**1. Database Creation**

CREATE DATABASE amazonfresh;

USE amazonfresh;

**2. Import CSV Data**

Use **MySQL Workbench > Table Data Import Wizard** for each file.

* Customers → customers.csv
* Orders → orders.csv
* Order\_Details → order\_details.csv
* Products → products.csv
* Reviews → reviews.csv
* Suppliers → suppliers.csv

**3. Primary Key Setup**

-- Customers

ALTER TABLE customers DROP PRIMARY KEY;

ALTER TABLE customers ADD PRIMARY KEY (CustomerID(36));

-- Orders

ALTER TABLE orders DROP PRIMARY KEY;

ALTER TABLE orders ADD PRIMARY KEY (OrderID(36));

-- Products

ALTER TABLE products DROP PRIMARY KEY;

ALTER TABLE products ADD PRIMARY KEY (ProductID(36));

-- Suppliers (fix column name if needed)

ALTER TABLE suppliers CHANGE `Uniquesupplier ID` SupplierID VARCHAR(36);

ALTER TABLE suppliers DROP PRIMARY KEY;

ALTER TABLE suppliers ADD PRIMARY KEY (SupplierID(36));

-- Reviews

ALTER TABLE reviews DROP PRIMARY KEY;

ALTER TABLE reviews ADD PRIMARY KEY (ReviewID(36));

-- Order\_Details (composite key)

ALTER TABLE order\_details DROP PRIMARY KEY;

ALTER TABLE order\_details ADD PRIMARY KEY (OrderID(36), ProductID(36));

**4. Foreign Key Setup**

-- Orders → Customers

ALTER TABLE orders

ADD CONSTRAINT fk\_orders\_customers

FOREIGN KEY (CustomerID) REFERENCES customers(CustomerID);

-- Order\_Details → Orders

ALTER TABLE order\_details

ADD CONSTRAINT fk\_orderdetails\_orders

FOREIGN KEY (OrderID) REFERENCES orders(OrderID);

-- Order\_Details → Products

ALTER TABLE order\_details

ADD CONSTRAINT fk\_orderdetails\_products

FOREIGN KEY (ProductID) REFERENCES products(ProductID);

-- Products → Suppliers

ALTER TABLE products

ADD CONSTRAINT fk\_products\_suppliers

FOREIGN KEY (SupplierID) REFERENCES suppliers(SupplierID);

-- Reviews → Products

ALTER TABLE reviews

ADD CONSTRAINT fk\_reviews\_products

FOREIGN KEY (ProductID) REFERENCES products(ProductID);

-- Reviews → Customers

ALTER TABLE reviews

ADD CONSTRAINT fk\_reviews\_customers

FOREIGN KEY (CustomerID) REFERENCES customers(CustomerID);

**5. Fix Missing Suppliers (if any foreign key fails)**

INSERT INTO suppliers (SupplierID, SupplierName, ContactPerson, Phone, City, State)

SELECT DISTINCT SupplierID, 'Unknown', 'Unknown', '0000000000', 'Unknown', 'Unknown'

FROM products

WHERE SupplierID NOT IN (SELECT SupplierID FROM suppliers);

**6. ER Diagram Creation Steps in MySQL Workbench**

1. **Open MySQL Workbench** and connect to your server.
2. Go to **Database > Reverse Engineer**.
3. Select your amazonfresh schema and click **Continue**.
4. Select all tables and relationships → click **Execute**.
5. The **ER Diagram (EER)** will be generated showing all tables and foreign keys.
6. Arrange layout if needed using the mouse.
7. To export the ER diagram:
   * Go to **File > Export > Export as PNG** or PDF.
   * Save the image and embed it in your GitHub README or PowerPoint.

**7. Business Insight SQL Queries**

**Top Performing Products by Revenue**

SELECT

p.ProductName,

SUM(od.Quantity \* od.UnitPrice \* (1 - od.Discount)) AS TotalRevenue,

SUM(od.Quantity) AS TotalUnitsSold

FROM

order\_details od

JOIN

products p ON od.ProductID = p.ProductID

GROUP BY

p.ProductName

ORDER BY

TotalRevenue DESC

LIMIT 10;

**New vs Returning Customers**

SELECT

CASE

WHEN order\_count = 1 THEN 'New Customer'

ELSE 'Returning Customer'

END AS CustomerType,

COUNT(\*) AS NumberOfCustomers

FROM (

SELECT

o.CustomerID,

COUNT(o.OrderID) AS order\_count

FROM

orders o

GROUP BY

o.CustomerID

) AS customer\_orders

GROUP BY

CustomerType;

**Sales Trend by Category**

SELECT

p.Category,

DATE\_FORMAT(o.OrderDate, '%Y-%m') AS Month,

SUM(od.Quantity \* od.UnitPrice \* (1 - od.Discount)) AS MonthlySales

FROM

order\_details od

JOIN

orders o ON od.OrderID = o.OrderID

JOIN

products p ON od.ProductID = p.ProductID

GROUP BY

p.Category, Month

ORDER BY

Month ASC, MonthlySales DESC;

**Customer Purchasing Patterns**

SELECT

c.CustomerID,

c.Name,

COUNT(o.OrderID) AS TotalOrders

FROM

customers c

JOIN

orders o ON c.CustomerID = o.CustomerID

GROUP BY

c.CustomerID, c.Name

HAVING

TotalOrders > 2

ORDER BY

TotalOrders DESC;

**Frequently Bought Together**

SELECT

od1.ProductID AS Product1,

od2.ProductID AS Product2,

COUNT(\*) AS BoughtTogetherCount

FROM

order\_details od1

JOIN

order\_details od2

ON od1.OrderID = od2.OrderID AND od1.ProductID < od2.ProductID

GROUP BY

od1.ProductID, od2.ProductID

ORDER BY

BoughtTogetherCount DESC

LIMIT 10;

**Inventory Analysis: Overstock / Stockouts**

SELECT

p.ProductID,

p.ProductName,

p.StockQuantity,

SUM(od.Quantity) AS UnitsSold

FROM

products p

LEFT JOIN

order\_details od ON p.ProductID = od.ProductID

GROUP BY

p.ProductID, p.ProductName, p.StockQuantity

ORDER BY

(p.StockQuantity - SUM(od.Quantity)) ASC;