Table of Contents

Introduction to Basket4U Online Grocery Store

Basket4U is an Online Grocery Store based in Yangon, Myanmar. Basket4U was established for the convenience of housewives to buy their daily requirements such as meat, vegetables and other essentials without necessary to go outside and shop through wet markets. They specialize in selling products from wet markets, supermarkets and delivering them free to customers' homes in Yangon.

Task 1

Description of Business

1 Task 1: Description of Business

1.1 Scenario

When Basket4U was initially founded in Yangon, Myanmar, customers were able to shop Products through facebook page, order from messenger or viber. During Covid-19 pandemic, their free delivery services were so beneficial for the customers in Yangon who do not want to go shopping through markets to avoid infection. As Basket4U becomes successful, they want to extend their business as Website and application for the convenience of customers and better customer-order handling. They want a digital online database to help manage their records related with purchases, products, orders and payments of their customers. This database will mainly focus on purchasing products and Ordering of customers. It will not explore details into the delivery and management processes.

Basket4U will have one and more than one Suppliers agreed to sell their products to Basket4U. Record related to Purchases from Suppliers, total products that they purchased from suppliers and unit purchase prices, etc…will be recorded.

Products they purchased will be classified into multiple Categories and each categories will have products that Basket4U currently owns to sell to customers. Products are defined by multiple Categories such as: Meat & Seafood, Fruits & Vegetables, Dairy, Breakfast, Ready to Eat, Snacks & Beverages, Flowers, etc.

A customer will be able to make one to more than one orders. Each orders made will be related to the only customer that has confirmed each orders. Each orders made will have at least one to more than one products bought.

Basket4U holds Promotion events where Customers can use different Promotion Codes. These codes will discount on the total amount of customers' Orders according to the Promotion Events descriptions.

The delivery will send at least one and more than one Orders to Customer's desired Delivery Addresses. Then, Customers will make specific payments for their each of their specific Orders. Customers can pay for each orders through different types of Payment. Basket4U accepts different types of payments: Cash on Delivery, Mobile banking payment and Credit/Debit card payment. Both order and payment status will be tracked.

As Basket4U's specialization is delivering customers' groceries to their homes in Yangon with free delivery, there will be no delivery charges in customers' payments.

Examples of data from Basket4U are shown as following:

1.1.1 Document 1: Purchase products from Suppliers

Purchase Record 1

Purchase ID: Pur-001

Purchase Date: 5-6-2022

Supplier ID: S-001

Supplier Name: Shwe Si Daw

Contact Phone: 09777777777

Description: Shwe Si Daw sells wet market products such as meats, vegetables, fruits.

Supplier Address: Tarmwe, Yangon.

Product ID	Product Name	Product Weight	Unit Price(ks)	Expiry Date	Purchase Line Quantity (Pcs)	Purchase Line Amount
P-001	Chicken Breast	0.3 Viss	4,000		30	120,000
P-002	Burmese Whole Chicken	0.5 Viss	6,000		25	150,000
P-003	Pork 3 layers	300 g	3,200		25	80,000
P-004	Chicken Thigh with skin	0.5 Viss	2,500		30	75,000
P-008	Pork Head	0.5 Viss	6,000		10	60,000
P-013	Cabbage 1 pack		900		20	18,000
P-014	Banana		2,200		20	44,000

	1 pack					
				Total Purchase Quantity	160	
					Total Purchase Amount	547,000

Purchase Record 2

Purchase ID: Pur-002

Purchase Date: 6-6-2022

Supplier ID: S-002

Supplier Name: Uncle Lay

Contact Phone: 09666666666

Description: Uncle Lay sells Snacks and Beverages.

Supplier Address: Yankin, Yangon.

Product ID	Product Name	Product Weight	Unit Price(ks)	Expiry Date	Purchase Line Quantity (Pcs)	Purchase Line Amount
P-005	Danisa Cookies	454 g	11,000	15-6-2024	40	440,000
P-006	Juicy Orange	750 ml	3,000	30-12-2024	30	90,000
P-007	Ovaltine Chocolate Bottle	400 g	9,000	30-12-2024	25	225,000
				Total Purchase Quantity	95	
					Total Purchase Amount	755,000

Purchase Record 3

Purchase ID: Pur-003

Purchase Date: 7-6-2022

Supplier ID: S-003

Supplier Name: Top Meat

Contact Phone: 09343434343

Description: Top Meat sells various types of meats and seafood.

Supplier Address: Ahlone, Yangon.

Product ID	Product Name	Product Weight	Unit Price(ks)	Expiry Date	Purchase Line Quantity (Pcs)	Purchase Line Amount
P-009	Pork Tenderloin	0.5 Viss	8,000		20	160,000
P-010	Pork Ribs Special	0.5 Viss	9,500		30	285,000
P-011	Chicken Breast	0.3 Viss	4,500		20	90,000
P-012	Burmese Whole Chicken	0.5 Viss	6,200		20	124,000
				Total Purchase Quantity	90	
					Total Purchase Amount	659,000

1.1.2 Document 2: Products classified into Categories

Category ID	Category Name	Category Description	Product ID	Product Name
Cat-001	Meats and Seafood	Includes chicken, pork, beef, seafood, etc…	P-001	Chicken Breast
Cat-001	Meats and Seafood	Includes chicken, pork, beef, seafood, etc…	P-002	Burmese Whole Chicken
Cat-001	Meats and Seafood	Includes chicken, pork, beef, seafood, etc…	P-003	Pork 3 layers
Cat-001	Meats and Seafood	Includes chicken, pork, beef, seafood, etc…	P-004	Chicken Thigh with skin
Cat-001	Meats and Seafood	Includes chicken, pork, beef, seafood, etc…	P-008	Pork Head
Cat-001	Meats and Seafood	Includes chicken, pork, beef, seafood, etc…	P-009	Pork Tenderloin
Cat-001	Meats and Seafood	Includes chicken, pork, beef, seafood, etc…	P-010	Pork Ribs Special
Cat-001	Meats and Seafood	Includes chicken, pork, beef, seafood, etc…	P-011	Chicken Breast
Cat-001	Meats and Seafood	Includes chicken, pork, beef, seafood, etc…	P-012	Burmese Whole Chicken

Cat-002	Fruits and Vegetables	Includes fruits and vegetables such as grapes, apples, oranges, potato, cabbages, gourd, radish etc…	P-013	Cabbage 1 pack
Cat-002	Fruits and Vegetables	Includes fruits and vegetables such as grapes, apples, oranges, potato, cabbages, gourd, radish etc…	P-014	Banana 1 pack
Cat-002	Fruits and Vegetables	Includes fruits and vegetables such as grapes, apples, oranges, potato, cabbages, gourd, radish etc…	P-015	Grapes
Cat-003	Snacks and Beverages	Includes snacks such as biscuits, crackers, noodles and beverages such as orange juices, coca cola, etc…	P-005	Danisa Cookies
Cat-003	Snacks and Beverages	Includes snacks such as biscuits, crackers, noodles and beverages such as orange juices, coca cola, etc…	P-006	Juicy Orange Bottle
Cat-003	Snacks and Beverages	Includes snacks such as biscuits, crackers, noodles and beverages such as orange juices, coca cola, etc…	P-007	Ovaltine Chocolate Bottle

1.1.3 Document 3: Customers Ordering Products

Order Record 1

Order ID: Ord-001

Order Date: 12-6-2022

Customer ID: Cus-001

Customer Name: Maung Maung

Customer Email: MaungMaung2@gmail.com

Customer Phone: 09454545454

Customer Address: No.19, Room 301, 123th street, MingalarTaungNyunt Tsp, Yangon.

Delivery Address: No.19, Room 301, 123th street, MingalarTaungNyunt Tsp, Yangon.

Product ID	Product Name	Weight	Unit Price	Order Line Quantity(Pcs)	Order Line Amount
P-001	Chicken Breast	0.3 Viss	4,000	2	8,000
P-015	Grapes	0.3 Viss	6,000	3	18,000
P-007	Ovaltine Chocolate Bottle	400 g	9,000	1	9,000
P-013	Cabbage 1 pack		900	1	900
P-009	Pork Tenderloin	0.5 Viss	8,000	2	16,000
			Total Order Quantity	9	
				Sub Total	51,900

	Promotion Discount Amount (B4Uoff10%)	-5,190
	Tax (5%)	2,595
	Total Order Amount	49,305

Employee ID: E-001
Employee Name: Ma Mya Mya
Delivery ID: Deli-001
Delivery Date: 13-6-2022
Order Status: Delivered
Payment Status: Paid
Payment Amount: 49,305
Payment Type: CB Bank

Order Record 2

Order ID: Ord-002

Order Date: 12-6-2022

Customer ID: Cus-001

Customer Name: Maung Maung

Customer Email: MaungMaung2@gmail.com

Customer Phone: 09454545454

Customer Address: No.19, Room 301, 123th street, MingalarTaungNyunt Tsp, Yangon.

Delivery Address: No.15, Room 101, Bayint Naung Road, 44th Ward, North Dagon Tsp, Yangon.

Product ID	Product Name	Weight	Unit Price	Order Line Quantity(Pcs)	Order Line Amount
P-011	Chicken Breast	0.3 Viss	4,500	3	13,500
			Total Order Quantity	3	
				Sub Total	13,500
				Promotion Discount Amount	0
				Tax (5%)	675
				Total Order Amount	14,175

Employee ID: E-001

Employee Name: Ma Mya Mya

Delivery ID: Deli-002

Delivery Date: 13-6-2022

Order Status: Delivered

Payment Status: Paid Payment Amount: 14,175 Payment Type: KBZ Pay

Order Record 3

Order ID: Ord-003

Order Date: 14-6-2022

Customer ID: Cus-002

Customer Name: Kaung Khant

Customer Email: KaungKhant2kk@gmail.com

Customer Phone: 09456712345

Customer Address: No.114, Room 201, 135th street, MingalarTaungNyunt Tsp, Yangon.

Delivery Address: No.114, Room 201, 135th street, MingalarTaungNyunt Tsp, Yangon.

Product ID	Product Name	Weight	Unit Price	Order Line Quantity(Pcs)	Order Line Amount
P-006	Banana 1 pack		2,200	2	4,400
P-015	Grapes	0.3 Viss	6,000	3	18,000
P-007	Juicy Orange	750 ml	3,000	1	3,000
P-013	Pork Ribs Special	0.5 Viss	9,500	1	9,500
P-009	Pork Tenderloin	0.5 Viss	8,000	1	8,000
			Total Order Quantity	8	
				Sub Total	42,900

	Promotion Discount Amount (B4Uoff10%)	-4,290
	Tax (5%)	2145
	Total Order Amount	40,755

Employee ID: E-002
Employee Name: Mg Aung Aung
Delivery ID: Deli-001
Delivery Date: 15-6-2022
Order Status: Deliverd
Payment Status: Paid
Payment Amount: 40,800
Payment Type: Cash on Delivery

1.1.4 Document 4: Applying Promotions to Orders

Promotion 1

Promotion ID: Promo-001

Promotion Code: B4Uoff15%

Promotion Type: Discount Percentage

Promotion Amount: 15%

Promotion Description: Customers whose order subtotal is 40,000 or above 40,000 kyats can use this promotion code to be benefitted with 15% discount over order subtotal amount. This code is not limited with numbers of orders

Promotion Active: No

Promotion Start Date: 13-6-2022

Promotion End Date: 18-6-2022

Promotion 2

Promotion ID: Promo-002

Promotion Code: B4Uoff10%

Promotion Type: Discount Percentage

Promotion Amount: 10%

Promotion Description: Customers whose order subtotal is 30,000 or above 30,000 kyats can use this promotion code to be benefitted with 10% discount over order subtotal amount. This code is not limited with numbers of orders

Promotion Active: Yes

Promotion Start Date: 5-6-2022

Promotion End Date: 5-7-2022

Promotion 3

Promotion ID: Promo-003

Promotion Code: B4Uoff5%

Promotion Type: Discount Percentage

Promotion Amount: 5%

Promotion Description: Customers whose order subtotal is 15,000 or above 15,000 kyats can use this promotion code to be benefitted with 5% discount over order subtotal amount. This code is not limited with numbers of orders.

Promotion Active: Yes

Promotion Start Date: 5-6-2022

Promotion End Date: 5-7-2022

Promotion codes on Orders

Order ID	Order Date	Sub Total	Promotion ID	Promotion Code	Promotion Discount Amount	Tax	Total Order Amount
Ord-001	12-6-2022	51,900	Promo-002	B4Uoff10%	5,190	2,595	49,305
Ord-002	12-6-2022	13,500				675	14,175
Ord-003	14-6-2022	42,900	Promo-002	B4Uoff10%	4,290	2,145	40,755
Ord-004	15-6-2022	16,000	Promo-003	B4Uoff5%	800	800	16,000

Ord-005	15-6-2022	20,000	Promo-003	B4Uoff5%	1,000	1,000	20,000
Ord-006	15-6-2022	31,000	Promo-002	B4Uoff10%	3,100	1,550	29,450
Ord-007	17-6-2022	10,000				500	10,500
Ord-008	20-6-2022	30,000	Promo-002	B4Uoff10%	3,000	1,500	31,500
Ord-009	20-6-2022	25,000	Promo-003	B4Uoff5%	1,250	1,250	25,000
Ord-010	20-6-2022	8,500				425	8,925

1.1.5 Document 5: Order Delivery Record

Order ID	Order Date	Order Status	Delivery Address	Delivery ID	Delivery Date
Ord-001	12-6-2022	Delivered	No.19, Room 301, 123th street, MingalarTaungNyunt Tsp, Yangon.	Deli-001	13-6-2022
Ord-002	12-6-2022	Delivered	No.15, Room 101, Bayint Naung Road, 44th Ward, North Dagon Tsp, Yangon.	Deli-002	13-6-2022
Ord-003	14-6-2022	Delivered	No.114, Room 201, 135th street, MingalarTaungNyunt Tsp, Yangon.	Deli-001	15-6-2022
Ord-004	15-6-2022	Delivered	No.64, Room 401, Thidar street, MingalarTaungNyunt Tsp, Yangon.	Deli-001	16-6-2022
Ord-005	15-6-2022	Delivered	No.64, Room 401, Thidar street, MingalarTaungNyunt Tsp, Yangon.	Deli-001	16-6-2022

Ord-006	15-6-2022	Delivered	No.103, Room 101, Bandar street, Thaketa Tsp, Yangon.	Deli-003	16-6-2022
Ord-007	17-6-2022	Delivered	No. 42, Room 201, Bo Min Yaung Road, North Dagon Tsp, Yangon.	Deli-003	18-6-2022
Ord-008	20-6-2022	On the way	No. 70, Room 301, U Pone Nya street, North Dagon Tsp, Yangon.	Deli-003	21-6-2022
Ord-009	20-6-2022	Preparing	No. 134, Room 501. Seikta Thukha street, PatheinNyunt Ward, Tarmwe Tsp, Yangon	Deli-001	23-6-2022
Ord-010	20-6-2022	Delivered	No.64, Room 401, Thidar street, MingalarTaungNyunt Tsp, Yangon.	Deli-003	21-6-2022

Task 2

ER and Data Dictionary

2 Task 2: ER and Data Dictionary

2.1 List of Entities of Basket4U

- Suppliers
- Purchases
- PurchasedProducts
- Categories
- Products
- OrderProducts
- Orders
- Promotions
- Customers
- Employees
- Delivery
- Payment

2.2 Relationship and Multiplicity of Basket4U

Entity	Multiplicity	Relationship	Multiplicity	Entity
Purchases	1…*	Purchased from	1	Suppliers
Products	1…*	Includes in	1…*	Purchases
Categories	1	Classified as	1…*	Products
Orders	1…*	Includes	1…*	Products
Customers	1	Make	1…*	Orders
Employees	1	Handles	1…*	Orders
Promotions	1	Used in	0…*	Orders
Orders	1…*	Send by	1	Delivery
Payment	1	Made for	1	Orders

2.3 Entity Relationship Diagram

Entity Relationship Diagram (or) ERD is a visual representation that shows relationships within different entities (such as persons, records of transactions and objects) in an organization or in a database. (Biscobing, n.d.) ERD is mainly drawn to start designing practically highly usable databases and offers overall views of data requirements that also matches operations of an organization. (Watt, 2014)

2.4 Entity Relationship Diagram (without Dummies) of Basket4U

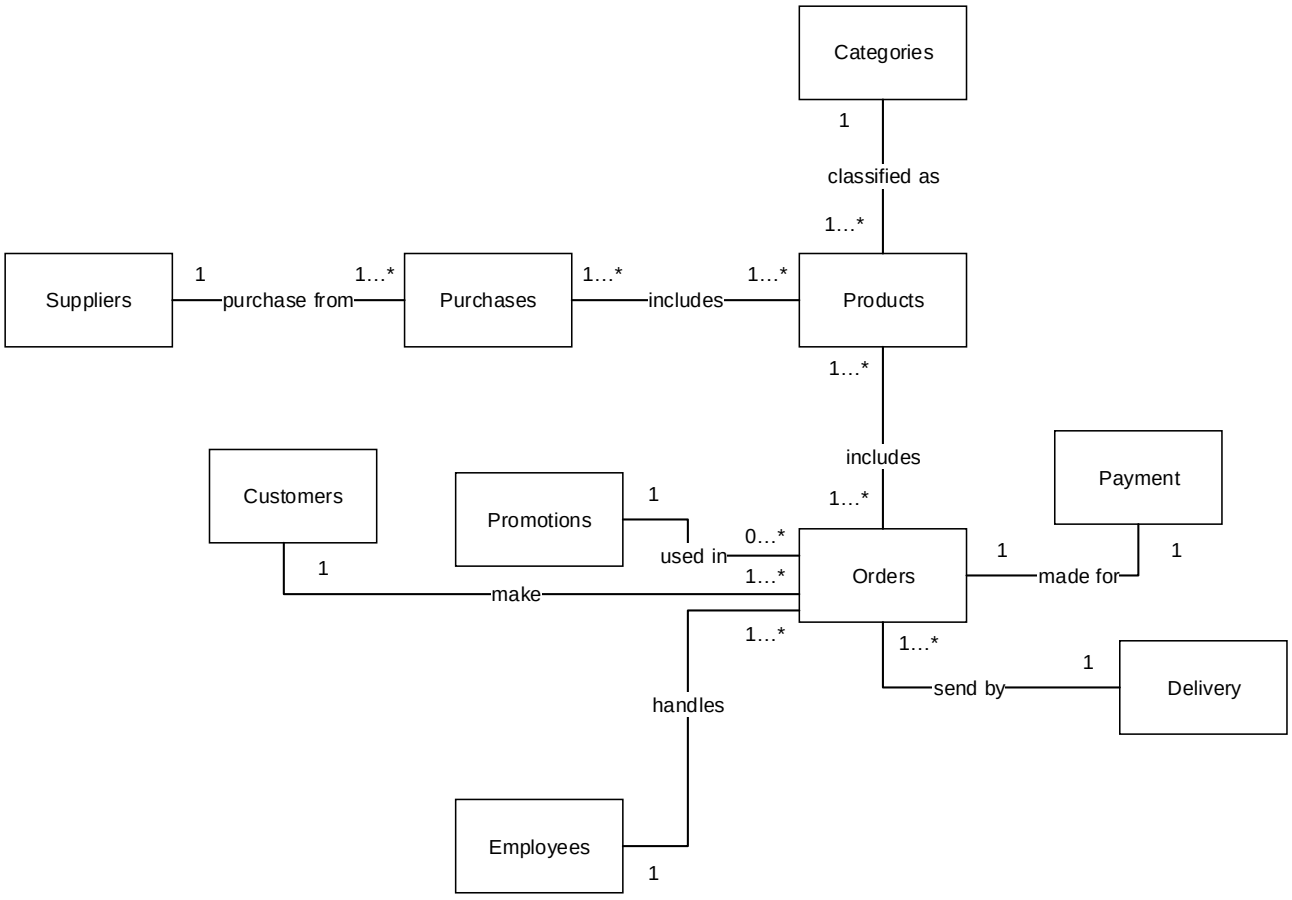


Figure 1: Basket4U ERD without Dummies

2.5 Entity Relationship Diagram (Modified, with Dummies) of Basket4U

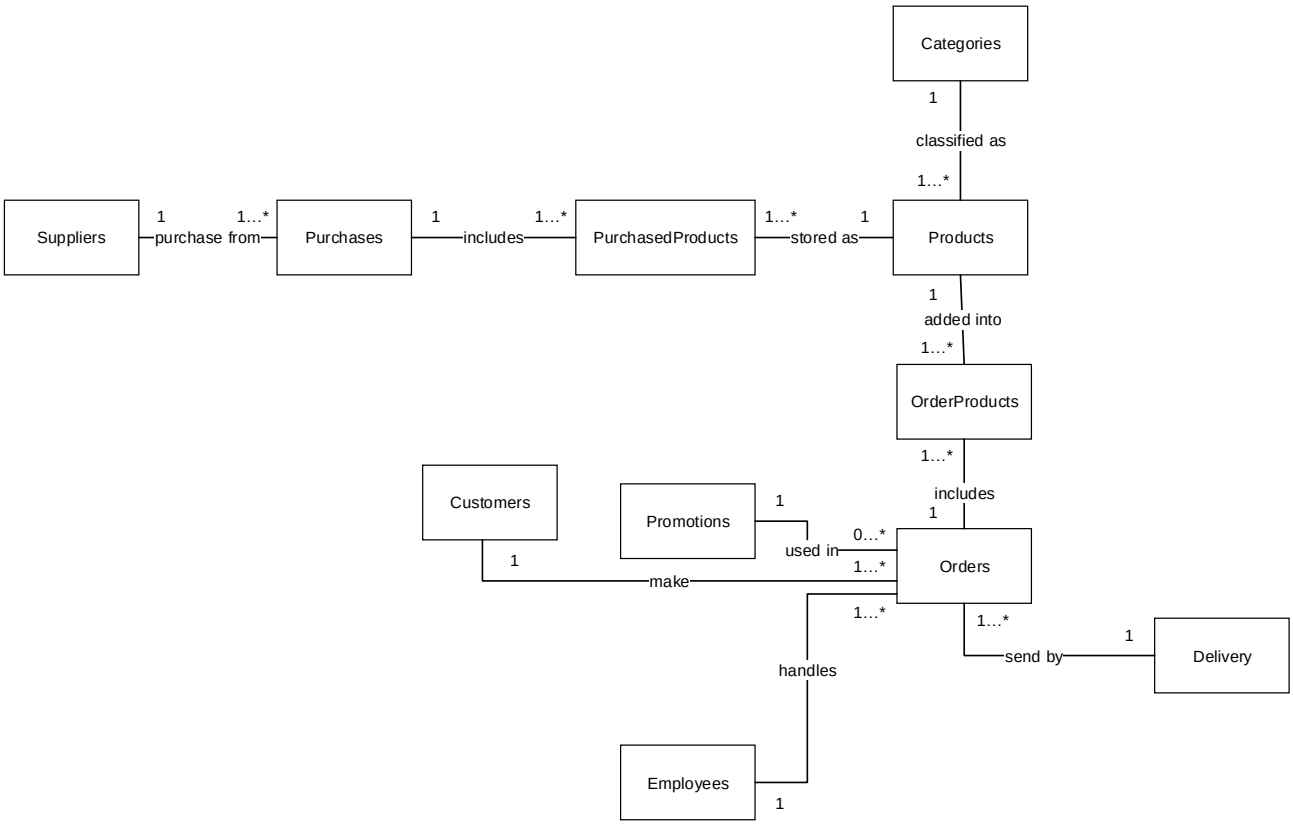


Figure 2: Basket4U ERD with Dummies

2.6 Data Dictionary and Data Dictionary of Basket4U

A data dictionary, or metadata repository also known as system catalogue is a centralized collection of data such as names, data attributes, meaning, relationships to other data, origin, usage, and data types. (Derda, 2020). It is also a main tool of the Database Administrators. For relational database, name descriptions, structure and relationships of data, indexes, data types and sizes of data types, primary keys, foreign keys and different kinds of constraints set on data are stored in Data Dictionary. Data Dictionary can be used to document data structures and information and it helps easier during data analysis, designing stages and implementation stages. (Derda, 2020).

2.6.1 Suppliers

Table Name : **Suppliers** Primary Key : SupplierID Foreign Key : None				
Attribute Name	Data Type	Domain Constraint	Integrity Constraint	Description
SupplierID	Varchar (10)	First Character will be 'S-' followed by a sequential number	Primary Key Not Null Unique	To uniquely record suppliers with alphanumeric numbers To use throughout the database with other tables
SupplierName	Varchar (60)		Not Null	To record Name of supplier/Org
ContactPhone	Varchar (20)			To record Phone no. of supplier
SupplierDescription	Varchar (100)			To record what kinds of products the supplier distributes
SupplierAddress	Varchar (150)			To record the supplier's office/org location

2.6.2 Purchases

Table Name : **Purchases** Primary Key : PurchaseID Foreign Key : SupplierID				
Attribute Name	Data Type	Domain Constraint	Integrity Constraint	Description
PurchaseID	Varchar (10)	First Character will be 'Pur-' followed by a sequential number	Primary Key Not Null Unique	To uniquely record purchases with alphanumeric numbers To use throughout the database with other tables
PurchaseDate	DATE		Not Null	To record Date of Purchase from supplier
SupplierID	Varchar (10)	First Character will be 'S-' followed by a sequential number	Foreign Key Not Null	To uniquely record suppliers with alphanumeric numbers
TotalPurchaseQuantity	Integer			To record the net total quantity of products purchased
TotalPurchaseAmount	Decimal (10,2)			To record the net total amount of money paid for purchase

Propagation Constraint

Add propagation constraints on Purchases table of SupplierID

Foreign Key (SupplierID) References Suppliers (SupplierID)

ON UPDATE CASCADE

ON DELETE NO ACTION

2.6.3 PurchasedProducts

Table Name : **PurchasedProducts** Composite Primary Key : PurchaseID, ProductID Foreign Key : PurchaseID, ProductID				
Attribute Name	Data Type	Domain Constraint	Integrity Constraint	Description
PurchaseID	Varchar (10)	First Character will be 'Pur-' followed by a sequential number	Foreign Key Not Null	To uniquely record purchases with alphanumeric numbers To use throughout the database with other tables
ProductID	Varchar(10)	First Character will be 'P-' followed by a sequential number	Foreign Key Not Null	To uniquely record products with alphanumeric numbers
PurchaseLineQuantity	Integer			To record the line total quantity for products purchased
PurchaseLineAmount	Decimal (10,2)			To record the net total amount of money paid for purchases

Propagation Constraint

Add propagation constraints on PurchaseProducts table of PurchaseID

Foreign Key (PurchaseID) References Purchases (PurchaseID)

ON UPDATE CASCADE

ON DELETE NO ACTION

Add propagation constraints on PurchaseProducts table of ProductID

Foreign Key (ProductID) References Products (ProductID)

ON UPDATE CASCADE

ON DELETE NO ACTION

2.6.4 Categories

Table Name : **Categories**

Primary Key : CategoryID

Foreign Key : None

Attribute Name	Data Type	Domain Constraint	Integrity Constraint	Description
CategoryID	Varchar (10)	First Character will be 'Cat-' followed by a sequential number	Primary Key Not Null Unique	To uniquely record Category of products with alphanumeric numbers To use throughout the database with other tables
CategoryName	Varchar(40)		Not Null	To record name of category
CategoryDescription	Varchar(150)			To record description about the category

2.6.5 Products

Table Name : **Products**

Primary Key : ProductID

Foreign Key : CategoryID

Attribute Name	Data Type	Domain Constraint	Integrity Constraint	Description
ProductID	Varchar (10)	First Character will be 'P-' followed by a sequential number	Primary Key Not Null Unique	To uniquely record Products with alphanumeric numbers To use throughout the database with other tables
ProductName	Varchar (100)		Not Null	To record name of products
CategoryID	Varchar (10)	First Character will be 'Cat-' followed by a sequential number	Foreign Key Not Null	To uniquely record Category of products with alphanumeric numbers
ProductWeight	Decimal (6,2)			To record the weights of products
WeightUnit	Varchar (10)	Domain constraint for 'Viss, 'g', 'kg', 'mL', 'L'		To record the unit type of weights of products
UnitPrice	Decimal (10,2)			To record the unit price of products per piece
ExpiryDate	DATE			To record the expire date of products

Add propagation constraints on Products table of CategoryID

Foreign Key (CategoryID) References Categories (CategoryID)

ON UPDATE CASCADE

ON DELETE NO ACTION

2.6.6 Orders

Table Name : **Orders** Primary Key : OrderID Foreign Key : CustomerID, PromotionID, EmployeeID, DeliveryID				
Attribute Name	Data Type	Domain Constraint	Integrity Constraint	Description
OrderID	Varchar (10)	First Character will be 'Ord-' followed by a sequential number	Primary Key Not Null Unique	To uniquely record Order with alphanumeric numbers To use throughout the database with other tables
CustomerID	Varchar (10)		Foreign Key Not Null	To uniquely record Customers with alphanumeric numbers
OrderDate	DATE		Not Null	To record the date when order was submitted
TotalOrderQuantity	Integer			To record the net total quantity of products ordered
SubTotal	Decimal (10,2)			To record the sum of OrderLineAmount
PromotionID	Varchar (15)	First Character will be 'Promo-' followed by a sequential number	Foreign Key	To uniquely identify Promotion events with alphanumeric numbers
Tax	Decimal (10,2)			To record the tax calculated from order
PromotionDiscount Amount	Decimal (10,2)			To record the promotion discounted amount from order

TotalOrderAmount	Decimal (10,2)			To record the net total amount of money to be paid for order
PaymentAmount	Decimal (10,2)			To record the amount of money customer paid for order
PaymentType	Varchar (20)	Domain constraint for 'Cash on Delivery', 'CBPay', 'KPay', 'Credit/Debit' Default 'Cash on Delivery'		To record the types of payment customer used
PaymentStatus	Varchar (15)	Domain constraint for 'Paid' , 'Not Paid' Default 'Not Paid'		To record whether the customer has paid for the order or not
OrderStatus	Varchar (20)	Domain constraint for 'Preparing', 'On the way', 'Delivered' Default 'Preparing'		To record the order preparation, delivery status of orders
EmployeeID	Varchar (10)	First Character will be 'E-' followed by a sequential number	Foreign Key	To uniquely record Employee the handles the order with alphanumeric numbers
DeliveryID	Varchar (20)	First Character will be 'Deli-' followed by a sequential number	Foreign Key	To uniquely record Delivery who sends the order with alphanumeric numbers
DeliveryAddress	Varchar (255)			To record the addresses to where the delivery has to send the orders

Add propagation constraints on Orders table of CustomerID

Foreign Key (CustomerID) References Customers (CustomerID)

ON UPDATE CASCADE

ON DELETE NO ACTION

Add propagation constraints on Orders table of PromotionID

Foreign Key (PromotionID) References Promotions (PromotionID)

ON UPDATE CASCADE

ON DELETE NO ACTION

Add propagation constraints on Orders table of EmployeeID

Foreign Key (EmployeeID) References Employees (EmployeeID)

ON UPDATE CASCADE

ON DELETE NO ACTION

Add propagation constraints on Orders table of DeliveryID

Foreign Key (DeliveryID) References Delivery (DeliveryID)

ON UPDATE CASCADE

ON DELETE NO ACTION

2.6.7 OrderProducts

Table Name : **OrderProducts** Composite Primary Key : OrderID, ProductID Foreign Key : OrderID, ProductID				
Attribute Name	Data Type	Domain Constraint	Integrity Constraint	Description
OrderID	Varchar (10)	First Character will be 'Ord-' followed by a sequential number	Foreign Key Not Null	To uniquely record Order with alphanumeric numbers
ProductID	Varchar (10)	First Character will be 'P-' followed by a sequential number	Foreign Key Not Null	To uniquely record Products with alphanumeric numbers
OrderLineQuantity	Integer			To record the line total quantity for products ordered
OrderLineAmount	Decimal (10,2)			To record the line total amount of money for products ordered

Add propagation constraints on OrderProducts table of OrderID

Foreign Key (OrderID) References Orders (OrderID)

ON UPDATE CASCADE

ON DELETE NO ACTION

Add propagation constraints on OrderProducts table of ProductID

Foreign Key (ProductID) References Products (ProductID)

ON UPDATE CASCADE

ON DELETE NO ACTION

2.6.8 Customers

Table Name : **Customers** Composite Primary Key : CustomerID Foreign Key : None				
Attribute Name	Data Type	Domain Constraint	Integrity Constraint	Description
CustomerID	Varchar (10)	First Character will be 'Cus-' followed by a sequential number	Primary Key Not Null Unique	To uniquely record Customers with alphanumeric numbers
CustomerName	Varchar (100)			To record the name of customers
CustomerEmail	Varchar (100)	Emails must end with '@gmail.com'		To record email of customers
CustomerPhone	Varchar (20)			To record the phone number of customers
CustomerAddress	Varchar (150)			To record the address of customers

2.6.9 Employees

Table Name : **Employees** Composite Primary Key : EmployeeID Foreign Key : None				
Attribute Name	Data Type	Domain Constraint	Integrity Constraint	Description
EmployeeID	Varchar (10)	First Character will be 'E-' followed by a sequential number	Primary Key Not Null Unique	To uniquely record Employees with alphanumeric numbers
EmployeeName	Varchar (100)		Not Null	To record the name of employees
Gender	Varchar (30)	Domain constraint for 'Male', 'Female', 'Others'		To record Gender of employees
Position	Varchar (30)		Not Null	To record the position of employees in org
EmployeeEmail	Varchar(100)	Emails must end with '@gmail.com'		To record the email address of employees
EmployeeAddress	Varchar(150)			To record the address of employees
DateOfBirth	DATE			To record the age/DOB of employees

2.6.10 Promotions

Table Name : **Promotions** Composite Primary Key : PromotionID Foreign Key : None				
Attribute Name	Data Type	Domain Constraint	Integrity Constraint	Description
PromotionID	Varchar (15)	First Character will be 'Promo-' followed by a sequential number	Primary Key Not Null Unique	To uniquely record Promotion events with alphanumeric numbers
PromotionCode	Varchar (20)		Not Null	To record the codes for promotions
PromotionType	Varchar (30)	Domain constraint for 'Discount Percentage', 'Discount Amount'		To record the types of promotions
PromotionDescription	Varchar (255)			To record the descriptions about the promotion events
PromotionAmount	Decimal (10,2)			To record the value of discount percentage/amount
PromotionActive	Varchar (20)	Domain constraint for 'Active', 'Expired'		To record the status of promotion event
PromotionStartDate	DATE			To record the start date of promotion event
PromotionEndDate	DATE			To record the end date of promotion event

2.6.11 Delivery

Table Name : **Delivery** Composite Primary Key : DeliveryID Foreign Key : None				
Attribute Name	Data Type	Domain Constraint	Integrity Constraint	Description
DeliveryID	Varchar (15)	First Character will be 'Deli-' followed by a sequential number	Primary Key Not Null Unique	To uniquely record Delivery who sends the order with alphanumeric numbers
DeliveryDate	DATE			To record the date of order delivery

Task 3

Normalization

3 Task 3: Normalization

3.1 Normalization and its Purpose

Normalization in Database is a process/technique utilized in structuring a Relational Database accordingly to a series of normal forms, in order to reduce data redundancy, improve data integrity and to beat potential anomalies (Insert/ Update/ Delete anomalies) due to data replication. (SINGH, 2015) It is a bottom-up technique to identify logical relationships between data items.

Definitions of each normal forms up to 3rd normal form:

- 1st Normal Form aims to get rid of repeating groups of information. An entity is in 1NF when it contains no repeating groups of data. (Watt, 2014)
- 2nd Normal Form aims to get rid of partial key dependencies. An entity is in 2NF when it is in 1NF and when all of its non-key attributes are fully depend on the primary key. (Watt, 2014)
- 3rd Normal Form aims to remove any non-key dependencies. An entity is in 3NF when it is in 2NF and when all of its attributes are directly depend on the primary key. (Watt, 2014)

3.2 Normalization and Why each of entities is in 3NF

3.2.1 Normalization for Document 1: Purchase products from Suppliers

UNF	Level	1NF	2NF	3NF
PurchaseID**(PK)** PurchaseDate SupplierID SupplierName ContactPhone SupplierDescription SupplierAddress ProductID ProductName ProductWeight WeightUnit UnitPrice ExpiryDate PurchaseLineQuantity PurchaseLineAmount TotalPurchaseQuantity TotalPurchaseAmount	1 1 1 1 1 1 2 2 2 2 2 2 2 2 1 1	**PurchaseID(PK)** PurchaseDate SupplierID SupplierName ContactPhone SupplierDescription SupplierAddress TotalPurchaseQuantity TotalPurchaseAmount PurchaseID(FK) ProductID ProductName ProductWeight WeightUnit UnitPrice ExpiryDate PurchaseLineQuantity PurchaseLineAmount	**PurchaseID(PK)** PurchaseDate SupplierID SupplierName ContactPhone SupplierDescription SupplierAddress TotalPurchaseQuantity TotalPurchaseAmount PurchaseID(FK,PK) ProductID(FK,PK) PurchaseLineQuantity PurchaseLineAmount **ProductID(PK)** ProductName ProductWeight WeightUnit UnitPrice ExpiryDate	**PurchaseID(PK)** SupplierID(FK) PurchaseDate TotalPurchaseQuantity TotalPurchaseAmount **SupplierID(PK)** SupplierName ContactPhone SupplierDescription SupplierAddress PurchaseID(FK,PK) ProductID(FK,PK) PurchaseLineQuantity PurchaseLineAmount **ProductID(PK)** ProductName ProductWeight WeightUnit UnitPrice ExpiryDate

Normalized and Optimized Tables for Document 1

Assumption

The rest of the required attributes are added to the Normalized and Optimized Table column, from the Data Dictionary.

Normalized and Optimized Tables
[Purchases] **PurchaseID(PK)** SupplierID(FK) PurchaseDate TotalPurchaseQuantity TotalPurchaseAmount **[Suppliers]** **SupplierID(PK)** SupplierName ContactPhone SupplierDescription SupplierAddress **[PurchasedProducts]** PurchaseID(FK) ProductID(FK) PurchaseLineQuantity PurchaseLineAmount **[Products]** **ProductID(PK)** ProductName ProductWeight WeightUnit

UnitPrice ExpiryDate

The resulting Entity Relationship Diagram from Document 1

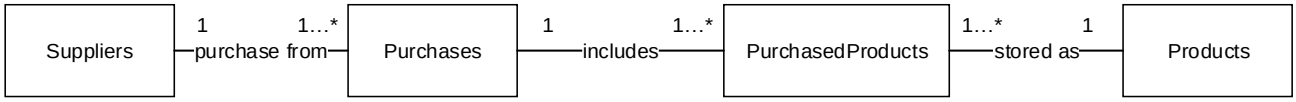


Figure 3: ERD result from Document 1 Normalization

The above entities are in 3NF because

- The entities have all atomic values in each attributes, no repeating groups of data. (Satisfied 1NF)
- The entities are already in 1NF and have no partial key dependencies, each non-key attributes are fully dependent on their primary key. (Satisfied 2NF) (2NF made sure ProductName, Weight, UnitPrice, ExpiryDate are fully depend on Primary Key, ProductID and ProductLineQuantity and ProductLineAmount fully depends on their Composite Primary key,PurchaseID + ProductID.)
- The entities are already in 2NF and have no non-key dependencies, all of the attributes are directly dependent on their primary key. (Satisfied 3NF) (3NF made sure that Supplier Information directly depend on Primary key, SupplierID by separating from Purchase entity.)

3.2.2 Normalization for Document 2: Products classified into Categories

UNF	Level	1NF	2NF	3NF
CategoryID	1	**CategoryID(PK)**	**CategoryID(PK)**	**CategoryID(PK)**
CategoryName	1	CategoryName	CategoryName	CategoryName
CategoryDescription	1	CategoryDescription	CategoryDescription	CategoryDescription
ProductID**(PK)**	2			
ProductName	2	ProductID(PK)	**ProductID(PK)**	**ProductID(PK)**
		CategoryID(FK)	CategoryID(FK)	CategoryID(FK)
		ProductName	ProductName	ProductName

Normalized and Optimized Tables for Document 2

Assumption

The rest of the required attributes are added to the Normalized and Optimized Table column, from the Data Dictionary.

Normalized and Optimized Tables
[Products]
ProductID(PK)
ProductName
ProductWeight
WeightUnit
UnitPrice
ExpiryDate
CategoryID(FK)
[Categories]
CategoryID(PK)
CategoryName
CategoryDescription

The resulting Entity Relationship Diagram from Document 2

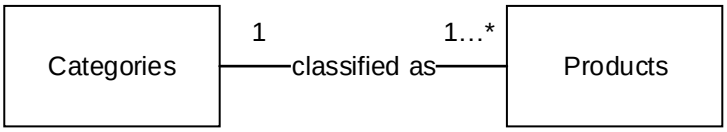


Figure 4: ERD result from Document 2 Normalization

The above entities are in 3NF because

- The entities have all atomic values in each attributes, no repeating groups of data. (Satisfied 1NF)
- The entities are already in 1NF and have no partial key dependencies, each non-key attributes are fully dependent on their primary key. (Satisfied 2NF) (All attributes here fully depend on their primary keys.)
- The entities are already in 2NF and have no non-key dependencies, all of the attributes are directly dependent on their primary key. (Satisfied 3NF) (All non-key attributes here directly depend on their primary keys.)

3.2.3 Normalization for Document 3: Customers Ordering Products

UNF	Level	1NF	2NF	3NF
OrderID**(PK)**	1	**OrderID(PK)**	**OrderID(PK)**	**OrderID(PK)**
OrderDate	1	OrderDate	OrderDate	OrderDate
CustomerID	1	CustomerID	CustomerID	CustomerID(FK)
CustomerName	1	CustomerName	CustomerName	DeliveryAddress
CustomerEmail	1	CustomerEmail	CustomerEmail	TotalOrderQuantity
CustomerPhone	1	CustomerPhone	CustomerPhone	SubTotal
CustomerAddress	1	CustomerAddress	CustomerAddress	PromotionID(FK)
DeliveryAddress	1	DeliveryAddress	DeliveryAddress	Tax
ProductID	2	TotalOrderQuantity	TotalOrderQuantity	TotalOrderAmount
ProductName	2	SubTotal	SubTotal	EmployeeID(FK)
ProductWeight	2	PromotionCode	PromotionCode	DeliveryID(FK)
WeightUnit	2	Tax	Tax	OrderStatus
UnitPrice	2	TotalOrderAmount	TotalOrderAmount	
OrderLineQuantity	2	EmployeeID	EmployeeID	
OrderLineAmount	2	EmployeeName	EmployeeName	PaymentStatus
TotalOrderQuantity	1	DeliveryID	DeliveryID	PaymentAmount
SubTotal	1	DeliveryDate	DeliveryDate	PaymentType
PromotionCode	1	OrderStatus	OrderStatus	
Tax	1	PaymentStatus	PaymentStatus	
TotalOrderAmount	1	PaymentAmount	PaymentAmount	**OrderID(FK,PK)**
EmployeeID	1	PaymentType	PaymentType	**ProductID(FK,PK)**
EmployeeName	1			OrderLineQuantity
DeliveryID	1			OrderLineAmount
DeliveryDate	1	**OrderID(FK)**	**OrderID(FK,PK)**	
OrderStatus	1	ProductID	**ProductID(FK,PK)**	
PaymentStatus	1	ProductName	OrderLineQuantity	**ProductID(PK)**
PaymentAmount	1	ProductWeight	OrderLineAmount	ProductName
PaymentType	1	WeightUnit		ProductWeight
		UnitPrice		WeightUnit
		OrderLineQuantity	**ProductID(PK)**	UnitPrice
		OrderLineAmount	ProductName	

			ProductWeight WeightUnit UnitPrice	**CustomerID(PK)** CustomerName CustomerEmail CustomerPhone CustomerAddress **EmployeeID(PK)** EmployeeName **DeliveryID(PK)** DeliveryDate **PromotionID(PK)** PromotionCode

Normalized and Optimized Tables for Document 3

Assumption

The rest of the required attributes are added to the Normalized and Optimized Table column, from the Data Dictionary.

Normalized and Optimized Tables
[Orders] **OrderID(PK)** OrderDate CustomerID(FK) DeliveryAddress TotalOrderQuantity SubTotal PromotionID(FK) Tax

TotalOrderAmount
EmployeeID(FK)
DeliveryID(FK)
OrderStatus
PaymentID(FK)

[Payment]
PaymentID(PK)
PaymentStatus
PaymentAmount
PaymentType

[OrderProducts]
OrderID(FK)
ProductID(FK)
OrderLineQuantity
OrderLineAmount

[Products]
ProductID(PK)
ProductName
CategoryID(FK)
ProductWeight
WeightUnit
UnitPrice
ExpiryDate

[Customers]
CustomerID(PK)
CustomerName
CustomerEmail
CustomerPhone
CustomerAddress

[Employees]
EmployeeID(PK)
EmployeeName
Gender
Position
EmployeeEmail
EmployeeAddress
DateofBirth

[Delivery]
DeliveryID(PK)
DeliveryDate

[Promotions]
PromotionID(PK)
PromotionCode

The resulting Entity Relationship Diagram from Document 3

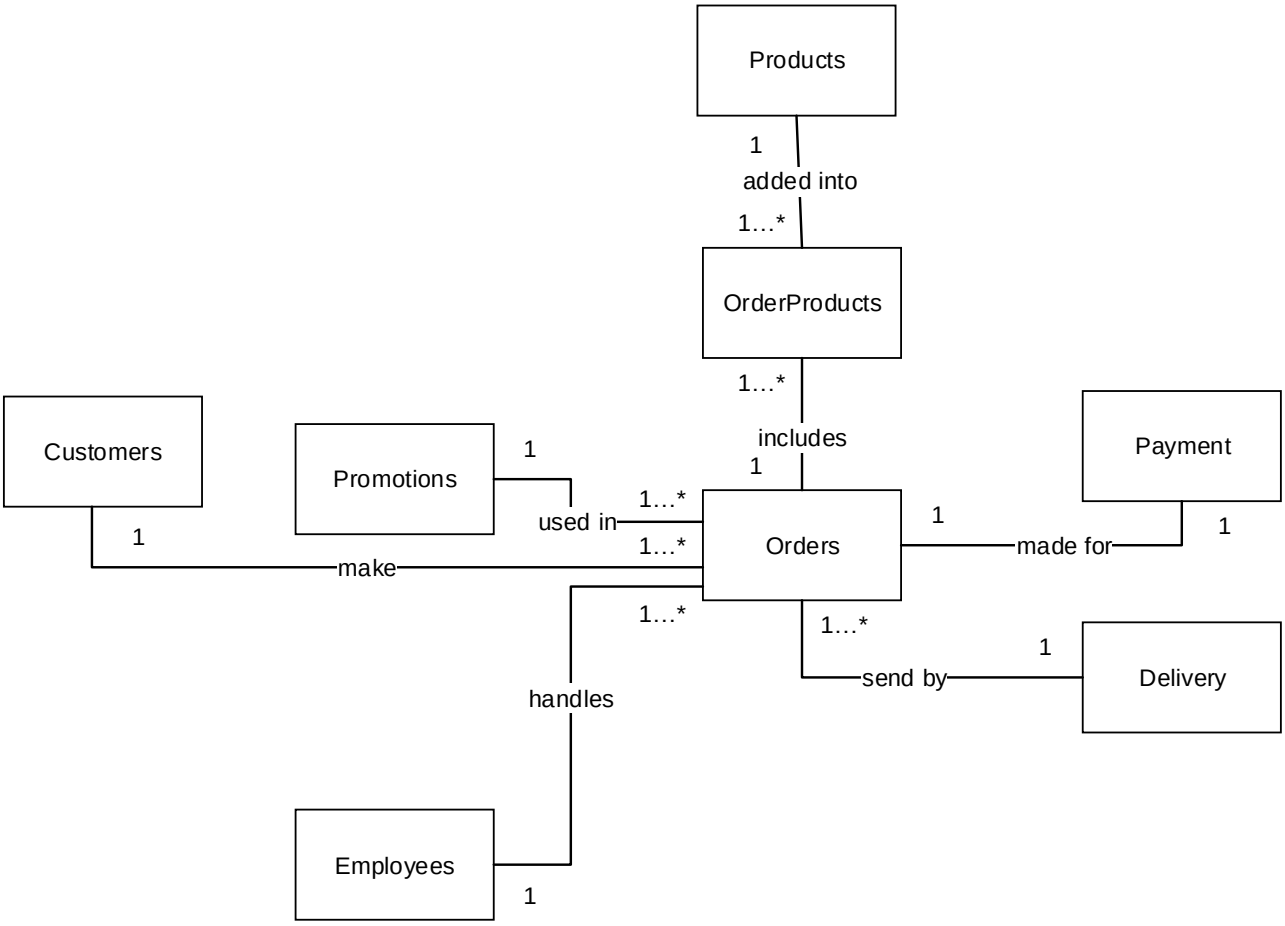


Figure 5: ERD result from Document 3 Normalization

The above entities are in 3NF because

- The entities have all atomic values in each attributes, no repeating groups of data. (Satisfied 1NF)
- The entities are already in 1NF and have no partial key dependencies, each non-key attributes are fully dependent on their primary key. (Satisfied 2NF) (2NF made sure ProductName, Weight, UnitPrice are fully depend on Primary Key, ProductID and OrderLineQuantity and OrderLineAmount fully depends on their Composite Primary key,OrderID + ProductID.)
- The entities are already in 2NF and have no non-key dependencies, all of the attributes are directly dependent on their primary key. (Satisfied 3NF) (3NF made sure that Customer Information directly depend on Primary key, CustomerID by separating from Orders entity.)

3.2.4 Normalization for Document 4: Applying Promotions to Orders

UNF	Level	1NF	2NF	3NF
PromotionID	1	**PromotionID(PK)**	**PromotionID(PK)**	**PromotionID(PK)**
PromotionCode	1	PromotionCode	PromotionCode	PromotionCode
PromotionType	1	PromotionType	PromotionType	PromotionType
PromotionAmount	1	PromotionAmount	PromotionAmount	PromotionAmount
PromotionDescription	1	PromotionDescription	PromotionDescription	PromotionDescription
PromotionActive	1	PromotionActive	PromotionActive	PromotionActive
PromotionStartDate	1	PromotionStartDate	PromotionStartDate	PromotionStartDate
PromotionEndDate	2	PromotionEndDate	PromotionEndDate	PromotionEndDate
OrderID**(PK)**	2			
OrderDate	2	PromotionID(FK)	PromotionID(FK)	PromotionID(FK)
SubTotal	2	OrderID	OrderID(PK)	**OrderID(PK)**
Tax	2	OrderDate	OrderDate	OrderDate
TotalOrderAmount		SubTotal	SubTotal	SubTotal
		Tax	Tax	Tax
		TotalOrderAmount	TotalOrderAmount	TotalOrderAmount

Normalized and Optimized Tables for Document 4

Assumption

The rest of the required attributes are added to the Normalized and Optimized Table column, from the Data Dictionary.

Normalized and Optimized Tables
[Orders] **OrderID(PK)** OrderDate

CustomerID(FK)
DeliveryAddress
TotalOrderQuantity
SubTotal
PromotionID(FK)
Tax
TotalOrderAmount
EmployeeID(FK)
DeliveryID(FK)
OrderStatus
PaymentID(FK)

[Promotions]
PromotionID(PK)
PromotionCode
PromotionType
PromotionAmount
PromotionDescription
PromotionActive
PromotionStartDate
PromotionEndDate

The resulting Entity Relationship Diagram from Document 4

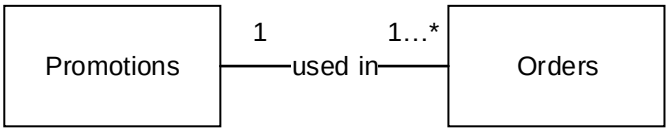


Figure 6: ERD result from Document 4 Normalization

The above entities are in 3NF because

- The entities have all atomic values in each attributes, no repeating groups of data. (Satisfied 1NF)

- The entities are already in 1NF and have no partial key dependencies, each non-key attributes are fully dependent on their primary key. (Satisfied 2NF) (All attributes here fully depend on their primary keys.)
- The entities are already in 2NF and have no non-key dependencies, all of the attributes are directly dependent on their primary key. (Satisfied 3NF) (All non-key attributes here directly depend on their primary keys.)

3.2.5 Normalization for Document 5: Order Delivery Record

UNF	Level	1NF	2NF	3NF
OrderID(PK)	2	DeliveryID(PK)	DeliveryID(PK)	DeliveryID(PK)
OrderDate	2	DeliveryDate	DeliveryDate	DeliveryDate
OrderStatus	2			
DeliveryAddress	2	**OrderID(PK)**	**OrderID(PK)**	**OrderID(PK)**
DeliveryID	1	DeliveryID(FK)	DeliveryID(FK)	DeliveryID(FK)
DeliveryDate	1	OrderDate	OrderDate	OrderDate
		OrderStatus	OrderStatus	OrderStatus

Normalized and Optimized Tables for Document 5

Assumption

The rest of the required attributes are added to the Normalized and Optimized Table column, from the Data Dictionary.

Normalized and Optimized Tables
[Orders]
OrderID(PK)
OrderDate
CustomerID(FK)
DeliveryAddress
TotalOrderQuantity
SubTotal
PromotionID(FK)
Tax
TotalOrderAmount
EmployeeID(FK)
DeliveryID(FK)
OrderStatus

PaymentID(FK) **[Delivery]** **DeliveryID(PK)** DeliveryDate

The resulting Entity Relationship Diagram from Document 5

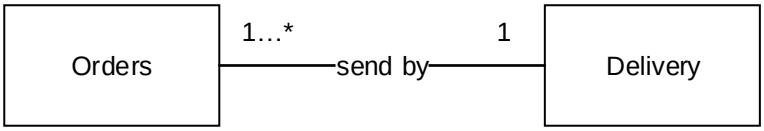


Figure 7: ERD result from Document 5 Normalization

The above entities are in 3NF because

- The entities have all atomic values in each attributes, no repeating groups of data. (Satisfied 1NF)
- The entities are already in 1NF and have no partial key dependencies, each non-key attributes are fully dependent on their primary key. (Satisfied 2NF) (All attributes here fully depend on their primary keys.)
- The entities are already in 2NF and have no non-key dependencies, all of the attributes are directly dependent on their primary key. (Satisfied 3NF) (All non-key attributes here directly depend on their primary keys.)

All the above 11 entities are in 3NF because

- The entities have all atomic values in each attributes, no repeating groups of data. (Satisfied 1NF)
- The entities are already in 1NF and have no partial key dependencies, each non-key attributes are fully dependent on their primary key. (Satisfied 2NF)
- The entities are already in 2NF and have no non-key dependencies, all of the attributes are directly dependent on their primary key. (Satisfied 3NF)

3.3 How Normalization is used to check tables are well-structured and How it solved the problem of update anomalies

A well-structured table contains a lowest possible amount of data redundancy and users can insert, modify, and delete the rows in a table without errors or inconsistency (by eliminating anomalies) and promotes data integrity and performance in database. (Jennifer, n.d.)

For instance, if we review the Normalization process for Document 3: Customers Ordering Products…

In 1NF Normalization, repeating groups were removed, (thus minimizing data duplicates) and made sure that each cells in the tables contains only atomic (single) value.

In 2NF normalization, partial key dependencies are removed (therefore, avoid anomalies problems). Product related columns can now be easily updated without update anomalies. If partial key dependencies were not removed and if a product's name is updated to a new one, every rows which include that specific product's name will need to be updated. If some rows were left not updated, there will be inconsistency in data integrity. If a product needs to be deleted, every rows which include that specific product will need to be deleted. To insert a new row for a new product, it is necessary to provide value for OrderID. After removing partial dependencies, every non-key attributes now fully dependent on the primary key and anomalies issues are solved.

In 3NF normalization, transitive/ non-key dependencies are removed (which also solves anomalies problems). Customer related columns can now be easily updated without update anomalies. If transitive dependencies were not removed and if a Customer's Name is updated, every rows that includes that specific customer's name will need to be updated. If some rows were left not updated, there will be inconsistency in data integrity. If customer's Info needs to be deleted, every rows that include customer's information will be deleted together with employee information and order, payment, delivery, promotion information. To insert a new customer in a new row, it will be needed to provide values for Order, Payment, Delivery, even though customer haven't made any orders. Once non-key dependencies are removed, all of the attributes are directly dependent on their primary key and eliminates anomalies.

Task 4

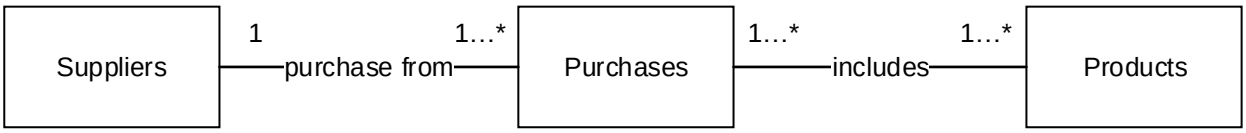
Assessment of Design

4 Task 4: Assessment of Design

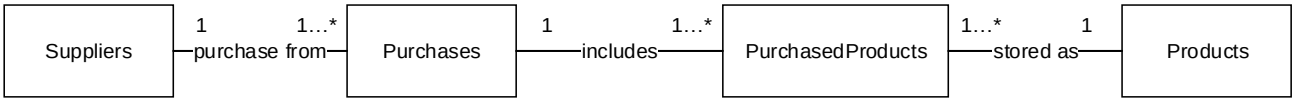
4.1 Mapped logical database design to physical database design

In this step, in order to start creating Physical database, the entities/tables resulted from Top-down method (Entity Relationship Diagram) and Bottom-Up method (Normalization) are mapped to reconsider and re-examine relationships, which entities should be decomposed further, which entities should be de-normalized or recombined and which attributes should be separated as a separated entities. Then, entities are changed into tables, attributes are changed into columns.

In Document 1: Purchase products from Suppliers, in conceptual and logical database design, one to many products can be purchased from one to many different suppliers and products can be purchased many times. Therefore, there is one to many relationship between Suppliers and Purchases entities and many to many relationship between Purchases and Products entities. Same relationships are also discovered in ERD.

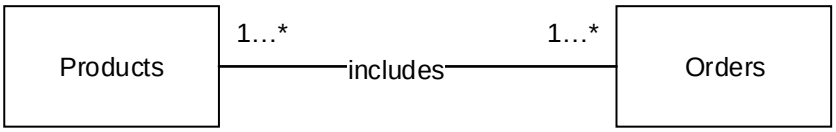


After normalization, finalized ERD results shows a new dummy table: “PurchasedProducts” is discovered with one to many relationship from Purchases and Products tables.

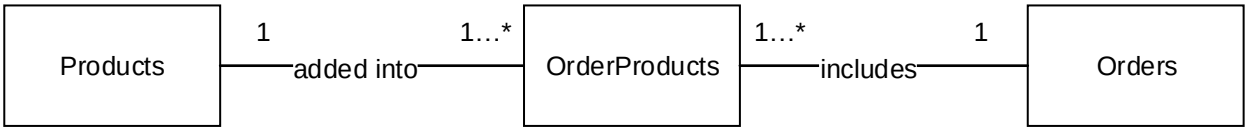


To create physical tables from entities, many to many relationships should be decomposed as it can add complexity and confusion to the model (IBM, 2021) so that we will accept this new dummy table as it supports users’ transactions and resolves many to many relationship between Purchase and Products.

In Document 3: Customers Ordering Products, in conceptual and logical database design, one to many products can be ordered in an order and orders can include one to many products. Therefore, there is many to many relationship between Orders and Products entities. Same relationships are also discovered in ERD.

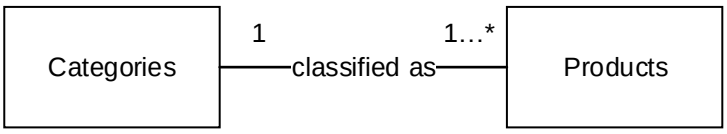


After normalization, finalized ERD results shows a new dummy table: “OrderProducts” is discovered with one to many relationship from Orders and Products tables.



To create physical tables from entities, many to many relationships should be decomposed as it can add complexity and confusion to the model (IBM, 2021) so that we will also accept this new dummy table as it also supports users’ transactions and resolves many to many relationship between Orders and Products.

In Document 2: Products classified into Categories, in conceptual and logical database design, Products will be classified into respective categories. Therefore, there is one to many relationship between Products and Categories entities.

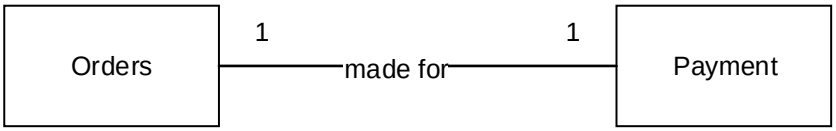


After normalization, finalized ERD results in the same one to one relationship. Categories table’s attributes can be combined within Products table. But in order to reduce data duplications and storage and as categories has attributes which has many values (such as “Meat and seafood”, “Fruits and Vegetables”, “Snacks and Beverages” and many more), these attributes will be separated into a new table: “Categories” to make dynamic.

The rest of the entities will transformed into physical database tables with the same relationships in logical design stage.

4.2 De-normalization

In Document 3: Customers Ordering Products, in conceptual and logical database design, Customer can make specific payment for each of their specific order. Therefore, there is one to one relationship between Orders and Payment entities.



After normalization, finalized ERD results in the same one to one relationship but the payment table can be de-normalized and attributes can be combined within orders tables. By doing so, it will help reduce JOINs in our relational database and retrieving data will be faster. Queries to retrieve data will be simpler and less mistake since we have to look at fewer tables. (Dhyawala, 2018)

4.3 Designed Tables for target DBMS

In this assignment, for physical design development, “Microsoft SQL Server 2019” DBMS is used. SQL language (DDL, DML, DCL, DIL) is used to develop and make changes in database tables all together with Aggregate functions, Order By, Group By, Having keywords.

Name of Table:	**Suppliers**
List of Columns and Constraints:	**SupplierID**: Varchar(10) NOT NULL First Character will be ‘S-’ followed by a sequential number for SupplierID Duplicates are not allowed for SupplierID, **SupplierName**: Varchar(60) NOT NULL, **ContactPhone**: Varchar(20), **SupplierDescription**: Varchar(100), **SupplierAddress**: Varchar(150)
Primary Key:	(SupplierID)
Foreign Key:	None

Name of Table:	**Purchases**
List of Columns and Constraints:	**PurchaseID**: Varchar(10) NOT NULL First Character will be ‘Pur-’ followed by a sequential number for PurchaseID Duplicates are not allowed for PurchaseID, **PurchaseDate**: DATE NOT NULL, **SupplierID**: Varchar(10) NOT NULL, **TotalPurchaseQuantity**: Integer, **TotalPurchaseAmount**: Decimal(10,2)
Primary Key:	(PurchaseID)
Foreign Key:	(SupplierID) ON UPDATE CASCADE ON DELETE NO ACTION

Name of Table:	**PurchasedProducts**
List of Columns and Constraints:	**PurchaseID**: Varchar(10) NOT NULL First Character will be 'Pur-' followed by a sequential number for PurchaseID, **ProductID**: Varchar(10) NOT NULL First Character will be 'P-' followed by a sequential number for ProductID, **PurchaseLineQuantity**: Integer, **PurchaseLineAmount**: Decimal(10,2)
Primary Key:	(PurchaseID,ProductID)
Foreign Key:	(PurchaseID) ON UPDATE CASCADE ON DELETE NO ACTION (ProductID) ON UPDATE CASCADE ON DELETE NO ACTION

Name of Table:	**Categories**
List of Columns and Constraints:	**CategoryID**: Varchar(10) NOT NULL First Character will be 'Cat-' followed by a sequential number for CategoryID Duplicates are not allowed for CategoryID, **CategoryName**: Varchar(40) NOT NULL, **CategoryDescription**: Varchar(150)
Primary Key:	(CategoryID)
Foreign Key:	None

Name of Table:	**Products**
List of Columns and Constraints:	**ProductID**: Varchar(10) NOT NULL First Character will be 'P-' followed by a sequential number for ProductID Duplicates are not allowed for ProductID, **ProductName**: Varchar(100) NOT NULL, **CategoryID**: Varchar(10) NOT NULL First Character will be 'Cat-' followed by a sequential number for CategoryID, **ProductWeight**: Decimal (6,2), **WeightUnit**: Varchar(10) Domain constraint for 'Viss, 'g', 'kg', 'mL', 'L', **UnitPrice**: Decimal (10,2), **ExpiryDate**: DATE
Primary Key:	(ProductID)
Foreign Key:	(CategoryID) ON UPDATE CASCADE ON DELETE NO ACTION

Name of Table:	**Orders**
List of Columns and Constraints:	**OrderID**: Varchar(10) NOT NULL First Character will be 'Ord-' followed by a sequential number for OrderID Duplicates are not allowed for OrderID, **CustomerID**: Varchar(10) NOT NULL, **OrderDate**: DATE NOT NULL, **TotalOrderQuantity**: Integer, **SubTotal**: Decimal (10,2), **PromotionID**: Varchar(15) First Character will be 'Promo-' followed by a sequential number for PromotionID, **PromotionDiscountAmount**: Decimal (10,2), **Tax**: Decimal (10,2), **TotalOrderAmount**: Decimal (10,2), **PaymentAmount**: Decimal (10,2), **PaymentType**: Varchar(20)

	Domain constraint for ‘Cash on Delivery’, ‘CBPay’, ‘KPay’, ‘Credit/Debit’ Default ‘Cash on Delivery’, **PaymentStatus**: Varchar(15) Domain constraint for ‘Paid’ , ‘Not Paid’ Default ‘Not Paid’, **OrderStatus**: Varchar(20) Domain constraint for ‘Preparing’, ‘On the way’, ‘Delivered’ Default ‘Preparing’, **EmployeeID**: Varchar(10) First Character will be ‘E-’ followed by a sequential number for EmployeeID, **DeliveryID**: Varchar(15) First Character will be ‘Deli-’ followed by a sequential number for DeliveryID, **DeliveryAddress**: Varchar(255)
Primary Key:	(OrderID)
Foreign Key:	(CustomerID) ON UPDATE CASCADE ON DELETE NO ACTION (PromotionID) ON UPDATE CASCADE ON DELETE NO ACTION (EmployeeID) ON UPDATE CASCADE ON DELETE NO ACTION (DeliveryID) ON UPDATE CASCADE ON DELETE NO ACTION

Name of Table:	**OrderProducts**
List of Columns and Constraints:	**OrderID**: Varchar(10) NOT NULL First Character will be ‘Ord-’ followed by a sequential number for OrderID, **ProductID**: Varchar(10) NOT NULL First Character will be ‘P-’ followed by a sequential number for ProductID, **OrderLineQuantity**: Integer, **OrderLineAmount**: Decimal(10,2)
Primary Key:	(OrderID,ProductID)
Foreign Key:	(OrderID) ON UPDATE CASCADE ON DELETE NO ACTION (ProductID) ON UPDATE CASCADE ON DELETE NO ACTION

Name of Table:	**Customers**
List of Columns and Constraints:	**CustomerID**: Varchar(10) NOT NULL First Character will be 'Cus-' followed by a sequential number for CustomerID Duplicates are not allowed for CustomerID, **CustomerName**: Varchar(100) , **CustomerEmail**: Varchar(100) Emails must end with '@gmail.com', **CustomerPhone**: Varchar(20), **CustomerAddress**: Varchar(150)
Primary Key:	(CustomerID)
Foreign Key:	None

Name of Table:	**Employees**
List of Columns and Constraints:	**EmployeeID**: Varchar(10) NOT NULL First Character will be 'E-' followed by a sequential number for EmployeeID Duplicates are not allowed for EmployeeID, **EmployeeName**: Varchar(100) NOT NULL, **Gender**: Varchar(30) Domain constraint for 'Male', 'Female', 'Others', **Position**: Varchar(30) NOT NULL, **EmployeeEmail**: Varchar(100) Emails must end with '@gmail.com', **EmployeeAddress**: Varchar(150), **DateOfBirth**: DATE
Primary Key:	(EmployeeID)
Foreign Key:	None

Name of Table:	**Promotions**
List of Columns and Constraints:	**PromotionID**: Varchar(15) NOT NULL First Character will be 'Promo-' followed by a sequential number for PromotionID Duplicates are not allowed for PromotionID, **PromotionCode**: Varchar(20) NOT NULL, **PromotionType**: Varchar(30) Domain constraint for 'Discount Percentage', 'Discount Amount', **PromotionDescription**: Varchar(255), **PromotionAmount**: Decimal(10,2), **PromotionActive**: Varchar(20) Domain constraint for 'Active', 'Expired', **PromotionStartDate**: DATE, **PromotionEndDate**: DATE
Primary Key:	(PromotionID)
Foreign Key:	None

Name of Table:	**Delivery**
List of Columns and Constraints:	**DeliveryID**: Varchar(15) NOT NULL First Character will be 'Deli-' followed by a sequential number for DeliveryID Duplicates are not allowed for DeliveryID, **DeliveryDate**: DATE
Primary Key:	(DeliveryID)
Foreign Key:	None

4.4 Identifying Derived Data

Product ID	Product Name	Weight	Unit Price	Order Line Quantity(Pcs)	Order Line Amount
P-001	Chicken Breast	0.3 Viss	4,000	2	8,000
P-015	Grapes	0.3 Viss	6,000	3	18,000
P-007	Ovaltine Chocolate Bottle	400 g	9,000	1	9,000
P-013	Cabbage 1 pack		900	1	900
P-009	Pork Tenderloin	0.5 Viss	8,000	2	16,000
			Total Order Quantity	9	
				Sub Total	51,900
				Promotion Discount Amount (B4Uoff10%)	-5,190
				Tax (5%)	2,595
				Total Order Amount	49,305

Figure 8: From document 3: Customers Ordering Products

In **OrderProducts** Table, OrderLineAmount is derived data columns.

OrderID (FK, PK)

ProductID (FK, PK)

OrderLineQuantity

/OrderLineAmount: (Products.UnitPrice) WHERE Products.ProductID = OrderProducts.ProductID

* (OrderProducts.OrderLineQuantity)

SQL Code for OrderLineAmount

```
UPDATE OrderProducts
SET OrderLineAmount = (OrderLineQuantity) * (SELECT UnitPrice FROM Products
                                    WHERE Products.ProductID = OrderProducts.ProductID);
```

In **Orders** Table, TotalOrderQuantity, SubTotal, Tax and TotalOrderAmount are derived data columns.

OrderID (PK)

CustomerID (FK)

OrderDate

/TotalOrderQuantity: (SUM (OrderProducts.OrderLineQuantity))

WHERE Orders.OrderID = OrderProducts.OrderID

/SubTotal: (SUM (OrderLineAmount))

WHERE Orders.OrderID = OrderProducts.OrderID

PromotionID

/Tax: Orders.SubTotal * 5%

/PromotionDiscountAmount: (Orders.SubTotal) * (PromotionAmount)

WHERE Orders.PromotionID = Promotions.PromotionID

/TotalOrderAmount: (Orders.SubTotal) - Orders.PromotionDiscountAmount + Tax

PaymentAmount

PaymentType

PaymentStatus

OrderStatus

EmployeeID (FK)

DeliveryID (FK)

SQL Codes for TotalOrderQuantity, SubTotal, Tax, PromotionDiscountAmount, TotalOrderAmount

```
UPDATE Orders
SET TotalOrderQuantity = (SELECT SUM(OrderLineQuantity) FROM OrderProducts
                          WHERE Orders.OrderID = OrderProducts.OrderID);
UPDATE Orders
SET SubTotal = (SELECT SUM(OrderLineAmount) FROM OrderProducts
                WHERE Orders.OrderID = OrderProducts.OrderID);
UPDATE Orders
SET Tax = (SubTotal) * 0.05;
UPDATE Orders
SET PromotionDiscountAmount = (Orders.SubTotal) * (SELECT PromotionAmount FROM
                              Promotions
                              WHERE Orders.PromotionID = Promotions.PromotionID);
UPDATE Orders
SET TotalOrderAmount = (Orders.SubTotal) - ISNULL(0,(Orders.PromotionDiscountAmount)) +
                       (Orders.Tax);
```

Product ID	Product Name	Product Weight	Unit Price(ks)	Expire Date	Purchase Line Quantity (Pcs)	Purchase Line Amount
P-001	Chicken Breast	0.3 Viss	4,000		30	120,000
P-002	Burmese Whole Chicken	0.5 Viss	6,000		25	150,000
P-003	Pork 3 layers	300 g	3,200		25	80,000
P-004	Chicken Thigh with skin	0.5 Viss	2,500		30	75,000
P-008	Pork Head	0.5 Viss	6,000		10	60,000
P-013	Cabbage 1 pack		900		20	18,000
P-014	Banana 1 pack		2,200		20	44,000
				Total Purchase Quantity	160	
					Total Purchase Amount	547,000

Figure 9: From Document 1: Purchase products from suppliers

In **PurchasedProducts** Table, PurchaseLineAmount is derived data columns.

PurchaseID (FK, PK)

ProductID (FK, PK)

PurchaseLineQuantity

/PurchaseLineAmount: (Products.UnitPrice)

WHERE Products.ProductID = PurchasedProducts.ProductID

* (PurchasedProducts.PurchaseLineQuantity)

SQL Code for PurchaseLineAmount

```
UPDATE PurchasedProducts
SET PurchaseLineAmount = (PurchaseLineQuantity) * (SELECT UnitPrice FROM Products
                         WHERE Products.ProductID = PurchasedProducts.ProductID);
```

In **Purchases** Table, TotalPurchaseQuantity and TotalPurchaseAmount are derived data columns.

PurchaseID (PK)

SupplierID (FK)

PurchaseDate

/TotalPurchaseQuantity: (SUM (PurchasedProducts.PurchaseLineQuantity))

WHERE Purchases.PurchaseID = PurchasedProducts.PurchaseID

/TotalOrderAmount: SUM(PurchasedProducts.PurchaseLineAmount)

WHERE Purchases.PurchaseID = PurchasedProducts.PurchaseID

SQL Code for TotalPurchaseQuantity, TotalPurchaseAmount

```
UPDATE Purchases
SET TotalPurchaseQuantity = (SELECT SUM(PurchaseLineQuantity) FROM PurchasedProducts
                         WHERE Purchases.PurchaseID = PurchasedProducts.PurchaseID);
UPDATE Purchases
SET TotalPurchaseAmount =(SELECT SUM(PurchaseLineAmount) FROM PurchasedProducts
                         WHERE Purchases.PurchaseID = PurchasedProducts.PurchaseID);
```

Task 5

Scripts to create Table structures

5 Task 5: Scripts to create Table structures

5.1 Suppliers Table

By using CREATE TABLE statement, “Suppliers” Table including SupplierID as Primary Key, domain constraint and integrity constraint is built successfully.

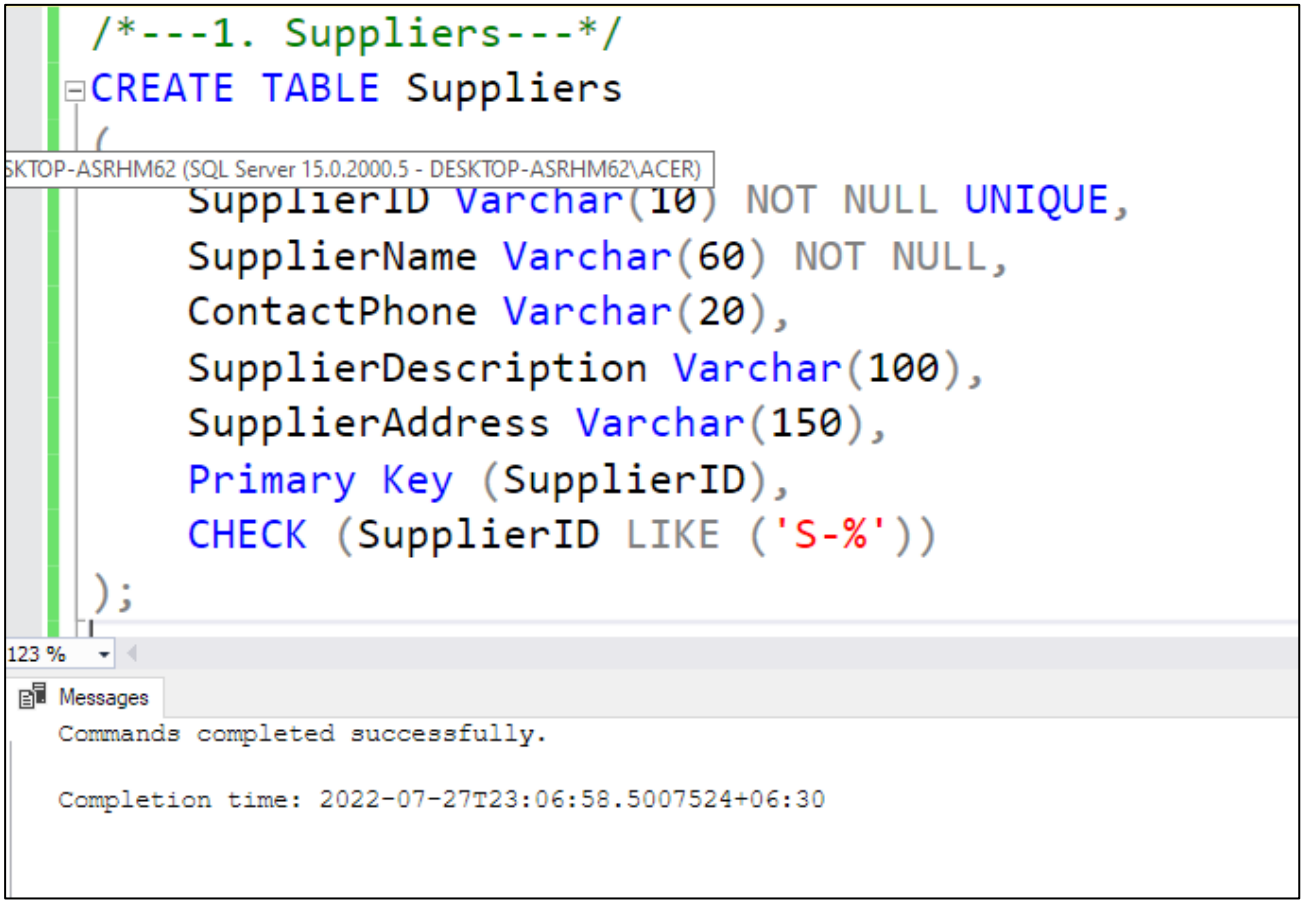


Figure 10: Create Query for Suppliers Table

Result/Output of Suppliers Table

The created Suppliers Table is then reviewed using SELECT statement.

Figure 11: Result from Create Query

5.2 Categories Table

By using CREATE TABLE statement, “Categories” Table including CategoryID as Primary Key, domain constraint and integrity constraint is built successfully.

```
/*---2. Categories---*/
CREATE TABLE Categories
(
    CategoryID Varchar(10) NOT NULL UNIQUE,
    CategoryName Varchar(40) NOT NULL,
    CategoryDescription Varchar(150),
    Primary Key (CategoryID),
    CHECK (CategoryID LIKE ('Cat-%'))
);
```

123 %

Messages

```
Commands completed successfully.

Completion time: 2022-07-27T23:08:14.6794339+06:30
```

Figure 12: Create Query for Categories Table

Result/Output of Categories Table

The created Categories Table is then reviewed using SELECT statement.

```
Select * FROM Categories;
```

120 %

Results | Messages

CategoryID	CategoryName	CategoryDescription

Figure 13: Result from Create Query

5.3 Customers Table

By using CREATE TABLE statement, “Customers” Table including CustomerID as Primary Key, domain constraint and integrity constraint is built successfully.

```
/*---3. Customers---*/
CREATE TABLE Customers
(
    CustomerID Varchar(10) NOT NULL UNIQUE,
    CustomerName Varchar(100),
    CustomerEmail Varchar(100),
    CustomerPhone Varchar(20),
    CustomerAddress Varchar(150),
    Primary Key (CustomerID),
    CHECK (CustomerID LIKE ('Cus-%') AND
    CustomerEmail LIKE ('%@gmail.com'))
);
```

123 %

Messages

```
Commands completed successfully.

Completion time: 2022-07-27T23:11:44.0414539+06:30
```

Figure 14: Create Query for Customers Table

Result/Output of Customers Table

The created Customers Table is then reviewed using SELECT statement.

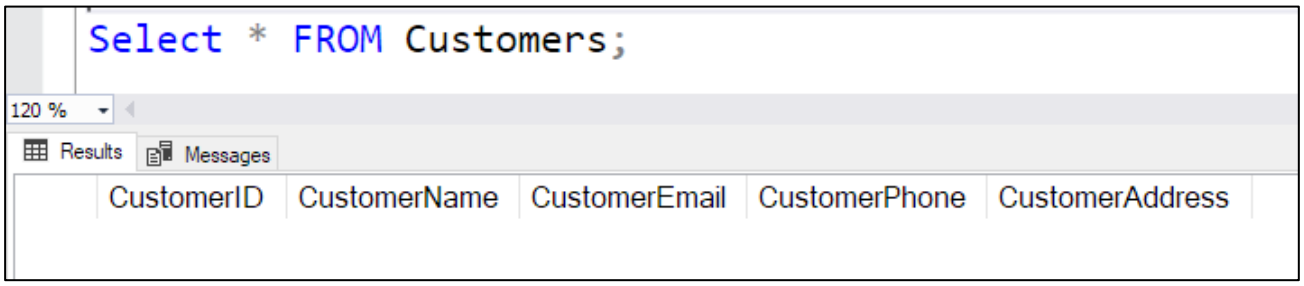


Figure 15: Result from Create Query

5.4 Employees

By using CREATE TABLE statement, “Employees” Table including EmployeeID as Primary Key, domain constraint and integrity constraint is built successfully.

```
/*---4. Employees---*/
CREATE TABLE Employees
(
    EmployeeID Varchar(10) NOT NULL UNIQUE,
    EmployeeName Varchar(100) NOT NULL,
    Gender Varchar(30),
    Position Varchar(30) NOT NULL,
    EmployeeEmail Varchar(100),
    EmployeeAddress Varchar(150),
    DateOfBirth DATE,
    Primary Key (EmployeeID),
    CHECK (EmployeeID LIKE ('E-%') AND
    Gender IN ('Male','Female','Others') AND
    EmployeeEmail LIKE ('%@gmail.com'))
);
```

```
123 %
Messages
Commands completed successfully.

Completion time: 2022-07-27T23:14:31.6411474+06:30
```

Figure 16: Create Query for Employees Table

Result/Output of Employees Table

The created Employees Table is then reviewed using SELECT statement.

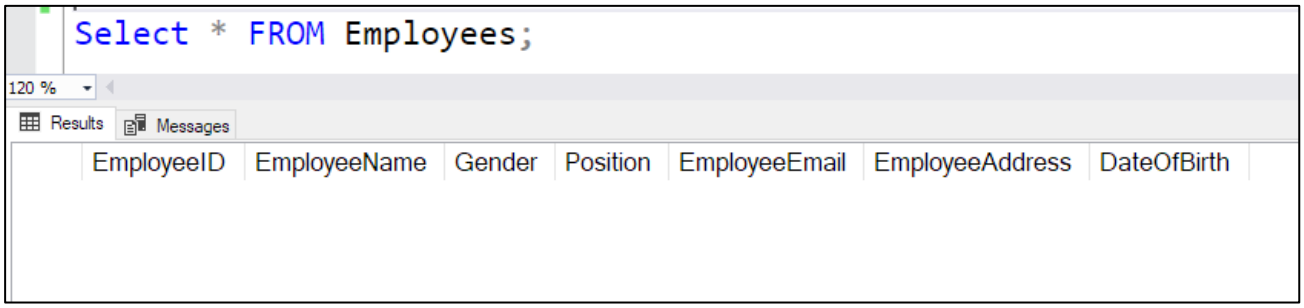


Figure 17: Result from Create Query

5.5 Promotions

By using CREATE TABLE statement, “Promotions” Table including PromotionID as Primary Key, domain constraint and integrity constraint is built successfully.

```
/*---5. Promotions---*/
CREATE TABLE Promotions
(
    PromotionID Varchar(15) NOT NULL UNIQUE,
    PromotionCode Varchar(20) NOT NULL,
    PromotionType Varchar(30),
    PromotionDescription Varchar(255),
    PromotionAmount Decimal(10,2),
    PromotionActive Varchar(20),
    PromotionStartDate DATE,
    PromotionEndDate DATE,
    Primary Key (PromotionID),
    CHECK (PromotionID LIKE ('Promo-%') AND
    PromotionType IN ('Discount Percentage','Discount Amount') AND
    PromotionActive IN ('Active','Expired'))
);
```

```
123 %
Messages
Commands completed successfully.

Completion time: 2022-07-27T23:16:28.3220906+06:30
```

Figure 18: Create Query for Promotions Table

Result/Output of Promotions Table

The created Promotions Table is then reviewed using SELECT statement.

```
Select * FROM Promotions;
```

120 %

Results | Messages

PromotionID	PromotionCode	PromotionType	PromotionDescription	PromotionAmount	PromotionActive	PromotionStartDate	PromotionEndDate

Figure 19: Result from Create Query

5.6 Delivery

By using CREATE TABLE statement, “Delivery” Table including DeliveryID as Primary Key, domain constraint and integrity constraint is built successfully.

```
/*---6. Delivery---*/
CREATE TABLE Delivery
(
    DeliveryID Varchar(15) NOT NULL UNIQUE,
    DeliveryDate DATE,
    Primary Key (DeliveryID),
    CHECK (DeliveryID LIKE ('Deli-%'))
);
```

120 %

Messages

```
Commands completed successfully.

Completion time: 2022-07-27T23:47:03.6304839+06:30
```

Figure 20: Create Query for Delivery Table

Result/Output of Delivery Table

The created Delivery Table is then reviewed using SELECT statement.

```
Select * FROM Delivery;
```

120 %

Results | Messages

DeliveryID	DeliveryDate

Figure 21: Result from Create Query

5.7 Purchases

By using CREATE TABLE statement, “Purchases” Table including PurchaseID as Primary Key, SupplierID as Foreign Key referencing related parent table together with Propagation constraints, domain constraint and integrity constraint is built successfully.

```
/*---7. Purchases---*/
CREATE TABLE Purchases
(
    PurchaseID Varchar(10) NOT NULL UNIQUE,
    PurchaseDate DATE NOT NULL,
    SupplierID Varchar(10) NOT NULL,
    Primary Key (PurchaseID),
    Foreign Key (SupplierID) REFERENCES Suppliers (SupplierID) ON UPDATE CASCADE ON DELETE NO ACTION,
    CHECK (PurchaseID LIKE ('Pur-%') AND
    SupplierID LIKE ('S-%'))
);
```

```
120 %
Messages
Commands completed successfully.

Completion time: 2022-07-27T23:48:52.0404164+06:30
```

Figure 22: Create Query for Purchases Table

Result/Output of Purchases Table

The created Purchases Table is then reviewed using SELECT statement.

```
Select * FROM Purchases;
```

120 %

Results | Messages

PurchaseID	PurchaseDate	SupplierID

Figure 23: Result from Create Query

5.8 Products

By using CREATE TABLE statement, “Products” Table including ProductID as Primary Key, CategoryID as Foreign Key referencing related parent table together with Propagation constraints, domain constraint and integrity constraint is built successfully.

```
/*---8. Products---*/
CREATE TABLE Products
(
    ProductID Varchar(10) NOT NULL UNIQUE,
    ProductName Varchar(100) NOT NULL,
    CategoryID Varchar(10) NOT NULL,
    ProductWeight Decimal(6,2),
    WeightUnit Varchar(10),
    UnitPrice Decimal(10,2),
    ExpiryDate DATE,
    Primary Key (ProductID),
    Foreign Key (CategoryID) REFERENCES Categories (CategoryID) ON UPDATE CASCADE ON DELETE NO ACTION,
    CHECK (ProductID LIKE ('P-%') AND
    CategoryID LIKE ('Cat-%') AND
    WeightUnit IN ('Viss','g','kg','mL','L'))
);
```

```
Messages
Commands completed successfully.

Completion time: 2022-07-27T23:51:58.2893790+06:30
```

Figure 24: Create Query for Products Table

Result/Output of Products Table

The created Products Table is then reviewed using SELECT statement.

```
Select * FROM Products;
```

Results | Messages

ProductID	ProductName	CategoryID	ProductWeight	WeightUnit	UnitPrice	ExpiryDate

Figure 25: Result from Create Query

5.9 PurchasedProducts

By using CREATE TABLE statement, “PurchasedProducts” Table including PurchaseID and ProductID as both Primary Key and Foreign Key referencing related parent tables together with Propagation constraints, domain constraint and integrity constraint is built successfully.

```
/*---9. PurchasedProducts---*/
CREATE TABLE PurchasedProducts
(
    PurchaseID Varchar(10) NOT NULL,
    ProductID Varchar(10) NOT NULL,
    PurchaseLineQuantity Integer,
    Primary Key (PurchaseID,ProductID),
    Foreign Key (PurchaseID) REFERENCES Purchases(PurchaseID) ON UPDATE CASCADE ON DELETE NO ACTION,
    Foreign Key (ProductID) REFERENCES Products(ProductID) ON UPDATE CASCADE ON DELETE NO ACTION,
    CHECK (PurchaseID LIKE ('Pur-%') AND
    ProductID LIKE ('P-%'))
);
```

120 %

Messages

```
Commands completed successfully.

Completion time: 2022-07-28T00:03:14.0773695+06:30
```

Figure 26: Create Query for PurchasedProducts Table

Result/Output of PurchasedProducts Table

The created PurchasedProducts Table is then reviewed using SELECT statement.

```
Select * FROM PurchasedProducts;
```

120 %

Results | Messages

PurchaseID	ProductID	PurchaseLineQuantity

Figure 27: Result from Create Query

5.10 Orders

By using CREATE TABLE statement, “Orders” Table including OrderID as Primary Key, CustomerID, PromotionID, EmployeeID and DeliveryID as Foreign Key referencing related parent tables together with Propagation constraints, domain constraint and integrity constraint is built successfully.

```
/*---10. Orders---*/
CREATE TABLE Orders
(
    OrderID Varchar(10) NOT NULL UNIQUE,
    CustomerID Varchar(10) NOT NULL,
    OrderDate DATE NOT NULL,
    PromotionID Varchar(15),
    PaymentAmount Decimal(10,2),
    PaymentType Varchar(20) DEFAULT 'Cash on Delivery',
    PaymentStatus Varchar(15) DEFAULT 'Not Paid',
    OrderStatus Varchar(20) DEFAULT 'Preparing',
    EmployeeID Varchar(10),
    DeliveryID Varchar(15),
    DeliveryAddress Varchar(255),
    Primary Key (OrderID),
    Foreign Key (CustomerID) REFERENCES Customers(CustomerID) ON UPDATE CASCADE ON DELETE NO ACTION,
    Foreign Key (PromotionID) REFERENCES Promotions (PromotionID) ON UPDATE CASCADE ON DELETE NO ACTION,
    Foreign Key (EmployeeID) REFERENCES Employees (EmployeeID) ON UPDATE CASCADE ON DELETE NO ACTION,
    Foreign Key (DeliveryID) REFERENCES Delivery (DeliveryID) ON UPDATE CASCADE ON DELETE NO ACTION,
    CHECK (OrderID LIKE ('Ord-%') AND
    PromotionID LIKE ('Promo-%') AND
    PaymentType IN ('Cash on Delivery','CBPay','KPay','Credit/Debit') AND
    PaymentStatus IN ('Paid','Not Paid') AND
    OrderStatus IN ('Preparing','On the way','Delivered') AND
    EmployeeID LIKE ('E-%') AND
    DeliveryID LIKE ('Deli-%'))
);
```

```
100 %
Messages
Commands completed successfully.

Completion time: 2022-07-28T00:05:36.1185377+06:30
```

Figure 28: Create Query for Orders Table

Result/Output of Orders Table

The created Orders Table is then reviewed using SELECT statement.

```
Select * FROM Orders;
```

120 % Results Messages

OrderID	CustomerID	OrderDate	PromotionID	PaymentAmount	PaymentType	PaymentStatus	OrderStatus	EmployeeID	DeliveryID	DeliveryAddress

Figure 29: Result from Create Query

5.11 OrderProducts

By using CREATE TABLE statement, “OrderProducts” Table including OrderID and ProductID as both Primary Key and Foreign Key referencing related parent tables together with Propagation constraints, domain constraint and integrity constraint is built successfully.

```
/*---11. OrderProducts---*/
CREATE TABLE OrderProducts
(
    OrderID Varchar(10) NOT NULL,
    ProductID Varchar(10) NOT NULL,
    OrderLineQuantity Integer,
    Primary Key (OrderID,ProductID),
    Foreign Key (OrderID) REFERENCES Orders (OrderID) ON UPDATE CASCADE ON DELETE NO ACTION,
    Foreign Key (ProductID) REFERENCES Products (ProductID) ON UPDATE CASCADE ON DELETE NO ACTION,
    CHECK (OrderID LIKE ('Ord-%') AND
    ProductID LIKE ('P-%'))
);
```

```
120 %
Messages
Commands completed successfully.

Completion time: 2022-07-28T00:07:06.7354730+06:30
```

Figure 30: Create Query for OrderProducs Table

Result/Output of OrderProducts Table

The created OrderProducts Table is then reviewed using SELECT statement.

```
Select * FROM OrderProducts;
```

120 % Results Messages

OrderID	ProductID	OrderLineQuantity

Figure 31: Result from Create Query

5.12 Explanation

In creation of Tables above, “CREATE TABLE” statement is used. Data types and sizes are added as planned in designed tables and data dictionary. Domain constraints such as checking prefix names are also added if necessary for data consistency. Primary keys are declared. Foreign keys together with Propagation constraints are declared. As for order of running, strong entities which are independent on other tables are executed and created first then the weak ones. The only issue encountered was that forgetting/ not noticing to double check whether correct database is selected in Available Databases drop down bar of Microsoft SQL Server, leading to create tables in undesired databases.

Task 6

Data Population

6 Task 6: Data Population

6.1 Suppliers Table Data insert script

By using INSERT INTO statement, data are inserted into columns of “Suppliers” table.

```
/*---1. Suppliers---*/
INSERT INTO Suppliers
        (SupplierID,SupplierName,ContactPhone,SupplierDescription,SupplierAddress)
VALUES  ('S-001','Shwe Si Daw','09565656565','Shwe Si Daw sells wet market products such as meats, vegetables, fruits','Tarmwe,
  Yangon'),
        ('S-002','Uncle Lay','09666666666','Uncle Lay sells Snacks and Beverages','Yankin, Yangon'),
        ('S-003','Top Meat','09564361237','Top Meat sells various types of meats and seafood','Ahlone, Yangon'),
        ('S-004','Aung Kabar','09324364784','Aung Kabar sells various cooking groceries such as rice, oil, condensed milk,
          coffeemix','Tarmwe, Yangon'),
        ('S-005','Three Nines','09635604826','Three Nines sells Wine bottles','Yankin, Yangon'),
        ('S-006','Mya Na Di','09669703970','Mya Na Di sells medicines and drugs','North Dagon, Yangon'),
        ('S-007','Aung Chan Thar','09798345137','Aung Chan Thar sells personal hygiene products','Tharkaeta, Yangon'),
        ('S-008','Pucci','09798523673','Pucci sells Pucci bakery products such as toast bread, cakes, puffs','Shwe Gone Dine,
          Yangon'),
        ('S-009','Baby Mall','09798351677','Baby Mall sells various products for Mothers and Child such as Diapers, Baby
          suppliments','Tarmwe, Yangon'),
        ('S-010','Two Brothers','09894127532','Two Brothers sells ready to eat food','Botahtaung, Yangon');
```

110 %

Messages

```
(10 rows affected)

Completion time: 2022-07-29T14:27:18.2746747+06:30
```

Figure 32: Insert data into Suppliers table

Result/Output of Suppliers Table

```
Select * FROM Suppliers;
```

110 %

Results | Messages

	SupplierID	SupplierName	ContactPhone	SupplierDescription	SupplierAddress
1	S-001	Shwe Si Daw	09565656565	Shwe Si Daw sells wet market products such as me...	Tarmwe, Yangon
2	S-002	Uncle Lay	09666666666	Uncle Lay sells Snacks and Beverages	Yankin, Yangon
3	S-003	Top Meat	09564361237	Top Meat sells various types of meats and seafood	Ahlone, Yangon
4	S-004	Aung Kabar	09324364784	Aung Kabar sells various cooking groceries such as ...	Tarmwe, Yangon
5	S-005	Three Nines	09635604826	Three Nines sells Wine bottles	Yankin, Yangon
6	S-006	Mya Na Di	09669703970	Mya Na Di sells medicines and drugs	North Dagon, Yangon
7	S-007	Aung Chan Thar	09798345137	Aung Chan Thar sells personal hygiene products	Tharkaeta, Yangon
8	S-008	Pucci	09798523673	Pucci sells Pucci bakery products such as toast bre...	Shwe Gone Dine, Yangon
9	S-009	Baby Mall	09798351677	Baby Mall sells various products for Mothers and Ch...	Tarmwe, Yangon
10	S-010	Two Brothers	09894127532	Two Brothers sells ready to eat food	Botahtaung, Yangon

Figure 33: Result from Insert Query

6.2 Categories Table Data insert script

By using INSERT INTO statement, data are inserted into columns of “Categories” table.

```
INSERT INTO Categories
        (CategoryID,CategoryName,CategoryDescription)
VALUES  ('Cat-001','Meats and Seafood','Includes chicken, pork, beef, seafood, etc'),
        ('Cat-002','Fruits and Vegetables','Includes fruits and vegetables such as grapes, apples, oranges, potato, cabbages,
          gourd, radish, etc'),
        ('Cat-003','Snacks and Beverages','Includes snacks such as biscuits, crackers, noodles and beverages such as orange juices,
          coca cola, etc'),
        ('Cat-004','Wine','Includes various kinds of wine bottles'),
        ('Cat-005','Ready To Eat','Includes ready to eat food such as fried dumplings, etc'),
        ('Cat-006','Mother and Childcare','Includes products for mothers and their child such as suppliments, diapers, etc'),
        ('Cat-007','Personal Hygiene','Includes personal hygiene products such as shampoo, hand wash, soaps, etc'),
        ('Cat-008','Medicines and Drugs','Includes medicines and drugs like Biogesic, Flumox, etc'),
        ('Cat-009','Pucci Cake and Bread','Includes products from Pucci such as English toasts, Cakes, etc'),
        ('Cat-010','Cooking Groceries','Includes daily necessary products such as rice, oil, salt, fish sauce, MSG, etc');
```

110 %

Messages

```
(10 rows affected)

Completion time: 2022-07-29T14:30:14.8045161+06:30
```

Figure 34: Insert data into Categories table

Result/Output of Categories Table

```
Select * FROM Categories;
```

110 %

Results | Messages

	CategoryID	CategoryName	CategoryDescription
1	Cat-001	Meats and Seafood	Includes chicken, pork, beef, seafood, etc
2	Cat-002	Fruits and Vegetables	Includes fruits and vegetables such as grapes, apples, oranges, potato, cabbages, gourd, radish, etc
3	Cat-003	Snacks and Beverages	Includes snacks such as biscuits, crackers, noodles and beverages such as orange juices, coca cola, etc
4	Cat-004	Wine	Includes various kinds of wine bottles
5	Cat-005	Ready To Eat	Includes ready to eat food such as fried dumplings, etc
6	Cat-006	Mother and Childcare	Includes products for mothers and their child such as suppliments, diapers, etc
7	Cat-007	Personal Hygiene	Includes personal hygiene products such as shampoo, hand wash, soaps, etc
8	Cat-008	Medicines and Drugs	Includes medicines and drugs like Biogesic, Flumox, etc
9	Cat-009	Pucci Cake and Bread	Includes products from Pucci such as English toasts, Cakes, etc
10	Cat-010	Cooking Groceries	Includes daily necessary products such as rice, oil, salt, fish sauce, MSG, etc

Figure 35: Result from Insert Query

6.3 Customers Table Data insert script

By using INSERT INTO statement, data are inserted into columns of “Customers” table.

```
INSERT INTO Customers
        (CustomerID,CustomerName,CustomerEmail,CustomerPhone,CustomerAddress)
VALUES  ('Cus-001','Maung Maung','MaungMaung2@gmail.com','09454545454','No.19, Room 301, 123th street, MingalarTaungNyunt Tsp,
  Yangon'),
        ('Cus-002','Kaung Khant','KaungKhant2kk@gmail.com','09456712345','No.114, Room 201, 135th street, MingalarTaungNyunt Tsp,
          Yangon'),
        ('Cus-003','Ci Ci','CiCi289@gmail.com','09756970488','No.15, Room 201, Aung Tha Pyay street, MingalarTaungNyunt Tsp,
          Yangon'),
        ('Cus-004','Aung Khant','AungKhant225@gmail.com','09798352794','No.31, Room 501, Aung Tha Pyay street, MingalarTaungNyunt
          Tsp, Yangon'),
        ('Cus-005','Arkar Bhone','AkBhone666@gmail.com','09798432643','No.69, Ground Floor, Witt Kyaung street, Yay Kyaw Tsp,
          Yangon'),
        ('Cus-006','Ingyin Khin','IngyinK612@gmail.com','09894102507','No.80, Room 301, Witt Kyaung street, Yay Kyaw Tsp, Yangon'),
        ('Cus-007','Mya Layy','Myalayy222@gmail.com','09894111245','No.55, Room 401, Aww Bar street, Kyauk Myaung Tsp, Yangon'),
        ('Cus-008','Myint Than','Umyintthan683@gmail.com','09798562684','No.134, Ground Floor, Kyi Taw street, Pazuntaung Tsp,
          Yangon'),
        ('Cus-009','Kyaw Shane','KyawS111@gmail.com','09894110503','No.64, Room 201, Than street, Hlaing Tsp, Yangon'),
        ('Cus-010','Aye Thin Khine','ATKhine999@gmail.com','09756848392','No.101, Ground Floor, 51st street, Botahtaung Tsp,
          Yangon');
```

```
(10 rows affected)

Completion time: 2022-07-29T14:33:10.5373058+06:30
```

Figure 36: Insert data into Customers table

Result/Output of Customers Table

```
SELECT * FROM Customers;
```

	CustomerID	CustomerName	CustomerEmail	CustomerPhone	CustomerAddress
1	Cus-001	Maung Maung	MaungMaung2@gmail.com	09454545454	No.19, Room 301, 123th street, MingalarTaungNyunt Tsp, Yangon
2	Cus-002	Kaung Khant	KaungKhant2kk@gmail.com	09456712345	No.114, Room 201, 135th street, MingalarTaungNyunt Tsp, Yangon
3	Cus-003	Ci Ci	CiCi289@gmail.com	09756970488	No.15, Room 201, Aung Tha Pyay street, MingalarTaungNyunt Tsp, Yangon
4	Cus-004	Aung Khant	AungKhant225@gmail.com	09798352794	No.31, Room 501, Aung Tha Pyay street, MingalarTaungNyunt Tsp, Yangon
5	Cus-005	Arkar Bhone	AkBhone666@gmail.com	09798432643	No.69, Ground Floor, Witt Kyaung street, Yay Kyaw Tsp, Yangon
6	Cus-006	Ingyin Khin	IngyinK612@gmail.com	09894102507	No.80, Room 301, Witt Kyaung street, Yay Kyaw Tsp, Yangon
7	Cus-007	Mya Layy	Myalayy222@gmail.com	09894111245	No.55, Room 401, Aww Bar street, Kyauk Myaung Tsp, Yangon
8	Cus-008	Myint Than	Umyintthan683@gmail.com	09798562684	No.134, Ground Floor, Kyi Taw street, Pazuntaung Tsp, Yangon
9	Cus-009	Kyaw Shane	KyawS111@gmail.com	09894110503	No.64, Room 201, Than street, Hlaing Tsp, Yangon
10	Cus-010	Aye Thin Khine	ATKhine999@gmail.com	09756848392	No.101, Ground Floor, 51st street, Botahtaung Tsp, Yangon

Figure 37: Result from Insert Query

6.4 Employees Table Data insert script

By using INSERT INTO statement, data are inserted into columns of “Employees” table.

```
INSERT INTO Employees
        (EmployeeID,EmployeeName,Gender,Position,EmployeeEmail,EmployeeAddress,DateOfBirth)
VALUES  ('E-001','Ei Chaw','Female','Cashier','EiChaw21@gmail.com','No.64, Room 401, Thidar street, MingalarTaungNyunt Tsp,
  Yangon','1989-09-21'),
        ('E-002','Arkar Min','Male','Cashier','ArkarM16@gmail.com','No.70, Room 301, U Pone Nya street, North Dagon Tsp,
          Yangon','1995-10-16'),
        ('E-003','Saw Nanda','Male','Book Keeper','SanNanda211@gmail.com','No.301, Ground Floor, 47th street, Botahtaung Tsp,
          Yangon','1989-12-01'),
        ('E-004','Kyi Phyu','Female','Warehouse Assistant','KyiPhyuKP221@gmail.com','No. 201, Room 301, Hnin Si street,
          MingalarTaungNyunt Tsp, Yangon','1980-06-25'),
        ('E-005','Nway Nway','Female','Accountant','Nway232@gmail.com','No.98, Room 501, Sapal street, MingalarTaungNyunt Tsp,
          Yangon','1986-11-23'),
        ('E-006','Kyaw Htet Aung','Male','Warehouse Assistant','kyawhtetaung29@gmail.com','No.120, Room 201, 124th street,
          MingalarTaungNyunt Tsp, Yangon','1992-07-18'),
        ('E-007','Kaung Htet','Male','Inventory Manager','KaungH212@gmail.com','No.43, Room 401, 51st street, Botahtaung Tsp,
          Yangon','1987-03-11'),
        ('E-008','Eaindray Khin','Female','Accountant','Eaindraykhin15@gmail.com','No.15, Room 301, 124th street,
          MingalarTaungNyunt Tsp, Yangon','1989-05-15'),
        ('E-009','Kyaw Khine','Male','Manager','KyawKhineM19@gmail.com','No.31, Ground Floor, 37th street, PanSoeDan Tsp,
          Yangon','1980-04-12'),
        ('E-010','Kaung Zaw','Male','Founder','KaungZaw111@gmail.com','No.35, Ground Floor, Aung Tha Pyay street,
          MingalarTaungNyunt Tsp, Yangon','1976-01-11');
```

110 %

Messages

```
(10 rows affected)

Completion time: 2022-07-29T14:35:14.3070610+06:30
```

Figure 38: Insert data into Employees table

Result/Output of Employees Table

```
SELECT * FROM Employees;
```

110 %

Results | Messages

	EmployeeID	EmployeeName	Gender	Position	EmployeeEmail	EmployeeAddress	DateOfBirth
1	E-001	Ei Chaw	Female	Cashier	EiChaw21@gmail.com	No.64, Room 401, Thidar street, MingalarTaungNyunt Tsp, Yangon	1989-09-21
2	E-002	Arkar Min	Male	Cashier	ArkarM16@gmail.com	No.70, Room 301, U Pone Nya street, North Dagon Tsp, Yangon	1995-10-16
3	E-003	Saw Nanda	Male	Book Keeper	SanNanda211@gmail.com	No.301, Ground Floor, 47th street, Botahtaung Tsp, Yangon	1989-12-01
4	E-004	Kyi Phyu	Female	Warehouse Assistant	KyiPhyuKP221@gmail.com	No. 201, Room 301, Hnin Si street, MingalarTaungNyunt Tsp, Yangon	1980-06-25
5	E-005	Nway Nway	Female	Accountant	Nway232@gmail.com	No.98, Room 501, Sapal street, MingalarTaungNyunt Tsp, Yangon	1986-11-23
6	E-006	Kyaw Htet Aung	Male	Warehouse Assistant	kyawhtetaung29@gmail.com	No.120, Room 201, 124th street, MingalarTaungNyunt Tsp, Yangon	1992-07-18
7	E-007	Kaung Htet	Male	Inventory Manager	KaungH212@gmail.com	No.43, Room 401, 51st street, Botahtaung Tsp, Yangon	1987-03-11
8	E-008	Eaindray Khin	Female	Accountant	Eaindraykhin15@gmail.com	No.15, Room 301, 124th street, MingalarTaungNyunt Tsp, Yangon	1989-05-15
9	E-009	Kyaw Khine	Male	Manager	KyawKhineM19@gmail.com	No.31, Ground Floor, 37th street, PanSoeDan Tsp, Yangon	1980-04-12
10	E-010	Kaung Zaw	Male	Founder	KaungZaw111@gmail.com	No.35, Ground Floor, Aung Tha Pyay street, MingalarTaungNyunt Tsp, Yangon	1976-01-11

Figure 39: Result from Insert Query

6.5 Promotions Table Data insert script

By using INSERT INTO statement, data are inserted into columns of “Promotions” table.

```
/*---5. Promotions---*/
INSERT INTO Promotions
    (PromotionID,PromotionCode,PromotionType,PromotionDescription,PromotionAmount,PromotionActive,PromotionStartDate,PromotionEndDa
     te)
VALUES  ('Promo-001','B4Uoff15%','Discount Percentage','Customers whose order subtotal is 40,000 or above 40,000 kyats can use this
   promotion code to be benefitted with 15% discount over order subtotal amount. This code is not limited with numbers of
  orders',0.15,'Expired','2022-06-13','2022-06-18'),
        ('Promo-002','B4Uoff10%','Discount Percentage','Customers whose order subtotal is 30,000 or above 30,000 kyats can use this
          promotion code to be benefitted with 10% discount over order subtotal amount. This code is not limited with numbers of
         orders',0.10,'Active','2022-06-05','2022-07-05'),
        ('Promo-003','B4Uoff5%','Discount Percentage','Customers whose order subtotal is 15,000 or above 15,000 kyats can use this
         promotion code to be benefitted with 5% discount over order subtotal amount. This code is not limited with numbers of
         orders',0.05,'Active','2022-06-05','2022-07-05');
```

```
(3 rows affected)

Completion time: 2022-07-29T14:37:57.9289027+06:30
```

Figure 40: Insert data into Promotions table

Result/Output of Promotions Table

```
SELECT * FROM Promotions;
```

	PromotionID	PromotionCode	PromotionType	PromotionDescription	PromotionAmount	PromotionActive	PromotionStartDate	PromotionEndDate
1	Promo-001	B4Uoff15%	Discount Percentage	Customers whose order subtotal is 40,000 or above 40,000 kyats ca...	0.15	Expired	2022-06-13	2022-06-18
2	Promo-002	B4Uoff10%	Discount Percentage	Customers whose order subtotal is 30,000 or above 30,000 kyats ca...	0.10	Active	2022-06-05	2022-07-05
3	Promo-003	B4Uoff5%	Discount Percentage	Customers whose order subtotal is 15,000 or above 15,000 kyats ca...	0.05	Active	2022-06-05	2022-07-05

Figure 41: Result from Insert Query

6.6 Purchases Table Data insert script

By using INSERT INTO statement, data are inserted into columns of “Purchases” table.

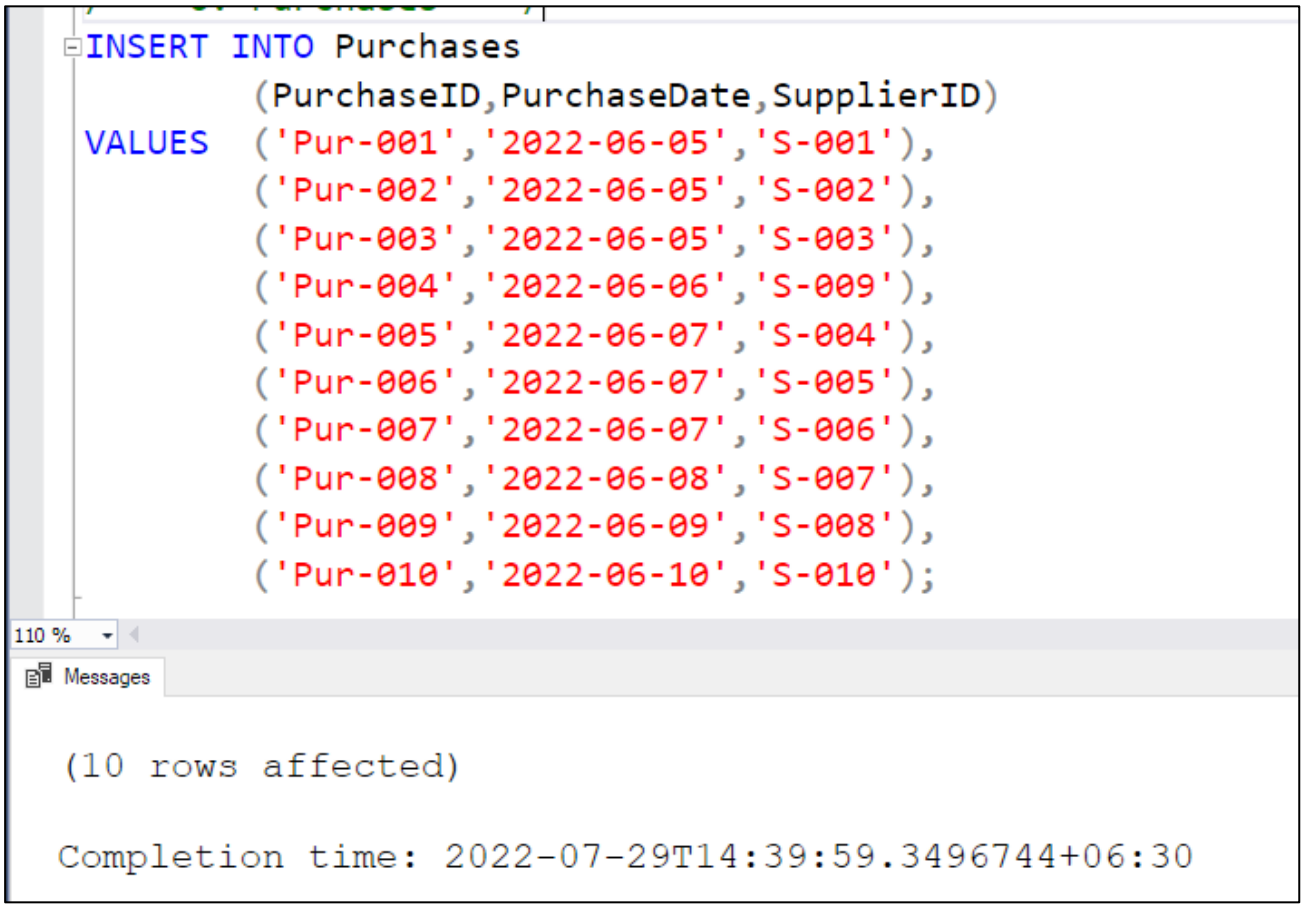


Figure 42: Insert data into Purchases table

Result/Output of Purchases Table

```
SELECT * FROM Purchases;
```

110 %

Results | Messages

	PurchaseID	PurchaseDate	SupplierID
1	Pur-001	2022-06-05	S-001
2	Pur-002	2022-06-05	S-002
3	Pur-003	2022-06-05	S-003
4	Pur-004	2022-06-06	S-009
5	Pur-005	2022-06-07	S-004
6	Pur-006	2022-06-07	S-005
7	Pur-007	2022-06-07	S-006
8	Pur-008	2022-06-08	S-007
9	Pur-009	2022-06-09	S-008
10	Pur-010	2022-06-10	S-010

Figure 43: Result from Insert Query

6.7 Products Table Data insert script

By using INSERT INTO statement, data are inserted into columns of “Products” table.

```
/*---7. Products---*/
INSERT INTO Products
        (ProductID,ProductName,CategoryID,ProductWeight,WeightUnit,UnitPrice,ExpiryDate)
VALUES  ('P-001','Chicken Breast','Cat-001',0.3,'Viss',4000,NULL),
        ('P-002','Burmese Whole Chicken','Cat-001',0.5,'Viss',6000,NULL),
        ('P-003','Pork 3 Layers','Cat-001',300,'g',3200,NULL),
        ('P-004','Chicken Thigh with skin','Cat-001',0.5,'Viss',2500,NULL),
        ('P-005','Danisa Cookies','Cat-003',454,'g',11000,'2024-06-15'),
        ('P-006','Juicy Orange','Cat-003',750,'mL',3000,'2024-12-30'),
        ('P-007','Ovaltine Chocolate Bottle','Cat-003',400,'g',9000,'2024-12-30'),
        ('P-008','Pork Head','Cat-001',0.5,'Viss',6000,NULL),
        ('P-009','Pork Tenderloin','Cat-001',0.5,'Viss',8000,NULL),
        ('P-010','Pork Ribs Special','Cat-001',0.5,'Viss',9500,NULL),
        ('P-011','Chicken Breast','Cat-001',0.3,'Viss',4500,NULL),
        ('P-012','Burmese Whole Chicken','Cat-001',0.5,'Viss',6200,NULL),
        ('P-013','Cabbage 1 pack','Cat-002',NULL,NULL,900,NULL),
        ('P-014','Banana 1 pack','Cat-002',NULL,NULL,2200,NULL),
        ('P-015','Grapes','Cat-002',0.3,'Viss',6000,NULL),
        ('P-016','DingWang Rice Cookies','Cat-003',360,'g',2250,'2024-06-16'),
        ('P-017','Shoon Fatt Corn Biscuit Box','Cat-003',1.5,'kg',12500,'2024-12-12'),
        ('P-018','Nestle Rice Chicken Nutrition Powder','Cat-006',250,'g',6000,'2023-12-01'),
        ('P-019','Komodo Tissues 70pcs','Cat-006',NULL,NULL,4000,NULL),
        ('P-020','Babymild Cherry Blossom Cream','Cat-006',180,'mL',5100,'2024-01-14'),
110 %
Messages

(55 rows affected)

Completion time: 2022-07-29T14:42:57.8504389+06:30
```

Figure 44: Insert data into Products table

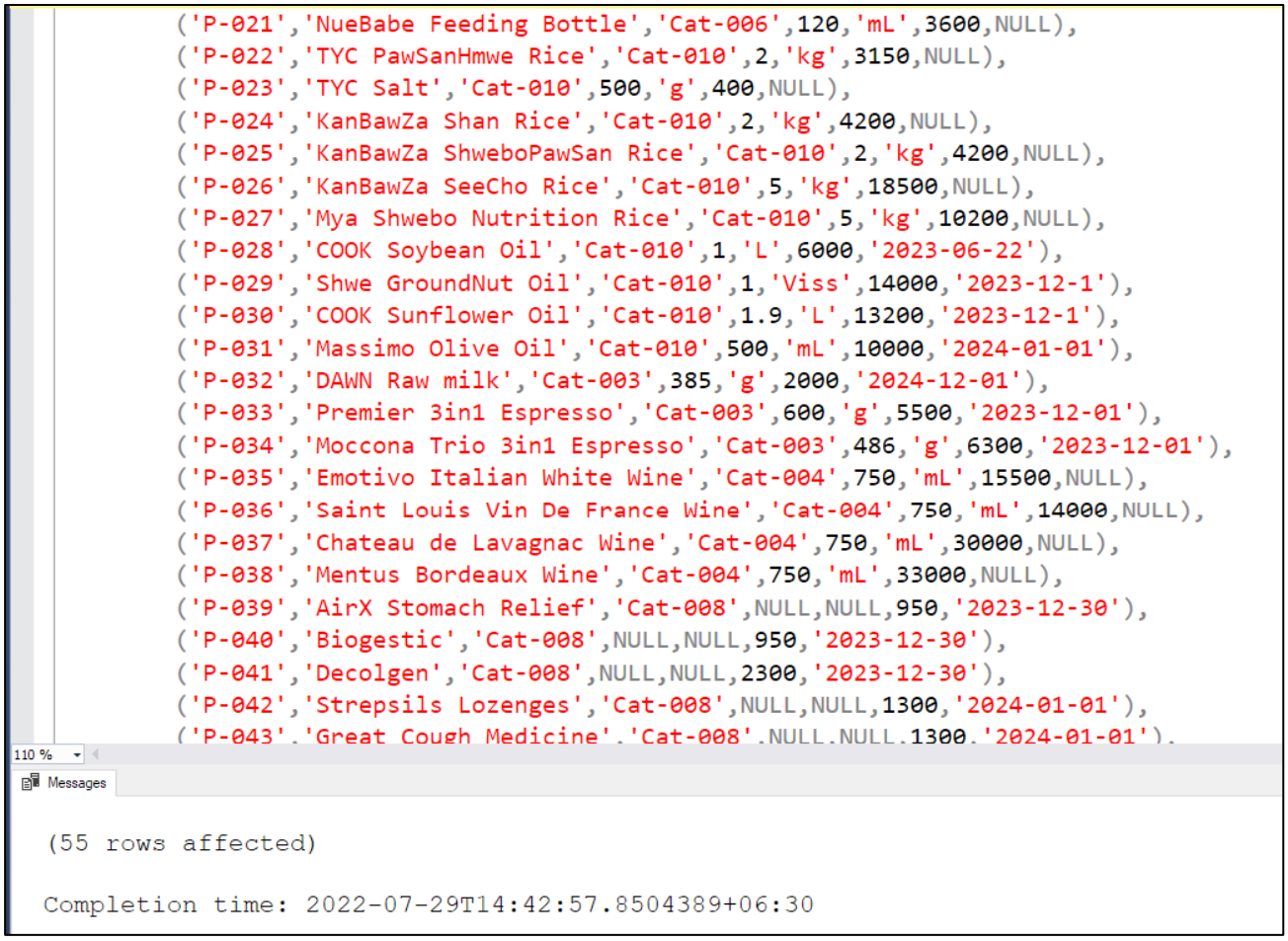


Figure 45: Insert data into Products table

```
        ('P-037','Chateau de Lavagnac Wine','Cat-004',750,'mL',30000,NULL),
        ('P-038','Mentus Bordeaux Wine','Cat-004',750,'mL',33000,NULL),
        ('P-039','AirX Stomach Relief','Cat-008',NULL,NULL,950,'2023-12-30'),
        ('P-040','Biogestic','Cat-008',NULL,NULL,950,'2023-12-30'),
        ('P-041','Decolgen','Cat-008',NULL,NULL,2300,'2023-12-30'),
        ('P-042','Strepsils Lozenges','Cat-008',NULL,NULL,1300,'2024-01-01'),
        ('P-043','Great Cough Medicine','Cat-008',NULL,NULL,1300,'2024-01-01'),
        ('P-044','Colgate Salt Charcoal Toothpaste','Cat-007',150,'g',3500,'2025-12-30'),
        ('P-045','Colgate Max Fresh Mint Toothpaste','Cat-007',160,'g',3500,'2025-12-30'),
        ('P-046','Clear Shampoo Men','Cat-007',170,'mL',3800,'2025-12-30'),
        ('P-047','Clear Shampoo Women','Cat-007',170,'mL',3800,'2025-12-30'),
        ('P-048','LifeBuyoy Lemon Liquid Soaps','Cat-007',500,'mL',6700,'2025-12-30'),
        ('P-049','Pucci Toast Bread Slices','Cat-009',150,'g',1000,'2022-07-28'),
        ('P-050','Pucci Mayonese Cheese Bread','Cat-009',200,'g',2000,'2022-07-28'),
        ('P-051','Pucci Mini Cakes','Cat-009',200,'g',2000,'2022-08-05'),
        ('P-052','Fried Fish balls','Cat-005',0.20,'Viss',4500,'2022-10-01'),
        ('P-053','Fried Spicy Fish balls','Cat-005',0.20,'Viss',3000,'2022-10-01'),
        ('P-054','Fried Pork Fish balls','Cat-005',0.20,'Viss',4400,'2022-08-15'),
        ('P-055','Fried Fish Pops','Cat-005',0.20,'Viss',5500,'2022-08-15');
110 %
Messages

(55 rows affected)

Completion time: 2022-07-29T14:42:57.8504389+06:30
```

Figure 46: Insert data into Products table

Result/Output of Products Table

```
SELECT * FROM Products;
```

110 %

Results | Messages

	ProductID	ProductName	CategoryID	ProductWeight	WeightUnit	UnitPrice	ExpiryDate
1	P-001	Chicken Breast	Cat-001	0.30	Viss	4000.00	NULL
2	P-002	Burmese Whole Chicken	Cat-001	0.50	Viss	6000.00	NULL
3	P-003	Pork 3 Layers	Cat-001	300.00	g	3200.00	NULL
4	P-004	Chicken Thigh with skin	Cat-001	0.50	Viss	2500.00	NULL
5	P-005	Danisa Cookies	Cat-003	454.00	g	11000.00	2024-06-15
6	P-006	Juicy Orange	Cat-003	750.00	mL	3000.00	2024-12-30
7	P-007	Ovaltine Chocolate Bottle	Cat-003	400.00	g	9000.00	2024-12-30
8	P-008	Pork Head	Cat-001	0.50	Viss	6000.00	NULL
9	P-009	Pork Tenderloin	Cat-001	0.50	Viss	8000.00	NULL
10	P-010	Pork Ribs Special	Cat-001	0.50	Viss	9500.00	NULL
11	P-011	Chicken Breast	Cat-001	0.30	Viss	4500.00	NULL
12	P-012	Burmese Whole Chicken	Cat-001	0.50	Viss	6200.00	NULL
13	P-013	Cabbage 1 pack	Cat-002	NULL	NULL	900.00	NULL
14	P-014	Banana 1 pack	Cat-002	NULL	NULL	2200.00	NULL
15	P-015	Grapes	Cat-002	0.30	Viss	6000.00	NULL
16	P-016	DingWang Rice Cookies	Cat-003	360.00	g	2250.00	2024-06-16
17	P-017	Shoon Fatt Corn Biscuit Box	Cat-003	1.50	kg	12500.00	2024-12-12
18	P-018	Nestle Rice Chicken Nutrition Powder	Cat-006	250.00	g	6000.00	2023-12-01
19	P-019	Komodo Tissues 70pcs	Cat-006	NULL	NULL	4000.00	NULL
20	P-020	Babymild Cherry Blossom Cream	Cat-006	180.00	mL	5100.00	2024-01-14
21	P-021	NueBabe Feeding Bottle	Cat-006	120.00	mL	3600.00	NULL
22	P-022	TYC PawSanHmwe Rice	Cat-010	2.00	kg	3150.00	NULL
23	P-023	TYC Salt	Cat-010	500.00	g	400.00	NULL
24	P-024	KanBawZa Shan Rice	Cat-010	2.00	kg	4200.00	NULL
25	P-025	KanBawZa ShweboPawSan Rice	Cat-010	2.00	kg	4200.00	NULL
26	P-026	KanBawZa SeeCho Rice	Cat-010	5.00	kg	18500.00	NULL
27	P-027	Mya Shwebo Nutrition Rice	Cat-010	5.00	kg	10200.00	NULL
28	P-028	COOK Soybean Oil	Cat-010	1.00	L	6000.00	2023-06-22
29	P-029	Shwe GroundNut Oil	Cat-010	1.00	Viss	14000.00	2023-12-01
30	P-030	COOK Sunflower Oil	Cat-010	1.90	L	13200.00	2023-12-01

Query executed successfully.

Figure 47: Result from Insert Query

```
SELECT * FROM Products;
```

110 %

Results | Messages

	ProductID	ProductName	CategoryID	ProductWeight	WeightUnit	UnitPrice	ExpiryDate
27	P-027	Mya Shwebo Nutrition Rice	Cat-010	5.00	kg	10200.00	NULL
28	P-028	COOK Soybean Oil	Cat-010	1.00	L	6000.00	2023-06-22
29	P-029	Shwe GroundNut Oil	Cat-010	1.00	Viss	14000.00	2023-12-01
30	P-030	COOK Sunflower Oil	Cat-010	1.90	L	13200.00	2023-12-01
31	P-031	Massimo Olive Oil	Cat-010	500.00	mL	10000.00	2024-01-01
32	P-032	DAWN Raw milk	Cat-003	385.00	g	2000.00	2024-12-01
33	P-033	Premier 3in1 Espresso	Cat-003	600.00	g	5500.00	2023-12-01
34	P-034	Moccona Trio 3in1 Espresso	Cat-003	486.00	g	6300.00	2023-12-01
35	P-035	Emotivo Italian White Wine	Cat-004	750.00	mL	15500.00	NULL
36	P-036	Saint Louis Vin De France Wine	Cat-004	750.00	mL	14000.00	NULL
37	P-037	Chateau de Lavagnac Wine	Cat-004	750.00	mL	30000.00	NULL
38	P-038	Mentus Bordeaux Wine	Cat-004	750.00	mL	33000.00	NULL
39	P-039	AirX Stomach Relief	Cat-008	NULL	NULL	950.00	2023-12-30
40	P-040	Biogestic	Cat-008	NULL	NULL	950.00	2023-12-30
41	P-041	Decolgen	Cat-008	NULL	NULL	2300.00	2023-12-30
42	P-042	Strepsils Lozenges	Cat-008	NULL	NULL	1300.00	2024-01-01
43	P-043	Great Cough Medicine	Cat-008	NULL	NULL	1300.00	2024-01-01
44	P-044	Colgate Salt Charcoal Toothpaste	Cat-007	150.00	g	3500.00	2025-12-30
45	P-045	Colgate Max Fresh Mint Toothpaste	Cat-007	160.00	g	3500.00	2025-12-30
46	P-046	Clear Shampoo Men	Cat-007	170.00	mL	3800.00	2025-12-30
47	P-047	Clear Shampoo Women	Cat-007	170.00	mL	3800.00	2025-12-30
48	P-048	LifeBuyoy Lemon Liquid Soaps	Cat-007	500.00	mL	6700.00	2025-12-30
49	P-049	Pucci Toast Bread Slices	Cat-009	150.00	g	1000.00	2022-07-28
50	P-050	Pucci Mayonese Cheese Bread	Cat-009	200.00	g	2000.00	2022-07-28
51	P-051	Pucci Mini Cakes	Cat-009	200.00	g	2000.00	2022-08-05
52	P-052	Fried Fish balls	Cat-005	0.20	Viss	4500.00	2022-10-01
53	P-053	Fried Spicy Fish balls	Cat-005	0.20	Viss	3000.00	2022-10-01
54	P-054	Fried Pork Fish balls	Cat-005	0.20	Viss	4400.00	2022-08-15
55	P-055	Fried Fish Pops	Cat-005	0.20	Viss	5500.00	2022-08-15

Query executed successfully.

Figure 48: Result from Insert Query

6.8 PurchasedProducts Table Data insert script

By using INSERT INTO statement, data are inserted into columns of “PurchasedProducts” table.

```
/*---8. PurchasedProducts---*/
INSERT INTO PurchasedProducts
        (PurchaseID,ProductID,PurchaseLineQuantity)
VALUES  ('Pur-001','P-001',40),
        ('Pur-001','P-002',25),
        ('Pur-001','P-003',25),
        ('Pur-001','P-004',30),
        ('Pur-001','P-008',10),
        ('Pur-001','P-013',20),
        ('Pur-001','P-014',20),
        ('Pur-001','P-015',20),
        ('Pur-002','P-005',40),
        ('Pur-002','P-006',30),
        ('Pur-002','P-007',25),
        ('Pur-002','P-016',30),
        ('Pur-002','P-017',30),
        ('Pur-003','P-009',30),
        ('Pur-003','P-010',25),
        ('Pur-003','P-011',25),
        ('Pur-003','P-012',30),
        ('Pur-004','P-018',30),
        ('Pur-004','P-019',35),
        ('Pur-004','P-020',20),
        ('Pur-004','P-021',15),
        ('Pur-005','P-022',20),
        ('Pur-005','P-023',30),
110 %
Messages

(55 rows affected)

Completion time: 2022-07-29T14:46:46.9238425+06:30
```

Figure 49: Insert data into PurchasedProducts table

Figure 50: Insert data into PurchasedProducts table

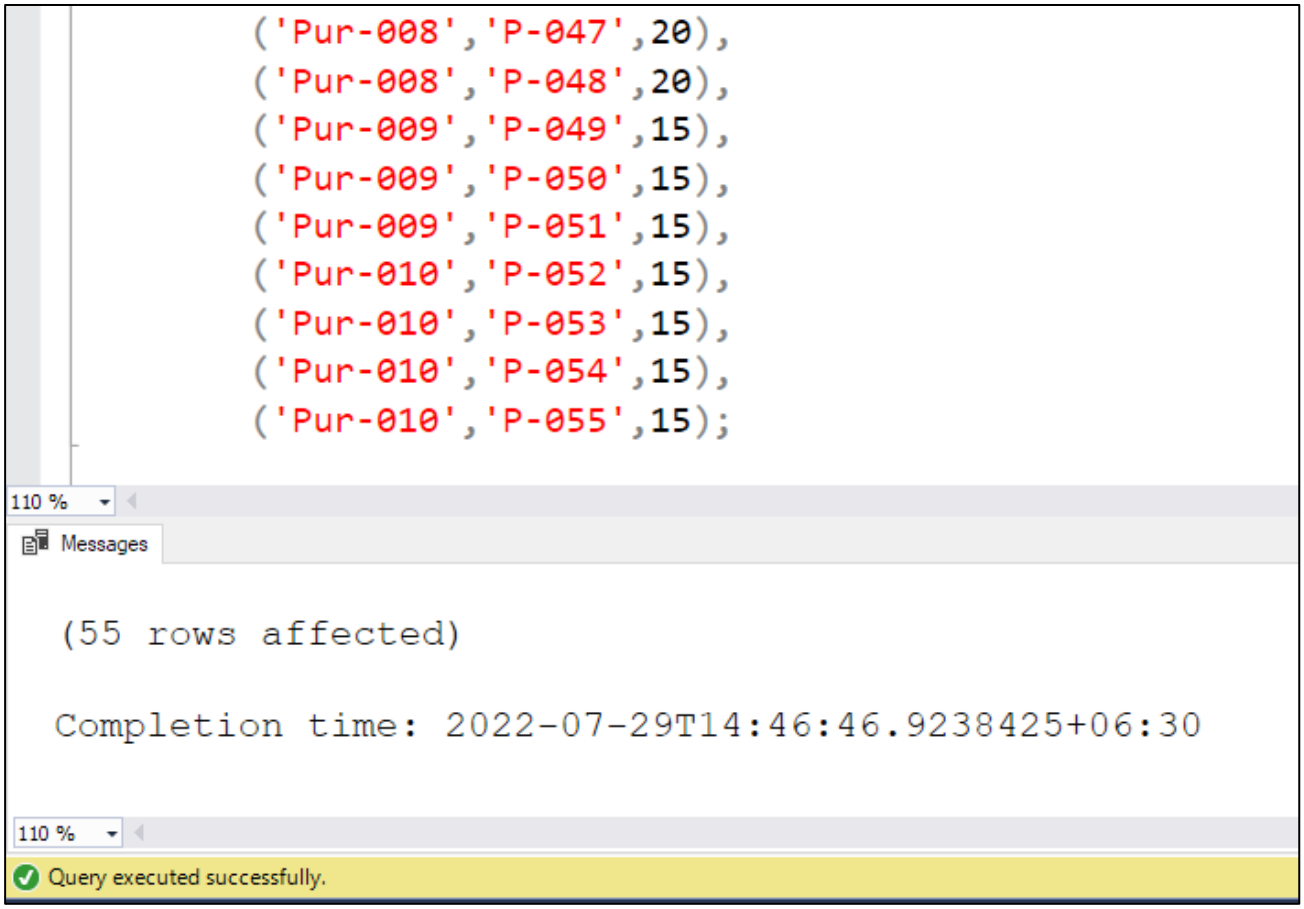


Figure 51: Insert data into PurchasedProducts table

Result/Output of PurchasedProducts Table

```
SELECT * FROM PurchasedProducts;
```

110 %

Results | Messages

	PurchaseID	ProductID	PurchaseLineQuantity
1	Pur-001	P-001	40
2	Pur-001	P-002	25
3	Pur-001	P-003	25
4	Pur-001	P-004	30
5	Pur-001	P-008	10
6	Pur-001	P-013	20
7	Pur-001	P-014	20
8	Pur-001	P-015	20
9	Pur-002	P-005	40
10	Pur-002	P-006	30
11	Pur-002	P-007	25
12	Pur-002	P-016	30
13	Pur-002	P-017	30
14	Pur-003	P-009	30
15	Pur-003	P-010	25
16	Pur-003	P-011	25
17	Pur-003	P-012	30
18	Pur-004	P-018	30
19	Pur-004	P-019	35
20	Pur-004	P-020	20
21	Pur-004	P-021	15
22	Pur-005	P-022	20
23	Pur-005	P-023	30
24	Pur-005	P-024	24
25	Pur-005	P-025	27
26	Pur-005	P-026	20
27	Pur-005	P-027	20

Figure 52: Result from Insert Query

```
SELECT * FROM PurchasedProducts;
```

110 %

Results | Messages

	PurchaseID	ProductID	PurchaseLineQuantity
28	Pur-005	P-028	50
29	Pur-005	P-029	45
30	Pur-005	P-030	40
31	Pur-005	P-031	20
32	Pur-005	P-032	20
33	Pur-005	P-033	20
34	Pur-005	P-034	20
35	Pur-006	P-035	10
36	Pur-006	P-036	10
37	Pur-006	P-037	10
38	Pur-006	P-038	10
39	Pur-007	P-039	20
40	Pur-007	P-040	20
41	Pur-007	P-041	20
42	Pur-007	P-042	20
43	Pur-007	P-043	20
44	Pur-008	P-044	20
45	Pur-008	P-045	30
46	Pur-008	P-046	20
47	Pur-008	P-047	20
48	Pur-008	P-048	20
49	Pur-009	P-049	15
50	Pur-009	P-050	15
51	Pur-009	P-051	15
52	Pur-010	P-052	15
53	Pur-010	P-053	15
54	Pur-010	P-054	15
55	Pur-010	P-055	15

Figure 53: Result from Insert Query

6.9 Delivery Table Data insert script

By using INSERT INTO statement, data are inserted into columns of “Delivery” table.

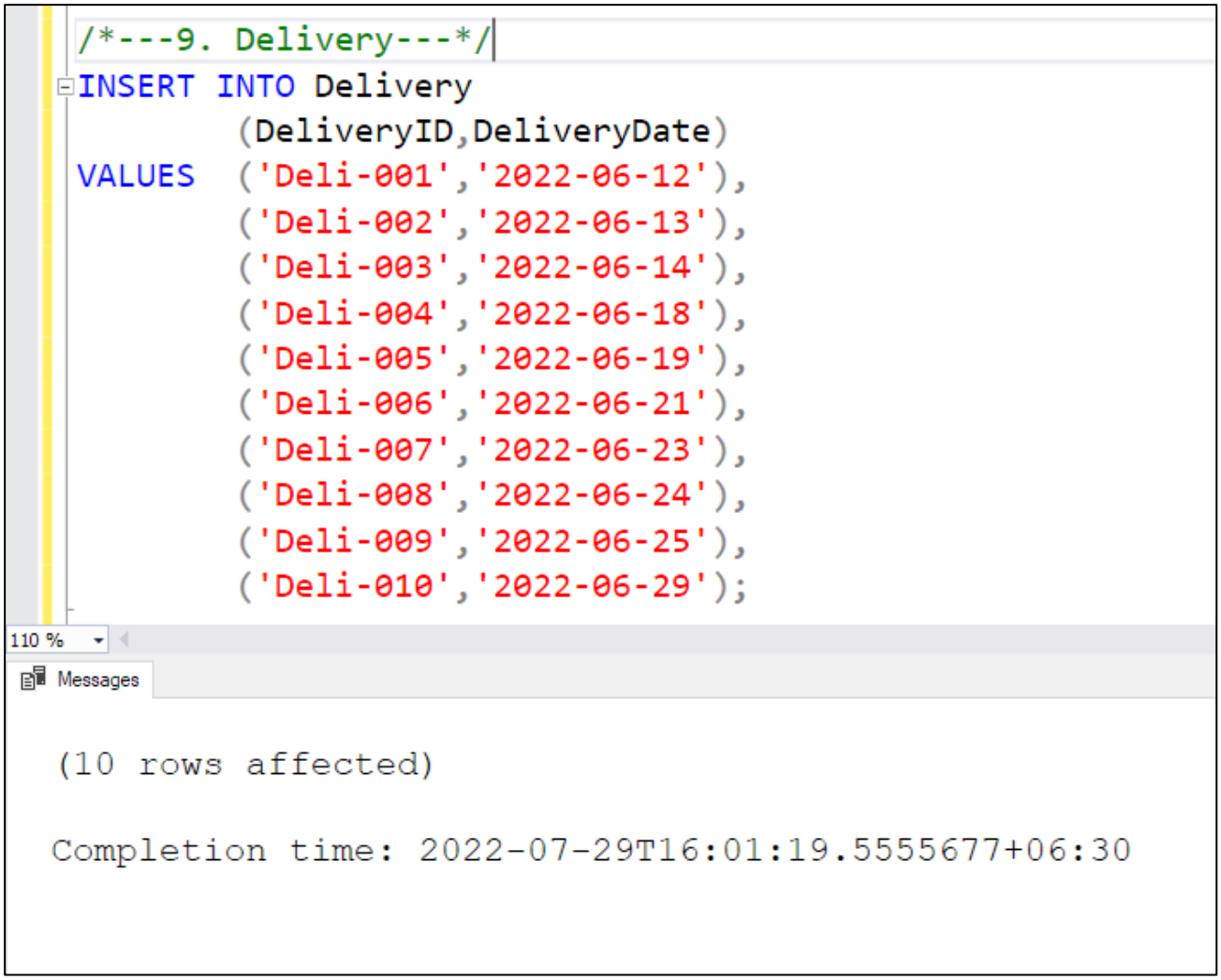


Figure 54: Insert data into Delivery table

Result/Output of Delivery Table

```
SELECT * FROM Delivery;
```

110 %

Results | Messages

	DeliveryID	DeliveryDate
1	Deli-001	2022-06-12
2	Deli-002	2022-06-13
3	Deli-003	2022-06-14
4	Deli-004	2022-06-18
5	Deli-005	2022-06-19
6	Deli-006	2022-06-21
7	Deli-007	2022-06-23
8	Deli-008	2022-06-24
9	Deli-009	2022-06-25
10	Deli-010	2022-06-29

Figure 55: Result from Insert Query

6.10 Orders Table Data insert script

By using INSERT INTO statement, data are inserted into columns of “Orders” table.

```
/*---10. Orders---*/
INSERT INTO Orders
    (OrderID,CustomerID,OrderDate,PromotionID,PaymentAmount,PaymentType,PaymentStatus,OrderStatus,EmployeeID,DeliveryID,DeliveryAddress)
VALUES  ('Ord-001','Cus-001','2022-06-11','Promo-002',52000,'CBPay','Paid','Delivered','E-001','Deli-001','No.19, Room 301, 123th street, MingalarTaungNyunt Tsp, Yangon'),
        ('Ord-002','Cus-002','2022-06-12','Promo-002',45800,'Cash on Delivery','Paid','Delivered','E-001','Deli-002','No.114, Room 201, 135th street, MingalarTaungNyunt Tsp, Yangon'),
        ('Ord-003','Cus-003','2022-06-12','Promo-002',34550,'CBPay','Paid','Delivered','E-002','Deli-002','No.15, Room 201, Aung Tha Pyay street, MingalarTaungNyunt Tsp, Yangon'),
        ('Ord-004','Cus-004','2022-06-13','Promo-001',53250,'KPay','Paid','Delivered','E-002','Deli-003','No.81, Room 501, Hnin Si street, MingalarTaungNyunt Tsp, Yangon'),
        ('Ord-005','Cus-005','2022-06-13','Promo-002',32100,'Cash on Delivery','Paid','Delivered','E-001','Deli-003','No.122, Ground Floor, Kyi Taw street, PaZunTaung Tsp, Yangon'),
        ('Ord-006','Cus-006','2022-06-17','Promo-003',29600,'Cash on Delivery','Paid','Delivered','E-002','Deli-004','No.80, Room 301, Witt Kyaung street, Yay Kyaw Tsp, Yangon'),
        ('Ord-007','Cus-003','2022-06-17','Promo-001',44000,'CBPay','Paid','Delivered','E-001','Deli-004','No.61, Room 501, Aung Tha Pyay street, MingalarTaungNyunt Tsp, Yangon'),
```

110 %

Messages

```
(13 rows affected)

Completion time: 2022-07-29T16:03:45.3534842+06:30
```

Figure 56: Insert data into Orders table

```
        ('Ord-005','Cus-005','2022-06-13','Promo-002',32100,'Cash on Delivery','Paid','Delivered','E-001','Deli-003','No.122, Ground Floor, Kyi Taw street, PaZunTaung Tsp, Yangon'),
        ('Ord-006','Cus-006','2022-06-17','Promo-003',29600,'Cash on Delivery','Paid','Delivered','E-002','Deli-004','No.80, Room 301, Witt Kyaung street, Yay Kyaw Tsp, Yangon'),
        ('Ord-007','Cus-003','2022-06-17','Promo-001',44000,'CBPay','Paid','Delivered','E-001','Deli-004','No.61, Room 501, Aung Tha Pyay street, MingalarTaungNyunt Tsp, Yangon'),
        ('Ord-008','Cus-006','2022-06-18','Promo-001',46500,'Cash on Delivery','Paid','Delivered','E-002','Deli-005','No.80, Room 301, Witt Kyaung street, Yay Kyaw Tsp, Yangon'),
        ('Ord-009','Cus-007','2022-06-20','Promo-003',28350,'KPay','Paid','Delivered','E-001','Deli-006','No.80, Room 401, ThuDaMar street, Kyauk Myaung Tsp, Yangon'),
        ('Ord-010','Cus-008','2022-06-22',NULL,12000,'Cash on Delivery','Paid','Delivered','E-002','Deli-007','No.15, Room 201, Aung Tha Pyay street, MingalarTaungNyunt Tsp, Yangon'),
        ('Ord-011','Cus-003','2022-06-23',NULL,10300,'CBPay','Paid','On the way','E-001','Deli-008','No.15, Room 101, Bayint Naung Road, North Dagon Tsp, Yangon'),
        ('Ord-012','Cus-009','2022-06-24','Promo-003',NULL,'Cash on Delivery','Not Paid','Preparing','E-002','Deli-009','No.64, Room 201, Than street, Hlaing Tsp, Yangon'),
        ('Ord-013','Cus-010','2022-06-28','Promo-003',NULL,'Cash on Delivery','Not Paid','Preparing','E-001','Deli-010','No.51, Ground Floor, 47th street, Botahtaung Tsp, Yangon');
```

110 %

Messages

```
(13 rows affected)

Completion time: 2022-07-29T16:03:45.3534842+06:30
```

Figure 57: Insert data into Orders table

Result/Output of Orders Table

```
SELECT * FROM Orders;
```

110 %

Results | Messages

	OrderID	CustomerID	OrderDate	PromotionID	PaymentAmount	PaymentType	PaymentStatus	OrderStatus	EmployeeID	DeliveryID	DeliveryAddress
1	Ord-001	Cus-001	2022-06-11	Promo-002	52000.00	CBPay	Paid	Delivered	E-001	Deli-001	No.19, Room 301, 123th street, MingalarTaungNyunt Tsp,...
2	Ord-002	Cus-002	2022-06-12	Promo-002	45800.00	Cash on Delivery	Paid	Delivered	E-001	Deli-002	No.114, Room 201, 135th street, MingalarTaungNyunt Ts...
3	Ord-003	Cus-003	2022-06-12	Promo-002	34550.00	CBPay	Paid	Delivered	E-002	Deli-002	No.15, Room 201, Aung Tha Pyay street, MingalarTaungN...
4	Ord-004	Cus-004	2022-06-13	Promo-001	53250.00	KPay	Paid	Delivered	E-002	Deli-003	No.81, Room 501, Hnin Si street, MingalarTaungNyunt Ts...
5	Ord-005	Cus-005	2022-06-13	Promo-002	32100.00	Cash on Delivery	Paid	Delivered	E-001	Deli-003	No.122, Ground Floor, Kyi Taw street, PaZunTaung Tsp, ...
6	Ord-006	Cus-006	2022-06-17	Promo-003	29600.00	Cash on Delivery	Paid	Delivered	E-002	Deli-004	No.80, Room 301, Witt Kyaung street, Yay Kyaw Tsp, Yan...
7	Ord-007	Cus-003	2022-06-17	Promo-001	44000.00	CBPay	Paid	Delivered	E-001	Deli-004	No.61, Room 501, Aung Tha Pyay street, MingalarTaungN...
8	Ord-008	Cus-006	2022-06-18	Promo-001	46500.00	Cash on Delivery	Paid	Delivered	E-002	Deli-005	No.80, Room 301, Witt Kyaung street, Yay Kyaw Tsp, Yan...
9	Ord-009	Cus-007	2022-06-20	Promo-003	28350.00	KPay	Paid	Delivered	E-001	Deli-006	No.80, Room 401, ThuDaMar street, Kyauk Myaung Tsp, ...
10	Ord-010	Cus-008	2022-06-22	NULL	12000.00	Cash on Delivery	Paid	Delivered	E-002	Deli-007	No.15, Room 201, Aung Tha Pyay street, MingalarTaungN...
11	Ord-011	Cus-003	2022-06-23	NULL	10300.00	CBPay	Paid	On the way	E-001	Deli-008	No.15, Room 101, Bayint Naung Road, North Dagon Tsp, ...
12	Ord-012	Cus-009	2022-06-24	Promo-003	NULL	Cash on Delivery	Not Paid	Preparing	E-002	Deli-009	No.64, Room 201, Than street, Hlaing Tsp, Yangon
13	Ord-013	Cus-010	2022-06-28	Promo-003	NULL	Cash on Delivery	Not Paid	Preparing	E-001	Deli-010	No.51, Ground Floor, 47th street, Botahtaung Tsp, Yangon

Figure 58: Result from Insert Query

6.11 OrderProducts Table Data insert script

By using INSERT INTO statement, data are inserted into columns of “OrderProducts” table.

```
/*---11. OrderProducts---*/
INSERT INTO OrderProducts
        (OrderID,ProductID,OrderLineQuantity)
VALUES  ('Ord-001','P-001',2),
        ('Ord-001','P-015',3),
        ('Ord-001','P-007',1),
        ('Ord-001','P-013',1),
        ('Ord-001','P-009',2),
        ('Ord-002','P-036',1),
        ('Ord-002','P-015',2),
        ('Ord-002','P-054',2),
        ('Ord-002','P-005',1),
        ('Ord-003','P-006',2),
        ('Ord-003','P-009',2),
        ('Ord-003','P-016',1),
        ('Ord-003','P-034',1),
        ('Ord-003','P-050',2),
        ('Ord-004','P-030',2),
        ('Ord-004','P-009',1),
        ('Ord-004','P-014',1),
        ('Ord-004','P-033',1),
        ('Ord-004','P-022',1),
```

110 %

Messages

```
(54 rows affected)

Completion time: 2022-07-29T16:06:27.1309511+06:30
```

Figure 59: Insert data into OrderProducts table

Figure 60: Insert data into OrderProducts table

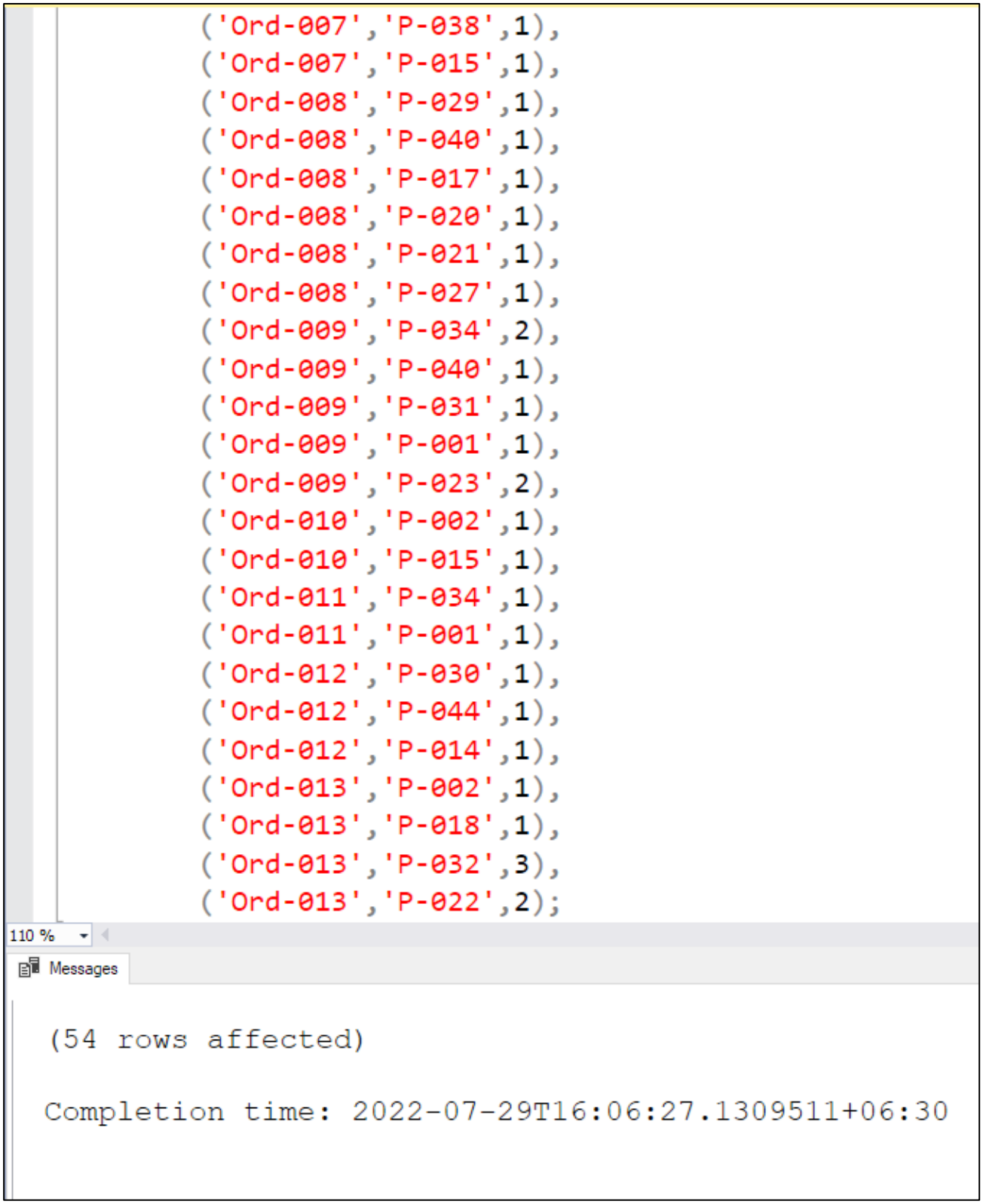


Figure 61: Insert data into OrderProducts table

Result/Output of OrderProducts Table

```
SELECT * FROM OrderProducts;
```

110 %

Results | Messages

	OrderID	ProductID	OrderLineQuantity
1	Ord-001	P-001	2
2	Ord-001	P-007	1
3	Ord-001	P-009	2
4	Ord-001	P-013	1
5	Ord-001	P-015	3
6	Ord-002	P-005	1
7	Ord-002	P-015	2
8	Ord-002	P-036	1
9	Ord-002	P-054	2
10	Ord-003	P-006	2
11	Ord-003	P-009	2
12	Ord-003	P-016	1
13	Ord-003	P-034	1
14	Ord-003	P-050	2
15	Ord-004	P-001	2
16	Ord-004	P-009	1
17	Ord-004	P-014	1
18	Ord-004	P-022	1
19	Ord-004	P-030	2
20	Ord-004	P-033	1
21	Ord-005	P-001	1
22	Ord-005	P-005	1
23	Ord-005	P-011	2
24	Ord-005	P-013	2
25	Ord-005	P-022	2
26	Ord-006	P-001	2

Figure 62: Result from Insert Query

```
SELECT * FROM OrderProducts;
```

110 %

Results | Messages

	OrderID	ProductID	OrderLineQuantity
26	Ord-006	P-001	2
27	Ord-006	P-014	1
28	Ord-006	P-025	2
29	Ord-006	P-033	2
30	Ord-007	P-015	1
31	Ord-007	P-028	2
32	Ord-007	P-038	1
33	Ord-008	P-017	1
34	Ord-008	P-020	1
35	Ord-008	P-021	1
36	Ord-008	P-027	1
37	Ord-008	P-029	1
38	Ord-008	P-040	1
39	Ord-009	P-001	1
40	Ord-009	P-023	2
41	Ord-009	P-031	1
42	Ord-009	P-034	2
43	Ord-009	P-040	1
44	Ord-010	P-002	1
45	Ord-010	P-015	1
46	Ord-011	P-001	1
47	Ord-011	P-034	1
48	Ord-012	P-014	1
49	Ord-012	P-030	1
50	Ord-012	P-044	1
51	Ord-013	P-002	1
52	Ord-013	P-018	1
53	Ord-013	P-022	2
54	Ord-013	P-032	3

Figure 63: Result from Insert Query

6.12 Explanation

In inserting data to above tables, data were carefully typed as there are domain constraint checkers checking for naming prefixes in IDs and not to add data into wrong column or omitting data. As for order of data populations, parent tables' data were needed to be inserted first then the child tables. The only issue encountered was accidently inserting Orders data first without inserting the Delivery table data first. This causes data unable to be inserted to Orders table as the parent table (Delivery table) has no data to refer as foreign keys in Orders table.

Task 7

Enhancement using SQL

7 Task 7: Enhancement using SQL

7.1 Altering OrderProducts Table

Adding OrderLineAmount Column

The above columns are needed to be added to OrderProducts tables due to operational requirements of Basket4U. OrderLineAmount is required for Basket4U when calculating SubTotal and TotalOrderAmount of Orders.

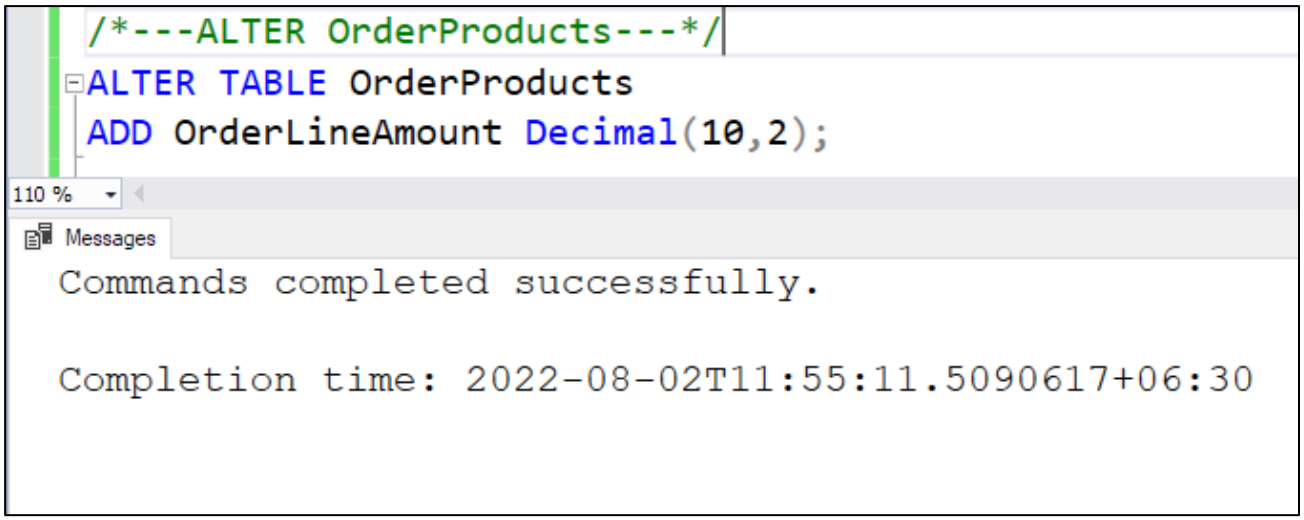


Figure 64: Altering OrderProducts Table

7.2 Updating OrderProducts Table

Updating OrderLineAmount Column

Being a Derived column, OrderLineAmount is calculated by multiplying OrderLineQuantity of OrderProducts Table with UnitPrice of Products table where ProductID of Products and OrderProducts tables are same.

```
UPDATE OrderProducts
SET OrderLineAmount =  (OrderLineQuantity)  *   (SELECT UnitPrice
                                                FROM Products
                                                WHERE Products.ProductID = OrderProducts.ProductID);
```

110 %

Messages

```
(54 rows affected)

Completion time: 2022-08-02T12:10:26.2696954+06:30
```

Figure 65: Updating OrderProducts Column in OrderProducts Table

7.3 Complete Query Result of OrderProducts Table

```
SELECT * FROM OrderProducts;
```

110 %

Results | Messages

	OrderID	ProductID	OrderLineQuantity	OrderLineAmount
1	Ord-001	P-001	2	8000.00
2	Ord-001	P-007	1	9000.00
3	Ord-001	P-009	2	16000.00
4	Ord-001	P-013	1	900.00
5	Ord-001	P-015	3	18000.00
6	Ord-002	P-005	1	11000.00
7	Ord-002	P-015	2	12000.00
8	Ord-002	P-036	1	14000.00
9	Ord-002	P-054	2	8800.00
10	Ord-003	P-006	2	6000.00
11	Ord-003	P-009	2	16000.00
12	Ord-003	P-016	1	2250.00
13	Ord-003	P-034	1	6300.00
14	Ord-003	P-050	2	4000.00
15	Ord-004	P-001	2	8000.00
16	Ord-004	P-009	1	8000.00
17	Ord-004	P-014	1	2200.00
18	Ord-004	P-022	1	3150.00
19	Ord-004	P-030	2	26400.00
20	Ord-004	P-033	1	5500.00

Query executed successfully.

Figure 66: Result after Alter and Update Query

```
SELECT * FROM OrderProducts;
```

110 %

Results | Messages

	OrderID	ProductID	OrderLineQuantity	OrderLineAmount
21	Ord-005	P-001	1	4000.00
22	Ord-005	P-005	1	11000.00
23	Ord-005	P-011	2	9000.00
24	Ord-005	P-013	2	1800.00
25	Ord-005	P-022	2	6300.00
26	Ord-006	P-001	2	8000.00
27	Ord-006	P-014	1	2200.00
28	Ord-006	P-025	2	8400.00
29	Ord-006	P-033	2	11000.00
30	Ord-007	P-015	1	6000.00
31	Ord-007	P-028	2	12000.00
32	Ord-007	P-038	1	33000.00
33	Ord-008	P-017	1	12500.00
34	Ord-008	P-020	1	5100.00
35	Ord-008	P-021	1	3600.00
36	Ord-008	P-027	1	10200.00
37	Ord-008	P-029	1	14000.00
38	Ord-008	P-040	1	950.00
39	Ord-009	P-001	1	4000.00
40	Ord-009	P-023	2	800.00

Query executed successfully.

Figure 67: Result after Alter and Update Query

```
SELECT * FROM OrderProducts;
```

110 %

Results | Messages

	OrderID	ProductID	OrderLineQuantity	OrderLineAmount
35	Ord-008	P-021	1	3600.00
36	Ord-008	P-027	1	10200.00
37	Ord-008	P-029	1	14000.00
38	Ord-008	P-040	1	950.00
39	Ord-009	P-001	1	4000.00
40	Ord-009	P-023	2	800.00
41	Ord-009	P-031	1	10000.00
42	Ord-009	P-034	2	12600.00
43	Ord-009	P-040	1	950.00
44	Ord-010	P-002	1	6000.00
45	Ord-010	P-015	1	6000.00
46	Ord-011	P-001	1	4000.00
47	Ord-011	P-034	1	6300.00
48	Ord-012	P-014	1	2200.00
49	Ord-012	P-030	1	13200.00
50	Ord-012	P-044	1	3500.00
51	Ord-013	P-002	1	6000.00
52	Ord-013	P-018	1	6000.00
53	Ord-013	P-022	2	6300.00
54	Ord-013	P-032	3	6000.00

Query executed successfully.

Figure 68: Result after Alter and Update Query

7.4 Altering Orders Table

Adding TotalOrderQuantity, SubTotal, PromotionDiscountAmount, Tax, TotalOrderAmount Columns to Orders Tables

The above columns are needed to be added to Orders tables due to operational requirements of Basket4U. All columns are required for Basket4U when calculating TotalOrderAmount of Orders for customers.

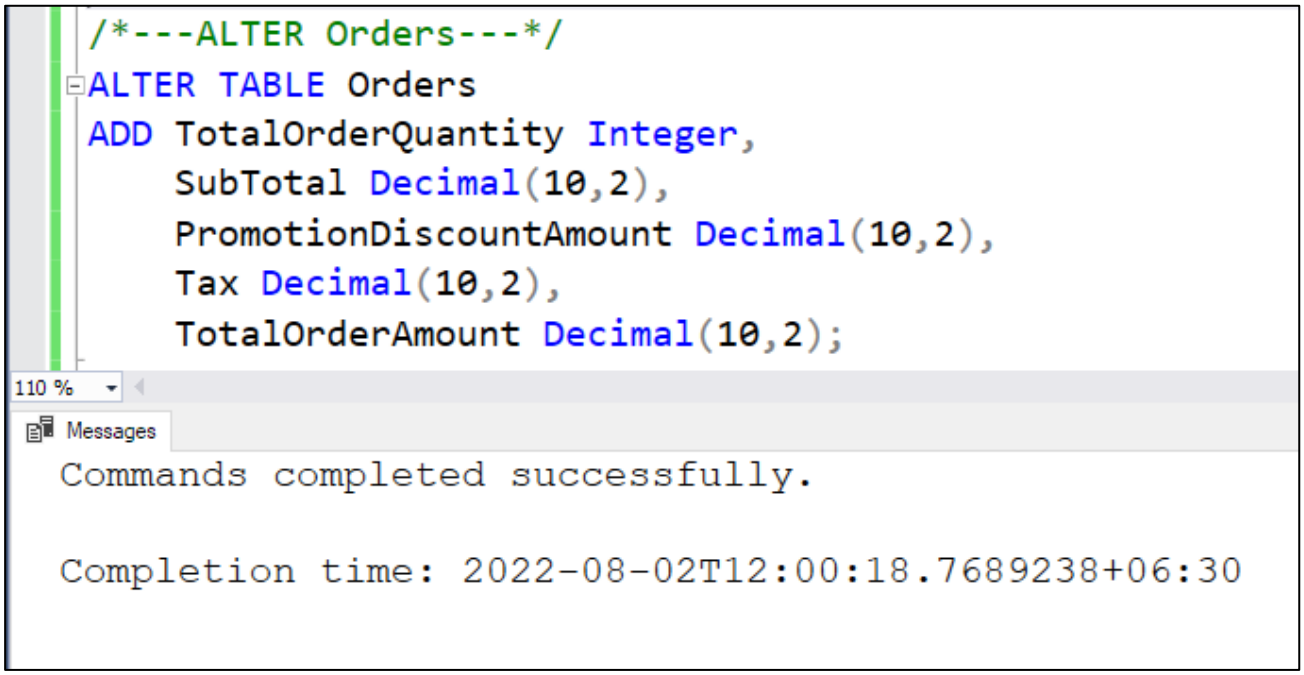


Figure 69: Altering Orders Table

7.5 Updating Orders Table

Updating TotalOrderQuantity Column

Being a Derived column, TotalOrderQuantity is calculated by SUM of all OrderLineQuantity of OrderProducts where OrderID of Orders and OrderProducts Tables are same.

```
UPDATE Orders
SET TotalOrderQuantity =    (SELECT SUM(OrderLineQuantity)
                             FROM OrderProducts
                             WHERE Orders.OrderID = OrderProducts.OrderID);
```

```
110 %
Messages

(13 rows affected)

Completion time: 2022-08-02T12:14:35.6578705+06:30
```

Figure 70: Updating TotalOrderQuantity Column in Orders Table

Updating SubTotal Column

Being a Derived column, SubTotal is calculated by SUM of all OrderLineAmount of OrderProducts Table where OrderID of Orders and OrderProducts Tables are same.

```
UPDATE Orders
SET SubTotal =  (SELECT SUM(OrderLineAmount)
                 FROM OrderProducts
                 WHERE Orders.OrderID = OrderProducts.OrderID);
```

```
110 %
Messages

(13 rows affected)

Completion time: 2022-08-02T12:15:15.1280968+06:30
```

Figure 71: Updating SubTotal Column in Orders Table

Updating Tax Column

Being a Derived column, Tax is calculated by multiplying Subtotal of Orders Tables with 5% Consumer Tax percentage.

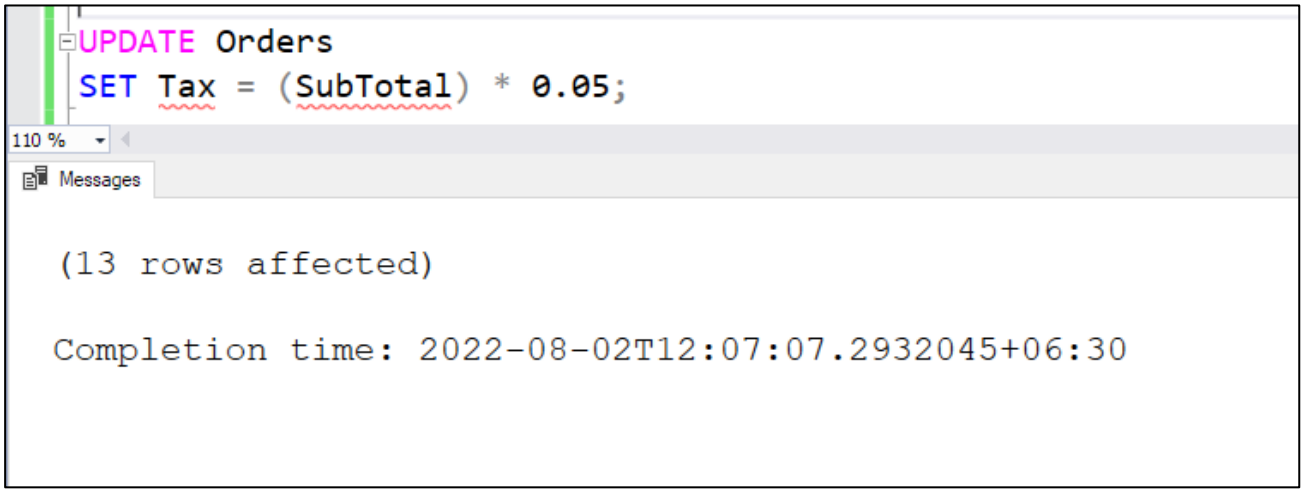


Figure 72: Updating Tax Column in Orders Table

Updating PromotionDiscountAmount Column

Being a Derived column, PromotionDiscountAmount is calculated by multiplying SubTotal from Orders Table and PromotionAmount from Promotions Table where PromotionID of Orders and Promotions Tables are same.

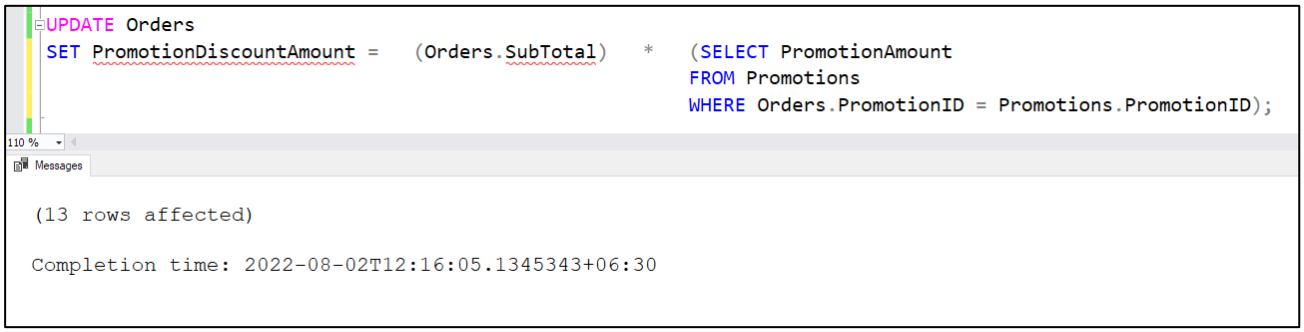


Figure 73: Updating PromotionDiscountAmount Column in Orders Table

Updating TotalOrderAmount Column

Being a Derived column, TotalOrderAmount is calculated by subtracting PromotionDiscountAmount from SubTotal and Adding Tax to SubTotal of Orders Table.

```
UPDATE Orders
SET TotalOrderAmount =  (Orders.SubTotal)   -   ISNULL(0,(Orders.PromotionDiscountAmount))  +   (Orders.Tax);
```

```
(13 rows affected)

Completion time: 2022-08-02T12:16:58.7911397+06:30
```

Figure 74: Updating TotalOrderAmount Column in Orders Table

7.6 Complete Query Result of Orders Table

```
SELECT * FROM Orders;
```

	OrderID	CustomerID	OrderDate	PromotionID	PaymentAmount	PaymentType	PaymentStatus	OrderStatus	EmployeeID	DeliveryID	DeliveryAddress	TotalOrderQuantity	SubTotal
1	Ord-001	Cus-001	2022-06-11	Promo-002	52000.00	CBPay	Paid	Delivered	E-001	Deli-001	No.19, Room 301, 123th street, MingalarTaungNyunt ...	9	51900.00
2	Ord-002	Cus-002	2022-06-12	Promo-002	45800.00	Cash on Delivery	Paid	Delivered	E-001	Deli-002	No.114, Room 201, 135th street, MingalarTaungNyunt...	6	45800.00
3	Ord-003	Cus-003	2022-06-12	Promo-002	34550.00	CBPay	Paid	Delivered	E-002	Deli-002	No.15, Room 201, Aung Tha Pyay street, MingalarTau...	8	34550.00
4	Ord-004	Cus-004	2022-06-13	Promo-001	53250.00	KPay	Paid	Delivered	E-002	Deli-003	No.81, Room 501, Hnin Si street, MingalarTaungNyunt...	8	53250.00
5	Ord-005	Cus-005	2022-06-13	Promo-002	32100.00	Cash on Delivery	Paid	Delivered	E-001	Deli-003	No.122, Ground Floor, Kyi Taw street, PaZunTaung Ts...	8	32100.00
6	Ord-006	Cus-006	2022-06-17	Promo-003	29600.00	Cash on Delivery	Paid	Delivered	E-002	Deli-004	No.80, Room 301, Witt Kyaung street, Yay Kyaw Tsp, ...	7	29600.00
7	Ord-007	Cus-003	2022-06-17	Promo-001	44000.00	CBPay	Paid	Delivered	E-001	Deli-004	No.61, Room 501, Aung Tha Pyay street, MingalarTau...	4	51000.00
8	Ord-008	Cus-006	2022-06-18	Promo-001	46500.00	Cash on Delivery	Paid	Delivered	E-002	Deli-005	No.80, Room 301, Witt Kyaung street, Yay Kyaw Tsp, ...	6	46350.00
9	Ord-009	Cus-007	2022-06-20	Promo-003	28350.00	KPay	Paid	Delivered	E-001	Deli-006	No.80, Room 401, ThuDaMar street, Kyauk Myaung T...	7	28350.00
10	Ord-010	Cus-008	2022-06-22	NULL	12000.00	Cash on Delivery	Paid	Delivered	E-002	Deli-007	No.15, Room 201, Aung Tha Pyay street, MingalarTau...	2	12000.00
11	Ord-011	Cus-003	2022-06-23	NULL	10300.00	CBPay	Paid	On the way	E-001	Deli-008	No.15, Room 101, Bayint Naung Road, North Dagon T...	2	10300.00
12	Ord-012	Cus-009	2022-06-24	Promo-003	NULL	Cash on Delivery	Not Paid	Preparing	E-002	Deli-009	No.64, Room 201, Than street, Hlaing Tsp, Yangon	3	18900.00
13	Ord-013	Cus-010	2022-06-28	Promo-003	NULL	Cash on Delivery	Not Paid	Preparing	E-001	Deli-010	No.51, Ground Floor, 47th street, Botahtaung Tsp, Ya...	7	24300.00

Figure 75: Result after Alter and Update Query

```
SELECT * FROM Orders;
```

	ntAmount	PaymentType	PaymentStatus	OrderStatus	EmployeeID	DeliveryID	DeliveryAddress	TotalOrderQuantity	SubTotal	PromotionDiscountAmount	Tax	TotalOrderAmount
1	.00	CBPay	Paid	Delivered	E-001	Deli-001	No.19, Room 301, 123th street, MingalarTaungNyunt ...	9	51900.00	5190.00	2595.00	54495.00
2	.00	Cash on Delivery	Paid	Delivered	E-001	Deli-002	No.114, Room 201, 135th street, MingalarTaungNyunt...	6	45800.00	4580.00	2290.00	48090.00
3	.00	CBPay	Paid	Delivered	E-002	Deli-002	No.15, Room 201, Aung Tha Pyay street, MingalarTau...	8	34550.00	3455.00	1727.50	36277.50
4	.00	KPay	Paid	Delivered	E-002	Deli-003	No.81, Room 501, Hnin Si street, MingalarTaungNyunt...	8	53250.00	7987.50	2662.50	55912.50
5	.00	Cash on Delivery	Paid	Delivered	E-001	Deli-003	No.122, Ground Floor, Kyi Taw street, PaZunTaung Ts...	8	32100.00	3210.00	1605.00	33705.00
6	.00	Cash on Delivery	Paid	Delivered	E-002	Deli-004	No.80, Room 301, Witt Kyaung street, Yay Kyaw Tsp, ...	7	29600.00	1480.00	1480.00	31080.00
7	.00	CBPay	Paid	Delivered	E-001	Deli-004	No.61, Room 501, Aung Tha Pyay street, MingalarTau...	4	51000.00	7650.00	2550.00	53550.00
8	.00	Cash on Delivery	Paid	Delivered	E-002	Deli-005	No.80, Room 301, Witt Kyaung street, Yay Kyaw Tsp, ...	6	46350.00	6952.50	2317.50	48667.50
9	.00	KPay	Paid	Delivered	E-001	Deli-006	No.80, Room 401, ThuDaMar street, Kyauk Myaung T...	7	28350.00	1417.50	1417.50	29767.50
10	.00	Cash on Delivery	Paid	Delivered	E-002	Deli-007	No.15, Room 201, Aung Tha Pyay street, MingalarTau...	2	12000.00	NULL	600.00	12600.00
11	.00	CBPay	Paid	On the way	E-001	Deli-008	No.15, Room 101, Bayint Naung Road, North Dagon T...	2	10300.00	NULL	515.00	10815.00
12		Cash on Delivery	Not Paid	Preparing	E-002	Deli-009	No.64, Room 201, Than street, Hlaing Tsp, Yangon	3	18900.00	945.00	945.00	19845.00
13		Cash on Delivery	Not Paid	Preparing	E-001	Deli-010	No.51, Ground Floor, 47th street, Botahtaung Tsp, Ya...	7	24300.00	1215.00	1215.00	25515.00

Figure 76: Result after Alter and Update Query

7.7 Altering PurchasedProducts Table

Adding PurchaseLineAmount Column to PurchasedProducts Table

The above columns are needed to be added to PurchasedProducts tables due to operational requirements of Basket4U. All are required for Basket4U when calculating TotalPurchaseQuantity and TotalPurchaseAmount of Purchases Table.

```
/*---ALTER PurchasedProducts---*/
ALTER TABLE PurchasedProducts
ADD PurchaseLineAmount Decimal(10,2);
```

110 %

Messages

```
Commands completed successfully.

Completion time: 2022-08-02T12:28:47.6620607+06:30
```

Figure 77: Altering PurchasedProducts Table

7.8 Updating PurchasedProducts Table

Updating PurchaseLineAmount Column

Being a Derived column, PurchaseLineAmount is calculated by multiplying PurchaseLineQuantity with UnitPrice of Products Table where ProductID of Products and PurchasedProducts Tables are same.

```
UPDATE PurchasedProducts
SET PurchaseLineAmount =    (PurchaseLineQuantity)  *   (SELECT UnitPrice
                                                        FROM Products
                                                        WHERE Products.ProductID = PurchasedProducts.ProductID);
```

110 %

Messages

```
(55 rows affected)

Completion time: 2022-08-02T12:30:04.1850938+06:30
```

Figure 78: Updating PurchaseLineAmount Column in PurchasedProducts Table

7.9 Complete Query Result of PurchasedProducts Table

```
SELECT * FROM PurchasedProducts;
```

110 %

Results | Messages

	PurchaseID	ProductID	PurchaseLineQuantity	PurchaseLineAmount
1	Pur-001	P-001	40	160000.00
2	Pur-001	P-002	25	150000.00
3	Pur-001	P-003	25	80000.00
4	Pur-001	P-004	30	75000.00
5	Pur-001	P-008	10	60000.00
6	Pur-001	P-013	20	18000.00
7	Pur-001	P-014	20	44000.00
8	Pur-001	P-015	20	120000.00
9	Pur-002	P-005	40	440000.00
10	Pur-002	P-006	30	90000.00
11	Pur-002	P-007	25	225000.00
12	Pur-002	P-016	30	67500.00
13	Pur-002	P-017	30	375000.00
14	Pur-003	P-009	30	240000.00
15	Pur-003	P-010	25	237500.00
16	Pur-003	P-011	25	112500.00
17	Pur-003	P-012	30	186000.00
18	Pur-004	P-018	30	180000.00
19	Pur-004	P-019	35	140000.00
20	Pur-004	P-020	20	102000.00
21	Pur-004	P-021	15	54000.00
22	Pur-005	P-022	20	63000.00
23	Pur-005	P-023	30	12000.00
24	Pur-005	P-024	24	100800.00

Query executed successfully.

Figure 79: Result after Alter and Update Query

```
SELECT * FROM PurchasedProducts;
```

110 %

Results | Messages

	PurchaseID	ProductID	PurchaseLineQuantity	PurchaseLineAmount
25	Pur-005	P-025	27	113400.00
26	Pur-005	P-026	20	370000.00
27	Pur-005	P-027	20	204000.00
28	Pur-005	P-028	50	300000.00
29	Pur-005	P-029	45	630000.00
30	Pur-005	P-030	40	528000.00
31	Pur-005	P-031	20	200000.00
32	Pur-005	P-032	20	40000.00
33	Pur-005	P-033	20	110000.00
34	Pur-005	P-034	20	126000.00
35	Pur-006	P-035	10	155000.00
36	Pur-006	P-036	10	140000.00
37	Pur-006	P-037	10	300000.00
38	Pur-006	P-038	10	330000.00
39	Pur-007	P-039	20	19000.00
40	Pur-007	P-040	20	19000.00
41	Pur-007	P-041	20	46000.00
42	Pur-007	P-042	20	26000.00
43	Pur-007	P-043	20	26000.00
44	Pur-008	P-044	20	70000.00
45	Pur-008	P-045	30	105000.00
46	Pur-008	P-046	20	76000.00
47	Pur-008	P-047	20	76000.00
48	Pur-008	P-048	20	134000.00

Query executed successfully.

Figure 80: Result after Alter and Update Query

```
SELECT * FROM PurchasedProducts;
```

110 %

Results | Messages

	PurchaseID	ProductID	PurchaseLineQuantity	PurchaseLineAmount
32	Pur-005	P-032	20	40000.00
33	Pur-005	P-033	20	110000.00
34	Pur-005	P-034	20	126000.00
35	Pur-006	P-035	10	155000.00
36	Pur-006	P-036	10	140000.00
37	Pur-006	P-037	10	300000.00
38	Pur-006	P-038	10	330000.00
39	Pur-007	P-039	20	19000.00
40	Pur-007	P-040	20	19000.00
41	Pur-007	P-041	20	46000.00
42	Pur-007	P-042	20	26000.00
43	Pur-007	P-043	20	26000.00
44	Pur-008	P-044	20	70000.00
45	Pur-008	P-045	30	105000.00
46	Pur-008	P-046	20	76000.00
47	Pur-008	P-047	20	76000.00
48	Pur-008	P-048	20	134000.00
49	Pur-009	P-049	15	15000.00
50	Pur-009	P-050	15	30000.00
51	Pur-009	P-051	15	30000.00
52	Pur-010	P-052	15	67500.00
53	Pur-010	P-053	15	45000.00
54	Pur-010	P-054	15	66000.00
55	Pur-010	P-055	15	82500.00

Query executed successfully.

Figure 81: Result after Alter and Update Query

7.10 Altering Purchases Table

Adding TotalPurchaseQuantity, TotalPurchaseAmount Column to Purchases Table

The above columns are needed to be added to Purchases tables due to operational requirements of Basket4U. All are required for Basket4U to know how many items and how much amount were spent in each of their purchases.

```
/*---ALTER Purchases---*/
ALTER TABLE Purchases
ADD TotalPurchaseQuantity Integer,
    TotalPurchaseAmount Decimal(10,2);
```

110 %

Messages

```
Commands completed successfully.

Completion time: 2022-08-02T12:33:36.5221734+06:30
```

Figure 82: Altering Purchases Table

7.11 Updating Purchases Table

Updating TotalPurchaseQuantity Column

Being a Derived column, TotalPurchaseQuantity is calculated by SUM of all PurchaseLineQuantity of PurchasedProducts Table where PurchaseID of Purchases and PurchasedProducts Tables are same.

```
UPDATE Purchases
SET TotalPurchaseQuantity = (SELECT SUM(PurchaseLineQuantity)
                             FROM PurchasedProducts
                             WHERE Purchases.PurchaseID = PurchasedProducts.PurchaseID);
```

```
110 %
Messages

(10 rows affected)

Completion time: 2022-08-02T12:34:39.2316037+06:30
```

Figure 83: Updating TotalPurchaseQuantity Column in Purchases Table

Updating TotalPurchaseAmount Column

Being a Derived column, TotalPurchaseAmount is calculated by SUM of all PurchaseLineAmount of PurchasedProducts Table where PurchaseID of Purchases and PurchasedProducts Tables are same.

```
UPDATE Purchases
SET TotalPurchaseAmount =   (SELECT SUM(PurchaseLineAmount)
                             FROM PurchasedProducts
                             WHERE Purchases.PurchaseID = PurchasedProducts.PurchaseID);
```

```
110 %
Messages

(10 rows affected)

Completion time: 2022-08-02T12:35:25.6295677+06:30
```

Figure 84: Updating TotalPurchaseAmount Column in Purchases Table

7.12 Complete Query Result of Purchases Table

```
SELECT * FROM Purchases;
```

110 %

Results | Messages

	PurchaseID	PurchaseDate	SupplierID	TotalPurchaseQuantity	TotalPurchaseAmount
1	Pur-001	2022-06-05	S-001	190	707000.00
2	Pur-002	2022-06-05	S-002	155	1197500.00
3	Pur-003	2022-06-05	S-003	110	776000.00
4	Pur-004	2022-06-06	S-009	100	476000.00
5	Pur-005	2022-06-07	S-004	356	2797200.00
6	Pur-006	2022-06-07	S-005	40	925000.00
7	Pur-007	2022-06-07	S-006	100	136000.00
8	Pur-008	2022-06-08	S-007	110	461000.00
9	Pur-009	2022-06-09	S-008	45	75000.00
10	Pur-010	2022-06-10	S-010	60	261000.00

Query executed successfully.

Figure 85: Result after Alter and Update Query

Task 8

SQL Reports

8 Task 8: SQL Reports

8.1 To show Top 5 Most Bought Products for June Month

The following query is retrieving data to show Top 5 Most Bought Products for June Month. This query is helpful for Basket4U as Business Intelligent in report as it shows list of products that customers buy most together with total quantity in a specific month. By knowing customers' demand, and most buying products, Basket4U can prepare in advance to refill products if necessary.

```
/*----------SQL REPORT QUERY CODES----------*/
/*---1. Top 5 most bought products---*/
SELECT TOP 5 p.ProductName as 'Top 5 Most Bought Products For June Month',
SUM(op.OrderLineQuantity) as TotalOrderQuantity
FROM OrderProducts as op,Products as p, Orders as o
WHERE op.ProductID = p.ProductID
AND op.OrderID = o.OrderID
AND o.OrderDate BETWEEN '2022-06-01' AND '2022-06-30'
GROUP BY op.ProductID, p.ProductName
ORDER BY SUM(op.OrderLineQuantity) DESC;
```

110 %

Results | Messages

	Top 5 Most Bought Products For June Month	TotalOrderQuantity
1	Chicken Breast	9
2	Grapes	7
3	Pork Tenderloin	5
4	TYC PawSanHmwe Rice	5
5	Moccona Trio 3in1 Espresso	4

Figure 86: SQL Report showing Top 5 Most Bought Products for June Month

8.2 To show Top 5 Least Bought Products for June Month

The following query is retrieving data to show Top 5 Least Bought Products for June Month. This query is helpful for Basket4U as Business Intelligent in report as it shows list of products that customers buy least together with total quantity in a specific month. By knowing customers' non-demands, and least buying products, Basket4U can prepare not to refill these products again from suppliers until empty and in turn, preventing unnecessary expenditure in Basket4U purchases for profits.

```
/*---2. Top 5 least bought products---*/
SELECT TOP 5 p.ProductName as 'Top 5 Least Bought Products For June Month',
SUM(op.OrderLineQuantity) as 'Total Order Quantity'
FROM OrderProducts as op,Products as p, Orders as o
WHERE op.ProductID = p.ProductID
AND op.OrderID = o.OrderID
AND o.OrderDate BETWEEN '2022-06-01' AND '2022-06-30'
GROUP BY op.ProductID, p.ProductName
ORDER BY SUM(op.OrderLineQuantity) ;
```

110 %

Results | Messages

	Top 5 Least Bought Products For June Month	Total Order Quantity
1	Ovaltine Chocolate Bottle	1
2	DingWang Rice Cookies	1
3	Shoon Fatt Corn Biscuit Box	1
4	Nestle Rice Chicken Nutrition Powder	1
5	Babymild Cherry Blossom Cream	1

Figure 87: SQL Report showing Top 5 Least Bought Products for June Month

8.3 To show Products that have not been bought together with Quantity left

The following query is retrieving data to show Products that were never bought (with Quantity left in Basket4U). This query is helpful for Basket4U as Business Intelligent in report as it shows list of products that customers do not buy together with total quantity left. By knowing customers' non-demands, and non-buying products, Basket4U can prepare not to buy these products again from suppliers until empty and in turn, preventing unnecessary expenditure in Basket4U purchases for profits.

```
/*---3. 10 Products that have not been bought together with Quantity left---*/
SELECT TOP 10 p.ProductName as 'Name of Products that have not been bought', p.ProductID,
pp.PurchaseLineQuantity as 'Quantity left'
FROM Products as p, PurchasedProducts as pp
WHERE NOT EXISTS (SELECT 1
                  FROM OrderProducts as op
                  WHERE p.ProductID = op.ProductID)
AND p.ProductID = pp.ProductID
ORDER BY 'Quantity left' DESC;
```

110 %

Results | Messages

	Name of Products that have not been bought	ProductID	Quantity left
1	Komodo Tissues 70pcs	P-019	35
2	Burmese Whole Chicken	P-012	30
3	Chicken Thigh with skin	P-004	30
4	Colgate Max Fresh Mint Toothpaste	P-045	30
5	Pork 3 Layers	P-003	25
6	Pork Ribs Special	P-010	25
7	KanBawZa Shan Rice	P-024	24
8	KanBawZa SeeCho Rice	P-026	20
9	AirX Stomach Relief	P-039	20
10	Decolgen	P-041	20

Figure 88: SQL Report showing Products that have not been bought together with Quantity left

8.4 To show Total Tax To Pay This Year

The following query is retrieving data to show Total Consumer Tax to be paid by Basket4U for this year. This query is helpful for Basket4U as Business Intelligent in report as it shows tax expenditure for specific year. By knowing tax expenditure, Basket4U can manage other controllable expenses for profits and be ready to pay for taxes.

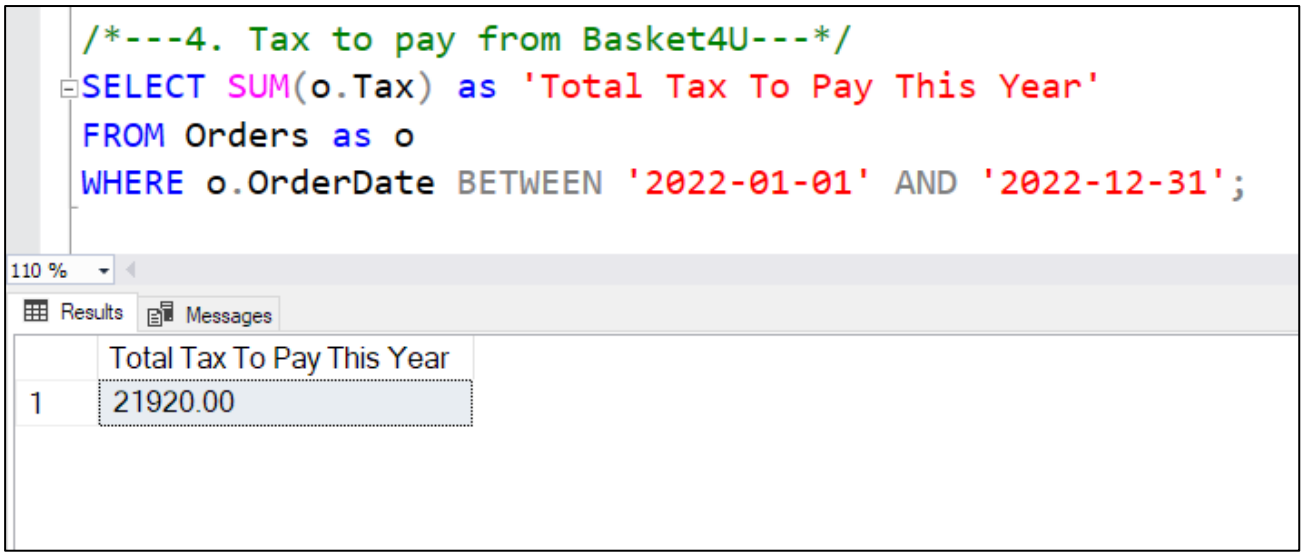


Figure 89: SQL Report showing Total Tax To Pay This Year

8.5 To show Most Used Promotion Code for the Year

The following query is retrieving data to show most used promotion code for the year. This query is helpful for Basket4U as Business Intelligent in report as it shows list of promotion code that are most used by customers. By knowing the codes, Basket4U can know customer's average spending ability and Promotion events that customers participated the most.

```
/*---5. Most Used Promotion Code For the year---*/
SELECT promo.PromotionCode as 'Most Used Promotion Code For The Year', COUNT(o.PromotionID) as 'Total Times Used'
FROM Orders as o, Promotions as promo
WHERE o.PromotionID = promo.PromotionID
AND o.OrderDate BETWEEN '2022-01-01' AND '2022-12-31'
GROUP BY promo.PromotionCode
ORDER BY 'Total Times Used' DESC;
```

110 %

Results | Messages

	Most Used Promotion Code For The Year	Total Times Used
1	B4Uoff10%	4
2	B4Uoff5%	4
3	B4Uoff15%	3

Figure 90: SQL Report showing Most Used Promotion Code for the Year

8.6 To show Top 5 Most Bought Categories with Total Quantity Bought For June

The following query is retrieving data to show Top 5 most bought categories for June . This query is helpful for Basket4U as Business Intelligent in report as it shows list of Product Categories that customers buy most together with total quantity in a specific month. By knowing customers' demands, and most buying categories, Basket4U can prepare to refill these categories with new and interesting products that customers could order and in turn, increasing more order products for profits for Basket4U.

```
/*---6. TOP 5 Most Bought Categories with Total Quantity Bought For June---*/
SELECT TOP 5 cat.CategoryName as 'Top 5 Most Bought Categories for June', cat.CategoryID,
SUM(op.OrderLineQuantity) as 'Total Quantity Bought'
FROM Categories as cat, Products as p, OrderProducts as op, Orders as o
WHERE p.ProductID = op.ProductID
AND p.CategoryID = cat.CategoryID
AND o.OrderID = op.OrderID
AND o.OrderDate BETWEEN '2022-01-01' AND '2022-12-31'
GROUP BY cat.CategoryID, cat.CategoryName
ORDER BY 'Total Quantity Bought' DESC;
```

110 %

Results | Messages

	Top 5 Most Bought Categories for June	CategoryID	Total Quantity Bought
1	Meats and Seafood	Cat-001	18
2	Snacks and Beverages	Cat-003	17
3	Cooking Groceries	Cat-010	17
4	Fruits and Vegetables	Cat-002	13
5	Mother and Childcare	Cat-006	3

Figure 91: SQL Report showing Top 5 Most Bought Categories with Total Quantity Bought For June

8.7 To show Top 5 Purchased Products from Suppliers for June Month with Total Purchase Quantity

The following query is retrieving data to show Top 5 Purchased Products from suppliers for June Month. This query is helpful for Basket4U as Business Intelligent in report as it shows list of products that Basket4U buys most together with total quantity in a specific month. By knowing most purchased products from suppliers, both suppliers side and Basket4U can prepare to be ready for refills whenever necessary.

```
/*---7. Top 5 Purchased Products From Suppliers for June Month with Total Purchase Quantity---*/
SELECT TOP 5 p.ProductName as 'Top 5 Most Purchased Products For June Month',pp.ProductID,
SUM(pp.PurchaseLineQuantity) as 'Total Purchase Quantity', s.SupplierName
FROM PurchasedProducts as pp,Products as p, Purchases as pur,Suppliers as s
WHERE pp.ProductID = p.ProductID
AND pp.PurchaseID = pur.PurchaseID
AND pur.SupplierID = s.SupplierID
AND pur.PurchaseDate BETWEEN '2022-06-01' AND '2022-06-30'
GROUP BY pp.ProductID, p.ProductName, s.SupplierName
ORDER BY SUM(pp.PurchaseLineQuantity) DESC;
```

110 %

Results | Messages

	Top 5 Most Purchased Products For June Month	ProductID	Total Purchase Quantity	SupplierName
1	COOK Soybean Oil	P-028	50	Aung Kabar
2	Shwe GroundNut Oil	P-029	45	Aung Kabar
3	COOK Sunflower Oil	P-030	40	Aung Kabar
4	Chicken Breast	P-001	40	Shwe Si Daw
5	Danisa Cookies	P-005	40	Uncle Lay

Figure 92: SQL Report showing Top 5 Purchased Products from Suppliers for June Month with Total Purchased Quantity

8.8 To show Top 3 Customers that orders the most times from Basket4U

The following query is retrieving data to show Top 3 Customers that order the most times from Basket4U. This query is helpful for Basket4U as Business Intelligent in report as it shows list of loyal customers that regularly and frequently orders from Basket4U. By knowing such loyal customers, Basket4U can prepare promotion event based on order times for such customers and establish more satisfying and long-lasting customer relationships with them.

```
/*---8. Top 3 Customers that orders the most times from Basket4U---*/
SELECT TOP 3 cus.CustomerName as 'Top 3 Customers that orders the most times from Basket4U', cus.CustomerID,
COUNT(o.OrderID) as 'Order Times'
FROM Orders as o, Customers as cus
WHERE o.CustomerID = cus.CustomerID
GROUP BY cus.CustomerName, cus.CustomerID
ORDER BY 'Order Times' DESC;
```

110 %

Results | Messages

	Top 3 Customers that orders the most times from Basket4U	CustomerID	Order Times
1	Ci Ci	Cus-003	3
2	Ingyin Khin	Cus-006	2
3	Aung Khant	Cus-004	1

Figure 93: SQL Report showing Top 3 Customers that orders the most times from Basket4U

8.9 To show Top 5 Customers that has the largest order values of all time

The following query is retrieving data to show Top 5 Customers that has the largest order values of all time. This query is helpful for Basket4U as Business Intelligent in report as it shows list of special customers who buy the largest values of total from Basket4U. By knowing such special and spendable customers, Basket4U can prepare promotion event for them and establish more satisfying and long-lasting customer relationships with them.

```
/*---9. Top 5 Customers that has the largest order values of all time---*/
SELECT TOP 5 cus.CustomerName as 'Top 5 Customers that has the largest order values of all time', cus.CustomerID,
SUM(o.TotalOrderAmount) as 'Order Values'
FROM Customers as cus, Orders as o
WHERE o.CustomerID = cus.CustomerID
GROUP BY cus.CustomerName, cus.CustomerID
ORDER BY 'Order Values' DESC;
```

110 %

Results | Messages

	Top 5 Customers that has the largest order values of all time	CustomerID	Order Values
1	Ci Ci	Cus-003	100642.50
2	Ingyin Khin	Cus-006	79747.50
3	Aung Khant	Cus-004	55912.50
4	Maung Maung	Cus-001	54495.00
5	Kaung Khant	Cus-002	48090.00

Figure 94: SQL Report showing Top 5 Customers that has the largest order values of all time

8.10 To show Top 3 Days that Delivery has to send most orders

The following query is retrieving data to show Top 3 Days that Delivery has to send most orders. This query is helpful for Basket4U as Business Intelligent in report as it shows lists of days that the deliveries will be busy by sending most orders. By knowing such days, delivery team can be aware and prepare logistics ahead for faster/ on time delivery to customers.

```
/*---10. Top 3 Days that Delivery has to send most orders---*/
SELECT TOP 3 deli.DeliveryDate 'Top 3 Days that Delivery has to send most orders', deli.DeliveryID,
Count(o.DeliveryID) as 'Total Orders'
FROM Orders as o, Delivery as deli
WHERE o.DeliveryID = deli.DeliveryID
GROUP BY deli.DeliveryDate,deli.DeliveryID
ORDER BY 'Total Orders' DESC;
```

110 %

Results | Messages

	Top 3 Days that Delivery has to send most orders	DeliveryID	Total Orders
1	2022-06-18	Deli-004	2
2	2022-06-14	Deli-003	2
3	2022-06-13	Deli-002	2

Figure 95: SQL Report showing Top 3 Days that Delivery has to send most orders

Task 9

Future development of a data warehouse

9 Task 9: Future development of a data warehouse

9.1 Common uses of a Data warehouse, Factors that might lead Basket4U to build a data warehouse and How it would operate

A data warehouse or DWH is a centralized, secured, homogenous information repository which can be utilized to store, filter out and analyze data to make more well-informed decisions and more precise reporting. (Gierc, 2019). It also serves as single version of truth as it collects and integrates data from various sources within an organization. (Brown, 2021).

An organization can have many advantages from building a data warehouse rather than using transactional databases. Data warehouses uses Dimensional modeling technique to provide reliable and precise way for businesses to keep and access uniformly formatted data which will help improve data access from different sources within organization. An organization's decision-making persons, data analysts, and experts are benefited by speedy, precise, both historic and current data provided by data warehouse. (Gierc, 2019). While transactional databases are aimed only for inserting and updating data into the database, data warehouse aims for time efficient and faster data reading. Transactional databases with poor query performance due to joining multiple tables with large data rows will result in significantly long query time or even query timeout/ unfinished query to start performing analysis and business intelligence. (Brown, 2021). Data warehouse keeps copies of original data and therefore providing both historical and real-time data. (Gierc, 2019). Transactional database overwrites prior data values with most current value from transaction so that it is hard to track history of data changes, which is important for data analysts. (Brown, 2021)

However, Data warehouses also have some disadvantages. Regular maintenance cost for the data warehouse can become expensive over the long term. It may also be time consuming to prepare a data warehouse as someone has to manually input the raw data for data warehouse initially. (Anon., 2015)

As for Basket4U online grocery store, it will be necessary to build a data warehouse as the business expands to online presence, to multiple regions/ countries and the products, purchases, sales, orders systems transactions data will become larger. Systems will be transformed into multiple sources coming from multiple regions and original transactional database is no longer feasible for querying. Therefore, Data warehouse becomes necessary to integrate and handle

these separated systems together. In requirement of Business Intelligence purposes to survive and effectively compete for market share in a competitive environment, the Basket4U's decision makers and marketing, sales analysts always require to know both historical and current transactions of the Basket4U to discover whether sales target are hit, customer buying patterns, products that are not sold well, products that customers always buy all the time, customer spending powers and customer demands. They also need to know whether the demand (buying) power and supply power balance of Basket4U. Data insights offered by data warehouse will enable Basket4U to make effective decisions.

Task 10

Reflection

10 Task 10: Reflection

10.1 Reflecting on learning undertaken using Rolfe, G., Freshwater, D. and Jasper, M. (2001) model

10.1.1 Description

In this assignment, a database was required to be developed accordingly to a chosen organization based on the learnings, knowledge and skills gained during the Database Design and Development Module classes. An industry of interest and an organization is needed to be selected. Being a former business management student, I have always wondered how data digitally operates in a store/ in an e-commerce website / in a shopping mall. Therefore, I have chosen Basket4U Online Grocery Store with a main focus on Purchase and Order operations. Materials related to the selected organization was also crucial to be collected.

Before starting tasks, a brief introduction to chosen organization Basket4U was written. For Task 1, the reason why Basket4U is in need of a database, their daily transactions and operations, scope that will mainly considered in the database were stated together with five types of material documents with many forms gathered from Basket4U. By analyzing the document data from Task 1, in Task 2, possible entities are listed, relationships and multiplicity were thought accordingly to operations of Basket4U and an initial Entity Relationship Diagram (including 11 entities) was drawn. For each entity from ERD with dummies, Data dictionary was also created together with descriptive attribute names, description of use, data type, size, keys and constraints. In Task 3, Normalization was done for documents gathered in Task 1. Additionally, purpose of normalization, why each entity is in 3NF, checking of well-structured tables, how normalization solves update anomalies in design of Basket4U were written in Task 3. In task 4, evaluation of task done so far to Task 3 was done. From mapping of logical to physical database design, Many to many relationships were broken down into one to many relationships with dummies. Some attributes were separated as a new entity to make dynamic. Some were de-normalized to reduce unnecessary JOINS for faster query. Designed tables for 11 entities were made to refer when creating physical database in Microsoft SQL Server DBMS. In Task 5, Tables of strong entities were created first followed by weak ones together with Primary keys, Foreign keys, Domain, Propagation, Integrity constraints with “CREATE TABLE” command in SQL Server, query results and explanations were added. In Task 6, new data were inserted into created tables with “INSERT INTO” command. Parent tables were inserted first as there are constraints in child tables checking on respective parent table data. In Task 7, new columns were added to some tables with “ALTER TABLE” command in order to

complete Basket4U's operational requirements and updated with SQL codes as some columns serves as Derived column. For Task 9, a discussion on factors that Basket4U might need to build a Data warehouse, advantages and disadvantages of it, how Data warehouse would operate and for what purposes it will be used in Basket4U Online Grocery Store. In this task 10, reflection of learnings done in order to finish this assignment was evaluated.

10.1.2 Analysis

As for difficulties encountered and studies, there were some difficulties in gathering documentations from Basket4U so that creating example documents were not easy. Changes in initial ERD were made again and again as its entity relationships were not connecting as intended as the Basket4U operations do. Additional studies on how to create a Data Dictionary were done as I have not seen a data dictionary before. After Normalization in Task 3, there were some difference in ERD results compared to ERD before normalization so that Final decision of which entities to take in, which ones to be de-normalized or separated were made in Task 4 to be able to create physical tables conveniently. The problem encountered while creating tables was forgetting to choose (or) double-check if Basket4U Database is chosen before executing queries. There were times when wrong master database was chosen automatically and table was not created in actual Basket4U database. Additional studies on how to create tables with constraints were also made. As data to Insert into tables were a lot in some tables, double-checks were needed to be done before executing INSERT queries. Extensive studies on how to implement derived columns were needed as I have no prior experience before in doing so. For SQL Reports, some queries took a lot of time to tweak and to successfully work as intended as these were complex enough and must have a purpose of use. Extensive readings on what a Data warehouse is, how it operates in an organization, its pros and cons were made in order to complete Task 9.

10.1.3 Action Plan

Ultimately, the assignment tasks are all successfully done. However, in creating tables in assignment Task 5, derived columns had to be left in create table statements but later added in Task 7 with ALTER TABLE statements and updated with SQL queries because it was complex enough that I could not able to create them together in creating tables. Therefore, if a chance is given to improve the process next time, I will learn to create derived columns all together while creating tables with CREATE TABLE statement. For this assignment, only Purchases and Orders

operations were considered in scope but delivery operations, pricing systems, inventory management systems were not considered in depth for Basket4U due to data limitation and these system can be large and complex enough to be implemented. If improvements are needed to be done and more time is given, this assignment will be enhanced with pricing, delivery and inventory systems for Basket4U despite the complexity and large tasks.

References

1. Anon., 2015. *The Pros and Cons of Data Warehouses.* [Online]
   Available at: https://businessimpactinc.com/blog/the-pros-cons-of-data-warehouses/
   [Accessed 1 August 2022].

2. Biscobing, J., n.d. *What is Entity Relationship Diagram (ERD)?.* [Online]
   Available at: https://www.techtarget.com/searchdatamanagement/definition/entity-relationship-diagram-ERD
   [Accessed 29 June 2022].

3. Brown, M., 2021. *What are the differences between a Data Warehouse and a Transactional Database?.* [Online]
   Available at: https://waterloodata.com/differences-between-data-warehouse-and-transactional-database/?fbclid=IwAR0nb5wVYtQPNPZjnk4Cac9-u2tcS7ceC9FbfaKOoDaRFFS43xkjyZS72oQ
   [Accessed 1 August 2022].

4. Derda, M., 2020. *What is a data dictionary?.* [Online]
   Available at: https://www.trifacta.com/blog/data-dictionary/#:~:text=Matt%20Derda
   [Accessed 28 June 2022].

5. Dhyawala, S., 2018. *Denormalization in Databases.* [Online]
   Available at: https://www.geeksforgeeks.org/denormalization-in-databases
   [Accessed 8 July 2022].

6. Gierc, M., 2019. *Why your business needs a data warehouse.* [Online]
   Available at: https://www.datasciencecentral.com/why-your-business-needs-a-data-warehouse/
   [Accessed 26 July 2022].

7. IBM, 2021. *Resolve m:n relationships.* [Online]
   Available at: https://www.ibm.com/docs/en/informix-servers/14.10?topic=relationships-resolve-mn
   [Accessed 8 July 2022].

8. Jennifer, n.d. *What Is A Well Structured Relation In Database Design?.* [Online]
   Available at: https://www.rkimball.com/what-is-a-well-structured-relation-in-database-design/
   [Accessed 10 July 2022].

9. SINGH, C., 2015. *Normalization in DBMS: 1NF, 2NF, 3NF and BCNF in Database.* [Online]
   Available at: https://beginnersbook.com/2015/05/normalization-in-dbms
   [Accessed 5 July 2022].

10. Watt, A., 2014. *Chapter 12 Normalization.* [Online]
    Available at: https://opentextbc.ca/dbdesign01/chapter/chapter-12-normalization
    [Accessed 5 July 2022].