Database Design for Genie Delivery Service

Task – 1

Task 1 – Business Overview and Requirements

Scenario

"Genie" is a delivery service company based in Yangon, Myanmar. They provide delivery services for various items – small to medium size, from a variety of shops in many townships. Items are delivered to customers by bike riders.

They want a database to help them manage ordering and delivering items as well as managing riders. The database will not be concerned, at least initially, with the allocation of payments to riders.

"Genie" receives tons of orders every day for items from different shops. Each customer can make many orders a day from different locations. Each order is assigned to one rider, who lives in the same township as the customer's delivery address. Customers are defined by member types – bronze, silver, gold and premium.

Payment can be made by different payment methods – cash on delivery, bank transfer and local digital money wallets such as KBZ Pay, Wave Pay or CB Pay. A single payment will have one specific payment method allocated to it.

Each rider will have several orders allocated to him and each order might include more than one item. A shop can have many items and each item is assumed to be associated with only one shop. Shops are defined by shop categories.

Examples of data are as shown in the tables below.

Sample Documents

Document 1 – Customers and Riders on Order Forms

Customer Name	Customer ID	Member Type	Order ID	Rider ID	Rider Name	Township
Kyaw Thu	GC-0012	Bronze	ORD-3111	GR-07	Soe Win	Bahan
Kyaw Thu	GC-0012	Bronze	ORD-3114	GR-08	Hla Myo Tun	Hlaing
Kyaw Thu	GC-0012	Bronze	ORD-3118	GR-09	Myint Thein	Dagon
Kyaw Thu	GC-0012	Bronze	ORD-3122	GR-07	Soe Win	Bahan
Hla Nu	GC-0028	Silver	ORD-3131	GR-12	Moe Hein	Hlaing
Hla Nu	GC-0028	Silver	ORD-3135	GR-13	Phyo Kyaw	Latha
Hla Nu	GC-0028	Silver	ORD-3138	GR-08	Hla Myo Tun	Hlaing
Htun Zaw	GC-0093	Gold	ORD-3143	GR-14	Thuta	Tamwae
Htun Zaw	GC-0093	Gold	ORD-3147	GR-12	Moe Hein	Hlaing
Htun Zaw	GC-0093	Gold	ORD-3149	GR-12	Moe Hein	Hlaing
Htun Zaw	GC-0093	Gold	ORD-3153	GR-10	Min Aung	Bahan
Maung Paing	GC-0004	Premium	ORD-3155	GR-11	Htun Aye	Bahan
Maung Paing	GC-0004	Premium	ORD-3158	GR-14	Thuta	Tamwae
Maung Paing	GC-0004	Premium	ORD-3170	GR-11	Htun Aye	Bahan

Document 2 – Items and Order Details

Order ID	Order Date	Payment Type	Item Name	Item Code	Quantity
ORD-3111	16/8/2022	KBZ Pay	Cloth Face Mask	I-1090	3
ORD-3111	16/8/2022	KBZ Pay	Air-X Tablet	I-1086	1
ORD-3111	16/8/2022	KBZ Pay	Y-Si Vitamin C	I-1077	1
ORD-3114	18/8/2022	COD	Cheesecake Slice	I-2144	3
ORD-3114	18/8/2022	COD	Vanilla Ice Cream	I-2150	1
ORD-3114	18/8/2022	COD	Chicken Curry Puff	I-2138	1
ORD-3118	1/9/2022	AYA Pay	Remax TWS Ear Buds	I-2048	1
ORD-3118	1/9/2022	AYA Pay	Green Tech wireless mouse	I-2067	1
ORD-3122	2/9/2022	COD	Cosrx Acne Patch	I-3144	4
ORD-3122	2/9/2022	COD	A&C Mask Sheet	I-3178	3
ORD-3135	4/9/2022	KBZ Pay	Nestle Milo Jar	I-0126	2
ORD-3135	4/9/2022	KBZ Pay	Jordan Toothbrush	I-0130	6
ORD-3135	4/9/2022	KBZ Pay	Shark Energy Drink Can	I-0148	3

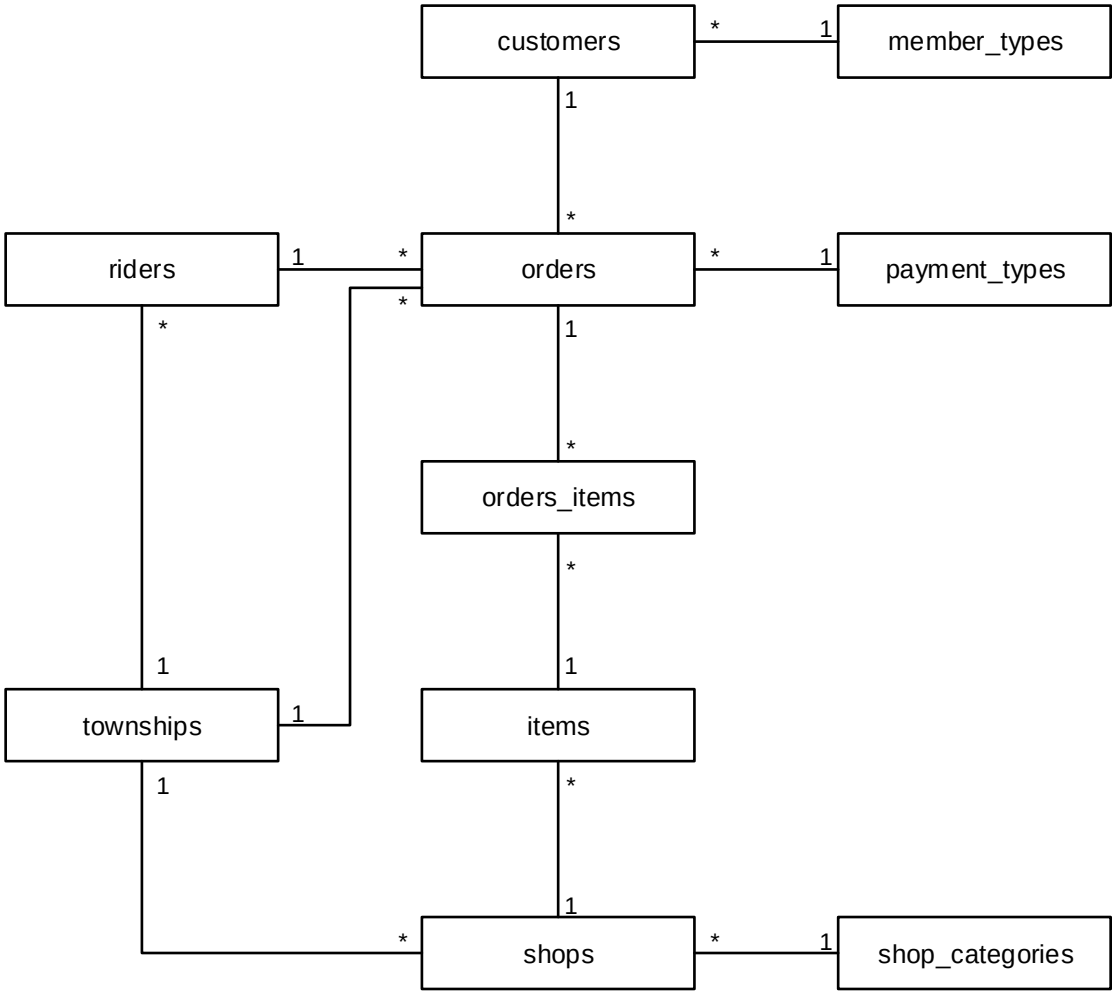
Document 3 – Shops and Items

Shop Name	Shop ID	Shop Category	Township	Item Code	Item Name	Unit Price (Kyats)
Moe's Bakery	S-097	Food	Hlaing	I-2144	Cheesecake Slice	4200
Moe's Bakery	S-097	Food	Hlaing	I-2145	Chocolate Cake	4000
Moe's Bakery	S-097	Food	Hlaing	I-2150	Vanilla Ice Cream	2700
Moe's Bakery	S-097	Food	Hlaing	I-2138	Chicken Curry Puff	5800
Shwe Oh Pharmacy	S-045	Health	Bahan	I-1077	Y-Si Vitamin-C	3200
Shwe Oh Pharmacy	S-045	Health	Bahan	I-1086	Air-X Tablet	1600
Shwe Oh Pharmacy	S-045	Health	Bahan	I-1090	Cloth Face Mask	1200
7Eleven	S-019	Grocery	Latha	I-0126	Nestle Milo Jar	6500
7Eleven	S-019	Grocery	Latha	I-0130	Jordan Toothbrush	2400
7Eleven	S-019	Grocery	Latha	I-0148	Shark Energy Drink Can	1400
Beauty Diary	S-114	Beauty	Bahan	I-3144	Cosrx Acne Patch	1800
Beauty Diary	S-114	Beauty	Bahan	I-3178	A&C Mask Sheet	2700
Tech Guru	S-012	Electronic	Dagon	I-2048	Remax TWS Ear Buds	32000
Tech Guru	S-012	Electronic	Dagon	I-2067	Green Tech wireless mouse	1650

Task – 2

Task 2 – ER and Data Dictionary

Entity Relationship Diagram



Data Dictionary

Table Name: townships

Primary Key: township_code

Foreign Key: totownnone

Attribute	Data Type	Size	Domain Constraint	Description
township_code	Integer	-	Unique Not Null Must be greater than 11000	Unique zip code for the township. Zip codes for Yangon region townships starts with '11'.
township_name	Varchar	50	Not Null	The name of the township

Table Name: member_types

Primary Key: member_type_id

Foreign Key: none

Attribute	Data Type	Size	Domain Constraint	Description
member_type_id	Integer	-	Unique Not Null	Unique identifier for the member type
member_type	Varchar	16	Not Null	Name of the member type

Table Name: shop_categories

Primary Key: shop_category_id

Foreign Key: none

Attribute	Data Type	Size	Domain Constraint	Description
shop_category_id	Integer	-	Unique Not Null	Unique identifier for the shop category
shop_category_name	Varchar	16	Not Null	Name of the shop category

Table Name: payment_types

Primary Key: payment_type_id

Foreign Key: none

Attribute	Data Type	Size	Domain Constraint	Description
payment_type_id	Integer	-	Unique Not Null	Unique identifier for the payment type
payment_type_name	Varchar	16	Not Null	Name of the payment type

Table Name: riders

Primary Key: rider_id

Foreign Key: township_code

Attribute	Data Type	Size	Domain Constraint	Propagation Constraint	Description
rider_id	Varchar	8	Unique Not Null Must start “R-” following by a sequential number	-	Unique identifier for the rider
township_code	Integer	-	Not Null Must be greater than 11000	On Update Cascade On Delete Restrict	Zip code of the township where the rider works
rider_name	Varchar	50	Not Null	-	Name of the rider
rider_phone	Integer	-	Not Null	-	Contact Number of the rider
rider_address	Varchar	100	Not Null	-	Address of the rider
date_of_birth	Date	-	Not Null	-	Rider’s birth date
age	Date	-	-	-	Rider’s age will be derived from date_of_birth field and it will be virtual
rider_registered_date	Date	-	Default current date	-	Registration date of the rider

Table Name: customers

Primary Key: customer_id

Foreign Key: member_type_id

Attribute	Data Type	Size	Domain Constraint	Propagation Constraint	Description
customer_id	Varchar	8	Unique Not Null Must start “C-” following by a sequential number	-	Unique identifier for the customer
member_type_id	Integer	-	Not Null	On Update Cascade On Delete Restrict	Member type id for the customer
customer_name	Varchar	50	Not Null	-	Full name of the customer
customer_email	Varchar	50	Not Null	-	Email of the customer
customer_phone	Integer	-	Not Null	-	Contact number of the customer
customer_registered_date	Date	-	Default current date	-	The date at when the customer registered at Genie Delivery

Table Name: shops

Primary Key: shop_id

Foreign Keys: township_code, shop_category_id

Attribute	Data Type	Size	Domain Constraint	Propagation Constraint	Description
shop_id	Varchar	8	Unique Not Null Must start with “S-” followed by a sequential number	-	Unique identifier of the shop
township_code	Integer	-	Not Null Must be greater than 11000	On Update Cascade On Delete Restrict	Zip code of the township the town is located
shop_category_id	Integer	-	Not Null	On Update Cascade On Delete Restrict	Category ID for the shop
shop_name	Varchar	50	Not Null	-	Name of the shop
shop_address	Varchar	100	Not Null	-	Address of the shop

Table Name: Items

Primary Key: item_code

Foreign Keys: shop_id

Attribute	Data Type	Size	Domain Constraint	Propagation Constraint	Description
item_code	Varchar	8	Unique Not Null Must start with “I-” followed by a sequential number	-	Unique identifier for the item
shop_id	Varchar	8	Not Null Must start with “S-” followed by a sequential number	On Update Cascade On Delete Restrict	The code of the shop from which the item is available
item_name	Varchar	50	Not Null	-	Name of the item
price	Decimal	(8,2)	Not Null	-	Unit price of the item
in_stock	Integer	-	Not Null	-	The number of in-stock items available at the shop

Table Name: orders

Primary Key: order_id

Foreign Keys: customer_id, rider_id, payment_type_id, township_code,

Attribute	Data Type	Size	Domain Constraint	Propagation Constraint	Description
order_id	Varchar	16	Unique Not Null Must start with “ORD-” followed by a sequential number	-	Unique identifier of the order
customer_id	Varchar	8	Not Null Must start “C-” followed by a sequential number	On Update Cascade On Delete Restrict	The ID of the customer who made the order
rider_id	Varchar	8	Not Null Must start “R-” followed by a sequential number	On Update Cascade On Delete Restrict	The ID of the rider who will be delivered the order
payment_type_id	Varchar	8	Not Null	On Update Cascade On Delete Restrict	The ID of the payment type for the order
township_code	Integer	-	Not Null Must be greater than 11000	On Update Cascade On Delete Restrict	Zip code of the township where the order is to be delivered

order_date	Date	-	Default current date	-	The date on when the customer made the order
order_time	Time	-	Default current time	-	The exact time at when the customer made the order
delivered_time	Time	-	-	-	The exact time at when the order is delivered by the rider
delivery_address	Varchar	100	Not Null	-	The address the order is to be delivered
status	Varchar	16	Default “pending” Must be either “pending”, “delivered” or “canceled”	-	The status of the order whether it is pending, delivered or canceled

Table Name: orders_items

Primary Key: order_id + item_id

Foreign Keys: order_id, item_id

Attribute	Data Type	Size	Domain Constraint	Propagation Constraint	Description
order_id	Varchar	16	Not Null Must start with “ORD-” followed by a sequential number	On Update Cascade On Delete Restrict	Unique identifier of the order
item_id	Varchar	8	Not Null Must start with “I-” followed by a sequential number	On Update Cascade On Delete Restrict	Unique identifier of the item
quantity	Integer	-	Not Null	-	Total quantity of the item in the order

Task – 3

Task 3 – Normalisation

Purpose of Normalisation

Normalisation is the process of organizing data in a database. The main purpose of normalisation is to minimize redundancy (duplicate data) and to overcome potential anomalies (eliminate inconsistent dependency). Normalisation can even increase security, improve workflow and lessen costs.

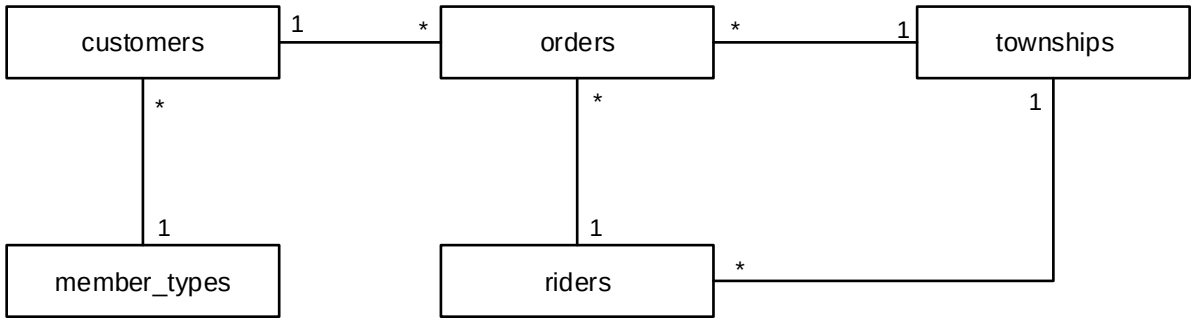
Normalisation for Document 1 – Customers and Riders on Order Forms

UNF	Level	1NF	2NF	3NF	Optimization
CustomerID	1	CustomerID(PK)	CustomerID(PK)	CustomerID(PK)	**[customers]**
CustomerName	1	CustomerName	CustomerName	CustomerName	customer_id(PK)
MemberType	1	MemberType	MemberType	MemberTypeID(FK)	member_type_id(FK)
OrderID	2				customer_name
RiderID	2	OrderID(PK)	OrderID(PK)	MemberTypeID(PK)	customer_email
RiderName	2	CustomerID(FK)	CustomerID(FK)	MemberType	customer_phone
Township	2	RiderID	RiderID(FK)		customer_address
		RiderName		OrderID(PK)	registered_date
		Township	RiderID(PK)	CustomerID(FK)	
			RiderName	RiderID(FK)	**[member_types]**
			Township	TownshipCode(FK)	member_type_id(PK)
					member_type
				RiderID(PK)	
				RiderName	**[orders]**
				TownshipCode(FK)	order_id(PK)
					customer_id(FK)
				TownshipCode(PK)	rider_id(FK)
				TownshipName	township_code(FK)
					order_date
					order_time
					delivered_date

					deliverey_address **[riders]** rider_id(PK) township_code(FK) rider_name rider_phone rider_address date_of_birth registered_date **[townships]** township_code(PK) towship_name

After normalisation for the first document, five tables are obtained. Although member types can be embedded in “Customers” table, it is decided to use a separate table aiming to store more information about each member. The same concept applies for townships table. It has no dependencies, yet a separate table is used. Another reason to store township in a separate table is that it is related to more than one entity. The ER Diagram is as shown in the following figure.

Entity Relationship Diagram

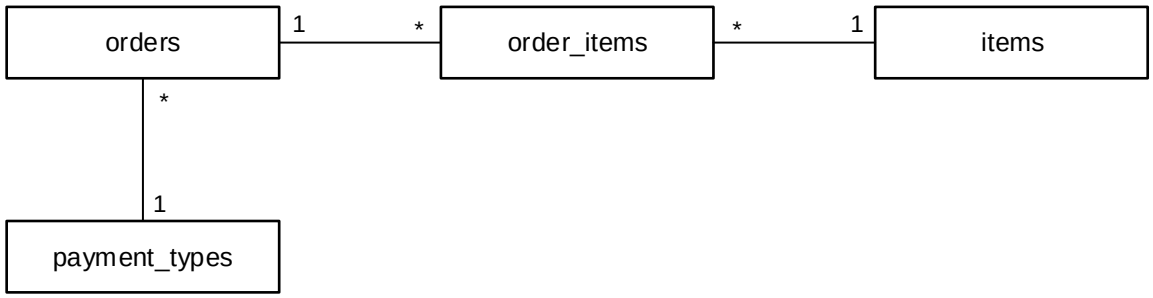


Normalisation for Document 2 – Items and Order Details

UNF	Level	1NF	2NF	3NF	Optimization
OrderID	1	OrderID(PK)	OrderID(PK)	OrderID(PK)	**[orders]**
OrderDate	1	OrderDate	OrderDate	OrderDate	order_id(PK)
PaymentType	1	PaymentType	PaymentType	PaymentTypeID(FK)	payment_type_id(FK)
ItemCode	2				order_date
ItemName	2	OrderID(FK) ⎫ PK	OrderID(FK,PK)	PaymentTypeID(PK)	order_time
Quantity	2	ItemCode ⎭ PK	ItemCode(FK,PK)	PaymentType	delivered_time
		ItemName	Quantity		delivery_address
		Quantity		OrderID(FK,PK)	
			ItemCode(PK)	ItemCode(FK,PK)	**[payment_types]**
			ItemName	Quantity	payment_type_id(PK)
					payment_type
				ItemCode(PK)	
				ItemName	**[orders_items]**
					order_id(FK,PK)
					item_code(FK,PK)
					quantity
					[items]
					item_code(PK)
					item_name
					price
					in_stock

The normalisation for the second document gives four tables. Orders and Items are related to each other with many to many relationship resulting a link (dummy) table with a composite primary key. Payment types are designed to store in a separate table to make the schema more flexible and to store additional information for each payment type. The ER Diagram is as shown below.

Entity Relationship Diagram

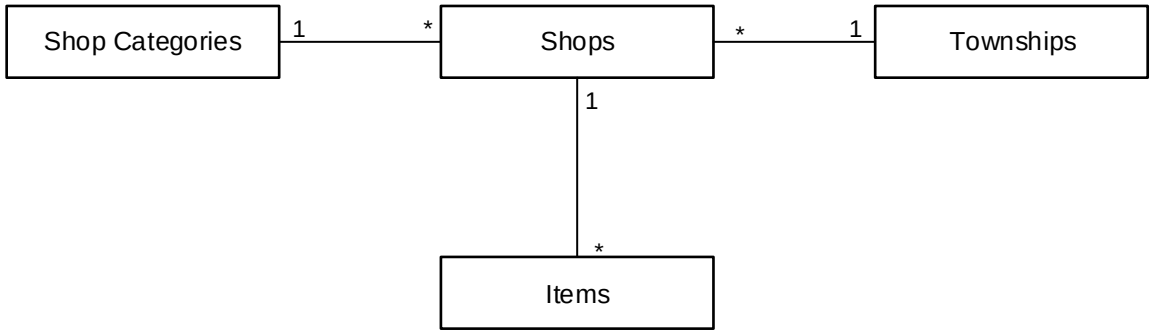


Normalisation for Document 3 – Shops and Items

UNF	Level	1NF	2NF	3NF	Optimization
ShopID	1	ShopID(PK)	ShopID(PK)	ShopID(PK)	**[shops]**
ShopName	1	ShopName	ShopName	ShopName	shop_id(PK)
ShopCategory	1	ShopCategory	ShopCategory	ShopCategoryID(FK)	shop_category_id(FK)
Township	1	Township	Township	TownshipCode(FK)	township_code(FK)
ItemCode	2				shop_name
ItemName	2	ItemCode(PK)	ItemCode(PK)	ShopCategoryID(PK)	shop_address
UnitPrice	2	ShopID(FK)	ShopID(FK)	CategoryName	
		ItemName	ItemName		**[shop_categories]**
		UnitPrice	UnitPrice	TownshipCode(PK)	shop_category_id(PK)
				TownshipName	shop_category_name
				ItemCode(PK)	**[townships]**
				ShopID(FK)	township_code(PK)
				ItemName	township_name
				UnitPrice	
					[Items]
					item_code(PK)
					shop_id(FK)
					item_name
					price
					in_stock

Normalisation for the third document gives four related tables. According to the given document, townships can be embedded into “shops” table. But it may be related to other entities and so it is decided to store in a separate table. The ER Diagram is as shown below.

Entity Relationship Diagram



How normalisation solves possible anomalies

Insert Anomalies

It is said that an order can be paid with different payment types. Suppose that payment types are embedded into orders. If we want to allow customers to use a new payment type and no one has not used that, it is not possible to insert into database because we need an order_id (primary key). Normalisation gives use a separate table for payment types allowing us to insert new methods and other information.

Update Anomalies

Alternative approach to designing shops and items is to embed shops in items table. But if a shop changes its name for some reason, we need to update every items from that shop. After normalisation, shops and items are split into separate tables allowing us to overcome the above problem.

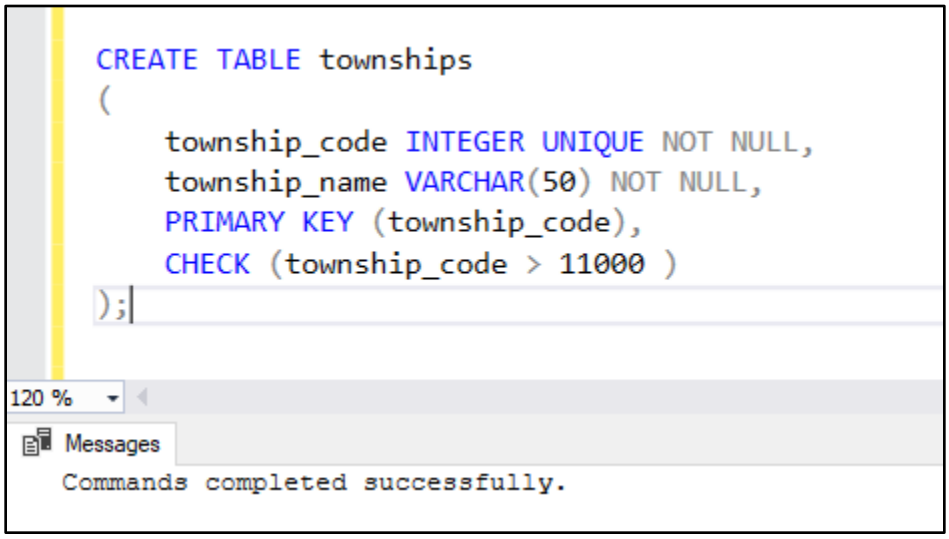
Delete Anomalies

This is a type of anomaly that cause losing information we don’t want to lose, because it is tied to data where it shouldn’t be. Normalisation helps us to overcome this kind of situation by placing every data in its related table and by splitting into separate tables if needed.

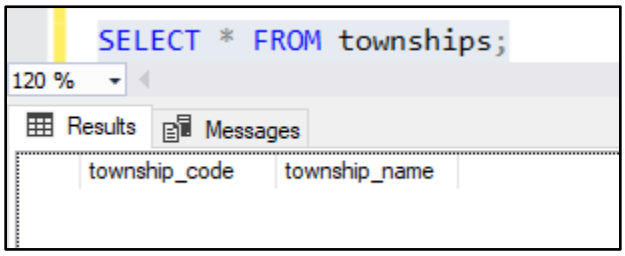
Task – 4

Task 4 – Scripts to create table structures

Townships Table



Result



Member Type Table

Result

Shop Categories Table

Result

Payment Type Table

Result

Riders Table

Result

Customers Table

Result

Shops Table

Result

Items Table

Result

Orders Table

Result

OrderItems Table

Result