4.2. Database Management System (DBMS)

Chosen DBMS: MySQL

- **Storage Requirements:**
  MySQL's InnoDB storage engine is ACID-compliant, supporting foreign keys and ensuring data integrity. It is scalable for medium to large datasets, and suitable for retail operations.
- **User Access Needs:**
  MySQL supports role-based access control to ensure secure and restricted data access for different roles, such as employees and managers. It also handles concurrent user connections seamlessly, critical for retail environments.
- **Data Manipulation and Retrieval:**
  MySQL provides robust SQL querying capabilities for data insertion, retrieval, updates, and deletion. Indexing optimizes query performance, and replication enhances read scalability for reporting needs.

4.3. Example Queries and Transactions

- **Data Insertion:**
- INSERT INTO customers (customerId, name, email) VALUES (1, 'John Doe', 'john@example.com');
- **Data Retrieval:**
- SELECT name, email FROM customers WHERE customerId = 1;
- **Data Update:**
- UPDATE customers SET email = 'newemail@example.com' WHERE customerId = 1;
- **Data Deletion:**
- DELETE FROM customers WHERE customerId = 1;
- **Specific Query (e.g., customer transactions):**
- SELECT transactions.* FROM transactions
- INNER JOIN customers ON transactions.customerId = customers.customerId
- WHERE customers.name = 'John Doe';

---

5. Data Management Pipeline

5.1. Data Capture

In the Retail Customer Service system, data is collected through:

- **User Inputs:** Customers and employees interact with the system via forms to input customer feedback, transaction details, and employee logs.
- **APIs:** Integration with external systems like payment gateways and customer relationship management (CRM) tools.
- **Manual Entry:** Managers enter data like employee performance reviews or store-specific updates.
- **File Uploads:** CSV files for importing bulk data such as customer lists or transaction histories.

5.2. Data Cleaning Process

Techniques Used:

- **Handling Missing Values:** Missing entries in non-critical fields are set to default values, while mandatory fields prompt user input before saving.
- **Normalizing Data:** Data is structured to ensure consistency, e.g., separating full names into first and last names.
- **Removing Duplicates:** Duplicate entries for customers or transactions are identified and removed using SQL queries.

Stages:

- **Extraction:** Data is pulled from APIs, CSV files, or user inputs.
- **Transformation:** Validation, deduplication, and normalization are applied using MySQL scripts and ETL tools like Talend or Apache NiFi.
- **Loading:** Cleaned data is saved to MySQL tables, ensuring integrity with foreign key constraints.

5.3. Data Storage and Security

Storage Strategies:

- **Indexing:** Primary and foreign keys are indexed to optimize query performance for relationships like customer-to-transaction lookups.
- **Partitioning:** Transactions are partitioned by date for better performance on time-based queries.

Security Measures:

- **Encryption:** Sensitive data like payment information is encrypted at rest and in transit.
- **Access Controls:** Role-based permissions ensure that employees can only access data relevant to their roles, with managers having broader privileges.
- **Backups:** Automated backups protect against data loss.

This pipeline ensures efficient, secure, and reliable data handling in the retail environment.