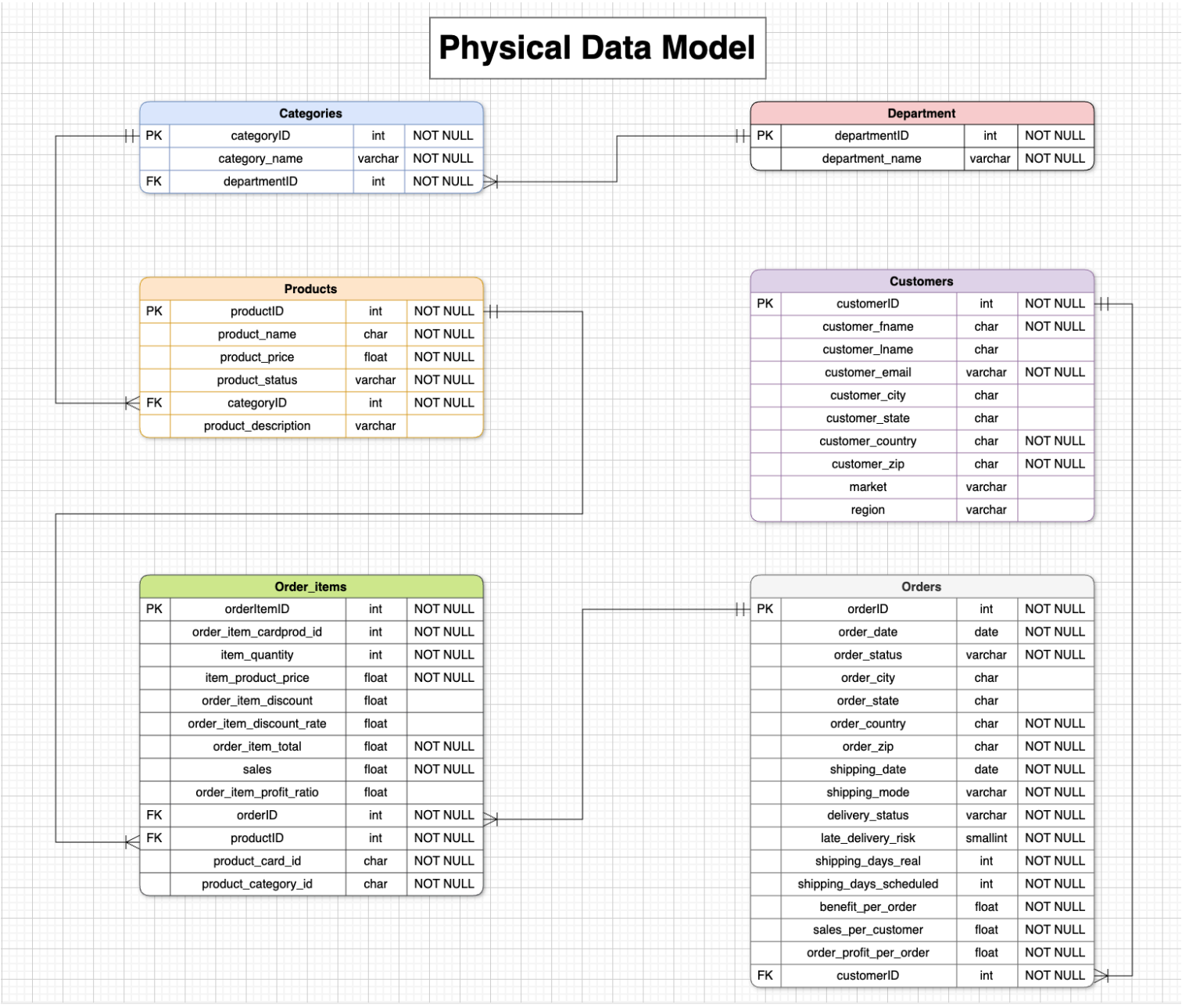
Final Project Report

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



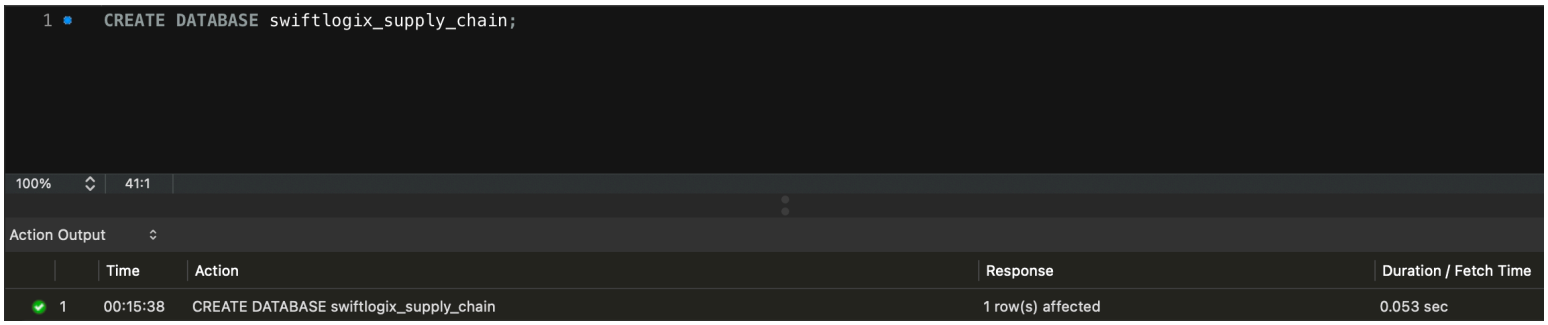
Here's the updated physical data model. I've made several revisions after scraping the data and generating individual table files in Python using a Jupyter notebook.

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.



Database Creation

```sql
USE swiftlogix_supply_chain;

# Department Table
CREATE TABLE Departments(
    departmentID INT NOT NULL PRIMARY KEY,
    department_name VARCHAR(100) NOT NULL
);
```

```sql
USE swiftlogix_supply_chain;

# Categories Table
CREATE TABLE Categories(
    categoryID INT NOT NULL PRIMARY KEY,
    category_name VARCHAR(150) NOT NULL,
    departmentID INT NOT NULL,
    CONSTRAINT fk_categories_department
        FOREIGN KEY (departmentID)
        REFERENCES Departments(departmentID)
        ON UPDATE RESTRICT
        ON DELETE RESTRICT
);
```

```sql
USE swiftlogix_supply_chain;

# Products Table
CREATE TABLE Products(
    productID BIGINT NOT NULL PRIMARY KEY,
    product_name VARCHAR(255) NOT NULL,
    product_price DECIMAL(12,2) NOT NULL,
    product_status VARCHAR(50) NOT NULL,
    product_description TEXT,
    categoryID INT NOT NULL,
    CONSTRAINT fk_products_categories
        FOREIGN KEY (categoryID)
        REFERENCES Categories(categoryID)
        ON UPDATE RESTRICT
        ON DELETE RESTRICT
);
```

```sql
USE swiftlogix_supply_chain;

# Customers Table
CREATE TABLE Customers (
    customerID INT NOT NULL PRIMARY KEY,
    customer_fname VARCHAR(100) NOT NULL,
    customer_lname VARCHAR(100),
    customer_email VARCHAR(255) NOT NULL,
    customer_city VARCHAR(150),
    customer_state VARCHAR(100),
    customer_country VARCHAR(100) NOT NULL,
    customer_zip VARCHAR(20) NOT NULL,
    market VARCHAR(100),
    region VARCHAR(100)
);
```

```sql
USE swiftlogix_supply_chain;

# Orders Table
CREATE TABLE Orders (
    orderID INT NOT NULL PRIMARY KEY,
    order_date DATE NOT NULL,
    order_status VARCHAR(50) NOT NULL,
    order_city VARCHAR(150),
    order_state VARCHAR(100),
    order_country VARCHAR(100) NOT NULL,
    order_zip VARCHAR(20) NOT NULL,
    shipping_date DATE NOT NULL,
    shipping_mode VARCHAR(50) NOT NULL,
    delivery_status VARCHAR(100) NOT NULL,
    late_delivery_risk SMALLINT NOT NULL,
    shipping_days_real INT NOT NULL,
    shipping_days_scheduled INT NOT NULL,
    benefit_per_order NUMERIC(14,2) NOT NULL,
    sales_per_customer NUMERIC(14,2) NOT NULL,
    order_profit_per_order NUMERIC(14,2) NOT NULL,
    customerID INT NOT NULL,
    CONSTRAINT fk_orders_customer
        FOREIGN KEY (customerID)
        REFERENCES Customers(customerID)
        ON UPDATE RESTRICT
        ON DELETE RESTRICT
);
```

```sql
USE swiftlogix_supply_chain;

# Order_items Table
CREATE TABLE Order_items (
    orderItemID INT NOT NULL PRIMARY KEY,
    order_item_cardprod_id INT NOT NULL,
    item_quantity INT NOT NULL,
    item_product_price NUMERIC(12,2) NOT NULL,
    order_item_discount NUMERIC(12,2),
    order_item_discount_rate NUMERIC(7,4),
    order_item_total NUMERIC(14,2) NOT NULL,
    sales NUMERIC(14,2) NOT NULL,
    order_item_profit_ratio NUMERIC(10,4),
    orderID INT NOT NULL,
    productID BIGINT NOT NULL,
    product_card_id VARCHAR(50) NOT NULL,
    product_category_id VARCHAR(50) NOT NULL,
    CONSTRAINT fk_orderitems_order
        FOREIGN KEY (orderID)
        REFERENCES Orders(orderID)
        ON UPDATE RESTRICT
        ON DELETE CASCADE,
    CONSTRAINT fk_orderitems_product
        FOREIGN KEY (productID)
        REFERENCES Products(productID)
        ON UPDATE RESTRICT
        ON DELETE RESTRICT
);
```

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:

```sql
USE swiftlogix_supply_chain;

-- 1. Departments
SELECT *
FROM Departments
ORDER BY departmentID;
```

100% 23:6

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

department...	department_na...
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

```sql
USE swiftlogix_supply_chain;

-- 2. Categories
SELECT *
FROM Categories
ORDER BY categoryID;

```

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

```sql
USE swiftlogix_supply_chain;

-- 3. Products
SELECT *
FROM Products
ORDER BY productID;
```

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

```sql
USE swiftlogix_supply_chain;

-- 4. Customers
SELECT *
FROM Customers
ORDER BY customerID;
```

100% 29:1

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

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

Customers 1

Customers

```sql
USE swiftlogix_supply_chain;

-- 5. Orders
SELECT *
FROM Orders
ORDER BY orderID;
```

100% 29:1

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

orderID	order_date	order_status	order_city	order_state	order_count...	order_zip	shipping_date	shipping_mode	delivery_status	late_delivery_ri...	shipping_days_r...	shipping_day
1	2015-01-01 00:00:00	CLOSED	Mexico City	Distrito Federal	M□xico	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	S□o Paulo	S□o Paulo	Brasil	Unknown	2015-01-04 02:06:00	Second Class	Late delivery	1	3	2
8	2015-01-01 02:27:00	PROCESSING	S□o Paulo	S□o 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	S□o Paulo	S□o 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	S□o Paulo	S□o 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	S□o Paulo	S□o 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	Bras□lia	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	Bras□lia	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	Panam□	Panam□	Unknown	2015-01-04 05:36:00	Standard Class	Advance shipping	0	3	4
18	2015-01-01 05:57:00	CLOSED	San Miguelito	Panam□	Panam□	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	Panam□	Panam□	Unknown	2015-01-06 06:18:00	Standard Class	Late delivery	1	5	4
20	2015-01-01 06:39:00	PROCESSING	San Miguelito	Panam□	Panam□	Unknown	2015-01-07 06:39:00	Standard Class	Late delivery	1	6	4
21	2015-01-01 07:00:00	PENDING	San Crist□bal...	Chiapas	M□xico	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	Panam□	Panam□	Unknown	2015-01-06 08:03:00	Second Class	Late delivery	1	5	2
25	2015-01-01 08:24:00	CLOSED	Hermosillo	Sonora	M□xico	Unknown	2015-01-07 08:24:00	Second Class	Late delivery	1	6	2
27	2015-01-01 09:06:00	PENDING_PAYMENT	Coyoac□n	Distrito Federal	M□xico	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 | Apply | Rev

Orders

```sql
USE swiftlogix_supply_chain;

-- 6. Order_items
SELECT *
FROM Order_items
ORDER BY orderItemID;
```

100% 29:1

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

orderItemID	order_item_cardprod...	item_quanti...	item_product_pr...	order_item_disco...	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

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:

```sql
# Question 2

USE swiftlogix_supply_chain;

SELECT
    oi.orderItemID,
    oi.orderID,
    o.order_date,
    oi.productID,
    p.product_name,
    oi.item_quantity,
    oi.sales
FROM Order_items AS oi
INNER JOIN Orders   AS o ON oi.orderID  = o.orderID
INNER JOIN Products AS p ON oi.productID = p.productID
ORDER BY oi.orderItemID;
```

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.

```sql
# Question 3

USE swiftlogix_supply_chain;

SELECT
    c.customerID,
    c.customer_fname,
    c.customer_lname,
    o.orderID,
    o.order_date
FROM Customers AS c
LEFT OUTER JOIN Orders AS o
    ON c.customerID = o.customerID
ORDER BY c.customerID, o.orderID;
```

100% 34:14

Result Grid Filter Rows: Search Export: Fetch rows:

customerID	customer_fna...	customer_lname	orderID	order_date
1	Richard	Hernandez	22945	2015-12-01 22:18:00
2	Mary	Barrett	15192	2015-08-10 18:05:00
2	Mary	Barrett	33865	2016-05-09 08:04:00
2	Mary	Barrett	57963	2017-04-26 02:40:00
2	Mary	Barrett	67863	2017-09-17 15:04:00
3	Ann	Smith	23662	2015-12-12 09:30:00
3	Ann	Smith	35158	2016-05-28 05:03:00
3	Ann	Smith	56178	2017-03-31 01:18:00
3	Ann	Smith	57617	2017-04-21 01:27:00
3	Ann	Smith	61453	2017-06-16 01:22:00
4	Mary	Jones	9023	2015-05-12 16:48:00
4	Mary	Jones	9704	2015-05-22 15:23:00
4	Mary	Jones	49339	2016-12-21 05:17:00
4	Mary	Jones	51157	2017-01-16 18:13:00
5	Robert	Hudson	36472	2016-06-16 09:24:00
5	Robert	Hudson	41333	2016-08-26 08:26:00
5	Robert	Hudson	45832	2016-10-31 00:38:00
6	Mary	Smith	7485	2015-04-20 05:58:00
6	Mary	Smith	7787	2015-04-24 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.

```sql
# Question 4

USE swiftlogix_supply_chain;

SELECT
    o.orderID,
    o.order_date,
    o.benefit_per_order,
    o.order_profit_per_order,
    o.customerID
FROM Orders AS o
WHERE o.benefit_per_order = (
    SELECT MAX(benefit_per_order)
    FROM Orders
)
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.

```sql
# Question 5

USE swiftlogix_supply_chain;

SELECT
    c.customerID,
    c.customer_fname,
    c.customer_lname,
    c.customer_country
FROM Customers AS c
WHERE c.customerID IN (
    SELECT o.customerID
    FROM Orders AS o
    GROUP BY o.customerID
    HAVING COUNT(*) > 5
)
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.

```sql
# Question 6

USE swiftlogix_supply_chain;

SELECT
    d.departmentID,
    d.department_name,
    c.categoryID,
    c.category_name,
    SUM(oi.sales)              AS total_sales,
    SUM(oi.order_item_total)   AS total_item_amount,
    AVG(oi.order_item_profit_ratio) AS avg_profit_ratio
FROM Departments AS d
JOIN Categories AS c  ON c.departmentID = d.departmentID
JOIN Products   AS p  ON p.categoryID   = c.categoryID
JOIN Order_items AS oi ON oi.productID  = p.productID
GROUP BY
    d.departmentID,
    d.department_name,
    c.categoryID,
    c.category_name
ORDER BY
    d.departmentID,
    c.categoryID;

```

23:19

Result Grid | Filter Rows: Search | Export:

department...	department_na...	categoryID	category_name	total_sales	total_item_amo...	avg_profit_ra...
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.

```sql
# Question 7

USE swiftlogix_supply_chain;

SELECT
    c.customerID,
    c.customer_fname,
    c.customer_lname,
    c.customer_country
FROM Customers AS c
WHERE c.customerID NOT IN (
    SELECT o.customerID
    FROM Orders AS o
    GROUP BY o.customerID
    HAVING COUNT(*) > 5
)
ORDER BY c.customerID;
```

100% 23:17

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
23	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.

```sql
# Question 8

USE swiftlogix_supply_chain;

SELECT
    o.orderID,
    o.order_date,
    o.shipping_date,
    o.late_delivery_risk,
    CASE
        WHEN o.late_delivery_risk = 1 THEN 'Late delivery'
        ELSE 'On time / on-schedule'
    END AS delivery_status_flag
FROM Orders AS o
ORDER BY o.orderID;
```

100% 20:15

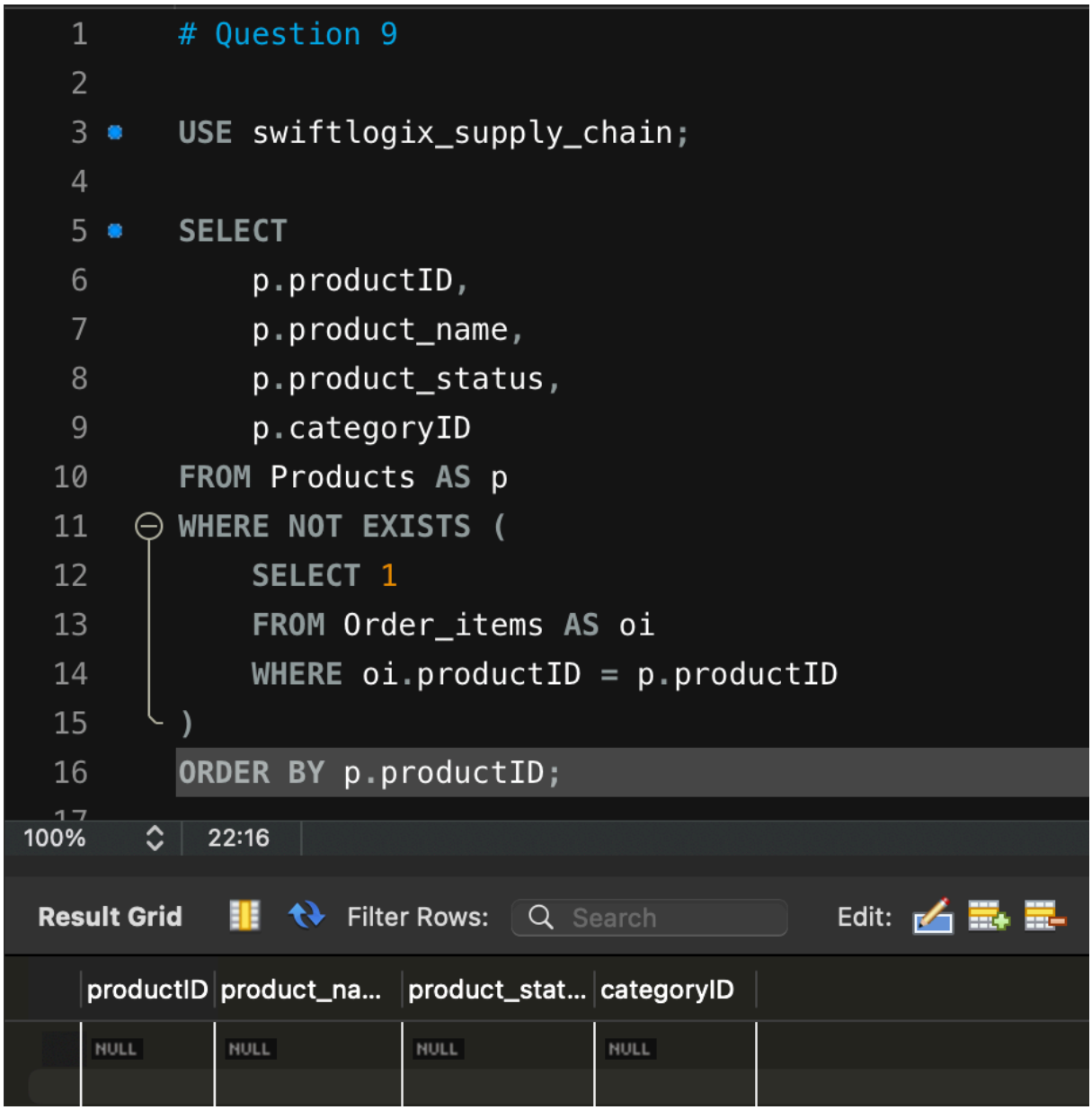
Result Grid | Filter Rows: Search | Export: | Fetch rows:

orderID	order_date	shipping_date	late_delivery_ri...	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.



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.

```sql
# Question 10

USE swiftlogix_supply_chain;

SELECT
    o.orderID,
    o.order_date,
    o.customerID,
    o.benefit_per_order,
    o.order_profit_per_order
FROM Orders AS o
WHERE o.orderID IN (
    SELECT DISTINCT oi.orderID
    FROM Order_items AS oi
    WHERE oi.order_item_discount IS NOT NULL
)
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	2015-01-01 00:00:00	11599	88.79	88.79
2	2015-01-01 00:21:00	256	91.18	91.18
4	2015-01-01 01:03:00	8827	33.59	33.59
5	2015-01-01 01:24:00	11318	9.38	9.38
7	2015-01-01 02:06:00	4530	120.95	120.95
8	2015-01-01 02:27:00	2911	73.11	73.11
9	2015-01-01 02:48:00	5657	3.80	3.80
10	2015-01-01 03:09:00	5648	72.00	72.00
11	2015-01-01 03:30:00	918	117.17	117.17
12	2015-01-01 03:51:00	1837	23.49	23.49
13	2015-01-01 04:12:00	9149	47.09	47.09
14	2015-01-01 04:33:00	9842	-10.62	-10.62
15	2015-01-01 04:54:00	2568	-254.19	-254.19
16	2015-01-01 05:15:00	7276	24.00	24.00
17	2015-01-01 05:36:00	2667	2.40	2.40

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.