ASSIGNMENT 1

Analyze a given business scenario and create an ER diagram that includes entities, relationships, attributes, and cardinality. Ensure that the diagram reflects proper normalization up to the third normal form.

Analyze a given business scenario of Online Book Store:
Step-by-Step Breakdown:

Step 1: Identify Entities and Attributes

1. Author
   - Author Id (Primary Key)
   - Book Id (Foreign Key)
   - Name
   - Description
   - Photo
   - DOB

2. Book
   - Book Id (Primary Key)
   - Title
   - Language Id (Foreign Key)
   - Publication Date
   - Author Id (Foreign Key)
   - Price
   - Image
   - Edition
   - Status (Available, Not available)
   - Stock
   - Genre

3. Review
   - Id (Primary Key)
   - Customer Id (Foreign Key)

- Book Id (Foreign Key)
- Review
- Date

4. User
   - Id (Primary Key)
   - First Name
   - Last Name
   - Email
   - Password
   - DOB
   - Address Line 1
   - Address Line 2
   - City Id (Foreign Key)
   - Phone Number (Unique)
   - Role

5. Cart
   - Id (Primary Key)
   - User Id (Foreign Key)
   - Book Id (Foreign Key)
   - Total Books
   - Total Price
   - Discount

6. Order
   - Id (Primary Key)
   - Cart Id (Foreign Key)
   - Quantity
   - Customer Id (Foreign Key)
   - Order Date
   - Dest Address (Composite Attribute)
   - Status (Delivery Status)
   - Payment Id (Foreign Key)

7. Payment
   - Id (Primary Key)
   - Customer Id (Foreign Key)
   - Order Id (Foreign Key)
   - Status
   - Payment Method

8. Language
   - Id (Primary Key)
   - Name

9. City
   - Name
   - Pincode (Primary Key, Foreign Key)
   - Country Id (Foreign Key)

10. Country
    - Id (Primary Key)
    - Name

Step 2: Defining Relationships and Cardinalities

1. Book and Author
   - One-to-Many Relationship
   - Each book is written by one author.
   - Each author can write multiple books.
   - Foreign Key: Author Id in Book table referencing Author.

2. Book and Language
   - Many-to-One Relationship
   - Each book is in one language.
   - Each language can have multiple books.
   - Foreign Key: Language Id in Book table referencing Language.

3. User and Review

- One-to-Many Relationship
- Each user can write multiple reviews.
- Each review is written by one user.
- Foreign Key: Customer Id in Review table referencing User.

4. Book and Review
- One-to-Many Relationship
- Each book can have multiple reviews.
- Each review is about one book.
- Foreign Key: Book Id in Review table referencing Book.

5. User and Cart
- One-to-Many Relationship
- Each user can have multiple carts.
- Each cart belongs to one user.
- Foreign Key: User Id in Cart table referencing User.

6. Book and Cart
- Many-to-Many Relationship (resolved by Cart table)
- Each book can be in multiple carts.
- Each cart can contain multiple books.
- Foreign Key: Book Id in Cart table referencing Book.

7. User and Order
- One-to-Many Relationship
- Each user can place multiple orders.
- Each order is placed by one user.
- Foreign Key: Customer Id in Order table referencing User.

8. Cart and Order
- One-to-Many Relationship
- Each cart can result in multiple orders (if split).
- Each order is associated with one cart.
- Foreign Key: Cart Id in Order table referencing Cart.

9. Order and Payment
   - One-to-One Relationship
   - Each order has one payment.
   - Each payment is for one order.
   - Foreign Key: Payment Id in Order table referencing Payment.

10. City and Country
    - Many-to-One Relationship
    - Each city is in one country.
    - Each country can have multiple cities.
    - Foreign Key: Country Id in City table referencing Country.

11. User and City
    - Many-to-One Relationship
    - Each user lives in one city.
    - Each city can have multiple users.
    - Foreign Key: City Id in User table referencing City.

Normalization Analysis

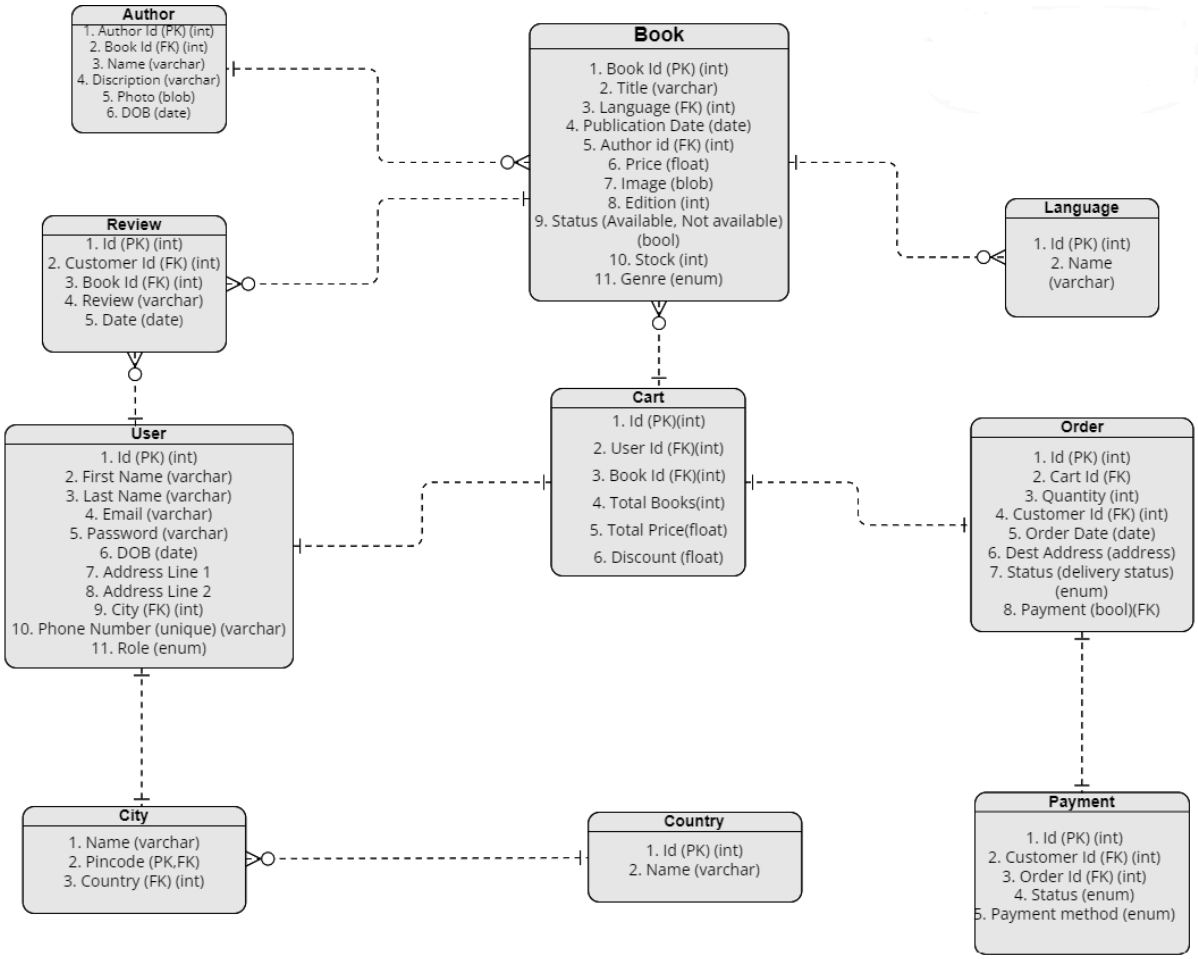
1. First Normal Form (1NF)
   - All attributes contain only atomic values.
   - No repeating groups or arrays.
   - The diagram satisfies 1NF.

2. Second Normal Form (2NF)
   - The table is in 1NF.
   - All non-key attributes are fully functionally dependent on the primary key.
   - The diagram satisfies 2NF.

3. Third Normal Form (3NF)
   - The table is in 2NF.
   - All attributes are directly dependent on the primary key, not on other non-key attributes (i.e., no transitive dependency).

- The diagram satisfies 3NF.

Final ER Diagram:



ASSIGNMENT 2

Design a database schema for a library system, including tables, fields, and constraints like NOT NULL, UNIQUE, and CHECK. Include primary and foreign keys to establish relationships between tables.

Creating a database schema for a library system involves defining tables that store information about books, authors, members, loans, and other related entities. Below is a detailed schema using Oracle SQL constraints, including primary keys, foreign keys, and other constraints like NOT NULL, UNIQUE, and CHECK.

Tables and Schema

1. Authors Table

Sql Query

```
CREATE TABLE Authors (
    AuthorID NUMBER PRIMARY KEY,
    FirstName VARCHAR2(50) NOT NULL,
    LastName VARCHAR2(50) NOT NULL,
    BirthDate DATE NOT NULL,
    UNIQUE (FirstName, LastName, BirthDate)
);
```

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
AUTHORS	AUTHORID	Number	-	-	-	1	-	-	-
	FIRSTNAME	Varchar2	50	-	-	-	-	-	-
	LASTNAME	Varchar2	50	-	-	-	-	-	-
	BIRTHDATE	Date	7	-	-	-	-	-	-
									1 - 4

2. Books Table

Sql Query

```
CREATE TABLE Books (
    BookID NUMBER PRIMARY KEY,
    Title VARCHAR2(100) NOT NULL,
    ISBN VARCHAR2(13) UNIQUE NOT NULL,
    Publisher VARCHAR2(50),
    PublicationYear NUMBER CHECK (PublicationYear >= 1450),
    AuthorID NUMBER,
    CONSTRAINT fk_author FOREIGN KEY (AuthorID) REFERENCES
Authors(AuthorID)
);
```

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
BOOKS	BOOKID	Number	-	-	-	1	-	-	-
	TITLE	Varchar2	100	-	-	-	-	-	-
	ISBN	Varchar2	13	-	-	-	-	-	-
	PUBLISHER	Varchar2	50	-	-	-	✓	-	-
	PUBLICATIONYEAR	Number	-	-	-	-	✓	-	-
	AUTHORID	Number	-	-	-	-	✓	-	-

1 - 6

3. Members Table

Sql Query

```
CREATE TABLE Members (
    MemberID NUMBER PRIMARY KEY,
    FirstName VARCHAR2(50) NOT NULL,
    LastName VARCHAR2(50) NOT NULL,
    Email VARCHAR2(100) UNIQUE NOT NULL,
    PhoneNumber VARCHAR2(15),
    Address VARCHAR2(200),
    MembershipDate DATE DEFAULT SYSDATE NOT NULL
);
```

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
MEMBERS	MEMBERID	Number	-	-	-	1	-	-	-
	FIRSTNAME	Varchar2	50	-	-	-	-	-	-
	LASTNAME	Varchar2	50	-	-	-	-	-	-
	EMAIL	Varchar2	100	-	-	-	-	-	-
	PHONENUMBER	Varchar2	15	-	-	-	✓	-	-
	ADDRESS	Varchar2	200	-	-	-	✓	-	-
	MEMBERSHIPDATE	Date	7	-	-	-	-	SYSDATE	-

1 - 7

4. Loans Table

Sql Query

```
CREATE TABLE Loans (
    LoanID NUMBER PRIMARY KEY,
    MemberID NUMBER NOT NULL,
    BookID NUMBER NOT NULL,
    LoanDate DATE DEFAULT SYSDATE NOT NULL,
    DueDate DATE NOT NULL,
    ReturnDate DATE,
    CONSTRAINT fk_member FOREIGN KEY (MemberID)
REFERENCES Members(MemberID),
    CONSTRAINT fk_book FOREIGN KEY (BookID) REFERENCES
Books(BookID),
    CONSTRAINT chk_due_date CHECK (DueDate > LoanDate)
);
```

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
LOANS	LOANID	Number	-	-	-	1	-	-	-
	MEMBERID	Number	-	-	-	-	-	-	-
	BOOKID	Number	-	-	-	-	-	-	-
	LOANDATE	Date	7	-	-	-	-	SYSDATE	-
	DUEDATE	Date	7	-	-	-	-	-	-
	RETURNDATE	Date	7	-	-	-	✓	-	-

1 - 6

5. Categories Table

Sql Query

```
CREATE TABLE Categories (
    CategoryID NUMBER PRIMARY KEY,
    CategoryName VARCHAR2(50) UNIQUE NOT NULL
);
```

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
CATEGORIES	CATEGORYID	Number	-	-	-	1	-	-	-
	CATEGORYNAME	Varchar2	50	-	-	-	-	-	-

1 - 2

6. BookCategories Table (Many-to-Many Relationship)

Sql Query

```
CREATE TABLE BookCategories (
    BookID NUMBER NOT NULL,
    CategoryID NUMBER NOT NULL,
    CONSTRAINT pk_book_category PRIMARY KEY (BookID,
CategoryID),
    CONSTRAINT fk_book_category_book FOREIGN KEY (BookID)
REFERENCES Books(BookID),
    CONSTRAINT fk_book_category_category FOREIGN KEY
(CategoryID) REFERENCES Categories(CategoryID)
);
```

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
BOOKCATEGORIES	BOOKID	Number	-	-	-	1	-	-	-
	CATEGORYID	Number	-	-	-	2	-	-	-

1 - 2

Explanation of Constraints

1. Primary Keys:
   - Ensures that each record in the table is uniquely identifiable.
   - Example: `AuthorID`, `BookID`, `MemberID`, `LoanID`, `CategoryID`.

2. Foreign Keys:
   - Establishes relationships between tables, ensuring data integrity.
   - Example: `AuthorID` in `Books` references `AuthorID` in `Authors`.
   - Example: `MemberID` in `Loans` references `MemberID` in `Members`.
   - Example: `BookID` in `Loans` references `BookID` in `Books`.

3. NOT NULL:
   - Ensures that a column cannot have NULL values.
   - Example: `FirstName` and `LastName` in `Authors` and `Members`.

4. UNIQUE:
   - Ensures all values in a column are unique.
   - Example: `ISBN` in `Books`, `Email` in `Members`, `CategoryName` in `Categories`.

5. CHECK:
   - Ensures that all values in a column satisfy a specific condition.
   - Example: `PublicationYear` in `Books` is within a reasonable range.
   - Example: `DueDate` in `Loans` must be after the `LoanDate`.

ASSIGNMENT 3

Explain the ACID properties of a transaction in your own words. Write SQL statements to simulate a transaction that includes locking and demonstrate different isolation levels to show concurrency control.

Here's a breakdown of the ACID properties with a transaction example:

1. Atomicity: It's like a bank transfer. When you transfer money from one account to another, either the entire transfer succeeds, and the money is deducted from one account and added to another, or it fails, and nothing changes. It's like the bank ensures that the entire transfer is completed or none of it is.

2. Consistency: In our banking app, consistency ensures that after a successful transfer, the total amount of money in all accounts remains the same. If $100 is transferred from Account A to Account B, the total amount of money in both accounts combined should remain constant. This maintains the integrity of the bank's total assets.

3. Isolation: Imagine two people trying to transfer money from the same account simultaneously. Isolation ensures that one person's transfer doesn't interfere with the other's. Even if both transfers happen at the same time, they should be processed as if they occurred sequentially, ensuring data integrity and preventing issues like overdrawing.

4. Durability: Once a transfer is completed, it should persist even if the system crashes. Just like how a successful bank transfer remains in your transaction history, even if you close the app or lose internet connection, database transactions should be durable and not lost due to system failures.

SQL statements to simulate a transaction that includes locking and demonstrate different isolation levels to show concurrency control.

```
-- Setting up a simple bank account table
CREATE TABLE BankAccount (
    AccountNumber INT PRIMARY KEY,
    Balance DECIMAL(10, 2)
);

-- Inserting some sample data
INSERT INTO BankAccount (AccountNumber, Balance) VALUES
(1, 1000.00),
(2, 2000.00);

-- Starting a transaction
BEGIN TRANSACTION;

-- Withholding money from one account
UPDATE BankAccount SET Balance = Balance - 500 WHERE
AccountNumber = 1;

-- Depositing the same amount into another account
UPDATE BankAccount SET Balance = Balance + 500 WHERE
AccountNumber = 2;

-- Committing the transaction
COMMIT;

-- Showing the updated balances
SELECT * FROM BankAccount;
```

Now, let's demonstrate different isolation levels:

```
-- Set isolation level to READ COMMITTED
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

-- Execute the same transaction simultaneously in two separate sessions
-- In one session:
BEGIN TRANSACTION;
-- Deducting money from Account 1
UPDATE BankAccount SET Balance = Balance - 200 WHERE
AccountNumber = 1;
-- Committing the transaction
COMMIT;

-- In another session:
BEGIN TRANSACTION;
-- Deducting money from Account 1
UPDATE BankAccount SET Balance = Balance - 300 WHERE
AccountNumber = 1;
-- Committing the transaction
COMMIT;

-- Show the updated balances
SELECT * FROM BankAccount;
```

ASSIGNMENT 4

Write SQL statements to CREATE a new database and tables that reflect the library schema you designed earlier. Use ALTER statements to modify the table structures and DROP statements to remove a redundant table.

Creating a new schema(database)

```
CREATE USER library_user IDENTIFIED BY PASSWORD;

GRANT RESOURCE  TO library_user;

GRANT CONNECT T
```

To modify the table structures and drop a redundant table, we can use ALTER and DROP statements.

1. Modify the `Members` Table:
   We'll add a column called `MembershipStatus` to the `Members` table to track the status of the membership.

```
ALTER TABLE Members
ADD MembershipStatus VARCHAR2(20) DEFAULT 'Active';
```

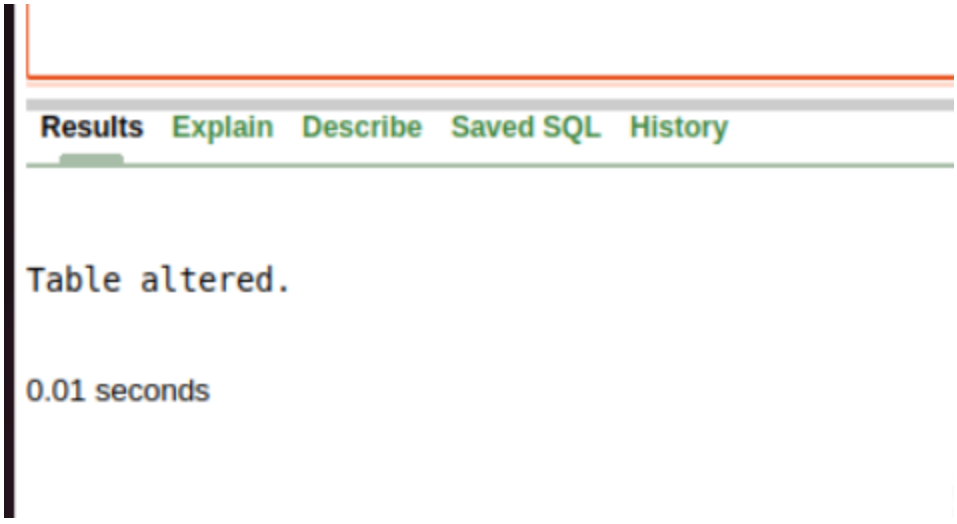


Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
MEMBERS	MEMBERID	Number	-	-	-	1	-	-	-
	FIRSTNAME	Varchar2	50	-	-	-	-	-	-
	LASTNAME	Varchar2	50	-	-	-	-	-	-
	EMAIL	Varchar2	100	-	-	-	-	-	-
	PHONENUMBER	Varchar2	15	-	-	-	✓	-	-
	ADDRESS	Varchar2	200	-	-	-	✓	-	-
	MEMBERSHIPDATE	Date	7	-	-	-	-	SYSDATE	-
	MEMBERSHIPSTATUS	Varchar2	20	-	-	-	✓	'Active'	-

1 - 8

2. Drop the `BookCategories` Table:
   The `BookCategories` table is not redundant, but if you want to drop it, you can do so using the following statement:

```
DROP TABLE BookCategories;
```

After executing these statements, the `Members` table will have a new column `MembershipStatus`, and the `BookCategories` table will be dropped from the schema.

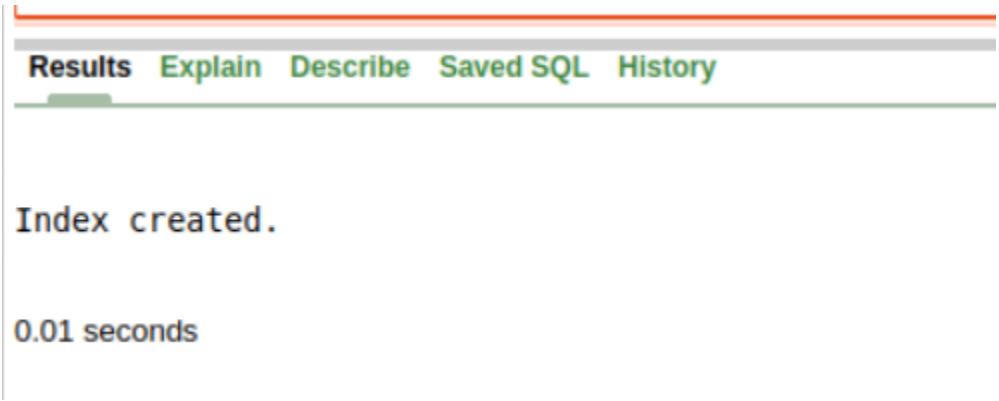
ASSIGNMENT 5

Demonstrate the creation of an index on a table and discuss how it improves query performance. Use a DROP INDEX statement to remove the index and analyze the impact on query execution.

To create an index on a table, we can use the `CREATE INDEX` statement in SQL. An index is a database object that improves the speed of data retrieval operations on a table by providing quick access to the rows based on the indexed columns. It does this by maintaining a sorted structure of the indexed column values, which allows the database engine to perform efficient searches.

Creating Index

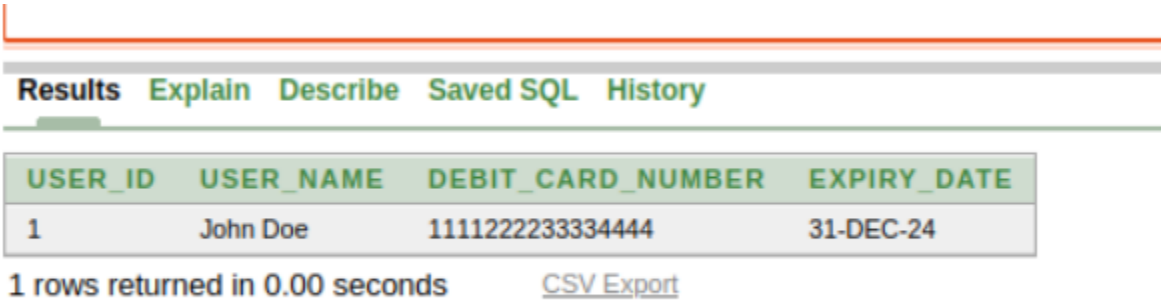
```
CREATE INDEX idx_debit_card_number ON Debit_Card_Users
(Debit_Card_Number);
```



use of the index
We'll perform a query that searches for a specific debit card number.

Here's how we can utilize the index in a query:
-- Query to retrieve user information based on debit card number

```
SELECT *
FROM Debit_Card_Users
WHERE Debit_Card_Number = '1111222233334444';
```



Results Explain Describe Saved SQL History

USER_ID	USER_NAME	DEBIT_CARD_NUMBER	EXPIRY_DATE
1	John Doe	1111222233334444	31-DEC-24

1 rows returned in 0.00 seconds CSV Export

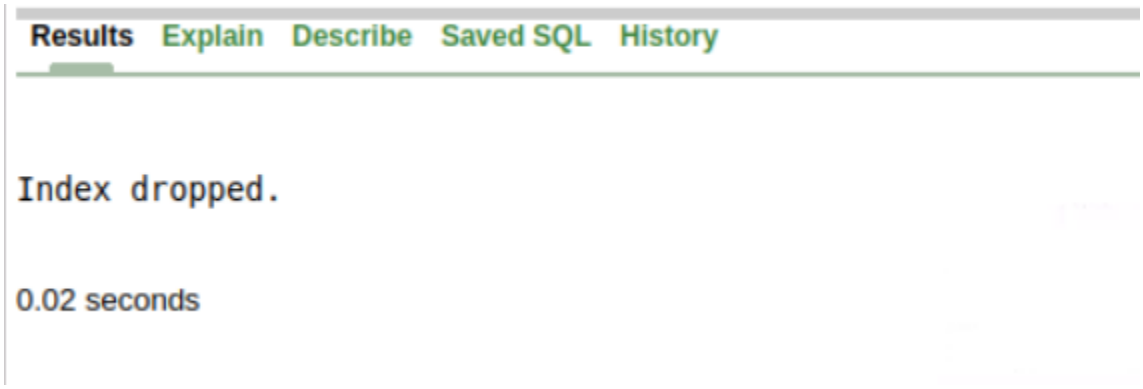
- In this query, we're retrieving all columns (`*`) from the `Debit_Card_Users` table where the `Debit_Card_Number` column matches the specified debit card number (`'1111222233334444'`).

- With the index `idx_debit_card_number` in place, the database engine can efficiently locate the rows that match the given debit card number using the index. Instead of performing a full table scan, it can quickly navigate the index to find the relevant rows, resulting in faster query execution.

- The use of the index ensures that the query is executed efficiently, even when the `Debit_Card_Number` column contains a large number of rows. The index provides quick access to the desired rows based on the indexed column, improving query performance significantly.

Dropping Index

```
DROP INDEX idx_debit_card_number;
```



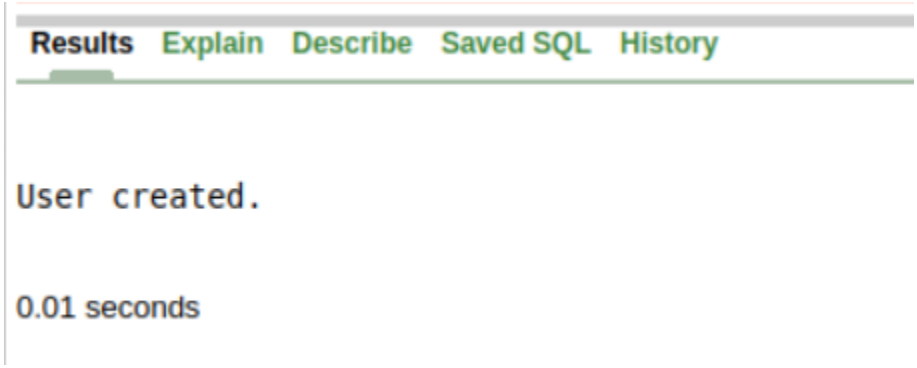
ASSIGNMENT 6

Create a new database user with specific privileges using the CREATE USER and GRANT commands. Then, write a script to REVOKE certain privileges and DROP the user.

Part 1: Creating a New User and Granting Privileges

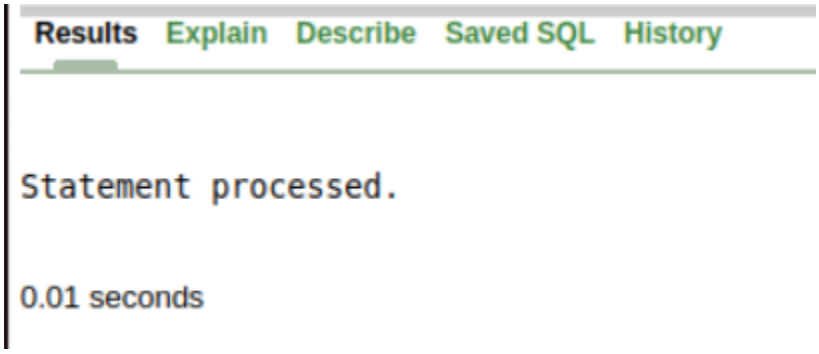
1. In the SQL Command Processor, executed the following SQL statement to create a new user with a password:

```
CREATE USER newuser IDENTIFIED BY password;
```



2. Granted the necessary privileges to the new user. (all ranted changes are automatically committed)

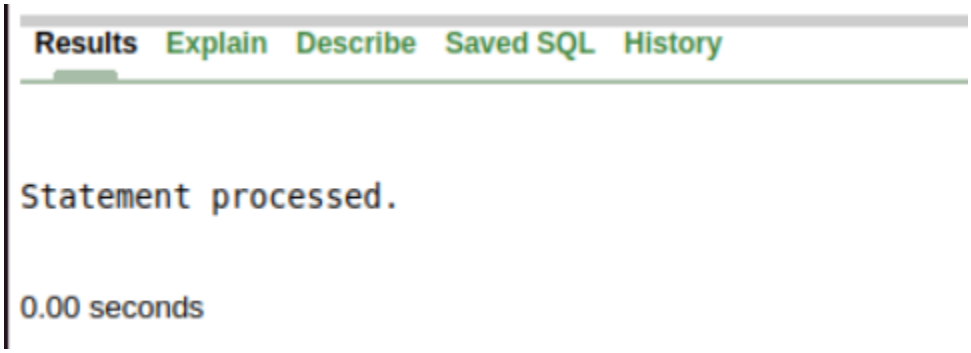
```
GRANT CREATE SESSION TO newuser;
GRANT CREATE TABLE TO newuser;
```



Part 2: Revoking Privileges and Dropping the User

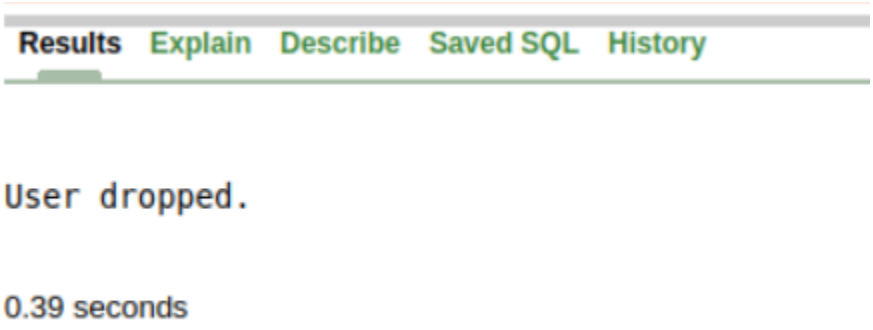
1. To revoke the `CREATE TABLE` privilege from the `newuser`, executed the following SQL statement:

```
REVOKE CREATE TABLE FROM newuser;
```



2. To drop the `newuser` from the database, executed the following SQL statement:

```
DROP USER newuser;
```



ASSIGNMENT 7

Prepare a series of SQL statements to INSERT new records into the library tables, UPDATE existing records with new information, and DELETE records based on specific criteria. Include BULK INSERT operations to load data from an external source.

1. INSERT Statements:

```
-- Insert new authors
INSERT INTO Authors(AuthorID, FirstName, LastName, BirthDate)
VALUES (1, 'Jane', 'Austen', TO_DATE('1775-12-16', 'YYYY-MM-DD'));

INSERT INTO Authors (AuthorID, FirstName, LastName, BirthDate)
VALUES (2, 'George', 'Orwell', TO_DATE('1903-06-25', 'YYYY-MM-DD'));
```

AUTHORID	FIRSTNAME	LASTNAME	BIRTHDATE
2	George	Orwell	25-JUN-03
1	Jane	Austen	16-DEC-75

2 rows returned in 0.00 seconds CSV Export

```
-- Insert new books
INSERT INTO Books (BookID, Title, ISBN, Publisher, PublicationYear,
AuthorID)
VALUES (1, 'Pride and Prejudice', '9780141439519', 'Penguin Classics',
1813, 1);

INSERT INTO Books (BookID, Title, ISBN, Publisher, PublicationYear,
AuthorID)
VALUES (2, '1984', '9780451524935', 'Signet Classics', 1949, 2);
```

BOOKID	TITLE	ISBN	PUBLISHER	PUBLICATIONYEAR	AUTHORID
1	Pride and Prejudice	9780141439519	Penguin Classics	1813	1
2	1984	9780451524935	Signet Classics	1949	2

2 rows returned in 0.00 seconds CSV Export

```
-- Insert new members
INSERT INTO Members (MemberID, FirstName, LastName, Email,
PhoneNumber, Address)
VALUES (1, 'John', 'Doe', 'john.doe@email.com', '1234567890', '123 Main
St, City, State');

INSERT INTO Members (MemberID, FirstName, LastName, Email,
PhoneNumber, Address)
VALUES (2, 'Jane', 'Smith', 'jane.smith@email.com', '9876543210', '456
Elm St, City, State');
```

MEMBERID	FIRSTNAME	LASTNAME	EMAIL	PHONENUMBER	ADDRESS	MEMBERSHIPDATE	MEMBERSHIPSTATUS
1	John	Doe	john.doe@email.com	1234567890	123 Main St, City, State	21-MAY-24	Active
2	Jane	Smith	jane.smith@email.com	9876543210	456 Elm St, City, State	21-MAY-24	Active

2 rows returned in 0.00 seconds CSV Export

```
-- Insert new loans
INSERT INTO Loans (LoanID, MemberID, BookID, DueDate)
VALUES (1, 1, 1, SYSDATE + 14);

INSERT INTO Loans (LoanID, MemberID, BookID, DueDate)
VALUES (2, 2, 2, SYSDATE + 21);
```

LOANID	MEMBERID	BOOKID	LOANDATE	DUEDATE	RETURNDATE
1	1	1	21-MAY-24	04-JUN-24	-
2	2	2	21-MAY-24	11-JUN-24	-

2 rows returned in 0.00 seconds CSV Export

```
-- Insert new categories
INSERT INTO Categories (CategoryID, CategoryName)
VALUES (1, 'Fiction');

INSERT INTO Categories (CategoryID, CategoryName)
VALUES (2, 'Non-Fiction');
```

CATEGORYID	CATEGORYNAME
1	Fiction
2	Non-Fiction

2 rows returned in 0.00 seconds CSV Export

2. UPDATE Statements:

```
-- Update book publisher
UPDATE Books
SET Publisher = 'Penguin Random House'
WHERE BookID = 1;
```

BOOKID	TITLE	ISBN	PUBLISHER	PUBLICATIONYEAR	AUTHORID
1	Pride and Prejudice	9780141439519	Penguin Random House	1813	1
2	1984	9780451524935	Signet Classics	1949	2

2 rows returned in 0.00 seconds CSV Export

```
-- Update member address
UPDATE Members
SET Address = '789 Oak St, City, State'
WHERE MemberID = 1;
```

MEMBERID	FIRSTNAME	LASTNAME	EMAIL	PHONENUMBER	ADDRESS	MEMBERSHIPDATE	MEMBERSHIPSTATUS
1	John	Doe	john.doe@email.com	1234567890	789 Oak St, City, State	21-MAY-24	Active
2	Jane	Smith	jane.smith@email.com	9876543210	456 Elm St, City, State	21-MAY-24	Active

2 rows returned in 0.01 seconds CSV Export

3. DELETE Statements:

```
-- Step 1: Delete the related loan records
DELETE FROM Loans WHERE BookID = 2;
```

Results Explain Describe Saved SQL History

1 row(s) deleted.

0.00 seconds

Results Explain Describe Saved SQL History

LOANID	MEMBERID	BOOKID	LOANDATE	DUEDATE	RETURNDATE
1	1	1	21-MAY-24	04-JUN-24	-

1 rows returned in 0.00 seconds CSV Export

```
-- Step 2: Delete the book record
DELETE FROM Books WHERE BookID = 2;
```

Results Explain Describe Saved SQL History

```
1 row(s) deleted.
```

0.00 seconds

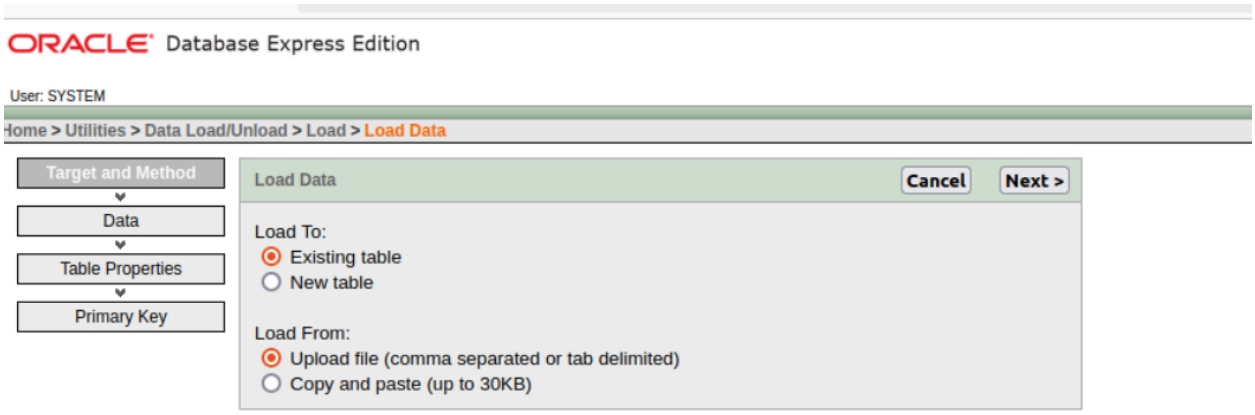
Results Explain Describe Saved SQL History

BOOKID	TITLE	ISBN	PUBLISHER	PUBLICATIONYEAR	AUTHORID
1	Pride and Prejudice	9780141439519	Penguin Random House	1813	1

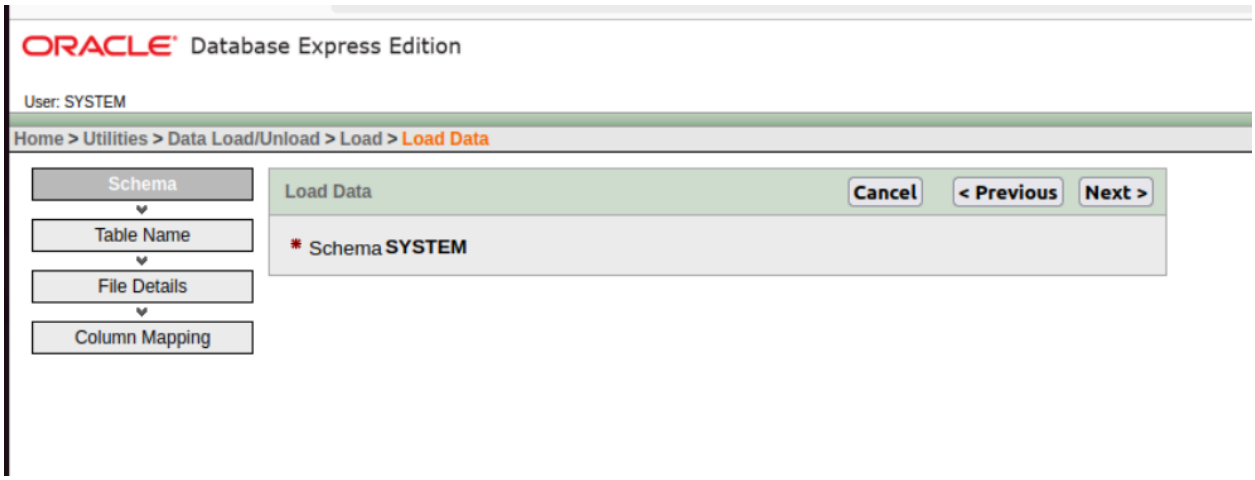
1 rows returned in 0.00 seconds CSV Export

4. BULK INSERT Operation:

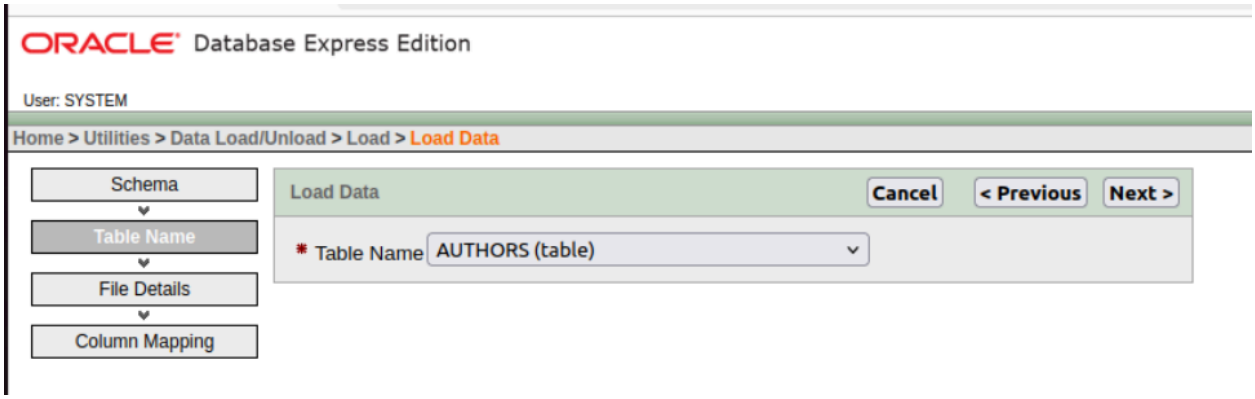
1. Go into Load Data Settings of oracle apex gui



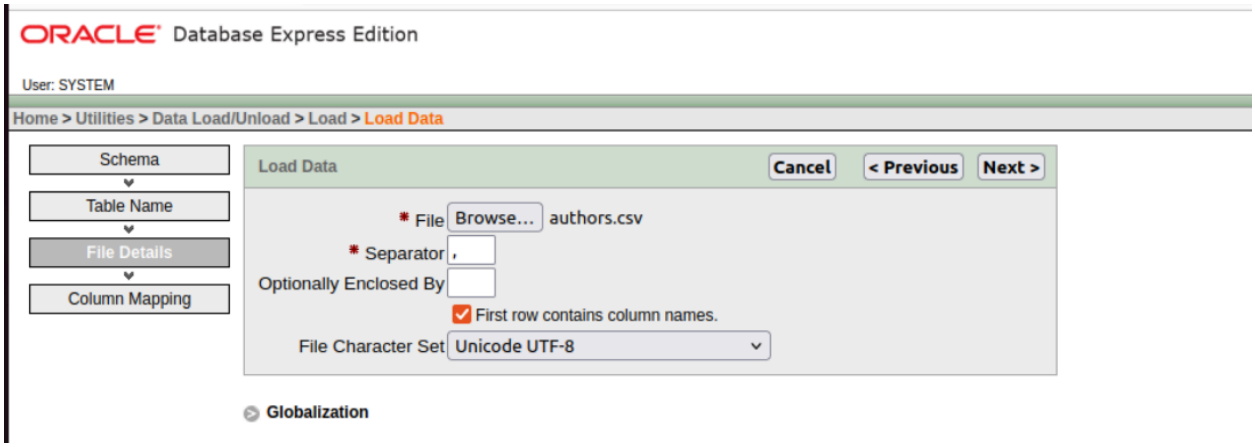
2. Select Schema , here i have selected SYSTEM schema



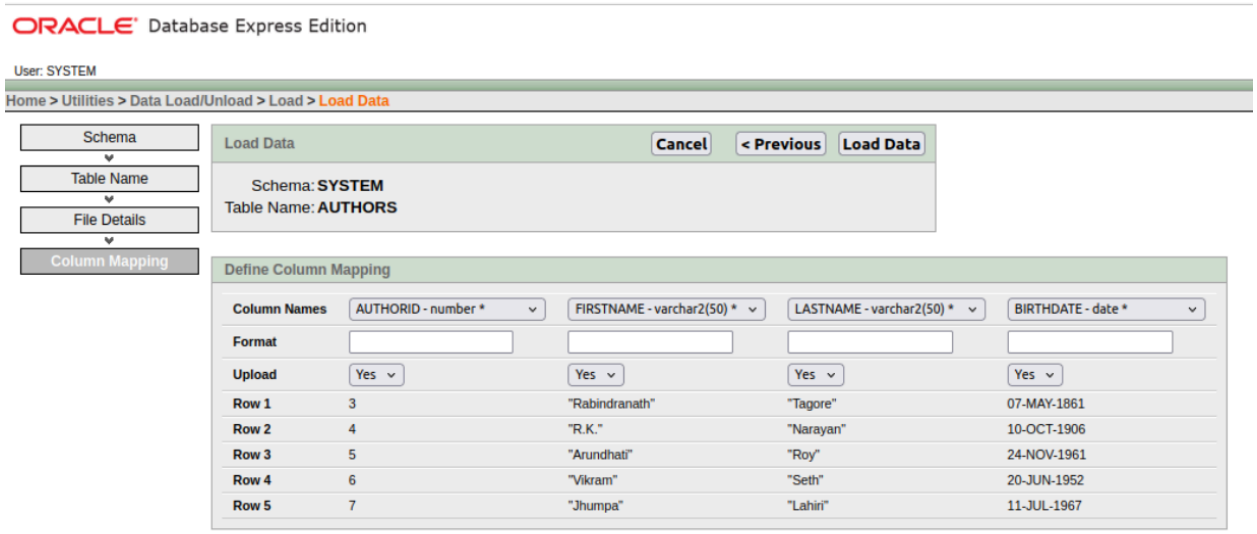
3. Selected Desired table where i wanted to insert csv data in bulk



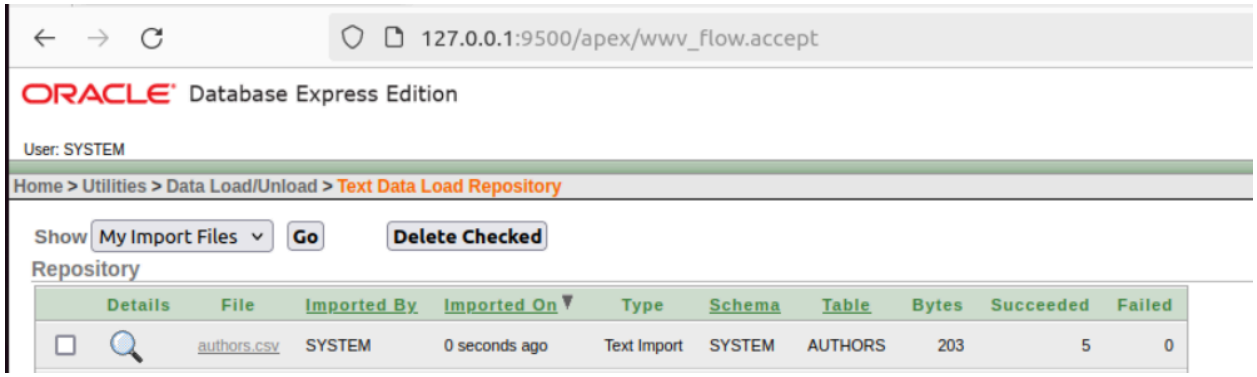
4. Browser into local system and selected desired csv file



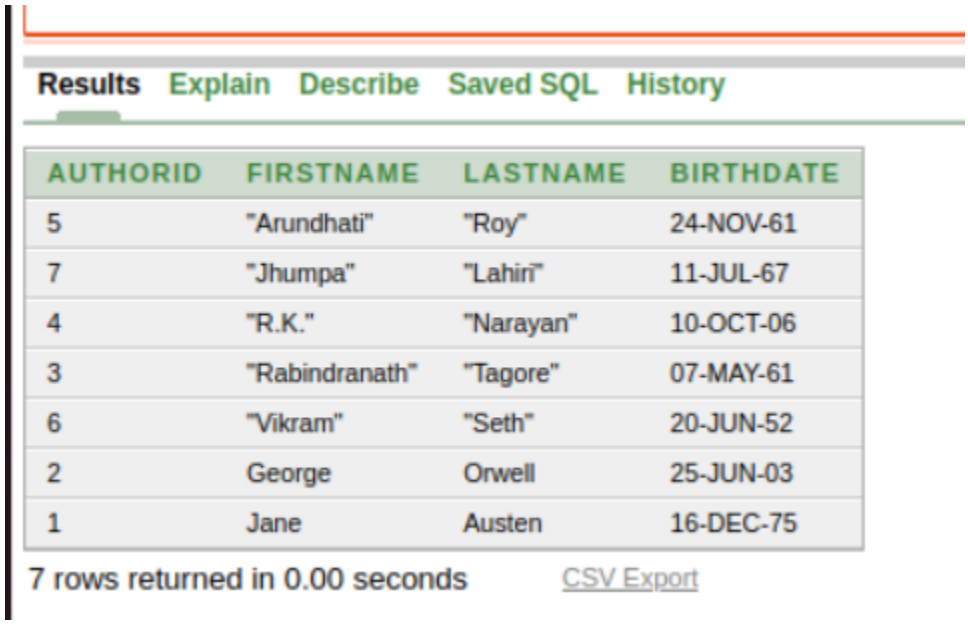
5. I have done than column mapping of csv file data to the existing table data



6. I have successfully loaded data from csv file to my table in oracle database



7. Here is table after bulk data load from csv



Results Explain Describe Saved SQL History

AUTHORID	FIRSTNAME	LASTNAME	BIRTHDATE
5	"Arundhati"	"Roy"	24-NOV-61
7	"Jhumpa"	"Lahiri"	11-JUL-67
4	"R.K."	"Narayan"	10-OCT-06
3	"Rabindranath"	"Tagore"	07-MAY-61
6	"Vikram"	"Seth"	20-JUN-52
2	George	Orwell	25-JUN-03
1	Jane	Austen	16-DEC-75

7 rows returned in 0.00 seconds CSV Export