**ASSIGNMENT -**1

**NAME**: D. SAIMANIKANTA

**REG NO**: 192372348

**COURSE**:DATA BASE MANAGEMENT

SYSTEM

**CODE**:CSA0593

**SCENERIO:**

**IoT Data Storage and Processing System Design a database for storing and processing data generated by IoT devices in real-time. Requirements: Model tables to handle devices, data readings, locations, and time-stamped data logs. Implement partitioning based on time intervals (e.g., daily) to efficiently manage large volumes of time-series data. Write queries to generate reports on device activity trends, detect anomalies, and visualize data by location. Create views or materialized views for summarizing data and improving performance of analytics queries.**

**IoT Data Storage and Processing System Design**

This design addresses the requirements of an **IoT data storage and processing system**. We will model tables to store data from IoT devices, handle real-time data readings, track device locations, and store time-stamped logs efficiently. We will also implement partitioning of time-series data, write SQL queries generate useful reports, and create materialized views to optimize performance

1. **Database Schema Design**

**Entities and Tables**

We will create four main tables to handle devices, data readings, locations, and time-stamped data logs.

**Table 1: Devices**

This table stores metadata about IoT devices.

|  |
| --- |
| CREATE TABLE Devices (  Device\_id= SERIAL PRIMARY KEY,  device\_name= VARCHAR(255),  device\_type= VARCHAR(255),  status VARCHAR(50), *-- e.g., active, inactive*  location\_id= INT REFERENCES Locations(location\_id), *-- Foreign key to Locations*  created\_at=TIMESTAMP DEFAULT CURRENT\_TIMESTAMP |

**Table 2: Locations**

This table stores location details for where the devices are deployed.

sql

Copy code

|  |
| --- |
| CREATE TABLE Locations (  location\_id SERIAL PRIMARY KEY,  location\_name VARCHAR(255),  latitude DOUBLE PRECISION,  longitude DOUBLE PRECISION,  description TEXT |

**Table 3: Data Readings**

This table stores individual readings from IoT devices. The readings will include data values such as temperature, humidity, etc.

Sql

|  |
| --- |
| CREATE TABLE DataReadings (  reading\_id SERIAL PRIMARY KEY,  device\_id INT REFERENCES Devices(device\_id), *-- Foreign key to Devices*  timestamp TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  data\_value DOUBLE PRECISION,  data\_type VARCHAR(255), *-- e.g., temperature, humidity, etc.*  sensor\_status VARCHAR(50) |

**SQL Queries for Reporting and Analytics**

**2.1. Device Activity Trend Report**

To track device activity trends (e.g., the number of readings per device per day), we can aggregate data by device\_id and timestamp.

|  |
| --- |
| SELECT device\_id,  DATE(timestamp) AS day,  COUNT(\*) AS readings\_count  FROM DataLogs  GROUP BY device\_id, day  ORDER BY day DESC; |

This query gives us the number of data readings recorded for each device, grouped by day.

**2.2. Anomaly Detection Query**

Anomalies can be defined as readings that exceed a specified threshold (e.g., temperature too high or low). Here’s an example query that detects temperature anomalies.

|  |
| --- |
| SELECT device\_id,  timestamp,  data\_value,  data\_type  FROM DataLogs  WHERE data\_type = 'temperature' AND (data\_value > 100 OR data\_value < -10); |

**Partitioning Strategy**

To efficiently manage time-series data, we use **partitioning by time** (daily partitions in this case). This improves performance by ensuring that only the relevant partitions are scanned when querying data for specific time ranges.

For example, querying data from the past 7 days can be optimized by only scanning partitions for the relevant days.

sql

Copy code

|  |
| --- |
| SELECT \*  FROM DataLogs  WHERE timestamp >= NOW() - INTERVAL '7 days'; |

**Conclusion**

**Key Design Aspects:**

* **Tables**: We have tables for Devices, Locations, DataReadings, and partitioned DataLogs.
* **Partitioning**: The DataLogs table is partitioned by time to manage large volumes of time-series data efficiently.
* **Queries**: Several SQL queries are designed to track device activity, detect anomalies, and visualize data by location.
* **Materialized Views**: Precomputed views improve performance for frequent analytics queries (e.g., daily averages, location-based aggregates).

This system design ensures that the IoT data is efficiently stored, processed, and analyzed.

**Conclusion**

**Key Design Aspects:**

* **Tables**: We have tables for Devices, Locations, DataReadings, and partitioned DataLogs.
* **Partitioning**: The DataLogs table is partitioned by time to manage large volumes of time-series data efficiently.
* **Queries**: Several SQL queries are designed to track device activity, detect anomalies, and visualize data by location.
* **Materialized Views**: Precomputed views improve performance for frequent analytics queries (e.g., daily averages, location-based aggregates).

This system design ensures that the IoT data is efficiently stored, processed, and analyzed in real time while maintaining scalability and performance for large volumes of time-series data.

in real time while maintaining scalability and performance for large volumes of time-series data.