

Student's Full Name: Rohit Yadav
Course Title: Database Design and Prototyping
Term name and year: Fall 2025
Submission Week: Week 15 Final Project Report
Instructor's Name: Nayem Rahman
Date of Submission: 17th December 2025

Project Title: Supply Chain and Logistics Management System

1. Brief Description of the Project

This project focuses on developing a relational database system for a logistics startup, SwiftLogix Inc., which manages the end-to-end flow of goods across its supply chain network. The company connects multiple suppliers, warehouses, transport carriers, and retail clients to ensure on-time and cost-efficient product delivery. The goal of this project is to design a centralized database that accurately tracks product movement, shipment schedules, warehouse inventory, and delivery performance metrics.

The proposed system will enable managers to view and analyze key logistics data such as order volumes, supplier reliability, carrier performance, and warehouse utilization. By integrating these elements into a single database, the company can reduce manual errors, improve real-time visibility into shipments, and enhance decision-making related to cost control, delivery routes, and vendor management. The project will eventually progress through the conceptual, logical, and physical stages of database design, ensuring normalization, referential integrity, and efficient query execution.

2. Data Used and Description

The dataset used for this project is the “DataCo Smart Supply Chain for Big Data Analysis” dataset, publicly available on Kaggle. It contains comprehensive information representing real-world supply chain operations. The data includes key entities such as Suppliers, Products, Orders, Shipments, Customers, and Carriers, each providing attributes necessary for tracking and analysis.

For example, the Orders table includes order dates, quantities, and shipment details; the Products table lists product categories, weights, and costs; and the Shipments table records carrier names, shipping modes, routes, and delivery times. The Customers and Suppliers tables link buyers and vendors to specific transactions, forming the basis for a multi-entity relational structure.

The dataset will be refined to a manageable size (around 100–200 rows) for the course requirements while maintaining relational integrity between entities. It will serve as the foundation for creating conceptual and logical models in subsequent phases and will support SQL queries related to performance metrics such as delivery delays, order frequency, and cost optimization.

3. Source of Data

The dataset is sourced from Kaggle:

DataCo Smart Supply Chain for Big Data Analysis (<https://www.kaggle.com/datasets/shashwatwork/dataco-smart-supply-chain-for-big-data-analysis>).

This open-source dataset offers structured supply-chain data that is ideal for academic modeling and relational database prototyping.

Supply Chain and Logistics Data Model Explanation

Overview

The Supply Chain and Logistics Management System database has been designed to organize, track, and analyze business operations from order placement to product delivery.

The system manages data about departments, categories, products, customers, orders, and individual order line items. The goal is to ensure smooth operations across multiple business functions, from product classification to customer transactions, while maintaining data integrity, normalization, and analytical efficiency.

Core Entities (5) and Junction Entity (1)

1. Departments

The Departments entity represents the top-level business divisions of the organization, such as Electronics, Apparel, or Fitness.

Each department groups related categories of products to provide an organized structure for reporting and management.

- Primary Key: department_id
- Attributes: department_name
- Relationship: One department can have many categories (1:M).
→ Each category must belong to exactly one department.

2. Categories

The Categories entity further classifies products under specific departments. It enables product segmentation based on functionality or target market (e.g., “Mobile Devices,” “Sporting Goods”).

- Primary Key: category_id
- Foreign Key: department_id → references Departments
- Attributes: category_name
- Relationship:
 - Many-to-One (M:1) with Departments

- One-to-Many (1:M) with Products

Each category can include multiple products, but every product must belong to one category.

3. Products

The Products entity contains detailed information about the items available for sale or shipment. It includes attributes such as product name, price, and description, enabling price tracking, performance analysis, and category-based sales insights.

- Primary Key: product_card_id
- Foreign Key: product_category_id → references Categories
- Attributes: product_name, product_price, product_status, product_description, product_image
- Relationship:
 - Many-to-One (M:1) with Categories
 - One-to-Many (1:M) with Order_Items

Products are linked to categories for classification and to order items to record which products were sold.

4. Customers

The Customers entity stores data about buyers who place orders. It includes personal and regional details, allowing the company to identify customer segments, regional sales patterns, and delivery locations.

- Primary Key: customer_id
- Attributes: customer_fname, customer_lname, customer_email, customer_city, customer_state, customer_country, customer_zipcode, market, region
- Relationship: One-to-Many (1:M) with Orders

Each customer can place multiple orders, but each order belongs to one customer.

5. Orders

The Orders entity represents transactions made by customers. It records details such as order date, shipping method, delivery status, and performance metrics like profit or benefit per order. It forms the backbone of the transactional system, linking customers to products through order items.

- Primary Key: order_id
- Foreign Key: customer_id → references Customers

- Attributes:

order_date, order_status, order_city, order_state, order_country, order_zipcode, order_region, shipping_date, shipping_mode, delivery_status, late_delivery_risk, days_for_shipping_real, days_for_shipment_scheduled, benefit_per_order, sales_per_customer, order_profit_per_order
- Relationship:
 - Many-to-One (M:1) with Customers
 - One-to-Many (1:M) with Order_Items

6. Order_Items (Junction Entity)

The Order_Items table is the junction entity that resolves the many-to-many relationship between Orders and Products.

Each row in this table represents one line item in an order, specifying which product was ordered, in what quantity, and at what price.

- Primary Key: order_item_id
- Foreign Keys:
 - order_id → references Orders
 - product_card_id → references Products
- Attributes:

order_item_cardprod_id, order_item_quantity, order_item_product_price, order_item_discount, order_item_discount_rate, order_item_total, sales, profit, order_item_profit_ratio
- Relationship:
 - Many-to-One (M:1) with Orders
 - Many-to-One (M:1) with Products
- Purpose:
 - Enables accurate tracking of individual product sales within each order.
 - Contains financial metrics that support profitability and performance reporting.

Q - Why Orders ↔ Products is Many-to-Many:

In a business like SwiftLogix (my supply chain system):

- A single order often includes multiple products (for example, one customer orders a smartwatch, headphones, and a yoga mat together).
- A single product can appear in multiple orders (for example, “Smartwatch” is purchased by many customers in different orders).

So, both sides can have multiple associations, this is the definition of a many-to-many (M:N) relationship.

For Example:

Products Table

| product_card_id | product_name | product_price |
|-----------------|------------------|---------------|
| 101 | Smartwatch | 327.75 |
| 102 | Yoga Mat | 45.50 |
| 103 | Wireless Earbuds | 89.99 |

Orders Table

| order_id | customer_id | order_date | shipping_mode |
|----------|-------------|------------|----------------|
| 5001 | 200 | 2025-10-12 | Standard Class |
| 5002 | 201 | 2025-10-14 | Second Class |
| 5003 | 202 | 2025-10-15 | Same Day |

Q - What if we directly connected Orders ↔ Products:

We'd have to add multiple product IDs into one order record, like:

| order_id | product_card_ids |
|----------|------------------|
| 5001 | 101,102,103 |

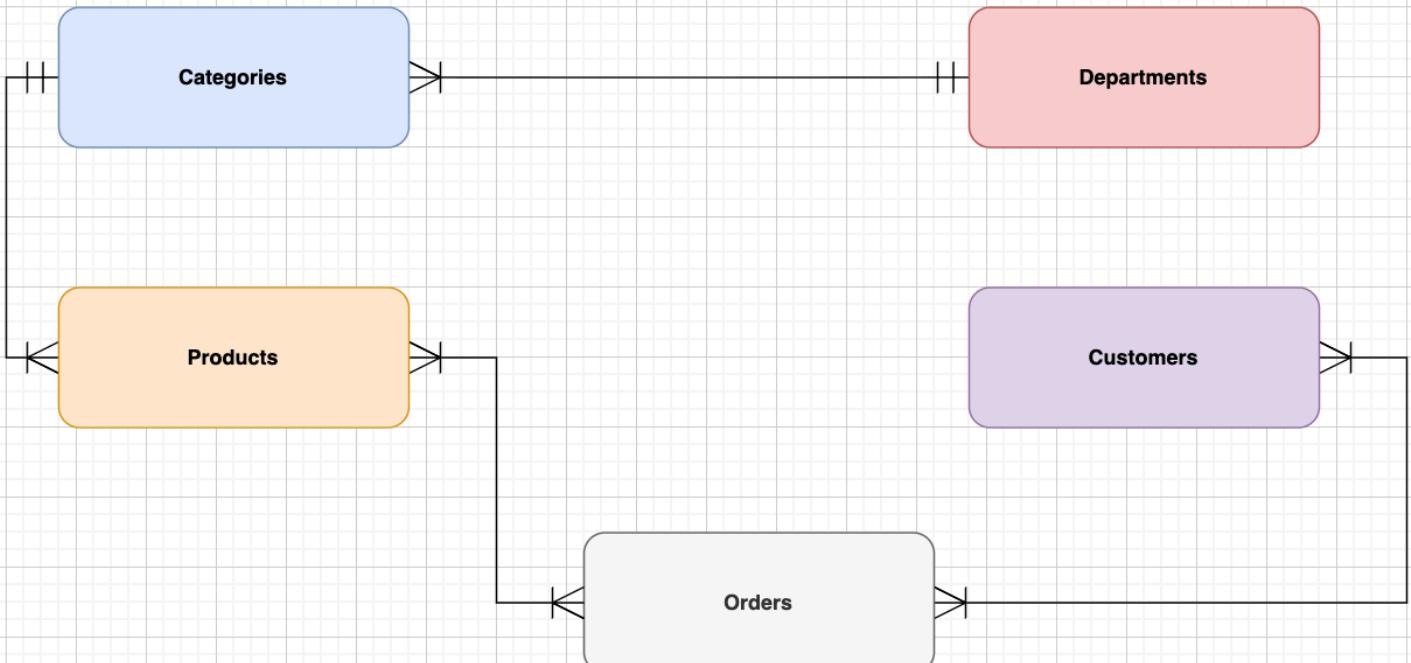
But relational databases don't allow multi-valued attributes, this violates First Normal Form (1NF). Also, if “Smartwatch” appears again in order 5002, I'd be duplicating the product data, which breaks data integrity and 3NF. So, we use a Junction Table: “Order_Items”

The junction table (Order_Items) transforms the many-to-many into two one-to-many relationships:

- Orders (1) → (M) Order_Items
- Products (1) → (M) Order_Items

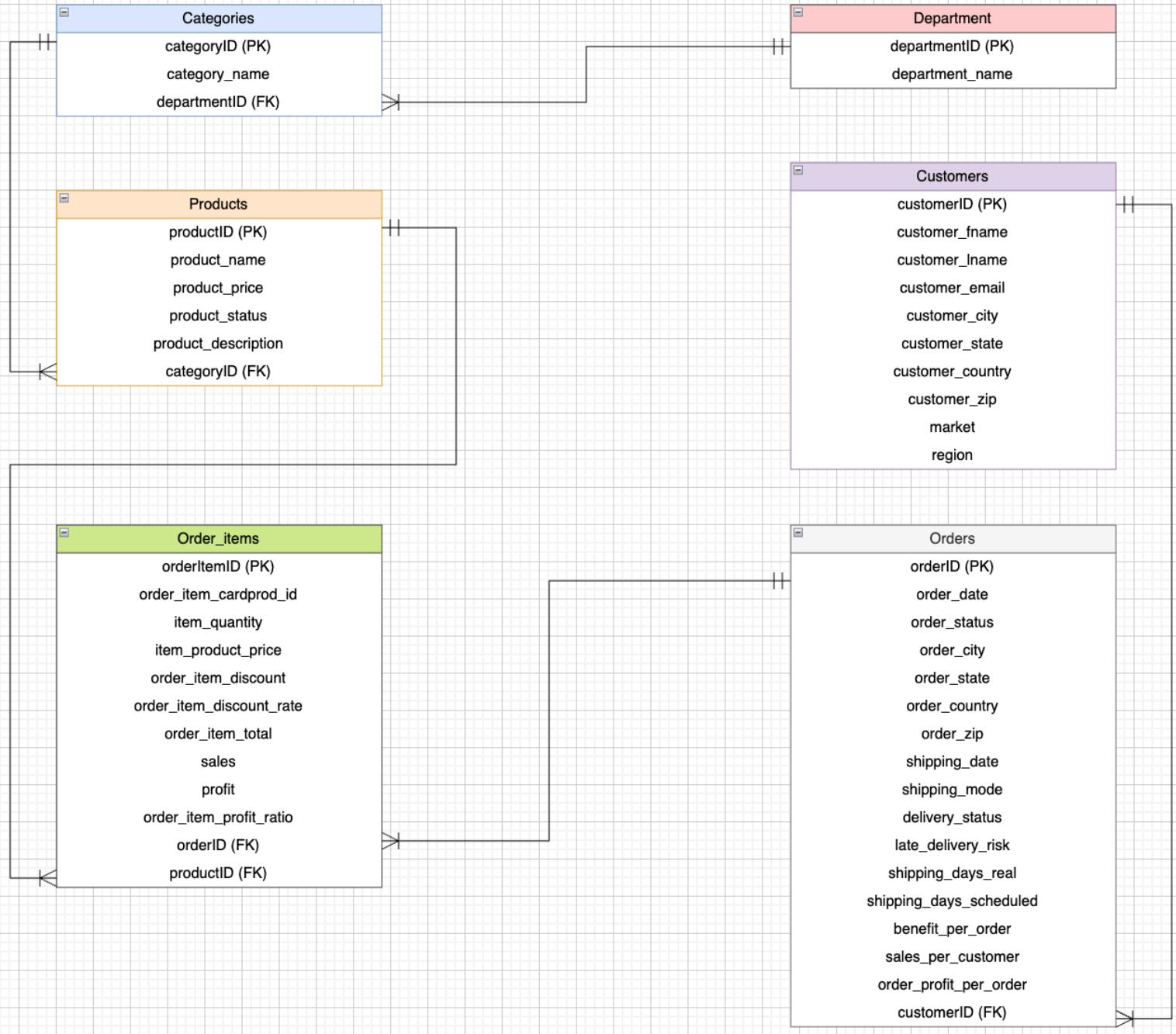
Conceptual Data Model

Conceptual Data Model



Logical Data Model

Logical Data Model



Physical Data Model

Physical Data Model

| Categories | | | |
|------------|---------------|---------|----------|
| PK | categoryID | int | NOT NULL |
| | category_name | varchar | NOT NULL |
| FK | departmentID | int | NOT NULL |

| Department | | | |
|------------|-----------------|---------|----------|
| PK | departmentID | int | NOT NULL |
| | department_name | varchar | NOT NULL |

| Products | | | |
|----------|---------------------|---------|----------|
| PK | productID | int | NOT NULL |
| | product_name | char | NOT NULL |
| | product_price | int | NOT NULL |
| | product_status | varchar | NOT NULL |
| FK | categoryID | int | NOT NULL |
| | product_description | varchar | NOT NULL |

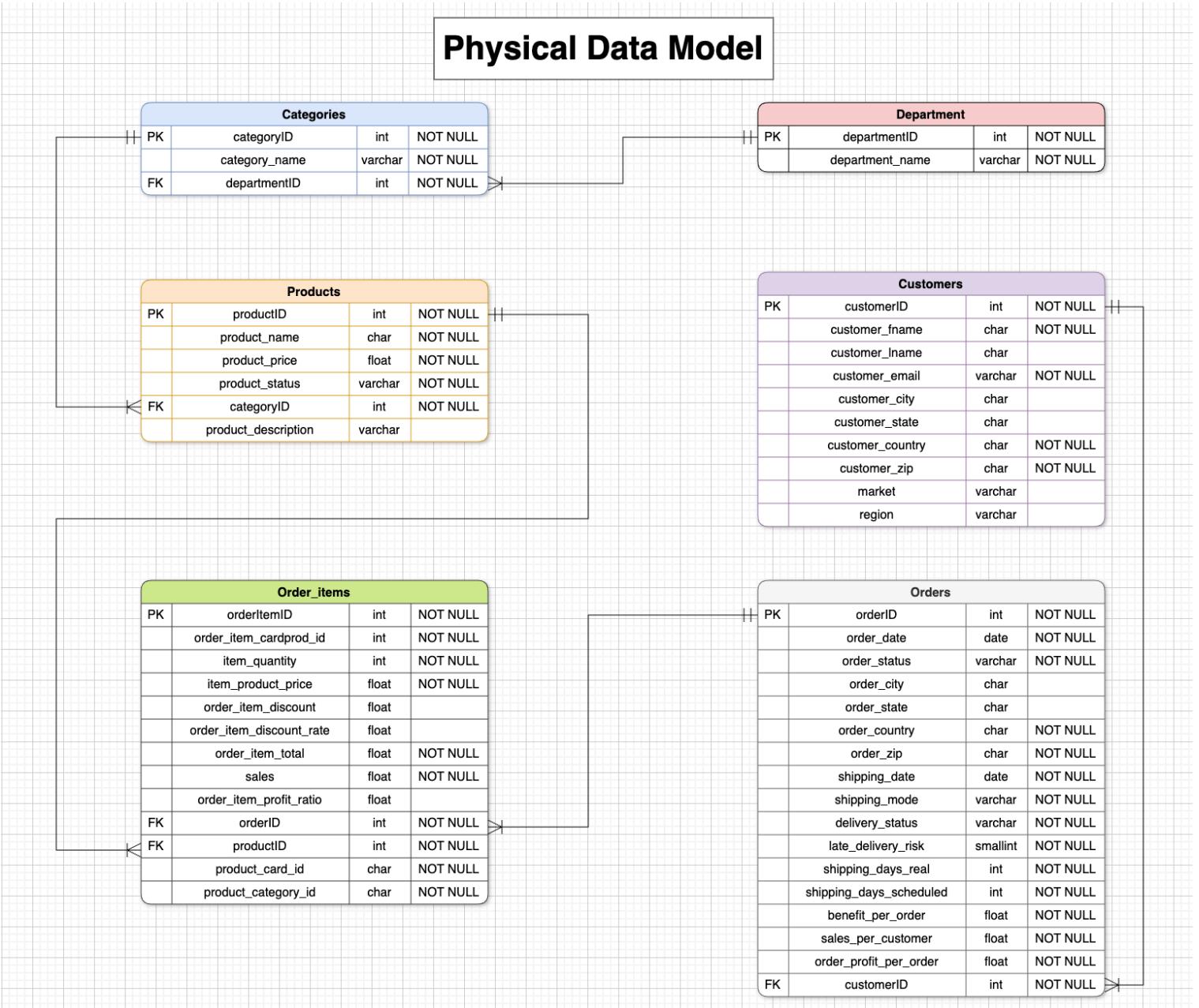
| Customers | | | |
|-----------|------------------|---------|----------|
| PK | customerID | int | NOT NULL |
| | customer_fname | char | NOT NULL |
| | customer_lname | char | |
| | customer_email | varchar | NOT NULL |
| | customer_city | char | |
| | customer_state | char | |
| | customer_country | char | NOT NULL |
| | customer_zip | int | NOT NULL |
| | market | varchar | |
| | region | varchar | |

| Order_items | | | |
|-------------|--------------------------|----------|----------|
| PK | orderItemID | int | NOT NULL |
| | order_item_cardprod_id | char | NOT NULL |
| | item_quantity | int | NOT NULL |
| | item_product_price | int | NOT NULL |
| | order_item_discount | smallint | |
| | order_item_discount_rate | int | |
| | order_item_total | int | NOT NULL |
| | sales | int | NOT NULL |
| | profit | int | NOT NULL |
| | order_item_profit_ratio | decimal | |
| FK | orderId | int | NOT NULL |
| FK | productID | int | NOT NULL |

| Orders | | | |
|--------|-------------------------|----------|----------|
| PK | orderID | int | NOT NULL |
| | order_date | char | NOT NULL |
| | order_status | varchar | NOT NULL |
| | order_city | char | |
| | order_state | char | |
| | order_country | char | NOT NULL |
| | order_zip | int | NOT NULL |
| | shipping_date | date | NOT NULL |
| | shipping_mode | varchar | NOT NULL |
| | delivery_status | varchar | NOT NULL |
| | late_delivery_risk | smallint | NOT NULL |
| | shipping_days_real | int | NOT NULL |
| | shipping_days_scheduled | int | NOT NULL |
| | benefit_per_order | int | NOT NULL |
| | sales_per_customer | int | NOT NULL |
| | order_profit_per_order | int | NOT NULL |
| FK | customerID | int | NOT NULL |

Final Project Report

Q. Develop the physical model based on the Logical Model.



Here's the updated physical data model. I've made several revisions after scraping the data and generating individual table files in Python using a Jupyter notebook.

I also adjusted the datatypes of several attributes during the data-cleaning process after examining the actual distribution and format of values, ensuring that each field was assigned the most appropriate and efficient datatype before loading it into MySQL.

Q. Create tables using a database system. Insert data into the database tables. You must provide the DDL (CREATE TABLE statements), INSERT statements, and SELECT statements.

Details: Create the tables that you have come up with (the table must be based on the Physical Model).

- (a) Columns, Primary Key (PK), Data Type, and length, and NULL/NOT NULL need to be implemented, per the Physical Model.
- (b) Show the table definition (DDL) that you implemented (not in a graphical view).
- (c) Insert the complete set of data that you have come up with and show the insert statements used.

```
1 • CREATE DATABASE swiftlogix_supply_chain;
```

100% 41:1

| Action Output | Time | Action | Response | Duration / Fetch Time |
|---------------|----------|---|-------------------|-----------------------|
| ✓ 1 | 00:15:38 | CREATE DATABASE swiftlogix_supply_chain | 1 row(s) affected | 0.053 sec |

Database Creation

```
1 • USE swiftlogix_supply_chain;
2
3 # Department Table
4 • (-) CREATE TABLE Departments(
5     departmentID INT NOT NULL PRIMARY KEY,
6     department_name VARCHAR(100) NOT NULL
7 );|
```

```
1 • USE swiftlogix_supply_chain;
2
3     # Categories Table
4 • ⊖ CREATE TABLE Categories(
5         categoryID INT NOT NULL PRIMARY KEY,
6         category_name VARCHAR(150) NOT NULL,
7         departmentID INT NOT NULL,
8         CONSTRAINT fk_categories_department
9             FOREIGN KEY (departmentID)
10            REFERENCES Departments(departmentID)
11            ON UPDATE RESTRICT
12            ON DELETE RESTRICT
13     );|
```

```
1 • USE swiftlogix_supply_chain;
2
3     # Products Table
4 • ⊖ CREATE TABLE Products(
5         productID BIGINT NOT NULL PRIMARY KEY,
6         product_name VARCHAR(255) NOT NULL,
7         product_price DECIMAL(12,2) NOT NULL,
8         product_status VARCHAR(50) NOT NULL,
9         product_description TEXT,
10        categoryID INT NOT NULL,
11        CONSTRAINT fk_products_categories
12            FOREIGN KEY (categoryID)
13            REFERENCES Categories(categoryID)
14            ON UPDATE RESTRICT
15            ON DELETE RESTRICT
16     );|
```

```
1 • USE swiftlogix_supply_chain;
2
3 # Customers Table
4 • ⊖ CREATE TABLE Customers (
5     customerID INT NOT NULL PRIMARY KEY,
6     customer_fname VARCHAR(100) NOT NULL,
7     customer_lname VARCHAR(100),
8     customer_email VARCHAR(255) NOT NULL,
9     customer_city VARCHAR(150),
10    customer_state VARCHAR(100),
11    customer_country VARCHAR(100) NOT NULL,
12    customer_zip VARCHAR(20) NOT NULL,
13    market VARCHAR(100),
14    region VARCHAR(100)
15 );
```

```
1 • USE swiftlogix_supply_chain;
2
3 # Orders Table
4 • ⊖ CREATE TABLE Orders (
5     orderID INT NOT NULL PRIMARY KEY,
6     order_date DATE NOT NULL,
7     order_status VARCHAR(50) NOT NULL,
8     order_city VARCHAR(150),
9     order_state VARCHAR(100),
10    order_country VARCHAR(100) NOT NULL,
11    order_zip VARCHAR(20) NOT NULL,
12    shipping_date DATE NOT NULL,
13    shipping_mode VARCHAR(50) NOT NULL,
14    delivery_status VARCHAR(100) NOT NULL,
15    late_delivery_risk SMALLINT NOT NULL,
16    shipping_days_real INT NOT NULL,
17    shipping_days_scheduled INT NOT NULL,
18    benefit_per_order NUMERIC(14,2) NOT NULL,
19    sales_per_customer NUMERIC(14,2) NOT NULL,
20    order_profit_per_order NUMERIC(14,2) NOT NULL,
21    customerID INT NOT NULL,
22    CONSTRAINT fk_orders_customer
23        FOREIGN KEY (customerID)
24        REFERENCES Customers(customerID)
25        ON UPDATE RESTRICT
26        ON DELETE RESTRICT
27 );|
```

```

1 • USE swiftlogix_supply_chain;
2
3 # Order_items Table
4 • ⊖ CREATE TABLE Order_items (
5     orderItemID INT NOT NULL PRIMARY KEY,
6     order_item_cardprod_id INT NOT NULL,
7     item_quantity INT NOT NULL,
8     item_product_price NUMERIC(12,2) NOT NULL,
9     order_item_discount NUMERIC(12,2),
10    order_item_discount_rate NUMERIC(7,4),
11    order_item_total NUMERIC(14,2) NOT NULL,
12    sales NUMERIC(14,2) NOT NULL,
13    order_item_profit_ratio NUMERIC(10,4),
14    orderID INT NOT NULL,
15    productID BIGINT NOT NULL,
16    product_card_id VARCHAR(50) NOT NULL,
17    product_category_id VARCHAR(50) NOT NULL,
18    CONSTRAINT fk_orderitems_order
19        FOREIGN KEY (orderID)
20        REFERENCES Orders(orderID)
21        ON UPDATE RESTRICT
22        ON DELETE CASCADE,
23    CONSTRAINT fk_orderitems_product
24        FOREIGN KEY (productID)
25        REFERENCES Products(productID)
26        ON UPDATE RESTRICT
27        ON DELETE RESTRICT
28 );

```

Q. Create a variety of SQL queries to retrieve data from one or many tables:

1. Retrieve the data from each table by using the SELECT * statement and order by PK column(s). Show the output. Make sure you show the print screen of the complete set of rows and columns. The rows must be ordered by PK column(s).

Screenshots:

```
1 * USE swiftlogix_supply_chain;
2
3 -- 1. Departments
4 * SELECT *
5 FROM Departments
6 ORDER BY departmentID;
```

100% ◁ 23:6

Result Grid Filter Rows: Search Edit: Export/Import:

| departmentID | departmentName |
|--------------|-------------------|
| 2 | Fitness |
| 3 | Footwear |
| 4 | Apparel |
| 5 | Golf |
| 6 | Outdoors |
| 7 | Fan Shop |
| 8 | Book Shop |
| 9 | Discs Shop |
| 10 | Technology |
| 11 | Pet Shop |
| 12 | Health and Beauty |
| NULL | NULL |

Departments

```

1 • USE swiftlogix_supply_chain;
2
3 -- 2. Categories
4 • SELECT *
5 FROM Categories
6 ORDER BY categoryID;
7

```

100% ▼ 29:1

Result Grid Filter Rows: Search Edit: Export/Import:

| | categoryID | category_name | department... | |
|--|--------------|---------------------|---------------|--|
| | 2 | Soccer | 2 | |
| | 3 | Baseball & Softball | 2 | |
| | 4 | Basketball | 2 | |
| | 5 | Lacrosse | 2 | |
| | 6 | Tennis & Racquet | 2 | |
| | 7 | Hockey | 2 | |
| | 9 | Cardio Equipment | 3 | |
| | 10 | Strength Training | 3 | |
| | 11 | Fitness Accessories | 3 | |
| | 12 | Boxing & MMA | 3 | |
| | 13 | Electronics | 3 | |
| | 16 | As Seen on TV! | 3 | |
| | 17 | Cleats | 4 | |
| | 18 | Men's Footwear | 4 | |
| | 24 | Women's Apparel | 5 | |
| | 26 | Girls' Apparel | 5 | |
| | 29 | Shop By Sport | 5 | |
| | 30 | Men's Golf Clubs | 6 | |
| | 31 | Women's Golf Clubs | 6 | |
| | 32 | Golf Apparel | 6 | |
| | 33 | Golf Shoes | 6 | |
| | 34 | Golf Bags & Carts | 6 | |
| | 35 | Golf Gloves | 6 | |
| | 36 | Golf Balls | 6 | |
| | 37 | Electronics | 6 | |
| | 38 | Kids' Golf Clubs | 6 | |
| | 40 | Accessories | 6 | |
| | 41 | Trade-In | 6 | |
| | 43 | Camping & Hiking | 7 | |
| | Categories 1 | | Categories | |

```

1 • USE swiftlogix_supply_chain;
2
3     -- 3. Products
4 • SELECT *
5     FROM Products
6     ORDER BY productID;
7

```

100% 29:1

Result Grid Filter Rows: Search Edit: Export/Import:

| productID | product_name | product_pri... | product_stat... | categoryID | product_descript... | |
|-----------|---|----------------|-----------------|------------|---------------------|--|
| 192 | Nike Men's Fingertrap Max Training Shoe | 124.99 | 0 | 2 | nan | |
| 242 | Elevation Training Mask 2.0 | 79.99 | 0 | 2 | nan | |
| 353 | adidas Brazuca 2014 Official Match Ball | 159.99 | 0 | 3 | nan | |
| 373 | adidas Kids' F5 Messi FG Soccer Cleat | 34.99 | 0 | 3 | nan | |
| 443 | adidas Men's F10 Messi TRX FG Soccer Cleat | 59.99 | 0 | 3 | nan | |
| 584 | Diamondback Boys' Insight 24 Performance Hybr | 299.99 | 0 | 4 | nan | |
| 604 | SOLE E25 Elliptical | 999.99 | 0 | 4 | nan | |
| 614 | Diamondback Girls' Clarity 24 Hybrid Bike 201 | 299.99 | 0 | 4 | nan | |
| 785 | Nike Kids' Grade School KD VI Basketball Shoe | 99.99 | 0 | 5 | nan | |
| 935 | Under Armour Men's Tech II T-Shirt | 24.99 | 0 | 5 | nan | |
| 1166 | Nike Men's Comfort 2 Slide | 44.99 | 0 | 6 | nan | |
| 1277 | Stiga Master Series ST3100 Competition Indoor | 329.99 | 0 | 7 | nan | |
| 1347 | Nike Women's Legend V-Neck T-Shirt | 25.00 | 0 | 7 | nan | |
| 1357 | Nike Dri-FIT Crew Sock 6 Pack | 22.00 | 0 | 7 | nan | |
| 1729 | Nike Women's Tempo Shorts | 30.00 | 0 | 9 | nan | |
| 1919 | Nike Men's Free 5.0+ Running Shoe | 99.99 | 0 | 9 | nan | |
| 20310 | GoPro HERO3+ Black Edition Camera | 399.99 | 0 | 10 | nan | |
| 20810 | SOLE E35 Elliptical | 1999.99 | 0 | 10 | nan | |
| 21610 | Yakima DoubleDown Ace Hitch Mount 4-Bike R... | 189.00 | 0 | 10 | nan | |
| 22611 | Bowflex SelectTech 1090 Dumbbells | 599.99 | 0 | 11 | nan | |
| 23511 | Under Armour Hustle Storm Medium Duffle Bag | 34.99 | 0 | 11 | nan | |
| 24912 | Under Armour Women's Micro G Skulpt Runnin... | 54.97 | 0 | 12 | nan | |
| 25112 | Brooks Women's Ghost 6 Running Shoe | 89.99 | 0 | 12 | nan | |
| 25812 | Nike Women's Free 5.0 TR FIT PRT 4 Training S | 94.99 | 0 | 12 | nan | |
| 27313 | Under Armour Kids' Mercenary Slide | 27.99 | 0 | 13 | nan | |
| 27613 | Under Armour Women's Ignite Slide | 31.99 | 0 | 13 | nan | |
| 27813 | Under Armour Men's Compression EV SL Slide | 44.99 | 0 | 13 | nan | |
| 28213 | Under Armour Women's Ignite PIP VI Slide | 31.99 | 0 | 13 | nan | |
| 29538 | Fitbit The One Wireless Activity & Sleep Trac | 99.95 | 0 | 38 | nan | |
| 30338 | Garmin Forerunner 910XT GPS Watch | 399.99 | 0 | 38 | nan | |

Products 2

Products

```

1 • USE swiftlogix_supply_chain;
2
3 -- 4. Customers
4 • SELECT *
5   FROM Customers
6   ORDER BY customerID;
7

```

100% | 29:1

Result Grid Filter Rows: Search Edit: Export/Import: Fetch rows:

| | customerID | customer_fn... | customer_ln... | customer_email | customer_ci... | customer_sta... | customer_coun... | customer_zip | market | region |
|----|-------------|----------------|----------------|----------------|----------------|-----------------|------------------|--------------|--------|--------|
| 1 | Richard | Hernandez | XXXXXXXXXX | Brownsville | TX | EE. UU. | 78521.0 | Pacific Asia | | |
| 2 | Mary | Barrett | XXXXXXXXXX | Littleton | CO | EE. UU. | 80126.0 | LATAM | | |
| 3 | Ann | Smith | XXXXXXXXXX | Caguas | PR | Puerto Rico | 725.0 | Pacific Asia | | |
| 4 | Mary | Jones | XXXXXXXXXX | San Marcos | CA | EE. UU. | 92069.0 | Pacific Asia | | |
| 5 | Robert | Hudson | XXXXXXXXXX | Caguas | PR | Puerto Rico | 725.0 | Africa | | |
| 6 | Mary | Smith | XXXXXXXXXX | Passaic | NJ | EE. UU. | 7055.0 | LATAM | | |
| 7 | Melissa | Wilcox | XXXXXXXXXX | Caguas | PR | Puerto Rico | 725.0 | Europe | | |
| 8 | Megan | Smith | XXXXXXXXXX | Lawrence | MA | EE. UU. | 1841.0 | Pacific Asia | | |
| 9 | Mary | Perez | XXXXXXXXXX | Caguas | PR | Puerto Rico | 725.0 | Europe | | |
| 10 | Melissa | Smith | XXXXXXXXXX | Stafford | VA | EE. UU. | 22554.0 | Africa | | |
| 11 | Mary | Huffman | XXXXXXXXXX | Caguas | PR | Puerto Rico | 725.0 | LATAM | | |
| 12 | Christopher | Smith | XXXXXXXXXX | San Antonio | TX | EE. UU. | 78227.0 | Africa | | |
| 13 | Mary | Baldwin | XXXXXXXXXX | Caguas | PR | Puerto Rico | 725.0 | Europe | | |
| 14 | Katherine | Smith | XXXXXXXXXX | Pico Rivera | CA | EE. UU. | 90660.0 | Pacific Asia | | |
| 15 | Jane | Luna | XXXXXXXXXX | Fontana | CA | EE. UU. | 92336.0 | Pacific Asia | | |
| 16 | Tiffany | Smith | XXXXXXXXXX | Caguas | PR | Puerto Rico | 725.0 | LATAM | | |
| 17 | Mary | Robinson | XXXXXXXXXX | Taylor | MI | EE. UU. | 48180.0 | LATAM | | |
| 18 | Robert | Smith | XXXXXXXXXX | Martinez | CA | EE. UU. | 94553.0 | LATAM | | |
| 19 | Stephanie | Mitchell | XXXXXXXXXX | Caguas | PR | Puerto Rico | 725.0 | Europe | | |
| 20 | Mary | Ellis | XXXXXXXXXX | West New Y... | NJ | EE. UU. | 7093.0 | Europe | | |
| 21 | William | Zimmerman | XXXXXXXXXX | Caguas | PR | Puerto Rico | 725.0 | Europe | | |
| 22 | Joseph | Smith | XXXXXXXXXX | North Bergen | NJ | EE. UU. | 7047.0 | Pacific Asia | | |
| 23 | Benjamin | Duarte | XXXXXXXXXX | San Juan | PR | Puerto Rico | 921.0 | LATAM | | |
| 24 | Mary | Smith | XXXXXXXXXX | Caguas | PR | Puerto Rico | 725.0 | Pacific Asia | | |
| 25 | Paul | Richardson | XXXXXXXXXX | Peoria | AZ | EE. UU. | 85345.0 | USCA | | |
| 26 | Johnny | Hood | XXXXXXXXXX | Glenview | IL | EE. UU. | 60025.0 | Pacific Asia | | |
| 27 | Mary | Vincent | XXXXXXXXXX | Caguas | PR | Puerto Rico | 725.0 | LATAM | | |
| 28 | Timothy | Smith | XXXXXXXXXX | Longview | WA | EE. UU. | 98632.0 | Pacific Asia | | |
| 29 | Mary | Humphrey | XXXXXXXXXX | Fort Worth | TX | EE. UU. | 76133.0 | LATAM | | |
| 30 | Patricia | Smith | XXXXXXXXXX | Omaha | NE | Puerto Rico | 705.0 | Pacific Asia | | |

Customers 1

Customers

```

1 • USE swiftlogix_supply_chain;
2
3 -- 5. Orders
4 • SELECT *
5   FROM Orders
6   ORDER BY orderID;
7

```

100% 29:1

| Result Grid | | | | | | | | | | | | | |
|-------------|---------------------|-----------------|------------------|------------------|----------------|-----------|---------------------|----------------|------------------|--------------------|--------------------|-----------------|--|
| orderID | order_date | order_status | order_city | order_state | order_count... | order_zip | shipping_date | shipping_mode | delivery_status | late_delivery_r... | shipping_days_r... | shipping_day... | |
| 1 | 2015-01-01 00:00:00 | CLOSED | Mexico City | Distrito Federal | México | Unknown | 2015-01-03 00:00:00 | Standard Class | Advance shipping | 0 | 2 | 4 | |
| 2 | 2015-01-01 00:21:00 | PENDING_PAYMENT | Dos Quebradas | Risaralda | Colombia | Unknown | 2015-01-04 00:21:00 | Standard Class | Advance shipping | 0 | 3 | 4 | |
| 4 | 2015-01-01 01:03:00 | CLOSED | Dos Quebradas | Risaralda | Colombia | Unknown | 2015-01-06 01:03:00 | Standard Class | Late delivery | 1 | 5 | 4 | |
| 5 | 2015-01-01 01:24:00 | COMPLETE | Dos Quebradas | Risaralda | Colombia | Unknown | 2015-01-07 01:24:00 | Standard Class | Late delivery | 1 | 6 | 4 | |
| 7 | 2015-01-01 02:06:00 | COMPLETE | São Paulo | São Paulo | Brasil | Unknown | 2015-01-04 02:06:00 | Second Class | Late delivery | 1 | 3 | 2 | |
| 8 | 2015-01-01 02:27:00 | PROCESSING | São Paulo | São Paulo | Brasil | Unknown | 2015-01-05 02:27:00 | Standard Class | Shipping on time | 0 | 4 | 4 | |
| 9 | 2015-01-01 02:48:00 | PENDING_PAYMENT | São Paulo | São Paulo | Brasil | Unknown | 2015-01-06 02:48:00 | Standard Class | Late delivery | 1 | 5 | 4 | |
| 10 | 2015-01-01 03:09:00 | PENDING_PAYMENT | São Paulo | São Paulo | Brasil | Unknown | 2015-01-07 03:09:00 | Standard Class | Late delivery | 1 | 6 | 4 | |
| 11 | 2015-01-01 03:30:00 | PAYMENT REVIEW | São Paulo | São Paulo | Brasil | Unknown | 2015-01-03 03:30:00 | Standard Class | Advance shipping | 0 | 2 | 4 | |
| 12 | 2015-01-01 03:51:00 | CLOSED | Managua | Managua | Nicaragua | Unknown | 2015-01-04 03:51:00 | Standard Class | Advance shipping | 0 | 3 | 4 | |
| 13 | 2015-01-01 04:12:00 | PENDING_PAYMENT | Managua | Managua | Nicaragua | Unknown | 2015-01-05 04:12:00 | Standard Class | Shipping on time | 0 | 4 | 4 | |
| 14 | 2015-01-01 04:33:00 | PROCESSING | Managua | Managua | Nicaragua | Unknown | 2015-01-06 04:33:00 | Standard Class | Late delivery | 1 | 5 | 4 | |
| 15 | 2015-01-01 04:54:00 | COMPLETE | Brasília | Distrito Federal | Brasil | Unknown | 2015-01-07 04:54:00 | Standard Class | Late delivery | 1 | 6 | 4 | |
| 16 | 2015-01-01 05:15:00 | PENDING_PAYMENT | Brasília | Distrito Federal | Brasil | Unknown | 2015-01-03 05:15:00 | Standard Class | Advance shipping | 0 | 2 | 4 | |
| 17 | 2015-01-01 05:36:00 | COMPLETE | San Miguelito | Panamá | Panamá | Unknown | 2015-01-04 05:36:00 | Standard Class | Advance shipping | 0 | 3 | 4 | |
| 18 | 2015-01-01 05:57:00 | CLOSED | San Miguelito | Panamá | Panamá | Unknown | 2015-01-05 05:57:00 | Standard Class | Shipping on time | 0 | 4 | 4 | |
| 19 | 2015-01-01 06:18:00 | PENDING_PAYMENT | San Miguelito | Panamá | Panamá | Unknown | 2015-01-06 06:18:00 | Standard Class | Late delivery | 1 | 5 | 4 | |
| 20 | 2015-01-01 06:39:00 | PROCESSING | San Miguelito | Panamá | Panamá | Unknown | 2015-01-07 06:39:00 | Standard Class | Late delivery | 1 | 6 | 4 | |
| 21 | 2015-01-01 07:00:00 | PENDING | San Cristóbal... | Chiapas | México | Unknown | 2015-01-03 07:00:00 | First Class | Late delivery | 1 | 2 | 1 | |
| 23 | 2015-01-01 07:42:00 | PENDING_PAYMENT | Santiago de C... | Santiago de... | Chile | Unknown | 2015-01-05 07:42:00 | Second Class | Late delivery | 1 | 4 | 2 | |
| 24 | 2015-01-01 08:03:00 | CLOSED | Panama City | Panamá | Panamá | Unknown | 2015-01-06 08:03:00 | Second Class | Late delivery | 1 | 5 | 2 | |
| 25 | 2015-01-01 08:24:00 | CLOSED | Hermosillo | Sonora | México | Unknown | 2015-01-07 08:24:00 | Second Class | Late delivery | 1 | 6 | 2 | |
| 27 | 2015-01-01 09:06:00 | PENDING_PAYMENT | Coyacán | Distrito Federal | México | Unknown | 2015-01-04 09:06:00 | Second Class | Late delivery | 1 | 3 | 2 | |
| 28 | 2015-01-01 09:27:00 | COMPLETE | Tegucigalpa | Francisco Mo... | Honduras | Unknown | 2015-01-05 09:27:00 | Standard Class | Shipping on time | 0 | 4 | 4 | |
| 29 | 2015-01-01 09:48:00 | PROCESSING | Tegucigalpa | Francisco Mo... | Honduras | Unknown | 2015-01-06 09:48:00 | Standard Class | Late delivery | 1 | 5 | 4 | |
| 30 | 2015-01-01 10:09:00 | PENDING_PAYMENT | Tegucigalpa | Francisco Mo... | Honduras | Unknown | 2015-01-07 10:09:00 | Standard Class | Late delivery | 1 | 6 | 4 | |
| 31 | 2015-01-01 10:30:00 | PAYMENT REVIEW | Tegucigalpa | Francisco Mo... | Honduras | Unknown | 2015-01-03 10:30:00 | Standard Class | Advance shipping | 0 | 2 | 4 | |
| 33 | 2015-01-01 11:12:00 | PENDING_PAYMENT | Tegucigalpa | Francisco Mo... | Honduras | Unknown | 2015-01-03 11:12:00 | First Class | Late delivery | 1 | 2 | 1 | |
| 34 | 2015-01-01 11:33:00 | PROCESSING | Tegucigalpa | Francisco Mo... | Honduras | Unknown | 2015-01-03 11:33:00 | First Class | Late delivery | 1 | 2 | 1 | |

Orders

```

1 • USE swiftlogix_supply_chain;
2
3 -- 6. Order_items
4 • SELECT *
5 FROM Order_items
6 ORDER BY orderItemID;
7

```

100% 29:1

Result Grid Filter Rows: Search Edit: Export/Import: Fetch rows:

| orderItemID | order_item_cardprod... | item_quantit... | item_product_pr... | order_item_discou... | order_item_discount_r... | order_item_to... | sales | order_item_profit_r... | orderID | productID | product_card... | product_catego... |
|-------------|------------------------|-----------------|--------------------|----------------------|--------------------------|------------------|--------|------------------------|---------|-----------|-----------------|-------------------|
| 1 | 957 | 1 | 299.98 | 60.00 | 0.2000 | 239.98 | 299.98 | 0.3700 | 1 | 95743 | 957 | 43 |
| 2 | 1073 | 1 | 199.99 | 6.00 | 0.0300 | 193.99 | 199.99 | 0.4700 | 2 | 107348 | 1073 | 48 |
| 3 | 502 | 5 | 50.00 | 22.50 | 0.0900 | 227.50 | 250.00 | 0.3000 | 2 | 50224 | 502 | 24 |
| 4 | 403 | 1 | 129.99 | 22.10 | 0.1700 | 107.89 | 129.99 | 0.3400 | 2 | 40318 | 403 | 18 |
| 5 | 897 | 2 | 24.99 | 9.00 | 0.1800 | 40.98 | 49.98 | 0.1000 | 4 | 89740 | 897 | 40 |
| 6 | 365 | 5 | 59.99 | 3.00 | 0.0100 | 296.95 | 299.95 | 0.0900 | 4 | 36517 | 365 | 17 |
| 7 | 502 | 3 | 50.00 | 27.00 | 0.1800 | 123.00 | 150.00 | 0.4900 | 4 | 50224 | 502 | 24 |
| 8 | 1014 | 4 | 49.98 | 39.98 | 0.2000 | 159.94 | 199.92 | 0.2100 | 4 | 101446 | 1014 | 46 |
| 9 | 957 | 1 | 299.98 | 54.00 | 0.1800 | 245.98 | 299.98 | 0.3800 | 5 | 95743 | 957 | 43 |
| 10 | 365 | 5 | 59.99 | 0.00 | 0.0000 | 299.95 | 299.95 | 0.4800 | 5 | 36517 | 365 | 17 |
| 11 | 1014 | 2 | 49.98 | 16.99 | 0.1700 | 82.97 | 99.96 | 0.1100 | 5 | 101446 | 1014 | 46 |
| 12 | 957 | 1 | 299.98 | 51.00 | 0.1700 | 248.98 | 299.98 | 0.4400 | 5 | 95743 | 957 | 43 |
| 13 | 403 | 1 | 129.99 | 20.80 | 0.1600 | 109.19 | 129.99 | 0.3200 | 5 | 40318 | 403 | 18 |
| 14 | 1073 | 1 | 199.99 | 4.00 | 0.0200 | 195.99 | 199.99 | 0.4000 | 7 | 107348 | 1073 | 48 |
| 15 | 957 | 1 | 299.98 | 48.00 | 0.1600 | 251.98 | 299.98 | 0.4800 | 7 | 95743 | 957 | 43 |
| 16 | 926 | 5 | 15.99 | 2.40 | 0.0300 | 77.55 | 79.95 | 0.0600 | 7 | 92641 | 926 | 41 |
| 17 | 365 | 3 | 59.99 | 30.59 | 0.1700 | 149.38 | 179.97 | 0.1100 | 8 | 36517 | 365 | 17 |
| 18 | 365 | 5 | 59.99 | 74.99 | 0.2500 | 224.96 | 299.95 | 0.3300 | 8 | 36517 | 365 | 17 |
| 19 | 1014 | 4 | 49.98 | 35.99 | 0.1800 | 163.93 | 199.92 | -0.8000 | 8 | 101446 | 1014 | 46 |
| 20 | 502 | 1 | 50.00 | 0.50 | 0.0100 | 49.50 | 50.00 | 0.0300 | 8 | 50224 | 502 | 24 |
| 21 | 191 | 2 | 99.99 | 10.00 | 0.0500 | 189.98 | 199.98 | 0.0200 | 9 | 1919 | 191 | 9 |
| 22 | 1073 | 1 | 199.99 | 2.00 | 0.0100 | 197.99 | 199.99 | 0.1000 | 9 | 107348 | 1073 | 48 |
| 23 | 1073 | 1 | 199.99 | 0.00 | 0.0000 | 199.99 | 199.99 | 0.4800 | 9 | 107348 | 1073 | 48 |
| 24 | 1073 | 1 | 199.99 | 50.00 | 0.2500 | 149.99 | 199.99 | 0.4800 | 10 | 107348 | 1073 | 48 |
| 25 | 1014 | 2 | 49.98 | 15.99 | 0.1600 | 83.97 | 99.96 | 0.2700 | 10 | 101446 | 1014 | 46 |
| 26 | 403 | 1 | 129.99 | 19.50 | 0.1500 | 110.49 | 129.99 | 0.2600 | 10 | 40318 | 403 | 18 |
| 27 | 917 | 1 | 21.99 | 0.66 | 0.0300 | 21.33 | 21.99 | -0.7300 | 10 | 91741 | 917 | 41 |
| 28 | 1073 | 1 | 199.99 | 40.00 | 0.2000 | 159.99 | 199.99 | -0.0900 | 10 | 107348 | 1073 | 48 |
| 29 | 365 | 1 | 59.99 | 12.00 | 0.2000 | 47.99 | 59.99 | 0.3500 | 11 | 36517 | 365 | 17 |

Order_items 1 Apply Re

Order_items

2. Write an SQL involving the junction table and two other related tables. You must use the INNER JOIN to connect with all three tables. The database that you created must be included in your SQL queries.

Screenshots:

```

1      # Question 2
2
3 • USE swiftlogix_supply_chain;
4
5 • SELECT
6      oi.orderItemID,
7      oi.orderID,
8      o.order_date,
9      oi.productID,
10     p.product_name,
11     oi.item_quantity,
12     oi.sales
13   FROM Order_items AS oi
14   INNER JOIN Orders   AS o ON oi.orderID  = o.orderID
15   INNER JOIN Products AS p ON oi.productID = p.productID
16   ORDER BY oi.orderItemID;
17

```

100% 25:16

Result Grid



Filter Rows:

Search

Export:



Fetch rows:



| orderItemID | orderID | order_date | productID | product_name | item_quanti... | sales |
|-------------|---------|---------------------|-----------|---|----------------|--------|
| 1 | 1 | 2015-01-01 00:00:00 | 95743 | Diamondback Women's Serene Classic Comfort... | 1 | 299.98 |
| 2 | 2 | 2015-01-01 00:21:00 | 107348 | Pelican Sunstream 100 Kayak | 1 | 199.99 |
| 3 | 2 | 2015-01-01 00:21:00 | 50224 | Nike Men's Dri-FIT Victory Golf Polo | 5 | 250.00 |
| 4 | 2 | 2015-01-01 00:21:00 | 40318 | Nike Men's CJ Elite 2 TD Football Cleat | 1 | 129.99 |
| 5 | 4 | 2015-01-01 01:03:00 | 89740 | Team Golf New England Patriots Putter Grip | 2 | 49.98 |
| 6 | 4 | 2015-01-01 01:03:00 | 36517 | Perfect Fitness Perfect Rip Deck | 5 | 299.95 |
| 7 | 4 | 2015-01-01 01:03:00 | 50224 | Nike Men's Dri-FIT Victory Golf Polo | 3 | 150.00 |
| 8 | 4 | 2015-01-01 01:03:00 | 101446 | O'Brien Men's Neoprene Life Vest | 4 | 199.92 |
| 9 | 5 | 2015-01-01 01:24:00 | 95743 | Diamondback Women's Serene Classic Comfort... | 1 | 299.98 |
| 10 | 5 | 2015-01-01 01:24:00 | 36517 | Perfect Fitness Perfect Rip Deck | 5 | 299.95 |
| 11 | 5 | 2015-01-01 01:24:00 | 101446 | O'Brien Men's Neoprene Life Vest | 2 | 99.96 |
| 12 | 5 | 2015-01-01 01:24:00 | 95743 | Diamondback Women's Serene Classic Comfort... | 1 | 299.98 |
| 13 | 5 | 2015-01-01 01:24:00 | 40318 | Nike Men's CJ Elite 2 TD Football Cleat | 1 | 129.99 |
| 14 | 7 | 2015-01-01 02:06:00 | 107348 | Pelican Sunstream 100 Kayak | 1 | 199.99 |
| 15 | 7 | 2015-01-01 02:06:00 | 95743 | Diamondback Women's Serene Classic Comfort... | 1 | 299.98 |
| 16 | 7 | 2015-01-01 02:06:00 | 92641 | Glove It Imperial Golf Towel | 5 | 79.95 |

Result 1

3. Write an SQL by including two or more tables and using the LEFT OUTER JOIN. Show the results and sort the results by key field(s). Interpret the results compared to what an INNER JOIN does.

```
1      # Question 3
2
3 •  USE swiftlogix_supply_chain;
4
5 •  SELECT
6      c.customerID,
7      c.customer_fname,
8      c.customer_lname,
9      o.orderID,
10     o.order_date
11    FROM Customers AS c
12    LEFT OUTER JOIN Orders AS o
13      ON c.customerID = o.customerID
14    ORDER BY c.customerID, o.orderID;
```

15
100%

34:14

Result Grid Filter Rows: Search Export: Fetch rows:

| | customerID | customer_fname | customer_lname | orderID | order_date | |
|---|------------|----------------|----------------|---------|---------------------|--|
| 1 | 1 | Richard | Hernandez | 22945 | 2015-12-01 22:18:00 | |
| 2 | 2 | Mary | Barrett | 15192 | 2015-08-10 18:05:00 | |
| 2 | 2 | Mary | Barrett | 33865 | 2016-05-09 08:04:00 | |
| 2 | 2 | Mary | Barrett | 57963 | 2017-04-26 02:40:00 | |
| 2 | 2 | Mary | Barrett | 67863 | 2017-09-17 15:04:00 | |
| 3 | 3 | Ann | Smith | 23662 | 2015-12-12 09:30:00 | |
| 3 | 3 | Ann | Smith | 35158 | 2016-05-28 05:03:00 | |
| 3 | 3 | Ann | Smith | 56178 | 2017-03-31 01:18:00 | |
| 3 | 3 | Ann | Smith | 57617 | 2017-04-21 01:27:00 | |
| 3 | 3 | Ann | Smith | 61453 | 2017-06-16 01:22:00 | |
| 4 | 4 | Mary | Jones | 9023 | 2015-05-12 16:48:00 | |
| 4 | 4 | Mary | Jones | 9704 | 2015-05-22 15:23:00 | |
| 4 | 4 | Mary | Jones | 49339 | 2016-12-21 05:17:00 | |
| 4 | 4 | Mary | Jones | 51157 | 2017-01-16 18:13:00 | |
| 5 | 5 | Robert | Hudson | 36472 | 2016-06-16 09:24:00 | |
| 5 | 5 | Robert | Hudson | 41333 | 2016-08-26 08:26:00 | |
| 5 | 5 | Robert | Hudson | 45832 | 2016-10-31 00:38:00 | |
| 6 | 6 | Mary | Smith | 7485 | 2015-04-20 05:58:00 | |
| 6 | 6 | Mary | Smith | 7797 | 2015-04-21 15:46:00 | |

Result 1

Interpretation:

LEFT OUTER JOIN returns all customers, including those who have no matching rows in Orders. For customers with no orders, orderID and order_date will be NULL.

If you changed this to an INNER JOIN, only customers who actually placed at least one order would appear. Customers with no orders would disappear from the result set.

4. Write a single-row subquery. Show the results and sort the results by key field(s). Interpret the output.

```
1      # Question 4
2
3 •   USE swiftlogix_supply_chain;
4
5 •   SELECT
6       o.orderID,
7       o.order_date,
8       o.benefit_per_order,
9       o.order_profit_per_order,
10      o.customerID
11     FROM Orders AS o
12     WHERE o.benefit_per_order = (
13         SELECT MAX(benefit_per_order)
14         FROM Orders
15     )
16     ORDER BY o.orderID;
```

100% ◇ 20:16

Result Grid Filter Rows: Search Edit: Export/Import:

| orderID | order_date | benefit_per_order | order_profit_per_order | customerID | |
|---------|---------------------|-------------------|------------------------|------------|--|
| 68883 | 2017-10-02 12:25:00 | 911.80 | 911.80 | 5533 | |

Interpretation:

The inner subquery returns a single value: the maximum benefit_per_order across all orders. The outer query then selects the order(s) whose benefit equals that maximum. This identifies the most profitable order(s) by benefit and orders the result by orderID.

5. Write a multiple-row subquery. Show the results and sort the results by key field(s). Interpret the output.

```
1      # Question 5
2
3 •  USE swiftlogix_supply_chain;
4
5 •  SELECT
6      c.customerID,
7      c.customer_fname,
8      c.customer_lname,
9      c.customer_country
10     FROM Customers AS c
11     ⊖ WHERE c.customerID IN (
12         SELECT o.customerID
13         FROM Orders AS o
14         GROUP BY o.customerID
15         HAVING COUNT(*) > 5
16     )
17     ORDER BY c.customerID;
```

100% | 23:17 |

Result Grid



Filter Rows:

Search

Edit:

| | customerID | customer_fname | customer_lname | customer_country | |
|--|------------|----------------|----------------|------------------|--|
| | 7 | Melissa | Wilcox | Puerto Rico | |
| | 8 | Megan | Smith | EE. UU. | |
| | 12 | Christopher | Smith | EE. UU. | |
| | 14 | Katherine | Smith | EE. UU. | |
| | 16 | Tiffany | Smith | Puerto Rico | |
| | 18 | Robert | Smith | EE. UU. | |
| | 19 | Stephanie | Mitchell | Puerto Rico | |
| | 23 | Benjamin | Duarte | Puerto Rico | |
| | 31 | Mary | Byrd | EE. UU. | |
| | 34 | Mary | Smith | Puerto Rico | |
| | 38 | Mary | Smith | EE. UU. | |
| | 40 | Mary | Smith | EE. UU. | |
| | 42 | Ethan | Smith | EE. UU. | |
| | 44 | Howard | Smith | EE. UU. | |
| | 51 | Jessica | Smith | Puerto Rico | |

Customers 1

Interpretation:

The subquery returns multiple customerIDs (one per customer with more than 5 orders). The outer query then returns full customer details for those IDs. This identifies high-activity customers and sorts them by primary key.

6. Write an SQL to aggregate the results by using multiple columns in the SELECT clause. Interpret the output.

```
1      # Question 6
2
3 •  USE swiftlogix_supply_chain;
4
5 •  SELECT
6      d.departmentID,
7      d.department_name,
8      c.categoryID,
9      c.category_name,
10     SUM(oi.sales)          AS total_sales,
11     SUM(oi.order_item_total) AS total_item_amount,
12     AVG(oi.order_item_profit_ratio) AS avg_profit_ratio
13   FROM Departments AS d
14   JOIN Categories AS c  ON c.departmentID = d.departmentID
15   JOIN Products    AS p  ON p.categoryID    = c.categoryID
16   JOIN Order_items AS oi ON oi.productID  = p.productID
17 GROUP BY
18      d.departmentID,
19      d.department_name,
20      c.categoryID,
21      c.category_name
22 ORDER BY
23      d.departmentID,
24      c.categoryID;
25
```

23:19

Result Grid Filter Rows: Search Export:

| departmentID | department_name | categoryID | category_name | total_sales | total_item_amount | avg_profit_ratio | |
|--------------|-----------------|------------|---------------------|-------------|-------------------|------------------|--|
| 2 | Fitness | 2 | Soccer | 26477.05 | 23909.95 | 0.14652174 | |
| 2 | Fitness | 3 | Baseball & Softball | 94057.15 | 84367.27 | 0.13795886 | |
| 2 | Fitness | 4 | Basketball | 27099.33 | 24705.33 | 0.10805970 | |
| 2 | Fitness | 5 | Lacrosse | 39464.79 | 35415.93 | 0.13142857 | |
| 2 | Fitness | 6 | Tennis & Racquet | 44585.09 | 40077.81 | 0.13359756 | |
| 2 | Fitness | 7 | Hockey | 48360.73 | 43618.36 | 0.12796417 | |
| 2 | Fitness | 73 | Sporting Goods | 117006.75 | 105063.61 | 0.12106443 | |
| 3 | Footwear | 9 | Cardio Equipment | 3694843.20 | 3320250.72 | 0.11883479 | |

Result 1

Interpretation:

This query aggregates several numeric measures total_sales, total_item_amount and avg_profit_ratio for each department–category combination. It shows which departments and categories contribute the most revenue and profit, which is crucial for supply chain planning (e.g., prioritizing inventory and logistics capacity).

7. Write a subquery using the NOT IN operator. Show the results and sort the results by key field(s). Interpret the output.

```
1      # Question 7
2
3 •   USE swiftlogix_supply_chain;
4
5 •   SELECT
6       c.customerID,
7       c.customer_fname,
8       c.customer_lname,
9       c.customer_country
10      FROM Customers AS c
11      WHERE c.customerID NOT IN (
12          SELECT o.customerID
13          FROM Orders AS o
14          GROUP BY o.customerID
15          HAVING COUNT(*) > 5
16      )
17      ORDER BY c.customerID;
```

Result Grid Filter Rows: Search Edit: Export/Import:

| customerID | customer_fname | customer_lname | customer_country |
|------------|----------------|----------------|------------------|
| 1 | Richard | Hernandez | EE. UU. |
| 2 | Mary | Barrett | EE. UU. |
| 3 | Ann | Smith | Puerto Rico |
| 4 | Mary | Jones | EE. UU. |
| 5 | Robert | Hudson | Puerto Rico |
| 6 | Mary | Smith | EE. UU. |
| 9 | Mary | Perez | Puerto Rico |
| 10 | Melissa | Smith | EE. UU. |
| 11 | Mary | Huffman | Puerto Rico |
| 13 | Mary | Baldwin | Puerto Rico |
| 15 | Jane | Luna | EE. UU. |
| 17 | Mary | Robinson | EE. UU. |
| 20 | Mary | Ellis | EE. UU. |
| 21 | William | Zimmerman | Puerto Rico |
| 22 | Joseph | Smith | EE. UU. |

Customers 4

Interpretation:

This query uses a subquery with the NOT IN operator to identify customers who are not in the group of high-frequency buyers (customers with more than 5 orders).

The inner query returns all customer IDs with more than 5 orders, and the outer query returns customers whose IDs are not in that set.

These customers represent low- to medium-activity segments, which can be useful for targeted marketing campaigns, retention strategies, or analyzing how order frequency varies across different customer groups.

8. Write a query using a CASE statement. Show the results and sort the results by key field(s). Interpret the output.

```
1      # Question 8
2
3 *  USE swiftlogix_supply_chain;
4
5 *  SELECT
6      o.orderID,
7      o.order_date,
8      o.shipping_date,
9      o.late_delivery_risk,
10     CASE
11         WHEN o.late_delivery_risk = 1 THEN 'Late delivery'
12         ELSE 'On time / on-schedule'
13     END AS delivery_status_flag
14     FROM Orders AS o
15     ORDER BY o.orderID;
```

16

100% 20:15

Result Grid Filter Rows: Search Export: Fetch rows:

| orderID | order_date | shipping_date | late_delivery_risk | delivery_status_flag |
|---------|---------------------|---------------------|--------------------|-----------------------|
| 1 | 2015-01-01 00:00:00 | 2015-01-03 00:00:00 | 0 | On time / on-schedule |
| 2 | 2015-01-01 00:21:00 | 2015-01-04 00:21:00 | 0 | On time / on-schedule |
| 4 | 2015-01-01 01:03:00 | 2015-01-06 01:03:00 | 1 | Late delivery |
| 5 | 2015-01-01 01:24:00 | 2015-01-07 01:24:00 | 1 | Late delivery |
| 7 | 2015-01-01 02:06:00 | 2015-01-04 02:06:00 | 1 | Late delivery |
| 8 | 2015-01-01 02:27:00 | 2015-01-05 02:27:00 | 0 | On time / on-schedule |
| 9 | 2015-01-01 02:48:00 | 2015-01-06 02:48:00 | 1 | Late delivery |
| 10 | 2015-01-01 03:09:00 | 2015-01-07 03:09:00 | 1 | Late delivery |
| 11 | 2015-01-01 03:30:00 | 2015-01-03 03:30:00 | 0 | On time / on-schedule |
| 12 | 2015-01-01 03:51:00 | 2015-01-04 03:51:00 | 0 | On time / on-schedule |
| 13 | 2015-01-01 04:12:00 | 2015-01-05 04:12:00 | 0 | On time / on-schedule |
| 14 | 2015-01-01 04:33:00 | 2015-01-06 04:33:00 | 1 | Late delivery |
| 15 | 2015-01-01 04:54:00 | 2015-01-07 04:54:00 | 1 | Late delivery |
| 16 | 2015-01-01 05:15:00 | 2015-01-03 05:15:00 | 0 | On time / on-schedule |
| 17 | 2015-01-01 05:36:00 | 2015-01-04 05:36:00 | 0 | On time / on-schedule |
| 18 | 2015-01-01 05:57:00 | 2015-01-05 05:57:00 | 0 | On time / on-schedule |
| 19 | 2015-01-01 06:18:00 | 2015-01-06 06:18:00 | 1 | Late delivery |

Result 1

Interpretation:

The CASE statement translates the numeric late_delivery_risk flag into a human-readable label. This makes it easier to quickly see which orders were late versus on time, and can be used later in reports or dashboards for supply chain performance analysis.

9. Write a query using the NOT EXISTS operator. Show the results and sort the results by key field(s). Interpret the output.

```
1      # Question 9
2
3 •   USE swiftlogix_supply_chain;
4
5 •   SELECT
6       p.productID,
7       p.product_name,
8       p.product_status,
9       p.categoryID
10      FROM Products AS p
11      WHERE NOT EXISTS (
12          SELECT 1
13          FROM Order_items AS oi
14          WHERE oi.productID = p.productID
15      )
16      ORDER BY p.productID;
```

100% 22:16

Result Grid



Filter Rows:



Search

Edit:



| productID | product_na... | product_stat... | categoryID |
|-----------|---------------|-----------------|------------|
| NULL | NULL | NULL | NULL |

Interpretation:

For each product, the NOT EXISTS subquery checks whether there is at least one matching row in Order_items. If no such row exists, the product has never appeared in any order. These are “dead inventory” or slow-moving SKUs that might need marketing action, price adjustment, or de-stocking.

10. Write a subquery using the NOT NULL operator in the inner query. Show the results and sort the results by key field(s). Interpret the output.

```
1  # Question 10
2
3 • USE swiftlogix_supply_chain;
4
5 • SELECT
6      o.orderID,
7      o.order_date,
8      o.customerID,
9      o.benefit_per_order,
10     o.order_profit_per_order
11    FROM Orders AS o
12   WHERE o.orderID IN (
13       SELECT DISTINCT oi.orderID
14       FROM Order_items AS oi
15      WHERE oi.order_item_discount IS NOT NULL
16   )
17   ORDER BY o.orderID;|
```

100% 20:17

Result Grid Filter Rows: Search Edit: Export/Import:

| | orderID | order_date | customerID | benefit_per_order | order_profit_per_order | |
|----|---------|---------------------|------------|-------------------|------------------------|--|
| 1 | 1 | 2015-01-01 00:00:00 | 11599 | 88.79 | 88.79 | |
| 2 | 2 | 2015-01-01 00:21:00 | 256 | 91.18 | 91.18 | |
| 4 | 4 | 2015-01-01 01:03:00 | 8827 | 33.59 | 33.59 | |
| 5 | 5 | 2015-01-01 01:24:00 | 11318 | 9.38 | 9.38 | |
| 7 | 7 | 2015-01-01 02:06:00 | 4530 | 120.95 | 120.95 | |
| 8 | 8 | 2015-01-01 02:27:00 | 2911 | 73.11 | 73.11 | |
| 9 | 9 | 2015-01-01 02:48:00 | 5657 | 3.80 | 3.80 | |
| 10 | 10 | 2015-01-01 03:09:00 | 5648 | 72.00 | 72.00 | |
| 11 | 11 | 2015-01-01 03:30:00 | 918 | 117.17 | 117.17 | |
| 12 | 12 | 2015-01-01 03:51:00 | 1837 | 23.49 | 23.49 | |
| 13 | 13 | 2015-01-01 04:12:00 | 9149 | 47.09 | 47.09 | |
| 14 | 14 | 2015-01-01 04:33:00 | 9842 | -10.62 | -10.62 | |
| 15 | 15 | 2015-01-01 04:54:00 | 2568 | -254.19 | -254.19 | |
| 16 | 16 | 2015-01-01 05:15:00 | 7276 | 24.00 | 24.00 | |
| 17 | 17 | 2015-01-01 05:36:00 | 2667 | 240 | 240 | |

Orders 1

Interpretation:

The inner subquery selects all orderIDs where at least one order item has a non-NULL discount value. The outer query then returns full order information for those discounted orders. This helps identify orders where discounts were applied and lets you analyze their impact on benefit and profit.

Summary:

In this project, I designed and implemented a complete relational database system aimed at supporting supply chain logistics analysis, beginning from a raw, multi-attribute Excel dataset and progressing through data modeling, transformation, loading, and analytical querying. I first examined the dataset to identify the primary entities relevant to supply chain operations, Departments, Categories, Products, Customers, Orders, and Order_Items. Based on these entities, I created a conceptual model that outlined the relationships and dependencies among the data elements, and then translated this model into a physical schema with carefully defined primary keys, foreign keys, and integrity constraints. This step ensured that the database structure accurately represented real world supply chain workflows. I then used Python and Pandas to perform detailed data preparation. This included systematically cleaning the dataset, resolving duplicates, enforcing mandatory attributes, correcting inconsistent datatypes, and engineering new attributes such as the unified ProductID derived from concatenating product_card_id and product_category_id. I also reorganized columns to match the physical schema, aggregated order-level fields to eliminate conflicting values, and validated referential integrity by checking for missing parent records across Orders, Products, and Customers. Through multiple iterations, I ensured each entity dataset adhered strictly to the database constraints so that the MySQL import process could occur without structural conflicts.

Following the data preparation phase, I imported the cleaned datasets into MySQL Workbench and configured the database with appropriate integrity rules and data types. During this stage, I encountered practical issues such as foreign key constraint failures, rejected rows from the import wizard, discrepancies between the Python-processed data and the SQL-stored results, and MySQL restrictions related to date formats and local file permissions. I resolved these challenges by refining the CSV outputs, adjusting datatype declarations, correcting parent-child mismatches, enabling local file loading, and verifying that the imported tables preserved the intended relational structure. Once the database was fully populated, I executed a comprehensive set of SQL analyses to demonstrate its functional value. These included INNER and LEFT OUTER JOIN operations, single-row and multiple-row subqueries, CASE-based conditional logic, group-based aggregations, and queries involving NOT IN and NOT EXISTS to test the database's relational depth. Each query helped reveal operational insights into product assortment, category performance, customer ordering behavior, delivery timelines, financial outcomes, and logistics risk patterns. Completing this project allowed me to transform a complex, unrefined dataset into a structured, query-ready system capable of supporting supply chain monitoring, decision-making, and analytical reporting. The resulting database provides

a scalable foundation for future enhancements such as forecasting, segmentation, and real-time logistics dashboards, fully aligning with the original objective of improving transparency and analytical readiness within supply chain operations.