**Chapter 1: Introduction**

**Aim:**

Design a database system that improves Swift Southern Railway's ability to manage train schedules, rolling stock, and goods transportation across the UK, ensuring regulatory compliance, operational efficiency, and improved delivery service. By maintaining an accurate record of the rail network infrastructure to assist in train routes, Managing the inventory of locomotives (Loco) and Freight Wagons, ensuring their best possible use, and keeping a record of the details of goods conveyed across the network, maintaining a history of consignments thereby Producing efficient train schedules that maximize the use of available locomotives, freight wagons, and crew members for the transportation of goods to companies.

**Objective:**

**Identify Entities and Relationships:** Identify Entities and Relationships the railway's operations, defining superclass and subclass entities to the data structure that accurately reflects business processes.

**Design an ER Diagram:** A detailed ER diagram that visually represents the entities, relationships, and constraints of the database, providing a layout for the relational model.

**Normalize Tables:** Apply normalization rules to the database tables to reduce data redundancy and improve data integrity, ensuring efficient data retrieval and updating processes.

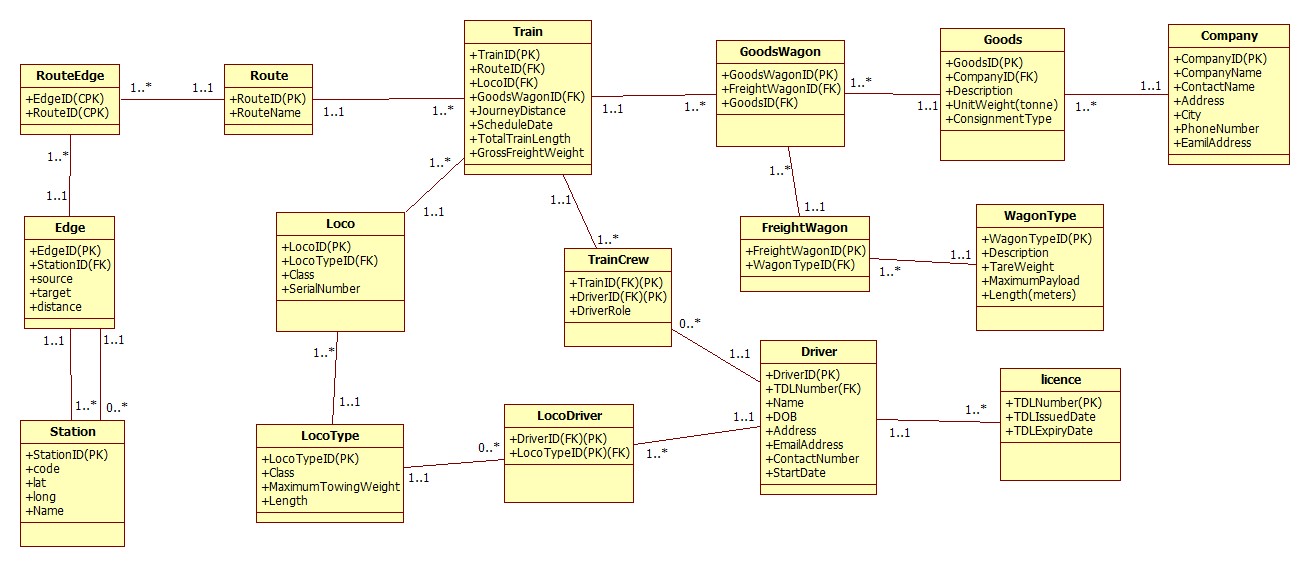
**Create a Data Model:** Design a data model that summarize the entities, attributes, and relationships for the railway’s goods transportation system.

**Implement the Database:** convert data model into a functional database system, ensuring that the implementation is done correctly according to the design and business requirements.

**Run queries:** Demonstrating queries for data retrieval using JOIN, AND, WHERE, commands.

**Chapter 2: The Class Diagram with constraints and assumptions**

**ER Diagram:**



**Assumptions:**

Each Station may have multiple Edges.

A Route must have one or more RouteEdge.

An Edge must connect one or more Stations.

A Train must be scheduled on one Route.

Each Loco may be assigned to only one Train.

A LocoType must be associated with at least one Loco.

Every Driver must hold at least one Licence.

A LocoDriver must pair a Driver with a LocoType they are qualified to operate.

Each Train may have multiple TrainCrew members (Driver and Co-driver).

A GoodsWagon can carry several Goods items.

Each FreightWagon must be of a specific WagonType.

A WagonType can be applicable to many FreightWagons.

Goods must be associated with one Company.

A Company can have many Goods.

A TrainCrew member may serve on several Trains.

Goods may be loaded onto different GoodsWagons.

**Chapter 3:**

**Design Decisions for Class Diagram:**

The class diagram for a database system, gathering data requirements for Swift Southern Railway's operations.

**Train Scheduling and Routing:** In the diagram the Train entity is important for managing train schedules. It's linked to the Route entity, indicating that each train is associated with a specific route. This relationship is important for planning and managing train movements. The Route entity is further broken down into RouteEdge and Edge, which represent part of the route and the specific paths between stations. This process is involved in planning efficient routes and ensuring that the rail network infrastructure is accurately represented, providing in optimal train routing. (Train) includes attributes like (ScheduleDate) and (TotalTrainLength), which are important for scheduling, planning, and resource allocation. On the other hand, (GoodsWagon) and (FreightWagon) were designed to summarize the specifics of rolling stock, such as weight and type, allowing inventory management.

**Rail Network Infrastructure:** The entities Route, RouteEdge, Edge, and Station appear to cover the rail network infrastructure, which can assist in train routing.

**Rolling Stock Management**: The diagram differentiates between locomotives (Loco), goods wagons (GoodsWagon), and freight wagons (FreightWagon), which enables the railway to manage different types of rolling stock inventory and allocate them as needed. Each locomotive and wagon type is linked to its respective type of entity (LocoType and WagonType), which suggests that the system can track specific attributes of the rolling stock, such as weight, and capacity. This level of detail supports the allocation of the right type of locomotives (Loco) and freight wagons (FreightWagon) to the right service, maximizing usage and efficiency.

**Goods Transportation:** The Goods entity represents the various items being transported, and it is related to the (GoodsWagon), indicating what goods are being carried by which wagon. This is essential for managing consignments and ensuring that goods are transported in compliance with regulatory requirements (e.g., weight limits). The connection between Goods and Company suggests that the system can track which goods are being transported for which company, providing improved delivery services to the company.

**Operational Efficiency:** The (LocoDriver) and (TrainCrew) entities include details about the staff operating the trains. This allows for the management of Drivers and ensures that trains are staffed appropriately, contributing to operational efficiency. The presence of a Licence entity suggests that the system tracks the necessary qualifications of the staff/Drivers.

Associations/relationships between classes were made. The one-to-many relationship between (Train) and (GoodsWagon), for instance, reflects the reality that one train can transport multiple wagons, yet a single wagon is bound to one train.

**Converting to Relational Model:**

Converting the class diagram into a relational model and to the constraints of a relational database. For instance, the (LocoDriver) foreign key in (TrainCrew) ensures the relational model retains the connection between a locomotive's driver and the crew, a link that is important for operations and tracking.

Normalizing the data to reduce redundancy and ensure the data does not duplicate. This involved breaking down classes like (Driver) into (LocoDriver) and (TrainCrew) to prevent duplicate information and to efficient the updating process. Such normalization is essential for maintaining a consistent and reliable database, especially in the context of a complex railway system.

During this change, specific constraints and indexes are required to ensure data uniqueness. Unique indexes were created for primary keys like (TrainID), and (DriverID), and for other tables for the uniqueness of each record and to make it easier for data retrieval.

**Station Table:** The Station table have the location information for each station within the railway network. The StationID serves as a unique identifier, while Code, Name, Lat, and Longitude provide a precise location and identification for each station. This table is for routes and scheduling as it shows the waypoints for train journeys.

**Edge Table**: The Edge table is important for mapping the railway network. Each record represents a direct link to tracks between two stations. It includes EdgeID as a primary key, references to StationID to link to the Station table, and details like source, target, and distance of the tracks. This is for calculating routes and distances.

**Route Table**: The Route table have the unique routes that trains can take. It is identified by RouteID and RouteName. This table showing the possibility of paths trains can follow on their schedules.

**RouteEdge Table**: This table is a centre point for combining EdgeID and RouteID to represent the edges that form a particular route. It is essential for handling the many-to-many relationship between routes and edges, thereby defining the specific path a train route can go.

**LocoType Table**: The LocoType table lists the different types of locomotives, each with a unique LocoTypeID. Attributes like Class, MaximumTowingWeight, and Length define the characteristics and capabilities of each locomotive type. This classification is important for assigning appropriate locomotives to various services, based on their towing capacity and size.

**Loco Table**: The Loco table have the details of individual locomotives, each identified by LocoID. It references LocoTypeID to indicate the type of locomotive and includes the Class and SerialNumber for additional identification. This table is for managing the locomotive inventory and maintenance records.

**Goods Table**: The Goods table records the details of items being transported, with GoodsID as the primary key. It links to the Company table through CompanyID, and includes attributes such as Description, UnitWeight, and ConsignmentType.

**Company Table**: The Company table contains information about companies that utilize the railway for goods transportation. It includes CompanyID, CompanyName, ContactName, Address, City, PhoneNumber, and EmailAddress for each company, which is essential for customer relations and service delivery.

**GoodsWagon Table**: This table have the lists of wagons that are specifically designed to carry goods, with GoodsWagonID as the primary key. It includes foreign keys to FreightWagonID and GoodsID, indicating the specific goods each wagon is carrying and linking back to the FreightWagon and Goods tables.

**WagonType Table**: The WagonType table classifies different types of wagons by WagonTypeID. Attributes like Description, TareWeight, MaximumPayload, and Length detail the specifications of each wagon type, which informs decisions on wagon allocation based on cargo needs.

**FreightWagon Table**: The FreightWagon table tracks individual freight wagons. Each wagon is identified by FreightWagonID and references WagonTypeID to indicate its classification. This enables the system to manage the freight wagon inventory and deployment effectively.

**Licence Table:** The Licence table stores information about the drivers’ licenses, with TDLNumber serving as the primary key, and includes TDLIssuedDate and TDLExpiryDate, which are crucial for ensuring compliance with regulatory requirements.

**Driver Table:** The Driver table maintains of all train drivers, with DriverID as the primary key. It references the Licence table through TDLNumber as foreign key and includes personal information and the StartDate of their employment.

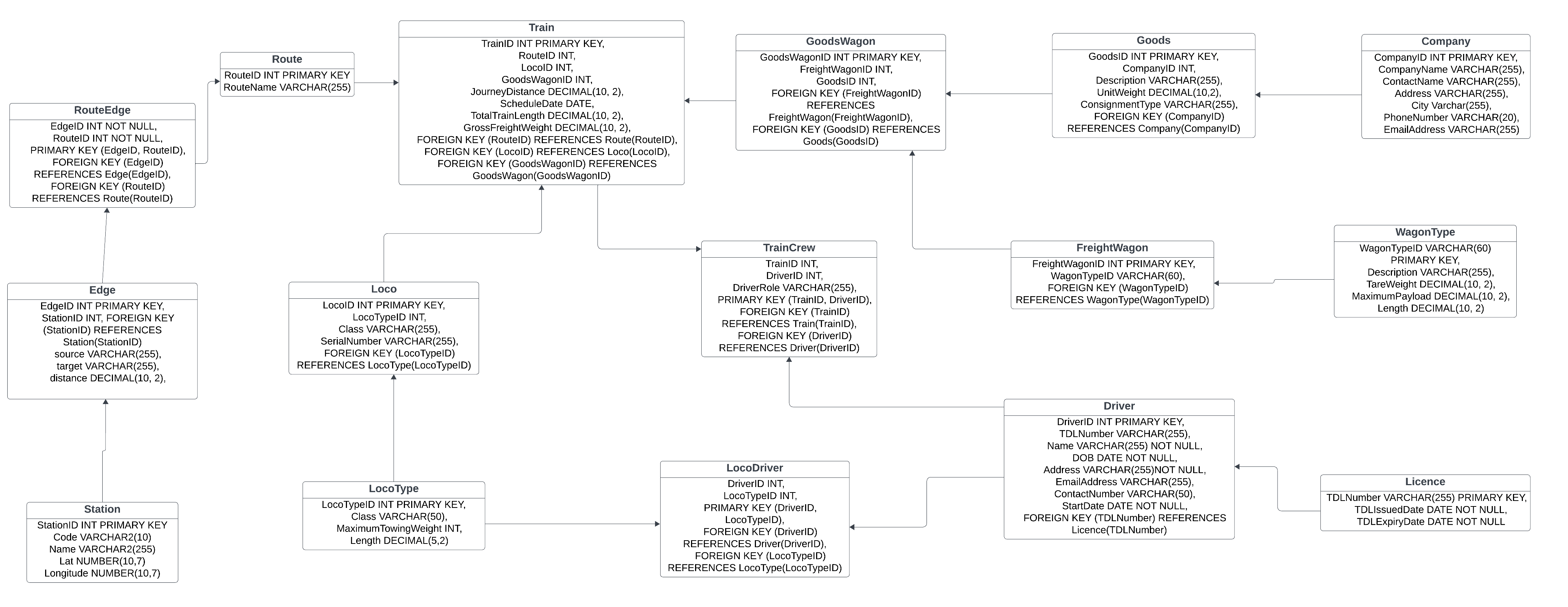
**LocoDriver Table:** The LocoDriver table represents the qualifications of drivers to operate different locomotive types, linking DriverID with LocoTypeID. This is essential for scheduling to ensure that drivers are matched with appropriate locomotives they are certified to operate.

**Train Table:** The Train table holds information about each train service, identified by TrainID. It includes foreign keys to RouteID, LocoID, and GoodsWagonID, as well as details like JourneyDistance, ScheduleDate, TotalTrainLength, and GrossFreightWeight.

**TrainCrew Table:** The TrainCrew table have the assignment of crew members to trains, connecting TrainID with DriverID and detailing their DriverRole. This ensures that each train has the necessary crew for each journey.

**Feedback Adjustments:**

In response to the feedback highlighted the overview of participation and cardinality in relationships, many revisions were made to the class diagram. Cardinality indicators were made to show into each association, correcting the oversight and ensuring a clear understanding of the multiple entities. The (Driver) class was corrected with attributes locomotive-type (LocoType) qualifications to address that the driver is qualified to drive. To correct the routes a (Route) class was implemented for more accurate picture within each journey from starting point to ending point of route. The implementation of a locomotive class to show a supertype/subtype led to the creation of a (RollingStock) superclass, from which came into (Loco) and (WagonType), providing a logical system that accurately shows their relationship. This restructuring also assists in the tracking of wagon allocation to trains and the assignment of goods to specific wagons for transportation, ensuring a diversified and precise representation within the relational model.

**Chapter 4: The Relational Model**

**Chapter 5: Implementation. Listing of SQL tables**

**SQL CREATE TABLE Statements**

Station

CREATE TABLE Station (

StationID INT PRIMARY KEY,

Code VARCHAR2(10),

Name VARCHAR2(255),

Lat NUMBER(10,7),

Longitude NUMBER(10,7)

);

Edge

CREATE TABLE Edge (

EdgeID INT PRIMARY KEY,

StationID INT,

source VARCHAR(255),

target VARCHAR(255),

distance DECIMAL(10, 2),

FOREIGN KEY (StationID) REFERENCES Station(StationID)

);

Route

CREATE TABLE Route (

RouteID INT PRIMARY KEY,

RouteName VARCHAR(255)

);

RouteEdge

CREATE TABLE RouteEdge (

EdgeID INT NOT NULL,

RouteID INT NOT NULL,

PRIMARY KEY (EdgeID, RouteID),

FOREIGN KEY (EdgeID) REFERENCES Edge(EdgeID),

FOREIGN KEY (RouteID) REFERENCES Route(RouteID)

);

LocoType

CREATE TABLE LocoType (

LocoTypeID INT PRIMARY KEY,

Class VARCHAR(50),

MaximumTowingWeight INT,

Length DECIMAL(5,2)

);

Loco

CREATE TABLE Loco (

LocoID INT PRIMARY KEY,

LocoTypeID INT,

Class VARCHAR(255),

SerialNumber VARCHAR(255),

FOREIGN KEY (LocoTypeID) REFERENCES LocoType(LocoTypeID)

);

Goods

CREATE TABLE Goods (

GoodsID INT PRIMARY KEY,

CompanyID INT,

Description VARCHAR(255),

UnitWeight DECIMAL(10,2),

ConsignmentType VARCHAR(255),

FOREIGN KEY (CompanyID) REFERENCES Company(CompanyID)

);

Company

CREATE TABLE Company (

CompanyID INT PRIMARY KEY,

CompanyName VARCHAR(255),

ContactName VARCHAR(255),

Address VARCHAR(255),

City Varchar(255),

PhoneNumber VARCHAR(20),

EmailAddress VARCHAR(255)

);

GoodsWagon

CREATE TABLE GoodsWagon (

GoodsWagonID INT PRIMARY KEY,

FreightWagonID INT,

GoodsID INT,

FOREIGN KEY (FreightWagonID) REFERENCES FreightWagon(FreightWagonID),

FOREIGN KEY (GoodsID) REFERENCES Goods(GoodsID)

);

WagonType

CREATE TABLE WagonType (

WagonTypeID VARCHAR(60) PRIMARY KEY,

Description VARCHAR(255),

TareWeight DECIMAL(10, 2),

MaximumPayload DECIMAL(10, 2),

Length DECIMAL(10, 2)

);

FreightWagon

CREATE TABLE FreightWagon (

FreightWagonID INT PRIMARY KEY,

WagonTypeID VARCHAR(60),

FOREIGN KEY (WagonTypeID) REFERENCES WagonType(WagonTypeID)

);

Licence

CREATE TABLE Licence (

TDLNumber VARCHAR(255) PRIMARY KEY,

TDLIssuedDate DATE NOT NULL,

TDLExpiryDate DATE NOT NULL

);

Driver

CREATE TABLE Driver (

DriverID INT PRIMARY KEY,

TDLNumber VARCHAR(255),

Name VARCHAR(255) NOT NULL,

DOB DATE NOT NULL,

Address VARCHAR(255)NOT NULL,

EmailAddress VARCHAR(255),

ContactNumber VARCHAR(50),

StartDate DATE NOT NULL,

FOREIGN KEY (TDLNumber) REFERENCES Licence(TDLNumber)

);

LocoDriver

CREATE TABLE LocoDriver (

DriverID INT,

LocoTypeID INT,

PRIMARY KEY (DriverID, LocoTypeID),

FOREIGN KEY (DriverID) REFERENCES Driver(DriverID),

FOREIGN KEY (LocoTypeID) REFERENCES LocoType(LocoTypeID)

);

Train

CREATE TABLE Train (

TrainID INT PRIMARY KEY,

RouteID INT,

LocoID INT,

GoodsWagonID INT,

JourneyDistance DECIMAL(10, 2),

ScheduleDate DATE,

TotalTrainLength DECIMAL(10, 2),

GrossFreightWeight DECIMAL(10, 2),

FOREIGN KEY (RouteID) REFERENCES Route(RouteID),

FOREIGN KEY (LocoID) REFERENCES Loco(LocoID),

FOREIGN KEY (GoodsWagonID) REFERENCES GoodsWagon(GoodsWagonID)

);

TrainCrew

CREATE TABLE TrainCrew (

TrainID INT,

DriverID INT,

DriverRole VARCHAR(255),

PRIMARY KEY (TrainID, DriverID),

FOREIGN KEY (TrainID) REFERENCES Train(TrainID),

FOREIGN KEY (DriverID) REFERENCES Driver(DriverID)

);

**Chapter 6: The six queries and output.**

**WORKSPACE:- 1946763**

**USERNAME:- k1946763@kingston.ac.uk**

**PASSWORD:- Hardiksinh@1**

**Query -1**



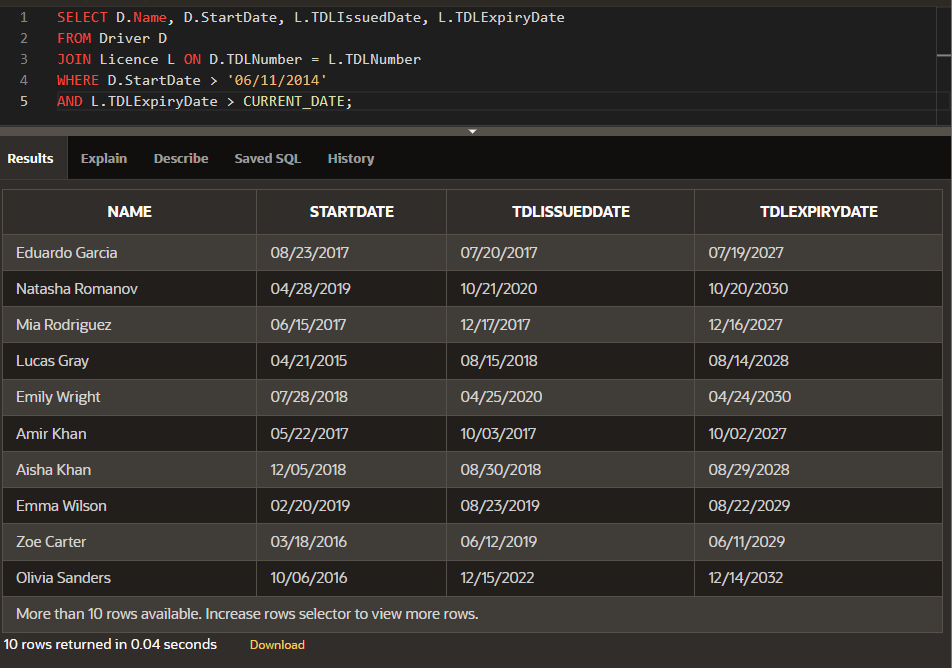
SELECT G.Description, C.CompanyName

FROM Goods G

JOIN Company C ON G.CompanyID = C.CompanyID

WHERE C.CompanyName = 'UK Widgets Ltd';

**Query -2**



SELECT D.Name, D.StartDate, L.TDLIssuedDate, L.TDLExpiryDate

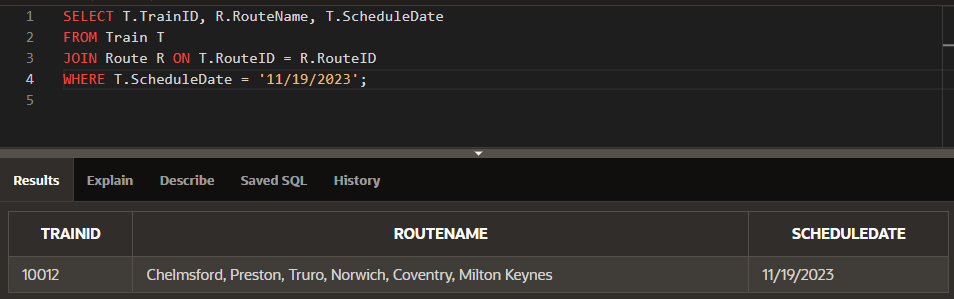
FROM Driver D

JOIN Licence L ON D.TDLNumber = L.TDLNumber

WHERE D.StartDate > '06/11/2014'

AND L.TDLExpiryDate > CURRENT\_DATE;

**Query -3**



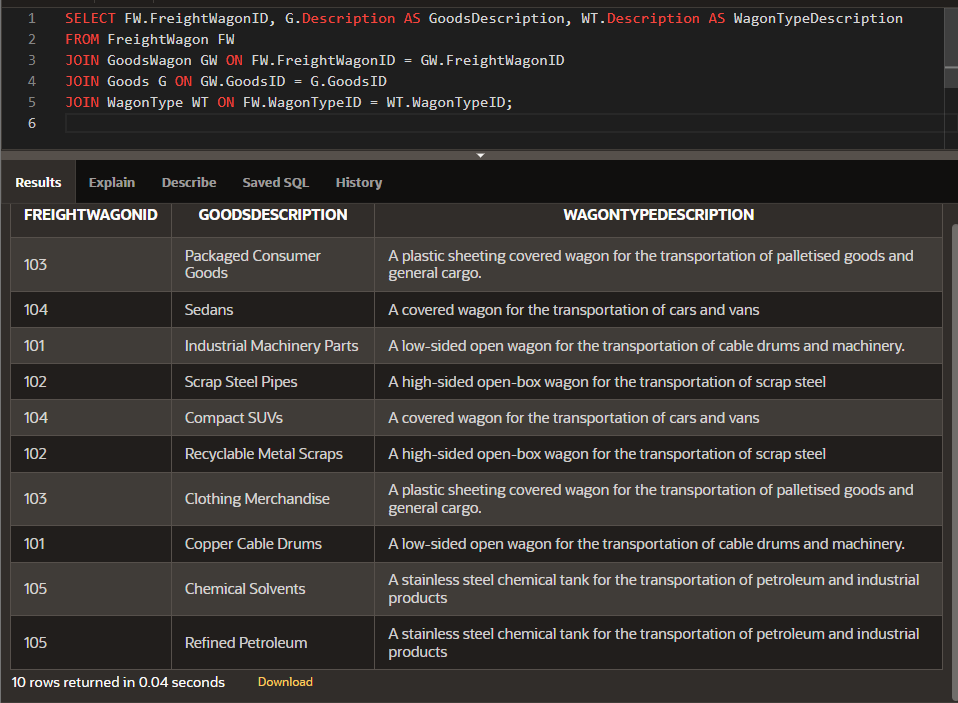
SELECT T.TrainID, R.RouteName, T.ScheduleDate

FROM Train T

JOIN Route R ON T.RouteID = R.RouteID

WHERE T.ScheduleDate = '11/19/2023';

**Query -4**



SELECT FW.FreightWagonID, G.Description AS GoodsDescription, WT.Description AS WagonTypeDescription

FROM FreightWagon FW

JOIN GoodsWagon GW ON FW.FreightWagonID = GW.FreightWagonID

JOIN Goods G ON GW.GoodsID = G.GoodsID

JOIN WagonType WT ON FW.WagonTypeID = WT.WagonTypeID;

**Query -5**

SELECT T.TrainID, LT.Class, TC.DriverRole, T.GrossFreightWeight

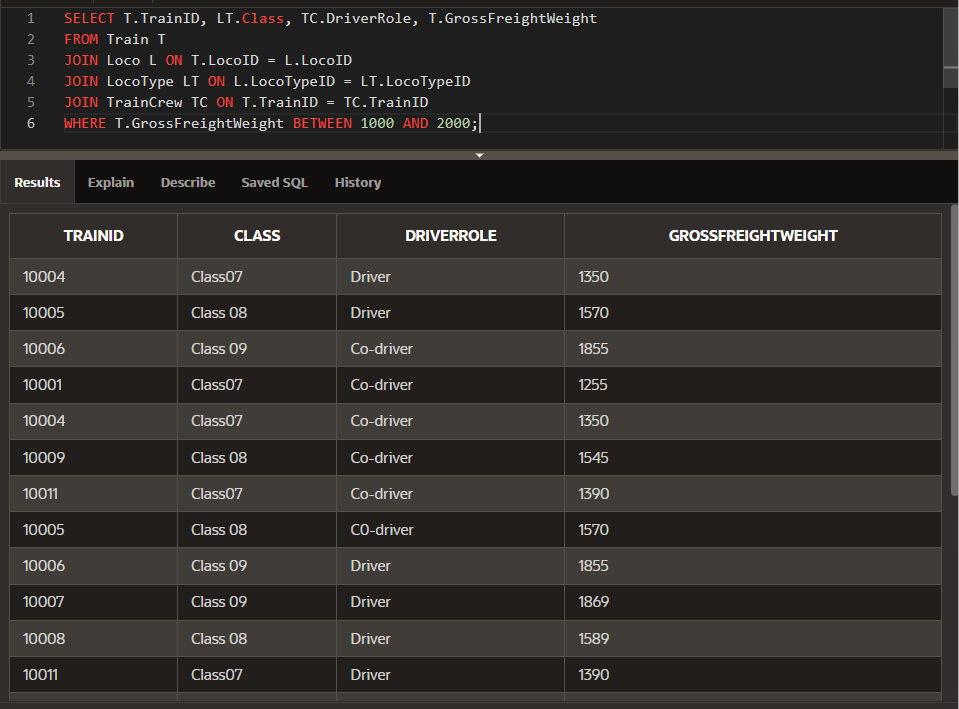
FROM Train T

JOIN Loco L ON T.LocoID = L.LocoID

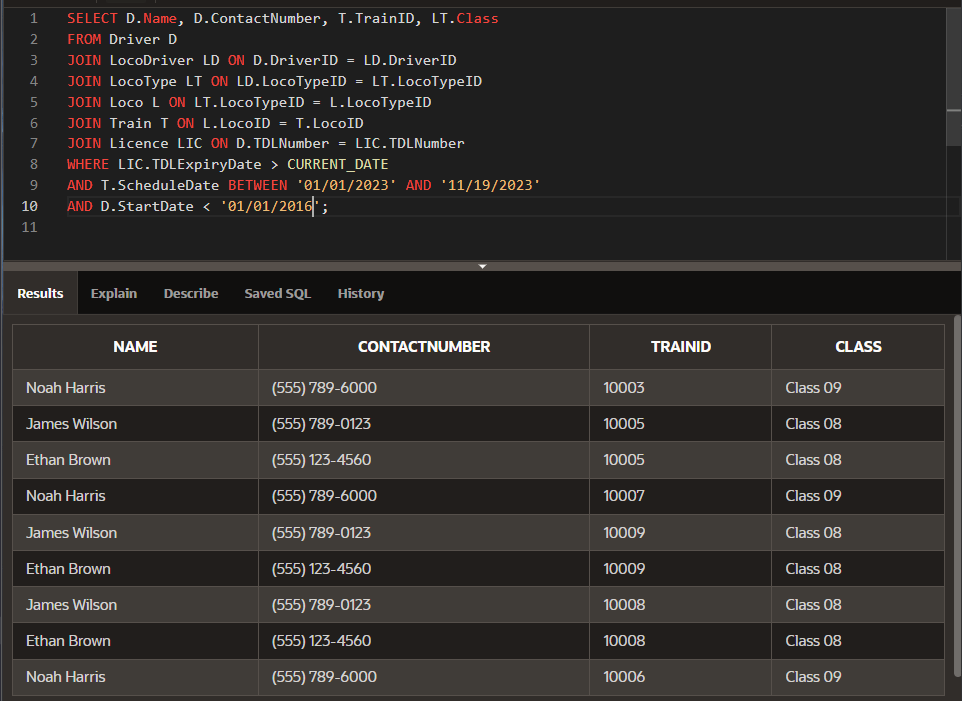
JOIN LocoType LT ON L.LocoTypeID = LT.LocoTypeID

JOIN TrainCrew TC ON T.TrainID = TC.TrainID

WHERE T.GrossFreightWeight BETWEEN 1000 AND 2000;



**Query -6**



SELECT D.Name, D.ContactNumber, T.TrainID, LT.Class

FROM Driver D

JOIN LocoDriver LD ON D.DriverID = LD.DriverID

JOIN LocoType LT ON LD.LocoTypeID = LT.LocoTypeID

JOIN Loco L ON LT.LocoTypeID = L.LocoTypeID

JOIN Train T ON L.LocoID = T.LocoID

JOIN Licence LIC ON D.TDLNumber = LIC.TDLNumber

WHERE LIC.TDLExpiryDate > CURRENT\_DATE

AND T.ScheduleDate BETWEEN '01/01/2023' AND '11/19/2023'

AND D.StartDate < '01/01/2016';

**Chapter 7: Conclusion**

This project is made with the specific goal of improving usage of rolling stock, train timetables, and goods transportation management in order to increase performance and improve services.

The starting point of the project was identifying the entities and relationships relevant to the Swift Southern Railway. This first step was important in deciding the structure of both the class diagram and the final relational model. Superclass and subclass entities were designed to appropriately reflect the business processes involved in railway management.

The ER diagram was designed to provide an overview of the database entities, relationships, and constraints. This diagram was used as a planning tool for the project.

Normalization of tables was done to reduce data duplication, resulting in a more efficient and accurate database. The normalized tables provided quicker data retrieval and updating processes, which improved the system's overall performance.

The relational data model was then created, summarising the attributes and relationships essential to the railway's goods transportation system. Each revision in the database design helps to improve the database system.

Implementation transformed through the relational data model into a functional database system and ensuring that the design was correctly reflected in the actual database and met all business needs. Then SQL queries were tested to demonstrate the system's capabilities, with JOIN, AND, WHERE commands used to retrieve data.

The project cycle, from design to implementation, was highlighted by a continuous feedback loop that allowed for iterative improvements. The class diagram was updated, including the addition of cardinality in relationships and the correction of route modelling.

In conclusion, the database system's outcome demonstrates the design and careful analysis that guided to completion of the project. The class diagram and relational model are efficient for Swift Southern Railway's goals by offering a complete structure for managing complex railway operations.