

**CSC 315-01**  
**5/6/2022**  
**Final Project Report**

Group Number: 01-4

Group/Project Name: TCNJ Energy Demand Tool

Group Members:

Bret Elphick: elphicb1@tcnj.edu

Dalton Hutchinson: hutchid6@tcnj.edu

Jimmy Fay: fayj4@tcnj.edu

Andrew Fellenz: fellena1@tcnj.edu

Brooks Watson: watsonb2@tcnj.edu

Katherine Gellman: gellmak1@tcnj.edu

Alexander Reyes: reyesa16@tcnj.edu

## **Inception: "Executive Summary"**

At the initial stakeholder meeting with Paul Romano he told us that TCNJ is planning to be carbon neutral by 2040. It is crucial that the TCNJ Energy Management Team reaches this goal as the effects of climate change become more severe. To help them in this process, our group has created an application that could address the economic and environmental sides of this problem. We hope that the application can help them reach their goal in a cost-effective way.

Our design and implementation approach changed throughout the semester. In the beginning, we planned for users to input a meter, a year, and a month. Once implementation began however, we realized that the data has many overlapping months. For example, some of the meter entries start and end in the middle of two different months. Instead, we opted for the user to input a meter, a start date, and an end date. This was a better solution for the developers, and it also allows users to get larger chunks of data at a time. After making this change, we wrote our queries to get the rows closest to the start date and end date, and everything in between for the selected meter. While this query sounds simple, it requires the use of subqueries and other SQL functions like absolute value. This query was crucial because users no longer had to know exact start and end dates in the database.

After obtaining the data, we used HTML, CSS, and Javascript to present the data in an effective way. At the top of our results page, there is a table to present the raw data to the user. We expect users, such as Paul Romano, to use the table often to get exact data points. Under the table, there are two line graphs. The graphs plot the change in energy and cost over the inputted time range. Both graphs use Chart.js and users can hover over the line to see the exact values for the data points. Under the charts, there are bootstrap cards displaying the average energy, sum of energy, average CO2e (using conversion factors from carbonfund.org), average cost, and sum of costs. To calculate these values, we used SQL aggregate functions. We hope they provide a convenient summary of the economical and environmental sides of the energy demand data for the inputted meter and time range. Both pages use Bootstrap Grid to improve accessibility and user-friendliness. Additionally, the form has error handling for cases where the user forgets to fill out fields or fills them out in the wrong order.

The most valuable aspect of our product is that it is geared towards the energy demand data that TCNJ has been collecting over the past several years. Our tool presents all relevant data in one, easy to use module. It addresses both the economic and environmental sides of the energy demand problem as Paul Romano required. Furthermore, it is easy to add more data to the database by simply running the shell script we wrote earlier in the semester. The only requirement is that the excel sheet is in the same format as the seed data.

Lastly, there is a small cost for the stakeholder to implement our application. The easiest way to implement the application would be linking the GitHub repository at <https://energy.tcnj.edu/>. People who want to use the application would have to follow the installation guide linked on the README. Next, the TCNJ Energy Demand Tool can exist as its own independent website. This solution is more costly since the application would have to be

hosted by a server, such as one on the TCNJ HPC. The website could be linked at <https://energy.tcnj.edu/> to get the application more exposure. The final and most costly way for the stakeholder to implement the application is to embed it at <https://energy.tcnj.edu/>. While we are not entirely sure of the logistics, it would likely require our stakeholder to reach out to the company that hosts the TCNJ website.

### **Elaboration: Project Proposal and Specifications**

**REVISIONS:** At this stage, we planned for the user to input up to 5 meters to display graphs and averages over yearly and monthly time intervals. In the end, we found start date and end date to be a more powerful input that does not limit the graphs and averages that can be queried and created. Furthermore, while we stuck to the original idea of keeping the UI simple, we changed the layout significantly. Lastly, we used conversion factors from carbonfund.org instead of epa.gov and eia.gov.

### **Problem Statement:**

With the effects of global warming becoming increasingly apparent, it is of the utmost importance for TCNJ to monitor its energy use. Companies and colleges around the world are striving to lessen their carbon footprint and become carbon neutral in the near future. Additionally, it is important to meet these environmental goals in an economically feasible way. To aid in this endeavor, our group wants to create an interactive software application for the TCNJ Energy Management Team that makes use of the energy demand data by building.

### **Objective of the Module:**

At Paul Romano's presentation, he stated that TCNJ wants to be carbon neutral by 2040. Keeping this goal in mind, the overall objective of the project is to make the energy demand and energy cost data easily accessible to the Energy Management Team at TCNJ and others who might find it useful. This application will be a useful tool for monitoring the data and making the right decisions for both the environmental and economic sides of this problem.

**Description of the desired end product, and the part you will develop for this class:** To start, the user will enter meters of interest (planning for a maximum of 5). Then, the application will be able to display line graphs for the meters plotting energy demand and energy cost data over time. Along with these line graphs, the application will display averages: average yearly energy demand, energy cost, and CO<sub>2</sub>e over time and average monthly energy demand, energy cost, and CO<sub>2</sub>e for a given year. These features will help the TCNJ Energy Management Team identify trends in cost, demand, and carbon footprint.

### **Description of the importance and need for the module, and how it addresses the problem:**

According to NOAA scientists, New Jersey temperatures have gone up by 3 degrees Fahrenheit over the past century. Also, over the past decade, we have seen record-setting storm seasons in

NJ. The data has made it clear that the effects of climate change are in motion. Therefore, it is important to create a tool that can be used by energy experts to reduce the carbon footprint of TCNJ. With our tool, TCNJ can monitor its energy demand and use that data to come up with a strategic environmental plan. Going along with that, the tool will also address cost, thus allowing TCNJ to tackle this problem in the most economically efficient way.

**Plan for how you will research the problem domain and obtain the data needed:** The TCNJ energy demand data appears to already be on Canvas. Also, to convert Kwh and therms to CO<sub>2</sub>e will we use formulas from reputable sources such as [epa.gov](http://epa.gov) and [eia.gov](http://eia.gov).

**Other similar systems/approaches that exist, and how your module is different or will add to the existing system:**

Reliant and Energy Star have energy usage tool for commercial buildings and residencies that makes comparisons to common averages. Additionally, there are also many applications where the user enters in their energy usage and it outputs the cost of the by day/month/year. While we might be incorporating some of their UI designs and their data for comparison purposes into our own implementation, ours is unique because it's specifically for colleges like TCNJ.

**Possible other applications of the system (how it could be modified and reused):** If our application is useful to the TCNJ Energy Management Team we could reach out to other colleges to see if they could benefit from the tool. Additionally, if companies track their energy usage they might be able to make use of it as well. We are hoping that it could be useful in monitoring energy demand, monitoring energy cost, identifying patterns, and reducing carbon footprint.

**Performance:**

As we learn more about proper ER design the CSC 315 portion of the team will come up with efficient access paths. Also, since we are not dealing with huge amounts of data, performance should not be a serious problem. We also plan to keep the user interface rather clean and simple so that no unnecessary UI features slow down the application.

**Security:**

Since the project will be on GitHub we do not have to deal with security for now. However, if we want to make the application publicly available we will use database security knowledge from CSC 315. As we create our application we will keep in mind techniques, such as defense in-depth, and make sure only approved admin accounts can change things in the database. We will use an already existing login system with encryption of login credentials to ensure the security of admin accounts.

**Backup and Recovery:**

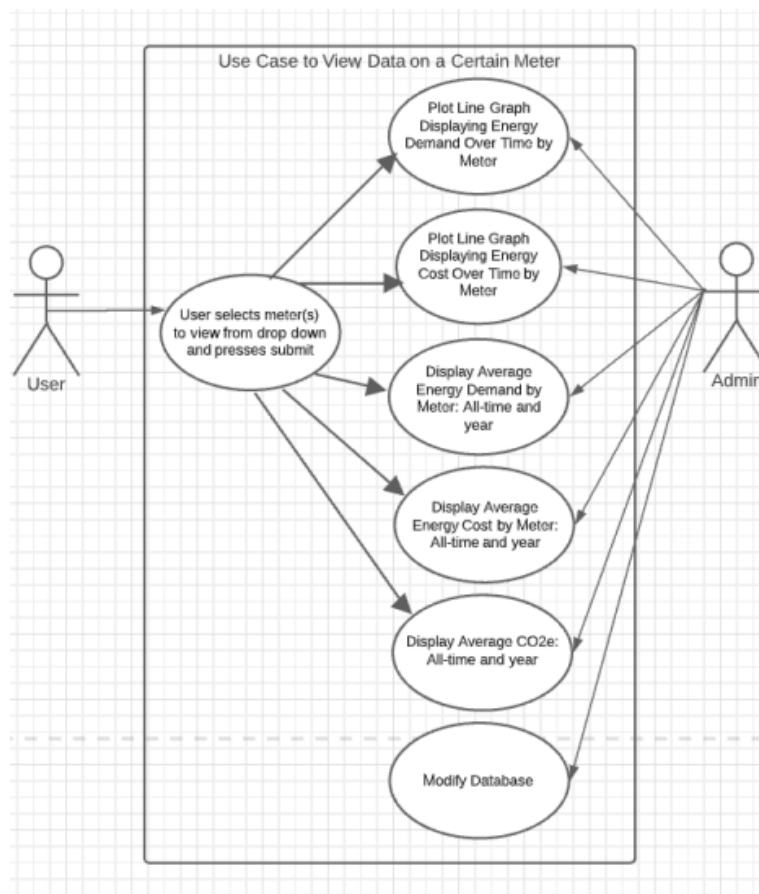
GitHub has features in which we can revert our project to previous versions if something were to go wrong. Since we have multiple computer science members who are experienced with

GitHub, the project should be safe if we need to recover it. However, if we want to make the application publicly available we will use database backup and recovery knowledge from CSC 315.

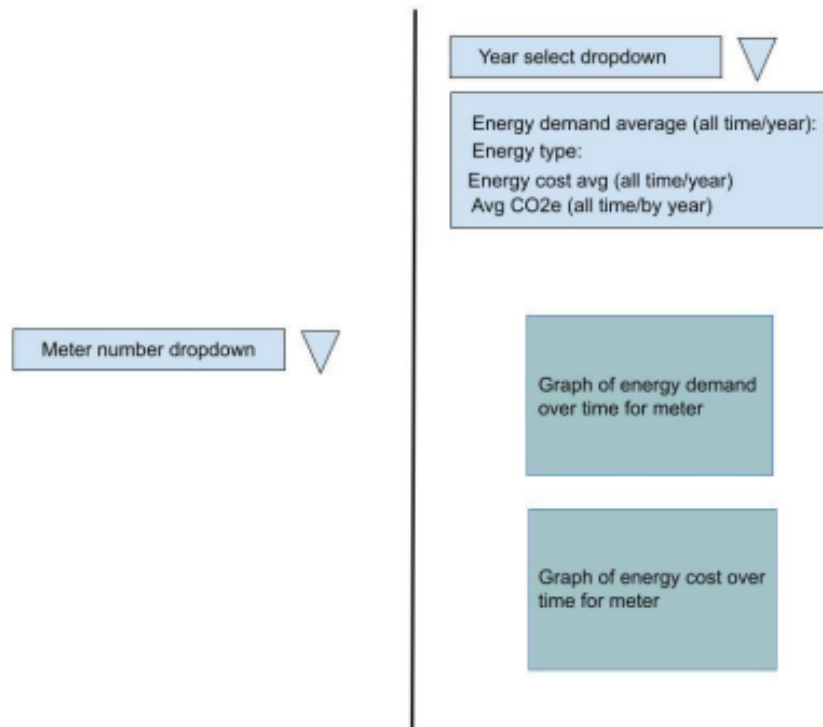
**Technologies and database concepts the group will need to learn, and a plan for learning these:**

We will need to learn more about proper relational database design. As of now, we have a basic understanding of entities, attributes, and the relationships amongst entities. But we will need to learn more about how to make efficient access paths and how to limit the number of NULL entries. As we learn more about databases we will be able to choose a technology that seems right for our project, whether it be Postgres, MySQL, etc.


**Diagrammatic Representation of System Boundaries (Use Case):**



## UI Design:



## Quad Chart:

<div><b>Energy Demand</b> Section 1 Group 4</div>	
<b><u>Need</u></b> <p>With the effects of climate change becoming more prominent, TCNJ and most other universities are striving to lessen their carbon footprint and become carbon neutral in the near future. We believe there is a need to track energy consumption in specific parts of our campus and make this data easily available to the people who can use it to the school's benefit.</p>	<b><u>Approach</u></b> <p>Our group wants to create an interactive software application for the TCNJ energy Management Team that makes use of the energy demand data. The application will be able to present data about building energy demand in both graphical and tabular forms depending on the user's request.</p>
<b><u>Benefit</u></b> <p>If our tool is useful, we could aid our university and possibly other universities in their goal to reach a carbon neutral status by showing useful information to the respective energy management teams, which they could react accordingly to.</p>	<b><u>Competition</u></b> <p>Reliant has an energy usage tool for residencies that makes comparisons to averages. There are also many applications where the user enters in their energy usage, and it outputs the cost of the by day/month/year. While we might be incorporating some of these ideas into our own implementation, ours is unique because it's specifically for colleges and should be able to provide feedback on how to reach carbon neutrality.</p>

## **Proposal Pitch Presentation**

**REVISIONS:** At this point in the project, we planned to use buildings as the main filter for our queries. For example, we wanted to allow the user to fetch the average energy used by a given building over a given year. Also, we wanted to add a school comparison feature that compared TCNJ energy demand data with other schools of similar size and in a similar region. In the end, we found that the excel sheet did not provide building-specific data, and we ran out of time before reaching out to other schools.

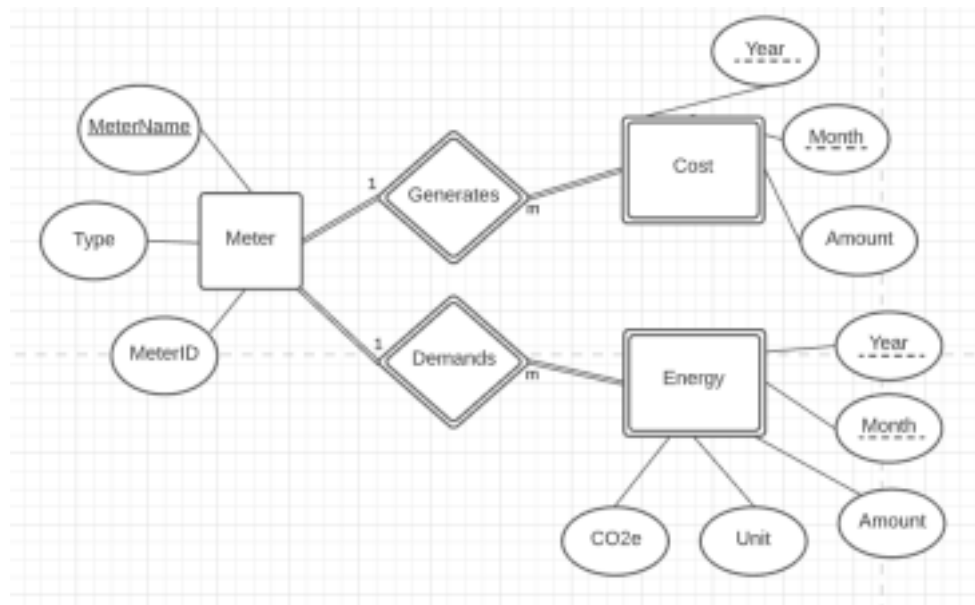
[https://docs.google.com/presentation/d/1lUeX46Y4lFBfLoqhOVffujV3\\_ZCzWnesArtRYRmXl38/edit?usp=sharing](https://docs.google.com/presentation/d/1lUeX46Y4lFBfLoqhOVffujV3_ZCzWnesArtRYRmXl38/edit?usp=sharing)

## **Elaboration: Design**

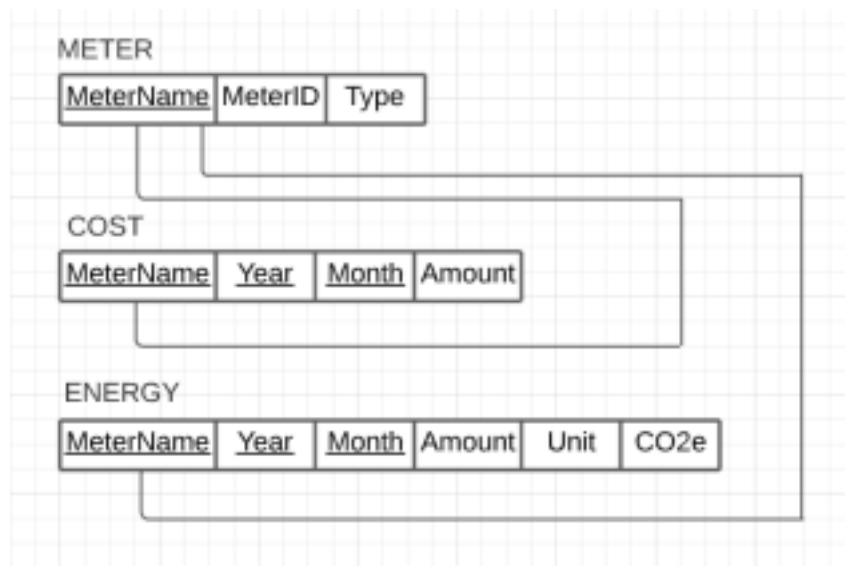
**This is the database model that we submitted for Stage III.**

**REVISIONS:** We ended up using and storing start date and end date rather than year and month. Also, in the process of normalization, “Unit” had to be stored in the meter relation rather than energy relation. This changed the initial database size (shown in 5b database model document below). Finally, CO2e is a derived attribute that is not directly stored in the database.

## **ER Diagram:**



## Relational Schema:



## Initial Database Size:

- 22 rows \* 3 columns = 66 records in Meter (22 meters and 3 values to keep track of per meter)
- 1426 rows \* 4 columns = 5704 records in Cost (1426 meter readings, and 4 values to keep track of per meter reading)
- 1426 rows \* 6 columns = 8556 records in Energy (1426 meter readings, and 6 values to keep track of per meter reading)

## Types and Average Number of Searches:

- Types of searches: join, project, grouping and aggregate functions
- To get the price, energy demand, and CO2e for a meter, year, and month combination we will need to join METER and COST. Each join will require the join condition to be checked 1426 times. We will then project the needed attributes.
- To calculate yearly averages we will use the AVG aggregate function. We will first select the tuples by the input year. Then, we will group by MeterName - This will require 12 additions and a division for each meter. Finally, we will select the input meter.

## Mid-Semester Project Presentation

**REVISIONS:** The previous revisions from “Elaboration: Design” apply here. Also, by using start date and end date rather than year and month, many of our queries changed. For example, all averages and sums are calculated for the inputted meter over the inputted time range rather than calculating averages over the inputted year as originally planned. Lastly, the UI layout changed quite drastically, as mentioned in previous revision comments. For instance, we opted to



use a form page and results page instead of putting everything on one page, and we added a table.

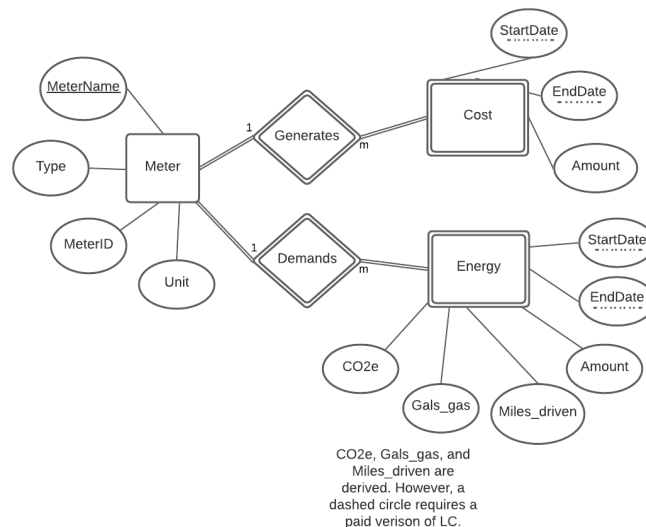
[https://docs.google.com/presentation/d/1-F0cJVGnpRk\\_lIBd-DKJfcl0P-KeFcqZ\\_3qqUZtptx4/edit?usp=sharing](https://docs.google.com/presentation/d/1-F0cJVGnpRk_lIBd-DKJfcl0P-KeFcqZ_3qqUZtptx4/edit?usp=sharing)

## Construction: Tables, Queries, and User Interface

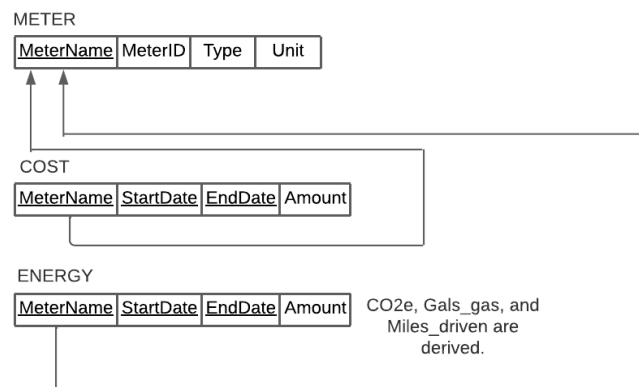
This is the database model document that we submitted for Stage 5a.

**REVISIONS:** The ER model and relational schema have remained the same from this iteration. However, a few of the SQL queries changed. For example, the query to sum/average the cost/energy amount had incorrect output in certain cases, so it was modified in 5b.

## ER Diagram



## Relational Schema



## SQL commands for new design

The same commands work for cost.

- To find the closest start date:

```
select energy.start_date from energy where metername = '<Meter Name>' order by  
abs(energy.start_date - '<User Start Date>') limit 1;
```

- To find the closest end date:

```
select energy.end_date from energy where metername = '<Meter Name>' order by  
abs(energy.end_date - '<User End Date>') limit 1;
```

- Find all tuples in this range using:

```
select * from energy where metername = '<Meter Name>' and start_date >= '<Closest Start  
Date>' and end_date <= '<Closest End Date>';
```

- We can sum or average the energy/cost amount using:

```
select avg(amount) from energy where amount in (select energy.amount from energy where  
metername='<Meter Name>' and start_date>=(select energy.start_date from energy where  
metername='<Meter Name>' order by abs(energy.start_date - '<User Start Date>') limit 1) and  
end_date<=(select energy.end_date from energy where metername='<Meter Name>' order by  
abs(energy.end_date - '<User End Date>') limit 1));
```

## Database Thoughts

- We have decided that CO2e, Gas\_gals, and Miles\_driven can be derived from Amount and Unit attributes in ENERGY. Therefore we are not going to directly store them in the database. LucidCharts requires a paid version to use dashed ovals, therefore we have added a note next to these attributes indicating that they are derived in the ER Diagram.
- As for the graphs, we will display a bar graph if less than 5 tuples are returned. If there are more than we will display a line graph.

## Initial Database Size

- 22 rows \* 4 columns = 88 records in Meter (22 meters and 3 values to keep track of per meter)
- 1426 rows \* 4 columns = 5704 records in Cost (1426 meter readings, and 4 values to keep track of per meter reading)
- 1426 rows \* 4 columns = 11,408 records in Energy (1426 meter readings, and 4 values to keep track of per meter reading)

## Types and Average Number of Searches

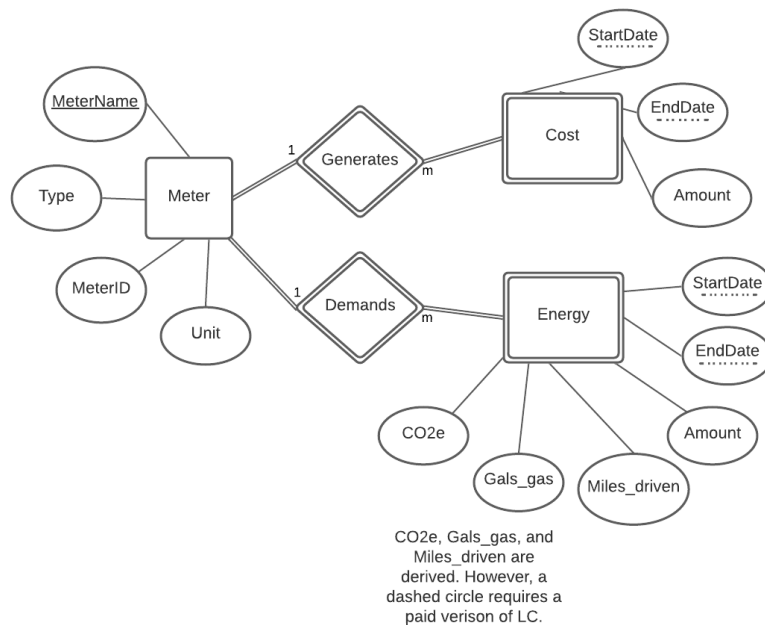
- Types of searches: join, project, grouping and aggregate functions
- To get the energy demand, and environmental statistics for a meter, start date, and end date combination we will need to join METER and ENERGY to get the type and unit.

- To calculate averages and sums for meter, start date, and end date combinations we will use the fourth query shown above. The number of tuples involved in the aggregate function depends on the amount returned by the subquery.

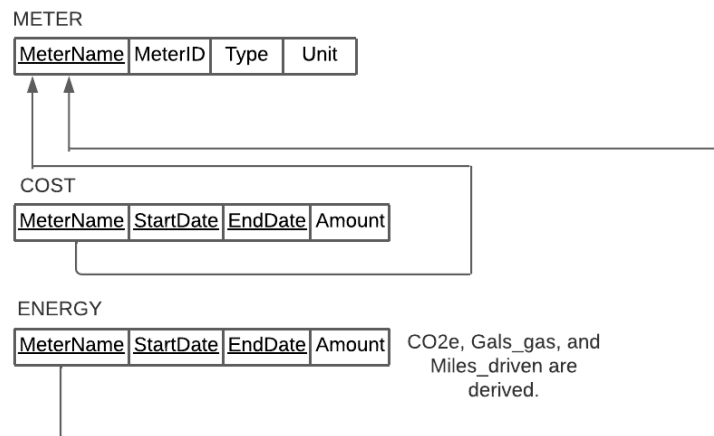
**This is the database model document that we submitted for Stage 5b.**

**REVISIONS:** n/a

## ER Diagram



## Relational Schema



## SQL commands for application

- **Commands to create tables and views**

```
CREATE TABLE METER(MeterName text, MeterID text, Type text, Unit text,  
PRIMARY KEY(MeterName));
```

```
CREATE TABLE COST(MeterName text, StartDate date, EndDate date, Amount  
float(2), PRIMARY KEY(MeterName, StartDate, EndDate), FOREIGN KEY  
(MeterName) REFERENCES METER(MeterName));
```

```
CREATE TABLE ENERGY(MeterName text, StartDate date, EndDate date, Amount  
integer, PRIMARY KEY(MeterName, StartDate, EndDate), FOREIGN KEY  
(MeterName) REFERENCES METER(MeterName));
```

```
CREATE VIEW ENERGY_COST AS SELECT ENERGY.MeterName,  
ENERGY.StartDate, ENERGY.EndDate, ENERGY.Amount AS EnergyAmount,  
COST.Amount AS CostAmount FROM ENERGY INNER JOIN COST ON  
ENERGY.MeterName = COST.MeterName AND ENERGY.StartDate=COST.StartDate  
AND ENERGY.EndDate = COST.EndDate;
```

- **Command to format data for the table**

```
select energy.startdate, energy.enddate, energy.amount, cost.amount from energy inner  
join cost on energy.metername = cost.metername and energy.startdate = cost.startdate and  
energy.enddate = cost.enddate where energy.metername="" + request.form['metername'] +  
"" and energy.startdate>=(select energy.startdate from energy where metername="" +  
request.form['metername'] + "" order by abs(energy.startdate - "" + request.form['startdate']  
+ "") limit 1) and energy.enddate<=(select energy.enddate from energy where  
metername="" + request.form['metername'] + "" order by abs(energy.enddate - "" +  
request.form['enddate'] + "") limit 1);
```

- **Command to get average energy amount**

```
select avg(amount) from energy where metername="" + request.form["metername"] + ""  
and startdate >=(select energy.startdate from energy where metername="" +  
request.form["metername"] + "" order by abs(energy.startdate - "" +  
request.form["startdate"] + "") limit 1) and enddate<=(select energy.enddate from energy  
where metername="" + request.form["metername"] + "" order by abs(energy.enddate - "" +  
request.form["enddate"] + "") limit 1);
```

The same command was used to find the sum of energy amounts by replacing “avg” with “sum”. Additionally, the same command was used to find average cost amount and sum of cost amounts by replacing “energy” with “cost”.

- **Command to get all distinct meters**

```
select distinct metername from meter;
```

- **Command to get the type and unit of inputted meter**

```
select type, unit from meter where metername = "" + request.form["metername"] + "";
```

- **Command to get all start dates and end dates in range of user input**

```
select startdate, enddate from energy where metername="" +  
request.form["metername"] + "" and startdate >=(select energy.startdate from energy  
where metername="" + request.form["metername"] + "" order by abs(energy.startdate - ""  
+ request.form["startdate"] + "") limit 1) and enddate<=(select energy.enddate from  
energy where metername="" + request.form["metername"] + "" order by  
abs(energy.enddate - "" + request.form["enddate"] + "") limit 1);
```

### **Database Thoughts**

- We have decided that CO2e, Gas\_gals, and Miles\_driven can be derived from Amount and Unit attributes in ENERGY. Therefore we are not going to directly store them in the database. LucidCharts requires a paid version to use dashed ovals, therefore we have added a note next to these attributes indicating that they are derived in the ER Diagram.
- As for the graphs, we will display a bar graph if less than 5 tuples are returned. If there are more than we will display a line graph.

### **Initial Database Size**

- 22 rows \* 4 columns = 88 records in Meter (22 meters and 3 values to keep track of per meter)
- 1426 rows \* 4 columns = 5704 records in Cost (1426 meter readings, and 4 values to keep track of per meter reading)
- 1426 rows \* 4 columns = 11,408 records in Energy (1426 meter readings, and 4 values to keep track of per meter reading)

### **Types and Average Number of Searches**

- Types of searches: join, project, grouping and aggregate functions


- To get the energy demand, and environmental statistics for a meter, start date, and end date combination we will need to join METER and ENERGY to get the type and unit.
- To calculate averages and sums for meter, start date, and end date combinations we will use the fourth query shown above. The number of tuples involved in the aggregate function depends on the amount returned by the subquery

### **Transition: Maintenance**

<https://github.com/TCNJ-degoodj/cab-project-01-4>

### **Final Project Demonstration**

<https://docs.google.com/presentation/d/1PAurO21FNe7oXjhqYi6Tyi3DY94QSIrgTZSh4qjMGxQ/edit?usp=sharing>




## CSC 315: Energy Demand Tool

Select Meter:

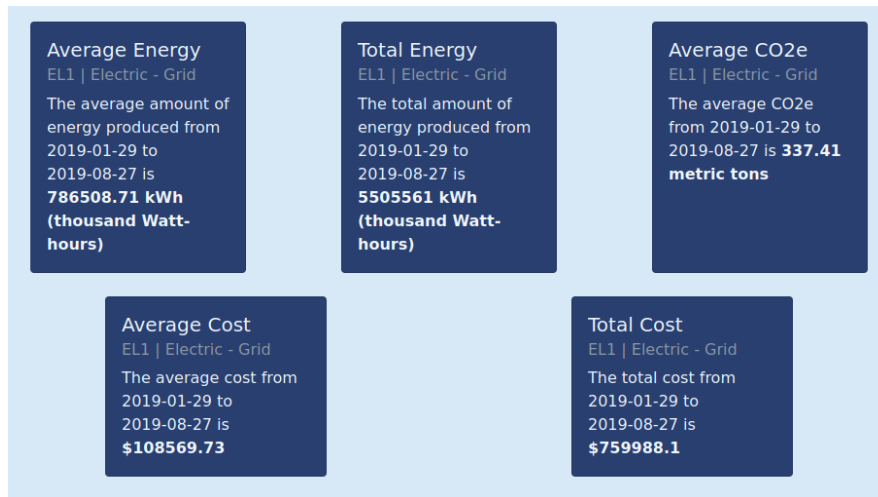
Enter Start Date:

Enter End Date:



## Results

Start Date	End Date	Energy Amount	Cost Amount
2019-01-29	2019-02-27	278158	85078.82
2019-02-28	2019-03-28	162008	77767.29
2019-03-29	2019-04-29	438928	84594.32
2019-04-30	2019-05-29	1774635	118147.01
2019-05-30	2019-06-27	770479	118522.84
2019-06-28	2019-07-29	1483767	162321.69
2019-07-30	2019-08-27	597586	113556.16



## Transition: Product Hand Over

Bret Elphick: <https://github.com/bretelphick/cab-project-01-4>

Dalton Hutchinson: <https://github.com/hutchid6/-cab-project-01-4>

Jimmy Fay: <https://github.com/LordOfLunch/cab-project>