

Name: Rathnayaka Samarakoon

Student Reference Number: 10820263

Module Code: PUSL2019	Module Name: Informational Management and Retrieval
Coursework Title: PUSL2019 Informational Management and Retrieval Coursework	
Deadline Date: 23/01/2023	Member of staff responsible for coursework: 06
Programme: Sql	

Please note that University Academic Regulations are available under Rules and Regulations on the University website www.plymouth.ac.uk/studenthandbook.

Group work: please list all names of all participants formally associated with this work and state whether the work was undertaken alone or as part of a team. Please note you may be required to identify individual responsibility for component parts.

Maleesha S Ramasinghe - 10820265
 Rankoth Jayasekara - 10820280
 Navoda Ravishani - 10820266
 Galappaththi Dinujaya - 10820276
 Bokaragoda Wanshanatha - 10820934

We confirm that we have read and understood the Plymouth University regulations relating to Assessment Offences and that we are aware of the possible penalties for any breach of these regulations. We confirm that this is the independent work of the group.

Signed on behalf of the group: 

Individual assignment: ***I confirm that I have read and understood the Plymouth University regulations relating to Assessment Offences and that I am aware of the possible penalties for any breach of these regulations. I confirm that this is my own independent work.***

Signed :

Use of translation software: failure to declare that translation software or a similar writing aid has been used will be treated as an assessment offence.

I *have not used translation software.

If used, please state name of software.....

Overall mark _____ % Assessors Initials _____ Date _____

*Please delete as appropriateSci/ps/d:/students/cwkfrontcover/2013/14

TABLE OF CONTENTS

SECTION 1.....	3
1.1 INTRODUCTION	3
1.2 EER DIAGRAM.....	4
1.3 EER DIAGRAM ASSUMPTIONS.....	5
1.4 RELATIONAL MAPPING.....	7
1.5 DATA NORMALIZATION.....	8
1.6 DATA DICTIONARY	12
SECTION 2.....	16
2.1 TABLE STATEMENTS	16
2.2 DATABASE DIAGRAMS	21
2.3 INSERT STATEMENTS	22
SECTION 3.....	30
3.1 TRIGGER STATEMENTS	30
3.2 FUNCTION STATEMENTS	33
3.3 VIEW STATEMENTS.....	37
3.4 PROCEDURE STATEMENTS.....	39
SECTION 4.....	42
4.1 CRITICAL APPRAISAL OF YOUR SOLUTION	42
4.1 COMMENTS ON FUTURE IMPLEMENTATION.....	42

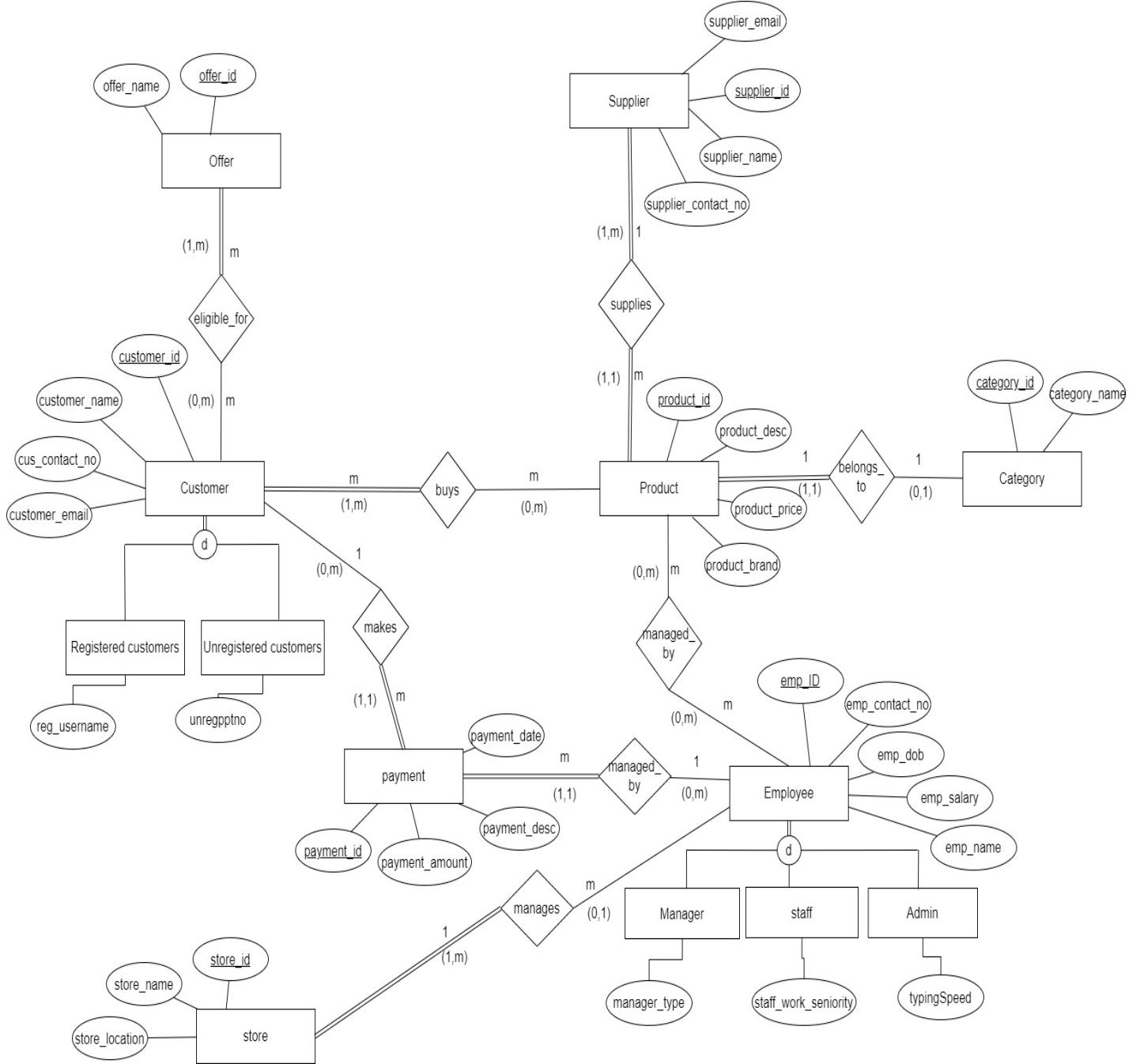
Section 1 - Introduction

This database application is created to a supermarket to manage their total tasks including the stock management operations, billing operations, and finance operations. A supermarket is a large collection of data, thus managing this data requires a complicated and advanced database. This will aid with time management, boost supermarket productivity, and provide a spotless service.

Customers browse the aisles of a supermarket and select the items they want to purchase. A customer takes his selected items to the cashier to pay for them after making the selection. A supermarket needs a database system that can properly determine the cost of a customer's purchases, work to guarantee that the supermarket doesn't run out of things that consumers wish to buy and accept payments from customers.

The database of the system will store all the necessary information such as customer details, employee details, product details, and payment details, category details, store details, offer details, and supplier details. The purpose of the database design for the supermarket management system is to support the management of the process for purchasing goods, billing, and making the purchase transaction quick and accurate while maintaining and protecting the sales records. It makes it easier for both the owner and the employee to control or influence corporate transactions.

EER Diagram



ER diagram assumptions

Entity related assumptions

- Assuming customer can either be registered or unregistered and cannot belong to both groups at the same time
- Assuming employee can be either a manager or admin or staff and one employee cannot belong to more than one of these groups.
- Assuming every employee should belong to one of the groups either manager or admin or staff
- Assuming every customer should belong to one of the groups either registered or unregistered

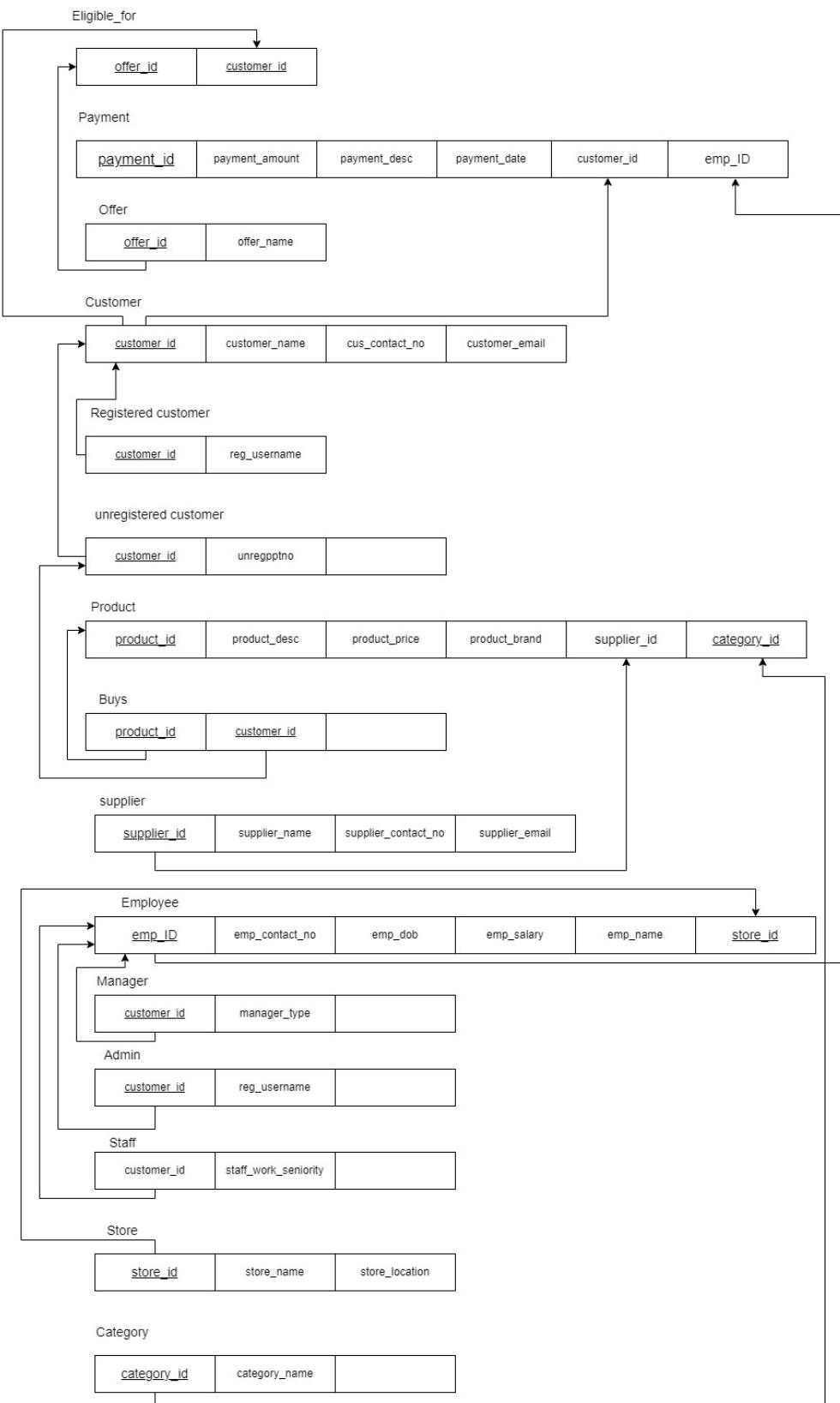
Connectivity related assumptions

- Assuming a customer can be eligible for many offers and one offer may be eligible for many customers
- Assuming a supplier can supply many products a product belongs to only one supplier
- Assuming a customer can buy many products and a product can be purchased by many customers
- Assuming a customer can make many payments but a payment can belong only to one customer
- Assuming a product can be managed by many employees and many employees can manage many products at the same time
- Assuming many employees can manage only one store at a particular time
- Assuming a product should have a category and a category can have a product

Cardinality related assumptions

- Assuming every payment has a customer and every customer does not make a payment
- Assuming every payment is handled by an employee but every employee does not handle payments
- Assuming every store is managed by an employee but every employee does not manage stores
- Assuming every product belongs to a category but every category does not have products
- Assuming each and every supplier supplies products and each and every product has a supplier
- Assuming every customer is not eligible for offers but every offer must have customers
- Assuming every customer purchase products but every product is not purchased by a customer

Relational mapping



Data Normalization

Customer

customer_id	customer_name	cus_contact_no	customer_email

The above table is 1st normal form because there are,

- No Non atomic values
- No nested relations
- No multi-valued attributes

The above table is in 2nd normal form because there are

- No partial dependencies

The above table is in 3rd normal form because there are

- No transitive dependencies

Product

product_id	product_desc	product_price	product_brand	supplier_id	category_id

The above table is 1st normal form because there are,

- No Non atomic values
- No nested relations
- No multi-valued attributes

The above table is in 2nd normal form because there are

- No partial dependencies

The above table is in 3rd normal form because there are

- No transitive dependencies

Payment

payment_id	payment_amount	payment_desc	payment_date	customer_id	emp_ID

The above table is 1st normal form because there are,

- No Non atomic values
- No nested relations
- No multi-valued attributes

The above table is in 2nd normal form because there are

- No partial dependencies

The above table is in 3rd normal form because there are

- No transitive dependencies

Employee

emp_ID	emp_contact_no	emp_dob	emp_salary	emp_name	store_id

The above table is 1st normal form because there are,

- No Non atomic values
- No nested relations
- No multi-valued attributes

The above table is in 2nd normal form because there are

- No partial dependencies

The above table is in 3rd normal form because there are

- No transitive dependencies

Store

store_id	store_name	store_location

The above table is 1st normal form because there are,

- No Non atomic values
- No nested relations
- No multi-valued attributes

The above table is in 2nd normal form because there are

- No partial dependencies

The above table is in 3rd normal form because there are

- No transitive dependencies

Category

category_id	category_name

The above table is 1st normal form because there are,

- No Non atomic values
- No nested relations
- No multi-valued attributes

The above table is in 2nd normal form because there are

- No partial dependencies

The above table is in 3rd normal form because there are

- No transitive dependencies

Offer

offer_id	offer_name

The above table is 1st normal form because there are,

- No Non atomic values
- No nested relations
- No multi-valued attributes

The above table is in 2nd normal form because there are

- No partial dependencies

The above table is in 3rd normal form because there are

- No transitive dependencies

Supplier

supplier_id	supplier_name	supplier_contact_no	supplier_email

The above table is 1st normal form because there are,

- No Non atomic values
- No nested relations
- No multi-valued attributes

The above table is in 2nd normal form because there are

- No partial dependencies

The above table is in 3rd normal form because there are

- No transitive dependencies

Data dictionary

customer

Field Name	Data type	Field size	Description	Constraints
Customer_id	int	-	Unique ID for all customers	Primary key
Customer_name	varchar	25	Name of customer	Not null
Cus_contact_no	char	10	Contact no of the customer	Not null
Customer_email	varchar	70	Email address of the customer	Not null

product

Field Name	Data type	Field size	Description	Constraints
Product_id	int	-	Unique ID for all products	Primary key
Product_des	varchar	130	Description of each product	Not null
Product_price	decimal	5	Price of each product	Not null
product_brand	varchar	20	Brand of each product	Not null
Supplier_id	int	-	supplier id for all suppliers	Foreign key Not null
Category_id	int	-	ID for all categories	Foreign key Not null

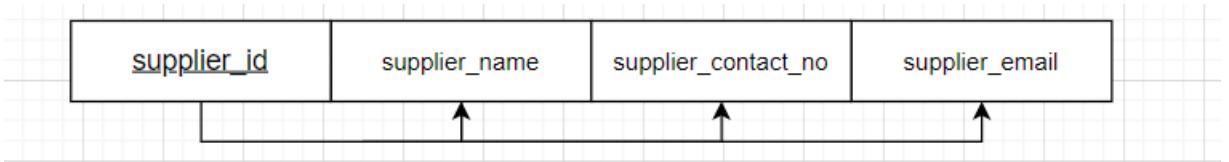
employee

Field Name	Data type	Field size	Description	Constraints
Emp_ID	int	-	Unique ID for all employees	Primary key
Emp_contact_no	char	10	Contact number of each employee	Not null
Emp_dob	date	-	Date of birth of employee	Not null
Emp_salary	decimal	10	Salary of employees	Not null
Emp_name	varchar	30	Name of employees	Not null
Store_id	int	-	ID of the store	Foreign key Not null

payment

Field Name	Data type	Field size	Description	Constraints
Payment_id	int	-	Unique ID for all payments	Primary key
payment_amount	decimal	10	Payment amount for each transaction	Not null
payment_desc	varchar	100	Description of the payment	Not null
payment_date	date	-	Date of the payment	Not null
customer_id	int	-	Id of the customer	Foreign key Not null
emp_ID	int	-	ID of the employee	Foreign key Not null

Supplier



Field Name	Data type	Field size	Description	Constraints
Supplier_id	int	-	Unique ID for all suppliers	Primary key
Supplier_name	varchar	30	Supplier name	Not null
Supplier_contact_no	char	10	Contact number of the supplier	Not null
Supplier_email	varchar	65	Email of the supplier	Not null

store

Field Name	Data type	Field size	Description	Constraints
store_id	int	-	Unique ID for all stores	Primary key
store_name	varchar	25	Store name	Not null
store_location	varchar	20	Location of the store	Not null

category

Field Name	Data type	Field size	Description	Constraints
Category_id	int	-	Unique ID for all categories	Not null Unique
Category_name	varchar	45	Name of the category	Not null

offer

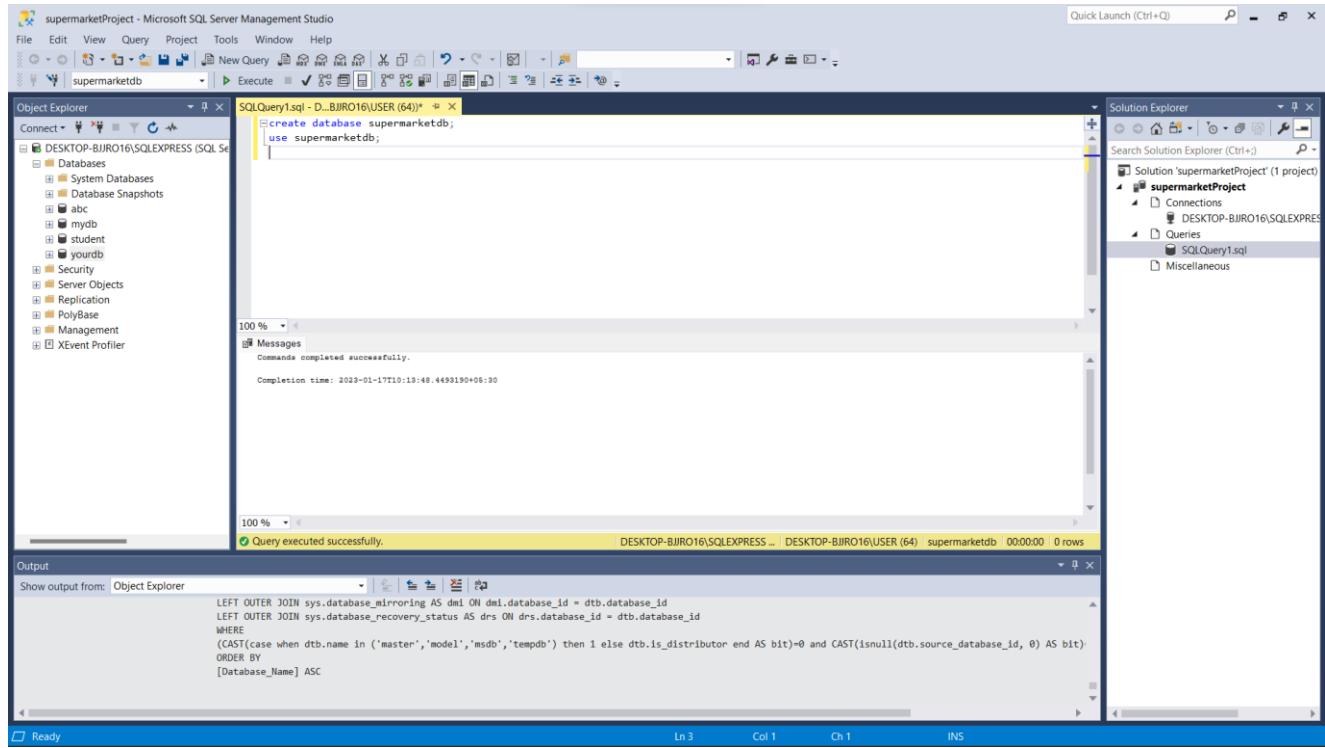
Field Name	Data type	Field size	Description	Constraints
Offer_id	int	-	Unique ID for all offers	Primary key
Offer_name	varchar	45	Name of the offer	Not null

supplier

Field Name	Data type	Field size	Description	Constraints
Supplier_id	int	-	Unique ID for all suppliers	Primary key
Supplier_name	Varchar	25	Name of the supplier	Not null
Supplier_email	varchar	20	Email address of the email	Not null
Supplier_contact_no	char	10	Contact number of the supplier	Not null

Section 2 - Create table statements

Database setup



The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a connection to 'DESKTOP-BJIR016\SQLEXPRESS' is selected, showing databases like 'master', 'model', 'msdb', 'tempdb', 'abc', 'mydb', 'student', 'yourdb', 'Security', 'Server Objects', 'Replication', 'PolyBase', and 'Management'. A new query window titled 'SQLQuery1.sql' is open, containing the following T-SQL code:

```
create database supermarketdb;
use supermarketdb;
```

The 'Messages' pane below the query window displays the output: 'Commands completed successfully.' and 'Completion time: 2023-01-17T10:13:48.4493190+05:30'. The 'Solution Explorer' pane on the right shows a project named 'supermarketProject' with a file 'SQLQuery1.sql' under the 'Queries' node.

Create table statements

Customer table

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a connection to 'DESKTOP-BJIR016\SQLEXPRESS' is selected, and the database 'supermarketdb' is chosen. In the center pane, a query window titled 'SQLQuery1.sql' contains the following SQL code:

```
CREATE TABLE customer(
    customer_id INT PRIMARY KEY,
    customer_name VARCHAR(25) NOT NULL,
    cus_contact_no CHAR(10) NOT NULL,
    customer_email VARCHAR(70) NOT NULL
);
```

The 'Messages' pane at the bottom shows the command completed successfully with a completion time of 2023-01-17T10:42:47.1089442+05:30.

In the Solution Explorer, the project 'supermarketProject' is expanded, showing 'Connections', 'Queries', and 'SQLQuery1.sql'. The Output pane displays the creation of the 'customer' table and the properties of the 'supermarketdb' database.

Supplier table

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a connection to 'DESKTOP-BJIR016\SQLEXPRESS' is selected, and the database 'supermarketdb' is chosen. In the center pane, a query window titled 'SQLQuery1.sql' contains the following SQL code:

```
CREATE TABLE supplier(
    supplier_id INT PRIMARY KEY,
    supplier_name VARCHAR(30),
    supplier_contact_no CHAR(10),
    supplier_email VARCHAR(65)
);
```

The 'Messages' pane at the bottom shows the command completed successfully with a completion time of 2023-01-17T10:42:47.1089442+05:30.

In the Solution Explorer, the project 'supermarketProject' is expanded, showing 'Connections', 'Queries', and 'SQLQuery1.sql'. The Output pane displays the creation of the 'supplier' table and the properties of the 'supermarketdb' database.

Category table

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the 'supermarketProject' database is selected. In the center pane, a query window displays the following SQL code:

```
CREATE TABLE category(
    category_id INT PRIMARY KEY,
    category_name VARCHAR(45) NOT NULL
);
```

The status bar at the bottom indicates "Query executed successfully." and "Completion time: 2023-01-17T10:42:47.1089442+05:30".

Product

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the 'supermarketProject' database is selected. In the center pane, a query window displays the following SQL code:

```
CREATE TABLE product(
    product_id INT PRIMARY KEY,
    product_des VARCHAR(60) NOT NULL,
    product_price DECIMAL(5,2) NOT NULL,
    product_brand VARCHAR(20) NOT NULL,
    supplier_id INT FOREIGN KEY REFERENCES supplier(supplier_id),
    category_id INT FOREIGN KEY REFERENCES category(category_id)
);
```

The status bar at the bottom indicates "Query executed successfully." and "Completion time: 2023-01-17T10:24:59.5599834+05:30".

Store table

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the 'supermarketProject' database is selected. In the center pane, a query window displays the creation of a 'store' table:

```
create table store(
    store_id int primary key,
    store_name varchar(25) not null,
    store_location varchar(20) not null
);
```

The 'Messages' pane shows the command completed successfully with a completion time of 2023-01-17T10:25:44.2811917+05:30.

The Solution Explorer on the right shows the project structure with a 'Queries' folder containing 'SQLQuery1.sql'.

Employee

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the 'supermarketProject' database is selected. In the center pane, a query window displays the creation of an 'employee' table:

```
create table employee(
    emp_ID int primary key,
    emp_contact_no char(10) not null,
    emp_dob date not null,
    emp_salary decimal(10,2) not null,
    emp_name varchar(30) not null,
    store_id int foreign key references store(store_id)
);
```

The 'Messages' pane shows the command completed successfully with a completion time of 2023-01-17T10:24:24.6753308+05:30.

The Solution Explorer on the right shows the project structure with a 'Queries' folder containing 'SQLQuery1.sql'.

Offer table

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the database 'supermarketdb' is selected. In the center pane, a query window displays the creation script for the 'offer' table:

```
CREATE TABLE offer(
offer_id int primary key,
offer_name varchar(45) not null
);
```

The status bar at the bottom indicates 'Query executed successfully.' and provides execution details: DESKTOP-BJIR016\SQLEXPRESS ... DESKTOP-BJIR016\USER (64) supermarketdb 00:00:00 0 rows.

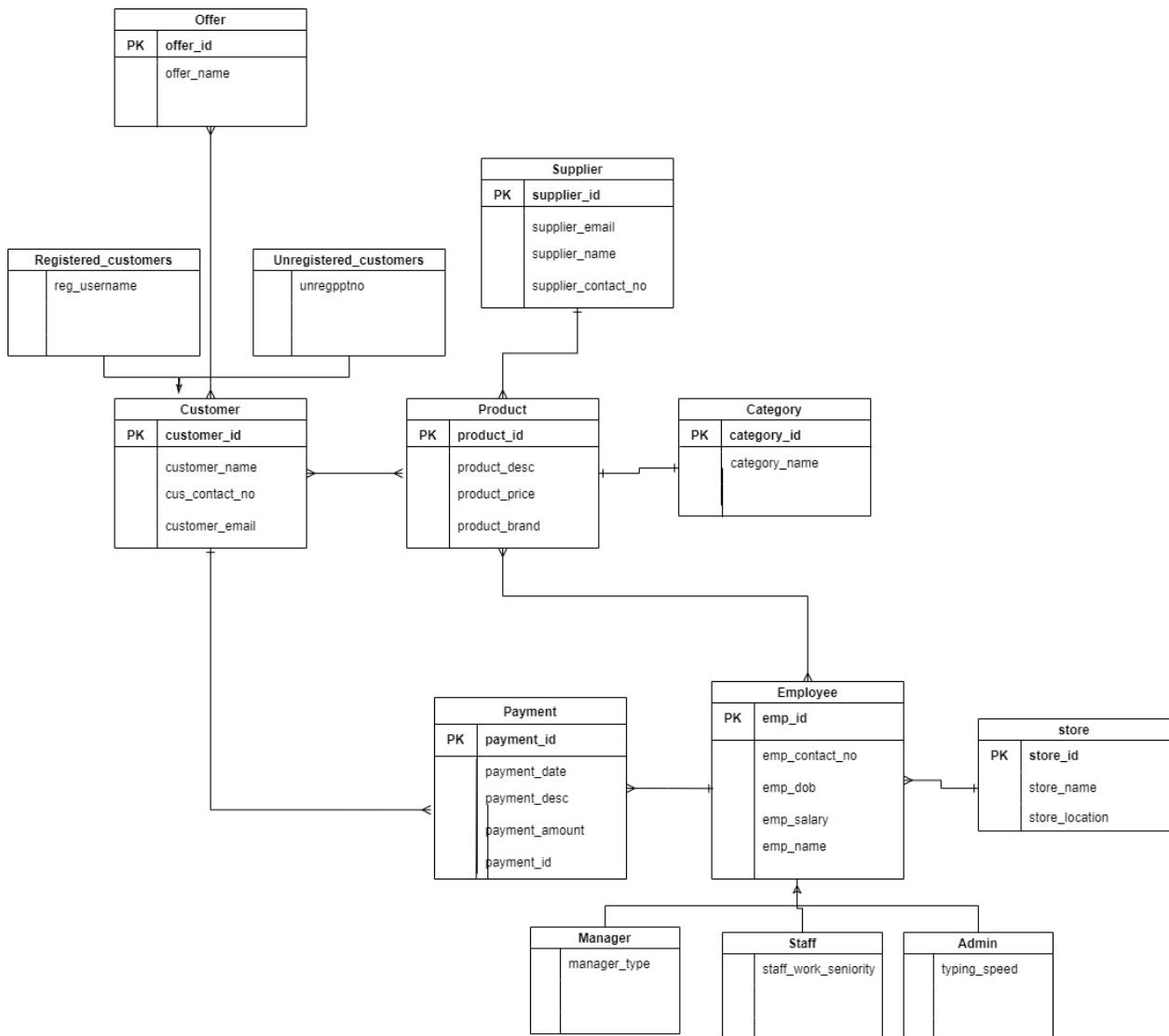
Payment table

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the database 'supermarketdb' is selected. In the center pane, a query window displays the creation script for the 'payment' table:

```
CREATE TABLE payment(
payment_id int primary key,
payment_amount decimal(10,2) not null,
payment_desc varchar(100) not null,
payment_date date not null,
customer_id int foreign key references customer(customer_id),
emp_ID int foreign key references employee(emp_ID)
);
```

The status bar at the bottom indicates 'Query executed successfully.' and provides execution details: DESKTOP-BJIR016\SQLEXPRESS ... DESKTOP-BJIR016\USER (64) supermarketdb 00:00:00 0 rows.

Database Diagram



Insert statements

Customer table

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database structure, including the supermarketdb database which contains tables like payment, customer, and employee. The main window displays a query results grid with 12 rows of data inserted into the customer table. The status bar at the bottom indicates the query completed with 0 errors.

customer_id	customer_name	customer_email
1	Senuda Dilsmith	'0778967973', 'senuda0@gmail.com'
2	Shihara Galappaththi	'0774759797', 'shihara@gmail.com'
3	Yuvindu Wanhanatha	'0776783970', 'ywindu@gmail.com'
4	Maleesha Shehari	'0773647867', 'maleesha@gmail.com'
5	Madhushika Lakkal	'0773678492', 'madhushika@gmail.com'
6	Navoda Ravishani	'0779867546', 'navoda@gmail.com'
7	Kasun Piyumal	'0773345612', 'kasun@gmail.com'
8	Shihan Sanjula	'0773457786', 'shihan@gmail.com'
9	Piyumi Hansamali	'0773457786', 'piyumi@gmail.com'
10	Sithara Sewandi	'0773457786', 'sithara@gmail.com'
11	Nuthara Dhananjalee	'0773457786', 'nuthara@gmail.com'
12	Newanji De Silva	'0773457786', 'newanji@gmail.com'

Supplier table

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database structure, including the 'supermarketdb' database which contains tables like 'customer' and 'supplier'. The main window displays a query results grid with 0 rows, showing the execution of an SQL script named 'SQLQuery3.sql'.

```
values(11,'Nuthara Dhananjalee','0773457786','nuthara@gmail.com');
insert into customer
values(12,'Newanji De Silva','0773457786','newanji@gmail.com');

select * from customer;

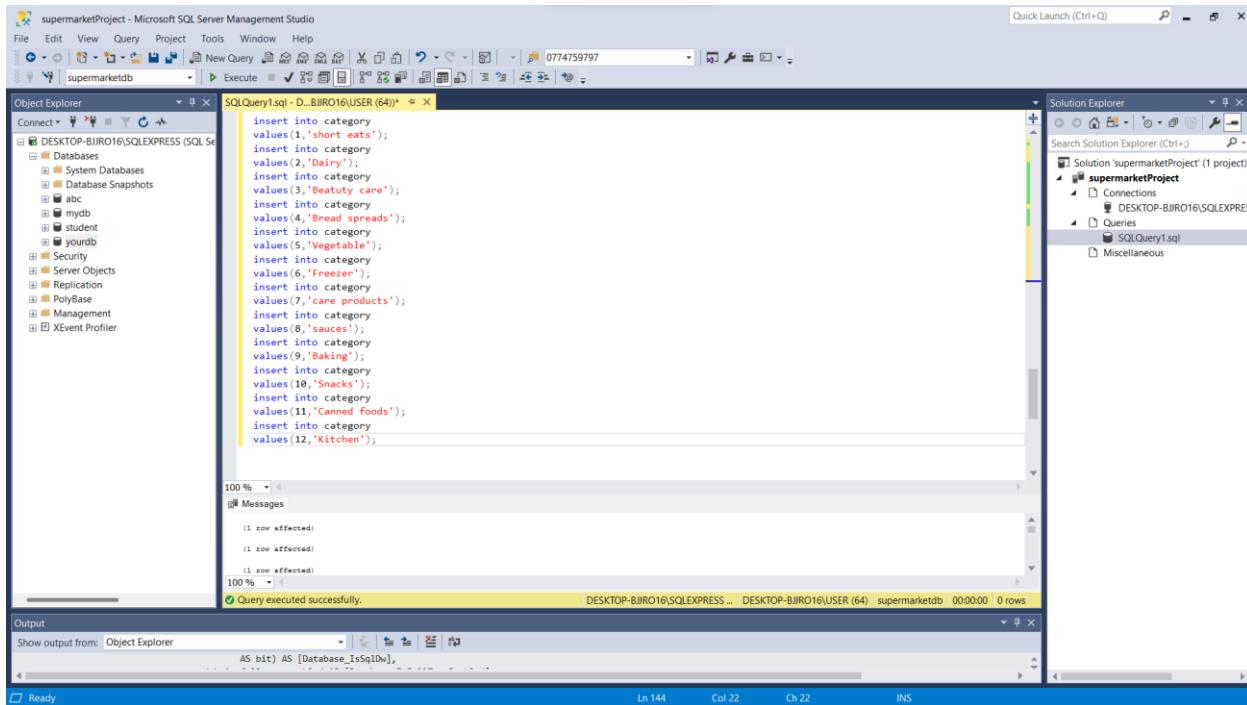
insert into supplier
values(101,'Kaveesha Sathsarani','0774567657','kaveesha@gmail.com');
insert into supplier
values(102,'Sachini Umayangana','0773334685','sachini@gmail.com');
insert into supplier
values(103,'Tharushi Devindi','0774567839','tharushi@gmail.com');
insert into supplier
values(104,'Kevin Nilnethu','0773409998','kevin@gmail.com');
insert into supplier
values(105,'Kushan Perera','0773432354','kushan@gmail.com');
insert into supplier
values(106,'Kavindu Hansaka','0773433457','kavindu@gmail.com');
insert into supplier
values(107,'Dasun Himayantha','0773798645','dasun@gmail.com');
insert into supplier
values(108,'Bihan Ranuka','0778899776','bihan@gmail.com');
insert into supplier
values(109,'Dilshan Perera','0771234987','dilshan@gmail.com');
insert into supplier
values(110,'Dumindu Chatthuranga','0779911224','dumindu@gmail.com');
insert into supplier
values(111,'Sanuka Nimsara','0778974568','sanuka@gmail.com');
insert into supplier
values(112,'Banuja Perera','0778886665','banuja@gmail.com');

select * from supplier;
```

Ln 51 Col 1 Ch 1 INS

ASUS-VIVOBOOK15\MySQL (16.0... | ASUS-VIVOBOOK15\kings... | supermarketdb | 00:00:00 | 0 rows

Category table



	category_id	category_name
1	1	short eats
2	2	Dairy
3	3	Beauty care
4	4	Bread spreads
5	5	Vegetable
6	6	Freezer
7	7	care products
8	8	sauces
9	9	Baking
10	10	Snacks
11	11	Canned foods
12	12	Kitchen

Product table

The screenshot shows the Microsoft SQL Server Management Studio interface. The central window displays an SQL query script named 'SQLQuery1.sql'. The script contains 12 INSERT statements for a 'product' table, each specifying a product ID, description, price, brand, supplier ID, and category ID. The execution results show 12 rows affected, indicating successful insertion. To the right, the Solution Explorer shows a project named 'supermarketProject' containing a connection to 'DESKTOP-BJIR016\SQLEXPRESS' and a query file 'SQLQuery1.sql'. Below the main window, the Output pane shows the results of the query execution.

```

SQLQuery1.sql - ...BJIR016(USER (64)) - X
[values(1,'submarine','540','JuicySub',101,1);
VALUES(2,'yoghurt','100','Newdale',102,2);
VALUES(3,'Hair gel','2000','A&D',103,3);
VALUES(4,'Peanut butter','3000','JK Products',104,4);
VALUES(5,'cucumber','200','keels',105,5);
VALUES(6,'smoothies','450','AB Smoothies',106,6);
VALUES(7,'toothpaste','200','Signal',107,7);
VALUES(8,'Hot sauce','1900','MD',108,8);
VALUES(9,'Cake icing','900','MR Cake',109,9);
VALUES(10,'cookies','1200','Little Lion',110,10);
VALUES(11,'Olives','1350','GreenVego',111,11);
VALUES(12,'Aluminium foils','890','KitchenEq',112,12);

100 % 100 %
1 row affected
100 % 100 %

Messages
[SQL] Query executed successfully.

Output
Show output from: Object Explorer
AS (bit) AS [Database_IsSqlDw], Ln 27 Col 1 Ch 1 INS
Ready

```

	product_id	product_des	product_price	product_brand	supplier_id	category_id
1	1	submarine	540.00	JuicySub	101	1
2	2	yoghurt	100.00	Newdale	102	2
3	3	Hair gel	2000.00	A&D	103	3
4	4	Peanut butter	3000.00	JK Products	104	4
5	5	cucumber	200.00	keels	105	5
6	6	smoothies	450.00	AB Smoothies	106	6
7	7	toothpaste	200.00	Signal	107	7
8	8	Hot sauce	1900.00	MD	108	8
9	9	Cake icing	900.00	MR Cake	109	9
10	10	cookies	1200.00	Little Lion	110	10
11	11	Olives	1350.00	GreenVego	111	11
12	12	Aluminium f...	890.00	KitchenEq	112	12

Store table

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a connection to 'DESKTOP-BJIRO16\SQLEXPRESS' is selected. In the Solution Explorer, there is a project named 'supermarketProject' with a file 'SQLQuery1.sql' containing the following script:

```
USE supermarketdb;
GO
CREATE TABLE store (
    store_id INT,
    store_name NVARCHAR(50),
    store_location NVARCHAR(50)
);
GO
INSERT INTO store
VALUES (1,'store1','Gampaha');
INSERT INTO store
VALUES (2,'store2','Homagama');
INSERT INTO store
VALUES (3,'store3','Aluthgama');
INSERT INTO store
VALUES (4,'store4','Minuwangoda');
INSERT INTO store
VALUES (5,'store5','Galle');
INSERT INTO store
VALUES (6,'store6','Matale');
INSERT INTO store
VALUES (7,'store7','Yakkala');
INSERT INTO store
VALUES (8,'store8','Mudungoda');
INSERT INTO store
VALUES (9,'store9','ududgampola');
INSERT INTO store
VALUES (10,'store10','Negambo');
INSERT INTO store
VALUES (11,'store11','Colombo');
INSERT INTO store
VALUES (12,'store12','Miriswatte');
```

The Messages pane shows the results of the execution:

```
1 row affected
1 row affected
1 row affected
100 %
```

A status bar at the bottom indicates "Query executed successfully." Below the main window, an Output tab is open, showing the results of the query:

store_id	store_name	store_location
1	store1	Gampaha
2	store2	Homagama
3	store3	Aluthgama
4	store4	Minuwangoda
5	store5	Galle
6	store6	Matale
7	store7	Yakkala
8	store8	Mudungoda
9	store9	ududgampola
10	store10	Negambo
11	store11	Colombo
12	store12	Miriswatte

Employee table

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database structure for 'supermarketdb'. The central pane displays a query window titled 'SQLQuery3.sql - A...BOOK15(kings (S4))' containing SQL code for inserting data into the 'store' and 'employee' tables. The status bar at the bottom indicates the session is connected to 'ASUS-VIVOBOOK15|MySQL (16.0...' and the command '00:00:00' has been executed.

```
insert into store
values(10,'store10','Negombo');
insert into store
values(11,'store11','Colombo');
insert into store
values(12,'store12','Miriswatte');

select * from store;

insert into employee
values(11,'0778999876','1990-11-13','20000','shehan perera',1);
insert into employee
values(12,'0773458642','1991-10-14','30000','kasun presanna',2);
insert into employee
values(13,'0779846859','1992-09-15','28000','kaushan dhammika',3);
insert into employee
values(14,'0773790380','1993-08-16','20000','dedunu gee',4);
insert into employee
values(15,'0771930987','1994-07-16','30000','yasiru rasanja',5);
insert into employee
values(16,'0779996378','1996-02-17','28000','shehana Krishmathi',6);
insert into employee
values(17,'0770983679','1997-03-21','30000','thumashi nawanjana',7);
insert into employee
values(18,'0773459874','1998-01-22','35000','milakshi de silva',8);
insert into employee
values(19,'0772348765','1999-09-17','28000','sahan dewmina',9);
insert into employee
values(20,'077986789','1999-05-01','30000','ayod perera',10);
insert into employee
values(21,'077344654','2000-02-17','35000','duleeka thiyangi',11);
insert into employee
values(22,'077345765','1999-02-19','28000','nimesh dilshan',12);

select * from employee;
```

Query completed with errors.

ASUS-VIVOBOOK15|MySQL (16.0... | ASUS-VIVOBOOK15|kings ... | supermarketdb | 00:00:00 | 0 rows

Offer table

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists databases like supermarketdb, mydb, student, and yourdb. The Solution Explorer on the right shows a project named 'supermarketProject' with a file 'SQLQuery1.sql'. The main pane displays an SQL script for inserting 12 offers into the 'offer' table. The status bar at the bottom indicates the query was executed successfully.

```
INSERT INTO offer
VALUES(1, 'Dairy 10% off');
INSERT INTO offer
VALUES(2, 'Kitchen items 5% off');
INSERT INTO offer
VALUES(3, 'Black friday 14% off');
INSERT INTO offer
VALUES(4, 'comBank cards 5% off');
INSERT INTO offer
VALUES(5, 'bills above 5000 5% off');
INSERT INTO offer
VALUES(6, 'fresh vegetables 25% off');
INSERT INTO offer
VALUES(7, '25% fruits discount');
INSERT INTO offer
VALUES(8, 'Thaipongal day 12% off');
INSERT INTO offer
VALUES(9, 'Freeezer items 16% off');
INSERT INTO offer
VALUES(10, 'bills above 10000 15% off');
INSERT INTO offer
VALUES(11, 'Dialog Finance card 13% off');
INSERT INTO offer
VALUES(12, 'NDB card holders 20% off');
```

Output window showing the results of the query execution:

offer_id	offer_name
1	Dairy 10% off
2	Kitchen items 5% off
3	Black friday 14% off
4	comBank cards 5% off
5	bills above 5000 5% off
6	fresh vegetables 25% off
7	25% fruits discount
8	Thaipongal day 12% off
9	Freeezer items 16% off
10	bills above 10000 15% off
11	Dialog Finance card 13% off
12	NDB card holders 20% off

Payment table

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the supermarketProject database is selected. In the Solution Explorer, there is a Queries folder containing a file named SQLQuery1.sql. The Query Editor pane contains the following SQL script:

```

INSERT INTO payment
VALUES(1,'4790','Bread,peanut butter,olives','2022-01-01',1,11)
INSERT INTO payment
VALUES(2,'3790','peanut butter,flour,olives','2022-05-10',2,12)
INSERT INTO payment
VALUES(3,'2900','Meat,peanut butter,yoghurt','2022-06-11',3,13)
INSERT INTO payment
VALUES(4,'7230','cookies,Aluminium foil,toothpaste','2022-09-17',4,14)
INSERT INTO payment
VALUES(5,'4790','cucumber,peanut butter,olives','2022-12-12',5,15)
INSERT INTO payment
VALUES(6,'5900','smoothies,peanut butter,olives','2022-04-18',6,16)
INSERT INTO payment
VALUES(7,'3560','submarine,olives','2022-04-18',7,17)
INSERT INTO payment
VALUES(8,'1190','flour','2022-05-16',2,18)
INSERT INTO payment
VALUES(9,'2900','Meat,yoghurt','2022-06-17',3,19)
INSERT INTO payment
VALUES(10,'3230','Aluminium foil,toothpaste','2022-09-14',4,20)
INSERT INTO payment
VALUES(11,'1790','cucumber,olives','2022-03-19',5,21)
INSERT INTO payment
VALUES(12,'3900','smoothies,peanut butter','2022-02-26',6,22)

```

The Output pane shows the results of the query execution:

```

1 row affected
1 row affected
1 row affected
1 row affected
100 % 100 % 100 %
Query executed successfully.

```

	payment_id	payment_amount	payment_desc	payment_date	customer_id	emp_ID
1	1	4790.00	Bread,peanut butter,olives	2022-01-01	1	11
2	2	3790.00	peanut butter,flour,olives	2022-05-10	2	12
3	3	2900.00	Meat,peanut butter,yoghurt	2022-06-11	3	13
4	4	7230.00	cookies,Aluminium foil,toothpaste	2022-09-17	4	14
5	5	4790.00	cucumber,peanut butter,olives	2022-12-12	5	15
6	6	5900.00	smoothies,peanut butter,olives	2022-04-18	6	16
7	7	3560.00	submarine,olives	2022-04-18	7	17
8	8	1190.00	flour	2022-05-16	2	18
9	9	2900.00	Meat,yoghurt	2022-06-17	3	19
10	10	3230.00	Aluminium foil,toothpaste	2022-09-14	4	20
11	11	1790.00	cucumber,olives	2022-03-19	5	21
12	12	3900.00	smoothies,peanut butter	2022-02-26	6	22

Section 3 - Triggers

Insert trigger – trigger to display all values in customer table when a record is inserted to the customer table

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a database named 'supermarketdb' is selected. In the center pane, a query window titled 'SQLQuery1.sql' contains the following T-SQL code:

```
create trigger customer_insert
on customer
after insert
as
select * from customer;

insert into customer
values(13,'Dominic famian','0778967999','dominic0@gmail.com');
```

Below the code, the Results tab displays a table with 13 rows of customer data. The columns are: customer_id, customer_name, cu_contact_no, and customer_email. The data includes entries for Domingo Cambridge, Flori Prestand, Klem Burton, Casper Janssen, Marlene Hulkes, Loraine Troker, Ines ave, Gerard Mallebone, Cynthia Twinkie, Kemila Acreman, Berkley Newell, Hamish Dunley, and Dominic famian.

The status bar at the bottom indicates 'Query executed successfully.' and provides other session details.

Update trigger – trigger to update values in product table when a record is updated

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a database named 'supermarketdb' is selected. In the center pane, a query window titled 'SQLQuery1.sql' contains the following T-SQL code:

```
create trigger product_price_update
on product
after update
as
begin
update product set product_price = '840' where product_id = 1
end

update product
set product_price = '80'
where product_id = 2;

select * from product;
```

Below the code, the Results tab displays a table with 12 rows of product data. The columns are: product_id, product_des, product_price, product_brand, supplier_id, and category_id. The data includes entries for submarine, yoghurt, Hair gel, Peanut butter, cucumber, smoothies, toothpaste, Honey sauce, Cake ring, cookies, Olives, and Aluminium foils.

The status bar at the bottom indicates 'Query executed successfully.' and provides other session details.

In the above trigger the price of submarine update from Rs.540.00 to Rs.840.00 when the update statement executed

Delete trigger – trigger to show table data once a record gets deleted

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a database named 'supermarketdb' is selected. In the center pane, a query window displays the creation of a trigger:

```
create trigger delete_offer
on offer
after delete
as
begin
select * from offer;
end
delete from offer where offer_id = 12;
```

Below the query window, the Results tab shows the output of the trigger's select statement:

offer_id	offer_name
1	Dairy 10% off
2	Wheat flour 5% off
3	Black Friday 14% off
4	comBank cards 5% off
5	bills above 5000 5% off
6	fresh vegetables 25% off
7	25% fruits discount
8	Thaipongal day 12% off
9	Freezer items 16% off
10	bills above 10000 15% off
11	Dialog Finance card 13% off

The status bar at the bottom indicates "Query executed successfully." and provides other details like the session ID and time.

Once we delete an offer from offer table the trigger will invoke and display all the details of offer table after deletion.

Delete trigger

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a trigger named 'filter_employee_id' is listed under the 'supermarketdb' database. The script pane contains the following T-SQL code:

```
--create trigger filter_employee_id
on product
after delete
as
begin
select *
from product
where product_id >=5;
end

delete from product where product_id = 11;
```

The Results pane displays the output of the 'select' part of the trigger script, showing product records with IDs 5 through 10. The Output pane at the bottom indicates that the query was executed successfully.

product_id	product_des	product_price	product_brand	supplier_id	category_id
5	cumber	200.00	keels	105	5
6	smoothies	450.00	AB Smoothies	106	6
7	toothpaste	200.00	Signal	107	7
8	Hot sauce	1900.00	MD	108	8
9	Cake icing	900.00	MR Cake	109	9
10	cookies	1200.00	Little Lion	110	10

Once your delete a record from product table the trigger will invoke and display all the product records which as an ID greater than or equal to 5.

Functions

Function to display the average marks of a product (Without parameter type function)

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the database 'supermarketdb' is selected. In the center pane, a query window displays the following T-SQL code:

```
CREATE FUNCTION findProductAvg()
RETURNS float(50)
AS
BEGIN
    DECLARE @avg float(50)
    SELECT @avg = avg(product_price)
    FROM product
    RETURN @avg
END

SELECT [dbo].findProductAvg() AS averagePrice;
```

Below the code, the Results tab shows the output:

averagePrice
1077

A status bar at the bottom indicates "Query executed successfully." and provides connection information: DESKTOP-BJIR016\SQLEXPRESS .. DESKTOP-BJIR016\USER (\$1) supermarketdb 00:00:00 1 rows.

Function to find the employee name when ID is given as input (with return type with parameter)

The screenshot shows the Microsoft SQL Server Management Studio interface. The Query Editor window contains the following T-SQL code:

```
create function findEmployee(@id int)
returns varchar(30)
as
begin
declare @name varchar(30)
select @name = emp_name
from employee
where emp_ID = @id
return @name
end

SELECT [dbo].findEmployee(15) as EmployeeName;
```

The Results pane displays the output of the query:

EmployeeName
Welba Jillet

Below the Results pane, a status bar indicates "Query executed successfully." and provides connection and session information.

Function to find category name when category ID is given as input Function to find the employee name when ID is given as input (with return type with parameter)

The screenshot shows the Microsoft SQL Server Management Studio interface. In the center, a query window displays a T-SQL script for creating a function named `findcategoryName`. The script uses a parameter `@id int` and returns a `varchar(100)` value. It performs a select operation from the `category` table where `category_ID = @id` and returns the result. Below the script, a `SELECT` statement is shown: `SELECT [dbo].findcategoryName(5) as CategoryName;`. To the right of the script, the results pane shows a single row with the value `Vegetable`. At the bottom of the screen, the status bar indicates the query was executed successfully.

```
create function findcategoryName(@id int)
returns varchar(100)
as
begin
declare @name varchar(100)
select @name = category_name
from category
where category_ID = @id
return @name
end

SELECT [dbo].findcategoryName(5) as CategoryName;
```

Query executed successfully.

Function that displays the product name if it is available in both product and category tables by considering the user input product ID

The screenshot shows the Microsoft SQL Server Management Studio interface. In the center, a query window displays a T-SQL script for creating a function named `findprodName`. The script uses an inner join between the `product` and `category` tables to find the product name corresponding to a given product ID. Below the script, a results grid shows one row with the value "Hair gel". At the bottom of the screen, the status bar indicates "Query executed successfully".

```
SELECT [dbo].findCategoryName(5) as CategoryName;
CREATE FUNCTION findprodName(@id int)
RETURNS VARCHAR(100)
AS
BEGIN
    DECLARE @name VARCHAR(100)
    SELECT @name = product.product_des
    FROM product
    INNER JOIN category
    ON product.product_id = @id
    RETURN @name
END
SELECT [dbo].findprodName(3) as productName;
```

Results

productName
Hair gel

Query executed successfully.

Views

View which displays all employees with an employee ID greater than 10

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the database 'supermarketdb' is selected. In the center pane, a query window titled 'SQLQuery1.sql' contains the following SQL code:

```
create view emp_view
as
select *
from employee where emp_ID>10;

select * from emp_view
```

Below the code, the results pane shows a table with 12 rows of employee data, filtered by emp_ID > 10. The columns are: emp_ID, emp_contact_no, emp_dob, emp_salary, emp_name, store_id. The data includes entries for employees like Peggy Adamik, Welbie Jillet, Doug Peaseman, etc.

At the bottom of the results pane, a message says "Query executed successfully."

View which displays employees with a salary greater than Rs.30,000

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the database 'supermarketdb' is selected. In the center pane, a query window titled 'SQLQuery1.sql' contains the following SQL code:

```
create view highest_paid_emp
as
select *
from employee
where emp_salary>30000;

select * from highest_paid_emp;
```

Below the code, the results pane shows a table with 2 rows of employee data, filtered by emp_salary > 30000. The columns are: emp_ID, emp_contact_no, emp_dob, emp_salary, emp_name, store_id. The data includes entries for employees like Lea elui, Dale Chanson, etc.

At the bottom of the results pane, a message says "Query executed successfully."

View which displays products that exceed the price Rs.1500

The screenshot shows the Microsoft SQL Server Management Studio interface. In the center, a query window displays the creation of a view named 'expensive_prod' that selects products with a price greater than 1500. Below the query, the results pane shows three rows of data from the 'product' table.

```
create view expensive_prod
as
select *
from product
where product_price > 1500

select * from expensive_prod;
```

product_id	product_desc	product_price	product_brand	supplier_id	category_id
1	Hair gel	2000.00	AAD	103	3
2	Peanut butter	3000.00	JK Products	104	4
3	Hot sauce	1900.00	MD	108	8

Query executed successfully.

Procedures

Procedure to display all values in employee table

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a file named 'SQLQuery1.sql' is open under 'Solution Explorer'. The code in the query window is:

```
create procedure displayEmp
as
select * from employee
exec displayEmp;
```

In the Results pane, the output of the executed query is displayed as a table:

emp_id	emp_contact_no	emp_dob	emp_salary	emp_name	store_id
1	0778999876	1990-11-13	20000.00	Peggy Adamik	1
2	0773458642	1991-10-14	30000.00	Welbie Jillet	2
3	0779846859	1992-09-15	28000.00	Doug Peaseman	3
4	0773790380	1993-08-16	20000.00	Peggy Adamik	4
5	0771930987	1994-07-17	30000.00	Welbie Jillet	5
6	0779965378	1996-02-17	28000.00	Haley Lasty	6
7	0770836379	1997-03-21	30000.00	Alix Cattroll	7
8	0773459674	1998-01-12	35000.00	Lea elui	8
9	0772348765	1999-09-17	28000.00	Brynn Ingarr	9
10	077989789	1999-05-01	30000.00	Alix Cattroll	10
11	077344654	2000-02-17	35000.00	Davie Chanson	11
12	077345765	1999-02-19	28000.00	Tyler creater	12

Below the results, a message indicates: "Query executed successfully."

Procedure to display the ID and name of user input customer id

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a stored procedure named `findCus` is listed under the `supermarketdb` database. The script pane contains the following code:

```
create procedure findCus
@id int
as
select customer_id,customer_name
from customer
where customer_id = @id
exec findCus @id = 7;
```

The Results pane displays the output of the query, which is a single row:

customer_id	customer_name
7	Ines ave

A status bar at the bottom indicates "Query executed successfully."

Procedure to display product ID, name and category name of products that are in both product and category table.

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a stored procedure named `prod_categ` is listed under the `supermarketdb` database. The script pane contains the following code:

```
create procedure prod_categ
as
select product.product_id,product.product_des,category.category_name
from product
inner join category
on product.product_id = category.category_id
exec prod_categ;
```

The Results pane displays the output of the query, which is 10 rows of data:

product_id	product_des	category_name
1	submarine	short eats
2	yoghurt	Dairy
3	Hair gel	Beauty care
4	Peanut butter	Bread spreads
5	cucumber	Vegetable
6	smoothies	Freezer
7	toothpaste	care products
8	Hot sauce	sauces
9	Cake icing	Baking
10	cookies	Snacks

A status bar at the bottom indicates "Query executed successfully."

Procedure to display all products despite whether it is present in category table or not

The screenshot shows the Microsoft SQL Server Management Studio interface. In the center pane, a query window displays a T-SQL script for creating a stored procedure:

```
exec prod_catege;
go
create procedure all_prods
as
select product.product_des,category.category_name
from product
left join category
on product.product_id = category.category_id
go
exec all_prods;
```

Below the script, the results pane shows a table with 10 rows of data:

product_des	category_name
1 submarine	short eats
2 Dair	Dairy
3 Hair gel	Bath&body care
4 Peanut butter	Bread spreads
5 cucumber	Vegetable
6 smoothies	Freezer
7 toothpaste	care products
8 Hot sauce	sauces
9 Cake icing	Baking
10 cookies	Snacks

The status bar at the bottom indicates "Query executed successfully." and provides details about the execution: DESKTOP-BJIR016\SQLEXPRESS ... DESKTOP-BJIR016\USER (51) supermarketdb 00:00:00 | 10 rows.

Critical Appraisal and Comments on Future Implementation

Critical appraisal

- Everything in this project including the queries is valid and you confirm this by crosschecking with other valid resources and confirm that this project information corroborate with the information of other sources
- Microsoft SQL Server Management Studio does not perform smoothly in low end computers. This can cause various problems such as system lagging and taking more time to fetch the output. The main reason for this is the high RAM usage for SQL server Management Studio

Solutions

- We solved the Microsoft SQL Server Management Studio performance drop by creating the database and implementing the queries in a high-performance pc

Future implementations

The main aim is to develop this system by including more subsystems like cash management systems and to optimize the process in this system to make the tasks more efficient and effective. Once the full optimized project is developed we can implement this system into supermarkets in Sri Lanka. In order to eliminate the need for customers to wait in the checkout line, we can also advance this project by using artificial intelligence to help recognize and process the items that are placed in trolleys.