Contents

Online Bookstore Management System
(Oracle-Database-Project)

1 Introduction

A modern database system underpins efficient data management for various tasks. This introduction offers an overview of our database designed to support an online bookstore management system.

Built on a well-structured Entity-Relationship diagram, the system connects entities like Customers, Books, Authors, Genres, Orders, and Payments, forming a comprehensive data model.

Key tables include Customer and Payment, storing crucial customer and financial data. The former aids personalized interactions, while the latter records secure transactions.

Data integrity is ensured through primary and foreign key constraints, while query tools enable analysis of customer behavior and book sales.

Security measures encompass access controls, authentication, and backups to protect sensitive data and ensure uninterrupted operations.

1.1 Motivation

In today’s digital age, online bookstores have become increasingly popular, catering to a vast and diverse audience of readers. To efficiently manage the extensive inventory, customer data, and financial transactions, a robust and scalable database system is essential. This document provides an overview of how the Online Bookstore Management Database System was conceptualized, designed, and implemented to streamline operations and enhance the user experience.

2 Scenario Description

In an Online Bookstore Management System, a customer can order many books. One book may be ordered by multiple customers. A customer is identified by a customer ID. The system also stores customers' first name, last name, E-mail, phone no. and address. A customer address is composed of house number and city. There may be multiple phone numbers of a customer. A book is identified by book ID, price, title, publish date and stocks of a book are also stored. While ordering the date of the ordering, total amount and order ID is stored. A customer can make one or multiple payments but a payment is made by one customer. The payment is identified by payment ID. The system also stores the date of the payment, payment method and amount. A book is written by at least one author. An author may write many books but the system stores information of those authors who has written at least one book stored in the bookstore. To identify an author the system stores author ID along with author name and their biography. A book may belong to multiple genre and for a genre there may be multiple books. Each genre has a name and the unique property of each genre is a genre ID.

3 Relationships & Cardinality

1. Customer - Order (One-to-Many):
   - Each customer can place zero or multiple orders.
   - An order is placed by one customer.

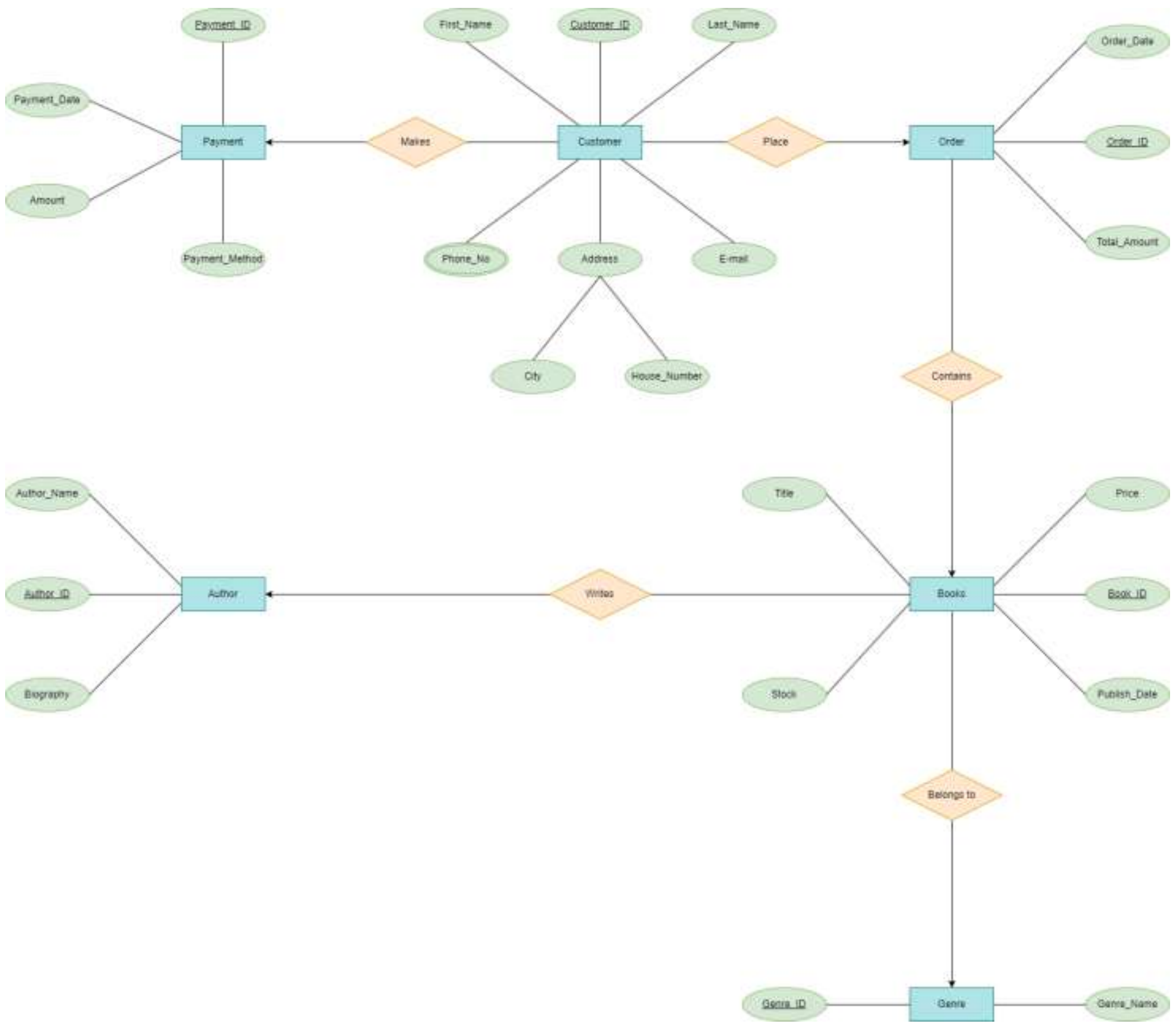
2. Customer - Payment (One-to-Many):
   - Each customer can make zero or multiple payments.
   - A payment is made by one customer.

3. Order - Book (Many-to-Many):
   - An order can contain one or more books.
   - A book can be a part of multiple orders.

4. Book - Author (One-to-Many):
   - A book can have only one author.
   - Each author can write one or multiple books.

5. Book - Genre (Many-to-Many):
   - A book may belong to multiple genre.
   - A genre can have multiple books.

4 Entity-Relationship Diagram



5 Normalization

5.1 Customer-Place-Order

Relation: One to Many

UNF: Customer_ID, First_Name, Last_Name, Phone_No, E-mail, City, House_Number,
Order_ID , Order_Date, Total_Amount

1NF: Multivalued attributes: Phone_no,
Customer_ID, First_Name, Last_Name, E-mail, City, House_Number,
Order_ID , Order_Date, Total_Amount

2NF:

1. Customer_ID, Phone no
2. Customer_ID, First_Name, Last_Name, E-mail, City, House_Number
3. Order_ID ,Order_Date, Total_Amount, Customer_ID

3NF:

1. Customer_ID, City, House_Number
2. Customer_ID, Phone_No
3. Customer_ID, First_Name, Last_Name, E-mail
4. Order_ID , Order_Date, Total_Amount, Customer_ID

5.2 Customer-Makes-Payment

Relation: One to Many

UNF: Customer_ID, First_Name, Last_Name,Phone_No, E-mail, City, House_Number,
Payment_ID, Payment_Date, Amount, Payment_Method

1NF: Multivalued attributes: Phone_no,
Customer_ID, First_Name, Last_Name, E-mail, City, House_Number,
Payment_ID, Payment_Date, Amount, Payment_Method

2NF:

1. Customer_ID, Phone_No
2. Customer_ID, First_Name, Last_Name, E-mail, City, House_Number
3. Payment_ID, Payment_Date, Amount, Payment_Method, Customer_ID

3NF:

1. Customer_ID, City, House_Number
2. Customer_ID, Phone_No
3. Customer_ID, First_Name, Last_Name, E-mail
4. Payment_ID, Payment_Date, Amount, Payment_Method, Customer_ID

5.3 Order-Contains-Books

Relation: Many to Many

UNF: Order_ID, Order_Date, Total_Amount, Book_ID, Publish_Date, Stock, Title, Price
1NF: Order_ID, Order_Date, Total_Amount, Book_ID, Publish_Date, Stock, Title, Price
2NF:

1. Order_ID, Order_Date, Total_Amount, Book_ID
2. Book_ID, Publish_Date, Stock, Title, Price, Order_ID

3NF:

1. Order_ID, Order_Date, Total_Amount, Book_ID
2. Book_ID, Publish_Date, Stock, Title, Price, Order_ID

5.4 Books-Writes-Author

Relation: One to Many

UNF: Book_ID, Publish_Date, Stock, Title, Price, Author_ID, Author_Name, Biography
1NF: Book_ID, Publish_Date, Stock, Title, Price, Author_ID, Author_Name, Biography
2NF:

1. Book_ID, Publish_Date, Stock, Title, Price, Author_ID
2. Author_ID, Author_Name, Biography

3NF:

1. Book_ID, Publish_Date, Stock, Title, Price, Author_ID
2. Author_ID, Author_Name, Biography

5.5 Books-Belongs to-Genre

Relation: Many to Many

UNF: Book_ID, Publish_Date, Stock, Title, Price, Genre_ID, Genre_Name
1NF: Book_ID, Publish_Date, Stock, Title, Price, Genre_ID, Genre_Name
2NF:

1. Book_ID, Publish_Date, Stock, Title, Price, Genre_ID
2. Genre_ID, Genre_Name, Book_ID

3NF:

1. Book_ID, Publish_Date, Stock, Title, Price, Genre_ID
2. Genre_ID, Genre_Name, Book_ID

6 Finalization

1. Customer_ID, City, House_Number
2. Customer_ID, Phone_No
3. Customer_ID, First_Name, Last_Name, E-mail
4. Order_ID , Order_Date, Total_Amount, Customer_ID
5. Customer_ID, City, House_Number
6. Customer_ID, Phone_No
7. Customer_ID, First_Name, Last_Name, E-mail
8. Payment_ID, Payment_Date, Amount, Payment_Method, Customer_ID
9. Order_ID, Order_Date, Total_Amount, Book_ID
10. Book_ID, Publish_Date, Stock, Title, Price, Order_ID
11. Book_ID, Publish_Date, Stock, Title, Price, Author_ID
12. Author_ID, Author_Name, Biography
13. Book_ID, Publish_Date, Stock, Title, Price, Genre_ID
14. Genre_ID, Genre_Name, Book_ID

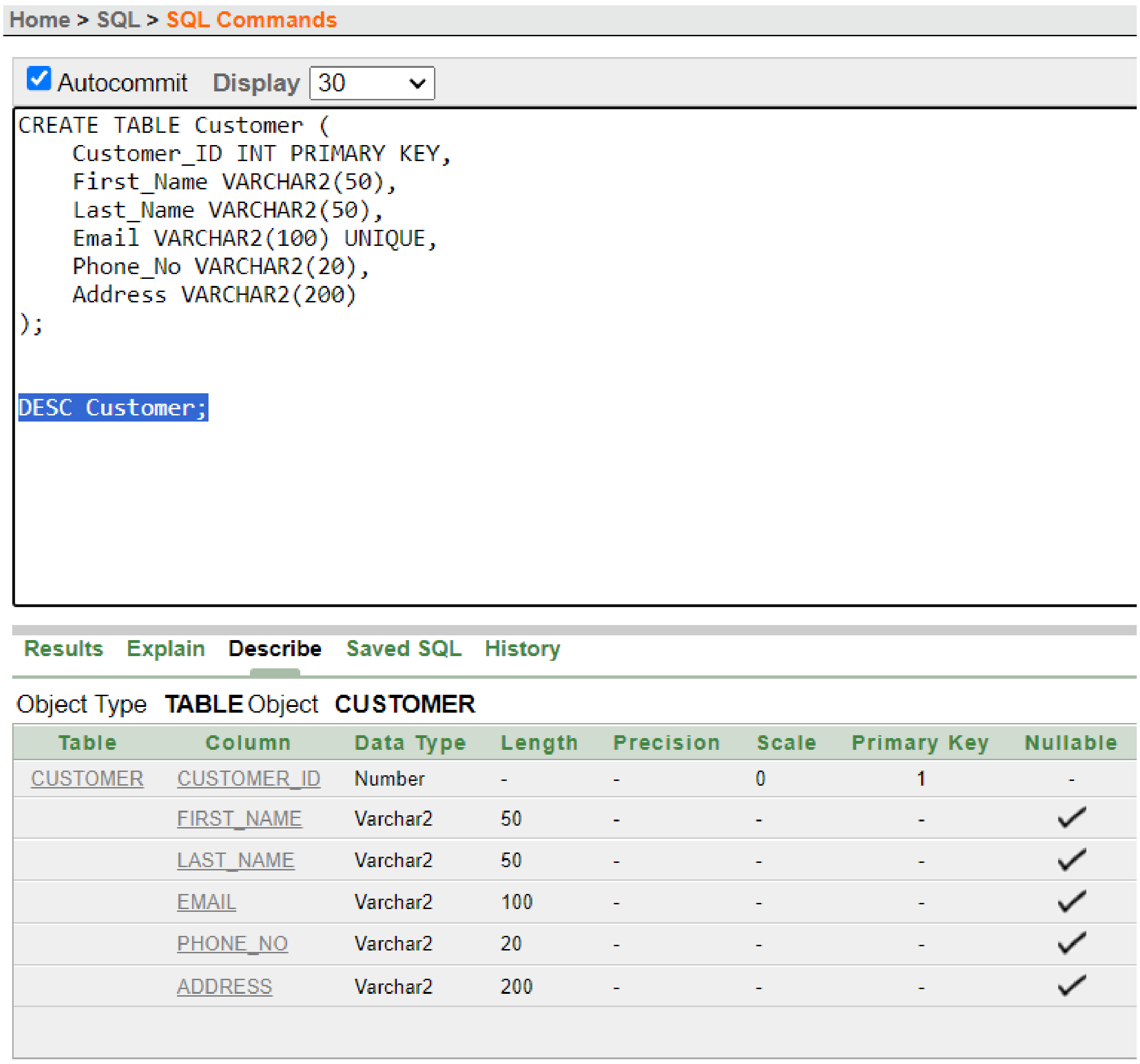
6.1 Optimization

1. Customer_ID, City, House_Number
2. Customer_ID, Phone_No
3. Customer_ID, First_Name, Last_Name, E-mail
4. Order_ID , Order_Date, Total_Amount, Customer_ID
5. Payment_ID, Payment_Date, Amount, Payment_Method, Customer_ID
6. Order_ID, Order_Date, Total_Amount, Book_ID
7. Book_ID, Publish_Date, Stock, Title, Price, Order_ID
8. Book_ID, Publish_Date, Stock, Title, Price, Author_ID
9. Author_ID, Author_Name, Biography
10. Book_ID, Publish_Date, Stock, Title, Price, Genre_ID
11. Genre_ID, Genre_Name, Book_ID

7 Table Creation

Creating Customer Table

```
   CREATE TABLE Customer (
   Customer_ID INT PRIMARY KEY,
   First_Name VARCHAR2(50),
   Last_Name VARCHAR2(50),
   Email VARCHAR2(100) UNIQUE,
   Phone_No VARCHAR2(20),
   Address VARCHAR2(200)
);
```


Home > SQL > SQL Commands
Autocommit Display 30
Results Explain Describe Saved SQL History

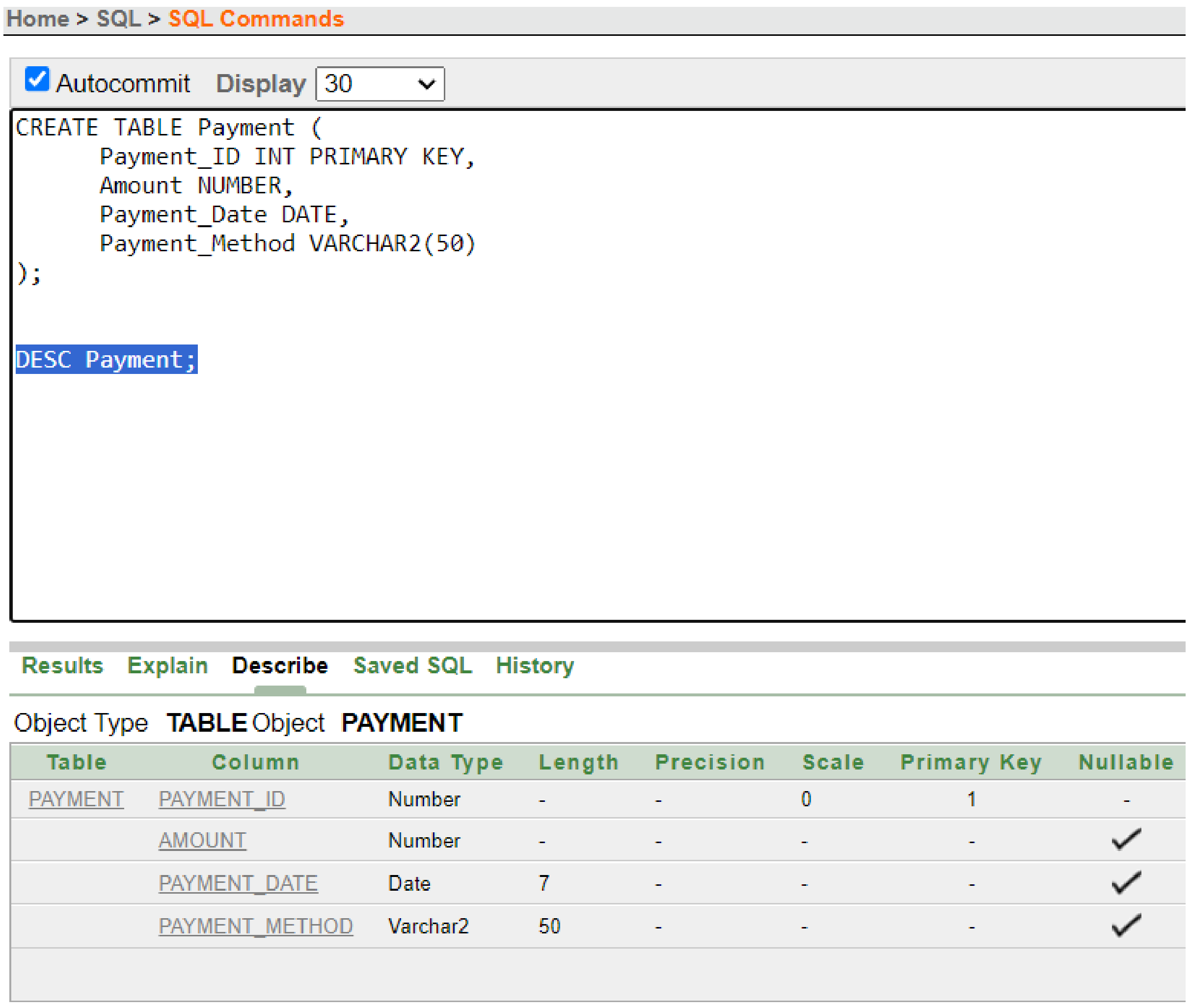
```
CREATE TABLE Customer (
    Customer_ID INT PRIMARY KEY,
    First_Name VARCHAR2(50),
    Last_Name VARCHAR2(50),
    Email VARCHAR2(100) UNIQUE,
    Phone_No VARCHAR2(20),
    Address VARCHAR2(200)
);


DESC Customer;
```

Object Type **TABLE** Object **CUSTOMER**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable
CUSTOMER	CUSTOMER_ID	Number	-	-	0	1	-
	FIRST_NAME	Varchar2	50	-	-	-	✓
	LAST_NAME	Varchar2	50	-	-	-	✓
	EMAIL	Varchar2	100	-	-	-	✓
	PHONE_NO	Varchar2	20	-	-	-	✓
	ADDRESS	Varchar2	200	-	-	-	✓

Creating Payment Table

```
    CREATE TABLE **Payment** (
    Payment_ID INT PRIMARY KEY,
    Amount NUMBER,
    Payment_Date DATE,
    Payment_Method VARCHAR2(50)
);
```



Home > SQL > SQL Commands
Autocommit Display 30
Results Explain Describe Saved SQL History

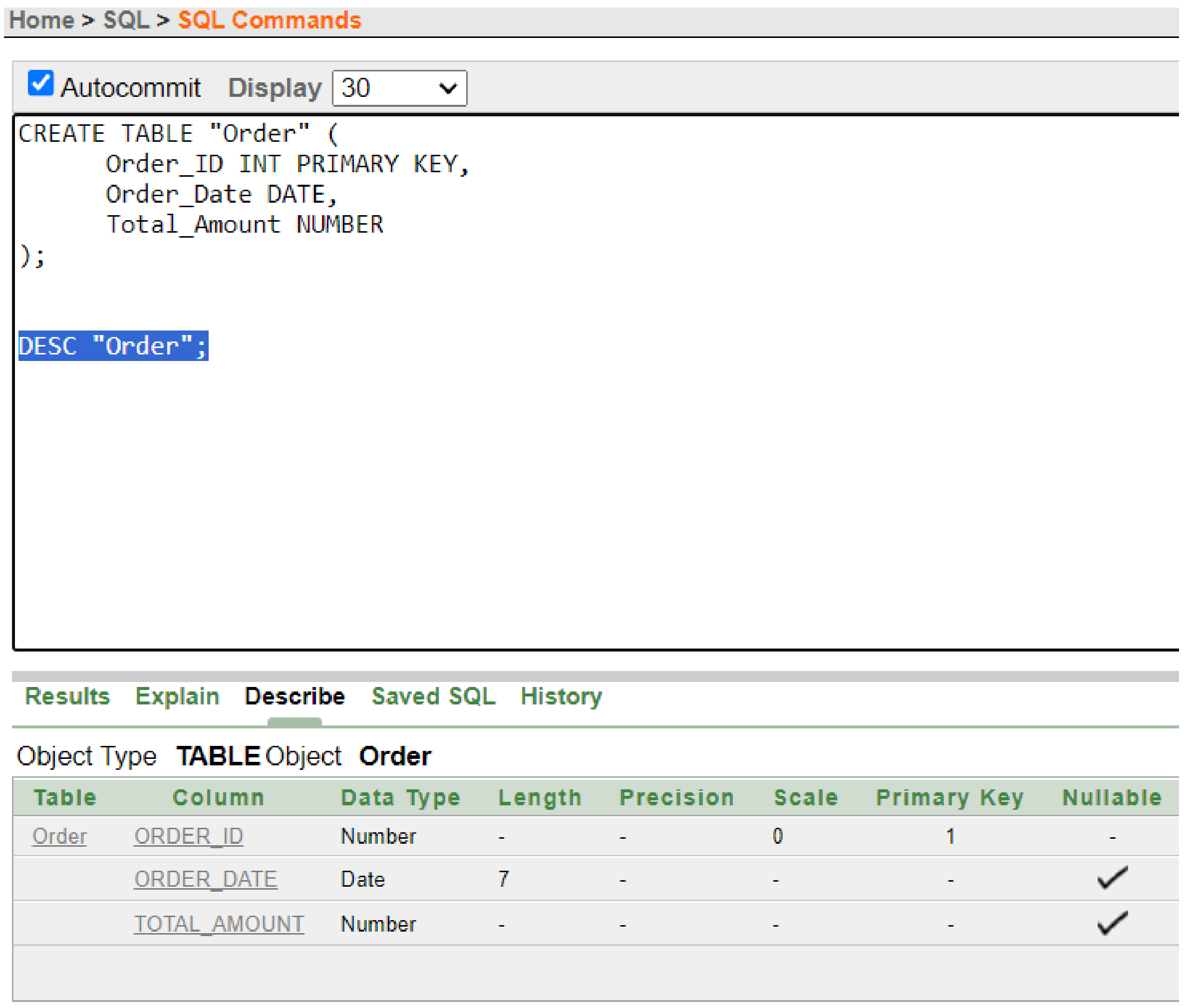
```
CREATE TABLE Payment (
      Payment_ID INT PRIMARY KEY,
      Amount NUMBER,
      Payment_Date DATE,
      Payment_Method VARCHAR2(50)
);


DESC Payment;
```

Object Type **TABLE** Object **PAYMENT**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable
PAYMENT	PAYMENT_ID	Number	-	-	0	1	-
	AMOUNT	Number	-	-	-	-	✓
	PAYMENT_DATE	Date	7	-	-	-	✓
	PAYMENT_METHOD	Varchar2	50	-	-	-	✓

Creating Order Table

```
    CREATE TABLE "Order" (
    Order_ID INT PRIMARY KEY,
    Order_Date DATE,
    Total_Amount NUMBER
);
```



Home > SQL > SQL Commands

Autocommit Display 30

```
CREATE TABLE "Order" (
      Order_ID INT PRIMARY KEY,
      Order_Date DATE,
      Total_Amount NUMBER
);


DESC "Order";
```

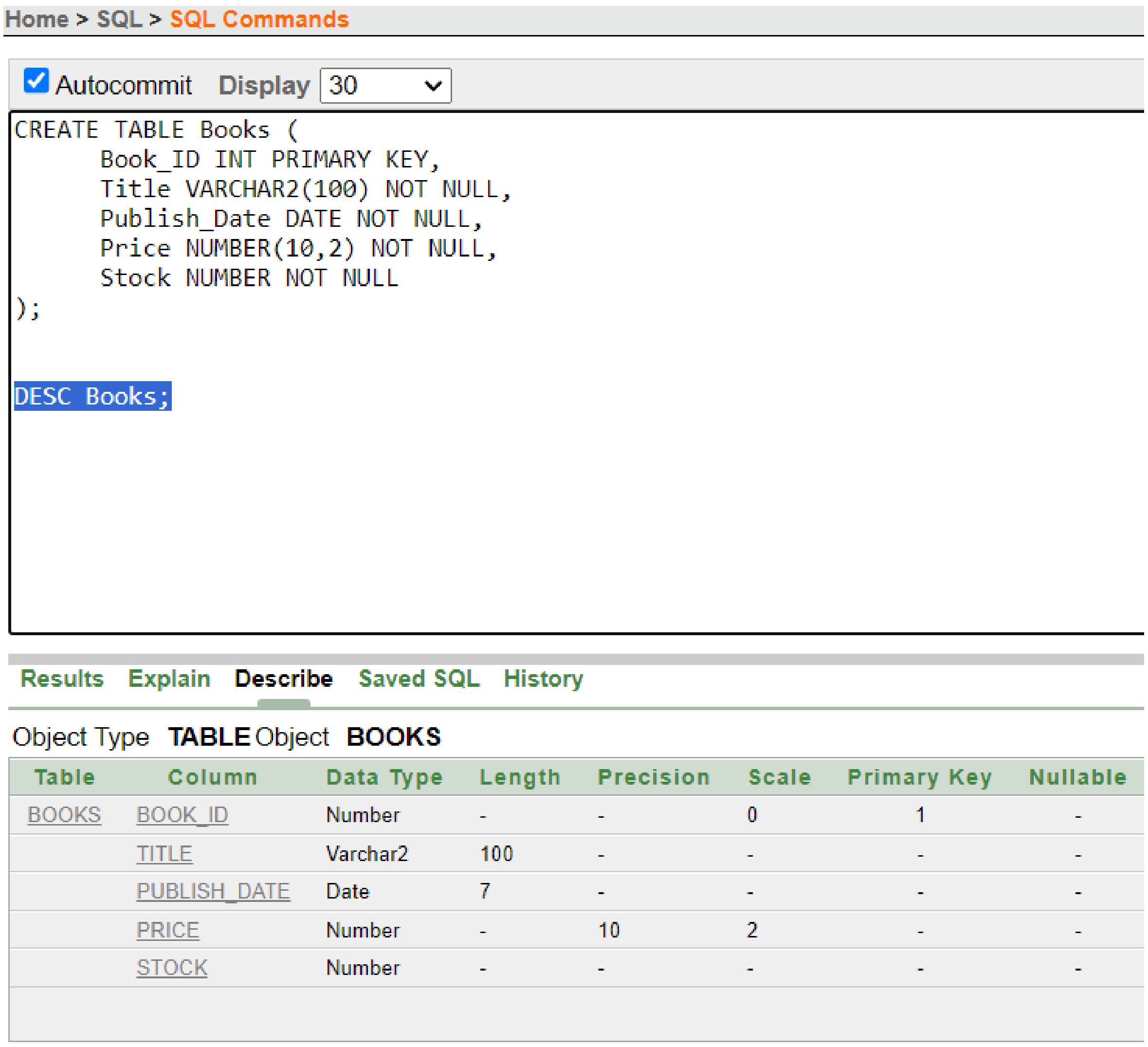
Results Explain Describe Saved SQL History

Object Type **TABLE** Object **Order**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable
Order	ORDER_ID	Number	-	-	0	1	-
	ORDER_DATE	Date	7	-	-	-	✓
	TOTAL_AMOUNT	Number	-	-	-	-	✓

Creating Books Table

```
    CREATE TABLE Books (
    Book_ID INT PRIMARY KEY,
    Title VARCHAR2(100) NOT NULL,
    Publish_Date DATE NOT NULL,
    Price NUMBER(10,2) NOT NULL,
    Stock NUMBER NOT NULL
);
```



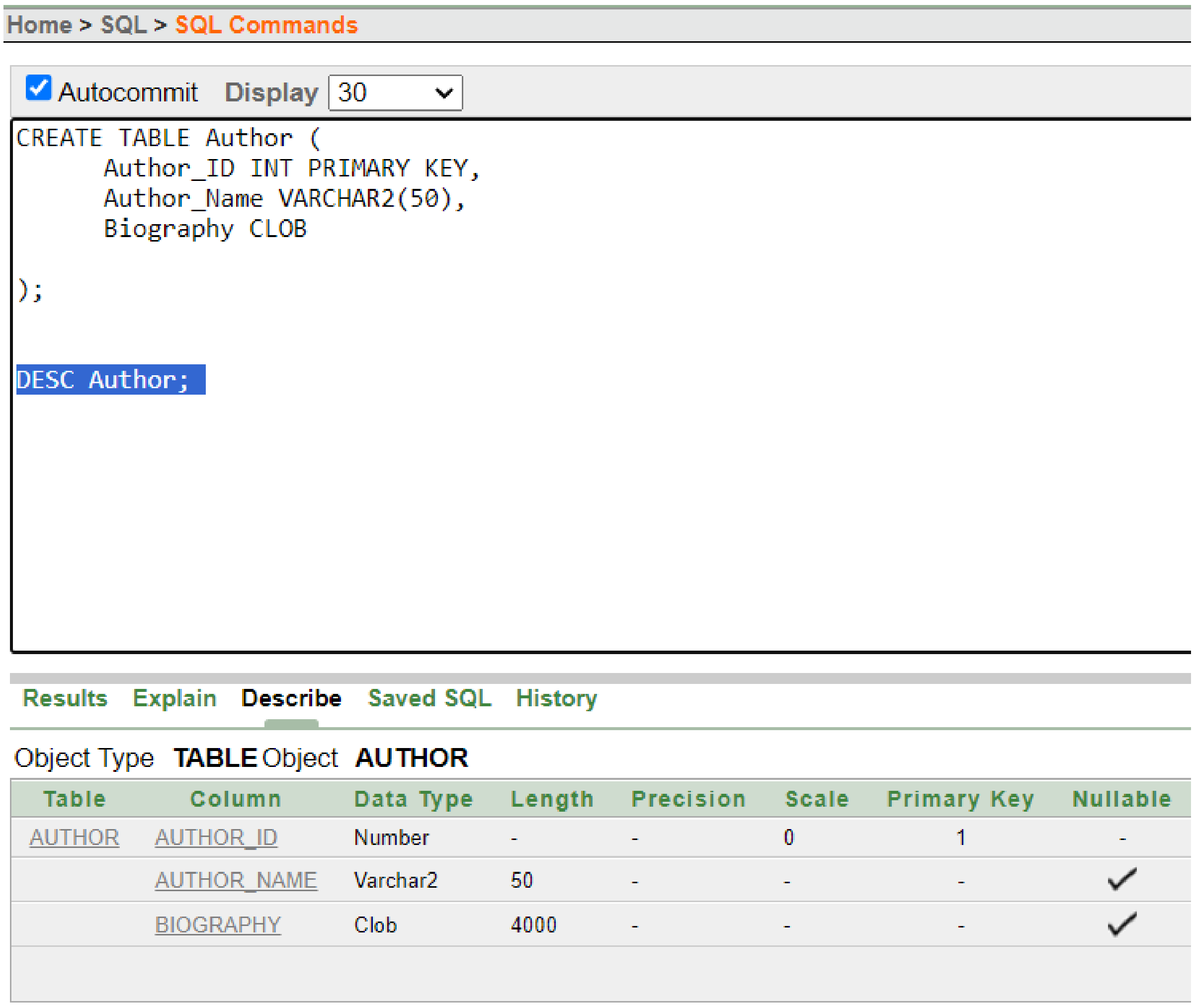
```
CREATE TABLE Books (
      Book_ID INT PRIMARY KEY,
      Title VARCHAR2(100) NOT NULL,
      Publish_Date DATE NOT NULL,
      Price NUMBER(10,2) NOT NULL,
      Stock NUMBER NOT NULL
);


DESC Books;
```

Object Type **TABLE** Object **BOOKS**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable
BOOKS	BOOK_ID	Number	-	-	0	1	-
	TITLE	Varchar2	100	-	-	-	-
	PUBLISH_DATE	Date	7	-	-	-	-
	PRICE	Number	-	10	2	-	-
	STOCK	Number	-	-	-	-	-

Creating Author Table

```
 CREATE TABLE Author (
Author_ID INT PRIMARY KEY,
Author_Name VARCHAR2(50),
Biography CLOB

);
```



Home > SQL > SQL Commands

Autocommit Display 30

```
CREATE TABLE Author (
      Author_ID INT PRIMARY KEY,
      Author_Name VARCHAR2(50),
      Biography CLOB

);


DESC Author;
```

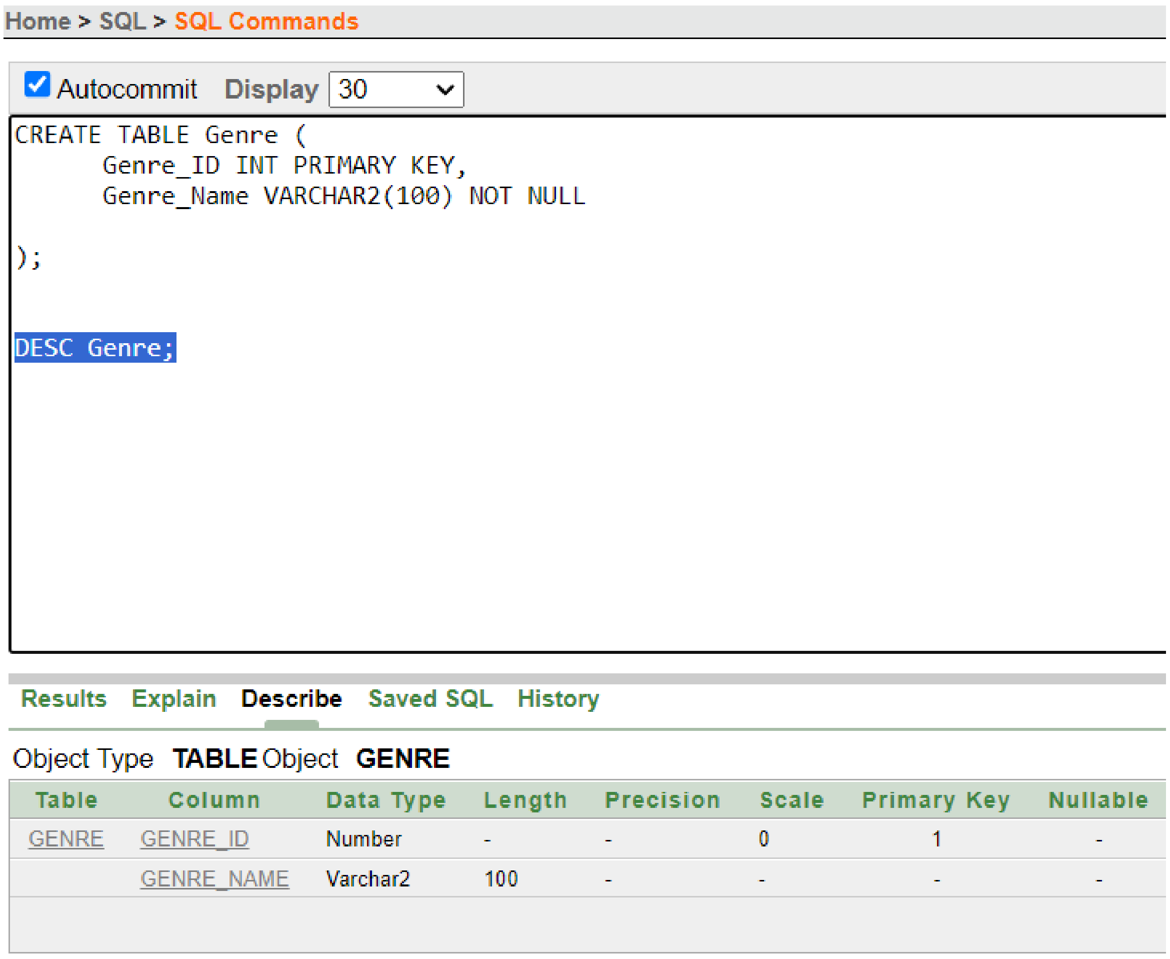
Results Explain Describe Saved SQL History

Object Type TABLE Object AUTHOR

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable
AUTHOR	AUTHOR_ID	Number	-	-	0	1	-
	AUTHOR_NAME	Varchar2	50	-	-	-	✓
	BIOGRAPHY	Clob	4000	-	-	-	✓

Creating Genre Table

```
CREATE TABLE Genre (
Genre_ID INT PRIMARY KEY,
Genre_Name VARCHAR2(100) NOT NULL
);
```



Home > SQL > SQL Commands

Autocommit Display 30

```
CREATE TABLE Genre (
      Genre_ID INT PRIMARY KEY,
      Genre_Name VARCHAR2(100) NOT NULL

);


DESC Genre;
```

Results Explain Describe Saved SQL History

Object Type **TABLE** Object **GENRE**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable
GENRE	GENRE_ID	Number	-	-	0	1	-
	GENRE_NAME	Varchar2	100	-	-	-	-

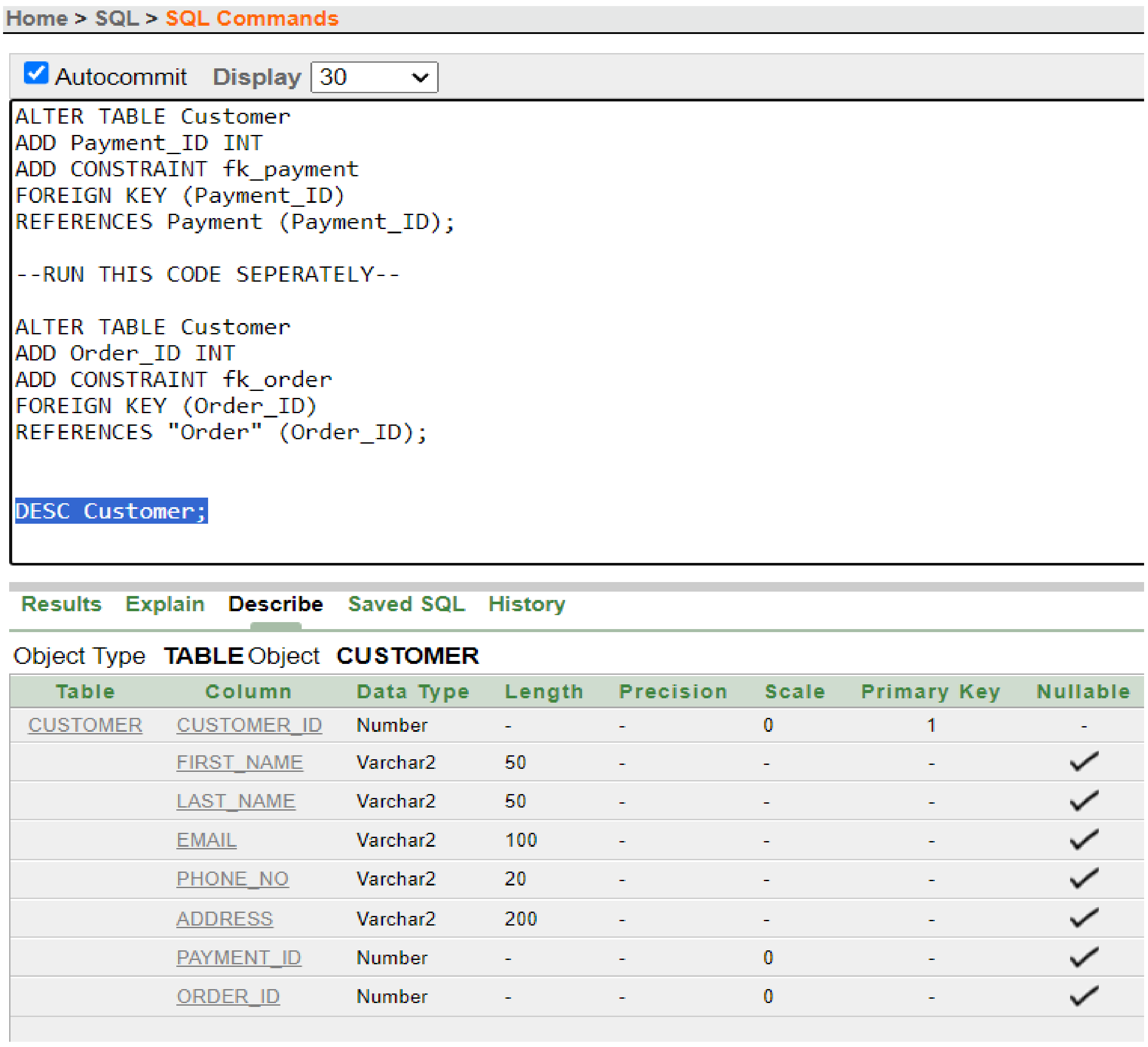
7.1 Setting Constraints

Constraint of Customer

Since the entity titled-'Customer' has a one-to-many relationship with the entities titled-'Payment' and 'Order'. So, the table-Customer has 2 foreign keys titled-'Payment_ID' and 'Order_ID'.

```
ALTER TABLE Customer
ADD Payment_ID INT
ADD CONSTRAINT fk_payment
FOREIGN KEY (Payment_ID)
REFERENCES Payment (Payment_ID);
```

```
ALTER TABLE Customer
ADD Order_ID INT
ADD CONSTRAINT fk_order
FOREIGN KEY (Order_ID)
REFERENCES "Order" (Order_ID);
```



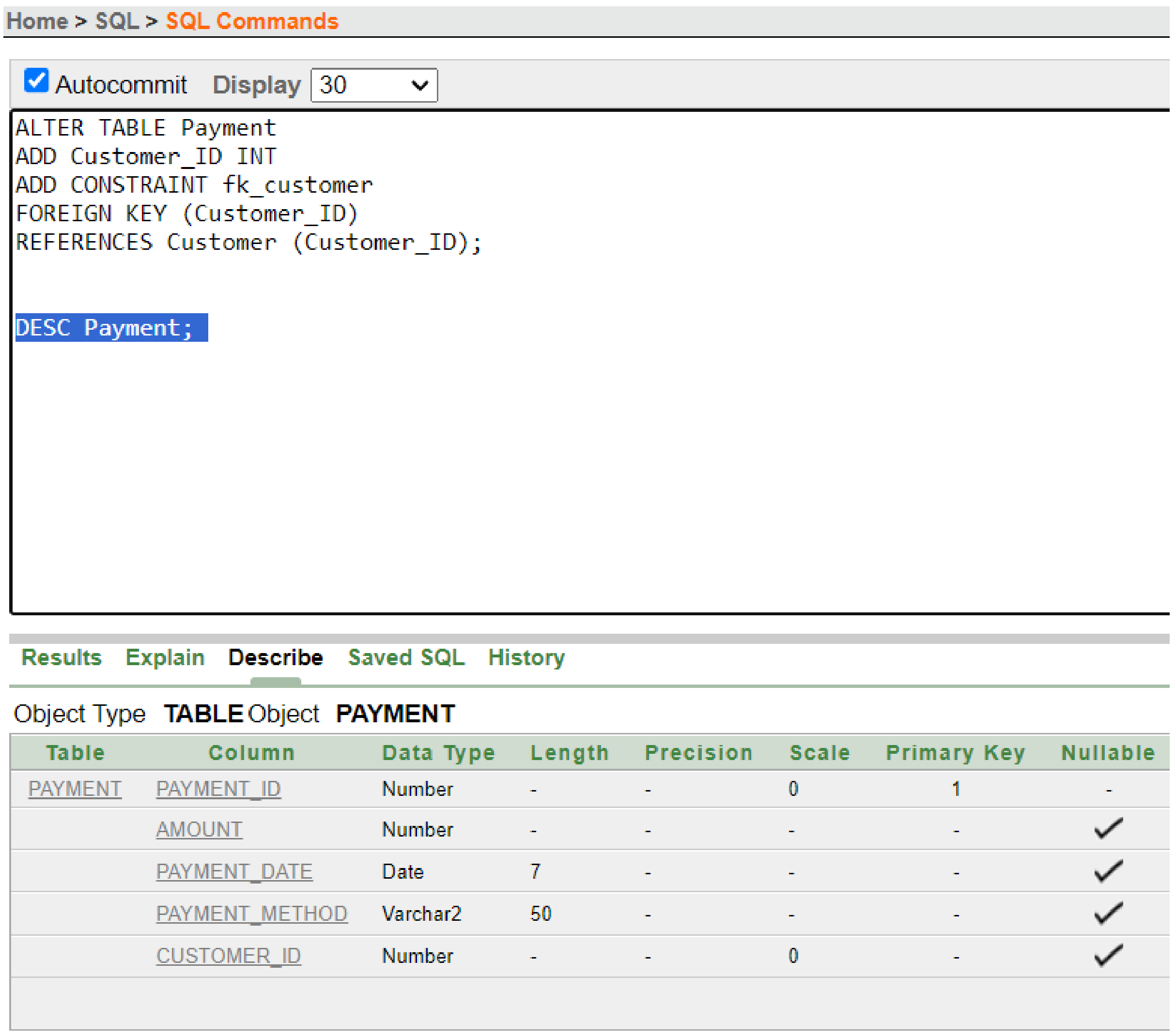
```
ALTER TABLE Customer
ADD Payment_ID INT
ADD CONSTRAINT fk_payment
FOREIGN KEY (Payment_ID)
REFERENCES Payment (Payment_ID);

--RUN THIS CODE SEPERATELY--

ALTER TABLE Customer
ADD Order_ID INT
ADD CONSTRAINT fk_order
FOREIGN KEY (Order_ID)
REFERENCES "Order" (Order_ID);


DESC Customer;
```

Object Type **TABLE** Object **CUSTOMER**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable
CUSTOMER	CUSTOMER_ID	Number	-	-	0	1	-
	FIRST_NAME	Varchar2	50	-	-	-	✓
	LAST_NAME	Varchar2	50	-	-	-	✓
	EMAIL	Varchar2	100	-	-	-	✓
	PHONE_NO	Varchar2	20	-	-	-	✓
	ADDRESS	Varchar2	200	-	-	-	✓
	PAYMENT_ID	Number	-	-	0	-	✓
	ORDER_ID	Number	-	-	0	-	✓

Constraint of Payment

```
ALTER TABLE Payment
ADD Customer_ID INT
ADD CONSTRAINT fk_customer
FOREIGN KEY (Customer_ID)
REFERENCES Customer (Customer_ID);
```



```
ALTER TABLE Payment
ADD Customer_ID INT
ADD CONSTRAINT fk_customer
FOREIGN KEY (Customer_ID)
REFERENCES Customer (Customer_ID);


DESC Payment;
```

Object Type **TABLE** Object **PAYMENT**

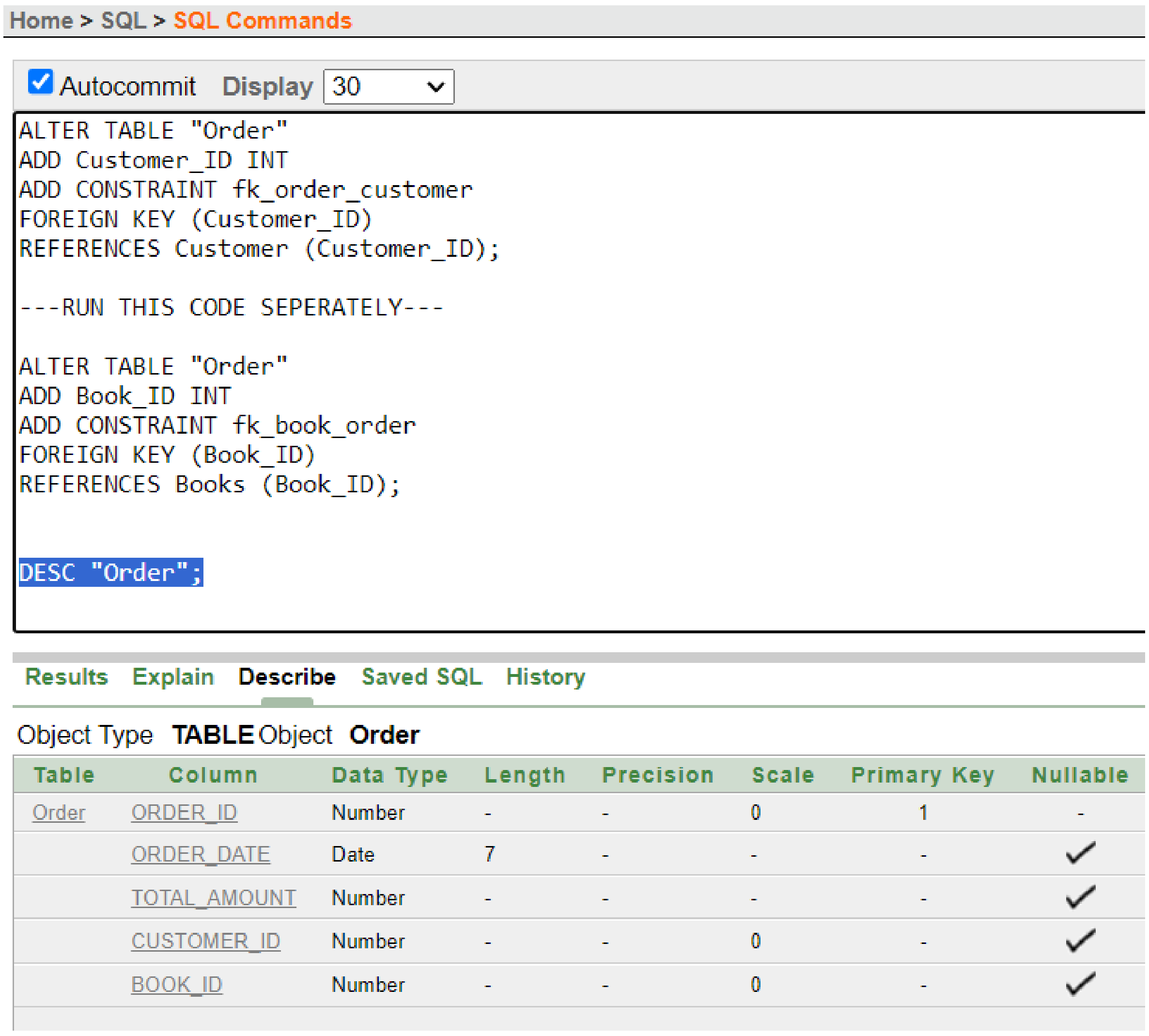
Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable
PAYMENT	PAYMENT_ID	Number	-	-	0	1	-
	AMOUNT	Number	-	-	-	-	✓
	PAYMENT_DATE	Date	7	-	-	-	✓
	PAYMENT_METHOD	Varchar2	50	-	-	-	✓
	CUSTOMER_ID	Number	-	-	0	-	✓

Constraint of Order

Since the entity titled-'Order' has a one-to-many relationship with the entities titled-'Customer' and 'Books'. So, the table-Order has 2 foreign keys titled-'Customer_ID' and 'Book_ID'.

```
ALTER TABLE "Order"
ADD Customer_ID INT
ADD CONSTRAINT fk_order_customer
FOREIGN KEY (Customer_ID)
REFERENCES Customer (Customer_ID);
```

```
ALTER TABLE "Order"
ADD Book_ID INT
ADD CONSTRAINT fk_book_order
FOREIGN KEY (Book_ID)
REFERENCES Books (Book_ID);
```



```
ALTER TABLE "Order"
ADD Customer_ID INT
ADD CONSTRAINT fk_order_customer
FOREIGN KEY (Customer_ID)
REFERENCES Customer (Customer_ID);

---RUN THIS CODE SEPERATELY---

ALTER TABLE "Order"
ADD Book_ID INT
ADD CONSTRAINT fk_book_order
FOREIGN KEY (Book_ID)
REFERENCES Books (Book_ID);


DESC "Order";
```

Object Type **TABLE** Object **Order**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable
Order	ORDER_ID	Number	-	-	0	1	-
	ORDER_DATE	Date	7	-	-	-	✓
	TOTAL_AMOUNT	Number	-	-	-	-	✓
	CUSTOMER_ID	Number	-	-	0	-	✓
	BOOK_ID	Number	-	-	0	-	✓

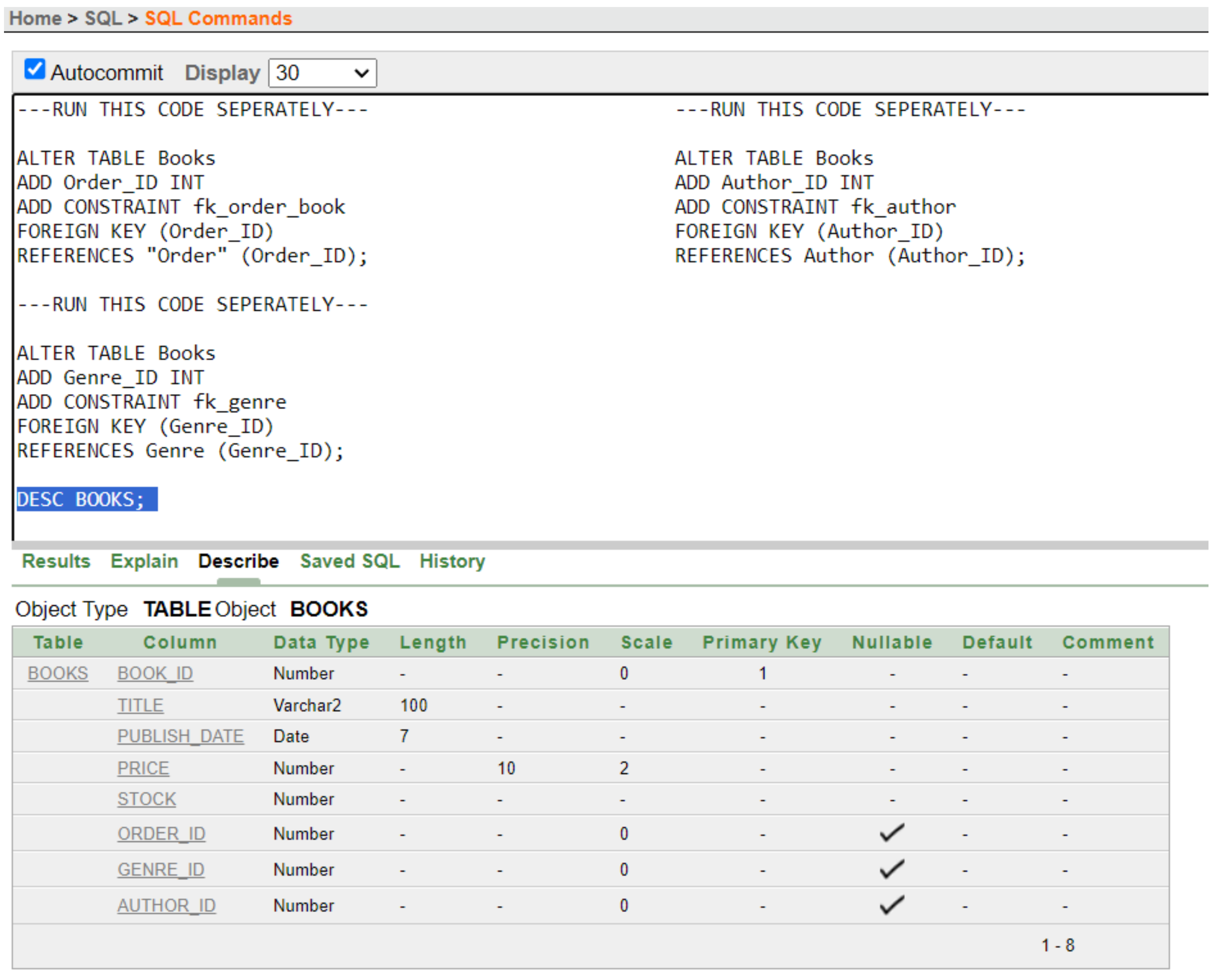
Constraint of Books

Since the entity titled-'Books' has a one-to-many relationship with the entities titled- 'Order', 'Author' and 'Genre'. So, the table-Books has 3 foreign keys titled- 'Order_ID', 'Author_ID' and 'Genre_ID'.

```
ALTER TABLE Books
ADD Order_ID INT
ADD CONSTRAINT fk_order_book
FOREIGN KEY (Order_ID)
REFERENCES "Order" (Order_ID);
```

```
ALTER TABLE Books
ADD Author_ID INT
ADD CONSTRAINT fk_author
FOREIGN KEY (Author_ID)
REFERENCES Author (Author_ID);
```

```
ALTER TABLE Books
ADD Genre_ID INT
ADD CONSTRAINT fk_genre
FOREIGN KEY (Genre_ID)
REFERENCES Genre (Genre_ID);
```


Home > SQL > SQL Commands
Autocommit Display 30
Results Explain Describe Saved SQL History

```
---RUN THIS CODE SEPERATELY---

ALTER TABLE Books
ADD Order_ID INT
ADD CONSTRAINT fk_order_book
FOREIGN KEY (Order_ID)
REFERENCES "Order" (Order_ID);

---RUN THIS CODE SEPERATELY---

ALTER TABLE Books
ADD Genre_ID INT
ADD CONSTRAINT fk_genre
FOREIGN KEY (Genre_ID)
REFERENCES Genre (Genre_ID);

DESC BOOKS;
```

```
---RUN THIS CODE SEPERATELY---

ALTER TABLE Books
ADD Author_ID INT
ADD CONSTRAINT fk_author
FOREIGN KEY (Author_ID)
REFERENCES Author (Author_ID);
```

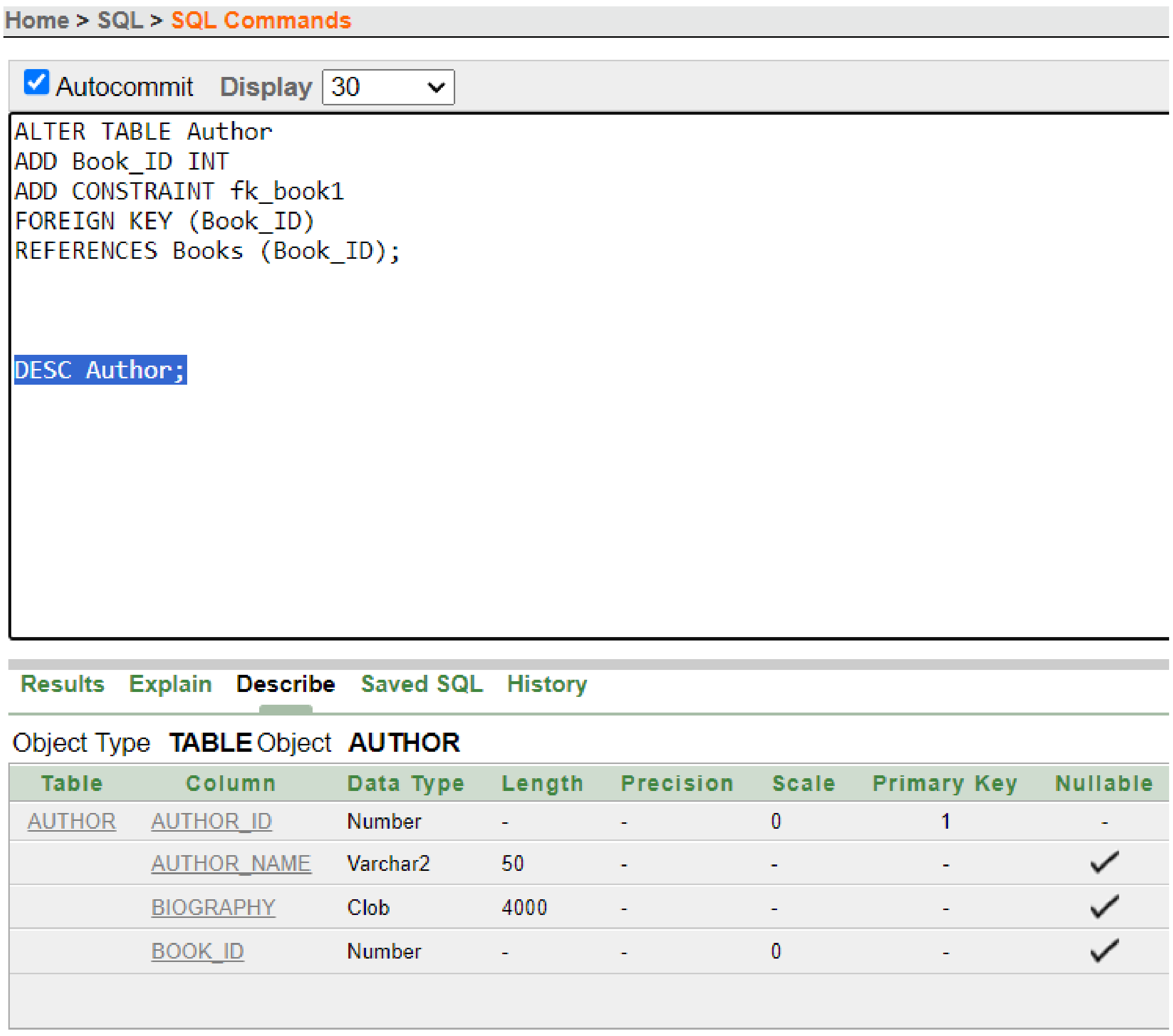
Object Type **TABLE** Object **BOOKS**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
BOOKS	BOOK_ID	Number	-	-	0	1	-	-	-
	TITLE	Varchar2	100	-	-	-	-	-	-
	PUBLISH_DATE	Date	7	-	-	-	-	-	-
	PRICE	Number	-	10	2	-	-	-	-
	STOCK	Number	-	-	-	-	-	-	-
	ORDER_ID	Number	-	-	0	-	✓	-	-
	GENRE_ID	Number	-	-	0	-	✓	-	-
	AUTHOR_ID	Number	-	-	0	-	✓	-	-

1 - 8

Constraint of Author

```
ALTER TABLE Author
ADD Book_ID INT
ADD CONSTRAINT fk_book1
FOREIGN KEY (Book_ID)
REFERENCES Books (Book_ID);
```


Home > SQL > SQL Commands
Autocommit Display 30
Results Explain Describe Saved SQL History

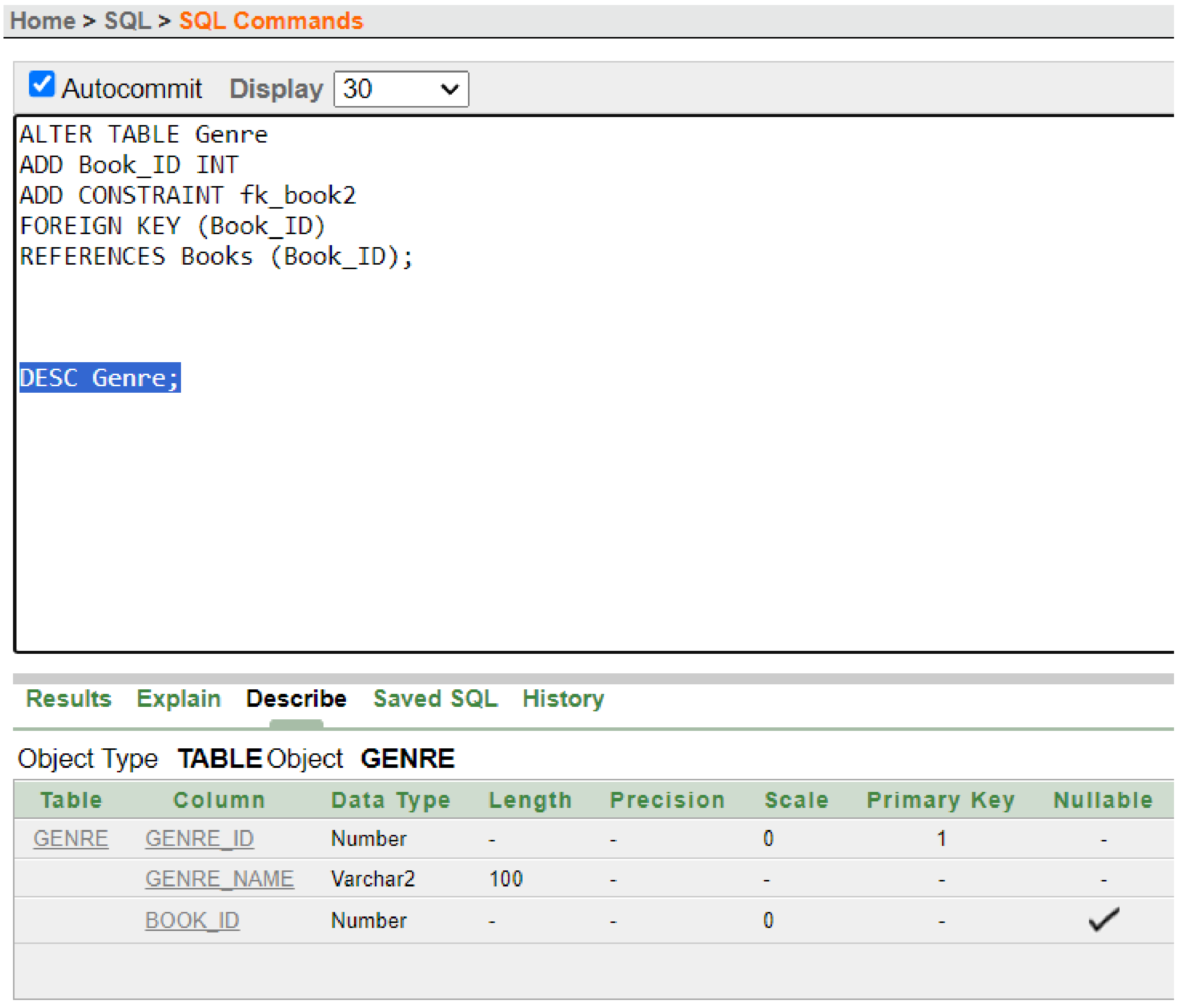
```
ALTER TABLE Author
ADD Book_ID INT
ADD CONSTRAINT fk_book1
FOREIGN KEY (Book_ID)
REFERENCES Books (Book_ID);



DESC Author;
```

Object Type **TABLE** Object **AUTHOR**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable
AUTHOR	AUTHOR_ID	Number	-	-	0	1	-
	AUTHOR_NAME	Varchar2	50	-	-	-	✓
	BIOGRAPHY	Clob	4000	-	-	-	✓
	BOOK_ID	Number	-	-	0	-	✓

Constraint of Genre

```
ALTER TABLE Genre
ADD Book_ID INT
ADD CONSTRAINT fk_book2
FOREIGN KEY (Book_ID)
REFERENCES Books (Book_ID);
```


Home > SQL > SQL Commands
Autocommit Display 30
Results Explain Describe Saved SQL History

```
ALTER TABLE Genre
ADD Book_ID INT
ADD CONSTRAINT fk_book2
FOREIGN KEY (Book_ID)
REFERENCES Books (Book_ID);



DESC Genre;
```

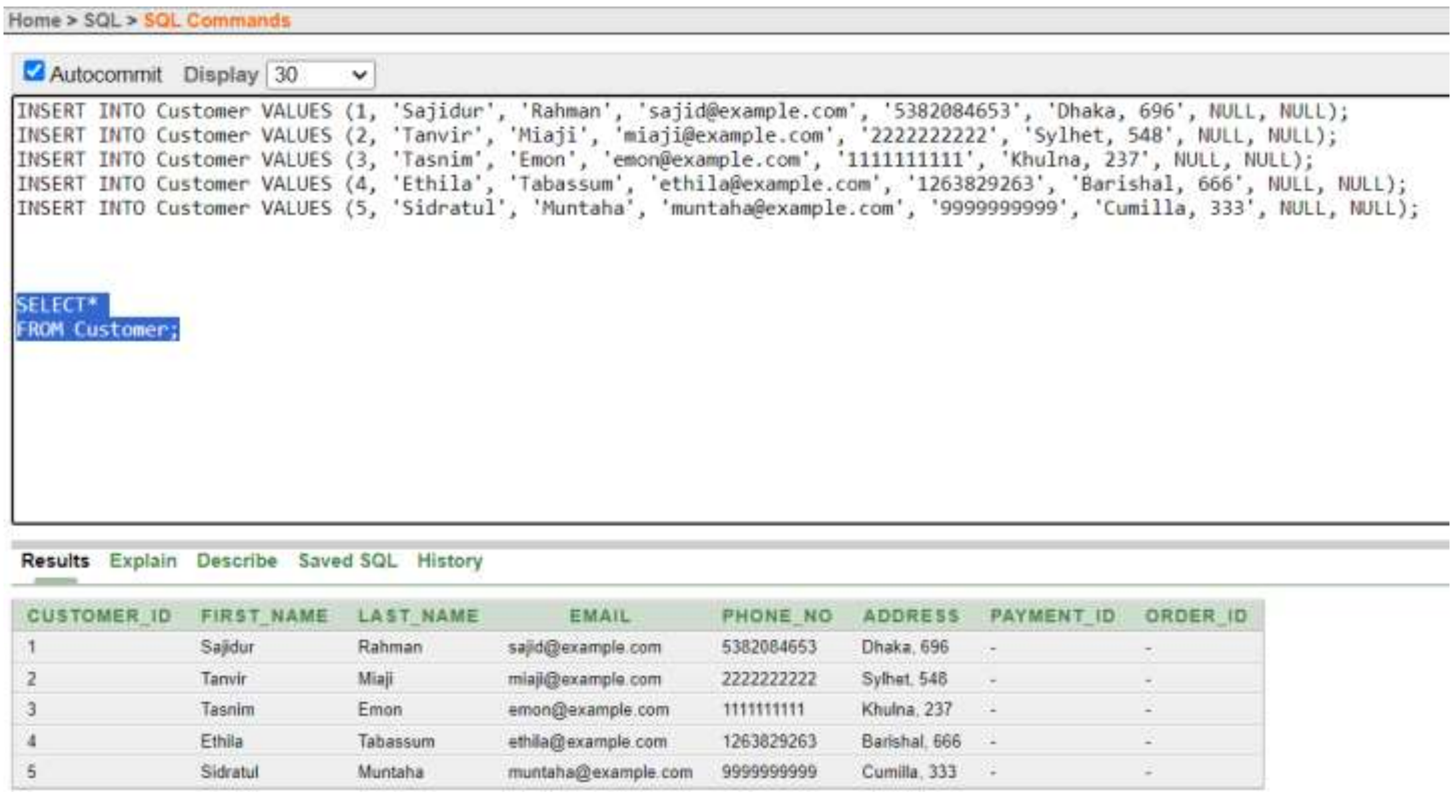
Object Type **TABLE** Object **GENRE**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable
GENRE	GENRE_ID	Number	-	-	0	1	-
	GENRE_NAME	Varchar2	100	-	-	-	-
	BOOK_ID	Number	-	-	0	-	✓

8 Data Insertion

Inserting Customer Data

INSERT INTO Customer VALUES (1, 'Sajidur', 'Rahman', 'sajid@example.com', '5382084653', 'Dhaka, 696', NULL, NULL);
INSERT INTO Customer VALUES (2, 'Tanvir', 'Miaji', 'miaji@example.com', '2222222222', 'Sylhet, 548', NULL, NULL);
INSERT INTO Customer VALUES (3, 'Tasnim', 'Emon', 'emon@example.com', '1111111111', 'Khulna, 237', NULL, NULL);
INSERT INTO Customer VALUES (4, 'Ethila', 'Tabassum', 'ethila@example.com', '1263829263', 'Barishal, 666', NULL, NULL);
INSERT INTO Customer VALUES (5, 'Sidratul', 'Muntaha', 'muntaha@example.com', '9999999999', 'Cumilla, 333', NULL, NULL);

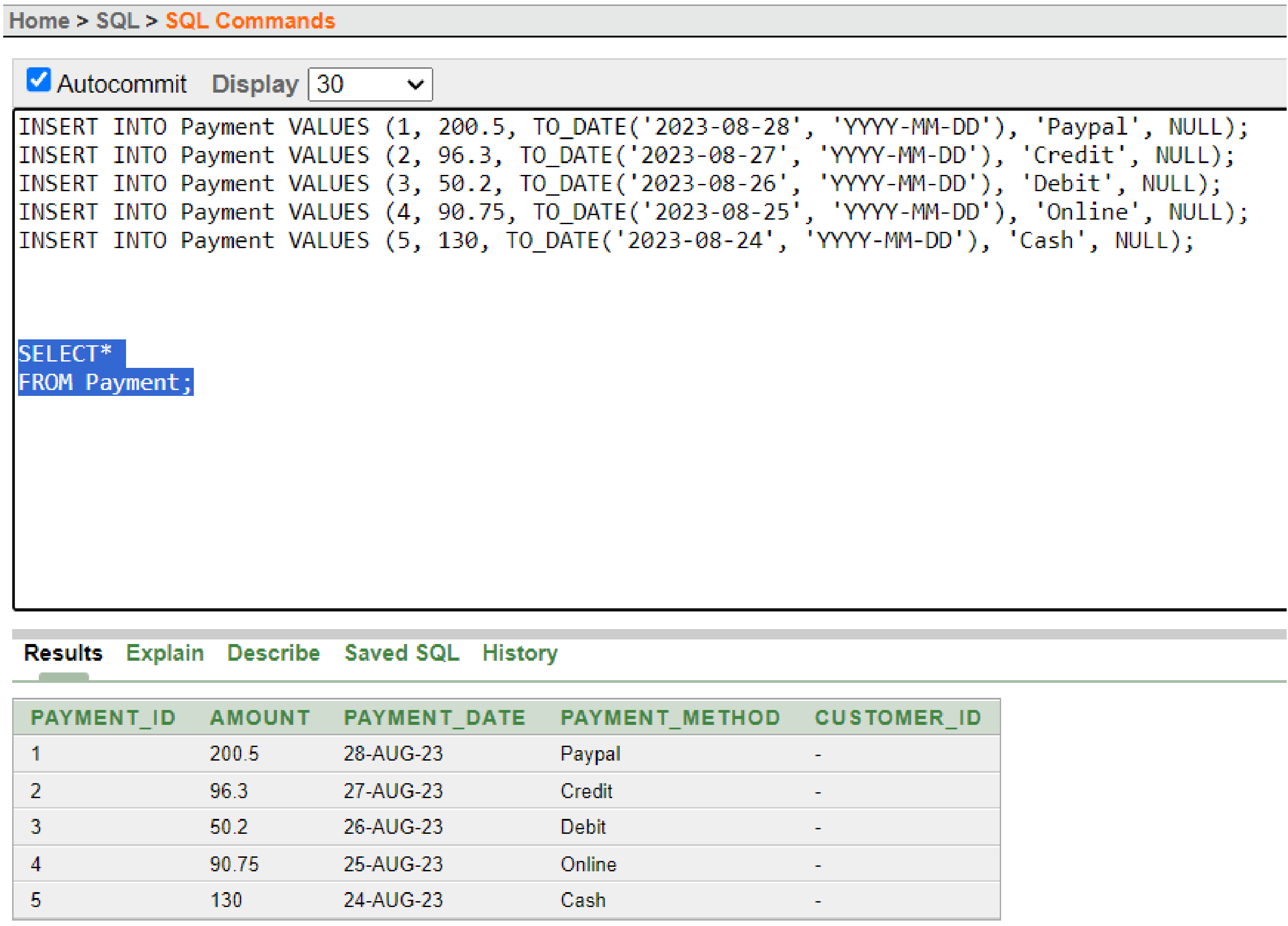


```
INSERT INTO Customer VALUES (1, 'Sajidur', 'Rahman', 'sajid@example.com', '5382084653', 'Dhaka, 696', NULL, NULL);
INSERT INTO Customer VALUES (2, 'Tanvir', 'Miaji', 'miaji@example.com', '2222222222', 'Sylhet, 548', NULL, NULL);
INSERT INTO Customer VALUES (3, 'Tasnim', 'Emon', 'emon@example.com', '1111111111', 'Khulna, 237', NULL, NULL);
INSERT INTO Customer VALUES (4, 'Ethila', 'Tabassum', 'ethila@example.com', '1263829263', 'Barishal, 666', NULL, NULL);
INSERT INTO Customer VALUES (5, 'Sidratul', 'Muntaha', 'muntaha@example.com', '9999999999', 'Cumilla, 333', NULL, NULL);


SELECT*
FROM Customer;
```

CUSTOMER_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NO	ADDRESS	PAYMENT_ID	ORDER_ID
1	Sajidur	Rahman	sajid@example.com	5382084653	Dhaka, 696	-	-
2	Tanvir	Miaji	miaji@example.com	2222222222	Sylhet, 548	-	-
3	Tasnim	Emon	emon@example.com	1111111111	Khulna, 237	-	-
4	Ethila	Tabassum	ethila@example.com	1263829263	Barishal, 666	-	-
5	Sidratul	Muntaha	muntaha@example.com	9999999999	Cumilla, 333	-	-

Inserting Payment Data

INSERT INTO Payment VALUES (1, 200.5, TO_DATE('2023-08-28', 'YYYY-MM-DD'), 'Paypal', NULL);
INSERT INTO Payment VALUES (2, 96.3, TO_DATE('2023-08-27', 'YYYY-MM-DD'), 'Credit', NULL);
INSERT INTO Payment VALUES (3, 50.2, TO_DATE('2023-08-26', 'YYYY-MM-DD'), 'Debit', NULL);
INSERT INTO Payment VALUES (4, 90.75, TO_DATE('2023-08-25', 'YYYY-MM-DD'), 'Online', NULL);
INSERT INTO Payment VALUES (5, 130, TO_DATE('2023-08-24', 'YYYY-MM-DD'), 'Cash', NULL);

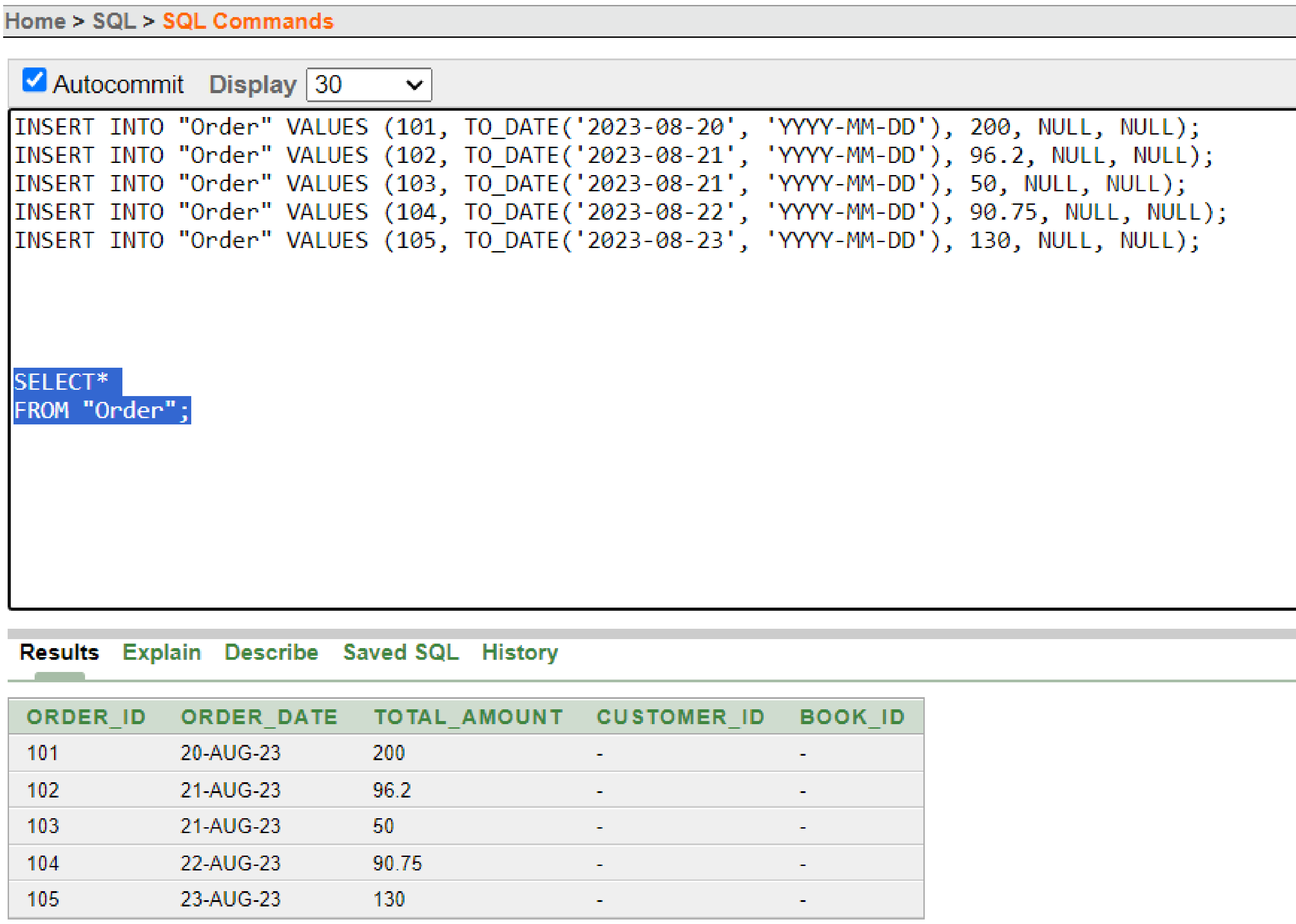


```
INSERT INTO Payment VALUES (1, 200.5, TO_DATE('2023-08-28', 'YYYY-MM-DD'), 'Paypal', NULL);
INSERT INTO Payment VALUES (2, 96.3, TO_DATE('2023-08-27', 'YYYY-MM-DD'), 'Credit', NULL);
INSERT INTO Payment VALUES (3, 50.2, TO_DATE('2023-08-26', 'YYYY-MM-DD'), 'Debit', NULL);
INSERT INTO Payment VALUES (4, 90.75, TO_DATE('2023-08-25', 'YYYY-MM-DD'), 'Online', NULL);
INSERT INTO Payment VALUES (5, 130, TO_DATE('2023-08-24', 'YYYY-MM-DD'), 'Cash', NULL);



SELECT*
FROM Payment;
```

PAYMENT_ID	AMOUNT	PAYMENT_DATE	PAYMENT_METHOD	CUSTOMER_ID
1	200.5	28-AUG-23	Paypal	-
2	96.3	27-AUG-23	Credit	-
3	50.2	26-AUG-23	Debit	-
4	90.75	25-AUG-23	Online	-
5	130	24-AUG-23	Cash	-

Inserting Order Data

INSERT INTO "Order" VALUES (101, TO_DATE('2023-08-20', 'YYYY-MM-DD'), 200, NULL, NULL);
INSERT INTO "Order" VALUES (102, TO_DATE('2023-08-21', 'YYYY-MM-DD'), 96.2, NULL, NULL);
INSERT INTO "Order" VALUES (103, TO_DATE('2023-08-21', 'YYYY-MM-DD'), 50, NULL, NULL);
INSERT INTO "Order" VALUES (104, TO_DATE('2023-08-22', 'YYYY-MM-DD'), 90.75, NULL, NULL);
INSERT INTO "Order" VALUES (105, TO_DATE('2023-08-23', 'YYYY-MM-DD'), 130, NULL, NULL);

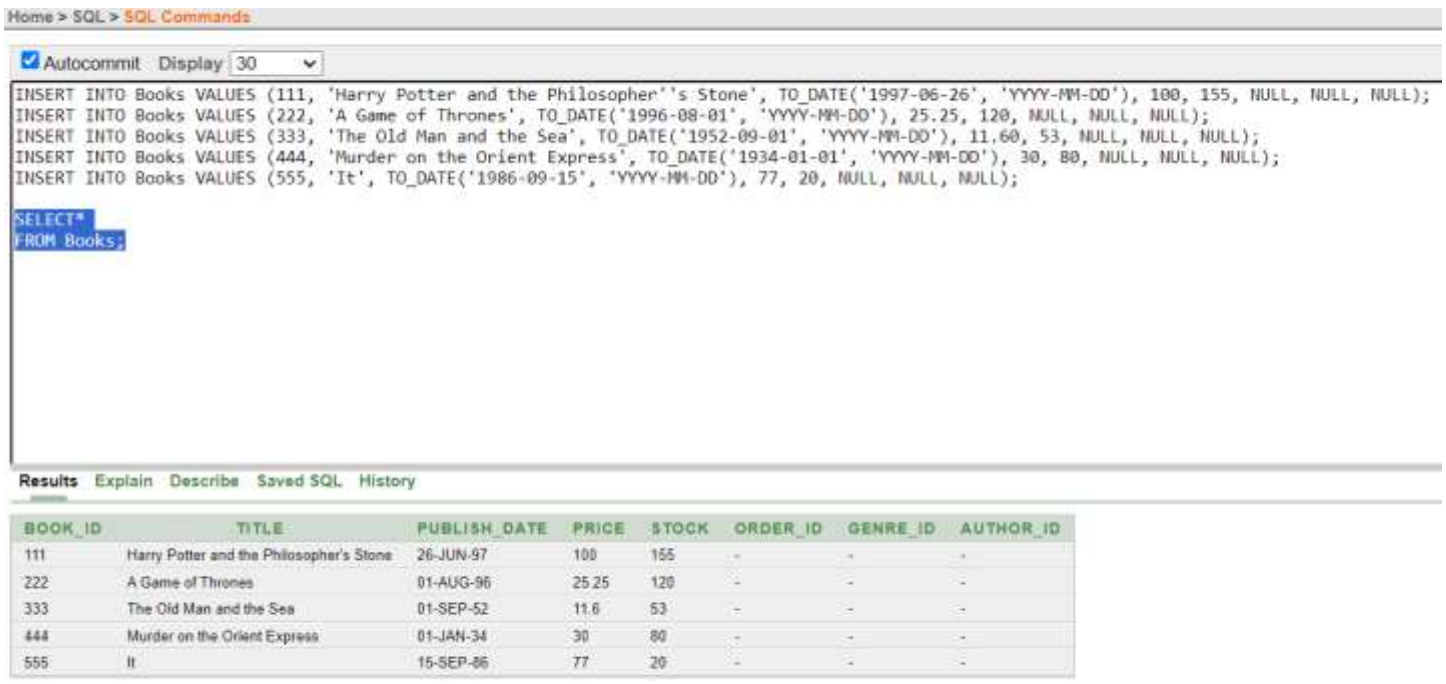


```
INSERT INTO "Order" VALUES (101, TO_DATE('2023-08-20', 'YYYY-MM-DD'), 200, NULL, NULL);
INSERT INTO "Order" VALUES (102, TO_DATE('2023-08-21', 'YYYY-MM-DD'), 96.2, NULL, NULL);
INSERT INTO "Order" VALUES (103, TO_DATE('2023-08-21', 'YYYY-MM-DD'), 50, NULL, NULL);
INSERT INTO "Order" VALUES (104, TO_DATE('2023-08-22', 'YYYY-MM-DD'), 90.75, NULL, NULL);
INSERT INTO "Order" VALUES (105, TO_DATE('2023-08-23', 'YYYY-MM-DD'), 130, NULL, NULL);



SELECT*
FROM "Order";
```

ORDER_ID	ORDER_DATE	TOTAL_AMOUNT	CUSTOMER_ID	BOOK_ID
101	20-AUG-23	200	-	-
102	21-AUG-23	96.2	-	-
103	21-AUG-23	50	-	-
104	22-AUG-23	90.75	-	-
105	23-AUG-23	130	-	-

Inserting Books Data

```
INSERT INTO Books VALUES (111, 'Harry Potter and the Philosopher''s Stone',
TO_DATE('1997-06-26', 'YYYY-MM-DD'), 100, 155, NULL, NULL , NULL);
INSERT INTO Books VALUES (222, 'A Game of Thrones', TO_DATE('1996-08-01', 'YYYY-MM-
DD'), 25.25, 120, NULL, NULL, NULL);
INSERT INTO Books VALUES (333, 'The Old Man and the Sea', TO_DATE('1952-09-01', 'YYYY-
MM-DD'), 11.60, 53, NULL, NULL, NULL);
INSERT INTO Books VALUES (444, 'Murder on the Orient Express', TO_DATE('1934-01-01',
'YYYY-MM-DD'), 30, 80, NULL, NULL, NULL);
INSERT INTO Books VALUES (555, 'It', TO_DATE('1986-09-15', 'YYYY-MM-DD'), 77, 20, NULL,
NULL, NULL);
```



```
INSERT INTO Books VALUES (111, 'Harry Potter and the Philosopher''s Stone', TO_DATE('1997-06-26', 'YYYY-MM-DD'), 100, 155, NULL, NULL, NULL);
INSERT INTO Books VALUES (222, 'A Game of Thrones', TO_DATE('1996-08-01', 'YYYY-MM-DD'), 25.25, 120, NULL, NULL, NULL);
INSERT INTO Books VALUES (333, 'The Old Man and the Sea', TO_DATE('1952-09-01', 'YYYY-MM-DD'), 11.60, 53, NULL, NULL, NULL);
INSERT INTO Books VALUES (444, 'Murder on the Orient Express', TO_DATE('1934-01-01', 'YYYY-MM-DD'), 30, 80, NULL, NULL, NULL);
INSERT INTO Books VALUES (555, 'It', TO_DATE('1986-09-15', 'YYYY-MM-DD'), 77, 20, NULL, NULL, NULL);

SELECT*
FROM Books;
```

BOOK_ID	TITLE	PUBLISH_DATE	PRICE	STOCK	ORDER_ID	GENRE_ID	AUTHOR_ID
111	Harry Potter and the Philosopher's Stone	26-JUN-97	100	155	-	-	-
222	A Game of Thrones	01-AUG-96	25.25	120	-	-	-
333	The Old Man and the Sea	01-SEP-52	11.6	53	-	-	-
444	Murder on the Orient Express	01-JAN-34	30	80	-	-	-
555	It	15-SEP-86	77	20	-	-	-

Inserting Author Data

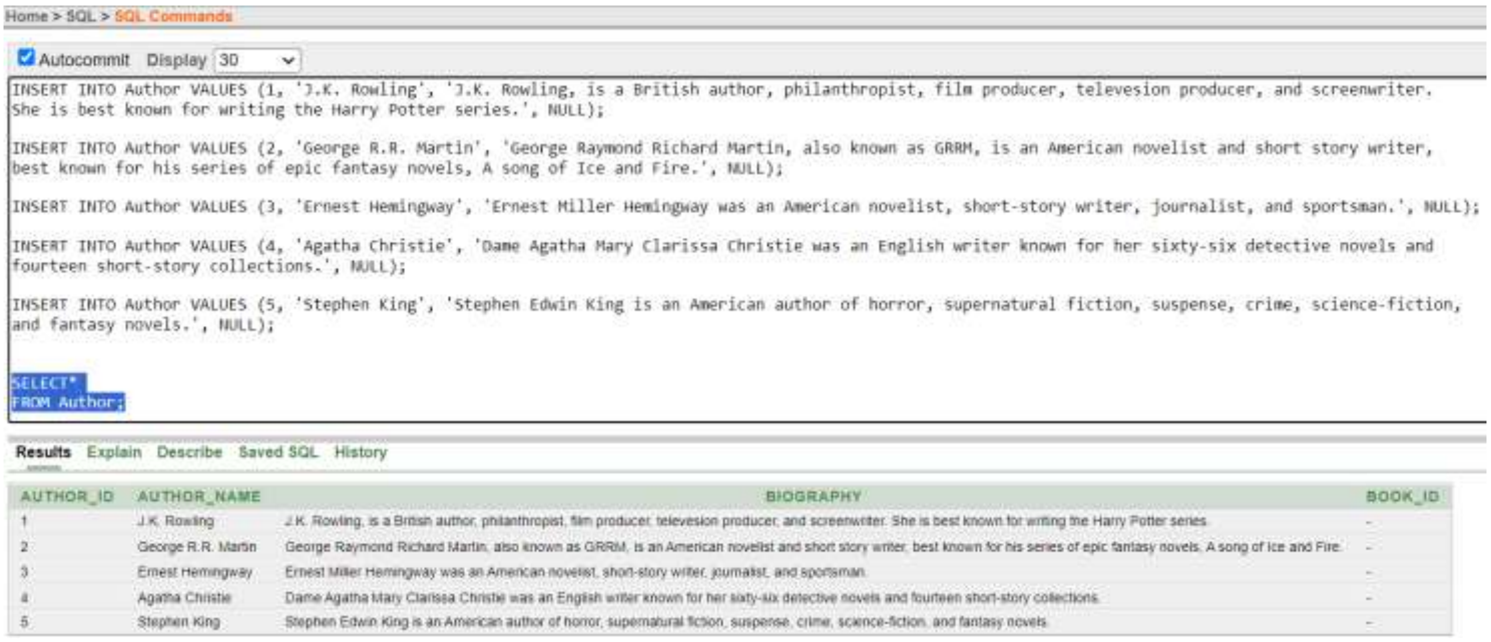
INSERT INTO Author VALUES (1, 'J.K. Rowling', 'J.K. Rowling, is a British author, philanthropist, film producer,
televesion producer, and screenwriter.She is best known for writing the Harry Potter series.', NULL);

INSERT INTO Author VALUES (2, 'George R.R. Martin', 'George Raymond Richard Martin, also known as GRRM, is an
American novelist and short story writer,best known for his series of epic fantasy novels, A song of Ice and Fire.', NULL);

INSERT INTO Author VALUES (3, 'Ernest Hemingway', 'Ernest Miller Hemingway was an American novelist, short-story writer,
journalist, and sportsman.', NULL);

INSERT INTO Author VALUES (4, 'Agatha Christie', 'Dame Agatha Mary Clarissa Christie was an English writer known
for her sixty-six detective novels and fourteen short-story collections.', NULL);

INSERT INTO Author VALUES (5, 'Stephen King', 'Stephen Edwin King is an American author of horror,
supernatural fiction, suspense, crime, science-fiction,and fantasy novels.', NULL);

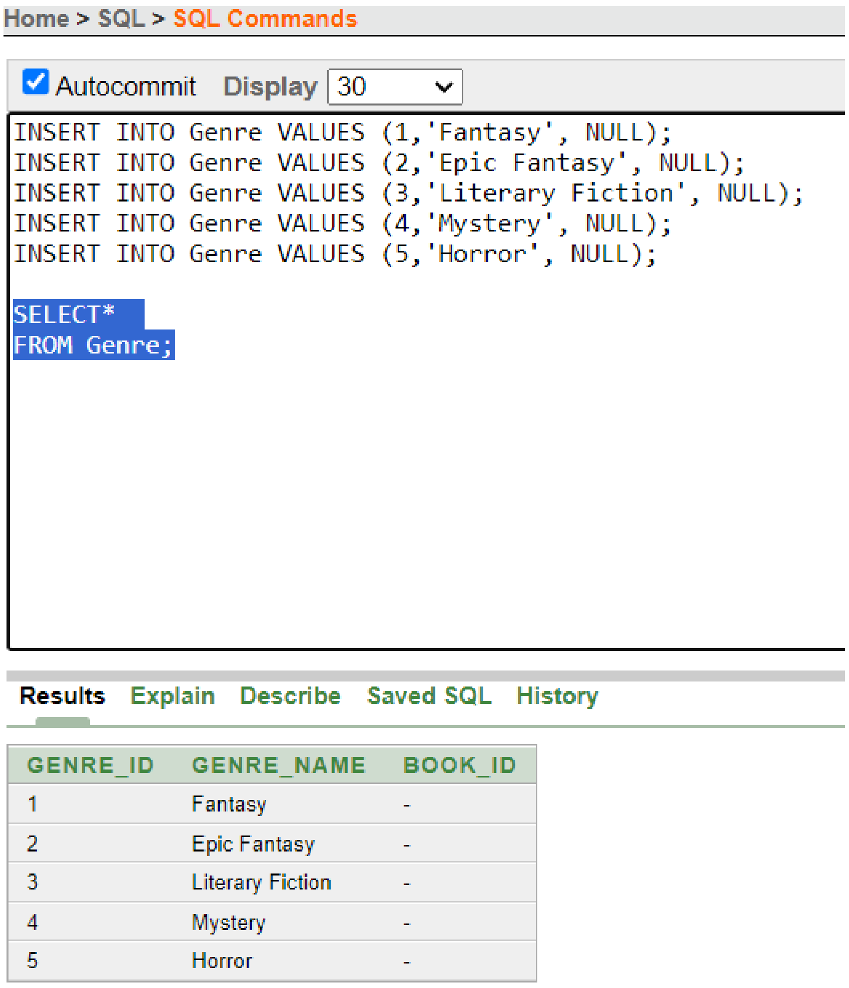


```
INSERT INTO Author VALUES (1, 'J.K. Rowling', 'J.K. Rowling, is a British author, philanthropist, film producer, televesion producer, and screenwriter.
She is best known for writing the Harry Potter series.', NULL);

INSERT INTO Author VALUES (2, 'George R.R. Martin', 'George Raymond Richard Martin, also known as GRRM, is an American novelist and short story writer,
best known for his series of epic fantasy novels, A song of Ice and Fire.', NULL);

INSERT INTO Author VALUES (3, 'Ernest Hemingway', 'Ernest Miller Hemingway was an American novelist, short-story writer, journalist, and sportsman.', NULL);

INSERT INTO Author VALUES (4, 'Agatha Christie', 'Dame Agatha Mary Clarissa Christie was an English writer known for her sixty-six detective novels and
fourteen short-story collections.', NULL);

INSERT INTO Author VALUES (5, 'Stephen King', 'Stephen Edwin King is an American author of horror, supernatural fiction, suspense, crime, science-fiction,
and fantasy novels.', NULL);


SELECT*
FROM Author;
```

AUTHOR_ID	AUTHOR_NAME	BIOGRAPHY	BOOK_ID
1	J.K. Rowling	J.K. Rowling, is a British author, philanthropist, film producer, televesion producer, and screenwriter. She is best known for writing the Harry Potter series.	-
2	George R.R. Martin	George Raymond Richard Martin, also known as GRRM, is an American novelist and short story writer, best known for his series of epic fantasy novels, A song of Ice and Fire.	-
3	Ernest Hemingway	Ernest Miller Hemingway was an American novelist, short-story writer, journalist, and sportsman.	-
4	Agatha Christie	Dame Agatha Mary Clarissa Christie was an English writer known for her sixty-six detective novels and fourteen short-story collections.	-
5	Stephen King	Stephen Edwin King is an American author of horror, supernatural fiction, suspense, crime, science-fiction, and fantasy novels.	-

Inserting Genre Data

```
INSERT INTO Genre VALUES (1,'Fantasy', NULL);
INSERT INTO Genre VALUES (2,'Epic Fantasy', NULL);
INSERT INTO Genre VALUES (3,'Literary Fiction', NULL);
INSERT INTO Genre VALUES (4,'Mystery', NULL);
INSERT INTO Genre VALUES (5,'Horror', NULL);
```


Home > SQL > SQL Commands
Autocommit Display 30
Results Explain Describe Saved SQL History

```
INSERT INTO Genre VALUES (1,'Fantasy', NULL);
INSERT INTO Genre VALUES (2,'Epic Fantasy', NULL);
INSERT INTO Genre VALUES (3,'Literary Fiction', NULL);
INSERT INTO Genre VALUES (4,'Mystery', NULL);
INSERT INTO Genre VALUES (5,'Horror', NULL);

SELECT*
FROM Genre;
```

GENRE_ID	GENRE_NAME	BOOK_ID
1	Fantasy	-
2	Epic Fantasy	-
3	Literary Fiction	-
4	Mystery	-
5	Horror	-

8.1 Data Updating

Updating Customer Data

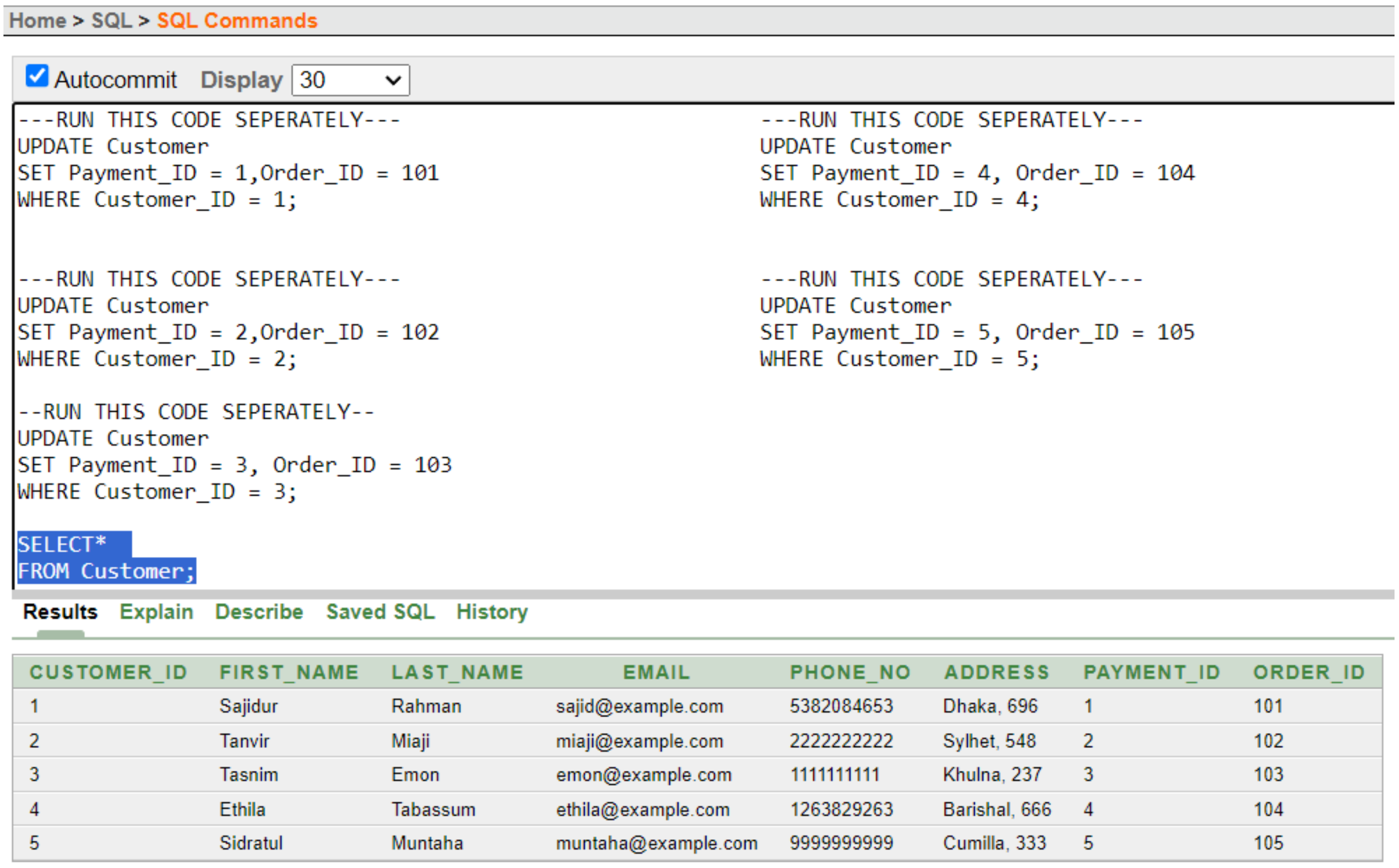
```
UPDATE Customer
SET Payment_ID = 1, Order_ID = 101
WHERE Customer_ID = 1;
```

```
UPDATE Customer
SET Payment_ID = 2, Order_ID = 102
WHERE Customer_ID = 2;
```

```
UPDATE Customer
SET Payment_ID = 3, Order_ID = 103
WHERE Customer_ID = 3;
```

```
UPDATE Customer
SET Payment_ID = 4, Order_ID = 104
WHERE Customer_ID = 4;
```

```
UPDATE Customer
SET Payment_ID = 5, Order_ID = 105
WHERE Customer_ID = 5;
```


Home > SQL > SQL Commands

Autocommit Display 30

```
---RUN THIS CODE SEPERATELY---
UPDATE Customer
SET Payment_ID = 1,Order_ID = 101
WHERE Customer_ID = 1;


---RUN THIS CODE SEPERATELY---
UPDATE Customer
SET Payment_ID = 2,Order_ID = 102
WHERE Customer_ID = 2;

--RUN THIS CODE SEPERATELY--
UPDATE Customer
SET Payment_ID = 3, Order_ID = 103
WHERE Customer_ID = 3;

SELECT*
FROM Customer;
```

```
---RUN THIS CODE SEPERATELY---
UPDATE Customer
SET Payment_ID = 4, Order_ID = 104
WHERE Customer_ID = 4;


---RUN THIS CODE SEPERATELY---
UPDATE Customer
SET Payment_ID = 5, Order_ID = 105
WHERE Customer_ID = 5;
```

Results Explain Describe Saved SQL History

CUSTOMER_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NO	ADDRESS	PAYMENT_ID	ORDER_ID
1	Sajidur	Rahman	sajid@example.com	5382084653	Dhaka, 696	1	101
2	Tanvir	Miaji	miaji@example.com	2222222222	Sylhet, 548	2	102
3	Tasnim	Emon	emon@example.com	1111111111	Khulna, 237	3	103
4	Ethila	Tabassum	ethila@example.com	1263829263	Barishal, 666	4	104
5	Sidratul	Muntaha	muntaha@example.com	9999999999	Cumilla, 333	5	105

Updating Order Data

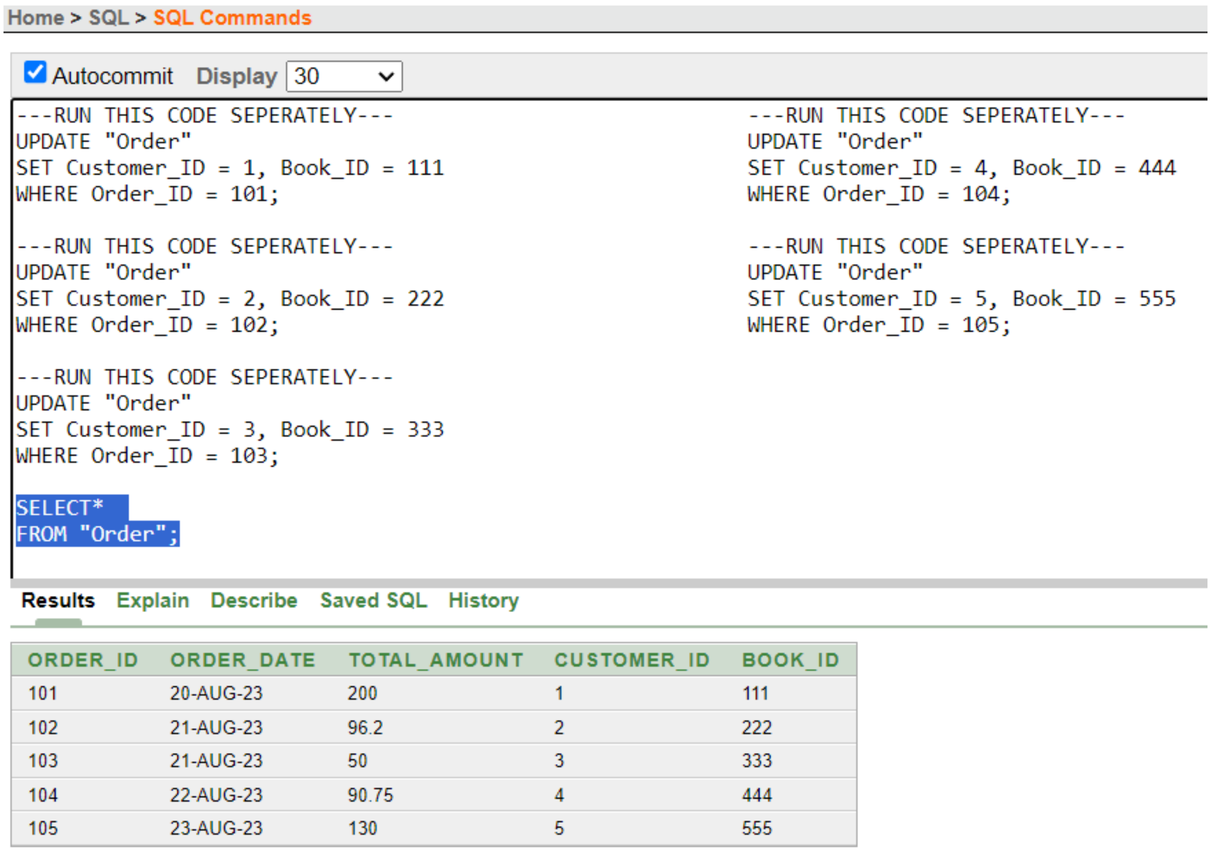
```
UPDATE "Order"
SET Customer_ID = 1, Book_ID = 111
WHERE Order_ID = 101;
```

```
UPDATE "Order"
SET Customer_ID = 2, Book_ID = 222
WHERE Order_ID = 102;
```

```
UPDATE "Order"
SET Customer_ID = 3, Book_ID = 333
WHERE Order_ID = 103;
```

```
UPDATE "Order"
SET Customer_ID = 4, Book_ID = 444
WHERE Order_ID = 104;
```

```
UPDATE "Order"
SET Customer_ID = 5, Book_ID = 555
WHERE Order_ID = 105;
```



```
---RUN THIS CODE SEPERATELY---
UPDATE "Order"
SET Customer_ID = 1, Book_ID = 111
WHERE Order_ID = 101;

---RUN THIS CODE SEPERATELY---
UPDATE "Order"
SET Customer_ID = 2, Book_ID = 222
WHERE Order_ID = 102;

---RUN THIS CODE SEPERATELY---
UPDATE "Order"
SET Customer_ID = 3, Book_ID = 333
WHERE Order_ID = 103;

SELECT*
FROM "Order";

---RUN THIS CODE SEPERATELY---
UPDATE "Order"
SET Customer_ID = 4, Book_ID = 444
WHERE Order_ID = 104;

---RUN THIS CODE SEPERATELY---
UPDATE "Order"
SET Customer_ID = 5, Book_ID = 555
WHERE Order_ID = 105;
```

ORDER_ID	ORDER_DATE	TOTAL_AMOUNT	CUSTOMER_ID	BOOK_ID
101	20-AUG-23	200	1	111
102	21-AUG-23	96.2	2	222
103	21-AUG-23	50	3	333
104	22-AUG-23	90.75	4	444
105	23-AUG-23	130	5	555

Updating Payment Data

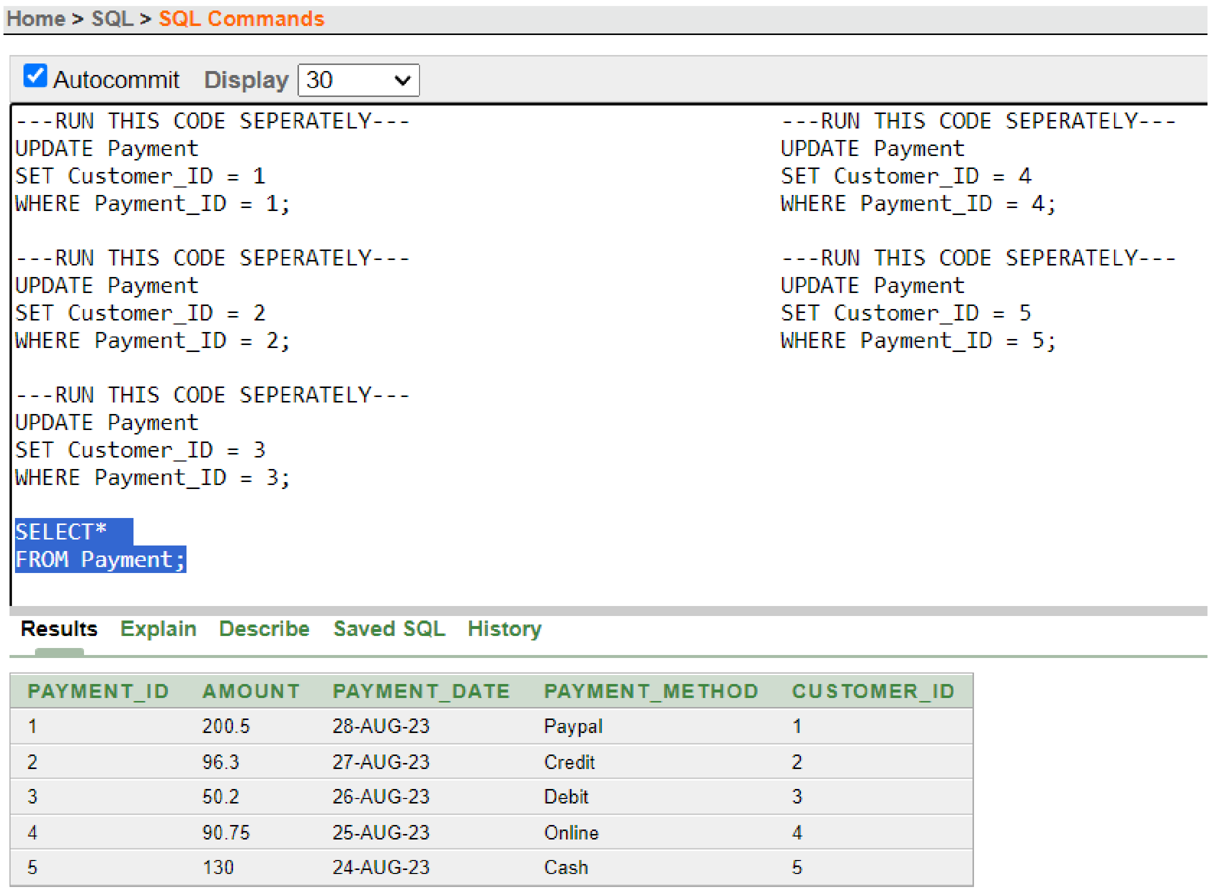
```
UPDATE Payment
SET Customer_ID = 1
WHERE Payment_ID = 1;
```

```
UPDATE Payment
SET Customer_ID = 2
WHERE Payment_ID = 2;
```

```
UPDATE Payment
SET Customer_ID = 3
WHERE Payment_ID = 3;
```

```
UPDATE Payment
SET Customer_ID = 4
WHERE Payment_ID = 4;
```

```
UPDATE Payment
SET Customer_ID = 5
WHERE Payment_ID = 5;
```


Home > SQL > SQL Commands
Autocommit Display 30
Results Explain Describe Saved SQL History

```
---RUN THIS CODE SEPERATELY---
UPDATE Payment
SET Customer_ID = 1
WHERE Payment_ID = 1;

---RUN THIS CODE SEPERATELY---
UPDATE Payment
SET Customer_ID = 2
WHERE Payment_ID = 2;

---RUN THIS CODE SEPERATELY---
UPDATE Payment
SET Customer_ID = 3
WHERE Payment_ID = 3;

SELECT*
FROM Payment;

---RUN THIS CODE SEPERATELY---
UPDATE Payment
SET Customer_ID = 4
WHERE Payment_ID = 4;

---RUN THIS CODE SEPERATELY---
UPDATE Payment
SET Customer_ID = 5
WHERE Payment_ID = 5;
```

PAYMENT_ID	AMOUNT	PAYMENT_DATE	PAYMENT_METHOD	CUSTOMER_ID
1	200.5	28-AUG-23	Paypal	1
2	96.3	27-AUG-23	Credit	2
3	50.2	26-AUG-23	Debit	3
4	90.75	25-AUG-23	Online	4
5	130	24-AUG-23	Cash	5

Updating Books Data

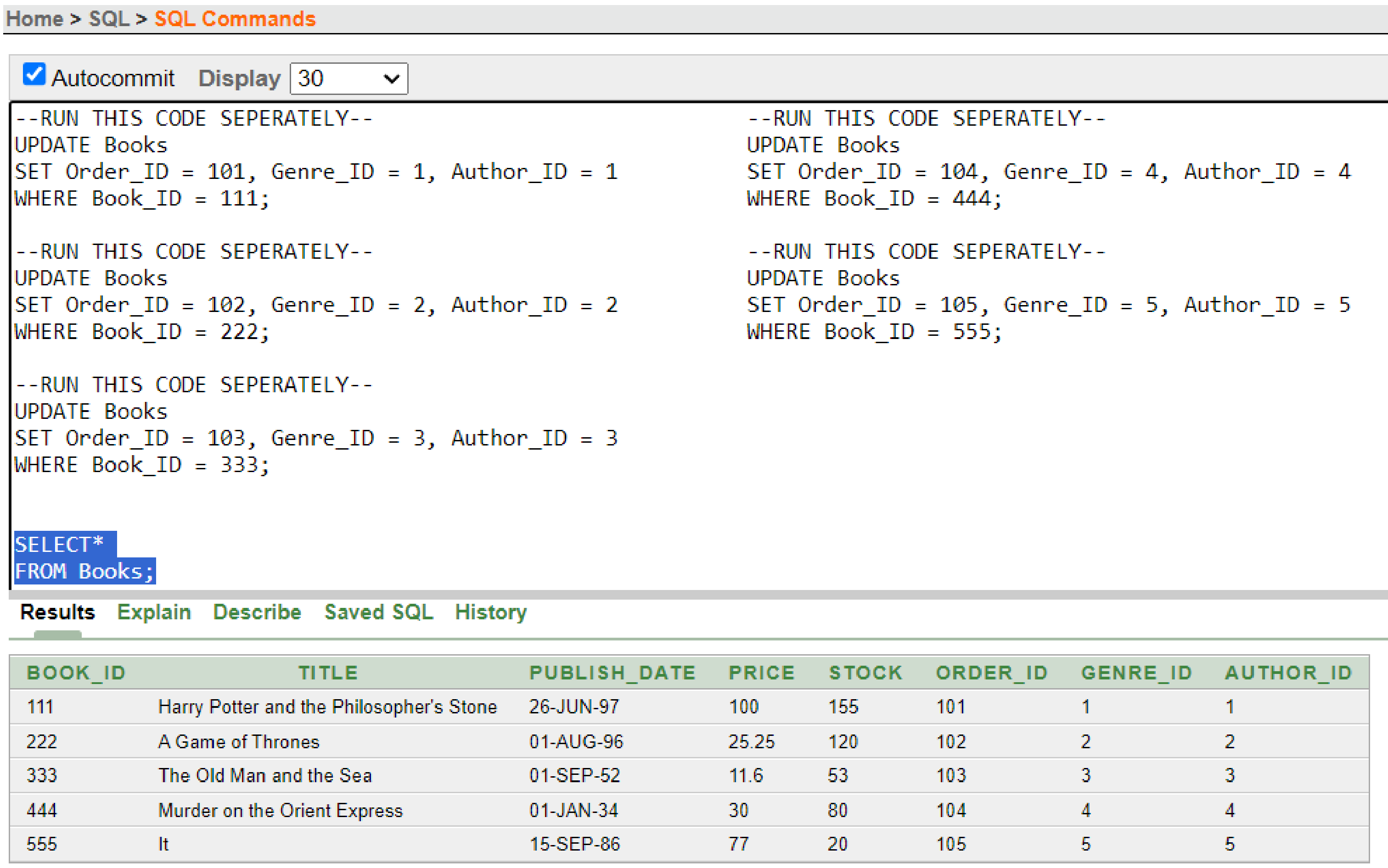
```
UPDATE Books
SET Order_ID = 101, Genre_ID = 1, Author_ID = 1
WHERE Book_ID = 111;
```

```
UPDATE Books
SET Order_ID = 102, Genre_ID = 2, Author_ID = 2
WHERE Book_ID = 222;
```

```
UPDATE Books
SET Order_ID = 103, Genre_ID = 3, Author_ID = 3
WHERE Book_ID = 333;
```

```
UPDATE Books
SET Order_ID = 104, Genre_ID = 4, Author_ID = 4
WHERE Book_ID = 444;
```

```
UPDATE Books
SET Order_ID = 105, Genre_ID = 5, Author_ID = 5
WHERE Book_ID = 555;
```



```
--RUN THIS CODE SEPERATELY--
UPDATE Books
SET Order_ID = 101, Genre_ID = 1, Author_ID = 1
WHERE Book_ID = 111;

--RUN THIS CODE SEPERATELY--
UPDATE Books
SET Order_ID = 102, Genre_ID = 2, Author_ID = 2
WHERE Book_ID = 222;

--RUN THIS CODE SEPERATELY--
UPDATE Books
SET Order_ID = 103, Genre_ID = 3, Author_ID = 3
WHERE Book_ID = 333;

--RUN THIS CODE SEPERATELY--
UPDATE Books
SET Order_ID = 104, Genre_ID = 4, Author_ID = 4
WHERE Book_ID = 444;

--RUN THIS CODE SEPERATELY--
UPDATE Books
SET Order_ID = 105, Genre_ID = 5, Author_ID = 5
WHERE Book_ID = 555;


SELECT*
FROM Books;
```

BOOK_ID	TITLE	PUBLISH_DATE	PRICE	STOCK	ORDER_ID	GENRE_ID	AUTHOR_ID
111	Harry Potter and the Philosopher's Stone	26-JUN-97	100	155	101	1	1
222	A Game of Thrones	01-AUG-96	25.25	120	102	2	2
333	The Old Man and the Sea	01-SEP-52	11.6	53	103	3	3
444	Murder on the Orient Express	01-JAN-34	30	80	104	4	4
555	It	15-SEP-86	77	20	105	5	5

Updating Author Data

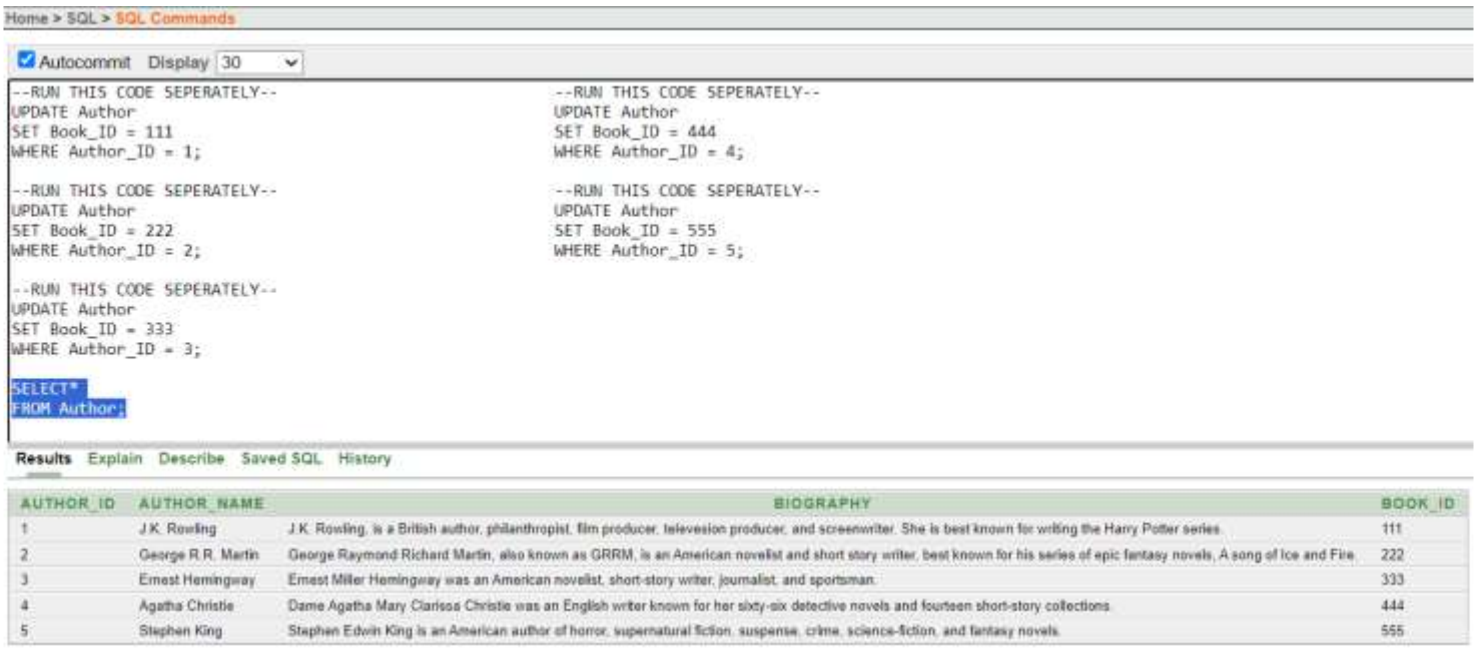
```
UPDATE Author
SET Book_ID = 111
WHERE Author_ID = 1;
```

```
UPDATE Author
SET Book_ID = 222
WHERE Author_ID = 2;
```

```
UPDATE Author
SET Book_ID = 333
WHERE Author_ID = 3;
```

```
UPDATE Author
SET Book_ID = 444
WHERE Author_ID = 4;
```

```
UPDATE Author
SET Book_ID = 555
WHERE Author_ID = 5;
```


Home > SQL > SQL Commands
Autocommit Display 30
Results Explain Describe Saved SQL History

```
--RUN THIS CODE SEPERATELY--
UPDATE Author
SET Book_ID = 111
WHERE Author_ID = 1;

--RUN THIS CODE SEPERATELY--
UPDATE Author
SET Book_ID = 222
WHERE Author_ID = 2;

--RUN THIS CODE SEPERATELY--
UPDATE Author
SET Book_ID = 333
WHERE Author_ID = 3;

SELECT*
FROM Author;

--RUN THIS CODE SEPERATELY--
UPDATE Author
SET Book_ID = 444
WHERE Author_ID = 4;

--RUN THIS CODE SEPERATELY--
UPDATE Author
SET Book_ID = 555
WHERE Author_ID = 5;
```

AUTHOR_ID	AUTHOR_NAME	BIOGRAPHY	BOOK_ID
1	J.K. Rowling	J.K. Rowling, is a British author, philanthropist, film producer, televesion producer, and screenwriter. She is best known for writing the Harry Potter series.	111
2	George R.R. Martin	George Raymond Richard Martin, also known as GRRM, is an American novelist and short story writer, best known for his series of epic fantasy novels, A song of Ice and Fire.	222
3	Ernest Hemingway	Ernest Miller Hemingway was an American novelist, short-story writer, journalist, and sportsman.	333
4	Agatha Christie	Dame Agatha Mary Clarissa Christie was an English writer known for her sixty-six detective novels and fourteen short-story collections.	444
5	Stephen King	Stephen Edwin King is an American author of horror, supernatural fiction, suspense, crime, science-fiction, and fantasy novels.	555

Updating Genre Data

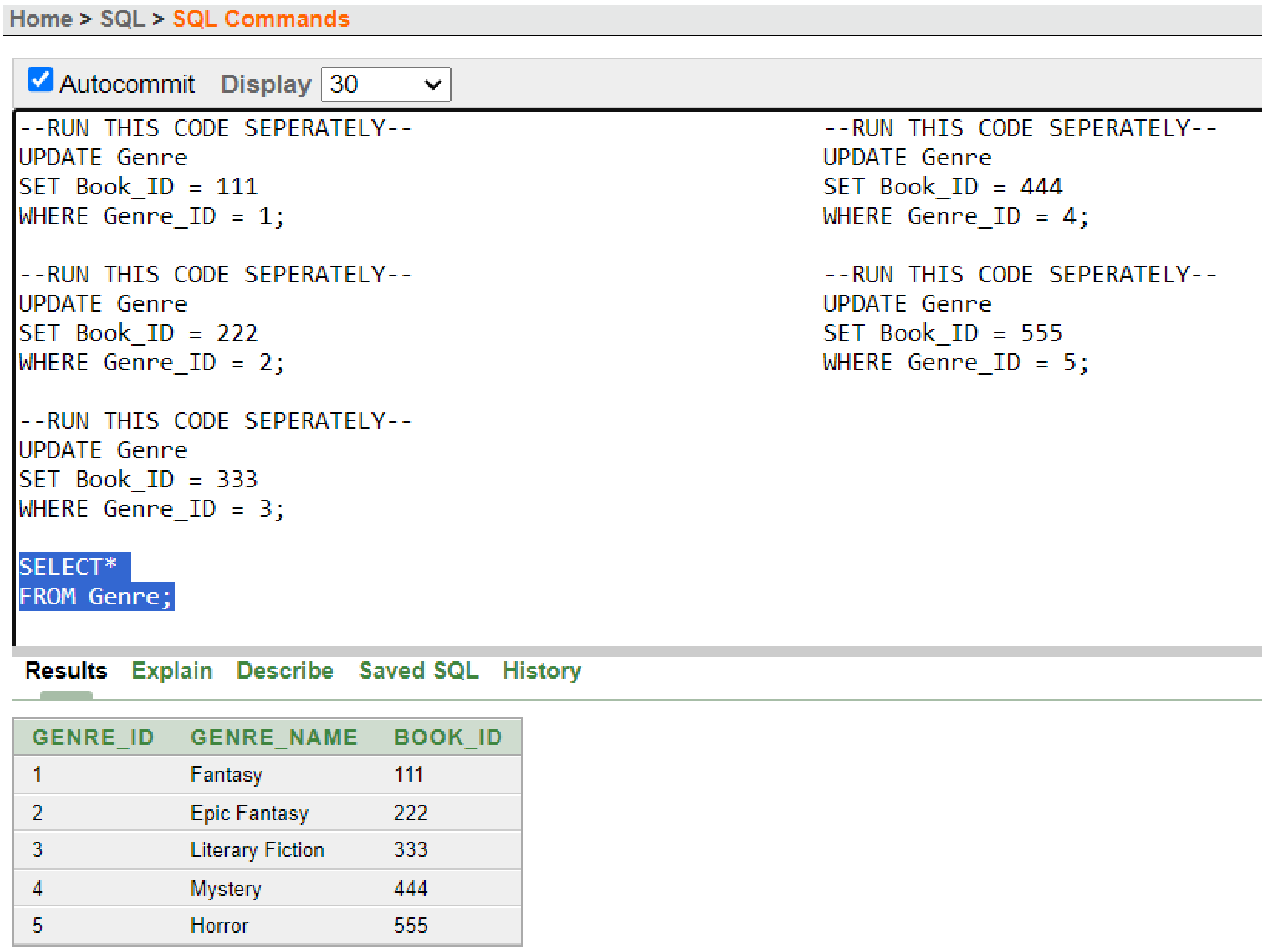
```
UPDATE Genre
SET Book_ID = 111
WHERE Genre_ID = 1;
```

```
UPDATE Genre
SET Book_ID = 222
WHERE Genre_ID = 2;
```

```
UPDATE Genre
SET Book_ID = 333
WHERE Genre_ID = 3;
```

```
UPDATE Genre
SET Book_ID = 444
WHERE Genre_ID = 4;
```

```
UPDATE Genre
SET Book_ID = 555
WHERE Genre_ID = 5;
```



```
--RUN THIS CODE SEPERATELY--
UPDATE Genre
SET Book_ID = 111
WHERE Genre_ID = 1;

--RUN THIS CODE SEPERATELY--
UPDATE Genre
SET Book_ID = 222
WHERE Genre_ID = 2;

--RUN THIS CODE SEPERATELY--
UPDATE Genre
SET Book_ID = 333
WHERE Genre_ID = 3;

SELECT*
FROM Genre;
```

```
--RUN THIS CODE SEPERATELY--
UPDATE Genre
SET Book_ID = 444
WHERE Genre_ID = 4;

--RUN THIS CODE SEPERATELY--
UPDATE Genre
SET Book_ID = 555
WHERE Genre_ID = 5;
```

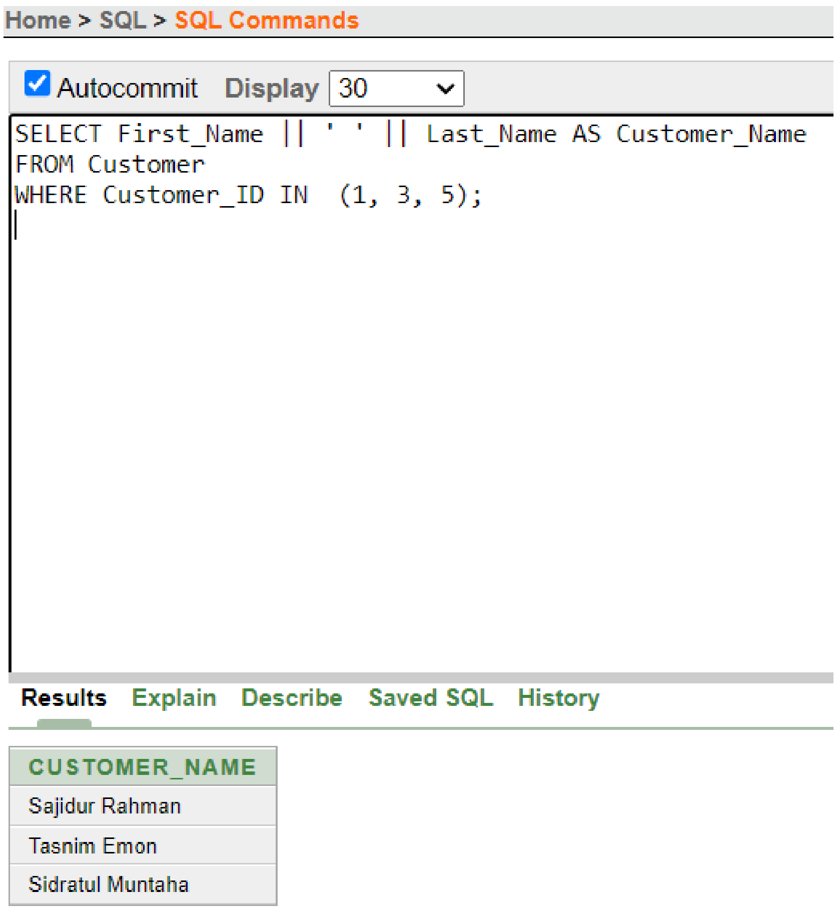
GENRE_ID	GENRE_NAME	BOOK_ID
1	Fantasy	111
2	Epic Fantasy	222
3	Literary Fiction	333
4	Mystery	444
5	Horror	555

9 Query Writing

➢ 9.1 Single-row function

1. Get the full name of customers based on their ‘Customer_ID’.

ANS : SELECT First_Name || ' ' || Last_Name AS Customer_Name
FROM Customer
WHERE Customer_ID IN (1, 3, 5);

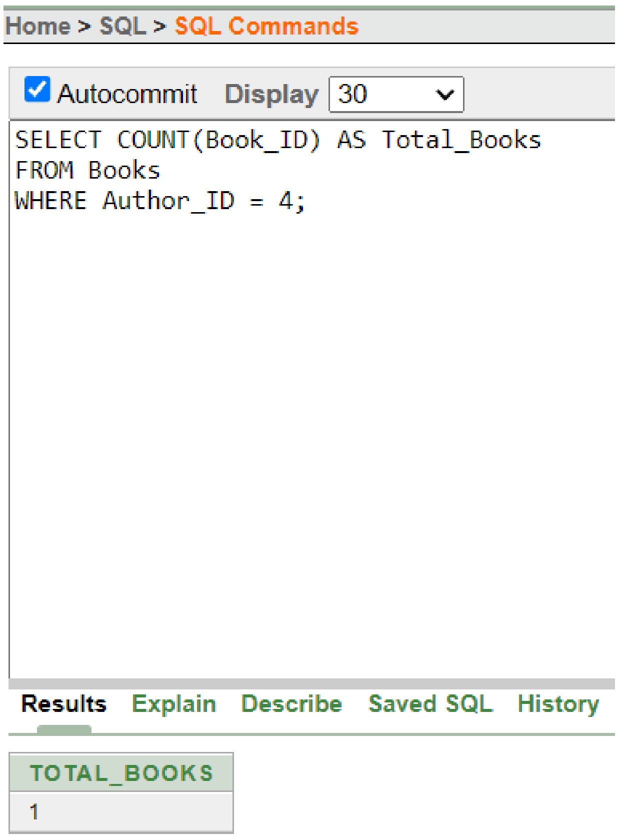

Home > SQL > SQL Commands
Autocommit Display 30
Results Explain Describe Saved SQL History

```
SELECT First_Name || ' ' || Last_Name AS Customer_Name
FROM Customer
WHERE Customer_ID IN  (1, 3, 5);
```

CUSTOMER_NAME
Sajidur Rahman
Tasnim Emon
Sidratul Muntaha

2. Get the total number of books written by an author based on their ‘Author_ID’.

ANS : SELECT COUNT(Book_ID) AS Total_Books
FROM Books
WHERE Author_ID = 4;



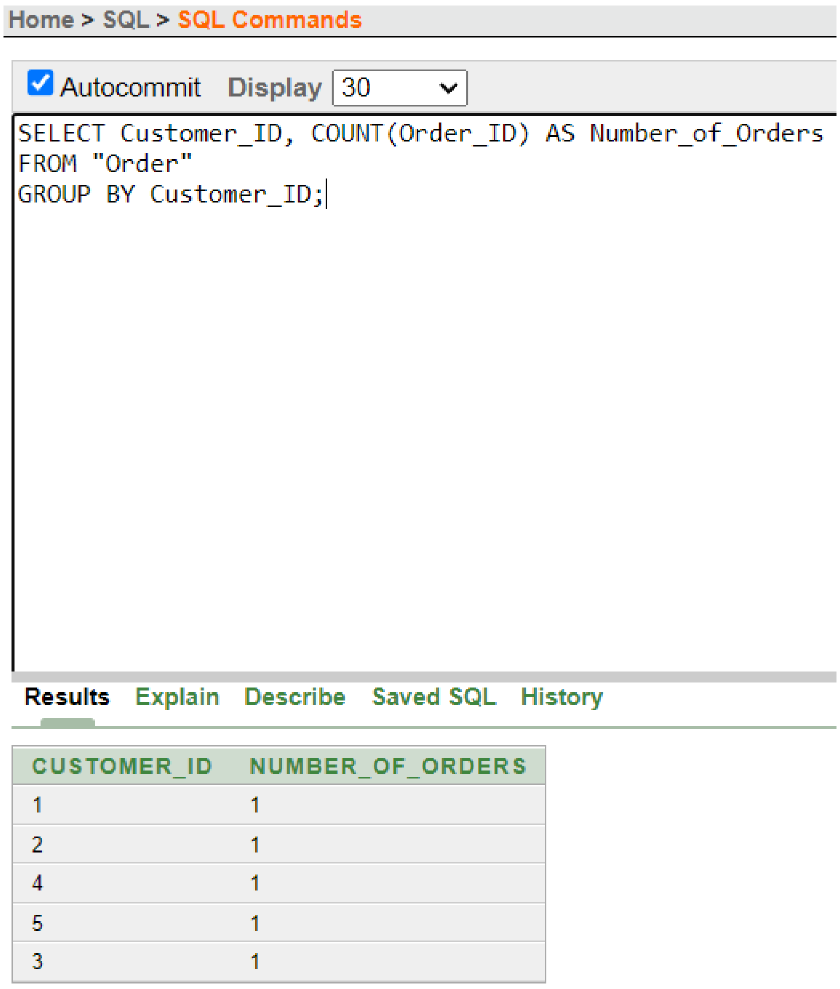
```
SELECT COUNT(Book_ID) AS Total_Books
FROM Books
WHERE Author_ID = 4;
```

```
TOTAL_BOOKS
1
```

➢ 9.2 Group function

1. Count the number of orders made by each customer.

ANS : SELECT Customer_ID, COUNT(Order_ID) AS Number_of_Orders
FROM "Order"
GROUP BY Customer_ID;

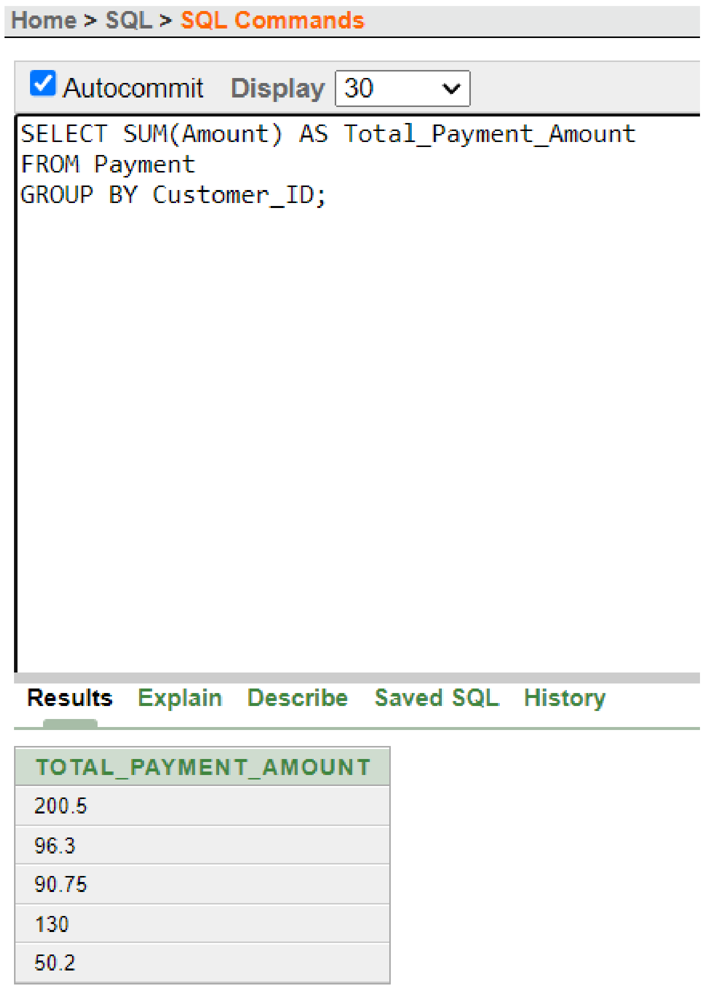


```
SELECT Customer_ID, COUNT(Order_ID) AS Number_of_Orders
FROM "Order"
GROUP BY Customer_ID;
```

CUSTOMER_ID	NUMBER_OF_ORDERS
1	1
2	1
4	1
5	1
3	1

2. Calculate the total payment amount made by all customers.

ANS : SELECT SUM(Amount) AS Total_Payment_Amount
FROM Payment
GROUP BY Customer_ID;



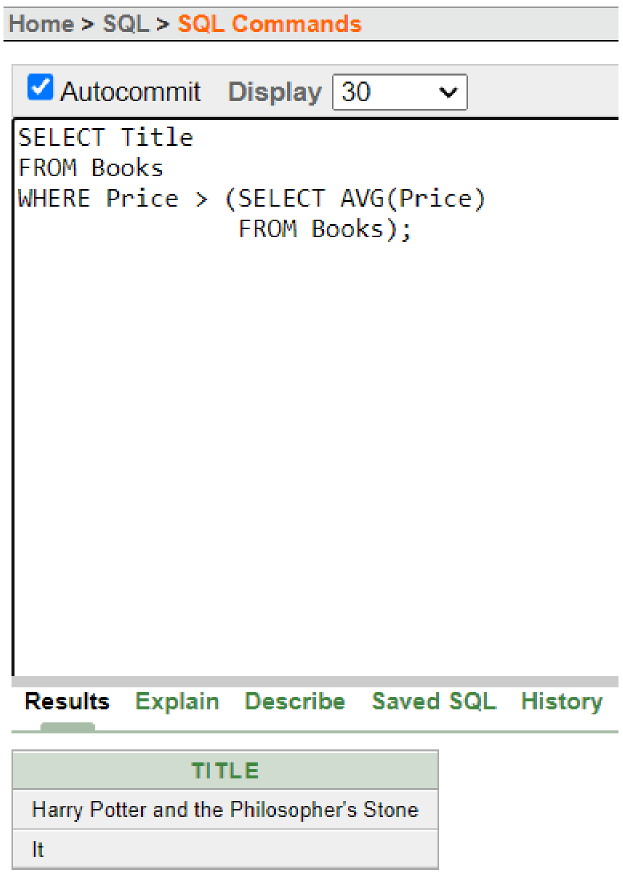
```
SELECT SUM(Amount) AS Total_Payment_Amount
FROM Payment
GROUP BY Customer_ID;
```

TOTAL_PAYMENT_AMOUNT
200.5
96.3
90.75
130
50.2

➢ 9.3 Subquery

1. Retrieve the titles of books that are priced higher than the average book price.

ANS :
```
SELECT Title
FROM Books
WHERE Price > (SELECT AVG(Price)
               FROM Books);
```

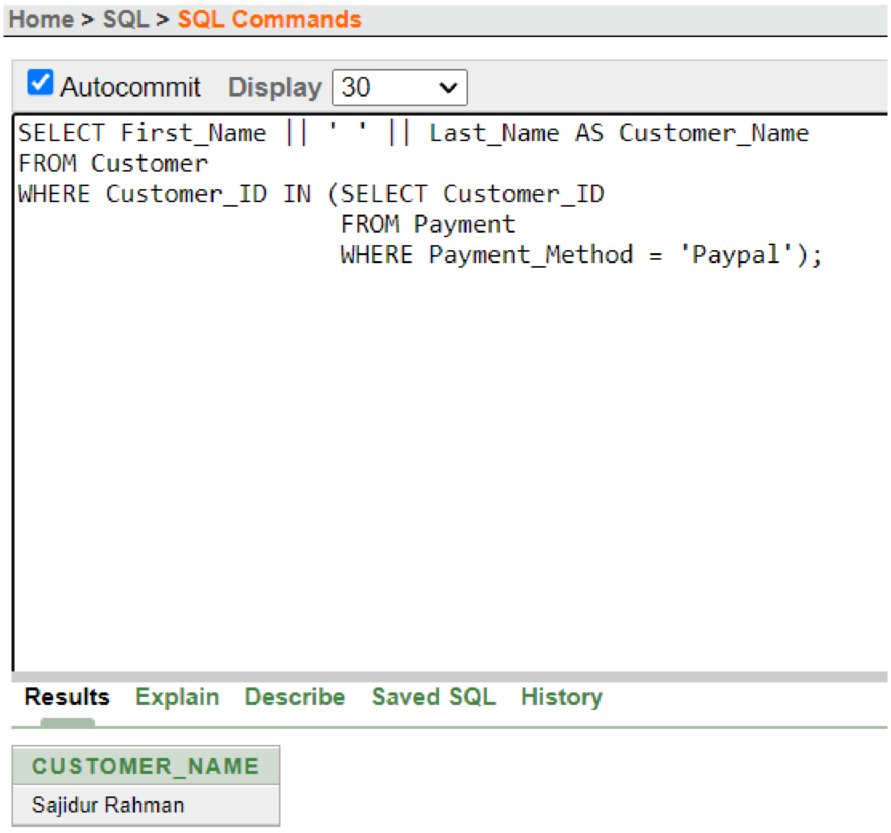


```
SELECT Title
FROM Books
WHERE Price > (SELECT AVG(Price)
               FROM Books);
```

TITLE
Harry Potter and the Philosopher's Stone
It

2. Retrieve the names of customers who have made payments using Paypal.

ANS :
```
SELECT First_Name || ' ' || Last_Name AS Customer_Name
FROM Customer
WHERE Customer_ID IN (SELECT Customer_ID
                      FROM Payment
                      WHERE Payment_Method = 'Paypal');
```


Home > SQL > SQL Commands
Autocommit Display 30
Results Explain Describe Saved SQL History

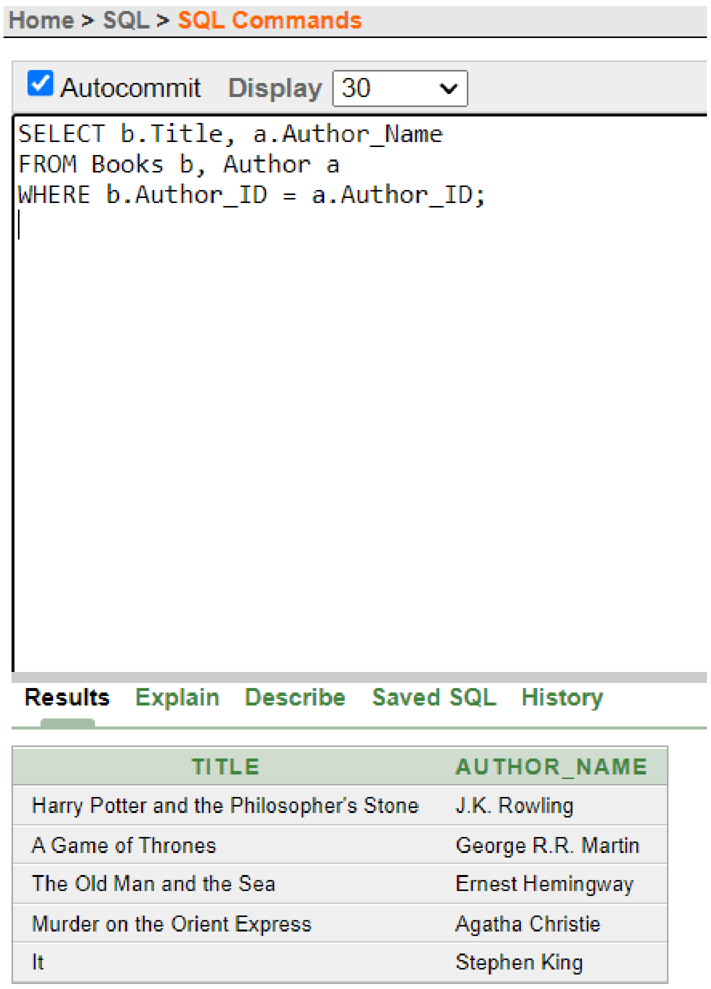
```
SELECT First_Name || ' ' || Last_Name AS Customer_Name
FROM Customer
WHERE Customer_ID IN (SELECT Customer_ID
                      FROM Payment
                      WHERE Payment_Method = 'Paypal');
```

CUSTOMER_NAME
Sajidur Rahman

➢ 9.4 Joining

1. Retrieve the titles of books along with the corresponding author names.

ANS :
```
SELECT b.Title, a.Author_Name
FROM Books b, Author a
WHERE b.Author_ID = a.Author_ID;
```

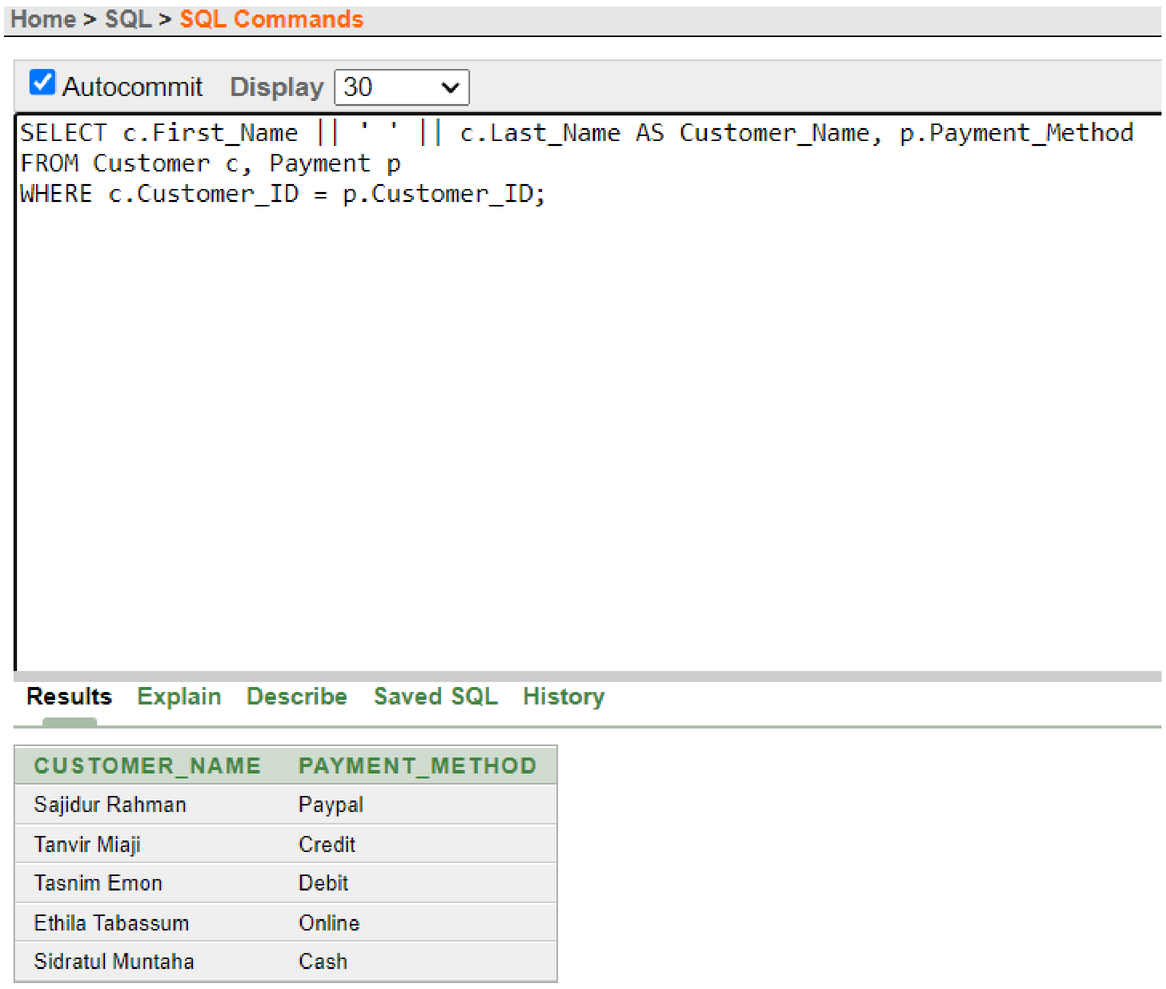


```
SELECT b.Title, a.Author_Name
FROM Books b, Author a
WHERE b.Author_ID = a.Author_ID;
```

TITLE	AUTHOR_NAME
Harry Potter and the Philosopher's Stone	J.K. Rowling
A Game of Thrones	George R.R. Martin
The Old Man and the Sea	Ernest Hemingway
Murder on the Orient Express	Agatha Christie
It	Stephen King

2. Retrieve customer names and their corresponding payment methods.

ANS :
```
SELECT c.First_Name || ' ' || c.Last_Name AS Customer_Name,
                                                   p.Payment_Method
FROM Customer c, Payment p
WHERE c.Customer_ID = p.Customer_ID;
```



```
SELECT c.First_Name || ' ' || c.Last_Name AS Customer_Name, p.Payment_Method
FROM Customer c, Payment p
WHERE c.Customer_ID = p.Customer_ID;
```

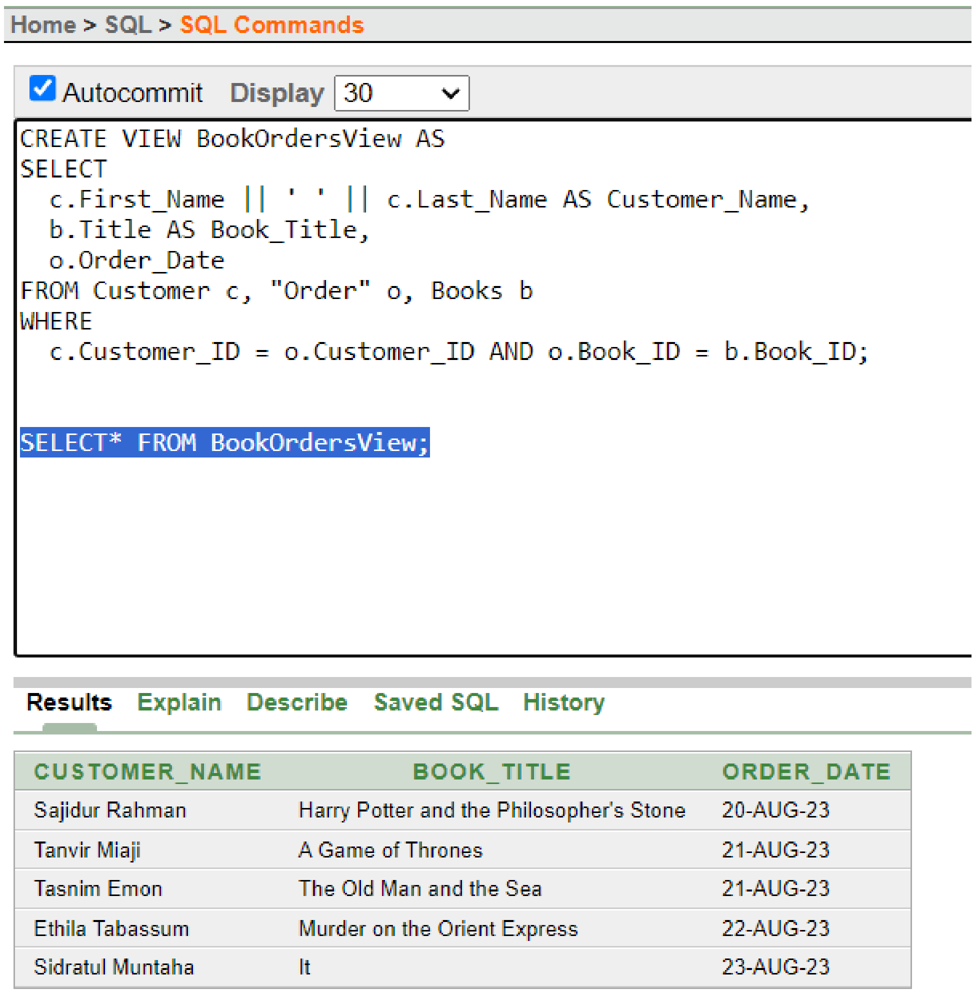
CUSTOMER_NAME	PAYMENT_METHOD
Sajidur Rahman	Paypal
Tanvir Miaji	Credit
Tasnim Emon	Debit
Ethila Tabassum	Online
Sidratul Muntaha	Cash

➢ 9.5 View

1. Create a view that shows details about book orders, including customer names, book titles and order dates.

ANS :

```
CREATE VIEW BookOrdersView AS
SELECT
c.First_Name || ' ' || c.Last_Name AS Customer_Name,
b.Title AS Book_Title,
o.Order_Date
FROM  Customer c, "Order" o, Books b
WHERE
c.Customer_ID = o.Customer_ID AND o.Book_ID = b.Book_ID;
```

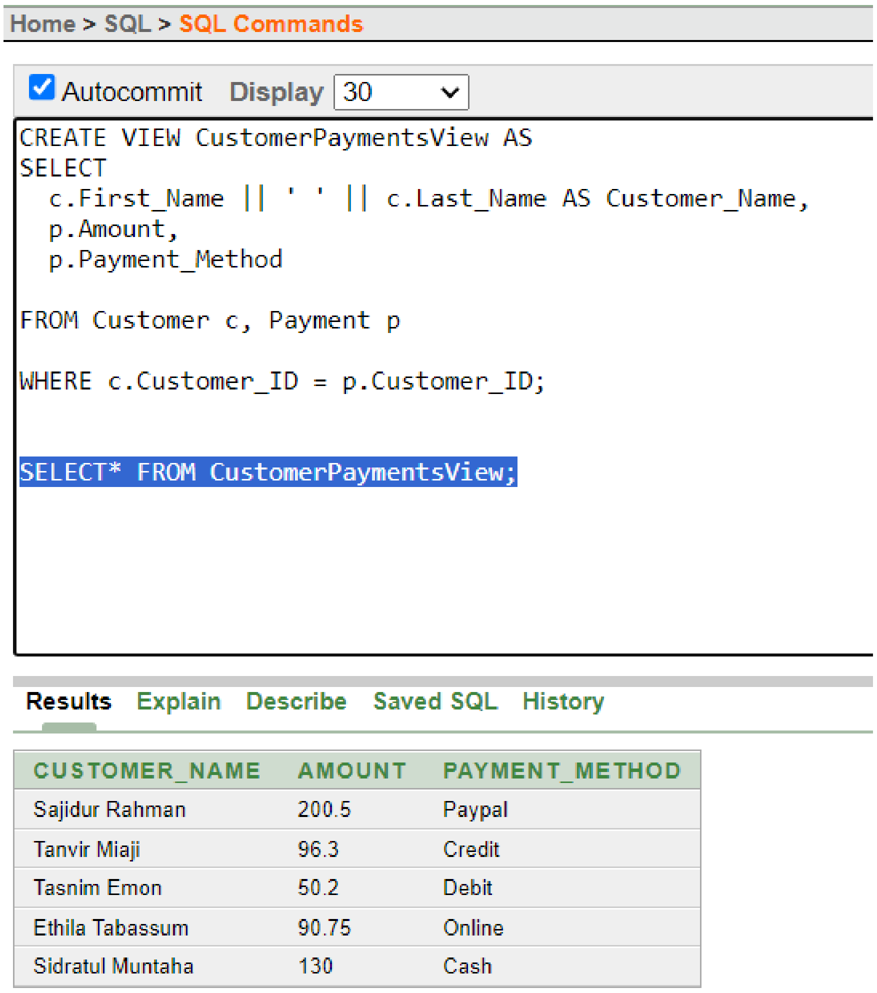

Home > SQL > SQL Commands
Autocommit Display 30
Results Explain Describe Saved SQL History

```
CREATE VIEW BookOrdersView AS
SELECT
  c.First_Name || ' ' || c.Last_Name AS Customer_Name,
  b.Title AS Book_Title,
  o.Order_Date
FROM Customer c, "Order" o, Books b
WHERE
  c.Customer_ID = o.Customer_ID AND o.Book_ID = b.Book_ID;


SELECT* FROM BookOrdersView;
```

CUSTOMER_NAME	BOOK_TITLE	ORDER_DATE
Sajidur Rahman	Harry Potter and the Philosopher's Stone	20-AUG-23
Tanvir Miaji	A Game of Thrones	21-AUG-23
Tasnim Emon	The Old Man and the Sea	21-AUG-23
Ethila Tabassum	Murder on the Orient Express	22-AUG-23
Sidratul Muntaha	It	23-AUG-23

2. Create a view that provides information about customer payments, including customer names, payment amounts and payment methods.

ANS : CREATE VIEW CustomerPaymentsView AS
SELECT
c.First_Name || ' ' || c.Last_Name AS Customer_Name,
p.Amount,
p.Payment_Method
FROM Customer c, Payment p
WHERE c.Customer_ID = p.Customer_ID;


Home > SQL > SQL Commands
Autocommit Display 30
Results Explain Describe Saved SQL History

```
CREATE VIEW CustomerPaymentsView AS
SELECT
  c.First_Name || ' ' || c.Last_Name AS Customer_Name,
  p.Amount,
  p.Payment_Method

FROM Customer c, Payment p

WHERE c.Customer_ID = p.Customer_ID;


SELECT* FROM CustomerPaymentsView;
```

CUSTOMER_NAME	AMOUNT	PAYMENT_METHOD
Sajidur Rahman	200.5	Paypal
Tanvir Miaji	96.3	Credit
Tasnim Emon	50.2	Debit
Ethila Tabassum	90.75	Online
Sidratul Muntaha	130	Cash

10 Conclusion

In conclusion, the development of the online bookstore management system has been successfully accomplished. The system efficiently handles various aspects of the bookstore's operations, including customer management, order processing, book inventory, and payment tracking. Through the utilization of a well-structured database and carefully designed tables, the project has provided an effective solution to manage and streamline bookstore activities.

10.1 Project Findings

Throughout the project, several key findings have emerged:
- The integration of tables and constraints has enabled the establishment of meaningful relationships between entities such as customers, orders, payments, books, authors, and genres.
- The use of views has simplified complex queries and provided a clear overview of specific aspects, such as customer payments and book orders.
- The implementation of subqueries and group functions has allowed for data extraction and analysis in a more versatile manner.

10.2 Future Work

Looking ahead, there are several avenues for enhancing and expanding the existing project:

- **User Interface Enhancement:** Incorporating a user-friendly interface would enable bookstore staff and customers to interact with the system more intuitively. This could involve creating a web-based dashboard or a mobile app for easy access and navigation.
- **Inventory Management:** Implementing a more advanced inventory management system that tracks stock levels, predicts demand, and automates reordering processes could optimize book availability and sales.
- **Reporting and Analytics:** Developing robust reporting and analytics capabilities could provide insights into sales trends, customer preferences, and financial performance, aiding in decision-making and strategic planning.
- **Integration with Online Sales Channels:** Integrating the system with online sales platforms could extend the bookstore's reach and allow customers to purchase books online, expanding revenue streams.
- **User Personalization:** Implementing personalized recommendations for customers based on their past purchases and preferences could enhance customer satisfaction and drive sales.

Current project meets core needs. Enhancements like user-friendly interfaces, advanced inventory management, insightful reporting, online integration, and personalization can d rive system efficiency, customer experiences, and business growth.