Chase Center SQL Database Project

This project involves creating and managing a SQL database for Chase Center, home stadium of the Golden State Warriors. The database structure manages the following operations:

- Ticket sales
- Merchandise sales
- Vendor and employee details

This project outlines table creation, data insertion, views, and functions necessary for running the stadium's database.

Key Features:

- **Table Creation**: For customers, tickets, employees, vendors, products, and orders.
- **Data Population**: Using external CSV files.
- **Views**: `CustomerOrderSummary` to display customer order details.
- **Functions**: Calculating total customer purchases.
- **Stored Procedures**: Fetching employee information.

Step 1: Create and Use Chase Center Database

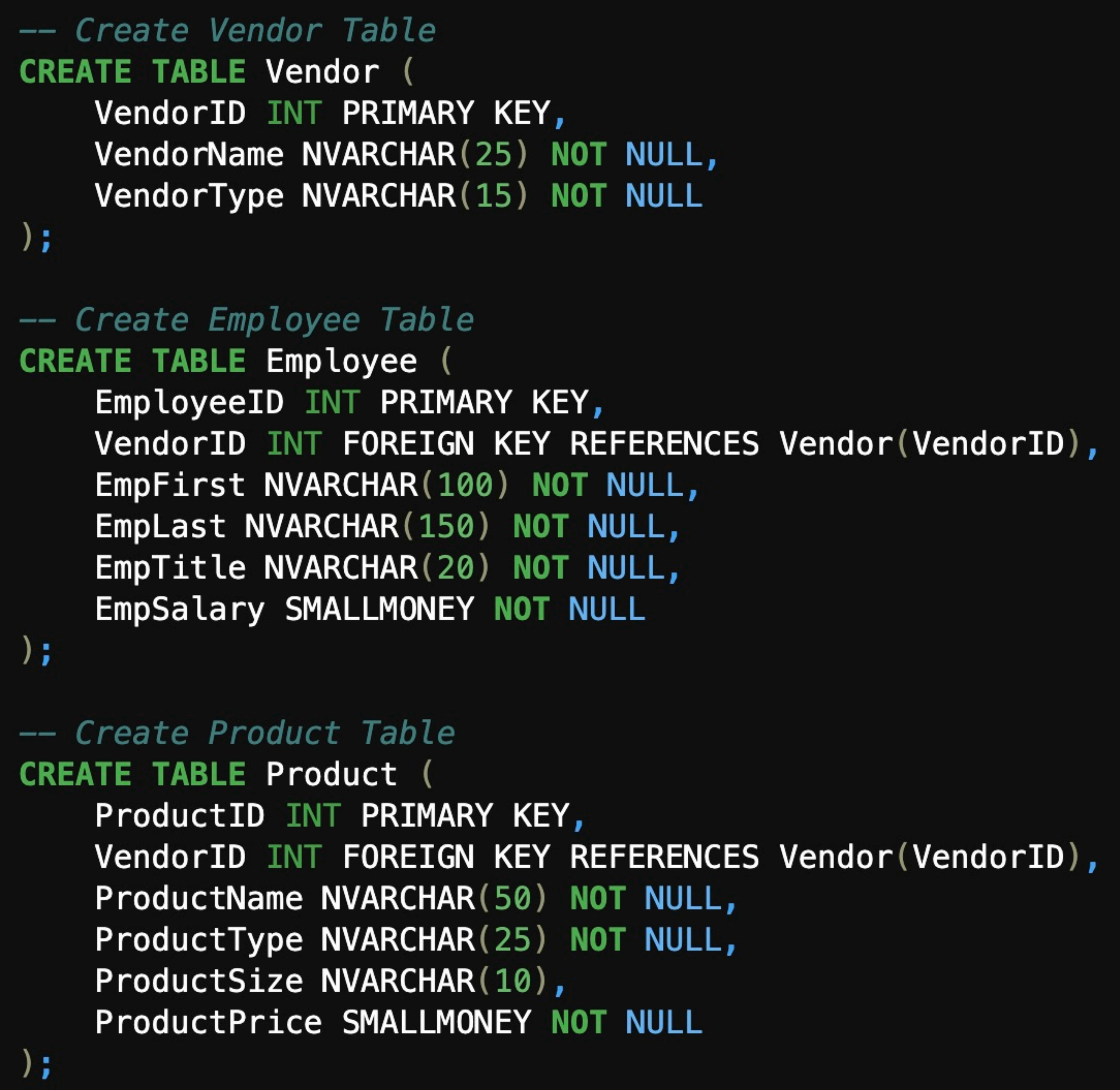
```sql
-- Create Chase Center Database and Set Data Path
IF NOT EXISTS(SELECT * FROM sys.databases WHERE NAME = N'ChaseCenter')
CREATE DATABASE ChaseCenter;
GO

-- Use the newly created Chase Center database
USE ChaseCenter;
GO

-- Define path for CSV data files
DECLARE @data_path NVARCHAR(256);
SELECT @data_path = 'C:\Users\Roshan.CranfordLuite\Desktop\ChaseCenterDatabase\';
GO
```

Step 2: Delete Existing Tables

```
-- Delete existing tables
IF EXISTS(SELECT * FROM sys.tables WHERE NAME = N'Order_Detail') DROP TABLE Order_Detail;

IF EXISTS(SELECT * FROM sys.tables WHERE NAME = N'Order') DROP TABLE [Order];

IF EXISTS(SELECT * FROM sys.tables WHERE NAME = N'Product') DROP TABLE Product;

IF EXISTS(SELECT * FROM sys.tables WHERE NAME = N'Employee') DROP TABLE Employee;

IF EXISTS(SELECT * FROM sys.tables WHERE NAME = N'Vendor') DROP TABLE Vendor;

IF EXISTS(SELECT * FROM sys.tables WHERE NAME = N'Customer_Ticket') DROP TABLE Customer_Ticket;

IF EXISTS(SELECT * FROM sys.tables WHERE NAME = N'Ticket') DROP TABLE Ticket;

IF EXISTS(SELECT * FROM sys.tables WHERE NAME = N'Customer') DROP TABLE Customer;
```

Step 3: Create Tables

```sql
-- Create Customer Table
CREATE TABLE Customer (
    CustomerID INT PRIMARY KEY,
    CustFirstName NVARCHAR(50) NOT NULL,
    CustLastName NVARCHAR(100) NOT NULL,
    CustEmail NVARCHAR(150) NOT NULL,
    CustPhone NVARCHAR(12) NOT NULL,
    CustCity NVARCHAR(50) NOT NULL,
    CustZip NVARCHAR(5) NOT NULL,
    CustState NVARCHAR(5) NOT NULL
);

-- Create Ticket Table
CREATE TABLE Ticket (
    TicketID INT PRIMARY KEY,
    TicketSection INT NOT NULL,
    TicketSeat INT NOT NULL,
    TicketRow INT NOT NULL,
    TicketPrice SMALLMONEY NOT NULL
);

-- Create Customer_Ticket Table
CREATE TABLE Customer_Ticket (
    CustomerTicketID INT PRIMARY KEY,
    CustomerID INT FOREIGN KEY REFERENCES Customer(CustomerID),
    TicketID INT FOREIGN KEY REFERENCES Ticket(TicketID),
    CustomerTicketDate DATE NOT NULL
);
```



```sql
-- Create Vendor Table
CREATE TABLE Vendor (
    VendorID INT PRIMARY KEY,
    VendorName NVARCHAR(25) NOT NULL,
    VendorType NVARCHAR(15) NOT NULL
);

-- Create Employee Table
CREATE TABLE Employee (
    EmployeeID INT PRIMARY KEY,
    VendorID INT FOREIGN KEY REFERENCES Vendor(VendorID),
    EmpFirst NVARCHAR(100) NOT NULL,
    EmpLast NVARCHAR(150) NOT NULL,
    EmpTitle NVARCHAR(20) NOT NULL,
    EmpSalary SMALLMONEY NOT NULL
);

-- Create Product Table
CREATE TABLE Product (
    ProductID INT PRIMARY KEY,
    VendorID INT FOREIGN KEY REFERENCES Vendor(VendorID),
    ProductName NVARCHAR(50) NOT NULL,
    ProductType NVARCHAR(25) NOT NULL,
    ProductSize NVARCHAR(10),
    ProductPrice SMALLMONEY NOT NULL
);
```

```sql
-- Create Order Table
CREATE TABLE [Order] (
    OrderID INT PRIMARY KEY,
    CustomerID INT FOREIGN KEY REFERENCES Customer(CustomerID),
    OrderDate DATE NOT NULL
);

-- Create Order_Detail Table
CREATE TABLE Order_Detail (
    OrderDetailID INT PRIMARY KEY,
    OrderID INT FOREIGN KEY REFERENCES [Order],
    ProductID INT FOREIGN KEY REFERENCES Product(ProductID),
    OrderDetailQty INT NOT NULL
);
```

Step 4: Populate Tables with CSV Data

For this project, CSV files are used to populate the tables. Here is a sample BULK INSERT statement:

```
EXECUTE (N'BULK INSERT Customer FROM ''' + @data_path + N'Customer.csv'' WITH (CHECK_CONSTRAINTS, FIELDTERMINATOR = '','',
ROWTERMINATOR = ''\n'')');
```

Step 5: Creating Views and Functions

```sql
-- Create a view to summarize customer orders
CREATE VIEW CustomerOrderSummary AS
SELECT o.OrderID, o.OrderDate, c.CustomerID,
       CONCAT(c.CustFirstName, ' ', c.CustLastName) AS CustomerName,
       p.ProductName, p.ProductType, p.ProductSize,
       od.OrderDetailQty, p.ProductPrice,
       (od.OrderDetailQty * p.ProductPrice) AS TotalPrice
FROM Customer AS c
INNER JOIN [Order] AS o ON c.CustomerID = o.CustomerID
INNER JOIN Order_Detail AS od ON o.OrderID = od.OrderID
INNER JOIN Product AS p ON p.ProductID = od.ProductID

-- Execute the view
SELECT * FROM CustomerOrderSummary;
```

Step 6: Running Queries and Creating Functions

```
-- Create a function to calculate the total merchandise purchased by a customer
CREATE FUNCTION dbo.CustMerchandiseTotal (@CustomerID INT)
RETURNS DECIMAL(10, 2) AS
BEGIN
    DECLARE @TotalSpent DECIMAL(10, 2);
END;

-- 2. Execute the function for a specific customer
SELECT dbo.CustMerchandiseTotal(59);
```

Step 7: Stored Procedure to Retrieve Employee Information

```sql
-- Create a procedure to retrieve all employee details
CREATE PROCEDURE EmployeeInfo AS
BEGIN
    SELECT * FROM Employee;
END;

-- Execute the procedure
EXECUTE EmployeeInfo;
```

Conclusion

This SQL project for Chase Center provides a detailed database schema and functionality for managing customer, ticket, merchandise, vendor, and employee data. The database design and queries are tailored for efficient stadium operations and customer management.