Library Database Management System

by Syrymbet Nuray 200109032

https://github.com/Nuray-web/DBMS-library - link to access tables.

a) List of project topics to choose from:

i) Introduction and database description.

A Library Management System is a software built to handle the primary housekeeping functions of a library. Libraries rely on library management systems to manage asset collections as well as relationships with their members. Library management systems help libraries keep track of the books and their checkouts, as well as members' including such information as reports, deliveries, borrow-return history and others. It allows for use of its functions right to facilitate end users' process of taking service.

ii) What functions should the system perform? For example, inventory control, billing, ordering, etc.

The database is the project to the website of the public library in a small town, which collects the information about its readers (users of the website library system), their book borrow-return history, calculations on fine related to made issue, delivery requests, notifications about some events in library or remind to return the book on time, etc. It should implement new rows according to certain conditions, filter the permissions to workers and other relatable things.

- Any library member should be able to search books by their title, author, subject category as well by the publication date.
- Each book will have a unique identification number and other details including copies.
- There could be more than one copy of a book, and library members should be able to check-out and reserve any copy.
- The system should be able to retrieve information like who took a particular book or what are the books checked-out by a specific library member.

- The system should be able to send notifications whenever the reserved books become available, as well as when the book is not returned within the due date.

iii) Who are the end users? Remember that the DBA is NOT an end user.
The end users are the readers who use the application or website to reserve, share books, request deliveries and other activities. Also it will facilitate the working process of employees, it would be much easier to search through the database to get information about books, approve the reservations, manage deliveries and notifications, develop the book sharing events by recommending different genres.

iv) How will data obsolescence be handled?

The useless information gonna be filtered from the database, and inactive but potentially needed data will be separated from active one to the tables or databases and saved for long-term usage.

v) Where did you get the idea for this project? Did you make it up, get it from work, or find it in a book? Please mention your sources. The idea may NOT be something solved in a book, nor may it be a simple add-on to an existing database.

The standard thing about library database was took from:

- https://www.freeprojectz.com/entity-relationship/library-management-system-er-diagram

- https://www.geeksforgeeks.org/er-diagram-of-library-management-system/#:~:text=ER%20Diagram%20is%20known%20as,provides%20a%20means%20of%20communication.

- https://123projectlab.com/er-diagram-for-library-management-system/

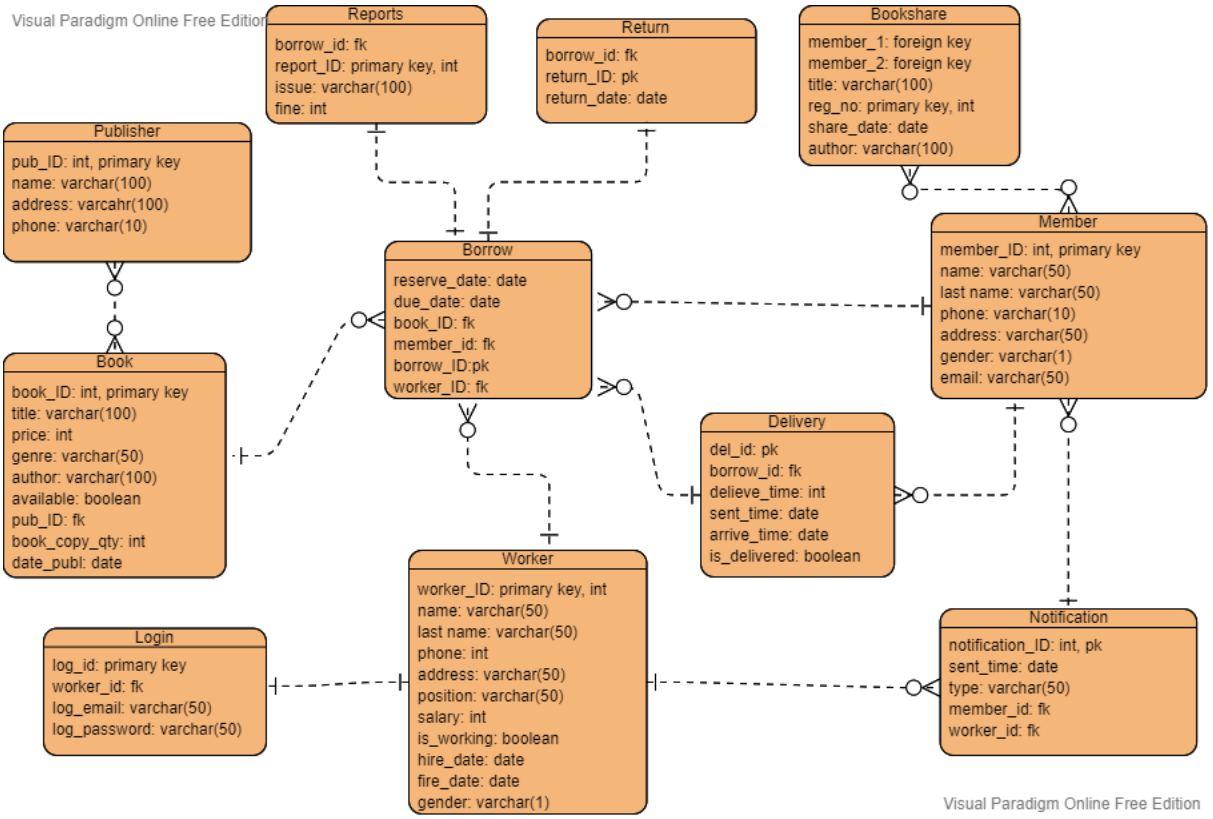
Using the sources above we got the idea of implementing 'events' related to the Library like the 'book sharing' process, where people

can swap books and get in contact with each other for future purposes.
Also for the people who are passionate about paper books but don't really have time to visit the library , we added the delivery system, which works only in the area of the town. And a notification system which will notify the readers about the due date to return the books, the reports on book issues, latest updates in the library system or events that are going to be held.

b) Entity Relationship Design:

Link to online ERD:
https://online.visual-paradigm.com/w/mhevfnbc/diagrams/#diagram:workspace=mhevfnbc&proj=0&id=1&type=ERDiagram

i) Draw the ER Diagram for your database.



Description of process:

- **Publisher** (that has a name, address and phone) publishes the book at date_publication. One publisher can publish many books. One book can be published by many publishers. (Depend on edition and translation)

- **Book** has its own title, author who wrote it, price it was bought at, status if there are any available editions or copies, genre, amount of copies and also the book has the mark of publisher. One book can be borrowed by many readers.
- **Members** (Readers) have the information about their name, last name, phone, address, gender. One member can borrow many books.
- **Workers** have the information about their name, last name, gender, phone, address, position (job title), salary for their job, status (if they're still working or not), hire_date and fire_date (if they don't work in library anymore, if they work it's just empty row). One worker can manage many reports, many borrows and many notifications, but each of them can be done only by one worker.
- **Report** is written on some certain book that was particularly damaged after usage by some member, so the worker that spotted the issue identifies the borrow_id (book_id, member_id in it), worker_id, damage type and fine. Fine is determined by the table below:

written pages	10000
mold	20000
torn pages	20000
water damage	10000
missing pages	15000
lost book	20000
damage outside	10000

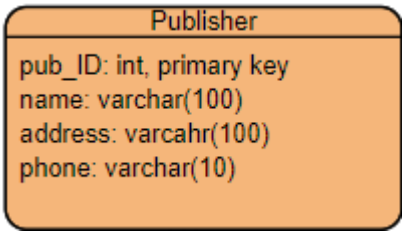
- **Borrow** is the table about process of taking book, member can anytime reserve the available books he/she wants on the site, it will also show the due_date it should be returned, book_id (that is gonna be reserved), member_id (who reserved), worker_id (who is responsible to approve the reservation).
- **Return** is the table about returnings history of books. There is borrow_id information and date of return.
- **Delivery** - any member can request the delivery of his/her books to their address. It contains the information about borrow, approximate time of delivery, sending item time and arriving_time, is it delivered and received by the reader.
- **Notification** - all the members get notifications of some certain events from the library like bookshare event, delivery, book return due date, etc. It has the sending date, type of the notification, the member who it should be sent and the worker who sends the info.
- **Bookshare** - the constant, independent library event which is not totally regulated, but stored as the data. It has the two acting members (giver and receiver), book, which includes the title and

author (we consider the situation that the book is the property of the member and maybe not stored in our library) and share date.

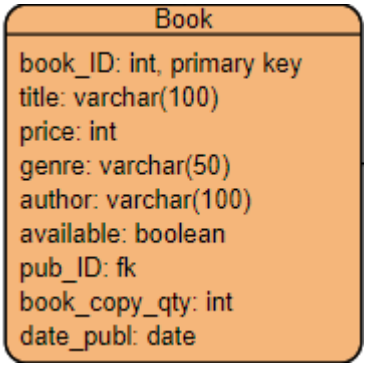
- **Login -** table contains information about all active and currently working employees, login email and login password to access the database.

Database explanation of each entity.

ii) Describe your entities. Be sure to define the meaning of each attribute. You must describe the "role" each attribute will play in your table (i.e. what is it and who will use it). You must have enough entities to ensure your project is not a "toy" system.

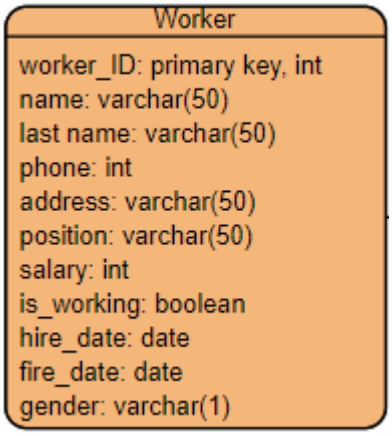


Publisher table - gonna be used by workers to define the type of book translation or edition if the customer is interested in particular Publisher or to request a new party of books from the certain publisher's product:
pub_ID - int, auto_increment, unique, not null, primary key
name - the name of publisher company, varchar(100), not null
phone - the phone number to contact the company, varchar(10), not null
address - location of company to get the delivery from, varchar(100), not null

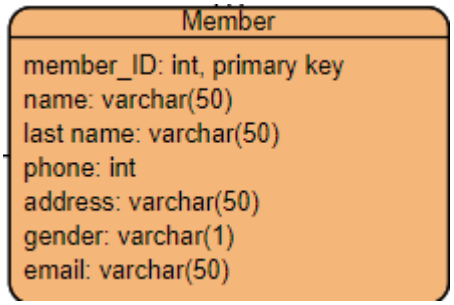


Books table - gonna be used by the workers as much as by readers, so the users could keep track of the books, what kind of genre it is, written by which author, the price the book was bought, if it is available to reserve, the quantity that in charge:
book_id - int, auto_increment, unique, not null, primary key

pub_id - foreign key (publishers)
title - title of the book, varchar(100), not null
author - name, surname of author, varchar(100), not null
date_pub - date of book publication, date
price - price of book, int, not null
genre - genre of the book, varchar(50), not null
book_copy_qty - the number of book copies, int, not null
available- if the book is available or not, boolean, not null

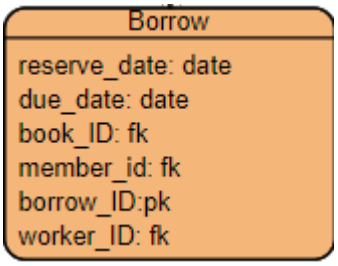


Worker table - mostly gonna be used by employers to keep track of the workers, pay them salaries, contact them if anything happens:
worker_ID - int, auto_increment, unique, not null, primary key
name - name of worker, varchar(50)
last name - last name of worker, varchar(50)
phone - phone number, int, not null
address - currently living location of member, varchar(100), not null
position - job status in library, varchar(100), not null
salary - int
status - active/inactive on job vacancy, boolean
hire_date - date of appointment to the job, date, not null
leave_date - date of getting fired, date, default = null
gender - varchar(1)

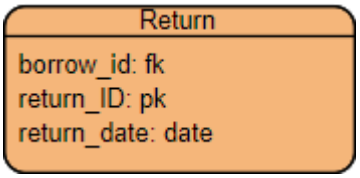


Member table - gonna be used by employees who will work with readers, approve their reservations, deliveries requests, notify for

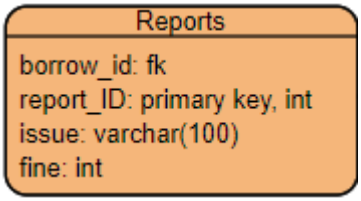
something important like due date coming, library indoor events like book sharing, etc :
member_id - int, auto_increment, unique, not null, primary key
name - name of member, varchar(100), not null
last name - last name of member, varchar(100), not null
phone - phone number, int, not null
address - currently living location of member, varchar(100), not null
status - active/inactive, boolean, not null
gender - boolean, not null
email - varchar(50)



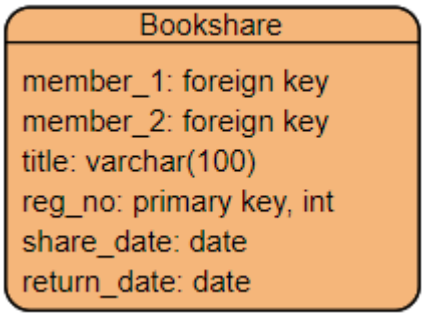
Borrow table - gonna be used by workers to store the borrow history where which book was gone, to which member and approved by which worker at what date:
borrow_id - primary key, not null
book_id - foreign key
member_id - foreign key
worker_id - foreign key
reserve_date - date that book was taken from library, date
due_date - date that book has to be returned to library, date



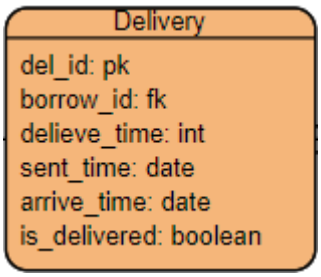
Return table - still gonna be used by workers to get information about if the book was returned to library by certain member:
return_id - primary key, int
borrow_id - foreign key, int
return_date - date that book was returned to library, date



Reports table - used by employees to store information about books state and quality it was given and received after usage:
report_id - primary key, int
book_id - foreign key, int
member_id - foreign key, int
issue - what kind of damage was given to the book, varchar(100)
fine - amount of money people should pay to library for returning book later than due date, int
worker_id - worker found the issue, int, foreign key

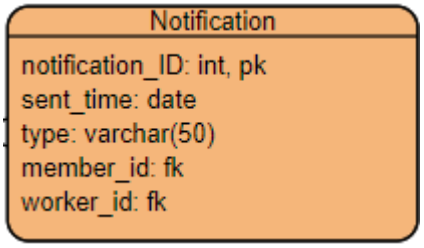


Bookshare table - gonna be used by both workers and readers, it will track the process of swapping their books and keep it informed about by whom the book was taken and by whom was given if the owner would like to return it or taker would like to reach out the owner for future shares and get in contact on base of similar interests:
member_1 - person who gave the book for bookshare, foreign key
member_2 - person who took the book, foreign key
title - title of the book, varchar(100)
reg_no - registration number of share, primary key
share_date - date for book_sharing, date

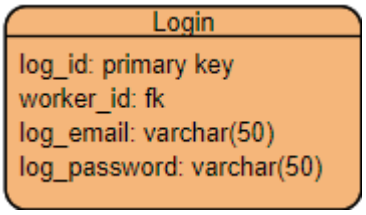


Delivery table - gonna be used by readers to get the book they want without going to library and gonna be used by workers to track where it is going, remark the sending time, the approximate time, arriving date and delivery approval:
del_id - primary key, not null, int
borrow_id - foreign key (borrow)

delive_time - amount of time to deliver (in hours), int
sent_time - time of sending, date
arrive_time - time of arriving, date
is_delivered - yes/no, boolean
member_id - receiver, foreign key



Notification table - gonna be used by workers to inform users which are readers about latest changes or updates or dues to do something, contains the time and type of message:
notification_id - primary key, int, not null
sent_time - time of sending notification, date
type - topic of notification, varchar(50)
member_id - foreign key
worker_id - foreign key



Login table - used by workers to log in into the system of LIbrary DBMS to provide the secure access:
log_id - primary key, int, not null, auto_increment
worker_id - foreign key (worker)
log_email - the user name workers use to access the DBMS and workspace chat
log_password - password to log in the system

iii) Describe your relationships and their type. Be sure to defend your choices. You may wish to give an example to illustrate your choice.

Possible relations

- One book can be borrowed by many readers;
- One member can borrow many books;
- One member can request many deliveries;

- One delivery can be requested by one member;
- Many borrows can be delivered at once;
- One borrow can be managed by one worker;
- One worker can manage many borrows;
- One worker can send many notifications;
- One notification can be sent only by one worker;
- One worker can manage many reports;
- One report can be done by only one worker;
- One worker can have one login username and password;
- One login information can be owned by one worker;

c) Normalization:

i) Show main functional dependencies,keys and superkeys, relationships and subclasses if any, etc.

1) Publishers:

pub_id → {name, address, phone}
pub_id → name
pub_id → address
pub_id → phone

2) Books:

book_id → {title, author, genre, price, available, pub_id, qty, date_pub}
book_id → {title, author, genre, price}
book_id → {pub_id, qty, date_pub}
book_id —> available
{title, author} → {genre, price}
pub_id → {genre, price, qty, date_pub}

3) Workers:

worker_id → {first_name, last_name, gender, phone, address, position, salary, hire_date, fire_date, is_working}
worker_id → {first_name, last_name, gender, phone, address}
worker_id → {position, salary, hire_date, fire_date, is_working}

4) Members:

member_id → {first_name, last_name, gender, phone, address, email}

5) Borrow:

borrow_id → {book_id, worker_id, member_id, reserve_date, due_date}
{book_id, worker_id, member_id} → {reserve_date, due_date}

6) Reports:
report_id → {borrow_id, issue, fine}
issue → fine

7) Returns:
return_id → {borrow_id, return_date}
borrow_id → {return_date}

8) Delivery:
del_id → {borrow_id, delieve_time, sent_time, arrive_time, is_delivered}
delieve_time → {sent_time, arrive_time}

9) Notification:
not_id → {sent_time, type, member_id, worker_id}

10) Login:
log_id → {worker_id, log_email, log_password}

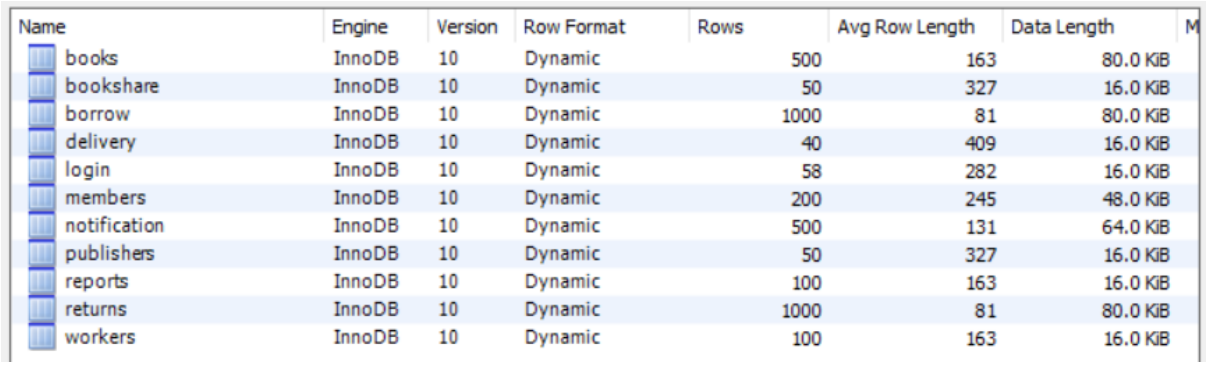
11) Bookshare:
reg_no → {member_give, member_take, title, author, share_date}
reg_no → {member_give, member_take, sahre_date}
reg_no → {title, author}

ii) Your database should be in Boyce-Codd Normal form.

Every relation contains only atomic values → it is in 1NF.
All relations are in 1NF and all non-key attributes are fully functional dependent on the primary key → it is in 2NF.
All relations are in 2NF and no transition dependency exists → it is in 3NF.
The tables are in 3NF form, and primary keys are candidate keys or super keys → it is in BCNF.

d) Physical Design (physical is like table in database)

i) How many tables are required:
- If 1 person in a team: minimum 10 tables;
- Fill with data. Data should be filled with enough records(each table should contain at least 100 tuples). It may be fake (You can use Mockaroo to generate fake data).



Name	Engine	Version	Row Format	Rows	Avg Row Length	Data Length	M
books	InnoDB	10	Dynamic	500	163	80.0 KiB	
bookshare	InnoDB	10	Dynamic	50	327	16.0 KiB	
borrow	InnoDB	10	Dynamic	1000	81	80.0 KiB	
delivery	InnoDB	10	Dynamic	40	409	16.0 KiB	
login	InnoDB	10	Dynamic	58	282	16.0 KiB	
members	InnoDB	10	Dynamic	200	245	48.0 KiB	
notification	InnoDB	10	Dynamic	500	131	64.0 KiB	
publishers	InnoDB	10	Dynamic	50	327	16.0 KiB	
reports	InnoDB	10	Dynamic	100	163	16.0 KiB	
returns	InnoDB	10	Dynamic	1000	81	80.0 KiB	
workers	InnoDB	10	Dynamic	100	163	16.0 KiB	

e) Requirements for Query part:
i) Your project should have at least 20 queries related to the business process of your project.
- Query should not be simple
- Query must be exactly solving some problem

1) Check-out for the borrow history for last year:

```
select m.first_name, m.last_name, m.email, m.address,
bw.reserve_date, b.title, b.author, b.genre
from final_project.members m
join final_project.borrow bw using (member_id)
join final_project.books b using (book_id)
where reserve_date >= '2022-01-01';
```

2) Check-out the late book returns for past year:

```
select * from (select m.first_name, m.last_name, m.address,
bw.reserve_date, b.title, b.author,  bw.due_date, r.return_date,
if(r.return_date>bw.due_date, datediff(r.return_date, bw.due_date), 0)
as late_sub
from final_project.members m
join final_project.borrow bw using (member_id)
join final_project.books b using (book_id)
join final_project.returns r using (borrow_id)
where bw.reserve_date >= '2022-01-01'
order by bw.reserve_date desc) c
where c.late_sub > 0;
```

3) **Check-out for the publishers book qty grouped by genre:**

```
select p.name, b.genre, sum(b.book_copy_qty) as amount
from final_project.books b
join final_project.publishers p using (pub_id)
group by p.name, b.genre
order by p.name, amount;
```

4) **First trigger** is the inserting of new employees to the authentication system, it does insert the worker_id, email and password.

Trigger after insert:

```
CREATE DEFINER=`root`@`localhost` TRIGGER `workers_AFTER_INSERT`
AFTER INSERT ON `workers` FOR EACH ROW BEGIN
        insert into final_project.login(worker_id, log_email, log_password)
    values (new.workers_id, concat(new.first_name, '.', new.last_name,
'@elibrary.com'),
    concat(left(upper(new.first_name),3), new.workers_id,
right(lower(new.last_name), 3),
    new.workers_id, right(upper(new.last_name),2)));
END
```

Insertion:

```
insert into final_project.workers(first_name, last_name, gender, phone,
address, position, salary, is_working, hire_date, fire_date)
values ('Nuray', 'Quartz', 'F', '705-739-9032', 'Konayev', 'Manager', 75000, 1,
'2022-12-12 03:38:16', null);
```

5) **Second trigger** is the deleting the login information and permissions from authentication system

```
CREATE DEFINER=`root`@`localhost` TRIGGER
`workers_AFTER_UPDATE` AFTER UPDATE ON `workers` FOR EACH
ROW BEGIN
if old.is_working != new.is_working then
        delete from final_project.login
    where worker_id = old.workers_id;
end if;
END
```

example check:

```
update final_project.workers
```

```
set is_working = 0
where workers_id = 101;
```

6) **Third trigger** adds the notification after reader borrowed the book

```
CREATE DEFINER=`root`@`localhost` TRIGGER
`borrow_AFTER_INSERT` AFTER INSERT ON `borrow` FOR EACH ROW
BEGIN
        insert into notification (sent_time, member_id, worker_id, type)
    values (now(), new.member_id, new.worker_id, 'book is borrowed');
END
```

check:

```
insert into borrow(member_id, worker_id, book_id, reserve_date,
due_date)
values (12, 101, 8, now(), now()+interval 30 day);
```

7) **Check-out the quantity frequency of notifications grouped by gender and type of the message**

```
select m.gender, n.type, count(type) amount_not from
final_project.members m
join final_project.notification n using (member_id)
group by m.gender, n.type
order by amount_not desc;
```

8) **Stats by years and months the distribution of borrowed and returned books**

```
(select year(reserve_date) Years, monthname(reserve_date) Months,
count(*) Qty, 'Borrowed' Descrip from final_project.borrow bw
group by Years, Months
order by Years, Months)

union all

(select year(return_date) Years, monthname(return_date) Months,
count(*) Qty, 'Returned' Descrip from final_project.returns r
group by Years, Months
order by Years, Months);
```

9) **View** - separated table for book title, author, qty, price and total value

```
create view qty_price as
select title, author, book_copy_qty, price, price*book_copy_qty as
total
from final_project.books;
```

10) **Check the most written genres**

```
select b.genre, count(*) freq from final_project.books b
group by genre
having freq = (select max(freq) from
(select b.genre, count(*) freq from final_project.books b
group by genre) a)
```

11) **Check most active reader who borrows more books**

```
select bw.member_id, m.first_name, m.last_name, count(*) freq from
final_project.borrow bw
join final_project.members m using (member_id)
group by bw.member_id
having freq = (select max(freq) from
(select bw.member_id, count(*) freq from final_project.borrow bw
group by bw.member_id) a)
```

12) **Readers who have borrowed books more than 10 times**

```
select bw.member_id, m.first_name, m.last_name, count(*) freq from
final_project.borrow bw
join final_project.members m using (member_id)
group by bw.member_id
having freq >= 10;
```

13) **Average salary on library job position**

```
select position, gender, salary, avg(salary) average
from final_project.workers
group by position, gender
order by average desc, gender, position;
```

14) Check on delivery status

```
select title, author, first_name, last_name, d.is_delievered from
final_project.delivery d
join final_project.borrow bw using (borrow_id)
join final_project.books b using(book_id)
join final_project.members m using(member_id);
```

15) Most published author

```
select b.author, count(*) freq from final_project.books b
group by author
having freq = (select max(freq) from
(select b.author, count(*) freq from final_project.books b
group by author) a)
```

16) Money spent on borrowed books by authors

```
select b.author, sum(price*book_copy_qty) circulation from
final_project.books b
where book_id in (select book_id from borrow where reserve_date
>= '2020-01-01')
group by author
order by circulation desc;
```

17) Earliest publishings of authors, who wrote more than 20 books

```
select author, min(date_publ) from books
group by author
having count(*)>=20;
```

18) Transaction on adding a new member!

```
start transaction;
select * from members;
insert into members(first_name, last_name, phone, address,
gender, email)
values ('Zhanna', 'Dark', '705-705-7050','Kaskelen', 'F',
'zhannathedark@gmail.com');
insert into notification(sent_time, type, member_id, worker_id)
```

```
values (now(), 'new member!', (select max(member_id) from
members), 101);
commit;
```