

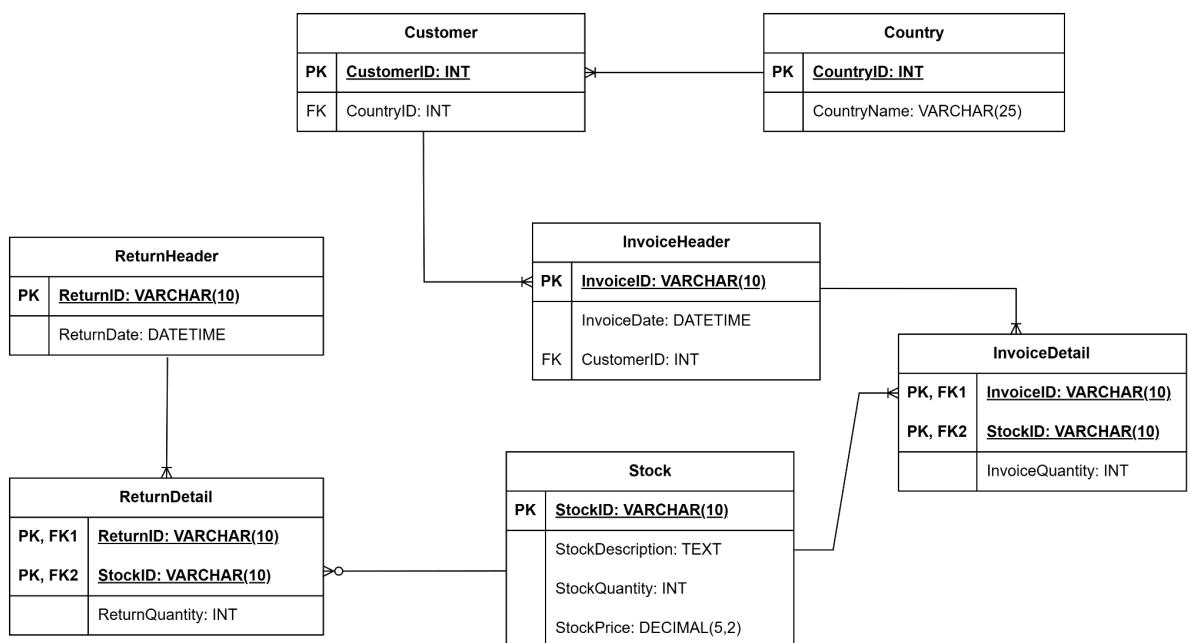
# DOKUMENTASI AoL

## DATABASE TECHNOLOGY

Anggota Kelompok :

- 2802403694 - Edward Aria Tanujaya
  - 2802400515 - Jeta Sunanda
  - 2802404734 - Varel Imanuel Putera
- 

### 1. Entity Relation Diagram (ERD)



Berdasarkan ERD yang dibuat, tabel/entities yang akan digunakan adalah:

- 1) **Customer**, untuk menyimpan data customer, terdiri dari:
  - CustomerID (PK) : sebagai identifier unik untuk customer dalam tabel Customer.
  - CountryID (FK) : mereferensikan tabel Country yang menyimpan negara asal customer.
- 2) **Country**, untuk menyimpan data negara asal customer, terdiri dari:
  - CountryID (PK) : sebagai identifier unik negara-negara dalam tabel Country.
  - CountryName : menyimpan nama negara asal customer.
- 3) **InvoiceHeader**, untuk menyimpan data transaksi yang dilakukan customer, terdiri dari:
  - InvoiceID (PK) : sebagai identifier unik invoice transaksi yang dilakukan customer dalam tabel InvoiceHeader.
  - InvoiceDate : untuk menyimpan tanggal dan waktu terjadinya transaksi.

- CustomerID (FK) : mereferensikan tabel Customer atau customer yang melakukan transaksi.
- 4) **InvoiceDetail**, untuk menyimpan detail transaksi yang telah dilakukan, terdiri dari:
- InvoiceID (PK, FK) : mereferensikan tabel InvoiceHeader dan juga sebagai partial key unik tabel InvoiceDetail.
  - StockID (PK, FK) : mereferensikan tabel Stock dan juga sebagai partial key unik tabel InvoiceDetail.
  - InvoiceQuantity : untuk menyimpan jumlah kuantitas barang yang dibeli oleh customer.
- 5) **Stock**, untuk menyimpan data stock barang/produk yang ada, terdiri dari:
- StockID (PK) : sebagai identifier unik barang/produk yang ada dalam Stock.
  - StockDescription : menyimpan deskripsi produk yang ada dalam Stock.
  - StockQuantity : menyimpan kuantitas/jumlah produk yang ada dalam Stock.
  - StockPrice : menyimpan harga barang/produk yang ada dalam Stock.
- 6) **ReturnHeader**, untuk menyimpan data pengembalian barang yang dilakukan, terdiri dari:
- ReturnID (PK) : sebagai identifier unik untuk setiap pengembalian barang yang ada dalam tabel ReturnHeader.
  - ReturnDate : menyimpan tanggal terjadinya pengembalian barang tertentu.
- 7) **ReturnDetail**, untuk menyimpan detail pengembalian barang yang dilakukan, terdiri dari:
- ReturnID (PK, FK) : mereferensikan tabel ReturnHeader dan sebagai partial key unik untuk tabel ReturnDetail.
  - StockID (PK, FK) : mereferensikan tabel Stock dan sebagai partial key unik untuk tabel ReturnDetail.
  - ReturnQuantity : untuk menyimpan kuantitas/jumlah produk yang dikembalikan.
- Berikut adalah relationship/hubungan antar tabel/entity yang ada berdasarkan ERD yang dibuat:
- a) **Country  $\leftrightarrow$  Customer : One to Many**
- Satu customer hanya bisa berasal dari satu country, namun satu country bisa menjadi asal negara dari banyak customer.
- b) **Customer  $\leftrightarrow$  InvoiceHeader : One to Many**
- Satu invoice atau transaksi hanya bisa dilakukan oleh satu customer, namun satu customer bisa melakukan banyak transaksi.
- c) **InvoiceHeader  $\leftrightarrow$  InvoiceDetail : One to Many**

Satu invoice header bisa memiliki banyak invoice detail, namun satu invoice detail hanya punya satu invoice header.

**d) Stock ↔ InvoiceDetail : One to Many**

Satu product yang berada di stock bisa masuk ke dalam banyak invoice, namun satu stockid hanya ada di satu invoice detail.

**e) ReturnHeader ↔ ReturnDetail : One to Many**

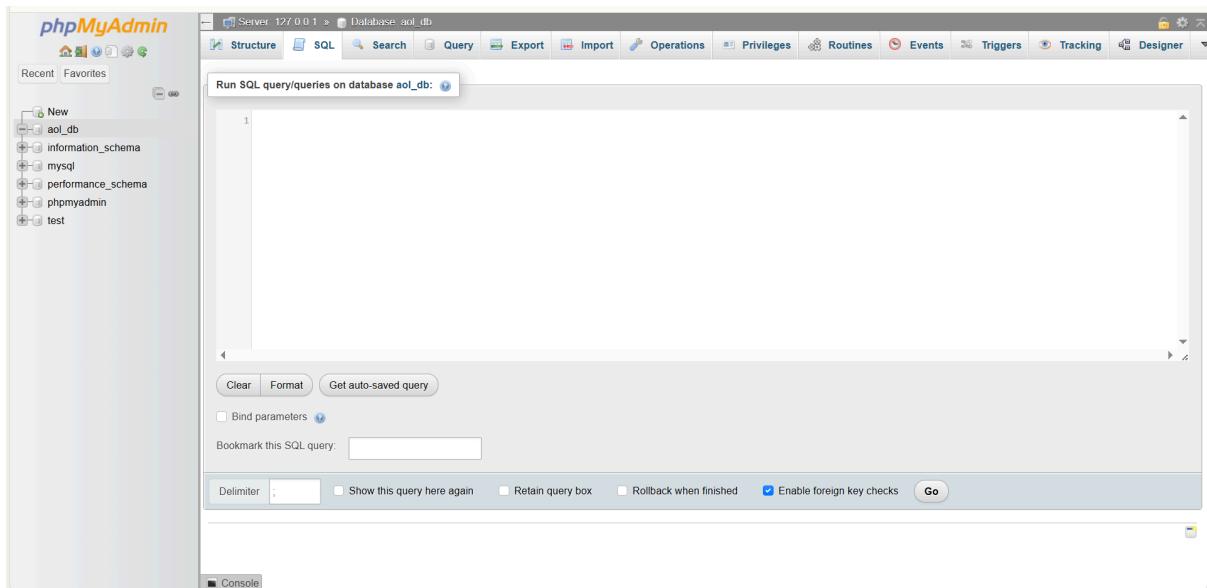
Satu return header bisa masuk ke dalam banyak return detail.

**f) Stock ↔ ReturnDetail : Zero to Many**

Satu stock bisa masuk ke dalam banyak return detail atau pun tidak sama sekali ada return.

## 2. Database Implementation (DDL)

- Sebelum menjalankan *schema.sql*



- Setelah menjalankan *schema.sql*

```

CREATE TABLE Country(
    CountryID INT NOT NULL PRIMARY KEY,
    CountryName VARCHAR(25) NOT NULL UNIQUE
);

CREATE TABLE Customer (
    CustomerID INT NOT NULL PRIMARY KEY,
    CountryID INT NOT NULL,
    CONSTRAINT fk_Customer
        FOREIGN KEY (CountryID)
        REFERENCES Country(CountryID)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

CREATE TABLE Stock (
    StockID VARCHAR(10) NOT NULL PRIMARY KEY,
    StockDescription TEXT NOT NULL,
    StockQuantity INT NOT NULL CHECK (StockQuantity >= 0),
    StockPrice DECIMAL(4,2) NOT NULL
);

CREATE TABLE InvoiceHeader (
    InvoiceID VARCHAR(10) PRIMARY KEY NOT NULL,
    InvoiceDate DATETIME,
    CustomerID INT NOT NULL,
    CONSTRAINT fk_invoice
        FOREIGN KEY (CustomerID)
        REFERENCES Customer(CustomerID)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

CREATE TABLE ReturnHeader (
    ReturnID VARCHAR(10) NOT NULL PRIMARY KEY,
    ReturnDate DATETIME
);

```

Isi *schema.sql*:

```

CREATE TABLE Country(
    CountryID INT NOT NULL PRIMARY KEY,
    CountryName VARCHAR(25) NOT NULL UNIQUE
);

CREATE TABLE Customer (
    CustomerID INT NOT NULL PRIMARY KEY,
    CountryID INT NOT NULL,
    CONSTRAINT fk_Customer
        FOREIGN KEY (CountryID)
        REFERENCES Country(CountryID)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

CREATE TABLE Stock (
    StockID VARCHAR(10) NOT NULL PRIMARY KEY,
    StockDescription TEXT NOT NULL,
    StockQuantity INT NOT NULL CHECK (StockQuantity >= 0),
    StockPrice DECIMAL(5,2) NOT NULL
);

CREATE TABLE InvoiceHeader (
    InvoiceID VARCHAR(10) PRIMARY KEY NOT NULL,
    InvoiceDate DATETIME,
    CustomerID INT NOT NULL,
    CONSTRAINT fk_invoice
        FOREIGN KEY (CustomerID)
        REFERENCES Customer(CustomerID)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

```

```

    REFERENCES Customer(CustomerID)
    ON UPDATE CASCADE
    ON DELETE CASCADE
);

CREATE TABLE ReturnHeader(
    ReturnID VARCHAR(10) NOT NULL PRIMARY KEY,
    ReturnDate DATETIME
);

CREATE TABLE InvoiceDetail(
    InvoiceID VARCHAR(10) NOT NULL,
    StockID VARCHAR(10) NOT NULL,
    InvoiceQuantity INT NOT NULL CHECK (InvoiceQuantity >= 0),

    CONSTRAINT pk_invoice_detail
        PRIMARY KEY (InvoiceID, StockID),

    CONSTRAINT fk_invoicedid_invoice_detail
        FOREIGN KEY (InvoiceID)
        REFERENCES InvoiceHeader(InvoiceID)
        ON UPDATE CASCADE
        ON DELETE CASCADE,

    CONSTRAINT fk_StockID_invoice_detail
        FOREIGN KEY (StockID)
        REFERENCES Stock(StockID)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

CREATE TABLE ReturnDetail (
    ReturnID VARCHAR(10) NOT NULL,
    StockID VARCHAR(10) NOT NULL,
    ReturnQuantity INT NOT NULL CHECK (ReturnQuantity < 0),
    CONSTRAINT pk_return_quantity
        PRIMARY KEY (ReturnID, StockID),

    CONSTRAINT fk_ReturnID_return_detail
        FOREIGN KEY (ReturnID)
        REFERENCES ReturnHeader(ReturnID)
        ON UPDATE CASCADE
        ON DELETE CASCADE,

    CONSTRAINT fk_StockID_return_detail
        FOREIGN KEY (StockID)

```

```
REFERENCES Stock(StockID)
ON UPDATE CASCADE
ON DELETE CASCADE
);
```

Berdasarkan *schema.sql* tersebut, referential integrity sudah terdapat di semua tabel yang mengambil foreign key dari tabel lainnya. Untuk setiap quantity yang berada di tabel baik InvoiceDetail maupun Stock, diberikan constraint untuk harus lebih dari atau sama dengan 0, dan untuk setiap quantity di tabel ReturnDetail harus lebih kecil dari 0. Pembentukan schema sql ini dibuat berdasarkan ERD yang telah dibuat.

Data Insertion:

**a) Tabel Country**

Setelah melakukan filterisasi dengan bantuan code Python, didapatkan sejumlah list country/negara yang unique, sehingga dari list tersebut bisa langsung dimasukkan ke dalam tabel Country.

Insertion Query:

```
INSERT INTO Country (CountryID, CountryName)
VALUES (1, 'United Kingdom'),
(2, 'France'),
(3, 'USA'),
(4, 'Belgium'),
(5, 'Australia'),
(6, 'EIRE'),
(7, 'Germany'),
(8, 'Portugal'),
(9, 'Japan'),
(10, 'Denmark'),
(11, 'Nigeria'),
(12, 'Netherlands'),
(13, 'Poland'),
(14, 'Spain'),
(15, 'Channel Islands'),
(16, 'Italy'),
(17, 'Cyprus'),
(18, 'Greece'),
(19, 'Norway'),
(20, 'Austria'),
(21, 'Sweden'),
(22, 'United Arab Emirates'),
(23, 'Finland'),
(24, 'Switzerland'),
(25, 'Unspecified'),
```

```
(26, 'Malta'),  
(27, 'Bahrain'),  
(28, 'RSA'),  
(29, 'Bermuda'),  
(30, 'Hong Kong'),  
(31, 'Singapore'),  
(32, 'Thailand'),  
(33, 'Israel'),  
(34, 'Lithuania'),  
(35, 'West Indies'),  
(36, 'Lebanon'),  
(37, 'Korea'),  
(38, 'Brazil'),  
(39, 'Canada'),  
(40, 'Iceland'),  
(41, 'Saudi Arabia'),  
(42, 'Czech Republic'),  
(43, 'European Community')
```

- Sebelum data dimasukkan ke dalam tabel Country

The screenshot shows the phpMyAdmin interface for the 'aol\_db' database. The 'Country' table is selected. The SQL query 'SELECT \* FROM Country;' is run, resulting in an empty result set. The interface includes a sidebar with database and schema navigation, and various management tabs like Browse, Structure, and Insert.

- Setelah data dimasukkan ke dalam table Country

	CountryID	CountryName
<input type="checkbox"/>	5	Australia
<input type="checkbox"/>	20	Austria
<input type="checkbox"/>	27	Bahrain
<input type="checkbox"/>	4	Belgium
<input type="checkbox"/>	29	Bermuda
<input type="checkbox"/>	38	Brazil
<input type="checkbox"/>	39	Canada
<input type="checkbox"/>	15	Channel Islands
<input type="checkbox"/>	17	Cyprus
<input type="checkbox"/>	42	Czech Republic
<input type="checkbox"/>	10	Denmark
<input type="checkbox"/>	6	EIRE
<input type="checkbox"/>	43	European Community
<input type="checkbox"/>	23	Finland
<input type="checkbox"/>	2	France
<input type="checkbox"/>	7	Germany
<input type="checkbox"/>	18	Greece

### b) Tabel Customer

Setelah melakukan filterisasi dengan bantuan code python, didapatkan sejumlah list customer yang unique, sehingga dari list tersebut bisa langsung dimasukkan ke dalam tabel Country.

Insertion Query:

```
INSERT INTO Customer (CustomerID, CountryID)
VALUES (12636,3),
(17592,1),
(17641,1),
(17056,1),
(14654,1),
.....
(17581,1),
(13777,1),
(15804,1),
(13113,1),
(12680,2)
```

Notes: Dikarenakan ada total 5942 data customer yang berbeda, maka tampilan query insertion untuk tabel Customer di atas kami ringkas.

- Sebelum data dimasukkan ke tabel customer

The screenshot shows the phpMyAdmin interface for the AOL database. The left sidebar lists various tables: New, aol\_db, New, country, customer, invoicedetail, invoiceheader, returndetail, returnheader, stock, information\_schema, mysql, performance\_schema, phpmyadmin, and test. The main area is titled 'Table: Customer' and displays the results of the query 'SELECT \* FROM Customer;'. A message at the top says 'MySQL returned an empty result set (i.e. zero rows). (Query took 0.0006 seconds.)'. Below the message, there is a table header 'CustomerID CountryID' and a section for 'Query results operations' which includes 'Create view' and 'Bookmark this SQL query' buttons.

- Setelah data dimasukkan ke tabel customer

The screenshot shows the same phpMyAdmin interface as above, but now the Customer table contains data. The message at the top says 'Showing rows 0 - 24 (5942 total, Query took 0.0004 seconds.)'. The table header is 'CustomerID CountryID'. The data is as follows:

CustomerID	CountryID
12346	1
12608	1
12745	1
12746	1
12747	1
12748	1
12749	1
12777	1
12819	1
12820	1
12821	1
12822	1
12823	1
12824	1

### c) Tabel Stock

Setelah melakukan filterisasi dengan menggunakan Python, didapatkan sejumlah list stock, sehingga dari list tersebut bisa langsung dimasukkan ke dalam table Stock.

Insertion Query :

```
INSERT INTO Stock (StockID, StockDescription, StockQuantity, StockPrice)
VALUES ("21754", "HOME BUILDING BLOCK WORD", 1000, 5.95),
          ("22119", "PEACE WOODEN BLOCK LETTERS", 1000, 6.95),
```

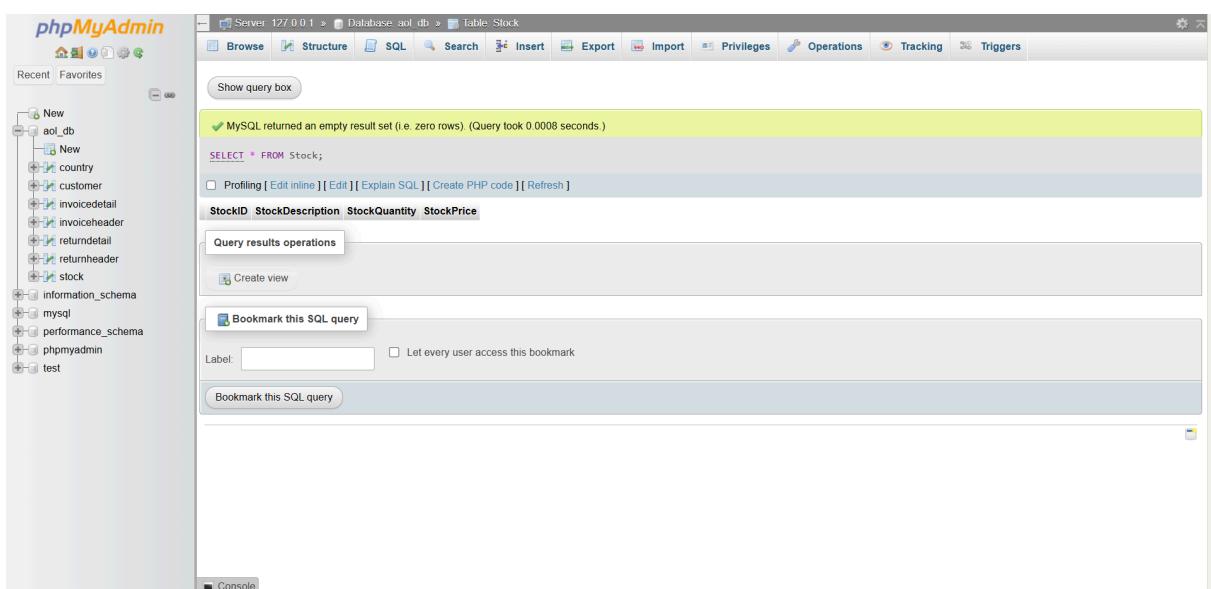
```

("21756","BATH BUILDING BLOCK WORD",1000,5.95),
("21360","JOY LARGE WOOD LETTERS",1000,9.95),
("20695","FLORAL BLUE MONSTER",1000,4.25),
.....
("90164A","PINK ROSEBUD PEARL BRACELET",1000,1.95),
("23695","DOILY THANK YOU CARD",1000,0.42),
("23070","EDWARDIAN HEART PHOTO FRAME",1000,4.15),
("23037","CANDLE HOLDER SILVER MADELINE",1000,3.29),
("23843","PAPER CRAFT , LITTLE BIRDIE",1000,2.08)

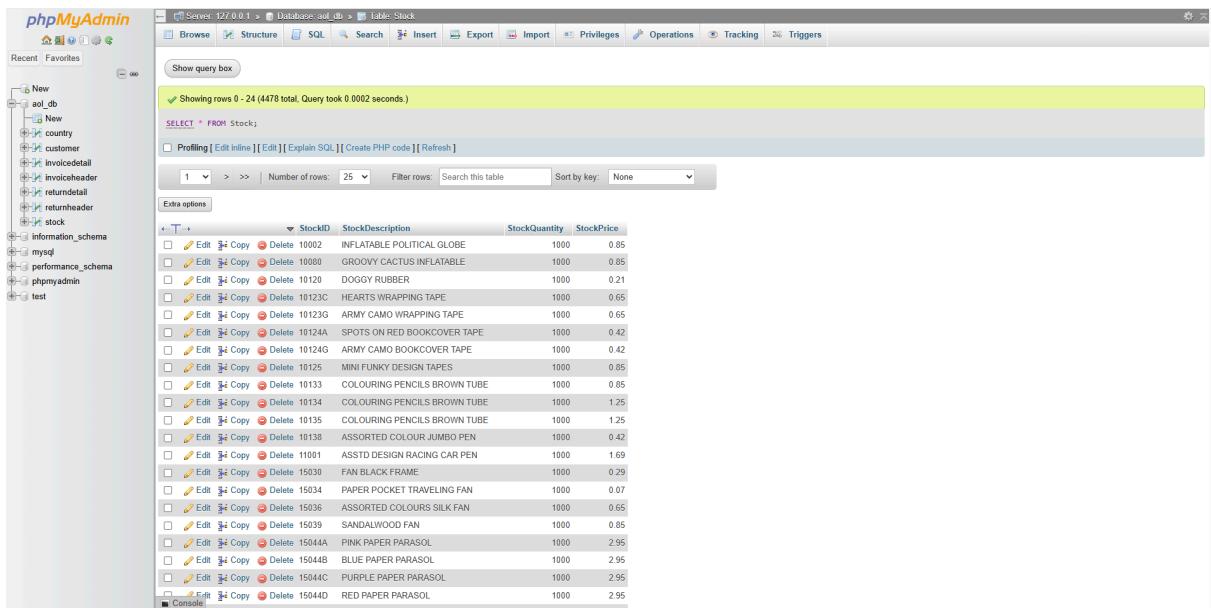
```

Notes: Dikarenakan ada total 4478 data stock yang berbeda, maka tampilan query untuk insertion untuk table stock diatas kami ringkas.

- Sebelum data dimasukkan ke tabel Stock



- Setelah data dimasukkan ke tabel Stock



The screenshot shows the phpMyAdmin interface with the 'Stock' table selected. The table has four columns: StockID, StockDescription, StockQuantity, and StockPrice. The data consists of 24 rows, each representing a different item in stock.

	StockID	StockDescription	StockQuantity	StockPrice
<input type="checkbox"/>	10002	INFLATABLE POLITICAL GLOBE	1000	0.85
<input type="checkbox"/>	10080	GROOVY CACTUS INFLATABLE	1000	0.85
<input type="checkbox"/>	10120	DOGGY RUBBER	1000	0.21
<input type="checkbox"/>	10123C	HEARTS WRAPPING TAPE	1000	0.65
<input type="checkbox"/>	10123G	ARMY CAMO WRAPPING TAPE	1000	0.65
<input type="checkbox"/>	10124A	SPOTS ON RED BOOKCOVER TAPE	1000	0.42
<input type="checkbox"/>	10124G	ARMY CAMO BOOKCOVER TAPE	1000	0.42
<input type="checkbox"/>	10125	MINI FUNKY DESIGN TAPES	1000	0.85
<input type="checkbox"/>	10133	COLOURING PENCILS BROWN TUBE	1000	0.85
<input type="checkbox"/>	10134	COLOURING PENCILS BROWN TUBE	1000	1.25
<input type="checkbox"/>	10135	COLOURING PENCILS BROWN TUBE	1000	1.25
<input type="checkbox"/>	10138	ASSORTED COLOUR JUMBO PEN	1000	0.42
<input type="checkbox"/>	11001	ASSTD DESIGN RACING CAR PEN	1000	1.69
<input type="checkbox"/>	15030	FAN BLACK FRAME	1000	0.29
<input type="checkbox"/>	15034	PAPER POCKET TRAVELING FAN	1000	0.07
<input type="checkbox"/>	15036	ASSORTED COLOURS SILK FAN	1000	0.65
<input type="checkbox"/>	15039	SANDALWOOD FAN	1000	0.85
<input type="checkbox"/>	15044A	PINK PAPER PARASOL	1000	2.95
<input type="checkbox"/>	15044B	BLUE PAPER PARASOL	1000	2.95
<input type="checkbox"/>	15044C	PURPLE PAPER PARASOL	1000	2.95
<input type="checkbox"/>	15044D	RED PAPER PARASOL	1000	2.95

#### d) Tabel ReturnHeader

Setelah melakukan filterisasi dengan menggunakan Python, didapatkan sejumlah list return, sehingga dari list tersebut bisa langsung dimasukkan ke dalam table ReturnHeader.

Insertion Query:

```
INSERT INTO ReturnHeader (ReturnID, ReturnDate)
VALUES ("C489449", "2009-12-01 10:33:00"),
("C489459", "2009-12-01 10:44:00"),
("489464", "2009-12-01 10:52:00"),
("489467", "2009-12-01 10:53:00"),
("C489476", "2009-12-01 10:55:00"),
.....
("C581466", "2011-12-08 19:20:00"),
("C581468", "2011-12-08 19:26:00"),
("C581470", "2011-12-08 19:28:00"),
("C581490", "2011-12-09 09:57:00"),
("C581568", "2011-12-09 11:57:00")
```

Notes: Dikarenakan ada total 9518 data ReturnHeader yang berbeda, maka tampilan query untuk insertion untuk table ReturnHeader di atas kami ringkas.

- Sebelum data dimasukkan ke dalam tabel ReturnHeader

The screenshot shows the phpMyAdmin interface for the 'aol\_db' database. The 'ReturnHeader' table is selected. The SQL query results show an empty set of rows. The table structure is defined by the columns 'ReturnID' and 'ReturnDate'.

- Setelah data dimasukkan ke dalam tabel ReturnHeader

The screenshot shows the 'ReturnHeader' table after data has been inserted. The table now contains 24 rows of data, each with a unique 'ReturnID' and a corresponding 'ReturnDate'. The data is listed in descending order of 'ReturnDate'.

	ReturnID	ReturnDate
1	489464	2009-12-01 10:52:00
2	489467	2009-12-01 10:53:00
3	489655	2009-12-01 17:26:00
4	489806	2009-12-02 12:42:00
5	489820	2009-12-02 13:23:00
6	489821	2009-12-02 13:25:00
7	489901	2009-12-03 09:47:00
8	490007	2009-12-03 12:09:00
9	490016	2009-12-03 12:30:00
10	490055	2009-12-03 13:22:00
11	490130	2009-12-03 18:28:00
12	490146	2009-12-04 09:29:00
13	490165	2009-12-04 11:32:00
14	490354	2009-12-04 16:39:00

### e) Tabel InvoiceHeader

Setelah melakukan filterisasi dengan menggunakan Python, didapatkan sejumlah list Invoice, sehingga dari list tersebut bisa langsung dimasukkan ke dalam table InvoiceHeader.

Insertion Query:

```
INSERT INTO InvoiceHeader (InvoiceID, InvoiceDate, CustomerID)
VALUES ("489434","2009-12-01 07:45:00",13085),
          ("489435","2009-12-01 07:46:00",13085),
```

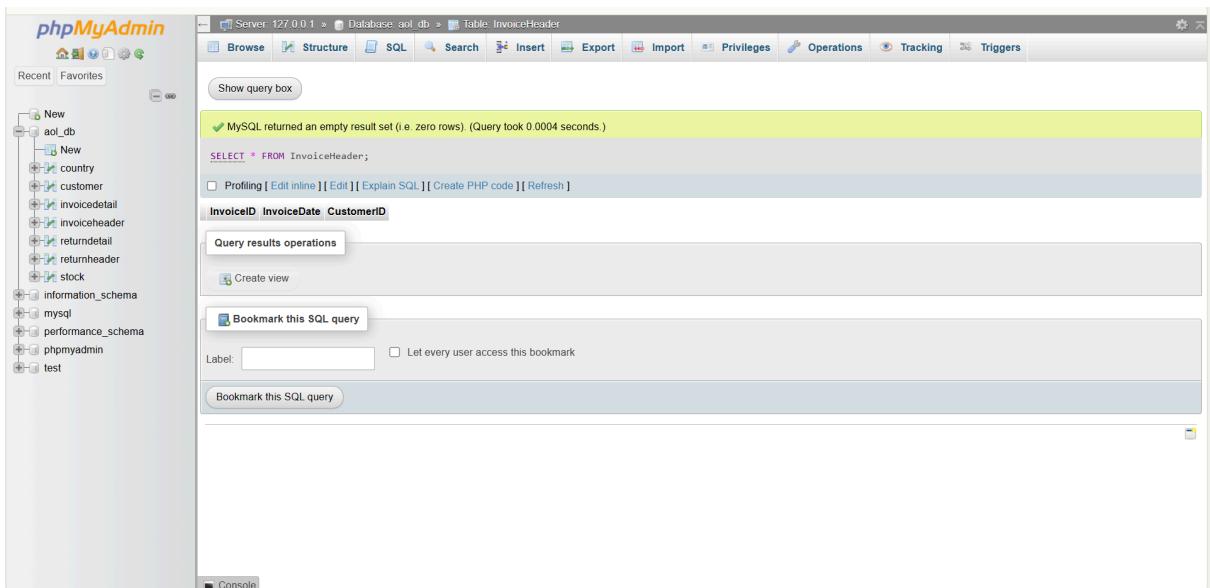
```

("489436","2009-12-01 09:06:00",13078),
("489437","2009-12-01 09:08:00",15362),
("489438","2009-12-01 09:24:00",18102),
.....
("581583","2011-12-09 12:23:00",13777),
("581584","2011-12-09 12:25:00",13777),
("581585","2011-12-09 12:31:00",15804),
("581586","2011-12-09 12:49:00",13113),
("581587","2011-12-09 12:50:00",12680)

```

Notes: Dikarenakan ada total 30900 data InvoiceHeader yang berbeda, maka tampilan query untuk insertion untuk table InvoiceHeader di atas kami ringkas.

- Sebelum data dimasukkan ke tabel InvoiceHeader



- Sesudah data dimasukkan ke tabel InvoiceHeader

The screenshot shows the phpMyAdmin interface for the 'aol\_db' database. The left sidebar lists tables: New, aol\_db (containing country, customer, invoicedetail, invoiceheader, returndetail, returnheader, stock), information\_schema, mysql, performance\_schema, phpmyadmin, and test. The main area is titled 'InvoiceHeader' and displays the results of the query 'SELECT \* FROM InvoiceHeader;'. The results table has columns: InvoiceID, InvoiceDate, and CustomerID. The data shows multiple rows of invoices, each with a unique ID, a specific date, and a corresponding customer ID.

	InvoiceID	InvoiceDate	CustomerID
<input type="checkbox"/>	489434	2009-12-01 07:45:00	13085
<input type="checkbox"/>	489435	2009-12-01 07:46:00	13085
<input type="checkbox"/>	489436	2009-12-01 09:06:00	13078
<input type="checkbox"/>	489437	2009-12-01 09:08:00	15362
<input type="checkbox"/>	489438	2009-12-01 09:24:00	18102
<input type="checkbox"/>	489439	2009-12-01 09:28:00	12682
<input type="checkbox"/>	489440	2009-12-01 09:43:00	18087
<input type="checkbox"/>	489441	2009-12-01 09:44:00	18087
<input type="checkbox"/>	489442	2009-12-01 09:46:00	13635
<input type="checkbox"/>	489443	2009-12-01 09:50:00	14110
<input type="checkbox"/>	489444	2009-12-01 09:55:00	12636
<input type="checkbox"/>	489445	2009-12-01 09:57:00	17519
<input type="checkbox"/>	489446	2009-12-01 10:06:00	13758
<input type="checkbox"/>	489447	2009-12-01 10:10:00	12362

#### f) Tabel InvoiceDetail

Setelah melakukan filterisasi dengan menggunakan Python, didapatkan sejumlah list InvoiceDetail, sehingga dari list tersebut bisa langsung dimasukkan ke dalam table InvoiceDetail.

Insertion Query:

```

INSERT INTO InvoiceDetail (InvoiceID, StockID, InvoiceQuantity)
VALUES ("489434","85048",12),
("489434", "22041",48),
("489434", "21232",24),
("489434", "22064",24),
("489434", "21871",24),
("489434", "21523",10),
.....
("581587", "22899",6),
("581587", "23254",4),
("581587", "23255",4),
("581587", "22138",3),
("581587", "POST",1),

```

Notes: Dikarenakan ada total 795711 data InvoiceDetail yang berbeda, maka tampilan query untuk insertion untuk table InvoiceDetail di atas kami ringkas.

Disini, kami tidak melakukan checking terhadap foreign key saat insert, tapi kami tetap melakukan deleting data yang id foreign key nya tidak ada di primary key:

## Deletion Query:

```
DELETE FROM InvoiceDetail  
WHERE InvoiceID NOT IN (SELECT InvoiceID FROM InvoiceHeader) OR StockID  
NOT IN (SELECT StockID FROM Stock);
```

- Sebelum data dimasukkan ke tabel InvoiceDetail

The screenshot shows the phpMyAdmin interface for the 'aol\_db' database. The left sidebar lists tables: New, aol\_db, New, country, customer, invoiceheader, invoicedetail, returndetail, returnheader, stock, information\_schema, mysql, performance\_schema, phpmyadmin, and test. The 'invoiceheader' table is selected. The main area displays the 'InvoiceDetail' table structure with columns: InvoiceID, StockID, and InvoiceQuantity. A query result window shows: MySQL returned an empty result set (i.e. zero rows). (Query took 0.0004 seconds.)

- Setelah data dimasukkan ke tabel InvoiceDetail

The screenshot shows the phpMyAdmin interface for the 'aol\_db' database. The left sidebar lists the same tables as before. The 'invoiceheader' table is selected. The main area displays the 'InvoiceDetail' table structure with columns: InvoiceID, StockID, and InvoiceQuantity. A query result window shows: Showing rows 0 - 24 (795711 total, Query took 0.0012 seconds.)

InvoiceID	StockID	InvoiceQuantity
489434	21232	24
489434	21523	10
489434	21871	24
489434	22041	48
489434	22064	24
489434	79323P	12
489434	79323W	12
489434	85048	12
489435	22195	24
489435	22349	12
489435	22350	12
489435	22353	12
489436	21181	12
489436	21333	6
489436	21754	3
489436	21755	18
489436	21756	3
489436	22107	4
489436	22109	16
489436	22111	24
489436	22119	3

### g) Tabel ReturnDetail

Setelah melakukan filterisasi dengan menggunakan Python, didapatkan sejumlah list ReturnDetail, sehingga dari list tersebut bisa langsung dimasukkan ke dalam table ReturnDetail.

Insertion Query:

```
INSERT INTO ReturnDetail (ReturnID, StockID, ReturnQuantity)
VALUES ("C489449","22087",-12),
("C489449","85206A",-6),
("C489449","21895",-4),
("C489449","21896",-6),
("C489449","22083",-12)
.....
("C581490","23144",-11),
("C581499","M",-1),
("C581568","21258",-5),
("C581569","84978",-1),
("C581569","20979",-5)
```

Notes: Dikarenakan ada total 16764 data ReturnDetail yang berbeda, maka tampilan query untuk insertion untuk table ReturnDetail di atas kami ringkas.

Disini, kami tidak melakukan checking terhadap foreign key saat insert, tapi kami tetap melakukan deleting data yang id foreign key nya tidak ada di primary key:

Deletion Query:

```
DELETE FROM ReturnDetail
WHERE ReturnID NOT IN (SELECT ReturnID FROM InvoiceHeader) OR StockID
NOT IN (SELECT StockID FROM Stock);
```

- Sebelum data dimasukkan ke dalam tabel ReturnDetail

The screenshot shows the phpMyAdmin interface for the 'aoi\_db' database. The left sidebar lists tables: 'New', 'aoi\_db' (containing 'New', 'country', 'customer', 'invoicedetail', 'invoiceheader', 'returndetail', 'returnheader', 'stock'), 'information\_schema', 'mysql', 'performance\_schema', 'phpmyadmin', and 'test'. The 'returndetail' table is selected. The top navigation bar includes 'Server 127.0.0.1', 'Database aoi\_db', 'Table returndetail', and tabs for 'Browse', 'Structure', 'SQL', 'Search', 'Insert', 'Export', 'Import', 'Privileges', 'Operations', 'Tracking', and 'Triggers'. A 'Show query box' button is visible. The main area displays a green success message: 'MySQL returned an empty result set (i.e. zero rows). (Query took 0.0004 seconds.)'. Below it is the SQL query: 'SELECT \* FROM returndetail;'. There are buttons for 'Profiling', 'Edit inline', 'Explain SQL', 'Create PHP code', and 'Refresh'. A 'ReturnID StockID ReturnQuantity' header row is shown above the data grid. The data grid has a single row with values: '1', '1', and '1'. Below the grid are 'Query results operations' (including 'Create view') and 'Bookmark this SQL query' buttons. A 'Label:' input field and a 'Let every user access this bookmark' checkbox are also present.

- Setelah data dimasukkan ke dalam tabel ReturnDetail

phpMyAdmin

Server: 127.0.0.1 Database: aol\_db Table: returndetail

Browse Structure SQL Search Insert Export Import Privileges Operations Tracking Triggers

Show query box

Showing rows 0 - 24 (16764 total, Query took 0.0002 seconds.)

SELECT \* FROM returndetail;

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	ReturnID	StockID	ReturnQuantity
<input type="checkbox"/>	489449	21871	-12
<input type="checkbox"/>	489449	21895	-4
<input type="checkbox"/>	489449	21896	-6
<input type="checkbox"/>	489449	22083	-12
<input type="checkbox"/>	489449	22087	-12
<input type="checkbox"/>	489449	22990	-12
<input type="checkbox"/>	489449	84946	-12
<input type="checkbox"/>	489449	84970S	-24
<input type="checkbox"/>	489449	85206A	-6
<input type="checkbox"/>	489459	90003B	-3
<input type="checkbox"/>	489459	90003C	-3
<input type="checkbox"/>	489459	90003D	-3
<input type="checkbox"/>	489459	90082D	-2
<input type="checkbox"/>	489459	90185B	-3
<input type="checkbox"/>	489459	90185C	-3
<input type="checkbox"/>	489459	90200A	-3
<input type="checkbox"/>	489459	90200B	-3
<input type="checkbox"/>	489459	90200C	-3
<input type="checkbox"/>	489459	90200D	-3
<input type="checkbox"/>	489459	90200E	-3
<input type="checkbox"/>	489459	90209A	-3

### 3. Data Transactions & Queries (DML)

Isi dari *query.sql*

```
# Number 1
INSERT INTO Stock (StockID, StockDescription, StockPrice)
VALUES ('42131A', 'GREEN SHOES', 7.95);

# Number 2
BEGIN;

INSERT INTO InvoiceHeader (InvoiceID, InvoiceDate, CustomerID)
VALUES ('56789', '2025-10-12 12:51:00', 17364);

INSERT INTO InvoiceDetail (InvoiceID, StockID, InvoiceQuantity)
VALUES ('56789', '22119', 10);

INSERT INTO InvoiceDetail (InvoiceID, StockID, InvoiceQuantity)
VALUES ('56789', '90003D', 5);

UPDATE Stock
SET StockQuantity = StockQuantity - 10
WHERE StockID = '22119';

UPDATE Stock
SET StockQuantity = StockQuantity - 5
WHERE StockID = '90003D';

COMMIT;

# Number 3
BEGIN;

INSERT INTO ReturnHeader
VALUES ('C54321', '2025-10-12');

INSERT INTO ReturnDetail
VALUES ('C54321', '90003D', -5);

UPDATE Stock
SET StockQuantity = StockQuantity + 5
WHERE StockID = '90003D';

COMMIT;

# Number 4
```

```

SELECT c.CustomerID, c.CountryID, SUM(p.StockPrice * i2.InvoiceQuantity)
AS total_spending
FROM Customer c
JOIN invoiceheader i1 ON i1.CustomerID = c.CustomerID
JOIN InvoiceDetail i2 ON i2.InvoiceID = i1.InvoiceID
JOIN Stock p ON p.StockID = i2.StockID
GROUP BY c.CustomerID, c.CountryID
ORDER BY total_spending DESC
LIMIT 10;

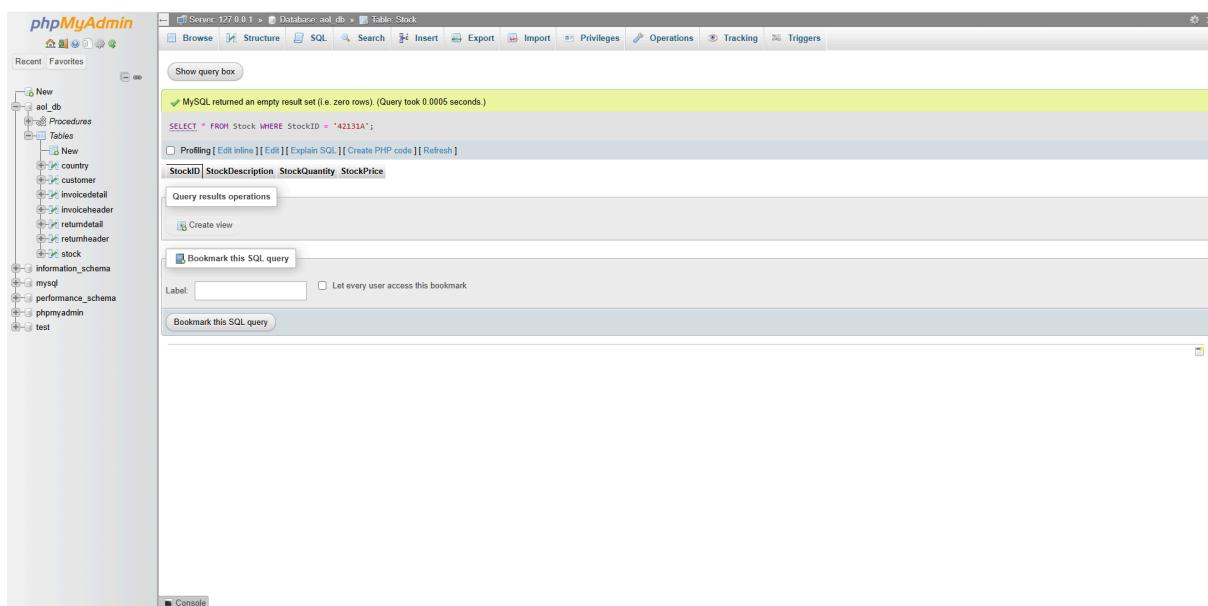
# Number 5
SELECT MONTH(i1.InvoiceDate) AS Month_Highest_Spending,
SUM(p.StockPrice * i2.InvoiceQuantity) AS Highest_spending
FROM Stock p
JOIN InvoiceDetail i2 ON p.StockID = i2.StockID
JOIN InvoiceHeader i1 ON i1.InvoiceID = i2.InvoiceID
WHERE YEAR(i1.InvoiceDate) = 2011
GROUP BY YEAR(i1.InvoiceDate), MONTH(i1.InvoiceDate)
ORDER BY Highest_spending DESC
LIMIT 1;

```

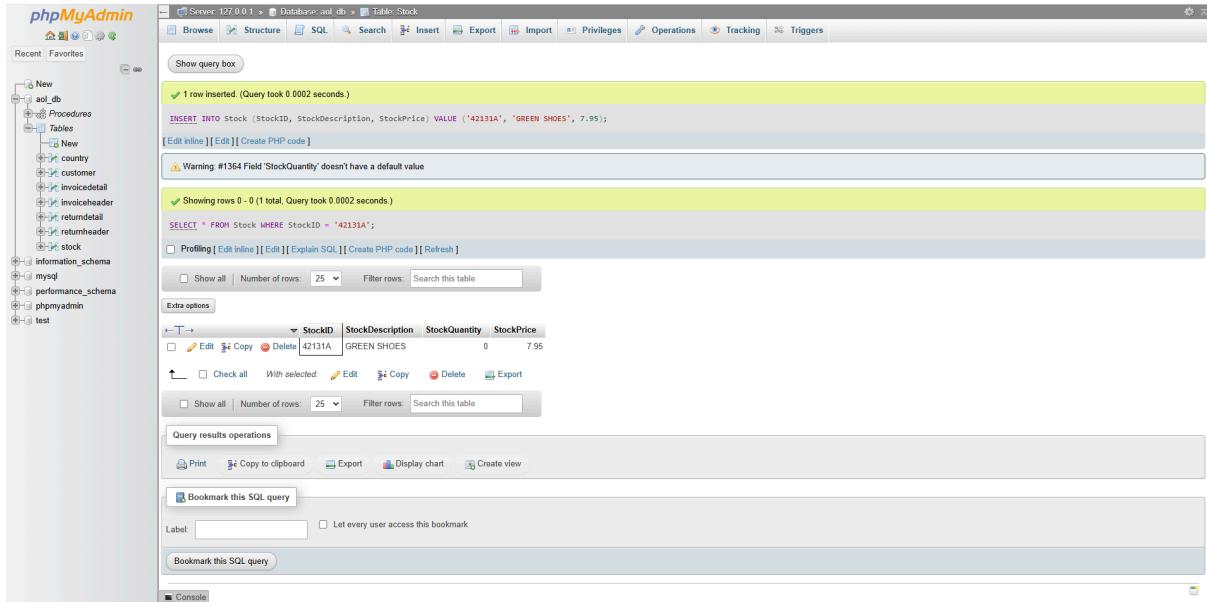
### Nomor 1:

Disini, kita buat skenario, bagaimana jika toko ingin menambahkan produk atau stock baru, kita menginput stok baru dengan id = ‘42131A’, deskripsi = ‘GREEN SHOES’, dan harga 7.25.

- Sebelum stock baru dimasukkan ke tabel Stock.



- Setelah stock baru dimasukkan ke tabel Stock.



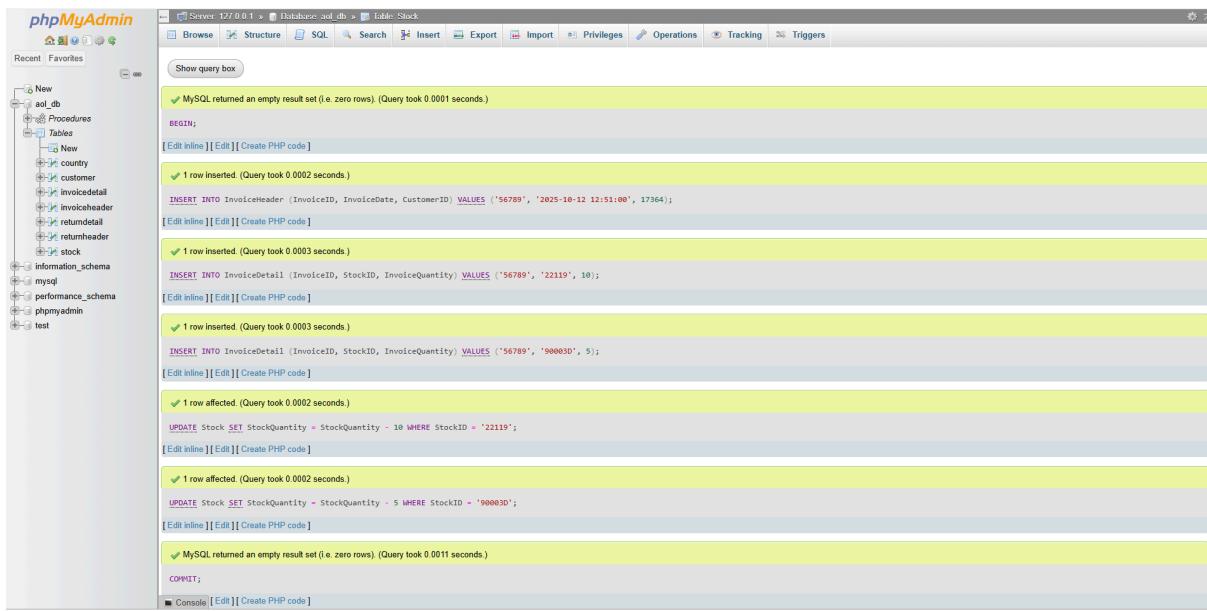
The screenshot shows the phpMyAdmin interface for the 'Stock' table in the 'aol\_db' database. A single row was inserted with the following values:

StockID	StockDescription	StockQuantity	StockPrice
42131A	GREEN SHOES	0	7.95

## Number 2:

Disini kita buat skenario, dimana jika customer dengan id 17364, harus melakukan transaksi dengan membeli 2 produk, dengan id produk masing masing : 22119 dan 90003D, dengan id invoice = 56789, pada tanggal 12-10-2025, pada jam 12:51:00.

- Setelah query transaksi dijalankan



The screenshot shows the phpMyAdmin interface with the following transaction log:

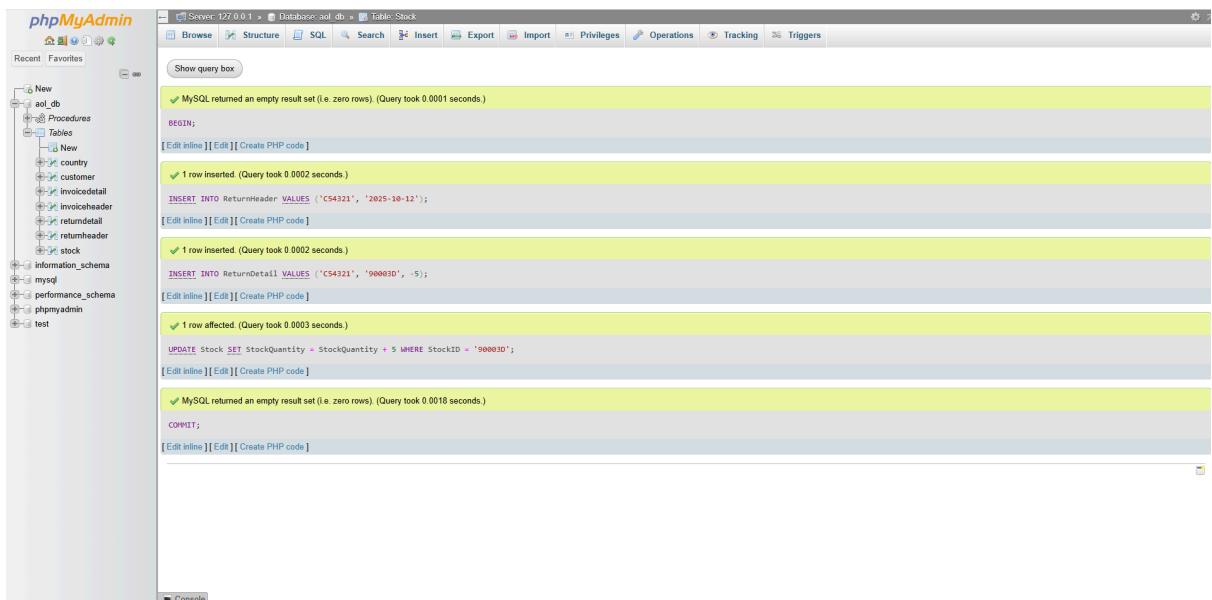
```

BEGIN;
INSERT INTO InvoiceHeader (InvoiceID, InvoiceDate, CustomerID) VALUES ('56789', '2025-10-12 12:51:00', 17364);
INSERT INTO InvoiceDetail (InvoiceID, StockID, InvoiceQuantity) VALUES ('56789', '22119', 10);
INSERT INTO InvoiceDetail (InvoiceID, StockID, InvoiceQuantity) VALUES ('56789', '90003D', 5);
UPDATE Stock SET StockQuantity = StockQuantity - 10 WHERE StockID = '22119';
UPDATE Stock SET StockQuantity = StockQuantity - 5 WHERE StockID = '90003D';
COMMIT;
  
```

### Number 3:

Disini, kita buat skenario, dimana pembeli ingin mengembalikan barang menggunakan transaksi dengan stockID 90003D, pada tanggal yang sama juga yaitu 2025-10-12, misalkan dia ingin mengembalikan sebanyak 5 buah barang, disini kita memang sengaja untuk setiap transaksi minus itu pakai returnId, jadi ga pakai InvoiceID, disini juga sudah diasumsikan bahwa customer itu sudah di validasi, sebelumnya dengan melakukan select, bahwa dia sudah melakukan order yang berkaitan dengan stockID 90003D, disini karena quantity value - nya sudah minus, maka kita hanya perlu melakukan operasi aritmatika yaitu tambah.

- Setelah query transaksi dijalankan



The screenshot shows the phpMyAdmin interface with the following details:

- Server:** 127.0.0.1 | **Database:** aol\_db | **Table:** Stock
- Recent Favorites:** (empty)
- Schemas:** information\_schema, mysql, performance\_schema, phpmyadmin, test
- Tables:** aol\_db (selected), Procedures, Tables, New, country, customer, invoicedetail, invoiceheader, returnedetail, returnheader, stock
- Operations:** Browse, Structure, SQL, Search, Insert, Export, Import, Privileges, Operations, Tracking, Triggers
- Query Results:**
  - MySQL returned an empty result set (i.e. zero rows). (Query took 0.0001 seconds.)
  - BEGIN;
  - 1 row inserted (Query took 0.0002 seconds.)  
INSERT INTO ReturnHeader VALUES ('C54321', '2025-10-12');
  - 1 row inserted (Query took 0.0002 seconds.)  
INSERT INTO ReturnDetail VALUES ('C54321', '90003D', -5);
  - 1 row affected (Query took 0.0003 seconds.)  
UPDATE Stock SET StockQuantity = StockQuantity + 5 WHERE StockID = '90003D';
  - 1 row inserted (Query took 0.0018 seconds.)  
COMMIT;

#### Number 4:

Disini, dalam mengambil top 10 customer berdasarkan total spending, kita menggunakan aggregate fungsi sum untuk mendapatkan total spending, berdasarkan price \* dengan quantity yang dibeli oleh customer, kita juga melakukan group by berdasarkan customerID dan CountryID, dan mengurutkannya secara desc, dan menggunakan keyword limit 10; untuk melimit output hanya ada 10.

- Setelah query select dijalankan

The screenshot shows the phpMyAdmin interface with the following details:

- Server:** 127.0.0.1 > Database: aol\_db > Table: Customers
- Query Results:**
  - Showing rows 0 - 9 (10 total). Query took 31.1661 seconds.
  - SQL Query:

```
SELECT c.CustomerID, c.CountryID, SUM(p.StockPrice * i2.InvoiceQuantity) AS total_spending FROM Customer c JOIN invoiceheader i1 ON i1.CustomerID = c.CustomerID JOIN InvoiceDetail i2 ON i2.InvoiceID = i1.InvoiceID JOIN Stock p ON p.StockID = i2.StockID GROUP BY c.CustomerID, c.CountryID ORDER BY total_spending DESC LIMIT 10;
```
  - Results Table:

CustomerID	CountryID	total_spending
18102	1	881750.58
14646	12	619891.78
13694	1	370063.03
14156	6	343060.48
14911	6	301669.53
13902	10	242783.92
17450	1	226078.20
17511	1	208714.59
16684	1	171118.30
16446	1	168472.50
- Operations:** Print, Copy to clipboard, Export, Display chart, Create view
- Bookmark:** Bookmark this SQL query (Label: [empty], Let every user access this bookmark)

## Number 5:

Dalam mengambil month dengan total spending tertinggi pada tahun 2011, kami menggunakan aggregate function sum berdasarkan price \* quantity yang dibeli oleh customer tersebut, dan menggunakan datefunction month, dan year, lalu di group berdasarkan month, dan di order secara desc, lalu di limit 1 untuk mencari yang total spendingnya tertinggi saja.

- Setelah query select dijalankan

The screenshot shows the phpMyAdmin interface with the following details:

- Server:** 127.0.0.1 > Database: ad\_db > Table: Stock
- Query Results:**
  - Showing rows 0 - 0 (1 total). Query took 1.9549 seconds.
  - SQL Query:

```
SELECT MONTH(l1.InvoiceDate) AS Month_Highest_Spending, SUM(p.StockPrice * l2.InvoiceQuantity) AS Highest_spending FROM Stock p JOIN InvoiceDetail l2 ON p.StockID = l2.StockID JOIN InvoiceHeader l1 ON l1.InvoiceID = l2.InvoiceID WHERE YEAR(l1.InvoiceDate) = 2011 GROUP BY YEAR(l1.InvoiceDate), MONTH(l1.InvoiceDate) ORDER BY Highest_spending DESC LIMIT 1;
```
  - Execution Options: Profiling, Edit inline, Explain SQL, Create PHP code, Refresh.
  - Table Headers: Month\_Highest\_Spending, Highest\_spending.
  - Data Row: 11 | 1267225.54
- Query Results Operations:** Print, Copy to clipboard, Export, Display chart, Create view.
- Bookmark:** Bookmark this SQL query, Label: (empty), Let every user access this bookmark.
- Console:** (bottom left)

#### 4. Advanced Features & Automation

Isi *advanced.sql*

```
# trigger
DELIMITER $$

CREATE TRIGGER Return_Stock
AFTER INSERT ON ReturnDetail
FOR EACH ROW
BEGIN
    UPDATE Stock
    SET StockQuantity = StockQuantity - NEW.ReturnQuantity
    WHERE StockID = NEW.StockID;
END $$

DELIMITER ;

DELIMITER $$

CREATE TRIGGER Sales_Stock
AFTER INSERT ON InvoiceDetail
FOR EACH ROW
BEGIN
    UPDATE Stock
    SET StockQuantity = StockQuantity - NEW.InvoiceQuantity
    WHERE StockID = NEW.StockID;
END $$

DELIMITER ;

# stored procedure
DELIMITER $$

CREATE PROCEDURE GetCustomerInvoiceHistory (
    p_CustomerID INT
)

BEGIN

    SELECT c.CustomerID, i1.InvoicesDate, SUM(s.StockPrice *
i2.InvoiceQuantity) as total_spending
    FROM Customer c
    JOIN InvoiceHeader i1 ON i1.CustomerID = c.CustomerID
    JOIN InvoiceDetail i2 ON i1.InvoiceID = i2.InvoiceID
    JOIN Stock s ON s.StockID = i2.StockID
    WHERE c.CustomerID = p_CustomerID
    GROUP BY c.CustomerID, i1.InvoiceDate;

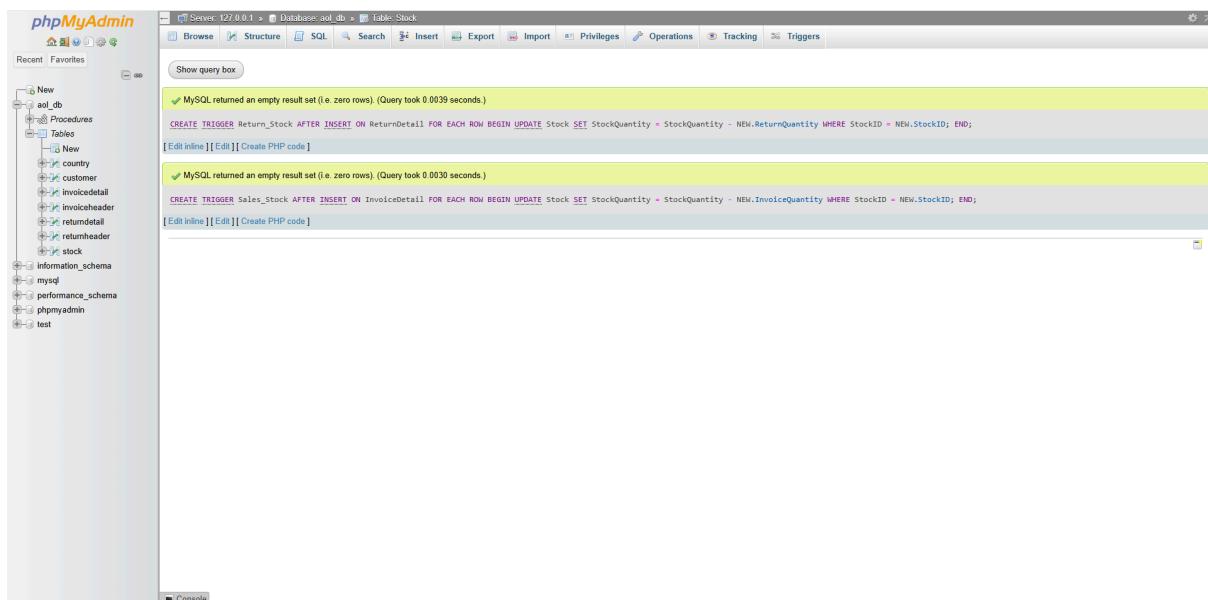
END$$
```

```
DELIMITER ;
```

### Trigger:

Trigger untuk return\_stock, pertama diganti dulu delimiternya menjadi \$\$, lalu untuk quantity stock sekarang kita kurang dengan quantity dari return (karena quantity return nya minus, otomatis dia akan menambah quantity di stock), lalu untuk sales\_stock itu sama, pertama diganti dulu delimiternya menjadi \$\$, lalu quantity stock sekarang kita kurang dengan quantity dari invoice (karena quantity invoice positif, maka quantity di stock akan berkurang)

- Setelah query trigger dijalankan.



## Procedure

Disini, kita buat procedure, dengan mengambil satu parameter yaitu customerID, kita akan display atau select customer id, invoice date, dan total\_spending, yang didapatkan dengan melakukan sum terhadap perkalian antara invoicequantity dan stockprice, lalu kita group berdasarkan customerID, dan invoiceDate nya, dan juga kita check hanya untuk customerId yang telah dimasukkan ke parameternya.

- Setelah query procedure dijalankan.

The screenshot shows the phpMyAdmin interface for a MySQL database named 'adl\_db'. The left sidebar lists tables: New, New, country, customer, invicedetail, invoiceheader, returndetail, returnheader, Stock. The main area shows a query results window with the following content:

```
MySQL returned an empty result set (i.e. zero rows). (Query took 0.0030 seconds.)  
CREATE PROCEDURE GetCustomerInvoiceHistory ( p_CustomerID INT ) BEGIN SELECT c.CustomerID, i1.InvoiceDate, SUM(s.StockPrice * i2.InvoiceQuantity) as total_spending FROM Customer c JOIN InvoiceHeader i1 ON i1.CustomerID = c.CustomerID JOIN InvoiceDetail i2 ON i1.InvoiceID = i2.InvoiceID JOIN Stock s ON s.StockID = i2.StockID WHERE c.CustomerID = p_CustomerID GROUP BY c.CustomerID, i1.InvoiceDate; END;
```

Below the results, there are buttons for 'Edit inline' and 'Create PHP code'.