TRADITI PROPERTY MANAGEMENT DATABASE

Developed by Brandon Traditi

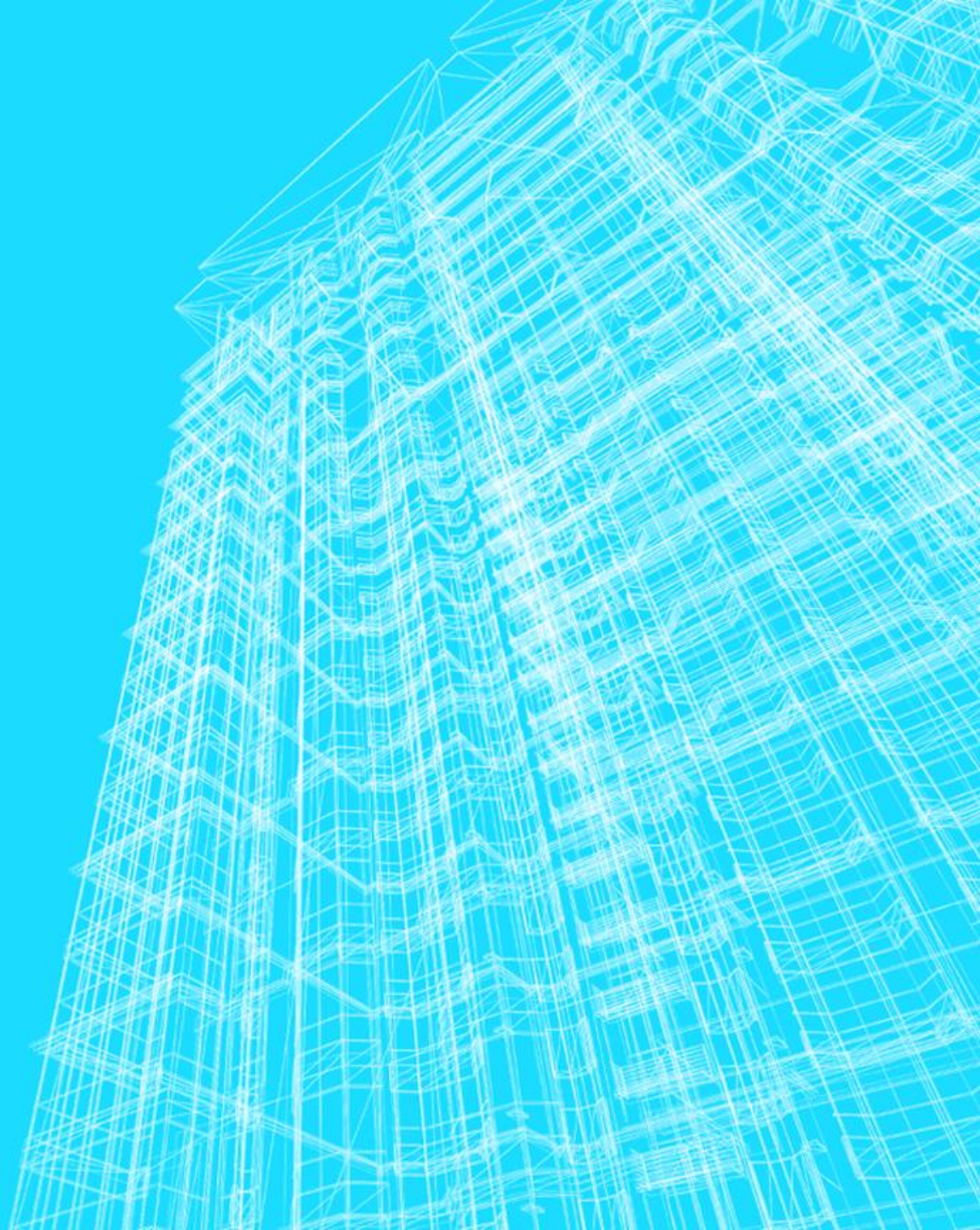
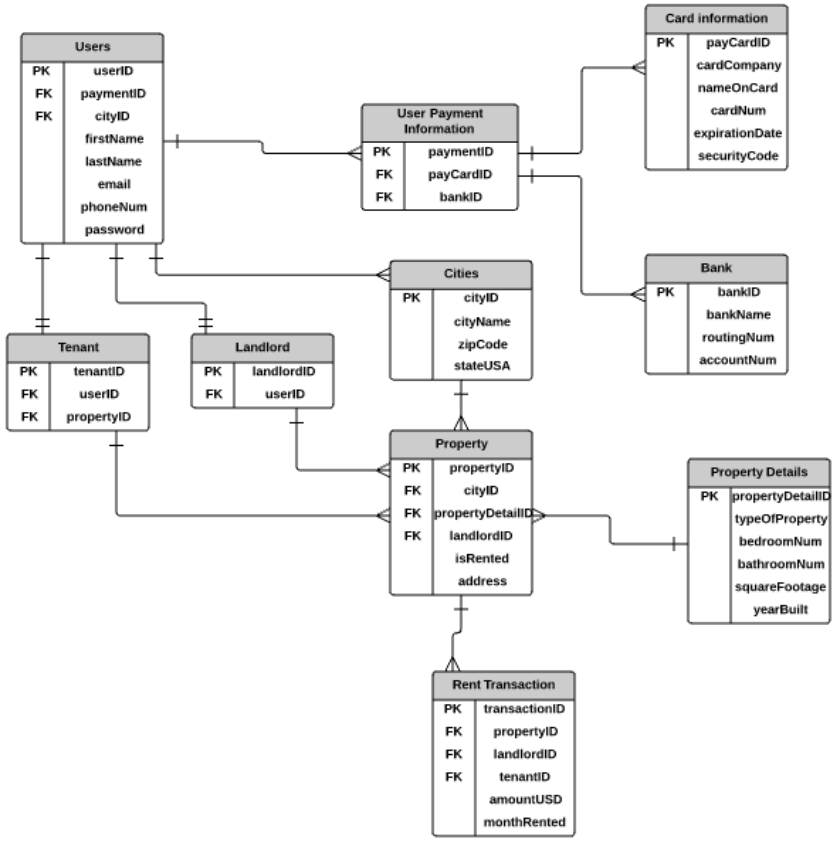


TABLE OF CONTENTS

EXECUTIVE SUMMARY

This document takes us through the database of a software for Traditi Property Management, a mock property management company. Traditi's company owns 14 properties, has 22 tenants, 3 landlords, over the span of 5 cities. This database mocks the backend to a software that Traditi's would use to manage the properties and interactions between tenants and landlords such as paying rent. With eagerness to purchase more properties, the database must be implemented to grow with multiple homes, tenants, and financial information. Even though the current data implemented into the database is fictional, the implementation is as if it were real. The objective of this database is to represent a real database for a property management company in third normal form.

E/R DIAGRAM


Users
PK userID
FK paymentID
FK cityID
firstName
lastName
email
phoneNum
password
User Payment Information
PK paymentID
FK payCardID
FK bankID
Card information
PK payCardID
cardCompany
nameOnCard
cardNum
expirationDate
securityCode
Bank
PK bankID
bankName
routingNum
accountNum
Cities
PK cityID
cityName
zipCode
stateUSA
Tenant
PK tenantID
FK userID
FK propertyID
Landlord
PK landlordID
FK userID
Property
PK propertyID
FK cityID
FK propertyDetailID
FK landlordID
isRented
address
Property Details
PK propertyDetailID
typeOfProperty
bedroomNum
bathroomNum
squareFootage
yearBuilt
Rent Transaction
PK transactionID
FK propertyID
FK landlordID
FK tenantID
amountUSD
monthRented

BANK TABLE

This table represents bank information entered by users that will be further referenced in the Users Payment Info table.

```
CREATE TABLE Bank (
    bankID char(6) not null,
    bankName text not null,
    routingNum int not null,
    accountNum int not null,
primary key(bankID)
);
```

Functional Dependences:
bankID ➡ bankName, routingNum, accountNum

Sample Data from Bank Table

	bankid character(6)	bankname text	routingnum integer	accountnum integer
1	bk001	Chase	198567575	100015889
2	bk002	Bank of America	156342008	300115581
3	bk003	Morgan Stanley	541288639	512544013
4	bk004	TD Bank	900310255	653295671
5	bk005	Capital One	850006732	123854401
6	bk006	Wells Fargo	650011255	320115586
7	bk007	Citigroup	990051236	455336201
8	bk008	Wells Fargo	650011300	653526301
9	bk009	Bank of America	156342960	543600189
10	bk010	Chase	198567200	638894521
11	bk011	TD Bank	900310555	639945231
12	bk012	Capital One	850006744	152239875
13	bk013	Chase	198567330	442158756
14	bk014	Morgan Stanley	541288452	542269785
15	bk015	Bank of America	156342100	302456112

CARD INFORMATION TABLE

This table represents credit card information entered by users that will be further referenced in the Users Payment Info table.

```
CREATE TABLE CardInformation (
    payCardID char(6) not null,
    cardCompany text not null,
    nameOnCard text not null,
    cardNum varchar(16) not null,
    expirationDate varchar(5) not null,
    securityCode int not null,
primary key(payCardID)
);
```

Functional Dependences:
payCardID ➡ cardCompany, nameOnCard, cardNum, expirationDate, securityCode

Sample Data from Card Information Table

	paycardid character(6)	cardcompany text	nameoncard text	cardnum character varying(16)	expirationdate character varying(5)	securitycode integer
1	pc001	Visa	Brandon Traditi	5245321065236895	07/19	131
2	pc002	Master Card	Allison McBride	7545215463120235	06/20	125
3	pc003	Discover	Frank Sills	7542865430215569	01/20	553
4	pc004	Visa	Adam West	8532412000365698	12/22	856
5	pc005	American Express	Jill Neal	4521963256859944	11/20	654
6	pc006	Discover	Lee Wi	1203458755623022	02/22	525
7	pc007	American Express	Keith Amerson	1523625895420345	05/21	778
8	pc008	Visa	George Anderson	4523698745201569	06/20	458
9	pc009	Master Card	Victoria Jackson	6325785512036958	12/19	856
10	pc010	Visa	Mary Hillbert	4525566323559863	09/18	320
11	pc011	Master Card	Donte Jones	023536987996510	06/20	111
12	pc012	Discover	Peter Griffin	4563022159875632	04/20	152
13	pc013	American Express	Courtney Killigan	9635458632158599	01/22	632
14	pc014	American Express	Fred Slate	4302125568663201	10/20	420
15	pc015	Visa	Alan Labouseur	7598653211202300	12/22	667

USERS PAYMENT INFO TABLE

This table represents payment ID's that each user will reference and each user may have a bank, card, or both information stored.

```
CREATE TABLE UsersPaymentInfo (
    paymentID char(6) not null,
    payCardID char(6) references CardInformation(payCardID),
    bankID char(6) references Bank(bankID),
primary key(paymentID)
);
```

Functional Dependences:
paymentID ➡ payCardID, bankID

Sample Data from Users Payment Info Table

	paymentid character(6)	paycardid character(6)	bankid character(6)
1	pid001	pc001	bk001
2	pid002		bk002
3	pid003	pc002	
4	pid004	pc003	bk003
5	pid005		bk004
6	pid006	pc004	
7	pid007	pc005	bk005
8	pid008	pc006	
9	pid009		bk006
10	pid010		bk007
11	pid011		bk008
12	pid012	pc007	
13	pid013		bk009
14	pid014	pc008	
15	pid015	pc009	
16	pid016	pc010	
17	pid017		bk010
18	pid018	pc011	bk011
19	pid019	pc012	
20	pid020		bk012

CITIES TABLE

This table represents all of the Cities that Traditi Property Management owns and operates properties in.

```
CREATE TABLE Cities(
    cityID char(6) not null,
    cityName text not null,
    zipCode int not null,
    stateUSA text not null,
primary key(cityID)
);
```

Functional Dependences:
cityID ➡ cityName, zipCode, stateUSA

Sample Data from Cities Table

	cityid character(6)	cityname text	zipcode integer	stateusa text
1	ct0010	Poughkeepsie	12601	New York
2	ct0001	New York	10001	New York
3	ct0100	Los Angeles	90001	California
4	ct0320	Miami	33124	Florida
5	ct0241	Honolulu	96801	Hawaii

USERS TABLE

This table represents all of the users and their basic information to create a distinct variable for each user.

```
CREATE TABLE Users (
   userID char(6) not null,
   paymentID char(6) not null references UsersPaymentInfo(paymentID),
   cityID char(6) not null references Cities(cityID),
   firstName text not null,
   lastName text not null,
   gender char(1) not null,
   email text not null,
   phoneNum varchar(10),
   password varchar(25) not null,
CONSTRAINT CHK_gender CHECK (gender = 'M' OR gender = 'F'),
primary key(userID)
);
```

Functional Dependences:
userID ➡ paymentID, cityID, firstName, lastName, gender, email, password

Check Constraint: User may only put M(male) or F(female) in gender category.

Sample Data from Users Table

	userid character(6)	paymentid character(6)	cityid character(6)	firstname text	lastname text	gender character(1)	email text	phonenum character varying(10)	password character varying(25)
1	uid001	pid001	ct0001	Brandon	Traditi	M	BrandonTraditi@gmail.com	8452263312	HeyItsMe101
2	uid002	pid002	ct0320	Julio	Johnson	M	Julio.Johnson@yahoo.com	9145563214	footballlover22
3	uid003	pid003	ct0241	Allison	McBride	F	McBride21@yahoo.com	2105633852	ddk211
4	uid004	pid004	ct0001	Frank	Sills	M	Frank.Sills10@hotmail.com	5421036645	bouldershoulder2
5	uid005	pid005	ct0320	Jessica	Linguini	F	LinguiniLover11@gmail.com	7523694512	Pastamaker22
6	uid006	pid006	ct0241	Adam	West	M	Adam.West100@gmail.com	1802301514	TownMayor123
7	uid007	pid007	ct0001	Jill	Neal	F	J.Neal1@hotmail.com	4102516485	InsertDogName111
8	uid008	pid008	ct0100	Lee	Wi	M	Lee.Wi@yahoo.com	9145623152	Mr.Fantastic101
9	uid009	pid009	ct0001	Derick	Wall	M	D.Wall@gmail.com	7501235208	Yankeesfan15
10	uid010	pid010	ct0100	Austin	Java	M	Javaman@aol.com	8501567542	coderForLife123
11	uid011	pid011	ct0010	Gale	Keller	F	GKell@hotmail.com	5186423301	Jamamaam56
12	uid012	pid012	ct0010	Keith	Amerson	M	KeithAmerson@gmail.com	7520561234	Uhj2kl2
13	uid013	pid013	ct0320	Paul	Mack	M	MackAttack@hotmail.com	4512236695	TakeOne1
14	uid014	pid014	ct0320	George	Anderson	M	George.Anderson1@yahoo.com	4506642152	HillsBills201
15	uid015	pid015	ct0100	Victoria	Jackson	F	JacksonFive@aol.com	4501251145	LikeMike5
16	uid016	pid016	ct0241	Mary	Hillbert	F	M.Hillbert@gmail.com	8025421123	JaxHil123
17	uid017	pid017	ct0001	Jerry	Hicks	M	Hicks.Jerry@aol.com	1502203062	Madmax560
18	uid018	pid018	ct0001	Donte	Jones	M	Donte.Jones45@gmail.com	8452562231	NFLbound350
19	uid019	pid019	ct0001	Peter	Griffin	M	email@gmail.com	5423026525	password
20	uid020	pid020	ct0100	Harry	Deforest	M	Deforest50@yahoo.com	450256623	JambaJuice82

PROPERTY DETAIL TABLE

This table represents the four different types of properties that Traditi Property Management own and creates an ID for each.

```
CREATE TABLE PropertyDetails(
   propertyDetailID char(6) not null,
   typeOfProperty text not null,
   bedroomNum varchar(2),
   bathroomNum varchar(2),
   squareFootage int,
   yearBuilt varchar(4),
CONSTRAINT CHK_type CHECK (typeOfProperty = 'Apartment' OR typeOfProperty = 'House'),
primary key(propertyDetailID)
);
```

Sample Data from Property Detail Table

	propertydetailid character(6)	typeofproperty text	bedroomnum character varying(2)	bathroomnum character varying(2)	squarefootage integer	yearbuilt character varying(4)
1	pdid01	Apartment	1	1	1000	2000
2	pdid02	Apartment	2	1	1500	2005
3	pdid03	House	3	2	3000	1990
4	pdid04	House	4	2	4000	2010

Functional Dependences:
propertyDetailID ➡ typeOfProperty

Check Constraint: The type of property can only be an Apartment or a House.

LANDLORD TABLE

This table represents all of the landlordID's and the landlords corresponding userID.

```
CREATE TABLE Landlord(
    landlordID char(6) not null,
    userID char(6) not null references Users(userID),
primary key(landlordID)
);
```

Functional Dependences:
landlordID ➡ userID

Sample Data from Landlord Table

	landlordid character(6)	userid character(6)
1	LLid01	uid001
2	LLid02	uid010
3	LLid03	uid025

PROPERTY TABLE

This table represents all of the properties and the corresponding details entailed in each.

```
CREATE TABLE Property(
    propertyID char(6) not null,
    cityID char(6) not null references Cities(cityID),
    propertyDetailID char(6) not null references PropertyDetails(propertyDetailID),
    landlordID char(6) not null references Landlord(landlordID),
    isRented boolean not null,
    address text not null,
primary key(propertyID)
);
```

Functional Dependences:

propertyID → cityID, propertyDetailID, landlordID, isRented, address

Sample Data from Property Table

	propertyid character(6)	cityid character(6)	propertydetailid character(6)	landlordid character(6)	isrented boolean	address text
1	prop01	ct0010	pdid02	LLid03	t	39 Inwood Avenue
2	prop02	ct0010	pdid02	LLid03	t	63 Sunset Avenue
3	prop03	ct0010	pdid01	LLid03	f	5 West Cedar
4	prop04	ct0001	pdid01	LLid01	t	201 Madison Avenue
5	prop05	ct0001	pdid01	LLid01	t	10 West Lane
6	prop06	ct0001	pdid02	LLid01	t	300 Wall Street
7	prop07	ct0001	pdid02	LLid01	t	250 Park Avenue
8	prop08	ct0001	pdid02	LLid01	t	300 East Lane
9	prop09	ct0100	pdid02	LLid02	t	361 Beverly Bloulevard
10	prop10	ct0100	pdid01	LLid02	t	42 Oak Drive
11	prop11	ct0100	pdid02	LLid02	f	1 Hills Lane
12	prop12	ct0320	pdid04	LLid03	t	100 Club Lane
13	prop13	ct0241	pdid03	LLid01	t	11 Sunny Lane
14	prop14	ct0241	pdid04	LLid01	f	25 Mountain Drive

TENANT TABLE

This table represents all of the tenantID's, their corresponding userID, and the property in which they reside in.

```
CREATE TABLE Tenant(
    tenantID char(6) not null,
    userID char(6) not null references Users(userID),
    propertyID char(6) not null references Property(propertyID),
primary key(tenantID)
);
```

Functional Dependences:
tenantID ➡ userID, propertyID

Sample Data from Tenant Table

	tenantid character(6)	userid character(6)	propertyid character(6)
1	Tid001	uid002	prop04
2	Tid002	uid003	prop13
3	Tid003	uid004	prop07
4	Tid004	uid005	prop04
5	Tid005	uid006	prop13
6	Tid006	uid007	prop08
7	Tid007	uid008	prop09
8	Tid008	uid009	prop05
9	Tid009	uid011	prop02
10	Tid010	uid012	prop01
11	Tid011	uid013	prop12
12	Tid012	uid014	prop12
13	Tid013	uid015	prop10
14	Tid014	uid016	prop13
15	Tid015	uid017	prop06
16	Tid016	uid018	prop07
17	Tid017	uid019	prop08
18	Tid018	uid020	prop09
19	Tid019	uid021	prop04
20	Tid020	uid022	prop01

RENT TRANSACTION TABLE

This table represents a few different rent payment transaction between the landlord and tenant.

```
CREATE TABLE RentTransaction(
    transactionID char(6) not null,
    propertyID char(6) not null references Property(propertyID),
    landlordID char(6) not null references Landlord(landlordID),
    tenantID char(6) not null references Tenant(tenantID),
    amountUSD int not null,
    monthRented char(9) not null,
primary key(transactionID)
);
```

Functional Dependences:
transactionID ➡ propertyID, landlordID, tenantID, amountUSD, monthRented

Sample Data from Rent Transaction Table

	transactionid character(6)	propertyid character(6)	landlordid character(6)	tenantid character(6)	amountusd integer	monthrented character(9)
1	Tran01	prop01	LLid03	Tid020	1000	March
2	Tran02	prop04	LLid01	Tid019	1250	September
3	Tran03	prop07	LLid01	Tid016	750	June
4	Tran04	prop10	LLid02	Tid013	900	November
5	Tran05	prop12	LLid03	Tid001	1500	June
6	Tran06	prop13	LLid01	Tid014	2000	July
7	Tran07	prop06	LLid01	Tid021	1000	January
8	Tran08	prop02	LLid03	Tid022	650	Febuary
9	Tran09	prop09	LLid02	Tid018	1200	October
10	Tran10	prop08	LLid01	Tid017	1200	March

STORED PROCEDURE UPDATE_PROPERTY_RENTED()

This stored procedure updates a home from not being rented to being rented when a transaction is made.

```
CREATE OR REPLACE FUNCTION update_property_rented()
RETURNS TRIGGER AS
$$
BEGIN
IF NEW.transactionID is NOT NULL THEN
UPDATE Property
SET isRented = TRUE
WHERE NEW.propertyID = Property.propertyID;
END IF;
RETURN NEW;
END;
$$
LANGUAGE PLPGSQL;
```

```
SELECT*
FROM property
Where landlordID = 'LLid01'
```

Before Transaction

	propertyid character(6)	cityid character(6)	propertydetailid character(6)	landlordid character(6)	isrented boolean	address text
1	prop04	ct0001	pdid01	LLid01	t	201 Madison Avenue
2	prop05	ct0001	pdid01	LLid01	t	10 West Lane
3	prop06	ct0001	pdid02	LLid01	t	300 Wall Street
4	prop07	ct0001	pdid02	LLid01	t	250 Park Avenue
5	prop08	ct0001	pdid02	LLid01	t	300 East Lane
6	prop13	ct0241	pdid03	LLid01	t	11 Sunny Lane
7	prop14	ct0241	pdid04	LLid01	f	25 Mountain Drive

After Transaction

	propertyid character(6)	cityid character(6)	propertydetailid character(6)	landlordid character(6)	isrented boolean	address text
1	prop04	ct0001	pdid01	LLid01	t	201 Madison Avenue
2	prop05	ct0001	pdid01	LLid01	t	10 West Lane
3	prop06	ct0001	pdid02	LLid01	t	300 Wall Street
4	prop07	ct0001	pdid02	LLid01	t	250 Park Avenue
5	prop08	ct0001	pdid02	LLid01	t	300 East Lane
6	prop13	ct0241	pdid03	LLid01	t	11 Sunny Lane
7	prop14	ct0241	pdid04	LLid01	t	25 Mountain Drive

```
INSERT INTO RentTransaction(transactionID, propertyID, landlordID, tenantID, amountUSD, monthRented)
Values('Tran11','prop14','LLid01','Tid017',1200,'March');
```

TRIGGER UPDATE_PROPERTY_TRIGGER()

This trigger links the stored procedure in the previous slide and triggers it whenever an insert into the Rent Transaction table is made

```
CREATE TRIGGER update_property_trigger
BEFORE INSERT ON RentTransaction
FOR EACH ROW
EXECUTE PROCEDURE update_property_rented();
```

VIEW AVAILABLEPROPERTIES

This view shows a table of available properties that a landlord owns.

```
DROP VIEW IF EXISTS availableProperties;
CREATE VIEW availableProperties as (
SELECT p.propertyID,
          p.landlordID
   FROM Property p INNER JOIN Landlord l
      ON p.landlordID = l.landlordID
   WHERE p.isRented = FALSE
);
```

Query Executed:

```
SELECT*
FROM availableProperties
WHERE landlordID = 'LLid02'
```

Sample Data from View

	propertyid character(6)	landlordid character(6)
1	prop11	LLid02

VIEW LANDLORDCITIES

This view shows a table of the cities that a particular landlord owns property in.

```
DROP VIEW IF EXISTS landlordCities;
CREATE VIEW landlordCities as (
SELECT c.cityName,
       p.landlordID
   FROM Property p INNER JOIN Cities c
       ON p.cityID = c.cityID
);
```

Query Executed:

```
SELECT*
FROM landlordCities
WHERE landlordID = 'LLid01'
```

Sample Data from View

	cityname text	landlordid character(6)
1	New York	LLid01
2	New York	LLid01
3	New York	LLid01
4	New York	LLid01
5	New York	LLid01
6	Honolulu	LLid01
7	Honolulu	LLid01

QUERY 1

This query returns the total revenue for LLid01 from the Rent Transaction table as totRevenue. This is useful for the landlord to know the amount of revenue they received in a given month

```
SELECT SUM(amountUSD) AS totRevenue
FROM rentTransaction
WHERE landlordID = 'LLid01' AND monthRented = 'March'
```

Sample Data from Query

	totrevenue bigint
1	1200

QUERY 2

This query returns each property, the tenants that reside in it and the landlord for that home. This is useful for reports to know which tenants belong to which property and landlord.

```
SELECT t.propertyID, t.tenantID, p.landlordID
FROM Property p INNER JOIN tenant t
        ON p.propertyID = t.propertyID
ORDER BY propertyID
```

Sample Data from Query

	propertyid character(6)	tenantid character(6)	landlordid character(6)
1	prop01	Tid010	LLid03
2	prop01	Tid020	LLid03
3	prop02	Tid009	LLid03
4	prop02	Tid022	LLid03
5	prop04	Tid001	LLid01
6	prop04	Tid004	LLid01
7	prop04	Tid019	LLid01
8	prop05	Tid008	LLid01
9	prop06	Tid015	LLid01
10	prop06	Tid021	LLid01
11	prop07	Tid016	LLid01
12	prop07	Tid003	LLid01
13	prop08	Tid006	LLid01
14	prop08	Tid017	LLid01
15	prop09	Tid007	LLid02
16	prop09	Tid018	LLid02
17	prop10	Tid013	LLid02
18	prop12	Tid011	LLid03
19	prop12	Tid012	LLid03
20	prop13	Tid002	LLid01
21	prop13	Tid014	LLid01
22	prop13	Tid005	LLid01

QUERY 3

This query returns each properties address that landlord01 owns. This is useful to know the amount of properties and where the properties are that the landlord own.

```
SELECT l.landlordID,
       p.address
   FROM Landlord l INNER JOIN Property p
      ON p.landlordID = l.landlordID
      WHERE p.landlordID = 'LLid01'
```

Sample Data from Query

	landlordid character(6)	address text
1	LLid01	201 Madison Avenue
2	LLid01	10 West Lane
3	LLid01	300 Wall Street
4	LLid01	250 Park Avenue
5	LLid01	300 East Lane
6	LLid01	11 Sunny Lane
7	LLid01	25 Mountain Drive

SECURITY

Security will Grant the ADMIN all access to manage the database as needed while the Users will have access to change their information along with payment info

```
DROP ROLE IF EXISTS ADMIN;
DROP ROLE IF EXISTS USERS;

CREATE ROLE ADMIN;
GRANT ALL ON ALL TABLES IN SCHEMA PUBLIC TO ADMIN;

CREATE ROLE USERS;
REVOKE ALL ON ALL TABLES IN SCHEMA PUBLIC FROM USERS;
GRANT INSERT ON Users, UsersPaymentInfo, cardInformation, Bank TO USERS;
GRANT UPDATE ON Users, UsersPaymentInfo, cardInformation, Bank TO USERS;
```

IMPLEMENTATION - KNOWN PROBLEMS - FUTURE ENHANCEMENTS

Implementation

- The larger the company grows, the more the software will have to keep up with the addition of identities of users tenants, landlords, cities, etc.
- Some type of encryption could be implemented to keep the bank and card information of the users more safe and secure

Known Problems

- More than one store procedure should be in place to make querying easier.
- Much more tables would have to be implemented to truly capture the potential of this software

Future Enhancements

- Dive deeper into the aspects of property management and have the database contain many more aspects Ex) Files, financial reports, cash flow analysis