Navigating Logistics

Insights from SQL Data Analysis

.....

Vedant Thorat

Email – vedant2000thorat@gmail.com

Table of content

Introduction

In the rapidly evolving landscape of logistics management, efficient handling of resources and timely delivery of goods are critical for the success of any organization. The logistics industry faces multifaceted challenges ranging from inventory management to customer satisfaction, necessitating the implementation of robust systems to streamline operations and enhance overall efficiency.

This report documents the development and implementation of a database management system tailored to address the intricacies of logistics management. The project aims to provide a comprehensive solution for recording, tracking, and managing various aspects of logistics operations, including shipment details, customer information, employee management, and payment processing.

By leveraging relational database technology, the project endeavors to facilitate seamless coordination among stakeholders, optimize resource utilization, and improve decision-making processes within the logistics company. Through the systematic organization of data and the implementation of tailored queries, stored procedures, and triggers, the database system aims to enhance operational transparency, minimize errors, and ultimately enhance the overall quality of service delivery.

In the following sections, we delve into the problem domain, outline the objectives of the project, detail the database schema, and provide insights into the methodology employed for the design and implementation of the database system. Furthermore, we discuss the queries, stored procedures, and triggers developed to address specific requirements and ensure the integrity and efficiency of the database system.

Overall, this project endeavors to contribute to the optimization of logistics management processes, thereby enabling organizations to meet the evolving demands of the industry and deliver superior value to customers.

Abstract

Logistics is the support function of an organization, and it means having the right object at the right place at the right time. Logistics deals with various kinds of methods to control the flow of resources from one place to another. One of the major and most important factors is cost, which is being dealt with the utmost attention. The project is being designed keeping in mind the details of the various requirements of logistics, such as keeping records of the goods, i.e., their details and the kind of content that is stored in the shipment that is to be delivered.

A relational database management system (RDBMS) is similar to a DBMS. The difference is that in RDBMS, the entities and values in tables are related to one another. Also, the tables are related to each other. Thus, it is called “relational.”.

Problem Description

The logistics company provides services in both the international and domestic sectors. Logistics management takes into consideration every facility that has an impact on cost. It plays an important role in making the product conform to customer requirements. Also, it involves efficient integration of suppliers, manufacturers, imports and exports, and other activities at many levels, from the strategic level through the tactical to the operational level.

Customers can send different types of shipping content. Payment is to be made at the same time the product is delivered to the client. The delivery boy and center head can update the status of the shipment. Create a database schema and table relationships that can be used with any technology.

Objectives

1. **Efficient Data Management:**

   Develop a database system to efficiently store, organize, and manage data related to shipments, customers, employees, and payments in the logistics domain.

2. **Improved Tracking and Monitoring:**

   Implement features to enable real-time tracking and monitoring of shipments, allowing stakeholders to monitor the progress of shipments from origin to destination.

3. **Enhanced Decision-Making:**

   Provide tools and functionalities for generating reports, analytics, and insights to support data-driven decision-making processes within the logistics company.

4. **Streamlined Operations:**

   Streamline logistics operations by automating routine tasks, such as invoice generation, payment processing, and shipment status updates, to improve operational efficiency and reduce manual errors.

5. **Customer Satisfaction:**

   Enhance customer satisfaction by providing accurate and timely information about shipment status, facilitating seamless communication, and ensuring prompt resolution of customer queries and issues.

6. **Employee Management:**

   Facilitate effective management of employees involved in logistics operations by providing tools for assigning tasks, tracking performance, and managing schedules.

7. **Compliance and Security:**

   Ensure compliance with regulatory requirements and industry standards for data security, privacy, and integrity, safeguarding sensitive information and preventing unauthorized access or misuse of data.

8. **Scalability and Flexibility:**

   Design the database system to be scalable and flexible, capable of accommodating future growth and evolving business requirements in the dynamic logistics industry.

9. **Integration with External Systems:**

   Enable seamless integration with external systems, such as ERP (Enterprise Resource Planning) systems, accounting software, and third-party logistics providers, to facilitate data exchange and interoperability.

10. **Continuous Improvement:**

    Establish mechanisms for collecting feedback, monitoring system performance, and identifying areas for improvement, ensuring the ongoing optimization and refinement of the logistics management database system.

These objectives outline the goals and purposes of the logistics management database project, providing a clear direction for its development and implementation. They address key aspects such as data management, operational efficiency, customer satisfaction, compliance, and continuous improvement, ultimately contributing to the success and competitiveness of the logistics company.

Scope

It is of critical importance to the organization how it delivers products and services to the customer, whether the product is tangible or intangible. The effective and efficient physical movement of the tangible product will speak of the intangible services associated with the product and the organization that is delivering it.

In the case of intangible products, the delivery of tangibles at the right place and time will speak volumes about their quality. On the macro level, infrastructure such as various modes of transport, transportation equipment, storage facilities, connectivity, and information processing is contributing to a large extent to the physical movement of goods produced in the manufacturing, mining, and agriculture sectors.

This speed and reliability in the distribution of products and services contribute to a great extent to the growth of a country's domestic and international trade.

Database Schemas

➢ Table Definition

1. Employee_Details Table:

   This table contains the information of the employees.

Column Name	Data Type	Description
Emp_ID	INT	Employee ID (Primary Key)
Emp_NAME	VARCHAR (30)	Name of the employee
Emp_Branch	VARCHAR (15)	Branch name
Emp_Designation	VARCHAR (40)	Designation of the employee
Emp_Addr	VARCHAR (100)	Address of the employee
Emp_Cont_No	VARCHAR (10)	Contact number of the employee

2. Membership Table:

   This table contains the membership details of the customer or client.

Column Name	Data Type	Description
M_ID	INT	Membership ID associated with the client (Primary Key)
Start_Date	TEXT	Start date of the membership
End_Date	TEXT	End date of the membership

3. Customer Table:

   This table contains the information of the customers or clients

Column Name	Data Type	Description
Cust_ID	INT	Client ID (Primary Key)
Cust_NAME	VARCHAR (30)	Name of the client
Cust_Email_ID	VARCHAR (50)	Email of the client
Cust_Cont_No	VARCHAR (10)	Contact number of the client
Cust_Addr	VARCHAR (100)	Address of the client
Cust_Type	VARCHAR (30)	Type of client (wholesale, Retail, Internal Goods)
Membership_M_ID	INT	Membership ID (Foreign Key)

4. Payment_Details Table:

   This table contain the payment details.

Column Name	Data Type	Description
Payment_ID	INT	Payment unique ID (Primary Key)
Amount	INT	Price to be paid by the client
Payment_Status	VARCHAR (10)	Payment Status (Paid / Not Paid)
Payment_Date	TEXT	Date when payment is made by the client
Payment_Mode	VARCHAR (25)	Mode of Payment (COD / Card Payment)
Shipment_SH_ID	INT	Shipment ID (Foreign Key)
Customer_Cust_ID	INT	Client ID (Foreign Key)

5. Shipment_Details Table:

   This table contains the shipment details.

Column Name	Data Type	Description
SD_ID	INT	Shipment ID (Primary Key)
SD_Content	VARCHAR (40)	Type of shipment content
SD_Domain	VARCHAR (15)	Shipment Domain (Internation / Domestic)
SD_Type	VARCHAR (15)	Service Type (Express / Regular)
SD_Weight	VARCHAR (10)	Shipment Weight
SD_Charges	INT	Shipment Charges
SD_Addr	VARCHAR (100)	Source Address
DS_Addr	VARCHAR (100)	Destination Address
Customer_Cust_ID	INT	Client ID (Foreign Key)

6. Status Table:

   This table contains details about the delivery status.

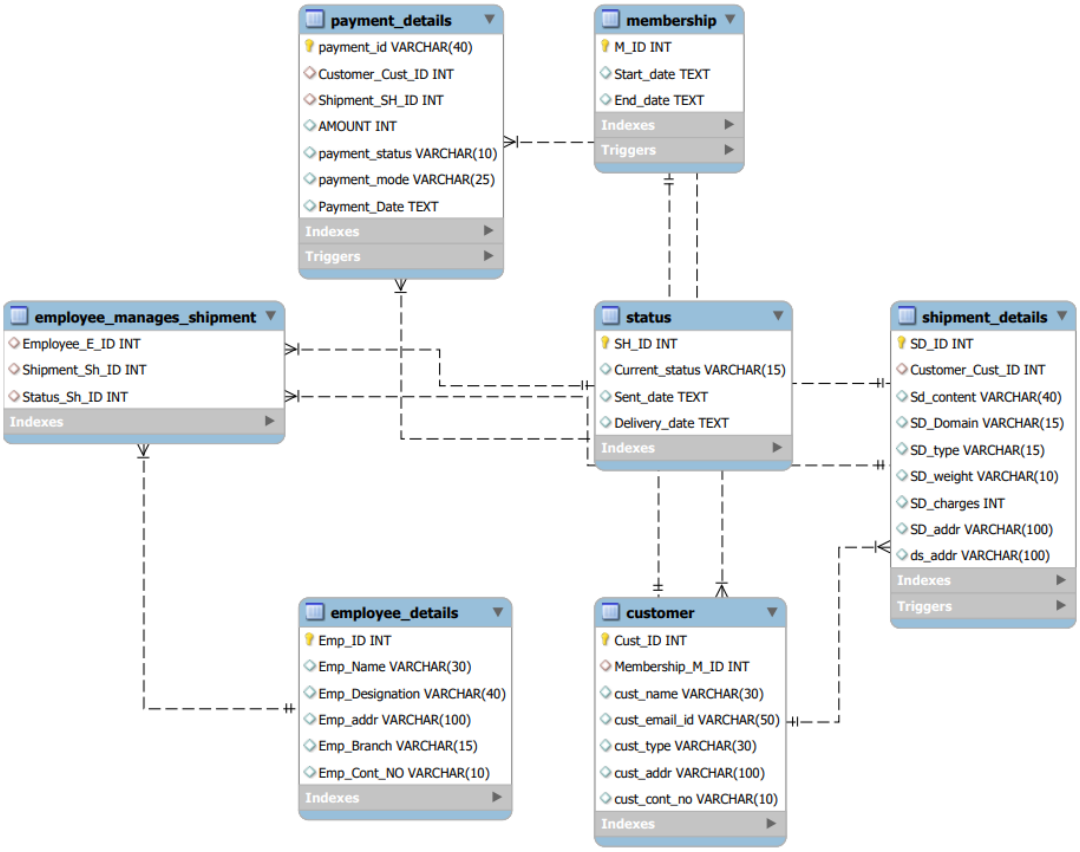
Column Name	Data Type	Description
Current_Status	VARCHAR (15)	Current status of the shipment
Sent_Date	Text	Date when shipment was sent
Delivery_Date	TEXT	Date when the product was/will be delivered
SH_ID	INT	Shipment_ID (Primary Key)

7. Employee_Manages_Shipment Table:

   This is relation table between the employee and the shipment table.

Column Name	Data Type	Description
Employee_Emp_ID	INT	Employee ID (Foreign Key)
Shipment_SH_ID	INT	Shipment ID form shipment details table (Foreign Key)
Status_SH_ID	INT	Shipment ID from status table (Foreign Key)

ER Diagram


payment_details
payment_id VARCHAR(40)
Customer_Cust_ID INT
Shipment_SH_ID INT
AMOUNT INT
payment_status VARCHAR(10)
payment_mode VARCHAR(25)
Payment_Date TEXT
Indexes
Triggers
membership
M_ID INT
Start_date TEXT
End_date TEXT
Indexes
Triggers
employee_manages_shipment
Employee_E_ID INT
Shipment_Sh_ID INT
Status_Sh_ID INT
Indexes
status
SH_ID INT
Current_status VARCHAR(15)
Sent_date TEXT
Delivery_date TEXT
Indexes
shipment_details
SD_ID INT
Customer_Cust_ID INT
Sd_content VARCHAR(40)
SD_Domain VARCHAR(15)
SD_type VARCHAR(15)
SD_weight VARCHAR(10)
SD_charges INT
SD_addr VARCHAR(100)
ds_addr VARCHAR(100)
Indexes
Triggers
employee_details
Emp_ID INT
Emp_Name VARCHAR(30)
Emp_Designation VARCHAR(40)
Emp_addr VARCHAR(100)
Emp_Branch VARCHAR(15)
Emp_Cont_NO VARCHAR(10)
Indexes
customer
Cust_ID INT
Membership_M_ID INT
cust_name VARCHAR(30)
cust_email_id VARCHAR(50)
cust_type VARCHAR(30)
cust_addr VARCHAR(100)
cust_cont_no VARCHAR(10)
Indexes

Methodology

The development and implementation of the logistics management database system followed a structured approach that encompassed several key steps:

1. **Requirement Analysis:**

   The project commenced with a thorough analysis of the requirements and objectives of the logistics company. This involved engaging with stakeholders, including management, employees, and customers, to understand their needs and pain points in logistics management.

2. **Database Design:**

   Based on the identified requirements, a comprehensive database schema was designed to model the various entities and relationships involved in logistics management. The schema was refined iteratively, incorporating feedback and ensuring alignment with industry standards and best practices.

3. **Database Implementation:**

   The designed database schema was implemented using a relational database management system (RDBMS). SQL (Structured Query Language) was utilized to create tables, define relationships, and establish constraints to ensure data integrity and consistency.

4. **Data Population:**

   Test data was generated and populated into the database to simulate real-world scenarios and validate the functionality of the system. This involved creating sample shipments, customers, employees, and payments to assess the efficacy of the database design and implementation.

5. **Query Development:**

A set of SQL queries was developed to retrieve, manipulate, and analyze data stored in the database. These queries were tailored to address specific business requirements, such as tracking shipments, calculating payment totals, and generating reports for stakeholders.

6. **Stored Procedures and Triggers:**

Stored procedures and triggers were implemented to automate routine tasks and enforce data validation rules within the database system. Stored procedures were created for inserting, updating, and deleting records, while triggers were used to enforce constraints and perform data validation checks.

7. **Testing and Validation:**

The database system underwent rigorous testing to validate its functionality, performance, and reliability. Unit tests were conducted to verify individual components, while integration testing was performed to ensure seamless interaction between different modules of the system.

8. **Deployment and Training:**

Upon successful testing and validation, the database system was deployed into the production environment. Training sessions were conducted to familiarize employees with the new system and ensure smooth adoption and usage across the organization.

9. **Monitoring and Maintenance:**

Post-deployment, the database system is being monitored closely to track performance metrics, identify potential issues, and ensure ongoing optimization

and maintenance. Regular updates and enhancements are planned to address evolving business requirements and technological advancements.

By following this methodology, the logistics management database system was developed and implemented in a systematic manner, aligning with the objectives of the project and meeting the needs of the logistics company.

Data Analysis

➢ Queries

1. Calculate total payments made by each customer.

To aggregate payment data and provide insights into customer spending behavior and revenue generation.

```
SELECT
    c.cust_id AS Customer_ID,
    c.cust_name AS Customer_Name,
    SUM(p.amount) AS Amount
FROM
    customer AS c
        JOIN
    payment_details AS p ON c.cust_id = p.customer_cust_id
GROUP BY p.customer_cust_id;
```

Enables analysis of customer payment patterns, identification of high-value customers, and assessment of revenue streams.

Customer_ID	Customer_Name	Amount
3	Janelle	98982
114	Tiffany	69113
175	Rayshawn	99492
207	Boyd	99367
230	Mitchell	49302
249	Rayshawn	47260
308	Morgan	39234
310	Michaela	32800
359	Chelsey	62151
390	Destiny	86040
515	Christina	1714
519	Rayburn	7068
563	Brenda	20187
584	Marie	5769
693	Stacy	73589
805	Korie	91926
896	Frances	28701
1087	Lydia	45852
1126	Steve	57460
1164	Maurice	30192
1201	Sydney	52868
1202	Chasity	41003

2. Get the current status of a shipment.

To retrieve real-time information about the status and location of a shipment.

```
SELECT
    CURRENT_Status, SENT_DATE, DELIVERY_DATE
FROM
    Status
WHERE
    SH_ID = '42';                    # here you can find current status of a shipment by entering Shipment ID
```

Facilitates proactive management of logistics operations and timely response to shipment-related issues or delays.

	CURRENT_Status	SENT_DATE	DELIVERY_DATE
▶	DELIVERED	1/20/1979	9/12/1979

3. List all shipments managed by a specific employee.

To identify shipments assigned to a particular employee for management.

```
SELECT
    e.emp_id AS Employee_ID,
    e.emp_name AS Employee_Name,
    ems.shipment_Sh_ID AS Shipment_Managed_By_Employee
FROM
    employee_details AS e
        JOIN
    employee_manages_shipment AS ems ON e.emp_id = ems.employee_E_ID
WHERE
    e.emp_id = 11;               # here you can find all of shipment managed by employee by there Employee ID
```

Helps supervisors track employee workload, assess performance, and ensure equitable distribution of tasks.

	Employee_ID	Employee_Name	Shipment_Managed_By_Employee
▶	11	Rita	895

4. Calculate the average shipment charge for each shipment domain.

To analyze pricing trends and profitability across different shipment domains.

```
SELECT
    SD_Domain AS Domain,
    ROUND(AVG(SD_charges), 2) AS Average_Charge
FROM
    shipment_details
GROUP BY SD_Domain;
```

Provides insights into pricing strategies, revenue generation, and potential areas for pricing optimization.

	Domain	Average_Charge
▶	International	940.19
	Domestic	936.12

5. Find the total number of shipments managed by each employee.

To evaluate employee productivity and workload distribution.

```
SELECT
    e.emp_id AS Employee_ID,
    e.emp_name AS Employee_Name,
    COUNT(ems.shipment_Sh_ID) AS Count_Shipment_Managed_By_Employee
FROM
    employee_details AS e
        JOIN
    employee_manages_shipment AS ems ON e.emp_id = ems.employee_E_ID
GROUP BY e.emp_id;
```

Facilitates workforce management, performance evaluation, and optimization of resource allocation.

Employee_ID	Employee_Name	Count_Shipment_Managed_By_Employee
582	Harriette	1
396	Matthew	1
545	Geraldine	1
770	Brenda	1
991	Malie	1
437	Fred	1
805	Clay	1
803	Alaysha	1
295	Kelli	1
804	Ryker	1
54	Johnnie	1
853	Zykeria	1
902	Selma	1
163	Kathryn	1
993	Karl	1
891	Freda	1
950	David	1
292	Winnifred	1
90	Bryce	1
49	Maeve	1
100	Seth	1
405	Maria	1
584	Taron	1
326	Bryce	1

Result 41

6. List all customers who have active memberships.

To identify customers with active memberships in the system.

```
SELECT
    c.*
FROM
    Customer as c
        JOIN
    Membership as m ON c.Cust_ID = m.M_ID
WHERE
    CURDATE() BETWEEN STR_TO_DATE(m.START_DATE, '%Y-%m-%d') AND STR_TO_DATE(m.END_DATE, '%Y-%m-%d');
```

Enables targeted communication and service offerings to active members, fostering customer loyalty and engagement.

7. Retrieve the details of the employee who manages the most shipments.

To link shipment data with customer information for comprehensive reporting.

```
SELECT
    *
FROM
    employee_details AS ed
        JOIN
    (SELECT
        employee_e_id AS Employee_ID, COUNT(*) AS Number_Shipment
    FROM
        employee_manages_shipment
    GROUP BY Employee_ID
    ORDER BY number_shipment DESC
    LIMIT 1) AS Most_Shipment ON ed.emp_id = Most_Shipment.Employee_ID;
```

Provides a complete view of shipments and their associated customers, facilitating personalized customer service and order tracking.

Emp_ID	Emp_Name	Emp_Designation	Emp_addr	Emp_Branch	Emp_Cont_NO	Employee_ID	Number_Shipment
2	Zoya	Transport manager	400 Block of MASON ST	TN	9250747856	2	1

8. Find the customers who have not made any payments.

This query aims to identify customers who have not made any payments within the specified timeframe or since their registration in the system.

```
SELECT
    c.cust_Id AS Customer_ID, c.cust_name AS Customer_Name
FROM
    customer AS c
        JOIN
    payment_details AS pd ON c.cust_id = pd.customer_cust_id
WHERE
    pd.payment_status = 'NOT PAID';
```

By executing this query, we can generate a list of customers who have yet to make payments, allowing the logistics company to follow up with them, investigate potential issues or barriers to

payment, and implement strategies to encourage payment completion. This information is valuable for financial tracking, customer engagement, and revenue optimization efforts.

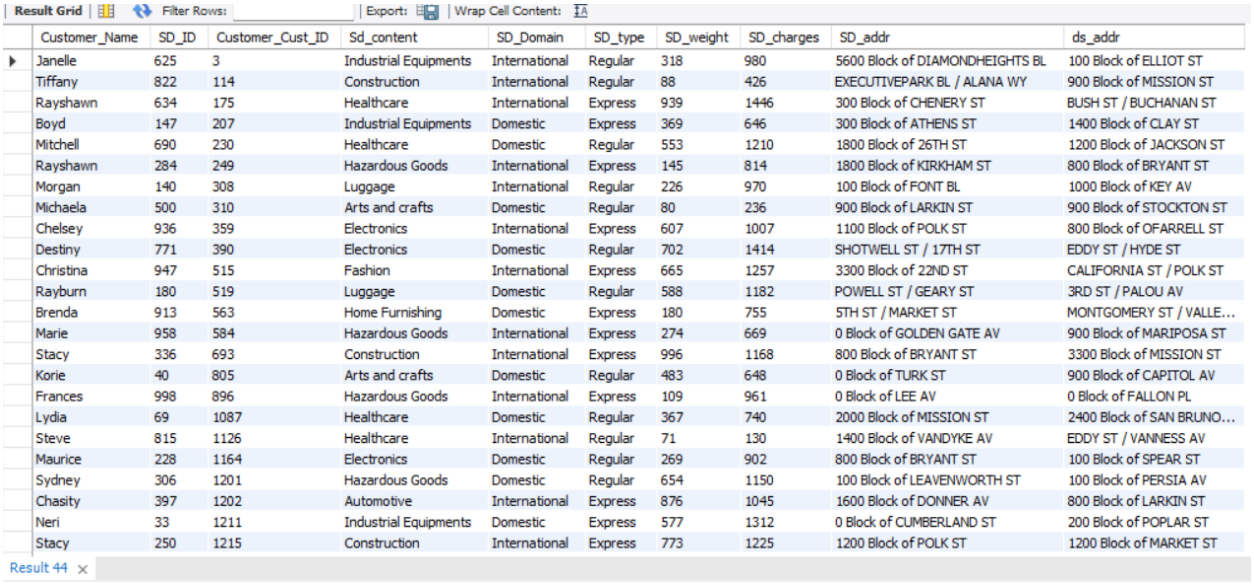
Customer_ID	Customer_Name
2216	Jaylene
1904	Stacie
7342	Jonathan
2154	Catherine
5543	Pierre
2332	Sheryl
4094	Rory
4988	Belle
175	Rayshawn
4351	Alonzo
5578	Ray
4523	Tiffany
4852	Sophie
9377	Elena
8893	Magdalene
1897	Norma
2241	Clay
6713	Merna
4283	Louise
2620	Eddie
4711	Taryn
4053	Lazaro
4272	Muriel
7005	Dejon

9. Retrieve the details of shipments along with the names of their corresponding customers. This query is designed to retrieve comprehensive information about shipments, including details such as shipment ID, content, domain, weight, charges, source address, destination address, and the names of the corresponding customers.

```
SELECT
    c.cust_name AS Customer_Name, sd.*
FROM
    customer AS c
        JOIN
    shipment_details AS sd ON c.cust_id = sd.customer_cust_id;
```

By executing this query, we can obtain a consolidated view of shipments and their associated customers. This information facilitates order tracking, customer service, and personalized

communication with customers regarding their shipments. It enables the logistics company to provide a seamless and transparent experience for customers by ensuring they have access to relevant information about their shipments and can easily reach out for support if needed.



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Customer_Name	SD_ID	Customer_Cust_ID	Sd_content	SD_Domain	SD_type	SD_weight	SD_charges	SD_addr	ds_addr
Janelle	625	3	Industrial Equipments	International	Regular	318	980	5600 Block of DIAMONDHEIGHTS BL	100 Block of ELLIOT ST
Tiffany	822	114	Construction	International	Regular	88	426	EXECUTIVEPARK BL / ALANA WY	900 Block of MISSION ST
Rayshawn	634	175	Healthcare	International	Express	939	1446	300 Block of CHENERY ST	BUSH ST / BUCHANAN ST
Boyd	147	207	Industrial Equipments	Domestic	Express	369	646	300 Block of ATHENS ST	1400 Block of CLAY ST
Mitchell	690	230	Healthcare	Domestic	Regular	553	1210	1800 Block of 26TH ST	1200 Block of JACKSON ST
Rayshawn	284	249	Hazardous Goods	International	Express	145	814	1800 Block of KIRKHAM ST	800 Block of BRYANT ST
Morgan	140	308	Luggage	International	Regular	226	970	100 Block of FONT BL	1000 Block of KEY AV
Michaela	500	310	Arts and crafts	Domestic	Regular	80	236	900 Block of LARKIN ST	900 Block of STOCKTON ST
Chelsey	936	359	Electronics	International	Express	607	1007	1100 Block of POLK ST	800 Block of OFARRELL ST
Destiny	771	390	Electronics	Domestic	Regular	702	1414	SHOTWELL ST / 17TH ST	EDDY ST / HYDE ST
Christina	947	515	Fashion	International	Express	665	1257	3300 Block of 22ND ST	CALIFORNIA ST / POLK ST
Rayburn	180	519	Luggage	Domestic	Regular	588	1182	POWELL ST / GEARY ST	3RD ST / PALOU AV
Brenda	913	563	Home Furnishing	Domestic	Express	180	755	5TH ST / MARKET ST	MONTGOMERY ST / VALLE...
Marie	958	584	Hazardous Goods	International	Express	274	669	0 Block of GOLDEN GATE AV	900 Block of MARIPOSA ST
Stacy	336	693	Construction	International	Express	996	1168	800 Block of BRYANT ST	3300 Block of MISSION ST
Korie	40	805	Arts and crafts	Domestic	Regular	483	648	0 Block of TURK ST	900 Block of CAPITOL AV
Frances	998	896	Hazardous Goods	International	Express	109	961	0 Block of LEE AV	0 Block of FALLON PL
Lydia	69	1087	Healthcare	Domestic	Regular	367	740	2000 Block of MISSION ST	2400 Block of SAN BRUNO...
Steve	815	1126	Healthcare	International	Regular	71	130	1400 Block of VANDYKE AV	EDDY ST / VANNESS AV
Maurice	228	1164	Electronics	Domestic	Regular	269	902	800 Block of BRYANT ST	100 Block of SPEAR ST
Sydney	306	1201	Hazardous Goods	Domestic	Regular	654	1150	100 Block of LEAVENWORTH ST	100 Block of PERSIA AV
Chasity	397	1202	Automotive	International	Express	876	1045	1600 Block of DONNER AV	800 Block of LARKIN ST
Neri	33	1211	Industrial Equipments	Domestic	Express	577	1312	0 Block of CUMBERLAND ST	200 Block of POPLAR ST
Stacy	250	1215	Construction	International	Express	773	1225	1200 Block of POLK ST	1200 Block of MARKET ST

Result 44

10. Find all customers who have made payments along with the payment details

This query aims to identify customers who have made payments and retrieve relevant payment details associated with each customer.

```
SELECT
    c.cust_Id AS Customer_ID,
    c.cust_name AS Customer_Name,
    pd.payment_id AS Payment_ID,
    pd.amount AS Amount,
    pd.payment_status AS Status,
    pd.payment_mode AS Mode,
    pd.payment_date AS Date
FROM
    customer AS c
        JOIN
    payment_details AS pd ON c.cust_id = pd.customer_cust_id
WHERE
    pd.payment_status = 'PAID';
```

By executing this query, we can generate a list of customers who have completed payment transactions, along with detailed information such as payment ID, payment amount, payment status, payment date, and payment mode. This information provides valuable insights into customer financial interactions, allowing the logistics company to track revenue, analyze payment trends, and manage customer accounts effectively. Additionally, it enables targeted communication with customers regarding their payment history and supports financial reporting and analysis efforts within the organization.

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Customer_ID	Customer_Name	Payment_ID	Amount	Status	Mode	Date
230	Mitchell	313cd69e-66f3-11ea-9879-7077813058ce	49302	PAID	CARD PAYMENT	2014-12-18
3189	Reginald	313dc140-66f3-11ea-a952-7077813058ce	78698	PAID	CARD PAYMENT	1997-07-10
7633	Italia	31411bc8-66f3-11ea-a4b7-7077813058ce	56881	PAID	CARD PAYMENT	1971-11-01
3042	Cecile	3145fd0a-66f3-11ea-958b-7077813058ce	95516	PAID	COD	1991-05-15
2220	Monte	31469934-66f3-11ea-8572-7077813058ce	62528	PAID	COD	1976-06-30
4233	Abby	3148bbc8-66f3-11ea-8acf-7077813058ce	60282	PAID	CARD PAYMENT	1987-08-09
2972	Rosalind	314b7a8a-66f3-11ea-97d4-7077813058ce	68227	PAID	CARD PAYMENT	1977-07-06
6153	Franklin	314c8bcc-66f3-11ea-b526-7077813058ce	77861	PAID	CARD PAYMENT	1997-09-08
8106	Matthew	314eae64-66f3-11ea-b08f-7077813058ce	83002	PAID	CARD PAYMENT	1999-03-05
3917	Larry	314f4a7a-66f3-11ea-8b83-7077813058ce	47650	PAID	COD	1977-11-08
5387	Brady	31511efa-66f3-11ea-bc4c-7077813058ce	39432	PAID	COD	2005-09-15
6513	Sapphire	3151e23a-66f3-11ea-9b38-7077813058ce	1421	PAID	COD	2019-12-12
3965	Gene	3152a56c-66f3-11ea-9340-7077813058ce	16113	PAID	COD	2013-11-15
390	Destiny	3154c800-66f3-11ea-8721-7077813058ce	86040	PAID	COD	2013-11-15
3633	Glory	3155d94c-66f3-11ea-b116-7077813058ce	56148	PAID	COD	2001-08-10
7828	Latasha	31575fac-66f3-11ea-8d99-7077813058ce	894	PAID	CARD PAYMENT	2006-07-17
896	Frances	3159d04c-66f3-11ea-95df-7077813058ce	28701	PAID	CARD PAYMENT	2013-11-15
6361	Dinah	315aba8a-66f3-11ea-9db6-7077813058ce	90380	PAID	CARD PAYMENT	1993-08-02
9486	Kaitlyn	315d041c-66f3-11ea-8af9-7077813058ce	24856	PAID	CARD PAYMENT	2002-10-20
308	Morgan	315da042-66f3-11ea-ac04-7077813058ce	39234	PAID	CARD PAYMENT	2006-07-17
8927	Courtney	315e3c68-66f3-11ea-8d08-7077813058ce	74222	PAID	COD	2004-11-20
249	Rayshawn	315f26ac-66f3-11ea-a883-7077813058ce	47260	PAID	CARD PAYMENT	1982-03-08
1164	Maurice	3160ad0c-66f3-11ea-961a-7077813058ce	30192	PAID	COD	1994-04-09
2308	Peter	31642f12-66f3-11ea-b553-7077813058ce	13740	PAID	COD	2008-08-09

Result 45

11. Retrieve all shipments along with their current status and the employee who manages them.

This query is designed to gather comprehensive information about all shipments, including their current status and the employee responsible for managing each shipment.

```
SELECT
    e.emp_id AS Employee_ID,
    e.emp_name AS Employee_Name,
    sd.sd_Id AS Shipment_ID,
    s.current_status AS Current_Status
FROM
    employee_details AS e
        JOIN
    employee_manages_shipment AS ems ON e.emp_id = ems.employee_e_id
        JOIN
    shipment_details AS sd ON sd.sd_id = ems.shipment_sh_id
        JOIN
    status AS s ON s.sh_id = ems.status_sh_id;
```

By executing this query, we can obtain a complete overview of all shipments in the system, along with real-time status updates and employee assignments. This information is crucial for logistics management, as it enables stakeholders to monitor the progress of shipments, track their locations, and ensure timely delivery. Additionally, knowing the employee responsible for each shipment facilitates effective communication and coordination within the logistics team, allowing for streamlined workflow management and proactive issue resolution. Overall, this query provides valuable insights into shipment status and employee workload distribution, supporting efficient logistics operations and customer satisfaction.

Result Grid | Filter Rows: | Export: | Wra

Employee_ID	Employee_Name	Shipment_ID	Current_Status
582	Harriette	690	DELIVERED
396	Matthew	933	DELIVERED
545	Geraldine	261	NOT DELIVERED
770	Brenda	445	NOT DELIVERED
991	Malie	722	NOT DELIVERED
437	Fred	129	DELIVERED
805	Clay	489	NOT DELIVERED
803	Alaysha	165	NOT DELIVERED
295	Kelli	164	NOT DELIVERED
804	Ryker	364	NOT DELIVERED
54	Johnnie	469	DELIVERED
853	Zykeria	158	DELIVERED
902	Selma	337	NOT DELIVERED
163	Kathryn	634	NOT DELIVERED
993	Karl	577	DELIVERED
891	Freda	907	NOT DELIVERED
950	David	870	NOT DELIVERED
292	Winnifred	982	NOT DELIVERED
90	Bryce	351	DELIVERED
49	Maeve	328	DELIVERED
100	Seth	242	NOT DELIVERED
405	Maria	421	DELIVERED
584	Taron	6	DELIVERED
326	Bryce	384	NOT DELIVERED

Result 46 ×

12. Find the customers who have spent more than the average payment amount.

This query aims to identify customers who have made payments exceeding the average payment amount across all customers.

```
SELECT
    c.cust_id AS Customer_ID, c.Cust_Name AS Customer_Name
FROM
    customer AS c
        JOIN
    (SELECT
        customer_cust_id, AVG(amount) AS Avg_amount
    FROM
        payment_details
    GROUP BY customer_cust_id
    HAVING AVG(amount) > (SELECT
            AVG(amount)
        FROM
            payment_details)) AS Cust_spend_more_than_avg ON c.cust_id = Cust_spend_more_than_avg.customer_cust_id;
```

By executing this query, we can generate a list of customers whose spending exceeds the average payment amount. This information is valuable for targeted marketing efforts, customer segmentation, and revenue optimization strategies. Identifying high-value customers who contribute disproportionately to revenue enables the logistics company to tailor its services, promotions, and loyalty programs to maximize customer satisfaction and retention. Additionally, this query supports data-driven decision-making by providing insights into customer spending behavior and preferences, allowing for more effective resource allocation and strategic planning.

Result Grid | Filter Rows:

Customer_ID	Customer_Name
3	Janelle
114	Tiffany
175	Rayshawn
207	Boyd
230	Mitchell
359	Chelsey
390	Destiny
693	Stacy
805	Korie
1126	Steve
1201	Sydney
1211	Neri
1215	Stacy
1246	Carlotta
1275	Marion
1334	Philip
1647	Bambi
2066	Catherine
2096	Jennifer
2154	Catherine
2183	Dennis
2216	Jaylene
2220	Monte
2332	Sheryl

Result 47

13. List all shipments that have not been delivered yet

This query is intended to identify shipments that are currently in transit and have not been delivered to their destination.

```
SELECT
    *
FROM
    status
WHERE
    current_status = 'NOT DELIVERED';
```

By executing this query, we can generate a list of shipments with pending delivery status. This information is valuable for logistics management as it allows stakeholders to track shipments that require attention, monitor delivery progress, and proactively address any delays or issues. Identifying shipments that have not been delivered yet enables the logistics team to prioritize resources, allocate additional support if necessary, and communicate effectively with customers regarding the status of their shipments. Additionally, this query supports performance evaluation and process improvement efforts by providing insights into delivery efficiency and customer service responsiveness.

➢ Stored Procedure

1. Create a stored procedure to insert a new employee into the employee_details table.

The purpose of this stored procedure is to streamline the process of adding new employees to the organization's workforce database.

```
delimiter //
create procedure Insert_Employee_Details(
in IE_Emp_ID int,
in IE_Emp_Name varchar(30),
in IE_Emp_Designation varchar(40),
in IE_Emp_Addr varchar(100),
in IE_Emp_Branch varchar(15),
in IE_Emp_Cont_No varchar(10))
begin
insert into employee_details (Emp_ID, Emp_Name, Emp_Designation, Emp_Addr, Emp_Branch, Emp_Cont_No)
values (IE_Emp_ID, IE_Emp_Name, IE_Emp_Designation, IE_Emp_Addr, IE_Emp_Branch, IE_Emp_Cont_No);
end //
```

With this stored procedure, HR personnel or system administrators can efficiently add new employee details to the employee_details table by providing the necessary parameters, such as employee ID, name, branch, designation, address, and contact number. The stored procedure validates and inserts the provided data into the database, ensuring that the employee information is accurately captured and stored for future reference. This enhances data integrity, facilitates workforce management, and supports regulatory compliance requirements.

```
CALL InsertEmployee(123, 'John Doe', 'Manager', '123 Main St', 'Branch Name', '1234567890');
```

997	John	Sales Manager	123 Main St, City	MA	1234567890

2. Create a stored procedure to insert a new shipment into the shipment_details table.

The purpose of this stored procedure is to streamline the process of adding new shipment details to the shipment_details table.

```
delimiter //
create procedure Insert_Shipment_Details(
    in IS_SD_ID int,
    in IS_Customer_Cust_ID int,
    in IS_SD_Content varchar(40),
    in IS_SD_Domain varchar(15),
    in IS_SD_Type varchar(15),
    in IS_SD_Weight varchar(10),
    in IS_SD_Charges int,
    in IS_SD_Addr varchar(100),
    in IS_DS_Addr varchar(100)
)
begin
    insert into shipment_details(SD_ID, Customer_Cust_ID, SD_Content, SD_Domain, SD_Type, SD_Weight, SD_Charges, SD_Addr, DS_Addr)
    values ( IS_SD_ID, IS_Customer_Cust_ID, IS_SD_Content, IS_SD_Domain, IS_SD_Type, IS_SD_Weight, IS_SD_Charges, IS_SD_Addr, IS_DS_Addr);
end //

delimiter ;
```

With this stored procedure, logistics personnel or system administrators can efficiently add new shipment details to the shipment_details table by providing the necessary parameters, such as shipment ID, content, domain, type, weight, charges, source address, destination address, and customer ID. The stored procedure validates and inserts the provided data into the database, ensuring that the shipment information is accurately captured and stored for future reference. This enhances data integrity, facilitates shipment management, and supports operational efficiency within the logistics management system.

```
call Insert_Shipment_details(22, 114, "Healthcare",  "International", "Regular",  "1117", 777, "17TH ST / 800 Block", "JONES ST / GOLDEN GATE AV");
```

22	114	Healthcare	International	Regular	1117	777	17TH ST / 800 Block	JONES ST / GOLDEN GATE AV

3. Create a stored procedure to insert a new payment into the payment_details table.

The purpose of this stored procedure is to facilitate the addition of new payment details to the payment_details table.

```
delimiter //

create procedure Insert_Payment_Details(
    in IP_Payment_ID varchar(40),
    in IP_Customer_Cust_ID int,
    in IP_Shipment_SH_ID int,
    in IP_Amount int,
    in IP_Payment_Status varchar(10),
    in IP_Payment_Mode varchar(25),
    in IP_Payment_Date text
)
begin
    insert into Payment_Details (Payment_ID, Customer_Cust_ID, Shipment_SH_ID, Amount, Payment_Status, Payment_Mode, Payment_Date)
    values (IP_Payment_ID, IP_Customer_Cust_ID, IP_Shipment_SH_ID, IP_Amount, IP_Payment_Status, IP_Payment_Mode, IP_Payment_Date);
end //

delimiter ;
```

With this stored procedure, finance personnel or system administrators can efficiently add new payment details to the payment_details table by providing the necessary parameters, such as payment ID, amount, status, date, mode, shipment ID, and client ID. The stored procedure validates and inserts the provided data into the database, ensuring that the payment information is accurately captured and stored for financial tracking and reporting purposes. This enhances data integrity, facilitates financial management, and supports operational efficiency within the logistics management system.

```
call Insert_Payment_Details('41affa0a-66f3-11ea-8464-7077813058ce', 2573, 536, 8206, 'PAID', 'CARD PAYMENT', '2016-09-21');
```

payment_id	Customer_Cust_ID	Shipment_SH_ID	AMOUNT	payment_status	payment_mode	Payment_Date
41affa0a-66f3-11ea-8464-7077813058ce	2573	536	8206	PAID	CARD PAYMENT	2016-09-21

4. Create a stored procedure to assign an employee to manage a shipment.

The purpose of this stored procedure is to assign an employee to manage a specific shipment within the logistics management system.

```
delimiter //

create procedure Insert_Employee_Manages_Shipment(
    in IEMS_Employee_E_ID int,
    in IEMS_Shipment_SH_ID int,
    in IEMS_Status_SH_ID int
)
begin
    insert into Employee_Manages_Shipment(Employee_E_ID, Shipment_SH_ID, Status_SH_ID)
    values (IEMS_Employee_E_ID, IEMS_Shipment_SH_ID, IEMS_Status_SH_ID);
end//

delimiter //

call Insert_Employee_Manages_Shipment('900', '45', '018');
```

With this stored procedure, logistics supervisors or system administrators can efficiently assign employees to manage shipments by providing the necessary parameters, such as employee ID and shipment ID. The stored procedure validates the provided data and updates the corresponding record in the database to reflect the employee's assignment to the shipment. This enhances workflow management, facilitates accountability, and supports effective oversight of shipment handling within the logistics management system.

5. Create a stored procedure to insert a new customer into the customer table.

The purpose of this stored procedure is to streamline the process of adding new customer details to the customer table.

```
delimiter //

create procedure Insert_Customer(
    in IC_Cust_ID int,
    in IC_Membership_M_ID int,
    in IC_Cust_Name varchar(30),
    in IC_Cust_Email_ID varchar(50),
    in IC_Cust_Type varchar(30),
    in IC_Cust_Addr varchar(100),
    in IC_Cust_Cont_No varchar(10)
)
begin
    insert into Customer(Cust_ID, Membership_M_ID, Cust_Name, Cust_Email_ID, Cust_Type, Cust_Addr, Cust_Cont_No)
    values (IC_Cust_ID, IC_Membership_M_ID, IC_Cust_Name, IC_Cust_Email_ID, IC_Cust_Type, IC_Cust_Addr, IC_Cust_Cont_No);
end//

delimiter ;

call Insert_Customer( 9969, 989, "Jack", "jack1845@gmail.com", "Internal Goods", "13th ST", "991288456") ;
```

With this stored procedure, sales personnel or system administrators can efficiently add new customer details to the customer table by providing the necessary parameters, such as customer ID, name, email, contact number, address, and type. The stored procedure validates and inserts the provided data into the database, ensuring that the customer information is accurately captured and stored for future reference. This enhances data integrity, facilitates customer relationship management, and supports targeted marketing and sales efforts within the logistics management system.

➢ Data Validation

1. Create a trigger to ensure that the End_Date in the membership table is always greater than the Start_Date.

The purpose of this trigger is to enforce data integrity and prevent inconsistencies in the membership table by ensuring that the End_Date is always greater than the Start_Date.

```
delimiter //

CREATE TRIGGER ensure_end_date_after_start_date
BEFORE INSERT ON membership
FOR EACH ROW
BEGIN
    IF NEW.End_Date <= NEW.Start_Date THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'End date must be greater than start date';
    END IF;
END;
//

delimiter ;
```

 Justification: Ensuring that the End_Date is greater than the Start_Date is essential to maintain the validity of membership records. Allowing End_Date to be less than or equal to Start_Date could lead to inaccurate representations of membership durations and potentially cause confusion or errors in membership management processes.

 Trigger Logic: When a new row is inserted or an existing row is updated in the membership table, the trigger activates. It checks if the End_Date is less than or equal to the Start_Date. If this condition is met, the trigger raises an error, preventing the insertion or update operation from completing successfully.

 Outcome: By implementing this trigger, the database ensures that all membership records adhere to the specified rule, maintaining consistency and accuracy in the data. Any attempt to violate the rule results in the trigger preventing the operation, thereby upholding data integrity and supporting reliable membership management within the logistics management system.

 Impact: This trigger provides an additional layer of validation and constraint enforcement, safeguarding the integrity of membership data and reducing the likelihood of data entry errors or inconsistencies. It enhances the reliability and trustworthiness of the database, supporting more effective decision-making and operations management.

2. Create a trigger to ensure that the Sh_Charge in the shipment_details table is always greater than zero.

The purpose of this trigger is to maintain data consistency and accuracy in the shipment_details table by ensuring that the Sh_Charges is always greater than zero.

```
delimiter //

CREATE TRIGGER ensure_positive_shipment_charge
BEFORE INSERT ON shipment_details
FOR EACH ROW
BEGIN
    IF NEW.SD_Charges <= 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Shipment charge must be greater than zero';
    END IF;
END;
//

delimiter ;
```

 Justification: Enforcing the condition that Sh_Charge must be greater than zero is crucial for accurate representation of shipment charges. Allowing zero or negative charges could lead to financial discrepancies, inaccurate cost calculations, and potential loss of revenue for the logistics company.

 Trigger Logic: When a new row is inserted or an existing row is updated in the shipment_details table, the trigger activates. It checks if the Sh_Charge value is less than or equal to zero. If this condition is met, the trigger raises an error, preventing the insertion or update operation from completing successfully.

 Outcome: By implementing this trigger, the database ensures that all shipment charges adhere to the specified rule, maintaining consistency and accuracy in the data. Any attempt to violate the rule results in the trigger preventing the operation, thereby upholding data integrity and supporting reliable financial management within the logistics management system.

 Impact: This trigger provides an additional layer of validation and constraint enforcement, safeguarding the accuracy of shipment charge data and preventing potential financial discrepancies. It enhances the reliability and trustworthiness of the database, supporting accurate cost calculations, financial reporting, and decision-making processes within the logistics management system.

3. Create a trigger to ensure that the Amount in the payment_details table is always greater than zero.

The purpose of this trigger is to maintain data consistency and accuracy in the payment_details table by ensuring that the Amount is always greater than zero.

```
delimiter //

CREATE TRIGGER ensure_positive_payment_amount
BEFORE INSERT ON payment_details
FOR EACH ROW
BEGIN
    IF NEW.Amount <= 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Payment amount must be greater than zero';
    END IF;
END;
//

delimiter ;
```

 Justification: Enforcing the condition that Amount must be greater than zero is crucial for accurate representation of payment amounts. Allowing zero or negative amounts could lead to financial discrepancies, inaccurate financial records, and potential loss of revenue for the logistics company.

 Trigger Logic: When a new row is inserted or an existing row is updated in the payment_details table, the trigger activates. It checks if the Amount value is less than or equal to zero. If this condition is met, the trigger raises an error, preventing the insertion or update operation from completing successfully.

 Outcome: By implementing this trigger, the database ensures that all payment amounts adhere to the specified rule, maintaining consistency and accuracy in the data. Any attempt to violate the rule results in the trigger preventing the operation, thereby upholding data integrity and supporting reliable financial management within the logistics management system.

 Impact: This trigger provides an additional layer of validation and constraint enforcement, safeguarding the accuracy of payment amount data and preventing potential financial discrepancies. It enhances the reliability and trustworthiness of the database, supporting accurate financial reporting, analysis, and decision-making processes within the logistics management system.

Conclusion

In the conclusion section, you summarize the key findings and outcomes of the project. This section should highlight the significance of the database system in addressing the logistics management requirements and achieving the project objectives. You can also discuss any challenges encountered during the implementation process and propose recommendations for future improvements or enhancements.

Appendices

The appendices section can include additional information that supports or complements the main body of the report. This may include:

1. **SQL Scripts:** Provide the SQL scripts for creating tables, stored procedures, triggers, and any other database objects implemented in the project.
2. **Sample Queries:** Include sample queries used in the project, along with their results, to demonstrate the functionality and effectiveness of the database system.
3. **Data Models:** Include entity-relationship diagrams (ERDs) or other data models that illustrate the database schema and relationships among various entities.
4. **Screenshots:** Include screenshots of the database interface or application used to interact with the database, demonstrating its usability and user interface design.
5. **Test Cases:** Document test cases used to validate the functionality and performance of the database system, along with the test results.