

IFT 530 Adv Database Management Systems**Project Name : A Virtual Supermarket**

Final Project Presentation

Ankitha Bhavana Gaddam

Hima sree Chalasani

Sai Sandeep Brundavanam

Masters in Information Technology

Arizona State University

Dr. Robert Rucker

April 30, 2023

Table of Contents

Section 1 – Intro and Context.....
Perks to choosing this project.....
Questions that our Database will answer:.....
Section 2 - ORM Diagram.....
Section 3 - Relational Schema.....
Section 4 – SQL Database.....
Description of Entities and Attributes.....
DDL Statements. (Creating Tables).....
DML Statements (Inserting data into tables).....
Additional DDL Query.....
Additional DML Queries.....
Stored Procedure – 1.....
Stored Procedure – 2.....
Trigger - 1.....
Trigger - 2.....
User Defined Function.....
Questions that are answered by our Database:.....
Section 6 - NoSQL.....
Section 6.1.....
Section 6.2 Bucket Information.....
Section 6.3.....
Section 6.4.....
USER INTERFACE.....

Section 7.....
Summary.....
Conclusion.....
References.....

Section 1 – Intro and Context

Over the past few years, there has been a substantial increase in the popularity of online shopping, particularly due to the COVID-19 pandemic which has prompted a rapid shift towards e-commerce. While shopping for items such as clothing and electronics online has been common for some time, there has also been a surge in the popularity of virtual supermarkets. These are online stores that allow customers to conveniently order groceries and have them delivered to their doorstep.

Virtual supermarkets are favored by customers because they offer a wide range of products, easy online ordering, and the ability to avoid crowded stores. They are particularly useful when seeking products that cater to specific culinary tastes. However, running a virtual supermarket requires efficient management of inventory, ordering, and payment processing. To effectively manage operations and provide a seamless customer experience, a robust database system is necessary.

This project aims to design and implement a database system for a virtual supermarket specializing in selling grocery items in the USA market. The database will store information on products, prices, nutritional information, and inventory levels, allowing the supermarket to effectively manage its operations. The system will also include an online ordering and payment processing system, ensuring a secure and straightforward checkout experience for customers. By adopting a comprehensive database system, virtual supermarkets can optimize their operations and enhance customer satisfaction, leading to greater business growth and success.

Perks to choosing this project

1. The process of buying things has been made easier.
2. Making stores and creating listings for products has become easier.
3. Convenient access to products.
4. The cost of marketing and advertising has reduced.
5. Consumers have greater flexibility.
6. There are no limitations on reaching clients.
7. It's easier for customers to compare products and prices.
8. Faster response to consumer and market demands.
9. Several payment methods are available.

Thus, developing a database system for them seems like a logical and feasible approach.

Context

During our initial phase, we focused on the ORM model and established primary foreign keys associated with it.

Subtypes were incorporated for the products, and we aimed to enhance the Purchaser's details. To achieve this, we utilized a Nested Query in Couchbase, merging the Address Information and Purchaser's Favorites List data into the same Purchaser bucket. During the final phase, adjustments were made to the Materials table. The most significant addition to the final project was the user interface (UI).

We developed our UI with a focus on user-friendliness, ensuring that there are distinct views for both Administrators and Purchasers.

In this context, the term "Purchaser" refers to the buyer. Similar to other ecommerce applications, we have included a login page, which users can access through Google login. New customers can create an account to join the Virtual Supermarket Application.

In contrast, the Administrator has their own set of privileges and user page. They can log in and monitor stockroom availability and inventory. Consequently, we have included the same database in both SQL and Couchbase.

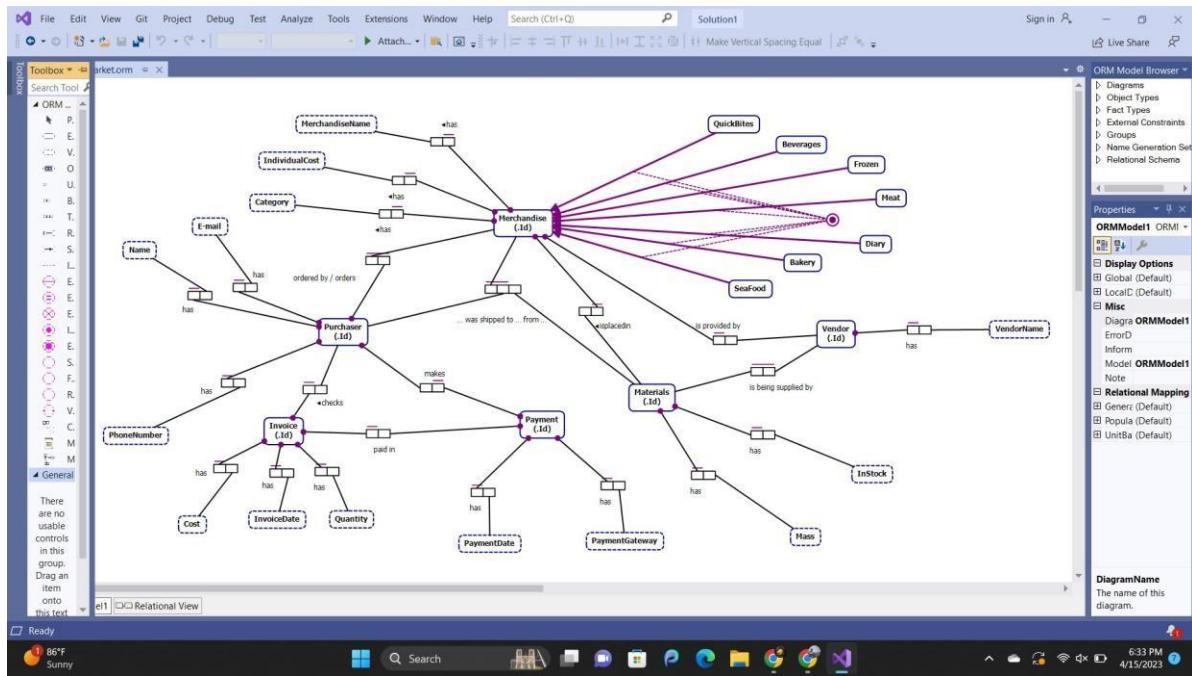
Vendors are responsible for adding stock to the Materials section, and these items are subsequently sold to Purchasers when ordered through our Materials section.

We have showcased our concepts using both SQL and SQL++.

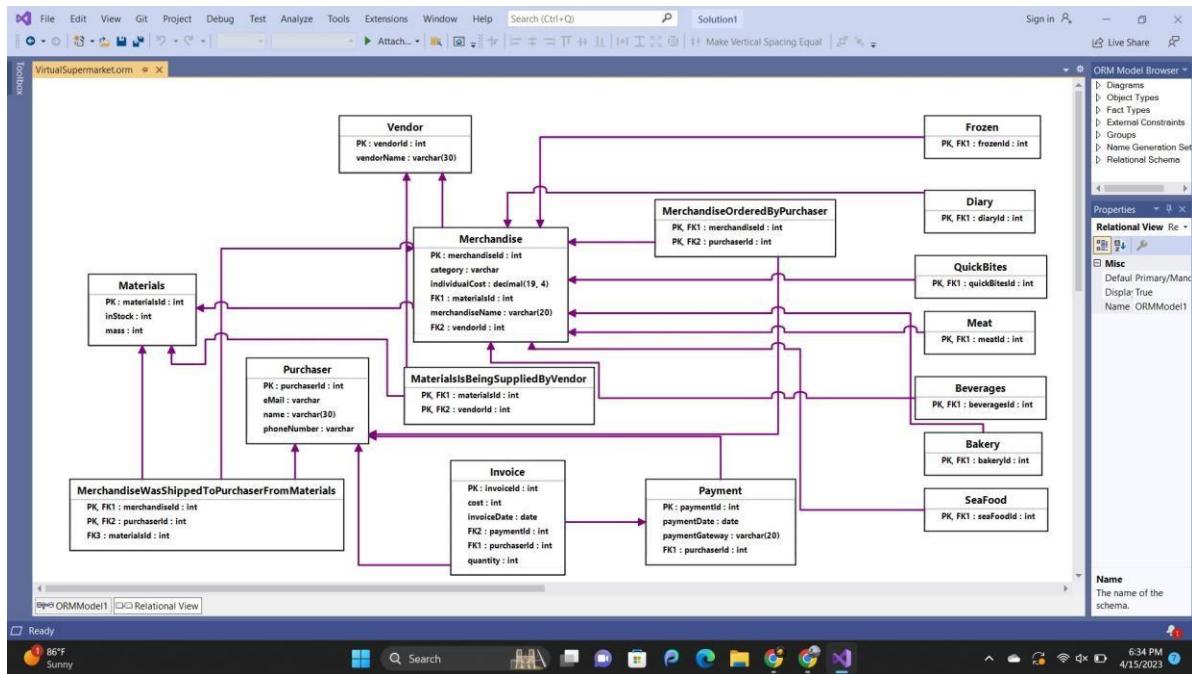
Questions that our Database will answer:

1. Which payment gateway is most frequently utilized by purchasers?
2. Which purchaser has used the greatest variety of payment methods when paying for an order?
3. Which purchaser has placed the largest number of orders in our system?
4. What is the ratio between the number of merchandise and their mass? This enables us to comprehend the constraints of materials space and whether we can meet the demands of supply and demand.
5. To how many departments does each vendor provide supplies?

Section 2 - ORM Diagram



Section 3 - Relational Schema



Section 4 – SQL Database

Description of Entities and Attributes

1. Merchandise:

Merchandise will provide a catalog of products that are identified by a unique MerchandiseId (Primary Key). The Merchandise Entity possesses the subsequent characteristics: MerchandiseId, MerchandiseName, ShelfLife, IndividualCost and Category. It has the following subtypes: QuickBites, Dairy, Beverages, Frozen, Meat, Pantry and Bakery.

2. Invoice:

Invoice entity holds a record of every order placed by purchasers. Every invoice is identified by its unique identifier i.e., InvoiceId. The Entity Invoice has the following attributes: InvoiceId, Quantity, InvoiceDate and Cost.

3. Vendor:

A vendor in the context of a database refers to a supplier who provides goods and replenishes the inventory. Vendors offer a variety of products, and some vendors may supply the same products as others. Each vendor has a unique identifier, known as VendorId, which serves as the primary key, along with the attribute VendorName.

4. Materials:

Materials entity is employed to depict the entirety of products accessible for purchase. The goods are dispensed to customers from this entity. Within this object, we can determine the quantity of stock for every individual product. Each merchandise is differentiated by its specific MaterialsId, and has the attributes of InStock and Mass.

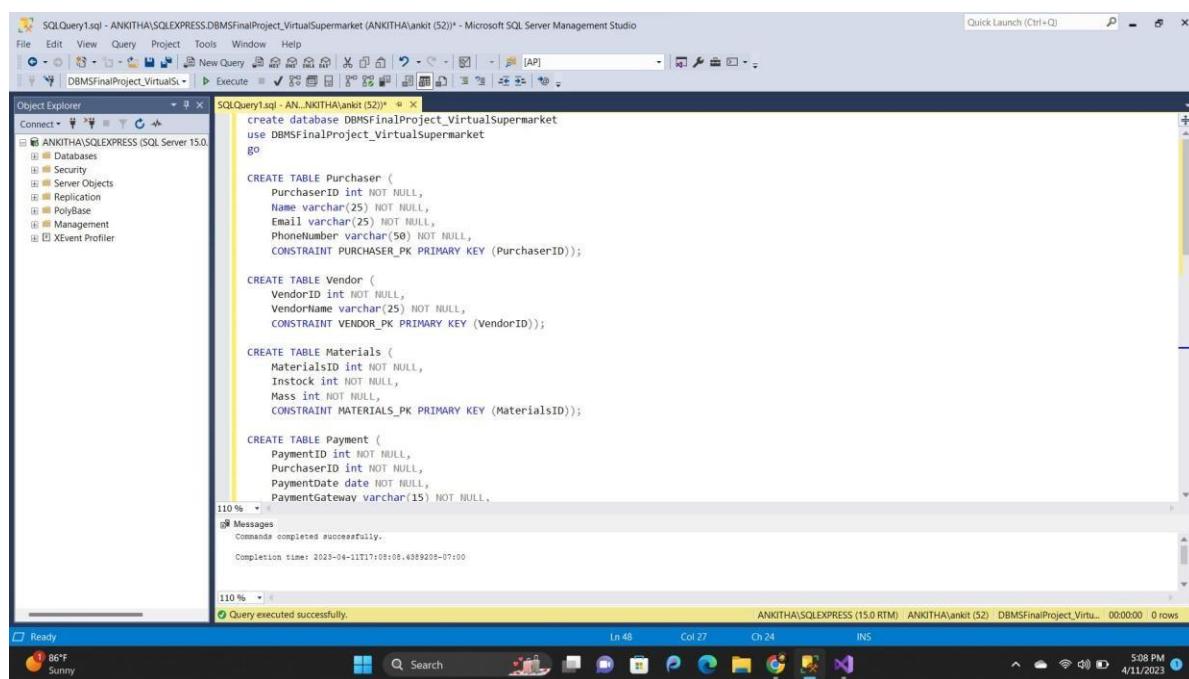
5. Purchaser:

A Purchaser refers to anyone who uses our services to order items from our list of available merchandise. Each Purchaser is given a distinct identifier, known as PurchaserId, which serves as the primary key. The Purchaser's attributes include PurchaserId, Name, E-mail, DateofBirth, and PhoneNumber.

6. Payment:

Payment entity keeps track of the payments done by the purchaser. Each record of payment can be tracked due to its primary key PaymentId, it also keeps track of purchasers and their corresponding payment records. It has the attributes PaymentGateway and PaymentDate.

The attached screenshots serve as evidence of the successful creation of our database.



```

SQLQuery1.sql - ANKITHA\ANKITHA (SQLEXPRESS) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
DBMSFinalProject_VirtualS... | Execute ✓
CREATE DATABASE DBMSFinalProject_VirtualSupermarket
USE DBMSFinalProject_VirtualSupermarket
GO

CREATE TABLE Purchaser (
    PurchaserID int NOT NULL,
    Name varchar(25) NOT NULL,
    Email varchar(25) NOT NULL,
    PhoneNumber varchar(50) NOT NULL,
    CONSTRAINT PURCHASER_PK PRIMARY KEY (PurchaserID));

CREATE TABLE Vendor (
    VendorID int NOT NULL,
    VendorName varchar(25) NOT NULL,
    CONSTRAINT VENDOR_PK PRIMARY KEY (VendorID));

CREATE TABLE Materials (
    MaterialsID int NOT NULL,
    InStock int NOT NULL,
    Mass int NOT NULL,
    CONSTRAINT MATERIALS_PK PRIMARY KEY (MaterialsID));

CREATE TABLE Payment (
    PaymentID int NOT NULL,
    PurchaserID int NOT NULL,
    PaymentDate date NOT NULL,
    PaymentGateway varchar(15) NOT NULL,
    CONSTRAINT PAYMENT_PK PRIMARY KEY (PaymentID));
  
```

Messages
Commands completed successfully.
Completion time: 2023-04-11T17:09:08.4999208-07:00

Query executed successfully.

Ready 86°F Sunny

```

SQLQuery1.sql - ANKITHA\SQLEXPRESS.DBMSFinalProject_VirtualSupermarket (ANKITHA\ankit (52)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query Object Explorer Task List Home Back Forward Stop Refresh [AP]
DBMSFinalProject_VirtualS... Execute ✓ New Query Object Explorer Task List Home Back Forward Stop Refresh [AP]
Object Explorer
Connect Connect...
ANKITHA\SQLEXPRESS (SQL Server 15.0.4000.200)
Databases Security Server Objects Replication Polybase Management XEvent Profiler
SQLQuery1.sql - ANKITHA\SQLEXPRESS (52)* - e X
PurchaserID int NOT NULL,
PaymentDate date NOT NULL,
PaymentGateway varchar(15) NOT NULL,
CONSTRAINT PAYMENT_PK PRIMARY KEY (PaymentID),
FOREIGN KEY (PurchaserID) REFERENCES Purchaser(PurchaserID));

CREATE TABLE Merchandise (
MerchandiseID int NOT NULL,
Category varchar(15) NOT NULL,
MaterialsID int NOT NULL,
MerchandiseName varchar(25),
VendorID int NOT NULL,
IndividualCost money NOT NULL,
CONSTRAINT MERCHANDISE_PK PRIMARY KEY (MerchandiseID),
FOREIGN KEY (MaterialsID) REFERENCES Materials(MaterialsID),
FOREIGN KEY (VendorID) REFERENCES Vendor(VendorID));

CREATE TABLE Invoice (
InvoiceID int NOT NULL,
PurchaserID int NOT NULL,
InvoiceDate date NOT NULL,
cost money NOT NULL,
Quantity int NOT NULL,
PaymentID int NOT NULL,
CONSTRAINT INVOICE_PK PRIMARY KEY (InvoiceID),
FOREIGN KEY (PurchaserID) REFERENCES Purchaser(PurchaserID),
FOREIGN KEY (PaymentID) REFERENCES Payment(PaymentID));

110 % Messages Commands completed successfully.
Completion time: 2023-04-11T17:09:08.4399208-07:00
110 % Query executed successfully.

Ln 48 Col 27 Ch 24 INS
ANKITHA\SQLEXPRESS (15.0 RTM) ANKITHA\ankit (52) DBMSFinalProject_Virtu... 00:00:00 0 rows
Ready 86°F Sunny 5:09 PM 4/11/2023

```

Below are the SQL queries that were used to create tables, define their properties and set their associated data types. They also contain key constraints such as primary keys and foreign keys.

DDL Statements

Creating Tables

We have created six tables in our database, namely: Purchaser, Merchandise, Invoice, Payment, Materials, and Vendor.

Source Code:

```
create database DBMSFinalProject_VirtualSupermarket
```

```
use DBMSFinalProject_VirtualSupermarket
```

```
go
```

```
CREATE TABLE Purchaser (
```

```
PurchaserID int NOT NULL,
```

```
Name varchar(25) NOT NULL,  
Email varchar(25) NOT NULL,  
PhoneNumber varchar(50) NOT NULL,  
CONSTRAINT PURCHASER_PK PRIMARY KEY (PurchaserID));
```

```
CREATE TABLE Vendor (
```

```
    VendorID int NOT NULL,
```

```
    VendorName varchar(25) NOT NULL,
```

```
CONSTRAINT VENDOR_PK PRIMARY KEY (VendorID));
```

```
CREATE TABLE Materials (
```

```
    MaterialsID int NOT NULL,
```

```
    Instock int NOT NULL,
```

```
    Mass int NOT NULL,
```

```
CONSTRAINT MATERIALS_PK PRIMARY KEY (MaterialsID));
```

```
CREATE TABLE Payment (
```

```
    PaymentID int NOT NULL,
```

```
    PurchaserID int NOT NULL,
```

```
    PaymentDate date NOT NULL,
```

```
    PaymentGateway varchar(15) NOT NULL,
```

```
CONSTRAINT PAYMENT_PK PRIMARY KEY (PaymentID),
```

```
FOREIGN KEY (PurchaserID) REFERENCES Purchaser(PurchaserID));
```

```
CREATE TABLE Merchandise (
    MerchandiseID int NOT NULL,
    Category varchar(15) NOT NULL,
    MaterialsID int NOT NULL,
    MerchandiseName varchar(25),
    VendorID int NOT NULL,
    IndividualCost money NOT NULL,
    CONSTRAINT Merchandise_PK PRIMARY KEY (MerchandiseID),
    FOREIGN KEY (MaterialsID) REFERENCES Materials(MaterialsID),
    FOREIGN KEY (VendorID) REFERENCES Vendor(VendorID));
```

```
CREATE TABLE Invoice (
    InvoiceID int NOT NULL,
    PurchaserID int NOT NULL,
    InvoiceDate date NOT NULL,
    Cost money NOT NULL,
    Quantity int NOT NULL,
    PaymentID int NOT NULL,
    CONSTRAINT Invoice_PK PRIMARY KEY (InvoiceID),
    FOREIGN KEY (PurchaserID) REFERENCES Purchaser(PurchaserID),
    FOREIGN KEY (PaymentID) REFERENCES Payment(PaymentID));
```

DML Statements - Inserting data into tables

Adding data to the tables that were created earlier using DML (Data Manipulation Language) statements.

- Inserting values into the Purchaser Table.

Insert into Purchaser values (1,'Ankitha','ankitha@gmail.com','4073886675');

Insert into Purchaser values (2,'Sandeep','sandeep@gmail.com','4073886674');

Insert into Purchaser values (3,'Himasree','himasree@gmail.com','4073886673');

Insert into Purchaser values (4,'Indu','indu@gmail.com','4073886672');

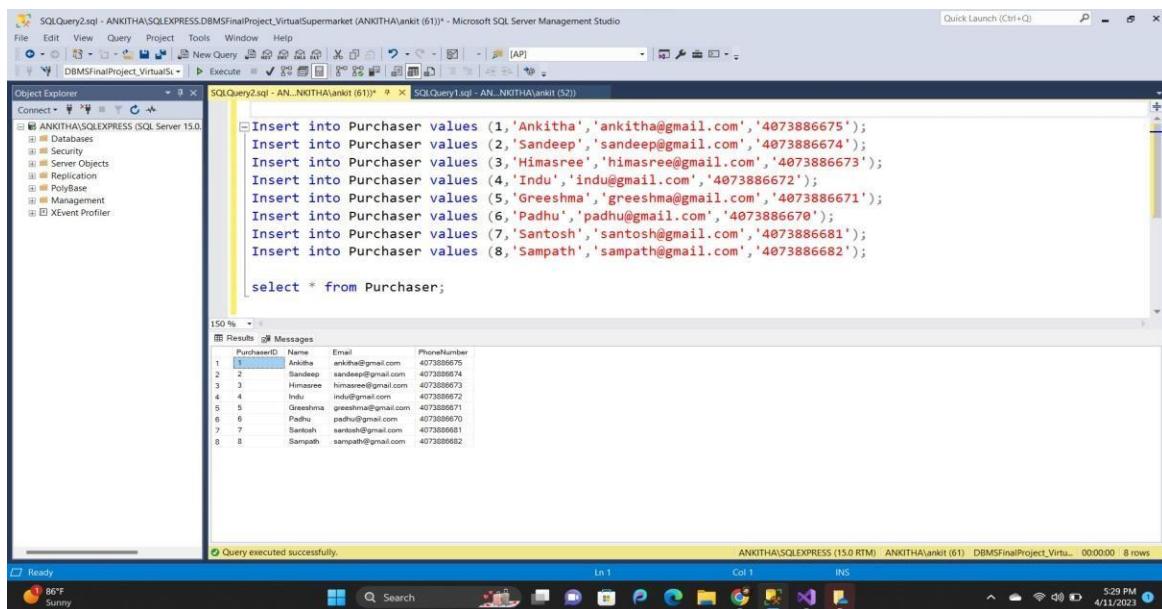
Insert into Purchaser values (5,'Greeshma','greeshma@gmail.com','4073886671');

Insert into Purchaser values (6,'Padhu','padhu@gmail.com','4073886670');

Insert into Purchaser values (7,'Santosh','santosh@gmail.com','4073886681');

Insert into Purchaser values (8,'Sampath','sampath@gmail.com','4073886682');

select * from Purchaser;



```

SQLQuery2.sql - ANKITHA\SQLEXPRESS.DBMSFinalProject_VirtualSupermarket (ANKITHA\ankit (61)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
DBMSFinalProject_VirtualS... Execute
Object Explorer Results Messages
SQLQuery2.sql - AN...NKITHA\ankit (61) SQLQuery1.sql - AN...NKITHA\ankit (52)
Insert into Purchaser values (1,'Ankitha','ankitha@gmail.com','4073886675');
Insert into Purchaser values (2,'Sandeep','sandeep@gmail.com','4073886674');
Insert into Purchaser values (3,'Himasree','himasree@gmail.com','4073886673');
Insert into Purchaser values (4,'Indu','indu@gmail.com','4073886672');
Insert into Purchaser values (5,'Greeshma','greeshma@gmail.com','4073886671');
Insert into Purchaser values (6,'Padhu','padhu@gmail.com','4073886670');
Insert into Purchaser values (7,'Santosh','santosh@gmail.com','4073886681');
Insert into Purchaser values (8,'Sampath','sampath@gmail.com','4073886682');

select * from Purchaser;

```

Results

PurchaserID	Name	Email	PhoneNumber
1	Ankitha	ankitha@gmail.com	4073886675
2	Sandeep	sandeep@gmail.com	4073886674
3	Himasree	himasree@gmail.com	4073886673
4	Indu	indu@gmail.com	4073886672
5	Greeshma	greeshma@gmail.com	4073886671
6	Padhu	padhu@gmail.com	4073886670
7	Santosh	santosh@gmail.com	4073886681
8	Sampath	sampath@gmail.com	4073886682

Query executed successfully.

ANKITHA\SQLEXPRESS (15.0 RTM) ANKITHA\ankit (61) DBMSFinalProject_Virtu... 00:00:00 8 rows

Ready 66°F Sunny

5:29 PM 4/11/2023

- Inserting values into Vendor Table

```
insert into Vendor values (202,' Great Value');
```

```
insert into Vendor values (203,'Kirkland');
```

```
insert into Vendor values (204,'Signature');
```

```
insert into Vendor values (205,'Oscar Mayer');
```

```
insert into Vendor values (105,' Gold Medal');
```

```
insert into Vendor values (106,'Heinz');
```

```
insert into Vendor values (107,'Ben and Jerry');
```

```
select * from Vendor;
```

The screenshot shows the Microsoft SQL Server Management Studio interface. There are three query panes open:

- SQLQuery2.sql - AN_NKITHA\ankit (61)**: Contains the SQL code for inserting data into the Vendor table.
- SQLQuery3.sql - AN_NKITHA\ankit (58)***: Contains the SQL code for selecting all rows from the Vendor table.
- SQLQuery1.sql - AN_NKITHA\ankit (52)**: Shows the results of the SELECT query, displaying 7 rows of data.

The results pane displays the following data:

VendorID	VendorName
1	Gold Medal
2	Heinz
3	Ben and Jerry
4	Great Value
5	Kirkland
6	Signature
7	Oscar Mayer

At the bottom of the screen, the taskbar shows the system status (86°F, Sunny), the date (4/11/2023), and the time (5:37 PM).

- Inserting values into Materials Table

```
insert into Materials values (301,'300','90');
```

```
insert into Materials values (302,'338','300');
```

```
insert into Materials values (303,'65','50');
```

```
insert into Materials values (304,'910','510');
```

```
insert into Materials values (305,'650','420');
```

```

insert into Materials values (306,'40','6');

insert into Materials values (307,'272','55');

insert into Materials values (308,'400','220');

insert into Materials values (309,'50','70');

insert into Materials values (310,'70','65');

insert into Materials values (311,'26','9');

insert into Materials values (312,'900','25');

insert into Materials values (313,'310','210');

insert into Materials values (314,'290','85');

```

select * from Materials;

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the database 'DBMSFinalProject_VirtualSupermarket' is selected. In the center pane, there is an SQL query window containing the following code:

```

insert into Materials values (306,'40','6');
insert into Materials values (307,'272','55');
insert into Materials values (308,'400','220');
insert into Materials values (309,'50','70');
insert into Materials values (310,'70','65');
insert into Materials values (311,'26','9');
insert into Materials values (312,'900','25');
insert into Materials values (313,'310','210');
insert into Materials values (314,'290','85');

select * from Materials;

```

Below the code, the results pane displays the data inserted into the Materials table:

	MaterialID	Length	Mass
1	301	300	80
2	302	328	300
3	303	65	80
4	304	910	510
5	305	650	420
6	306	40	6
7	307	272	55
8	308	400	220
9	309	50	70
10	310	70	85
11	311	26	9
12	312	900	25
13	313	310	210
14	314	290	85

At the bottom of the screen, the taskbar shows the date and time as 4/11/2023 5:42 PM.

- Inserting values into Payment Table

```

insert into Payment values (1300,5,'2023-03-11','Credit');

insert into Payment values (1301,2,'2023-03-07','CashOnDelivery');

insert into Payment values (1302,7,'2023-02-09','Apple Pay');

insert into Payment values (1303,6,'2023-02-02','Apple Pay');

```

```

insert into Payment values (1304,3,' 2023-02-03','Google pay');

insert into Payment values (1305,1,' 2023-01-22','Google pay');

insert into Payment values (1306,4,'2023-01-19','Credit');

insert into Payment values (1307,8,' 2023-03-28','Debit');

insert into Payment values (1308,4,'2023-02-03','Google pay');

insert into Payment values (1309,7,'2023-02-03','Apple pay');

insert into Payment values (1310,1,'2023-04-10','CashOnDelivery');

insert into Payment values (1311,1,'2023-04-15','Apple pay');

select * from Payment;

```

The screenshot shows the Microsoft SQL Server Management Studio interface. There are four tabs open in the query editor:

- SQLQuery2.sql - AN_NKITHA\ankit (61) #
- SQLQuery5.sql - AN_NKITHA\ankit (56)*
- SQLQuery4.sql - AN_NKITHA\ankit (70)
- SQLQuery3.sql - AN_NKITHA\ankit (58)*

The SQLQuery2.sql tab contains the following SQL code:

```

insert into Payment values (1300,5,' 2023-03-11','Credit');
insert into Payment values (1301,2,' 2023-03-07','CashOnDelivery');
insert into Payment values (1302,7,' 2023-02-09','Apple Pay');
insert into Payment values (1303,6,' 2023-02-02','Apple Pay');
insert into Payment values (1304,3,' 2023-02-03','Google pay');
insert into Payment values (1305,1,' 2023-01-22','Google pay');
insert into Payment values (1306,4,' 2023-01-19','Credit');
insert into Payment values (1307,8,' 2023-03-28','Debit');
insert into Payment values (1308,4,' 2023-02-03','Google pay');
insert into Payment values (1309,7,' 2023-02-03','Apple pay');
insert into Payment values (1310,1,' 2023-04-10','CashOnDelivery');
insert into Payment values (1311,1,' 2023-04-15','Apple pay');

select * from Payment;

```

The Results tab displays the output of the last query, showing 12 rows of payment data:

PaymentID	PurchaserID	PaymentDate	PaymentGateway	
1	1300	5	2023-03-11	Credit
2	1301	2	2023-03-07	CashOnDelivery
3	1302	7	2023-02-09	Apple Pay
4	1303	6	2023-02-02	Apple Pay
5	1304	3	2023-02-03	Google pay
6	1305	1	2023-01-22	Google pay
7	1306	4	2023-01-19	Credit
8	1307	8	2023-03-28	Debit
9	1308	4	2023-02-03	Google pay
10	1309	7	2023-02-03	Apple pay
11	1310	1	2023-04-10	CashOnDelivery
12	1311	1	2023-04-15	Apple pay

A message at the bottom of the results pane says "Query executed successfully."

- Inserting values into Invoice Table

```

insert into Invoice values (1600,5,'2023-01-21','350','25',1300);

insert into Invoice values (1601,2,'2023-01-17','800','5',1301);

insert into Invoice values (1602,7,'2023-01-19','240','21',1302);

insert into Invoice values (1603,6,'2023-02-12','470','8',1303);

insert into Invoice values (1604,3,'2023-02-13','660','11',1304);

```

```
insert into Invoice values (1605,1,'2023-03-12','325','23',1305);
```

```
insert into Invoice values (1606,4,'2023-03-29','410','14',1306);
```

```
insert into Invoice values (1607,8,'2023-02-18','381','5',1307);
```

```
insert into Invoice values (1608,4,'2023-02-13','535','25',1308);
```

```
insert into Invoice values (1609,7,'2023-04-03','669','19',1309);
```

```
insert into Invoice values (1610,1,'2023-04-10','2011','5',1310);
```

```
insert into Invoice values (1611,1,'2023-04-15','918','8',1311);
```

```
select * from Invoice;
```

The screenshot shows the Microsoft SQL Server Management Studio interface. There are three tabs open in the query editor:

- SQLQuery2.sql - AN_NKITHA\ankit (61): Contains the SQL code for inserting 12 rows into the Invoice table.
- SQLQuery6.sql - AN_NKITHA\ankit (65)*: Contains the SQL code for inserting 12 rows into the Invoice table.
- SQLQuery4.sql - AN_NKITHA\ankit (70): Contains the SQL code for selecting all columns from the Invoice table.

The results pane shows the output of the SELECT query, displaying 12 rows of data:

InvoiceID	PurchasedID	InvoiceDate	Cost	Quantity	PaymentID
1	1600	5	2023-01-17	800.00	5
2	1601	2	2023-01-17	800.00	5
3	1602	7	2023-01-19	240.00	21
4	1603	8	2023-01-22	60.00	21
5	1604	3	2023-01-22	600.00	11
6	1605	1	2023-01-22	326.00	23
7	1606	4	2023-03-29	410.00	14
8	1607	8	2023-02-18	381.00	5
9	1608	4	2023-02-13	535.00	25
10	1609	7	2023-04-03	669.00	19
11	1610	1	2023-04-10	2011.00	5
12	1611	1	2023-04-15	918.00	8

At the bottom of the screen, the taskbar shows the system tray with icons for battery, signal, and time (5:58 PM, 4/11/2023).

- Inserting values into Merchandise Table

```
insert into Merchandise values (1900,'Dairy',301,'Yogurt',202,' 50');
```

```
insert into Merchandise values (1901,'Dairy',301,'AlmondMilk',203,' 30');
```

```
insert into Merchandise values (1902,'Dairy',301,'Cheese',202,' 55');
```

```
insert into Merchandise values (1903,'QuickBites',303,'Lays',205,'90');
```

```
insert into Merchandise values (1904,'QuickBites',303,'Doritos',205,'85');
```

```
insert into Merchandise values (1905,'Frozen',303,'Mixed Vegetables',203,'70');
```

```

insert into Merchandise values (1906,'Beverages',302,'Sprite',205,'40');

insert into Merchandise values (1907,'Beverages',302,'Dr.Pepper',205,'45');

insert into Merchandise values (1908,'Meat',309,'Chicken Tenders',106,'90');

insert into Merchandise values (1909,'Meat',309,'Lamb',106,'120');

insert into Merchandise values (1910,'Bakery',311,'Cupcakes',204,'30');

insert into Merchandise values (1911,'Bakery',311,'Cheesecake',204,'35');

insert into Merchandise values (1912,'SeaFood',313,'Prawn',107,'35');

insert into Merchandise values (1913,'SeaFood',313,'Fish',107,'80');

select * from Merchandise;

```

The screenshot shows the Microsoft SQL Server Management Studio interface. There are four tabs open in the query editor:

- SQLQuery7.sql - AN_NKITHA\ankit (61) #
- SQLQuery7.sql - AN_NKITHA\ankit (73)*
- SQLQuery6.sql - AN_NKITHA\ankit (65)
- SQLQuery5.sql - AN_NKITHA\ankit (56)

The second tab contains the following SQL code:

```

insert into Merchandise values (1900,'Dairy',301,'Yogurt',202,'50');
insert into Merchandise values (1901,'Dairy',301,'AlmondMilk',203,'30');
insert into Merchandise values (1902,'Dairy',301,'Cheese',202,'55');
insert into Merchandise values (1903,'QuickBites',303,'Lays',205,'90');
insert into Merchandise values (1904,'QuickBites',303,'Doritos',205,'85');
insert into Merchandise values (1905,'Frozen',303,'Mixed Vegetables',203,'70');
insert into Merchandise values (1906,'Beverages',302,'Sprite',205,'40');
insert into Merchandise values (1907,'Beverages',302,'Dr.Pepper',205,'45');
insert into Merchandise values (1908,'Meat',309,'Chicken Tenders',106,'90');
insert into Merchandise values (1909,'Meat',309,'Lamb',106,'120');
insert into Merchandise values (1910,'Bakery',311,'Cupcakes',204,'30');
insert into Merchandise values (1911,'Bakery',311,'Cheesecake',204,'35');
insert into Merchandise values (1912,'SeaFood',313,'Prawn',107,'35');
insert into Merchandise values (1913,'SeaFood',313,'Fish',107,'80');

select * from Merchandise;

```

The results pane shows the following data:

MerchandiseID	Category	MaterialID	MerchandiseName	VendorID	IndividualCost
1	1900	301	Dairy	202	50.00
2	1901	301	Dairy	203	30.00
3	1902	301	Dairy	202	55.00
4	1903	303	QuickBites	205	90.00
5	1904	303	QuickBites	205	85.00
6	1905	303	Frozen	203	70.00
7	1906	302	Beverages	205	40.00
8	1907	302	Beverages	205	45.00
9	1908	309	Meat	106	90.00
10	1909	309	Meat	106	120.00
11	1910	311	Bakery	204	30.00
12	1911	311	Bakery	204	35.00
13	1912	313	SeaFood	107	35.00
14	1913	313	SeaFood	107	80.00

At the bottom of the screen, the taskbar shows the following information:

- Ready
- 86°F Sunny
- Search
- File Explorer
- Task View
- PowerShell
- Edge
- File Explorer
- Google Chrome
- Visual Studio
- File Explorer
- PowerShell
- 6:26 PM 4/11/2023

Additional DDL Query

- **Displaying the PaymentID, Purchaser Name, and payment Date for the transactions made by purchasers who utilized Apple Pay as their chosen payment gateway.**

Query:

```
SELECT Payment.PaymentID, Purchaser.Name, Payment.PaymentDate
```

```
FROM Payment
```

```
INNER JOIN Purchaser ON Payment.PurchaserID = Purchaser.PurchaserID where  
PaymentGateway = 'Apple Pay';
```

Output:

```

SQLQuery1.sql - ANKITHA\SQLEXPRESS.DBMSFinalProject_VirtualSupermarket (ANKITHA\ankit (58)) - Microsoft SQL Server Management Studio
File Edit View Project Tools Window Help
New Query New Item Task List [AP] Execute
[DBMSFinalProject_VirtualS...]
SELECT Payment.PaymentID, Purchaser.Name, Payment.PaymentDate
FROM Payment
INNER JOIN Purchaser ON Payment.PurchaserID = Purchaser.PurchaserID WHERE PaymentGateway = 'Apple Pay';

Results Messages
PaymentID Name PaymentDate
1 1302 Santosh 2023-02-09
2 1303 Padhu 2023-02-02
3 1309 Santosh 2023-02-03
4 1311 Ankitha 2023-04-15

Query executed successfully.
ANKITHA\SQLEXPRESS (15.0 RTM) ANKITHA\ankit (58) DBMSFinalProject_Virtu... 00:00:00 4 rows
Ready
7:01 PM 4/15/2023

```

Additional DML Queries

- Adding the column Sex to Purchaser table

Query:

```
ALTER TABLE Purchaser ADD Sex varchar(10);
```

```
select * from Purchaser;
```

Output:

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the connection is to ANKITHA\SQLEXPRESS. In the main pane, a query window is open with the following SQL code:

```
ALTER TABLE Purchaser ADD Sex varchar(10);
select * from Purchaser;
```

The results pane displays the data from the 'Purchaser' table, which contains 8 rows of data with the newly added 'Sex' column set to NULL.

PurchaserID	Name	Email	PhoneNumber	Sex
1	Ankitha	ankitha@gmail.com	4073886675	NULL
2	Sandeep	sandeep@gmail.com	4073886674	NULL
3	Himasree	himasree@gmail.com	4073886673	NULL
4	Indu	indu@gmail.com	4073886672	NULL
5	Greeshma	greeshma@gmail.com	4073886671	NULL
6	Padhu	padhu@gmail.com	4073886670	NULL
7	Santosh	santosh@gmail.com	4073886681	NULL
8	Sampath	sampath@gmail.com	4073886682	NULL

At the bottom of the results pane, it says "Query executed successfully." and "8 rows".

The screenshot shows the results grid from the previous query execution. The data is identical to the one shown in the previous screenshot, with 8 rows of data in the 'Purchaser' table.

PurchaserID	Name	Email	PhoneNumber	Sex
1	Ankitha	ankitha@gmail.com	4073886675	NULL
2	Sandeep	sandeep@gmail.com	4073886674	NULL
3	Himasree	himasree@gmail.com	4073886673	NULL
4	Indu	indu@gmail.com	4073886672	NULL
5	Greeshma	greeshma@gmail.com	4073886671	NULL
6	Padhu	padhu@gmail.com	4073886670	NULL
7	Santosh	santosh@gmail.com	4073886681	NULL
8	Sampath	sampath@gmail.com	4073886682	NULL

- Deleting the column Sex from Purchaser table

Query:

```
ALTER TABLE Purchaser DROP COLUMN Sex;
```

```
select * from Purchaser;
```

Output:

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

```
ALTER TABLE Purchaser DROP COLUMN Sex;
select * from Purchaser;
```

The results pane shows a table with 8 rows of data:

PurchaserID	Name	Email	PhoneNumber
1	Ankitha	ankitha@gmail.com	4073886675
2	Sandeep	sandeep@gmail.com	4073886674
3	Himasree	himasree@gmail.com	4073886673
4	Indu	indu@gmail.com	4073886672
5	Greeshma	greeshma@gmail.com	4073886671
6	Padhu	padhu@gmail.com	4073886670
7	Santosh	santosh@gmail.com	4073886681
8	Sampath	sampath@gmail.com	4073886682

At the bottom of the results pane, it says "Query executed successfully." and "8 rows".

A zoomed-in view of the results grid from the previous screenshot. The grid has columns: PurchaserID, Name, Email, and PhoneNumber. The data is identical to the previous table.

PurchaserID	Name	Email	PhoneNumber
1	Ankitha	ankitha@gmail.com	4073886675
2	Sandeep	sandeep@gmail.com	4073886674
3	Himasree	himasree@gmail.com	4073886673
4	Indu	indu@gmail.com	4073886672
5	Greeshma	greeshma@gmail.com	4073886671
6	Padhu	padhu@gmail.com	4073886670
7	Santosh	santosh@gmail.com	4073886681
8	Sampath	sampath@gmail.com	4073886682

Stored Procedure – 1

Creating a search query that identifies merchandise with names related to the specified search term. For instance, a search for 'cake' would return all items containing 'cake' in their names, such as Cupcake and Cheesecake."

Query:

```
CREATE PROCEDURE SimilarNameLookup
```

@Similar varchar(25) as begin

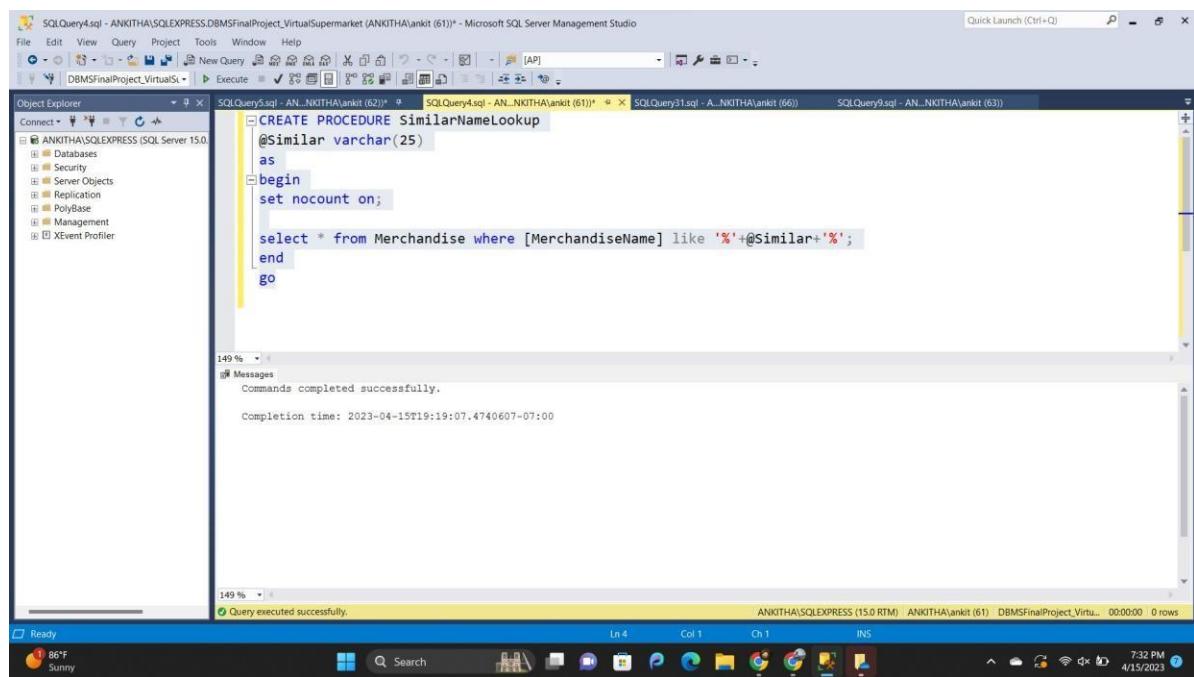
set nocount on;

select * from Merchandise where [MerchandiseName] like '%' + @Similar + '%';

end

go

Output:



The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a database named 'DBMSFinalProject_VirtualSupermarket' is selected. In the main query editor window, a new stored procedure is being created with the following T-SQL code:

```

CREATE PROCEDURE SimilarNameLookup
    @Similar varchar(25)
    as
    begin
        set nocount on;

        select * from Merchandise where [MerchandiseName] like '%' + @Similar + '%';
    end
    go

```

Below the code, the message pane displays:

```

149 % 149 %
Messages Commands completed successfully.

Completion time: 2023-04-15T19:19:07.4740607-07:00

```

The status bar at the bottom right shows the system is 86°F and sunny, the date is 4/15/2023, and the time is 7:32 PM.

Creating a query to execute the stored procedure:

exec SimilarNameLookup 'cake';

```

CREATE PROCEDURE SimilarNameLookup
    @Similar varchar(25)
    as
begin
    set nocount on;

    select * from Merchandise where [MerchandiseName] like '%' + @Similar + '%';
end
go
exec SimilarNameLookup 'cake';

```

Query executed successfully.

	MerchandiseID	Category	MaterialsID	MerchandiseName	VendorID	IndividualCost
1	1910	Bakery	311	Cupcakes	204	30.00
2	1911	Bakery	311	Cheesecake	204	35.00

149 %

Results **Messages**

	MerchandiseID	Category	MaterialsID	MerchandiseName	VendorID	IndividualCost
1	1910	Bakery	311	Cupcakes	204	30.00
2	1911	Bakery	311	Cheesecake	204	35.00

Stored Procedure – 2

Generating a query that displays the PaymentID, PurchaserID, PaymentGateway, and date of payment for transactions occurring within a specified time period. The query includes data validation measures to confirm that the date is entered in the correct format.

Query:

```
CREATE PROCEDURE DateRange
```

```
@RangeStart varchar(50) = NULL, @RangeEnd varchar(50) = NULL AS
```

```
IF @RangeStart IS NULL OR @RangeEnd IS NULL
```

THROW 65000, 'The query requires the start and end values of the range to be specified in order to execute.', 1;

IF NOT (ISDATE(@RangeStart) = 1 AND ISDATE(@RangeEnd) = 1)

THROW 65001, 'Please ensure that the date is entered in the correct format of YYYY-MM-DD.', 1;

IF CAST(@RangeStart AS date) > CAST(@RangeEnd AS date)

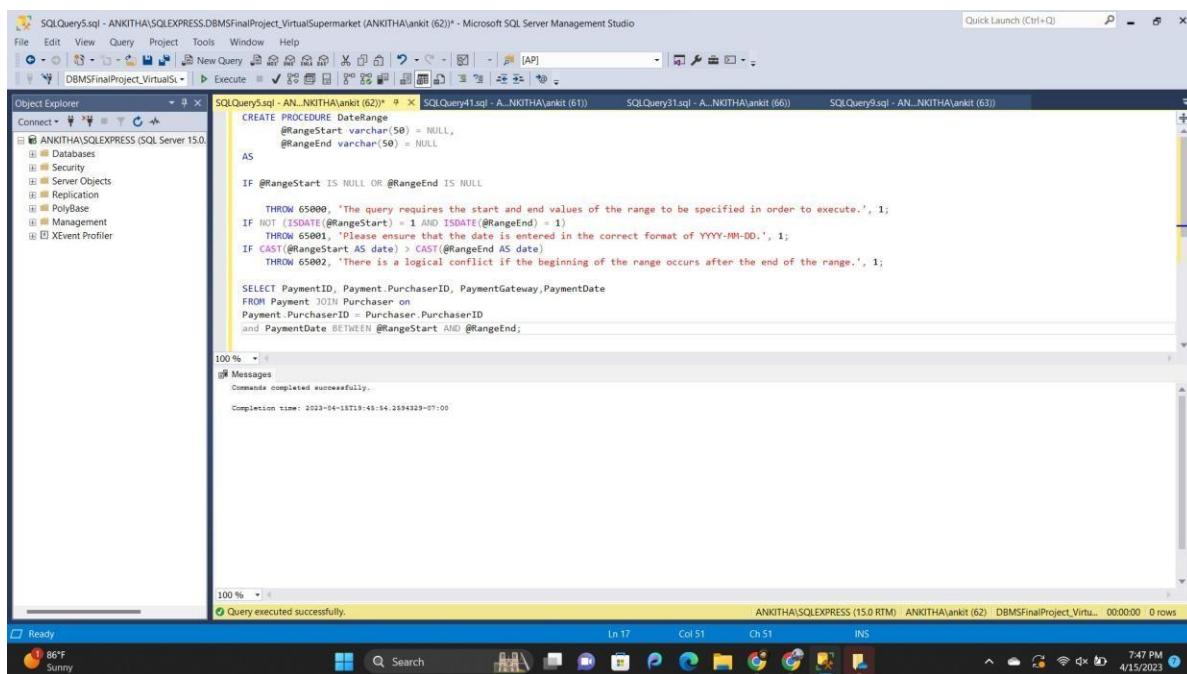
THROW 65002, 'There is a logical conflict if the beginning of the range occurs after the end of the range.', 1;

SELECT PaymentID, Payment.PurchaserID, PaymentGateway, PaymentDate

FROM Payment JOIN Purchaser on

Payment.PurchaserID = Purchaser.PurchaserID

and PaymentDate BETWEEN @RangeStart AND @RangeEnd;



```

CREATE PROCEDURE DateRange
    @RangeStart varchar(50) = NULL,
    @RangeEnd varchar(50) = NULL
AS
BEGIN
    IF (@RangeStart IS NULL OR @RangeEnd IS NULL)
        THROW 65000, 'The query requires the start and end values of the range to be specified in order to execute.', 1;
    IF NOT (ISDATE(@RangeStart) = 1 AND ISDATE(@RangeEnd) = 1)
        THROW 65001, 'Please ensure that the date is entered in the correct format of YYYY-MM-DD.', 1;
    IF CAST(@RangeStart AS date) > CAST(@RangeEnd AS date)
        THROW 65002, 'There is a logical conflict if the beginning of the range occurs after the end of the range.', 1;

    SELECT PaymentID, Payment.PurchaserID, PaymentGateway, PaymentDate
    FROM Payment JOIN Purchaser on
        Payment.PurchaserID = Purchaser.PurchaserID
    and PaymentDate BETWEEN @RangeStart AND @RangeEnd;
END

```

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer pane on the left shows the database structure. The main pane contains the T-SQL code for the stored procedure. The status bar at the bottom indicates the command was completed successfully.

Creating a query to execute the stored procedure:

```
execute DateRange @RangeStart = '2023-03-11', @RangeEnd = '2023-04-15';
```

The screenshot shows the Microsoft SQL Server Management Studio interface. A query window is open with the following T-SQL code:

```

AS
IF @RangeStart IS NULL OR @RangeEnd IS NULL
    THROW 65000, 'The query requires the start and end values of the range to be specified in order to execute.', 1;
IF NOT (ISDATE(@RangeStart) = 1 AND ISDATE(@RangeEnd) = 1)
    THROW 65001, 'Please ensure that the date is entered in the correct format of YYYY-MM-DD.', 1;
IF CAST(@RangeStart AS date) > CAST(@RangeEnd AS date)
    THROW 65002, 'There is a logical conflict if the beginning of the range occurs after the end of the range.', 1;

SELECT PaymentID, Payment.PurchaserID, PaymentGateway, PaymentDate
FROM Payment JOIN Purchaser ON
Payment.PurchaserID = Purchaser.PurchaserID
and PaymentDate BETWEEN @RangeStart AND @RangeEnd;

execute DateRange @RangeStart = '2023-03-11', @RangeEnd = '2023-04-15';

```

The results grid displays the following data:

PaymentID	PurchaserID	PaymentGateway	PaymentDate
1300	5	Credit	2023-03-11
1307	8	Debit	2023-03-28
1310	1	CashOnDelivery	2023-04-10
1311	1	ApplePay	2023-04-15

A message at the bottom of the results pane says "Query executed successfully."

If the end date is specified before the start date, the Stored Procedure will generate the following error.

```
execute DateRange @RangeStart = '2023-03-11', @RangeEnd = '2023-02-15';
```

The screenshot shows the Microsoft SQL Server Management Studio interface. A query window is open with the same T-SQL code as the previous screenshot, but it ends with an invalid date range:

```

execute DateRange @RangeStart = '2023-03-11', @RangeEnd = '2023-02-15';

execute DateRange @RangeStart = '2023-03-11', @RangeEnd = '2023-02-15';

```

The results grid shows the following error message:

```

Msg 65002, Level 16, State 1, Procedure DateRange, Line 12 [Batch Start Line 20]
There is a logical conflict if the beginning of the range occurs after the end of the range.
Completion time: 2023-04-15T19:52:35.8852464+07:00

```

A message at the bottom of the results pane says "Query completed with errors."

Trigger - 1

A Delete Trigger that activates when triggered: Whenever an attempt is made to delete a record from the Invoice table, this trigger will be activated.

Query:

```
CREATE TRIGGER TrigOrder ON Invoice
```

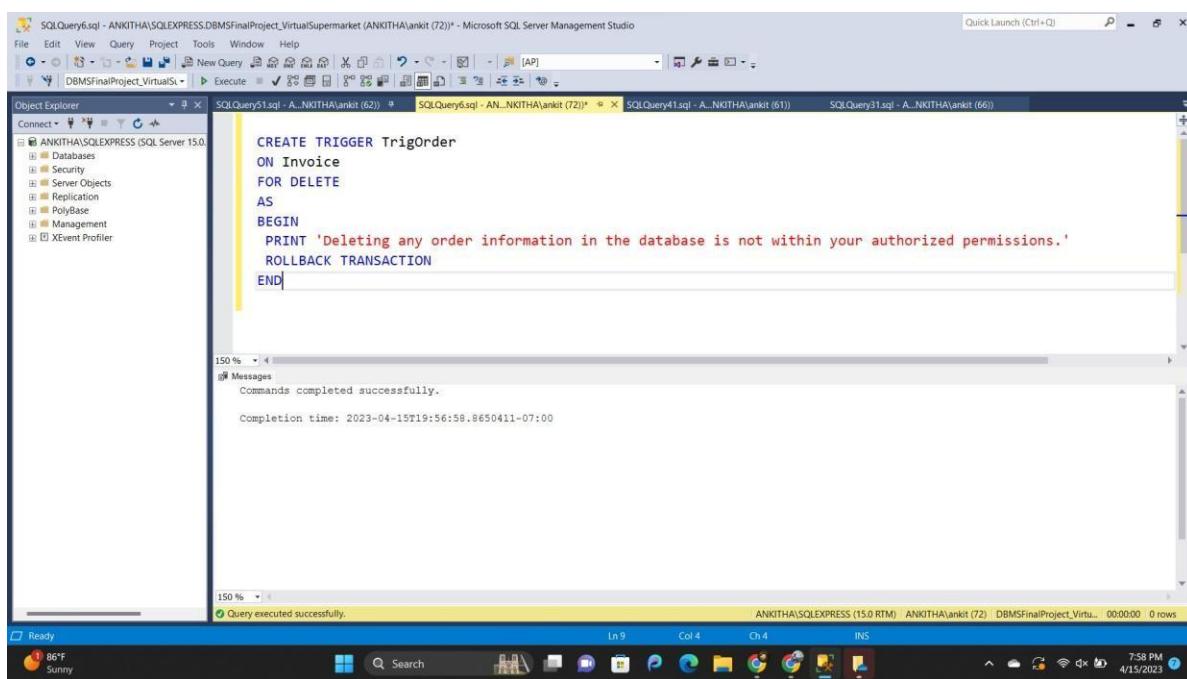
```
FOR DELETE AS BEGIN
```

```
PRINT 'Deleting any order information in the database is not within your authorized
permissions.'
```

```
ROLLBACK TRANSACTION
```

```
END
```

Output:



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

```
CREATE TRIGGER TrigOrder
ON Invoice
FOR DELETE
AS
BEGIN
    PRINT 'Deleting any order information in the database is not within your authorized permissions.'
    ROLLBACK TRANSACTION
END
```

Below the code, the message 'Commands completed successfully.' is displayed in the 'Messages' pane. At the bottom of the screen, the taskbar shows the system tray with a weather icon (86°F, Sunny) and the date/time (4/15/2023, 7:58 PM).

Creating a Query to Execute the Trigger:

Delete Invoice where PurchaserID = 1;

Output:

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

```

FOR DELETE
AS
BEGIN
    PRINT 'Deleting any order information in the database is not within your authorized permissions.'
    ROLLBACK TRANSACTION
END

Delete Invoice where PurchaserID = 1;

```

Below the code, the 'Messages' tab shows the following error output:

```

Deleting any order information in the database is not within your authorized permissions.
Msg 3609, Level 16, State 1, Line 11
The transaction ended in the trigger. The batch has been aborted.

Completion time: 2023-04-15T20:00:17.5250661-07:00

```

At the bottom of the screen, the taskbar shows the date and time as 4/15/2023 8:01 PM.

Trigger - 2

Creating a trigger on the 'invoice' table that activates every time a new invoice is added to the table. The trigger will automatically populate the 'invoiceDate' column with today's date.

Query:

Create trigger DateUpdate on Invoice

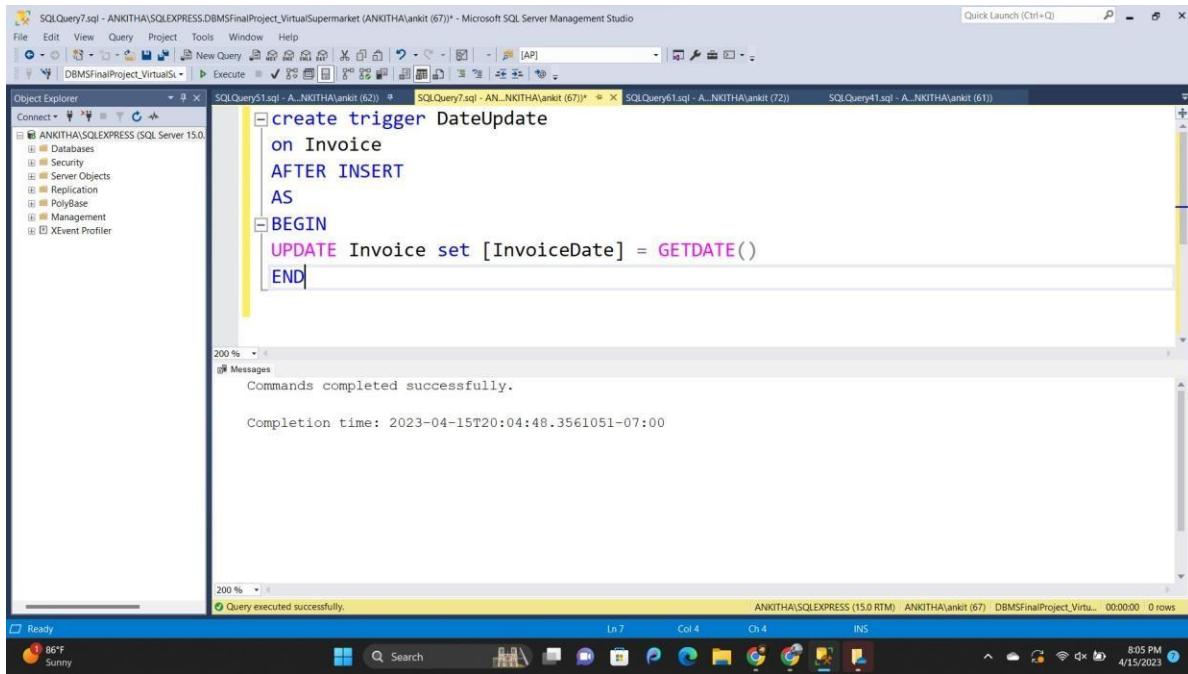
AFTER INSERT

AS BEGIN

UPDATE Invoice set [InvoiceDate] = GETDATE()

END

Output



```

SQLQuery7.sql - ANKITHA\SQLEXPRESS.DBMSFinalProject_VirtualSupermarket (ANKITHA\ankit (67)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
DBMSFinalProject_VirtualS...
New Query Execute SQL Object Explorer
create trigger DateUpdate
on Invoice
AFTER INSERT
AS
BEGIN
    UPDATE Invoice set [InvoiceDate] = GETDATE()
END
  
```

Commands completed successfully.

Completion time: 2023-04-15T20:04:48.3561051-07:00

Query executed successfully.

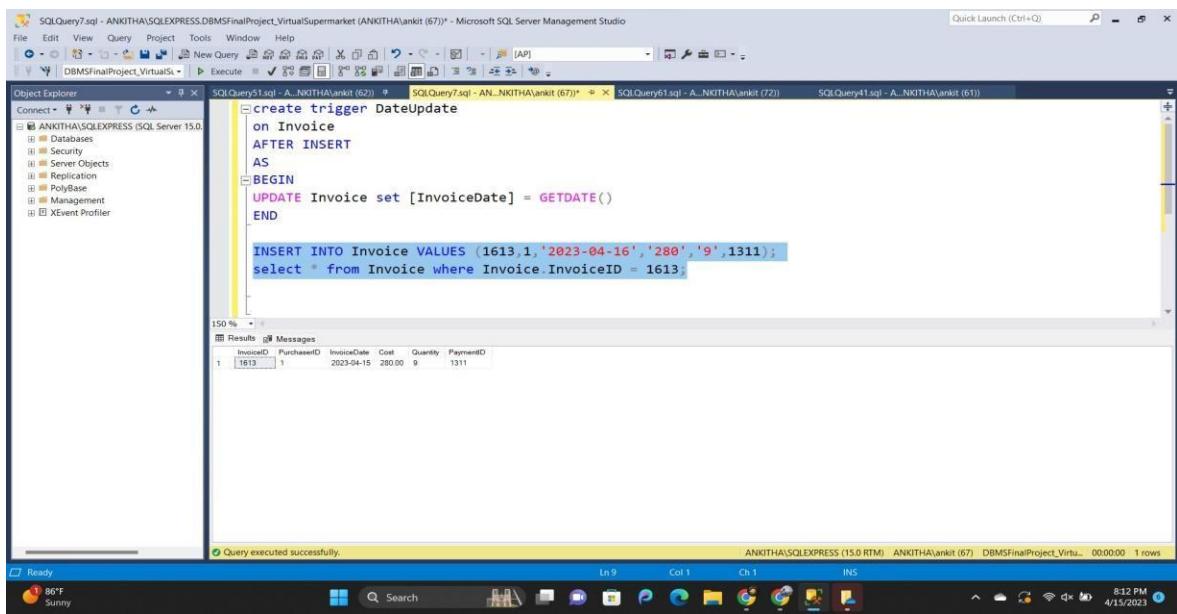
86°F Sunny 8:05 PM 4/15/2023

Creating a Query to Execute the Trigger:

```
INSERT INTO Invoice VALUES (1613,1,'2023-04-16','280','9',1311);
```

```
select * from Invoice where Invoice.InvoiceID = 1613;
```

Output:



```

SQLQuery7.sql - ANKITHA\SQLEXPRESS.DBMSFinalProject_VirtualSupermarket (ANKITHA\ankit (67)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
DBMSFinalProject_VirtualS...
New Query Execute SQL Object Explorer
create trigger DateUpdate
on Invoice
AFTER INSERT
AS
BEGIN
    UPDATE Invoice set [InvoiceDate] = GETDATE()
END

INSERT INTO Invoice VALUES (1613,1,'2023-04-16','280','9',1311);
select * from Invoice where Invoice.InvoiceID = 1613;
  
```

InvoiceID	PurchasedID	InvoiceDate	Cost	Quantity	PaymentID
1613	1	2023-04-16	280.00	9	1311

Query executed successfully.

ANKITHA\SQLEXPRESS (15.0 RTM) ANKITHA\ankit (67) DBMSFinalProject_Virtu... 00:00:00 1 rows

86°F Sunny 8:12 PM 4/15/2023

User Defined Function

Creating a User-Defined Function that accepts a 'Category' parameter, calculates the average unit cost for that specific category, and returns the resulting value.

Query:

```
IF OBJECT_ID (N' dbo.getAverageIndividualCost', N'IF') IS NOT NULL
```

```
DROP FUNCTION dbo.getAverageIndividualCost;
```

```
GO
```

```
CREATE FUNCTION dbo.getAverageIndividualCost (
```

```
    @category VARCHAR(100))
```

```
RETURNS VARCHAR(30) AS
```

```
BEGIN
```

```
    DECLARE @return_value VARCHAR(30);
```

```
    Select @return_value = AVG(IndividualCost)
```

```
    FROM Merchandise
```

```
    WHERE Category = @category
```

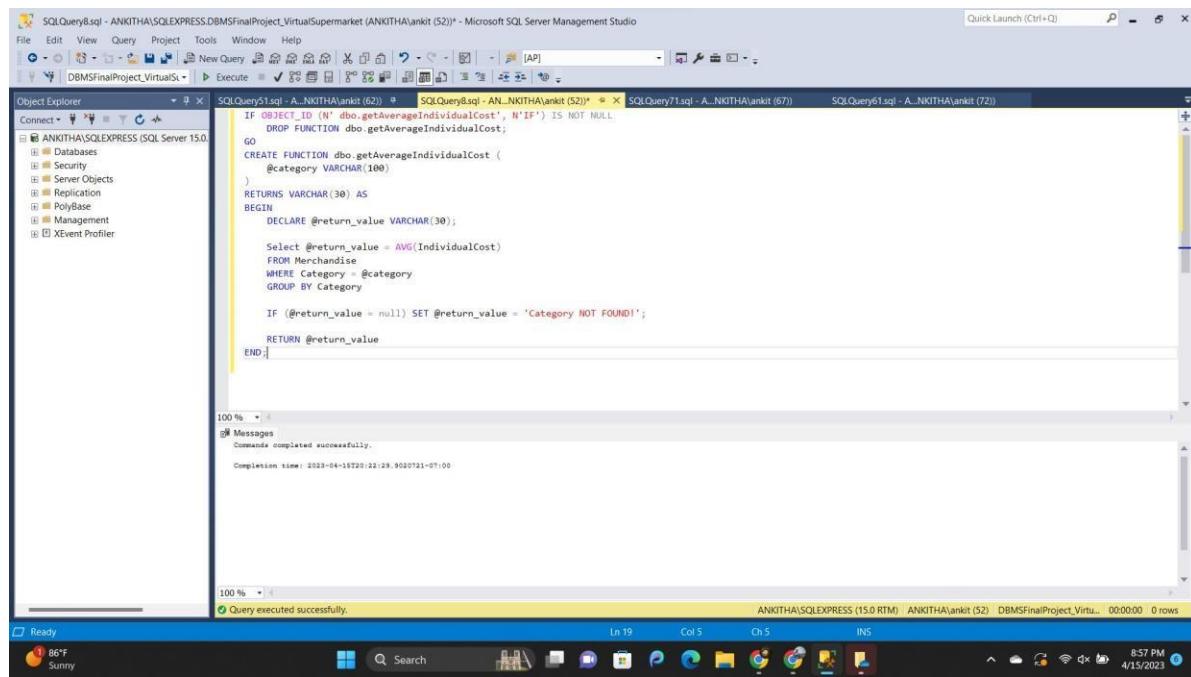
```
    GROUP BY Category
```

```
    IF (@return_value = null) SET @return_value = 'Category NOT FOUND!';
```

```
    RETURN @return_value
```

```
END;
```

Output:



```

SQLQuery8.sql - ANKITHA\SQLEXPRESS.DBMSFinalProject_VirtualSupermarket (ANKITHA\ankit (52)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
DBMSFinalProject_VirtualS... Execute ✓ SQL Server Object Explorer Object Explorer
Object Explorer
Connect Connect to Database...
ANKITHA\SQLEXPRESS (SQL Server 15.0)
Databases Security Server Objects Replication PolyBase Management XEvent Profiler
SQLQuery8.sql : A..._N0THA(ankit (62)) SQLQuery8.sql - AN..._N0THA(ankit (52)) SQLQuery71.sql : A..._N0THA(ankit (67)) SQLQuery61.sql : A..._N0THA(ankit (72))
IF OBJECT_ID ('[dbo].[getAverageIndividualCost]', 'IF') IS NOT NULL
DROP FUNCTION dbo.getAverageIndividualCost;
GO
CREATE FUNCTION dbo.getAverageIndividualCost (
    @category VARCHAR(100)
)
RETURNS VARCHAR(30) AS
BEGIN
    DECLARE @return_value VARCHAR(30);
    Select @return_value = AVG(IndividualCost)
    FROM Merchandise
    WHERE Category = @category
    GROUP BY Category
    IF (@return_value = null) SET @return_value = 'Category NOT FOUND!';
    RETURN @return_value
END;

```

100 %

Messages

Commands completed successfully.

Completion time: 2023-04-15T20:22:29.9020721-07:00

100 %

Query executed successfully.

ANKITHA\SQLEXPRESS (15.0 RTM) ANKITHA\ankit (52) DBMSFinalProject_Virtu... 00:00:00 0 rows

Ready 86°F Sunny

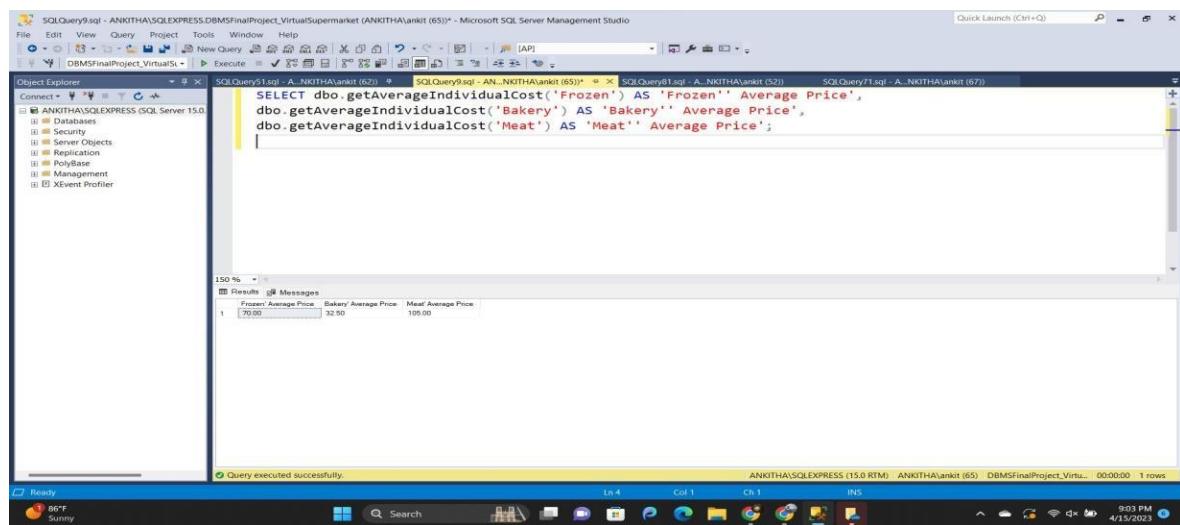
Creating a query to invoke the function:

SELECT dbo.getAverageIndividualCost('Frozen') AS 'Frozen" Average Price',

dbo.getAverageIndividualCost('Bakery') AS 'Bakery" Average Price',

dbo.getAverageIndividualCost('Meat') AS 'Meat" Average Price';

Output:



```

SQLQuery9.sql - ANKITHA\SQLEXPRESS.DBMSFinalProject_VirtualSupermarket (ANKITHA\ankit (65)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
DBMSFinalProject_VirtualS... Execute ✓ SQL Server Object Explorer Object Explorer
Object Explorer
Connect Connect to Database...
ANKITHA\SQLEXPRESS (SQL Server 15.0)
Databases Security Server Objects Replication PolyBase Management XEvent Profiler
SQLQuery9.sql : A..._N0THA(ankit (65)) SQLQuery9.sql - AN..._N0THA(ankit (65)) SQLQuery71.sql : A..._N0THA(ankit (67))
SELECT dbo.getAverageIndividualCost('Frozen') AS 'Frozen'' Average Price',
dbo.getAverageIndividualCost('Bakery') AS 'Bakery'' Average Price',
dbo.getAverageIndividualCost('Meat') AS 'Meat'' Average Price';

```

Frozen Average Price	Bakery Average Price	Meat Average Price
70.00	32.50	105.00

150 %

Results

Messages

Query executed successfully.

ANKITHA\SQLEXPRESS (15.0 RTM) ANKITHA\ankit (65) DBMSFinalProject_Virtu... 00:00:00 1 rows

Ready 86°F Sunny

Queries that our database can respond to:

1. Which payment gateway is most frequently utilized by purchasers?

Query:

```
SELECT TOP 1 PaymentGateway
```

```
AS 'Frequently used Payment Gateway'
```

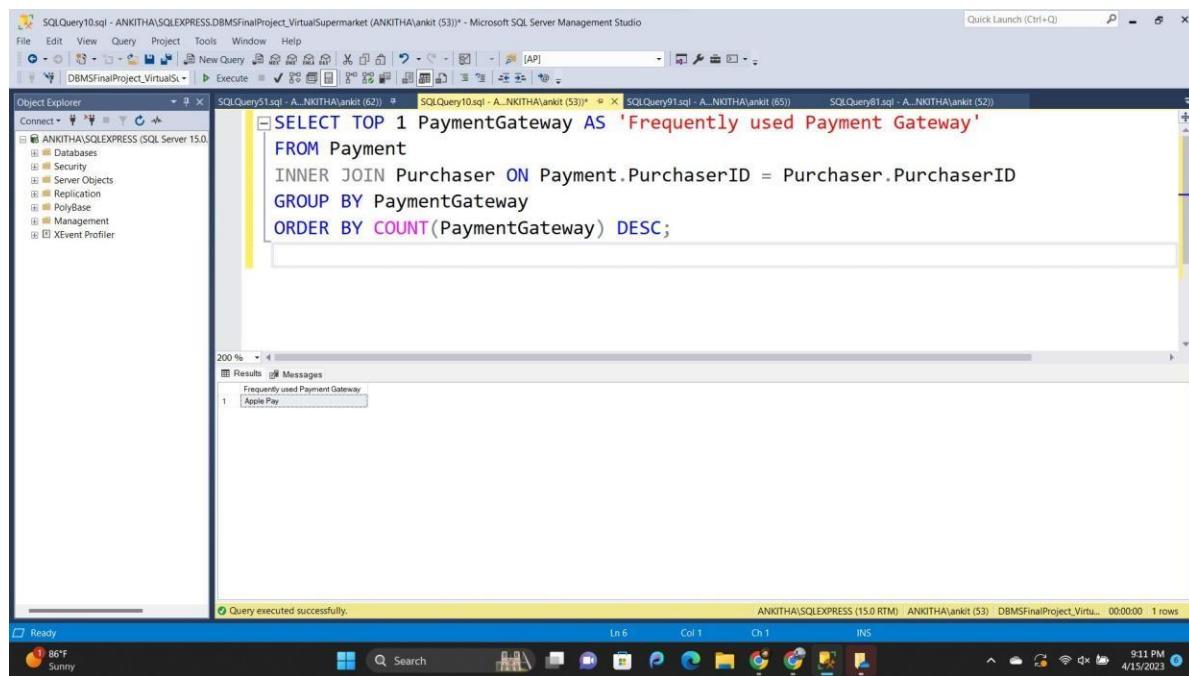
```
FROM Payment
```

```
INNER JOIN Purchaser ON Payment.PurchaserID = Purchaser.PurchaserID
```

```
GROUP BY PaymentGateway
```

```
ORDER BY COUNT(PaymentGateway) DESC;
```

Output:



The screenshot shows the Microsoft SQL Server Management Studio interface. In the center pane, a query window displays the following T-SQL code:

```
SELECT TOP 1 PaymentGateway AS 'Frequently used Payment Gateway'
FROM Payment
INNER JOIN Purchaser ON Payment.PurchaserID = Purchaser.PurchaserID
GROUP BY PaymentGateway
ORDER BY COUNT(PaymentGateway) DESC;
```

Below the code, the results pane shows a single row:

Frequently used Payment Gateway
Apple Pay

A status bar at the bottom indicates "Query executed successfully." and "1 rows".

2. Which purchaser has used the greatest variety of payment gateway when paying for an order?

Query:

```
SELECT TOP 1 Purchaser.Name
```

AS 'Purchaser used the greatest variety of payment gateway when paying for an order'

FROM Purchaser

INNER JOIN Payment ON Payment.PurchaserID = Purchaser.PurchaserID

GROUP BY Purchaser.PurchaserID, Purchaser.Name

ORDER BY COUNT(PaymentGateway) DESC;

Output:

```

SELECT TOP 1 Purchaser.Name AS 'Purchaser used the greatest variety of payment gateway when paying for an order'
FROM Purchaser
INNER JOIN Payment ON Payment.PurchaserID = Purchaser.PurchaserID
GROUP BY Purchaser.PurchaserID, Purchaser.Name
ORDER BY COUNT(PaymentGateway) DESC;

```

Results Messages

Purchaser used the greatest variety of payment gateway when paying for an order
1 Ankit

Query executed successfully.

3. Which purchaser has placed the largest number of orders in our system?

Query:

SELECT TOP 1 Purchaser.Name

AS 'Purchaser who placed the largest number of orders in our system'

FROM Purchaser

INNER JOIN Invoice ON Invoice.PurchaserID = Purchaser.PurchaserID

GROUP BY Purchaser.PurchaserID, Purchaser.Name

ORDER BY COUNT(Invoice.PurchaserID) DESC;

Output:

```

SQLQuery12.sql - ANKITHA\SQLEXPRESS.DBMSFinalProject_VirtualSupermarket (ANKITHA\ankit (69)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
DBMSFinalProject_VirtualS... Execute SQL Object Explorer
Object Explorer
ANKITHA\SQLEXPRESS (SQL Server 15.0)
Databases Security Server Objects Replication Polybase Management XEvent Profiler
SELECT TOP 1 Purchaser.Name AS 'Purchaser who placed the largest number of orders in our system'
FROM Purchaser
INNER JOIN Invoice ON Invoice.PurchaserID = Purchaser.PurchaserID
GROUP BY Purchaser.PurchaserID, Purchaser.Name
ORDER BY COUNT(Invoice.PurchaserID) DESC;
Results Messages
Purchaser who placed the largest number of orders in our system
1 Akhila
Query executed successfully.
ANKITHA\SQLEXPRESS (15.0 RTM) ANKITHA\ankit (69) DBMSFinalProject_Virtu... 00:00:00 1 rows
Ln 6 Col 1 Ch 1 INS
Ready 86°F Sunny 9:29 PM 4/15/2023

```

4. What is the ratio between the number of merchandise and their mass? This enables us to comprehend the constraints of materials space and whether we can meet the demands of supply and demand.

Query:

SELECT Merchandise.MerchandiseName,

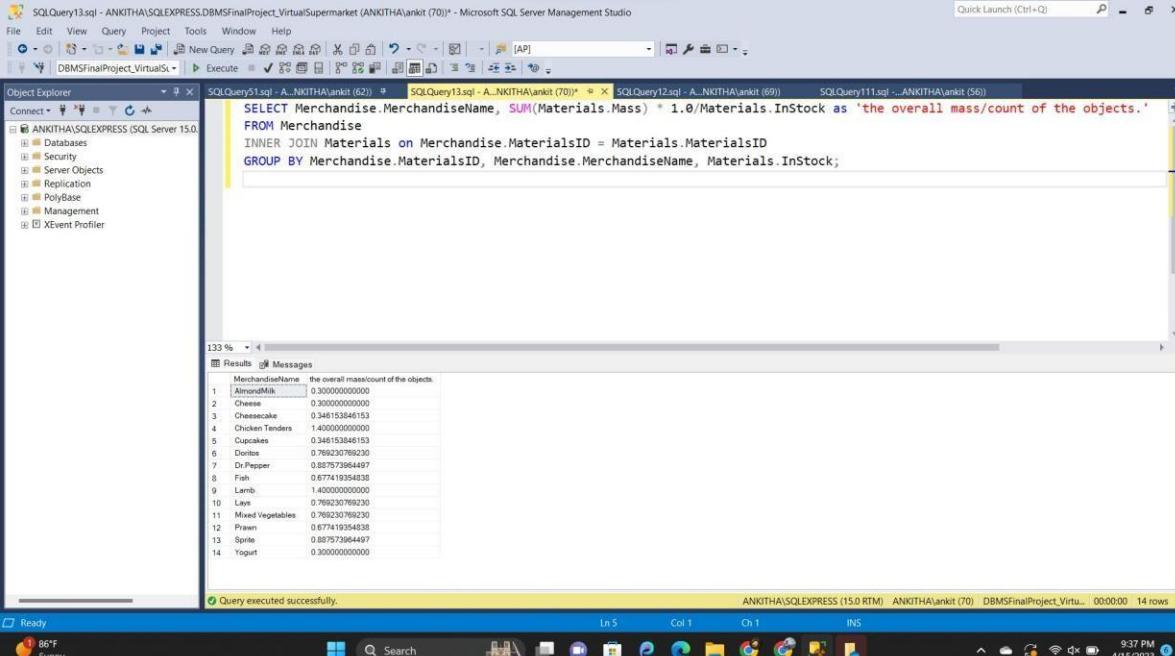
SUM(Materials.Mass) * 1.0/Materials.InStock as 'the overall mass/count of the objects.'

FROM Merchandise

INNER JOIN Materials on Merchandise.MaterialsID = Materials.MaterialsID

GROUP BY Merchandise.MaterialsID, Merchandise.MerchandiseName, Materials.InStock;

Output:



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

```
SELECT Merchandise.MerchandiseName, SUM(Materials.Mass) * 1.0 / Materials.InStock as 'the overall mass/count of the objects.'
FROM Merchandise
INNER JOIN Materials on Merchandise.MaterialsID = Materials.MaterialsID
GROUP BY Merchandise.MaterialsID, Merchandise.MerchandiseName, Materials.InStock;
```

The results grid shows 14 rows of data, each mapping a merchandise name to its calculated value. The columns are 'MerchandiseName' and 'the overall mass/count of the objects.'.

MerchandiseName	the overall mass/count of the objects.
AlmondMilk	0.300000000000000
Cheese	0.300000000000000
Cheesecake	0.346153846153
Chicken Tenders	1.40000000000000
Cupcakes	0.346153846153
Doritos	0.769230769230
Dr.Pepper	0.887573964497
Fish	0.677419354838
Lamb	1.40000000000000
Lays	0.769230769230
Mixed Vegetables	0.769230769230
Prawn	0.677419354838
Sprite	0.887573964497
Yogurt	0.300000000000000

At the bottom of the results grid, a message says "Query executed successfully." The status bar at the bottom right shows the date and time: "4/15/2023 9:37 PM".

Results		Messages
	MerchandiseName	the overall mass/count of the objects.
1	AlmondMilk	0.300000000000000
2	Cheese	0.300000000000000
3	Cheesecake	0.346153846153
4	Chicken Tenders	1.40000000000000
5	Cupcakes	0.346153846153
6	Doritos	0.769230769230
7	Dr.Pepper	0.887573964497
8	Fish	0.677419354838
9	Lamb	1.40000000000000
10	Lays	0.769230769230
11	Mixed Vegetables	0.769230769230
12	Prawn	0.677419354838
13	Sprite	0.887573964497
14	Yogurt	0.300000000000000

5. To how many categories does each vendor provide supplies?

Query:

```
SELECT Vendor.VendorID,
```

```
Vendor.VendorName,
```

COUNT(Merchandise.Category)

AS ' the count of different types of merchandise categories that each vendor is providing to.'

FROM Vendor

INNER JOIN Merchandise on Merchandise.VendorID = Vendor.VendorID

GROUP BY Vendor.VendorID, Vendor.VendorName;

Output:

```

SQLQuery14.sql - ANKITHA\SQLEXPRESS.DBMSFinalProject_VirtualSupermarket (ANKITHA\ankit (71)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
DBMSFinalProject_VirtualS... Execute ✓
SELECT Vendor.VendorID, Vendor.VendorName, COUNT(Merchandise.Category) as 'the count of different types of merchandise categories that each vendor is providing to.'
FROM Vendor
INNER JOIN Merchandise on Merchandise.VendorID = Vendor.VendorID
GROUP BY Vendor.VendorID, Vendor.VendorName;

```

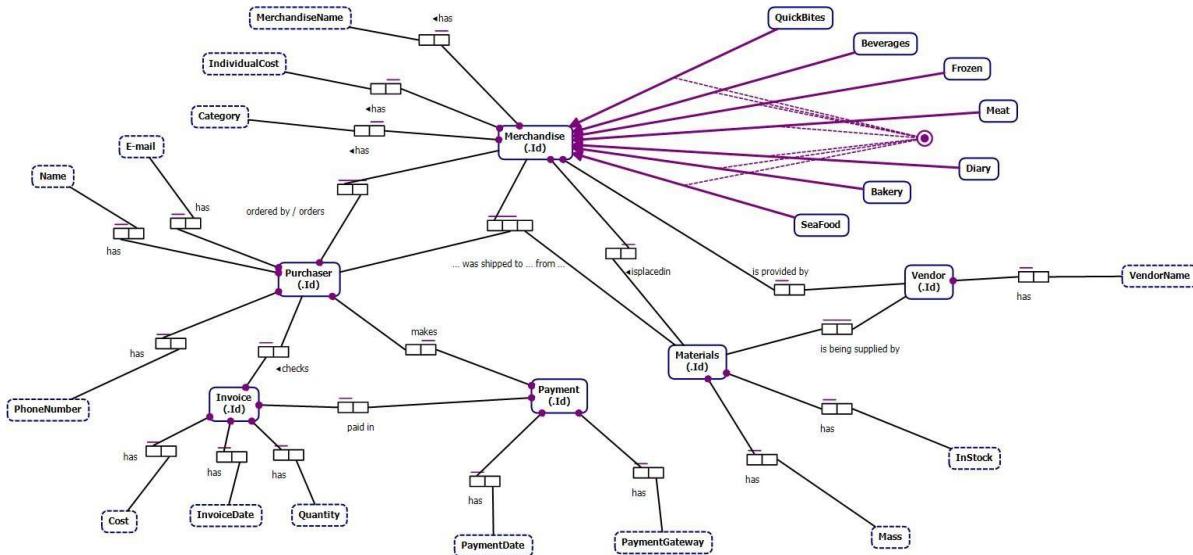
VendorID	VendorName	the count of different types of merchandise categories that each vendor is providing to.
106	Heinz	2
107	Ben and Jerry	2
202	Great Value	2
203	Kirkland	2
204	Signature	2
205	Oscar Mayer	4

Query executed successfully.

	VendorID	VendorName	the count of different types of merchandise categories that each vendor is providing to.
1	106	Heinz	2
2	107	Ben and Jerry	2
3	202	Great Value	2
4	203	Kirkland	2
5	204	Signature	2
6	205	Oscar Mayer	4

Section 6 - NoSQL

Section 6.1



An ORM (Object-Relational Mapping) diagram viewed from a NoSQL standpoint.

There are several subcategories of merchandise available, such as "Meat", "Dairy", "Quick bites", and "Frozen". Each of these subcategories is identified by a unique identifier called MerchandiseId.

The following are the six primary buckets that we have created:

- Purchaser
- Materials
- Merchandise
- Invoice
- Vendor
- Payment

Section 6.2 Bucket Information

name	items	resident	ops/sec	RAM used/quota	disk used	Documents	Scopes & Collections
Invoice	0	100%	0	31.9MB / 500MB	8.06MB	Documents	Scopes & Collections
Materials	0	100%	0	31.9MB / 500MB	8.08MB	Documents	Scopes & Collections
Merchandise	0	100%	0	31.9MB / 500MB	8.06MB	Documents	Scopes & Collections
Payment	0	100%	0	31.9MB / 500MB	0B	Documents	Scopes & Collections
Purchaser	0	100%	0	31.9MB / 500MB	8.08MB	Documents	Scopes & Collections
Vendor	0	100%	0	31.9MB / 500MB	8.06MB	Documents	Scopes & Collections

Adding Documents in Materials Bucket:

insert into Materials (key,value)

values ("301",

{ "MaterialsId": 301,

"InStock": 300,

"Mass": 90 }),

("302", { "MaterialsId": 302,

"InStock": 338,

"Mass": 300 }),

("303", { "MaterialsId": 303,

"InStock": 65,

"Mass": 50 }),
("304", { "MaterialsId": 304,
"InStock": 910,
"Mass": 510 }),
("305",
{ "MaterialsId": 305,
"InStock": 650,
"Mass": 420 }),
("306",
{ "MaterialsId": 306,
"InStock": 40,
"Mass": 6 }),
("307",
{ "MaterialsId": 307,
"InStock": 272,
"Mass": 55 }),
("308",
{ "MaterialsId": 308,
"InStock": 400,
"Mass": 220 }),

("309",
{ "MaterialsId": 309,
"InStock": 50,
"Mass": 70 }),

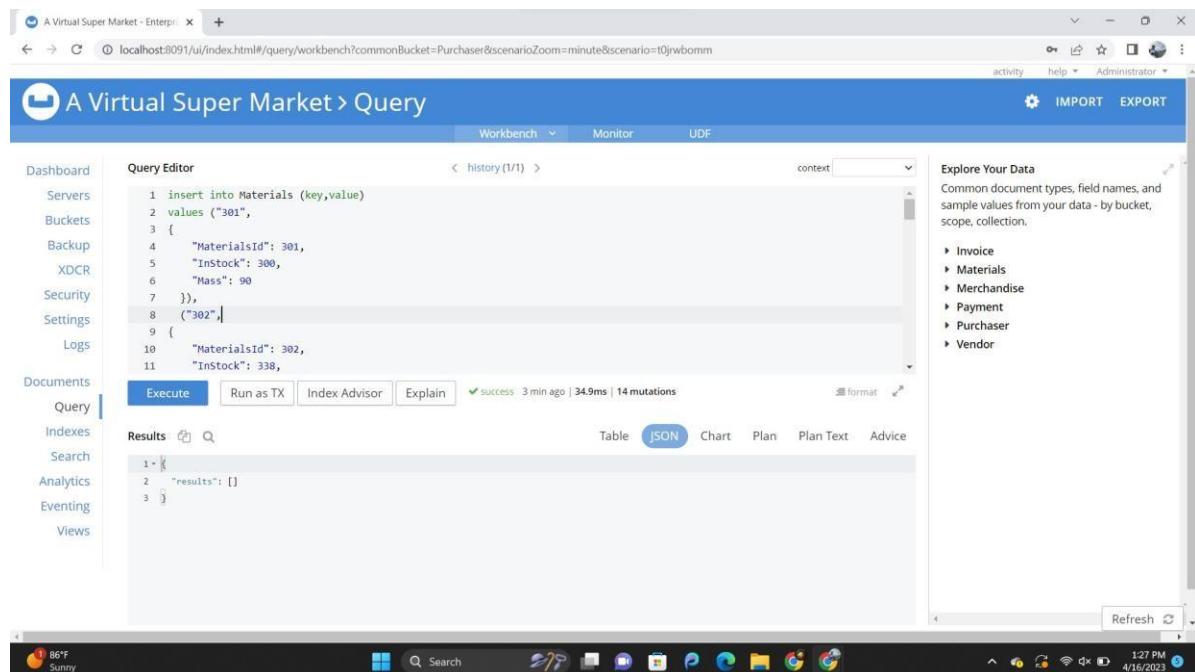
("310",
{ "MaterialsId": 310,
"InStock": 70,
"Mass": 65 }),

("311",
{ "MaterialsId": 311,
"InStock": 26,
"Mass": 9 }),

("312",
{ "MaterialsId": 312,
"InStock": 900,
"Mass": 25 }),

("313",
{ "MaterialsId": 313,
"InStock": 310,
"Mass": 210 }),

```
("314",
{ "MaterialsId": 314,
  "InStock": 290,
  "Mass": 85 })
```



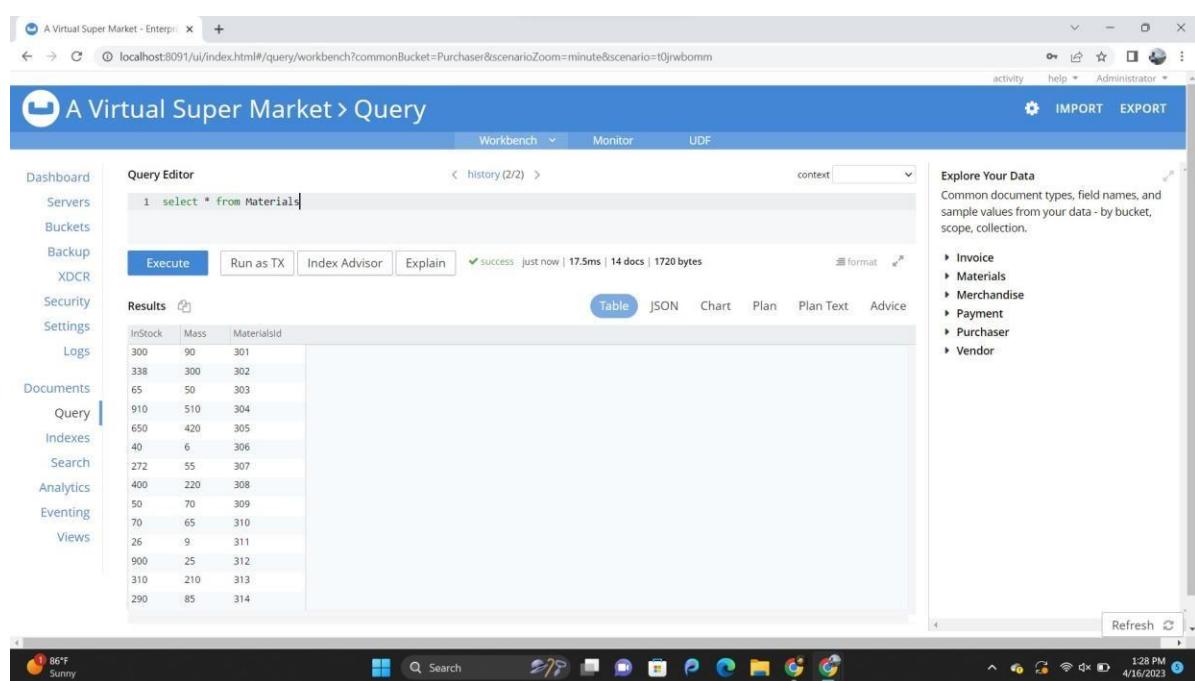
The screenshot shows the 'Query' tab of the A Virtual Super Market - Enterprise interface. In the 'Query Editor', the following code was run:

```
1 insert into Materials (key,value)
2 values ("301",
3 {
4   "MaterialsId": 301,
5   "InStock": 300,
6   "Mass": 90
7 },
8 ("302",
9 {
10  "MaterialsId": 302,
11  "InStock": 338,
```

The results show a single document with the key '302' and the following value:

```
1 {
2   "results": []
3 }
```

The status bar indicates success: 3 min ago | 34.9ms | 14 mutations.



The screenshot shows the 'Query' tab of the A Virtual Super Market - Enterprise interface. In the 'Query Editor', the following code was run:

```
1 select * from Materials
```

The results show a table with the following data:

MaterialsId	Mass	InStock
301	90	300
302	338	300
303	50	65
304	510	910
305	420	650
306	6	40
307	55	272
308	220	400
309	70	50
310	65	70
311	9	26
312	25	900
313	210	310
314	85	290

The status bar indicates success: just now | 17.5ms | 14 docs | 1720 bytes.

Adding Documents in Merchandise Bucket:

```
insert into Merchandise (key,value)
```

```
values ("1900",
```

```
{
```

```
    "MerchandiseID": 1900,
```

```
    "Category": "Dairy",
```

```
    "MaterialsId": 301,
```

```
    "MerchandiseName": "Yogurt",
```

```
    "VendorId": 202 ,
```

```
    "IndividualCost": 50
```

```
}),
```

```
("1901",
```

```
{
```

```
    "MerchandiseID": 1901,
```

```
    "Category": "Dairy",
```

```
    "MaterialsId": 301,
```

```
    "MerchandiseName": "AlmondMilk",
```

```
    "VendorId": 203 ,
```

```
    "IndividualCost": 30
```

```
}),
```

```
{"1902",  
{  
    "MerchandiseID": 1902,  
    "Category": "Dairy",  
    "MaterialsId": 301,  
    "MerchandiseName": "Cheese",  
    "VendorId": 202 ,  
    "IndividualCost": 55  
}),  
  
("1903",  
{  
    "MerchandiseID": 1903,  
    "Category": "QuickBites",  
    "MaterialsId": 303,  
    "MerchandiseName": "Lays",  
    "VendorId": 205 ,  
    "IndividualCost": 90  
}),  
  
("1904",  
{
```

```
"MerchandiseID": 1904,  
"Category": "QuickBites",  
"MaterialsId": 303,  
"MerchandiseName": "Doritos",  
"VendorId": 205 ,  
"IndividualCost": 85  
}),  
("1905",  
{  
"MerchandiseID": 1905,  
"Category": "Frozen",  
"MaterialsId": 303,  
"MerchandiseName": "Mixed Vegetables",  
"VendorId": 203 ,  
"IndividualCost": 70  
}),  
("1906",  
{  
"MerchandiseID": 1906,  
"Category": "Beverages",
```

```
"MaterialsId": 302,  
"MerchandiseName": "Sprite",  
"VendorId": 205 ,  
"IndividualCost": 40  
}),  
("1907",  
{  
"MerchandiseID": 1907,  
"Category": "Beverages",  
"MaterialsId": 302,  
"MerchandiseName": "Dr.Pepper",  
"VendorId": 205 ,  
"IndividualCost": 45  
}),  
("1908",  
{  
"MerchandiseID": 1908,  
"Category": "Meat",  
"MaterialsId": 309,  
"MerchandiseName": "Chicken Tenders",
```

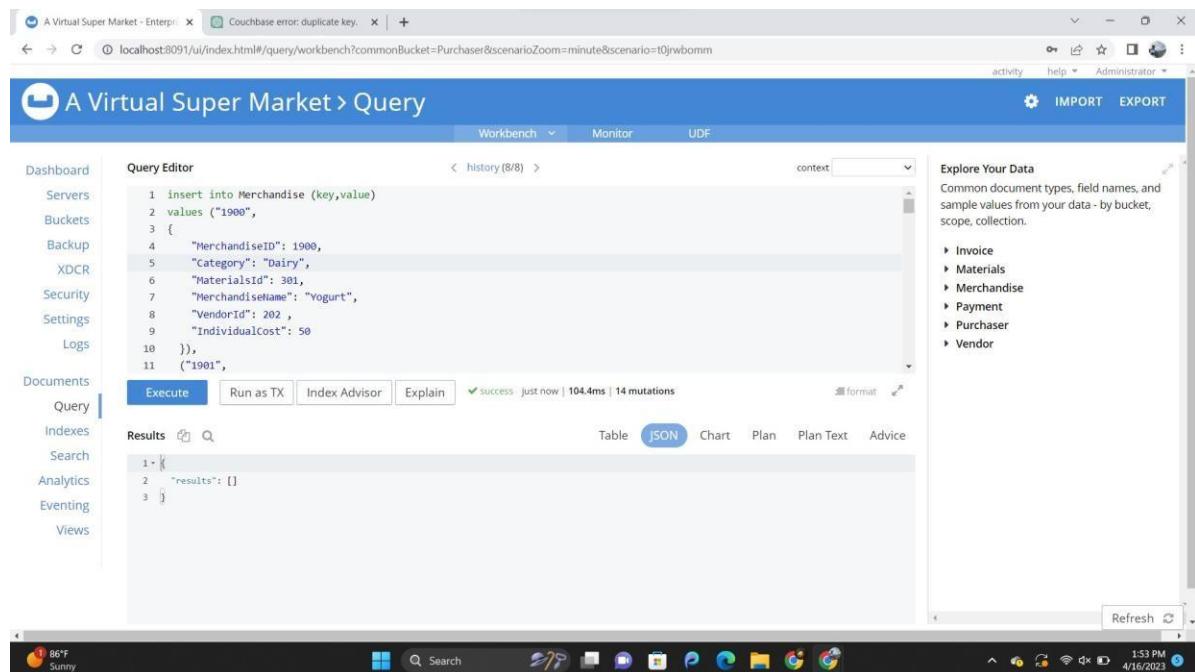
```
"VendorId": 106 ,  
"IndividualCost": 90  
}),  
("1909",  
{  
"MerchandiseID": 1909,  
"Category": "Meat",  
"MaterialsId": 309,  
"MerchandiseName": "Lamb",  
"VendorId": 106 ,  
"IndividualCost": 120  
}),  
("1910",  
{  
"MerchandiseID": 1910,  
"Category": "Bakery",  
"MaterialsId": 311,  
"MerchandiseName": "Cupcakes",  
"VendorId": 204 ,  
"IndividualCost": 30
```

```
}),
("1911",
{
    "MerchandiseID": 1911,
    "Category": "Bakery",
    "MaterialsId": 311,
    "MerchandiseName": "Cheesecake",
    "VendorId": 204 ,
    "IndividualCost": 35
}),
("1912", {
    "MerchandiseID": 1912,
    "Category": "SeaFood",
    "MaterialsId": 313,
    "MerchandiseName": "Prawn",
    "VendorId": 107 ,
    "IndividualCost": 35 },
),
("1913",
{
    "MerchandiseID": 1913,
    "Category": "SeaFood",
```

```

"MaterialsId": 313,
"MerchandiseName": "Fish",
"VendorId": 107 ,
"IndividualCost": 80 }

```



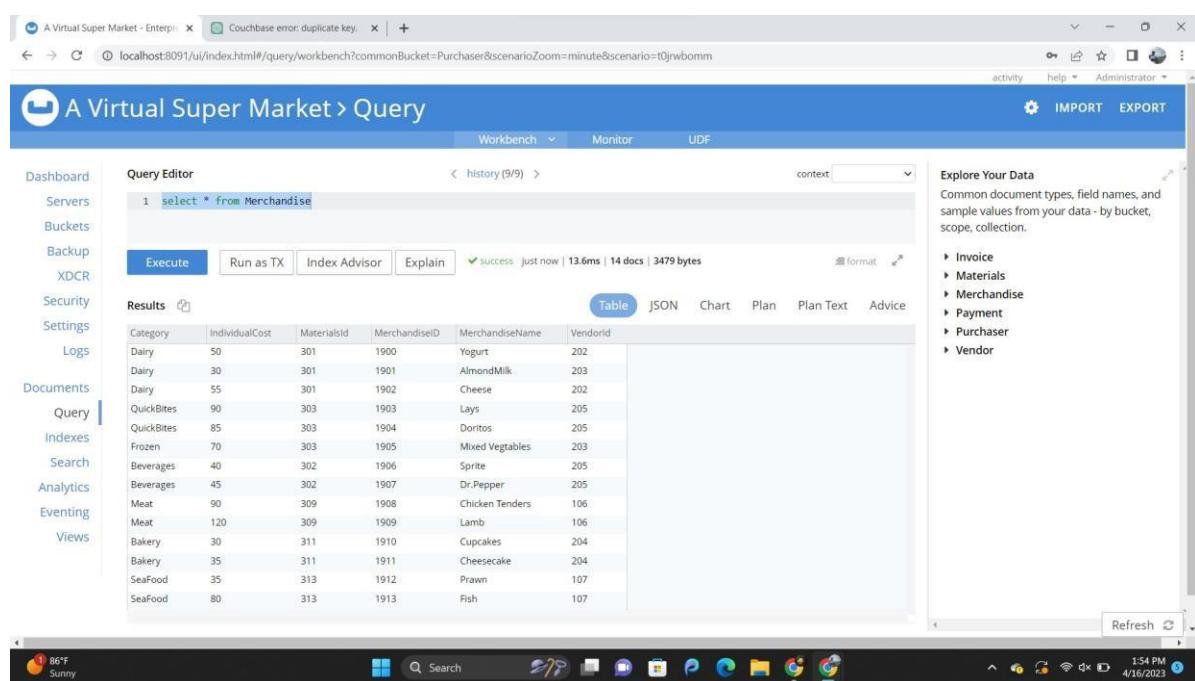
The screenshot shows the 'Query' tab of the A Virtual Super Market - Enterprise interface. In the 'Query Editor', the following code was run:

```

1 insert into Merchandise (key,value)
2 values ("1900",
3 {
4   "MerchandiseID": 1900,
5   "Category": "Dairy",
6   "MaterialsId": 301,
7   "MerchandiseName": "Yogurt",
8   "VendorId": 202 ,
9   "IndividualCost": 50
10 }),
11 ("1901",

```

The results show a single document with the key "1900" and a value object containing the specified fields and their values.



The screenshot shows the 'Query' tab of the A Virtual Super Market - Enterprise interface. In the 'Query Editor', the following code was run:

```

1 select * from Merchandise

```

The results show a table of merchandise items with columns: Category, IndividualCost, MaterialsId, MerchandiseID, MerchandiseName, and Vendorid. The data is as follows:

Category	IndividualCost	MaterialsId	MerchandiseID	MerchandiseName	Vendorid
Dairy	50	301	1900	Yogurt	202
Dairy	30	301	1901	AlmondMilk	203
Dairy	55	301	1902	Cheese	202
QuickBites	90	303	1903	Lays	205
QuickBites	85	303	1904	Doritos	205
Frozen	70	303	1905	Mixed Vegetables	203
Beverages	40	302	1906	Sprite	205
Beverages	45	302	1907	Dr.Pepper	205
Meat	90	309	1908	Chicken Tenders	106
Meat	120	309	1909	Lamb	106
Bakery	30	311	1910	Cupcakes	204
Bakery	35	311	1911	Cheesecake	204
SeaFood	35	313	1912	Prawn	107
SeaFood	80	313	1913	Fish	107

Adding Documents in Invoice Bucket:

```
insert into Invoice (key,value)
```

```
values ("1600",
```

```
{
```

```
    "InvoiceId": 1600,
```

```
    "PurchaserId": 5,
```

```
    "InvoiceDate": "01/21/2022",
```

```
    "Cost": 350,
```

```
    "Quantity": 25,
```

```
    "PaymentID":1300
```

```
}),
```

```
("1601",
```

```
{
```

```
    "InvoiceId": 1601,
```

```
    "PurchaserId": 2,
```

```
    "InvoiceDate": "01/17/2023",
```

```
    "Cost": 800,
```

```
    "Quantity": 5,
```

```
    "PaymentID":1301
```

```
}),
```

```
{"1602",  
{  
    "InvoiceId": 1602,  
    "PurchaserId": 7,  
    "InvoiceDate": "01/19/2023",  
    "Cost": 240,  
    "Quantity": 21,  
    "PaymentID":1302  
}),  
  
("1603",  
{  
    "InvoiceId": 1603,  
    "PurchaserId": 6,  
    "InvoiceDate": "02/12/2023",  
    "Cost": 470,  
    "Quantity": 8,  
    "PaymentID":1303  
}),  
  
("1604",  
{
```

"InvoiceId": 1604,
"PurchaserId": 3,
"InvoiceDate": "02/13/2023",
"Cost": 660,
"Quantity": 11,
"PaymentID":1304
}),
("1605",
{
"InvoiceId": 1605,
"PurchaserId": 1,
"InvoiceDate": "03/12/2023",
"Cost": 325,
"Quantity": 23,
"PaymentID":1305
}),
("1606",
{
"InvoiceId": 1606,
"PurchaserId": 4,

"InvoiceDate": "03/29/2023",

"Cost": 410,

"Quantity": 14,

"PaymentID":1306

}),

("1607",

{

"InvoiceId": 1607,

"PurchaserId": 8,

"InvoiceDate": "02/18/2023",

"Cost": 381,

"Quantity": 5,

"PaymentID":1307

}),

("1608", {

"InvoiceId": 1608,

"PurchaserId": 4,

"InvoiceDate": "02/13/2023",

"Cost": 535,

"Quantity": 25,

"PaymentID":1308 }),
("1609", {
"InvoiceId": 1609,
"PurchaserId": 7,
"InvoiceDate": "03/04/2023",
"Cost": 669,
"Quantity": 19,
"PaymentID":1309
}),
("1610", {
"InvoiceId": 1610,
"PurchaserId": 1,
"InvoiceDate": "04/10/2023",
"Cost": 2011,
"Quantity": 5,
"PaymentID":1310
}),
("1611", {
"InvoiceId": 1611,
"PurchaserId": 1,

"InvoiceDate": "04/10/2023",

"Cost": 918,

"Quantity": 8,

"PaymentID":1311 })

The screenshot shows the 'Query Editor' section of the application. The code entered is:

```

1 insert into Invoice (key,value)
2 values ("1600",
3 {
4   "InvoiceId": 1600,
5   "PurchaserId": 5,
6   "InvoiceDate": "01/21/2022",
7   "Cost": 350,
8   "Quantity": 25,
9   "PaymentId":1300
10 },
11 ("1601",

```

The status bar indicates success: just now | 26.6ms | 12 mutations.

The 'Results' section shows the response:

```

1 [
2   "results": []
3 ]

```

The screenshot shows the 'Query Editor' section with the query:

```

1 select * from Invoice

```

The status bar indicates success: just now | 9.9ms | 12 docs | 2672 bytes.

The 'Results' section shows the table output:

Cost	InvoiceDate	Invoiceld	PaymentID	PurchaserId	Quantity
350	01/21/2022	1600	1300	5	25
800	01/17/2023	1601	1301	2	5
240	01/19/2023	1602	1302	7	21
470	02/12/2023	1603	1303	6	8
660	02/13/2023	1604	1304	3	11
325	03/12/2023	1605	1305	1	23
410	03/29/2023	1606	1306	4	14
381	02/18/2023	1607	1307	8	5
535	02/13/2023	1608	1308	4	25
669	03/04/2023	1609	1309	7	19
2011	04/10/2023	1610	1310	1	5
918	04/10/2023	1611	1311	1	8

Adding Documents in Vendor Bucket:

insert into Vendor (key,value)

values ("202",

{ "VendorId": 202,

 "VendorName": "Great Value" }),

 ("203",

{ "VendorId": 203,

 "VendorName": "Kirkland" }),

 ("204",

{ "VendorId": 204,

 "VendorName": "Signature" }),

 ("205",

{ "VendorId": 205,

 "VendorName": "Oscar Mayer" }),

 ("105",

{ "VendorId": 105,

 "VendorName": "Gold Medal" }),

 ("106",

{ "VendorId": 106,

 "VendorName": "Heinz" }),

```
("107",
{ "VendorId": 107,
  "VendorName": "Ben and Jerry"
})
```

The screenshot shows the 'Query Editor' section of the application. The code entered is:

```
1 insert into Vendor (key,value)
2 values ("202",
3 {
4   "VendorId": 202,
5   "VendorName": "Great Value"
6 }),
7 ("203",
8 {
9   "VendorId": 203,
10  "VendorName": "Kirkland"
11 })
```

The results are displayed in JSON format:

```
1 [
2   "results": []
3 ]
```

The status bar indicates success: 1 min ago | 11.3ms | 7 mutations.

The screenshot shows the 'Query Editor' section with the query:

```
1 select * from Vendor
```

The results are displayed in Table format:

VendorId	VendorName
105	Gold Medal
106	Heinz
107	Ben and Jerry
202	Great Value
203	Kirkland
204	Signature
205	Oscar Mayer

The status bar indicates success: just now | 11.4ms | 7 docs | 732 bytes.

Adding Documents in Payment Bucket:

```
insert into Payment (key,value)
```

```
values ("1300",
```

```
{
```

```
    "PaymentID": 1300,
```

```
    "PurchaserId": 5,
```

```
    "PaymentDate": "03/11/2023",
```

```
    "PaymentGateway": "Credit"
```

```
}),
```

```
("1301",
```

```
{
```

```
    "PaymentID": 1301,
```

```
    "PurchaserId": 2,
```

```
    "PaymentDate": "03/07/2023",
```

```
    "PaymentGateway": "CashOnDelivery"
```

```
}),
```

```
("1302",
```

```
{
```

```
    "PaymentID": 1302,
```

```
    "PurchaserId": 7,
```

"PaymentDate": "02/09/2023",

"PaymentGateway": "Apple pay"

}),

("1303",

{

"PaymentID": 1303,

"PurchaserId": 6,

"PaymentDate": "02/02/2023",

"PaymentGateway": "Apple pay"

}),

("1304",

{

"PaymentID": 1304,

"PurchaserId": 3,

"PaymentDate": "02/03/2023",

"PaymentGateway": "Google pay"

}),

("1305",

{

"PaymentID": 1305,

```
"PurchaserId": 1,  
"PaymentDate": "01/22/2023",  
"PaymentGateway": "Google pay"  
}),  
("1306",  
{  
"PaymentID": 1306,  
"PurchaserId": 4,  
"PaymentDate": "01/19/2023",  
"PaymentGateway": "Credit"  
}),  
("1307",  
{  
"PaymentID": 1307,  
"PurchaserId": 8,  
"PaymentDate": "03/28/2023",  
"PaymentGateway": "Debit"  
}),  
("1308",  
{
```

"PaymentID": 1308,
"PurchaserId": 4,
"PaymentDate": "02/03/2023",
"PaymentGateway": "Google pay"
}),
("1309",
{
"PaymentID": 1309,
"PurchaserId": 7,
"PaymentDate": "02/03/2023",
"PaymentGateway": "Apple pay"
}),
("1310",
{
"PaymentID": 1310,
"PurchaserId": 1,
"PaymentDate": "04/10/2023",
"PaymentGateway": "CashOnDelivery"
}),
("1311",

```
{
  "PaymentID": 1311,
  "PurchaserId": 1,
  "PaymentDate": "04/15/2023",
  "PaymentGateway": "Apple pay"
})
```

The screenshot shows the 'Query Editor' tab in the 'Workbench' section of the interface. The code entered is:

```
1 insert into Payment (key,value)
2 values ("1300",
3 {
4   "PaymentID": 1300,
5   "PurchaserId": 5,
6   "PaymentDate": "03/11/2023",
7   "PaymentGateway": "Credit"
8 },
9 ("1301",
10 {
11   "PaymentID": 1301,
```

The status bar indicates success: just now | 52ms | 12 mutations.

The 'Results' section shows the query results in JSON format:

```
1 [
2   "results": []
3 ]
```

The screenshot shows the 'Query Editor' tab in the 'Workbench' section of the interface. The code entered is:

```
1 select * from Payment
```

The status bar indicates success: just now | 15.6ms | 12 docs | 2187 bytes.

The 'Results' section shows the query results in Table format:

PaymentDate	PaymentGateway	PaymentID	PurchaserId
03/11/2023	Credit	1300	5
03/07/2023	CashOnDelivery	1301	2
02/09/2023	Apple pay	1302	7
02/02/2023	Apple pay	1303	6
02/03/2023	Google pay	1304	3
01/22/2023	Google pay	1305	1
01/19/2023	Credit	1306	4
03/28/2023	Debit	1307	8
02/03/2023	Google pay	1308	4
02/03/2023	Apple pay	1309	7
04/10/2023	CashOnDelivery	1310	1
04/15/2023	Apple pay	1311	1

Section 6.3

One of the conditions was to demonstrate the nesting of two buckets within one. In this case, the Purchaser bucket contains both the Address information and Favourites List, which are combined or nested together.

Customer Bucket Document:

[

```
{
  "Purchaser": {
    "PurchaserId": 1,
    "Email": "ankitha@gmail.com",
    "Name": "Ankitha",
    "Phone": 4073886675,
    "address": {
      "city": "Tempe",
      "street": "1-125, Pyramid road",
      "zipcode": "58321"
    },
    "Favourites List": "Fish"
  },
  "Purchaser": {
    "PurchaserId": 2,
    "Email": "sandeep@gmail.com",
    "Name": "Sandeep"
  }
}
```

"Phone": 4073886674,

"address": {

 "city": "Tempe",

 "street": "142/2, Yendada",

 "zipcode": "53681" },

 "Favourites List": "Mixed Vegetables" } },

{ "Purchaser": {

 "PurchaserId": 3,

 "Email": "himasree@gmail.com",

 "Name": "Himasree",

 "Phone": 4073886673,

 "address": {

 "city": "North Carolina",

 "street": "431, Gudiwada",

 "zipcode": "52361" },

 "Favourites List": "Cupcakes" } },

 { "Purchaser": {

 "PurchaserId": 4,

 "Email": "indu@gmail.com",

 "Name": "Indu",

"Phone": 4073886672,

"address": {

 "city": "Seattle",

 "street": "751, Gitam",

 "zipcode": "52561" },

 "Favourites List": "Milk" } },

{ "Purchaser": {

 "PurchaserId": 5,

 "Email": "greeshma@gmail.com",

 "Name": "Greeshma",

 "Phone": 4073886671,

 "address": {

 "city": "Texas",

 "street": "754, University st",

 "zipcode": "68731" },

 "Favourites List": "Cheese" } },

 "Purchaser": {

 "PurchaserId": 6,

 "Email": "padhu@gmail.com",

 "Name": "Padhu",

"Phone": 4073886670,

"address": { "city": "Boston",

 "street": "871, yendada",

 "zipcode": "56214" },

 "Favourites List": "Lays" } },

 {"Purchaser": {

 "PurchaserId": 7,

 >Email": "santosh@gmail.com",

 >Name": "Santosh",

 "Phone": 4073886681,

 "address": { "city": "North Carolina",

 "street": "5621, Anna st",

 "zipcode": "45631" },

 "Favourites List": "Almond Milk" } },

 {"Purchaser": {

 "PurchaserId": 8,

 >Email": "sampath@gmail.com",

 >Name": "Sampath",

 "Phone": 4073886682,

 "address": {

```

"city": "Illinoi",
"street": "82, Samay st",
"zipcode": "87321" },
"Favourites List": "Cheesecake" } }]

```

A Virtual Super Market - Enterprise

localhost:8091/ui/index.html#/docs/editor?commonBucket=Purchaser&scenarioZoom=minute&scenario=t0jrbomm&bucket=Purchaser

A Virtual Super Market > Documents

ADD DOCUMENT

Dashboard Servers Buckets Backup XDCR Security Settings Logs

Documents

Query Indexes Search Analytics Eventing Views

Keyspace bucket.scope.collection: Purchaser _default _default Limit: 10 Offset: 0 Document ID: optional... show range N1QL WHERE: optional... Retrieve Docs

1 Results for select meta().id from 'Purchaser'.'_default'._default data order by meta().id limit 10 offset 0

enable field editing < prev batch | next batch >

id	
1	[{"Purchaser": {"PurchaserId": 1, "Email": "ankitha@gmail.com", "Name": "Ankitha", "Phone": "4073886675", "address": {"city": "Tempe", "street": "1-125, Pyramid road", "zipcode": "58321"}, "wishlist": "Fish"}]

A Virtual Super Market - Enterprise

localhost:8091/ui/index.html#/query/workbench?commonBucket=Purchaser&scenarioZoom=minute&scenario=t0jrbomm

A Virtual Super Market > Query

Workbench Monitor UDF

Dashboard Servers Buckets Backup XDCR Security Settings Logs

Documents

Query

Indexes Search Analytics Eventing Views

Query Editor history (18/18) context:

1 select * from Purchaser

Execute Run as TX Index Advisor Explain success 10 min ago | 1.7ms | 1 docs | 3901 bytes

Results Table JSON Chart Plan Plan Text Advice

Purchaser

Email	Name	Phone	PurchaserId	address	wishlist
ankitha@gmail.com	Ankitha	4073886675	1	Tempe 1-125, Pyramid road	Fish
sandeep@gmail.com	Sandeep	4073886674	2	Tempe 142/2, Yendada	Mixed Vegetables
himasree@gmail.com	Himasree	4073886673	3	North Carolina 431, Gudiwada	Cupcakes
indu@gmail.com	Indu	4073886672	4	Seattle 751, Gitam	Milk
greshma@gmail.com	Greshma	4073886671	5	Texas 754, University st	Cheese

Explore Your Data

Common document types, field names, and sample values from your data - by bucket, scope, collection.

- Invoice
- Materials
- Merchandise
- Payment
- Purchaser
- Vendor

Section 6.4

In order to perform Analytics using Couchbase, we initially generated the Datasets.

CREATE DATASET ON Merchandise;

CREATE DATASET ON Vendor;

CREATE DATASET ON Purchaser;

CREATE DATASET ON Materials;

CREATE DATASET ON Invoice;

CREATE DATASET ON Payment;

Query 1: Showing the MaterialsId for which the stock quantity is 100 or lower.

FROM Materials AS i

WHERE i.InStock <= 100

SELECT i.MaterialsId AS MaterialsId

The screenshot shows the 'A Virtual Super Market - Enterprise' application running on a Windows desktop. The main window is titled 'A Virtual Super Market > Query'. The left sidebar contains navigation links for Dashboard, Servers, Buckets, Backup, XDCR, Security, Settings, Logs, Documents, Query (which is selected), Indexes, Search, Analytics, Eventing, and Views. The central area has a 'Query Editor' tab with the following SQL code:

```

1 FROM Materials AS i
2 WHERE i.InStock <= 100
3 SELECT i.MaterialsId AS MaterialsId
  
```

The 'Execute' button is highlighted. Below the editor, the 'Results' section displays a table with one column labeled 'MaterialsId' containing the values 303, 306, 309, 310, and 311. To the right of the results, there's an 'Explore Your Data' sidebar with a tree view showing categories: Invoice, Materials, Merchandise, Payment, Purchaser, and Vendor. The bottom status bar shows the date and time as 4/16/2023 4:05 PM.

Query 2: Which customers used Apple Pay to pay for their orders?

FROM Payment AS p

WHERE p.PaymentGateway = 'Credit'

SELECT p.PurchaserId AS PurchaserId

The screenshot shows the 'A Virtual Super Market > Query' interface. The left sidebar has 'Query' selected. The main area has the following code in the 'Query Editor':

```

1 FROM Payment AS p
2 WHERE p.PaymentGateway = 'Credit'
3 SELECT p.PurchaserId AS PurchaserId

```

Below the code, there are buttons for 'Execute', 'Run as TX', 'Index Advisor', and 'Explain'. The status bar indicates 'success: just now | 4.3ms | 2 docs | 64 bytes'. The results are displayed in JSON format, showing two documents:

```

1 [
2   {
3     "PurchaserId": 5
4   },
5   {
6     "PurchaserId": 4
7   }
8 ]

```

To the right, there's a sidebar titled 'Explore Your Data' with a list of document types: Invoice, Materials, Merchandise, Payment, Purchaser, Vendor, and Merchandise.

Query 3: Displaying Vendor Id who are selling Kirkland merchandise.

FROM Vendor AS v

WHERE v.VendorName = "Kirkland"

SELECT v.VendorId AS VendorId

A Virtual Super Market > Query

Query Editor

```
1 FROM Vendor AS v
2 WHERE v.VendorName = "Kirkland"
3 SELECT v.VendorId AS VendorId
```

Execute Run as TX Index Advisor Explain ✓ success 1 min ago | 4.7ms | 1 docs | 31 bytes

Results

VendorId
203

Table JSON Chart Plan Plan Text Advice

Explore Your Data

Common document types, field names, and sample values from your data - by bucket, scope, collection.

- Invoice
- Materials
- Merchandise
- Payment
- Purchaser
- Vendor

Query 4: Displaying the invoice generated by the purchaser with ID=7.

FROM Invoice AS i

WHERE i.PurchaserId = 7

SELECT i.InvoiceId, i.Quantity, PurchaserId

A Virtual Super Market > Query

Query Editor

```
1 FROM Invoice AS i
2 WHERE i.PurchaserId = 7
3 SELECT i.InvoiceId, i.Quantity, PurchaserId
```

Execute Run as TX Index Advisor Explain ✓ success just now | 4.1ms | 2 docs | 166 bytes

Results

InvoiceId	PurchaserId	Quantity
1602	7	21
1609	7	19

Table JSON Chart Plan Plan Text Advice

Explore Your Data

Common document types, field names, and sample values from your data - by bucket, scope, collection.

- Invoice
- Materials
- Merchandise
- Payment
- Purchaser
- Vendor

Query 5: Performing a join operation on the purchaser bucket.

```
FROM Purchaser AS p JOIN Invoice AS i
```

```
ON p.PurchaserId = i.PurchaserId
```

```
WHERE i.InvoiceId = 1605
```

```
SELECT i.InvoiceId,
```

```
p.Name AS Name,
```

```
p.address,
```

```
p.PaymentID AS PaymentID;
```

The screenshot shows a browser window with multiple tabs open, including 'ULC Course Grader', 'a - ankithagaddam99', '530 Final Project Doc', 'Inbox (926)', 'A Virtual Super Market', 'ipl points table - Google', and 'Time and Attendance'. The main content area is titled 'A Virtual Super Market > Query'.

The 'Query Editor' pane contains the following SQL query:

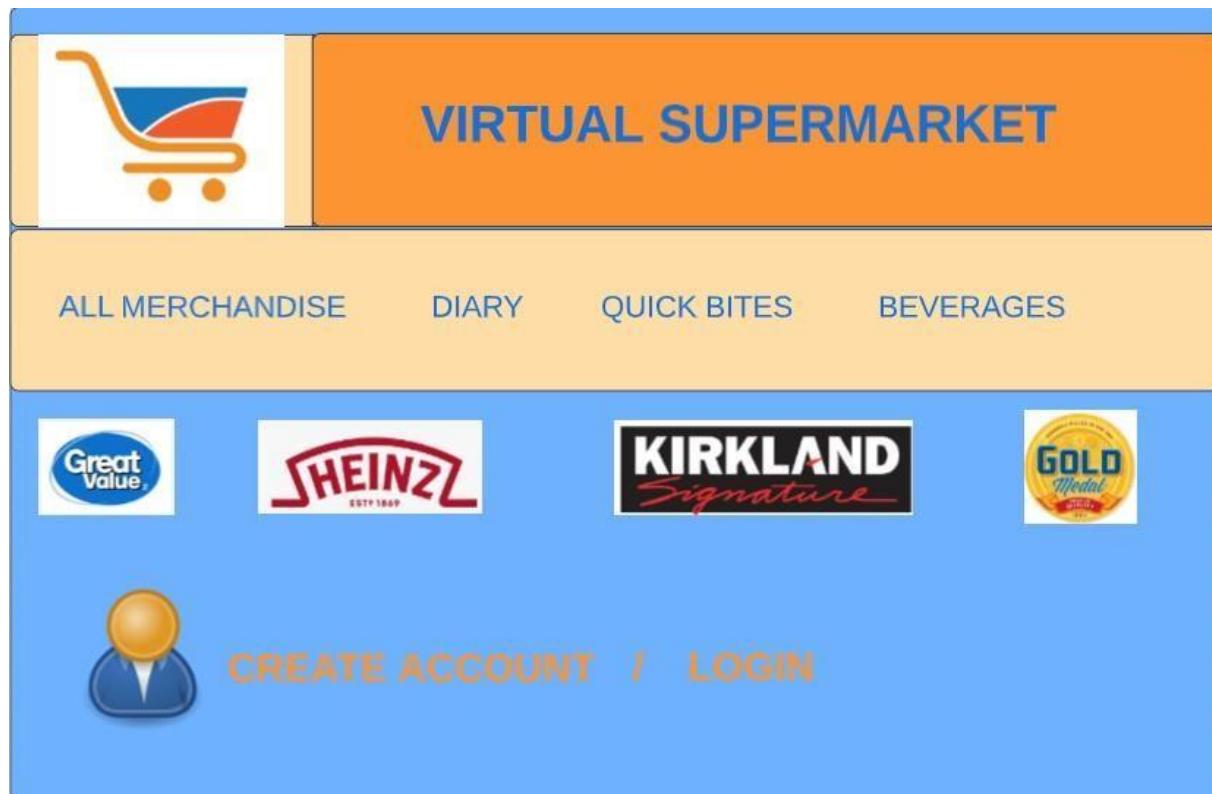
```
1 FROM Purchaser AS p JOIN Invoice AS i
2 ON p.PurchaserId = i.PurchaserId
3 WHERE i.InvoiceId = 1605
4 SELECT i.InvoiceId,
5 p.Name AS Name,
6 p.address,
7 p.PaymentID AS PaymentID;
```

The 'Results' pane displays the query results in a table:

Invoiced	Name	Paymentid	address
1605	Ankitha	1305	Tempe 1-125, Pyramid road 58321

The 'Explore Your Data' sidebar lists document types: Invoice, Materials, Merchandise, Payment, Purchaser, and Vendor.

USER INTERFACE



LOGIN

USERNAME
PASSWORD

LOGIN

Login with Google

SIGNIN

ENTER FIRST NAME
ENTER LAST NAME
CREATE PASSWORD
RE-ENTER PASSWORD

SIGNIN

Sign up with Google

CUSTOMER VIEW**ADMIN VIEW**

Section 7

Summary

The purpose of the database system is to efficiently manage the supermarket's operations, including product materials, pricing, stock availability and payment processing. The system will include an online ordering platform that enables purchasers to easily select and purchase products, as well as a secure payment processing system to ensure a seamless checkout experience. By adopting a comprehensive database system, the virtual supermarket will be able to optimize its operations, enhance customer satisfaction, and ultimately achieve greater business growth and success. To improve our database, we added subcategories for every item in the merchandise table by utilizing a distinct merchandise ID. In the invoice table, we merged the purchaser's ID and the total quantity of items purchased columns, which helped us to determine which merchandise was purchased the most by a certain purchaser. We also added a PaymentGateway column in the Payment's table to check payment methods for an invoice, and an InStock column in our Materials table to verify if the purchaser's invoice is available in the stock. To ensure the flexibility and scalability of our database, we utilized NoSQL, which allowed for quick development and iteration. Finally, to create a more realistic approach, we also designed a user interface for our database.