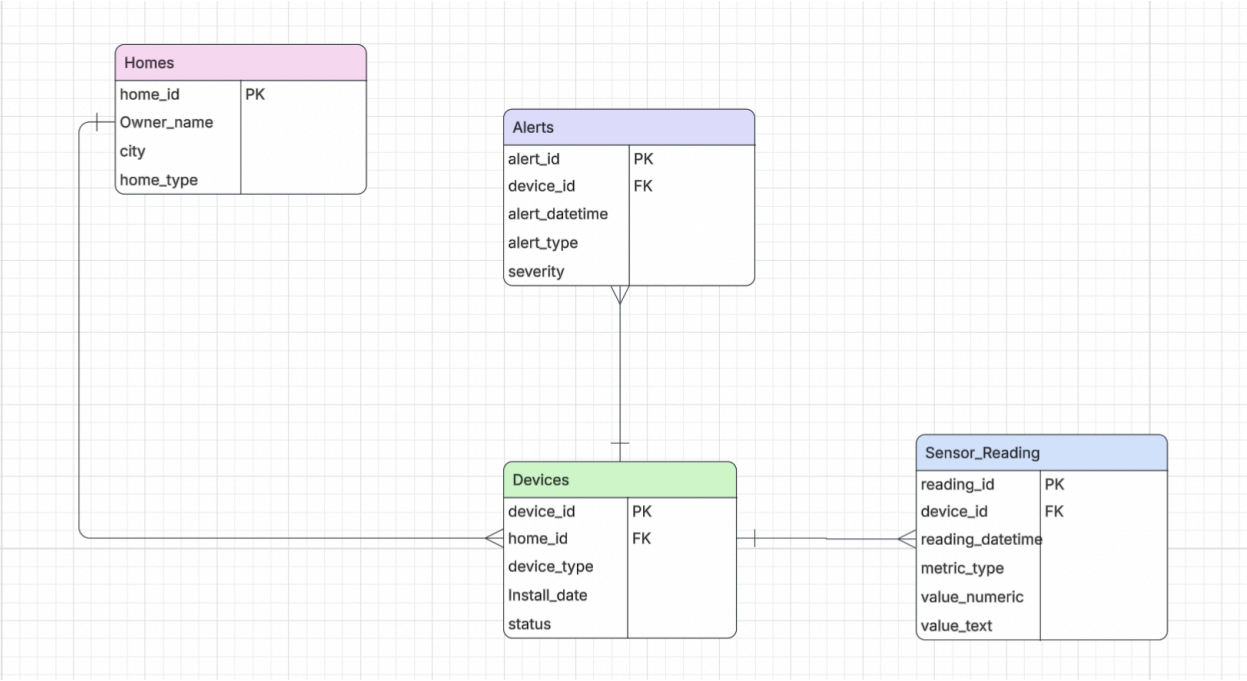
AD599 SMART HOME IoT

Team 3: Afreen Alam, Sydney Dao, Manjari Kannan, Veena Komatineni, Shreya Tarat

TASK 1

Reverse-Engineered ERD



Business Understanding Summary

The Smart Home IoT domain would cover a large array of devices being installed in residential homes, all with the aim of improving security, comfort, and energy efficiency. Each home may be equipped with many smart devices, including motion sensors, cameras, thermostats, and smart door locks, among others. There are various continuous sets of sensor readings that these devices transmit, for instance, motion detection, the state of the door, and temperature. The devices produce alerts upon events occurring that may be important, such as intrusions with each alert having a severity level. In total, this information can enable the business to monitor device health, understand how smart-home products are used by customers, and identify opportunities for improving service quality and product design.

The core "entities" in this business are homes, devices, sensor readings, and alerts. Every home represents one customer account and includes basic information about the owner, city and type of home; for example, apartment, condo, or single-family home. Devices are the hardware installed in each home, such as a thermostat or motion sensor, and their install date with their current status-whether active or offline. Sensor readings are the time-stamped data streams generated from these devices, which might include temperature values or motion flags. Alerts are higher-stakes events, such as intrusion detections, that capture when they happened and their severity level (e.g, medium, high).

From a business perspective, this environment supports the following key use cases:

- Home security monitoring, where motion, camera, and door-related readings feed into intrusion alerts. This enables the company to understand when and where security events take place and assess whether or not the system catches them reliably.
- Energy and comfort management are driven by thermostat actions and temperature readings. The business can analyze typical daily patterns of temperatures across homes and cities, and design better automated thermostat settings or new product features.
- Device lifecycle and reliability management, whereby device status-in other words, active versus offline-and accompanying patterns of alerts enable the firm to detect devices that might be failing or homes with a high proportion of offline hardware so as to proactively engage the customers.

Thus, this database represents how a smart-home company operates at a large scale: with onboarding homes to installing devices, receiving continuous sensor data, and escalating important events as alerts.

Key Business Processes (In Depth)

1. Customer onboarding and home setup
    - Creating a new home record when a customer signs up.
    - Information captured: owner details, city, and home type.
2. Device installation & lifecycle management
    - Installing devices into homes with an install date and device type.
    - Tracking device status, such as active versus offline, to monitor the health and connectivity of a device.
3. Sensor data collection
    - Record time-stamped sensor readings from devices, including motion, camera motion, door state, and temperature.
    - Using these readings to understand daily patterns and device performance.
4. Detection & incident monitoring
    - Generate intrusion alerts from the underlying sensor activity.
    - Logging what alerts occur, on which device they occurred, and the level of severity.
5. Operational monitoring & customer support
    - Identify homes with the most offline devices or high alert volumes.
    - Prioritize outreach, troubleshooting, or recommending upgrades or maintenance of devices.

Table	Column	Type	Definition
homes	home_id	INTEGER	Unique home identifier
	owner_name	TEXT	Name of homeowner
	city	TEXT	Home's city
	home_type	TEXT	Home category (apartment, condo, etc.)
devices	device_id	INTEGER	Unique device ID
	home_id	INTEGER	Home that owns the device

	device_type	TEXT	Type of device (thermostat, camera, etc.)
	install_date	DATE	Date device installed
	status	TEXT	Device state (active/offline)
sensor_readings	reading_id	INTEGER	Unique reading ID
	device_id	INTEGER	Device that generated reading
	reading_datetime	DATETIME	Timestamp of reading
	metric_type	TEXT	Measurement type (temperature, motion, etc.)
	value_numeric	REAL	Numeric reading (temp, motion=1/0)
	value_text	TEXT	Text reading (door open/closed)
alerts	alert_id	INTEGER	Unique alert ID
	device_id	INTEGER	Device that triggered alert
	alert_datetime	DATETIME	Timestamp of alert
	alert_type	TEXT	Alert type (intrusion)
	severity	TEXT	Alert severity (medium/high)

Proposed 10 Business Questions

1. In which type of home were the most number of high-severity alerts detected?
2. In 2025, which month had the most number of high severity alerts detected?
3. Which city has the most intrusions?
4. How many devices were installed per year?
5. The most number of high severity alerts are detected by which device type?
6. What are the top 3 cities with the fewest total alerts?
7. In New York, which month had the most alerts in 2025?
8. On average, homeowners install how many different devices in their homes?
9. What type of device is most commonly used in Boston homes?
10. What is the average number of alerts per home?

TASK 2

One Proposed Business Question solved using SQL queries

1. What are the top 3 cities with the fewest total alerts?

```
SELECT h.city, COUNT(a.alert_id) AS TotalAlerts
FROM homes h
LEFT JOIN devices d
   ON h.home_id = d.home_id
LEFT JOIN alerts a
   ON a.device_id = d.device_id
GROUP BY h.city
ORDER BY TotalAlerts ASC
LIMIT 3;
```

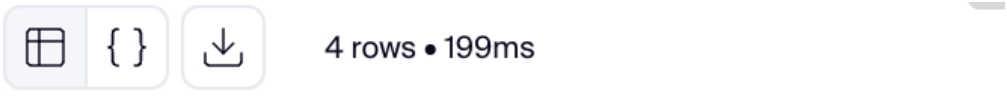
city	TotalAlerts
Boston	10
Chicago	10
Seattle	12

This query identifies the top three cities with the lowest number of alerts. Since all alerts represent intrusion events, cities with the lowest number of alerts reflect regions with the fewest security incidents, which could suggest safer environments or more effective protective measures. By using LEFT JOINs from homes to devices to alerts, the query ensures that cities with zero alerts are still captured. Aggregating using COUNT groups all alerts by city, and ordering the results in ascending order together with the LIMIT clause surfaces the 3 lowest-alert locations. This insight helps the business identify areas with strong security performance, understand geographic risk patterns, and prioritize security upgrades or customer outreach in higher-risk regions.

Analytical Questions

1. Which device types generate the highest number of alerts?

```
SELECT d.device_type, COUNT(a.alert_id) AS NumAlerts
FROM devices AS d
LEFT JOIN alerts AS a on a.device_id = d.device_id
GROUP BY d.device_type
ORDER BY NumAlerts DESC;
```



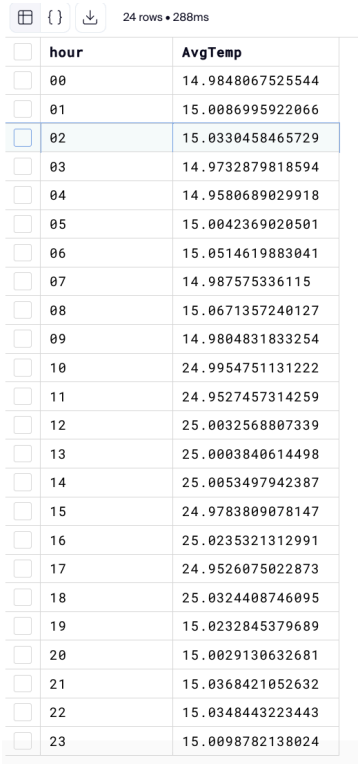
device_type	NumAlerts
motion_sensor	77
thermostat	0
door_lock	0
camera	0

This shows that a motion sensor is not only the device type that generates the highest number of alerts, but the only type that has generated alerts. This matters to the business because if no other device type is providing alerts, that may mean something is wrong with those device types if they are intended to provide alerts as well. We used a left join

instead of a regular join here to ensure all device types were visible, even though the other 3 didn't have any alerts. The left join connects the device table with the alerts table on device ID, so that we can then group the number of alerts by device ID.

2. What does the average daily temperature profile look like hour by hour?

   SELECT strftime('%H', reading_datetime) AS hour, AVG(value_numeric) AS AvgTemp
   FROM sensor_readings
   WHERE metric_type = 'temperature'
   GROUP BY hour
   ORDER BY hour;



24 rows • 288ms

hour	AvgTemp
00	14.9848067525544
01	15.0086995922066
02	15.0330458465729
03	14.9732879818594
04	14.9580689029918
05	15.0042369020501
06	15.0514619883041
07	14.987575336115
08	15.0671357240127
09	14.9804831833254
10	24.9954751131222
11	24.9527457314259
12	25.0032568807339
13	25.0003840614498
14	25.0053497942387
15	24.9783809078147
16	25.0235321312991
17	24.9526075022873
18	25.0324408746095
19	15.0232845379689
20	15.0029130632681
21	15.0368421052632
22	15.0348443223443
23	15.0098782138024

This shows that the average daily temperature profile is higher during the day than at night. It is highest, especially around mid-day (12 pm-2 pm). This data matters to the business because the business may want to use this information in developing future thermostat products. Additionally, smart thermostats typically have features where they adjust their set temperature automatically and this information can be used to better calibrate those settings. The sensor readings included other sensors, not just temperature readings, which is why we had to filter using WHERE for the metric to be temperature. To separate the readings hour by hour, we couldn't directly use reading_datetime and had to separate the hour using the strftime function. We were thenable to group by the newly created 'hour' to find the average temp for each hour as found using AVG.

3. Which homes have the highest proportion of offline or failing devices?

   SELECT h.home_id, COUNT(d.device_id) as NumDevices,

```
SUM(d.status = 'offline') AS OfflineDevices, SUM(d.status = 'offline') *1.0 /
COUNT(d.device_id) AS OfflineProportion
FROM homes as h
LEFT JOIN devices as d ON h.home_id = d.home_id
GROUP BY h.home_id
ORDER BY OfflineProportion DESC;
```

200 rows • 208ms

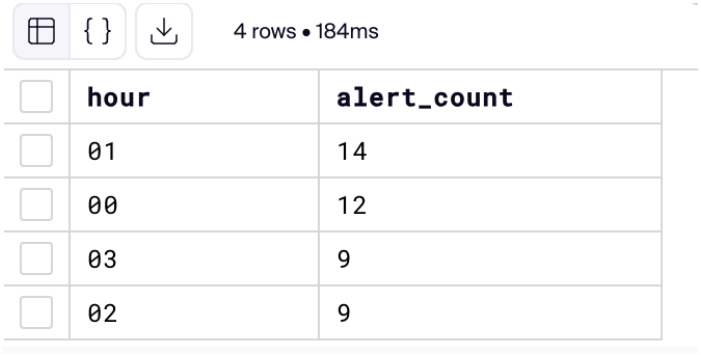
home_id	NumDevices	OfflineDevices	OfflineProportion
63	6	5	0.833333333333333
135	5	4	0.8
164	5	4	0.8
183	4	3	0.75
35	3	2	0.666666666666667
36	3	2	0.666666666666667
76	3	2	0.666666666666667
98	3	2	0.666666666666667
99	3	2	0.666666666666667
123	3	2	0.666666666666667
186	3	2	0.666666666666667
193	6	4	0.666666666666667
4	5	3	0.6
18	5	3	0.6
28	5	3	0.6
43	5	3	0.6
49	5	3	0.6
92	5	3	0.6
121	5	3	0.6
142	5	3	0.6
171	5	3	0.6
10	4	2	0.5
12	4	2	0.5
15	6	3	0.5
22	6	3	0.5

This shows that most homes have at least one device that is offline, as only 54 out of the 200 homes have 0 offline devices. This is arguably the most important of the analytical queries for the business because if a device is offline, it's not doing what it's supposed to do for the customer. This information is important because the business can identify homes with a high proportion of offline devices, such as home 63 which has 6 devices of which 5 are offline, and reach out to them to try to fix the issues for customer service or try to get them to buy new devices. The join used here is from the homes table to device table to identify which devices belong to which homes, and figure out the proportion from there. We used a left join to ensure all homes were displayed. We had to first find

the number of devices and number of offline devices using COUNT and SUM, before finding the bad device proportion. To find the homes with the highest offline device proportion, we grouped by home ID and then ordered by offline proportion in descending order.

4. When do nighttime intrusion alerts spike the most?

```
SELECT strftime('%H', alert_datetime) AS hour, COUNT(*) AS alert_count
FROM alerts
WHERE alert_type = 'intrusion' AND (
CAST(strftime('%H', alert_datetime) AS INTEGER) BETWEEN 20 AND 23
OR CAST(strftime('%H', alert_datetime) AS INTEGER)BETWEEN 0 AND 3)
GROUP BY hour
ORDER BY alert_count DESC;
```



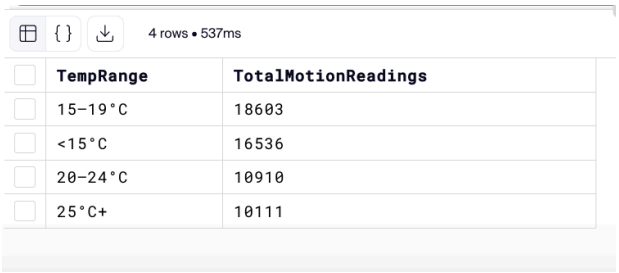
4 rows • 184ms

hour	alert_count
01	14
00	12
03	9
02	9

This shows that most nighttime intrusion alerts occurred around midnight and 1am, with some more occurring around 2am and 3am—implying that most intrusions occur during this time. This matters for the business as it should want to make sure the devices are definitely active or working during those times, so may want to avoid any patches or system upgrades etc… happening during these times. The question does not specify the definition of nighttime, but we chose to define it as from 8pm to 4am, which is why we filtered accordingly using WHERE. To check for intrusion alerts, we also used WHERE. Similar to question 2, we had to separate out the hours from alert_datetime before being able to use COUNT to find the alert count and then group by alert_count descending.

5. How do motion readings correlate with temperature changes?

```
WITH hourly AS (
  SELECT strftime('%Y-%m-%d %H', reading_datetime) AS hour,
  AVG(value_numeric) FILTER (WHERE metric_type = 'temperature') AS AvgTemp,
  COUNT(value_numeric) FILTER (WHERE metric_type = 'motion') AS NumMotion
  FROM sensor_readings
  GROUP BY hour),
TempRanges (Label, MinTemp, MaxTemp) AS (
  VALUES
  ('<15°C', NULL, 15),
  ('15–19°C', 15, 20),
  ('20–24°C', 20, 25),
  ('25°C+', 25, NULL)
)
SELECT tr.Label AS TempRange,SUM(h.NumMotion) AS TotalMotionReadings
FROM TempRanges as tr
JOIN hourly AS h
  ON (tr.MinTemp IS NULL OR h.AvgTemp >= tr.MinTemp)
 AND (tr.MaxTemp IS NULL OR h.AvgTemp < tr.MaxTemp)
GROUP BY TempRange
ORDER BY TotalMotionReadings DESC;
```



4 rows • 537ms

TempRange	TotalMotionReadings
15–19°C	18603
<15°C	16536
20–24°C	10910
25°C+	10111

This shows that the most motion readings occur when it's cold but not too cold (range of 15-19 degrees Celsius), with a lot also occurring when it's cold (<15 degrees Celsius). There are several thousand more motion readings when it's colder versus when it's warmer (20-24 and 25+). This is important to the business as it may want to investigate whether the sensors still work accurately in extreme temperatures. For instance, some types of motion sensors use temperature to help detect motion based on heat changes so

perhaps they're hypersensitive when it's colder. Similar to questions 2 and 4, we start by using the strftime function to get the hour. There are a lot of different readings, so in order to effectively compare motion readings and temperature, we created ranges for the temperatures as well as looked at average hourly temperature and the number of motion readings within that hour, instead of looking at every single reading separately. The SQL query here uses VALUES to establish the ranges we want to break the temperatures into for comparison. Ultimately, we were then able to GROUP BY the different temperature ranges we had established to compare the number of motion readings in each of those ranges.

Window Function Questions

1. What is the most recent reading recorded for each device?

```
SELECT
  device_id,
  reading_datetime,
  metric_type,
  value_numeric,
  value_text
FROM (
  SELECT
    *,
    ROW_NUMBER() OVER (
      PARTITION BY device_id
      ORDER BY reading_datetime DESC
    ) as rn
  FROM sensor_readings
) sub
WHERE rn = 1;
```

925 rows • 650ms

device_id	reading_datetime	metric_type	value_numeric	value_text
1	2025-11-16 15:12:14	motion	0	NULL
2	2025-11-16 12:31:14	motion	0	NULL
3	2025-11-16 21:17:19	camera_motion	0	NULL
4	2025-11-16 16:33:11	camera_motion	1	NULL
5	2025-11-16 17:47:57	temperature	22.9	NULL
6	2025-11-16 19:21:01	door_state	NULL	OPEN
7	2025-11-16 22:54:11	door_state	NULL	CLOSED
8	2025-11-16 09:57:38	temperature	15.2	NULL
9	2025-11-16 13:04:56	door_state	NULL	OPEN
10	2025-11-16 15:15:07	door_state	NULL	OPEN
11	2025-11-16 22:37:21	door_state	NULL	OPEN
12	2025-11-16 21:18:10	temperature	12.8	NULL
13	2025-11-16 22:19:14	camera_motion	0	NULL
14	2025-11-16 15:12:54	door_state	NULL	OPEN
15	2025-11-16 17:07:37	camera_motion	0	NULL
16	2025-11-16 18:35:55	temperature	24.6	NULL
17	2025-11-16 16:16:48	temperature	26.8	NULL
18	2025-11-16 16:04:52	motion	1	NULL

This result provides a real-time snapshot of the status of every device in the smart home ecosystem. For example, it immediately reveals the last recorded temperature, whether a specific door is currently open, or the previous humidity reading. It is a critical check on the current operational state of all sensors. The query primarily uses a Window Function (ROW_NUMBER()) without any explicit JOINs or standard GROUP BY aggregation. Window Logic would partition the data by device, rank the readings by time (most recent gets rank 1), and keep all original rows. This insight is vital for system monitoring and immediate decision-making. A business needs this information for customer service (quickly diagnosing "my device isn't working" issues) and alerting (trigger notifications only when the status changes).

2. What is the rolling 3-day average temperature for each home?

```
WITH
  DailyTemps AS (
    SELECT
      d.home_id,
      DATE(s.reading_datetime) AS reading_date,
      AVG(s.value_numeric) AS daily_avg_temp
```

```
    FROM
      sensor_readings s
      JOIN devices d ON s.device_id = d.device_id
    WHERE
      s.metric_type = 'temperature'
    GROUP BY
      d.home_id,
      DATE(s.reading_datetime)
  )
SELECT
  home_id,
  reading_date,
  daily_avg_temp,
  AVG(daily_avg_temp) OVER (
    PARTITION BY
      home_id
    ORDER BY
      reading_date ROWS BETWEEN 2 PRECEDING
      AND CURRENT ROW
  ) AS rolling_3day_avg
FROM
  DailyTemps
ORDER BY
  home_id,
  reading_date;
```

8160 rows • 598ms

home_id	reading_date	daily_avg_temp	rolling_3day_avg
1	2025-09-18	13.8	13.8
1	2025-09-19	19.125	16.4625
1	2025-09-20	18.7	17.2083333333333
1	2025-09-21	14.85	17.5583333333333
1	2025-09-22	19.4	17.65
1	2025-09-23	17.65	17.3
1	2025-09-24	18.85	18.6333333333333
1	2025-09-25	20.9	19.1333333333333
1	2025-09-26	19.975	19.9083333333333
1	2025-09-27	16.85	19.2416666666667
1	2025-09-28	16.675	17.8333333333333
1	2025-09-29	19.025	17.5166666666667
1	2025-09-30	17.425	17.7083333333333
1	2025-10-01	23.075	19.8416666666667
1	2025-10-02	18.775	19.7583333333333

The result is a smoothed temperature trend that removes the "noise" of hourly fluctuations. If a home's temperature spikes briefly from opening an oven door, the rolling average will barely move. It highlights sustainable heating/cooling patterns and long-term comfort levels over rapid, temporary changes. The query involves a Common Table Expression (CTE) for pre-processing and a Window Function for the final calculation. We use a standard GROUP BY home_id, reading_date to aggregate all raw readings into a single, clean daily average. The AVG, OVER window function calculates the mean across the three-day range. The ROWS BETWEEN 2 PRECEDING AND CURRENT ROW clause defines the rolling 3-day window frame. This insight is key for efficiency and predictive modeling. A business needs this information for forecasting and energy audits to better predict future temperature requirements and measure the true efficiency of systems.