

Library Management System

Entities Detail:

The database is designed for a **Library Management System**, encompassing essential entities for efficient library operations. **Authors** are recorded with their details like name, birthdate, nationality, and biography. **Borrowers** are identified by unique IDs and have personal information like name, address, contact details, membership type, and expiry date. **Genres** categorize books based on type, with additional attributes for language, rating, and description. **Publishers** are listed with their IDs, names, addresses, and contact information.

Books form the **core** of the system, each having a unique ID, title, publication year, and associated author, publisher, and genre IDs. **Fiction books** and **non-fiction books** are **specialized** types, each with unique identifiers and specific attributes like themes, summaries, subjects, and academic levels. **Transactions** track book borrowing, including details like transaction IDs, book IDs, borrower IDs, dates of issue, due dates, return dates, fine amounts, and branch IDs.

The system also manages **staff members** with IDs, names, positions, joining dates, salaries, contact details, and branch assignments. **Branches** represent the physical locations of the library, each with an ID, name, location, contact number, and manager ID.

Business rules:

Business rules dictate categorizing books into fiction and non-fiction genres, tracking borrower memberships and due dates to avoid fines, and efficiently managing staff responsibilities across different branches. Overall, the database ensures smooth library functions, from cataloging books to managing loans, fines, and staff duties.

Entities:

Here are the entities for LMS:

- ❖ Authors
- ❖ Borrowers
- ❖ Genres
- ❖ Publishers
- ❖ Books
- ❖ FictionBooks
- ❖ NonFictionBooks
- ❖ Transactions
- ❖ Staff
- ❖ Branches

Tables and Attributes:

Here are the tables and attributes for LMS:

✓ **Authors:**

- ❖ AuthorId (Primary Key)
- ❖ AuthorName
- ❖ BirthDate
- ❖ Nationality
- ❖ Biography

✓ **Borrowers:**

- ❖ BorrowerId (Primary Key)
- ❖ BorrowerName
- ❖ Address
- ❖ PhoneNo
- ❖ Email
- ❖ MembershipType
- ❖ MembershipExpiry

✓ **Genres:**

- ❖ GenreId (Primary Key)
- ❖ GenreType
- ❖ GenreLanguage
- ❖ Rating
- ❖ Description

✓ **Publishers:**

- ❖ PublisherId (Primary Key)
- ❖ PublisherName
- ❖ Address
- ❖ PhoneNo

✓ **Books:**

- ❖ BookId (Primary Key)
- ❖ Title
- ❖ PubYear
- ❖ AuthorId (Foreign Key referencing Authors)
- ❖ PublisherId (Foreign Key referencing Publishers)
- ❖ GenreId (Foreign Key referencing Genres)

✓ **Fiction Book:**

- ❖ FictionBookId (Primary Key)
- ❖ BookId (Foreign Key referencing Books)
- ❖ Theme
- ❖ Summary

✓ **Non-Fiction Book:**

- ❖ NonFictionBookId (Primary Key)
- ❖ BookId (Foreign Key referencing Books)
- ❖ Subject
- ❖ AcademicLevel

✓ **Transactions:**

- ❖ TransactionId (Primary Key)
- ❖ BookId (Foreign Key referencing Books)
- ❖ BorrowerId (Foreign Key referencing Borrowers)
- ❖ IssueDate
- ❖ DueDate
- ❖ ReturnDate
- ❖ FineAmount
- ❖ BranchId (Foreign Key referencing Branches)

✓ **Staff:**

- ❖ StaffId (Primary Key)
- ❖ Name
- ❖ Position
- ❖ JoiningDate
- ❖ Salary
- ❖ PhoneNo
- ❖ BranchId (Foreign Key referencing Branches)

✓ **Branches:**

- ❖ BranchId (Primary Key)
- ❖ BranchName
- ❖ Location
- ❖ BranchContactId
- ❖ BranchManagerId (Foreign Key referencing Staff)

Relationships:

Here are the relationships between the tables in the LMS database schema:

✚ Authors and Books:

One-to-Many relationship: An author can write multiple books, but a book is written by only one author.

✚ Publishers and Books:

One-to-Many relationship: A publisher can publish multiple books, but a book is published by only one publisher.

✚ Genres and Books:

One-to-Many relationship: A genre can be assigned to multiple books, but a book is categorized under only one genre.

✚ Fiction Books and Non-fiction Books:

Inheritance relationship: Both fiction books and non-fiction books are types of books, where fiction books have additional attributes like theme and summary, and non-fiction books have attributes like subject and academic level.

✚ Transactions and Borrowers:

One-to-Many relationship: A borrower can have multiple transactions (e.g., borrowing multiple books), but each transaction is associated with only one borrower.

✚ Branches and Staff:

One-to-Many relationship: A branch can have multiple staff members, but each staff member is assigned to only one branch.

✚ Branches and Transactions:

One-to-Many relationship: A branch can have multiple transactions (e.g., books borrowed from that branch), but each transaction is associated with only one branch.

DATABASE SYSTEMS FINAL PROJECT

LMS

QUERIES

Update Queries:

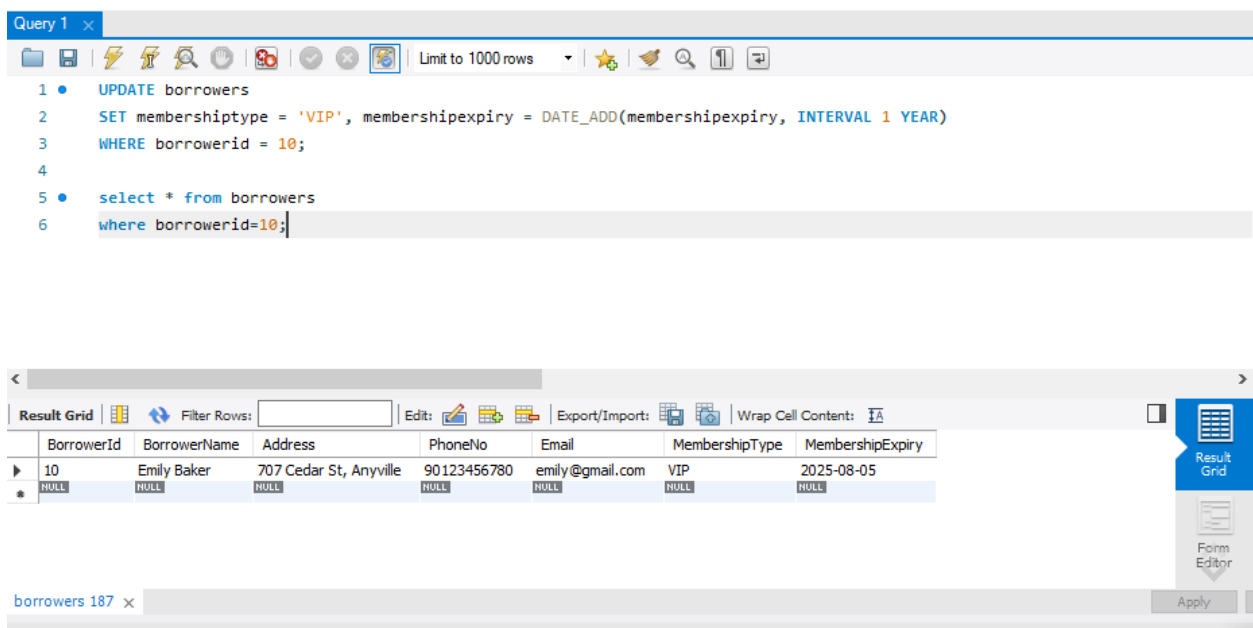
Query Number 1:

Query:

UPDATE borrowers

SET membertype = 'VIP', membershipexpiry = DATE_ADD(membershipexpiry, INTERVAL 1 YEAR)

WHERE borrowerid = 101;



The screenshot displays a database management interface. The top section shows a query editor with the following SQL code:

```
1 • UPDATE borrowers
2   SET membertype = 'VIP', membershipexpiry = DATE_ADD(membershipexpiry, INTERVAL 1 YEAR)
3   WHERE borrowerid = 10;
4
5 • select * from borrowers
6   where borrowerid=10;
```

Below the query editor, the 'Result Grid' is visible, showing the results of the query. The grid has the following columns: BorrowerId, BorrowerName, Address, PhoneNo, Email, MembershipType, and MembershipExpiry. The results are as follows:

BorrowerId	BorrowerName	Address	PhoneNo	Email	MembershipType	MembershipExpiry
10	Emily Baker	707 Cedar St, Anyville	90123456780	emily@gmail.com	VIP	2025-08-05
* NULL	NULL	NULL	NULL	NULL	NULL	NULL

The interface also includes a toolbar with various icons for editing and viewing data, and a status bar at the bottom indicating 'borrowers 187'.

Explanation:

This SQL statement updates the "membertype" column to 'VIP' and extends the "membershipexpiry" date by one year for the borrower with ID 101 in the "borrowers" table.

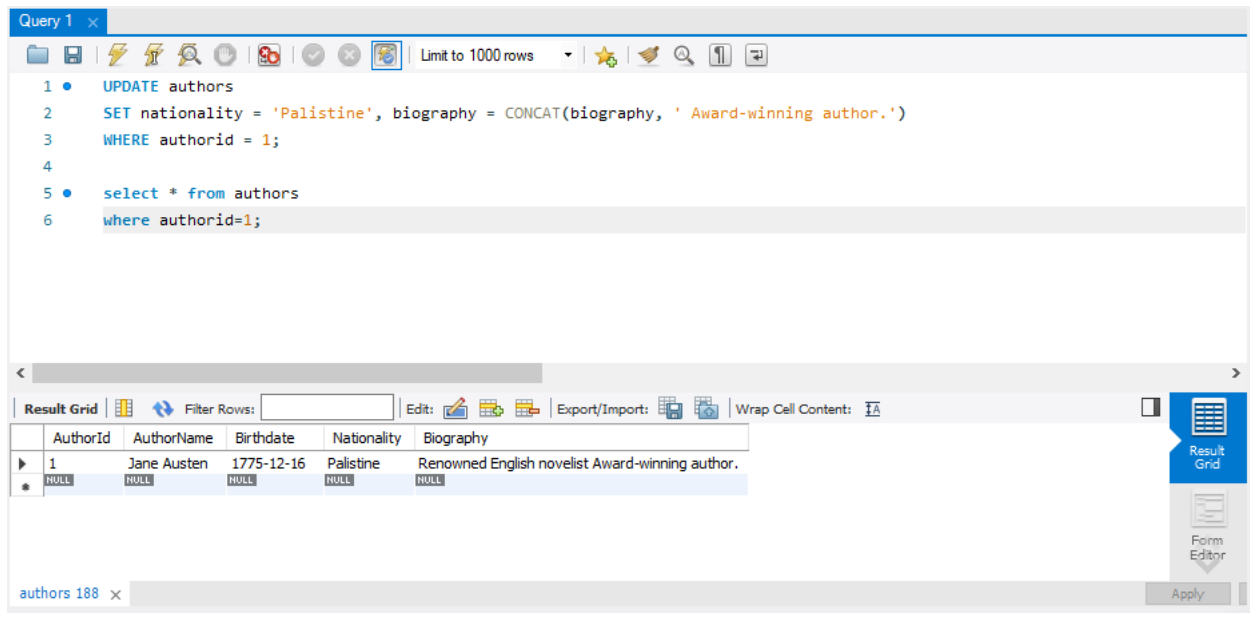
Query Number 2:

Query:

UPDATE authors

SET nationality = 'Palistine', biography = CONCAT(biography, ' Award-winning author.')

WHERE authorid = 1;



The screenshot shows a database query editor interface. The top section displays the SQL query:

```
1 • UPDATE authors
2   SET nationality = 'Palistine', biography = CONCAT(biography, ' Award-winning author.')
3   WHERE authorid = 1;
4
5 • select * from authors
6   where authorid=1;
```

Below the query editor, the "Result Grid" is visible, showing the results of the query. The grid has five columns: AuthorId, AuthorName, Birthdate, Nationality, and Biography. The first row shows the updated record for Jane Austen.

AuthorId	AuthorName	Birthdate	Nationality	Biography
1	Jane Austen	1775-12-16	Palistine	Renowned English novelist Award-winning author.

The interface also includes a toolbar with various icons for query execution, a "Limit to 1000 rows" dropdown, and a "Filter Rows" input field. The bottom status bar shows "authors 188" and an "Apply" button.

Explanation:

This SQL query updates the "nationality" column to 'Palestine' and appends 'Award-winning author.' to the "biography" column for the author with ID 1 in the "authors" table.

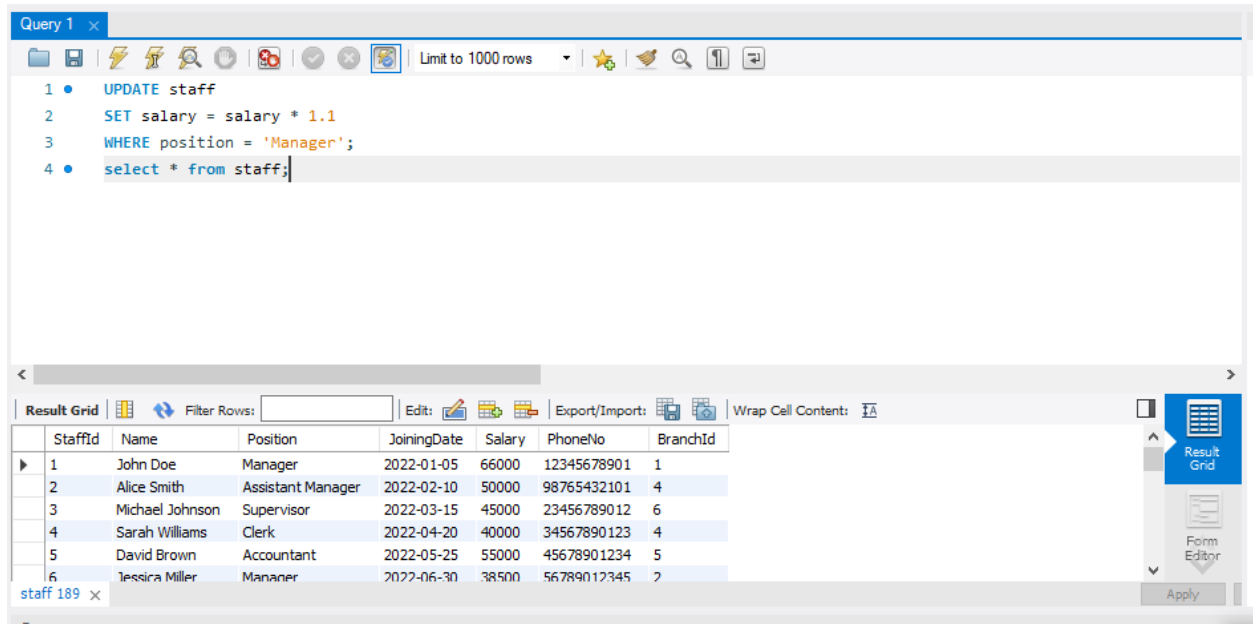
Query Number 3:

Query:

UPDATE staff

SET salary = salary * 1.1

WHERE position = 'Manager';



The screenshot shows a database query editor window titled "Query 1". The query text is as follows:

```
1 • UPDATE staff
2   SET salary = salary * 1.1
3   WHERE position = 'Manager';
4 • select * from staff;
```

Below the query editor, there is a "Result Grid" showing the results of the query. The grid has the following columns: StaffId, Name, Position, JoiningDate, Salary, PhoneNo, and BranchId. The results are as follows:

StaffId	Name	Position	JoiningDate	Salary	PhoneNo	BranchId
1	John Doe	Manager	2022-01-05	66000	12345678901	1
2	Alice Smith	Assistant Manager	2022-02-10	50000	98765432101	4
3	Michael Johnson	Supervisor	2022-03-15	45000	23456789012	6
4	Sarah Williams	Clerk	2022-04-20	40000	34567890123	4
5	David Brown	Accountant	2022-05-25	55000	45678901234	5
6	Jessica Miller	Manager	2022-06-30	38500	56789012345	2

The "Result Grid" tab is selected, and the "Apply" button is visible at the bottom right.

Explanation:

This SQL query increases the "salary" by 10% for all staff members whose "position" is 'Manager' in the "staff" table.

Query Number 4:

Query:

UPDATE nonfictionbook AS nf

SET academiclevel = CASE

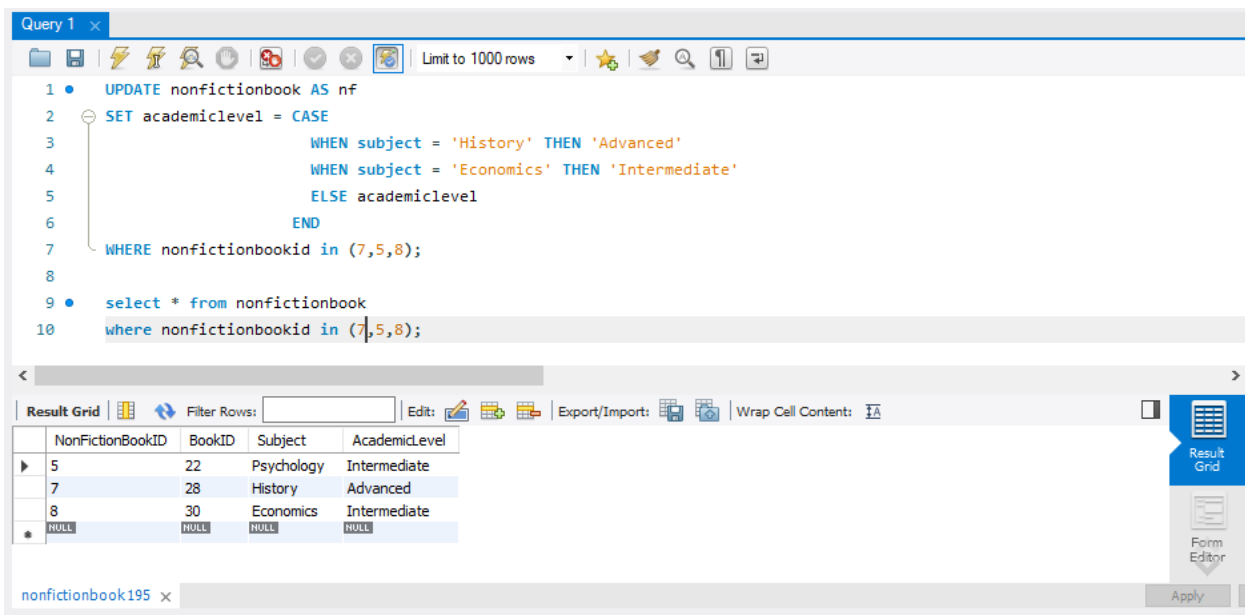
 WHEN subject = 'History' THEN 'Advanced'

 WHEN subject = 'Economics' THEN 'Intermediate'

 ELSE academiclevel

END

WHERE nonfictionbookid in (7,5,8);



The screenshot shows a SQL query editor with a query window and a result grid. The query is as follows:

```
1 • UPDATE nonfictionbook AS nf
2   SET academiclevel = CASE
3       WHEN subject = 'History' THEN 'Advanced'
4       WHEN subject = 'Economics' THEN 'Intermediate'
5       ELSE academiclevel
6   END
7   WHERE nonfictionbookid in (7,5,8);
8
9 • select * from nonfictionbook
10  where nonfictionbookid in (7,5,8);
```

The result grid shows the following data:

NonFictionBookID	BookID	Subject	AcademicLevel
5	22	Psychology	Intermediate
7	28	History	Advanced
8	30	Economics	Intermediate
*	NULL	NULL	NULL

The bottom of the screenshot shows the text "nonfictionbook195" and an "Apply" button.

Explanation:

This SQL statement updates the "academiclevel" column for non-fiction books with IDs 7, 5, and 8 in the "nonfictionbook" table. If the book's "subject" is 'History', it sets the "academiclevel" to 'Advanced'. If the subject is 'Economics', it sets the "academiclevel" to 'Intermediate'. Otherwise, it leaves the "academiclevel" unchanged.

Query Number 5:

Query:

UPDATE genres

SET rating = 4.5, description = 'A captivating genre'

WHERE genreid = 5;

The screenshot shows a database query editor interface. The top section displays the SQL query:

```
1 • UPDATE genres SET rating = 4.5,  
2   description = 'A captivating genre'  
3   WHERE genreid = 5;  
4  
5 • select * from genres  
6   where genreid=5;
```

Below the query editor, the "Result Grid" is visible, showing the results of the query. The grid has the following columns: GenreId, GenreType, rating, description, and genrelanguage. The first row shows the updated record for GenreId 5.

GenreId	GenreType	rating	description	genrelanguage
5	Mystery	4.5	A captivating genre	Spanish

The interface also includes a toolbar with various icons for query execution, a "Limit to 1000 rows" dropdown, and a "Form Editor" button on the right side.

Explanation:

This SQL query updates the "rating" to 4.5 and sets the "description" to 'A captivating genre' for the genre with ID 5 in the "genres" table.

Query Number 6:

Query:

UPDATE borrowers

SET email = 'abc@hotmail.com'

WHERE borrowerid = 10;

The screenshot shows a SQL query editor window titled "Query 1". The query text is as follows:

```
1 • UPDATE borrowers SET email = 'abc@hotmail.com' WHERE borrowerid = 10;  
2  
3  
4 • select * from borrowers  
5 where borrowerid = 10;
```

Below the query editor, the "Result Grid" is displayed, showing the data for the borrower with ID 10. The grid has the following columns: BorrowerId, BorrowerName, Address, PhoneNo, Email, MembershipType, and MembershipExpiry.

BorrowerId	BorrowerName	Address	PhoneNo	Email	MembershipType	MembershipExpiry
10	Emily Baker	707 Cedar St, Anyville	90123456780	abc@hotmail.com	VIP	2025-08-05

The bottom of the window shows a status bar with "borrowers 197" and an "Apply" button.

Explanation:

This SQL query updates the "email" to 'abc@hotmail.com' for the borrower with ID 10 in the "borrowers" table.

Query Number 7:

Query:

UPDATE books

SET title = 'Haunted King', pubyear = "2023-02-05"

WHERE bookid = 15;

The screenshot shows a database query editor interface. The top toolbar includes icons for file operations, query execution, and a 'Limit to 1000 rows' dropdown. The query editor contains the following SQL code:

```
1 • UPDATE books SET title = 'Haunted King', pubyear = "2023-02-05" WHERE bookid = 15;
2
3 • select * from books
4 where bookid = 15;
```

Below the query editor is a 'Result Grid' section. It includes a 'Filter Rows' input field, an 'Edit' button, and an 'Export/Import' button. The grid displays the following data:

	BookId	Title	PubYear	AuthorId	PublisherId	GenreId
▶	15	Haunted King	2023-02-05	1	1	1
*	NULL	NULL	NULL	NULL	NULL	NULL

On the right side of the interface, there are buttons for 'Result Grid' and 'Form Editor'. At the bottom, a status bar shows 'books 198' and an 'Apply' button.

Explanation:

This SQL query updates the "title" to 'Haunted King' and the "pubyear" to '2023-02-05' for the book with ID 15 in the "books" table.

Query Number 8:

Query:

UPDATE books

SET publisherid = (SELECT publisherid FROM publishers WHERE publishername = 'Penguin House'),

genreid = (SELECT genreid FROM genres WHERE genretypes = 'Sci-Fi')

WHERE bookid = 12;

The screenshot shows a SQL query editor window titled "Query 1" with a toolbar and a list of queries. The first query is selected and contains the following SQL code:

```
1 • UPDATE books
2   SET publisherid = (SELECT publisherid FROM publishers WHERE publishername = 'Penguin House'),
3     genreid = (SELECT genreid FROM genres WHERE genretypes = 'Sci-Fi')
4   WHERE bookid = 12;
5 • select * from books
6   where bookid=12;
```

Below the query editor is a "Result Grid" window showing the results of the query. The grid has columns: BookId, Title, PubYear, AuthorId, PublisherId, and GenreId. The first row shows the book with ID 12, titled "Harry Potter and the Philosopher's Stone", published in 1997-06-26, by author 12, with publisher 1, and genre NULL. The second row shows a book with ID NULL, title NULL, pub year NULL, author NULL, publisher NULL, and genre NULL.

BookId	Title	PubYear	AuthorId	PublisherId	GenreId
12	Harry Potter and the Philosopher's Stone	1997-06-26	12	1	NULL
NULL	NULL	NULL	NULL	NULL	NULL

At the bottom of the result grid, there is a tab labeled "books 200" and an "Apply" button.

Explanation:

This SQL query updates the "publisherid" to the ID of the publisher 'Penguin House' and the "genreid" to the ID of the genre 'Sci-Fi' for the book with ID 12 in the "books" table.

Query Number 9:

Query:

UPDATE borrowers

SET membershiptype = 'Premium'

WHERE borrowerid = 6;

The screenshot shows a SQL query editor with the following code:

```
1 • UPDATE borrowers SET membershiptype = 'Premium' WHERE borrowerid = 6;  
2  
3 • select * from borrowers  
4 where borrowerid=6;
```

Below the query editor, a "Result Grid" is displayed with the following data:

BorrowerId	BorrowerName	Address	PhoneNo	Email	MembershipType	MembershipExpiry
6	Jessica Taylor	303 Cedar St, Elsewhere	56789012340	jessica@gmail.com	Premium	2024-11-30

The interface also includes a toolbar with various icons, a "Filter Rows" input field, and a "Form Editor" button.

Explanation:

This SQL query updates the "membershiptype" to 'Premium' for the borrower with ID 6 in the "borrowers" table.

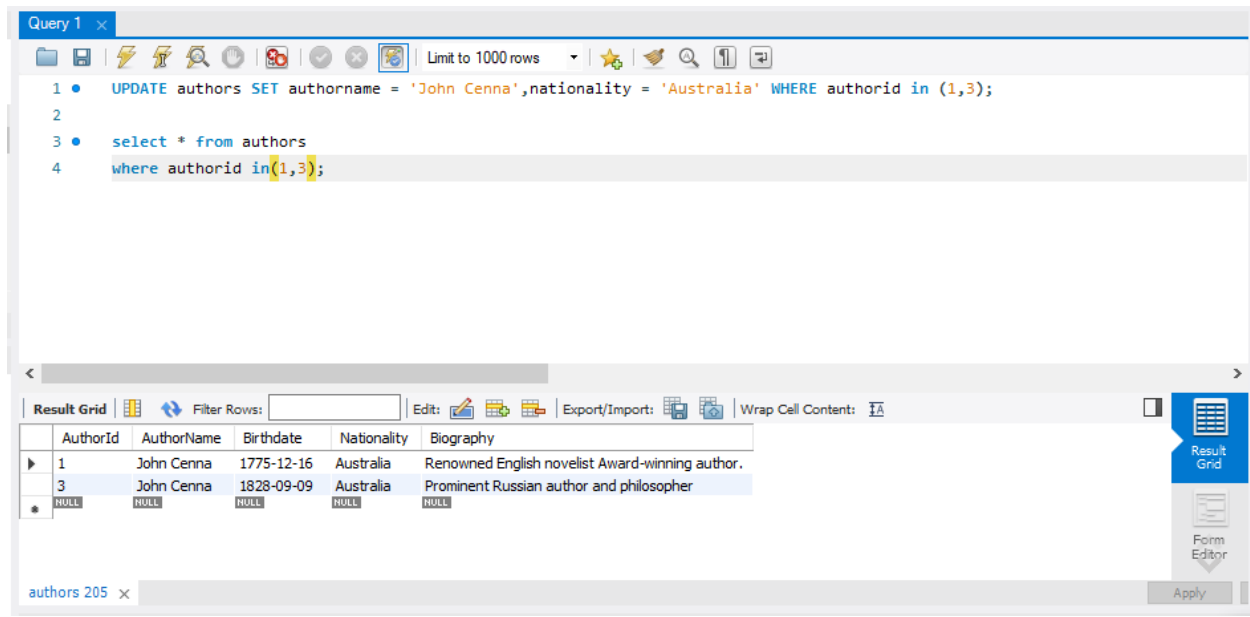
Query Number 10:

Query:

UPDATE authors

SET nationality = 'Australia'

WHERE authorid in (1,3);



Query 1

```
1 • UPDATE authors SET authorname = 'John Cenna',nationality = 'Australia' WHERE authorid in (1,3);
2
3 • select * from authors
4   where authorid in(1,3);
```

Limit to 1000 rows

Result Grid

AuthorId	AuthorName	Birthdate	Nationality	Biography
1	John Cenna	1775-12-16	Australia	Renowned English novelist Award-winning author.
3	John Cenna	1828-09-09	Australia	Prominent Russian author and philosopher
*	NULL	NULL	NULL	NULL

authors 205

Apply

Explanation:

This SQL query updates the "nationality" to 'Australia' for the authors with IDs 1 and 3 in the "authors" table.

Query Number 11:

Query:

UPDATE books

SET authorid = 8, publisherid = 3

WHERE bookid in (2,6,10);

The screenshot shows a database query editor interface. The top section displays the SQL query: `UPDATE books SET authorid = 8, publisherid = 3 WHERE bookid in (2,6,10);` followed by `select * from books where bookid in(2,6,10);`. A tooltip indicates to 'Execute the statement under the keyboard cursor'. Below the query editor, the 'Result Grid' is visible, showing the results of the SELECT statement. The table has columns: BookId, Title, PubYear, AuthorId, PublisherId, and GenreId. The results show three rows: BookId 2 (Great Expectations), BookId 6 (The Complete Poems of Emily Dickinson), and BookId 10 (The Old Man and the Sea). All three rows have AuthorId 8 and PublisherId 3. A fourth row shows NULL values for all columns. The interface also includes a toolbar with various icons and a status bar at the bottom indicating 'books 207'.

BookId	Title	PubYear	AuthorId	PublisherId	GenreId
2	Great Expectations	1861-12-19	8	3	2
6	The Complete Poems of Emily Dickinson	1955-01-01	8	3	5
10	The Old Man and the Sea	1952-09-01	8	3	4
*	NULL	NULL	NULL	NULL	NULL

Explanation:

This SQL query updates the "authorid" to 8 and the "publisherid" to 3 for the books with IDs 2, 6, and 10 in the "books" table.

Query Number 12:

Query:

UPDATE branches

SET location = 'Downtown'

WHERE branchid = 4;

The screenshot shows a SQL query editor window titled "Query 1". The query text is as follows:

```
1 • UPDATE branches SET location = 'Downtown' WHERE branchid = 4;
2
3 • select * from branches
4 • where branchid =4;
```

Below the query editor is a "Result Grid" showing the results of the query. The grid has five columns: BranchId, BranchName, Location, BranchManagerId, and branchcontactno. The first row shows the result for branchid 4, which is 'Westside Branch' at 'Downtown', managed by '16', with contact number '1597538460'. The second row shows 'NULL' for all columns.

BranchId	BranchName	Location	BranchManagerId	branchcontactno
4	Westside Branch	Downtown	16	1597538460
NULL	NULL	NULL	NULL	NULL

At the bottom of the window, there is a tab labeled "branches 208" and an "Apply" button.

Explanation:

This SQL query updates the "location" to 'Downtown' for the branch with ID 4 in the "branches" table.

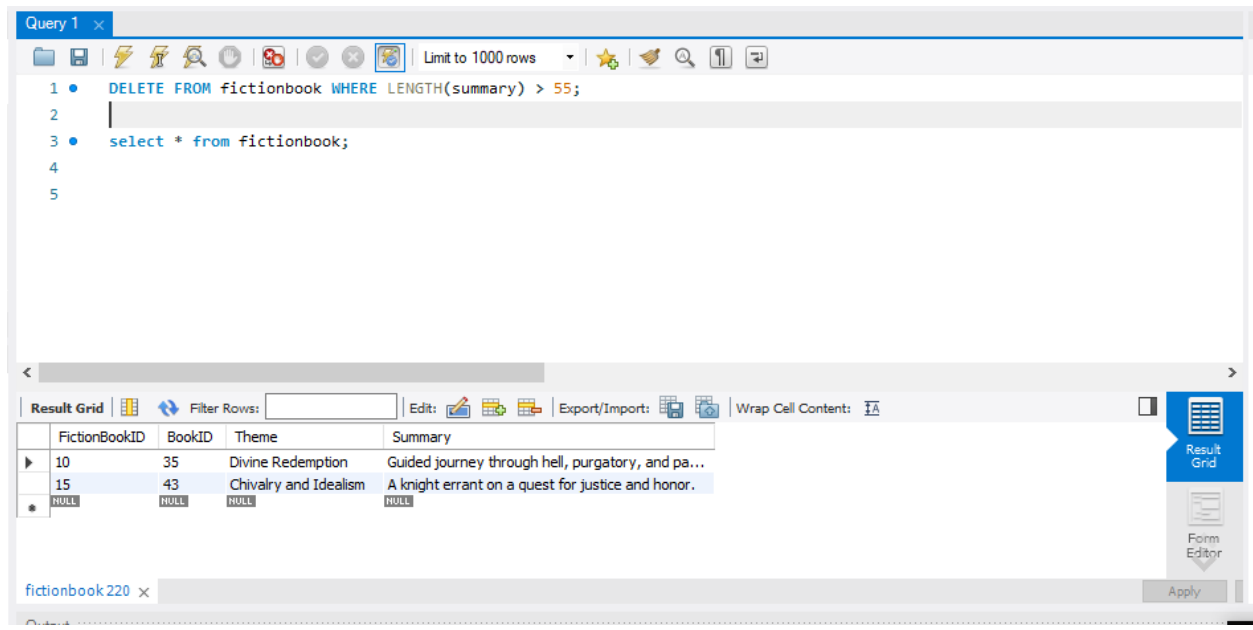
Delete Queries:

Query Number 1:

Query:

DELETE FROM fictionbook

WHERE LENGTH(summary) > 55;



Explanation:

This SQL query deletes rows from the "fictionbook" table where the length of the "summary" column is greater than 55 characters.

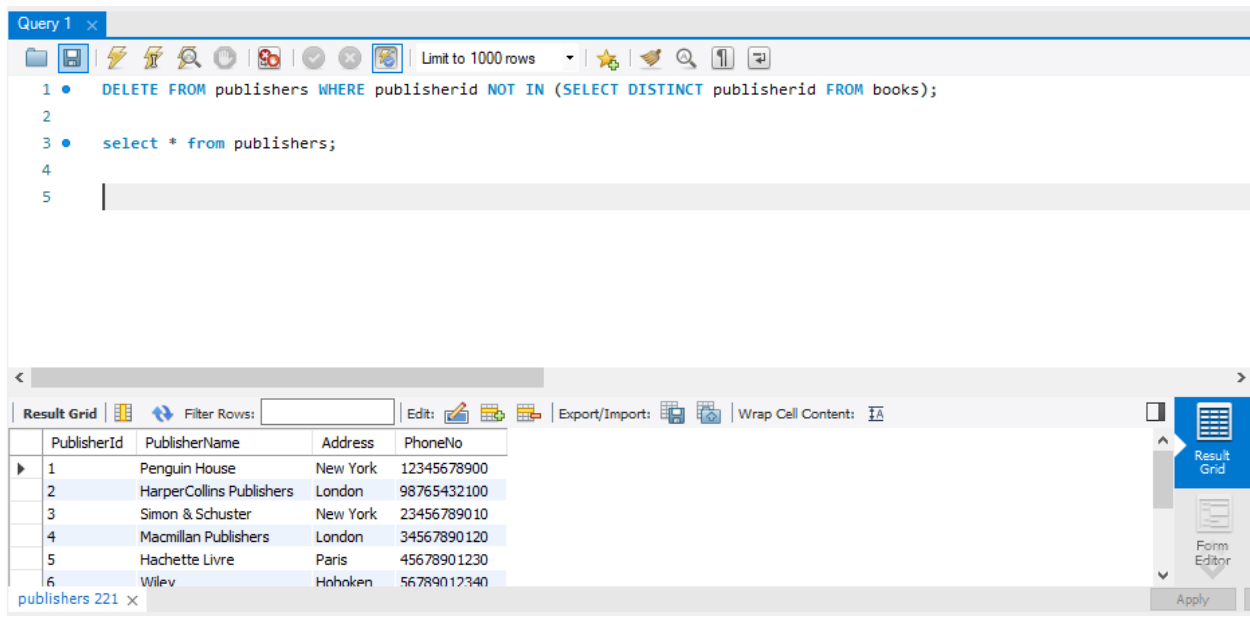
Query Number 2:

Query:

DELETE FROM publishers

WHERE publisherid NOT IN

(SELECT DISTINCT publisherid FROM books);



The screenshot shows a SQL query editor window titled "Query 1". The query text is as follows:

```
1 • DELETE FROM publishers WHERE publisherid NOT IN (SELECT DISTINCT publisherid FROM books);
2
3 • select * from publishers;
4
5
```

Below the query editor, the "Result Grid" is displayed, showing the results of the second query. The grid has four columns: PublisherId, PublisherName, Address, and PhoneNo. The results are as follows:

	PublisherId	PublisherName	Address	PhoneNo
1	1	Penguin House	New York	12345678900
2	2	HarperCollins Publishers	London	98765432100
3	3	Simon & Schuster	New York	23456789010
4	4	Macmillan Publishers	London	34567890120
5	5	Hachette Livre	Paris	45678901230
6	6	Wiley	Hoboken	56789012340

The status bar at the bottom indicates "publishers 221".

Explanation:

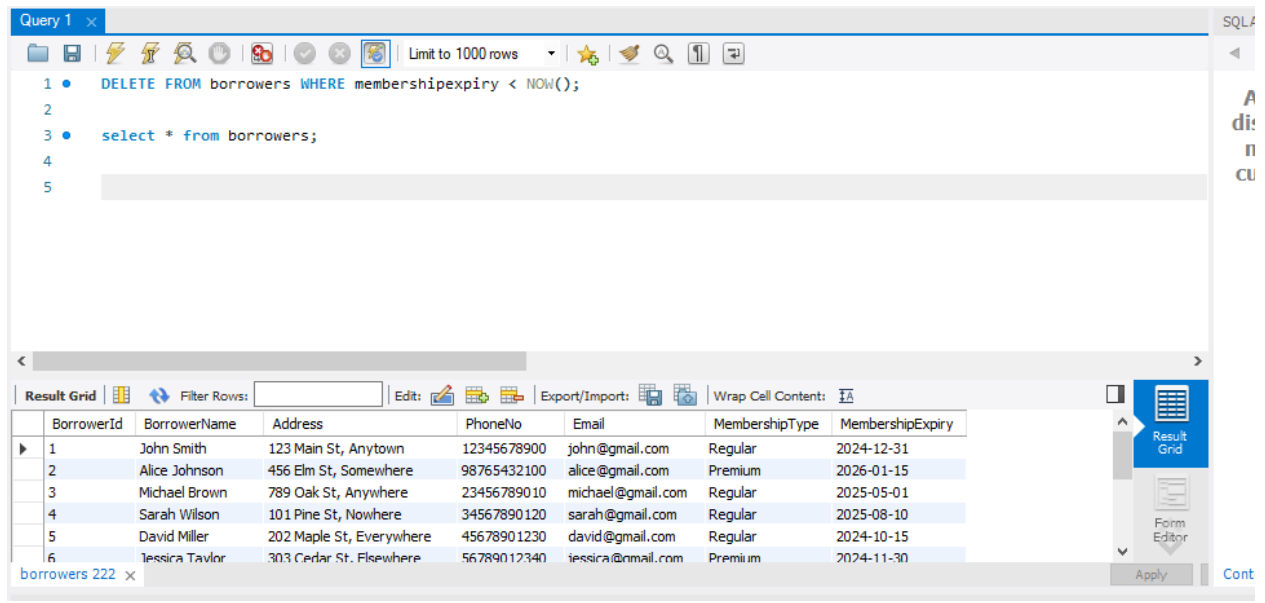
This SQL query deletes rows from the "publishers" table where the "publisherid" is not present in the list of distinct "publisherid" values from the "books" table.

Query Number 3:

Query:

DELETE FROM borrowers

WHERE membershipexpiry < NOW();



The screenshot shows a SQL query editor interface. The query is as follows:

```
1 • DELETE FROM borrowers WHERE membershipexpiry < NOW();
2
3 • select * from borrowers;
4
5
```

Below the query editor, the 'Result Grid' displays the following data:

BorrowerId	BorrowerName	Address	PhoneNo	Email	MembershipType	MembershipExpiry
1	John Smith	123 Main St, Anytown	12345678900	john@gmail.com	Regular	2024-12-31
2	Alice Johnson	456 Elm St, Somewhere	98765432100	alice@gmail.com	Premium	2026-01-15
3	Michael Brown	789 Oak St, Anywhere	23456789010	michael@gmail.com	Regular	2025-05-01
4	Sarah Wilson	101 Pine St, Nowhere	34567890120	sarah@gmail.com	Regular	2025-08-10
5	David Miller	202 Maple St, Everywhere	45678901230	david@gmail.com	Regular	2024-10-15
6	Jessica Taylor	303 Cedar St, Elsewhere	56789012340	jessica@gmail.com	Premium	2024-11-30

The interface also includes a toolbar with various icons, a 'Limit to 1000 rows' dropdown, and a 'Result Grid' button on the right side.

Explanation:

This SQL query deletes rows from the "borrowers" table where the "membershipexpiry" date is earlier than the current date and time.

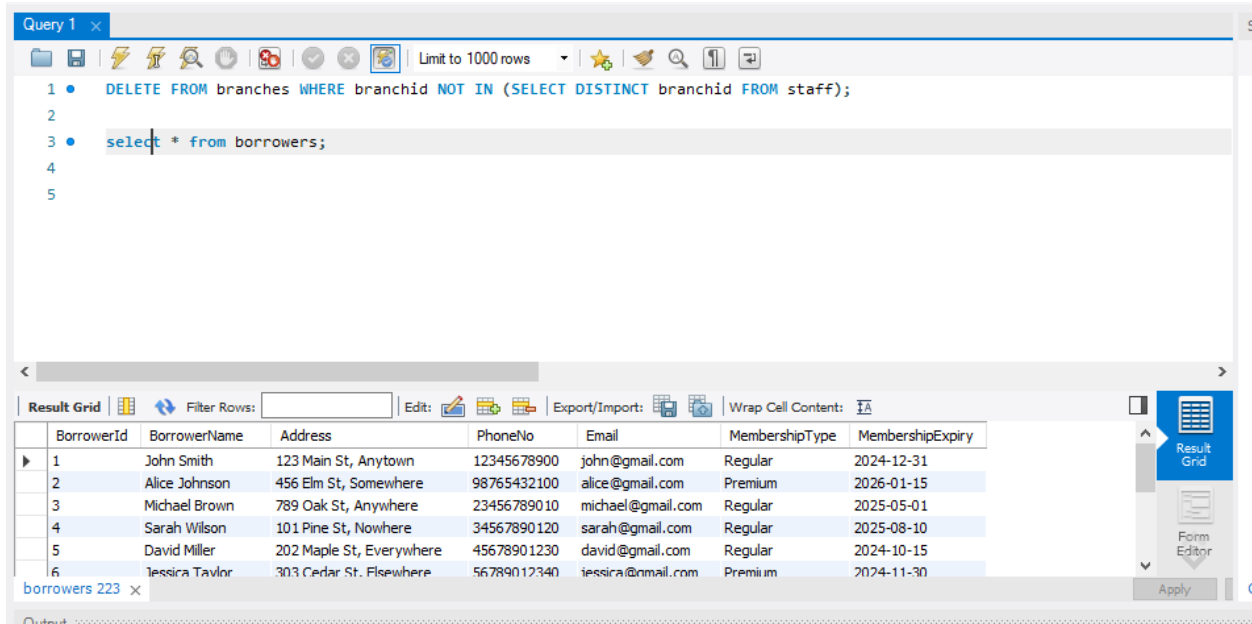
Query Number 4:

Query:

DELETE FROM branches

WHERE branchid NOT IN

(SELECT DISTINCT branchid FROM staff);



The screenshot shows a database query editor with a toolbar at the top. The query text is as follows:

```
1 • DELETE FROM branches WHERE branchid NOT IN (SELECT DISTINCT branchid FROM staff);
2
3 • select * from borrowers;
4
5
```

Below the query editor is a "Result Grid" showing data from the "borrowers" table. The grid has 7 columns: BorrowerId, BorrowerName, Address, PhoneNo, Email, MembershipType, and MembershipExpiry. There are 6 rows of data.

BorrowerId	BorrowerName	Address	PhoneNo	Email	MembershipType	MembershipExpiry
1	John Smith	123 Main St, Anytown	12345678900	john@gmail.com	Regular	2024-12-31
2	Alice Johnson	456 Elm St, Somewhere	98765432100	alice@gmail.com	Premium	2026-01-15
3	Michael Brown	789 Oak St, Anywhere	23456789010	michael@gmail.com	Regular	2025-05-01
4	Sarah Wilson	101 Pine St, Nowhere	34567890120	sarah@gmail.com	Regular	2025-08-10
5	David Miller	202 Maple St, Everywhere	45678901230	david@gmail.com	Regular	2024-10-15
6	Jessica Taylor	303 Cedar St, Elsewhere	56789012340	jessica@gmail.com	Premium	2024-11-30

At the bottom left of the result grid, it says "borrowers 223 x". On the right side, there are buttons for "Result Grid", "Form Editor", and "Apply".

Explanation:

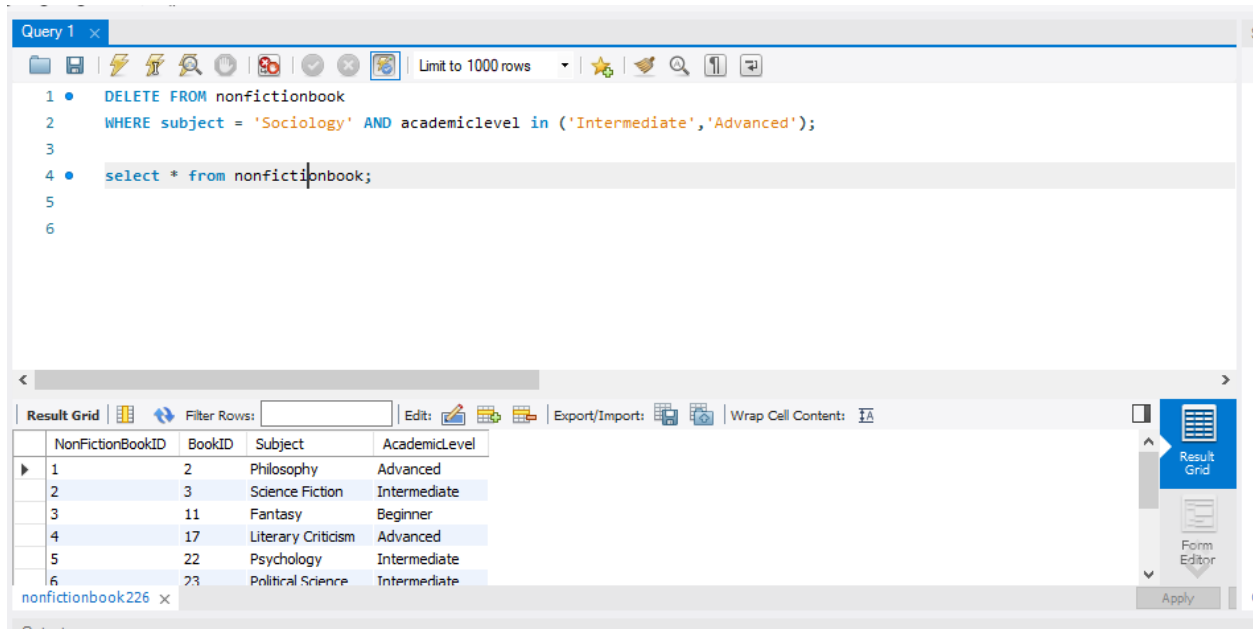
This SQL query deletes rows from the "branches" table where the "branchid" is not present in the list of distinct "branchid" values from the "staff" table.

Query Number 5:

Query:

DELETE FROM nonfictionbook

WHERE subject = 'Sociology' AND academiclevel IN ('Intermediate','Advanced');



The screenshot shows a database query editor window titled "Query 1". The query text is as follows:

```
1 • DELETE FROM nonfictionbook
2   WHERE subject = 'Sociology' AND academiclevel in ('Intermediate','Advanced');
3
4 • select * from nonfictionbook;
5
6
```

Below the query editor, there is a "Result Grid" tab. The grid displays the following data:

	NonFictionBookID	BookID	Subject	AcademicLevel
1	1	2	Philosophy	Advanced
2	2	3	Science Fiction	Intermediate
3	3	11	Fantasy	Beginner
4	4	17	Literary Criticism	Advanced
5	5	22	Psychology	Intermediate
6	6	23	Political Science	Intermediate

At the bottom left of the result grid, there is a tab labeled "nonfictionbook226". On the right side of the result grid, there are buttons for "Result Grid", "Form Editor", and "Apply".

Explanation:

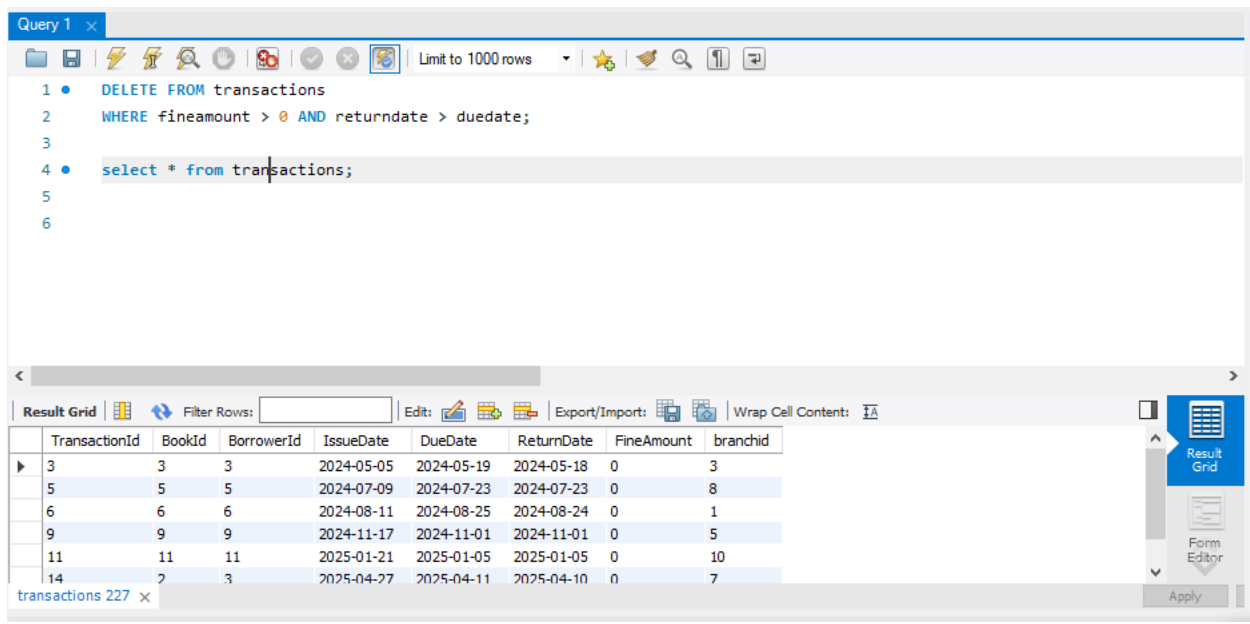
This SQL query deletes rows from the "nonfictionbook" table where the "subject" is 'Sociology' and the "academiclevel" is either 'Intermediate' or 'Advanced'.

Query Number 6:

Query:

DELETE FROM transactions

WHERE fineamount > 0 AND returndate > duedate;



The screenshot shows a database query editor interface. The top toolbar includes icons for saving, undo, redo, and other standard editing functions. Below the toolbar, the query text is displayed in a monospaced font. The query consists of two parts: a DELETE statement and a SELECT statement. The DELETE statement is on lines 1 and 2, and the SELECT statement is on line 4. The results of the query are shown in a table at the bottom. The table has columns for TransactionId, BookId, BorrowerId, IssueDate, DueDate, ReturnDate, FineAmount, and branchid. The results show 7 rows of data. The bottom of the interface includes a 'Filter Rows' section, an 'Edit' button, and an 'Export/Import' button. The 'Apply' button is also visible.

```
1 • DELETE FROM transactions
2   WHERE fineamount > 0 AND returndate > duedate;
3
4 • select * from transactions;
5
6
```

TransactionId	BookId	BorrowerId	IssueDate	DueDate	ReturnDate	FineAmount	branchid
3	3	3	2024-05-05	2024-05-19	2024-05-18	0	3
5	5	5	2024-07-09	2024-07-23	2024-07-23	0	8
6	6	6	2024-08-11	2024-08-25	2024-08-24	0	1
9	9	9	2024-11-17	2024-11-01	2024-11-01	0	5
11	11	11	2025-01-21	2025-01-05	2025-01-05	0	10
14	2	3	2025-04-27	2025-04-11	2025-04-10	0	7

Explanation:

This SQL query deletes rows from the "transactions" table where the "fineamount" is greater than 0 and the "returndate" is later than the "duedate".

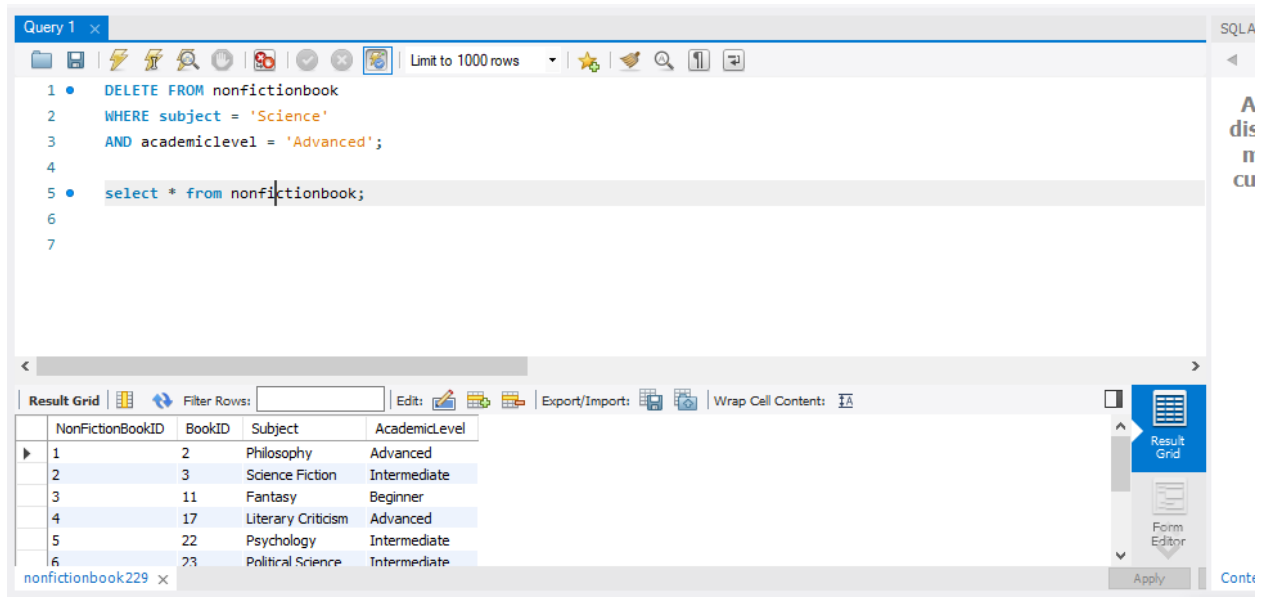
Query Number 7:

Query:

DELETE FROM nonfictionbook

WHERE subject = 'Science'

AND academic_level = 'Advanced';



The screenshot shows a SQL query editor window titled "Query 1". The query text is as follows:

```
1 • DELETE FROM nonfictionbook
2   WHERE subject = 'Science'
3   AND academic_level = 'Advanced';
4
5 • select * from nonfictionbook;
6
7
```

Below the query editor, the "Result Grid" is displayed, showing the following data:

NonFictionBookID	BookID	Subject	AcademicLevel
1	2	Philosophy	Advanced
2	3	Science Fiction	Intermediate
3	11	Fantasy	Beginner
4	17	Literary Criticism	Advanced
5	22	Psychology	Intermediate
6	23	Political Science	Intermediate

The interface includes a toolbar with various icons, a "Limit to 1000 rows" dropdown, and a "Result Grid" button on the right side. The bottom status bar shows "nonfictionbook229 x" and "Apply" buttons.

Explanation:

This SQL query deletes rows from the "nonfictionbook" table where the "subject" is 'Science' and the "academic_level" is 'Advanced'.

Query Number 8:

Query:

DELETE FROM transactions

WHERE fineamount = 0;

The screenshot shows a database query editor interface. The top toolbar includes icons for file operations, execution, and a 'Limit to 1000 rows' dropdown. The query editor contains the following SQL code:

```
1 • DELETE FROM transactions
2   WHERE fineamount = 0;
3
4
5 • select * from transactions;
6
7
```

Below the query editor is the 'Result Grid' section. It includes a 'Filter Rows' input field, an 'Edit' button, and an 'Export/Import' button. The result grid displays a single row of data with the following columns and values:

TransactionId	BookId	BorrowerId	IssueDate	DueDate	ReturnDate	FineAmount	branchid
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

On the right side of the result grid, there are buttons for 'Result Grid' and 'Form Editor'. At the bottom, a status bar shows 'transactions 230' and an 'Apply' button.

Explanation:

This SQL query deletes rows from the "transactions" table where the "fineamount" is equal to 0.

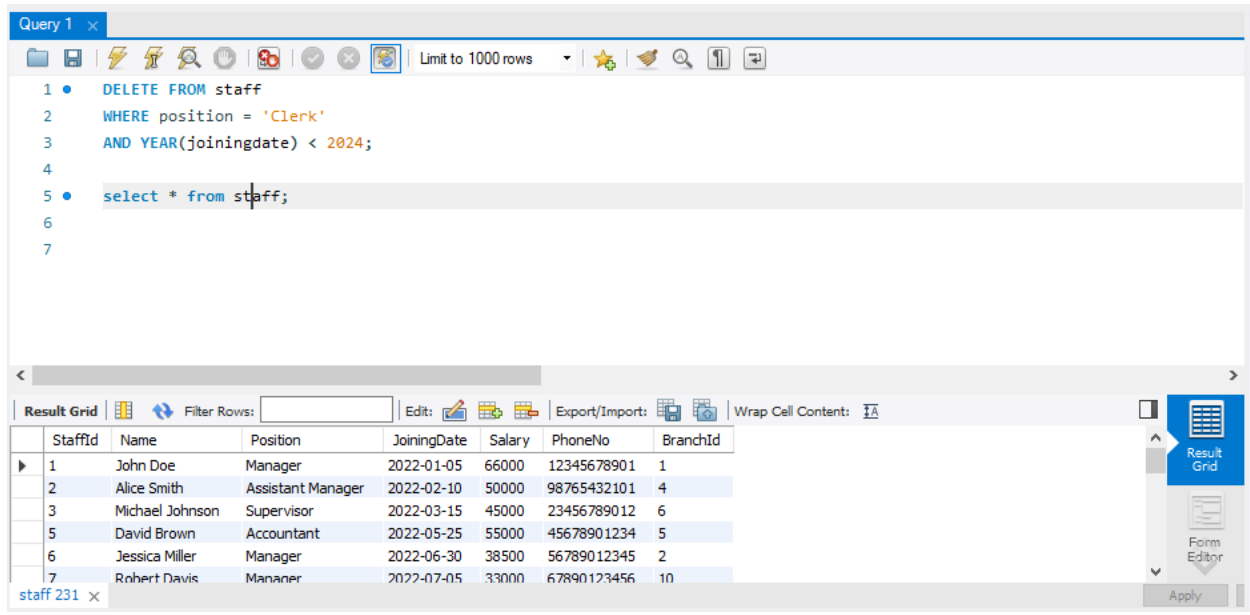
Query Number 9:

Query:

DELETE FROM staff

WHERE position = 'Clerk'

AND YEAR(joiningdate) < 2024;



The screenshot shows a database query editor interface. The top section displays the SQL query:

```
1 • DELETE FROM staff
2   WHERE position = 'Clerk'
3   AND YEAR(joiningdate) < 2024;
4
5 • select * from staff;
6
7
```

Below the query editor, the 'Result Grid' is visible, showing a table with 7 rows and 7 columns: StaffId, Name, Position, JoiningDate, Salary, PhoneNo, and BranchId. The data is as follows:

StaffId	Name	Position	JoiningDate	Salary	PhoneNo	BranchId
1	John Doe	Manager	2022-01-05	66000	12345678901	1
2	Alice Smith	Assistant Manager	2022-02-10	50000	98765432101	4
3	Michael Johnson	Supervisor	2022-03-15	45000	23456789012	6
5	David Brown	Accountant	2022-05-25	55000	45678901234	5
6	Jessica Miller	Manager	2022-06-30	38500	56789012345	2
7	Robert Davis	Manager	2022-07-05	33000	67890123456	10

The interface also includes a toolbar with various icons for file operations, a 'Limit to 1000 rows' dropdown, and a 'Filter Rows' input field. The bottom right corner features a 'Form Editor' button and an 'Apply' button.

Explanation:

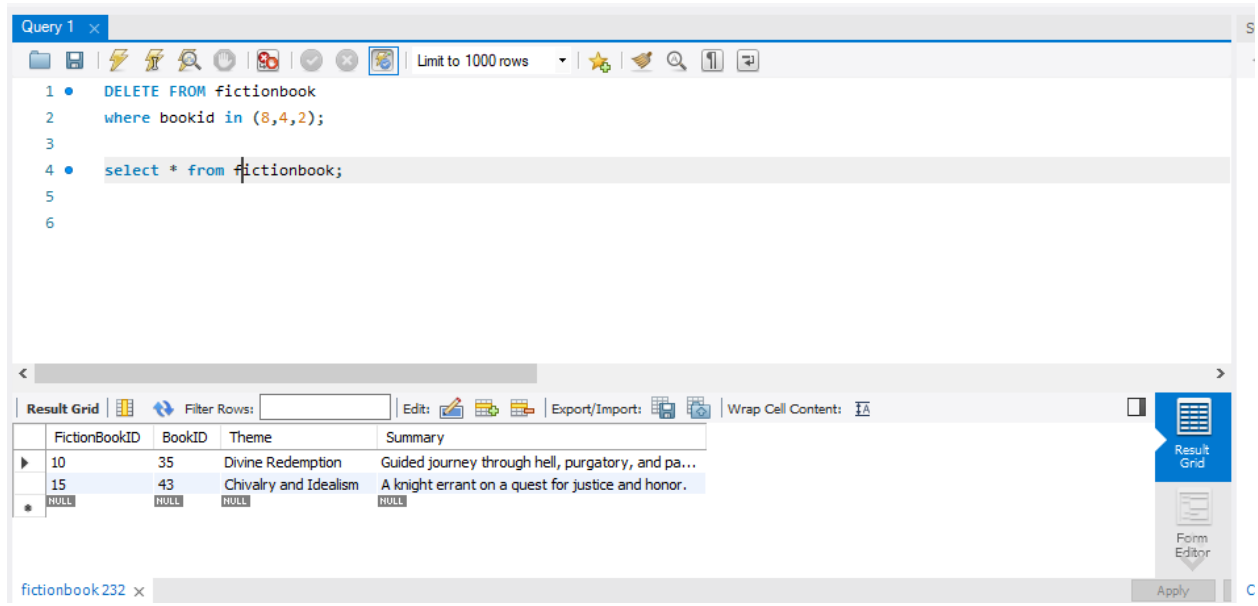
This SQL query deletes rows from the "staff" table where the "position" is 'Clerk' and the year of the "joiningdate" is before 2024.

Query Number 10:

Query:

DELETE FROM fictionbook

WHERE bookid IN (8,4,2);



The screenshot shows a SQL query editor window titled "Query 1". The query text is as follows:

```
1 • DELETE FROM fictionbook
2   where bookid in (8,4,2);
3
4 • select * from fictionbook;
5
6
```

Below the query editor is a "Result Grid" showing the results of the second query. The grid has four columns: FictionBookID, BookID, Theme, and Summary. The data is as follows:

FictionBookID	BookID	Theme	Summary
10	35	Divine Redemption	Guided journey through hell, purgatory, and pa...
15	43	Chivalry and Idealism	A knight errant on a quest for justice and honor.
* NULL	NULL	NULL	NULL

The interface includes various toolbars for editing, executing, and viewing the results. The "Result Grid" tab is active, and the "Form Editor" is also visible on the right side.

Explanation:

This SQL query deletes rows from the "fictionbook" table where the "bookid" is either 8, 4, or 2.

Query Number 11:

Query:

DELETE FROM genres

WHERE genretype = 'Horror';

The screenshot shows a database query editor window titled "Query 1" with a toolbar and a list of queries. The first query is highlighted:

```
1 • DELETE FROM genres where genretype = 'Horror';
2
3 • select * from genres;
4
5
```

Below the query editor is a "Result Grid" window showing a table with 6 rows and 5 columns: GenreId, GenreType, rating, description, and genrelanguage. The table data is as follows:

	GenreId	GenreType	rating	description	genrelanguage
▶	1	Fiction	4.5	Science fiction genre	Latin
	2	Non-Fiction	3.8	Fantasy genre	English
	3	Fantasy	4.2	Mystery genre	Latin
	4	Science Fiction	3.5	Romance genre	English
	5	Mystery	4.5	A captivating genre	Spanish
	6	Romance	3.9	Horror genre	German

The "Result Grid" window also includes a "Filter Rows" field, an "Edit" button, an "Export/Import" button, and a "Wrap Cell Content" button. A "Form Editor" button is visible on the right side of the window.

Explanation:

This SQL query deletes rows from the "genres" table where the "genretype" is 'Horror'.

Join Queries:

Query Number 1:

Query:

```
SELECT b.Title, a.AuthorName
```

```
FROM books b
```

```
INNER JOIN authors a ON b.AuthorId = a.AuthorId;
```

The screenshot shows a database management interface. On the left is a 'Navigator' pane with a 'SCHEMAS' section. Under 'lab8_bscs23122', there is a database named 'lms'. Inside 'lms', there are 'Tables' (authors, books, borrowers, branches, fictionbook, nonfictionbook, publishers, staff, transactions) and 'Views'. Below the tables, there are 'Stored Procedures'. The 'Administration' tab is selected, and the 'Schemas' sub-tab is active. The 'Information' pane at the bottom shows details for the 'authors' table: 'Table: authors' and 'Columns: AuthorId int PK'. The main area displays 'Query 1' with the following SQL code:

```
107
108 • SELECT b.Title, a.AuthorName
109 FROM books b
110 INNER JOIN authors a ON b.AuthorId = a.AuthorId;
111
```

Below the query is a 'Result Grid' showing the results of the query. The grid has two columns: 'Title' and 'AuthorName'. The results are as follows:

Title	AuthorName
Pride and Prejudice	Jane Austen
The Great Gatsby	Jane Austen
Frankenstein	Jane Austen
The Sun Also Rises	Jane Austen
Les Misérables	Jane Austen
Anna Karenina	Jane Austen
The Brothers Karamazov	Jane Austen
Great Expectations	Charles Dickens
To Kill a Mockingbird	Charles Dickens
The Odyssey	Charles Dickens
Alice's Adventures in W...	Charles Dickens
Crime and Punishment	Charles Dickens

Explanation:

This query retrieves book titles and their corresponding author names by joining the books and authors tables based on the author's ID.

Query Number 2:

Query:

SELECT b.Title, g.GenreType

FROM books b

INNER JOIN genres g

ON b.GenreId = g.GenreId;

The screenshot shows a database query editor with the following SQL query:

```
110 INNER JOIN authors a ON b.AuthorId = a.AuthorId;
111
112 • SELECT b.Title, g.GenreType
113 FROM books b
114 INNER JOIN genres g ON b.GenreId = g.GenreId;
```

Below the query editor, the 'Result Grid' tab is active, displaying the results of the query. The results are as follows:

Title	GenreType
Pride and Prejudice	Fiction
Jane Eyre	Fiction
The Great Gatsby	Fiction
The Great Gatsby	Fiction
To Kill a Mockingbird	Fiction
Moby Dick	Fiction
The Odyssey	Fiction
The Lord of the Rings	Fiction
Don Quixote	Fiction
The Grapes of Wrath	Fiction
The Sun Also Rises	Fiction
Heart of Darkness	Fiction

The interface also includes a 'Filter Rows' field, an 'Export' button, and a 'Wrap Cell Content' checkbox. The bottom of the window shows a status bar with 'PK char(100)' and '12'.

Explanation:

This query fetches book titles along with their corresponding genre types by joining the books and genres tables based on the genre ID.

Query Number 3:

Query:

SELECT t.TransactionId, b.Title

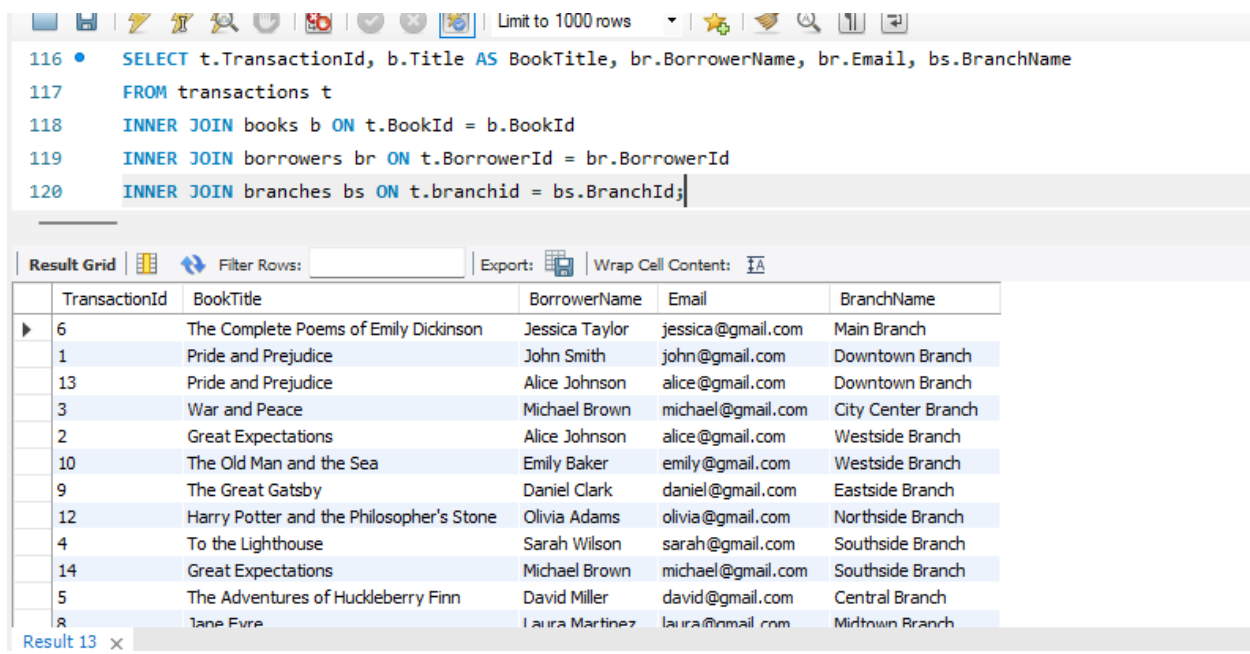
AS BookTitle, br.BorrowerName, br.Email, bs.BranchName

FROM transactions t

INNER JOIN books b ON t.BookId = b.BookId

INNER JOIN borrowers br ON t.BorrowerId = br.BorrowerId

INNER JOIN branches bs ON t.branchid = bs.BranchId;



The screenshot shows a SQL query editor with a toolbar at the top. The query is as follows:

```
116 • SELECT t.TransactionId, b.Title AS BookTitle, br.BorrowerName, br.Email, bs.BranchName
117 FROM transactions t
118 INNER JOIN books b ON t.BookId = b.BookId
119 INNER JOIN borrowers br ON t.BorrowerId = br.BorrowerId
120 INNER JOIN branches bs ON t.branchid = bs.BranchId;
```

Below the query editor is the 'Result Grid' tab, which displays the results of the query. The grid has columns for TransactionId, BookTitle, BorrowerName, Email, and BranchName. The results are as follows:

	TransactionId	BookTitle	BorrowerName	Email	BranchName
▶	6	The Complete Poems of Emily Dickinson	Jessica Taylor	jessica@gmail.com	Main Branch
	1	Pride and Prejudice	John Smith	john@gmail.com	Downtown Branch
	13	Pride and Prejudice	Alice Johnson	alice@gmail.com	Downtown Branch
	3	War and Peace	Michael Brown	michael@gmail.com	City Center Branch
	2	Great Expectations	Alice Johnson	alice@gmail.com	Westside Branch
	10	The Old Man and the Sea	Emily Baker	emily@gmail.com	Westside Branch
	9	The Great Gatsby	Daniel Clark	daniel@gmail.com	Eastside Branch
	12	Harry Potter and the Philosopher's Stone	Olivia Adams	olivia@gmail.com	Northside Branch
	4	To the Lighthouse	Sarah Wilson	sarah@gmail.com	Southside Branch
	14	Great Expectations	Michael Brown	michael@gmail.com	Southside Branch
	5	The Adventures of Huckleberry Finn	David Miller	david@gmail.com	Central Branch
	8	Jane Eyre	Laura Martinez	laura@gmail.com	Midtown Branch

At the bottom left of the result grid, it says 'Result 13' with a close button (X).

Explanation:

This query retrieves transaction IDs, book titles, borrower names, borrower emails, and branch names by joining the transactions, books, borrowers, and branches tables based on their respective IDs.

Query Number 4:

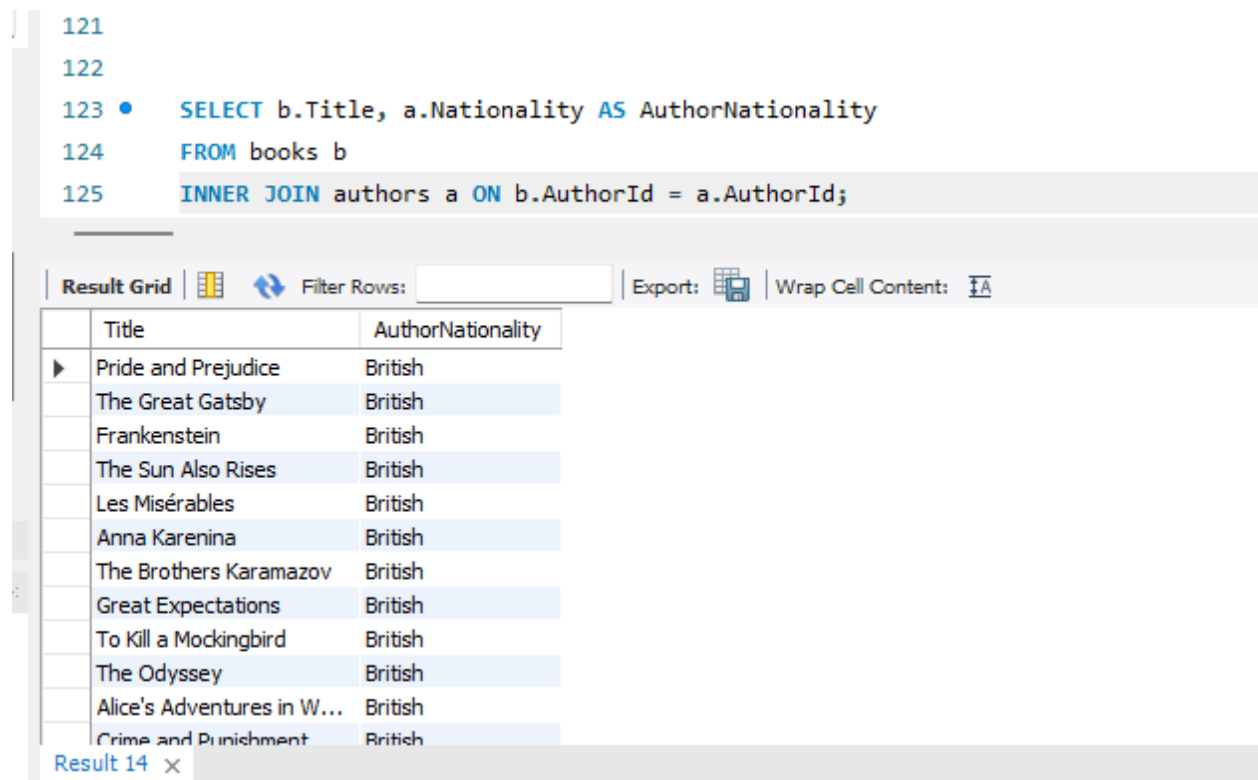
Query:

```
SELECT b.Title, a.Nationality
```

```
AS AuthorNationality
```

```
FROM books b
```

```
INNER JOIN authors a ON b.AuthorId = a.AuthorId;
```



```
121
122
123 • SELECT b.Title, a.Nationality AS AuthorNationality
124 FROM books b
125 INNER JOIN authors a ON b.AuthorId = a.AuthorId;
```

Title	AuthorNationality
Pride and Prejudice	British
The Great Gatsby	British
Frankenstein	British
The Sun Also Rises	British
Les Misérables	British
Anna Karenina	British
The Brothers Karamazov	British
Great Expectations	British
To Kill a Mockingbird	British
The Odyssey	British
Alice's Adventures in W...	British
Crime and Punishment	British

Result 14 x

Explanation:

This query fetches book titles along with the nationality of their respective authors by joining the books and authors tables based on the author's ID.

Query Number 5:

Query:

```
SELECT t.TransactionId, b.Title AS BookTitle, br.BorrowerName, br.Email, bs.BranchName,  
t.FineAmount
```

```
FROM transactions t
```

```
INNER JOIN books b ON t.BookId = b.BookId
```

```
INNER JOIN borrowers br ON t.BorrowerId = br.BorrowerId
```

```
INNER JOIN branches bs ON t.branchid = bs.BranchId
```

```
WHERE t.FineAmount > 0;
```

```
126 • SELECT t.TransactionId, b.Title AS BookTitle, br.BorrowerName, br.Email, bs.BranchName, t.FineAmount
127 FROM transactions t
128 INNER JOIN books b ON t.BookId = b.BookId
129 INNER JOIN borrowers br ON t.BorrowerId = br.BorrowerId
130 INNER JOIN branches bs ON t.branchid = bs.BranchId
131 WHERE t.FineAmount > 0;
```

TransactionId	BookTitle	BorrowerName	Email	BranchName	FineAmount
1	Pride and Prejudice	John Smith	john@gmail.com	Downtown Branch	200
2	Great Expectations	Alice Johnson	alice@gmail.com	Westside Branch	400
4	To the Lighthouse	Sarah Wilson	sarah@gmail.com	Southside Branch	500
7	1984	Robert Anderson	robert@gmail.com	Uptown Branch	300
8	Jane Eyre	Laura Martinez	laura@gmail.com	Midtown Branch	100
10	The Old Man and the Sea	Emily Baker	emily@gmail.com	Westside Branch	200
12	Harry Potter and the Philosopher's Stone	Olivia Adams	olivia@gmail.com	Northside Branch	800
13	Pride and Prejudice	Alice Johnson	alice@gmail.com	Downtown Branch	1200

Result 15

Explanation:

This query retrieves transaction IDs, book titles, borrower names, borrower emails, branch names, and fine amounts for transactions with fines greater than 0, joining the transactions, books, borrowers, and branches tables based on their respective IDs.

Query Number 6:

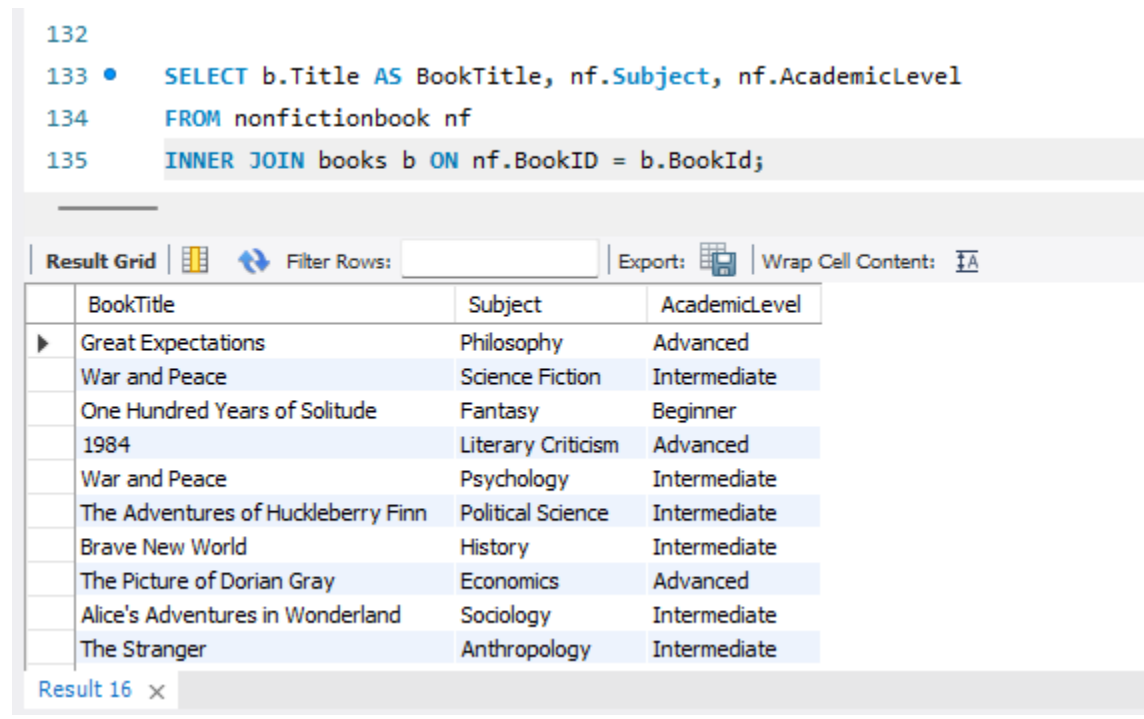
Query:

SELECT b.Title

AS BookTitle, nf.Subject, nf.AcademicLevel

FROM nonfictionbook nf

INNER JOIN books b ON nf.BookID = b.BookId;



```
132
133 • SELECT b.Title AS BookTitle, nf.Subject, nf.AcademicLevel
134 FROM nonfictionbook nf
135 INNER JOIN books b ON nf.BookID = b.BookId;
```

BookTitle	Subject	AcademicLevel
Great Expectations	Philosophy	Advanced
War and Peace	Science Fiction	Intermediate
One Hundred Years of Solitude	Fantasy	Beginner
1984	Literary Criticism	Advanced
War and Peace	Psychology	Intermediate
The Adventures of Huckleberry Finn	Political Science	Intermediate
Brave New World	History	Intermediate
The Picture of Dorian Gray	Economics	Advanced
Alice's Adventures in Wonderland	Sociology	Intermediate
The Stranger	Anthropology	Intermediate

Result 16 x

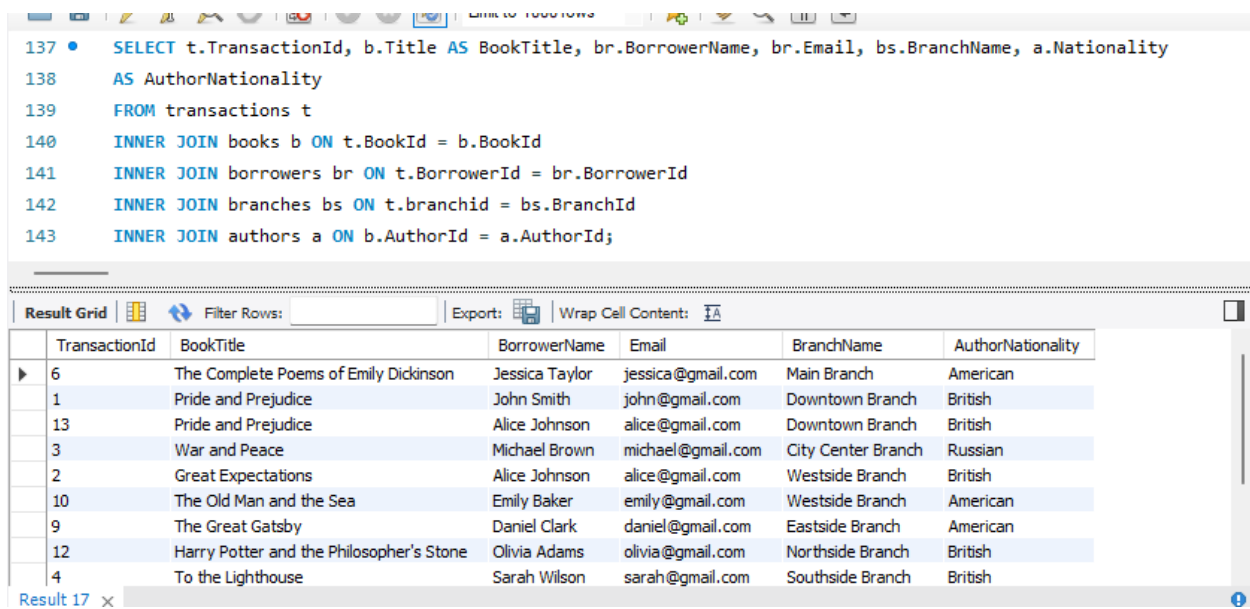
Explanation:

This query selects the titles of non-fiction books along with their subjects and academic levels, joining the nonfictionbook and books tables based on the book IDs.

Query Number 7:

Query:

```
SELECT t.TransactionId, b.Title  
  
AS BookTitle, br.BorrowerName, br.Email, bs.BranchName, a.Nationality  
  
AS AuthorNationality  
  
FROM transactions t  
  
INNER JOIN books b ON t.BookId = b.BookId  
  
INNER JOIN borrowers br ON t.BorrowerId = br.BorrowerId  
  
INNER JOIN branches bs ON t.branchid = bs.BranchId  
  
INNER JOIN authors a ON b.AuthorId = a.AuthorId;
```



```
137 • SELECT t.TransactionId, b.Title AS BookTitle, br.BorrowerName, br.Email, bs.BranchName, a.Nationality  
138 AS AuthorNationality  
139 FROM transactions t  
140 INNER JOIN books b ON t.BookId = b.BookId  
141 INNER JOIN borrowers br ON t.BorrowerId = br.BorrowerId  
142 INNER JOIN branches bs ON t.branchid = bs.BranchId  
143 INNER JOIN authors a ON b.AuthorId = a.AuthorId;
```

TransactionId	BookTitle	BorrowerName	Email	BranchName	AuthorNationality
6	The Complete Poems of Emily Dickinson	Jessica Taylor	jessica@gmail.com	Main Branch	American
1	Pride and Prejudice	John Smith	john@gmail.com	Downtown Branch	British
13	Pride and Prejudice	Alice Johnson	alice@gmail.com	Downtown Branch	British
3	War and Peace	Michael Brown	michael@gmail.com	City Center Branch	Russian
2	Great Expectations	Alice Johnson	alice@gmail.com	Westside Branch	British
10	The Old Man and the Sea	Emily Baker	emily@gmail.com	Westside Branch	American
9	The Great Gatsby	Daniel Clark	daniel@gmail.com	Eastside Branch	American
12	Harry Potter and the Philosopher's Stone	Olivia Adams	olivia@gmail.com	Northside Branch	British
4	To the Lighthouse	Sarah Wilson	sarah@gmail.com	Southside Branch	British

Explanation:

This query retrieves transaction IDs, book titles, borrower names, borrower emails, branch names, and author nationalities by joining the transactions, books, borrowers, branches, and authors tables based on their respective IDs.

Query Number 8:

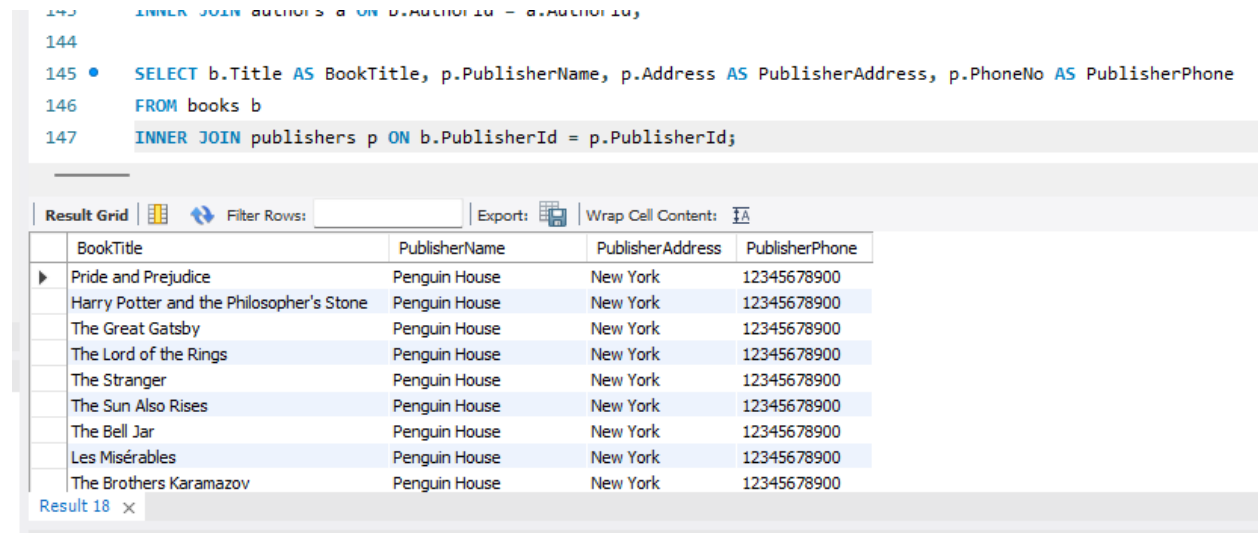
Query:

```
SELECT b.Title AS BookTitle, p.PublisherName, p.Address
```

```
AS PublisherAddress, p.PhoneNo AS PublisherPhone
```

```
FROM books b
```

```
INNER JOIN publishers p ON b.PublisherId = p.PublisherId;
```



The screenshot shows a SQL query editor with the following query:

```
143 INNER JOIN publishers p ON b.PublisherId = p.PublisherId;
144
145 • SELECT b.Title AS BookTitle, p.PublisherName, p.Address AS PublisherAddress, p.PhoneNo AS PublisherPhone
146 FROM books b
147 INNER JOIN publishers p ON b.PublisherId = p.PublisherId;
```

Below the query, there is a toolbar with options: Result Grid, Filter Rows, Export, and Wrap Cell Content. The results are displayed in a table with the following columns: BookTitle, PublisherName, PublisherAddress, and PublisherPhone.

BookTitle	PublisherName	PublisherAddress	PublisherPhone
Pride and Prejudice	Penguin House	New York	12345678900
Harry Potter and the Philosopher's Stone	Penguin House	New York	12345678900
The Great Gatsby	Penguin House	New York	12345678900
The Lord of the Rings	Penguin House	New York	12345678900
The Stranger	Penguin House	New York	12345678900
The Sun Also Rises	Penguin House	New York	12345678900
The Bell Jar	Penguin House	New York	12345678900
Les Misérables	Penguin House	New York	12345678900
The Brothers Karamazov	Penguin House	New York	12345678900

Result 18 x

Explanation:

This query selects the titles of books along with their publisher names, addresses, and phone numbers by joining the books and publishers tables based on the publisher IDs.

Query Number 9:

Query:

```
SELECT b.BranchName, s.Name AS ManagerName
```

```
FROM branches b
```

```
INNER JOIN staff s ON b.BranchManagerId = s.StaffId;
```

149

150 • `SELECT b.BranchName, s.Name AS ManagerName`

151 `FROM branches b`

152 `INNER JOIN staff s ON b.BranchManagerId = s.StaffId;`

Result Grid			Filter Rows:	Export:	Wrap Cell Content:
	BranchName	ManagerName			
▶	Main Branch	John Doe			
	Downtown Branch	Jessica Miller			
	City Center Branch	William Anderson			
	Westside Branch	Sophia Martinez			
	Eastside Branch	James Taylor			
	Northside Branch	Sophia Clark			
	Southside Branch	Emma White			
	Central Branch	William Wilson			
	Midtown Branch	Christopher Moore			

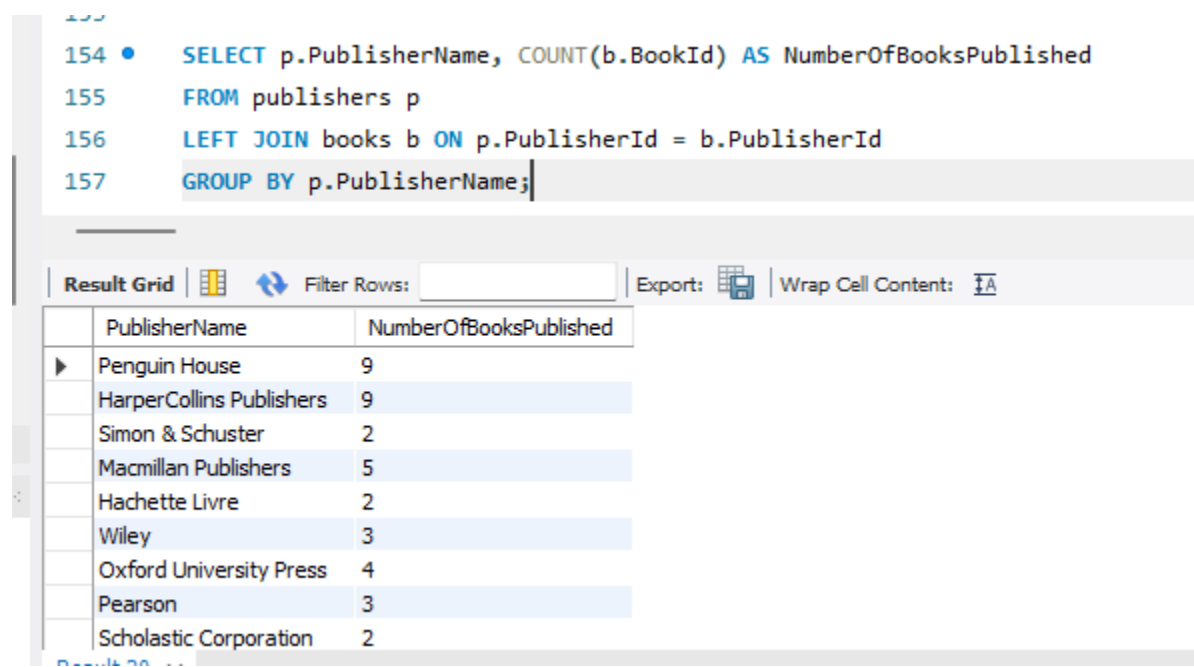
Explanation:

This query retrieves the branch names along with the names of their respective managers by joining the branches and staff tables based on the manager's staff ID.

Query Number 10:

Query:

```
SELECT p.PublisherName, COUNT(b.BookId)
AS NumberOfBooksPublished
FROM publishers p
LEFT JOIN books b ON p.PublisherId = b.PublisherId
GROUP BY p.PublisherName;
```



The screenshot shows a SQL query editor with the following query:

```
154 • SELECT p.PublisherName, COUNT(b.BookId) AS NumberOfBooksPublished
155 FROM publishers p
156 LEFT JOIN books b ON p.PublisherId = b.PublisherId
157 GROUP BY p.PublisherName;
```

Below the query, there is a toolbar with options: Result Grid, Filter Rows, Export, and Wrap Cell Content. The results are displayed in a table with two columns: PublisherName and NumberOfBooksPublished.

PublisherName	NumberOfBooksPublished
Penguin House	9
HarperCollins Publishers	9
Simon & Schuster	2
Macmillan Publishers	5
Hachette Livre	2
Wiley	3
Oxford University Press	4
Pearson	3
Scholastic Corporation	2

Explanation:

This query counts the number of books published by each publisher, listing their publisher names alongside the corresponding count. It performs a left join between the publishers and books tables on the publisher ID, grouping the results by publisher name.

Query Number 11:

Query:

```
SELECT a.AuthorName, COUNT(b.BookId)
```

```
AS NumberOfBooks
```

```
FROM authors a
```

```
LEFT JOIN books b ON a.AuthorId = b.AuthorId
```

```
GROUP BY a.AuthorName;
```

```
159 • SELECT a.AuthorName, COUNT(b.BookId) AS NumberOfBooks
160 FROM authors a
161 LEFT JOIN books b ON a.AuthorId = b.AuthorId
162 GROUP BY a.AuthorName;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	AuthorName	NumberOfBooks		
▶	Jane Austen	7		
	Charles Dickens	7		
	Leo Tolstoy	6		
	Virginia Woolf	2		
	Mark Twain	4		
	Emily Dickinson	4		
	George Orwell	4		
	Charlotte Bronte	2		
	F. Scott Fitzgerald	2		

Explanation:

This query counts the number of books authored by each author, displaying their names alongside the corresponding count. It utilizes a left join between the authors and books tables on the author ID, grouping the results by author name.

Query Number 12:

Query:

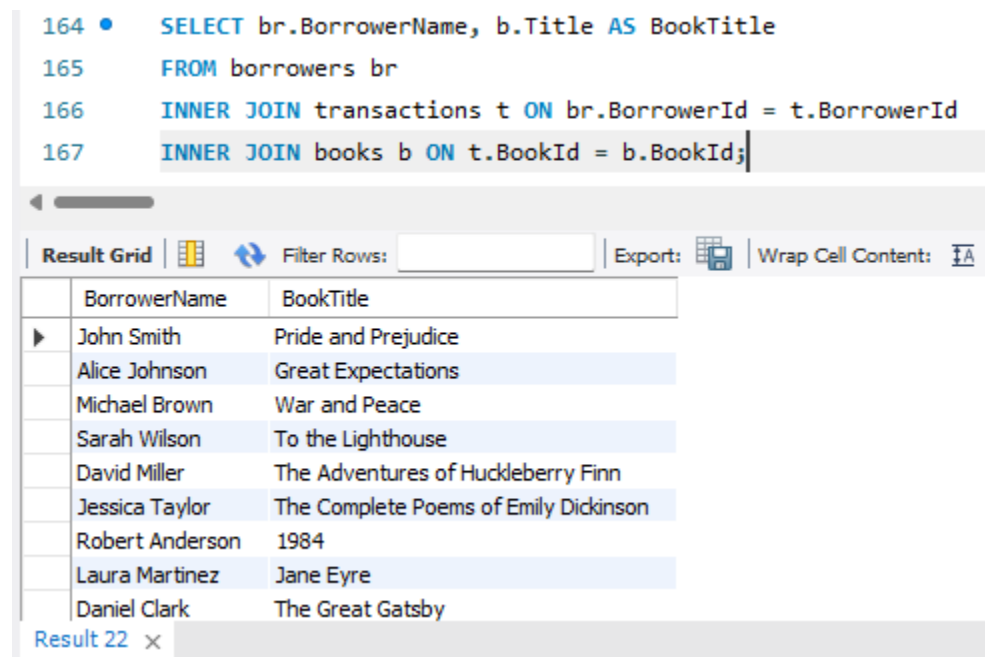
```
SELECT br.BorrowerName, b.Title
```

```
AS BookTitle
```

```
FROM borrowers br
```

```
INNER JOIN transactions t ON br.BorrowerId = t.BorrowerId
```

```
INNER JOIN books b ON t.BookId = b.BookId;
```



```
164 • SELECT br.BorrowerName, b.Title AS BookTitle
165 FROM borrowers br
166 INNER JOIN transactions t ON br.BorrowerId = t.BorrowerId
167 INNER JOIN books b ON t.BookId = b.BookId;
```

	BorrowerName	BookTitle
▶	John Smith	Pride and Prejudice
	Alice Johnson	Great Expectations
	Michael Brown	War and Peace
	Sarah Wilson	To the Lighthouse
	David Miller	The Adventures of Huckleberry Finn
	Jessica Taylor	The Complete Poems of Emily Dickinson
	Robert Anderson	1984
	Laura Martinez	Jane Eyre
	Daniel Clark	The Great Gatsby

Result 22 x

Explanation:

This query retrieves borrower names along with the titles of books they have borrowed by joining the borrowers, transactions, and books tables based on their respective IDs.

Query Number 13:

Query:

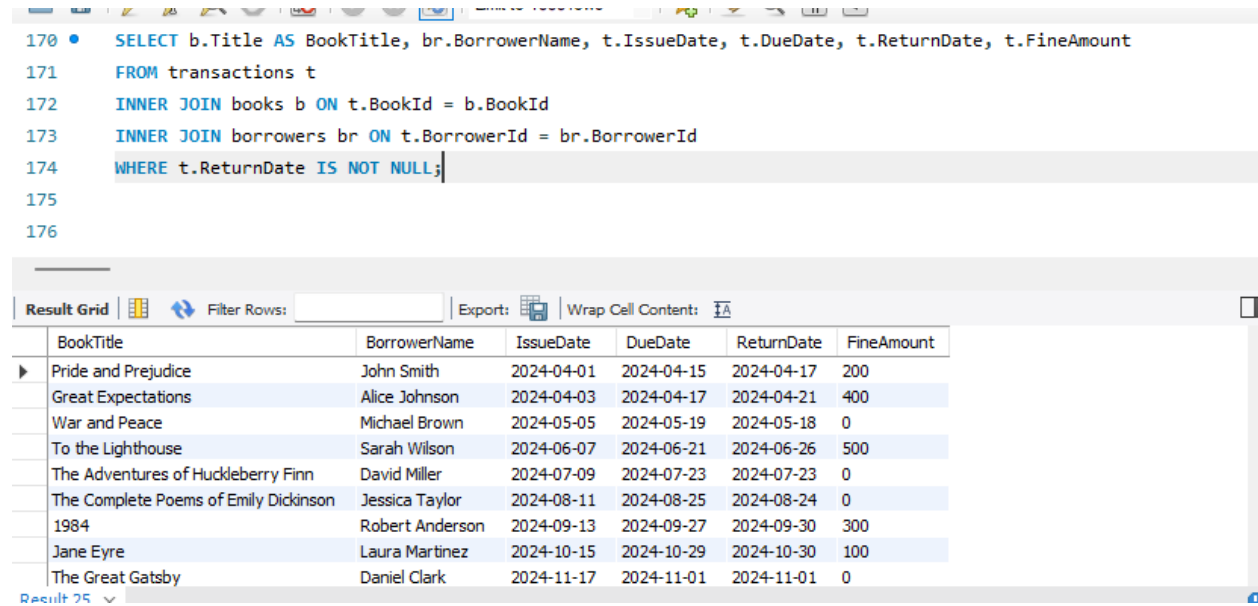
```
SELECT b.Title AS BookTitle, br.BorrowerName, t.IssueDate, t.DueDate, t.ReturnDate,  
t.FineAmount
```

```
FROM transactions t
```

```
INNER JOIN books b ON t.BookId = b.BookId
```

```
INNER JOIN borrowers br ON t.BorrowerId = br.BorrowerId
```

```
WHERE t.ReturnDate IS NOT NULL;
```



The screenshot shows a SQL query editor with the following query:

```
170 • SELECT b.Title AS BookTitle, br.BorrowerName, t.IssueDate, t.DueDate, t.ReturnDate, t.FineAmount  
171 FROM transactions t  
172 INNER JOIN books b ON t.BookId = b.BookId  
173 INNER JOIN borrowers br ON t.BorrowerId = br.BorrowerId  
174 WHERE t.ReturnDate IS NOT NULL;  
175  
176
```

Below the query editor is a 'Result Grid' showing the results of the query. The grid has columns for BookTitle, BorrowerName, IssueDate, DueDate, ReturnDate, and FineAmount. The results are as follows:

BookTitle	BorrowerName	IssueDate	DueDate	ReturnDate	FineAmount
Pride and Prejudice	John Smith	2024-04-01	2024-04-15	2024-04-17	200
Great Expectations	Alice Johnson	2024-04-03	2024-04-17	2024-04-21	400
War and Peace	Michael Brown	2024-05-05	2024-05-19	2024-05-18	0
To the Lighthouse	Sarah Wilson	2024-06-07	2024-06-21	2024-06-26	500
The Adventures of Huckleberry Finn	David Miller	2024-07-09	2024-07-23	2024-07-23	0
The Complete Poems of Emily Dickinson	Jessica Taylor	2024-08-11	2024-08-25	2024-08-24	0
1984	Robert Anderson	2024-09-13	2024-09-27	2024-09-30	300
Jane Eyre	Laura Martinez	2024-10-15	2024-10-29	2024-10-30	100
The Great Gatsby	Daniel Clark	2024-11-17	2024-11-01	2024-11-01	0

Explanation:

This query selects the title of books, borrower names, issue date, due date, return date, and fine amount for transactions where the return date is not null, by joining the transactions, books, and borrowers tables based on their respective IDs.

Query Number 14:

Query:

```
SELECT br.BorrowerName, COUNT(t.TransactionId)
AS TotalTransactions
FROM borrowers br
INNER JOIN transactions t ON br.BorrowerId = t.BorrowerId
GROUP BY br.BorrowerName;
```

170

```
171 • SELECT br.BorrowerName, COUNT(t.TransactionId) AS TotalTransactions
172 FROM borrowers br
173 INNER JOIN transactions t ON br.BorrowerId = t.BorrowerId
174 GROUP BY br.BorrowerName;
```

Result Grid			Filter Rows:	Export:	Wrap Cell Content:
	BorrowerName	TotalTransactions			
▶	John Smith	1			
	Alice Johnson	2			
	Michael Brown	2			
	Sarah Wilson	1			
	David Miller	1			
	Jessica Taylor	1			
	Robert Anderson	1			
	Laura Martinez	1			
	Daniel Clark	1			

Explanation:

This query counts the total number of transactions for each borrower, displaying their names alongside the corresponding count. It performs an inner join between the borrowers and transactions tables on the borrower ID, grouping the results by borrower name.

Query Number 15:

Query:

```
SELECT b.MembershipType, SUM(t.FineAmount)
AS TotalFine
FROM borrowers b
INNER JOIN transactions t ON b.BorrowerId = t.BorrowerId
GROUP BY b.MembershipType
HAVING TotalFine > 500;
```

```
177 • SELECT b.MembershipType, SUM(t.FineAmount) AS TotalFine
178 FROM borrowers b
179 INNER JOIN transactions t ON b.BorrowerId = t.BorrowerId
180 GROUP BY b.MembershipType
181 HAVING TotalFine > 500;
```

Result Grid		 Filter Rows: <input type="text"/>	Export: 	Wrap Cell Content: 
	MembershipType	TotalFine		
▶	Regular	2100		
	Premium	1600		

Explanation:

This query calculates the total fine amount for each membership type of borrowers, filtering the results to only include those with a total fine amount greater than 500. It involves an inner join between the borrowers and transactions tables on the borrower ID, grouping the results by membership type, and applying the filter using the HAVING clause.

Members:

M.Talha Qureshi (BSCS-23122)

Abdullah Hussain Yasim (BSCS-23008)

M.Ibrahim Butt (BSCS-23086)