IST 423: Application Development Case Study

I. Background:

Beyond Borders: Books and More, a California-based chain of five bookstores, has recently appointed Zachary Frazier, formerly the CIO, as the new CEO. The company aims to transition from a vendor-based system to a centralized "world brain" or master database that encompasses all business operations. As part of my role as a student in the IST 423: Information Storage & Retrieval course, I have been tasked to develop the framework for this database. The initial phase focuses on creating a database covering the core areas of the business: Books, Sales, Café, and Staff. Additionally, the database should be designed to eventually integrate further details concerning products, publishers, and vendors related to Books, Music, Movies, and Merchandise. The CEO emphasizes that effectively handling the four primary elements is crucial for the initial phase of the database development.

Operations can be categorized into:

Essential Areas:

- **Books:** Managing details related to books sold or stocked.

- **Sales:** Tracking sales transactions across all stores.
- **Café:** Operations associated with the café within the stores.
- **Staff:** Information about employee hours, roles, and other employment details.

Extra Credit Areas:

- **Products, Publishers, and Vendors:** Additional information for books, music, movies, and merchandise.
- **Music, Movies, Merch:** Expanding the inventory database to include these categories.

Key Views for the Database:

- **Store Inventory:** Tracks how many copies of each book are available in a particular store.
- **Store Sales:** Data specific to sales in a single store.
- **Company-Wide Sales/Stock Data:** Aggregated data showing total copies of books sold and remaining stock across all locations.
- **Sales/Stock Data for Each Store:** Detailed data for individual stores concerning sales and remaining stock of specific books.

- **Employee Hours:** Records hours worked by each employee within a given period.
- **Weekly Store Profits:** Calculates profits for each store on a weekly basis, derived from sales minus staff costs.

Requirements for the Successful Completion of Case Study:

- **ER Diagram:** To visually represent entities in the database and their relationships.
- **SQL Queries/Views:** Necessary for fetching and managing the views mentioned above.
- **Data Type and Field Definition:** Specific definitions for each column in the database, ensuring appropriate data handling and storage.

II. Project Scope/Parts

- **What I planned to do and How I did it:**

  - Understand the exact requirements of the case study by just understanding the case scenario and breaking it into parts.
    - **How I did it:** Wrote down what I understood for future reference.

- Grasp the exact entities and their associated attributes, in addition to the relationships needed for the database.
    - **How I did it:** I created a physical visual of how the ER Diagram might look like.
- Practice one diagram in ERDPlus.com.
    - **How I did it:** Accessed the tool online and connected the entities I had previously written down.
- Create a final ER Diagram for the Beyond Borders database.
    - **How I did it:** Using the ERDPlus tool.
- Convert the ER Diagram into a Relational Schema.
    - **How I did it:** Using the ERDPlus tool and then fixing issues with the automatically converted one with things like primary and foreign keys.
- Convert schema to SQL code.
    - **How I did it:** Using ERDPlus and fixed issues with code, specifically getting the datatypes right.
- Create fictional data.
    - **How I did it:** Plugged in all entities as sheets in an Excel workbook and navigated to Mockaroo to generate data.
- Access pgadmin to create the database.
    - **How I did it:** Visiting Azure Labs and turning the virtual machine on.
- Create the tables using the SQL code.

    - **How I did it:** Run the entire code.
- Import the data generated from Mockaroo into the tables.
    - **How I did it:** Using COPY file statements after having moved the files to the C:\ drive to resolve the "denied permission" issue.
- Create each view.
    - **How I did it:** Using CREATE VIEW statements and joining multiple tables depending on the view.

III. Challenges

- Some challenges after having imported the data to create views. Getting the items related to only things on the Café menu. It turned out that I had used the same product IDs for all product types whether it was Café items, books, movies, or music. This issue was easily resolved after I went back to the data generated and recreated unique product IDs for each item of every type. I had to then delete the entire database and recreate a new one with the new data.
- Issues with denied permission to copy flat file data into local tables. I resolved that by copying the files to the C:\ Drive.

- Issues with Mockaroo and manually telling it (through their AI tool) to generate the data I wanted to see for things, not under the list of data types they offer.

IV. Benefits

- Ensures that all information is accessible from a single source.
- Real-time access to inventory data across all locations. The database can help manage stock levels more effectively. This includes tracking how many copies of each item are available, which can aid in optimizing stock orders.
- Tracking of sales data at both individual store levels and company-wide.
- Managing staff schedules, payroll, and compliance with labor laws more efficiently.
- Views of financial data such as weekly profits per store, integrating sales data with staff costs.
- Customized reports can be generated to meet specific needs as well.
- Improved management of inventory and sales data not only enhances operational efficiency but also improves customer service.

V. Citations

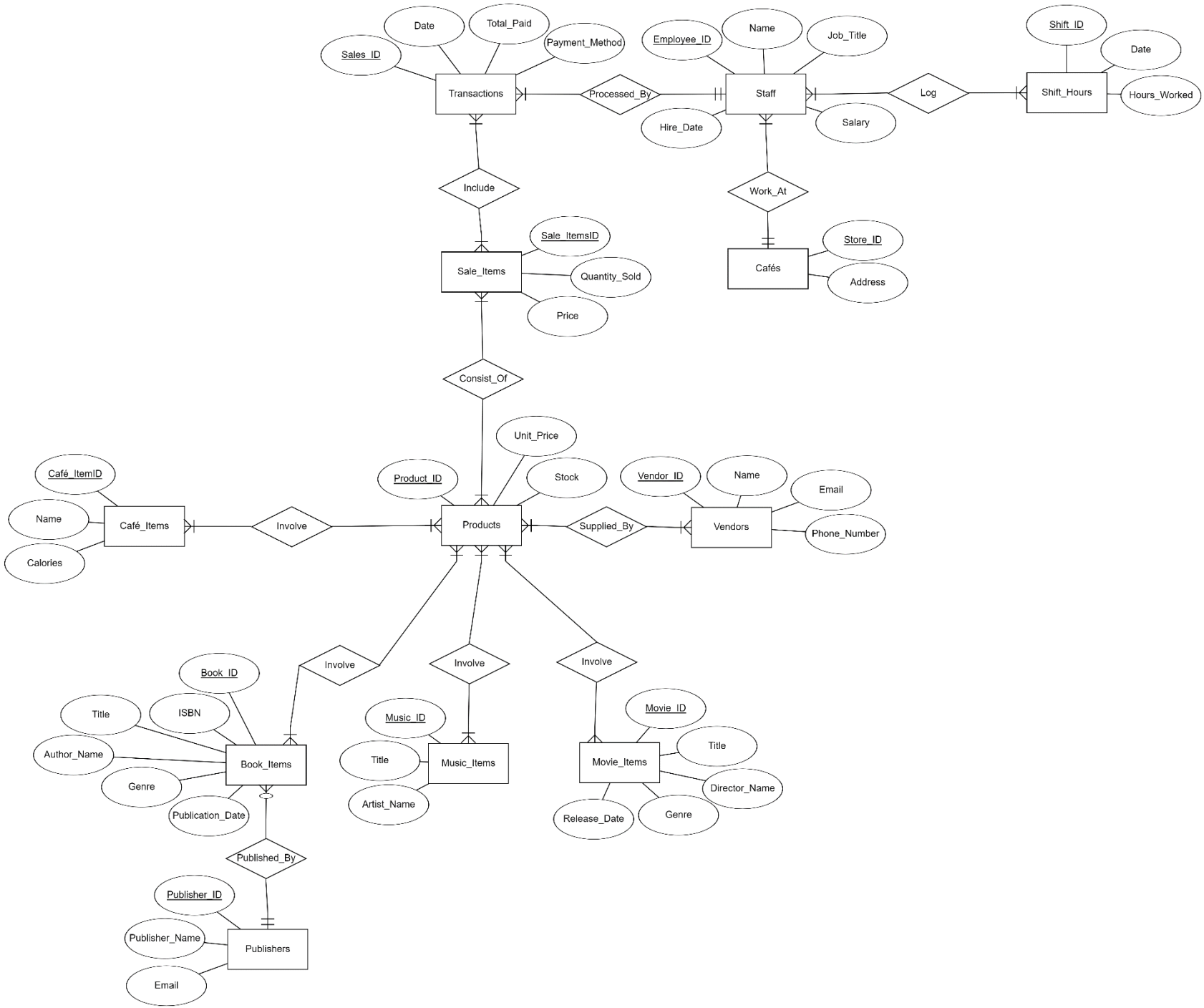
ERDPlus. (n.d.). ERD. Retrieved April 13, 2024, from https://erdplus.com/

Mockaroo - random data generator and API mocking tool. (n.d.). Mockaroo. Retrieved April 13, 2024, from https://mockaroo.com/

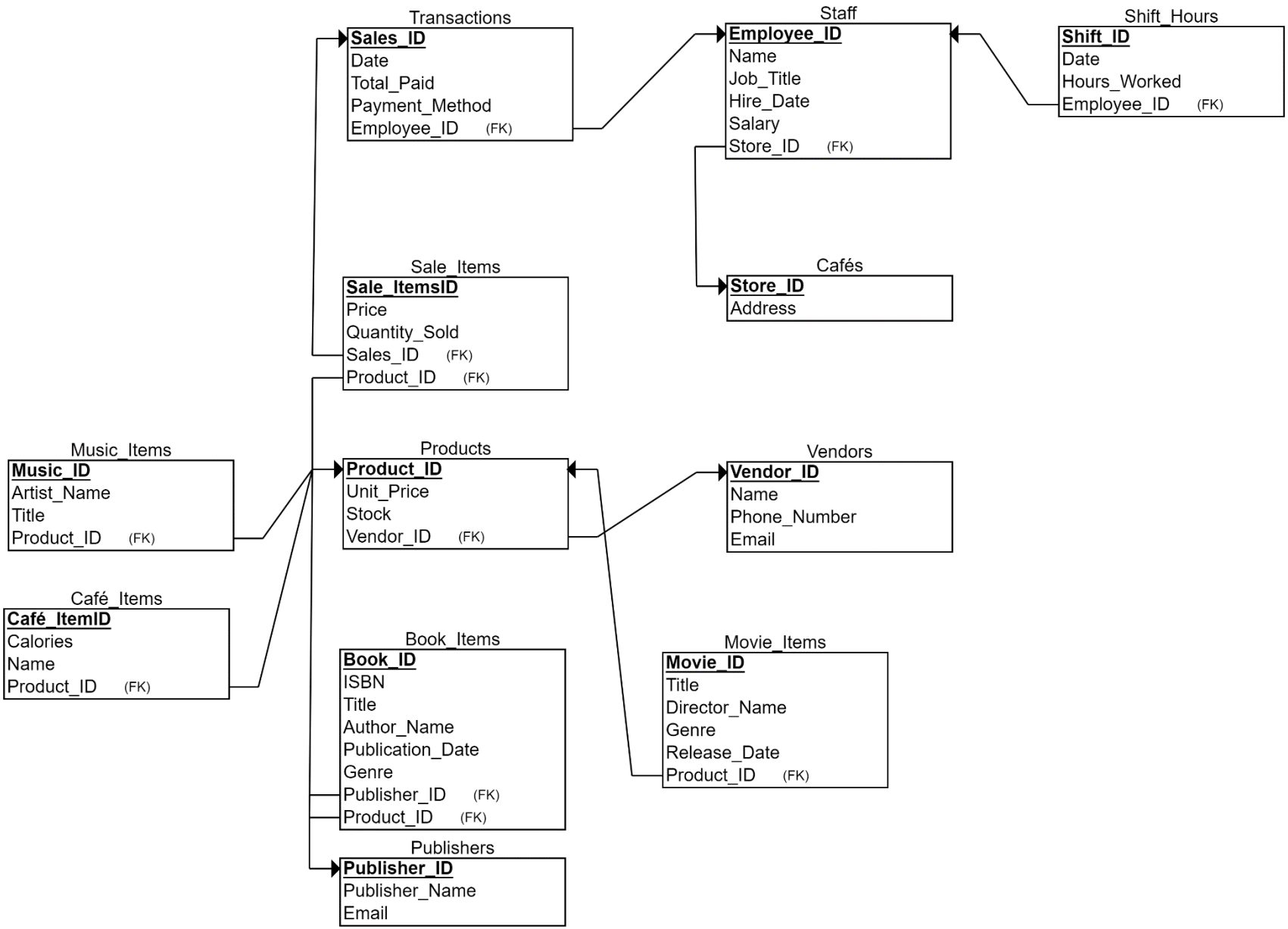
PostgreSQL: documentation. (n.d.). The PostgreSQL Global Development Group. https://www.postgresql.org/docs/

VI. Appendices

- **ER Diagram:**


Date
Total_Paid
Payment_Method
Sales_ID
Transactions
Processed_By
Employee_ID
Name
Job_Title
Staff
Hire_Date
Salary
Log
Shift_ID
Date
Shift_Hours
Hours_Worked
Include
Work_At
Sale_ItemsID
Sale_Items
Quantity_Sold
Price
Store_ID
Cafés
Address
Consist_Of
Unit_Price
Product_ID
Stock
Products
Café_ItemID
Name
Café_Items
Calories
Involve
Supplied_By
Vendor_ID
Name
Email
Vendors
Phone_Number
Involve
Involve
Involve
Book_ID
Title
ISBN
Author_Name
Book_Items
Genre
Publication_Date
Music_ID
Title
Music_Items
Artist_Name
Movie_ID
Title
Movie_Items
Director_Name
Release_Date
Genre
Published_By
Publisher_ID
Publisher_Name
Publishers
Email

- **Relational Schema:**



- SQL Queries for Tables Creation:

```
/*
Creation of Tables for Beyond Borders Database.
*/

-- Table 1: Cafes

CREATE TABLE Cafes (
    Store_ID VARCHAR(50) PRIMARY KEY,
    Address VARCHAR(200) NOT NULL
);

-- Table 2: Staff

CREATE TABLE Staff (
    Employee_ID VARCHAR(50) PRIMARY KEY,
    Name VARCHAR(50) NOT NULL,
    Job_Title VARCHAR(50) NOT NULL,
    Hire_Date DATE NOT NULL,
    Salary NUMERIC(10, 2) NOT NULL,
    Store_ID VARCHAR(50),
    FOREIGN KEY (Store_ID) REFERENCES Cafes(Store_ID)
);


-- Table 3: Transactions

CREATE TABLE Transactions (
    Sales_ID VARCHAR(255) PRIMARY KEY,
       Date DATE NOT NULL,
    Total_Paid NUMERIC(10, 2) NOT NULL,
    Payment_Method VARCHAR(50) NOT NULL,
    Employee_ID VARCHAR(50),
    FOREIGN KEY (Employee_ID) REFERENCES Staff(Employee_ID)
);

-- Table 4: Shift_Hours

CREATE TABLE Shift_Hours (
```

```
    Shift_ID SERIAL PRIMARY KEY,
    Date DATE NOT NULL,
    Hours_Worked NUMERIC(10, 2) NOT NULL,
    Employee_ID VARCHAR(50),
    FOREIGN KEY (Employee_ID) REFERENCES Staff(Employee_ID)
);

-- Table 5: Vendors

CREATE TABLE Vendors (
    Vendor_ID SERIAL PRIMARY KEY,
    Name VARCHAR(255) NOT NULL,
    Phone_Number VARCHAR(50),
    Email VARCHAR(200)
);

-- Table 6: Products

CREATE TABLE Products (
    Product_ID VARCHAR(255) PRIMARY KEY,
    Unit_Price NUMERIC(10, 2) NOT NULL,
       Stock INTEGER NOT NULL,
    Vendor_ID INTEGER,
    FOREIGN KEY (Vendor_ID) REFERENCES Vendors(Vendor_ID)
);

-- Table 7: Sale Items

CREATE TABLE Sale_Items (
    Sale_ItemsID SERIAL PRIMARY KEY,
       Price NUMERIC(10, 2) NOT NULL,
    Quantity_Sold INTEGER NOT NULL,
    Sales_ID VARCHAR(255) REFERENCES Transactions(Sales_ID),
    Product_ID  VARCHAR(255) REFERENCES Products(Product_ID)
);

-- Table 8: Cafe Items

CREATE TABLE Cafe_Items (
    Cafe_ItemID VARCHAR(255) PRIMARY KEY,
    Name TEXT NOT NULL,
    Calories INTEGER NOT NULL,
```

```
    Product_ID VARCHAR(255) REFERENCES Products(Product_ID)
);

-- Table 9: Movie Items

CREATE TABLE Movie_Items (
    Movie_ID SERIAL PRIMARY KEY,
    Title TEXT NOT NULL,
    Director_Name TEXT NOT NULL,
    Genre TEXT NOT NULL,
    Release_Date DATE NOT NULL,
    Product_ID VARCHAR(255) REFERENCES Products(Product_ID)
);

-- Table 10: Music Items

CREATE TABLE Music_Items (
    Music_ID SERIAL PRIMARY KEY,
    Title TEXT NOT NULL,
    Artist_Name TEXT NOT NULL,
    Product_ID VARCHAR(255) REFERENCES Products(Product_ID)
);


-- Table 11: Book Publishers

CREATE TABLE Publishers (
    Publisher_ID SERIAL PRIMARY KEY,
    Publisher_Name TEXT NOT NULL,
    Email TEXT NOT NULL
);

-- Table 12: Book Items

CREATE TABLE Book_Items (
    Book_ID SERIAL PRIMARY KEY,
    ISBN VARCHAR(50) NOT NULL,
    Title TEXT NOT NULL,
    Author_Name TEXT NOT NULL,
    Genre TEXT NOT NULL,
    Publication_Date DATE NOT NULL,
    Product_ID VARCHAR(255) REFERENCES Products(Product_ID),
```

```
    Publisher_ID INTEGER REFERENCES Publishers(Publisher_ID)
);
```

- **SQL Queries to Import Data from Flat Files into Local Tables:**

```
/*
Importing data from my flat files to the local tables created.
*/

-- Table 1: Cafes

Copy Cafes
FROM 'C:\Cafes.csv'
DELIMITER ','
CSV HEADER;

-- Table 2: Staff

Copy Staff
FROM 'C:\Staff.csv'
DELIMITER ','
CSV HEADER;

-- Table 3: Transactions

Copy Transactions
FROM 'C:\Transactions.csv'
DELIMITER ','
CSV HEADER;

-- Table 4: Shift_Hours

Copy Shift_Hours
FROM 'C:\Shift Hours.csv'
DELIMITER ','
CSV HEADER;

-- Table 5: Vendors
```

```
Copy Vendors
FROM 'C:\Vendors.csv'
DELIMITER ','
CSV HEADER;

-- Table 6: Products

Copy Products
FROM 'C:\Products.csv'
DELIMITER ','
CSV HEADER;

-- Table 7: Sale Items

Copy Sale_Items
FROM 'C:\Sale Items.csv'
DELIMITER ','
CSV HEADER;

-- Table 8: Cafe Items

Copy Cafe_Items
FROM 'C:\Cafe Items.csv'
DELIMITER ','
CSV HEADER;

-- Table 9: Movie Items

Copy Movie_Items
FROM 'C:\Movie Items.csv'
DELIMITER ','
CSV HEADER;

-- Table 10: Music Items

Copy Music_Items
FROM 'C:\Music Items.csv'
DELIMITER ','
CSV HEADER;

-- Table 11: Publishers
```

```
Copy Publishers
FROM 'C:\Publishers.csv'
DELIMITER ','
CSV HEADER;

-- Table 12: Book Items

Copy Book_Items
FROM 'C:\Book Items.csv'
DELIMITER ','
CSV HEADER;
```

- **SQL Query for Store Inventory View:**

```
CREATE VIEW Story_Inventory AS
SELECT
    c.Store_ID,
    c.Address,
    p.Product_ID,
    COALESCE(ci.Name, mi.Title, mui.Title, bi.Title) AS Product_Name,
    p.Stock
FROM
    Cafes c
CROSS JOIN
    Products p
LEFT JOIN
    Cafe_Items ci ON p.Product_ID = ci.Product_ID
LEFT JOIN
    Movie_Items mi ON p.Product_ID = mi.Product_ID
LEFT JOIN
    Music_Items mui ON p.Product_ID = mui.Product_ID
LEFT JOIN
    Book_Items bi ON p.Product_ID = bi.Product_ID;

SELECT * FROM Story_Inventory
```

```
ORDER BY Store_ID, Stock DESC;
```

- **SQL Query for Store Sales View:**

```
CREATE VIEW Store_Sales AS
SELECT
    s.Store_ID,
    c.Address,
    COUNT(t.Sales_ID) AS Number_of_Sales,
    SUM(t.Total_Paid) AS Total_Sales
FROM
    Cafes c
JOIN
    Staff s ON c.Store_ID = s.Store_ID
JOIN
    Transactions t ON s.Employee_ID = t.Employee_ID
GROUP BY
    s.Store_ID, c.Address;


select * from Store_Sales
```

- **SQL Query for Company Wide Sales & Stock View:**

```
/*
Please note that if any Product_ID does not have any corresponding sales, it will show
as having 0.
*/

CREATE VIEW Company_Wide_Sales_Stock AS
SELECT
    p.Product_ID,
```

```
    COALESCE(ci.Name, mi.Title, mui.Title, bi.Title) AS Product_Name,
    COALESCE(SUM(si.Quantity_Sold), 0) AS Total_Copies_Sold,
    p.Stock AS Initial_Stock_Number,
    CASE
        WHEN COALESCE(SUM(si.Quantity_Sold), 0) = 0 THEN p.Stock
        ELSE p.Stock - COALESCE(SUM(si.Quantity_Sold), 0)
    END AS Copies_Remain
FROM
    Products p
LEFT JOIN
    Sale_Items si ON p.Product_ID = si.Product_ID
LEFT JOIN
    Cafe_Items ci ON p.Product_ID = ci.Product_ID
LEFT JOIN
    Movie_Items mi ON p.Product_ID = mi.Product_ID
LEFT JOIN
    Music_Items mui ON p.Product_ID = mui.Product_ID
LEFT JOIN
    Book_Items bi ON p.Product_ID = bi.Product_ID
GROUP BY
    p.Product_ID, p.Stock, ci.Name, mi.Title, mui.Title, bi.Title;


select * from Company_Wide_Sales_Stock
```

- **SQL Query for Store Specific Sales & Stock View:**

```
CREATE VIEW Store_Specific_Sales_Stock AS
SELECT
    c.Store_ID,
    c.Address,
    p.Product_ID,
    COALESCE(ci.Name, mi.Title, mui.Title, bi.Title) AS Product_Name,
    COALESCE(SUM(si.Quantity_Sold), 0) AS Total_Copies_Sold,
    p.Stock AS Initial_Stock_Number,
    CASE
        WHEN COALESCE(SUM(si.Quantity_Sold), 0) = 0 THEN p.Stock
```

```
        ELSE p.Stock - COALESCE(SUM(si.Quantity_Sold), 0)
    END AS Copies_Remain
FROM
    Cafes c
JOIN
    Staff s ON c.Store_ID = s.Store_ID
JOIN
    Transactions t ON s.Employee_ID = t.Employee_ID
JOIN
    Sale_Items si ON t.Sales_ID = si.Sales_ID
JOIN
    Products p ON si.Product_ID = p.Product_ID
LEFT JOIN
    Cafe_Items ci ON p.Product_ID = ci.Product_ID
LEFT JOIN
    Movie_Items mi ON p.Product_ID = mi.Product_ID
LEFT JOIN
    Music_Items mui ON p.Product_ID = mui.Product_ID
LEFT JOIN
    Book_Items bi ON p.Product_ID = bi.Product_ID
GROUP BY
    c.Store_ID, c.Address, p.Product_ID, p.Stock, ci.Name, mi.Title, mui.Title, bi.Title;


select * from Store_Specific_Sales_Stock
```

- **SQL Query for Employee Hours View:**

```
CREATE VIEW Employee_Hours AS
SELECT
    s.Employee_ID,
    s.Name AS Employee_Name,
    s.Store_ID,
    SUM(sh.Hours_Worked) AS Total_Hours_Worked
FROM
```

```
    Staff s
JOIN
    Shift_Hours sh ON s.Employee_ID = sh.Employee_ID
GROUP BY
    s.Employee_ID, s.Name, s.Store_ID;

select * from Employee_Hours
```

- **SQL Query for Weekly Store Profits View:**

```
CREATE VIEW Weekly_Store_Profits AS
SELECT
    c.Store_ID,
    c.Address,
    DATE_TRUNC('week', t.Date) AS Week,
    SUM(t.Total_Paid) AS Weekly_Sales,
    (SELECT SUM(Salary) / 52 FROM Staff s WHERE s.Store_ID = c.Store_ID) AS Weekly_Staff_Cost,
    SUM(t.Total_Paid) - (SELECT SUM(Salary) / 52 FROM Staff s WHERE s.Store_ID = c.Store_ID) AS Weekly_Profit
FROM
    Cafes c
JOIN
    Staff s ON c.Store_ID = s.Store_ID
JOIN
    Transactions t ON s.Employee_ID = t.Employee_ID
GROUP BY
    c.Store_ID, c.Address, DATE_TRUNC('week', t.Date);


select * from Weekly_Store_Profits
order by Store_ID, Week asc
```

- **Field Definition, Data Types, & Constraints:**

Table Name	Field Name	Data Type	Constraints
Cafes	Store_ID	VARCHAR(50)	PRIMARY KEY
	Address	VARCHAR(200)	NOT NULL
Staff	Employee_ID	VARCHAR(50)	PRIMARY KEY
	Name	VARCHAR(50)	NOT NULL
	Job_Title	VARCHAR(50)	NOT NULL
	Hire_Date	DATE	NOT NULL
	Salary	NUMERIC(10, 2)	NOT NULL
	Store_ID	VARCHAR(50)	FOREIGN KEY REFERENCES Cafes
Transactions	Sales_ID	VARCHAR(255)	PRIMARY KEY
	Date	DATE	NOT NULL
	Total_Paid	NUMERIC(10, 2)	NOT NULL
	Payment_Method	VARCHAR(50)	NOT NULL
	Employee_ID	VARCHAR(50)	FOREIGN KEY REFERENCES Staff
Shift_Hours	Shift_ID	SERIAL	PRIMARY KEY
	Date	DATE	NOT NULL
	Hours_Worked	NUMERIC(10, 2)	NOT NULL
	Employee_ID	VARCHAR(50)	FOREIGN KEY REFERENCES Staff
Vendors	Vendor_ID	SERIAL	PRIMARY KEY
	Name	VARCHAR(255)	NOT NULL
	Phone_Number	VARCHAR(50)	
	Email	VARCHAR(200)	
Products	Product_ID	VARCHAR(255)	PRIMARY KEY
	Unit_Price	NUMERIC(10, 2)	NOT NULL
	Stock	INTEGER	NOT NULL
	Vendor_ID	INTEGER	FOREIGN KEY REFERENCES Vendors
Sale Items	Sale_ItemsID	SERIAL	PRIMARY KEY

	Price	NUMERIC(10, 2)	NOT NULL
	Quantity_Sold	INTEGER	NOT NULL
	Sales_ID	VARCHAR(255)	REFERENCES Transactions
	Product_ID	VARCHAR(255)	REFERENCES Products
Cafe Items	Cafe_ItemID	VARCHAR(255)	PRIMARY KEY
	Name	TEXT	NOT NULL
	Calories	INTEGER	NOT NULL
	Product_ID	VARCHAR(255)	REFERENCES Products
Movie Items	Movie_ID	SERIAL	PRIMARY KEY
	Title	TEXT	NOT NULL
	Director_Name	TEXT	NOT NULL
	Genre	TEXT	NOT NULL
	Release_Date	DATE	NOT NULL
	Product_ID	VARCHAR(255)	REFERENCES Products
Music Items	Music_ID	SERIAL	PRIMARY KEY
	Title	TEXT	NOT NULL
	Artist_Name	TEXT	NOT NULL
	Product_ID	VARCHAR(255)	REFERENCES Products
Publishers	Publisher_ID	SERIAL	PRIMARY KEY
	Publisher_Name	TEXT	NOT NULL
	Email	TEXT	NOT NULL
Book Items	Book_ID	SERIAL	PRIMARY KEY
	ISBN	VARCHAR(50)	NOT NULL
	Title	TEXT	NOT NULL
	Author_Name	TEXT	NOT NULL
	Genre	TEXT	NOT NULL
	Publication_Date	DATE	NOT NULL
	Product_ID	VARCHAR(255)	REFERENCES Products
	Publisher_ID	INTEGER	REFERENCES Publishers