MSIS 2603

TEAM PROJECT

SK Property Management Database

Guojun Wang
Shenghua Yue

11.26.2018

Section 1

1.1 Business/Organization Description

SK Property Management, Inc is a family owned business in the SF bay area. It is recognized as one of the most trusted family-owned business enterprise by local news. The long-term experience in the industry, along with the extensive relationships it has established with partners brings great value to its customers. Its success substantially results from the quick adoption of property management database which streamlines the operation and improves performance. SK only specializes in the management of apartments and provides on-site property management services. Currently, it manages 3 different communities with a total of 300 apartments. Each community is unique in location, view, and services offered.

1.2 Database description

Overall, the scope of the SK database encompasses customers, employees, leases and payment, etc. The database should also provide a solution to manage work orders, parking, and events, etc.

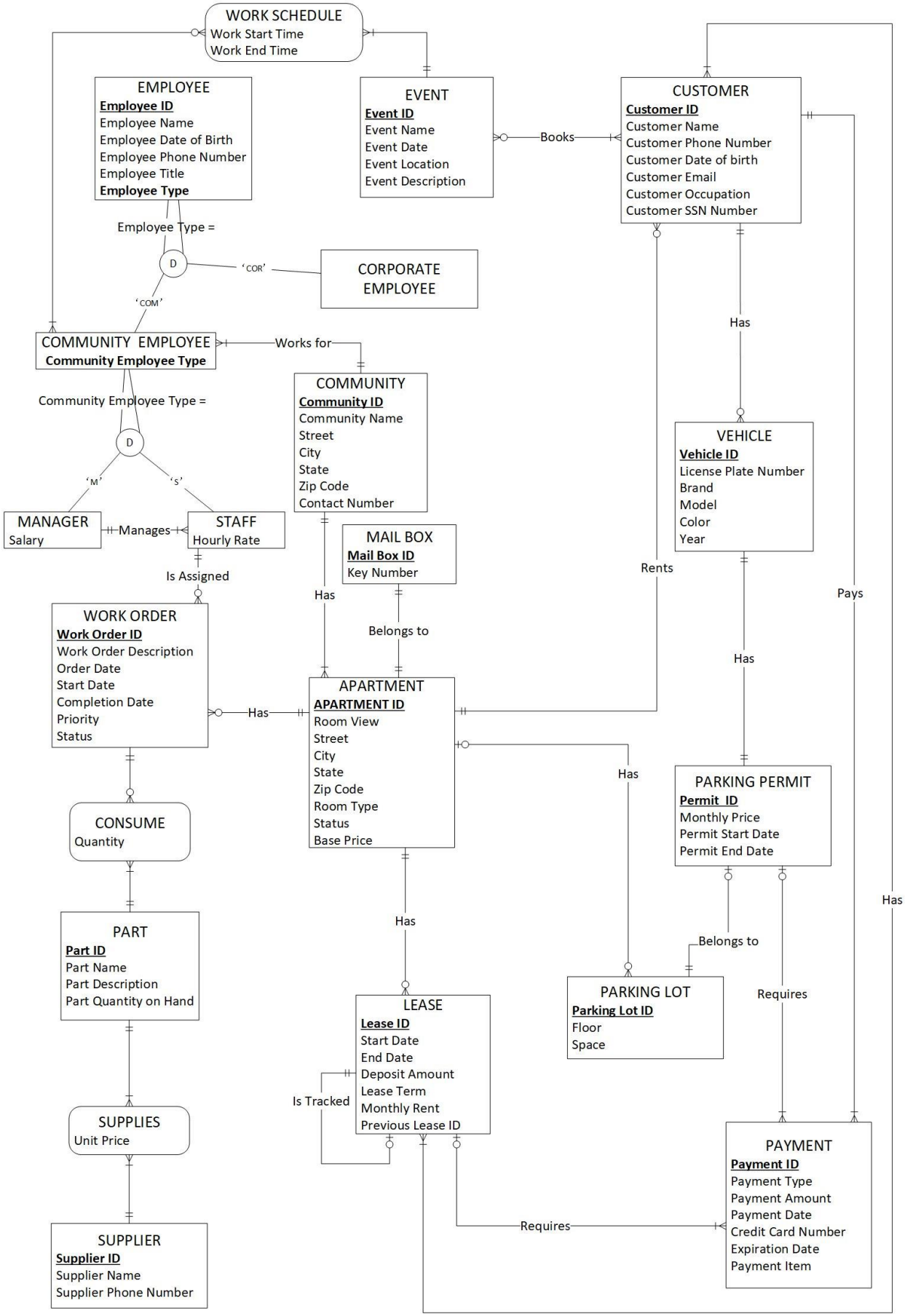
In detail, SK management would like to record information about **customers**, such as name, phone number, date of birth, email, occupation, and SSN. One or several customers sign one lease to move in. **Lease** is described by start date, end date, deposit amount, monthly rent, and lease term. Customer may renew the lease after it is expired and the previous lease is recorded for the tracking purpose. Multiple **payments** are allowed to be made for one lease. Company collects payment information such as payment type, payment amount, payment date, credit card No., expiration date, and payment item which is specified with 'R' for rent or 'P' for parking permit. A customer may have **vehicle(s)** to be parked at the designated parking lot with a valid parking permit. One car requires one permit. Information about vehicle brand, license plate number, model, year, and color should be recorded. The **parking lot** uses a directory system to mark each location with floor number and space number. A parking lot may be assigned to only one apartment. Given one apartment, there may be many parking lots. A customer needs to pay for the parking permit. **Parking permits** are described by monthly price, start date and end date. **An apartment** has its view, street, city, state, zip code, room type, status and base price. The base price should be viewed as a reference price and it associates with the lease term of the apartment. Each apartment may have a lease or be vacant at the moment. An apartment may or may not be rented to customers at the certain time. One customer can only rent one apartment at a time. A **mailbox** is assigned to each apartment and its key number is recorded. Each **community** has its unique address, name, and contact phone number. Each community has minimal 50 apartments. SK property has many **employees**, each of them is described by name, date of birth, phone number, title and employee type. The employees are categorized into **corporate employee** type and **community employee** type. The community employees can only work for one community at a time, and it has two types, **staff**

and **manager**. Staff is hourly paid and manager is salary based. Each manager supervises at least one employee. At least one community employee is responsible to organize one event for customers. **Event** is described by its name, date, location and description. Each event includes at least one **work schedule of community employees.** Work Schedule is described by start time and end time. At least one customer needs to register the event, otherwise it will be canceled. One customer can attend more than one event. SK management is also interested in recording information on the work order. A **work order** is described by, order date, start date, completion date, priority, status and description. One apartment may or may not place many work orders. One community staff may or may not be responsible for work orders. A work order may consume certain quantity of the parts. **Parts** are described by name, description and quantity on hand. One part could be supplied by many suppliers, each with different unit price. The **suppliers** are described by name and phone number.

Section 2

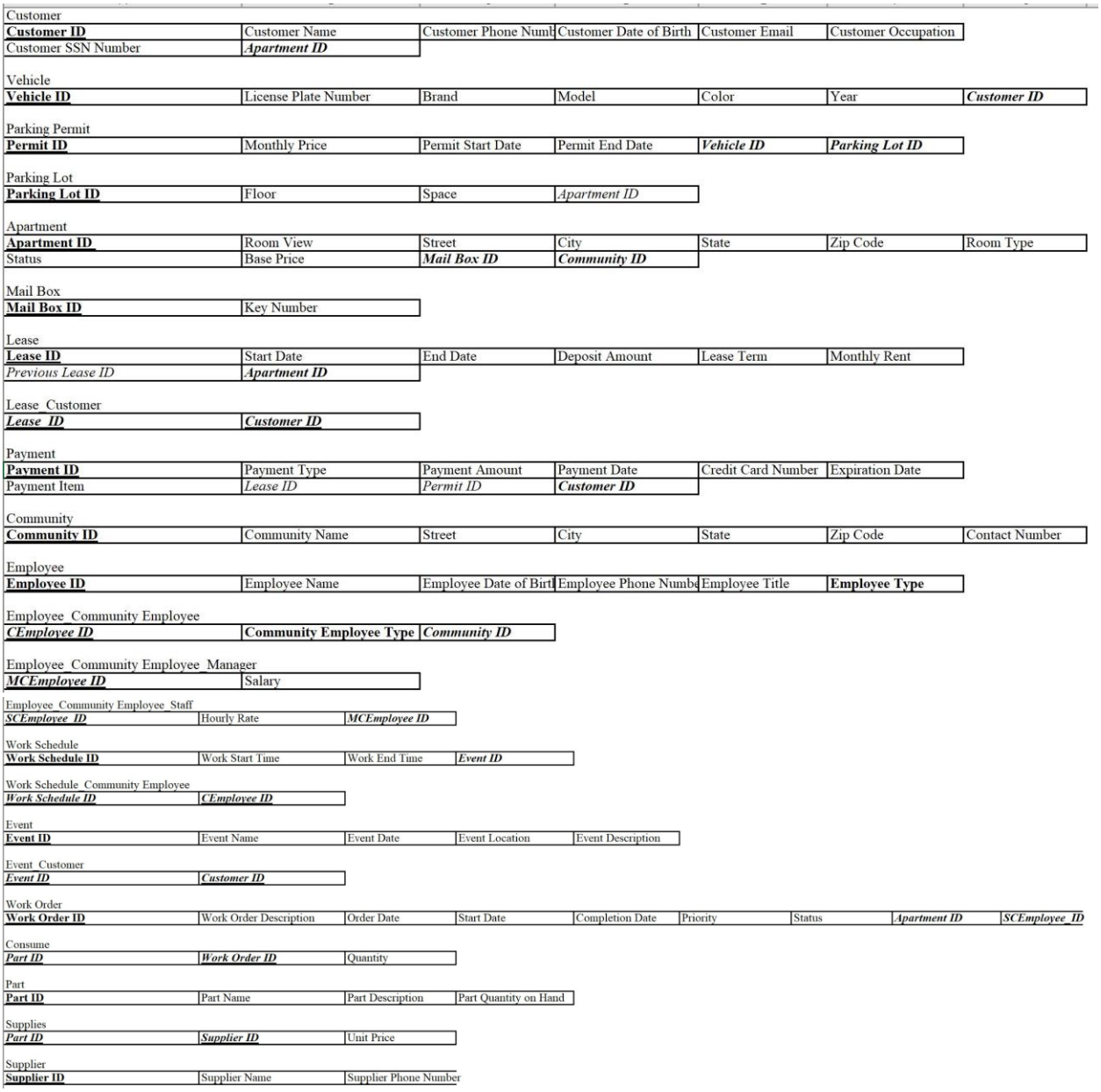
2.1 Conceptual Schema –ER model

Based on the entities and business rules defined above, ER model for SK Property Management is developed as follows. The ER model includes 17 strong entities and 2 super-subtype structures.


WORK SCHEDULE
Work Start Time
Work End Time
EMPLOYEE
Employee ID
Employee Name
Employee Date of Birth
Employee Phone Number
Employee Title
Employee Type
EVENT
Event ID
Event Name
Event Date
Event Location
Event Description
Books
CUSTOMER
Customer ID
Customer Name
Customer Phone Number
Customer Date of birth
Customer Email
Customer Occupation
Customer SSN Number
Employee Type =
D
'COR'
CORPORATE EMPLOYEE
'COM'
COMMUNITY EMPLOYEE
Community Employee Type
Works for
Has
COMMUNITY
Community ID
Community Name
Street
City
State
Zip Code
Contact Number
Community Employee Type =
D
'M'
'S'
VEHICLE
Vehicle ID
License Plate Number
Brand
Model
Color
Year
MANAGER
Salary
Manages
STAFF
Hourly Rate
MAIL BOX
Mail Box ID
Key Number
Rents
Pays
Is Assigned
Has
Belongs to
Has
WORK ORDER
Work Order ID
Work Order Description
Order Date
Start Date
Completion Date
Priority
Status
Has
APARTMENT
APARTMENT ID
Room View
Street
City
State
Zip Code
Room Type
Status
Base Price
Has
PARKING PERMIT
Permit ID
Monthly Price
Permit Start Date
Permit End Date
CONSUME
Quantity
Has
Has
PART
Part ID
Part Name
Part Description
Part Quantity on Hand
Belongs to
PARKING LOT
Parking Lot ID
Floor
Space
Requires
LEASE
Lease ID
Start Date
End Date
Deposit Amount
Lease Term
Monthly Rent
Previous Lease ID
Is Tracked
SUPPLIES
Unit Price
PAYMENT
Payment ID
Payment Type
Payment Amount
Payment Date
Credit Card Number
Expiration Date
Payment Item
Requires
SUPPLIER
Supplier ID
Supplier Name
Supplier Phone Number

2.2 Logical Schema - Relational Table Design

The description of the overall logical structure is presented in the schema with 23 tables in the third normal form.



Customer

Customer ID	Customer Name	Customer Phone Numb	Customer Date of Birth	Customer Email	Customer Occupation	
Customer SSN Number	***Apartment ID***					

Vehicle

Vehicle ID	License Plate Number	Brand	Model	Color	Year	***Customer ID***

Parking Permit

Permit ID	Monthly Price	Permit Start Date	Permit End Date	***Vehicle ID***	***Parking Lot ID***

Parking Lot

Parking Lot ID	Floor	Space	*Apartment ID*

Apartment

Apartment ID	Room View	Street	City	State	Zip Code	Room Type
Status	Base Price	***Mail Box ID***	***Community ID***			

Mail Box

Mail Box ID	Key Number

Lease

Lease ID	Start Date	End Date	Deposit Amount	Lease Term	Monthly Rent
Previous Lease ID	***Apartment ID***				

Lease_Customer

Lease_ID	***Customer ID***

Payment

Payment ID	Payment Type	Payment Amount	Payment Date	Credit Card Number	Expiration Date
Payment Item	*Lease ID*	*Permit ID*	***Customer ID***		

Community

Community ID	Community Name	Street	City	State	Zip Code	Contact Number

Employee

Employee ID	Employee Name	Employee Date of Birt	Employee Phone Numbe	Employee Title	**Employee Type**

Employee_Community Employee

CEmployee ID	**Community Employee Type**	***Community ID***

Employee_Community Employee_Manager

MCEmployee ID	Salary

Employee_Community Employee_Staff

SCEmployee_ID	Hourly Rate	***MCEmployee ID***

Work Schedule

Work Schedule ID	Work Start Time	Work End Time	***Event ID***

Work Schedule_Community Employee

Work Schedule ID	***CEmployee ID***

Event

Event ID	Event Name	Event Date	Event Location	Event Description

Event_Customer

Event ID	***Customer ID***

Work Order

Work Order ID	Work Order Description	Order Date	Start Date	Completion Date	Priority	Status	***Apartment ID***	***SCEmployee_ID***

Consume

Part ID	***Work Order ID***	Quantity

Part

Part ID	Part Name	Part Description	Part Quantity on Hand

Supplies

Part ID	***Supplier ID***	Unit Price

Supplier

Supplier ID	Supplier Name	Supplier Phone Number

2.3 Data dictionary

The metadata are documented in the data dictionary. Each attribute in the table is specified with its data type, constraints, description, example value and data relationships to other objects (PK or FK).

Customer

Name	Data Type	Contraints	Key	Description	Example Value
Customer ID	int	>0	PK	Unique identifier of a customer	390
Customer Name	nvarchar(50)			First and last name of a customer	Michael Jean
Customer Phone Number	char(10)			Phone number of a customer	6505479012
Customer Date of Birth	date			Date of birth for a customer	5/1/1978
Customer Email	nvarchar(50)			Email address of a customer	michael78@gmail.com
Customer Occupation	nvarchar(50)			Occupation of a customer	teacher
Customer SSN Number	char(9)			SSN Number of a customer	522071120
Apartment ID	int	>0	FK	Unique identifier of an apartment	76

VEHICLE

Name	Data Type	Contraints	Key	Description	Example Value
Vehicle ID	int	>0	PK	Unique identifier of a vehicle	20
License Plate Number	nvarchar(50)			License Plate Number of a car	6TRJ286
Brand	nvarchar(50)			Brand of a car	Audi
Model	nvarchar(50)			Model of a car	A4
Color	nvarchar(50)			Color of a car	black
Year	char(4)			Year of model	2000
Customer ID	int	>0	FK	Unique identifier for a customer	390

PARKING PERMIT

Name	Data Type	Contraints	Key	Description	Example Value
Permit ID	int	>0	PK	Unique identifier of a parking permit	66
Monthly Price	decimal(9,2)				50.00
Permit Start Date	date			Start date for a permit	11/20/2017
Permit End Date	date			End date for a permit	11/20/2018
Vehicle ID	int	>0	FK	Unique identifier of a vehicle	20
Parking Lot ID	int	>0	FK	Unique identifier of a parking lot	50

PARKING LOT

Name	Data Type	Contraints	Key	Description	Example Value
Parking Lot ID	int	>0	PK	Unique identifier of a parking lot	50
Floor	char(1)	('1','2')		Floor number of a parking lot	2
Space	int			Space number of a parking lot	30
Apartment ID	int	>0	FK	Unique identifier of an apartment	76

APARTMENT

Name	Data Type	Contraints	Key	Description	Example Value
Apartment ID	int	>0	PK	Unique identifier of an apartment	76
Room View	char(1)	('R','M','N')		View of an apartment as R (River), M(Mountain), or N (None)	R
Unit	nvarchar(10)			Unit number for an apartment	unit 309
Street	nvarchar(100)			Street name for an apartment	302 castro street
City	nvarchar(100)			City name for an apartment	Santa Clara
State	char(2)			State name for an apartment	California
Zip Code	char(5)			Zip code for an apartment	95131
Room Type	varchar(3)	('S', '1BD',		Room type of an apartment as S (studio),1BD (one bedroom), 2BD (2	S
Status	varchar(2)	('A','OC')		Status of apartment as A(available) or OC(occupied)	OC
Base Price	decimal(9,2)			Monthly base price for an apartment	2000.00
Mail Box ID	int	>0	FK	Unique identifier of a mail box	311
Community ID	int	>0	FK	Unique identifier of a community	12456

MAIL BOX

Name	Data Type	Contraints	Key	Description	Example Value
Mail Box ID	int	>0	PK	Unique identifier of a mail box	311
Key Number	nvarchar(25)			Key number of a mail box	311

LEASE

Name	Data Type	Contraints	Key	Description	Example Value
Lease ID	int	>0	PK	Unique identifier of a lease	9511350
Start Date	date			Start date for a lease	11/20/2017
End Date	date			End date for a lease	11/20/2018
Deposit Amount	int	>0		Deposit amount for an apartment	2000.00
Lease Term	char(1)	('M','Q','A')		Lease term for an apartment as monthly paryment ('M'), quarterly payment ('Q') and annual payment ('A')	M
Monthly Rent	decimal(9,2)			Monthly rent for an apartment	2100.00
Previous Lease ID	int	>0	FK	Previous lease ID for an apartment	9511087
Apartment ID	int	>0	FK	Unique identifier for an apartment	76

LEASE_CUSTOMER

Name	Data Type	Contraints	Key	Description	Example Value
Lease ID	int	>0	FK	Unique identifier for a lease	9511350
Customer ID	int	>0	FK	Unique identifier for a customer	390

PAYMENT

Name	Data Type	Contraints	Key	Description	Example Value
Payment ID	int	>0	PK	Unique identifier for a payment	4599
Payment Type	varchar(2)	('CC','DC', 'C')		Payment type as credit card ('CC'), debit card ('DC') and cheque ('C')	
Payment Amount	decimal(9,2)			Payment amount for a payment	2000.00
Payment Date	datetime			Payment date for a payment	12/21/2017 12:00 am
Credit Card Number	char(16)			Credit card number used for a payment	123890998145690
Expiration Date	date			Expiration date for used credit card	3/20/2022
Payment Item	char(1)			Payment item for a customer as apartment rent ('R') and parking permit ('P')	R
Lease ID	int	>0	FK	Unique identifier for a lease	9511350
Permit ID	int	>0	FK	Unique identifier of a parking permit	66
Customer ID	int	>0	FK	Unique identifier of a customer	76

Community

Name	Data Type	Contraints	Key	Description	Example Value
Community ID	int	>0	PK	Unique identifier of an manager from a community	12456
Community Name	nvarchar(50)			Name of a community	Pace
Street	nvarchar(100)			Street name for a community	302 castro street
City	nvarchar(100)			City name for an community	Santa Clara
State	char(2)			State name for a community	California
Zip Code	char(5)			Zip code for a community	95131
Contact Number	char(10)			Contact number for a community	5506507878

Employee

Name	Data Type	Contraints	Key	Description	Example Value
Employee ID	int	>0	PK	Unique identifier for an employee	12456
Employee Name	nvarchar(50)			First and last name of an employee	John Lee
Employee Date of Birth	date			Date of birth for an employee	10/10/1990
Employee Phone Number	char(10)			Phone number for an employee	6507778888
Employee Title	nvarchar(50)			Title of an employee	Customer Associate
Employee Type	char(3)	('Cor', 'Com')		minator for an employee type, community employee (Com) and corporate en	Com

Employee_Community Employee

Name	Data Type	Contraints	Key	Description	Example Value
CEmployee ID	int	>0	PK, FK	Unique identifier for an employee from a community	12456
Community Employee Type	char(1)	('M','S')		Discriminator for a community type, manager(M) or staff (S)	M
Community ID	int	>0	FK	n employee belonging to a community;Unique identifier of a communi	12456

Employee_Community Employee_Manager

Name	Data Type	Contraints	Key	Description	Example Value
MCEmployee ID	int	>0	PK, FK	Unique identifier for an manager from a community	12456
Salary	decimal(9,2)	>0		Pay Rate for an employee for a hour of work	2000

Employee_Community Employee_Staff

Name	Data Type	Contraints	Key	Description	Example Value
SCEmployee ID	int	>0	PK, FK	Unique identifier for a staff from a community	12456
Hourly Rate	decimal(9,2)	>0		Pay Rate for an employee for a hour of work	15.65
MCEmployee ID	int	>0		An manager from community supervising a staff;Unique identifier for a	12456

Work Schedule

Name	Data Type	Contraints	Key	Description	Example Value
Work Schedule ID	int	>0	PK	Unique identifier for an work schedule	12456
Work Start Time	datetime			Start time for the work	11/11/2018 11:00 PM
Work End Time	datetime			End time for the work	11/11/2018 11:00 PM
Event ID	int	>0	FK	Event assoicated with a work schedule;Unique identifier for an event	12456

Work Schedule_Community Employee

Name	Data Type	Contraints	Key	Description	Example Value
Work Schedule ID	int	>0	FK	rk schedule made for an employee;Unique identifier for an work sched	12456
CEmployeeID	int	>0	FK	Community employee working;Unique identifier for a community	12456

Event

Name	Data Type	Contraints	Key	Description	Example Value
Event ID	int	>0	PK	Unique identifier for an event	12456
Event Name	nvarchar(50)			Name of an event	John Legend
Event Date	datetime			Date and time of an event	11/11/2018 11:00 PM
Event Location	nvarchar(50)			Location of an event	Orchard Park
Event Description	nvarchar(100)			Description of an event	Community Safety

Event_Customer

Name	Data Type	Contraints	Key	Description	Example Value
Event ID	int	>0	FK	Event attended by customer;Unique identifier for an event	12456
Customer ID	int	>0	FK	Customer Attending;Unique identifier for a customer	12456

WORK ORDER

Name	Data Type	Contraints	Key	Description	Example Value
Work Order ID	int	>0	PK	Unique identifier for a work order	12456
Work Order Description	nvarchar(100)			Description of a work order	Street needs cleaning
Order Date	datetime			Date and time of work order being placed	10/10/2018 11:00 PM
Start Date	datetime			Date and time of work order to start	11/10/2018 11:00 PM
Completion Date	datetime			Date and time of work order to be completed	12/10/2018 11:00 PM
Priority	char(1)	('L','M','H')		Priority of work order as L (Low), M(Medium), or H (High)	H
Status	char(1)	('P','C','D')		Status of work order as P(pending),C(completed), or D(Delayed)	C
Apartment ID	int	>0	FK	Apartment that needs a work order;Unique identifier for an apartment	12346
SCEmployee ID	int	>0	FK	A staff is assigned to a work order; Unique identifier for a staff	12456

CONSUME

Name	Data Type	Contraints	Key	Description	Example Value
Part ID	int	>0	FK	Part being consumed;Unique identifier for a part	12456
Work Order ID	int	>0	FK	Work order consuming a part;Unique identifier for a work order	12456
Quantity	int	>0		Quantity of a part required to perform the work order	10

PART

Name	Data Type	Contraints	Key	Description	Example Value
Part ID	int	>0	PK	Unique identifier for a part	12456
Part Name	nvarchar(50)			Name of a part	Bulb
Part Description	nvarchar(50)			Description of a part	50w 220 v
Part Quantity on Hand	int	>=0		Quantity of a part on hand in inventory	10

SUPPLIES

Name	Data Type	Contraints	Key	Description	Example Value
Supplier ID	int	>0	FK	Supplier supplying an part; Unique identifier for a supplier	12345
Part ID	int	>0	FK	Part supplied by a supplier;Unique identifier for a part	12456
Unit Price	decimal(9,2)	>0		Cost per unit of a part	65.22

SUPPLIER

Name	Data Type	Contraints	Key	Description	Example Value
Supplier ID	int	>0	PK	Unique identifier for a supplier	12345
Supplier Name	nvarchar(50)			First and last name of a supplier	Joe Biden
Supplier Phone Number	char(10)			Phone number of a supplier	8081234567

Section 3

3.1 SQL statements to create each table

According to the data dictionary described above, 23 tables are created by the following statements in the SQL management studio. In each table, the data type of each attribute and its corresponding constraints are identified. Moreover, the referential integrity constraints are defined.

```
Mail Box
CREATE TABLE Mail_Box
(MailBoxID int not null check (MailBoxID>0),
KeyNumber nvarchar(25),
CONSTRAINT Mail_Box_PK PRIMARY KEY (MailBoxID));

Community
CREATE TABLE Community
(CommunityID int not null check (CommunityID>0),
CommunityName nvarchar(50),
Street nvarchar(100),
City nvarchar(100),
State char(2),
ZipCode char(5),
ContactNumber char(10),
CONSTRAINT Community_PK PRIMARY KEY (CommunityID));

Apartment
CREATE TABLE Apartment
(ApartmentID int not null check (ApartmentID>0),
RoomView char(1) CHECK (RoomView IN ('R','M','N')),
Unit nvarchar(10),
Street nvarchar(100),
City nvarchar(100),
State char(2),
ZipCode char(5),
RoomType varchar(3) CHECK (RoomType IN ('S', '1BD', '2BD', '3BD')),
Status varchar(2) CHECK (Status IN ('A','OC')),
BasePrice decimal(9,2),
MailBoxID int not null,
CommunityID int not null,
CONSTRAINT Apartment_PK PRIMARY KEY (ApartmentID),
CONSTRAINT Apartment_FK1 FOREIGN KEY (MailBoxID) REFERENCES Mail_Box(MailBoxID),
CONSTRAINT Apartment_FK2 FOREIGN KEY (CommunityID) REFERENCES Community(CommunityID));
```

```
Parking Lot
CREATE TABLE Parking_Lot
(ParkingLotID int not null check (ParkingLotID>0),
Floor char(1) CHECK (Floor IN ('1','2')),
Space int not null,
ApartmentID int,
CONSTRAINT Parking_Lot_PK PRIMARY KEY (ParkingLotID),
CONSTRAINT Parking_Lot_FK FOREIGN KEY (ApartmentID) REFERENCES Apartment(ApartmentID));

Customer
CREATE TABLE Customer
(CustomerID int not null check (CustomerID >0),
CustomerName nvarchar(50) not null,
CustomerPhoneNumber char(10),
CustomerBirthday date,
CustomerEmail nvarchar(50),
CustomerOccupation nvarchar(50),
CustomerSSNNumber char(9),
ApartmentID int not null,
CONSTRAINT Customer_PK PRIMARY KEY (CustomerID),
CONSTRAINT Customer_FK FOREIGN KEY (ApartmentID) REFERENCES Apartment(ApartmentID));

Vehicle
CREATE TABLE Vehicle
(VehicleID int not null check (VehicleID>0),
LicensePlateNumber      nvarchar(50),
Brand   nvarchar(50),
Model   nvarchar(50),
Color   nvarchar(50),
Year    char(4),
CustomerID      int not null,
CONSTRAINT Vehicle_PK PRIMARY KEY (VehicleID),
CONSTRAINT Vehicle_FK FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID));
```

```
Parking Permit
CREATE TABLE Parking_Permit
(PermitID int not null check (PermitID>0),
MonthlyPrice decimal(9,2),
PermitStartDate date,
PermitEndDate   date,
VehicleID int not null,
ParkingLotID int not null,
CONSTRAINT Parking_Permit_PK PRIMARY KEY (PermitID),
CONSTRAINT Parking_Permit_FK1 FOREIGN KEY (VehicleID) REFERENCES Vehicle(VehicleID),
CONSTRAINT Parking_Permit_FK2 FOREIGN KEY (ParkingLotID) REFERENCES Parking_Lot(ParkingLotID));

Lease
CREATE TABLE Lease
(LeaseID int not null check (LeaseID>0),
StartDate       date,
EndDate date,
DepositAmount int check (DepositAmount>0),
LeaseTerm       char(1) CHECK (LeaseTerm IN ('M','Q','A')),
MonthlyRent decimal(9,2),
PreviousLeaseID int,
ApartmentID     int     not null,
CONSTRAINT Lease_PK PRIMARY KEY (LeaseID),
CONSTRAINT Lease_FK1 FOREIGN KEY (PreviousLeaseID) REFERENCES Lease(LeaseID),
CONSTRAINT Lease_FK2 FOREIGN KEY (ApartmentID) REFERENCES Apartment(ApartmentID));

Lease_Customer
CREATE TABLE Lease_Customer
(LeaseID int not null,
CustomerID int not null,
CONSTRAINT Lease_Customer_PK PRIMARY KEY (LeaseID, CustomerID),
CONSTRAINT Lease_Customer_FK1 FOREIGN KEY (LeaseID) REFERENCES Lease(LeaseID),
CONSTRAINT Lease_Customer_FK2 FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID));
Payment
CREATE TABLE Payment
(PaymentID      int     not null check (PaymentID>0),
PaymentType     varchar(2)      CHECK (PaymentType IN ('CC','DC', 'C')),
PaymentAmount   decimal(9,2),
PaymentDate     datetime,
CreditCardNumber        char(16),
ExpirationDate  date,
PaymentItem     char(1) CHECK (PaymentItem IN ('P','R')),
LeaseID int,
PermitID int,
CONSTRAINT Payment_PK PRIMARY KEY (PaymentID),
CONSTRAINT Payment_FK1 FOREIGN KEY (LeaseID) REFERENCES Lease(LeaseID),
CONSTRAINT Payment_FK2 FOREIGN KEY (PermitID) REFERENCES Parking_Permit(PermitID));

Employee
CREATE TABLE Employee
(EmployeeID     int     not null check (EmployeeID>0),
EmployeeName nvarchar(50) not null,
EmployeeDateofBirth     date,
EmployeePhoneNumber     char(10),
EmployeeTitle nvarchar(50),
EmployeeType char(3) not null CHECK (EmployeeType IN ('Cor', 'Com')),
CONSTRAINT Employee_PK PRIMARY KEY (EmployeeID));

Employee_Community_Employee
CREATE TABLE Community_Employee
(CEmployeeID    int not null check (CEmployeeID>0),
CommunityEmployeeType char(1) not null CHECK (CommunityEmployeeType IN ('M','S')),
CommunityID     int not null,
CONSTRAINT Community_Employee_PK PRIMARY KEY (CEmployeeID),
CONSTRAINT Community_Employee_FK1 FOREIGN KEY (CEmployeeID) REFERENCES Employee(EmployeeID),
CONSTRAINT Community_Employee_FK2 FOREIGN KEY (CommunityID) REFERENCES Community(CommunityID));
```

```
Employee_Community Employee_Manager
CREATE TABLE Community_Employee_Manager
(MCEmployeeID   int not null check (MCEmployeeID>0),
Salary  decimal(9,2) check (Salary>0),
CONSTRAINT Community_Employee_Manager_PK PRIMARY KEY (MCEmployeeID),
CONSTRAINT Community_Employee_Manager_FK1 FOREIGN KEY (MCEmployeeID) REFERENCES Community_Employee(CEmployeeID));


Employee_Community Employee_Staff
CREATE TABLE Community_Employee_Staff
(SCEmployeeID   int     not null check (SCEmployeeID>0),
HourlyRate decimal(9,2) check (HourlyRate>0),
MCEmployeeID int not null,
CONSTRAINT Community_Employee_Staff_PK PRIMARY KEY (SCEmployeeID),
CONSTRAINT Community_Employee_Staff_FK1 FOREIGN KEY (SCEmployeeID) REFERENCES Community_Employee(CEmployeeID),
CONSTRAINT Community_Employee_Staff_FK2 FOREIGN KEY (MCEmployeeID) REFERENCES Community_Employee_Manager(MCEmployeeID));


Event
CREATE TABLE Event
(EventID int not null   check(EventID>0),
EventName       nvarchar(50),
EventDate       datetime,
EventLocation   nvarchar(50),
EventDescription        nvarchar(100),
CONSTRAINT Event_PK PRIMARY KEY (EventID));

Work Schedule
CREATE TABLE Work_Schedule
(WorkScheduleID int not null check(WorkScheduleID>0),
WorkStartTime   datetime,
WorkEndTime     datetime,
EventID int     not null
CONSTRAINT Work_Schedule_PK PRIMARY KEY (WorkScheduleID),
CONSTRAINT Work_Schedule_FK1 FOREIGN KEY (EventID) REFERENCES Event(EventID));
Work Schedule_Community Employee
CREATE TABLE Work_Schedule_Community_Employee
(WorkScheduleID int not null,
CEmployeeID     int     not null,
CONSTRAINT Work_Schedule_Community_Employee_PK PRIMARY KEY (WorkScheduleID, CEmployeeID),
CONSTRAINT Work_Schedule_Community_Employee_FK1 FOREIGN KEY (WorkScheduleID) REFERENCES Work_Schedule(WorkScheduleID),
CONSTRAINT Work_Schedule_Community_Employee_FK2 FOREIGN KEY (CEmployeeID) REFERENCES Community_Employee(CEmployeeID));


Event_Customer
CREATE TABLE Event_Customer
(EventID        int     not null,
CustomerID      int not null,
CONSTRAINT Event_Customer_PK PRIMARY KEY (EventID, CustomerID),
CONSTRAINT Event_Customer_FK1 FOREIGN KEY (EventID) REFERENCES Event(EventID),
CONSTRAINT Event_Customer_FK2 FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID));


Work Order
CREATE TABLE Work_Order
(WorkOrderID    int     not null check(WorkOrderID>0),
WorkOrderDescription    nvarchar(100),
OrderDate       datetime,
StartDate       datetime,
CompletionDate  datetime,
Priority        char(1) check(Priority in ('L','M','H')),
Status  char(1) check(Status in('P','C','D')),
ApartmentID     int     not null,
SCEmployeeID int        not null,
CONSTRAINT Work_Order_PK PRIMARY KEY (WorkOrderID),
CONSTRAINT Work_Order_FK1 FOREIGN KEY (ApartmentID) REFERENCES Apartment(ApartmentID),
CONSTRAINT Work_Order_FK2 FOREIGN KEY (SCEmployeeID) REFERENCES Community_Employee_Staff(SCEmployeeID));
```

```
Part
CREATE TABLE Part
(PartID int not null check(PartID>0),
PartName nvarchar(50),
PartDescription nvarchar(100),
PartQuantityonHand int  check(PartQuantityonHand>=0),
CONSTRAINT Part_PK PRIMARY KEY (PartID));

Consume
CREATE TABLE Consume
(PartID int     not null,
WorkOrderID     int     not null,
Quantity        int     check(Quantity>0),
CONSTRAINT Consume_PK PRIMARY KEY (PartID, WorkOrderID),
CONSTRAINT Consume_FK1 FOREIGN KEY (PartID) REFERENCES Part(PartID),
CONSTRAINT Consume_FK2 FOREIGN KEY (WorkOrderID) REFERENCES Work_Order(WorkOrderID));

Supplier
CREATE TABLE Supplier
(SupplierID     int not null check(SupplierID > 0),
SupplierName nvarchar(50),
SupplierPhoneNumber     char(10),
CONSTRAINT Supplier_PK PRIMARY KEY (SupplierID));

Supplies
CREATE TABLE Supplies
(SupplierID     int     not null,
PartID  int     not null,
UnitPrice decimal(9,2) check(UnitPrice > 0),
CONSTRAINT Supplies_PK PRIMARY KEY (SupplierID, PartID),
CONSTRAINT Supplies_FK1 FOREIGN KEY (SupplierID) REFERENCES Supplier(SupplierID),
CONSTRAINT Supplies_FK2 FOREIGN KEY (PartID) REFERENCES Part(PartID));
```

3.2 SQL statements for materialized views with justifications

To improve the database security and efficiency, 6 materialized views for 3 distinctive users (2 views for each) are designed in this section. Accounting department is interested in tracking the payments of lease and parking permit for our customers. Lease payment detail and permit payment detail are created for this purpose. Views of Available Apartment in Community and Available Event Detail are designed to improve customer experience. Furthermore, view of Staff Bonus Detail from Event facilitates our internal management when evaluating the performance bonus and counting total working hours. Work Order Cost view ordered by year and month gives managers a holistic view of total cost of work orders in each month and immediately uncovers potential abnormal amount in a certain month. The last two views are organized as summary tables based on aggregations of the data from master tables.

User Group 1: Accounting Department

View1: Lease Payment Detail

Query:

```
CREATE TABLE LeasePaymentDetail_view
(CustomerID int not null,
CustomerName nvarchar(50),
CustomerPhoneNumber char(10),
CustomerEmail nvarchar(50),
ApartmentID int not null,
LeaseEndDate date,
PaymentType     varchar(2)      CHECK (PaymentType IN ('CC','DC', 'C')),
PaymentDate datetime,
MonthlyRent decimal(9,2),
PaymentAmount decimal(9,2),
Balance decimal(9,2));

create procedure RefleshLeasePaymentDetail_View as
delete from LeasePaymentDetail_view
insert into LeasePaymentDetail_view
select Customer.CustomerID, Customer.CustomerName, Customer.CustomerPhoneNumber, Customer.CustomerEmail,
Customer.ApartmentID, Lease.EndDate, Payment.PaymentType, Payment.PaymentDate,
Lease.MonthlyRent, Payment.PaymentAmount, (Payment.PaymentAmount - Lease.MonthlyRent) as Balance
from Customer, Lease, Payment
where Customer.ApartmentID = Lease.ApartmentID
and Lease.LeaseID = Payment.LeaseID
and Payment.PaymentItem = 'R'
order by payment.PaymentDate;
```

Results Messages

	CustomerID	CustomerName	CustomerPhoneNumber	CustomerEmail	ApartmentID	EndDate	PaymentType	PaymentDate	MonthlyRent	PaymentAmount	Balance
1	4001	Banana	6567872222	Banana@yahoo.com	101	2017-12-16	CC	2017-09-17 00:00:00.000	3200.00	3000.00	-200.00
2	4022	Babara	1236927689	babaralucky@gmail.com	101	2017-12-16	CC	2017-09-17 00:00:00.000	3200.00	3000.00	-200.00
3	4001	Banana	6567872222	Banana@yahoo.com	101	2018-12-16	DC	2017-12-17 00:00:00.000	3000.00	3000.00	0.00
4	4022	Babara	1236927689	babaralucky@gmail.com	101	2018-12-16	DC	2017-12-17 00:00:00.000	3000.00	3000.00	0.00
5	4017	Avjduo	6502906666	Avacado@hotmail.com	117	2019-01-14	C	2018-01-15 00:00:00.000	2100.00	2100.00	0.00
6	4009	Gendjs	9351102080	Ban@yahoo.com	109	2019-01-16	DC	2018-01-17 00:00:00.000	2500.00	2500.00	0.00
7	4003	Honet	8141339696	bee@gmail.com	120	2019-02-16	CC	2018-02-17 00:00:00.000	2500.00	2500.00	0.00
8	4015	Adde	4087029899	apple@gmail.com	115	2019-02-16	CC	2018-02-17 00:00:00.000	2500.00	2500.00	0.00
9	4013	Hwow	8141102896	bee@gmail.com	113	2018-01-16	CC	2018-10-17 00:00:00.000	3200.00	3200.00	0.00
10	4019	dkis	9827438080	Banana@yahoo.com	119	2018-01-18	C	2018-10-19 00:00:00.000	3200.00	3200.00	0.00
11	4010	Apeld	4082769899	apple@gmail.com	110	2019-10-29	DC	2018-10-30 00:00:00.000	2100.00	2100.00	0.00
12	4006	Baa	6528972222	Banana@yahoo.com	106	2018-12-01	C	2018-11-02 00:00:00.000	5600.00	5600.00	0.00
13	4007	Avado	6507988666	Avacado@hotmail.com	123	2019-02-06	DC	2018-11-07 00:00:00.000	3200.00	3200.00	0.00
14	4011	Banekw	6567829222	Banana@yahoo.com	111	2018-12-14	DC	2018-11-15 00:00:00.000	5600.00	5600.00	0.00
15	4020	Mega	6567890564	mega@gmail.com	111	2018-12-14	DC	2018-11-15 00:00:00.000	5600.00	5600.00	0.00
16	4018	Hmsk	8142927696	apple@gmail.com	118	2018-12-15	C	2018-11-16 00:00:00.000	5600.00	5600.00	0.00
17	4000	Apple	4087679899	apple@gmail.com	100	2018-12-15	CC	2018-11-16 00:00:00.000	5600.00	5600.00	0.00
18	4021	Tony	4502908789	tony85@gmail.com	100	2018-12-15	CC	2018-11-16 00:00:00.000	5600.00	5600.00	0.00
19	4008	Hondset	8141555696	bee@gmail.com	108	2019-11-17	DC	2018-11-18 00:00:00.000	2100.00	2100.00	0.00
20	4014	Gendhs	9351431720	Ban@yahoo.com	114	2019-11-17	CC	2018-11-18 00:00:00.000	2100.00	2000.00	-100.00
21	4002	Avacado	6507876666	Avacado@hotmail.com	102	2019-11-17	CC	2018-11-18 00:00:00.000	2100.00	2100.00	0.00
22	4012	Avawlo	6507809066	Avacado@hotmail.com	122	2018-12-24	DC	2018-11-25 00:00:00.000	5600.00	5600.00	0.00
23	4004	Gen	9351438080	Ban@yahoo.com	104	2019-11-29	C	2018-11-30 00:00:00.000	2100.00	2100.00	0.00
24	4016	Badha	6567555222	Banana@yahoo.com	121	2019-11-29	CC	2018-11-30 00:00:00.000	2100.00	2100.00	0.00
25	4005	App	4089029899	apple@gmail.com	105	2019-01-14	C	2018-12-15 00:00:00.000	5600.00	5600.00	0.00
26	4011	Banekw	6567829222	Banana@yahoo.com	111	2019-12-14	DC	2018-12-15 00:00:00.000	2100.00	2100.00	0.00
27	4020	Mega	6567890564	mega@gmail.com	111	2019-12-14	DC	2018-12-15 00:00:00.000	2100.00	2100.00	0.00
28	4000	Apple	4087679899	apple@gmail.com	100	2018-01-15	DC	2018-12-16 00:00:00.000	5600.00	5600.00	0.00
29	4021	Tony	4502908789	tony85@gmail.com	100	2018-01-15	DC	2018-12-16 00:00:00.000	5600.00	5600.00	0.00

Justification

The above view facilitates employees from accounting department to find out which customers did not pay the exact lease amount. The employees have an access to only certain information about customers which is sufficient enough to finish their audit work. With the certain information, for example, customer Banana pays less than the monthly rent, an accountant is able to forward this information to a community staff member to investigate further. The community staff should be able to easily find the customer apartment and contact information to go after the customer.

View2: Permit Payment Detail

Query:

```
CREATE TABLE PermitPaymentDetail_view
(CustomerID int not null,
CustomerName nvarchar(50),
CustomerPhoneNumber char(10),
CustomerEmail nvarchar(50),
VehicleID int not null,
LicensePlateNumber nvarchar(50),
PaymentType     varchar(2)      CHECK (PaymentType IN ('CC','DC', 'C')),
PaymentDate datetime,
MonthlyPrice decimal(9,2),
PaymentAmount decimal(9,2),
Balance decimal(9,2));

create procedure RefleshPermitPaymentDetail_View as
delete from PermitPaymentDetail_view
insert into PermitPaymentDetail_view
select Customer.CustomerID, Customer.CustomerName, Customer.CustomerPhoneNumber, Customer.CustomerEmail,
Vehicle.VehicleID, Vehicle.LicensePlateNumber, Payment.PaymentType, Payment.PaymentDate,
Parking_Permit.MonthlyPrice, Payment.PaymentAmount, (Payment.PaymentAmount - Parking_Permit.MonthlyPrice) as Balance
from Customer, Vehicle, Parking_Permit, Payment
where Customer.CustomerID = Vehicle.CustomerID
and Vehicle.VehicleID = Parking_Permit.VehicleID
and Payment.PermitID = Parking_Permit.PermitID
and Payment.PaymentItem = 'P'
order by payment.PaymentDate;
```

Results | Messages

	CustomerID	CustomerName	CustomerPhoneNumber	CustomerEmail	VehicleID	LicensePlateNumber	PaymentType	PaymentDate	MonthlyPrice	PaymentAmount	Balance
1	4001	Banana	6567872222	Banana@yahoo.com	700000001	CHD789	DC	2017-09-17 00:00:00.000	50.00	100.00	50.00
2	4001	Banana	6567872222	Banana@yahoo.com	700000001	CHD789	C	2017-12-17 00:00:00.000	50.00	100.00	50.00
3	4017	Avjduo	6502906666	Avacado@hotmail.com	700000017	C77777	CC	2018-01-15 00:00:00.000	50.00	200.00	150.00
4	4009	Gendjs	9351102080	Ban@yahoo.com	700000009	BV6272	C	2018-01-17 00:00:00.000	50.00	50.00	0.00
5	4003	Honet	8141339696	bee@gmail.com	700000003	BV64JD7	CC	2018-02-17 00:00:00.000	50.00	50.00	0.00
6	4015	Adde	4087029899	apple@gmail.com	700000015	Bci77792	DC	2018-02-17 00:00:00.000	50.00	30.00	-20.00
7	4013	Hwow	8141102896	bee@gmail.com	700000013	CHD2836	DC	2018-10-17 00:00:00.000	50.00	100.00	50.00
8	4019	dkis	9827438080	Banana@yahoo.com	700000019	CH862	CC	2018-10-19 00:00:00.000	50.00	100.00	50.00
9	4010	Apeld	4082769899	apple@gmail.com	700000010	CHD79277	DC	2018-10-30 00:00:00.000	50.00	100.00	50.00
10	4006	Baa	6528972222	Banana@yahoo.com	700000006	BV6287	CC	2018-11-02 00:00:00.000	50.00	50.00	0.00
11	4007	Avado	6507988666	Avacado@hotmail.com	700000007	CHDd765	C	2018-11-07 00:00:00.000	50.00	100.00	50.00
12	4011	Banekw	6567829222	Banana@yahoo.com	700000011	C7003865	DC	2018-11-15 00:00:00.000	50.00	200.00	150.00
13	4018	Hmsk	8142927696	apple@gmail.com	700000018	B8884JD	CC	2018-11-16 00:00:00.000	50.00	50.00	0.00
14	4000	Apple	4087679899	apple@gmail.com	700000000	BV64JD	DC	2018-11-16 00:00:00.000	50.00	50.00	0.00
15	4008	Hondset	8141555696	bee@gmail.com	700000008	C702762	C	2018-11-18 00:00:00.000	50.00	100.00	50.00
16	4014	Gendhs	9351431720	Ban@yahoo.com	700000014	C703866	DC	2018-11-18 00:00:00.000	50.00	200.00	150.00
17	4002	Avacado	6507876666	Avacado@hotmail.com	700000002	C699999	DC	2018-11-18 00:00:00.000	50.00	200.00	150.00
18	4012	Avawlo	6507809066	Avacado@hotmail.com	700000012	BV8265	DC	2018-11-25 00:00:00.000	50.00	50.00	0.00
19	4004	Gen	9351438080	Ban@yahoo.com	700000004	CHD790ds	CC	2018-11-30 00:00:00.000	50.00	100.00	50.00
20	4016	Badha	6567555222	Banana@yahoo.com	700000016	CHDd872	CC	2018-11-30 00:00:00.000	50.00	100.00	50.00
21	4005	App	4089029899	apple@gmail.com	700000005	C700028	CC	2018-12-15 00:00:00.000	50.00	200.00	150.00
22	4022	Babara	1236927689	babaralucky@gmail.c...	700000020	BB862VM89	C	2018-12-15 00:00:00.000	50.00	200.00	150.00
23	4000	Apple	4087679899	apple@gmail.com	700000000	BV64JD	C	2018-12-16 00:00:00.000	50.00	50.00	0.00

Justification:

The above view facilitates employees from accounting department to find out which customers did not pay the monthly parking permit. The employees have an access to only certain information about customers which is sufficient enough to finish their audit work. With the certain information, for example, many customers pay more than the monthly price, an accountant is able to forward the limited customer information to a community staff member to investigate further. The community staff should be able to conclude that customers would like to pay the parking permit for more than one month at a time. Since the parking permit monthly fee is cheap, it is more convenient for customers to pay lump sum.

User Group 2: Customers

View1: Available Apartment in Community

Query:

```
CREATE TABLE AvailAparmentDetail_view
(ApartmentID int not null,
CommunityName nvarchar(50),
Street nvarchar(100),
City nvarchar(100),
ContactNumber char(10),
Unit nvarchar(10),
RoomType varchar(3) CHECK (RoomType IN ('S', '1BD', '2BD', '3BD')),
RoomView char(1) CHECK (RoomView IN ('R','M','N')),
BasePrice decimal(9,2));

create procedure RefreshAvailAparmentDetail_view as
delete from AvailAparmentDetail_view
insert into AvailAparmentDetail_view
Select Apartment.ApartmentID, Community.CommunityName, Community.Street,  Community.City, Community.ContactNumber,
Apartment.Unit, Apartment.RoomType, Apartment.RoomView, Apartment.BasePrice
from Apartment, Community
where Apartment.CommunityID = Community.CommunityID and Apartment.Status='A'
order by Apartment.ApartmentID;
```

	ApartmentID	CommunityName	Street	City	ContactNumber	Unit	RoomType	RoomView	BasePrice
1	103	Pace	58 Lundy Ave	Milpitas	6502631268	11	1BD	M	2800.00
2	107	Ashbury	98 Fafa street	Sunnyvale	6502631269	17	1BD	M	2500.00
3	112	Pace	58 Lundy Ave	Milpitas	6502631268	22	1BD	M	1110.00
4	116	Ashbury	98 Fafa street	Sunnyvale	6502631269	26	1BD	M	2800.00
5	123	Traverse	1555 Ambergrove Dr	San Jose	6502631246	35	S	N	2500.00
6	124	Traverse	1555 Ambergrove Dr	San Jose	6502631246	36	S	R	2500.00
7	125	Traverse	1555 Ambergrove Dr	San Jose	6502631246	37	S	M	2500.00
8	126	Traverse	1555 Ambergrove Dr	San Jose	6502631246	38	1BD	N	2500.00
9	127	Traverse	1555 Ambergrove Dr	San Jose	6502631246	39	S	R	2500.00
10	128	Traverse	1555 Ambergrove Dr	San Jose	6502631246	40	S	M	2500.00
11	129	Traverse	1555 Ambergrove Dr	San Jose	6502631246	41	2BD	N	2500.00
12	130	Traverse	1555 Ambergrove Dr	San Jose	6502631246	42	S	R	2500.00
13	131	Ashbury	98 Fafa street	Sunnyvale	6502631269	43	S	M	2500.00
14	132	Ashbury	98 Fafa street	Sunnyvale	6502631269	44	S	N	2500.00
15	133	Ashbury	98 Fafa street	Sunnyvale	6502631269	45	2BD	R	2500.00
16	134	Ashbury	98 Fafa street	Sunnyvale	6502631269	46	S	M	2500.00
17	135	Pace	58 Lundy Ave	Milpitas	6502631268	47	S	N	2500.00
18	136	Pace	58 Lundy Ave	Milpitas	6502631268	48	2BD	R	2500.00
19	137	Pace	58 Lundy Ave	Milpitas	6502631268	49	1BD	M	2500.00
20	138	Pace	58 Lundy Ave	Milpitas	6502631268	50	1BD	N	2500.00

Justification:

The view gives customers who would like to transfer units or help friends find available units an overview of current available units in our different communities. It includes apartment ID, community name, address, contact number, unit number, room type, room view and base price. This view provides customers a quick access to the database without confidential and unnecessary information.

View2: Available Event Detail

Query:

```
CREATE TABLE EventDetail_view
(EventID int not null,
EventName       nvarchar(50),
EventDate       datetime,
EventDescription nvarchar(100),
EventLocation   nvarchar(50),
NumberofCustomers int,
EmployeeName nvarchar(50),
EmployeePhoneNumber     char(10));

create procedure RefreshEventDetail_view as
delete from EventDetail_view
insert into EventDetail_view
Select Event.EventID, Event.EventName, Event.EventDate, Event.EventDescription, Event.EventLocation,
count(event_customer.customerid) as NumberofCustomers, Employee.EmployeeName, Employee.EmployeePhoneNumber
from Event, Work_Schedule, Work_Schedule_Community_Employee, Event_Customer, Employee
where Event.EventID = Work_Schedule.EventID and Work_Schedule.WorkScheduleID = Work_Schedule_Community_Employee.WorkScheduleID
and Event.EventID = Event_Customer.EventID and Employee.EmployeeID = Work_Schedule_Community_Employee.CEmployeeID
and Year(Event.EventDate) = Year(GETDATE()) and Month(Event.EventDate) in (Month(GETDATE()), Month(GETDATE()+1))
group by Event.EventID, Event.EventName, Event.EventDate, Event.EventDescription, Event.EventLocation,
Employee.EmployeeName, Employee.EmployeePhoneNumber
order by Event.EventDate;
```

Results | Messages

	EventID	EventName	EventDate	EventDescription	EventLocation	NumberofCustomers	EmployeeName	EmployeePhoneNumber
1	200000039	New Comer	2018-11-09 00:00:00.000	"Music, Games"	Pace	3	Huju	5630836372

Justification:

This view is for customers who would like to check how many residents will attend the events in this month and the following month, and who is the main organizer of the event. With the above information, a customer can contact the organizer for queries or different needs regarding an event.

User Group 3: Management

View1: Staff Bonus Detail from Event

Query:

```
CREATE TABLE StaffBonusDetail_view
(EmployeeID int not null,
EmployeeName nvarchar(50),
TotalHoursWorked int,
TotalAmountPaid decimal(9,2),
Year int,
Month int);

create procedure RefreshStaffBonusDetail_view as
delete from StaffBonusDetail_view
insert into StaffBonusDetail_view
Select Employee.EmployeeID, Employee.EmployeeName,
sum(datediff(hour,work_schedule.WorkStartTime,Work_Schedule.WorkEndTime)) as TotalHoursWorked,
sum(datediff(hour,work_schedule.WorkStartTime,Work_Schedule.WorkEndTime))*Community_Employee_Staff.HourlyRate as TotalAmountPaid,
year(Work_Schedule.WorkEndTime) as Year, month(Work_Schedule.WorkEndTime) as Month
from Work_Schedule_Community_Employee, Work_Schedule, Community_Employee_Staff, Employee
where Work_Schedule.WorkScheduleID = Work_Schedule_Community_Employee.WorkScheduleID
and Community_Employee_Staff.SCEmployeeID = Work_Schedule_Community_Employee.CEmployeeID
and Employee.EmployeeID=Community_Employee_Staff.SCEmployeeID
group by Employee.EmployeeID, Employee.EmployeeName, Community_Employee_Staff.HourlyRate, month(Work_Schedule.WorkEndTime),
year(Work_Schedule.WorkEndTime)
order by Year, Month, Employee.EmployeeID;
```

Results Messages

	EmployeeID	EmployeeName	TotalHoursWorked	TotalAmountPaid	Year	Month
1	100000034	Njjjj	2	24.00	2017	12
2	100000035	Jsysh	2	24.00	2017	12
3	100000036	Lkskssk	2	24.00	2017	12
4	100000037	Hstgh	2	24.00	2017	12
5	100000038	Losu	2	24.00	2017	12
6	100000039	Mksi	2	24.00	2018	1
7	100000040	UYhs	2	24.00	2018	2
8	100000041	Yhst	2	24.00	2018	2
9	100000042	Yshys	2	24.00	2018	2
10	100000043	Uyshy	2	24.00	2018	2
11	100000044	Bgay	2	24.00	2018	2
12	100000045	Puyst	2	24.00	2018	2
13	100000046	Lyesd	2	24.00	2018	3
14	100000047	Ytssg	2	24.00	2018	3
15	100000048	Vage	2	24.00	2018	3
16	100000049	Tgst	2	24.00	2018	3
17	100000050	Rtegs	2	24.00	2018	3
18	100000051	Mae	2	24.00	2018	4
19	100000052	Rgey	2	24.00	2018	4
20	100000053	Tge	2	24.00	2018	4
21	100000054	Jiye	2	24.00	2018	4
22	100000055	Pop	2	24.00	2018	5
23	100000056	Ujei	2	24.00	2018	5
24	100000057	They	2	24.00	2018	5
25	100000058	Uye	2	24.00	2018	5
26	100000059	Uye	2	24.00	2018	5
27	100000060	Mal	2	24.00	2018	6
28	100000061	Nhus	2	24.00	2018	7
29	100000062	Loke	2	24.00	2018	7
30	100000063	Loki	2	24.00	2018	7
31	100000064	Ujei	2	24.00	2018	8
32	100000065	Ytse	2	24.00	2018	8
33	100000066	Yheys	2	24.00	2018	8
34	100000067	Kue	2	24.00	2018	9
35	100000068	Loke	2	24.00	2018	9
36	100000069	Jusye	2	24.00	2018	9
37	100000070	JUye	2	24.00	2018	10
38	100000071	Kijs	2	24.00	2018	10
39	100000072	Jui	2	24.00	2018	10
40	100000073	Huju	2	24.00	2018	11

Justification:

This view allows managers to view how much should be paid to an employee associated with an event in each month. Managers could use this information to find the total amount paid to event employees and hours worked for events each month. This will help managers to stay in his budget or keep a track of employees work hours for events.

View2: Work Order Cost by Year and Month

Query:

```
CREATE TABLE PartCostDetail_view
(CommunityID int not null,
CommunityName nvarchar(50),
TotalCost int,
Year int,
Month int);

create procedure RefreshPartCostDetail_view as
delete from PartCostDetail_view
insert into PartCostDetail_view
select Community.CommunityID, Community.CommunityName, sum(consume.quantity*supplies.unitprice) as TotalCost,
year(Work_Order.CompletionDate) as Year, month(Work_Order.CompletionDate) as Month
from Community, Apartment, Work_Order, Consume, Supplies
where Community.CommunityID = Apartment.CommunityID and Apartment.ApartmentID = Work_Order.ApartmentID
and Work_Order.WorkOrderID = Consume.WorkOrderID and Consume.PartID = Supplies.PartID and Work_Order.CompletionDate is not null
group by Community.CommunityID, Community.CommunityName, month(Work_Order.CompletionDate), year(Work_Order.CompletionDate)
order by Year, Month, Community.CommunityID;
```

Results | Messages

	CommunityID	CommunityName	TotalCost	Year	Month
1	2	Ashbury	1019.70	2017	10
2	1	Pace	225.89	2018	1
3	3	Traverse	17963.90	2018	1
4	1	Pace	801.95	2018	2
5	1	Pace	2069.68	2018	3
6	3	Traverse	613.78	2018	3
7	1	Pace	320.75	2018	5
8	2	Ashbury	844.45	2018	10
9	1	Pace	2238.45	2018	11
10	2	Ashbury	1442.25	2018	11
11	3	Traverse	351.28	2018	11
12	1	Pace	87.38	2018	12
13	2	Ashbury	1603.90	2018	12
14	3	Traverse	3694.40	2018	12
15	2	Ashbury	8981.95	2019	1
16	3	Traverse	10.75	2019	1
17	3	Traverse	1220.75	2019	12

Justification:

This view gives managers a holistic view of how much is total cost of work orders each month in different communities. They should be able to pinpoint the abnormal amount in a certain month. For example, in Jan 2018, Traverse community with community ID 3 has incurred 17963.90 of the total work order cost. Comparing with total cost of other months, 17963.90 seems unacceptable. It also implies that something might go wrong in that community and managers should pay attention to the quality and lifetime of some facilities and equipment.

3.3 SQL statements for database triggers with justifications

To ensure the actions and enforce data integrity when users make changes to the data, four triggers are created. Among them, two (i.e., trigger 1 and 2) are used for the logs of information about data updates. The other two (i.e., trigger 3 and 4) are to promote consistency of use within the database.

Trigger for Logging Purpose

Trigger 1 automatically records information about base price changes when updating row/rows in the apartment table and Trigger 2 records unit price changes in the supplies table.

Trigger 1 - Base Price

```
CREATE TABLE BasePriceUpdates_Log
(ApartmentID int,
OldBasePrice decimal(9,2),
NewBasePrice decimal(9,2),
UpdateDate datetime)

create trigger  BasePriceUpdate on Apartment
for update
as
if update(BasePrice)
begin
insert into BasePriceUpdates_Log (ApartmentID, OldBasePrice, NewBasePrice, UpdateDate)
select inserted.ApartmentID, deleted.BasePrice, inserted.BasePrice, GETDATE() from inserted,
deleted where inserted.ApartmentID = deleted.ApartmentID
End

update Apartment set BasePrice = 2900 where ApartmentID = 100;
select * from BasePriceUpdates_Log
```

Results | Messages

	ApartmentID	OldBasePrice	NewBasePrice	UpdateDate
1	100	2800.00	2900.00	2018-11-24 20:14:27.540

Justification:

This trigger enables managers to detect any change in the base price of the apartment with records of apartment id, original price, updated price and update date. In the example above, apartment 100 has been increased its base price from 2800 to 2900 on 11/24/2018.

Trigger 2 - Unit Price

```
trigger 2
Loggin for UnitPrice

CREATE TABLE UnitPriceUpdates_Log
(SupplierID int,
PartID int,
OldUnitPrice decimal(9,2),
NewUnitPrice decimal(9,2),
UpdateDate datetime)

create trigger  UnitPriceUpdate on Supplies
for update
as
if update(UnitPrice)
begin
insert into UnitPriceUpdates_Log (SupplierID, PartID, OldUnitPrice, NewUnitPrice, UpdateDate)
select inserted.SupplierID, inserted.PartID, deleted.UnitPrice, inserted.UnitPrice,
GETDATE() from inserted, deleted where inserted.SupplierID = deleted.SupplierID and inserted.PartID = deleted.PartID
End

update Supplies set UnitPrice = 920 where SupplierID = 600000000 and PartID = 500000000;
select * from UnitPriceUpdates_Log
```

Results | Messages

	SupplierID	PartID	OldUnitPrice	NewUnitPrice	UpdateDate
1	600000000	500000000	898.95	920.00	2018-11-24 17:46:56.183

Justification:

This trigger indicates the change of unit price for certain part. In the example above, Part 500000000 from supplier 600000000 grows to 920 from 898.95 on 11/24/2018.

Trigger for Data Integrity

Trigger 3 enforces integrity constraints of work order table, in which the order status should be consistent with the completion date and completion date could not be earlier than start date. Trigger 4 is associated with the business rule that customer can only rent the apartment when it is available. These triggers in our system are to prevent the entry of invalid information.

Trigger 3 - Work order status, completion date

```
trigger 3
data integrity

Work order status, completion date
create trigger CompletionDateStatusCheck on Work_Order
for insert
as
declare @stu char(1), @compdate datetime, @StartDate datetime
select @stu = inserted.Status, @compdate = inserted.CompletionDate, @StartDate = inserted.StartDate from inserted
if ( @stu = 'c' and @compdate is null )
begin
print 'Please update the completion date when work order is done!'
rollback
end

if ((@stu = 'P' or @stu = 'D') and @compdate is not null)
begin
print 'Please update the status of the work order!'
rollback
end

if ( @compdate < @StartDate )
begin
print 'Please check the completion date!'
rollback
end

insert into Work_Order values (400000050, 'Floor Repair', '11/23/2018', '11/25/2018', null, 'L', 'C', 118, 100000030);
insert into Work_Order values (400000051, 'Floor Repair', '11/23/2018', '11/25/2018', '11/26/2018', 'L', 'D', 118, 100000030);
insert into Work_Order values (400000052, 'Floor Repair', '11/23/2018', '11/25/2018', '11/21/2018', 'L', 'C', 118, 100000030);
```

```
Please update the completion date when work order is completed!
Msg 3609, Level 16, State 1, Line 26
The transaction ended in the trigger. The batch has been aborted.
Please update the status of the work order!
Msg 3609, Level 16, State 1, Line 25
The transaction ended in the trigger. The batch has been aborted.
Please check the completion date!
Msg 3609, Level 16, State 1, Line 25
The transaction ended in the trigger. The batch has been aborted.
```

Justification:

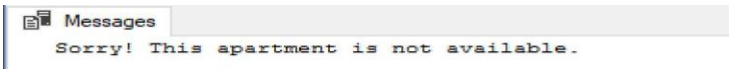
This trigger is used to detect the inconsistency between completion date and status of a work order. Three scenarios are considered and errors are treated with detailed warning messages. In the first example, the completion date is missing while the status is changed to ‘completed’. In the second one, the work is done with specific completion date while the status remains ‘delayed’. In the third one, the updated completion date is earlier than the start date of the work order.

Trigger 4 - Apartment status, customer

```
trigger 4
data integrity

Apartment status, customer
create trigger ApartmentCustomerCheck on Customer
for insert
as
declare  @apartmentid int
select @apartmentid = inserted.Apartmentid from inserted
if ( (select status from Apartment where Apartmentid = @apartmentid) = 'OC')
begin
print 'Sorry! This apartment is not available.'
rollback
end

insert into Customer values (4023, 'Tina',      1236927689,     '2/1/1993',     'tina@gmail.com', 'Engineer', 123135852, 101);
insert into Customer values (4023, 'Tina',      1236927689,     '2/1/1993',     'tina@gmail.com', 'Engineer', 123135852, 138);
select * from Customer;
```



Justification:

This trigger prevents a customer to rent an apartment which is not available (i.e., status = 'OC'). As the example above, if Tina rents apartment 101, it will raise an warning "Sorry! This apartment is not available".

3.4 SQL statements for procedures with justifications

Procedure 1 - Discount Unit Price for Parts

```
Alter Table supplies Add DiscountUnitPrice float
Create Procedure PartDiscountUnitPrice as
Update Supplies set DiscountUnitPrice = .9*UnitPrice where PartID=500000001
Update Supplies set DiscountUnitPrice = .7*UnitPrice where PartID=500000002
Update Supplies set DiscountUnitPrice = .6*UnitPrice where PartID=500000003
Update Supplies set DiscountUnitPrice = .5*UnitPrice where PartID=500000004
Update Supplies set DiscountUnitPrice = .4*UnitPrice where PartID=500000005
Update Supplies set DiscountUnitPrice = .3*UnitPrice where PartID=500000006
Execute PartDiscountUnitPrice
Select* from Supplies
```

	SupplierID	PartID	UnitPrice	DiscountUnitPrice
1	600000000	500000000	898.95	NULL
2	600000001	500000001	120.75	108.675
3	600000002	500000002	25.89	18.123
4	600000003	500000003	8981.95	5389.17
5	600000004	500000004	1220.75	610.375
6	600000005	500000005	5.89	2.356
7	600000006	500000006	88.95	26.685
8	600000007	500000007	10.75	NULL
9	600000008	500000008	225.89	NULL
10	600000009	500000009	801.95	NULL
11	600000010	500000010	110.75	NULL
12	600000011	500000011	54.89	NULL
13	600000012	500000012	818.95	NULL
14	600000013	500000013	1220.75	NULL
15	600000014	500000014	52.89	NULL
16	600000015	500000015	981.95	NULL
17	600000016	500000016	320.75	NULL
18	600000017	500000017	562.00	NULL
19	600000018	500000018	18.80	NULL
20	600000019	500000019	12.75	NULL
21	600000020	500000020	75.60	NULL

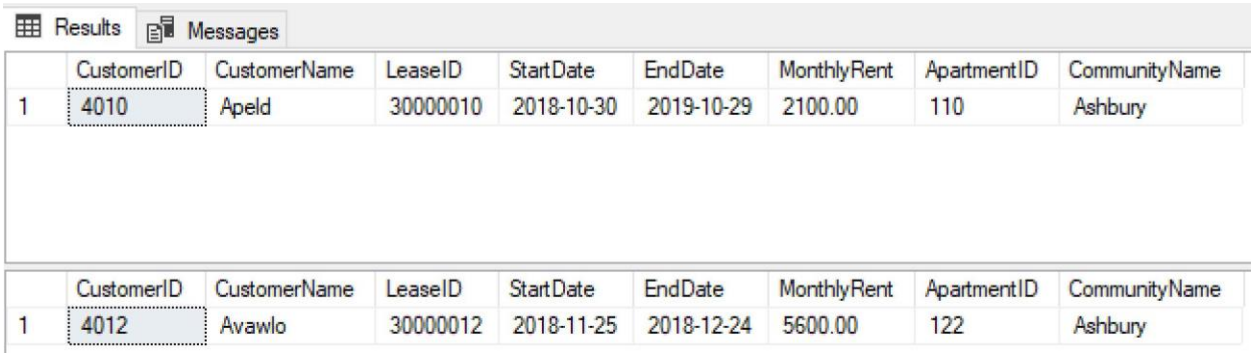
Justification:

This procedure allows faster execution without changing the original data. By adding PartDiscountUnitPrice procedure, users can quickly update base price information in a new column.

Procedure 2 - Retrieve Lease Details

```
Create procedure retriveleasedetails
@leaseid int as
Select Customer.CustomerID,Customer.CustomerName,
lease.LeaseID,lease.StartDate,Lease.EndDate,Lease.MonthlyRent,
Apartment.ApartmentID,Community.CommunityName
from Customer,Lease,Apartment,Community
where Customer.ApartmentID=Lease.ApartmentID and
Lease.ApartmentID=Apartment.ApartmentID
and Apartment.CommunityID=Community.CommunityID and Lease.LeaseID=@leaseid

execute retriveleasedetails 30000010
execute retriveleasedetails 30000012
```



Results Messages

	CustomerID	CustomerName	LeaseID	StartDate	EndDate	MonthlyRent	ApartmentID	CommunityName
1	4010	Apeld	30000010	2018-10-30	2019-10-29	2100.00	110	Ashbury

	CustomerID	CustomerName	LeaseID	StartDate	EndDate	MonthlyRent	ApartmentID	CommunityName
1	4012	Avawlo	30000012	2018-11-25	2018-12-24	5600.00	122	Ashbury

Justification:

This procedure allows faster execution and reduces network traffic without changing the original data. By adding retriveleasedetails, users can quickly access information that are useful to them and save a lot of time by avoiding unnecessary message; therefore，it reduces network traffic and processing workload.