CDA – 502 DATABASE MANAGEMENT SYSTEMS

PROJECT PAPER

INTEGRATED SUPPLY CHAIN AND FINANCIAL MANAGEMENT SYSTEM

Team-13 Sec
on – S4

Prepared By:

Jairam Marndi (50496788)

(Developer and Co-Researcher):

- Technical support and analysis of database design.
- Coordinates meetings based on others' availability and monitors progress.

Haarika Polukanti (50495776)

(Developer and Co-Researcher):

- Verifies business requirements and collaborates with the professor.
- Ensures the quality of the sprint.

Jagadeesh Kamara (50495713)

(Developer and Co-Researcher):

- Focuses on detailed documentation.

Sai Chandan Vuda (50495338)

(Developer and Co-Researcher):

- Version controls the development.

Problem Statement:

Given that your company operates with multiple functional areas and manages various stakeholders for development and production, including manufacturing, supply chain, vendor details, raw material specifics, sales records, and invoice processing, there is a clear need for a well-structured centralized system. We propose the implementation of an effective and dependable database management system for the storage and retrieval of this critical information.

Proposed Solution:

Our goal is to deliver a solid database management system [2] that can effectively store and access data from several domains like Employee data, Department, Product details, Production line information, Vendor details, Warehouse details and Invoice information in real-time. Complex structures will be handled efficiently where multiple stake holders are closely connected. Data security will be ensured via access permissions that are role based and customized to functional areas so that there will not be distinctive roles. To protect present functionality against changes in the future, we strive to establish programmed data independence. This idea facilitates scalability, improves adaptation to shifting company needs, preserves data integrity, and generates long-term cost savings. Informed decision-making will also be made easier with the system's assistance in data summary.

Design approach:

We plan to use the Agile technique, enabling us to work on several projects at once and ensuring your active involvement at each level of functionality development. As part of Agile we will:

- Iterative technique: Agile is a technique that divides the project into shorter, more manageable iterations or sprints, usually lasting between two and four weeks.
- Customer-Centric: It puts the needs of the customer first and values their input and collaboration to make sure the product fulfils those demands and changes as necessary.
- Flexibility: Throughout the project, agile adjusts to shifting priorities and requirements, encouraging responsiveness and agility.
- Agile teams strive to offer a usable product piecemeal, enabling early and frequent releases.
- Core Frameworks: There are several popular Agile frameworks, including Scrum, Kanban, and Extreme Programming (XP), each with its own set of guidelines and practices. Scrum will be preferred because of its structured framework, iterative process, and focus on producing value quickly. It is a popular option for Agile software development because it empowers teams, encourages openness, and develops a culture of continuous improvement.

We'll be able to technically improve certain aspects with your continued support and insightful input. We will define deadlines once we have reached a shared understanding, and this collaborative approach ensures that you can track our development.

About Repository:

All business rules and metadata were included in the repository, which acted as the project's foundation and main source of information. While the metadata described the qualities and attributes of each entity's stored data, including defining allowed values to enhance dataviabiliy, the customer supplied the business rules that guided our rule design. Before finalizing, we held in-depth conversations to build the repository, confirming alignment with your requirements and technological viability.

MetaData (Repositories)

Product:

Data Item Name	**Data Item Type**	**Data Item Length**	**MetaData Minimum**	**MetaData Maximum**	**Data Source**
ProductNumber	Varchar2	5	5	5	Product
ProductType	Varchar2	15	3	15	Product
ProductionName	Varchar2	20	3	20	Product
Price	Number	2	1	2	Product
Cost	Number	2	1	2	Product
Color	Varchar2	6	3	6	Product
Weight (lbs)	Number	3	1	3	Product
DesignerID	Varchar2	4	4	4	Line Employee

Description:

Product table gives information about the product number, name and its type. It demonstrates the price, cost, color and weight (lbs.) of the product. In this product table we are choosing ProductNumber as a Primary Key. It is the unique identifier of the Product. Product Type is the categorization of the product. Production Name is the name or label associated with the production.Price is the actual price of the product whereas Cost defined as the cost associated with producing the product.Color and Weight (lbs.) describes the physical appearance of the product .DesignerID gives information about the designer who designed that product.

ProductLine :

Data Item Name	**DataItem Type**	**Data Item Length**	**MetaData Minimum**	**Metadata Maximum**	**MetaData Source**
LineNumber	Varchar2	4	4	4	ProductLine
LineCapacity	Number	3	3	3	ProductLine
PhoneNumber	Number	10	10	10	ProductLine
StreetName	Varchar2	40	30	40	ProductLine
Area	Varchar2	20	10	20	ProductLine
State	Varchar2	2	2	2	ProductLine
ZipCode	Number	5	5	5	ProductLine

Description:

ProductLine describes about the line number, line capacity of the Product. It also describes about where the product is created. LineNumber is the identifier of the product line. Capacity of the product line, represented as a numerical value. Phone number associated with the product line. StreetName gives Name of the street where the product line is located. Area gives information about the Area or region associated with the product line. State is State abbreviation where the product line is located. Zip code provides the zip code of the product line location.

Warehouse:

DataItem Name	**Data Item Type**	**Data Item Length**	**MetaData Minimum**	**Metadata Maximum**	**MetaData Source**
WarehouseNumber	Varchar2	5	5	5	Warehouse
StreetName	Varchar2	30	15	30	Warehouse
City	Varchar2	20	10	20	Warehouse
PhoneNumber	Varchar2	10	10	10	Warehouse

Description:

This provides information about the data items, their types, lengths, metadata constraints, and the source of the data within the context of the "Warehouse" data source. WarehouseNumber -Unique identifier for each warehouse. StreetName gives the Name of the street where the warehouse is located. City states about where the warehouse is situated. Phone number - Phone number associated with the warehouse.

Prod_Raw Material:

DataItem Name	Data Item Type	Data Item Length	MetaData Minimum	Metadata Maximum	MetaData Source
Production Number	Varchar2	5	5	5	Raw Material
RawMaterial Name	Varchar2	15	5	15	Raw Material

Description: There are two attributes in the Prod_Raw Material table. They are ProductionNumber and RawMaterialName, both are primary keys in this tables.

Employee:

DataItem Name	Data Item Type	Data Item Length	MetaData Minimum	Metadata Maximum	MetaData Source
EmployeeID	Varchar2	4	4	4	Employee
FirstName	Varchar2	20	10	20	Employee
LastName	Varchar2	20	10	20	Employee
Position	Varchar2	20	10	20	Employee
Salary	Number	6	5	6	Employee
WorkCategory	Varchar2	20	10	20	Employee
FieldOfWork	Varchar	1	1	1	

Description:

Employee table explains about the details of employee. In this table EmployeeID is the identifier for each employee. Employee table gives information about first name, last name, salary and position of the employee. It also explains the type of work in which employee is engaged in and the category of work it belongs.

Line_Employee:

Data Item Name	**Data Item Type**	**Data Item Length**	**MetaData Minimum**	**Metadata Maximum**	**MetaData Source**
EmployeeID	Varchar2	4	4	4	Employee
LineNumber	Varchar2	4	4	4	Production Line

Description: This table will have two attributes. Employee table is generalized and divided into three unique tables. This is one among them. EmployeeID is the primary key and LineNumber is the foreign Key

WarehouseEmployee:

DataItem Name	**Data Item Type**	**Data Item Length**	**MetaData Minimum**	**Metadata Maximum**	**MetaData Source**
EmployeeID	Varchar2	4	4	4	Employee
Warehouse Number	Varchar2	5	5	5	Employee

Description:

This. Table contains two attributes. EmployeeID is the primary key and WarehouseNumber is the foreign key. This table is extracted from Employee table and unique and employees of warehouse.

Department_Employee:

DataItem Name	Data Item Type	Data Item Length	Meta Data Minimum	Metadata Maximum	MetaData Source
EmployeeID	Varchar2	4	4	4	Employee
DepartmentID	Varchar2	4	4	4	Employee

Description:

This. Table contains two attributes. EmployeeID is the primary key and DepartmentNumber is the foreign key. This table is extracted from Employee table and unique and employees of Department

Department:

Data Item Name	Data Item Type	Data Item Length	MetaData Minimum	Metadata Maximum	MetaData Source
DepartmentID	Varchar2	4	4	4	Department
Department Name	Varchar2	20	10	20	Department
Location	Varchar2	30	15	30	Department
NumberoOfEmployees	Number	1	1	1	Department
PhoneNumber	Varchar2	10	10	10	Department

Description:

This table explains about the departments involved. DepartmentID is the Unique identifier for each department. Department Name is the Name or label of the department. Location is Physical location or address of the department. This table give information of number of employees in the department and the contact number of each department

Invoice:

Data Item Name	**Data Item Type**	**Data Item Length**	**MetaData Minimum**	**Metadata Maximum**	**MetaData Source**
Invoice Number	Varchar 2	7	7	7	Invoice
TotalAm ount	Number	4	3	4	Invoice

Description:

This table gives information about the invoice and the total amount on the invoice. Invoice Number is the primary key.

Vendor:

DataItem Name	**Data Item Type**	**Data Item Length**	**MetaData Minimum**	**Metadata Maximum**	**MetaData Source**
VendorID	Varchar2	**4**	**4**	**4**	Vendor
VendorName	Varchar2	**30**	**10**	**30**	Vendor
StreetName	Varchar2	**30**	**3**	**30**	Vendor
City	Varchar2	**20**	**3**	**20**	Vendor
PhoneNumber	Number	**10**	**10**	**10**	Vendor

Description:

This table gives information about the Vendor details. Here VendorID is the identifier of the Vendor. It gives the address of Vendor location. It gives information about the name and contact number of the vendor who sends product.

Vend_Payment:

Data Item Name	**Data Item Type**	**Data Item Length**	**MetaData Minimum**	**Metadata Maximum**	**MetaData Source**
InvoiceNumber	Varchar2	7	7	7	Invoice
VendorNumber	Varchar2	4	4	4	Vendor
VendorPaymentType	Varchar2	20	3	20	Vendor
DepartmentID	Varchar2	4	4	4	Department

Description:

This table has 4 attributes. Each attribute is taken from different tables. InvoiceNumber is the primary key for this table. VendorNumber and DepartmentID are foreign keys and taken from Vednor and Department tables respectively.

Prod_Line_Mgt:

Data Item Name	Data Item Type	Data Item Length	MetaData Minimum	Metadata Maximum	MetaData Source
ProductNumber	Varchar2	5	5	5	Invoice
LineNumber	Varchar2	4	4	4	Line_Employee

Description:

This table have 2 attributes . ProductNumber is identifier and LineNumber is foreign key. ProductNumber is taken from Invoice table and LineNumber is taken from Line_Employee.

Prod_RawMaterial:

Data Item Name	Data Item Type	Data Item Length	MetaData Minimum	Metadata Maximum	MetaData Source
ProductNumber	Varchar2	4	4	4	Invoice
RawMaterialName	Varchar2	20	5	20	Supply_Schedule

Description:

This table contains 2 attributes. ProuductNumber is primary key and it is taken from Invoice . RawMaterialName is primary key and it is taken from Supply_Schedule.

Supply_Shedule:

Data Item Name	**Data Item Type**	**Data Item Length**	**MetaData Minimum**	**Metadata Maximum**	**MetaData Source**
SupplyCode	Varchar2	5	5	5	Invoice
ProductNumber	Varchar2	5	5	5	Product
RawmaterialName	Varchar2	20	5	20	RawMaterial
Warehouse Number	Varchar2	5	5	5	Warehouse
VendorID	Varchar2	4	4	4	Vendor
Date	Date	N/A	N/A	N/A	

Description:

This is an associative entity . It has 6 attributes and taken from different tables. Supply_code is taken from Invoice and it is the primary key. ProductNumber is taken from Product,RawMaterialName is taken from RawMaterial,WarehouseNumber is taken from Warehouse ,VendorID is taken from Vendor and all these are foreign key

BUSINESS RULES

Rule B1-1: One-to-One Relationship Between Product and Production Line

Constraint: Each product (defined by attributes like ProductNumber, ProductName, Price, etc.) is produced on a uniquely assigned production line (identified by LineNumber, LineCapacity, etc.).

Cardinality: This represents a one-to-one relationship in the database. For every product record, there is exactly one corresponding production line record and vice versa.

Implication: In the database schema, this relationship would be maintained through a foreign key in either the Product or Production line table, ensuring that each product is linked to only one production line, and also they mentioned that Raw material exists, so between Product and ProductionLine, So we added the Prod_Line_Mgmt, so as to ensure the relationship between the Product and Product Line Management.

Rule B1-2: Exclusive One-to-One Relationship for Production Line and Product Type

Constraint: Each production line is exclusively dedicated to the production of one specific type of product.

Cardinality: This is also a one-to-one relationship, but with the emphasis on exclusivity. A production line cannot be linked to more than one product type.

Implication: This would be enforced in the database design by unique constraints on the foreign key relationships, ensuring that once a production line is assigned to a product, it cannot be assigned to another.

Rule B1-3: We have Optional Cardinality between the Product and ProductLine

Constraint: Production lines may have periods where they do not produce any products, typically during maintenance or repairs.

Cardinality: This is considered as a optional between Prod_Line_Mgmt as it is given that during the repair it may not produce the products. A production line might have a product assigned (one) or might not have any during maintenance (zero).

Implication: In the database, this could be managed by having optional cardinality in the relationship field during the period of maintenance.

Rule B2-1: Each vendor can supply many raw materials to any number of warehouses.

Many-to-Many Relationship Between Vendors and Warehouses via Supply Schedule and raw materials entity

Constraint: A vendor can supply a variety of raw materials to multiple warehouses. Here we have many to many relationship which is broken via Associative entity known as **Supply Schedule,** where initially we have many to many relationships is broken into two “one to many” relationships.

Cardinality: This establishes a many-to-many relationship between vendors and warehouses, mediated by raw materials. We use **Prod_Raw Material** to establish the relationship between the Product and Supply schedule which in turn is related between the Vendor and warehouse.

By analysing the data, We found that particular product has fixed set of Raw materials like POO1 has Polyester and Vinyl and also particular Raw materials which use the products are produced by specific Line number.

Implication: This relationship would typically be implemented using a junction table in the database, linking VendorID and WarehouseNumber through RawMaterialName and an associated identifier.

Rule B2-2: Raw materials are supplied by any number of warehouses, which are supplied by any number of vendors.

Complex Many-to-Many Relationships:

Constraint: Raw materials can be supplied by several warehouses, which in turn receive supplies from multiple vendors.

Cardinality: This is a many-to-many relationship between raw materials and warehouses and between warehouses and vendors. This many-to-many relationship is likely captured through the Prod_Raw_Material entity, which holds the RawMaterialName and is linked to both the Warehouse and Vendor. This entity serves as a junction table that enables many raw materials to be associated with many warehouses and vendors.

Implication: This would require a more complex relationship structure, possibly involving multiple junction tables to accurately.

Rule B2-3: Raw materials may also be directly supplied by vendors

Constraint: Vendors have the option to supply raw materials directly, bypassing warehouses. This direct supply relationship can be depicted in the ERD by the link between the Vendor and Prod_Raw_Material entities. It indicates that vendors have the capability to supply raw materials directly, without going through a warehouse.

Cardinality: This suggests an additional many-to-many relationship between vendors and raw materials, which are divided into Prod_Raw_Material table

Implication: A direct relationship between vendors and raw materials is modelled in the database, potentially using a separate junction table Supply schedule which has ternary dependency

Rule B2-4: Each warehouse can be supplied with any number of raw materials from more than one vendor, but each warehouse must be supplied with at least one raw material.

Constraint: The one-to-many relationship from Warehouse to Prod_Raw_Material, indicating that each warehouse can have multiple types of raw materials supplied by different vendors. The mandatory aspect of having at least one raw material is a minimum cardinality constraint, which can be enforced in the database design by ensuring that no Warehouse record exists without at least

one associated Prod_Raw_Material record. Each warehouse must receive at least one type of raw material from one or more vendors, but can receive multiple types.

Cardinality: This is a one-to-many relationship from the perspective of the warehouse to raw materials.

Implication: In the database, this rule is enforced by ensuring that each warehouse record in the database is linked to at least one raw material record, which in turn is linked to one or more vendor records.

B3-1: The company has departments, warehouses, and production lines.

This rule indicates that there are separate entities for departments, warehouses, and production lines within the company's database structure.

Entities: Departments, Warehouses, Production Lines.

Attributes: Each entity has unique identifiers (DepartmentID, WarehouseNumber, LineNumber)

Implication : The cardinality between these entities and the company is one-to-many (1:N). That is, one company has many departments, warehouses, and product lines.

B3-2: The company designs and produces products.

This rule establishes that there is an entity for products.

The relationship between the company and products is one-to-many, as a company designs and produces many products. This rule emphasizes the inclusion of products and their design process in the database, linking products to designers. We see that in Products table we have **Designer id**, where we there are specific employee who design the products.

B3-3: Each department, warehouse, and production line have multiple employees.

Relationship: One to many relation between Departments, Warehouses, Production Lines, and Employees.

Constraint: Foreign keys in the Employee table (DepartmentID, WarehouseNumber, LineNumber) to maintain relationships.

Implication: Each unit (department, warehouse, production line) has multiple employees, enforced by foreign key relationships.

B3-4: Each employee works in only one department, warehouse, or production line.

Relationship: Disjoint relation between Departments, Warehouses, Production Lines, and Employees.

Constraint: Each Employee is associated with only one Department, Warehouse, or Production Line.

Cardinality: One-to-one within the scope of employment.

Implication: Ensures data integrity by preventing an employee from being assigned to multiple units. Here we have used the disjoint because when we see the "**Employees**" data the employee is either Department Employee or Warehouse employee or Line Employee. It is either of the case any of the two like that, So we use Disjoint and divided the **table Line_Employee, Department_Employee, Warehouse_Employee.**

B3-5: Only Employees within the design department design products.

This implies a relationship between the department and employees that is one-to-many, with the additional rule that only employees in a specific department (design) are related to the product entity in the context of designing, when we see the Department table data "**D002**" which is the Department id of Design department, which consist of 4 employees **"E016, E017, E018, E019".**

Constraint: A subset of employees (designers) is linked to the design department.

Implication: There is specific attribute in the Employee table i.e **"Field of Work"** to identify designers or a separate table linking designers to the design department.

B3-6: Each designer can design multiple products.

The relationship between designers (a subset of employees) and products is one-to-many. A designer (employee) can design multiple products, but each product is designed by only one designer. Here in our ER diagram, we have connected Line employees with the ProductionLine as ,Production line is in turn connected to Products. The 4 employees **"E016, E017, E018, E019" will design 75 products** which is one to many.

Implication: This enables tracking of which designer is responsible for designing which products.

B3-7: Each product has exactly one designer.

This enforces a one-to-one relationship between the product and the designer. While a designer can design many products, each product is associated with exactly one designer.

Implication: Ensures that each product is associated with exactly one designer.

B4-1: Vendors submit an invoice when they supply a raw material.

Constraint: When a vendor supplies raw material, an entry is made in the Invoice table, capturing the details of the transaction.

Implication : This rule implies that there is a one-to-many relationship between a vendor and invoices since a single vendor can submit multiple invoices for different supplies of raw materials.

The cardinality can be defined as one-to-many (1:N) from the vendor to invoices, with the minimum cardinality being zero (if a vendor has not supplied rawmaterial then we don't get the invoices. anything yet) and the maximum being many (as a vendor can supply multiple times).

Here when we observe the data there is a partial dependency existing that's the reason we have splitted the **Invoice table** with the attributes **Invoice number and Total Amoun**t and Vend_Payement with the attributes Invoice Number, Vendor_Number, Vendor_payment_type, DepartmentID,

B4-2: Invoices are processed by the accounting department.

Constraint: In the database, this could be implemented by either adding a field to the Invoice table to indicate the DepartmentID responsible for processing the invoice (if it varies) or, if it's always the accounting department, this is a fixed process rule rather than a database attribute.

Implication : This rule suggests that invoices are associated with the accounting department, which processes them. Since the rule specifically mentions the accounting department, it implies that all invoice processing is centralized within this particular department.

The relationship between the accounting department and invoices is likely one-to-many, as one department processes many invoices. However, there is an implicit one-to-one relationship between each invoice and the department because each individual invoice is processed by only the accounting department.

The cardinality could be defined as one-to-many (1:N) from the accounting department to invoices, with the minimum cardinality being at least one (since the accounting department must process at least one invoice) and the maximum being many (since they process all invoices). The cardinality between the Vend_Payment and Department is that all the departments does not process the invoices, only accounting department process the queries, so it is optional.

Primary and Foreign Keys:

Pre-Normalized

S. No	Table Name	Primary Key	Foreign Key(s)	Reference Table(s)	Reference Table Attribute(s)
1	Department	Department_ID			
2	Employee	EmployeeID			
3	Warehouse	WarehouseNumber			
4	Vendor	VendorNumber			
5	Invoice	InvoiceNumber	VendorNumber	Vendor	VendorNumber
6	Supply Schedule	SupplyCode	ProductNumber, WarehouseNumber, VendorID	Raw Material, Warehouse, Vendor	ProductNumber, WarehouseNumber, VendorNumber

7	Raw Material	ProductNumber			
8	Product Design	DesignerID			
9	Product	ProductNumber	DesignerID	Product Design	DesignerID
10	Warehouse Employee	EmployeeID	WarehouseNumber	Warehouse	WarehouseNumber
11	Department Employee	EmployeeID	DepartmentID	Department	DepartmentID

Post normalized:

S. No	**Table Name**	**Primary Key**	**Foreign Key(s)**	**Reference Table(s)**	**Reference Table Attribute(s)**
1	Employee	EmployeeID	None	Department_Employee, Warehouse_Employee, Line_Employee	DepartmentNumber, WarehouseNumber, LineNumber
2	Department	Department_ID	None	Department_Employee, Vend_Payment	DepartmentNumber, DepartmentID
3	Department_Employee	EmployeeID	DepartmentNumber, EmployeeID	Department, Employee	Department_ID, EmployeeID
4	Warehouse	Warehouse Number	None	Warehouse_Employee	WarehouseNumber
5	Warehouse_Employee	EmployeeID	Warehouse Number, EmployeeID	Warehouse, Employee	WarehouseNumber, EmployeeID
6	Line_Employee	EmployeeID	LineNumber, EmployeeID	ProductionLine, Employee	LineNo, EmployeeID

7	ProductionLine	LineNo	None	Prod_Line_Mgmt, Line_Employee	LineNumber, LineNumber
8	Prod_Line_Mgmt	ProductNumber	LineNumber, ProductNumber	ProductionLine, Product	LineNo, ProductNumber
9	Product	ProductNumber	None	Prod_Line_Mgmt	ProductNumber
10	Vendor	VendorNumber	None	Supply_Schedule, Vend_Payment	VendorNumber, VendorNumber
11	Vend_Payment	Invoice Number	VendorNumber, Department ID	Vendor, Department	VendorNumber, Department_ID
12	Invoice	InvoiceNumber	None	Vend_Payment	Invoice Number
13	Supply_Schedule	SupplyCode	ProductNumber, RawMaterialName, VendorNumber, Date	Prod_Raw_Material, Vendor	ProductNumber, VendorNumber
14	Prod_Raw_Material	ProductNumber, RawMaterialName	None	Supply_Schedule	SupplyCode

Entities of ERP: The Enterprise Resource Planning (ERP) Database Schema for a manufacturing and supply chain company, the entities can be understood in the following context:

Management of Employees Across Departments, Warehouses, and Production Lines:

The **Department Dataset** is crucial for understanding the organizational structure and roles within various departments.

The **Employee Dataset** provides detailed information about each employee, including their position, department, and association with specific warehouses or production lines. This aids in allocating employees distinctly to specific entities within the organization.

Overseeing the Product Lifecycle:

The **Product Dataset** is central to tracking the lifecycle of products from concept to market. It contains details on the type, design, and cost of each product.

The **Production Line Dataset** aligns with the manufacturing aspect, detailing the capacity and location of each production line. This dataset is essential for planning and optimizing the manufacturing process.

Managing the Flow of Raw Materials:

The **RawMaterial** Dataset provides insights into the materials required for each product and their corresponding production lines. This helps in streamlining the material requirement process.

The **Warehouse Dataset and Supply_Schedule** Dataset are key to managing the storage and supply of raw materials, whether stored in-house or supplied by vendors, ensuring a consistent material flow for production.

Tracking and Processing of Vendor-Supplied Invoices:

The **Invoice Dataset** is pivotal for the financial management aspect, especially for tracking and processing payments to vendors. It contains detailed information on each invoice, including the amount, vendor details, and payment type.

The **Vendor Dataset** offers comprehensive information about each vendor, which is essential for validating invoices and managing vendor relationships.

Detailed Scenario: Manufacturing a Soccer Jersey

1. Department Entity

Role in ERP: Manages organizational structures and roles.

Example: The 'Design' department (e.g., DepartmentID: D002) is tasked with creating new product designs. They collaborate with the 'Marketing' department to understand market trends and customer preferences.

2. Employee Entity

Role in ERP: Manages workforce data, roles, and responsibilities.

Example: A designer, say Emma (EmployeeID: E017), in the 'Design' department, is responsible for the new jersey design. Her profile in the ERP includes her role, department, contact info, and work history.

3. Product Entity

Role in ERP: Manages product information, specifications, and lifecycle.

Example: Once Emma finalizes the jersey design, it's logged into the ERP as a new product (ProductNumber: PR002), including details like type, size range, expected price, and cost.

4. Raw Material Entity

Role in ERP: Tracks materials required for each product.

Example: The ERP system lists materials needed for the jersey, such as polyester and vinyl. It checks current inventory and flags if new orders are needed.

5. Vendor and Invoice Entities

Role in ERP: Manages supplier relationships, procurement, and financial transactions.

Example: If polyester stock is low, the ERP initiates procurement. It selects a vendor based on cost and quality (VendorNumber: V001) and generates a purchase order. When materials are delivered, the vendor's invoice (InvoiceNumber: INV0030) is processed and recorded in the ERP.

6. Supply Schedule Entity

Role in ERP: Manages the scheduling and delivery of materials.

Example: The ERP schedules the delivery date for the polyester (SupplyCode: S0003) and logs it in the system. This ensures the material is available when production is scheduled to start.

7. Warehouse Entity

Role in ERP: Manages storage and inventory.

Example: The received materials are stored in a warehouse (WarehouseNumber: WH001). The ERP updates inventory levels and tracks where each material is stored within the warehouse.

8. Production Line Entity

Role in ERP: Oversees manufacturing processes, line capacities, and scheduling.

Example: The production of the jersey is scheduled on Line L002. The ERP considers line capacity, current workload, and material availability to optimize production scheduling.

9. Employee (Production Staff) Entity

Role in ERP: Allocates staff to specific tasks and production lines.

Example: The ERP assigns staff, (EmployeeID: E004), to Line L002 based on their skills and experience. the schedule, performance, and production line assignment are all managed within the ERP.

Integration and Workflow in the ERP System:

Design to Production: The ERP integrates the design and production processes. Once a product is designed and materials are scheduled, it efficiently transitions to production planning.

Material Requirement Planning (MRP): The ERP's MRP module analyzes product requirements, forecasts material needs, and initiates procurement processes to ensure timely availability of materials.

Vendor Management: The ERP maintains a database of vendors, managing orders, invoices, and payments. It ensures procurement is cost-effective and meets quality standards.

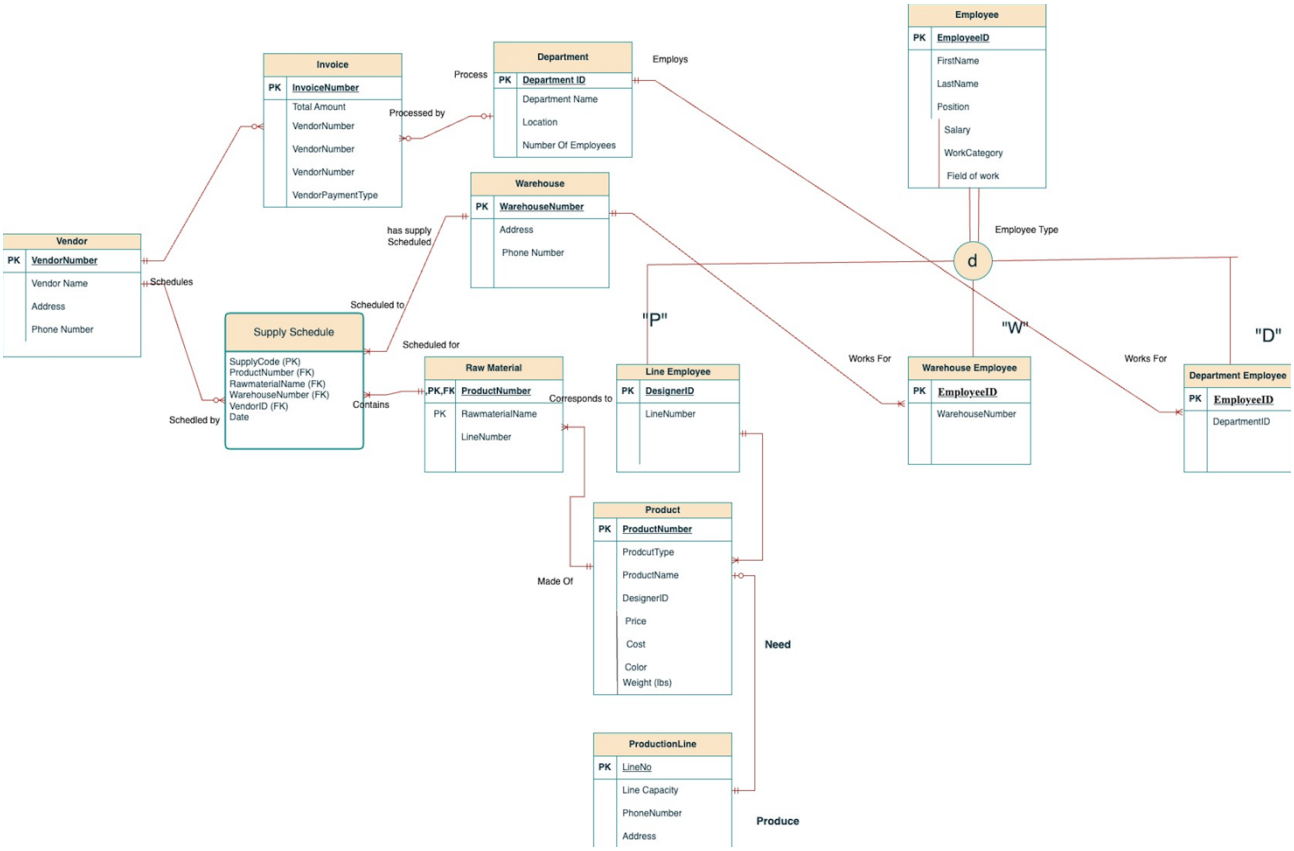
Inventory Management: The ERP tracks real-time inventory levels, helping manage warehouse space and ensuring materials are available for production when needed.

Production Management: The ERP schedules production, assigns staff, and monitors progress. It ensures optimal use of resources and timely product completion.

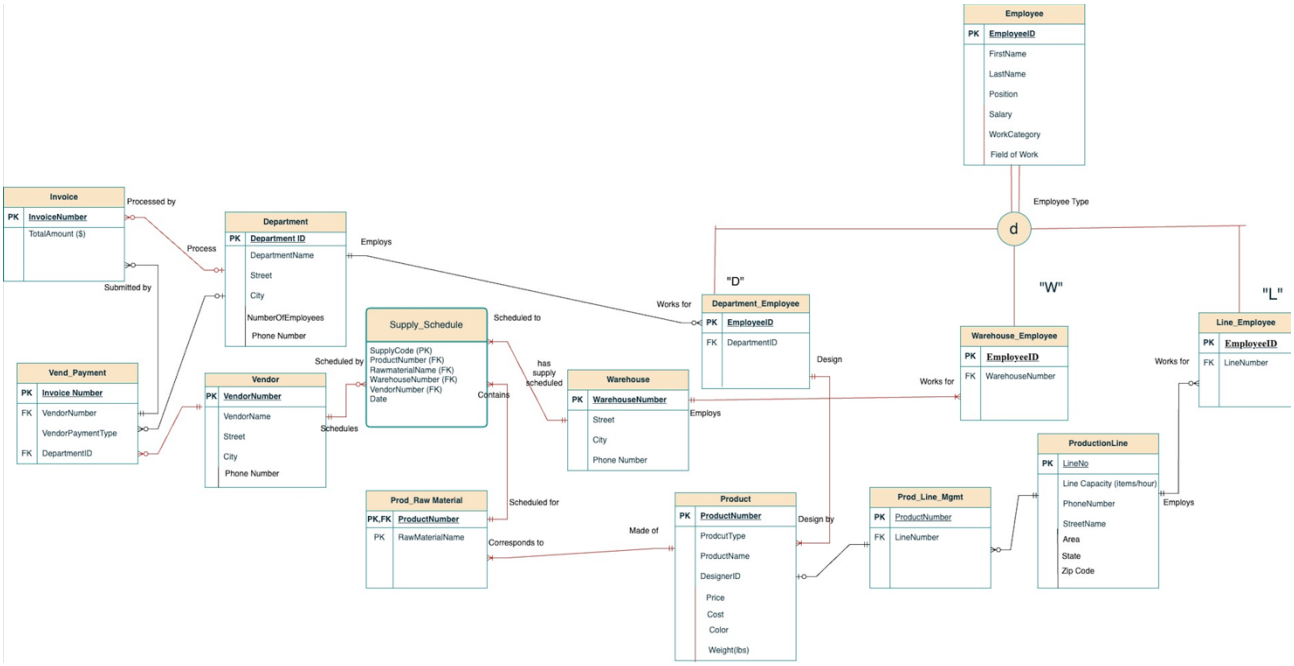
In this detailed scenario, the ERP system acts as the central nervous system of the company, integrating various departments, processes, and data flows. It enhances operational efficiency, decision-making, and responsiveness to market demands, essential for a contemporary manufacturing entity.

ER Diagrams:

Pre-Normalization:

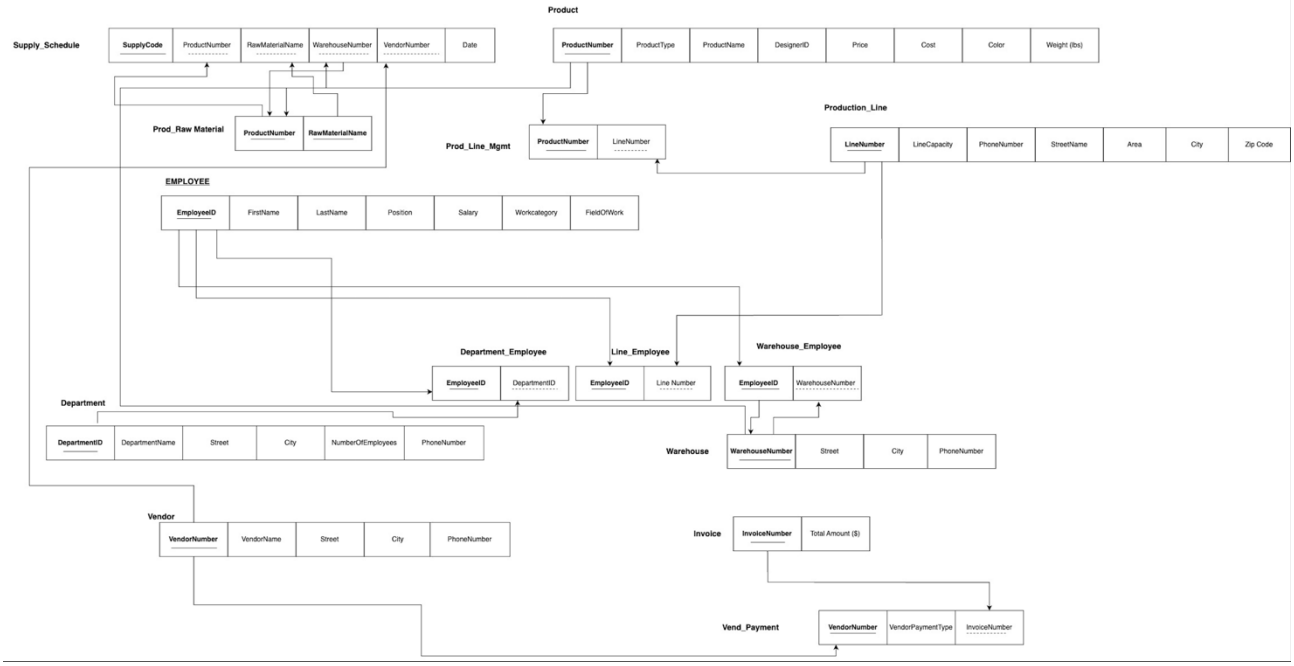


Post Normalization:



Relational Schema:

After Normalization:



Domain Constraint

A domain constraint specifies the permissible values for a given attribute. The domain refers to the type of data, range of values, and format that can be stored in an attribute.

In our data, when we observe the metadata, we have the data types, length of the data an the range of values. When we observe a Phone number, the maximul limit of the phone number is 14, While we initially inserted the data with the data type of Number with the precision of 10 and scale of 2 , we faced an error with domain Constraint stating that the maximum length of the data is 14 and the precison we mentioned as 10, which we modified to 20 as precision in the perspective of future.

Entity Integrity Constraint

An entity constraint, also known as an entity integrity constraint, refers to the concept that every entity must be uniquely identifiable. This typically involves specifying a primary key for each table in the database. A primary key is a unique identifier for a row within a table and cannot be NULL, ensuring that each record is unique and can be distinguished from all others.

Most of the data does not have null values on the primary keys.

In our data, the “Raw Materials” data does not have primary key , So we have divided the data into two tables , one is Prod_mgmt and other is Prod_Raw material to make sure that the primary key exists and is connected to other tables like **Product, Product line**. Also, Prod_Raw_Material is connected to **SupplySchedule and Product** and in that table**,** we have **composite Primary** key that is one product has set of Raw materials so, we keep the composite primary key, to make it as unique identifier.

Referential Constraint

A referential constraint, also known as referential integrity, is a rule that maintains consistency among the rows of two related tables. This is accomplished using foreign keys. A foreign key in one table points to a primary key in another table. Referential integrity ensures that the relationship between two tables remains consistent

Also the Referential integrity Constraint, when we observe the data in Supply schedule,

S0062	PR029	Cotton blend	WH003	V001	10-12-2023
S0063	PR030	Wool	WH002	V003	11-12-2023
S0064	PER030	Cotton blend	WH001	V004	12-12-2023

We observe the ProductNumber which is FK of this table is PER030, but the other productcode are PR030, which could not be mapped with the Pk of the table as in the main table it is mentioned as PR030 which raised an constraint.

Anomalies:

Deletion Anomality: Happen when deleting data in one dataset inadvertently removes necessary data from the database

Deletion Anomaly with Vendor: The check reveals an anomaly. Removing a vendor from the "Vendor file might affect the Supply Schedule and "Invoice data, where the vendor is referenced. This could result in invoices and supply schedules pointing to vendors that no longer exist in your records.

Deletion Anomaly with Product: The check indicates an anomaly. This means that if a product is removed from the "Product data, it might affect the "Supply_Schedule data where the product is referenced. This could lead to references to non-existent products in the supply schedule..

Update Anomaly: Arise when changes to data in one dataset require multiple updates in other datasets, leading to potential inconsistencies.

There is an anomaly detected . Modifications in warehouse details in the "Warehouse data could fail to be reflected in the "Supply_Schedule data and Employee data. For example, if a warehouse's number is changed, the supply schedule and employee records may continue to reference the outdated number.

Insertion Anomalies: Occur when you can't add data to a database without adding additional, perhaps unrelated data. For example, being unable to add a new warehouse without having an employee assigned to it.

Relationships:

Unary relationships

A unary relationship occurs when an association is maintained within a single entity type. In other words, the relationship is with the same entity, in our case , we don’t have any unary relationships

Binary relationships

Binary relationships involve two different entities. They are the most common types of relationships in database design. Most of the relationships are binary.

A relationship between the "Invoices" and "Department" entities can be binary. Every invoice might be processed by the accounting department. This "processed in" relationship links department employee to a invoice.

Ternary Relationships:

Ternary relationships involve three different entity types. They are used to model complex relationships where a binary relationship is insufficient.

The Supply Schedule is an Associative entity , where initially there is many to many relationship exists between the vendor, Warehouse and raw materials according to the business rules B2-1 and B2-2 and that was the reason, we had a terenary relationship. We have two binary relationships. Each record in "Supply_Schedule" would then represent a unique combination where a specific product (from "Prod_raw Material") is supplied by a particular vendor (from "Vendor") to a certain warehouse (from "Warehouse"). This is a classic ternary relationship as it requires all three entities to provide the complete picture of the supply schedule as associative entity. This ternary relationship is necessary because the supply information depends simultaneously on the specific product_raw material, vendor, and warehouse.

Dependencies:

Types of dependencies:

1. **Total Dependency:**
   Total dependency occurs when a non-key attribute is entirely dependent on the entire primary key. In a table with a composite primary key, the non-key attribute relies on the combination of all primary key components. Resolving total dependencies often involves breaking down tables to eliminate redundancy and ensure that non-key attributes are fully dependent on the primary key.
2. **Partial Dependency:**
   Partial dependency arises when a non-key attribute is dependent on only a portion of the composite primary key, rather than the complete key. In this scenario, the non-key attribute relies on a subset of the primary key. Database no components and to mitigate partial dependencies by restructuring tables, possibly decomposing them into smaller, more manageable components, and ensuring that each attribute is fully dependent on the primary key.
3. **Transitive Dependency:**
   Transitive dependency occurs when an attribute is dependent on another non-key attribute, rather than directly on the primary key. In a table with attributes A (primary key), B, and C, if C depends on B, and B depends on A (the primary key), it indicates a transitive dependency. Addressing transitive dependencies involves normalizing the database structure to enhance data integrity and maintainability.

In our case we have the below Pre-Normalized Tables:

Table Name	Type of Dependency
Product	Complete Dependency
ProductionLine	Complete Dependency
SupplySchedule	Complete Dependency
Employee	Complete Dependency

Warehouse	Complete Dependency
Department	Complete Dependency
Vendor	Complete Dependency
RawMaterial	Partial Dependency
Invoice	Transitive Dependency

The RawMaterial table utilizes 'ProductNumber' as its primary key, with 'RawmaterialName' being derived from 'ProductNumber'. However, the absence of any specific definition for Line Number results in a partial dependency within this table.

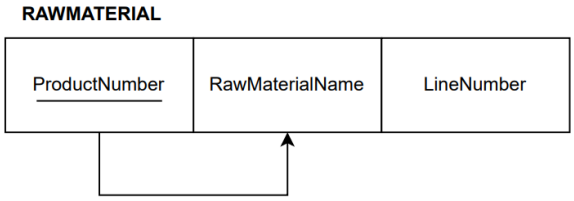


Figure 1: Illustration of dependencies of attributes for invoice table

The Invoice table is structured with 'InvoiceNumber' as its primary key. The 'TotalAmount' attribute is determined by 'InvoiceNumber,' while 'VendorPaymentType' is influenced by 'VendorName,' which is not a primary key. Consequently, a transitive dependency exists in this table.

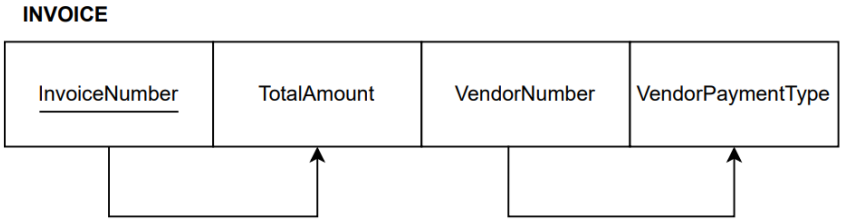


Figure 2: Illustration of dependencies of attributes for invoice table

Below will be the new tables:

Old Table	**New Table**	**Attributes**
RawMaterial	Prod_Raw Material	ProductNumber RawMaterialName
	Prod_line_Mgmt	ProductNumber LineNumber
Invoice	Invoice	InvoiceNumber TotalAmount
	Vend_Payment	InvoiceNumber VendorNumber VendorPaymentType

		DepartmentId

Final list of tables will be:

Table Name	**Type of Dependency**
Product	Complete Dependency
ProductionLine	Complete Dependency
SupplySchedule	Complete Dependency
Employee	Complete Dependency
Line_Employee	Complete Dependency
Department_Employee	Complete Dependency
Warehouse_Employee	Complete Dependency
Warehouse	Complete Dependency
Department	Complete Dependency
Invoice	Complete Dependency
Vend_Payment	Complete Dependency
Vendor	Complete Dependency
Prod_Raw Material	Complete Dependency
Prod_line_Mgmt	Complete Dependency

Normalization:

Normalization involves iteratively restructuring relations to eliminate anomalies and create more concise, well-organized relations. The primary objectives of normalization include:

1. Minimizing Data Redundancy: This helps prevent anomalies and conserves storage space.
2. Simplifying Referential Integrity: The enforcement of referential integrity constraints becomes more straightforward.
3. Facilitating Data Maintenance: It becomes easier to perform data operations such as insertion, updating, and deletion.
4. Enhancing Design Quality: The result is a more refined representation of the real-world scenario and a more robust foundation for future expansion.

The normalization process adheres to specific forms:

1. **First Normal Form (1NF)**: This form ensures the removal of multivalued attributes (repeating groups), resulting in a table where each intersection of row and column contains a single value, potentially null.

2. **Second Normal Form (2NF):** The elimination of partial functional dependencies is the focus here, ensuring that non-key attributes are identified by the entire primary key.

3. **Third Normal Form (3NF):** The removal of transitive dependencies is the goal, ensuring that non-key attributes are identified solely by the primary key.

In the following list of tables, there is an 'Address' field representing a composite entity comprising street name, city, area, state, and zip code. It is necessary to separate the 'Address' column into individual columns to achieve 1NF.

According to the updated list of tables, namely Production_Line, Warehouse, Vendor, and Department, each contains a composite attribute 'address' that will be dissected into distinct components, representing different parts of the address.

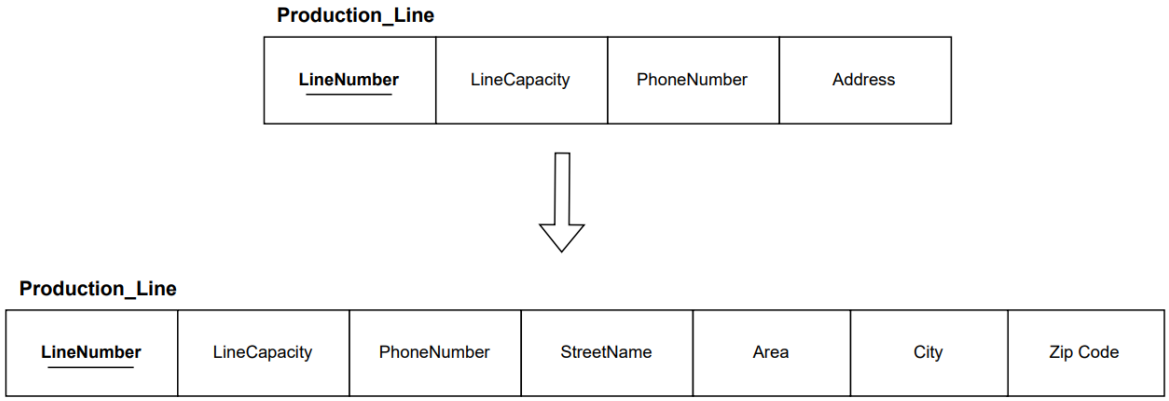


Figure 3: Conversion of Production_Line table to 1NF

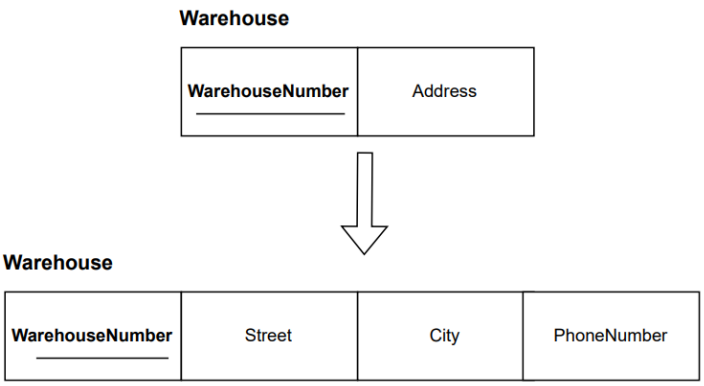


Figure 4: Conversion of Warehouse table to 1NF

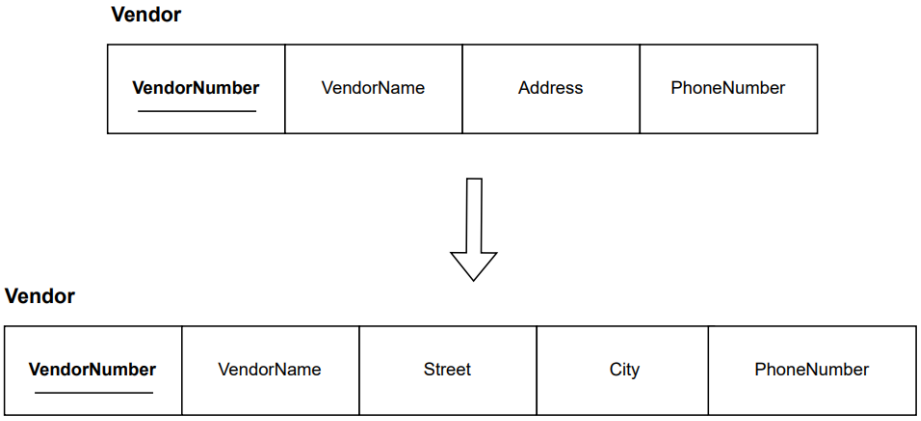


Figure 5: Conversion of Vendor to 1NF

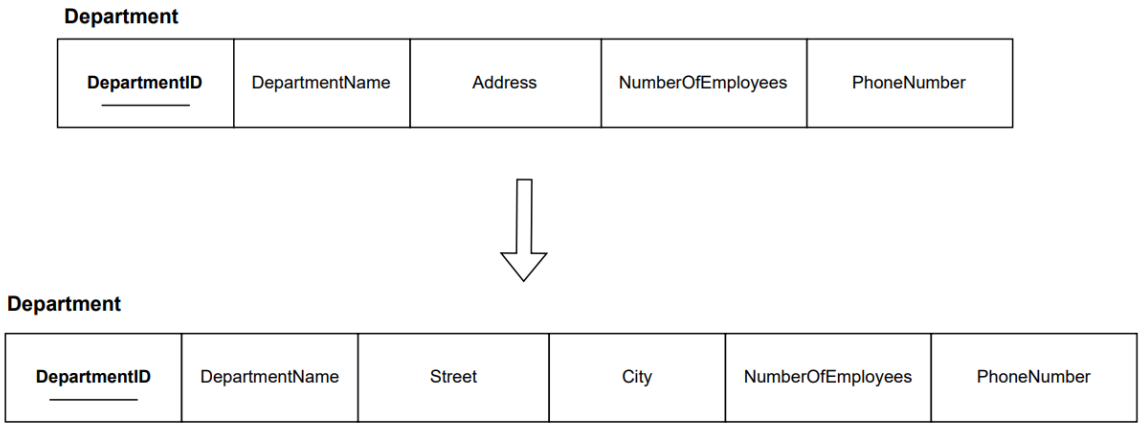
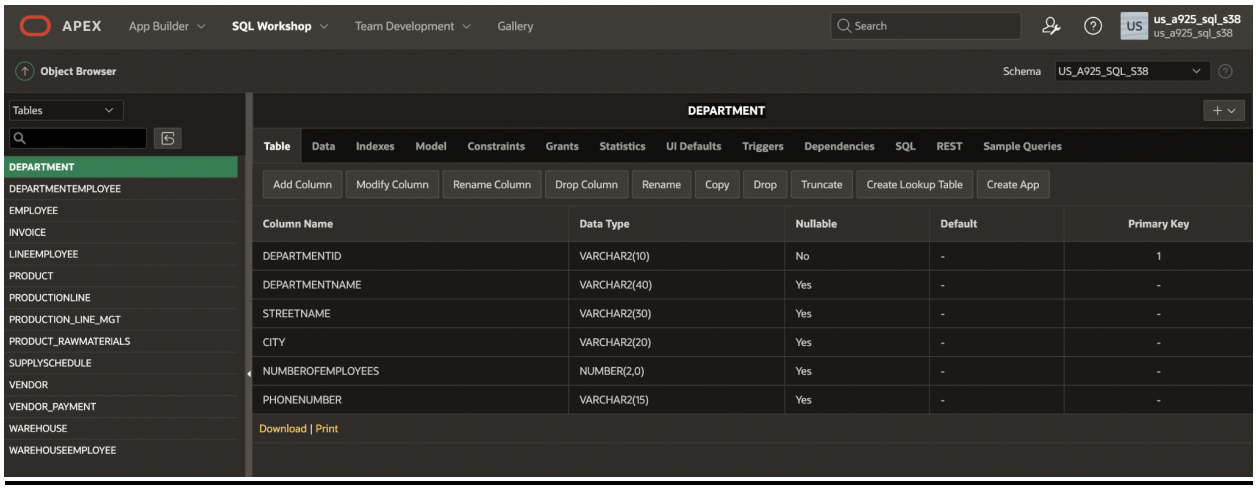


Figure 6: Conversion of Department table to 1NF

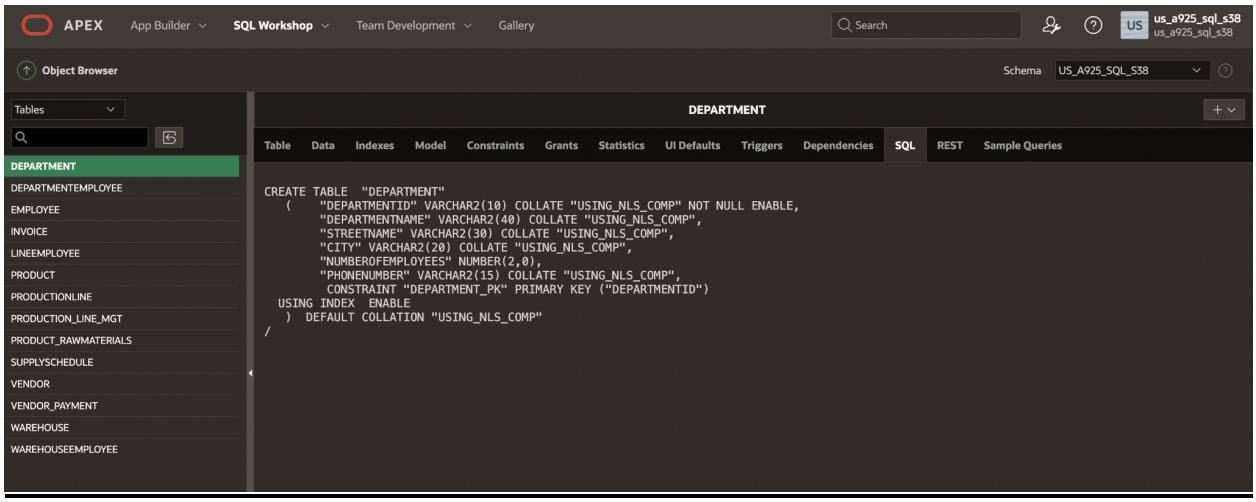
TABLES AND QUERIES:

1.DEPARTMENT:

TABLE:

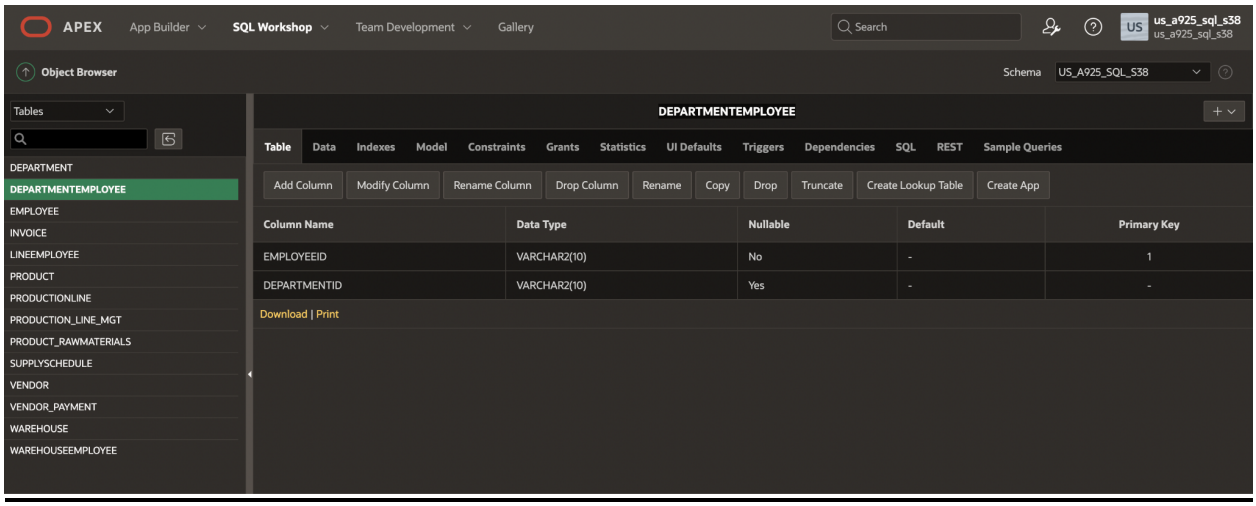


QUERY:

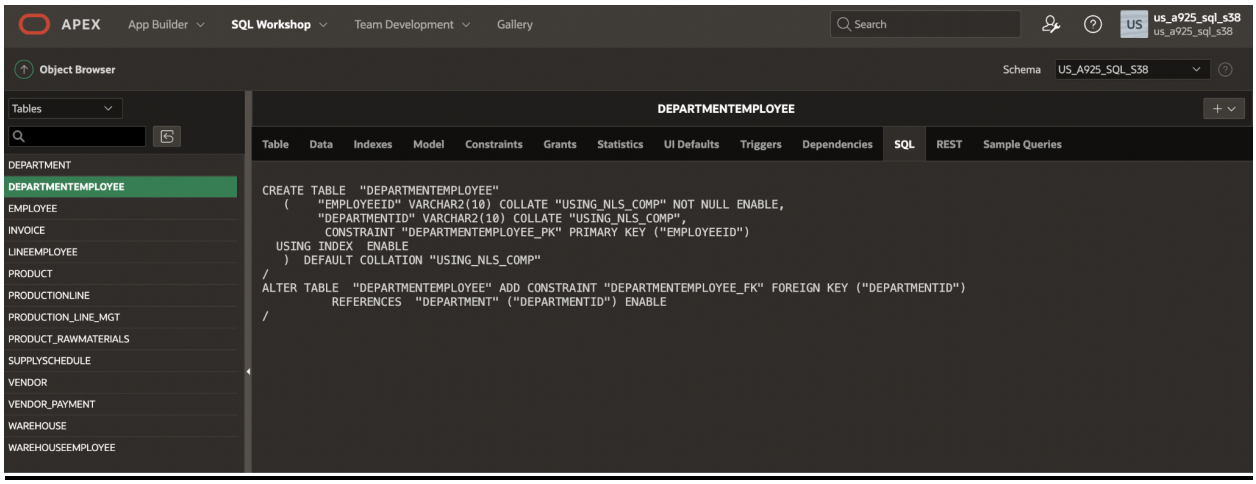


2.DEPARTMENTEMPLOYEE:

TABLE:

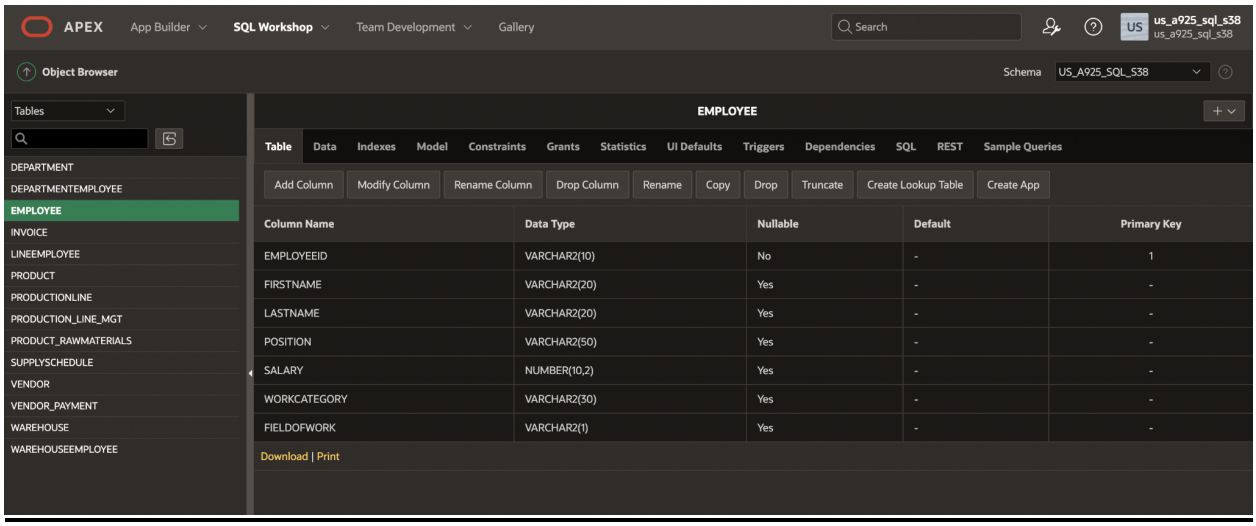


QUERY:

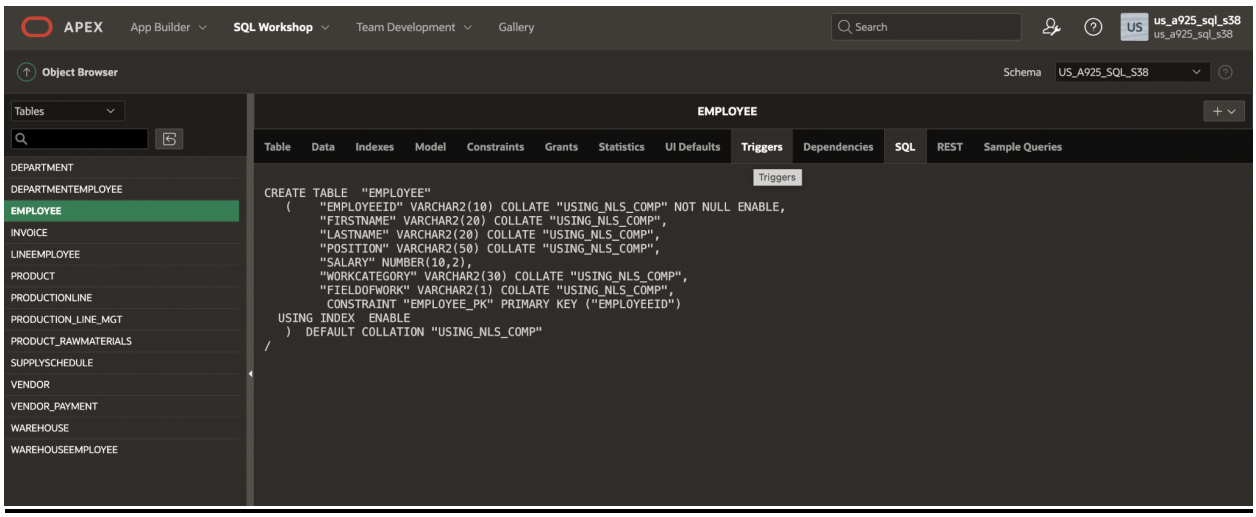


3.EMPLOYEE:

TABLE:

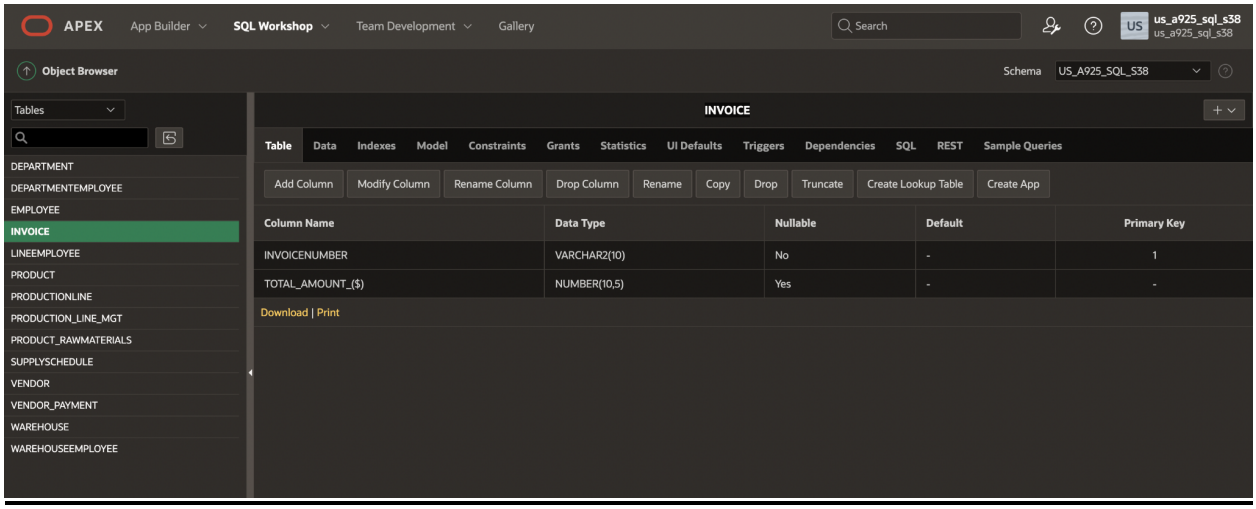


QUERY:

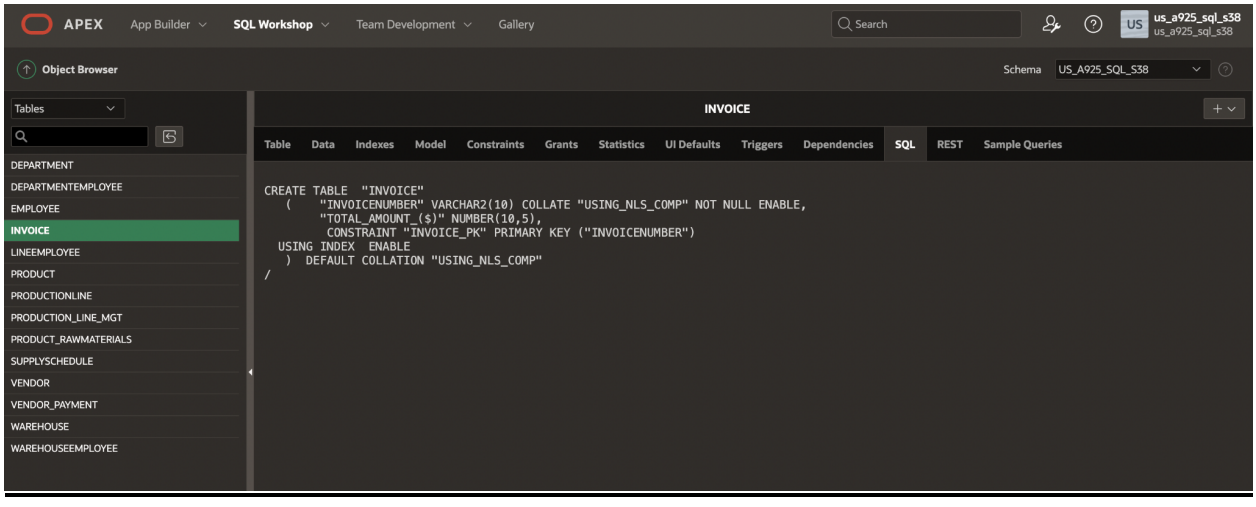


4. INVOICE:

TABLE:

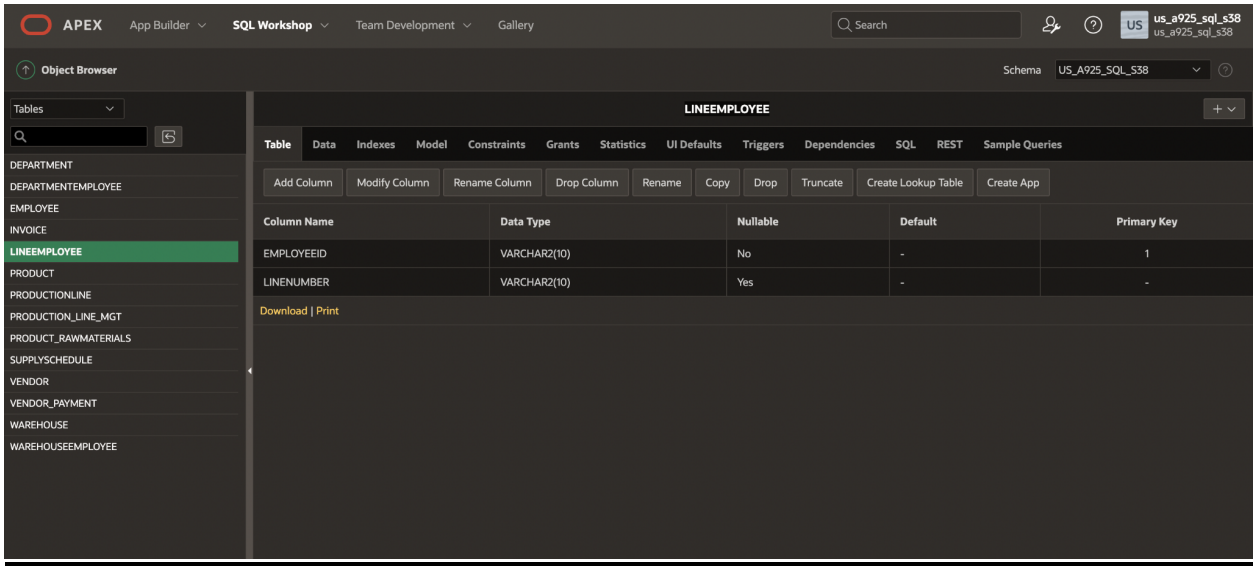


QUERY:

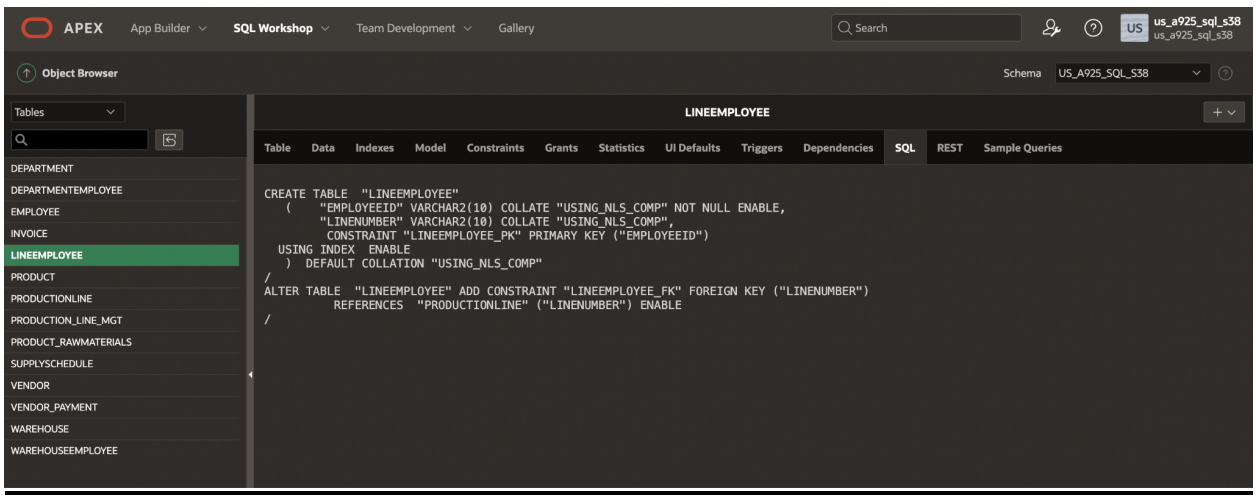


5.LINEEMPLOYEE:

TABLE:

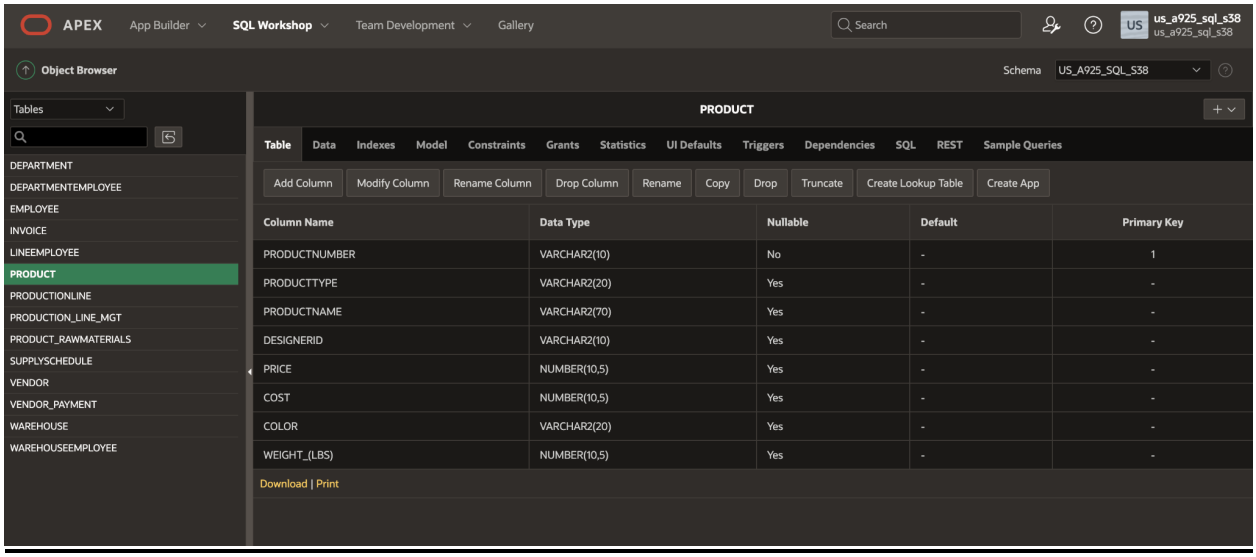


QUERY:

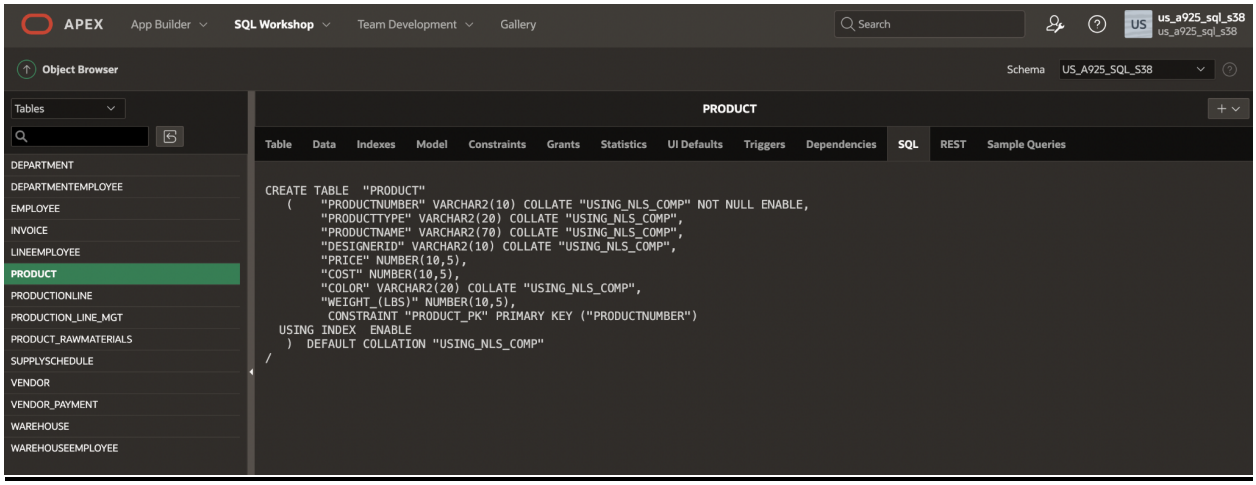


6.PRODUCT:

TABLE:

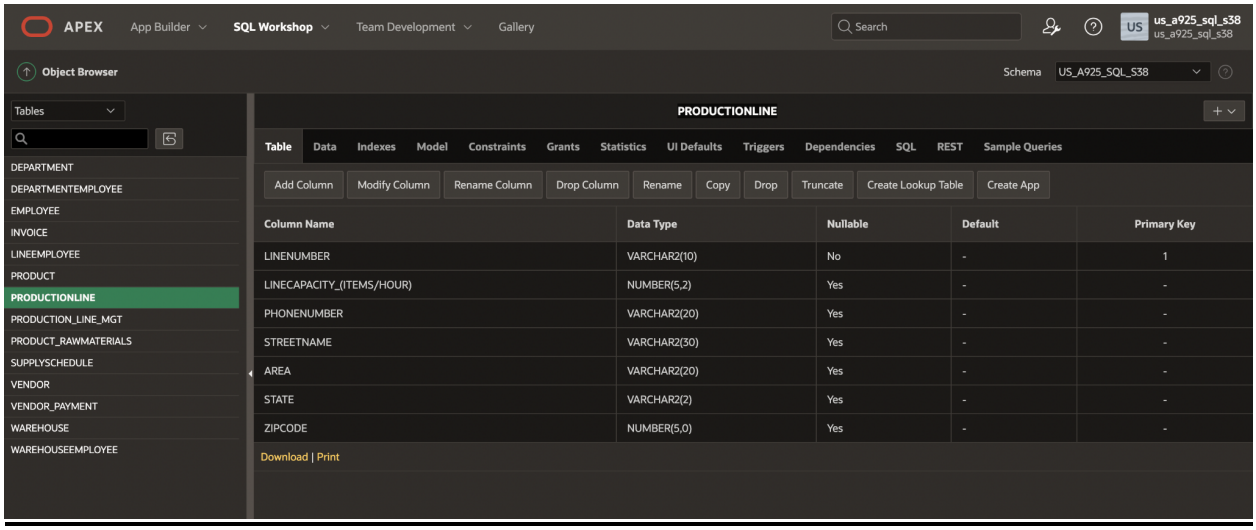


QUERY:

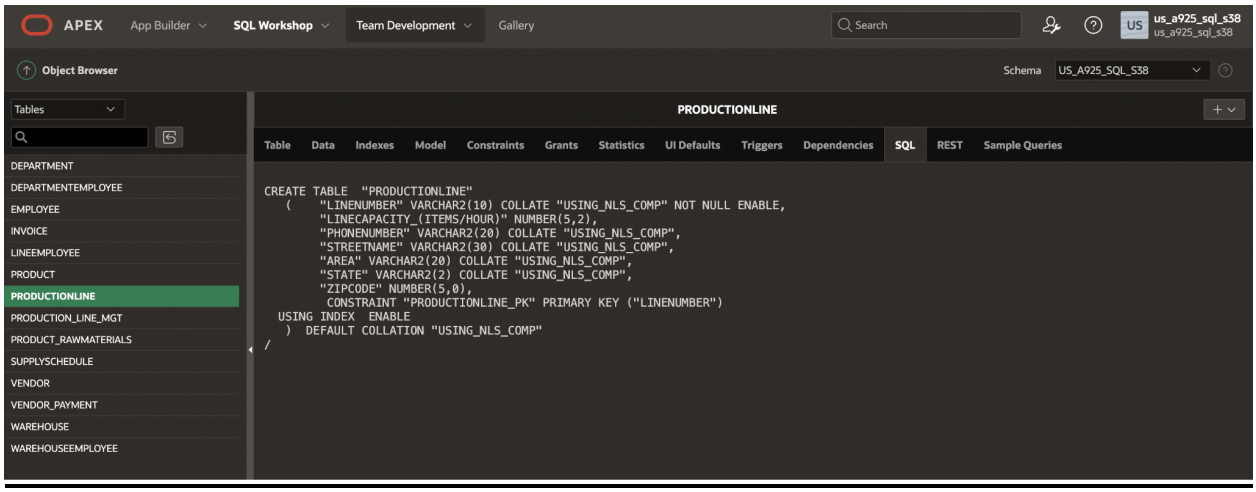


7.PRODUCTIONLINE:

TABLE:

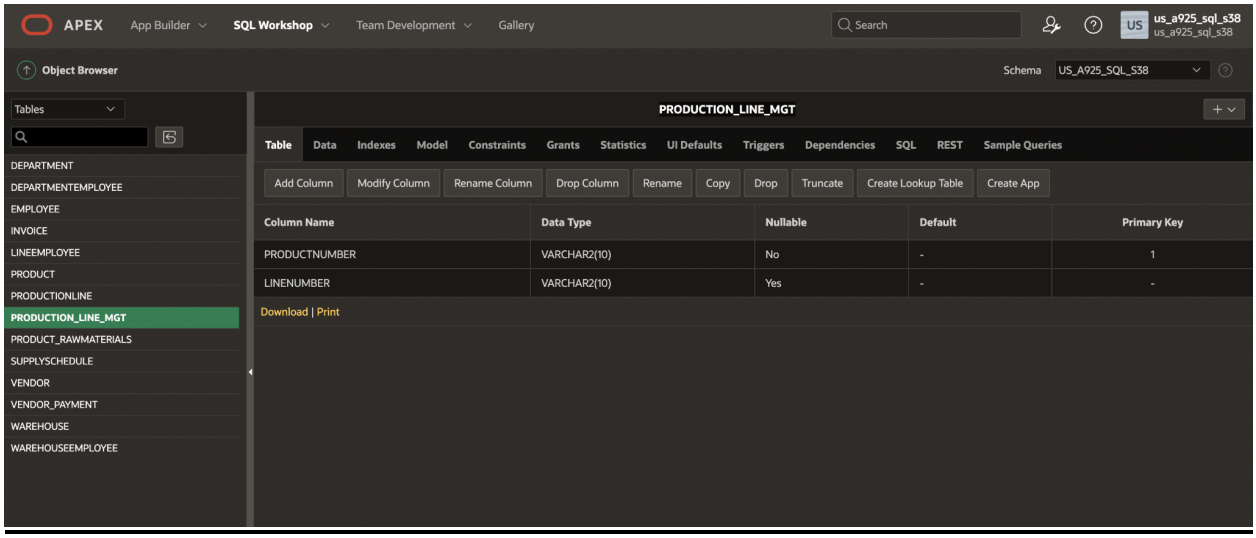


QUERY:

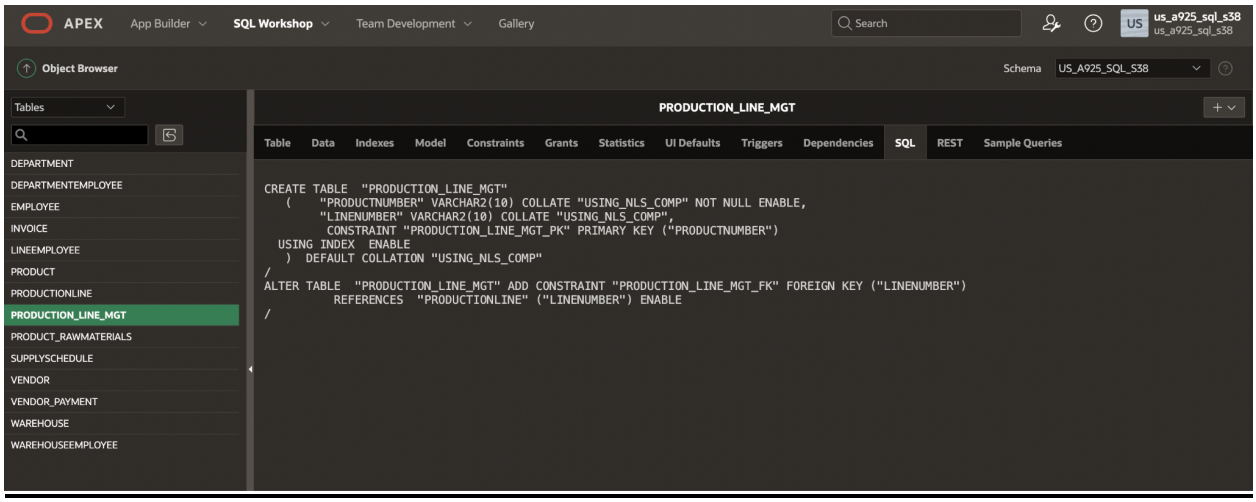


8.PRODUCTION_LINE_MGT:

TABLE:

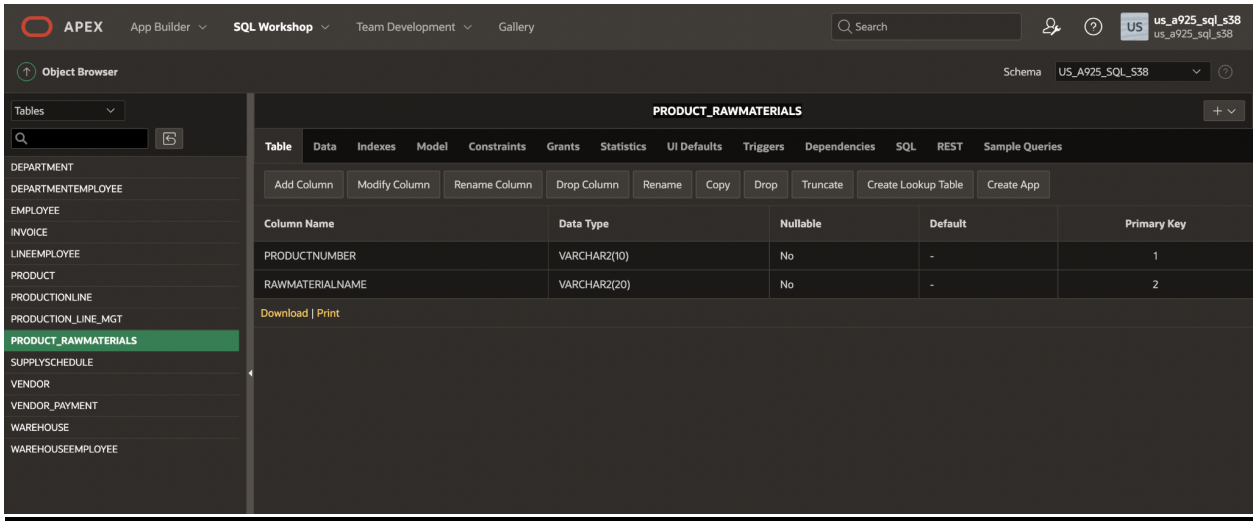


QUERY:

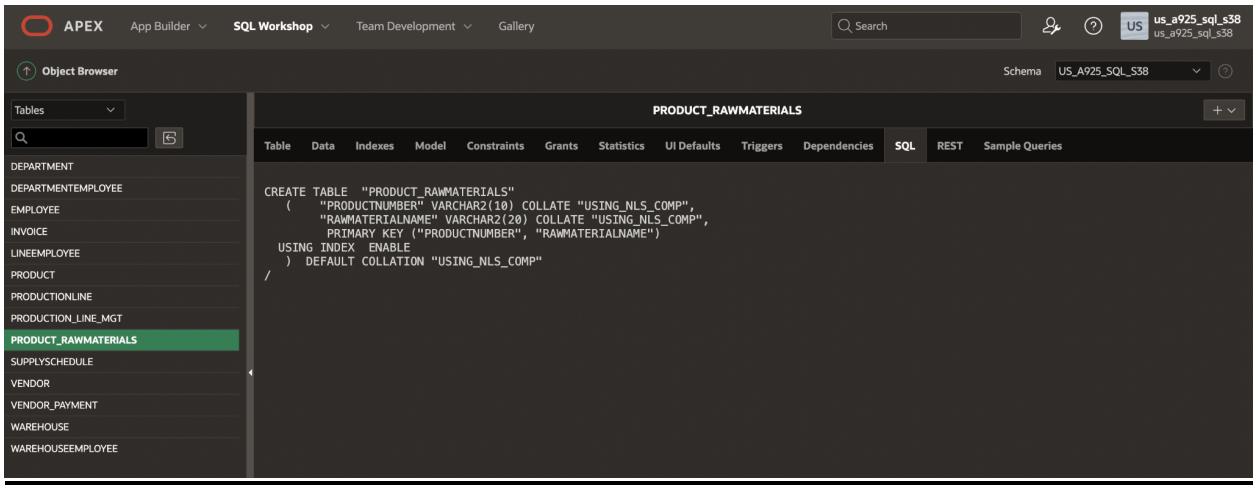


9.PRODUCT_RAWMATERIALS:

TABLE:

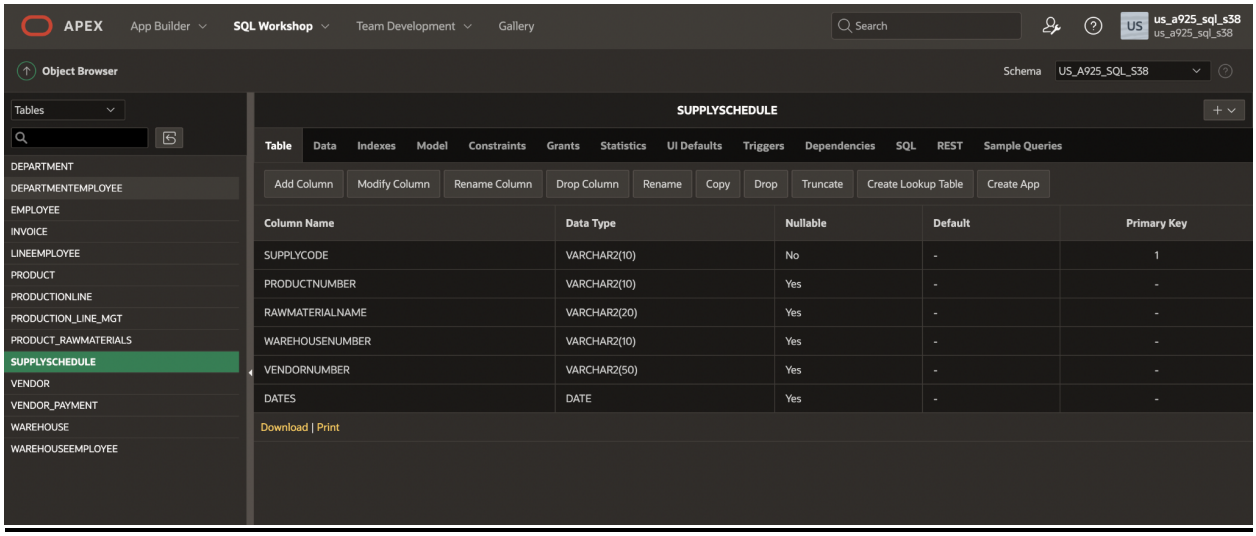


QUERY:

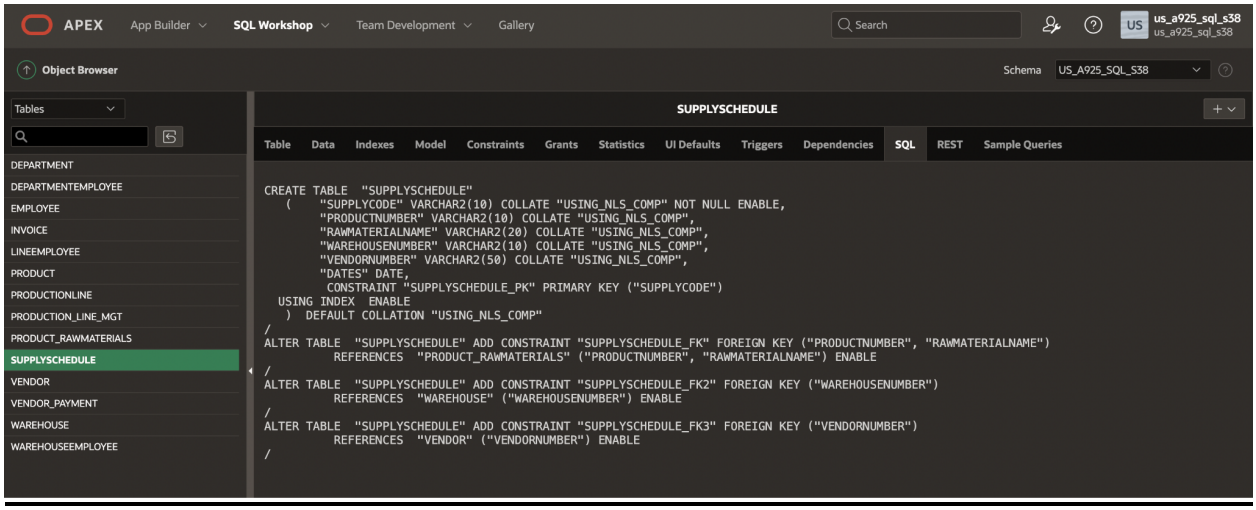


10.SUPPLYSCHEDULE:

TABLE:

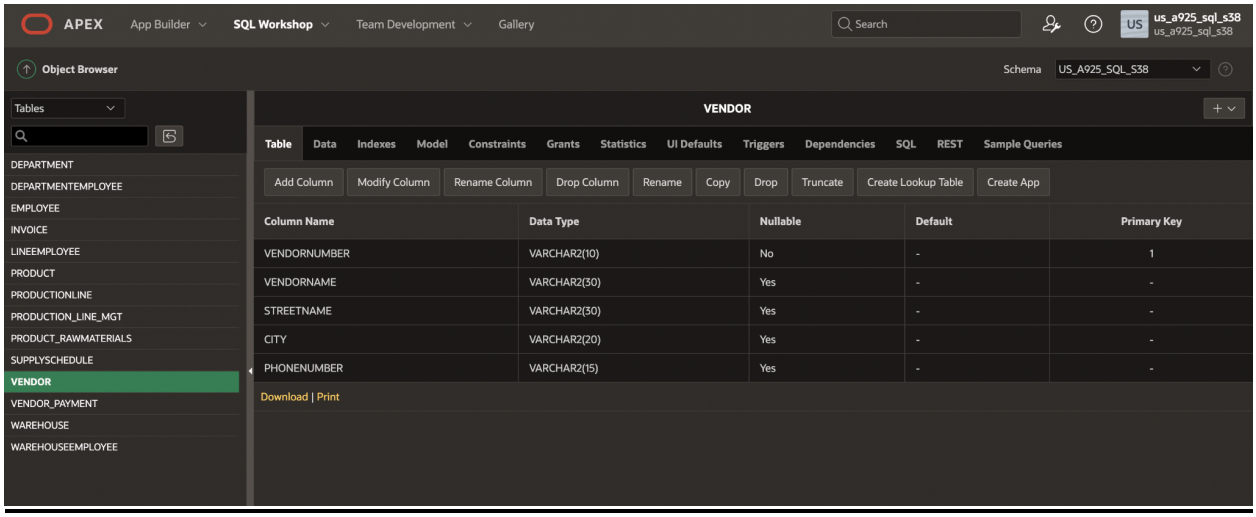


QUERY:

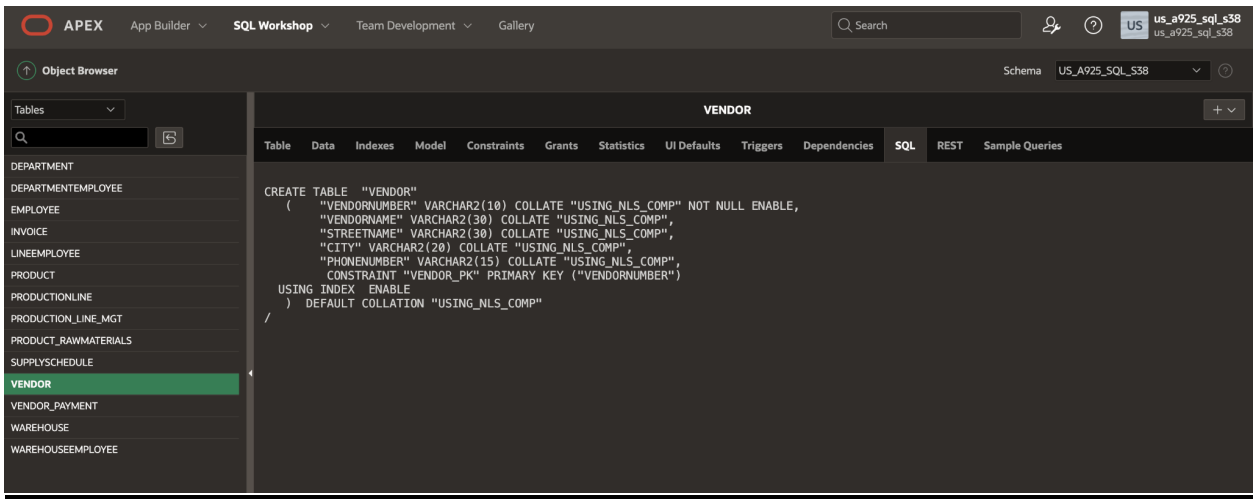


11.VENDOR:

TABLE:

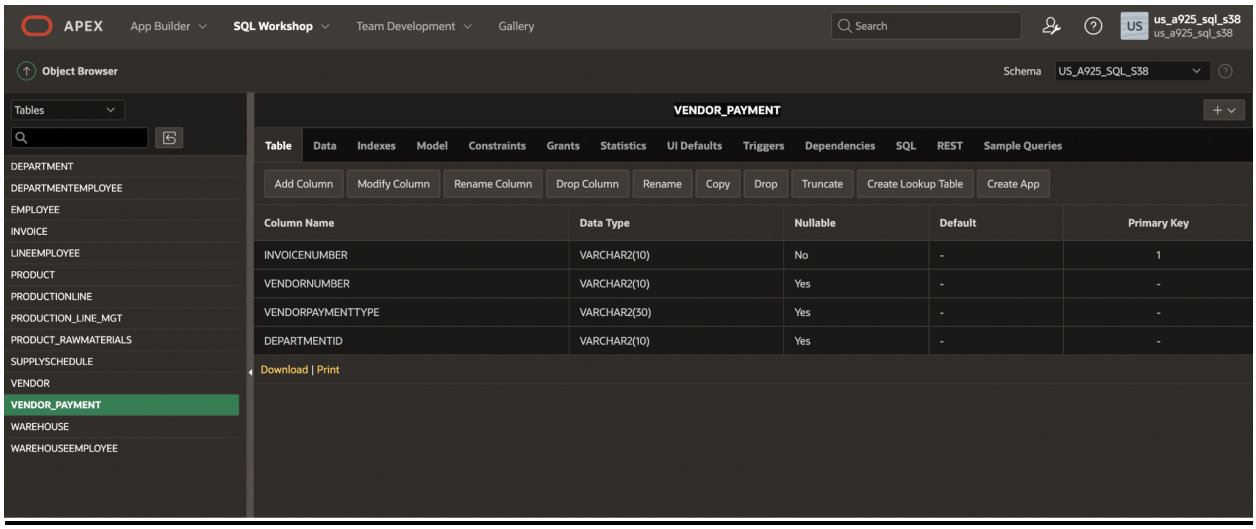


QUERY:

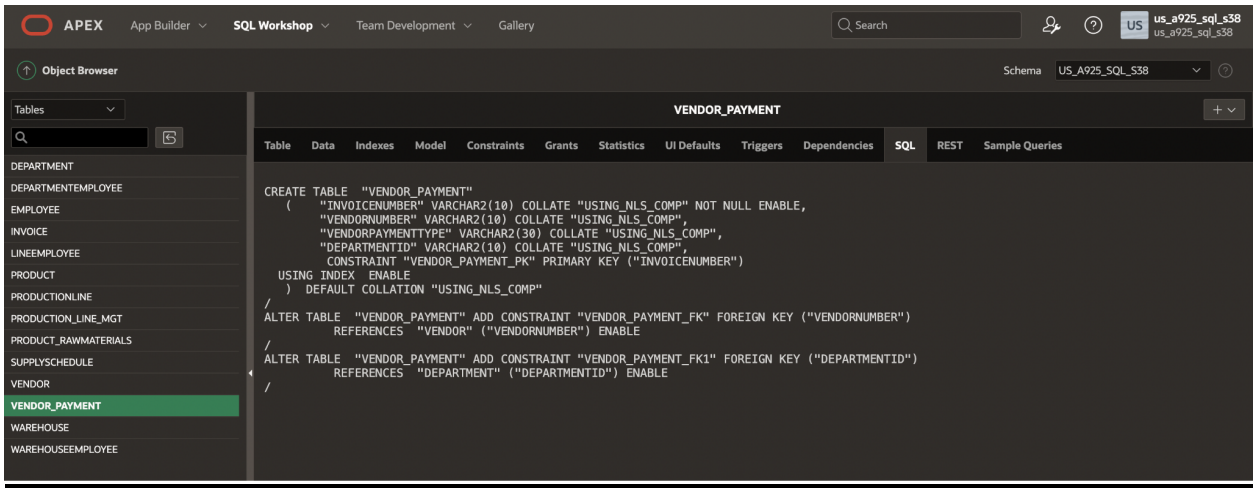


12.VENDOR_PAYMENT:

TABLE:

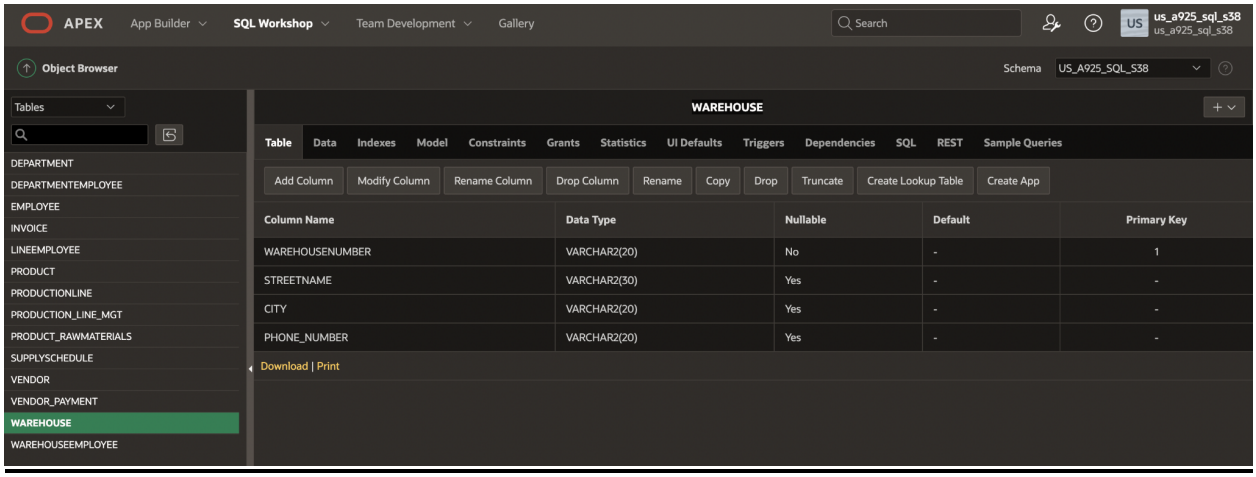


QUERY:

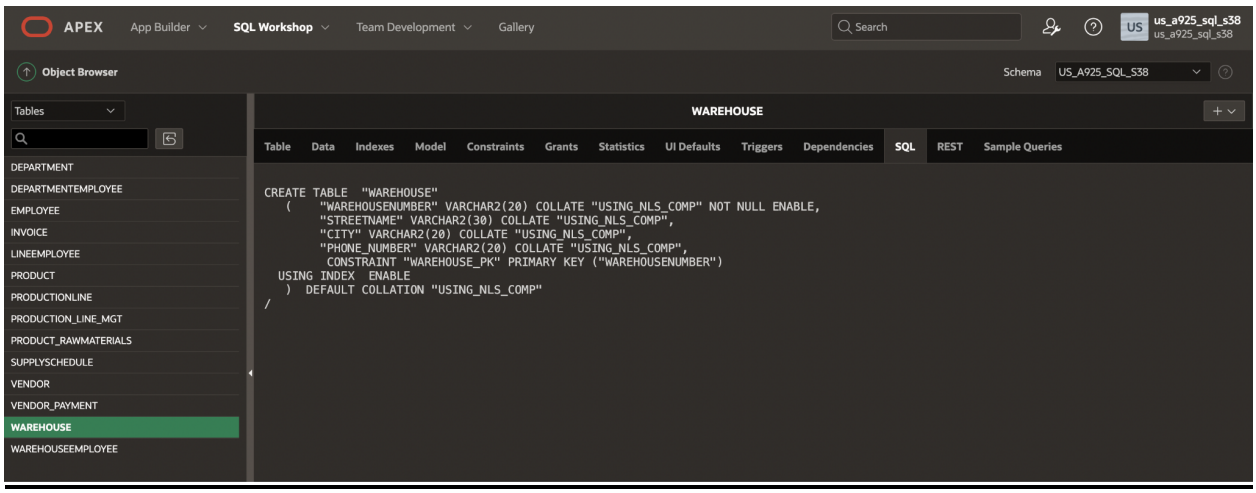


13.WAREHOUSE:

TABLE:

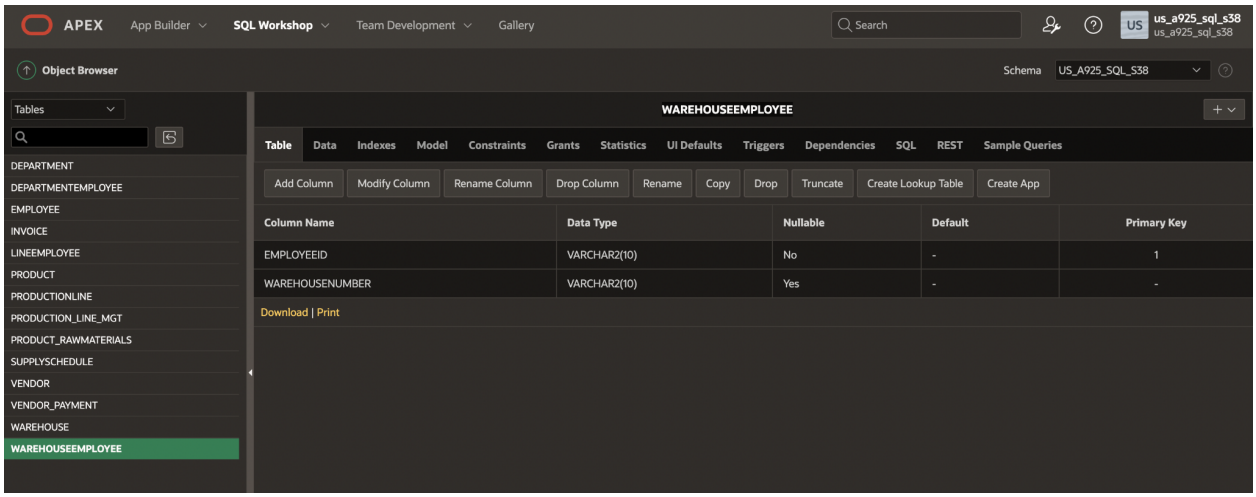


QUERY:

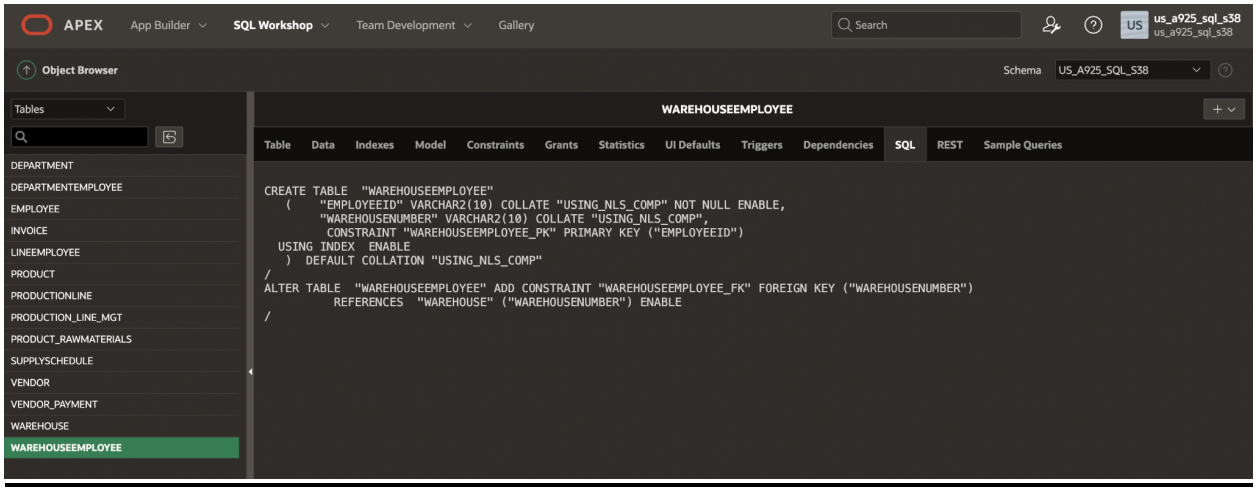


14.WAREHOUSEEMPLOYEE:

TABLE:



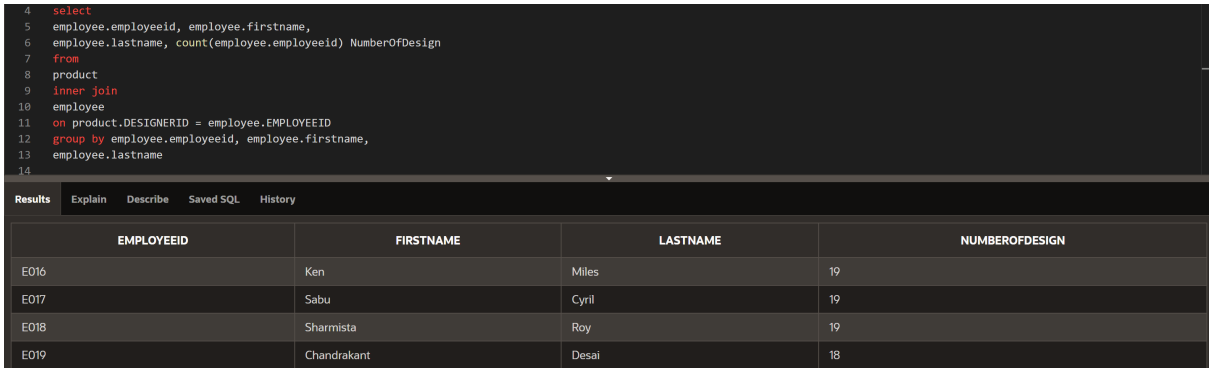
QUERY:



Use Case of Database:

Query 1: To find the number of products designed by each designer.

```
select
employee.employeeid, employee.firstname,
employee.lastname, count(employee.employeeid) NumberOfDesign
from
product
inner join
employee
on product.DESIGNERID = employee.EMPLOYEEID
group by employee.employeeid, employee.firstname,
employee.lastname
```



Results | Explain | Describe | Saved SQL | History

EMPLOYEEID	FIRSTNAME	LASTNAME	NUMBEROFDESIGN
E016	Ken	Miles	19
E017	Sabu	Cyril	19
E018	Sharmista	Roy	19
E019	Chandrakant	Desai	18

Query 2:
Query to get the details of Vendor submitting the maximum invoice value and its detail.

```
select
vendor_payment.INVOICENUMBER,
vendor_payment.VENDORNUMBER,
vendor_payment.VENDORPAYMENTTYPE,
invoice.total_amount,
Vendor.VENDORNAME,
Vendor.STREETNAME,
Vendor.CITY,
Vendor.PHONENUMBER
from invoice inner join vendor_payment
on
invoice.INVOICENUMBER = vendor_payment.INVOICENUMBER
inner join Vendor on Vendor.VENDORNUMBER = vendor_payment.VENDORNUMBER
where invoice.TOTAL_AMOUNT = (
select max(invoice.TOTAL_AMOUNT)
from invoice inner join vendor_payment
on invoice.INVOICENUMBER = vendor_payment.INVOICENUMBER
)
```

```
select
vendor_payment.INVOICENUMBER,
vendor_payment.VENDORNUMBER,
vendor_payment.VENDORPAYMENTTYPE,
invoice.total_amount,
Vendor.VENDORNAME,
Vendor.STREETNAME,
Vendor.CITY,
Vendor.PHONENUMBER
from invoice inner join vendor_payment
on
invoice.INVOICENUMBER = vendor_payment.INVOICENUMBER
inner join Vendor on Vendor.VENDORNUMBER = vendor_payment.VENDORNUMBER
where invoice.TOTAL_AMOUNT = (
select max(invoice.TOTAL_AMOUNT)
from invoice inner join vendor_payment
on invoice.INVOICENUMBER = vendor_payment.INVOICENUMBER
)
```

Results | Explain | Describe | Saved SQL | History

INVOICENUMBER	VENDORNUMBER	VENDORPAYMENTTYPE	TOTAL_AMOUNT	VENDORNAME	STREETNAME	CITY	PHONENUMBER
INV0017	V007	Cash	5750	Universal Prints	404 Galaxy Gt.	Unity City	(555)7890123

Query 3:
Group all raw material in descending order on the basis of their consumption.

select RAWMATERIALNAME, count(RAWMATERIALNAME) as Frequency
from PRODUCT_RAWMATERIALS
group by RAWMATERIALNAME
order by Frequency desc

Output:

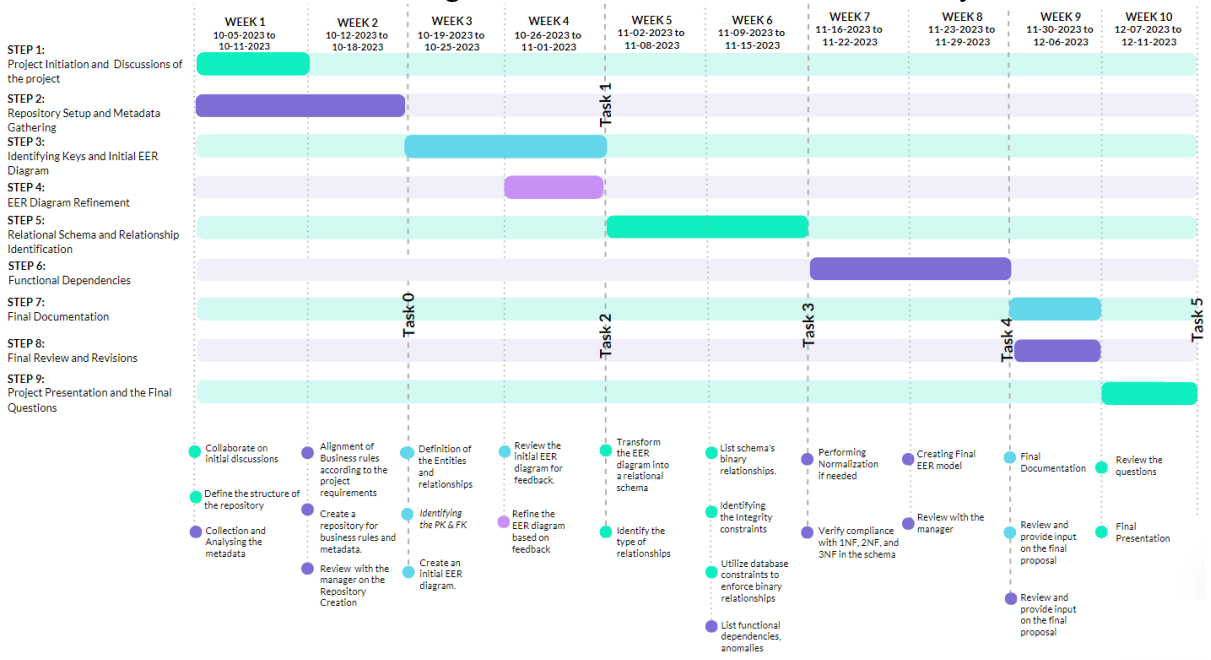
```
select RAWMATERIALNAME, count(RAWMATERIALNAME) as Frequency
from PRODUCT_RAWMATERIALS
group by RAWMATERIALNAME
order by Frequency desc
```

Results | Explain | Describe | Saved SQL | History

RAWMATERIALNAME	FREQUENCY
Plastic	21
Polyester	20
Leather	14
Cotton	14
Metal	12
Paper	9
Rubber	9
Vinyl	6
Wood	6

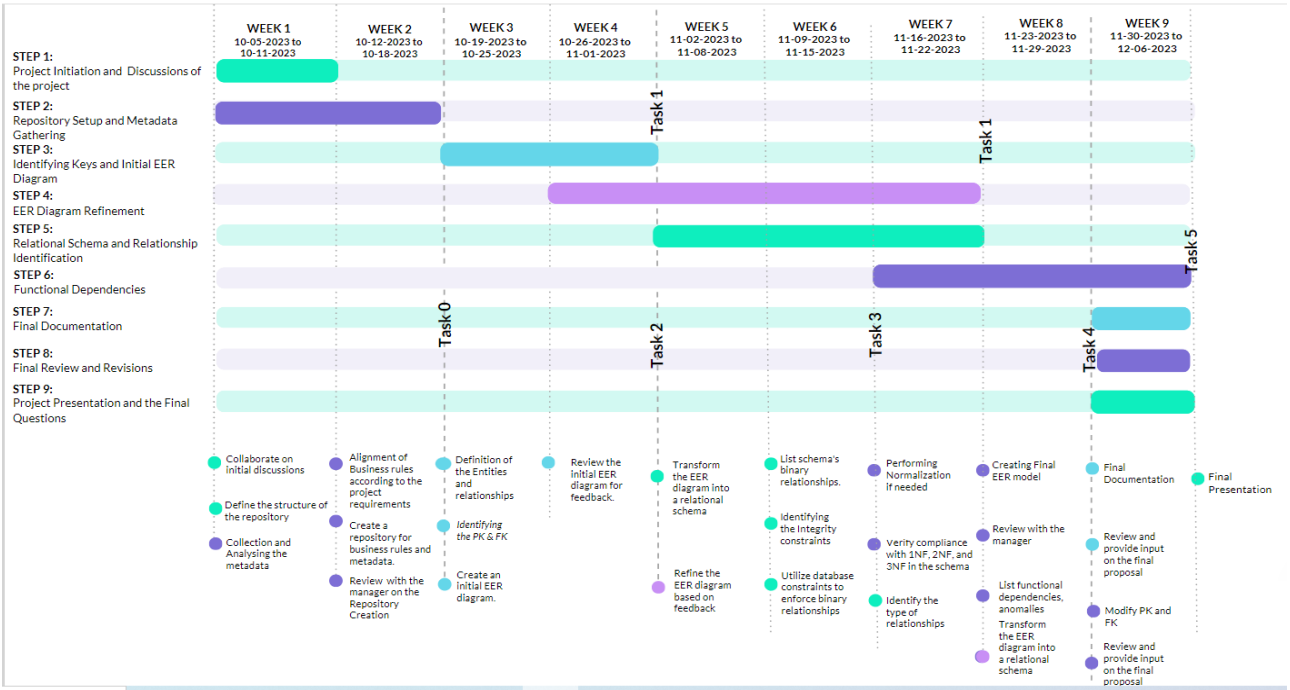
Timelines

Initially, we aimed to commence the database construction project with a meticulously designed timeline, as illustrated below. This plan was formulated based on our insights from the study, specifically considering the existing processes within your organization and acknowledging the complexity of the project. Subsequently, due to intricacies in the data and a few misconceptions, our plans underwent adjustments. We had initially outlined tasks to be completed in a specific order and timeframe, and the following timeline reflects the schedule we initialy intended to adhere



to.

The timeline reflects the actual work we did, including when "we" started and finished each task, but there we took time for Finalization of EER diagram and also there are some discrepancies in the “Invoice” data and a small discrepancy in the “Supply Schedule which we fixed it.



Additionally, all team members participated in development and quality assurance tasks. Final decisions were made through our discussions with Prof. Arman Falahati.

We convened in person every Friday on the 3rd floor of Abott Library, South Campus, University at Buffalo, New York, USA. During these meetings, we address the current status of ongoing tasks, upcoming assignments, identified any bottlenecks, and explored solutions.

Zoomlink to discuss the progress with professor:
https://buffalo.zoom.us/j/5541359212?pwd=V3l6UUo5TTFwQ2haTWMxY2NnOFdtZz09

Conclusion:

In conclusion, we designed a rigid solid database management system poised to enhance data storage, retrieval, and security, thereby fostering improved decision-making capabilities. By embracing an Agile methodology, we commit to fostering a collaborative partnership with your team, ensuring continuous engagement throughout the development stages. Leveraging a structured framework and an iterative process, we delivered a high-quality solution that aligns seamlessly with your organizational needs. We eagerly anticipate fruitful discussions driven by our mutual vision of realizing an Integrated Supply Chain and Financial Management System.

References:

[1] https://berteig.com/how-to-apply-agile/agile-methods-quick-reference-and-selection-guide__trashed/

[2] https://www.nutanix.com/info/databasemanagement#:~:text=A%20DBMS%20provides%20organizations%20a,risk%20of%20a%20data%20breach.