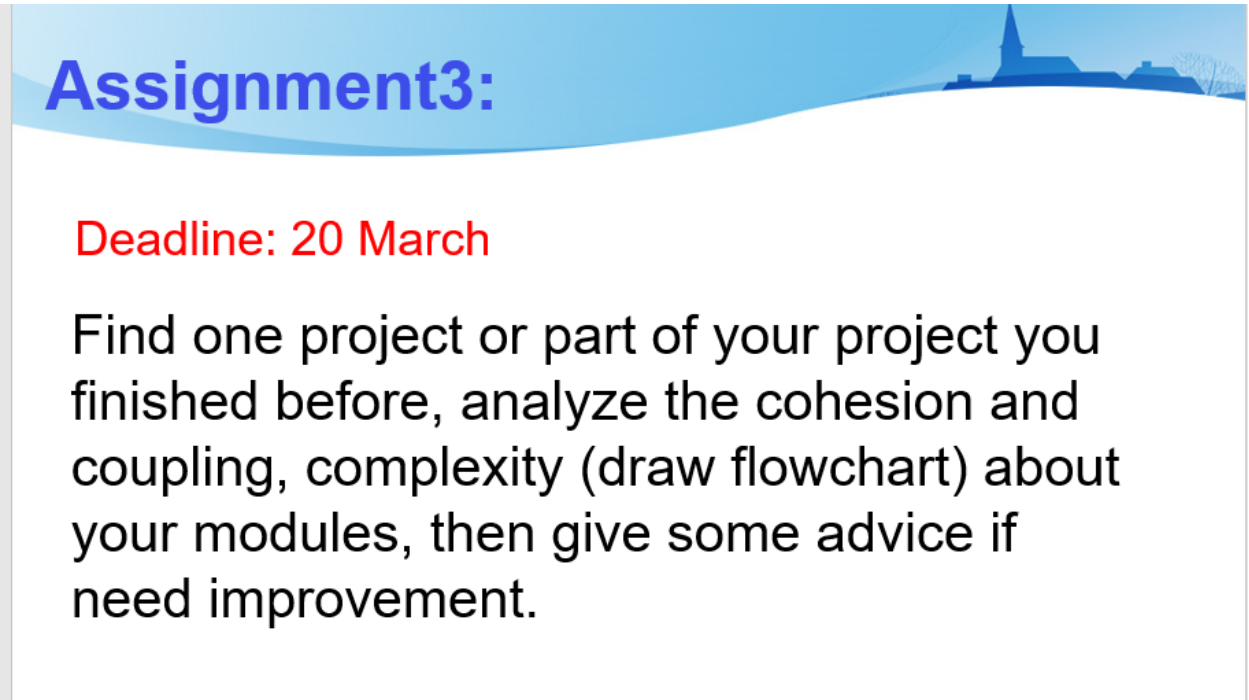
Software Engineering

Assignment 3

Name : ABID ALI

Roll : 2019380141

Send to : lining@nwpu.edu.cn , 480755534@qq.com



Solution:

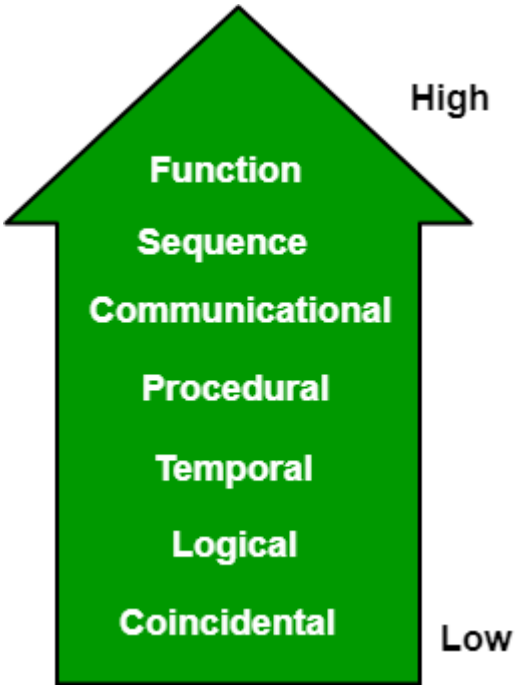
Modularization: Modularization is the process of dividing a software system into multiple independent modules where each module works independently. There are many advantages of Modularization in software engineering. Some of these are given below:

- Easy to understand the system.
- System maintenance is easy.
- A module can be used many times as their requirements. No need to write it again and again.

Cohesion:

Cohesion is a measure of functional strength of a module.

*A cohesive module perform a single task or function.



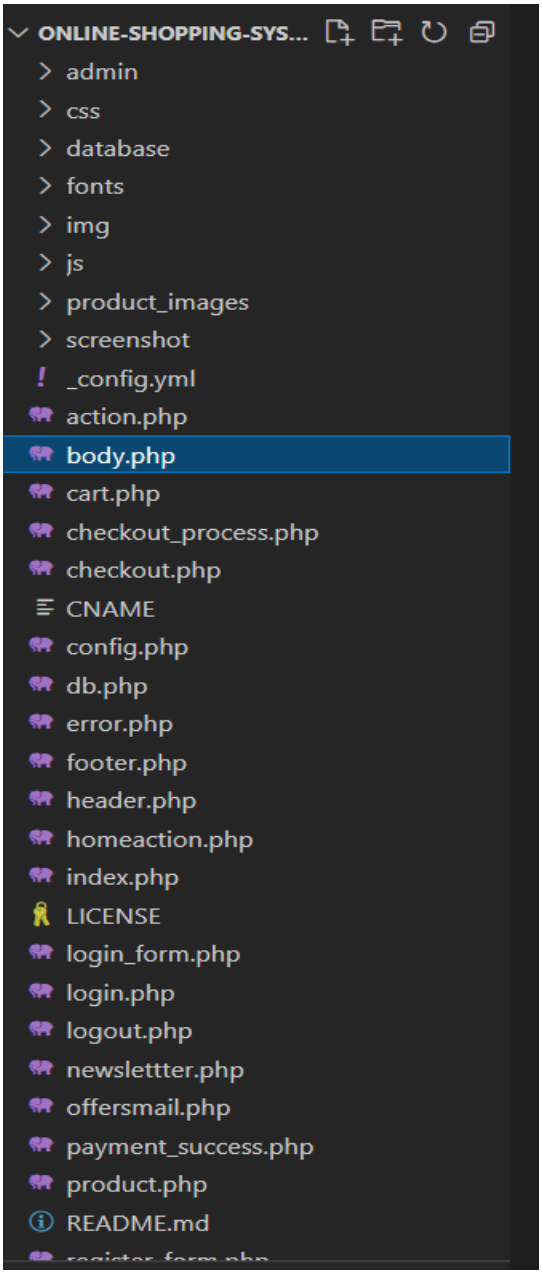
Types of Cohesion:

- **Functional Cohesion:** Every essential element for a single computation is contained in the component. A functional cohesion performs the task and functions. It is an ideal situation.
- **Sequential Cohesion:** An element outputs some data that becomes the input for other element, i.e., data flow between the parts. It occurs naturally in functional programming languages.
- **Communicational Cohesion:** Two elements operate on the same input data or contribute towards the same output data. Example- update record in the database and send it to the printer.
- **Procedural Cohesion:** Elements of procedural cohesion ensure the order of execution. Actions are still weakly connected and unlikely to be reusable. Ex- calculate student GPA, print student record, calculate cumulative GPA, print cumulative GPA.

- **Temporal Cohesion:** The elements are related by their timing involved. A module connected with temporal cohesion all the tasks must be executed in the same time span. This cohesion contains the code for initializing all the parts of the system. Lots of different activities occur, all at unit time.
- **Logical Cohesion:** The elements are logically related and not functionally. Ex- A component reads inputs from tape, disk, and network. All the code for these functions is in the same component. Operations are related, but the functions are significantly different.
- **Coincidental Cohesion:** The elements are not related(unrelated). The elements have no conceptual relationship other than location in source code. It is accidental and the worst form of cohesion. Ex- print next line and reverse the characters of a string in a single component.

For the cohesion analysis I took different program I've been working now .

My program is using PHP to create a little web application using back(Mysql) and front (PHP). It has different classes that are up to completing different tasks.



For the php it uses 2 classes which tasks are to describe and object "login" and "logout", the "db" class is for connecting to database .

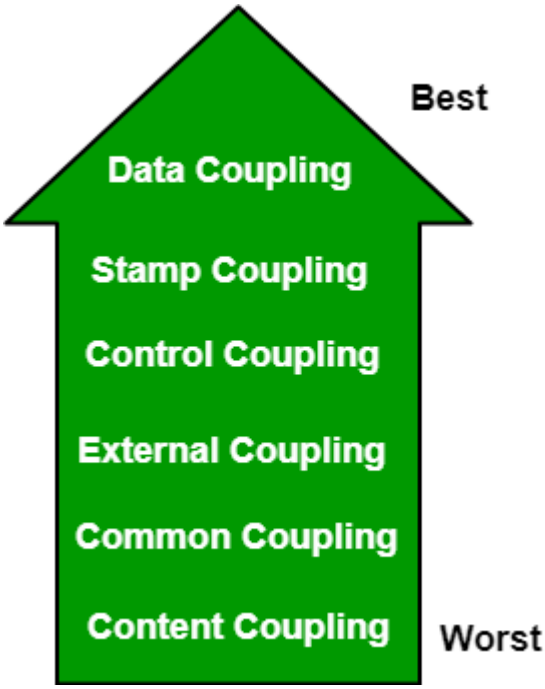
Different servlets classes are for implementing the different tasks according to their class name. And connect to the web application.

As you might see the files themselves are separated too and each for a specific task implementation. Unlike the program where I have a single class performing many different tasks, here I have created a several different classes, each class performing a specific specialized tasks, leading to an easy creation and modification of these classes. And it is a “High cohesion “classes.

Coupling:

Coupling can be defined as measuring of the degree of interdependence or interaction between the two modules.

*A good software will have low coupling.



Types of Coupling:

- **Data Coupling:** If the dependency between the modules is based on the fact that they communicate by passing only data, then the modules are said to be data coupled. In data coupling, the components are independent of each

other and communicate through data. Module communications don't contain tramp data. Example-customer billing system.

- **Stamp Coupling** In stamp coupling, the complete data structure is passed from one module to another module. Therefore, it involves tramp data. It may be necessary due to efficiency factors- this choice was made by the insightful designer, not a lazy programmer.
- **Control Coupling:** If the modules communicate by passing control information, then they are said to be control coupled. It can be bad if parameters indicate completely different behavior and good if parameters allow factoring and reuse of functionality. Example- sort function that takes comparison function as an argument.
- **External Coupling:** In external coupling, the modules depend on other modules, external to the software being developed or to a particular type of hardware. Ex- protocol, external file, device format, etc.
- **Common Coupling:** The modules have shared data such as global data structures. The changes in global data mean tracing back to all modules which access that data to evaluate the effect of the change. So it has got disadvantages like difficulty in reusing modules, reduced ability to control data accesses, and reduced maintainability.
- **Content Coupling:** In a content coupling, one module can modify the data of another module, or control flow is passed from one module to the other module. This is the worst form of coupling and should be avoided.

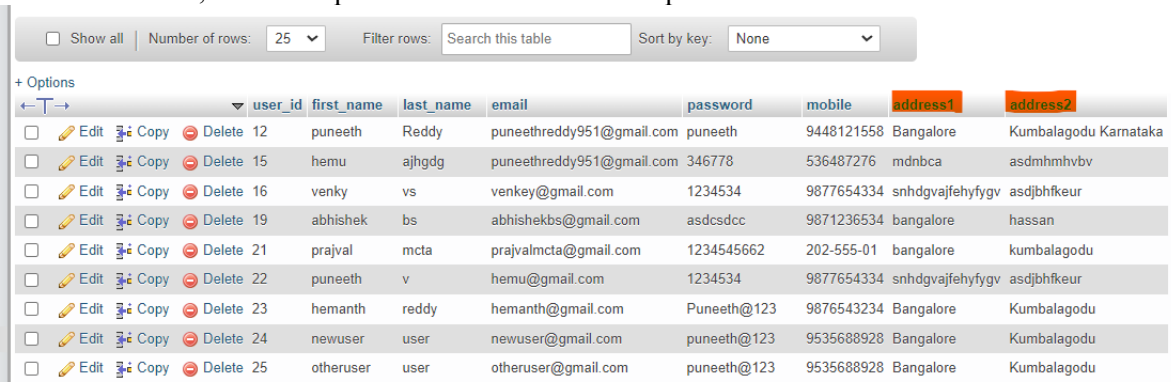
NORMALIZATION: -

1st normalization:

Atomic Values

We cannot have two datas in the same column.

We can see here ,we have separated the 2 address in 2 separate column.



Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

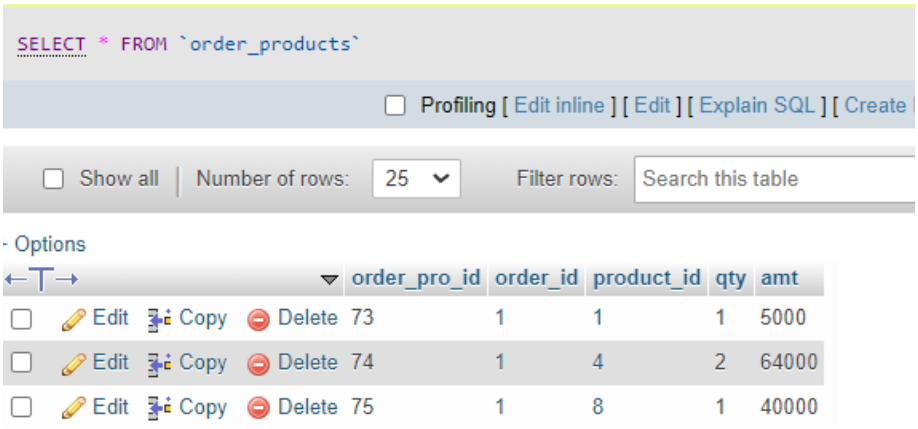
	user_id	first_name	last_name	email	password	mobile	address1	address2
Edit Copy Delete	12	puneeth	Reddy	puneethreddy951@gmail.com	puneeth	9448121558	Bangalore	Kumbalagodu Karnataka
Edit Copy Delete	15	hemu	ajhgdg	puneethreddy951@gmail.com	346778	536487276	mdnbca	asdmhmhvbv
Edit Copy Delete	16	venky	vs	venkey@gmail.com	1234534	9877654334	snhdgvajfehyfygv	asdjbhfkeur
Edit Copy Delete	19	abhishek	bs	abhishekbs@gmail.com	asdcsdcc	9871236534	bangalore	hassan
Edit Copy Delete	21	prajval	mcta	prajvalmcta@gmail.com	1234545662	202-555-01	bangalore	kumbalagodu
Edit Copy Delete	22	puneeth	v	hemu@gmail.com	1234534	9877654334	snhdgvajfehyfygv	asdjbhfkeur
Edit Copy Delete	23	hemanth	reddy	hemanth@gmail.com	Puneeth@123	9876543234	Bangalore	Kumbalagodu
Edit Copy Delete	24	newuser	user	newuser@gmail.com	puneeth@123	9535688928	Bangalore	Kumbalagodu
Edit Copy Delete	25	otheruser	user	otheruser@gmail.com	puneeth@123	9535688928	Bangalore	Kumbalagodu

We can see here ,we have separated the 2 address in 2 separate column.

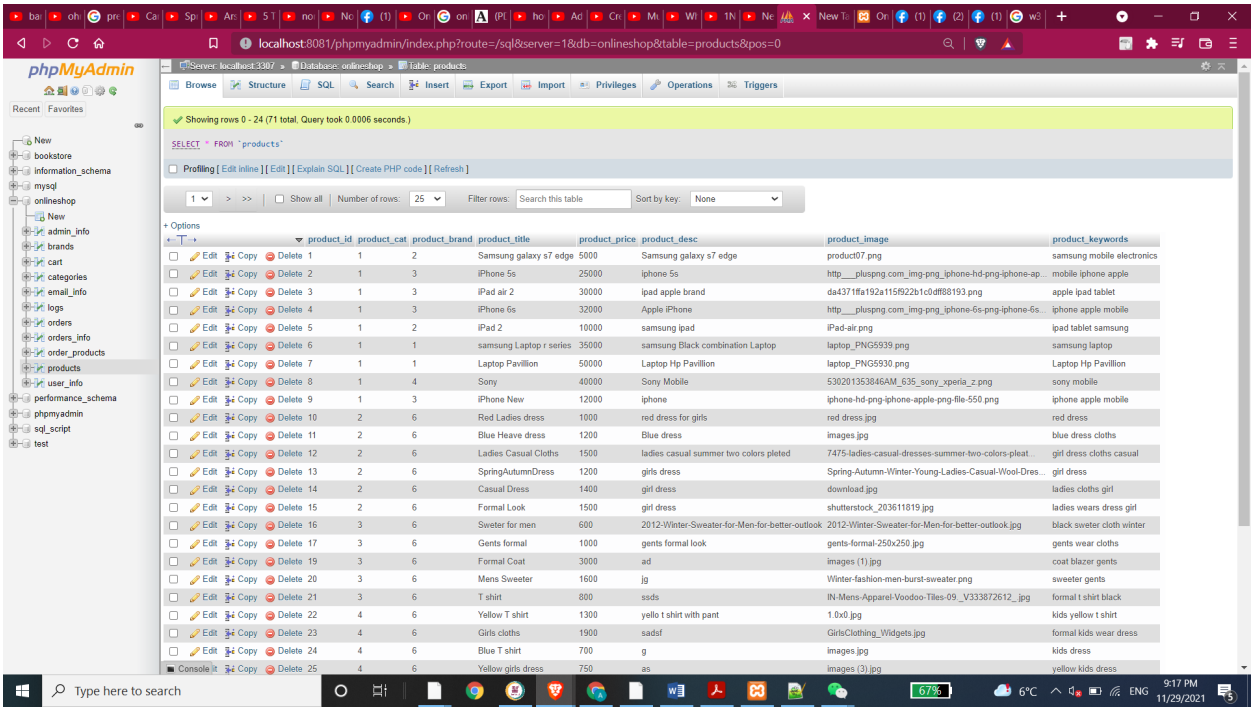
Therefore,we can say that we have fulfilled the 1st Normalization.

2nd normalization:

As we fulfilled the 1st normalization then we can move to 2nd normalization.
For example, in the above picture we can see there is column named product_id which will connect with another table name product. Therefore,we have to create a separate table named products for that purpose.If we don't do that we can see redundancy and partial dependency.



For example, in the above picture we can see there is column named product_id which will connect with another table name product. Therefore,we have to create a separate table named products for that purpose.If we don't do that we can see redundancy and partial dependency.

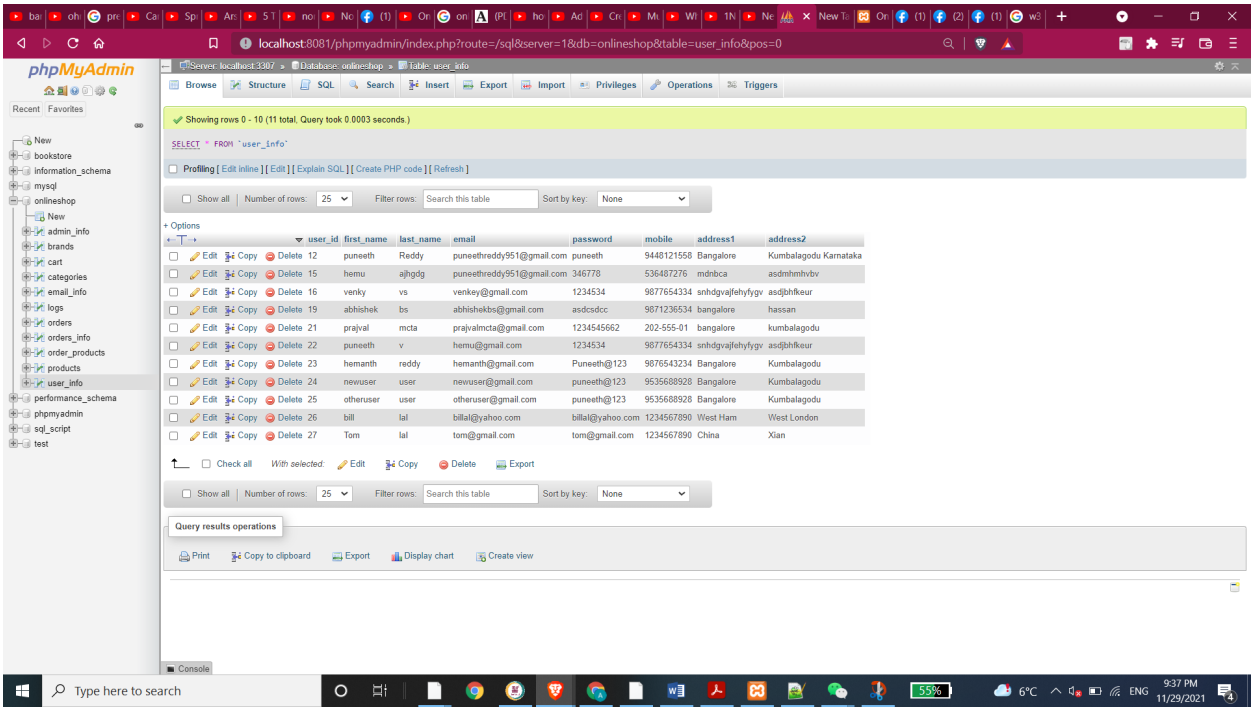


By creating a separate of table containing all the items of product available. We create 2nd normalization.By using 2nd normalization, we can solve the problem of update abnomaly.We can see at the above there is no partial dependency and the columns depends on whole key.

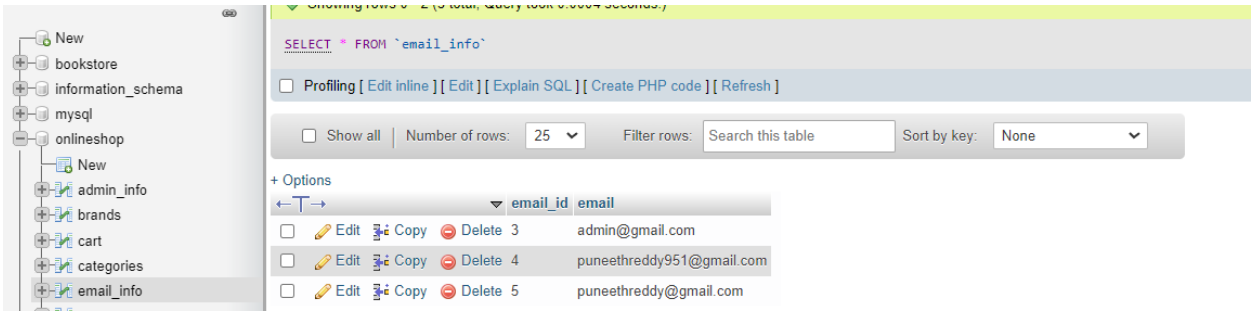
Therefore,we can say that we have fulfilled the 2nd Normalization.

3rd normalization:

As we fulfilled the 1st normalization and 2nd normalization then we can move to 3rd normalization.



As we can see, a user or customer can more than one email.If we add their email address each time in the user table then it will redundancy and each time they buy anything we have to the copy their email address in the table .



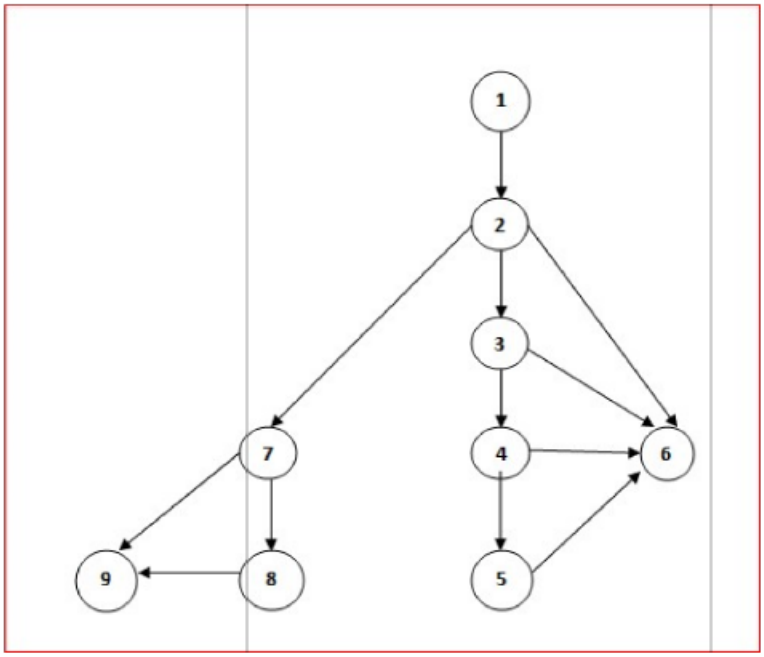
By making another table where information about Emails are stored.By doing that,we solve the update anamoly problem.Each time value of email will be stored in this table and won't store in the main table which would look messy.
Hence,we can say that, all of it's columns are not transitively dependent on the primary key

Therefore,we can say that we have fulfilled the 3rd Normalization.

As we can in my project ,we have 3rd Degree Normalization in my program the dependence between the 2 modules has been reduced significantly. Low coupling can be seen.

Complexity of Flow Graph given below:

FLOW GRAPH



Number of regions=E-V+2

=12-9+3

=5

Number of regions=number of predicate nodes +1

=4+1

=5

Some advice can be presented by try to keep the code more loosely coupled situation compared to present situation, then one module can perform without impacting other modules . Trying to make it more easier so that we can easily write DRY code that is easy to work with.