1. Jaya Sabarish Reddy Remala (jr6421)
2. Sumedh Sandeep Parvatikar (sp7479)

SHEMS - PDS Project Part 1

Smart Home Energy Management System **1st December 2023**

Introduction:

We will be designing an efficient Smart Home Energy Management System (SHEMS) using a relational data model to store and display necessary information to the customer. We will be dividing our project into two parts according to the guidelines. In the first part we will deep dive into the relational schema for the system. This report contains all the details from ER diagram to the relational schema along with SQL queries for some questions provided in the handout.
For our project, we will be using **PostgreSQL** to design the database and the respective relational model.

Schema:

Address(**address_id**, unit_number, address_line, city, state, zipcode)

Customers(**customer_id**, first_name, Last_name, *billing_address_id*)

ServiceLocations(**sl_id**, *customer_id, address_id*, start_date, apt_area, num_bedrooms, num_occupants)

Device(**model_id**, device_type, model_number, brand, release_year, manufacture_year)

EnrolledDevices(**enrolled_device_id**, *sl_id, model_id*)

DeviceData(**data_id**, *enrolled_device_id*, event_timestamp, event_label, event_value)

EnergyPrices(**zipcode**, **price_timestamp**, price_per_kwh);

1. **Address Table:**
    - **Columns:**
        - `address_id`: Serial primary key.
        - `unit_number`: VARCHAR(50), not null.
        - `address_line`: VARCHAR(100), not null.
        - `city`: VARCHAR(100), not null.
        - `state`: VARCHAR(2), not null.

        - `zipcode`: INT, not null.

This table is designed to store address information, with a unique identifier (`address_id`) as a **primary key** and details such as unit number, address line, city, state, and zipcode.

2. **Customers Table:**
    - **Columns:**
        - `customer_id`: Serial primary key.
        - `first_name`: VARCHAR(100), not null.
        - `last_name`: VARCHAR(100), not null.
        - `billing_address_id`: INT, not null, foreign key referencing `Address(address_id)`.

This table represents customer information with a unique identifier (`customer_id`) as **the primary key**. It also includes the *billing_address_id* as a **foreign Key**, referencing the `Address` table.

3. **ServiceLocations Table:**
    - **Columns:**
        - `sl_id`: Serial primary key.
        - `customer_id`: INT, foreign key referencing `Customers(customer_id)`
        - `address_id`: INT, UNIQUE foreign key referencing `Address(address_id)`
        - `start_date`: DATE.
        - `apt_area`: DECIMAL(18,6).
        - `num_bedrooms`: INT.
        - `num_occupants`: INT.

This table stores information about service locations, including the **sl_id** as **primary Key,** start date, apartment area, number of bedrooms, number of occupants, and **foreign keys** *customer_id,address_id* (Unique) referencing the `Customers` and `Address` tables.

4. **Device Table:**
    - **Columns:**
        - `model_id`: VARCHAR(100) primary key.
        - `device_type`: VARCHAR(100), not null.
        - `model_number`: VARCHAR(100), not null.
        - `brand`: VARCHAR(100).
        - `release_year`: INT.
        - `manufacture_year`: INT.

This table represents information about electronic devices, with a unique identifier (`model_id`) as the **primary key** and details such as device type, model number, brand, release year, and manufacture year.

5. **EnrolledDevices Table:**
    - **Columns:**
        - `enrolled_device_id`: Serial primary key.
        - `sl_id`: INT, not null, foreign key referencing `ServiceLocations(sl_id)`.
        - `model_id`: VARCHAR(64), not null, foreign key referencing `Device(model_id)`.

This table links devices to service locations. It includes a unique identifier (`enrolled_device_id`) as the **primary key** and foreign keys *sl_id*, *model_id* referencing both the `ServiceLocations` and `Device` tables.

6. **DeviceData Table:**
    - Stores data related to events from enrolled devices.
    - **Columns:**
        - `data_id` (Serial, Primary Key): Unique identifier for each data entry.
        - `enrolled_device_id` (INT, NOT NULL): Foreign key referencing `EnrolledDevices(enrolled_device_id)`.
        - `event_timestamp` (TIMESTAMP): Timestamp of the event.
        - `event_label` (VARCHAR(100), NOT NULL): Label describing the event.
        - `event_value` (DECIMAL(18,6)): Value associated with the event.

This table stores the records sent by the smart devices being **data_id** as the **primary key and** *enrolled_device_id* as the foreign key references Enrolled Devices Table follows the event timestamp, label and value as the other attributes.

7. **EnergyPrices Table:**
    - **Columns:**
        - `zipcode`: INT, not null.
        - `price_timestamp`: TIMESTAMP, not null.
        - `price_per_kwh`: DECIMAL(18,6), not null.
        - **Primary Key:** (`zipcode`, `price_timestamp`).

This table stores energy prices with a **primary key** consisting of `zipcode` and `price_timestamp`. It includes the price per kilowatt-hour (`price_per_kwh`) at a specific timestamp for a given zipcode.

These tables are interconnected using foreign keys to establish relationships between different entities in the schema. The schema provides a structured way to store and organise information related to addresses, customers, service locations, devices, enrolled devices, and energy prices.

Normalization:

Normalization is the process of organizing data to reduce redundancy and improve data integrity. The schema exhibits normalization principles:

- **Address Table:**
  - Normalized by having a separate table for addresses, reducing redundancy.
  - Address information is not duplicated; instead, references (foreign keys) are used.
- **Customers Table:**
  - Normalized by using a separate table for customer information.
  - The `billing_address_id` foreign key a relationship with the `Address` table, avoiding duplicate address data.
- **ServiceLocations Table:**
  - Normalized by using separate tables for service location, customer, and address information.
  - Foreign keys link to the `Customers` and `Address` tables, preventing redundancy.
- **Device Table:**
  - Normalized by having a separate table for device information.
  - Device details are not duplicated across multiple tables.
- **EnrolledDevices Table:**
  - Normalized by using foreign keys to establish relationships with the `ServiceLocations` and `Device` tables.
  - Device and service location details are not duplicated.
- **DeviceData Table:**
  - Normalized by using a foreign key to establish a relationship with the `EnrolledDevices` tables.
  - Event_label and Event_value changes according to the record and device.
- **EnergyPrices Table:**
  - Normalized by using a composite primary key to represent the unique identifier for energy prices.
  - The table structure avoids redundant storage of price information for the same zipcode and timestamp.

Note: Even though the customer billing Address and the service location address overlap with the same details we can uniquely identify them via address_id.

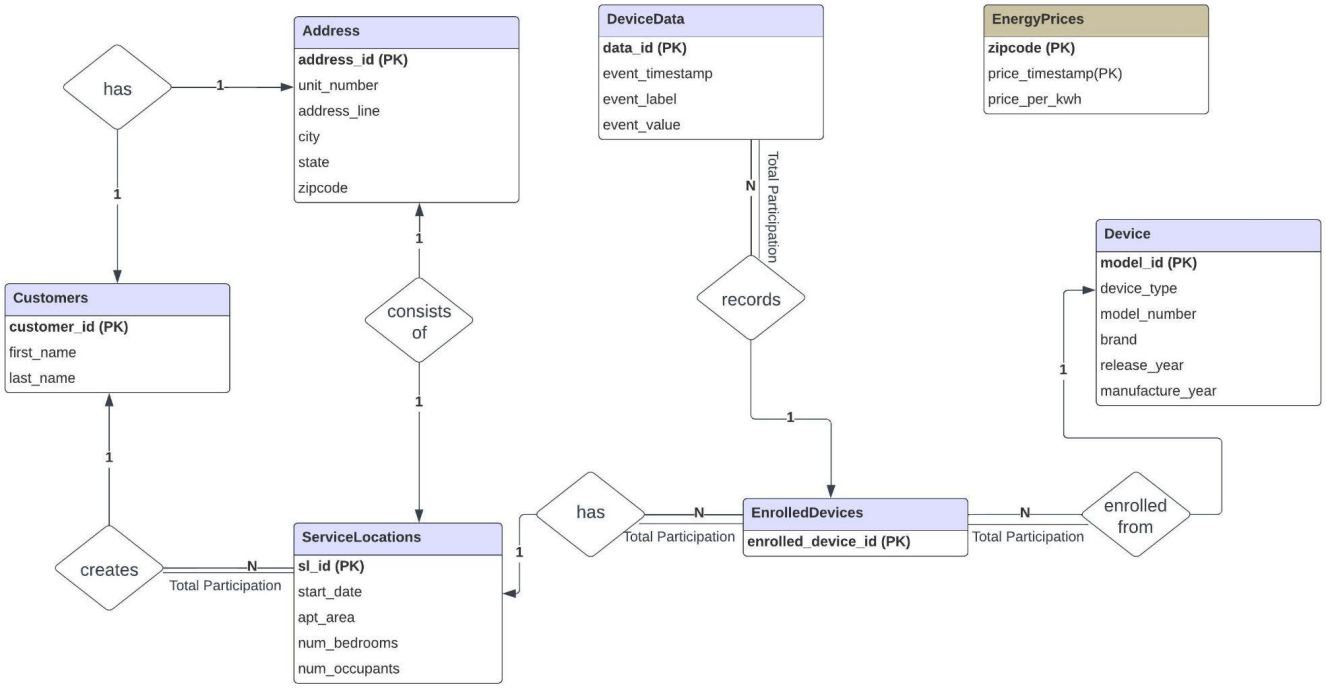
Storage Efficiency: As all the tables were being normalized, there were rare chances of expecting data redundancy at the maximum sample data. DeviceData Table might expect more records as the devices were smart and continuously sent data every **10 minutes** in addition to if some events occurred**.**

Note: We followed the process to normalize our schema to the highest form (BCNF) by splitting the tables and avoiding redundant data.

Assumptions:

- Of the above schema how the data is being sent/received to SHEMS by smart devices was not a concern at this level.
- The time interval that every electronic smart device such as AC, Refrigerator, Dryer and light will record the data(energy consumed) to SHEMS every **10 minutes** with the **'energy use'** as the event_label continuously till switched off.
- We are considering the energy consumption of every device if the event_label is 'energy use' and the value associated with the record is the amount of energy used in KWH.
- We consider other records as **events** such (switched off, switched on, temp lowered, temp increased, door open, and door closed). For these records the event_value is NULL/0. It means that we are storing this as an event that occurs in b/w the 10-minute interval.
- If the energy consumption is high due to any event that occurs, particularly for the refrigerator when the door opens and closes. We assume the smart devices are capable of calculating the additional energy used until the effect of the particular event is over.
- If the door opens in the middle of 12:10 am - 12:20 am @ 12:18 am, then the door closes at @12:22 am. The data will trigger as **12: 18 am - 'door opened' - NULL → 12: 20 am - 'energy use' - 0.67 KWH → 12:22 am - 'door closed' - NULL → 12: 30 am - 'energy use' - 0.89KWH** and follows... Likewise applicable to AC systems as well when the temp lowers or temp increases.
- We assume that the data would be preloaded for each zipcode for every hour of the day along with the price information. The table would be used for reference for calculating the energy prices. The energy prices could be loaded from external sources based on the demand and supply of the energy.
- We assume that there may be a failure detection system in all smart devices considering the minimum amount of power required to send the data when the device is switched off, recording it as an event in the DeviceData table.
- We assume that every customer would have one billing address associated with him/her which can be used as a service location address.
- We make an assumption that the database system would be capable enough to handle the traffic of storing the sensor data into the respective table without DB system failures.

ER Diagram:



Our **ER diagram** essentially contains 6 entities among which 5 are related to each other in some way and 1 entity (EnergyPrices) is not schematically related to other entities directly.

- *Customer* entity and *Address* entity are connected with each other in a 1:1 relationship as we assume that each customer would have one billing address associated with it.
- *Customer* entity and *ServiceLocations* entity have a 1:N cardinality relationship as one customer can create and manage multiple service locations and each service location would be associated with only one *customer*. Moreover, all service location records would have a customer entity associated with it, hence providing total participation.
- *ServiceLocations* entity and *Address* entity have a 1:1 cardinality relationship as every service location would have one unique address associated with it from the address table.
- *ServiceLocations* entity connects with the *EnrolledDevices* entity and has 1:N cardinality relationship as for each service location, we can define multiple devices.
- *EnrolledDevices* entity gets all the devices from a pre-stored *Device* entity which has 1:N cardinal relationship because we can enroll multiple devices from the Device table.
- The *EnrolledDevices* entity also connects with the *DeviceData* entity which has a 1:N cardinality relationship as the device data table receives multiple records associated with the same enrolled device.

- The *EnergyPrices* entity contains hourly energy prices based on zipcode. As the energy prices are based on zipcode and each zipcode can constitute multiple addresses it cannot directly be related to any other entity available in the ER diagram space, thus essentially serving as a reference table consisting of static data about the energy prices. Hence, we've not linked the EnergyPrices entity with any other entity.

(b) Use a relational database system to create the schema, together with key, foreign key, and other constraints.

Following are the queries designing our schema:

```
CREATE TABLE Address(
        address_id SERIAL PRIMARY KEY,
        unit_number varchar(50) NOT NULL,
        address_line varchar(100) NOT NULL,
        city varchar(100) NOT NULL,
        state varchar(2) NOT NULL,
        zipcode int NOT NULL
);


CREATE TABLE Customers(
        customer_id SERIAL PRIMARY KEY,
        first_name varchar(100) NOT NULL,
        last_name varchar(100) NOT NULL,
        billing_address_id INT NOT NULL,
        FOREIGN KEY(billing_address_id) REFERENCES Address(address_id)
);



CREATE TABLE ServiceLocations(
        sl_id SERIAL PRIMARY KEY,
        customer_id INT,
        address_id INT UNIQUE,
        start_date date,
        apt_area decimal(18,6),
        num_bedrooms INT,
        num_occupants INT,
        FOREIGN KEY(customer_id) REFERENCES Customers(customer_id),
        FOREIGN KEY(address_id) REFERENCES Address(address_id)
);
```

```sql
CREATE TABLE Device(
        model_id varchar(100) PRIMARY KEY,
        device_type varchar(100) NOT NULL,
        model_number varchar(100) NOT NULL,
        brand varchar(100),
        release_year INT,
        manufacture_year INT
);
```

```sql
CREATE TABLE EnrolledDevices(
        enrolled_device_id SERIAL PRIMARY KEY,
        sl_id INT NOT NULL,
        model_id varchar(64) NOT NULL,
        FOREIGN KEY(sl_id) REFERENCES ServiceLocations(sl_id),
        FOREIGN KEY(model_id) REFERENCES Device(model_id)
);
```

```sql
CREATE TABLE DeviceData(
        data_id SERIAL PRIMARY KEY,
        enrolled_device_id INT NOT NULL,
        event_timestamp timestamp,
        event_label varchar(100) NOT NULL,
        event_value decimal(18,6),
        FOREIGN KEY(enrolled_device_id) REFERENCES EnrolledDevices(enrolled_device_id)
);
```

```sql
CREATE TABLE EnergyPrices(
        zipcode INT NOT NULL,
        price_timestamp timestamp NOT NULL,
        price_per_kwh decimal(18,6) NOT NULL,
        PRIMARY KEY(zipcode, price_timestamp)
);
```

SQL Queries:

1. **List all enrolled devices with their total energy consumption in the last 24 hours, for a specific customer identified by customer ID.**

```sql
SELECT
 DD.enrolled_device_id as Enrolled_Device_Id,
 D.device_type as Device_Type,
 SUM(DD.event_value) AS Total_energy_consumption
 FROM
 DeviceData DD
 JOIN
```

```
EnrolledDevices E ON DD.enrolled_device_id = E.enrolled_device_id
JOIN
Device D ON E.model_id = D.model_id
JOIN
ServiceLocations SL ON E.sl_id = SL.sl_id
WHERE
SL.customer_id = 1  -- Replace with the specific customer ID
AND DD.event_label = 'energy use'
AND DD.event_timestamp BETWEEN NOW() - INTERVAL '24 hours' AND NOW()
GROUP BY
DD.enrolled_device_id, D.device_type;
```

```
SELECT
 DD.enrolled_device_id as Enrolled_Device_Id,
 D.device_type as Device_Type,
 SUM(DD.event_value) AS Total_energy_consumption
 FROM
 DeviceData DD
 JOIN
 EnrolledDevices E ON DD.enrolled_device_id = E.enrolled_device_id
 JOIN
 Device D ON E.model_id = D.model_id
 JOIN
 ServiceLocations SL ON E.sl_id = SL.sl_id
 WHERE
 SL.customer_id = 1  -- Replace with the specific customer ID
 AND DD.event_label = 'energy use'
 AND DD.event_timestamp BETWEEN NOW() - INTERVAL '24 hours' AND NOW()
 GROUP BY
 DD.enrolled_device_id, D.device_type;
```

Data Output | Messages | Notifications

	enrolled_device_id integer	device_type character varying (100)	total_energy_consumption numeric
1	1	Bulb	0.900000
2	2	Bulb	1.200000
3	3	AC System	0.600000

Explanation: To retrieve the energy consumption of the enrolled devices, we first join the respective tables from *DeviceData*, *EnrolledDevices* and *ServiceLocations* and filter out the rows thus joining only the filtered records of those tables. The filtered conditions are basically the customer_id provided, *event_label* set to 'energy use' and only the energy consumption in the past 24 hours. Finally, as we need the total energy per device we use a group by command by device_id and device_type to get the total energy for those devices.

2. **Calculate the average monthly energy consumption per device type, for the month of August 2022, considering only devices that have been on (i.e., reported data) at least once during that month.**

```
SELECT
D.device_type,
AVG(DD.event_value) AS avg_monthly_energy_consumption
FROM
DeviceData DD
JOIN
```

```
EnrolledDevices ED ON DD.enrolled_device_id = ED.enrolled_device_id
JOIN
Device D ON ED.model_id = D.model_id
WHERE
DD.event_label = 'energy use'
AND DD.event_timestamp >= '2022-08-01'  -- Start of August
AND DD.event_timestamp < '2022-09-01'   -- Start of September
GROUP BY
D.device_type
```

```
SELECT
D.device_type,
AVG(DD.event_value) AS avg_monthly_energy_consumption
FROM
DeviceData DD
JOIN
EnrolledDevices ED ON DD.enrolled_device_id = ED.enrolled_device_id
JOIN
Device D ON ED.model_id = D.model_id
WHERE
DD.event_label = 'energy use'
AND DD.event_timestamp >= '2022-08-01'  -- Start of August
AND DD.event_timestamp < '2022-09-01'   -- Start of September
GROUP BY
D.device_type

Select * from servicelocations
Select * from EnrolledDevices where sl_id in (1, 2)
```

Data Output | Messages | Notifications

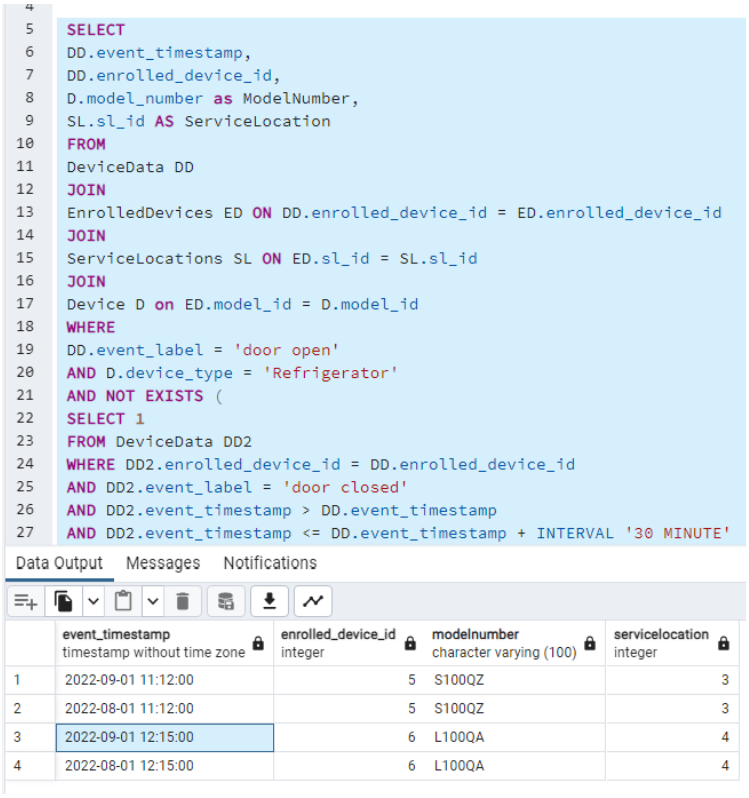
	device_type character varying (100)	avg_monthly_energy_consumption numeric
1	AC System	0.40000000000000000000
2	Bulb	0.80000000000000000000
3	Refrigerator	0.88750000000000000000

Explanation: We will get energy used data from (device data joins Enrolled Devices joins Device) tables, then we will filter the aug dates, calculating the avg energy consumption based on the grouping of device_type attribute.

3. **Identify cases where a refrigerator door was left open for more than 30 minutes. Output the date and time, the service location, the device ID, and the refrigerator model.**

```
SELECT
 DD.event_timestamp,
 DD.enrolled_device_id,
 D.model_number as ModelNumber,
 SL.sl_id AS ServiceLocation
 FROM
 DeviceData DD
 JOIN
 EnrolledDevices ED ON DD.enrolled_device_id = ED.enrolled_device_id
 JOIN
 ServiceLocations SL ON ED.sl_id = SL.sl_id
 JOIN
 Device D on ED.model_id = D.model_id
 WHERE
 DD.event_label = 'door open'
```

```
AND D.device_type = 'Refrigerator'
AND NOT EXISTS (
SELECT 1
FROM DeviceData DD2
WHERE DD2.enrolled_device_id = DD.enrolled_device_id
AND DD2.event_label = 'door closed'
AND DD2.event_timestamp > DD.event_timestamp
AND DD2.event_timestamp <= DD.event_timestamp + INTERVAL '30 MINUTE'
)
```



```
SELECT
DD.event_timestamp,
DD.enrolled_device_id,
D.model_number as ModelNumber,
SL.sl_id AS ServiceLocation
FROM
DeviceData DD
JOIN
EnrolledDevices ED ON DD.enrolled_device_id = ED.enrolled_device_id
JOIN
ServiceLocations SL ON ED.sl_id = SL.sl_id
JOIN
Device D on ED.model_id = D.model_id
WHERE
DD.event_label = 'door open'
AND D.device_type = 'Refrigerator'
AND NOT EXISTS (
SELECT 1
FROM DeviceData DD2
WHERE DD2.enrolled_device_id = DD.enrolled_device_id
AND DD2.event_label = 'door closed'
AND DD2.event_timestamp > DD.event_timestamp
AND DD2.event_timestamp <= DD.event_timestamp + INTERVAL '30 MINUTE'
```

Data Output | Messages | Notifications

	event_timestamp timestamp without time zone	enrolled_device_id integer	modelnumber character varying (100)	servicelocation integer
1	2022-09-01 11:12:00	5	S100QZ	3
2	2022-08-01 11:12:00	5	S100QZ	3
3	2022-09-01 12:15:00	6	L100QA	4
4	2022-08-01 12:15:00	6	L100QA	4

Explanation: The outer query in the above first determines whether the event "door opened" exists. If yes, the inner query then determines whether the event "door closed" occurs within thirty minutes of the original event (door opened). If this is the case, the outer query event is ignored; otherwise, it is taken into consideration.

4. **Calculate the total energy cost for each service location during August 2022, considering the hourly changing energy prices based on zip code.**

```
SELECT
 SL.sl_id,
 SUM(DD.event_value * EP.price_per_kwh) AS "TotalEnergyCost"
 FROM
 ServiceLocations SL
 JOIN
 Address A on A.address_id = SL.address_id
 JOIN
 EnrolledDevices ED ON SL.sl_id = ED.sl_id
 JOIN
```

```
 DeviceData DD ON ED.enrolled_device_id = DD.enrolled_device_id
 JOIN
 EnergyPrices EP ON A.zipcode = EP.zipcode AND extract(hour from DD.event_timestamp) =
extract(hour from EP.price_timestamp)
 WHERE
        DD.event_label = 'energy use' and
        DD.event_timestamp >= '2022-08-01' and -- Start of August
     DD.event_timestamp < '2022-09-01'    -- Start of September
 GROUP BY
 SL.sl_id
```

```
4
5  SELECT
6   SL.sl_id,
7   SUM(DD.event_value * EP.price_per_kwh) AS "TotalEnergyCost"
8   FROM
9   ServiceLocations SL
10  JOIN
11  Address A on A.address_id = SL.address_id
12  JOIN
13  EnrolledDevices ED ON SL.sl_id = ED.sl_id
14  JOIN
15  DeviceData DD ON ED.enrolled_device_id = DD.enrolled_device_id
16  JOIN
17  EnergyPrices EP ON A.zipcode = EP.zipcode AND extract(hour from DD.event_timestamp) = extract(hour from EP.price_timest
18  WHERE
19      DD.event_label = 'energy use' and
20      DD.event_timestamp >= '2022-08-01' and -- Start of August
21      DD.event_timestamp < '2022-09-01'   -- Start of Septembe
22  GROUP BY
23  SL.sl_id
24
```

Data Output | Messages | Notifications

	sl_id [PK] integer	TotalEnergyCost numeric
1	1	3.056000000000
2	2	0.976000000000
3	3	6.033000000000
4	4	13.731000000000

Explanation: We get the total energy cost by multiplying the energy consumed with the energy price based on the zipcode. We join the energy prices table based on zipcode and the hourly timestamp column with the device data timestamp column and the address zipcode column. Once after we join the necessary tables we filter to get only *event_label* = 'energy_use' and only the energy cost for the month of August. In the end we group by based on the service location to get the total cost for that particular service location.

5. **For each service location, compute its total energy consumption during August 2022, as a percentage of the average total energy consumption during the same time of other service locations that have a similar square footage (meaning, at most 5% higher or lower square footage). Thus, you would output 150% if a service location with 1000 sqft had 50% higher energy consumption than the average of other service locations that have between 950 and 1050 sqft.**

```
WITH
   EnergyConsumptionByLocation AS (
```

```
        SELECT
            ED.sl_id,
            SUM(DD.event_value) AS total_energy_consumption
        FROM
            EnrolledDevices ED
        JOIN
            DeviceData DD ON DD.enrolled_device_id = ED.enrolled_device_id
        JOIN
            ServiceLocations SL ON SL.sl_id = ED.sl_id
        WHERE
             DD.event_label = 'energy use'
               AND DD.event_timestamp >= '2022-08-01'  -- Start of August
               AND DD.event_timestamp < '2022-09-01'   -- Start of September
        GROUP BY
            ED.sl_id
    ),
       AugECL AS (
              Select ECL.*, SL.apt_area
              FROM
                     EnergyConsumptionByLocation ECL
              JOIN
                     ServiceLocations SL on SL.sl_id = ECL.sl_id
       ),
    AvgPower AS (
        SELECT
            A1.sl_id,
            AVG(A2.total_energy_consumption) AS avg_energy_consumed
        FROM
            AugECL A1
        JOIN
            AugECL A2 on A1.apt_area >= 0.95 * A2.apt_area and A1.apt_area <= 1.05 *
A2.apt_area
        GROUP BY
            A1.sl_id
    )
 SELECT
    ECL.sl_id,
    case
       when (AP.avg_energy_consumed = 0) then 0
       else (ECL.total_energy_consumption / AP.avg_energy_consumed) * 100
       end AS Percentage_of_Avg
FROM
    EnergyConsumptionByLocation ECL
LEFT JOIN
    AvgPower AP ON ECL.sl_id = AP.sl_id;
```

```
            DD.event_label = 'energy use'
            AND DD.event_timestamp >= '2022-08-01'  -- Start of August
            AND DD.event_timestamp < '2022-09-01'   -- Start of September
        GROUP BY
            ED.sl_id
    ),
    AugECL AS (
        Select ECL.*, SL.apt_area
        FROM
            EnergyConsumptionByLocation ECL
        JOIN
            ServiceLocations SL on SL.sl_id = ECL.sl_id
    ),
    AvgPower AS (
        SELECT
            A1.sl_id,
            AVG(A2.total_energy_consumption) AS avg_energy_consumed
        FROM
            AugECL A1
        JOIN
            AugECL A2 on A1.apt_area >= 0.95 * A2.apt_area and A1.apt_area <= 1.05 * A2.apt_area
        GROUP BY
            A1.sl_id
    )
SELECT
    ECL.sl_id,
    case
```

Data Output | Messages | Notifications

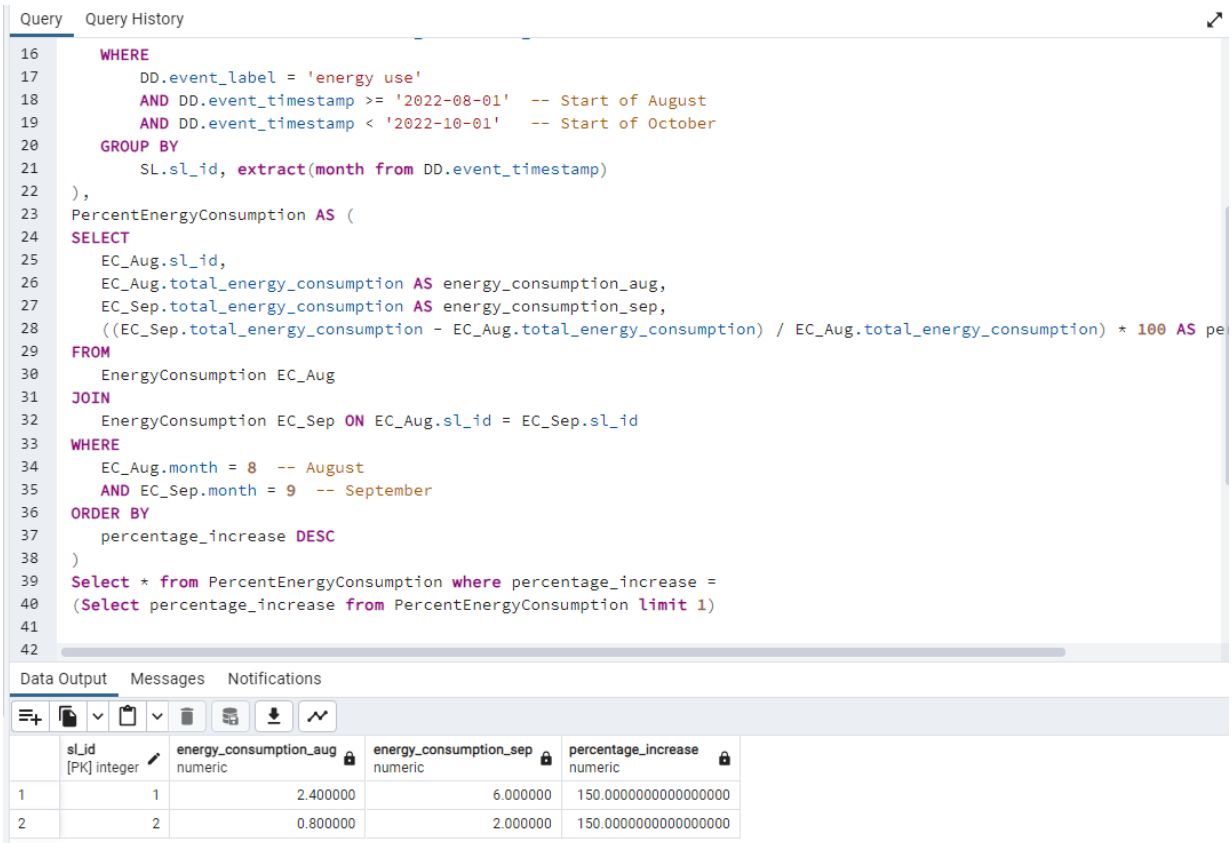
	sl_id integer	percentage_of_avg numeric
1	1	46.601941747572815534 00
2	2	50.000000000000000000
3	3	100.000000000000000000
4	4	167.2354948805460800

Explanation: In this particular query, we divide the operations into multiple steps. We first calculate the energy consumption by the service location for every service location. After this step, we then join the *apt_area* column to the previous dataset which is required for our further analysis. As part of the next step, we then find the average energy consumed for properties with similar area with a 5% threshold above and below the limits. Finally, we then calculate the percentage of average by dividing the total energy consumption by average energy consumption on similar properties.

6. Identify service location(s) that had the highest percentage increase in energy consumption between August and September of 2022.

```
WITH EnergyConsumption AS (
    SELECT
        SL.sl_id,
        extract(month from DD.event_timestamp) AS month,
        SUM(DD.event_value) AS total_energy_consumption
    FROM
        DeviceData DD
    JOIN
        EnrolledDevices ED ON DD.enrolled_device_id = ED.enrolled_device_id
    JOIN
        ServiceLocations SL ON ED.sl_id = SL.sl_id
    WHERE
```

```
        DD.event_label = 'energy use'
        AND DD.event_timestamp >= '2022-08-01'  -- Start of August
        AND DD.event_timestamp < '2022-10-01'   -- Start of October
    GROUP BY
        SL.sl_id, extract(month from DD.event_timestamp)
 ),
 PercentEnergyConsumption AS (
 SELECT
    EC_Aug.sl_id,
    EC_Aug.total_energy_consumption AS energy_consumption_aug,
    EC_Sep.total_energy_consumption AS energy_consumption_sep,
    ((EC_Sep.total_energy_consumption - EC_Aug.total_energy_consumption) /
EC_Aug.total_energy_consumption) * 100 AS percentage_increase
 FROM
    EnergyConsumption EC_Aug
 JOIN
    EnergyConsumption EC_Sep ON EC_Aug.sl_id = EC_Sep.sl_id
 WHERE
    EC_Aug.month = 8  -- August
    AND EC_Sep.month = 9  -- September
 ORDER BY
    percentage_increase DESC
 )
 Select * from PercentEnergyConsumption where percentage_increase =
 (Select percentage_increase from PercentEnergyConsumption limit 1)
```



Explanation: In this query, we firstly calculate the total energy consumption for the month of August and September by filtering based on datetime and event_label of only 'energy use'. We then calculate the percentage increase by calculating the fractional increase from the previous month and multiplying by 100 (monthly_consumption in Sep - monthly_consumption in Aug) / monthly_consumption in Aug. Finally we display only the records which have the highest percentage_increase value.

Sample Data:

Address Table

```
Insert into Address(unit_number, address_line, city, state, zipcode)
values('2F', '123 Main St', 'Brooklyn', 'NY', 12345),
         ('Apt 20', '230 Nevins St', 'Brooklyn', 'NY', 12347),
         ('1F', '1120 67th Street', 'Brooklyn', 'NY', 12358),
         ('1F', '12340 86th Street', 'Brooklyn', 'NY', 12358),
         ('3F', '1250 Grove St', 'Brooklyn', 'NY', 12345)
        ('2F', '123 Mcarther St', 'Manhattan', 'NY',11309),
         ('Apt 20', '230 Nevins St', 'StatenIsland', 'NY', 11809),
         ('1F', '1120 67th Street', 'Queens', 'NY', 12367),
         ('1F', '12340 86th Street', 'Bronx', 'NY', 12987),
         ('3F', '1250 10th St', 'Manhattan', 'NY', 12389)
```

Customer Table

```
Insert into Customers(first_name, last_name, billing_address_id)
values('John','Doe', 1), ('Jane','Smith', 3), ('Rakesh','R', 2),
('Jack','Ryan', 2), ('Rebba','John', 4),('Samantha','Ruth Prabhu', 5),
('Tom','Cruise', 4), ('Klin','John', 7),('Nayantara','S', 8)
```

ServiceLocations Table

```
Insert into
ServiceLocations(customer_id,address_id,start_date,apt_area,num_bedrooms,num_occupants)
values(1, 1, '2022-08-01',1000, 2, 3),
         (1, 2, '2022-08-10',1050, 2, 1),
         (2, 3, '2022-07-30',1200, 3, 5),
         (2, 4, '2022-08-15',980, 1, 2),
         (2, 5, '2022-08-16',1160, 2, 2),
         (4, 6, '2022-09-30',1200, 5, 2),
         (5, 7, '2022-10-15',980, 3, 2),
         (6, 8, '2022-11-16',1160, 4, 5)
```

Device

```
Insert into Device(model_id,device_type,model_number,brand,release_year,manufacture_year)
values('model1','Refrigerator','Cafe 100','GE',2010, 2010),
         ('model2','AC System','A100','GE',2011, 2011),
         ('model3','Refrigerator','S100QZ','Samsung',2011, 2011),
         ('model4','Refrigerator','L100QA','LG',2011, 2011),
```

```
('model5','AC System','S100AC','Samsung',2011, 2012),
('model6','Dryer','D101','GE',2015, 2015),
('model7','Bulb','SM1','Philips',2016, 2016),
('model8','Bulb','SB1234','Samsung',2017, 2017),
('model9','Bulb','A101','Apple',2018, 2018),
('model10','Dryer','D1938L','LG',2016, 2016)
```

EnrolledDevices

```
Insert into EnrolledDevices(sl_id, model_id)
values(1, 'model7'), - Bulb 1
         (1, 'model8'), - Bulb 2
         (2, 'model2'), - AC 3
         (3, 'model5'), - AC 4
         (3, 'model3'), - Refrigerator 5
         (4, 'model4') - Refrigerator 6
```

DeviceData

```
Insert into DeviceData(enrolled_device_id,event_timestamp,event_label,event_value)
values (1,'2023-12-01 10:30:00', 'energy use', 0.9),
        (2,'2023-12-01 09:30:00', 'energy use', 0.4),
        (3,'2023-12-01 11:30:00', 'energy use', 0.2),
        (4,'2023-12-01 02:30:00', 'energy use', 0.33),
         (2,'2023-12-01 09:40:00', 'energy use', 0.4),
        (3,'2023-12-01 11:40:00', 'energy use', 0.2),
        (4,'2023-12-01 02:40:00', 'energy use', 0.33),
         (2,'2023-12-01 09:50:00', 'energy use', 0.4),
        (3,'2023-12-01 11:50:00', 'energy use', 0.2),
        (4,'2023-12-01 02:50:00', 'energy use', 0.33),
         (1,'2022-08-01 10:30:00', 'switched on', NULL),
         (1,'2022-08-01 10:40:00', 'energy use', 0.8),
         (1,'2022-08-01 10:50:00', 'energy use', 0.8),
         (1,'2022-08-01 11:00:00', 'energy use', 0.8),
        (1,'2022-08-01 11:00:00', 'switched off', NULL),
         (3,'2022-08-01 11:28:00', 'switched on', NULL),
         (3,'2022-08-01 11:29:00', 'temp lowered', 23.6),
         (3,'2022-08-01 11:30:00', 'energy use', 0.6),
         (6,'2022-08-01 11:33:00', 'switched on', NULL),
        (6,'2022-08-01 11:40:00', 'energy use', 0.9),
         (6,'2022-08-01 11:50:00', 'energy use', 0.9),
        (3,'2022-08-01 11:50:00', 'energy use', 0.2),
         (3,'2022-08-01 11:50:00', 'switched off', NULL),
         (6,'2022-08-01 12:00:00', 'energy use', 0.9),
         (6,'2022-08-01 12:10:00', 'energy use', 0.9),
         (6,'2022-08-01 12:15:00', 'door open', NULL),
         (6,'2022-08-01 12:20:00', 'energy use', 0.99),
         (6,'2022-08-01 12:30:00', 'energy use', 0.99),
         (6,'2022-08-01 12:40:00', 'energy use', 1.02),
         (6,'2022-08-01 12:50:00', 'energy use', 1.05),
         (6,'2022-08-01 12:50:00', 'door closed', NULL),
         (6,'2022-08-01 12:50:00', 'energy use', 1.0),
```

```
(6,'2022-08-01 13:00:00', 'energy use', 0.9),
(6,'2022-08-01 13:10:00', 'energy use', 0.9),
(6,'2022-08-01 13:20:00', 'energy use', 0.9),
(6,'2022-08-01 13:30:00', 'energy use', 0.9),
(6,'2022-08-01 13:33:00', 'door open', NULL),
(6,'2022-08-01 13:38:00', 'door closed', NULL),
(6,'2022-08-01 13:38:00', 'switched off', NULL),
(5,'2022-08-01 10:30:00', 'switched on', NULL),
(5,'2022-08-01 10:40:00', 'energy use', 0.7),
(5,'2022-08-01 10:50:00', 'energy use', 0.7),
(5,'2022-08-01 11:00:00', 'energy use', 0.7),
(5,'2022-08-01 11:10:00', 'energy use', 0.7),
(5,'2022-08-01 11:12:00', 'door open', NULL),
(5,'2022-08-01 11:20:00', 'energy use', 0.9),
(5,'2022-08-01 11:30:00', 'energy use', 0.9),
(5,'2022-08-01 11:40:00', 'energy use', 0.9),
(5,'2022-08-01 11:43:00', 'door closed', NULL),
(1,'2022-09-01 10:30:00', 'switched on', NULL),
(1,'2022-09-01 10:40:00', 'energy use', 1.2),
(1,'2022-09-01 10:50:00', 'energy use', 1.2),
(1,'2022-09-01 11:00:00', 'energy use', 1.2),
(1,'2022-09-01 11:00:00', 'switched off', NULL),
(3,'2022-09-01 11:28:00', 'switched on', NULL),
(3,'2022-09-01 11:29:00', 'temp increased', 25.6),
(3,'2022-09-01 11:30:00', 'energy use', 1.0),
(6,'2022-09-01 11:33:00', 'switched on', NULL),
(6,'2022-09-01 11:40:00', 'energy use', 1.4),
(6,'2022-09-01 11:50:00', 'energy use', 1.4),
(3,'2022-09-01 11:50:00', 'energy use', 1.0),
(3,'2022-09-01 11:50:00', 'switched off', NULL),
(6,'2022-09-01 12:00:00', 'energy use', 1.4),
(6,'2022-09-01 12:10:00', 'energy use', 1.3),
(6,'2022-09-01 12:15:00', 'door open', NULL),
(6,'2022-09-01 12:20:00', 'energy use', 1.4),
(6,'2022-09-01 12:30:00', 'energy use', 1.4),
(6,'2022-09-01 12:40:00', 'energy use', 1.4),
(6,'2022-09-01 12:50:00', 'energy use', 1.4),
(6,'2022-09-01 12:50:00', 'door closed', NULL),
(6,'2022-09-01 12:50:00', 'energy use', 1.4),
(6,'2022-09-01 13:00:00', 'energy use', 1.45),
(6,'2022-09-01 13:10:00', 'energy use', 1.45),
(6,'2022-09-01 13:20:00', 'energy use', 1.46),
(6,'2022-09-01 13:30:00', 'energy use', 1.48),
(6,'2022-09-01 13:33:00', 'door open', NULL),
(6,'2022-09-01 13:38:00', 'door closed', NULL),
(6,'2022-09-01 13:38:00', 'switched off', NULL),
(5,'2022-09-01 10:30:00', 'switched on', NULL),
(5,'2022-09-01 10:40:00', 'energy use', 0.5),
(5,'2022-09-01 10:50:00', 'energy use', 0.5),
(5,'2022-09-01 11:00:00', 'energy use', 0.5),
(5,'2022-09-01 11:10:00', 'energy use', 0.5),
```

```
(5,'2022-09-01 11:12:00', 'door open', NULL),
(5,'2022-09-01 11:20:00', 'energy use', 0.7),
(5,'2022-09-01 11:30:00', 'energy use', 0.7),
(5,'2022-09-01 11:40:00', 'energy use', 0.7),
(5,'2022-09-01 11:43:00', 'door closed', NULL),
(2,'2022-09-01 11:42:00', 'switched on', NULL),
(2,'2022-09-01 11:50:00', 'energy use', 0.6),
(2,'2022-09-01 12:00:00', 'energy use', 0.6),
(2,'2022-09-01 12:10:00', 'energy use', 0.6),
(2,'2022-09-01 12:20:00', 'energy use', 0.6),
(2,'2022-09-01 12:20:00', 'switched off', 0.6);
```

Energy Prices:

```
INSERT INTO EnergyPrices (zipcode, price_timestamp, price_per_kwh)
VALUES
(12345, '2022-08-01 12:00:00', 1.15),
(12347, '2022-08-01 12:00:00', 1.12),
(12358, '2022-08-01 12:00:00', 1.14),
(12345, '2022-08-01 13:00:00', 0.9),
(12347, '2022-08-01 13:00:00', 0.7),
(12358, '2022-08-01 13:00:00', 1.08),
(12345, '2022-08-01 09:00:00', 1.6),
(12347, '2022-08-01 09:00:00', 1.11),
(12358, '2022-08-01 09:00:00', 1.09),
(12345, '2022-08-01 11:00:00', 1.34),
(12347, '2022-08-01 11:00:00', 1.22),
(12358, '2022-08-01 11:00:00', 1.13),
(12345, '2022-08-01 10:00:00', 1.24),
(12347, '2022-08-01 10:00:00', 1.12),
(12358, '2022-08-01 10:00:00', 1.00);
```