Project 1
in

Data Management

MIS 6326.001

MASTER OF SCIENCE
IN
INFORMATION TECHNOLOGY AND MANAGEMENT

GAYATRI SHARMA KURMATEY
GXK200021@UTDALLAS.EDU

Department of Information Technology and Management

Naveen Jindal School of Management

THE UNIVERSITY OF TEXAS AT DALLAS

800 W Campbell Rd, Richardson, TX 75080

Comet Creed

As a Comet, I pledge honesty, integrity, and service in all I do.

Abstract

Part I of the project involves design, development and implementation of the Database that could be used by client-server application (to be utilized by librarians).

High level functionality to be developed and executed:
1. Load initial data files (provided) using SQL Developer Import utility (no need to develop UI) into temporary tables. One temporary
table per file
2. Create DB tables (write and execute SQL statements, do not use UI) described in the high-level schema
3. Write SQLs to Normalize and populate data into tables in #2
4. Generate test data to simulate borrowing and fines transactions
5. Reports and Search SQL

All tables should be created by following naming convention below:
Temp/Initial load Table names should be as following:
1. project1_books_load
2. project1_book_copies_load
3. project1_borrowers_load
4. project1_library_branch_load
Final tables (see DB schema for full list of names):
1. project1_books
2. project1_book_authors
3. .....
4. project1_fines
Final tables **columns** should have names as specified in the DB design part.

INITIAL FILE LOADS

There are four files provided for the initial load. The files will be available on eLearning. By loading data into application tables, we will practice following important concepts:

- Ability to load various file formats into DB using SQL developer guided import interface
- Ability to understand “raw” data
- Ability to translate given data into normalized version and match application DB design

Files:

- Books file: Books information, such as ISBN, authors, title, etc.
- Book copies file: Number of copies of each book per library location
- Borrowers file: Borrowers’ information, such as name, SSN (dummy), address, etc.
- Library branches file: branches address information.

FUNCTIONAL REQUIREMENTS

Design and DB Architecture:

The initial import of the dataset is an administration task and can be performed by utilizing SQL Developer Import functionality while data normalization should be performed using SQL scripts.

Deliverables:
1. ER diagram
2. Database objects create statements: create table, foreign keys, indexes, etc.

Data Load, Normalization, data generation:

1. Load dataset files (provided with the project requirements) utilizing SQL Developer guided interface. Pay attention to the file format/delimiter, SQL Developer is able to recognize proper delimiter when selecting “text” in input file options.
2. Write SQL scripts to populate target application tables while normalizing the data loaded in #1.
3. Write SQL to generate:
   a) At least 500 books check-outs for at least 200 different borrowers and 100 different books
   b) At least 50 fines records for 20 different borrowers

Deliverables:

1. Initial load and normalized tables should be present in your DB account
2. SQL statements

Book Search and Availability:

Book search functionality should allow setting a specific library branch location context or performing it globally for all locations.
Write SQL that will provide a single search functionality to locate a book given any combination of ISBN, title, and/or Author(s). Your query
should support substring matching. For example, search for “will” should return results whose title include “will” or “willing”, or whose
author name contains “Will”, “Willy”, “William”, etc.

Important: The SQL statement should search normalized tables, don't search tables created to load original files. Search should display the following information for each book in the result set:

1) ISBN
2) Book title
3) All book author(s)
4) Availability status at the currently selected branch (if branch is selected)

Deliverables: SQL statements, Screen Shots with output of the search for 3 search inputs.

Reports:
Design and write SQL for 3 reports. The reports can accept input parameters, for example branch or date.

- Each report will have information presented from at least 3 tables
- Each report will have aggregation and accumulation functionality

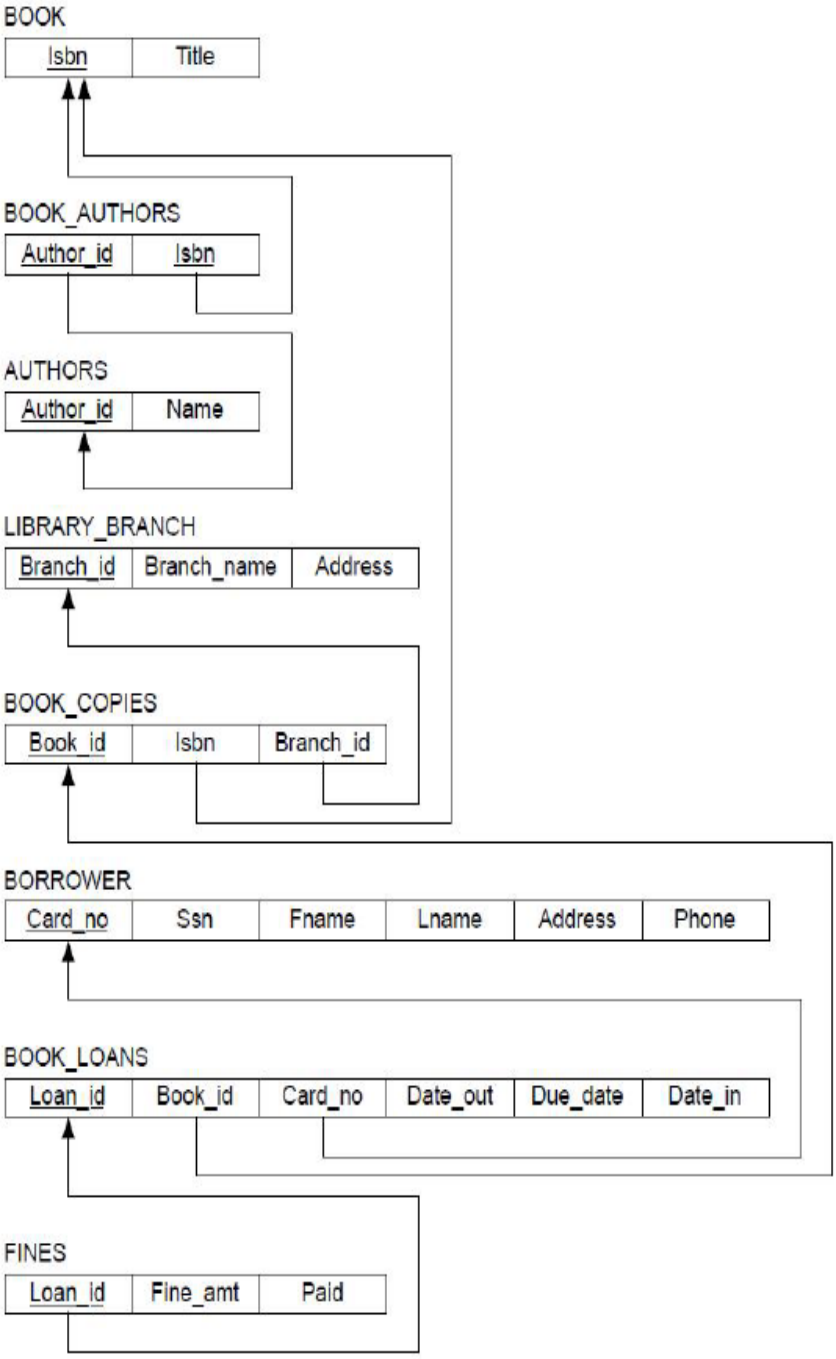
Examples of the reports (don't use example reports):

- Top 10 most popular books in the last month
- Top 10 books that were checked in late

Deliverables: SQL statement for each report

High Level Data Model

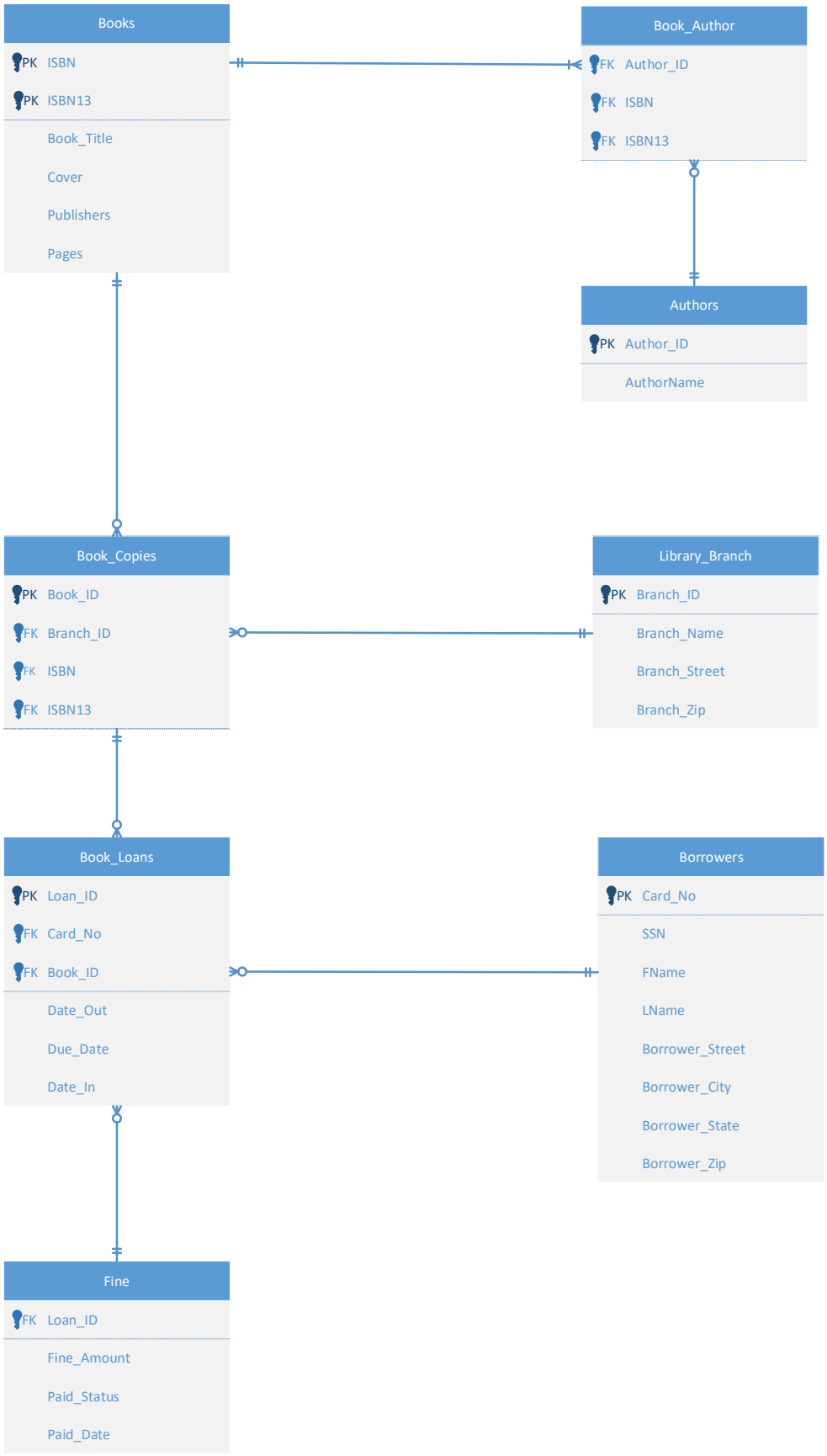
The high-level schema for this project is provided below.



SYSTEM ANALYSIS & ERD

Software Requirements:

- Operating System : Windows 10
- Technology : Oracle SQL
- Database : Oracle Database 19
- Graphic Application : MS Visio


Books
PK ISBN
PK ISBN13
Book_Title
Cover
Publishers
Pages
Book_Author
FK Author_ID
FK ISBN
FK ISBN13
Authors
PK Author_ID
AuthorName
Book_Copies
PK Book_ID
FK Branch_ID
FK ISBN
FK ISBN13
Library_Branch
PK Branch_ID
Branch_Name
Branch_Street
Branch_Zip
Book_Loans
PK Loan_ID
FK Card_No
FK Book_ID
Date_Out
Due_Date
Date_In
Borrowers
PK Card_No
SSN
FName
LName
Borrower_Street
Borrower_City
Borrower_State
Borrower_Zip
Fine
FK Loan_ID
Fine_Amount
Paid_Status
Paid_Date

DELIVERIES

CREATE TABLE STATEMENT

1. Creating project1_books table:

```
CREATE TABLE project1_books(ISBN varchar2(255), ISBN13 varchar2(255),
Title varchar2(255),
Cover varchar2(255), Publisher varchar2(255), Pages varchar2(255),
PRIMARY KEY (ISBN, ISBN13));
```

2. Creating project1_authors table:

```
CREATE TABLE project1_authors (Author_ID NUMBER(38) GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
Author_Name varchar2(255));
```

3. Creating project1_book_authors table:

```
CREATE TABLE project1_book_authors(Author_ID NUMBER(38),
ISBN VARCHAR2(255), ISBN13 VARCHAR2(255),
FOREIGN KEY (Author_ID) REFERENCES project1_authors(Author_ID),
FOREIGN KEY (ISBN,ISBN13) REFERENCES project1_books_load (ISBN10, ISBN13));
```

4. Creating project1_library_branch table:

```
CREATE TABLE project1_library_branch(Branch_ID varchar2(255) PRIMARY KEY,
Branch_Name varchar2(255),Branch_Street varchar2(255), Branch_Zip varchar2(255));
```

5. Creating project1_borrowers table:

```
CREATE TABLE project1_borrowers(Card_No varchar2(255), SSN varchar2(255),
FName varchar2(255), LName varchar2(255), Email varchar2(255),
Street varchar2(255), City varchar2(255),State varchar2(255),
Phone varchar2(255), PRIMARY KEY (Card_No));
```

6. Creating project1_book_copies table:

```
CREATE TABLE project1_book_copies (Book_ID NUMBER(38)generated always as identity,
Branch_ID varchar2(255), ISBN varchar2(255), ISBN13 varchar2(255),
PRIMARY KEY (Book_ID),
FOREIGN KEY (Branch_ID) REFERENCES project1_library_branch(Branch_ID),
FOREIGN KEY (ISBN, ISBN13) REFERENCES project1_books_load (ISBN10, ISBN13));
```

7. Creating project1_book_loans table:

```
CREATE TABLE project1_book_loans (Loan_ID NUMBER(38) GENERATED ALWAYS AS
IDENTITY,
Book_ID NUMBER(38), Card_No varchar2(255), Date_Out DATE, Due_Date DATE,
Date_In DATE,
PRIMARY KEY (Loan_ID),
FOREIGN KEY(Book_ID) REFERENCES project1_book_copies (Book_ID),
FOREIGN KEY (Card_No) REFERENCES project1_borrowers(CARD_NO));
```

8. Creating project1_fine table:

```
CREATE TABLE project1_fine (Loan_ID NUMBER(38), Fine_Amt NUMBER(7,2),
Paid varchar2(255), Paid_date varchar2(255),
FOREIGN KEY (Loan_ID) REFERENCES project1_book_loans(Loan_ID));
```

INSERTING VALUES INTO THE TABLE

9. Inserting into project1_books table:

```
INSERT INTO project1_books(ISBN, Title, ISBN13, Cover, Publisher, Pages)
SELECT ISBN10, Title, ISBN13, Cover, Publisher, Pages FROM Project1_books_load;
```

10. Inserting into project1_authors table:

```
INSERT INTO project1_authors(Author_Name)
select DISTINCT Author_Name from project1_books_names;
```

11. Inserting into project1_book_authors table:

```
INSERT INTO project1_book_authors (
SELECT  DISTINCT A.author_id, b.isbn10, b.isbn13
FROM project1_books_names b, project1_authors A
WHERE A.author_name=b.author_name);
```

12. Inserting into project1_library_branch table:

```
INSERT INTO project1_library_branch(Branch_ID, Branch_name, Branch_Street, Branch_Zip)
SELECT branch_id, branch_name, street1, zip1 FROM(
SELECT Branch_ID, Branch_name,
REGEXP_SUBSTR(address,'[^,]+',1,1)as street1,
REGEXP_SUBSTR(address,'[^,]+',1,2) as zip1
FROM project1_library_branch_load lb);
```

13. Inserting into project1_borrowers table:

```
INSERT INTO project1_borrowers(card_no, ssn, fname,
lname, email, street, city, STATE, phone)
SELECT * FROM project1_borrowers_load;
```

14. Inserting into project1_book_copies table:

```
 INSERT INTO project1_book_copies (branch_id, isbn, isbn13)
SELECT bcl.branch_id, bl.isbn10, bl.isbn13
FROM project1_book_copies_load bcl
INNER JOIN project1_books_load bl
ON bcl.book_id=bl.isbn10
WHERE bcl.no_of_copies='1';

INSERT INTO project1_book_copies (branch_id,isbn,isbn13)
SELECT bcl.branch_id, bl.isbn10, bl.isbn13
FROM project1_book_copies_load bcl
INNER JOIN project1_books_load bl
ON bcl.book_id=bl.isbn10
WHERE bcl.no_of_copies='2';

INSERT INTO project1_book_copies (branch_id,isbn,isbn13)
SELECT bcl.branch_id, bl.isbn10, bl.isbn13
FROM project1_book_copies_load bcl
INNER JOIN project1_books_load bl
ON bcl.book_id=bl.isbn10
WHERE bcl.no_of_copies='2';
```

15. Inserting into project1_book_loans table:

```
INSERT INTO project1_book_loans(book_id, card_no)
SELECT bc.*,pb.*
FROM
(SELECT *
FROM (
SELECT DISTINCT book_id
FROM project1_book_copies bc
ORDER BY dbms_random.random)
WHERE ROWNUM<101) bc,
(SELECT *
FROM (
SELECT DISTINCT card_no
FROM project1_borrowers pb
ORDER BY dbms_random.random)
WHERE ROWNUM<201) pb;

UPDATE project1_book_loans SET
date_out=trunc(SYSDATE + dbms_random.value(-365,365));

UPDATE project1_book_loans SET
due_date=trunc(date_out + 15),
date_in=trunc(date_out + dbms_random.value(2,30));
```

16. Inserting into project1_fine table:

```
INSERT INTO project1_fine (Loan_ID, Fine_Amt, Paid, Paid_date)
SELECT Loan_ID,
CASE WHEN (Date_In-Due_Date)*2 <0 THEN 0
ELSE (Date_In-Due_Date)*2 END,
CASE WHEN (Date_In-Due_Date)*2 <0 THEN NULL
ELSE (CASE WHEN dbms_random.value(0,1)>0.5 THEN 'PAID' ELSE 'NOT PAID' END)
END ,
CASE WHEN (Date_In-Due_Date)*2 <0 THEN NULL
ELSE TRUNC(date_in + dbms_random.value(0,30))
END
FROM project1_book_loans;
```

SELECT STATEMENTS

```
SELECT * FROM project1_books;
SELECT * FROM project1_authors;
SELECT * FROM project1_book_authors;
SELECT * FROM project1_library_branch;
SELECT * FROM project1_book_copies;
SELECT * FROM project1_borrowers;
SELECT * FROM project1_book_loans;
SELECT * FROM project1_fine;
```

Book Search and Availability

Book search functionality should allow setting a specific library branch location context or performing it globally for all locations.
Write SQL that will provide a single search functionality to locate a book given any combination of ISBN, title, and/or Author(s). Your query
should support substring matching. For example, search for “will” should return results whose title include “will” or “willing”, or whose
author name contains “Will”, “Willy”, “William”, etc.

Search should display the following information for each book in the result set:
ISBN
Book title
All book author(s)
Availability status at the currently selected branch (if branch is selected)

Query:

```
SELECT DISTINCT pb.isbn, pb.title, pa.author_name,
CASE
WHEN date_in > sysdate THEN 'not available'
WHEN date_in < sysdate THEN 'available'
WHEN bc.book_id NOT IN (SELECT book_id FROM project1_book_loans) THEN 'available'
END AS status_of_availability
FROM project1_book_copies bc, project1_book_loans bl, project1_books pb, project1_authors pa,
project1_book_authors ba
WHERE bc.book_id = bl.book_id
AND bc.isbn = pb.isbn
AND pb.isbn= ba.isbn
AND ba.author_id = pa.author_id
AND bc.branch_id =5;
```

Oracle SQL Developer : C:\Users\gxk200021\AppData\Roaming\SQL Developer\tabletestsql

File Edit View Navigate Run Source Team Tools Window Help

PROJCREATETABLEsql | projselect.sql | tabletestsql | Untitled16.sql | Untitled18.sql | Untitled19.sql | Untitled20.sql | Untitled21.sql

test user

Worksheet | Query Builder

```sql
SELECT DISTINCT pb.isbn, pb.title, pa.author_name,
CASE
WHEN date_in > sysdate THEN 'not available'
WHEN date_in < sysdate THEN 'available'
WHEN bc.book_id NOT IN (SELECT book_id FROM project1_book_loans) THEN 'available'
END AS status_of_availability
FROM project1_book_copies bc, project1_book_loans bl, project1_books pb, project1_authors pa, project1_book_authors ba
WHERE bc.book_id = bl.book_id
AND bc.isbn = pb.isbn
AND pb.isbn= ba.isbn
AND ba.author_id = pa.author_id
AND bc.branch_id =5;
```

Query Result | Script Output | Query Result 1

SQL | All Rows Fetched: 42 in 0.135 seconds

	ISBN	TITLE	AUTHOR_NAME	STATUS_OF_AVAILABILITY
1	0786707550	The Fear Sign (Allingham, Margery)	Margery Allingham	not available
2	0440188601	Time Out Of Joint	Philip K. Dick	available
3	0767902890	The Things They Carried	Tim O'Brien	not available
4	0312205309	Dr. Laura: The Unauthorized Biography	Vickie L. Bane	available
5	0486264785	The Importance Of Being Earnest (Dover Thrift Editions)	Oscar Wilde	not available
6	0767902890	The Things They Carried	Tim O'Brien	available
7	0376026138	Starbucks Passion For Coffee	Sunset Books	available

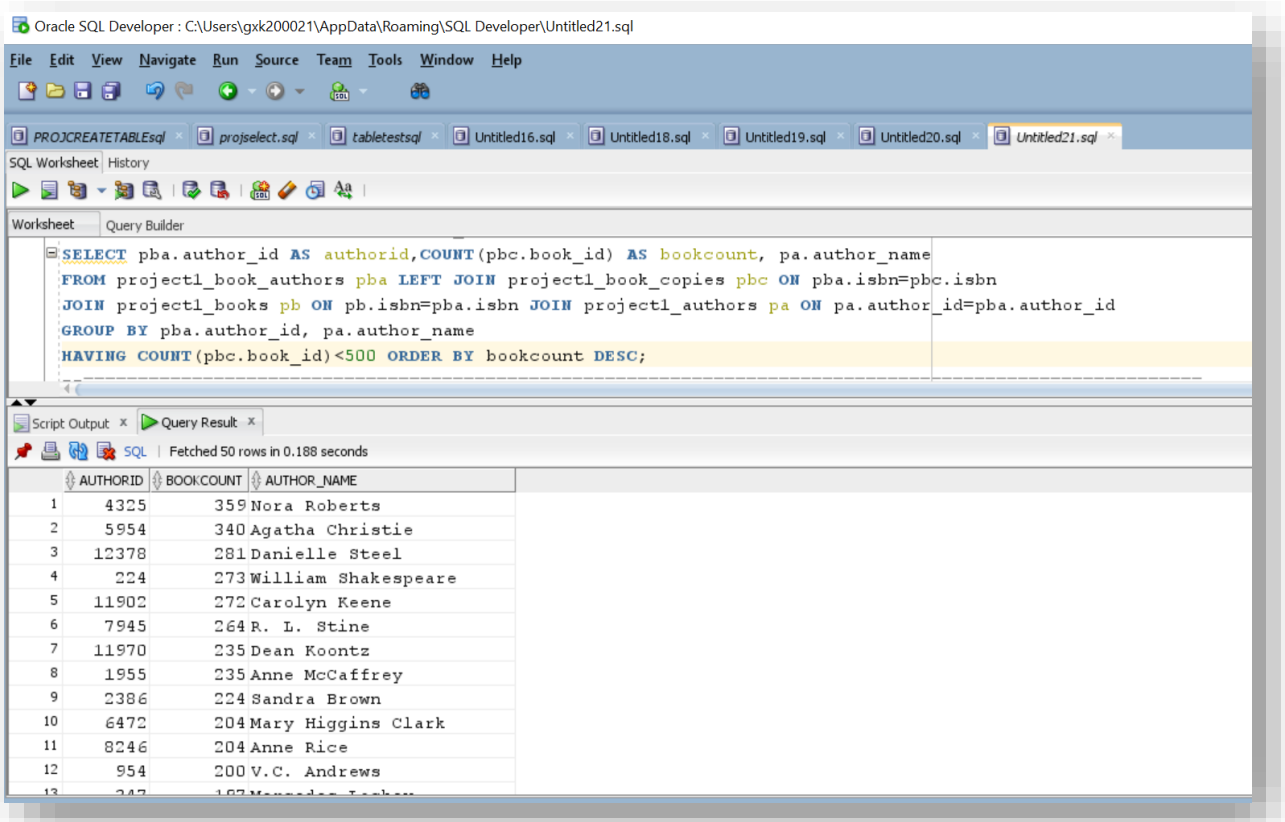
Requirement:

Design and write SQL for 3 reports. The reports can accept input parameters, for example branch or date.

- Each report will have information presented from at least 3 tables
- Each report will have aggregation and accumulation functionality

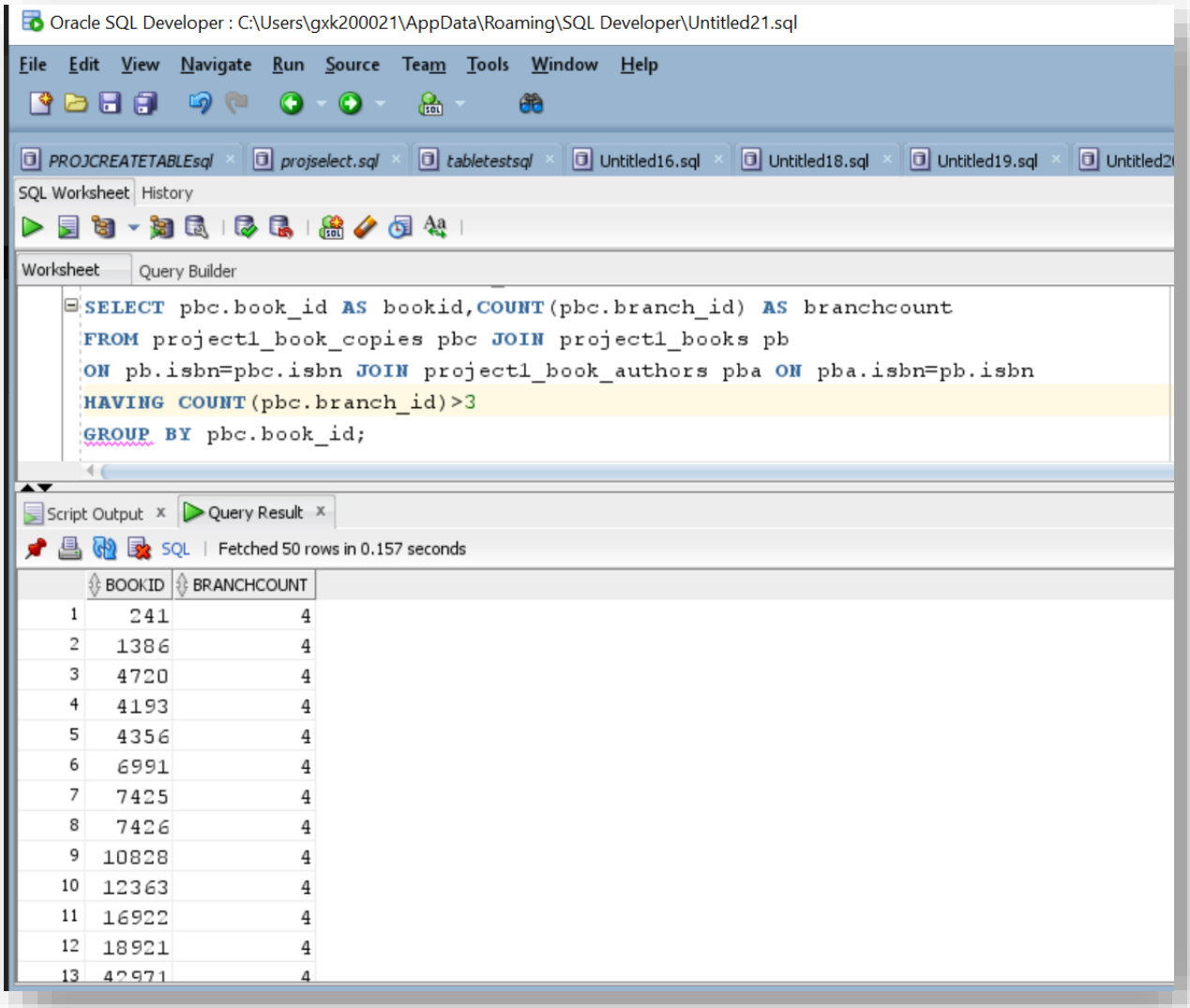
1) Author details whose count of book_ids are less than 500 books.

```
SELECT pba.author_id AS authorid,COUNT(pbc.book_id) AS bookcount, pa.author_name
FROM project1_book_authors pba LEFT JOIN project1_book_copies pbc ON pba.isbn=pbc.isbn
JOIN project1_books pb ON pb.isbn=pba.isbn JOIN project1_authors pa ON pa.author_id=pba.author_id
GROUP BY pba.author_id, pa.author_name
HAVING COUNT(pbc.book_id)<500 ORDER BY bookcount DESC;
```



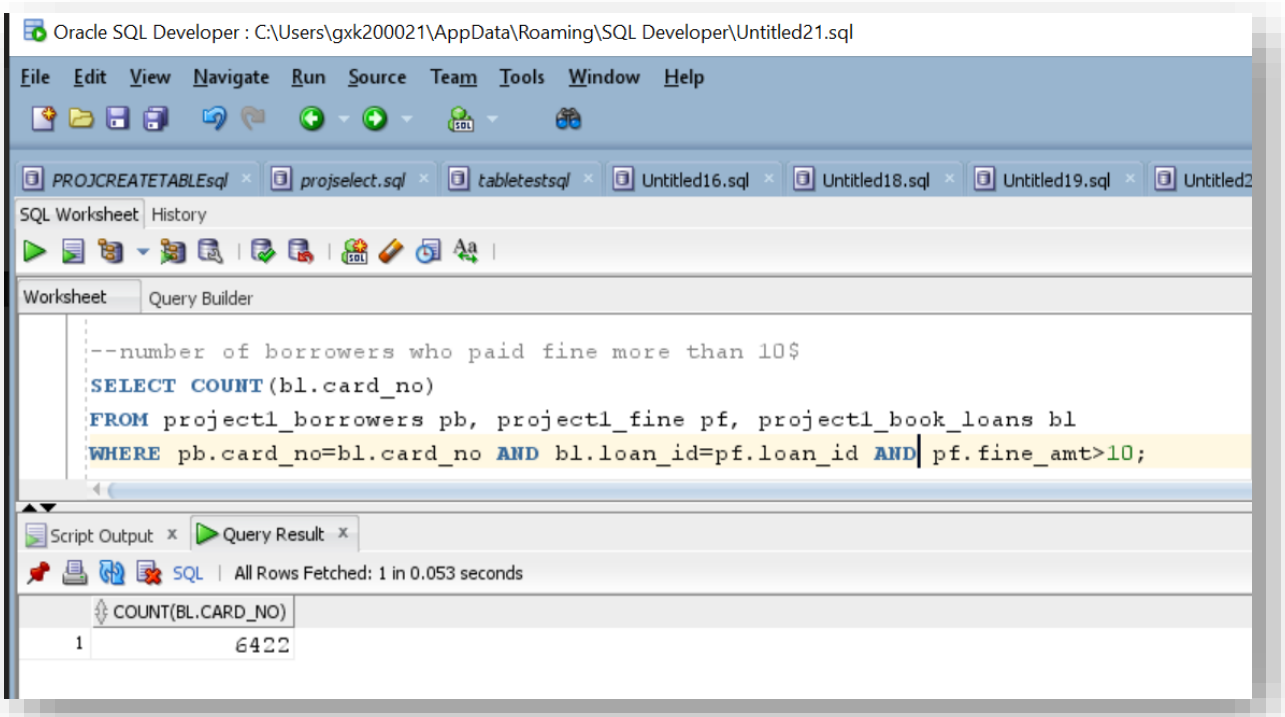
2) Number of book ids in branch IDs greater than 3.

```
SELECT pbc.book_id AS bookid,COUNT(pbc.branch_id) AS branchcount
FROM project1_book_copies pbc JOIN project1_books pb
ON pb.isbn=pbc.isbn JOIN project1_book_authors pba ON pba.isbn=pb.isbn
HAVING COUNT(pbc.branch_id)>3
GROUP BY pbc.book_id;
```



3) Number of borrowers who paid fine more than 10$.

```
SELECT COUNT(bl.card_no)
FROM project1_borrowers pb, project1_fine pf, project1_book_loans bl
WHERE pb.card_no=bl.card_no AND bl.loan_id=pf.loan_id AND pf.fine_amt>10;
```

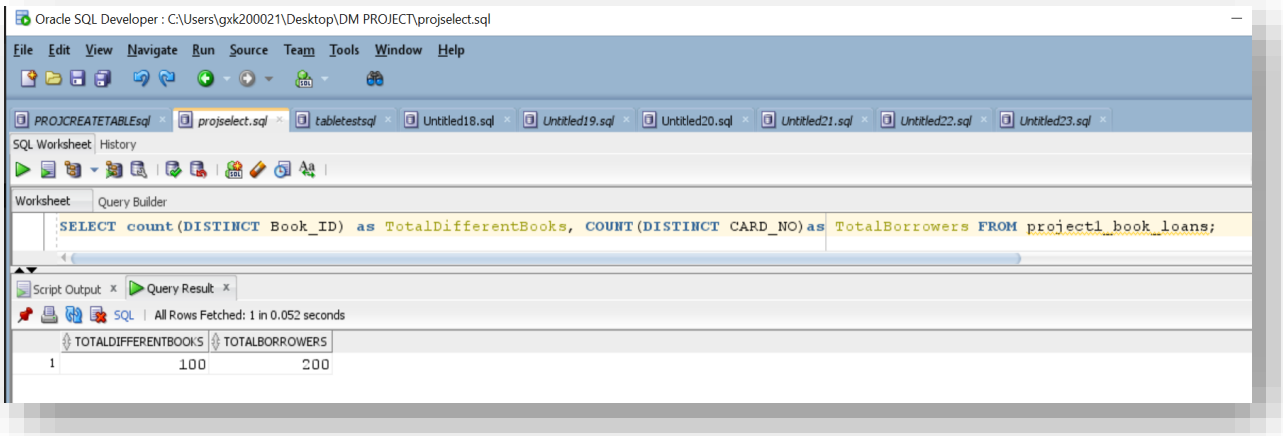


Data Load, Normalization, data generation

Ques: Write SQL to generate:

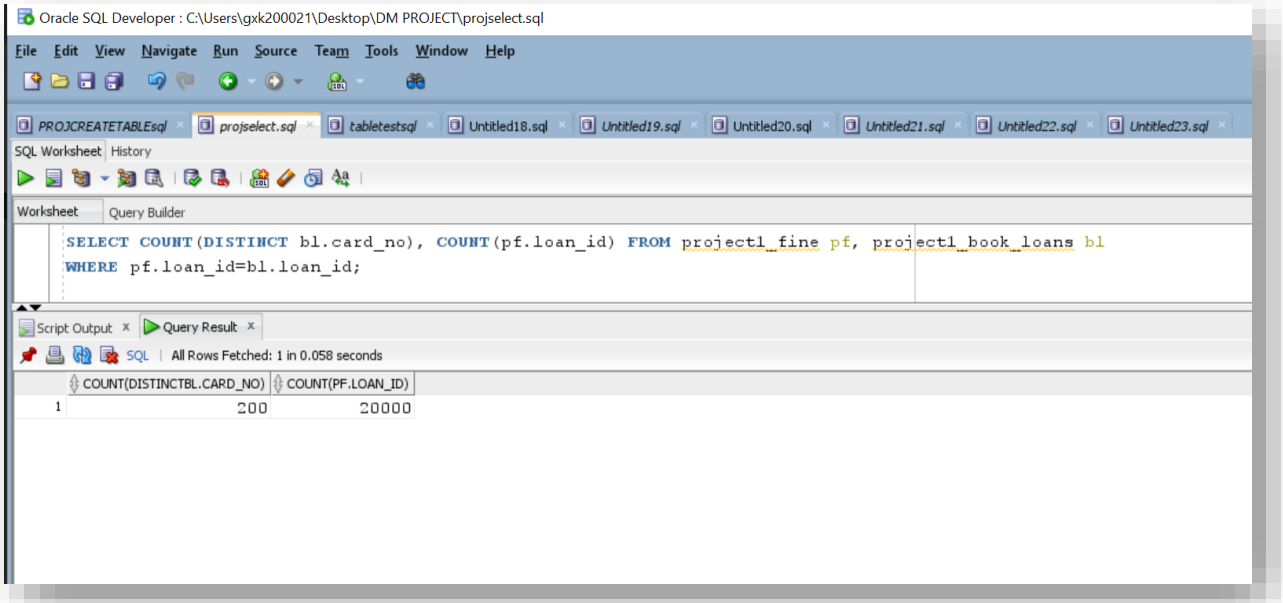
1. At least 500 books check-outs for at least 200 different borrowers and 100 different books

```
SELECT count(DISTINCT Book_ID) as TotalDifferentBooks, COUNT(DISTINCT
CARD_NO)as TotalBorrowers FROM project1_book_loans;
```



2. At least 50 fines records for 20 different borrowers.

```
SELECT COUNT(DISTINCT bl.card_no), COUNT(pf.loan_id) FROM project1_fine pf, project1_book_loans bl
WHERE pf.loan_id=bl.loan_id;
```



APPENDIX

Created an additional project1_book_names table where the names of the authors are normalized. I used this table to populate project1_authors table so that the mapping is easier.

Creating project1_books_name table:

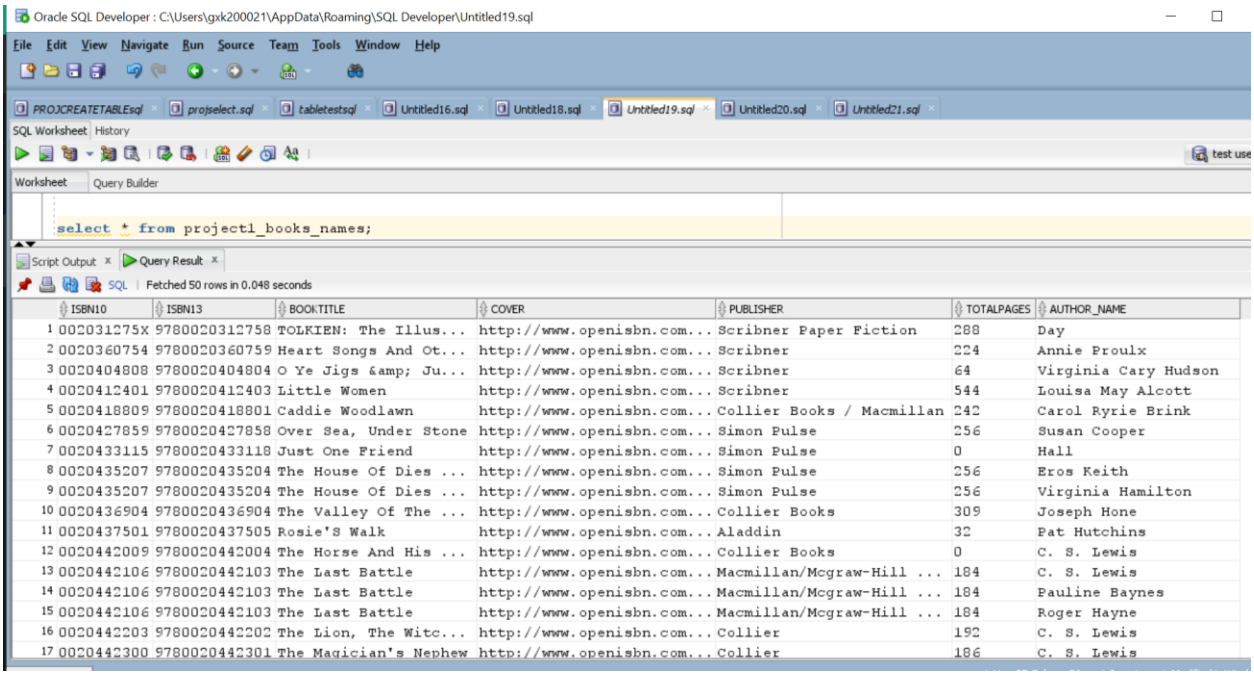
```
CREATE TABLE project1_books_names(isbn10 VARCHAR2(255), isbn13 VARCHAR2(255), booktitle VARCHAR2(255),
cover VARCHAR2(255), publisher VARCHAR2(255), totalpages VARCHAR2(255), author_name VARCHAR2(255),
PRIMARY KEY (isbn10, isbn13,author_name));
```

Inserting values in project1_books_name:

```
INSERT INTO project1_books_names(ISBN10, ISBN13,BookTitle, Cover, Publisher, TotalPages, Author_Name)
SELECT ISBN10, ISBN13,Title,
Cover, Publisher, pages,
REGEXP_SUBSTR(authro,'[^,;]+',1,1) FROM project1_books_load
WHERE REGEXP_SUBSTR(authro,'[^,;]+',1,1) IS NOT NULL

union
SELECT ISBN10, ISBN13,Title, Cover, Publisher, pages,
REGEXP_SUBSTR(authro,'[^,;]+',1,2) FROM project1_books_load
WHERE REGEXP_SUBSTR(authro,'[^,;]+',1,2) IS NOT NULL

union
SELECT ISBN10, ISBN13,Title, Cover, Publisher, pages,
REGEXP_SUBSTR(authro,'[^,;]+',1,3) FROM project1_books_load
WHERE REGEXP_SUBSTR(authro,'[^,;]+',1,3) IS NOT NULL

union
SELECT ISBN10, ISBN13,Title, Cover, Publisher, pages,
REGEXP_SUBSTR(authro,'[^,;]+',1,4) FROM project1_books_load
WHERE REGEXP_SUBSTR(authro,'[^,;]+',1,4) IS NOT NULL

union
SELECT ISBN10, ISBN13,Title, Cover, Publisher, pages,
REGEXP_SUBSTR(authro,'[^,;]+',1,5) FROM project1_books_load
WHERE REGEXP_SUBSTR(authro,'[^,;]+',1,5) IS NOT NULL;
```

Select statement to view project1_books_name table:

```
SELECT * FROM project1_books_name;
```



	ISBN10	ISBN13	BOOKTITLE	COVER	PUBLISHER	TOTALPAGES	AUTHOR_NAME
1	002031275X	9780020312758	TOLKIEN: The Illus...	http://www.openisbn.com...	Scribner Paper Fiction	288	Day
2	0020360754	9780020360759	Heart Songs And Ot...	http://www.openisbn.com...	Scribner	224	Annie Proulx
3	0020404808	9780020404804	O Ye Jigs & Ju...	http://www.openisbn.com...	Scribner	64	Virginia Cary Hudson
4	0020412401	9780020412403	Little Women	http://www.openisbn.com...	Scribner	544	Louisa May Alcott
5	0020418809	9780020418801	Caddie Woodlawn	http://www.openisbn.com...	Collier Books / Macmillan	242	Carol Ryrie Brink
6	0020427859	9780020427858	Over Sea, Under Stone	http://www.openisbn.com...	Simon Pulse	256	Susan Cooper
7	0020433115	9780020433118	Just One Friend	http://www.openisbn.com...	Simon Pulse	0	Hall
8	0020435207	9780020435204	The House Of Dies ...	http://www.openisbn.com...	Simon Pulse	256	Eros Keith
9	0020435207	9780020435204	The House Of Dies ...	http://www.openisbn.com...	Simon Pulse	256	Virginia Hamilton
10	0020436904	9780020436904	The Valley Of The ...	http://www.openisbn.com...	Collier Books	309	Joseph Hone
11	0020437501	9780020437505	Rosie'S Walk	http://www.openisbn.com...	Aladdin	32	Pat Hutchins
12	0020442009	9780020442004	The Horse And His ...	http://www.openisbn.com...	Collier Books	0	C. S. Lewis
13	0020442106	9780020442103	The Last Battle	http://www.openisbn.com...	Macmillan/Mcgraw-Hill ...	184	C. S. Lewis
14	0020442106	9780020442103	The Last Battle	http://www.openisbn.com...	Macmillan/Mcgraw-Hill ...	184	Pauline Baynes
15	0020442106	9780020442103	The Last Battle	http://www.openisbn.com...	Macmillan/Mcgraw-Hill ...	184	Roger Hayne
16	0020442203	9780020442202	The Lion, The Witc...	http://www.openisbn.com...	Collier	192	C. S. Lewis
17	0020442300	9780020442301	The Magician's Nephew	http://www.openisbn.com...	Collier	186	C. S. Lewis