REPORT ON

DBMS PROJECT ON LIBRARY RESOURCES AND STAFF

CS504

Principles of data management and data mining

SAIPHANI CHANDRA VUPPALA
svuppal3@gmu.edu
G01440607

Contents

1. Introduction

A database management system (DBMS) is a sophisticated software program that enables users to efficiently organize, manage, and retrieve data. Due to its ability to help businesses store and manage massive volumes of data, streamline operations, and improve decision-making processes, DBMS systems have grown to be a crucial part of modern society and business. This document tries to shed light on the core ideas of creating, implementing, and querying a public library database. By using such a system, library employees may effectively manage their resources, which include a variety of things including books, periodicals, digital media, and other things.

2. Entity and Relation

Material Represents individual items available in the library, such as books, magazines, e-books, and audio books.

Attributes:

- Material_ID: A unique identifier for each material.
- Title: The title of the material.
- Publication_Date: The date of publication of the material.
- Catalog_ID: A reference to the catalog entry for the material.
- Genre_ID: A reference to the genre of the material.

2. Catalog Represents a record of library materials with information on their availability and location. Attributes:

- Catalog_ID: A unique identifier for each catalog entry.
- CName: The name of the catalog.
- LLocation: The location of the material within the library.

3. Genre Represents the various genres or categories of library materials.

Attributes:

- Genre_ID: A unique identifier for each genre.
- GName: The name of the genre.
- Description: The brief introduction of the genre.

4. Borrow Represents the borrowing activity of library materials by members.

Attributes:

- Borrow_ID: A unique identifier for each borrowing transaction.
- BMaterial_ID: A reference to the borrowed material.
- BMember_ID: A reference to the member who borrowed the material.
- BStaff_ID: A reference to the staff who processed the transaction.
- Borrow_Date: The date the material was borrowed.
- Due_Date: The date the material is due.
- Return_Date: The date the material is returned.

5. Author Represents authors who have created library materials.

Attributes:

- Author_ID: A unique identifier for each author.
- AName: The name of the author.
- Birth_Date: The birth date of the author.
- Nationality: The nationality of the author.

6. Authorship Represents the relationship between authors and the materials they have created. Attributes:

- Authorship_ID: A unique identifier for each authorship record.
- AAuthor_ID: A reference to the author.
- AMaterial_ID: A reference to the material authored.

7. Member Represents library members who can borrow and reserve materials.

Attributes:

- Member_ID: A unique identifier for each member.
- MName: The name of the member.
- MContact_Info: Email address (or phone number) of the member.
- Join_Date: The date the member joined the library.

8. Staff Represents library staff who manage library resources and assist members.

Attributes:

- Staff_ID: A unique identifier for each staff member.
- SName: The name of the staff member.
- SContact_Info: Email address (or phone number) of the member.
- Job_Title: The job title of the staff member (e.g., librarian, assistant librarian).
- Hire_Date: The date the staff member was hired by the library

3. Design

3.1 ER diagram

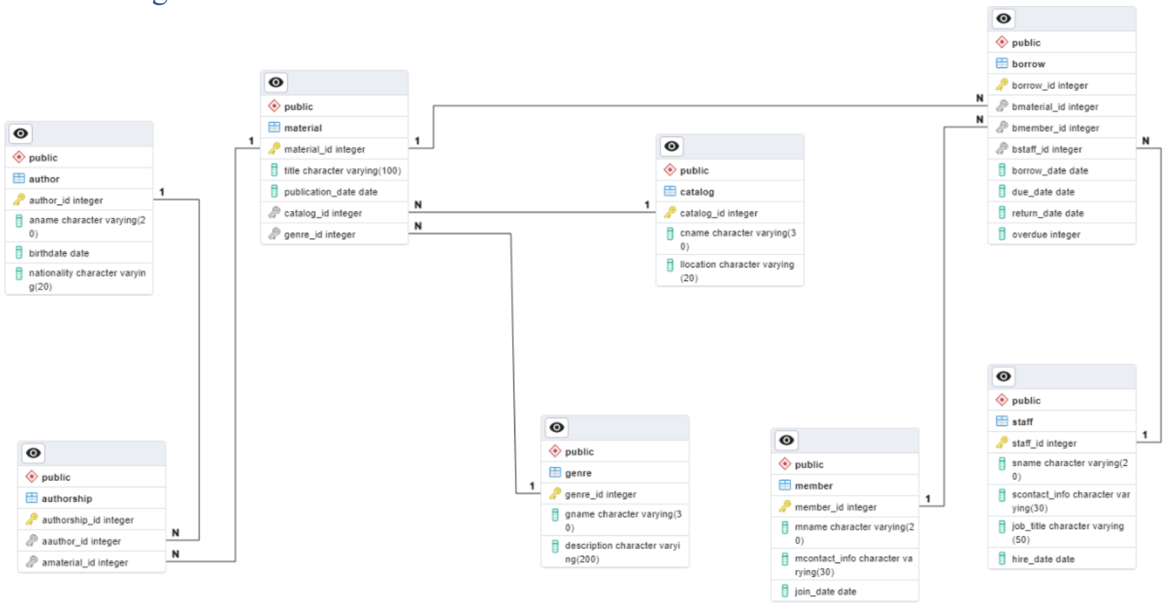


Fig 1.1 : Physical ER model

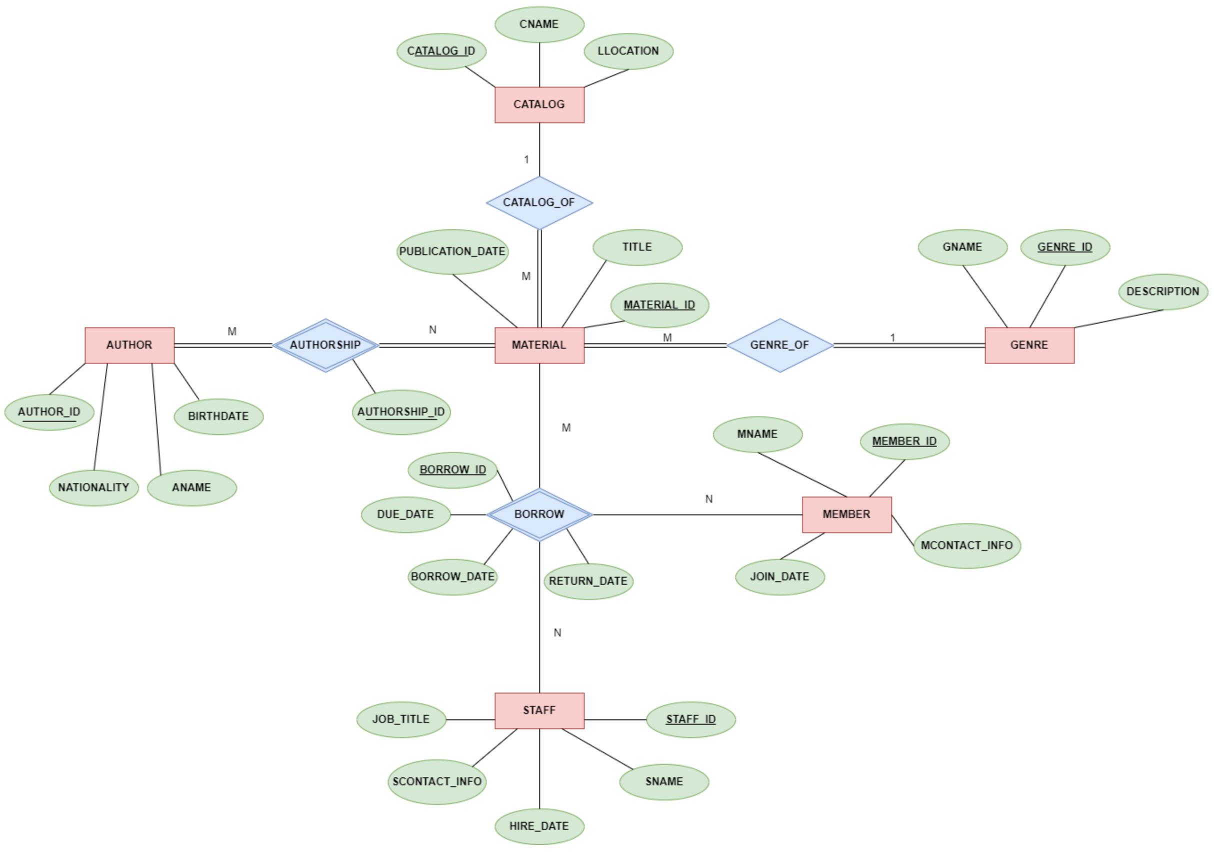


Fig 1.2: Physical ER diagram

The ER diagram, for the given database model consists of several entities, relationships, and attributes(In query they are case insensitive).

The main entities in the ER diagram are:

- Member: This entity contains attributes such as member Id, Mname, Mcontact_info and Join_date.
- Staff: This entity contains attributes such as Sname, Scontact_info, Job_title, Hire_date.
- Material: This entity contains attributes such as material_id, title, Publication_date.
- Catalog: This entity contains attributes such as catalog_id, cname, llocation
- Genre: This entity contains attributes such as Genre_id, gname, description.
- Author: This entity contains attributes such as Author_id, Aname, birthdate, nationality.

Relationships:

- Borrow: This relationship connects the member entity and the material entity and also . It has attributes such as borrow_id, borrow_date, due_date, and return_date, and it is weak relationship.
- Authorship: This relationship connects author entity and material entity. It has attributes such as Authorship_id, and it is weak relationship.
- Catalog_of: This relationship connects material entity and catalog entity.
- Genre_of: This relationship connects material entity and Genre entity.

Cardinality Ratios:

- Borrow: The cardinality ratio between material entity and member entity is (M:N). Which means member can borrow M materials and a material can be borrowed by N people.
- Authorship: The cardinality ratio between material entity and author entity is (M:N). Which means author can write M materials and a material can be written by N author. Both the entities have total participation with each other.
- Catalog_of: The cardinality ratio between material entity and catalog entity is (1:N). Which means catalog can have N materials and a material can have 1 catalog_id, with total participation with catalog_of.
- Genre_of: The cardinality ratio between material entity and genre entity is (M:1). Which means genre can have M materials and a material can have only 1 genre, Genre have total participation with genre_of.

3.2 Relational model

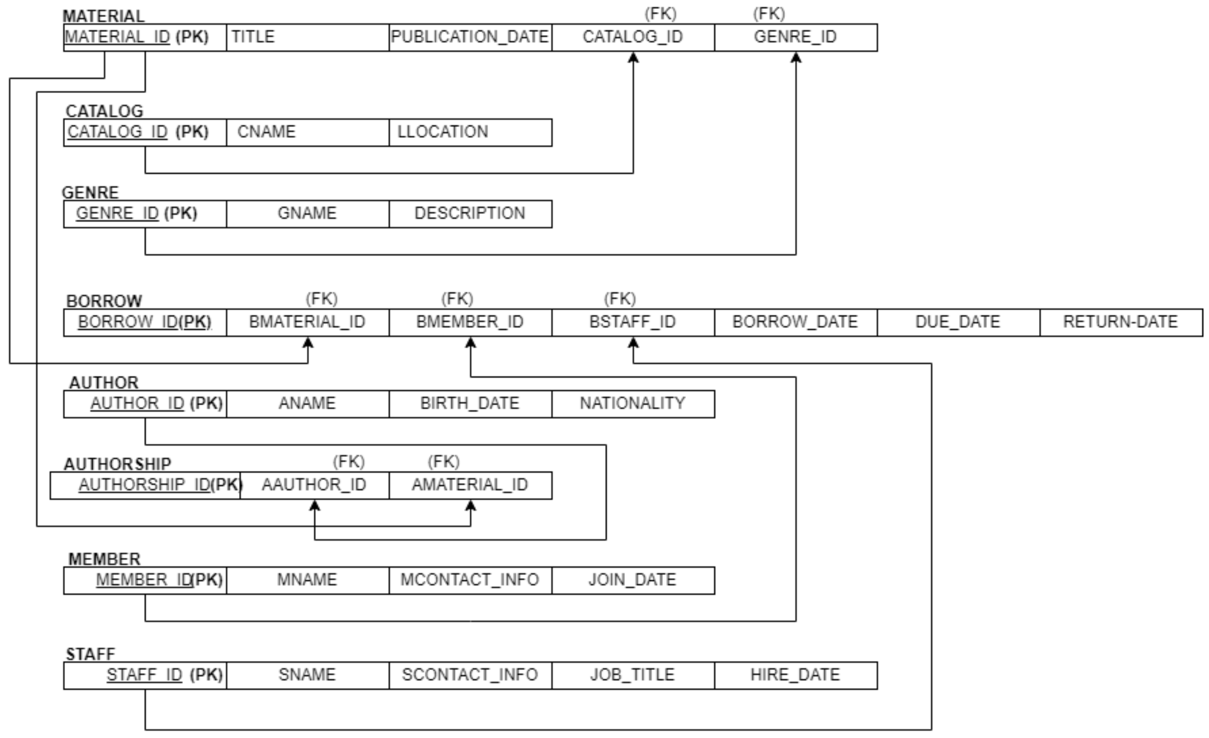


Fig 1.3: Relational model

3.3 Implementation

We must first create the database and add the tables that will contain the data before we can implement the insertion of data into it. This entails deciding which entities will be kept in the database, figuring out what each entity is like, and defining the connections between entities.

A database management system (DBMS) can be used to build the tables after the database structure has been set. The names of the tables, the column headings, the data types, and any necessary constraints or indexes must all be specified. After the tables are created, the data can be added to the database using SQL commands.

3.3.1 Creating database

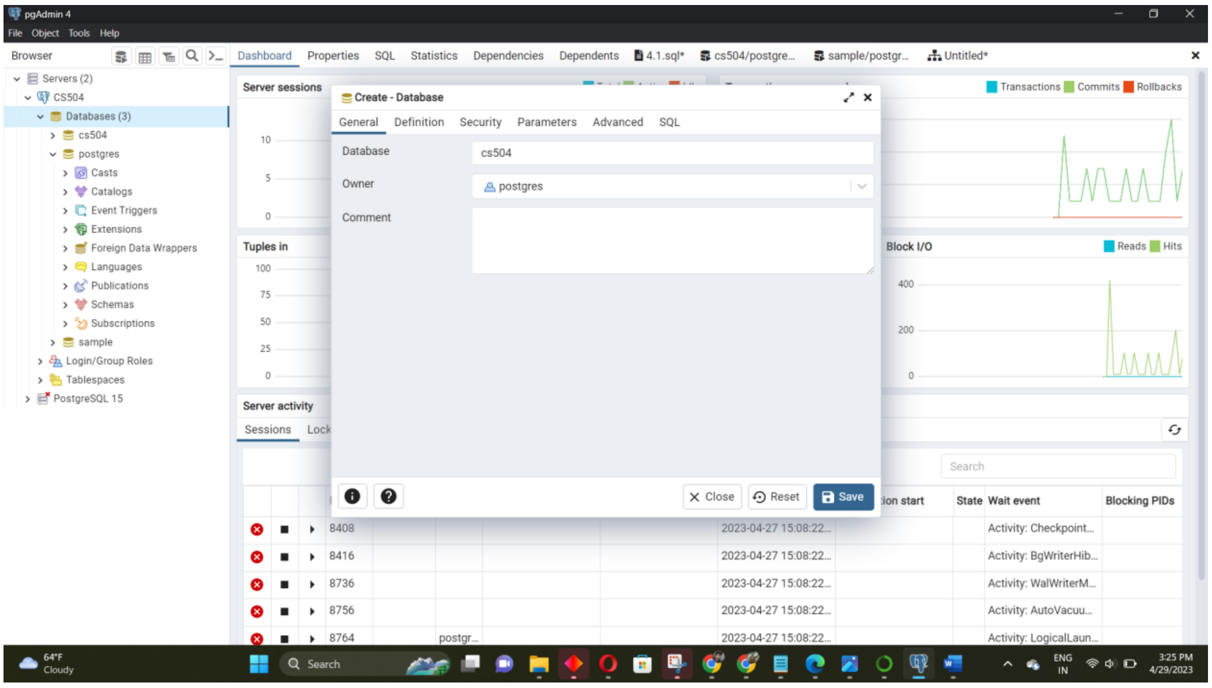
Creating database follow following steps,

Step-1: open pgadmin4.

Step-2: Navigate to browser.

Step-3: choose server.

Step-4: right-click on databases and select to create and Name the database.



Database: cs504

Owner: postgres

Comment:

			PID				Backend start	State	Wait event	Blocking PIDs
			8408				2023-04-27 15:08:22...		Activity: Checkpoint...	
			8416				2023-04-27 15:08:22...		Activity: BgWriterHib...	
			8736				2023-04-27 15:08:22...		Activity: WalWriterM...	
			8756				2023-04-27 15:08:22...		Activity: AutoVacuu...	
			8764	postgr...			2023-04-27 15:08:22...		Activity: LogicalLaun...	

3.3.2 Creating table in database

Before inserting the data into the database, we need to create the tables in the database.

Query:

```
create table Catalog(
Catalog_id int not null primary key,
CName varchar(30),
Llocation varchar(20)-- is location of material within library
)

create table Genre(
Genre_id int not null primary key,
GName varchar(30),-- Genre of the material
Description varchar(200)
)

create table Author(
Author_id int not null primary key,
AName varchar(20),--AName= Name of the author
Birthdate date,
Nationality varchar(20)

);
create table Member(
Member_id int not null primary key,
MName varchar(20), -- name of the member
MContact_info varchar(30), -- email address
Join_date date
);

create table Staff(
Staff_id int not null primary key,
SName varchar(20),-- staff name
SContact_info varchar(30),
Job_title varchar(50),
Hire_date date
);

create  table Material(
Material_id int not null primary key,
Title varchar(60),
Publication_date date,
Catalog_id int,
Genre_id int,
foreign key(Catalog_id) references Catalog(Catalog_id),
foreign key(Genre_id) references Genre(Genre_id)
);

create table Borrow(
Borrow_id int not null primary key,
BMaterial_id int,
BMember_id int,
Bstaff_id int,
Borrow_date date,
```

Due_date date,

Return_date date,

foreign key(BMaterial_id) references Material(Material_id),

foreign key(Bmember_id) references Member(Member_id),

foreign key (Bstaff_id) references Staff(Staff_id)

);

create table Authorship(

Authorship_id int not null primary key,

AAuthor_id int,

AMaterial_id int,

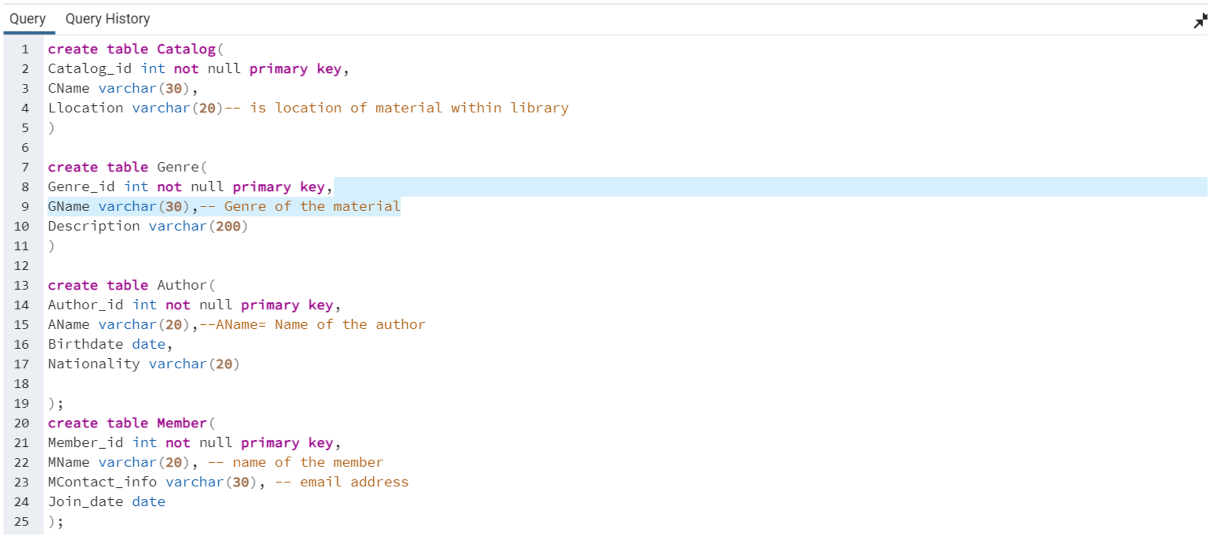
foreign key(AAuthor_id) references Author(Author_id),

foreign key(AMaterial_id) references Material(Material_id)

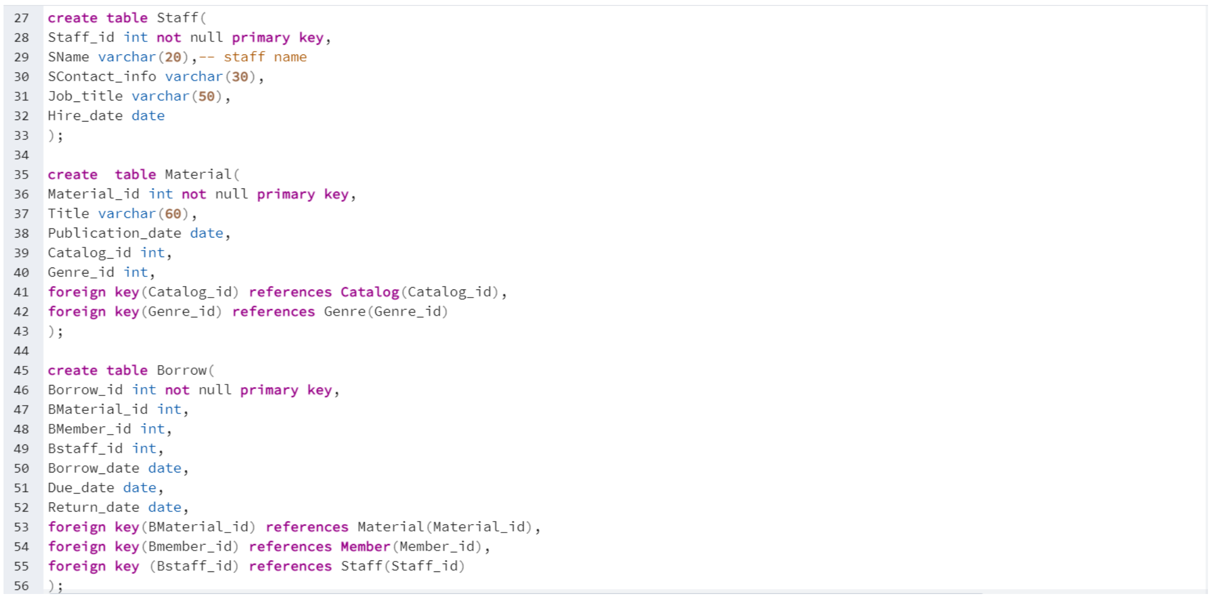
);

Explanation:

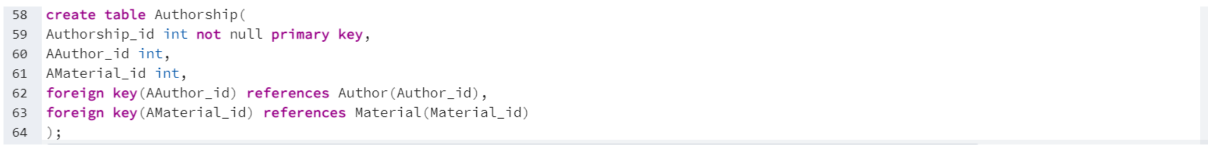
Inorder to create the database, we need to have Entity name, Attribute name, Table description and Table relation



```
create table Catalog(
Catalog_id int not null primary key,
CName varchar(30),
Llocation varchar(20)-- is location of material within library
)

create table Genre(
Genre_id int not null primary key,
GName varchar(30),-- Genre of the material
Description varchar(200)
)

create table Author(
Author_id int not null primary key,
AName varchar(20),--AName= Name of the author
Birthdate date,
Nationality varchar(20)

);
create table Member(
Member_id int not null primary key,
MName varchar(20), -- name of the member
MContact_info varchar(30), -- email address
Join_date date
);
```



```sql
create table Staff(
Staff_id int not null primary key,
SName varchar(20),-- staff name
SContact_info varchar(30),
Job_title varchar(50),
Hire_date date
);

create  table Material(
Material_id int not null primary key,
Title varchar(60),
Publication_date date,
Catalog_id int,
Genre_id int,
foreign key(Catalog_id) references Catalog(Catalog_id),
foreign key(Genre_id) references Genre(Genre_id)
);

create table Borrow(
Borrow_id int not null primary key,
BMaterial_id int,
BMember_id int,
Bstaff_id int,
Borrow_date date,
Due_date date,
Return_date date,
foreign key(BMaterial_id) references Material(Material_id),
foreign key(Bmember_id) references Member(Member_id),
foreign key (Bstaff_id) references Staff(Staff_id)
);
```



```sql
create table Authorship(
Authorship_id int not null primary key,
AAuthor_id int,
AMaterial_id int,
foreign key(AAuthor_id) references Author(Author_id),
foreign key(AMaterial_id) references Material(Material_id)
);
```

3.3.3 Inserting the values.

The primary operation of a relational database is to add values to the table. Existing tables must be updated to include the data rows.

It is vital to bear in mind that the values given in the insert statement must match the data types of the table's fields. For instance, you cannot place a text value into a numeric column and vice versa.

Additionally, it must be certain that the values being added to the foreign key columns of a table with foreign key restrictions already exist in the table being referenced. The insert statement will be unsuccessful if the foreign key condition is not followed.

Query:

insert into Catalog(Catalog_ID, CName, Llocation)

values (1,'Books','A1.1'),

(2,'Magazines','B2.1'),

(3,'E-Books','C3.1'),

(4,'Audiobooks','D4.1'),

(5,'Journals','E5.1'),

(6,'Newspaper','F6.1'),

(7,'Maps','G7.1'),

(8,'Novels','H8.1'),

(9,'SheetMusic','I9.1'),

(10,'Educational','J10.1');

```
insert into Genre(Genre_id,GName,Description)


values(1,'General Fiction','Literary works with a focus on character and plot development, exploring various themes and human experiences.'),

(2,'Mystery & Thriller','Suspenseful stories centered around crime, investigation, or espionage with an emphasis on tension and excitement.'),

(3,'Science Fiction & Fantasy','Imaginative works that explore alternate realities, futuristic concepts, and magical or supernatural elements.'),

(4,'Horror & Suspense','Stories designed to evoke fear, unease, or dread, often featuring supernatural or psychological elements.'),

(5,'Dystopian & Apocalyptic','Depictions of societies in decline or collapse, often exploring themes of political and social oppression or environmental disaster.'),

(6,'Classics','Enduring works of literature that have stood the test of time, often featuring rich language and complex themes.'),

(7,'Historical Fiction','Fictional stories set in the past, often based on real historical events or figures, and exploring the customs and experiences of that time.'),

(8,'Epic Poetry & Mythology','Ancient or traditional stories and poems, often featuring heroes, gods, and mythical creatures, and exploring cultural values and beliefs');



insert into Author (Author_id,AName,Birthdate,Nationality)

values(1,'Jane Austen','1775-12-16','British'),

(2,'Ernest Hemingway','1899-07-21','American'),

(3,'George Orwell','1903-06-25','British'),

(4,'Scott Fitzgerald','1896-09-24','American'),

(5,'J.K. Rowling','1965-07-31','British'),

(6,'Mark Twain','1835-11-30','American'),

(7,'Leo Tolstoy','1828-09-09','Russian'),

(8,'Virginia Woolf','1882-01-25','British'),

(9,'Gabriel Márquez','1927-03-06','Colombian'),

(10,'Charles Dickens','1812-02-07','British'),

(11,'Harper Lee','1926-04-28','American'),

(12,'Oscar Wilde','1854-10-16','Irish'),

(13,'William Shakespeare','1564-04-26','British')

,(14,'Franz Kafka','1883-07-03','Czech')

,(15,'James Joyce','1882-02-02','Irish')

,(16,'J.R.R. Tolkien','1892-01-03','British')
```

```
,(17,'Emily Brontë','1818-07-30','British')
,(18,'Toni Morrison','1931-02-18','American')
,(19,'Fyodor Dostoevsky','1821-11-11','Russian')
,(20,'Lucas Piki','1847-10-16','British');


insert into Member(Member_id,MName,MContact_info,Join_Date)
values (1,'Alice Johnson','alice.johnson@email.com','2018-01-10'),
(2,'Bob Smith','bob.smith@email.com','2018-03-15'),
(3,'Carol Brown','carol.brown@email.com','2018-06-20'),
(4,'David Williams','david.williams@email.com', '2018-09-18'),
(5,'Emily Miller','emily.miller@email.com','2019-02-12'),
(6,'Frank Davis','frank.davis@email.com','2019-05-25'),
(7,'Grace Wilson','grace.wilson@email.com','2019-08-15'),
(8,'Harry Garcia','harry.garcia@email.com','2019-11-27'),
(9,'Isla Thomas','isla.thomas@email.com','2020-03-04'),
(10,'Jack Martinez','jack.martinez@email.com','2020-07-01'),
(11,'Kate Anderson','kate.anderson@email.com','2020-09-30'),
(12,'Luke Jackson','luke.jackson@email.com','2021-01-18'),
(13,'Mia White','mia.white@email.com','2021-04-27'),
(14,'Noah Harris','noah.harris@email.com','2021-07-13'),
(15,'Olivia Clark','olivia.clark@email.com','2021-10-05'),
(16,'Peter Lewis','peter.lewis@email.com','2021-12-01'),
(17,'Quinn Hall','quinn.hall@email.com','2022-02-28'),
(18,'Rachel Young','rachel.young@email.com','2022-06-17'),
(19,'Sam Walker','sam.walker@email.com','2022-09-25'),
(20,'Tiffany Allen','tiffany.allen@email.com','2022-12-10');


insert into Staff(Staff_id,SName,SContact_info,Job_title,Hire_date)
values(1,'Amy Green','amy.green@email.com','Librarian','2017-06-01'),
(2,'Brian Taylor','brian.taylor@email.com','Library Assistant','2018-11-15'),
(3,'Christine King','chris.king@email.com','Library Assistant','2019-05-20'),
(4,'Daniel Wright','dan.wright@email.com','Library Technician','2020-02-01');
```

```
alter table Material
alter column Title type varchar(100);

insert into Material(Material_id,Title,Publication_date,Catalog_id,Genre_iD)
values(1,'The Catcher in the Rye','1951-07-16',1,1),
(2,'To Kill a Mockingbird','1960-07-11',2,1),
(3,'The Da Vinci Code','2003-04-01',3,2),
(4,'The Hobbit','1937-09-21',4,3),
(5,'The Shining','1977-01-28',5,4),
(6,'Pride and Prejudice','1813-01-28',1,1),
(7,'The Great Gatsby','1925-04-10',2,1),
(8,'Moby Dick','1851-10-18',3,1),
(9,'Crime and Punishment','1866-01-01',4,1),
(10,'The Hitchhikers Guide to the Galaxy','1979-10-12',5,3),
(11,'1984','1949-06-08',1,5),
(12,'Animal Farm','1945-08-17',2,5),
(13,'The Haunting of Hill House','1959-10-17',3,4),
(14,'Brave New World','1932-08-01',4,5),
(15,'The Chronicles of Narnia: The Lion, the Witch and the Wardrobe','1950-10-16',5,3),
(16,'The Adventures of Huckleberry Finn','1884-12-10',6,1),
(17,'The Catch-22','1961-10-11',7,1),
(18,'The Picture of Dorian Gray','1890-07-01',8,1),
(19,'The Call of Cthulhu','1928-02-01',9,4),
(20,'Harry Potter and the Philosopher''s Stone','1997-06-26',10,3),
(21,'Frankenstein','1818-01-01',6,4),
(22,'A Tale of Two Cities','1859-04-30',7,1),
(23,'The Iliad','1750-01-01',8,6),
(24,'The Odyssey','1725-01-01',9,6),
(25,'The Brothers Karamazov','1880-01-01',10,1),
(26,'The Divine Comedy','1320-01-01',6,6),
(27,'The Grapes of Wrath','1939-04-14',7,1),
(28,'The Old Man and the Sea','1952-09-01',8,1),
(29,'The Count of Monte Cristo','1844-01-01',9,1),
(30,'A Midsummer Nights Dream','1596-01-01',10,7),
(31,'The Tricky Book','1888-01-01',10,7);

insert into
Borrow(Borrow_id,BMaterial_id,BMember_id,BStaff_id,Borrow_date,Due_date,Return_date)
values(1,1,1,1,'2018-09-12','2018-10-03','2018-09-30'),
(2,2,2,1,'2018-10-15','2018-11-05','2018-10-29'),
(3,3,3,1,'2018-12-20','2019-01-10','2019-01-08'),
(4,4,4,1,'2019-03-11','2019-04-01','2019-03-27'),
(5,5,5,1,'2019-04-20','2019-05-11','2019-05-05'),
(6,6,6,1,'2019-07-05','2019-07-26','2019-07-21'),
(7,7,7,1,'2019-09-10','2019-10-01','2019-09-25'),
(8,8,8,1,'2019-11-08','2019-11-29','2019-11-20'),
(9,9,9,1,'2020-01-15','2020-02-05','2020-02-03'),
(10,10,10,1,'2020-03-12','2020-04-02','2020-03-28'),
(11,1,11,2,'2020-05-14','2020-06-04','2020-05-28'),
(12,2,12,2,'2020-07-21','2020-08-11','2020-08-02'),
(13,3,13,2,'2020-09-25','2020-10-16','2020-10-15'),
(14,4,1,2,'2020-11-08','2020-11-29','2020-11-24'),
(15,5,2,2,'2021-01-03','2021-01-24','2021-01-19'),
(16,6,3,2,'2021-02-18','2021-03-11','2021-03-12'),
(17,17,4,2,'2021-04-27','2021-05-18','2021-05-20'),
(18,18,5,2,'2021-06-13','2021-07-04','2021-06-28'),
(19,19,6,2,'2021-08-15','2021-09-05','2021-09-03'),
(20,20,7,2,'2021-10-21','2021-11-11','2021-11-05'),
(21,21,1,3,'2021-11-29','2021-12-20',NULL),
(22,22,2,3,'2022-01-10','2022-01-31','2022-01-25'),
(23,23,3,3,'2022-02-07','2022-02-28','2022-02-23'),
(24,24,4,3,'2022-03-11','2022-04-01','2022-03-28'),
(25,25,5,3,'2022-04-28','2022-05-19','2022-05-18'),
(26,26,6,3,'2022-06-22','2022-07-13','2022-07-08'),
(27,27,7,3,'2022-08-04','2022-08-25','2022-08-23'),
(28,28,8,3,'2022-09-13','2022-10-04','2022-09-28'),
(29,29,9,3,'2022-10-16','2022-11-06','2022-11-05'),
(30,30,8,3,'2022-11-21','2022-12-12','2022-12-05'),
(31,1,9,4,'2022-12-28','2023-01-18',NULL),
(32,2,1,4,'2023-01-23','2023-02-13',NULL),
(33,3,10,4,'2023-02-02','2023-02-23','2023-02-17'),
(34,4,11,4,'2023-03-01','2023-03-22',NULL),
(35,5,12,4,'2023-03-10','2023-03-31',NULL),
(36,6,13,4,'2023-03-15','2023-04-05',NULL),
(37,7,17,4,'2023-03-25','2023-04-15',NULL),
(38,8,8,4,'2023-03-30','2023-04-20',NULL),
(39,9,9,4,'2023-03-26','2023-04-16',NULL),
(40,10,20,4,'2023-03-28','2023-04-18',NULL);

insert into Authorship(Authorship_id,AAuthor_id,AMaterial_id)

values (1,1,1),
(2,2,2),
(3,3,3),
(4,4,4),
(5,5,5),
(6,6,6),
(7,7,7),
(8,8,8),
(9,9,9),
(10,10,10),
(11,11,11),
(12,12,12),
(13,13,13),
(14,14,14),
(15,15,15),
(16,16,16),
(17,17,17),
(18,18,18),
(19,19,19),
(20,20,20),
```

(21,1,21),

(22,2,22),

(23,3,23),

(24,4,24),

(25,5,25),

(26,6,26),

(27,7,27),

(28,8,28),

(29,19,28),

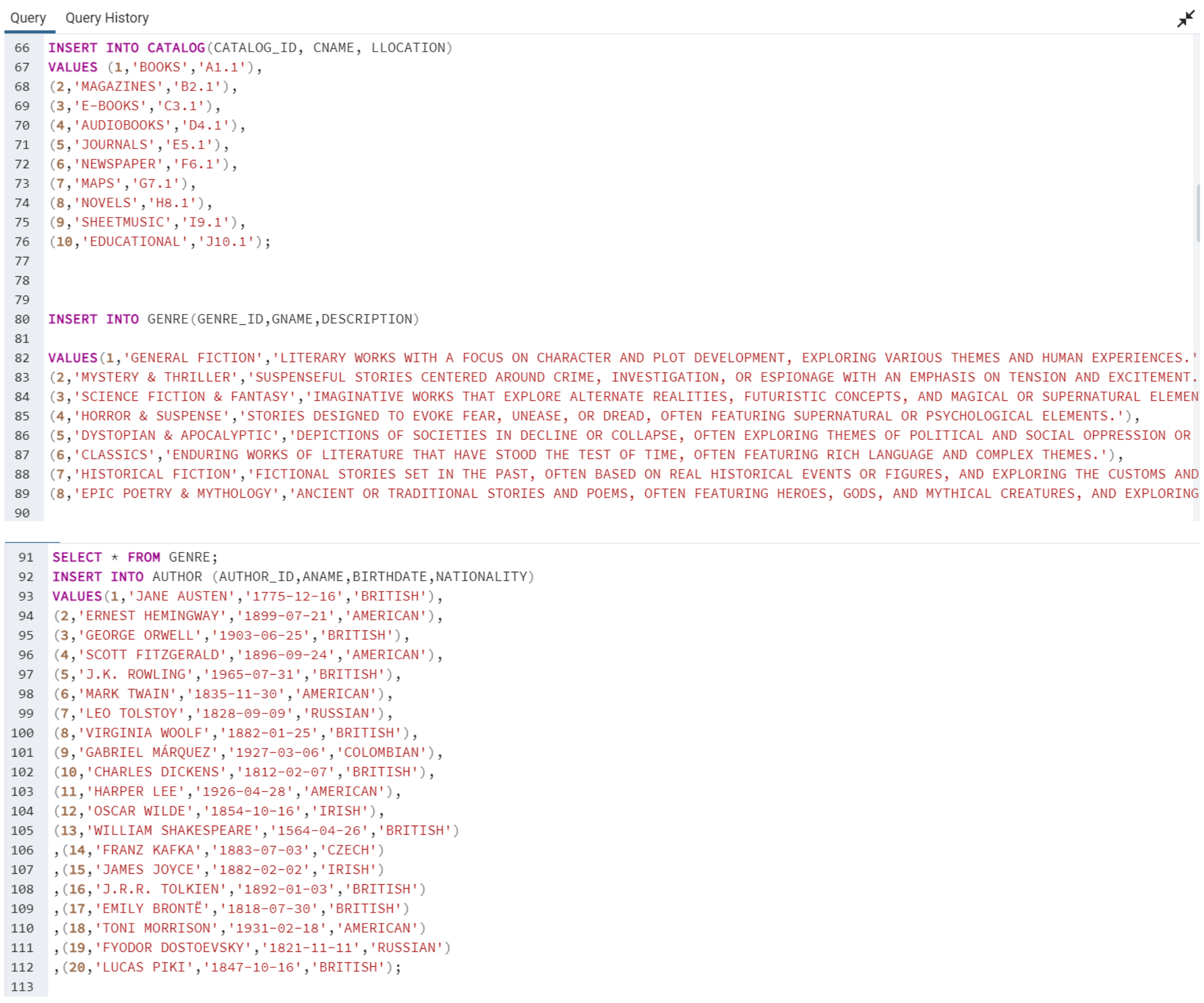
(30,9,29),

(31,10,30),

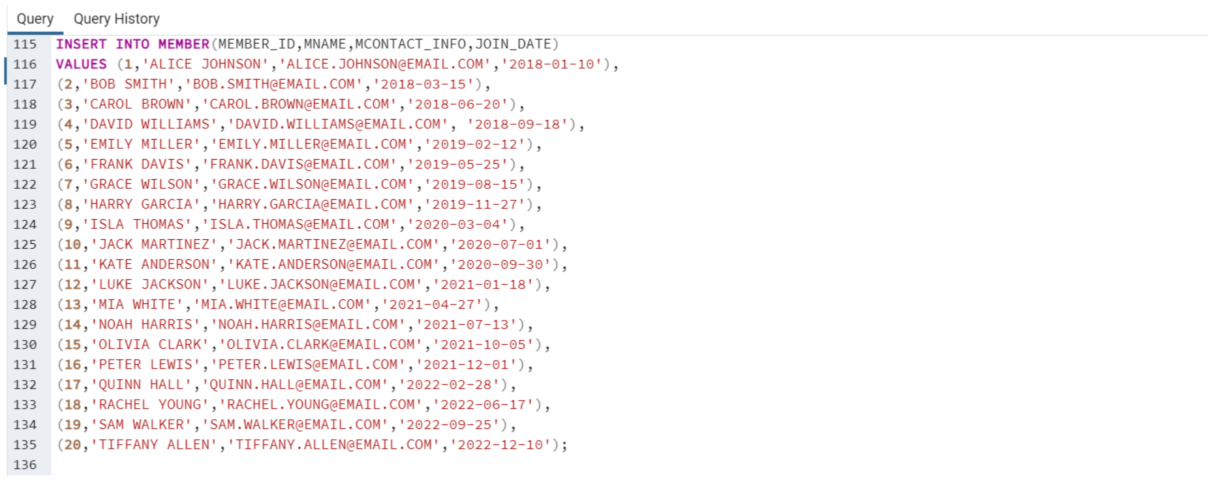
(32,8,30),

(33,2,29);

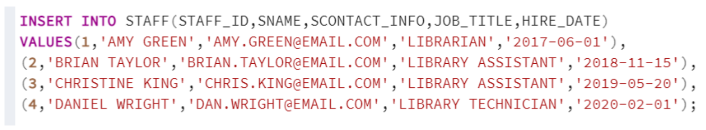
Outputs:



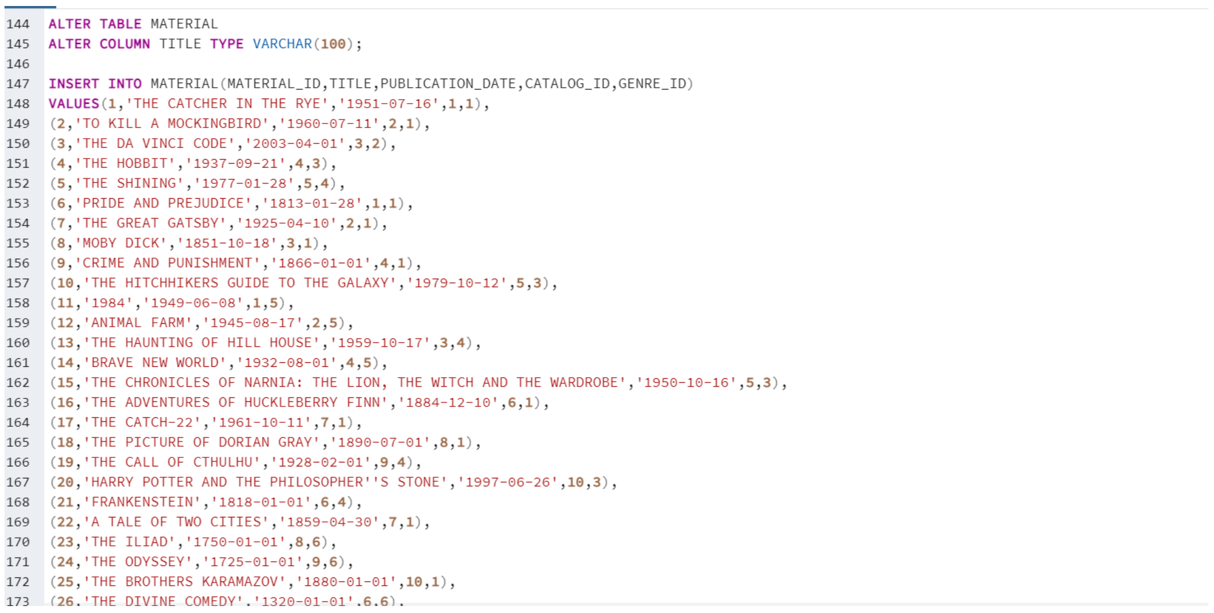
```
INSERT INTO CATALOG(CATALOG_ID, CNAME, LLOCATION)
VALUES (1,'BOOKS','A1.1'),
(2,'MAGAZINES','B2.1'),
(3,'E-BOOKS','C3.1'),
(4,'AUDIOBOOKS','D4.1'),
(5,'JOURNALS','E5.1'),
(6,'NEWSPAPER','F6.1'),
(7,'MAPS','G7.1'),
(8,'NOVELS','H8.1'),
(9,'SHEETMUSIC','I9.1'),
(10,'EDUCATIONAL','J10.1');



INSERT INTO GENRE(GENRE_ID,GNAME,DESCRIPTION)

VALUES(1,'GENERAL FICTION','LITERARY WORKS WITH A FOCUS ON CHARACTER AND PLOT DEVELOPMENT, EXPLORING VARIOUS THEMES AND HUMAN EXPERIENCES.'
(2,'MYSTERY & THRILLER','SUSPENSEFUL STORIES CENTERED AROUND CRIME, INVESTIGATION, OR ESPIONAGE WITH AN EMPHASIS ON TENSION AND EXCITEMENT.
(3,'SCIENCE FICTION & FANTASY','IMAGINATIVE WORKS THAT EXPLORE ALTERNATE REALITIES, FUTURISTIC CONCEPTS, AND MAGICAL OR SUPERNATURAL ELEMEN
(4,'HORROR & SUSPENSE','STORIES DESIGNED TO EVOKE FEAR, UNEASE, OR DREAD, OFTEN FEATURING SUPERNATURAL OR PSYCHOLOGICAL ELEMENTS.'),
(5,'DYSTOPIAN & APOCALYPTIC','DEPICTIONS OF SOCIETIES IN DECLINE OR COLLAPSE, OFTEN EXPLORING THEMES OF POLITICAL AND SOCIAL OPPRESSION OR
(6,'CLASSICS','ENDURING WORKS OF LITERATURE THAT HAVE STOOD THE TEST OF TIME, OFTEN FEATURING RICH LANGUAGE AND COMPLEX THEMES.'),
(7,'HISTORICAL FICTION','FICTIONAL STORIES SET IN THE PAST, OFTEN BASED ON REAL HISTORICAL EVENTS OR FIGURES, AND EXPLORING THE CUSTOMS AND
(8,'EPIC POETRY & MYTHOLOGY','ANCIENT OR TRADITIONAL STORIES AND POEMS, OFTEN FEATURING HEROES, GODS, AND MYTHICAL CREATURES, AND EXPLORING

SELECT * FROM GENRE;
INSERT INTO AUTHOR (AUTHOR_ID,ANAME,BIRTHDATE,NATIONALITY)
VALUES(1,'JANE AUSTEN','1775-12-16','BRITISH'),
(2,'ERNEST HEMINGWAY','1899-07-21','AMERICAN'),
(3,'GEORGE ORWELL','1903-06-25','BRITISH'),
(4,'SCOTT FITZGERALD','1896-09-24','AMERICAN'),
(5,'J.K. ROWLING','1965-07-31','BRITISH'),
(6,'MARK TWAIN','1835-11-30','AMERICAN'),
(7,'LEO TOLSTOY','1828-09-09','RUSSIAN'),
(8,'VIRGINIA WOOLF','1882-01-25','BRITISH'),
(9,'GABRIEL MÁRQUEZ','1927-03-06','COLOMBIAN'),
(10,'CHARLES DICKENS','1812-02-07','BRITISH'),
(11,'HARPER LEE','1926-04-28','AMERICAN'),
(12,'OSCAR WILDE','1854-10-16','IRISH'),
(13,'WILLIAM SHAKESPEARE','1564-04-26','BRITISH')
,(14,'FRANZ KAFKA','1883-07-03','CZECH')
,(15,'JAMES JOYCE','1882-02-02','IRISH')
,(16,'J.R.R. TOLKIEN','1892-01-03','BRITISH')
,(17,'EMILY BRONTË','1818-07-30','BRITISH')
,(18,'TONI MORRISON','1931-02-18','AMERICAN')
,(19,'FYODOR DOSTOEVSKY','1821-11-11','RUSSIAN')
,(20,'LUCAS PIKI','1847-10-16','BRITISH');
```



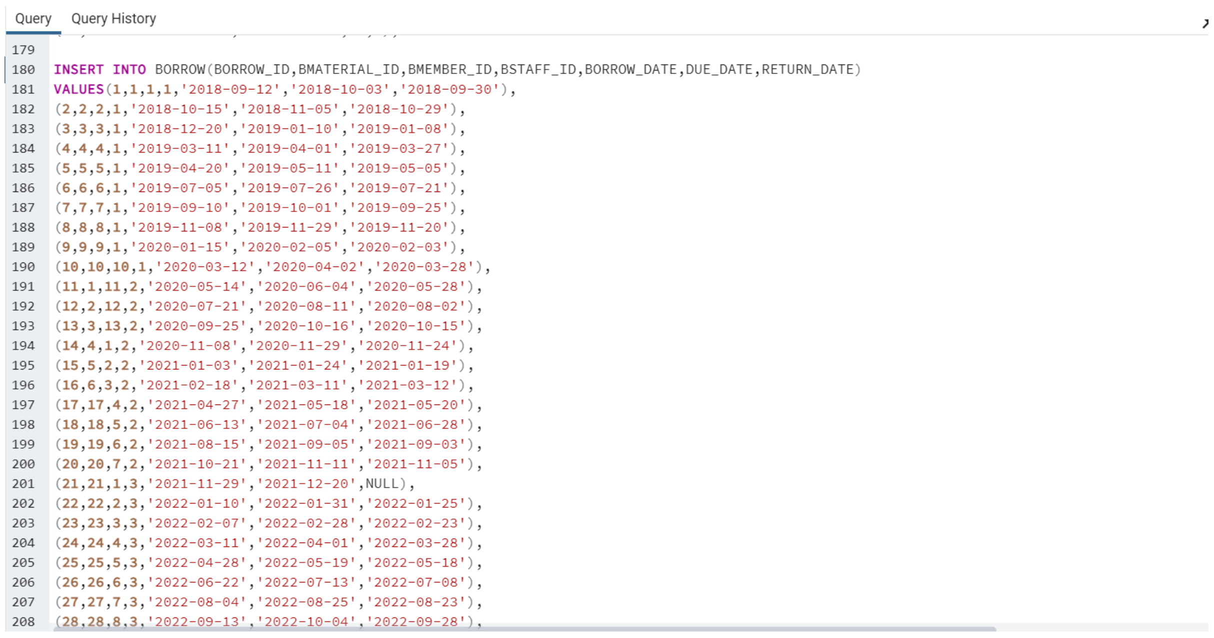
```sql
INSERT INTO MEMBER(MEMBER_ID,MNAME,MCONTACT_INFO,JOIN_DATE)
VALUES (1,'ALICE JOHNSON','ALICE.JOHNSON@EMAIL.COM','2018-01-10'),
(2,'BOB SMITH','BOB.SMITH@EMAIL.COM','2018-03-15'),
(3,'CAROL BROWN','CAROL.BROWN@EMAIL.COM','2018-06-20'),
(4,'DAVID WILLIAMS','DAVID.WILLIAMS@EMAIL.COM', '2018-09-18'),
(5,'EMILY MILLER','EMILY.MILLER@EMAIL.COM','2019-02-12'),
(6,'FRANK DAVIS','FRANK.DAVIS@EMAIL.COM','2019-05-25'),
(7,'GRACE WILSON','GRACE.WILSON@EMAIL.COM','2019-08-15'),
(8,'HARRY GARCIA','HARRY.GARCIA@EMAIL.COM','2019-11-27'),
(9,'ISLA THOMAS','ISLA.THOMAS@EMAIL.COM','2020-03-04'),
(10,'JACK MARTINEZ','JACK.MARTINEZ@EMAIL.COM','2020-07-01'),
(11,'KATE ANDERSON','KATE.ANDERSON@EMAIL.COM','2020-09-30'),
(12,'LUKE JACKSON','LUKE.JACKSON@EMAIL.COM','2021-01-18'),
(13,'MIA WHITE','MIA.WHITE@EMAIL.COM','2021-04-27'),
(14,'NOAH HARRIS','NOAH.HARRIS@EMAIL.COM','2021-07-13'),
(15,'OLIVIA CLARK','OLIVIA.CLARK@EMAIL.COM','2021-10-05'),
(16,'PETER LEWIS','PETER.LEWIS@EMAIL.COM','2021-12-01'),
(17,'QUINN HALL','QUINN.HALL@EMAIL.COM','2022-02-28'),
(18,'RACHEL YOUNG','RACHEL.YOUNG@EMAIL.COM','2022-06-17'),
(19,'SAM WALKER','SAM.WALKER@EMAIL.COM','2022-09-25'),
(20,'TIFFANY ALLEN','TIFFANY.ALLEN@EMAIL.COM','2022-12-10');
```



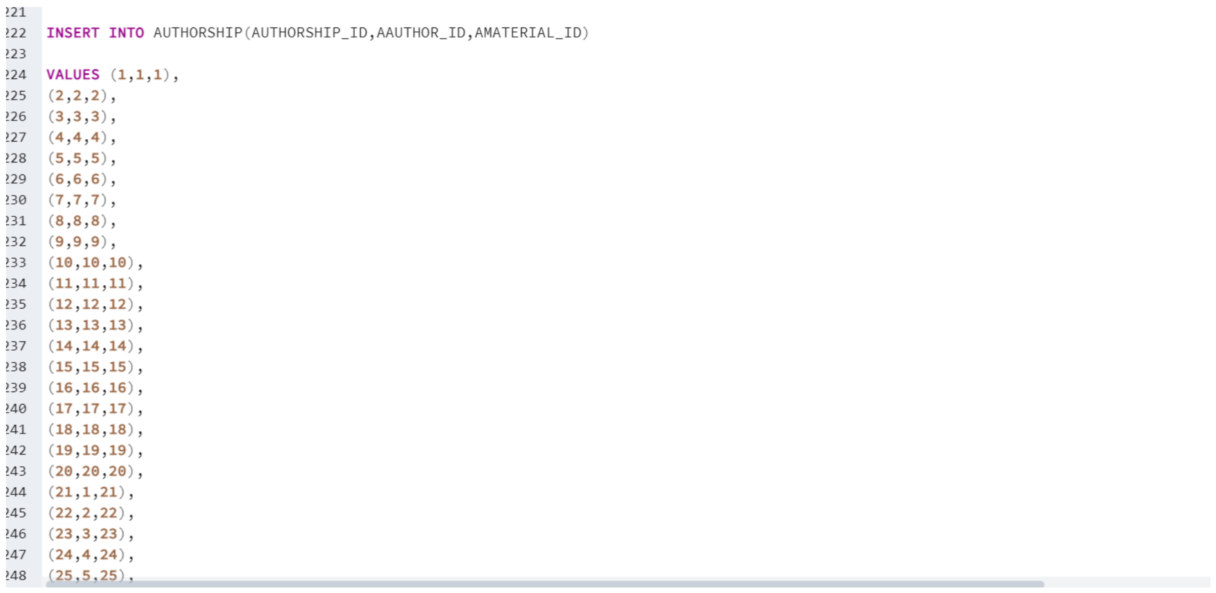
```sql
INSERT INTO STAFF(STAFF_ID,SNAME,SCONTACT_INFO,JOB_TITLE,HIRE_DATE)
VALUES(1,'AMY GREEN','AMY.GREEN@EMAIL.COM','LIBRARIAN','2017-06-01'),
(2,'BRIAN TAYLOR','BRIAN.TAYLOR@EMAIL.COM','LIBRARY ASSISTANT','2018-11-15'),
(3,'CHRISTINE KING','CHRIS.KING@EMAIL.COM','LIBRARY ASSISTANT','2019-05-20'),
(4,'DANIEL WRIGHT','DAN.WRIGHT@EMAIL.COM','LIBRARY TECHNICIAN','2020-02-01');
```



```sql
ALTER TABLE MATERIAL
ALTER COLUMN TITLE TYPE VARCHAR(100);

INSERT INTO MATERIAL(MATERIAL_ID,TITLE,PUBLICATION_DATE,CATALOG_ID,GENRE_ID)
VALUES(1,'THE CATCHER IN THE RYE','1951-07-16',1,1),
(2,'TO KILL A MOCKINGBIRD','1960-07-11',2,1),
(3,'THE DA VINCI CODE','2003-04-01',3,2),
(4,'THE HOBBIT','1937-09-21',4,3),
(5,'THE SHINING','1977-01-28',5,4),
(6,'PRIDE AND PREJUDICE','1813-01-28',1,1),
(7,'THE GREAT GATSBY','1925-04-10',2,1),
(8,'MOBY DICK','1851-10-18',3,1),
(9,'CRIME AND PUNISHMENT','1866-01-01',4,1),
(10,'THE HITCHHIKERS GUIDE TO THE GALAXY','1979-10-12',5,3),
(11,'1984','1949-06-08',1,5),
(12,'ANIMAL FARM','1945-08-17',2,5),
(13,'THE HAUNTING OF HILL HOUSE','1959-10-17',3,4),
(14,'BRAVE NEW WORLD','1932-08-01',4,5),
(15,'THE CHRONICLES OF NARNIA: THE LION, THE WITCH AND THE WARDROBE','1950-10-16',5,3),
(16,'THE ADVENTURES OF HUCKLEBERRY FINN','1884-12-10',6,1),
(17,'THE CATCH-22','1961-10-11',7,1),
(18,'THE PICTURE OF DORIAN GRAY','1890-07-01',8,1),
(19,'THE CALL OF CTHULHU','1928-02-01',9,4),
(20,'HARRY POTTER AND THE PHILOSOPHER''S STONE','1997-06-26',10,3),
(21,'FRANKENSTEIN','1818-01-01',6,4),
(22,'A TALE OF TWO CITIES','1859-04-30',7,1),
(23,'THE ILIAD','1750-01-01',8,6),
(24,'THE ODYSSEY','1725-01-01',9,6),
(25,'THE BROTHERS KARAMAZOV','1880-01-01',10,1),
(26,'THE DIVINE COMEDY','1320-01-01',6,6),
```



```
INSERT INTO BORROW(BORROW_ID,BMATERIAL_ID,BMEMBER_ID,BSTAFF_ID,BORROW_DATE,DUE_DATE,RETURN_DATE)
VALUES(1,1,1,1,'2018-09-12','2018-10-03','2018-09-30'),
(2,2,2,1,'2018-10-15','2018-11-05','2018-10-29'),
(3,3,3,1,'2018-12-20','2019-01-10','2019-01-08'),
(4,4,4,1,'2019-03-11','2019-04-01','2019-03-27'),
(5,5,5,1,'2019-04-20','2019-05-11','2019-05-05'),
(6,6,6,1,'2019-07-05','2019-07-26','2019-07-21'),
(7,7,7,1,'2019-09-10','2019-10-01','2019-09-25'),
(8,8,8,1,'2019-11-08','2019-11-29','2019-11-20'),
(9,9,9,1,'2020-01-15','2020-02-05','2020-02-03'),
(10,10,10,1,'2020-03-12','2020-04-02','2020-03-28'),
(11,1,11,2,'2020-05-14','2020-06-04','2020-05-28'),
(12,2,12,2,'2020-07-21','2020-08-11','2020-08-02'),
(13,3,13,2,'2020-09-25','2020-10-16','2020-10-15'),
(14,4,1,2,'2020-11-08','2020-11-29','2020-11-24'),
(15,5,2,2,'2021-01-03','2021-01-24','2021-01-19'),
(16,6,3,2,'2021-02-18','2021-03-11','2021-03-12'),
(17,17,4,2,'2021-04-27','2021-05-18','2021-05-20'),
(18,18,5,2,'2021-06-13','2021-07-04','2021-06-28'),
(19,19,6,2,'2021-08-15','2021-09-05','2021-09-03'),
(20,20,7,2,'2021-10-21','2021-11-11','2021-11-05'),
(21,21,1,3,'2021-11-29','2021-12-20',NULL),
(22,22,2,3,'2022-01-10','2022-01-31','2022-01-25'),
(23,23,3,3,'2022-02-07','2022-02-28','2022-02-23'),
(24,24,4,3,'2022-03-11','2022-04-01','2022-03-28'),
(25,25,5,3,'2022-04-28','2022-05-19','2022-05-18'),
(26,26,6,3,'2022-06-22','2022-07-13','2022-07-08'),
(27,27,7,3,'2022-08-04','2022-08-25','2022-08-23'),
(28,28,8,3,'2022-09-13','2022-10-04','2022-09-28'),
```



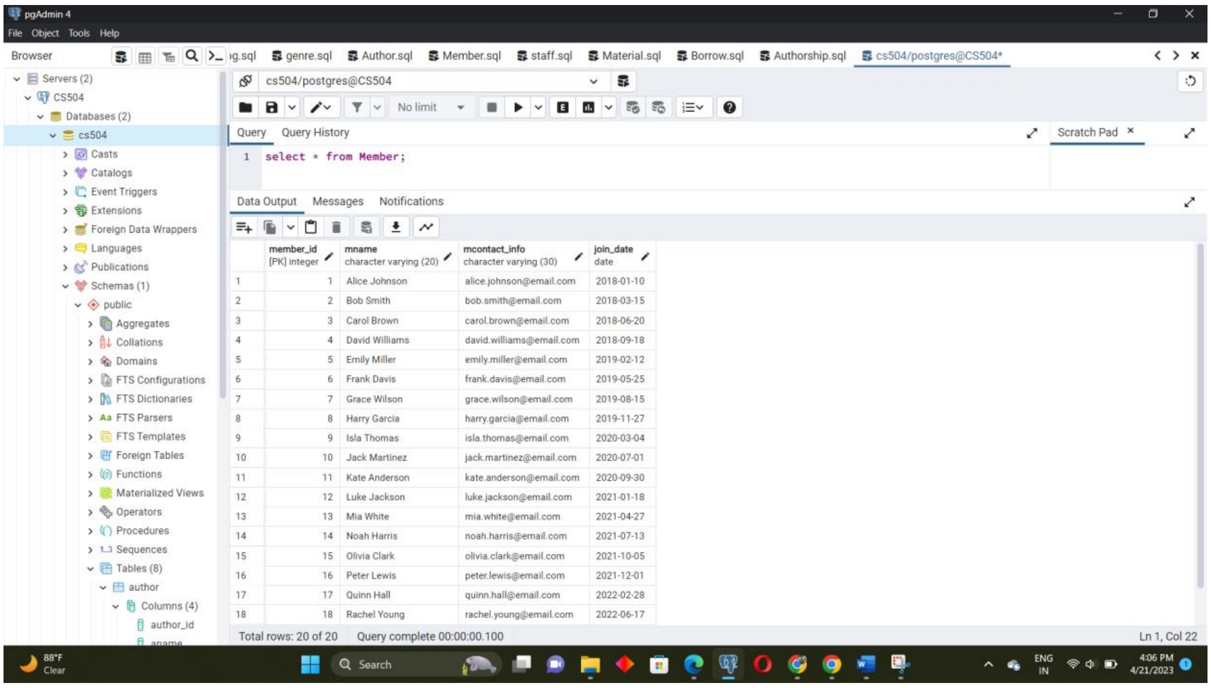
```
INSERT INTO AUTHORSHIP(AUTHORSHIP_ID,AAUTHOR_ID,AMATERIAL_ID)

VALUES (1,1,1),
(2,2,2),
(3,3,3),
(4,4,4),
(5,5,5),
(6,6,6),
(7,7,7),
(8,8,8),
(9,9,9),
(10,10,10),
(11,11,11),
(12,12,12),
(13,13,13),
(14,14,14),
(15,15,15),
(16,16,16),
(17,17,17),
(18,18,18),
(19,19,19),
(20,20,20),
(21,1,21),
(22,2,22),
(23,3,23),
(24,4,24),
(25,5,25),
```

4 Queries and Updates

4.0.1. Basic

Select:

Searching operation in the SQL is used to search the data according to the requirement. “select” is the primary command used to search the data.

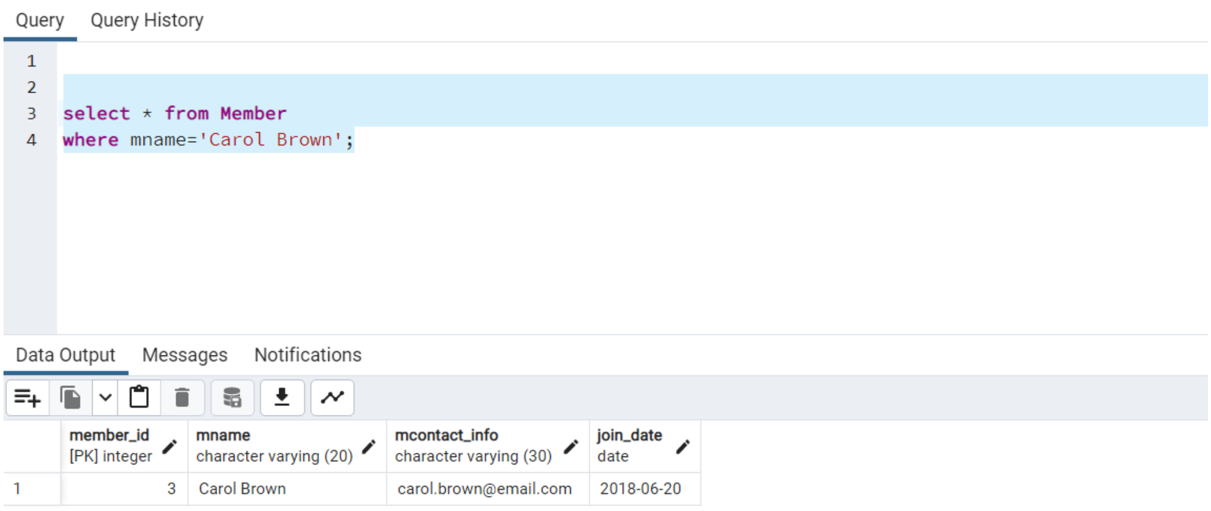


```
select * from Member;
```

	member_id [PK] integer	mname character varying (20)	mcontact_info character varying (30)	join_date date
1	1	Alice Johnson	alice.johnson@email.com	2018-01-10
2	2	Bob Smith	bob.smith@email.com	2018-03-15
3	3	Carol Brown	carol.brown@email.com	2018-06-20
4	4	David Williams	david.williams@email.com	2018-09-18
5	5	Emily Miller	emily.miller@email.com	2019-02-12
6	6	Frank Davis	frank.davis@email.com	2019-05-25
7	7	Grace Wilson	grace.wilson@email.com	2019-08-15
8	8	Harry Garcia	harry.garcia@email.com	2019-11-27
9	9	Isla Thomas	isla.thomas@email.com	2020-03-04
10	10	Jack Martinez	jack.martinez@email.com	2020-07-01
11	11	Kate Anderson	kate.anderson@email.com	2020-09-30
12	12	Luke Jackson	luke.jackson@email.com	2021-01-18
13	13	Mia White	mia.white@email.com	2021-04-27
14	14	Noah Harris	noah.harris@email.com	2021-07-13
15	15	Olivia Clark	olivia.clark@email.com	2021-10-05
16	16	Peter Lewis	peter.lewis@email.com	2021-12-01
17	17	Quinn Hall	quinn.hall@email.com	2022-02-28
18	18	Rachel Young	rachel.young@email.com	2022-06-17

Total rows: 20 of 20   Query complete 00:00:00.100

Searching a data using the condition,

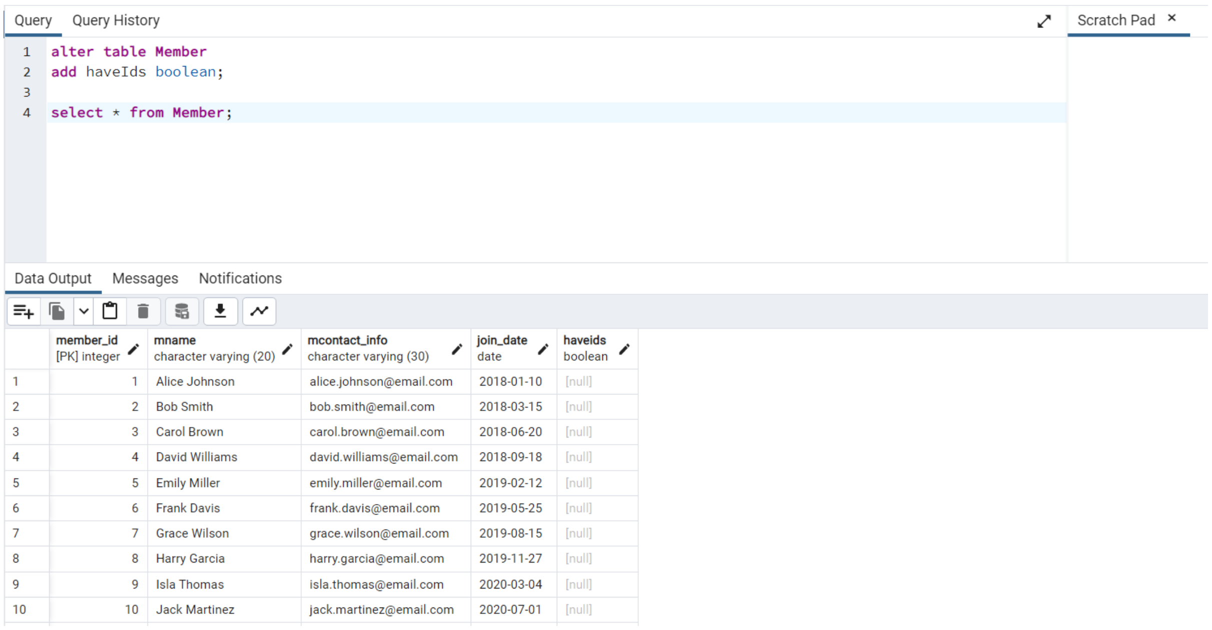


```


select * from Member
where mname='Carol Brown';
```

	member_id [PK] integer	mname character varying (20)	mcontact_info character varying (30)	join_date date
1	3	Carol Brown	carol.brown@email.com	2018-06-20

Alter:

This command is used to make modification in the current existing database, like adding a column in the table.(I have create a column haveIds in tables Member)

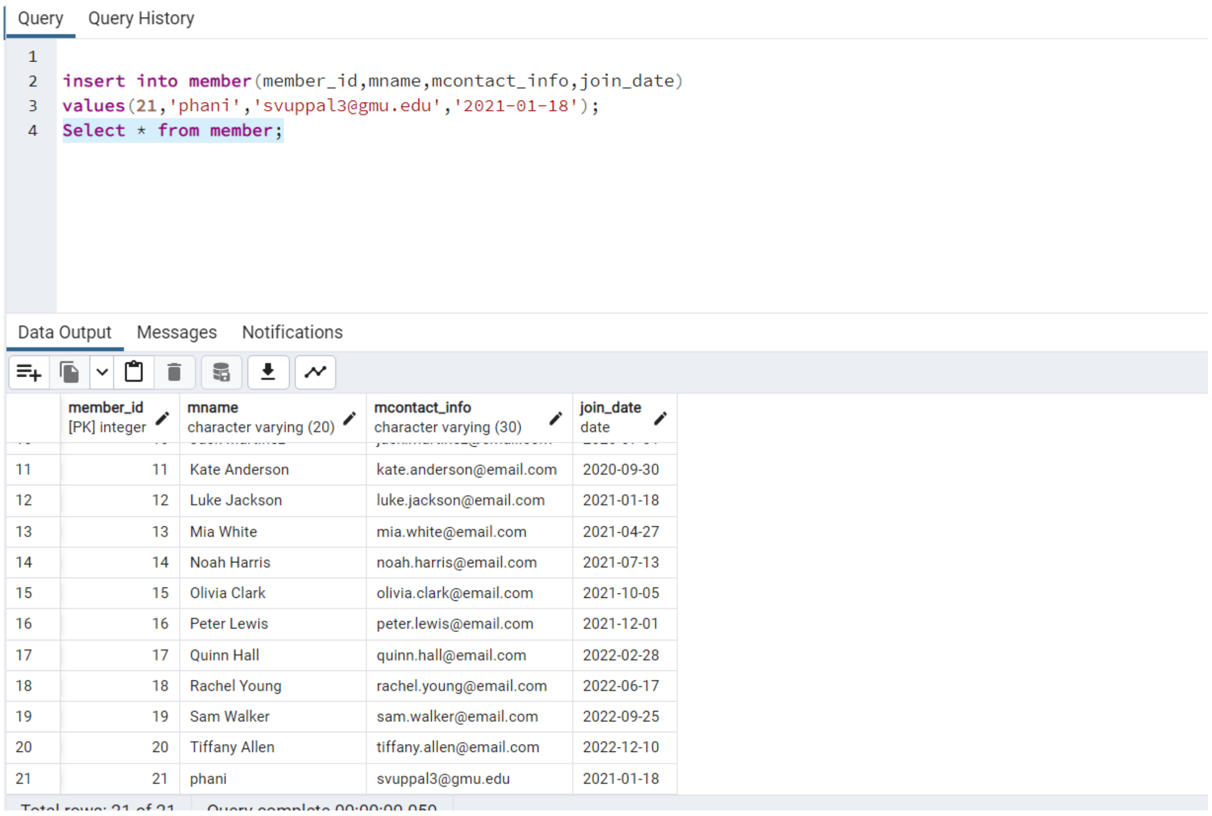


```
alter table Member
add haveIds boolean;

select * from Member;
```

	member_id [PK] integer	mname character varying (20)	mcontact_info character varying (30)	join_date date	haveids boolean
1	1	Alice Johnson	alice.johnson@email.com	2018-01-10	[null]
2	2	Bob Smith	bob.smith@email.com	2018-03-15	[null]
3	3	Carol Brown	carol.brown@email.com	2018-06-20	[null]
4	4	David Williams	david.williams@email.com	2018-09-18	[null]
5	5	Emily Miller	emily.miller@email.com	2019-02-12	[null]
6	6	Frank Davis	frank.davis@email.com	2019-05-25	[null]
7	7	Grace Wilson	grace.wilson@email.com	2019-08-15	[null]
8	8	Harry Garcia	harry.garcia@email.com	2019-11-27	[null]
9	9	Isla Thomas	isla.thomas@email.com	2020-03-04	[null]
10	10	Jack Martinez	jack.martinez@email.com	2020-07-01	[null]

Inserting records:

A record is inserted

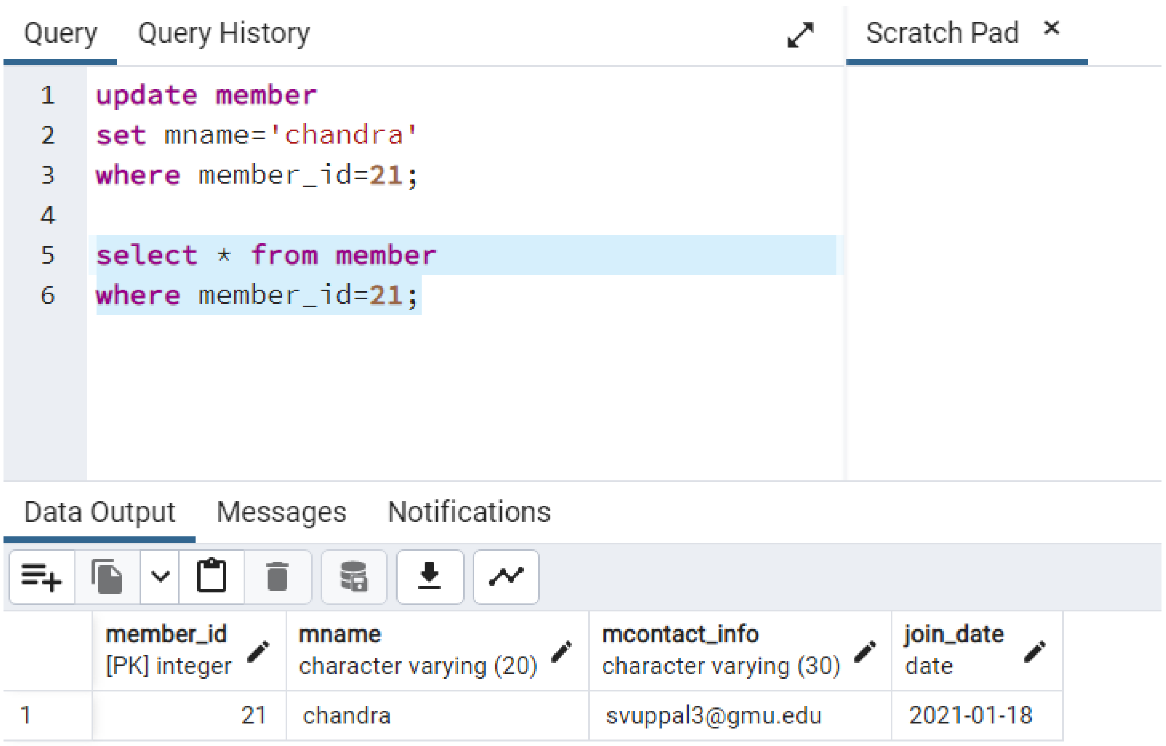


```

insert into member(member_id,mname,mcontact_info,join_date)
values(21,'phani','svuppal3@gmu.edu','2021-01-18');
Select * from member;
```

	member_id [PK] integer	mname character varying (20)	mcontact_info character varying (30)	join_date date
11	11	Kate Anderson	kate.anderson@email.com	2020-09-30
12	12	Luke Jackson	luke.jackson@email.com	2021-01-18
13	13	Mia White	mia.white@email.com	2021-04-27
14	14	Noah Harris	noah.harris@email.com	2021-07-13
15	15	Olivia Clark	olivia.clark@email.com	2021-10-05
16	16	Peter Lewis	peter.lewis@email.com	2021-12-01
17	17	Quinn Hall	quinn.hall@email.com	2022-02-28
18	18	Rachel Young	rachel.young@email.com	2022-06-17
19	19	Sam Walker	sam.walker@email.com	2022-09-25
20	20	Tiffany Allen	tiffany.allen@email.com	2022-12-10
21	21	phani	svuppal3@gmu.edu	2021-01-18

Total rows: 21 of 21 Query complete 00:00:00.050

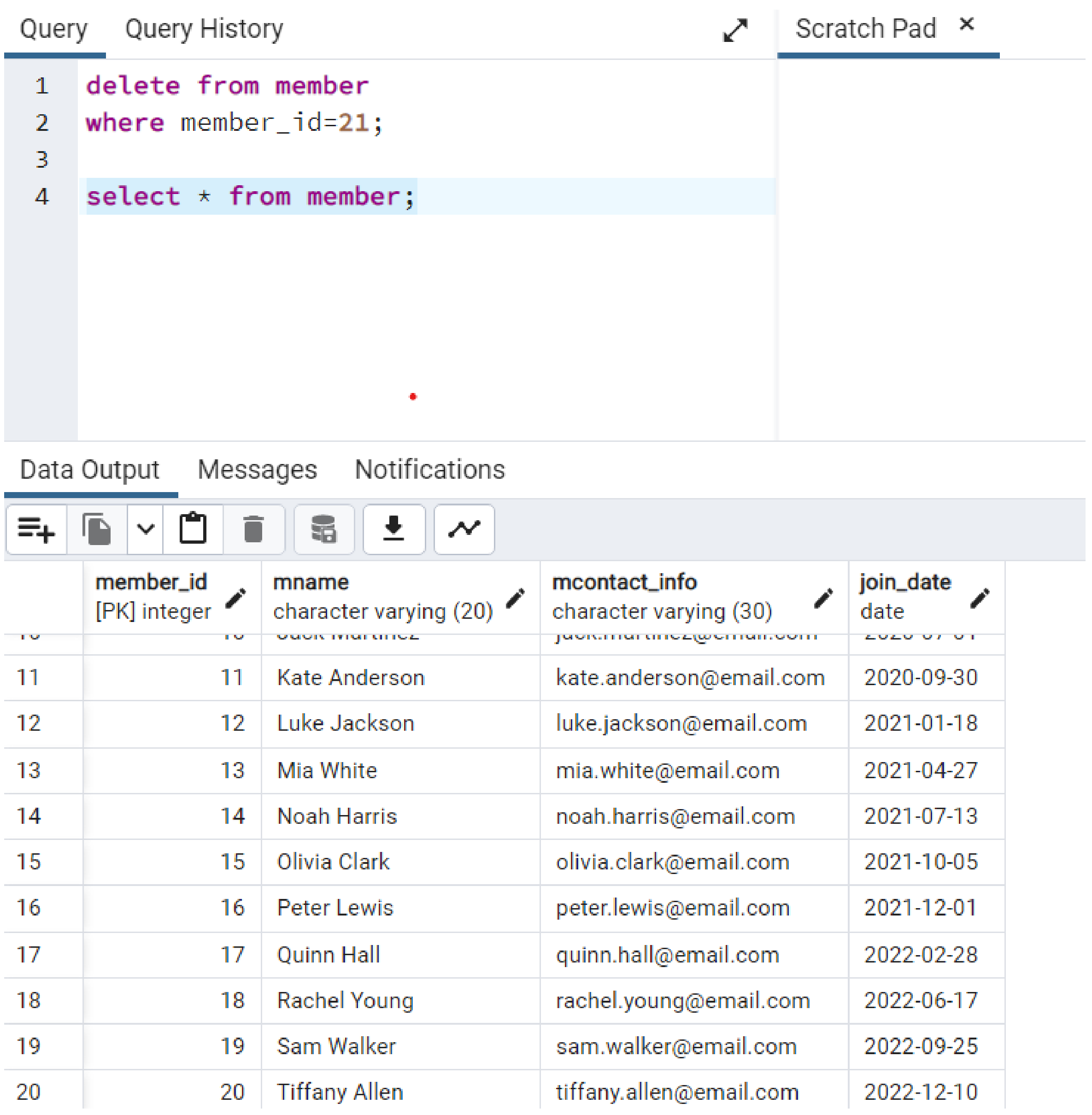
Updating records:



```
update member
set mname='chandra'
where member_id=21;

select * from member
where member_id=21;
```

	member_id [PK] integer	mname character varying (20)	mcontact_info character varying (30)	join_date date
1	21	chandra	svuppal3@gmu.edu	2021-01-18

Deleting records:



```sql
delete from member
where member_id=21;

select * from member;
```

	member_id [PK] integer	mname character varying (20)	mcontact_info character varying (30)	join_date date
11	11	Kate Anderson	kate.anderson@email.com	2020-09-30
12	12	Luke Jackson	luke.jackson@email.com	2021-01-18
13	13	Mia White	mia.white@email.com	2021-04-27
14	14	Noah Harris	noah.harris@email.com	2021-07-13
15	15	Olivia Clark	olivia.clark@email.com	2021-10-05
16	16	Peter Lewis	peter.lewis@email.com	2021-12-01
17	17	Quinn Hall	quinn.hall@email.com	2022-02-28
18	18	Rachel Young	rachel.young@email.com	2022-06-17
19	19	Sam Walker	sam.walker@email.com	2022-09-25
20	20	Tiffany Allen	tiffany.allen@email.com	2022-12-10

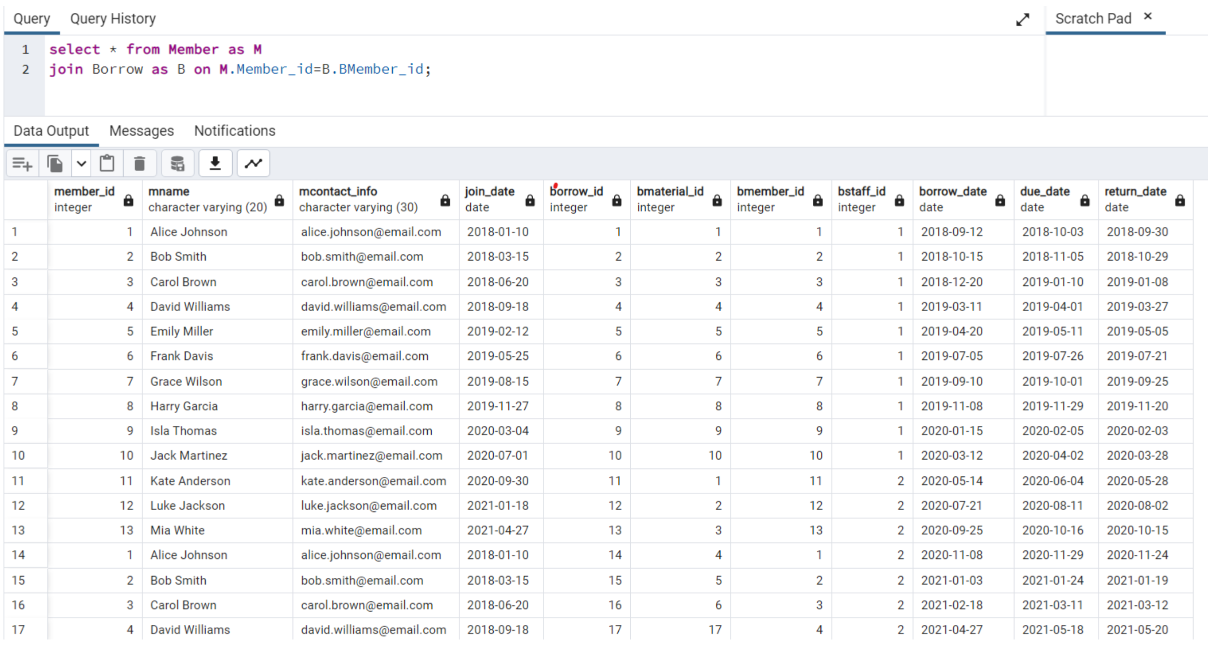
4.0.2. Advance querying techniques:

a) Joining Tables:

Query:

select * from Member as M

join Borrow as B on M.Member_id=B.BMember_id;


Query Query History Scratch Pad

```
select * from Member as M
join Borrow as B on M.Member_id=B.BMember_id;
```

Data Output Messages Notifications

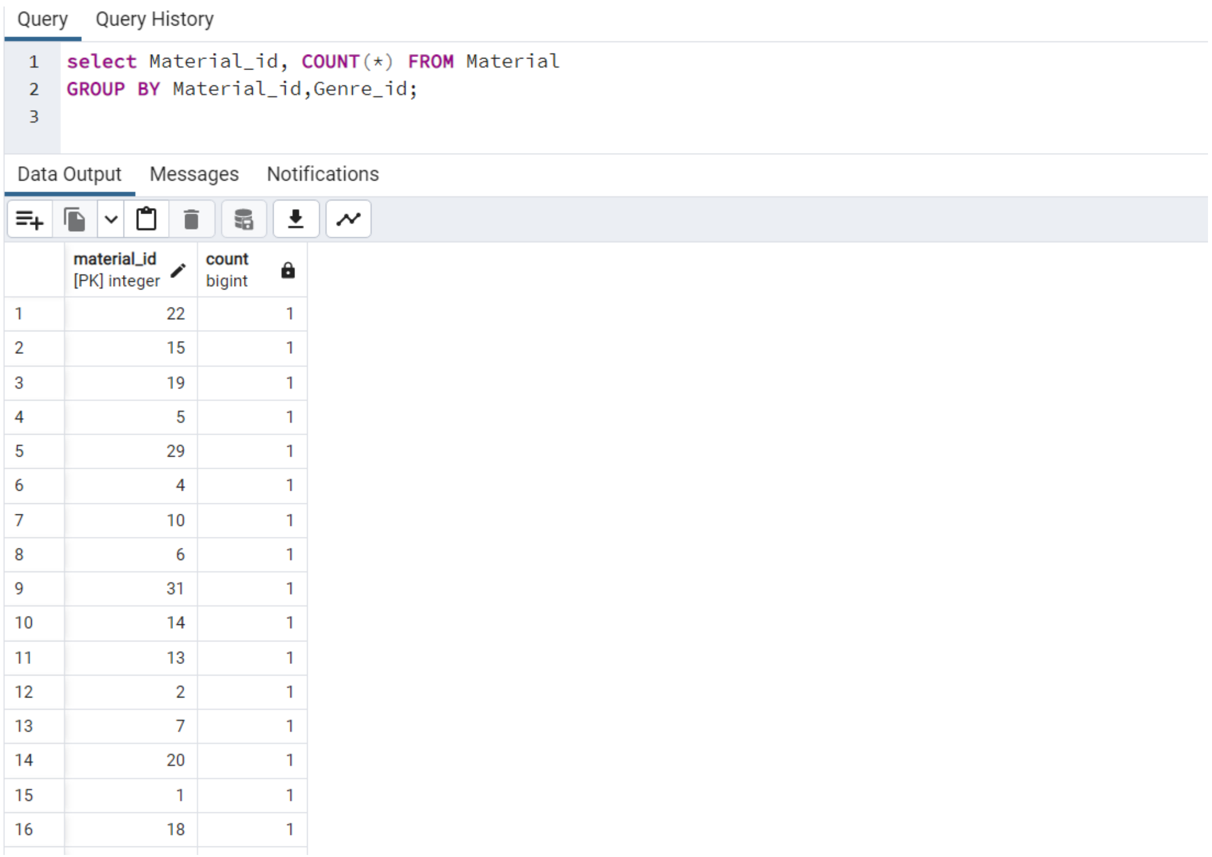
	member_id integer	mname character varying (20)	mcontact_info character varying (30)	join_date date	borrow_id integer	bmaterial_id integer	bmember_id integer	bstaff_id integer	borrow_date date	due_date date	return_date date
1	1	Alice Johnson	alice.johnson@email.com	2018-01-10	1	1	1	1	2018-09-12	2018-10-03	2018-09-30
2	2	Bob Smith	bob.smith@email.com	2018-03-15	2	2	2	1	2018-10-15	2018-11-05	2018-10-29
3	3	Carol Brown	carol.brown@email.com	2018-06-20	3	3	3	1	2018-12-20	2019-01-10	2019-01-08
4	4	David Williams	david.williams@email.com	2018-09-18	4	4	4	1	2019-03-11	2019-04-01	2019-03-27
5	5	Emily Miller	emily.miller@email.com	2019-02-12	5	5	5	1	2019-04-20	2019-05-11	2019-05-05
6	6	Frank Davis	frank.davis@email.com	2019-05-25	6	6	6	1	2019-07-05	2019-07-26	2019-07-21
7	7	Grace Wilson	grace.wilson@email.com	2019-08-15	7	7	7	1	2019-09-10	2019-10-01	2019-09-25
8	8	Harry Garcia	harry.garcia@email.com	2019-11-27	8	8	8	1	2019-11-08	2019-11-29	2019-11-20
9	9	Isla Thomas	isla.thomas@email.com	2020-03-04	9	9	9	1	2020-01-15	2020-02-05	2020-02-03
10	10	Jack Martinez	jack.martinez@email.com	2020-07-01	10	10	10	1	2020-03-12	2020-04-02	2020-03-28
11	11	Kate Anderson	kate.anderson@email.com	2020-09-30	11	1	11	2	2020-05-14	2020-06-04	2020-05-28
12	12	Luke Jackson	luke.jackson@email.com	2021-01-18	12	2	12	2	2020-07-21	2020-08-11	2020-08-02
13	13	Mia White	mia.white@email.com	2021-04-27	13	3	13	2	2020-09-25	2020-10-16	2020-10-15
14	1	Alice Johnson	alice.johnson@email.com	2018-01-10	14	4	1	2	2020-11-08	2020-11-29	2020-11-24
15	2	Bob Smith	bob.smith@email.com	2018-03-15	15	5	2	2	2021-01-03	2021-01-24	2021-01-19
16	3	Carol Brown	carol.brown@email.com	2018-06-20	16	6	3	2	2021-02-18	2021-03-11	2021-03-12
17	4	David Williams	david.williams@email.com	2018-09-18	17	17	4	2	2021-04-27	2021-05-18	2021-05-20

b) Aggregating data:

Query:

select Material_id, COUNT(*) FROM Material

GROUP BY Material_id,Genre_id;


Query Query History

```
select Material_id, COUNT(*) FROM Material
GROUP BY Material_id,Genre_id;
```

Data Output Messages Notifications

	material_id [PK] integer	count bigint
1	22	1
2	15	1
3	19	1
4	5	1
5	29	1
6	4	1
7	10	1
8	6	1
9	31	1
10	14	1
11	13	1
12	2	1
13	7	1
14	20	1
15	1	1
16	18	1

3) Using Subqueries:

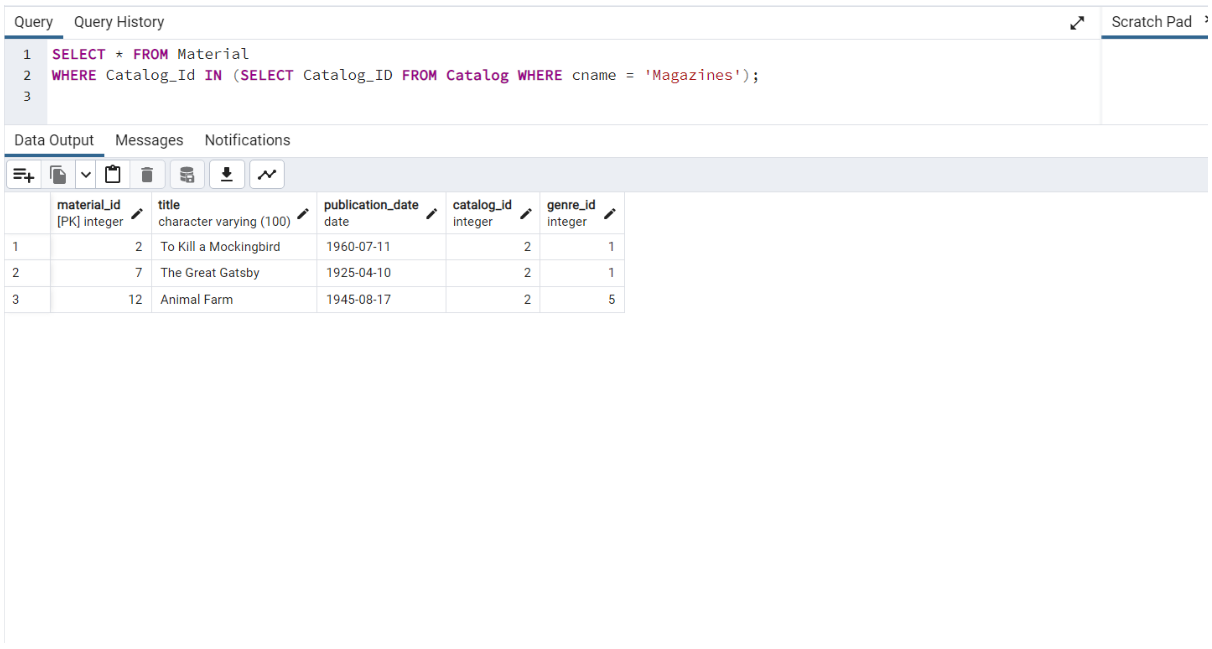
Query:

SELECT * FROM Material

WHERE Catalog_Id IN (SELECT Catalog_ID FROM Catalog WHERE cname = 'Magazines');

Explanation:

This SQL query will retrieve all rows from the "Material" table where the "Catalog_Id" column matches a value in the "Catalog_Id" column of the "Catalog" table where "cname" equals "Magazines".



```
SELECT * FROM Material
WHERE Catalog_Id IN (SELECT Catalog_ID FROM Catalog WHERE cname = 'Magazines');
```

	material_id [PK] integer	title character varying (100)	publication_date date	catalog_id integer	genre_id integer
1	2	To Kill a Mockingbird	1960-07-11	2	1
2	7	The Great Gatsby	1925-04-10	2	1
3	12	Animal Farm	1945-08-17	2	5

4.1. Queries/Updates

4.1.1 Which materials are currently available in the library?

Query:

SELECT TITLE FROM MATERIAL

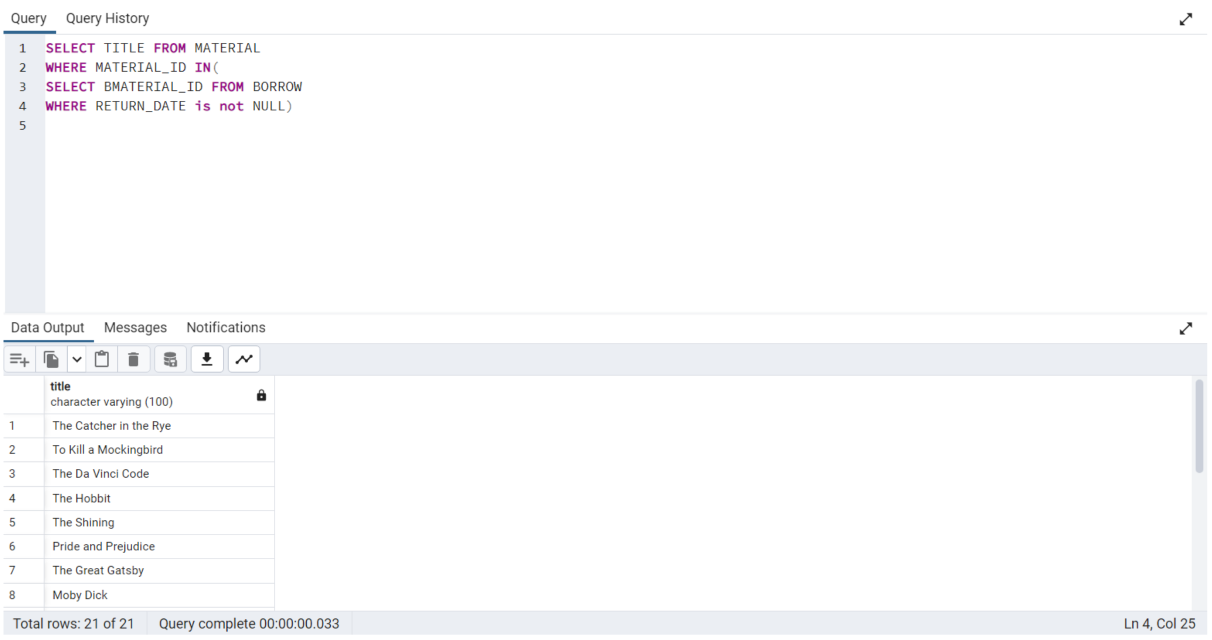
WHERE MATERIAL_ID IN(

SELECT BMATERIAL_ID FROM BORROW

WHERE RETURN_DATE is not NULL)

Explanation: In this query, the inner query(inside the bracket) is executed first, It returns the material_id from the borrow table where return date is null. Later it compare all the ids with the material ids from the material table, and returns all the titles which matches.

Output:



```
SELECT TITLE FROM MATERIAL
WHERE MATERIAL_ID IN(
SELECT BMATERIAL_ID FROM BORROW
WHERE RETURN_DATE is not NULL)
```

	title character varying (100)
1	The Catcher in the Rye
2	To Kill a Mockingbird
3	The Da Vinci Code
4	The Hobbit
5	The Shining
6	Pride and Prejudice
7	The Great Gatsby
8	Moby Dick

Total rows: 21 of 21 Query complete 00:00:00.033 Ln 4, Col 25

4.1.2 Which materials are currently overdue? Suppose today is 04/01/2023, and show the borrow date and due date of each material?

Query:

SELECT M.TITLE, B.BORROW_DATE, B.DUE_DATE

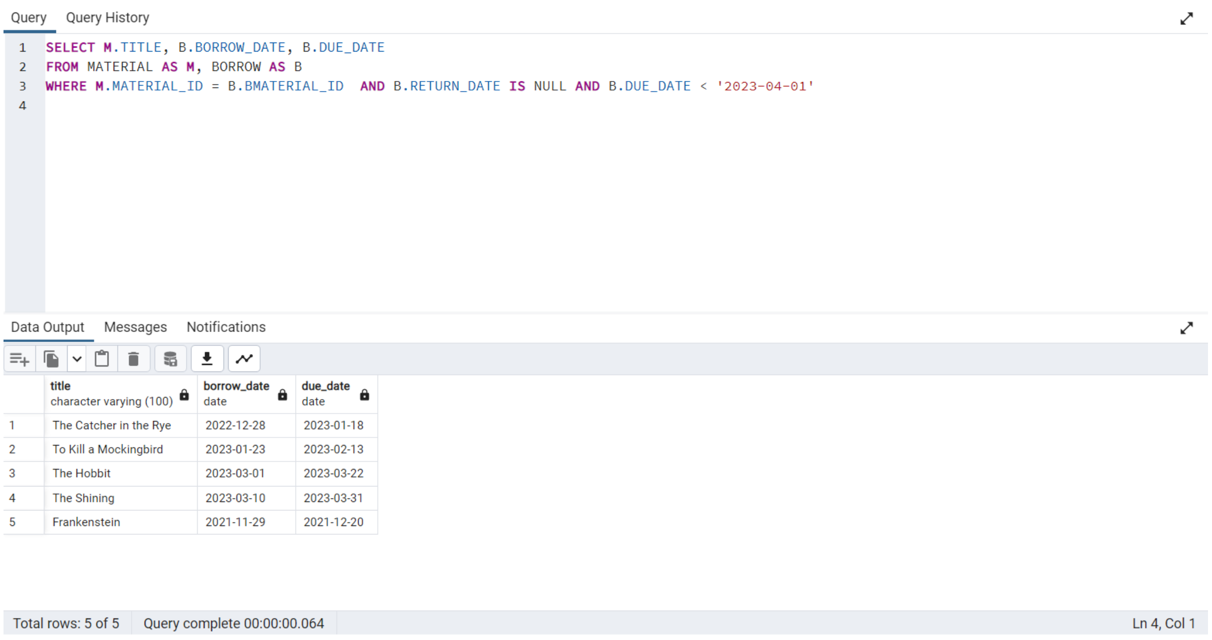
FROM MATERIAL AS M, BORROW AS B

WHERE M.MATERIAL_ID = B.BMATERIAL_ID AND B.RETURN_DATE IS NULL AND B.DUE_DATE < '2023-04-01'

Explanation:

This query returns the borrow date, due date from the borrow table and title from material table, If due date is less than 2023-04-01 and return date is NULL.

Output:



```
SELECT M.TITLE, B.BORROW_DATE, B.DUE_DATE
FROM MATERIAL AS M, BORROW AS B
WHERE M.MATERIAL_ID = B.BMATERIAL_ID  AND B.RETURN_DATE IS NULL AND B.DUE_DATE < '2023-04-01'
```

	title character varying (100)	borrow_date date	due_date date
1	The Catcher in the Rye	2022-12-28	2023-01-18
2	To Kill a Mockingbird	2023-01-23	2023-02-13
3	The Hobbit	2023-03-01	2023-03-22
4	The Shining	2023-03-10	2023-03-31
5	Frankenstein	2021-11-29	2021-12-20

Total rows: 5 of 5 Query complete 00:00:00.064 Ln 4, Col 1

4.1.3 What are the top 10 most borrowed materials in the library? Show the title of each material and order them based on their available counts ?

Query:

SELECT M.TITLE, COUNT(*) AS BORROW_COUNT

FROM MATERIAL AS M

INNER JOIN BORROW AS B ON M.MATERIAL_ID = B.BMATERIAL_ID

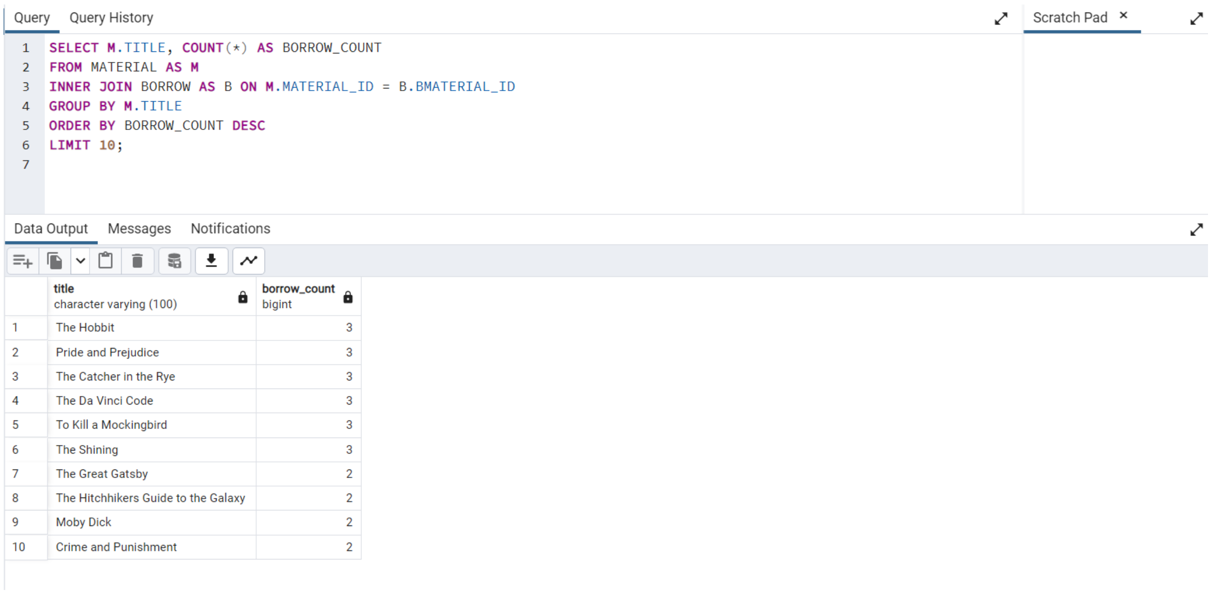
GROUP BY M.TITLE

ORDER BY BORROW_COUNT DESC

LIMIT 10;

Explanation: This query joins the materials ids from material and borrow tables, then group by titles and count them and top 10 are extracted in descending order.

Output:



```
SELECT M.TITLE, COUNT(*) AS BORROW_COUNT
FROM MATERIAL AS M
INNER JOIN BORROW AS B ON M.MATERIAL_ID = B.BMATERIAL_ID
GROUP BY M.TITLE
ORDER BY BORROW_COUNT DESC
LIMIT 10;
```

	title character varying (100)	borrow_count bigint
1	The Hobbit	3
2	Pride and Prejudice	3
3	The Catcher in the Rye	3
4	The Da Vinci Code	3
5	To Kill a Mockingbird	3
6	The Shining	3
7	The Great Gatsby	2
8	The Hitchhikers Guide to the Galaxy	2
9	Moby Dick	2
10	Crime and Punishment	2

4.1.4 How many books has the author Lucas Piki written?

Query:

SELECT NAME,COUNT(M.Material_ID)

FROM Material AS M

JOIN Authorship ON M.Material_ID = Authorship.Material_ID

JOIN Author ON Authorship.Author_ID = Author.Author_ID

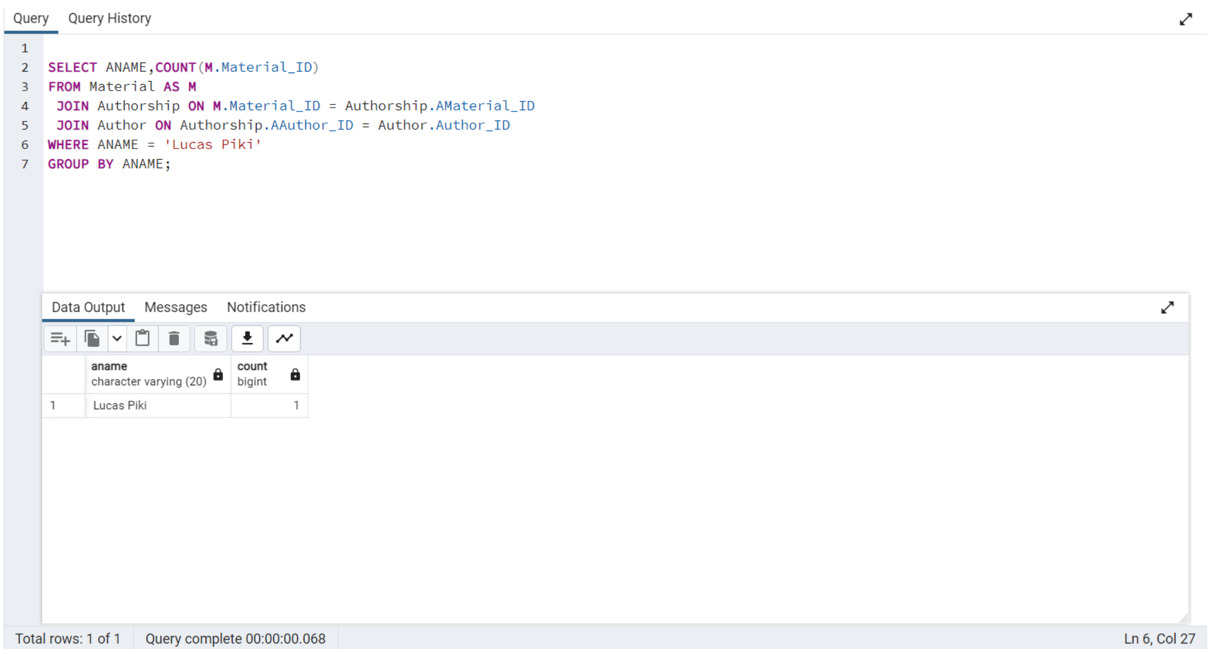
WHERE NAME = 'Lucas Piki'

GROUP BY NAME;

Explanation:

Here, we joined 3 tables material, author, authorship, and counted all the materials written by lucas pikki.

Output:


Query Query History

```
SELECT ANAME,COUNT(M.Material_ID)
FROM Material AS M
 JOIN Authorship ON M.Material_ID = Authorship.AMaterial_ID
 JOIN Author ON Authorship.AAuthor_ID = Author.Author_ID
WHERE ANAME = 'Lucas Piki'
GROUP BY ANAME;
```

Data Output Messages Notifications

	aname character varying (20)	count bigint
1	Lucas Piki	1

Total rows: 1 of 1 Query complete 00:00:00.068 Ln 6, Col 27

4.1.5 How many books were written by two or more authors?

Query:

SELECT COUNT(*)

FROM (

SELECT AMATERIAL_ID

FROM AUTHORSHIP

GROUP BY AMATERIAL_ID

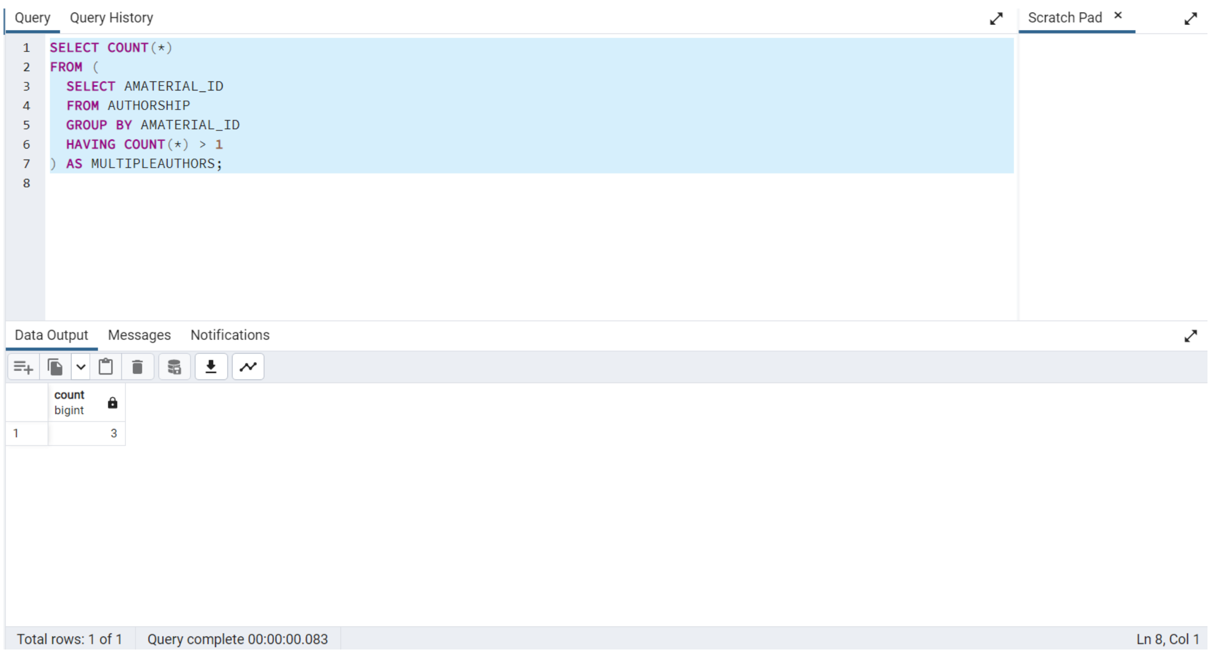
HAVING COUNT(*) > 1

) AS MULTIPLEAUTHORS;

Explanation:

In the inner query, the material ids are selected from the authorship table and grouped together by material from authorship, in outer query the table extracted from inner table is counted.

Output:



```
SELECT COUNT(*)
FROM (
  SELECT AMATERIAL_ID
  FROM AUTHORSHIP
  GROUP BY AMATERIAL_ID
  HAVING COUNT(*) > 1
) AS MULTIPLEAUTHORS;
```

```
  count
  bigint
1      3
```

Total rows: 1 of 1   Query complete 00:00:00.083   Ln 8, Col 1

4.1.6 What are the most popular genres in the library?

Query:

SELECT G.GNAME, COUNT(*) AS BORROW_COUNT

FROM GENRE AS G, MATERIAL AS M

WHERE G.GENRE_ID=M.GENRE_ID

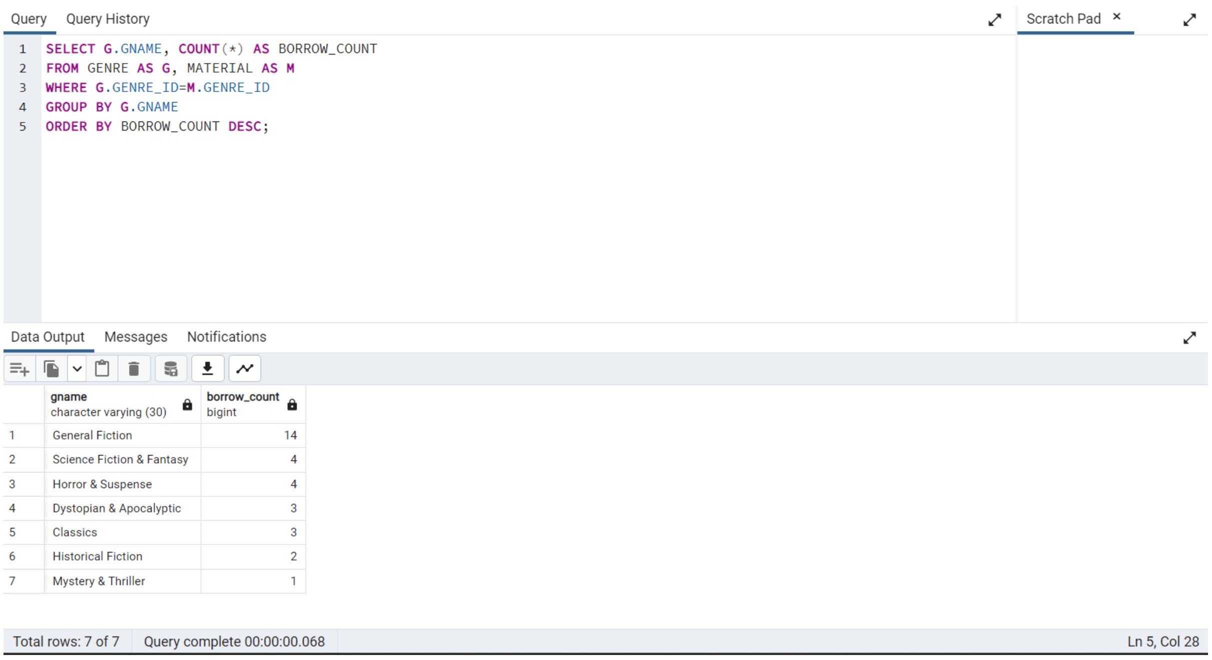
GROUP BY G.GNAME

ORDER BY BORROW_COUNT DESC;

Explanation:

This query joins the GENRE and MATERIAL tables on the GENRE_ID field and retrieves the name of each genre and the number of times that any material with that genre has been borrowed. It groups the records by genre name and orders them in descending order by the borrow count. Therefore, the result will show the most popular genres in the library based on the number of times they have been borrowed.

Output:



```
SELECT G.GNAME, COUNT(*) AS BORROW_COUNT
FROM GENRE AS G, MATERIAL AS M
WHERE G.GENRE_ID=M.GENRE_ID
GROUP BY G.GNAME
ORDER BY BORROW_COUNT DESC;
```

	gname character varying (30)	borrow_count bigint
1	General Fiction	14
2	Science Fiction & Fantasy	4
3	Horror & Suspense	4
4	Dystopian & Apocalyptic	3
5	Classics	3
6	Historical Fiction	2
7	Mystery & Thriller	1

Total rows: 7 of 7 Query complete 00:00:00.068 Ln 5, Col 28

4.1.7 How many materials have been borrowed from 09/2020-10/2020?

Query:

SELECT M.TITLE, COUNT(*)

FROM BORROW AS B

INNER JOIN MATERIAL AS M ON M.MATERIAL_ID= B.BMATERIAL_ID

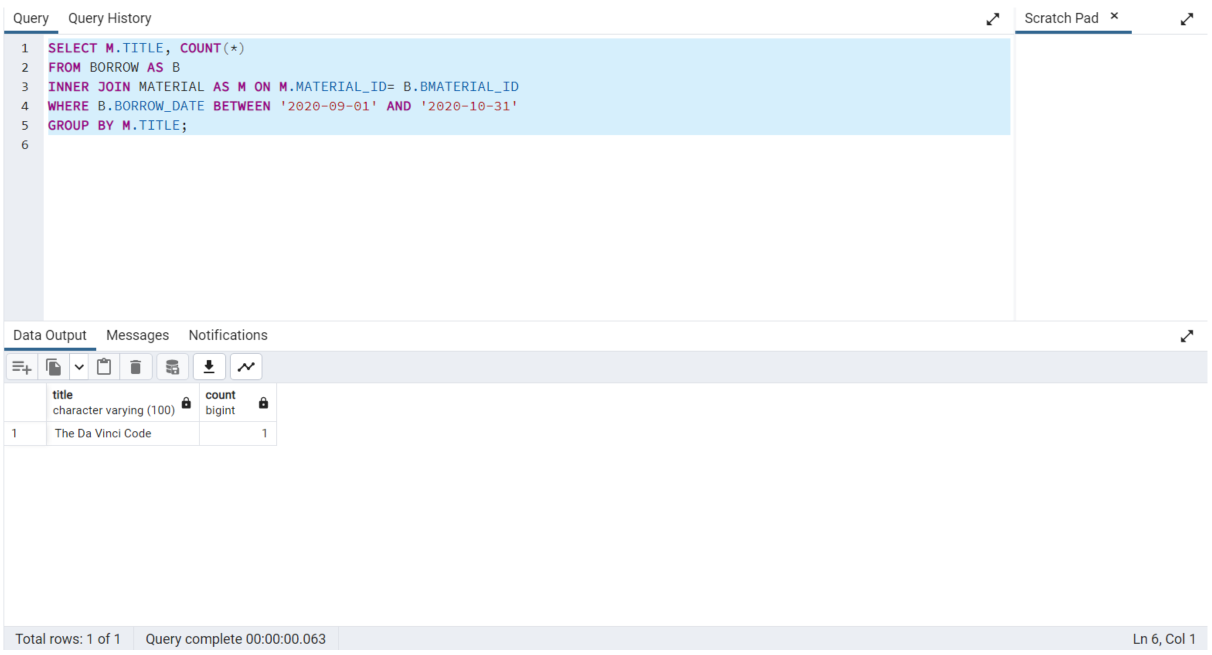
WHERE B.BORROW_DATE BETWEEN '2020-09-01' AND '2020-10-31'

GROUP BY M.TITLE;

Explanation:

It joins the BORROW and MATERIAL tables on the MATERIAL_ID and BMATERIAL_ID fields, respectively, and filters only the records where BORROW_DATE is between September 1, 2020, and October 31, 2020. It then groups the records by title and counts the number of times each title appears. Finally, it orders the results by title in ascending order.

Output:



```
SELECT M.TITLE, COUNT(*)
FROM BORROW AS B
INNER JOIN MATERIAL AS M ON M.MATERIAL_ID= B.BMATERIAL_ID
WHERE B.BORROW_DATE BETWEEN '2020-09-01' AND '2020-10-31'
GROUP BY M.TITLE;
```

	title character varying (100)	count bigint
1	The Da Vinci Code	1

Total rows: 1 of 1 Query complete 00:00:00.063 Ln 6, Col 1

4.1.8 How do you update the “Harry Potter and the Philosopher's Stone” when it is returned on 04/01/2023?

Query:

UPDATE BORROW

SET RETURN_DATE = '2023-04-01'

WHERE BMATERIAL_ID = (SELECT MATERIAL_ID FROM MATERIAL WHERE TITLE = 'Harry Potter and the Philosophers Stone');

SELECT M.MATERIAL_ID,M.TITLE, B.RETURN_DATE

FROM MATERIAL AS M

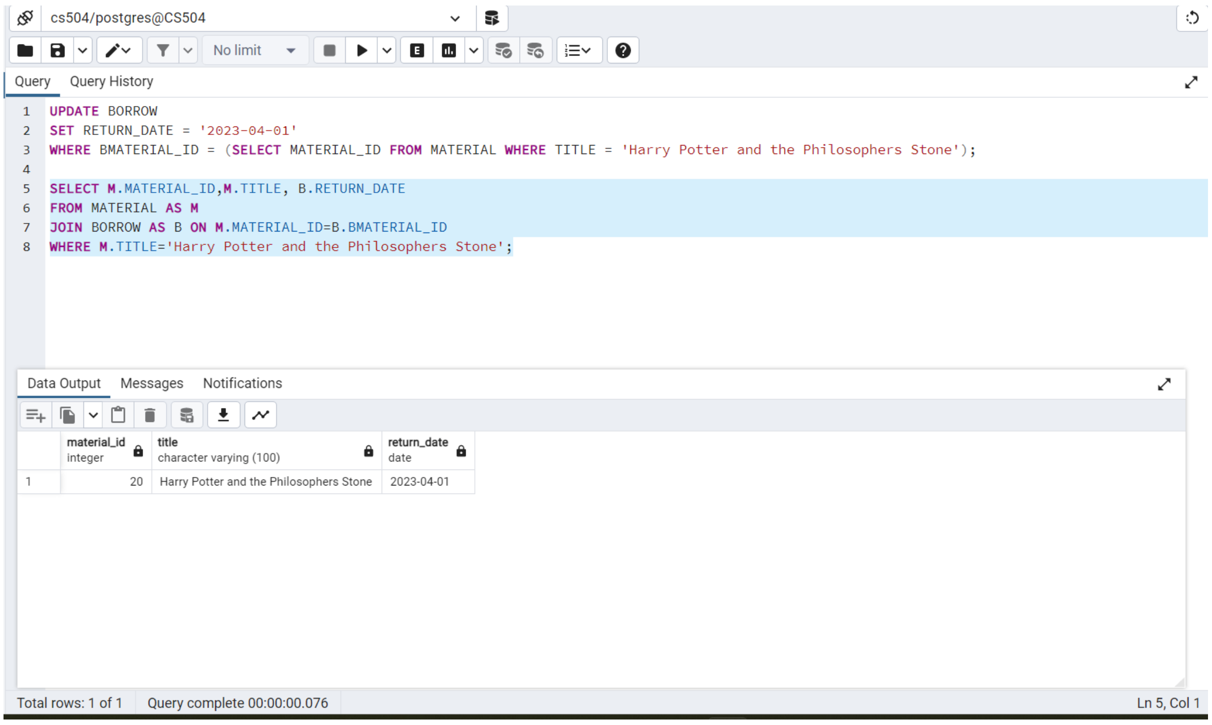
JOIN BORROW AS B ON M.MATERIAL_ID=B.BMATERIAL_ID

WHERE M.TITLE='Harry Potter and the Philosophers Stone';

Explanation:

This query will update the BORROW table by setting the RETURN_DATE to '2023-04-01' for the book with the title Harry Potter and the Philosopher's Stone that has not been returned yet. Note that this assumes that there is only one book with the title 'Harry Potter and the Philosopher's Stone' in the library and that the book has been borrowed but not returned yet. If there are multiple copies of the book or if the book has already been returned, the query may need to be modified accordingly.

Output:



cs504/postgres@CS504

Query | Query History

```
UPDATE BORROW
SET RETURN_DATE = '2023-04-01'
WHERE BMATERIAL_ID = (SELECT MATERIAL_ID FROM MATERIAL WHERE TITLE = 'Harry Potter and the Philosophers Stone');

SELECT M.MATERIAL_ID,M.TITLE, B.RETURN_DATE
FROM MATERIAL AS M
JOIN BORROW AS B ON M.MATERIAL_ID=B.BMATERIAL_ID
WHERE M.TITLE='Harry Potter and the Philosophers Stone';
```

Data Output | Messages | Notifications

	material_id integer	title character varying (100)	return_date date
1	20	Harry Potter and the Philosophers Stone	2023-04-01

Total rows: 1 of 1 | Query complete 00:00:00.076 | Ln 5, Col 1

4.1.9 How do you delete the member Emily Miller and all her related records from the database?

Query:

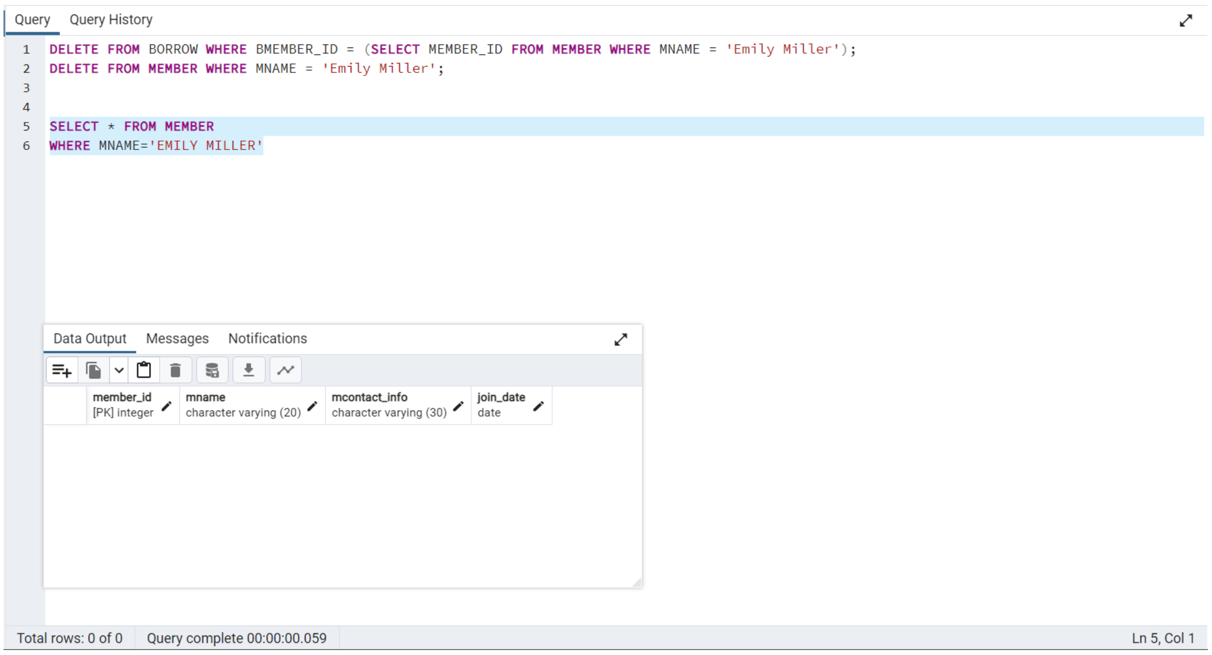
DELETE FROM BORROW WHERE BMEMBER_ID = (SELECT MEMBER_ID FROM MEMBER WHERE MNAME = 'Emily Miller');

DELETE FROM MEMBER WHERE MNAME = 'Emily Miller';

Explanation:

The first command deletes all the borrow records associated with Emily Miller by using her member ID, which is obtained through a subquery. The second command deletes the member record itself by matching her name to the MNAME attribute in the MEMBER table.

Output:


Query Query History

```
DELETE FROM BORROW WHERE BMEMBER_ID = (SELECT MEMBER_ID FROM MEMBER WHERE MNAME = 'Emily Miller');
DELETE FROM MEMBER WHERE MNAME = 'Emily Miller';


SELECT * FROM MEMBER
WHERE MNAME='EMILY MILLER'
```

Data Output Messages Notifications

member_id [PK] integer	mname character varying (20)	mcontact_info character varying (30)	join_date date

Total rows: 0 of 0 Query complete 00:00:00.059 Ln 5, Col 1

4.1.10 How do you add the following material to the database? Title: New book, Date: 2020-08-01 Catalog: E-Books ,Genre: Mystery & Thriller, Author: Lucas Pipi

Query:

INSERT INTO MATERIAL (MATERIAL_ID,TITLE, PUBLICATION_DATE, CATALOG_ID, GENRE_ID)

VALUES (32,'New book', '2020-08-01', (SELECT CATALOG_ID FROM CATALOG WHERE CNAME = 'E-Books'), (SELECT GENRE_ID FROM GENRE WHERE GNAME = 'Mystery & Thriller'))

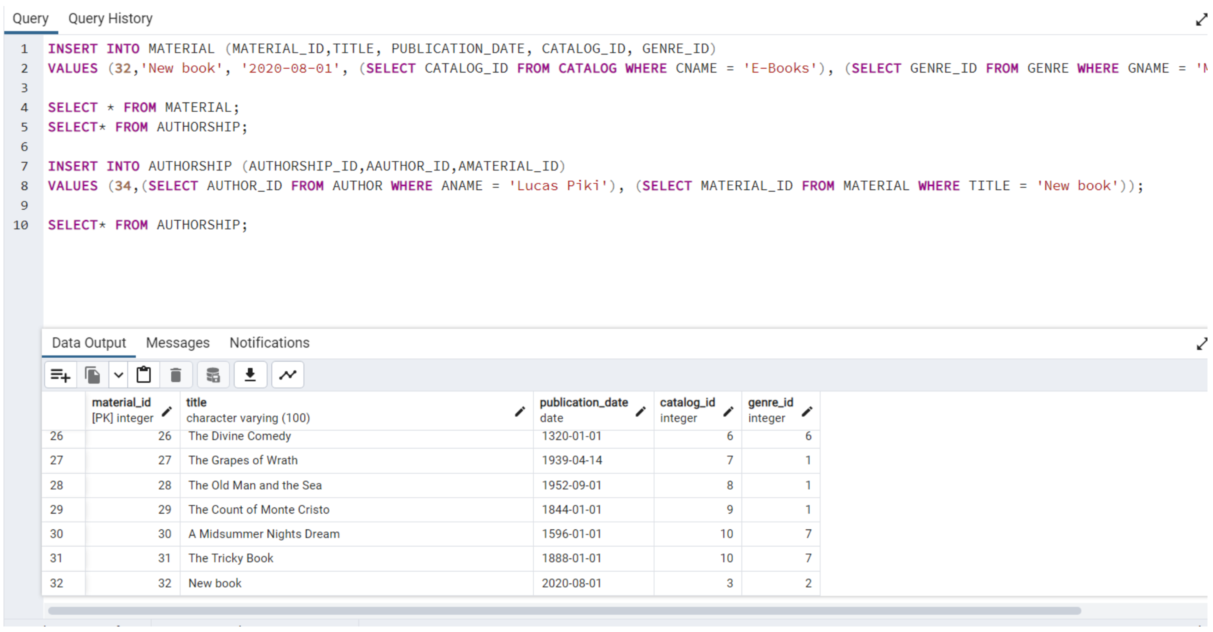
INSERT INTO AUTHORSHIP (AUTHORSHIP_ID,AAUTHOR_ID,AMATERIAL_ID)

VALUES (34,(SELECT AUTHOR_ID FROM AUTHOR WHERE ANAME = 'Lucas Piki'), (SELECT MATERIAL_ID FROM MATERIAL WHERE TITLE = 'New book'));

Explanation:

The INSERT statement adds the new material to the MATERIAL table, and the subqueries retrieve the corresponding CATALOG_ID and GENRE_ID values based on the provided catalog and genre names. The second INSERT statement adds a new record to the AUTHORSHIP table, associating the material with the author Lucas Pipi using their respective IDs retrieved via subqueries

Output:

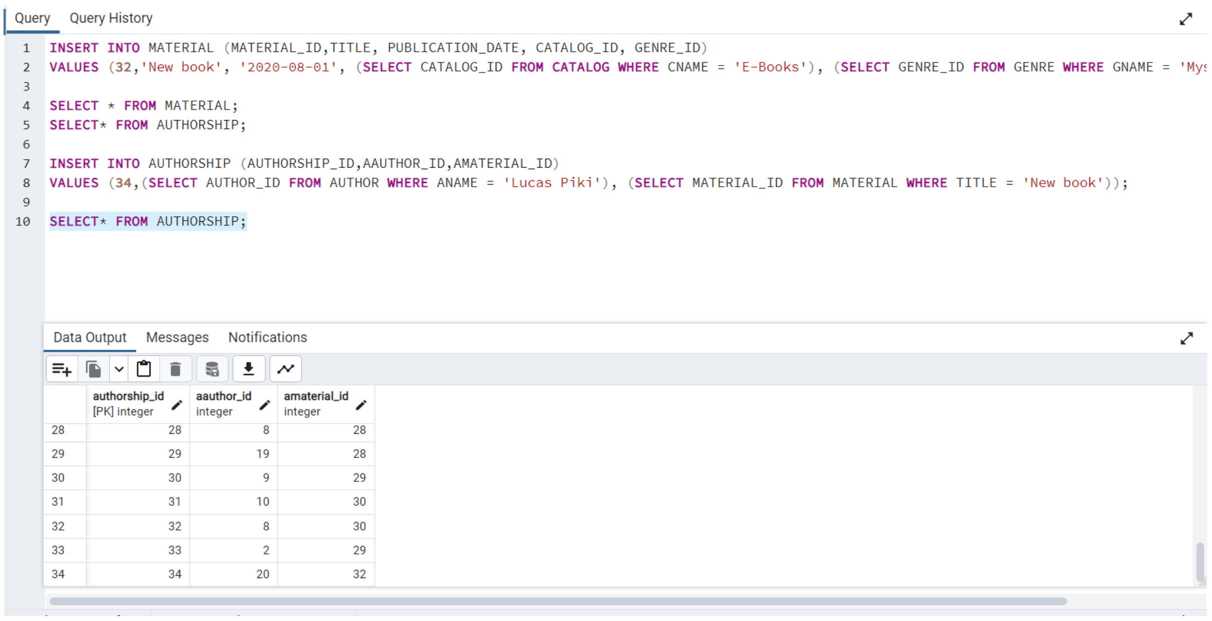


Query   Query History

```
INSERT INTO MATERIAL (MATERIAL_ID,TITLE, PUBLICATION_DATE, CATALOG_ID, GENRE_ID)
VALUES (32,'New book', '2020-08-01', (SELECT CATALOG_ID FROM CATALOG WHERE CNAME = 'E-Books'), (SELECT GENRE_ID FROM GENRE WHERE GNAME = 'N

SELECT * FROM MATERIAL;
SELECT* FROM AUTHORSHIP;

INSERT INTO AUTHORSHIP (AUTHORSHIP_ID,AAUTHOR_ID,AMATERIAL_ID)
VALUES (34,(SELECT AUTHOR_ID FROM AUTHOR WHERE ANAME = 'Lucas Piki'), (SELECT MATERIAL_ID FROM MATERIAL WHERE TITLE = 'New book'));

SELECT* FROM AUTHORSHIP;
```

Data Output   Messages   Notifications

	material_id [PK] integer	title character varying (100)	publication_date date	catalog_id integer	genre_id integer
26	26	The Divine Comedy	1320-01-01	6	6
27	27	The Grapes of Wrath	1939-04-14	7	1
28	28	The Old Man and the Sea	1952-09-01	8	1
29	29	The Count of Monte Cristo	1844-01-01	9	1
30	30	A Midsummer Nights Dream	1596-01-01	10	7
31	31	The Tricky Book	1888-01-01	10	7
32	32	New book	2020-08-01	3	2



Query   Query History

```
INSERT INTO MATERIAL (MATERIAL_ID,TITLE, PUBLICATION_DATE, CATALOG_ID, GENRE_ID)
VALUES (32,'New book', '2020-08-01', (SELECT CATALOG_ID FROM CATALOG WHERE CNAME = 'E-Books'), (SELECT GENRE_ID FROM GENRE WHERE GNAME = 'Mys

SELECT * FROM MATERIAL;
SELECT* FROM AUTHORSHIP;

INSERT INTO AUTHORSHIP (AUTHORSHIP_ID,AAUTHOR_ID,AMATERIAL_ID)
VALUES (34,(SELECT AUTHOR_ID FROM AUTHOR WHERE ANAME = 'Lucas Piki'), (SELECT MATERIAL_ID FROM MATERIAL WHERE TITLE = 'New book'));

SELECT* FROM AUTHORSHIP;
```

Data Output   Messages   Notifications

	authorship_id [PK] integer	aauthor_id integer	amaterial_id integer
28	28	8	28
29	29	19	28
30	30	9	29
31	31	10	30
32	32	8	30
33	33	2	29
34	34	20	32

4.2. Extending Database

4.2.1 Alert staff about overdue materials on a daily-basis?

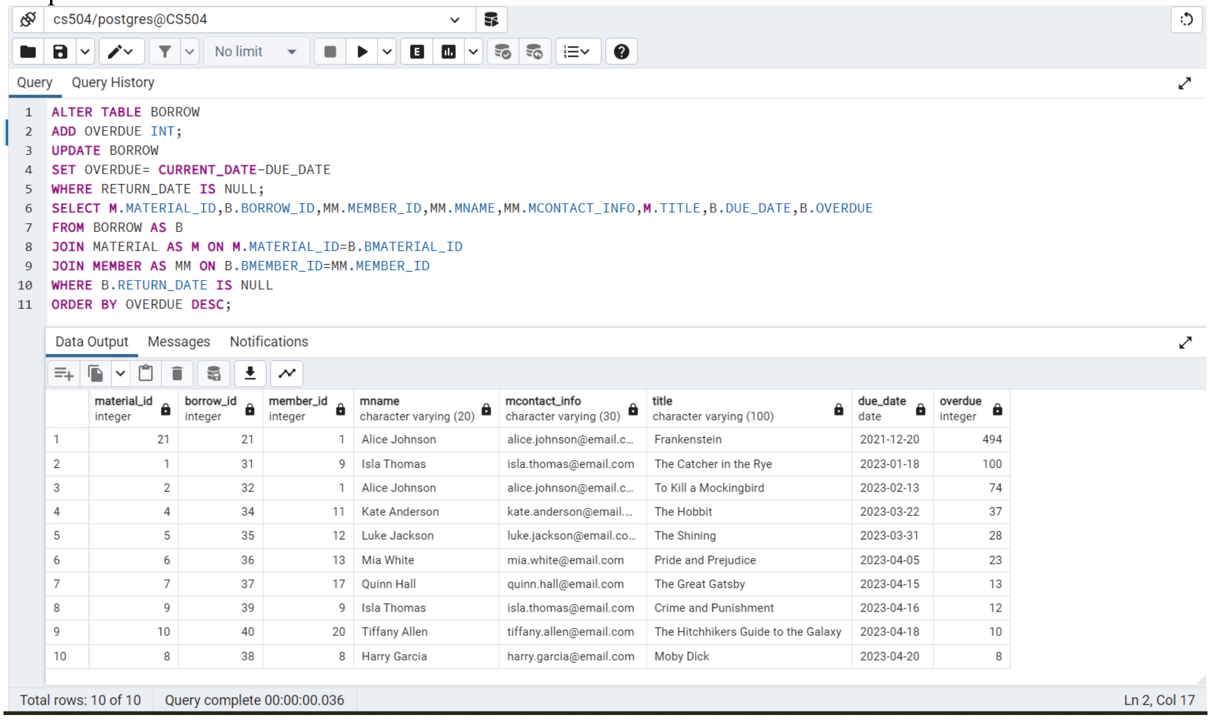
Query:

```
ALTER TABLE BORROW
ADD OVERDUE INT;

UPDATE BORROW
SET OVERDUE= CURRENT_DATE-DUE_DATE
WHERE RETURN_DATE IS NULL;
```

SELECT
M.MATERIAL_ID,B.BORROW_ID,MM.MEMBER_ID,MM.MNAME,MM.MCONTACT_INFO,M.TITLE,B.DUE_DATE,B.OVERDUE
FROM BORROW AS B
JOIN MATERIAL AS M ON M.MATERIAL_ID=B.BMATERIAL_ID
JOIN MEMBER AS MM ON B.BMEMBER_ID=MM.MEMBER_ID
WHERE B.RETURN_DATE IS NULL
ORDER BY OVERDUE DESC;

Explanation:
I have added the column in the borrow table and named it overdue and we perform the operation current_date - due_date , where return_date is NULL because all the people who didn't return will have NULL's in the return date.

Output:



```
ALTER TABLE BORROW
ADD OVERDUE INT;
UPDATE BORROW
SET OVERDUE= CURRENT_DATE-DUE_DATE
WHERE RETURN_DATE IS NULL;
SELECT M.MATERIAL_ID,B.BORROW_ID,MM.MEMBER_ID,MM.MNAME,MM.MCONTACT_INFO,M.TITLE,B.DUE_DATE,B.OVERDUE
FROM BORROW AS B
JOIN MATERIAL AS M ON M.MATERIAL_ID=B.BMATERIAL_ID
JOIN MEMBER AS MM ON B.BMEMBER_ID=MM.MEMBER_ID
WHERE B.RETURN_DATE IS NULL
ORDER BY OVERDUE DESC;
```

Data Output Messages Notifications

	material_id integer	borrow_id integer	member_id integer	mname character varying (20)	mcontact_info character varying (30)	title character varying (100)	due_date date	overdue integer
1	21	21	1	Alice Johnson	alice.johnson@email.c...	Frankenstein	2021-12-20	494
2	1	31	9	Isla Thomas	isla.thomas@email.com	The Catcher in the Rye	2023-01-18	100
3	2	32	1	Alice Johnson	alice.johnson@email.c...	To Kill a Mockingbird	2023-02-13	74
4	4	34	11	Kate Anderson	kate.anderson@email....	The Hobbit	2023-03-22	37
5	5	35	12	Luke Jackson	luke.jackson@email.co...	The Shining	2023-03-31	28
6	6	36	13	Mia White	mia.white@email.com	Pride and Prejudice	2023-04-05	23
7	7	37	17	Quinn Hall	quinn.hall@email.com	The Great Gatsby	2023-04-15	13
8	9	39	9	Isla Thomas	isla.thomas@email.com	Crime and Punishment	2023-04-16	12
9	10	40	20	Tiffany Allen	tiffany.allen@email.com	The Hitchhikers Guide to the Galaxy	2023-04-18	10
10	8	38	8	Harry Garcia	harry.garcia@email.com	Moby Dick	2023-04-20	8

Total rows: 10 of 10 Query complete 00:00:00.036 Ln 2, Col 17

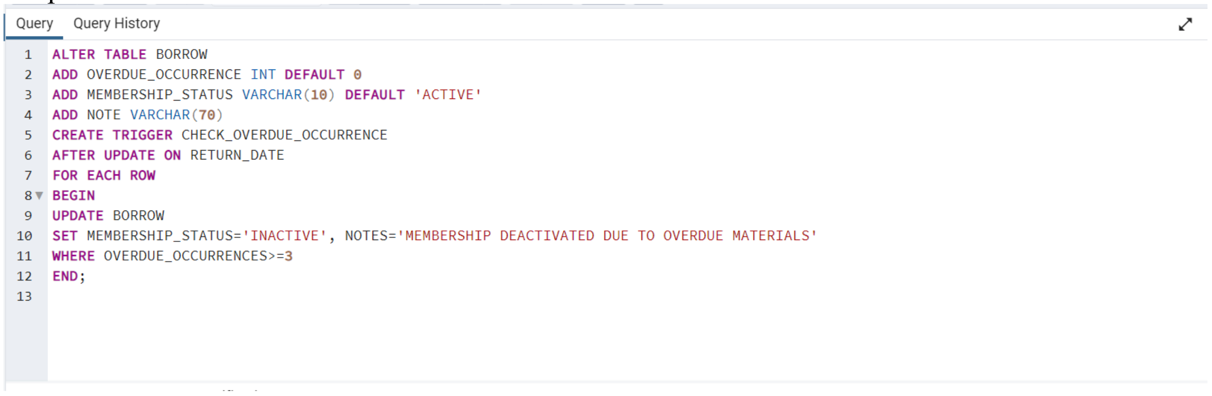
Here, overdue date have values where current_date is subtracted from the due_date, if the overdue represents the no of days passed from the due_date. Here in the above example, member_id =1, alice johnson have passed 494 days from the due_date and not returned book yet. Using the same logic we can know how many people did not submit the book.

4.2.2 Automatically deactivate the membership based on the member's overdue occurrence (>= three times). And reactivate the membership once the member pays the overdue fee

Query:
ALTER TABLE BORROW
ADD OVERDUE_OCCURRENCE INT DEFAULT 0
ADD MEMBERSHIP_STATUS VARCHAR(10) DEFAULT 'ACTIVE'
ADD NOTE VARCHAR(70)
CREATE TRIGGER CHECK_OVERDUE_OCCURRENCE
AFTER UPDATE ON RETURN_DATE
FOR EACH ROW
BEGIN

```
UPDATE BORROW
SET MEMBERSHIP_STATUS='INACTIVE', NOTES='MEMBERSHIP DEACTIVATED
DUE TO OVERDUE MATERIALS'
WHERE OVERDUE_OCCURRENCES>=3
END;
```

Output:



```
ALTER TABLE BORROW
ADD OVERDUE_OCCURRENCE INT DEFAULT 0
ADD MEMBERSHIP_STATUS VARCHAR(10) DEFAULT 'ACTIVE'
ADD NOTE VARCHAR(70)
CREATE TRIGGER CHECK_OVERDUE_OCCURRENCE
AFTER UPDATE ON RETURN_DATE
FOR EACH ROW
BEGIN
UPDATE BORROW
SET MEMBERSHIP_STATUS='INACTIVE', NOTES='MEMBERSHIP DEACTIVATED DUE TO OVERDUE MATERIALS'
WHERE OVERDUE_OCCURRENCES>=3
END;
```

Explanation:

Automatically deactivate the membership based on the member's overdue occurrence(>=3). And reactivate the membership once the member pays the overdue fee:

To implement this feature, we need to create trigger that runs after each material is checked in. This trigger would check the member's occurrence field and, if it is greater than or equal to e, would automatically deactivate the membership status. This trigger would also add a note to the member's record indicating that their membership has been deactivate due to overdue materials. Once the member pays the overdue fee, you would need to create a script or stored procedure that reactivates the member's membership status and resets their overdue occurrences field to zero.

Recommendations:

By using the above algorithm they can get the data of the people who haven't checked in their books more than 3 time, the staff can access the data and cancel their membership, when ever the payment is maid they can restart their subscription and restart the membership.

5. Conclusion

An overview of the project with implementation and offers ideas for future improvement.