VIT-AP
UNIVERSITY

UNIMART

Centralized Inventory Management for Campus Stores

Professor: Dr. Anupriya Krishnamurthy

Team members:

Tadimarri Mahammad Azhar	(23BCE9422)
Vikhil Sai	(23BCE9428)
M V Pramodh Krishna	(23BCE9464)
Preethi Darahasa	(23BCE8318)

Slot: D1

Abstract

The lack of a centralized inventory management system in campus stores often leads to inefficiencies for both students and vendors. Hostel students frequently face difficulties in locating specific items, resulting in time-consuming searches and frustration, especially when products are unavailable or out of stock. This lack of coordination between campus stores leads to significant inconvenience for students and a lack of real-time visibility for vendors.

To address this issue, the objective of this project is to design and implement a comprehensive Database Management System (DBMS) that centralizes the inventory data of all campus stores. This system will allow vendors to seamlessly update their stock information, ensuring accurate and up-to-date availability details across all participating stores. By integrating inventory data, the system will provide a unified platform that vendors can access to maintain consistency and visibility of their products.

In addition, the system will enable students to efficiently query and locate desired items within campus stores. Through this centralized approach, students will be able to make informed purchase decisions without wasting time on searching through different stores or dealing with out-of-stock products. The DBMS will streamline the entire process, making it easier for students to access necessary products while allowing vendors to manage their inventory effectively.

Ultimately, the proposed system aims to enhance the overall shopping experience for students and optimize inventory tracking for campus vendors. By centralizing the inventory information and improving accessibility, the system will reduce inefficiencies and foster a more organized and responsive campus retail environment. This integrated solution will contribute to a more convenient and time-efficient purchasing experience for students and ensure better management of stock levels for vendors.

Problem stateme

Problem Statement –UniMart Inventory Challenges and Opportunities

Hostel students at campus facilities frequently face challenges in locating specific items within various campus stores due to the lack of a centralized inventory management system. Each store operates independently, leading to disorganized and inconsistent stock information across different locations. This fragmentation makes it difficult for students to easily find the products they need, forcing them to spend excessive amounts of time searching through multiple stores. Often, items are unavailable or out of stock, adding to the frustration and inconvenience for students. As a result, students struggle to efficiently manage their shopping experience, which can lead to unnecessary delays and unmet needs.

Furthermore, vendors managing campus stores face their own difficulties due to the absence of a cohesive inventory system. Without real-time visibility into the stock levels across all stores, vendors are often unaware of which items need to be restocked or which items are overstocked. This lack of synchronization leads to inefficiencies in inventory management and poor decision-making when it comes to stock replenishment. Additionally, without a unified database, vendors and students must manually track stock, which increases the chances of errors or missed opportunities to optimize stock levels.

In addition to these challenges, **many students possess several used reference books and textbooks that are no longer of use to them. These books often end up stored away or discarded as waste.** However, they could be extremely valuable to other students, particularly those who are unable to afford high-priced new books. There is a clear opportunity to create a platform where students can register to sell their used books at affordable prices. This not only promotes a culture of reuse and affordability but also ensures that learning resources are accessible to a broader group of students.

To address these issues, a centralized system that integrates inventory data from all campus stores is necessary. Such a system would enable students to quickly locate and purchase the items they need by providing real-time stock availability, product details, and store locations. Simultaneously, it would allow vendors to efficiently update inventory levels, ensuring that stock information is always accurate and up-to-date. By centralizing inventory management and providing a unified platform for both students and vendors, this system would eliminate the time-consuming searches and inconveniences students currently face while helping vendors maintain optimal inventory levels. Additionally, by including a student-driven resale feature for used books, the system would foster affordability, sustainability, and greater access to essential academic materials.

UniMart Business Rules

1. Student Registration

Every student must register with a unique studentID (RegNo).

Required fields: Name, Email, Phone number

Students can place and track orders, maintain a wish list, and sell personal items.

2. Store Management

Each store is uniquely identified by a storeID.

Stores must have: storeName, location, and managerID.

A store can maintain multiple products (1:N relationship with STOCK).

A store can receive multiple orders (1:N relationship with ORDER).

Stores may list contact details such as phone number and email.

3. Product Inventory (STOCK)

Each product is uniquely identified by a productID (UPC in the diagram).

A product is associated with one store via storeID (M:1 relationship with STORE).

Product details include: itemName, category (optional), brand (optional), specification (optional), price, stockCount, and available (status).

Stock levels must be updated after every purchase.

If stock is zero, the product becomes unavailable for new orders.

STOCK is involved in several relationships:

- Contained in orders (M:N with ORDER).

- Sold to students (M:N with STUDENT via purchases).
- Can also be listed by students for sale.

4. Order Processing

Orders are uniquely identified by orderID.

Each order must:

- Be placed by one student (M:1 with STUDENT).
- Belong to one store (M:1 with STORE).
- Have attributes: orderDate, orderStatus (Pending, Confirmed, Cancelled), totalAmount, paymentMode, and grOrderID (external or grouped reference).

An order may contain multiple products (M:N with STOCK).

5. Order Details

Managed through the contains relationship between ORDER and STOCK.

Each order entry records:

- productID, quantity, and price at the time of purchase.

Total order amount is the sum of all product prices multiplied by quantities.

6. Payment Handling

Payments can be made:

- Offline (cash or physical POS systems).
- Online using store QR codes.

All payments must be associated with an order record.

7. Stock Management

Products that are out of stock cannot be added to orders.

Store managers are responsible for:

- Updating product availability.
- Managing stock levels regularly and accurately.

8. Student-Sellable Items

Students are allowed to sell personal items (e.g., used books) through Uni Mart.

A relationship (lists) exists between STUDENT and STOCK (1:N or M:N as per system design):

- A student can list multiple items for sale.
- Each item listed is recorded in the STOCK table like store items.

The system must capture:

- Listing student's ID
- Product details: itemName, category, price, available status
- Seller type (to differentiate between store and student items)

These listings follow the same order and payment processes as store-sold items.

9. Campus-Only Operations

Uni Mart is strictly for on-campus use only.

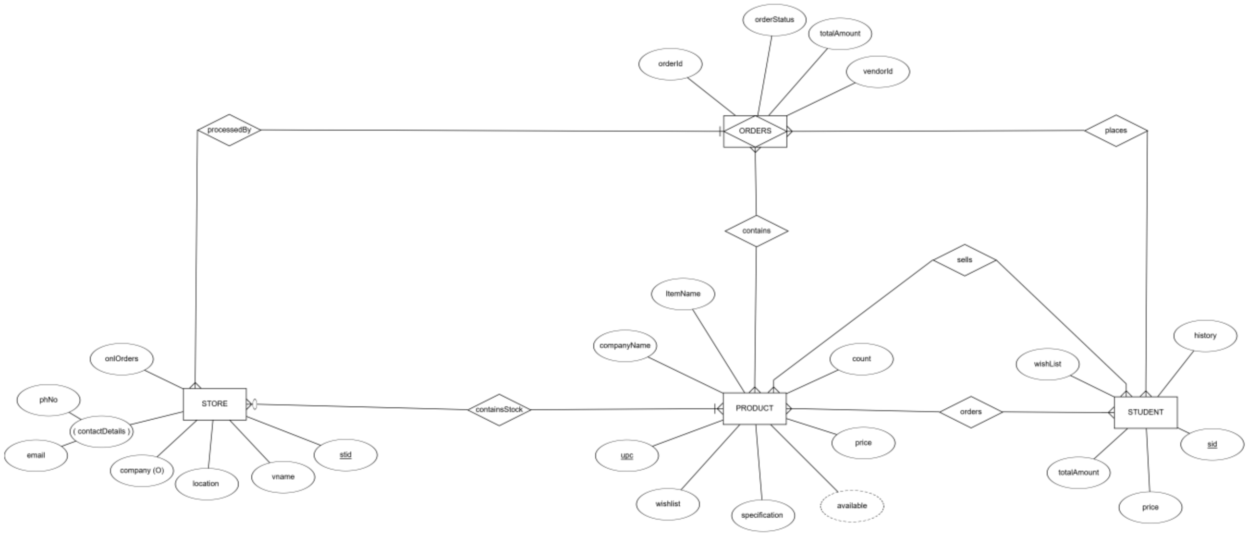
No services, listings, or deliveries are permitted outside university grounds.

Registration and transactions are limited to university-affiliated users and stores.

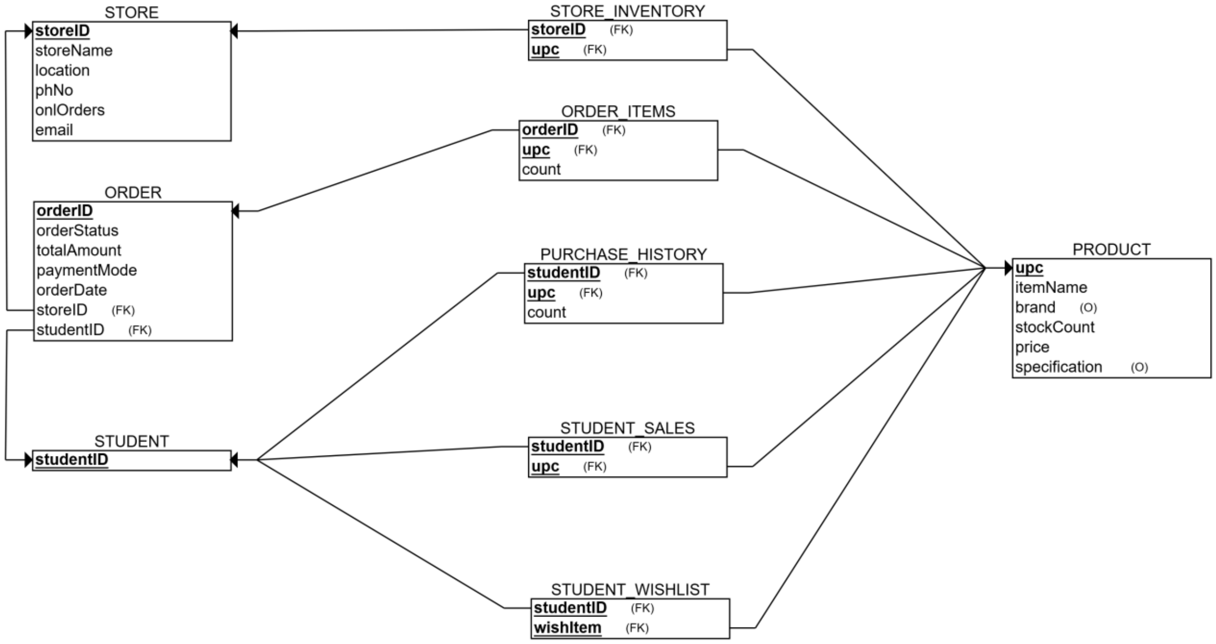
Entity Relationships

1. **Student –Order**
   - **1:N** (One student can place many orders)
2. **Order –Stock**
   - **M:N** (An order can contain many stock items, and a stock item can be in many orders)
3. **Store –Stock**
   - **1:N** (One store maintains many stock items)
4. **Store –Order**
   - **1:N** (One store can receive many orders)
5. **Stock –Student (Purchases)**
   - **M:N** (Many students can purchase many stock items)
6. **Student –Stock (Lists items for sale)**
   - **1:N** or **M:N** depending on implementation
(One student can list many items; co-listing is optional)

Entity relation diagram



Relational schema



STORE
storeID
storeName
location
phNo
onlOrders
email

STORE_INVENTORY
storeID (FK)
upc (FK)

ORDER_ITEMS
orderID (FK)
upc (FK)
count

ORDER
orderID
orderStatus
totalAmount
paymentMode
orderDate
storeID (FK)
studentID (FK)

PURCHASE_HISTORY
studentID (FK)
upc (FK)
count

PRODUCT
upc
itemName
brand (O)
stockCount
price
specification (O)

STUDENT
studentID

STUDENT_SALES
studentID (FK)
upc (FK)

STUDENT_WISHLIST
studentID (FK)
wishItem (FK)

Data Base creation:

```sql
CREATE TABLE UniMart_Store (
   storeId INT PRIMARY KEY,
   storeName VARCHAR2(50),
   Location VARCHAR2(100),
   phNo NUMBER,
   onlOrder VARCHAR2(3) CHECK(onlOrder IN ('yes', 'no')),
   email VARCHAR2(50)
);

CREATE TABLE UniMart_Student (
   studentId INT PRIMARY KEY
);

CREATE TABLE UniMart_Product (
   Upc INT PRIMARY KEY,
   itemName VARCHAR2(50) NOT NULL,
   brand VARCHAR2(50),
   stockCount INT NOT NULL,
   price NUMBER NOT NULL,
   specification VARCHAR2(300)
);

CREATE TABLE UniMart_Order (
   orderId INT PRIMARY KEY,
   totalAmount NUMBER,
   paymentMode VARCHAR2(50),
   orderDate DATE,
   storeId INT,
   studentId INT,
   FOREIGN KEY (storeId) REFERENCES UniMart_Store(storeId),
   FOREIGN KEY (studentId) REFERENCES UniMart_Student(studentId)
);

CREATE TABLE UniMart_Store_Inventory (
   storeId INT,
   Upc INT,
   PRIMARY KEY(storeId, Upc),
   FOREIGN KEY (storeId) REFERENCES UniMart_Store(storeId),
   FOREIGN KEY (Upc) REFERENCES UniMart_Product(Upc)
);

CREATE TABLE UniMart_Order_Items (
   orderId INT,
   Upc INT,
   count INT,
   PRIMARY KEY(orderId, Upc),
   FOREIGN KEY (orderId) REFERENCES UniMart_Order(orderId),
   FOREIGN KEY (Upc) REFERENCES UniMart_Product(Upc)
);

CREATE TABLE UniMart_Purchase_History (
   studentId INT,
   Upc INT,
   count INT,
   PRIMARY KEY(studentId, Upc),
   FOREIGN KEY (studentId) REFERENCES UniMart_Student(studentId),
   FOREIGN KEY (Upc) REFERENCES UniMart_Product(Upc)
);

CREATE TABLE UniMart_Student_Sales (
   studentId INT,
   Upc INT,
   PRIMARY KEY(studentId, Upc),
   FOREIGN KEY (studentId) REFERENCES UniMart_Student(studentId),
   FOREIGN KEY (Upc) REFERENCES UniMart_Product(Upc)
);

CREATE TABLE UniMart_Student_WishList (
   studentId INT,
   Upc INT,
   PRIMARY KEY(studentId, Upc),
   FOREIGN KEY (studentId) REFERENCES
UniMart_Student(studentId),
   FOREIGN KEY (Upc) REFERENCES
UniMart_Product(Upc)
);


-- UniMart_Store
INSERT INTO UniMart_Store VALUES (1, 'Bits
and Bites', 'Mens_Hostel1_PettyShop',
9876543210, 'yes', 'bits@unimart.in');
INSERT INTO UniMart_Store VALUES (2,
'Shakers and Movers',
'Mens_Hostel2_PettyShop', 9876543211, 'no',
'shakers@unimart.in');
INSERT INTO UniMart_Store VALUES (3, 'Zuzu
Zone', 'Mens_Hostel3_PettyShop', 9876543212,
'yes', 'zuzu@unimart.in');
INSERT INTO UniMart_Store VALUES (4, 'Chat',
'Mens_Hostel4_PettyShop', 9876543213, 'no',
'chat@unimart.in');
INSERT INTO UniMart_Store VALUES (5, 'Maggie
Hotspot', 'Mens_Hostel5_PettyShop',
9876543214, 'yes', 'maggie@unimart.in');
INSERT INTO UniMart_Store VALUES (6, 'Swagat
Canteen', 'Mens_Hostel6_PettyShop',
9876543215, 'yes', 'swagat@unimart.in');
INSERT INTO UniMart_Store VALUES (7, 'Ladies
Zone', 'Ladies_Hostel1_PettyShop',
9876543216, 'yes', 'ladieszone@unimart.in');

-- UniMart_Student
INSERT INTO UniMart_Student VALUES (101);
INSERT INTO UniMart_Student VALUES (102);
INSERT INTO UniMart_Student VALUES (103);
INSERT INTO UniMart_Student VALUES (104);
INSERT INTO UniMart_Student VALUES (105);
INSERT INTO UniMart_Student VALUES (106);
INSERT INTO UniMart_Student VALUES (107);
INSERT INTO UniMart_Student VALUES (108);
INSERT INTO UniMart_Student VALUES (109);
INSERT INTO UniMart_Student VALUES (110);

-- UniMart_Product
INSERT INTO UniMart_Product VALUES
(8901030371213, 'Parle-G Biscuit', 'Parle', 200,
10, 'Glucose biscuits 80g');
INSERT INTO UniMart_Product VALUES
(8901058845662, 'Maggie Noodles', 'Nestle',
150, 15, 'Instant noodles 70g');
INSERT INTO UniMart_Product VALUES
(8901491100019, 'Lays Chips', 'PepsiCo', 100,
20, 'Masala flavor 50g');
INSERT INTO UniMart_Product VALUES
(8901102063045, 'Appy Fizz', 'Parle Agro', 80,
25, 'Carbonated Apple Drink 250ml');
INSERT INTO UniMart_Product VALUES
(8901764061112, 'Thumbs Up', 'Coca-Cola',
120, 35, 'Soft Drink 500ml');
INSERT INTO UniMart_Product VALUES
(9002490100014, 'Red Bull', 'Red Bull', 50, 110,
'Energy Drink 250ml');
INSERT INTO UniMart_Product VALUES
(8901063010308, 'Britannia Cake', 'Britannia',
90, 25, 'Eggless Chocolate Cake 100g');
INSERT INTO UniMart_Product VALUES
(8901063901021, 'Good Day Cookies',
'Britannia', 130, 20, 'Cashew Cookies 60g');
INSERT INTO UniMart_Product VALUES
(8901262011133, 'Amul Kool', 'Amul', 70, 25,
'Flavored Milk 200ml');
INSERT INTO UniMart_Product VALUES
(8901233022232, 'Dairy Milk', 'Cadbury', 180,
40, 'Milk Chocolate 50g');

-- UniMart_Store_Inventory
INSERT INTO UniMart_Store_Inventory VALUES
(1, 8901030371213);
INSERT INTO UniMart_Store_Inventory VALUES
(1, 8901058845662);
INSERT INTO UniMart_Store_Inventory VALUES
(2, 8901491100019);
```

```
INSERT INTO UniMart_Store_Inventory VALUES
(3, 8901102063045);
INSERT INTO UniMart_Store_Inventory VALUES
(4, 8901764061112);
INSERT INTO UniMart_Store_Inventory VALUES
(5, 9002490100014);
INSERT INTO UniMart_Store_Inventory VALUES
(6, 8901063010308);
INSERT INTO UniMart_Store_Inventory VALUES
(7, 8901063901021);
INSERT INTO UniMart_Store_Inventory VALUES
(7, 8901262011133);
INSERT INTO UniMart_Store_Inventory VALUES
(7, 8901233022232);

-- UniMart_Order
INSERT INTO UniMart_Order VALUES (201, 50,
'UPI', TO_DATE('2024-03-01', 'YYYY-MM-DD'), 1,
101);
INSERT INTO UniMart_Order VALUES (202, 70,
'Cash', TO_DATE('2024-03-02', 'YYYY-MM-DD'),
2, 102);
INSERT INTO UniMart_Order VALUES (203, 45,
'Paytm', TO_DATE('2024-03-03', 'YYYY-MM-
DD'), 3, 103);
INSERT INTO UniMart_Order VALUES (204, 80,
'Google Pay', TO_DATE('2024-03-04', 'YYYY-
MM-DD'), 4, 104);
INSERT INTO UniMart_Order VALUES (205, 35,
'UPI', TO_DATE('2024-03-05', 'YYYY-MM-DD'), 5,
105);

-- UniMart_Order_Items
INSERT INTO UniMart_Order_Items VALUES
(201, 8901030371213, 2);
INSERT INTO UniMart_Order_Items VALUES
(201, 8901058845662, 1);
INSERT INTO UniMart_Order_Items VALUES
(202, 8901491100019, 2);
INSERT INTO UniMart_Order_Items VALUES
(203, 8901102063045, 1);
INSERT INTO UniMart_Order_Items VALUES
(204, 8901764061112, 2);
INSERT INTO UniMart_Order_Items VALUES
(205, 9002490100014, 1);

-- UniMart_Purchase_History
INSERT INTO UniMart_Purchase_History
VALUES (101, 8901030371213, 5);
INSERT INTO UniMart_Purchase_History
VALUES (102, 8901491100019, 2);
INSERT INTO UniMart_Purchase_History
VALUES (103, 8901102063045, 3);
INSERT INTO UniMart_Purchase_History
VALUES (104, 8901764061112, 1);
INSERT INTO UniMart_Purchase_History
VALUES (105, 9002490100014, 4);

-- UniMart_Student_Sales
INSERT INTO UniMart_Student_Sales VALUES
(106, 8901063010308);
INSERT INTO UniMart_Student_Sales VALUES
(107, 8901063901021);
INSERT INTO UniMart_Student_Sales VALUES
(108, 8901262011133);
INSERT INTO UniMart_Student_Sales VALUES
(109, 8901233022232);

-- UniMart_Student_WishList
INSERT INTO UniMart_Student_WishList
VALUES (110, 8901030371213);
INSERT INTO UniMart_Student_WishList
VALUES (110, 8901058845662);
INSERT INTO UniMart_Student_WishList
VALUES (108, 8901233022232);
```

SQL queries and their respective Relational algebra queries:

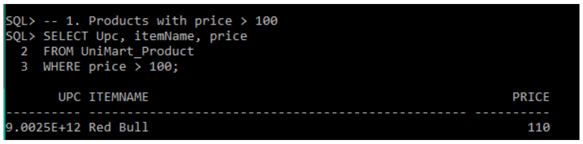
Here’s the text with the S
relational algebra expressions bolded for easy pasting into Word:

1. **SQL Query**:
   SELECT Upc, itemName, price
   FROM UniMart_Product
   WHERE price > 100;

Relational Algebra:
π_Upc, itemName, price(σ_price > **100(UniMart_Product))**

Explanation:

- σ (Selection) filters the rows where the price is greater than 100.
- π (Projection) selects the columns Upc, itemName, and price from the filtered table.



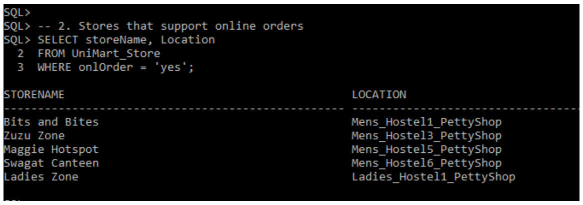
```
SQL> -- 1. Products with price > 100
SQL> SELECT Upc, itemName, price
  2  FROM UniMart_Product
  3  WHERE price > 100;

       UPC ITEMNAME                                                PRICE
---------- -------------------------------------------------- ----------
9.0025E+12 Red Bull                                                  110
```

2. **SQL Query**:
   SELECT storeName, Location
   FROM UniMart_Store
   WHERE onlOrder = 'yes';

Relational Algebra:
π_storeName, Location(σ_onlOrder = **'yes'(UniMart_Store))**

Explanation:

- σ filters for stores where online orders are enabled.
- π selects the storeName and Location columns from the filtered stores.



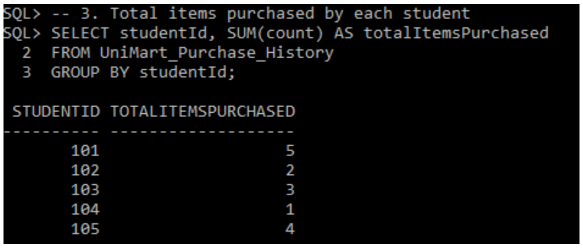
```
SQL>
SQL> -- 2. Stores that support online orders
SQL> SELECT storeName, Location
  2  FROM UniMart_Store
  3  WHERE onlOrder = 'yes';

STORENAME                                          LOCATION
-------------------------------------------------- ----------------------------------
Bits and Bites                                     Mens_Hostel1_PettyShop
Zuzu Zone                                          Mens_Hostel3_PettyShop
Maggie Hotspot                                     Mens_Hostel5_PettyShop
Swagat Canteen                                     Mens_Hostel6_PettyShop
Ladies Zone                                        Ladies_Hostel1_PettyShop
```

3. **SQL Query**:
   SELECT studentId, SUM(count) AS totalItemsPurchased
   FROM UniMart_Purchase_History
   GROUP BY studentId;

Relational Algebra:
γ_studentId, SUM(count → **totalItemsPurchased)(UniMart_Purchase_History)**

Explanation:

- γ (Aggregation) is used to compute the sum of count for each studentId, with the result being aliased as totalItemsPurchased.



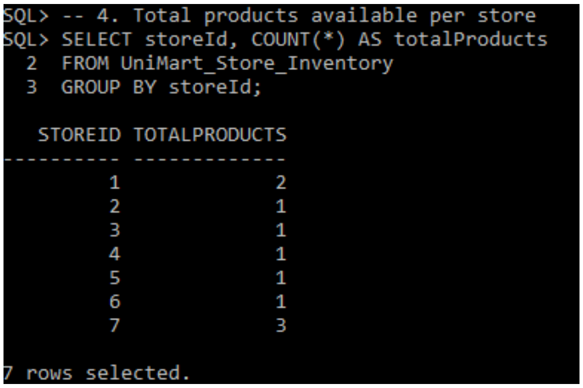
```
SQL> -- 3. Total items purchased by each student
SQL> SELECT studentId, SUM(count) AS totalItemsPurchased
  2  FROM UniMart_Purchase_History
  3  GROUP BY studentId;

 STUDENTID TOTALITEMSPURCHASED
---------- -------------------
       101                   5
       102                   2
       103                   3
       104                   1
       105                   4
```

4. **SQL Query**:
   SELECT storeId, COUNT(*) AS totalProducts
   FROM UniMart_Store_Inventory
   GROUP BY storeId;

Relational Algebra:
γ_storeId, COUNT(*) → **totalProducts(UniMart_Store_Inventory)**

Explanation:

- γ is used to count the products in each store.



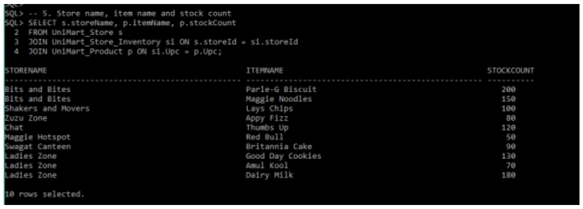
```
SQL> -- 4. Total products available per store
SQL> SELECT storeId, COUNT(*) AS totalProducts
  2  FROM UniMart_Store_Inventory
  3  GROUP BY storeId;

   STOREID TOTALPRODUCTS
---------- -------------
         1             2
         2             1
         3             1
         4             1
         5             1
         6             1
         7             3

7 rows selected.
```

5. **SQL Query**:
   SELECT s.storeName, st.itemName, st.stockCount
   FROM UniMart_Store s
   JOIN UniMart_Store_Inventory si ON s.storeId = si.storeId
   JOIN UniMart_Stock st ON si.Upc = st.Upc;

Relational Algebra:
π_storeName, itemName, **stockCount((UniMart_Store UniMart_Store_Inventory) UniMart_Stock)**

Explanation:

(Join) is used to combine the UniMart_Store, UniMart_Store_Inventory, and UniMart_Stock tables.



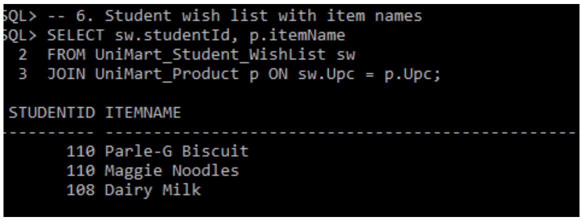
```
SQL> -- 5. Store name, item name and stock count
SQL> SELECT s.storeName, p.itemName, p.stockCount
  2  FROM UniMart_Store s
  3  JOIN UniMart_Store_Inventory si ON s.storeId = si.storeId
  4  JOIN UniMart_Product p ON si.Upc = p.Upc;

STORENAME                                          ITEMNAME                                           STOCKCOUNT
-------------------------------------------------- -------------------------------------------------- ----------
Bits and Bites                                     Parle-G Biscuit                                           200
Bits and Bites                                     Maggie Noodles                                            150
Shakers and Movers                                 Lays Chips                                                100
Zuzu Zone                                          Appy Fizz                                                  80
Chat                                               Thumbs Up                                                 120
Maggie Hotspot                                     Red Bull                                                   50
Swagat Canteen                                     Britannia Cake                                             90
Ladies Zone                                        Good Day Cookies                                          130
Ladies Zone                                        Amul Kool                                                  70
Ladies Zone                                        Dairy Milk                                                180

10 rows selected.
```

6. **SQL Query**:
   SELECT sw.studentId, st.itemName
   FROM UniMart_Student_WishList sw
   JOIN UniMart_Stock st ON sw.Upc = st.Upc;

Relational Algebra:
π_studentId, **itemName(UniMart_Student_WishList UniMart_Stock)**

Explanation:

(Join) combines UniMart_Student_WishList and UniMart_Stock based on selects the studentId and itemName.



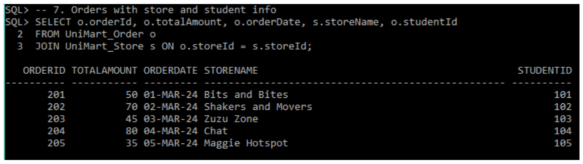
```
SQL> -- 6. Student wish list with item names
SQL> SELECT sw.studentId, p.itemName
  2  FROM UniMart_Student_WishList sw
  3  JOIN UniMart_Product p ON sw.Upc = p.Upc;

 STUDENTID ITEMNAME
---------- --------------------------------------------------
       110 Parle-G Biscuit
       110 Maggie Noodles
       108 Dairy Milk
```

7. **SQL Query**:
   SELECT o.orderId, o.totalAmount, o.orderDate, s.storeName, o.studentId
   FROM UniMart_Order o
   JOIN UniMart_Store s ON o.storeId = s.storeId;

Relational Algebra:
π_orderId, totalAmount, or
storeName, studentId(UniMart_Order UniMart_Store)

Explanation:

(Join) combines UniMart_Order and UniMart_Store based on storeId, and π projects the required columns.



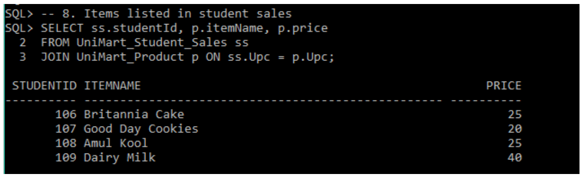
```
SQL> -- 7. Orders with store and student info
SQL> SELECT o.orderId, o.totalAmount, o.orderDate, s.storeName, o.studentId
  2  FROM UniMart_Order o
  3  JOIN UniMart_Store s ON o.storeId = s.storeId;

   ORDERID TOTALAMOUNT ORDERDATE STORENAME                                           STUDENTID
---------- ----------- --------- -------------------------------------------------- ----------
       201          50 01-MAR-24 Bits and Bites                                            101
       202          70 02-MAR-24 Shakers and Movers                                        102
       203          45 03-MAR-24 Zuzu Zone                                                 103
       204          80 04-MAR-24 Chat                                                      104
       205          35 05-MAR-24 Maggie Hotspot                                            105
```

8. **SQL Query**:
   SELECT ss.studentId, st.itemName, st.price
   FROM UniMart_Student_Sales ss
   JOIN UniMart_Stock st ON ss.Upc = st.Upc;

Relational Algebra:
π_studentId, itemName, **price(UniMart_Student_Sales UniMart_Stock)**

Explanation:

(Join) connects UniMart_Student_Sales and UniMart_Stock on Upc, and π selects the studentId, itemName, and price.



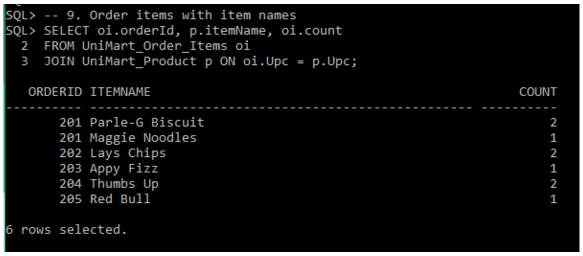
```
SQL> -- 8. Items listed in student sales
SQL> SELECT ss.studentId, p.itemName, p.price
     FROM UniMart_Student_Sales ss
     JOIN UniMart_Product p ON ss.Upc = p.Upc;

 STUDENTID ITEMNAME                                                PRICE
---------- -------------------------------------------------- ----------
       106 Britannia Cake                                             25
       107 Good Day Cookies                                           20
       108 Amul Kool                                                  25
       109 Dairy Milk                                                 40
```

9. **SQL Query**:
SELECT oi.orderId, st.itemName, oi.count
FROM UniMart_Order_Items oi
JOIN UniMart_Stock st ON oi.Upc = st.Upc;

Relational Algebra:
π_orderId, itemName, **count(UniMart_Order_Items UniMart_Stock)**

Explanation:

(Join) connects UniMart_Order_Items and UniMart_Stock, and πselects the orderId, itemName, and count.



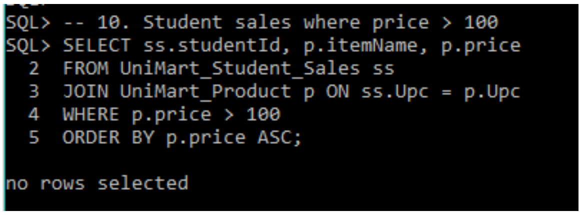
```
SQL> -- 9. Order items with item names
SQL> SELECT oi.orderId, p.itemName, oi.count
     FROM UniMart_Order_Items oi
     JOIN UniMart_Product p ON oi.Upc = p.Upc;

   ORDERID ITEMNAME                                                COUNT
---------- -------------------------------------------------- ----------
       201 Parle-G Biscuit                                             2
       201 Maggie Noodles                                              1
       202 Lays Chips                                                  2
       203 Appy Fizz                                                   1
       204 Thumbs Up                                                   2
       205 Red Bull                                                    1

6 rows selected.
```

10. **SQL Query**:
SELECT ss.studentId, s.itemName, s.price
FROM UniMart_Student_Sales ss
JOIN UniMart_Stock s ON ss.Upc = s.Upc
WHERE s.price > 100
ORDER BY price ASC;

Relational Algebra:
π_studentId, itemName, pri
100(UniMart_Student_Sales UniMart_Stock))

Explanation:

σ (Selection) filters f
greater than 100, and π
required columns.



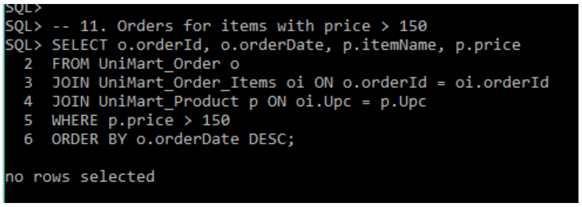
```
SQL> -- 10. Student sales where price > 100
SQL> SELECT ss.studentId, p.itemName, p.price
     FROM UniMart_Student_Sales ss
     JOIN UniMart_Product p ON ss.Upc = p.Upc
     WHERE p.price > 100
     ORDER BY p.price ASC;

no rows selected
```

11. **SQL Query**:
SELECT o.orderId, o.orderDate, s.itemName, s.price
FROM UniMart_Order o
JOIN UniMart_Order_Items oi ON o.orderId = oi.orderId
JOIN UniMart_Stock s ON oi.Upc = s.Upc
WHERE s.price > 150
ORDER BY o.orderDate DESC;

Relational Algebra:
π_orderId, orderDate, item
price(σ_price > 150((UniMa
UniMart_Order_Items) UniMart_Stock))

Explanation:

σ (Selection) filters f
greater than 150, and π
required columns.



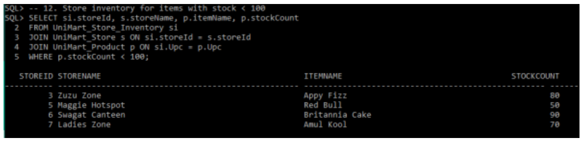
```
SQL>
SQL> -- 11. Orders for items with price > 150
SQL> SELECT o.orderId, o.orderDate, p.itemName, p.price
     FROM UniMart_Order o
     JOIN UniMart_Order_Items oi ON o.orderId = oi.orderId
     JOIN UniMart_Product p ON oi.Upc = p.Upc
     WHERE p.price > 150
     ORDER BY o.orderDate DESC;

no rows selected
```

12. **SQL Query**:
    SELECT si.storeId, s.storeName, st.itemName, st.stockCount
    FROM UniMart_Store_Inventory si
    JOIN UniMart_Store s ON si.storeId = s.storeId
    JOIN UniMart_Stock st ON si.Upc = st.Upc
    WHERE st.stockCount < 100;

Relational Algebra:
π_storeId, storeName, itemName, stockCount(σ_stockCount < **100((UniMart_Store_Inventory UniMart_Store) UniMart_Stock))**

Explanation:

σ filters for stores where the stock count is less than 100.



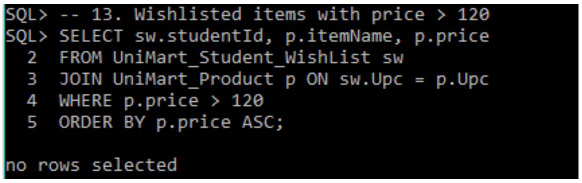
```
SQL> -- 12. Store inventory for items with stock < 100
SQL> SELECT si.storeId, s.storeName, p.itemName, p.stockCount
     FROM UniMart_Store_Inventory si
     JOIN UniMart_Store s ON si.storeId = s.storeId
     JOIN UniMart_Product p ON si.Upc = p.Upc
     WHERE p.stockCount < 100;

 STOREID STORENAME                                          ITEMNAME                                           STOCKCOUNT
-------- -------------------------------------------------- -------------------------------------------------- ----------
       3 Zuzu Zone                                          Appy Fizz                                                  80
       5 Maggie Hotspot                                     Red Bull                                                   50
       6 Swagat Canteen                                     Britannia Cake                                             90
       7 Ladies Zone                                        Amul Kool                                                  70
```

13. **SQL Query**:
    SELECT sw.studentId, st.itemName, st.price
    FROM UniMart_Student_WishList sw
    JOIN UniMart_Stock st ON sw.Upc = st.Upc
    WHERE st.price > 120
    ORDER BY price ASC;

Relational Algebra:
π_studentId, itemName, price(σ_price > **120(UniMart_Student_WishList UniMart_Stock))**

Explanation:

σ filters for wishlist greater than 120, and π required columns.



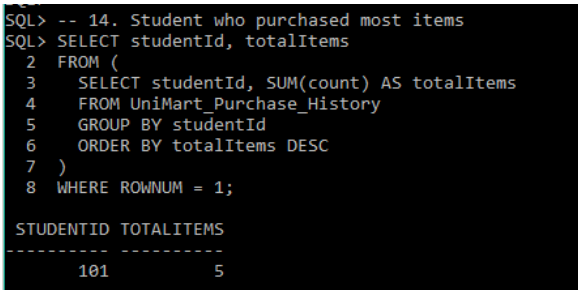
```
SQL> -- 13. Wishlisted items with price > 120
SQL> SELECT sw.studentId, p.itemName, p.price
     FROM UniMart_Student_WishList sw
     JOIN UniMart_Product p ON sw.Upc = p.Upc
     WHERE p.price > 120
     ORDER BY p.price ASC;

no rows selected
```

14. **SQL Query**:
    SELECT studentId, totalItems
    FROM (
    SELECT studentId, SUM(count) AS totalItems
    FROM UniMart_Purchase_History
    GROUP BY studentId
    ORDER BY totalItems DESC
)
    WHERE ROWNUM = 1;

Relational Algebra:
π_studentId, totalItems(σ_ 1(γ_studentId, SUM(count) **totalItems(UniMart_Purchase_History)))**

Explanation:

γ (Aggregation) compute number of items purchased by each student, ordered by the total count in descending order, and π student with the highest count.



```
SQL>
SQL> -- 14. Student who purchased most items
SQL> SELECT studentId, totalItems
     FROM (
       SELECT studentId, SUM(count) AS totalItems
       FROM UniMart_Purchase_History
       GROUP BY studentId
       ORDER BY totalItems DESC
     )
     WHERE ROWNUM = 1;

 STUDENTID TOTALITEMS
---------- ----------
       101          5
```

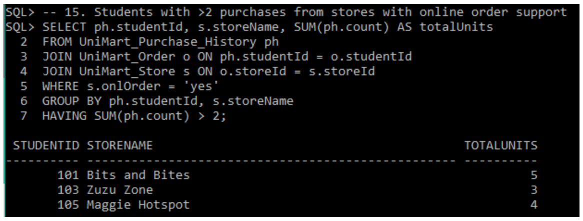
15. **SQL Query**:
    SELECT ph.studentId, s.storeName, SUM(ph.count) AS totalUnits
    FROM UniMart_Purchase_History ph
    JOIN UniMart_Order o ON ph.studentId

= o.studentId
JOIN UniMart_Store s ON o.storeId =
s.storeId
WHERE s.onlOrder = 'yes'
GROUP BY ph.studentId, s.storeName
HAVING SUM(ph.count) > 2;

Relational Algebra:
π_studentId, storeName,
totalUnits(σ_onlOrder = '
storeName, SUM(count) →
totalUnits(UniMart_Purchase_History
UniMart_Order UniMart_Store)))

Explanation:

σ (Selection) filters stores where online
orders are enabled.
γ (Aggregation) groups by studentId and
storeName, then sums the count of
items purchased.



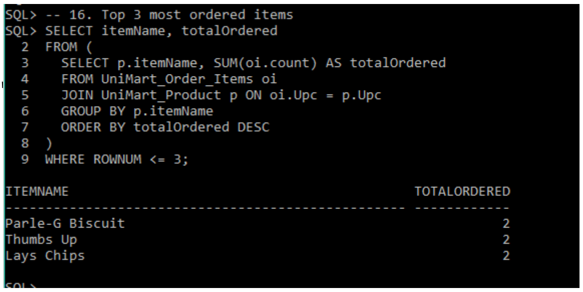
```
SQL> -- 15. Students with >2 purchases from stores with online order support
SQL> SELECT ph.studentId, s.storeName, SUM(ph.count) AS totalUnits
  FROM UniMart_Purchase_History ph
  JOIN UniMart_Order o ON ph.studentId = o.studentId
  JOIN UniMart_Store s ON o.storeId = s.storeId
  WHERE s.onlOrder = 'yes'
  GROUP BY ph.studentId, s.storeName
  HAVING SUM(ph.count) > 2;

 STUDENTID STORENAME                                               TOTALUNITS
---------- -------------------------------------------------- ----------
       101 Bits and Bites                                               5
       103 Zuzu Zone                                                    3
       105 Maggie Hotspot                                               4
```

16. **SQL Query**:
SELECT itemName, totalOrdered
FROM (
SELECT st.itemName, SUM(oi.count) AS
totalOrdered
FROM UniMart_Order_Items oi
JOIN UniMart_Stock st ON oi.Upc =
st.Upc
GROUP BY st.itemName
ORDER BY totalOrdered DESC
)
WHERE ROWNUM <= 3;

Relational Algebra:
π_itemName, totalOrdered(σ_ROWNUM <=
3(γ_itemName, SUM(count) →
totalOrdered(UniMart_Order_Items
UniMart_Stock)))

Explanation:

γ (Aggregation) groups
and sums the count, ordered by the
totalOrdered in descend
selects the top 3 items.



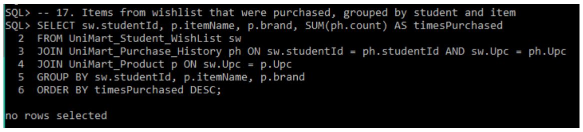
```
SQL> -- 16. Top 3 most ordered items
SQL> SELECT itemName, totalOrdered
  FROM (
    SELECT p.itemName, SUM(oi.count) AS totalOrdered
    FROM UniMart_Order_Items oi
    JOIN UniMart_Product p ON oi.Upc = p.Upc
    GROUP BY p.itemName
    ORDER BY totalOrdered DESC
  )
  WHERE ROWNUM <= 3;

ITEMNAME                                           TOTALORDERED
-------------------------------------------------- ------------
Parle-G Biscuit                                               2
Thumbs Up                                                     2
Lays Chips                                                    2

SQL>
```

17. **SQL Query**:
SELECT sw.studentId, st.itemName,
st.brand, SUM(ph.count) AS
timesPurchased
FROM UniMart_Student_WishList sw
JOIN UniMart_Purchase_History ph
ON sw.studentId = ph.studentId AND
sw.Upc = ph.Upc
JOIN UniMart_Stock st ON sw.Upc =
st.Upc
GROUP BY sw.studentId, st.itemName,
st.brand
ORDER BY timesPurchased DESC;

Relational Algebra:
π_studentId, itemName, bra
timesPurchased(γ_studentId
brand, SUM(count) →
timesPurchased((UniMart_Student_WishList
UniMart_Purchase_History)
UniMart_Stock))

Explanation:

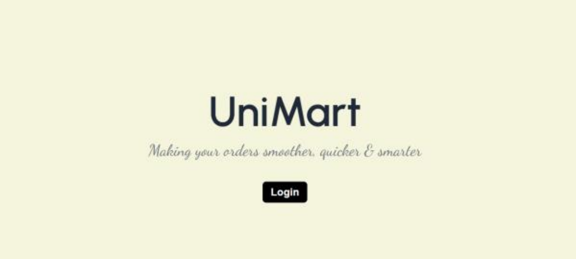
γ (Aggregation) groups
itemName, and brand, then sums the
count of items purchased, ordered by
timesPurchased in descending order.



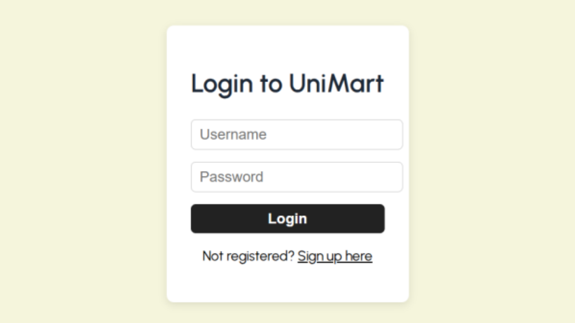
```
SQL> -- 17. Items from wishlist that were purchased, grouped by student and item
SQL> SELECT sw.studentId, p.itemName, p.brand, SUM(ph.count) AS timesPurchased
  2  FROM UniMart_Student_WishList sw
  3  JOIN UniMart_Purchase_History ph ON sw.studentId = ph.studentId AND sw.Upc = ph.Upc
  4  JOIN UniMart_Product p ON sw.Upc = p.Upc
  5  GROUP BY sw.studentId, p.itemName, p.brand
  6  ORDER BY timesPurchased DESC;

no rows selected
```

Progress on developing a user interface to interact with the Data base:

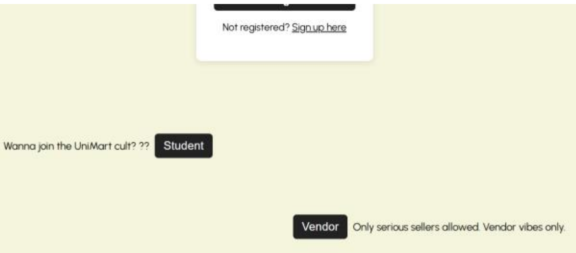
Introduction page of the website:



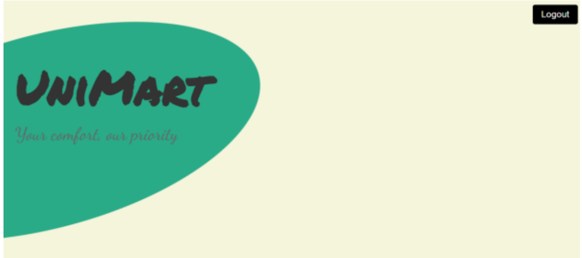
Login page:



Sign-in page:


Not registered? Sign up here
Wanna join the UniMart cult? ??
Student
Vendor
Only serious sellers allowed. Vendor vibes only.

Home page:



To view the website click here