Building Relation = {portfolio_manager_id, property_name, property_type, year_built, gross_floor_area, specification_type, specification_value}

= {ppid, pn, pt, y, a, st, sv} = R1

ppid + = ppid, pn, pt, y, a

(ppid, st) + = ppid, st, sv

Key: ppid, st

FD1 = {(ppid -> pn, pt, y, a), (ppid, st, sv)}

This relation is 1NF. It passes 1NF because there are no composite attributes, no multivalued attributes, and no nested relations. However, it does not pass 2NF because the non-prime attributes that are given by just ppid are not fully functionally dependent on the primary key. They are only partially dependent. Therefore, this relation can be broken down into the two below:

R11 = {ppid, pn, pt, y, a}
R12 = {ppid, st, sv}

This relation is in BCNF. It now passes 2NF because each non-prime attribute is fully functionally dependent on the key within their relations. It passes 3NF because there are no transitive dependencies. It is in BCNF because the key is a superkey that will uniquely identify tuples.

Meter Relation = {meter_name, portfolio_manager_meter_id, meter_type, units}

= {mn, pmid, mt, u} = R1

mn+ = mn, pmid, mt, u

Key: mn

This relation is in BCNF. It passes 1NF because there are no composite attributes, no multivalued attributes, and no nested relations. It passes 2NF because each non-prime attribute is fully functionally dependent on the key. It passes 3NF because there are no transitive dependencies. It is in BCNF because the key is a superkey that will uniquely identify tuples.

Meter Entry Relation = {meter_name, portfolio_manager_meter_id, meter_type, units, meter_consumption_id, usage_quantity_total, cost_total, start_date, end_date}

= {mn, pmid, mt, u, mcid, ut, ct, sd, ed} = R1
mn+ = mn, pmid, mt, u

mcid + = mcid, ut, ct, sd, ed, mn

Key: mcid

FD1 = {(mcid -> ut, ct, sd, ed, mn, pmid, mt, u), (mn -> pmid, mt, u)}

This relation is 2NF. It passes 1NF because there are no composite attributes, no multivalued attributes, and no nested relations. It passes 2NF because each non-prime attribute is fully functionally dependent on the key. However, it does not pass 3NF because there are transitive dependencies due to meter_name. Therefore, this relation can be broken down into the two below:

R11 = {mn, pmid, mt, u}
R12 = {mcid, ut, ct, sd, ed, mn}

This relation is in BCNF. It now passes 3NF because there are no transitive dependencies. It is in BCNF because the key is a superkey that will uniquely identify tuples.

Metered By Relation = {portfolio_manager_id, meter_name}
= {ppid, mn} = R1

Key: ppid, mn

This relation is in BCNF. It passes 1NF because there are no composite attributes, no multivalued attributes, and no nested relations. It passes 2NF because there are no non-prime attributes as the relation is only the key. It passes 3NF because there are no transitive dependencies. It is in BCNF because the key is a superkey that will uniquely identify tuples.

Query View Based on Dates

CREATE VIEW julyCosts AS
SELECT property_name AS name, gross_floor_area AS area, usage_quantity_total AS use, cost_total AS cost

FROM meterEntries
JOIN meteredBy
ON meterEntries.meter_name = meteredBy.meter_name
JOIN property
ON meteredBy.portfolio_manager_id = property.portfolio_manager_id

WHERE start_date = '2009/07/01' AND end_date = '2009/07/31';
This view will allow a focus of data on a specific time frame. This example uses the whole month of July. This will allow users to gather information on the specific month and determine what factors may have contributed to a higher or lower than expected energy use or cost.

Data Requirements/Transactions:
Input of a time period
Query falls within the bounds of the data
Information from pre-existing tables, such as building names, must have values (not null)
Transaction of reading the information from a specific time period and retrieving the data

Queries include:
What was the cost of each building this month?
Did size contribute to higher energy use?

Queries in SQL:

SELECT name, cost FROM julyCosts;
SELECT name, area, use FROM julyCosts;

Alternatively, a view could be made without the specification of WHERE, and this variable can be left to the query itself. For example, to compare two different months with a "monthCosts" virtual table could look like this:

SELECT name, cost, use FROM monthCosts WHERE (start_date = '2009/07/01' AND end_date = '2009/07/31') OR (start_date = '2009/08/01' AND end_date = '2009/08/31');

Query View Based on Usage

```
CREATE VIEW highUsage AS
SELECT property_name AS name, gross_floor_area AS area, usage_quantity_total AS use,
start_date AS start, end_date AS end

FROM meterEntries
JOIN meteredBy
ON meterEntries.meter_name = meteredBy.meter_name
JOIN property
ON meteredBy.portfolio_manager_id = property.portfolio_manager_id

WHERE cost_total >= 2500;
```

This view will allow a focus of data on a specific price range. This example uses a total cost of a month at $2500 or more. This will allow users to gather information about the buildings that have been overusing the energy and determine which ones need to be looked at in the future. This will also help in the calculation of an efficiency and sufficiency score of each building.

Data Requirements/Transactions:
Input of a price range
Time range should be inputted to avoid generating too many tuples in the query
Query falls within the bounds of the data
Information from pre-existing tables, such as building names, must have values (not null)
Transaction of reading the information with specific price points and retrieving the data

Queries include:
Which buildings cost the most overall?
Of the buildings that had high costs, what were the time periods that had a spike in cost?

Queries in SQL:

```
SELECT name, cost FROM highCost WHERE start = '2009/01/01' AND end = '2009/09/30';
SELECT name, start, end FROM highCost;
```

Alternatively, a view could be made without the specification of WHERE, and this variable can be left to the query itself. For example, to compare two different buildings' max costs with an "overallCost" virtual table could look like this:

```
SELECT name, max(cost) FROM overallCost WHERE name = 'Forcina';
SELECT name, max(cost) FROM overallCost WHERE name = 'STEM Building';
```