**SQL Queries:**

use gada\_electronics;

-- Retrieve all products with their respective quantities available in a specific warehouse by Pincode.

SELECT p.Product\_ID, p.Name, p.Price, w.Warehouse\_Quantity

FROM Product p

JOIN Warehouse w ON p.Product\_ID = w.Product\_ID

WHERE w.Pincode = '62704';

-- Update the quantity of a product in a customer's cart.

UPDATE Cart

SET Quantity = 2

WHERE Cart\_ID = 'CART001' AND Product\_ID = 'PROD001';

-- List customers who have made a purchase, along with their email and the total amount spent (consider discounts in calculations).

SELECT c.Customer\_ID, c.Email, SUM((crt.Price \* (1 - crt.Offer / 100)) \* crt.Quantity) AS Total\_Spent

FROM Customer c

JOIN Cart crt ON c.Customer\_ID = crt.Customer\_ID

JOIN Payment p ON crt.Cart\_ID = p.Cart\_ID

WHERE p.Status = 'Completed'

GROUP BY c.Customer\_ID;

-- Find all products that have never been added to a cart (i.e., identify unsold products).

SELECT p.Product\_ID, p.Name

FROM Product p

WHERE p.Product\_ID NOT IN (SELECT DISTINCT Product\_ID FROM Cart);

-- Show the total number of orders completed for each customer.

SELECT c.Customer\_ID, c.Name, COUNT(o.Order\_ID) AS Orders\_Completed

FROM Customer c

JOIN Orders o ON c.Customer\_ID = o.Customer\_ID

JOIN Payment p ON o.Payment\_ID = p.Payment\_ID

WHERE p.Status = 'Completed'

GROUP BY c.Customer\_ID;

-- Increment the discount on all products by 5% up to a maximum of 20% discount.

UPDATE Product

SET Discount = LEAST(Discount + 5, 20);

-- List all customers and the number of addresses they have registered, including those with no registered address.

SELECT c.Customer\_ID, c.Name, COUNT(a.Address\_ID) AS Address\_Count

FROM Customer c

LEFT JOIN Address a ON c.Customer\_ID = a.Customer\_ID

GROUP BY c.Customer\_ID;

-- Show total sales per product, only listing those with sales exceeding 1,000 rupees.

SELECT p.Product\_ID, p.Name, SUM(c.Quantity \* (c.Price - (c.Price \* c.Offer / 100))) AS TotalSales

FROM Product p

JOIN Cart c ON p.Product\_ID = c.Product\_ID

JOIN Payment pay ON c.Cart\_ID = pay.Cart\_ID AND pay.Status = 'Completed'

GROUP BY p.Product\_ID

HAVING TotalSales > 1000;

-- Display 'Address Not Provided' if Customer doesn’t have address.

SELECT

c.Customer\_ID,

c.Name AS CustomerName,

IFNULL(CONCAT(a.Street, ', ', a.City, ', ', a.State), 'Address Not Provided') AS Address

FROM

Customer c

LEFT JOIN Address a ON c.Customer\_ID = a.Customer\_ID;

-- Display top 3 most frequently purchased products.

SELECT Product\_ID, Name

FROM (

SELECT p.Product\_ID, p.Name,

RANK() OVER (PARTITION BY p.Product\_ID ORDER BY COUNT(c.Cart\_ID) DESC) AS PurchaseRank

FROM Product p

JOIN Cart c ON p.Product\_ID = c.Product\_ID

JOIN Payment pay ON c.Cart\_ID = pay.Cart\_ID AND pay.Status = 'Completed'

GROUP BY p.Product\_ID

) AS RankedProducts

WHERE PurchaseRank <= 3;

**Relational Algebra:**

1. Retrieve all products with their respective quantities available in a specific warehouse by Pincode:

π\_Product\_ID, Name, Price, Warehouse\_Quantity (σ\_Pincode='62704' (Product ⨝ Product.Product\_ID = Warehouse.Product\_ID Warehouse))

1. Update the quantity of a product in a customer's cart:

Does not Exist

1. List customers who have made a purchase, along with their email and the total amount spent:

γ\_Customer\_ID, Email; SUM((Price \* (1 - Offer / 100)) \* Quantity)->Total\_Spent (Customer ⨝ Customer.Customer\_ID = Cart.Customer\_ID (Cart ⨝ Cart.Cart\_ID = Payment.Cart\_ID σ\_Status='Completed' Payment))

1. Find all products that have never been added to a cart:

π\_Product\_ID, Name (Product) - π\_Product\_ID (Product ⨝ Product.Product\_ID = Cart.Product\_ID Cart)

1. Show the total number of orders completed for each customer:

γ\_Customer\_ID, Name; COUNT(Order\_ID)->Orders\_Completed (Customer ⨝ Customer.Customer\_ID = Orders.Customer\_ID (Orders ⨝ Orders.Payment\_ID = Payment.Payment\_ID σ\_Status='Completed' Payment))

1. Increment the discount on all products by 5% up to a maximum of 20% discount:

Does not Exist

1. List all customers and the number of addresses they have registered:

γ\_Customer\_ID, Name; COUNT(Address\_ID)->Address\_Count (Customer ⟕ Customer.Customer\_ID = Address.Customer\_ID Address)

1. Show total sales per product, only listing those with sales exceeding 1,000 rupees:

γ\_Product\_ID, Name; SUM(Quantity \* (Price - (Price \* Offer / 100)))->TotalSales (Product ⨝ Product.Product\_ID = Cart.Product\_ID (Cart ⨝ Cart.Cart\_ID = Payment.Cart\_ID σ\_Status='Completed' Payment)) σ\_TotalSales > 1000

1. Display 'Address Not Provided' if Customer doesn’t have address:

Does not Exist

1. Display top 3 most frequently purchased products:

Top3\_Ranked\_Products = π\_Product\_ID, Name (γ\_Product\_ID; RANK() OVER (PARTITION BY Product\_ID ORDER BY COUNT(Cart.Cart\_ID) DESC)->PurchaseRank (Product ⨝ Product.Product\_ID=Cart.Product\_ID (Cart ⨝ Cart.Cart\_ID=Payment.Cart\_ID σ\_Status='Completed' Payment)))

σ\_PurchaseRank≤3 (Top3\_Ranked\_Products)

**Relational Schema:**

The Relational Schema outlines each table along with its attributes, primary keys (PK), foreign keys (FK), and constraints like unique (U) and check constraints (C).

The Relational Schema represents the structure of our database, including:

* The interconnection between the tables through primary and foreign key relationships.
* The necessary constraints that ensure data integrity and uniqueness.















