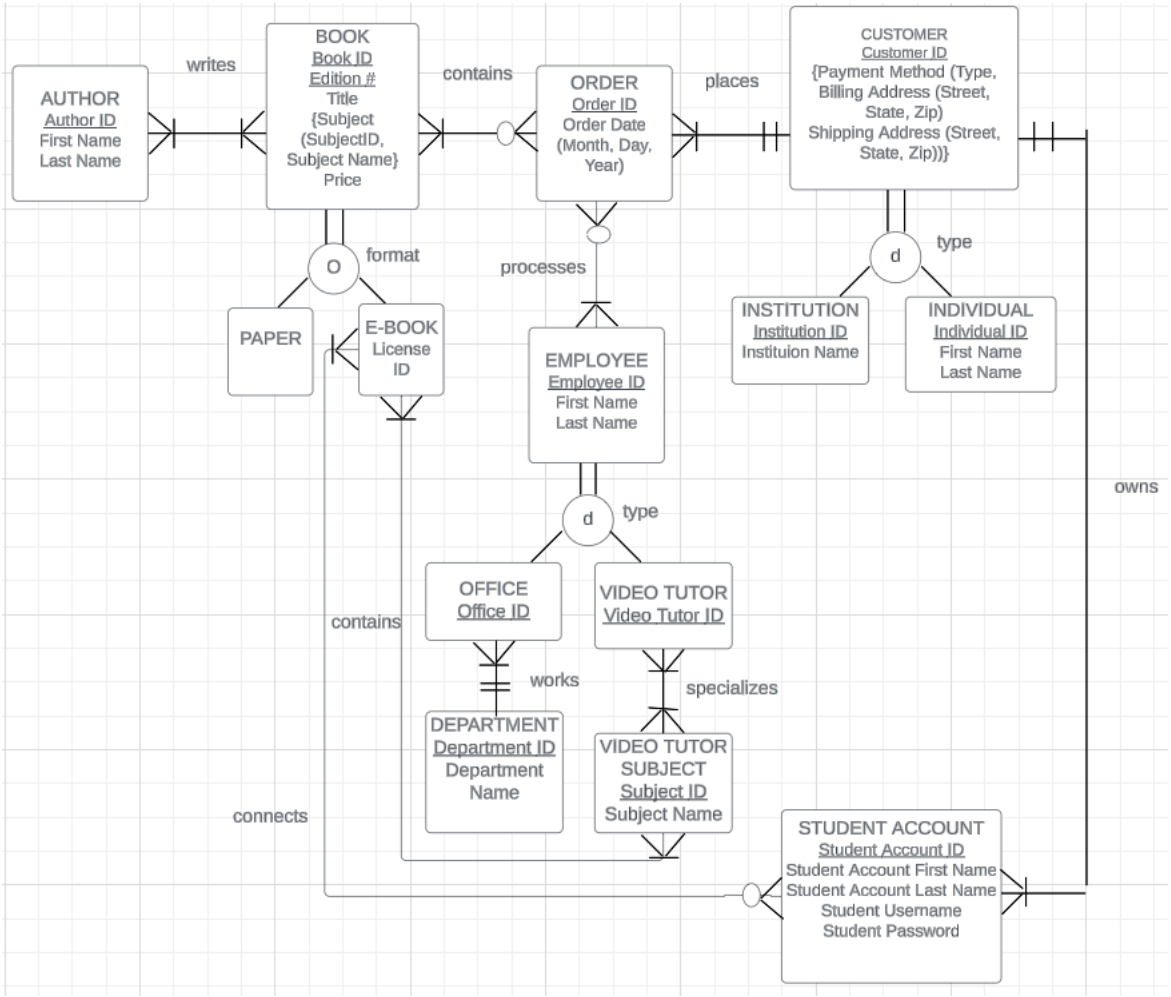
Hanna Wierszok
Kevin Nagpal
Ruojia Kuang
Ryan Perez
Sean Patrick King

Project 1B

Original E-R Diagram



Normalization

In this E-R diagram, the subject attribute was moved under the book entity. The subject is dependent on the book entity. We created a separate table for the subject, with a foreign key referring to the primary key of the book table. The payment method attribute was also moved under the customer table to reduce redundancy and to ensure the data integrity.

Create Database

```
DROP DATABASE IF EXISTS Textbooks;
CREATE DATABASE Textbooks;
USE Textbooks;

# AUTHOR Table
CREATE TABLE Author (
        AuthorID INT,
        AuthorFirstName VARCHAR(50),
        AuthorLastName VARCHAR(50),

        PRIMARY KEY (AuthorID)
);

# BOOK Table
CREATE TABLE Book (
        BookID INT,
        EditionNumber INT,
        Title VARCHAR(100),
  Price INT,

        PRIMARY KEY (BookID, EditionNumber)
);

# Subject Table
CREATE TABLE `Subject` (
  SubjectID INT PRIMARY KEY,
  SubjectName VARCHAR(50) UNIQUE
);

# Book Subjects Junction Table
CREATE TABLE BookSubject (
  BookID INT,
```

```
        EditionNumber INT,
    SubjectID INT,
    PRIMARY KEY (BookID, EditionNumber, SubjectID),
        FOREIGN KEY (BookID, EditionNumber) REFERENCES Book(BookID, EditionNumber),
    FOREIGN KEY (SubjectID) REFERENCES `Subject`(SubjectID)
);

-- Subtype table for PAPER books
CREATE TABLE Paper (
    BookID INT,
    EditionNumber INT,

        PRIMARY KEY (BookID, EditionNumber),

    FOREIGN KEY (BookID, EditionNumber) REFERENCES Book(BookID, EditionNumber)
);

-- Subtype table for EBOOK books
CREATE TABLE Ebook (
    BookID INT,
    EditionNumber INT,
    -- Additional attributes specific to Ebooks
    LicenseID INT,

    PRIMARY KEY (BookID, EditionNumber),
    FOREIGN KEY (BookID, EditionNumber) REFERENCES Book(BookID, EditionNumber)

);

# CUSTOMER table
CREATE TABLE Customer (
        CustomerID INT PRIMARY KEY
);

# PaymentMethod Table
CREATE TABLE PaymentMethod (
        PaymentID INT PRIMARY KEY,
    PaymentType VARCHAR(50),
    BillingStreet VARCHAR(100), -- Billing address: Street
    BillingState VARCHAR(50),   -- Billing address: State
    BillingZip VARCHAR(20),     -- Billing address: Zip code
    ShippingStreet VARCHAR(100), -- Shipping address: Street
    ShippingState VARCHAR(50),   -- Shipping address: State
    ShippingZip VARCHAR(20)      -- Shipping address: Zip code
```

```
);

# PaymentMethod Customer Junction Table
CREATE TABLE CustomerPaymentMethod (
   CustomerID INT,
         PaymentID INT,
   PRIMARY KEY (CustomerID, PaymentID),

   FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID),
   FOREIGN KEY (PaymentID) REFERENCES PaymentMethod(PaymentID)
);


-- Subtype table for INSTITUTION customer
CREATE TABLE Institution (
   CustomerID INT PRIMARY KEY,
   -- Additional attributes specific to mammals
   InstitutionID INT,
   InstitutionName VARCHAR(100),

   FOREIGN KEY (CustomerID) REFERENCES Customer (CustomerID)
);

-- Subtype table for INDIVIDUAL customer
CREATE TABLE Individual (
   CustomerID INT PRIMARY KEY,
   -- Additional attributes specific to mammals
   IndividualID INT,
   IndividualFirstName VARCHAR(100),
   IndividualLastName VARCHAR(100),

   FOREIGN KEY (CustomerID) REFERENCES Customer (CustomerID)
);

# ORDER Table
CREATE TABLE `Order` (
   OrderID INT PRIMARY KEY,
         OrderDate DATE,

   CustomerID INT, -- b/c 1:M relationship w Customer
   FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)
);
```

```
# STUDENT ACCOUNT table
CREATE TABLE StudentAccount (
   StudentAccountNumber INT PRIMARY KEY,
   StudentAccountFirstName VARCHAR(50),
   StudentAccountLastName VARCHAR(50),
   StudentAccountUsername VARCHAR(50),
   StudentAccountPassword VARCHAR(50),

   -- b/c 1:M for StudentAccount w/ Customer
   CustomerID INT,
   FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)
);



# EMPLOYEE table
CREATE TABLE Employee (
         EmployeeID INT PRIMARY KEY,
         EmployeeFirstName VARCHAR(100),
         EmployeeLastName VARCHAR(100),
   AnnualSalary INT
);

-- Subtype table for OFFICE employee
CREATE TABLE Office (
   EmployeeID INT PRIMARY KEY,
   -- Additional attributes specific to Office employees
   OfficeID INT,

   FOREIGN KEY (EmployeeID) REFERENCES Employee (EmployeeID)
);


-- Subtype table for VIDEO TUTOR employee
CREATE TABLE VideoTutor (
   EmployeeID INT PRIMARY KEY,
   -- Additional attributes specific to Video Tutors
   VideoTutorID INT,

   FOREIGN KEY (EmployeeID) REFERENCES Employee (EmployeeID)
);

# DEPARTMENT table
CREATE TABLE Department (
```

```
	DepartmentID INT PRIMARY KEY,
	DepartmentName VARCHAR(100)
);


# VIDEO TUTOR SUBJECT table
CREATE TABLE VideoTutorSubject (
	VideoTutorSubjectID INT PRIMARY KEY,
	VideoTutorSubjectName VARCHAR(100)
);


-- Junction table for the many-to-many relationship b/w Book and Author
CREATE TABLE AuthorBook (
   AuthorID INT,
   BookID INT,
   EditionNumber INT,
   PRIMARY KEY (AuthorID, BookID, EditionNumber),
   FOREIGN KEY (AuthorID) REFERENCES Author(AuthorID),
   FOREIGN KEY (BookID, EditionNumber) REFERENCES Book(BookID, EditionNumber)
);

-- JUNCTION TABLE for the many-to-many relationship b/w Book and Order
CREATE TABLE BookOrder (
   BookID INT,
   EditionNumber INT,
   OrderID INT,
   Quantity INT,
   PRIMARY KEY (BookID, EditionNumber, OrderID),
   FOREIGN KEY (BookID, EditionNumber) REFERENCES Book(BookID, EditionNumber),
   FOREIGN KEY (OrderID) REFERENCES `Order`(OrderID)
);


-- JUNCTION TABLE for the many-to-many relationship b/w Order and Employee
CREATE TABLE OrderEmployee (
   OrderID INT,
   EmployeeID INT,
   PRIMARY KEY (OrderID, EmployeeID),
   FOREIGN KEY (OrderID) REFERENCES `Order`(OrderID),
   FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID)
);
```

```
-- JUNCTION TABLE for the many-to-many relationship b/w VideoTutor and VideoTutorSubject
CREATE TABLE VideoTutorVideoTutorSubject (
   EmployeeID INT,
   VideoTutorSubjectID INT,
   PRIMARY KEY (EmployeeID, VideoTutorSubjectID),
   FOREIGN KEY (EmployeeID) REFERENCES VideoTutor (EmployeeID),
   FOREIGN KEY (VideoTutorSubjectID) REFERENCES VideoTutorSubject (VideoTutorSubjectID)
);


-- Junction table for the many-to-many relationship b/w Office employees and Department
CREATE TABLE OfficeDepartment (
   EmployeeID INT,
   DepartmentID INT,
   PRIMARY KEY (EmployeeID, DepartmentID),
   FOREIGN KEY (EmployeeID) REFERENCES Office (EmployeeID),
   FOREIGN KEY (DepartmentID) REFERENCES Department (DepartmentID)
);

-- Junction table for the many-to-many relationship b/w VideoTutorSubject and Ebook
CREATE TABLE EbookVideoTutorSubject (
   BookID INT,
   EditionNumber INT,
   VideoTutorSubjectID INT,
   PRIMARY KEY (BookID, EditionNumber, VideoTutorSubjectID),
   FOREIGN KEY (BookID, EditionNumber) REFERENCES Ebook(BookID, EditionNumber),
   FOREIGN KEY (VideoTutorSubjectID) REFERENCES VideoTutorSubject(VideoTutorSubjectID)
);

-- JUNCTION TABLE for the many-to-many relationship b/w StudentAccount and Ebook
CREATE TABLE StudentEbook (
   StudentAccountNumber INT,
   BookID INT,
   EditionNumber INT,
   PRIMARY KEY (StudentAccountNumber, BookID, EditionNumber),
   FOREIGN KEY (StudentAccountNumber) REFERENCES
StudentAccount(StudentAccountNumber),
   FOREIGN KEY (BookID, EditionNumber) REFERENCES Ebook(BookID, EditionNumber)
);
```

Populate

```
-- Inserting sample data into AUTHOR table
INSERT INTO Author (AuthorID, AuthorFirstName, AuthorLastName)
VALUES
   (1, 'John', 'Doe'),
   (2, 'Jane', 'Smith'),
   (3, 'David', 'Williams'),
   (4, 'Emily', 'Johnson'),
   (5, 'Michael', 'Brown'),
   (6, 'Samantha', 'Miller'),
   (7, 'Robert', 'Jones'),
   (8, 'Olivia', 'Davis'),
   (9, 'Daniel', 'Taylor'),
   (10, 'Emma', 'Moore');

-- Inserting sample data into BOOK table
INSERT INTO Book (BookID, EditionNumber, Title, Price)
VALUES
   (101, 1, 'Introduction to SQL', 25),
   (102, 2, 'Data Structures in Python', 30),
   (103, 1, 'Java Programming Basics', 20),
   (104, 2, 'Web Development Fundamentals', 35),
   (105, 1, 'C++ Programming Guide', 28),
   (106, 1, 'Algorithms and Complexity', 40),
   (107, 3, 'Advanced Database Management', 45),
   (108, 2, 'Python for Data Science', 33),
   (109, 1, 'Software Engineering Principles', 38),
   (110, 1, 'Networking Essentials', 30);

-- Inserting sample data into SUBJECT table
INSERT INTO Subject (SubjectID, SubjectName)
VALUES
   (1, 'Database Management'),
   (2, 'Programming'),
   (3, 'Data Structures'),
   (4, 'Web Development'),
   (5, 'Algorithms'),
   (6, 'Data Science'),
   (7, 'Software Engineering'),
   (8, 'Networking'),
   (9, 'Computer Security'),
   (10, 'Artificial Intelligence');
```

```
-- Inserting sample data into BOOKSUBJECT table
INSERT INTO BookSubject (BookID, EditionNumber, SubjectID)
VALUES
   (101, 1, 1),
   (102, 2, 3),
   (103, 1, 2),
   (104, 2, 4),
   (105, 1, 5),
   (106, 1, 5),
   (107, 3, 1),
   (108, 2, 6),
   (109, 1, 7),
   (110, 1, 8);

-- Inserting sample data into CUSTOMER table
INSERT INTO Customer (CustomerID)
VALUES
   (1001),
   (1002),
   (1003),
   (1004),
   (1005),
   (1006),
   (1007),
   (1008),
   (1009),
   (1010);

-- Inserting sample data into PAYMENTMETHOD table
INSERT INTO PaymentMethod (PaymentID, PaymentType, BillingStreet, BillingState, BillingZip,
ShippingStreet, ShippingState, ShippingZip)
VALUES
   (1, 'Credit Card', '123 Main St', 'CA', '90001', '123 Main St', 'CA', '90001'),
   (2, 'PayPal', '456 Oak St', 'NY', '10001', '456 Oak St', 'NY', '10001'),
   (3, 'Debit Card', '789 Elm St', 'TX', '75001', '789 Elm St', 'TX', '75001'),
   (4, 'Bank Transfer', '555 Maple St', 'FL', '33101', '555 Maple St', 'FL', '33101'),
   (5, 'Bank Transfer', '888 Pine St', 'WA', '98101', '888 Pine St', 'WA', '98101'),
   (6, 'Credit Card', '111 Birch St', 'IL', '60601', '111 Birch St', 'IL', '60601'),
   (7, 'Credit Card', '222 Cedar St', 'OH', '44101', '222 Cedar St', 'OH', '44101'),
   (8, 'Bank Transfer', '333 Walnut St', 'PA', '19101', '333 Walnut St', 'PA', '19101'),
   (9, 'Bank Transfer', '444 Fir St', 'GA', '30301', '444 Fir St', 'GA', '30301'),
   (10, 'PayPal', '999 Spruce St', 'NC', '28201', '999 Spruce St', 'NC', '28201');
```

```
-- Inserting sample data into CUSTOMERPAYMENTMETHOD table
INSERT INTO CustomerPaymentMethod (CustomerID, PaymentID)
VALUES
   (1001, 1),
   (1002, 2),
   (1003, 3),
   (1004, 4),
   (1005, 5),
   (1006, 6),
   (1007, 7),
   (1008, 8),
   (1009, 9),
   (1010, 10);

-- Inserting sample data into INSTITUTION table
INSERT INTO Institution (CustomerID, InstitutionID, InstitutionName)
VALUES
   (1001, 5001, 'ABC University'),
   (1003, 5002, 'XYZ College'),
   (1005, 5003, 'LMN Institute'),
   (1007, 5004, 'PQR Academy'),
   (1009, 5005, 'EFG School'),
   (1002, 5006, 'JKL College'),
   (1004, 5007, 'GHI University'),
   (1006, 5008, 'RST Institute'),
   (1008, 5009, 'UVW Academy'),
   (1010, 5010, 'NOP School');

-- Inserting sample data into INDIVIDUAL table
INSERT INTO Individual (CustomerID, IndividualID, IndividualFirstName, IndividualLastName)
VALUES
   (1002, 6001, 'Alice', 'Johnson'),
   (1004, 6002, 'Chris', 'Taylor'),
   (1006, 6003, 'Megan', 'Clark'),
   (1008, 6004, 'Ryan', 'Moore'),
   (1010, 6005, 'Sophie', 'Wright'),
   (1001, 6006, 'David', 'Hall'),
   (1003, 6007, 'Emma', 'Turner'),
   (1005, 6008, 'Jordan', 'Walker'),
   (1007, 6009, 'Ava', 'Hill'),
   (1009, 6010, 'Michael', 'Cooper');

-- Inserting sample data into ORDER table
INSERT INTO `Order` (OrderID, CustomerID, OrderDate)
```

```
VALUES
   (2001, 1001, STR_TO_DATE('3/20/2024', '%m/%d/%Y')),
   (2002, 1002, STR_TO_DATE('2/10/2021', '%m/%d/%Y')),
   (2003, 1003, STR_TO_DATE('7/4/2022', '%m/%d/%Y')),
   (2004, 1004, STR_TO_DATE('8/15/2023', '%m/%d/%Y')),
   (2005, 1005, STR_TO_DATE('6/20/2022', '%m/%d/%Y')),
   (2006, 1006, STR_TO_DATE('11/23/2023', '%m/%d/%Y')),
   (2007, 1007, STR_TO_DATE('12/22/2021', '%m/%d/%Y')),
   (2008, 1008, STR_TO_DATE('9/2/2022', '%m/%d/%Y')),
   (2009, 1009, STR_TO_DATE('1/30/2024', '%m/%d/%Y')),
   (2010, 1010, STR_TO_DATE('7/12/2023', '%m/%d/%Y'))
   ;

-- Inserting sample data into STUDENTACCOUNT table
INSERT INTO StudentAccount (StudentAccountNumber, StudentAccountFirstName,
StudentAccountLastName, StudentAccountUsername, StudentAccountPassword, CustomerID)
VALUES
   (3001, 'Bob', 'Miller', 'bob_miller123', 'password123', 1003),
   (3002, 'Lily', 'Carter', 'lily_carter456', 'securepass456', 1006),
   (3003, 'Tom', 'Brown', 'tom_brown789', 'strongpass789', 1009),
   (3004, 'Ella', 'White', 'ella_white101', 'mypassword101', 1002),
   (3005, 'Jack', 'King', 'jack_king202', 'secure123', 1004),
   (3006, 'Sophia', 'Adams', 'sophia_adams303', 'password456', 1008),
   (3007, 'William', 'Evans', 'william_evans404', 'passw0rd404', 1005),
   (3008, 'Grace', 'Baker', 'grace_baker505', 'myp@ssword505', 1007),
   (3009, 'Aiden', 'Fisher', 'aiden_fisher606', 'password606', 1001),
   (3010, 'Mia', 'Ross', 'mia_ross707', 'securepass707', 1010);

-- Inserting sample data into EMPLOYEE table
INSERT INTO Employee (EmployeeID, EmployeeFirstName, EmployeeLastName, AnnualSalary)
VALUES
   (4001, 'Sam', 'Jones', 60000),
   (4002, 'Emily', 'Davis', 70000),
   (4003, 'Daniel', 'Clark', 75000),
   (4004, 'Olivia', 'Hill', 80000),
   (4005, 'Michael', 'Turner', 85000),
   (4006, 'Sophie', 'Wright', 90000),
   (4007, 'Ethan', 'Cooper', 95000),
   (4008, 'Ava', 'Walker', 100000),
   (4009, 'Jacob', 'Hill', 105000),
   (4010, 'Emma', 'Turner', 110000);

-- Inserting sample data into OFFICE table
INSERT INTO Office (EmployeeID, OfficeID)
```

```
VALUES
   (4001, 10001),
   (4002, 10002),
   (4003, 10003),
   (4004, 10004),
   (4005, 10005),
   (4006, 10006),
   (4007, 10007),
   (4008, 10008),
   (4009, 10009),
   (4010, 10010);

-- Inserting sample data into VIDEOTUTOR table
INSERT INTO VideoTutor (EmployeeID, VideoTutorID)
VALUES
   (4001, 20001),
   (4002, 20002),
   (4003, 20003),
   (4004, 20004),
   (4005, 20005),
   (4006, 20006),
   (4007, 20007),
   (4008, 20008),
   (4009, 20009),
   (4010, 20010);

-- Inserting sample data into DEPARTMENT table
INSERT INTO Department (DepartmentID, DepartmentName)
VALUES
   (10001, 'IT Department'),
   (10002, 'HR Department'),
   (10003, 'Finance Department'),
   (10004, 'Marketing Department'),
   (10005, 'Sales Department'),
   (10006, 'Research and Development'),
   (10007, 'Customer Support'),
   (10008, 'Quality Assurance'),
   (10009, 'Legal Department'),
   (10010, 'Management');

-- Inserting sample data into VIDEOTUTORSUBJECT table
INSERT INTO VideoTutorSubject (VideoTutorSubjectID, VideoTutorSubjectName)
VALUES
   (1, 'Database Design'),
```

```
    (2, 'Python Programming'),
    (3, 'Web Development'),
    (4, 'Algorithms'),
    (5, 'Data Science'),
    (6, 'Software Engineering'),
    (7, 'Networking'),
    (8, 'Computer Security'),
    (9, 'Artificial Intelligence'),
    (10, 'Machine Learning');

-- Inserting sample data into AUTHORBOOK table
INSERT INTO AuthorBook (AuthorID, BookID, EditionNumber)
VALUES
    (1, 101, 1),
    (2, 102, 2),
    (3, 103, 1),
    (4, 104, 2),
    (5, 105, 1),
    (6, 106, 1),
    (7, 107, 3),
    (8, 108, 2),
    (9, 109, 1),
    (10, 110, 1);

-- Inserting sample data into BOOKORDER table
INSERT INTO BookOrder (BookID, EditionNumber, OrderID, Quantity)
VALUES
    (101, 1, 2001, 10),
    (102, 2, 2002, 15),
    (103, 1, 2003, 1),
    (104, 2, 2004, 2),
    (105, 1, 2005, 32),
    (106, 1, 2006, 1),
    (107, 3, 2007, 2),
    (108, 2, 2008, 5),
    (109, 1, 2009, 29),
    (110, 1, 2010, 16);

-- Inserting sample data into ORDEREMPLOYEE table
INSERT INTO OrderEmployee (OrderID, EmployeeID)
VALUES
    (2001, 4001),
    (2002, 4002),
    (2003, 4003),
```

```
    (2004, 4004),
    (2005, 4005),
    (2006, 4006),
    (2007, 4007),
    (2008, 4008),
    (2009, 4009),
    (2010, 4010);

-- Inserting sample data into VIDEOTUTORVIDEOTUTORSUBJECT table
INSERT INTO VideoTutorVideoTutorSubject (EmployeeID, VideoTutorSubjectID)
VALUES
    (4001, 1),
    (4002, 2),
    (4003, 3),
    (4004, 4),
    (4005, 5),
    (4006, 6),
    (4007, 7),
    (4008, 8),
    (4009, 9),
    (4010, 10);

-- Inserting sample data into OFFICEDEPARTMENT table
INSERT INTO OfficeDepartment (EmployeeID, DepartmentID)
VALUES
    (4001, 10001),
    (4002, 10002),
    (4003, 10003),
    (4004, 10004),
    (4005, 10005),
    (4006, 10006),
    (4007, 10007),
    (4008, 10008),
    (4009, 10009),
    (4010, 10010);

-- Inserting sample data into EBOOK table
INSERT INTO Ebook (BookID, EditionNumber, LicenseID)
VALUES
    (101, 1, 1001),
    (102, 2, 1002),
    (103, 1, 1003),
    (104, 2, 1004),
    (105, 1, 1005),
```

```
   (106, 1, 1006),
   (107, 3, 1007),
   (108, 2, 1008),
   (109, 1, 1009),
   (110, 1, 1010);

-- Inserting sample data into EBOOKVIDEOTUTORSUBJECT table
INSERT INTO EbookVideoTutorSubject (BookID, EditionNumber, VideoTutorSubjectID)
VALUES
   (102, 2, 2),
   (104, 2, 4),
   (106, 1, 6),
   (108, 2, 8),
   (110, 1, 10),
   (103, 1, 3),
   (105, 1, 5),
   (107, 3, 7),
   (109, 1, 9),
   (101, 1, 1);


-- Inserting sample data into STUDENTEBOOK table
INSERT INTO StudentEbook (StudentAccountNumber, BookID, EditionNumber)
VALUES
   (3001, 102, 2),
   (3002, 104, 2),
   (3003, 106, 1),
   (3004, 108, 2),
   (3005, 110, 1),
   (3006, 103, 1),
   (3007, 105, 1),
   (3008, 107, 3),
   (3009, 109, 1),
   (3010, 101, 1);
```

Testing

1. <u>Query 1: Top selling book:</u>

   ```
   SELECT b.BookID,  b.Title, b.EditionNumber,  COUNT(o.OrderID) AS TotalOrders
   FROM  Book b
   ```

```
JOIN BookOrder bo ON b.BookID = bo.BookID AND b.EditionNumber =
bo.EditionNumber
JOIN `Order` o ON bo.OrderID = o.OrderID
GROUP BY b.BookID, b.Title, b.EditionNumber
ORDER BY TotalOrders DESC
LIMIT 1;
```

Result Grid Filter Rows: Search Export: Fetch rows:

BookID	Title	EditionNumber	TotalOrders
101	Introduction to SQL	1	1

Result 1

2. <u>Query 2: Employees with highest salary in each department:</u>

```
SELECT EmployeeID, EmployeeFirstName, EmployeeLastName, DepartmentName,
AnnualSalary
FROM Employee
JOIN (
   SELECT DepartmentID, MAX(AnnualSalary) AS MaxSalary
   FROM Employee
   JOIN OfficeDepartment USING (EmployeeID)
   GROUP BY DepartmentID
) AS MaxSalaries
ON Employee.AnnualSalary = MaxSalaries.MaxSalary
JOIN Department USING (DepartmentID);
```

Result Grid Filter Rows: Search Export:

EmployeeID	EmployeeFirstName	EmployeeLastName	DepartmentName	AnnualSalary
4001	Sam	Jones	IT Department	60000
4002	Emily	Davis	HR Department	70000
4003	Daniel	Clark	Finance Department	75000
4004	Olivia	Hill	Marketing Department	80000
4005	Michael	Turner	Sales Department	85000
4006	Sophie	Wright	Research and Development	90000
4007	Ethan	Cooper	Customer Support	95000
4008	Ava	Walker	Quality Assurance	100000
4009	Jacob	Hill	Legal Department	105000
4010	Emma	Turner	Management	110000

Result 2

3. <u>Query 3: Total number of books ordered by each institution:</u>

```
SELECT i.InstitutionName, COUNT(bo.BookID) AS TotalBooksOrdered
FROM Institution i
JOIN Customer c ON i.CustomerID = c.CustomerID
JOIN `Order` o ON c.CustomerID = o.CustomerID
JOIN BookOrder bo ON o.OrderID = bo.OrderID
GROUP BY i.InstitutionName;
```

Result Grid Filter Rows: Search Export:

InstitutionNa...	TotalBooksOrder...
ABC University	1
JKL College	1
XYZ College	1
GHI University	1
LMN Institute	1
RST Institute	1
PQR Academy	1
UVW Academy	1
EFG School	1
NOP School	1

Result 3

4. <u>Query 4: List of books ordered by XYZ college:</u>

```
SELECT BookID, EditionNumber, Title
FROM Book
JOIN Ebook USING (BookID, EditionNumber)
JOIN StudentEbook USING (BookID, EditionNumber)
JOIN StudentAccount USING (StudentAccountNumber)
WHERE CustomerID IN (SELECT CustomerID FROM Institution WHERE InstitutionID =
5002);
```

Result Grid Filter Rows: Search Export:

BookID	EditionNumber	Title
102	2	Data Structures in Python

5. <u>Query 5: Top 3 individual customers by total order amount</u>

```
SELECT c.CustomerID, i.IndividualFirstName, i.IndividualLastName, SUM(b.Price *
bo.Quantity) AS TotalOrderAmount
FROM Customer c
JOIN `Order` o ON c.CustomerID = o.CustomerID
JOIN BookOrder bo ON o.OrderID = bo.OrderID
JOIN Book b ON bo.BookID = b.BookID AND bo.EditionNumber = b.EditionNumber
JOIN Individual i ON c.CustomerID = i.CustomerID
GROUP BY c.CustomerID
ORDER BY TotalOrderAmount DESC
LIMIT 3;
```

Result Grid Filter Rows: Search Export: Fetch rows:

CustomerID	IndividualFirstNa...	IndividualLastNa...	TotalOrderAmou...
1009	Michael	Cooper	1102
1005	Jordan	Walker	896
1010	Sophie	Wright	480

Result 14