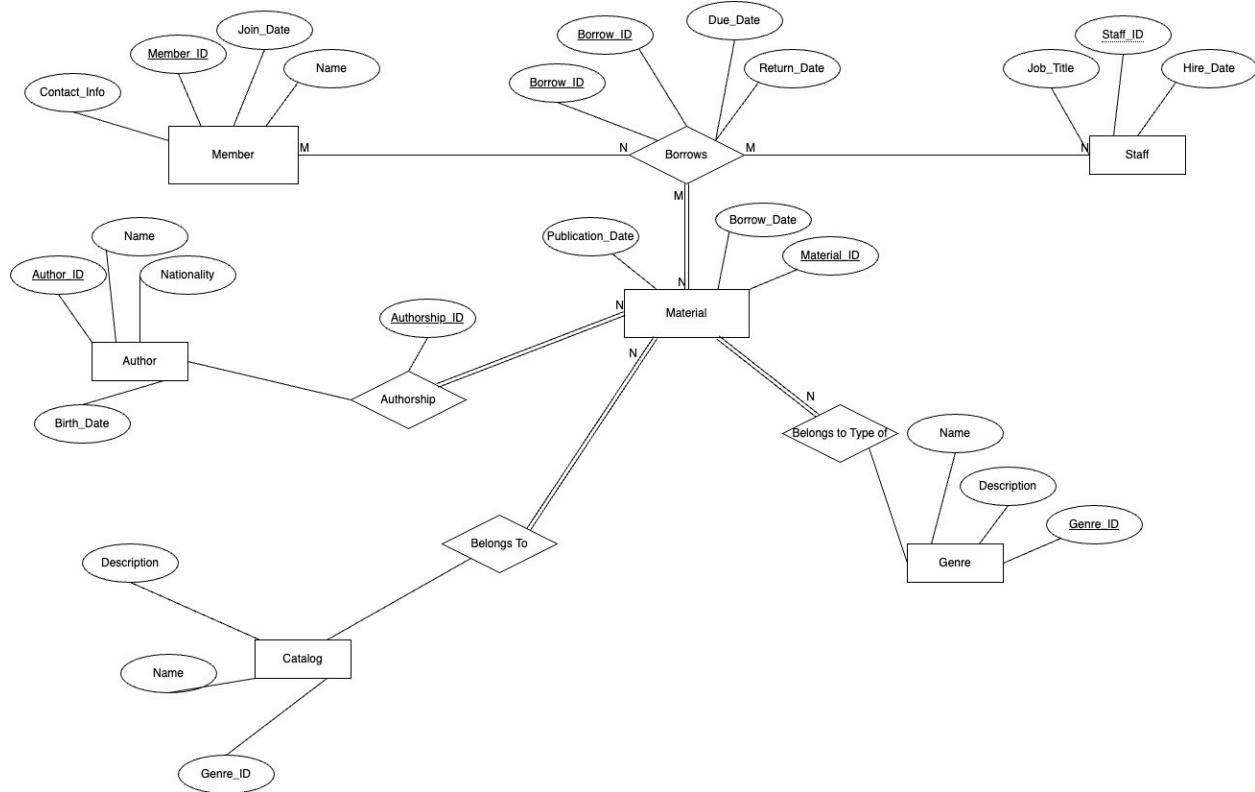


CS 504 – DB Project

ER Diagram



The purpose of this ER diagram is to visually represent the structure of the database that supports the management of library materials, members, staff, and borrowing activities.

Relationships:

- From the above diagram, we can see Authorship is a relationship which has complete or total participation with Material entity and partial participation with Author entity.
- The "Material" entity is related to "Author" through "Authorship" relationship with Author_ID and Material_ID attributes are foreign key constraints in Authorship table.
- The "Material" entity is related to the "Catalog" entity through the "Catalog_ID" attribute, establishing a link between individual materials and their catalog entries.
- The "Material" entity is related to the "Genre" entity through the "Genre_ID" attribute, associating materials with their respective genres.
- The "Borrow" entity is related to the "Material" entity through the "Material_ID" attribute, representing the borrowing activity of library materials by members.

Database Implementation

Table creation:

Author Entity:

```
Create table public."Author" (
    "Author_ID" integer NOT NULL ,
    "Name" character varying COLLATE pg_catalog."default",
    "Birth_Date" date,
    "Nationality" character varying COLLATE
pg_catalog."default",
    CONSTRAINT "Author_ID" PRIMARY KEY ("Author_ID")
) ;
```

Staff Entity

```
Create table public."Staff" (
    "Staff_ID" integer NOT NULL,
    "Name" character varying COLLATE pg_catalog."default",
    "Contact_Info" character varying COLLATE
pg_catalog."default",
    "Job_Title" character varying COLLATE pg_catalog."default",
    "Hire_Date" date,
    CONSTRAINT "Staff_ID" PRIMARY KEY ("Staff_ID")
) ;
```

Member Entity:

```
Create table public."Member" (
    "Member_ID" integer NOT NULL,
    "Name" character varying COLLATE pg_catalog."default",
    "Contact_Info" character varying COLLATE pg_catalog."default",
    "Join_Date" date,
    CONSTRAINT "Member_ID" PRIMARY KEY ("Member_ID")
) ;
```

Genre Entity:

```
Create table public."Genre" (
    "Genre_ID" integer NOT NULL,
    "Name" character varying COLLATE pg_catalog."default",
    "Description" character varying COLLATE
pg_catalog."default",
    CONSTRAINT "Genre_ID" PRIMARY KEY ("Genre_ID")
) ;
```

Catalog Entity:

```
Create table public."Catalog" (
    "Catalog_ID" integer NOT NULL,
    "Name" character varying COLLATE pg_catalog."default",
    "Location" character varying COLLATE pg_catalog."default",
    CONSTRAINT "Catalog_ID" PRIMARY KEY ("Catalog_ID")
) ;
```

Material Entity:

```
CREATE TABLE public."Material" (
    Material_ID SERIAL PRIMARY KEY,
    Title VARCHAR(255) NOT NULL,
    Publication_Date DATE,
    Catalog_ID INT REFERENCES public."Catalog"("Catalog_ID"),
    Genre_ID INT REFERENCES public."Genre"("Genre_ID")
) ;
```

Borrow Entity:

```
CREATE TABLE public."Borrow" (
    Borrow_ID SERIAL PRIMARY KEY,
    Material_ID INT REFERENCES public."Material"("material_id"),
    Member_ID INT REFERENCES public."Member"("Member_ID"),
    Staff_ID INT REFERENCES public."Staff"("Staff_ID"),
    Borrow_Date DATE,
    Due_Date DATE,
    Return_Date DATE
) ;
```

Insertion:

I have used ‘Import/Export Data’ feature in PgAdmin to insert values to all the tables from the required CSV files.

Queries:

1.Which materials are currently available in the library? If a material is borrowed and not returned, it's not considered as available.

```
Select
a."material_id" ,
a."title"
From public."Material" as a
where a."material_id" not in (
Select b."material_id"
from public."Borrow" as b
where b."return_date" is null or b."return_date">>CURRENT_DATE);
```

The screenshot shows the pgAdmin 4 interface with the following details:

- Toolbar:** Dashboard, Properties, SQL, Dependents, Processes, Create.sql, Library/postgr..., Untitled*, Library/postgres@PostgreSQL 16*, Library/postgr... (with a refresh icon).
- Query Editor:** The tab "Library/postgres@PostgreSQL 16" is selected. The query is:

```
1 Select
2 a."material_id" ,
3 a."title"
4 From public."Material" as a
5 where a."material_id" not in (
6 Select b."material_id"
7 from public."Borrow" as b
8 where b."return_date" is null or b."return_date">>CURRENT_DATE)
```

- Data Output:** The results are displayed in a table titled "Data Output". The columns are "material_id" [PK] integer and "title" character varying (255). The data consists of 20 rows:

material_id	title
1	3 The Da Vinci Code
2	11 1984
3	12 Animal Farm
4	13 The Haunting of Hill House
5	14 Brave New World
6	15 The Chronicles of Narnia: The Lion the Witch and the Wardro...
7	16 The Adventures of Huckleberry Finn
8	17 The Catch-22
9	18 The Picture of Dorian Gray
10	19 The Call of Cthulhu
11	22 A Tale of Two Cities
12	23 The Iliad
13	24 The Odyssey
14	25 The Brothers Karamazov
15	26 The Divine Comedy
16	27 The Grapes of Wrath
17	28 The Old Man and the Sea
18	29 The Count of Monte Cristo
19	30 A Midsummer Night's Dream
20	31 The Tricky Book

- Message Bar:** Total rows: 20 of 20 | Query complete 00:00:00.048 | Ln 1, Col 1

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

```
Select
m."Material_ID",
m."Title",
b."Borrow_Date",
b."Due_Date"
from public."Material" as m
inner join public."Borrow" as b on
b."Material_ID"=m."Material_ID"
WHERE b."Due_Date" < '2023-04-01' AND b."Return_Date" IS NULL
order by m."Material_ID";
```

The screenshot shows the pgAdmin 4 interface with the following details:

- Top Bar:** Dashboard, Properties, SQL, Dependents, Processes, Create.sql, Library/postgr..., Untitled*, Library/postgres@PostgreSQL 16*
- Toolbar:** Includes icons for New, Open, Save, Run, Stop, Refresh, and Help.
- Query History:** Shows the query being run.
- Messages Tab:** Shows "Total rows: 6 of 6" and "Query complete 00:00:00.043".
- Data Output Tab:** Displays the results of the query as a table:

	material_id	title	borrow_date	due_date
1	20	Harry Potter and the Philosopher's Stone	2021-10-21	2021-11-11
2	21	Frankenstein	2021-11-29	2021-12-20
3	1	The Catcher in the Rye	2022-12-28	2023-01-18
4	2	To Kill a Mockingbird	2023-01-23	2023-02-13
5	4	The Hobbit	2023-03-01	2023-03-22
6	5	The Shining	2023-03-10	2023-03-31

Q3. 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.

```

SELECT
m."title",
COUNT(*) AS borrow_count
FROM public."Material" m
INNER JOIN public."Borrow" b ON m."material_id" =
b."material_id"
GROUP BY m."material_id"
ORDER BY borrow_count DESC
LIMIT 10;

```

The screenshot shows the pgAdmin 4 interface with the following details:

- Top Bar:** Dashboard, Properties, SQL, Dependents, Processes, Create.sql, Library/postgr..., Untitled*, Library/postgres@PostgreSQL 16*
- Toolbar:** Includes icons for New, Open, Save, Run, Stop, Refresh, and Help.
- Query Editor:** Contains the SQL code for the query.
- Result Grid:** Displays the results of the query, showing 10 rows of data.
- Messages:** Shows a success message: "Successfully run. Total query runtime: 86 msec. 10 rows affected."
- Status Bar:** Total rows: 10 of 10, Query complete 00:00:00.086, Ln 34, Col 1

	title	borrow_count
1	The Hobbit	3
2	To Kill a Mockingbird	3
3	The Shining	3
4	Pride and Prejudice	3
5	The Catcher in the Rye	3
6	The Da Vinci Code	3
7	The Great Gatsby	2
8	Crime and Punishment	2
9	The Hitchhiker's Guide to the Galaxy	2
10	Moby Dick	2

Q4. How many materials has the author Lucas Piki written?

```
select
COUNT(*) as number_of_books
FROM public."Authorship" a
WHERE "author_id" IN (
    SELECT "Author_ID" FROM public."Author" b
    WHERE b."Name"='Lucas Piki');
```

The screenshot shows the pgAdmin 4 interface with a query editor and a data output viewer.

Query Editor:

```
26 SELECT
27 m."title",
28 COUNT(*) AS borrow_count
29 FROM public."Material" m
30 INNER JOIN public."Borrow" b ON m."material_id" = b."material_id"
31 GROUP BY m."material_id"
32 ORDER BY borrow_count DESC
33 LIMIT 10;
34
35 /*How many books has the author Lucas Piki written*/
36 select
37 COUNT(*) as number_of_books
38 FROM public."Authorship" a
39 WHERE "author_id" IN (
40     SELECT "Author_ID" FROM public."Author" b
41     WHERE b."Name"='Lucas Piki');
42
43
44
45 Select * from public."Authorship"
```

Data Output:

	number_of_books
1	1

Total rows: 1 of 1 | Query complete 00:00:00.058 | Ln 42, Col 1

Q5. How many materials were written by two or more authors?

```

SELECT
COUNT(*) as material_count
FROM (
    SELECT COUNT(DISTINCT a."Author_ID")
    FROM public."Author" a
    JOIN public."Authorship" b ON a."Author_ID" = b."author_id"
    GROUP BY b."material_id"
    HAVING COUNT(DISTINCT b."author_id") > 1
) AS subquery;

```

The screenshot shows the pgAdmin 4 interface with the following details:

- Toolbar:** Includes Dashboard, Properties, SQL, Dependents, Processes, Create.sql, Library/postgr..., Untitled*, Library/postgres@PostgreSQL 16*, Library/postgr... (with a refresh icon).
- Query Editor:** Shows the query code with line numbers 40 to 60. Lines 40-47 are part of a larger script and are not highlighted. Lines 48-56 are highlighted in blue, representing the subquery being explained. Line 57 is also highlighted in blue.
- Data Output:** A table showing the result of the query:

	material_count
1	4
- Status Bar:** Total rows: 1 of 1 | Query complete 00:00:00.052 | Ln 48, Col 1

Q6. What are the most popular genres in the library ranked by the total number of borrowed times of each genre?

```

SELECT
g."Genre_ID",
g."Name" AS Genre_Name,
COUNT(b."borrow_id") AS Borrowed_times
FROM public."Genre" as g
LEFT JOIN public."Material" m ON g."Genre_ID" = m."genre_id"
LEFT JOIN public."Borrow" b ON m."material_id" =
b."material_id"
GROUP BY g."Genre_ID",g."Name"
ORDER BY Borrowed_times DESC;

```

The screenshot shows the pgAdmin interface with the following details:

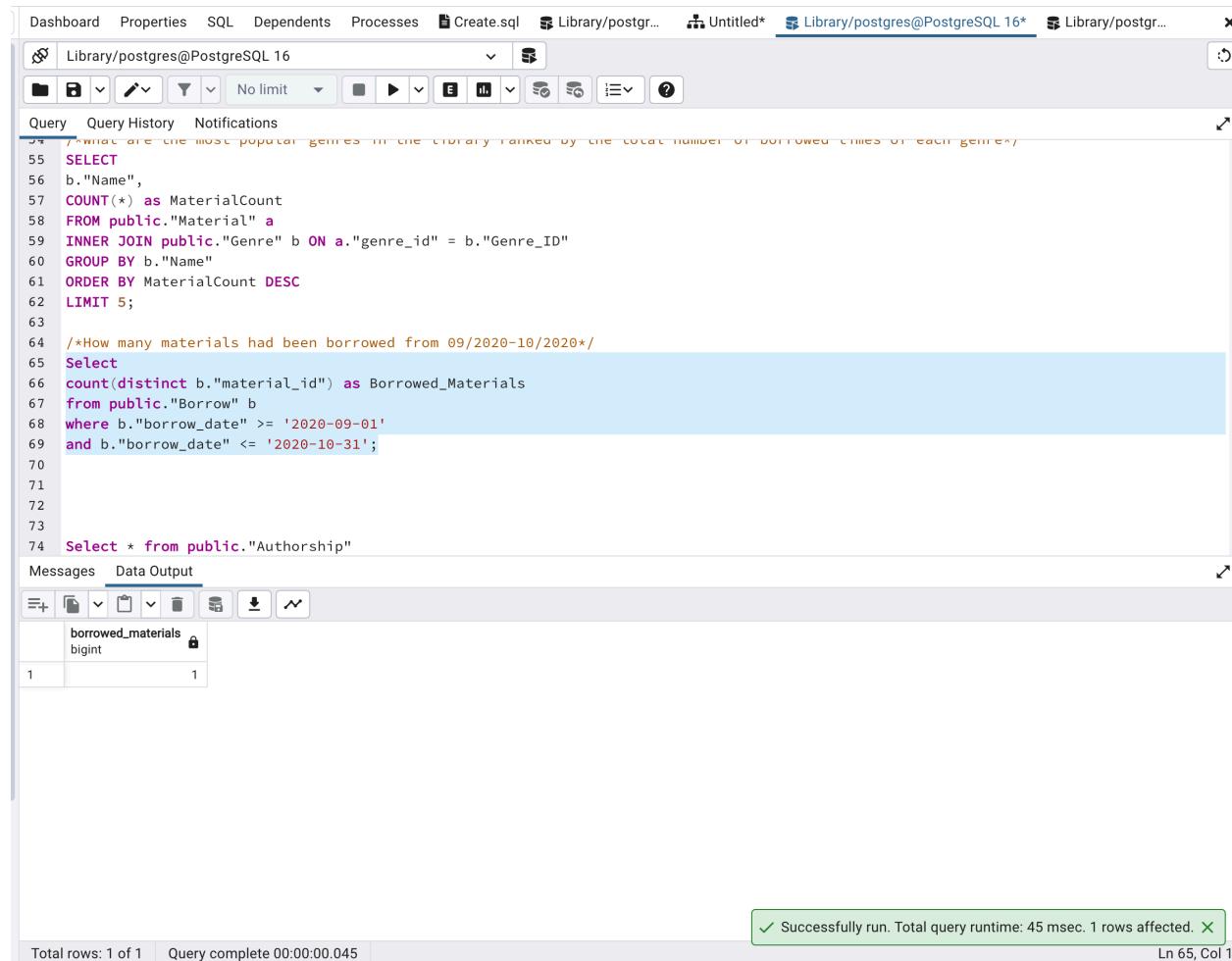
- Toolbar:** Includes Dashboard, Properties, SQL, Dependents, Processes, and a tab labeled "Select_Queries.sql".
- Message Bar:** Displays a warning: "You are currently running version 7.7 of pgAdmin 4, however the current version is 7.8. Please click [here](#) for more information."
- Query Editor:** Shows the SQL query from the code block above.
- Data Output:** A table showing the results of the query. The table has three columns: "Genre_ID", "genre_name", and "borrowed_times". The data is as follows:

Genre_ID	genre_name	borrowed_times
1	General Fiction	22
3	Science Fiction & Fantasy	6
4	Horror & Suspense	5
2	Mystery & Thriller	3
6	Classics	3
7	Historical Fiction	1
8	Epic Poetry & Mythology	0
5	Dystopian & Apocalyptic	0

- Status Bar:** Shows "Total rows: 8 of 8" and "Query complete 00:00:00.060".
- Bottom Right:** A note "Ln 68, Col 30" is visible.

Q7. How many materials had been borrowed from 09/2020-10/2020?

```
Select
count(distinct b."Material_ID") as Borrowed_Materials
from public."Borrow" b
where b."Borrow_Date" >= '2020-09-01'
and b."Borrow_Date" <= '2020-10-31';
```



The screenshot shows the pgAdmin 4 interface with the following details:

- Toolbar:** Dashboard, Properties, SQL, Dependents, Processes, Create.sql, Library/postgr..., Untitled*, Library/postgres@PostgreSQL 16*
- Query Editor:** Shows the SQL query for Q7.
- Messages Tab:** Displays "Successfully run. Total query runtime: 45 msec. 1 rows affected." and "Ln 65, Col 1".
- Data Output Tab:** Shows a table with one row and one column named "borrowed_materials" containing the value 1.

Q8. How do you update the “Harry Potter and the Philosopher's Stone” when it is returned on 04/01/2023?

```
UPDATE public."Borrow"
SET return_date = '2023-04-01'
WHERE Material_ID=(SELECT material_id from public."Material" m
WHERE m.title='Harry Potter and the Philosopher''s Stone')
AND Return_Date IS NULL;
```

```
SELECT * from public."Borrow";
```

The screenshot shows the pgAdmin 4 interface. At the top, there are tabs for Dashboard, Properties, SQL, Dependents, Processes, Create.sql, Library/postgr..., Untitled*, and Library/postgres@PostgreSQL 16*. Below the tabs is a toolbar with various icons for database management. The main area contains a code editor with the following SQL script:

```
64 /*How many materials had been borrowed from 09/2020-10/2020*/
65 Select
66 count(distinct b."material_id") as Borrowed_Materials
67 from public."Borrow" b
68 where b."borrow_date" >= '2020-09-01'
69 and b."borrow_date" <= '2020-10-31';
70
71
72 /**
73
74 UPDATE public."Borrow"
75 SET return_date = '2023-04-01'
76 WHERE Material_ID=(SELECT material_id from public."Material" m WHERE m.title='Harry Potter and the Philosopher''s Stone')
77 AND Return_Date IS NULL;
78
79 SELECT * from public."Borrow";
80
81
82
83
```

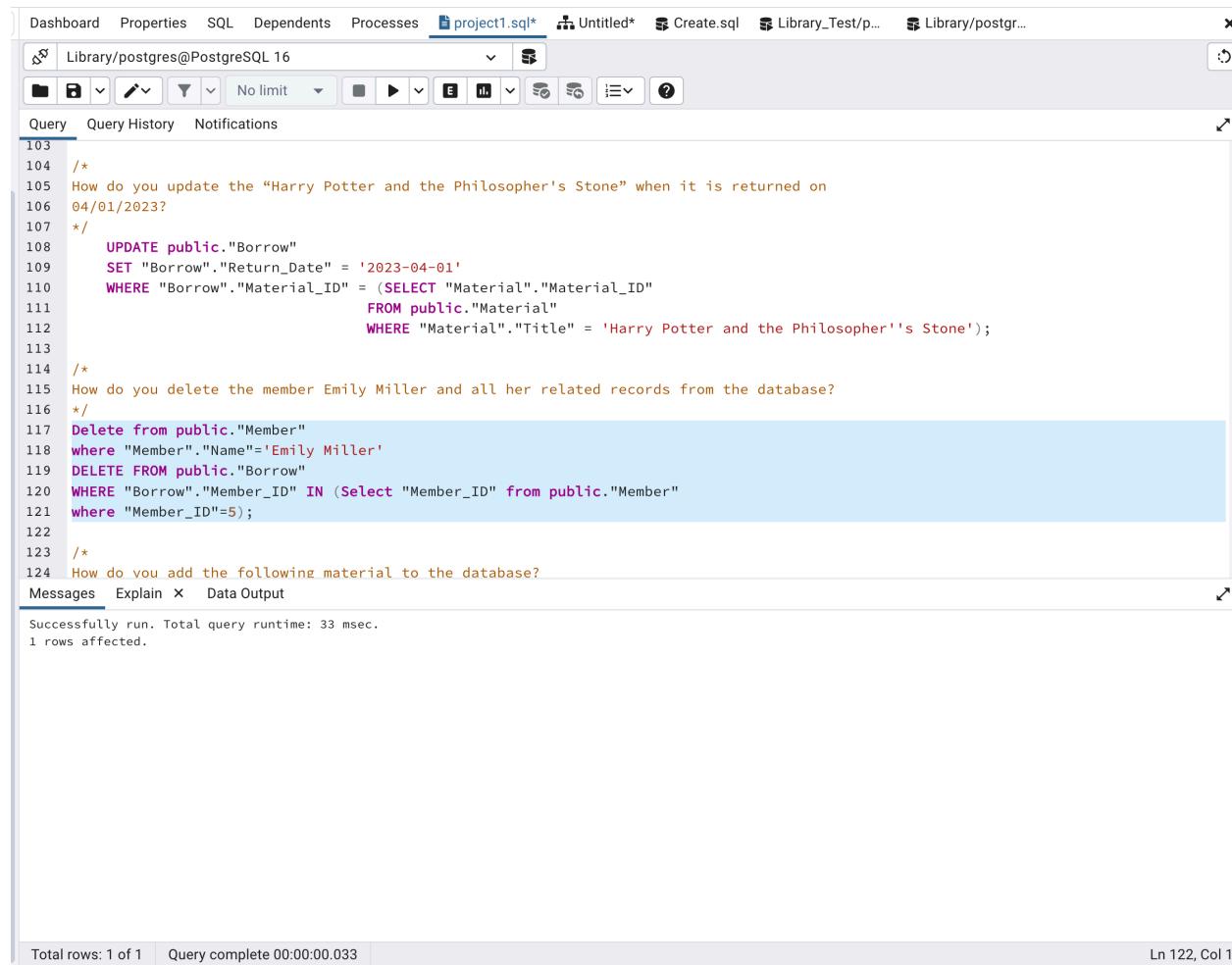
Below the code editor is a "Messages" tab and a "Data Output" tab. The "Data Output" tab displays a table with the following data:

	borrow_id [PK] integer	material_id integer	member_id integer	staff_id integer	borrow_date date	due_date date	return_date date
30	31	1	9	4	2022-12-28	2023-01-18	[null]
31	32	2	1	4	2023-01-23	2023-02-13	[null]
32	33	3	10	4	2023-02-02	2023-02-23	2023-02-17
33	34	4	11	4	2023-03-01	2023-03-22	[null]
34	35	5	12	4	2023-03-10	2023-03-31	[null]
35	36	6	13	4	2023-03-15	2023-04-05	[null]
36	37	7	17	4	2023-03-25	2023-04-15	[null]
37	38	8	8	4	2023-03-30	2023-04-20	[null]
38	39	9	9	4	2023-03-26	2023-04-16	[null]
39	40	10	20	4	2023-03-28	2023-04-18	[null]
40	20	20	7	2	2021-10-21	2021-11-11	2023-04-01

Total rows: 40 of 40 Query complete 00:00:00.054 Ln 79, Col 1

Q9. How do you delete the member Emily Miller and all her related records from the database?

```
Delete from public."Member"
where "Member"."Name"='Emily Miller'
DELETE FROM public."Borrow"
WHERE "Borrow"."Member_ID" IN (Select "Member_ID" from
public."Member"
where "Member_ID"=5);
```



The screenshot shows the pgAdmin 4 interface with the following details:

- Toolbar:** Includes Dashboard, Properties, SQL, Dependents, Processes, project1.sql*, Untitled*, Create.sql, Library_Test/p..., Library/postgr... buttons.
- Query Bar:** Shows Library/postgres@PostgreSQL 16 and various icons for file operations.
- Panel Tabs:** Query, Query History, Notifications.
- Code Area:** Displays the SQL code for question Q9. The code is highlighted in blue, indicating it is selected or being edited.
- Message Area:** Shows "Successfully run. Total query runtime: 33 msec." and "1 rows affected."
- Status Bar:** Shows "Total rows: 1 of 1" and "Query complete 00:00:00.033".
- Bottom Right:** Shows "Ln 122, Col 1".

Q10. How do you add the following material to the database? Title: New book Date: 2020-08-01 Catalog: E-Books13 Genre: Mystery & Thriller Author: Lucas Luke

First I have loaded Author name into the author table

```
INSERT INTO public."Author"
    "Author_ID", "Name")
VALUES (21, 'Lucas Luke');
```

```
Select * from public."Author"
```

```

99
100 /*Title: New book Date: 2020-08-01 Catalog: E-Books  Genre: Mystery & Thriller Author: Lucas Luke*/
101 INSERT INTO public."Author"
102     "Author_ID", "Name")
103     VALUES (21, 'Lucas Luke');
104
105 Select * from public."Author"
106
107
108 INSERT into public."Material" ("material_id", "title", "publication_date", "catalog_id", "genre_id")
109 VALUES (32, 'New book', '2020-08-01',
110         (SELECT "Catalog_ID" FROM public."Catalog" WHERE "Name" = 'E-Books'),

```

Author_ID [PK] integer	Name character varying	Birth_Date date	Nationality character varying
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
21	Lucas Luke	[null]	[null]

As we can see from the above screenshot “Lucas Luke” has been updated as 21st record in Author Table.

```
INSERT into public."Material" ("material_id", "title",
"publication_date", "catalog_id", "genre_id")
VALUES (32, 'New book', '2020-08-01',
        (SELECT "Catalog_ID" FROM public."Catalog" WHERE
"Name" = 'E-Books'),
        (SELECT "Genre_ID" FROM public."Genre" WHERE "Name" =
'Mystery & Thriller'));
```

```
SELECT * FROM public."Material"
```

The screenshot shows the pgAdmin 4 interface. The top bar has tabs for SQL, Dependents, Processes, Create.sql, Library/postgr..., Untitled*, New_Project.sql*, Library/postgr..., and Library/postgr... . Below the tabs is a toolbar with various icons. The main area has two tabs: Query and Query History. The Query tab contains the following SQL code:

```
105 Select * from public."Author"
106
107
108 INSERT into public."Material" ("material_id", "title", "publication_date", "catalog_id", "genre_id")
109 VALUES (32, 'New book', '2020-08-01',
110         (SELECT "Catalog_ID" FROM public."Catalog" WHERE "Name" = 'E-Books'),
111         (SELECT "Genre_ID" FROM public."Genre" WHERE "Name" = 'Mystery & Thriller'));
112
113 SELECT * FROM public."Material"
114
115 INSERT INTO public."Authorship" (Authorship_ID, Material_ID, Author_ID)
116 VALUES (34,
```

Below the code, there are tabs for Messages and Data Output. The Data Output tab shows a table with the following data:

	material_id [PK] integer	title character varying (255)	publication_date date	catalog_id integer	genre_id integer
16	16	The Adventures of Huckleberry Finn	1884-12-10	6	1
17	17	The Catch-22	1961-10-11	7	1
18	18	The Picture of Dorian Gray	1890-07-01	8	1
19	19	The Call of Cthulhu	1928-02-01	9	4
20	20	Harry Potter and the Philosopher's Stone	1997-06-26	10	3
21	21	Frankenstein	1818-01-01	6	4
22	22	A Tale of Two Cities	1859-04-30	7	1
23	23	The Iliad	1750-01-01	8	6
24	24	The Odyssey	1725-01-01	9	6
25	25	The Brothers Karamazov	1880-01-01	10	1
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 Night's 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

Total rows: 32 of 32 Query complete 00:00:00.080 Ln 112, Col 1

From the above screenshot we can see that we have inserted 32nd row with title as 'New Book' with Publication Date="2020-08-01"

```

INSERT INTO public."Authorship" (Authorship_ID, Material_ID,
Author_ID)
VALUES (35,
        (SELECT "material_id" FROM public."Material" WHERE
"title" = 'New book'),
        (SELECT "Author_ID" FROM public."Author" WHERE "Name"
= 'Lucas Luke'));
SELECT * FROM public."Authorship"

```

The screenshot shows a pgAdmin 4 interface. In the top navigation bar, there are tabs for SQL, Dependents, Processes, Create.sql, Library/postgr..., Untitled*, New_Project.sql*, Library/postgr..., and Library/postgr... . Below the tabs is a toolbar with various icons for file operations. The main area has two tabs: 'Query' and 'Data Output'. The 'Query' tab contains the following SQL code:

```

109 VALUES (32, 'New book', '2020-08-01',
110   (SELECT "Catalog_ID" FROM public."Catalog" WHERE "Name" = 'E-Books'),
111   (SELECT "Genre_ID" FROM public."Genre" WHERE "Name" = 'Mystery & Thriller'));
112
113 SELECT * FROM public."Material"
114
115 INSERT INTO public."Authorship" (Authorship_ID, Material_ID, Author_ID)
116 VALUES (35,
117   (SELECT "material_id" FROM public."Material" WHERE "title" = 'New book'),
118   (SELECT "Author_ID" FROM public."Author" WHERE "Name" = 'Lucas Luke'));
119 SELECT * FROM public."Authorship"
120

```

The 'Data Output' tab displays the contents of the 'Authorship' table:

	authorship_id [PK] integer	author_id integer	material_id integer
19	19	19	19
20	20	20	20
21	21	1	21
22	22	2	22
23	23	3	22
24	24	3	23
25	25	4	24
26	26	5	25
27	27	6	26
28	28	7	27
29	29	8	28
30	30	19	28
31	31	9	29
32	32	10	30
33	33	8	30
34	34	2	29
35	35	21	32

Total rows: 35 of 35 Query complete 00:00:00.052 Ln 119, Col 1

As we can see last row has been updated with all the three id's

Q11. Calculate the average number of materials borrowed per member.

```
Select
a."Member_ID",
a."Name" AS Member_Name,
count(b."Member_ID") as Borrow_count
from public."Member" a
left outer join public."Borrow" b on b."Member_ID"=a."Member_ID"
group by a."Member_ID",
a."Name"
order by Borrow_count;
```

The screenshot shows the pgAdmin 4 interface with the following details:

- Toolbar:** Dashboard, Properties, SQL, Dependents, Processes, project1.sql*, Untitled*, Create.sql, Library_Test/p..., Library/postgr...
- Query Editor:** Shows the SQL code for the query.
- Results Table:**

	Member_ID [PK] integer	member_name character varying (100)	borrow_count bigint
1	19	Sam Walker	0
2	14	Noah Harris	0
3	15	Olivia Clark	0
4	18	Rachel Young	0
5	16	Peter Lewis	0
6	17	Quinn Hall	1
7	20	Tiffany Allen	1
8	11	Kate Anderson	2
9	10	Jack Martinez	2
10	13	Mia White	2
11	12	Luke Jackson	2
12	2	Bob Smith	3
13	4	David Williams	3
14	7	Grace Wilson	3
15	3	Carol Brown	3
16	6	Frank Davis	3
17	1	Alice Johnson	4
18	9	Henry Garcia	4
- Message Bar:** Total rows: 19 of 19, Query complete 00:00:00.105, Ln 144, Col 23

Q12. List materials with their genres and the names of authors who created them.

```
Select
a."Title" ,
b."Name" as Genre_Name,
d."Name" as Author_Name
from public."Material" a
left outer join public."Genre" b on a."Genre_ID"=b."Genre_ID"
left outer join public."Authorship" c on
c."Material_ID"=a."Material_ID"
Left outer join public."Author" d on d."Author_ID"=c."Author_ID"
```

The screenshot shows the pgAdmin 4 interface with a query editor and a results grid.

Query Editor:

```

148 /*
149 Select
150 a."Title" ,
151 b."Name" as Genre_Name,
152 d."Name" as Author_Name
153 from public."Material" a
154 left outer join public."Genre" b on a."Genre_ID"=b."Genre_ID"
155 left outer join public."Authorship" c on
156 c."Material_ID"=a."Material_ID"
157 Left outer join public."Author" d on d."Author_ID"=c."Author_ID"
158 */
159 Find the staff members who processed borrowings along with the respective member names.
160

```

Results Grid:

	Title	genre_name	author_name
1	The Catcher in the Rye	General Fiction	Jane Austen
2	To Kill a Mockingbird	General Fiction	Ernest Hemingway
3	Th Da Vinci Code	Mystery & Thriller	George Orwell
4	The Hobbit	Science Fiction & Fantasy	Scott Fitzgerald
5	The Shining	Horror & Suspense	J.K. Rowling
6	Pride and Prejudice	General Fiction	Mark Twain
7	The Great Gatsby	General Fiction	Leo Tolstoy
8	Moby Dick	General Fiction	Virginia Woolf
9	Crime and Punishment	General Fiction	Gabriel Márquez
10	The Hitchhiker's Guide to the Galaxy	Science Fiction & Fantasy	Charles Dickens
11	1984	Dystopian & Apocalyptic	Harper Lee
12	Animal Farm	Dystopian & Apocalyptic	Oscar Wilde
13	The Haunting of Hill House	Horror & Suspense	William Shakespeare
14	Brave New World	Dystopian & Apocalyptic	Franz Kafka
15	The Chronicles of Narnia: The Lion the Witch and the Wardro...	Science Fiction & Fantasy	James Joyce
16	The Adventures of Huckleberry Finn	General Fiction	J.R.R. Tolkien
17	The Catch-22	General Fiction	Emily Brontë
18	The Picture of Dorian Gray	General Fiction	Toni Morrison

Total rows: 35 of 35 Query complete 00:00:00.038 Ln 157, Col 1

Future Work:

1. Alert staff about overdue materials on a daily-basis?

To alert staff about overdue materials , we have to create table called as alert_staff

```
CREATE TABLE Alert_staff (
    Alert_ID SERIAL PRIMARY KEY,
    Material_ID INTEGER,
    Member_ID INTEGER,
    Staff_ID INTEGER,
    Alert_Date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Created a stored procedure to generate alert and insert into alert_staff table, so that we will get to know when we run select query for alert staff table.

```
CREATE OR REPLACE FUNCTION generate_overdue_alert()
RETURNS VOID AS $$ 
BEGIN
    INSERT INTO Alert_staff (Material_ID, Member_ID, Staff_ID)
    SELECT b."material_id", b."member_id", b."staff_id"
    FROM public."Borrow" b
    WHERE b."return_date" IS NULL AND b."due_date" <
CURRENT_DATE;
END;
$$ LANGUAGE plpgsql;
```

Using pg_notify(it is asynchronous feature of pg admin) we have created a function called schedule daily alert

```
CREATE OR REPLACE FUNCTION schedule_daily_alerts()
RETURNS TRIGGER AS $$ 
BEGIN
    PERFORM pg_notify('alert_staff', 'generate_overdue_alert');
    RETURN NULL;
END;
$$ LANGUAGE plpgsql;
```

Created trigger which it inserts data into alert_staff table whenever there is overdue materials

```
CREATE TRIGGER daily_notification_alert
AFTER INSERT ON public."Alert_staff"
EXECUTE FUNCTION schedule_daily_notifications();
```

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.

We have added new column into member to get membership_status

```
ALTER TABLE public."Member"
ADD COLUMN membership_status BOOLEAN DEFAULT TRUE ;
```

We have created new table Payments which will store the payment information for each member.

```
CREATE TABLE public."payments" (
    payment_id SERIAL PRIMARY KEY,
    member_id INT REFERENCES public."Member" ("Member_ID"),
    payment_date DATE,
    payment_amount DECIMAL(10, 2)
);
```

The update_membership_function is a trigger function that will automatically run after an insert or update operation on the member and payments tables.

```
CREATE OR REPLACE FUNCTION update_membership_status() RETURNS
TRIGGER AS $$
BEGIN

    IF NEW.overdue_occurrences >= 3 THEN
        UPDATE members SET membership_status = FALSE WHERE id =
NEW.id;
    END IF;

    IF TG_TABLE_NAME = 'payments' THEN
```

```
        UPDATE members SET membership_status = TRUE WHERE id =  
NEW.member_id;  
        END IF;  
  
        RETURN NEW;  
END;  
$$ LANGUAGE plpgsql;
```

Created two triggers to check the status and payment

```
CREATE TRIGGER check_membership_status  
AFTER INSERT OR UPDATE ON public."Member"  
FOR EACH ROW EXECUTE FUNCTION update_membership_status();  
  
CREATE TRIGGER check_payment  
AFTER INSERT ON public."payments"  
FOR EACH ROW EXECUTE FUNCTION update_membership_status();
```

INSTRUCTIONS TO RUN THE CODE

1. Create Database in Pg Admin .
2. Create all the tables using create statements mentioned in the table queries section.
3. Now Insert values using ‘Import/Export Data’ wizard for the standalone tables which do not have any constraints like ‘Author’,‘Genre’,‘Catalog’,‘Member’,‘Staff’ and later insert into ‘Material’ ,‘Borrow’ and ‘Authorship’ table as they have foreign key constraints . Follow this sequence respectively.
4. Now Execute the queries as mentioned in Queries section.