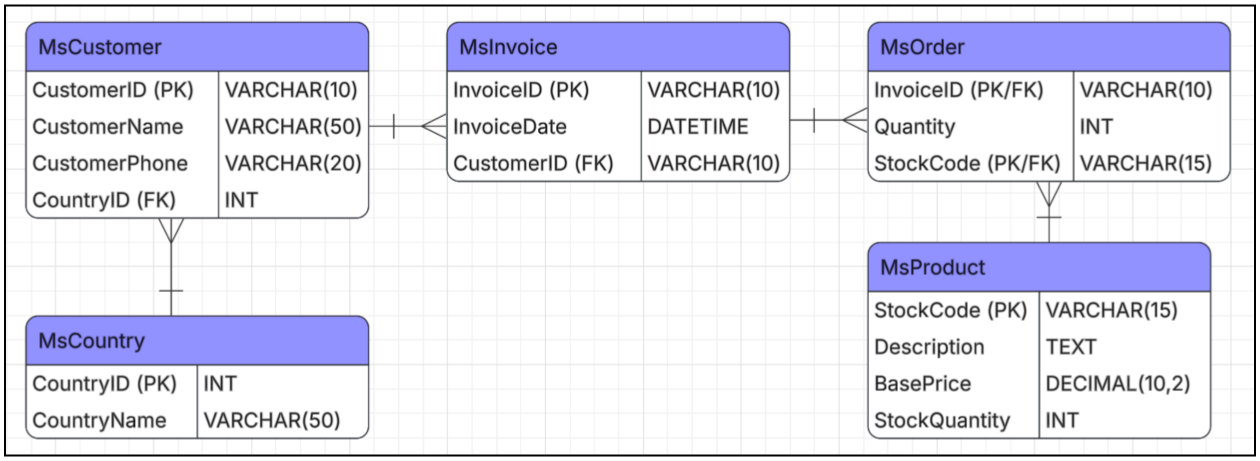
Dokumentasi AOL (Assurance of Learning)
DataBase Technology

Group 5

- Aurelius Elbert Panatasetya - 2802391555
- Brandon Maximillian Avalokita - 2802418512
- Jordhy Alexander Wibisono - 2802389216

Soal:
https://samuelphilip.notion.site/Project-Relational-Database-for-Tokopee-Inc-277b0d533eb180e2b363f439fc225b1d?pvs=143

Phase 1: ERD



MsCustomer	
CustomerID (PK)	VARCHAR(10)
CustomerName	VARCHAR(50)
CustomerPhone	VARCHAR(20)
CountryID (FK)	INT

MsInvoice	
InvoiceID (PK)	VARCHAR(10)
InvoiceDate	DATETIME
CustomerID (FK)	VARCHAR(10)

MsOrder	
InvoiceID (PK/FK)	VARCHAR(10)
Quantity	INT
StockCode (PK/FK)	VARCHAR(15)

MsCountry	
CountryID (PK)	INT
CountryName	VARCHAR(50)

MsProduct	
StockCode (PK)	VARCHAR(15)
Description	TEXT
BasePrice	DECIMAL(10,2)
StockQuantity	INT

ERD diatas terdiri dari 5 tabel dengan atribut masing masing. Berikut adalah penjelasan yang fungsi dari masing masing tabel.

1. Country, berfungsi untuk menghemat memori saat ingin melihat values dari tabel customer. Contohnya kita ada data customer sebanyak 1000 dan masing masing terdiri dari country dengan >5 character, maka memory yang akan digunakan bisa menjadi 5000+. Namun disaat kita assign suatu country dengan ID terkhusus, misal menjadi hanya 3 character, maka hanya dibutuhkan memori sebanyak 3000 dan kita telah menghemat space sebanyak 2000 memori. Ini dapat bermanfaat disaat data terlalu besar dan akan membutuhkan waktu yang lama untuk membuka filenya.
    - countryId (INT) disini berperan sebagai primary key
    - countryName (VARCHAR(50)) merupakan nama nama negara yang lengkap.

2. Customer, berfungsi untuk menampung data atau informasi dari para customer yang pernah menggunakan tokopee. Tabel ini terdiri dari beberapa atribut seperti customerId (sebagai primary key), customerName, customerAge, dan countryId (sebagai foreign key).

    - customerId (VARCHAR(10)) disini berperan sebagai primary key
    - customerName (VARCHAR(50)) merupakan atribut untuk menampung nama nama dari masing masing user berdasarkan customerId yang telah diberikan.

- customerPhone (VARCHAR(20)) merupakan atribut untuk menampung nomor telepon dari masing masing customer
- countryId (INT) disini berperan sebagai foreign key. Oleh karena itu, tabel customer hanya akan memiliki singkatan dari negara asal customer tersebut. Untuk melihat selengkapnya, bisa dicek di tabel Country.

3. Invoice, berfungsi untuk menampung atribut invoiceDate dan juga customerId. Invoice tersendiri dapat diartikan sebagai resi atau pencatatan untuk setiap transaksi yang telah dilakukan oleh seorang user.

   - invoiceId (INT) disini berperan sebagai primary key.
   - invoiceDate (DATETIME) merupakan atribut untuk menampung tanggal dan jam terjadinya transaksi tersebut.
   - customerId (INT) disini berperan sebagai foreign key.

4. Product, Tabel ini berfungsi sebagai master data untuk inventaris barang. Tabel ini menyimpan informasi detail mengenai barang-barang yang dijual di platform (misalnya Tokopee), termasuk harga dan ketersediaan stoknya.

   - stockCode (VARCHAR(15)) berperan sebagai Primary Key (PK). Ini adalah kode unik untuk setiap jenis barang agar tidak tertukar dengan barang lain.
   - description (TEXT) merupakan atribut yang berisi deskripsi atau nama lengkap dari produk tersebut agar pembeli tahu detail barangnya.
   - price (FLOAT) merupakan atribut untuk menyimpan harga satuan dari barang tersebut. Menggunakan tipe data float untuk mengakomodasi kemungkinan harga yang memiliki nilai desimal atau sen.
   - stockAmount (INT) merupakan atribut yang mencatat jumlah stok fisik barang yang tersedia saat ini di gudang.

5. Order, Tabel ini berfungsi sebagai tabel transaksi detail (sering disebut sebagai *Order Details* atau *Line Items*). Tabel ini menjadi penghubung antara Invoice dan Product.Fungsinya sangat krusial karena satu nomor Invoice bisa memuat banyak jenis produk, dan sebaliknya satu jenis produk bisa muncul di banyak Invoice yang berbeda. Tabel ini mencatat "barang apa" dan "berapa banyak" yang dibeli dalam satu struk transaksi tertentu.

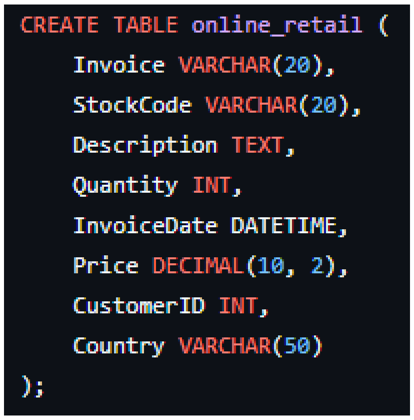
   - orderQuantity (INT) atribut ini menyimpan angka jumlah barang yang dibeli untuk spesifik item tersebut dalam satu transaksi.
   - invoiceId (VARCHAR(10)) berperan sebagai Foreign Key (FK). Atribut ini merujuk kembali ke tabel Invoice, menandakan bahwa item ini adalah bagian dari nomor struk/transaksi tersebut.
   - stockCode (VARCHAR(15)) berperan sebagai Foreign Key (FK). Atribut ini merujuk kembali ke tabel Product, untuk mengidentifikasi barang mana yang sebenarnya dibeli.

ERD Relation:

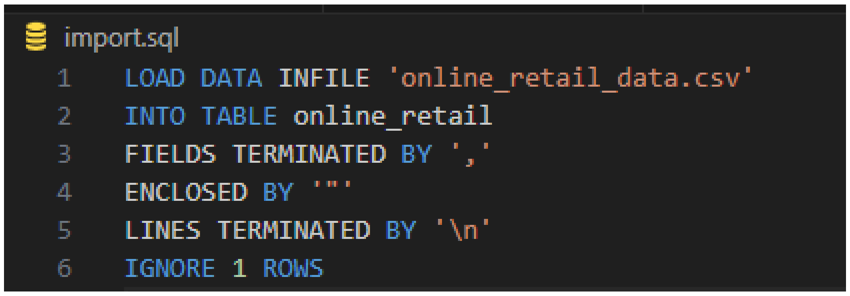
- Customer ke Country: Relasi ini memastikan data negara tersentralisasi sehingga tidak terjadi redundansi penulisan nama negara (One-to-Many).

- Customer ke Invoice: Satu customer bisa melakukan banyak transaksi/memiliki banyak invoice (One-to-Many).
- Invoice ke Order & Product ke Order: Ini memecahkan hubungan Many-to-Many antara Invoice dan Product. Tabel Order mencatat irisan tersebut (misal: Pada Invoice A, ada Produk B sebanyak 2 buah).

Import online_retail_data ke Database



```sql
CREATE TABLE online_retail (
    Invoice VARCHAR(20),
    StockCode VARCHAR(20),
    Description TEXT,
    Quantity INT,
    InvoiceDate DATETIME,
    Price DECIMAL(10, 2),
    CustomerID INT,
    Country VARCHAR(50)
);
```



import.sql

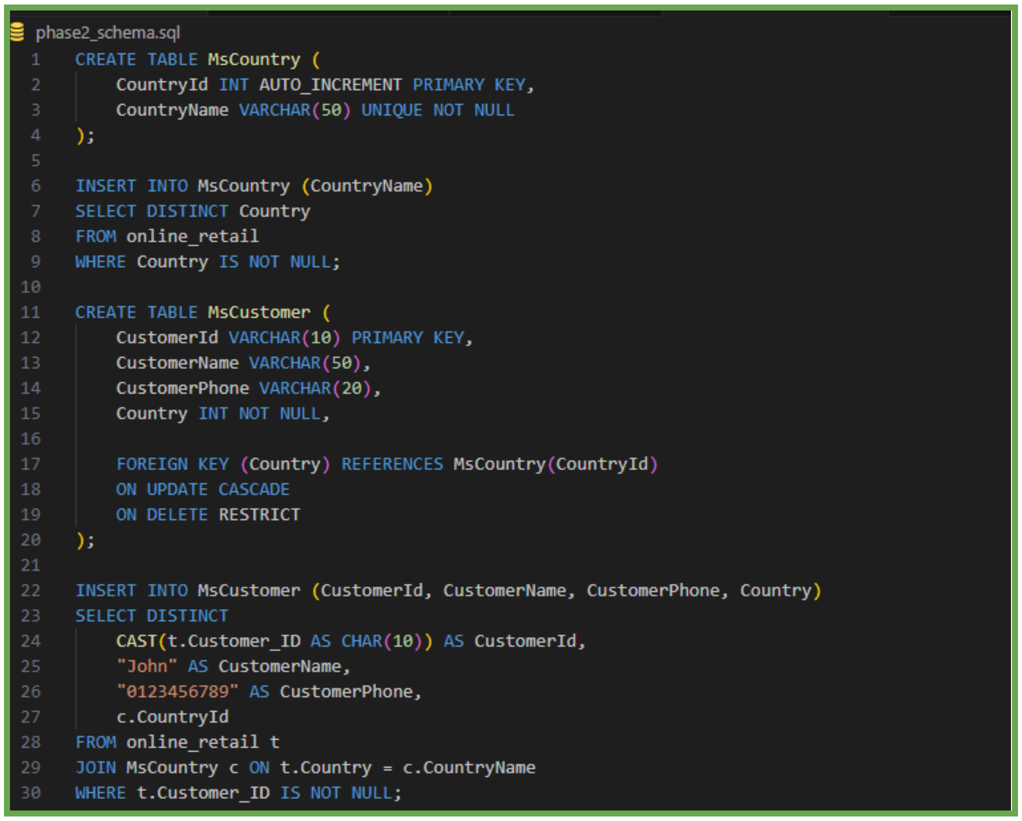
```sql
LOAD DATA INFILE 'online_retail_data.csv'
INTO TABLE online_retail
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
```

Sebelum masuk ke phase 2 kami mengimport terlebih dahulu data dari file online_retail_data.csv ke dalam database kami. Dimana query ini disimpan dalam file ‘import.sql’. Perlu diketahui dalam mengimport ke database kami, file online_retail_data.csv dimasukkan ke dalam folder local database XAMPP terlebih dahulu (C:\xampp\mysql\data\[nama db]). Jika sudah, run script import.sql yang akan membuat table online_retail untuk menahan data dalam bentuk unnormalized dan transfer data dari .csv ke table tersebut

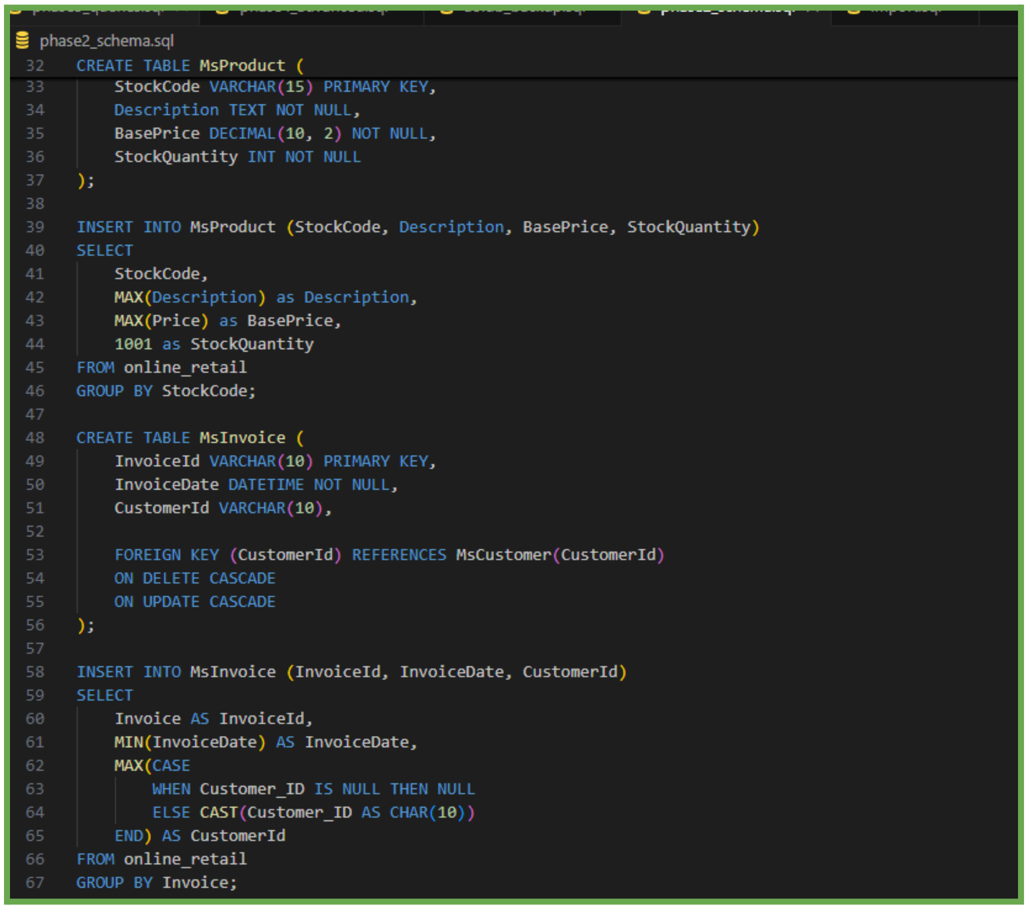
Phase 2 (DDL):

Dalam fase ini kami membuat tabel sesuai ERD sekaligus memindahkan data dari online_retail_data yang sudah di import sebelumnya ke dalam struktur Database yang baru. Pada proses pemindahan data, untuk data yang tidak ada dalam kolom kami, seperti CustomerName kami menggunakan default value seperti ‘John’ yang dimana hal ini juga berlaku untuk kolom-kolom lainnya.

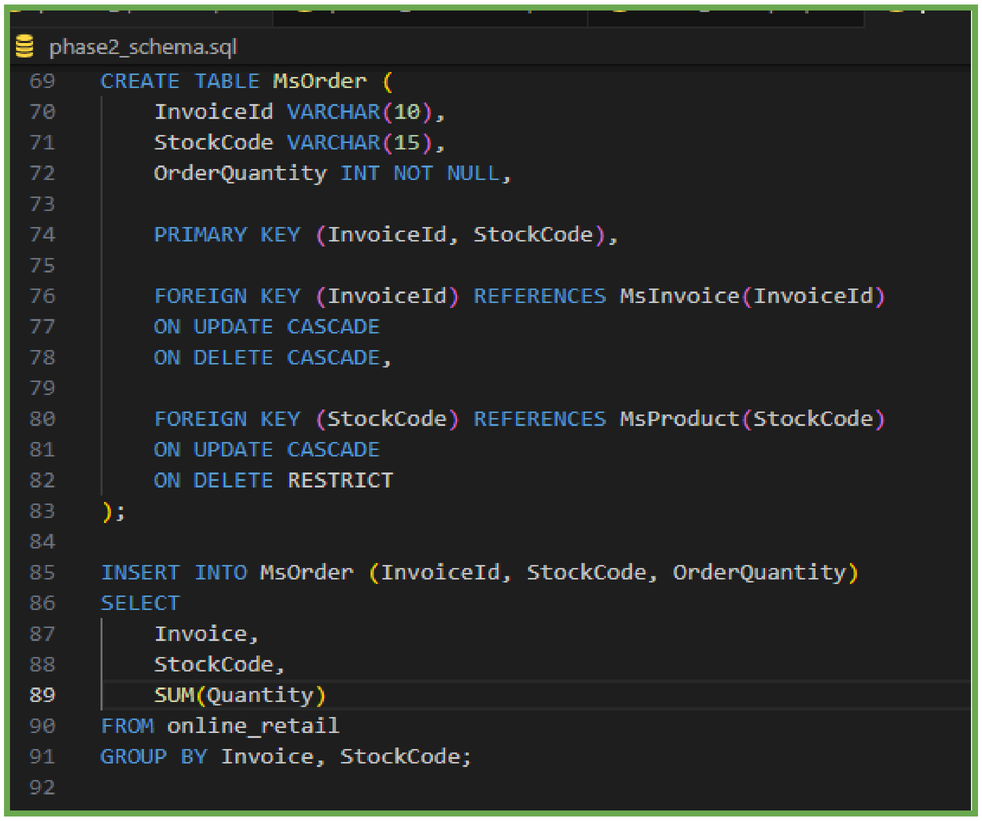
Berikut adalah dokumentasi query fase 2 kami:
(Terlampir di halam berikutnya)


phase2_schema.sql

```sql
CREATE TABLE MsCountry (
    CountryId INT AUTO_INCREMENT PRIMARY KEY,
    CountryName VARCHAR(50) UNIQUE NOT NULL
);

INSERT INTO MsCountry (CountryName)
SELECT DISTINCT Country
FROM online_retail
WHERE Country IS NOT NULL;

CREATE TABLE MsCustomer (
    CustomerId VARCHAR(10) PRIMARY KEY,
    CustomerName VARCHAR(50),
    CustomerPhone VARCHAR(20),
    Country INT NOT NULL,

    FOREIGN KEY (Country) REFERENCES MsCountry(CountryId)
    ON UPDATE CASCADE
    ON DELETE RESTRICT
);

INSERT INTO MsCustomer (CustomerId, CustomerName, CustomerPhone, Country)
SELECT DISTINCT
    CAST(t.Customer_ID AS CHAR(10)) AS CustomerId,
    "John" AS CustomerName,
    "0123456789" AS CustomerPhone,
    c.CountryId
FROM online_retail t
JOIN MsCountry c ON t.Country = c.CountryName
WHERE t.Customer_ID IS NOT NULL;
```


phase2_schema.sql

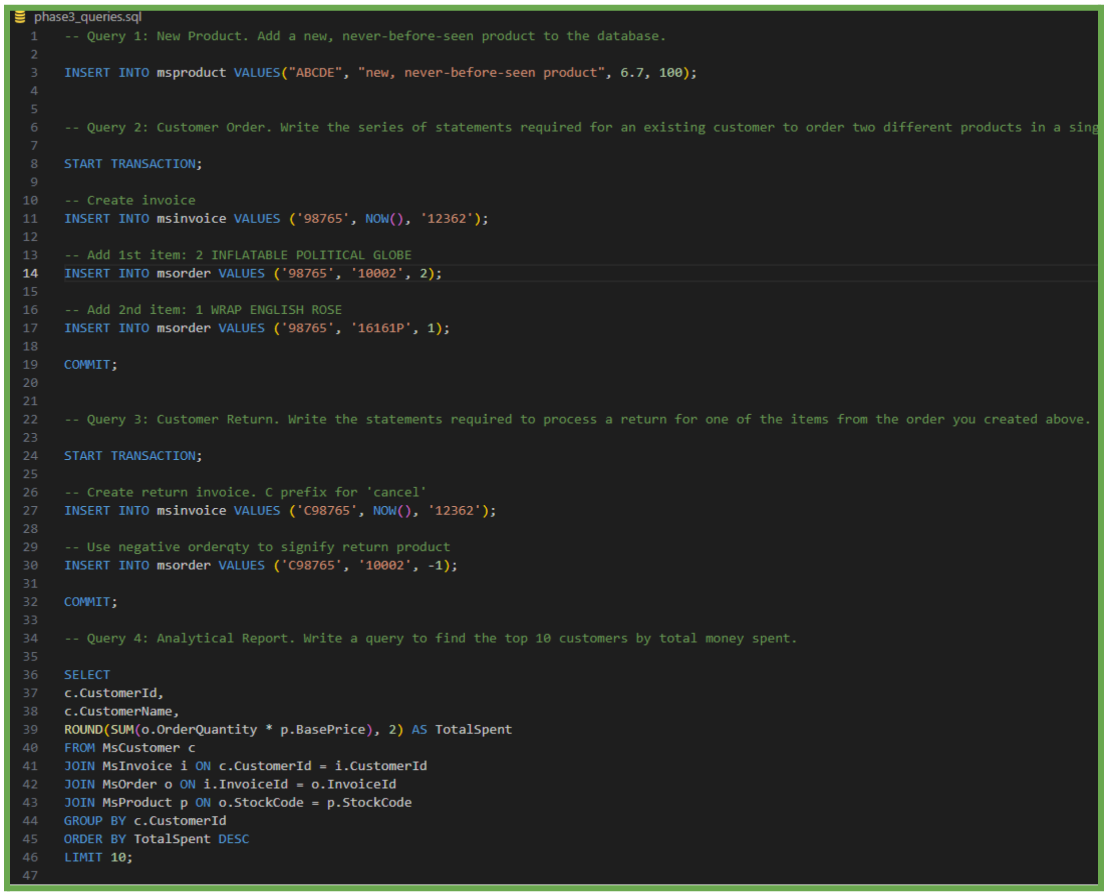
```sql
CREATE TABLE MsProduct (
    StockCode VARCHAR(15) PRIMARY KEY,
    Description TEXT NOT NULL,
    BasePrice DECIMAL(10, 2) NOT NULL,
    StockQuantity INT NOT NULL
);

INSERT INTO MsProduct (StockCode, Description, BasePrice, StockQuantity)
SELECT
    StockCode,
    MAX(Description) as Description,
    MAX(Price) as BasePrice,
    1001 as StockQuantity
FROM online_retail
GROUP BY StockCode;

CREATE TABLE MsInvoice (
    InvoiceId VARCHAR(10) PRIMARY KEY,
    InvoiceDate DATETIME NOT NULL,
    CustomerId VARCHAR(10),

    FOREIGN KEY (CustomerId) REFERENCES MsCustomer(CustomerId)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);

INSERT INTO MsInvoice (InvoiceId, InvoiceDate, CustomerId)
SELECT
    Invoice AS InvoiceId,
    MIN(InvoiceDate) AS InvoiceDate,
    MAX(CASE
        WHEN Customer_ID IS NULL THEN NULL
        ELSE CAST(Customer_ID AS CHAR(10))
    END) AS CustomerId
FROM online_retail
GROUP BY Invoice;
```



phase2_schema.sql

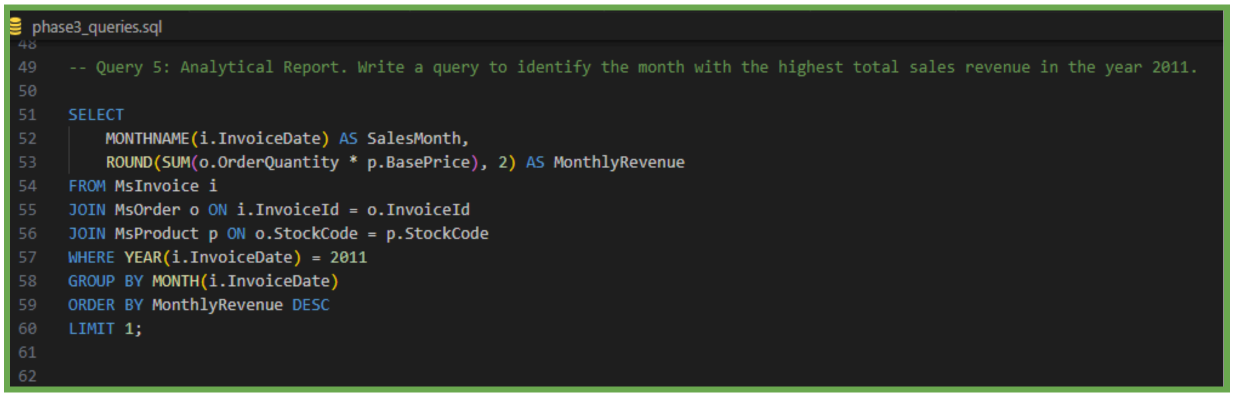
```sql
CREATE TABLE MsOrder (
    InvoiceId VARCHAR(10),
    StockCode VARCHAR(15),
    OrderQuantity INT NOT NULL,

    PRIMARY KEY (InvoiceId, StockCode),

    FOREIGN KEY (InvoiceId) REFERENCES MsInvoice(InvoiceId)
    ON UPDATE CASCADE
    ON DELETE CASCADE,

    FOREIGN KEY (StockCode) REFERENCES MsProduct(StockCode)
    ON UPDATE CASCADE
    ON DELETE RESTRICT
);

INSERT INTO MsOrder (InvoiceId, StockCode, OrderQuantity)
SELECT
    Invoice,
    StockCode,
    SUM(Quantity)
FROM online_retail
GROUP BY Invoice, StockCode;

```

Phase 3 (DML): Dalam fase ini, kami membuat beberapa query untuk memenuhi scenario yang diberi di soal



phase3_queries.sql

```sql
-- Query 1: New Product. Add a new, never-before-seen product to the database.

INSERT INTO msproduct VALUES("ABCDE", "new, never-before-seen product", 6.7, 100);


-- Query 2: Customer Order. Write the series of statements required for an existing customer to order two different products in a sing

START TRANSACTION;

-- Create invoice
INSERT INTO msinvoice VALUES ('98765', NOW(), '12362');

-- Add 1st item: 2 INFLATABLE POLITICAL GLOBE
INSERT INTO msorder VALUES ('98765', '10002', 2);

-- Add 2nd item: 1 WRAP ENGLISH ROSE
INSERT INTO msorder VALUES ('98765', '16161P', 1);

COMMIT;


-- Query 3: Customer Return. Write the statements required to process a return for one of the items from the order you created above.

START TRANSACTION;

-- Create return invoice. C prefix for 'cancel'
INSERT INTO msinvoice VALUES ('C98765', NOW(), '12362');

-- Use negative orderqty to signify return product
INSERT INTO msorder VALUES ('C98765', '10002', -1);

COMMIT;

-- Query 4: Analytical Report. Write a query to find the top 10 customers by total money spent.

SELECT
c.CustomerId,
c.CustomerName,
ROUND(SUM(o.OrderQuantity * p.BasePrice), 2) AS TotalSpent
FROM MsCustomer c
JOIN MsInvoice i ON c.CustomerId = i.CustomerId
JOIN MsOrder o ON i.InvoiceId = o.InvoiceId
JOIN MsProduct p ON o.StockCode = p.StockCode
GROUP BY c.CustomerId
ORDER BY TotalSpent DESC
LIMIT 10;

```

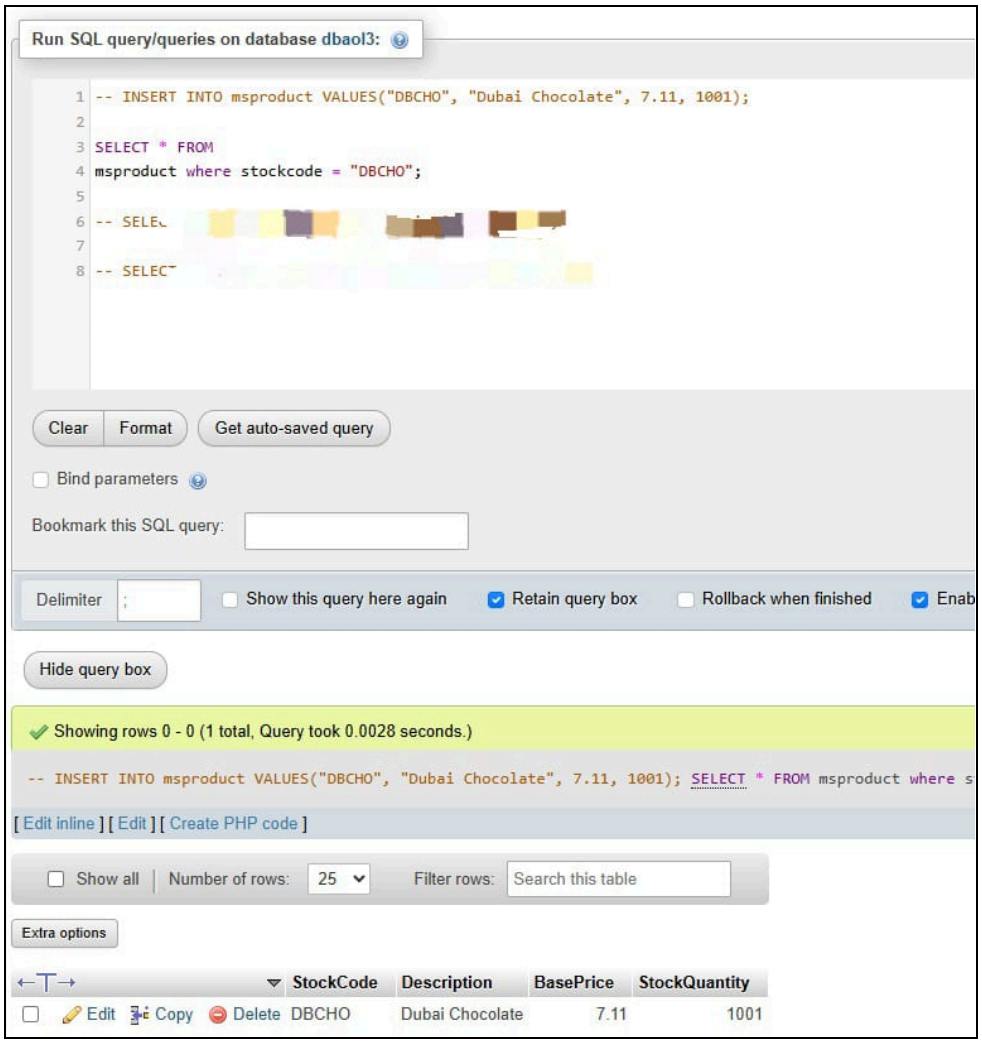


phase3_queries.sql

```sql
-- Query 5: Analytical Report. Write a query to identify the month with the highest total sales revenue in the year 2011.

SELECT
    MONTHNAME(i.InvoiceDate) AS SalesMonth,
    ROUND(SUM(o.OrderQuantity * p.BasePrice), 2) AS MonthlyRevenue
FROM MsInvoice i
JOIN MsOrder o ON i.InvoiceId = o.InvoiceId
JOIN MsProduct p ON o.StockCode = p.StockCode
WHERE YEAR(i.InvoiceDate) = 2011
GROUP BY MONTH(i.InvoiceDate)
ORDER BY MonthlyRevenue DESC
LIMIT 1;
```

Dalam fase ini, kami membuat beberapa query untuk memenuhi scenario yang diberi di soal

-- Query 1: New Product



Run SQL query/queries on database dbaol3:

```sql
-- INSERT INTO msproduct VALUES("DBCHO", "Dubai Chocolate", 7.11, 1001);

SELECT * FROM
msproduct where stockcode = "DBCHO";

-- SELE[redacted]

-- SELEC[redacted]
```

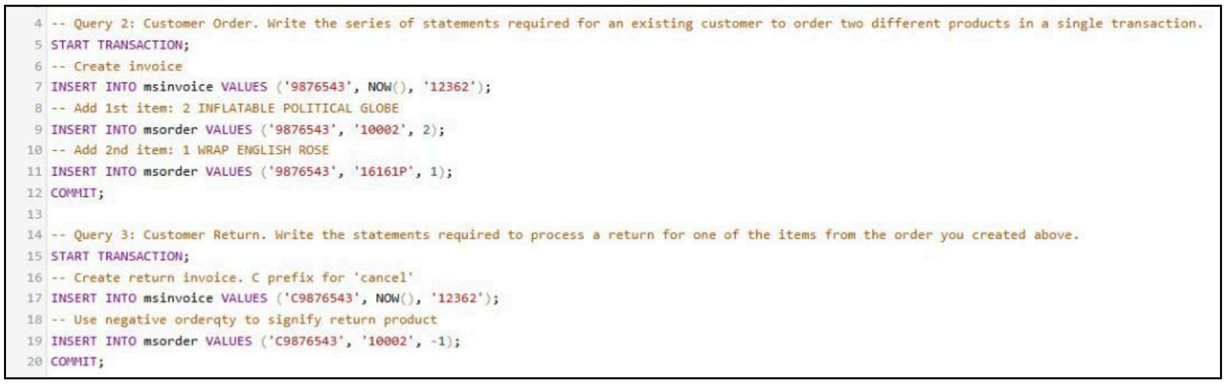
✓ Showing rows 0 - 0 (1 total, Query took 0.0028 seconds.)

```
-- INSERT INTO msproduct VALUES("DBCHO", "Dubai Chocolate", 7.11, 1001); SELECT * FROM msproduct where s
```

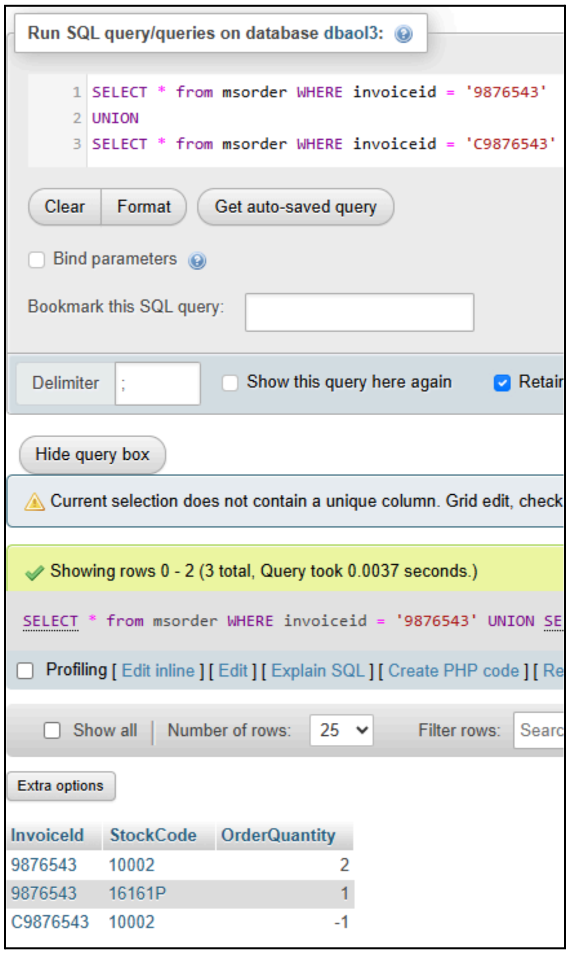
[Edit inline] [Edit] [Create PHP code]
Show all
Number of rows: 25
Filter rows: Search this table
Extra options

	StockCode	Description	BasePrice	StockQuantity
Edit Copy Delete	DBCHO	Dubai Chocolate	7.11	1001

– Query 2 & 3: Customer Transactions



```
-- Query 2: Customer Order. Write the series of statements required for an existing customer to order two different products in a single transaction.
START TRANSACTION;
-- Create invoice
INSERT INTO msinvoice VALUES ('9876543', NOW(), '12362');
-- Add 1st item: 2 INFLATABLE POLITICAL GLOBE
INSERT INTO msorder VALUES ('9876543', '10002', 2);
-- Add 2nd item: 1 WRAP ENGLISH ROSE
INSERT INTO msorder VALUES ('9876543', '16161P', 1);
COMMIT;

-- Query 3: Customer Return. Write the statements required to process a return for one of the items from the order you created above.
START TRANSACTION;
-- Create return invoice. C prefix for 'cancel'
INSERT INTO msinvoice VALUES ('C9876543', NOW(), '12362');
-- Use negative orderqty to signify return product
INSERT INTO msorder VALUES ('C9876543', '10002', -1);
COMMIT;
```



Run SQL query/queries on database dbaol3:

```
SELECT * from msorder WHERE invoiceid = '9876543'
UNION
SELECT * from msorder WHERE invoiceid = 'C9876543'
```

Clear
Format
Get auto-saved query
Bind parameters
Bookmark this SQL query:
Delimiter ;
Show this query here again
Retair
Hide query box

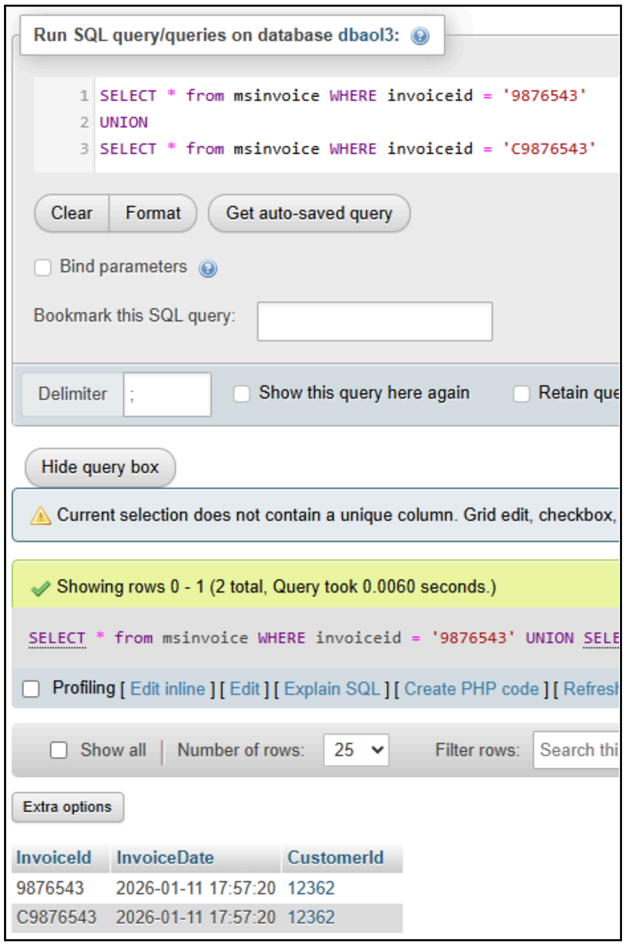
Current selection does not contain a unique column. Grid edit, check

Showing rows 0 - 2 (3 total, Query took 0.0037 seconds.)

```
SELECT * from msorder WHERE invoiceid = '9876543' UNION SE
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Re
Show all | Number of rows: 25 Filter rows: Searc
Extra options

InvoiceId	StockCode	OrderQuantity
9876543	10002	2
9876543	16161P	1
C9876543	10002	-1



Run SQL query/queries on database dbaol3:

```
SELECT * from msinvoice WHERE invoiceid = '9876543'
UNION
SELECT * from msinvoice WHERE invoiceid = 'C9876543'
```

Clear
Format
Get auto-saved query
Bind parameters
Bookmark this SQL query:
Delimiter ;
Show this query here again
Retain que
Hide query box

Current selection does not contain a unique column. Grid edit, checkbox,

Showing rows 0 - 1 (2 total, Query took 0.0060 seconds.)

```
SELECT * from msinvoice WHERE invoiceid = '9876543' UNION SELE
```

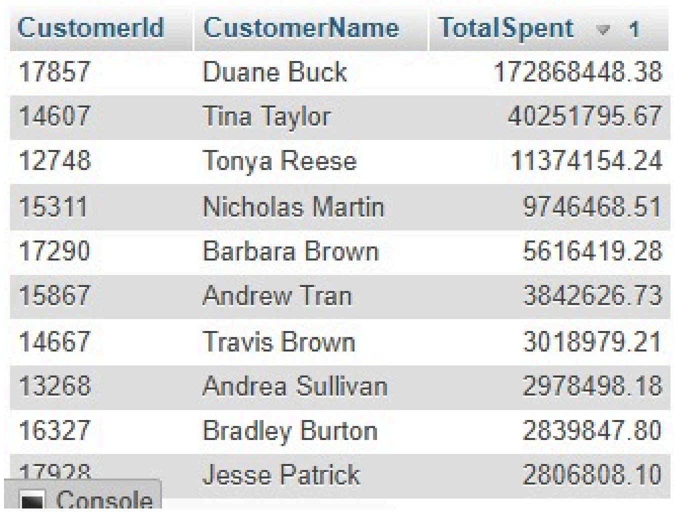
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresl
Show all | Number of rows: 25 Filter rows: Search thi
Extra options

InvoiceId	InvoiceDate	CustomerId
9876543	2026-01-11 17:57:20	12362
C9876543	2026-01-11 17:57:20	12362

– Query 4 & 5: Reports
(Query 4)
SELECT
c.CustomerId,
c.CustomerName,
ROUND(SUM(o.OrderQuantity * p.BasePrice), 2) AS TotalSpent
FROM MsCustomer c

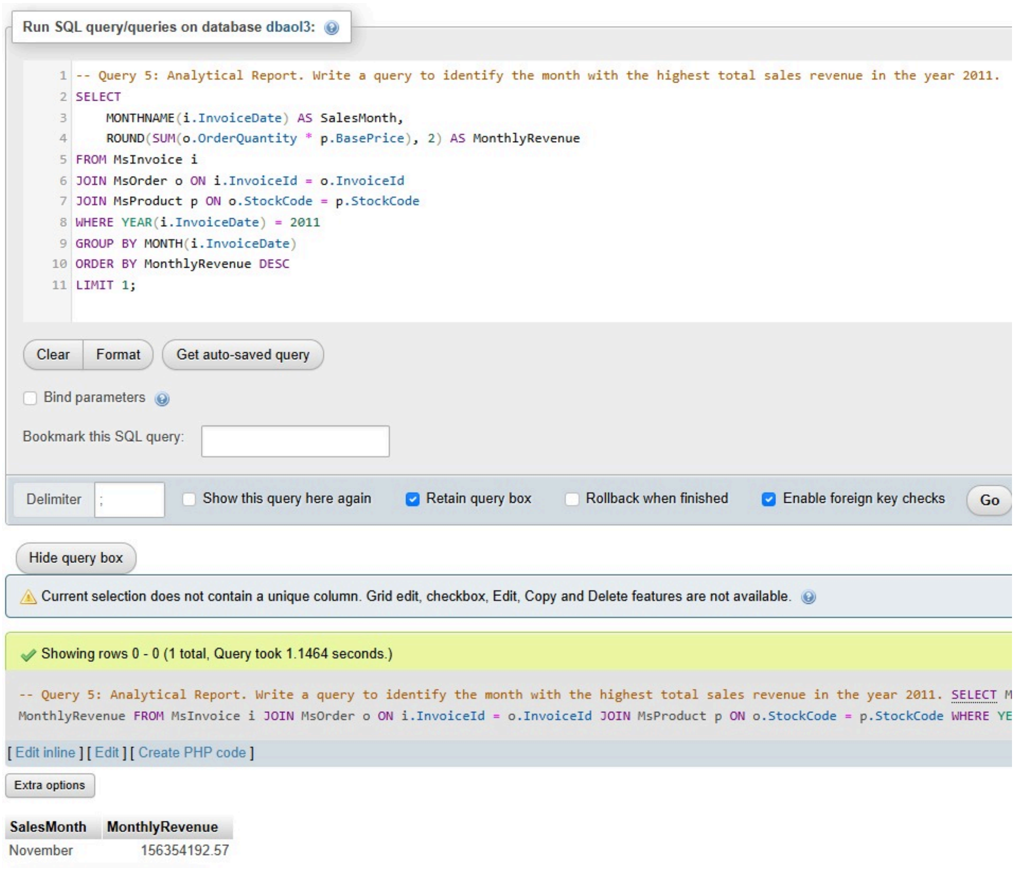
JOIN MsInvoice i ON c.CustomerId = i.CustomerId
JOIN MsOrder o ON i.InvoiceId = o.InvoiceId
JOIN MsProduct p ON o.StockCode = p.StockCode
WHERE c.CustomerID != 0
GROUP BY c.CustomerId
ORDER BY TotalSpent DESC
LIMIT 10;

(Query 4 Result)



CustomerId	CustomerName	TotalSpent 1
17857	Duane Buck	172868448.38
14607	Tina Taylor	40251795.67
12748	Tonya Reese	11374154.24
15311	Nicholas Martin	9746468.51
17290	Barbara Brown	5616419.28
15867	Andrew Tran	3842626.73
14667	Travis Brown	3018979.21
13268	Andrea Sullivan	2978498.18
16327	Bradley Burton	2839847.80
17928	Jesse Patrick	2806808.10

(Query 5)



Run SQL query/queries on database dbaol3:

```
-- Query 5: Analytical Report. Write a query to identify the month with the highest total sales revenue in the year 2011.
SELECT
    MONTHNAME(i.InvoiceDate) AS SalesMonth,
    ROUND(SUM(o.OrderQuantity * p.BasePrice), 2) AS MonthlyRevenue
FROM MsInvoice i
JOIN MsOrder o ON i.InvoiceId = o.InvoiceId
JOIN MsProduct p ON o.StockCode = p.StockCode
WHERE YEAR(i.InvoiceDate) = 2011
GROUP BY MONTH(i.InvoiceDate)
ORDER BY MonthlyRevenue DESC
LIMIT 1;
```

Delimiter ;
Show this query here again [ ] Retain query box [✓] Rollback when finished [ ] Enable foreign key checks [✓]

Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

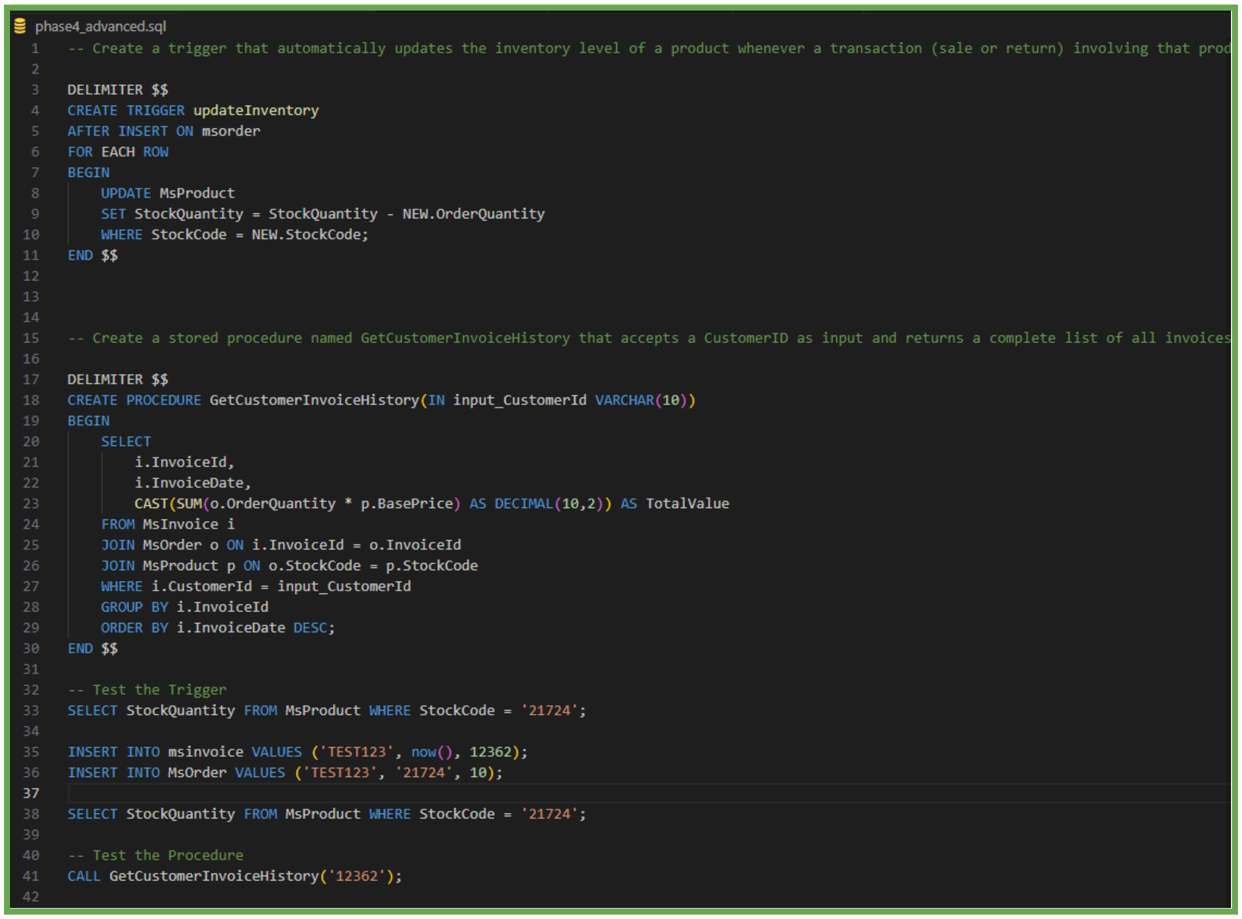
Showing rows 0 - 0 (1 total, Query took 1.1464 seconds.)

```
-- Query 5: Analytical Report. Write a query to identify the month with the highest total sales revenue in the year 2011. SELECT M
MonthlyRevenue FROM MsInvoice i JOIN MsOrder o ON i.InvoiceId = o.InvoiceId JOIN MsProduct p ON o.StockCode = p.StockCode WHERE YE
```

SalesMonth	MonthlyRevenue
November	156354192.57

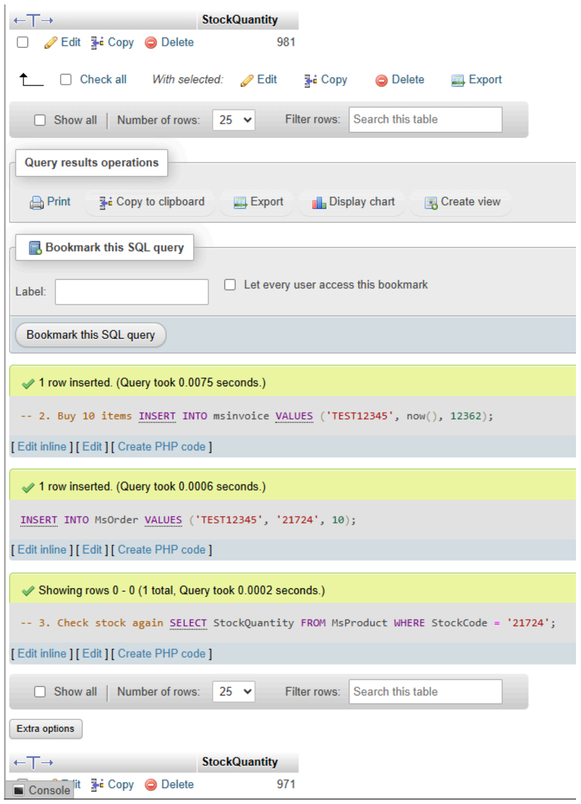
Phase 4 (Trigger & Procedure):

Dalam fase ini kami membuat trigger & prosedur berdasarkan kasus dalam soal.

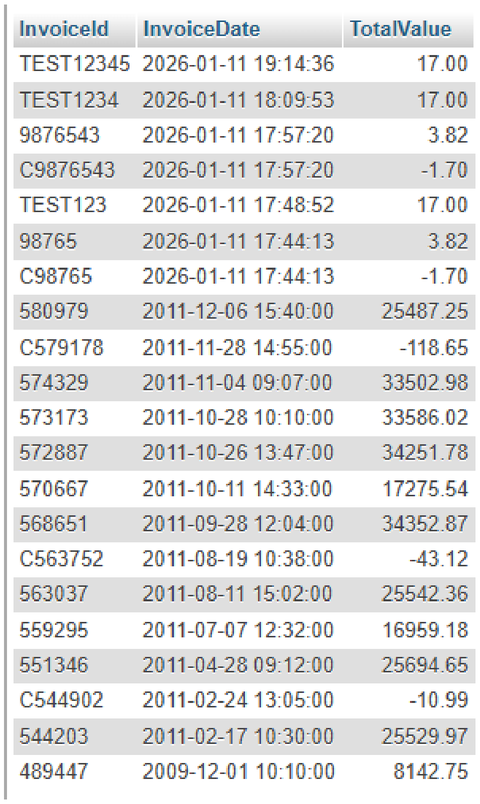


```sql
phase4_advanced.sql
-- Create a trigger that automatically updates the inventory level of a product whenever a transaction (sale or return) involving that prod

DELIMITER $$
CREATE TRIGGER updateInventory
AFTER INSERT ON msorder
FOR EACH ROW
BEGIN
    UPDATE MsProduct
    SET StockQuantity = StockQuantity - NEW.OrderQuantity
    WHERE StockCode = NEW.StockCode;
END $$



-- Create a stored procedure named GetCustomerInvoiceHistory that accepts a CustomerID as input and returns a complete list of all invoices

DELIMITER $$
CREATE PROCEDURE GetCustomerInvoiceHistory(IN input_CustomerId VARCHAR(10))
BEGIN
    SELECT
        i.InvoiceId,
        i.InvoiceDate,
        CAST(SUM(o.OrderQuantity * p.BasePrice) AS DECIMAL(10,2)) AS TotalValue
    FROM MsInvoice i
    JOIN MsOrder o ON i.InvoiceId = o.InvoiceId
    JOIN MsProduct p ON o.StockCode = p.StockCode
    WHERE i.CustomerId = input_CustomerId
    GROUP BY i.InvoiceId
    ORDER BY i.InvoiceDate DESC;
END $$

-- Test the Trigger
SELECT StockQuantity FROM MsProduct WHERE StockCode = '21724';

INSERT INTO msinvoice VALUES ('TEST123', now(), 12362);
INSERT INTO MsOrder VALUES ('TEST123', '21724', 10);

SELECT StockQuantity FROM MsProduct WHERE StockCode = '21724';

-- Test the Procedure
CALL GetCustomerInvoiceHistory('12362');

```

(Trigger) (Procedure)



```
StockQuantity
Edit  Copy  Delete    981

Check all  With selected:  Edit  Copy  Delete  Export
Show all | Number of rows: 25   Filter rows: Search this table

Query results operations
Print  Copy to clipboard  Export  Display chart  Create view

Bookmark this SQL query
Label:    Let every user access this bookmark
Bookmark this SQL query

✓ 1 row inserted. (Query took 0.0075 seconds.)
-- 2. Buy 10 items INSERT INTO msinvoice VALUES ('TEST12345', now(), 12362);
[ Edit inline ] [ Edit ] [ Create PHP code ]

✓ 1 row inserted. (Query took 0.0006 seconds.)
INSERT INTO MsOrder VALUES ('TEST12345', '21724', 10);
[ Edit inline ] [ Edit ] [ Create PHP code ]

✓ Showing rows 0 - 0 (1 total, Query took 0.0002 seconds.)
-- 3. Check stock again SELECT StockQuantity FROM MsProduct WHERE StockCode = '21724';
[ Edit inline ] [ Edit ] [ Create PHP code ]

Show all | Number of rows: 25   Filter rows: Search this table
Extra options

StockQuantity
Edit  Copy  Delete    971
Console
```



InvoiceId	InvoiceDate	TotalValue
TEST12345	2026-01-11 19:14:36	17.00
TEST1234	2026-01-11 18:09:53	17.00
9876543	2026-01-11 17:57:20	3.82
C9876543	2026-01-11 17:57:20	-1.70
TEST123	2026-01-11 17:48:52	17.00
98765	2026-01-11 17:44:13	3.82
C98765	2026-01-11 17:44:13	-1.70
580979	2011-12-06 15:40:00	25487.25
C579178	2011-11-28 14:55:00	-118.65
574329	2011-11-04 09:07:00	33502.98
573173	2011-10-28 10:10:00	33586.02
572887	2011-10-26 13:47:00	34251.78
570667	2011-10-11 14:33:00	17275.54
568651	2011-09-28 12:04:00	34352.87
C563752	2011-08-19 10:38:00	-43.12
563037	2011-08-11 15:02:00	25542.36
559295	2011-07-07 12:32:00	16959.18
551346	2011-04-28 09:12:00	25694.65
C544902	2011-02-24 13:05:00	-10.99
544203	2011-02-17 10:30:00	25529.97
489447	2009-12-01 10:10:00	8142.75

LAMPIRAN

Link Full Pengerjaan Project: https://github.com/Elbrtt/AOL_DB