Sherali Ozodov
ISTA 431: Data Warehousing and Analytics in the Cloud
Fall 2024
Week 16: Final Project Update
Nayem Rahman
November 26, 2024

For my final project, I am developing an E-commerce Sales System that will manage customer orders, product inventory, and payments. The system tracks customer information, the products they purchase, and the details of each transaction. The project includes creating a conceptual, logical, and physical data model for an e-commerce platform. I will design tables for customers, products, orders, order details, payments, and product categories. SQL scripts will be used to implement the database, insert data, and perform queries.

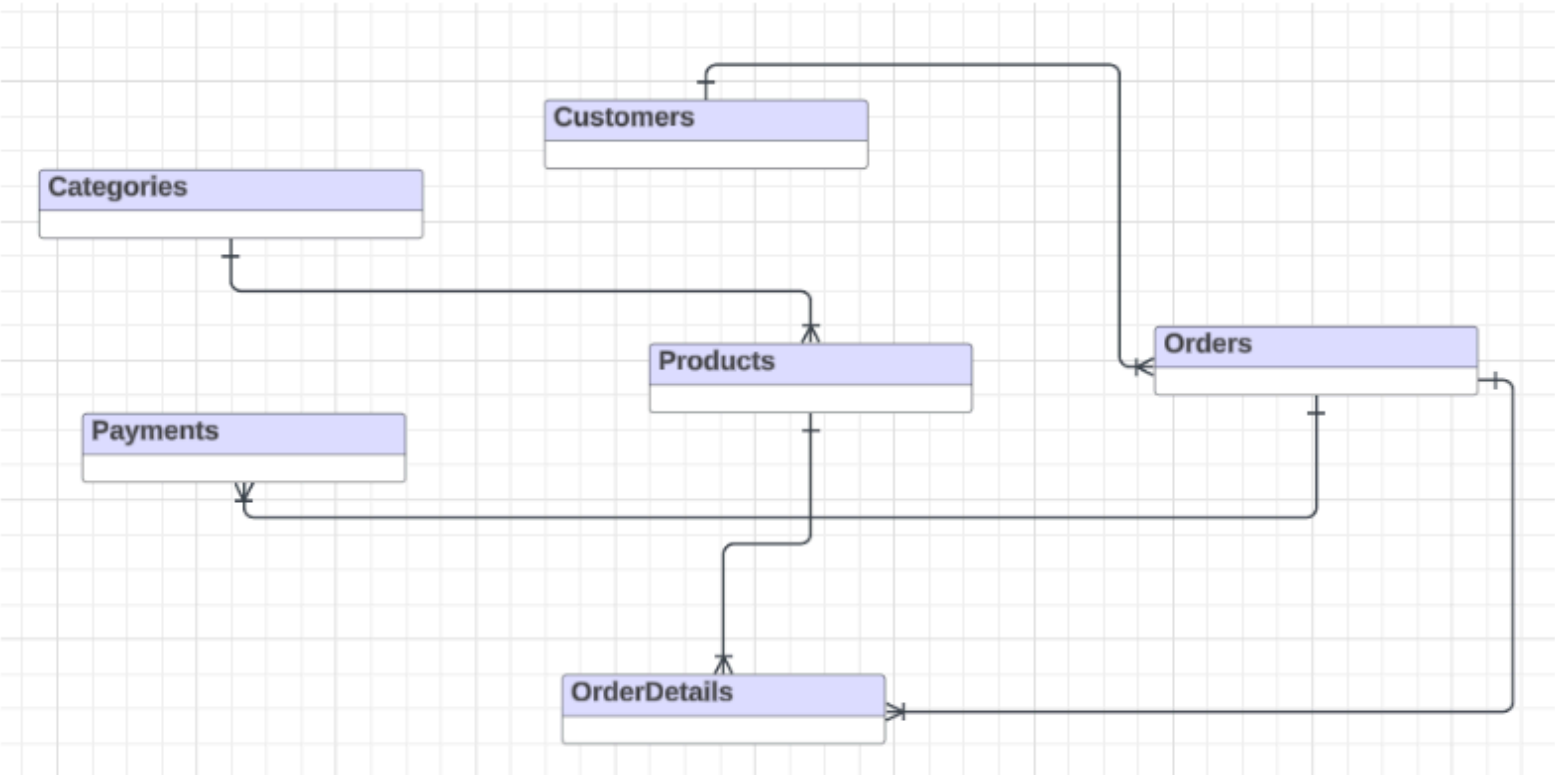
The data for this project will be created based on a realistic e-commerce business scenario. The dataset will include customer information, product listings, order records, and payment details. I will design the data based on typical operations within an e-commerce platform, such as customer orders, product inventory, and payments. The data is not sourced from an external source but will be manually created to simulate a real-world environment.

For this project, I will use:

- Database Management System (DBMS): MySQL will be used to create and manage the database.
- Database Visualization Tools: DBeaver will be used for visualizing and managing the database schema.

1. Develop the physical model based on the Logical Model. You must upload the data file in CSV or link to the data source.

Conceptual Model:



In this database design, the many-to-many relationship exists between Orders and Products.

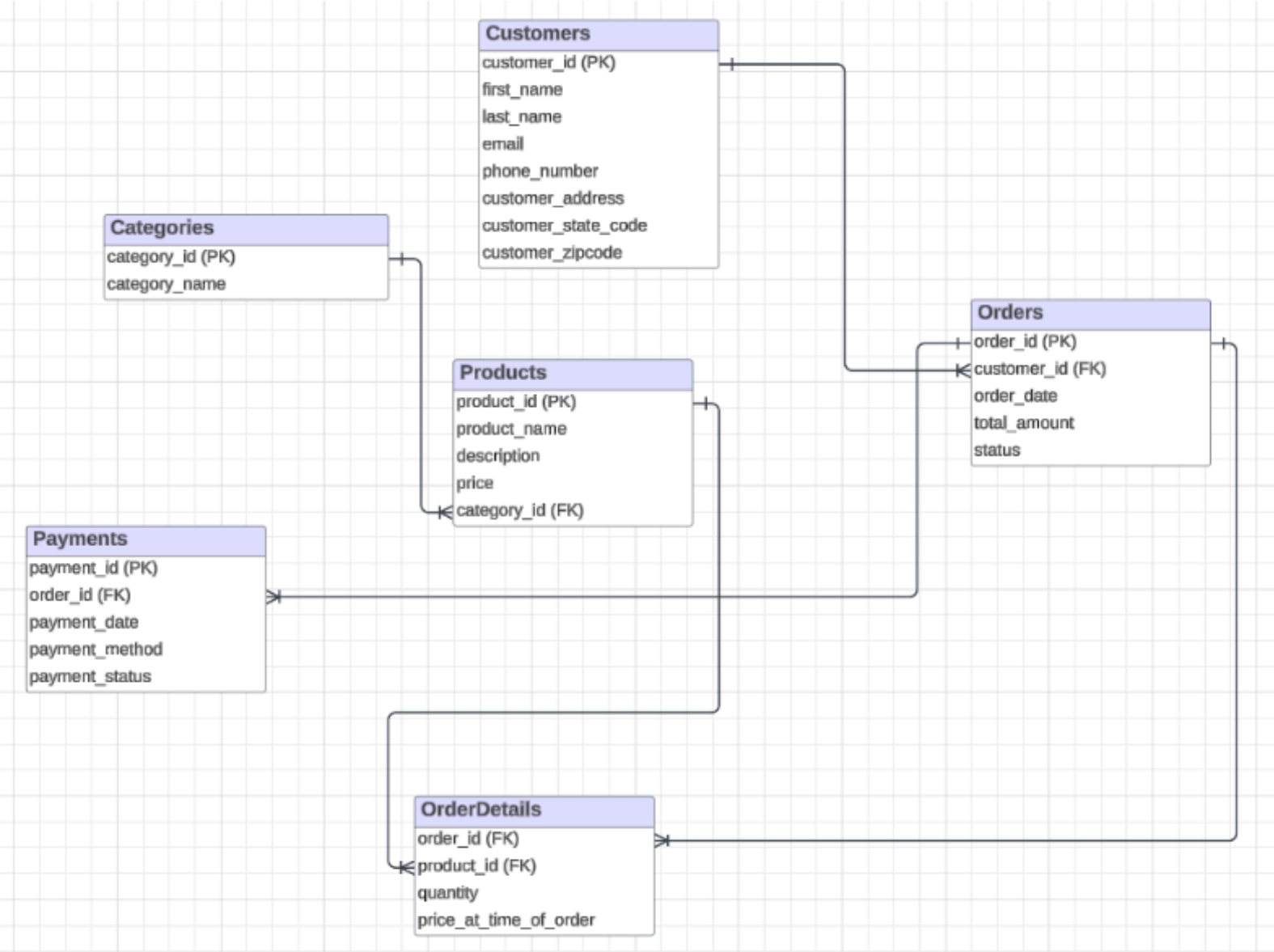
Why it's a Many-to-Many Relationship:

- Each Order can include multiple Products. For example, a customer might place an order that includes a laptop, a tablet, and a pair of headphones. This means an order could have multiple product entries.
- Conversely, each Product can be included in multiple Orders. For example, the same laptop model could be ordered by multiple customers in various orders.

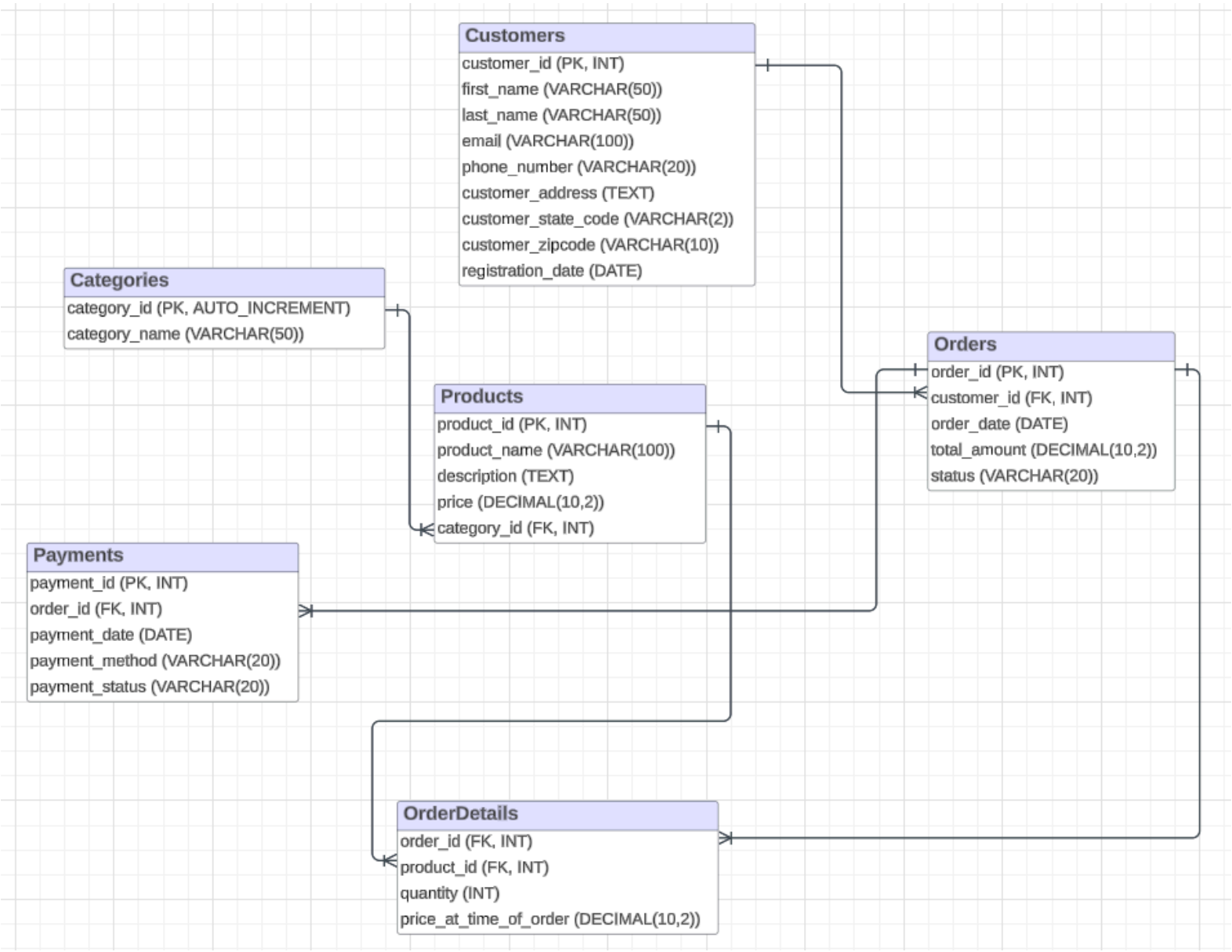
Corresponding Junction Table:

- To handle this many-to-many relationship, we create a junction table called OrderDetails. This table acts as a bridge, linking Orders and Products by recording each instance of a product within an order.

Logical Model:



Physical Model:



2. 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.

*<localhost> Script-2

```sql
CREATE TABLE Categories (
    category_id INT PRIMARY KEY AUTO_INCREMENT,
    category_name VARCHAR(50)
);
```

Statistics 1

Name	Value
Updated Rows	0
Query	CREATE TABLE Categories (
	category_id INT PRIMARY KEY AUTO_INCREMENT,
	category_name VARCHAR(50)
	)
Start time	Mon Nov 25 17:15:15 MST 2024
Finish time	Mon Nov 25 17:15:15 MST 2024

Created a table called Categories.

*<localhost> Script-2

```sql
-- 2. Customers table
CREATE TABLE Customers (
    customer_id INT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    email VARCHAR(100),
    phone_number VARCHAR(20),
    customer_address TEXT,
    customer_state_code VARCHAR(2),
    customer_zipcode VARCHAR(10),
    registration_date DATE
);
```

Statistics 1

Name	Value
Updated Rows	0
Query	-- 2. Customers table
	CREATE TABLE Customers (
	customer_id INT PRIMARY KEY,
	first_name VARCHAR(50),
	last_name VARCHAR(50),
	email VARCHAR(100),
	phone_number VARCHAR(20),
	customer_address TEXT,
	customer_state_code VARCHAR(2),
	customer_zipcode VARCHAR(10),
	registration_date DATE
	)
Start time	Mon Nov 25 17:18:04 MST 2024
Finish time	Mon Nov 25 17:18:04 MST 2024

Created a table called Customers.

```sql
-- 3. Products Table
CREATE TABLE Products (
    product_id INT PRIMARY KEY,
    product_name VARCHAR(100),
    description TEXT,
    price DECIMAL(10, 2),
    category_id INT,
    FOREIGN KEY (category_id) REFERENCES Categories(category_id)
);
```

Statistics 1

Name	Value
Updated Rows	0
Query	-- 3. Products Table
	CREATE TABLE Products (
	product_id INT PRIMARY KEY,
	product_name VARCHAR(100),
	description TEXT,
	price DECIMAL(10, 2),
	category_id INT,
	FOREIGN KEY (category_id) REFERENCES Categories(category_id)
	)
Start time	Mon Nov 25 17:20:07 MST 2024
Finish time	Mon Nov 25 17:20:07 MST 2024

Created a table called Products.

```sql
-- 4. Orders Table
CREATE TABLE Orders (
    order_id INT PRIMARY KEY,
    customer_id INT,
    order_date DATE,
    total_amount DECIMAL(10, 2),
    status VARCHAR(20),
    FOREIGN KEY (customer_id) REFERENCES Customers(customer_id)
);
```

Statistics 1

Name	Value
Updated Rows	0
Query	-- 4. Orders Table
	CREATE TABLE Orders (
	order_id INT PRIMARY KEY,
	customer_id INT,
	order_date DATE,
	total_amount DECIMAL(10, 2),
	status VARCHAR(20),
	FOREIGN KEY (customer_id) REFERENCES Customers(customer_id)
	)
Start time	Mon Nov 25 17:20:53 MST 2024
Finish time	Mon Nov 25 17:20:53 MST 2024

Created a table called Orders.

```sql
-- 5. Order Details Table
CREATE TABLE OrderDetails (
    order_id INT,
    product_id INT,
    quantity INT,
    price_at_time_of_order DECIMAL(10, 2),
    PRIMARY KEY (order_id, product_id),
    FOREIGN KEY (order_id) REFERENCES Orders(order_id),
    FOREIGN KEY (product_id) REFERENCES Products(product_id)
);
```

Statistics 1

Name	Value
Updated Rows	0
Query	-- 5. Order Details Table
	CREATE TABLE OrderDetails (
	order_id INT,
	product_id INT,
	quantity INT,
	price_at_time_of_order DECIMAL(10, 2),
	PRIMARY KEY (order_id, product_id),
	FOREIGN KEY (order_id) REFERENCES Orders(order_id),
	FOREIGN KEY (product_id) REFERENCES Products(product_id)
	)
Start time	Mon Nov 25 17:21:27 MST 2024
Finish time	Mon Nov 25 17:21:27 MST 2024

Created a table called OrderDetails.

```sql
-- 6. Payments Table
CREATE TABLE Payments (
    payment_id INT PRIMARY KEY,
    order_id INT,
    payment_date DATE,
    payment_method VARCHAR(20),
    payment_status VARCHAR(20),
    FOREIGN KEY (order_id) REFERENCES Orders(order_id)
);
```

Statistics 1

Name	Value
Updated Rows	0
Query	-- 6. Payments Table
	CREATE TABLE Payments (
	payment_id INT PRIMARY KEY,
	order_id INT,
	payment_date DATE,
	payment_method VARCHAR(20),
	payment_status VARCHAR(20),
	FOREIGN KEY (order_id) REFERENCES Orders(order_id)
	)
Start time	Mon Nov 25 17:22:11 MST 2024
Finish time	Mon Nov 25 17:22:11 MST 2024

Created a table called Payments.

Insert data into tables:

```sql
-- Insert into Categories Table
INSERT INTO Categories (category_name)
VALUES ('Electronics'), ('Appliances'), ('Apparel'), ('Books'), ('Toys');
```

Statistics 1

Name	Value
Updated Rows	5
Query	-- Insert into Categories Table
	INSERT INTO Categories (category_name)
	VALUES ('Electronics'), ('Appliances'), ('Apparel'), ('Books'), ('Toys')
Start time	Mon Nov 25 17:26:50 MST 2024
Finish time	Mon Nov 25 17:26:50 MST 2024

Inserted into Categories Table.

```sql
-- Insert into Customers
INSERT INTO Customers (customer_id, first_name, last_name, email, phone_number, customer_address, customer_state_code, customer_zipcode, registration_date)
VALUES
(3, 'Steven', 'Guzman', 'herringwilliam@hotmail.com', '319-150-8751', '53373 James Loop', 'GA', '08148', '2020-05-11'),
(4, 'Cory', 'Pena', 'bakermark@yahoo.com', '520-844-2197', '265 Espinoza Grove', 'CT', '89745', '2022-02-05'),
(5, 'Robert', 'Ferguson', 'dpatterson@gmail.com', '831-828-2224', '738 Logan Passage', 'AL', '23677', '2022-08-16'),
(13, 'Christopher', 'Taylor', 'zbrown@gmail.com', '847-689-6816', '39122 Charles Stravenue', 'ID', '23380', '2023-11-13'),
(18, 'Sean', 'Cook', 'alan69@yahoo.com', '554-977-9877', '537 Krueger Road', 'NE', '36881', '2021-05-21'),
(21, 'Sara', 'Holloway', 'michele85@yahoo.com', '169-181-1444', '410 Steven Forges', 'MO', '03231', '2022-08-16'),
(23, 'Donna', 'Davidson', 'patriciahamilton@gmail.com', '748-995-0405', '834 Tammy Spur', 'SC', '03367', '2023-05-03'),
(26, 'Shaun', 'Burns', 'pbarrett@gmail.com', '441-266-8305', '739 Amanda Parks', 'MD', '72928', '2024-07-16'),
(27, 'Tina', 'Butler', 'sgraham@hotmail.com', '896-802-4008', '553 Becker Stravenue', 'WA', '63047', '2020-01-15'),
(28, 'Jason', 'Cook', 'matthewsantiago@hotmail.com', '484-139-7774', '55841 Andrea Island', 'ME', '61731', '2020-08-18'),
(33, 'Michael', 'Thompson', 'masseykatrina@gmail.net', '840-234-8780', '57798 Timothy Motorway', 'SD', '03395', '2021-08-20'),
(35, 'Ryan', 'Solis', 'william08@gmail.com', '461-636-4098', '38468 Christine Underpass', 'OH', '33245', '2024-03-27');
```

Statistics 1

Name	Value
Updated Rows	12
Query	-- Insert into Customers
	INSERT INTO Customers (customer_id, first_name, last_name, email, phone_number, customer_address, customer_state_code, customer_zipcode, registration_date)
	VALUES
	(3, 'Steven', 'Guzman', 'herringwilliam@hotmail.com', '319-150-8751', '53373 James Loop', 'GA', '08148', '2020-05-11'),
	(4, 'Cory', 'Pena', 'bakermark@yahoo.com', '520-844-2197', '265 Espinoza Grove', 'CT', '89745', '2022-02-05'),
	(5, 'Robert', 'Ferguson', 'dpatterson@gmail.com', '831-828-2224', '738 Logan Passage', 'AL', '23677', '2022-08-16'),
	(13, 'Christopher', 'Taylor', 'zbrown@gmail.com', '847-689-6816', '39122 Charles Stravenue', 'ID', '23380', '2023-11-13'),
	(18, 'Sean', 'Cook', 'alan69@yahoo.com', '554-977-9877', '537 Krueger Road', 'NE', '36881', '2021-05-21'),
	(21, 'Sara', 'Holloway', 'michele85@yahoo.com', '169-181-1444', '410 Steven Forges', 'MO', '03231', '2022-08-16'),
	(23, 'Donna', 'Davidson', 'patriciahamilton@gmail.com', '748-995-0405', '834 Tammy Spur', 'SC', '03367', '2023-05-03'),
	(26, 'Shaun', 'Burns', 'pbarrett@gmail.com', '441-266-8305', '739 Amanda Parks', 'MD', '72928', '2024-07-16'),
	(27, 'Tina', 'Butler', 'sgraham@hotmail.com', '896-802-4008', '553 Becker Stravenue', 'WA', '63047', '2020-01-15'),
	(28, 'Jason', 'Cook', 'matthewsantiago@hotmail.com', '484-139-7774', '55841 Andrea Island', 'ME', '61731', '2020-08-18'),
	(33, 'Michael', 'Thompson', 'masseykatrina@gmail.net', '840-234-8780', '57798 Timothy Motorway', 'SD', '03395', '2021-08-20'),
	(35, 'Ryan', 'Solis', 'william08@gmail.com', '461-636-4098', '38468 Christine Underpass', 'OH', '33245', '2024-03-27')
Start time	Mon Nov 25 17:27:46 MST 2024
Finish time	Mon Nov 25 17:27:46 MST 2024

MST | en_US | Writable | Smart Insert | 108 : 1 : 3258

Inserted into Customers Table.

```sql
-- Insert into Products
INSERT INTO Products (product_id, product_name, description, price, category_id)
VALUES
(108, 'Blender', 'High-speed blender', 100, 2),
(105, 'Smartwatch', 'Fitness tracking watch', 250, 1),
(103, 'Coffee Maker', '12-cup coffee maker', 80, 2),
(110, 'Tablet', '10-inch tablet', 300, 1),
(101, 'Laptop', 'High-performance laptop', 1200, 1),
(102, 'Headphones', 'Noise-cancelling headphones', 150, 1),
(107, 'Toy Car', 'Remote-controlled toy car', 40, 5),
(106, 'Novel', 'Bestselling fiction novel', 20, 4),
(109, 'T-shirt', 'Cotton t-shirt', 25, 3),
(104, 'Running Shoes', 'Lightweight running shoes', 60, 3);
```

Statistics 1

Name	Value
Updated Rows	10
Query	-- Insert into Products
	INSERT INTO Products (product_id, product_name, description, price, category_id)
	VALUES
	(108, 'Blender', 'High-speed blender', 100, 2),
	(105, 'Smartwatch', 'Fitness tracking watch', 250, 1),
	(103, 'Coffee Maker', '12-cup coffee maker', 80, 2),
	(110, 'Tablet', '10-inch tablet', 300, 1),
	(101, 'Laptop', 'High-performance laptop', 1200, 1),
	(102, 'Headphones', 'Noise-cancelling headphones', 150, 1),
	(107, 'Toy Car', 'Remote-controlled toy car', 40, 5),
	(106, 'Novel', 'Bestselling fiction novel', 20, 4),
	(109, 'T-shirt', 'Cotton t-shirt', 25, 3),
	(104, 'Running Shoes', 'Lightweight running shoes', 60, 3)
Start time	Mon Nov 25 17:28:39 MST 2024
Finish time	Mon Nov 25 17:28:39 MST 2024

Inserted into Products Table.

```sql
-- Insert into Orders
INSERT INTO Orders (order_id, customer_id, order_date, total_amount, status)
VALUES
(1098, 3, '2024-01-13', 976.9, 'cancelled'),
(1073, 4, '2024-06-27', 1485.58, 'cancelled'),
(1049, 4, '2024-09-24', 353.44, 'cancelled'),
(1018, 4, '2024-03-02', 464.89, 'cancelled'),
(1012, 5, '2024-08-17', 1306.7, 'pending'),
(1057, 5, '2024-04-08', 1362.64, 'delivered'),
(1023, 5, '2024-07-19', 113.1, 'pending'),
(1100, 5, '2024-07-08', 471.53, 'cancelled'),
(1030, 5, '2024-04-07', 428.43, 'cancelled'),
(1050, 5, '2024-01-23', 246.62, 'delivered'),
(1053, 5, '2024-07-08', 690.26, 'cancelled'),
(1036, 13, '2024-08-16', 673.75, 'delivered'),
(1026, 18, '2024-05-22', 173.74, 'delivered'),
(1003, 18, '2024-07-05', 357.22, 'delivered'),
(1086, 21, '2024-09-29', 63.37, 'cancelled'),
(1060, 23, '2024-07-14', 1135.98, 'shipped'),
(1019, 26, '2024-05-15', 320.13, 'delivered'),
(1033, 28, '2024-04-22', 123.79, 'shipped'),
(1001, 28, '2024-03-15', 517.53, 'shipped'),
(1037, 28, '2024-05-31', 128.7, 'shipped'),
(1041, 28, '2024-02-16', 444.85, 'delivered'),
(1011, 33, '2024-03-22', 93.95, 'delivered'),
(1031, 35, '2024-03-03', 609.31, 'pending'),
(1061, 35, '2024-07-13', 213.29, 'cancelled'),
(1038, 35, '2024-07-07', 1464.76, 'delivered'),
(1078, 35, '2024-05-04', 1467.32, 'delivered');
```

Statistics 1

Name	Value
Updated Rows	26
Query	-- Insert into Orders
	INSERT INTO Orders (order_id, customer_id, order_date, total_amount, status)
	VALUES
	(1098, 3, '2024-01-13', 976.9, 'cancelled'),
	(1073, 4, '2024-06-27', 1485.58, 'cancelled'),
	(1049, 4, '2024-09-24', 353.44, 'cancelled'),
	(1018, 4, '2024-03-02', 464.89, 'cancelled'),
	(1012, 5, '2024-08-17', 1306.7, 'pending'),
	(1057, 5, '2024-04-08', 1362.64, 'delivered'),
	(1023, 5, '2024-07-19', 113.1, 'pending'),

MST | en_US | Writable

Inserted into Orders Table.

```sql
-- Insert into OrderDetails
INSERT INTO OrderDetails (order_id, product_id, quantity, price_at_time_of_order)
VALUES
(1098, 108, 5, 500),
(1073, 105, 1, 250),
(1049, 103, 4, 320),
(1018, 110, 2, 600),
(1012, 110, 5, 1500),
(1057, 103, 1, 80),
(1023, 108, 5, 500),
(1100, 110, 2, 600),
(1030, 101, 3, 3600),
(1050, 105, 5, 1250),
(1053, 110, 2, 600),
(1036, 103, 4, 320),
(1026, 110, 1, 300),
(1003, 102, 3, 450),
(1086, 107, 5, 200),
(1060, 106, 4, 80),
(1019, 102, 3, 450),
(1086, 108, 3, 300),
(1053, 102, 4, 600),
(1033, 102, 2, 300),
(1001, 102, 4, 600),
(1037, 110, 2, 600),
(1041, 105, 1, 250),
(1011, 102, 2, 300),
(1031, 109, 2, 50),
(1061, 106, 4, 80),
(1038, 107, 4, 160),
(1078, 110, 5, 1500),
(1033, 110, 4, 1200),
(1050, 110, 2, 600);
```

Statistics 1

Name	Value
Updated Rows	30
Query	-- Insert into OrderDetails
	INSERT INTO OrderDetails (order_id, product_id, quantity, price_at_time_of_order)
	VALUES
	(1098, 108, 5, 500),
	(1073, 105, 1, 250),
	(1049, 103, 4, 320),
	(1018, 110, 2, 600),
	(1012, 110, 5, 1500),

Inserted into OrderDetails Table.

```sql
-- Insert into Payments
INSERT INTO Payments (payment_id, order_id, payment_date, payment_method, payment_status)
VALUES
(5089, 1098, '2024-04-08', 'Credit Card', 'successful'),
(5064, 1073, '2024-08-02', 'PayPal', 'successful'),
(5072, 1049, '2024-09-28', 'Debit Card', 'failed'),
(5046, 1018, '2024-07-19', 'PayPal', 'pending'),
(5056, 1012, '2024-09-17', 'PayPal', 'failed'),
(5017, 1023, '2024-07-25', 'Credit Card', 'successful'),
(5030, 1100, '2024-08-12', 'Credit Card', 'successful'),
(5069, 1030, '2024-05-20', 'Debit Card', 'successful'),
(5025, 1050, '2024-10-17', 'Credit Card', 'successful'),
(5059, 1053, '2024-08-16', 'Debit Card', 'successful'),
(5011, 1036, '2024-09-04', 'Debit Card', 'successful'),
(5093, 1026, '2024-06-11', 'Debit Card', 'pending'),
(5022, 1003, '2024-08-13', 'Debit Card', 'successful'),
(5020, 1086, '2024-10-15', 'Credit Card', 'failed'),
(5023, 1060, '2024-07-20', 'PayPal', 'successful'),
(5052, 1019, '2024-06-15', 'Credit Card', 'pending'),
(5065, 1086, '2024-10-09', 'Credit Card', 'pending'),
(5088, 1053, '2024-09-19', 'PayPal', 'pending'),
(5042, 1033, '2024-10-14', 'Debit Card', 'successful'),
(5006, 1001, '2024-07-19', 'Debit Card', 'pending'),
(5054, 1037, '2024-06-09', 'Credit Card', 'successful'),
(5048, 1011, '2024-09-18', 'Credit Card', 'failed'),
(5028, 1031, '2024-06-26', 'Debit Card', 'pending'),
(5029, 1061, '2024-07-30', 'Credit Card', 'pending'),
(5040, 1038, '2024-07-26', 'PayPal', 'failed'),
(5009, 1078, '2024-05-19', 'PayPal', 'successful'),
(5034, 1050, '2024-08-17', 'Debit Card', 'failed'),
(5060, 1053, '2024-09-29', 'Debit Card', 'pending');
```

Statistics 1

Name	Value
Updated Rows	28
Query	-- Insert into Payments
	INSERT INTO Payments (payment_id, order_id, payment_date, payment_method, payment_status)
	VALUES
	(5089, 1098, '2024-04-08', 'Credit Card', 'successful'),
	(5064, 1073, '2024-08-02', 'PayPal', 'successful'),
	(5072, 1049, '2024-09-28', 'Debit Card', 'failed'),
	(5046, 1018, '2024-07-19', 'PayPal', 'pending'),
	(5056, 1012, '2024-09-17', 'PayPal', 'failed'),

Inserted into Payments Table.

3. 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).

```
SELECT * FROM Categories
ORDER BY category_id
```

categories 1

SELECT * FROM Categories ORDER BY category_id

Enter a SQL expression to filter results (use Ctrl+Space)

	category_id	category_name
1	1	Electronics
2	2	Appliances
3	3	Apparel
4	4	Books
5	5	Toys

Retrieved the data from the table by using the SELECT * statement and order by PK column.

```
SELECT * FROM Customers
ORDER BY customer_id
```

customers 1

SELECT * FROM Customers ORDER BY customer_id | Enter a SQL expression to filter results (use Ctrl+Space)

	customer_id	first_name	last_name	email	phone_number	customer_address	customer_state_code	customer_zipcode	registration_date
1	3	Steven	Guzman	herringwilliam@hotmail.com	319-150-8751	53373 James Loop	GA	08148	2020-05-11
2	4	Cory	Pena	bakermark@yahoo.com	520-844-2197	265 Espinoza Grove	CT	89745	2022-02-05
3	5	Robert	Ferguson	dpatterson@gmail.com	831-828-2224	738 Logan Passage	AL	23677	2022-08-16
4	13	Christopher	Taylor	zbrown@gmail.com	847-689-6816	39122 Charles Stravenue	ID	23380	2023-11-13
5	18	Sean	Cook	alan69@yahoo.com	554-977-9877	537 Krueger Road	NE	36881	2021-05-21
6	21	Sara	Holloway	michele85@yahoo.com	169-181-1444	410 Steven Forges	MO	03231	2022-08-16
7	23	Donna	Davidson	patriciahamilton@gmail.com	748-995-0405	834 Tammy Spur	SC	03367	2023-05-03
8	26	Shaun	Burns	pbarrett@gmail.com	441-266-8305	739 Amanda Parks	MD	72928	2024-07-16
9	27	Tina	Butler	sgraham@hotmail.com	896-802-4008	553 Becker Stravenue	WA	63047	2020-01-15
10	28	Jason	Cook	matthewsantiago@hotmail.com	484-139-7774	55841 Andrea Island	ME	61731	2020-08-18
11	33	Michael	Thompson	masseykatrina@gmail.net	840-234-8780	57798 Timothy Motorway	SD	03395	2021-08-20
12	35	Ryan	Solis	william08@gmail.com	461-636-4098	38468 Christine Underpass	OH	33245	2024-03-27

Retrieved the data from the table by using the SELECT * statement and order by PK column.

```
SELECT * FROM Products
ORDER BY product_id
```

products 1

SELECT * FROM Products ORDER BY product_id | Enter a SQL expression to filter results (use Ctrl+Space)

	product_id	product_name	description	price	category_id
1	101	Laptop	High-performance laptop	1,200	1
2	102	Headphones	Noise-cancelling headphones	150	1
3	103	Coffee Maker	12-cup coffee maker	80	2
4	104	Running Shoes	Lightweight running shoes	60	3
5	105	Smartwatch	Fitness tracking watch	250	1
6	106	Novel	Bestselling fiction novel	20	4
7	107	Toy Car	Remote-controlled toy car	40	5
8	108	Blender	High-speed blender	100	2
9	109	T-shirt	Cotton t-shirt	25	3
10	110	Tablet	10-inch tablet	300	1

Retrieved the data from the table by using the SELECT * statement and order by PK column.

```
SELECT * FROM Orders
```

orders 1

SELECT * FROM Orders | Enter a SQL expression to filter results (use Ctrl+Space)

	order_id	customer_id	order_date	total_amount	status
1	1,001	28	2024-03-15	517.53	shipped
2	1,003	18	2024-07-05	357.22	delivered
3	1,011	33	2024-03-22	93.95	delivered
4	1,012	5	2024-08-17	1,306.7	pending
5	1,018	4	2024-03-02	464.89	cancelled
6	1,019	26	2024-05-15	320.13	delivered
7	1,023	5	2024-07-19	113.1	pending
8	1,026	18	2024-05-22	173.74	delivered
9	1,030	5	2024-04-07	428.43	cancelled
10	1,031	35	2024-03-03	609.31	pending
11	1,033	28	2024-04-22	123.79	shipped
12	1,036	13	2024-08-16	673.75	delivered
13	1,037	28	2024-05-31	128.7	shipped
14	1,038	35	2024-07-07	1,464.76	delivered
15	1,041	28	2024-02-16	444.85	delivered
16	1,049	4	2024-09-24	353.44	cancelled
17	1,050	5	2024-01-23	246.62	delivered
18	1,053	5	2024-07-08	690.26	cancelled
19	1,057	5	2024-04-08	1,362.64	delivered
20	1,060	23	2024-07-14	1,135.98	shipped
21	1,061	35	2024-07-13	213.29	cancelled
22	1,073	4	2024-06-27	1,485.58	cancelled
23	1,078	35	2024-05-04	1,467.32	delivered
24	1,086	21	2024-09-29	63.37	cancelled
25	1,098	3	2024-01-13	976.9	cancelled
26	1,100	5	2024-07-08	471.53	cancelled

Retrieved the data from the table by using the SELECT * statement and order by PK column.

```sql
SELECT * FROM OrderDetails
ORDER BY order_id, product_id
```

orderdetails 1

SELECT * FROM OrderDetails ORDER BY order_id, product_id | Enter a SQL expression to filter results (use Ctrl+Spa

	order_id	product_id	quantity	price_at_time_of_order
1	1,001	102	4	600
2	1,003	102	3	450
3	1,011	102	2	300
4	1,012	110	5	1,500
5	1,018	110	2	600
6	1,019	102	3	450
7	1,023	108	5	500
8	1,026	110	1	300
9	1,030	101	3	3,600
10	1,031	109	2	50
11	1,033	102	2	300
12	1,033	110	4	1,200
13	1,036	103	4	320
14	1,037	110	2	600
15	1,038	107	4	160
16	1,041	105	1	250
17	1,049	103	4	320
18	1,050	105	5	1,250
19	1,050	110	2	600
20	1,053	102	4	600
21	1,053	110	2	600
22	1,057	103	1	80
23	1,060	106	4	80
24	1,061	106	4	80
25	1,073	105	1	250
26	1,078	110	5	1,500
27	1,086	107	5	200
28	1,086	108	3	300
29	1,098	108	5	500
30	1,100	110	2	600

Refresh | Save | Cancel | Export data | 200 | 30

Retrieved the data from the table by using the SELECT * statement and order by PK column.

```sql
SELECT * FROM Payments
ORDER BY payment_id
```

payments 1

SELECT * FROM Payments ORDER BY payment_id | Enter a SQL expression to filter results (use Ctrl+Space)

	payment_id	order_id	payment_date	payment_method	payment_status
1	5,006	1,001	2024-07-19	Debit Card	pending
2	5,009	1,078	2024-05-19	PayPal	successful
3	5,011	1,036	2024-09-04	Debit Card	successful
4	5,017	1,023	2024-07-25	Credit Card	successful
5	5,020	1,086	2024-10-15	Credit Card	failed
6	5,022	1,003	2024-08-13	Debit Card	successful
7	5,023	1,060	2024-07-20	PayPal	successful
8	5,025	1,050	2024-10-17	Credit Card	successful
9	5,028	1,031	2024-06-26	Debit Card	pending
10	5,029	1,061	2024-07-30	Credit Card	pending
11	5,030	1,100	2024-08-12	Credit Card	successful
12	5,034	1,050	2024-08-17	Debit Card	failed
13	5,040	1,038	2024-07-26	PayPal	failed
14	5,042	1,033	2024-10-14	Debit Card	successful
15	5,046	1,018	2024-07-19	PayPal	pending
16	5,048	1,011	2024-09-18	Credit Card	failed
17	5,052	1,019	2024-06-15	Credit Card	pending
18	5,054	1,037	2024-06-09	Credit Card	successful
19	5,056	1,012	2024-09-17	PayPal	failed
20	5,059	1,053	2024-08-16	Debit Card	successful
21	5,060	1,053	2024-09-29	Debit Card	pending
22	5,064	1,073	2024-08-02	PayPal	successful
23	5,065	1,086	2024-10-09	Credit Card	pending
24	5,069	1,030	2024-05-20	Debit Card	successful
25	5,072	1,049	2024-09-28	Debit Card	failed
26	5,088	1,053	2024-09-19	PayPal	pending
27	5,089	1,098	2024-04-08	Credit Card	successful
28	5,093	1,026	2024-06-11	Debit Card	pending

Retrieved the data from the table by using the SELECT * statement and order by PK column.

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.

*<localhost> eCommerce

```
SELECT * FROM eCommerce.OrderDetails od
INNER JOIN eCommerce.Orders o ON od.order_id = o.order_id
INNER JOIN eCommerce.Products p ON p.product_id  = od.product_id
```

orderdetails(+) 1

SELECT * FROM eCommerce.OrderDetails od INNER JC | Enter a SQL expression to filter results (use Ctrl+Space)

	order_id	product_id	quantity	price_at_time_of_order	order_id	customer_id	order_date	total_amount	status	product_id	product_name	des
1	1,001	102	4	600	1,001	28	2024-03-15	517.53	shipped	102	Headphones	Noise-
2	1,003	102	3	450	1,003	18	2024-07-05	357.22	delivered	102	Headphones	Noise-
3	1,011	102	2	300	1,011	33	2024-03-22	93.95	delivered	102	Headphones	Noise-
4	1,012	110	5	1,500	1,012	5	2024-08-17	1,306.7	pending	110	Tablet	10-incl
5	1,018	110	2	600	1,018	4	2024-03-02	464.89	cancelled	110	Tablet	10-incl
6	1,019	102	3	450	1,019	26	2024-05-15	320.13	delivered	102	Headphones	Noise-
7	1,023	108	5	500	1,023	5	2024-07-19	113.1	pending	108	Blender	High-s
8	1,026	110	1	300	1,026	18	2024-05-22	173.74	delivered	110	Tablet	10-incl
9	1,030	101	3	3,600	1,030	5	2024-04-07	428.43	cancelled	101	Laptop	High-p
10	1,031	109	2	50	1,031	35	2024-03-03	609.31	pending	109	T-shirt	Cotton
11	1,033	102	2	300	1,033	28	2024-04-22	123.79	shipped	102	Headphones	Noise-
12	1,033	110	4	1,200	1,033	28	2024-04-22	123.79	shipped	110	Tablet	10-incl
13	1,036	103	4	320	1,036	13	2024-08-16	673.75	delivered	103	Coffee Maker	12-cup
14	1,037	110	2	600	1,037	28	2024-05-31	128.7	shipped	110	Tablet	10-incl
15	1,038	107	4	160	1,038	35	2024-07-07	1,464.76	delivered	107	Toy Car	Remot
16	1,041	105	1	250	1,041	28	2024-02-16	444.85	delivered	105	Smartwatch	Fitness
17	1,049	103	4	320	1,049	4	2024-09-24	353.44	cancelled	103	Coffee Maker	12-cup
18	1,050	105	5	1,250	1,050	5	2024-01-23	246.62	delivered	105	Smartwatch	Fitness
19	1,050	110	2	600	1,050	5	2024-01-23	246.62	delivered	110	Tablet	10-incl
20	1,053	102	4	600	1,053	5	2024-07-08	690.26	cancelled	102	Headphones	Noise-
21	1,053	110	2	600	1,053	5	2024-07-08	690.26	cancelled	110	Tablet	10-incl
22	1,057	103	1	80	1,057	5	2024-04-08	1,362.64	delivered	103	Coffee Maker	12-cup
23	1,060	106	4	80	1,060	23	2024-07-14	1,135.98	shipped	106	Novel	Bestse
24	1,061	106	4	80	1,061	35	2024-07-13	213.29	cancelled	106	Novel	Bestse
25	1,073	105	1	250	1,073	4	2024-06-27	1,485.58	cancelled	105	Smartwatch	Fitness
26	1,078	110	5	1,500	1,078	35	2024-05-04	1,467.32	delivered	110	Tablet	10-incl
27	1,086	107	5	200	1,086	21	2024-09-29	63.37	cancelled	107	Toy Car	Remot
28	1,086	108	3	300	1,086	21	2024-09-29	63.37	cancelled	108	Blender	High-s
29	1,098	108	5	500	1,098	3	2024-01-13	976.9	cancelled	108	Blender	High-s
30	1,100	110	2	600	1,100	5	2024-07-08	471.53	cancelled	110	Tablet	10-incl

SELECT * FROM eCommerce.OrderDetails od
INNER JOIN eCommerce.Orders o ON od.order_id = o.order_id
INNER JOIN eCommerce.Products p ON p.product_id = od.product_id

Wrote a SQL query using the junction table and two other related tables.

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
SELECT
    Customers.customer_id,
    Customers.first_name,
    Customers.last_name,
    Orders.order_id,
    Orders.order_date,
    Orders.total_amount
FROM Customers
LEFT OUTER JOIN Orders ON Customers.customer_id = Orders.customer_id
ORDER BY Customers.customer_id;
```

customers(+) 1

SELECT Customers.customer_id, Customers.first_name, Custc | Enter a SQL expression to filter results (use Ctrl+Space)

	customer_id	first_name	last_name	order_id	order_date	total_amount
1	3	Steven	Guzman	1,098	2024-01-13	976.9
2	4	Cory	Pena	1,018	2024-03-02	464.89
3	4	Cory	Pena	1,049	2024-09-24	353.44
4	4	Cory	Pena	1,073	2024-06-27	1,485.58
5	5	Robert	Ferguson	1,012	2024-08-17	1,306.7
6	5	Robert	Ferguson	1,023	2024-07-19	113.1
7	5	Robert	Ferguson	1,030	2024-04-07	428.43
8	5	Robert	Ferguson	1,050	2024-01-23	246.62
9	5	Robert	Ferguson	1,053	2024-07-08	690.26
10	5	Robert	Ferguson	1,057	2024-04-08	1,362.64
11	5	Robert	Ferguson	1,100	2024-07-08	471.53
12	13	Christopher	Taylor	1,036	2024-08-16	673.75
13	18	Sean	Cook	1,003	2024-07-05	357.22
14	18	Sean	Cook	1,026	2024-05-22	173.74
15	21	Sara	Holloway	1,086	2024-09-29	63.37
16	23	Donna	Davidson	1,060	2024-07-14	1,135.98
17	26	Shaun	Burns	1,019	2024-05-15	320.13
18	27	Tina	Butler	[NULL]	[NULL]	[NULL]
19	28	Jason	Cook	1,001	2024-03-15	517.53
20	28	Jason	Cook	1,033	2024-04-22	123.79
21	28	Jason	Cook	1,037	2024-05-31	128.7
22	28	Jason	Cook	1,041	2024-02-16	444.85
23	33	Michael	Thompson	1,011	2024-03-22	93.95
24	35	Ryan	Solis	1,031	2024-03-03	609.31
25	35	Ryan	Solis	1,038	2024-07-07	1,464.76
26	35	Ryan	Solis	1,061	2024-07-13	213.29

The results of a LEFT OUTER JOIN include all rows from the left table (Customers), even if there is no matching record in the right table (Orders). Unmatched rows from the right table show NULL values. In contrast, an INNER JOIN only includes rows where there is a match in both tables, excluding any customers without orders. This makes the LEFT OUTER JOIN useful for identifying customers who have not placed any orders, while the INNER JOIN focuses only on customers with orders.

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

```
SELECT c.customer_id, c.first_name, c.last_name, o.total_amount
FROM eCommerce.Customers c
INNER JOIN eCommerce.Orders o ON c.customer_id = o.customer_id
WHERE o.total_amount = (
    SELECT MAX(total_amount)
    FROM eCommerce.Orders
)
ORDER BY c.customer_id;
```

customers(+) 1

SELECT c.customer_id, c.first_name, c.last_name, o.total_amc

Enter a SQL expression to filter results (use Ctrl+Space)

	customer_id	first_name	last_name	total_amount
1	4	Cory	Pena	1,485.58

I wanted to find the details of the customer who placed the most expensive order. The query found the customer Cory Pena (customer_id: 4) who placed the most expensive order with a total amount of 1,485,58.

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

```sql
SELECT o.order_id, o.customer_id, o.order_date, o.total_amount
FROM eCommerce.Orders o
WHERE o.customer_id IN (
    SELECT customer_id
    FROM eCommerce.Customers
    WHERE YEAR(registration_date) = 2022
)
ORDER BY o.order_id;
```

orders 1

SELECT o.order_id, o.customer_id, o.order_date, o.total_amou

Enter a SQL expression to filter results (use Ctrl+Space)

	order_id	customer_id	order_date	total_amount
1	1,012	5	2024-08-17	1,306.7
2	1,018	4	2024-03-02	464.89
3	1,023	5	2024-07-19	113.1
4	1,030	5	2024-04-07	428.43
5	1,049	4	2024-09-24	353.44
6	1,050	5	2024-01-23	246.62
7	1,053	5	2024-07-08	690.26
8	1,057	5	2024-04-08	1,362.64
9	1,073	4	2024-06-27	1,485.58
10	1,086	21	2024-09-29	63.37
11	1,100	5	2024-07-08	471.53

The query identified all orders placed by customers who registered in 2022.

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

```sql
SELECT
    customer_id,
    SUM(total_amount) AS total_spent,
    COUNT(order_id) AS total_orders
FROM Orders
GROUP BY customer_id;
```

orders 1

SELECT customer_id, SUM(total_amount) AS total_spent, COU

Enter a SQL expression to filter results (use Ctrl+Space)

	customer_id	total_spent	total_orders
1	3	976.9	1
2	4	2,303.91	3
3	5	4,619.28	7
4	13	673.75	1
5	18	530.96	2
6	21	63.37	1
7	23	1,135.98	1
8	26	320.13	1
9	28	1,214.87	4
10	33	93.95	1
11	35	3,754.68	4

The query found the total amount spent (total_spent) and the total number of orders (total_orders) for each customer, grouped by their customer_id.

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

```sql
SELECT p.product_id, p.product_name
FROM Products p
WHERE p.product_id NOT IN (
    SELECT DISTINCT od.product_id
    FROM OrderDetails od
)
ORDER BY p.product_id;
```

products 1

SELECT p.product_id, p.product_name FROM Products p WHE | Enter a SQL expression to filter results (use Ct

	product_id	product_name
1	104	Running Shoes

The query listed all products that have not been sold, along with their product_id and product_name.

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

```sql
SELECT
    o.order_id,
    o.total_amount,
    CASE
        WHEN o.total_amount > 1000 THEN 'High Value'
        WHEN o.total_amount BETWEEN 500 AND 1000 THEN 'Medium Value'
        ELSE 'Low Value'
    END AS order_value_category
FROM Orders o
ORDER BY o.order_id;
```

orders 1

SELECT o.order_id, o.total_amount, CASE WHEN o.total_amou | Enter a SQL expression to filter results (use Ctrl+Space)

	order_id	total_amount	order_value_category
1	1,001	517.53	Medium Value
2	1,003	357.22	Low Value
3	1,011	93.95	Low Value
4	1,012	1,306.7	High Value
5	1,018	464.89	Low Value
6	1,019	320.13	Low Value
7	1,023	113.1	Low Value
8	1,026	173.74	Low Value
9	1,030	428.43	Low Value
10	1,031	609.31	Medium Value
11	1,033	123.79	Low Value
12	1,036	673.75	Medium Value
13	1,037	128.7	Low Value
14	1,038	1,464.76	High Value
15	1,041	444.85	Low Value
16	1,049	353.44	Low Value
17	1,050	246.62	Low Value
18	1,053	690.26	Medium Value
19	1,057	1,362.64	High Value
20	1,060	1,135.98	High Value
21	1,061	213.29	Low Value
22	1,073	1,485.58	High Value
23	1,078	1,467.32	High Value
24	1,086	63.37	Low Value
25	1,098	976.9	Medium Value
26	1,100	471.53	Low Value

The output categorized each order as high, medium, or low value based on the total amount.

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

```sql
SELECT c.customer_id, c.first_name, c.last_name
FROM Customers c
WHERE NOT EXISTS (
    SELECT 1
    FROM Orders o
    WHERE o.customer_id = c.customer_id
)
ORDER BY c.customer_id;
```

ustomers 1

ELECT c.customer_id, c.first_name, c.last_name FROM Custo | Enter a SQL expression to filter results (use C

customer_id	first_name	last_name
27	Tina	Butler

The query returned the customer_id, first_name, and last_name of all customers who do not have any matching entries in the Orders table.

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
SELECT p.product_id, p.product_name, p.price
FROM Products p
WHERE p.product_id IN (
    SELECT od.product_id
    FROM OrderDetails od
    WHERE od.quantity IS NOT NULL
)
ORDER BY p.product_id;
```

ducts 1

LECT p.product_id, p.product_name, p.price FROM Product | Enter a SQL expression to filter results (use

product_id	product_name	price
101	Laptop	1,200
102	Headphones	150
103	Coffee Maker	80
105	Smartwatch	250
106	Novel	20
107	Toy Car	40
108	Blender	100
109	T-shirt	25
110	Tablet	300

The query listed products that have been part of orders with a valid (NOT NULL) quantity. Products that have not been ordered or have invalid quantities are excluded.

Summary of Work:

This project focused on designing and building a database system to manage e-commerce transactions effectively. The process started with conceptual modeling, where the relationships between key entities like customers, products, orders, and payments were defined. This was followed by logical modeling to create normalized SQL tables, ensuring the database was both efficient and well-structured. Subsequently, a physical model was developed to implement the logical design in a real database system. Advanced SQL queries were created to demonstrate database functionality, including joins, subqueries, aggregate functions, and conditional logic. These queries highlighted the connections between tables and provided meaningful insights.

In addition to creating the database, the project involved interpreting the results of SQL queries. Examples included categorizing orders by value, filtering data with NOT EXISTS and NOT NULL, and aggregating information across multiple columns. Each query was explained with its purpose and results clearly outlined. Overall, the project provided hands-on experience in designing, implementing, and querying databases, showcasing the critical role of data organization and analysis in supporting e-commerce operations.