A Report on

**DBMS Project**

Submitted By

**Arjun Mehta (K036)**

**SAP ID:70102300018**

Under The Guidance Of

**Ma’am Bhavna Bose**



**Mukesh Patel School of Technology & Management Engineering**

**Department of B.Tech CSE (Cybersecurity)**

**Vile Parle (w), Mumbai- 400056**

**TABLE OF CONTENTS:**

|  |  |  |
| --- | --- | --- |
| **Chapter** | **Title** | **Page No** |
| 1 | Problem Statement | 3 |
| 2 | ER Diagram | 4 |
| 3 | ER Diagram to Schema | 5 |
| 4 | Deliverables | 6 |
| 5 | Functional Dependencies | 7 |
| 6 | Normalisation | 8 |
| 7 | Creation of Tables | 9-14 |
| 8 | Inserting Values in Tables | 15-21 |
| 9 | Creating View of Each deliverable and displaying the values | 22-28 |
| 10 | Conclusion | 29 |

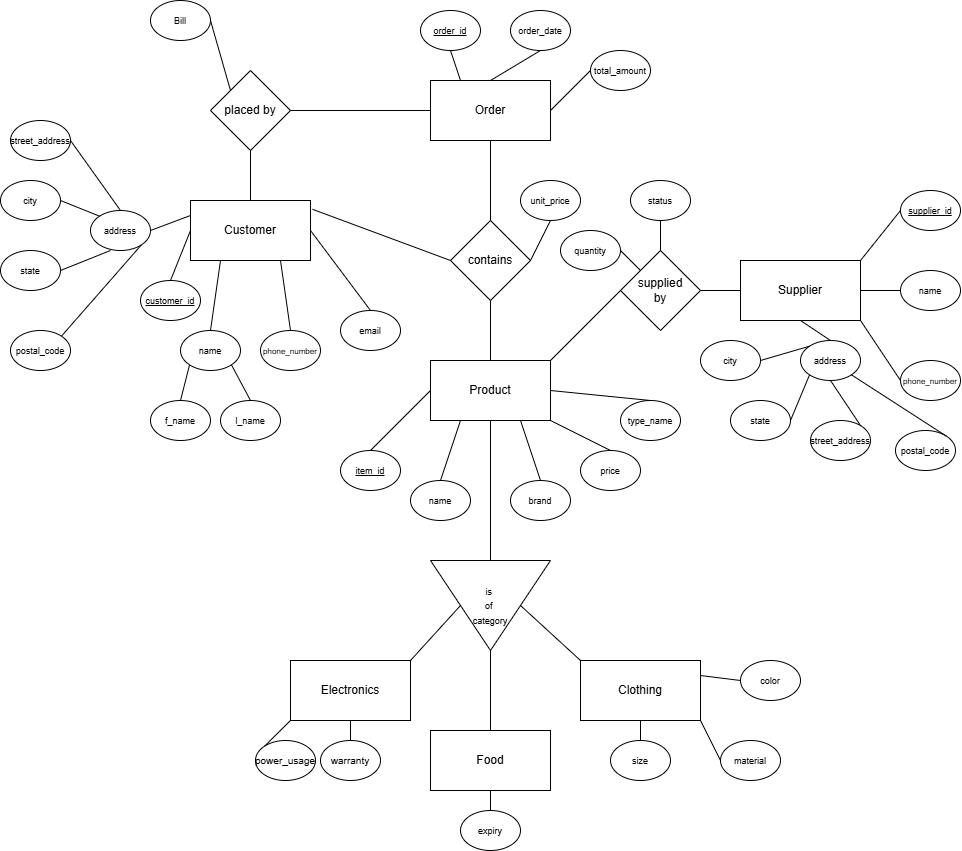
**Chapter-1: Problem Statement**

An e-commerce company wants to develop a comprehensive database to better manage its online store's operations. The company sells products like electronics, food items, and clothing. Each product has unique attributes. The database must track products, suppliers, orders, and customers efficiently.

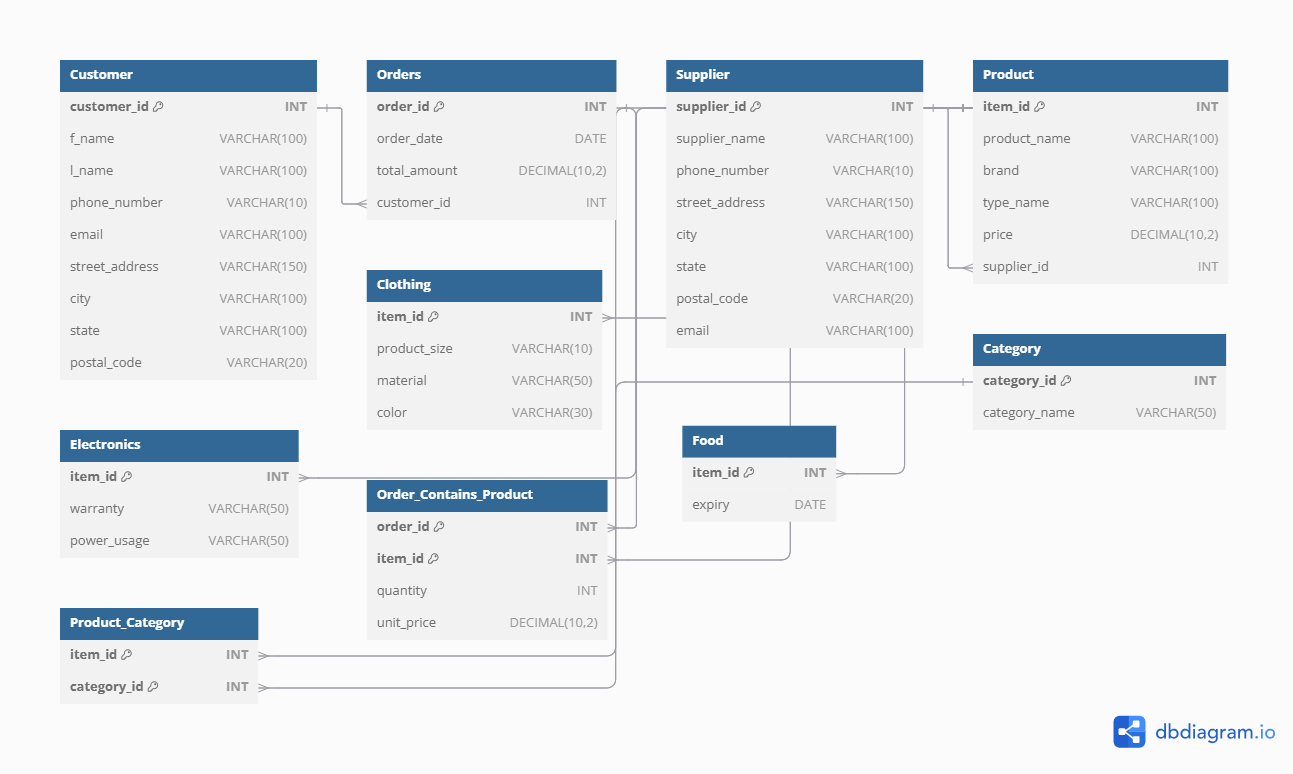
**Requirements:**

1. **Product Management**
   * Each product is identified by a unique item\_id and includes attributes like name, brand, price, and type\_name.
   * Categories:
     1. **Electronics**: Has attributes for warranty and power usage.
     2. **Food**: Includes an expiry date.
     3. **Clothing**: Includes size, color, and material.
2. **Supplier Management**
   * Each supplier is identified by a unique supplier\_id and includes name, phone number, etc.
   * Tracks products supplied by each supplier with attributes like quantity and status.
3. **Order Processing**
   * Orders are identified by order\_id and include order date, total amount, etc.
   * Orders contain multiple products and are linked to customers.
4. **Customer Management**
   * Customers are identified by customer\_id and include personal details like name, phone number, email, etc.

**Chapter-2:** **ER Diagram**



**Chapter-3: Er diagram to Schema**



**Chapter-4: Deliverables**

* **Customer Demographics and Contact Information:**  
  Report on customer details like contact information and demographics for CRM.
* **Order and Sales Analysis:**  
  Report on orders, order dates, total amounts, and customer purchasing behavior.
* **Supplier Performance Report:**  
  Details on suppliers, including the products they supply, and their contact/location info.
* **Product Inventory and Category Insights:**  
  Breakdown of available products by category, including attributes like price, color, etc.
* **Revenue and Product Sales Tracking:**  
  Report tracking revenue by product and category for sales forecasting.

**Chapter-5: Functional Dependencies**

1. **Customer Table**  
   customer\_id → f\_name, l\_name, phone\_number, email, street\_address, city, state, postal\_code
2. **Orders Table**  
   order\_id → order\_date, total\_amount, customer\_id
3. **Supplier Table**  
   supplier\_id → supplier\_name, phone\_number, street\_address, city, state, postal\_code, email
4. **Product Table**  
   item\_id → product\_name, brand, type\_name, price, supplier\_id
5. **Electronics Table**  
   item\_id → warranty, power\_usage
6. **Clothing Table**  
   item\_id → product\_size, material, color
7. **Food Table**  
   item\_id → expiry
8. **Category Table**  
   category\_id → category\_name
9. **Product\_Category Table**  
   (item\_id, category\_id) → item\_id, category\_id
10. **Order\_Contains\_Product Table**  
    (order\_id, item\_id) → quantity, unit\_price

**Chapter-6: Normalization of the Database Schema**

**1NF (First Normal Form):**  
The schema is in 1NF as each table contains only atomic values, with no repeating groups or arrays. Every field contains indivisible values, ensuring compliance with the first normal form.

**2NF (Second Normal Form):**  
The schema adheres to 2NF. All non-primary key attributes are fully functionally dependent on the primary key. For example, in the Product table, attributes such as brand, type\_name, and price depend solely on the item\_id, the table's primary key.

**3NF (Third Normal Form):**  
The schema is in 3NF since there are no transitive dependencies. Non-key attributes depend only on the primary key and not on other non-key attributes. For instance, in the Customer table, fields like phone\_number and email depend only on the customer\_id.

**BCNF (Boyce-Codd Normal Form):**  
The schema satisfies BCNF, as every determinant is a candidate key. For example, in the Orders table, order\_id determines all other attributes, and no anomalies exist with functional dependencies.

**4NF (Fourth Normal Form):**  
The schema is in 4NF as there are no multi-valued dependencies. Each table represents a single relationship, ensuring there are no redundancies or anomalies related to multiple independent attributes.

**5NF (Fifth Normal Form):**  
The schema complies with 5NF. It cannot be further decomposed without losing information or causing redundancy. All possible join dependencies are properly handled.

**Chapter-7: Table Creation**

CREATE TABLE Customer (

customer\_id INT PRIMARY KEY,

f\_name VARCHAR(100) NOT NULL,

l\_name VARCHAR(100) NOT NULL,

phone\_number VARCHAR(10) UNIQUE,

email VARCHAR(100) NOT NULL UNIQUE,

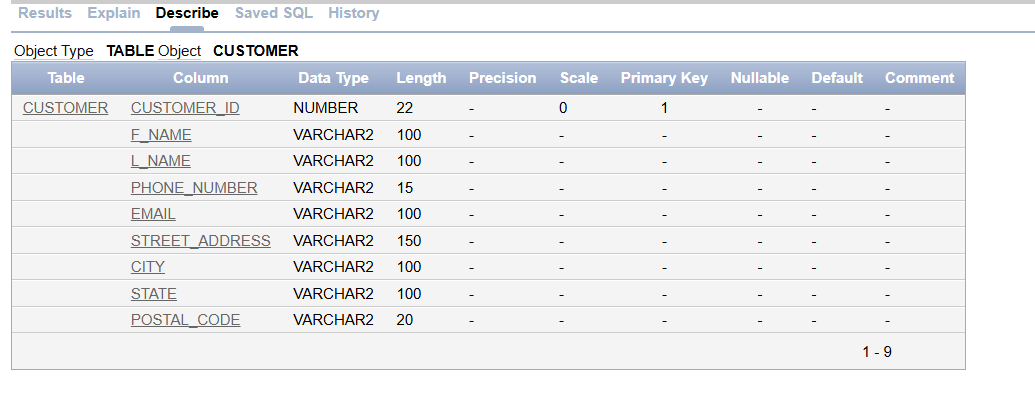
street\_address VARCHAR(150) NOT NULL,

city VARCHAR(100) NOT NULL,

state VARCHAR(100) NOT NULL,

postal\_code VARCHAR(20) NOT NULL

);



CREATE TABLE Orders (

order\_id INT PRIMARY KEY,

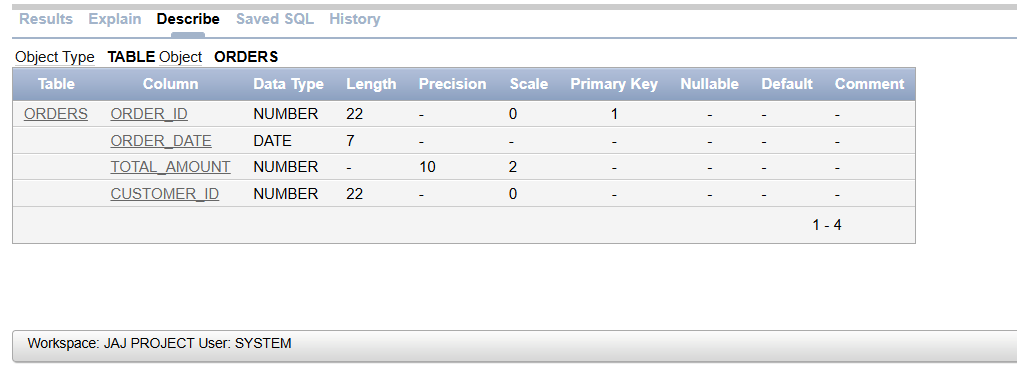
order\_date DATE NOT NULL,

total\_amount DECIMAL(10, 2) NOT NULL,

customer\_id INT CONSTRAINT FK\_Orders\_Customer REFERENCES Customer(customer\_id)

NOT NULL

);



CREATE TABLE Supplier (

supplier\_id INT PRIMARY KEY,

supplier\_name VARCHAR(100) NOT NULL,

phone\_number VARCHAR(10) UNIQUE,

street\_address VARCHAR(150) NOT NULL,

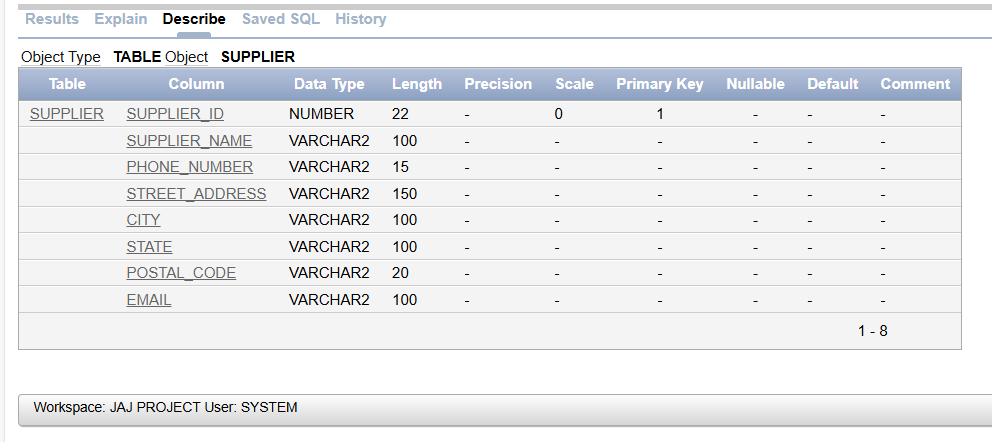
city VARCHAR(100) NOT NULL,

state VARCHAR(100) NOT NULL,

postal\_code VARCHAR(20) NOT NULL,

email VARCHAR(100) NOT NULL UNIQUE

);



CREATE TABLE Product (

item\_id INT PRIMARY KEY,

product\_name VARCHAR(100) NOT NULL,

brand VARCHAR(100) NOT NULL,

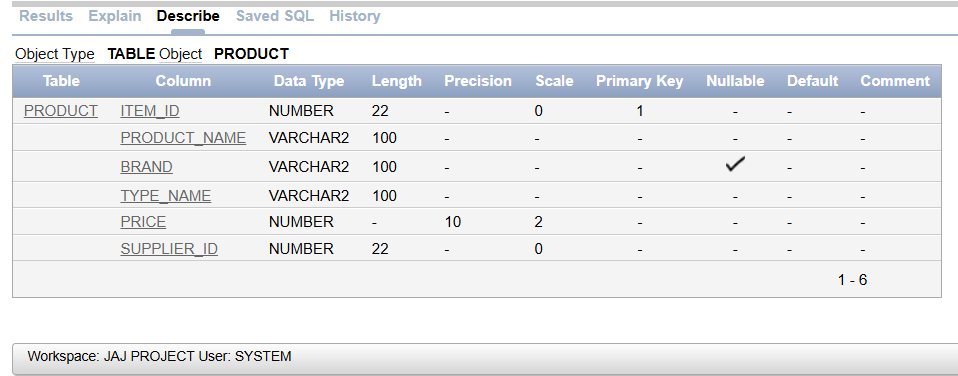
type\_name VARCHAR(100) NOT NULL,

price DECIMAL(10, 2) NOT NULL,

supplier\_id INT CONSTRAINT FK\_Product\_Supplier REFERENCES Supplier(supplier\_id)

NOT NULL

);



CREATE TABLE Electronics (

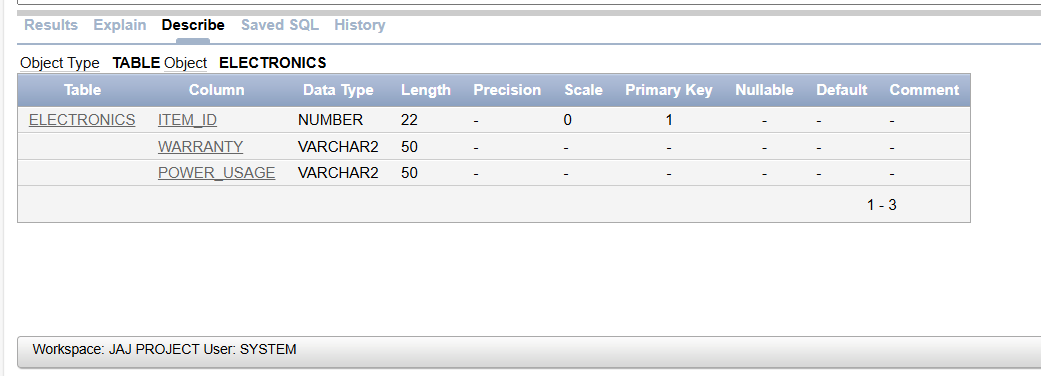
item\_id INT NOT NULL CONSTRAINT FK\_Electronics\_Product REFERENCES Product(item\_id),

warranty VARCHAR(50) NOT NULL,

power\_usage VARCHAR(50) NOT NULL,

PRIMARY KEY(item\_id)

);



CREATE TABLE Clothing (

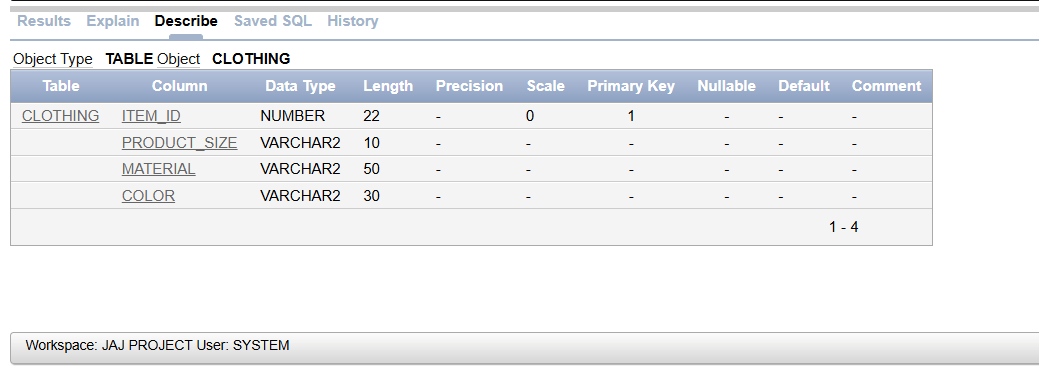
item\_id INT NOT NULL CONSTRAINT FK\_Clothing\_Product REFERENCES Product(item\_id),

product\_size VARCHAR(10) NOT NULL,

material VARCHAR(50) NOT NULL,

color VARCHAR(30) NOT NULL,

PRIMARY KEY (item\_id)

);

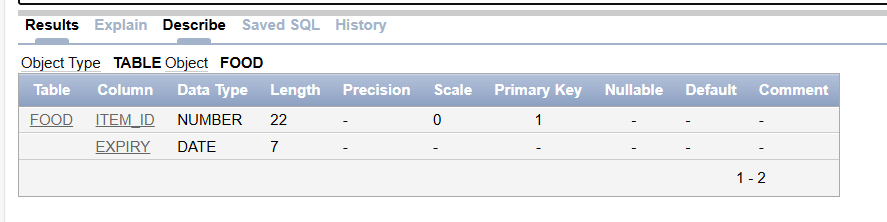
CREATE TABLE Food (

item\_id INT NOT NULL CONSTRAINT FK\_Food\_Product REFERENCES Product(item\_id),

expiry DATE NOT NULL,

PRIMARY KEY (item\_id)

);

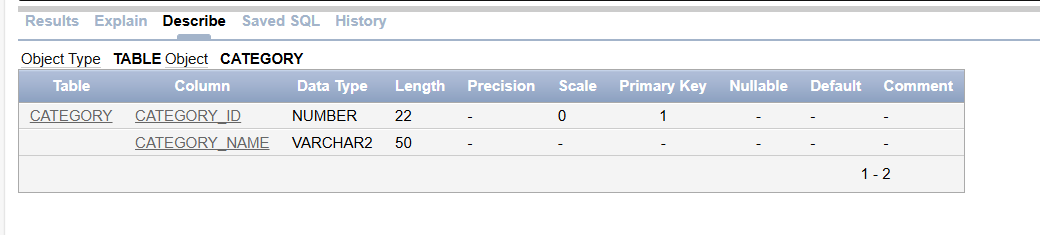


CREATE TABLE Category (

category\_id INT PRIMARY KEY,

category\_name VARCHAR(50) NOT NULL UNIQUE

);

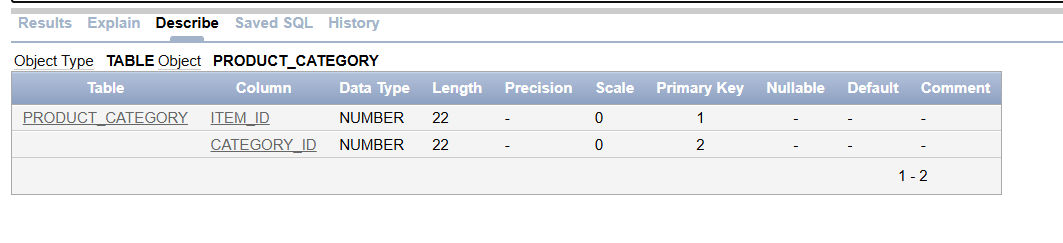


CREATE TABLE Product\_Category (

item\_id INT NOT NULL CONSTRAINT FK\_Product\_Category\_Product FOREIGN KEY REFERENCES Product(item\_id),

category\_id INT NOT NULL CONSTRAINT FK\_Product\_Category\_Category FOREIGN KEY REFERENCES Category(category\_id),

PRIMARY KEY (item\_id, category\_id)

);CREATE TABLE Order\_Contains\_Product (

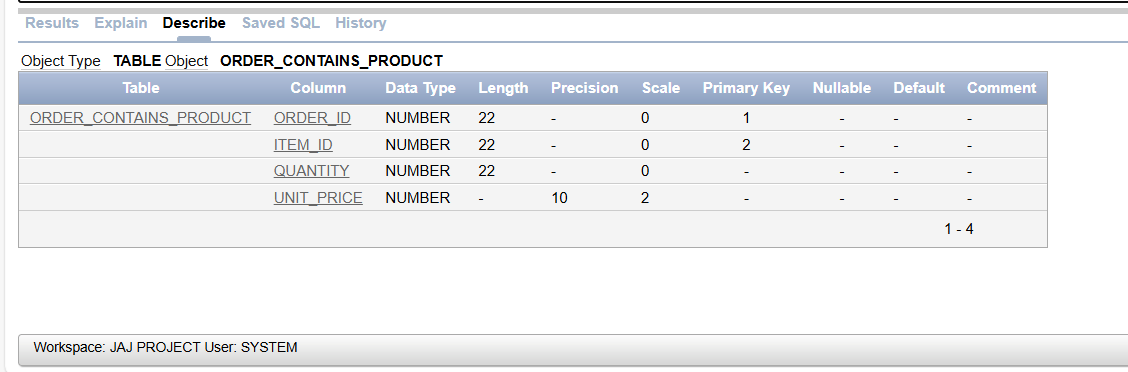
order\_id INT NOT NULL CONSTRAINT FK\_Order\_Contains\_Product\_Order FOREIGN KEY REFERENCES Orders(order\_id),

item\_id INT NOT NULL CONSTRAINT FK\_Order\_Contains\_Product\_Product FOREIGN KEY REFERENCES Product(item\_id),

quantity INT NOT NULL CHECK (quantity > 0),

unit\_price DECIMAL(10, 2) NOT NULL,

PRIMARY KEY (order\_id, item\_id)

);

**Chapter-8: Inserting Values**

1.) Inserting values in the Customer table:

CODE:

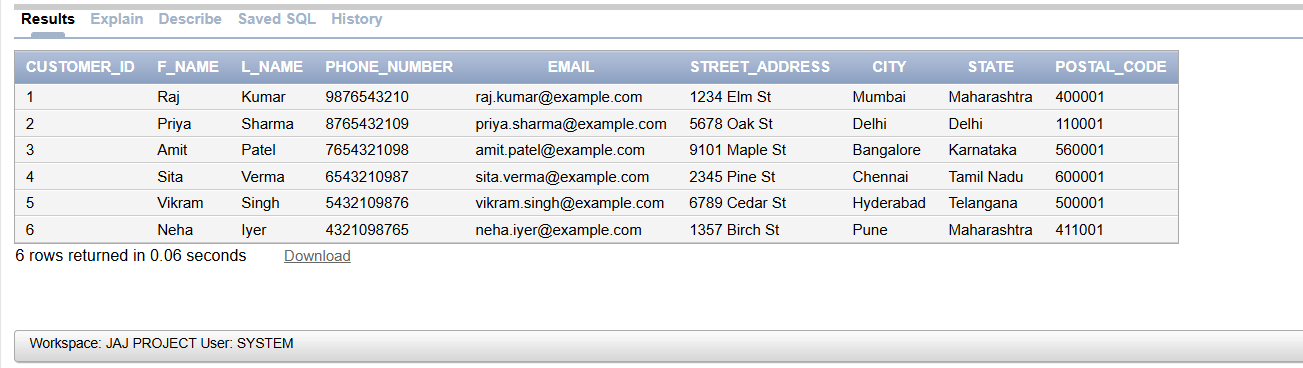
INSERT INTO CUSTOMER VALUES ('1', 'Raj', 'Kumar', '9876543210', 'raj.kumar@example.com', '1234 Elm St', 'Mumbai', 'Maharashtra', '400001');

INSERT INTO CUSTOMER VALUES ('2', 'Priya', 'Sharma', '8765432109', 'priya.sharma@example.com', '5678 Oak St', 'Delhi', 'Delhi', '110001');

INSERT INTO CUSTOMER VALUES ('3', 'Amit', 'Patel', '7654321098', 'amit.patel@example.com', '9101 Maple St', 'Bangalore', 'Karnataka', '560001');

INSERT INTO CUSTOMER VALUES ('4', 'Sita', 'Verma', '6543210987', 'sita.verma@example.com', '2345 Pine St', 'Chennai', 'Tamil Nadu', '600001');

INSERT INTO CUSTOMER VALUES ('5', 'Vikram', 'Singh', '5432109876', 'vikram.singh@example.com', '6789 Cedar St', 'Hyderabad', 'Telangana', '500001');

INSERT INTO CUSTOMER VALUES ('6', 'Neha', 'Iyer', '4321098765', 'neha.iyer@example.com', '1357 Birch St', 'Pune', 'Maharashtra', '411001');

2.) Inserting Values in the Orders Table:

CODE:

INSERT INTO ORDERS VALUES ('1', TO\_DATE('01/10/2024', 'MM/DD/YYYY'), '141000', '1');

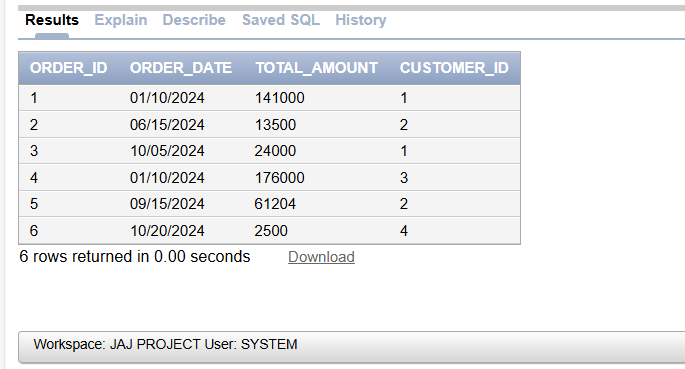
INSERT INTO ORDERS VALUES ('2', TO\_DATE('06/15/2024', 'MM/DD/YYYY'), '13500', '2');

INSERT INTO ORDERS VALUES ('3', TO\_DATE('10/05/2024', 'MM/DD/YYYY'), '24000', '1');

INSERT INTO ORDERS VALUES ('4', TO\_DATE('01/10/2024', 'MM/DD/YYYY'), '176000', '3');

INSERT INTO ORDERS VALUES ('5', TO\_DATE('09/15/2024', 'MM/DD/YYYY'), '61204', '2');

INSERT INTO ORDERS VALUES ('6', TO\_DATE('10/20/2024', 'MM/DD/YYYY'), '2500', '4');



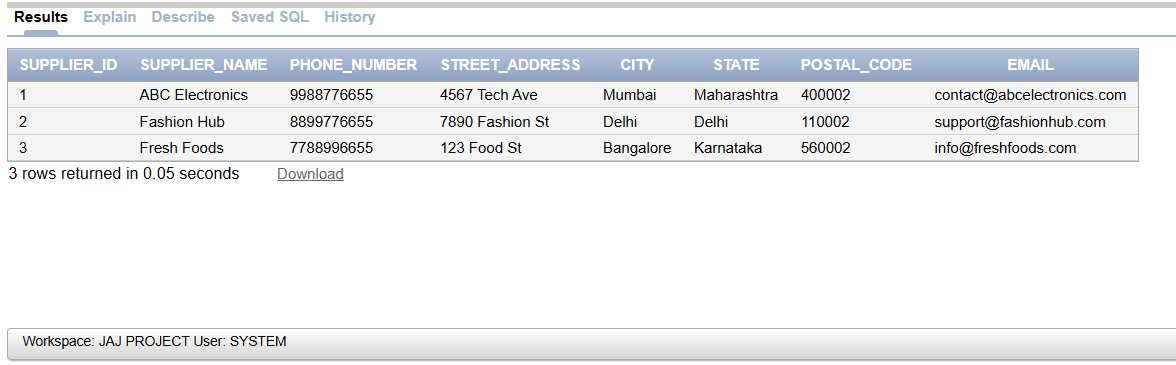
3.) Inserting Values in Supplier table:

CODE:

INSERT INTO SUPPLIER VALUES ('1', 'ABC Electronics', '9988776655', '4567 Tech Ave', 'Mumbai', 'Maharashtra', '400002', 'contact@abcelectronics.com');

INSERT INTO SUPPLIER VALUES ('2', 'Fashion Hub', '8899776655', '7890 Fashion St', 'Delhi', 'Delhi', '110002', 'support@fashionhub.com');

INSERT INTO SUPPLIER VALUES ('3', 'Fresh Foods', '7788996655', '123 Food St', 'Bangalore', 'Karnataka', '560002', 'info@freshfoods.com');



4.) Inserting Values in Product Table:

CODE:

INSERT INTO PRODUCT VALUES (1, 'Laptop', 'Brand A', 'Electronics', 45000, 1);

INSERT INTO PRODUCT VALUES (2, 'T-shirt', 'Brand B', 'Clothing', 500, 2);

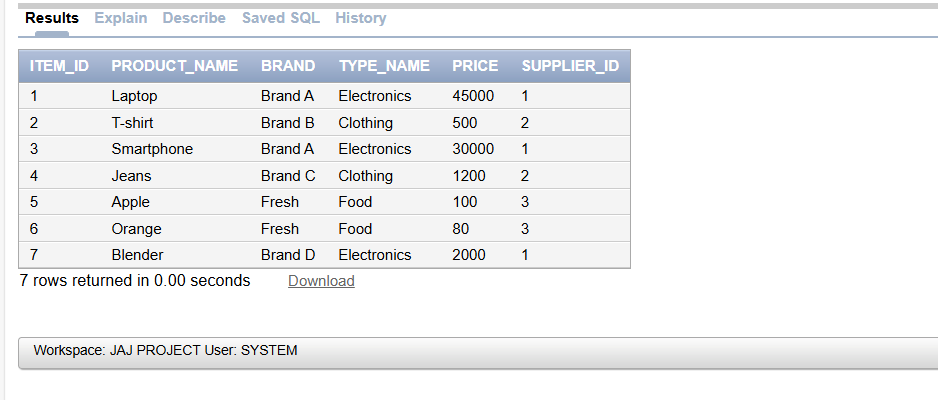
INSERT INTO PRODUCT VALUES (3, 'Smartphone', 'Brand A', 'Electronics', 30000, 1);

INSERT INTO PRODUCT VALUES (4, 'Jeans', 'Brand C', 'Clothing', 1200, 2);

INSERT INTO PRODUCT VALUES (5, 'Apple', 'Fresh', 'Food', 100, 3);

INSERT INTO PRODUCT VALUES (6, 'Orange', 'Fresh', 'Food', 80, 3);

INSERT INTO PRODUCT VALUES (7, 'Blender', 'Brand D', 'Electronics', 2000, 1);



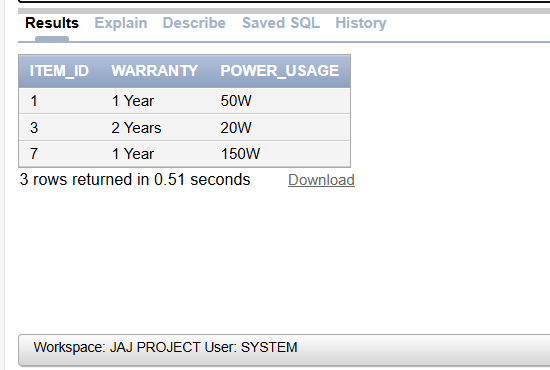
5.) Inserting Values in Electronics Table:

CODE:

INSERT INTO ELECTRONICS VALUES (1, '1 Year', 50);

INSERT INTO ELECTRONICS VALUES (3, '2 Years', 20);

INSERT INTO ELECTRONICS VALUES (7, '1 Year', 150);

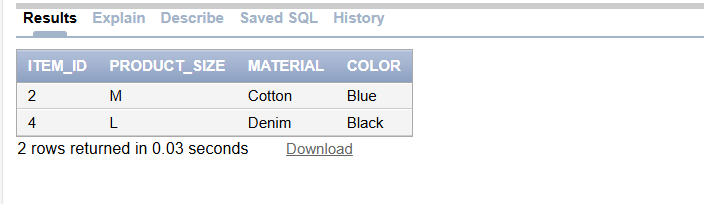


6.) Inserting Values in Clothing Table:

CODE:

INSERT INTO CLOTHING VALUES (2, 'M', 'Cotton', 'Blue');

INSERT INTO CLOTHING VALUES (4, 'L', 'Denim', 'Black');

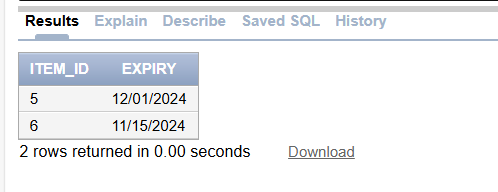


7.) Inserting Values in Food Table:

CODE:

INSERT INTO FOOD VALUES (5, '12/01/2024');

INSERT INTO FOOD VALUES (6, '11/15/2024');



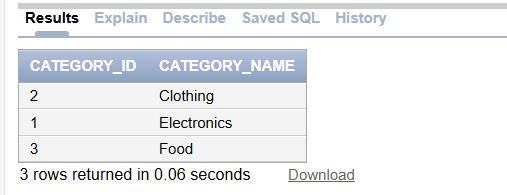
8.) Inserting Values Category Table:

CODE:

INSERT INTO CATEGORY VALUES (2, 'Clothing');

INSERT INTO CATEGORY VALUES (1, 'Electronics');

INSERT INTO CATEGORY VALUES (3, 'Food');



9.) Inserting Values in Product\_Category Table:

CODE:

INSERT INTO PRODUCT\_CATEGORY VALUES (1, 1);

INSERT INTO PRODUCT\_CATEGORY VALUES (2, 2);

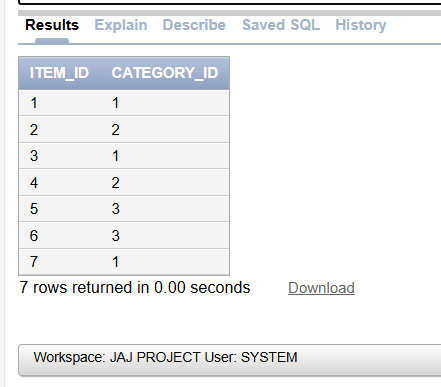
INSERT INTO PRODUCT\_CATEGORY VALUES (3, 1);

INSERT INTO PRODUCT\_CATEGORY VALUES (4, 2);

INSERT INTO PRODUCT\_CATEGORY VALUES (5, 3);

INSERT INTO PRODUCT\_CATEGORY VALUES (6, 3);

INSERT INTO PRODUCT\_CATEGORY VALUES (7, 1);



10.) Inserting Values in order\_contains\_product Table:

CODE:

INSERT INTO ORDER\_CONTAINS\_PRODUCT VALUES (1, 1, 2, 52000);

INSERT INTO ORDER\_CONTAINS\_PRODUCT VALUES (1, 2, 1, 32000);

INSERT INTO ORDER\_CONTAINS\_PRODUCT VALUES (2, 5, 15, 100);

INSERT INTO ORDER\_CONTAINS\_PRODUCT VALUES (3, 4, 10, 1800);

INSERT INTO ORDER\_CONTAINS\_PRODUCT VALUES (4, 2, 5, 35000);

INSERT INTO ORDER\_CONTAINS\_PRODUCT VALUES (5, 6, 8, 150.5);

INSERT INTO ORDER\_CONTAINS\_PRODUCT VALUES (6, 7, 1, 2500);

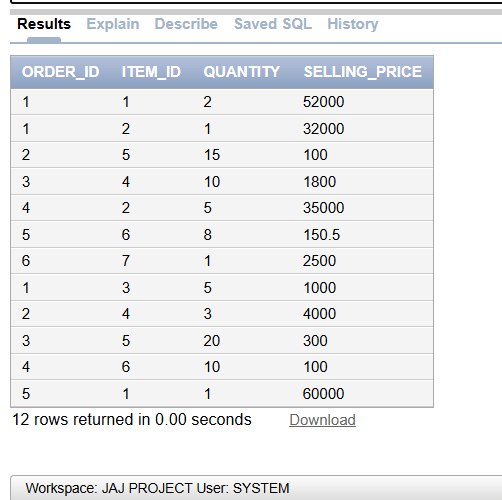
INSERT INTO ORDER\_CONTAINS\_PRODUCT VALUES (1, 3, 5, 1000);

INSERT INTO ORDER\_CONTAINS\_PRODUCT VALUES (2, 4, 3, 4000);

INSERT INTO ORDER\_CONTAINS\_PRODUCT VALUES (3, 5, 20, 300);

INSERT INTO ORDER\_CONTAINS\_PRODUCT VALUES (4, 6, 10, 100);

INSERT INTO ORDER\_CONTAINS\_PRODUCT VALUES (5, 1, 1, 60000);



**Chapter-9: Creating View for Each deliverable**

1. Retrieve all customers whose last name ends with 'ar' and whose phone number starts with '98'.

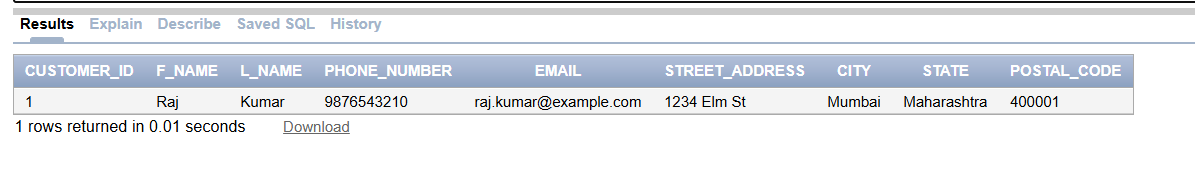
CODE:

CREATE VIEW nameendsar AS

SELECT \* FROM Customer

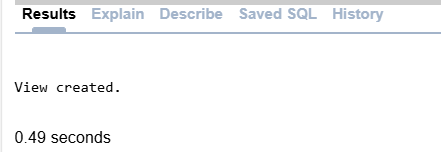
WHERE l\_name LIKE '%ar'

AND phone\_number LIKE '98%';



1. Retrieve all customers whose email contains 'example' and live in 'Mumbai' or 'Delhi'.

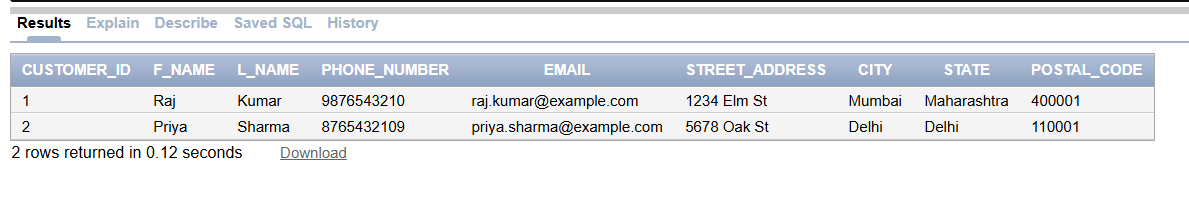
CODE:

CREATE VIEW emailcity AS

SELECT \* FROM Customer

WHERE email LIKE '%example%'

AND (city = 'Mumbai' OR city = 'Delhi');

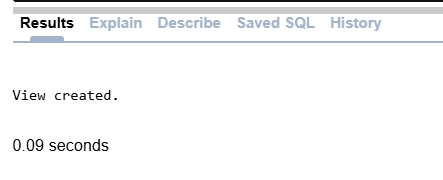


1. Find the top 3 customers based on their total purchase amount.

CODE:

CREATE VIEW Top3CustomersBySpending AS

SELECT c.customer\_id, c.f\_name, c.l\_name, SUM(o.total\_amount) AS total\_spent

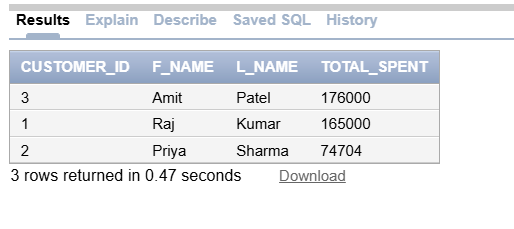
FROM Orders o

JOIN Customer c ON o.customer\_id = c.customer\_id

GROUP BY c.customer\_id, c.f\_name, c.l\_name

ORDER BY total\_spent DESC

FETCH FIRST 3 ROWS ONLY;

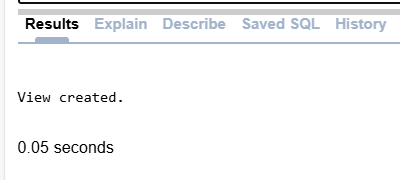


1. Retrieve the total sales amount for each order, along with the order date and customer details..

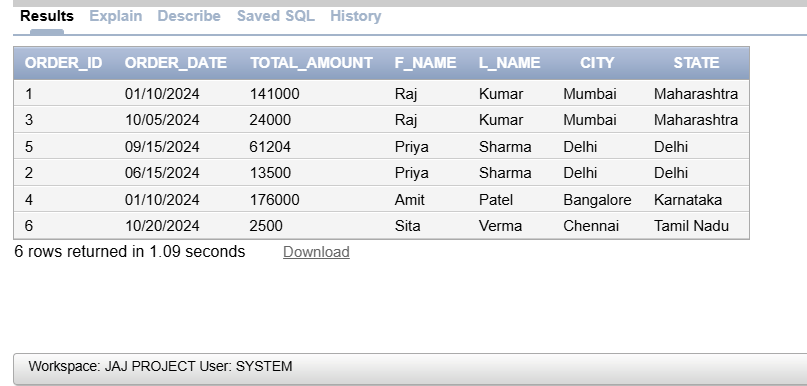
CODE:

CREATE VIEW OrderCustomerDetails AS

SELECT o.order\_id, o.order\_date, o.total\_amount, c.f\_name, c.l\_name, c.city, c.state

FROM Orders o

JOIN Customer c ON o.customer\_id = c.customer\_id;

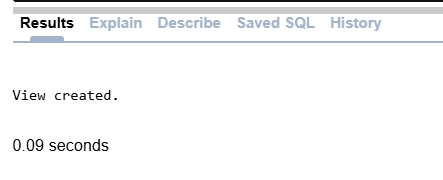


1. Find the number of products supplied by each supplier.

CODE:

CREATE VIEW Productsupplies AS

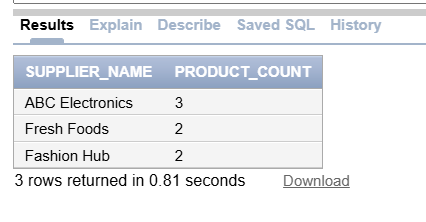
SELECT s.supplier\_name, COUNT(p.item\_id) AS product\_count

FROM Supplier s

JOIN Product p ON s.supplier\_id = p.supplier\_id

GROUP BY s.supplier\_name

ORDER BY product\_count DESC;

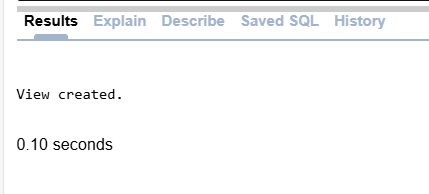


1. List all products by category, including their price and supplier.

CODE:

CREATE VIEW Categorypricesupplier AS

SELECT c.category\_name, p.product\_name, p.price, s.supplier\_name

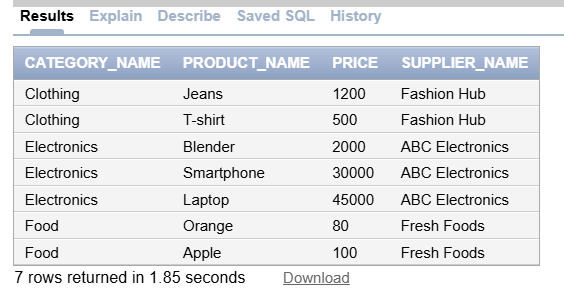
FROM Product p

JOIN Product\_Category pc ON p.item\_id = pc.item\_id

JOIN Category c ON pc.category\_id = c.category\_id

JOIN Supplier s ON p.supplier\_id = s.supplier\_id

ORDER BY c.category\_name;



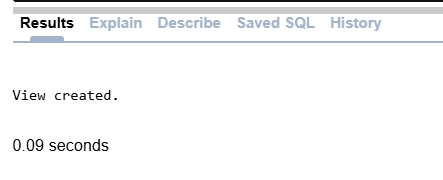
1. Find the average price of products in each category.

CODE:

CREATE VIEW AveragePriceofproduct AS

SELECT c.category\_name, AVG(p.price) AS avg\_price

FROM Product p

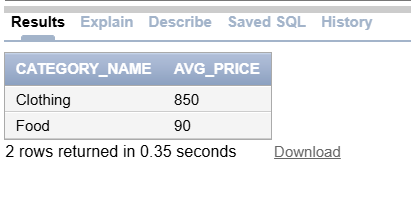
JOIN Product\_Category pc ON p.item\_id = pc.item\_id

JOIN Category c ON pc.category\_id = c.category\_id

WHERE c.category\_name IN ('Clothing', 'Food')

GROUP BY c.category\_name

ORDER BY avg\_price DESC;



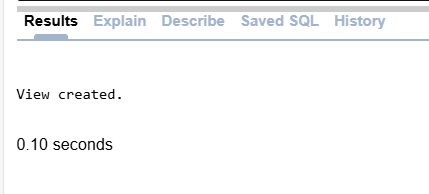
1. Calculate the total revenue generated from each product category.

CODE:

CREATE VIEW Revenueofeachproduct AS

SELECT c.category\_name, SUM(op.quantity \* p.price) AS total\_revenue

FROM Order\_Contains\_Product op

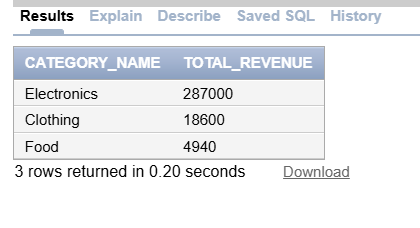
JOIN Product p ON op.item\_id = p.item\_id

JOIN Product\_Category pc ON op.item\_id = pc.item\_id

JOIN Category c ON pc.category\_id = c.category\_id

GROUP BY c.category\_name

ORDER BY total\_revenue DESC;

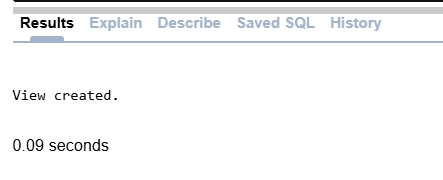


1. Find the 3 most frequently ordered products (by total quantity ordered).

CODE:

CREATE VIEW threemostordereditems AS

SELECT p.product\_name, SUM(op.quantity) AS total\_quantity\_ordered

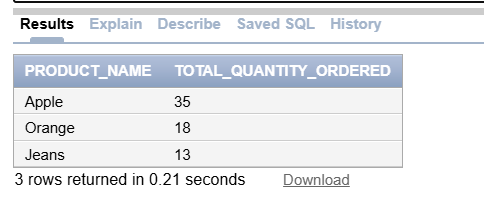
FROM Order\_Contains\_Product op

JOIN Product p ON op.item\_id = p.item\_id

GROUP BY p.product\_name

ORDER BY total\_quantity\_ordered DESC

FETCH FIRST 3 ROWS ONLY;

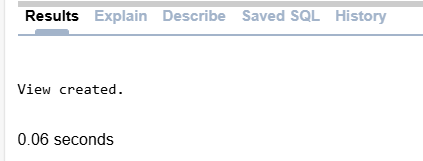


1. Retrieve all orders placed on date 15 and the associated customer details, and order them by the total amount in descending order.

CODE:

CREATE VIEW Orderon15 AS

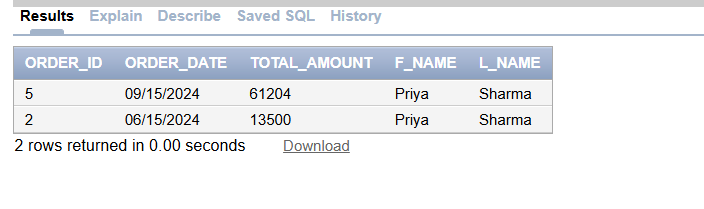
SELECT o.order\_id, o.order\_date, o.total\_amount, c.f\_name, c.l\_name

FROM Orders o

INNER JOIN Customer c ON o.customer\_id = c.customer\_id

WHERE EXTRACT(DAY FROM o.order\_date) = 15

ORDER BY o.total\_amount DESC;



1. Retrieve all clothing items that are blue in color and made of cotton, along with the product name.

CODE:

CREATE VIEW clothlbluecotton AS

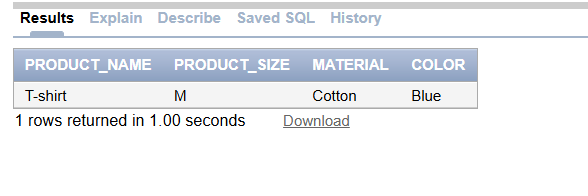
SELECT p.product\_name, cl.product\_size, cl.material, cl.color

FROM Clothing cl

INNER JOIN Product p ON cl.item\_id = p.item\_id

WHERE cl.color = 'Blue'

AND cl.material = 'Cotton';



**Chapter-10: Conclusion**

In conclusion, this database management project effectively fulfils the e-commerce company’s requirements by establishing a structured and efficient database for managing products, orders, customers, and suppliers. The project began with a comprehensive ER diagram that captured essential relationships, followed by a relational schema translated into an optimized database structure. Through normalization to the Fifth Normal Form (5NF), data redundancy and anomalies were minimized, ensuring data integrity across all tables. Functional dependencies were thoroughly examined to uphold normalization standards, strengthening the consistency and reliability of the database. Additionally, tailored SQL queries provide valuable insights into customer demographics, purchasing trends, inventory details, and supplier performance, supporting the company in data-driven decision-making. This well-rounded database solution enables streamlined operations and enhances the company’s ability to maintain high standards in customer satisfaction and operational efficiency.