

Database CA Report

Part 1- Conceptual Design

1. Introduction

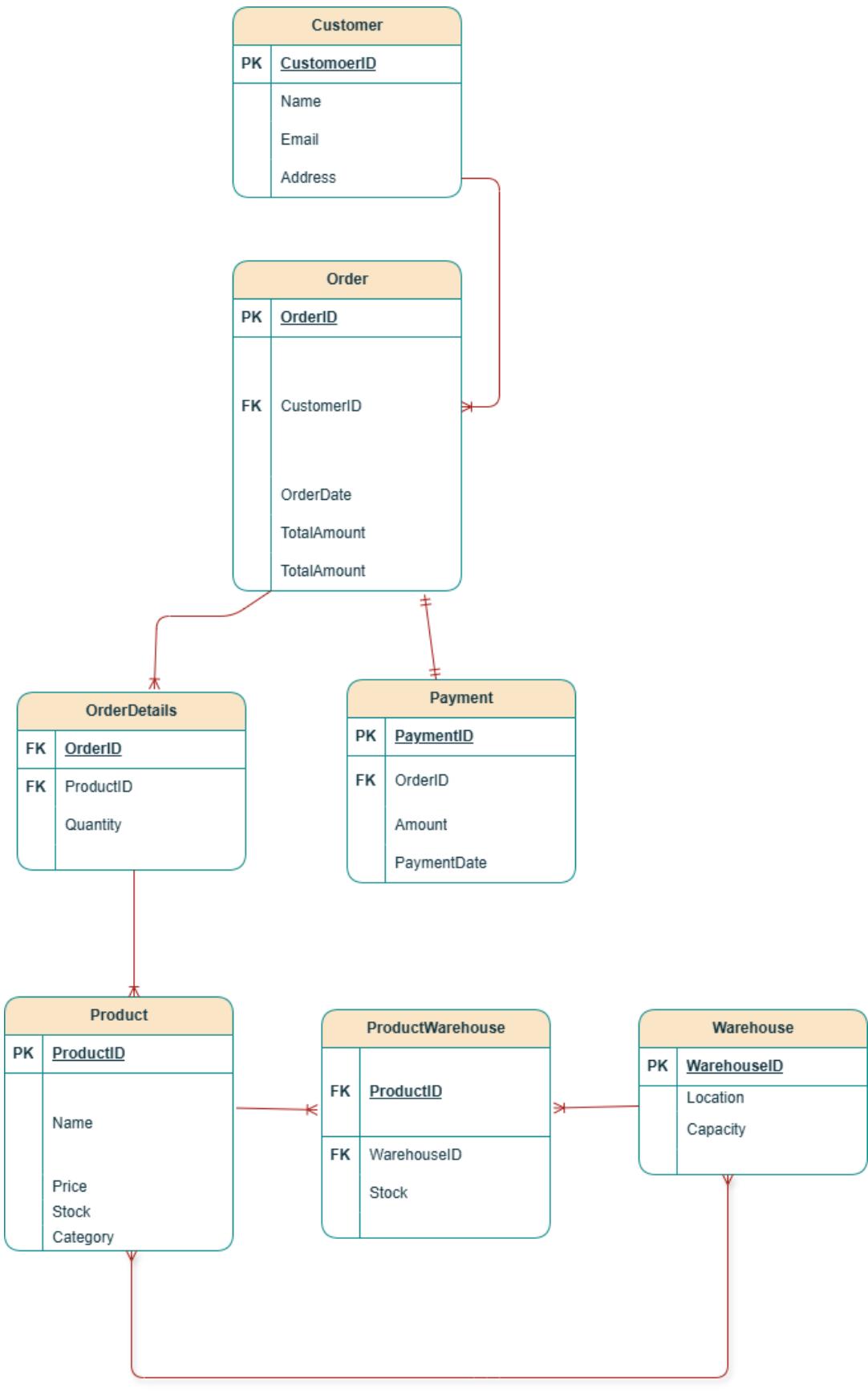
To demonstrate the use of databases, I wish to create a collection and launch a website for jewellery designs. This collection will include pieces like: earrings, necklaces, bracelets, and rings. These will be handcrafted, and the target audience will be women. It can be difficult to manage information such as orders, inventory etc. Therefore, building a database system to keep everything organised is important.

The information managed by the system is:

1. **Customer Information:** Store details like contact information, preferences, and purchase history.
2. **Product Management:** Classify jewellery items into categories (e.g., earrings, necklaces), along with information such as inventory levels, prices, and descriptions.
3. **Order and Payment Tracking:** Allow customers to order multiple items, enable inventory updates, and manage payment methods and statuses.
4. **Sales Insights:** Generate monthly and quarterly reports to highlight trends, like the most popular items, and aid decisions on inventory and marketing.
5. **Design Records:** Maintain a record of design concepts and production costs for each piece.

By implementing this database system the website can run smoothly. This setup will also free up more time for what truly matters i.e designing beautiful pieces that align with the customers' interests.

2. ER design



3. Conversion to Relational model

3.1 Introduction

The conceptual design of the online jewelry shop database was converted into a relational model to create normalized tables that support efficient data storage and retrieval. The relational model was designed to meet the requirements of Third Normal Form (3NF) to ensure minimal data redundancy and maintain data integrity. This section details the relational model and its SQL implementation.

3.2 Entities and Relationships

3.2.1. Customer Entity

- **Description:** Stores customer information for the jewelry shop.
 - **Attributes:**
 - CustomerID (PK) – Unique identifier for each customer.
 - Name – Name of the customer.
 - Email – Email address of the customer.
 - Address – Physical address of the customer.
 - **Relationships:**
 - **One-to-Many:** A customer can place multiple orders. (CustomerID is referenced in the Order table as a foreign key.)
-

3.2.2. Order Entity

- **Description:** Represents customer orders in the jewelry shop.
 - **Attributes:**
 - OrderID (PK) – Unique identifier for each order.
 - CustomerID (FK) – Links the order to a customer.
 - OrderDate – Date the order was placed.
 - TotalAmount – Total monetary value of the order.
 - **Relationships:**
 - **One-to-Many:** A customer can place multiple orders, but each order belongs to one customer.
 - **One-to-Many:** An order can have multiple line items, as represented in the OrderDetails table.
-

3.2.3. Product Entity

- **Description:** Stores details about the jewelry products.
 - **Attributes:**
 - ProductID (PK) – Unique identifier for each product.
 - Name – Name of the product.
 - Price – Price of the product.
 - Stock – Current stock level of the product.
 - Category – Category of the product (e.g., necklace, bracelet, ring).
 - **Relationships:**
 - **Many-to-Many:** Products are included in multiple orders, represented by the OrderDetails table.
 - **Many-to-Many:** Products are stored in multiple warehouses, represented by the ProductWarehouse table.
-

3.2.4. OrderDetails Entity

- **Description:** Serves as a junction table for the many-to-many relationship between Order and Product.
 - **Attributes:**
 - OrderID (FK) – Links to the Order table.
 - ProductID (FK) – Links to the Product table.
 - Quantity – Number of units of the product in the order.
 - **Relationships:**
 - **Many-to-One:** Each line item belongs to one order.
 - **Many-to-One:** Each line item references one product.
-

3.2.5. Payment Entity

- **Description:** Stores payment information for orders.
 - **Attributes:**
 - PaymentID (PK) – Unique identifier for each payment.
 - OrderID (FK) – Links the payment to an order.
 - Amount – Total payment amount.
 - PaymentDate – Date the payment was made.
 - **Relationships:**
 - **One-to-One:** Each order is associated with one payment.
-

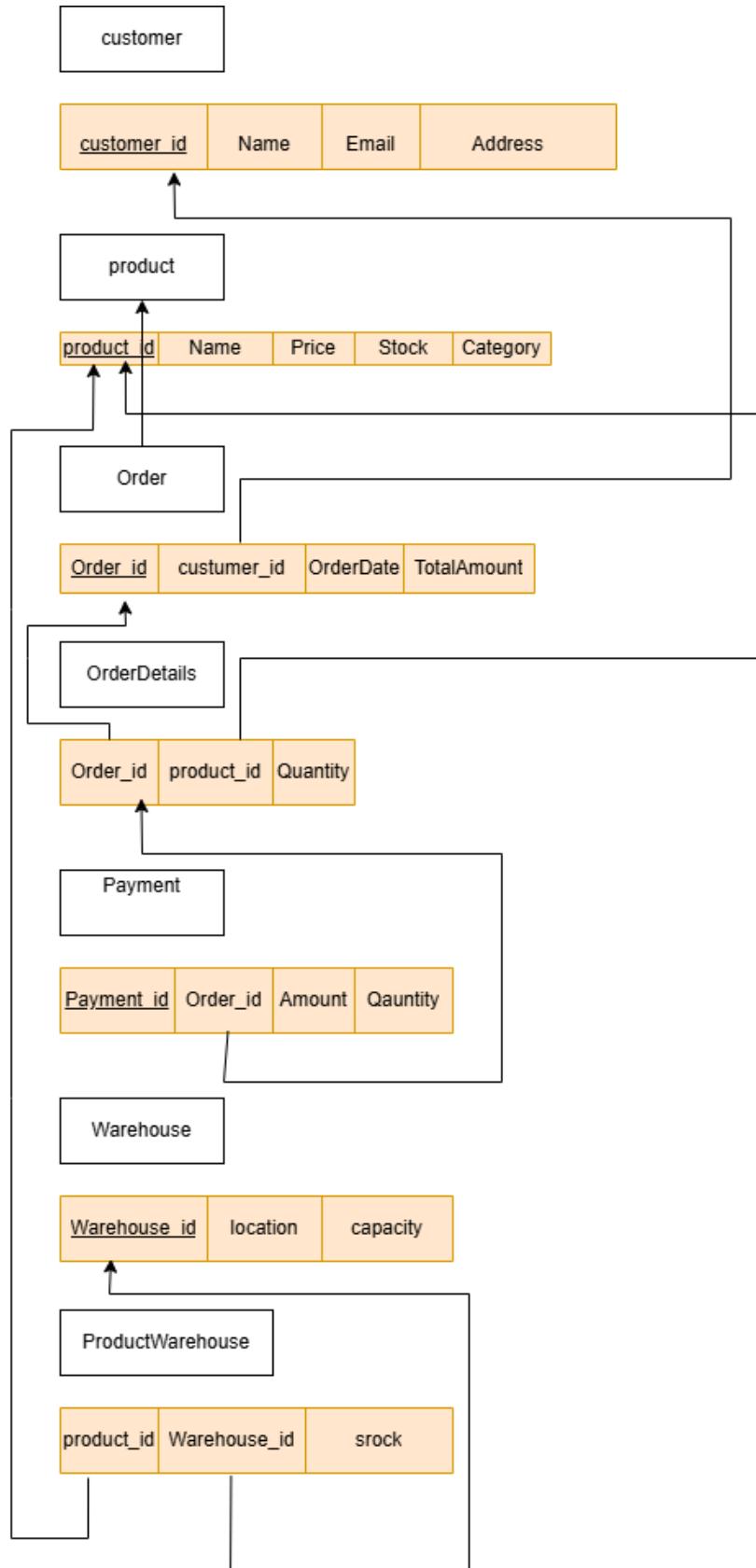
3.2.6. Warehouse Entity

- **Description:** Stores information about the physical locations where products are stored.
 - **Attributes:**
 - WarehouseID (PK) – Unique identifier for each warehouse.
 - Location – Physical location of the warehouse.
 - Capacity – Maximum storage capacity of the warehouse.
 - **Relationships:**
 - **Many-to-Many:** A warehouse can store multiple products, and a product can be stored in multiple warehouses. (ProductWarehouse table represents this relationship.)
-

3.2.7. ProductWarehouse Entity

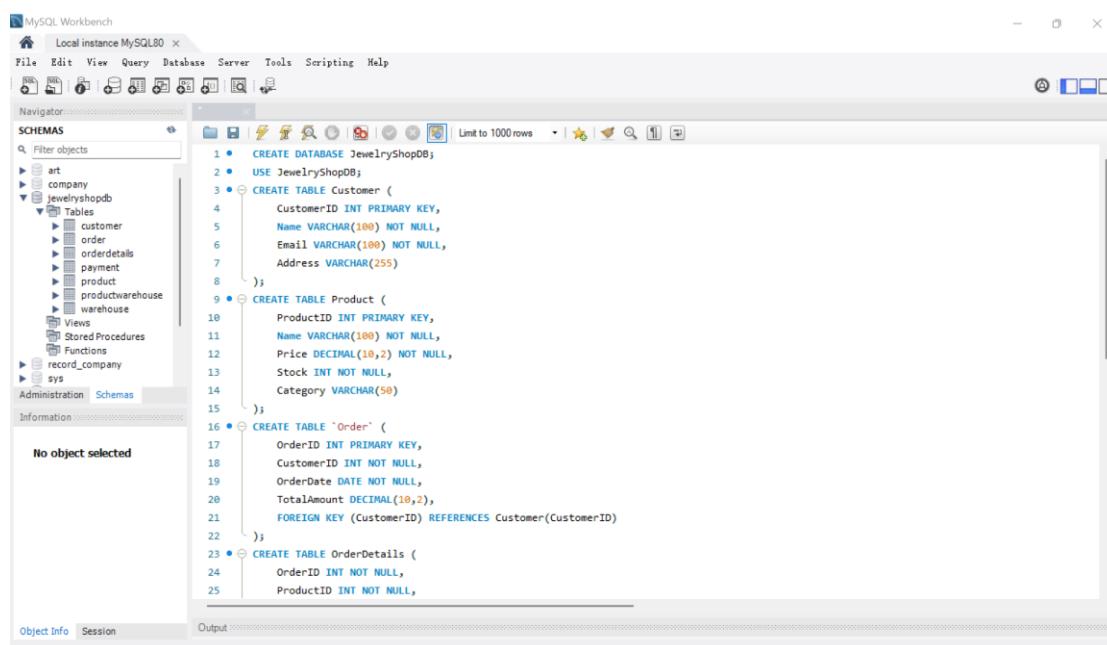
- **Description:** Serves as a junction table for the many-to-many relationship between Product and Warehouse.
- **Attributes:**
 - ProductID (FK) – Links to the Product table.
 - WarehouseID (FK) – Links to the Warehouse table.
 - Stock – Number of units of the product stored in the warehouse.
- **Relationships:**
 - **Many-to-One:** A product can be stored in multiple warehouses.
 - **Many-to-One:** A warehouse can store multiple products.

3.3 Relational Model



Part 2- Physical design

1. Create the corresponding database using DDL and Create all the necessary tables identified above using DDL



The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the database structure:
 - Schemas: Local instance MySQL80, JewelryShopDB (selected)
 - JewelryShopDB: Tables (customer, order, orderdetails, payment, product, productwarehouse, warehouse), Views, Stored Procedures, Functions, record_company, sys.
- SQL Editor:** Displays the SQL code for creating the database and its tables.

```
1 • CREATE DATABASE JewelryShopDB;
2 • USE JewelryShopDB;
3 • CREATE TABLE Customer (
4     CustomerID INT PRIMARY KEY,
5     Name VARCHAR(100) NOT NULL,
6     Email VARCHAR(100) NOT NULL,
7     Address VARCHAR(255)
8 );
9 • CREATE TABLE Product (
10    ProductID INT PRIMARY KEY,
11    Name VARCHAR(100) NOT NULL,
12    Price DECIMAL(10,2) NOT NULL,
13    Stock INT NOT NULL,
14    Category VARCHAR(50)
15 );
16 • CREATE TABLE `Order` (
17    OrderID INT PRIMARY KEY,
18    CustomerID INT NOT NULL,
19    OrderDate DATE NOT NULL,
20    TotalAmount DECIMAL(10,2),
21    FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)
22 );
23 • CREATE TABLE OrderDetails (
24    OrderID INT NOT NULL,
25    ProductID INT NOT NULL,
```
- Object Info:** Tab selected in the bottom left.
- Output:** Tab selected in the bottom right.

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator: SCHEMAS

Local instance MySQL80 ×

Limit to 1000 rows

Information

No object selected

```
22 );
23 • CREATE TABLE OrderDetails (
24     OrderID INT NOT NULL,
25     ProductID INT NOT NULL,
26     Quantity INT NOT NULL,
27     PRIMARY KEY (OrderID, ProductID),
28     FOREIGN KEY (OrderID) REFERENCES `Order` (OrderID),
29     FOREIGN KEY (ProductID) REFERENCES Product (ProductID)
30 );
31 • CREATE TABLE Payment (
32     PaymentID INT PRIMARY KEY,
33     OrderID INT NOT NULL,
34     Amount DECIMAL(10,2) NOT NULL,
35     PaymentDate DATE NOT NULL,
36     FOREIGN KEY (OrderID) REFERENCES `Order` (OrderID)
37 );
38 • CREATE TABLE Warehouse (
39     WarehouseID INT PRIMARY KEY,
40     Location VARCHAR(100) NOT NULL,
41     Capacity INT NOT NULL
42 );
43 • CREATE TABLE ProductWarehouse (
44     ProductID INT NOT NULL,
45     WarehouseID INT NOT NULL,
46     Stock INT NOT NULL,
47     PRIMARY KEY (ProductID, WarehouseID),
48     FOREIGN KEY (ProductID) REFERENCES Product (ProductID),
49     FOREIGN KEY (WarehouseID) REFERENCES Warehouse (WarehouseID)
50 );
51
```

Object Info Session Output

SQL Editor Opened.

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator: SCHEMAS

Local instance MySQL80 ×

Limit to 1000 rows

Information

No object selected

```
28 );
29 FOREIGN KEY (OrderID) REFERENCES `Order` (OrderID),
30 FOREIGN KEY (ProductID) REFERENCES Product (ProductID)
31 );
32 CREATE TABLE Payment (
33     PaymentID INT PRIMARY KEY,
34     OrderID INT NOT NULL,
35     Amount DECIMAL(10,2) NOT NULL,
36     PaymentDate DATE NOT NULL,
37     FOREIGN KEY (OrderID) REFERENCES `Order` (OrderID)
38 );
39 CREATE TABLE Warehouse (
40     WarehouseID INT PRIMARY KEY,
41     Location VARCHAR(100) NOT NULL,
42     Capacity INT NOT NULL
43 );
44 CREATE TABLE ProductWarehouse (
45     ProductID INT NOT NULL,
46     WarehouseID INT NOT NULL,
47     Stock INT NOT NULL,
48     PRIMARY KEY (ProductID, WarehouseID),
49     FOREIGN KEY (ProductID) REFERENCES Product (ProductID),
50     FOREIGN KEY (WarehouseID) REFERENCES Warehouse (WarehouseID)
51
```

Object Info Session Output

SQL Editor Opened.

2. Populate Tables Using DML

2.1 insert statements in customer table

The screenshot shows the MySQL Workbench interface with the 'customer' table selected in the 'customer' schema. The code pane contains the following SQL:

```
CREATE TABLE Warehouse (
    WarehouseID INT PRIMARY KEY,
    Location VARCHAR(100) NOT NULL,
    Capacity INT NOT NULL
);

CREATE TABLE ProductWarehouse (
    ProductID INT NOT NULL,
    WarehouseID INT NOT NULL,
    Stock INT NOT NULL,
    PRIMARY KEY (ProductID, WarehouseID),
    FOREIGN KEY (ProductID) REFERENCES Product(ProductID),
    FOREIGN KEY (WarehouseID) REFERENCES Warehouse(WarehouseID)
);

INSERT INTO Customer (Name, Email, Address)
VALUES
('Alice Johnson', 'alice.johnson@example.com', '123 Main St'),
('Bob Smith', 'bob.smith@example.com', '456 Elm St'),
('Cathy Brown', 'cathy.brown@example.com', '789 Oak St');
```

The output pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
31	18:02:51	CREATE TABLE Warehouse (WarehouseID INT PRIMARY KEY, Location V... 0 row(s) affected		0.000 sec
32	18:02:51	CREATE TABLE ProductWarehouse (ProductID INT NOT NULL, Warehou... 0 row(s) affected		0.015 sec
33	18:03:46	INSERT INTO Customer (Name, Email, Address) VALUES ('Alice Johnson', 'alice.jo... 3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0		0.016 sec
34	18:03:56	SELECT * FROM jewelryshopdb.customer LIMIT 0, 50000	3 row(s) returned	0.000 sec / 0.000 sec

The screenshot shows the MySQL Workbench interface with the 'customer' table selected in the 'customer' schema. The code pane contains the following SQL:

```
SELECT * FROM jewelryshopdb.customer;
```

The results grid shows the data inserted earlier:

CustomerID	Name	Email	Address
1	Alice Johnson	alice.johnson@example.com	123 Main St
2	Bob Smith	bob.smith@example.com	456 Elm St
3	Cathy Brown	cathy.brown@example.com	789 Oak St
*	NULL	NULL	NULL

The output pane shows the execution log, which is identical to the one in the previous screenshot.

2.2 insert statement in product table

The screenshot shows the MySQL Workbench interface with the 'JewelryShopDB' schema selected. In the SQL editor tab, the following SQL script is displayed:

```
CREATE TABLE Warehouse (
    WarehouseID INT AUTO_INCREMENT PRIMARY KEY,
    Location VARCHAR(100) NOT NULL,
    Capacity INT NOT NULL
);

CREATE TABLE ProductWarehouse (
    ProductID INT NOT NULL,
    WarehouseID INT NOT NULL,
    Stock INT NOT NULL,
    PRIMARY KEY (ProductID, WarehouseID),
    FOREIGN KEY (ProductID) REFERENCES Product(ProductID),
    FOREIGN KEY (WarehouseID) REFERENCES Warehouse(WarehouseID)
);

INSERT INTO Product (Name, Price, Stock, Category)
VALUES
    ('Gold Necklace', 199.99, 50, 'Necklace'),
    ('Silver Bracelet', 99.99, 100, 'Bracelet'),
    ('Diamond Ring', 499.99, 20, 'Ring');
```

The Output tab shows the execution log:

#	Time	Action	Message	Duration / Fetch
48	18:11:29	CREATE TABLE Payment (PaymentID INT AUTO_INCREMENT PRIMARY KEY,	0 row(s) affected	0.016 sec
49	18:11:29	CREATE TABLE Warehouse (WarehouseID INT AUTO_INCREMENT PRIMAR...	0 row(s) affected	0.016 sec
50	18:11:29	CREATE TABLE ProductWarehouse (ProductID INT NOT NULL, Warehou...	0 row(s) affected	0.016 sec
51	18:12:04	INSERT INTO Product (Name, Price, Stock, Category) VALUES ('Gold Neckla...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.000 sec

The screenshot shows the MySQL Workbench interface with the 'JewelryShopDB' schema selected. In the SQL editor tab, the following SQL query is displayed:

```
SELECT * FROM jewelryshopdb.product;
```

The Results Grid shows the data from the 'product' table:

ProductID	Name	Price	Stock	Category
1	Gold Necklace	199.99	50	Necklace
2	Silver Bracelet	99.99	100	Bracelet
3	Diamond Ring	499.99	20	Ring
NULL	NULL	NULL	NULL	NULL

The Output tab shows the execution log:

#	Time	Action	Message	Duration / Fetch
49	18:11:29	CREATE TABLE Warehouse (WarehouseID INT AUTO_INCREMENT PRIMAR...	0 row(s) affected	0.016 sec
50	18:11:29	CREATE TABLE ProductWarehouse (ProductID INT NOT NULL, Warehou...	0 row(s) affected	0.016 sec
51	18:12:04	INSERT INTO Product (Name, Price, Stock, Category) VALUES ('Gold Neckla...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.000 sec
52	18:12:36	SELECT * FROM jewelryshopdb.product LIMIT 0.50000	3 row(s) returned	0.000 sec / 0.000 sec

2.3 insert statements in warehouse table

The screenshot shows the MySQL Workbench interface with the 'JewelryShopDB' database selected. In the 'Schemas' tree, the 'Tables' node is expanded, showing the 'ProductWarehouse' table. The 'Query' tab contains the following SQL code:

```
44 • CREATE TABLE ProductWarehouse (
45     ProductID INT NOT NULL,
46     WarehouseID INT NOT NULL,
47     Stock INT NOT NULL,
48     PRIMARY KEY (ProductID, WarehouseID),
49     FOREIGN KEY (ProductID) REFERENCES Product(ProductID),
50     FOREIGN KEY (WarehouseID) REFERENCES Warehouse(WarehouseID)
51 );
52 • INSERT INTO Product (Name, Price, Stock, Category)
53     VALUES
54     ('Gold Necklace', 199.99, 50, 'Necklace'),
55     ('Silver Bracelet', 99.99, 100, 'Bracelet'),
56     ('Diamond Ring', 499.99, 20, 'Ring');
57
58 • INSERT INTO Warehouse (Location, Capacity)
59     VALUES
60     ('Dublin', 1000),
61     ('Cork', 800),
62     ('Galway', 500);
```

The 'Output' tab shows the execution results:

#	Time	Action	Message	Duration / Fetch
50	18:11:29	CREATE TABLE ProductWarehouse (ProductID INT NOT NULL, Warehou...	0 row(s) affected	0.016 sec
51	18:12:04	INSERT INTO Product (Name, Price, Stock, Category) VALUES ('Gold Neckla...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.000 sec
52	18:12:36	SELECT * FROM JewelryShopDB.product LIMIT 0,50000	3 row(s) returned	0.000 sec / 0.000 sec
53	18:14:14	INSERT INTO Warehouse (Location, Capacity) VALUES ('Dublin', 1000), ('Cork', 8...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.000 sec

The screenshot shows the MySQL Workbench interface with the 'JewelryShopDB' database selected. In the 'Schemas' tree, the 'Tables' node is expanded, showing the 'warehouse' table. The 'Query' tab contains the following SQL code:

```
1 • SELECT * FROM jewelryshopdb.warehouse;
```

The 'Result Grid' shows the following data:

WarehouseID	Location	Capacity
1	Dublin	1000
2	Cork	800
3	Galway	500
*	NULL	NULL

The 'Output' tab shows the execution results:

#	Time	Action	Message	Duration / Fetch
51	18:12:04	INSERT INTO Product (Name, Price, Stock, Category) VALUES ('Gold Neckla...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.000 sec
52	18:12:36	SELECT * FROM JewelryShopDB.product LIMIT 0,50000	3 row(s) returned	0.000 sec / 0.000 sec
53	18:14:14	INSERT INTO Warehouse (Location, Capacity) VALUES ('Dublin', 1000), ('Cork', 8...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.000 sec
54	18:15:45	SELECT * FROM jewelryshopdb.warehouse LIMIT 0,50000	3 row(s) returned	0.000 sec / 0.000 sec

3. Populate database with a large data set representing a one-year transaction

3.1 import customer data

The screenshot shows the MySQL Workbench interface with the 'customer' table selected in the 'customer' schema. The data grid displays 1000 rows of customer information, including CustomerID, Name, Email, and Address. The 'Action Output' pane shows the execution history of the query.

CustomerID	Name	Email	Address
992	Rickard Buddles	rbuddles@spotify.com	24 Declaration Road
993	Humphred Pixton	hpixtonr@51.ln	4759 Jana Crossing
994	Baudoin Pattini	bpattini@bay.co.uk	789 Forest Run Trail
995	Chlo Fansy	cfansym@reference.com	14 Service Street
996	Connie Oryx	coryxn@eepurl.com	1677 Tony Park
997	Warde Peltzer	wpelzer0@privacy.gov.au	5 Gina Way
998	Catharina Bellon	cbellonp@twipic.com	23 Merrick Road
999	Vonn Longrigg	vlongrigg@princeton.edu	03 West Crossing
1000	Corale Dyke	cdyke@krea.com	59 American Circle

3.2 import product data

The screenshot shows the MySQL Workbench interface with the 'product' table selected in the 'customer' schema. The data grid displays 5 products with columns: ProductID, Name, Price, Stock, and Category. The 'Action Output' pane shows the execution history of the query.

ProductID	Name	Price	Stock	Category
97	Gold Necklace	341.17	187	Necklace
98	Pearl Bracelet	316.43	86	Pendant
99	Diamond Ring	322.93	478	Ring
100	Gemstone Earrings	365.64	701	Bracelet

3.3 import order data

The screenshot shows the MySQL Workbench interface with the 'JewelryShopDB' database selected. In the top query editor, the following SQL query is run:

```
SELECT * FROM jewelryshopdb.order;
```

The results are displayed in a grid:

OrderID	CustomerID	OrderDate	TotalAmount
997	817	2024-04-01	266.83
998	779	2024-11-09	317.60
999	596	2024-04-30	315.64
1000	874	2024-06-19	419.11
NULL	NULL	NULL	NULL

Below the grid, the 'Action Output' pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
377	23:31:10	SELECT * FROM jewelryshopdb.customer LIMIT 0, 50000	1000 row(s) returned	0.000 sec / 0.000 sec
378	23:32:05	SELECT * FROM jewelryshopdb.customer LIMIT 0, 50000	1000 row(s) returned	0.000 sec / 0.000 sec
379	23:34:02	SELECT * FROM jewelryshopdb.product LIMIT 0, 50000	100 row(s) returned	0.000 sec / 0.000 sec
380	23:36:30	SELECT * FROM jewelryshopdb.order LIMIT 0, 50000	1000 row(s) returned	0.000 sec / 0.000 sec

3.4 import orderdetails data

The screenshot shows the MySQL Workbench interface with the 'JewelryShopDB' database selected. In the top query editor, the following SQL query is run:

```
SELECT * FROM jewelryshopdb.orderdetails;
```

The results are displayed in a grid:

OrderID	ProductID	Quantity
997	24	10
998	53	5
999	6	2
1000	33	8
NULL	NULL	NULL

Below the grid, the 'Action Output' pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
378	23:32:05	SELECT * FROM jewelryshopdb.customer LIMIT 0, 50000	1000 row(s) returned	0.000 sec / 0.000 sec
379	23:34:02	SELECT * FROM jewelryshopdb.product LIMIT 0, 50000	100 row(s) returned	0.000 sec / 0.000 sec
380	23:36:30	SELECT * FROM jewelryshopdb.order LIMIT 0, 50000	1000 row(s) returned	0.000 sec / 0.000 sec
381	23:37:35	SELECT * FROM jewelryshopdb.orderdetails LIMIT 0, 50000	1000 row(s) returned	0.000 sec / 0.000 sec

3.5 import payment data

The screenshot shows the MySQL Workbench interface with the 'payment' table selected in the 'JewelryShopDB' database. The 'Result Grid' pane displays the following data:

PaymentID	OrderID	Amount	PaymentDate
994	347	765.35	2024-06-14
995	945	434.89	2024-06-02
996	152	493.54	2024-05-26
997	206	727.69	2024-12-06
998	454	777.35	2024-03-10

The 'Action Output' pane shows the following log entries:

#	Time	Action	Message	Duration / Fetch
379	23:34:02	SELECT * FROM jewelryshopdb.product LIMIT 0, 50000	100 row(s) returned	0.000 sec / 0.000 sec
380	23:36:30	SELECT * FROM jewelryshopdb.order LIMIT 0, 50000	1000 row(s) returned	0.000 sec / 0.000 sec
381	23:37:35	SELECT * FROM jewelryshopdb.orderdetails LIMIT 0, 50000	1000 row(s) returned	0.000 sec / 0.000 sec
382	23:38:06	SELECT * FROM jewelryshopdb.payment LIMIT 0, 50000	1000 row(s) returned	0.000 sec / 0.000 sec

3.6 import warehouse data

The screenshot shows the MySQL Workbench interface with the 'warehouse' table selected in the 'JewelryShopDB' database. The 'Result Grid' pane displays the following data:

WarehouseID	Location	Capacity
7	Vitarte	2366
8	Tampere	676
9	Soeng Sang	1851
10	Kayun	3718
11	Mulu	None

The 'Action Output' pane shows the following log entries:

#	Time	Action	Message	Duration / Fetch
380	23:36:30	SELECT * FROM jewelryshopdb.order LIMIT 0, 50000	1000 row(s) returned	0.000 sec / 0.000 sec
381	23:37:35	SELECT * FROM jewelryshopdb.orderdetails LIMIT 0, 50000	1000 row(s) returned	0.000 sec / 0.000 sec
382	23:38:06	SELECT * FROM jewelryshopdb.payment LIMIT 0, 50000	1000 row(s) returned	0.000 sec / 0.000 sec
383	23:38:44	SELECT * FROM jewelryshopdb.warehouse LIMIT 0, 50000	10 row(s) returned	0.000 sec / 0.000 sec

3.7 import productwarehouse data

The screenshot shows the MySQL Workbench interface with the 'productwarehouse' tab selected. In the results grid, there are five rows of data:

ProductID	WarehouseID	Stock
1	1	367
1	2	284
1	6	96
1	7	354
1	10	466

The execution history shows four SELECT statements:

#	Time	Action	Message	Duration / Fetch
381	23:37:35	SELECT * FROM jewelryshopdb.orderdetails LIMIT 0, 50000	1000 row(s) returned	0.000 sec / 0.000 sec
382	23:38:06	SELECT * FROM jewelryshopdb.payment LIMIT 0, 50000	1000 row(s) returned	0.000 sec / 0.000 sec
383	23:38:44	SELECT * FROM jewelryshopdb.warehouse LIMIT 0, 50000	10 row(s) returned	0.000 sec / 0.000 sec
384	23:39:17	SELECT * FROM jewelryshopdb.productwarehouse LIMIT 0, 50000	37 row(s) returned	0.000 sec / 0.000 sec

Part 3- SQL statements

1. update entry

```
74      -- Part3_Q1 --
75 •  SET SQL_SAFE_UPDATES = 0;
76 •  ALTER TABLE Product
77      MODIFY COLUMN Price DECIMAL(10,2);
78 •  UPDATE Product
79      SET Price = Price * 1.05;
```

MySQL Workbench

Local instance MySQL80

Schemas: jewelryshopdb

Tables: customer, order, orderdetails, payment, product, productwarehouse, warehouse

Query:

```
1 • SELECT * FROM jewelryshopdb.product;
```

Result Grid:

ProductID	Name	Price	Stock	Category
97	Gold Necklace	358.23	187	Necklace
98	Pearl Bracelet	332.25	86	Pendant
99	Diamond Ring	339.08	478	Ring
100	Gemstone Earrings	383.92	701	Bracelet

2. delete entry

```
80      -- Part3 Q2 --
81 • ALTER TABLE Product
82     DROP COLUMN Stock;
83
```

MySQL Workbench

Local instance MySQL80

Schemas: jewelryshopdb

Tables: customer, order, orderdetails, payment, product, productwarehouse, warehouse

Query:

```
64 ('Silver Bracelet', 99.99, 100, 'Bracelet'),
65 ('Diamond Ring', 499.99, 20, 'Ring')
66
67 • INSERT INTO Warehouse (Location, Capacity)
68   VALUES
69   ('Dublin', 1000),
70   ('Cork', 800),
71   ('Galway', 500);
72
73 -- Part3 Q1 --
74 • SET SQL_SAFE_UPDATES = 0;
75 • ALTER TABLE Product
76   MODIFY COLUMN Price DECIMAL(10,2);
77 • UPDATE Product
78   SET Price = Price * 1.05;
79 -- Part3 Q2 --
80 • ALTER TABLE Product
81   DROP COLUMN Stock;
```

Action Output:

#	Time	Action	Message	Duration / Fetch
389	23:46:45	CREATE DATABASE JewelryShopDB	Error Code: 1007. Can't create database 'jewelryshopdb'; database exists	0.000 sec
390	23:47:00	SELECT * FROM jewelryshopdb.product LIMIT 0, 50000	100 row(s) returned	0.000 sec / 0.000 sec
391	23:50:28	ALTER TABLE Product DROP COLUMN Stock	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.016 sec
392	23:50:35	SELECT * FROM jewelryshopdb.product LIMIT 0, 50000	100 row(s) returned	0.000 sec / 0.000 sec

MySQL Workbench

Schemas: jewelryshopdb

Tables: customer, order, orderdetails, payment, product, productwarehouse, warehouse

Result Grid:

ProductID	Name	Price	Category
4	Diamond Ring	414.12	Ring
5	Diamond Ring	134.19	Necklace
6	Crystal Pendant Handmade Bracelet Platinum Ring	334.65	Earrings
7	Silver Necklace	164.77	Earrings
8	Gold Necklace	119.14	Necklace

Action Output:

#	Time	Action	Message	Duration / Fetch
390	23:47:00	SELECT * FROM jewelryshopdb.product LIMIT 0, 50000	100 row(s) returned	0.000 sec / 0.000 sec
391	23:50:28	ALTER TABLE Product DROP COLUMN Stock	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.016 sec
392	23:50:35	SELECT * FROM jewelryshopdb.product LIMIT 0, 50000	100 row(s) returned	0.000 sec / 0.000 sec
393	23:51:40	SELECT * FROM jewelryshopdb.product LIMIT 0, 50000	100 row(s) returned	0.000 sec / 0.000 sec

3. Show transactions with top sold/listed entry

```

84      -- Part3 03--
85 •  SELECT COUNT(*) AS TotalTransactions, ProductID, SUM(Quantity) AS TotalSold
86      FROM OrderDetails
87      GROUP BY ProductID
88      ORDER BY TotalSold DESC
89      LIMIT 1;

```

MySQL Workbench

Schemas: jewelryshopdb

Tables: customer, order, orderdetails, payment, product, productwarehouse, warehouse

Result Grid:

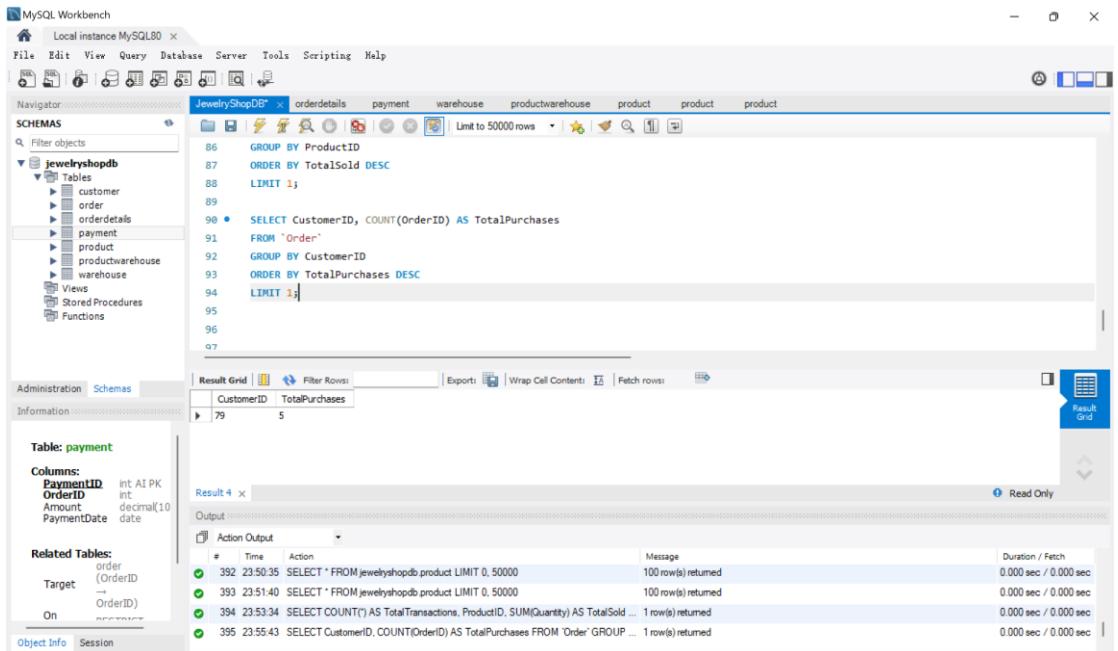
TotalTransactions	ProductID	TotalSold
17	37	119

Action Output:

#	Time	Action	Message	Duration / Fetch
391	23:50:28	ALTER TABLE Product DROP COLUMN Stock	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.016 sec
392	23:50:35	SELECT * FROM jewelryshopdb.product LIMIT 0, 50000	100 row(s) returned	0.000 sec / 0.000 sec
393	23:51:40	SELECT * FROM jewelryshopdb.product LIMIT 0, 50000	100 row(s) returned	0.000 sec / 0.000 sec
394	23:53:34	SELECT COUNT(*) AS TotalTransactions, ProductID, SUM(Quantity) AS TotalSold ...	1 row(s) returned	0.000 sec / 0.000 sec

```

91 •      SELECT CustomerID, COUNT(OrderID) AS TotalPurchases
92      FROM `Order`
93      GROUP BY CustomerID
94      ORDER BY TotalPurchases DESC
95      LIMIT 1;
;
```



4. Order by and Group by

```

97      -- part3 Q4 --
98 •      SELECT MONTH(OrderDate) AS OrderMonth, SUM(TotalAmount) AS MonthlySales
99      FROM `Order`
100     GROUP BY OrderMonth
101    ORDER BY MonthlySales DESC;
```

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
JewelryShopDB | orderdetails payment warehouse productwarehouse product product
schemas tables filters limit 50000 rows
SELECT * FROM `Order` GROUP BY CustomerID ORDER BY TotalPurchases DESC LIMIT 1;
-- part3 Q4 --
SELECT MONTH(OrderDate) AS OrderMonth, SUM(TotalAmount) AS MonthlySales
FROM `Order` GROUP BY Month(OrderDate) ORDER BY OrderMonth DESC LIMIT 1;

```

Result Grid | Filter Rows | Export | Wrap Cell Content | Result Grid

OrderMonth	MonthlySales
5	25898.64
3	25560.80
6	24122.01
4	23049.90
11	21889.10

Output

Action	Time	Action	Message	Duration / Fetch
393	23:51:40	SELECT * FROM jewelryshopdb.product LIMIT 0, 50000	100 row(s) returned	0.000 sec / 0.000 sec
394	23:53:34	SELECT COUNT(*) AS TotalTransactions, ProductID, SUM(Quantity) AS TotalSold ... 1 row(s) returned		0.000 sec / 0.000 sec
395	23:55:43	SELECT CustomerID, COUNT(OrderID) AS TotalPurchases FROM `Order` GROUP ... 1 row(s) returned		0.000 sec / 0.000 sec
396	23:57:18	SELECT MONTH(OrderDate) AS OrderMonth, SUM(TotalAmount) AS MonthlySales... 12 row(s) returned		0.000 sec / 0.000 sec

5. Pattern matching

1.

```

103    -- part3 Q5 --
104  •  SELECT *
105    FROM Customer
106   WHERE Address LIKE '% Junction%';

```

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
JewelryShopDB | orderdetails payment warehouse productwarehouse product product customer
schemas tables filters limit 50000 rows
SELECT * FROM Customer WHERE Address LIKE '% Junction%';

```

Result Grid | Filter Rows | Edit | Export/Import | Wrap Cell Content | Result Grid

CustomerID	Name	Email	Address
4	Deann Giovaniardi	dgiovanardi@mashable.com	78575 Walton Junction
5	Paul Kilmartin	pkilmartin@about.com	144 Victoria Junction
8	Maryou Dodwell	mdodwell7@live.com	6 Löken Junction
16	Crasie Snadden	cnsnadden@wp.com	09242 Shelley Junction
47	Dex Keysel	dkyssel1a@columbia.edu	18 Blue Bill Park Junction

Output

Action	Time	Action	Message	Duration / Fetch
396	23:57:18	SELECT MONTH(OrderDate) AS OrderMonth, SUM(TotalAmount) AS MonthlySales... 12 row(s) returned		0.000 sec / 0.000 sec
397	00:00:32	SELECT * FROM Customer WHERE Address LIKE '%Main Street%' LIMIT 0, 50000	0 row(s) returned	0.000 sec / 0.000 sec
398	00:02:04	SELECT * FROM jewelryshopdb.customer LIMIT 0, 50000	1000 row(s) returned	0.000 sec / 0.000 sec
399	00:02:24	SELECT * FROM Customer WHERE Address LIKE '% Junction%' LIMIT 0, 50000	51 row(s) returned	0.000 sec / 0.000 sec

2.

```
108 •   SELECT *
109     FROM Customer
110    WHERE Name LIKE '%John%';
```

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the `JewelryShopDB` schema with tables: customer, order, orderdetails, payment, product, productwarehouse, warehouse.
- SQL Editor:** Displays the executed SQL code:

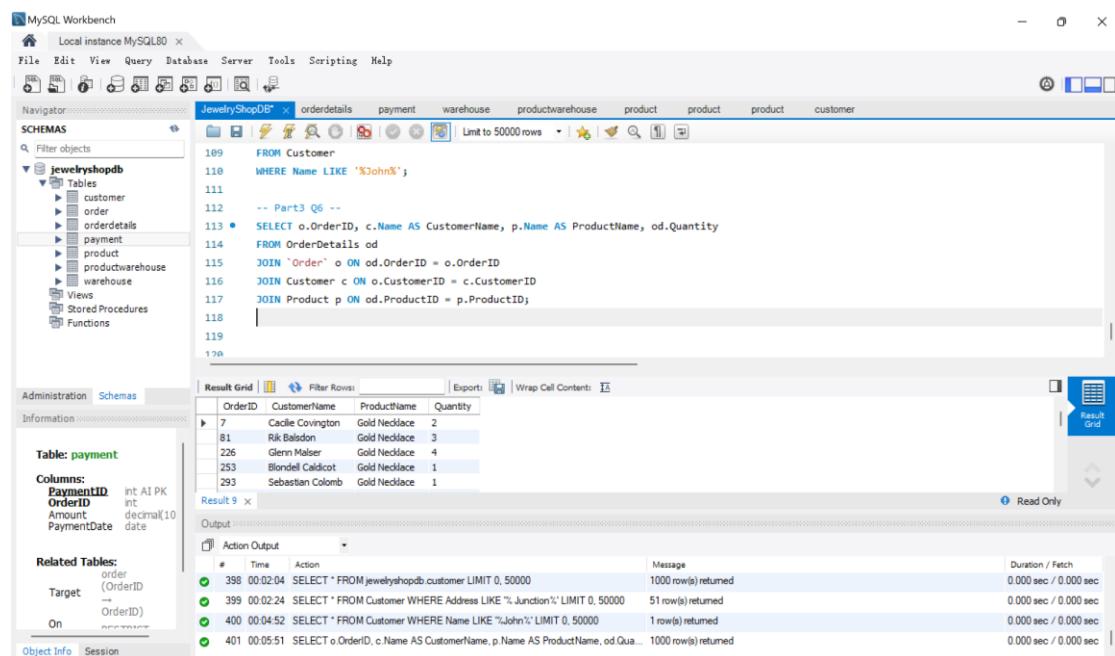
```
98 •   SELECT MONTH(OrderDate) AS OrderMonth, SUM(TotalAmount) AS MonthlySales
99     FROM `Order`
100   GROUP BY OrderMonth
101   ORDER BY MonthlySales DESC;
102
103   -- part3 Q5 --
104 •   SELECT *
105     FROM Customer
106    WHERE Address LIKE '% Junction%';
107
108 •   SELECT *
109     FROM Customer
110    WHERE Name LIKE '%John%';
```
- Result Grid:** Shows the result of the last query:

CustomerID	Name	Email	Address
1	Alice Johnson	alice.johnson@example.com	123 Main St
- Object Info:** Shows information for the `payment` table, including columns (`PaymentID`, `OrderID`, `Amount`, `PaymentDate`), related tables (`order`, `Target`, `On`), and constraints.
- Action Output:** Shows the history of actions taken:

#	Time	Action	Message	Duration / Fetch
397	00:00:32	SELECT * FROM Customer WHERE Address LIKE '%Main Street%'; LIMIT 0, 50000	0 row(s) returned	0.000 sec / 0.000 sec
398	00:02:04	SELECT * FROM JewelryShopDB.customer LIMIT 0, 50000	1000 row(s) returned	0.000 sec / 0.000 sec
399	00:02:24	SELECT * FROM Customer WHERE Address LIKE '% Junction%'; LIMIT 0, 50000	51 row(s) returned	0.000 sec / 0.000 sec
400	00:04:52	SELECT * FROM Customer WHERE Name LIKE '%John%'; LIMIT 0, 50000	1 row(s) returned	0.000 sec / 0.000 sec

6. Information from multiple tables using Join

```
112      -- Part3 Q6 --
113 •   SELECT o.OrderID, c.Name AS CustomerName, p.Name AS ProductName, od.Quantity
114     FROM OrderDetails od
115     JOIN `Order` o ON od.OrderID = o.OrderID
116     JOIN Customer c ON o.CustomerID = c.CustomerID
117     JOIN Product p ON od.ProductID = p.ProductID;
```



7. View of most recent transactions

```
119      -- Part3 Q7 --
120 •   CREATE VIEW MostFrequentTransactions AS
121     SELECT ProductID, SUM(Quantity) AS TotalSold
122     FROM OrderDetails
123     GROUP BY ProductID
124     ORDER BY TotalSold DESC
125     LIMIT 7;
```

The screenshot shows the MySQL Workbench interface. The top menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The main window has tabs for orderdetails, payment, warehouse, productwarehouse, product, customer, and a limit of 50000 rows. The left sidebar shows the 'jewelryshopdb' schema with tables like customer, order, orderdetails, payment, product, productwarehouse, warehouse, views, stored procedures, and functions. The central pane displays a query editor with the following SQL code:

```

118 WHERE Name LIKE '%John%';
119
120 -- Part3 Q6 --
121 • SELECT o.OrderID, c.Name AS CustomerName, p.Name AS ProductName, od.Quantity
122 FROM OrderDetails od
123 JOIN `Order` o ON od.OrderID = o.OrderID
124 JOIN Customer c ON o.CustomerID = c.CustomerID
125 JOIN Product p ON od.ProductID = p.ProductID;
126
127
128
129
130
131
132
133
134
135

```

The 'Information' pane on the right shows details for the 'payment' table, including columns (PaymentID, OrderID, Amount, PaymentDate) and related tables (order). The 'Output' pane at the bottom shows the execution log with four entries:

#	Time	Action	Message	Duration / Fetch
399	00:02:24	SELECT * FROM Customer WHERE Address LIKE "%Junction%" LIMIT 0, 50000	51 row(s) returned	0.000 sec / 0.000 sec
400	00:04:52	SELECT * FROM Customer WHERE Name LIKE "%John%" LIMIT 0, 50000	1 row(s) returned	0.000 sec / 0.000 sec
401	00:05:51	SELECT o.OrderID, c.name AS CustomerName, p.Name AS ProductName, od.Q... 1000 row(s) returned		0.000 sec / 0.000 sec
402	00:06:38	CREATE VIEW MostFrequentTransactions AS SELECT ProductID, SUM(Quantity) ... 0 row(s) affected		0.000 sec

8. Summary of annual transactions

8.1 Total Number of Transactions and Details per Month

```

128 -- Part3 Q8 Shows the total number of transactions with corresponding details every month --
129 • SELECT
130     MONTH(OrderDate) AS OrderMonth,
131     COUNT(OrderID) AS TotalTransactions,
132     SUM(TotalAmount) AS TotalSales
133 FROM `Order`
134 GROUP BY OrderMonth
135 ORDER BY OrderMonth;

```

The screenshot shows the MySQL Workbench interface. The top menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The left sidebar displays the Navigator, Schemas, and the current database, jewelryshopdb. The Tables section under jewelryshopdb lists customer, order, orderdetails, payment, product, productwarehouse, and warehouse. Below these are Views, Stored Procedures, and Functions. The main workspace contains a query editor with the following script:

```
118 -- Part3 Q7 --
119 • CREATE VIEW MostFrequentTransactions AS
120     SELECT ProductID, SUM(Quantity) AS TotalSold
121     FROM OrderDetails
122     GROUP BY ProductID
123     ORDER BY TotalSold DESC
124
125     LIMIT 7;
126
127 -- Part3 Q8 --
128 • SELECT
129     MONTH(OrderDate) AS OrderMonth,
```

Below the script is a results grid showing the output of the query:

OrderMonth	TotalTransactions	TotalSales
1	82	19955.25
2	76	1994.21
3	90	25560.80
4	81	23049.90
5	99	25898.64

The bottom pane shows the Action Output log:

#	Time	Action	Message	Duration / Fetch
400	00:04:52	SELECT * FROM Customer WHERE Name LIKE "%John%" LIMIT 0, 50000	1 row(s) returned	0.000 sec / 0.000 sec
401	00:05:51	SELECT o.OrderID, c.Name AS CustomerName, p.Name AS ProductName, od.Qty...	10000 row(s) returned	0.000 sec / 0.000 sec
402	00:06:38	CREATE VIEW MostFrequentTransactions AS SELECT ProductID, SUM(Quantity) ...	0 row(s) affected	0.000 sec
403	00:08:54	SELECT MONTH(OrderDate) AS OrderMonth, COUNT(OrderID) AS TotalTra...	12 row(s) returned	0.000 sec / 0.000 sec

8.2. Customer Purchase Value Per Month

```
137      -- Part3 Q8 Shows customer purchase value per month --
138 •  SELECT
139      CustomerID,
140      MONTH(OrderDate) AS OrderMonth,
141      SUM(TotalAmount) AS TotalPurchase
142      FROM `Order`
143      GROUP BY CustomerID, OrderMonth
144      ORDER BY CustomerID, OrderMonth;
```

The screenshot shows the MySQL Workbench interface. The top navigation bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The left sidebar displays the Navigator, Schemas (with 'jewelryshopdb' selected), Tables, Views, Stored Procedures, and Functions. The main area shows a query editor with the following SQL code:

```
135
136 • SELECT
137     CustomerID,
138     MONTH(OrderDate) AS OrderMonth,
139     SUM(TotalAmount) AS TotalPurchase
140
141 FROM `Order`
142 GROUP BY CustomerID, OrderMonth
143
144
145
```

Below the query editor is a Result Grid tab with the following data:

CustomerID	OrderMonth	TotalPurchase
3	4	470.58
3	5	246.68
3	10	565.00
4	5	123.98
8	6	398.53

The bottom section shows the Action Output log:

#	Time	Action	Message	Duration / Fetch
1	04/05/21 00:05:51	SELECT o.OrderID, c.Name AS CustomerName, p.Name AS ProductName, od.Quantity AS Quantity, od.UnitPrice AS UnitPrice, od.TotalAmount AS TotalAmount FROM `Order` o JOIN `Customer` c ON o.CustomerID = c.CustomerID JOIN `Product` p ON o.ProductID = p.ProductID JOIN `OrderDetail` od ON o.OrderID = od.OrderID WHERE o.OrderDate > '2021-01-01' AND o.OrderDate < '2021-02-01' ORDER BY o.OrderDate	1000 row(s) returned	0.000 sec / 0.000 sec
2	04/05/21 00:06:38	CREATE VIEW MostFrequentTransactions AS SELECT ProductID, SUM(Quantity) AS TotalQuantity, COUNT(OrderID) AS TotalTransactions FROM `OrderDetail` GROUP BY ProductID	0 row(s) affected	0.000 sec
3	04/05/21 00:08:54	SELECT MONTH(OrderDate) AS OrderMonth, COUNT(OrderID) AS TotalTransactions, SUM(TotalAmount) AS TotalPurchase FROM `Order` GROUP BY MONTH(OrderDate)	12 row(s) returned	0.000 sec / 0.000 sec
4	04/05/21 00:10:10	SELECT CustomerID, MONTH(OrderDate) AS OrderMonth, SUM(TotalAmount) AS TotalPurchase FROM `Order` GROUP BY CustomerID, MONTH(OrderDate)	956 row(s) returned	0.016 sec / 0.000 sec

8.3 Product Sales Quantity Per Month

```

146      -- Part3 Q8 Shows name of product and number sold each month --
147  •  SELECT
148      MONTH(o.OrderDate) AS OrderMonth,
149      p.Name AS ProductName,
150      SUM(od.Quantity) AS TotalSold
151  FROM OrderDetails od
152  JOIN `Order` o ON od.OrderID = o.OrderID
153  JOIN Product p ON od.ProductID = p.ProductID
154  GROUP BY OrderMonth, ProductName
155  ORDER BY OrderMonth, TotalSold DESC;

```

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** Local instance MySQL80.
- Navigator:** Shows the schema **JewelryShopDB** with tables: customer, order, orderdetails, payment, product, productwarehouse, warehouse.
- Query Editor:** Displays the SQL query from the code block above.
- Result Grid:** Shows the output of the query:

OrderMonth	ProductName	TotalSold
1	Diamond Ring	106
1	Gold Necklace	92
1	Silver Necklace	78
1	Crystal Pendant Handmade Bracelet Platinum Ring	61
1	Pearl Bracelet	50
- Output:** Shows the history of actions and messages:

#	Action	Message	Duration / Fetch
402	CREATE VIEW MostFrequentTransactions AS SELECT ProductID, SUM(Quantity) ...	0 row(s) affected	0.000 sec
403	SELECT MONTH(OrderDate) AS OrderMonth, COUNT(OrderID) AS TotalTra... 12 row(s) returned		0.000 sec / 0.000 sec
404	SELECT CustomerID, MONTH(OrderDate) AS OrderMonth, SUM(TotalAmo... 956 row(s) returned		0.016 sec / 0.000 sec
405	SELECT MONTH(o.OrderDate) AS OrderMonth, p.Name AS ProductName, ... 79 row(s) returned		0.000 sec / 0.000 sec