A Project Report on

Community Management System

In partial fulfilment for the award of the degree

Of

Master's

In

Information System

Submitted By
Aalap Doshi (W1353286)
Nishant Singh (W1353007)
Jui Shaligram (W1360842)

Guided By

Prof. Shailesh Agarwal

Information Systems Department
Leavey School of Business, Santa Clara University
500 El Camino Real

2017-2018

ACKNOWLEDGEMENT

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. We would like to extend our sincere thanks to all of them.

We are highly indebted to **Prof. Shailesh Agarwal** for their guidance and constant supervision as well as for providing necessary information regarding the project. We take this opportunity to thank all our friends and colleagues who started us out on the topic and provided extremely useful review feedback and for their all-time support and help in each and every aspect of the course of our project preparation. We are grateful to our college Santa Clara University, Santa Clara, CA for providing us all required resources and good working environment.

Aalap Doshi (W1353286)
Nishant Singh (W1353007)
Jui Shaligram (W1360842)

Santa Clara University

[UNDERTAKING ABOUT ORIGINALITY OF WORK]

We hereby certify that we are the sole authors of this project report and that neither any part of this project report nor the whole of the Project report has been submitted for a degree by other students to any other University or Institution.

We certify that, to the best of our knowledge, the current Project report does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations or any other material from the work of other people included in our Project report, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that we have included copyrighted material that surpasses the boundary of fair dealing within the meaning of the American Copyright (Amendment) Act 2012, we certify that we have obtained a written permission from the copyright owner to include such materials in the current Project report and have included copies of such copyright clearances to our appendix.

We have checked the write up of the present Project report using anti-plagiarism database and it is in the allowable limit. In case of any complaints pertaining to plagiarism, we certify that we shall be solely responsible for the same and we understand that as per norms, University can even revoke Master's degree conferred upon the students submitting this Project report, in case it is found to be plagiarised.

Table of Contents

Chapter:1 Introduction

1.1Business Need

- People are social creatures by nature.
- They need some kind of dependency in their lives.
- They crave interaction.
- They prefer an environment where they can live peacefully.
- People need a feel of commonness in the same community.
- People tend to share their life through various events and communication process.

1.2 Business Description

- Community management system will allow basically all the household activities to be done online via the system. People can do all the process just a click away with very easy processes.
- A person can also give its feedbacks and share various things and put up ads in the community management system.

1.3 Business Requirements

- Platform to manage rents and lease online
- Raise and resolve maintenance requests through an online recording and managing system.
- Reserve amenities, plan social events and provide feedback via the system sitting at home.
- Check for apartment availability.
- A platform to enhance lifestyle.

1.4 Project Detail:

Project Name:	Community Management System
Developed by	DataHolics
Team Size	3 Members
Documentation	MS Word, MS Visio, MS Excel, Prezi
Tools	SQLite
Back End	SQLite
Internal Guide	Prof. Shailesh Agarwal

1.5 Business Value

- Ease of the process for House related things as everything can be done online.
- Ease of reporting as you can report and provide feedback online.
- Repeating of complains as reduced as you have the record of doing it through the system.
- People in the same community can find a way to easily communicate and contact with each other.
- Raising the number of requests, feedbacks also number of amenities bookings through the CMS.

1.6 Business Metrics

1) **Lease Details**: Managers will be able to view how many leases are starting every year. This helps the community to analyse their marketing strategies and develop them as per needed.

2) **Prospective Resident Analysis**: A manger can see number of prospective residents converted to new residents in CMS which lets him know if their marketing strategies are having either a positive or a negative effect and what can they do more.

3) **Service Request Business Query**: This would help the Manager to understand that how many queries for Service request are generated in different categories grouping either by years or months. This gives him an idea of which category needs more maintenance and he can allot resources then accordingly.

4) **Feedback Business Query**: This further helps the management to understand about the feedback and the source from which they are generated or the mode from which they were communicated.

5) **Amenities Business Query** This allows the staff to understand via which medium the amenity is booked and help them to promote the usage for those mediums more. Also they Let them how to manage the resources more effectively so that every resident can avail them.

6) **Groups:** This graph can help the manager understand and see the various groups in CMS and their member details. This could help them better understand the community and cater to the needs of each and every residents.

Chapter 2: User

2.1 User for this Application

Residents	The main users of the application who live in the community and will be using the system to communicate, provide feedback ,reserve amenities etc.
Prospective Residents	People who can visit the system as future residents who can further become residents in near time with certain accessible features only.
Employees	People who are working within the community to provide the best of the services to the residents
Vendors	People who may or may not be living in the community to sell their products or give advertisements about their products.

2.2 Use cases for every user

1. Residents

- Create account after moving in
- Use CMS to manage lease and pay rent online
- Raise request for any maintenance and track status using CMS
- Book amenities using CMS for events
- Create carpooling for different purposes using CMS
- Create groups and events & Provide reviews
- Reserve parking spots for guest
- Tracking of packages
- Residents can Sell things using CMS

2. Prospective Residents

- Create account
- Check CMS for apartment availability & prices
- Book viewing dates
- Submit details for contact
- Contact resident representative upon request

3. Employee

- Post updates on CMS.
- View Rent details, available apartment details
- Send reminders regarding rent and other payments using CMS
- See prospective resident details
- Update user info, rent changes in the CMS
- Manage accounts
- Use CMS to view all pending request
- Update valid status of pending request in CMS

4. Vendors

- Check CMS for special services requests
- Create and update their company profile
- Edit their charges

- Post advertisements related to their products.

2.3 Queries for each type of User

1. Residents

- Resident can check for the status of their complaints.
- Residents can know about the various user in a group
- A residents can insert a carpool record as well add himself to an already existing carpool record.
- A resident can check and also insert a feedback from its own account.

2. Prospective Residents

- It can check for apartment availability and also look the rent for the same.
- It can create its own account as a prospective resident user and check for feedbacks.

3. Employee

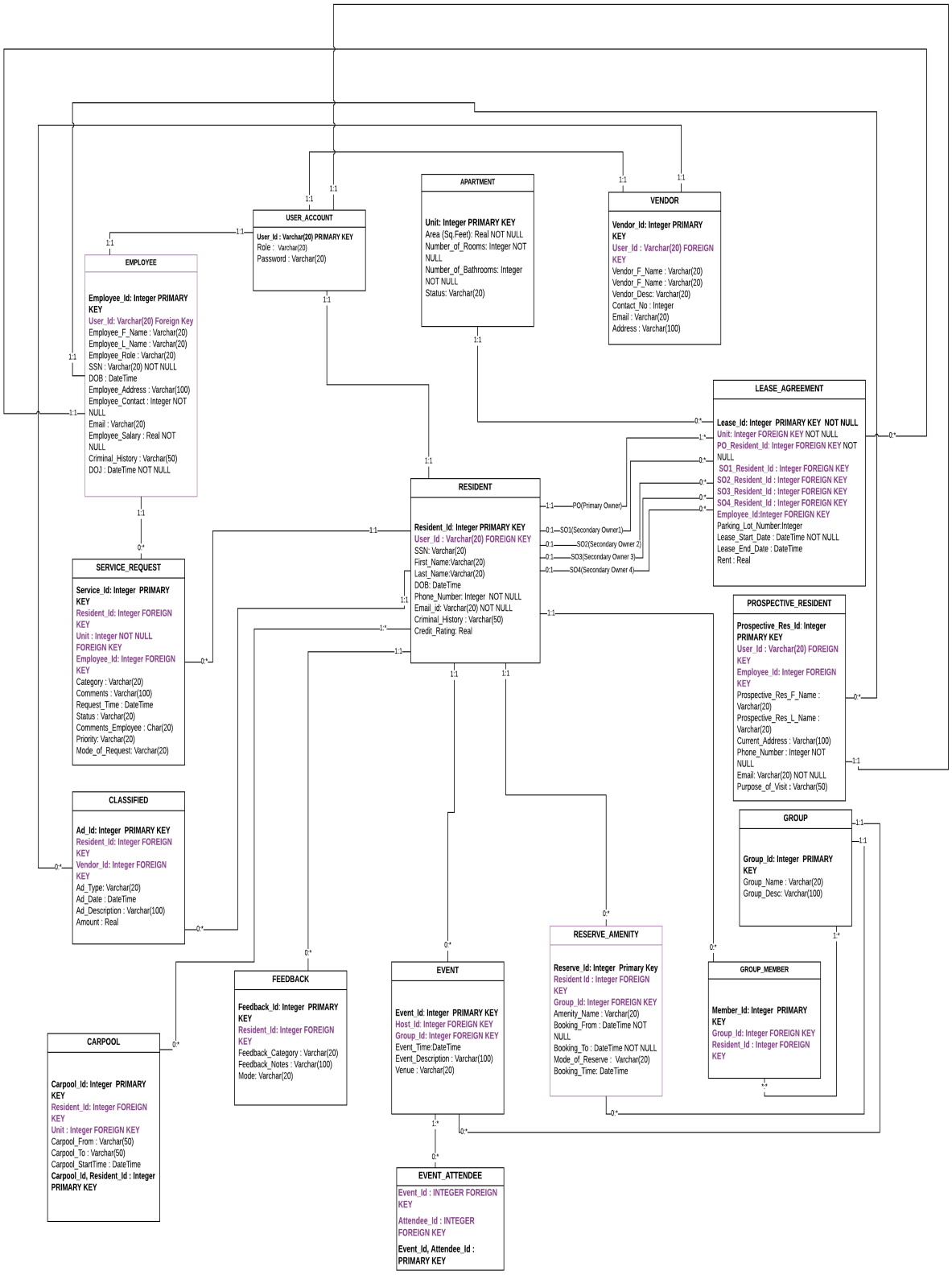
- Can check for a lease start date and end data for a particular apartment
- Can enter a new lease agreement for an apartment.
- Also can check and add various residents for a particular apartment.
- Can look for various incoming requests for maintenance.

4. Vendors

- Can edits information about its own company and products.
- Can insert, update or delete classified for various classifieds.

Chapter:3 Models

3.1 Logical Schema (UML model)



3.2 Physical Schema

Table: APARTMENT		
Column	Data type	Constraints
Unit	INTEGER	•PRIMARY KEY AUTOINCREMENT
Area	REAL	•NOT NULL
Number_of_Rooms	INTEGER	•NOT NULL •DEFAULT (1)
Number_of_Bathrooms	INTEGER	•NOT NULL •DEFAULT (1)
Status	VARCHAR	•DEFAULT Empty

Table: CARPOOL		
Column	Data type	Constraints
Carpool_Id	INTEGER	
Resident_Id	INTEGER	•REFERENCES RESIDENT (Resident_Id)
Unit	INTEGER	•REFERENCES APARTMENT (Unit)
Carpool_From	VARCHAR	
Carpool_To	VARCHAR	
Carpool_StartTime	DATETIME	
Global table constraints		
•PRIMARY KEY (Carpool_Id, Resident_Id)		

Table: CLASSIFIED		
Column	Data type	Constraints
Ad_Id	INTEGER	•PRIMARY KEY AUTOINCREMENT
Resident_Id	INTEGER	•REFERENCES RESIDENT (Resident_Id)
Vendor_Id	INTEGER	•REFERENCES VENDOR (Vendor_Id)
Ad_Type	VARCHAR	
Ad_date	DATETIME	
Ad_Description	VARCHAR	
Amount	REAL	

Table: EMPLOYEE		
Column	Data type	Constraints
Employee_Id	INTEGER	•PRIMARY KEY AUTOINCREMENT
User_Id	VARCHAR	•REFERENCES USER_ACCOUNT (User_Id)
Employee_F_Name	VARCHAR	
Employee_L_Name	VARCHAR	
Employee_Role	VARCHAR	
SSN	VARCHAR	•NOT NULL
DOB	DATETIME	
Employee_Address	VARCHAR	
Employee_Contact	INTEGER	•NOT NULL
Email	VARCHAR	
Employee_Salary	REAL	•NOT NULL
Criminal_History	VARCHAR	•DEFAULT Clean
DOJ	DATETIME	•NOT NULL

Table: EVENT_ATTENDEE		
Column	Data type	Constraints
Event_Id	INTEGER	•REFERENCES EVENT (Event_Id)
Attendee_Id	INTEGER	•REFERENCES RESIDENT (Resident_Id)
Global table constraints		
•PRIMARY KEY (Event_Id, Attendee_Id)		

Table: EVENT		
Column	Data type	Constraints
Event_Id	INTEGER	•PRIMARY KEY AUTOINCREMENT
Host_Id	INTEGER	•REFERENCES RESIDENT (Resident_Id)
Group_Id	INTEGER	•REFERENCES "GROUP" (Group_Id)
Event_Time	DATETIME	
Event_Description	VARCHAR	
Venue	VARCHAR	

Table: FEEDBACK		
Column	Data type	Constraints
Feedback_Id	INTEGER	•PRIMARY KEY AUTOINCREMENT
Resident_Id	INTEGER	•REFERENCES RESIDENT
Feedback_Category	VARCHAR	
Feedback_Notes	VARCHAR	
Mode	VARCHAR	

Table: GROUP_MEMBER		
Column	Data type	Constraints
Member_Id	INTEGER	•PRIMARY KEY AUTOINCREMENT
Group_Id	INTEGER	•REFERENCES "GROUP" (Group_Id)
Resident_Id	INTEGER	•REFERENCES RESIDENT

Table: GROUP		
Column	Data type	Constraints
Group_Id	INTEGER	•PRIMARY KEY AUTOINCREMENT
Group_Name	VARCHAR	
Group_Desc	VARCHAR	

Table: LEASE_AGREEMENT		
Column	Data type	Constraints
Lease_Id	INTEGER	•PRIMARY KEY AUTOINCREMENT •NOT NULL
Unit	INTEGER	•NOT NULL •REFERENCES APARTMENT (Unit)
PO_Resident_Id	INTEGER	•NOT NULL •REFERENCES RESIDENT (Resident_Id)
SO1_Resident_ID	INTEGER	•REFERENCES RESIDENT (Resident_Id)
SO2_Resident_ID	INTEGER	•REFERENCES RESIDENT (Resident_Id)
SO3_Resident_ID	INTEGER	•REFERENCES RESIDENT (Resident_Id)
SO4_Resident_ID	INTEGER	•REFERENCES RESIDENT (Resident_Id)
Employee_Id	INTEGER	•REFERENCES EMPLOYEE
Parking_Lot_Number	INTEGER	
Lease_Start_Date	DATETIME	•NOT NULL
Lease_End_Date	DATETIME	•DEFAULT NULL
Rent	REAL	

Table: PROSPECTIVE_RESIDENT		
Column	Data type	Constraints
Prospective_Res_Id	INTEGER	•PRIMARY KEY AUTOINCREMENT
User_Id	VARCHAR	•REFERENCES USER_ACCOUNT (User_Id)
Employee_Id	INTEGER	•REFERENCES EMPLOYEE
Prospective_Res_F_Name	VARCHAR	
Prospective_Res_L_Name	VARCHAR	
Current_Address	VARCHAR	
Phone_Number	INTEGER	•NOT NULL
Email	VARCHAR	•NOT NULL
Purpose_of_Visit	VARCHAR	

Table: RESERVE_AMENITY		
Column	Data type	Constraints
Reserve_Id	INTEGER	•PRIMARY KEY AUTOINCREMENT
Resident_Id	INTEGER	•REFERENCES RESIDENT
Group_Id	INTEGER	•REFERENCES "GROUP" (Group_Id)
Amenity_Name	VARCHAR	
Booking_From	DATETIME	•NOT NULL
Booking_To	DATETIME	•NOT NULL
Mode_of_Reserve	VARCHAR	
Booking_Time	DATETIME	

Table: RESIDENT		
Column	Data type	Constraints
Resident_Id	INTEGER	•PRIMARY KEY AUTOINCREMENT
User_Id	VARCHAR	•REFERENCES USER_ACCOUNT (User_Id)
SSN	VARCHAR	
First_Name	VARCHAR	
Last_Name	VARCHAR	
DOB	DATETIME	
Phone_Number	INTEGER	•NOT NULL
Email_Id	VARCHAR	•NOT NULL
Criminal_History	VARCHAR	
Credit_Rating	REAL	

Service_Id	INTEGER	•PRIMARY KEY AUTOINCREMENT
Resident_Id	INTEGER	•REFERENCES RESIDENT (Resident_Id)
Unit	INTEGER	•NOT NULL •REFERENCES APARTMENT (Unit)
Employee_Id	INTEGER	•REFERENCES EMPLOYEE
Category	VARCHAR	
Comments	VARCHAR	
Request_Time	DATETIME	
Status	VARCHAR	
Comments_Employee	CHAR	
Priority	VARCHAR	
Mode_of_Request	VARCHAR	

Table: USER_ACCOUNT		
Column	Data type	Constraints
User_Id	VARCHAR	•PRIMARY KEY ON CONFLICT FAIL •UNIQUE
Role	VARCHAR	
Password	VARCHAR	

Table: VENDOR		
Column	Data type	Constraints
Vendor_Id	INTEGER	•PRIMARY KEY
User_Id	VARCHAR	•REFERENCES USER_ACCOUNT (User_Id)
Vendor_F_Name	VARCHAR	
Vendor_L_Name	VARCHAR	
Vendor_Desc	VARCHAR	
Contact_No	INTEGER	
Email	VARCHAR	
Address	VARCHAR	

Chapter 4: Implementation

4.1 DDL Statements

Creation of Tables

1. CREATE TABLE APARTMENT (
   Unit INTEGER PRIMARY KEY AUTOINCREMENT,
   Area REAL NOT NULL,
   Number_of_Rooms INTEGER NOT NULL DEFAULT (1),
   Number_of_Bathrooms INTEGER NOT NULL DEFAULT (1),
   Status VARCHAR (20) DEFAULT Empty);

2. CREATE TABLE CARPOOL (
   Carpool_Id INTEGER,
   Resident_Id INTEGER REFERENCES RESIDENT (Resident_Id),
   Unit INTEGER REFERENCES APARTMENT (Unit),
   Carpool_From VARCHAR (50),
   Carpool_To VARCHAR (50),
   Carpool_StartTime DATETIME,
   PRIMARY KEY (Carpool_Id, Resident_Id));

3. CREATE TABLE CLASSIFIED (
   Ad_Id INTEGER PRIMARY KEY AUTOINCREMENT,
   Resident_Id INTEGER REFERENCES RESIDENT (Resident_Id),
   Vendor_Id INTEGER REFERENCES VENDOR (Vendor_Id),
   Ad_Type VARCHAR (20),
   Ad_date DATETIME,
   Ad_Description VARCHAR (100),
   Amount REAL);

4. CREATE TABLE EMPLOYEE (
   Employee_Id INTEGER PRIMARY KEY AUTOINCREMENT,
   User_Id VARCHAR (20) REFERENCES USER_ACCOUNT (User_Id),
   Employee_F_Name VARCHAR (20),
   Employee_L_Name VARCHAR (20),
   Employee_Role VARCHAR (20),
   SSN VARCHAR (20) NOT NULL,
   DOB DATETIME,
   Employee_Address VARCHAR (100),
   Employee_Contact INTEGER NOT NULL,
   Email VARCHAR (20),
   Employee_Salary REAL NOT NULL,
   Criminal_History VARCHAR (50) DEFAULT Clean,
   DOJ DATETIME NOT NULL);

5. CREATE TABLE EVENT (
   Event_Id INTEGER PRIMARY KEY AUTOINCREMENT,
   Host_Id INTEGER REFERENCES RESIDENT (Resident_Id),
   Group_Id INTEGER REFERENCES [GROUP] (Group_Id),
   Event_Time DATETIME,
   Event_Description VARCHAR (100),

Venue VARCHAR (20));

6. CREATE TABLE EVENT_ATTENDEE (
   Event_Id INTEGER REFERENCES EVENT (Event_Id),
   Attendee_Id INTEGER REFERENCES RESIDENT (Resident_Id),
   PRIMARY KEY (Event_Id, Attendee_Id));

7. CREATE TABLE FEEDBACK (
   Feedback_Id INTEGER PRIMARY KEY AUTOINCREMENT,
   Resident_Id INTEGER REFERENCES RESIDENT,
   Feedback_Category VARCHAR (20),
   Feedback_Notes VARCHAR (100),
   Mode VARCHAR (20));

8. CREATE TABLE [GROUP] (
   Group_Id INTEGER PRIMARY KEY AUTOINCREMENT,
   Group_Name VARCHAR (20),
   Group_Desc VARCHAR (100));

9. CREATE TABLE GROUP_MEMBER (
   Member_Id INTEGER PRIMARY KEY AUTOINCREMENT,
   Group_Id INTEGER REFERENCES [GROUP] (Group_Id),
   Resident_Id INTEGER REFERENCES RESIDENT);

10. CREATE TABLE LEASE_AGREEMENT (
    Lease_Id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
    Unit INTEGER NOT NULL REFERENCES APARTMENT (Unit),
    PO_Resident_Id INTEGER NOT NULL
    REFERENCES RESIDENT (Resident_Id),
    SO1_Resident_ID INTEGER REFERENCES RESIDENT (Resident_Id),
    SO2_Resident_ID INTEGER REFERENCES RESIDENT (Resident_Id),
    SO3_Resident_ID INTEGER REFERENCES RESIDENT (Resident_Id),
    SO4_Resident_ID INTEGER REFERENCES RESIDENT (Resident_Id),
    Employee_Id INTEGER REFERENCES EMPLOYEE,
    Parking_Lot_Number INTEGER,
    Lease_Start_Date DATETIME NOT NULL,
    Lease_End_Date DATETIME DEFAULT NULL,
    Rent REAL);

11. CREATE TABLE PROSPECTIVE_RESIDENT (
    Prospective_Res_Id INTEGER PRIMARY KEY AUTOINCREMENT,
    User_Id VARCHAR (20) REFERENCES USER_ACCOUNT (User_Id),
    Employee_Id INTEGER REFERENCES EMPLOYEE,
    Prospective_Res_F_Name VARCHAR (20),
    Prospective_Res_L_Name VARCHAR (20),
    Current_Address VARCHAR (100),
    Phone_Number INTEGER NOT NULL,
    Email VARCHAR (20) NOT NULL,
    Purpose_of_Visit VARCHAR (50));

12. CREATE TABLE RESERVE_AMENITY (
    Reserve_Id INTEGER PRIMARY KEY AUTOINCREMENT,

```
    Resident_Id INTEGER REFERENCES RESIDENT,
    Group_Id INTEGER REFERENCES [GROUP] (Group_Id),
    Amenity_Name VARCHAR (20),
    Booking_From DATETIME NOT NULL,
    Booking_To DATETIME NOT NULL,
    Mode_of_Reserve VARCHAR (20),
    Booking_Time DATETIME);
```

13.
```
CREATE TABLE RESIDENT (
    Resident_Id INTEGER PRIMARY KEY AUTOINCREMENT,
    User_Id VARCHAR (20) REFERENCES USER_ACCOUNT (User_Id),
    SSN VARCHAR (20),
    First_Name VARCHAR (20),
    Last_Name VARCHAR (20),
    DOB DATETIME,
    Phone_Number INTEGER NOT NULL,
    Email_Id VARCHAR (20) NOT NULL,
    Criminal_History VARCHAR (50),
    Credit_Rating REAL);
```

14.
```
CREATE TABLE SERVICE_REQUEST (
    Service_Id INTEGER PRIMARY KEY AUTOINCREMENT,
    Resident_Id INTEGER REFERENCES RESIDENT (Resident_Id),
    Unit INTEGER NOT NULL
    REFERENCES APARTMENT (Unit),
    Employee_Id INTEGER REFERENCES EMPLOYEE,
    Category VARCHAR (20),
    Comments VARCHAR (100),
    Request_Time DATETIME,
    Status VARCHAR (20),
    Comments_Employee CHAR (20),
    Priority VARCHAR (20),
    Mode_of_Request VARCHAR (20));
```

15.
```
CREATE TABLE USER_ACCOUNT (
    User_Id VARCHAR (20) PRIMARY KEY ON CONFLICT FAIL
    UNIQUE,
    Role VARCHAR (20),
    Password VARCHAR (20));
```

16.
```
CREATE TABLE VENDOR (
    Vendor_Id INTEGER PRIMARY KEY,
    User_Id VARCHAR (20) REFERENCES USER_ACCOUNT (User_Id),
    Vendor_F_Name VARCHAR (20),
    Vendor_L_Name VARCHAR (20),
    Vendor_Desc VARCHAR (20),
    Contact_No INTEGER,
    Email VARCHAR (20),
    Address VARCHAR (100));
```

4.2 DML Statements

Select all statements

1. Select * from APARTMENT

	Unit	Area	Number_of_Rooms	Number_of_Bathrooms	Status
1	1105	2000	2	2	Rented
2	1205	3000	3	2	Rented
3	2105	3000	3	3	Rented
4	2205	2000	2	2	Rented
5	3105	3000	3	3	Available
6	3205	3000	2	2	Available
7	4105	3000	3	3	Rented
8	5105	2000	2	2	Available
9	6105	2000	2	2	Rented
10	7105	2500	2	2	Rented
11	8105	2500	2	2	Rented
12	9105	2000	2	2	Rented

2. Select * from LEASE_AGREMENT

Lease_Id	Unit	PO_Resident_Id	SO1_Resident_ID	SO2_Resident_ID	SO3_Resident_ID	SO4_Resident_ID	Employee_Id	Parking_Lot_Number
1	1105	101	106	107	108	NULL	121001	10
2	2105	102	109	110	NULL	NULL	121001	20
3	3105	103	NULL	NULL	NULL	NULL	121001	30
4	4105	104	111	112	113	NULL	121001	40
5	5105	105	114	NULL	NULL	NULL	121001	50
6	1105	115	116	117	NULL	NULL	121001	10
7	3105	118	119	120	121	122	121001	30
8	4105	123	124	NULL	NULL	NULL	121001	40
9	6105	113	110	109	NULL	NULL	121001	60
10	7105	106	NULL	NULL	NULL	NULL	121001	70
11	8105	107	108	NULL	NULL	NULL	121001	80
12	9105	110	112	117	NULL	NULL	121001	90
13	1205	115	116	118	119	NULL	121001	100
14	2205	120	121	122	123	NULL	121001	110
15	3205	124	NULL	NULL	NULL	NULL	121001	120

Lease_Start_Date	Lease_End_Date	Rent
01/02/2015	01/01/2016	3000
02/02/2016	02/01/2019	2850
05/05/2014	05/04/2016	2900
08/08/2015	08/07/2016	3100
10/11/2014	10/10/2016	3000
03/01/2016	08/28/2017	2800
05/06/2016	05/05/2017	3000
08/08/2016	08/07/2017	3150
01/05/2017	12/31/2017	2700
02/06/2017	02/05/2018	2800
03/07/2017	03/08/2018	2900
04/08/2017	04/07/2018	2950
04/20/2017	04/19/2018	3000
05/21/2017	05/20/2018	3050
06/20/2017	06/19/2018	2500

3. Select * from RESIDENT

Resident_Id	User_Id	SSN	First_Name	Last_Name	DOB	Phone_Number	Email_Id	Criminal_History	Credit_Rating
101	nsingh	123-456-7899	Nishant	Singh	12/21/1993	6698542365	nsingh@gmail.com		900
102	adoshi	234-567-8910	Aalap	Doshi	07/13/1993	5634256635	adoshi@gmail.com		855
103	jshaligram	125-456-852	Jui	Shaligram	06/25/1990	8546325452	jshaligram@gmail.com		789
104	tmishra	652-563-7845	Tulika	Mishra	02/29/1992	7865231459	tmishra@gmail.com		689
105	bhargav1	546-851-4562	Bhargav	Ember	11/20/1990	4785693215	ebhargav@gmail.com	Parking tickets	840
106	agasatheya	785-486-9982	Agasthiya	Kundarthy	05/14/1995	7854589632	agastheya@gmail.com		560
107	nikhil	785-985-7895	Nikhil	Singh	05/23/1991	7854568965	nikhil@yahoo.com		755
108	ngoushal	652-546-8456	Neha	Goushal	12/30/1996	7854214565	ngoushal@gmail.com		788
109	pnirpase	986-326-4521	Priyanka	Nirpase	04/14/1987	7541236549	pnirpase@scu.edu		800
110	asultan	865-965-7458	Adeel	Sultan	12/27/1989	7854123652	asultan@hotmail.com		850
111	parvathi	264-456-1234	Parvathi	Varma	12/12/1982	4521362546	parvathi@rediffmail.com		950
112	ramesh	452-123-4568	Ramesh	Babu	02/31/1985	1235654578	rbabu@yahoo.co.in		562
113	bob	456-312-4568	Bob	Cartner	07/15/1955	7845122356	bcarpenter@gmail.com		852
114	mark	965-321-4569	Mark	Benson	08/08/1965	5689784512	mbenson@gmail.com		641
115	john	785-123-4569	John	Terry	09/09/1982	2345895678	jterry@yahoo.com		983
116	mary	561-234-5678	Mary	Jane	10/10/1986	1597538426	mjane@hotmail.com		756
117	eddie	294-521-2365	Eddie	Gurrero	11/11/1975	2684753951	egurrero@yahoo.com		842
118	ghangsu	895-623-1425	Ghanghsu	Mia	08/10/1999	4523567845	gmia@gmail.com		952
119	ashely	963-145-2356	Ashely	Cole	05/14/1986	2558744136	acole@gmail.com		654
120	michal	789-642-3146	Michal	Ballack	01/25/1976	6998877441	mballack@yahoo.com		942
121	frank	987-889-7898	Frank	Lampard	12/17/1989	1226688412	flampard@gmail.com		753
122	didier	456-545-6545	Didier	Drogba	02/23/1995	5654983245	ddrogba@scu.edu		951
123	ronaldo	123-456-2125	Ronaldo	Cole	02/23/1972	1230321566	rcole@yahoo.com		864
124	messi	753-951-4268	Messi	Siemen	05/14/1987	8532158967	msiemen@gmail.com		759

4. Select * from EMPLOYEE

Employee_Id	User_Id	Employee_F_Name	Employee_L_Name	Employee_Role	SSN	DOB	Employee_Address
121001	harperjack	Jack	Harper	Manager	168-456-7899	12/18/1970	Apt#123,New Hall,SC-95050
121002	whitecharli	Charli	White	Gardener	456-123-7896	03/16/1985	789, Fox Avenue, San Jose-95054
121003	cruzted	Ted	Cruz	Electrician	789-074-8867	04/12/1990	759, Fox Avenue, San Jose-95054
121004	hoffmandustin	Dustin	Hoffman	Carpenter	541-231-9999	07/31/1992	Apt#245, Redwood, San Jose-95059
121006	andrewsjose	Andrews	Jose	Plumber	876-354-7182	02/18/1988	Apt#671, Ginger Street, San Jose-9509

Employee_Contact	Email	Employee_Salary	Criminal_History	DOJ
6693048967	harper78@gmail.com	$70000	No	01/01/2000
6693048968	charliwhite@yahoo.com	$30000	No	04/31/2005
6693048969	cruzted@yahoo.com	$40000	No	07/12/2006
6693047099	hoffmandustin@aolmail.com	$45000	No	07/01/2006
6693048972	joseandrews@gmail.com	$40000	No	11/23/2010

5. Select * from PROSPECTIVE_RESIDENT

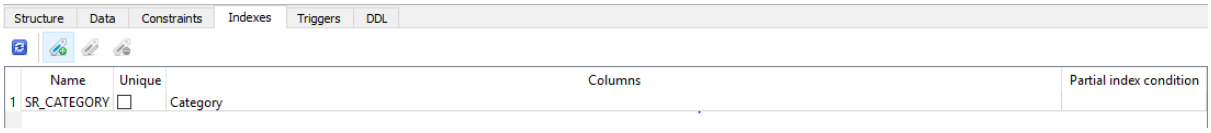
Total rows loaded: 6

	Prospective_Res_Id	User_Id	Employee_Id	Prospective_Res_F_Name	Prospective_Res_L_Name	Current_Address	Phone_Number	Email	
1	21013	hankstom	121001	Tom	Hanks	Apt#345, Mcclen St, Freemont-95060	4044873241	tomhanks11@gmail.com	A
2	21014	jenkinssherry	121001	Sherry	Jenkins	124, Flora Vista Avenue, San Jose-961170	5641873902	sherryjenkin78@yahoo.com	A
3	21015	frank1	121001	Frank	Lampard	apt# 25 Seemon ST, SJ-95050	1226688412	flampard@gmail.com	A
4	21016	didi1	121001	Didier	Drogba	apt#45 Lehman St, freemont-95050	5654983246	ddrogba@scu.edu	A
5	21017	ronaldo123	121001	Ronaldo	Cole	apt#56 Deran St, SJ-95045	1230321566	rcole@yahoo.com	A
6	21018	messi123	121001	Messi	Siemen	34# lowell st, sunnyvale-75684	8532158967	msiemen@gmail.com	A

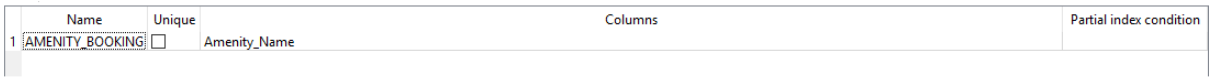
Phone_Number	Email	Purpose_of_Visit
4044873241	tomhanks11@gmail.com	Apartment Viewing
5641873902	sherryjenkin78@yahoo.com	Apartment Viewing
1226688412	flampard@gmail.com	Apartment Viewing
5654983246	ddrogba@scu.edu	Apartment Viewing
1230321566	rcole@yahoo.com	Apartment Viewing
8532158967	msiemen@gmail.com	Apartment Viewing

INDEXES

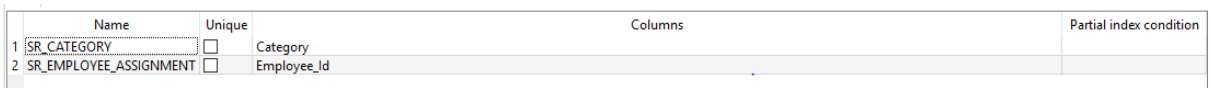
1) CREATE INDEX SR_CATEGORY ON SERVICE_REQUEST(Category);



2) CREATE INDEX AMENITY_BOOKING ON RESERVE_AMENITY(Amenity_Name);

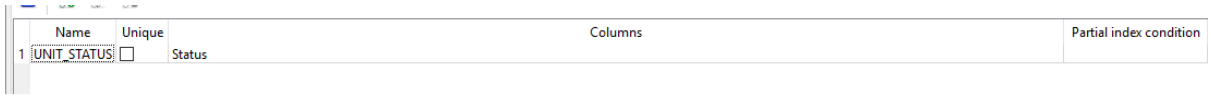


3) CREATE INDEX SR_EMPLOYEE_ASSIGNMENT ON SERVICE_REQUEST(Employee_Id);



4) CREATE INDEX UNIT_LEASE_DETAILS ON LEASE_AGREEMENT(Unit);

5) CREATE INDEX UNIT_STATUS ON APARTMENT(Status);



Views

1) **APARTMENTS_AVAILABLE**

```
CREATE VIEW APARTMENTS_AVAILABLE AS
SELECT Number_of_Rooms AS [Number of Bedrooms],
Number_of_Bathrooms AS [Number of Bathrooms],
Area,
COUNT(Number_of_Rooms) AS [Available Units]
FROM APARTMENT
WHERE STATUS = 'Available'
GROUP BY Number_of_Rooms,
Number_of_Bathrooms,
        Area;
```

This is a function for Prospective Residents. They will be able to view the following.

```
SELECT * FROM APARTMENTS_AVAILABLE;
```

Grid view | Form view

Total rows loaded: 3

	Number of Bedrooms	Number of Bathrooms	Area	Available Units
1	2	2	2000	1
2	2	2	3000	1
3	3	3	3000	1

2) BOOKED_AMENITIES

```
CREATE VIEW BOOKED_AMENITIES AS
SELECT Amenity_Name AS [Amenity Name],
Booking_From AS [Booked From],
Booking_To AS [Booked To]
FROM RESERVE_AMENITY;

SELECT * FROM BOOKED_AMENITIES;
```

Grid view | Form view

Total rows loaded: 10

	Amenity Name	Booked From	Booked To	ResCol_0
1	Community Hall	06/25/2017 10:00:00	06/25/2017 12:00:00	100001
2	Pool	07/04/2017 20:00:00	07/04/ 2017 20:00:00	100002
3	Community Hall	07/15/2017 18:00:00	07/15/2017 20:00:00	100003
4	Pool	06/25/2017 09:00:00	06/25/2017 12:00:00	100004
5	Pool	06/26/2017 13:00:00	06/26/2017 17:00:00	100005
6	Community Hall	06/26/2017 14:00:00	06/26/2017 16:00:00	100006
7	Pool	06/27/2017 10:00:00	06/27/2017 14:00:00	100007
8	Community Hall	06/28/2017 17:00:00	06/28/2017 20:00:00	100008
9	Pool	06/29/2017 13:00:00	06/29/2017 19:00:00	100009
10	Community Hall	06/30/2017 15:00:00	06/30/2017 19:00:00	1000010

Insert

1. INSERT INTO APARTMENT (Unit, Area, Number_of_Rooms, Number_of_Bathrooms, Status) VALUES (1105, 2000, 2, 2, 'Rented');
2. INSERT INTO CARPOOL (Carpool_Id, Resident_Id, Unit, Carpool_From, Carpool_To, Carpool_StartTime) VALUES (710001, 118, 1105, '1190 Benton St.', 'mountain view', '07-02-2017 18:00:00');
3. INSERT INTO EMPLOYEE (Employee_Id, User_Id, Employee_F_Name, Employee_L_Name, Employee_Role, SSN, DOB, Employee_Address, Employee_Contact, Email, Employee_Salary, Criminal_History, DOJ) VALUES (121001, 'harperjack', 'Jack', 'Harper', 'Manager', '168-456-7899', '12/18/1970', 'Apt#123,New Hall,SC-95050', 6693048967, 'harper78@gmail.com', '$70000', 'No', '01/01/2000');

4. INSERT INTO LEASE_AGREEMENT (Lease_Id, Unit, PO_Resident_Id, SO1_Resident_ID, SO2_Resident_ID, SO3_Resident_ID, SO4_Resident_ID, Employee_Id, Parking_Lot_Number, Lease_Start_Date, Lease_End_Date, Rent) VALUES (1, 1105, 101, 106, 107, 108, NULL, 121001, 10, '01/02/2015', '01/01/2016', 3000);
5. INSERT INTO PROSPECTIVE_RESIDENT (Prospective_Res_Id, User_Id, Employee_Id, Prospective_Res_F_Name, Prospective_Res_L_Name, Current_Address, Phone_Number, Email, Purpose_of_Visit) VALUES (21013, 'hankstom', 121001, 'Tom ', 'Hanks ', 'Apt#345, Mcclen St, Freemont-95060', 4044873241, 'tomhanks11@gmail.com', 'Apartment Viewing');
6. INSERT INTO RESIDENT (Resident_Id, User_Id, SSN, First_Name, Last_Name, DOB, Phone_Number, Email_Id, Criminal_History, Credit_Rating) VALUES (101, 'nsingh', '123-456-7899', 'Nishant', 'Singh', '12/21/1993', 6698542365, 'nsingh@gmail.com', '', 900);

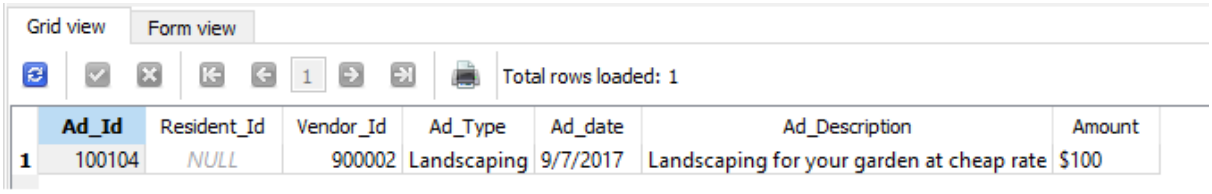
Special Queries

VENDOR QUERIES

1. Put a Classified on CMS

   INSERT Query:
   INSERT INTO CLASSIFIED (Ad_Id, Resident_Id, Vendor_Id, Ad_Type, Ad_date, Ad_Description, Amount) VALUES (100104, NULL, 900002, 'Landscaping', '9/7/2017', 'Landscaping for your garden at cheap rate', '$100');

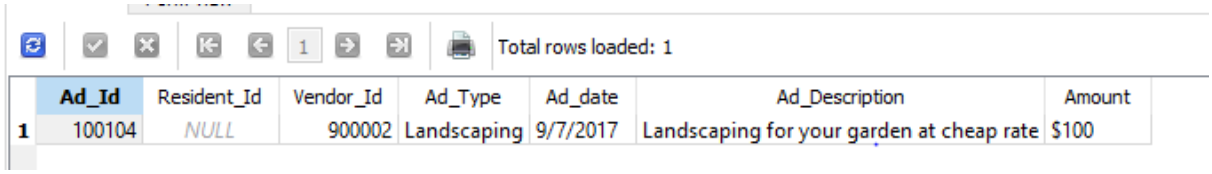
   SELECT * FROM CLASSIFIED WHERE Vendor_Id = 900002;



	Ad_Id	Resident_Id	Vendor_Id	Ad_Type	Ad_date	Ad_Description	Amount
1	100104	NULL	900002	Landscaping	9/7/2017	Landscaping for your garden at cheap rate	$100

2. Update Classified Details on CMS

   Ad Details before updating



	Ad_Id	Resident_Id	Vendor_Id	Ad_Type	Ad_date	Ad_Description	Amount
1	100104	NULL	900002	Landscaping	9/7/2017	Landscaping for your garden at cheap rate	$100

   UPDATE Query:

   UPDATE CLASSIFIED
   SET Ad_Description = 'Check the most attractive landscaping solution',
   Amount = '$70'
   WHERE Ad_Id = 100104;

Ad Details after Updating

SELECT * FROM CLASSIFIED WHERE Vendor_Id = 900002;

	Ad_Id	Resident_Id	Vendor_Id	Ad_Type	Ad_date	Ad_Description	Amount
1	100104	NULL	900002	Landscaping	9/7/2017	Check the most attractive landscaping solution	$70

Another vendor updates his classified details

SELECT * FROM CLASSIFIED WHERE Ad_Id = 100103;

Total rows loaded: 1

	Ad_Id	Resident_Id	Vendor_Id	Ad_Type	Ad_date	Ad_Description	Amount
1	100103	NULL	900001	TV	5/4/2017	50% off SALE your local Firniture Shope	$500-$5000

Update

```
UPDATE CLASSIFIED
SET
Ad_Description = '80% off SALE @ your local Furniture Shop '
WHERE Ad_Id = 100103;
```

Ad details after updating

Total rows loaded: 1

	Ad_Id	Resident_Id	Vendor_Id	Ad_Type	Ad_date	Ad_Description	Amount
1	100103	NULL	900001	TV	5/4/2017	80% off SALE @ your local Furniture Shop	$500-$5000

3. Delete a classified from CMS (Assumption: A vendor can delete only classifieds raised by himself)

Ad Details before Deleting

Ads Raised By Vendor

SELECT * FROM CLASSIFIED WHERE Vendor_Id = 900002;

	Ad_Id	Resident_Id	Vendor_Id	Ad_Type	Ad_date	Ad_Description	Amount
1	100104	NULL	900002	Landscaping	9/7/2017	Check the most attractive landscaping solution	$70

All Ads in CLASSIFIED table

SELECT * FROM CLASSIFIED;

Total rows loaded: 4

	Ad_Id	Resident_Id	Vendor_Id	Ad_Type	Ad_date	Ad_Description	Amount
1	100101	107	NULL	Sofa	6/7/2017	A year old sofa for grabs	$99
2	100102	121	NULL	PS4	5/7/2017	2 Year old PS4	$100
3	100103	NULL	900001	TV	5/4/2017	50% off SALE your local Firniture Shope	$500-$5000
4	100104	NULL	900002	Landscaping	9/7/2017	Check the most attractive landscaping solution	$70

Vendor 900002 deletes his Ad

```
DELETE FROM CLASSIFIED
WHERE Ad_Id = '100104';
```

Ad Details after Deleting

Ads Raised By Vendor

```
SELECT * FROM CLASSIFIED WHERE Vendor_Id = 900002;
```

Total rows loaded: 0

Ad_Id	Resident_Id	Vendor_Id	Ad_Type	Ad_date	Ad_Description	Amount

```
SELECT * FROM CLASSIFIED;
```

Total rows loaded: 3

	Ad_Id	Resident_Id	Vendor_Id	Ad_Type	Ad_date	Ad_Description	Amount
1	100101	107	NULL	Sofa	6/7/2017	A year old sofa for grabs	$99
2	100102	121	NULL	PS4	5/7/2017	2 Year old PS4	$100
3	100103	NULL	900001	TV	5/4/2017	50% off SALE your local Firniture Shope	$500-$5000

4. Vendor can view all the ad details posted on CMS

```
SELECT * FROM CLASSIFIED;
```

Total rows loaded: 3

	Ad_Id	Resident_Id	Vendor_Id	Ad_Type	Ad_date	Ad_Description	Amount
1	100101	107	NULL	Sofa	6/7/2017	A year old sofa for grabs	$99
2	100102	121	NULL	PS4	5/7/2017	2 Year old PS4	$100
3	100103	NULL	900001	TV	5/4/2017	50% off SALE your local Firniture Shope	$500-$5000

5. A vendor can change his details once he logs into CMS

```
SELECT * FROM VENDOR WHERE User_Id = 'raviembar';
```

	Vendor_Id	User_Id	Vendor_F_Name	Vendor_L_Name	Vendor_Desc	Contact_No	Email	Address
1	900001	raviembar	Bhargav	Embar	Pest Control	4785693215	ebhargav@pestcontrol.com	Apt#5105, 1090 Benton St., SC-95050

This vendor wants to update description of what he does.

```
UPDATE VENDOR
SET Vendor_Desc = 'Get rid of pests'
WHERE User_Id = 'raviembar';
```

Vendor details after updating

Total rows loaded: 1

	Vendor_Id	User_Id	Vendor_F_Name	Vendor_L_Name	Vendor_Desc	Contact_No	Email	Address
1	900001	raviembar	Bhargav	Embar	Get rid of pests	4785693215	ebhargav@pestcontrol.com	Apt#5105, 1090 Benton St., SC-95050

EMPLOYEE QUERIES

1. An employee can view all the service requests assigned to him

```
SELECT * FROM SERVICE_REQUEST WHERE Employee_Id = 121003;
```

Total rows loaded: 3

	Service_Id	Resident_Id	Unit	Employee_Id	Category	Comments	Request_Time	Status	Comments_Employee	Priority	Mode_of_Request
1	650001	118	1105	121003	Electrical	Bedroom Switch light not working	1/7/2017	IN-PROGRESS	Employee Dispatched to fix the issue	MEDIUM	CMS
2	650004	117	1105	121003	Electrical	Kitchen power socket not working	07/08/17	ASSIGNED	Employee Dispatched to fix the issue	MEDIUM	CMS
3	650008	106	1105	121003	Electrical	NO Power in house	07/01/2017	RESOLVED	Resolved Same Day	URGENT	CMS

2. An employee can update Service Request Comments and Status assigned to him.

Service Request Details before Updating

Total rows loaded: 1

	Service_Id	Resident_Id	Unit	Employee_Id	Category	Comments	Request_Time	Status	Comments_Employee	Priority	Mode_of_Request
1	650001	118	1105	121003	Electrical	Bedroom Switch light not working	1/7/2017	IN-PROGRESS	Employee Dispatched to fix the issue	MEDIUM	CMS

Update

```
UPDATE SERVICE_REQUEST
SET Status = 'RESOLVED', Comments_Employee = 'Lights fixed'
WHERE Service_Id = 650001;
```

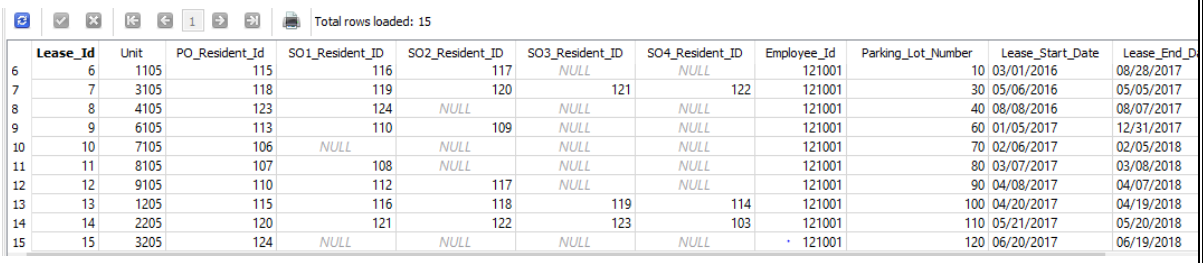
Service Request Details after Updating

Total rows loaded: 1

	Service_Id	Resident_Id	Unit	Employee_Id	Category	Comments	Request_Time	Status	Comments_Employee	Priority	Mode_of_Request
1	650001	118	1105	121003	Electrical	Bedroom Switch light not working	1/7/2017	RESOLVED	Lights fixed	MEDIUM	CMS

3. An employee (manager) can view all lease agreement details

SELECT * FROM LEASE_AGREEMENT;



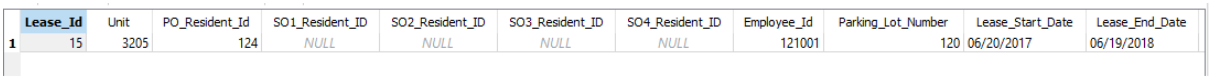
Total rows loaded: 15

	Lease_Id	Unit	PO_Resident_Id	SO1_Resident_ID	SO2_Resident_ID	SO3_Resident_ID	SO4_Resident_ID	Employee_Id	Parking_Lot_Number	Lease_Start_Date	Lease_End_D
6	6	1105	115	116	117	NULL	NULL	121001	10	03/01/2016	08/28/2017
7	7	3105	118	119	120	121	122	121001	30	05/06/2016	05/05/2017
8	8	4105	123	124	NULL	NULL	NULL	121001	40	08/08/2016	08/07/2017
9	9	6105	113	110	109	NULL	NULL	121001	60	01/05/2017	12/31/2017
10	10	7105	106	NULL	NULL	NULL	NULL	121001	70	02/06/2017	02/05/2018
11	11	8105	107	108	NULL	NULL	NULL	121001	80	03/07/2017	03/08/2018
12	12	9105	110	112	117	NULL	NULL	121001	90	04/08/2017	04/07/2018
13	13	1205	115	116	118	119	114	121001	100	04/20/2017	04/19/2018
14	14	2205	120	121	122	123	103	121001	110	05/21/2017	05/20/2018
15	15	3205	124	NULL	NULL	NULL	NULL	121001	120	06/20/2017	06/19/2018

4. An employee can update details in a lease agreement

Lease agreement details before updating

SELECT * FROM LEASE_AGREEMENT WHERE Lease_Id = 15;

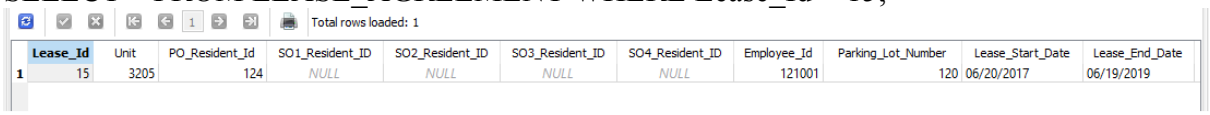


	Lease_Id	Unit	PO_Resident_Id	SO1_Resident_ID	SO2_Resident_ID	SO3_Resident_ID	SO4_Resident_ID	Employee_Id	Parking_Lot_Number	Lease_Start_Date	Lease_End_Date
1	15	3205	124	NULL	NULL	NULL	NULL	121001	120	06/20/2017	06/19/2018

UPDATE LEASE_AGREEMENT
SET Lease_End_Date = 06/19/2019
WHERE Lease_Id = 15;

Lease agreement details after updating

SELECT * FROM LEASE_AGREEMENT WHERE Lease_Id = 15;



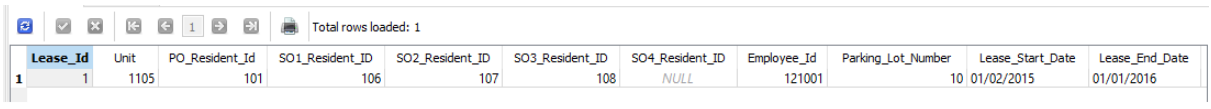
Total rows loaded: 1

	Lease_Id	Unit	PO_Resident_Id	SO1_Resident_ID	SO2_Resident_ID	SO3_Resident_ID	SO4_Resident_ID	Employee_Id	Parking_Lot_Number	Lease_Start_Date	Lease_End_Date
1	15	3205	124	NULL	NULL	NULL	NULL	121001	120	06/20/2017	06/19/2019

5. An employee can insert details in lease agreement

INSERT INTO LEASE_AGREEMENT (Lease_Id, Unit, PO_Resident_Id, SO1_Resident_ID, SO2_Resident_ID, SO3_Resident_ID, SO4_Resident_ID, Employee_Id, Parking_Lot_Number, Lease_Start_Date, Lease_End_Date, Rent)
VALUES (1, 1105, 101, 106, 107, 108, NULL, 121001, 10, '01/02/2015', '01/01/2016', 3000);

SELECT * FROM LEASE_AGREEMENT WHERE Lease_Id = 1;



Total rows loaded: 1

	Lease_Id	Unit	PO_Resident_Id	SO1_Resident_ID	SO2_Resident_ID	SO3_Resident_ID	SO4_Resident_ID	Employee_Id	Parking_Lot_Number	Lease_Start_Date	Lease_End_Date
1	1	1105	101	106	107	108	NULL	121001	10	01/02/2015	01/01/2016

6. An employee can view Prospective Resident details assigned to him for management so that he can contact them.

SELECT * FROM PROSPECTIVE_RESIDENT WHERE Employee_Id = 121001;

	Prospective_Res_Id	User_Id	Employee_Id	Prospective_Res_F_Name	Prospective_Res_L_Name	Current_Address	Phone_Number	Email	
1	21013	hankstom	121001	Tom	Hanks	Apt#345, Mcclen St, Freemont-95060	4044873241	tomhanks11@gmail.com	A
2	21014	jenkinssherry	121001	Sherry	Jenkins	124, Flora Vista Avenue, San Jose-961170	5641873902	sherryjenkin78@yahoo.com	A
3	21015	frank1	121001	Frank	Lampard	apt# 25 Seemon ST, SJ-95050	1226688412	flampard@gmail.com	A
4	21016	didi1	121001	Didier	Drogba	apt#45 Lehman St, freemont-95050	5654983246	ddrogba@scu.edu	A
5	21017	ronaldo123	121001	Ronaldo	Cole	apt#56 Deran St, SJ-95045	1230321566	rcole@yahoo.com	A
6	21018	messi123	121001	Messi	Siemen	34# lowell st, sunnyvale-75684	8532158967	msiemen@gmail.com	A

Prospective Resident Queries

1. A prospective resident can update his details on CMS.

   Details before updating

   SELECT * FROM PROSPECTIVE_RESIDENT WHERE User_Id = 'hankstom';

Total rows loaded: 1

	Prospective_Res_Id	User_Id	Employee_Id	Prospective_Res_F_Name	Prospective_Res_L_Name	Current_Address	Phone_Number	Email	Purpose
1	21013	hankstom	121001	Tom	Hanks	Apt#345, Mcclen St, Freemont-95060	4044873241	tomhanks11@gmail.com	Apartmer

   Update

   UPDATE PROSPECTIVE_RESIDENT
   SET Current_Address = 'Apt#710 McCllen St, Fremont - 95060'
   WHERE User_Id = 'hankstom'

   Details after updating
   SELECT * FROM PROSPECTIVE_RESIDENT WHERE User_Id = 'hankstom';

Total rows loaded: 1

	Prospective_Res_Id	User_Id	Employee_Id	Prospective_Res_F_Name	Prospective_Res_L_Name	Current_Address	Phone_Number	Email	Purpose
1	21013	hankstom	121001	Tom	Hanks	Apt#710 McCllen St, Fremont - 95060	4044873241	tomhanks11@gmail.com	Apartme

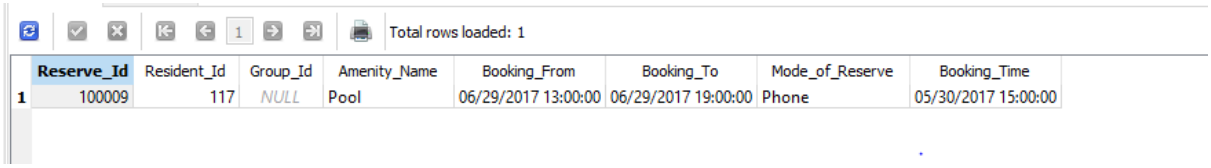
RESIDENTS QUERIES

There are many functionalities for a resident. Out of them, few queries are given.

1. A resident can update details of amenities he has reserved.

   Details before Updating

   SELECT * FROM RESERVE_AMENITY WHERE Reserve_Id = 100009;



Total rows loaded: 1

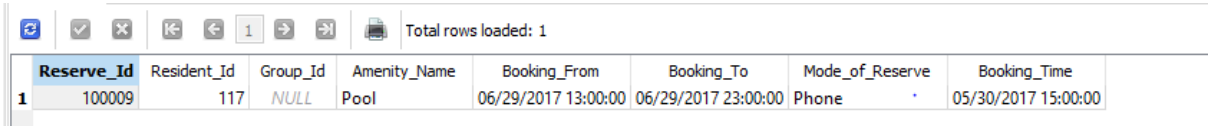
	Reserve_Id	Resident_Id	Group_Id	Amenity_Name	Booking_From	Booking_To	Mode_of_Reserve	Booking_Time
1	100009	117	NULL	Pool	06/29/2017 13:00:00	06/29/2017 19:00:00	Phone	05/30/2017 15:00:00

Update Details

```
UPDATE RESERVE_AMENITY
SET Booking_To = '06/29/2017 23:00:00'
WHERE Reserve_Id=100009;
```

Details after updating

```
SELECT * FROM RESERVE_AMENITY WHERE Reserve_Id = 100009;
```



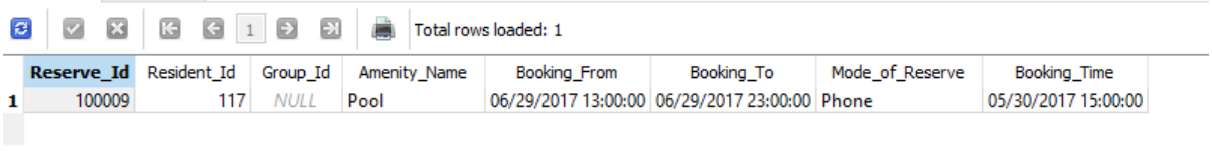
Total rows loaded: 1

	Reserve_Id	Resident_Id	Group_Id	Amenity_Name	Booking_From	Booking_To	Mode_of_Reserve	Booking_Time
1	100009	117	NULL	Pool	06/29/2017 13:00:00	06/29/2017 23:00:00	Phone	05/30/2017 15:00:00

2. A resident can delete the amenities he has reserved.

Details before deleting

```
SELECT * FROM RESERVE_AMENITY WHERE Resident_Id = 117;
```



Total rows loaded: 1

	Reserve_Id	Resident_Id	Group_Id	Amenity_Name	Booking_From	Booking_To	Mode_of_Reserve	Booking_Time
1	100009	117	NULL	Pool	06/29/2017 13:00:00	06/29/2017 23:00:00	Phone	05/30/2017 15:00:00

All reserved amenities details

```
SELECT * FROM RESERVED_AMENITY
```

	Reserve_Id	Resident_Id	Group_Id	Amenity_Name	Booking_From	Booking_To	Mode_of_Reserve	Booking_Time
1	100001	NULL	7000001	Community Hall	06/25/2017 10:00:00	06/25/2017 12:00:00	CMS	06/01/2017 09:00:00
2	100002	124	NULL	Pool	07/04/2017 20:00:00	07/04/ 2017 20:00:00	CMS	06/01/2017 16:00:00
3	100003	113	NULL	Community Hall	07/15/2017 18:00:00	07/15/2017 20:00:00	CMS	07/07/2017 18:00:00
4	100004	111	NULL	Pool	06/25/2017 09:00:00	06/25/2017 12:00:00	CMS	06/02/2017 01:00:00
5	100005	112	NULL	Pool	06/26/2017 13:00:00	06/26/2017 17:00:00	CMS	05/31/2017 10:30:00
6	100006	114	NULL	Community Hall	06/26/2017 14:00:00	06/26/2017 16:00:00	CMS	05/30/2017 09:30:00
7	100007	115	NULL	Pool	06/27/2017 10:00:00	06/27/2017 14:00:00	CMS	05/30/2017 11:30:00
8	100008	116	NULL	Community Hall	06/28/2017 17:00:00	06/28/2017 20:00:00	Phone	05/29/2017 13:00:00
9	100009	117	NULL	Pool	06/29/2017 13:00:00	06/29/2017 23:00:00	Phone	05/30/2017 15:00:00
10	1000010	NULL	7000001	Community Hall	06/30/2017 15:00:00	06/30/2017 19:00:00	Visit to Office	06/02/2017 09:00:00

Delete Query

```
DELETE FROM RESERVE_AMENITY
WHERE Reserve_Id = 100009;
```

Amenities reserved after deleting

	Reserve_Id	Resident_Id	Group_Id	Amenity_Name	Booking_From	Booking_To	Mode_of_Reserve	Booking_Time
1	100001	NULL	7000001	Community Hall	06/25/2017 10:00:00	06/25/2017 12:00:00	CMS	06/01/2017 09:00:00
2	100002	124	NULL	Pool	07/04/2017 20:00:00	07/04/ 2017 20:00:00	CMS	06/01/2017 16:00:00
3	100003	113	NULL	Community Hall	07/15/2017 18:00:00	07/15/2017 20:00:00	CMS	07/07/2017 18:00:00
4	100004	111	NULL	Pool	06/25/2017 09:00:00	06/25/2017 12:00:00	CMS	06/02/2017 01:00:00
5	100005	112	NULL	Pool	06/26/2017 13:00:00	06/26/2017 17:00:00	CMS	05/31/2017 10:30:00
6	100006	114	NULL	Community Hall	06/26/2017 14:00:00	06/26/2017 16:00:00	CMS	05/30/2017 09:30:00
7	100007	115	NULL	Pool	06/27/2017 10:00:00	06/27/2017 14:00:00	CMS	05/30/2017 11:30:00
8	100008	116	NULL	Community Hall	06/28/2017 17:00:00	06/28/2017 20:00:00	Phone	05/29/2017 13:00:00
9	1000010	NULL	7000001	Community Hall	06/30/2017 15:00:00	06/30/2017 19:00:00	Visit to Office	06/02/2017 09:00:00

Amenities reserved by Resident after deleting

SELECT * FROM RESERVE_AMENITY WHERE Resident_Id = 117;

Total rows loaded: 0

Reserve_Id	Resident_Id	Group_Id	Amenity_Name	Booking_From	Booking_To	Mode_of_Reserve	Booking_Time

3. A resident can update pick up time in carpool

Details before updating

SELECT * FROM CARPOOL WHERE Carpool_Id = 710001 AND Resident_Id = 115;

Total rows loaded: 1

	Carpool_Id	Resident_Id	Unit	Carpool_From	Carpool_To	Carpool_StartTime
1	710001	115	2105	1190 Benton St.	mountain view	07-02-2017 18:00:02

Update Query
UPDATE CARPOOL
SET Carpool_StartTime = '07-02-2017 18:00:03'
WHERE Resident_Id=115 and Carpool_Id=710001;

Details after updating

SELECT * FROM CARPOOL WHERE Carpool_Id = 710001 AND Resident_Id = 115;

Total rows loaded: 1

	Carpool_Id	Resident_Id	Unit	Carpool_From	Carpool_To	Carpool_StartTime
1	710001	115	2105	1190 Benton St.	mountain view	07-02-2017 18:00:03

4. A resident can look up group members of the group they are in

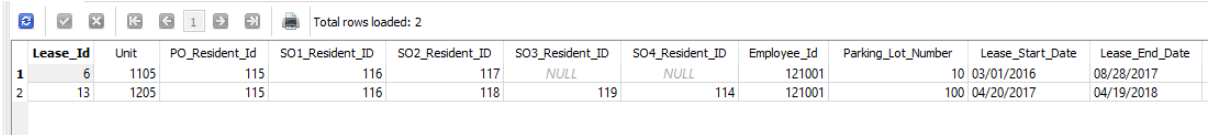
Select Resident_Id from GROUP_MEMBER
Where Group_Id= 7000001 ;

Total rows loaded: 3

	Resident_Id
1	109
2	113
3	115

5. A resident can look up for their lease details

```
Select * From LEASE_AGREEMENT
Where PO_Resident_Id= 115;
```



Total rows loaded: 2

	Lease_Id	Unit	PO_Resident_Id	SO1_Resident_ID	SO2_Resident_ID	SO3_Resident_ID	SO4_Resident_ID	Employee_Id	Parking_Lot_Number	Lease_Start_Date	Lease_End_Date
1	6	1105	115	116	117	NULL	NULL	121001	10	03/01/2016	08/28/2017
2	13	1205	115	116	118	119	114	121001	100	04/20/2017	04/19/2018

4.3 Business Metrics

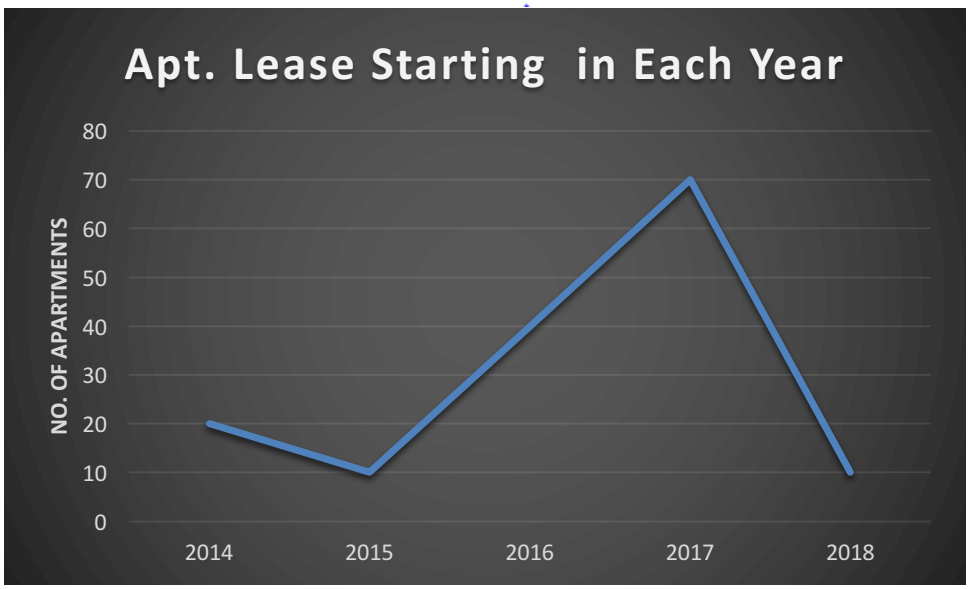
1) Managers will be able to view how many leases are starting every year. This helps the community to analyse their marketing strategies and develop them as per needed.

	Id	Unit	PO_Resident_Id	SO1_Resident_ID	SO2_Resident_ID	SO3_Resident_ID	SO4_Resident_ID	Employee_Id	Parking_Lot_Number	Lease_Start_Date	Lease_End_Date	Rent
1	1	1105	101	106	107	108	NULL	121001	10	2015-01-02	2016-01-01	3000
2	2	2105	102	109	110	NULL	NULL	121001	20	2016-02-02	2019-02-01	2850
3	3	3105	103	NULL	NULL	NULL	NULL	121001	30	2014-05-05	2016-05-04	2900
4	4	4105	104	111	112	113	NULL	121001	40	2018-08-08	2016-08-07	3100
5	5	5105	105	114	NULL	NULL	NULL	121001	50	2014-10-11	2016-10-10	3000
6	6	1105	115	116	117	NULL	NULL	121001	10	2016-03-01	2017-08-28	2800
7	7	3105	118	119	120	121	122	121001	30	2016-05-06	2017-05-05	3000
8	8	4105	123	124	NULL	NULL	NULL	121001	40	2016-08-08	2017-08-07	3150
9	9	6105	113	110	109	NULL	NULL	121001	60	2017-01-05	2017-12-31	2700
10	10	7105	106	NULL	NULL	NULL	NULL	121001	70	2017-02-06	2018-02-05	2800
11	11	8105	107	108	NULL	NULL	NULL	121001	80	2017-03-07	2018-03-08	2900
12	12	9105	110	112	117	NULL	NULL	121001	90	2017-04-08	2018-04-07	2950
13	13	1205	115	116	118	119	114	121001	100	2017-04-20	2018-04-19	3000
14	14	2205	120	121	122	123	103	121001	110	2017-05-21	2018-05-20	3050
15	15	3205	124	NULL	NULL	NULL	NULL	121001	120	2017-06-20	2018-06-19	2500

```
SELECT strftime('%Y', Lease_Start_Date) AS 'Lease_Start_Year', COUNT(Lease_Id) AS
'Number of Leases'
FROM LEASE_AGREEMENT
GROUP BY Lease_Start_Year;
```

Total rows loaded: 5

	Lease_Start_Year	Number of Leases
1	2014	2
2	2015	1
3	2016	4
4	2017	7
5	2018	1



** For plotting graph, additional data has been assumed by multiplying by 10 to show clear flow

Number of Leases Ending in Different Months of Current Year

```
SELECT strftime('%m', Lease_End_Date) AS 'Month_of_LeaseEnd', COUNT(Lease_Id) AS
'Number of Leases Ending'
FROM LEASE_AGREEMENT
WHERE strftime('%Y', Lease_End_Date) = strftime('%Y', 'now')
GROUP BY Month_of_LeaseEnd;
```

Total rows loaded: 3

	Month_of_LeaseEnd	Number of Leases Ending
1	05	1
2	08	2
3	12	1

** For the purpose of plotting graph additional data has been assumed by multiplying by 10 to show clear flow

2) a) Number of prospective residents converted to residents

```
SELECT COUNT(*) AS 'Number of Prospective Residents Converted'
FROM RESIDENT R, PROSPECTIVE_RESIDENT PR
WHERE R.Email_Id = PR.Email;
```

Total rows loaded: 1

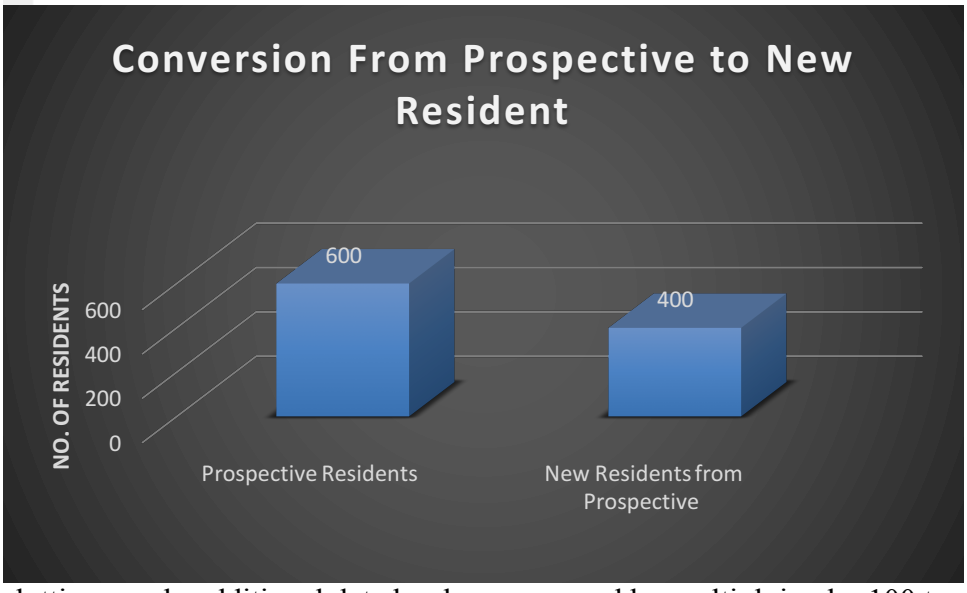
	Number of Prospective Residents Converted
1	4

b) Number of prospective residents not converted to residents

```
SELECT COUNT(*) AS 'Number of Prospective Residents Not Converted' FROM
PROSPECTIVE_RESIDENT PR1
WHERE PR1.Email NOT IN
(SELECT PR.Email
FROM RESIDENT R, PROSPECTIVE_RESIDENT PR
WHERE R.Email_Id = PR.Email);
```

Total rows loaded: 1

	Number of Prospective Residents Not Converted
1	2



** For plotting graph, additional data has been assumed by multiplying by 100 to show clear flow

3) Service Request Business Query

Total rows loaded: 16

	Service_Id	Resident_Id	Unit	Employee_Id	Category	Comments	Request_Time	Status	Comments_Employee	Priority	Mode_of_Request
1	650001	118	1105	121003	Electrical	Bedroom Switch light not working	2017-01-07	IN-PROGRESS	Employee Dispatched to fix the issue	MEDIUM	CMS
2	6500015	115	4105	121006	Plumbing	Kitchen Disposer not functioning	2017-01-28	RESOLVED	Resolved Same Day	High	CMS
3	650002	119	4105	121006	Plumbing	Kitchen Disposer not functioning	2017-06-28	RESOLVED	Resolved Same Day	High	CMS
4	6500012	123	4105	121004	Carpenting	Main Door Step Broken	2017-03-04	RESOLVED	Issue solved	Low	CMS
5	6500014	104	1205	121004	Carpenting	Problem in bedroom door	2017-04-04	RESOLVED	NULL	Low	CMS
6	6500013	113	2105	121006	Plumbing	Tap leaking	2017-04-04	RESOLVED	Issue resolved	Medium	CMS
7	650011	124	1205	121006	Plumbing	Leakage in bathroom	2017-03-08	IN-PROGRESS	Employee is finding the root cause	HIGH	CMS
8	650003	119	4105	121001	External	Bed Bug Treatment	2017-05-07	ASSIGNED	Contract given to External Vendor	HIGH	CMS
9	6500016	118	1105	121003	Electrical	Kitchen power socket not working	2017-05-08	Resolved	Resolved	MEDIUM	CMS
10	650004	117	1105	121003	Electrical	Kitchen power socket not working	2017-05-08	ASSIGNED	Employee Dispatched to fix the issue	MEDIUM	CMS
11	650005	114	3105	121004	Carpenting	Main Door Step Broken	2017-02-07	RECEIVED	Took one day to resolve	LOW	CMS
12	650006	101	6105	121004	Carpenting	Kitchen cabinet broken	2017-06-06	RESOLVED	Cabinet screw replaced	LOW	CMS
13	650007	105	5105	121001	External	Floor Carpet Cleaning	2017-07-03	ASSIGNED	Assigned to vendor	LOW	CMS
14	650008	106	1105	121003	Electrical	NO Power in house	2017-07-01	RESOLVED	Resolved Same Day	URGENT	CMS
15	650009	103	3105	121004	Carpenting	Master Bedroom door jamming	2017-06-09	IN-PROGRESS	Employee Dispatched to fix the issue	LOW	Phone
16	650010	123	4105	121006	Plumbing	Kitchen Faucet leaking	2017-06-02	ASSIGNED	Employee Dispatched to fix the issue	HIGH	Visit to Office

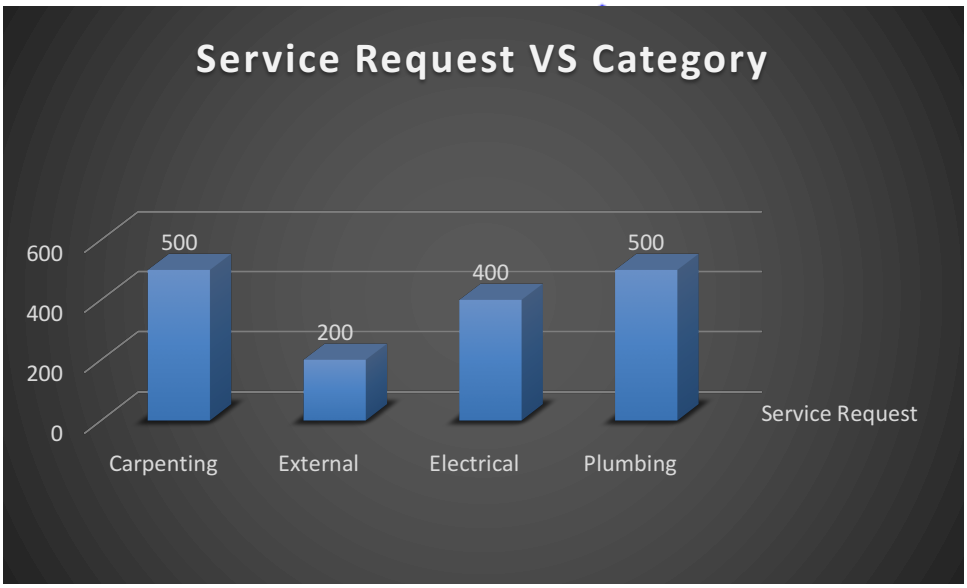
No of Service request by Category

Select Category, COUNT (Category) AS 'Number of Requests Raised' from SERVICE_REQUEST
GROUP BY Category;

Grid view | Form view

Total rows loaded:

	Category	Number of Requests Raised
1	Carpenting	5
2	Electrical	4
3	External	2
4	Plumbing	5



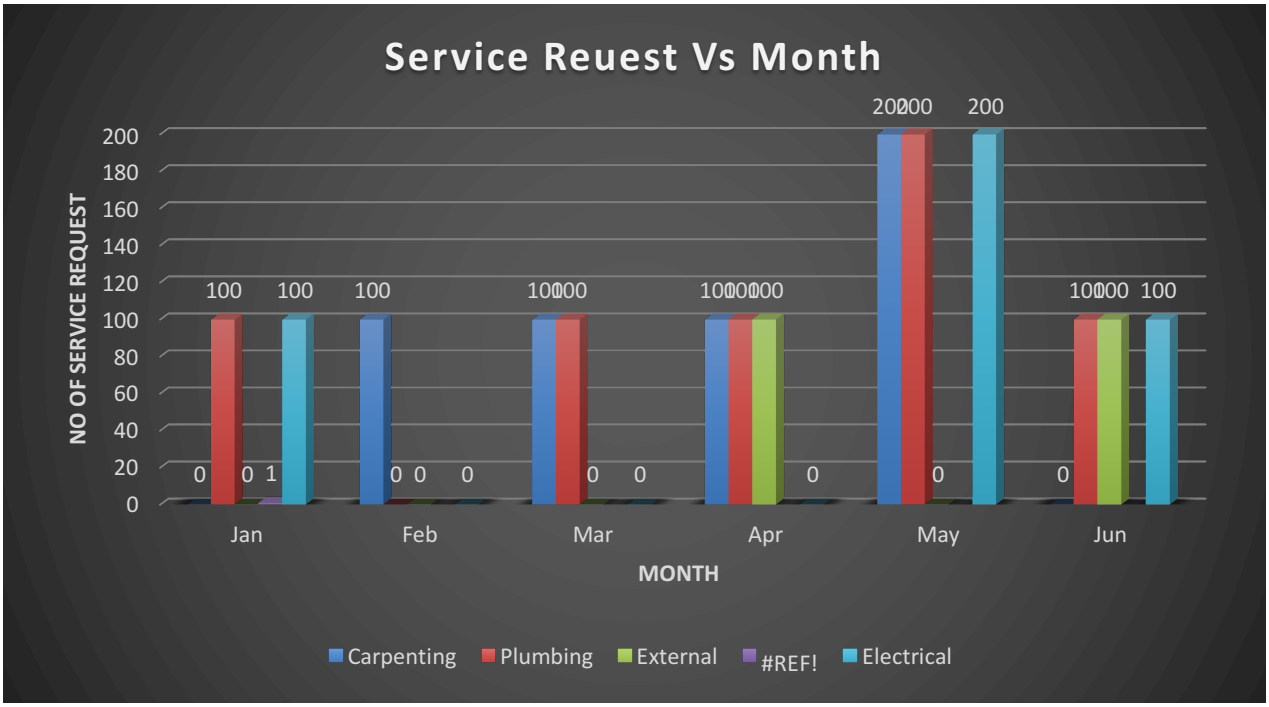
** For the purpose of plotting graph additional data has been assumed by multiplying by 100 to show clear flow

No of request Raised per month per Category for the year 2017

SELECT strftime('%m', Request_Time) AS 'Month_of_Raising_Requests', COUNT(Service_Id) AS 'Number of Service Requests Raised', Category FROM SERVICE_REQUEST
WHERE strftime('%Y', Request_Time) = '2017'
GROUP BY Month_of_Raising_Requests, Category;

Total rows loaded: 13

	Month_of_Raising_Requests	Number of Service Requests Raised	Category
1	01	1	Electrical
2	01	1	Plumbing
3	02	1	Carpenting
4	03	1	Carpenting
5	03	1	Plumbing
6	04	1	Carpenting
7	04	1	Plumbing
8	05	2	Electrical
9	05	1	External
10	06	2	Carpenting
11	06	2	Plumbing
12	07	1	Electrical
13	07	1	External



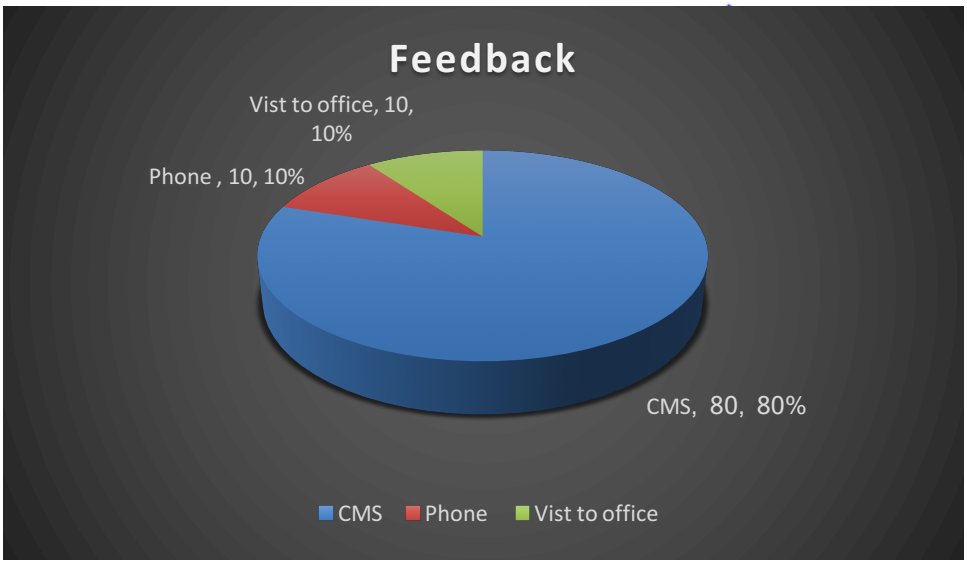
**For the purpose of plotting graph additional data has been assumed multiplying by 100 to show clear flow

4) Feedback Business Query

Select Mode, COUNT(Mode) AS 'Number of Feedbacks Given'
from FEEDBACK
GROUP BY Mode

Total rows loaded: 3

	Mode	Number of Feedbacks Given
1	CMS	8
2	Phone	1
3	Visit to Office	1



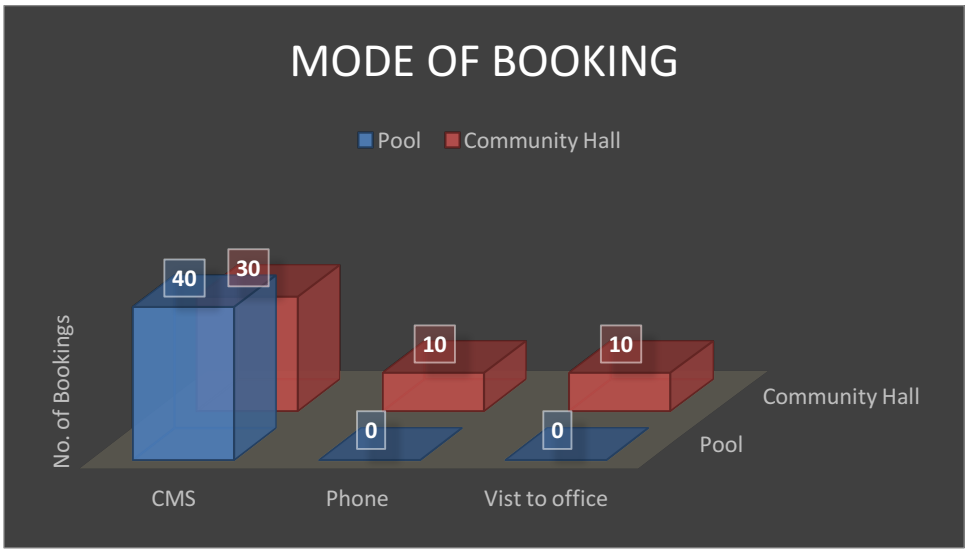
** For the purpose of plotting graph additional data has been assumed by multiplying by 10 to show clear flow

5) Amenities Business Query

Select Amenity_Name AS 'Amenity Name', Mode_of_Reserve AS 'Mode of Reserve', COUNT(Mode_of_Reserve) AS 'Number of Times Amenities Reserved' from RESERVE_AMENITY
GROUP BY Mode_of_Reserve, Amenity_Name;

Total rows loaded: 4

	Amenity Name	Mode of Reserve	Number of Times Amenities Reserved
1	Community Hall	CMS	3
2	Pool	CMS	4
3	Community Hall	Phone	1
4	Community Hall	Visit to Office	1



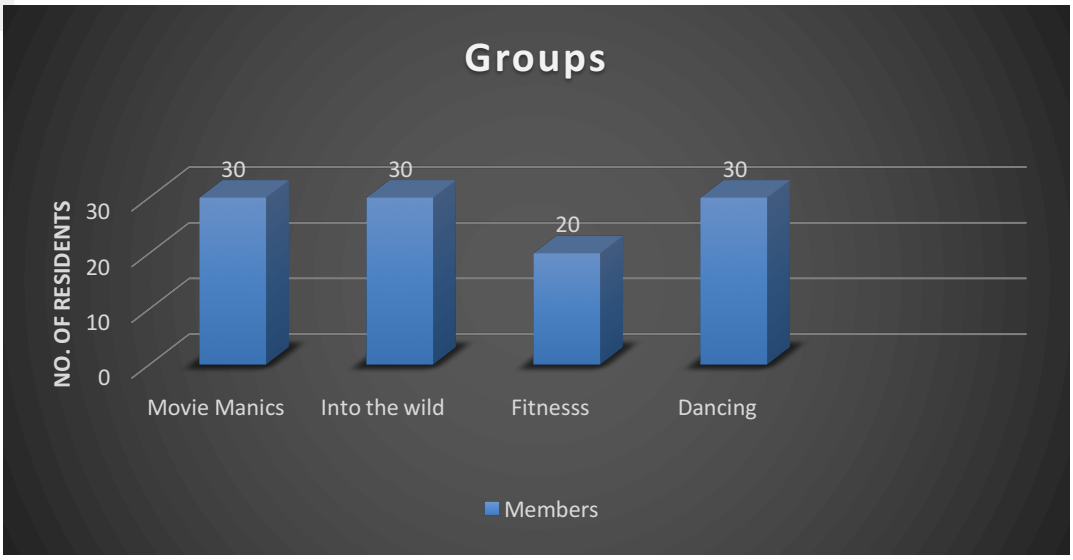
** For the purpose of plotting graph additional data has been assumed by multiplying by 10 to show clear flow

6) Groups Information

```
SELECT G.Group_Id,
Group_Name,
COUNT(GM.Member_Id) AS 'Number of Group Members'
FROM [GROUP] G, GROUP_MEMBER GM
WHERE GM.Group_Id = G.Group_Id
GROUP BY GM.Group_Id;
```

Total rows loaded: 4

	Group_Id	Group_Name	Number of Group Members
1	7000001	Movie Maniacs	3
2	7000002	Into the wild	3
3	7000004	DancingForal	3
4	7000005	Fitness	2



** For plotting graph, additional data has been assumed by multiplying by 10 to show clear flow

Chapter 4:Project Summary

Experience:

This was a very good learning experience, where our concepts of UML, DDL, DML were tested to the core. Project Planning and adhering to the timeline was a key factor that we had keep track of. During the UML phase, we had to go back and forward many times and think about the project and their users logically. The hardest part we feel is the UML and that took most of time we had allocated for the project. Designing tables and identifying attributes and relationships to reduce and redundancy and anomalies was also a challenging experience. The overall project was a huge learning experience and will go a long way to clear our concepts in DBMS.

Hardest Part of the Project:

Main Challenges:

- Limiting the scope of our Business Application.
- Designing the UML.
- Business Metrics to justify the application

Problems:

- Solving Cardinality Issues in our Schema
- Database Population due to multiple Foreign Keys in Most of the tables

Business metrics
Identifying the business metrics and coming up with the queries related to the metrics.

Solution

Database Population: We solved this problem by referring to the UML diagram. This helped us identify the primary and foreign keys of different tables.

Business Metrics: We identified the business metrics by analyzing what business value the metrics would add to our application.

A Different approach to the project

- Implement the wide scope of the project
- Add more functionalities to the existing project
- Make use of newer innovative technologies at our disposal

Suggestions for refining the project

- Clear and early UML preparation
- Use of analysis tools such as Tableau and Large Data Sets to have good flow in Graph