

Abdullahi Mohamed (am3630@live.mdx.ac.uk), Hafsat Musa (hm1141@live.mdx.ac.uk)

Middlesex University, London

Table of Contents

Part A Question 1	Page 2
Part A Question 2	Page 3-5
Part A Question 3	Page 6-11
Part A Question 4	Page 11
Part A Question 5	Page 12-21
Part B Question 1	Page 22
Part B Question 2	Page 23-25
Part B Question 3	Page 25-31
Part B Question 4	Page 31-33
Part B Question 5	Page 33-35
Reflection	Page 36
References	Page 37

Part A – Abdullahi

PART A Question 1: construct a mini-world use case based on the daily operations of the organisation.

The case diagram helps outline the superstores day to day activities, the purpose is to define system requirements. For instance, the superstore owner interacts with all parts of the system, whether it's the customer or inventory management. The analyst of superstore will be responsible for monitoring sales report, whilst the inventory management system will update stock levels and display product availability. Warehouse is to prepare stock and shipping to customers from their ordered products. This use case diagram helps the user understand the requirements needed to operate a superstore.

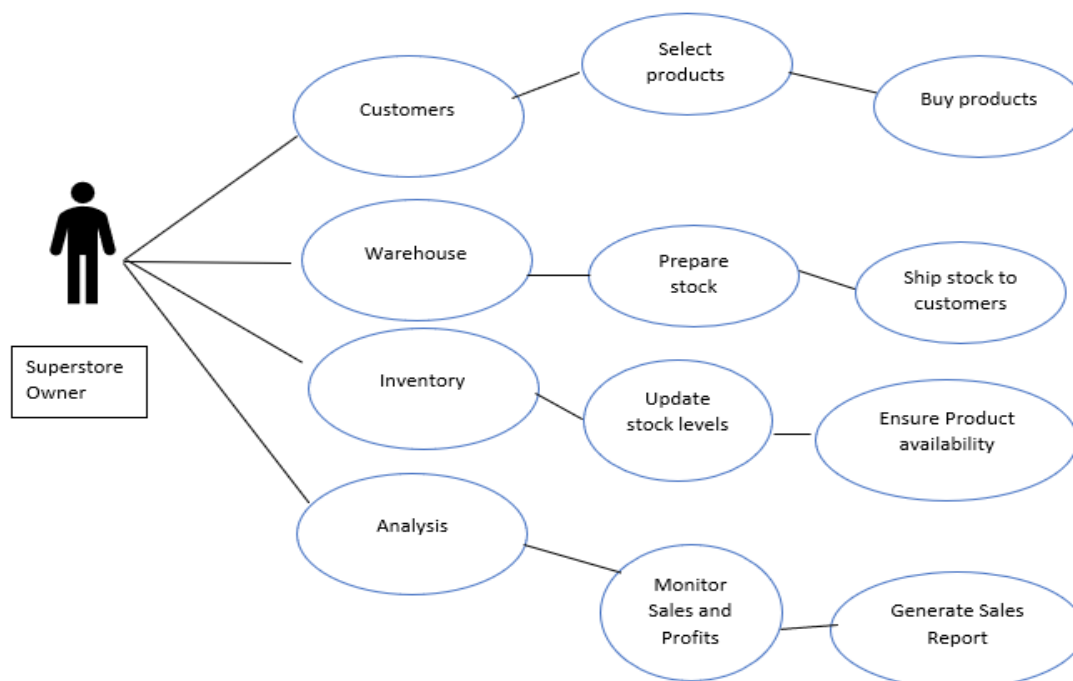


Figure 1 - Use Case Diagram

PART A Question 2. identify key information that needs to be stored within the organisation to support the daily operations, including entities and their relationships (ER modelling).

Entities and Attributes

Key information can be stored and managed using Entity-Relationship (ER) models, it is when relationships are presented among concepts in IT system. The superstore requires data storage for many departments, including customers, products, phone number and addresses. Products have attributes such as price, description with a link to suppliers table. There is a lot of overlapping departments, for example shipping tables will need the order id of customers to display the support of daily operations. This ER model captures all business processes to run a superstore effectively and accurately within operations.

1. **Customers**
 - Attributes:
CustomerID (PK), Name, PhoneNumber, Address, Age
 - Description: Stores customer details for orders, memberships, reviews, and loyalty rewards.
2. **Products**
 - Attributes:
ProductID (PK), Name, Description, Price, Quantity, CategoryID (FK), SupplierID (FK)
 - Description: Captures product details, stock levels, associated categories, and suppliers.
3. **Categories**
 - Attributes:
CategoryID (PK), CategoryName, Description
 - Description: Groups products into logical categories.
4. **Orders**
 - Attributes:
OrderID (PK), CustomerID (FK), OrderDate, TotalAmount
 - Description: Records orders placed by customers.
5. **OrderItems**
 - Attributes:
OrderItemID (PK), OrderID (FK), ProductID (FK), Quantity, Price
 - Description: Tracks individual items within an order.
6. **Suppliers**
 - Attributes:
SupplierID (PK), Name, Address, PhoneNumber
 - Description: Maintains supplier information for product sourcing.
7. **Catalogue**
 - Attributes:
CatalogueID (PK), ProductID (FK), SupplierID (FK), QuantityAvailable
 - Description: Links suppliers to products, including available stock levels.
8. **Employees**
 - Attributes:
EmployeeID (PK), Name, PhoneNumber, Address, JobRole, HireDate, Salary

- Description: Stores employee details for payroll, training, and assignments.
- 9. Payments**
 - Attributes:
 - PaymentID (PK), OrderID (FK), PaymentDate, Amount, PaymentMethod
 - Description: Manages payment details for orders.
- 10. Shipping**
 - Attributes:
 - ShippingID (PK), OrderID (FK), ShippingAddress, ShippingDate, DeliveryDate, ShippingCost
 - Description: Tracks shipping details for each order.
- 11. Reviews**
 - Attributes:
 - ReviewID (PK), ProductID (FK), CustomerID (FK), StarRating, ReviewDate
 - Description: Records customer feedback on products.
- 12. Discounts**
 - Attributes:
 - DiscountID (PK), ProductID (FK), DiscountPercentage, StartDate, EndDate
 - Description: Captures discount offers on products.
- 13. Returns**
 - Attributes:
 - ReturnID (PK), OrderItemID (FK), ReturnDate, Reason, RefundAmount
 - Description: Tracks product returns and refunds.
- 14. Transactions**
 - Attributes:
 - TransactionID (PK), PaymentID (FK), TransactionDate, TransactionStatus
 - Description: Manages transaction logs linked to payments.
- 15. Memberships**
 - Attributes:
 - MembershipID (PK), CustomerID (FK), MembershipType, EnrollmentDate
 - Description: Tracks customer memberships and their details.
- 16. Supplier Contracts**
 - Attributes:
 - ContractID (PK), SupplierID (FK), ContractStartDate, ContractEndDate, ContractDetail
 - Description: Records contracts between suppliers and the organization.
- 17. Customer Loyalty Rewards**
 - Attributes:
 - LoyaltyID (PK), CustomerID (FK), PointsAccumulated, RewardLevel
 - Description: Tracks loyalty rewards earned by customers.
- 18. Warehouse Inventory**
 - Attributes:
 - InventoryID (PK), WarehouseID, ProductID (FK), QuantityInStock
 - Description: Manages inventory levels in the warehouse.
- 19. Employee Training**
 - Attributes:
 - TrainingID (PK), TrainingName, TrainingDate, EmployeeID (FK), Location, Duration
 - Description: Records training sessions attended by employees.

20. Frequently Asked Questions (FAQs)

- Attributes:
FaqlD (PK), Question, Answer, Category, DateAdded
- Description: Stores FAQs for customer reference.

Relationships Between Entities

The superstore's ER model defines relationships, places customer orders, links each to shipping order items. Categorises products and supplier sourced with contracts. These relationships help process orders, manage inventory and find customers queries.

1. **Shipping – Orders (1:1)**
Each order has exactly one shipping record.
2. **Reviews – Products (M:1)**
Each product can have multiple reviews from different customers.
3. **Discounts – Products (M:1)**
Each product can have multiple discounts applied at different times.
4. **Returns – Order Items (1)**
Each return corresponds to multiple order items.
5. **Transactions – Payments (1:1)**
Each transaction leads to exactly one payment.
6. **Memberships – Customers (1:1)**
Each membership is linked to a single customer.
7. **Supplier Contracts – Suppliers (M:1)**
Suppliers can have multiple contracts.
8. **Customer Loyalty Rewards – Customers (1:1)**
Each loyalty reward record corresponds to one customer.
9. **Warehouse Inventory – Products (1:1)**
Each inventory record tracks a single product.
10. **Employee Training – Employees (1:1)**
Each training session is associated with a specific employee.
11. **Customers – Orders (1:M)**
A single customer can place multiple orders.
12. **Orders – Order Items (1:M)**
Each order can contain multiple order items.
13. **Categories – Products (1:M)**
Many products can fall under a single category.
14. **Suppliers – Products (1:M)**

A supplier can supply many products.
15. **Payments – Orders (M:1)**
Multiple payments can be linked to a single order.

PART A Question 3: design a logical model based on the ER modelling, including keys and constraints.

Logical Model 1: Customer Management and Orders

1. Customers

- **Attributes:** CustomerID (PK), OrderID (FK), OrderItem (FK), PhoneNumber, Address, Age
- **Constraints:** CustomerID is unique for each customer.

2. Orders

- **Attributes:** OrderID (PK), ProductID (FK), OrderID (FK), Price, Quantity
- **Constraints:** OrderID is unique; CustomerID is a foreign key referencing Customers.

3. OrderItem

- **Attributes:** OrderItemID (PK), OrderID (FK), ProductID (FK), Quantity, Price
- **Constraints:** OrderID is a foreign key referencing Orders; ProductID is a foreign key referencing Products.

4. Payments

- **Attributes:** PaymentID (PK), OrderID (FK), PaymentDate,
- **Constraints:** OrderID is a foreign key referencing Orders.

5. Shipping

- **Attributes:** ShippingID (PK), OrderID (FK), ShippingAddress,
- **Constraints:** OrderID is a foreign key referencing Orders.

Normalisation to 3NF: Each table has a primary key, and there is no dependency. For instance, OrderID in Payments and Shipping references Orders, ensures the model is normalized.

The model for customer management order is logically designed to efficiently handle customer transactions safely. The customers table stores useful information and connects to the Orders table through the CustomerID foreign key, allowing each customer to place multiple orders. The payment table maintains transaction details referencing OrderID to ensure financial safety, Shipping table links to Orders table, to ensure a smooth delivery process.

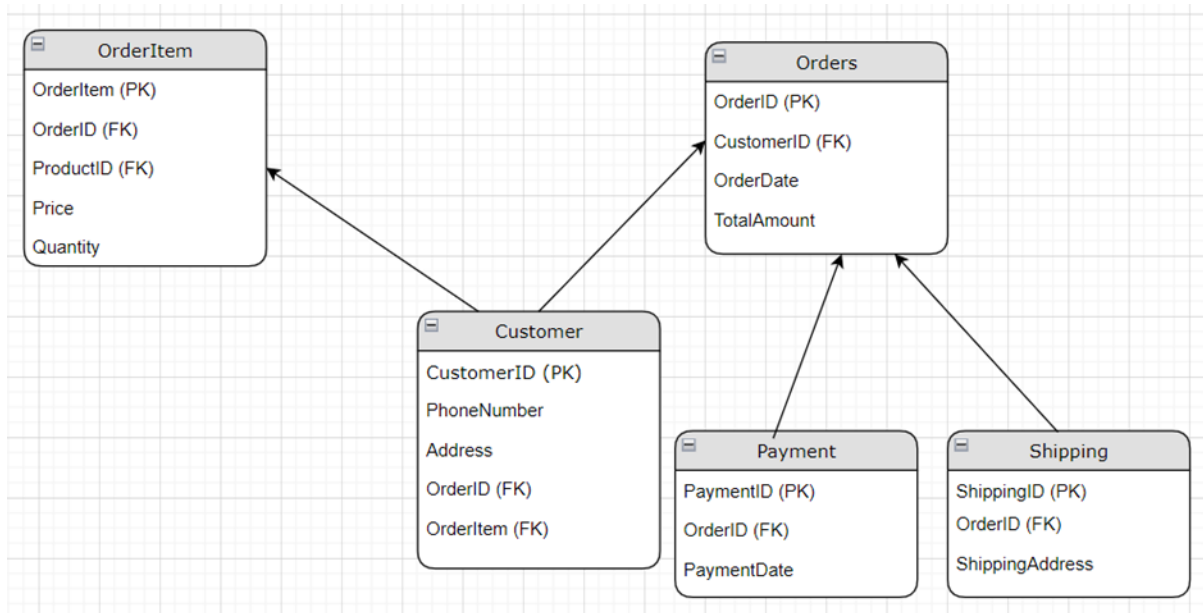


Figure 2 - Logical Model 1

Logical Model 2: Product and Supplier Management

1. Products

- **Attributes:** ProductID (PK), SupplierID (FK), CategoryID (FK), ProductName, Quantity
- **Constraints:** CategoryID is a foreign key referencing Category.

2. Category

- **Attributes:** CategoryID (PK), CategoryName, Description
- **Constraints:** CategoryID is unique for each category.

3. Supplier

- **Attributes:** SupplierID (PK), Name, Address, PhoneNumber, ProductID(FK)
- **Constraints:** SupplierID is unique for each supplier.

4. Catalogue

- **Attributes:** CatalogueID (PK), ProductID (FK), SupplierID (FK), QuantityAvailable
- **Constraints:** ProductID is a foreign key referencing Products; SupplierID is a foreign key referencing Supplier.

5. Supplier Contracts

- **Attributes:** ContractID (PK), SupplierID (FK), ContractDetail
- **Constraints:** SupplierID is a foreign key referencing Supplier.

Normalization to 3NF: Each table has a unique primary key, and all attributes are directly dependent on the primary key. No transitive or partial dependencies are present.

The product supplier management model tracks product inventory, supplier relationships and product categorisation. The catalogue tables connect the supplier table with the products, this helps to manage stock. Supplier contracts ensure documentation is secure using the supplier table to connect.

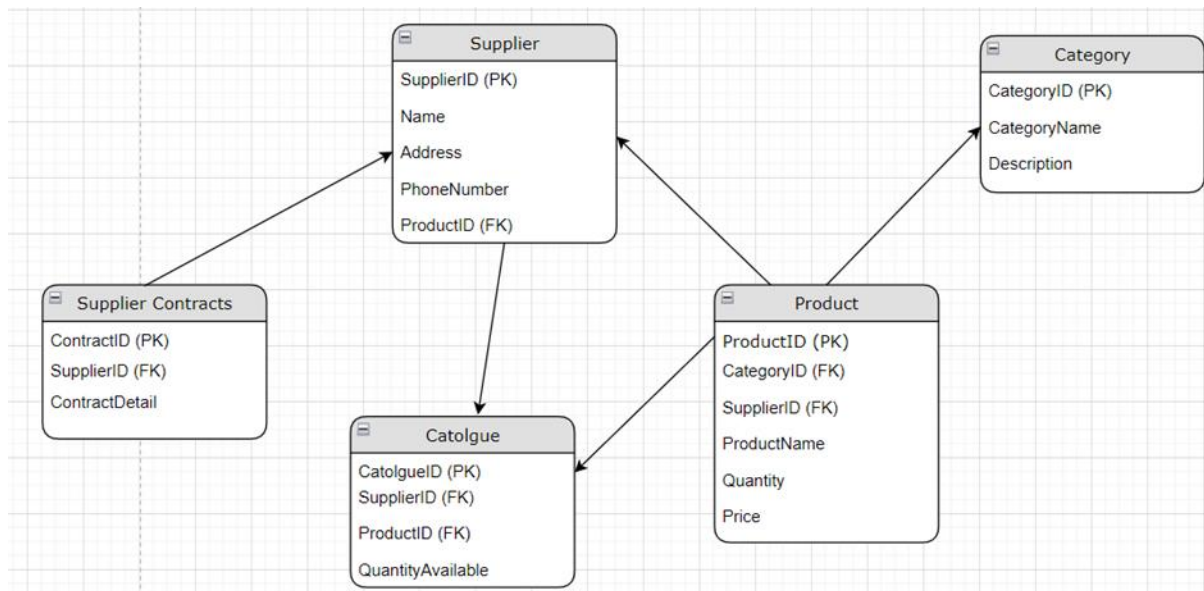


Figure 3 - Logical Model 2

Logical Model 3: Customer Relationship and Loyalty

1. Membership

- **Attributes:** MembershipID (PK), CustomerID (FK), MembershipType,
- **Constraints:** CustomerID is a foreign key referencing Customers.

2. Customer Loyalty Rewards

- **Attributes:** LoyaltyID (PK), MembershipID (FK), PointsAccumulated, RewardLevel
- **Constraints:** CustomerID is a foreign key referencing Customers.

3. Reviews

- **Attributes:** ReviewID (PK), ReturnID (FK), FaqID (FK), StarRating, ReviewDate
- **Constraints:** CustomerID is a foreign key referencing Customers; ProductID is a foreign key referencing Products.

4. Returns

- **Attributes:** ReturnID (PK), OrderItemID (FK), Reason
- **Constraints:** OrderItemID is a foreign key referencing OrderItem.

5. FAQ

- **Attributes:** FaqID (PK), Answer, DateAdded
- **Constraints:** Primary key is FaqID.

Normalization to 3NF: All tables in this logical model meet the 3NF requirement, as each attribute is directly related to the primary key without transitive or partial dependencies.

The customer relationship loyalty model is designed to manage customer engagement, feedback and reward them effectively. The membership table stores loyalty points accumulated by each customer; the review table displays customer feedback on products. Lastly, FAQ displays common inquiries from customers and the date added to show how recent the claim is.

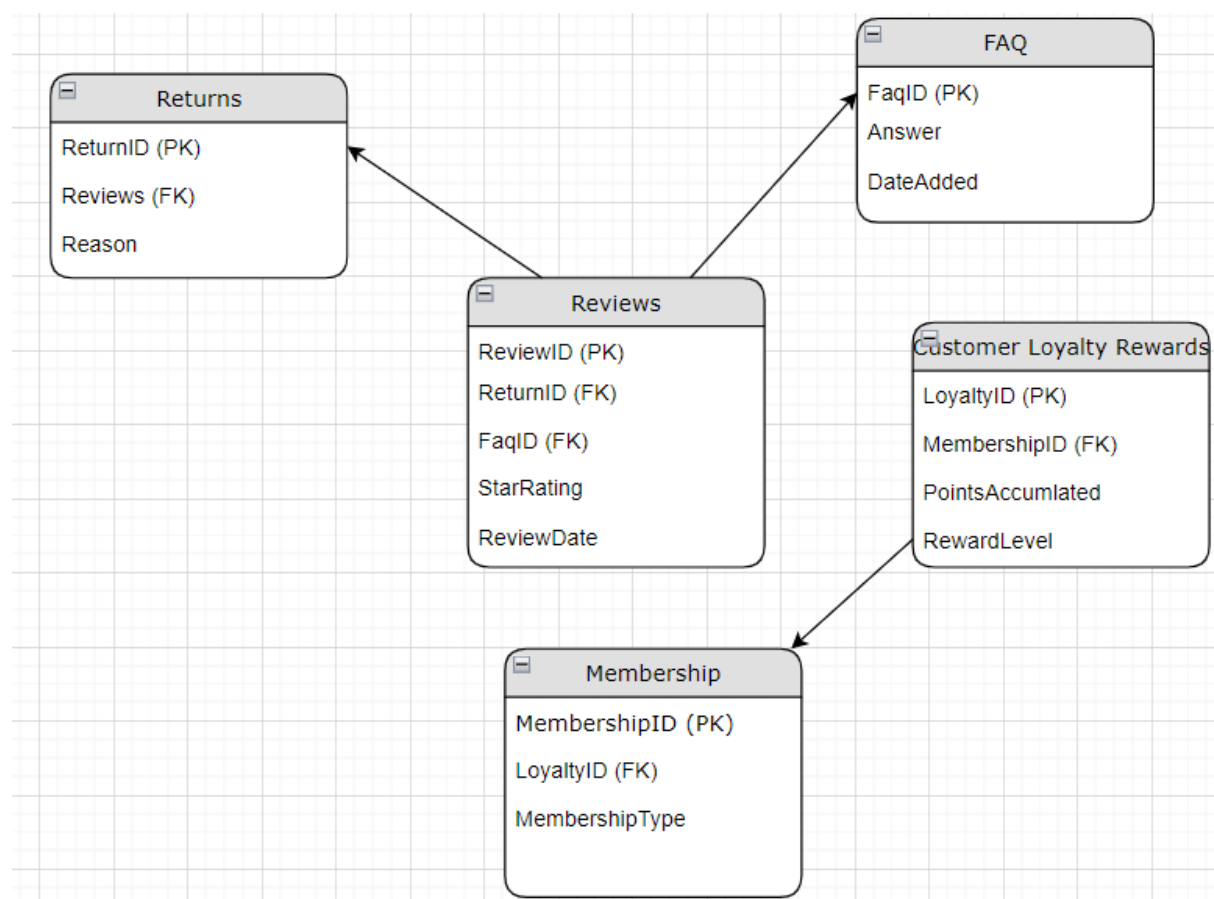


Figure 4 - Logical Model 3

Logical Model 4: Inventory and Employee Management

1. Warehouse Inventory

- **Attributes:** InventoryID (PK), EmployeesID (FK), DiscountID (FK), TransactionDate
- **Constraints:** EmployeesID is a foreign key referencing Employees.

2. Employees

- **Attributes:** EmployeeID (PK), Name, PhoneNumber, JobRole, HireDate
- **Constraints:** Each employee has a unique EmployeeID.

3. Employee Training

- **Attributes:** TrainingID (PK), TrainingName, TrainingDate,
- **Constraints:** Each trainer has a unique TrainingID

4. Discount

- **Attributes:** DiscountID (PK), Inventory (FK), DiscountPercentage, EndDate
- **Constraints:** Inventory is a foreign key referencing Warehouse Inventory.

5. Transactions

- **Attributes:** TransactionID (PK), EmployeesID (FK), TransactionDate, Discount (FK)
- **Constraints:** EmployeesID is a foreign key referencing the Employee Table.

Normalization to 3NF: All tables comply with 3NF requirements as each attribute directly depends on the primary key without partial or transitive dependencies.

The inventory with employee management is rather complex as it manages stock, employee data and discount added from transactions. Warehouse inventory manages stock and records transactions. The employee table tracks job roles, training data, then discount table is applied to inventory items.

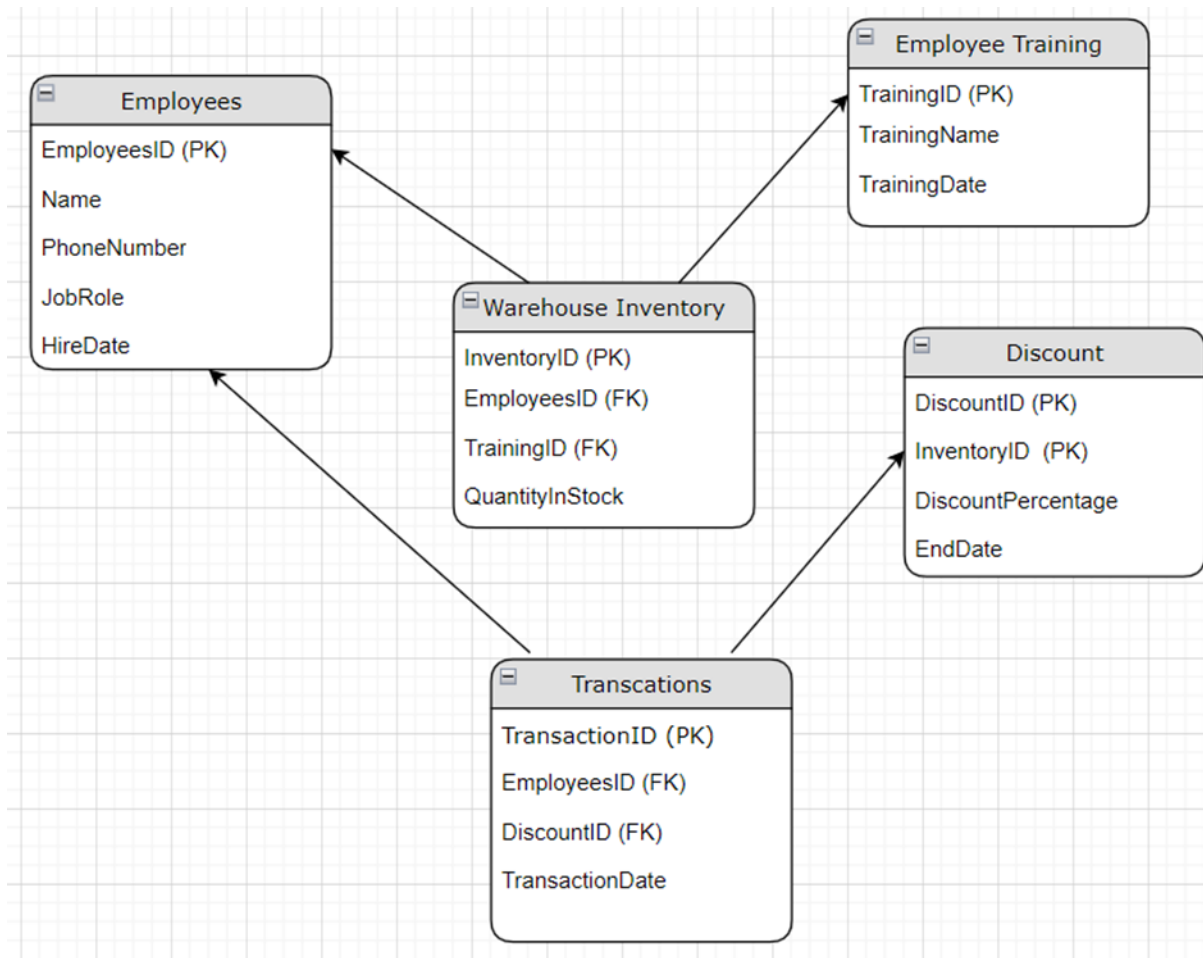


Figure 5 - Logical Model 4

PART A Question 4. make sure the design is normalised to 3NF. 10 marks

Yes, see question 3.

PART A Question 5: implement the database in MS Access or similar SQL-related tool using SQL table creation and modification queries. 10 marks

The SQL queries display 20 interconnected successful tables, forming structured relational database for superstores. The aim is design and manage products, customers, orders, suppliers, inventory, ensuring all are stored and retrieved. Each table had a Primary Key (PK) meaning that is uniquely identifiable, and some even having a Foreign Key, this is when a relationship is established between tables with a common identifier. By clear definition of relationships, the model allows for operations across different departments, from ordering to training new employees and controlling customer engagement, these tables cover all aspects of superstore.

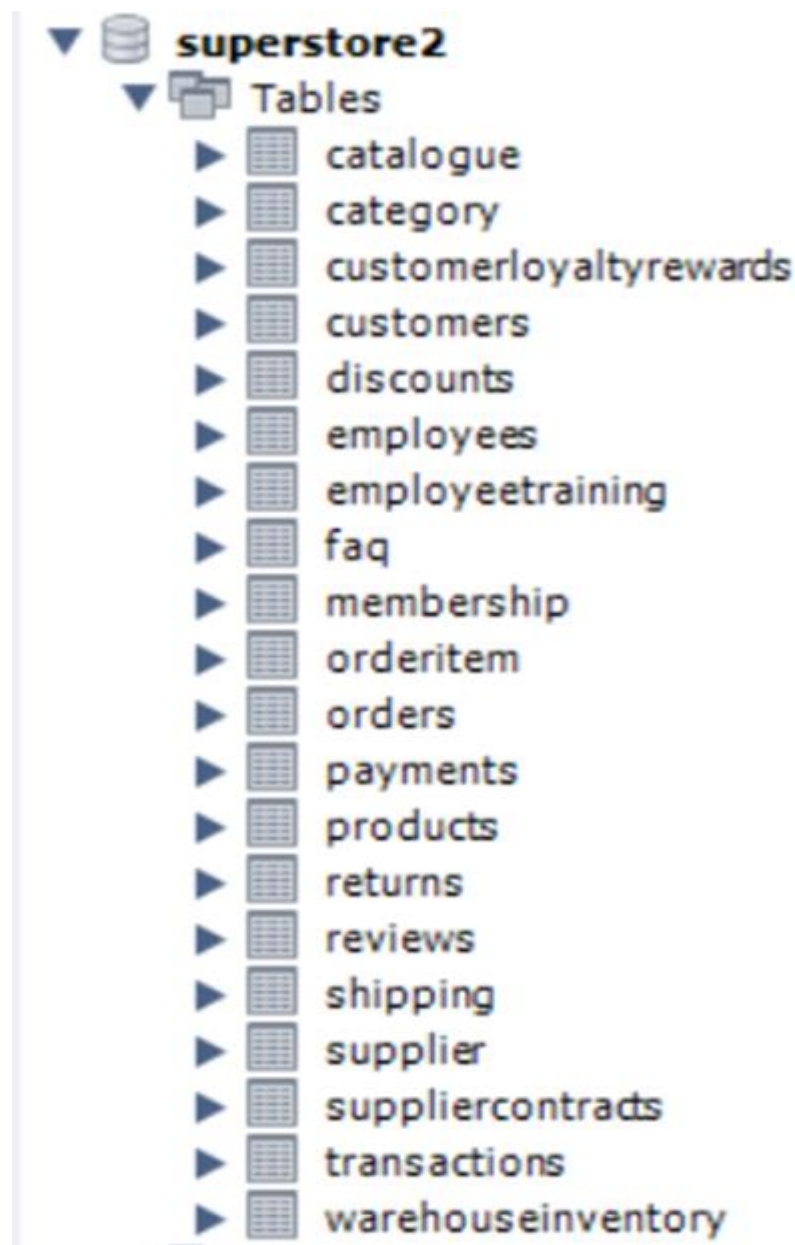


Figure 6 - SQL Tables

SQL QUERIES –

-- Table 1: Customers

```
CREATE TABLE Customers (  
    CustomerID INT PRIMARY KEY,  
    PhoneNumber VARCHAR(15),  
    Address VARCHAR(25),  
    Age INT  
);
```

-- Table 2: Category

```
CREATE TABLE Category (  
    CategoryID INT PRIMARY KEY,  
    CategoryName VARCHAR(20),  
    Description TEXT  
);
```

-- Table 3: Supplier

```
CREATE TABLE Supplier (  
    SupplierID INT PRIMARY KEY,  
    Name VARCHAR(50),  
    Address VARCHAR(25),  
    PhoneNumber VARCHAR(15)  
);
```

-- Table 4: Products

```
CREATE TABLE Products (  
    ProductID INT PRIMARY KEY,  
    SupplierID INT,
```

```
CategoryID INT,  
ProductName VARCHAR(20),  
Quantity INT,  
FOREIGN KEY (SupplierID) REFERENCES Supplier(SupplierID),  
FOREIGN KEY (CategoryID) REFERENCES Category(CategoryID));
```

-- Table 5: Orders

```
CREATE TABLE Orders (  
    OrderID INT PRIMARY KEY,  
    CustomerID INT,  
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID));
```

-- Table 6: OrderItem

```
CREATE TABLE OrderItem (  
    OrderItemID INT PRIMARY KEY,  
    OrderID INT,  
    ProductID INT,  
    Quantity INT,  
    Price DECIMAL(10, 2),  
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),  
    FOREIGN KEY (ProductID) REFERENCES Products(ProductID));
```

-- Table 7: Payments

```
CREATE TABLE Payments (  
    PaymentID INT PRIMARY KEY,  
    OrderID INT,  
    PaymentDate DATE,  
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID));
```

-- Table 8: Shipping

```
CREATE TABLE Shipping (  
    ShippingID INT PRIMARY KEY,  
    OrderID INT,  
    ShippingAddress VARCHAR(255),  
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID));
```

-- Table 9: Catalogue

```
CREATE TABLE Catalogue (  
    CatalogueID INT PRIMARY KEY,  
    ProductID INT,  
    SupplierID INT,  
    QuantityAvailable INT,  
    FOREIGN KEY (ProductID) REFERENCES Products(ProductID),  
    FOREIGN KEY (SupplierID) REFERENCES Supplier(SupplierID)  
);
```

-- Table 10: Supplier Contracts

```
CREATE TABLE SupplierContracts (  
    ContractID INT PRIMARY KEY,  
    SupplierID INT,  
    ContractDetail TEXT,  
    FOREIGN KEY (SupplierID) REFERENCES Supplier(SupplierID)  
);
```

-- Table 11: Membership

```
CREATE TABLE Membership (  
    MembershipID INT PRIMARY KEY,  
    CustomerID INT,  
    MembershipType VARCHAR(50),
```

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID));

-- Table 12: Customer Loyalty Rewards

```
CREATE TABLE CustomerLoyaltyRewards (  
    LoyaltyID INT PRIMARY KEY,  
    MembershipID INT,  
    PointsAccumulated INT,  
    RewardLevel VARCHAR(50),  
    FOREIGN KEY (MembershipID) REFERENCES Membership(MembershipID)  
);
```

-- Table 13: Reviews

```
CREATE TABLE Reviews (  
    ReviewID INT PRIMARY KEY,  
    ProductID INT,  
    CustomerID INT,  
    StarRating INT,  
    ReviewDate DATE,  
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),  
    FOREIGN KEY (ProductID) REFERENCES Products(ProductID));
```

-- Table 14: Returns

```
CREATE TABLE Returns (  
    ReturnID INT PRIMARY KEY,  
    OrderItemID INT,  
    Reason VARCHAR(255),  
    FOREIGN KEY (OrderItemID) REFERENCES OrderItem(OrderItemID));
```


-- Table 15: FAQ

```
CREATE TABLE FAQ (  
    FaqID INT PRIMARY KEY,  
    Question TEXT,  
    Answer TEXT,  
    DateAdded DATE);
```

-- Table 16: Employees

```
CREATE TABLE Employees (  
    EmployeeID INT PRIMARY KEY,  
    Name VARCHAR(100),  
    PhoneNumber VARCHAR(15),  
    JobRole VARCHAR(100),  
    HireDate DATE);
```

-- Table 17: Employee Training

```
CREATE TABLE EmployeeTraining (  
    TrainingID INT PRIMARY KEY,  
    TrainingName VARCHAR(100),  
    TrainingDate DATE,  
    EmployeeID INT,  
    FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID));
```

-- Table 18: Warehouse Inventory

```
CREATE TABLE WarehouseInventory (  
    InventoryID INT PRIMARY KEY,  
    EmployeeID INT,  
    DiscountID INT,  
    TransactionDate DATE,
```

```

        FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID)
    );

-- Table 19: Discounts

CREATE TABLE Discounts (
    DiscountID INT PRIMARY KEY,
    InventoryID INT,
    DiscountPercentage DECIMAL(5, 2),
    EndDate DATE,
    FOREIGN KEY (InventoryID) REFERENCES WarehouseInventory(InventoryID)
);

```

-- Table 20: Transactions

```

CREATE TABLE Transactions (
    TransactionID INT PRIMARY KEY,
    EmployeeID INT,
    TransactionDate DATE,
    DiscountID INT,
    FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID),
    FOREIGN KEY (DiscountID) REFERENCES Discounts(DiscountID)
);

```

Modification Queries

The following SQL modification queries display the insertion of sample data in key tables, this is to ensure the database is functional and ready for use. Values have been inserted into Customer, Category, Supplier, Products and Catalogue tables to help users understand what type of data will be stored in there. It also helps test relationships between entities, query execution and highlight the database is prepared for further operation.

Customer table

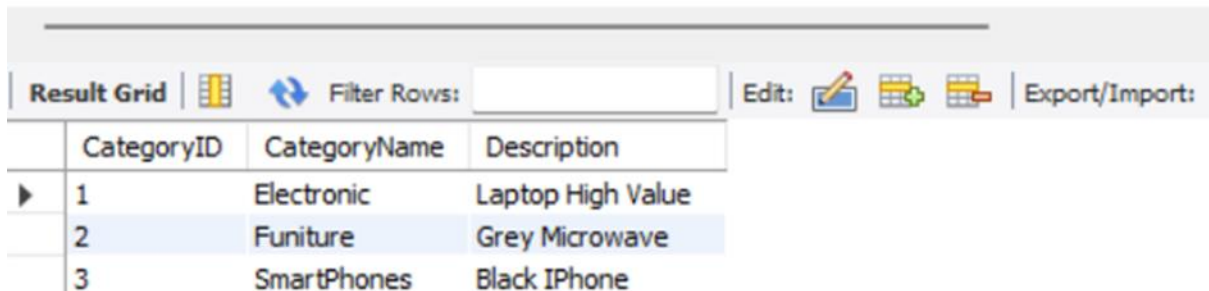
```
1 • INSERT INTO customers (CustomerID,PhoneNumber,Address,Age)
2   VALUES ('3', '111-111-111', 'Meal 10', 87);
3
4
5 • select * from customers;
```

CustomerID	PhoneNumber	Address	Age
1	444-444-444	Logan 21	34
2	333-333-333	Till 40	56
3	111-111-111	Meal 10	87

Figure 7- SQL Customer Query 1

Category table

```
1 • INSERT INTO category (CategoryID,CategoryName, Description)
2   VALUES ('3', 'SmartPhones', 'Black iPhone');
3
4 • select * from category;
```

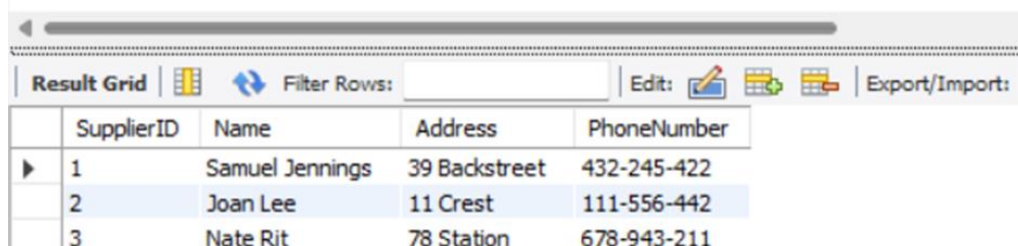


	CategoryID	CategoryName	Description
▶	1	Electronic	Laptop High Value
	2	Furniture	Grey Microwave
	3	SmartPhones	Black iPhone

Figure 8 - SQL Category Query

Supplier Table

```
1 • INSERT INTO supplier (SupplierID,Name, Address,PhoneNumber)
2   VALUES ('3','Nate Rit', '78 Station', '678-943-211');
3
4 • select * from supplier;
```

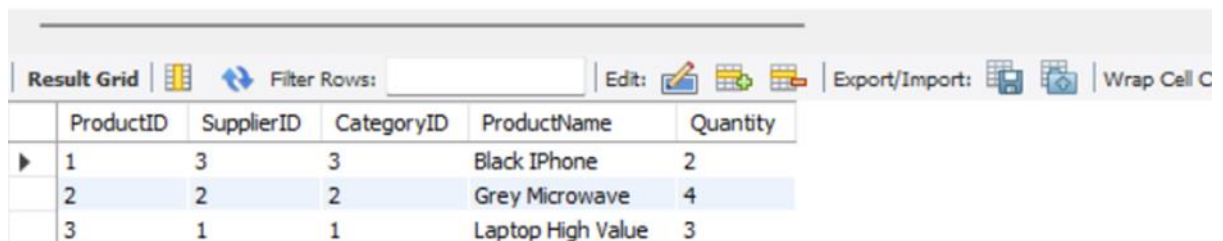


	SupplierID	Name	Address	PhoneNumber
▶	1	Samuel Jennings	39 Backstreet	432-245-422
	2	Joan Lee	11 Crest	111-556-442
	3	Nate Rit	78 Station	678-943-211

Figure 9 - SQL Supplier Query

Products Table

```
1 • INSERT INTO products(ProductID,SupplierID,CategoryID,ProductName, Quantity)
2   VALUES ('3','1','1', 'Laptop High Value', '3');
3
4 • select * from products;
```

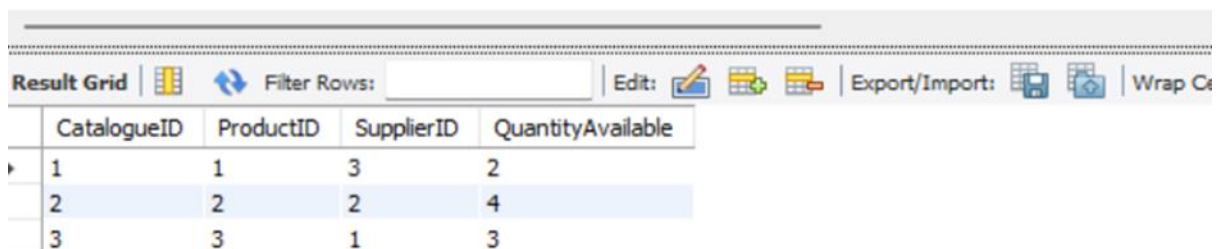


	ProductID	SupplierID	CategoryID	ProductName	Quantity
▶	1	3	3	Black iPhone	2
	2	2	2	Grey Microwave	4
	3	1	1	Laptop High Value	3

Figure 10 - SQL Products Query

Catalogue Table

```
1 • INSERT INTO catalogue(CatalogueID,ProductID,SupplierID,QuantityAvailable)
2   VALUES ('3','3','1', '3');
3
4 • select * from catalogue;
```



	CatalogueID	ProductID	SupplierID	QuantityAvailable
▶	1	1	3	2
	2	2	2	4
	3	3	1	3

Figure 11 - SQL Catalogue Query

Part B - HAFSAT

PART B Question 1. identify possible analytical tasks that can be carried out on historical data of the organisation to improve the business performance.

There are multiple analytical tasks that can be performed on historical data to improve the business performance of the superstore. The following are key analytical challenges and their potential value:

1. Customer Purchase Pattern Analysis

- a. **Objective:** building focused marketing strategies and enhancing sales predictions.
- b. **Approach:** Explore sales fact data, with dimensions like product category, purchase frequency & demographics.
- c. **Example:** Isolating high-spending customer groups and personalizing promotions for their preferred product range.

2. Inventory Turnover Rate Optimization

- a. **Objective:** Manage inventory levels and holding costs.
- b. **Approach:** Use data from the Warehouse inventory Table associated with sales data to monitor stock movement
- c. **Example:** Find and Analyse slow-moving items, Re-Tail for bottlenecks

3. Sales Performance and Trend Analysis

- a. **Objective:** determine best-selling products and seasonal sales trends.
- b. **Approach:** Utilize the Time Dimension to visualize quarterly and annual sales trends.
- c. **Example:** Noticing that electronics sales seem to spike in Q4 and staging inventory and marketing strategies to align with this.

4. Customer Loyalty and Retention Strategies

- a. **Objective:** Retain more customers by offering them specific rewards programs.
- b. **Approach:** Analyse purchase frequency trends + total accumulated points using the Customer Loyalty Rewards Table
- c. **Example:** Offering special discounts to VIP or loyal customers.

PART B Question 2. design a data warehouse schema that supports the analytical tasks discussed above. 10 marks

Star Schema Design

The purpose of the **star schema** is solely to support analytical tasks through efficiently organizing data in a centralized manner.

Table: Sales Data Warehouse Schema. Source: Microsoft, (2024).

Table Name	Description	Key Attributes
Fact Table: Sales Fact Table	Stores numerical metrics for sales transactions	Total sales amount, Quantity sold, Discount applied
Customer Dimension	Contains customer-related data	CustomerID, Name, Location, Age, Membership Level
Product Dimension	Stores product-related attributes	<u>ProductID</u> , Name, Category, Supplier, Price
Time Dimension	Maintains time-based attributes	<u>DateID</u> , Year, Quarter, Month, Day
Geography Dimension	Stores geographic information	<u>RegionID</u> , Country, City, Store
Inventory Dimension	Manages stock levels and warehouse locations	<u>InventoryID</u> , Stock Level, Warehouse Location

Example Table Design

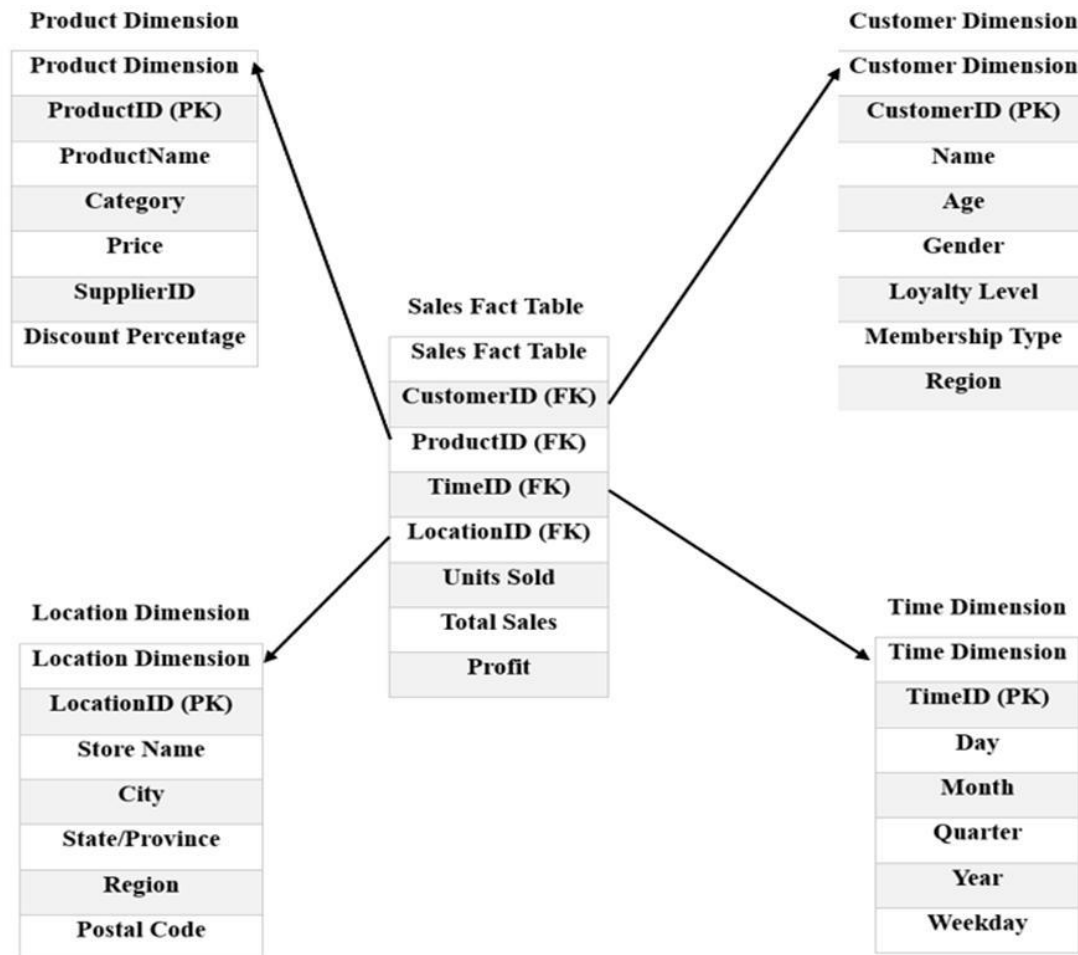


Figure 12 -Star Schema

Key Features of the Design

Centralized Analysis

Star schema usually supports Online Analytical Processing (OLAP) operations like drill down, drill up, slice and dice making data analysis easier (Kimball & Ross, 2013).

Efficient Querying

The star schema helps with reporting and querying by separating descriptive data in dimension tables, while fact tables contain numerical measurements, furthering query optimization (Inmon, 2005).

Scalability

The schema is very scalable, as new dimensions and/or metrics can be added with little change to the structure (Golf Arelli & Rizzi, 2009).

Why Use a Star Schema?

The star schema reduces the complexity of a query operation by pre-aggregating data in its associated fact table, thus improving the space of its analytical performance (Kimball &

Ross, 2013). As well as that, its denormalized structure effectively enhances query performance while easing the computational burden on databases. It also creates a clear relationship between the dimensions and data surfaces an intuitive way for analysts to derive insights and trends.

PART B Question 3. Illustrate with examples how different OLAP operations can be applied to support the data analytics tasks discussed above. 10 marks

1. Slice

- **Definition:** base on one dimension, a subset can be extracted.

Example: Analysis of total sales category for the "Electronics

Sql Query:

```
SELECT ProductCategory,  
TotalSales,  
UnitsSold  
FROM  
SalesFactTable  
WHERE ProductCategory = 'Electronics';
```

ProductCategory	TotalSales	UnitsSold
Electronics	50000	50

Figure 13 - SQL Query

Purpose: At every time and location, analyse and understand the way electronics function.

Illustration: Data Cube: The "slice" in both time and space would focus on the layers of the Electronics

2. Dice

Definition: Data filtering accordingly based on several dimensions.

- **Example:** In both the "East" and "West" regions, Analysis of the sales for "Electronics" and "Furniture"

Sql Query:

```
SELECT Region,  
ProductCategory,  
TotalSales,  
Profit  
FROM
```

SalesFactTable

WHERE Region IN ('East', 'West') AND ProductCategory IN ('Electronics', 'Furniture');

Region	ProductCategory	TotalSales	Profit
East	Electronics	50000	10000
West	Furniture	40000	8000

Figure 14 - SQL Query 2

Purpose: Analyse the sales metrics in terms of the product specific and selection of regions.

Illustration: Reduce the data cube, take electronics and furniture, and focus on East and West regions.

3. Drill-Down

- **Definition:** By moving from the summarize to the detailed views, data granularity increases

Example: Both the moth and yearly sales can be viewed

Sql Query:

“SELECT Region, ProductCategory, TotalSales, Profit

FROM SalesFactTable

WHERE Region IN ('East', 'West') AND ProductCategory IN ('Electronics', 'Furniture');”

Month	TotalSales
January	80000
February	70000
March	60000
April	40000

Figure 15 - SQL Query 3

Purpose: the purpose is to analyse the differences in the variations of monthly sales over a one-year period.

illustration:

Start with the annual sales data for 2024 and then expand the hierarchy to include monthly totals.

4. Pivot

- **Definition:** To understand the alternative perspectives, it allows for data Reorients
- Example: The sales performance data is compared not by the products but by region

Sql Query:

```
SELECT Location, TotalSales
FROM SalesFactTable
GROUP BY Location
```

Location	TotalSales
East	50000
West	40000
North	30000
South	20000

Figure 16 - SQL Query 4 Example

Purpose: Compare and contrast the sales performance of disparate locations

Illustration: Data Cube: Rearrange the axis, so that sales are view from a location perspective instead of a product perspective.

Key Points to Highlight

1. **Efficiency:** These operations make complex queries more intuitive due to the ability for rapid, interactive analysis that OLAP provides.
2. **Flexibility:** When analysts look at customer demographics or regional sales trends, they can adjust their lens to certain business needs.
3. **Practical Application:** The superstore can drive strategic decisions by leveraging the operations to detect trends in sales data they may not be aware of.

1. Slice Operation (Total Sales for Electronics)

- **Description:**
In terms of data reduction, a slice operation targets certain data by selecting a single dimension from a multi-dimensional OLAP (Online Analytical Processing) cube.
- **Example:**
selecting the total sales in each region for the Electronics category.

Visualization

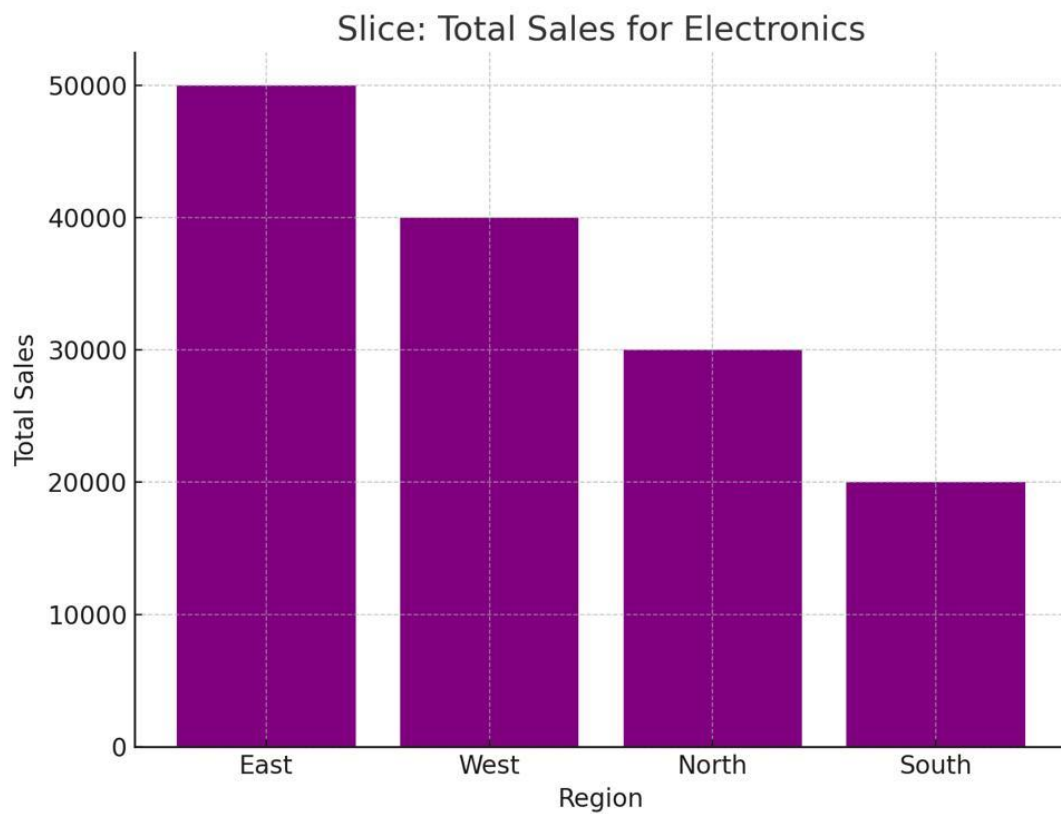


Figure 17 - Slice Operations for Electronics

2. Dice Operation (Total Sales for Electronics & Furniture in East & West Regions)

- **Description:**

Dice is a data selective process based on one or more dimension, and it uses static data, which is operated on OLAP cube.

- **Example:**

comparing sales of category furniture and electronics between East and West regions.

Visualization

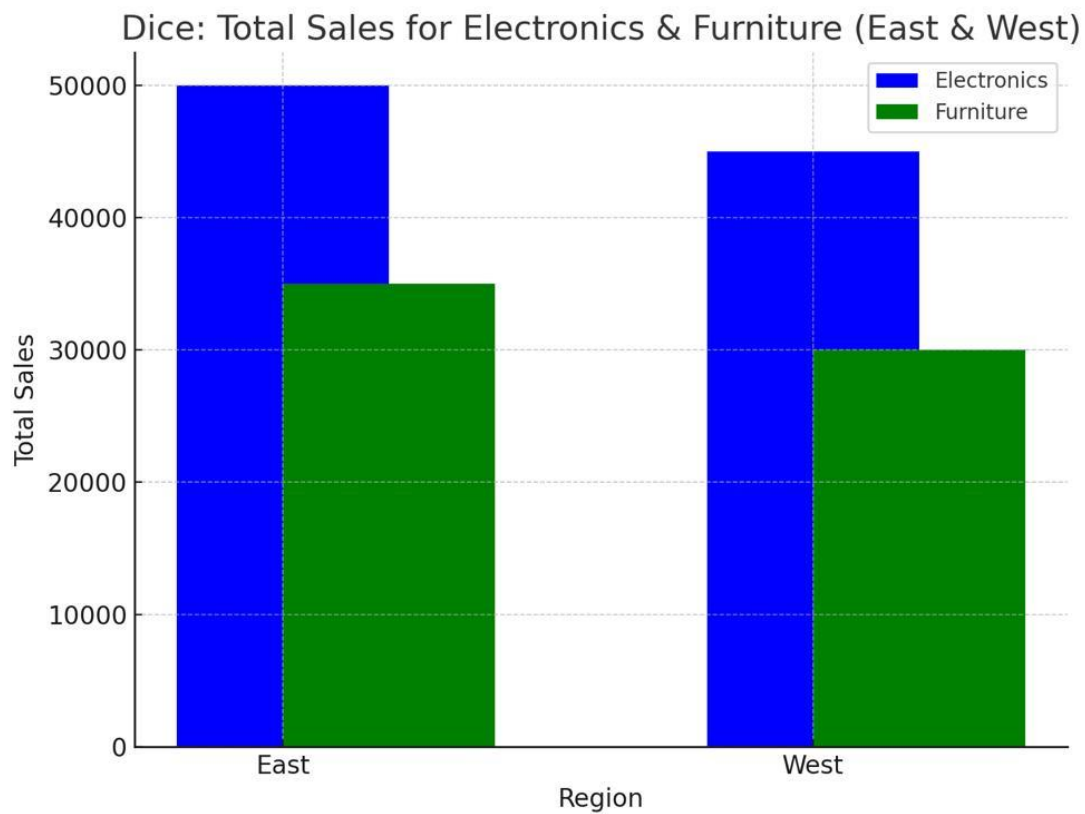


Figure 18 Dice Operations for Regions

3. Drill-Down Operation (Quarterly Sales Trends)

- **Description:**

Drill-down operation: they are used to divide the aggregated information into a more fine-grained data.

- **Example:**

For example, drilling down from sales on an annual basis to sales trends on a quarterly basis over multiple years.

Visualization

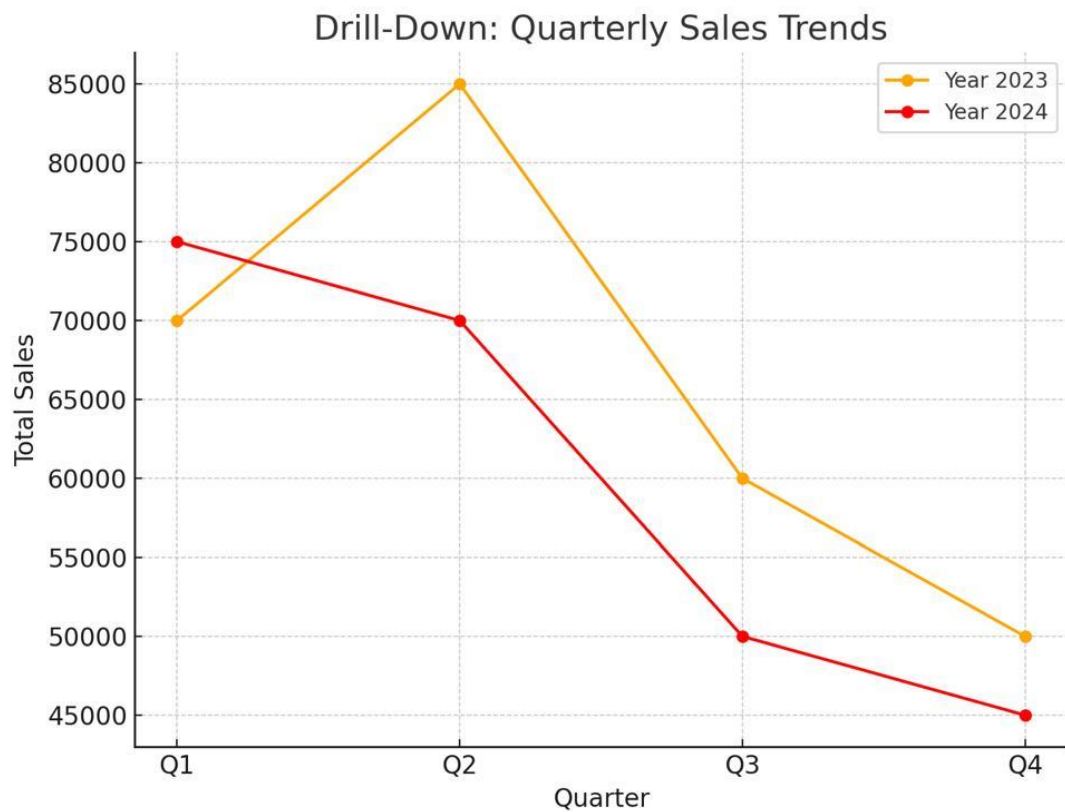


Figure 19 - Drill Down Quarterly Sales

4. Pivot Operation (Total Sales by Region and Product Category)

- **Description:**
Pivoting aligns rows and columns such that data is rearranged, enabling different perspective examinations.
- **Example:**
Total sales by product category and region is one way to look for trends.

Visualization

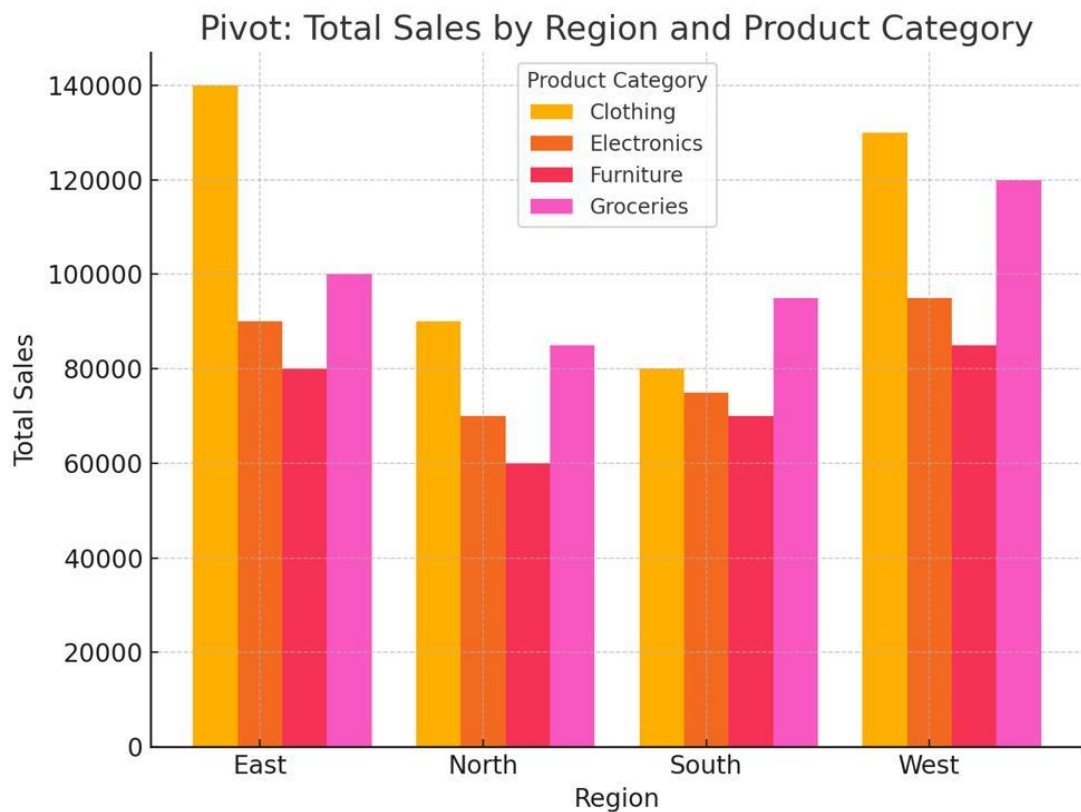


Figure 20 - Pivot Operations by Region

PART B Question 4. discuss possible challenges in the ELT process.

1. Data Extraction Complexity Due to Multiple Sources

Issue: Getting data from varied segments (CRM, ERP, sales databases) spread with unstructured and multiple formats.

Examples from the Superstore: Pulling in product details from vendor databases; pulling in stock counts from warehouse systems; pulling in purchase data from transactional systems.

Complications: Heterogeneity: Several systems use different data models and schemas.

Performance: High-frequency extraction can put a strain on the source systems, potentially impacting their performance.

Solution: Use middleware tools like Oracle Warehouse Builder to standardize and automate data extraction process.

2. Transformation Issues Like Cleaning and Deduplication

Issue: Data quality is affected by duplication, missing values, and inconsistencies.

Examples from the Superstore: Cleaning up Client Data Maintaining uniform forms for names, addresses, and loyalty levels. At places where clients can submit similar orders multiple times by mistake, deduplicating orders.

Complications:

Data Quality: Many sources provide data of poor quality, which can lead to wrong conclusions.

Moving from pipelined transformations to multistage transformations can introduce complexity.

Solution: Enriching data without integrity can lead to further issues in later stages of the pipeline; ensure data integrity.

3. Loading Challenges: Handling Large Volumes of Data Efficiently

Issue: Excessive resource usage during high-frequency data loads

Examples from the Superstore: Hot reloading of multi-million rows sales transaction data weekly with integrity constraints & exact index.

Complications: Volume: Too much data can overwhelm system and network resources. Periodic updates must be performed only in short time windows in order not to interfere with analytical procedures.

Solution: Questions of the implementation of incremental loading techniques.

Conclusion of ETL

The ELT process faces significant challenges at every stage:

In order to extract data, many disparate sources need to be blended, and advanced technologies and meticulous planning are required alongside a great understanding of data cleansing, deduplication and consistency; transforming data is complicated because quality

and consistency are vital for analysis, however loading large datasets into the warehouse without issue is critical to enabling analytics quickly and accurately; all of this at the same time, in one data stream, would require a streamlined process, modern ETL technologies, and with the matching of technical and business requirements, which would ultimately allow a stable, high performance, cloud based data warehouse; supporting decision-making.

PART B Question 5. discuss how big data analytics techniques, such as map-reduce, can help with the processing and aggregation of the data.

Discussing Big Data Analytics Techniques

Tools for big data analytics such as Hadoop and MapReduce play a crucial role in processing and analysing huge number of datasets that traditional data warehouses face problems in managing (Dean & Ghemawat, 2004). Below are details on how they were applied in a superstore context.

1. MapReduce

Definition: MapReduce is a programming model for processing large datasets in two phases (Dean & Ghemawat, 2004):

1. **Map Phase:** This phase converts the raw data into a tuple format of key-value pairs.
2. **Reduce Phase:** Results are mapped to pairs (key, value) which produces final result.

How it works

MapReduce takes a huge dataset, breaks it down into smaller pieces, computes them in parallel, and merges the results. Especially useful in serving big data for batch processing and unstructured data analyses, such as social media, logs from transactions (Chen et al, 2014).

Applications in the Superstore:

- **Sales Trend Analysis:** Analyses monthly sales trends by products.

Example: The map portion organizes the sales data, whereas the reduce portion aggregates revenue across product lines.

- **Customer Segmentation:** segments customers by demographics, buying behaviour, and loyalty as an example.

Example: For example, assume we use the map function to detect purchasing habits, and the reduce function would aggregate the data and create segments such as “frequent buyers” or “seasonal shoppers

Advantages: Scalable to large datasets (Jin et al., 2015). Ans It is automatically managed and fault-tolerant against the failure of nodes.

Challenges: Needs advanced programming skills to be implemented efficiently and immediately look at batch-processing capabilities.

2. Hadoop

Definition: Hadoop is an open-source framework that facilitates the storage and analysis of massive amounts of unstructured data (White, 2012). It comprises:

- **HDFS (Hadoop Distributed File System):** Stores data across multiple nodes.
- **MapReduce:** Processes distributed data efficiently.

How It Works:

On the other hand, Hadoop disperses the data through inexpensive clusters, replicating information as fault tolerance but balancing computational efficiency (Shvachko et al. 2010).

Applications in the Superstore:

- **Handling Unstructured Data:** Stores and processes customer reviews from various sources for sentiment analysis.
- **Example:** review data is stored in HDFS and sentiment patterns are discovered using map reduce

Inventory Optimization: Scrutinizes inventory levels, supplier shipments and seasonal demand.

- **Example:** For instance: The map function pulls stock data, and the reduce function predicts next-level stock requirements.

Advantages:

- Designed to store all three structures of data (Ziko Poulos et al., 2013), the system works on commodity hardware at no cost and runs in conjunction with tools like Hive and Pig to improve querying functionality.

Challenges:

This way is efficient but needs experience in distributed systems, because the indirect inefficient use of resources will lead to lower performance.

Comparing MapReduce and Hadoop

Feature	MapReduce	Hadoop
Focus	Data processing	Data storage and processing
Data Type	Structured and semi structured	Unstructured, semi structured, and structured
Scalability	Limited to processing tasks	Highly scalable for storage and analytics
Use Case	Batch processing (e.g. customer segmentation)	Distributed data storage and large-scale analysis

Figure 21 - Handling Unstructured Data

Overview

MapReduce and Hadoop are transformative tools in big data analytics:

MapReduce is ideal for computation-heavy tasks such as client segmentation, sales trends, and batch processing, whilst Hadoop handles large amounts of distributed, unstructured data such as customer reviews, thereby allowing organizations (supermarkets) to increase operational efficiency, support data-driven decision making, and obtain further insights, so these are what allow for the solutions to what we call modern-day big data.

Reflection on our Group Work

Part A Reflection

The database design phase for the superstore was successful, this included a use case diagram, detailed ER modelling with 20 entities, and four logical models covering various different aspects. SQL queries were implemented and a sample data inserted helped make the project be practical.

Part B Reflection

The data warehouse and analytics was effectively mapped out to analyse possibilities, also an appropriate schema was designed and a demonstration of OLAP operations. The discussion of ETL challenges and big data analytics techniques provided valuable insights into large – scale data processing considerations.

Areas for Improvement

Database Design

- Develop more constraints
- Insert more values to tables
- Inner Join tables using SQL
- Build an error handling system.

Data Warehouse

- Create more sophisticated ETL processes
- Add historical data
- Add in data quality checks

Technical Documentation

- Develop a user guide
- Provide training materials
- Implement testing procedures

Analytics

- Use Power BI to build interactive dashboards.
- Add in AI or Machine Learning to predict future sales performance.
- Create automated reporting system.

System

- Add in audit trails
- Develop system scalability plans.

This project helped equipped both of us in gaining experience in designing databases, data warehouses, big data analytics and performing queries in SQL. Learning from experience, if more time could be allocated, developing a more detailed documentation, improving security implementation can take our project to a higher standard.

References

- Chen, J. (2014) 'CRESP: Towards optimal resource provisioning for MapReduce computing in public clouds', *ResearchGate*. Available at: https://www.researchgate.net/publication/262231799_CRESP_Towards_Optimal_Resource_Provisioning_for_MapReduce_Computing_in_Public_Clouds (Accessed: 11 January 2025).
- Dean, J. and Ghemawat, S. (2004) 'MapReduce: Simplified data processing on large clusters', *ACM OSDI 2004*, 19–22 October, San Francisco, USA. Available at: <https://static.googleusercontent.com/media/research.google.com/en/archive/mapreduce-osdi04.pdf> (Accessed: 19 January 2025).
- Inmon, W. H. (2005) General architecture of a data warehouse. [online] Available at: https://www.researchgate.net/figure/General-architecture-of-a-data-warehouse-Inmon-2005_fig1_316340915 [Accessed: 30 January 2025].
- Jin, X. et al. (2015) 'Community structure mining in big data social media networks with MapReduce', *ResearchGate*. Available at: https://www.researchgate.net/publication/276464522_Community_structure_mining_in_big_data_social_media_networks_with_MapReduce (Accessed: 8 January 2025).
- Kimball, R. and Ross, M. (2013) Dimensional modelling techniques. [online] Available at: <https://www.kimballgroup.com/wp-content/uploads/2013/08/2013.09-Kimball-Dimensional-Modeling-Techniques11.pdf> [Accessed: 9 January 2025].
- Kimball, R., & Ross, M. (2009). *Data warehouse design: Modern principles and methodology*. Wiley.
- Microsoft (2024) Title of the page. [online] Available at: <https://learn.microsoft.com/en-us/fabric/data-warehouse/tables> [Accessed: 7 January 2025].
- Shvachko, K. et al. (2010) 'HDFS architecture design and implementation', Available at: <https://pages.cs.wisc.edu/~akella/CS838/F15/838-CloudPapers/hdfs.pdf> (Accessed: 31 January 2025).
- White, T. (2012) *Hadoop: The definitive guide*. 3rd edn. January. Available at: <https://www.isical.ac.in/~acmsc/WBDA2015/slides/hq/Oreilly.Hadoop.The.Definitive.Guide.3rd.Edition.Jan.2012.pdf> (Accessed: 5 January 2025).
- Zikopoulos, P., et al. (2013) 'A conceptual architecture of big data analytics', *ResearchGate*. Available at: https://www.researchgate.net/figure/A-conceptual-Architecture-of-Big-Data-Analytics-Zikopoulos-et-al-2013_fig1_328531358 (Accessed: 31 January 2025).