|  |  |
| --- | --- |
| Class: | 3432 Database Systems |
| Professor: | **Dr. J.** |
| Term / Year: | **Fall 2018** |
| Effort [minutes] | **16 weeks – roughly 2-8 hrs a week** |
| Student Name (s) | **Tyler Hodzen, Troy Purvis, Jenna Lovett** |
| Assignment: | **Final Project** |
| Date of Submission: | **November 30th, 2018** |

**Jenna Lovett**

Your satisfaction with learning-experience with this assignment (please check one):

* A. Very Useful

🞎 B. Somewhat Useful

🞎 C. Useless

**Post Mortem**

My greatest experience with this project was with data modeling. In class, we learned that the data model will be the make-or-break point for any database, and I had experience with this during our project. During the prototype stage, our data model was not quite correct (see the Data Model version 1 below). When inserting, deleting, and updating data within the database, we ran into errors. Data was not being inserted properly because the relationships were not properly established. I also enjoyed making the model the most out of everything we did for this project. Understanding how entity relationships work, including the RI, was a new concept that I enjoyed learning.

Another experience that I learned from this project is the importance of testing. Choosing the correct queries to test the data within the database is very important. Querying is a multi-use tool. It allows the users to test for data integrity and data modeling correctness.

**Release Note:**

**We**, the undersigned, take full responsibility for the work performed related to this assignment and for the work records attached, and also identify both as **our** own:

Signature 1: Tyler Hodzen

*Signature 2:* Troy Purvis

*Signature 3:* Jenna Lovett

**Executive Summary**

This report contains every facet of information related to the CSCI 3432 Database Systems semester-long project. In this report, you will find project scope documentation, including the roles delegated to all team members, a project timeline, details regarding the creation of the database, which technologies were used in the stack, details regarding the creation of the UI, and information about the database model itself. Other parts of the report include email correspondence between the customer advocate (Jenna Lovett) and the customer (Dr. J), screenshots of various application/database creations, so on and so forth. This report summarizes and expands on all workings of the project – from the beginning to the end.

To summarize the goal for our project, we decided to base our entire project on a fictional apartment complex – Pink Paradise Plateau Apartments. The application (UI) is meant to act as a portal for the apartment complex administrators to be able to manage data related to the complex. The application is not meant to be handled by low-level staff or anyone not employed by Pink Paradise Plateau Apartments.

Firstly, this report begins by introducing the project goals, narrative, and team members. Jenna Lovett is the team leader and customer advocate. Tyler Hodzen is the QA/Security leader. Troy Purvis is the DBA.

Next, the report delves into the project timeline. The timeline has bullet points listing which tasks need to be completed and when. Once a task is complete, the bullet point is changed to a checkmark to signify finalization.

Afterwards, the report shows proof of customer and advocate interaction through email.

Next, the report shows a complete listing the script used to create the database. The script is written in pure SQL, and the method used to create the database was data modeling. The project team hand-wrote the entirety of the script based off the data model (including RI). Then the team filled in test data with pure SQL. After this, certain views were created that would aid in the search functionality of the application. The views are meant to provide quick snapshots of related information, while not overwhelming the end user.

Next, the report details all versions of the project data model. Firstly, the project team decided to drop the tables that were associated with team members who dropped the course. However, after speaking with the customer advocate, the team decided to re-add necessary tables.

Next, the report delves into the 10 queries per team member. The queries are segmented into smaller groupings (based off of the data model) and delegated to each team member.

Next, the report shows the evolution from the Mock UI stage of the project to the Final UI stage of the project. These two sections of the report shows the gradual and major changes associated with the changing of the project tables, as well as team responsibilities.

**Project Scope Documentation**

|  |  |
| --- | --- |
| Project Name | CSCI 3432 Group Project – Pink Paradise Plateau Apartments |
| Project Team | * Team Leader and Customer Advocate: Jenna Lovett * DB Application Developers: ~~Kirk Alexander~~ and Troy Purvis * DBA, Server and Configuration Management: ~~Dallas Martin~~ * QA/Test, Security, and Tech Support: Tyler Hodzen |
| Project Narrative | We will be creating a mock database for an apartment complex. This data base will contain dummy data sourced from online repositories. This database will be devised of relatable data, including resident, vehicle, employee, ~~maintenance~~, invoice, ~~department~~, event, pet, lease, and apartment unit information. Each team member will be responsible for gathering sample data throughout the project, building the database from the ground up (see page below). Each member will have a specific role (as listed above) but will also gain experience by sharing roles with all team members. |
| Project Requirements | 1. Create a GUI Mockup 2. Perform a fitness test (assuming model has sufficient info)    1. Trace each question on the model    2. Add more information on second iteration    3. Trace all transactions and select critical one or two 3. Live Tests    1. Prepare a small set of test data    2. Run the SQL queries on the project deliverables    3. Prepare additional exploratory questions and experiment    4. Enrich the model on third iteration 4. Test and Improve    1. Generate volume tests    2. Conduct performance testing    3. Denormalize if necessary on fourth iteration 5. Develop app to update DB    1. Conduct usability testing    2. Specify improvements for next release on fifth iteration |

|  |  |
| --- | --- |
| Team Member Name | Roles & Responsibilities |
| Jenna Lovett | Responsible for creating Residents table and Pets table, including columns within both tables. Responsible for gathering sample data pertaining to residents (ie. First names, Last names, apartment numbers, etc.). Responsible for testing small sets of data from these tables to ensure proper functionality. Responsible for ensuring data from Residents and Pets tables relates to other tables (ie. Resident\_ID from Residents table works with Resident\_ID(fk) from Vehicles table). Responsible for reviewing relevant tables (Residents and Pets) on the DB model/schema. Responsible for ensuring each team member completes their required duties and following up with Dr. J on a regular basis. Responsible for reviewing and curating project documentation, such as scope. Responsible for building his/her portion of the GUI that relates to updating his/her tables. Responsible for creating 10 queries/questions. |
| Tyler Hodzen | Responsible for creating Employees table and Maintenance table, including columns within both tables. Responsible for gathering sample data pertaining to employees/maintenance (ie. First names, Last names, department numbers, etc.). Responsible for testing small sets of data from these tables to ensure proper functionality. Responsible for ensuring data from Employees and Maintenance tables relates to other tables. Responsible for reviewing relevant tables (Employees and Maintenance) on the DB model/schema. Responsible for assisting each team member with security questions and technical support. Responsible for building his/her portion of the GUI that relates to updating his/her tables. Responsible for creating 10 queries/questions. |
| ~~Dallas Martin~~ | ~~Responsible for creating Invoices table and Departments table, including columns within both tables. Responsible for gathering sample data pertaining to invoices/departments (ie. invoiceamount, payment\_date, etc.). Responsible for testing small sets of data from these tables to ensure proper functionality. Responsible for ensuring data from Invoices and Departments tables relates to other tables. Responsible for reviewing relevant tables (Invoices and Departments) on the DB model/schema. Responsible for assisting each team member with server configuration and management. Responsible for building his/her portion of the GUI that relates to updating his/her tables. Responsible for creating 10 queries/questions.~~ |
| Troy Purvis | Responsible for creating Vehicles table and Events table, including columns within both tables. Responsible for gathering sample data pertaining to vehicles/events (ie. License\_plate, event\_date, etc.). Responsible for testing small sets of data from these tables to ensure proper functionality. Responsible for ensuring data assigned tables relates to other tables. Responsible for reviewing relevant tables (Events and Vehicles) on the DB model/schema. Responsible for assisting each team member with application development, particularly backend. Responsible for building his/her portion of the GUI that relates to updating his/her tables. Responsible for creating 10 queries/questions. |
| ~~Kirk Alexander~~ | ~~Responsible for creating Leases table and Apartment Units table, including columns within both tables. Responsible for gathering sample data pertaining to leases/units (ie. unit\_number, sign\_date, etc.). Responsible for testing small sets of data from these tables to ensure proper functionality. Responsible for ensuring data assigned tables relates to other tables. Responsible for reviewing relevant tables (leases and units) on the DB model/schema. Responsible for assisting each team member with application development. Responsible for building his/her portion of the GUI that relates to updating his/her tables. Responsible for creating 10 queries/questions.~~ |
| UPDATED 10/31 | Kirk and Dallas dropped the course. However, we still need to cover essential tables that were assigned to these two group members. These tables include the Apartments table, which is the defining table for this project. |

**Project Timeline**

Week 2&3

* Finalized group/team with corresponding team members

Week 4

* Drafted and submitted scope statement, which includes project narrative and member responsibilities

Week 5

* Drafted and submitted 10 queries per member and approved by Dr. J

Week 6

* Created GUI MockUps for Residents, Vehicles, and Employees tables

Week 9

* Created and submitted prototypes (POCs) that were connected to our database using Visual Studio/C#/SQL Server

Week 10

* Created a draft data model with 6 tables

Week 11

* Reviewed initial draft of data model in IDEF1X and created a detailed logical model with Entities, Relationships, Attributes in Erwin

Week 12

* Created new database script based off of Week 11’s progress
* Filled with test data
* Testing with RI

Week 13

* Loaded initial test DB into SQL Server and experimented with relationships and RI, and all schema
* Application planning documented, transactions analyzed, and security and physical parameters assessed
* Revised relevant queries

Week 14

* GUI completed for application – key transactions tested

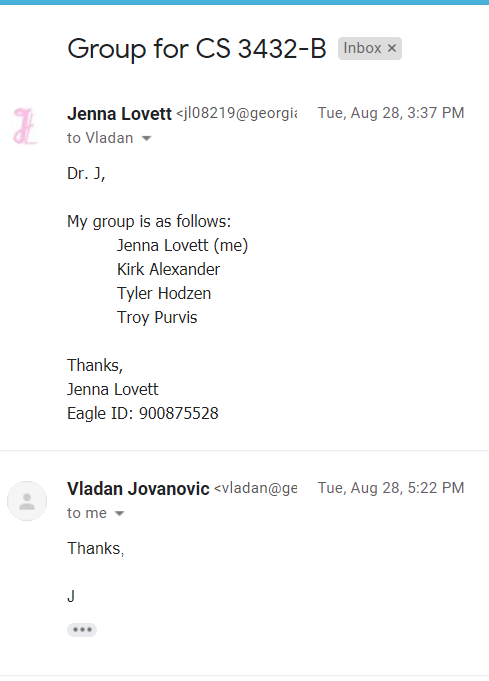
Week 15

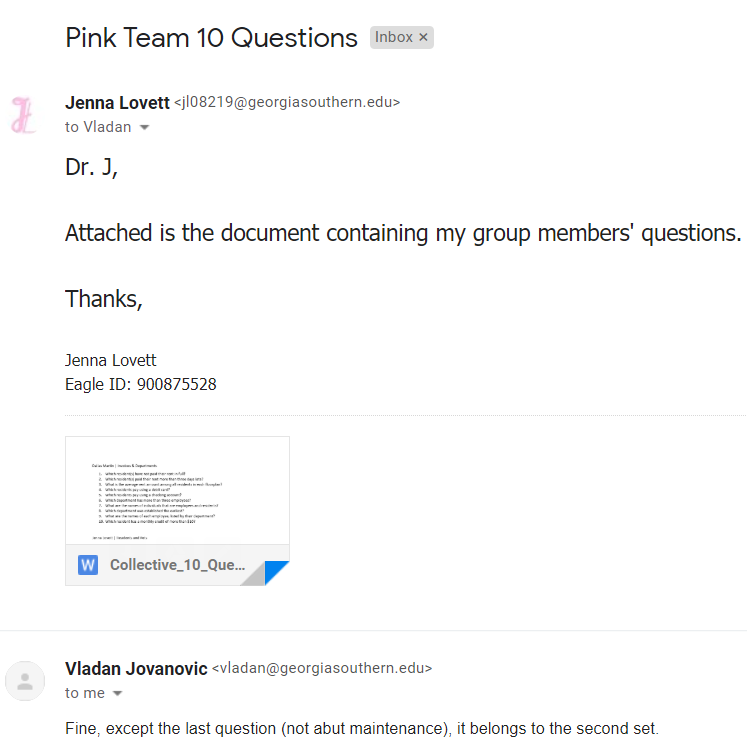
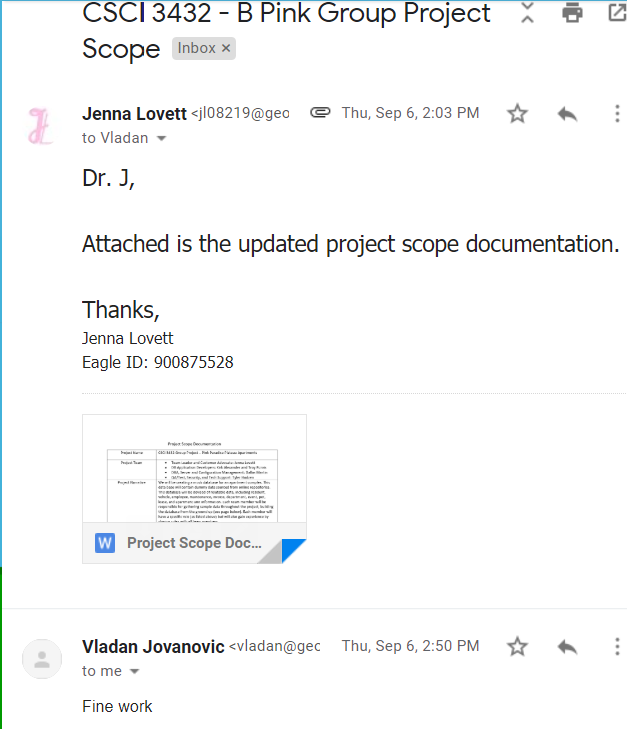
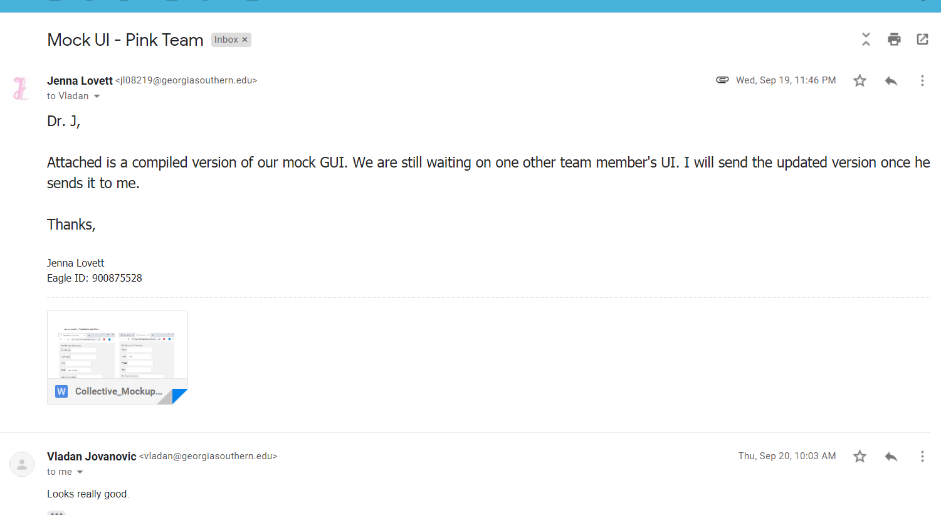
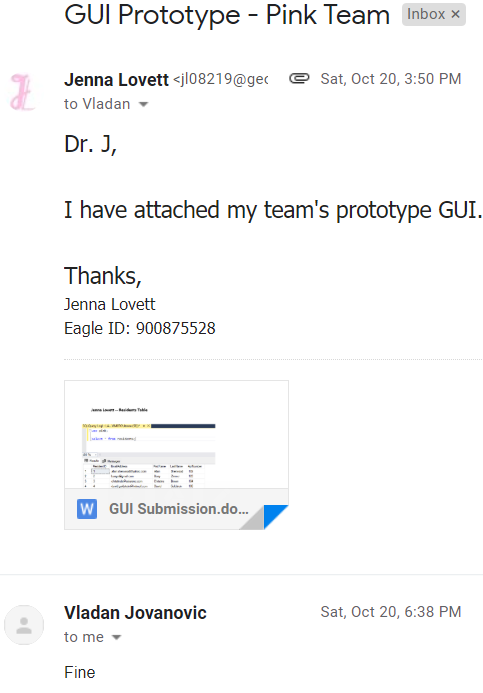
* Report and presentation completed and application effectiveness demonstrated

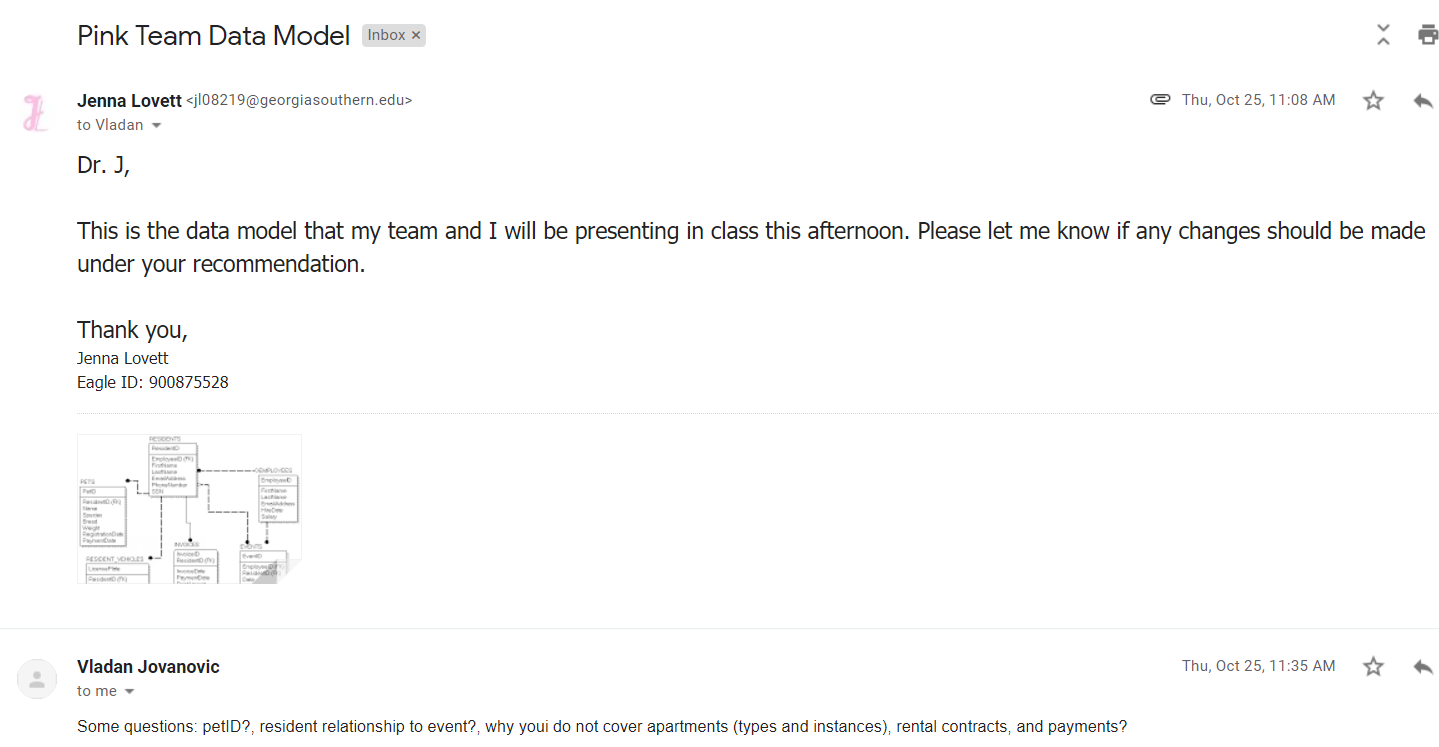
Week 16

* Oral Exam given over project

**Email Correspondence: Customer Advocate Work (Jenna Lovett)**







**Pink Database Script and Views (Tyler Hodzen, Troy Purvis, Jenna Lovett)**

USE master;

GO

IF DB\_ID('Pink') IS NOT NULL

DROP DATABASE Pink;

GO

CREATE DATABASE Pink;

GO

USE Pink;

-- create the tables for the database

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY IDENTITY,

EmailAddress VARCHAR(255) NOT NULL,

JobTitle CHAR(20) NOT NULL,

FirstName VARCHAR(255) NOT NULL,

LastName VARCHAR(255) NOT NULL,

HireDate DATETIME NOT NULL,

Salary DECIMAL(9,2) NOT NULL

);

CREATE TABLE Residents(

ResidentID INT PRIMARY KEY IDENTITY,

EmployeeID INT FOREIGN KEY REFERENCES Employees(EmployeeID) DEFAULT NULL,

EmailAddress VARCHAR(255) NOT NULL UNIQUE,

FirstName VARCHAR(60) NOT NULL,

LastName VARCHAR(60) NOT NULL,

PhoneNumber VARCHAR(20) NOT NULL

);

CREATE TABLE Payments (

PaymentID INT IDENTITY,

ResidentID INT NOT NULL,

PaymentDate DATETIME NOT NULL,

RentAmount MONEY NOT NULL,

CreditAmount MONEY NOT NULL,

DueDate DATETIME DEFAULT NULL,

CardType VARCHAR(50) NOT NULL,

CardNumber CHAR(16) NOT NULL,

constraint PK\_Payments primary key (PaymentID, ResidentID),

constraint FK\_Payments foreign key (ResidentID) references Residents(ResidentID)

);

CREATE TABLE Pets (

ResidentID int NOT NULL,

Species varchar(60),

Breed varchar(60),

RegistrationDate DateTime,

PaymentDate DateTime,

constraint PK\_PETS primary key (ResidentID),

constraint FK\_PETS\_ResidentID foreign key (ResidentID) references Residents(ResidentID)

);

CREATE TABLE ResidentVehicles (

LicensePlate varchar(60) NOT NULL,

ResidentID int NOT NULL,

Make varchar(20),

Model varchar(20),

Color varchar(20),

RegistrationDate DateTime,

DecalSticker varchar(20),

constraint PK\_RESIDENT\_VEHICLES primary key (LicensePlate),

constraint FK\_RESIDENT\_VEHICLES foreign key (ResidentID) REFERENCES Residents(ResidentID)

);

CREATE TABLE EventsLog (

EventID INT Not Null IDENTITY,

EmployeeID INT Foreign key References Employees (EmployeeID) Not Null,

ResidentID INT Foreign Key References Residents (ResidentID) Not Null,

EDate DateTime Not Null,

ELocation VARCHAR(50) Not Null,

EDesc VARCHAR(50) Not Null

constraint PK\_Events primary key (EventID)

);

CREATE TABLE Apartments (

ApartmentNum INT NOT NULL PRIMARY KEY IDENTITY,

NumBeds INT NOT NULL,

NumBaths INT NOT NULL,

Sqft INT NOT NULL

);

CREATE TABLE Contracts (

ContractID INT NOT NULL IDENTITY,

ResidentID INT FOREIGN KEY REFERENCES Residents(ResidentID) NOT NULL,

ApartmentNum INT FOREIGN KEY REFERENCES Apartments(ApartmentNum) NOT NULL,

Price MONEY NOT NULL,

StartDate DATE NOT NULL,

EndDate DATE NOT NULL,

PetsAllowed BIT NOT NULL,

SmokingAllowed BIT NOT NULL,

Specials VARCHAR(60) DEFAULT NULL,

constraint PK\_Contracts primary key (ContractID, ResidentID, ApartmentNum)

);

-- Insert data into the tables

SET IDENTITY\_INSERT Apartments ON;

INSERT INTO Apartments(ApartmentNum, NumBeds, NumBaths, Sqft) VALUES

(100, 4, 4, 1000),

(101, 4, 4, 1000),

(102, 4, 4, 1000),

(103, 4, 4, 1000),

(104, 4, 4, 1000),

(105, 4, 4, 1000),

(106, 4, 4, 1000),

(107, 4, 4, 1000),

(108, 4, 4, 1000),

(109, 4, 4, 1000),

(110, 4, 4, 1000);

SET IDENTITY\_INSERT Apartments OFF;

SET IDENTITY\_INSERT Employees ON;

INSERT INTO Employees(EmployeeID, EmailAddress, JobTitle, FirstName, LastName, HireDate, Salary) VALUES

(1, 'joe@pinkapartments.com', 'Groundsman', 'Joe', 'Davis', '2016-05-07 03:53:06.000', '50000.00'),

(2, 'mark@pinkapartments.com', 'Groundsman', 'Mark', 'Davis', '2016-05-07 03:53:06.000', '25000.00'),

(3, 'jerry@pinkapartments.com', 'Leaser', 'Jerry', 'Davis', '2016-05-07 03:53:06.000', '30000.00'),

(4, 'julie@pinkapartments.com', 'Leaser', 'Julie', 'Davis', '2016-05-07 03:53:06.000', '30000.00'),

(5, 'amy@myguitarshop.com', 'Leaser', 'Amy', 'Davis', '2016-05-07 03:53:06.000', '30000.00'),

(6, 'veronica@myguitarshop.com', 'Property Manager', 'Veronica', 'Davis', '2016-05-07 03:53:06.000', '30000.00'),

(7, 'david@pinkapartments.com', 'Security', 'David', 'Davis', '2016-05-07 03:53:06.000', '30000.00'),

(8, 'mary@pinkapartments.com', 'Security', 'Mary', 'Davis', '2016-05-07 03:53:06.000', '30000.00'),

(9, 'andy@myguitarshop.com', 'Electrician', 'Andy', 'Davis', '2016-05-07 03:53:06.000', '30000.00'),

(10, 'wilson@myguitarshop.com', 'Groundsman', 'Wilson', 'Davis', '2016-05-07 03:53:06.000', '30000.00');

SET IDENTITY\_INSERT Employees OFF;

SET IDENTITY\_INSERT Residents ON;

INSERT INTO Residents(ResidentID, EmployeeID, EmailAddress, FirstName, LastName, PhoneNumber) VALUES

(1, 1, 'allan.sherwood@yahoo.com', 'Allan', 'Sherwood', '555-555-5551'),

(2, NULL, 'barryz@gmail.com', 'Barry', 'Zimmer', '555-555-5552'),

(3, NULL, 'christineb@solarone.com', 'Christine', 'Brown', '555-555-5553'),

(4, NULL, 'david.goldstein@hotmail.com', 'David', 'Goldstein', '555-555-5554'),

(5, 2, 'erinv@gmail.com', 'Erin', 'Valentino', '555-555-5555'),

(6, NULL, 'frankwilson@sbcglobal.net', 'Frank Lee', 'Wilson', '555-555-5556'),

(7, NULL, 'gary\_hernandez@yahoo.com', 'Gary', 'Hernandez', '555-555-5557'),

(8, NULL, 'heatheresway@mac.com', 'Heather', 'Esway', '555-555-5558'),

(9, 3, 'jbutt@gmail.com', 'James', 'Butt', '555-555-5559'),

(10, 4, 'josephine\_darakjy@darakjy.org', 'Josephine', 'Darakjy', '555-555-5510');

SET IDENTITY\_INSERT Residents OFF;

SET IDENTITY\_INSERT Contracts ON;

INSERT INTO Contracts(ContractID, ResidentID, ApartmentNum, Price, StartDate, EndDate, PetsAllowed, SmokingAllowed, Specials) VALUES

(1, 1, 100, 1000.00, '2015-9-5', '2016-9-5', 1, 1, 'Free XBox'),

(2, 2, 101, 1000.00, '2015-9-5', '2016-9-5', 1, 1, 'Free XBox'),

(3, 3, 102, 1000.00, '2015-9-5', '2016-9-5', 1, 1, 'Free XBox'),

(4, 4, 103, 1000.00, '2015-9-5', '2016-9-5', 1, 1, 'Free XBox'),

(5, 5, 104, 1000.00, '2015-9-5', '2016-9-5', 1, 1, 'Free XBox'),

(6, 6, 105, 1000.00, '2015-9-5', '2016-9-5', 1, 1, 'Free XBox'),

(7, 7, 106, 1000.00, '2015-9-5', '2016-9-5', 1, 1, 'Free XBox'),

(8, 8, 107, 1000.00, '2015-9-5', '2016-9-5', 1, 1, 'Free XBox'),

(9, 9, 108, 1000.00, '2015-9-5', '2016-9-5', 1, 1, 'Free XBox'),

(10, 10, 109, 1000.00, '2015-9-5', '2016-9-5', 1, 1, 'Free XBox');

SET IDENTITY\_INSERT Contracts OFF;

SET IDENTITY\_INSERT EventsLog ON;

INSERT INTO EventsLog(EventID, EmployeeID, ResidentID, EDate, ELocation, EDesc) VALUES

(1, 1, 1, '2015-9-5', 'Theater', 'Watched a movie'),

(2, 1, 1, '2015-10-5', 'Pool', 'Swimming Match'),

(3, 1, 1, '2015-9-8', 'Pool', 'Swimming Match'),

(4, 1, 1, '2015-9-12', 'Grill', 'BBQ'),

(5, 1, 1, '2015-10-31', 'Pool', 'Swimming Match'),

(6, 1, 1, '2016-9-10', 'Theater', 'Watched a movie'),

(7, 1, 1, '2015-12-5', 'Lounge', 'Watched a movie'),

(8, 1, 1, '2015-11-15', 'Bar', 'Drank'),

(9, 1, 1, '2015-9-15', 'Tennis Courts', 'Played tennis'),

(10, 1, 1, '2015-9-15', 'Bar', 'Drank'),

(11, 1, 1, '2015-9-16', 'Pool', 'Swimming Match');

SET IDENTITY\_INSERT EventsLog OFF;

INSERT INTO Pets (ResidentID, Species, Breed, RegistrationDate, PaymentDate)

VALUES (1, 'Dog', 'German Shepherd', '2016-5-5', '2016-5-10'),

(2, 'Cat', 'Calico', '2015-8-1', '2015-9-5'),

(3, 'Dog', 'Tibby', '2016-1-5', '2016-1-30'),

(4, 'Cat', 'Alley', '2017-3-3', '2017-4-20'),

(5, 'Dog', 'Pug', '2016-10-10', '2016-11-12');

INSERT INTO ResidentVehicles (LicensePlate, ResidentID, Make, Model, Color, RegistrationDate, DecalSticker)

VALUES ( 'abc-123', 1, 'Honda', 'Civic', 'Black', '2016-5-5', 'a0sijf'),

( 'zbd-14d', 2, 'Ford', 'F-150', 'Grey', '2015-1-30', 'asoihf'),

( 'bdb-322', 3, 'Tesla', 'Model 3', 'Red', '2018-5-3', 'memes'),

( 'asd-fgh', 4, 'Nissan', 'GTR', 'Black', '2013-9-4', 'zxcvgr'),

( 'jkhdgfg', 5, 'Volkswagen', 'Beatle', 'Blue', '2018-2-3', 'qwertyu');

SET IDENTITY\_INSERT Payments ON;

INSERT INTO Payments(PaymentID, ResidentID, PaymentDate, RentAmount, CreditAmount, DueDate, CardType, CardNumber) VALUES

(1, 1, '2016-03-28 09:40:28.000', 588.75, 0.0, '2016-03-31 09:41:11.000', 'Visa', '4111111111111111'),

(2, 2, '2016-03-28 09:40:28.000', 600.75, 0.0, '2016-03-31 09:41:11.000', 'American Express', '3782822463100005'),

(3, 3, '2016-03-28 09:40:28.000', 475.75, 0.0, '2016-03-31 09:41:11.000', 'Discover', '6011111111111117'),

(4, 4, '2016-03-28 09:40:28.000', 999.75, 0.0, '2016-03-31 09:41:11.000', 'MasterCard', '5555555555554444'),

(5, 5, '2016-03-28 09:40:28.000', 475.75, 0.0, '2016-03-31 09:41:11.000', 'Visa', '101010101010'),

(6, 6, '2016-03-28 09:40:28.000', 588.75, 0.0, '2016-03-31 09:41:11.000', 'MasterCard', '999999999999'),

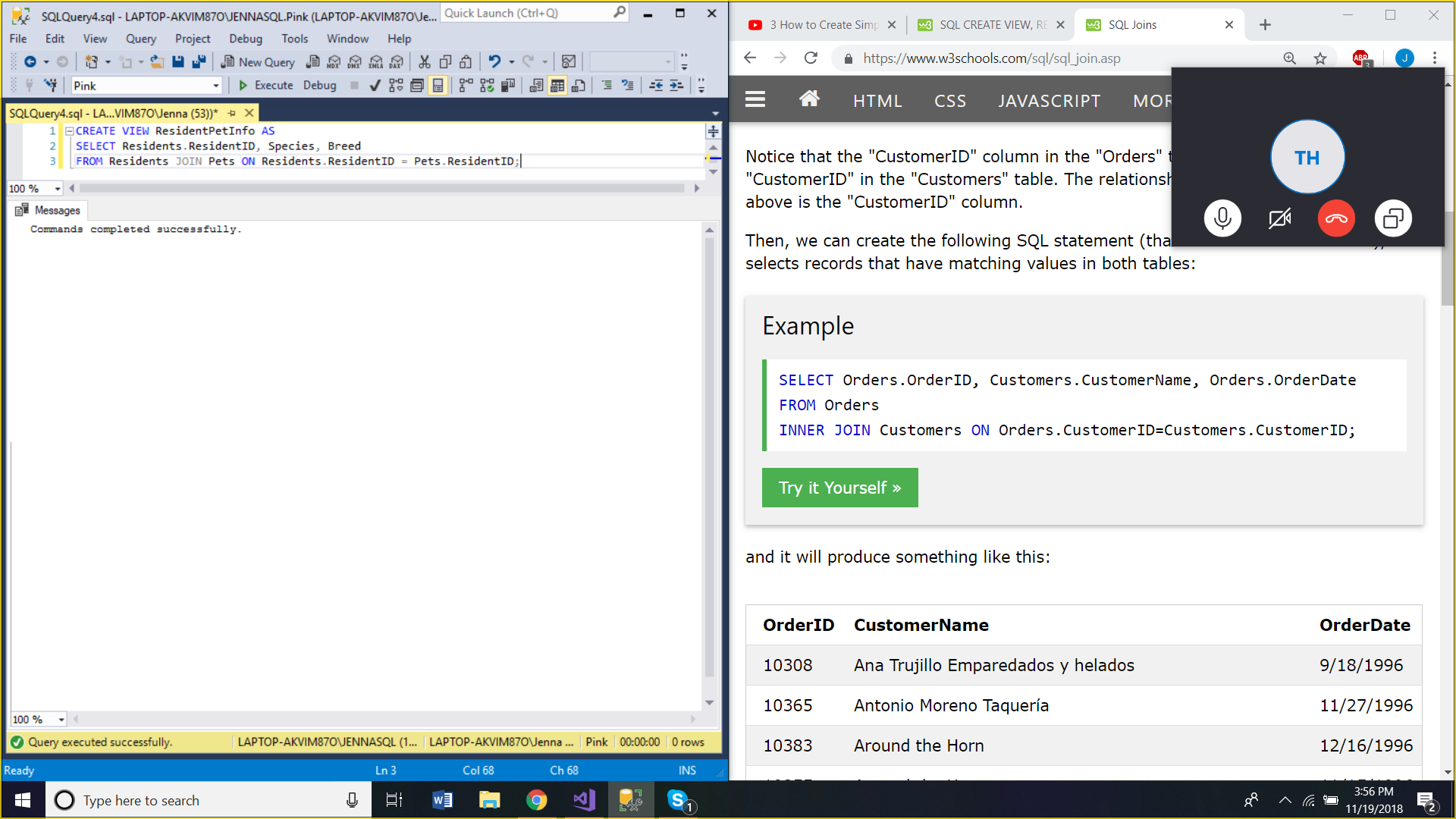
(7, 7, '2016-03-28 09:40:28.000', 500.75, 0.0, '2016-03-31 09:41:11.000', 'Visa', '98989898989898'),

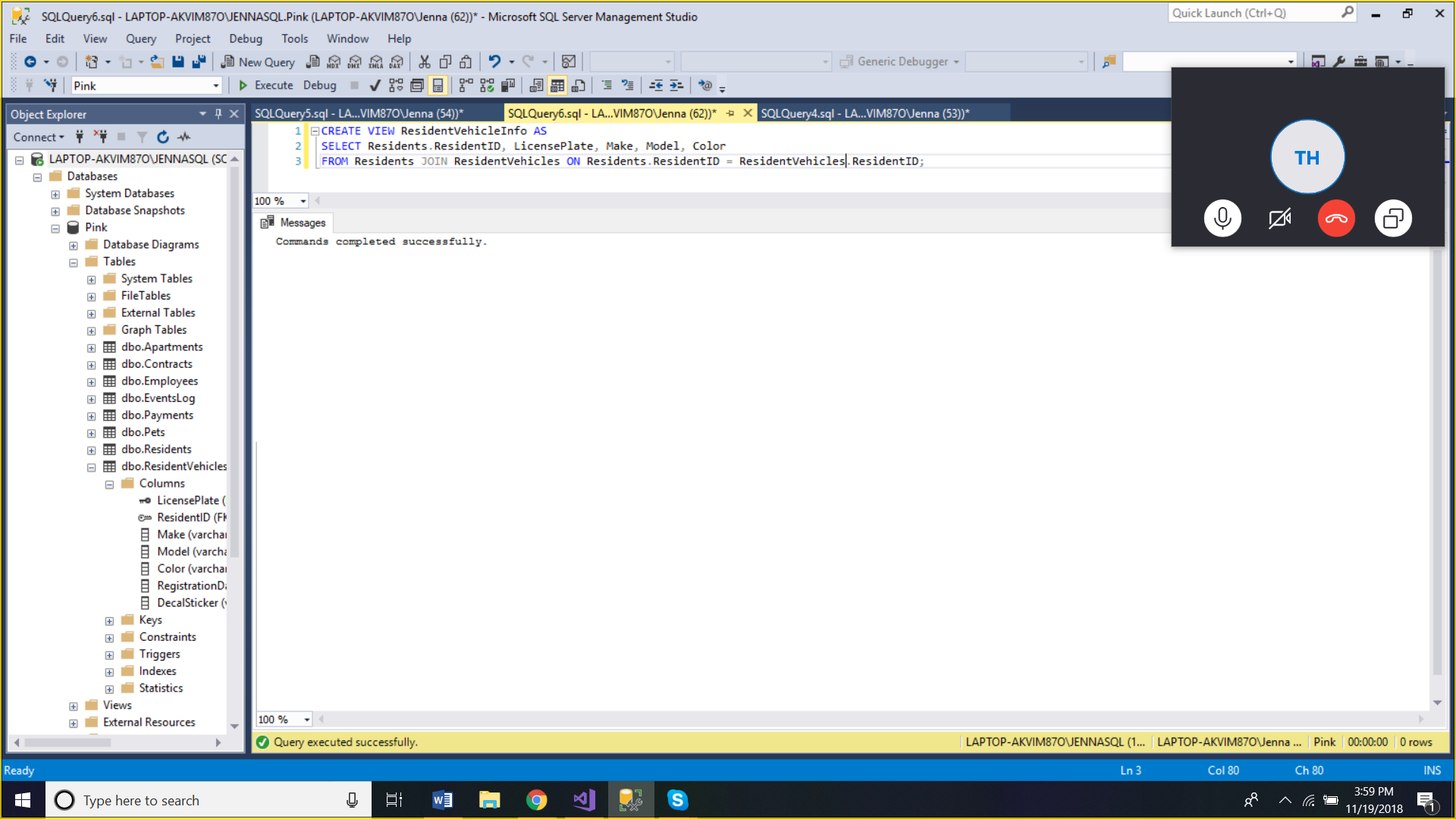
(8, 8, '2016-03-28 09:40:28.000', 475.75, 0.0, '2016-03-31 09:41:11.000', 'Visa', '4111111111111111'),

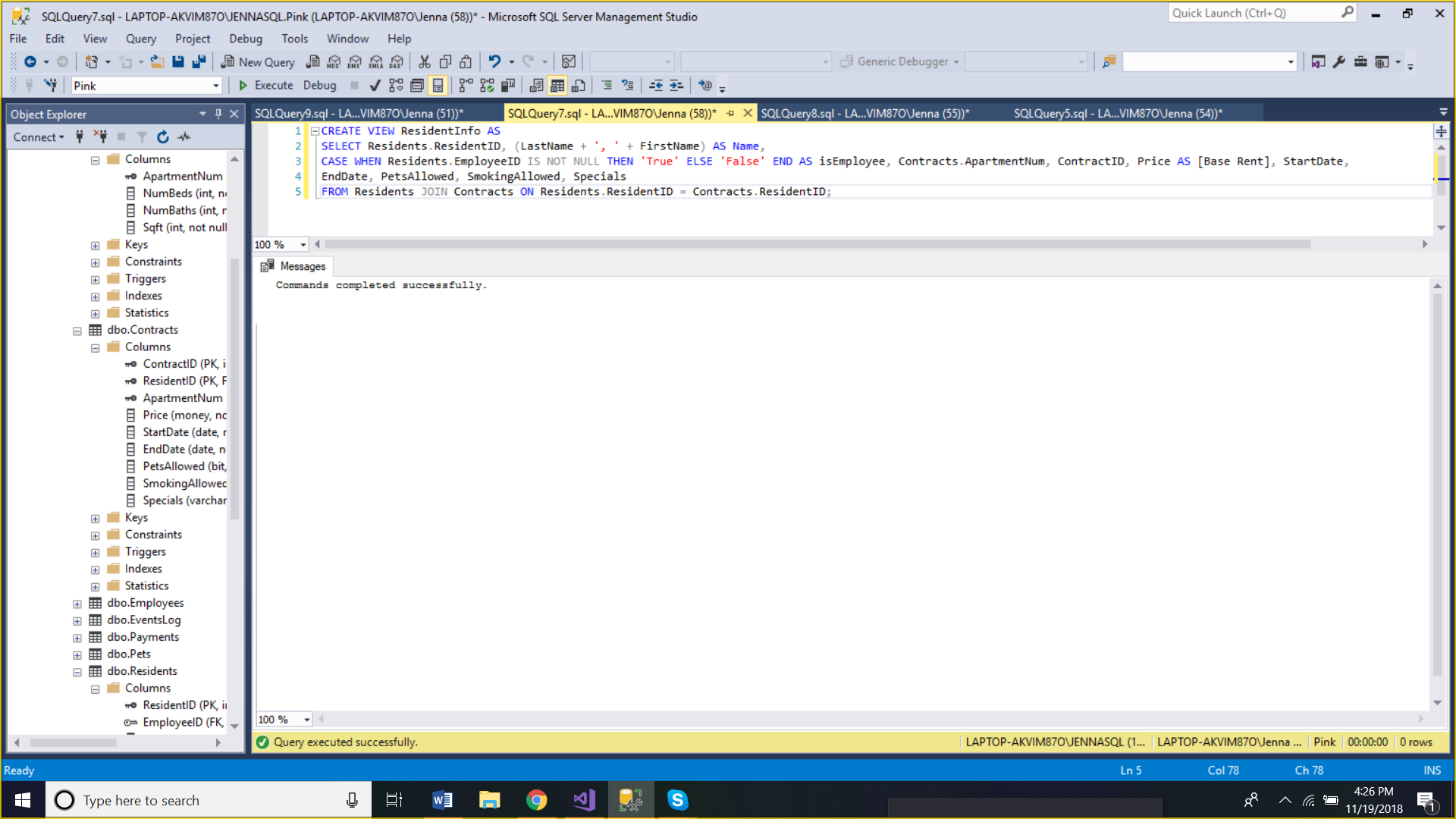
(9, 9, '2016-03-28 09:40:28.000', 588.75, 0.0, '2016-03-31 09:41:11.000', 'Visa', '4111111111111111'),

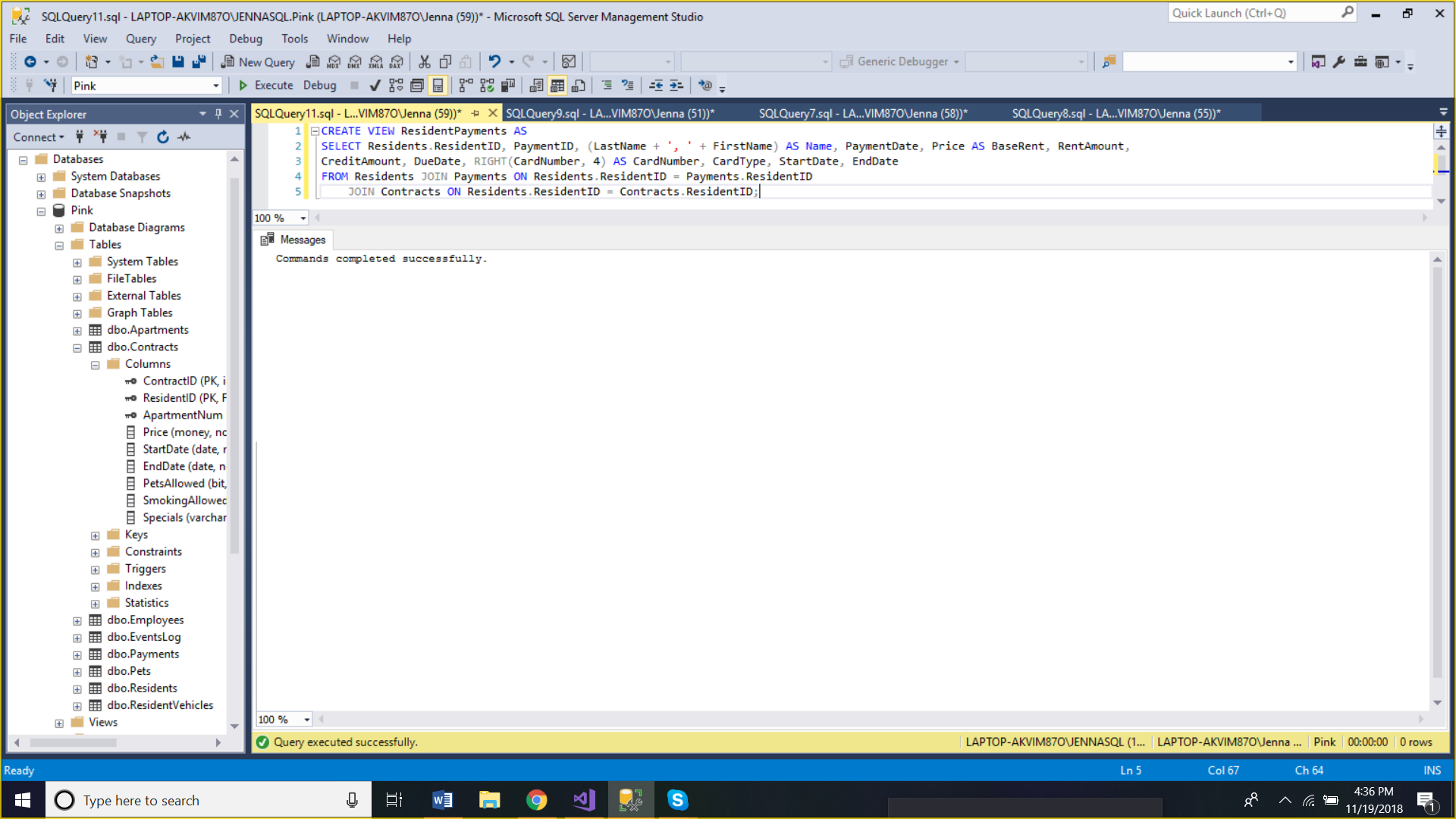
(10, 10, '2016-03-28 09:40:28.000', 588.75, 0.0, '2016-03-31 09:41:11.000', 'Visa', '4111111111111111');

SET IDENTITY\_INSERT Payments OFF;

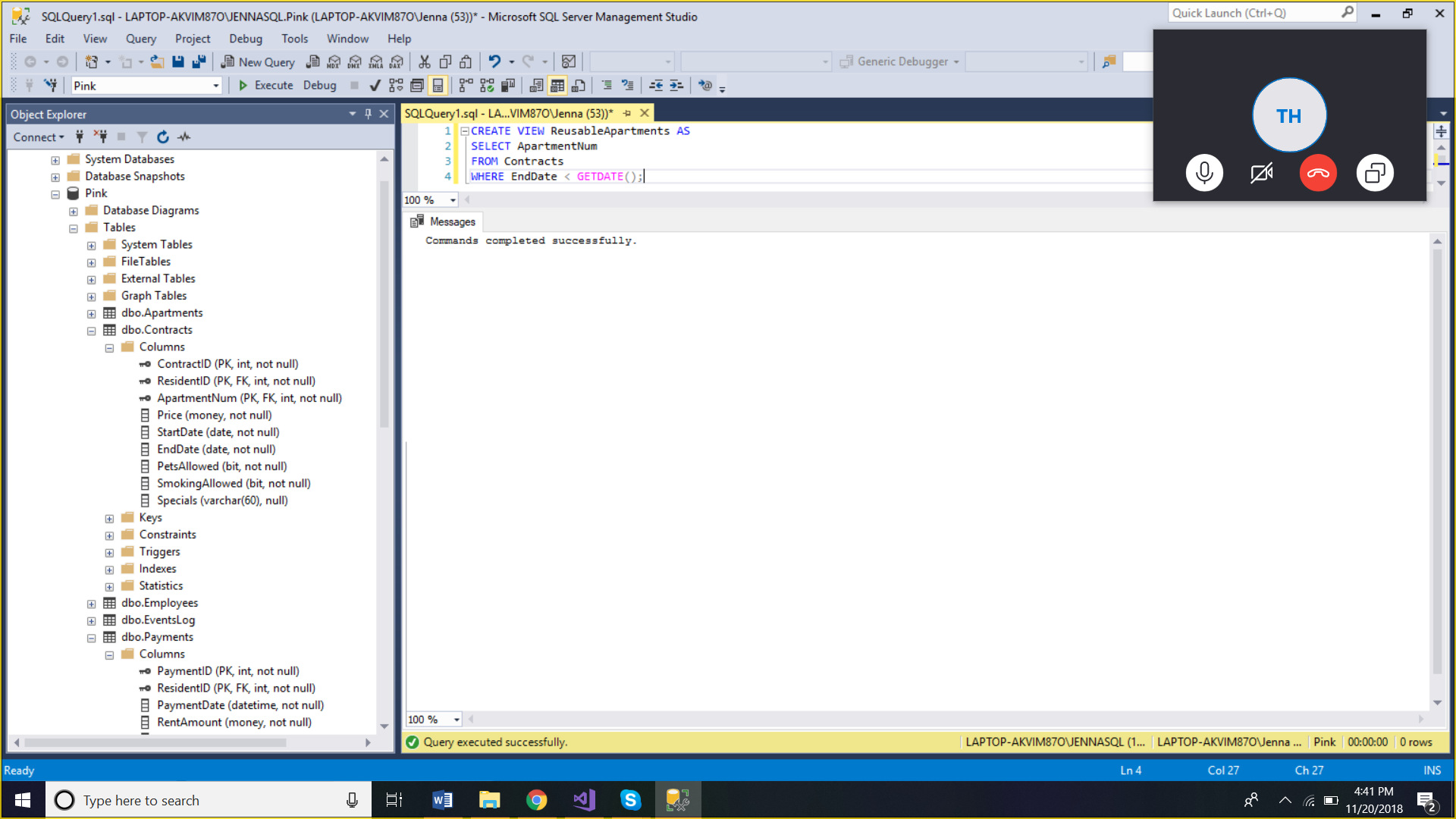


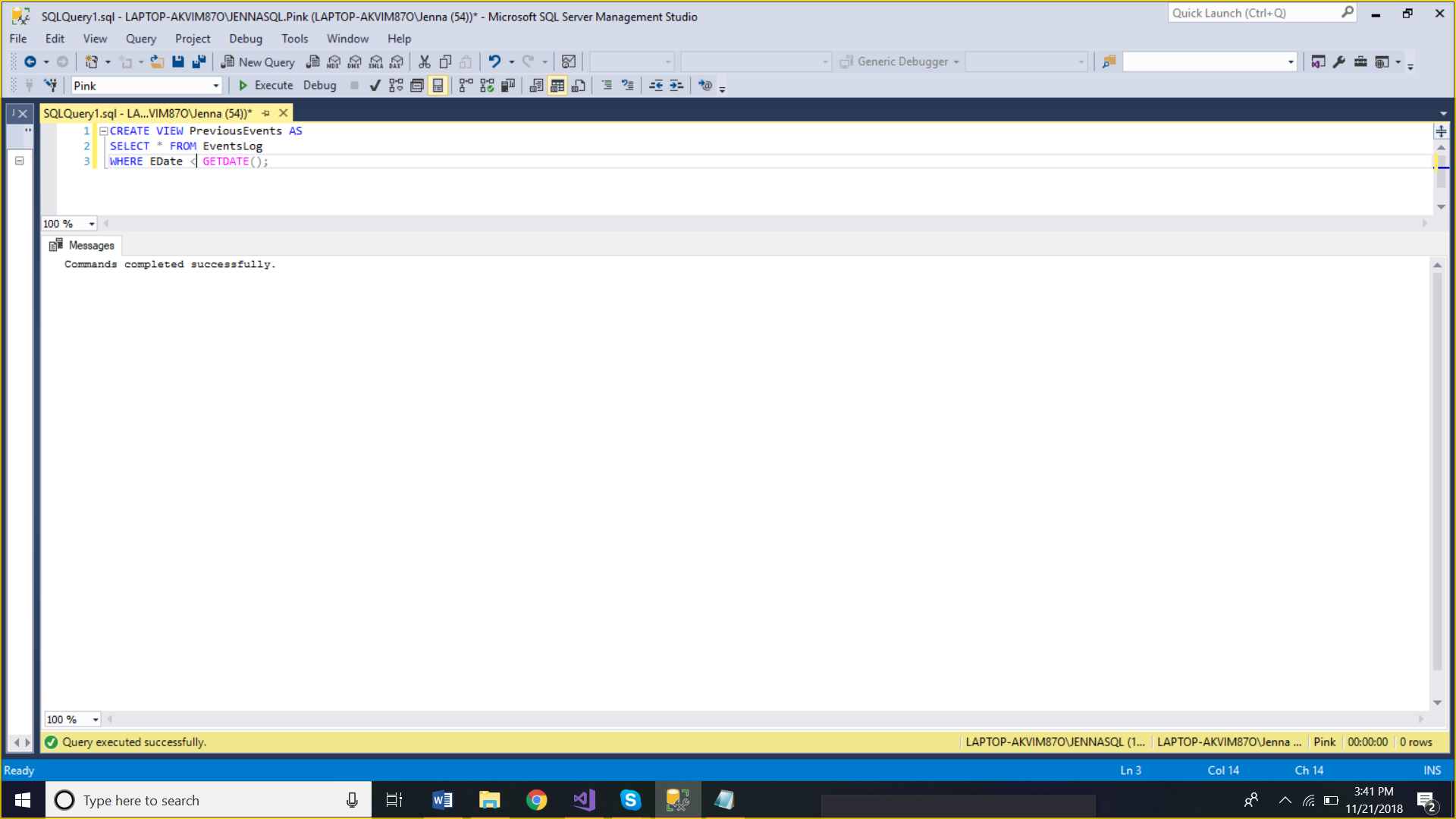


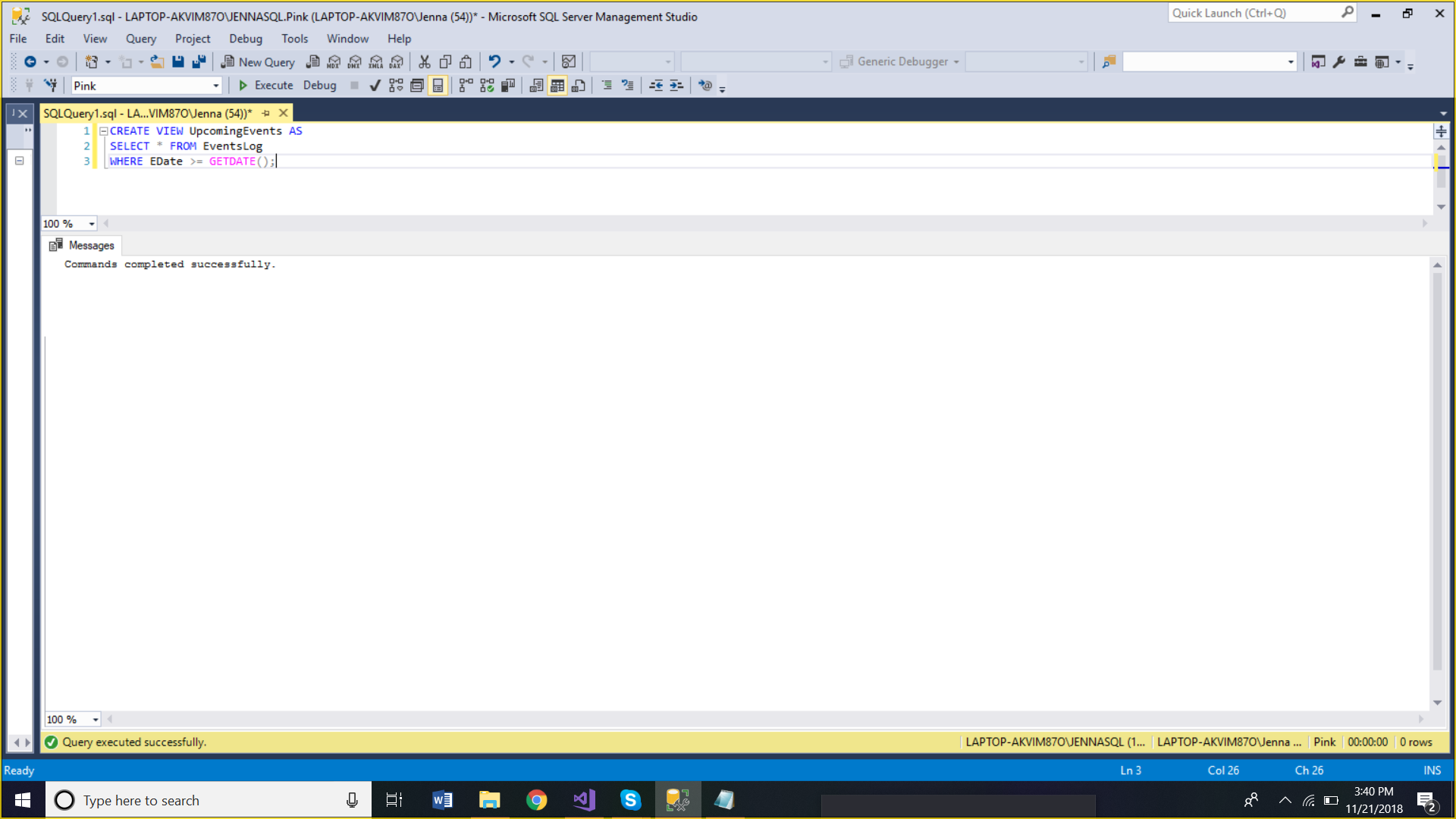




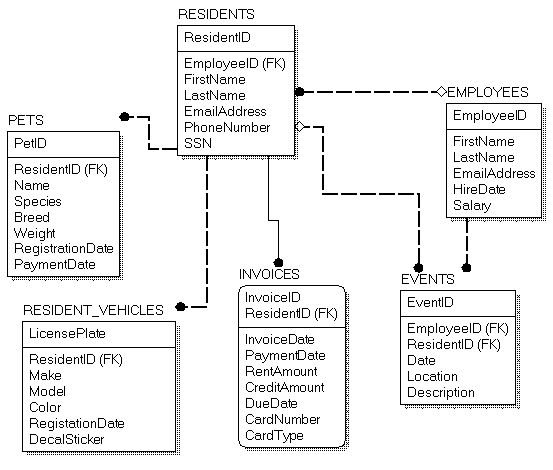




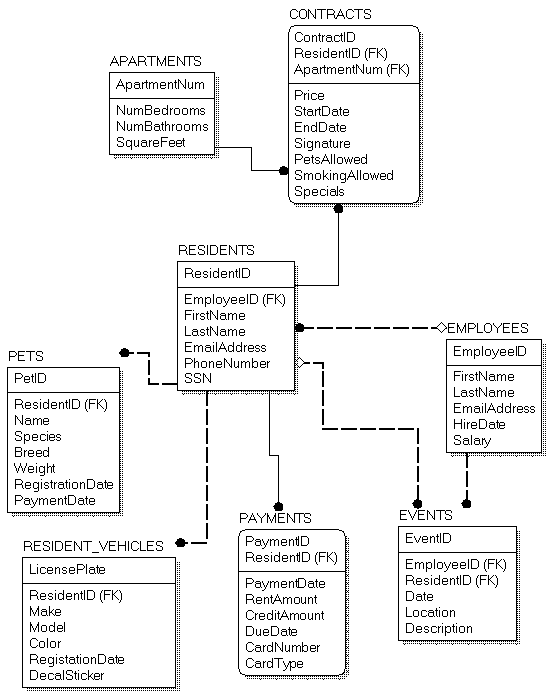




**Data Model** (Tyler Hodzen, Jenna Lovett, Troy Purvis)



Version 1



Version 2

**Collective 10 Questions**

Jenna Lovett | Residents and Pets

1. Which pet weighs the most, what is its breed, and which resident does it belong to?
2. What is the most common pet name?
3. What is the average age of residents?
4. Who are the residents with last names between B and L and first names between J and R?
5. Which resident lives in apartment 101 and do they have pets?
6. Which residents have indicated that they will register a pet when signing their lease, but have not registered the pet?
7. Which resident has lived on premises the longest?
8. How many residents own pets?
9. How many pets are dogs with a primarily black coat?
10. How many pets are fish and what is the most common primary color of said fish?

Troy Purvis | Vehicles and Events tables

1. Return all vehicles with the same make/model/color (include resident ID, Make, model, & Color)

2. Return a specific vehicle’s info given the decal sticker or license plate number

3. Return all residents with a valid registered vehicle (registration dates could expire in some arbitrary time frame)

4. Return all vehicles that have a certain pattern in their license plate (use regex)

5. Return all vehicles that have a registration date that has expired or will be expiring soon

6. Return only employees and their vehicles that have valid registration dates (not residents)

7. Return all vehicles that were registered on a specific day/month/year (Prefix matching)

8. Return all events happening in the next day/month/year

9. Return all vehicles that are owned by a resident with a pet

10. Return the vehicles of people with apartment sizes > 1000 square feet

Tyler Hodzen | Maintenance and Employees

1. How many employees are a part of the maintenance staff?

2. How many employees live in the apartment complex?

3. List all the maintenance staff-members that have at least 1 workorder to complete.

4. How many employees are considered fulltime staff (30+ hours)?

5. List all the job positions, and how many employees are in those positions, in descending order.

6. Which employee has called out of work the most?

7. List the top 5 most requested workorders (ex: toilet repair, sink repair, thermometer repair, etc.).

8. List the name of all employees starting with the employee who has been an employee the longest.

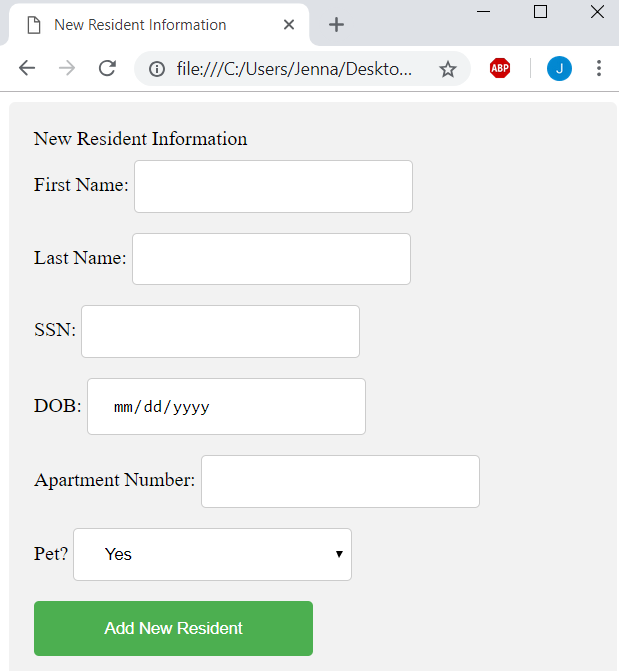
9. List all the 3rd party companies the apartment complex must use (ISP company, cable company, etc.)

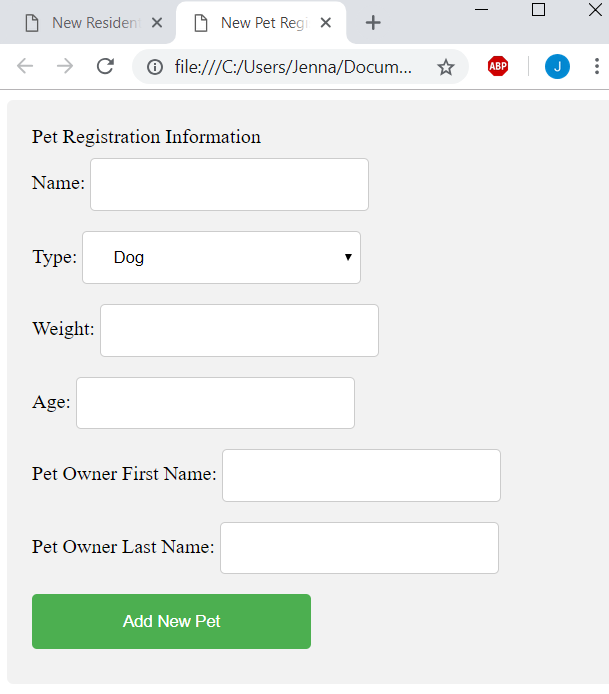
10. How many employees own pets and which employee owns the most?

~~Dallas Martin | Invoices & Departments~~

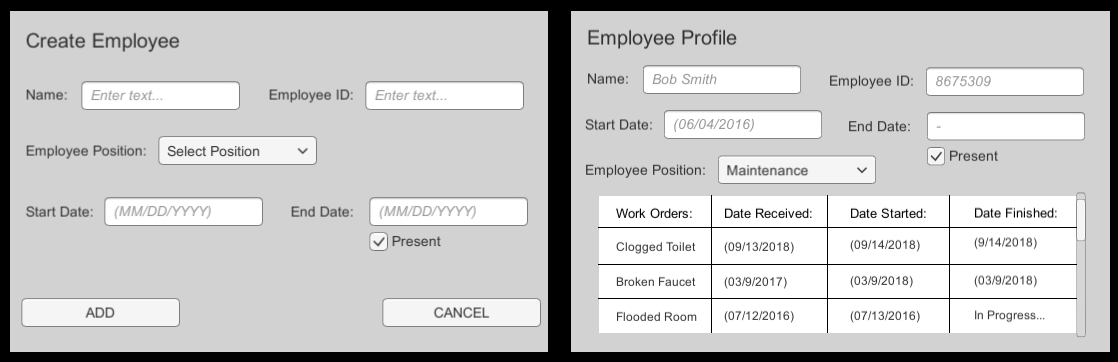
1. ~~Which resident(s) have not paid their rent in full?~~
2. ~~Which resident(s) paid their rent more than three days late?~~
3. ~~What is the average rent amount among all residents in each floorplan?~~
4. ~~Which residents pay using a debit card?~~
5. ~~Which residents pay using a checking account?~~
6. ~~Which department has more than three employees?~~
7. ~~What are the names of individuals that are employees and residents?~~
8. ~~Which department was established the earliest?~~
9. ~~What are the names of each employee, listed by their department?~~
10. ~~Which resident has a monthly credit of more than $10?~~

**Mock UI**

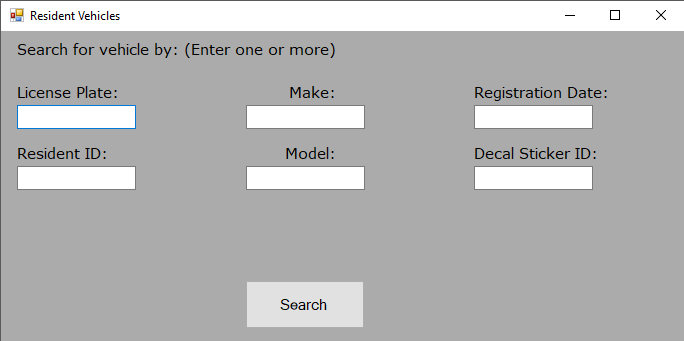
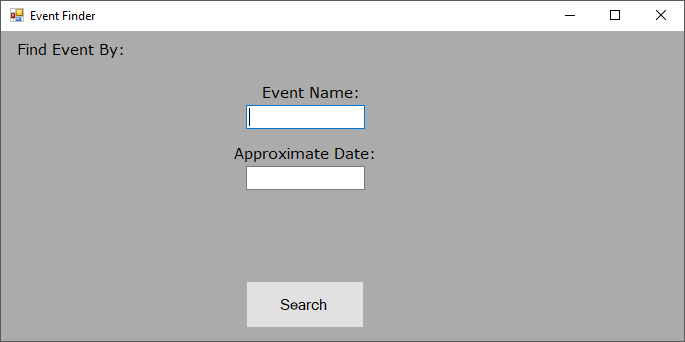
Jenna Lovett | Residents and Pets

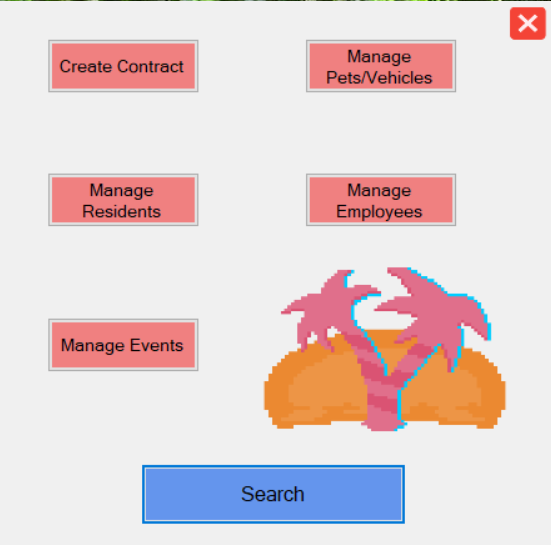
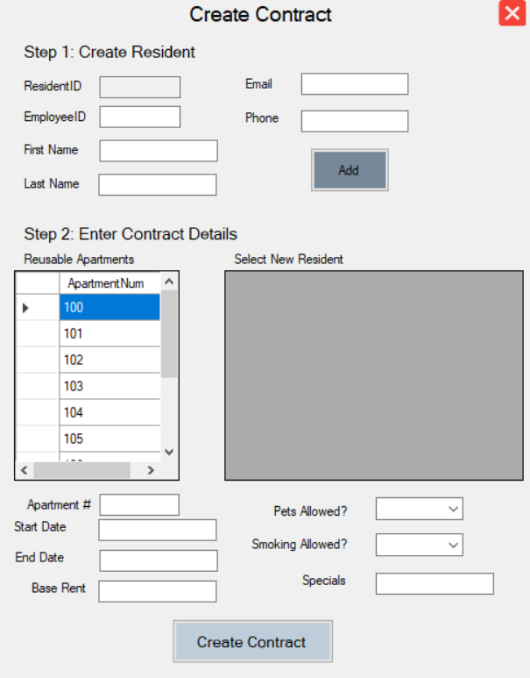


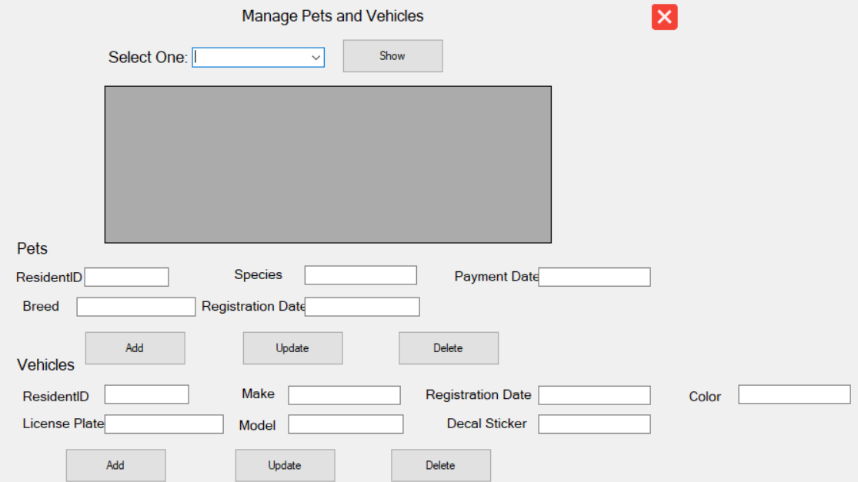
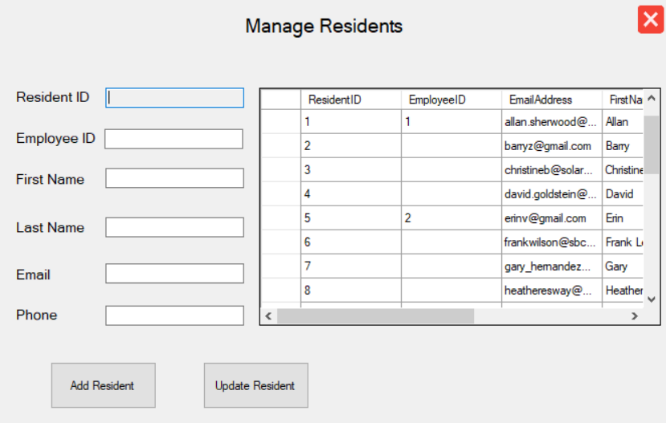
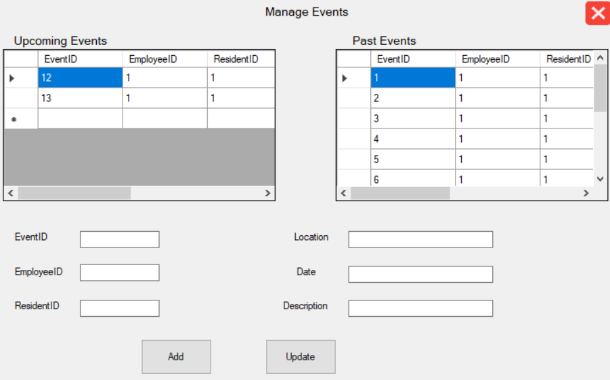
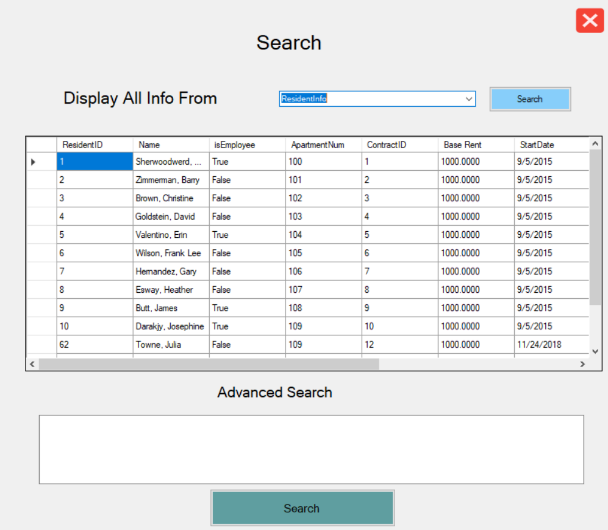
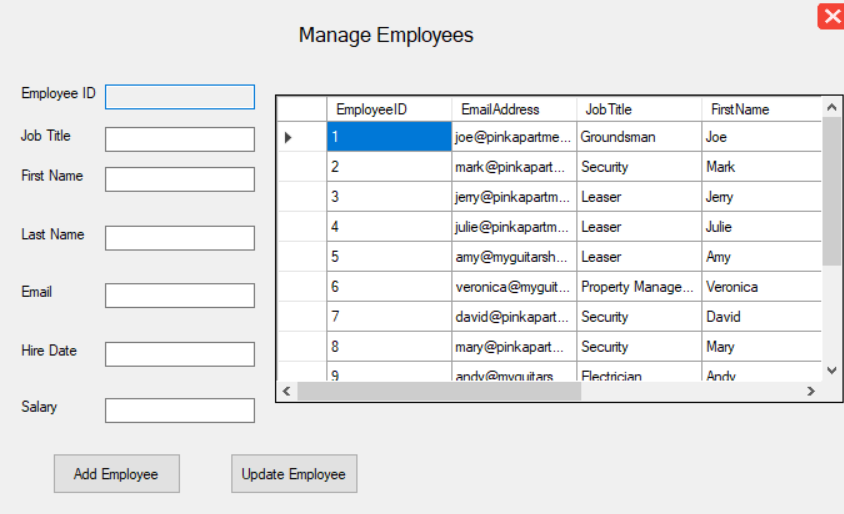
Tyler Hodzen |Employees and Maintenance

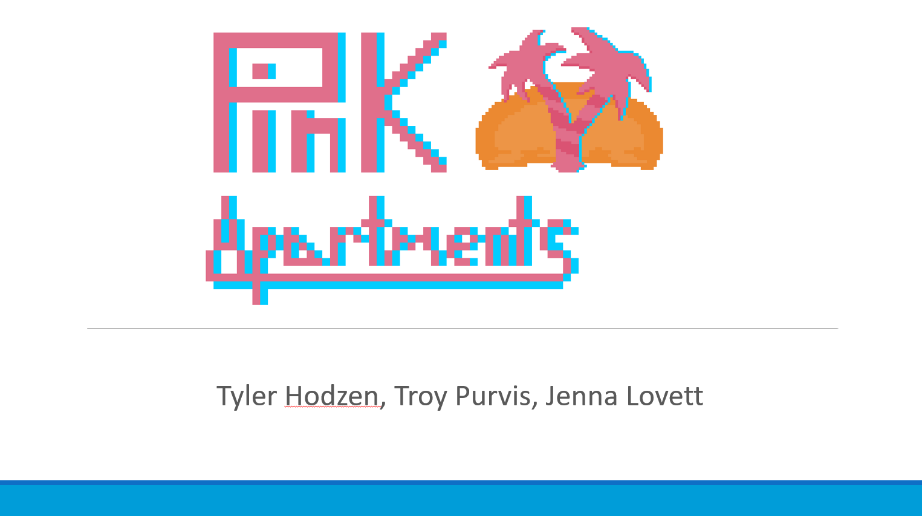
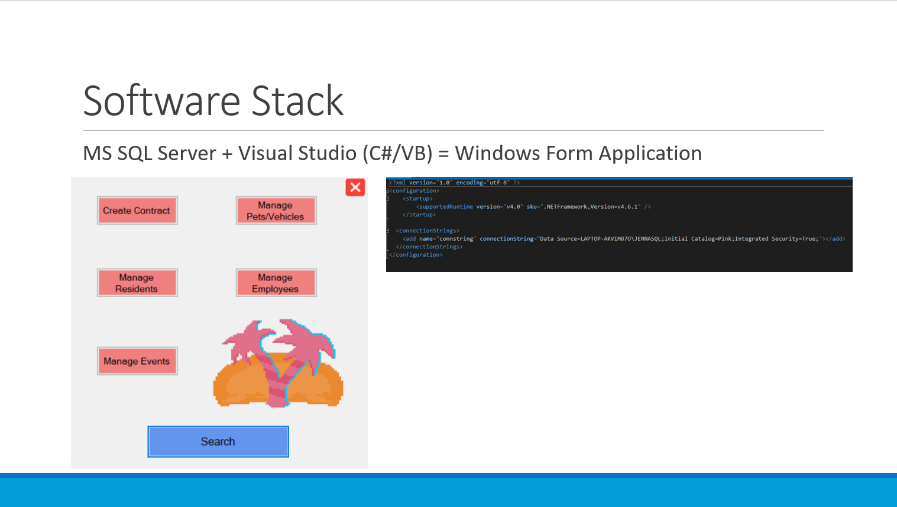
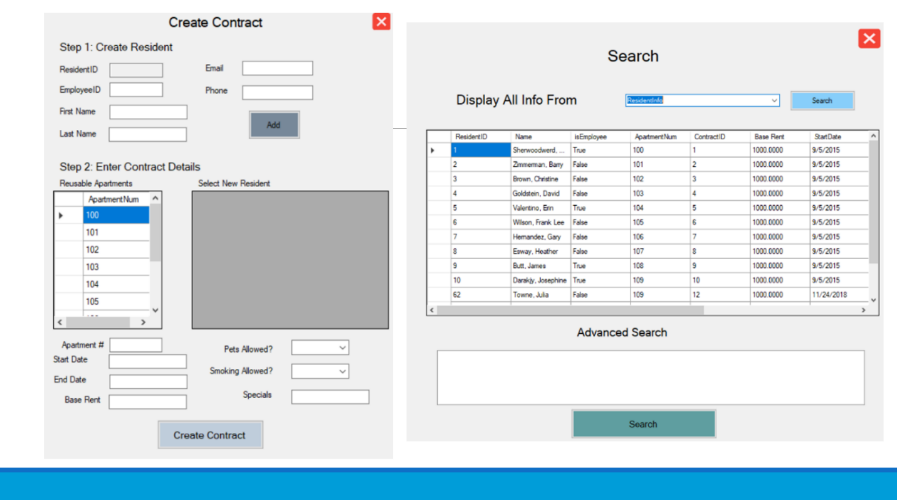


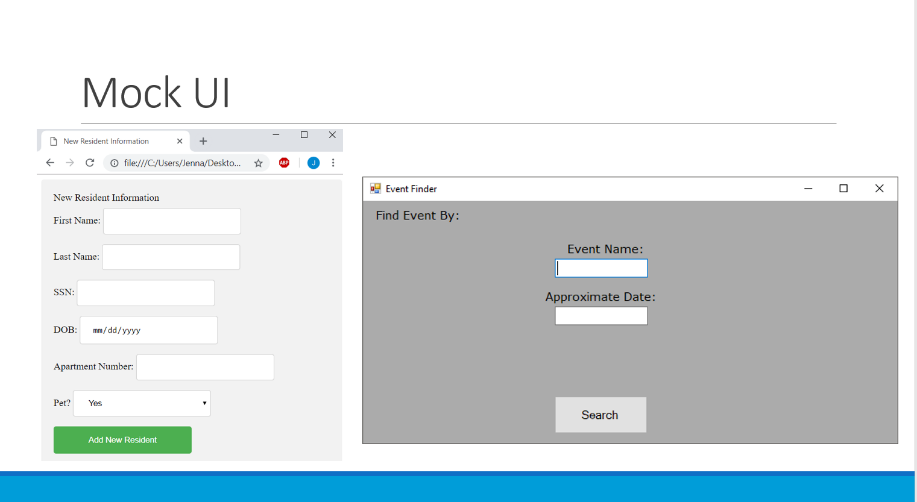
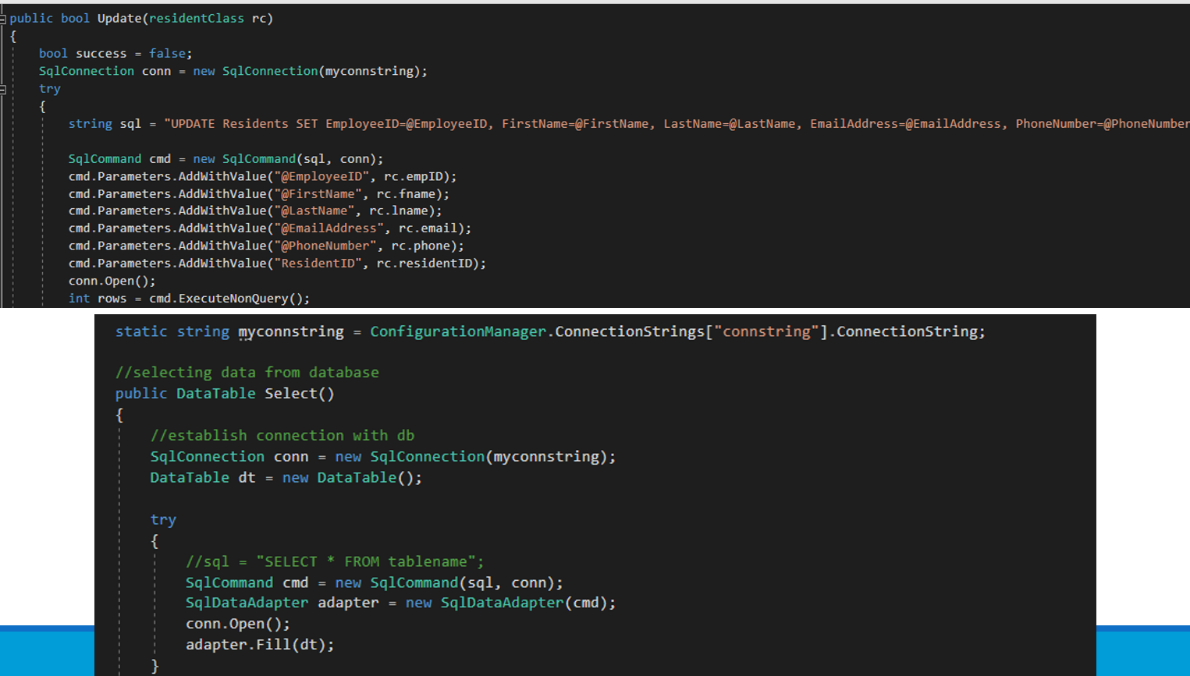
Troy Purvis | Events and Vehicles

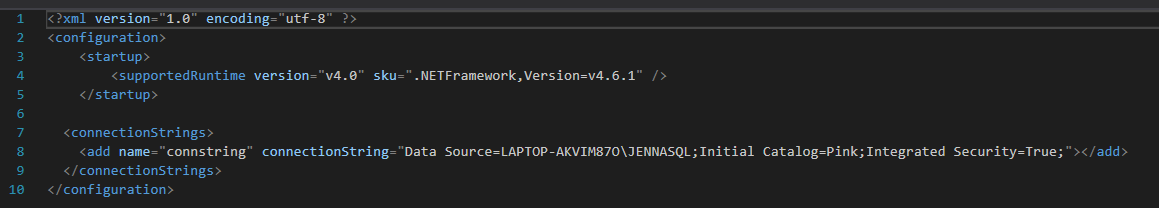


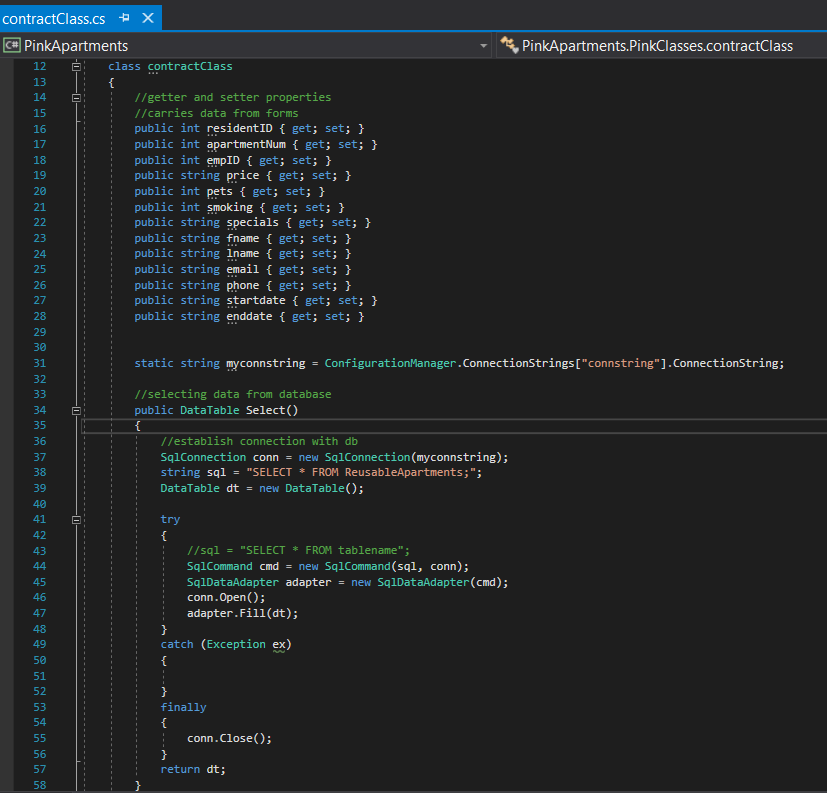
**Final UI (**Tyler Hodzen, Troy Purvis, Jenna Lovett)

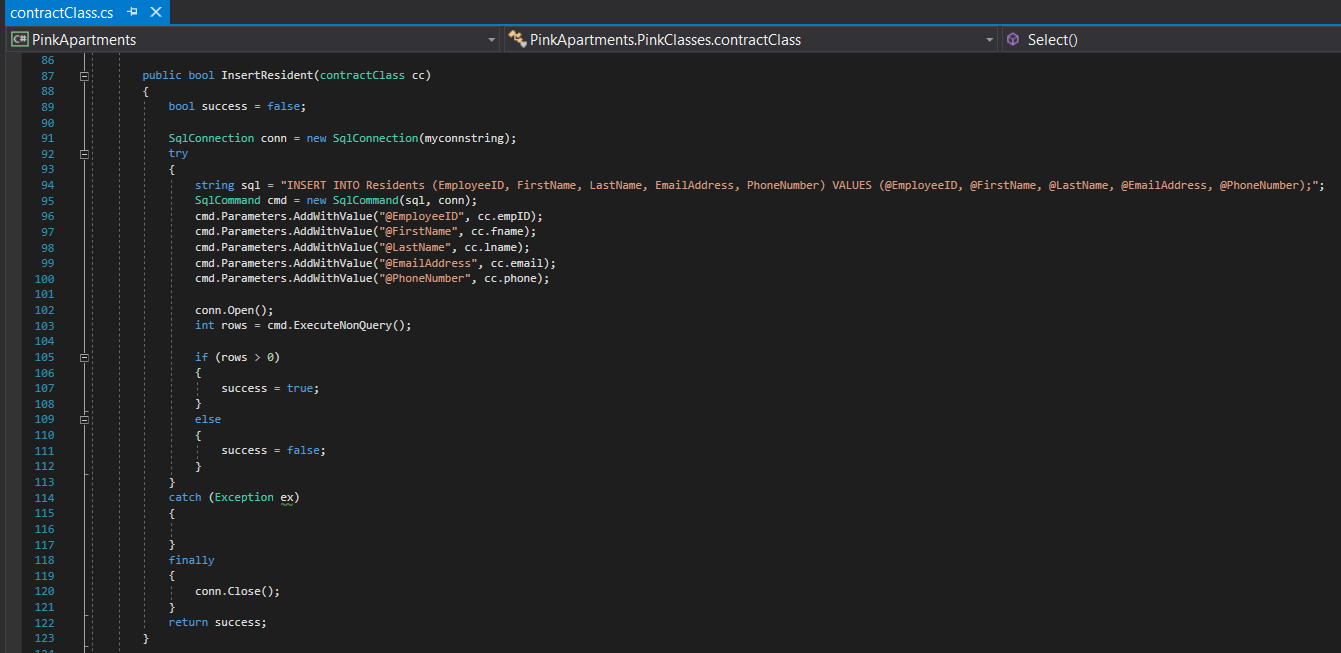


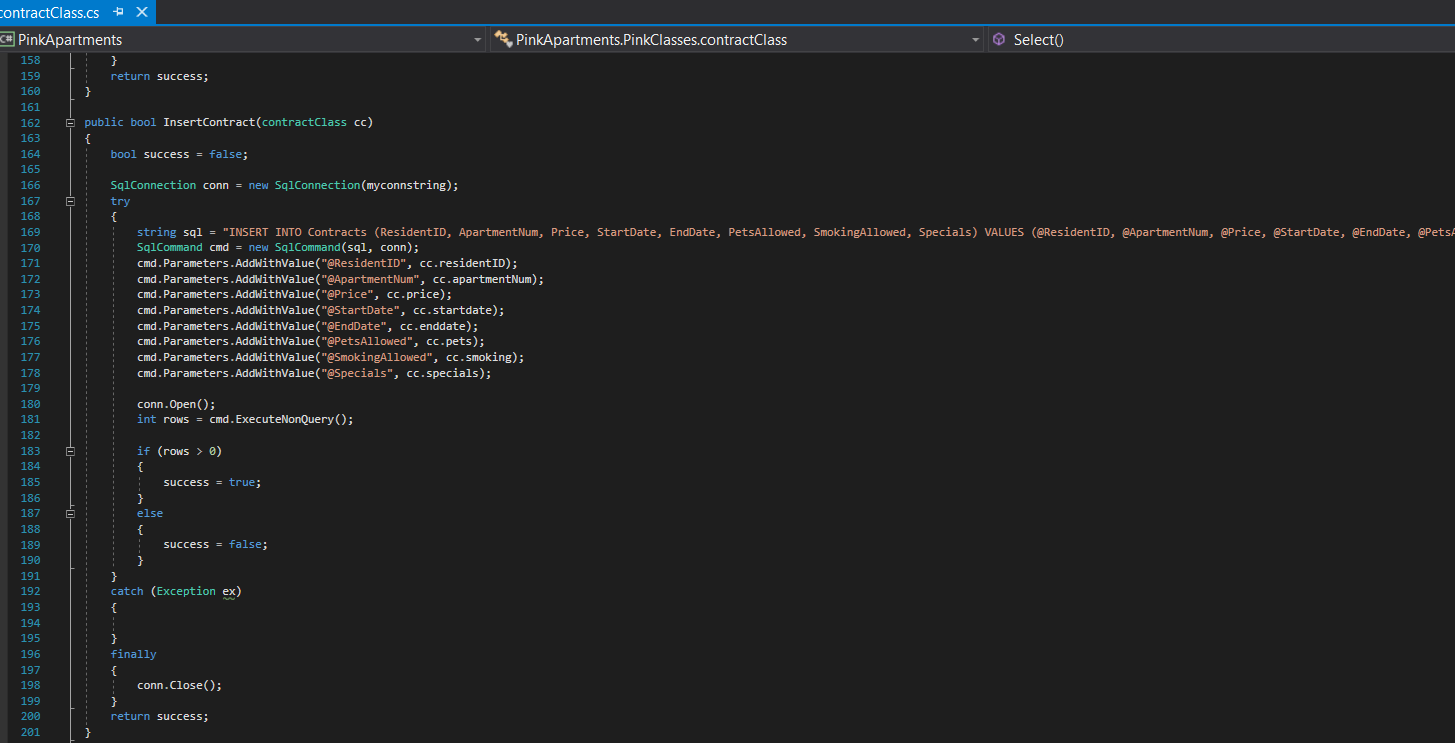
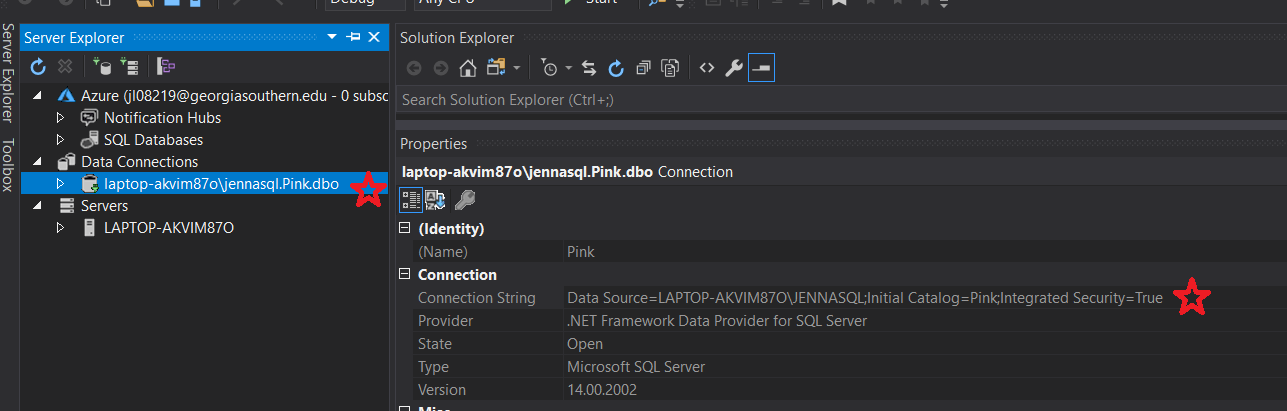
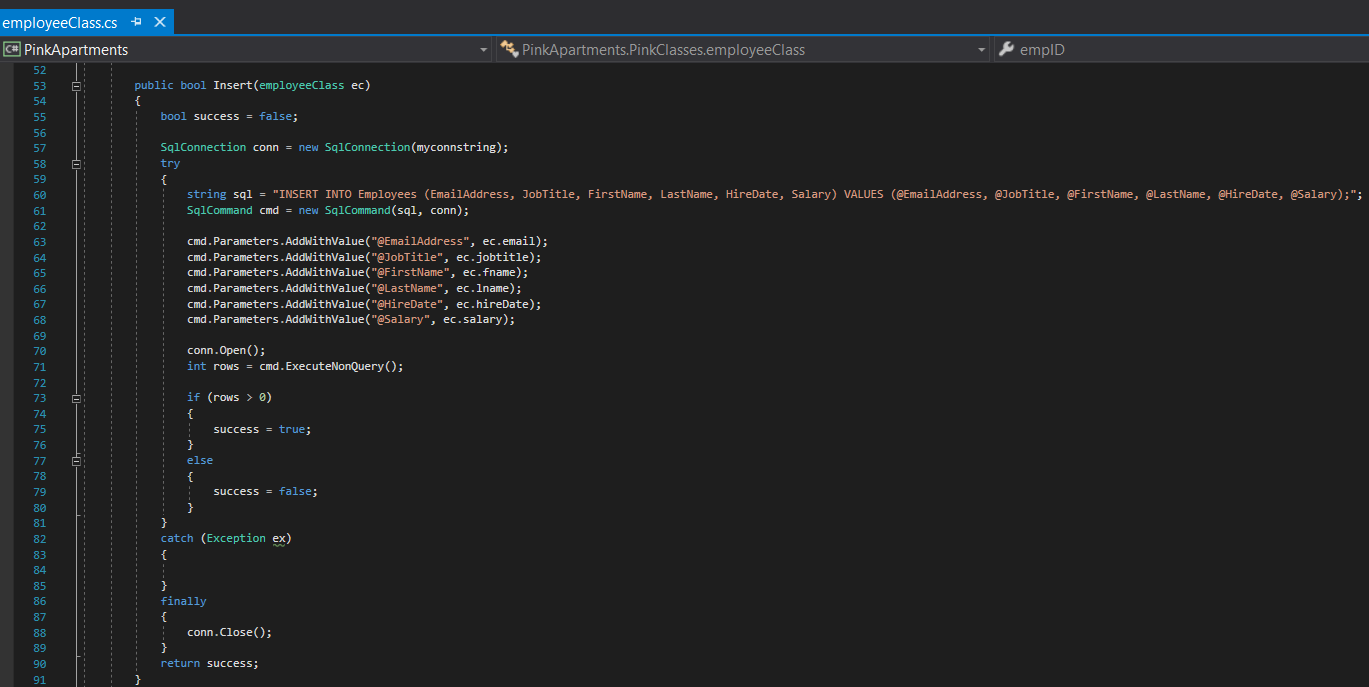
**Final Presentation**

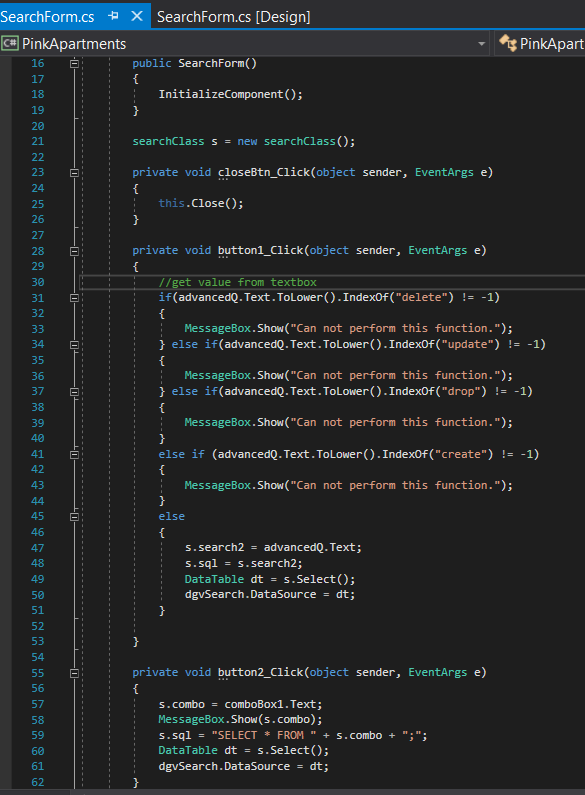


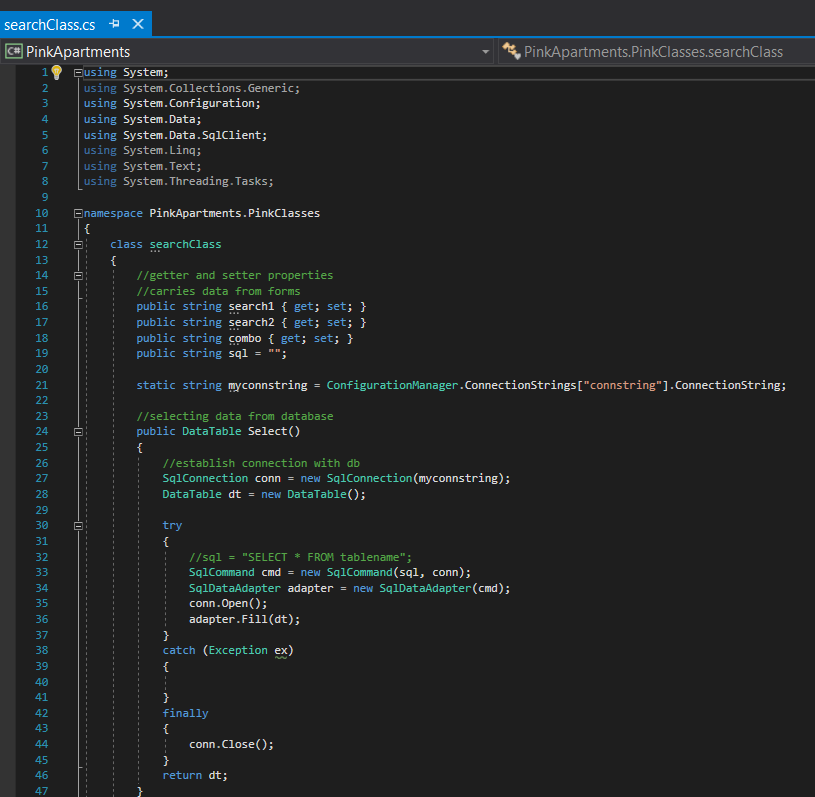
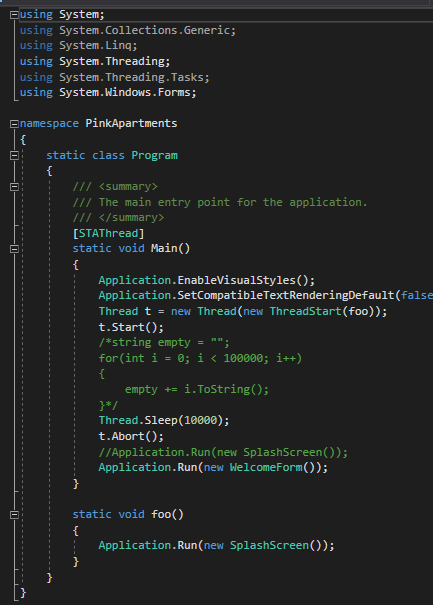
**Code Snippets**

In order to run this application on a different machine, the connectionString in the above snippet will need to be changed accordingly. This connection string can be found in MS SQL Server OR in Visual Studio after connecting to the database.







This snippet shows search security for our search form. Users will not be able to add, delete, drop, update, or create from the window’s application.

**SQL Coding Standards**

During this project, team members will NOT:

* Use CamelCase naming conventions as this makes it difficult to scan over the query in a quick manner
* Use prefixes such as res\_pet or res\_vehicle
* Use plurals
* Enforce Object Oriented Design principles

During this project, naming conventions include:

* Ensuring uniqueness of names
* Capitalizing all reserved keywords (SELECT instead of select)
* Only use letters, numbers, and keep underscores to a minimum
* Avoid abbreviations
* Use singular name for columns
* DO NOT use ID for a primary key name in ANY table
* DO NOT name a column the same name as the table
* DO NOT concatenate table names

Aliasing

* Use the AS keyword when aliasing
* If using multiple aliases for one table/column/attribute, append a number to the first letter of that entity

Querying

* Use spaces before and after equals(=) and commas (,) and when using parentheses (‘’)
* Include spaces/newlines before AND/OR
* Use indentation following the SELECT statement for any WHERE, FROM, or GROUP BY clauses

**Functional Requirements**

* The application must function in a way that provides benefits to leasing office administrators at Pink Plateau Paradise Apartments.
* The search feature will allow end users to choose from an assortment of views or make use of the advanced search functionality by providing actual SQL queries.
* The create contracts feature will allow users to generate contracts for new residents. The create contracts feature will be a step-wise process. A resident must first be created, then the user will provide details regarding the contract, including the apartment number, start date, end date, base rent, so and so forth.
* The manage pets and vehicles feature will allow for the creation and updating of pets and vehicles and to assign both to a specific resident.
* The manage employees feature will allow for the creation and updating of employees.
* The manage residents feature will allow for the creation of a new resident and updating of residents. This feature will allow the user to relate an employee to a resident if said resident is an employee for the apartment complex.
* The apartments table will never be directly accessed by end users. If changes need to be made to the apartment table in the future, users will have to bring in outside resources (IT team) from the property management company.
* The payments table will never be directly accessed, as this table will be used by a futuristic resident portal.
* The search feature will check for malicious queries, such as attempts to manipulate data within the database through SQL injections. This security feature will be tested accordingly.
* The application will be built using Visual Studio 2017, C#, VB to create an Windows form application.
* The database will be built using a script that is hand engineered from a data model. The database will be implemented in Microsoft SQL Server 2017.

**Future Opportunities**

* Create a resident portal for residents to request events, pet additions/deletions, vehicle additions/deletions, and payment viewing/actioning.
* Streamline advanced search functionality to perform searches based off keywords instead of SQL queries.
* Create a web-based application so that data can be accessed online versus on a local machine. This would bring security risks, which would need proper assessment before implementation. This would be necessary for a customer-facing (resident) portal.

**References**

Complete YouTube playlist tutorial for creating application: <https://www.youtube.com/watch?v=ApPWGEKFnU4&list=PLBLPjjQlnVXVLh8nQWR9M5ADH1iZ6cX5D>

Make a foreign key a primary key (SQL):

<https://stackoverflow.com/questions/16686575/how-to-make-a-foreign-key-a-primary-key-in-sql-server>

Tips on SQL Coding Standards:

<https://www.sqlstyle.guide/>