# **ONLINE ALLOCATION OF CAB MANAGEMENT SYSTEM**

-BY

*JULAKANTI HARSHITHA*

*(U101115FEC017)*

CONTENTS

1. Introduction
2. Problem Statement:
3. Functioning
4. ER Diagram
5. Conversion of ER to Relational Model
6. MySQL code execution
7. Normalization
8. Conclusion

Introduction:

The database “Online allocation of cab using GPS” is constructed to minimize redundancy. It has been elaborated on further. The aim was to make the existing cab booking system as optimal and inclusive as possible. . We have used MySQL Database in this project.

In the system, the transactions/bookings are done only manually but in proposed system we have to computerize all the transaction/bookings using the software online allocation of cab using this database

We are developing an instance of the back end in order to gain greater understanding and clear our fundamental concepts of database design such as modelling and normalization. Online cab allotment Database Provides information about basic requirements like cab availability, Pre-bookings, online money payment etc.., This database is managed in such a way that respective category will be shown respective needs in certain purpose.

Problem Statement:

A computer based management system is designed to handle the entire primary information required to manage the whole data. Separate database is maintained to handle all the details required for the correct statement calculation and generations. This project intends to introduce more user friendly in the various activities such as record updating, maintenance and searching. The objective and scope of my project Online Cab allocation System is to record the details various activities of user. It will simplify the task and reduce the paper work. To produce a web-based system that allow customer to register and reserve cab online during implementation every user will be given appropriate training to suit their specific needs Training will be provided on a timely basis, and you will be trained as the new is Cab allocation System rolled out to your area of responsibility.

To produce a web-based system that allow customer to register and reserve cab online and for the company to effectively manage their Cab allocating business. To ease customer’s task whenever they need to rent a cab or hire a cab.

Functioning of Online Allocation System:

Online Cab Service acts like a bridge between the cab operators & the customers/ users/ people who book a cab. This is the online cab booking service provided to customers. This bridges together the registration travel agencies/ cab operators/ cab owners & the customers.

A Cab Booking/allocating is a system that can be used temporarily for a period of time with a fee. Allocating a cab assists people to get around even when they do not have access to their own personal vehicle or don't own a vehicle at all. The individual who want to hire/rent a car must first contact the cab allocating company for the desire vehicle, this can be done online. At this point, this person has to supply some information such as: dates of rental, time of rental and type of car. After these details are worked out, the individual renting the car must present a valid Identification Card. Most companies throughout the industry make a profit based of the type of cars. The allocation cabs are categorized into economy, compact, compact premium; premium and luxury & customers are free to choose any car of their choice based on their purse and availability of such car at the time of reservation. Customers can even book a sharing cab which they have to travel with other customers if the both destinations are same.

**The following Online Cab Booking having the following services:-**

1. Enhance Business Processes: To be able to use internet technology to project the rental company to the global world instead of limiting their services to their local domain alone, thus increase their Return on Investment (ROI).

2. Traveller’s registration: A registration portal to hold traveller’s details, monitor the irtransaction and used same to offer better and improve services to them.

3. Group bookings: Allows the customer to book space for a group in the case of weeding or corporate parties or meetings.

4. Eco-friendly: The monitoring of the vehicle activity and the overall business becomes easy and includes the least of paper work.

5. Availability: The software acts as an office that is open 24/7.

6. Efficient: It increases the efficiency of the management at offering quality services to the customers.

7. User friendly: It provides custom features development and support with the software’s.

8. Security: The subsystem should provide a high level of security and integrity of the data held by the system, only authorized personnel of the company can gain access to the company’s secured page on the system; and only users with valid password and username can login to view user’s page.

Online Cab Allotment system follows the Processes-

1. Cab Search - Users can search cab for a particular location here. Users required to enter source, pick up address, & drop address where he wants to go.

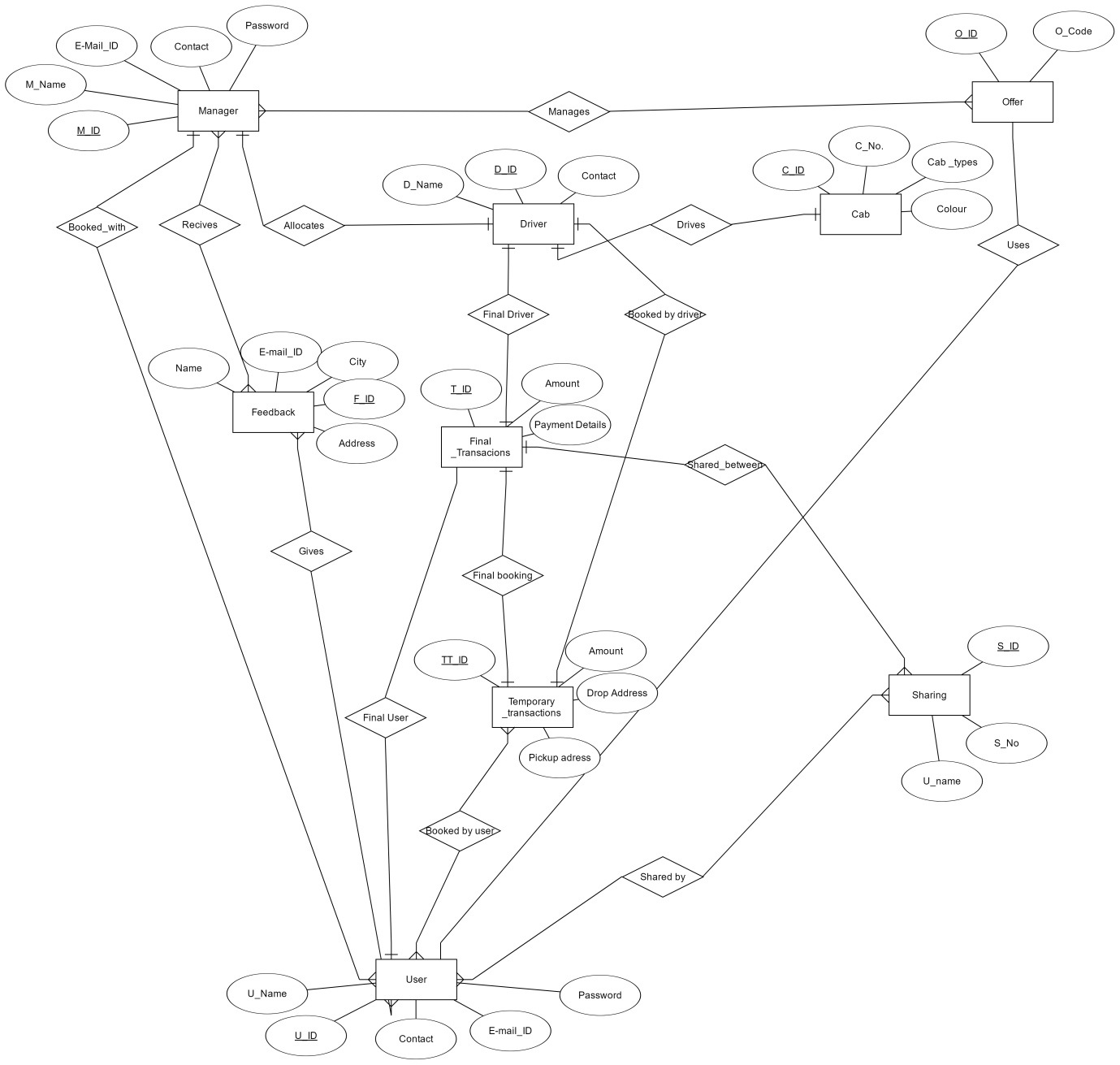
2. Login Search-

In the customers has to give out the login details i.e. users id and password and then only he can be logged on. The user id and password given by the customers are checked from the data stored in the database.

3. Registration Process-

User must be registered before booking a cab. Proper validations will be provided to keep only authenticated users i.e. those users who will provide correct information. All the data supply by the user will be stored in database and it will be used for further validations and authenticated. During registration, users have to give login and password of their choice. Login names and password will be stored in the databases so that the users can directly login without registration again and again

ER DIAGRAM:



**Entities Taken For ER:**

1.Manager: (M\_Name , Password, E-Mail, Contact);

Pri\_Key: (M\_ID);

2.User: (U\_Name);

Pri\_key(U\_ID);

Foreign\_Key: (Password, E-Mail, Contact);

3.Temporary\_Transactions: (Amount ,Pickup Address, Drop address);

Pri\_key: (TT\_ID);

4.Final\_Transactions: (Payment \_Details);

Pri\_key: (T\_ID);

Foreign\_Key: (Amount);

5.Sharing: ( S\_no,);

Pri\_key: (S\_ID);

Foreign\_Key: (U\_Name);

6.Cab: (Cab\_number,Cab\_types,colour);

Pri\_Key: (C\_ID);

7.Drivers: (D\_Name);

Pri\_Key: (D\_ID);

Foreign \_Key(Contact);

8.Feedback: (City,Address);

Pri\_Key: (F\_ID);

Foreign key: (U\_Name,E-mail adrdress);

9.Offers: (O\_code);

Pri\_key: (O\_ID);

RELATIONAL SCHEMA:

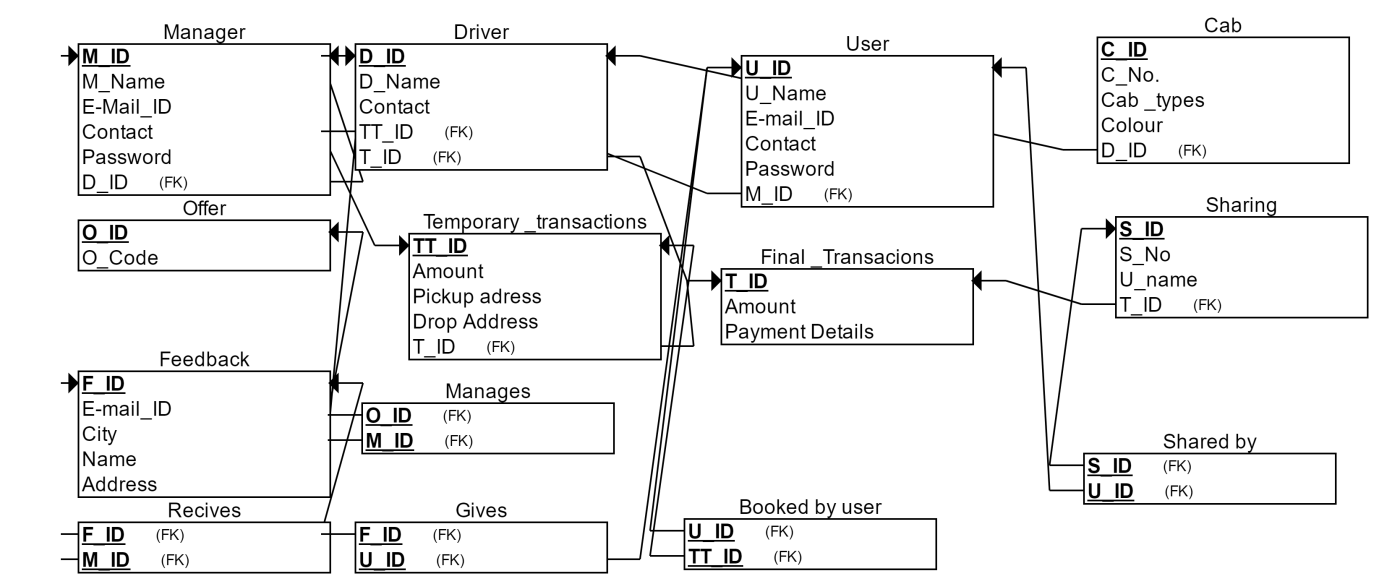


TABLE SCHEMA:

Manager (M\_ID,M\_Name,E-Mail\_ID,Contact,Password)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| M\_ID | M\_Name | E-Mail\_ID | Contact | Password |

User (U\_ID,U\_Name,E-Mail \_ID,Contact,Password)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| U\_ID | U\_Name | E-Mail\_ID | Contact | Password |

Temporary \_Transactions (TT\_ID,Amount,PickUP\_Address,Drop\_Address)

|  |  |  |  |
| --- | --- | --- | --- |
| TT\_ID | Amount | Pickup\_Address | Drop\_Address |

Fina l\_Transactions(T\_ID,Amount,Payment Details)

|  |  |  |
| --- | --- | --- |
| T\_ID | Amount | Payment\_Details |

Sharing (S\_ID,S\_No,S\_Name)

|  |  |  |
| --- | --- | --- |
| S\_ID | S\_No | U\_Name |

Cab (C\_ID,C\_No,Cab\_Types,Colour)

|  |  |  |  |
| --- | --- | --- | --- |
| C\_ID | C\_No | Cab\_Types | Colour |

Drivers(D\_ID,D\_Name,Contact)

|  |  |  |
| --- | --- | --- |
| D\_ID | D\_Name | Contact |

Feedback ( F\_ID,U\_Name,Address,City,E-mail\_ID,)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| F\_ID | U\_Name | Address | City | E-Mail\_ID |

Offers (O\_ID,O\_Code)

|  |  |
| --- | --- |
| O\_ID | O\_Code |

MySQL Executable code for Tables:

CREATE TABLE Offer

(

O\_Code INT NOT NULL,

O\_ID INT NOT NULL,

PRIMARY KEY (O\_ID)

);

CREATE TABLE Final\_\_Transacions

(

T\_ID INT NOT NULL,

Amount INT NOT NULL,

Payment\_Details INT NOT NULL,

PRIMARY KEY (T\_ID)

);

CREATE TABLE Sharing

(

S\_ID INT NOT NULL,

S\_No INT NOT NULL,

U\_name INT NOT NULL,

T\_ID INT NOT NULL,

PRIMARY KEY (S\_ID),

FOREIGN KEY (T\_ID) REFERENCES Final\_\_Transacions(T\_ID)

);

CREATE TABLE Feedback

(

E-mail\_ID INT NOT NULL,

City INT NOT NULL,

Name INT NOT NULL,

Address INT NOT NULL,

F\_ID INT NOT NULL,

PRIMARY KEY (F\_ID)

);

CREATE TABLE Temporary\_\_transactions

(

TT\_ID INT NOT NULL,

Amount INT NOT NULL,

Pickup\_adress INT NOT NULL,

Drop\_Address INT NOT NULL,

T\_ID INT NOT NULL,

PRIMARY KEY (TT\_ID),

FOREIGN KEY (T\_ID) REFERENCES Final\_\_Transacions(T\_ID)

);

CREATE TABLE Driver

(

D\_ID INT NOT NULL,

D\_Name INT NOT NULL,

Contact INT NOT NULL,

TT\_ID INT NOT NULL,

T\_ID INT NOT NULL,

PRIMARY KEY (D\_ID),

FOREIGN KEY (TT\_ID) REFERENCES Temporary\_\_transactions(TT\_ID),

FOREIGN KEY (T\_ID) REFERENCES Final\_\_Transacions(T\_ID)

);

CREATE TABLE Cab

(

C\_ID INT NOT NULL,

C\_No. INT NOT NULL,

Cab\_\_types INT NOT NULL,

Colour INT NOT NULL,

D\_ID INT NOT NULL,

PRIMARY KEY (C\_ID),

FOREIGN KEY (D\_ID) REFERENCES Driver(D\_ID)

);

CREATE TABLE Manager

(

M\_ID INT NOT NULL,

M\_Name INT NOT NULL,

E-Mail\_ID INT NOT NULL,

Contact INT NOT NULL,

Password INT NOT NULL,

D\_ID INT NOT NULL,

PRIMARY KEY (M\_ID),

FOREIGN KEY (D\_ID) REFERENCES Driver(D\_ID)

);

CREATE TABLE User

(

U\_ID INT NOT NULL,

U\_Name INT NOT NULL,

E-mail\_ID INT NOT NULL,

Contact INT NOT NULL,

Password INT NOT NULL,

M\_ID INT NOT NULL,

PRIMARY KEY (U\_ID),

FOREIGN KEY (M\_ID) REFERENCES Manager(M\_ID)

);

|  |  |  |
| --- | --- | --- |
| **ENTITIES** | **SQL SYNTAX** | |
| *MANAGER* | M\_ID INT NOT NULL,  M\_Name INT NOT NULL,  E-Mail\_ID INT NOT NULL,  Contact INT NOT NULL,  Password INT NOT NULL, | |
| *USER* | U\_ID INT NOT NULL,  U\_Name INT NOT NULL,  E-mail\_ID INT NOT NULL,  Contact INT NOT NULL,  Password INT NOT NULL, | |
| *TEMPORARY\_TRANSACTIONS* | Amount INT NOT NULL,  Pickup\_adress INT NOT NULL,  Drop\_Address INT NOT NULL,  T\_ID INT NOT NULL, | |
| **ENTITIES** | | **SQL SYNTAX** | |
| *FINAL\_TRASACTIONS* | | T\_ID INT NOT NULL,  Amount INT NOT NULL,  Payment\_Details INT NOT NULL | |
| *CAB* | | C\_ID INT NOT NULL,  C\_No. INT NOT NULL,  Cab\_\_types INT NOT NULL,  Colour INT NOT NULL, | |
| *DRIVER* | | D\_ID INT NOT NULL,  D\_Name INT NOT NULL,  Contact INT NOT NULL, | |
| *SHARING* | | S\_ID INT NOT NULL,  S\_No INT NOT NULL,  U\_name INT NOT NULL, | |
| *FEEDBACK* | | E-mail\_ID INT NOT NULL,  City INT NOT NULL,  Name INT NOT NULL,  Address INT NOT NULL,  F\_ID INT NOT NULL, | |
| *OFFERS* | | O\_Code INT NOT NULL,  O\_ID INT NOT NULL, | |

FUNCTIONAL DEPENDENCIES:

**Manager**

M\_ID -> M\_Name,Password; M\_Name -> E-Mail\_ID,Contact,Password; E-Mail -> Password

**User**

U\_Id -> U\_Name,Password; U\_Name -> E-Mail\_ID,Contact,Password ;E-Mail\_id -> Password

**Temporary\_Transaction**

TT\_ID ->Amount,Pickup\_Address,Drop\_Address

**Final\_Transaction**

T\_ID -> Amount,Details

**Sharing**

S\_ID ->S\_No.

**Cab**

C\_ID -> C\_No.,;C\_No ->Cab\_Types,Colour

**Drivers**

D\_ID ->D\_Name

**Feedback**

F\_ID -> U\_Name; U\_Name -> Address,City,E-Mail\_ID ;Address ->City

**Offers**

O\_ID -> O\_Code

NORMALIZATION:

Normalisation is a technique of organizing the data in the database. It is a systematic approach

of decomposing tables used for mainly two purposes -

 Eliminating redundant (useless) data and undesirable characteristics like Insertion,

Update and Deletion Anamolies.

 Ensuring data dependencies make sense i.e data is logically stored.

It is a multi-step process that puts data into tabular form by removing duplicated data from the

relation tables. Without Normalization, it becomes difficult to handle and update the database,

without facing data loss.

Our goals of database design with functional dependencies are:

1. 3NF

2. BCNF

**NORMALIZATION OF EACH TABLE IN DATABASE:**

1. Manager (M\_ID,M\_Name,E-Mail\_ID,Contact,Password)

This is in 2NF form.

Following Steps to be followed to convert into 3NF:

Step 1: Find the minimal cover of FDs, which contains   
M\_ID --> M\_Name  
M\_Name --> E-Mail\_ID  
M\_Name --> Contact  
E-Mail\_ID --> Password

Step 2. Find all candidate keys. The set of candidates keys is { (M\_ID), }.  
The set of key attributes is: { M\_ID }.

Step 3: Merge FDs with same LHS and whose RHS are non-key attributes, we get the set F1 which contains:  
M\_ID --> M\_Name  
M\_Name --> Contact,E-Mail\_ID  
E-Mail\_ID --> Password

Step 4: Check each FD in the set F1 for violation of 3NF, and split table accordingly.   
  
  
Checking FD M\_ID --> M\_Name  
FD does not violate 3NF  
Checking FD M\_Name --> Contact,E-Mail\_ID  
The FD violates 3NF as its LHS is not a superkey (and RHS is a set of non-key attributes).   
The following 3NF table is obtained:   
  
M\_Name,Contact,E-Mail\_ID  
with FDs   
M\_Name --> E-Mail\_ID,Contact  
  
  
  
Checking FD E-Mail\_ID --> Password  
The FD violates 3NF as its LHS is not a superkey (and RHS is a set of non-key attributes).   
The following 3NF table is obtained:   
  
E-Mail\_ID,Password  
with FDs   
E-Mail\_ID --> Password

Step 5: Finally, add the following table into normalized 3NF table set (obtained by removing RHS attributes of FDs using which we produced a table):   
  
M\_ID,M\_Name  
with FDs   
M\_ID --> M\_Name

1. User (U\_ID,U\_Name,E-Mail \_ID,Contact,Password)

This is in the form of 2NF.

Following Steps to be followed to convert into 3NF:

Step 1: Find the minimal cover of FDs, which contains   
U\_ID --> U\_Name  
U\_Name --> E-Mail\_ID  
U\_Name --> Contact  
E-Mail\_ID --> Password

Step 2. Find all candidate keys. The set of candidates keys is { (U\_ID), }.  
The set of key attributes is: { U\_ID }.

Step 3: Merge FDs with same LHS and whose RHS are non-key attributes, we get the set F1 which contains:  
U\_ID --> U\_Name  
U\_Name --> Contact,E-Mail\_ID  
E-Mail\_ID --> Password

Step 4: Check each FD in the set F1 for violation of 3NF, and split table accordingly.   
  
  
Checking FD U\_ID --> U\_Name  
FD does not violate 3NF  
Checking FD U\_Name --> Contact,E-Mail\_ID  
The FD violates 3NF as its LHS is not a superkey (and RHS is a set of non-key attributes).   
The following 3NF table is obtained:   
  
U\_Name,Contact,E-Mail\_ID  
with FDs   
U\_Name --> E-Mail\_ID,Contact  
  
  
  
Checking FD E-Mail\_ID --> Password  
The FD violates 3NF as its LHS is not a superkey (and RHS is a set of non-key attributes).   
The following 3NF table is obtained:   
  
E-Mail\_ID,Password  
with FDs   
E-Mail\_ID --> Password

Step 5: Finally, add the following table into normalized 3NF table set (obtained by removing RHS attributes of FDs using which we produced a table):   
  
U\_ID,U\_Name  
with FDs   
U\_ID --> U\_Name

1. Temporary \_Transactions (TT\_ID,Amount,PickUP\_Address,Drop\_Address)

This entity is already in 2NF,3NF,BCNF.No need to convert.

1. Final\_Transactions(T\_ID,Amount,Payment Details)

This entity is already in 2NF,3NF,BCNF.No need to convert.

1. Sharing (S\_ID,S\_No,S\_Name)

This entity is in 1NF.

Following Steps to be followed to convert into 2NF,3NF:

2NF: First, find the minimal cover of the FDs, which includes the FDs   
S\_ID --> S\_No  
  
Initially rel[1] is the original table:

Step1: checking table rel[1]   
  
The table is not in 2NF.  
The FD [S\_ID --> S\_No] is a partial dependency (i.e., LHS is a proper subset of some CK), the table is split into:   
  
rel[2] = (S\_ID,S\_No), with FDs:   
S\_ID --> S\_No  
  
rel[3] = (S\_ID,S\_Name), with FDs:

Step2: checking table rel[2]   
  
The table is in 2NF already, send it to output

Step3: checking table rel[3]   
  
The table is in 2NF already, send it to output

3NF:

Step 1: Find the minimal cover of FDs, which contains   
S\_ID --> S\_No

Step 2. Find all candidate keys. The set of candidates keys is { (S\_ID,S\_Name), }.  
The set of key attributes is: { S\_ID,S\_Name }.

Step 3: Merge FDs with same LHS and whose RHS are non-key attributes, we get the set F1 which contains:  
S\_ID --> S\_No

Step 4: Check each FD in the set F1 for violation of 3NF, and split table accordingly.   
  
  
Checking FD S\_ID --> S\_No  
The FD violates 3NF as its LHS is not a superkey (and RHS is a set of non-key attributes).   
The following 3NF table is obtained:   
  
S\_ID,S\_No  
with FDs   
S\_ID --> S\_No

Step 5: Finally, add the following table into normalized 3NF table set (obtained by removing RHS attributes of FDs using which we produced a table):   
  
S\_ID,S\_Name  
with FDs

1. Cab (C\_ID,C\_No,Cab\_Types,Colour)

This is in the form of 2NF.

Following Steps to be followed to convert into 3NF:

Step 1: Find the minimal cover of FDs, which contains   
C\_ID --> C\_No  
C\_No --> Cab\_Types  
C\_No --> Colour

Step 2. Find all candidate keys. The set of candidates keys is { (C\_ID), }.  
The set of key attributes is: { C\_ID }.

Step 3: Merge FDs with same LHS and whose RHS are non-key attributes, we get the set F1 which contains:  
C\_ID --> C\_No  
C\_No --> Colour,Cab\_Types

Step 4: Check each FD in the set F1 for violation of 3NF, and split table accordingly.   
  
  
Checking FD C\_ID --> C\_No  
FD does not violate 3NF  
Checking FD C\_No --> Colour,Cab\_Types  
The FD violates 3NF as its LHS is not a superkey (and RHS is a set of non-key attributes).   
The following 3NF table is obtained:   
  
C\_No,Colour,Cab\_Types  
with FDs   
C\_No --> Colour,Cab\_Types

Step 5: Finally, add the following table into normalized 3NF table set (obtained by removing RHS attributes of FDs using which we produced a table):   
  
C\_ID,C\_No  
with FDs   
C\_ID --> C\_No

1. Drivers(D\_ID,D\_Name,Contact)

This entity is in 1NF.

Following Steps to be followed to convert into 2NF,3NF:

2NF:

First, find the minimal cover of the FDs, which includes the FDs   
D\_ID --> D\_Name  
  
Initially rel[1] is the original table:

Round1: checking table rel[1]   
  
The table is not in 2NF.  
The FD [D\_ID --> D\_Name] is a partial dependency (i.e., LHS is a proper subset of some CK), the table is split into:   
  
rel[2] = (D\_ID,D\_Name), with FDs:   
D\_ID --> D\_Name  
  
rel[3] = (D\_ID,Contact), with FDs:

Round2: checking table rel[2]   
  
 The table is in 2NF already, send it to output

Round3: checking table rel[3]   
  
The table is in 2NF already, send it to output

3NF:

Step 1: Find the minimal cover of FDs, which contains   
D\_ID --> D\_Name

Step 2. Find all candidate keys. The set of candidates keys is { (D\_ID,Contact), }.  
The set of key attributes is: { D\_ID,Contact }.

Step 3: Merge FDs with same LHS and whose RHS are non-key attributes, we get the set F1 which contains:  
D\_ID --> D\_Name

Step 4: Check each FD in the set F1 for violation of 3NF, and split table accordingly.   
  
  
Checking FD D\_ID --> D\_Name  
The FD violates 3NF as its LHS is not a superkey (and RHS is a set of non-key attributes).   
The following 3NF table is obtained:   
  
D\_ID,D\_Name  
with FDs   
D\_ID --> D\_Name

Step 5: Finally, add the following table into normalized 3NF table set (obtained by removing RHS attributes of FDs using which we produced a table):   
  
D\_ID,Contact  
with FDs

1. Feedback ( F\_ID,U\_Name,Address,City,E-mail\_ID,)

This is in the form of 2NF.

Following Steps to be followed to convert into 3NF:

Step 1: Find the minimal cover of FDs, which contains   
F\_ID --> U\_Name  
U\_Name --> Address  
U\_Name --> E-mail\_ID  
Address --> City

Step 2. Find all candidate keys. The set of candidates keys is { (F\_ID), }.  
The set of key attributes is: { F\_ID }.

Step 3: Merge FDs with same LHS and whose RHS are non-key attributes, we get the set F1 which contains:  
F\_ID --> U\_Name  
U\_Name --> E-mail\_ID,Address  
Address --> City

Step 4: Check each FD in the set F1 for violation of 3NF, and split table accordingly.   
  
  
Checking FD F\_ID --> U\_Name  
FD does not violate 3NF  
Checking FD U\_Name --> E-mail\_ID,Address  
The FD violates 3NF as its LHS is not a superkey (and RHS is a set of non-key attributes).   
The following 3NF table is obtained:   
  
U\_Name,E-mail\_ID,Address  
with FDs   
U\_Name --> Address,E-mail\_ID  
  
  
  
Checking FD Address --> City  
The FD violates 3NF as its LHS is not a superkey (and RHS is a set of non-key attributes).   
The following 3NF table is obtained:   
  
Address,City  
with FDs   
Address --> City

1. Offers (O\_ID,O\_Code)

This entity is already in 2NF,3NF,BCNF. No need to convert.

SAMPLE OUTPUTS WITH SCREENSHOTS

**1.MANAGER**

CREATE TABLE Manager

(

M\_ID INT NOT NULL,

M\_Name INT NOT NULL,

E-Mail\_ID INT NOT NULL,

Contact INT NOT NULL,

Password INT NOT NULL,

D\_ID INT NOT NULL,

PRIMARY KEY (M\_ID),

FOREIGN KEY (D\_ID) REFERENCES Driver(D\_ID)

);

INSERT INTO Manager VALUES(601,RAVI,ravi@gmail.com,936-3624,99999,401,)

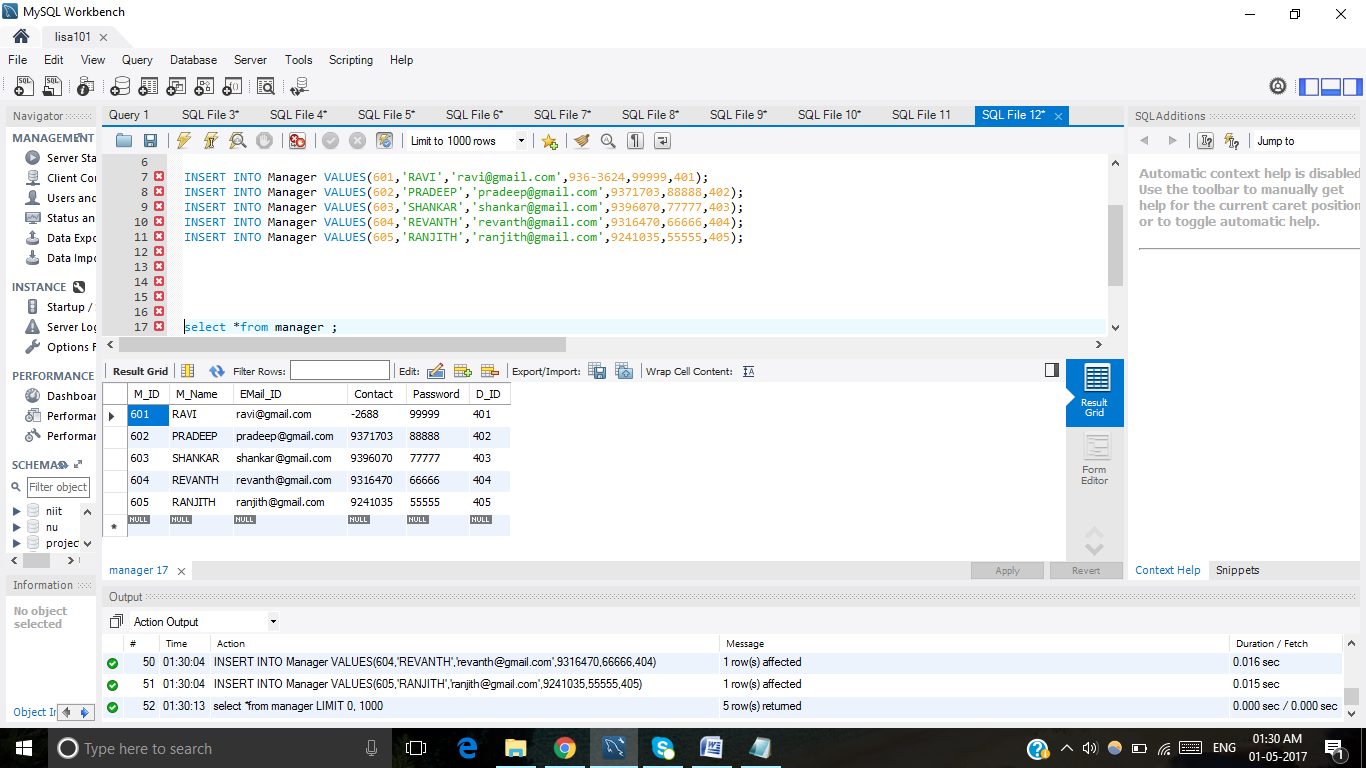
INSERT INTO Manager VALUES(602,PRADEEP,pradeep@gmail.com,937-1703,88888,402)

INSERT INTO Manager VALUES(603,SHANKAR,shankar@gmail.com,939-6070,77777,403)

INSERT INTO Manager VALUES(604,REVANTH,revanth@gmail.com,931-6470,66666,404)

INSERT INTO Manager VALUES(605,RANJITH,ranjith@gmail.com,924-1035,55555,405)

Select\*from Manager

****

**2.USER**

CREATE TABLE User

(

U\_ID INT NOT NULL,

U\_Name INT NOT NULL,

E-mail\_ID INT NOT NULL,

Contact INT NOT NULL,

Password INT NOT NULL,

M\_ID INT NOT NULL,

PRIMARY KEY (U\_ID),

FOREIGN KEY (M\_ID) REFERENCES Manager(M\_ID)

);

INSERT INTI User VALUES(701,VINAY,vinay@gmail.com,723-0290,44444,601)

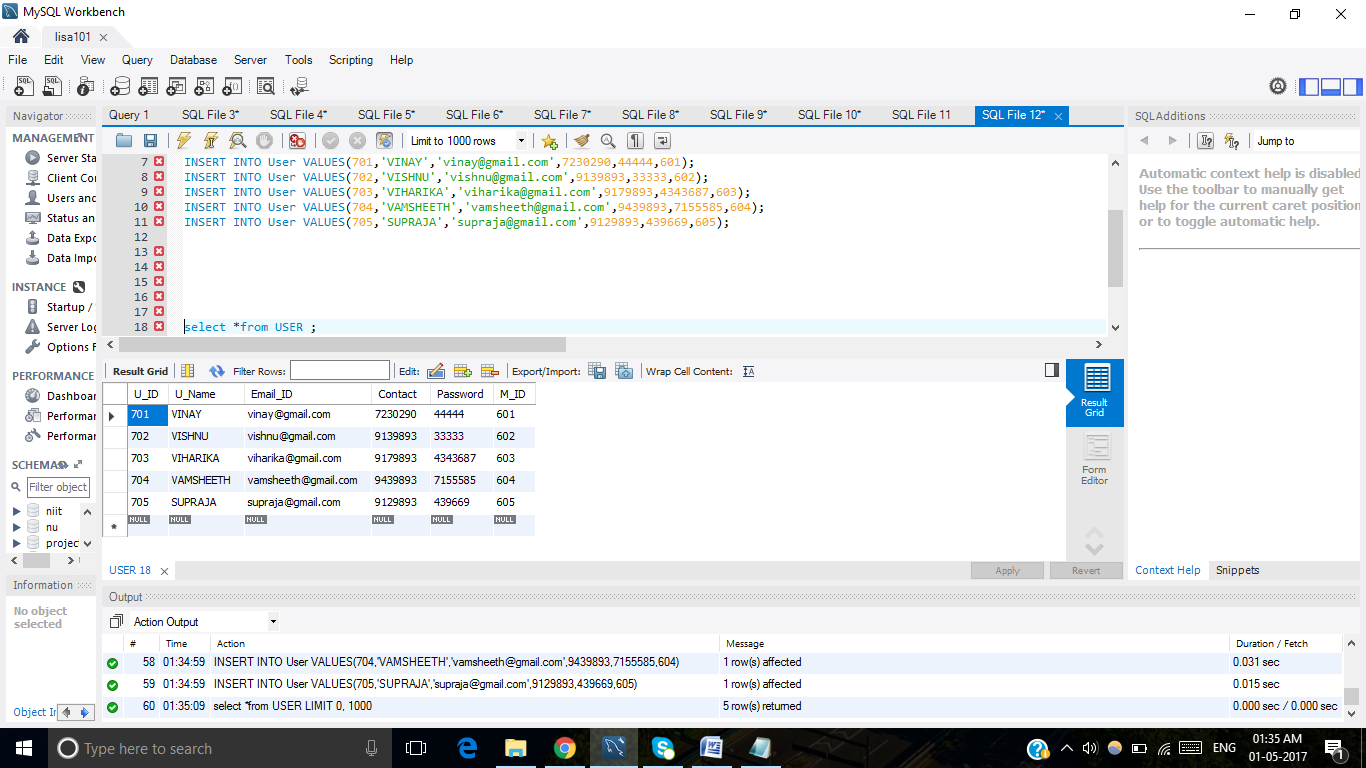
INSERT INTI User VALUES(702,VISHNU,vishnu@gmail.com,913-9893,33333,602)

INSERT INTI User VALUES(703,VIHARIKA,viharika@gmail.com,434-3687,603)

INSERT INTI User VALUES(704,VAMSHEETH,vamsheeth@gmail.com,715-5585,604)

INSERT INTI User VALUES(705,SUPRAJA,supraja@gmail.com,439-1669)

Select\*from User



**3.Temporary\_Transaction**

CREATE TABLE Temporary\_\_transactions

(

TT\_ID INT NOT NULL,

Amount INT NOT NULL,

Pickup\_adress INT NOT NULL,

Drop\_Address INT NOT NULL,

T\_ID INT NOT NULL,

PRIMARY KEY (TT\_ID),

FOREIGN KEY (T\_ID) REFERENCES Final\_\_Transacions(T\_ID)

);

INSERT INTO Temporary\_\_transactios VALUES(301,1000,NEEMRANA,SAROJINI\_BAZAR,001)

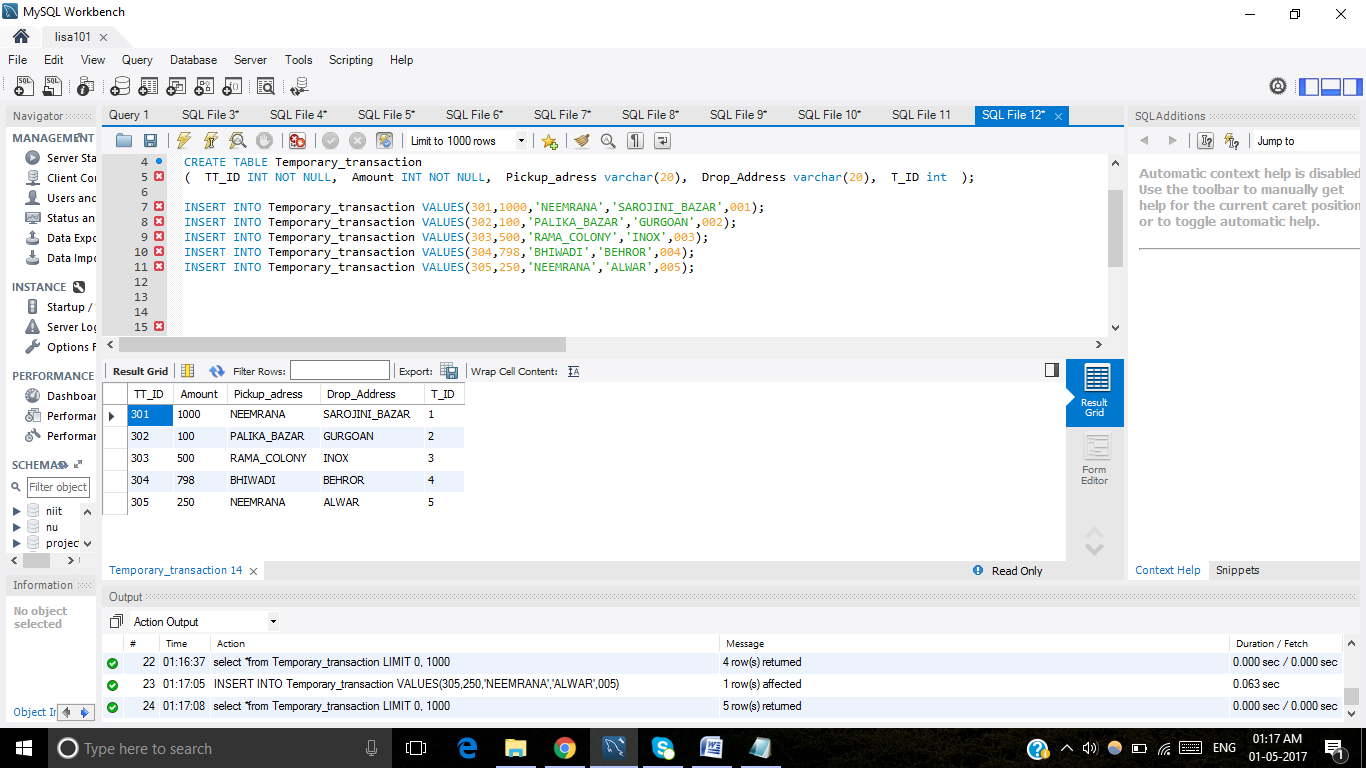
INSERT INTO Temporary\_\_transactios VALUES(302,100,PALIKA\_BAZAR,GURGOAN,002)

INSERT INTO Temporary\_\_transactios VALUES(303,500,RAMA\_COLONY,INOX,003)

INSERT INTO Temporary\_\_transactios VALUES(304,798,BHIWADI,BEHROR,004)

INSERT INTO Temporary\_\_transactios VALUES(305,250,NEEMRANA,ALWAR,005)

Select\*from Temporary\_Transactions



**4.Final\_Transaction**

CREATE TABLE Final\_\_Transacions

(

T\_ID INT NOT NULL,

Amount INT NOT NULL,

Payment\_Details INT NOT NULL,

PRIMARY KEY (T\_ID)

);

INSERT INTO Final\_\_Transacions VALUES(101,1000,PAID,)

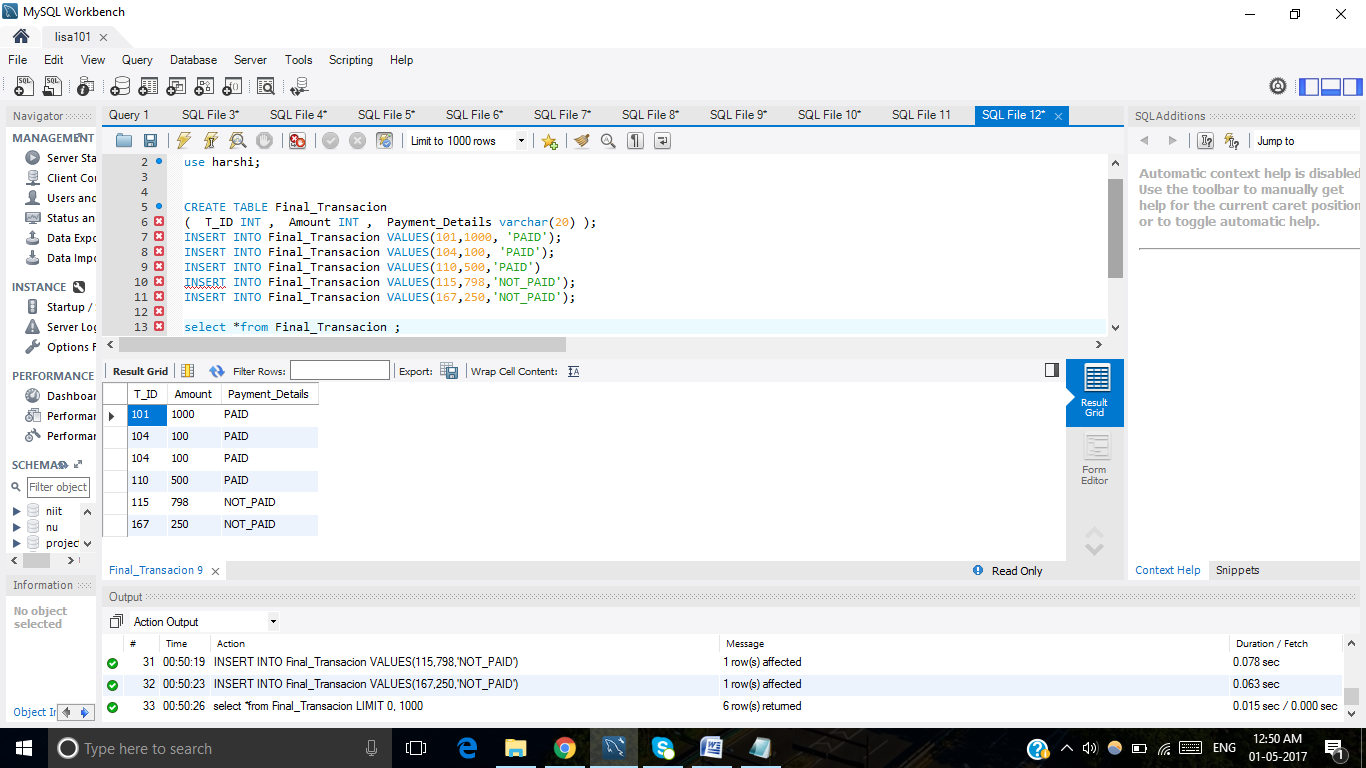
INSERT INTO Final\_\_Transacions VALUES(104,100,PAID)

INSERT INTO Final\_\_Transacions VALUES(110,500,PAID)

INSERT INTO Final\_\_Transacions VALUES(115,798,NOT PAID)

INSERT INTO Final\_\_Transacions VALUES(167,250,NOT PAID)

Select\*from Final\_Transactions



**5.SHARING**

CREATE TABLE Sharing

(

S\_ID INT NOT NULL,

S\_No INT NOT NULL,

U\_name INT NOT NULL,

T\_ID INT NOT NULL,

PRIMARY KEY (S\_ID),

FOREIGN KEY (T\_ID) REFERENCES Final\_\_Transacions(T\_ID)

);

INSERT INTO sharing VALUES(001,AP2120,VINAY,101)

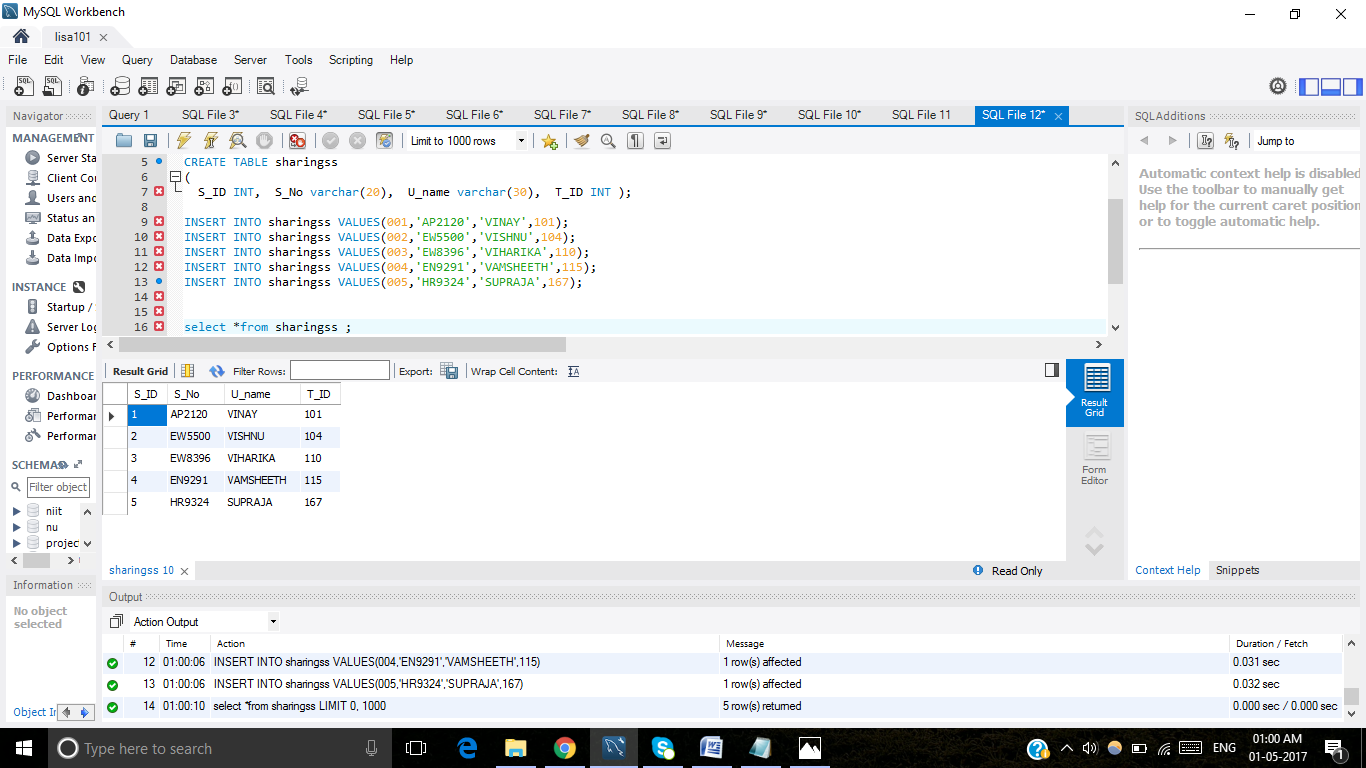
INSERT INTO sharing VALUES(002,EW5500,VISHNU,104)

INSERT INTO sharing VALUES(003,EW8396,VIHARIKA,110)

INSERT INTO sharing VALUES(004,EN9291,VAMSHEETH,115)

INSERT INTO sharing VALUES(005,HR9324,SUPRAJA,167)

Select\*from Sharing



**6.CAB**

CREATE TABLE Cab

(

C\_ID INT NOT NULL,

C\_No. INT NOT NULL,

Cab\_\_types INT NOT NULL,

Colour INT NOT NULL,

D\_ID INT NOT NULL,

PRIMARY KEY (C\_ID),

FOREIGN KEY (D\_ID) REFERENCES Driver(D\_ID)

);

INSERT INTO Cab VALUES (501,WB5977,4-SEATER,BLUE,401,)

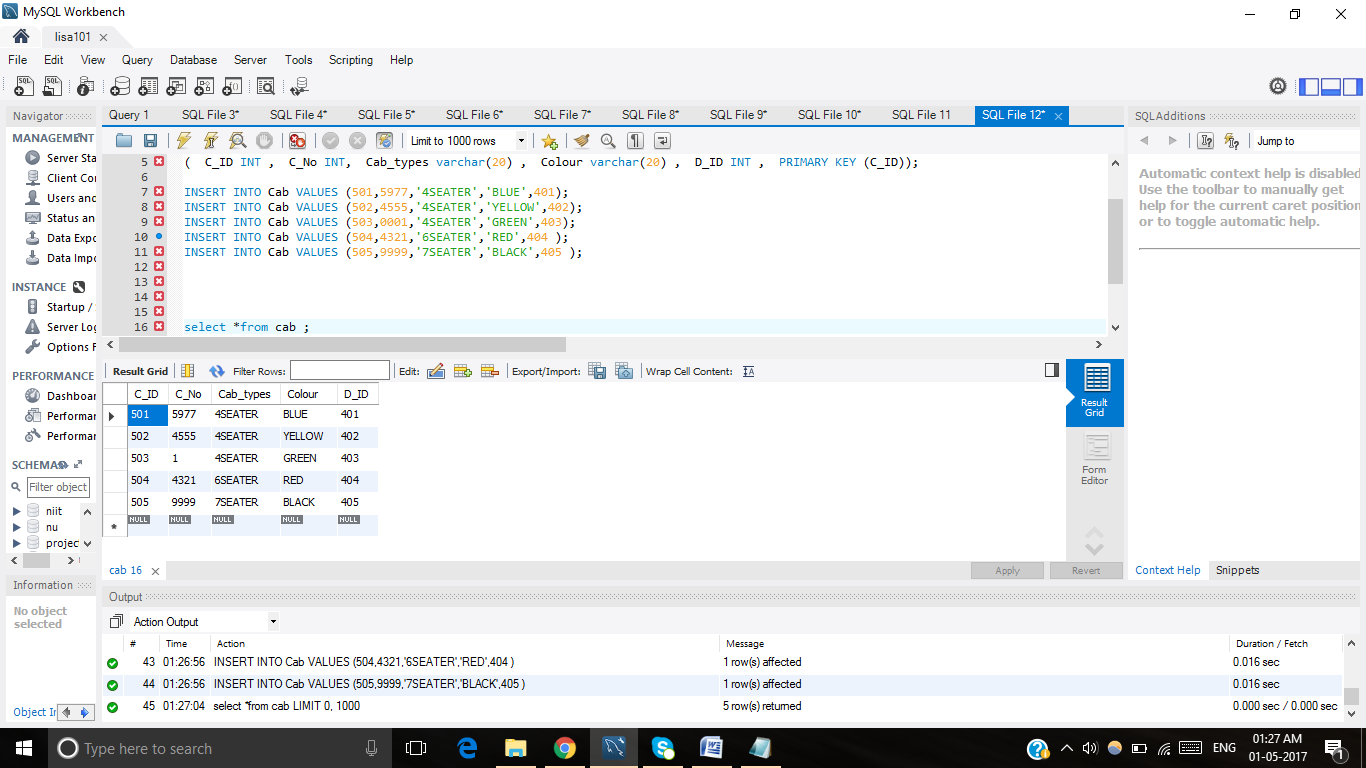
INSERT INTO Cab VALUES (502,IM4555,4-SEATER,YELLOW,402)

INSERT INTO Cab VALUES (503,CC0001,4-SEATER,GREEN,403)

INSERT INTO Cab VALUES (504,XC4321,6-SEATER,RED,404 )

INSERT INTO Cab VALUES (505,WY9999,7-SEATER,BLACK,405 )

Select\*from Cab



**7.DRIVER**

CREATE TABLE Driver

(

D\_ID INT NOT NULL,

D\_Name INT NOT NULL,

Contact INT NOT NULL,

TT\_ID INT NOT NULL,

T\_ID INT NOT NULL,

PRIMARY KEY (D\_ID),

FOREIGN KEY (TT\_ID) REFERENCES Temporary\_\_transactions(TT\_ID),

FOREIGN KEY (T\_ID) REFERENCES Final\_\_Transacions(T\_ID)

);

INSERT INTO Driver(401,RAMA,712-5757,301,001)

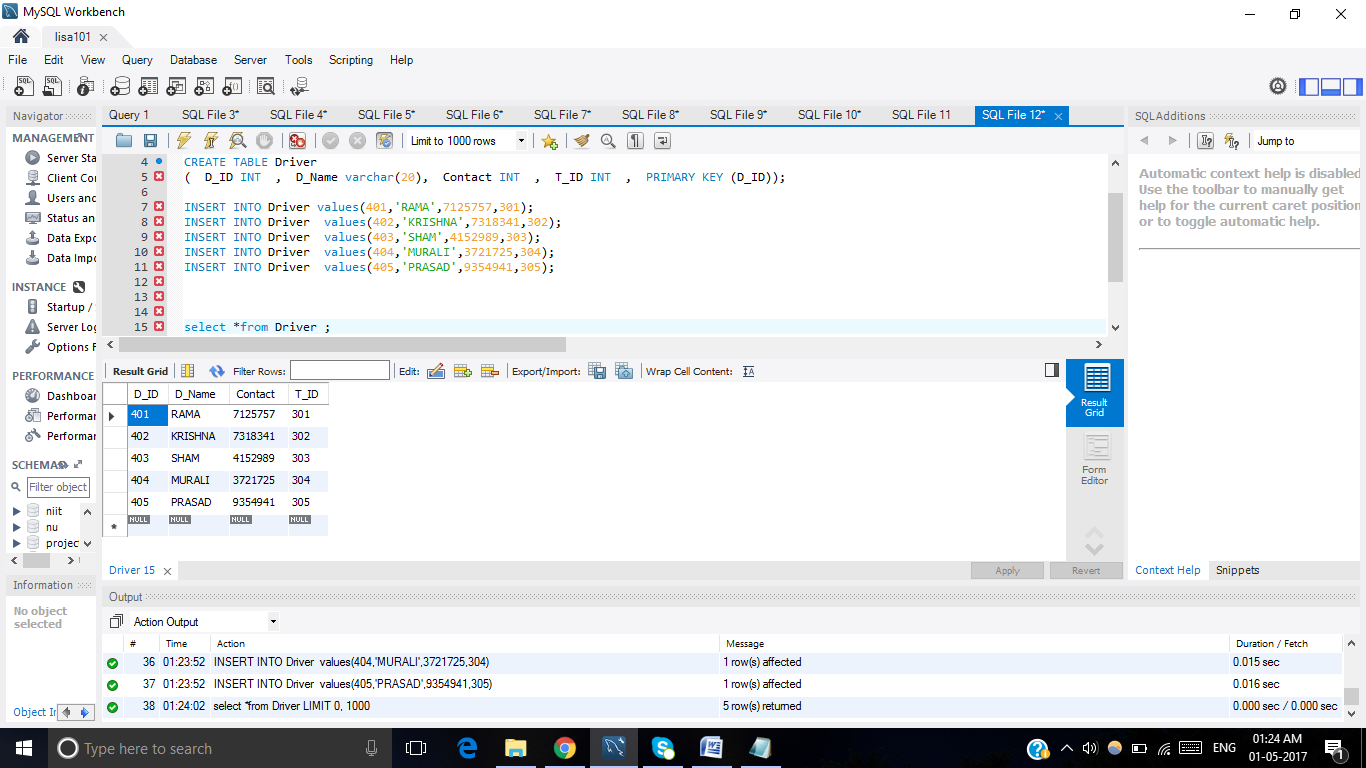
INSERT INTO Driver(402,KRISHNA,731-8341,302,002)

INSERT INTO Driver(403,SHAM,411-2989,303,003)

INSERT INTO Driver(404,MURALI,372-1725,304,004)

INSERT INTO Driver(405,PRASAD,935-4941,305,005)

Select\*from Driver



**8. FEEDBACK**

CREATE TABLE Feedback

(

E-mail\_ID INT NOT NULL,

City INT NOT NULL,

U\_Name INT NOT NULL,

Address INT NOT NULL,

F\_ID INT NOT NULL,

PRIMARY KEY (F\_ID)

);

INSERT INTO Feedback VALUES(vinay@gmail.com,PHOENIX,VINAY,1107,201)

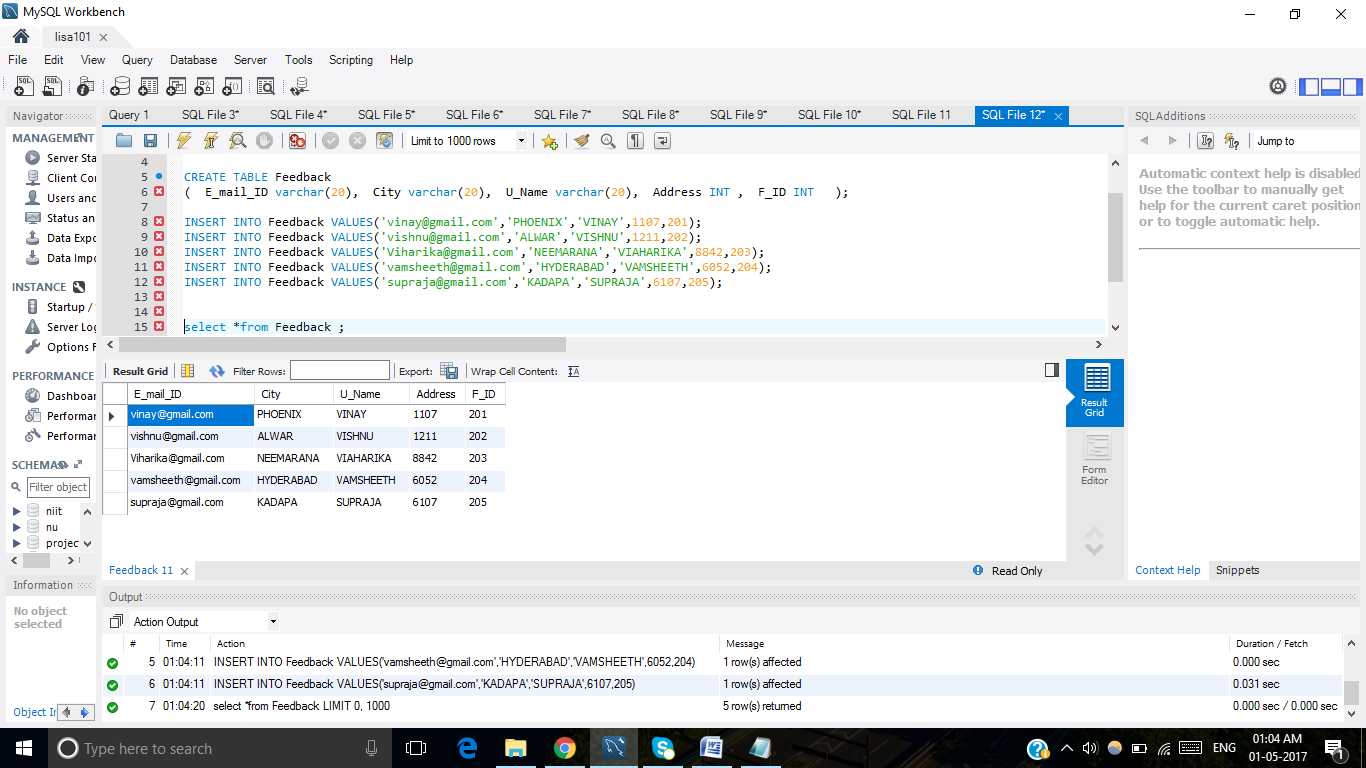
INSERT INTO Feedback VALUES(vishnu@gmail.com,ALWAR,VISHNU,1211,202)

INSERT INTO Feedback VALUES(Viharika@gmail.com,NEEMARANA,VIAHARIKA,8842,203)

INSERT INTO Feedback VALUES(vamsheeth@gmail.com,HYDERABAD,VAMSHEETH,6052,204)

I NSERT INTO Feedback VALUES(supraja@gmail.com,KADAPA,SUPRAJA,6107,205)

Select\*from Feedback



**9.OFFER**

CREATE TABLE Offer

(

O\_Code INT NOT NULL,

O\_ID INT NOT NULL,

PRIMARY KEY (O\_ID)

);

INSERT INTO Offer VALUES(01,100)

INSERT INTO Offer VALUES(02,500)

INSERT INTO Offer VALUES(03,750)

INSERT INTO Offer VALUES(04,1000)

INSERT INTO Offer VALUES(05,1500)

Select\*from Offer

