Syracuse University
MSADS Program

Interior Design Service Database Project

Fall 2021 IST 659 Section M402

Nora Lin

Background:

Company X is an interior design firm based in NYC that services the five boroughs. Customers come to Company X with a home or several homes that they would like to have interior designed. Each customer's home can include one or several rooms that the designer employed by Company X will design. The current set up lends to multiple projects getting lost in Monday.com or designer's personal notebooks. Without a database, Company X has been running into issues regarding project tracking and payment processing which has turned several customer relationship sour and invoice mix-ups.

This database is intended to provide a cohesive and exclusive environment for customers to view which room(s) in which home(s) the design service is ordered for. In turn, designers will be able to keep track of the various projects they are assigned to. Status attributes will allow both the owners and any stakeholders to view what portion of projects were completed on schedule from the estimated end date as well as what portion of room budget were over or within budget. We propose this database in five parts.

- **Part 1:** introduce the stakeholders, a data glossary, list of business rules, and some quantitative business questions that this database can answer.
- **Part 2:** displays the proposed conceptual and normalized logical models
- **Part 3:** displays the physical database designs
- **Part 4:** maintenance forms using Microsoft Access
- **Part 5:** reports using Microsoft Access
- **Part 6:** data creation and manipulation with answers to previously posed quantitative questions
- **Part 7:** reflection and summary

Our proposal outcomes expectations are that Company X will be able to review this proposal and consider its implementation based on the variety of both financial and customer relation benefits outlined.

Part 1

Stakeholder List:

The stakeholders include the owners, investors, customers, and employed designers. The implementation of this database will not only provide easy of usability for customers and designers to track the various interior design projects, but will also provide financial updates to investors.

The overall profits of the busines as well as profits by home type and room type will instrumental for making decisions about the business' future trajectory. Through this database, owners will also be able to see which designers complete high budget projects or deadline driven projects which creates opportunities for promotions and increased scale. Customers with multiple homes can easily track the progress of the design process. Lastly, designers will be able to analyze their project due date accuracy based on home type or room type to make improvements for increased precision in projected project completion dates.

Glossary:

- A **customer** is a person who places an order for interior design services by Company X. Customer information includes the customer's name, email address, and phone number.
- A **designer** is a person employed by Company X. Designer information includes the designer's name and phone number.
- A **designer payment** are earnings that a designer has earned for completing a service. Designer payment information includes the amount, the payment date, and which designer earned the payment.
- A **designer payment status** updates whether a designer has been paid for their work. Designer payment status can be *paid*, which means the designer has received the earnings, or *unpaid*, which means the earnings have not been received.
- A **project** is the interior design service(s) that is ordered by a customer and designed by a designer. Project information includes the start date, estimated end date, the actual end date, the customer who ordered the project, and the designer who is assigned to the project.
- A **project status** updates whether a project is completed before the designer's estimated end date or after the designer's estimated end date. A project status can be *before estimated end* date or *after estimated end date*.
- A **home** is a physical house/apartment/studio/loft that belongs to paying customer. Home information includes the home address, home type, and which project the home is included in.

- A **home type** updates whether a customer's home is an Apartment, Townhouse or House.
- A **room** is a physical space within a customer's home. Room information includes the room type, actual room budget, estimated room budget, room budget status, and which home the room is contained in.
- A **room type** updates whether a room is a Bathroom, Bedroom, Living room, Kitchen or Study.
- A **room budget status** updates whether a room's actual budget is over or within its estimated budget. A room budget status can be *over*, which means that the room's actual budget is greater than the room's estimated budget, or *within*, which means that the room's actual budget is either less than or equal to the room's estimated budget.
- An **invoice** includes a financial statement of service costs that the customer pays Company X after the design services are complete. An invoice information includes the amount, the invoice status, and which project the invoice was created by.
- An **invoice status** updates whether a customer has paid for the design services. Invoice status can be *paid*, which means Company X has received the payment, or *unpaid*, which means Company X has not received the payment.

Business Rules:

- A customer can place many projects
- A project must be placed by one customer.
- A project must only have one designer.
- A designer can design multiple projects.
- A project can include multiple homes.
- A home must be included in only one project.
- A home can contain multiple rooms
- A room must be contained in only one home.
- A home must be of only one home type.
- A home type can be for multiple homes.
- A room must be of only one room type.

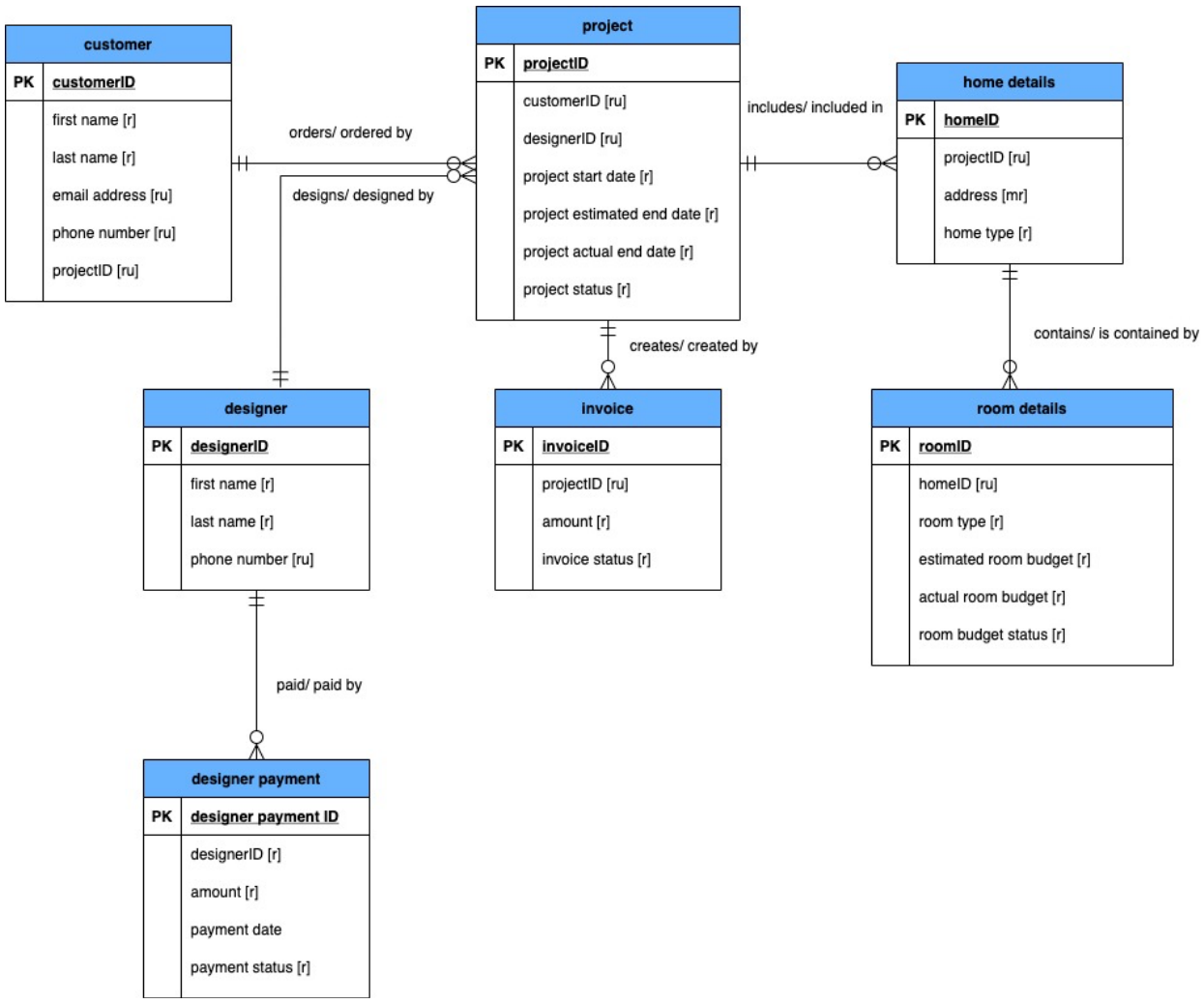
- A room type can be for multiple rooms.
- A designer can be paid in multiple payments.
- A designer payment is paid to only one designer.
- A project can create multiple invoices.
- An invoice is created by only one project.
- An invoice status updates the invoice.
- A project status updates the project.

Quantitative Data Questions:

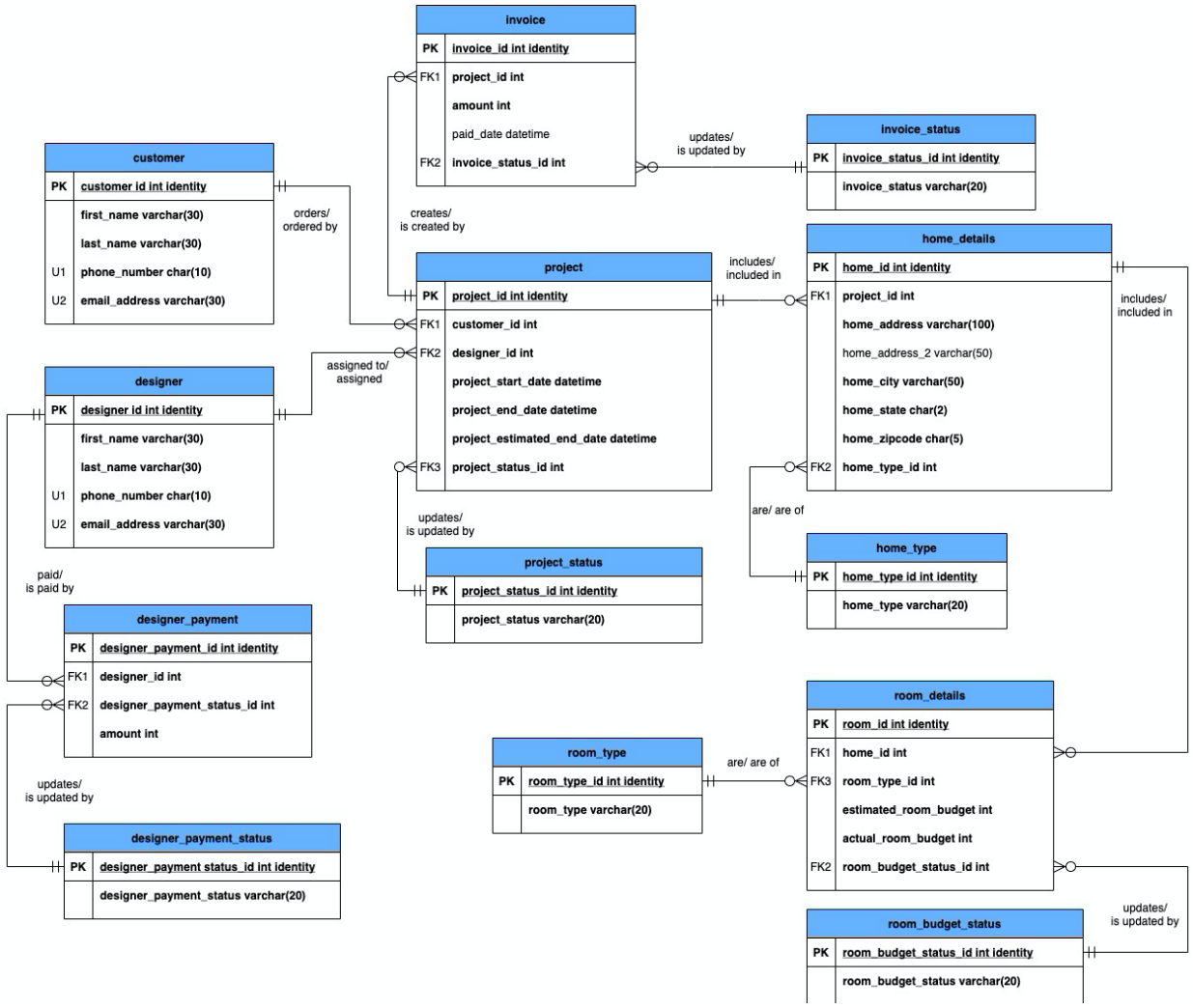
- 1) What the number of homes per project ?
- 2) What is average room budget for each type of room ?
- 3) What is the number of homes in the design process by Zipcode ?
- 4) What portion of projects are completed before project's estimated end date? What portion of designers complete their assigned projects before the estimated end date?
- 5) What percentage of rooms are within budget or overbudget?

Part 2

Conceptual Model:



Normalized Logical Model:



Part 3

Physical Database Design:

```
--Drop tables in reverse order of their dependencies:
DROP TABLE IF EXISTS room_details
GO

DROP TABLE IF EXISTS room_budget_status;
GO

DROP TABLE IF EXISTS room_type;
GO

DROP TABLE IF EXISTS home_details;
GO

DROP TABLE IF EXISTS home_type;
GO

DROP TABLE IF EXISTS invoice;
GO

DROP TABLE IF EXISTS invoice_status;
GO

DROP TABLE IF EXISTS project;
GO

DROP TABLE IF EXISTS project_status;
GO

DROP TABLE IF EXISTS designer_payment;
GO

DROP TABLE IF EXISTS designer_payment_status;
GO

DROP TABLE IF EXISTS designer;
GO

DROP TABLE IF EXISTS customer;
GO

--drop all procedures
DROP PROCEDURE IF EXISTS ChangeAptNumber;
GO

DROP PROCEDURE IF EXISTS UpdateDesignerPaid;
GO

--drop all functions:
DROP FUNCTION IF EXISTS ZipcodeToBorough;
```

```
GO

DROP FUNCTION IF EXISTS CustomerHomesCounts;
GO

DROP FUNCTION IF EXISTS ZipCodeCounts;
GO

DROP FUNCTION IF EXISTS HomeCounts;
GO

DROP FUNCTION IF EXISTS RoomCounts;
GO

DROP FUNCTION IF EXISTS RoomAverages;
GO

--drop all views
DROP VIEW IF EXISTS CustomerDesigner;
GO

DROP VIEW IF EXISTS ProjectsOverEstimate;
GO

DROP VIEW IF EXISTS DesignerPayment;
GO

DROP VIEW IF EXISTS NumberHomesPerZipCode;
GO

DROP VIEW IF EXISTS CustomerDesignerHome;
GO

DROP VIEW IF EXISTS RoomsBudget;
GO

DROP VIEW IF EXISTS HomeAndRooms;
GO

DROP VIEW IF EXISTS InvoiceUnpaided;
GO

--finished dropping tables, views, procedures and functions
--Creating tables in order of their dependencies:

--customer table:
CREATE TABLE customer(
        --columns
        customer_id int identity,
        first_name varchar(30) not null,
        last_name varchar(30) not null,
        phone_number char(10) not null,
        email_address varchar(30) not null,
        --constraints
        CONSTRAINT PK_customer PRIMARY KEY (customer_id),
```

```
        CONSTRAINT U1_customer UNIQUE (phone_number),
        CONSTRAINT U2_customer UNIQUE (email_address)
        );
GO
--end

--designer table:
        CREATE TABLE designer(
        --columns
        designer_id int identity,
        first_name varchar(30) not null,
        last_name varchar(30) not null,
        phone_number varchar(10) not null,
        email_address varchar(30) not null,
        --constraints
        CONSTRAINT PK_designer PRIMARY KEY (designer_id),
        CONSTRAINT U1_designer UNIQUE (phone_number),
        CONSTRAINT U2_designer UNIQUE (email_address)
        );
GO
--end

--desinger_payment_status table:
CREATE TABLE designer_payment_status(
        --columns
        designer_payment_status_id int identity,
        designer_payment_status varchar(20) not null,
        --constraints
        CONSTRAINT PK_designer_payment_status PRIMARY KEY (designer_payment_status_id)
        );
GO
--end

--designer_payment table:
CREATE TABLE designer_payment(
        --columns
        designer_payment_id int identity,
        designer_id int not null,
        designer_payment_status_id int not null,
        amount int not null,
        --constraints
        CONSTRAINT PK_designer_payment PRIMARY KEY (designer_payment_id),
        CONSTRAINT FK1_designer_payment FOREIGN KEY (designer_id) REFERENCES
        designer(designer_id),
        CONSTRAINT FK2_designer_payment FOREIGN KEY (designer_payment_status_id) REFERENCES
        designer_payment_status(designer_payment_status_id)
        );
GO
--end

--project_status table:
CREATE TABLE project_status(
        --columns
        project_status_id int identity,
        project_status varchar(50) not null,
```

```
        --constraints
        CONSTRAINT PK_project_status PRIMARY KEY (project_status_id)
        );
GO
--end

--project table:
CREATE TABLE project(
        --columns
        project_id int identity,
        customer_id int not null,
        designer_id int not null,
        project_start_date date not null,
        project_end_date date not null,
        project_estimated_end_date date not null,
        project_status_id int not null,
        --constraints
        CONSTRAINT PK_project PRIMARY KEY (project_id),
        CONSTRAINT FK1_project FOREIGN KEY (customer_id) REFERENCES customer(customer_id),
        CONSTRAINT FK2_project FOREIGN KEY (designer_id) REFERENCES designer(designer_id),
        CONSTRAINT FK3_project FOREIGN KEY (project_status_id) REFERENCES
        project_status(project_status_id)
        );
GO
--end

--invoice_status table:
CREATE TABLE invoice_status(
        --columns
        invoice_status_id int identity,
        invoice_status varchar(20) not null,
        --constraints
        CONSTRAINT PK_invoice_status PRIMARY KEY (invoice_status_id)
        );
GO
--end


-- invoice table:
CREATE TABLE invoice(
        --columns
        invoice_id int identity,
        project_id int not null,
        invoice_amount int not null,
        invoice_status_id int not null,
        --constraints
        CONSTRAINT PK_invoice PRIMARY KEY (invoice_id),
        CONSTRAINT FK1_invoice FOREIGN KEY (project_id) REFERENCES project(project_id),
        CONSTRAINT FK2_invoice FOREIGN KEY (invoice_status_id) REFERENCES
        invoice_status(invoice_status_id)
        );
GO
--end

--home_type table:
```

```
CREATE TABLE home_type(
	--columns
	home_type_id int identity,
	home_type varchar(20) not null,
	--constraints
	CONSTRAINT PK_home_type PRIMARY KEY (home_type_id)
	);
GO
--end

--home_details table:
CREATE TABLE home_details(
	--columns
	home_id int identity,
	project_id int not null,
	home_address varchar(100) not null,
	home_address_2 varchar(50),
	home_city varchar(50) not null,
	home_state char(2) not null,
	home_zipcode char(5) not null,
	home_borough varchar(20),
	home_type_id int not null,
	--constraints
	CONSTRAINT PK_home_details PRIMARY KEY (home_id),
	CONSTRAINT FK1_home_details FOREIGN KEY (project_id) REFERENCES project(project_id),
	CONSTRAINT FK2_home_details FOREIGN KEY (home_type_id) REFERENCES
	home_type(home_type_id)
	);
GO
--end

-- room_budget_status table:
CREATE TABLE room_budget_status(
	--columns
	room_budget_status_id i int identity,
	room_budget_status varchar(20) not null,
	--constraints
	CONSTRAINT PK_room_budget_status PRIMARY KEY (room_budget_status_id)
	);
GO
--end

-- room_budget_status table:
CREATE TABLE room_type(
	--columns
	room_type_id int identity,
	room_type varchar(20) not null,
	--constraints
	CONSTRAINT PK_room_type PRIMARY KEY (room_type_id)
	);
GO
--end

--room_details table:
CREATE TABLE room_details(
```

```
	--columns
	room_id int identity,
	home_id int not null,
	room_type_id int not null,
	estimated_room_budget int not null,
	actual_room_budget int not null,
	room_budget_status_id int not null,
	--constraints
	CONSTRAINT PK_room_details PRIMARY KEY (room_id),
	CONSTRAINT FK1_room_details FOREIGN KEY (home_id) REFERENCES home_details(home_id),
	CONSTRAINT FK2_room_details FOREIGN KEY (room_budget_status_id) REFERENCES
	room_budget_status(room_budget_status_id),
	CONSTRAINT FK3_room_details FOREIGN KEY (room_type_id) REFERENCES
	room_type(room_type_id)
	);
GO
--end
--finished creating all tables

--inserting customer records
INSERT INTO customer(first_name,last_name,phone_number,email_address)
VALUES('Olivia','Jade','2128570526', 'oliviajade@gmail.com'),
	('Liam','Cohen', '2124880933', 'liamcohen@gmail.com'),
	('Amelia','Morgan', '2127773339','ameliamorgan@gmail.com'),
	('Mia','David','2124438043','miadavid@gmail.com'),
	('Levi','Stanley','2123305268','levistanley@gmail.com'),
	('Teddy','Yang','2128883450','teddyyang@gmail.com')
--looking at customers
SELECT *
FROM customer
--end
```

	customer_id	first_name	last_name	phone_number	email_address
1	1	Olivia	Jade	2128570526	oliviajade@gmail.com
2	2	Liam	Cohen	2124880933	liamcohen@gmail.com
3	3	Amelia	Morgan	2127773339	ameliamorgan@gmail.com
4	4	Mia	David	2124438043	miadavid@gmail.com
5	5	Levi	Stanley	2123305268	levistanley@gmail.com
6	6	Teddy	Yang	2128883450	teddyyang@gmail.com

```
--inserting designer records:
INSERT INTO designer(first_name,last_name,phone_number,email_address)
VALUES ('Ellie','Adams','2125839032','jane.adams@companyx.com'),
		('Mary','Jones','2125839031', 'mary.jones@companyx.com'),
		('Christine','Smith','2125839033', 'christine.smith@companyx.com'),
		('Bill','Taylor','2125839034','bill.taylor@companyx.com')
--looking at designers
SELECT *
FROM designer
```

--end

	designer_id	first_name	last_name	phone_number	email_address
1	1	Ellie	Adams	2125839032	jane.adams@companyx.com
2	2	Mary	Jones	2125839031	mary.jones@companyx.com
3	3	Christine	Smith	2125839033	christine.smith@companyx.com
4	4	Bill	Taylor	2125839034	bill.taylor@companyx.com

```
--creating records for project_status:
--filling in invoice_status choices: paid or unpaid
INSERT INTO project_status(project_status)
VALUES ('Within Estimated End Date'), ('Over Estimated End Date')
--identifying id numbers for paid and unpaid invoice status
SELECT *
FROM project_status
--invoice_status = 1 = Within Estimated End Date
--invoice_status = 2 = Over Estimated End Date
--end

--inserting project records:
INSERT INTO
project(customer_id,designer_id,project_start_date,project_end_date,project_estimated_end_date,project_
status_id)
VALUES((SELECT customer_id FROM customer WHERE email_address= 'oliviajade@gmail.com'),
(SELECT designer_id FROM designer WHERE
phone_number='2125839031'),'2/15/2016','2/1/2018','1/1/2018','1'),
((SELECT customer_id FROM customer WHERE email_address= 'liamcohen@gmail.com'),
(SELECT designer_id FROM designer WHERE
phone_number='2125839032'),'2/17/2018','2/1/2019','1/1/2019','1'),
((SELECT customer_id FROM customer WHERE email_address= 'ameliamorgan@gmail.com'),
(SELECT designer_id FROM designer WHERE
phone_number='2125839031'),'6/30/2018','7/21/2019','7/21/2019','1'),
((SELECT customer_id FROM customer WHERE email_address= 'miadavid@gmail.com'),
(SELECT designer_id FROM designer WHERE
phone_number='2125839034'),'11/3/2018','4/5/2019','3/29/2018','2'),
((SELECT customer_id FROM customer WHERE email_address= 'levistanley@gmail.com'),
(SELECT designer_id FROM designer WHERE
phone_number='2125839033'),'10/11/2017','11/5/2019','12/5/2019','2'),
((SELECT customer_id FROM customer WHERE email_address= 'teddyyang@gmail.com'),
(SELECT designer_id FROM designer WHERE
phone_number='2125839034'),'08/6/2017','10/6/2018','10/5/2018','1')

--looking at project:
SELECT *
FROM project
```

```
--Now, lets create a view so that we can see the Customer Name and Designer Names within the projects records:

GO
CREATE VIEW CustomerDesigner AS
SELECT
        customer.first_name + ' ' +  customer.last_name AS CustomerName,
        designer.first_name + ' ' + designer.last_name AS DesignerName
FROM project
RIGHT JOIN designer ON project.designer_id=designer.designer_id
JOIN customer ON project.customer_id=customer.customer_id;
GO

--let's take a look at the view:
SELECT *
FROM CustomerDesigner
```

	CustomerName	DesignerName
1	Olivia Jade	Mary Jones
2	Liam Cohen	Ellie Adams
3	Amelia Morgan	Mary Jones
4	Mia David	Bill Taylor
5	Levi Stanley	Christine Smith
6	Teddy Yang	Bill Taylor

```
--that's pretty cool!

--filling in records for designer_payment_status choices: paid or unpaid
INSERT INTO designer_payment_status(designer_payment_status)
VALUES ('PAID'), ('UNPAID')

-- identifying id numbers for paid and upaid designer payment status:
SELECT *
FROM designer_payment_status
--designer_payment_status_id = 1 = PAID
--designer_payment_status_id = 2 = UNPAID
--end

--filling in records for designer payment information:
INSERT INTO designer_payment(designer_id,designer_payment_status_id,amount)
VALUES ((SELECT designer_id FROM designer WHERE phone_number = '2125839032'), '1','10000'),
                ((SELECT designer_id FROM designer WHERE phone_number = '2125839031'), '1',
                '15000'),
                ((SELECT designer_id FROM designer WHERE phone_number = '2125839033'), '2', '9000'),
((SELECT designer_id FROM designer WHERE phone_number = '2125839034'), '1', '12000')

--looking at designer payment information
SELECT *
FROM designer_payment
--end
```

```
--lets create another view so we can see which designers have been paid and how much:
GO
CREATE VIEW DesignerPayment AS
SELECT
        designer_payment.designer_payment_id AS DesignerPaymentID,
        designer.first_name + ' ' + designer.last_name AS DesignerName,
        designer_payment_status.designer_payment_status AS Status
FROM designer_payment
RIGHT JOIN designer ON designer_payment.designer_id=designer.designer_id
JOIN designer_payment_status ON
designer_payment_status.designer_payment_status_id=designer_payment.designer_payment_status_id
GO

SELECT *
FROM DesignerPayment
```

	DesignerPaymentID	DesignerName	Status
1	1	Ellie Adams	PAID
2	2	Mary Jones	PAID
3	3	Christine Smith	UNPAID
4	4	Bill Taylor	PAID

```
--let's make a procedure that updates that when designer have been paid:
GO
CREATE PROCEDURE UpdateDesignerPaid(@designer_ID int) AS
BEGIN
        UPDATE designer_payment SET designer_payment_status_id=1
        WHERE designer_id = @designer_ID
END

--Let's update Christine's record to paid
GO
DECLARE @paiddesigner int
SET @paidDesigner = (SELECT designer_id FROM designer WHERE designer.first_name='Christine')
EXEC UpdateDesignerPaid @paiddesigner

--Let's check christine's updated file
SELECT *
FROM DesignerPayment
WHERE DesignerName='Christine Smith'
```

	DesignerPaymentID	DesignerName	Status
1	3	Christine Smith	PAID

```
--STATUS tables:
--filling in room_budget_status choices: over or within
INSERT INTO room_budget_status(room_budget_status)
VALUES ('WITHIN'), ('OVER')

--identifying id numbers for WITHIN and OVER:
SELECT * FROM room_budget_status
```

```
--room_budget_status_ID = 1 = WITHIN
--room_budget_status_ID = 2 = OVER
--end

--filling in invoice_status choices: paid or unpaid
INSERT INTO invoice_status(invoice_status)
VALUES ('PAID'), ('UNPAID')

--identifying id numbers for paid and unpaid invoice status
SELECT * FROM invoice_status
--invoice_status = 1 = PAID
--invoice_status = 2 = UNPAID
--end

--filling in invoice information:
INSERT INTO invoice(project_id,invoice_amount,invoice_status_id)
VALUES ('1', '400000','1'),('2','100000','2'),('3','300000','1'),
('4', '78000','1'), ('5','520000','2'), ('6', '37000','1')

--looking at invoice table
SELECT *
FROM invoice

--filling in home types information:
INSERT INTO  home_type(home_type)
VALUES ('Apartment'),('House'),('Townhouse')

--looking at home_type table
SELECT *
FROM home_type

--filling in home information:
INSERT INTO
home_details(project_id,home_address,home_address_2,home_city,home_state,home_zipcode,home_typ
e_id)
VALUES ('1','860 5th Ave','APT 514','New York','NY','10065','1'),
        ('1','10-19 47th St',Null,'Long Island City','NY','11101','2'),
        ('1','175 Kent Ave','APT 222','Brooklyn','NY','11249','1'),
        ('2','162 W 81 St',Null,'New York','NY','10024','3'),
        ('2','172 W 81 St','APT 317','New York','NY','10024','1'),
        ('3','12 Eckford St','APT 806','Brooklyn','NY','11222','1'),
        ('3','89 Murrary St','APT 1207','New York','NY','10007','1'),
        ('4','128 E 74th St',Null,'New York','NY','10021','3'),
        ('5','500 W 56th St','APT 112','New York','NY','10019','1'),
        ('5','24 Central Park S',Null,'New York','NY','10019','3'),
        ('6','985 5th Ave','Penthouse','New York','NY','10075','1')
SELECT *
FROM home_details
```

--Let's create a view to see Customer Designer and Home information:

	CustomerName	DesignerName	HomeAddress	Address2	City	State	ZipCode	HomeType
1	Olivia Jade	Mary Jones	860 5th Ave	APT 514	New York	NY	10065	Apartment
2	Olivia Jade	Mary Jones	10-19 47th St	NULL	Long Island City	NY	11101	House
3	Olivia Jade	Mary Jones	175 Kent Ave	APT 222	Brooklyn	NY	11249	Apartment
4	Liam Cohen	Ellie Adams	162 W 81 St	NULL	New York	NY	10024	Townhouse
5	Liam Cohen	Ellie Adams	172 W 81 St	APT 317	New York	NY	10024	Apartment
6	Amelia Morgan	Mary Jones	12 Eckford St	APT 806	Brooklyn	NY	11222	Apartment
7	Amelia Morgan	Mary Jones	89 Murrary St	APT 1207	New York	NY	10007	Apartment
8	Mia David	Bill Taylor	128 E 74th St	NULL	New York	NY	10021	Townhouse
9	Levi Stanley	Christine Smith	500 W 56th St	APT 112	New York	NY	10019	Apartment
10	Levi Stanley	Christine Smith	24 Central Park S	NULL	New York	NY	10019	Townhouse
11	Teddy Yang	Bill Taylor	985 5th Ave	Penthouse	New York	NY	10075	Apartment

GO

```
CREATE VIEW CustomerDesignerHome AS
SELECT
        customer.first_name + ' ' + customer.last_name AS CustomerName,
        designer.first_name + ' ' + designer.last_name AS DesignerName,
        home_details.home_address AS HomeAddress,
        home_details.home_address_2 AS Address2,
        home_details.home_city AS City,
        home_details.home_state AS State,
        home_details.home_zipcode AS ZipCode,
        home_type.home_type AS HomeType
FROM home_details
RIGHT JOIN project ON project.project_id=home_details.project_id
JOIN customer ON customer.customer_id=project.customer_id
JOIN designer ON designer.designer_id=project.designer_id
JOIN home_type ON home_details.home_type_id=home_type.home_type_id
GO

SELECT *
FROM CustomerDesignerHome
```

	CustomerName	DesignerName	HomeAddress	Address2	City	State	ZipCode	HomeType
1	Olivia Jade	Mary Jones	860 5th Ave	APT 514	New York	NY	10065	Apartment
2	Olivia Jade	Mary Jones	10-19 47th St	NULL	Long Island City	NY	11101	House
3	Olivia Jade	Mary Jones	175 Kent Ave	APT 222	Brooklyn	NY	11249	Apartment
4	Liam Cohen	Ellie Adams	162 W 81 St	NULL	New York	NY	10024	Townhouse
5	Liam Cohen	Ellie Adams	172 W 81 St	APT 317	New York	NY	10024	Apartment
6	Amelia Morgan	Mary Jones	12 Eckford St	APT 806	Brooklyn	NY	11222	Apartment
7	Amelia Morgan	Mary Jones	89 Murrary St	APT 1207	New York	NY	10007	Apartment
8	Mia David	Bill Taylor	128 E 74th St	NULL	New York	NY	10021	Townhouse
9	Levi Stanley	Christine Smith	500 W 56th St	APT 112	New York	NY	10019	Apartment
10	Levi Stanley	Christine Smith	24 Central P...	NULL	New York	NY	10019	Townhouse
11	Teddy Yang	Bill Taylor	985 5th Ave	Penthou...	New York	NY	10075	Apartment

```
--Creating Procedure for updating Apartment Number
--Let's say that one of the customers entered in the wrong Apartment number to their home on 500 W 56th.
--She entered the apartment number as Apt 112  but it actually is 212
SELECT *
FROM home_details
WHERE home_address='500 W 56th St'
```

	home_id	project_id	home_address	home_address_2	home_city	home_state	home_zipcode	home_type_id
1	9	5	500 W 56th St	APT 112	New York	NY	10019	1

```
GO
CREATE PROCEDURE ChangeAptNumber(@home_address varchar(100),@newaptnumber varchar(20))
AS
BEGIN
	UPDATE home_details SET home_address_2= @newaptnumber
	WHERE home_address = @home_address
END

--lets use this procedure to correct Amelia's mistake
EXEC ChangeAptNumber'500 W 56th St','Apt 212'
--checking to see if it worked
SELECT *
FROM home_details
WHERE home_address='500 W 56th St'
```

	home_id	project_id	home_address	home_address_2	home_city	home_state	home_zipcode	home_type_id
1	9	5	500 W 56th St	Apt 212	New York	NY	10019	1

```
--Now let's create a view that allows us to view which customer has not completed their invoice and by how much:
GO
CREATE VIEW InvoiceUnPaided AS
SELECT
	customer.first_name + ' ' + customer.last_name AS CustomerName,
	invoice.invoice_amount AS Amount,
	invoice_status.invoice_status AS Status
FROM invoice
RIGHT JOIN project ON project.project_id=invoice.project_id
JOIN customer ON customer.customer_id=project.customer_id
JOIN invoice_status ON invoice_status.invoice_status_id=invoice.invoice_status_id
GO

--Now let's view which customers have unpaided invoices
SELECT *
FROM InvoiceUnPaided
WHERE status='UNPAID'
```

	CustomerName	Amount	Status
1	Liam Cohen	100000	UNPAID
2	Levi Stanley	520000	UNPAID

```
--filling in room types information:
INSERT INTO  room_type(room_type)
VALUES ('Living Room'),('Bedroom'),('Bathroom'),('Study'),('Kitchen')
SELECT *
FROM room_type

--filling in room_details
INSERT INTO
room_details(home_id,room_type_id,estimated_room_budget,actual_room_budget,room_budget_status_i
d)
VALUES ((SELECT home_id FROM home_details WHERE home_address='860 5th
Ave'),'1','100000','120000','2'),
((SELECT home_id FROM home_details WHERE home_address='860 5th Ave'),'2','90000','90000','1'),
((SELECT home_id FROM home_details WHERE home_address='10-19 47th St'),'5','30000','40000','2'),
((SELECT home_id FROM home_details WHERE home_address='175 Kent Ave'),'3','25000','23000','1'),
((SELECT home_id FROM home_details WHERE home_address='162 W 81 St'),'2','50000','49000','1'),
((SELECT home_id FROM home_details WHERE home_address='172 W 81 St'),'3','50000','51000','2'),
((SELECT home_id FROM home_details WHERE home_address='12 Eckford St'),'4','100000','98000','1'),
((SELECT home_id FROM home_details WHERE home_address='89 Murrary St'),'2','200000','202000','2'),
((SELECT home_id FROM home_details WHERE home_address='128 E 74th St'),'4','80000','78000','1'),
((SELECT home_id FROM home_details WHERE home_address='500 W 56th St'),'1','100000','120000','2'),
((SELECT home_id FROM home_details WHERE home_address='24 Central Park S'),'2','90000','100000','2'),
((SELECT home_id FROM home_details WHERE home_address='24 Central Park S'),'4','60000','55000','1'),
((SELECT home_id FROM home_details WHERE home_address='24 Central Park
S'),'5','245000','245000','1'),
((SELECT home_id FROM home_details WHERE home_address='985 5th Ave'),'4','36000','37000','2')

--looking at  room_detail table
SELECT *
FROM room_details;
GO

--let's create a view so that we can see the home address
--for each of these rooms and whether they were over or under budget
GO
CREATE VIEW HomeAndRooms AS
SELECT
        home_details.home_address AS HomeAddress,
        room_type.room_type AS RoomType,
        room_budget_status.room_budget_status AS  BudgetStatus
FROM room_details
RIGHT JOIN home_details ON home_details.home_id=room_details.home_id
JOIN room_budget_status ON
room_budget_status.room_budget_status_id=room_details.room_budget_status_id
JOIN room_type ON room_type.room_type_id=room_details.room_type_id
GO

--Looking at the view:
SELECT *
FROM HomeAndRooms
WHERE BudgetStatus='OVER' ;
GO
```

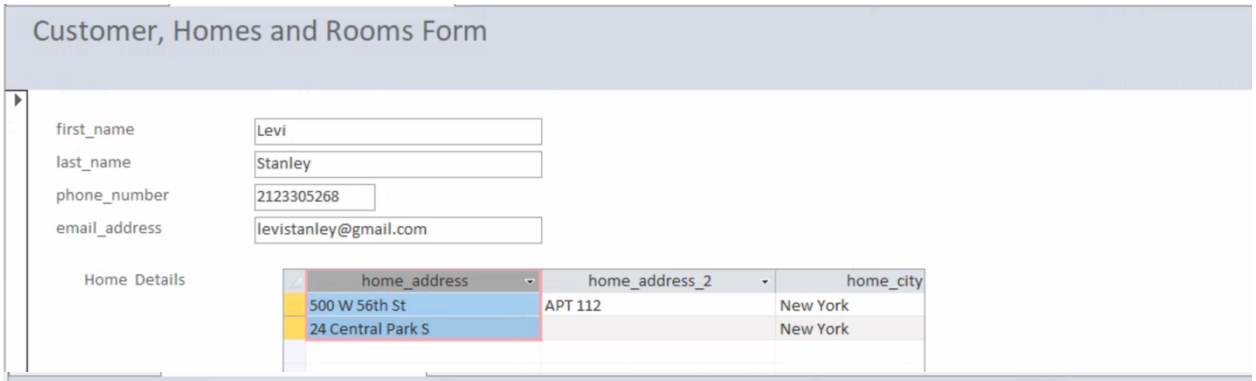
	HomeAddress	RoomType	BudgetStatus
1	860 5th Ave	Living Room	OVER
2	10-19 47th St	Kitchen	OVER
3	172 W 81 St	Bathroom	OVER
4	89 Murrary St	Bedroom	OVER
5	500 W 56th St	Living Room	OVER
6	24 Central Park S	Bedroom	OVER
7	985 5th Ave	Study	OVER

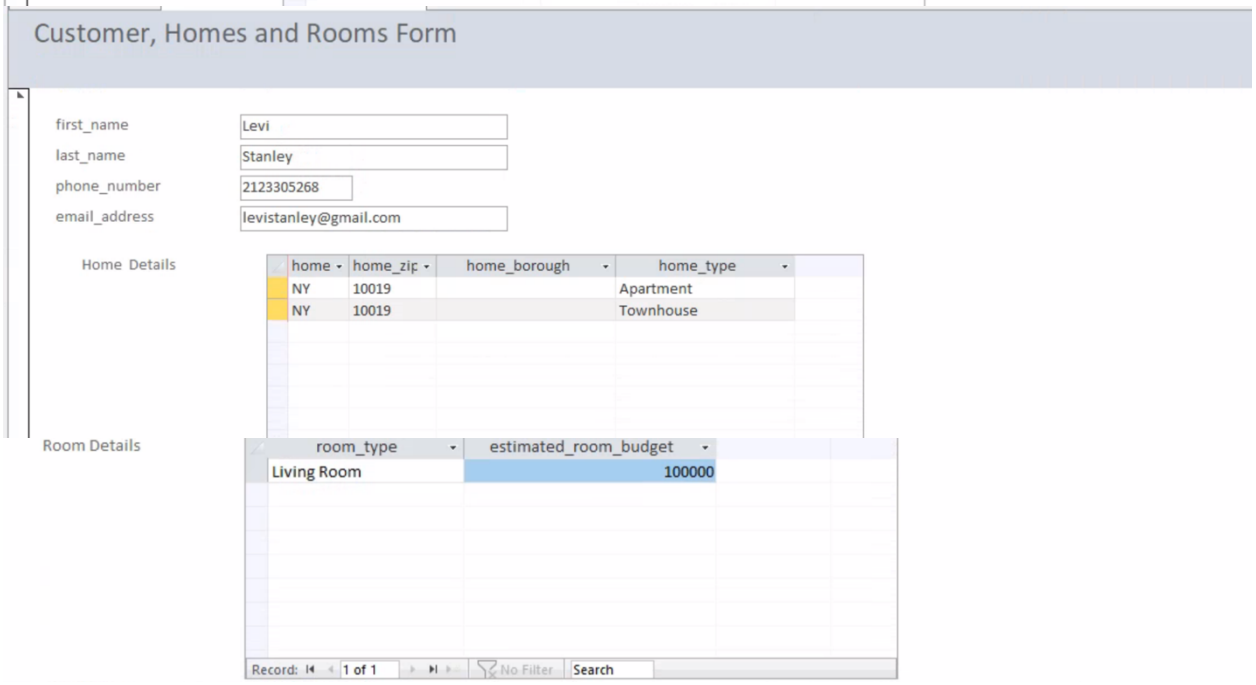
```
--Let's create a table that has the different type of homes and the counts for each type:
GO
CREATE FUNCTION HomeCounts(@home_type_id int)
RETURNS int AS
BEGIN
	DECLARE @returnValue int
	SELECT @returnValue=COUNT(home_type_id) FROM home_details
	WHERE home_details.home_type_id=@home_type_id
	RETURN @returnValue
END
GO

SELECT
	home_type.home_type AS HomeType,
	[dbo].[HomeCounts](home_type_id) AS HomeTypeCounts
FROM home_type
GROUP BY home_type
ORDER BY HomeTypeCounts DESC;
GO
```

	HomeID	HomeType	HomeTypeCounts
1	1	Apartment	7
2	3	Townhouse	3
3	2	House	1

Part 4

Form Created Using Microsoft Access:





Part 5

Reports Created Using Microsoft Access:

Designer Payment

Designer Name	Status
Ellie Adams	PAID
Mary Jones	PAID
Christine Smith	PAID
Bill Taylor	PAID

Invoice UnPaided

Customer Name	Amount	Status
Olivia Jade	400000	PAID
Liam Cohen	100000	UNPAID
Amelia Morgan	300000	PAID
Mia David	78000	PAID
Levi Stanley	520000	UNPAID
Teddy Yang	37000	PAID

Customer Designer Project Details

Customer Name	DesignerName	HomeAddress	Address 2	City	State	Zip Code	Home Type
Olivia Jade	Mary Jones	860 5th Ave	APT 514	New York	NY	10065	Apartment
Olivia Jade	Mary Jones	10-19 47th St		Long Islan	NY	11101	House
Olivia Jade	Mary Jones	175 Kent Ave	APT 222	Brooklyn	NY	11249	Apartment
Liam Cohen	Ellie Adams	162 W 81 St		New York	NY	10024	Townhouse
Liam Cohen	Ellie Adams	172 W 81 St	APT 317	New York	NY	10024	Apartment
Amelia Morgan	Mary Jones	12 Eckford St	APT 806	Brooklyn	NY	11222	Apartment
Amelia Morgan	Mary Jones	89 Murrary St	APT 1207	New York	NY	10007	Apartment
Mia David	Bill Taylor	128 E 74th St		New York	NY	10021	Townhouse
Levi Stanley	Christine Smith	500 W 56th St	APT 112	New York	NY	10019	Apartment
Levi Stanley	Christine Smith	24 Central Park S		New York	NY	10019	Townhouse
Teddy Yang	Bill Taylor	985 5th Ave	Penthous	New York	NY	10075	Apartment

Projects Over Estimate by Number of Days

Project ID	Designer Name	Project Status	Days Over Estimated End Date
1	Mary Jones	Before Estimated End Date	31
2	Ellie Adams	Before Estimated End Date	31
3	Mary Jones	Before Estimated End Date	0
4	Bill Taylor	Before Estimated End Date	372
5	Christine Smith	After Estimated End Date	-30
6	Bill Taylor	Before Estimated End Date	1

Rooms Budget

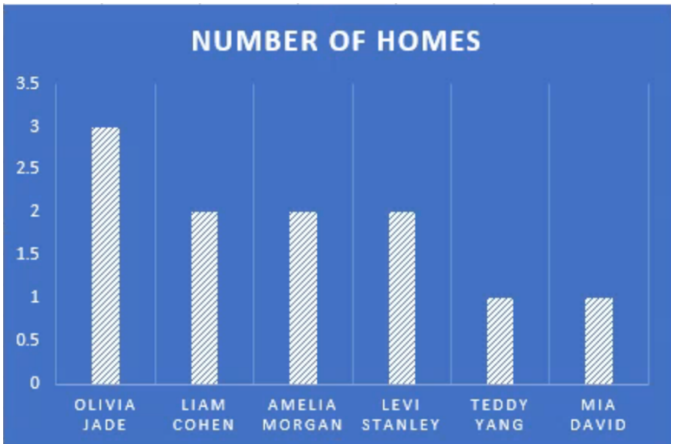
Designer Name	Budget Status	Room Type	Amount Difference between Estimated and Actual Budget
Mary Jones	OVER	Living Room	20000
Mary Jones	WITHIN	Bedroom	0
Mary Jones	OVER	Kitchen	10000
Mary Jones	WITHIN	Bathroom	-2000
Ellie Adams	WITHIN	Bedroom	-1000
Ellie Adams	OVER	Bathroom	1000
Mary Jones	WITHIN	Study	-2000
Mary Jones	OVER	Bedroom	2000
Bill Taylor	WITHIN	Study	-2000
Christine Smith	OVER	Living Room	20000
Christine Smith	OVER	Bedroom	10000
Christine Smith	WITHIN	Study	-5000
Christine Smith	WITHIN	Kitchen	0
Bill Taylor	OVER	Study	1000

Part 6

- Question 1: What the number of homes per project?

```
--Let's create a function to see how many homes each customer has work on:
GO
CREATE FUNCTION CustomerHomesCounts(@project_id int)
RETURNS int AS
BEGIN
    DECLARE @returnValue int
    SELECT @returnValue=COUNT(project_id) FROM home_details
    WHERE home_details.project_id= @project_id
    RETURN @returnValue
END
GO

--Let's view the results in a table:
GO
SELECT
    project.project_id AS ProjectID,
    customer.first_name+ ' ' + customer.last_name AS CustomerName,
    [dbo].[CustomerHomesCounts](project.project_id) AS NumberOfHomes
FROM project
RIGHT JOIN customer ON customer.customer_id=project.customer_id
JOIN home_details ON project.project_id=home_details.project_id
GROUP BY project.project_id, customer.first_name,customer.last_name
ORDER BY NumberOfHomes DESC;
GO
```

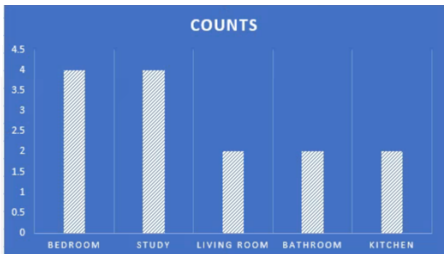
	ProjectID	CustomerName	NumberOfHomes
1	1	Olivia Jade	3
2	2	Liam Cohen	2
3	3	Amelia Morgan	2
4	5	Levi Stanley	2
5	6	Teddy Yang	1
6	4	Mia David	1

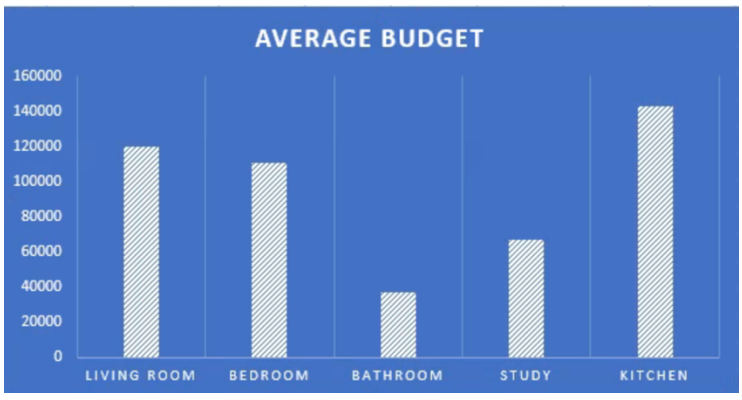


- Question 2: What is average room budget for each type of room?

```
--Let's create a table that has the different type of rooms and the counts for each type,
--and the average budget for each type of  room

GO
CREATE FUNCTION RoomCounts(@room_type_id int)
RETURNS int AS
BEGIN
    DECLARE @returnValue int
    SELECT @returnValue=COUNT(room_type_id) FROM room_details
    WHERE room_details.room_type_id=@room_type_id
    RETURN @returnValue
END
GO

GO
CREATE FUNCTION RoomAverages(@room_type_id int)
RETURNS int AS
BEGIN
    DECLARE @returnValue int
    SELECT @returnValue=AVG(actual_room_budget) FROM room_details
    WHERE room_details.room_type_id=@room_type_id
    RETURN @returnValue
END
GO

SELECT
    room_type.room_type AS RoomType,
    [dbo].[RoomCounts](room_type_id) AS Counts,
    [dbo].[RoomAverages](room_type_id) AS AverageBudget
FROM room_type
GO
```

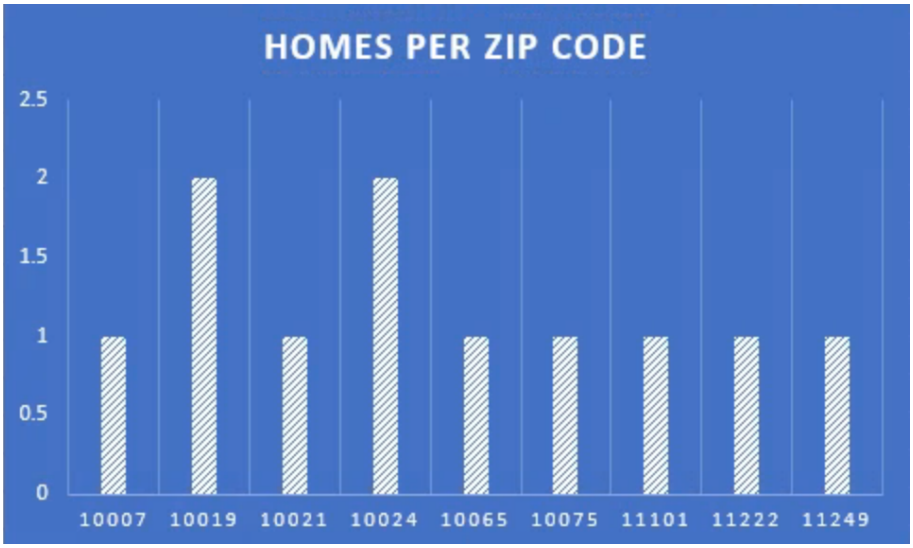
	RoomType	Counts	AverageBudget
1	Bedroom	4	110250
2	Study	4	67000
3	Kitchen	2	142500
4	Bathroom	2	37000
5	Living Room	2	120000





- Question 3: What is the number of homes in the design process by Zipcode?

```
--creating function for counting zipcodes:
GO
CREATE FUNCTION ZipCodeCounts(@zipcode int)
RETURNS int AS
BEGIN
        DECLARE @returnValue int
        SELECT @returnValue=COUNT(home_zipcode) FROM home_details
        WHERE home_details.home_zipcode= @zipcode
        RETURN @returnValue
END
GO

--Creating view to see Number of Homes per ZipCode:
GO
CREATE VIEW NumberHomesPerZipCode AS
SELECT
        home_details.home_zipcode AS ZipCode,
        [dbo].[ZipCodeCounts](home_details.home_zipcode) AS Counts
FROM home_details
GROUP BY home_details.home_zipcode
GO
--let's view
SELECT *
FROM NumberHomesPerZipCode
```



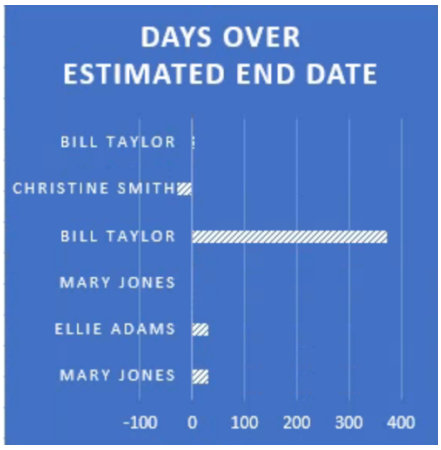

```
--function can be used to figure out which borough a zipcode is part of:
GO
CREATE FUNCTION ZipcodeToBorough(@zipcode int)
RETURNS varchar(20) AS
BEGIN
DECLARE @borough varchar(20);
        IF @zipcode BETWEEN 10000 AND 10286
        BEGIN
        SET @borough='Manhattan'
```

```
        END
        IF @zipcode BETWEEN 10452 AND 10473
        BEGIN
        SET @borough='Bronx'
        END
        IF @zipcode BETWEEN 11201 AND 11252
        BEGIN
        SET @borough='Brooklyn'
        END
        IF @zipcode BETWEEN 11004 AND 11697
        BEGIN
        SET @borough='Queens'
        END
        IF @zipcode BETWEEN 10301 AND 10314
        BEGIN
        SET @borough='Staten Island'
        END
RETURN @borough
END
GO
--example
SELECT [dbo].[ZipcodeToBorough](10301)
```

- Question 4: What portion of projects are completed before project's estimated end date?

```
--let's create a view to see which designers have project that were finished after their estimated end date:
GO
CREATE VIEW ProjectsOverEstimate AS
SELECT
	project.project_id AS ProjectID,
	designer.first_name + ' ' + designer.last_name AS DesignerName,
	project_status.project_status AS ProjectStatus,
	Datediff(y,project.project_estimated_end_date,project.project_end_date) AS
DaysOverEstimatedEndDate
FROM project
RIGHT JOIN designer ON designer.designer_id=project.designer_id
JOIN project_status ON project.project_status_id=project_status.project_status_id
GROUP BY project.project_id,
designer.first_name,
designer.last_name,
project_status.project_status,
project.project_end_date,
project.project_estimated_end_date
GO

--let's take a look at the view
SELECT *
FROM ProjectsOverEstimate
```

	ProjectID	DesignerName	ProjectStatus	DaysOverEstimatedEndDate
1	1	Mary Jones	Before Estimated End Date	31
2	2	Ellie Adams	Before Estimated End Date	31
3	3	Mary Jones	Before Estimated End Date	0
4	4	Bill Taylor	Before Estimated End Date	372
5	5	Christine Smith	After Estimated End Date	-30
6	6	Bill Taylor	Before Estimated End Date	1



- Question 5: What percentage of rooms are within budget or overbudget?

```
--let's create a view which designers went over the room budget and by how much
GO
CREATE VIEW RoomsBudget AS
SELECT
        designer.first_name + ' '+ designer.last_name AS DesignerName,
        room_budget_status.room_budget_status AS  BudgetStatus,
        room_type.room_type AS RoomType,
        (room_details.actual_room_budget - room_details.estimated_room_budget) AS
        AmountDifEstimatedBudget
FROM room_details
RIGHT JOIN home_details ON home_details.home_id=room_details.home_id
JOIN project ON project.project_id=home_details.project_id
JOIN designer ON designer.designer_id=project.designer_id
JOIN room_budget_status ON
room_budget_status.room_budget_status_id=room_details.room_budget_status_id
JOIN room_type ON room_type.room_type_id=room_details.room_type_id
GO

--let's take a look:
SELECT *
FROM RoomsBudget
ORDER BY AmountDifEstimatedBudget DESC;
GO
```

	DesignerName	BudgetStatus	RoomType	AmountDifEstimatedBudget
1	Mary Jones	OVER	Living Room	20000
2	Christine Smith	OVER	Living Room	20000
3	Christine Smith	OVER	Bedroom	10000
4	Mary Jones	OVER	Kitchen	10000
5	Mary Jones	OVER	Bedroom	2000
6	Ellie Adams	OVER	Bathroom	1000
7	Bill Taylor	OVER	Study	1000
8	Christine Smith	WITHIN	Kitchen	0
9	Mary Jones	WITHIN	Bedroom	0
10	Ellie Adams	WITHIN	Bedroom	-1000
11	Mary Jones	WITHIN	Bathroom	-2000
12	Mary Jones	WITHIN	Study	-2000
13	Bill Taylor	WITHIN	Study	-2000
14	Christine Smith	WITHIN	Study	-5000

Part 7

Reflection:

Going into this class, I didn't know anything about databases or how to code in SQL. My only assumption going into the class was that databases would be extremely complicated. I really appreciated the gradual progression in the course curriculum that paralleled with the project components. The conceptual and logical models were instrumental to the success of this project. Understanding how to normalize conceptual models also helped me understand how the core structure of the database evolves. I tried to pace myself so that I applied the new materials to my project as we learned it. A full-circle moment for me was when we learned how to connect the database to Microsoft Access and Excel. All the hard work to make sure my database tables were created and inserted correctly really paid off. If there's anything I would change about this project would be to expand the project to include a project system in addition to the current payment and order system. I really enjoyed all the different components of this project.

Summary:

Databases can be useful and powerful for almost every business. In order to design and implement a good database, one must be extremely meticulous. Querying, inserting, updating and deleting are powerful database tools but database programming and advanced querying really allows us to generate insights from the database. The power of Open Database Connectivity (ODBC) cannot be emphasized more. Not only are we able to connect our database to Microsoft Access but also to Excel, RStudio, C/C++, Python, and more.