**Library Management system**

**By**

**Mital Patel**

**Jay Patel**

**Bhavin Patel**

**Ami Patel**

**Monroe College**

**ABSTRACT**

Library Managing System is a web based application. The fundamental subject of the database is to store the details about books in a library and keep track of books the books in the library and out the library. Using this database we can easily find the book as per our input source. This will diminish the weight on clients who will keep up all records physically. General library administration is the most complex thing as it includes a great deal of database with in that. Overseeing Books in classification wise dispensing books to users is finished by manual procedure; computerization of this procedure will enhance the execution and decrease the mistakes.

**INTRODUCTION**

Library Managing System is a web based application the main theme of the application storing the details about the books in a library and their count and searching to gain one idea about the libraries books and we can also search categories wise books and book names and also administrator can issue books return books and add publisher add categories.

* **Purpose of the Project**

The Library Management System helps to improve efficiency of admin to keep track of all books easily. In addition, just searching with any one criterion they can find the book status.

* The main theme of our application is storing the book names .add category wise books and searching the books according to categories and details of return books and librarian can also remove the books.
* The system makes the overall task much easier and flexible.
* It can be accessed over the Internet/Internet.
* There is no risk of data mismanagement at any level while the project development is under process.
* **Functional Requirements:**

For every Library Managing System is below process for a receipt and issuance of books in the library along with the student’s details. The books received in the library are entered in Books Entry form and the new student is entered in the student/staff entry form. When the student/staff wants to get the desired book the same is issued on the availability basis to the student/staff.

* Admin enter his user id and password for login.
* User will enter his user id and password for login.
* New users give his completed personnel, address and phone details for registration.
* Admin gives different kind of user information to the user to search data.
* User will put a request for a new connection to the admin.
* Administrator will assign task to employees to provide services to the customers.
* Admin can have his own home page.
* User can have own login forms.
* The user requests and queries can store in the centralized database.
* Admin will get the login information of a particular user.
* The new user’s data will be stored in the centralized database.
* **Existing System:**

We have entered all the book details manually and book category information is a more importantly it is error prone. And searching books is also take long time it is very Burdon the man power searching and retrieving and mainly counting of the books. The following disadvantages due to manual system

* Searching books is also take long time it is very Burdon the man power searching and retrieving.
* It consumes lot of manpower.
* It deals with the managing books of users.
* There is no Data sharing Concept
* This process will improve the information briefly
* It lacks data security.
* Retrieval of data takes a lot of time.
* Percentage of accuracy is less.
* Reports take time to produce.
* **Features of the System:**
* Library Management System stored all information of student/staff in database
* Student/Staff can easily search book in library
* User can easily issue book and they can able know the how many copy of book available in to library
* Library Management System is easily to maintain all books records
* Admin user can see the list of all issue books record
* The system can fine automatically for late submission of book
* The system also show up the due-date and remain to user for last date
* **Modules for the Library Management System**

After looking at the functional requirements it is identified that we can divide the system into mainly following modules:

* **Admin Module**

Admin enter his user id and password for login. Admin gives different kind of user information to the user to search data. Administrator will assign task to faculty to provide services to the users. Admin can have his own home page.

**The task for Admin modules:**

* Admin can give permission to user for login
* Admin can give a permission to user for search a book
* Admin can update the system
* Admin has hole recorded of the books transaction
* Admin also see the user activity on the system
* **User Module**

User will enter his user id and password for login. New users give his completed personnel, address and phone details for registration. User will put a request for a new connection to the admin.

**The task for Admin modules:**

* User can search the books
* User can issue books
* User can able to see status and remainder on the system

**SYSTEM DESIGN**

* **Normalized each tables for minimize data redundancy**

We started with first normalized form of each tables in which we defined the primary key and identified dependencies for each tables. In addition, in the first normalization form, we eliminated repeating groups. Secondly, it is required for the new tables to remove partial dependencies. Afterwards, we had assigned corresponding attribute for dependent. At last, in third normalization, it is necessary to remove transitive dependency. So we removed it and again reassigned the corresponding attribute for dependant.

**FIRST NORMAL FORM**:

A relation is said to be in first normal form if the values in the relation are atomic for every attribute in the relation. By this we mean simply that no attribute value can be a set of values or, as it is sometimes expressed, a repeating group.

1. Eliminate repeating groups

2. Identify PK

3. Identify all dependencies

**SECOND NORMAL FORM**:

A relation is said to be in second Normal form is it is in first normal form and it should satisfy any one of the following rules.

1. Make new tables to remove partial dependency
2. Reassign corresponding dependent attribute

**THIRD NORMAL FORM**:

A relation is said to be in third normal form if their exits no transitive dependencies.

1. Make new tables to remove transitive dependencies

2. Reassign corresponding dependent attributes

**Transitive Dependency**: If two non key attributes depend on each other as well as on the primary key then they are said to be transitively dependent.

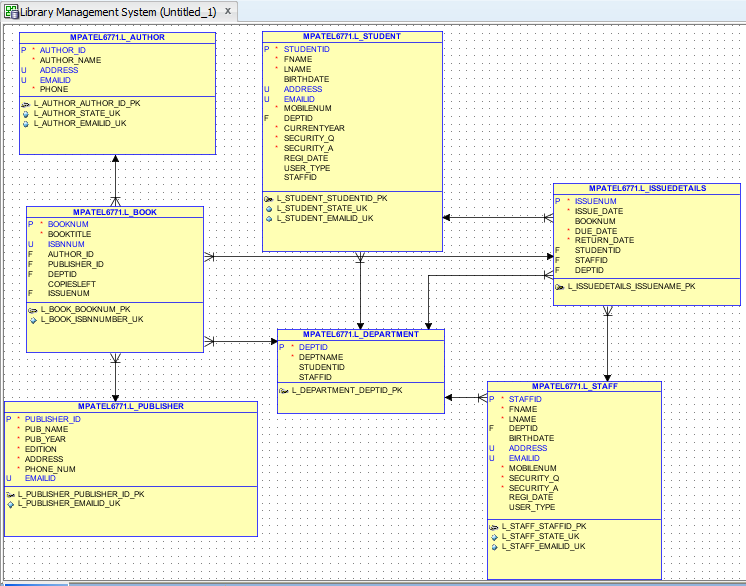
The above normalization principles were applied to decompose the data in multiple tables thereby making the data to be maintained in a consistent state.

**List of tables for Library Management System**

1. L\_STUDENT
2. L\_STAFF
3. L\_DEPARTMENT
4. L\_BOOK
5. L\_AUTHOR
6. L\_ISSUEDETAILS
7. L\_PUBLISHER

We have created seven tables and the beginning letter with underscore represents tables for Library Management System. At the beginning, we created table with primary key constraint but the foreign key constraint was added after values added into each tables. Additionally, we added genuine values to the each tables to make the project error resist. Furthermore, Constraints provides one methods of implementing business rules in the database. Constraint restricts the data that stored in the relation. Many constraints like NOT NULL, FOREING KEY, UNIQUE, CHECK are used in this project. These help the users and admin both.

* **E-R Diagram**
* The relation upon the system is structure through a conceptual ER-Diagram, which not only specifics the existential entities but also the standard relations through which the system exists and the cardinalities that are necessary for the system state to continue.
* The Entity Relationship Diagram (ERD) depicts the relationship between the data objects. The ERD is the notation that is used to conduct the date modelling activity the attributes of each data object noted is the ERD can be described resign a data object descriptions.
* The set of primary components that are identified by the ERD are
  + Data object
  + Relationships
  + Attributes
  + Various types of indicators.
* The primary purpose of the ERD is to represent data objects and their relationships.



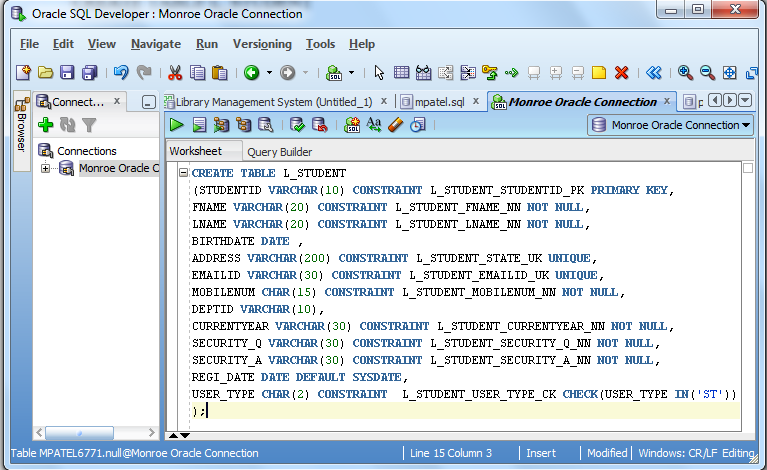
**Figure: Entity-Relationship Diagram (ERD)**

* **DDL Queries:**

SQL (Structural Query Language): Here, it used the query to confirm the user’s requirements. First of all, DML (Data Manipulation Language) which was used here to insert new rows and removing rows from the existing tables. Secondly, DDL (Data Definition Language) which helped us to create the tables, drop the tables. These are CREATE, ALTER, DROP and RENAME. Also, this DDL helps us to insert the appropriate data in to each table.

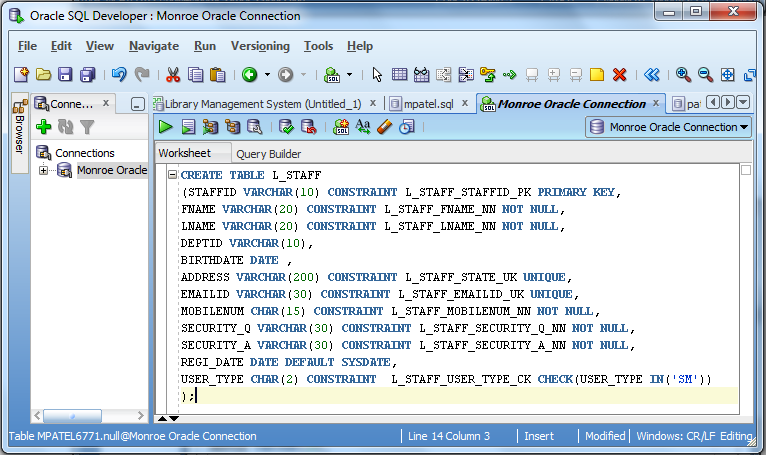
* **DDL Queries**
* **Queries for table Creation:**

1. **Create L\_STUDENT table**



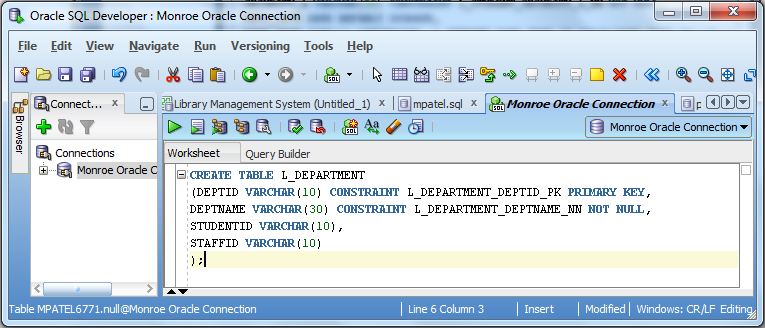
**OUTPUT: table L\_STUDENT created.**

1. **Create L\_STAFF table**



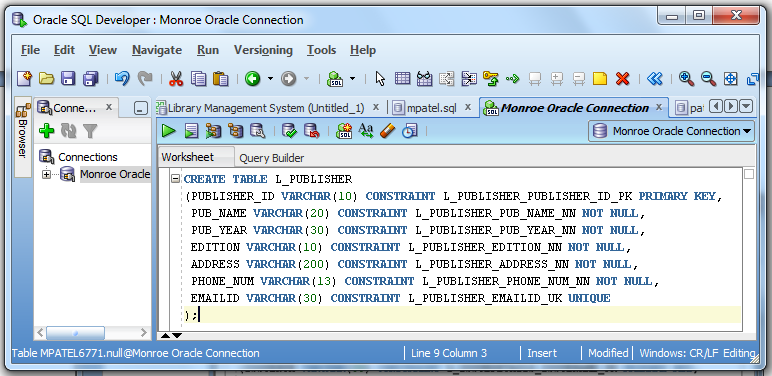
**OUTPUT: table L\_STAFF created.**

1. **Create L\_DEPARTMENT table**



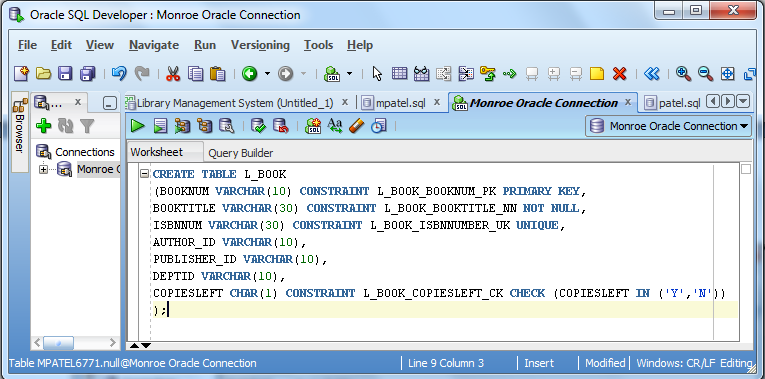
**OUTPUT: table L\_DEPARTMENT created.**

1. **Create L\_PUBLISHER table**



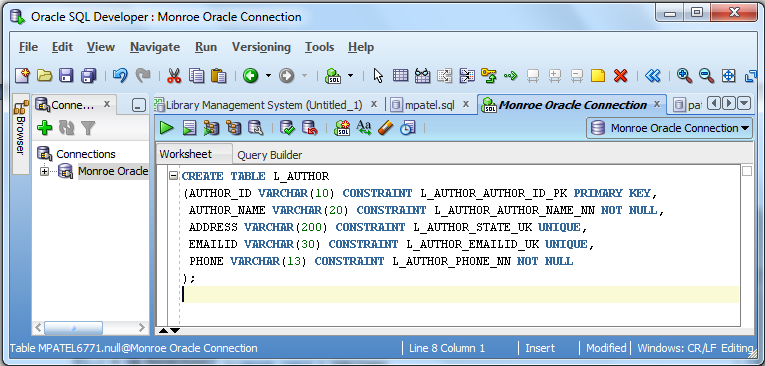
**OUTPUT: table L\_PUBLISHER created.**

1. **Create L\_BOOK table**



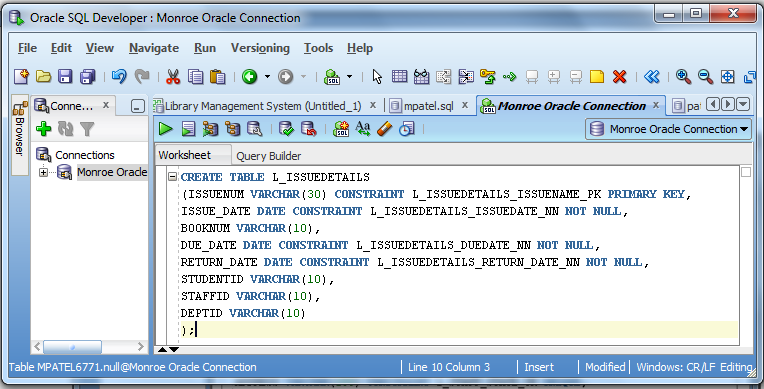
**OUTPUT: table L\_BOOK created.**

1. **Create L\_AUTHOR table**



**OUTPUT: table L\_AUTHOR created.**

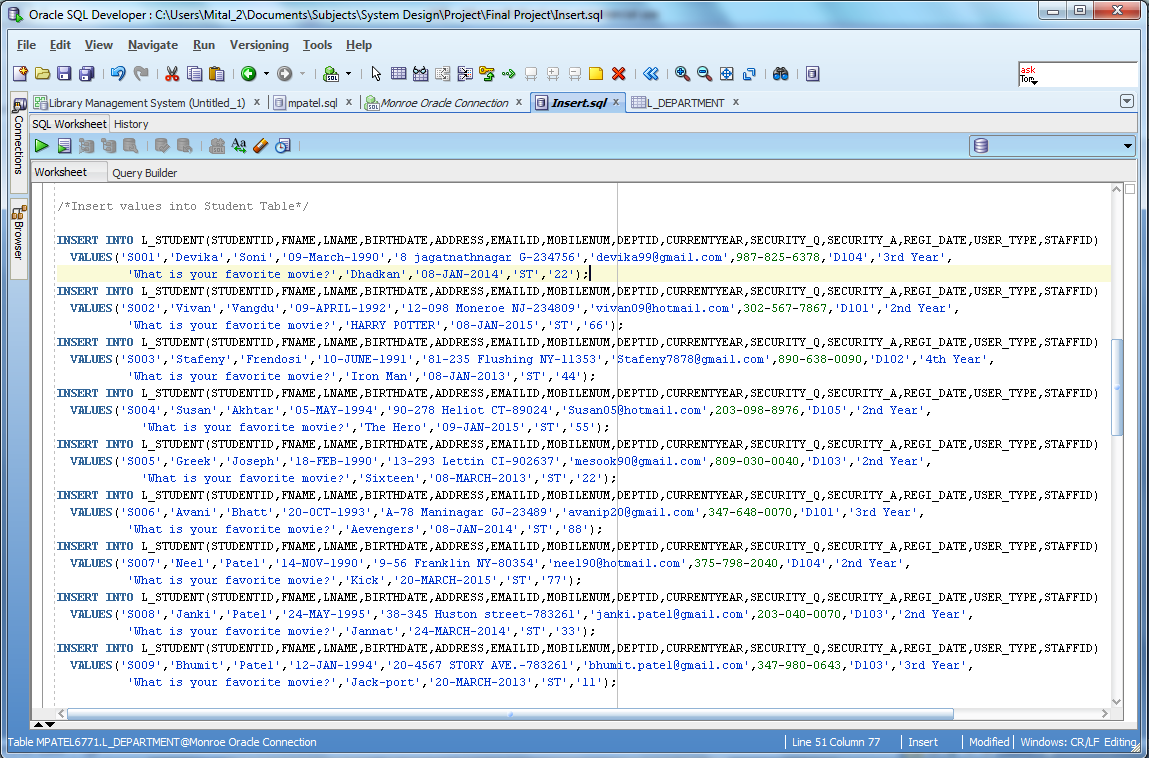
1. **Create L\_ISSUEDETAILS table**



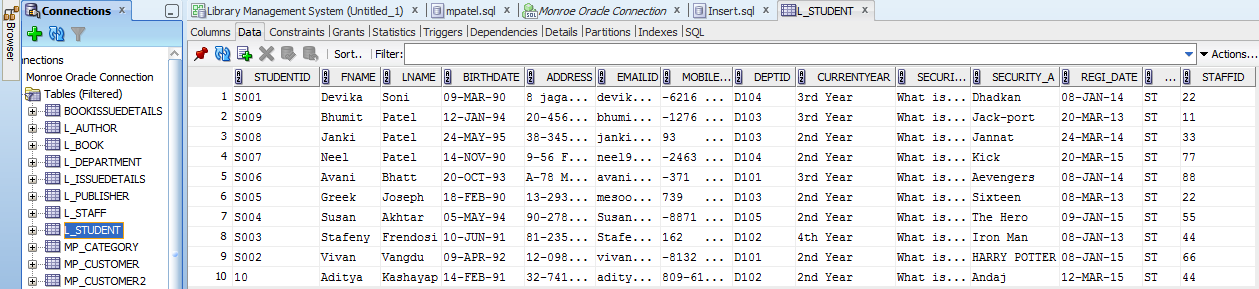
**OUTPUT: table L\_ISSUEDETAILS created.**

* **Queries for Insert values in to tables:**

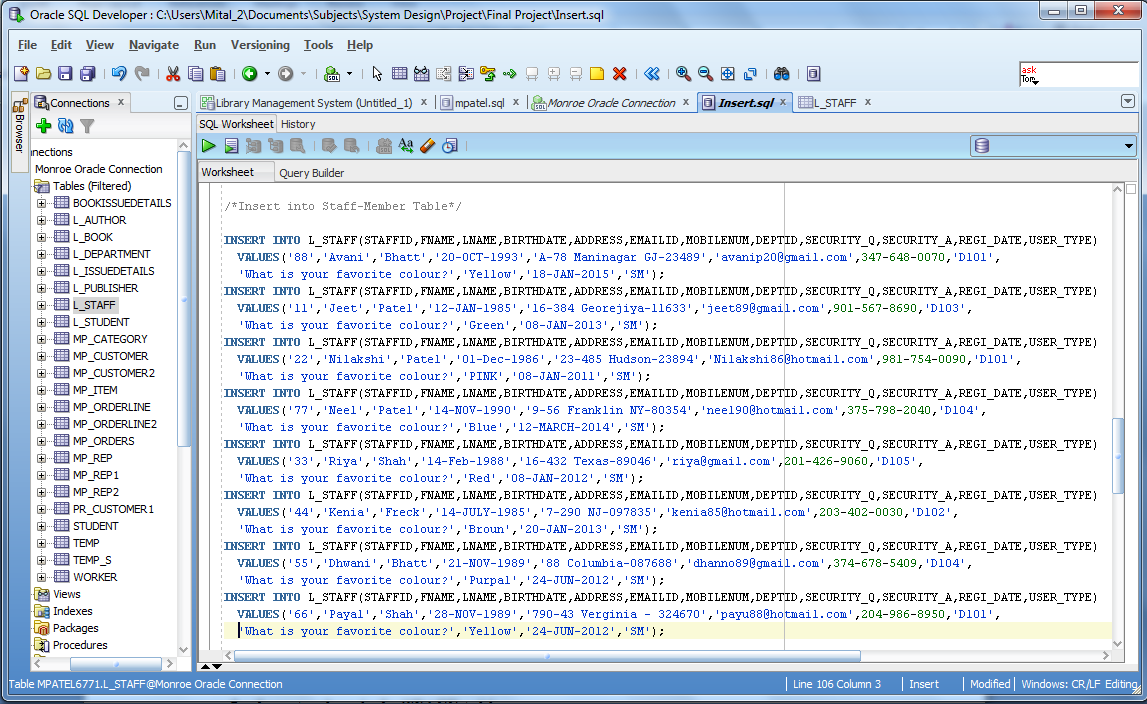
1. **Insert values in L\_STUDENT table**



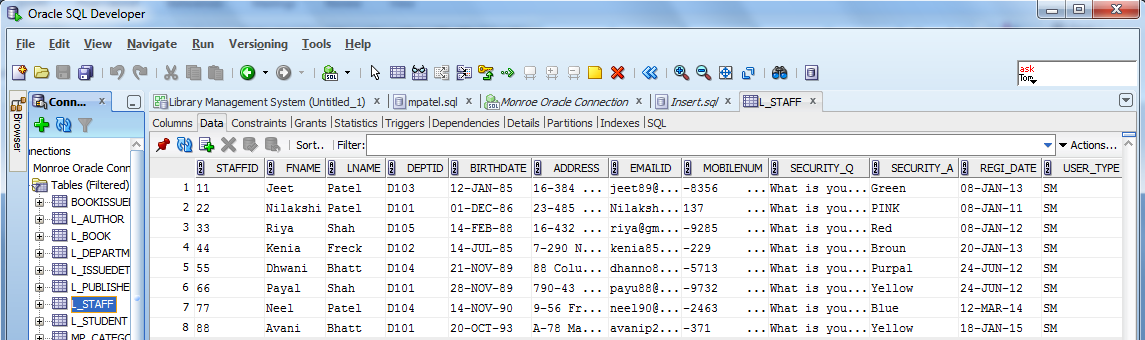
**OUTPUT:**

****

1. **Insert values in L\_STAFF table**



**OUTPUT:**

****

1. **Insert values in L\_BOOK table**

/\* Insert into book Table \*/

INSERT INTO MPATEL6771.L\_BOOK (BOOKNUM,BOOKTITLE,ISBNNUM,AUTHOR\_ID,PUBLISHER\_ID,DEPTID,COPIESLEFT,ISSUENUM)

VALUES ('B901','COMPUTER ARCHITECHTURE','ISBN5001','A105','P201','D105','N','I1007');

INSERT INTO MPATEL6771.L\_BOOK (BOOKNUM,BOOKTITLE,ISBNNUM,AUTHOR\_ID,PUBLISHER\_ID,DEPTID,COPIESLEFT,ISSUENUM)

VALUES ('B905','SOFTWARE ENGINEERING','ISBN5005','A101','P205','D105','Y','I1001');

INSERT INTO MPATEL6771.L\_BOOK (BOOKNUM,BOOKTITLE,ISBNNUM,AUTHOR\_ID,PUBLISHER\_ID,DEPTID,COPIESLEFT,ISSUENUM)

VALUES ('B902','DATABASE SYSTEMS','ISBN5004','A102','P202','D105','N','1002');

INSERT INTO MPATEL6771.L\_BOOK (BOOKNUM,BOOKTITLE,ISBNNUM,AUTHOR\_ID,PUBLISHER\_ID,DEPTID,COPIESLEFT,ISSUENUM)

VALUES ('B903','ECONOMICS','ISBN5002','A103','P203','D102','Y','1005');

INSERT INTO MPATEL6771.L\_BOOK (BOOKNUM,BOOKTITLE,ISBNNUM,AUTHOR\_ID,PUBLISHER\_ID,DEPTID,COPIESLEFT,ISSUENUM)

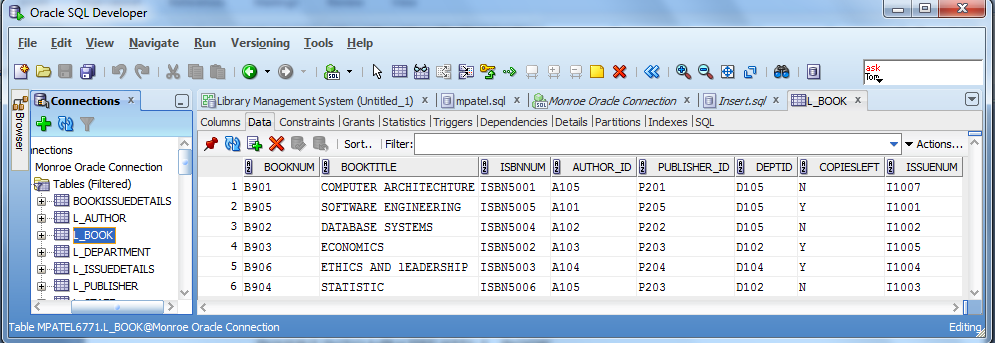
VALUES ('B906','ETHICS AND LEADERSHIP','ISBN5003','A104','P204','D104','Y','1004');

INSERT INTO MPATEL6771.L\_BOOK (BOOKNUM,BOOKTITLE,ISBNNUM,AUTHOR\_ID,PUBLISHER\_ID,DEPTID,COPIESLEFT,ISSUENUM)

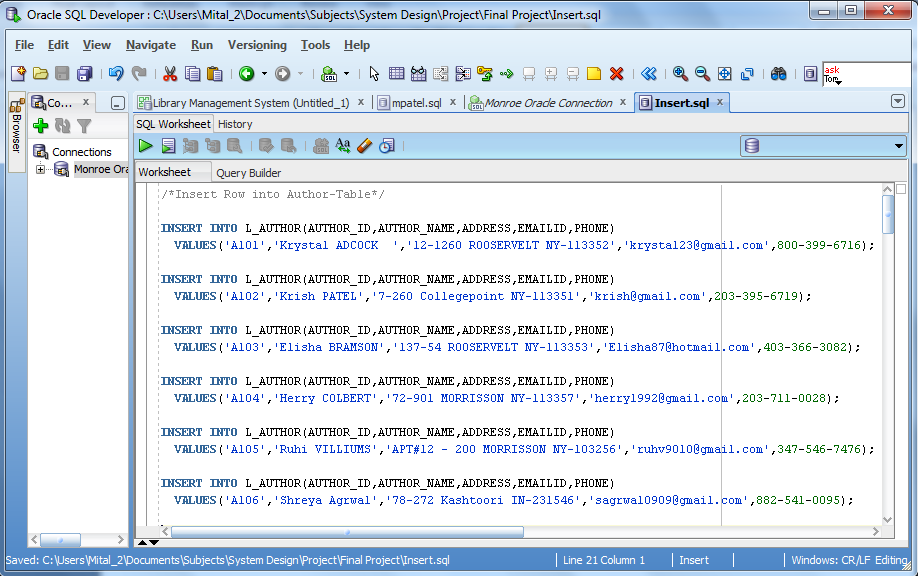
VALUES ('B904','STATISTIC','ISBN5006','A105','P203','D102','N','1003');

COMMIT MPATEL6771.L\_BOOK;

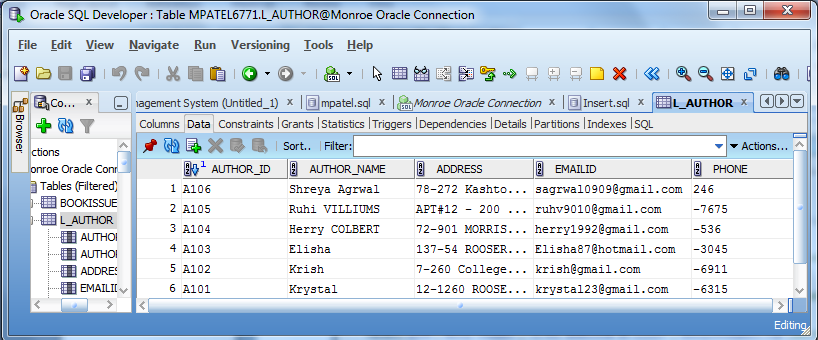
**OUTPUT:**

****

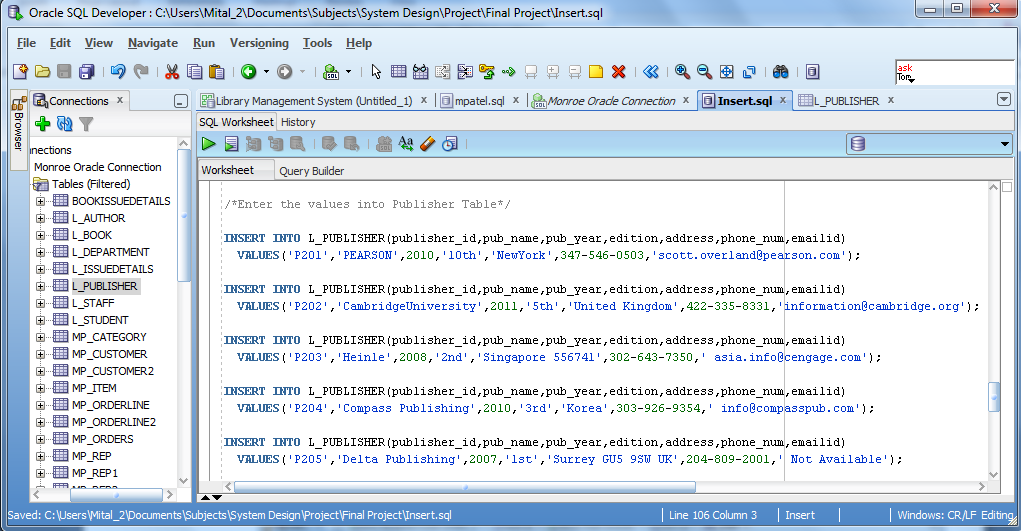
1. **Insert values in L\_AUTHOR table**



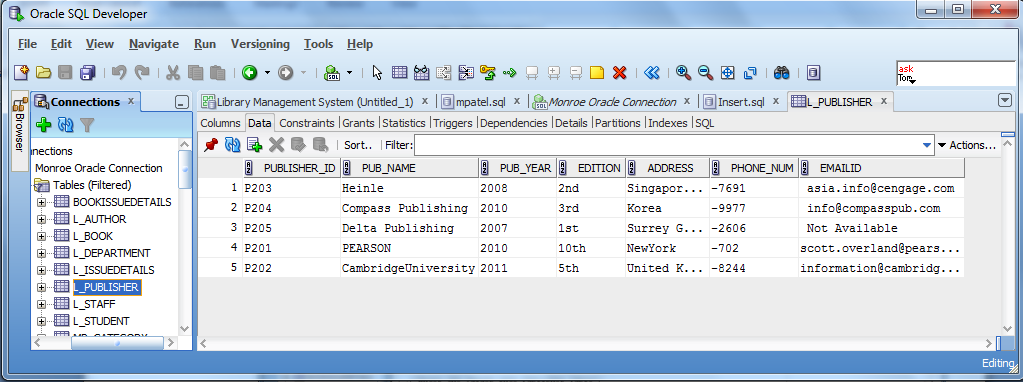
**OUTPUT:**

****

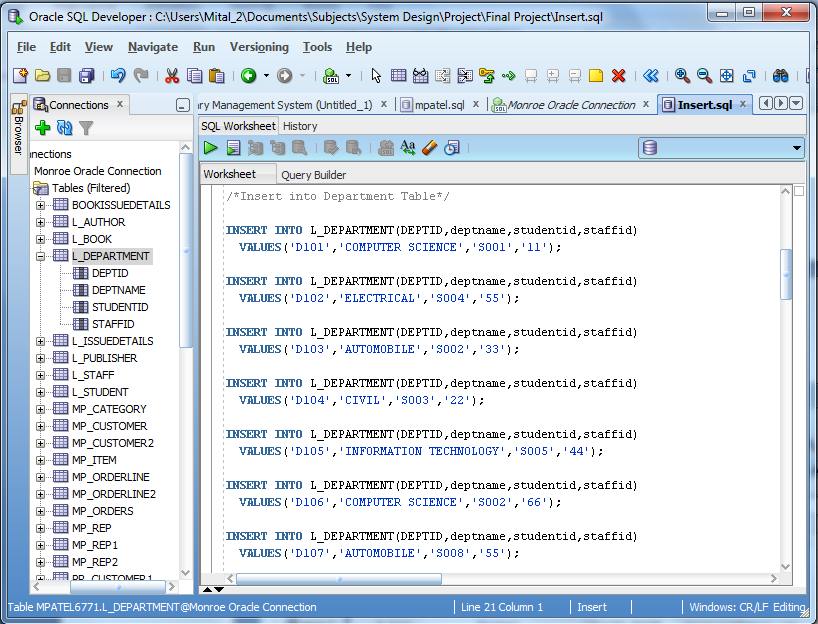
1. **Insert values in L\_PUBLISHER table**



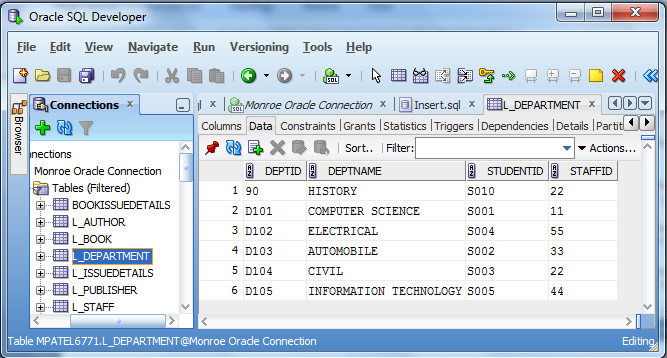
**OUTPUT:**

****

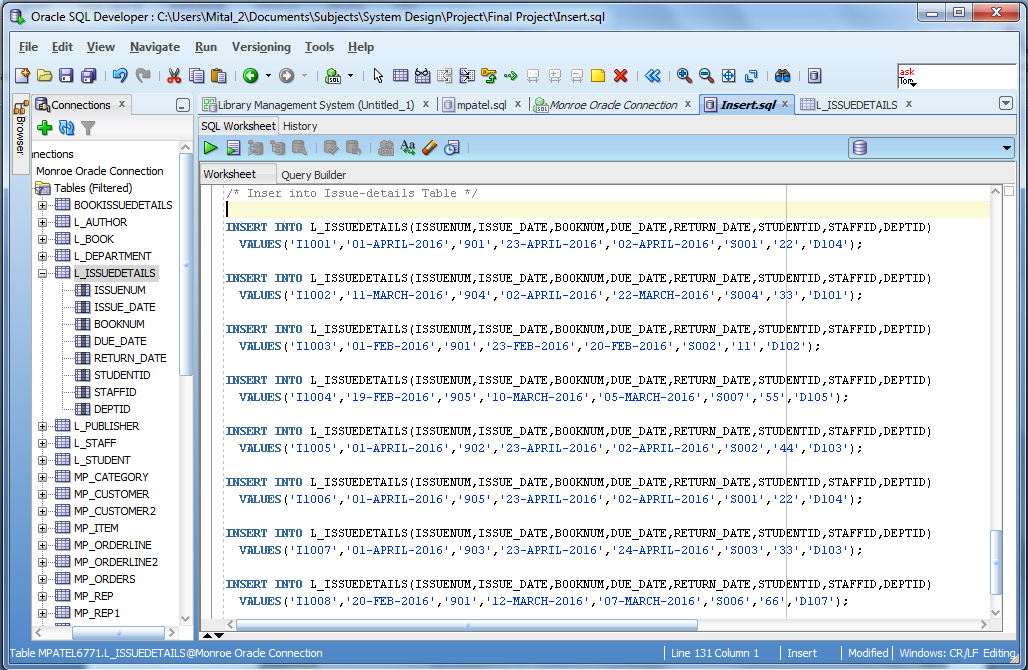
1. **Insert values in L\_DEPARTMENT table**



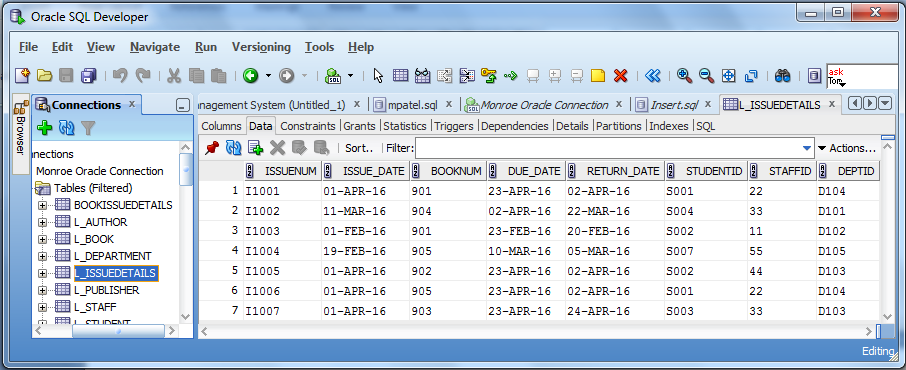
**OUTPUT:**



1. **Insert values in L\_ISSUEDETAILS table**



**OUTPUT:**

****

* **Queries for simple/complex views for the end-users**

**VIEW**: A view is an Oracle object that gives a user logical view of data from an underlying table or tables. You can restrict what users can view by allowing them to see only a few columns from a table. . Views also hide names of the underlying tables, so the user does not know where the data is coming from. In short, a view is a logical representation of a subset of data from one or more tables. A view is stored as a SELECT statement in the data dictionary. There are two types of views: simple and complex.

|  |  |  |
| --- | --- | --- |
| **Feature** | **Simple Views** | **Complex Views** |
| **Number of tables** | One | One or more |
| **Contain functions** | No | Yes |
| **Contain groups of data** | No | Yes |
| **DML operations through a view** | Yes | Not always |

**Difference between simple views and complex views**

**SYNONYM:** Sometimes object names are very long. If a query uses the object’s name more than once, the user has to type that long name many times. You already know the use of table aliases to shorten table names. A synonym name must be different than all other objects owned by the user.

**SEQUENCE:** A sequence is an Oracle object that is used to generate sequence of numbers. Many times you create a table with surrogate key column such as Student ID, Faculty ID, or Employee ID. These columns have a numeric data type. Sequencing is a perfect solution for generating values for such numeric columns.

**INDEX:** An index is another Oracle object that is used for faster retrieval of rows from a table. An index can be created explicitly by using the CREATE INDEX statement or implicitly by Oracle. Once an index exists for a table, the user does not have to open or use the index with a command or a statement. The Oracle server uses the index to search for a row rather than scanning through the entire table. Indexing reduces the search time and disk I/O

**SUBQUERIES:** Sub queries are also known as nested queries. A sub query is usually a SELECT query within one of the clauses in another SELECT query. Very powerful queries can be designed by using simple sub queries. A sub query is very useful when a query based on a table depends on the data in that sub query itself.

* **Queries for Alter tables:**
* **Simple/complex View Create**
* **Data Retrieval Queries**
* **Queries for joins and functions:**
* **Queries for Set Operators:**
* **Queries for Nested(Sub query) :**
* **Various Database Objects**
* **Queries for Synonym:**
* **Queries for View:**
* **Queries for Sequence:**
* **Queries for Index:**
* **User access control for different objects**
* **PL/SQL Blocks for data manipulation**