Database Design and Implementation

MOD002589

Faculty: Science and Technology

Department: Computing and Technology

Academic Year: 2020/21

Trimester: 1

Contents

1 Requirement Analysis (15%)..... 4
1.1 Description of the three websites chosen..... 4
1.1.1 http://www.pizzahut.ae 4
1.1.2 http://www.papajohns.ae 4
1.1.3 http://www.dominos.ae 4
1.2 List of data fields (Entities and their attributes)..... 4
1.2.1 List of data fields from website 1 5
1.2.2 List of data fields from website 2 13
1.2.3 List of data fields from website 3 21
1.3 Finalised List 29
1.3.1 The generic list 29
2 Database design (25%) 30
2.1.2 Extended Entity Relationship Model 31
2.2 Normalised Model 32
2.3 Database Schema..... 34
3. Database implementation (10%) 40
4. SQL Queries (50%) 53
4.1 Query 1 54
4.1.1 For what purpose will this query be used in business terms?.... 54
4.1.2 Query in natural language 54
4.1.3 SQL Code and output..... 54
4.1.4 Explain the output of the data (was this what was predicted?) .. 54
4.2 Query 2 55
4.2.1 For what purpose will this query be used in business terms?.... 55
4.2.2 Query in natural language 55

4.2.3 SQL Code and output 55
4.2.4 Explain the output of the data (was this what was predicted?) .. 55
4.3 Query 3 56
4.3.1 For what purpose will this query be used in business terms?.... 56
4.3.2 Query in natural language 56
4.3.3 SQL Code and output 56
4.3.4 Explain the output of the data (was this what was predicted?) .. 56
4.4 Query 4 57
4.4.1 For what purpose will this query be used in business terms?.... 57
4.4.2 Query in natural language 57
4.4.3 SQL Code and output 57
4.4.4 Explain the output of the data (was this what was predicted?) .. 58
4.5 Query 5 59
4.5.1 For what purpose will this query be used in business terms?.... 59
4.5.2 Query in natural language 59
4.5.3 SQL Code and output 59
4.5.4 Explain the output of the data (was this what was predicted?) .. 60
5. References 61

1. Requirement Analysis

1.1 Description of the three websites chosen

All the three websites chosen are the storefront of companies that compete within the same industry: Restaurants. After analysing the website's purchasing function involved registration of a new customer, browsing different types of pizzas, selecting a pizza, adding it to cart and finally making a payment.

1.1.1 www.pizzahut.ae

1.1.2 www.papajohns.ae

1.1.3 www.dominos.ae

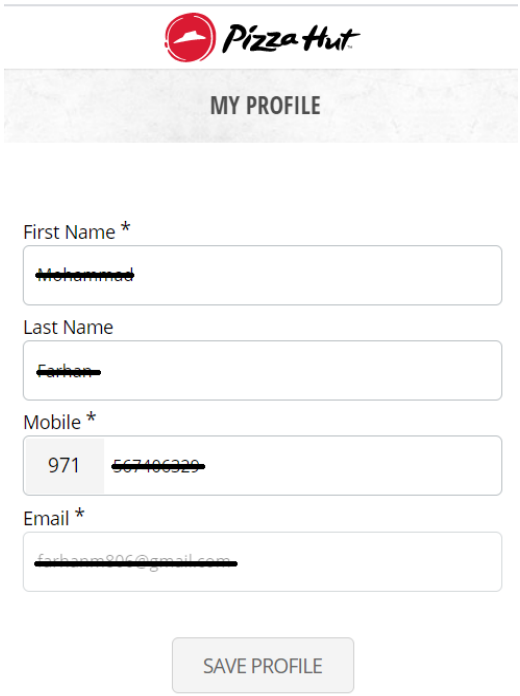
1.2 List of Data-fields

Three lists of data-fields are presented in the sub-sections below. The attributes have been collected after analysing the steps required to purchase a pizza on all of the three websites successfully. The steps include: Login/Sign-up a new customer, browsing the pizzas', browsing products other than the pizza, adding the products selected to cart, option to add extras , inputting the delivery address, making a payment, and track the order.

Some of the attributes are selected based on the user's input, others were selected based on the information displayed on the website.

1.2.1 List of data-fields from pizzahut.ae website

- **My Profile page**



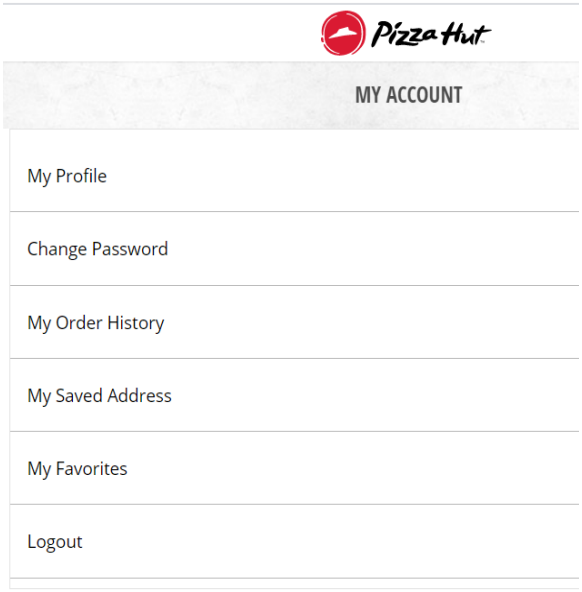


Figure 1: Screenshot of “My Profile” section of “My Account” page on the Pizza Hut website.

- **Address Page**

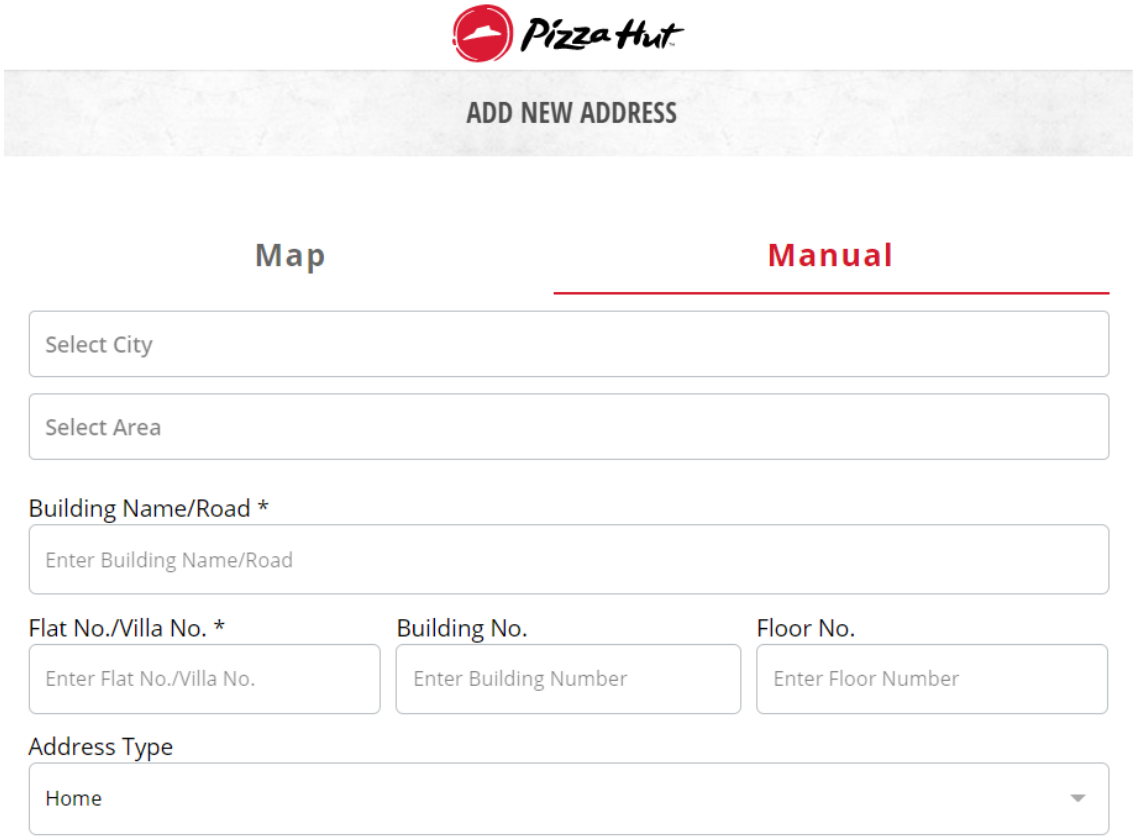


Figure 2: Screenshot of “Add new address” page on the Pizza Hut website.

.

- **Menus Page**

Figure 3: Screenshot of the “Menus” on the Pizza Hut website

This screenshot shows the different sections on the website that have a name each and upon clicking a section it would show the description of the chosen section.

- **Pizza Menu**

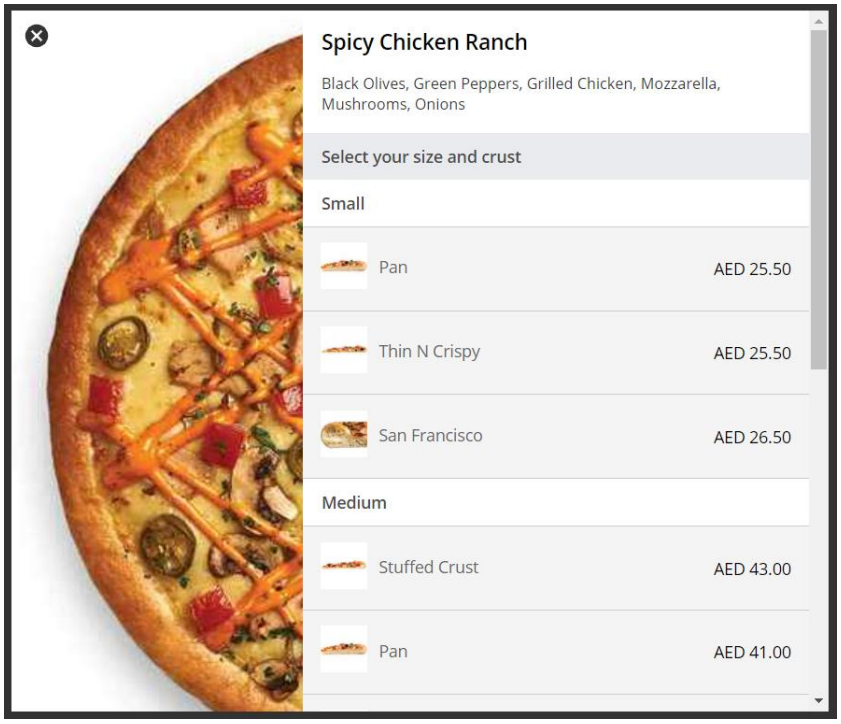
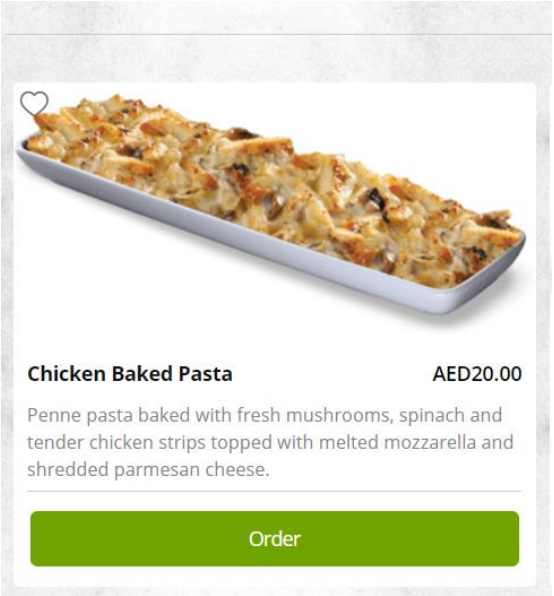


Figure 4: Screenshot of the list of attributes specific to the pizza on Pizza Hut website.

- **Non-Pizza**



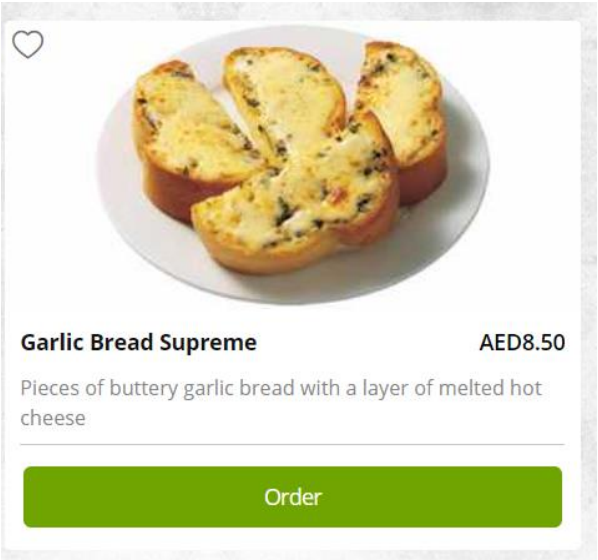


Figure 5: Screenshot of list of attributes of specific to non-pizza items in Pizza Hut website.

- **Your Cart**

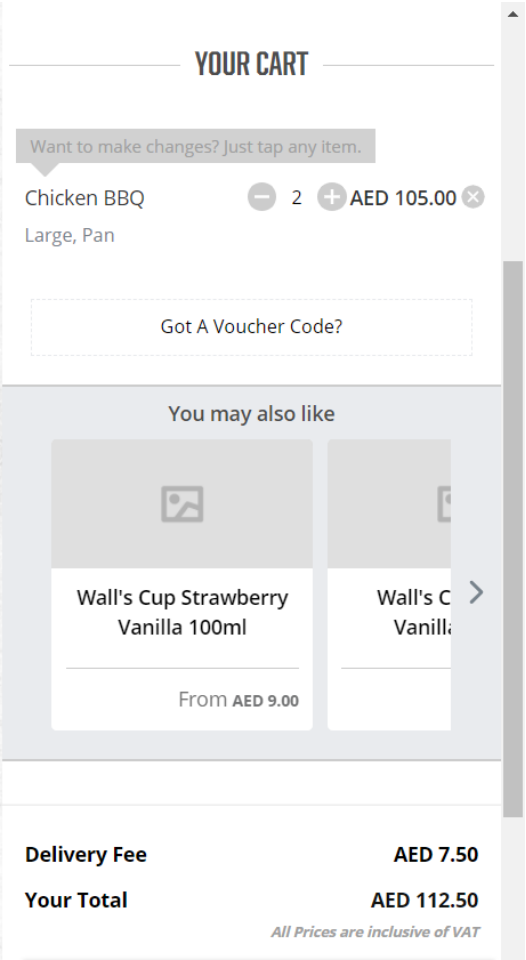


Figure 6: Screenshot of the Your Cart page on Pizza Hut website.

- **Orders Details**

Pizza Hut

Hi VAHEED MOHAMMAD

Thank you for ordering from www.pizzahut.ae!

Order Details

Order ID: 57076117

10-Nov-2020

Your ordered can be collected from

Pizza Hut Hamdan St. - Abu Dhabi,
Hamdan Street , Building 10
Get directions to our store

What you ordered:

1 X BUY 1 PIZZA GET 1 FREE

- +Chicken Super Supreme (Pan, Large) AED 56.500
- +Spicy Chicken Ranch (Pan, Large)

Payment: Online Payment

Grand Total AED 56.500

Figure 7: Screenshot of “Order summary” section on the Pizza Hut website.

- **Payment**

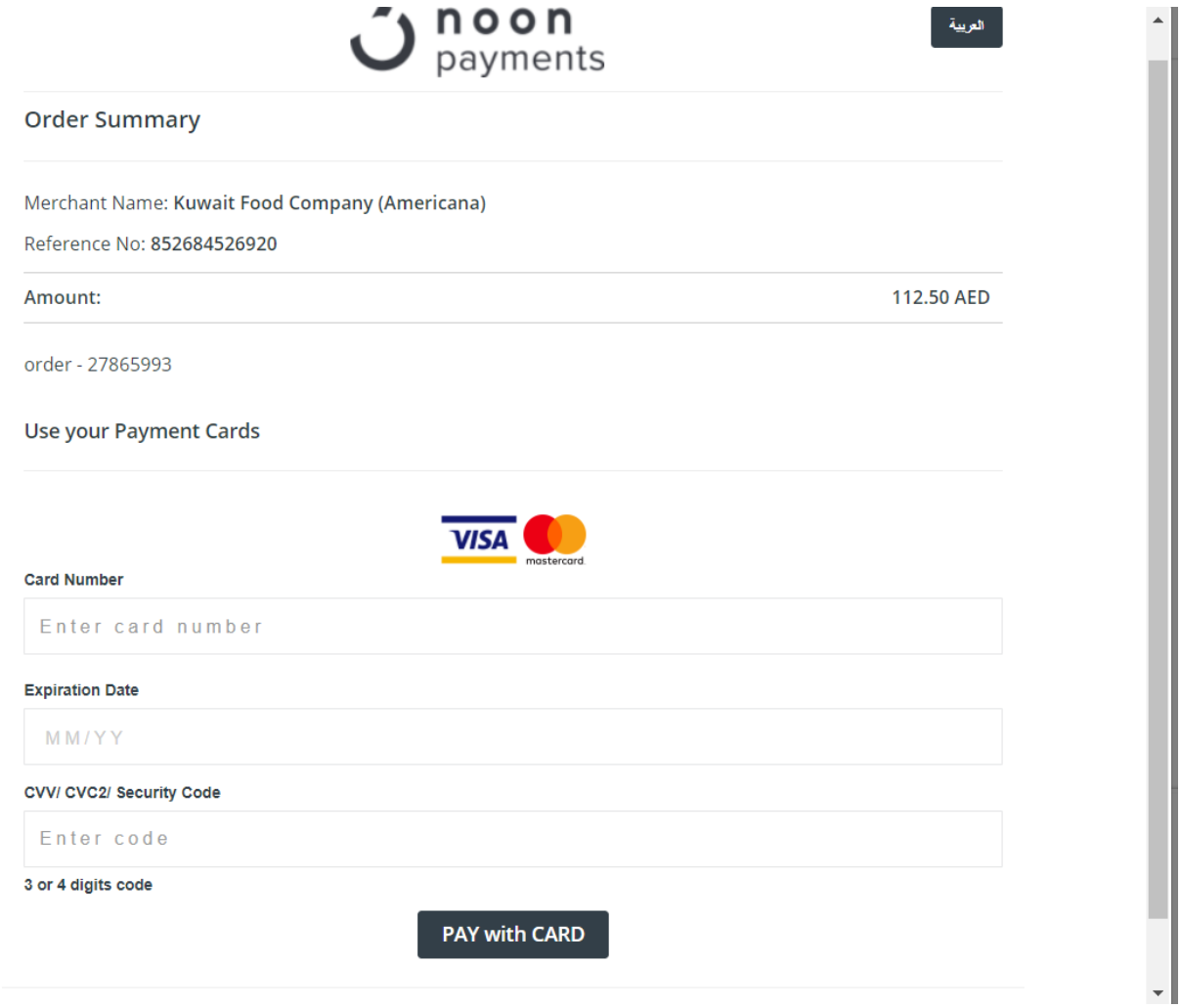


Figure 8: Screenshot of “Payment” page on the Pizaa Hut website.

Table 1, below, contains data-fields extracted during the analysis of the pizahut.ae website.

CUSTOMER	**PIZZA**	Delivery_fee
First_Name	Pizza_name	Total_price
Last_Name	Description	**ORDERS**
Mobile	Size	Order_number
Email	Crust	Order_date
Password	Add-ons	Restaurant_address
Orders*	Price	Order_items(*)
Address*	**NON-PIZZA**	Order_price
Favourites*	Name	Delivery_fee
ADDRESS	Description	Total_price
City	Price	Payment_info
Area	Menu	**PAYMENT**
Building_name	**CART**	Merchant_name
Flat_no	Food_name	Reference_number
Building_no	Food_description	Amount_pay
Floor_no	Food_quantity	Order_number
Address_Type	Price	Card_number
MENUS	Voucher_code	Expiry_date
Name	Recommendations	Security_code
Description	Price_from	

1.2.2 List of data-fields from dominos.ae website.

- **My Profile Page**

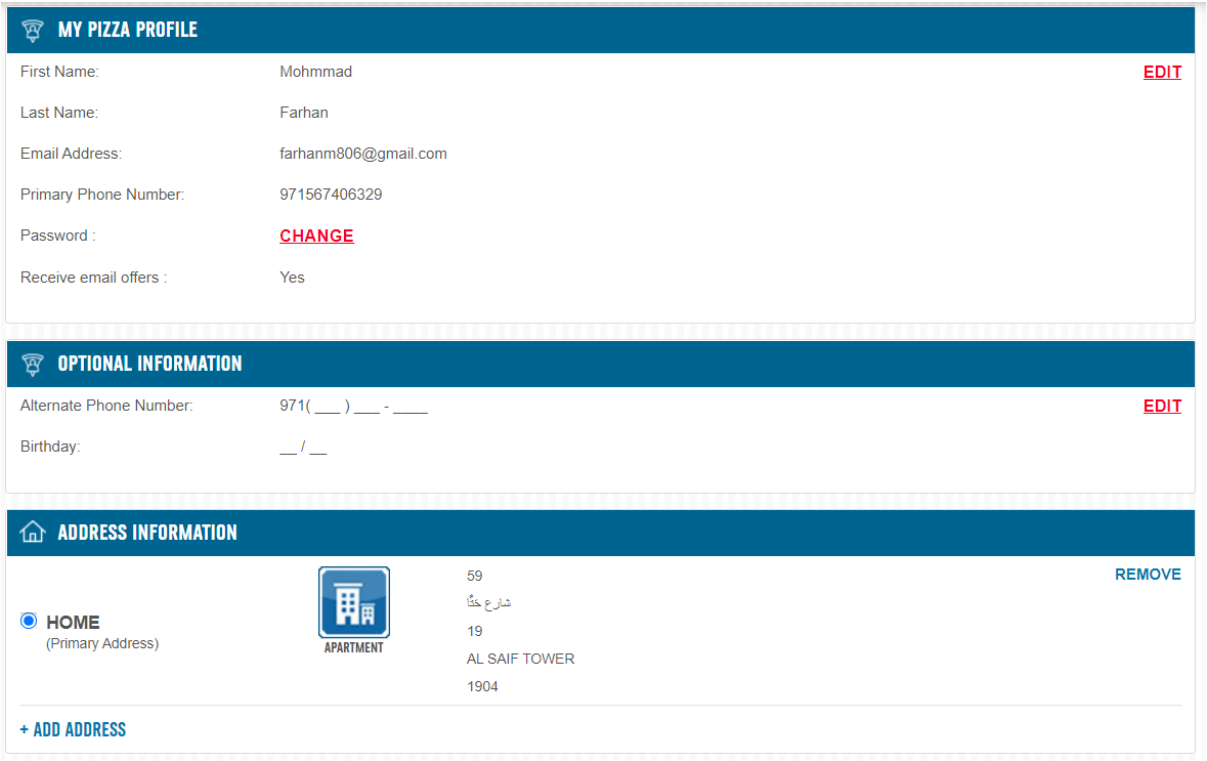


Figure 8: Screenshot of the “My profile” page on the Dominos website.

- **Address Page**

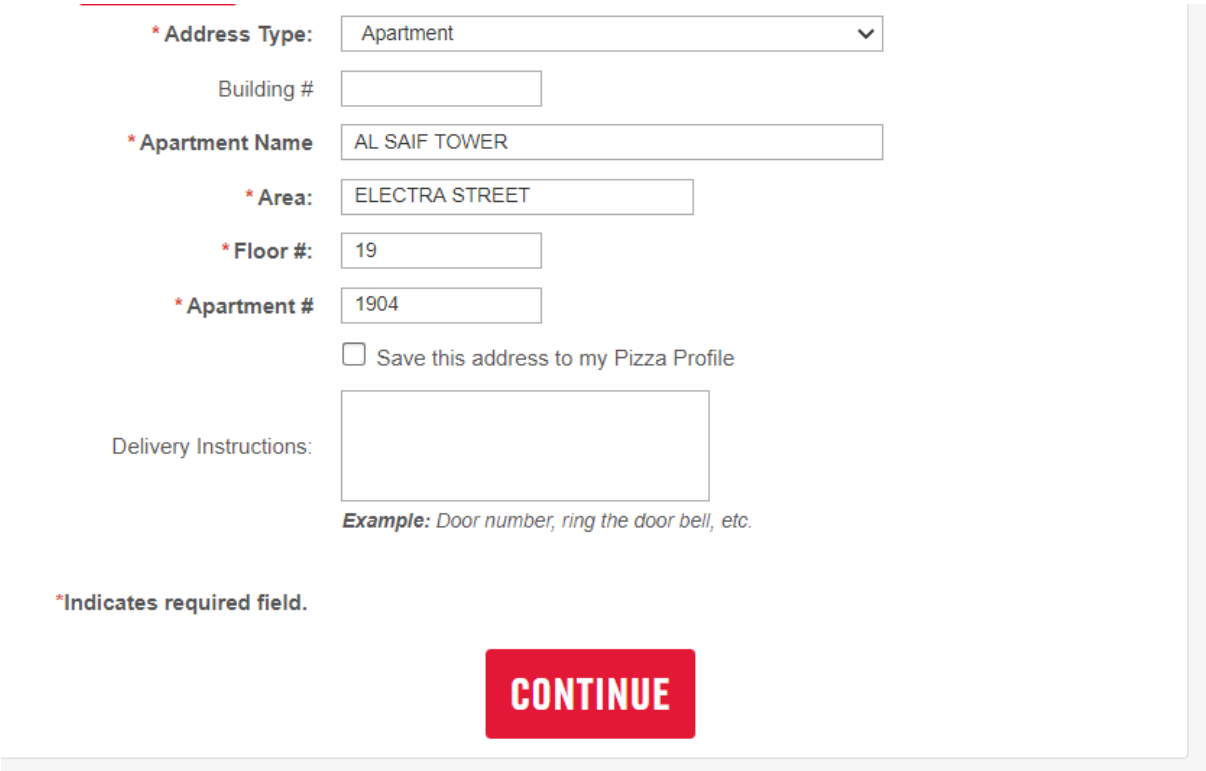


Figure 9: Screenshot of “Add address” page on Dominos website.

- **Menus**

Figure 10: Screenshot of the “menus” of the Dominos website.

- **Pizza Menu**

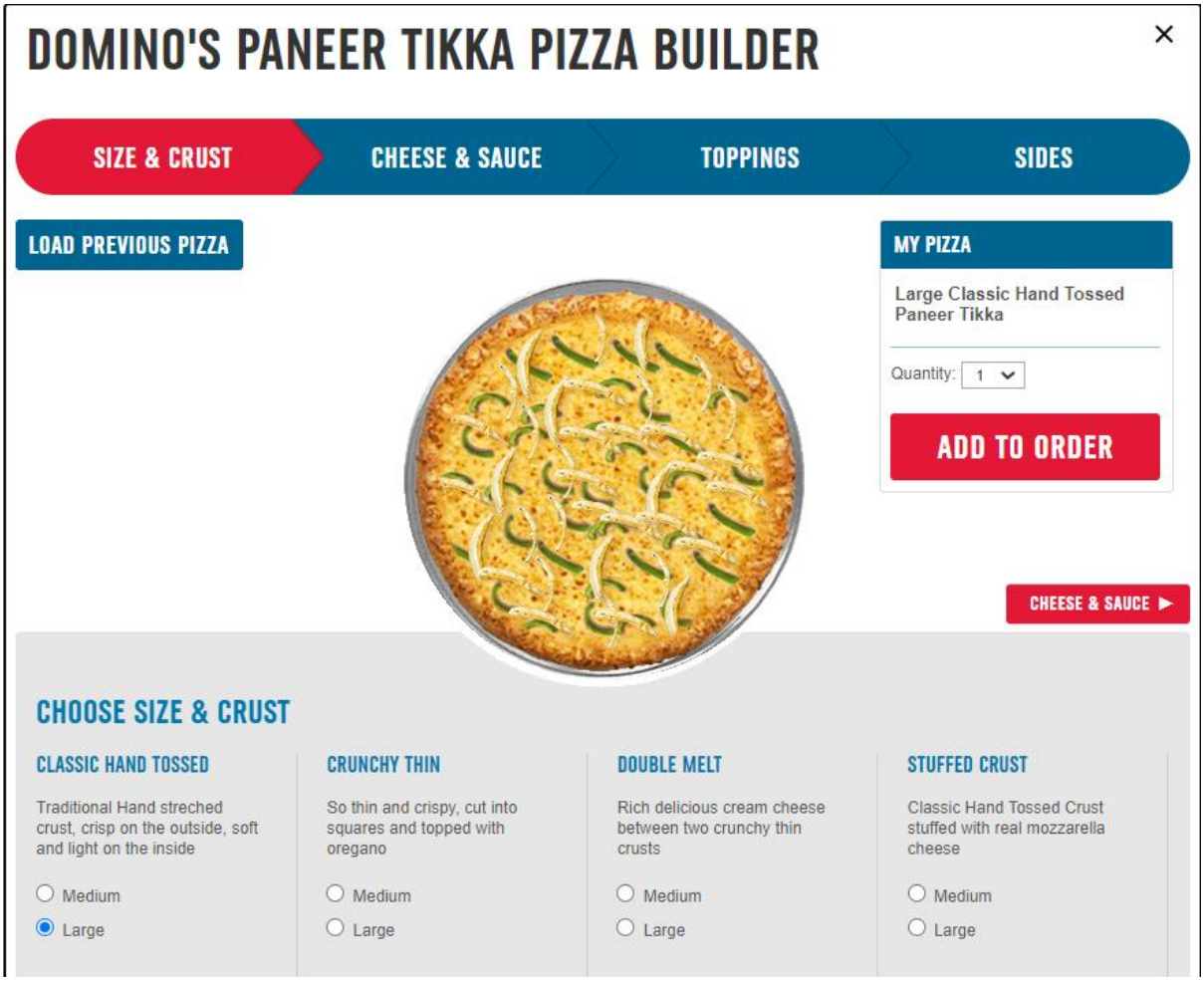


Figure 11: Screenshot of the “Pizza menu” of the Dominos website.

- **Non_pizza Menu**

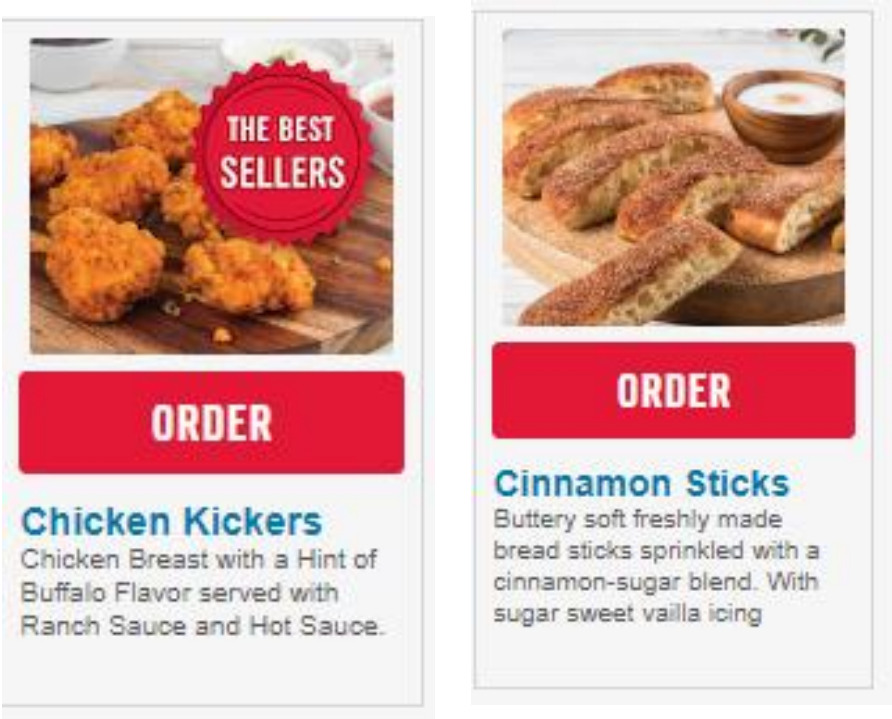


Figure 12: Screenshot of Non-Pizza items on the Dominos website.

- **Your Cart**

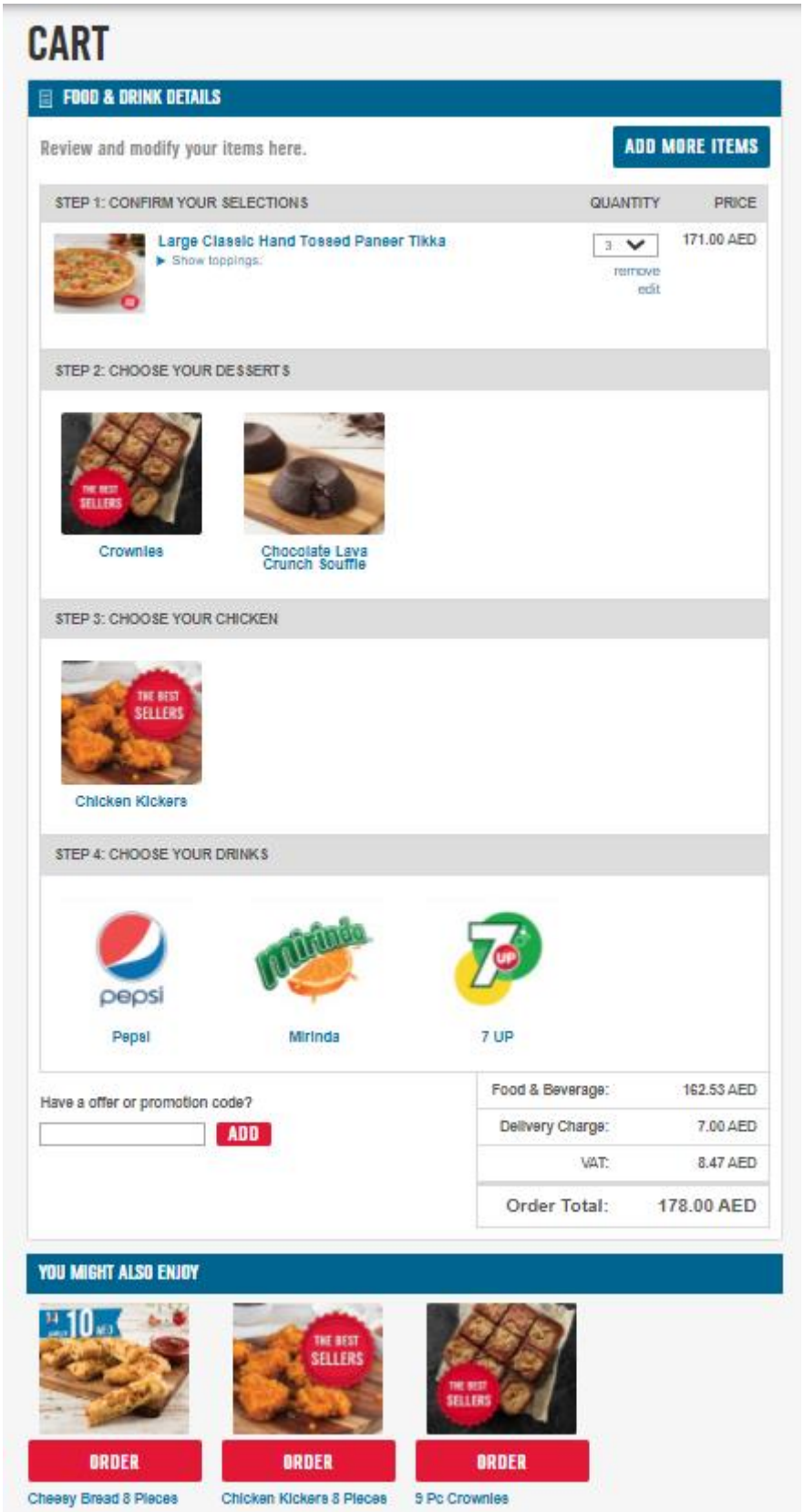


Figure 13: Screenshot of Cart page on the Dominos website.

- **Orders**

Customer Information
Name on Order: Derek Plevy
Delivery Address: 532 E MAIN ST 203, BOUND BROOK, NJ 08805-2196
Callback Phone #: 943-941-9252
Your Domino's Store (4551): 60c Main St South Bound Brook NJ 08880 | 732-563-0330
Delivery Time: Approximately 22-32 minutes

Order Details
Order #: 291935
Date: 06/12/2016 12:57AM

The following order is being delivered hot and fresh to your door:

Quantity	Description	Amount
	X-Large (16") Hand Tossed Philly Cheese Steak	
1	**Whole:** Onions, Green Peppers, Mushrooms, Philly Steak, Shredded Provolone Cheese, American Cheese	$18.99
	X-Large (16") Hand Tossed Ultimate Pepperoni Feast™	
1	**Whole:** Extra Cheese, Robust Inspired Tomato Sauce, Extra Pepperoni, Shredded Parmesan, Shredded Provolone Cheese	$20.98

Food & Bev Total:	$39.97
Tax:	$2.97
Bottle Deposit:	$0.00
Delivery Charge:	$2.50
Total:	**$45.44**

Payment Details
Payment Method: Pay with Cash upon Delivery $45.44
Any Delivery Charge is not a tip paid to your driver.

Figure 14: Screenshot of the “Order summary” section on the Dominos website.

- **Payment Page**

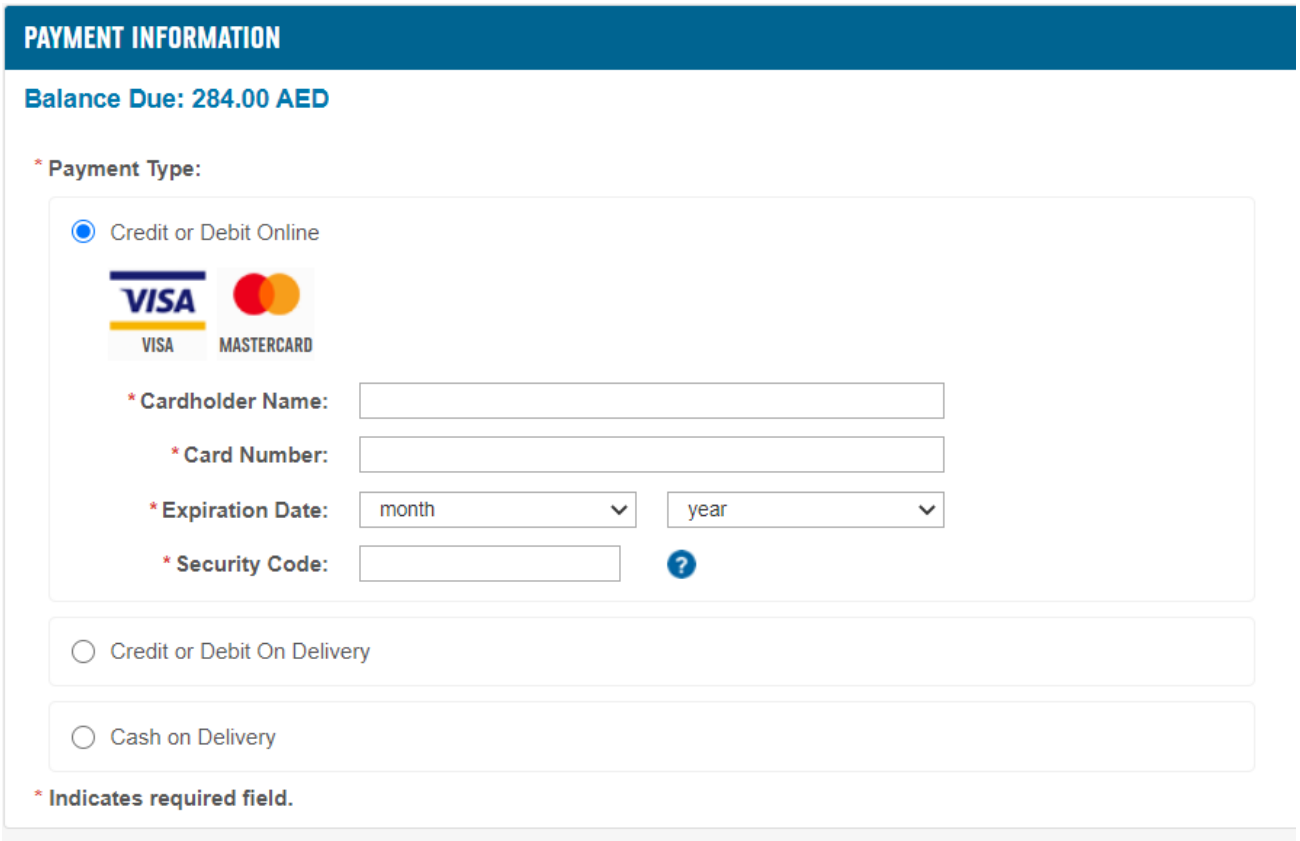


Figure 15: Screenshot of the Payment page on the dominos website.

Table 2, below, contains data-fields extracted during the analysis of the dominos.ae website.

CUSTOMER	Description	Recommendations
First_Name	Size	**ORDERS**
Last_Name	Crust	Order_number
Email	Cheese	Order_date
Phone_number	Sauce	Restaurant_address
Password	Toppings	Delivery_time
Alternate_number	**NON_PIZZA**	Quantity
Birthday	Name	Description
Address	Description	Order_total
ADDRESS	Price	Tax
Address_Type	Menu	Deposit
Building_number	**CART**	Delivery_charge
Apartment_name	Food_name	Total
Area	Food_quantity	Payment_info
Floor_number	Choose_dessert	**PAYMENT**
Apartment_number	Choose_chicken	Payment_Type
Delivery_instructions	Choose_drinks	Cardholder_name
MENUS	Offer_code	Card_number
Name	Food_price	Expiration_date
Description	Delievry_charge	Security_code
PIZZA	VAT	
Pizza_name	Order_total	

1.2.3 List of data-fields from dominos.ae website.

- **My Profile Page**

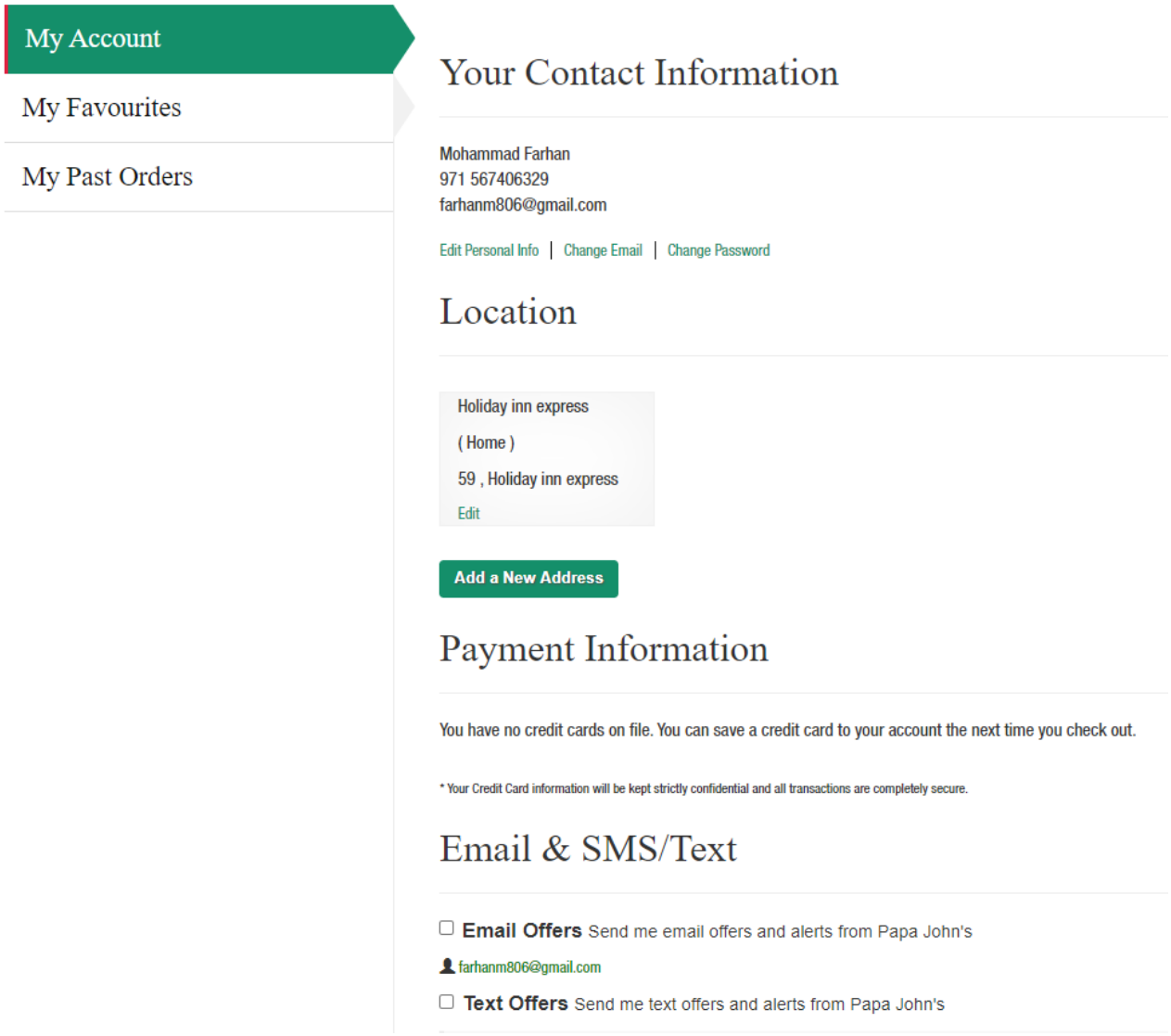


Figure 16: Screenshot of the “My account” page on the Papa Johns’ website.

- **Address Page**

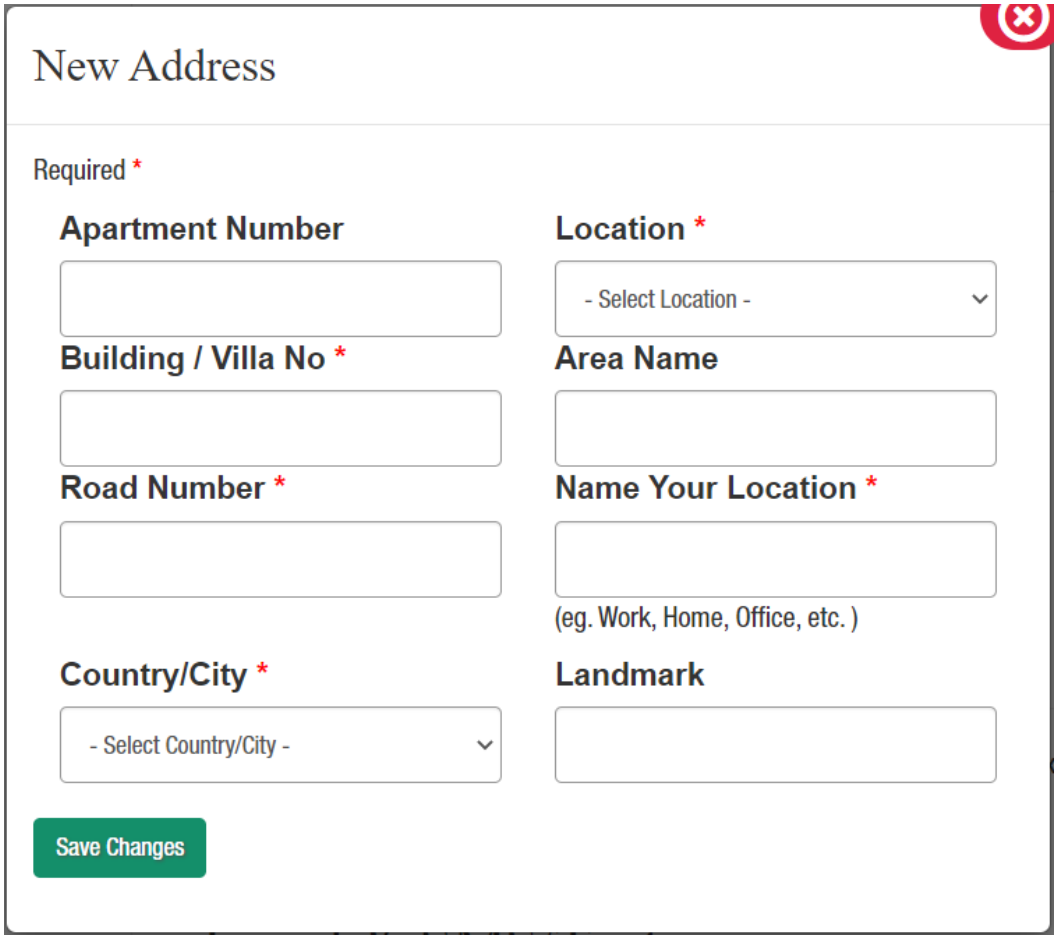


Figure 17: Screenshot of the “Add address” page on the Papa Johns’ website.

- **Menus**

Figure 8: Screenshot of the Menus on the Papa Johns' website.

- **Pizza Menu**

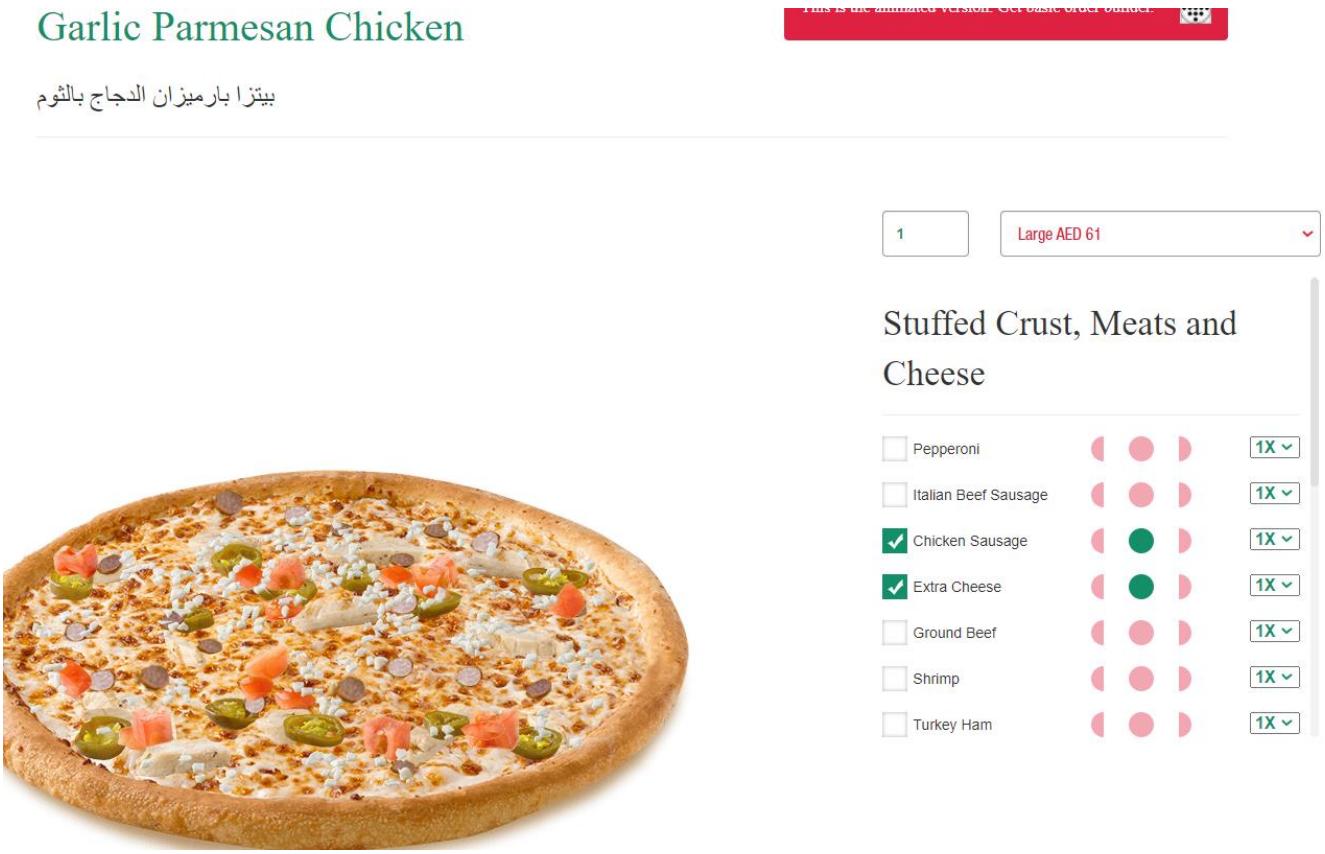


Figure 18: Screenshot of the Pizza Menus on the Papa Johns' website.

- **Non-Pizza Menu**

Figure 19: Screenshot of the “Non_pizza” menus on the Papa Johns’ website.

- **Your Cart**

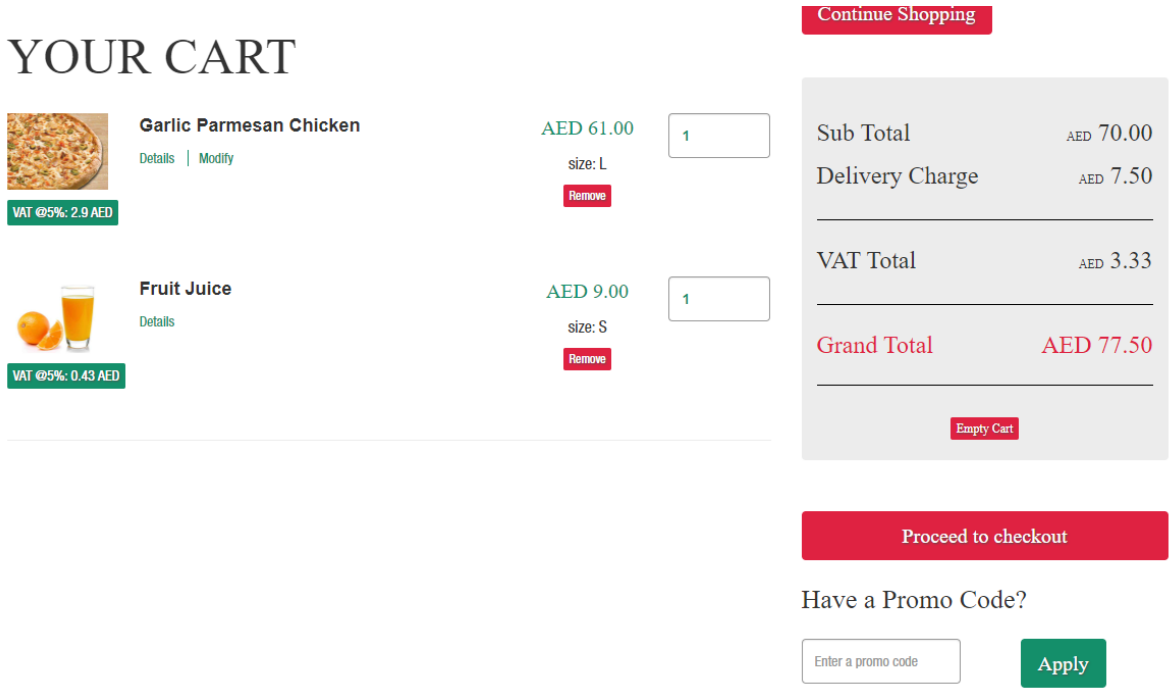
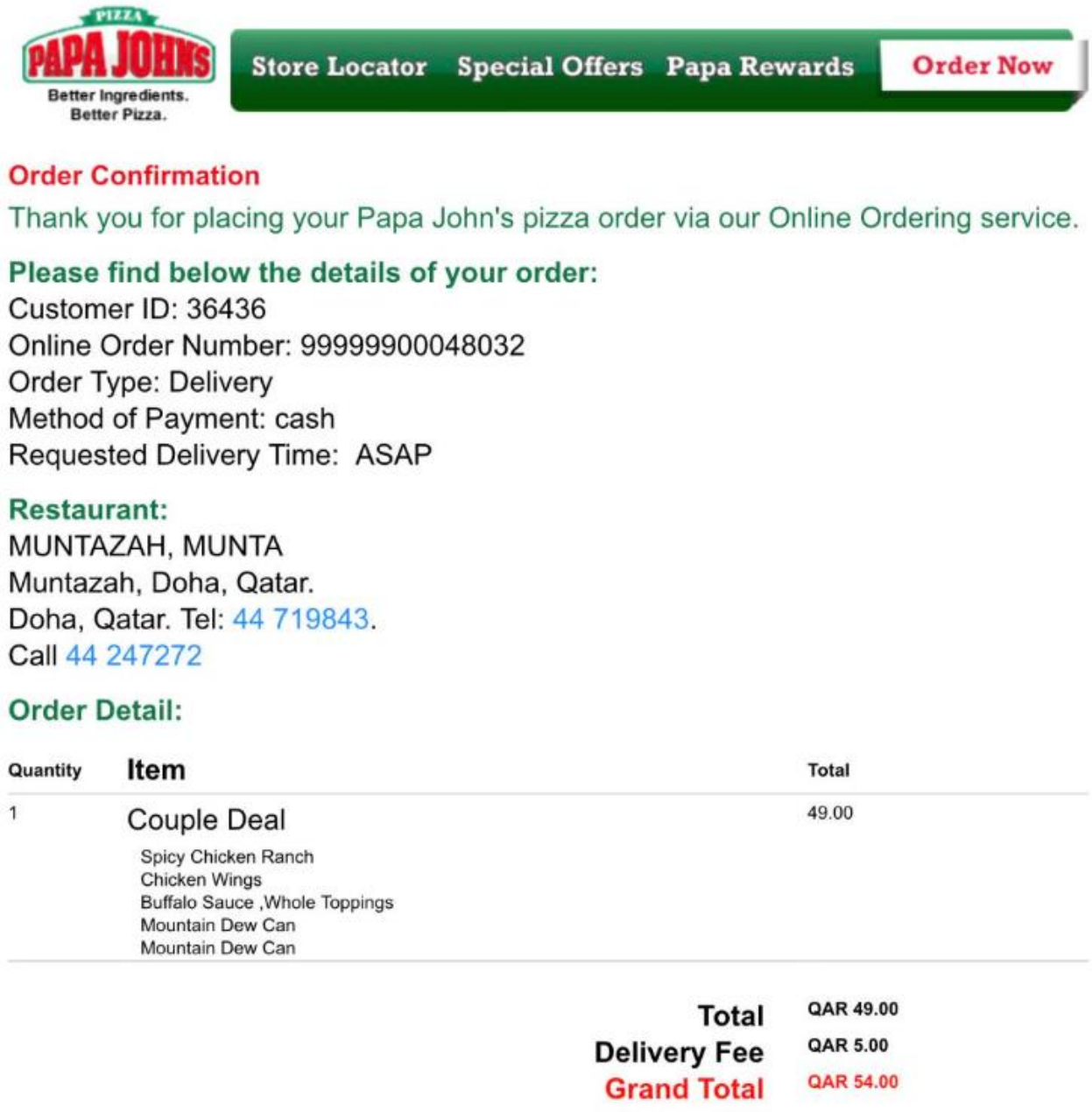


Figure 20: Screenshot of Cart page on the Papa Johns' website.

- **Orders**



PAPA JOHNS
Better Ingredients. Better Pizza.

Store Locator Special Offers Papa Rewards Order Now

Order Confirmation

Thank you for placing your Papa John's pizza order via our Online Ordering service.

Please find below the details of your order:

Customer ID: 36436
Online Order Number: 99999900048032
Order Type: Delivery
Method of Payment: cash
Requested Delivery Time: ASAP

Restaurant:

MUNTAZAH, MUNTA
Muntazah, Doha, Qatar.
Doha, Qatar. Tel: 44 719843.
Call 44 247272

Order Detail:

Quantity	Item	Total
1	Couple Deal Spicy Chicken Ranch Chicken Wings Buffalo Sauce ,Whole Toppings Mountain Dew Can Mountain Dew Can	49.00

Total	QAR 49.00
Delivery Fee	QAR 5.00
Grand Total	QAR 54.00

Figure 21: Screenshot of the “Order Summary” section on the Papa Johns’ website.

- **Payment Page**

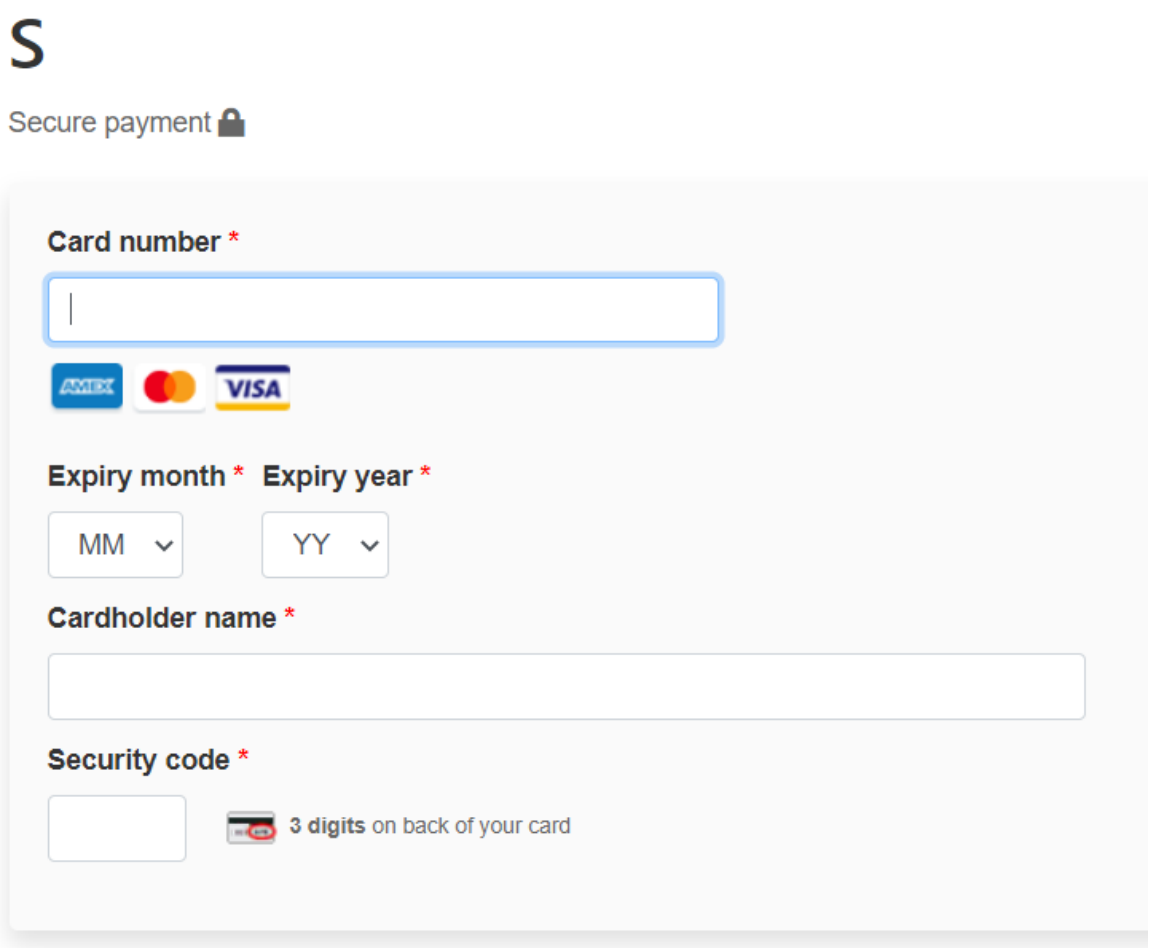


Figure 22: Screenshot of the “Payment” section on the Papa Johns’ website.

Table 3, below, contains data-fields extracted during the analysis of the papajohns.ae website.

CUSTOMER	Description	VAT_total
First_name	**PIZZA**	Grand_total
Last_name	Name	Promo_code
Email	Quantity	**ORDERS**
Mobile_number	Size	Customer_ID
Phone	Price	Order_number
Password	Description	Order_type
Email_deals	Toppings	Payment_method
Date_of_birth	**NON_PIZZA**	Delivery_time
Address	Name	Restaurant_address
Text_deals	Quantity	Order_items
ADDRESS	Size	Quantity
Apartment_number	Price	Total
Location	Description	Delivery_fee
Building_number	Menu	Grand_total
Area_name	**CART**	**PAYMENT**
Road_number	Food_name	Card_number
Name_Location	Food_price	Expiry_month
City	Food_quantity	Expiry_year
Landmark	Size	Cardholder_name
MENUS	Sub_total	Securiy_code
Name	Delivery_charge	

1.3 Finalised List

Table 4, below, contains the final list of data-fields chosen after the analysis of the three websites chosen.

CUSTOMER	Date_added	Grand_total
Customer_ID	**PIZZA**	**PAYMENT**
First_name	Pizza_ID	Payment_ID
Last_name	Name	Payment_reference
Email	Price	Amount
Password	**NON_PIZZA**	Payment_date
Date_of_birth	Non_Pizza_ID	**INGREDIENTS**
Address	Name	Ingredient_ID
Email_offers	Price	Name
ADDRESS	Menu	Price
Address_ID	**CART**	**STAFF**
City	Cart_ID	Staff_ID
Area	Promo_code	Fist_name
Building_number	Recommendations	Last_name
Floor_number	Quantity	Email
Apartment_number	**ORDERS**	Password
Address_type	Order_ID	
MENUS	Order_date	
Menu_ID	Restaurant_address	
Name	Order_items	
Description	Order_total	

2. Database design

2.1 Extended Entity Relationship Model

Figure 23, on the next page represents the Extended Entity Relationship Model, which is base for a normalised model shown in figure 24.

As shown in the model shown in figure 23:

- Customer can have “0 to many” addresses
- An address can belong to only one customer
- Customer can have “0 to many” payments made
- A payment can only be unique to the specific customer
- Customer can have “0 to many” orders
- Each order can be placed by one customer only
- A staff can register “0 to many” orders
- Each order can only be registered by one staff only
- Store can have one or many staff members
- Each staff member can only be working in one store
- Orders can have one or more products
- A product can belong to “0 to many” orders
- A store can have 1 or many menus
- A menu can belong to only one store
- A menu can have 1 or many products
- A product can belong to 1 or many menus
- A Pizza can have 1 or many ingredients
- An ingredient can belong to 1 to many pizzas
- Products is a parent table and has the child tables; pizza and non_pizza.

Extended relationships between entities have been added. Aggregation is when if one entity instance is deleted the related instances will exist. Composition is when destroying one entity will destroy the other entity as well.

Extended Entity Relationship Model

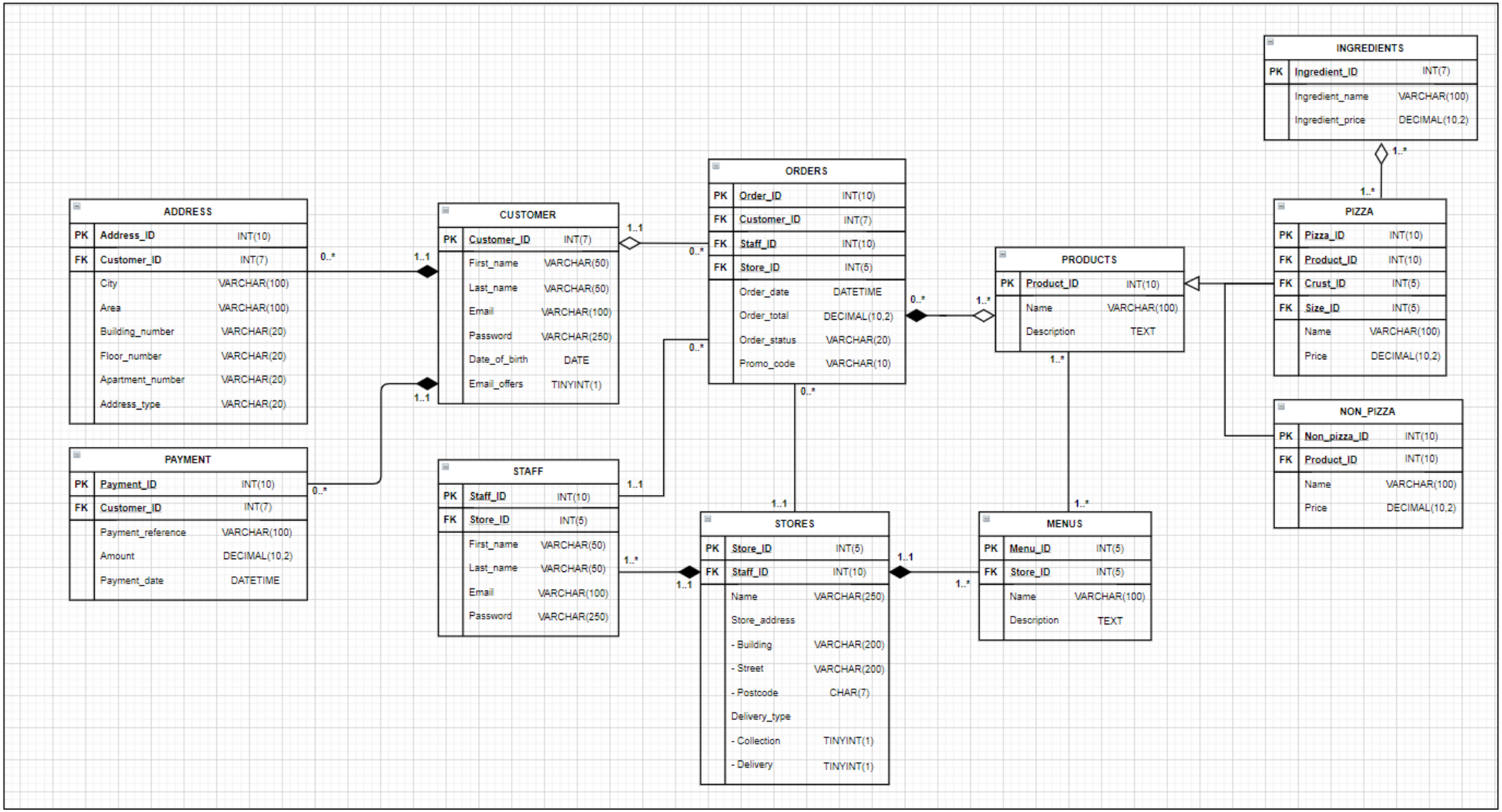


Figure 23: Extended Entity Relationship Model

2.2 Normalised Model

The EERD in figure 23 has been normalised using normalisation techniques. Normalisation involves examining the relationships between attributes and ensure that there is no data redundancy to improve consistency.

The EERD in figure 23 is normalised to 3NF (Third normal form).

1NF is when an entity has an identifying key and there is no repeating attributes. The EERD has unique identifying keys but has repeating attributes in store, products, pizza, crust_ID, size_ID and ingredients tables. The EERD fails 1NF. To pass 1NF separate tables have been made for all attrubutes. Hence the EERD passes 1NF.

The Staff_ID has been removed from the stores table.

The initial model did not contain any composite primary keys, therefore it passes the second normal form (2NF).

The 2NF EERD passes 3NF as it is in 2NF and no non-key attribute depends on any other non-key attribute.

- Pizza_ingredients table added to store ingredients used in each pizza.
- Crust and size tables added to store the attributes of the pizza.
- Pizza_specifications added to store the attributes of each pizza depending on the customer’s choice
- Missing attributes are added to the tables.
- Order_items to store details of the products ordered.
- Data types have been checked

Normalised Extended Entity Relationship Model

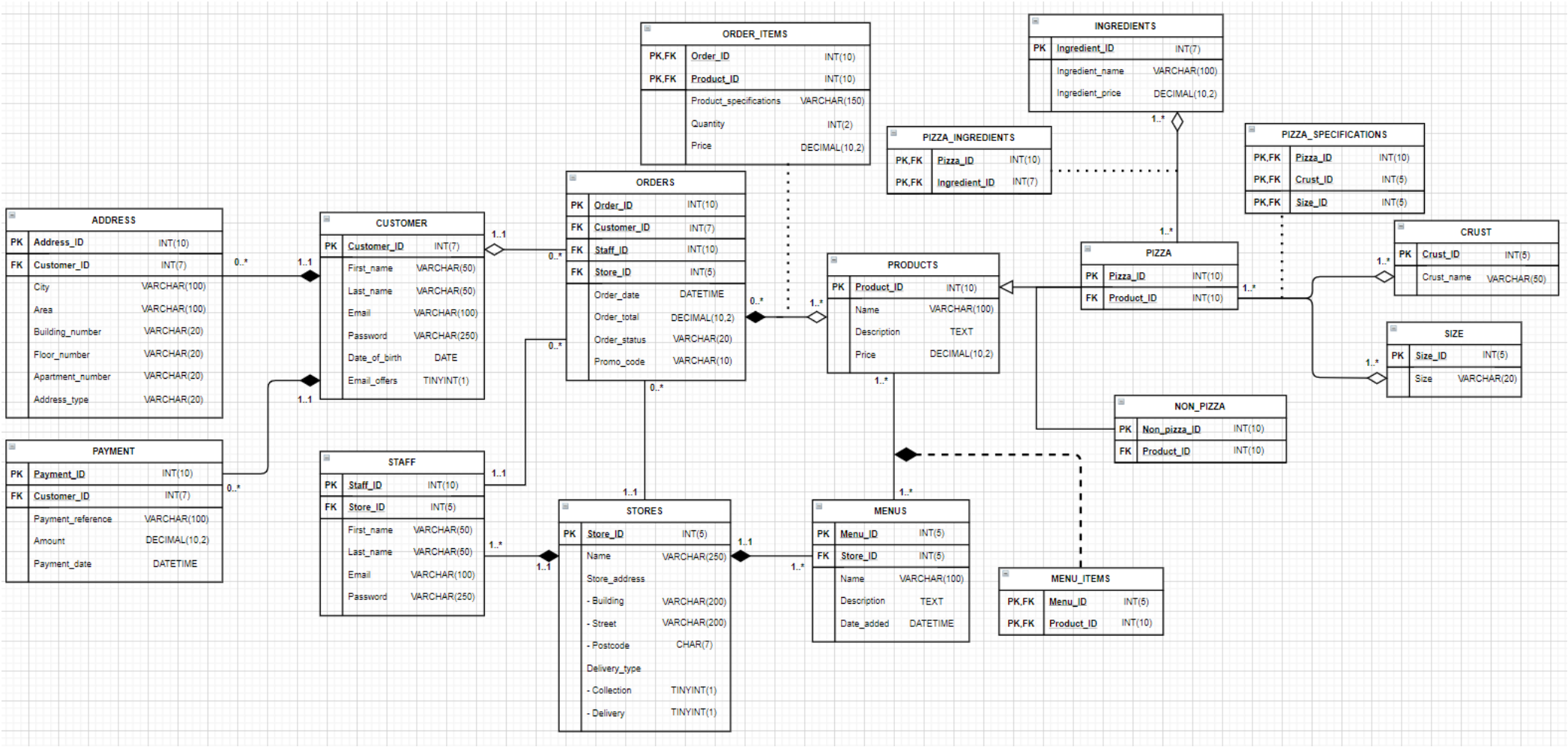


Figure 24: Normalised Extended Entity Relationship Model

2.3 Database Schema

Customer		
Attribute Name	**Type**	**Description**
Customer_ID	Int(7)	Primary Key, indexed, unique. 107 billion customers
First_name	Varchar(50)	Holds customer's first name
Last_name	Varchar(50)	Holds customer's last name
Email	Varchar(100)	Holds customer's email
Password	Varchar(250)	Holds customer's password. Encrypted form
Date_of_birth	Date	Holds customer's date of birth
Email_offers	Tinyint(1)	Holding customer's choice on receiving email offers

Figure 25: Description of attributes stored in the customer table.

Address		
Attribute Name	**Type**	**Description**
Address_ID	Int(10)	Primary key, 107 billion possible addresses
Customer_ID	Int(7)	Holds customer. Indexed. Referenced from customer table
City	Varchar(100)	Holds customer's city
Area	Varchar(100)	Holds customer's area
Building_number	Varchar(20)	Holds customer's building number
Floor_number	Varchar(20)	Holds customer's floor number
Apartment_number	Varchar(20)	Holds customer's apartment number
Address_type	Varchar(20)	Holds address per address_ID; "Home", "Business"

Figure 26: Description of attributes stored in the address table.

Store		
Attribute Name	**Type**	**Description**
Store_ID	Int(5)	Primary key, Unique
Name	Varchar(250)	Holds store's name
Building	Varchar(200)	Holds store's building
Street	Varchar(200)	Holds store's street
Postcode	Char(7)	Holds store's postcode
Delivery	Tinyint(1)	Holding store's choice of offering delivery option
Collection	Tinyint(1)	Holding store's choice of offering collection option

Figure 27: Description of attributes stored in the store table.

Staff		
Attribute Name	**Type**	**Description**
Staff_ID	Int(10)	Primary key, unique
Store_ID	Int(5)	Holds stores, Indexed, Referenced from stores table
First_name	Varchar(50)	Holds staff's first name
Last_name	Varchar(50)	Holds staff's last name
Email	Varchar(100)	Holds staff's email
Password	Varchar(250)	Holds staff's password. Encrypted form

Figure 28: Description of attributes stored in the staff table.

Menus		
Attribute Name	**Type**	**Description**
Menu_ID	Int(5)	Primary key, unique
Store_ID	Int(5)	Holds stores', Referenced from stores table
Name	Varchar(100)	Holds menu's name
Description	Text	Holds menu's description
Date_added	Datetime	Holds the date the menu was added

Figure 29: Description of attributes stored in the menus table.

Menu_items		
Attribute Name	**Type**	**Description**
Menu_ID	Int(5)	Holds menu, composite key, referenced from menu table
Product_ID	Int(5)	Holds product, composite key, referenced from product table

Figure 28: Description of attributes stored in the menu_items table.

Products		
Attribute Name	**Type**	**Description**
Product_ID	Int(10)	Primary key, unique
Name	Varchar(100)	Holds products' name
Description	Text	Holds products' description
Price	Decimal(10,2)	Holds products' price

Figure 28: Description of attributes stored in the product table.

Pizza		
Attribute Name	**Type**	**Description**
Pizza_ID	Int(10)	Primary key, unique
Product_ID	Int(10)	Holds products, referenced from products table

Figure 29: Description of attributes stored in the pizza table.

Pizza_Specifications		
Attribute Name	**Type**	**Description**
Pizza_ID	Int(10)	Holds pizza, composite key, referenced from pizza table
Crust_ID	Int(5)	Holds crust, composite key, referenced from crust table
Size_ID	Int(5)	Holds size, composite key, referenced from size table

Figure 30: Description of attributes stored in the pizza_specifications table.

Crust		
Attribute Name	**Type**	**Description**
Crust_ID	Int(5)	Primary key, unique
Crust_name	Varchar(50)	Holds crusts' name

Figure 31: Description of attributes stored in the crust table.

Size		
Attribute Name	**Type**	**Description**
Size_ID	Int(5)	Primary key, unique
Size	Enum('small', 'medium', 'large')	Holds sizes available as a drop down. Pre-defined

Figure 32: Description of attributes stored in the size table.

Ingredients		
Attribute Name	**Type**	**Description**
Ingredient_ID	Int(7)	Primary key, unique
Ingredient_name	Varchar(100)	Holds ingredient's name
Ingredient_price	Decimal(10,2)	Holds ingredient's price

Figure 33: Description of attributes stored in the ingredients table.

Pizza_ingredients		
Attribute Name	**Type**	**Description**
Pizza_ID	Int(10)	Holds pizza, composite key, referenced from pizza table
Ingredient_ID	Int(7)	Holds ingredients, composite key, referenced from ingredients table

Figure 34: Description of attributes stored in the Pizza_ingredients table.

Non_Pizza		
Attribute Name	**Type**	**Description**
Non_pizza_ID	Int(10)	Primary key, unique
Product_ID	Int(10)	Holds product, referenced from product table

Figure 35: Description of attributes stored in the Non_pizza table.

Orders		
Attribute Name	**Type**	**Description**
Order_ID	Int(10)	Primary key, unique
Customer_ID	Int(7)	Holds customer, referenced from customer table
Staff_ID	Int(10)	Holds staff, referenced from staff table
Store_ID	Int(5)	Holds store, referenced from store table
Order_date	Datetime	Holds the date the order was made
Delivery_fee	Decimal(5,2)	Holds delivery fee of an order
Order_total	Decimal(10,2)	Holds total price of an order
Order_status	Enum(	Holds order status
Promo_code	Varchar(10)	Holds promo codes

Figure 36: Description of attributes stored in the order table.

Order_items		
Attribute Name	**Type**	**Description**
Order_ID	Int(10)	Holds orders, composite key, referenced from orders table
Product_ID	Int(10)	Holds product, composite key, referenced from products table
Quantity	Int(2)	Holds quantity of the products being ordered
Price	Decimal(10,2)	Holds price

Figure 37: Description of attributes stored in the order_items table.

Payment		
Attribute Name	**Type**	**Description**
Payment_ID	Int(10)	Primary key, unique
Customer_ID	Int(7)	Holds customer, referenced from customer table
Payment_reference	Varchar(100)	Holds payment reference for a payment
Amount	Decimal(10,2)	Holds amount to be paid
Payment_date	Datetime	Holds date the payment was made

Figure 38: Description of attributes stored in payment table.

3. Database implementation

The database for the proposed system is built using phpMyAdmin, version 5.0.3. The database server provided is MariaDB, version 10.4.14.

Figure 39 shows the list of tables that use the storage engine (InnoDB). The number of rows represent the number of records (test data) that was entered. Some records were entered manually, whereas others (i.e rows in the customer, address, payment and staff tables) have been auto generated using filldb.info website (FillDB, 2016).

Table	Action	Rows	Type	Collation	Size	Overhead
address	Browse Structure Search Insert Empty Drop	20	InnoDB	utf8mb4_general_ci	32.0 KiB	-
crust	Browse Structure Search Insert Empty Drop	6	InnoDB	utf8mb4_general_ci	16.0 KiB	-
customer	Browse Structure Search Insert Empty Drop	10	InnoDB	utf8mb4_general_ci	16.0 KiB	-
ingredients	Browse Structure Search Insert Empty Drop	12	InnoDB	utf8mb4_general_ci	16.0 KiB	-
menus	Browse Structure Search Insert Empty Drop	20	InnoDB	utf8mb4_general_ci	48.0 KiB	-
menu_items	Browse Structure Search Insert Empty Drop	20	InnoDB	utf8mb4_general_ci	32.0 KiB	-
non_pizza	Browse Structure Search Insert Empty Drop	21	InnoDB	utf8mb4_general_ci	32.0 KiB	-
orders	Browse Structure Search Insert Empty Drop	8	InnoDB	utf8mb4_general_ci	64.0 KiB	-
order_items	Browse Structure Search Insert Empty Drop	12	InnoDB	utf8mb4_general_ci	48.0 KiB	-
payment	Browse Structure Search Insert Empty Drop	10	InnoDB	utf8mb4_general_ci	32.0 KiB	-
pizza	Browse Structure Search Insert Empty Drop	6	InnoDB	utf8mb4_general_ci	32.0 KiB	-
pizza_ingredients	Browse Structure Search Insert Empty Drop	12	InnoDB	utf8mb4_general_ci	32.0 KiB	-
pizza_specifications	Browse Structure Search Insert Empty Drop	7	InnoDB	utf8mb4_general_ci	64.0 KiB	-
products	Browse Structure Search Insert Empty Drop	26	InnoDB	utf8mb4_general_ci	16.0 KiB	-
size	Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_general_ci	16.0 KiB	-
staff	Browse Structure Search Insert Empty Drop	20	InnoDB	utf8mb4_general_ci	32.0 KiB	-
store	Browse Structure Search Insert Empty Drop	10	InnoDB	utf8mb4_general_ci	16.0 KiB	-
17 tables	Sum	223	InnoDB	utf8mb4_general_ci	544.0 KiB	0 B

Figure 40: Details of tables in the proposed database.

The SQL statement generated upon creating and inserting data into the tables is displayed on the next page. The details of indexes, constraints and auto increments for respective tables have also been shown in pages 49, 50 and 51 resepctively.

```
--
-- Table structure for table `customer`
--

CREATE TABLE `customer` (
  `customer_id` int(10) NOT NULL,
  `first_name` varchar(50) NOT NULL,
  `last_name` varchar(50) NOT NULL,
  `email` varchar(100) NOT NULL,
  `password` varchar(250) NOT NULL,
  `date_of_birth` date NOT NULL,
  `email_offers` tinyint(1) NOT NULL COMMENT 'If the customer chooses to receive offers by email'
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='table to store customer''s data';

--
-- Dumping data for table `customer`
--

INSERT INTO `customer` (`customer_id`, `first_name`, `last_name`, `email`, `password`, `date_of_birth`, `email_offers`) VALUES
(1, 'Tessie', 'Gusikowski', 'dayne04@example.net', '08e492bbf62800a1bd1b42cbd57c9441', '2010-08-30', 0),
(2, 'Romaine', 'Swaniawski', 'chris46@example.com', '3b38b2c82c3c8214ab15aac2086e69af', '2007-05-03', 1),
(3, 'Darrel', 'Braun', 'bergnaum.providenci@example.com', 'a46689264b20005a7796eb21771685ff', '1991-08-10', 1),
(4, 'Stanley', 'Dickens', 'thea.little@example.org', 'd1c6b045bc9d5b3cae9218737da85078', '2000-03-26', 0),
(5, 'Bonita', 'O\'Keefe', 'rempel.theresia@example.com', '263bb9b207368b3f5925e59f98bf30b4', '1989-03-15', 1),
(6, 'Devan', 'Luettgen', 'arvel26@example.org', 'bc1380211cf81bfab58d03eb5327132d', '1976-10-05', 0),
(7, 'Francisca', 'Upton', 'lkihn@example.net', '3e159cbf13fdb5709baeef594ae9ae7e', '1977-03-06', 0),
(8, 'Amya', 'Renner', 'lindsay.block@example.net', '9e5cefc6e661f8bd6fcc9c3f0ce595c3', '2014-01-01', 0),
(9, 'Fritz', 'Adams', 'oritchie@example.net', '4ef1e29347387123e73bf53680ee1e3f', '2010-03-08', 0),
(10, 'Kylee', 'Haley', 'gerardo.mraz@example.net', '711f500487b376e8fe1178df53c12ade', '2002-03-12', 1);
```

Figure 40: Details of implementation of customer table

```
--
-- Table structure for table `address`
--

CREATE TABLE `address` (
  `address_id` int(10) NOT NULL,
  `customer_id` int(7) NOT NULL,
  `city` varchar(100) NOT NULL,
  `area` varchar(100) NOT NULL,
  `building_number` varchar(20) NOT NULL,
  `floor_number` varchar(20) NOT NULL,
  `apartment_number` varchar(20) NOT NULL,
  `address_type` varchar(20) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='table to store addresses';

--
-- Dumping data for table `address`
--

INSERT INTO `address` (`address_id`, `customer_id`, `city`, `area`, `building_number`, `floor_number`, `apartment_number`, `address_type`) VALUES
(1, 1, 'Kimberlymouth', 'Erdman Lodge', '38793', '13', 'Apt. 510', 'Business'),
(2, 2, 'Swaniawskifurt', 'Sauer Port', '403', '5', 'Suite 705', 'Home'),
(3, 3, 'West Verdiemouth', 'Kitty Wells', '90620', '24', 'Apt. 045', 'Work'),
(4, 4, 'West Dorris', 'Altenwerth Prairie', '6093', '13', 'Apt. 319', 'Work'),
(5, 5, 'Kiehnmouth', 'Lyric Creek', '39594', '11', 'Suite 449', 'Home'),
(6, 6, 'Gutmannville', 'Gleichner Ferry', '3163', '23', 'Suite 889', 'Business'),
(7, 7, 'Ebertborough', 'Americo Ports', '8110', '19', 'Suite 857', 'Home'),
(8, 8, 'Pagacchester', 'Rosamond Brook', '8696', '15', 'Suite 013', 'Home'),
(9, 9, 'Lake Giovannyshire', 'Hermina Haven', '96019', '19', 'Apt. 274', 'Work'),
(10, 10, 'Abu Dhabi', 'Electra Street', '473', '28', 'Suite 64-B', 'Home'),
(11, 1, 'Downtown Dubai', 'Burj al arab Street', '643', '23', 'Apt. 32-F', 'Work'),
(12, 3, 'Kimberlymouth', 'Erdman Lodge', '3893', '13', 'Apt. 510', 'Home'),
(13, 3, 'Swanskifurt', 'Sauer Pot', '403', '5', 'Suite 75', 'Home'),
(14, 4, 'Verdiemouth', 'Kitty Wells', '920', '24', 'Apt. 05', 'Work'),
(15, 5, 'Dorris', 'Altenwerth Prairie', '6093', '13', 'Apt. 19', 'Work'),
(16, 5, 'Kiemouth', 'Lyric Creek', '3594', '11', 'Suite 49', 'Business'),
(17, 4, 'Gutville', 'Gleichner Ferry', '3163', '23', 'Suite 89', 'Business'),
(18, 8, 'Eberough', 'Americo Ports', '810', '19', 'Suite 57', 'Business'),
(19, 9, 'Pagester', 'Rosamond Brook', '8696', '15', 'Suite 03', 'Work'),
(20, 10, 'Giovannyshire', 'Hermina Haven', '9619', '19', 'Apt. 24', 'Home');
```

Figure 41: Details of implementation of address table

```
--
-- Table structure for table `store`
--

CREATE TABLE `store` (
  `store_id` int(5) NOT NULL,
  `name` varchar(250) NOT NULL,
  `building` varchar(200) NOT NULL,
  `street` varchar(200) NOT NULL,
  `city` varchar(100) NOT NULL,
  `postcode` char(7) NOT NULL,
  `delivery` tinyint(1) NOT NULL,
  `collection` tinyint(1) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='table for store';

--
-- Dumping data for table `store`
--

INSERT INTO `store` (`store_id`, `name`, `building`, `street`, `city`, `postcode`, `delivery`, `collection`) VALUES
(1, 'Pizza Hut - Fermville', '77521', '27281 Spinka Villages', 'Ferminville', '739', 1, 1),
(2, 'Pizza Hut - New soledad', '62651', '633 Jane Crest', 'New Soledad', '443', 1, 0),
(3, 'Pizza Hut - South', '42526', '969 Von Highway Apt. 958', 'South Wymanview', '615', 0, 1),
(4, 'Pizza Hut - West', '7782', '9007 Nolan Oval Suite 953', 'West Cordelia', '229', 0, 0),
(5, 'Pizza Hut - Jasper', '14165', '49603 Lamont Court Apt. 976', 'Jasperfort', '173', 0, 1),
(6, 'Pizza Hut - Lionel', '09360', '16679 Conor Mount Suite 256', 'Lionelfort', '033', 1, 0),
(7, 'Pizza Hut - Rowland', '95851', '5654 Wehner Neck Apt. 607', 'Rowlandchester', '453', 0, 0),
(8, 'Pizza Hut - Lockman', '376', '13422 Malcolm Lock', 'Lockmanview', '835', 0, 1),
(9, 'Pizza Hut - Harmon', '7814', '8276 Lubowitz Corner Apt. 422', 'Harmonside', '081', 0, 1),
(10, 'Pizza Hut - Lake', '0965', '416 Lind Falls', 'Lake Alisachester', '421', 1, 1);
```

Figure 42: Details of implementation of store table

```
--
-- Table structure for table `staff`
--

CREATE TABLE `staff` (
  `staff_id` int(10) NOT NULL,
  `store_id` int(5) NOT NULL,
  `first_name` varchar(50) NOT NULL,
  `last_name` varchar(50) NOT NULL,
  `email` varchar(100) NOT NULL,
  `password` varchar(250) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='table for store staff ';

--
-- Dumping data for table `staff`
--

INSERT INTO `staff` (`staff_id`, `store_id`, `first_name`, `last_name`, `email`, `password`) VALUES
(1, 1, 'Camila', 'Reilly', 'lehner.roma@example.com', '2bdb95b4152267974b26cb8383f9c546'),
(2, 1, 'Paris', 'Brekke', 'yswift@example.org', 'ce3bd90c98f493438dea7c08fabaee5c'),
(3, 2, 'Rosalinda', 'Buckridge', 'brisa.kemmer@example.org', 'ee91e105ae8c0aa1bef9ba532f3a8a7f'),
(4, 2, 'Viola', 'Heaney', 'taufderhar@example.com', 'ca3e4f82be871dea516542a5f5148f28'),
(5, 3, 'Johan', 'Gibson', 'haskell.rohan@example.net', 'c1ca36573060cd34ce42bff5c0112967'),
(6, 3, 'Annie', 'Padberg', 'logan80@example.com', 'c3f2e523f6af6a2800e05c71ee29b2f8'),
(7, 4, 'Odell', 'Fahey', 'egrant@example.com', '47d5331d4ec94740014b28cb84f32284'),
(8, 4, 'Russell', 'Doyle', 'metz.judge@example.com', '7f82d31b677a8dbe8f7eb9e62c497964'),
(9, 5, 'Crystel', 'Cormier', 'sibyl.hickle@example.org', '9796e49d32aaa233dd608c9112778e06'),
(10, 5, 'Clarissa', 'Tillman', 'koch.darian@example.org', '1b29a2a9b31d72293fa0e7f3357c2e0b'),
(11, 6, 'Demetrius', 'Ziemann', 'carmen.schaefer@example.net', 'e58a09d4f39533ae8fd3eb5c2a809d2c'),
(12, 6, 'Sierra', 'Zulauf', 'cronin.esta@example.org', '3092afc2e3ce521144b64c81dc20485d'),
(13, 7, 'Quinn', 'Lockman', 'justice.murray@example.net', '9d4eae8430e82ca2ed594f6c1e0e8371'),
(14, 7, 'Lonie', 'Dare', 'domenica85@example.org', '485214b8833d7dc322caffdf5c5efd9f'),
(15, 8, 'Pansy', 'Sauer', 'rey64@example.org', '73b8c308aca0fd19f00c19e32bc50964'),
(16, 8, 'Jovan', 'Kautzer', 'lauren.dicki@example.com', '96bd6cd07b4082f258b518e2b928f336'),
(17, 9, 'Alex', 'Frami', 'morris32@example.org', '8407f682903a524fee7fc1911140f6d2'),
(18, 9, 'Alessandra', 'Conn', 'pbins@example.com', 'd8d33cd175e83505fb2ea7096fcef879'),
(19, 10, 'Araceli', 'Mertz', 'qmurazik@example.org', '3400730fdb3e285c69d4f20e27a58141'),
(20, 10, 'Missouri', 'Kutch', 'ikautzer@example.com', 'b7bbb78133d74384613bc4d65376206d');
```

Figure 43: Details of implementation of staff table

```
--
-- Table structure for table `menus`
--

CREATE TABLE `menus` (
  `menu_id` int(5) NOT NULL,
  `store_id` int(5) NOT NULL,
  `name` varchar(100) NOT NULL,
  `description` text NOT NULL,
  `date_added` datetime NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `menus`
--

INSERT INTO `menus` (`menu_id`, `store_id`, `name`, `description`, `date_added`) VALUES
(1, 1, 'Starters and pizzas for Ferm store', 'This menu consists of all pizzas - Fermville', '2017-12-09 09:51:01'),
(2, 1, 'Drinks and sides menu for Ferm store', 'This menu consists of all sides - Fermville', '2015-12-02 09:51:01'),
(3, 2, 'Drinks menu for soledad store', 'This menu consists of all the drinks - soledad', '2012-12-07 14:42:37'),
(4, 2, 'Starters and pizzas for new soledad store', 'This is the starters menu for soledad Store', '2020-12-07 03:38:46'),
(5, 3, 'Pizzas menu for south store', 'this is the menu for all pizzas in south store', '2020-12-01 03:38:46'),
(6, 3, 'Drinks menu for south store', 'This menu stores all drinks for south store', '2020-12-10 03:43:19'),
(7, 4, 'Starters menu for west store', 'Stores all starters for the west store', '2020-12-03 03:43:19'),
(8, 4, 'Drinks menu for west', 'This stores all drinks for west store\r\n', '2020-11-10 03:45:27'),
(9, 5, 'Starters menu for jasper store', 'Stores all starters for the jasper store', '2020-12-01 03:47:38'),
(10, 5, 'Drinks menu for jasper store', 'This stores all drinks for jasper store', '2020-11-02 03:47:53'),
(11, 6, 'Starters menu for lionel store', 'Stores all starters for the lionel store', '2020-10-09 03:48:00'),
(12, 6, 'Drinks menu for leionl store', 'This stores all drinks for lionel store', '2020-10-18 03:48:05'),
(13, 7, 'Starters menu for rowland store', 'Stores all starters for the rowland store', '2020-12-05 03:48:16'),
(14, 7, 'Drinks menu for rowland store', 'This stores all drinks for rowland store', '2020-12-03 03:48:22'),
(15, 8, 'Starters menu for lockman store', 'Stores all starters for the lockman store', '2020-11-20 03:48:27'),
(16, 8, 'Drinks menu for lockman store', 'This stores all drinks for lockman store', '2020-11-22 03:48:32'),
(17, 9, 'Starters menu for harmon store', 'Stores all starters for the harmon store', '2020-11-25 03:48:38'),
(18, 9, 'Drinks menu for harmon store', 'This stores all drinks for harmon store', '2020-10-21 03:48:45'),
(19, 10, 'Starters menu for lake store', 'Stores all starters for the lake store', '2020-11-29 03:48:50'),
(20, 10, 'Drinks menu for lake store', 'This stores all drinks for lake store', '2020-10-09 03:48:56');

--
```

Figure 44: Details of implementation of menus table

```
--
-- Table structure for table `menu_items`
--

CREATE TABLE `menu_items` (
  `menu_id` int(5) NOT NULL,
  `product_id` int(10) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='table to store menu items';

--
-- Dumping data for table `menu_items`
--

INSERT INTO `menu_items` (`menu_id`, `product_id`) VALUES
(1, 2),
(2, 22),
(3, 26),
(4, 1),
(5, 4),
(6, 24),
(7, 6),
(8, 20),
(9, 3),
(10, 13),
(11, 2),
(12, 25),
(13, 5),
(14, 22),
(15, 11),
(16, 26),
(17, 12),
(18, 23),
(19, 17),
(20, 16);
```

Figure 45: Details of implementation of menu_items table

```
--
-- Table structure for table `products`
--

CREATE TABLE `products` (
  `product_id` int(10) NOT NULL,
  `name` varchar(100) NOT NULL,
  `description` text NOT NULL,
  `price` decimal(10,2) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='table to store products';

--
-- Dumping data for table `products`
--

INSERT INTO `products` (`product_id`, `name`, `description`, `price`) VALUES
(1, 'Spicy Chicken Ranch', 'Black Olives, Green Peppers, Grilled Chicken, Mozzarella, Mushrooms, Onions', '25.00'),
(2, 'Cheeky Chicken', 'Green Peppers, Grilled Chicken, Mozzarella, Onions, Tomato', '27.00'),
(3, 'Super Supreme', 'Black Olives, Green Peppers, Ground Beef, Mozzarella, Mushrooms, Onions, Pepperoni', '30.00'),
(4, 'Dynamite Shrimp', 'Black Olives, Garlic, Green Peppers, Mozzarella, Shrimp, Tomato', '25.00'),
(5, 'Hot Stuff', 'Green Chilli, Ground Beef, Mozzarella, Onions, Tomato', '22.00'),
(6, 'Fiery chicken pizza', 'Feiry chicken with blazing hot sauce', '35.00'),
(7, 'Salad', 'Various fresh salads and veggies with a wide range of toppings and dressings.', '3.00'),
(8, '6 pc dynamite shrimps', '6 pieces Shrimp covered in Spicy Dynamite sauce topped spring onion.', '5.00'),
(9, 'Garlic bread supreme', 'Pieces of buttery garlic bread with a layer of melted hot cheese', '4.00'),
(10, 'BBQ chicken spin rolls', 'Juicy slices of Italian chicken breast, cheddar cheese and red onions wrapped in tortilla bread and oven-baked. Served with BBQ sauce.', '7.00'),
(11, 'Chicken baked pasta', 'Penne pasta baked with fresh mushrooms, spinach and tender chicken strips topped with melted mozzarella and shredded parmesan cheese.', '8.00'),
(12, 'Foot long lasagna', 'Penne pasta baked with fresh mushrooms, spinach and tender chicken strips topped with melted mozzarella and shredded parmesan cheese.', '12.00'),
(13, 'Veggie baked pasta', 'Penne pasta baked with fresh mushrooms, spinach and tender chicken strips topped with melted mozzarella and shredded parmesan cheese.', '10.00'),
(14, 'Buffalo wings', 'Juicy chicken wings coated and fried to perfection. Available in selected stores only.', '14.00'),
(15, 'Wingstreet - bone in', 'Juicy chicken wings coated and fried to perfection. Available in selected stores only.', '15.00'),
(16, 'Wingstreet - bone out', 'Juicy chicken wings coated and fried to perfection. Available in selected stores only.', '14.00'),
(17, 'Walls Cup Strawberry Vanilla 100ml', 'Walls Cup Cocoa Vanilla, ready to eat single serve ice cream for ice cream fans', '10.00'),
(18, 'Walls Cup Strawberry Chocolate 100ml', 'Walls Cup Cocoa Choco, ready to eat single serve ice cream for ice cream fans', '10.00'),
(19, 'Chocolate cake', 'Cake made with chocolate', '8.00'),
(20, 'Hersheys cookie', '8 slices of ooey gooey goodness, jam packed with 100% genuine Hershey's chocolate chips.', '3.00'),
(21, 'Pepsi', 'Soft-drink', '2.00'),
(22, 'Cola', 'Soft-drink', '2.00'),
(23, 'Orange juice', 'Freshly squeezed orange juice', '4.00'),
(24, 'Sprite', 'Soft-drink', '2.00'),
(25, 'Milkshake', 'Soft-drink', '5.00'),
(26, 'Mirinda', 'Soft-drink', '2.00');

--
```

Figure 46: Details of implementation of products table

```
--
-- Table structure for table `pizza`
--

CREATE TABLE `pizza` (
  `pizza_id` int(10) NOT NULL,
  `product_id` int(10) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='table to store pizza';

--
-- Dumping data for table `pizza`
--

INSERT INTO `pizza` (`pizza_id`, `product_id`) VALUES
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5),
(6, 6);
```

Figure 47: Details of implementation of pizza table

```
--
-- Table structure for table `pizza_specifications`
--

CREATE TABLE `pizza_specifications` (
  `pizza_id` int(10) NOT NULL,
  `crust_id` int(5) NOT NULL,
  `size_id` int(5) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='table to store pizza attributes';

--
-- Dumping data for table `pizza_specifications`
--

INSERT INTO `pizza_specifications` (`pizza_id`, `crust_id`, `size_id`) VALUES
(2, 1, 3),
(2, 3, 3),
(2, 4, 2),
(3, 1, 3),
(3, 2, 2),
(4, 5, 1),
(4, 6, 3);
```

Figure 48: Details of implementation of pizza_specifications table

```
--
-- Table structure for table `crust`
--

CREATE TABLE `crust` (
  `crust_id` int(5) NOT NULL,
  `crust_name` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='table to store pizza crusts''';

--
-- Dumping data for table `crust`
--

INSERT INTO `crust` (`crust_id`, `crust_name`) VALUES
(1, 'Thin \'n crispy'),
(2, 'Pan'),
(3, 'Pizza mia'),
(4, 'Cheese crust'),
(5, 'Stuffed crust'),
(6, 'Cheese stuffed crust');
```

Figure 49: Details of implementation of crust table

```
--
-- Table structure for table `size`
--

CREATE TABLE `size` (
  `size_id` int(5) NOT NULL,
  `size_name` enum('small','medium','large') NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='table to store pizza sizes''';

--
-- Dumping data for table `size`
--

INSERT INTO `size` (`size_id`, `size_name`) VALUES
(1, 'small'),
(2, 'medium'),
(3, 'large');
```

Figure 50: Details of implementation of size table

```
--
-- Table structure for table `ingredients`
--

CREATE TABLE `ingredients` (
  `ingredient_id` int(7) NOT NULL,
  `ingredient_name` varchar(100) NOT NULL,
  `ingredient_price` decimal(10,2) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='table to store ingredients';

--
-- Dumping data for table `ingredients`
--

INSERT INTO `ingredients` (`ingredient_id`, `ingredient_name`, `ingredient_price`) VALUES
(1, 'Cheese', '5.00'),
(2, 'Chicken Breast', '7.00'),
(3, 'Tomatoes', '3.00'),
(4, 'Garlic', '2.00'),
(5, 'Olives', '4.00'),
(6, 'Chilli Sauce', '0.50'),
(7, 'Onions', '1.50'),
(8, 'Green peppers', '0.50'),
(9, 'Pepperoni', '2.00'),
(10, 'Beef', '9.00'),
(11, 'Shrimps', '3.00'),
(12, 'Garlic Sauce', '0.50');
```

Figure 51: Details of implementation of ingredients table

```
--
-- Table structure for table `pizza_ingredients`
--

CREATE TABLE `pizza_ingredients` (
  `pizza_id` int(10) NOT NULL,
  `ingredient_id` int(7) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='table to store pizza ingredients';

--
-- Dumping data for table `pizza_ingredients`
--

INSERT INTO `pizza_ingredients` (`pizza_id`, `ingredient_id`) VALUES
(1, 2),
(1, 3),
(2, 4),
(2, 6),
(3, 7),
(3, 8),
(4, 11),
(4, 12),
(5, 5),
(5, 10),
(6, 1),
(6, 5);
```

Figure 52: Details of implementation of pizza_ingredients table

```
--
-- Table structure for table `non_pizza`
--

CREATE TABLE `non_pizza` (
  `non_pizza_id` int(10) NOT NULL,
  `product_id` int(10) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='table to store non-pizza products';

--
-- Dumping data for table `non_pizza`
--

INSERT INTO `non_pizza` (`non_pizza_id`, `product_id`) VALUES
(1, 6),
(2, 7),
(3, 8),
(4, 9),
(5, 10),
(6, 11),
(7, 12),
(8, 13),
(9, 14),
(10, 15),
(11, 16),
(12, 17),
(13, 18),
(14, 19),
(15, 20),
(16, 21),
(17, 22),
(18, 23),
(19, 24),
(20, 25),
(21, 26);
```

Figure 53: Details of implementation of non_pizza table

```
--
-- Table structure for table `orders`
--

CREATE TABLE `orders` (
  `order_id` int(10) NOT NULL,
  `customer_id` int(7) NOT NULL,
  `staff_id` int(10) NOT NULL,
  `store_id` int(5) NOT NULL,
  `order_date` datetime NOT NULL,
  `delivery_fee` decimal(5,2) DEFAULT NULL,
  `order_total` decimal(10,2) NOT NULL,
  `order_status` enum('received','being prepared','out for delivery','ready for collection','completed') NOT NULL,
  `promo_code` varchar(10) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='table to store orders''';

--
-- Dumping data for table `orders`
--

INSERT INTO `orders` (`order_id`, `customer_id`, `staff_id`, `store_id`, `order_date`, `delivery_fee`, `order_total`, `order_status`, `promo_code`) VALUES
(1, 1, 1, 1, '2020-12-09 21:50:23', '5.00', '30.00', 'out for delivery', 'PZ50'),
(2, 2, 1, 1, '2019-12-09 23:49:15', NULL, '27.00', 'ready for collection', NULL),
(3, 3, 1, 1, '2017-12-09 23:02:31', '6.50', '40.00', 'out for delivery', NULL),
(4, 1, 3, 2, '2020-12-10 07:58:20', '5.00', '45.00', 'completed', NULL),
(5, 4, 6, 3, '2018-01-09 02:59:39', NULL, '36.00', 'being prepared', 'RE20'),
(6, 8, 1, 1, '2017-12-09 17:12:47', '4.50', '52.00', 'completed', NULL),
(7, 2, 13, 7, '2020-12-07 17:18:35', NULL, '33.00', 'ready for collection', '45WE'),
(8, 3, 13, 7, '2020-12-09 23:03:04', '4.50', '47.50', 'ready for collection', 'SD35');
```

Figure 54: Details of implementation of orders table

```
--
-- Table structure for table `order_items`
--

CREATE TABLE `order_items` (
  `order_id` int(10) NOT NULL,
  `product_id` int(10) NOT NULL,
  `product_specification` varchar(150) DEFAULT NULL COMMENT 'Product Specifciations chosen by the customers are shown here',
  `quantity` int(2) NOT NULL,
  `price` decimal(10,2) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='table to store order items''';

--
-- Dumping data for table `order_items`
--

INSERT INTO `order_items` (`order_id`, `product_id`, `product_specification`, `quantity`, `price`) VALUES
(1, 2, 'Thin\'n crispy crust and large size', 1, '42.00'),
(1, 9, 'No cheese', 2, '10.00'),
(1, 21, 'Medium Pepsi', 1, '4.00'),
(2, 3, 'Pan crust and medium sized', 2, '40.00'),
(2, 7, NULL, 4, '16.00'),
(2, 22, 'Large Cola', 1, '3.00'),
(3, 4, 'Stuffed crust and small sized', 3, '65.00'),
(4, 4, 'Cheese stuffed crust and large sized', 2, '34.00'),
(5, 2, 'Pizza mia crust and large sized', 1, '40.00'),
(5, 25, 'Large milkshake', 2, '10.00'),
(7, 3, 'Thin\'n crispy crust and large size', 4, '32.00'),
(8, 2, 'Cheese crust and medium size', 2, '40.00');
```

Figure 55: Details of implementation of order_items table

```sql
--
-- Table structure for table `payment`
--

CREATE TABLE `payment` (
  `payment_id` int(10) NOT NULL,
  `customer_id` int(7) NOT NULL,
  `payment_reference` varchar(100) NOT NULL,
  `amount` decimal(10,2) NOT NULL,
  `payment_date` datetime NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='table to store payments''';

--
-- Dumping data for table `payment`
--

INSERT INTO `payment` (`payment_id`, `customer_id`, `payment_reference`, `amount`, `payment_date`) VALUES
(1, 1, '73484010', '34.00', '2020-12-10 03:58:44'),
(2, 2, '725981730', '23.00', '2020-11-10 04:58:44'),
(3, 3, '3886437', '33.00', '2020-07-10 04:59:45'),
(4, 4, '736739109', '12.00', '2020-04-10 05:00:09'),
(5, 5, '93714309', '44.00', '2020-02-10 05:00:09'),
(6, 6, '94367910', '36.00', '2020-12-09 05:00:55'),
(7, 7, '8874910', '43.00', '2020-12-06 05:00:55'),
(8, 8, '0043837', '35.00', '2020-12-01 05:00:55'),
(9, 9, '7463562', '27.00', '2020-12-02 05:00:55'),
(10, 10, '98558785', '52.00', '2020-12-01 05:00:55');
```

Figure 56: Details of implementation of payment table

```
--
-- Indexes for dumped tables
--
--
-- Indexes for table `address`
--
ALTER TABLE `address`
  ADD PRIMARY KEY (`address_id`),
  ADD KEY `customer_id` (`customer_id`);

--
-- Indexes for table `customer`
--
ALTER TABLE `customer`
  ADD PRIMARY KEY (`customer_id`);

--
-- Indexes for table `store`
--
ALTER TABLE `store`
  ADD PRIMARY KEY (`store_id`);

--
-- Indexes for table `staff`
--
ALTER TABLE `staff`
  ADD PRIMARY KEY (`staff_id`),
  ADD KEY `store_id` (`store_id`);

--
-- Indexes for table `menus`
--
ALTER TABLE `menus`
  ADD PRIMARY KEY (`menu_id`),
  ADD KEY `store_id` (`store_id`);

--
-- Indexes for table `menu_items`
--
ALTER TABLE `menu_items`
  ADD PRIMARY KEY (`menu_id`,`product_id`),
  ADD KEY `product_id` (`product_id`);

--
-- Indexes for table `products`
--
ALTER TABLE `products`
  ADD PRIMARY KEY (`product_id`);

--
-- Indexes for table `pizza`
--
ALTER TABLE `pizza`
  ADD PRIMARY KEY (`pizza_id`),
  ADD KEY `product_id` (`product_id`);

--
-- Indexes for table `pizza_specifications`
--
ALTER TABLE `pizza_specifications`
  ADD PRIMARY KEY (`pizza_id`,`crust_id`,`size_id`),
  ADD KEY `pizza_id` (`pizza_id`),
  ADD KEY `crust_id` (`crust_id`),
  ADD KEY `size_id` (`size_id`);

--
-- Indexes for table `crust`
--
ALTER TABLE `crust`
  ADD PRIMARY KEY (`crust_id`);
```

```
--
-- Indexes for table `size`
--
ALTER TABLE `size`
  ADD PRIMARY KEY (`size_id`);

--
-- Indexes for table `ingredients`
--
ALTER TABLE `ingredients`
  ADD PRIMARY KEY (`ingredient_id`);

--
-- Indexes for table `pizza_ingredients`
--
ALTER TABLE `pizza_ingredients`
  ADD PRIMARY KEY (`pizza_id`,`ingredient_id`),
  ADD KEY `ingredient_id` (`ingredient_id`);

--
-- Indexes for table `non_pizza`
--
ALTER TABLE `non_pizza`
  ADD PRIMARY KEY (`non_pizza_id`),
  ADD KEY `product_id` (`product_id`);

--
-- Indexes for table `orders`
--
ALTER TABLE `orders`
  ADD PRIMARY KEY (`order_id`),
  ADD KEY `customer_id` (`customer_id`),
  ADD KEY `staff_id` (`staff_id`),
  ADD KEY `store_id` (`store_id`);

--
-- Indexes for table `order_items`
--
ALTER TABLE `order_items`
  ADD PRIMARY KEY (`order_id`,`product_id`),
  ADD KEY `product_id` (`product_id`);

--
-- Indexes for table `payment`
--
ALTER TABLE `payment`
  ADD PRIMARY KEY (`payment_id`),
  ADD KEY `customer_id` (`customer_id`);
```

Figure 57: Details of the indexes

```sql
-- Constraints for table `address`
--
ALTER TABLE `address`
  ADD CONSTRAINT `address_ibfk_1` FOREIGN KEY (`customer_id`) REFERENCES `customer` (`customer_id`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Constraints for table `staff`
--
ALTER TABLE `staff`
  ADD CONSTRAINT `staff_ibfk_1` FOREIGN KEY (`store_id`) REFERENCES `store` (`store_id`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Constraints for table `menus`
--
ALTER TABLE `menus`
  ADD CONSTRAINT `menus_ibfk_1` FOREIGN KEY (`store_id`) REFERENCES `store` (`store_id`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Constraints for table `menu_items`
--
ALTER TABLE `menu_items`
  ADD CONSTRAINT `menu_items_ibfk_2` FOREIGN KEY (`product_id`) REFERENCES `products` (`product_id`) ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `menu_items_ibfk_3` FOREIGN KEY (`menu_id`) REFERENCES `menus` (`menu_id`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Constraints for table `pizza`
--
ALTER TABLE `pizza`
  ADD CONSTRAINT `pizza_ibfk_1` FOREIGN KEY (`product_id`) REFERENCES `products` (`product_id`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Constraints for table `pizza_specifications`
--
ALTER TABLE `pizza_specifications`
  ADD CONSTRAINT `pizza_specifications_ibfk_1` FOREIGN KEY (`pizza_id`) REFERENCES `pizza` (`pizza_id`) ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `pizza_specifications_ibfk_2` FOREIGN KEY (`crust_id`) REFERENCES `crust` (`crust_id`) ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `pizza_specifications_ibfk_3` FOREIGN KEY (`size_id`) REFERENCES `size` (`size_id`) ON DELETE CASCADE ON UPDATE CASCADE;


-- Constraints for table `pizza_ingredients`
--
ALTER TABLE `pizza_ingredients`
  ADD CONSTRAINT `pizza_ingredients_ibfk_1` FOREIGN KEY (`ingredient_id`) REFERENCES `ingredients` (`ingredient_id`) ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `pizza_ingredients_ibfk_2` FOREIGN KEY (`pizza_id`) REFERENCES `pizza` (`pizza_id`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Constraints for table `non_pizza`
--
ALTER TABLE `non_pizza`
  ADD CONSTRAINT `non_pizza_ibfk_1` FOREIGN KEY (`product_id`) REFERENCES `products` (`product_id`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Constraints for table `orders`
--
ALTER TABLE `orders`
  ADD CONSTRAINT `orders_ibfk_1` FOREIGN KEY (`customer_id`) REFERENCES `customer` (`customer_id`),
  ADD CONSTRAINT `orders_ibfk_2` FOREIGN KEY (`store_id`) REFERENCES `store` (`store_id`),
  ADD CONSTRAINT `orders_ibfk_3` FOREIGN KEY (`staff_id`) REFERENCES `staff` (`staff_id`);

--
-- Constraints for table `order_items`
--
ALTER TABLE `order_items`
  ADD CONSTRAINT `order_items_ibfk_1` FOREIGN KEY (`product_id`) REFERENCES `products` (`product_id`),
  ADD CONSTRAINT `order_items_ibfk_2` FOREIGN KEY (`order_id`) REFERENCES `orders` (`order_id`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Constraints for table `payment`
--
ALTER TABLE `payment`
  ADD CONSTRAINT `payment_ibfk_1` FOREIGN KEY (`customer_id`) REFERENCES `customer` (`customer_id`) ON DELETE CASCADE ON UPDATE CASCADE;
```

Figure 58: Details of the constraints

```
--
-- AUTO_INCREMENT for table `address`
--
ALTER TABLE `address`
  MODIFY `address_id` int(10) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=21;

--
-- AUTO_INCREMENT for table `customer`
--
ALTER TABLE `customer`
  MODIFY `customer_id` int(10) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=11;

--
-- AUTO_INCREMENT for table `store`
--
ALTER TABLE `store`
  MODIFY `store_id` int(5) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=11;

--
-- AUTO_INCREMENT for table `staff`
--
ALTER TABLE `staff`
  MODIFY `staff_id` int(10) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=21;

--
-- AUTO_INCREMENT for table `menus`
--
ALTER TABLE `menus`
  MODIFY `menu_id` int(5) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=21;

--
-- AUTO_INCREMENT for table `products`
--
ALTER TABLE `products`
  MODIFY `product_id` int(10) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=27;

--
-- AUTO_INCREMENT for table `pizza`
--
ALTER TABLE `pizza`
  MODIFY `pizza_id` int(10) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=7;

--
-- AUTO_INCREMENT for table `crust`
--
ALTER TABLE `crust`
  MODIFY `crust_id` int(5) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=7;

--
-- AUTO_INCREMENT for table `size`
--
ALTER TABLE `size`
  MODIFY `size_id` int(5) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;

--
-- AUTO_INCREMENT for table `ingredients`
--
ALTER TABLE `ingredients`
  MODIFY `ingredient_id` int(7) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=13;

--
-- AUTO_INCREMENT for table `non_pizza`
--
ALTER TABLE `non_pizza`
  MODIFY `non_pizza_id` int(10) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=22;

--
-- AUTO_INCREMENT for table `orders`
--
ALTER TABLE `orders`
  MODIFY `order_id` int(10) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=9;


--
-- AUTO_INCREMENT for table `payment`
--
ALTER TABLE `payment`
  MODIFY `payment_id` int(10) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=11;
```

Figure 59: Details of auto increment

4. SQL Queries

The queries in this section consist of SQL SELECT queries designed to imitate real business situations and their business purpose.

There are a total of 5 queries shown and all of them are SELECT queries. SELECT statement selects data from the database and stores the data in a result table. The queries also consists of at least a JOIN and WHERE or any other function.

4.1 Query 1

4.1.1 Business purpose of the query

As a marketing manager, I would like to know the customers who have an account on the website, but failed to make an order, providing they have agreed to receive offers through email. I would like to send them a voucher coupon.

4.1.2 Query in natural language

Show names and emails of customers who have an account on the website but have failed to make an order. Customers who agreed to receive email offers have been displayed.

4.1.3 SQL code and output

```
SELECT DISTINCT CONCAT ( customer.first_name,' ', customer.last_name ) AS 'Customer Name',
                     customer.email AS 'Email'
     FROM
        customer

            LEFT JOIN orders
            ON orders.customer_id = customer.customer_id
     WHERE orders.order_id IS NULL
     AND customer.email_offers != 0

     GROUP BY customer.last_name ASC
```

Figure 60: Query 1 SQL code

Customer Name	Email
Kylee Haley	gerardo.mraz@example.net
Bonita O'Keefe	rempel.theresia@example.com

Figure 61: Query 1 output

4.1.4 Explanation of query's output

The query produced the output that was expected, out of 10 customers from the database there are 5 customers who haven't made an order but only 2 have agreed to receive email offers and others have not.

4.2 Query 2

4.2.1 Business purpose of the query

As a purchasing manager, I would like to know which pizza was best selling, so I can put it in an upcoming promotional offer.

4.2.2 Query in natural language

Show the names and product ID's of the best selling products along with the number of items sold. Order the list by number of items sold, in descending order.

4.2.3 SQL code and output

```
SELECT products.product_id AS 'Product ID',
       products.name AS 'Product name',
       SUM(order_items.quantity) AS 'Number sold'

       FROM
           order_items

       JOIN products
           ON(products.product_id = order_items.product_id)
       INNER JOIN pizza
           WHERE(pizza.product_id = order_items.product_id)


       GROUP BY products.product_id
       ORDER BY SUM(order_items.quantity) DESC
       LIMIT 1
```

Figure 63: Query 2 SQL code

Product ID	Product name	Number sold
3	Super Supreme	6

Figure 64: Query 2 output

4.2.4 Explanation of query's output

Product ID	Product name	Number sold
3	Super Supreme	6
4	Dynamite Shrimp	5
2	Cheeky Chicken	4

The table above shows all the pizzas that were sold in descending order and the query produces the best selling pizza which was "Super Supreme".

4.3 Query 3

4.3.1 Business purpose for the query

As the line manager, I want to know how many orders a staff member has registered and the total sales generated by the staff member, providing they have registered an order, and to give the staff member with the most registered orders a pay raise.

4.3.2 Query in natural language

Displays the name of the staff member that has registered the most orders and the total sales generated by the staff member.

4.3.3 SQL code and output

```
SELECT          staff.staff_id AS 'Staff ID',
                CONCAT (staff.first_name,' ', staff.last_name) AS 'Staff name',
                COUNT(orders.staff_id) AS 'Number of Orders',
                SUM(orders.order_total) AS 'Total sales'
          FROM
              staff

              INNER JOIN orders
              ON staff.staff_id = orders.staff_id

              GROUP BY orders.staff_id
              ORDER BY COUNT(orders.staff_id) DESC
```

Figure 65: SQL code for Query 3

Staff ID	Staff name	Number of Orders	Total sales
1	Camila Reilly	4	149.00
13	Quinn Lockman	2	80.50
6	Annie Padberg	1	36.00
3	Rosalinda Buckridge	1	45.00

Figure 66: output for query 3

4.3.4 Explanation of query's output

The query produces all the names of the staff member and their staff IDs' of staff that have registered an order and displays the number of orders each staff has registered in descending order and total sales generated by the staff member.

4.4 Query 4

4.4.1 Business purpose for the query

As a software developer, I would like to send an email to the customer to notify them that they can collect the order once the order status changes to “ready for collection”.

4.4.2 Query in natural language

Send an email to the customer regarding order collection as the order status is “ready for collection”.

4.4.3 SQL code and output

```
SELECT orders.order_id AS 'Order ID',
       orders.order_status AS 'Order status',
       CONCAT (customer.first_name,' ', customer.last_name) AS 'Customer name',
       customer.email AS 'Customer Email'

       FROM
           customer

           LEFT JOIN orders
               ON customer.customer_id = orders.customer_id

       WHERE orders.order_status = "ready for collection"

       GROUP BY customer.last_name ASC
```

Figure 67: SQL code for query 4

Order ID	Order status	Customer Name	Customer Email
8	ready for collection	Darrel Braun	bergnaum.providenci@example.com
2	ready for collection	Bonita O'Keefe	rempel.theresia@example.com
7	ready for collection	Romaine Swaniawski	chris46@example.com

Figure 68: output for query 4

4.4.4 Explanation of query's output

The query produces the result expected and displays all the orders that are ready to be collected. The table below shows all the orders placed. Exactly 3 orders are ready for collection.

order_id	customer_id	staff_id	store_id	order_date	delivery_fee	order_total	order_status	promo_code
1	1	1	1	2020-12-09 21:50:23	5.00	30.00	out for delivery	PZ50
2	5	1	1	2019-12-09 23:49:15	NULL	27.00	ready for collection	NULL
3	3	1	1	2017-12-09 23:02:31	6.50	40.00	out for delivery	NULL
4	1	3	2	2020-12-10 07:58:20	5.00	45.00	completed	NULL
5	4	6	3	2018-01-09 02:59:39	NULL	36.00	being prepared	RE20
6	8	1	1	2017-12-09 17:12:47	4.50	52.00	completed	NULL
7	2	13	7	2020-12-07 17:18:35	NULL	33.00	ready for collection	45WE
8	3	13	7	2020-12-09 23:03:04	4.50	47.50	ready for collection	SD35

4.5 Query 5

4.5.1 Business purpose for the query

As the store manager I would like to notify the store to update its menu if it has been added before the year 2017 to attract more customers.

4.5.2 query in natural language

Show the name of the stores that have had their menu added before the year 2017

4.5.3 SQL code and output

```
SELECT store.name AS 'Store Name',
       menus.menu_id AS 'Menu ID',
       menus.date_added AS 'Date Added for menu'

FROM store
       INNER JOIN menus
        ON menus.store_id = store.store_id

        WHERE YEAR(menus.date_added) < '2017'

        GROUP BY store.name ASC
```

Figure 70: SQL code for query 5

Store Name	Menu ID	Date Added for menu
Pizza Hut - Fermville	2	2015-12-02 09:51:01
Pizza Hut - New soledad	3	2012-12-07 14:42:37

Figure 71: output for query 5

4.5.4 Explanation of query's output

The query produces all the names of the stores that have had their menu added before 2017 with store name in ascending order. The table below shows all the menus added for each store and it can be seen that only 2 stores have had their menus added before 2017.

menu_id	store_id	name	description	date_added
1	1	Starters and pizzas for Ferm store	This menu consists of all pizzas - Fermville	2017-12-09 09:51:01
2	1	Drinks and sides menu for Ferm store	This menu consists of all sides - Fermville	2015-12-02 09:51:01
3	2	Drinks menu for soledad store	This menu consists of all the drinks - soledad	2012-12-07 14:42:37
4	2	Starters and pizzas for new soledad store	This is the starters menu for soledad Store	2020-12-07 03:38:46
5	3	Pizzas menu for south store	this is the menu for all pizzas in south store	2020-12-01 03:38:46
6	3	Drinks menu for south store	This menu stores all drinks for south store	2020-12-10 03:43:19
7	4	Starters menu for west store	Stores all starters for the west store	2020-12-03 03:43:19
8	4	Drinks menu for west	This stores all drinks for west store	2020-11-10 03:45:27
9	5	Starters menu for jasper store	Stores all starters for the jasper store	2020-12-01 03:47:38
10	5	Drinks menu for jasper store	This stores all drinks for jasper store	2020-11-02 03:47:53
11	6	Starters menu for lionel store	Stores all starters for the lionel store	2020-10-09 03:48:00
12	6	Drinks menu for leionl store	This stores all drinks for lionel store	2020-10-18 03:48:05
13	7	Starters menu for rowland store	Stores all starters for the rowland store	2020-12-05 03:48:16
14	7	Drinks menu for rowland store	This stores all drinks for rowland store	2020-12-03 03:48:22
15	8	Starters menu for lockman store	Stores all starters for the lockman store	2020-11-20 03:48:27
16	8	Drinks menu for lockman store	This stores all drinks for lockman store	2020-11-22 03:48:32
17	9	Starters menu for harmon store	Stores all starters for the harmon store	2020-11-25 03:48:38
18	9	Drinks menu for harmon store	This stores all drinks for harmon store	2020-10-21 03:48:45
19	10	Starters menu for lake store	Stores all starters for the lake store	2020-11-29 03:48:50
20	10	Drinks menu for lake store	This stores all drinks for lake store	2020-10-09 03:48:56

REFERENCES

items, H., Vernon, M. and Altamimi, A., 2020. *How To Get A List/Result Of Best Selling Items*. [online] Database Administrators Stack Exchange. Available at: <https://dba.stackexchange.com/questions/52476/how-to-get-a-list-result-of-best-selling-items> [Accessed 2 December 2020].

Drkusic, E., 2020. *Learn SQL: SQL Query Examples*. [online] SQL Shack - articles about database auditing, server performance, data recovery, and more. Available at: <https://www.sqlshack.com/learn-sql-sql-query-examples/> [Accessed 11 November 2020].

Guru99.com. 2020. *What Is Normalization? 1NF, 2NF, 3NF, BCNF Database Example*. [online] Available at: <https://www.guru99.com/database-normalization.html> [Accessed 3 December 2020].

W3schools.com. 2020. *PHP Mysql Insert Multiple Records*. [online] Available at: <https://www.w3schools.com/php/php_mysql_insert_multiple.asp> [Accessed 6 December 2020].