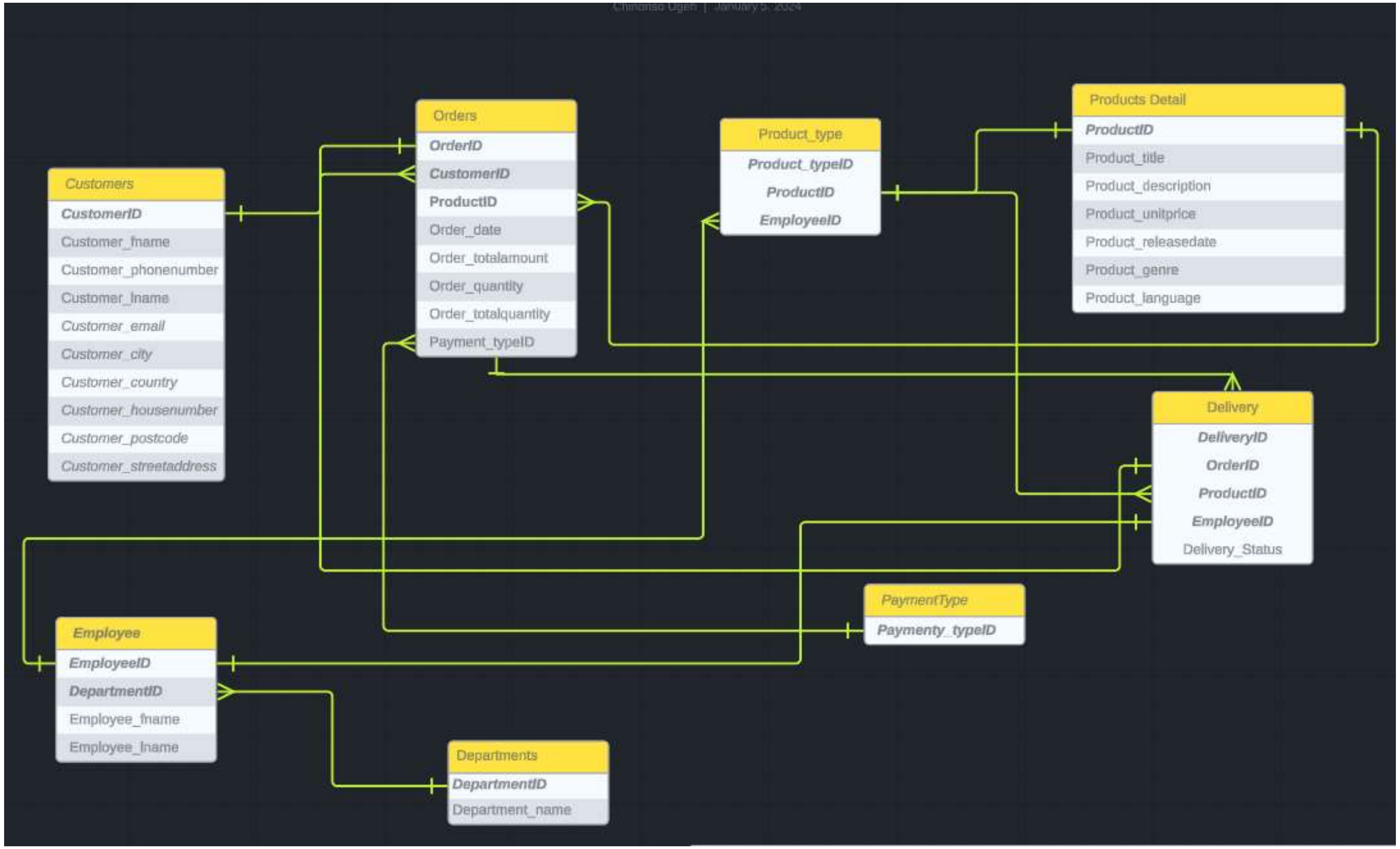
ARM | Classification: CONFIDENTIAL-INTERNALCONTROL

Table of Content

Introduction

This portfolio dives into the database that powers an online E shop, which sells a variety of products including music records, movies, and books. We'll cover how our database is structured, its key components, and why it's crucial for our day-to-day operations. From managing products and orders to understanding our customers, this system is the heart of our e-commerce business. Exploring how it all comes together to keep our e-shop running smoothly and efficiently.

1. ER Diagram for Online E-shop



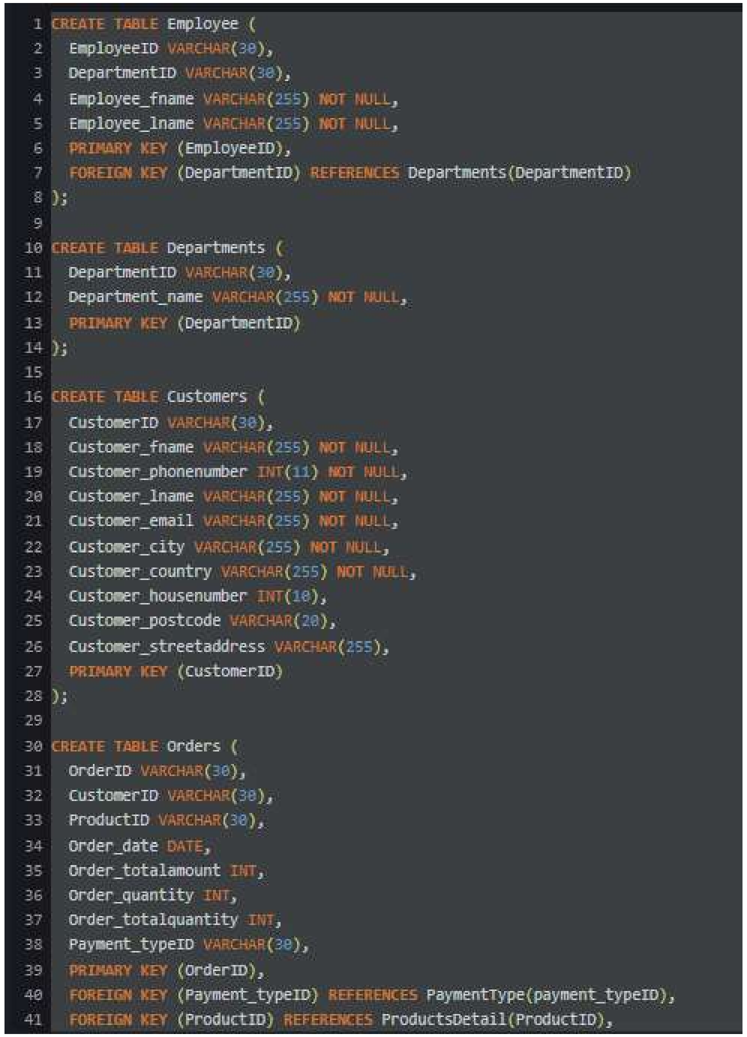
Explaining the relationships in the ER Diagram

- ER diagram expresses the overall logical structure of a database in a clear and simple image using entity sets, relationship sets and attributes. It was designed to facilitate in the development of database design and management. (Silberschatz, 2020). Expressing the number of entities to which another entity can be associated, via a relationship set is through cardinality mapping, which could be one to one, one to many, many to one and many to many (Silberschatz, 2020).

For the online E-shop database design above;

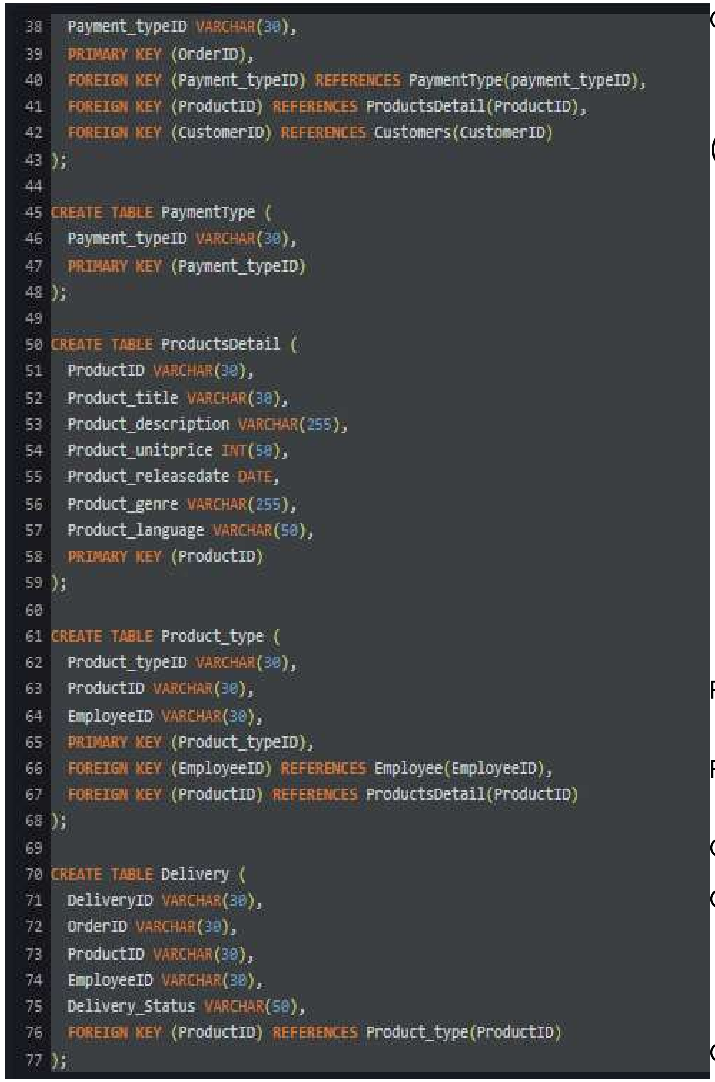
- A customer can submit one orders for a product, which is a one-to-one relationship, a customer can also place multiple orders for products, which is a one-to-many relationship.
- Employees of the store has control over the products; that is a many-to-many relationship, the employee is responsible for collecting the products for delivery, which is a many-to-one relationship
- A customer can decide to place an order using a debit or credit card or cash,; one-to-many relationship, each order is linked to a delivery and employee; that is many-to-many relationship.
- Each employee is registered to only one department, one-to-one. While a department can have multiple employees; one-to-many relationship.
- Each delivery can have only one order, which is one-to-one relationship, and can have many products; which is many-to-one relationship.

2. SQL Scripts for the ER diagram database



```
CREATE TABLE Employee (
  EmployeeID VARCHAR(30),
  DepartmentID VARCHAR(30),
  Employee_fname VARCHAR(255) NOT NULL,
  Employee_lname VARCHAR(255) NOT NULL,
  PRIMARY KEY (EmployeeID),
  FOREIGN KEY (DepartmentID) REFERENCES Departments(DepartmentID)
);

CREATE TABLE Departments (
  DepartmentID VARCHAR(30),
  Department_name VARCHAR(255) NOT NULL,
  PRIMARY KEY (DepartmentID)
);

CREATE TABLE Customers (
  CustomerID VARCHAR(30),
  Customer_fname VARCHAR(255) NOT NULL,
  Customer_phonenumber INT(11) NOT NULL,
  Customer_lname VARCHAR(255) NOT NULL,
  Customer_email VARCHAR(255) NOT NULL,
  Customer_city VARCHAR(255) NOT NULL,
  Customer_country VARCHAR(255) NOT NULL,
  Customer_housenumber INT(10),
  Customer_postcode VARCHAR(20),
  Customer_streetaddress VARCHAR(255),
  PRIMARY KEY (CustomerID)
);

CREATE TABLE Orders (
  OrderID VARCHAR(30),
  CustomerID VARCHAR(30),
  ProductID VARCHAR(30),
  Order_date DATE,
  Order_totalamount INT,
  Order_quantity INT,
  Order_totalquantity INT,
  Payment_typeID VARCHAR(30),
  PRIMARY KEY (OrderID),
  FOREIGN KEY (Payment_typeID) REFERENCES PaymentType(payment_typeID),
  FOREIGN KEY (ProductID) REFERENCES ProductsDetail(ProductID),
```

```
CREATE TABLE Employee (
 EmployeeID VARCHAR(30),
 DepartmentID VARCHAR(30),
 Employee_fname VARCHAR(255) NOT NULL,
 Employee_lname VARCHAR(255) NOT NULL,
 PRIMARY KEY (EmployeeID),
 FOREIGN KEY (DepartmentID) REFERENCES
Departments(DepartmentID));
CREATE TABLE Departments (
 DepartmentID VARCHAR(30),
 Department_name VARCHAR(255) NOT NULL,
 PRIMARY KEY (DepartmentID));
CREATE TABLE Customers (
 CustomerID VARCHAR(30),
 Customer_fname VARCHAR(255) NOT NULL,
 Customer_phonenumber INT(11) NOT NULL,
 Customer_lname VARCHAR(255) NOT NULL,
 Customer_email VARCHAR(255) NOT NULL,
 Customer_city VARCHAR(255) NOT NULL,
 Customer_country VARCHAR(255) NOT NULL,
Customer_housenumber INT(10),
```



```
  Payment_typeID VARCHAR(30),
  PRIMARY KEY (OrderID),
  FOREIGN KEY (Payment_typeID) REFERENCES PaymentType(payment_typeID),
  FOREIGN KEY (ProductID) REFERENCES ProductsDetail(ProductID),
  FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);

CREATE TABLE PaymentType (
  Payment_typeID VARCHAR(30),
  PRIMARY KEY (Payment_typeID)
);

CREATE TABLE ProductsDetail (
  ProductID VARCHAR(30),
  Product_title VARCHAR(30),
  Product_description VARCHAR(255),
  Product_unitprice INT(50),
  Product_releasedate DATE,
  Product_genre VARCHAR(255),
  Product_language VARCHAR(50),
  PRIMARY KEY (ProductID)
);

CREATE TABLE Product_type (
  Product_typeID VARCHAR(30),
  ProductID VARCHAR(30),
  EmployeeID VARCHAR(30),
  PRIMARY KEY (Product_typeID),
  FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID),
  FOREIGN KEY (ProductID) REFERENCES ProductsDetail(ProductID)
);

CREATE TABLE Delivery (
  DeliveryID VARCHAR(30),
  OrderID VARCHAR(30),
  ProductID VARCHAR(30),
  EmployeeID VARCHAR(30),
  Delivery_Status VARCHAR(50),
  FOREIGN KEY (ProductID) REFERENCES Product_type(ProductID)
);
```

```
Customer_postcode VARCHAR(20),
 Customer_streetaddress VARCHAR(255),
 PRIMARY KEY (CustomerID));CREATE TABLE Orders
(
 OrderID VARCHAR(30),
 CustomerID VARCHAR(30),
 ProductID VARCHAR(30),
 Order_date DATE,
 Order_totalamount INT,
 Order_quantity INT,
 Order_totalquantity INT,
 Payment_typeID VARCHAR(30),
 PRIMARY KEY (OrderID),
 FOREIGN KEY (Payment_typeID) REFERENCES
PaymentType(payment_typeID),
 FOREIGN KEY (ProductID) REFERENCES
ProductsDetail(ProductID),
 FOREIGN KEY (CustomerID) REFERENCES
Customers(CustomerID));
CREATE TABLE PaymentType (
 Payment_typeID VARCHAR(30),
 PRIMARY KEY (Payment_typeID));
CREATE TABLE ProductsDetail (
ProductID VARCHAR(30),
 Product_title VARCHAR(30),
 Product_description VARCHAR(255),
 Product_unitprice INT(50),
 Product_releasedate DATE,
 Product_genre VARCHAR(255),
 Product_language VARCHAR(50),
 PRIMARY KEY (ProductID) );CREATE TABLE Product_type
(
 Product_typeID VARCHAR(30),
 ProductID VARCHAR(30),
 EmployeeID VARCHAR(30),
 PRIMARY KEY (Product_typeID),
 FOREIGN KEY (EmployeeID) REFERENCES
Employee(EmployeeID),
 FOREIGN KEY (ProductID) REFERENCES
ProductsDetail(ProductID));
CREATE TABLE Delivery (
 DeliveryID VARCHAR(30),
 OrderID VARCHAR(30),
 ProductID VARCHAR(30),
 EmployeeID VARCHAR(30),
 Delivery_Status VARCHAR(50),
 FOREIGN KEY (ProductID) REFERENCES
Product_type(ProductID));
```

Explaining the SQL Scripts in the ER Diagram

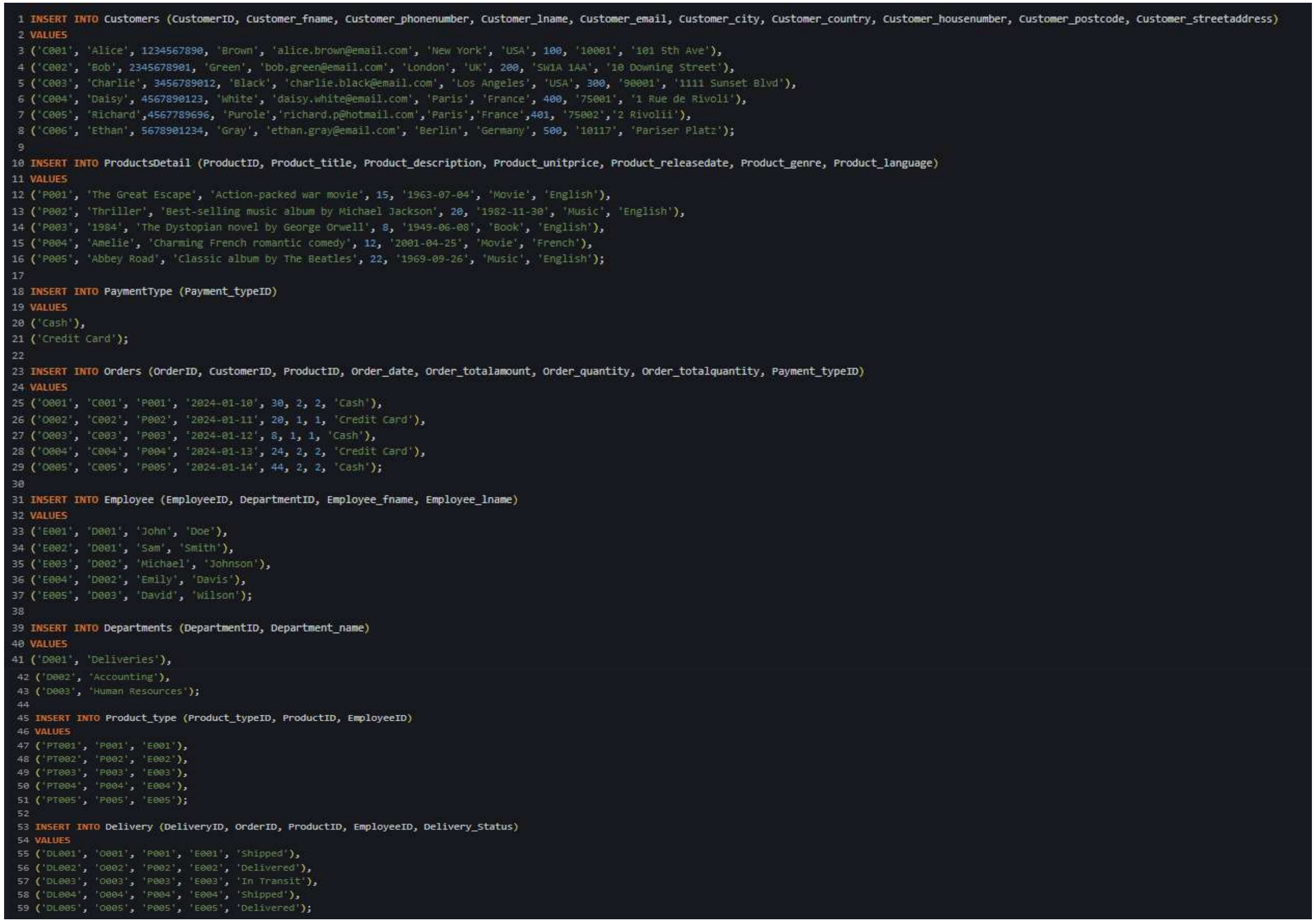
The scripts provided above in slide 3 defines a comprehensive database schema tailored for the case study an online e-shop, that deals in a variety of products such as music records, movies, and books. This schema is designed based on standard database modeling concepts and includes a series of interconnected tables, each serving a specific function in the overall data model. Key components of this schema are the use of Primary Keys, Foreign Keys, and specific Data Types, as outlined in Connolly and Begg (2015).

- **Primary Keys**: Each table in the schema has a Primary Key. A primary key is a unique identifier for each record in that table. For example, the Employee table has EmployeeID as its Primary Key. The primary purpose of a Primary Key is to ensure that each record can be uniquely identified, maintaining data integrity and data retrieval. (Date, 2019). Without Primary Keys, the risk of data duplication increases, and the relational aspect of databases would be severely compromised. (Date, 2019).

- **Foreign Keys**: The schema also extensively uses Foreign Keys, which are essential in relational database design (Silberschatz, 2020). According to Vassiliadis et al. (2019) a foreign Key in one table points to a Primary Key in another table, creating a link between them. For instance, in the Orders table, CustomerID serves as a Foreign Key linking back to the Customers table. This linkage reflects real-world relationships and dependencies, such as orders being placed by customers.

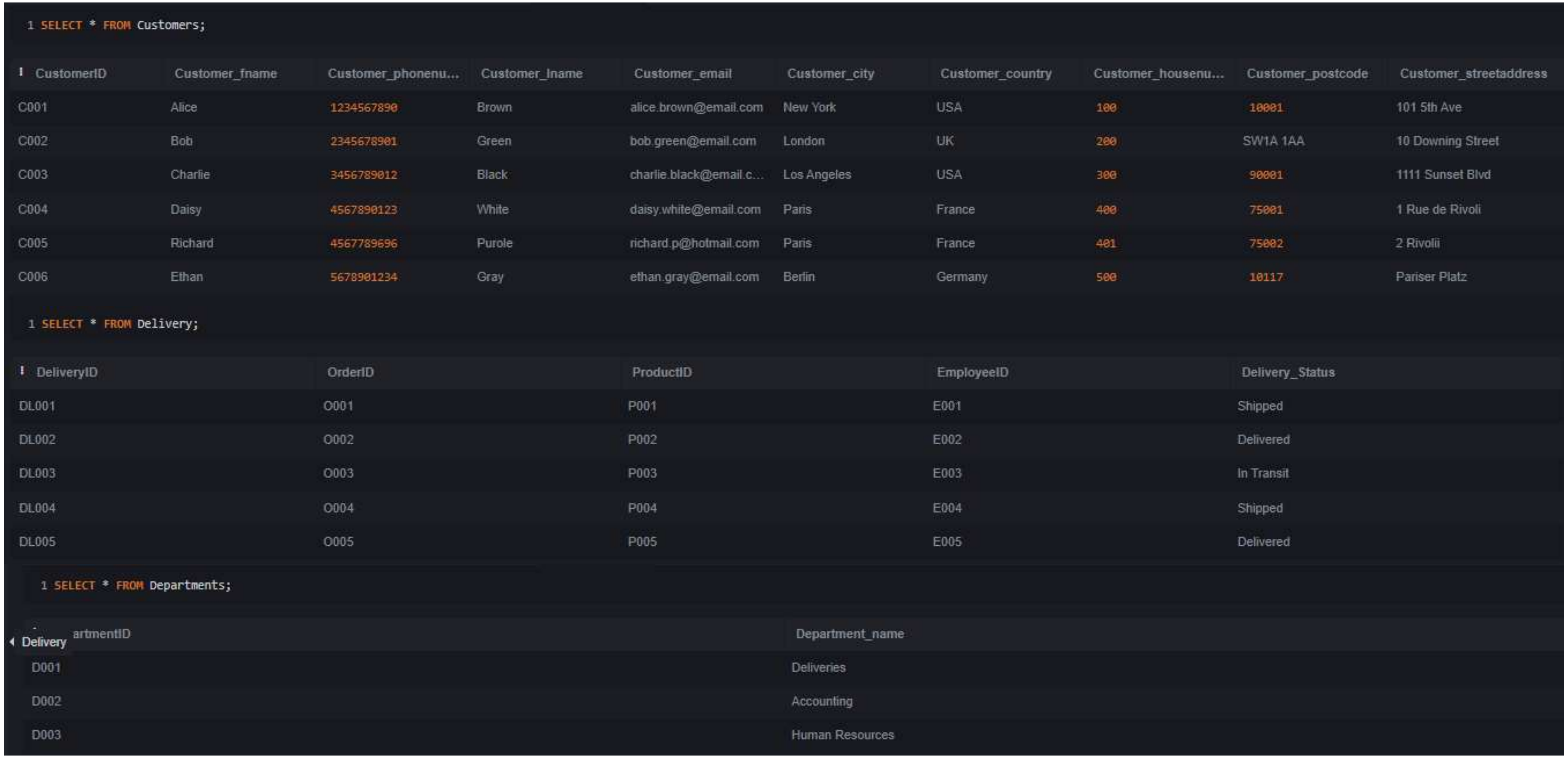
- **Data Types**: Each column in a table has been assigned a specific data Type, which would determine the kind of data (Text, number) that can be stored in that column. For example, the Customer_phonenumber column in the Customers table uses an Integer data type, appropriate for storing phone numbers. Data Types help in optimizing database storage and enhancing query performance. They also play a critical role in validating the data being entered into the database (Harrington, 2016).

3. SQL Scripts to insert records into the database



```sql
INSERT INTO Customers (CustomerID, Customer_fname, Customer_phonenumber, Customer_lname, Customer_email, Customer_city, Customer_country, Customer_housenumber, Customer_postcode, Customer_streetaddress)
VALUES
('C001', 'Alice', 1234567890, 'Brown', 'alice.brown@email.com', 'New York', 'USA', 100, '10001', '101 5th Ave'),
('C002', 'Bob', 2345678901, 'Green', 'bob.green@email.com', 'London', 'UK', 200, 'SW1A 1AA', '10 Downing Street'),
('C003', 'Charlie', 3456789012, 'Black', 'charlie.black@email.com', 'Los Angeles', 'USA', 300, '90001', '1111 Sunset Blvd'),
('C004', 'Daisy', 4567890123, 'White', 'daisy.white@email.com', 'Paris', 'France', 400, '75001', '1 Rue de Rivoli'),
('C005', 'Richard',4567789696, 'Purole','richard.p@hotmail.com','Paris','France',401, '75002','2 Rivolii'),
('C006', 'Ethan', 5678901234, 'Gray', 'ethan.gray@email.com', 'Berlin', 'Germany', 500, '10117', 'Pariser Platz');

INSERT INTO ProductsDetail (ProductID, Product_title, Product_description, Product_unitprice, Product_releasedate, Product_genre, Product_language)
VALUES
('P001', 'The Great Escape', 'Action-packed war movie', 15, '1963-07-04', 'Movie', 'English'),
('P002', 'Thriller', 'Best-selling music album by Michael Jackson', 20, '1982-11-30', 'Music', 'English'),
('P003', '1984', 'The Dystopian novel by George Orwell', 8, '1949-06-08', 'Book', 'English'),
('P004', 'Amelie', 'Charming French romantic comedy', 12, '2001-04-25', 'Movie', 'French'),
('P005', 'Abbey Road', 'Classic album by The Beatles', 22, '1969-09-26', 'Music', 'English');

INSERT INTO PaymentType (Payment_typeID)
VALUES
('Cash'),
('Credit Card');

INSERT INTO Orders (OrderID, CustomerID, ProductID, Order_date, Order_totalamount, Order_quantity, Order_totalquantity, Payment_typeID)
VALUES
('O001', 'C001', 'P001', '2024-01-10', 30, 2, 2, 'Cash'),
('O002', 'C002', 'P002', '2024-01-11', 20, 1, 1, 'Credit Card'),
('O003', 'C003', 'P003', '2024-01-12', 8, 1, 1, 'Cash'),
('O004', 'C004', 'P004', '2024-01-13', 24, 2, 2, 'Credit Card'),
('O005', 'C005', 'P005', '2024-01-14', 44, 2, 2, 'Cash');

INSERT INTO Employee (EmployeeID, DepartmentID, Employee_fname, Employee_lname)
VALUES
('E001', 'D001', 'John', 'Doe'),
('E002', 'D001', 'Sam', 'Smith'),
('E003', 'D002', 'Michael', 'Johnson'),
('E004', 'D002', 'Emily', 'Davis'),
('E005', 'D003', 'David', 'Wilson');

INSERT INTO Departments (DepartmentID, Department_name)
VALUES
('D001', 'Deliveries'),
('D002', 'Accounting'),
('D003', 'Human Resources');

INSERT INTO Product_type (Product_typeID, ProductID, EmployeeID)
VALUES
('PT001', 'P001', 'E001'),
('PT002', 'P002', 'E002'),
('PT003', 'P003', 'E003'),
('PT004', 'P004', 'E004'),
('PT005', 'P005', 'E005');

INSERT INTO Delivery (DeliveryID, OrderID, ProductID, EmployeeID, Delivery_Status)
VALUES
('DL001', 'O001', 'P001', 'E001', 'Shipped'),
('DL002', 'O002', 'P002', 'E002', 'Delivered'),
('DL003', 'O003', 'P003', 'E003', 'In Transit'),
('DL004', 'O004', 'P004', 'E004', 'Shipped'),
('DL005', 'O005', 'P005', 'E005', 'Delivered');
```

```sql
INSERT INTO Customers (CustomerID, Customer_fname, Customer_phonenumber, Customer_lname, Customer_email, Customer_city,
Customer_country, Customer_housenumber, Customer_postcode, Customer_streetaddress)

VALUES

('C001', 'Alice', 1234567890, 'Brown', 'alice.brown@email.com', 'New York', 'USA', 100, '10001', '101 5th Ave'),

('C002', 'Bob', 2345678901, 'Green', 'bob.green@email.com', 'London', 'UK', 200, 'SW1A 1AA', '10 Downing Street'),

('C003', 'Charlie', 3456789012, 'Black', 'charlie.black@email.com', 'Los Angeles', 'USA', 300, '90001', '1111 Sunset Blvd'),

('C004', 'Daisy', 4567890123, 'White', 'daisy.white@email.com', 'Paris', 'France', 400, '75001', '1 Rue de Rivoli'),

('C005', 'Richard',4567789696, 'Purole','richard.p@hotmail.com','Paris','France',401, '75002','2 Rivolii'),

('C006', 'Ethan', 5678901234, 'Gray', 'ethan.gray@email.com', 'Berlin', 'Germany', 500, '10117', 'Pariser Platz');

INSERT INTO ProductsDetail (ProductID, Product_title, Product_description, Product_unitprice, Product_releasedate, Product_genre, Product_language)

VALUES

('P001', 'The Great Escape', 'Action-packed war movie', 15, '1963-07-04', 'Movie', 'English'),

('P002', 'Thriller', 'Best-selling music album by Michael Jackson', 20, '1982-11-30', 'Music', 'English'),

('P003', '1984', 'The Dystopian novel by George Orwell', 8, '1949-06-08', 'Book', 'English'),

('P004', 'Amelie', 'Charming French romantic comedy', 12, '2001-04-25', 'Movie', 'French'),

('P005', 'Abbey Road', 'Classic album by The Beatles', 22, '1969-09-26', 'Music', 'English');

INSERT INTO PaymentType (Payment_typeID)

VALUES

('Cash'),

('Credit Card');
```



```sql
SELECT * FROM Customers;
```

CustomerID	Customer_fname	Customer_phonenu...	Customer_lname	Customer_email	Customer_city	Customer_country	Customer_housenu...	Customer_postcode	Customer_streetaddress
C001	Alice	1234567890	Brown	alice.brown@email.com	New York	USA	100	10001	101 5th Ave
C002	Bob	2345678901	Green	bob.green@email.com	London	UK	200	SW1A 1AA	10 Downing Street
C003	Charlie	3456789012	Black	charlie.black@email.c...	Los Angeles	USA	300	90001	1111 Sunset Blvd
C004	Daisy	4567890123	White	daisy.white@email.com	Paris	France	400	75001	1 Rue de Rivoli
C005	Richard	4567789696	Purole	richard.p@hotmail.com	Paris	France	401	75002	2 Rivolii
C006	Ethan	5678901234	Gray	ethan.gray@email.com	Berlin	Germany	500	10117	Pariser Platz

```sql
SELECT * FROM Delivery;
```

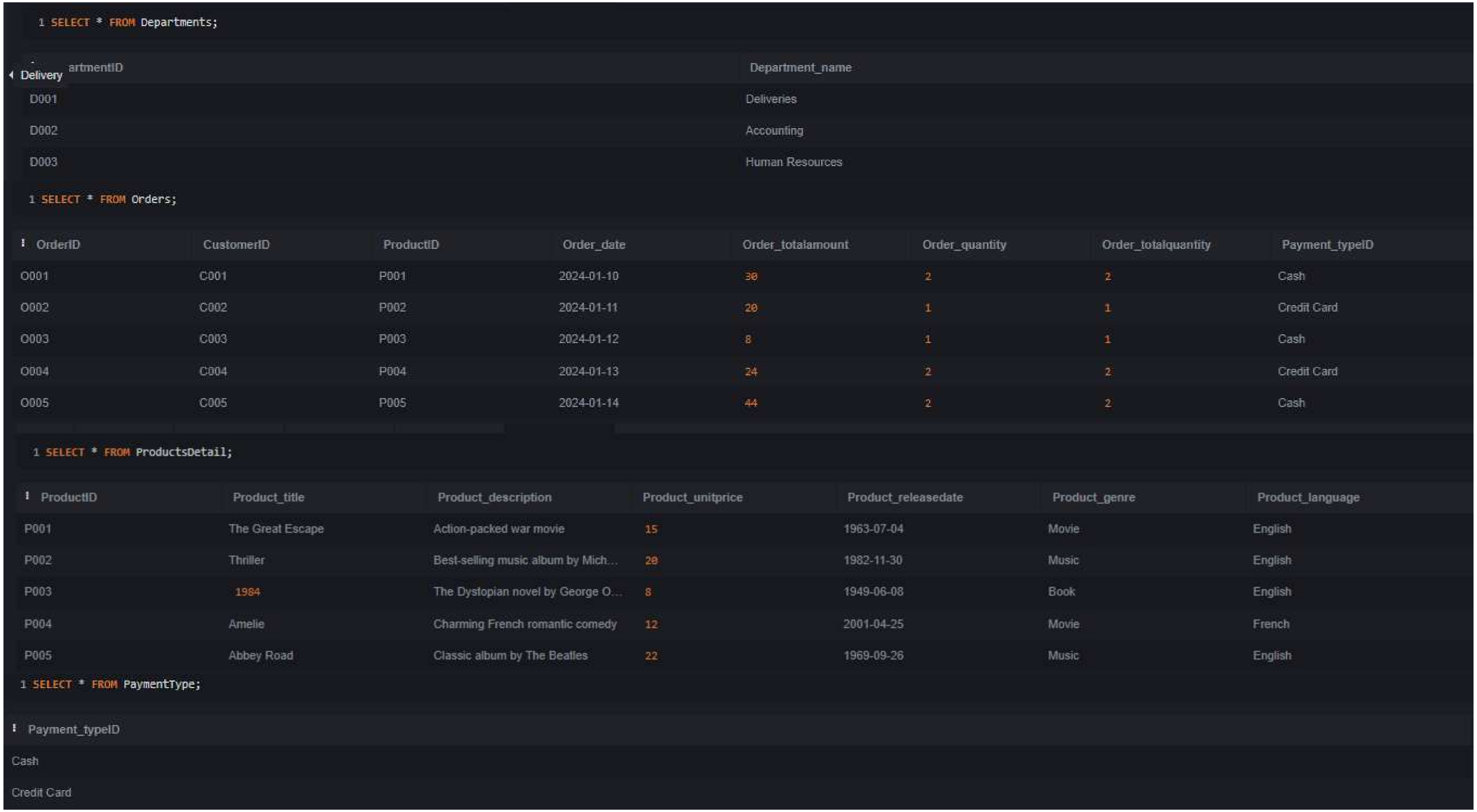
DeliveryID	OrderID	ProductID	EmployeeID	Delivery_Status
DL001	O001	P001	E001	Shipped
DL002	O002	P002	E002	Delivered
DL003	O003	P003	E003	In Transit
DL004	O004	P004	E004	Shipped
DL005	O005	P005	E005	Delivered

```sql
SELECT * FROM Departments;
```

artmentID	Department_name
D001	Deliveries
D002	Accounting
D003	Human Resources

```
INSERT INTO Orders (OrderID, CustomerID, ProductID, Order_date,
Order_totalamount, Order_quantity, Order_totalquantity,
Payment_typeID)

VALUES

('O001', 'C001', 'P001', '2024-01-10', 30, 2, 2, 'Cash'),

('O002', 'C002', 'P002', '2024-01-11', 20, 1, 1, 'Credit Card'),

('O003', 'C003', 'P003', '2024-01-12', 8, 1, 1, 'Cash'),

('O004', 'C004', 'P004', '2024-01-13', 24, 2, 2, 'Credit Card'),

('O005', 'C005', 'P005', '2024-01-14', 44, 2, 2, 'Cash');

INSERT INTO Employee (EmployeeID, DepartmentID,
Employee_fname, Employee_lname)

VALUES

('E001', 'D001', 'John', 'Doe'),

('E002', 'D001', 'Sam', 'Smith'),

('E003', 'D002', 'Michael', 'Johnson'),

('E004', 'D002', 'Emily', 'Davis'),

('E005', 'D003', 'David', 'Wilson');

INSERT INTO Departments (DepartmentID, Department_name)

VALUES

('D001', 'Deliveries'),

('D002', 'Accounting'),

('D003', 'Human Resources');
```

```
INSERT INTO Product_type (Product_typeID, ProductID, EmployeeID)

VALUES

('PT001', 'P001', 'E001'),

('PT002', 'P002', 'E002'),

('PT003', 'P003', 'E003'),

('PT004', 'P004', 'E004'),

('PT005', 'P005', 'E005');


INSERT INTO Delivery (DeliveryID, OrderID, ProductID, EmployeeID,
Delivery_Status)

VALUES

('DL001', 'O001', 'P001', 'E001', 'Shipped'),

('DL002', 'O002', 'P002', 'E002', 'Delivered'),

('DL003', 'O003', 'P003', 'E003', 'In Transit'),

('DL004', 'O004', 'P004', 'E004', 'Shipped'),

('DL005', 'O005', 'P005', 'E005', 'Delivered');
```



```sql
SELECT * FROM Departments;
```

Delivery

artmentID	Department_name
D001	Deliveries
D002	Accounting
D003	Human Resources

```sql
SELECT * FROM Orders;
```

OrderID	CustomerID	ProductID	Order_date	Order_totalamount	Order_quantity	Order_totalquantity	Payment_typeID
O001	C001	P001	2024-01-10	30	2	2	Cash
O002	C002	P002	2024-01-11	20	1	1	Credit Card
O003	C003	P003	2024-01-12	8	1	1	Cash
O004	C004	P004	2024-01-13	24	2	2	Credit Card
O005	C005	P005	2024-01-14	44	2	2	Cash

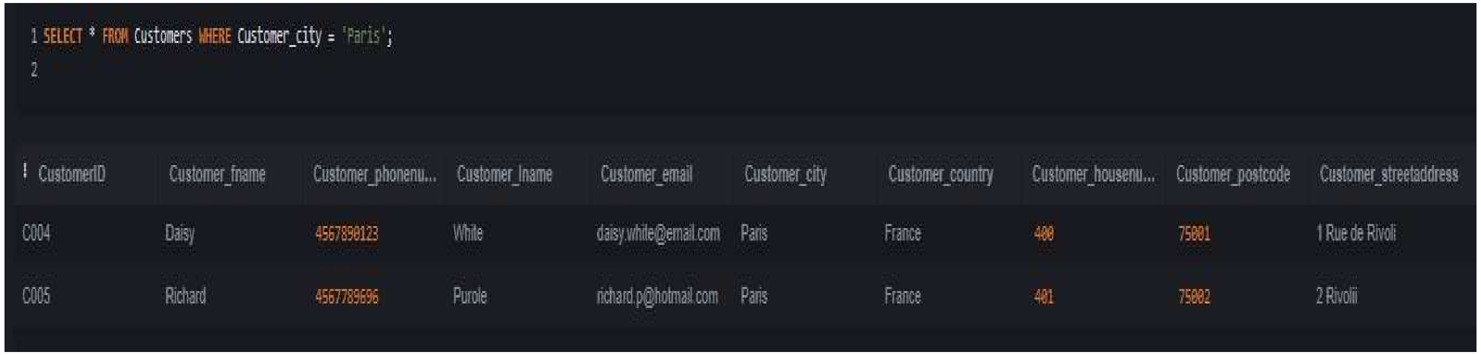
```sql
SELECT * FROM ProductsDetail;
```

ProductID	Product_title	Product_description	Product_unitprice	Product_releasedate	Product_genre	Product_language
P001	The Great Escape	Action-packed war movie	15	1963-07-04	Movie	English
P002	Thriller	Best-selling music album by Mich…	20	1982-11-30	Music	English
P003	1984	The Dystopian novel by George O…	8	1949-06-08	Book	English
P004	Amelie	Charming French romantic comedy	12	2001-04-25	Movie	French
P005	Abbey Road	Classic album by The Beatles	22	1969-09-26	Music	English

```sql
SELECT * FROM PaymentType;
```

Payment_typeID
Cash
Credit Card

4a. Extract Customers from a Specific City:



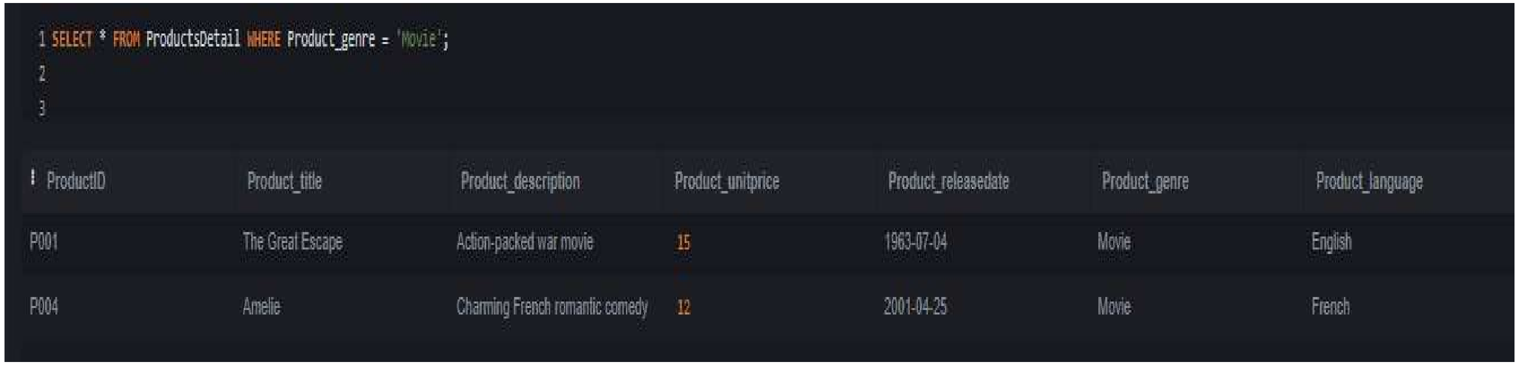
```sql
SELECT * FROM Customers WHERE Customer_city = 'Paris';
```

CustomerID	Customer_fname	Customer_phonenu...	Customer_lname	Customer_email	Customer_city	Customer_country	Customer_housenu...	Customer_postcode	Customer_streetaddress
C004	Daisy	4567890123	White	daisy.white@email.com	Paris	France	400	75001	1 Rue de Rivoli
C005	Richard	4567789696	Purole	richard.p@hotmail.com	Paris	France	401	75002	2 Rivolii

SELECT * FROM Customers WHERE Customer_city = ‘Paris';

This query retrieves all records from the Customers table where the Customer_city column matches a specified city name (Paris). It's useful for localized customer analysis.

4b. Search for a product of a specific genre:



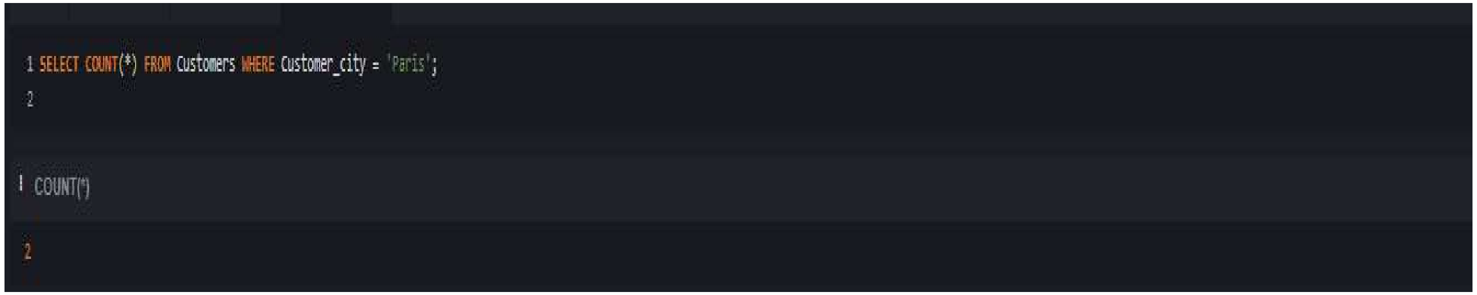
```
SELECT * FROM ProductsDetail WHERE Product_genre = 'Movie';
```

ProductID	Product_title	Product_description	Product_unitprice	Product_releasedate	Product_genre	Product_language
P001	The Great Escape	Action-packed war movie	15	1963-07-04	Movie	English
P004	Amelie	Charming French romantic comedy	12	2001-04-25	Movie	French

SELECT * FROM ProductsDetail WHERE Product_genre = ‘Movie’;

This query selects all products from the ProductsDetail table that belong to a specified genre which was movies. It's helpful for inventory checks or marketing analysis within a certain product category.

4c. Count Customers from a Specific City:



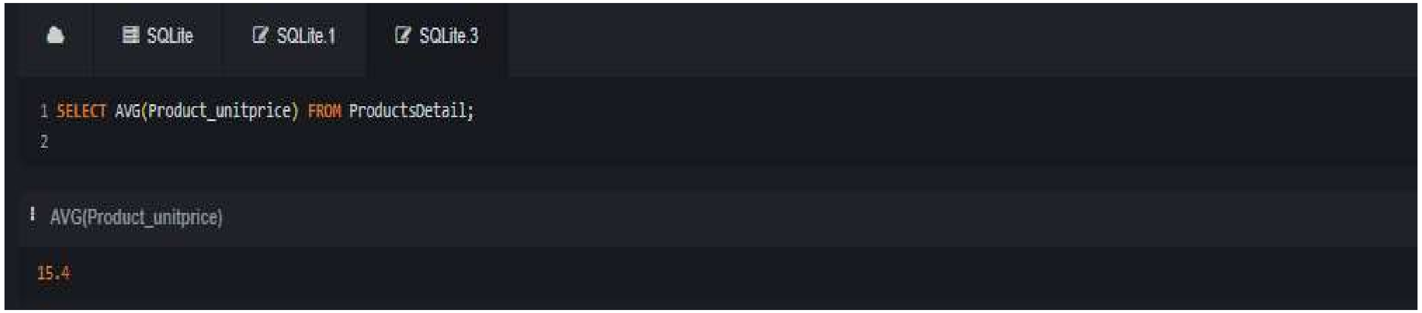
```
SELECT COUNT(*) FROM Customers WHERE Customer_city = 'Paris';
```

```
COUNT(*)
2
```

SELECT COUNT(*) FROM Customers WHERE Customer_city = ‘Paris';

This SQL statement counts the number of customers residing in a specific city, Paris. It's useful for understanding the customer distribution geographically.

4d. Calculate Average Unit Price:



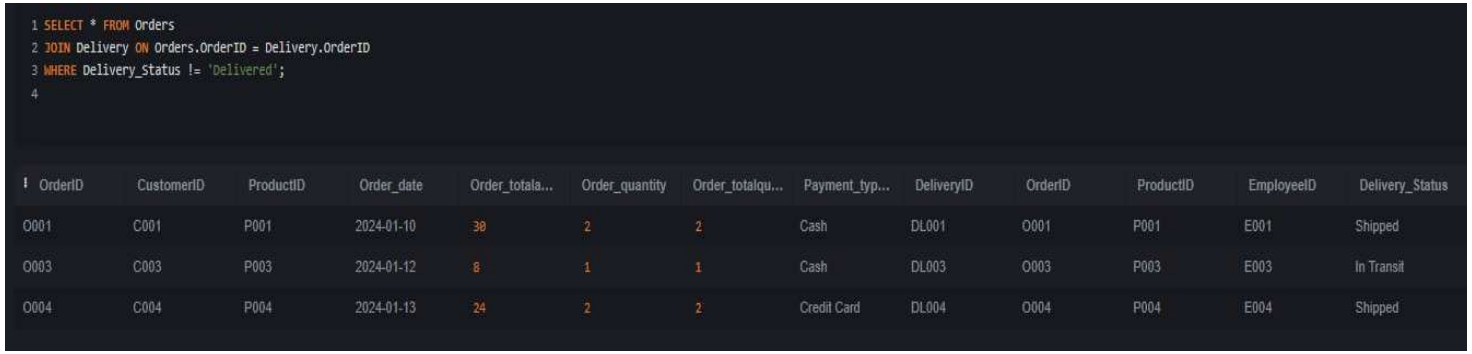
```
SELECT AVG(Product_unitprice) FROM ProductsDetail;
```

```
AVG(Product_unitprice)
15.4
```

SELECT AVG(Product_unitprice) FROM ProductsDetail;

This query calculates the average price of products from the ProductsDetail table. This can provide insights into the general pricing strategy of the e-shop.

4e. Extract all current orders:



```
SELECT * FROM Orders
JOIN Delivery ON Orders.OrderID = Delivery.OrderID
WHERE Delivery_Status != 'Delivered';
```

OrderID	CustomerID	ProductID	Order_date	Order_totala...	Order_quantity	Order_totalqu...	Payment_typ...	DeliveryID	OrderID	ProductID	EmployeeID	Delivery_Status
O001	C001	P001	2024-01-10	30	2	2	Cash	DL001	O001	P001	E001	Shipped
O003	C003	P003	2024-01-12	8	1	1	Cash	DL003	O003	P003	E003	In Transit
O004	C004	P004	2024-01-13	24	2	2	Credit Card	DL004	O004	P004	E004	Shipped

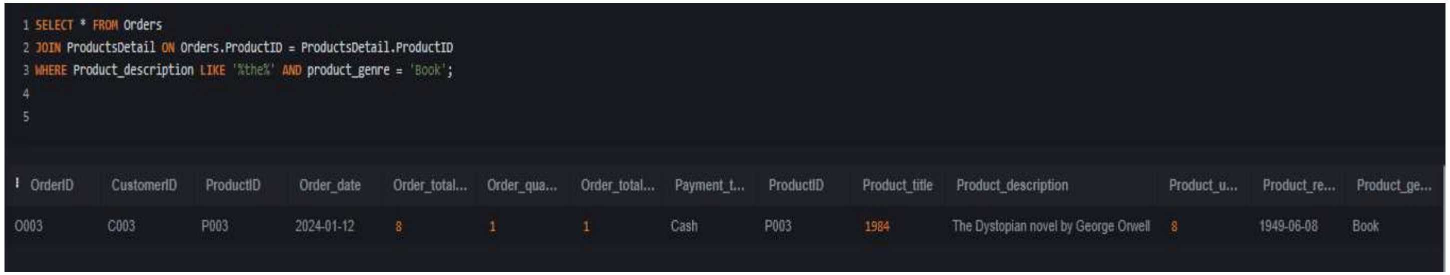
SELECT * FROM Orders

JOIN Delivery ON Orders.OrderID = Delivery.OrderID

WHERE Delivery_Status != 'Delivered';

This query selects all orders that are yet to be delivered. It joins Orders and Delivery tables and filters by the delivery status. It's crucial for managing and tracking ongoing orders.

4f. Orders for books with the word “the” in their description:



```
SELECT * FROM Orders
JOIN ProductsDetail ON Orders.ProductID = ProductsDetail.ProductID
WHERE Product_description LIKE '%the%' AND product_genre = 'Book';


```

OrderID	CustomerID	ProductID	Order_date	Order_total...	Order_qua...	Order_total...	Payment_t...	ProductID	Product_title	Product_description	Product_u...	Product_re...	Product_ge...
O003	C003	P003	2024-01-12	8	1	1	Cash	P003	1984	The Dystopian novel by George Orwell	8	1949-06-08	Book

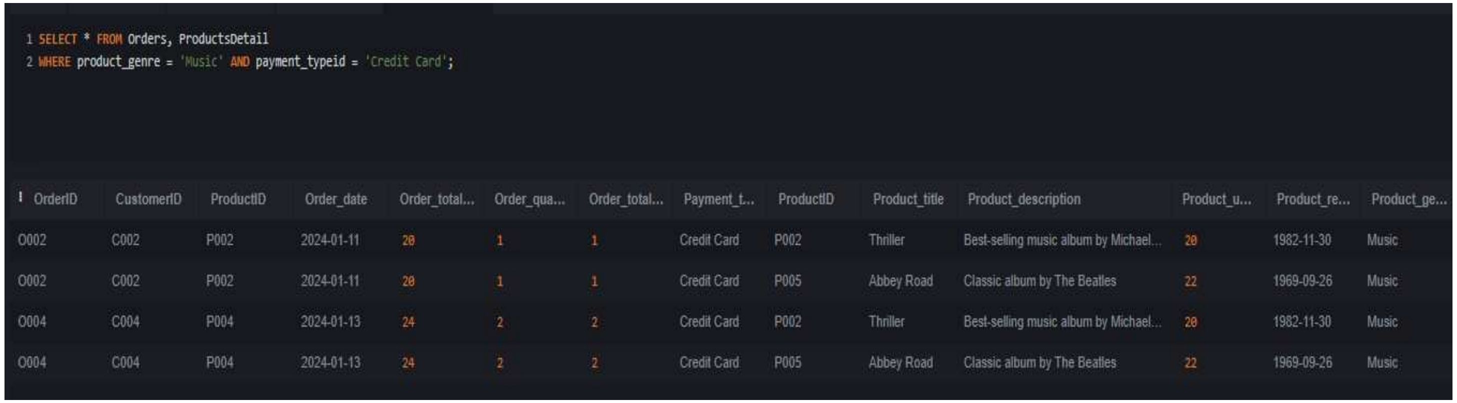
SELECT Orders.* FROM Orders

JOIN ProductsDetail ON Orders.ProductID = ProductsDetail.ProductID

WHERE Product_description LIKE '%the%' AND Product_type = 'Book';

This query finds all orders for books where the book description contains the keyword "the." It joins Orders with ProductsDetail and filters by product description and type. It's useful for targeted marketing or stock analysis.

4g. Extract Payments with Credit Cards for Music Records:



```
SELECT * FROM Orders, ProductsDetail
WHERE product_genre = 'Music' AND payment_typeid = 'Credit Card';
```

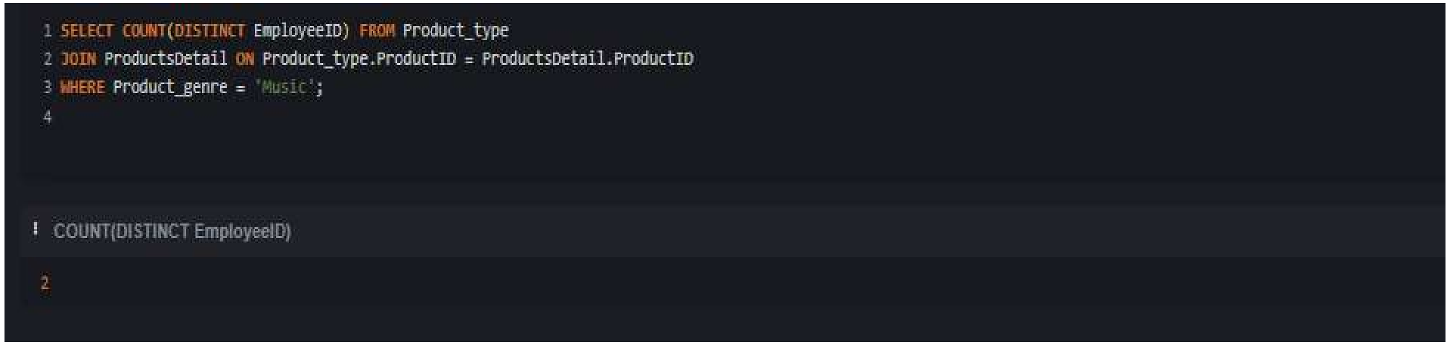
OrderID	CustomerID	ProductID	Order_date	Order_total...	Order_qua...	Order_total...	Payment_t...	ProductID	Product_title	Product_description	Product_u...	Product_re...	Product_ge...
O002	C002	P002	2024-01-11	20	1	1	Credit Card	P002	Thriller	Best-selling music album by Michael...	20	1982-11-30	Music
O002	C002	P002	2024-01-11	20	1	1	Credit Card	P005	Abbey Road	Classic album by The Beatles	22	1969-09-26	Music
O004	C004	P004	2024-01-13	24	2	2	Credit Card	P002	Thriller	Best-selling music album by Michael...	20	1982-11-30	Music
O004	C004	P004	2024-01-13	24	2	2	Credit Card	P005	Abbey Road	Classic album by The Beatles	22	1969-09-26	Music

SELECT * FROM Orders, ProductsDetail

WHERE product_genre = 'Music' AND payment_typeid = 'Credit Card';

This query combines Order and ProductDetials table, and filter with "Music" and "Credit cards. This can be useful in understanding customer's payment preference when buying a particular item.

4h. Count Employees Handling Music Records:



```
SELECT COUNT(DISTINCT EmployeeID) FROM Product_type
JOIN ProductsDetail ON Product_type.ProductID = ProductsDetail.ProductID
WHERE Product_genre = 'Music';
```

```
COUNT(DISTINCT EmployeeID)
2
```

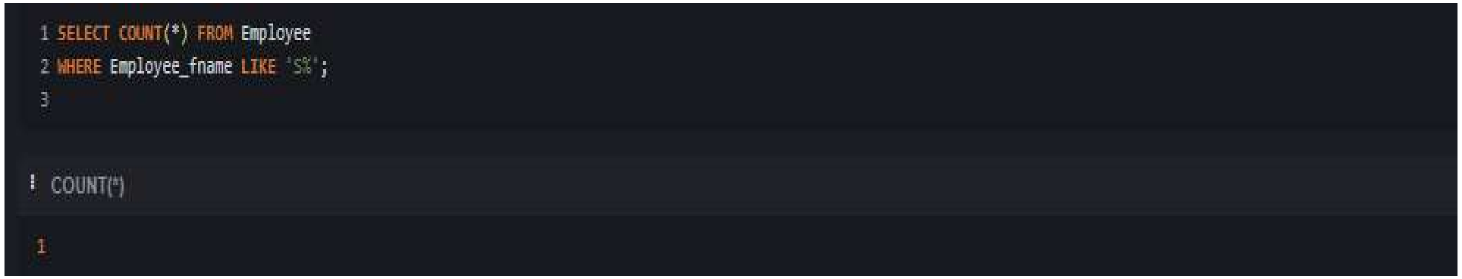
SELECT COUNT(DISTINCT EmployeeID) FROM Product_type

JOIN ProductsDetail ON Product_type.ProductID = ProductsDetail.ProductID

WHERE Product_genre = 'Music';

This query counts the number of unique employees involved with music records by joining the Product_type and ProductsDetail tables. It's useful for workforce planning in specific product categories.

4i. Employees with the first name starting with the letter 'S':



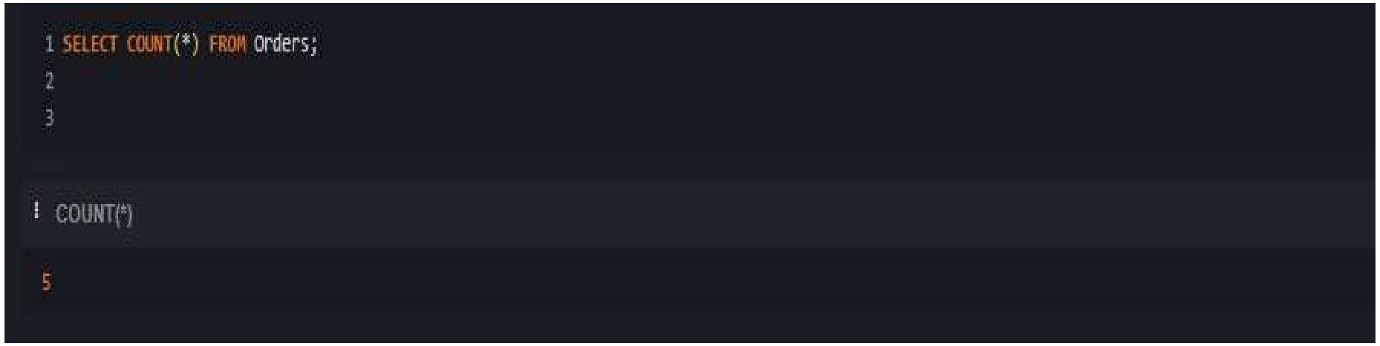
```
SELECT COUNT(*) FROM Employee
WHERE Employee_fname LIKE 'S%';
```

```
COUNT(*)
1
```

SELECT COUNT(*) FROM Employee WHERE Employee_fname LIKE 'S%';

This statement counts all employees whose first names start with 'S' in the Employee table. It's a simple query useful for specific personnel-related inquiries or analyses.

4j. Count how many orders are in the system:



```
SELECT COUNT(*) FROM Orders;
```

```
COUNT(*)
5
```

SELECT COUNT(*) FROM Orders;

This straightforward query counts the total number of orders in the Orders table. It provides a quick overview of the total business volume in terms of orders.

Explaining the functions used in the SQL Scripts above

According to W3schools (2020) each function and clause plays a crucial role in querying and managing the data in SQL databases. They are fundamental tools for extracting, analyzing, and reporting data stored in relational databases just like the E-shop.

- SELECT: Retrieves data from one or more tables. You specify the columns you want.
- FROM: Specifies the table(s) from which to retrieve data. Used in conjunction with SELECT.
- WHERE: Filters the results to only include rows that meet the specified condition.
- COUNT(): An aggregate function that returns the number of rows that match a specified criterion.
- AVG(): Another aggregate function; it calculates the average value of a specified column.
- JOIN: This is used to combine rows from two or more tables based. (e.g., INNER JOIN, LEFT JOIN).
- ON: Used with JOIN, it specifies the matching condition for joining tables.
- LIKE: Used in WHERE clause for pattern matching. '%' is a wildcard character representing zero, one, or multiple characters.
- DISTINCT: Used to remove duplicate values from the result set.

Use Case

Management: The use case for management would be initiated by product managers, procurement managers, suppliers and vendors activity related users. This ER database would ensure customers have access to the latest available items and helps the e-shop manage inventory effectively, reducing overstock or stockout situations as determined by Ordonez et al. (2014). Also providing analysis, This includes not just the basic information like title and price but also stock levels, supplier details, product categories (such as music records, movies, books), and manage information about the products details like title, description, price, genre, and release date on which product the store would sell.

Order Processing: The use case for order processing would be their customers. This database would facilitate the creation and tracking of customer orders, managing details such as order date, total amount, quantity, and payment information. As well as every aspect of an order, from the moment it's placed to the final delivery, including payment processing, order status updates (like processing, shipped, delivered), and any returns or cancellations. Efficient order processing enhances customer satisfaction by ensuring timely and accurate delivery. It also streamlines operations, reducing errors and delays, and strengthen management decision making. (Lu, Liu and Xu, 2021)

Customer : This use case is targeted at customers for the purpose of relationship management. This will maintain a database of customer information, including names, contact details, and address information. Beyond basic information, this can include purchase history, customer preferences, loyalty program details, and customer feedback or reviews. The database would assist management strategy and marketing department in understanding customers better, which allows for personalized marketing and improved customer service according to research (Webber, 2013). It also helps in building long-term customer relationships. (Elaheh Taghavi Shavazi, 2013)

Use Case

Employees : This use can will be initiated by the human resources team, which will include employee roles and responsibilities, performance tracking, shift scheduling, and departmental alignments. This would enable the E-shop to ensures that the right personnel is available, and is available at the right time and place, has adequate skills needed to improving overall operational efficiency selling any product or performing other activities (Budhwar et al., 2023).

Delivery Tracking: This use case is for gathering real-time tracking information and delivery personnel details, a database able to gather these information would provide transparency and trust for customers, ensuring they are informed about their order status. It also helps in optimizing delivery routes for efficiency as determined by Salah et al. (2020)

Data Analytics: Nowadays most organization are leveraging data stored in the database for business intelligence. Analyzing sales trends by Sales Managers, customer behavior, and operational efficiency by the analytical team. These can lead to informed decision-making and strategy development across departments and the organization. (Trujillo et al., 2021)

Integration with Other Systems: This use case would be required by the information technology team, software developers and other whom require system integration. This e-shop's ER database would enable the users to integrate with external systems such as payment gateways, CRM software, or logistics tracking systems for a seamless operation (Li et al., 2020).

Conclusion

In conclusion, the database for any organization such as the e-shop case discussed in the portfolio is not just a repository of information but an essential dynamic tool that drives the business's operational efficiency, informs strategic decisions, enhances customer experience, and ensures regulatory compliance and security. Stakeholders from customers and employees to administrators and suppliers all interact with the database, making its design and management crucial step to the online business to be successful.

Reflection

Working on the e-shop's database system has been an eye-opening experience. It highlighted the crucial role of technology in connecting with customers and managing commerce efficiently. Beyond just scripts writing using SQL and data management, this portfolio underlined the importance of understanding customer needs and how every technical aspect such as the ER Diagram, ties back to enhancing their experience. It also reinforced the need for adaptability in technology, preparing for future trends and evolving customer expectations. Overall, this journey has been a blend of technological innovation and a commitment to better service, showcasing how technology is vital in driving a business forward while keeping it closely connected to its customer base.

References

- Budhwar, P., Chowdhury, S., Wood, G., Aguinis, H., Bamber, G.J., Beltran, J.R., Boselie, P., Fang Lee Cooke, Decker, S., DeNisi, A., Prasanta Kumar Dey, Guest, D., Knoblich, A.J., Malik, A., Paauwe, J., Savvas Papagiannidis, Patel, C., Pereira, V., Ren, S. and Rogelberg, S. (2023). Human resource management in the age of generative artificial intelligence. *Human resource management in the age of generative artificial intelligence:* , [online] 33(3). doi:https://doi.org/10.1111/1748-8583.12524.

- Connolly, T.M. and Begg, C.E. (2015). *Database systems : a practical approach to design, implementation, and management*. 6th ed. Harlow, Essex, England: Pearson Education Limited.

- Date, C.J. (2019). *Database design and relational theory : normal forms and all that jazz*. Berkeley, California? Apress.

- Elaheh Taghavi Shavazi, E.T.S. (2013). Customer Relationship Management And Organizational Performance: A Conceptual Framework Based On The Balanced Scorecard (Study Of Iranian Banks). *IOSR Journal of Business and Management*, 10(6), pp.18–26. doi:https://doi.org/10.9790/487x-1061826.

- Harrington, J.L. (2016). *Relational Database Design and Implementation Clearly Explained*. San Francisco Elsevier Science & Technology Ann Arbor, Michigan Proquest.

- Li, B.-H., Liu, Y., Zhang, A.-M., Wang, W.-H. and Wan, S. (2020). A Survey on Blocking Technology of Entity Resolution. *Journal of Computer Science and Technology*, 35(4), pp.769–793. doi:https://doi.org/10.1007/s11390-020-0350-4.

- Lu, P., Liu, P. and Xu, J. (2021). Design of Intelligent Warehouse Management System Based on MVC. *International Journal of Advanced Network, Monitoring and Controls*, 6(2), pp.79–87. doi:https://doi.org/10.21307/ijanmc-2021-020.