Smart Home Database Project

Report

Curricular Unit - Base de Dados (BASDAD)

february 2024

Group 9 (Turma B)

André Ferreira

Inês Lemos

José Pedro Rodrigues

Miguel Couto

Pedro Siqueira

Entity-Relationship Diagram

(Annex A)

Our database model is designed to comprehensively manage a smart home ecosystem, focusing on the interconnectivity between residents, devices, and the physical layout of homes. The structure facilitates the tracking and control of various aspects, such as device usage, energy production, and consumption, while also accommodating the roles and permissions of different residents within the household.

At the core of the model are the **Resident** and **House** tables, which store information about the individuals living in the homes and the homes themselves, including location details refined by the **GPS_Location** table. The model ensures data integrity and logical consistency through constraints, such as ensuring valid email formats and geographical coordinates within acceptable ranges.

The **Device**, **Sensor**, and **Actuator** tables, along with their respective type tables (**Device_Type**, **Sensor_Type**, **Actuator_Type**), capture the diverse range of smart devices within a home, from generic appliances to specialized sensors and actuators. Each device is associated with a specific room and can be linked to operational programs that dictate their behavior.

Energy management is an important aspect of the model, addressed by the **Power_Source** and **Energy_Production** tables. These tables allow for the monitoring of energy sources within the home, such as solar panels, including their capacity, output, and the periods during which energy is produced.

The **Role**, **Permission**, and **Resident_Role** tables establish a flexible permission system, enabling the assignment of different roles to residents, each with specific permissions that govern their interaction with the system. This setup enhances security and customization by controlling access to various functionalities based on the resident's role.

The **Active_Periods** table is pivotal for tracking the usage of devices, detailing when and by whom devices were activated, along with the energy consumed during these periods. This data is vital for understanding household energy consumption patterns and optimizing energy usage.

The **Characteristics** table enriches our understanding of devices by detailing specific attributes, such as size or capacity, crucial for tailoring device functionality to suit various needs and preferences. The **Readings** table stands as a cornerstone of the system's intelligence, capturing real-time data from sensors—this could range from temperature fluctuations to energy consumption rates, providing a foundation for responsive and adaptive home automation.

The **Room_Owner** table assigns each room a responsible resident, ensuring clear accountability for the room's devices and settings. This structure allows for personalized management and enhances the efficiency of household operations by delineating ownership within the home's shared spaces.

Together, these tables contribute significantly to the system's versatility and effectiveness, supporting a wide range of functionalities from basic automation to advanced energy management and personalized user experiences.

Query a)

List for the houses/apartments in the cities of Braga and Porto all the connected devices. The result should be sorted alphabetically by the city name and in descending order by device type.

```
SELECT h.houseID, h.city, dt.name AS deviceType
FROM device_type dt
INNER JOIN device d ON dt.deviceTypeID = d.deviceTypeID
INNER JOIN room r ON d.roomID = r.roomID
INNER JOIN house h ON r.houseID = h.houseID
WHERE h.city IN ('Braga', 'Porto')
ORDER BY h.city, dt.name DESC;
```

Functional explanation: This query is designed to retrieve a list of houses with connected devices, but only for those located in the cities of Braga and Porto. It accomplishes this by joining several tables: device_type, device, room, and house. The WHERE clause filters the results to only include houses from Braga and Porto. Finally, the results are organized first by the city name in ascending order, and then by the device type in descending order, ensuring a structured and easy-to-read output of device types within houses from these two cities.

Result:

HOUSEID	CITY	DEVICETYPE
1	Braga	Smart Thermostat
1	Braga	Smart Thermostat
2	Braga	Smart Thermostat
2	Braga	Smart TV
1	Braga	Smart Plug
2	Braga	Smart Lock
1	Braga	Smart Gas Detector
2	Braga	Security Camera
1	Braga	Light Bulb
1	Braga	Light Bulb
3	Porto	Smart Thermostat
3	Porto	Smart Oven
3	Porto	Smart Coffee Maker
3	Porto	Security Camera
3	Porto	Electric Bike

Query b)

List the names of all rooms that have devices that have been turned off for more than 48 hours.

```
SELECT DISTINCT r.roomID, r.name
FROM room r
INNER JOIN device d ON r.roomID = d.roomID
INNER JOIN active_periods ap ON d.deviceID = ap.deviceID
WHERE ap.endTime < (CURRENT_TIMESTAMP - INTERVAL '48' HOUR);
```

Functional explanation: This query is designed to extract a unique list of room names where devices have been inactive for at least 48 hours. It systematically links the room, device, and active_periods tables to track device activity within rooms. The core of this query lies in its WHERE clause, which filters out devices whose last active period (endTime) falls more than 48 hours prior to the current moment. By selecting distinct room names, the query ensures that each room is listed only once, highlighting locations with devices that have recently been inactive, all without repeating roomID or delving into individual device details.

Result:

ROOMID	NAME
7	Living Room
8	Kitchen
10	Living Room
13	Bedroom
3	Toilet
15	Kitchen
20	Garage
1	Living Room
5	Dining Room
22	Garage
4	Living Room
6	Bedroom
2	Kitchen
9	Bedroom
21	Garage

Query c)

List the name of the user in the house who has turned on more devices than all users in the houses/ apartments.

```
SELECT res.name
FROM resident res
JOIN active_periods ap ON ap.userIDTurnedON = res.numCC
JOIN resident_role rr ON rr.numCC = res.numCC
JOIN house h ON h.houseID = rr.houseID
WHERE h.city <> 'Porto'
GROUP BY res.name
HAVING COUNT(ap.userIDTurnedON) > (
    SELECT COUNT(ap_sub.userIDTurnedON)
    FROM active_periods ap_sub
    JOIN resident r_sub ON ap_sub.userIDTurnedON = r_sub.numCC
    JOIN resident_role rr_sub ON rr_sub.numCC = r_sub.numCC
    JOIN house h_sub ON h_sub.houseID = rr_sub.houseID
    WHERE h_sub.city = 'Porto'
    GROUP BY r_sub.name
    ORDER BY COUNT(ap_sub.userIDTurnedON) DESC
    FETCH FIRST 1 ROW ONLY);
```

Functional explanation: This query identifies residents who have activated devices more often than the most active resident in Porto, excluding those in Porto. It links residents to device activations and houses, grouping by resident name to count activations. The key condition excludes Porto residents and compares each count against the highest device activation count in Porto, determined by a subquery. Only residents with activations exceeding this benchmark are listed, highlighting high device engagement outside Porto.

Result:

NAME
João Silva
Maria Oliveira
Carlos Pereira

Query d)

List for all devices that were turned on between 9 am and 10 am and turned off after 5 minutes, all their programs and active periods, as well as all the information about the house/apartment to which they belong.

```
SELECT DISTINCT d.deviceName, h.*, ap.*, p.*
FROM device d
INNER JOIN active_periods ap ON d.deviceID = ap.deviceID
INNER JOIN program p ON ap.programID = p.programID
INNER JOIN room r ON d.roomID = r.roomID
INNER JOIN house h ON r.houseID = h.houseID
WHERE EXISTS (
    SELECT 1
    FROM active_periods ap_check
    WHERE ap_check.deviceID = d.deviceID
    AND EXTRACT(HOUR FROM ap_check.startTime) BETWEEN 9 AND 10
    AND ap_check.endTime = ap_check.startTime + INTERVAL '5' MINUTE);
```

Functional explanation: This query retrieves unique devices and their associated house, active period, and program details, specifically targeting devices activated between 9 and 10 AM for exactly 5 minutes. It joins the device, active_periods, program, room, and house tables to compile comprehensive information. The key filter, applied through a `WHERE EXISTS` subclause, identifies devices with the specified activation pattern, ensuring only those meeting this criteria are included in the final results.

Result:

DEVICENAME	HOUSEID	USERIDHOUSEOWNER	FLOORS	CONTRACTEDMAXPOWER	STREET	DOORNUMBER	APARTMENTNUMBER	ZIPCODE	CITY	GPSLOCATIONID	ACTIVEPERIODID	SCHEDULED	PROGRAMID	DEVICEID
SmartThermostat	1	11111111	2	10000	Street	654	AA	4890-076	Braga	5	101	1	1	1
SmartCoffeeMaker	3	33333333	2	12000	Downtown St	456	B	4000-002	Porto	2	29	1	7	9
SmartCoffeeMaker	3	33333333	2	12000	Downtown St	456	B	4000-002	Porto	2	69	1	7	9
SmartThermostat	1	11111111	2	10000	Street	654	AA	4890-076	Braga	5	21	1	6	1
SmartThermostat	1	11111111	2	10000	Street	654	AA	4890-076	Braga	5	102	1	1	1
SmartThermostat	1	11111111	2	10000	Street	654	AA	4890-076	Braga	5	11	1	6	1
SmartCoffeeMaker	3	33333333	2	12000	Downtown St	456	B	4000-002	Porto	2	9	1	7	9
SmartThermostat	1	11111111	2	10000	Street	654	AA	4890-076	Braga	5	31	1	6	1
SmartCoffeeMaker	3	33333333	2	12000	Downtown St	456	B	4000-002	Porto	2	19	1	7	9
SmartThermostat	1	11111111	2	10000	Street	654	AA	4890-076	Braga	5	103	1	1	1
SmartThermostat	1	11111111	2	10000	Street	654	AA	4890-076	Braga	5	1	1	6	1
SmartCoffeeMaker	3	33333333	2	12000	Downtown St	456	B	4000-002	Porto	2	99	0	7	9

DEVICEID	STARTTIME	ENDTIME	USERIDTURNEDON	ENERGYCONSUMEDKWH	PROGRAMID	DURATION	DESCRIPTION	DEVICETYPEID
1	02-JAN-23 02.41.59.000000 PM	02-JAN-23 03.44.59.000000 PM	11111111	1	1	30	Lighting Automation	1
9	10-MAR-23 09.30.00.000000 AM	10-MAR-23 10.30.00.000000 AM	88888888	15	7	20	Wake-Up Brew	5
9	01-JAN-24 09.30.00.000000 AM	01-JAN-24 09.35.00.000000 AM	88888888	15	7	20	Wake-Up Brew	5
1	10-SEP-23 06.00.00.000000 PM	10-SEP-23 07.30.00.000000 PM	11111111	5	6	20	Comfort Morning	5
1	05-JAN-23 02.18.19.000000 PM	05-JAN-23 03.44.19.000000 PM	11111111	1	1	30	Lighting Automation	1
1	10-NOV-23 06.00.00.000000 PM	10-NOV-23 07.30.00.000000 PM	11111111	5	6	20	Comfort Morning	5
9	07-JUL-23 09.30.00.000000 AM	07-JUL-23 09.35.00.000000 AM	88888888	15	7	20	Wake-Up Brew	5
1	10-JUL-23 10.00.00.000000 AM	10-JUL-23 11.15.00.000000 AM	11111111	6	6	20	Comfort Morning	5
9	10-MAY-23 09.30.00.000000 AM	10-MAY-23 10.30.00.000000 AM	77777777	15	7	20	Wake-Up Brew	5
1	06-JAN-23 10.29.35.000000 AM	06-JAN-23 11.40.35.000000 AM	11111111	1	1	30	Lighting Automation	1
1	15-FEB-23 09.00.00.000000 AM	15-FEB-23 09.05.00.000000 AM	11111111	5	6	20	Comfort Morning	5
9	01-JAN-23 10.00.00.000000 AM	02-FEB-23 11.00.00.000000 AM	33333333	900	7	20	Wake-Up Brew	5

Query e)

List the cities in which the total number of devices in each home is greater than the average number of devices in houses with two floors.

```
SELECT DISTINCT h.city
FROM house h
JOIN (
    SELECT h.houseID, COUNT(d.deviceID) AS dispositivos_por_casa
    FROM house h
    JOIN room r ON h.houseID = r.houseID
    JOIN device d ON r.roomID = d.roomID
    GROUP BY h.houseID
) casas_com_dispositivos ON h.houseID = casas_com_dispositivos.houseID
WHERE casas_com_dispositivos.dispositivos_por_casa > (
    SELECT AVG(dispositivos_por_casa)
    FROM (
        SELECT h.houseID, COUNT(d.deviceID) AS dispositivos_por_casa
        FROM house h
        JOIN room r ON h.houseID = r.houseID
        JOIN device d ON r.roomID = d.roomID
        GROUP BY h.houseID));
```

Functional explanation: This query identifies cities where houses have more devices than the average number of devices per house across all cities. It starts by joining the house, room, and device tables to count the number of devices in each house, grouping the results by house ID. This count is then used in a subquery to calculate the average number of devices per house across all houses. The outer query filters cities by selecting those houses whose device count exceeds this average. The use of DISTINCT ensures each city is listed only once, regardless of how many houses within it meet the criteria.

Result:

CITY
Lisbon
Braga
Porto

Query f)

List the house and the name of the owner that has devices with a nominal power lower than any device located in the rooms of the houses registered in the database.

```
SELECT h.houseID, r.name
FROM house h
JOIN resident r ON h.userIDHouseOwner = r.numCC
JOIN room ON h.houseID = room.houseID
JOIN device d ON room.roomID = d.roomID
JOIN device_type dt ON d.devicetypeID = dt.devicetypeID
JOIN characteristics ch ON dt.devicetypeID = ch.devicetypeID
WHERE ch.name = 'Nominal Power'
  AND ch.value = (
    SELECT min(characteristics.value)
    FROM characteristics
    WHERE characteristics.name = 'Nominal Power');
```

Functional explanation: This query is designed to identify houses along with their owners' names, where the devices have the lowest nominal power among all devices in the system. It cross-references tables for houses, residents, rooms, devices, and device characteristics to pinpoint those that match the criterion of having the minimal nominal power value, indicating a focus on energy-efficient devices within the household context.

Result:

HOUSEID	NAME
2	Maria Oliveira
5	Carlos Pereira

Query g)

List the names of the owners who have the lowest number of sensors installed in their homes and have the two devices that had the highest energy consumption in the last two years.

```
WITH device_energy_consumption AS (
   SELECT d.deviceID, sum(ap.energyConsumedkWh) AS totalEnergy
   FROM device d
   JOIN active_periods ap ON d.deviceID = ap.deviceID
   WHERE ap.startTime >= current_date - interval '2' YEAR
   GROUP BY d.deviceID
   ORDER BY totalEnergy DESC
   FETCH FIRST 2 ROWS ONLY
),
owners_top_devices AS (
   SELECT DISTINCT h.userIDHouseOwner
   FROM device_energy_consumption dec
   JOIN device d ON dec.deviceID = d.deviceID
   JOIN room r ON d.roomID = r.roomID
   JOIN house h ON r.houseID = h.houseID
),
sensor_count_per_owner AS (
   SELECT h.userIDHouseOwner, count(s.sensorID) AS sensorCount
   FROM house h
   JOIN room r ON h.houseID = r.houseID
   JOIN device d ON r.roomID = d.roomID
   JOIN sensor s ON d.deviceID = s.deviceID
   GROUP BY h.userIDHouseOwner
),
minimum_sensor_count_per_owner AS (
   SELECT min(scpo.sensorCount) AS minSensorCount
   FROM sensor_count_per_owner scpo
   JOIN owners_top_devices otd ON scpo.userIDHouseOwner = otd.userIDHouseOwner
)
SELECT res.name
FROM sensor_count_per_owner scpo
JOIN owners_top_devices otd ON scpo.userIDHouseOwner = otd.userIDHouseOwner
JOIN resident res ON scpo.userIDHouseOwner = res.numCC
JOIN minimum_sensor_count_per_owner mscpo ON scpo.sensorCount = mscpo.minSensorCount;
```

Functional explanation: This query identifies the names of homeowners who not only own the top two highest energy-consuming devices over the past two years but also have the fewest sensors among such top device owners. It first determines which devices have consumed the most energy, then finds the owners of these devices, and finally, among these owners, it identifies those with the

least number of sensors, highlighting a specific group of homeowners focused on high energy use and minimal sensor deployment.

Result:

NAME
Pedro Santos

Query h)

List the total nominal power connected for all houses/apartments with more than two floors. Note: the total nominal power is equal to the sum of the nominal power of all devices in all rooms of the house.

```
SELECT h.houseID, sum(ch.value) AS totalNominalPower
FROM characteristics ch
JOIN device_type dt ON dt.devicetypeID = ch.devicetypeID
JOIN device d ON dt.devicetypeID = d.devicetypeID
JOIN room r ON d.roomID = r.roomID
JOIN house h ON r.houseID = h.houseID
WHERE h.floors > 2 AND ch.name = 'Nominal Power'
GROUP BY h.houseID;
```

Functional explanation: This query calculates the total nominal power of devices for houses with more than two floors. It aggregates the nominal power values (identified by the characteristic 'Nominal Power') of all devices within such houses, presenting a sum for each house. Essentially, it provides a snapshot of the power capacity linked to devices in larger, multi-floor homes.

Result:

HOUSEID	TOTALNOMINALPOWER
2	1000
5	1200

Query i)

List which house reached the maximum temperature between 12/03/2023 and 20/3/2023. Also, include the value of the maximum temperature and the day of the week when this value was reached.

```
WITH max_temperature_per_house AS (
    SELECT h.houseID,
           max(r.reading) AS maxTemperature
    FROM house h
    JOIN room rm ON h.houseID = rm.houseID
    JOIN device d ON rm.roomID = d.roomID
    JOIN sensor s ON d.deviceID = s.deviceID
    JOIN readings r ON s.sensorID = r.sensorID
    JOIN sensor_type st ON s.sensorTypeID = st.sensorTypeID
    WHERE st.name = 'Temperature Sensor'
    GROUP BY h.houseID)
SELECT mth.houseID, mth.maxTemperature, TO_CHAR(r.time, 'Day') AS
dayOfWeek
FROM max_temperature_per_house mth
JOIN readings r ON mth.maxTemperature = r.reading
JOIN sensor s ON r.sensorID = s.sensorID
JOIN device d ON s.deviceID = d.deviceID
JOIN room rm ON d.roomID = rm.roomID
WHERE rm.houseID = mth.houseID AND r.time BETWEEN to_date('12-03-2023',
'DD-MM-YYYY') AND to_date('20-03-2023', 'DD-MM-YYYY')
ORDER BY mth.maxTemperature DESC;
```

Functional explanation: This query retrieves the maximum temperature readings along with their corresponding house IDs and the day of the week during a specified date range. In summary, the query aims to provide information about the highest temperatures recorded in each house with temperature sensors during a specific period, along with the corresponding day of the week.

Result:

HOUSEID	MAXTEMPERATURE	DAYOFWEEK
1	35	Sunday

Query j)

List the moments when the temperature in a room (or rooms) dropped below the comfort level. Consider only the moments obtained in the period between April and August of the current year.

Note: We have considered a comfort level of 20 degrees Celsius.

```
SELECT time
FROM readings
WHERE reading < 20 AND sensorID IN (
    SELECT sensorID
    FROM sensor
    WHERE sensorTypeID = (
        SELECT sensorTypeID
        FROM sensor_type
        WHERE name = 'Temperature Sensor'))
AND extract(month from time) BETWEEN 4 AND 8
AND extract(year from time) = 2023;
```

Functional explanation: This query is designed to find the moments when the temperature in a room dropped below the comfort level, which is considered to be 20 degrees Celsius. It focuses on readings from sensors classified as "Temperature Sensor" types during the summer period from April to August 2023, identifying instances of unusually low temperatures within this timeframe.

Result:

TIME
01-APR-23 10.30.00.000000 AM
11-JUN-23 12.00.00.000000 PM
27-JUL-23 01.30.00.000000 PM
05-AUG-23 02.30.00.000000 PM
01-APR-23 10.30.00.000000 AM
11-APR-23 11.00.00.000000 AM
12-MAY-23 11.30.00.000000 AM
11-JUN-23 12.00.00.000000 PM
27-JUL-23 01.30.00.000000 PM
05-AUG-23 02.30.00.000000 PM
15-AUG-23 03.00.00.000000 PM

Query k)

List which sensor recorded the last motion detection. It's also important to list the house and the room where this detection occurred.

```
SELECT r.sensorID, r.time AS lastMotionDetection, h.houseID, ro.roomID
FROM readings r
JOIN sensor s ON r.sensorID = s.sensorID
JOIN device d ON s.deviceID = d.deviceID
JOIN room ro ON d.roomID = ro.roomID
JOIN house h ON ro.houseID = h.houseID
WHERE s.sensorTypeID = (
    SELECT sensorTypeID
    FROM sensor_type
    WHERE name = 'Motion Sensor'
) AND r.reading = 1
ORDER BY r.time DESC
FETCH FIRST 1 row ONLY;
```

Functional explanation: This query is crafted to pinpoint the latest occurrence when motion was detected within any house, focusing specifically on readings from 'Motion Sensor' types. It searches for instances where the sensor reading was '1', indicating motion, and retrieves details such as the sensor ID, the precise time of the last motion detection, and the associated house and room IDs. By ordering the results by the detection time in descending order and selecting only the topmost row, the query ensures that the most recent motion event is identified, providing key information about where and when this activity took place.

Result:

SENSORID	LASTMOTIONDETECTION	HOUSEID	ROOMID
3	23-JAN-24 05.00.00.000000 PM	2	4

Query l)

List, for each type of sensor in houses with gardens, the average temperature readings. For sensor types that cannot read temperatures, a column titled "OBS:" should appear with the message: "Does not have the capacity to read temperatures."

```
SELECT h.houseID, r.roomID, st.name AS sensorType, round(avg(
     CASE WHEN st.name = 'Temperature Sensor' THEN rd.reading else null
end), 2) AS avgTemperature,
     CASE WHEN st.name <> 'Temperature Sensor' THEN 'Does not have the
capacity to read temperatures.' ELSE NULL END AS obs
FROM house h
JOIN (
     SELECT houseID FROM room WHERE name = 'Garden') g ON h.houseID =
g.houseID
JOIN room r ON h.houseID = r.houseID
JOIN device d ON r.roomID = d.roomID
JOIN sensor s ON d.deviceID = s.deviceID
JOIN sensor_type st ON s.sensorTypeID = st.sensorTypeID
LEFT join readings rd ON s.sensorID = rd.sensorID
GROUP BY h.houseID, r.roomID, st.name;
```

Functional explanation: This query retrieves information about sensor readings in houses with gardens. It identifies houses with a 'Garden' room, then joins this data with rooms, devices, sensors, sensor types, and readings. For each room, it lists the sensor type and calculates the average temperature reading for 'Temperature Sensor' types. For other sensor types, it adds a note stating 'Does not have the capacity to read temperatures'.

Result:

HOUSEID	ROOMID	SENSORTYPE	AVGTEMPERATURE	OBS
1	2	Temperature Sensor	22.7	–
5	15	Gas Sensor	–	Does not have the capacity to read temperatures.
1	1	Temperature Sensor	26.25	–
5	14	Temperature Sensor	21	–
1	1	Gas Sensor	–	Does not have the capacity to read temperatures.
1	1	Light Sensor	–	Does not have the capacity to read temperatures.
5	18	Motion Sensor	–	Does not have the capacity to read temperatures.
5	20	AirSpeed Sensor	–	Does not have the capacity to read temperatures.
1	3	Humidity Sensor	–	Does not have the capacity to read temperatures.
5	15	Motion Sensor	–	Does not have the capacity to read temperatures.
1	3	Motion Sensor	–	Does not have the capacity to read temperatures.
1	3	Temperature Sensor	–	–
5	14	Motion Sensor	–	Does not have the capacity to read temperatures.
5	16	Motion Sensor	–	Does not have the capacity to read temperatures.

Query m)

List the total number of "Heaters" devices for houses/apartments that, during the month of February 2023, had an average temperature of 30 degrees Celsius and possess more than 3 motion sensors.

```
SELECT h.houseID, r.roomID, count(DISTINCT d.deviceID) AS totalHeaters
FROM house h
JOIN room r ON h.houseID = r.houseID
JOIN device d ON r.roomID = d.roomID
JOIN sensor s ON d.deviceID = s.deviceID
JOIN sensor_type st ON s.sensorTypeID = st.sensorTypeID
JOIN (
    SELECT h.houseID, avg(rd.reading) AS avgTemperature
    FROM house h
    JOIN room r ON h.houseID = r.houseID
    JOIN device d ON r.roomID = d.roomID
    JOIN sensor s ON d.deviceID = s.deviceID
    JOIN readings rd ON s.sensorID = rd.sensorID
     WHERE rd.time BETWEEN timestamp '2023-02-01 00:00:00' AND timestamp
'2023-02-28 23:59:59'
    GROUP BY h.houseID
) temp ON h.houseID = temp.houseID
JOIN (
    SELECT h.houseID, count(s.sensorID) AS motionSensorCount
    FROM house h
    JOIN room r ON h.houseID = r.houseID
    JOIN device d ON r.roomID = d.roomID
    JOIN sensor s ON d.deviceID = s.deviceID
    JOIN sensor_type st ON s.sensorTypeID = st.sensorTypeID
    WHERE st.name = 'Motion Sensor'
    GROUP BY h.houseID
) motion ON h.houseID = motion.houseID
WHERE temp.avgTemperature = 30 AND motion.motionSensorCount > 3 AND
d.deviceName = 'Heater'
GROUP BY h.houseID, r.roomID;
```

Functional explanation: This query is designed to count the total number of "Heater" devices in houses that, during February 2023, had an average temperature of 30 degrees Celsius and have more than 3 motion sensors. It first calculates the average temperature for each house during February 2023 and the count of motion sensors in each house. Then, it joins these results with the data about houses, rooms, and devices. The query filters for houses where the average temperature is 30 degrees, the count of motion sensors is more than 3, and the device name is 'Heater'. The result is grouped by house and room IDs, providing the total count of "Heater" devices under these conditions.

Result:

HOUSEID	ROOMID	TOTALHEATERS
5	14	1

Query n)

List, for all rooms in houses/apartments, the minimum temperature recorded by temperature sensors, the maximum value of brightness recorded by light sensors, and the total number of gas detections recorded by gas sensors. Rooms in the house that do not have these types of sensors should be excluded.

```
SELECT r.roomID, MIN(t.reading) AS min_temperature, MAX(l.reading) AS
max_brightness, COUNT(g.reading) AS total_gas_detections
FROM room r
INNER JOIN device d ON r.roomID = d.roomID
INNER JOIN sensor s ON d.deviceID = s.deviceID
LEFT JOIN readings t ON s.sensorID = t.sensorID
AND t.reading IS NOT NULL AND s.sensorTypeID = (
    SELECT sensorTypeID
    FROM Sensor_Type
    WHERE name = 'Temperature Sensor')
LEFT JOIN readings l ON s.sensorID = l.sensorID AND l.reading IS NOT
NULL AND s.sensorTypeID = (
    SELECT sensorTypeID
    FROM Sensor_Type
    WHERE name = 'Light Sensor')
LEFT JOIN readings g ON s.sensorID = g.sensorID AND g.reading IS NOT
NULL AND s.sensorTypeID = (
    SELECT sensorTypeID
    FROM Sensor_Type
    WHERE name = 'Gas Sensor')
GROUP BY r.roomID
HAVING COUNT(t.reading) > 0 AND COUNT(l.reading) > 0 AND
COUNT(g.reading) > 0;
```

Functional explanation: This query is designed to gather aggregated sensor readings for each room, focusing on temperature, brightness, and gas detection. The LEFT JOINs with readings are used to capture temperature, brightness, and gas readings separately based on their corresponding sensor types. The aggregated results include the minimum temperature, maximum brightness, and total gas detections for each room. The HAVING clause filters out rooms where at least one of the specified conditions is met, ensuring that only relevant data is included in the final result set.

Result:

ROOMID	MIN_TEMPERATURE	MAX_BRIGHTNESS	TOTAL_GAS_DETECTIONS
1	22	1500	4

Query o)

List the date, time, and type of devices turned on after 12:30 PM more than 2 years ago. Present the day of the week when the sensor was activated in a column named "Day of the Week." The results should be displayed in descending order of date and ascending order of time.

```
SELECT to_char(ap.startTime, 'YYYY-MM-DD') AS "date",
to_char(ap.startTime, 'HH24:MI:SS') AS "time", dt.name AS "deviceType",
to_char(ap.startTime, 'Day') AS "dayOfWeek"
FROM active_periods ap
JOIN device d ON ap.deviceID = d.deviceID
JOIN device_type dt ON d.deviceTypeID = dt.deviceTypeID
WHERE to_char(ap.startTime, 'HH24:MI:SS') > '12:30:00' AND ap.startTime
< sysdate - interval '2' YEAR
ORDER BY to_char(ap.startTime, 'YYYY-MM-DD') DESC, to_char(ap.startTime,
'HH24:MI:SS') ASC;
```

Functional explanation: This query retrieves information about active periods for devices, presenting the date, time, device type, and day of the week. The WHERE clause filters the results to include only active periods that started after 12:30:00, and the active periods that occurred more than two years ago from the current date. The select clause utilizes the TO_CHAR function to format the start time into date, time, device type, and day of the week. The ORDER BY clause then arranges the results first by descending date and then by ascending time, creating a well-organized output.

Result:

date	time	deviceType	dayOfWeek
2019-04-10	18:34:00	Smart Plug	Wednesday
2019-02-15	12:41:00	Smart Plug	Friday

Query p)

List the location of the house that has equipment operating on weekends.

```
SELECT distinct h.gpsLocationID, h.street, h.doorNumber,
h.apartmentNumber, h.zipCode, h.city
FROM house h
JOIN room r ON h.houseID = r.houseID
JOIN device d ON r.roomID = d.roomID
JOIN active_periods ap ON d.deviceID = ap.deviceID
WHERE (to_char(ap.startTime, 'D') IN ('1', '7') OR to_char(ap.endTime,
'D') in ('1', '7'));
```

Functional explanation: This query retrieves distinct information about houses based on their GPS location and address details. The WHERE clause filters the results to include only houses where the active periods associated with their devices occur on either Sunday ('1') or Saturday ('7'). The DISTINCT keyword ensures that only unique house records meeting these criteria are included in the final result set.

Result:

GPSLOCATIONID	STREET	DOORNUMBER	APARTMENTNUMBER	ZIPCODE	CITY
1	Main St	123	A	4700-001	Braga
2	Downtown St	456	B	4000-002	Porto
3	Hill Ave	789	C	3000-987	Coimbra
4	Mountain Rd	101	D	2982-009	Lisbon
6	Main Street	1	A	1000-100	Madeira
5	Street	654	AA	4890-076	Braga

Query q)

List the property location and the reference of devices that are scheduled to operate on Mondays, Thursdays, and Fridays between 10:00 AM and 3:45 PM. The result should be sorted in descending order of device reference.

```
SELECT DISTINCT h.street, h.doorNumber, h.apartmentNumber, h.zipCode,
h.city, d.deviceID
FROM device d
JOIN room r ON d.roomID = r.roomID
JOIN house h ON r.houseID = h.houseID
WHERE EXISTS (
    SELECT 1
    FROM active_periods ap
    WHERE ap.deviceID = d.deviceID
    AND to_char(ap.startTime, 'DY') = 'MON'
    AND (
        (to_char(ap.startTime, 'HH24:MI') >= '10:00' AND
        to_char(ap.startTime, 'HH24:MI') <= '15:45')
        OR (to_char(ap.endTime, 'HH24:MI') >= '10:00' AND
        to_char(ap.endTime, 'HH24:MI') <= '15:45')))
AND EXISTS (
    SELECT 1
    FROM active_periods ap
    WHERE ap.deviceID = d.deviceID
    AND to_char(ap.startTime, 'DY') = 'THU'
    AND (
        (to_char(ap.startTime, 'HH24:MI') >= '10:00' AND
        to_char(ap.startTime, 'HH24:MI') <= '15:45')
        OR (to_char(ap.endTime, 'HH24:MI') >= '10:00' AND
        to_char(ap.endTime, 'HH24:MI') <= '15:45')))
AND EXISTS (
    SELECT 1
    FROM active_periods ap
    WHERE ap.deviceID = d.deviceID
    AND to_char(ap.startTime, 'DY') = 'FRI'
    AND (
        (to_char(ap.startTime, 'HH24:MI') >= '10:00' AND
        to_char(ap.startTime, 'HH24:MI') <= '15:45')
        OR (to_char(ap.endTime, 'HH24:MI') >= '10:00' AND
        to_char(ap.endTime, 'HH24:MI') <= '15:45')));
```

Functional explanation: This query retrieves distinct information about houses and associated devices based on specific criteria related to their active periods. The WHERE clause includes three subqueries, each checking for the existence of active periods satisfying conditions on specific days ('SEG', 'QUI', 'SEX') and within a specified time range ('10:00' to '15:45'). The DISTINCT keyword

ensures that only unique combinations of house details and device identifiers meeting these criteria are included in the final result set. The query essentially identifies houses with devices that have active periods on Mondays, Thursdays, and Fridays within the specified time range.

Result:

STREET	DOORNUMBER	APARTMENTNUMBER	ZIPCODE	CITY	DEVICEID
Street	654	AA	4890-076	Braga	2
Street	654	AA	4890-076	Braga	1
Street	654	AA	4890-076	Braga	3

Query r)

List the electricity consumption/production for the month of January of this year for houses that have more than two devices with nominal power higher than all devices in houses that have an airspeed sensor.

```
SELECT h.houseID,
       SUM(ap.energyconsumedkwh) AS totalEnergyConsumed
FROM device d
JOIN device_type dt ON d.deviceTypeID = dt.deviceTypeID
JOIN CHARACTERISTICS c ON dt.deviceTypeID = c.deviceTypeID
AND c.name = 'Nominal Power'
JOIN room r ON d.roomID = r.roomID
JOIN house h ON r.houseID = h.houseID
JOIN active_periods ap ON d.deviceID = ap.deviceID
WHERE c.value >
    (SELECT MAX(c.value)
     FROM house h
     JOIN room r ON h.houseID = r.houseID
     JOIN device d ON r.roomID = d.roomID
     JOIN sensor s ON d.deviceID = s.deviceID
     JOIN sensor_type st ON s.sensorTypeID = st.sensorTypeID
     AND st.name = 'AirSpeed Sensor'
     JOIN device_type dt ON d.deviceTypeID = dt.deviceTypeID
     JOIN CHARACTERISTICS c ON dt.deviceTypeID = c.deviceTypeID
     AND c.name = 'Nominal Power')
  AND h.houseID IN
    (SELECT h.houseID
     FROM house h
     JOIN room r ON h.houseID = r.houseID
     JOIN device d ON r.roomID = d.roomID
     GROUP BY h.houseID
     HAVING count(DISTINCT d.deviceID) > 2)
  AND ((to_char(ap.starttime, 'YYYY') = '2024'
        AND to_char(ap.starttime, 'MM') = '01')
       OR (to_char(ap.endtime, 'YYYY') = '2024'
           AND to_char(ap.endtime, 'MM') = '01'))
GROUP BY h.houseID;
```

Functional explanation: This query retrieves the total energy consumption of the houses in which more than two devices have a nominal power greater than the maximum nominal power of all devices in houses equipped with an airspeed sensor. It links houses to devices, rooms, and active periods, filtering based on nominal power and airspeed sensor presence. The query groups the results by house ID and calculates the total energy consumed for each house for the month of January in the current year. Houses meeting the specified criteria are included, highlighting those with high-power devices compared to airspeed sensor-equipped houses.

Result:

HOUSEID	TOTALENERGYCONSUMED
3	19

Query s)

List, for each month, only the last 6 months preceding the current date, the motion sensor that detected the most motion in houses where electricity consumption/production was higher than the average of houses with a single floor.

```
WITH single_floor_houses AS
  (SELECT h.houseid,
          avg(ap.energyconsumedkwh) AS avgEnergy
   FROM house h
   JOIN room r ON h.houseid = r.houseid
   JOIN device d ON r.roomid = d.roomid
   JOIN active_periods ap ON d.deviceid = ap.deviceid
   WHERE h.floors = 1
   GROUP BY h.houseid),
     avg_energy AS
  (SELECT avg(avgenergy) AS avgEnergy
   FROM single_floor_houses),
     high_energy_houses AS
  (SELECT h.houseid
   FROM house h
   JOIN room r ON h.houseid = r.houseid
   JOIN device d ON r.roomid = d.roomid
   JOIN active_periods ap ON d.deviceid = ap.deviceid
   GROUP BY h.houseid
   HAVING sum(ap.energyconsumedkwh) >
     (SELECT avgenergy
      FROM avg_energy)),
     motion_data AS
  (SELECT h.houseid,
          s.sensorid,
          count(*) AS motionCount,
          extract(MONTH
                  FROM ap.starttime) AS MONTH,
          extract(YEAR
                  FROM ap.starttime) AS YEAR
   FROM high_energy_houses h
   JOIN room r ON h.houseid = r.houseid
   JOIN device d ON r.roomid = d.roomid
   JOIN sensor s ON d.deviceid = s.deviceid
   JOIN active_periods ap ON s.sensorid = ap.deviceid
   WHERE s.sensortypeid = 3
     AND ap.starttime BETWEEN add_months(sysdate, -6) AND sysdate
   GROUP BY h.houseid,
            s.sensorid,
            extract(MONTH
```

```
                    FROM ap.starttime),
            extract(YEAR
                    FROM ap.starttime))
SELECT MONTH,
       YEAR,
       sensorid
FROM
  (SELECT MONTH,
          YEAR,
          sensorid,
          rank() OVER (PARTITION BY MONTH,
                                    YEAR
                       ORDER BY motioncount DESC) AS rank
   FROM motion_data)
WHERE rank = 1;
```

Functional explanation: This query identifies the motion sensors with the highest activity each month over the past six months in houses that have higher energy consumption than the average of single-floor houses. It first calculates the average energy consumption for single-floor houses, then finds houses that exceed this average. For these high-energy houses, the query tracks motion sensor activity, ultimately highlighting the most active sensor for each month within the specified period.

Result:

MONTH	YEAR	SENSORID
1	2024	31
8	2023	3
9	2023	31
10	2023	3
11	2023	31
12	2023	3

Conclusions

The group work on the project showcased a collaborative approach, starting with a foundational relational model and then systematically addressing various tasks, leading to iterative refinements. Each member was responsible for specific segments, contributing through targeted INSERT statements to fulfill distinct requirements. This cooperative dynamic facilitated a comprehensive exploration and implementation of the database functionalities.

In concluding, the project underscored the effectiveness of the relational model in managing a smart home system. The model's adaptability to evolving requirements, coupled with its capacity to support intricate queries and data analytics, highlighted the project's success in creating a robust and scalable database solution. The collaborative effort not only achieved the project's objectives but also provided valuable insights into the complexities of database design and management within a smart home context.

Annex A - Entity-Relationship Diagram