Marmara University Engineering Faculty
Department of Computer Engineering

CSE3055 – Database Systems Project Report

Database Assisted Metal Manufacturing Company Logistics Iteration #3

Instructor: Mustafa Ağaoğlu

January 1, 2024

STUDENTS

Şükrü Can MAYDA - 150120031 (GrRep)
Nurbetül ÇAKIR - 150121545
Musa ÖZKAN - 150121058
Ali ÇETİNKAYA - 150119606

1 Description

Our team embarks on a tailor-made database system project for our esteemed client, Renkli Metal Sanayi ve Ticaret A.Ş., a prominent player in the non-ferrous metal market. Specializing in a diverse inventory that includes brass, copper, stainless steel, plastics, and aluminum in multifarious shapes like rods (round, squared, hexagonal, and rectangular), tubes, and pipes, Renkli Metal meets the nuanced demands of various industries.

The database encapsulates a rich dataset of customer profiles, encompassing company names, addresses, contact numbers, and web details. This reflects the vibrant tapestry of entities populating Istanbul's metal sector, with records spanning across different professions, registration timelines, and financial forays. Our project, christened **"Database Assisted Metal Manufacturing Company Logistics"**, aspires to capture the dynamism and breadth of the regional metal commerce.

2 Project Scope

The project, *"Database Assisted Metal Manufacturing Company Logistics,"* aims to design and implement a comprehensive database system for Renkli Metal Sanayi ve Ticaret A.Ş. This system will facilitate efficient management of logistics operations involving non-ferrous metal products. The primary focus of the system is to streamline processes related to customer information, materials, orders, and transportation.

Objectives:

1. Develop a database system that centralizes and organizes information about the company, products, depots, orders, customers, shipments, and trucks.
2. Enable efficient order management, including order submission, calculation, prioritization, loading, and transportation routing.
3. Implement functionalities for logistics operations, such as route planning, truck allocation, and adherence to truck load restrictions.
4. Ensure system security through the implementation of 2FA for user account safety and data protection.
5. Design an intuitive user interface for internal (staff) and external (customer) users, focusing on ease of navigation and interaction.
6. Ensure system accessibility for users with disabilities, adhering to accessibility standards and guidelines.
7. Provide comprehensive training materials and documentation for system users to effectively utilize its functionalities.

3 Data and Requirement Analysis

Business Processes:

The system will adhere to various rules and constraints such as truck load restrictions, fixed delivery schedules on Tuesdays and Thursdays, order processing timelines, dispatch conditions based on truck capacity and schedule, and distance limitations for daily truck travel.

This analysis provides an overview of the project's scope, entities, business processes, functional and non-functional requirements, as well as business rules and constraints. The subsequent phases will involve system design, development, testing, implementation, and ongoing support to fulfill the outlined objectives.

The system will manage logistics operations, including:

- Route planning
- Truck allocation
- Order management and prioritization
- Order separation and loading

- Transportation routing
- Truck load consideration

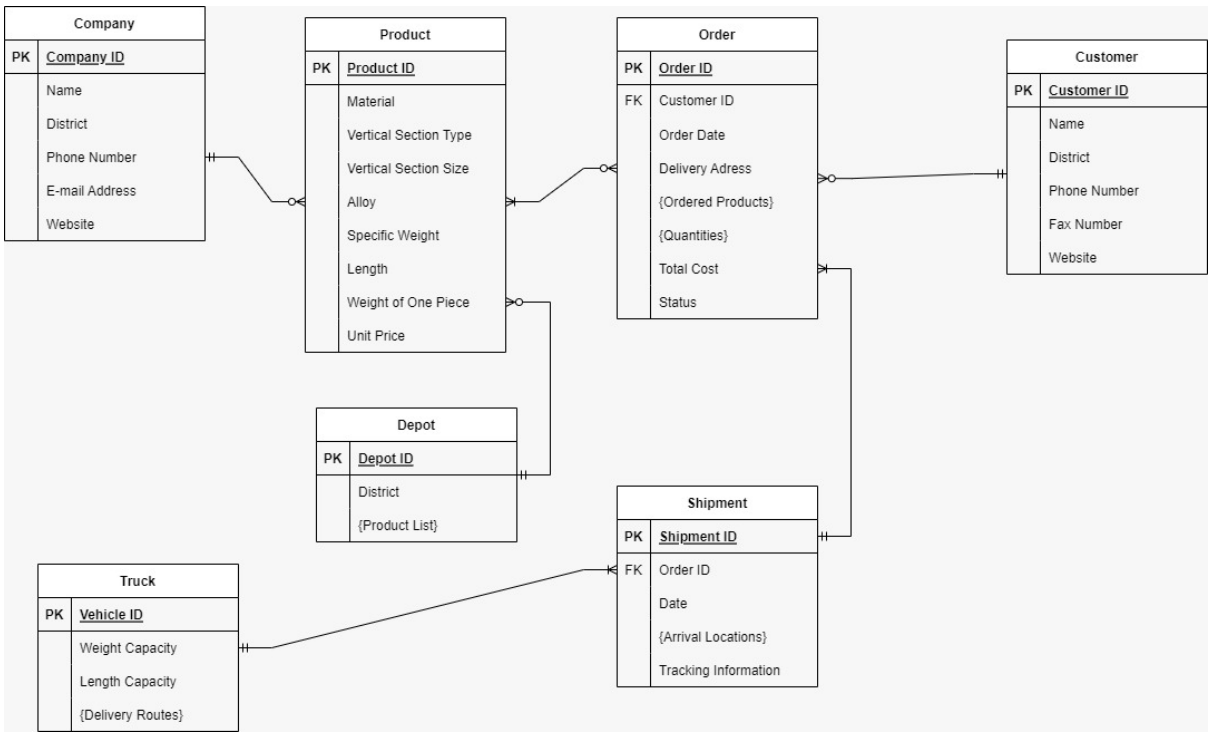
Functional Requirements:
The system will include functionalities for:

- Order submission and management by customers
- Calculation and prioritization of orders based on weight, length, and customer location
- Grouping orders by proximity and route optimization
- Truck loading optimization within predefined limits
- Selection of optimal routes considering distance and truck capacity
- Adherence to truck weight and length limits during loading and transportation

Non-functional Requirements:
The system will adhere to the following non-functional requirements:

- Security measures including 2FA for user account safety
- Intuitive and user-friendly UI for both internal and external users
- Accessibility standards for users with disabilities
- Comprehensive documentation and training materials for effective system utilization

4 Entity Relationship Diagram of Whole Database



4.1 Company

Table storing information about different companies. Includes details such as CompanyID, Name, District, PhoneNumber, MailAddress, and Website.
The Company ID is the primary key (PK), uniquely identifying records in the Company table. Also, Company table has 2 check constraint named checkID and checkCustomerID for checking CompanyID and CustomerID.

```sql
alter table Company
add constraint checkID check (CompanyID like '[0-9][0-9][0-9][0-9][0-9]');
```

100 %

Messages

```
Commands completed successfully.

Completion time: 2023-12-30T18:32:47.4947337+03:00
```

```sql
alter table Company
add constraint checkCustomerID check (CompanyID like '[0-9][0-9][0-9][0-9][0-9]');
```

00 %

Messages

```
Commands completed successfully.

Completion time: 2023-12-30T18:33:25.1933780+03:00
```

In addition to all of those, Company Table has 2 triggers named checkDate and checkPhoneNumber which controls if the Date and PhoneNumber attributes are in format.

```sql
Create Trigger check_Date on Shipment
for Insert
as

if(exists(  Select s.OrderID
            From Order_ o, Shipment s
            Where o.OrderID = s.OrderID and
            s.Date > o.OrderDate ))
Begin
    raiserror('order date cannot be later from shipment date',1,1)
    rollback transaction
End
```

00 %

Messages

```
Commands completed successfully.

Completion time: 2023-12-30T18:06:13.5735315+03:00
```

```sql
Create Trigger check_PhoneNumber on Company
for Insert
as

if(exists   (   Select co.CompanyID, cu.CustomerID
            From Company co , Customer cu
            Where co.PhoneNumber = cu.PhoneNumber
            )
  )
Begin
    raiserror('The companys phone number cannot be same with the customers phone number',1,1)
    rollback transaction
End
```

100 %

Messages

```
Commands completed successfully.

Completion time: 2023-12-30T16:52:40.5597258+03:00
```

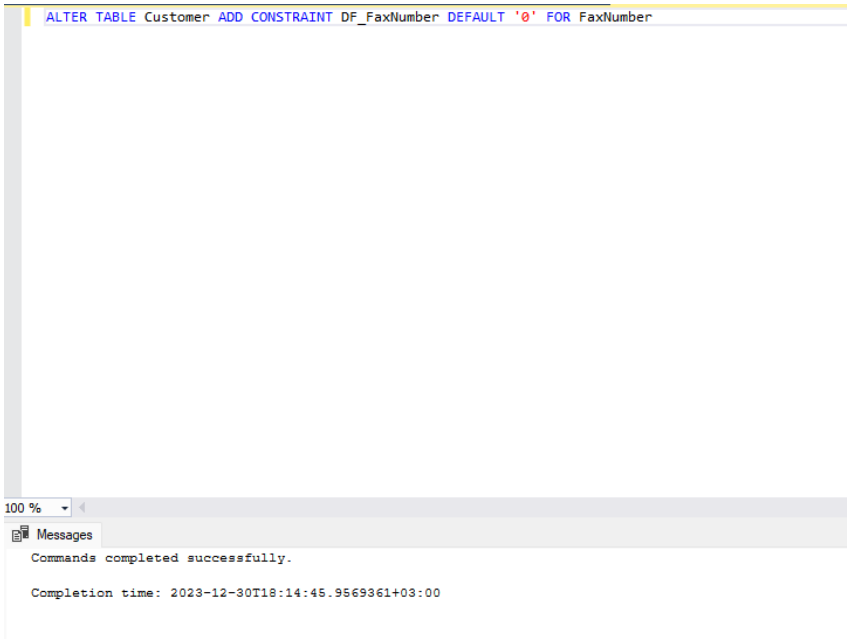
- **CompanyID:** Integer that uniquely identifies a company.
- **Name:** Name of the company, stored as a string (nvarchar) with a maximum length of 50 characters.
- **District:** District where the company is located, stored as a string (nvarchar) with a maximum length of 50 characters.
- **PhoneNumber:** Phone number of the company, stored as a string (nvarchar) with a length of 10 characters.
- **MailAddress:** Email address or mailing address of the company, stored as a string (nvarchar) with a maximum length of 50 characters.
- **Website:** Website URL of the company, stored as a string (nvarchar) with a maximum length of 50 characters.

CompanyID	Name	District	PhoneNumber	MailAddress	Website
int	nvarchar(50)	nvarchar(50)	nvarchar(10)	nvarchar(50)	nvarchar(50)

Table 1: Company Table Description

4.2 Customer

Table containing records of customers. It includes CustomerID, Name, District, PhoneNumber, FaxNumber, and Website.
The Customer ID is the primary key. Also, Customer table has a default constraint for its PhoneNumber and FaxNumber attributes.



- **CustomerID:** Integer that uniquely identifies a customer.
- **Name:** Name of the customer, stored as a string (nvarchar) with a maximum length of 100 characters.
- **District:** District where the customer is located, stored as a string (nvarchar) with a maximum length of 50 characters.
- **PhoneNumber:** Phone number of the customer, stored as a string (nvarchar) with a length of 11 characters.
- **FaxNumber:** Fax number of the customer, stored as a string (nvarchar) with a length of 11 characters.
- **Website:** Website URL of the customer, stored as a string (nvarchar) with a maximum length of 50 characters.

CustomerID	Name	District	PhoneNumber	FaxNumber	Website
int	nvarchar(100)	nvarchar(50)	nvarchar(11)	nvarchar(11)	nvarchar(50)

Table 2: Customer Table Description

4.3 Depot

Repository for depot-related data. Contains details about different depots, specifying DepotID and District.
The Depot ID is the primary key.

- **DepotID:** Integer that uniquely identifies a depot.
- **District:** District where the depot is located, stored as a string (nvarchar) with a maximum length of 50 characters.

DepotID	District
int	nvarchar(50)

Table 3: Depot Table Description

4.4 DepotProductList

Associates specific products with their respective depots. Contains DepotID, DepotProductID, and DepotProduct.

- **DepotID:** Integer referencing a specific depot.
- **DepotProductID:** Integer identifying a product in the depot.
- **DepotProduct:** Name or description of the product in the depot, stored as a string (nvarchar) with a maximum length of 50 characters.

DepotID	DepotProductID	DepotProduct
int	int	nvarchar(50)

Table 4: Depot Product List Table Description

4.5 Order_

Records orders made, consisting of OrderID, CustomerID, OrderDate, DeliveryAddress, TotalCost, and Status.
The Order ID is the primary key, and Customer ID is a foreign key (FK) pointing to the Customer table. Also, Order Table has a computed column named TotalCost which is a summation of UnitPrice and SpecificWeightInKG attributes from Product Table.

```
ALTER TABLE dbo.Order_ ADD TotalCost as cast([dbo].[Product].[UnitPrice] * [dbo].[Product].[SpecificWeightInKG] as int)
```

- **OrderID:** Integer that uniquely identifies an order.
- **CustomerID:** Integer referencing a specific customer associated with the order.
- **OrderDate:** Date when the order was placed.
- **DeliveryAddress:** Address where the order will be delivered, stored as a string (nvarchar) with a maximum length of 50 characters.
- **TotalCost:** Total cost of the order.
- **Status:** Status of the order, represented as a single character.

OrderID	CustomerID	OrderDate	DeliveryAddress	TotalCost	Status
int	int	date	nvarchar(50)	int	char(1)

Table 5: Order Table Description

4.6 Order_OrderedProducts

Contains the list of products included in different orders. It has OrderedProductID, OrderedProduct, and OrderID.

- **OrderedProductID:** Integer that uniquely identifies an ordered product.
- **OrderedProduct:** Name or description of the ordered product, stored as a string (nvarchar) with a maximum length of 50 characters.
- **OrderID:** Integer referencing a specific order associated with the product.

OrderedProductID	OrderedProduct	OrderID
int	nvarchar(50)	int

Table 6: Ordered Product Table Description

4.7 Order_Quantities

Stores information about quantities in different orders. Holds QuantityID, Quantity, and OrderID.

- **QuantityID:** Integer that uniquely identifies a quantity record.
- **Quantity:** Quantity description, stored as a string (nvarchar) with a maximum length of 30 characters.
- **OrderID:** Integer referencing a specific order associated with the quantity.

QuantityID	Quantity	OrderID
int	nvarchar(30)	int

Table 7: Quantity Table Description

4.8 Product

Information repository for various products offered. Includes ProductID, Material, VerticalSectionType, VerticalSectionSize, Alloy, SpecificWeightInKG, LengthInMeters, WeightOnePieceInKG, and UnitPrice. The Product ID is the primary key, uniquely identifying records in the Product table.

- **ProductID:** Identifier for a product, stored as a string (nvarchar) with a length of 20 characters.
- **Material:** Material of the product, stored as a string (nvarchar) with a length of 20 characters.
- **VerticalSectionType:** Type of vertical section, stored as a string (nvarchar) with a length of 20 characters.
- **VerticalSectionSize:** Size of the vertical section, stored as a string (nvarchar) with a length of 20 characters.
- **Alloy:** Alloy information of the product, stored as a string (nvarchar) with a length of 20 characters.
- **SpecificWeightInKG:** Specific weight of the product in kilograms.
- **LengthInMeters:** Length of the product in meters.
- **WeightOnePieceInKG:** Weight of one piece of the product in kilograms.
- **UnitPrice:** Price of the product per unit.

ProductID	Material	VerticalSectionType	VerticalSectionSize	Alloy	SpecificWeightInKG	LengthInMeters	WeightOnePieceInKG	UnitPrice
nvarchar(20)	nvarchar(20)	nvarchar(20)	nvarchar(20)	nvarchar(20)	float	int	float	int

Table 8: Product Table Description

4.9 Shipment

Keeps track of shipments, containing ShipmentID, OrderID, Date, and TrackingInformation.
The Shipment ID is the primary key, and Order ID is a foreign key linking to the Order table. Also, Shipment table has a unique constraint named Date.

```
ALTER TABLE Shipment ADD CONSTRAINT uniqueShipmentDate UNIQUE (Date)
```

100 %

Messages

```
Commands completed successfully.

Completion time: 2023-12-30T18:25:41.6965585+03:00
```

- **ShipmentID:** Integer that uniquely identifies a shipment.
- **OrderID:** Integer referencing a specific order associated with the shipment.
- **Date:** Date of the shipment.
- **TrackingInformation:** Tracking information related to the shipment, stored as a string (nvarchar) with a maximum length of 50 characters.

ShipmentID	OrderID	Date	TrackingInformation
int	int	date	nvarchar(50)

Table 9: Shipment Table Description

4.10 Shipment_ArrivalLocations

Stores details about the arrival locations associated with shipments. Includes ShipmentID, ArrivalLocationID, and ArrivalLocation.

- **ShipmentID:** Integer referencing a specific shipment.
- **ArrivalLocationID:** Integer identifying an arrival location associated with the shipment.
- **ArrivalLocation:** Description of the arrival location, stored as a string (nvarchar) with a maximum length of 20 characters.

ShipmentID	ArrivalLocationID	ArrivalLocation
int	int	nvarchar(20)

Table 10: Shipment Arrival Locations Table Description

4.11 Truck

Contains details of trucks or vehicles, specifying VehicleID, WeightCap, and LengthCap.
The Vehicle ID is the primary key.

- **VehicleID:** Integer that uniquely identifies a truck or vehicle.
- **WeightCap:** Weight capacity of the truck.
- **LengthCap:** Length capacity of the truck.

VehicleID	WeightCap	LengthCap
int	int	int

Table 11: Truck Table Description

4.12 Truck_DeliveryRoutes

Associates delivery routes with specific trucks. Holds VehicleID, DeliveryRouteID, and DeliveryRoute.

- **VehicleID:** Integer referencing a specific truck or vehicle.
- **DeliveryRouteID:** Integer identifying a delivery route associated with the truck.
- **DeliveryRoute:** Description of the delivery route, stored as a string (nvarchar) with a maximum length of 100 characters.

VehicleID	DeliveryRouteID	DeliveryRoute
int	int	nvarchar(100)

Table 12: Truck Delivery Routes Table Description

5 Views Created At The Database

5.1 show_Kadikoy

This view displays customer information specific to Kadikoy.

- **CustomerID:** Integer representing the customer identification number.
- **Name:** Name of the customer, stored as a string with a maximum length of 100 characters.
- **District:** Name of the district associated with the customer, stored as a string with a maximum length of 50 characters.
- **PhoneNumber:** Contact phone number of the customer, stored as a string with a length of 11 characters.

```sql
Create View show_Kadikoy as

Select CustomerID, Name, District, PhoneNumber
From Customer
Where District like '%KADIKÖY%'
```

Results | Messages

	CustomerID	Name	District	PhoneNumber
1	505110	3-A ALÜMİNYUM ALAŞIMLARI AŞ	KADIKÖY	2163025450
2	19660	ACERMAN ALÜMİNYUM SAN TİC LTD ŞTİ	KADIKÖY	2165081004
3	177666	ALUMİL EGE ALÜMİNYUM SAN TİC AŞ İSTANBUL ŞUBESİ	KADIKÖY	2163024182
4	144648	ALUTM ALÜMİNYUM PROFİL VE PLASTİK SANAYİ VE DIŞ T...	KADIKÖY	2167480722
5	204083	ARC ÇİNKO METAL VE KİMYA SAN TİC AŞ	KADIKÖY	5427953421
6	160359	ARITAŞ DIŞ TİCARET LTD ŞTİ	KADIKÖY	2163548618
7	236479	ASPEN METAL VE MADENCİLİK TİCARET LTD ŞTİ	KADIKÖY	5525596139
8	7112	BARAN MAKİNA SANAYİ VE DIŞ TİCARET LTD ŞTİ	KADIKÖY	2163499024
9	727140	BASIC PETROKİMYA LTD ŞTİ	KADIKÖY	2163803820
10	265776	BROMETA MATERIALS DIŞ TİCARET LTD ŞTİ	KADIKÖY	2167060207
11	395590	CT METAL SANAYİ İÇ VE DIŞ TİCARET LTD ŞTİ	KADIKÖY	2164642505
12	361369	ÇULCUOĞLU METAL ELEKTRONİK VE EL ALETLERİ SAN Tİ...	KADIKÖY	2164493657
13	751543	DOSAMET METAL DIŞ TİCARET LTD ŞTİ	KADIKÖY	2164051478
14	811793	ERDOĞANLAR İNŞAAT TURİZM SAN TİC LTD ŞTİ	KADIKÖY	2164646772
15	220021	GİZA MADEN TİCARET LTD ŞTİ	KADIKÖY	5314364688
16	960336	GRİDSAN MADENCİLİK KİMYA SAN TİC LTD ŞTİ	KADIKÖY	2163364554
17	233534	HEN METAL TİCARETİ LTD ŞTİ	KADIKÖY	2125229005
18	226634	LOGOS METAL ELEKTRİK MOBİLYA İNŞAAT SAN TİC LTD ...	KADIKÖY	NULL
19	2674	MAHMUT HALUK ORHAN	KADIKÖY	NULL
20	136261	MAT PROFİL VE ALÜMİNYUM KAPLAMA SAN TİC LTD ŞTİ	KADIKÖY	NULL
21	63085	MATKA PASLANMAZ DIŞ TİCARET AŞ	KADIKÖY	2163172919
22	199041	MEB METAL VE BİLEŞİKLERİ SAN TİC AŞ İSTANBUL KADIK...	KADIKÖY	2626417710
23	246827	METALTEK YAPI SİSTEMLERİ İMALAT VE TİCARET LTD ŞTİ	KADIKÖY	2162284217
24	467286	METHAN METAL ELEKTRİK TİCARET VE DANIŞMANLIK LT...	KADIKÖY	2164181668
25	450103	MOUNTAİN VİEW MİNİNG DEĞERLİTAŞ VE MADENCİLİK AŞ	KADIKÖY	2164111530
26	693934	OREKS MADENCİLİK LTD ŞTİ	KADIKÖY	2164502780
27	196558	ORHAN NİĞDELİOĞLU	KADIKÖY	NULL
28	694275	OR-METAL LTD ŞTİ	KADIKÖY	2164502780
29	8729	PERTAŞ METAL İNŞAAT TURİZM SAN TİC LTD ŞTİ	KADIKÖY	2163268159
30	223104	SİLVAN DIŞ VE İÇ TİCARET AŞ	KADIKÖY	2163803618

Query executed successfully.

CustomerID	Name	District	PhoneNumber
int	nvarchar(100)	nvarchar(50)	nvarchar(11)

Table 13: show_Kadikoy View Description

5.2 view_averageWeightOfPiece

This view provides the average weight of one piece of a product.

- **AverageWeightOfOnePiece:** Floating-point number representing the average weight of a single product piece.

```sql
Create View view_OnePieceKg as

Select ProductID, Material + VerticalSectionType as MaterialName, WeightOnePieceInKG, UnitPrice
From Product
Where WeightOnePieceInKG < 6.6 and WeightOnePieceInKG > 1
```

Results | Messages

	ProductID	MaterialName	WeightOnePieceInKG	UnitPrice
1	A10	ALÜMİNYUMALTIKÖŞE	1,89	160
2	A11	ALÜMİNYUMALTIKÖŞE	2,36	160
3	A12	ALÜMİNYUMALTIKÖŞE	3,16	160
4	A13	ALÜMİNYUMALTIKÖŞE	3,76	160
5	A14	ALÜMİNYUMALTIKÖŞE	4,76	160
6	A15	ALÜMİNYUMALTIKÖŞE	5,88	160
7	A23	PİRİNÇALTIKÖŞE	1,01	180
8	A24	PİRİNÇALTIKÖŞE	1,31	180
9	A25	PİRİNÇALTIKÖŞE	1,66	180
10	A26	PİRİNÇALTIKÖŞE	2,05	180
11	A27	PİRİNÇALTIKÖŞE	2,48	180
12	A28	PİRİNÇALTIKÖŞE	2,95	180
13	A29	PİRİNÇALTIKÖŞE	3,47	180
14	A30	PİRİNÇALTIKÖŞE	4,02	180
15	A31	PİRİNÇALTIKÖŞE	4,62	180
16	A32	PİRİNÇALTIKÖŞE	5,25	180
17	A33	PİRİNÇALTIKÖŞE	5,93	180
18	A6	ALÜMİNYUMALTIKÖŞE	1,1	160
19	A7	ALÜMİNYUMALTIKÖŞE	1,28	160
20	A8	ALÜMİNYUMALTIKÖŞE	1,47	160
21	A9	ALÜMİNYUMALTIKÖŞE	1,67	160
22	C117	BAKIRYUVARLAK	2,11	260
23	C118	BAKIRYUVARLAK	3,04	260
24	C119	BAKIRYUVARLAK	4,14	260
25	C120	BAKIRYUVARLAK	4,75	260
26	C121	BAKIRYUVARLAK	5,4	260
27	C122	BAKIRYUVARLAK	6,1	260
28	C146	DELRİNYUVARLAK	1,34	110
29	C147	DELRİNYUVARLAK	2,09	110
30	C148	DELRİNYUVARLAK	3,01	110
31	C149	DELRİNYUVARLAK	4,1	110
32	C150	DELRİNYUVARLAK	5,35	110
33	C180	KESTAMİTYUVARLAK	1,13	130
34	C181	KESTAMİTYUVARLAK	2,54	130
35	C182	KESTAMİTYUVARLAK	3,46	130
36	C183	KESTAMİTYUVARLAK	4,52	130

Query executed successfully.

AverageWeightOfOnePiece
float

Table 14: view_averageWeightOfPiece View Description

5.3 view_density

This view includes information related to product density.

- **ProductID:** Identification code for the product, stored as a string with a maximum length of 20 characters.
- **Material:** Material information for the product, stored as a string with a maximum length of 20 characters.
- **VerticalSectionType:** Type of vertical section associated with the product, stored as a string with a maximum length of 20 characters.
- **Densityx1000:** Density value for the product multiplied by 1000, stored as an integer.

```sql
Create View view_density as

Select ProductID, Material, VerticalSectionType, cast(SpecificWeightInKG as int) * 1000 / VerticalSectionSize as Densityx1000
From Product
Where Material like '%ALÜMİNYUM%' and VerticalSectionSize not like '%*%'
```

Results | Messages

	ProductID	Material	VerticalSectionType	Densityx1000
1	A1	ALÜMİNYUM	ALTIKÖŞE	333
2	A10	ALÜMİNYUM	ALTIKÖŞE	117
3	A11	ALÜMİNYUM	ALTIKÖŞE	105
4	A12	ALÜMİNYUM	ALTIKÖŞE	90
5	A13	ALÜMİNYUM	ALTIKÖŞE	83
6	A14	ALÜMİNYUM	ALTIKÖŞE	74
7	A15	ALÜMİNYUM	ALTIKÖŞE	66
8	A16	ALÜMİNYUM	ALTIKÖŞE	62
9	A17	ALÜMİNYUM	ALTIKÖŞE	55
10	A18	ALÜMİNYUM	ALTIKÖŞE	52
11	A19	ALÜMİNYUM	ALTIKÖŞE	48
12	A2	ALÜMİNYUM	ALTIKÖŞE	285
13	A3	ALÜMİNYUM	ALTIKÖŞE	250
14	A4	ALÜMİNYUM	ALTIKÖŞE	181
15	A5	ALÜMİNYUM	ALTIKÖŞE	166
16	A6	ALÜMİNYUM	ALTIKÖŞE	153
17	A7	ALÜMİNYUM	ALTIKÖŞE	142
18	A8	ALÜMİNYUM	ALTIKÖŞE	133
19	A9	ALÜMİNYUM	ALTIKÖŞE	125
20	C1	ALÜMİNYUM	YUVARLAK	200
21	C10	ALÜMİNYUM	YUVARLAK	30
22	C100	ALÜMİNYUM	YUVARLAK	14
23	C101	ALÜMİNYUM	YUVARLAK	13
24	C102	ALÜMİNYUM	YUVARLAK	12
25	C103	ALÜMİNYUM	YUVARLAK	11
26	C104	ALÜMİNYUM	YUVARLAK	10
27	C105	ALÜMİNYUM	YUVARLAK	10
28	C106	ALÜMİNYUM	YUVARLAK	9
29	C107	ALÜMİNYUM	YUVARLAK	9
30	C108	ALÜMİNYUM	YUVARLAK	8
31	C109	ALÜMİNYUM	YUVARLAK	8
32	C11	ALÜMİNYUM	YUVARLAK	26
33	C110	ALÜMİNYUM	YUVARLAK	8
34	C111	ALÜMİNYUM	YUVARLAK	7
35	C112	ALÜMİNYUM	YUVARLAK	7
36	C113	ALÜMİNYUM	YUVARLAK	7
37	C114	ALÜMİNYUM	YUVARLAK	6
38	C115	ALÜMİNYUM	YUVARLAK	5

ProductID	Material	VerticalSectionType	Densityx1000
nvarchar(20)	nvarchar(20)	nvarchar(20)	int

Table 15: view_density View Description

5.4 view_OnePieceKg

This view displays details regarding the weight and unit price of a single piece of a product.

- **ProductID:** Identification code for the product, stored as a string with a maximum length of 20 characters.
- **MaterialName:** Name of the material related to the product, stored as a string with a maximum length of 20 characters.
- **WeightOnePieceInKG:** Floating-point number representing the weight of a single product piece in kilograms.
- **UnitPrice:** Integer value denoting the unit price of a single product piece.

```sql
Create View view_averageWeightOfPiece as

Select Avg(WeightOnePieceInKG) as AverageWeightOfOnePiece
From Product
```

Results | Messages

	AverageWeightOfOnePiece
1	118,684395604396

ProductID	MaterialName	WeightOnePieceInKG	UnitPrice
nvarchar(20)	nvarchar(20)	float	int

Table 16: view_OnePieceKg View Description

6 Triggers Created At The Database

We have 2 triggers named in our Database as "checkDate" and "checkPhoneNumber".

6.1 checkDate

Objective: This trigger aims to prevent the insertion of a new record into the Company table if the inserted phone number matches any existing phone number in the Customer table.

Event: Triggered after an `INSERT` operation on the Company table (`FOR INSERT`).

Logic:

- It checks if there exists a matching `PhoneNumber` between the Company and Customer tables.
- If such a match is found, it raises an error with the message "The company's phone number cannot be the same as the customer's phone number" using `RAISEERROR`.
- It rolls back the transaction using `ROLLBACK TRANSACTION` to prevent the insertion of conflicting data.

```
Create Trigger check_Date on Shipment
for Insert
as

if(exists(  Select s.OrderID
            From Order_ o, Shipment s
            Where o.OrderID = s.OrderID and
            s.Date > o.OrderDate ))
Begin
    raiserror('order date cannot be later from shipment date',1,1)
    rollback transaction
End
```

```
00 %
Messages
Commands completed successfully.

Completion time: 2023-12-30T18:06:13.5735315+03:00
```

6.2 checkPhoneNumber

Objective: This trigger seems to have the same logic and functionality as the `check_PhoneNumber` trigger, but it's named differently (`check_table`) and operates on the Company table.

Event: Triggered after an `INSERT` operation on the Company table (`FOR INSERT`).

Logic:

- Similar to the `check_PhoneNumber` trigger, it checks for a matching `PhoneNumber` between the Company and Customer tables.
- If a match exists, it raises the same error message and rolls back the transaction.

```
Create Trigger check_PhoneNumber on Company
for Insert
as

if(exists   (   Select co.CompanyID, cu.CustomerID
            From Company co , Customer cu
            Where co.PhoneNumber = cu.PhoneNumber
            )
  )
Begin
    raiserror('The companys phone number cannot be same with the customers phone number',1,1)
    rollback transaction
End
```

```
100 %
Messages
Commands completed successfully.

Completion time: 2023-12-30T16:52:40.5597258+03:00
```

7 Stored Procedures Created At The Database

7.1 delete_customer

This stored procedure deletes a customer from the Customer table based on the provided CustomerID.

```
Create Procedure delete_customer(@CustomerID    int)
as

Begin

    Delete From Customer
    Where CustomerID=@CustomerID

End
```

```
100 %
Messages
Commands completed successfully.

Completion time: 2023-12-30T18:35:27.8719948+03:00
```

7.2 delete_Product

This procedure deletes a product from the Product table based on the provided ProductID.

```sql
Create Procedure new_customer ( @CustomerID     int,
                                @Name           nvarchar(100),
                                @District       nvarchar(50),
                                @PhoneNumber    nvarchar(11),
                                @FaxNumber      nvarchar(11),
                                @Website        nvarchar(50))
As

Begin

    SET IDENTITY_INSERT Customer ON

    Insert Into Customer (  customerID,
                            Name,
                            District,
                            PhoneNumber,
                            FaxNumber,
                            Website)
    Values( @CustomerID     ,
            @Name           ,
            @District       ,
            @PhoneNumber    ,
            @FaxNumber      ,
            @Website        )


    SET IDENTITY_INSERT Customer OFF

End
```

00 %

Messages

```
Commands completed successfully.

Completion time: 2023-12-30T18:40:36.2665929+03:00
```

7.3 new_customer

This procedure adds a new customer to the Customer table. It requires input parameters such as CustomerID, Name, District, PhoneNumber, FaxNumber, and Website.

```sql
Create Procedure new_Product (  @ProductID              nvarchar(20),
                                @Material               nvarchar(20),
                                @VerticalSectionType    nvarchar(20),
                                @VerticalSectionSize    nvarchar(20),
                                @Alloy                  nvarchar(20),
                                @SpecificWeightInKG     float,
                                @LengthInMeters         int,
                                @WeightOnePieceInKG     float,
                                @UnitPrice              int)
As

Begin

    SET IDENTITY_INSERT Customer ON

    Insert Into Product (   ProductID               ,
                            Material                ,
                            VerticalSectionType ,
                            VerticalSectionSize ,
                            Alloy                   ,
                            SpecificWeightInKG      ,
                            LengthInMeters          ,
                            WeightOnePieceInKG      ,
                            UnitPrice                 )

    Values(  @ProductID             ,
             @Material              ,
             @VerticalSectionType   ,
             @VerticalSectionSize   ,
             @Alloy                 ,
             @SpecificWeightInKG    ,
             @LengthInMeters        ,
             @WeightOnePieceInKG    ,
             @UnitPrice               )


    SET IDENTITY_INSERT Customer OFF

End
```

```
%
Messages
Commands completed successfully.

Completion time: 2023-12-30T18:46:54.6276907+03:00
```

7.4 new_Product

This procedure adds a new product to the Product table. It requires input parameters such as ProductID, Material, VerticalSectionType, VerticalSectionSize, Alloy, SpecificWeightInKG, LengthInMeters, WeightOnePieceInKG, and UnitPrice.

```
Create Procedure delete_Product(@productID  nvarchar(20))
as

Begin

    Delete From Product
    Where ProductID=@productID

End
```

100 %

Messages

```
Commands completed successfully.

Completion time: 2023-12-30T18:48:23.7933871+03:00
```

7.5 total_lengthCap

This procedure calculates the total length capacity of all trucks by summing up the LengthCap column values from the Truck table.

```
Create Procedure total_lengthCap
as
Begin

    Select Sum(LengthCap) as TotalLengthCapacityOfTrucks
    From Truck

End
```

```
00 %
Messages
Commands completed successfully.

Completion time: 2023-12-30T18:50:52.4455319+03:00
```

7.6 total_weightCap

This procedure calculates the total weight capacity of all trucks by summing up the WeightCap column values from the Truck table.

```
Create Procedure total_weightCap
as
Begin

    Select Sum(WeightCap) as TotalWeightCapacityOfTrucks
    From Truck

End
```

00 %

Messages

```
Commands completed successfully.

Completion time: 2023-12-30T18:51:48.1952416+03:00
```

7.7 update_FaxNumber

This procedure updates the FaxNumber of a customer in the Customer table based on the provided CustomerID.

```
Create Procedure update_unitPrice(  @ProductID          nvarchar(20),
                                    @UnitPrice          int)
as

Begin

    Update  Product
    Set     unitPrice = @UnitPrice
    Where   ProductID = @ProductID

End
```

.00 %

Messages

```
Commands completed successfully.

Completion time: 2023-12-30T18:54:17.8112576+03:00
```

7.8 update_PhoneNumber

This procedure updates the PhoneNumber of a customer in the Customer table based on the provided CustomerID.

```
Create Procedure update_PhoneNumber(@CustomerID        int,
                                    @PhoneNumber       nvarchar(11))
as

Begin

    Update  Customer
    Set     PhoneNumber = @PhoneNumber
    Where   CustomerID = @CustomerID

End
```

100 %

Messages

```
Commands completed successfully.

Completion time: 2023-12-30T18:56:21.8695544+03:00
```

7.9 update_TrackingInformation

This procedure updates the TrackingInformation in the Shipment table based on the provided ShipmentID.

```
Create Procedure update_FaxNumber(  @CustomerID        int,
                                    @FaxNumber         nvarchar(11))
as

Begin

    Update  Customer
    Set     FaxNumber = @FaxNumber
    Where   CustomerID = @CustomerID

End
```

100 %

Messages

```
Commands completed successfully.

Completion time: 2023-12-30T18:57:15.9050726+03:00
```

7.10 update_unitPrice

This procedure updates the UnitPrice of a product in the Product table based on the provided ProductID.

```
Create Procedure update_Website(    @CustomerID         int,
                                    @Website            nvarchar(50))
as

Begin

    Update  Customer
    Set     Website     = @Website
    Where   CustomerID  = @CustomerID

End
```

100 %

Messages

```
Commands completed successfully.

Completion time: 2023-12-30T18:58:26.9780347+03:00
```

7.11 update_Website

This procedure updates the Website of a customer in the Customer table based on the provided CustomerID.

```
Create Procedure update_TrackingInformation(    @ShipmentID             int,
                                                @TrackingInformation    nvarchar(50))
as

Begin

    Update  Shipment
    Set     TrackingInformation = @TrackingInformation
    Where   ShipmentID = @ShipmentID

End
```

100 %

Messages

```
Commands completed successfully.

Completion time: 2023-12-30T19:00:56.0645628+03:00
```