**SQL**

create database Phase1Evaluation

go

use Phase1Evaluation

go

create table authorTable(

authorId int not null primary key,

authorName varchar(40)

)

go

create table member(

memberId int not null primary key,

memberName varchar(40)

)

go

create table publication(

publicationId int not null primary key,

memberName varchar(40)

)

go

create table genre(

genreId int not null primary key,

genreName varchar(40)

)

go

create table bookTable(

bookId int not null Primary key,

bookName varchar(40),

publicationId int foreign key references publication(publicationId),

genreId int foreign key references genre(genreId),

price int,

authorId int foreign key references authorTable(authorId)

)

go

create table rentTable (

rentId int not null primary key,

bookId int foreign key references bookTable(bookId),

rentPrice int,

bookTakenDate date,

BookReturnDate date

)

Go

insert into authorTable (authorId, authorName)

values

( 1,'Niall Ferguson'),

( 2,'Agatha Christie'),

( 3,'Arthur Conan Doyle'),

( 4,'Robert Ludlum')

Go

insert into member(memberID, memberName)

values

(01,'Abhinav'),

(02,'Rakesh'),

(03,'Peter'),

(04,'Parker'),

(05,'Bruce'),

(06,'Wayne')

Go

insert into publication(publicationId, publicationName)

values

(001,'Penguin'),

(002,'DC Books'),

(003,'Marvel')

Go

insert into genre(genreId, genreName)

values

(100,'Fiction'),

(101,'Non-fiction'),

(102,'Biography')

Go

insert into bookTable(bookId, bookName, publicationId, genreId, price, authorId)

values

(1000,'Empire',001, 100,1000,1),

(1002,'Spider-Man',003, 101,2000,3),

(1003,'Star-Wars',002, 102,500,2),

(1004,'Modern India',001, 101,1500,1),

(1005,'Spectrum',003, 102,800,4),

(1006,'James Bond',001, 101,1200,3),

(1007,'Indian Polity',003, 100,1400,4),

(1008,'Themes in Indian History',002, 102,200,2),

(1009,'Anthropology',001, 101,400,1)

Go

alter table rentTable

add memberId int foreign key references member(memberId)

Go

insert into rentTable (rentId, bookId, memberId, rentPrice, bookTakenDate, bookReturnDate)

values

(10000,1000,01,50,'2020-05-10','2020-05-25'),

(10001,1003,03,25,'2020-10-15','2020-10-25'),

(10002,1004,02,74,'2020-12-01','2020-12-20'),

(10003,1002,05,100,'2020-11-11','2020-11-25'),

(10004,1005,01,40,'2020-01-12','2020-01-30'),

(10005,1006,06,60,'2020-04-10','2020-04-30'),

(10006,1007,05,70,'2020-06-10','2020-06-26'),

(10007,1008,02,10,'2020-05-10','2020-05-25'),

(10008,1009,04,20,'2020-08-4','2020-08-24')

—---------------------------------1—---------------------------------------------------

1. List of books with book details, price and rental amount

—----------------

select distinct bookName, publicationName, authorName,price, rentPrice, genreName

from bookTable

join authorTable on bookTable.authorId = authorTable.authorId

join publication on bookTable.publicationId = publication.publicationId

join rentTable on bookTable.BookId = rentTable.BookId

join genre on bookTable.genreId = genre.genreId

—-------------output—--------

bookName publicationName authorName price rentPrice genreName

Anthropology Penguin Niall Ferguson 400 20 Non-fiction

Empire Penguin Niall Ferguson 1000 50 Fiction

Indian Polity Marvel Robert Ludlum 1400 70 Fiction

James Bond Penguin Arthur Conan Doyle 1200 60 Non-fiction

Modern India Penguin Niall Ferguson 1500 74 Non-fiction

Spectrum Marvel Robert Ludlum 800 40 Biography

Spider-Man Marvel Arthur Conan Doyle 2000 100 Non-fiction

Star-Wars DC Books Agatha Christie 500 25 Biography

Themes in Indian History DC Books Agatha Christie 200 10 Biography

—--------------------------2—------------------------------

2. List of all members who has fine and the amount

select bookName, memberName, day(bookReturnDate)-day(bookTakenDate)+10 as'fine amount'

from rentTable

join bookTable on bookTable.bookId = rentTable.bookId

join member on member.memberId = rentTable.memberId

group by bookName,memberName, BookReturnDate,bookTakenDate

having(day(bookReturnDate) - day(bookTakenDate)) >10

—--------------------------output—--------------

bookName memberName fine amount

Anthropology Parker 30

Empire Abhinav 25

Indian Polity Bruce 26

James Bond Wayne 30

Modern India Rakesh 29

Spectrum Abhinav 28

Spider-Man Bruce 24

Themes in Indian History Rakesh 25

—----------------------------3—------------------------------

3. No of books in each category

select count(genreName) as 'no of books in each genre', genreName

from bookTable

join genre on bookTable.genreId = genre.genreId

group by genreName;

—------output—------------

no of books in each genre genreName

3 Biography

2 Fiction

4 Non-fiction

—----------------------------------------**MONGODB**—-------------------------

use abhinav

switched to db abhinav

abhinav> db.collection.insertMany([{ "BookName": "Anthropology", "Author": "Niall Ferguson", "Publication": "Penguin", "price": 400, "rent": 20 },

{ "BookName": "Empire", "Author": "Niall Ferguson", "Publication": "Penguin", "price": 1000, "rent": 50 },

{ "BookName": "Indian Polity", "Author": "Laxmikanth", "Publication": "Marvel", "price": 1400, "rent": 70 },

{ "BookName": "James Bond", "Author": "Robert Ludlum", "Publication": "D C Books", "price": 1200, "rent": 60 }, { "BookName": "Modern India", "Author": "Niall Ferguson", "Publication": "Penguin", "price": 200, "rent": 20 }, { "BookName": "Star-Wars", "Author": "Arthur Conan Doyle", "Publication": "D C Books", "price": 100, "rent": 5 }, { "BookName": "Spider-Man", "Author": "Stan Lee", "Publication": "Marve;", "price": 300, "rent": 15 }])

{

acknowledged: true,

insertedIds: {

'0': ObjectId("61e28cea1d0be338043ae83c"),

'1': ObjectId("61e28cea1d0be338043ae83d"),

'2': ObjectId("61e28cea1d0be338043ae83e"),

'3': ObjectId("61e28cea1d0be338043ae83f"),

'4': ObjectId("61e28cea1d0be338043ae840"),

'5': ObjectId("61e28cea1d0be338043ae841"),

'6': ObjectId("61e28cea1d0be338043ae842")

}

}

abhinav> db.collection.find();

[

{

\_id: ObjectId("61e28cea1d0be338043ae83c"),

BookName: 'Anthropology',

Author: 'Niall Ferguson',

Publication: 'Penguin',

price: 400,

rent: 20

},

{

\_id: ObjectId("61e28cea1d0be338043ae83d"),

BookName: 'Empire',

Author: 'Niall Ferguson',

Publication: 'Penguin',

price: 1000,

rent: 50

},

{

\_id: ObjectId("61e28cea1d0be338043ae83e"),

BookName: 'Indian Polity',

Author: 'Laxmikanth',

Publication: 'Marvel',

price: 1400,

rent: 70

},

{

\_id: ObjectId("61e28cea1d0be338043ae83f"),

BookName: 'James Bond',

Author: 'Robert Ludlum',

Publication: 'D C Books',

price: 1200,

rent: 60

},

{

\_id: ObjectId("61e28cea1d0be338043ae840"),

BookName: 'Modern India',

Author: 'Niall Ferguson',

Publication: 'Penguin',

price: 200,

rent: 20

},

{

\_id: ObjectId("61e28cea1d0be338043ae841"),

BookName: 'Star-Wars',

Author: 'Arthur Conan Doyle',

Publication: 'D C Books',

price: 100,

rent: 5

},

{

\_id: ObjectId("61e28cea1d0be338043ae842"),

BookName: 'Spider-Man',

Author: 'Stan Lee',

Publication: 'Marve;',

price: 300,

rent: 15

}

]

—-------------------4—------------

4. Write MongoDB query to find all books where price is greater than 400 in descending order

db.collection.find({price:{$gt:400}}).sort({price:-1});

Output:

{

\_id: ObjectId("61e28cea1d0be338043ae83e"),

BookName: 'Indian Polity',

Author: 'Laxmikanth',

Publication: 'Marvel',

price: 1400,

rent: 70

},

{

\_id: ObjectId("61e28cea1d0be338043ae83f"),

BookName: 'James Bond',

Author: 'Robert Ludlum',

Publication: 'D C Books',

price: 1200,

rent: 60

},

{

\_id: ObjectId("61e28cea1d0be338043ae83d"),

BookName: 'Empire',

Author: 'Niall Ferguson',

Publication: 'Penguin',

price: 1000,

rent: 50

}

5. Write MongoDB query to calculate the total rent amount by aggregating rental amount in all documents

db.collection.aggregate({ $match:{}}, {$group:{\_id:null,sum : {$sum:"$rent"} }})

[ { \_id: null, sum: 240 } ]

—-----------------------------------------**REST API DESIGN**—---------------------------------

Get Booklist

GET/api/books

Status : 200

ADD Member

POST/members/

{

name:XXXXXXX

phoneNumber:XXXXXXX

emailID:XXXXXXXXXX

Address:XXXXXXXXX

}

Status : 201

Update Booklist

POST/api/books

{}

Status: 200

Update profile

POST/api/members/{id}

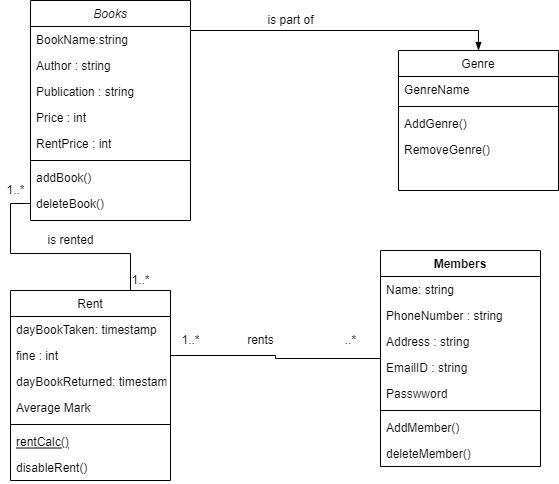
{

“name”:”ABHINAV”,

“phoneNumber”:”9009203”

}

Class Diagram



DB Design

