId)
);


-- Table: applicable_coupon (references order, coupon, and product tables)
CREATE TABLE applicable_coupon (
    couponId    INT NOT NULL,
    productId   INT NOT NULL,

    PRIMARY KEY (couponId, productId),
    FOREIGN KEY (couponId) REFERENCES coupon(couponId),
    FOREIGN KEY (productId) REFERENCES product(productId)
);
```

```sql
-- Table: payment (references order and product tables)
CREATE TABLE payment (
    paymentId               INT PRIMARY KEY AUTO_INCREMENT,
    paidUsing               VARCHAR(50) NOT NULL,
    transactionReference    VARCHAR(100) NOT NULL,
    paidAmount              DECIMAL(12, 2) NOT NULL,
    paymentDate             DATETIME NOT NULL,
    orderId                 INT UNIQUE NOT NULL,

    CHECK (paidUsing IN ('cash', 'credit card')),
    FOREIGN KEY (orderId) REFERENCES `order`(orderId)
);


-- Table: pickupDeliveryInformation (references deliveryInformation and
branch tables)
CREATE TABLE pickupDeliveryInformation (
    deliveryInformationId   INT PRIMARY KEY,
    pickupTime              TIME NOT NULL,
    branchId                INT NOT NULL,

    FOREIGN KEY (deliveryInformationId) REFERENCES
deliveryInformation(deliveryInformationId),
    FOREIGN KEY (branchId) REFERENCES branch(branchId)
);


-- Table: homeDeliveryInformation (references deliveryInformation,
savedDeliveryAddress, and employee tables)
CREATE TABLE homeDeliveryInformation (
    deliveryInformationId   INT PRIMARY KEY,
    savedDeliveryAddressId  INT NOT NULL,
    deliveryTime            TIME NOT NULL,
    deliveryCharge          DECIMAL(12, 2) NOT NULL,
    employeeId              INT NOT NULL,

    FOREIGN KEY (deliveryInformationId) REFERENCES
deliveryInformation(deliveryInformationId),
    FOREIGN KEY (savedDeliveryAddressId) REFERENCES
savedDeliveryAddress(savedDeliveryAddressId),
    FOREIGN KEY (employeeId) REFERENCES deliveryDriver(employeeId)
);
```

3.2 SQL Command for inserting data into the database

```
INSERT INTO user
(firstName,lastName,gender,dateOfBirth,phoneNumber,email,passwordHash,taxId
,company,taxinformationAddress,postalCode)
VALUES
('Natchapol','Lebkrut','M','2003-06-30','0611111111','natchapol.lebkrut@db.
com','$2b$12$CjP2UWnaD3s9QvHg/Mh.BulDQAi75.pgeu8jXZmoCQQaVRTieOjta','111111
1111','SIIT','Khlong Luang District, Pathum Thani.','12120'),
('Vayuphak','Saengthong','M','2003-09-15','0622222222','vayuphak.saengthong
@db.com','$2b$12$4QXcxxg/Iis0HD9nxMjsyOfuT9KbErEl5ulDlnfgZh9neGu9IwItq','22
22222222','Thai Japan Company Ltd.','Bangkok, Thailand','10110'),
('Napat', 'Decha', 'M', '2002-08-22','0633333333', 'napat.decha@db.com',
'$2b$12$9MSuJdQ9zRbdrk5Dp9hKTO9Dylw69T1Cx/jErQ6.Siw.y4z3HvMte',
'3333333333', 'False Corp.','Bangkok, Thailand','10500'),
('Napat', 'Tatiyakaroonwong', 'M', '2002-11-12','0644444444',
'napat.tatiya@db.com',
'$2b$12$.Ln77qph1xM9IQEesFHDVeZl0cvVEdvIfBzmf7HVeR9bGsAlZkpxy',
'4444444444', 'SIA Corp.','Bangkok, Thailand','10600'),
('Chatdanai', 'Wongsuwan', 'M', '2001-03-29','0655555555',
'chatdanai.wongsuwan@db.com',
'$2b$12$e1pL7KtEAvIzqlyjzL7HUuuMLbcbRdPfQNkGDCq3bIRwdPio3YFz2',
'5555555555', 'Thai German Company Ltd.','Khlong Luang District, Pathum
Thani.','10400'),
('Haseeb','Ali','M','2003-07-04','0666666666','haseeb.ai@db.com','$2b$12$1w
LlV5dcGvcWRcAqlRnVee.LKNW3.S23GkHcxybr10tcOGYvED0bu',NULL,NULL,NULL,NULL),
('Somsri', 'Hiran', 'F', '1995-05-10','0677777777', 'somsak.hiran@db.com',
'$2b$12$Hpu4WjlxyT2lvTb/ceEjceH9Uih3jxn5Lkge0HGVEDdVbPCDpqH4e',NULL,NULL,NU
LL,NULL),
('Yingrat', 'Rattana', 'F', '1998-02-18','0688888888',
'yingrat.rattana@db.com',
'$2b$12$9jsqZ23PIon45zctdOa5m.JYOa4DnEuV5QvHnFrFgKoIuEqGBzn0a',NULL,NULL,NU
LL,NULL),
('Itthirath', 'Jeamanukul', 'M', '2000-12-07','0699999999',
'itthirath.jeamanukul@db.com',
'$2b$12$7G2OtsQxDCGYYfrqRdo85.fLLri1LsMW7qbfLl2SGjMb98ZPq4IK2',NULL,NULL,NU
LL,NULL),
('Pikul', 'Watdao', 'F', '1996-08-25','0612222222', 'pikul.watdao@db.com',
'$2b$12$CCuI1IxDfT/E6.2Vf2fxzeXS.jh9zEMO/6UM/jR9w9EUZTytfZoFq',NULL,NULL,NU
LL,NULL);
```

```sql
INSERT INTO membership (userId, totalPoint, tierUpdateDate) VALUES
(1, 250, '2025-03-15'),
(2, 45, '2025-02-10'),
(3, 400, '2025-03-05'),
(4, 150, '2025-04-01'),
(5, 160, '2025-05-20'),
(6, 40, '2025-06-12'),
(7, 320, '2025-07-30'),
(8, 170, '2025-08-18'),
(9, 0, '2024-12-25'),
(10, 100, '2024-12-03');
```

```
INSERT INTO savedDeliveryAddress (phoneNumber, remarks, houseNumber,
buildingNumber, mapAddress, addressLabel, userId) VALUES
('0611111111', NULL, '827', NULL, '3J94+HX4', 'Wangmai, Pathumwan, Bangkok
10330', 1),
('0611111111', 'close to Lotus', '940/1', 'B', '3J94+FJ3', 'Sam Sen Nok,
Huai Khwang, Bangkok 10310', 1),
('0622222222', NULL, '141', NULL, '3J94+HQ2', 'Wangburapa, PraNaKorn,
Bangkok 10200', 2),
('0633333333', NULL, '149', 'C', '3J94+H9V', 'Wangburapa, PraNaKorn,
Bangkok 10200', 3),
('0644444444', 'next to BigC', '670', NULL, '3J94+M83', 'Ladyao, Jatujak,
Bangkok 10900', 4),
('0655555555', NULL, '104', NULL, '3J94+Q2M', 'Bangkrasor, Muang,
Nonthaburi 11000', 5),
('0666666666', 'Biggest House', '90', NULL, '3J94+5X4', 'Klongluang,
Pathumthani 12120', 6),
('0677777777', NULL, '96', 'D', '3J94+6GG', 'Klong 1, Klongluang,
Pathumthani 12120', 7),
('0688888888', 'close to toll way', '139', NULL, '3J94+HQW', 'Wangburapa,
PraNaKorn, Bangkok 10200', 8),
('0699999999', NULL, '291/1', 'E', '3J94+HX4', 'Wangmai, Pathumwan, Bangkok
10330', 9),
('0612222222', 'next to the school', '393/9', 'F', '3J94+F3J', 'Sam Sen
Nok, Huai Khwang, Bangkok 10310', 10),
('0611111111', NULL, '510', NULL, '3J94+FF9', 'Sam Sen Nok, Huai Khwang,
Bangkok', 1),
('0611111111', 'next to the train station', '345', 'G', '3J94+H2V',
'Wangmai, Pathumwan, Bangkok 10330', 1),
('0633333333', NULL, '123', 'H', '3J94+QM7', 'Bangkrasor, Muang, Nonthaburi
11000', 3),
('0644444444', NULL, '829', NULL, '3J94+M83', 'Ladyao, Jatujak, Bangkok
10900', 4);
```

```
INSERT INTO branch (name, phoneNumber, houseNumber, subDistrict, district,
province, openingTime, closingTime) VALUES
('Pizza Com Lotus Chareonpol Tesco Lotus Rama 1', '026126882', '831',
'Wangmai', 'Pathumwan', 'Bangkok', '09:00:00', '21:00:00'),
('Pizza Com Samyan Mitrtown', '0654541638', '944/1', 'Wangmai',
'Pathumwan', 'Bangkok', '10:00:00', '22:00:00'),
('Pizza Com Meng-Jai', '0614120318', '507-509', 'Sam  Sen Nok', 'Huai
Khwang', 'Bangkok', '10:00:00', '24:00:00'),
('Pizza Com Ratchabophit', '0655076352', '148', 'Wangburapa', 'PraNaKorn',
'Bangkok', '08:00:00', '24:00:00'),
('Pizza Com Big C Ladprao', '029837398', '669', 'Ladyao', 'Jatujak',
'Bangkok', '10:00:00', '21:00:00'),
('Pizza Com Manor Sanambin-Nam', '0922810673', '102-103', 'Bangkrasor',
'Muang', 'Nonthaburi', '08:00:00', '22:00:00'),
('Pizza Com PTT Sribuathong', '0922810619', '46/21', 'Sono Loy', 'Bang Bua
Thong', 'Nonthaburi', '08:00:00', '22:00:00'),
('Pizza Com Thummasart Rangsit', '0922810584', '95', 'Klong 1',
'Klongluang', 'Pathumthani', '09:00:00', '21:50:00'),
('Pizza Com Klongluang', '0922810725', '7/39-40', 'Klongsong',
'Klongluang', 'Pathumthani', '10:00:00', '24:00:00'),
('Pizza Com Rangsit Klong 10', '0922810600', '40/2', 'Buengsanan',
'Thanyaburi', 'Pathumthani', '09:00:00', '22:00:00');
```

```
INSERT INTO employee (branchId, joiningDate, department, citizenID,
firstName, lastName, dateOfBirth, phoneNumber, gender, position, salary,
isFullTime) VALUES
(1, '2017-03-10', 'Management', '4567890723457', 'Anan', 'Phongchai',
'1995-02-28', '0612345673', 'M', 'Branch Manager', 35000.00, 1),
(1, '2020-05-10', 'Cooking', '1234567890173', 'Thanakorn', 'Srisuk',
'1990-04-15', '0812345670', 'M', 'Pizza Baker', 25000.00, 1),
(1, '2023-01-20', 'Cashier', '3406789012346', 'Sirin', 'Kittithorn',
'1985-12-05', '0812345672', 'F', 'Cashier', 15000.00, 1),
(1, '2023-02-25', 'Delivery', '5648901234568', 'Nipa', 'Wanich',
'1988-11-14', '0812345674', 'F', 'Delivery Driver', 20000.00, 1),
(1, '2021-07-15', 'Delivery', '2745678901235', 'Nattapong', 'Boonmee',
'1992-07-20', '0612345671', 'M', 'Delivery Driver', NULL, 0),
(2, '2017-11-05', 'Management', '9712345678903', 'Kanok', 'Phongpan',
'1991-08-25', '0812345678', 'M', 'Branch Manager', 35000.00, 1),
(2, '2019-02-10', 'Cooking', '6789012345679', 'Somsak', 'Thammasak',
'1993-09-22', '0612345675', 'M', 'Pizza Baker', 25000.00, 1),
(2, '2023-06-10', 'Cashier', '8971234567892', 'Preecha', 'Wiriyaporn',
'1982-01-10', '0612345677', 'M', 'Cashier', 15000.00, 1),
(2, '2023-03-01', 'Delivery', '7123456789014', 'Saowaluk', 'Yamsri',
'1989-03-12', '0612345679', 'F', 'Delivery Driver', 20000.00, 1),
(2, '2021-12-12', 'Delivery', '7890123456731', 'Thida', 'Sukjai',
'1990-06-30', '0812345676', 'F', 'Delivery Driver', NULL, 0),
(3, '2017-04-10', 'Management', '4356789712348', 'Manas', 'Phutthachot',
'1991-06-20', '0612345683', 'M', 'Branch Manager', 35000.00, 1),
(3, '2019-06-15', 'Cooking', '1723456789015', 'Wiroj', 'Suchat',
'1987-05-16', '0812345680', 'M', 'Pizza Baker', 25000.00, 1),
(3, '2023-04-25', 'Cashier', '3245678971237', 'Weerachai', 'Phrompakdee',
'1988-03-30', '0812345682', 'M', 'Cashier', 15000.00, 1),
(3, '2023-03-10', 'Delivery', '5467897123459', 'Kamol', 'Sriboon',
'1994-09-22', '0812345684', 'M', 'Delivery Driver', 20000.00, 1),
(3, '2020-07-30', 'Delivery', '2134567897126', 'Patcharaporn', 'Nontaket',
'1995-10-10', '0612345681', 'F', 'Delivery Driver', NULL, 0),
(4, '2016-01-25', 'Management', '9901234567893', 'Pornchai', 'Thammasak',
'1995-02-14', '0812345688', 'M', 'Branch Manager', 35000.00, 1),
(4, '2018-10-12', 'Cooking', '6578901234560', 'Supaporn', 'Yokchai',
'1986-08-30', '0612345685', 'F', 'Pizza Baker', 25000.00, 1),
(4, '2023-03-18', 'Cashier', '8790123456782', 'Narong', 'Tosapon',
'1989-12-17', '0612345687', 'M', 'Cashier', 15000.00, 1),
(4, '2023-07-11', 'Delivery', '1112345678904', 'Ratchanee', 'Sungthong',
'1990-11-11', '0612345689', 'F', 'Delivery Driver', 20000.00, 1),
(4, '2020-03-18', 'Delivery', '7689012345671', 'Suchada', 'Sutham',
```

```
'1983-07-24', '0812345686', 'F', 'Delivery Driver', NULL, 0),
(5, '2017-02-05', 'Management', '2223456789015', 'Rungnapa', 'Chaisarn',
'1985-01-02', '0812345690', 'F', 'Branch Manager', 35000.00, 1),
(5, '2019-09-01', 'Cooking', '3334567890126', 'Yuthana', 'Namwong',
'1993-11-11', '0612345691', 'M', 'Pizza Baker', 25000.00, 1),
(5, '2023-01-15', 'Cashier', '4445678901237', 'Somchai', 'Kittipong',
'1982-05-30', '0812345692', 'M', 'Cashier', 15000.00, 1),
(5, '2023-02-10', 'Delivery', '6667890123459', 'Rattana', 'Anantachai',
'1986-02-10', '0812345694', 'F', 'Delivery Driver', 20000.00, 1),
(5, '2021-04-20', 'Delivery', '5556789012348', 'Udom', 'Jaiklang',
'1990-08-18', '0612345693', 'M', 'Delivery Driver', NULL, 0),
(6, '2017-07-30', 'Management', '1123456789013', 'Chompoo', 'Rattanasuk',
'1990-10-05', '0812345703', 'F', 'Branch Manager', 35000.00, 1),
(6, '2019-01-05', 'Cooking', '7778901234567', 'Wanchai', 'Prasert',
'1984-12-20', '0812345700', 'M', 'Pizza Baker', 25000.00, 1),
(6, '2023-03-12', 'Cashier', '8889012345678', 'Anong', 'Pimkarn',
'1991-03-25', '0812345701', 'F', 'Cashier', 15000.00, 1),
(6, '2023-03-22', 'Delivery', '2234567890124', 'Phirun', 'Chakrit',
'1987-05-12', '0812345704', 'M', 'Delivery Driver', 20000.00, 1),
(6, '2023-06-08', 'Delivery', '9990123456789', 'Pakorn', 'Sukwattana',
'1988-09-15', '0812345702', 'M', 'Delivery Driver', NULL, 0),
(7, '2017-09-10', 'Management', '3345678901235', 'Yingyai', 'Sudjai',
'1989-08-24', '0812345705', 'F', 'Branch Manager', 35000.00, 1),
(7, '2020-05-25', 'Cooking', '5567890123457', 'Kasem', 'Sukthang',
'1991-04-08', '0612345706', 'M', 'Pizza Baker', 25000.00, 1),
(7, '2023-01-03', 'Cashier', '6678901234561', 'Ritthikorn', 'Nakarach',
'1992-09-17', '0812345707', 'M', 'Cashier', 15000.00, 1),
(7, '2023-08-20', 'Delivery', '3345678901237', 'Chaisiri', 'Chirawat',
'1995-02-02', '0612345709', 'M', 'Delivery Driver', 20000.00, 1),
(7, '2021-06-30', 'Delivery', '2134567890123', 'Siriporn', 'Dhanabut',
'1994-04-10', '0812345708', 'F', 'Delivery Driver', NULL, 0),
(8, '2017-08-24', 'Management', '8890123456780', 'Burin', 'Somjit',
'1985-12-08', '0812345710', 'M', 'Branch Manager', 35000.00, 1),
(8, '2021-01-15', 'Cooking', '1112345678902', 'Kanlayanee', 'Pathawee',
'1991-08-11', '0812345712', 'F', 'Pizza Baker', 25000.00, 1),
(8, '2023-11-05', 'Cashier', '9901234567897', 'Korn', 'Kritsana',
'1994-02-14', '0812345711', 'M', 'Cashier', 15000.00, 1),
(8, '2023-07-27', 'Delivery', '3334567890124', 'Yuwadee', 'Niwat',
'1992-11-25', '0812345714', 'F', 'Delivery Driver', 20000.00, 1),
(8, '2023-03-19', 'Delivery', '2223456789013', 'Pharanee', 'Jirasri',
'1987-09-05', '0812345713', 'F', 'Delivery Driver', NULL, 0),
(9, '2017-10-15', 'Management', '4445678901235', 'Wut', 'Saengchan',
```

```
'1988-01-30', '0812345715', 'M', 'Branch Manager', 35000.00, 1),
(9, '2020-05-16', 'Cooking', '6667890123457', 'Thitiporn', 'Sutthipong',
'1989-04-13', '0812345717', 'F', 'Pizza Baker', 25000.00, 1),
(9, '2023-02-27', 'Cashier', '5556789012346', 'Kusuma', 'Wongthong',
'1995-07-08', '0812345716', 'F', 'Cashier', 15000.00, 1),
(9, '2023-09-30', 'Delivery', '7778901234568', 'Chanchai', 'Tanasuk',
'1984-06-21', '0812345718', 'M', 'Delivery Driver', 20000.00, 1),
(9, '2023-11-12', 'Delivery', '8889012345679', 'Noknoi', 'Pattana',
'1990-08-19', '0812345719', 'F', 'Delivery Driver', NULL, 0),
(10, '2017-12-18', 'Management', '9990123456780', 'Prapai', 'Chayachote',
'1987-03-11', '0812345720', 'F', 'Branch Manager', 35000.00, 1),
(10, '2020-08-14', 'Cooking', '2222345678902', 'Chinnakorn', 'Namchai',
'1993-10-02', '0812345722', 'M', 'Pizza Baker', 25000.00, 1),
(10, '2023-04-01', 'Cashier', '1111234567891', 'Worachai', 'Phongpan',
'1985-12-25', '0812345721', 'M', 'Cashier', 15000.00, 1),
(10, '2023-03-23', 'Delivery', '3333456789013', 'Siriporn', 'Jindaporn',
'1992-04-18', '0812345723', 'F', 'Delivery Driver', 20000.00, 1),
(10, '2023-06-02', 'Delivery', '4444567890124', 'Phairoj', 'Thongdee',
'1989-05-07', '0812345724', 'M', 'Delivery Driver', NULL, 0),
(1, '2023-11-15', 'Cashier', '5001234567890', 'Kittisak', 'Sittiporn',
'1997-05-23', '0912345700', 'M', 'Cashier', 15000.00, 1),
(1, '2023-11-18', 'Cooking', '5002345678901', 'Narumon', 'Rattanasak',
'1994-07-05', '0912345701', 'F', 'Pizza Baker', 25000.00, 1),
(2, '2023-11-12', 'Cashier', '6001234567890', 'Phatsara', 'Chaiyot',
'1993-09-12', '0923456700', 'F', 'Cashier', 15000.00, 1),
(2, '2023-11-14', 'Cooking', '6002345678901', 'Wuttichai', 'Sukraw',
'1995-11-18', '0923456701', 'M', 'Pizza Baker', 25000.00, 1),
(3, '2023-11-13', 'Cashier', '7001234567890', 'Nathawat', 'Srisai',
'1998-02-03', '0934567800', 'M', 'Cashier', 15000.00, 1),
(3, '2023-11-16', 'Cooking', '7002345678901', 'Chanin', 'Rattanapong',
'1996-03-19', '0934567801', 'F', 'Pizza Baker', 25000.00, 1),
(4, '2023-11-14', 'Cashier', '8001234567890', 'Kittipong', 'Wongthong',
'1991-06-12', '0945678900', 'M', 'Cashier', 15000.00, 1),
(4, '2023-11-17', 'Cooking', '8002345678901', 'Pimnara', 'Mongkol',
'1994-08-29', '0945678901', 'F', 'Pizza Baker', 25000.00, 1),
(5, '2023-11-12', 'Cashier', '9001234567890', 'Sutthira', 'Sukee',
'1996-01-08', '0956789000', 'F', 'Cashier', 15000.00, 1),
(5, '2023-11-15', 'Cooking', '9002345678901', 'Jiratchaya', 'Sangthong',
'1995-12-20', '0956789001', 'M', 'Pizza Baker', 25000.00, 1),
(6, '2023-11-10', 'Cashier', '1001234567890', 'Napatsorn', 'Chutintorn',
'1994-04-25', '0967890100', 'F', 'Cashier', 15000.00, 1),
(6, '2023-11-13', 'Cooking', '1002345678901', 'Worraya', 'Tanatat',
```

```
'1993-07-14', '0967890101', 'M', 'Pizza Baker', 25000.00, 1),
(7, '2023-11-13', 'Cashier', '1101234567890', 'Pimpisa', 'Suwanwat',
'1992-09-07', '0978901230', 'F', 'Cashier', 15000.00, 1),
(7, '2023-11-16', 'Cooking', '1102345678901', 'Chalermpol', 'Kongkaew',
'1995-05-22', '0978901231', 'M', 'Pizza Baker', 25000.00, 1),
(8, '2023-11-14', 'Cashier', '1201234567890', 'Kanokwan', 'Chongkittikul',
'1997-02-05', '0989012340', 'F', 'Cashier', 15000.00, 1),
(8, '2023-11-18', 'Cooking', '1202345678901', 'Pongsakorn', 'Khachok',
'1993-09-30', '0989012341', 'M', 'Pizza Baker', 25000.00, 1),
(9, '2023-11-10', 'Cashier', '1301234567890', 'Suwanit', 'Sathian',
'1994-11-14', '0990123450', 'M', 'Cashier', 15000.00, 1),
(9, '2023-11-13', 'Cooking', '1302345678901', 'Nattapong', 'Sukkhaphirom',
'1996-04-21', '0990123451', 'F', 'Pizza Baker', 25000.00, 1),
(10, '2023-11-12', 'Cashier', '1401234567890', 'Akkarawit', 'Suwannarat',
'1995-09-02', '0901234560', 'M', 'Cashier', 15000.00, 1),
(10, '2023-11-15', 'Cooking', '1402345678901', 'Nuttapong', 'Chintana',
'1994-12-10', '0901234561', 'F', 'Pizza Baker', 25000.00, 1);


INSERT INTO shift (employeeId, shift) VALUES
(1, 'morning'), (1, 'evening'),
(2, 'morning'), (2, 'evening'),
(3, 'morning'), (3, 'evening'),
(4, 'morning'), (4, 'evening'),
(5, 'night'),
(6, 'morning'), (6, 'evening'),
(7, 'morning'), (7, 'evening'),
(8, 'morning'), (8, 'evening'),
(9, 'morning'), (9, 'evening'),
(10, 'night'),
(11, 'morning'), (11, 'evening'),
(12, 'morning'), (12, 'evening'),
(13, 'morning'), (13, 'evening'),
(14, 'morning'), (14, 'evening'),
(15, 'night'),
(16, 'morning'), (16, 'evening'),
(17, 'morning'), (17, 'evening'),
(18, 'morning'), (18, 'evening'),
(19, 'morning'), (19, 'evening'),
(20, 'night'),
(21, 'morning'), (21, 'evening'),
(22, 'morning'), (22, 'evening'),
```

```
(23, 'morning'), (23, 'evening'),
(24, 'morning'), (24, 'evening'),
(25, 'night'),
(26, 'morning'), (26, 'evening'),
(27, 'morning'), (27, 'evening'),
(28, 'morning'), (28, 'evening'),
(29, 'morning'), (29, 'evening'),
(30, 'night'),
(31, 'morning'), (31, 'evening'),
(32, 'morning'), (32, 'evening'),
(33, 'morning'), (33, 'evening'),
(34, 'morning'), (34, 'evening'),
(35, 'night'),
(36, 'morning'), (36, 'evening'),
(37, 'morning'), (37, 'evening'),
(38, 'morning'), (38, 'evening'),
(39, 'morning'), (39, 'evening'),
(40, 'night'),
(41, 'morning'), (41, 'evening'),
(42, 'morning'), (42, 'evening'),
(43, 'morning'), (43, 'evening'),
(44, 'morning'), (44, 'evening'),
(45, 'night'),
(46, 'morning'), (46, 'evening'),
(47, 'morning'), (47, 'evening'),
(48, 'morning'), (48, 'evening'),
(49, 'morning'), (49, 'evening'),
(50, 'night'),
(51, 'night'),
(52, 'night'),
(53, 'night'),
(54, 'night'),
(55, 'night'),
(56, 'night'),
(57, 'night'),
(58, 'night'),
(59, 'night'),
(60, 'night');
```

```sql
INSERT INTO deliveryDriver (employeeId, licensePlate) VALUES
(4, 'กข1234 กรุงเทพมหานคร'),
(5, 'คง5678 ชลบุรี'),
(9, 'จฉ9012 นครราชสีมา'),
(10, 'ฉท3456 ภูเก็ต'),
(14, 'ชพ7890 เชียงใหม่'),
(15, 'ซอ1234 ขอนแก่น'),
(19, 'ฌญ5678 สระบุรี'),
(20, 'ณฑ9012 นครปฐม'),
(24, 'ดบ3456 ปทุมธานี'),
(25, 'ตจ7890 นนทบุรี'),
(29, 'ทม1234 สุพรรณบุรี'),
(30, 'ธบ5678 ระยอง'),
(34, 'นย9012 อยุธยา'),
(35, 'บค3456 สมุทรปราการ'),
(39, 'ปจ7890 สงขลา'),
(40, 'ฝช1234 สุราษฎร์ธานี'),
(44, 'พธ5678 อุบลราชธานี'),
(45, 'ฟน9012 พิษณุโลก'),
(49, 'มป3456 บุรีรัมย์'),
(50, 'ยธ7890 เลย');
```

```sql
INSERT INTO operationZone (employeeId, operationZone) VALUES
(4, 'Wangmai'),
(5, 'Wangmai'),
(5, 'Pathumwan'),
(9, 'Pathumwan'),
(10, 'Pathumwan'),
(14, 'Sam Sen Nok'),
(15, 'Sam Sen Nok'),
(15, 'Sam Sen Nai'),
(19, 'Wangburapa'),
(20, 'Wangburapa'),
(24, 'Ladyao'),
(25, 'Ladyao'),
(25, 'Bang Khen'),
(29, 'Bangkrasor'),
(30, 'Bangkrasor'),
(34, 'Sono Loy'),
(35, 'Sono Loy'),
(39, 'Klong nueng'),
(40, 'Klong nueung '),
(40, 'Klong song'),
(44, 'Klong song'),
(45, 'Klong song'),
(45, 'Klong nueng'),
(49, 'Buengsanan'),
(50, 'Buengsanan');
```

```
INSERT INTO deliveryInformation (deliveredDate) VALUES
('2024-11-01'),
('2024-11-01'),
('2024-11-01'),
('2024-11-02'),
('2024-11-02'),
('2024-11-02'),
('2024-11-03'),
('2024-11-03'),
('2024-11-03'),
('2024-11-04'),
('2024-11-04'),
('2024-11-04'),
('2024-11-05'),
('2024-11-05'),
('2024-11-05'),
('2024-11-06'),
('2024-11-06'),
('2024-11-06'),
('2024-11-07'),
('2024-11-07'),
('2024-11-07'),
('2024-11-07'),
('2024-11-07'),
('2024-11-07'),
('2024-11-07'),
('2024-11-07'),
('2024-11-07'),
('2024-11-07'),
('2024-11-07');
```

```
INSERT INTO product(name,type,price) VALUES
('doble cheese','pizza',419.00),
('doble pepperoni','pizza',419.00),
('crazy cheesy','bite',129.00),
('the bite box setA','set for one',119.00),
('korean style chicken wings','chicken',149.00),
('stir fried macaroni ham and omelet','pasta',129.00),
('french fries','appetizer',69.00),
('caesar salad','salad',99.00),
('pork chop steak with garlic bread','steak',219.00),
('coke','drink',45.00),
('hawaiian delight', 'pizza', 419.00),
('seafood supreme', 'pizza', 459.00),
('mushroom melt', 'pizza', 399.00),
('spicy seafood combo', 'bite', 149.00),
('the bite box set B', 'set for one', 149.00),
('honey bbq chicken wings', 'chicken', 159.00),
('carbonara spaghetti', 'pasta', 149.00),
('crispy chicken strips', 'appetizer', 89.00),
('garden fresh salad', 'salad', 99.00),
('grilled salmon with rice', 'steak', 259.00),
('orange juice', 'drink', 49.00),
('bbq chicken pizza', 'pizza', 429.00),
('bacon explosion', 'pizza', 439.00),
('cheesy bites', 'bite', 139.00),
('the super bite box set A', 'set for one', 169.00),
('spicy korean chicken wings', 'chicken', 169.00),
('spaghetti bolognese', 'pasta', 139.00),
('garlic breadsticks', 'appetizer', 59.00),
('tuna salad', 'salad', 109.00),
('grilled ribeye steak', 'steak', 299.00),
('iced lemon tea', 'drink', 45.00),
('pepperoni lovers', 'pizza', 419.00),
('margherita classic', 'pizza', 389.00),
('crispy cheese sticks', 'bite', 129.00),
('the bite box set C', 'set for one', 149.00),
('garlic parmesan chicken wings', 'chicken', 159.00),
('pesto pasta with chicken', 'pasta', 149.00),
('mozzarella sticks', 'appetizer', 79.00),
('chef's salad', 'salad', 119.00),
('sirloin steak with mashed potatoes', 'steak', 289.00);
```

```
INSERT INTO `order` (orderDate, userID, branchID, deliveryInformationId)
VALUES
('2024-11-01 10:03:17', 1, 1, 1),
('2024-11-01 11:22:05', 2, 2, 2),
('2024-11-01 19:48:31', 3, 3, 3),
('2024-11-02 21:10:44', 4, 4, 4),
('2024-11-02 09:28:57', 5, 5, 5),
('2024-11-02 16:03:22', 6, 6, 6),
('2024-11-03 12:36:41', 7, 7, 7),
('2024-11-03 13:11:58', 8, 8, 8),
('2024-11-03 20:15:10', 9, 9, 9),
('2024-11-04 14:32:11', 10, 10, 10),
('2024-11-04 11:07:14', 1, 1, 11),
('2024-11-04 13:23:47', 2, 2, 12),
('2024-11-05 15:12:05', 3, 3, 13),
('2024-11-05 10:39:49', 4, 4, 14),
('2024-11-05 16:35:42', 5, 5, 15),
('2024-11-06 14:00:17', 6, 6, 16),
('2024-11-06 12:15:23', 7, 7, 17),
('2024-11-06 10:30:42', 8, 8, 18),
('2024-11-07 09:45:13', 9, 9, 19),
('2024-11-07 13:00:29', 10, 10, 20),
('2024-11-07 11:30:07', 1, 1, 21),
('2024-11-07 15:00:53', 2, 2, 22),
('2024-11-07 22:00:18', 3, 3, 23),
('2024-11-07 20:30:09', 4, 4, 24),
('2024-11-07 14:30:33', 5, 5, 25),
('2024-11-07 16:00:45', 6, 6, 26),
('2024-11-07 13:30:11', 7, 7, 27),
('2024-11-07 10:45:27', 8, 8, 28),
('2024-11-07 12:00:32', 9, 9, 29),
('2024-11-07 14:15:19', 10, 10, 30);
```

```
INSERT INTO orderItem (orderId, orderItemId, productId, quantity) VALUES
(1, 1, 5, 3),
(1, 2, 15, 2),
(1, 3, 22, 1),
(2, 1, 9, 4),
(2, 2, 13, 2),
(2, 3, 27, 3),
(3, 1, 7, 1),
(3, 2, 18, 2),
(3, 3, 3, 5),
(4, 1, 12, 4),
(4, 2, 19, 3),
(4, 3, 25, 2),
(5, 1, 30, 1),
(5, 2, 10, 4),
(5, 3, 6, 5),
(6, 1, 16, 2),
(6, 2, 24, 3),
(6, 3, 1, 4),
(7, 1, 11, 2),
(7, 2, 8, 3),
(7, 3, 20, 1),
(8, 1, 28, 5),
(8, 2, 4, 2),
(8, 3, 2, 4),
(9, 1, 17, 3),
(9, 2, 23, 2),
(9, 3, 29, 1),
(10, 1, 21, 4),
(10, 2, 14, 3),
(10, 3, 26, 2),
(11, 1, 18, 3),
(11, 2, 27, 2),
(11, 3, 13, 5),
(12, 1, 25, 1),
(12, 2, 12, 4),
(12, 3, 5, 3),
(13, 1, 30, 2),
(13, 2, 6, 4),
(13, 3, 15, 1),
(14, 1, 3, 3),
(14, 2, 9, 2),
```

```
(14, 3, 7, 5),
(15, 1, 19, 4),
(15, 2, 4, 1),
(15, 3, 14, 3),
(16, 1, 22, 2),
(16, 2, 10, 5),
(16, 3, 8, 3),
(17, 1, 26, 3),
(17, 2, 23, 1),
(17, 3, 12, 4),
(18, 1, 5, 4),
(18, 2, 20, 3),
(18, 3, 27, 2),
(19, 1, 8, 1),
(19, 2, 16, 5),
(19, 3, 3, 4),
(20, 1, 6, 3),
(20, 2, 30, 2),
(20, 3, 1, 4),
(21, 1, 7, 5),
(21, 2, 11, 3),
(21, 3, 17, 2),
(22, 1, 12, 4),
(22, 2, 14, 1),
(22, 3, 18, 3),
(23, 1, 29, 2),
(23, 2, 9, 3),
(23, 3, 20, 5),
(24, 1, 4, 3),
(24, 2, 5, 2),
(24, 3, 30, 1),
(25, 1, 19, 4),
(25, 2, 8, 1),
(25, 3, 22, 3),
(26, 1, 2, 2),
(26, 2, 27, 5),
(26, 3, 25, 3),
(27, 1, 17, 1),
(27, 2, 18, 3),
(27, 3, 3, 4),
(28, 1, 28, 2),
(28, 2, 10, 4),
```

```
(28, 3, 13, 3),
(29, 1, 21, 5),
(29, 2, 30, 3),
(29, 3, 6, 2),
(30, 1, 16, 4),
(30, 2, 9, 1),
(30, 3, 24, 5);


INSERT INTO coupon (name, discountPercent, orderId) VALUES
('Discount10', 10, 1),
('Holiday20', 20, 2),
('Summer15', 15, 3),
('Winter25', 25, 4),
('Promo5', 5, 5),
('Festive30', 30, 6),
('BlackFriday40', 40, 7),
('NewYear50', 50, 8),
('Easter15', 15, 9),
('Christmas25', 25, 10),
('FlashSale30', 30, 11),
('VIPCustomer20', 20, 12),
('Clearance10', 10, 13),
('ExclusiveOffer20', 20, 14),
('EarlyBird10', 10, 15),
('SuperSale50', 50, 11),
('SuperCustomer90', 90, NULL);
```

```
INSERT INTO applicable_coupon (couponId, productId) VALUES
(1, 5),
(2, 13),
(3, 18),
(4, 25),
(5, 30),
(6, 16),
(7, 20),
(8, 2),
(9, 23),
(10, 21),
(11, 27),
(12, 12),
(13, 15),
(14, 7),
(15, 4),
(16, 13),
(17, 1),
(17, 2);
```

```
INSERT INTO `payment` (paidUsing, transactionReference, paidAmount,
paymentDate, orderId) VALUES
('credit card', 'TXN-4829176542', 1129.30, '2024-11-01 10:11:36', 1),
('credit card', 'TXN-1739456201', 1931.40, '2024-11-01 11:36:30', 2),
('credit card', 'TXN-3956217480', 865.30, '2024-11-01 20:05:13', 3),
('cash', 'TXN-4862173950', 2386.5, '2024-11-02 21:35:12', 4),
('credit card', 'TXN-5028341792', 1109.05, '2024-11-02 09:45:33', 5),
('cash', 'TXN-6192837450', 2315.60, '2024-11-02 16:24:50', 6),
('cash', 'TXN-7263194805', 1290.40, '2024-11-03 12:48:02', 7),
('cash', 'TXN-8321746593', 1371.00, '2024-11-03 13:28:41', 8),
('credit card', 'TXN-9317264085', 1302.30, '2024-11-03 20:41:37', 9),
('cash', 'TXN-0418263957', 932.00, '2024-11-04 14:47:56', 10),
('credit card', 'TXN-1592837465', 1459.10, '2024-11-04 11:21:09', 11),
('cash', 'TXN-2683741590', 2084.80, '2024-11-04 13:46:33', 12),
('credit card', 'TXN-3472815906', 1248.10, '2024-11-05 15:31:19', 13),
('cash', 'TXN-4738201659', 1101.00, '2024-11-05 11:08:21', 14),
('cash', 'TXN-5817264903', 950.10, '2024-11-05 16:56:07', 15),
('cash', 'TXN-6928374150', 1380.00, '2024-11-06 14:13:38', 16),
('credit card', 'TXN-7319504826', 2782.00, '2024-11-06 12:33:23', 17),
('cash', 'TXN-8091762543', 1651.00, '2024-11-06 10:42:36', 18),
('credit card', 'TXN-9031654827', 1410.00, '2024-11-07 09:59:42', 19),
('credit card', 'TXN-1572938462', 2661, '2024-11-07 13:15:45', 20),
('credit card', 'TXN-2654718392', 1900.00, '2024-11-07 11:54:19', 21),
('cash', 'TXN-3816542790', 2252.00, '2024-11-07 15:29:14', 22),
('credit card', 'TXN-4928371506', 2170.00, '2024-11-07 22:21:08', 23),
('cash', 'TXN-5039284176', 954.00, '2024-11-07 20:46:11', 24),
('credit card', 'TXN-6172935804', 1782.00, '2024-11-07 14:54:46', 25),
('cash', 'TXN-7283059416', 2040.00, '2024-11-07 16:12:53', 26),
('credit card', 'TXN-8361742905', 932.00, '2024-11-07 13:45:09', 27),
('cash', 'TXN-9173062548', 1495.00, '2024-11-07 10:53:54', 28),
('credit card', 'TXN-0248175936', 1400.00, '2024-11-07 12:29:32', 29),
('cash', 'TXN-1593740286', 1550.00, '2024-11-07 14:38:43', 30);
```

```
INSERT INTO pickupDeliveryInformation (deliveryInformationId, pickupTime,
branchId) VALUES
(1, '10:28:54', 1),
(2, '12:01:17', 2),
(3, '20:40:18', 3),
(4, '22:18:39', 4),
(5, '10:16:02', 5),
(6, '17:21:55', 6),
(7, '13:22:07', 7),
(8, '14:19:48', 8),
(9, '21:23:41', 9),
(10, '15:31:46', 10),
(11, '12:03:55', 1),
(12, '15:14:19', 2),
(13, '16:07:02', 3),
(14, '12:14:29', 4),
(15, '17:45:35', 5);


INSERT INTO homeDeliveryInformation (deliveryInformationId,
savedDeliveryAddressId, deliveryTime, deliveryCharge, employeeId) VALUES
(16, 1, '14:30:18', 150.00, 29),
(17, 2, '12:47:36', 180.00, 34),
(18, 3, '11:02:59', 200.00, 39),
(19, 4, '10:16:27', 160.00, 44),
(20, 5, '13:43:16', 220.00, 49),
(21, 6, '12:05:39', 250.00, 4),
(22, 7, '15:45:28', 140.00, 9),
(23, 8, '22:50:08', 210.00, 15),
(24, 9, '21:06:52', 190.00, 20),
(25, 10, '15:17:46', 230.00, 24),
(26, 11, '16:39:59', 160.00, 29),
(27, 12, '14:13:04', 180.00, 34),
(28, 13, '11:21:41', 210.00, 39),
(29, 14, '12:58:19', 170.00, 44),
(30, 15, '15:07:56', 200.00, 49);
```

5. Data Query

Business Process	Tables Associated With
Branch Operation	Employee, deliveryDriver, operationZone, shift
User Registration and Membership	User, savedDeliveryAddress, membership
Ordering Food	Order, product, orderitem, pickupDeliveryInformation, homeDeliveryInformation, deliveryInformation
Payment of Order	payment
Discount	Coupon, applicable_coupon
Delivery	deliveryInformation, pickupDeliveryInformation, homeDeliveryInformation, deliveryDriver, operationZone,

User Registration and Membership

1. **Report:** Show user count in each province based on the saved delivery address of each user. One user can have more than one saved delivery address.

   Objective: To analyze user distribution across provinces using saved delivery addresses for marketing strategic planning on each region.

```
SELECT
    CASE
        WHEN addressLabel LIKE '%Bangkok%' THEN 'Bangkok'
        WHEN addressLabel LIKE '%Pathumthani%' THEN 'Pathumthani'
        WHEN addressLabel LIKE '%Nonthaburi%' THEN 'Nonthaburi'
    END AS Province,
    COUNT(userId) AS UserCount
FROM `savedDeliveryAddress`
WHERE
    addressLabel LIKE '%Bangkok%'
    OR addressLabel LIKE '%Pathumthani%'
    OR addressLabel LIKE '%Nonthaburi%'
GROUP BY Province;
```

	Province	UserCount
▶	Bangkok	11
	Nonthaburi	2
	Pathumthani	2

Figure 27: The result of the query in User Registration and Membership process

2. **Report:** Total points from membership of each user and their tier list.

   Objective: To give the membership benefits based on their current tier, which is calculated by their current point.

   `Bronze` for points > 0 and < 50.
   `Silver` for points >= 50 and < 300.
   `Gold` for points >= 300.

```sql
SELECT
    CONCAT(firstname," ",lastname) AS Fullname,
    m.totalPoint,
    (
        CASE
            WHEN m.totalPoint >= 0 AND m.totalPoint < 50 THEN 'Bronze'
            WHEN m.totalPoint >= 50 AND m.totalPoint < 300 THEN 'Silver'
            WHEN m.totalPoint >= 300 THEN 'Gold'
            ELSE 'Unknown'
        END
     ) AS membershipTier
FROM `user` u
JOIN `membership` m
ON u.userId = m.userId
WHERE m.totalPoint > 0
ORDER BY m.totalPoint ASC;
```

Fullname	totalPoint	membershipTier
Haseeb Ali	40	Bronze
Vayuphak Saengthong	45	Bronze
Pikul Watdao	100	Silver
Napat Tatiyakaroonwong	150	Silver
Chatdanai Wongsuwan	160	Silver
Yingrat Rattana	170	Silver
Natchapol Lebkrut	250	Silver
Somsri Hiran	320	Gold
Napat Decha	400	Gold

Figure 28: The result of the query in User Registration and Membership process

3. **Report:** To retrieve order information for a specific user by their full name.

   Objective: To analyze the ordering habits or history of a particular user.

   Use ‘Vayuphak Saengthong’ name as an example

```
SELECT
  concat(u.firstName,' ',u.lastName) AS Fullname,
  b.Name AS `Branch Name`,
  o.orderDate
FROM `order` o
INNER JOIN `user` u
ON o.userID = u.userId
INNER JOIN `branch` b
ON o.branchId=b.branchId
WHERE
   u.firstName = 'Vayuphak' AND
   u.lastName = 'Saengthong';
```

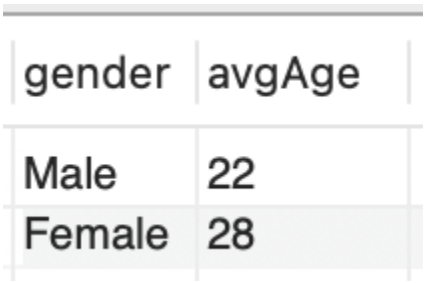
Fullname	Branch Name	orderDate
Vayuphak Saengthong	Pizza Com Samyan Mitrtown	2024-11-01 11:22:05
Vayuphak Saengthong	Pizza Com Samyan Mitrtown	2024-11-04 13:23:47
Vayuphak Saengthong	Pizza Com Samyan Mitrtown	2024-11-07 15:00:53

Figure 29: The result of the query in User Registration and Membership process

4. **Report:** Find the average age of the users group by gender (male and female).

   Objective: To use the information as one of the factors to determine the direction of the marketing strategy based on the average age of all users and their gender.

```
SELECT
    CASE
        WHEN gender = 'M' THEN 'Male'
        WHEN gender = 'F' THEN 'Female'
    END AS gender,
    ROUND(AVG(YEAR(CURDATE()) - YEAR(dateOfBirth)),0) AS avgAge
FROM `user`
GROUP BY gender;
```



gender	avgAge
Male	22
Female	28

Figure 30: The result of the query in User Registration and Membership process

5. **Report:** To show total order amount and membership points by user's age group (e.g., 18-25, 26-35, 36-45, 46+).

   Objective: To analyze spending patterns and loyalty behavior across different demographic groups.

```
SELECT
    CASE
        WHEN TIMESTAMPDIFF(YEAR, u.dateOfBirth, CURDATE()) BETWEEN 18 AND
25 THEN '18-25'
        WHEN TIMESTAMPDIFF(YEAR, u.dateOfBirth, CURDATE()) BETWEEN 26 AND
35 THEN '26-35'
        WHEN TIMESTAMPDIFF(YEAR, u.dateOfBirth, CURDATE()) BETWEEN 36 AND
45 THEN '36-45'
        ELSE '46+'
    END AS age_group,
    SUM(oi.quantity * p.price) AS total_order_amount,
    SUM(m.totalPoint) AS total_points
FROM `user` u
JOIN membership m ON u.userId = m.userId
JOIN `order` o ON u.userId = o.userID
JOIN orderItem oi ON o.orderId = oi.orderId
JOIN product p ON oi.productId = p.productId
GROUP BY age_group;
```

	age_group	total_order_amount	total_points
▶	18-25	35272.00	9405
	26-35	15655.00	5310

Figure 31: The result of the query in User Registration and Membership process

Ordering Food, Payment

1. **Report:** Display 3 top spenders of our company.

   Objective: To identify top spenders allows the company to reward them with loyalty.

```
SELECT
    CONCAT(firstname," ",lastname) AS Fullname,
    SUM(p.paidAmount) AS total_paid
FROM `user` u
LEFT JOIN `order` o
ON u.userId = o.userID
LEFT JOIN `payment` p
ON o.orderId = p.orderId
GROUP BY u.userId
ORDER BY total_paid DESC
LIMIT 3;
```

	Fullname	total_paid
▶	Vayuphak Saengthong	6268.20
	Haseeb Ali	5735.60
	Pikul Watdao	5143.00

Figure 32: The result of the query in Ordering food and Payment process

2. **Report:** Find users who placed at least 2 orders between 2024-11-01 and 2024-11-07.

   Objective: As part of a loyalty campaign, we can identify users that have at least 2 orders within the given time period and give them rewards according to our campaign.

```
SELECT
      CONCAT(firstname," ",lastname) AS Fullname,
    order_count
FROM `user` u
JOIN (
    SELECT
        o.userID AS userId,
        COUNT(o.orderId) AS order_count
    FROM `order` o
    WHERE o.orderDate BETWEEN '2024-11-01' AND '2024-11-07'
    GROUP BY o.userID
    HAVING COUNT(o.orderId) >= 2
) AS order_summary
ON u.userId = order_summary.userId;
```

Fullname	order_count
Natchapol Lebkrut	2
Vayuphak Saengthong	2
Napat Decha	2
Napat Tatiyakaroonwong	2
Chatdanai Wongsuwan	2
Haseeb Ali	2
Somsri Hiran	2
Yingrat Rattana	2

Figure 33: The result of the query in Ordering food and Payment process

3. **Report:** To find the order, ordertype, payment amount and date as well branch information.

   Objective: To retrieve and analyze order details for each branch.

```
SELECT
    o.orderId AS `OrderID`,
    o.orderDate AS `Order Date`,
    CASE
        WHEN pdi.deliveryInformationId IS NOT NULL THEN 'Pickup'
        WHEN hdi.deliveryInformationId IS NOT NULL THEN 'Home Delivery'
        ELSE 'Unknown'
    END AS `OrderType`,
    CONCAT(d.firstName, ' ', d.lastName) AS `Driver`,
    p.paidAmount AS `Paid Amount`,
    p.paymentDate AS `Paid Date`,
    di.deliveredDate AS `Delivered Date`,
    b.name AS 'Pickup From'
FROM `order` o
LEFT JOIN `deliveryInformation` di
ON o.deliveryInformationId = di.deliveryInformationId
LEFT JOIN `homeDeliveryInformation` hdi
ON di.deliveryInformationId = hdi.deliveryInformationId
LEFT JOIN `pickupDeliveryInformation` pdi
ON di.deliveryInformationId = pdi.deliveryInformationId
LEFT JOIN `employee` d
ON hdi.employeeId = d.employeeId -- Driver for home delivery
LEFT JOIN `payment` p
ON o.orderId = p.orderId
LEFT JOIN `branch` b
ON b.branchId = pdi.branchId
ORDER BY o.orderId ASC, o.orderDate ASC;
```

OrderID	Order Date	OrderType	Driver	Paid Amount	Paid Date	Delivered Date	Pickup From
1	2024-11-01 10:03:17	Pickup	NULL	1129.30	2024-11-01 10:11:36	2024-11-01	Pizza Com Lotus Chareonpol Tesc
2	2024-11-01 11:22:05	Pickup	NULL	1931.40	2024-11-01 11:36:30	2024-11-01	Pizza Com Samyan Mitrtown
3	2024-11-01 19:48:31	Pickup	NULL	865.30	2024-11-01 20:05:13	2024-11-01	Pizza Com Meng-Jai
4	2024-11-02 21:10:44	Pickup	NULL	2386.50	2024-11-02 21:35:12	2024-11-02	Pizza Com Ratchabophit
5	2024-11-02 09:28:57	Pickup	NULL	1109.05	2024-11-02 09:45:33	2024-11-02	Pizza Com Big C Ladprao
6	2024-11-02 16:03:22	Pickup	NULL	2315.60	2024-11-02 16:24:50	2024-11-02	Pizza Com Manor Sanambin-Nam
7	2024-11-03 12:36:41	Pickup	NULL	1290.40	2024-11-03 12:48:02	2024-11-03	Pizza Com PTT Sribuathong
8	2024-11-03 13:11:58	Pickup	NULL	1371.00	2024-11-03 13:28:41	2024-11-03	Pizza Com Thummasart Rangsit
9	2024-11-03 20:15:10	Pickup	NULL	1302.30	2024-11-03 20:41:37	2024-11-03	Pizza Com Klongluang
10	2024-11-04 14:32:11	Pickup	NULL	932.00	2024-11-04 14:47:56	2024-11-04	Pizza Com Rangsit Klong 10

Figure 34: The result of the query in Ordering food and Payment process

4. **Report:** To find the paid amount made by the user on each day of the week.

   Objective: To analyze the insight in each day of the week that can be used to improve and revise the marketing strategy differently in each day of the week.

```
SELECT
    CASE
        WHEN DAYOFWEEK(paymentDate) = 1 THEN 'Sunday'
        WHEN DAYOFWEEK(paymentDate) = 2 THEN 'Monday'
        WHEN DAYOFWEEK(paymentDate) = 3 THEN 'Tuesday'
        WHEN DAYOFWEEK(paymentDate) = 4 THEN 'Wednesday'
        WHEN DAYOFWEEK(paymentDate) = 5 THEN 'Thursday'
        WHEN DAYOFWEEK(paymentDate) = 6 THEN 'Friday'
        WHEN DAYOFWEEK(paymentDate) = 7 THEN 'Saturday'
    END AS DayOfWeek,
    SUM(payment.paidAmount) AS TotalPaidAmount
FROM payment
GROUP BY DayOfWeek
ORDER BY TotalPaidAmount DESC;
```

DayOfWeek	TotalPaidAmount
Thursday	20546.00
Wednesday	5813.00
Saturday	5811.15
Monday	4475.90
Sunday	3963.70
Friday	3926.00
Tuesday	3299.20

Figure 35: The result of the query in Ordering food and Payment process

5. **Report:** Identify 3 customers with the lowest spending.

   Objective: To identify the lowest spending on their total spending amount, guiding marketing/sales strategies aimed at increasing spending.

```
SELECT
    CONCAT(firstname," ",lastname) AS Fullname,
    total_spending
FROM `user` u
INNER JOIN (
    SELECT
        o.userID AS userId,
        SUM(p.paidAmount) AS total_spending
    FROM `order` o
    INNER JOIN `payment` p
    ON o.orderId = p.orderId
    GROUP BY o.userID
) spending
ON u.userId = spending.userId
ORDER BY total_spending ASC
LIMIT 3;
```

Fullname	total_spending
Chatdanai Wongsuwan	3841.15
Itthirath Jeamanukul	4112.30
Napat Decha	4283.40

Figure 36: The result of the query in Ordering food and Payment process

6. **Report:** List the order, order items with product info and applied coupons if any, and discounted price along with the final price after discount of each order item.

   Objective: To get in-depth information on each order item, what coupon is used on which order items, and their discounts. This can be put on the receipt of the user's order as evidence to the user and the company.

```
SELECT
    `orderItem`.orderId,
    `orderItem`.orderItemId,
    `orderItem`.quantity,
    `orderItem`.productId,
    `product`.name AS productName,
    `product`.price,
    `coupon`.couponId,
    `coupon`.name AS couponName,
    COALESCE(`coupon`.discountPercent,0) AS discountPercent,
    `product`.price * `orderItem`.quantity AS originalPrice,
    (
        `product`.price * `orderItem`.quantity *
        (COALESCE(`coupon`.discountPercent,0) / 100)
    ) AS discount,
    (
        `product`.price * `orderItem`.quantity *
        (1 - (COALESCE(`coupon`.discountPercent,0) / 100))
    ) AS discountedPrice
FROM `orderItem`
INNER JOIN `product`
ON `orderItem`.productId = `product`.productId
LEFT JOIN `applicable_coupon`
ON `product`.productId = `applicable_coupon`.productId
LEFT JOIN `coupon`
ON
    `orderItem`.orderId = `coupon`.orderId AND
    `applicable_coupon`.couponId = `coupon`.couponId
WHERE
    -- you can put orderId here
    `orderItem`.orderId = 11 AND
    (
        `coupon`.couponId IS NOT NULL OR
        NOT EXISTS (
            SELECT 1
            FROM `orderItem` oi
            INNER JOIN `product` p
            ON oi.productId = p.productId
            INNER JOIN `applicable_coupon` ac
            ON p.productId = ac.productId
            INNER JOIN `coupon` c
            ON
                oi.orderId = c.orderId AND
                ac.couponId = c.couponId
            WHERE
                oi.orderId = `orderItem`.orderId AND
                oi.orderItemId = `orderItem`.`orderItemId`
        )
```

```
    )
ORDER BY `orderItem`.orderItemId ASC;
```

orderId	orderItemId	quantity	productId	productName	price	couponId	couponName	discountPercent	originalPrice	discount	discountedPrice
11	1	3	18	crispy chicken strips	89.00	NULL	NULL	0	267.00	0.000000	267.000000
11	2	2	27	spaghetti bolognese	139.00	11	FlashSale30	30	278.00	83.400000	194.600000
11	3	5	13	mushroom melt	399.00	16	SuperSale50	50	1995.00	997.500000	997.500000

Figure 37: The result of the query in Ordering food and Payment process

7. **Report:** Calculates the total price of all orders after applying coupons.

   Objective: provides more view of actual income generated from sales, as it accounts for any reductions in the price due to coupons.

```
SELECT
  `itemDiscounts`.orderId,
  (
    CASE
      WHEN SUM(
        `itemDiscounts`.quantity * `itemDiscounts`.price *
        COALESCE(-`itemDiscounts`.discountPercent, 100)/100
      ) < 0 THEN 0
      ELSE SUM(
        `itemDiscounts`.quantity * `itemDiscounts`.price *
        COALESCE(-`itemDiscounts`.discountPercent, 100)/100
      )
    END
  ) AS total_price
FROM (
  SELECT
    `orderItem`.orderId,
    `orderItem`.quantity,
    `product`.price,
    `coupon`.discountPercent
  FROM `orderItem`
  INNER JOIN `product`
  ON `orderItem`.productId = `product`.productId
  LEFT JOIN `coupon`
  ON `orderItem`.orderId = `coupon`.orderId
  INNER JOIN `applicable_coupon`
  ON
    `orderItem`.productId = `applicable_coupon`.productId AND
    `coupon`.couponId = `applicable_coupon`.couponId
  UNION
  SELECT
    `orderItem`.orderId,
    `orderItem`.quantity,
    `product`.price,
    NULL AS discountPercent
  FROM `orderItem`
  INNER JOIN `product`
  ON `orderItem`.productId = `product`.productId
) AS `itemDiscounts`
GROUP BY `itemDiscounts`.orderId
ORDER BY `itemDiscounts`.orderId ASC;
```

orderId	total_price
1	1129.300000
2	1931.400000
3	865.300000
4	2386.500000
5	1109.050000
6	2315.600000
7	1290.400000
8	1371.000000
9	1302.300000
10	932.000000
11	1459.100000
12	2084.800000
13	1248.100000
14	1101.000000
15	950.100000
16	1380.000000
17	2782.000000
18	1651.000000
19	1410.000000
20	2661.000000
21	1900.000000
22	2252.000000
23	2170.000000
24	954.000000
25	1782.000000

Figure 38: The result of the query in Ordering food and Payment process(first 25 output)

8. **Report:** List all orders placed by a user who paid using a credit card.

   Objective: To retrieve and display information of all orders placed with a credit card.

```
SELECT
    o.orderDate,
    u.firstName,
    u.lastName,
    u.email,
    p.paidAmount,
    p.paidUsing,
    p.paymentDate,
    oi.quantity,
    pr.name AS productName,
    pr.price AS productPrice,
    pr.type AS productType
FROM `order` o
INNER JOIN `user` u
ON o.userID = u.userId
INNER JOIN `payment` p
ON o.orderId = p.orderId
LEFT JOIN `orderItem` oi
ON o.orderId = oi.orderId
LEFT JOIN `product` pr
ON oi.productId = pr.productId
WHERE p.paidUsing = 'credit card';
```

orderDate	firstName	lastName	email	paidAmount	paidUsing	paymentDate	quantity	productName	productPrice	productType
2024-11-01 10:03:17	Natchapol	Lebkrut	natchapol.lebkrut@db.com	1129.30	credit card	2024-11-01 10:11:36	3	korean style chicken wings	149.00	chicken
2024-11-01 10:03:17	Natchapol	Lebkrut	natchapol.lebkrut@db.com	1129.30	credit card	2024-11-01 10:11:36	2	the bite box set B	149.00	set for one
2024-11-01 10:03:17	Natchapol	Lebkrut	natchapol.lebkrut@db.com	1129.30	credit card	2024-11-01 10:11:36	1	bbq chicken pizza	429.00	pizza
2024-11-01 11:22:05	Vayuphak	Saengthong	vayuphak.saengthong@db.com	1931.40	credit card	2024-11-01 11:36:30	4	pork chop steak with garlic bread	219.00	steak
2024-11-01 11:22:05	Vayuphak	Saengthong	vayuphak.saengthong@db.com	1931.40	credit card	2024-11-01 11:36:30	2	mushroom melt	399.00	pizza
2024-11-01 11:22:05	Vayuphak	Saengthong	vayuphak.saengthong@db.com	1931.40	credit card	2024-11-01 11:36:30	3	spaghetti bolognese	139.00	pasta
2024-11-01 19:48:31	Napat	Decha	napat.decha@db.com	865.30	credit card	2024-11-01 20:05:13	1	french fries	69.00	appetizer
2024-11-01 19:48:31	Napat	Decha	napat.decha@db.com	865.30	credit card	2024-11-01 20:05:13	2	crispy chicken strips	89.00	appetizer
2024-11-01 19:48:31	Napat	Decha	napat.decha@db.com	865.30	credit card	2024-11-01 20:05:13	5	crazy cheesy	129.00	bite
2024-11-02 09:28:57	Chatdanai	Wongsuwan	chatdanai.wongsuwan@db.com	1109.05	credit card	2024-11-02 09:45:33	1	grilled ribeye steak	299.00	steak
2024-11-02 09:28:57	Chatdanai	Wongsuwan	chatdanai.wongsuwan@db.com	1109.05	credit card	2024-11-02 09:45:33	4	coke	45.00	drink
2024-11-02 09:28:57	Chatdanai	Wongsuwan	chatdanai.wongsuwan@db.com	1109.05	credit card	2024-11-02 09:45:33	5	stir fried macaroni ham and omelet	129.00	pasta
2024-11-03 20:15:10	Itthirath	Jeamanukul	itthirath.jeamanukul@db.com	1302.30	credit card	2024-11-03 20:41:37	3	carbonara spaghetti	149.00	pasta
2024-11-03 20:15:10	Itthirath	Jeamanukul	itthirath.jeamanukul@db.com	1302.30	credit card	2024-11-03 20:41:37	2	bacon explosion	439.00	pizza
2024-11-03 20:15:10	Itthirath	Jeamanukul	itthirath.jeamanukul@db.com	1302.30	credit card	2024-11-03 20:41:37	1	tuna salad	109.00	salad
2024-11-04 11:07:14	Natchapol	Lebkrut	natchapol.lebkrut@db.com	1459.10	credit card	2024-11-04 11:21:09	3	crispy chicken strips	89.00	appetizer
2024-11-04 11:07:14	Natchapol	Lebkrut	natchapol.lebkrut@db.com	1459.10	credit card	2024-11-04 11:21:09	2	spaghetti bolognese	139.00	pasta
2024-11-04 11:07:14	Natchapol	Lebkrut	natchapol.lebkrut@db.com	1459.10	credit card	2024-11-04 11:21:09	5	mushroom melt	399.00	pizza

Figure 39: The result of the query in Ordering food and Payment process

9. **Report:** List all orders along with the customer name, total amount, and payment status.

   Objective: To track whether an order has been paid or not, improving cash flow monitoring.

```
SELECT
   o.orderID,
   u.firstName,
   u.lastName,
   p.paidAmount,
   (
       CASE
          WHEN p.paymentID IS NOT NULL
          THEN 'Paid' ELSE 'Not Paid'
       END
   ) AS paymentStatus
FROM `order` o
INNER JOIN `user` u
ON o.userID = u.userID
LEFT JOIN `payment` p
ON o.orderID = p.orderID
ORDER BY paidAmount;
```

orderID	firstName	lastName	paidAmount	paymentStatus
3	Napat	Decha	865.30	Paid
27	Somsri	Hiran	932.00	Paid
10	Pikul	Watdao	932.00	Paid
15	Chatdanai	Wongsuwan	950.10	Paid
24	Napat	Tatiyakaroonwong	954.00	Paid
14	Napat	Tatiyakaroonwong	1101.00	Paid
5	Chatdanai	Wongsuwan	1109.05	Paid
1	Natchapol	Lebkrut	1129.30	Paid
13	Napat	Decha	1248.10	Paid
7	Somsri	Hiran	1290.40	Paid
9	Itthirath	Jeamanukul	1302.30	Paid
8	Yingrat	Rattana	1371.00	Paid
16	Haseeb	Ali	1380.00	Paid

Figure 40: The result of the query in Ordering food and Payment process

10. **Report:** Find the top 3 most frequently ordered pizzas.

    Objective: To identify the top 3 pizzas that are ordered the most frequently, understanding customer preferences, and allowing to promote popular pizzas.

```
SELECT
    p.name AS productName,
    COUNT(oi.orderId) AS orderCount
FROM `product` p
INNER JOIN `orderItem` oi
ON p.productId = oi.productId
INNER JOIN `order` o
ON oi.orderId = o.orderId
WHERE p.type = 'pizza'
GROUP BY p.name
ORDER BY orderCount DESC
LIMIT 3;
```

productName	orderCount
seafood supreme	4
mushroom melt	3
bbq chicken pizza	3

Figure 41: The result of the query in Ordering food and Payment process

11. **Report:** User who orders 5 different products and counts their distinct.

    Objective: To get insight into customer behavior on the choice of their ordering product.

```
SELECT
   u.userId,
   u.firstName,
   u.lastName,
   COUNT(DISTINCT oi.productId) AS productCount
FROM `user` u
INNER JOIN `order` o
ON u.userId = o.userID
INNER JOIN `orderItem` oi
ON o.orderId = oi.orderId
GROUP BY
   u.userId,
   u.firstName,
   u.lastName
HAVING productCount > 5;
```

	userId	firstName	lastName	productCount
▶	1	Natchapol	Lebkrut	6
	2	Vayuphak	Saengthong	6
	3	Napat	Decha	6
	4	Napat	Tatiyakaroonwong	6
	5	Chatdanai	Wongsuwan	6

Figure 42: The result of the query in Ordering food and Payment process

Branch Operation

1. **Report:** Count the full-time and non-full-time employees for each branch.

   Objective: This can be utilized to schedule employees so that we can redirect the workforce for different tasks. It can also be used for hiring purposes to both assess employee requirements and maybe in order to mitigate costs when needed.

```
SELECT
    b.name AS branch_name,
    SUM(CASE WHEN e.isFullTime = 1 THEN 1 ELSE 0 END) AS `Number of
Fulltime`,
    SUM(CASE WHEN e.isFullTime = 0 THEN 1 ELSE 0 END) AS `Number of
Parttime`
FROM `branch` b
LEFT JOIN `employee` e
ON b.branchId = e.branchId
GROUP BY b.branchId, b.name
ORDER BY b.name;
```

branch_name	Number of Fulltime	Number of Parttime
Pizza Com Big C Ladprao	6	1
Pizza Com Klongluang	6	1
Pizza Com Lotus Chareonpol Tesco Lotus Rama 1	6	1
Pizza Com Manor Sanambin-Nam	6	1
Pizza Com Meng-Jai	6	1
Pizza Com PTT Sribuathong	6	1
Pizza Com Rangsit Klong 10	6	1
Pizza Com Ratchabophit	6	1
Pizza Com Samyan Mitrtown	6	1
Pizza Com Thummasart Rangsit	6	1

Figure 43: The result of the query in Branch operation process

2. **Report:** List all employees with more than one shift.

   Objective: It can be used in identifying employees with multiple shifts, helping manage workload, avoid burnout, and ensure efficient shift allocation.

```
SELECT
    b.name AS branch_name,
    e.firstName,
    e.lastName,
    COUNT(DISTINCT s.shift) AS shift_count
FROM `branch` b
LEFT JOIN `employee` e
ON b.branchId = e.branchId
LEFT JOIN `shift` s
ON e.employeeId = s.employeeId
GROUP BY e.employeeId, b.branchId
HAVING shift_count > 1;
```

branch_name	firstName	lastName	shift_count
Pizza Com Lotus Chareonpol Tesco Lotus Rama 1	Anan	Phongchai	2
Pizza Com Lotus Chareonpol Tesco Lotus Rama 1	Thanakorn	Srisuk	2
Pizza Com Lotus Chareonpol Tesco Lotus Rama 1	Sirin	Kittithorn	2
Pizza Com Lotus Chareonpol Tesco Lotus Rama 1	Nipa	Wanich	2
Pizza Com Samyan Mitrtown	Kanok	Phongpan	2
Pizza Com Samyan Mitrtown	Somsak	Thammasak	2
Pizza Com Samyan Mitrtown	Preecha	Wiriyaporn	2
Pizza Com Samyan Mitrtown	Saowaluk	Yamsri	2
Pizza Com Meng-Jai	Manas	Phutthachot	2
Pizza Com Meng-Jai	Wiroj	Suchat	2
Pizza Com Meng-Jai	Weerachai	Phrompakdee	2
Pizza Com Meng-Jai	Kamol	Sriboon	2
Pizza Com Ratchabophit	Pornchai	Thammasak	2
Pizza Com Ratchabophit	Supaporn	Yokchai	2
Pizza Com Ratchabophit	Narong	Tosapon	2
Pizza Com Ratchabophit	Ratchanee	Sungthong	2
Pizza Com Big C Ladprao	Rungnapa	Chaisarn	2
Pizza Com Big C Ladprao	Yuthana	Namwong	2
Pizza Com Big C Ladprao	Somchai	Kittipong	2
Pizza Com Big C Ladprao	Rattana	Anantachai	2
Pizza Com Manor Sanambin-Nam	Chompoo	Rattanasuk	2
Pizza Com Manor Sanambin-Nam	Wanchai	Prasert	2
Pizza Com Manor Sanambin-Nam	Anong	Pimkarn	2
Pizza Com Manor Sanambin-Nam	Phirun	Chakrit	2
Pizza Com PTT Sribuathong	Yingyai	Sudjai	2
Pizza Com PTT Sribuathong	Kasem	Sukthang	2
Pizza Com PTT Sribuathong	Ritthikorn	Nakarach	2
Pizza Com PTT Sribuathong	Chaisiri	Chirawat	2
Pizza Com Thummasart Rangsit	Burin	Somjit	2
Pizza Com Thummasart Rangsit	Kanlayanee	Pathawee	2
Pizza Com Thummasart Rangsit	Korn	Kritsana	2
Pizza Com Thummasart Rangsit	Yuwadee	Niwat	2
Pizza Com Klongluang	Wut	Saengchan	2
Pizza Com Klongluang	Thitiporn	Sutthipong	2
Pizza Com Klongluang	Kusuma	Wongthong	2
Pizza Com Klongluang	Chanchai	Tanasuk	2
Pizza Com Rangsit Klong 10	Prapai	Chayachote	2
Pizza Com Rangsit Klong 10	Chinnakorn	Namchai	2
Pizza Com Rangsit Klong 10	Worachai	Phongpan	2
Pizza Com Rangsit Klong 10	Siriporn	Jindaporn	2

Figure 44: The result of the query in Branch operation process

3. **Report:** Finds the branch with the highest revenue.

   Objective: To find the branch with the highest revenue to be used as the successful model for the other branch in order to increase the revenue.

```
SELECT
    b.name AS branch_name,
    SUM(p.paidAmount) AS total_revenue
FROM `branch` b
LEFT JOIN `order` o
ON b.branchId = o.branchID
LEFT JOIN `payment` p
ON o.orderId = p.orderId
GROUP BY b.branchId
ORDER BY total_revenue DESC
LIMIT 1;
```

	branch_name	total_revenue
▶	Pizza Com Samyan Mitrtown	6268.20

Figure 45: The result of the query in Branch operation process

4. **Report:** Find each branch home orders ratio to total orders count and total driver count.

   Objective: For each branch, we can see how frequent the home orders are relative to the total number of orders from the ratio then, looking at the total driver count, determine if there are enough drivers for that branch or not so we can hire more drivers if needed.

```
SELECT
    `branch_order_delivery`.branchId,
    `branch_order_delivery`.name,
    (pickup_count + home_count) AS total_orders_count,
    (home_count / (pickup_count + home_count)) AS home_orders_ratio,
    COUNT(*) AS total_driver_count
FROM (
    SELECT
        `branch`.branchId,
        `branch`.name,
        `branch`.phoneNumber,
        COUNT(
            CASE WHEN `pickupDeliveryInformation`.deliveryInformationId IS NOT NULL
            THEN 1 ELSE NULL END
        ) AS pickup_count,
        COUNT(
            CASE WHEN `homeDeliveryInformation`.deliveryInformationId IS NOT NULL
            THEN 1 ELSE NULL END
        ) AS home_count
    FROM `order`
    INNER JOIN `branch`
    ON `order`.branchId = `branch`.branchId
    LEFT JOIN `pickupDeliveryInformation`
    ON `order`.deliveryInformationId = `pickupDeliveryInformation`.deliveryInformationId
    LEFT JOIN `homeDeliveryInformation`
    ON `order`.deliveryInformationId = `homeDeliveryInformation`.deliveryInformationId
    GROUP BY `branch`.branchId
) AS `branch_order_delivery`
INNER JOIN `employee`
ON `branch_order_delivery`.branchId = `employee`.branchId
INNER JOIN `deliveryDriver`
ON `employee`.employeeId = `deliveryDriver`.employeeId
GROUP BY `branch_order_delivery`.branchId
ORDER BY home_orders_ratio DESC, total_orders_count DESC, total_driver_count ASC;
```

branchId	name	total_orders_count	home_orders_ratio	total_driver_count
6	Pizza Com Manor Sanambin-Nam	3	0.6667	2
7	Pizza Com PTT Sribuathong	3	0.6667	2
8	Pizza Com Thummasart Rangsit	3	0.6667	2
9	Pizza Com Klongluang	3	0.6667	2
10	Pizza Com Rangsit Klong 10	3	0.6667	2
1	Pizza Com Lotus Chareonpol Tesco Lotus Rama 1	3	0.3333	2
2	Pizza Com Samyan Mitrtown	3	0.3333	2
3	Pizza Com Meng-Jai	3	0.3333	2
4	Pizza Com Ratchabophit	3	0.3333	2
5	Pizza Com Big C Ladprao	3	0.3333	2

Figure 46: The result of the query in Branch operation process

5. **Report:** Counts the number of employees for each shift of each branch.

   Objective: For each branch, we can balance the number of employees in each shift to ensure that there are enough employees available at all times.

```
SELECT
    `branch`.branchId,
    `branch`.name,
    COUNT(
        CASE WHEN `shift`.shift = 'morning'
        THEN 1 ELSE NULL END
    ) AS morning_count,
    COUNT(
        CASE WHEN `shift`.shift = 'evening'
        THEN 1 ELSE NULL END
    ) AS evening_count,
    COUNT(
        CASE WHEN `shift`.shift = 'night'
        THEN 1 ELSE NULL END
    ) AS night_count
FROM `shift`
INNER JOIN `employee`
ON `shift`.employeeId = `employee`.employeeId
INNER JOIN `branch`
ON `employee`.branchId = `branch`.branchId
GROUP BY `branch`.branchId
ORDER BY night_count ASC;
```

branchId	name	morning_count	evening_count	night_count
6	Pizza Com Manor Sanambin-Nam	4	4	1
7	Pizza Com PTT Sribuathong	4	4	1
8	Pizza Com Thummasart Rangsit	4	4	1
9	Pizza Com Klongluang	4	4	1
10	Pizza Com Rangsit Klong 10	4	4	1
1	Pizza Com Lotus Chareonpol Tesco Lotus Rama 1	4	4	3
2	Pizza Com Samyan Mitrtown	4	4	3
3	Pizza Com Meng-Jai	4	4	3
4	Pizza Com Ratchabophit	4	4	3
5	Pizza Com Big C Ladprao	4	4	3

Figure 47: The result of the query in Branch operation process

6. **Report:** Display the top 10 employees with the longest tenure in the company.

   Objective: To identify employees with the longest tenure to acknowledge and reward their loyalty to the company.

```
SELECT
    `employee`.employeeId,
    `employee`.firstName,
    `employee`.lastName,
    YEAR(CURDATE()) - YEAR(`employee`.joiningDate) AS active_years
FROM `employee`
ORDER BY
    active_years DESC,
    `employee`.firstName ASC
LIMIT 10;
```

employeeId	firstName	lastName	active_years
16	Pornchai	Thammasak	8
1	Anan	Phongchai	7
36	Burin	Somjit	7
26	Chompoo	Rattanasuk	7
6	Kanok	Phongpan	7
11	Manas	Phutthachot	7
46	Prapai	Chayachote	7
21	Rungnapa	Chaisarn	7
41	Wut	Saengchan	7
31	Yingyai	Sudjai	7

Figure 48: The result of the query in Branch operation process

7. **Report:** Finds the past 5 years, calculating the average, standard deviation, minimum, and maximum of the monthly revenue of the whole company.

   Objective: This query aids in performance analysis, providing insights into the company's financial trends over the past five years to support strategic planning and revenue forecasting.

```
SELECT
    `monthly_revenue`.year,
    ROUND(AVG(`monthly_revenue`.revenue),0) AS avg_monthly_revenue,
    ROUND(STDDEV(`monthly_revenue`.revenue),0) AS stddev_monthly_revenue,
    ROUND(MIN(`monthly_revenue`.revenue),0) AS min_monthly_revenue,
    ROUND(MAX(`monthly_revenue`.revenue),0) AS max_monthly_revenue
FROM (
    SELECT
        year,
        month,
        SUM(COALESCE(paidAmount,0)) AS revenue
    FROM (
        SELECT YEAR(CURDATE()) as year
        UNION ALL SELECT YEAR(CURDATE()) - 1
        UNION ALL SELECT YEAR(CURDATE()) - 2
        UNION ALL SELECT YEAR(CURDATE()) - 3
        UNION ALL SELECT YEAR(CURDATE()) - 4
    ) as `last_5_years`
    INNER JOIN (
        SELECT 1 as month
        UNION ALL SELECT 2 UNION ALL SELECT 3 UNION ALL SELECT 4
        UNION ALL SELECT 5 UNION ALL SELECT 6 UNION ALL SELECT 7
        UNION ALL SELECT 8 UNION ALL SELECT 9 UNION ALL SELECT 10
        UNION ALL SELECT 11 UNION ALL SELECT 12
    ) as `months`
    ON true
    LEFT JOIN `order`
    ON
        YEAR(`order`.orderDate) = year AND
        MONTH(`order`.orderDate) = month
    LEFT JOIN `payment`
    ON `order`.orderId = `payment`.orderId
    GROUP BY month,year
) AS `monthly_revenue`
GROUP BY `monthly_revenue`.year
ORDER BY `monthly_revenue`.year ASC;
```

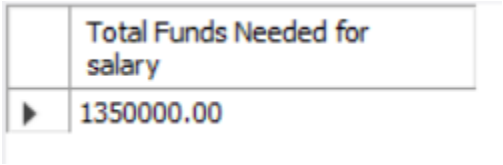
year	avg_monthly_revenue	stddev_monthly_reven...	min_monthly_reven...	max_monthly_reven...
2020	0	0	0	0
2021	0	0	0	0
2022	0	0	0	0
2023	0	0	0	0
2024	3986	13221	0	47835

Figure 49: The result of the query in Branch operation process

8. **Report:** Calculates the funds required for base salaries of all employees across all branches**.**

   Objective: To know how much funds to allocate to employee salaries each month.

```
SELECT
    SUM(`employee`.salary) AS "Total Funds Needed for salary"
FROM `employee`;
```



	Total Funds Needed for salary
▶	1350000.00

Figure 50: The result of the query in Branch operation process

9. **Report:** calculates the frequencies of the method of payment used at each branch for all of the orders handled by that particular branch.

   Objective: This query supports payments by analyzing payment method preferences at each branch, helping tailor services, optimize payment options, and improve customer experience.

```
SELECT
    `branch`.branchId,
    `branch`.name,
    COUNT(
        CASE WHEN `payment`.paidUsing = 'credit card'
        THEN 1 ELSE NULL END
    ) AS paid_by_credit_card,
    COUNT(
        CASE WHEN `payment`.paidUsing = 'cash'
        THEN 1 ELSE NULL END
    ) AS paid_by_cash
FROM `payment`
INNER JOIN `order`
ON `payment`.orderId = `order`.orderId
INNER JOIN `branch`
ON `order`.branchId = `branch`.branchId
GROUP BY `branch`.branchId
ORDER BY `branch`.branchId;
```

branchId	name	paid_by_credit_card	paid_by_cash
1	Pizza Com Lotus Chareonpol Tesco Lotus Rama 1	3	0
2	Pizza Com Samyan Mitrtown	1	2
3	Pizza Com Meng-Jai	3	0
4	Pizza Com Ratchabophit	0	3
5	Pizza Com Big C Ladprao	2	1
6	Pizza Com Manor Sanambin-Nam	0	3
7	Pizza Com PTT Sribuathong	2	1
8	Pizza Com Thummasart Rangsit	0	3
9	Pizza Com Klongluang	3	0
10	Pizza Com Rangsit Klong 10	1	2

Figure 51: The result of the query in Branch operation process

10. **Report:** Finds all of the employees who have joined the Pizza Company in the last 2 years, along with their respective departments and information.

    Objective: This query supports Branch Operation by tracking recent hires, their roles, and departments, ensuring proper onboarding, resource allocation, and workforce planning.

```sql
SELECT
    employeeId,
    firstName,
    lastName,
    department,
    joiningDate
FROM `employee`
WHERE joiningDate >= DATE_SUB(CURDATE(), INTERVAL 2 YEAR);
```

Fullname	Department	JoiningDate
Narumon Rattanasak	Cooking	2023-11-18
Pongsakorn Khachok	Cooking	2023-11-18
Pimnara Mongkol	Cooking	2023-11-17
Chanin Rattanapong	Cooking	2023-11-16
Chalermpol Kongkaew	Cooking	2023-11-16
Nuttapong Chintana	Cooking	2023-11-15
Kittisak Sittiporn	Cashier	2023-11-15
Jiratchaya Sangthong	Cooking	2023-11-15
Kanokwan Chongkittikul	Cashier	2023-11-14
Kittipong Wongthong	Cashier	2023-11-14
Wuttichai Sukraw	Cooking	2023-11-14
Pimpisa Suwanwat	Cashier	2023-11-13
Worraya Tanatat	Cooking	2023-11-13

Figure 52: The result of the query in Branch operation process

11. **Report:** Finds all of the employees who have been assigned to a morning shift.

    Objective: This query helps Branch Operation by tracking morning shift employees for schedule management, staffing, and accurate compensation. It can also be used to identify employees for any activity that might require morning shift employees.

```
SELECT
CONCAT(firstname," ",lastname) AS Fullname,s.shift
FROM `employee` e
INNER JOIN `shift` s
ON e.employeeId = s.employeeId
WHERE s.shift = 'morning'
ORDER BY Fullname;
```

Fullname	shift
Anan Phongchai	morning
Anong Pimkarn	morning
Burin Somjit	morning
Chaisiri Chirawat	morning
Chanchai Tanasuk	morning
Chinnakorn Namchai	morning
Chompoo Rattanasuk	morning
Kamol Sriboon	morning
Kanlayanee Pathawee	morning
Kanok Phongpan	morning
Kasem Sukthang	morning
Korn Kritsana	morning
Kusuma Wongthong	morning
Manas Phutthachot	morning
Narong Tosapon	morning
Nipa Wanich	morning

Figure 53: The result of the query in Branch operation process

12. **Report:** Lists all of the delivery drivers with their license plates ordered by salary non-increasing.

    Objective: This query can be used for salary analysis and reviewing salary distribution among delivery drivers for budgeting and fairness.

```
SELECT
    e.employeeId,
    e.firstName,
    e.lastName,
    e.department,
    d.licensePlate,
    e.salary
FROM `employee` e
INNER JOIN `deliveryDriver` d
ON e.employeeId = d.employeeId
ORDER BY e.salary DESC;
```

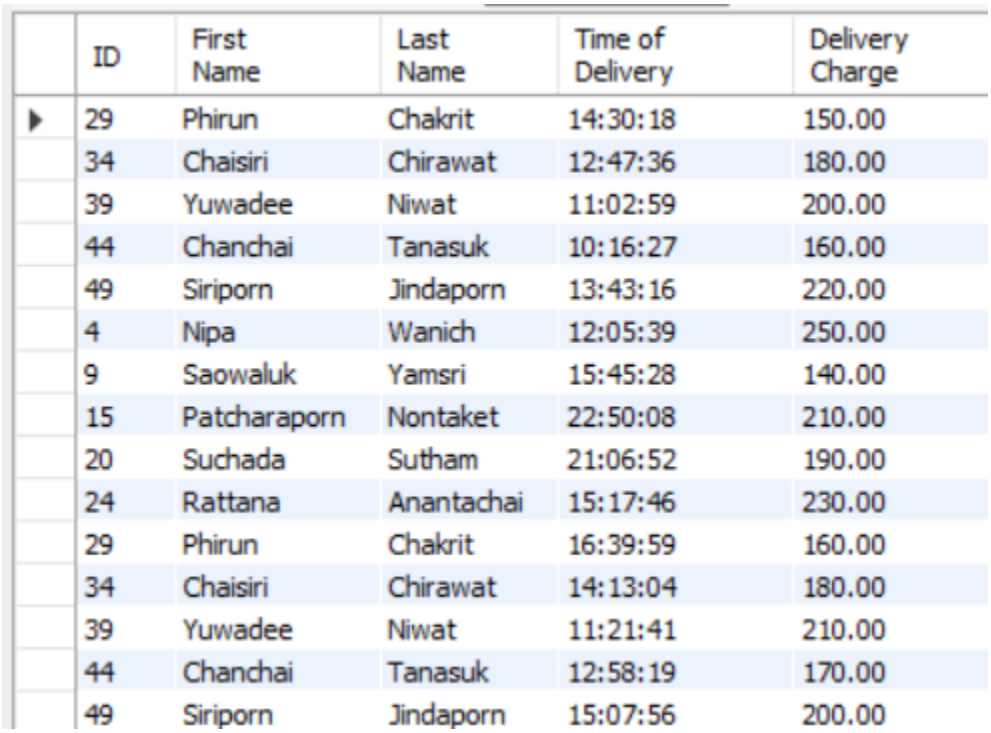
employeeId	firstName	lastName	department	licensePlate	salary
4	Nipa	Wanich	Delivery	กข1234 กรุงเทพมหานคร	20000.00
9	Saowaluk	Yamsri	Delivery	จฉ9012 นครราชสีมา	20000.00
14	Kamol	Sriboon	Delivery	ชพ7890 เชียงใหม่	20000.00
19	Ratchanee	Sungthong	Delivery	ฌญ5678 สระบุรี	20000.00
24	Rattana	Anantachai	Delivery	ดบ3456 ปทุมธานี	20000.00
29	Phirun	Chakrit	Delivery	ทม1234 สุพรรณบุรี	20000.00
34	Chaisiri	Chirawat	Delivery	นย9012 อยุธยา	20000.00
39	Yuwadee	Niwat	Delivery	ปจ7890 สงขลา	20000.00
44	Chanchai	Tanasuk	Delivery	พธ5678 อุบลราชธานี	20000.00
49	Siriporn	Jindaporn	Delivery	มป3456 บุรีรัมย์	20000.00
5	Nattapong	Boonmee	Delivery	คง5678 ชลบุรี	NULL
10	Thida	Sukjai	Delivery	ฉท3456 ภูเก็ต	NULL
15	Patcharap...	Nontaket	Delivery	ชอ1234 ขอนแก่น	NULL
20	Suchada	Sutham	Delivery	ฌท9012 นครปฐม	NULL
25	Udom	Jaiklang	Delivery	ตจ7890 นนทบุรี	NULL
30	Pakorn	Sukwattana	Delivery	ธบ5678 ระยอง	NULL
35	Siriporn	Dhanabut	Delivery	บค3456 สมุทรปราการ	NULL
40	Pharanee	Jirasri	Delivery	ฝช1234 สุราษฎร์ธานี	NULL
45	Noknoi	Pattana	Delivery	ฟน9012 พิษณุโลก	NULL
50	Phairoj	Thongdee	Delivery	ยธ7890 เลย	NULL

Figure 54: The result of the query in Branch operation process

13. **Report:** Lists delivery driver details and their corresponding delivery time and charge from the homeDeliveryInformation.

    Objective: It can be used to ensure that delivery times align with company promises, like 30-minute delivery. Furthermore, it can be used to evaluate individual driver performance based on delivery efficiency and charges.

```
SELECT
    e.employeeId AS `ID`,
    e.firstName AS `First Name`,
    e.lastName AS `Last Name`,
    hd.deliveryTime AS `Time of Delivery`,
    hd.deliveryCharge AS `Delivery Charge`
FROM `employee` e
INNER JOIN `homeDeliveryInformation` hd
ON e.employeeId = hd.employeeId;
```



ID	First Name	Last Name	Time of Delivery	Delivery Charge
29	Phirun	Chakrit	14:30:18	150.00
34	Chaisiri	Chirawat	12:47:36	180.00
39	Yuwadee	Niwat	11:02:59	200.00
44	Chanchai	Tanasuk	10:16:27	160.00
49	Siriporn	Jindaporn	13:43:16	220.00
4	Nipa	Wanich	12:05:39	250.00
9	Saowaluk	Yamsri	15:45:28	140.00
15	Patcharaporn	Nontaket	22:50:08	210.00
20	Suchada	Sutham	21:06:52	190.00
24	Rattana	Anantachai	15:17:46	230.00
29	Phirun	Chakrit	16:39:59	160.00
34	Chaisiri	Chirawat	14:13:04	180.00
39	Yuwadee	Niwat	11:21:41	210.00
44	Chanchai	Tanasuk	12:58:19	170.00
49	Siriporn	Jindaporn	15:07:56	200.00

Figure 55: The result of the query in Branch operation process

14. **Report:** Retrieves the operation zone of all the highest-paid delivery drivers along with their salaries.

    Objective: It can be used to identify top-paid drivers and assign them to critical or high-demand operation zones. Ensuring that the highest-paid drivers are operating in areas that align with their experience and compensation.

```
SELECT
    e.employeeId,
    e.firstName,
    e.lastName,
    o.operationZone,
    e.salary
FROM `employee` e
INNER JOIN `deliveryDriver` d
ON e.employeeId = d.employeeId
INNER JOIN `operationZone` o
ON o.employeeId = d.employeeId
WHERE e.salary = (
    SELECT MAX(e.salary)
    FROM `employee` ee
    INNER JOIN `deliveryDriver` dd
    ON ee.employeeId = dd.employeeId
);
```

employeeId	firstName	lastName	operationZone	salary
4	Nipa	Wanich	Wangmai	20000.00
9	Saowaluk	Yamsri	Pathumwan	20000.00
14	Kamol	Sriboon	Sam Sen Nok	20000.00
19	Ratchanee	Sungthong	Wangburapa	20000.00
24	Rattana	Anantachai	Ladyao	20000.00
29	Phirun	Chakrit	Bangkrasor	20000.00
34	Chaisiri	Chirawat	Sono Loy	20000.00
39	Yuwadee	Niwat	Klong nueng	20000.00
44	Chanchai	Tanasuk	Klong song	20000.00
49	Siriporn	Jindaporn	Buengsanan	20000.00

Figure 56: The result of the query in Branch operation process

15. Report: Find drivers along with their number of deliveries in a given operation zone.

Objective: It can be used in zone optimization to ensure that the delivery load is evenly distributed across drivers in each zone to maintain service quality.

```
SELECT
    e.employeeId,
    e.firstName,
    e.lastName,
    d.licensePlate,
    COUNT(hd.deliveryInformationId) AS totalDeliveries,
    o.operationZone
FROM `employee` e
INNER JOIN `deliveryDriver` d
ON e.employeeId = d.employeeId
INNER JOIN `operationZone` o
ON d.employeeId = o.employeeId
INNER JOIN `homeDeliveryInformation` hd
ON d.employeeId = hd.employeeId
GROUP BY e.employeeId, o.operationZone, d.licensePlate
ORDER BY totalDeliveries DESC;
```

employeeId	firstName	lastName	licensePlate	totalDeliveries	operationZone
29	Phirun	Chakrit	ทม1234 สุพรรณบุรี	2	Bangkrasor
34	Chaisiri	Chirawat	นย9012 อยุธยา	2	Sono Loy
39	Yuwadee	Niwat	ปจ7890 สงขลา	2	Klong nueng
44	Chanchai	Tanasuk	พธ5678 อุบลราชธานี	2	Klong song
49	Siriporn	Jindaporn	มป3456 บุรีรัมย์	2	Buengsanan
4	Nipa	Wanich	กข1234 กรุงเทพมหานคร	1	Wangmai
9	Saowaluk	Yamsri	จฉ9012 นครราชสีมา	1	Pathumwan
15	Patcharaporn	Nontaket	ชอ1234 ขอนแก่น	1	Sam Sen Nai
15	Patcharaporn	Nontaket	ชอ1234 ขอนแก่น	1	Sam Sen Nok
20	Suchada	Sutham	ณท9012 นครปฐม	1	Wangburapa
24	Rattana	Anantachai	ดบ3456 ปทุมธานี	1	Ladyao

Figure 57: The result of the query in Branch operation process

16. **Report:** Lists all the employees who were associated with deliveries on a given date (2024-11-05), including their names and branch contact details as well as their department.

    Objective: In case of any issues that may arise from that date, we can get the associated employees to question and ask information from.

```
SELECT
    e.employeeId,
    e.firstName,
    e.lastName,
    b.name AS branchName,
    b.phoneNumber,
    d.deliveredDate,
    e.department
FROM `employee` e
INNER JOIN `branch` b
ON e.branchId = b.branchId
INNER JOIN `pickupDeliveryInformation` p
ON b.branchId = p.branchId
INNER JOIN `deliveryInformation` d
ON p.deliveryInformationId = d.deliveryInformationId
WHERE d.deliveredDate = '2024-11-5'
ORDER BY e.firstName ASC;
```

employeeId	firstName	lastName	branchName	phoneNumber	deliveredDate	department
56	Chanin	Rattanapong	Pizza Com Meng-Jai	0614120318	2024-11-05	Cooking
60	Jiratchaya	Sangthong	Pizza Com Big C Ladprao	029837398	2024-11-05	Cooking
14	Kamol	Sriboon	Pizza Com Meng-Jai	0614120318	2024-11-05	Delivery
57	Kittipong	Wongthong	Pizza Com Ratchabophit	0655076352	2024-11-05	Cashier
11	Manas	Phutthachot	Pizza Com Meng-Jai	0614120318	2024-11-05	Management
18	Narong	Tosapon	Pizza Com Ratchabophit	0655076352	2024-11-05	Cashier
55	Nathawat	Srisai	Pizza Com Meng-Jai	0614120318	2024-11-05	Cashier
15	Patcharaporn	Nontaket	Pizza Com Meng-Jai	0614120318	2024-11-05	Delivery
58	Pimnara	Mongkol	Pizza Com Ratchabophit	0655076352	2024-11-05	Cooking
16	Pornchai	Thammasak	Pizza Com Ratchabophit	0655076352	2024-11-05	Management
19	Ratchanee	Sungthong	Pizza Com Ratchabophit	0655076352	2024-11-05	Delivery
24	Rattana	Anantachai	Pizza Com Big C Ladprao	029837398	2024-11-05	Delivery
21	Rungnapa	Chaisarn	Pizza Com Big C Ladprao	029837398	2024-11-05	Management
23	Somchai	Kittipong	Pizza Com Big C Ladprao	029837398	2024-11-05	Cashier
20	Suchada	Sutham	Pizza Com Ratchabophit	0655076352	2024-11-05	Delivery
17	Supaporn	Yokchai	Pizza Com Ratchabophit	0655076352	2024-11-05	Cooking
59	Sutthira	Sukee	Pizza Com Big C Ladprao	029837398	2024-11-05	Cashier
25	Udom	Jaiklang	Pizza Com Big C Ladprao	029837398	2024-11-05	Delivery
13	Weerachai	Phrompakdee	Pizza Com Meng-Jai	0614120318	2024-11-05	Cashier
12	Wiroj	Suchat	Pizza Com Meng-Jai	0614120318	2024-11-05	Cooking
22	Yuthana	Namwong	Pizza Com Big C Ladprao	029837398	2024-11-05	Cooking

Figure 58: The result of the query in Branch operation process

17. **Report:** shows all branches and their total employee counts.
    Objective: To ensure that each branch is appropriately staffed according to its operation needs.

```
SELECT
    b.branchId as `ID`,
    b.name `Branch Name`,
    COUNT(*) as `Total Number of Employees`
FROM `branch` b
INNER JOIN `employee` e
ON b.branchID = e.branchID
GROUP BY b.branchId;
```

ID	Branch Name	Total Number of Employees
1	Pizza Com Lotus Chareonpol Tesco Lotus Rama 1	7
2	Pizza Com Samyan Mitrtown	7
3	Pizza Com Meng-Jai	7
4	Pizza Com Ratchabophit	7
5	Pizza Com Big C Ladprao	7
6	Pizza Com Manor Sanambin-Nam	7
7	Pizza Com PTT Sribuathong	7
8	Pizza Com Thummasart Rangsit	7
9	Pizza Com Klongluang	7
10	Pizza Com Rangsit Klong 10	7

Figure 59: The result of the query in Branch operation process

18. **Report:** This query calculates the income for all branches.

    Objective: It can be used to track the performance of each branch and profitability as well as compare their performance with one another, which might also lead to decisions like expansion or closure of each of the branches.

```
SELECT
    b.branchId AS `Branch ID`,
    b.name AS `Branch Name`,
    SUM(p.paidAmount) AS `Total Income`
FROM `branch` b
INNER JOIN `order` o
ON b.branchId = o.branchID
INNER JOIN `payment` p
ON o.orderId = p.orderId
GROUP BY b.branchId;
```

Branch ID	Branch Name	Total Income
1	Pizza Com Lotus Chareonpol Tesco Lotus Rama 1	4488.40
2	Pizza Com Samyan Mitrtown	6268.20
3	Pizza Com Meng-Jai	4283.40
4	Pizza Com Ratchabophit	4441.50
5	Pizza Com Big C Ladprao	3841.15
6	Pizza Com Manor Sanambin-Nam	5735.60
7	Pizza Com PTT Sribuathong	5004.40
8	Pizza Com Thummasart Rangsit	4517.00
9	Pizza Com Klongluang	4112.30
10	Pizza Com Rangsit Klong 10	5143.00

Figure 60: The result of the query in Branch operation process

19. **Report:** This query gives the number of male/female employees at a branch.

    Objective: This can be used to promote gender diversity at each branch and ensure a balanced and diverse workforce. It may also be used to ensure adherence to diversity and inclusion policies within the company.

```
SELECT
    b.branchId AS ID,
    b.name AS Branch,
    (
        CASE
            WHEN e.gender = 'M'
            THEN 'Male'
            ELSE 'Female'
        END
    ) AS Gender,
    COUNT(e.gender) AS `Gender Count`
FROM `branch` b
INNER JOIN `employee` e
ON e.branchId = b.branchId
GROUP BY b.branchId, e.gender;
```

ID	Branch	Gender	Gender Count
1	Pizza Com Lotus Chareonpol Tesco Lotus Rama 1	Male	4
1	Pizza Com Lotus Chareonpol Tesco Lotus Rama 1	Female	3
2	Pizza Com Samyan Mitrtown	Male	4
2	Pizza Com Samyan Mitrtown	Female	3
3	Pizza Com Meng-Jai	Male	5
3	Pizza Com Meng-Jai	Female	2
4	Pizza Com Ratchabophit	Male	3
4	Pizza Com Ratchabophit	Female	4
5	Pizza Com Big C Ladprao	Female	3
5	Pizza Com Big C Ladprao	Male	4
6	Pizza Com Manor Sanambin-Nam	Female	3
6	Pizza Com Manor Sanambin-Nam	Male	4
7	Pizza Com PTT Sribuathong	Female	3
7	Pizza Com PTT Sribuathong	Male	4
8	Pizza Com Thummasart Rangsit	Male	3
8	Pizza Com Thummasart Rangsit	Female	4
9	Pizza Com Klongluang	Male	3
9	Pizza Com Klongluang	Female	4
10	Pizza Com Rangsit Klong 10	Female	3
10	Pizza Com Rangsit Klong 10	Male	4

Figure 61: The result of the query in Branch operation process

20. **Report:** This query gives the male/female ratio of all the employees.

    Objective: This can be used to ensure compliance to policies of gender equality set by both the company and maybe the governments as well as providing data for reports. It can also be used to support and drive recruitment and staffing policies in the future.

```
SELECT
    (
        CASE
            WHEN e.gender = 'M'
            THEN 'Male'
            ELSE 'Female'
        END
    ) AS Employees,
    COUNT(gender) as `Count`
FROM `employee` e
GROUP BY e.gender;
```

Employees	Count
Female	32
Male	38

Figure 62: The result of the query in Branch operation process

21. **Report:** This query calculate the average number of orders in a Samyan Mitrtown.

    Objective: It can be to identify the trends in order to optimize the delivery process as well as workflow and resources at branches around the specific areas (Samyan Mitrtown here).

```
SELECT
    b.branchId,
    b.name AS branchName,
    ROUND(AVG(orderCount), 0) AS avgOrderCount
FROM `branch` b
INNER JOIN `order` o
ON b.branchId = o.branchId
INNER JOIN (
    SELECT
        branchId,
        COUNT(orderId) AS orderCount
    FROM `order`
    GROUP BY branchId
) AS `branchOrderCounts`
ON b.branchId = 2
GROUP BY b.branchId, b.name;
```

branchId	branchName	avgOrderCount
2	Pizza Com Samyan Mitrtown	3

Figure 63: The result of the query in Branch operation process

22. **Report:** Show the list of pizzas that have never been ordered in a specific branch.

    Objective: This can be used for menu optimization to better understand the customer preferences as well as keep a well-informed menu, so more popular ones are promoted while unpopular ones may be identified and removed.

```
SELECT *
FROM `product` p
WHERE
    p.type = 'pizza' AND
    NOT EXISTS (
        SELECT 1
        FROM orderItem oi
        INNER JOIN `order` o
        ON oi.orderId = o.orderId
        WHERE
            o.branchID = 9 AND
            oi.productId = p.productId
    );
```

productId	name	type	price
1	doble cheese	pizza	419.00
2	doble pepperoni	pizza	419.00
11	hawaiian delight	pizza	419.00
12	seafood supreme	pizza	459.00
13	mushroom melt	pizza	399.00
22	bbq chicken pizza	pizza	429.00
32	pepperoni lovers	pizza	419.00
33	margherita classic	pizza	389.00

Figure 64: The result of the query in Branch operation process

23. Report: List Employees and Their Total Working Hours per day.

Objective: This can be used to track daily work hours to ensure proper staffing and avoid overworking employees. It can also be used to calculate accurate working hours for salary and overtime calculations.

```
SELECT
    e.employeeId,
    e.firstName,
    e.lastName,
    COALESCE(SUM(
        CASE
            WHEN s.shift = 'morning' THEN 8
            WHEN s.shift = 'evening' THEN 8
            WHEN s.shift = 'night' THEN 8
            ELSE 0
        END
    ), 0) AS totalScheduledHours
FROM `employee` e
LEFT JOIN `shift` s
ON e.employeeId = s.employeeId
GROUP BY e.employeeId, e.firstName, e.lastName;
```

employeeId	firstName	lastName	totalScheduledHours
1	Anan	Phongchai	16
2	Thanakorn	Srisuk	16
3	Sirin	Kittithorn	16
4	Nipa	Wanich	16
5	Nattapong	Boonmee	8
6	Kanok	Phongpan	16
7	Somsak	Thammasak	16
8	Preecha	Wiriyaporn	16
9	Saowaluk	Yamsri	16
10	Thida	Sukjai	8
11	Manas	Phutthachot	16
12	Wiroj	Suchat	16
13	Weerachai	Phrompakdee	16
14	Kamol	Sriboon	16
15	Patchara...	Nontaket	8
16	Pornchai	Thammasak	16
17	Supaporn	Yokchai	16
18	Narong	Tosapon	16
19	Ratchanee	Sungthong	16
20	Suchada	Sutham	8
21	Rungnapa	Chaisarn	16
22	Yuthana	Namwong	16
23	Somchai	Kittipong	16
24	Rattana	Anantachai	16
25	Udom	Jaiklang	8
26	Chompoo	Rattanasuk	16
27	Wanchai	Prasert	16

Figure 65: The result of the query in Branch operation process

24. **Report:** Show the total sales and number of transactions per day across all branches.

    Objective: It can be used for providing data for budgeting and identifying trends in sales over time. And to track daily sales and transaction volume to better adapt to catering to these more effectively and efficiently.

```
SELECT
    DATE(orderDate) AS order_date,
    COUNT(o.orderId) AS total_transactions,
    SUM(p.paidAmount) AS total_sales
FROM `order` o
JOIN payment p ON o.orderId = p.orderId
GROUP BY DATE(orderDate);
```

order_date	total_transactions	total_sales
2024-11-01	3	3926.00
2024-11-02	3	5811.15
2024-11-03	3	3963.70
2024-11-04	3	4475.90
2024-11-05	3	3299.20
2024-11-06	3	5813.00
2024-11-07	12	20546.00

Figure 66: The result of the query in Branch operation process

Delivery

1. **Report:** This query counts the amount of deliveries handled by each delivery driver.

   Objective: It can be used to identify drivers with high delivery volumes for possible rewards or additional support. And to ensure that deliveries are evenly distributed among drivers to avoid burnout and improve service quality.

```
SELECT
    e.firstName,
    e.lastName,
    COUNT(hdi.deliveryInformationId) AS deliveries_handled
FROM `employee` e
INNER JOIN `deliveryDriver` dd
ON e.employeeId = dd.employeeId
LEFT JOIN `homeDeliveryInformation` hdi
ON dd.employeeId = hdi.employeeId
GROUP BY e.employeeId
ORDER BY e.firstName ASC, e.lastName ASC;
```

firstName	lastName	deliveries_handled
Chaisiri	Chirawat	2
Chanchai	Tanasuk	2
Kamol	Sriboon	0
Nattapong	Boonmee	0
Nipa	Wanich	1
Noknoi	Pattana	0
Pakorn	Sukwattana	0
Patcharaporn	Nontaket	1
Phairoj	Thongdee	0
Pharanee	Jirasri	0
Phirun	Chakrit	2
Ratchanee	Sungthong	0
Rattana	Anantachai	1

Figure 67: The result of the query in Delivery process (first 13 output)

2. **Report:** This query shows delivery drivers and license plates.

   Objective: To provide a clear mapping between drivers and their vehicles, useful for tracking which drivers are using which vehicles.

```
SELECT
    dd.employeeId AS `Employee ID`,
    dd.licensePlate AS `License No.`
FROM `deliveryDriver` dd
INNER JOIN `homeDeliveryInformation` hdi
ON hdi.employeeId = dd.employeeId
GROUP BY dd.employeeId;
```

Employee ID	License No.
4	กข1234 กรุงเทพมหานคร
9	จฉ9012 นครราชสีมา
15	ซอ1234 ขอนแก่น
20	ณฑ9012 นครปฐม
24	ดบ3456 ปทุมธานี
29	ทม1234 สุพรรณบุรี
34	นย9012 อยุธยา
39	ปจ7890 สงขลา
44	พธ5678 อุบลราชธานี
49	มป3456 บุรีรัมย์

Figure 68: The result of the query in Delivery process

3. **Report:** This query shows all of the orders completed by delivery drivers along with their location details.

   Objective: To monitor driver assignments and address details for operational management.

```
SELECT
    dd.employeeId AS `Employee ID`,
    dd.licensePlate AS `License No.`,
    sda.houseNumber AS `House Number`,
    sda.buildingNumber AS `Building Number`,
    sda.addressLabel AS `Additional Details`
FROM `deliveryDriver` dd
INNER JOIN `homeDeliveryInformation` hdi
ON hdi.employeeId = dd.employeeId
INNER JOIN `savedDeliveryAddress` sda
ON hdi.savedDeliveryAddressId = sda.savedDeliveryAddressId;
```

Employee ID	License No.	House Number	Building Number	Additional Details
29	ทม1234 สุพรรณบุรี	827	NULL	Wangmai, Pathumwan, Bangkok 10330
34	นย9012 อยุธยา	940/1	B	Sam Sen Nok, Huai Khwang, Bangkok 10310
39	ปจ7890 สงขลา	141	NULL	Wangburapa, PraNaKorn, Bangkok 10200
44	พธ5678 อุบลราชธานี	149	C	Wangburapa, PraNaKorn, Bangkok 10200
49	มป3456 บุรีรัมย์	670	NULL	Ladyao, Jatujak, Bangkok 10900
4	กข1234 กรุงเทพมหานคร	104	NULL	Bangkrasor, Muang, Nonthaburi 11000
9	จฉ9012 นครราชสีมา	90	NULL	Klongluang, Pathumthani 12120
15	ซอ1234 ขอนแก่น	96	D	Klong 1, Klongluang, Pathumthani 12120
20	ฌฑ9012 นครปฐม	139	NULL	Wangburapa, PraNaKorn, Bangkok 10200
24	ดบ3456 ปทุมธานี	291/1	E	Wangmai, Pathumwan, Bangkok 10330
29	ทม1234 สุพรรณบุรี	393/9	F	Sam Sen Nok, Huai Khwang, Bangkok 10310
34	นย9012 อยุธยา	510	NULL	Sam Sen Nok, Huai Khwang, Bangkok
39	ปจ7890 สงขลา	345	G	Wangmai, Pathumwan, Bangkok 10330
44	พธ5678 อุบลราชธานี	123	H	Bangkrasor, Muang, Nonthaburi 11000
49	มป3456 บุรีรัมย์	829	NULL	Ladyao, Jatujak, Bangkok 10900

Figure 69: The result of the query in Delivery process

4. **Report:** Find the total amount of night shift deliveries together with the driver name and the operation zone.

   Objective: It can be used to assess night shift delivery performance by operation zone to ensure timely service during off-hours, and ensure that workload distribution is balanced among zones and drivers.

```sql
SELECT
    e.employeeId,
    e.firstName AS driverFirstName,
    e.lastName AS driverLastName,
    o.operationZone,
    COUNT(hd.deliveryInformationId) AS lateDeliveries
FROM `employee` e
INNER JOIN `deliveryDriver` d
ON e.employeeId = d.employeeId
INNER JOIN `operationZone` o
ON d.employeeId = o.employeeId
INNER JOIN `homeDeliveryInformation` hd
ON d.employeeId = hd.employeeId
WHERE
    hd.deliveryTime >= '20:00:00' OR
    hd.deliveryTime BETWEEN '00:00:00' AND '08:00:00'
GROUP BY e.employeeId, e.firstName, e.lastName, o.operationZone
HAVING lateDeliveries > 0
ORDER BY lateDeliveries DESC;
```

employeeId	driverFirstName	driverLastName	operationZone	Night Delivery
15	Patcharaporn	Nontaket	Sam Sen Nai	1
15	Patcharaporn	Nontaket	Sam Sen Nok	1
20	Suchada	Sutham	Wangburapa	1

Figure 70: The result of the query in Delivery process

5. **Report:** Show the faster delivery driver delivering time in average and the average of delivery time.

   Objective: To evaluate the best performance of the delivery driver since the order needs to be delivered within 30 minutes after the payment.

```
SELECT
      CONCAT(sub.firstname," ",sub.lastname) AS Fullname,
    AVG(sub.time_diff_minutes) AS `Average delivery time in minutes`
FROM (
    SELECT
        d.employeeId,
        e.firstName,
        e.lastName,
        DATE_FORMAT(hd.deliveryTime, '%H:%i:%s') AS delivery_time,
        DATE_FORMAT(p.paymentDate, '%H:%i:%s') AS payment_time,
        ROUND((TIME_TO_SEC(hd.deliveryTime) -
TIME_TO_SEC(DATE_FORMAT(p.paymentDate, '%H:%i:%s'))) / 60) AS
time_diff_minutes
    FROM `deliveryDriver` d
    INNER JOIN `employee` e
    ON d.employeeId = e.employeeId
    INNER JOIN `homeDeliveryInformation` hd
    ON d.employeeId = hd.employeeId
    INNER JOIN `deliveryInformation` di
    ON di.deliveryInformationId = hd.deliveryInformationId
    INNER JOIN `order` o
    ON di.deliveryInformationId = o.deliveryInformationId
    INNER JOIN `payment` p
    ON o.orderId = p.orderId
) AS `sub`
GROUP BY sub.employeeId
ORDER BY `Average delivery time in minutes` ASC
LIMIT 1;
```

Fullname	Average delivery time in minutes
Nipa Wanich	11.0000

Figure 71: The result of the query in Delivery process

Coupon and Discount

1. **Report:** Find how many coupons are used by each user.

   Objective: To analyze the effectiveness of the existing coupons by the total amount of each user in order to improve and revise the strategy of designing promotions.

```
SELECT
    CONCAT(firstname," ",lastname) AS Fullname,
    COUNT(c.couponId) AS `Total Coupon`
FROM
    coupon c
JOIN `order` o ON c.orderId = o.orderId
JOIN user u ON o.userId = u.userId
GROUP BY
    u.userId, u.firstName, u.lastName
ORDER BY
    `Total Coupon` DESC;
```

Fullname	Total Coupon
Natchapol Lebkrut	3
Vayuphak Saengthong	2
Napat Decha	2
Napat Tatiyakaroonwong	2
Chatdanai Wongsuwan	2
Haseeb Ali	1
Somsri Hiran	1
Yingrat Rattana	1
Itthirath Jeamanukul	1
Pikul Watdao	1

Figure 72: The result of the query in Coupon and Discount process

2. **Report:** This code finds all orders on which an unused coupon can be applied as well as what product makes it eligible for that.

   Objective: To find products with coupons that have not yet been applied, helping them monitor coupon usage.

```
SELECT
    o.orderId AS `Order`,
    oi.productId AS `ProductID`,
    ac.couponId AS `ApplicableCoupon`
FROM `order` o
LEFT JOIN `orderItem` oi
ON oi.orderId = o.orderId
LEFT JOIN `applicable_coupon` ac
ON ac.productId = oi.productId
LEFT JOIN `coupon` unused_c
ON ac.couponId = unused_c.couponId
-- Only include applicable coupons that have not been applied
WHERE
    unused_c.orderId IS NULL AND
    ac.couponId IS NOT NULL
ORDER BY o.orderId ASC;
```

Order	ProductID	ApplicableCoupon
6	1	17
8	2	17
20	1	17
26	2	17

Figure 73: The result of the query in Coupon and Discount process

3. **Report:** This code shows all order with information on what coupon has been applied to which product, along with the coupon names.

   Objective: Show all orders with detailed information about the coupons to each product, tracking coupon usage on specific products.

```
SELECT
    o.orderId AS `Order`,
    oi.productId AS `ProductID`,
    c.couponId AS `CouponApplied`,
    c.name AS `AppliedCouponName`
FROM `order` o
INNER JOIN `coupon` c
ON c.orderId = o.orderId
INNER JOIN `orderItem` oi
ON oi.orderId = o.orderId;
```

Order	ProductID	CouponApplied	AppliedCouponName
1	5	1	Discount10
1	15	1	Discount10
1	22	1	Discount10
2	9	2	Holiday20
2	13	2	Holiday20
2	27	2	Holiday20
3	7	3	Summer15
3	18	3	Summer15
3	3	3	Summer15
4	12	4	Winter25
4	19	4	Winter25
4	25	4	Winter25
5	30	5	Promo5
5	10	5	Promo5
5	6	5	Promo5
6	16	6	Festive30
6	24	6	Festive30
6	1	6	Festive30
7	11	7	BlackFriday40
7	8	7	BlackFriday40
7	20	7	BlackFriday40
8	28	8	NewYear50
8	4	8	NewYear50
8	2	8	NewYear50
9	17	9	Easter15
9	23	9	Easter15
9	29	9	Easter15
10	21	10	Christmas25
10	14	10	Christmas25
10	26	10	Christmas25
11	18	11	FlashSale30
11	27	11	FlashSale30
11	13	11	FlashSale30
12	25	12	VIPCustomer20
12	12	12	VIPCustomer20
12	5	12	VIPCustomer20
13	30	13	Clearance10

Figure 74: The result of the query in Coupon and Discount process

6. References

[1] *The Pizza Company 1112 Online Ordering Delivery Takeaway*, https://www.1112.com/en