```
-- Use My Databse
USE e_commerce;

-- Encrypt Database
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'password1';
-- Create certificate to protect symmetric key
CREATE CERTIFICATE TestCertificate
WITH SUBJECT = 'project test certificate',
EXPIRY_DATE = '2028-06-01';

-- Create symmetric key to encrypt data
CREATE SYMMETRIC KEY TestSymmetricKey
WITH ALGORITHM = AES_128
ENCRYPTION BY CERTIFICATE TestCertificate;

-- Open symmetric key
OPEN SYMMETRIC KEY TestSymmetricKey
DECRYPTION BY CERTIFICATE TestCertificate;

-- Create Date Table
CREATE TABLE dbo.Date(
      date_id int NOT NULL CONSTRAINT PK_date_id Primary Key,
      full_date date NOT NULL,
      day_of_week varchar(20) NOT NULL,
      date_number int NOT NULL
            CHECK (date_number BETWEEN 1 AND 31),
      month_name varchar(15) NOT NULL,
      month_number int NOT NULL
            CHECK (month_number BETWEEN 1 AND 12),
      quarter_number int NOT NULL
            CHECK (quarter_number BETWEEN 1 AND 4),
      calendar_year int NOT NULL
);
-- Add a computed column for fiscal period
ALTER TABLE Date ADD fiscal_period AS
   CASE
       WHEN quarter_number = 1 THEN 'Q1-' + CAST(calendar_year AS
VARCHAR)
       WHEN quarter_number = 2 THEN 'Q2-' + CAST(calendar_year AS
VARCHAR)
       WHEN quarter_number = 3 THEN 'Q3-' + CAST(calendar_year AS
VARCHAR)
       WHEN quarter_number = 4 THEN 'Q4-' + CAST(calendar_year AS
VARCHAR)
   END;

-- Imported Data Using DBeaver
-- Select statement to see if the data imported correctly
SELECT * FROM Date;

-- Create Payment Table
CREATE TABLE dbo.Payment(
      payment_id int NOT NULL CONSTRAINT PK_payment_id Primary Key,
      payment_method varchar(20) NOT NULL,
```

```
      payment_amount money NOT NULL,
      payment_status varchar(20) NOT NULL
);
-- Imported Data Using DBeaver
-- Select statement to see if the data imported correctly
SELECT * FROM Payment;

-- Create Review Table
CREATE TABLE dbo.Review (
      review_id int NOT NULL CONSTRAINT PK_review_id Primary Key,
      review_date date,
      rating int NOT NULL
            CHECK (rating BETWEEN 1 AND 5),
      review_text varchar(225) NOT NULL
);
-- Imported Data Using DBeaver
-- Select statement to see if the data imported correctly
SELECT * FROM Review;

-- Create Shipment Table
CREATE TABLE dbo.Shipment(
      shipment_id int NOT NULL CONSTRAINT PK_shipment_id Primary Key,
      shipment_date date NOT NULL,
      carrier varchar(40) NOT NULL,
      tracking_number varchar(40) NOT NULL,
      shipment_status varchar(40) NOT NULL,
      delivery_date date NOT NULL,
      ship_time AS (DATEDIFF(dd, shipment_date, delivery_date)) -- added
a computed column to figure out the ship time
);
-- Imported Data Using DBeaver
-- Select statement to see if the data imported correctly
SELECT * FROM Shipment;

-- Create Returns Table
-- Needed to add [] to return because it is a reserved keyword in SQL
CREATE TABLE dbo.[Return](
      return_id int NOT NULL CONSTRAINT PK_return_id Primary Key,
      return_date date NOT NULL,
      return_status varchar(40) NOT NULL,
      return_reason varchar(40) NOT NULL,
      refund_amount money NOT NULL
);
-- Imported Data Using DBeaver
-- Select statement to see if the data imported correctly
SELECT * FROM [Return];

-- Create Contact Table
CREATE TABLE dbo.Contact(
      contact_id int NOT NULL CONSTRAINT PK_contact_id Primary Key,
      phone varchar(20) NOT NULL,
      email varchar(40) NOT NULL
);
-- Imported Data Using DBeaver
```

```
-- Select statement to see if the data imported correctly
SELECT * FROM Contact;

-- Create Location Table
CREATE TABLE dbo.Location(
      location_id int NOT NULL CONSTRAINT PK_location_id Primary Key,
      street_address varchar(40) NOT NULL,
      city varchar(40) NOT NULL,
      state_province varchar(5) NOT NULL,
      postal_code varchar(10) NOT NULL,
      country varchar(5) NOT NULL
);
-- Imported Data Using DBeaver
-- Select statement to see if the data imported correctly
SELECT * FROM Location;

-- Create Customer Table
CREATE TABLE dbo.Customer(
      customer_id int NOT NULL CONSTRAINT PK_customer_id Primary Key,
      location_id int NOT NULL CONSTRAINT FK_customer_location_id Foreign
Key (location_id) REFERENCES dbo.Location(location_id),
      contact_id int NOT NULL CONSTRAINT FK_customer_contact_id Foreign
Key (contact_id) REFERENCES dbo.Contact(contact_id),
      first_name varchar(40) NOT NULL,
      last_name varchar(40) NOT NULL
);
-- Imported Data Using DBeaver
-- Select statement to see if the data imported correctly
SELECT * FROM Customer;

-- Create Supplier Table
CREATE TABLE dbo.Supplier(
      supplier_id int NOT NULL CONSTRAINT PK_supplier_id Primary Key,
      location_id int NOT NULL CONSTRAINT FK_supplier_location_id Foreign
Key (location_id) REFERENCES dbo.Location(location_id),
      contact_id int NOT NULL CONSTRAINT FK_supplier_contact_id Foreign
Key (contact_id) REFERENCES dbo.Contact(contact_id),
      supplier_name varchar(30) NOT NULL
);
-- Imported Data Using DBeaver
-- Select statement to see if the data imported correctly
SELECT * FROM Supplier;

-- Create Warehouse Table
CREATE TABLE dbo.Warehouse(
      warehouse_id int NOT NULL CONSTRAINT PK_warehouse_id Primary Key,
      location_id int NOT NULL CONSTRAINT FK_warehouse_location_id
Foreign Key (location_id) REFERENCES dbo.Location(location_id),
      contact_id int NOT NULL CONSTRAINT FK_warehouse_contact_id Foreign
Key (contact_id) REFERENCES dbo.Contact(contact_id),
      warehouse_name varchar(40) NOT NULL
);
-- Imported Data Using DBeaver
-- Select statement to see if the data imported correctly
```

```
SELECT * FROM Warehouse;

-- Create Inventory Table
CREATE TABLE dbo.Inventory(
      inventory_id int NOT NULL CONSTRAINT PK_inventory_id Primary Key,
      stock_quantity int NOT NULL
);
-- Imported Data Using DBeaver
-- Select statement to see if the data imported correctly
SELECT * FROM Inventory;

-- Create Product_Category Table
CREATE TABLE dbo.Product_Category(
      category_id int NOT NULL CONSTRAINT PK_category_id Primary Key,
      product_category varchar(20) NOT NULL
);
-- Imported Data Using DBeaver
-- Select statement to see if the data imported correctly
SELECT * FROM Product_Category;

-- Create Product Table
CREATE TABLE dbo.Product(
      product_id int NOT NULL CONSTRAINT PK_product_id Primary Key,
      supplier_id int NOT NULL CONSTRAINT FK_product_supplier_id Foreign
Key (supplier_id) REFERENCES dbo.Supplier(supplier_id),
      category_id int NOT NULL CONSTRAINT FK_prouct_category_id Foreign
Key (category_id) REFERENCES dbo.Product_Category(category_id),
      inventory_id int NOT NULL CONSTRAINT FK_product_inventory_id
Foreign Key (inventory_id) REFERENCES dbo.Inventory(inventory_id),
      warehouse_id int NOT NULL CONSTRAINT FK_product_warehouse_id
Foreign Key (warehouse_id) REFERENCES dbo.Warehouse(warehouse_id),
      product_name varchar(40) NOT NULL,
      product_price money NOT NULL
);
-- Imported Data Using DBeaver
-- Select statement to see if the data imported correctly
SELECT * FROM Product;

-- Create Order_Detail
CREATE TABLE dbo.Order_Detail(
      order_detail_id int NOT NULL CONSTRAINT PK_order_detail_id Primary
Key,
      product_id int NOT NULL CONSTRAINT FK_od_product_id Foreign Key
(product_id) REFERENCES dbo.Product(product_id),
      product_quantity_sold int
);
-- Imported Data Using DBeaver
-- Select statement to see if the data imported correctly
SELECT * FROM Order_Detail;

-- Create Order_Header
CREATE TABLE dbo.Order_Header(
      order_header_id int NOT NULL CONSTRAINT PK_order_id Primary Key,
```

```
      order_detail_id int NOT NULL CONSTRAINT FK_order_detail_id Foreign
Key (order_detail_id) REFERENCES dbo.order_detail(order_detail_id),
      order_date int NOT NULL CONSTRAINT FK_order_date Foreign Key
(order_date) REFERENCES dbo.Date(date_id),
      customer_id int NOT NULL CONSTRAINT FK_customer_id Foreign Key
(customer_id) REFERENCES dbo.Customer(customer_id),
      payment_id int NOT NULL CONSTRAINT FK_payment_id Foreign Key
(payment_id) REFERENCES dbo.Payment(payment_id),
      shipment_id int NOT NULL CONSTRAINT FK_shipment_id Foreign Key
(shipment_id) REFERENCES dbo.Shipment(shipment_id),
      review_id int NOT NULL CONSTRAINT FK_review_id Foreign Key
(review_id) REFERENCES dbo.Review(review_id),
      return_id int CONSTRAINT FK_return_id Foreign Key (return_id)
REFERENCES dbo.[Return](return_id)
);
-- Imported Data Using DBeaver
-- Select statement to see if the data imported correctly
SELECT * FROM Order_Header;


/* VIEWS */
-- Create a view for date dimension reporting
CREATE VIEW DateDimensionReport AS
SELECT
   date_id,
   full_date,
   day_of_week,
   month_name,
   quarter_number,
   calendar_year,
   fiscal_period
FROM
   Date;

-- Create a view for locations by country using FOR XML PATH instead of
STRING_AGG
CREATE VIEW LocationsByCountry AS
SELECT
   country,
   COUNT(*) as location_count,
   STUFF((
       SELECT ', ' + city
       FROM Location t
       WHERE t.country = l.country
       GROUP BY city
       ORDER BY city
       FOR XML PATH('')
   ), 1, 2, '') as cities
FROM
   Location l
GROUP BY
   country;

-- Create a view for encrypted data
```

```
CREATE VIEW SecureLocationView AS
SELECT
   location_id,
   ENCRYPTBYPASSPHRASE('encryption_key', street_address) as
encrypted_address,
   city,
   state_province,
   LEFT(postal_code, 3) + '***' as masked_postal_code,
   country
FROM
   Location;

-- customer order summary view
-- overview of customer orders, including total order amount, order date,
and payment details
-- left join if we want all customers, even if they haven't placed an
order
-- inner join if we only want to see customers who placed orders
CREATE VIEW CustomerOrderSummary AS
SELECT
    oh.order_header_id,
    c.customer_id,
    CONCAT(c.first_name, ' ', c.last_name) AS CustomerName,
    oh.order_date,
    COALESCE(SUM(p.payment_amount), 0) AS TotalPaid,  -- Ensures NULL
payments show as 0, returns the first non-null value
    COALESCE(p.payment_method, 'Unknown') AS payment_method, -- In case
no payment method is found
    COUNT(DISTINCT od.product_id) AS TotalProductsOrdered  -- Avoids
duplicate counts
FROM Order_Header oh
INNER JOIN Customer c ON oh.customer_id = c.customer_id
LEFT OUTER JOIN Payment p ON oh.payment_id = p.payment_id
LEFT OUTER JOIN Order_Detail od ON oh.order_detail_id =
od.order_detail_id  -- Ensure correct join condition here
GROUP BY
    oh.order_header_id,
    c.customer_id,
    c.first_name,
    c.last_name,  -- Group by first_name and last_name explicitly
    oh.order_date,
    p.payment_method;

-- product sales summary view
-- total quantity and revenue per product.
CREATE VIEW ProductSales AS
SELECT
    p.product_id,
    p.product_name,
    pc.product_category,
    SUM(od.product_quantity_sold) AS TotalQuantitySold,
    SUM(od.product_quantity_sold * p.product_price) AS TotalRevenue
FROM Order_Detail od
INNER JOIN Product p ON od.product_id = p.product_id
```

```
INNER JOIN Product_Category pc ON p.category_id = pc.category_id
GROUP BY p.product_id, p.product_name, pc.product_category;

-- inventory stock Levels View
-- current stock levels of products in warehouses
CREATE VIEW InventoryStock AS
SELECT
    i.inventory_id,
    p.product_name,
    w.warehouse_name,
    i.stock_quantity,
    co.Phone,
    co.Email,
    l.city,
    l.state_province
FROM Inventory i
INNER JOIN Product p ON i.inventory_id = p.inventory_id
INNER JOIN Warehouse w ON p.warehouse_id = w.warehouse_id
LEFT JOIN Contact co ON w.contact_id = co.contact_id
LEFT JOIN Location l ON w.location_id = l.location_id;

--customer order history View
--shows each customer's past orders with review details
CREATE VIEW View_CustomerOrderHistory AS
SELECT
    c.customer_id,
    CONCAT(c.first_name, ' ', c.last_name) AS CustomerName,
    oh.order_header_id,
    oh.order_date,
    rev.rating AS review_rating,
    rev.review_text AS review_comment
FROM Customer c
INNER JOIN Order_Header oh ON c.customer_id = oh.customer_id
LEFT OUTER JOIN Review rev ON oh.review_id = rev.review_id;


-- shipment tracking View
-- details on shipments, including their current status
CREATE VIEW ShipmentTracking AS
SELECT
    sh.shipment_id,
    oh.order_header_id,
    oh.customer_id,
    CONCAT(c.first_name, ' ', c.last_name) AS CustomerName,
    sh.shipment_status,
    sh.delivery_date,
    CASE
        WHEN sh.shipment_status = 'Delivered' THEN 'Completed'
        WHEN sh.shipment_status = 'Shipped' THEN 'Shipped'
        WHEN sh.shipment_status = 'Cancelled' THEN 'Cancelled'
        WHEN sh.shipment_status = 'Pending' THEN 'Pending'
        ELSE 'Unknown'  -- A default case in case there are any other
unexpected statuses
    END AS ShipmentProgress
```

```
FROM Shipment sh
INNER JOIN Order_Header oh ON sh.shipment_id = oh.shipment_id
INNER JOIN Customer c ON oh.customer_id = c.customer_id;

--create supplier product list
-- list of all suppliers and products they provide
CREATE VIEW SupplierProductList AS
SELECT
    s.supplier_id,
    s.supplier_name,
    p.product_id,
    p.product_name,
    p.product_price,
    l.city,
    l.state_province,
    c.phone,
    c.email
FROM Supplier s
JOIN Product p ON s.supplier_id = p.supplier_id
LEFT JOIN Location l ON s.location_id = l.location_id
LEFT JOIN Contact c ON s.contact_id = c.contact_id;


-- create a return and review table
-- details on returns, and review ratings and text to see why they made
the return
CREATE VIEW ReturnAndReview AS
SELECT
    re.return_id,
    oh.order_header_id,
    od.product_id,
    p.product_name,
    re.return_date,
    re.return_reason,
    re.refund_amount,
    re.return_status,
    rev.rating AS review_rating,
    rev.review_text AS review_comment,
    oh.customer_id,
    CONCAT(c.first_name, ' ', c.last_name) AS CustomerName
FROM Order_Header oh
INNER JOIN Order_Detail od ON oh.order_detail_id = od.order_detail_id
INNER JOIN [Return] re ON re.return_id = oh.return_id
INNER JOIN Product p ON od.product_id = p.product_id
LEFT JOIN Review rev ON oh.review_id = rev.review_id
LEFT JOIN Customer c ON oh.customer_id = c.customer_id;
```