**Description:**

Library:

A library is a collection of organized information and resources which is made accessible to a well-defined community for borrowing or reference sake. The collection of the resources and information are provided in digital or physical format in either a building/room or in a virtual space or even both. Library’s resources and collections may include newspapers, books, films, prints, maps, CDs, tapes, videotapes, microform, database etc.

This Database is Library management system of such a library that has several branches in different cities with employees, where customers can borrow (issue) books that are available in the library. The database provides some information about a book’s author and its publisher. It keeps track of a book’s issue date, return date, and salary of the library’s employees.

**Schemas/Tables in the Library Database:**

\* Author (ID, Name, Gender, Qualification)

\* Publisher (ID, Name, City, Zip)

\* Books (ID, Name, Name, ISBN, Author\_ID, Publisher\_ID, No\_of\_Books)

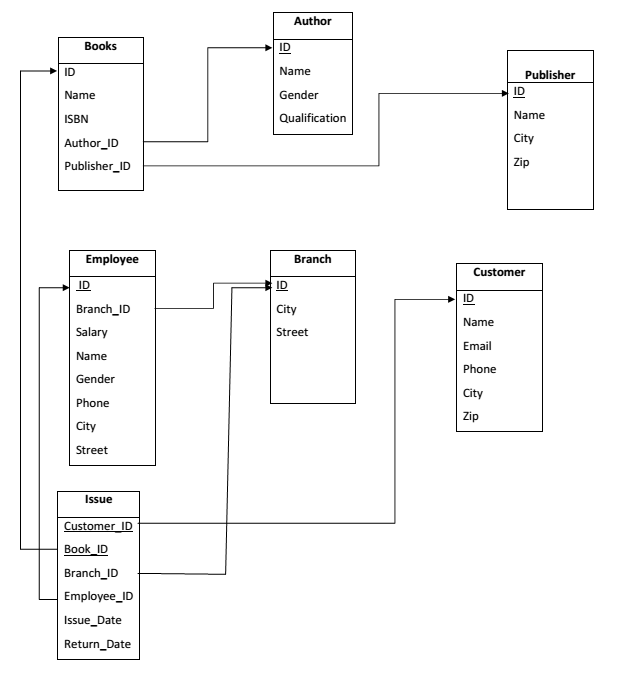
\* Branch (ID, City, Street)

\* Employee (ID, Branch\_ID, Name, Salary, Gender, Phone, City, Street)

\* Customer (ID, Name, Email, Phone, City, Zip)

\* Issue (Customer\_ID, Book\_ID, Branch\_ID, Employee\_ID, Issue\_Date, Return\_Date)

**Schema Diagram for Library Database:**



**Creating the Tables and Inserting the Data:**

1. **Author:**

create table Author

(

ID integer,

Name varchar(100) constraint "Author\_Name\_CHK\_NL" not null,

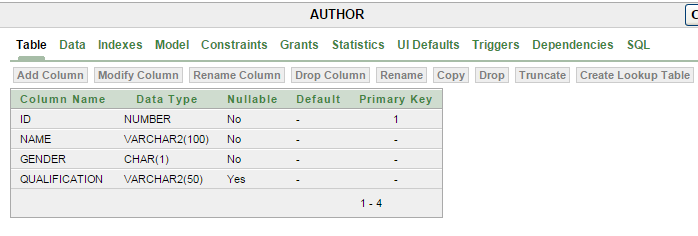
Gender char(1) constraint "Author\_Gender\_CHK\_NL" not null,

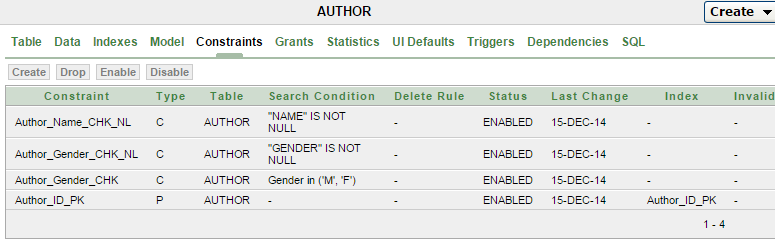
Qualification varchar(50),

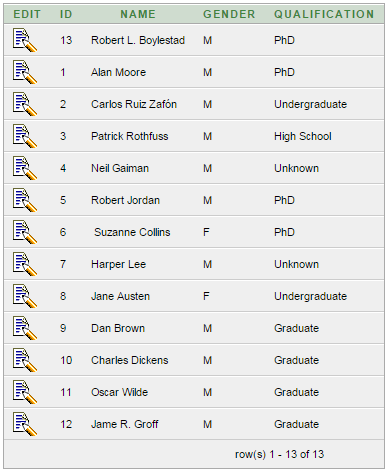
constraint "Author\_Gender\_CHK" check (Gender in ('M', 'F')),

constraint "Author\_ID\_PK" primary key (ID)

);

****

****

****

1. **Publisher:**

CREATE TABLE "PUBLISHER"

( "ID" integer,

"NAME" VARCHAR2(100) CONSTRAINT "Publisher\_Name\_CHK\_NL" NOT NULL ENABLE,

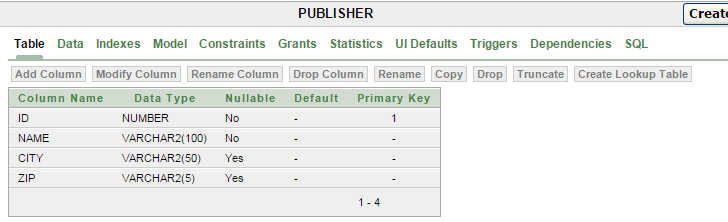
"CITY" VARCHAR2(50),

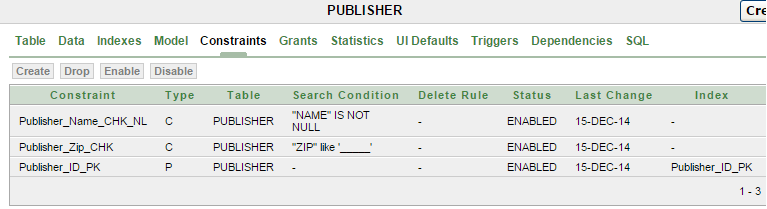
"ZIP" VARCHAR2(5),

CONSTRAINT "Publisher\_ID\_PK" PRIMARY KEY ("ID") ENABLE,

CONSTRAINT "Publisher\_Zip\_CHK" CHECK ( "ZIP" like '\_\_\_\_\_') ENABLE

);







1. **Books:**

create table Books

(

ID integer,

Name varchar(100),

ISBN varchar(30) constraint "Books\_ISBN\_CHK1" not null,

Author\_ID numeric(10),

Publisher\_ID numeric(10),

No\_of\_Books numeric(10, 0),

constraint "Books\_No\_Of\_Bks\_CHK2" check (No\_of\_Books >= 0),

constraint "Books\_ISBN\_CHK" check (ISBN like '%-%-%-%'),

constraint "Books\_ISBN\_UK" unique (ISBN),

constraint "Books\_Author\_ID\_FK" foreign key (Author\_ID)

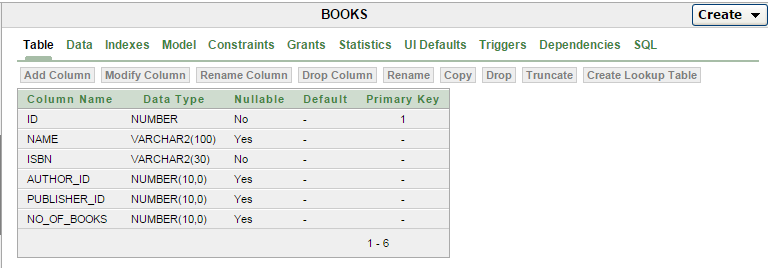
references Author(ID) on delete set null,

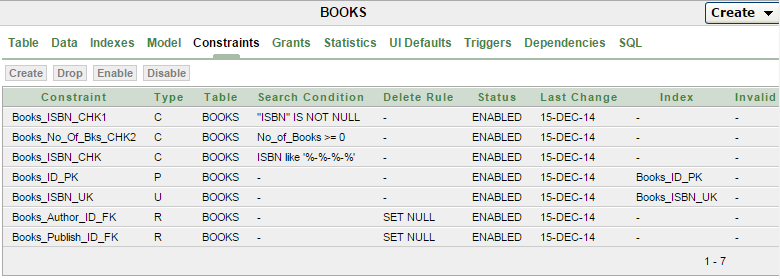
constraint "Books\_Publish\_ID\_FK" foreign key (Publisher\_ID)

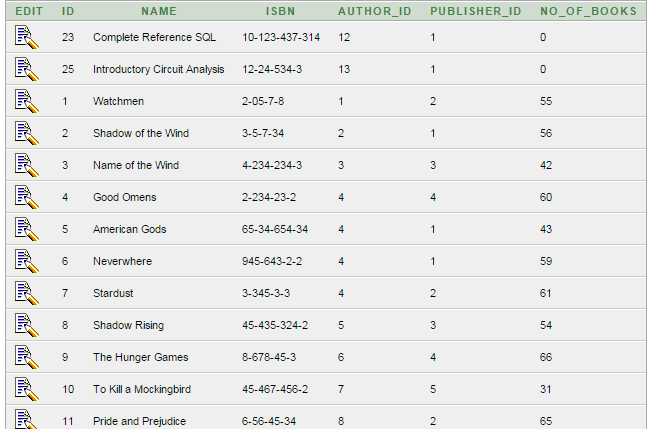
references Publisher(ID) on delete set null,

constraint "Books\_ID\_PK" primary key (ID)

);







25 rows.

1. **Branch:**

create table Branch

(

ID varchar(3),

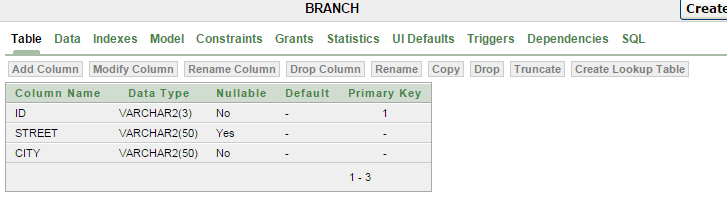
Street varchar(50),

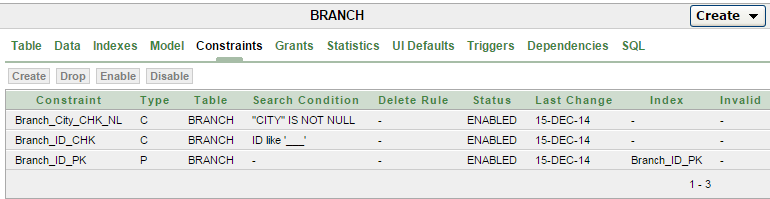
City varchar(50) constraint "Branch\_City\_CHK\_NL" not null,

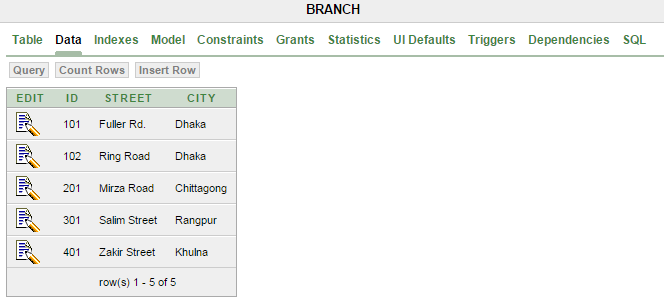
constraint "Branch\_ID\_PK" primary key (ID),

constraint "Branch\_ID\_CHK" check (ID like '\_\_\_')

);







1. **Employee:**

create table Employee

(

ID integer,

Branch\_ID varchar(3) constraint "Employee\_Br\_ID\_CHK" not null,

Salary numeric(10, 2),

Name varchar(100) constraint "Employee\_Dt\_Name\_CHK\_NL" not null,

Gender char(1) constraint "Employee\_Dt\_Gender\_CHK\_NL" not null,

Phone varchar(11) constraint "Employee\_Dt\_Phone\_CHK\_NL" not null,

City varchar(50) constraint "Employee\_Dt\_City\_CHK\_NL" not null,

Street varchar(50),

constraint "Employee\_Salary\_CHK" check (Salary >= 2000),

constraint "Employee\_Branch\_ID\_FK" foreign key (Branch\_ID)

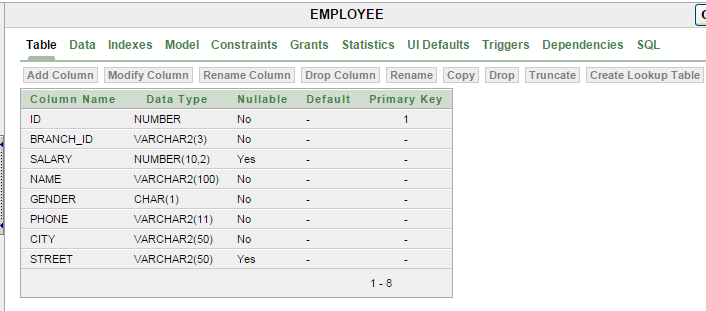
references Branch(ID) on delete cascade,

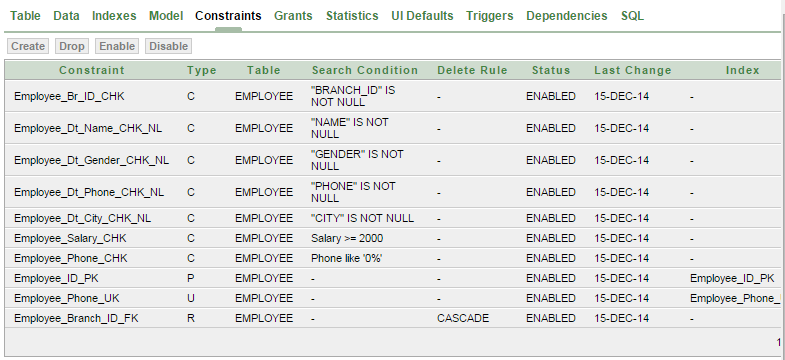
constraint "Employee\_Phone\_CHK" check (Phone like '0%'),

constraint "Employee\_Phone\_UK" unique (Phone),

constraint "Employee\_ID\_PK" primary key (ID)

);







1. **Customer:**

create table Customer

(

ID integer,

Name varchar(100) constraint "Customer\_Name\_CHK\_NL" not null,

Email varchar(100),

Phone varchar(11),

City varchar(50),

Zip varchar(4),

constraint "Customer\_Zip\_CHK" check (Zip like '\_\_\_\_\_'),

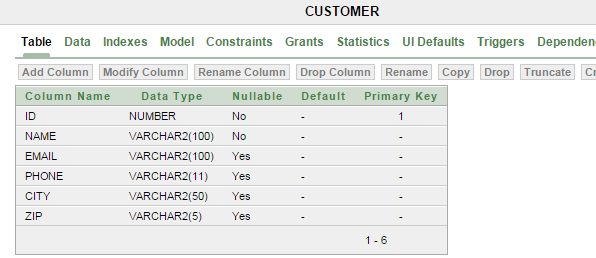
constraint "Customer\_Phone\_CHK" check (Phone like '0%'),

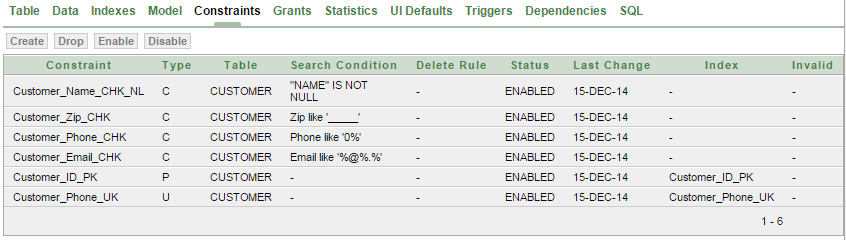
constraint "Customer\_Phone\_UK" unique (Phone),

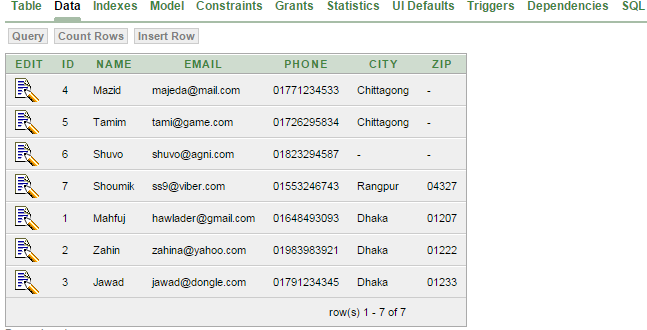
constraint "Customer\_Email\_CHK" check (Email like '%@%.%'),

constraint "Customer\_ID\_PK" primary key (ID)

);

****

****

****

1. **ISSUE:**

create table Issue

(

Customer\_ID integer constraint "Issue\_Cust\_ID\_CHK" not null,

Book\_ID integer constraint "Issue\_BK\_ID\_CHK" not null,

Branch\_ID varchar(3) constraint "Issue\_Br\_ID\_CHK" not null,

Employee\_ID integer constraint "Emp\_ID\_CHK" not null,

Issue\_Date date,

Return\_Date date,

constraint "Issue\_Cust\_ID\_FK" foreign key (Customer\_ID)

references Customer(ID),

constraint "Issue\_Book\_ID\_FK" foreign key (Book\_ID)

references Books(ID),

constraint "Issue\_Branch\_ID\_FK" foreign key (Branch\_ID)

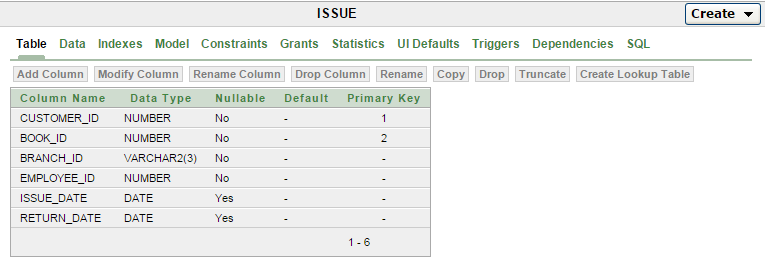
references Branch(ID),

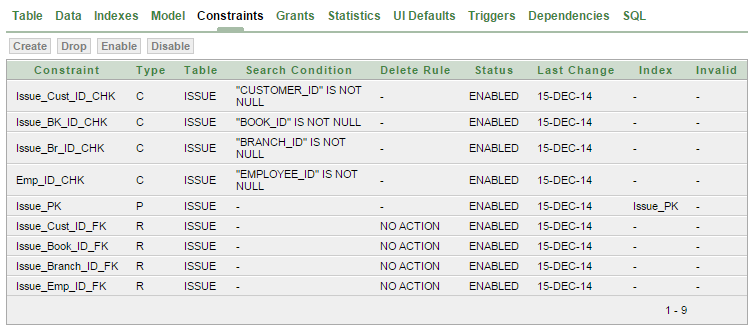
constraint "Issue\_Emp\_ID\_FK" foreign key (Employee\_ID)

references Employee(ID),

constraint "Issue\_PK" primary key (Customer\_ID, Book\_ID)

);







**Queries:**

1. ID, name, and phone of employees who work Dhaka shown in descending order of their names:

select A.ID, A.Branch\_ID, A.NAME, A.PHONE

from EMPLOYEE A, BRANCH B

where A.BRANCH\_ID = B.ID and LOWER(B.CITY) = 'dhaka'

order by A.NAME desc



1. Details of those customers who use grameen phone and live in Dhaka:

select \*

from customer

where phone like '017%' and lower(city) = 'chittagong'



1. Number of books issued by those employees who have a salary greater than 5000:

select A.Employee\_ID, count((A.BOOK\_ID))

from ISSUE A

group by A.Employee\_ID

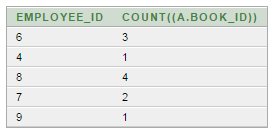
having A.Employee\_ID in (

select B.ID

from Employee B

where B.salary > 5000

)



1. Customer ID, Customer Name and Book ID, Book Name, Author Name, Issue date and Return date of those customers and the books they have issued.

select customer\_ID, customer.name "Customer Name", book\_ID, books.name "book name",

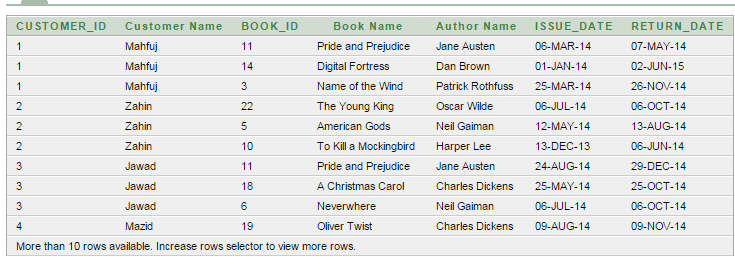
author.name "Author Name", Issue.issue\_date, Issue.Return\_date

from ((issue inner join customer

on customer\_ID = customer.id) inner join books on book\_ID = Books.id)

inner join author on books.author\_id = author.ID

order by customer.ID



1. Book\_ID issued between between '1-jan-2014' to '31-dec-2014' :

select distinct(book\_ID)

from ISSUE

where issue\_date between '1-jan-2014' and '31-dec-2014'

order by book\_id



1. ID, name and salary of those employees whose salary is greater than average salary of all employees.

select ID, name, salary

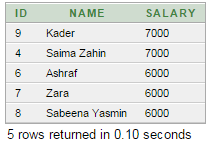
from employee

where salary > (

select avg(salary)

from employee

)



1. Book\_ID and Name of those books that were issued whose publisher name starts with the characters ‘mc’

select Book\_ID, books.name

from issue inner join books on issue.Book\_id = Books.id

where Book\_ID in (

select B.ID

from Books B, Publisher

where B.Publisher\_ID = Publisher.ID and upper(Publisher.name) like (upper('Mc%'))

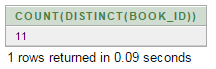
)



1. Number of different books that have been issued:

select count(distinct(Book\_ID))

from issue



1. Branch\_ID and City name of those Braches whose avg(salary) > the salary of atleast one female employee:

select B.ID, B.city

from branch B inner join Employee E on B.ID = E.Branch\_ID

group by B.ID, B.city

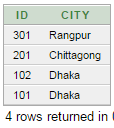
having avg(salary) > some(

select salary

from Employee Y

where Y.Gender = 'F'

)



1. Branch\_ID and City name of those Braches whose avg(salary) > the salary of all employee:

select B.ID, B.city

from branch B inner join Employee E on B.ID = E.Branch\_ID

group by B.ID, B.city

having avg(salary) > all(

select salary

from Employee Y

where Y.Gender = 'F'

)



1. Check if no book with book\_ID = 25 has been issued and then show details of all issues involved with customer\_ID = 1

select \*

from issue

where not exists

(

select \*

from issue

where book\_id = 25

) and customer\_ID = 1



1. Information about those boooks that have "l" in its name and whose no of books is not 0

select \*

from books A

where lower(A.name) like '%l%' and exists

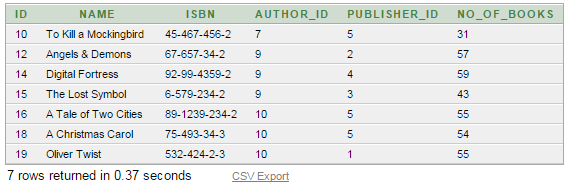
(

select \*

from books B

where A.name = B.name and no\_of\_books <> 0

)



1. Cities in branch or employee table:

select city from branch

union

select city from employee



1. Increase salary by 1000 for those employees whose salary is < average salary of all employees:

update employee

set salary = salary+1000

where salary < (

select avg(salary)

from employee

)



1. Delete those books that have an ID > 15 and has never been issued.

delete from books

where ID > 15 and ID not in (

select BOOK\_ID

from issue

)



1. Employee ID, name, and number of books sold by each employee whose salary > 5000:

with tmp as(

select A.Employee\_ID EMP, count((A.BOOK\_ID)) sold

from ISSUE A

group by A.Employee\_ID

having A.Employee\_ID in (

select B.ID

from Employee B

where B.salary > 5000

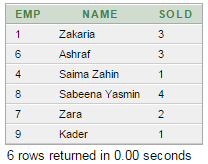
)

)

select EMP, name, sold

from employee, tmp

where employee.ID = tmp.EMP



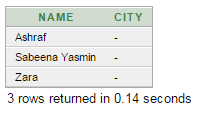
1. Those employees whose cities don’t match with Branch cities:

select distinct(employee.name), branch.city

from employee left join BRANCH

on employee.city = branch.city

where branch.city is null



**Discussions:**

This is a sample database. There are some limitations. It requires many improvements. I will add more facilities to this database in the near future, to make the database more user-friendly and efficient.