

MySQL PROJECT

Create a database named library and following TABLES in the database:

1. Branch
2. Employee
3. Books
4. Customer
5. IssueStatus
5. ReturnStatus

Attributes for the tables:

1. Branch
 - Branch_no - Set as PRIMARY KEY
 - Manager_Id
 - Branch_address
 - Contact_no

The screenshot displays the MySQL Workbench interface. The top pane shows SQL queries being executed. The bottom pane shows the results of these queries in a table format.

SQL Queries:

```
52 foreign key (isbn_book2) references books(isbn) on delete cascade
53 );
54
55 • insert into branch values(1,101,'Kochi',999999),
56 (2,102,'Mumbai',999998),(3,103,'Delhi',999997),
57 (4,104,'Kolkata',999996),(5,105,'Chennai',999995);
58
59 • select * from Branch;
```

Result Grid:

	Branch_no	Manager_Id	Branch_address	Contact_no
▶	1	101	Kochi	999999
	2	102	Mumbai	999998
	3	103	Delhi	999997
	4	104	Kolkata	999996
	5	105	Chennai	999995
*	NULL	NULL	NULL	NULL

Branch 1 x

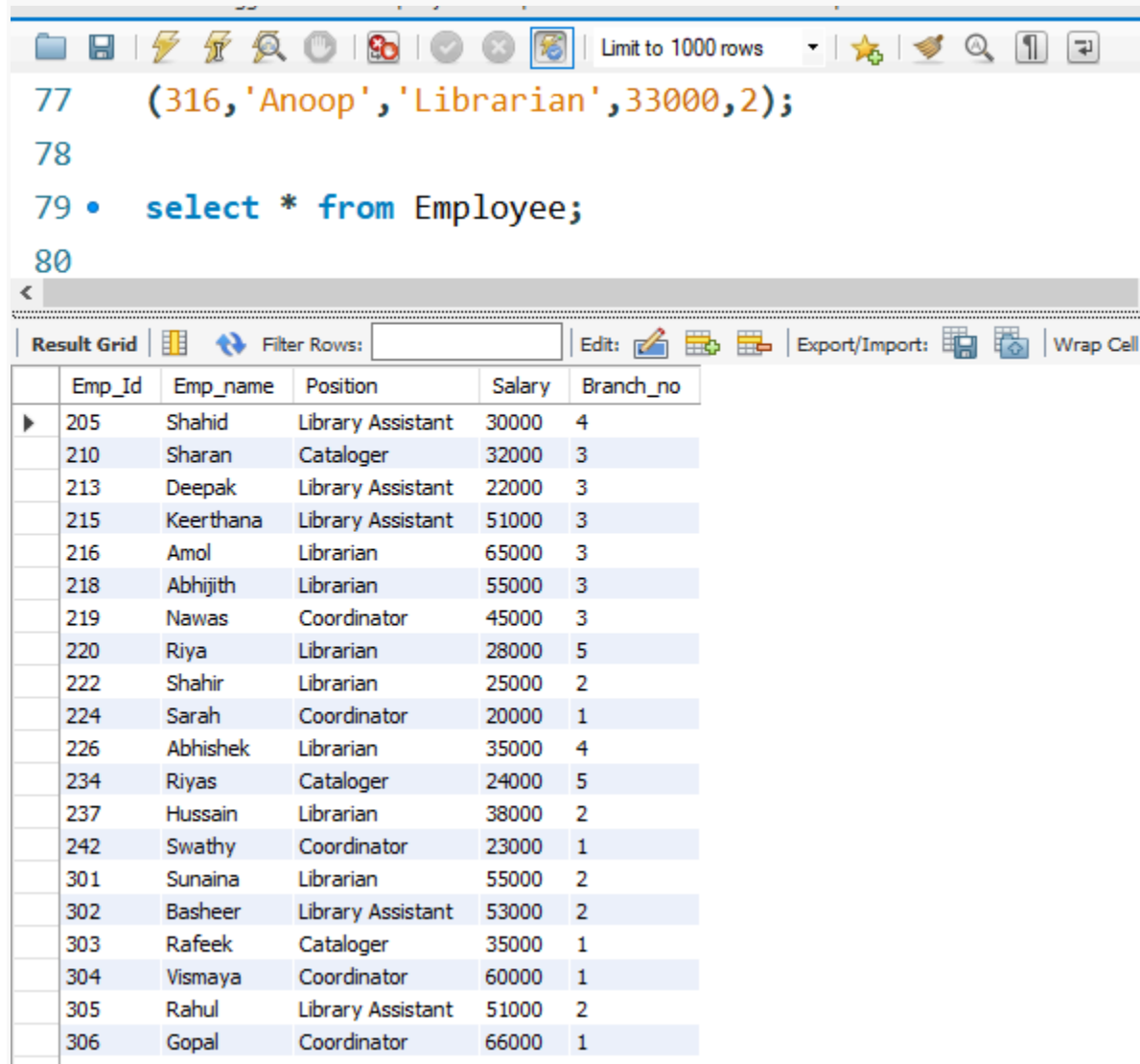
Output:

Action Output

#	Time	Action	Message
✓ 1	22:36:14	use library	0 row(s) affected
✓ 2	22:38:32	select * from Branch LIMIT 0, 1000	5 row(s) returned

2. Employee

- Emp_Id – Set as PRIMARY KEY
- Emp_name
- Position
- Salary
- Branch_no - Set as FOREIGN KEY and it refer Branch_no in Branch table



The screenshot shows a database management tool interface. At the top, there is a toolbar with various icons and a dropdown menu set to "Limit to 1000 rows". Below the toolbar, the SQL editor contains the following code:

```
77 (316,'Anoop','Librarian',33000,2);
78
79 • select * from Employee;
80
```

Below the SQL editor, there is a "Result Grid" section. It includes a "Filter Rows:" input field, an "Edit:" button, and "Export/Import:" and "Wrap Cell" options. The result grid displays a table with the following data:

	Emp_Id	Emp_name	Position	Salary	Branch_no
▶	205	Shahid	Library Assistant	30000	4
	210	Sharan	Cataloger	32000	3
	213	Deepak	Library Assistant	22000	3
	215	Keerthana	Library Assistant	51000	3
	216	Amol	Librarian	65000	3
	218	Abhijith	Librarian	55000	3
	219	Nawas	Coordinator	45000	3
	220	Riya	Librarian	28000	5
	222	Shahir	Librarian	25000	2
	224	Sarah	Coordinator	20000	1
	226	Abhishek	Librarian	35000	4
	234	Riyas	Cataloger	24000	5
	237	Hussain	Librarian	38000	2
	242	Swathy	Coordinator	23000	1
	301	Sunaina	Librarian	55000	2
	302	Basheer	Library Assistant	53000	2
	303	Rafeek	Cataloger	35000	1
	304	Vismaya	Coordinator	60000	1
	305	Rahul	Library Assistant	51000	2
	306	Gopal	Coordinator	66000	1

3. Books

- ISBN - Set as PRIMARY KEY
- Book_title
- Category
- Rental_Price
- Status [Give yes if book available and no if book not available]
- Author
- Publisher

```
101 ('ISBN:87055','A BRIEF HISTORY','HISTORY',900,default,'STEPHEN HAWKING','BANTAM');
102
103 • select * from Books;
104
```

ISBN	Book_title	Category	Rental_Price	Status	Author	Publisher
ISBN:7803	THE HISTORY OF LOVE	HISTORY	400	YES	NICOLE KRAUSS	W.W.NORTON&COMPANY
ISBN:781400	THE SECRET HISTORY	HISTORY	600	YES	DONNA TARTT	VINTAGE
ISBN:87055	A BRIEF HISTORY	HISTORY	900	YES	STEPHEN HAWKING	BANTAM
ISBN13:978-00623	SAPIENS	HISTORY	700	NO	YUVAL NOAH HARARI	HARPER PERENNIAL
ISBN13:978-006231	DRIVE	BUSINESS	300	YES	DANIEL H.PINK	RIVERHEAD BOOKS
ISBN13:978-03075	GONE GIRL	MYSTERY/THRILLER	250	YES	GILLIAN FLYNN	BROADWAY BOOKS
ISBN13:978-04156	BOSSYPANTS	BIOGRAPHY	4500	YES	TINA FEY	BACK BAY BOOKS
ISBN13:978-04411	DUNE	SCIENCE FICTION	500	YES	FRANK HERBERT	ACE BOOKS
ISBN13:978-04415	NEUROMANCER	SCIENCE FICTION	500	YES	WILLIAM GIBSON	ACE BOOKS
ISBN13:978-17130	SHARIKUM PONNUM	FICTION	550	YES	MUHAMMED BASHEER	DC BOOKS
ISBN13:978-81264	OZHICHUVIDUKA	FICTION	450	NO	SARAH JOSEPH	DC BOOKS
ISBN13:978-8165	NALLAVANNAM	FICTION	200	NO	C.RADHAKRISHNAN	DC BOOKS
ISBN13:978-81713	AANAKKALLOPILLI	FICTION	350	YES	O.V.VIJAYAN	DC BOOKS
ISBN13:978006	THE ALCHEMIST	FICTION	800	YES	PAULO COELHO	HARPERONE
ISBN13:978030	THE DAVINCI CODE	MYSTERY/THRILLER	900	YES	DAN BROWN	ANCHOR
ISBN13:978055	A GAME OF THRONES	FANTASY	900	YES	GEORGE R.R. MARTIN	BANTAM
ISBN13:978059	HARRY POTTER	FANTASY	800	NO	J.K.ROWLING	SCHOLASTIC INC
ISBN13:978145	STEVE JOBS	BIOGRAPHY	650	YES	WALTER ISAACSON	SIMON & SCHUSTER
ISBN13:978152	BECOMING	BIOGRAPHY	800	YES	MICHELLE OBAMA	CROWN PUBLISHING
ISBN13:978349	THE HUNGER GAMES	FICTION	600	YES	SUZANNE COLLINS	SCHOLASTIC PRESS
ISBN13:978547	THE HOBBIT	FANTASY	800	YES	J.R.R. TOLKIEN	MARINER BOOKS

4. Customer

- Customer_Id - Set as PRIMARY KEY
- Customer_name
- Customer_address
- Reg_date

The screenshot displays a database management interface. At the top, a toolbar includes icons for file operations, a search icon, and a 'Limit to 1000 rows' dropdown. Below the toolbar, a SQL editor shows the following queries:

```
93 (1014, 'Sarsha', 'Kerala', '2023-12-30'), (1015, 'Lincy', 'Chennai', '2024-01-08'),
94 (1016, 'Jeweria', 'Mumbai', '2024-02-10'), (1017, 'Chadrani', 'Delhi', '2024-02-20');
95
96 • select * from Customer;
```

Below the editor is a 'Result Grid' section with a 'Filter Rows' input field and buttons for 'Edit', 'Export/Import', and 'Wrap Cell Content'. The grid contains the following data:

	Customer_Id	Customer_name	Customer_address	Reg_date
▶	1001	Adheena	Kerala	2021-05-11
	1002	Shyam	Karnataka	2021-10-22
	1003	Deepu	Mumbai	2021-11-10
	1004	Aleesha	kolkata	2021-12-22
	1005	Adarsh	Kerala	2022-06-23
	1006	Salman	Delhi	2022-08-10
	1007	Rajesh	Chennai	2022-10-27
	1008	Uma	Mumbai	2022-12-23
	1009	Sathvik	Chennai	2023-01-11
	1010	Tridev	Delhi	2023-02-10
	1011	Aiswarya	Kerala	2023-05-30
	1012	Vijay	Kolkata	2023-07-15
	1013	Komal	kolkata	2023-09-23
	1014	Sarsha	Kerala	2023-12-30
	1015	Lincy	Chennai	2024-01-08
	1016	Jeweria	Mumbai	2024-02-10
	1017	Chadrani	Delhi	2024-02-20

Below the result grid is a tab labeled 'Customer 4'. Underneath is an 'Output' section with a dropdown menu set to 'Action Output'. The output table shows the following entry:

#	Time	Action	Message
4	22:42:04	select * from Books LIMIT 0 1000	10 row(s) returned

5. IssueStatus

- Issue_Id - Set as PRIMARY KEY
- Issued_cust – Set as FOREIGN KEY and it refer customer_id in CUSTOMER table Issued_book_name
- Issue_date
- Isbn_book – Set as FOREIGN KEY and it should refer isbn in BOOKS table

The screenshot shows a database management tool interface. At the top, there is a toolbar with various icons and a dropdown menu set to 'Limit to 1000 rows'. Below the toolbar, a SQL query is entered in a text area:

```
124 (020,1010,'2023-06-22','ISBN13:978547');
125
126 • select * from IssueStatus;
127
```

Below the query, there is a 'Result Grid' section. It includes a 'Filter Rows' input field, an 'Edit' button, and 'Export/Import' buttons. The grid displays the results of the SQL query, showing 20 rows of data with the following columns: Issue_Id, issued_cust, Issue_date, and isbn_book.

	Issue_Id	issued_cust	Issue_date	isbn_book
▶	1	1003	2021-11-15	ISBN13:978-04411
	2	1005	2022-07-13	ISBN13:978-006231
	3	1007	2022-11-01	ISBN13:978-04415
	4	1012	2021-12-30	ISBN13:978-04156
	5	1010	2023-02-15	ISBN13:978-81264
	6	1004	2023-01-05	ISBN13:978-8165
	7	1016	2024-02-15	ISBN13:978-81713
	8	1007	2021-11-15	ISBN13:978-00623
	9	1017	2024-02-21	ISBN13:978-00623
	10	1008	2023-01-02	ISBN13:978-04411
	11	1009	2023-01-30	ISBN13:978-03075
	12	1011	2024-02-11	ISBN13:978-17130
	13	1005	2022-11-01	ISBN13:978-17130
	14	1014	2024-02-21	ISBN13:978-04415
	15	1013	2023-10-15	ISBN13:978-04411
	16	1015	2024-02-10	ISBN13:978-8165
	17	1006	2022-09-10	ISBN13:978-03075
	18	1013	2023-06-06	ISBN13:978-04411
	19	1014	2023-06-10	ISBN13:978055
	20	1010	2023-06-22	ISBN13:978547

6. ReturnStatus

- Return_Id - Set as PRIMARY KEY
- Return_cust
- Return_book_name
- Return_date
- Isbn_book2 - Set as FOREIGN KEY and it should refer isbn in BOOKS table

The screenshot shows a database IDE interface. At the top, there's a toolbar with various icons and a 'Limit to 1000 rows' dropdown. Below the toolbar, SQL queries are entered in a text area. The first query is a multi-line INSERT statement. The second query is a SELECT statement. Below the queries, a 'Result Grid' tab is active, displaying a table with 6 columns: Return_Id, Return_cust, Return_book_name, Return_date, and isbn_book2. The table contains 18 rows of data, with the last row showing NULL values. At the bottom, there's a tab labeled 'ReturnStatus 6' and an 'Output' section.

```
119      (516, 'LINCY', 'NALLAVANNAM', '2024-02-22', 'ISBN13:978-8165'), (517, 'S
120      (518, 'KOMAL', 'DUNE', '2023-10-22', 'ISBN13:978-04411');
121
122 •    select * from ReturnStatus;
```

Return_Id	Return_cust	Return_book_name	Return_date	isbn_book2
501	DEEPU	DUNE	2022-02-28	ISBN13:978-04411
502	ADARSH	DRIVE	2022-12-30	ISBN13:978-006231
503	RAJESH	NUEROMANCER	2023-04-03	ISBN13:978-04415
504	VIJAY	BOSSYPANTS	2024-01-20	ISBN13:978-04156
505	TRIDEV	OZHICHUVIDUKA	2023-12-25	ISBN13:978-81264
506	ALEESHA	NALLAVANNAM	2023-06-30	ISBN13:978-8165
507	JEWERIA	AANAKKALLOPILLI	2024-02-22	ISBN13:978-81713
508	RAJESH	SAPIENS	2022-11-08	ISBN13:978-00623
509	CHADRANI	SAPIENS	2024-02-21	ISBN13:978-00623
510	UMA	DUNE	2023-12-01	ISBN13:978-04411
511	SATHVIK	GONE GIRL	2023-02-05	ISBN13:978-03075
512	AISWARYA	SHARIKUM PONNUM	2024-02-21	ISBN13:978-17130
513	ADARSH	SHARIKUM PONNUM	2023-03-07	ISBN13:978-17130
514	SARSHA	NEUROMANCER	2024-02-22	ISBN13:978-04415
515	KOMAL	DUNE	2024-02-10	ISBN13:978-04411
516	LINCY	NALLAVANNAM	2024-02-22	ISBN13:978-8165
517	SALMAN	GONE GIRL	2022-12-31	ISBN13:978-03075
518	KOMAL	DUNE	2023-10-22	ISBN13:978-04411
*	NULL	NULL	NULL	NULL

ReturnStatus 6 ×

Output

Display all the tables and Write the queries for the following :

1. Retrieve the book title, category, and rental price of all available books.

```
142 • select * from ReturnStatus;
143
144 #Retrieve the book title, category, and rental price of all available books.
145 • select book_title,category,rental_price from books where status = 'YES';
146
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [IA](#)

book_title	category	rental_price
THE SECRET HISTORY	HISTORY	600
A BRIEF HISTORY	HISTORY	900
DRIVE	BUSINESS	300
GONE GIRL	MYSTERY/THRILLER	250
BOSSYPANTS	BIOGRAPHY	4500
DUNE	SCIENCE FICTION	500
NEUROMANCER	SCIENCE FICTION	500
SHARIKUM PONNUM	FICTION	550
AANAKKALLOPILLI	FICTION	350
THE ALCHEMIST	FICTION	800
THE DAVINCI CODE	MYSTERY/THRILLER	900
A GAME OF THRONES	FANTASY	900
STEVE JOBS	BIOGRAPHY	650
BECOMING	BIOGRAPHY	800
THE HUNGER GAMES	FICTION	600
THE HOBBIT	FANTASY	800

The screenshot shows a database management tool interface. At the top, there's a toolbar with various icons like save, undo, redo, search, etc., followed by a dropdown menu set to "Limit to 1000 rows". Below the toolbar, two SQL queries are displayed:

```
141 • select book_title,category,rental_price from books;
142
143 • select emp_name,position,salary from employee order by salary desc;
144
```

A horizontal scrollbar is visible below the queries. Underneath, the "Result Grid" tab is active, displaying the output of the second query. It includes options for "Filter Rows:" (with an input field), "Export:", and "Wrap Cell Content:". The result grid itself contains three columns: "emp_name", "position", and "salary", with data sorted in descending order of salary.

	emp_name	position	salary
▶	Prasad	Librarian	70000
	Gopal	Coordinator	66000
	Amol	Librarian	65000
	Vismaya	Coordinator	60000
	Mercy	Cataloger	59000
	Abhijith	Librarian	55000
	Sunaina	Librarian	55000
	Basil	Coordinator	55000
	Basheer	Library Assistant	53000
	Vimal	Library Assistant	53000
	Anzy	Library Assistant	53000
	Keerthana	Library Assistant	51000
	Rahul	Library Assistant	51000
	Satheesh	Library Assistant	50000
	Mary	Library Assistant	49000
	Nawas	Coordinator	45000
	Hussain	Librarian	38000
	Abhishek	Librarian	35000
	Rafeek	Cataloger	35000
	Vishal	Librarian	33000

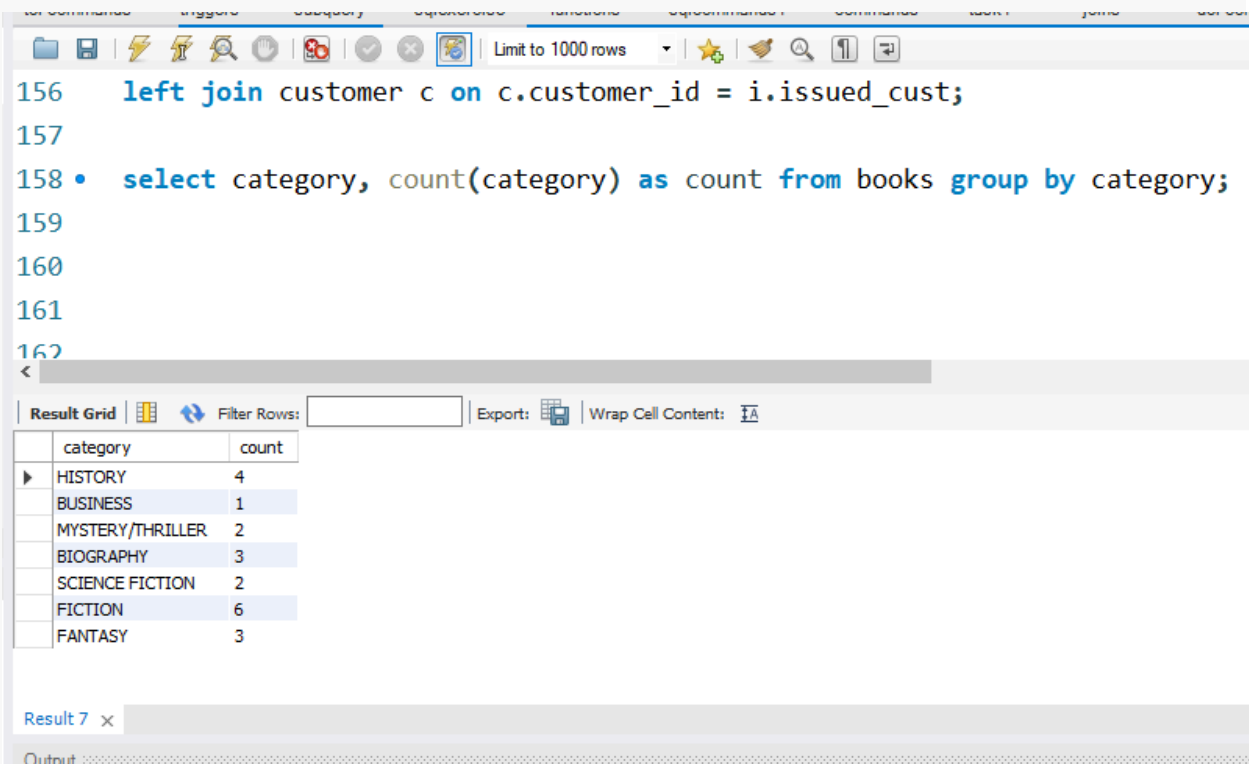
3. Retrieve the book titles and the corresponding customers who have issued those books.

```
150 • select i.issued_cust,b.book_title,c.customer_name
151 from issuestatus i left join books b
152 on i.isbn_book = b.isbn
153 left join customer c on c.customer_id = i.issued_cust;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [IA](#)

	issued_cust	book_title	customer_name
▶	1003	DUNE	Deepu
	1005	DRIVE	Adarsh
	1007	NEUROMANCER	Rajesh
	1012	BOSSYPANTS	Vijay
	1010	OZHICHUVIDUKA	Tridev
	1004	NALLAVANNAM	Aleesha
	1016	AANAKKALLOPILLI	Jeweria
	1007	SAPIENS	Rajesh
	1017	SAPIENS	Chadrani
	1008	DUNE	Uma
	1009	GONE GIRL	Sathvik
	1011	SHARIKUM PONNUM	Aiswarya
	1005	SHARIKUM PONNUM	Adarsh
	1014	NEUROMANCER	Sarsha
	1013	DUNE	Komal
	1015	NALLAVANNAM	Lincy
	1006	GONE GIRL	Salman
	1013	DUNE	Komal
	1014	A GAME OF THRO...	Sarsha
	1010	THE HOBBIT	Tridev

4. Display the total count of books in each category.



The screenshot shows a SQL IDE interface. The query editor contains the following SQL code:

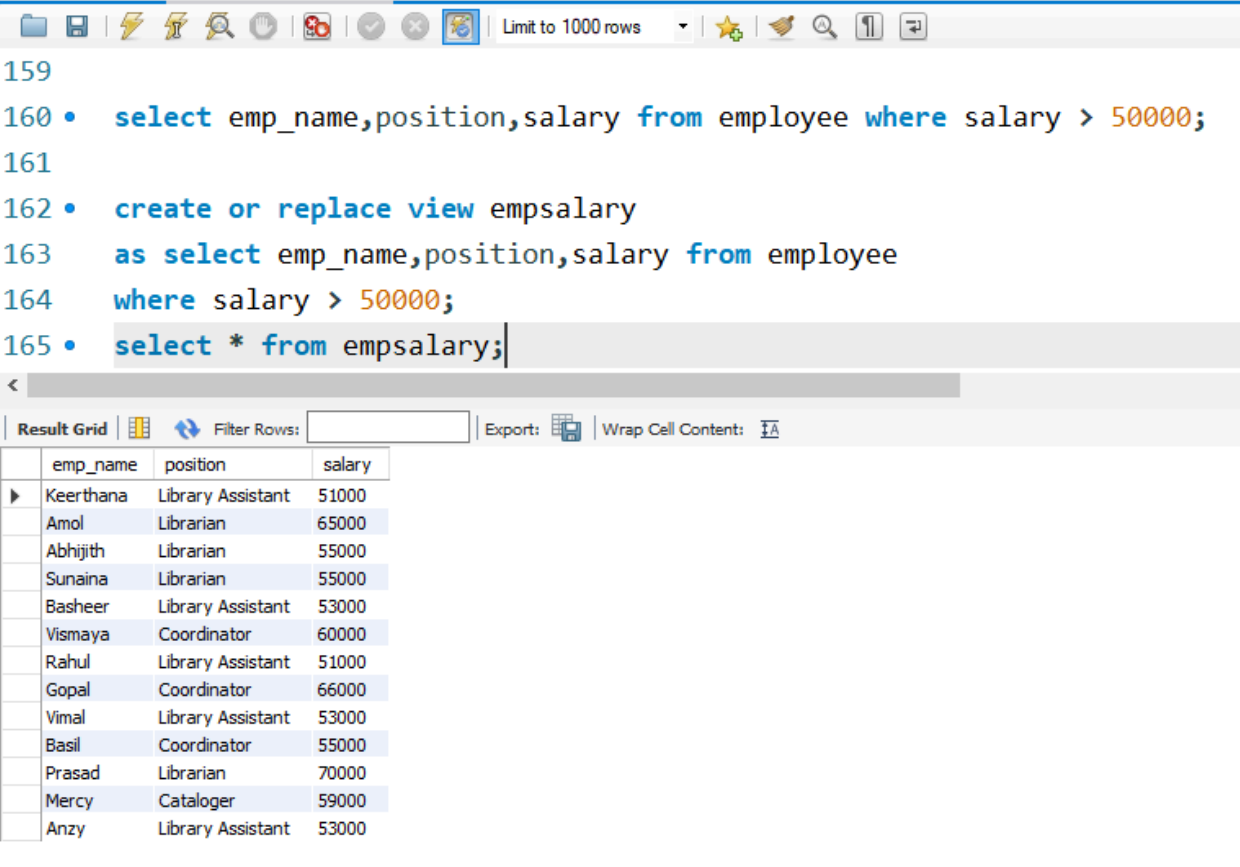
```
156 left join customer c on c.customer_id = i.issued_cust;
157
158 • select category, count(category) as count from books group by category;
159
160
161
162
```

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

category	count
HISTORY	4
BUSINESS	1
MYSTERY/THRILLER	2
BIOGRAPHY	3
SCIENCE FICTION	2
FICTION	6
FANTASY	3

At the bottom of the IDE, there is an 'Output' section.

5. Retrieve the employee names and their positions for the employees whose salaries are above Rs.50,000.



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

```
159
160 • select emp_name,position,salary from employee where salary > 50000;
161
162 • create or replace view empsalary
163   as select emp_name,position,salary from employee
164   where salary > 50000;
165 • select * from empsalary;
```




Below the editor is a 'Result Grid' section with a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' checkbox. The grid displays the results of the query, showing columns for emp_name, position, and salary.

emp_name	position	salary
Keerthana	Library Assistant	51000
Amol	Librarian	65000
Abhijith	Librarian	55000
Sunaina	Librarian	55000
Basheer	Library Assistant	53000
Vismaya	Coordinator	60000
Rahul	Library Assistant	51000
Gopal	Coordinator	66000
Vimal	Library Assistant	53000
Basil	Coordinator	55000
Prasad	Librarian	70000
Mercy	Cataloger	59000
Anzy	Library Assistant	53000

6. List the customer names who registered before 2022-01-01 and have not issued any books yet.

```
150 • select customer_name,reg_date from customer where reg_date < '2022-01-01';
151
152 • select c.customer_id,c.customer_name,c.reg_date,i.issue_id
153 from customer c left join issuestatus i
154 on c.customer_id=i.issued_cust
155 where c.reg_date < '2022-01-01' and i.issue_id is null;
156
157
158
159
160
```

<

Result Grid |  Filter Rows: | Export:  | Wrap Cell Content: 

	customer_id	customer_name	reg_date	issue_id
▶	1001	Adheena	2021-05-11	NULL
	1002	Shyam	2021-10-22	NULL

Result 4 x

7. Display the branch numbers and the total count of employees in each branch.

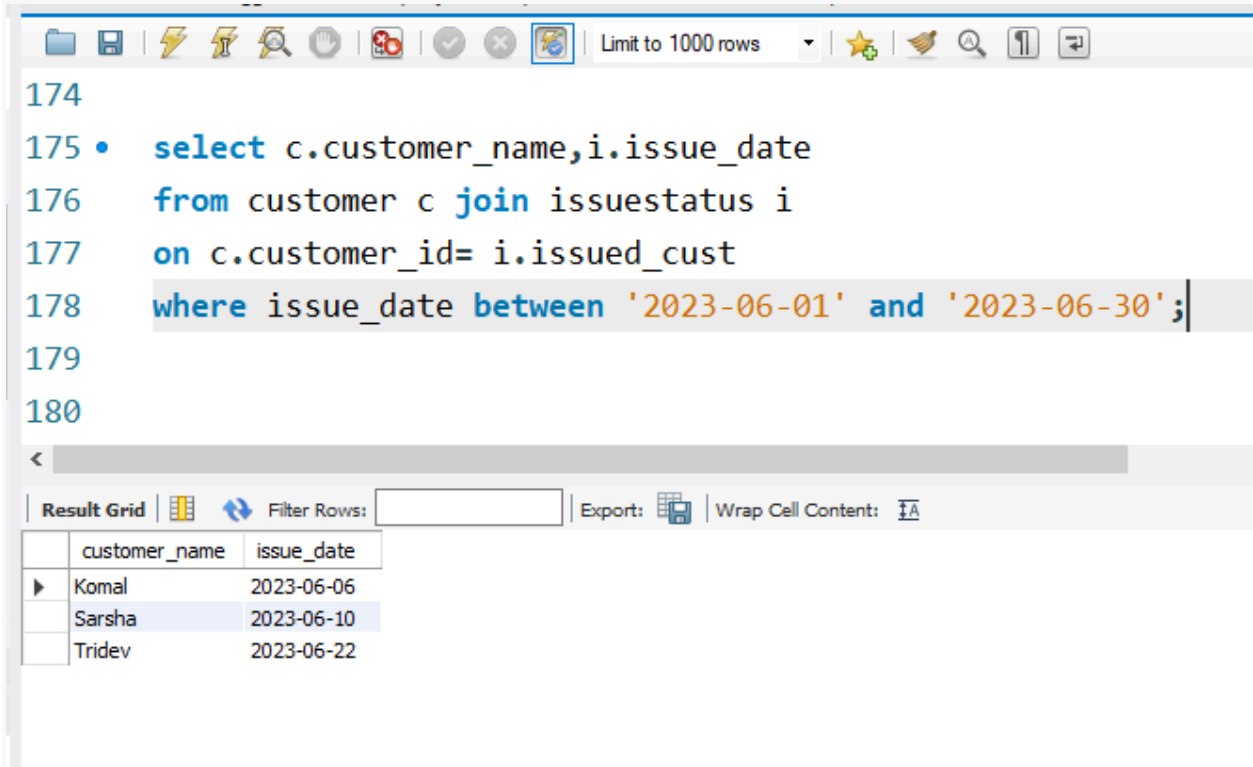
```
183
184 • select b.branch_no,b.branch_address,count(e.branch_no) as 'count of employees'
185 from branch b left join employee e
186 on b.branch_no=e.branch_no
187 group by b.branch_no,b.branch_address;
188
189
```

<

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [IA](#)

	branch_no	branch_address	count of employees
▶	1	Kochi	5
	2	Mumbai	9
	3	Delhi	6
	4	Kolkata	5
	5	Chennai	5

8. Display the names of customers who have issued books in the month of June 2023.



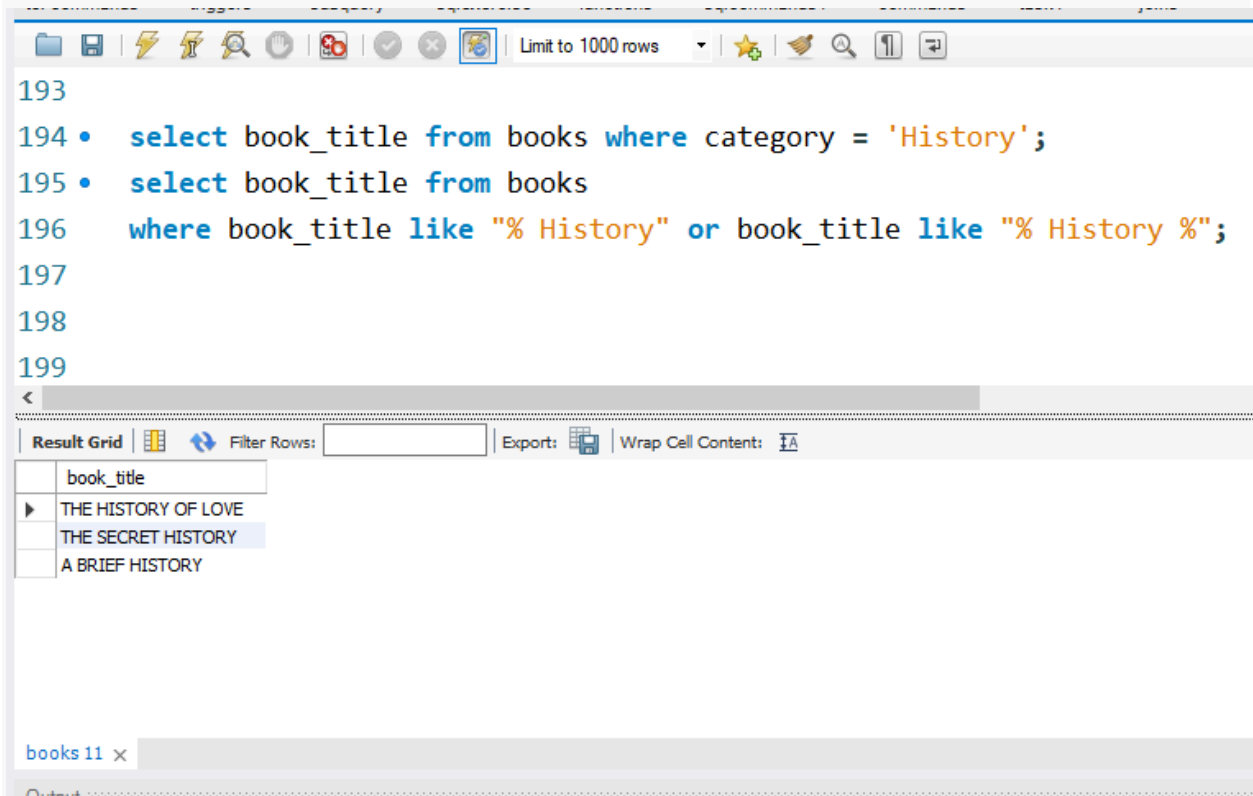
The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and a 'Limit to 1000 rows' dropdown. The SQL editor contains the following query:

```
174  
175 • select c.customer_name,i.issue_date  
176 from customer c join issuestatus i  
177 on c.customer_id= i.issued_cust  
178 where issue_date between '2023-06-01' and '2023-06-30';  
179  
180
```

Below the editor is a 'Result Grid' section with a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' checkbox. The results are displayed in a table:

	customer_name	issue_date
▶	Komal	2023-06-06
	Sarsha	2023-06-10
	Tridev	2023-06-22

9. Retrieve book_title from book table containing history.



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

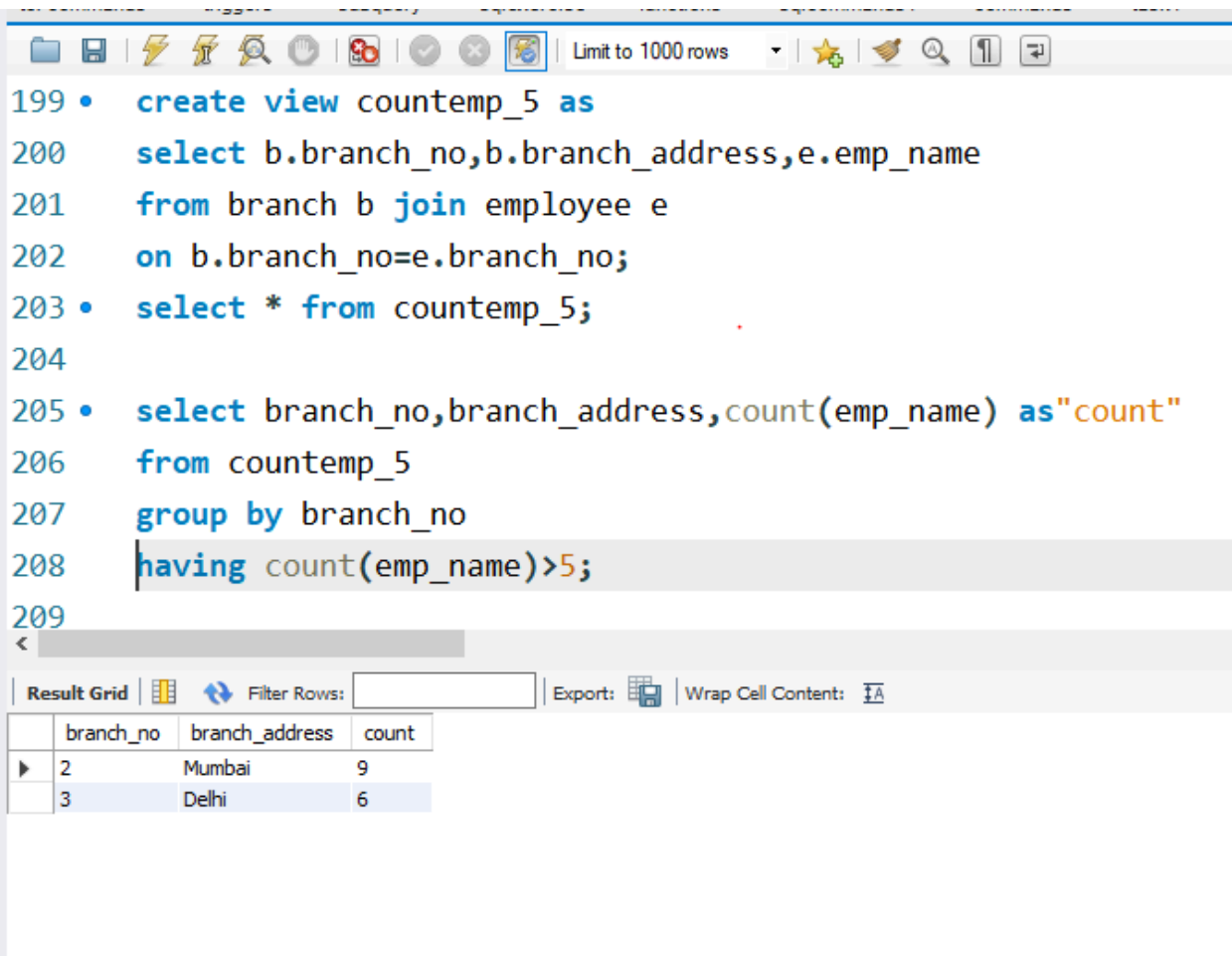
```
193
194 • select book_title from books where category = 'History';
195 • select book_title from books
196   where book_title like "% History" or book_title like "% History %";
197
198
199
```

Below the editor is a 'Result Grid' section with a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' checkbox. The results are displayed in a table:

book_title
THE HISTORY OF LOVE
THE SECRET HISTORY
A BRIEF HISTORY

At the bottom, a tab labeled 'books 11' is visible, and the 'Output' pane is empty.

10.Retrieve the branch numbers along with the count of employees for branches having more than 5 employees



```
199 • create view countemp_5 as
200   select b.branch_no,b.branch_address,e.emp_name
201   from branch b join employee e
202   on b.branch_no=e.branch_no;
203 • select * from countemp_5;
204
205 • select branch_no,branch_address,count(emp_name) as"count"
206   from countemp_5
207   group by branch_no
208   having count(emp_name)>5;
209
```

Result Grid

	branch_no	branch_address	count
▶	2	Mumbai	9
	3	Delhi	6

11.To add a new customer to the customer table.

The screenshot shows a database management tool interface with a SQL editor and a results pane. The SQL editor contains the following code:

```
219 delimiter $
220 • create procedure new_cust(cust_id int,cust_name varchar(20),cust_adress varchar(20),r_date date)
221 • begin
222     insert into customer values(cust_id,cust_name,cust_adress,r_date);
223 • end $
224 delimiter ;
225
226 • call new_cust(1018,'Nidhi','Chennai',curdate());
227 • select *from customer;
```

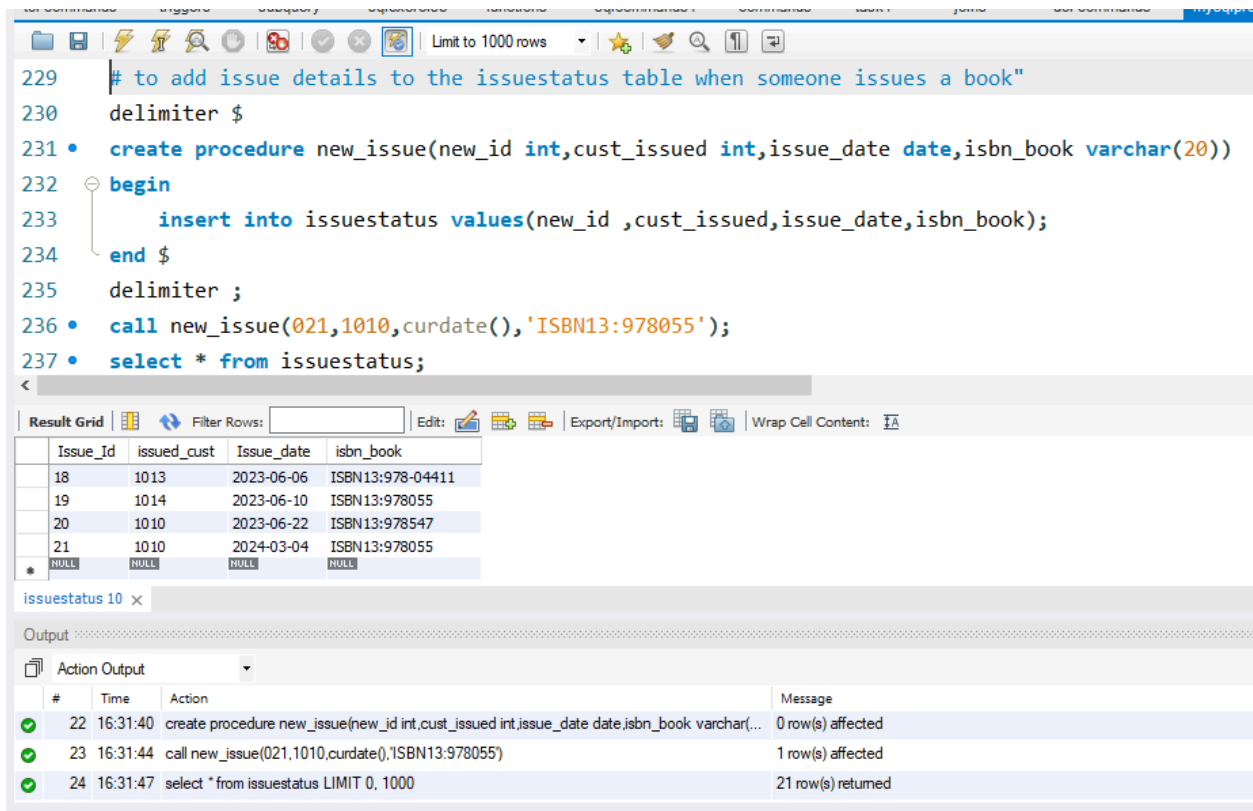
Below the SQL editor, the 'Result Grid' shows the output of the SQL statements. The first statement is a 'call' statement, and the second is a 'select' statement. The 'select' statement returns 18 rows of data from the 'customer' table.

Customer_Id	Customer_name	Customer_adress	Reg_date
1015	Lincy	Chennai	2024-01-08
1016	Jeweria	Mumbai	2024-02-10
1017	Chadrani	Delhi	2024-02-20
1018	Nidhi	Chennai	2024-03-04

The 'Output' pane shows the execution log for the SQL statements:

#	Time	Action	Message
✓ 14	16:25:43	create procedure new_cust(cust_id int,cust_name varchar(20),cust_adress varchar(20),r_d...	0 row(s) affected
✓ 15	16:25:49	call new_cust(1018,'Nidhi','Chennai',curdate())	1 row(s) affected
✓ 16	16:26:08	select *from customer LIMIT 0, 1000	18 row(s) returned

12.To add issue details to the issuestatus table when someone issues a book"



The screenshot displays a database management interface with a SQL editor and a results pane. The SQL editor contains the following code:

```
229 # to add issue details to the issuestatus table when someone issues a book"
230 delimiter $
231 • create procedure new_issue(new_id int,cust_issued int,issue_date date,isbn_book varchar(20))
232 • begin
233     insert into issuestatus values(new_id ,cust_issued,issue_date,isbn_book);
234 • end $
235 delimiter ;
236 • call new_issue(021,1010,curdate(),'ISBN13:978055');
237 • select * from issuestatus;
```

Below the code, the 'Result Grid' shows the contents of the 'issuestatus' table:

Issue_Id	issued_cust	Issue_date	isbn_book
18	1013	2023-06-06	ISBN13:978-04411
19	1014	2023-06-10	ISBN13:978055
20	1010	2023-06-22	ISBN13:978547
21	1010	2024-03-04	ISBN13:978055
•	NULL	NULL	NULL

The 'Output' pane shows the 'Action Output' for the executed SQL statements:

#	Time	Action	Message
✓ 22	16:31:40	create procedure new_issue(new_id int,cust_issued int,issue_date date,isbn_book varchar(...	0 row(s) affected
✓ 23	16:31:44	call new_issue(021,1010,curdate(),'ISBN13:978055')	1 row(s) affected
✓ 24	16:31:47	select * from issuestatus LIMIT 0, 1000	21 row(s) returned

13. Find employees whose salary < the avg salary of all employees?

```
241 #find employees whose salary < the avg salary of all employees.
242 • select avg(salary) as sal_avg from employee;
243 • select * from employee where salary < 43266.6667;
244 • select emp_name, salary from employee where salary < (select avg(salary) from employee);
245
246 • select emp_name, salary,
247 • case
248 • when salary > (select avg(salary) as sal_avg from employee)
249 • then "above average"
250 • else "below average"
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	emp_name	salary
▶	Shahid	30000
	Sharan	32000
	Deepak	22000
	Riya	28000
	Shahir	25000
	Sarah	20000
	Abhishek	35000
	Riyas	24000
	Hussain	38000

14. Display name and position of employee who has the highest salary.

The screenshot shows a SQL IDE interface. The top pane contains a SQL script with the following lines:

```
248 when salary > (select avg(salary) as sal_avg from employee)
249 then "above average"
250 else "below average"
251 end as salary_level from employee;
252
253 #display name and position of employee who have the highest salary.
254 • select emp_name, position, salary from employee where salary = (select max(salary) from employee);
255
```

The bottom pane shows the results of the query. It includes a "Result Grid" with the following data:

emp_name	position	salary
Prasad	Librarian	70000

Below the result grid, there is an "Output" section with a tab labeled "employee 18 x". The "Action Output" tab is selected, showing the following log entries:

#	Time	Action	Message	Duration
31	16:36:04	select emp_name, salary from employee where salary < (select avg(salary) from employee) LI...	14 row(s) returned	0.000
32	16:37:13	select emp_name, position, salary from employee where salary = (select max(salary) from empl...	1 row(s) returned	0.000

15. Find the second largest salary earned employee.

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

```
257 #find the second largest salary earned employee.
258 • select emp_name,salary from employee order by salary desc limit 1 offset 1;
259
260 • select max(salary) from employee;
261 • select max(salary) from employee where salary < (select max(salary) from employee);
262
263 • select emp_name, salary AS second_largest_salary
264   from employee
265   where salary = (select MAX(salary) from employee
266   where salary < (select MAX(salary) from employee));
```

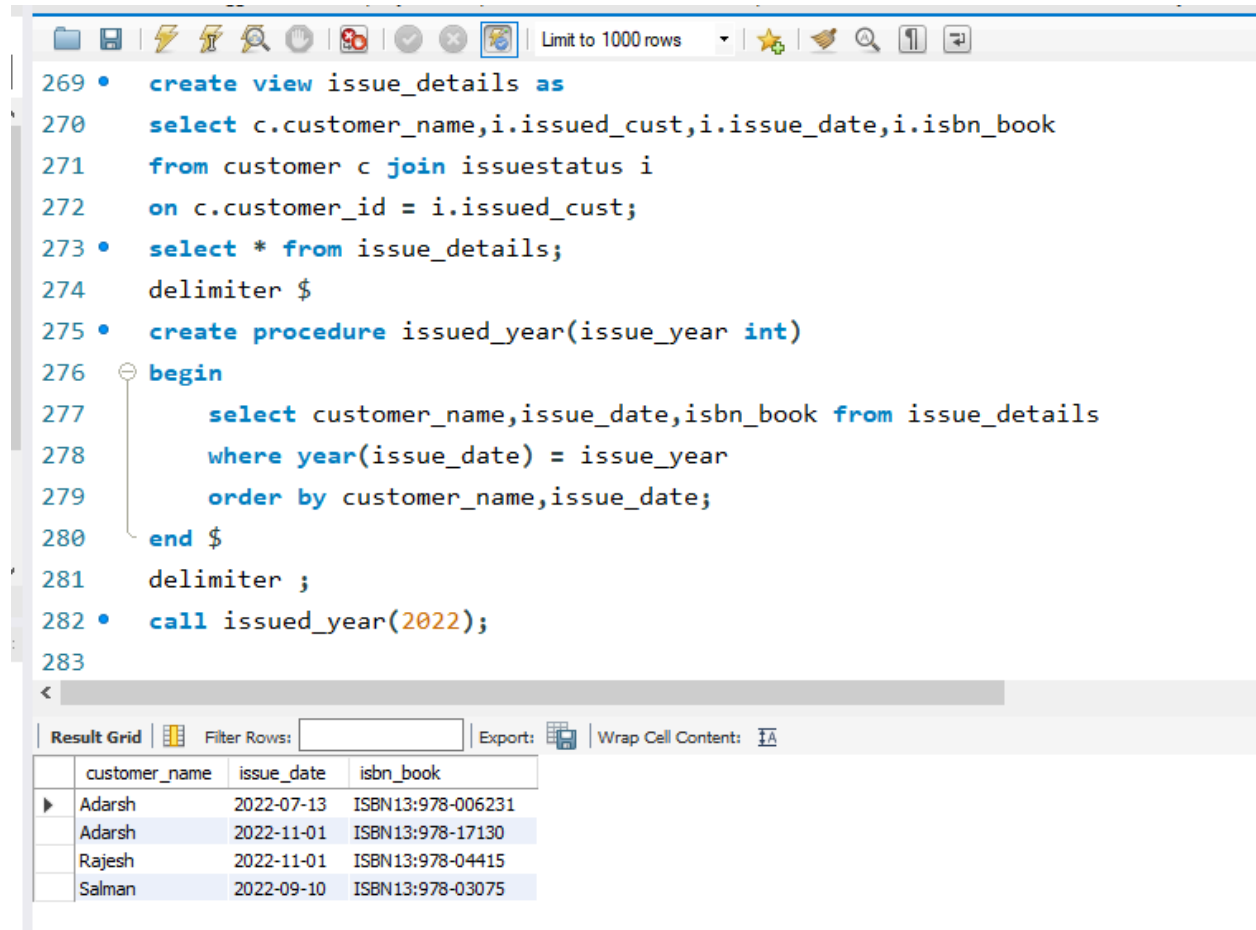
The result grid shows the following data:

emp_name	second_largest_salary
Gopal	66000

The output pane shows the following messages:

#	Time	Action	Message
34	16:38:35	select max(salary) from employee where salary < (select max(salary) from employee) LIMIT 0,...	1 row(s) returned
35	16:38:39	select emp_name, salary AS second_largest_salary from employee where salary = (select M...	1 row(s) returned

16.Retrieve the details of customers who have issued books separately in the years 2021, 2022, 2023, and 2024?

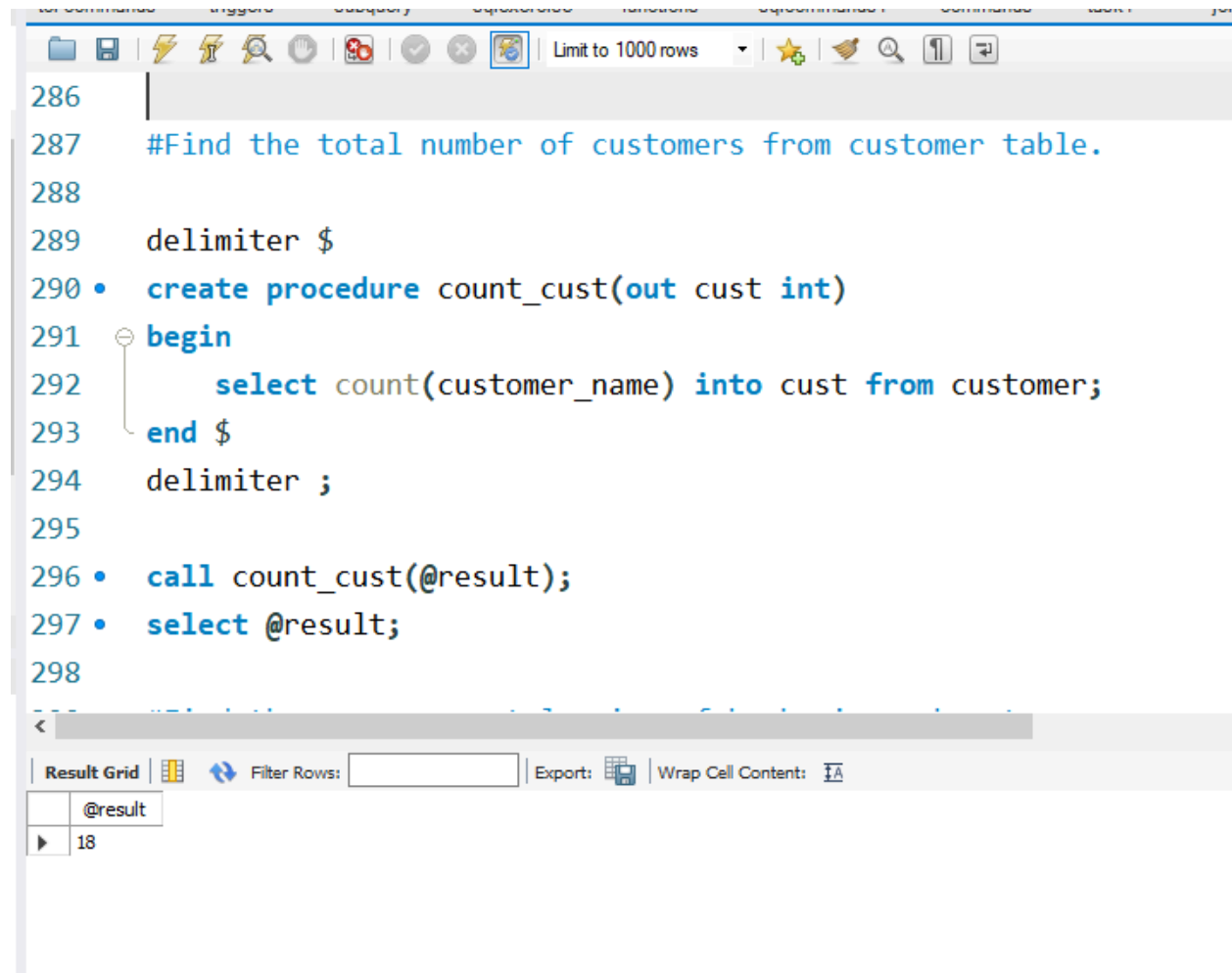


```
269 • create view issue_details as
270     select c.customer_name,i.issued_cust,i.issue_date,i.isbn_book
271     from customer c join issuestatus i
272     on c.customer_id = i.issued_cust;
273 • select * from issue_details;
274     delimiter $
275 • create procedure issued_year(issue_year int)
276     begin
277         select customer_name,issue_date,isbn_book from issue_details
278         where year(issue_date) = issue_year
279         order by customer_name,issue_date;
280     end $
281     delimiter ;
282 • call issued_year(2022);
283
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	customer_name	issue_date	isbn_book
▶	Adarsh	2022-07-13	ISBN13:978-006231
	Adarsh	2022-11-01	ISBN13:978-17130
	Rajesh	2022-11-01	ISBN13:978-04415
	Salman	2022-09-10	ISBN13:978-03075

17. Find the total number of customers from customer table.



The screenshot shows a SQL IDE interface. The main editor displays a SQL script with line numbers 286 through 298. The script defines a procedure named `count_cust` that takes an output parameter `cust` of type `int`. The procedure body consists of a single `select` statement that counts the number of rows in the `customer` table and stores the result in the `cust` parameter. The script then calls the procedure with `@result` as the argument and selects the value of `@result`.

```
286  
287 #Find the total number of customers from customer table.  
288  
289 delimiter $  
290 • create procedure count_cust(out cust int)  
291 • begin  
292     select count(customer_name) into cust from customer;  
293 • end $  
294 delimiter ;  
295  
296 • call count_cust(@result);  
297 • select @result;  
298
```

Below the editor, the 'Result Grid' tab is active, showing a single row with the value 18, which is the result of the `count` operation.

	@result
▶	18

18. Find the average rental price of books in each category.

298
299
300 #Find the average rental price of books in each category.
301 • `select category, avg(rental_price) as avg_rental_price from books group by category;`
302
303
304

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [IA](#)

category	avg_rental_price
HISTORY	650.0000
BUSINESS	300.0000
MYSTERY/THRILLER	575.0000
BIOGRAPHY	1983.3333
SCIENCE FICTION	500.0000
FICTION	491.6667
FANTASY	833.3333

Result 26 x

Output

Action Output

#	Time	Action	Message
✓ 42	16:44:24	select @result LIMIT 0, 1000	1 row(s) returned
✓ 43	16:46:22	select category, avg(rental_price) as avg_rental_price from books group by category LIMIT 0...	7 row(s) returned

19. Identify the branches where the average salary of employees is below Rs. 45,000.

The screenshot shows a database query editor with a SQL query and its results. The query is as follows:

```
301  
302  
303  
304 #Identify the branches where the average salary of employees is below Rs. 45,000.  
305 • select branch_no, avg(salary) as avg_sal from employee group by branch_no having avg(salary) < 45000;  
306  
307
```

The results are displayed in a table with the following columns: branch_no, avg_sal.

branch_no	avg_sal
1	40800.0000
2	44444.4444
4	40600.0000
5	44200.0000

The results are also displayed in the 'Output' section, showing the execution time and the number of rows returned for each query.

#	Time	Action	Message	Duration
62	17:02:51	select branch_no, avg(salary) as avg_sal from employee group by branch_no having avg(sal...	0 row(s) returned	0.000 sec
63	17:03:08	select branch_no, avg(salary) as avg_sal from employee group by branch_no having avg(sal...	4 row(s) returned	0.000 sec

20. List the books that have not been issued by any customer.

The screenshot shows a database management interface with a SQL editor and a results pane. The SQL editor contains a query to find books not issued by any customer. The results pane displays a table with 9 rows of book information.

```
304
305  #List the books that have not been issued by any customer.
306 •  select b.isbn,b.book_title from books b left join
307      issuestatus i on b.isbn = i.isbn_book
308      where i.isbn_book is null;
309
```

isbn	book_title
ISBN:7803	THE HISTORY OF LOVE
ISBN:781400	THE SECRET HISTORY
ISBN:87055	A BRIEF HISTORY
ISBN13:978006	THE ALCHEMIST
ISBN13:978030	THE DAVINCI CODE
ISBN13:978059	HARRY POTTER
ISBN13:978145	STEVE JOBS
ISBN13:978152	BECOMING
ISBN13:978349	THE HUNGER GAMES

Result 28 x

Output

Action Output

#	Time	Action	Message
✓ 44	16:51:47	select branch_no,avg(salary) as avg_sal from employee group by branch_no having avg(sal...	4 row(s) returned
✓ 45	16:52:51	select b.isbn,b.book_title from books b left join issuestatus i on b.isbn = i.isbn_book where i.i...	9 row(s) returned

21. Which publisher has the highest number of books in the library.

The screenshot shows a database management interface with a SQL editor and a results pane. The SQL editor contains the following query:

```
309
310 #Which publisher has the highest number of books in the library.
311 • select publisher, count(*) as total_books
312   from books group by publisher
313   order by total_books desc
314   limit 3;
```

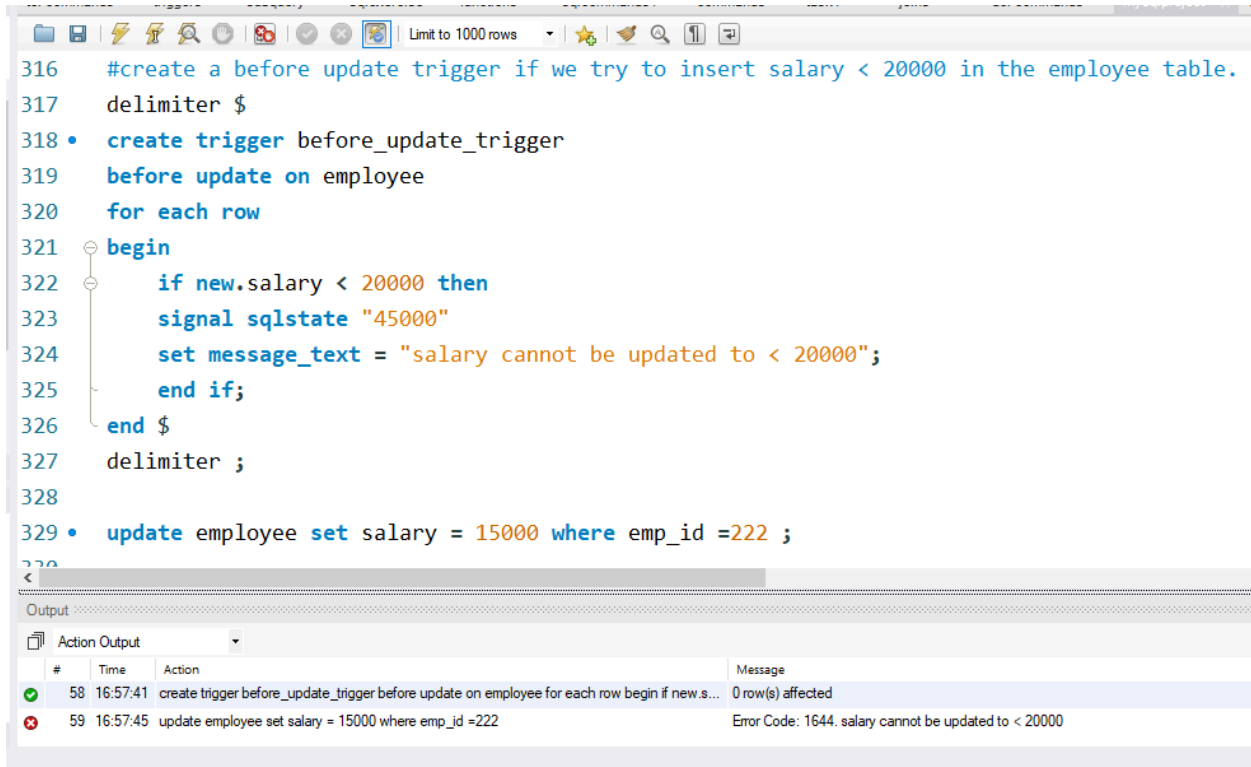
Below the editor, the "Result Grid" shows the following data:

	publisher	total_books
▶	DC BOOKS	4
	BANTAM	2
	ACE BOOKS	2

The "Output" pane shows the execution log:

#	Time	Action	Message
✓ 45	16:52:51	select b.isbn, b.book_title from books b left join issuestatus i on b.isbn = i.isbn_book where i.i...	9 row(s) returned
✓ 46	16:54:14	select publisher, count(*) as total_books from books group by publisher order by total_books ...	3 row(s) returned

22.create a before update trigger if we try to insert salary < 20000 in the employee table.

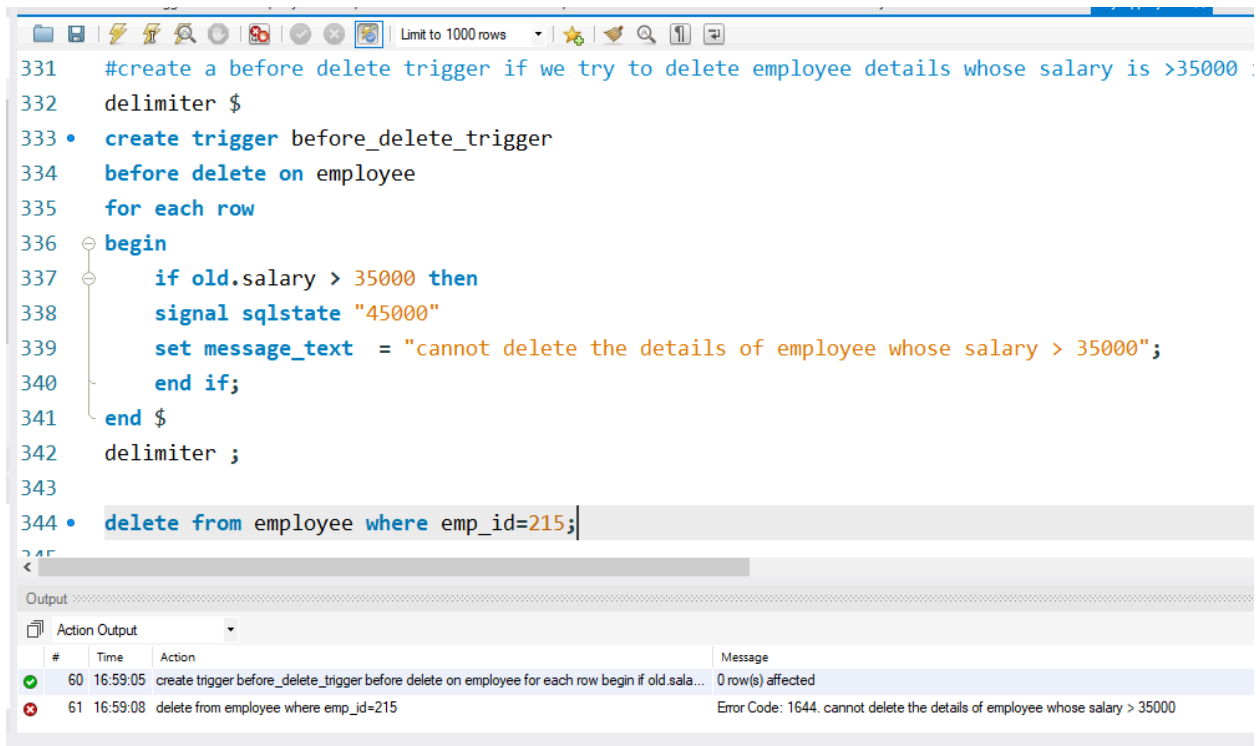


```
316 #create a before update trigger if we try to insert salary < 20000 in the employee table.
317 delimiter $
318 • create trigger before_update_trigger
319 before update on employee
320 for each row
321 begin
322     if new.salary < 20000 then
323         signal sqlstate "45000"
324         set message_text = "salary cannot be updated to < 20000";
325     end if;
326 end $
327 delimiter ;
328
329 • update employee set salary = 15000 where emp_id =222 ;
330
```

Output

#	Time	Action	Message
✓ 58	16:57:41	create trigger before_update_trigger before update on employee for each row begin if new.s...	0 row(s) affected
✗ 59	16:57:45	update employee set salary = 15000 where emp_id =222	Error Code: 1644. salary cannot be updated to < 20000

23.create a before delete trigger if we try to delete employee details whose salary is >35000 in the employee table.



```
331 #create a before delete trigger if we try to delete employee details whose salary is >35000
332 delimiter $
333 • create trigger before_delete_trigger
334 before delete on employee
335 for each row
336 begin
337     if old.salary > 35000 then
338         signal sqlstate "45000"
339         set message_text = "cannot delete the details of employee whose salary > 35000";
340     end if;
341 end $
342 delimiter ;
343
344 • delete from employee where emp_id=215;
```

Output

#	Time	Action	Message
60	16:59:05	create trigger before_delete_trigger before delete on employee for each row begin if old.sala...	0 row(s) affected
61	16:59:08	delete from employee where emp_id=215	Error Code: 1644. cannot delete the details of employee whose salary > 35000