Queries

- Sorting

```
SELECT Title
FROM book
ORDER BY title ASC
```

```
SELECT Title, pubYear
FROM book
ORDER BY pubYear DESC
```

- Not returned books

```
SELECT Title
FROM book
WHERE book.ISBN= (
    SELECT ISBN
    FROM borrows
    WHERE memberID=' ' AND date_of_return IS NULL
    );
```

- Books/category

```
SELECT categoryName, COUNT(ISBN)
FROM belongs_to
GROUP BY categoryName;
```

- Sorting

```
SELECT title,numpages
FROM Book
ORDER by numpages DESC
```

- Μέλη με δανεισμένα βιβλία

```
Select member.MFirst,member.MLast, borrows.date_of_borrowing, borrows.ISBN
FROM member
join borrows on member.memberId = borrows.memberId
```

- Μέλη με δανεισμένα βιβλία και τα δανεισμένα βιβλία με τα στοιχεία τους

```
Select member.MFirst,Mlast,book.title,date_of_borrowing
from member
join borrows on member.memberid = borrows.memberID
join book on book.ISBN = borrows.ISBN
```

- Μέλη και αριθμός βιβλίων που έχουν δανειστεί

```
Select MFirst,MLast,count(*) BookCount
from members
join borrows on member.memberId = borrows.memberId
group by member.memberId,MFirst,MLast
order by BookCount
```

- Συγκερκιμένο ISBN πόσες φορές δανειστεί

```
Select title, count(*)
from borrows as bo,book as b
where bo.ISBN=b.ISBN
group by bo.ISBN
HAVING bo.ISBN='{$ISBN}'
```

- Πίνακας για αποθέματα βιβλίων

```
CREATE table sumbooks
AS SELECT DISTINCT ISBN, COUNT(CopyNr) as CopyNum
FROM copies
GROUP BY ISBN
```

- Όψη μη ενημερώσιμη που κρατάει από τα βιβλία που έχουμε εισάγει ποια δεν έχουν επιστραφεί

```
Creat vew b as
SELECT ISBN
From borrows
Where borrows.date_of_return IS NULL
```

- Πίνακας που κρατάει τον αριθμό των αντιτύπων των δανεισμένων βιβλίων

```
Create table borrowed as
SELECT s.ISBN,
CopyNum - 1 as leftovers
FROM sumbooks AS s
JOIN b ON b.ISBN= s.ISBN
```

- Ανανέωση πίνακα που κρατάει όλα τα βιβλία και τη διαθεσιμότητά τους

```
UPDATE sumbooks
JOIN borrowed ON sumbooks.ISBN=borrowed.isbn
SET sumbooks.copynum= borrowed.leftovers
```

- Πόσες φορές έχει δανειστεί το κάθε βιβλίο

```
SELECT title, count(*) as Popularity
FROM borrows AS bo,book as b
WHERE bo.ISBN=b.ISBN
GROUP BY bo.ISBN
ORDER BY title
```

- Αριθμός υπαλλήλων κάθε κατηγορίας

```
SELECT
(SELECT
        COUNT(temporary_employee.empID) from temporary_employee) as temporary,
(Select COUNT(permanent_employee.empID) from permanent_employee) as permanent

SELECT AVG(salary)
from employee
```

- Ταξινόμηση

```
SELECT ISBN,title,pubYear,numpages, pubName
FROM book
ORDER BY Title ASC

SELECT ISBN,title,pubYear,numpages, pubName
FROM publisher
ORDER BY numpages DESC


SELECT ELast, Efirst, empID, salary
FROM employee
ORDER BY Elast ASC

SELECT ELast, Efirst, empID
FROM book
ORDER BY empID ASC
```

Triggers

```
DELIMITER $$

CREATE TRIGGER overdue
BEFORE INSERT ON borrows
FOR EACH ROW
BEGIN
IF (  ((SELECT COUNT(*) FROM borrows WHERE memberID = new.memberID AND date_of_return IS
NULL AND DATEDIFF(CURRENT_TIMESTAMP,date_of_borrowing) > 30) > 0) OR ((SELECT
COUNT(*) FROM borrows WHERE memberID = new.memberID AND date_of_return IS NULL) >= 5) OR
((SELECT copyNum FROM sumbooks WHERE ISBN = new.ISBN) < 1))
THEN SIGNAL SQLSTATE "45000"
    SET MESSAGE_TEXT = "Member Cannot Borrow Book";
END IF;
```

```
END $$
DELIMITER ;

DELIMITER $$

CREATE TRIGGER diagrafimelous
BEFORE DELETE ON member
FOR EACH ROW
BEGIN
IF (SELECT COUNT(*) FROM borrows WHERE memberID = old.memberID AND date_of_return IS
NULL) > 0
THEN SIGNAL SQLSTATE "45000"
    SET MESSAGE_TEXT = "You can’t delete this member because he/she has to return a book";
END IF;
END $$
DELIMITER ;
```

VIEWS

```
CREATE VIEW employee_view AS
SELECT
ELast, Efirst, empID
FROM employee

CREATE VIEW permanent_view AS
SELECT employee.ELast,employee.EFirst, permanent_employee.HiringDate
FROM employee JOIN permanent_employee ON employee.EmpID = permanent_employee.EmpID


CREATE VIEW temporary_view AS
SELECT employee.ELast,employee.EFirst, temporary_employee.ContractNr
FROM employee JOIN temporary_employee ON employee.EmpID = temporary_employee.EmpID
ORDER BY Elast ASC
```