

1.Data retrieval:

Get a list of members borrowed a specific authors book.

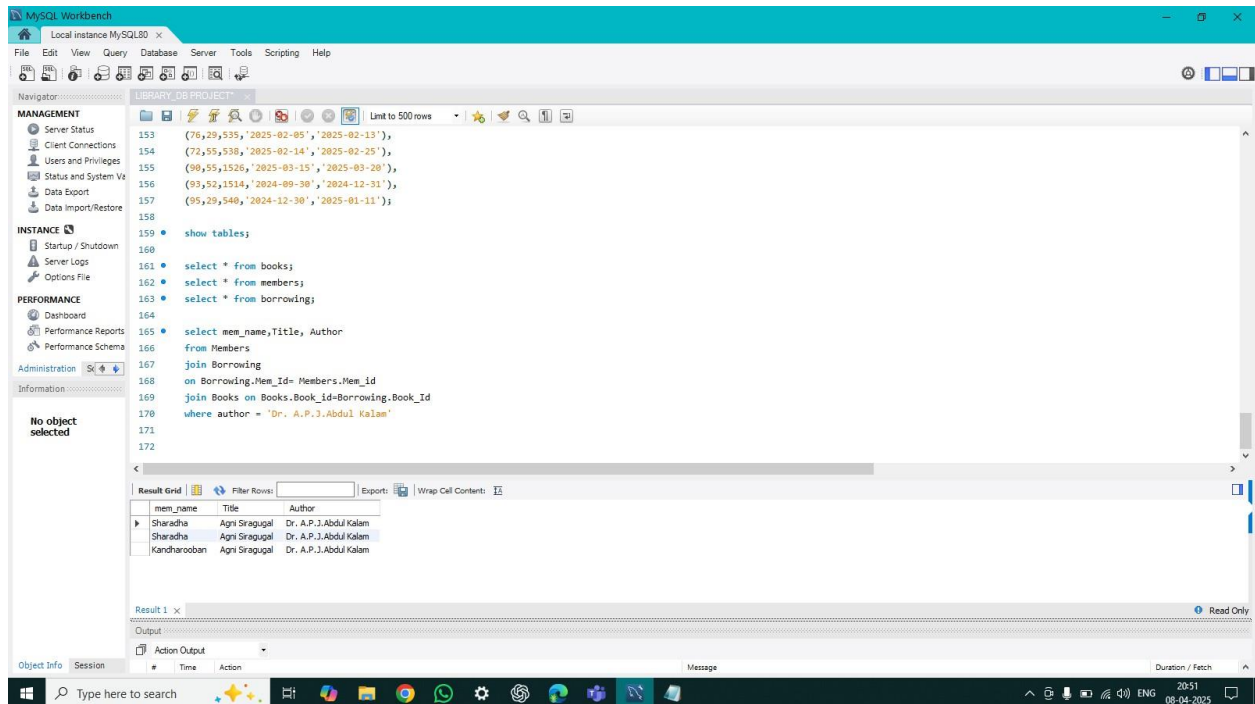
Select mem_name, Title, Author

from Members join Borrowing

On Borrowing.Mem_Id=Members.Mem_id

join Books on Books.Book_id=Borrowing.Book_Id

where author = 'Dr. A.P.J.Abdul Kalam';



The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
153 (76,29,535,'2025-02-05','2025-02-13'),
154 (72,55,538,'2025-02-14','2025-02-25'),
155 (98,55,1526,'2025-03-15','2025-03-28'),
156 (93,52,1514,'2024-09-30','2024-12-31'),
157 (95,29,540,'2024-12-30','2025-01-11');
158
159 show tables;
160
161 select * from books;
162 select * from members;
163 select * from borrowings;
164
165 select mem_name,Title, Author
166 from Members
167 join Borrowing
168 on Borrowing.Mem_Id= Members.Mem_id
169 join Books on Books.Book_id=Borrowing.Book_Id
170 where author = 'Dr. A.P.J.Abdul Kalam';
171
172
```

The Results grid shows the following data:

mem_name	Title	Author
Sharadha	Agni Sriragugal	Dr. A.P.J.Abdul Kalam
Sharadha	Agni Sriragugal	Dr. A.P.J.Abdul Kalam
Kandharooban	Agni Sriragugal	Dr. A.P.J.Abdul Kalam

The bottom status bar shows the system clock as 20:51 on 08-04-2025.

2. Query to update the available copies in specified id by 1.

Update books set Available_Copies=Available_copies+1,where book_id=577;

The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL code:

```
169 join Books on Books.Book_Id=Borrowing.Book_Id
170 where author = 'Dr. A.P.J.Abdul Kalam'
171
172 select Title,mem_name,Borrow_Date,Return_Date
173 from Borrowing
174 join Books
175 on Books.Book_Id= Borrowing.Book_Id
176 join Members on members.Mem_Id=Borrowing.mem_Id
177 where Title = 'SQL for experts';
178
179 update books
180 set Available_Copies= Available_copies + 1
181 where book_id=577;
```

The Results grid shows a table with columns: book_id, title, author, genre, published_year, available_copies, lang. The data includes books like 'Logics in Java Programming', 'Thannir', 'Harry potter I', 'Harry potter II', and 'Harry potter III'.

The Output tab shows the execution of the update query. The message indicates: "0 row(s) affected Rows matched: 0 Changed: 0 Warnings: 0".

The screenshot shows the MySQL Workbench interface with multiple queries executed. The query editor contains the following SQL code:

```
160
161 select * from books;
162 select * from members;
163 select * from borrowing;
164
165 select mem_name,Title, Author
166 from Members
167 join Borrowing
168 on Borrowing.Mem_Id= Members.Mem_Id
169 join Books on Books.Book_Id=Borrowing.Book_Id
170 where author = 'Dr. A.P.J.Abdul Kalam'
171
172 select Title,mem_name,Borrow_Date,Return_Date
173 from Borrowing
174 join Books
175 on Books.Book_Id= Borrowing.Book_Id
```

The Results grid shows a table with columns: book_id, title, author, genre, published_year, available_copies, lang. The data includes books like 'Java Programming', 'SQL,PL/SQL', 'Introducing MS-SQL Server...', 'Logics in Java Programming', and 'Thannir'.

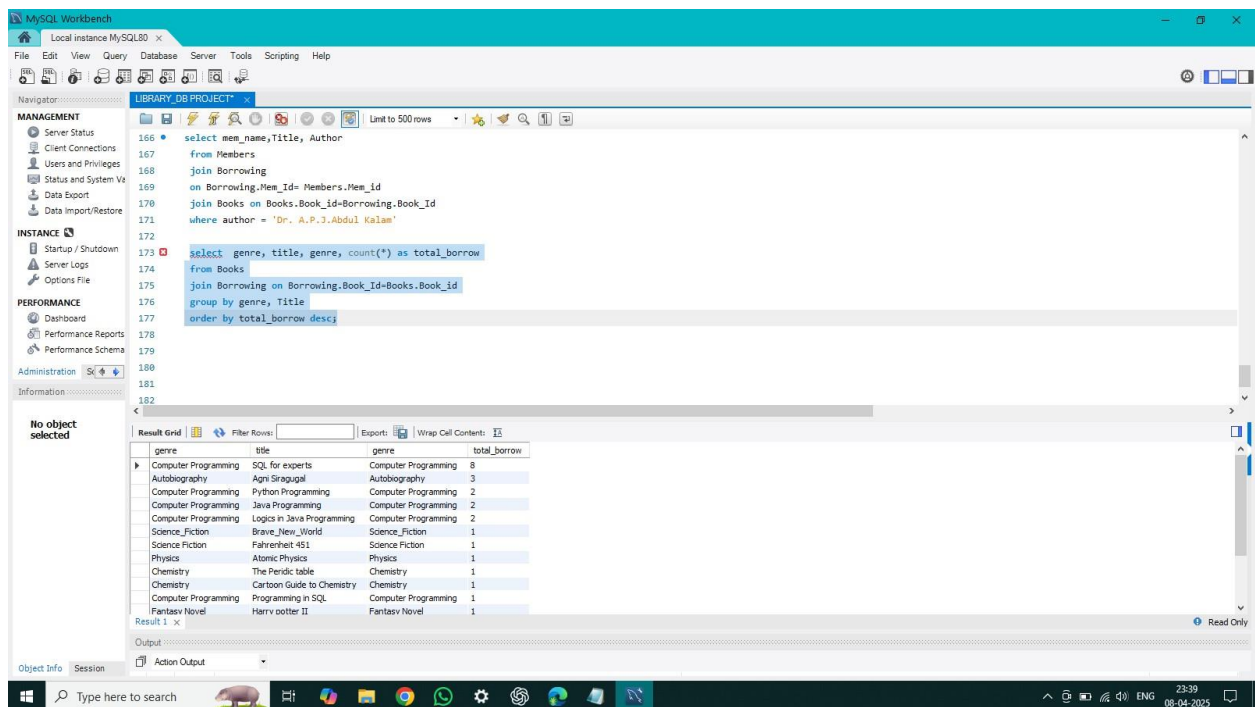
The Output tab shows the execution of multiple queries. The messages indicate the number of rows affected, rows matched, rows changed, and warnings for each query.

3. Find the most popular genre in the library. Display the genre with the highest total number of books borrowed.

Select genre, title, genre, count(*) as total_borrow

From Books join Borrowing on Borrowing.Book_Id=Books.Book_id

group by genre, Title order by total_borrow desc;



The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
select mem_name, Title, Author
from Members
join Borrowing
on Borrowing.Mem_Id= Members.Mem_Id
join Books on Books.Book_Id=Borrowing.Book_Id
where author = 'Dr. A.P.J. Abdul Kalam'

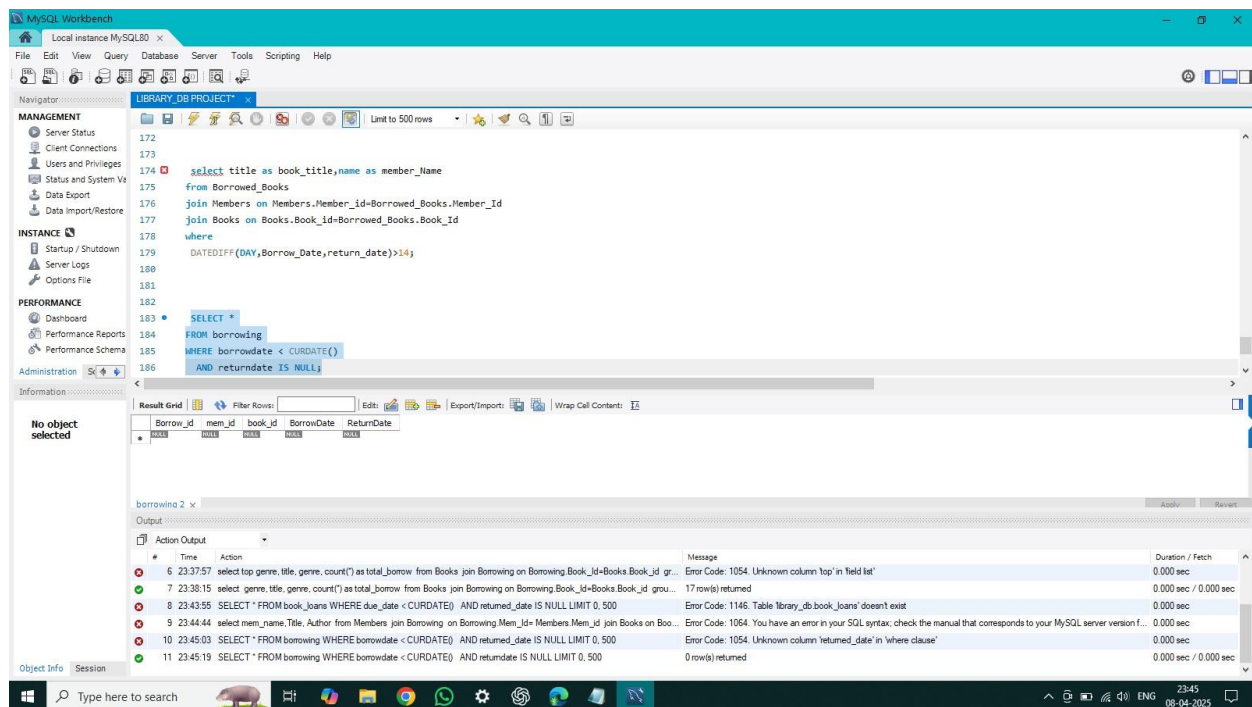
select genre, title, genre, count(*) as total_borrow
from Books
join Borrowing on Borrowing.Book_Id=Books.Book_Id
group by genre, Title
order by total_borrow desc;
```

The Results grid displays the following data:

genre	title	genre	total_borrow
Computer Programming	SQL for experts	Computer Programming	8
Autobiography	Agni Sriragopal	Autobiography	3
Computer Programming	Python Programming	Computer Programming	2
Computer Programming	Java Programming	Computer Programming	2
Computer Programming	Logics in Java Programming	Computer Programming	2
Science_Fiction	Brave_New_World	Science_Fiction	1
Science Fiction	Fahrenheit 451	Science Fiction	1
Physics	Atomic Physics	Physics	1
Chemistry	The Periodic table	Chemistry	1
Chemistry	Cartoon Guide to Chemistry	Chemistry	1
Computer Programming	Programming in SQL	Computer Programming	1
Fantasy Novel	Harry potter II	Fantasy Novel	1

4. Query to calculate the overdue books at current date.

SELECT* FROM borrowing WHERE borrowdate <CURDATE());



5. Query to find Overdue Books Based on Entered Dates:

```
SELECT DATE_ADD(borrowDate,INTERVAL14DAY)ASdue_date, CASE WHEN  
returnDate IS NULL THEN 'Overdue' WHEN returnDate > DATE_ADD  
(borrowDate,INTERVAL14DAY) THEN 'Overdue' ELSE 'OnTime' END AS status  
FROM borrowing AND returndate IS NULL;
```

The screenshot displays the MySQL Workbench interface. The SQL editor contains the following query:

```
SELECT *,  
DATE_ADD(borrowDate, INTERVAL 14 DAY) AS due_date,  
CASE WHEN returnDate IS NULL THEN 'Overdue'  
WHEN returnDate > DATE_ADD(borrowDate, INTERVAL 14 DAY) THEN 'Overdue'  
ELSE 'On Time'  
END AS status FROM borrowings;
```

The Results Grid shows the output of the query, listing 40 rows of data. The columns are: Borrow_id, mem_id, book_id, BorrowDate, ReturnDate, due_date, and status. The status column indicates whether a book is 'On Time' or 'Overdue' based on the due date and return date.

Borrow_id	mem_id	book_id	BorrowDate	ReturnDate	due_date	status
12	44	1526	2025-01-03	2025-01-15	2025-01-17	On Time
20	64	514	2025-01-12	2025-01-20	2025-01-26	On Time
22	50	574	2025-03-03	2025-03-11	2025-03-17	On Time
23	35	542	2024-12-01	2025-01-04	2024-12-15	Overdue
24	12	542	2024-08-12	2024-09-05	2024-08-26	Overdue
25	68	1526	2024-11-23	2024-12-17	2024-12-07	Overdue
31	41	1526	2024-12-12	2024-12-23	2024-12-26	On Time
32	41	1526	2024-12-03	2024-12-20	2024-12-17	Overdue
34	24	1526	2024-09-12	2024-10-23	2024-09-26	Overdue
36	33	543	2024-12-03	2024-12-31	2024-12-17	Overdue
37	54	530	2025-03-04	2025-03-11	2025-03-18	On Time
40	54	530	2025-03-10	2025-04-05	2025-03-24	Overdue
43	33	1524	2025-01-12	2025-01-21	2025-01-26	On Time
44	50	658	2024-12-24	2025-01-04	2025-01-07	On Time
46	38	577	2025-03-21	2025-04-02	2025-04-04	On Time
47	68	574	2025-01-04	2025-01-30	2025-01-18	Overdue
50	38	533	2025-03-12	2025-03-20	2025-03-26	On Time
52	35	1526	2025-02-04	2025-02-16	2025-02-18	On Time
54	50	530	2025-01-09	2025-01-19	2025-01-23	On Time
55	56	1522	2025-04-01	2025-04-13	2025-04-15	On Time
58	12	577	2025-01-13	2025-01-20	2025-01-27	On Time
66	64	1510	2025-03-03	2025-03-15	2025-03-17	On Time
72	55	538	2025-02-14	2025-02-25	2025-02-28	On Time
76	29	535	2025-02-05	2025-02-13	2025-02-19	On Time
86	38	1511	2024-12-24	2025-01-03	2025-01-07	On Time
90	65	1526	2025-01-15	2025-01-30	2025-01-29	On Time

6. Use of DDL COMMAND:

Alter table borrowing with fine.

ALTER TABLE borrowing

ADD COLUMN fine DECIMAL(10,2) DEFAULT 0;

The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL code:

```
175 FROM borrowing
176 WHERE RETURNdate < CURDATE()
177 AND returndate IS NULL;
178
179 • SELECT *,
180     DATE_ADD(borrowDate, INTERVAL 14 DAY) AS due_date,
181     CASE WHEN returnDate IS NULL THEN 'Overdue'
182         WHEN returnDate > DATE_ADD(borrowDate, INTERVAL 14 DAY) THEN 'Overdue'
183         ELSE 'On Time'
184     END AS status FROM borrowings;
185
186 • ALTER TABLE borrowing
187     ADD COLUMN fine DECIMAL(10,2) DEFAULT 0;
188
189 • SELECT * FROM BORROWING;
```

The Results Grid shows the following data:

Borrow_id	mem_id	book_id	BorrowDate	ReturnDate	fine
12	44	1526	2025-01-03	2025-01-15	0.00
20	64	514	2025-01-12	2025-01-20	0.00
22	50	574	2025-03-03	2025-03-11	0.00
23	35	542	2024-12-01	2025-01-04	0.00
24	12	542	2024-08-12	2024-09-05	0.00
25	68	1526	2024-11-23	2024-12-17	0.00
31	41	1526	2024-12-12	2024-12-23	0.00
32	41	1526	2024-12-03	2024-12-20	0.00
34	24	1526	2024-09-12	2024-10-23	0.00

The Output pane shows the following messages:

```
13 23:55:31 UPDATE borrowing SET status = 'Overdue' WHERE due_date < CURDATE() AND return_date IS NULL
14 23:56:06 UPDATE borrowing SET status = 'Overdue' WHERE RETURNDATE < CURDATE() AND returndate IS NULL
```

The Message pane shows the following error messages:

```
Error Code: 1054. Unknown column 'due_date' in 'where clause'
Error Code: 1054. Unknown column 'status' in 'field list'
```

7. Find members who return books quickly. Calculate the average duration it takes for each member to return borrowed books and then identify members whose average return time is less than 14 days.

```
SELECT borrow_id, book_id, mem_id, DATE_ADD(borrowDate,  
INTERVAL 14 DAY) AS due_date, CASE WHEN return Date ISNULL THEN  
'Overdue' WHEN returnDate>DATE_ADD(borrowDate,INTERVAL14DAY)  
THEN 'Overdue' ELSE 'OnTime' END AS status FROM borrowing
```

```
WHERE returnDate IS NULL OR returnDate>DATE_ADD(borrowDate,  
INTERVAL 14 DAY);
```

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following queries:

```
182 WHEN returnDate > DATE_ADD(borrowDate, INTERVAL 14 DAY) THEN 'Overdue'
183 ELSE 'On Time'
184 END AS status FROM borrowing;
185
186 ALTER TABLE borrowing
187 ADD COLUMN fine DECIMAL(10,2) DEFAULT 0;
188
189 SELECT * FROM BORROWING;
190
191
192 SELECT mem_id,
193 AVG(DATEDIFF(returndate, borrowdate)) AS avg_return_days
194 FROM borrowing
195 WHERE returndate IS NOT NULL
196 GROUP BY mem_id
197 HAVING avg_return_days > 14;
```

The Results tab shows the output of the last query:

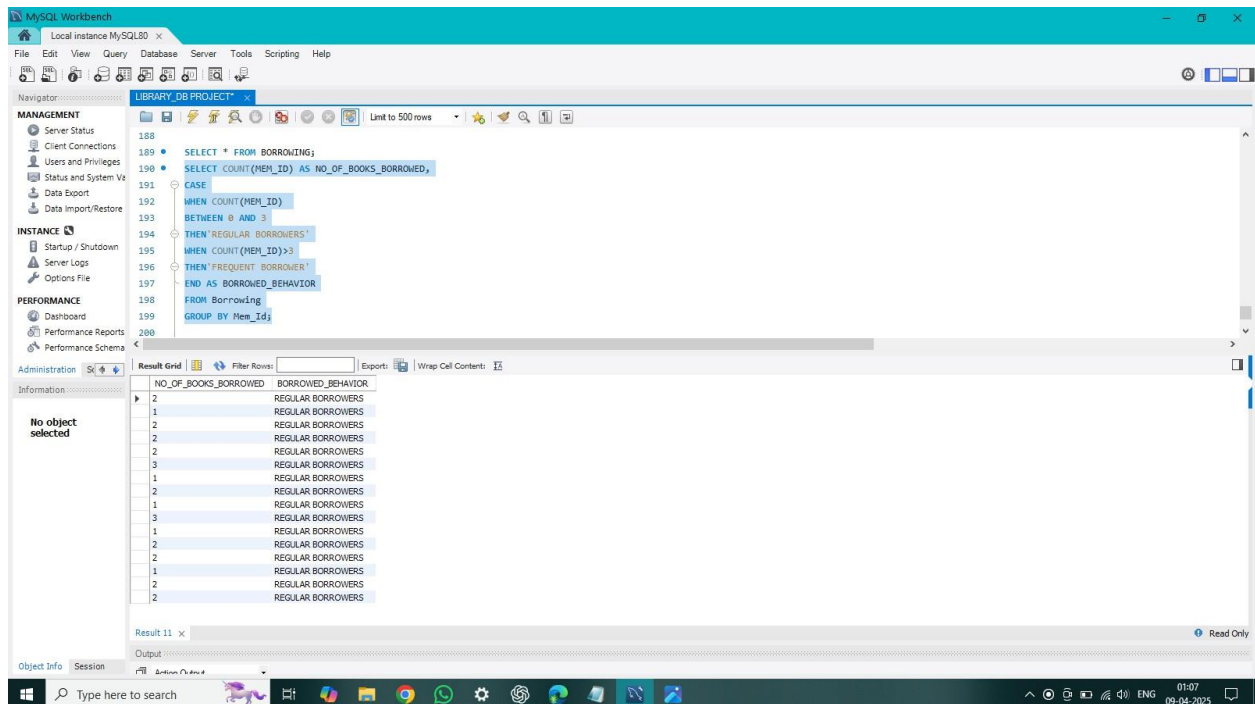
mem_id	avg_return_days
12	15.5000
24	41.0000
33	18.5000
35	23.0000
52	92.0000
54	16.5000
68	25.0000

The Output tab shows the execution log:

```
32 00:16:12 SELECT mem_id, AVG(DATEDIFF(returndate, borrowdate)) AS avg_return_days FROM borrowing WHERE returndate ... 15 row(s) returned 0.000 sec / 0.000 sec
33 00:17:20 SELECT mem_id, AVG(DATEDIFF(returndate, borrowdate)) AS avg_return_days FROM borrowing WHERE returndate ... 7 row(s) returned 0.000 sec / 0.000 sec
```


8. Calculate the average duration in months for which books are borrowed. Display the book title and the average duration in months.

```
SELECT COUNT(MEM_ID) AS NO_OF_BOOKS_BORROWED,  
CASE WHEN COUNT(MEM_ID)  
BETWEEN 0 AND 3  
THEN 'REGULAR BORROWERS'  
WHEN COUNT(MEM_ID) > 3  
THEN 'FREQUENT BORROWER'  
END AS BORROWED_BEHAVIOR  
FROM Borrowing  
GROUP BY Mem_Id;
```



The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
SELECT * FROM BORROWING;  
SELECT COUNT(MEM_ID) AS NO_OF_BOOKS_BORROWED,  
CASE  
WHEN COUNT(MEM_ID)  
BETWEEN 0 AND 3  
THEN 'REGULAR BORROWERS'  
WHEN COUNT(MEM_ID) > 3  
THEN 'FREQUENT BORROWER'  
END AS BORROWED_BEHAVIOR  
FROM Borrowing  
GROUP BY Mem_Id;
```

The Results Grid shows the output of the query:

NO_OF_BOOKS_BORROWED	BORROWED_BEHAVIOR
2	REGULAR BORROWERS
1	REGULAR BORROWERS
2	REGULAR BORROWERS
2	REGULAR BORROWERS
2	REGULAR BORROWERS
3	REGULAR BORROWERS
1	REGULAR BORROWERS
2	REGULAR BORROWERS
1	REGULAR BORROWERS
3	REGULAR BORROWERS
1	REGULAR BORROWERS
2	REGULAR BORROWERS
2	REGULAR BORROWERS
1	REGULAR BORROWERS
2	REGULAR BORROWERS
2	REGULAR BORROWERS

9. Calculate the late fees for each overdue book by multiples of 50.

SELECT Borrow_id, mem_id, book_id, BorrowDate, ReturnDate,

DATEDIFF (IF(ReturnDate IS NULL, CURDATE(),
ReturnDate), DATE_ADD (BorrowDate, INTERVAL 14 DAY)) AS
days_overdue, CASE WHEN DATEDIFF (IF(ReturnDate IS NULL,
CURDATE(), ReturnDate), DATE_ADD (BorrowDate, INTERVAL
14 DAY)) > 0 THEN DATEDIFF (IF(ReturnDate IS NULL,
CURDATE(), ReturnDate), DATE_ADD (BorrowDate, INTERVAL
14 DAY)) * 50 ELSE 0 END AS fine_amount FROM Borrowing;

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

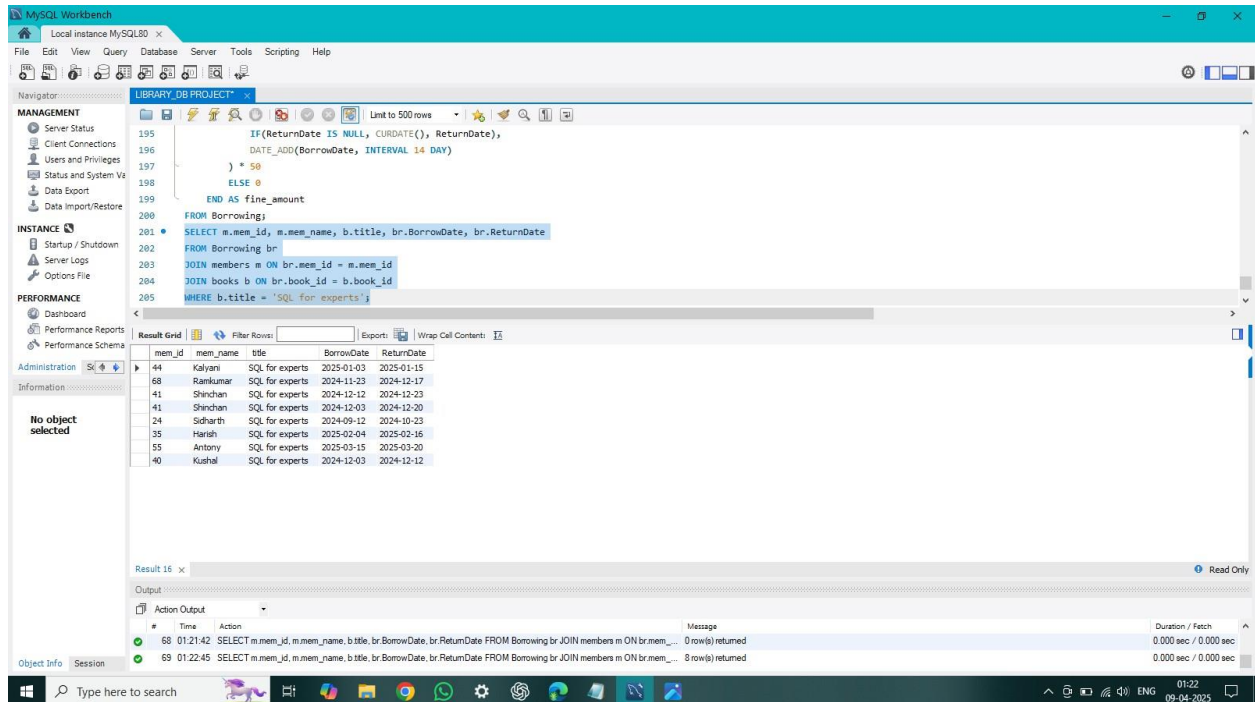
```
SELECT * FROM BORROWING;  
  
SELECT Borrow_id, mem_id, book_id, BorrowDate, ReturnDate,  
DATEDIFF (IF(ReturnDate IS NULL, CURDATE(), ReturnDate), DATE_ADD (BorrowDate, INTERVAL 14 DAY) ) AS days_overdue,  
CASE WHEN DATEDIFF (IF(ReturnDate IS NULL, CURDATE(), ReturnDate), DATE_ADD (BorrowDate, INTERVAL  
14 DAY)) > 0 THEN DATEDIFF (IF(ReturnDate IS NULL,  
CURDATE(), ReturnDate), DATE_ADD (BorrowDate, INTERVAL  
14 DAY)) * 50 ELSE 0  
END AS fine_amount  
FROM Borrowing;
```

The Results grid shows the following data:

Borrow_id	mem_id	book_id	BorrowDate	ReturnDate	days_overdue	fine_amount
12	44	1526	2025-01-03	2025-01-15	-2	0
20	64	514	2025-01-12	2025-01-20	-6	0
22	50	574	2025-03-03	2025-03-11	-6	0
23	35	542	2024-12-01	2025-01-04	20	1000
24	12	542	2024-08-12	2024-09-05	10	500
25	68	1536	2024-11-23	2024-12-17	10	500
31	41	1526	2024-12-12	2024-12-23	-3	0
32	41	1526	2024-12-03	2024-12-20	3	150
34	24	1526	2024-09-12	2024-10-23	27	1350
36	33	543	2024-12-03	2024-12-31	14	700
37	54	530	2025-03-04	2025-03-11	-7	0
40	54	530	2025-03-10	2025-04-05	12	600
43	33	1524	2025-01-12	2025-01-21	-5	0
44	50	698	2024-12-24	2025-01-04	-3	0
46	38	577	2025-03-21	2025-04-02	-2	0
47	68	574	2025-01-04	2025-01-30	12	600
50	38	533	2025-03-12	2025-03-20	-6	0
51	15	1536	2025-03-04	2025-03-16	-7	0

10. Find members who borrowed a specific book by book title

```
SELECT m.mem_id, m.mem_name, b.title, br.BorrowDate, br.ReturnDate  
FROM Borrowing br JOIN members m ON br.mem_id=m.mem_id JOIN books  
b ON br.book_id = b.book_id WHERE b.title = 'SQL for experts';
```



The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
IF(ReturnDate IS NULL, CURDATE(), ReturnDate),  
DATE_ADD(BorrowDate, INTERVAL 14 DAY)  
) * 50  
ELSE 0  
END AS fine_amount  
FROM Borrowing;  
201 SELECT m.mem_id, m.mem_name, b.title, br.BorrowDate, br.ReturnDate  
202 FROM Borrowing br  
203 JOIN members m ON br.mem_id = m.mem_id  
204 JOIN books b ON br.book_id = b.book_id  
205 WHERE b.title = 'SQL for experts';
```

The Results tab displays the following data:

mem_id	mem_name	title	BorrowDate	ReturnDate
44	Kalyani	SQL for experts	2025-01-03	2025-01-15
68	Rankumar	SQL for experts	2024-11-23	2024-12-17
41	Shinchuan	SQL for experts	2024-12-12	2024-12-23
41	Shinchuan	SQL for experts	2024-12-03	2024-12-20
24	Sidharth	SQL for experts	2024-09-12	2024-10-23
35	Harish	SQL for experts	2025-02-04	2025-02-16
55	Antony	SQL for experts	2025-03-15	2025-03-20
40	Kushal	SQL for experts	2024-12-03	2024-12-12

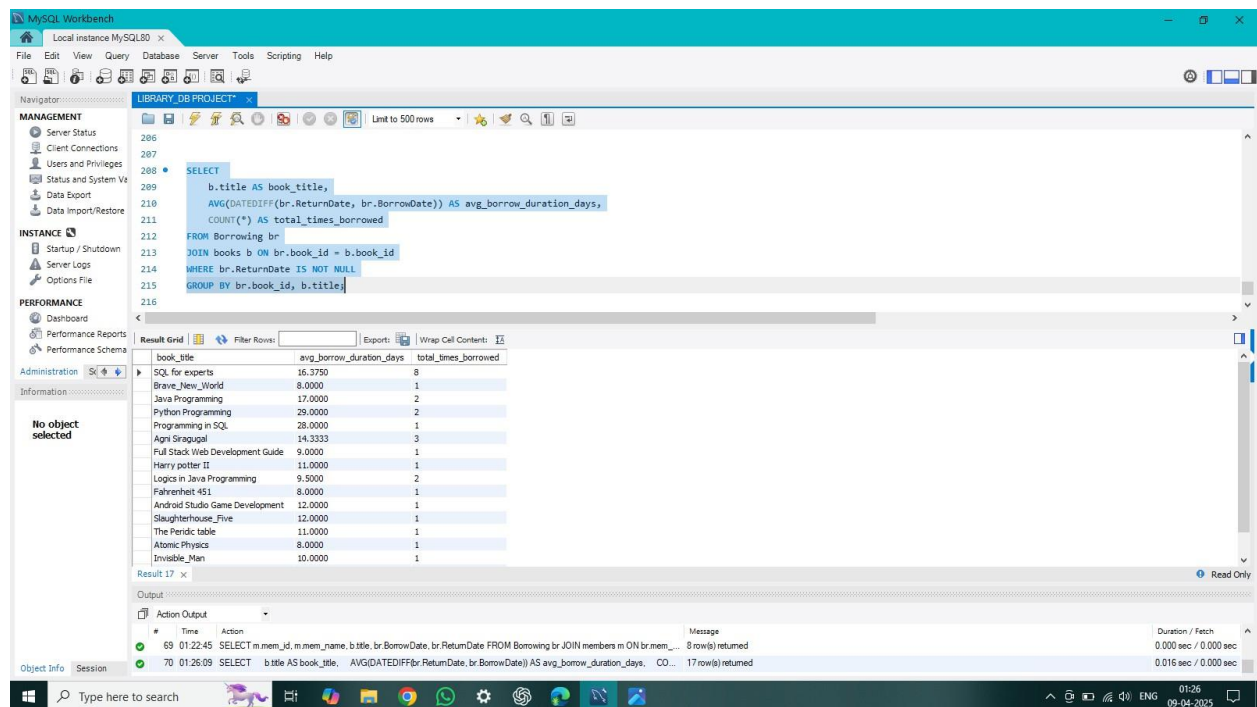
The Output tab shows the execution of the query:

```
68 01:21:42 SELECT m.mem_id, m.mem_name, b.title, br.BorrowDate, br.ReturnDate FROM Borrowing br JOIN members m ON br.mem_id = m.mem_id JOIN books b ON br.book_id = b.book_id WHERE b.title = 'SQL for experts'; 0 row(s) returned 0.000 sec / 0.000 sec  
69 01:22:45 SELECT m.mem_id, m.mem_name, b.title, br.BorrowDate, br.ReturnDate FROM Borrowing br JOIN members m ON br.mem_id = m.mem_id JOIN books b ON br.book_id = b.book_id WHERE b.title = 'SQL for experts'; 8 row(s) returned 0.000 sec / 0.000 sec
```

11. Calculate the average duration a book is borrowed by members. Display the book title, the average duration in days and the number of times it has been borrowed.

```
SELECT b.title AS book_title,AVG (DATEDIFF (br.ReturnDate, br.BorrowDate))
AS avg_borrow_duration_days, COUNT(*)AS total_times_borrowed

FROM Borrowing br JOIN books b ON br.book_id=b.book_id
WHERE br.ReturnDate IS NOT NULL GROUP BY br.book_id, b.title;
```



The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
SELECT
    b.title AS book_title,
    AVG(DATEDIFF(br.ReturnDate, br.BorrowDate)) AS avg_borrow_duration_days,
    COUNT(*) AS total_times_borrowed
FROM Borrowing br
JOIN books b ON br.book_id = b.book_id
WHERE br.ReturnDate IS NOT NULL
GROUP BY br.book_id, b.title;
```

The Results Grid displays the following data:

book_title	avg_borrow_duration_days	total_times_borrowed
SQL for experts	16.3750	8
Brave New World	8.0000	1
Java Programming	17.0000	2
Python Programming	29.0000	2
Programming in SQL	28.0000	1
Agri Siragugal	14.3333	3
Full Stack Web Development Guide	9.0000	1
Harry potter II	11.0000	1
Logics in Java Programming	9.5000	2
Fahrenheit 451	8.0000	1
Android Studio Game Development	12.0000	1
Slaughterhouse_Five	12.0000	1
The Peridic table	11.0000	1
Atomic Physics	8.0000	1
Invisible Man	10.0000	1

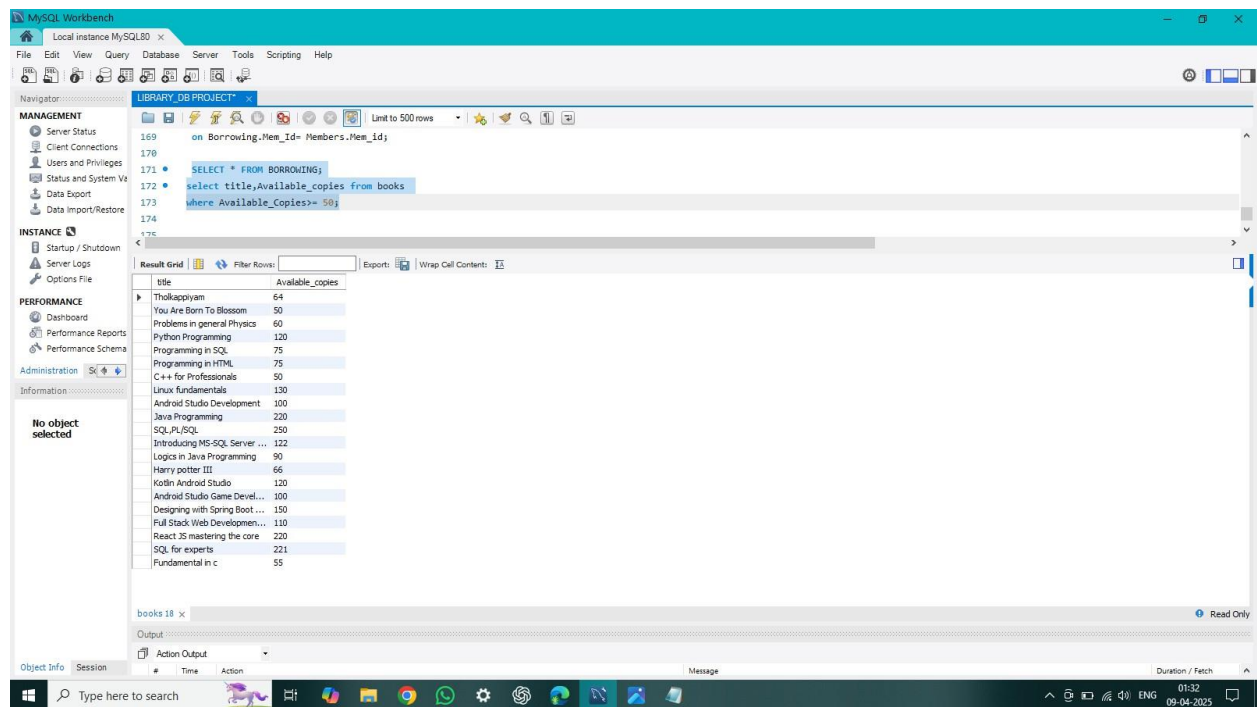
The Output pane shows the execution of the query, indicating that 17 rows were returned.

12. Calculate the average duration a book is borrowed by members. Display the book title, the average duration in days, and the number of times it has been borrowed.

SELECT * FROM BORROWING;

select title, Available_copies from books

where Available_Copies >= 50;

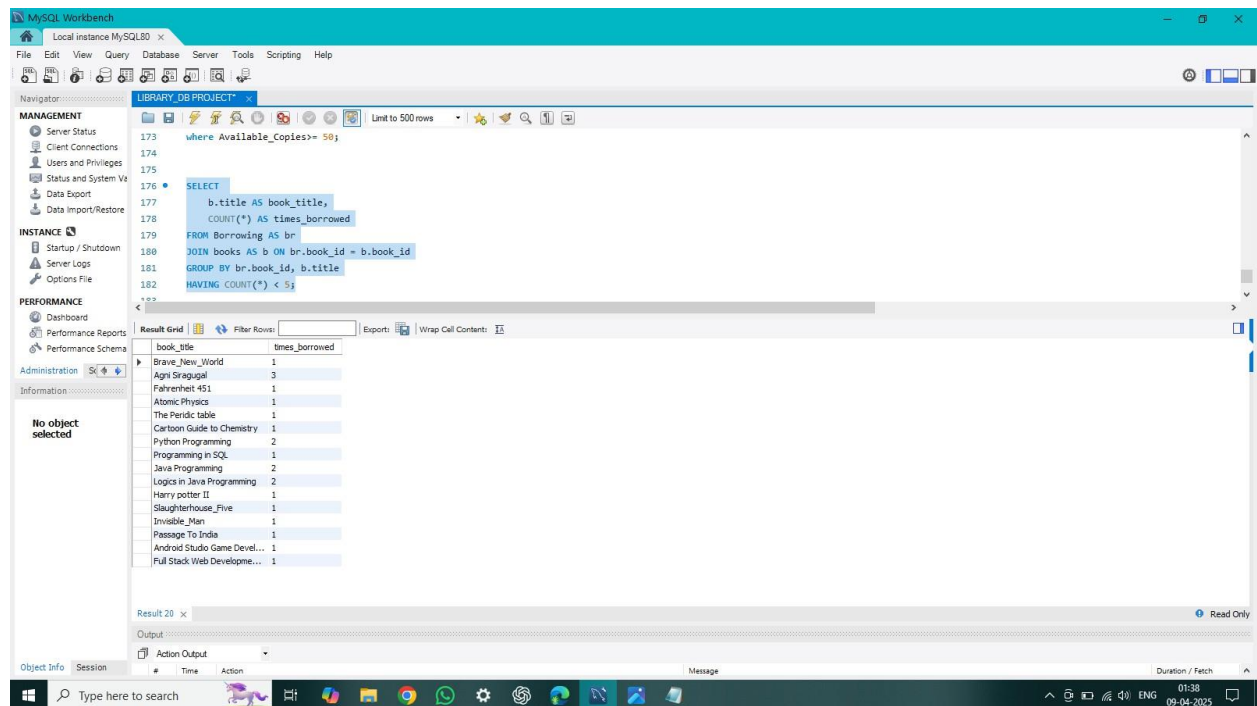


13. Filter to show only books borrowed less than 5 times.

```
SELECT b.title AS book_title, COUNT(*) AS times_borrowed
```

```
FROM Borrowing AS br JOIN books AS b ON br.book_id=b.book_id GROUP BY
```

```
br.book_id, b.title HAVING COUNT(*)<5;
```



The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
SELECT  
  b.title AS book_title,  
  COUNT(*) AS times_borrowed  
FROM Borrowing AS br  
JOIN books AS b ON br.book_id = b.book_id  
GROUP BY br.book_id, b.title  
HAVING COUNT(*) < 5;
```

The Results Grid displays the following data:

book_title	times_borrowed
Brave New World	1
Agni Sragugal	3
Fahrenheit 451	1
Atomic Physics	1
The Periodic table	1
Cartoon Guide to Chemistry	1
Python Programming	2
Programming in SQL	1
Java Programming	2
Logics in Java Programming	2
Harry potter II	1
Slaughterhouse_Five	1
Invisible_Man	1
Passage To India	1
Android Studio Game Devel...	1
Full Stack Web Developme...	1

14. Find the most popular genre in the library. Display the genre with the highest total number of books borrowed.

```
SELECT b.genre, COUNT(*) AS total_borrowed FROM  
borrowing br JOIN books b ON br.book_id = b.book_id  
GROUP BY b.genre ORDER BY total_borrowed  
DESC LIMIT 1;
```

The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL code:

```
249 FROM borrowing  
250 group by book_id  
251 ORDER BY total_borrowed DESC  
252 LIMIT 1;  
253 • SELECT b.genre, COUNT(*) AS total_borrowed  
254 FROM borrowing br  
255 JOIN books b ON br.book_id = b.book_id  
256 GROUP BY b.genre  
257 ORDER BY total_borrowed DESC  
258 LIMIT 1;
```

The left sidebar shows the 'LIBRARY_DB PROJECT' with a tree view containing 'MANAGEMENT', 'INSTANCE', and 'PERFORMANCE' sections. The 'Result Grid' at the bottom shows the execution results of the query. The results are as follows:

#	Time	Action	Message	Duration / Fetch
4	18:32:05	SELECT book_id, COUNT(*) AS total_borrowed FROM borrowing GROUP BY genre ORDER BY total_borrowed DESC LIMIT 1	Error Code: 1054. Unknown column 'genre' in 'group statement'	0.000 sec
5	18:32:20	SELECT book_id, COUNT(*) AS total_borrowed FROM borrowing ORDER BY total_borrowed DESC LIMIT 1	Error Code: 1140. In aggregated query without GROUP BY, expression #1 of SELECT list contains nonaggregated column 1...	0.016 sec
6	18:33:02	SELECT book_id, COUNT(*) AS total_borrowed FROM borrowing group by book_id ORDER BY total_borrowed DESC LIMIT 1	1 row(s) returned	0.000 sec / 0.000 sec
7	18:34:33	SELECT b.genre, COUNT(*) AS total_borrowed FROM borrowing br JOIN books b ON br.book_id = b.book_id GROUP BY b.genre ORDER BY total_borrowed DESC LIMIT 1	1 row(s) returned	0.031 sec / 0.000 sec

15. Find the top 5 members who have borrowed the most books. Display their names and the number of books they have borrowed.

```
SELECT m.mem_name, COUNT(*) AS total_borrowed
FROM borrowing br
JOIN members m ON br.mem_id = m.mem_id GROUP
BY m.mem_name
ORDER BY total_borrowed DESC
LIMIT 5;
```

The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL code:

```
255 JOIN books b ON br.book_id = b.book_id
256 GROUP BY b.genre
257 ORDER BY total_borrowed DESC
258 LIMIT 1;
259 SELECT m.mem_name, COUNT(*) AS total_borrowed
260 FROM borrowing br
261 JOIN members m ON br.mem_id = m.mem_id
262 GROUP BY m.mem_name
263 ORDER BY total_borrowed DESC
264 LIMIT 5;
```

The Results grid shows the following data:

mem_name	total_borrowed
Kandharoban	3
Krishnaraj	3
Mohammed	2
Gokulsinh	2
Manoj	2

The Action Output pane shows the following messages:

#	Time	Action	Message	Duration / Fetch
5	18:32:20	SELECT book_id, COUNT(*) AS total_borrowed FROM borrowing ORDER BY total_borrowed DESC LIMIT 1	Error Code: 1140. In aggregated query without GROUP BY, expression #1 of SELECT list contains nonaggregated column 1...	0.016 sec
6	18:33:02	SELECT book_id, COUNT(*) AS total_borrowed FROM borrowing group by book_id ORDER BY total_borrowed DESC LIMIT 1	1 row(s) returned	0.000 sec / 0.000 sec
7	18:34:33	SELECT b.genre, COUNT(*) AS total_borrowed FROM borrowing br JOIN books b ON br.book_id = b.book_id GROUP BY ...	1 row(s) returned	0.031 sec / 0.000 sec
8	18:39:08	SELECT m.mem_name, COUNT(*) AS total_borrowed FROM borrowing br JOIN members m ON br.mem_id = m.mem_id GR...	5 row(s) returned	0.016 sec / 0.000 sec

16. To calculate over due borrowings. Write a query that compares the Return Date with the due date.

```
SELECT b.Borrow_id, b.mem_id, b.book_id, b.BorrowDate, b.ReturnDate, CASE WHEN  
b.ReturnDate IS NULL AND CURRENT_DATE > b.BorrowDate +  
INTERVAL '14' DAY THEN 'Overdue'
```

```
WHEN b.ReturnDate IS NOT NULL AND b.ReturnDate > b.BorrowDate  
+ INTERVAL '14' DAY THEN 'Returned Late'
```

```
ELSE 'On Time' END
```

```
AS Status FROM
```

```
borrowing b;
```

The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL code:

```
265 (152, 42, 659, '2025-03-24', CURRENT_DATE),  
266 (33, 56, 660, '2025-03-30', CURRENT_DATE),  
267 (49, 41, 661, '2025-03-15', CURRENT_DATE),  
268 (59, 64, 662, '2025-04-03', CURRENT_DATE),  
269 • select * from borrowing;  
270 • SELECT  
271 b.Borrow_id, b.mem_id, b.book_id, b.BorrowDate, b.ReturnDate,  
272 CASE WHEN b.ReturnDate IS NULL AND CURRENT_DATE > b.BorrowDate + INTERVAL '14' DAY THEN 'Overdue'  
273 WHEN b.ReturnDate IS NOT NULL AND b.ReturnDate > b.BorrowDate + INTERVAL '14' DAY THEN 'Returned Late'  
274 ELSE 'On Time'  
275 END AS Status  
276 FROM borrowing b;  
277
```

The Results tab shows the following data:

Borrow_id	mem_id	book_id	BorrowDate	ReturnDate	Status
12	44	1526	2025-01-03	2025-01-15	On Time
20	64	514	2025-01-12	2025-01-20	On Time
22	50	574	2025-03-03	2025-03-11	On Time
23	35	542	2024-12-01	2025-01-04	Returned Late
24	12	542	2024-08-12	2024-09-05	Returned Late
25	68	1526	2024-11-23	2024-12-17	Returned Late
31	41	1526	2024-12-12	2024-12-23	On Time
32	41	1526	2024-12-03	2024-12-20	Returned Late
33	56	660	2025-03-30	2025-04-09	On Time
34	24	1526	2024-09-12	2024-10-23	Returned Late
36	33	543	2024-12-03	2024-12-31	Returned Late
37	54	530	2025-03-04	2025-03-11	On Time
40	54	530	2025-03-10	2025-04-05	Returned Late
41	40	1526	2025-03-21	2025-04-09	Returned Late
43	33	1524	2025-01-12	2025-01-21	On Time
44	50	658	2024-12-24	2025-01-04	On Time
46	38	577	2025-03-21	2025-04-02	On Time

17. Query to add status in borrowing table as default ALTER TABLE Borrowing.

```
ADD COLUMN status VARCHAR(20)DEFAULT 'OnTime';
UPDATE   Borrowing
SET status =CASE
    WHEN ReturnDate IS NULLANDCURRENT_DATE>BorrowDate+INTERVAL'14'
DAY THEN 'Overdue'
    WHEN ReturnDate ISNOT NULL AND ReturnDate > BorrowDate + INTERVAL'14'
DAY THEN 'Returned Late'
    ELSE'OnTime'      END;
```

The screenshot shows the MySQL Workbench interface. The query editor displays the following SQL code:

```
269 select * from borrowing;
270
271 SELECT b.Borrow_id, b.mem_id, b.book_id, b.BorrowDate, b.ReturnDate,
272 CASE
273     WHEN b.ReturnDate IS NULL AND CURRENT_DATE > b.BorrowDate + INTERVAL '14' DAY THEN 'Overdue'
274     WHEN b.ReturnDate IS NOT NULL AND b.ReturnDate > b.BorrowDate + INTERVAL '14' DAY THEN 'Returned Late'
275     ELSE 'On Time'
276 END AS Status FROM Borrowing b;
277
278 ALTER TABLE Borrowing
279 ADD COLUMN status VARCHAR(20) DEFAULT 'On Time';
280
281 UPDATE Borrowing
282 SET status = CASE
283     WHEN ReturnDate IS NULL AND CURRENT_DATE > BorrowDate + INTERVAL '14' DAY THEN 'Overdue'
284     WHEN ReturnDate IS NOT NULL AND ReturnDate > BorrowDate + INTERVAL '14' DAY THEN 'Returned Late'
285     ELSE 'On Time' END;
```

The result grid shows the following data:

Borrow_id	mem_id	book_id	BorrowDate	ReturnDate	fine	status
12	44	1526	2025-01-03	2025-01-15	0.00	On Time
20	64	514	2025-01-12	2025-01-20	0.00	On Time
22	50	574	2025-03-03	2025-03-11	0.00	On Time
23	35	542	2024-12-01	2025-01-04	0.00	Returned Late
24	12	542	2024-08-12	2024-09-05	0.00	Returned Late
25	68	1526	2024-11-23	2024-12-17	0.00	Returned Late
31	41	1526	2024-12-12	2024-12-23	0.00	On Time
32	41	1526	2024-12-03	2024-12-20	0.00	Returned Late
33	56	660	2025-03-30	2025-04-09	0.00	On Time
34	24	1526	2024-09-12	2024-10-23	0.00	Returned Late
36	33	543	2024-12-03	2024-12-31	0.00	Returned Late
37	54	530	2025-03-04	2025-03-11	0.00	On Time
40	54	530	2025-03-10	2025-04-05	0.00	Returned Late
41	40	1526	2025-03-21	2025-04-09	0.00	Returned Late
43	33	1524	2025-01-12	2025-01-21	0.00	On Time
44	50	658	2024-12-24	2025-01-04	0.00	On Time
46	38	577	2025-03-21	2025-04-02	0.00	On Time

18. Calculate and add fine in borrowing table.

```
UPDATE Borrowing
SET fine= CASE
WHEN Return Date IS NOT NULL AND ReturnDate>BorrowDate+INTERVAL14 DAY
THEN
DATEDIFF(ReturnDate,BorrowDate+INTERVAL14DAY)*50
WHEN ReturnDate ISNULL AND CURRENT_DATE>BorrowDate+ INTERVAL 14
DAY THEN
DATEDIFF(CURRENT_DATE,BorrowDate+INTERVAL14DAY)*50 ELSE0
END;
```

The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL code:

```
268 (59, 64, 662, '2025-04-03', CURRENT_DATE);
269 • select * from borrowing;
270 • SELECT b.Borrow_id, b.mem_id, b.book_id, b.BorrowDate, b.ReturnDate,
271 • CASE WHEN b.ReturnDate IS NULL AND CURRENT_DATE > b.BorrowDate + INTERVAL '14' DAY THEN 'Overdue'
272 • WHEN b.ReturnDate IS NOT NULL AND b.ReturnDate > b.BorrowDate + INTERVAL '14' DAY THEN 'Returned Late'
273 • ELSE 'On Time' END AS Status FROM Borrowing b;
274 • ALTER TABLE Borrowing
275 • ADD COLUMN status VARCHAR(20) DEFAULT 'On Time';
276 • UPDATE Borrowing
277 • SET status = CASE
278 • WHEN ReturnDate IS NULL AND CURRENT_DATE > BorrowDate + INTERVAL '14' DAY THEN 'Overdue'
279 • WHEN ReturnDate IS NOT NULL AND ReturnDate > BorrowDate + INTERVAL '14' DAY THEN 'Returned Late'
280 • ELSE 'On Time' END;
```

The result grid shows the following data:

Borrow_id	mem_id	book_id	BorrowDate	ReturnDate	fine	status
22	50	574	2025-03-03	2025-03-11	0.00	On Time
23	35	542	2024-12-01	2025-01-04	1000.00	Returned Late
24	12	542	2024-08-12	2024-09-05	500.00	Returned Late
25	68	1526	2024-11-23	2024-12-17	500.00	Returned Late
31	41	1526	2024-12-12	2024-12-23	0.00	On Time
32	41	1526	2024-12-03	2024-12-20	150.00	Returned Late
33	56	660	2025-03-30	2025-04-09	0.00	On Time
34	24	1526	2024-09-12	2024-10-23	1350.00	Returned Late
36	33	543	2024-12-03	2024-12-31	700.00	Returned Late
37	54	530	2025-03-04	2025-03-11	0.00	On Time
40	54	530	2025-03-10	2025-04-05	600.00	Returned Late
41	40	1526	2025-03-21	2025-04-05	250.00	Returned Late
43	33	1524	2025-01-12	2025-01-21	0.00	On Time
44	50	658	2024-12-24	2025-01-04	0.00	On Time
46	38	577	2025-03-21	2025-04-02	0.00	On Time
47	68	574	2025-01-04	2025-01-30	600.00	Returned Late
49	41	661	2025-03-15	2025-04-09	550.00	Returned Late