

INTEGRATED SUPPLY CHAIN AND FINANCIAL MANAGEMENT SYSTEM

Submitted to: University at Buffalo

MGS 613 – DBMS (December 2023)

Team 4 (SQL SYNDICATES)

Pradhakshana Duraiswamy
Anam Haseeb
Mini Palepu
Evan Li

Contents

| | |
|--|----|
| Introduction..... | 3 |
| TimeLine..... | 4 |
| Project Implementations | 5 |
| Task 0 | 5 |
| Establishing a unified Metadata and Business Rules Repository | 5 |
| TASK 1:..... | 10 |
| Strategic Identification of Primary and Foreign Keys in the EER Model..... | 10 |
| TASK 2: | 14 |
| Crafting a comprehensive Pre-Normalization EER Diagram | 14 |
| TASK 3: | 15 |
| Crafting a Relational Schema with comprehensive relationship definitions and Integrity | 15 |
| TASK 4: | 19 |
| Normalization and Functional Dependencies Analysis for Relational schema Design | 19 |
| TASK 5: | 24 |
| Crafting SQL queries for Data retrieval and Analysis | 24 |
| Conclusion | 41 |
| References..... | 42 |

Introduction:

In the dynamic landscape of contemporary manufacturing and supply chain operations, the SQL Syndicates team embarks on a transformative journey to articulate a robust Enhanced Entity-Relationship (EER) Database Schema. This endeavor is meticulously tailored to navigate the intricacies of a manufacturing enterprise, ensuring seamless coordination from product conceptualization to production, and meticulous oversight of internal structures and material procurement.

Project Overview

The primary objective of this project is to develop an EER Database Schema that not only streamlines workflow but also enhances the efficiency of a manufacturing entity. By focusing on the pivotal components of employee management, product lifecycle oversight, raw material flow, and meticulous handling of vendor invoices, the team aims to deliver a comprehensive solution that ensures peak performance, precise employee allocation, and meticulous financial control.

Key Components of the Database

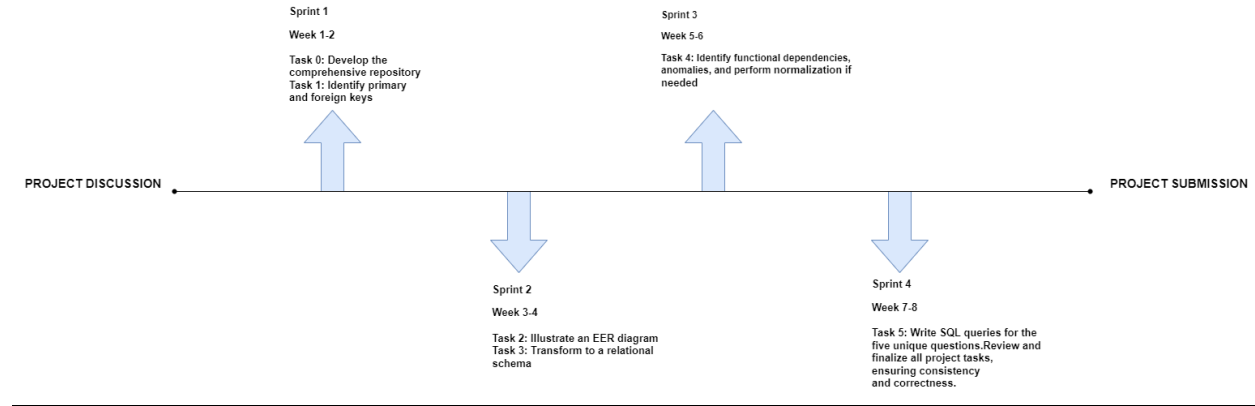
Employee Management: Our database will intricately manage employees across departments, warehouses, and production lines, ensuring every individual is distinctly allocated to a specific entity. This feature aims to optimize workforce efficiency and facilitate seamless collaboration.

Product Lifecycle Oversight: From the inception of designs in the conceptualization phase to the actual manufacturing on production lines, the database will provide comprehensive oversight of the entire product lifecycle. This ensures a smooth transition from design to production, enhancing overall operational efficiency.

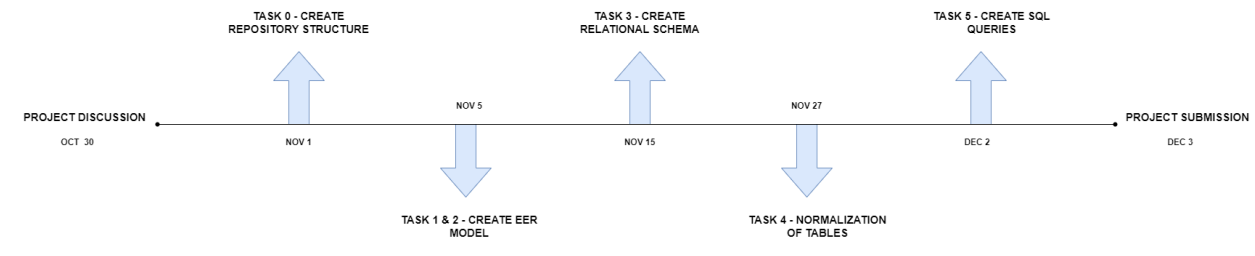
Raw Material Flow Management: Efficient handling of raw materials is critical in a manufacturing environment. The database will facilitate the management of material flow, whether sourced from warehouses or directly supplied by vendors, ensuring a constant and reliable supply chain.

Vendor Invoice Processing: A pivotal component of financial management, the database will adeptly handle the tracking and processing of vendor-supplied invoices. Clear sender identification and meticulous processing by the accounting department will be ensured, promoting financial transparency and efficiency.

Proposed Time Line



Actual Time Line



Project Implementations:

TASK 0

Establishing a Unified Metadata and Business Rules Repository.

Employee

| Element | Data Type | Length | Max | Min | Description | Source |
|------------------|-----------|--------|-------|---------|---|-----------------------|
| Employee ID | VARCHAR2 | 9 | | | Primary Key | AutoNumber |
| First Name | VARCHAR2 | 30 | | | A first name by which an employee can be known | HRM |
| Last Name | VARCHAR2 | 30 | | | A last name by which an employee can be known | HRM |
| Salary | NUMBER | 9 | 40000 | 1500000 | Periodic payment from an employer to an employee | Payroll |
| Position | VARCHAR2 | 30 | | | Job title of the employee based on job duties | HRM |
| Work Category | VARCHAR2 | 30 | | | Employer created job classifications | HRM |
| Department ID | VARCHAR2 | 9 | | | Foreign Key (Unique identifier of Department table) | Department table |
| Warehouse Number | VARCHAR2 | 9 | | | Foreign Key (Unique identifier of warehouse table) | Warehouse table |
| Line Number | VARCHAR2 | 9 | | | Foreign Key (Unique identifier of production line) | Production Line Table |

The EMPLOYEE table preserve important data information on an organization's workforce. To aligned with the objectives and standards of the business, it is insured that the consistency and accuracy of this table is handled with care. The columns Employee ID, First Name, Last Name, Position, Work Category, Department ID, Warehouse Number, and Line Number fields are all VARCHAR2 types, containing alphanumeric characteristics, and each character has a different length of data item. Salary column is NUMERIC, and has minimum and maximum values to ensure that correct value is entered.

Department

| Element | Data Type | Length | Max | Min | Description | Source |
|---------------------|-----------|--------|-----|-----|---|------------------------------|
| Department ID | VARCHAR2 | 4 | | | PK- Unique identifier | Auto generated (Dept. Table) |
| Department Name | VARCHAR2 | 50 | | | Functional divisions in an organization | HRM |
| Location | VARCHAR2 | 50 | | | A place where company's divisions are located | HRM |
| Number of Employees | NUMBER | 2 | | | Number of employees by department | HRM |
| Phone Number | NUMBER | 10 | | | Employee's contact number | HRM |

The DEPARTMENT table holds important data regarding organization's divisions; therefore, the structure of the data is consistent, accurate and precise. The Number of Employees and Phone Number columns are NUMERIC, containing numerical values, while Department ID, Department Name, Location, are all VARCHAR2, carrying alphabetic characteristics.

Product

| Element | Data Type | Length | Max | Min | Description | Source |
|----------------|-----------|--------|-----|------|---|-----------------------------------|
| Product Number | VARCHAR2 | 5 | | | PK- Unique identifier | Auto Generated |
| Product Type | VARCHAR2 | 30 | | | Category of a product | Manufacturing department |
| Product Name | VARCHAR2 | 100 | | | Identifies and distinguishes a product from other products | Design department |
| Designer ID | VARCHAR2 | 4 | | | Identifier for the designer who designed the specific product | HRM |
| Price | NUMBER | 3 | 150 | 4 | Amount at which the product is sold to the end customer. | Sales and Marketing Department |
| Cost | NUMBER | 2 | 90 | 2 | Cost of goods sold | Sale and Manufacturing department |
| Color | VARCHAR2 | 6 | | | Product with a particular color | Manufacturing department |
| Weight(lbs) | NUMBER | 4 | 2.0 | 0.02 | Weight of the product, | Manufacturing dept |

The Product Table provides a structured format to store product details systematically and organized. Each row in the table represents a unique product offered, and each column corresponds to the product's specific attributes. These attributes range from essential data for organizing product information efficiently. Product Number, Product Type, Product Name, Designer ID, and Color is VARCHAR2 showing that data carrying alphabetic characteristics, while Price, Cost, Weight (lbs.) are NUMERIC type and can contain decimal numbers. Each characteristic has a different length of data item, and Price, Cost, Weight has minimum and maximum values to ensure a correct range is entered, as well as correct precision and scale.

Raw Material

| Element | Data Type | Length | Max | Min | Description | Source |
|-------------------|-----------|--------|-----|-----|-------------|----------------------|
| Product Number | VARCHAR2 | 10 | | | PK | Product table |
| Raw Material Name | VARCHAR2 | 50 | | | PK | Vendor and warehouse |
| Line Number | VARCHAR2 | 10 | | | FK | Production line |

A raw material table stores information about the raw materials that are used to manufacture a product. To ensure the accuracy and complete identification of raw material, the data related to this table is stored under composite primary key Product Number, Raw Material Name, and Line Number data type is VARCHAR2, which shows that it carries alphabetic characteristics.

Production Line

| Element | Data Type | Length | Max | Min | Description | Source |
|---------------|-----------|--------|-----|-----|--|------------------------------------|
| Line Number | VARCHAR2 | 4 | | | PK | Auto Generated |
| Line Capacity | NUMBER | 3 | 230 | 130 | The limit at which products can be manufactured. | Line Manager/ Supervisor HRM |
| Phone Number | NUMBER | 10 | | | Contact number for production line | Line Manager/ Supervisor HRM |
| Address | VARCHAR2 | 100 | | | The place where the production line is located. | Line Manager/ Supervisor HRM |

A production line table in a database is a table that stores information about the processes and operations involved in producing a product or service. It can help to track the progress. Line Number, Address contains VARCHAR2 data type, and Line Capacity and Phone Number holds NUMERIC type and minimum maximum value requirement to input correct value.

Vendor

| Element | Data Type | Length | Max | Min | Description | Source |
|---------------|-----------|--------|-----|-----|---|---------------------|
| Vendor Number | VARCHAR2 | 4 | | | Unique identifier identifies the source of the raw material- PK | Registration Number |
| Vendor Name | VARCHAR2 | 100 | | | The name of a business or individual vendor that provides raw material. | Registration Number |
| Address | VARCHAR2 | 100 | | | The place where the vendor's business is located. | Registration Number |
| Phone Number | NUMBER | 10 | | | Contact Number for the Vendor | Registration Number |

A VENDOR table is a database table that stores information about the vendors that supply an enterprise. It contains data about vendors in order to maintain their accurate and complete records. Vendor Number, Vendor Name, and Address carrying VARCHAR2 data type, defines alphanumeric characteristics, and while Phone Number is NUMERIC, all of them hold different length.

Warehouse

| Element | Data Type | Length | Max | Min | Description | Source |
|------------------|-----------|--------|-----|-----|---|----------------|
| Warehouse Number | VARCHAR2 | 5 | | | Unique identifier - PK | Auto Generated |
| Address | VARCHAR2 | 100 | | | The place where the warehouse is located. | HRM |
| Phone Number | NUMBER | 10 | | | Contact Number for the warehouse | HRM |

The WAREHOUSE table provides information about organization's storehouse, where products or materials can get stored before they move to another location. It's a crucial element of supply chain management. The data related to Warehouse Number and Address are stored as VARCHAR2 type and Phone Number as NUMERIC, each characteristic has a different length of data item.

Invoice

| Element | Data Type | Length | Max | Min | Description | Source |
|---------------------|-----------|--------|------|-----|---|----------------|
| Invoice Number | VARCHAR2 | 10 | | | Unique record number assigned to each invoice. | Auto Generated |
| Total Amount (\$) | NUMBER | 10 | 5750 | 750 | The amount due to the vendor | Purchase order |
| Vendor Number | VARCHAR2 | 10 | | | FK | Purchase order |
| Vendor Payment Type | VARCHAR2 | 50 | | | The mode of payment through which a vendor will be able to receive the amount | Purchase order |

An invoice table is important because it stores information about the transactions between Vendor and Organization. To provide continuous validity of data in this table, Total Amount (\$) column contains NUMERIC data type with minimum and maximum values to ensure the requirement of input values are correct. Invoice Number, Vendor Number, and Vendor Payment Type carries VARCHAR2 datatypes, and each characteristics has a different length of data item.

Supply schedule

| Element | Data Type | Length | Max | Min | Description | Source |
|-------------------|-----------|--------|-----|-----|--|--------------------|
| Supply Code | VARCHAR2 | 10 | | | PK (Unique Identifier) | Auto Generated |
| Product Number | VARCHAR2 | 10 | | | FK | Product Table |
| Raw Material Name | VARCHAR2 | 50 | | | FK | Raw Material Table |
| Warehouse Number | VARCHAR2 | 10 | | | FK | Warehouse table |
| Vendor Number | VARCHAR2 | 10 | | | FK | Vendor Table |
| Date | VARCHAR2 | 10 | | | Date to track when the raw materials were supplied | Purchase order |

A supply schedule table is used to determine how much Raw Material have been supplied from which vendor and warehouse for manufacturing the product and helps to track the date and time of the supply. To track this information accurately and completely each characteristic is specified with certain data type and length.

TASK 1

Strategic Identification of Primary and Foreign Keys in the EER Model.

| Table-wise Key Identification | Primary Key/Keys | Foreign Key/Keys |
|--------------------------------------|-------------------------------------|--|
| Product | Product Number | Designer ID |
| Production Line | Line Number | |
| Raw Material | (Product Number, Raw Material Name) | |
| Raw Material New | Product Number | Line Number |
| Employee | Employee ID | |
| Line Employee | Employee ID | Line Number |
| Warehouse Employee | Employee ID | Warehouse Number |
| Department Employee | Employee ID | Department ID |
| Department | Department ID | |
| Invoice | Invoice Number | (Vendor Number, Department ID, Payment ID) |
| Payment | Payment ID | Vendor Number |
| Warehouse | Warehouse Number | |
| Vendor | Vendor Number | |
| Supply Schedule | Supply Code | Product Number, Raw Material name, Warehouse Number, Vendor Number |

Product Table Key Identification:

Primary Key: Product Number

Within the Product table, the Product Number assumes the role of the primary key. This designation is attributed to its unique identification capability, ensuring that each row in the product table is distinctively identified [1].

Foreign Key: Designer ID

In accordance with the company's operational guidelines, where products are designed and manufactured by employees within the design department, the Designer ID serves as a foreign key in the Product table. Under the established business rules, each designer can be responsible for multiple products, while each product is uniquely associated with a single designer. This arrangement establishes a 1:M (One-to-Many) relationship between the Department Employees and Product tables.

As the Product side encompasses multiple instances, and department employees represent the mandatory one side, a foreign key (Designer ID) has been integrated into the Product table. This foreign key references the specific employee within the design department responsible for designing the respective product, creating a linkage that aligns with the established business rules.

Production Line Table Key Identification:

Primary Key: Line Number

The Line Number is designated as the primary key for the Production Line table. During our data analysis, it became evident that the rows associated with the line number were not repetitive. However, upon closer examination, we observed repetitive entries in line capacity, phone number, and address. Consequently, it was deemed unsuitable to select any of these attributes as a unique identifier, necessitating the utilization of Line Number as the primary key for maintaining distinctiveness in the table.

Raw Material Table Key Design:

Primary Key: Composite keys (Product Number, Raw Material Name)

The adoption of a composite primary key is essential, given the repetitive nature of individual attributes within the Raw Material table.

Raw Material New Table Key Design:

Primary Key: Product Number

A strategic shift has been made in the Raw Material New table, focusing on the singular Product Number as the primary key.

Foreign Key: Line Number

The introduction of the Raw Material New table facilitates the removal of partial dependency by shifting the Line Number attribute to serve as a foreign key. This restructuring enhances data normalization and integrity within the database schema.

Employee Table Key Identification:

Primary Key: Employee ID

The Employee ID stands as the primary key in the Employee table, providing a unique identifier for each row. Upon examination of additional columns such as employee's first name, last name, position, salary, and work category, it became apparent that these attributes lack the necessary uniqueness to qualify as a primary key. Issues such as repetitive and null values emerged, reinforcing the rationale that Employee ID remains the sole identifier capable of ensuring distinctiveness among rows in the table.

Establishing Subtype Relations in Employee Categorization: A Specialization Approach

In alignment with the outlined business rules, which define the organizational structure of departments, warehouses, and production lines, we have adopted a Supertype-Subtype relationship under Specialization [2] for Line Employees, Warehouse Employees, and Department Employees. This strategic decision is aimed at mitigating the occurrence of NULL values within the EMPLOYEE table.

Our approach involves the creation of subtypes based on the distinct work categories of employees within Production Lines, Warehouses, and Departments. Subsequent analysis revealed a Total Specialization,

covering all work categories and ensuring that each employee belongs to precisely one subtype. This realization led us to enforce a disjoint constraint.

Production Line Employee:

Primary Key: Employee ID

This attribute serves as the unique identifier for employees associated with production lines.

Foreign Key: Line Number

A 1:M relationship exists between the Production Line and Production Line Employee tables. By referencing Line Number, we establish a clear connection, specifying which Production Line Employee works in a particular production line.

Warehouse Employee:

Primary Key: Employee ID

Uniquely identifying warehouse employees within this subtype.

Foreign Key: Warehouse Number

Reflecting the relationship between Warehouse Employee and Warehouse tables, the foreign key in Warehouse Employee allows us to determine which warehouse an employee is associated with.

Department Employee:

Primary Key: Employee ID

Serving as the unique identifier for department employees.

Foreign Key: Department ID

As stipulated by business rules, employees work in one department. This necessitates a foreign key in the Department Employee table, referencing the Department ID in the Department table to determine the department in which an employee works.

By implementing these primary and foreign key relationships, we establish a robust and normalized database structure that aligns seamlessly with the operational intricacies outlined in the business rules.

Department Table Key Identification:

Primary Key: Department ID

Functioning as a unique identifier for Department Name, Location, Number of Employees, and Phone Number.

Invoice Table Key Identification:

Primary Key: Invoice Number

Foreign Keys: (Vendor Number, Department ID, Payment ID)

The Invoice table plays the many side in relationships with the Department and Vendor tables. Foreign keys, namely Vendor Number and Department ID, establish crucial connections with their respective tables. Additionally, the introduction of Payment ID as a foreign key ensures a comprehensive linkage to the Payment table.

Payment Table Key Identification:

Primary Key: Payment ID

Foreign Key: Vendor Number

To address transitive dependency concerns, the Invoice table has been strategically divided into separate Invoice and Payment tables. The Payment table now boasts its own primary key (Payment ID) and maintains a foreign key relationship with the Vendor table, enhancing data normalization and refining the database structure.

Warehouse Table Key Identification:

Primary Key: Warehouse Number

Serving as the unique identifier for each row in the Warehouse table.

Vendor Table Key Identification:

Primary Key: Vendor Number

Identified as the most feasible primary key due to the complexity of combining multiple attributes into a composite key. The surrogate primary key ensures efficiency.

Supply Schedule Table Key Identification:

Primary Key: Supply Code

Employing a primary key was deemed optimal to create a unique identifier for the Supply Schedule, given the challenge of combining more than two composite attributes.

Foreign Keys: (Product Number, Raw Material Name, Warehouse Number, Vendor ID)

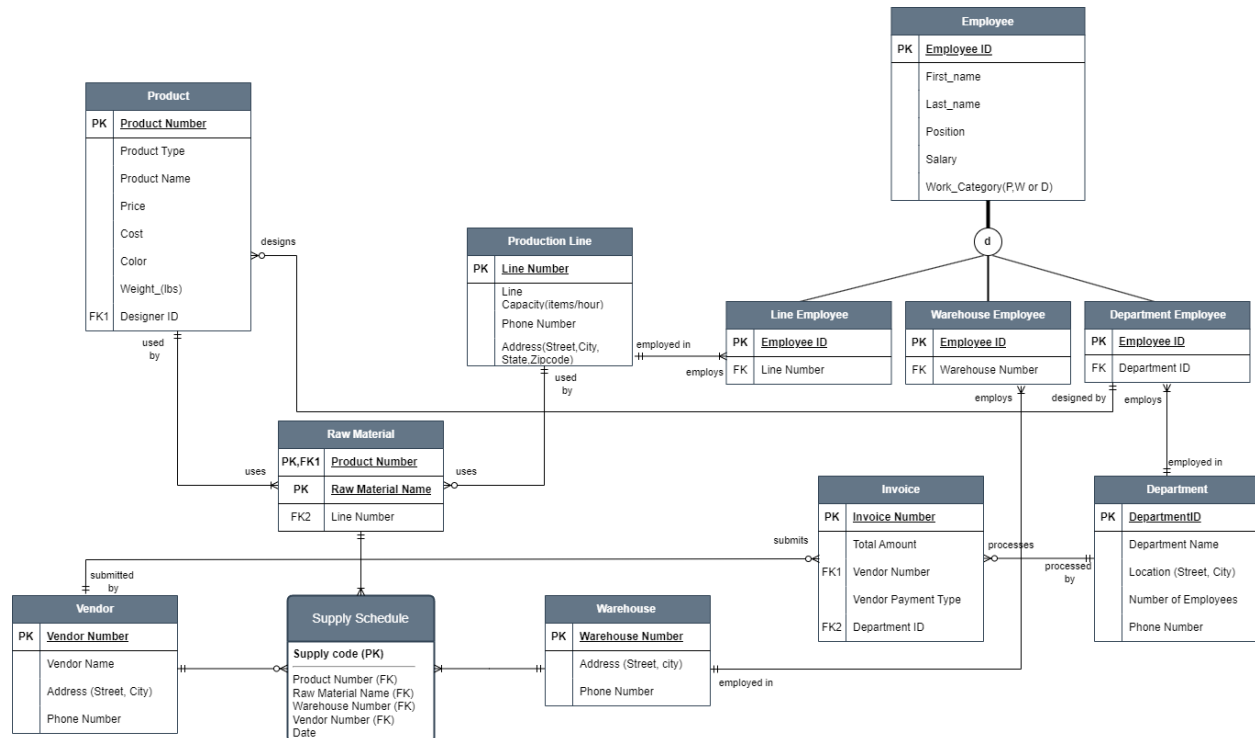
These primary and foreign key relationships establish a well-structured database design, aligning seamlessly with the specific requirements of each table and fostering effective data management within the system.

TASK 2

Crafting a Comprehensive Pre-Normalization EER Diagram.

Comprehensive EER Diagram:

In the below presented EER diagram, a meticulous depiction of relationships, attributes, and cardinalities among various entities has been crafted. This preliminary rendition serves as a pre-normalization model, with functional dependencies and anomalies yet to be addressed.



The employee table has undergone a strategic normalization process through specialization techniques, resulting in three distinct tables: Line Employee, Warehouse Employee, and Department Employee. Each of these tables retains their special attributes, linked to the primary key EmployeeID from the parent Employee table. This approach adheres to a total specialization and disjoint structure.

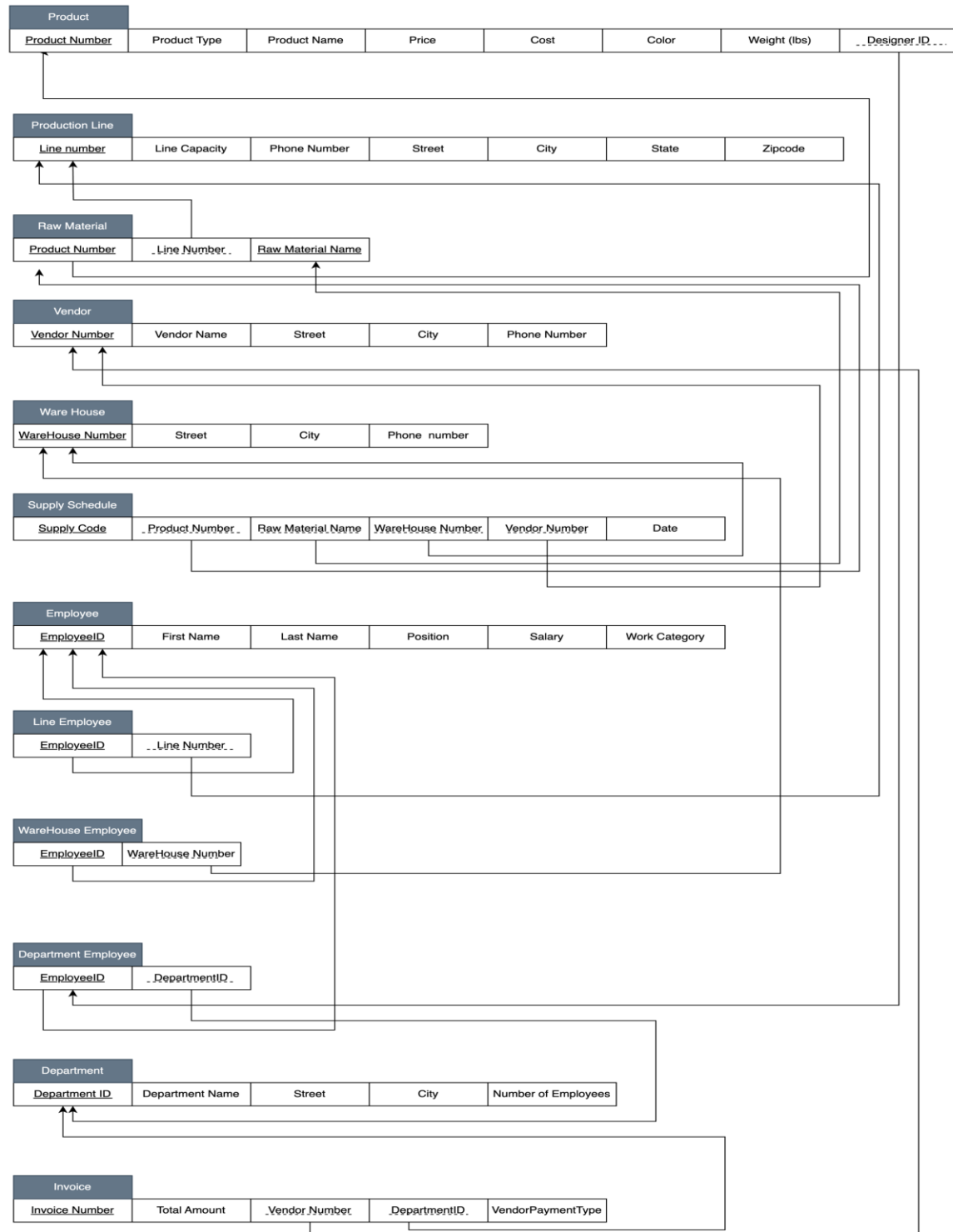
Addressing a ternary relationship, an associative entity has been introduced, effectively breaking it into three binary relationships. This ensures a streamlined representation within the EER diagram, maintaining clarity and precision.

Foreign keys and their corresponding primary keys have been meticulously noted throughout the diagram. Composite attributes, such as Address represented as (Street, City, State, Zip code), are presented in a clear and concise manner.

As we move forward, our focus will shift towards normalizing the model, addressing functional dependencies, and eliminating anomalies. The resulting refined EER model will embody a robust and efficient database structure, poised to meet the specific requirements of our project.

TASK 3

Crafting a Relational Schema with Comprehensive Relationship and Integrity Constraints.



In translating the conceptual EER diagram into a relational schema. Notably, no unary relationships were found, while a single ternary relationship emerged from the business rules pertaining to the interaction between vendors, raw materials, and warehouses.

Ternary Relationship:

The ternary relationship arises from the business rules B2-1 to B2-4,

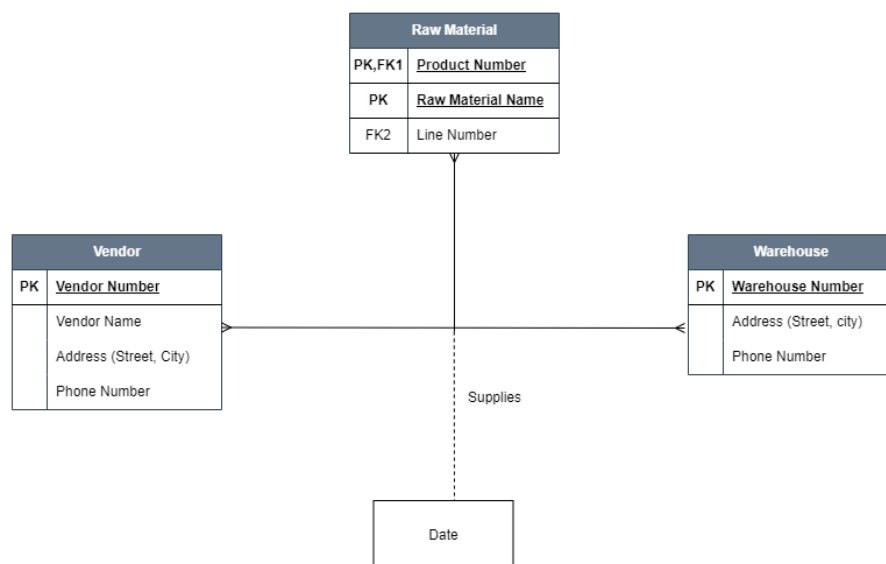
B2-1: Vendors supply raw materials to warehouses.

B2-2: Raw materials are supplied by warehouses and vendors.

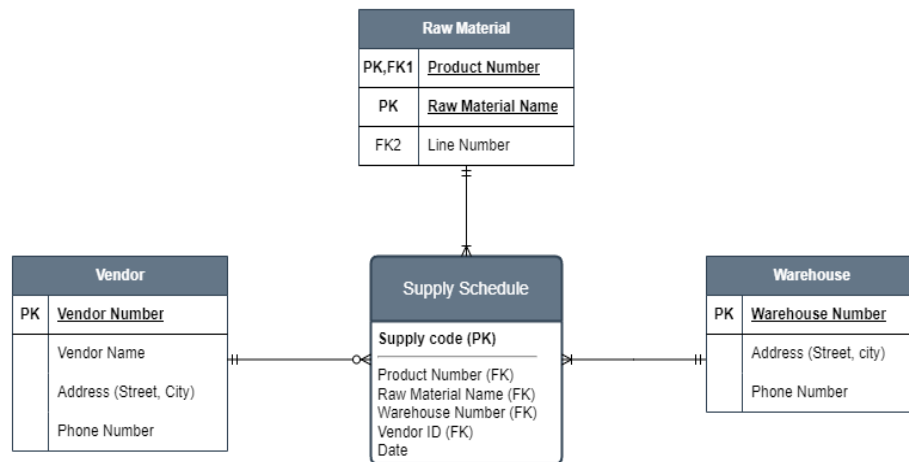
B2-3: Raw materials can be directly supplied by vendors.

B2-4: Warehouses can be supplied with multiple raw materials from various vendors but each warehouse must be supplied with at least one raw material.

which implies an association among three different entities [3] vendors, raw materials, and warehouses. To simplify and enhance data integrity, an associative entity named Supply Schedule was introduced. Supply Schedule features a primary key (Supply Code) and foreign keys (Product Number, Raw Material Name, Warehouse Number, Vendor ID) along with an attribute Date. This transformation decomposes the original ternary relationship into three distinct binary relationships, improving the overall structure and clarity of the relational schema.



After the addition of Associative Entity:



Binary Relationships:

A series of binary relationships were identified from the EER diagram apart from the three binary relationships that resulted from adding an associative entity, and they are as follows:

| TABLE 1 | TABLE 2 |
|---------------------|---------------------|
| Product | Raw Material |
| Production Line | Raw Material |
| Production Line | Line Employee |
| Warehouse | Warehouse Employee |
| Department | Department Employee |
| Vendor | Invoice |
| Department | Invoice |
| Department Employee | Product |

Database Integrity Constraints

In establishing a robust and dependable database schema, several integrity constraints have been implemented to ensure data accuracy, consistency, and reliability. These constraints fall into three primary categories: entity constraints, referential integrity constraints, and domain integrity constraints.

Referential Integrity Constraints:

All foreign keys within the schema have been meticulously mapped to their corresponding primary keys, establishing a robust framework for referential integrity.

Relationships, such as those between entities like Vendor-Invoice, Department-Invoice, and Department Employee-Product, are maintained through foreign key constraints, ensuring the validity of relationships between related tables.

Entity Integrity Constraints:

Primary keys, acting as unique identifiers for each entity, have been carefully crafted to eliminate null values and enforce uniqueness. For example, composite primary keys, such as (Product Number, Raw Material Name) in the Raw Material table, have been employed where necessary to maintain entity integrity.

Domain Integrity Constraints:

The metadata provides a comprehensive outline of allowable types of values, character lengths, and acceptable maximum and minimum values for each attribute within the database schema.

Data types specified during table creation align with the defined domain constraints, ensuring that values inserted into the database conform to the expected types and ranges.

Nullability constraints, whether a column allows null values or not, have been explicitly indicated during the table creation process. This ensures that fields essential for establishing relationships or serving as primary keys are not left null.

This proactive approach to integrity constraints will contribute significantly to the consistency and accuracy of the database as it evolves and grows over time.

TASK 4

Normalization and Functional Dependencies Analysis for Relational Schema Design.

Functional Dependencies, Anomalies, and Normalization

In the pursuit of database integrity and efficiency, an analysis of functional dependencies, anomalies, and subsequent normalization processes was conducted for each relation. Functional dependencies were categorized into partial dependencies and transitive dependencies, and anomalies were identified, prompting the need for normalization. The entire process aligns with the fundamental principles of normalization, specifically achieving 1NF, 2NF, and 3NF.

Functional Dependencies:

Partial Dependencies:

Definition: In the context of relational databases, a partial dependency occurs when one or more non-prime attributes are functionally dependent on only part of the primary key.

Example: The Raw Material table presented a partial dependency where the line number was dependent on the product number, indicating the need for a more refined structure.

Transitive Dependencies:

Definition: A transitive dependency exists when a non-prime attribute is functionally dependent on another non-prime attribute, creating a chain of dependencies beyond the primary key.

Example: The Invoice table exhibited transitive dependency, with the non-key attribute Vendor Payment Type relying on the non-key attribute Vendor Number, which, in turn, depended on the primary key attribute Invoice Number.

Full Dependencies:

For all other relations, full dependencies exist, implying that all non-prime attributes are fully functionally dependent on every candidate key. This conforms to the standard normalization principles.

Anomalies:

Modification Anomaly:

Definition: A modification anomaly occurs when making changes to the database, such as updates, leads to inconsistencies or challenges due to the structure of the tables.

The Raw Material table presents a modification anomaly due to the repetitive values in the Product Number. If an update to the Line Number of a product is required, it necessitates updates in multiple places. This creates challenges in maintaining consistency and can lead to errors during modification operations.

Deletion Anomaly:

Definition: A deletion anomaly occurs when removing a row from a table results in the loss of critical data, impacting the integrity and completeness of the dataset.

The Invoice table exhibits a deletion anomaly, where removing a row could result in the loss of critical data, such as the Total Amount, associated with a specific vendor. This deletion could adversely impact the vendor's business, financial records, and overall transaction history within the company.

Normalization was initiated to address the observed anomalies and improve overall database organization and efficiency.

Normalization Process Overview

Normalization is a critical step in database design, aiming to eliminate redundancy and enhance data integrity. The process involves achieving successive normal forms (1NF, 2NF, 3NF) by addressing specific conditions.

Step 1: Achieving 1NF

Condition: All tables requires that there be no multi-valued attributes, and that there be no repeating groups [4]. The data cleaning and splitting of composite and multi-valued attributes align with the requirements of 1NF.

Action: The Raw Material, initially a multivalued attribute of the Product table, was separated into a new table named Raw Material to satisfy the 1NF condition.

| | Raw Material |
|---------|-------------------|
| PK, FK1 | Product number |
| PK | Raw Material Name |
| FK2 | Line Number |

Step 2: Achieving 2NF

Condition: No partial dependencies. All non-prime attributes should be fully functionally dependent on the primary key. The division and restructuring of tables to address anomalies contribute to achieving 2NF.

Action: The Raw Material table was divided into two separate tables to eliminate partial dependencies. Below are the two tables:

| | Raw Material |
|----|-------------------|
| PK | Product number |
| PK | Raw Material Name |

| | Raw Material New |
|----|------------------|
| PK | Product number |
| FK | Line Number |

Step 3: Achieving 3NF

Condition: No transitive dependencies should exist where non-prime attributes are functionally dependent on other non-prime attributes.

Action: To remove transitive dependencies, a new primary key, 'Payment ID,' was introduced. The Invoice table was adjusted accordingly.

| | Invoice |
|----|----------------|
| PK | Invoice number |
| | Total Amount |
| FK | Vendor Number |
| FK | Department ID |
| FK | Payment ID |

| | Payment |
|----|---------------------|
| PK | Payment ID |
| FK | Vendor Number |
| | Vendor Payment Type |

Normalization has significantly improved data integrity by eliminating redundancy and addressing anomalies in the Raw Material and Invoice tables. The final EER model or Relational Schema adheres to 1NF, 2NF, and 3NF conditions, ensuring a well-structured and normalized database that aligns with the project's objectives.

Data Cleaning:

The normalization process in this project involved several steps of data cleaning to enhance data integrity and remove redundancies. Some key transformations and cleaning activities include:

Splitting Composite Attributes:

Composite attributes like "Address" were divided into distinct components such as Street, City, State, and Zip code. This separation improves clarity, ease of retrieval, and adherence to normalization principles.

Phone Number Formatting:

Special characters in attributes, specifically in phone numbers, were removed to standardize the format. This ensures uniformity and simplifies data handling.

Normalization and Division of Tables:

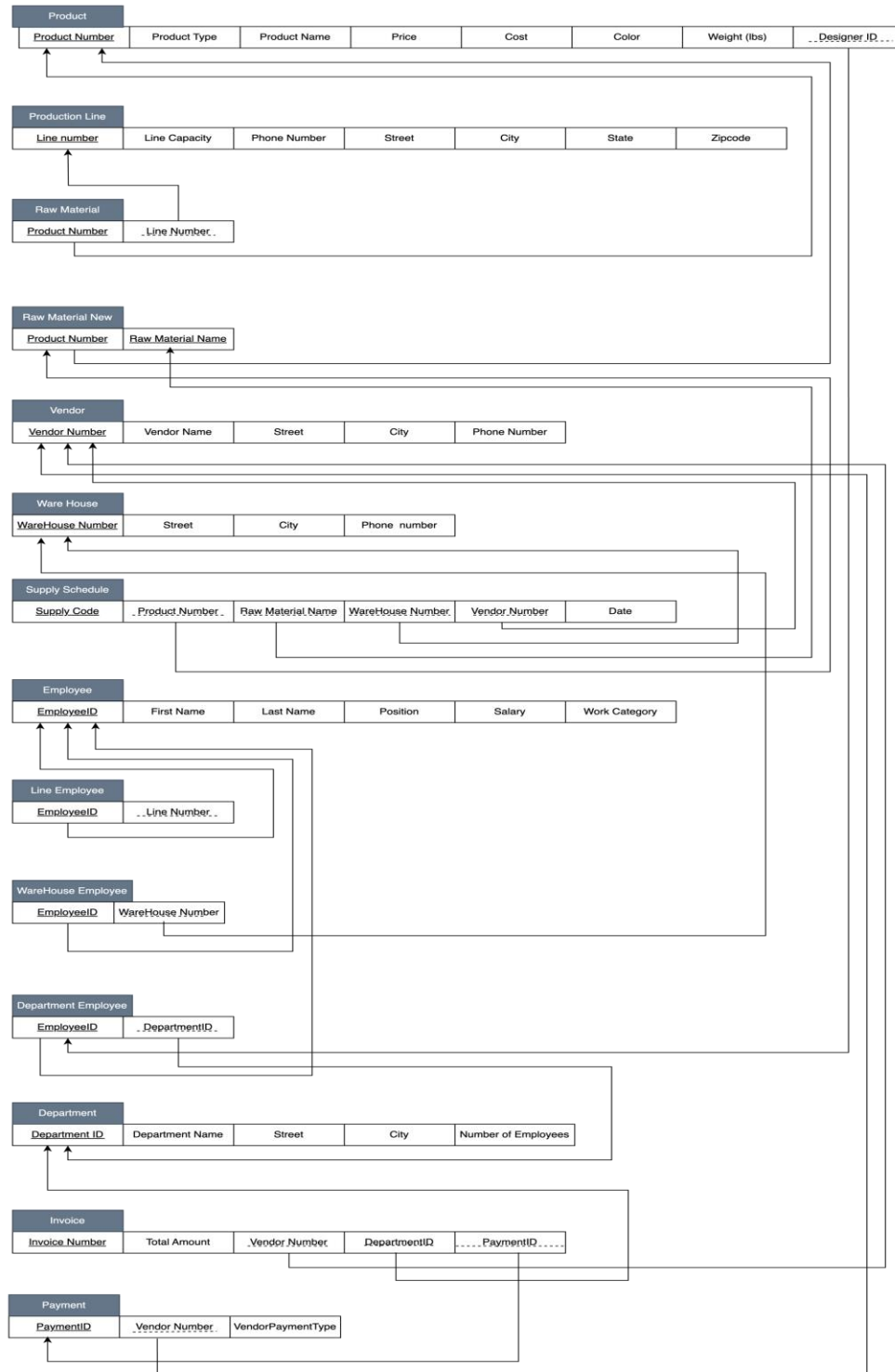
Tables susceptible to anomalies, such as Raw Material (partial dependency and modification anomaly) and Invoice (transitive dependency and deletion anomaly), were normalized by breaking them into two separate tables. This restructuring addressed the identified issues and promoted more robust data management.

Specialization of Employee Table:

The Employee table underwent specialization, resulting in the creation of subtypes like Line Employee, Warehouse Employee, and Department Employee. This facilitated the removal of null values associated with special attributes of these subtypes, contributing to a more refined and structured dataset.

Relational Schema after Normalization

By showcasing the below final relational schema, the project not only demonstrates adherence to normalization principles but also exhibits a more refined, organized, and efficient data structure that aligns with industry best practices.

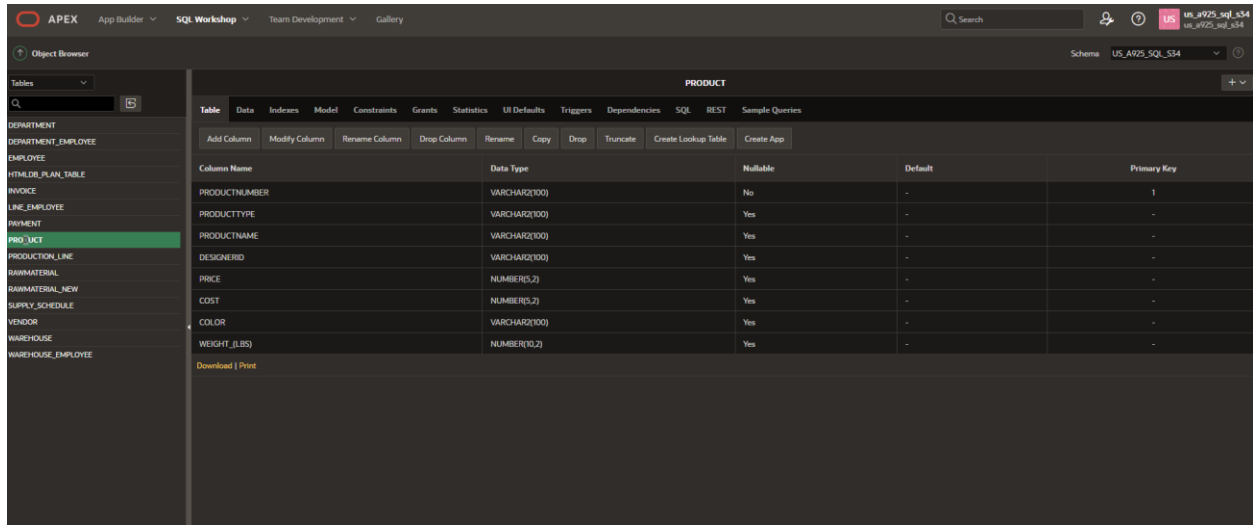


TASK 5

Crafting SQL Queries for Data Retrieval and Analysis.

Below is a snapshot of the 14 tables successfully created in Oracle APEX.

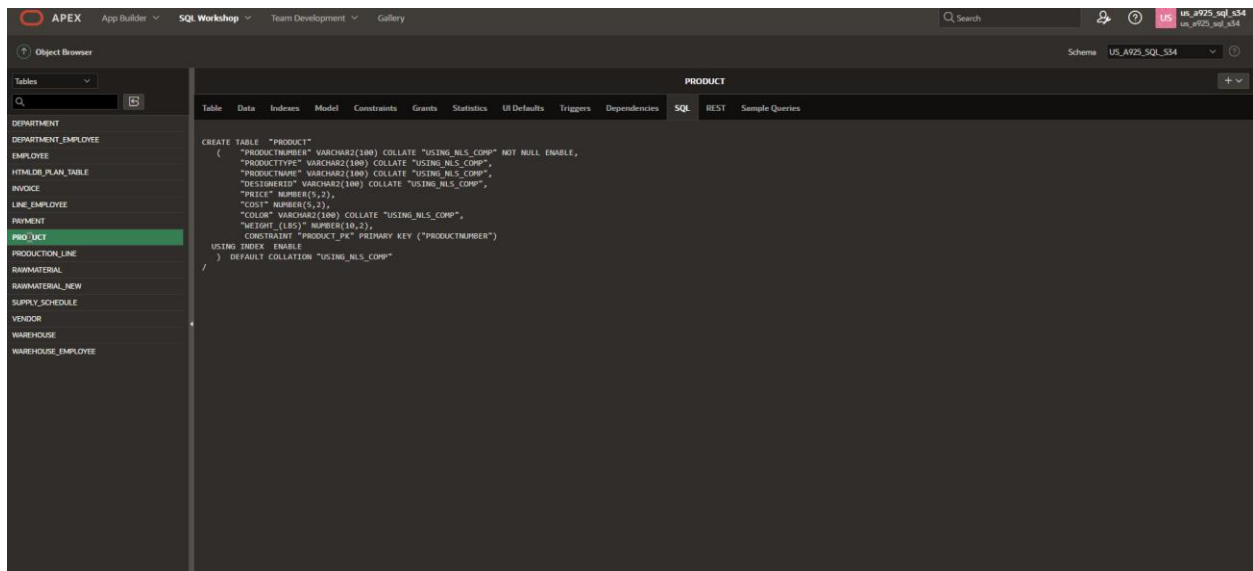
Product table:



The screenshot shows the Oracle APEX SQL Workshop interface. The left sidebar lists 14 tables: DEPARTMENT, DEPARTMENT_EMPLOYEE, EMPLOYEE, HTMLDB_PLAN_TABLE, INVOICE, LINE_EMPLOYEE, PAYMENT, **PRODUCT**, PRODUCTION_LINE, RAWMATERIAL, RAWMATERIAL_NEW, SUPPLY_SCHEDULE, VENDOR, WAREHOUSE, and WAREHOUSE_EMPLOYEE. The main panel displays the structure of the **PRODUCT** table with the following columns:

| Column Name | Data Type | Nullable | Default | Primary Key |
|---------------|--------------|----------|---------|-------------|
| PRODUCTNUMBER | VARCHAR(100) | No | - | 1 |
| PRODUCTTYPE | VARCHAR(100) | Yes | - | - |
| PRODUCTNAME | VARCHAR(100) | Yes | - | - |
| DESIGNERID | VARCHAR(100) | Yes | - | - |
| PRICE | NUMBER(5,2) | Yes | - | - |
| COST | NUMBER(5,2) | Yes | - | - |
| COLOR | VARCHAR(100) | Yes | - | - |
| WEIGHT_LBS | NUMBER(10,2) | Yes | - | - |

Below the table structure, there are links for [Download](#) and [Print](#).



The screenshot shows the Oracle APEX SQL Workshop interface with the **SQL** tab selected. The main panel displays the SQL code for creating the **PRODUCT** table:

```
CREATE TABLE "PRODUCT"
(
  "PRODUCTNUMBER" VARCHAR(100) COLLATE "USING_NLS_COMP" NOT NULL ENABLE,
  "PRODUCTTYPE" VARCHAR(100) COLLATE "USING_NLS_COMP",
  "PRODUCTNAME" VARCHAR(100) COLLATE "USING_NLS_COMP",
  "DESIGNERID" VARCHAR(100) COLLATE "USING_NLS_COMP",
  "PRICE" NUMBER(5,2),
  "COST" NUMBER(5,2),
  "COLOR" VARCHAR(100) COLLATE "USING_NLS_COMP",
  "WEIGHT_LBS" NUMBER(10,2),
  CONSTRAINT "PRODUCT_PK" PRIMARY KEY ("PRODUCTNUMBER")
) USING INDEX ENABLE
  ) DEFAULT COLLATION "USING_NLS_COMP"
```


Production Line:

APEX App Builder SQL Workshop Team Development Gallery

Object Browser

Tables

DEPARTMENT
DEPARTMENT_EMPLOYEE
EMPLOYEE
HTMLDL_PLAN_TABLE
INVOICE
LINE_EMPLOYEE
PAYMENT
PRODUCT
PRODUCTION_LINE
RAWMATERIAL
RAWMATERIAL_NEW
SUPPLY_SCHEDULE
VENDOR
WAREHOUSE
WAREHOUSE_EMPLOYEE

Schema US_A925_SQL_534

PRODUCTION_LINE

Table Data Indexes Model Constraints Grants Statistics UI Defaults Triggers Dependencies SQL REST Sample Queries

Add Column Modify Column Rename Column Drop Column Rename Copy Drop Truncate Create Lookup Table Create App

| Column Name | Data Type | Nullable | Default | Primary Key |
|-------------------------|--------------|----------|---------|-------------|
| LINENUMBER | VARCHAR(4) | No | - | 1 |
| LINECAPACITY_ITEMS/HOUR | NUMBER(5,0) | Yes | - | - |
| PHONENUMBER | NUMBER(10,0) | Yes | - | - |
| STREET | VARCHAR(250) | Yes | - | - |
| CITY | VARCHAR(50) | Yes | - | - |
| STATE | VARCHAR(2) | Yes | - | - |
| ZIPCODE | NUMBER(5,0) | Yes | - | - |

Download | Print

APEX App Builder SQL Workshop Team Development Gallery

Object Browser

Tables

DEPARTMENT
DEPARTMENT_EMPLOYEE
EMPLOYEE
HTMLDL_PLAN_TABLE
INVOICE
LINE_EMPLOYEE
PAYMENT
PRODUCT
PRODUCTION_LINE
RAWMATERIAL
RAWMATERIAL_NEW
SUPPLY_SCHEDULE
VENDOR
WAREHOUSE
WAREHOUSE_EMPLOYEE

Schema US_A925_SQL_534

PRODUCTION_LINE

Table Data Indexes Model Constraints Grants Statistics UI Defaults Triggers Dependencies SQL REST Sample Queries

```
CREATE TABLE "PRODUCTION_LINE"
(
  "LINENUMBER" VARCHAR(4) COLLATE "USING_NLS_COMP" NOT NULL ENABLE,
  "LINECAPACITY_ITEMS/HOUR" NUMBER(5,0),
  "PHONENUMBER" NUMBER(10,0),
  "STREET" VARCHAR(250) COLLATE "USING_NLS_COMP",
  "CITY" VARCHAR(50) COLLATE "USING_NLS_COMP",
  "STATE" VARCHAR(2) COLLATE "USING_NLS_COMP",
  "ZIPCODE" NUMBER(5,0),
  CONSTRAINT "PRODUCTION_LINE_PK" PRIMARY KEY ("LINENUMBER")
  USING INDEX ENABLE
)
DEFAULT COLLATION "USING_NLS_COMP"
```

Raw Material Table:

APEX App Builder SQL Workshop Team Development Gallery

Object Browser

Tables

DEPARTMENT
DEPARTMENT_EMPLOYEE
EMPLOYEE
HTHLDL_PLAN_TABLE
INVOICE
LINE_EMPLOYEE
PAYMENT
PRODUCT
PRODUCTION_LINE
RAWMATERIAL
RAWMATERIAL_NEW
SUPPLY_SCHEDULE
VENDOR
WAREHOUSE
WAREHOUSE_EMPLOYEE

Schema US_A925_SQL_554

RAWMATERIAL

Table Data Indexes Model Constraints Grants Statistics UI Defaults Triggers Dependencies SQL REST Sample Queries

Add Column Modify Column Rename Column Drop Column Rename Copy Drop Truncate Create Lookup Table Create App

| Column Name | Data Type | Nullable | Default | Primary Key |
|---------------|-------------|----------|---------|-------------|
| PRODUCTNUMBER | VARCHAR(20) | No | - | 1 |
| LINENUMBER | VARCHAR(4) | Yes | - | - |

Download | Print

APEX App Builder SQL Workshop Team Development Gallery

Object Browser

Tables

DEPARTMENT
DEPARTMENT_EMPLOYEE
EMPLOYEE
HTHLDL_PLAN_TABLE
INVOICE
LINE_EMPLOYEE
PAYMENT
PRODUCT
PRODUCTION_LINE
RAWMATERIAL
RAWMATERIAL_NEW
SUPPLY_SCHEDULE
VENDOR
WAREHOUSE
WAREHOUSE_EMPLOYEE

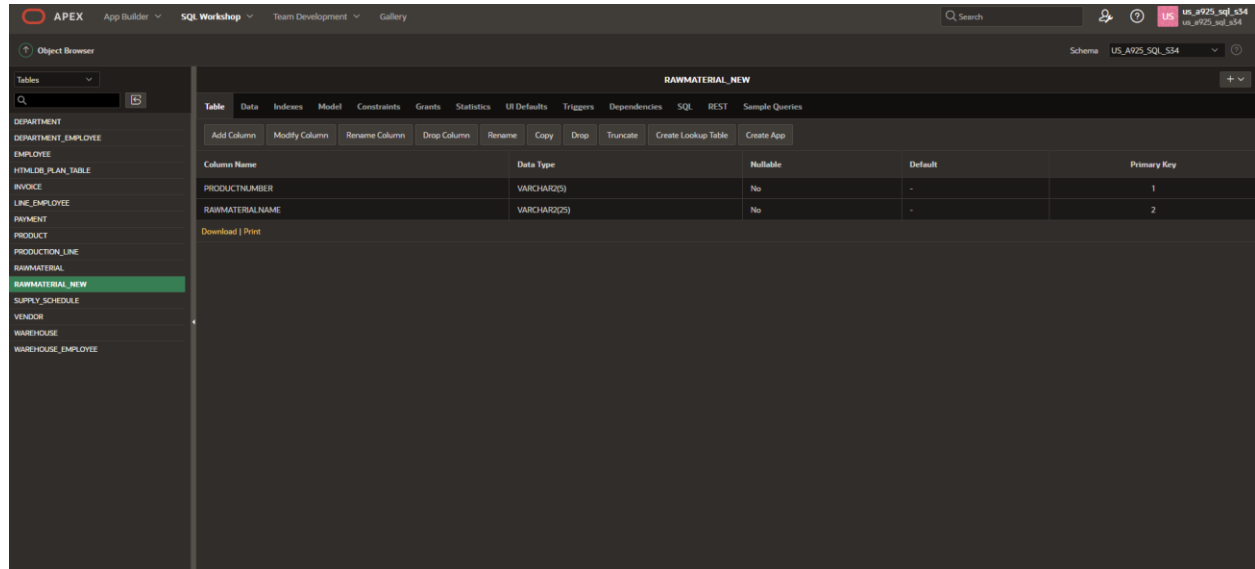
Schema US_A925_SQL_554

RAWMATERIAL

Table Data Indexes Model Constraints Grants Statistics UI Defaults Triggers Dependencies SQL REST Sample Queries

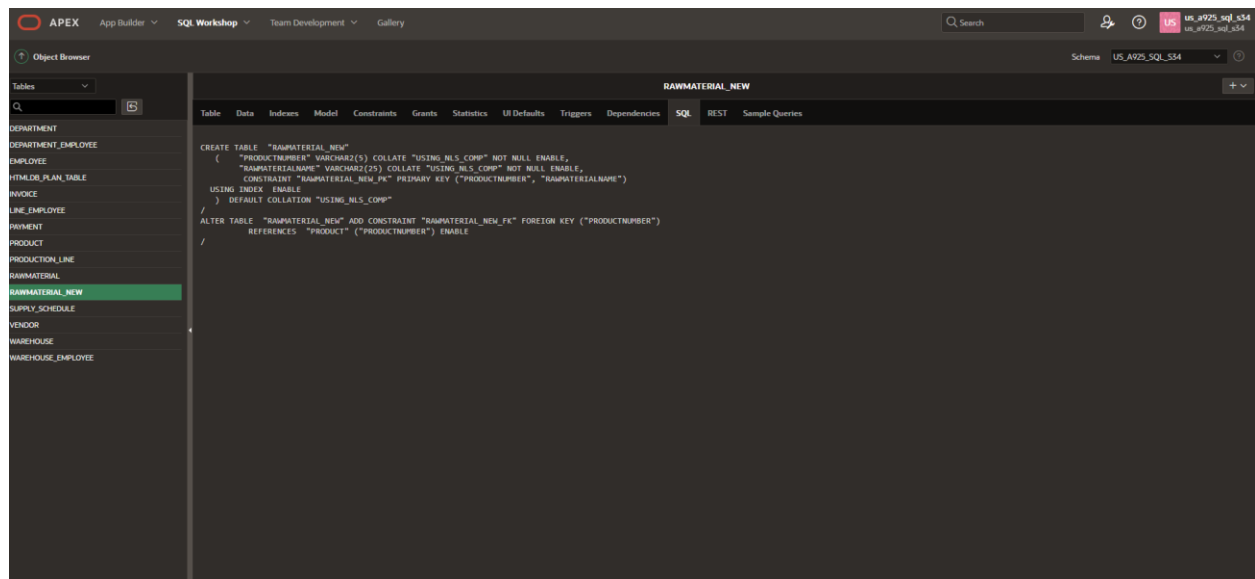
```
CREATE TABLE "RAWMATERIAL"
(
  "PRODUCTNUMBER" VARCHAR(20) COLLATE "USING_NLS_COMP" NOT NULL ENABLE,
  "LINENUMBER" VARCHAR(4) COLLATE "USING_NLS_COMP",
  CONSTRAINT "RAWMATERIAL_PK" PRIMARY KEY ("PRODUCTNUMBER")
  USING INDEX ENABLE
) DEFAULT COLLATION "USING_NLS_COMP"
/
ALTER TABLE "RAWMATERIAL" ADD CONSTRAINT "RAWMATERIAL_FK" FOREIGN KEY ("LINENUMBER")
REFERENCES "PRODUCTION_LINE" ("LINENUMBER") ENABLE
/
```

Raw Material New Table:



The screenshot shows the Oracle APEX SQL Workshop interface. The left sidebar displays the Object Browser with a tree view of database objects. The main panel shows the structure of the RAWMATERIAL_NEW table, including columns, data types, nullability, defaults, and primary keys.

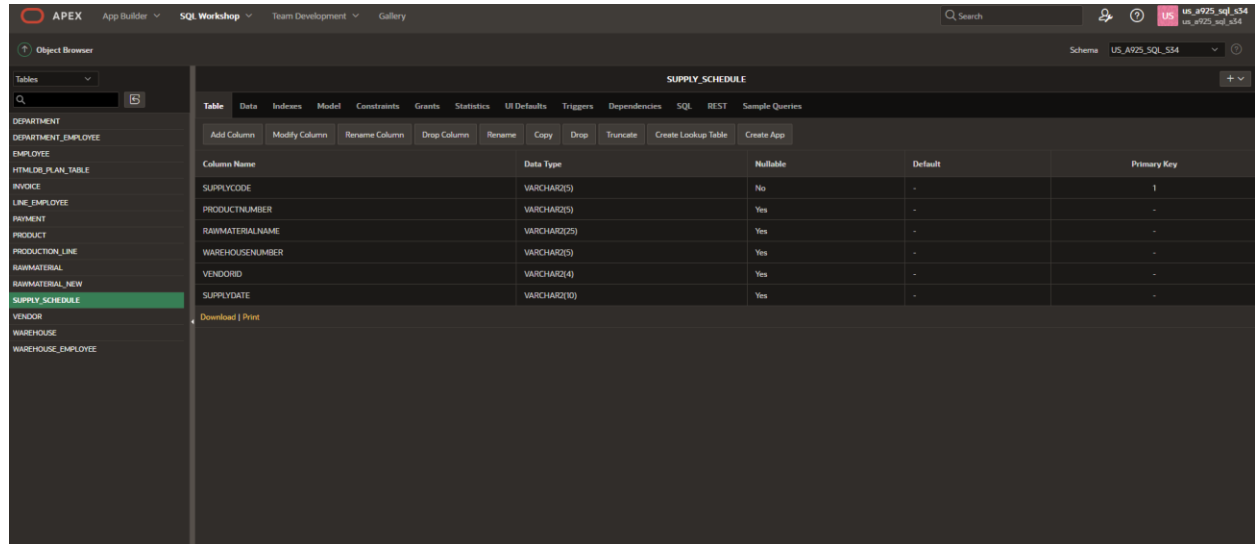
| Column Name | Data Type | Nullable | Default | Primary Key |
|-----------------|-------------|----------|---------|-------------|
| PRODUCTNUMBER | VARCHAR(25) | No | - | 1 |
| RAWMATERIALNAME | VARCHAR(25) | No | - | 2 |



The screenshot shows the Oracle APEX SQL Workshop interface with the SQL tab selected. The main panel displays the SQL script for creating the RAWMATERIAL_NEW table, including the CREATE TABLE statement, column definitions, constraints, and the ALTER TABLE statement to add a foreign key.

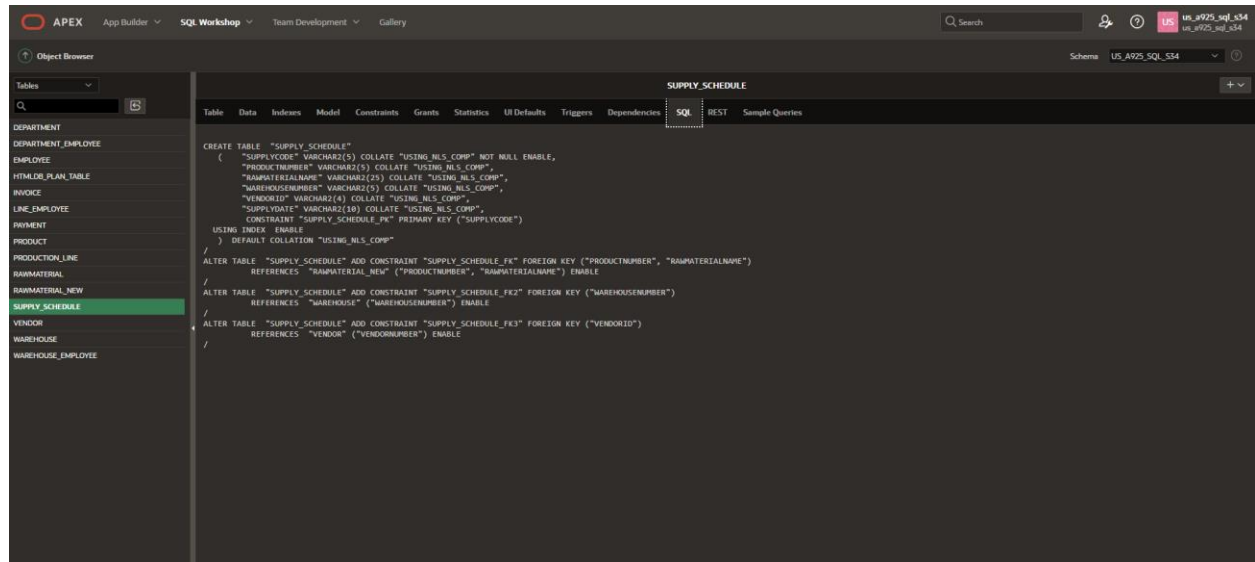
```
CREATE TABLE "RAWMATERIAL_NEW"
(
  "PRODUCTNUMBER" VARCHAR(25) COLLATE "USING_NLS_COMP" NOT NULL ENABLE,
  "RAWMATERIALNAME" VARCHAR(25) COLLATE "USING_NLS_COMP" NOT NULL ENABLE,
  CONSTRAINT "RAWMATERIAL_NEW_PK" PRIMARY KEY ("PRODUCTNUMBER", "RAWMATERIALNAME")
  USING INDEX ENABLE
) DEFAULT COLLATION "USING_NLS_COMP"
/
ALTER TABLE "RAWMATERIAL_NEW" ADD CONSTRAINT "RAWMATERIAL_NEW_FK" FOREIGN KEY ("PRODUCTNUMBER")
REFERENCES "PRODUCT" ("PRODUCTNUMBER") ENABLE
/
```

Supply Schedule Table:



The screenshot shows the Oracle APEX SQL Workshop interface. On the left is the Object Browser with a tree view of database objects. The main pane displays the structure of the SUPPLY_SCHEDULE table. The table has six columns: SUPPLYCODE, PRODUCTNUMBER, RAWMATERIALNAME, WAREHOUSENUMBER, VENDORD, and SUPPLYDATE. SUPPLYCODE is the primary key. The interface includes tabs for Table, Data, Indexes, Model, Constraints, Grants, Statistics, UI Defaults, Triggers, Dependencies, SQL, REST, and Sample Queries. Below the column list are buttons for 'Download' and 'Print'.

| Column Name | Data Type | Nullable | Default | Primary Key |
|-----------------|-------------|----------|---------|-------------|
| SUPPLYCODE | VARCHAR(25) | No | - | 1 |
| PRODUCTNUMBER | VARCHAR(25) | Yes | - | - |
| RAWMATERIALNAME | VARCHAR(25) | Yes | - | - |
| WAREHOUSENUMBER | VARCHAR(25) | Yes | - | - |
| VENDORD | VARCHAR(4) | Yes | - | - |
| SUPPLYDATE | VARCHAR(10) | Yes | - | - |



The screenshot shows the same Oracle APEX SQL Workshop interface, but the SQL tab is selected. It displays the SQL script to create the SUPPLY_SCHEDULE table, including column definitions, a primary key constraint, and foreign key constraints.

```
CREATE TABLE "SUPPLY_SCHEDULE"
(
  "SUPPLYCODE" VARCHAR(25) COLLATE "USING_NLS_COMP" NOT NULL ENABLE,
  "PRODUCTNUMBER" VARCHAR(25) COLLATE "USING_NLS_COMP",
  "RAWMATERIALNAME" VARCHAR(25) COLLATE "USING_NLS_COMP",
  "WAREHOUSENUMBER" VARCHAR(25) COLLATE "USING_NLS_COMP",
  "VENDORD" VARCHAR(4) COLLATE "USING_NLS_COMP",
  "SUPPLYDATE" VARCHAR(10) COLLATE "USING_NLS_COMP",
  CONSTRAINT "SUPPLY_SCHEDULE_PK" PRIMARY KEY ("SUPPLYCODE")
) USING INDEX ENABLE
) DEFAULT COLLATION "USING_NLS_COMP"
/
ALTER TABLE "SUPPLY_SCHEDULE" ADD CONSTRAINT "SUPPLY_SCHEDULE_FK1" FOREIGN KEY ("PRODUCTNUMBER", "RAWMATERIALNAME")
REFERENCES "RAWMATERIAL_NAME" ("PRODUCTNUMBER", "RAWMATERIALNAME") ENABLE
/
ALTER TABLE "SUPPLY_SCHEDULE" ADD CONSTRAINT "SUPPLY_SCHEDULE_FK2" FOREIGN KEY ("WAREHOUSENUMBER")
REFERENCES "WAREHOUSE" ("WAREHOUSENUMBER") ENABLE
/
ALTER TABLE "SUPPLY_SCHEDULE" ADD CONSTRAINT "SUPPLY_SCHEDULE_FK3" FOREIGN KEY ("VENDORD")
REFERENCES "VENDOR" ("VENDORD") ENABLE
/
```

Vendor Table:

APEX App Builder SQL Workshop Team Development Gallery

Object Browser

Tables

DEPARTMENT
DEPARTMENT_EMPLOYEE
EMPLOYEE
HTMLDB_PLAN_TABLE
INVOICE
LINE_EMPLOYEE
PAYMENT
PRODUCT
PRODUCTION_LINE
RAWMATERIAL
RAWMATERIAL_NEW
SUPPLY_SCHEDULE
VENDOR
WAREHOUSE
WAREHOUSE_EMPLOYEE

VENDOR

Table Data Indices Model Constraints Grants Statistics UI Defaults Triggers Dependencies SQL REST Sample Queries

Add Column Modify Column Rename Column Drop Column Rename Copy Drop Truncate Create Lookup Table Create App

| Column Name | Data Type | Nullable | Default | Primary Key |
|--------------|--------------|----------|---------|-------------|
| VENDORNUMBER | VARCHAR(4) | No | - | 1 |
| VENDORNAME | VARCHAR(50) | Yes | - | - |
| STREET | VARCHAR(25) | Yes | - | - |
| CITY | VARCHAR(25) | Yes | - | - |
| PHONENUMBER | NUMBER(10,0) | Yes | - | - |

Download | Print

APEX App Builder SQL Workshop Team Development Gallery

Object Browser

Tables

DEPARTMENT
DEPARTMENT_EMPLOYEE
EMPLOYEE
HTMLDB_PLAN_TABLE
INVOICE
LINE_EMPLOYEE
PAYMENT
PRODUCT
PRODUCTION_LINE
RAWMATERIAL
RAWMATERIAL_NEW
SUPPLY_SCHEDULE
VENDOR
WAREHOUSE
WAREHOUSE_EMPLOYEE

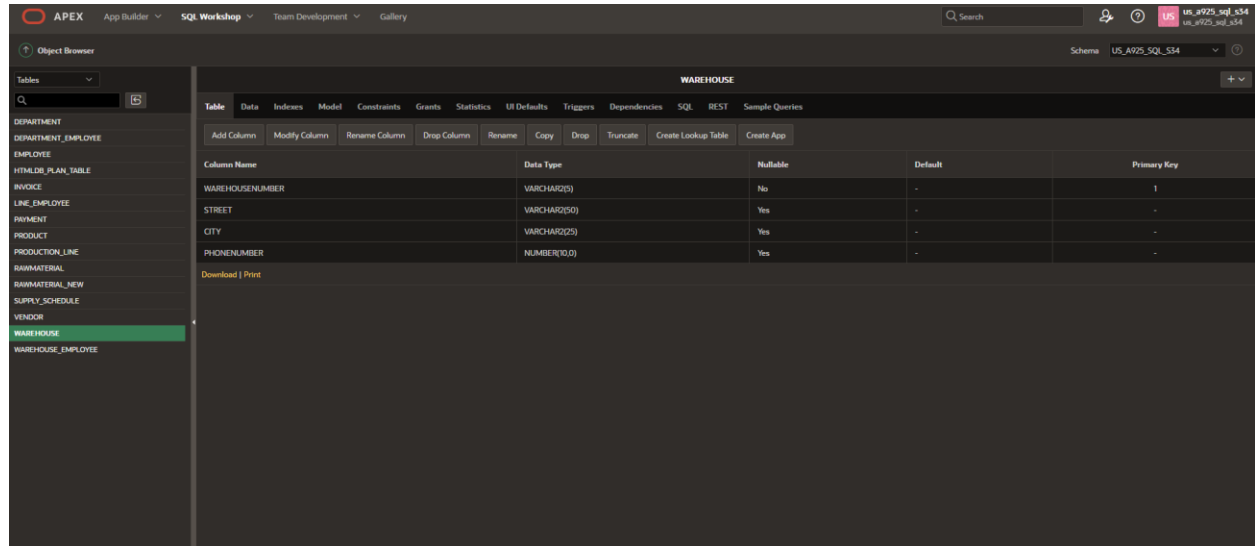
VENDOR

Table Data Indices Model Constraints Grants Statistics UI Defaults Triggers Dependencies SQL REST Sample Queries

```

CREATE TABLE "VENDOR"
(
  "VENDORNUMBER" VARCHAR(4) COLLATE "USING_NLS_COMP" NOT NULL ENABLE,
  "VENDORNAME" VARCHAR(50) COLLATE "USING_NLS_COMP",
  "STREET" VARCHAR(25) COLLATE "USING_NLS_COMP",
  "CITY" VARCHAR(25) COLLATE "USING_NLS_COMP",
  "PHONENUMBER" NUMBER(10,0),
  CONSTRAINT "VENDOR_PK" PRIMARY KEY ("VENDORNUMBER")
  USING INDEX ENABLE
) DEFAULT COLLATION "USING_NLS_COMP"
  
```

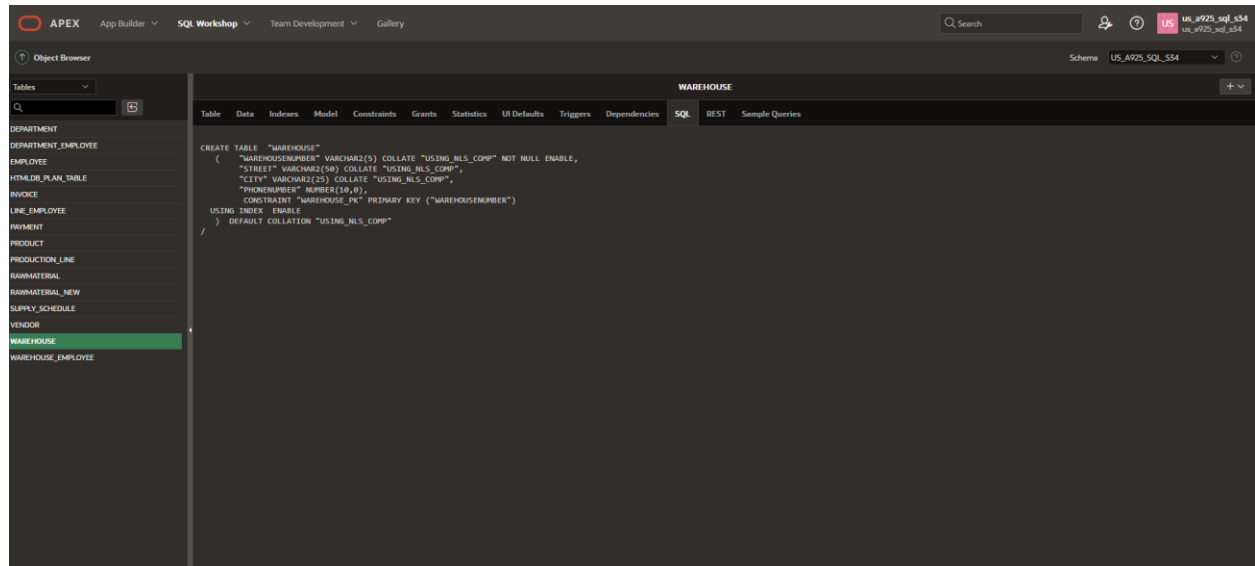
Warehouse Table:



The screenshot shows the Oracle APEX SQL Workshop interface. The left sidebar displays a list of database objects, with 'WAREHOUSE' selected. The main panel shows the 'Table' tab for the 'WAREHOUSE' table. The table structure is as follows:

| Column Name | Data Type | Nullable | Default | Primary Key |
|-----------------|--------------|----------|---------|-------------|
| WAREHOUSENUMBER | VARCHAR(25) | No | - | 1 |
| STREET | VARCHAR(250) | Yes | - | - |
| CITY | VARCHAR(25) | Yes | - | - |
| PHONENUMBER | NUMBER(10,0) | Yes | - | - |

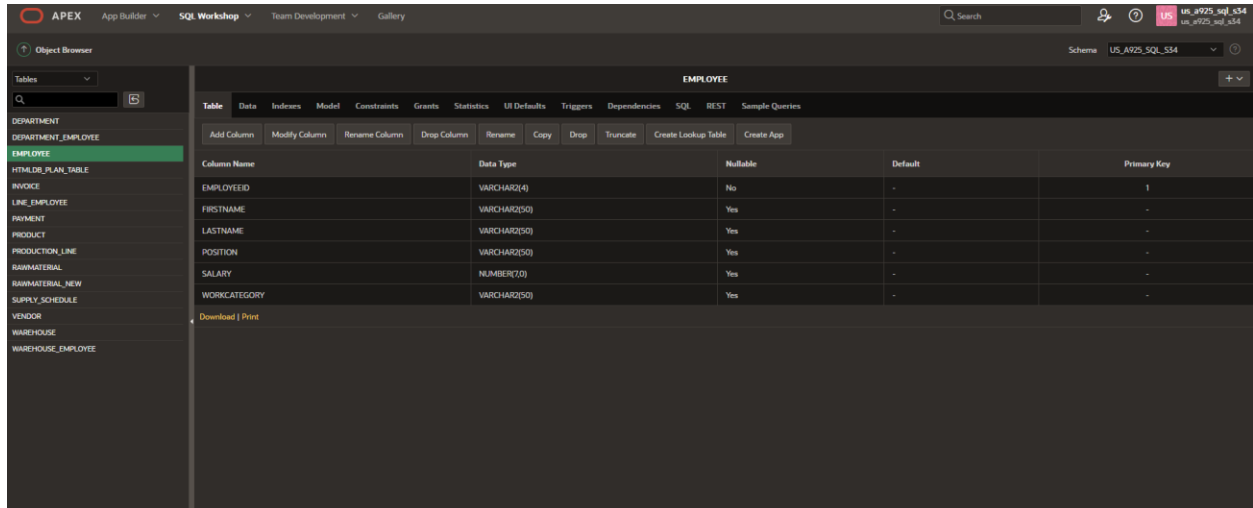
Below the table structure, there are links for 'Download' and 'Print'.



The screenshot shows the Oracle APEX SQL Workshop interface with the 'SQL' tab selected for the 'WAREHOUSE' table. The SQL script for creating the table is displayed:

```
CREATE TABLE "WAREHOUSE"
(
  "WAREHOUSENUMBER" VARCHAR(25) COLLATE "USING_NLS_COMP" NOT NULL ENABLE,
  "STREET" VARCHAR(250) COLLATE "USING_NLS_COMP",
  "CITY" VARCHAR(25) COLLATE "USING_NLS_COMP",
  "PHONENUMBER" NUMBER(10,0),
  CONSTRAINT "WAREHOUSE_PK" PRIMARY KEY ("WAREHOUSENUMBER")
) DEFAULT COLLATION "USING_NLS_COMP"
/
```

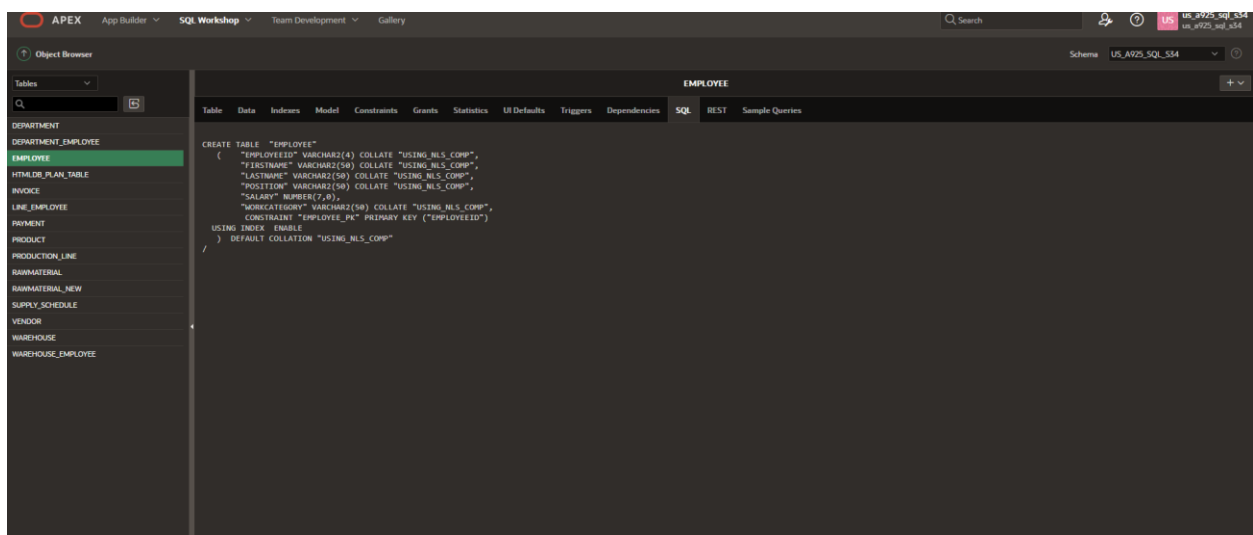
Employee Table:



The screenshot shows the Oracle APEX SQL Workshop interface. The left sidebar displays the Object Browser with the following items: DEPARTMENT, DEPARTMENT_EMPLOYEE, EMPLOYEE (highlighted), HTMLDB_PLAN_TABLE, INVOICE, LINE_EMPLOYEE, PAYMENT, PRODUCT, PRODUCTION_LINE, RAWMATERIAL, RAWMATERIAL_NEW, SUPPLY_SCHEDULE, VENDOR, WAREHOUSE, and WAREHOUSE_EMPLOYEE. The main panel shows the structure of the EMPLOYEE table with the following columns:

| Column Name | Data Type | Nullable | Default | Primary Key |
|--------------|-------------|----------|---------|-------------|
| EMPLOYEEID | VARCHAR(4) | No | - | 1 |
| FIRSTNAME | VARCHAR(50) | Yes | - | - |
| LASTNAME | VARCHAR(50) | Yes | - | - |
| POSITION | VARCHAR(50) | Yes | - | - |
| SALARY | NUMBER(7,0) | Yes | - | - |
| WORKCATEGORY | VARCHAR(50) | Yes | - | - |

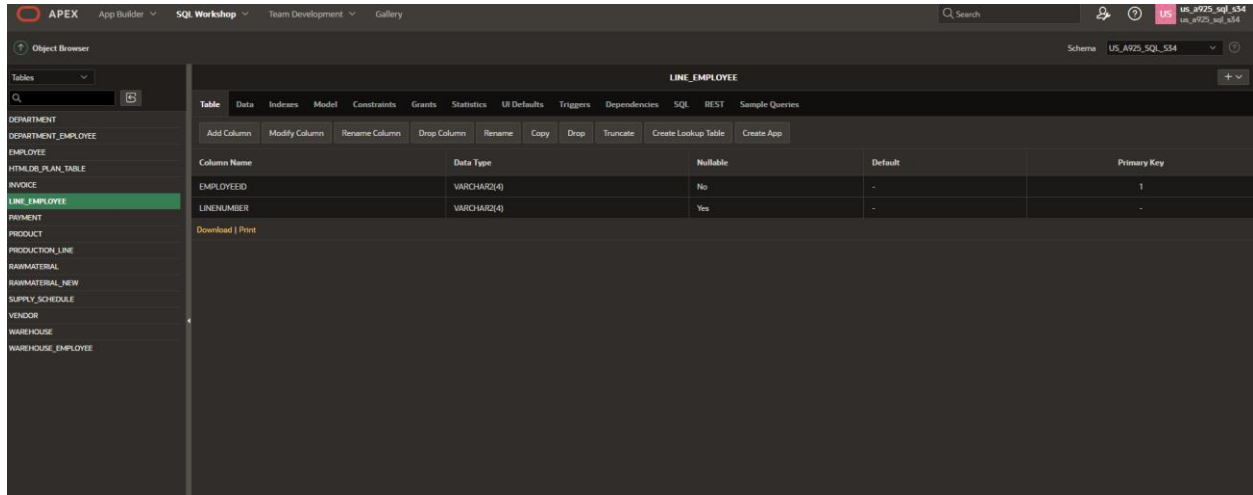
Below the table structure, there are links for [Download](#) and [Print](#).



The screenshot shows the Oracle APEX SQL Workshop interface with the SQL tab selected. The left sidebar is the same as the previous screenshot. The main panel displays the SQL DDL for the EMPLOYEE table:

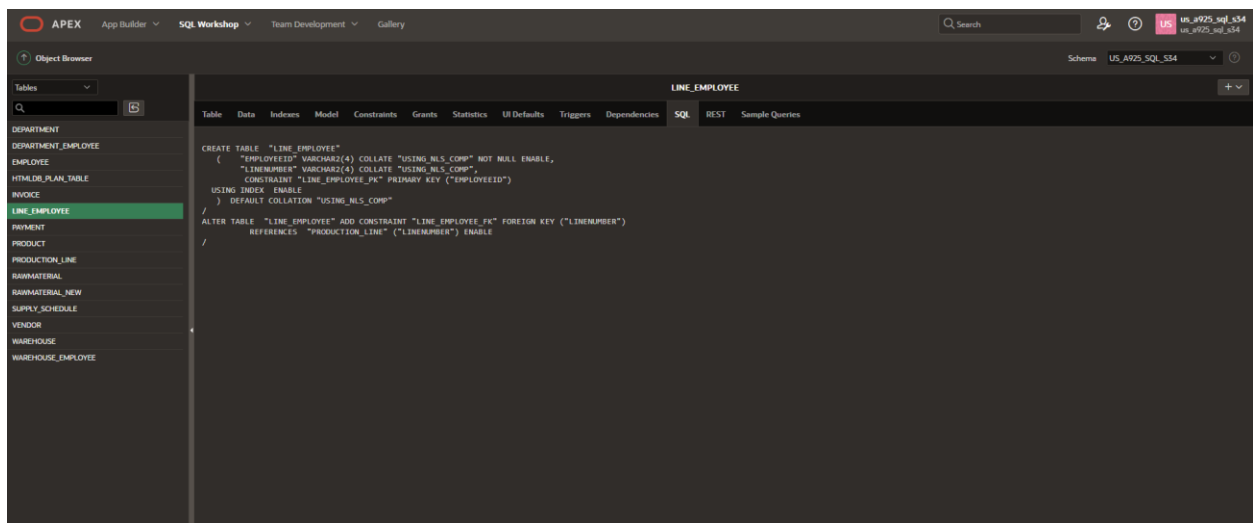
```
CREATE TABLE "EMPLOYEE"
(
  "EMPLOYEEID" VARCHAR(4) COLLATE "USING_NLS_COMP",
  "FIRSTNAME" VARCHAR(50) COLLATE "USING_NLS_COMP",
  "LASTNAME" VARCHAR(50) COLLATE "USING_NLS_COMP",
  "POSITION" VARCHAR(50) COLLATE "USING_NLS_COMP",
  "SALARY" NUMBER(7,0),
  "WORKCATEGORY" VARCHAR(50) COLLATE "USING_NLS_COMP",
  CONSTRAINT "EMPLOYEE_PK" PRIMARY KEY ("EMPLOYEEID")
) USING INDEX ENABLE
) DEFAULT COLLATION "USING_NLS_COMP"
```

Line Employee Table:



The screenshot shows the APEX SQL Workshop interface. On the left, the Object Browser lists various database objects, with **LINE_EMPLOYEE** selected. The main panel displays the table structure for **LINE_EMPLOYEE**. The table has two columns: **EMPLOYEEID** and **LINENUMBER**. **EMPLOYEEID** is a VARCHAR2(4) column, not nullable, with a default value of '-' and is the primary key. **LINENUMBER** is a VARCHAR2(4) column, nullable, with a default value of '-'.

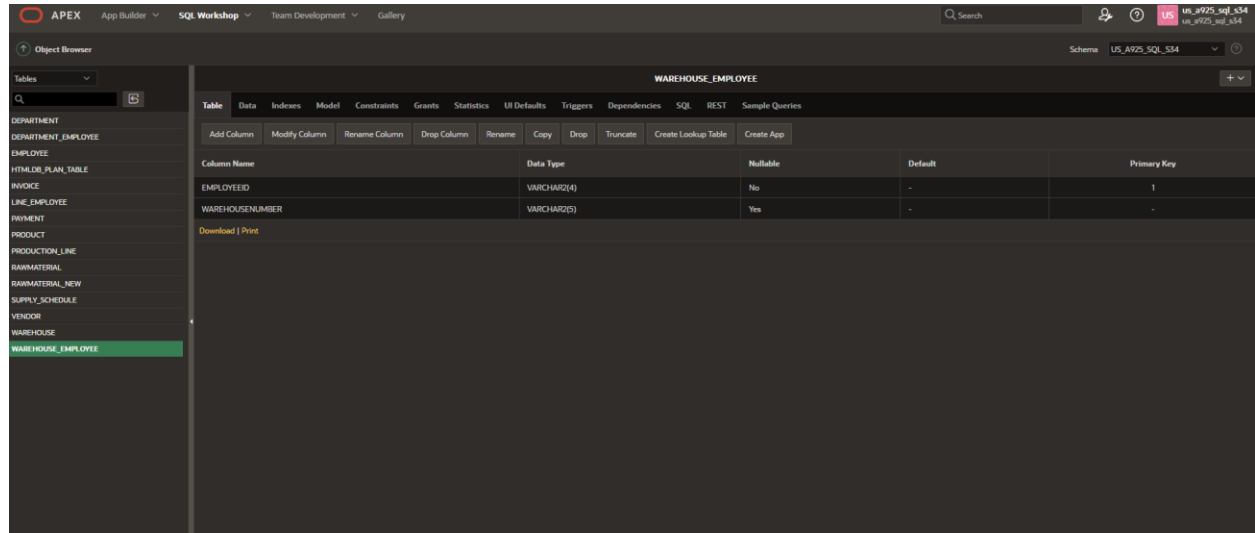
| Column Name | Data Type | Nullable | Default | Primary Key |
|-------------|-------------|----------|---------|-------------|
| EMPLOYEEID | VARCHAR2(4) | No | - | 1 |
| LINENUMBER | VARCHAR2(4) | Yes | - | - |



The screenshot shows the APEX SQL Workshop interface with the SQL tab selected. The SQL script for creating the **LINE_EMPLOYEE** table is displayed. The script includes a **CREATE TABLE** statement for **LINE_EMPLOYEE** with columns **EMPLOYEEID** and **LINENUMBER**. **EMPLOYEEID** is a VARCHAR2(4) column, not nullable, with a default value of '-'. **LINENUMBER** is a VARCHAR2(4) column, nullable, with a default value of '-'. A primary key constraint **LINE_EMPLOYEE_PK** is defined on **EMPLOYEEID**. A foreign key constraint **LINE_EMPLOYEE_FK** is added to **LINENUMBER**, referencing the **PRODUCTION_LINE** table.

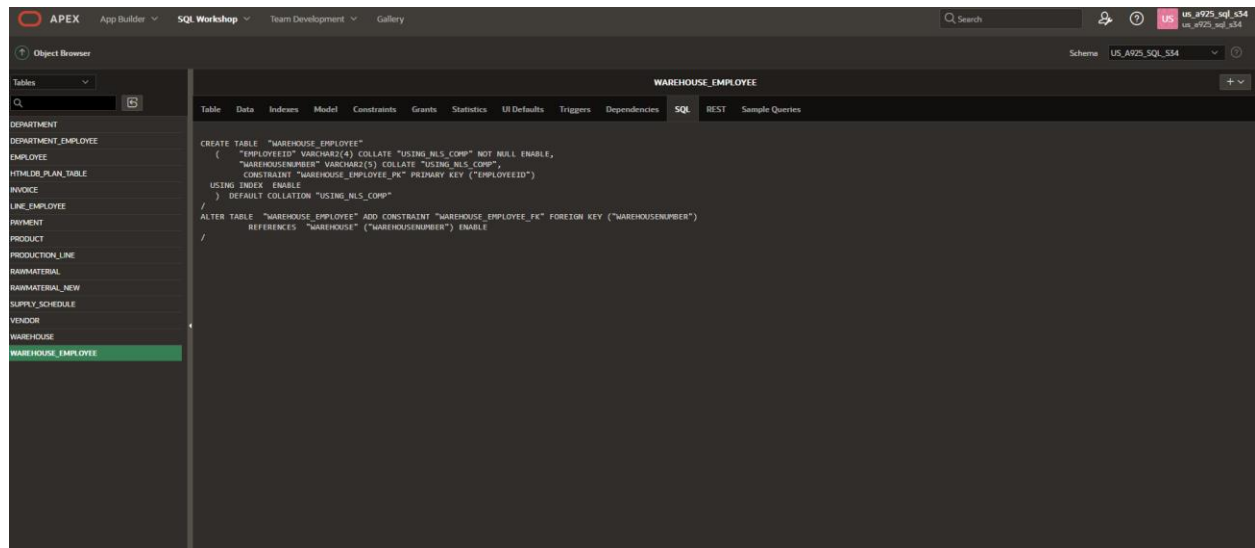
```
CREATE TABLE "LINE_EMPLOYEE"
(
  "EMPLOYEEID" VARCHAR2(4) COLLATE "USING_NLS_Comp" NOT NULL ENABLE,
  "LINENUMBER" VARCHAR2(4) COLLATE "USING_NLS_Comp",
  CONSTRAINT "LINE_EMPLOYEE_PK" PRIMARY KEY ("EMPLOYEEID")
  USING INDEX ENABLE
) DEFAULT COLLATION "USING_NLS_Comp"
/
ALTER TABLE "LINE_EMPLOYEE" ADD CONSTRAINT "LINE_EMPLOYEE_FK" FOREIGN KEY ("LINENUMBER")
REFERENCES "PRODUCTION_LINE" ("LINENUMBER") ENABLE
/
```


Ware House Employee Table:



The screenshot shows the Oracle APEX SQL Workshop interface. The left sidebar lists database objects, with 'WAREHOUSE_EMPLOYEE' selected. The main panel displays the table's structure with columns: EMPLOYEEID, WAREHOUSENUMBER, and their respective data types and constraints.

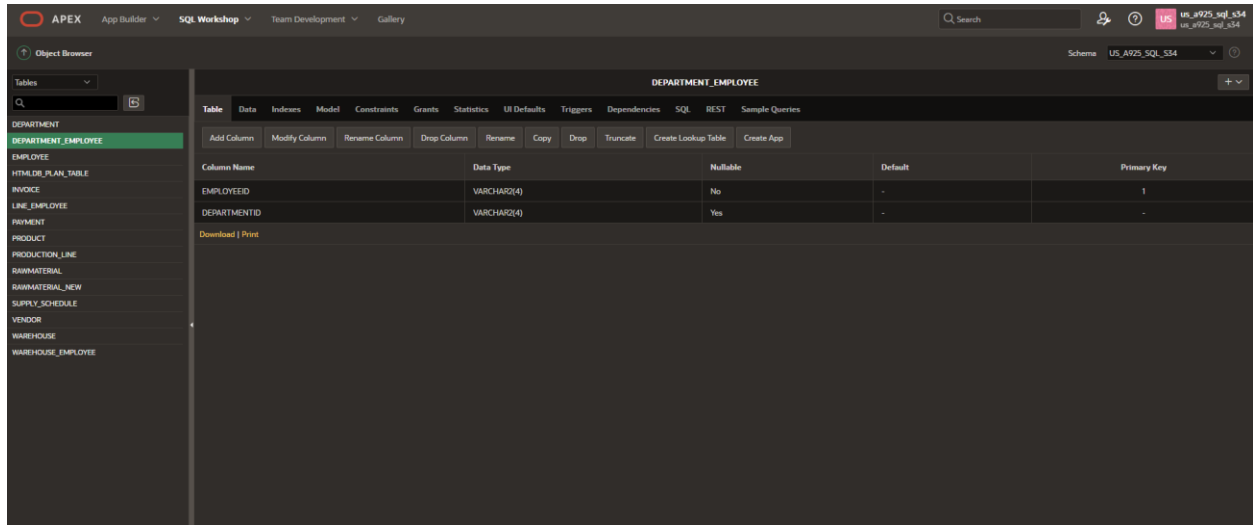
| Column Name | Data Type | Nullable | Default | Primary Key |
|-----------------|-------------|----------|---------|-------------|
| EMPLOYEEID | VARCHAR2(4) | No | - | 1 |
| WAREHOUSENUMBER | VARCHAR2(5) | Yes | - | - |



The screenshot shows the same Oracle APEX SQL Workshop interface, but with the 'SQL' tab selected. It displays the SQL script used to create the WAREHOUSE_EMPLOYEE table, including column definitions, constraints, and an index.

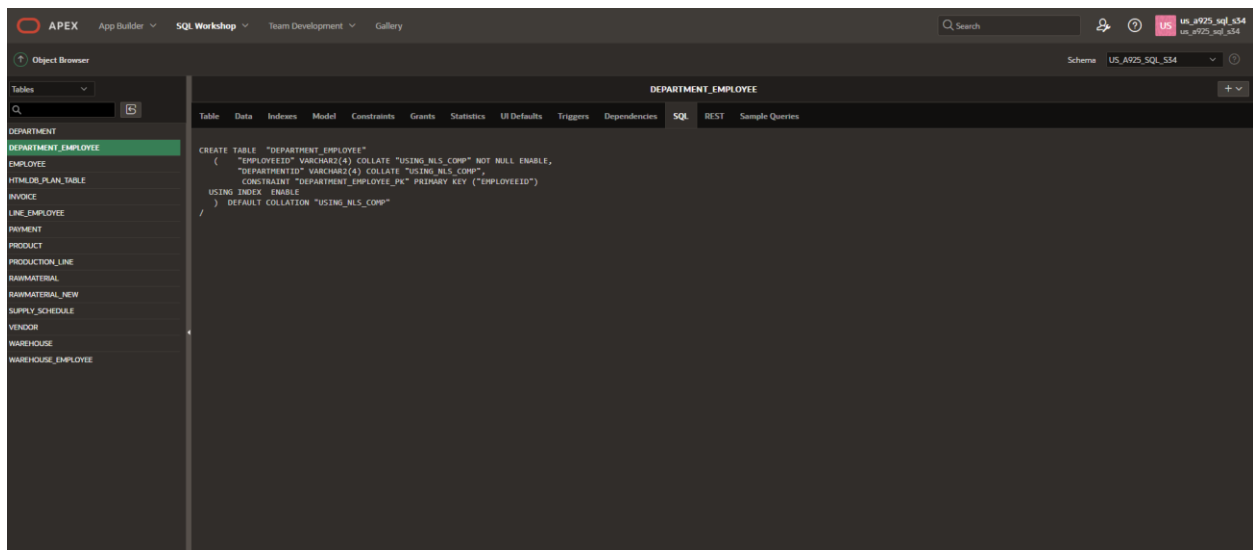
```
CREATE TABLE "WAREHOUSE_EMPLOYEE"
(
  "EMPLOYEEID" VARCHAR2(4) COLLATE "USING_NLS_Comp" NOT NULL ENABLE,
  "WAREHOUSENUMBER" VARCHAR2(5) COLLATE "USING_NLS_Comp",
  CONSTRAINT "WAREHOUSE_EMPLOYEE_PK" PRIMARY KEY ("EMPLOYEEID")
  USING INDEX ENABLE
) DEFAULT COLLATION "USING_NLS_Comp"
/
ALTER TABLE "WAREHOUSE_EMPLOYEE" ADD CONSTRAINT "WAREHOUSE_EMPLOYEE_FK" FOREIGN KEY ("WAREHOUSENUMBER")
REFERENCES "WAREHOUSE" ("WAREHOUSENUMBER") ENABLE
/
```

Department Employee Table:



The screenshot shows the Oracle APEX SQL Workshop interface. The left sidebar lists various database objects, with 'DEPARTMENT_EMPLOYEE' selected under the 'DEPARTMENT' schema. The main panel displays the table's structure with columns: EMPLOYEEID, DEPARTMENTID, and their respective data types and constraints.

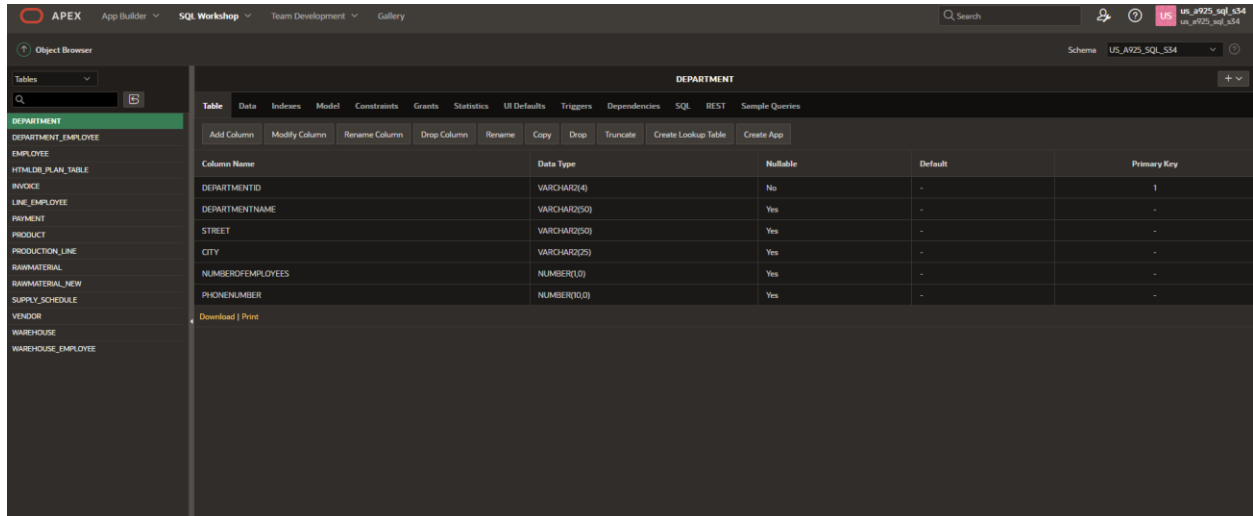
| Table | Data | Indexes | Model | Constraints | Grants | Statistics | UI Defaults | Triggers | Dependencies | SQL | REST | Sample Queries |
|---------------------|-------------|----------|---------|-------------|--------|------------|-------------|----------|--------------|-----|------|----------------|
| DEPARTMENT_EMPLOYEE | | | | | | | | | | | | |
| Column Name | Data Type | Nullable | Default | Primary Key | | | | | | | | |
| EMPLOYEEID | VARCHAR2(4) | No | - | 1 | | | | | | | | |
| DEPARTMENTID | VARCHAR2(4) | Yes | - | - | | | | | | | | |



The screenshot shows the Oracle APEX SQL Workshop interface with the 'SQL' tab selected. It displays the SQL code used to create the DEPARTMENT_EMPLOYEE table, including column definitions, constraints, and index creation.

```
CREATE TABLE "DEPARTMENT_EMPLOYEE"
(
  "EMPLOYEEID" VARCHAR2(4) COLLATE "USING_NLS_COMP" NOT NULL ENABLE,
  "DEPARTMENTID" VARCHAR2(4) COLLATE "USING_NLS_COMP",
  CONSTRAINT "DEPARTMENT_EMPLOYEE_PK" PRIMARY KEY ("EMPLOYEEID")
  USING INDEX ENABLE
)
DEFAULT COLLATION "USING_NLS_COMP"
```

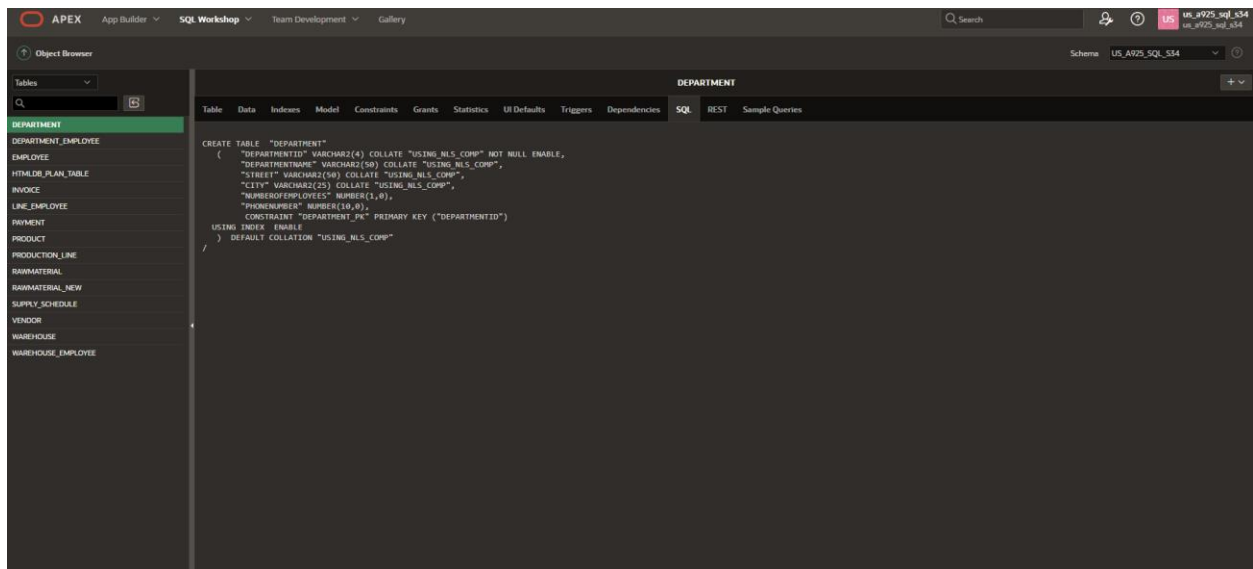
Department Table:



The screenshot shows the Oracle APEX SQL Workshop interface. The left sidebar displays a list of database objects, with 'DEPARTMENT' selected. The main panel shows the 'DEPARTMENT' table structure with the following columns:

| Column Name | Data Type | Nullable | Default | Primary Key |
|-------------------|--------------|----------|---------|-------------|
| DEPARTMENTID | VARCHAR2(4) | No | - | 1 |
| DEPARTMENTNAME | VARCHAR2(30) | Yes | - | - |
| STREET | VARCHAR2(50) | Yes | - | - |
| CITY | VARCHAR2(25) | Yes | - | - |
| NUMBEROFEMPLOYEES | NUMBER(10) | Yes | - | - |
| PHONENUMBER | NUMBER(10,0) | Yes | - | - |

Below the table structure, there are links for 'Download' and 'Print'.



The screenshot shows the Oracle APEX SQL Workshop interface with the 'SQL' tab selected. The main panel displays the SQL statement to create the 'DEPARTMENT' table:

```
CREATE TABLE "DEPARTMENT"
(
  "DEPARTMENTID" VARCHAR2(4) COLLATE "USING_NLS_COMP" NOT NULL ENABLE,
  "DEPARTMENTNAME" VARCHAR2(30) COLLATE "USING_NLS_COMP",
  "STREET" VARCHAR2(50) COLLATE "USING_NLS_COMP",
  "CITY" VARCHAR2(25) COLLATE "USING_NLS_COMP",
  "NUMBEROFEMPLOYEES" NUMBER(1,0),
  "PHONENUMBER" NUMBER(10,0),
  CONSTRAINT "DEPARTMENT_PK" PRIMARY KEY ("DEPARTMENTID")
) USING INDEX ENABLE
  DEFAULT COLLATION "USING_NLS_COMP"
```

Invoice Table:

APEX App Builder SQL Workshop Team Development Gallery

Object Browser

Schema US_A925_SQL_S54

Tables

DEPARTMENT
DEPARTMENT_EMPLOYEE
EMPLOYEE
HTMLDB_PLAN_TABLE
INVOICE
LINE_EMPLOYEE
PAYMENT
PRODUCT
PRODUCTION_LINE
RAWMATERIAL
RAWMATERIAL_NEW
SUPPLY_SCHEDULE
VENDOR
WAREHOUSE
WAREHOUSE_EMPLOYEE

INVOICE

Table Data Indexes Model Constraints Grants Statistics UI Defaults Triggers Dependencies SQL REST Sample Queries

Add Column Modify Column Rename Column Drop Column Rename Copy Drop Truncate Create Lookup Table Create App

| Column Name | Data Type | Nullable | Default | Primary Key |
|---------------|--------------|----------|---------|-------------|
| INVOICENUMBER | VARCHAR2(15) | No | - | 1 |
| TOTAL_AMOUNT | NUMBER(4,2) | Yes | - | - |
| VENDORNUMBER | VARCHAR2(4) | Yes | - | - |
| PAYMENTID | VARCHAR2(4) | Yes | - | - |
| DEPARTMENTID | VARCHAR2(4) | Yes | - | - |

Download | Print

APEX App Builder SQL Workshop Team Development Gallery

Object Browser

Schema US_A925_SQL_S54

Tables

DEPARTMENT
DEPARTMENT_EMPLOYEE
EMPLOYEE
HTMLDB_PLAN_TABLE
INVOICE
LINE_EMPLOYEE
PAYMENT
PRODUCT
PRODUCTION_LINE
RAWMATERIAL
RAWMATERIAL_NEW
SUPPLY_SCHEDULE
VENDOR
WAREHOUSE
WAREHOUSE_EMPLOYEE

INVOICE

Table Data Indexes Model Constraints Grants Statistics UI Defaults Triggers Dependencies SQL REST Sample Queries

```

CREATE TABLE "INVOICE"
(
  "INVOICENUMBER" VARCHAR2(15) COLLATE "USING_NLS_COMP" NOT NULL ENABLE,
  "TOTAL_AMOUNT" NUMBER(4,2),
  "VENDORNUMBER" VARCHAR2(4) COLLATE "USING_NLS_COMP",
  "PAYMENTID" VARCHAR2(4) COLLATE "USING_NLS_COMP",
  "DEPARTMENTID" VARCHAR2(4) COLLATE "USING_NLS_COMP",
  CONSTRAINT "INVOICE_PK" PRIMARY KEY ("INVOICENUMBER")
)
USING INDEX ENABLE
)
DEFAULT COLLATION "USING_NLS_COMP"

/
ALTER TABLE "INVOICE" ADD CONSTRAINT "INVOICE_fk" FOREIGN KEY ("VENDORNUMBER")
REFERENCES "VENDOR" ("VENDORNUMBER") ENABLE

/
ALTER TABLE "INVOICE" ADD CONSTRAINT "INVOICE_fk1" FOREIGN KEY ("PAYMENTID")
REFERENCES "PAYMENT" ("PAYMENTID") ENABLE

/
ALTER TABLE "INVOICE" ADD CONSTRAINT "INVOICE_fk2" FOREIGN KEY ("DEPARTMENTID")
REFERENCES "DEPARTMENT" ("DEPARTMENTID") ENABLE

/

```

Payment Table:

APEX App Builder SQL Workshop Team Development Gallery

Search

Object Browser

Schema US_A925_SQL_S34

Tables

DEPARTMENT
DEPARTMENT_EMPLOYEE
EMPLOYEE
HTMLDB_PLAN_TABLE
INVOICE
LINE_EMPLOYEE
PAYMENT
PRODUCT
PRODUCTION_LINE
RAWMATERIAL
RAWMATERIAL_NEW
SUPPLY_SCHEDULE
VENDOR
WAREHOUSE
WAREHOUSE_EMPLOYEE

PAYMENT

Table Data Indexes Model Constraints Grants Statistics UI Defaults Triggers Dependencies SQL REST Sample Queries

Add Column Modify Column Rename Column Drop Column Rename Copy Drop Truncate Create Lookup Table Create App

| Column Name | Data Type | Nullable | Default | Primary Key |
|-------------------|--------------|----------|---------|-------------|
| PAYMENTID | VARCHAR2(4) | No | - | 1 |
| VENDORNUMBER | VARCHAR2(4) | Yes | - | - |
| VENDORPAYMENTTYPE | VARCHAR2(25) | Yes | - | - |

Download | Print

APEX App Builder SQL Workshop Team Development Gallery

Search

Object Browser

Schema US_A925_SQL_S34

Tables

DEPARTMENT
DEPARTMENT_EMPLOYEE
EMPLOYEE
HTMLDB_PLAN_TABLE
INVOICE
LINE_EMPLOYEE
PAYMENT
PRODUCT
PRODUCTION_LINE
RAWMATERIAL
RAWMATERIAL_NEW
SUPPLY_SCHEDULE
VENDOR
WAREHOUSE
WAREHOUSE_EMPLOYEE

PAYMENT

Table Data Indexes Model Constraints Grants Statistics UI Defaults Triggers Dependencies SQL REST Sample Queries

```
CREATE TABLE "PAYMENT"
(
  "PAYMENTID" VARCHAR2(4) COLLATE "USING_NLS_COMP" NOT NULL ENABLE,
  "VENDORNUMBER" VARCHAR2(4) COLLATE "USING_NLS_COMP",
  "VENDORPAYMENTTYPE" VARCHAR2(25) COLLATE "USING_NLS_COMP",
  CONSTRAINT "PAYMENT_PK" PRIMARY KEY ("PAYMENTID")
  USING INDEX ENABLE
) DEFAULT COLLATION "USING_NLS_COMP"
/
ALTER TABLE "PAYMENT" ADD CONSTRAINT "PAYMENT_FK" FOREIGN KEY ("VENDORNUMBER")
REFERENCES "VENDOR" ("VENDORNUMBER") ENABLE
/
```

Query 1:

Filter employees who designed products with the product type 'Jersey'.

```
SELECT e.employeeid, e.FirstName, e.LastName, e.position, e.workcategory  
  
FROM Employee e  
  
JOIN department_employee de ON e.employeeid = de.employeeid  
  
JOIN product p ON de.employeeid = p.DesignerID  
  
WHERE p.producttype = 'Jersey';
```

The screenshot shows the APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. A search bar and user information 'us_a925_sql_s34' are on the right. The 'SQL Commands' panel shows the query being executed. The 'Results' panel at the bottom displays the output of the query in a table format.

| EMPLOYEEID | FIRSTNAME | LASTNAME | POSITION | WORKCATEGORY |
|------------|-----------|----------|-------------------|--------------|
| E016 | Ken | Miles | Design Engineer | Design |
| E017 | Sabu | Cyril | System Architect | Design |
| E018 | Sharmista | Roy | Database designer | Design |

Query 2:

Calculate the total amount paid to each vendor using sub-query.

SELECT

v.vendornumber,

v.vendorname,

(

SELECT COALESCE(SUM(i.total_amount), 0)

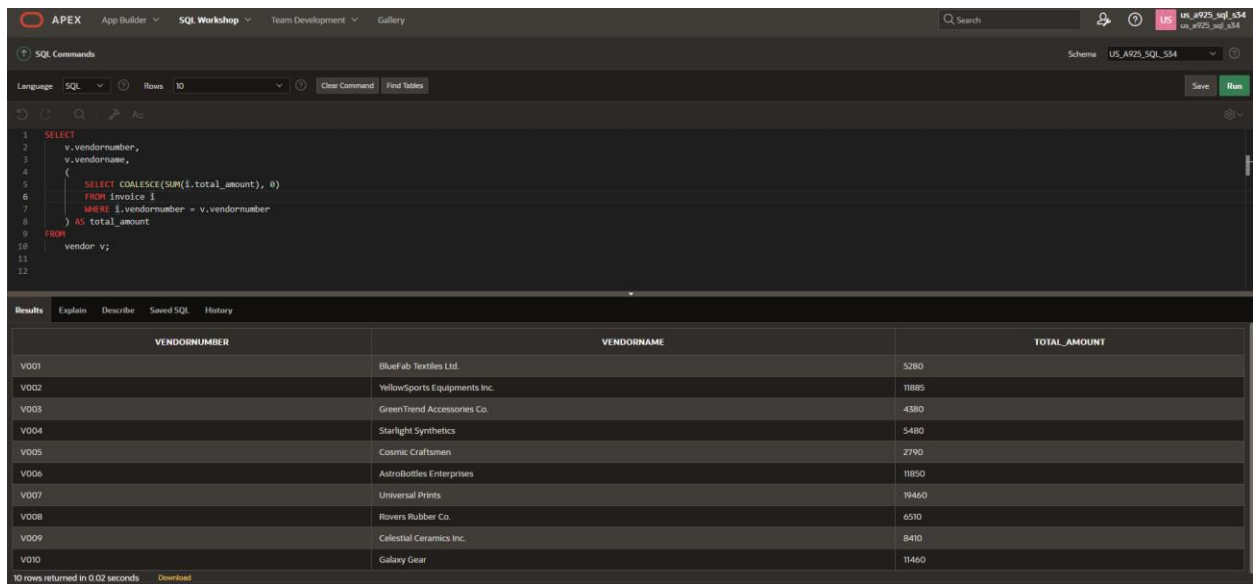
FROM invoice i

WHERE i.vendornumber = v.vendornumber

) AS total_amount

FROM

vendor v;



The screenshot shows the APEX SQL Workshop interface. The SQL Commands pane contains the following query:

```
1 SELECT
2   v.vendornumber,
3   v.vendorname,
4   (
5     SELECT COALESCE(SUM(i.total_amount), 0)
6     FROM invoice i
7     WHERE i.vendornumber = v.vendornumber
8   ) AS total_amount
9 FROM
10  vendor v;
```

The Results pane displays the output of the query, showing 10 rows of data. The columns are VENDORNUMBER, VENDORNAME, and TOTAL_AMOUNT.

| VENDORNUMBER | VENDORNAME | TOTAL_AMOUNT |
|--------------|------------------------------|--------------|
| V001 | BlueFab Textiles Ltd. | 5280 |
| V002 | YellowSports Equipments Inc. | 11885 |
| V003 | GreenTrend Accessories Co. | 4380 |
| V004 | Starlight Synthetics | 5480 |
| V005 | Cosmic Craftsmen | 2790 |
| V006 | AstroBottles Enterprises | 11850 |
| V007 | Universal Prints | 19480 |
| V008 | Rovers Rubber Co. | 6510 |
| V009 | Celestial Ceramics Inc. | 8410 |
| V010 | Galaxy Gear | 11460 |

10 rows returned in 0.02 seconds

Query 3:

Retrieve department details along with the total salary spent on each department using Joins and Sub-Query.

SELECT

d.departmentid,

d.departmentname,

d.numberofemployees,

(

SELECT COALESCE(SUM(e.salary), 0)

FROM department_employee ed

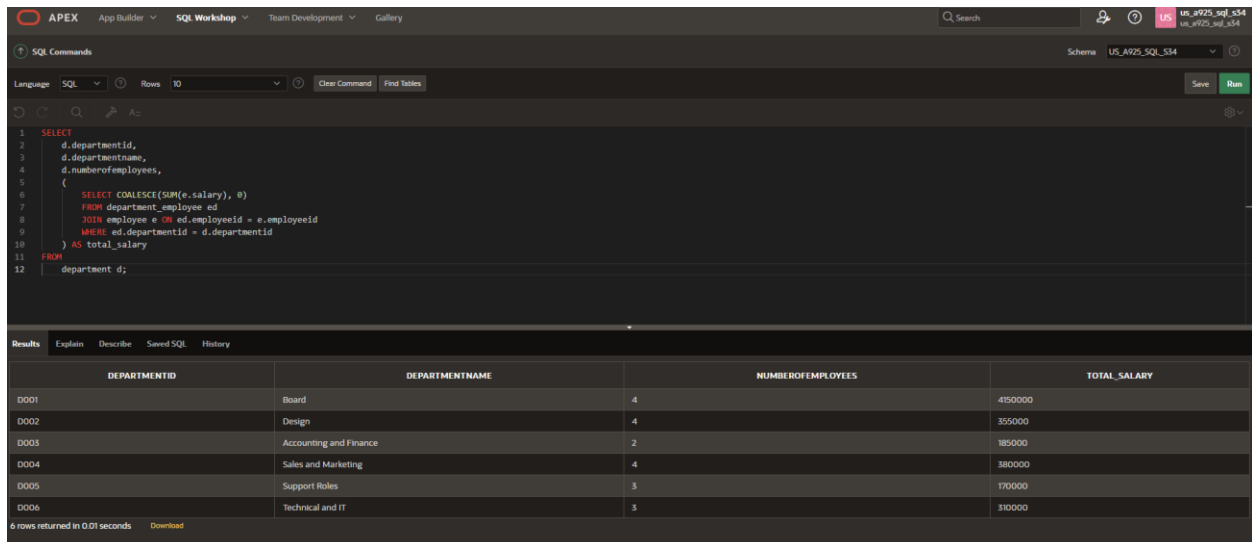
JOIN employee e ON ed.employeeid = e.employeeid

WHERE ed.departmentid = d.departmentid

) AS total_salary

FROM

department d;



The screenshot shows the APEX SQL Workshop interface. The SQL command window contains the following query:

```
1 SELECT
2   d.departmentid,
3   d.departmentname,
4   d.numberofemployees,
5   (
6     SELECT COALESCE(SUM(e.salary), 0)
7   FROM department_employee ed
8   JOIN employee e ON ed.employeeid = e.employeeid
9   WHERE ed.departmentid = d.departmentid
10  ) AS total_salary
11 FROM
12  department d;
```

The Results window displays the following data:

| DEPARTMENTID | DEPARTMENTNAME | NUMBEROFEMPLOYEES | TOTAL_SALARY |
|--------------|------------------------|-------------------|--------------|
| D001 | Board | 4 | 4150000 |
| D002 | Design | 4 | 355000 |
| D003 | Accounting and Finance | 2 | 185000 |
| D004 | Sales and Marketing | 4 | 380000 |
| D005 | Support Roles | 3 | 170000 |
| D006 | Technical and IT | 3 | 310000 |

6 rows returned in 0.01 seconds

Conclusion

In this transformative project, our team crafted an Enhanced Entity-Relationship (EER) Database Schema for modern manufacturing and supply chain operations. The focus was on optimizing workflow efficiency, particularly in employee management, product lifecycle oversight, raw material flow, and vendor invoice processing. Beginning with a Unified Metadata and Business Rules Repository, the team strategically identified primary and foreign keys, leading to a pre-normalization EER diagram. A relational schema was developed, normalized, and refined to address anomalies. Comprehensive integrity constraints were implemented to ensure data accuracy. The resulting database, adhering to normalization principles, offers a structured and efficient solution. SQL queries demonstrated the schema's practical application. Overall, the project delivers a robust foundation for manufacturing entities, emphasizing adaptability to evolving business requirements.

References

1. Guide to different keys and constraints
<https://www.essentialsql.com/get-ready-to-learn-sql-7-simplified-data-modeling/>
2. Guide to supertype and subtype relationship
<https://medium.com/nerd-for-tech/subtypes-and-supertypes-ef3b6b37c250>
3. Coronel, C., Morris, S., & Rob, P. (2010). *Database Systems: Design, Implementation, and Management* (9th ed.). Joe Sabatino.
4. A simple, understandable guide to first through three normal forms. Retrieved from <https://condor.depaul.edu/gandrus/240IT/accesspages/normalization2.htm>
5. DatabaseModels. The Best Database Support Community. Retrieved from <https://datamodels.databases.biz/>
6. Ramakrishnan, R., & Gehrke, J. (2003). *Database Management Systems* (2nd ed.). McGraw-Hill.
7. Hoffer, J. A., Ramesh, V., & Topi, H. (2019). *Modern Database Management* (13th ed.). Pearson.