Gisma University of Applied Sciences

“Smart Energy Management System: SQL and NoSQL Database Application “

- - - M605 ADVANCED DATABASES - - -

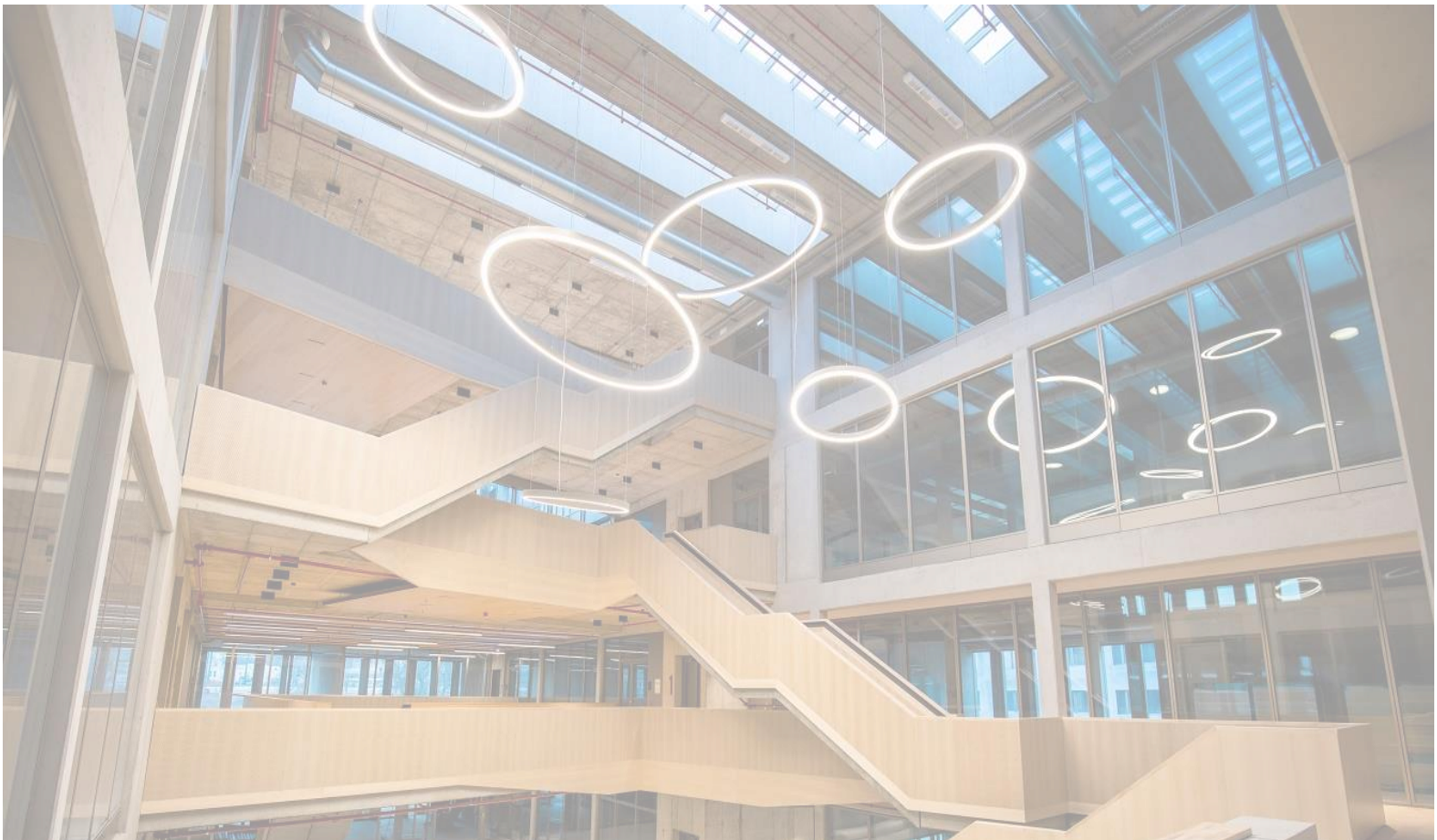
Batuhan ÖZTÜRK - GH1031500

Code and Project Report :

https://github.com/BatuhanOzturk0/SMART-_ENERGY_MANAGEMENT.git

Project video:

https://drive.google.com/file/d/1yhIv7nbAsknYd1ZKl85RIRVRv7b3bafu/view?usp=sharing



ABSTRACT

With today's technology , the energy sector relies heavily on 'big data' management. 'Variable consumption model' amd rising energy consumption have necessitated the processing and analysis of large amounts of data .Traditional SQL based databases, however might not be adequate for real time data processing.

The goal of this project is to create a hybrid energy management system using SQL and NoSQL.While MongoDB is preferred for processing real time consumption data and alerts,MySQL is used to manage user data and billing records.This hybrid design enables rapid processing of extensive data while ensuring secure storage of structured information.

The project's major objective is to gacilitate consumers in optimizing their energy usage behavious through accelerated data analsis.To do this, Aggregation Framework and Indexing techniques have been utilized to improve the performance of data queries.

This system aims to create a fundemantal paradigm for future applications in IoT and smart city infrastructure through an efficient approach to smart energy management and big data analysis.

INTRODUCTION

These days, energy consumption is rapidly increasing and sustinable energy management is beginning to gain traction.Businesses, governments and individual consumers are all looking for ways to use energy sources more efficiently.Efficiently is decreasing because current energy management systems are unable to effectively store and analyze large amounts of data.

NoSQL systems allow for flexible and real time data management,whereas traditional SQL systems offer data consistency and reliability.Thus, by combining MySQL and MongoDB in this project, a more powerful data processing infrastructure has been created.By creating a smart energy management systems and utilizing a SQL and NoSQL hybrid database model for big data analysis ,this project aims to provide a more effective solution.

What do we have to offer people ?

This system is intended for people who want to analyze and optimize their energy use.For household users. They can reduce their costs by monitoring their daily energy consumption.

For large companies and businesses: Improves energy efficiency through in depth data analysis.Energy providers can enhance theri energy management by examining consumer usage trends.

The SQL and NoSQL combination,especially for large scale data management,enables the systems to run searches and analyze data faster ,enabling both individual and corporate users to monitor and evaluate their energy use instantly.Additionally,insights into costumer usage patterns enable energy suppliers to refine their energy management strategies.

The hybridization of SQL and NoSQL facilitates swift data searches and analysis ,allowing both individual and corporate users to monitor and evaluate their energy consumption in real time .The primary objectibves of the system include:

- Real-time analysis of energy consumption data.
- Provision of efficient forecasts and reports to mitigate unnecessary energy usage.
- Encouragement of conscious enegry consumption among users.
- Delivery of cost effective energy management by optimizing big data analysis.

This project is more than just an energy tracking tool; it represents a concept that can be integrated with Internet Of Things devices and future smart city management projects .This model shows how data handling procedures may be accelerated and improved in terms of efficiency.

As a result, this initiative provides a more scalable and efficient approach to data handling than typical energy management systems.The combination of SQL and NoSQL technology offers a distinct perspective customized to the energy industry.

LITERATURE REVIEW

The importance of energy management has grown in recent years, owing to rising energy consumption and environmental imperatives.Large scale energy management systems aim to optimize energy use , using data analysis based decision making methodologies .However , standard relational database systems are limited in managing rapidly developing large scale data.

This literature study examines the use of hybrid database systems in energy management,including both successful and failed deployments.Furthermore,the research will evulate the structurak characteristics of similar applications in divers industries , as well as the competitive benefits these systems provide.

- Comperative Studies and Academic Insights

According to research on hybrid database systems , combining SQL and NoSQL to manage massive amounts of data greatly increases data processing performance.

*Chandra et aI.(2021) indvestigated the comparative flexibility of NoSQL in the field of big data management, as well as SQL's trustworthy data processing capabilities.The data indicated that a hybrid model reaches peak performance.

*Zhang and Li(2022) investigated appropriate storage methodologies for IoT based data within energy management,highlighting the advantages of NoSQL for real time data management.Nevertheless ,it was acknowledged that SQL continues to play an essential role in the safeguarding of relational data

*The smart energy management system developed by IBM(2020) optimizes energy consumption through the application of big data and artificial intelligence driven analytics.This framework demonstrated the ability to process large scale data efficiently by utilizing an itegrated approach that combines SQL and NoSQL.

-Diverse Strategies and Competition in Hybrid Data Management

The implementation of SQL and NoSQL hybrid models across several sectors illustrates the effects oh these systems on efficiency and performance.

Major corporations like Google and Amazon have adopted NoSQL systems for large scale data management while preserving the safe and consistent data architecture of SQL. Energy corporations like Tesla and Siemens use NoSQL for real time data analysis and SQL for financial transactions.Some small scale efforts have attempted to use NoSQL for rapid data processing benefits;however, they ultimately failed owing to data consistency challenges.The integration of SQL and NoSQL yields optimal outcomes by ensuring both data security and superior and superior performance.

-Reasons and Results of Effective and Not Successful Efforts

Many efforts have been made on large data systems in energy management.While some have failed other have victory.

*** Attributes of Successful Projects ***

- Optimization of Data Processing : The integration of SQL and NoSQL systems has resulted in enhanced efficiency.
- Real Time Data Analysis: Prompt analyses have been conducted on extensive data streams.
- High Scalability: The implementation of NoSQL technology has rendered the system more adaptable and extensible.

*** Common Challenges Associated with Ineffective Initiatives***

- Utilizing exclusively SQL and NoSQL has resulted in deficiencies in managing data on a large scale.
- The absence of real time data has rendered conventional solutions ineffective in addressing rapidly changing data environments.
- Issues related to data security and consistency have emerged ,particularly in systems that depend solely on NoSQL, leading to security.

Research shows that hybrid SQL and NoSQL database models have been used successfully in energy management systems.Real time analysis is made possible by NoSQL's ability to process large data sources quickly, while SQL offers dependable relational data management.By utilizing the advantages of both systems, the hybrid approach consistently produces the best outcomes.

METHODOLOGY

Managing huge energy consumption data has become essential given the increased need for sustainable energy management.Real time, high frequency energy usage logs can not be adequately handled in convential SQL based reltional databases.While lacking structural consistency of SQL, NoSQL databases provide scalability and flexibility.

This project utilizes a hybrid database model that integrates SQL (mysql workbench) and NoSQL (MongoDB) to provide reliable structured storage and facilitate real time data processing to address this challenge.The effective documentation,analysis, and visualization of energy consumption by the system highlight areas that require improvement.

- Data Model and Structure Selection

This project employsa hybrid database design , where SQL is used for structured transactional data and NoSQL handles dynamic and high frequency energy logs.

*** SQL Usage ***

- Users table : Stores user information (ID,name,email,password details)
- Devices Table: Manages smart energy devices linked to users.
- Billing Table : Recording monthly energy consumption bills.

*** NoSQL Usage **

- Energy Logs Collection : Watching real time energy consumption data from devices.
- Alerts Collection : Stores system generated notifications for unusual energy consumption patterns.
- Aggregation and Indexing : Optimizes query performance and enables complex data analysis.

- Database Integration Approach

The integration of SQL and NoSQL databases is a essential component of this system.The main techniques utilized for data synchronization are :

Data Flow between SQL and NoSQL

- SQL stores user and device details , NoSQL handles real time energy logs
- Updates in the SQL database are reflected in MongoDB collections(user add to new devices)

Query Optimization with Indexing and Aggregation

- SQL : Use SQL JOIN queries to assure relational data integrity.
- NoSQL: Aggregation searches ($group,$match,$sort) in NoSQL provide real time statistics.
- Indexing helps to improve performance (create index on timestamp,device_id,user_id)

*** Efficient Energy Consumption Analysis***

- In MongoDB using this queries ‘db.energy_logs.aggregate’ optimize large scale energy usage data analysiz.

-Technologies Used

Database Management

- MySQL Workbench: SQL database schema design and queries.
- MongoDb Compass : NoSQL data modeling ,CRUD operations, and aggreagation framework,

*** Development & Query Execution***

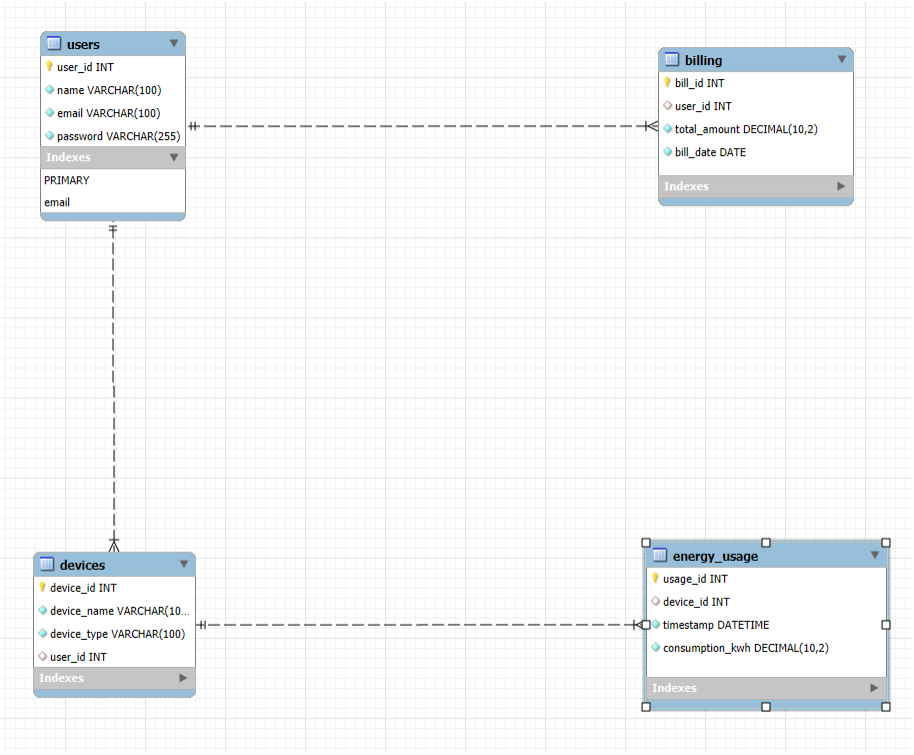
- VS Code : MongoDB playqround and SQL execution.

-Conclusion for Methodology

- Real time energy data processing is enabled
- Query execulation is optimized using indexing and aggregation.
- Large scale energy consumption data is securely stored and analyzed in this model type.

SYSTEM DESIGN

1-SQL DATABASE DESIGN (EER DIAGRAM)



The SQL database for this project selecting ‘a relational data model’ ensuring ‘structured and reliable data storage’.The database is designed these table:

- ***Users Table :*** Stores user information like user_id,name,email,password.

-Devices Table: Managing energy consuming devices. For example device_id, device_name, device_type, user_id.

-***Energy Usage Table*:** Recording device energy consumption all time (usage_id, device_id, timestamp, consumption_kwh)

-***Billing Table* :** Stores energy billing information for users (bill_id, user_id, total_amount, bill_date)

All table selecting some relational integrity with primary foreign keys , enabling efficient data management. We explains relational type :

*One to Many relationship exist between users and devices, so user can own multiple devices.

*One to Many realtionship exist between devices and energy usage, every device to have multiple energy consumption records.

‘ JOIN operations’ using for can retrive user-device relationship, energy consumption records , billing details efficiently.This relaitonal model ensures that data does not change during time consistent, safety and optimized for SQL queries.

2-NoSQL (MongoDB,Visual Studio Code) DOCUMENT MODEL

Replaces SQL’s relational and structured data model, MongoDB provides a flexible and sclable NoSQL document based model. This system has got two main collections:

- Energy_logs collection: Stores Real time energy consumption data , saved per devices.

- Alerts collection: Stores system generated notifications for strange energy consumption.

Every document in MongoDB follows a JSON structure ensuring fast and dynamic data retrieval.

We used the MongoDB program via Visual Studio Code.Below you will see the data written via Visual Studio Code and how it looks on MongoDB

- - - For energy logs Collection: Stores real time energy data using , you can see in the 2 examples below:

Visual Studio Code


```
[
  {
    "_id": {
      "$oid": "67d9dae00acf4687b4ebfc0f"
    },
    "device_id": 1,
    "timestamp": {
      "$date": "2025-02-17T09:36:00Z"
    },
    "consumption_kwh": 3.5
  },
  {
    "_id": {
      "$oid": "67d9dae00acf4687b4ebfc10"
    },
    "device_id": 1,
    "timestamp": {
      "$date": "2025-02-17T12:12:36Z"
    },
    "consumption_kwh": 2.1
  },
```

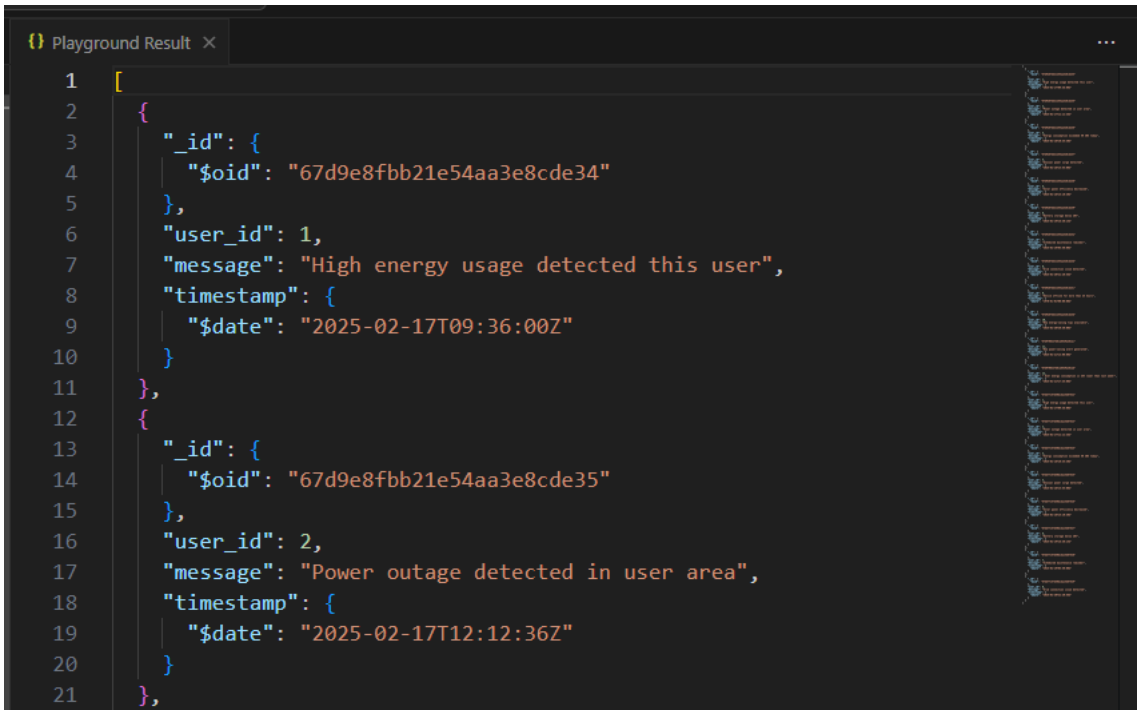
MongoDB

```
_id: ObjectId('67d9dae00acf4687b4ebfc0f')
device_id : 1
timestamp : 2025-02-17T09:36:00.000+00:00
consumption_kwh : 3.5
```

```
_id: ObjectId('67d9dae00acf4687b4ebfc10')
device_id : 1
timestamp : 2025-02-17T12:12:36.000+00:00
consumption_kwh : 2.1
```

- - - For alerts Collection : Record system generated warnings like below:

Visual Studio Code



MongoDB

```
_id: ObjectId('67d9e8fbb21e54aa3e8cde34')
user_id : 1
message : "High energy usage detected this user"
timestamp : 2025-02-17T09:36:00.000+00:00
```

```
_id: ObjectId('67d9e8fbb21e54aa3e8cde35')
user_id : 2
message : "Power outage detected in user area"
timestamp : 2025-02-17T12:12:36.000+00:00
```

If we want to use Real time data , should be select NoSQL . SQL generally using for structured data. Hybrid system provides high performance processing , fast query execution and optimized energy displaying.

3- How can entegration SQL and NoSQL

***** SQL Handles*** :**

-User and device management for ensuring structured data storage

-Billing and Transactions for saving financial data integrity.

-Relational database constraint for ensuring data consistency with foreign key relationship.

*****NoSQL Handles***:**

-Real time energy logs for fast adding and taking of energy usage data.

-Alerts and notifications for managing stystem generated.

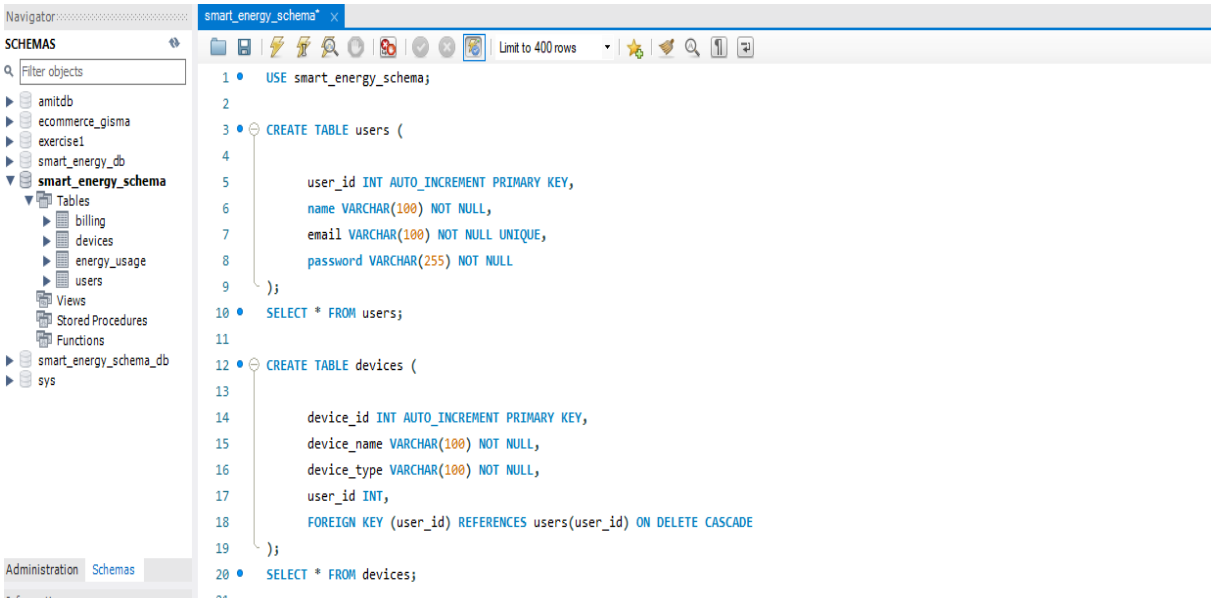
- Big data aggregation using $group,$match,$sort operations for analytics.

<u>Entegration following this way :</u>

Users and devices are stored in SQL and energy logs are stored in MongoDB . MongoDB's aggregation framework used for anlayzing energy consumption trends.This system taking user-device relationship from SQL And real time consumption data from MongoDB.

<u>IMPLEMENTATION</u>

<<<< 1- SQL Part Codding Explain. >>>>



CREATE DATABASE smart_energy_schema_db; this part creating area in MySQL workbench because we working in this part.

USE smart_energy_schema; Using this code for databases in the current session, all operations performed in this schema

CREATE TABLE users : This code created to store user information .

-user_id = Each users defined as a primary key that automatically increments.

-name = Users name , should be maximum 100 characters long

-email = Users email address need to unique.

-password = User's password , selected 255 characters for security reason

CREATE TABLE devices : Table Created to store devices for connected to users.

- device_id = For primary key specific to devices.

-device_name = Nmae of the device

- device_type = Type for device like smart sockets ,electricity meters.

- Foreign key = Using connection that user_id is linked to user_id in users table

- ON DELETE CASCADE= if any users deleted , all devices connected to user are deleted.

This model has got usin one-to-many relationship

SELECT * FROM users AND SELECT * FROM devices; These queries are using to list data in users and devices table .

```sql
CREATE TABLE energy_usage(

	usage_id INT AUTO_INCREMENT PRIMARY KEY,
	device_id INT,
	timestamp DATETIME NOT NULL,
	consumption_kwh DECIMAL(10,2) NOT NULL,
	FOREIGN KEY (device_id) REFERENCES devices(device_id) ON DELETE CASCADE

);
SELECT * FROM energy_usage;

CREATE TABLE billing (
	bill_id INT AUTO_INCREMENT PRIMARY KEY,
	user_id INT,
	total_amount DECIMAL(10,2) NOT NULL,
	bill_date DATE NOT NULL,
	FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE
);

SELECT * FROM billing;

```

CREATE TABLE energy_usage: Created to storage energy consumption data of devices.

-usage_id = Primary key for each energy usage record.

-device_id = ID of the device usage saving.

-timestmap = Date and time recorded.

-consumption_kwh= Amount of energy consumed.

- Foreign key = Using connection that user_id is linked to user_id in users table

- ON DELETE CASCADE= if any users deleted , all devices connected to user are deleted.

This structure allows us to all record how much energy using each device at certain time.

CREATE TABLE billing: Created to storage bill records and created based users energy consumption.

-bill_id = Primary key each all records:

-user_id= Displaying which users the bill belongs to.

-total_amount= Total energy bill amount that the user must pay for month

-bill_date= Date the bill was created.

- Foreign key = Using connection that user_id is linked to user_id in users table

- ON DELETE CASCADE= if any users deleted , all devices connected to user are deleted.

This structure allows us to trach monthly energy bill record for each users.

```sql
INSERT INTO users (name, email, password) VALUES
('Batuhan Ozturk','batuhan@example.com','14124124'),
('Munevver Ozturk','munevver@example.com','64646341'),
('Harun Ozturk','harun@example.com','12312477'),
('Vehbi Okumus','vehbi@exmaple.com','53678042'),
('Ismet Eren','ismet@example.com','19283640'),
('Semih Tarcan','semih@example.com','82659172'),
('Mehmet Ali','mehmet@example.com','46837482'),
('Samet Akbas','samet@example.com','15235376'),
('Ekin Kaya','ekin@example.com','37590127'),
('Omer Akin','omer@example.com','48362412');

UPDATE users SET email = 'batuhan@example.com' WHERE user_id = 1;
DELETE FROM users WHERE user_id =10;


INSERT INTO devices (device_name, device_type, user_id) VALUES
('Smart Meter 1','Electricity',1),
('Smart Thermostat','Heating',1),
('Solar Panel 1','Renewable',2),
('Battery Storage','Storage',3),
('EV Charger','Charging',4),
('Smart Fridge','Appliance',5),
('Wind Turbine 1','Renewable',6),
('Smart Bulb 1','Lighting',7),
('Water Heater','Heating',8),
('Security Camera','Surveillance',9);

UPDATE devices SET device_name ='Smart Meter 2' WHERE device_id = 1;
DELETE FROM devices WHERE device_id = 5;
```

```sql
INSERT INTO energy_usage (device_id, timestamp, consumption_kwh) VALUES
(1,'2025-02-17 09:36:00',3.5),
(1,'2025-02-17 12:12:36',2.1),
(2,'2025-02-18 10:35:30',1.8),
(3,'2025-02-18 14:45:00',4.2),
(4,'2025-02-18 15:10:30',5.3),
(5,'2025-02-19 16:30:15',2.9),
(6,'2025-02-19 04:40:00',3.7),
(7,'2025-02-20 22:20:56',1.3),
(8,'2025-02-20 08:00:50',2.8),
(9,'2025-02-20 20:30:00',4.9);

UPDATE energy_usage SET consumption_kwh = 4.0 WHERE usage_id = 2;
DELETE FROM energy_usage WHERE usage_id = 3;

INSERT INTO billing (user_id, total_amount, bill_date) VALUES
(1,50.75,'2025-03-01'),
(2,60.20,'2025-03-02'),
(3,45.10,'2025-03-03'),
(4,55.90,'2025-03-04'),
(5,70.30,'2025-03-05'),
(6,48.60,'2025-03-06'),
(7,65.80,'2025-03-07'),
(8,52.40,'2025-03-08'),
(9,58.90,'2025-03-09'),
(10,62.10,'2025-03-10');

UPDATE billing SET total_amount = 75.50 WHERE bill_id = 3;
DELETE FROM billing WHERE bill_id = 2;
```

INSERT INTO = This SQL query add 10 different users to user table .This comment same for all upper code(users,energy_usage,devices.billing)

UPDATE ____ SET ____ WHERE ___ = If we want to update any device,user, energy_usage, billing using 'Update' comment. We use 'SET' comment for new data and using 'WHERE' comment for last data adresses.

For example : UPDATE energy_usage SET consumption_kwh = 4.0 WHERE usage_id = 2;

İn this example we want to update energy_usage , and add to new data consumption_kwh = 4.0 and selecting adresses from usage_id=2.

```sql
/*This query is INNER JOIN query incudes:
-user information table
-device information table
-Energy consumption table
*/

SELECT
    users.user_id,
    users.name,
    devices.device_id,
    devices.device_name,
    energy_usage.timestamp,
    energy_usage.consumption_kwh

FROM users
INNER JOIN devices ON users.user_id = devices.device_id
INNER JOIN energy_usage ON devices.device_id = energy_usage.device_id
ORDER BY energy_usage.timestamp DESC;


-- List all users and the devices they own , if any

SELECT

    users.user_id,
    users.name,
    devices.device_id,
    devices.device_name

FROM users
LEFT JOIN devices ON users.user_id = devices.user_id
ORDER BY users.user_id;
```

INNER JOIN = Shows devices each user uses and how much energy those devices consume. Also sorts the energy consumption saving from most recent to oldest

FROM users = First , we get users from the user table ,

INNER JOIN devices ON users.user_id = devices.device_id = then add the devices owned by those users using the devices table.

INNER JOIN energy_usage ON devices.device_id = energy_usage.device_id = We add how much energy each device consumed at what time using the energy usage table.

ORDER BY energy_usage.timestamp DESC; Finally we sort the results by the most recent energy consumption data.

We can see that devices users have and can learn when and how much energy these devices consume. We can quickly access the latest consumption data by sorting the data from the most rencet to oldest.

LEFT JOIN=This query showing all users registered in the system

FROM users= We get all users in the user table

LEFT JOIN devices ON users.user_id = devices.user_id= we add the devices owned by each user from the devices table and if a user does not have a device , the device_id and name are NULL.

ORDER BY users.user_id= We sort the users by their ID number.

We can list all users registered in the system and their devices.If a user does not have a device ,we can still make it appear in the list .INNER JOIN is used to analyze users devices and their energy consumption.LEFT JOIN is used to see which users have devices in the system. These queries are critical to analyzing our smart energy management systems data and understanding the relationship between users and devices.

```
141     -- list all devices and the users they own ,if any
142
143 •   SELECT
144
145         devices.device_id,
146         devices.device_name,
147         users.user_id,
148         users.name
149
150     FROM devices
151     RIGHT JOIN users ON devices.user_id = users.user_id
152     ORDER BY devices.device_id;
153
```

RIGHT JOIN= This step list all devices and which user the belong to.

FROM devices= We get all the devices in the devices using the users table

RIGHT JOIN users ON devices.user_id = users.user_id=if a device dosent have an owner the user_id and name are NULL

ORDER BY devices.device_id;= We list the devices in the order of device_id.

```
155     -- Show all users and devices with UNION instead of FULL JOIN
156
157 •   SELECT
158
159         users.user_id,
160         users.name,
161         devices.device_id,
162         devices.device_name
163
164     FROM users
165     LEFT JOIN devices ON users.user_id = devices.user_id
166
167
168     UNION
169
170     SELECT
171         users.user_id,
172         users.name,
173         devices.device_id,
174         devices.device_name
175
176     FROM  users
177
178     RIGHT JOIN devices ON users.user_id = devices.user_id
179     ORDER BY user_id;
180
```

LEFT JOIN brings users and devices ,then RIGHT JOIN brings devices and users. UNION using for combines both queries.We using UNION because its automatically removes duplicate our records.SQL has not FULL JOIN comment so we use UNION comment. This query lists devices and users ,then allowing you to identify which users have which devices and which devices are unowned .

```sql
/*
will calculate total number of devices each user "COUNT" this comment
This part showing the users name and ID
Code explain highlight users with the most devices with "ORDER BY total_devices"

*/

SELECT
   users.user_id,
   users.name,
   COUNT(devices.device_id) AS total_devices
FROM users
LEFT JOIN devices ON users.user_id = devices.user_id
GROUP BY users.user_id, users.name
ORDER BY total_devices DESC;


/* In this part calculate total energy consumption each users devices "SUM" comment
if user has not any devices , total consumption is NULL
We use ORDER BY total_consumption for the  who consume most energy

*/

SELECT
    users.user_id,
    users.name,
    SUM(energy_usage.consumption_kwh) AS total_consumption

FROM users
LEFT JOIN devices ON users.user_id = devices.device_id
LEFT JOIN energy_usage ON devices.device_id = energy_usage.device_id
GROUP BY users.user_id, users.name
ORDER BY total_consumption DESC;

```

COUNT = This comment we use for calculates the total numberof devices each user has . This code lists the user upper to lower. We combined users and their devices after than use COUNT to count the devices each user has . We sort the results by total_devices DESC to highlight the users with the most devices.

SUM= We calcultes the total amount of energy consumed by each users devices.

LEFT JOIN devices ON users.user_id = devices.device_id= We combine user and their devices.

LEFT JOIN energy_usage ON devices.device_id = energy_usage.device_id= Then we add energy consumed by these devices.

ORDER BY total_consumption DESC; We sort the results by total_consumption DESC to highlight the users who consume the most energy.

```sql
    -- In this part calculate average energy consumption each users "AVERAGE"

     SELECT
         users.user_id,
         users.name,
         AVG(energy_usage.consumption_kwh) AS avg_consumption

    FROM users
    LEFT JOIN devices ON users.user_id = devices.user_id
    LEFT JOIN energy_usage ON devices.device_id = energy_usage.device_id
    GROUP BY  users.user_id, users.name
    ORDER BY avg_consumption DESC;
```

AVERAGE = This method we use calculates the average energy consumption for the devices owned by every user. First we merge the users and devices tables. Then we add the energy_usage table,which contains the nergy consumed by these devices. We use AVG function to get the average energy consumed by each users devices.

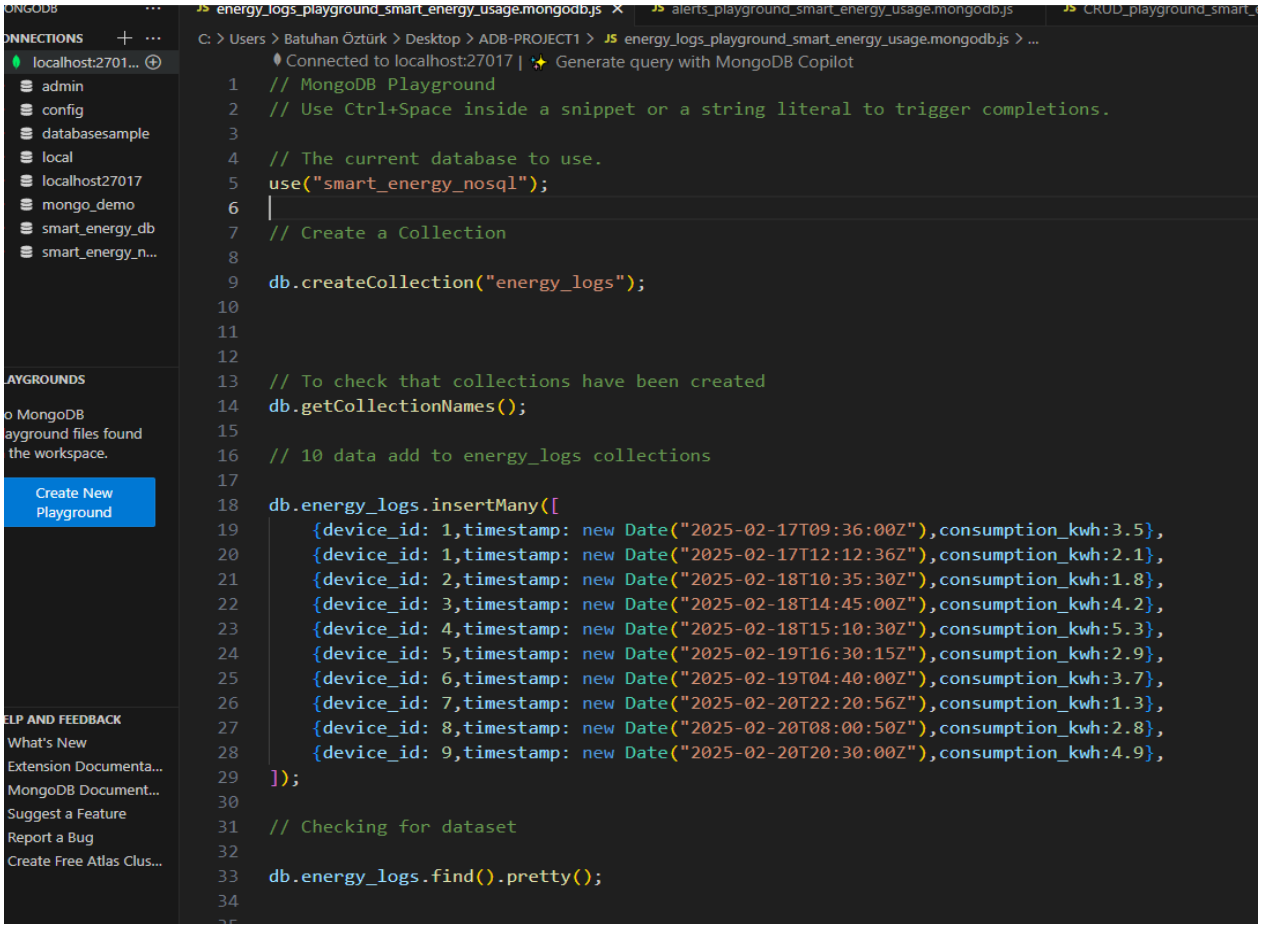
```sql
-- Only show users with average consumption higher than 3.0 kwh "HAVING"
 SELECT
     users.user_id,
     users.name,
     AVG(energy_usage.consumption_kwh) AS avg_consumption

FROM users
LEFT JOIN devices ON users.user_id = devices.user_id
LEFT JOIN energy_usage ON devices.device_id = energy_usage.device_id
GROUP BY  users.user_id, users.name
HAVING avg_consumption > 3.0
ORDER BY avg_consumption DESC;
```

HAVING = We use this function for calculates the average energy consumption of each users devices.We add to energy_usage table ,which contains the energy consumed by these devices. Then, AVG function to tget the average energy consumed by each users device . With HAVING avg_consumptiom >… we only return users who consume more than ….kwh.

```sql
-- Retrieving the highest consumption user with " SUBQUERIES "


SELECT  user_id ,name, total_consumption
FROM (

    SELECT
       users.user_id,
       users.name,
       SUM(energy_usage.consumption_kwh) AS total_consumption
    FROM users
    LEFT JOIN devices ON users.user_id = devices.user_id
    LEFT JOIN energy_usage ON devices.device_id = energy_usage.device_id
    GROUP BY users.user_id, users.name
) AS consumption_table
ORDER BY total_consumption DESC
LIMIT 1 ;
```

SUBQUERY = We calculate all users total energy conusmption, after than using with ORDER BY , brings the user who consumes the most energy to top.Finally , this code showing us only the single user who consumes the most with LIMIT 1. We use GROUP BY for groups the users.

2-NoSQL Part Codding Explain



use("smart_energy_nosql"); = We selected Databases (if we have not any database , this code will create)

db.createCollection("energy_logs"); = Creating new collection

db.getCollectionNames(); By listening the available collections , we check that the energy_logs collection was created successfully.

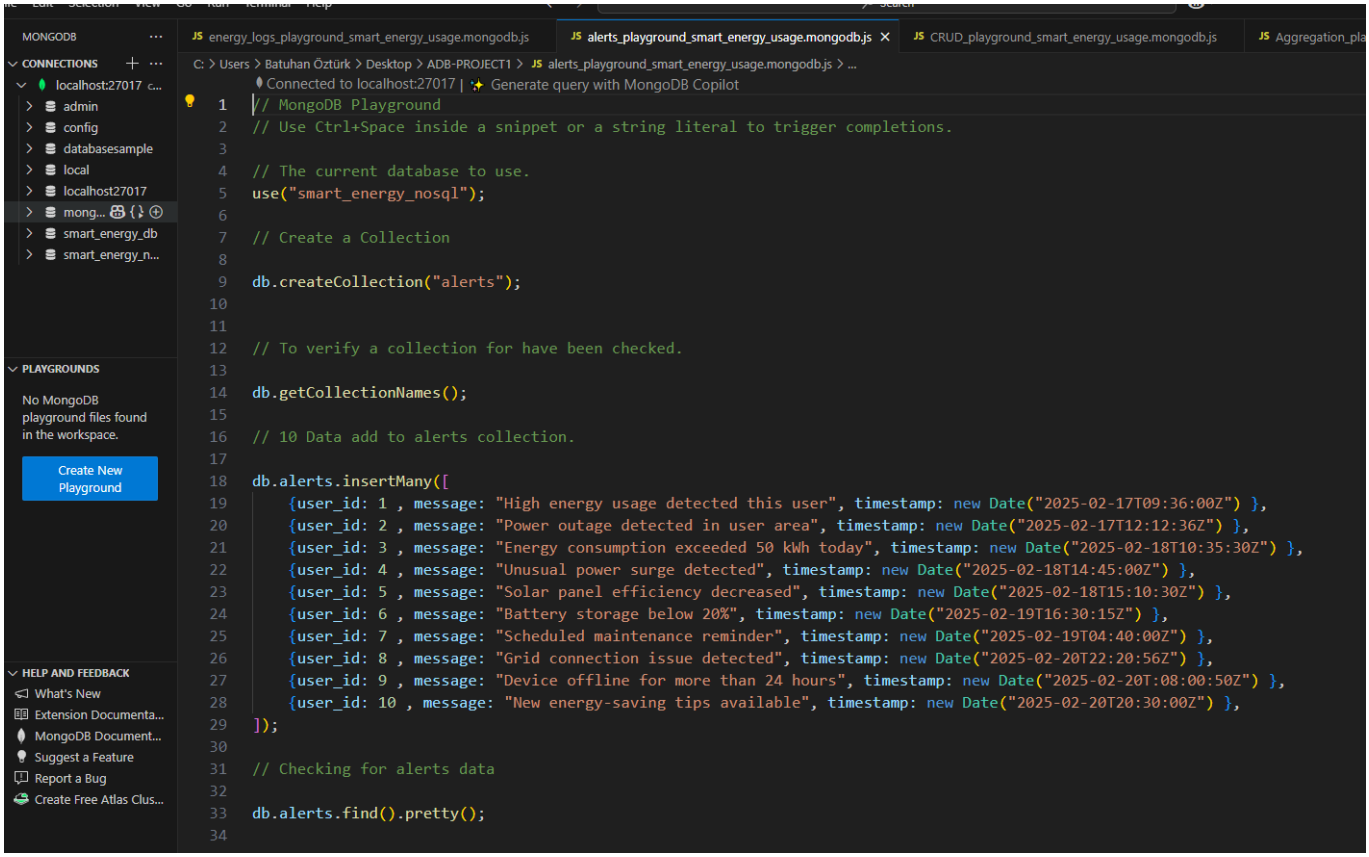
db.energy_logs.insertMany = Heree we added 10 data to energy_logs collection using this function. This document has got same contains:

- device_id = Showing which device is producing data

- timestmap = That code contains the timestamp of data saved.

- consumption_kwh = Stores the amount of energy consumed by the device at a given value .

db.energy_logs.find().pretty(); We use this command display all the documents in the energy_logs collection in available type.



use("smart_energy_nosql"); = We selected Databases (if we have not any database , this code will create)

db.createCollection("alerts"); = Creating new collection

db.getCollectionNames() = By listening the available collections , we check that the alerts collection was created successfully.

db.alerts.insertMany = Heree we added 10 data to alerts collection using this function. This document has got same contains:

- device_id = Showing which device is producing data

- timestmap = That code contains the timestamp of data saved.

- consumption_kwh = Stores the amount of energy consumed by the device at a given value .

db.alerts.find().pretty(); We use this command display all the documents in the alerts collection in available type.

CREATE = In this comment , we added to new data to energy logs and alerts table using MongoDB

- Device_id = Write to device number

- User_id = Which users it has to.

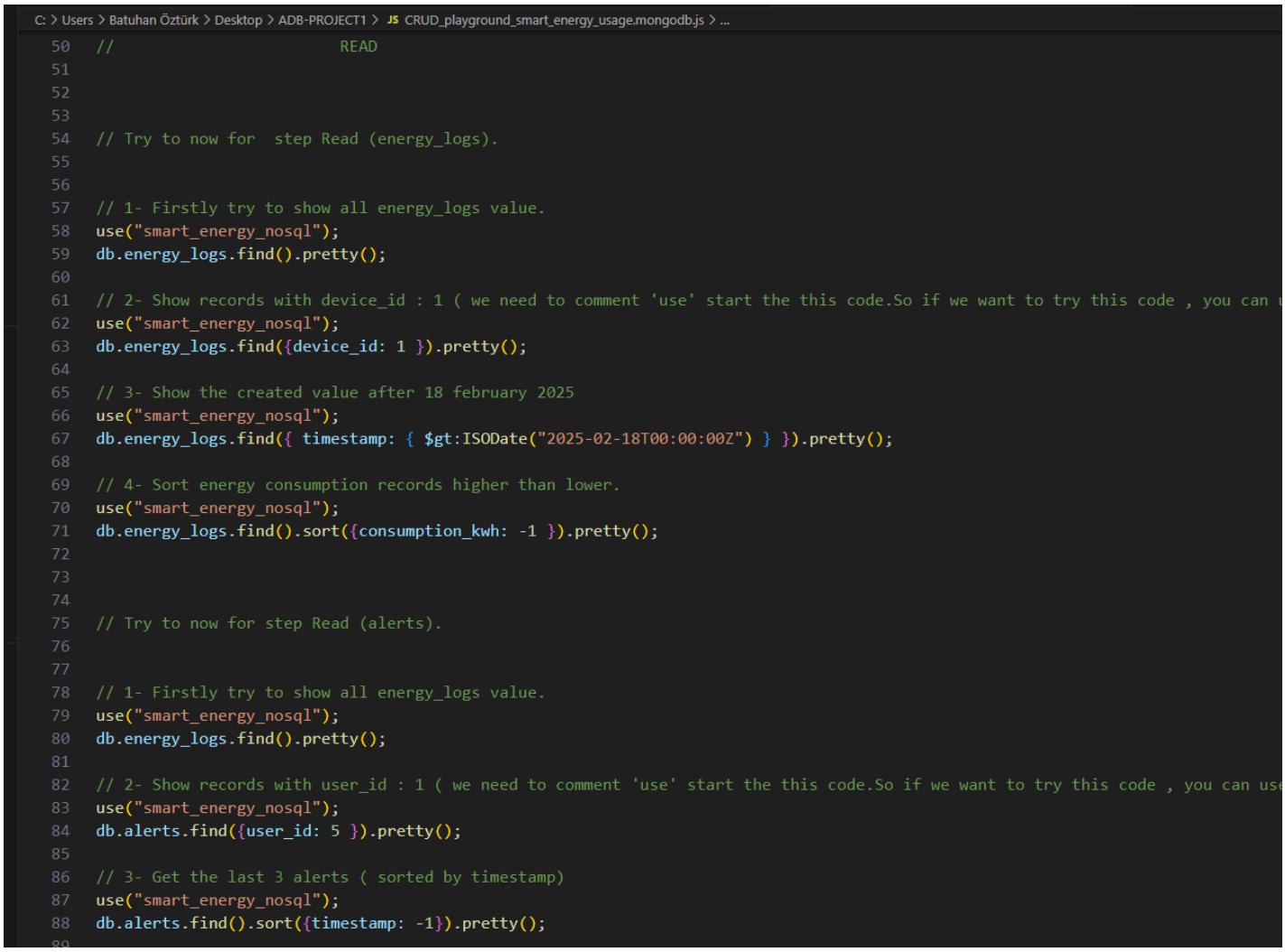
- message = Send to user for alert message

- timestamp = time the alert was created.

- consumption_kwh = During the time of energy the device consumes at a given moment.

db.energy_logs.insertMany & db.alerts.insertMany = We added new data these two table using insertMany function.

db.energy_logs.find().pretty() & db.alerts.find().pretty() = Display to all data energy_logs and alerts collection using this function.



```
C: > Users > Batuhan Öztürk > Desktop > ADB-PROJECT1 > JS CRUD_playground_smart_energy_usage.mongodb.js > ...
50  //                          READ
51
52
53
54  // Try to now for  step Read (energy_logs).
55
56
57  // 1- Firstly try to show all energy_logs value.
58  use("smart_energy_nosql");
59  db.energy_logs.find().pretty();
60
61  // 2- Show records with device_id : 1 ( we need to comment 'use' start the this code.So if we want to try this code , you can
62  use("smart_energy_nosql");
63  db.energy_logs.find({device_id: 1 }).pretty();
64
65  // 3- Show the created value after 18 february 2025
66  use("smart_energy_nosql");
67  db.energy_logs.find({ timestamp: { $gt:ISODate("2025-02-18T00:00:00Z") } }).pretty();
68
69  // 4- Sort energy consumption records higher than lower.
70  use("smart_energy_nosql");
71  db.energy_logs.find().sort({consumption_kwh: -1 }).pretty();
72
73
74
75  // Try to now for step Read (alerts).
76
77
78  // 1- Firstly try to show all energy_logs value.
79  use("smart_energy_nosql");
80  db.energy_logs.find().pretty();
81
82  // 2- Show records with user_id : 1 ( we need to comment 'use' start the this code.So if we want to try this code , you can use
83  use("smart_energy_nosql");
84  db.alerts.find({user_id: 5 }).pretty();
85
86  // 3- Get the last 3 alerts ( sorted by timestamp)
87  use("smart_energy_nosql");
88  db.alerts.find().sort({timestamp: -1}).pretty();
```

READ = In this step , we perform data query operations from the energy_logs and alerts collection.

--- For energy_logs ---

use("smart_energy_nosql") & db.energy_logs.find().pretty(); This query returns all documents in energy_logs collection in a readable format.

db.energy_logs.find({device_id: 1 }).pretty(); = This query returns only documents with device _id value 1

db.energy_logs.find({ timestamp: { $gt:ISODate("2025-02-18T00:00:00Z") } }).pretty(); = This query returns energy consumption records after 2025-02-18. $gt (greater than) operator is used to select records greater than a certain value.

db.energy_logs.find().sort({consumption_kwh: -1 }).pretty() = In this function sorts energy consumption records from highest to lowest.

--- For Alerts ---

use("smart_energy_nosql"); & db.energy_logs.find().pretty(); This function lists all documents in the alerts collection.

use("smart_energy_nosql"); & db.alerts.find({user_id: 5 }).pretty();query = This query return all alerts with used_id value 5.

use("smart_energy_nosql"); db.alerts.find().sort({timestamp: -1}).pretty(); = It sorts from largest to smallest(-1) according to the timestamp value . if this number equals = 1, we show smallest to largest according to .

These operations using for allow us to provide efficient data acquisition for energy monitoring warning systems.

```
92
93
94  //                      UPDATE
95
96
97  // For energy_logs
98
99  use("smart_energy_nosql");
100
101 // Update the consumption the record device_id: 10
102 db.energy_logs.updateOne(
103     {device_id: 10},
104     {$set:{consumption_kwh:9.0} }
105 );
106
107 // For alerts
108
109 use("smart_energy_nosql");
110
111 // Update the message the record user_id: 11
112 db.alerts.updateOne(
113     {user_id:11},
114     {$set: {message:"Updated : User power-saving alert has been modified"}}
115 );
116
117
```

UPDATE = This function , we perform update opreations on the energy logs and alerts collections using MongoDB. To change data in specific documents.

db.energy_logs.updateOne(

{device_id: 10},

{$set:{consumption_kwh:9.0} } = This commend updates the consumption_kwh value of the record with device_id value 10 to 9.0 . The $set commend is used to change only specified field.

db.alerts.updateOne(

{user_id:11},

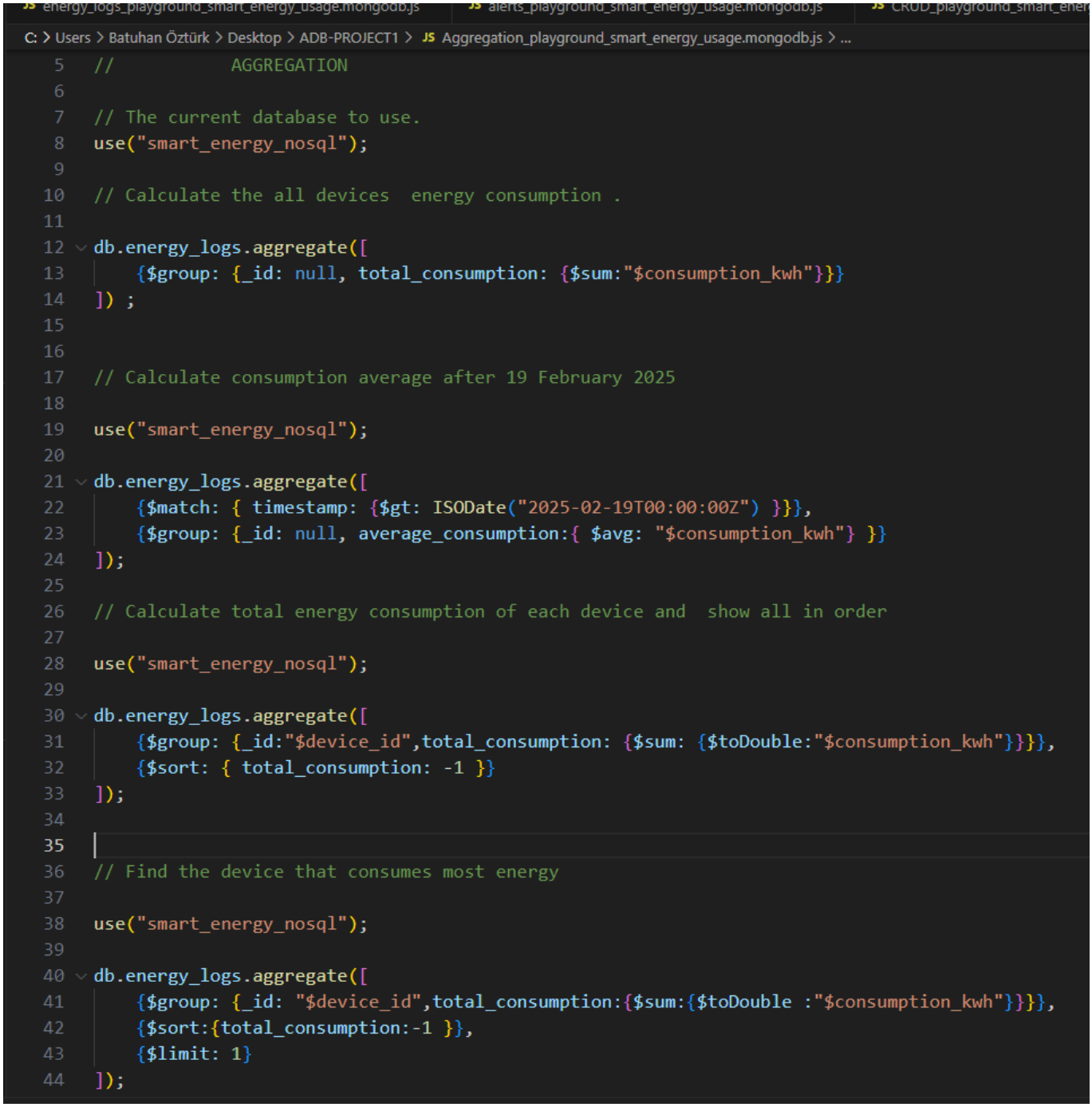
{$set: {message:"Updated : User power-saving alert has been modified"}} = This function updates the message field of the record with user_id value 11 with a new message.

```
//                    DELETE


// For energy_logs

use("smart_energy_nosql");

// Delete device_id: 10
db.energy_logs.deleteOne({ device_id: 10});


// For alerts

use("smart_energy_nosql");

// Delete user_id: 11
db.alerts.deleteOne({ user_id: 11 });
```

DELETE = In this step , we perform delete opreations on the energy_logs and alets collections using MongoDB.

db.energy_logs.deleteOne({ device_id: 10}) = This command removes the first document with device_id value 10 from energy logs collection.The deleteOne() function only deletes the first matching record.

db.alerts.deleteOne({ user_id: 11 }) = This command removes the first document with user_id 11 from the alerts colection.



AGGREGATION = In this performing advanced data operations to extract meaningful information from large data sets.

db.energy_logs.aggregate([{$group: {_id: null, total_consumption: {$sum:"$consumption_kwh"}}}

]) = This query calculates the total energy consumption of all devices. Using the $sum operator, all values in the consumption_kwh field are added together.

db.energy_logs.aggregate([{$match: { timestamp: {$gt: ISODate("2025-02-19T00:00:00Z") }}},

{$group: {_id: null, average_consumption:{ $avg: "$consumption_kwh"} }}]) = This query filters consumption data after 2025-02-19($match) , and calculates the average($avg) oh this data.

use("smart_energy_nosql"); db.energy_logs.aggregate([{$group: { _id:"$device_id",total_consumption: {$sum: {$toDouble:"$consumption_kwh"}}}},{$sort: { total_consumption: -1 }]); = This query calculates the total consumption($sum) for each device_id.Then sorts from largest to smallest ($sort = -1) according to the total_consumption value. The $toDouble operator converts the data type to double , allowing for more precise calculations.

**db.energy_logs.aggregate([{$group: { _id: "$device_id",total_consumption:{$sum:{$toDouble:"$consumption_kwh"}}}}, {$sort:{total_consumption:-1 }}, {$limit: 1 }); =** This query is used to find the device that consumes the most energy, First it calculates the total consumption by device ($sum).Then it sorts($sort : -1). It returns only the first results to get the device that consumes the most energy ($limit : 1).

```
47
48
49  //          INDEXING
50
51
52  use("smart_energy_nosql");
53
54  // Add index to 'timestamp' field for to speed up date-based
55  db.energy_logs.createIndex({timestamp: 1});
56
57  // Add index to 'device_id' field for to speed up device-based
58  db.energy_logs.createIndex({device_id:1});
59
60  // Add index to 'user_id' field for to speed up user-based
61  db.energy_logs.createIndex({user_id: 1});
```

INDEXING = Speeding up queries on large data sets.

db.energy_logs.createIndex({timestamp: 1}) = This query add an index to the timestamp field, which speeds up date-based queries. For example ,queries that retrieve records after a certain date will run faster.

db.energy_logs.createIndex({device_id:1}); This query adds an index to the device_id field , speeding up the search for data on specific devices.For example , queries that analyze the energy consumption of a device will be more efficient.

db.energy_logs.createIndex({user_id: 1}); In this comment adds an index to device_id field , speeding up the search for data on specific devices.

CHALLENGES AND SOLUTIONS

1 - Designing SQL and NoSQL Data Structures.

Challenge : Since the data structures between SQL and NoSQL are different, it was necessary to carefully desing the data model while creating tables and collections. It was diffucult to decide whether to embed or referencing the connecrions created with Foreign key in the relational database on the NoSQL side.

Solution : Users,devices ,billing energy_usage tables were created according to classical Normalization principles on the SQL side. On the NoSQL side , energy_logs and alerts collections were optimized with the embedding method.

2- Difference Between SQL and NoSQL in CRUD operations

Challenge : With data is processed with INSERT,UPDATE,DELETE command in SQL , commands such as insertMany(), updateOne(),deleteOne() are used in NoSQL .It took time for those who are accustomed to traditional SQL queries to adapt to the JSON based structure of MongoDB.

Solution: SQL and NoSQL CRUD operatiion were compared one on one and data insertion, update and deletion operations were performed step by step in both structures.Thanks to the use of Playground,testing operations in MongoDB became easier.

3- Data Consistency in NoSQL

Challenge : While mechanism such as Primary Key and Foreign Key ensure data consistency in SQL, there are no such requirements in NoSQL.It was necessary to ensure that the user's devices and energy consumption record were synchronized.

Solution: Data related to the user_id and device_id fields was created using the referencing method in the data model. During UPDATE operations in MongoDB the related records were updated.

4-Playground Usage and MongoDB Shell saving Problem .

Challenge : when using MongoDB Playground, it was diffucult to save and re-run th history of queries in MongoDB Shell, the.mongosh_repl history file was sometimes not found or gave errors.

Solution : Playground files were saved separatly (.mongodb.js). Code loss was prevented by backing up to GitHub.

RESULTS

SQL

- - - FOR USERS STEP - - -

```sql
CREATE TABLE users (

    user_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL
);
SELECT * FROM users;
```

```sql
INSERT INTO users (name, email, password) VALUES
('Batuhan Ozturk','batuhan@example.com','14124124'),
('Munevver Ozturk','munevver@example.com','64646341'),
('Harun Ozturk','harun@example.com','12312477'),
('Vehbi Okumus','vehbi@exmaple.com','53678042'),
('Ismet Eren','ismet@example.com','19283640'),
('Semih Tarcan','semih@example.com','82659172'),
('Mehmet Ali','mehmet@example.com','46837482'),
('Samet Akbas','samet@example.com','15235376'),
('Ekin Kaya','ekin@example.com','37590127'),
('Omer Akin','omer@example.com','48362412');
```

Result Grid | Filter Rows: | Edit: | Export

user_id	name	email	password
1	Batuhan Ozturk	batuhan@example.com	14124124
2	Munevver Ozturk	munevver@example.com	64646341
3	Harun Ozturk	harun@example.com	12312477
4	Vehbi Okumus	vehbi@exmaple.com	53678042
5	Ismet Eren	ismet@example.com	19283640
6	Semih Tarcan	semih@example.com	82659172
7	Mehmet Ali	mehmet@example.com	46837482
8	Samet Akbas	samet@example.com	15235376
9	Ekin Kaya	ekin@example.com	37590127
10	Omer Akin	omer@example.com	48362412
NULL	NULL	NULL	NULL

users 1

- We create the ‘users’ table with the ‘CREATE’ command and after adding data with the INSERT INTO command , we fill the users table with the ‘ SELECT ‘ command.

```sql
UPDATE users SET email = 'batuhan123@example.com' WHERE user_id = 1;
DELETE FROM users WHERE user_id =10;
```

1	Batuhan Ozturk	batuhan123@example.com	14124124
2	Munevver Ozturk	munevver@example.com	64646341
3	Harun Ozturk	harun@example.com	12312477
4	Vehbi Okumus	vehbi@exmaple.com	53678042
5	Ismet Eren	ismet@example.com	19283640
6	Semih Tarcan	semih@example.com	82659172
7	Mehmet Ali	mehmet@example.com	46837482
8	Samet Akbas	samet@example.com	15235376
9	Ekin Kaya	ekin@example.com	37590127
10	Omer Akin	omer@example.com	48362412
NULL	NULL	NULL	NULL

user_id	name	email	password
1	Batuhan Ozturk	batuhan123@example.com	14124124
2	Munevver Ozturk	munevver@example.com	64646341
3	Harun Ozturk	harun@example.com	12312477
4	Vehbi Okumus	vehbi@exmaple.com	53678042
5	Ismet Eren	ismet@example.com	19283640
6	Semih Tarcan	semih@example.com	82659172
7	Mehmet Ali	mehmet@example.com	46837482
8	Samet Akbas	samet@example.com	15235376
9	Ekin Kaya	ekin@example.com	37590127
NULL	NULL	NULL	NULL

- If we want to UPDATE the users table we add with he update command . If we want to delete we use the DELETE command.

- - - FOR DEVICES STEP - - -

```sql
CREATE TABLE devices (

    device_id INT AUTO_INCREMENT PRIMARY KEY,
    device_name VARCHAR(100) NOT NULL,
    device_type VARCHAR(100) NOT NULL,
    user_id INT,
    FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE
);
SELECT * FROM devices;

```

```sql
INSERT INTO devices (device_name, device_type, user_id) VALUES
('Smart Meter 1','Electricity',1),
('Smart Thermostat','Heating',1),
('Solar Panel 1','Renewable',2),
('Battery Storage','Storage',3),
('EV Charger','Charging',4),
('Smart Fridge','Appliance',5),
('Wind Turbine 1','Renewable',6),
('Smart Bulb 1','Lighting',7),
('Water Heater','Heating',8),
('Security Camera','Surveillance',9);
```

device_id	device_name	device_type	user_id
1	Smart Meter 1	Electricity	1
2	Smart Thermostat	Heating	1
3	Solar Panel 1	Renewable	2
4	Battery Storage	Storage	3
5	EV Charger	Charging	4
6	Smart Fridge	Appliance	5
7	Wind Turbine 1	Renewable	6
8	Smart Bulb 1	Lighting	7
9	Water Heater	Heating	8
10	Security Camera	Surveillance	9

```sql

UPDATE devices SET device_name ='Smart Meter 2' WHERE device_id = 1;
DELETE FROM devices WHERE device_id = 5;

```

Result Grid | Filter Rows: | Edit:

device_id	device_name	device_type	user_id
1	Smart Meter 2	Electricity	1
2	Smart Thermostat	Heating	1
3	Solar Panel 1	Renewable	2
4	Battery Storage	Storage	3
5	EV Charger	Charging	4
6	Smart Fridge	Appliance	5
7	Wind Turbine 1	Renewable	6
8	Smart Bulb 1	Lighting	7
9	Water Heater	Heating	8
10	Security Camera	Surveillance	9

**** WE APPLY THE SAME OPERATIONS WHEN CREATING A TABLE ,ADDING DATA ,UPDATING AND INSERTING DATA, AS IN THE USERS TABLE. ****

- - - FOR ENERGY_USAGE - - -

```sql
CREATE TABLE energy_usage(

    usage_id INT AUTO_INCREMENT PRIMARY KEY,
    device_id INT,
    timestamp DATETIME NOT NULL,
    consumption_kwh DECIMAL(10,2) NOT NULL,
    FOREIGN KEY (device_id) REFERENCES devices(device_id) ON DELETE CASCADE

);

SELECT * FROM energy_usage;
```

```sql
INSERT INTO energy_usage (device_id, timestamp, consumption_kwh) VALUES
(1,'2025-02-17 09:36:00',3.5),
(1,'2025-02-17 12:12:36',2.1),
(2,'2025-02-18 10:35:30',1.8),
(3,'2025-02-18 14:45:00',4.2),
(4,'2025-02-18 15:10:30',5.3),
(5,'2025-02-19 16:30:15',2.9),
(6,'2025-02-19 04:40:00',3.7),
(7,'2025-02-20 22:20:56',1.3),
(8,'2025-02-20 08:00:50',2.8),
(9,'2025-02-20 20:30:00',4.9);
```

Result Grid | Filter Rows: | Edit:

usage_id	device_id	timestamp	consumption_kwh
1	1	2025-02-17 09:36:00	3.50
2	1	2025-02-17 12:12:36	2.10
3	2	2025-02-18 10:35:30	1.80
4	3	2025-02-18 14:45:00	4.20
5	4	2025-02-18 15:10:30	5.30
6	5	2025-02-19 16:30:15	2.90
7	6	2025-02-19 04:40:00	3.70
8	7	2025-02-20 22:20:56	1.30
9	8	2025-02-20 08:00:50	2.80
10	9	2025-02-20 20:30:00	4.90

energy_usage 5

**** WE APPLY THE SAME OPERATIONS WHEN CREATING A TABLE ,ADDING DATA ,UPDATING AND INSERTING DATA, AS IN THE USERS TABLE. ****

```sql
UPDATE energy_usage SET consumption_kwh = 4.0 WHERE usage_id = 2;
DELETE FROM energy_usage WHERE usage_id = 3;
```

usage_id	device_id	timestamp	consumption_kwh
1	1	2025-02-17 09:36:00	3.50
2	1	2025-02-17 12:12:36	4.00
3	2	2025-02-18 10:35:30	1.80
4	3	2025-02-18 14:45:00	4.20
5	4	2025-02-18 15:10:30	5.30
6	5	2025-02-19 16:30:15	2.90
7	6	2025-02-19 04:40:00	3.70
8	7	2025-02-20 22:20:56	1.30
9	8	2025-02-20 08:00:50	2.80
10	9	2025-02-20 20:30:00	4.90

energy_usage 6

- - - FOR BILLS - - -

**** WE APPLY THE SAME OPERATIONS WHEN CREATING A TABLE ,ADDING DATA ,UPDATING AND INSERTING DATA, AS IN THE USERS TABLE. ****

```sql
CREATE TABLE billing (
    bill_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT,
    total_amount DECIMAL(10,2) NOT NULL,
    bill_date DATE NOT NULL,
    FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE
);

SELECT * FROM billing;
```

```sql
INSERT INTO billing (user_id, total_amount, bill_date) VALUES
(1,50.75,'2025-03-01'),
(2,60.20,'2025-03-02'),
(3,45.10,'2025-03-03'),
(4,55.90,'2025-03-04'),
(5,70.30,'2025-03-05'),
(6,48.60,'2025-03-06'),
(7,65.80,'2025-03-07'),
(8,52.40,'2025-03-08'),
(9,58.90,'2025-03-09'),
(10,62.10,'2025-03-10');
```

Result Grid | Filter Rows: | Edit:

bill_id	user_id	total_amount	bill_date
1	1	50.75	2025-03-01
2	2	60.20	2025-03-02
3	3	45.10	2025-03-03
4	4	55.90	2025-03-04
5	5	70.30	2025-03-05
6	6	48.60	2025-03-06
7	7	65.80	2025-03-07
8	8	52.40	2025-03-08
9	9	58.90	2025-03-09
NULL	NULL	NULL	NULL

billing 7 ×

```sql
UPDATE billing SET total_amount = 75.50 WHERE bill_id = 3;
DELETE FROM billing WHERE bill_id = 2;
```

Result Grid | Filter Rows: | Ed

bill_id	user_id	total_amount	bill_date
1	1	50.75	2025-03-01
2	2	60.20	2025-03-02
3	3	75.50	2025-03-03
4	4	55.90	2025-03-04
5	5	70.30	2025-03-05
6	6	48.60	2025-03-06
7	7	65.80	2025-03-07
8	8	52.40	2025-03-08
9	9	58.90	2025-03-09
NULL	NULL	NULL	NULL

billing 8 ×

```sql
/*This query is INNER JOIN query incudes:
-user information table
-device information table
-Energy consumption table
*/

SELECT
    users.user_id,
    users.name,
    devices.device_id,
    devices.device_name,
    energy_usage.timestamp,
    energy_usage.consumption_kwh

FROM users
INNER JOIN devices ON users.user_id = devices.device_id
INNER JOIN energy_usage ON devices.device_id = energy_usage.device_id
ORDER BY energy_usage.timestamp DESC;
```

Result Grid

user_id	name	device_id	device_name	timestamp	consumption_kwh
7	Mehmet Ali	7	Wind Turbine 1	2025-02-20 22:20:56	1.30
9	Ekin Kaya	9	Water Heater	2025-02-20 20:30:00	4.90
8	Samet Akbas	8	Smart Bulb 1	2025-02-20 08:00:50	2.80
5	Ismet Eren	5	EV Charger	2025-02-19 16:30:15	2.90
6	Semih Tarcan	6	Smart Fridge	2025-02-19 04:40:00	3.70
4	Vehbi Okumus	4	Battery Storage	2025-02-18 15:10:30	5.30
3	Harun Ozturk	3	Solar Panel 1	2025-02-18 14:45:00	4.20
2	Munevver Ozturk	2	Smart Thermostat	2025-02-18 10:35:30	1.80
1	Batuhan Ozturk	1	Smart Meter 1	2025-02-17 12:12:36	4.00
1	Batuhan Ozturk	1	Smart Meter 1	2025-02-17 09:36:00	3.50

Result 9

-In this SQL query , we combine data from three different tables using INNER JOIN : users table,devices table, energy_usage table. It analysis which devices users use and how much energy they consume .

```sql
-- List all users and the devices they own , if any

SELECT

    users.user_id,
    users.name,
    devices.device_id,
    devices.device_name

FROM users
LEFT JOIN devices ON users.user_id = devices.user_id
ORDER BY users.user_id;
```

Result Grid

user_id	name	device_id	device_name	timestamp	consumption_kwh
7	Mehmet Ali	7	Wind Turbine 1	2025-02-20 22:20:56	1.30
9	Ekin Kaya	9	Water Heater	2025-02-20 20:30:00	4.90
8	Samet Akbas	8	Smart Bulb 1	2025-02-20 08:00:50	2.80
5	Ismet Eren	5	EV Charger	2025-02-19 16:30:15	2.90
6	Semih Tarcan	6	Smart Fridge	2025-02-19 04:40:00	3.70
4	Vehbi Okumus	4	Battery Storage	2025-02-18 15:10:30	5.30
3	Harun Ozturk	3	Solar Panel 1	2025-02-18 14:45:00	4.20
2	Munevver Ozturk	2	Smart Thermostat	2025-02-18 10:35:30	1.80
1	Batuhan Ozturk	1	Smart Meter 1	2025-02-17 12:12:36	4.00
1	Batuhan Ozturk	1	Smart Meter 1	2025-02-17 09:36:00	3.50

Result 9

-In the result above : In this SQL query, we join the users and devices tables using LEFT JOIN. They query lists the devices that each user has

```sql
-- list all devices and the users they own ,if any

SELECT

    devices.device_id,
    devices.device_name,
    users.user_id,
    users.name

FROM devices
RIGHT JOIN users ON devices.user_id = users.user_id
ORDER BY devices.device_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content

device_id	device_name	user_id	name
1	Smart Meter 1	1	Batuhan Ozturk
2	Smart Thermostat	1	Batuhan Ozturk
3	Solar Panel 1	2	Munevver Ozturk
4	Battery Storage	3	Harun Ozturk
5	EV Charger	4	Vehbi Okumus
6	Smart Fridge	5	Ismet Eren
7	Wind Turbine 1	6	Semih Tarcan
8	Smart Bulb 1	7	Mehmet Ali
9	Water Heater	8	Samet Akbas
10	Security Camera	9	Ekin Kaya

Result 10

- In the result above : In this SQL query, we join the users and devices tables using RIGHT JOIN. They query lists the users that each devices has .

```sql
-- Show all users and devices with UNION instead of FULL JOIN

SELECT
    users.user_id,
    users.name,
    devices.device_id,
    devices.device_name

FROM users
LEFT JOIN devices ON users.user_id = devices.user_id

UNION

SELECT
    users.user_id,
    users.name,
    devices.device_id,
    devices.device_name

FROM  users
RIGHT JOIN devices ON users.user_id = devices.user_id
ORDER BY user_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell

user_id	name	device_id	device_name
1	Batuhan Ozturk	1	Smart Meter 1
1	Batuhan Ozturk	2	Smart Thermostat
1	Batuhan Ozturk	11	Smart Meter 1
1	Batuhan Ozturk	12	Smart Thermostat
1	Batuhan Ozturk	21	Smart Meter 1
1	Batuhan Ozturk	22	Smart Thermostat
2	Munevver Ozturk	3	Solar Panel 1
2	Munevver Ozturk	13	Solar Panel 1
2	Munevver Ozturk	23	Solar Panel 1
3	Harun Ozturk	4	Battery Storage
3	Harun Ozturk	14	Battery Storage
3	Harun Ozturk	24	Battery Storage
4	Vehbi Okumus	5	EV Charger
4	Vehbi Okumus	15	EV Charger
4	Vehbi Okumus	25	EV Charger
5	Ismet Eren	6	Smart Fridge
5	Ismet Eren	16	Smart Fridge
5	Ismet Eren	26	Smart Fridge
6	Semih Tarcan	7	Wind Turbine 1
6	Semih Tarcan	17	Wind Turbine 1
6	Semih Tarcan	27	Wind Turbine 1
7	Mehmet Ali	8	Smart Bulb 1
7	Mehmet Ali	18	Smart Bulb 1
7	Mehmet Ali	28	Smart Bulb 1
8	Samet Akbas	9	Water Heater
8	Samet Akbas	19	Water Heater
8	Samet Akbas	29	Water Heater
9	Ekin Kaya	10	Security Camera
9	Ekin Kaya	20	Security Camera
9	Ekin Kaya	30	Security Camera

Result 11

This query combines the users and devices tables using UNION to create a complete list.

```sql
/*
will calculate total number of devices each user "COUNT" this comment
This part showing the users name and ID
Code explain highlight users with the most devices with "ORDER BY total_devices'

*/

SELECT
   users.user_id,
   users.name,
   COUNT(devices.device_id) AS total_devices
FROM users
LEFT JOIN devices ON users.user_id = devices.user_id
GROUP BY users.user_id, users.name
ORDER BY total_devices DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

user_id	name	total_devices
1	Batuhan Ozturk	6
2	Munevver Ozturk	3
3	Harun Ozturk	3
4	Vehbi Okumus	3
5	Ismet Eren	3
6	Semih Tarcan	3
7	Mehmet Ali	3
8	Samet Akbas	3
9	Ekin Kaya	3

- We use this SQL query to calculate and list the total number of devices eache user has. The reason why there is so much total device right now is because we ran the select command 3 times and it collects the same data again . ** You can see LEFT JOIN section real total number **

- Batuhan ÖZTÜRK has 2 devices

- Mehmet Ali has 1 devices

```sql
/* In this part calculate total energy consumption each users devices "SUM" comment
if user has not any devices , total consumption is NULL
We use ORDER BY total_consumption for the  who consume most energy

*/

SELECT
    users.user_id,
    users.name,
    SUM(energy_usage.consumption_kwh) AS total_consumption

FROM users
LEFT JOIN devices ON users.user_id = devices.device_id
LEFT JOIN energy_usage ON devices.device_id = energy_usage.device_id
GROUP BY users.user_id, users.name
ORDER BY total_consumption DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

user_id	name	total_consumption
1	Batuhan Ozturk	7.50
4	Vehbi Okumus	5.30
9	Ekin Kaya	4.90
3	Harun Ozturk	4.20
6	Semih Tarcan	3.70
5	Ismet Eren	2.90
8	Samet Akbas	2.80
2	Munevver Ozturk	1.80
7	Mehmet Ali	1.30

Used to identify users who consume the most energy. The most energy used is Batuhan Öztürk 7.50 kwh

```sql
-- In this part calculate average energy consumption each users "AVERAGE"

 SELECT
     users.user_id,
     users.name,
     AVG(energy_usage.consumption_kwh) AS avg_consumption

FROM users
LEFT JOIN devices ON users.user_id = devices.user_id
LEFT JOIN energy_usage ON devices.device_id = energy_usage.device_id
GROUP BY  users.user_id, users.name
ORDER BY avg_consumption DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

user_id	name	avg_consumption
3	Harun Ozturk	5.300000
8	Samet Akbas	4.900000
2	Munevver Ozturk	4.200000
5	Ismet Eren	3.700000
1	Batuhan Ozturk	3.100000
4	Vehbi Okumus	2.900000
7	Mehmet Ali	2.800000
6	Semih Tarcan	1.300000
9	Ekin Kaya	NULL

Result 14

It is used to determine users energy consumption trends. İt can be used to identify users with high average consumption and offer savings suggestions. Harun öztürk is the user who consumes the highest energy with an average of 5.3 kwh.

```sql
-- Only show users with average consumption higher than 3.0 kwh "HAVING"
 SELECT
     users.user_id,
     users.name,
     AVG(energy_usage.consumption_kwh) AS avg_consumption

FROM users
LEFT JOIN devices ON users.user_id = devices.user_id
LEFT JOIN energy_usage ON devices.device_id = energy_usage.device_id
GROUP BY  users.user_id, users.name
HAVING avg_consumption > 3.0
ORDER BY avg_consumption DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

user_id	name	avg_consumption
3	Harun Ozturk	5.300000
8	Samet Akbas	4.900000
2	Munevver Ozturk	4.200000
5	Ismet Eren	3.700000
1	Batuhan Ozturk	3.100000

We used this SQL query to list users whose average energy consumption is greater than 3.0 kwh.It used to identify users with high average consumption and to offer energy saving measures.

```sql
SELECT  user_id ,name, total_consumption
FROM (
    SELECT
        users.user_id,
        users.name,
        SUM(energy_usage.consumption_kwh) AS total_consumption
    FROM users
    LEFT JOIN devices ON users.user_id = devices.user_id
    LEFT JOIN energy_usage ON devices.device_id = energy_usage.device_id
    GROUP BY users.user_id, users.name
) AS consumption_table
ORDER BY total_consumption DESC
LIMIT 1 ;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

user_id	name	total_consumption
1	Batuhan Ozturk	9.30

In this query , we determine the user who consumes the most energy in the system . Batuhan Öztürk is the user with the highest energy consumption with a total consumption of 9.30 kwh.

NoSQL

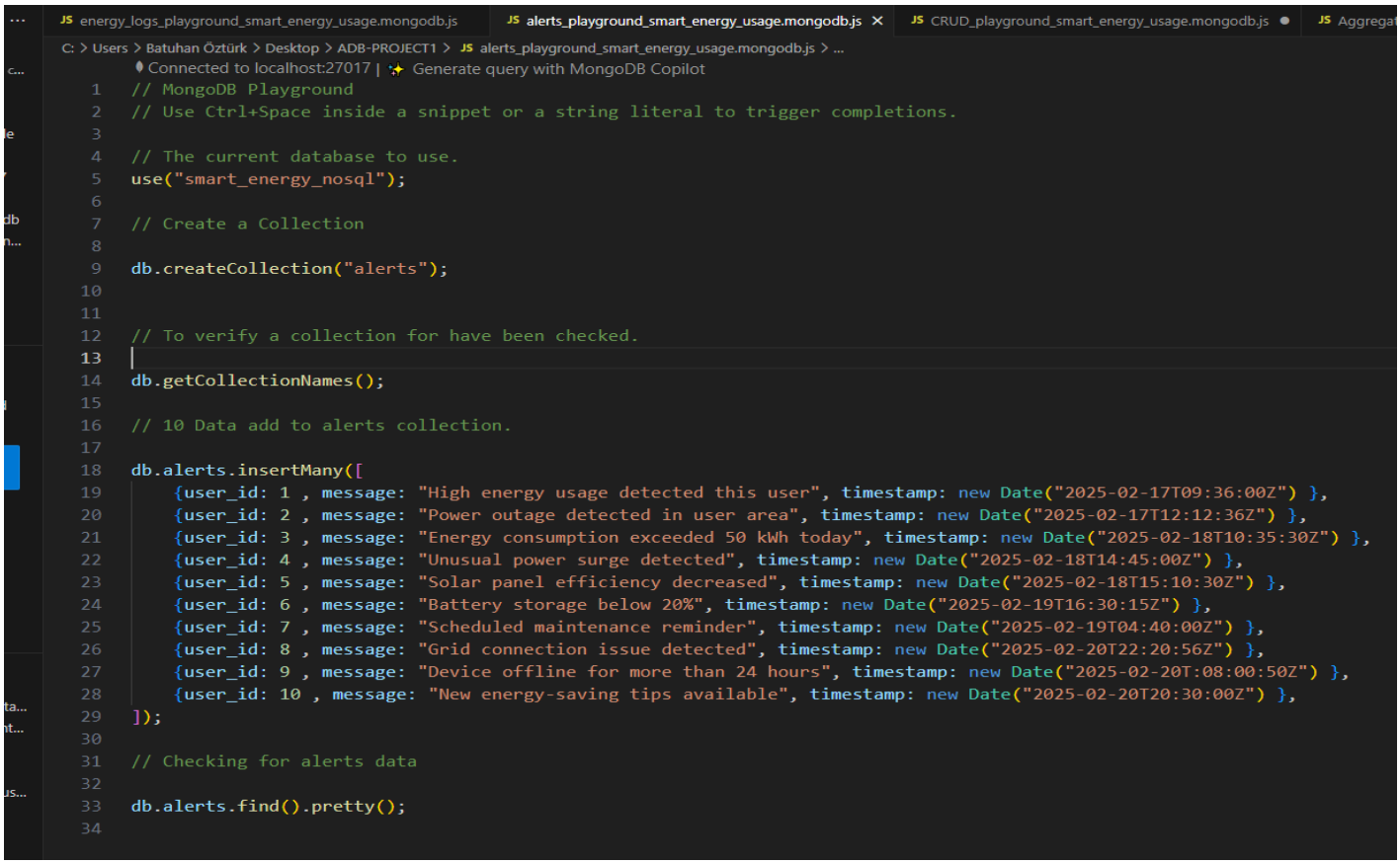
energy_logs_playground_smart_energy_usage.mongodb.js | alerts_playground_smart_energy_usage.mongodb.js | CRUD_playgroun

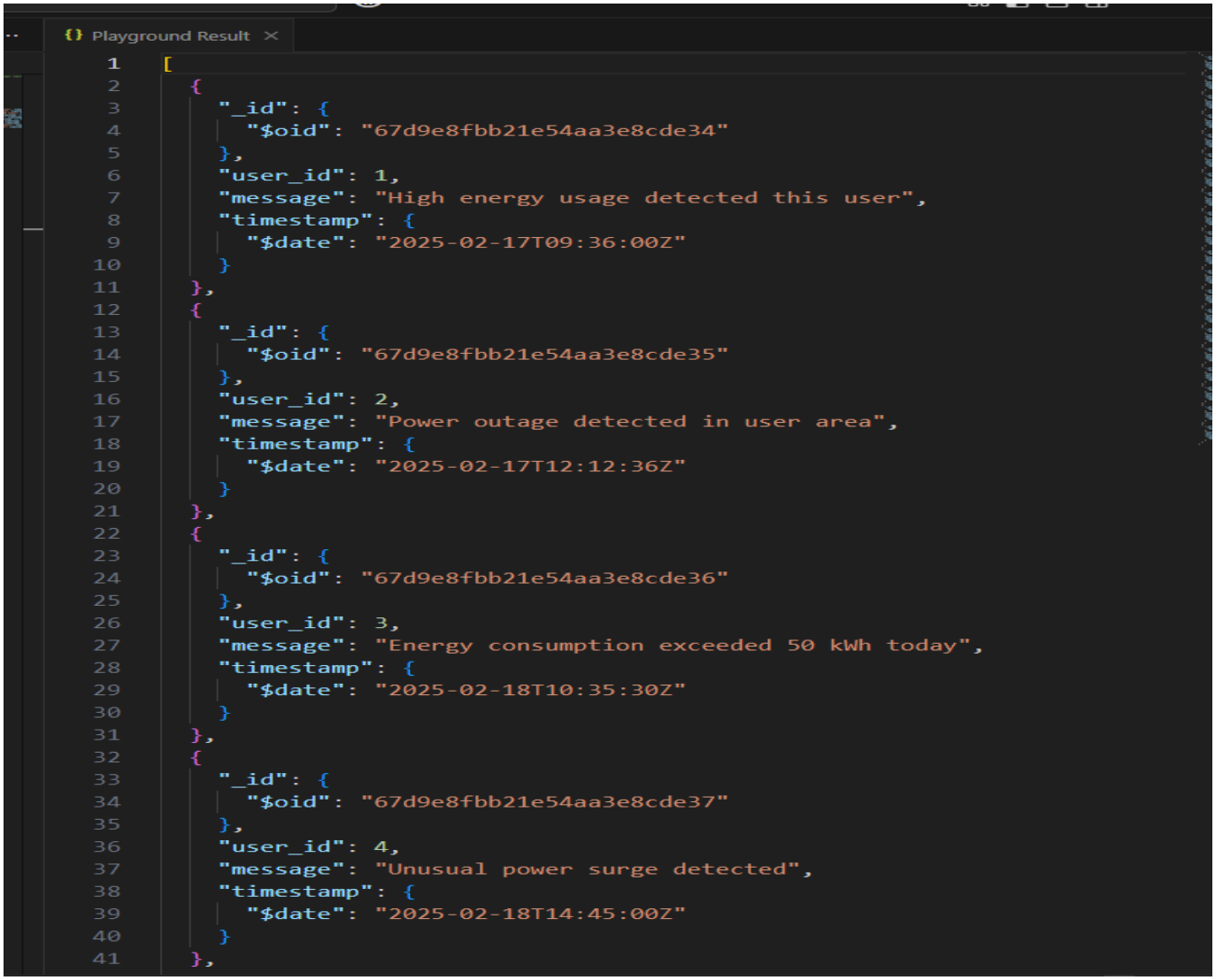
C: > Users > Batuhan Öztürk > Desktop > ADB-PROJECT1 > energy_logs_playground_smart_energy_usage.mongodb.js > ...

Connected to localhost:27017 | Generate query with MongoDB Copilot

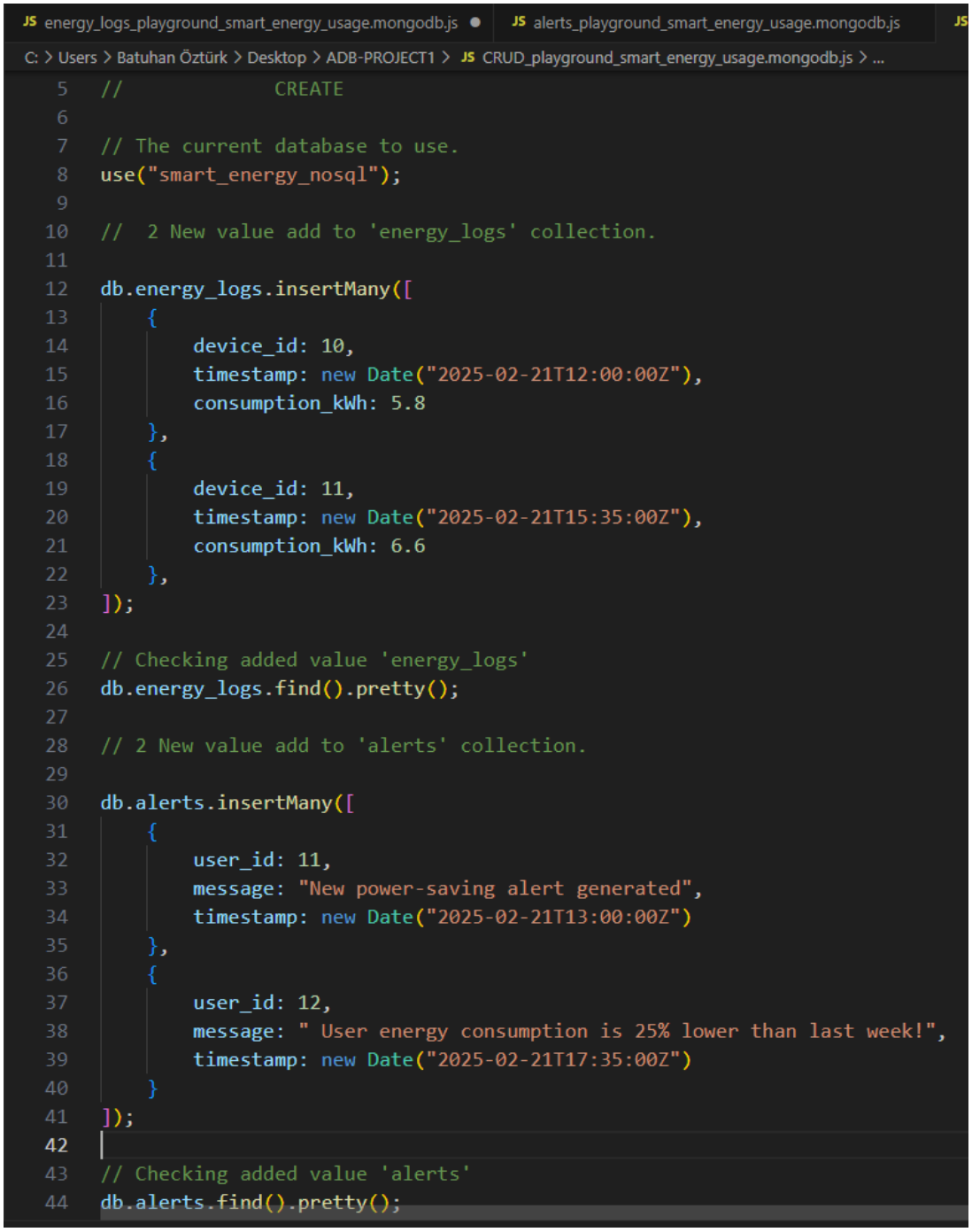
```javascript
// MongoDB Playground
// Use Ctrl+Space inside a snippet or a string literal to trigger completions.

// The current database to use.
use("smart_energy_nosql");

// Create a Collection

db.createCollection("energy_logs");



// To check that collections have been created
db.getCollectionNames();

// 10 data add to energy_logs collections

db.energy_logs.insertMany([
    {device_id: 1,timestamp: new Date("2025-02-17T09:36:00Z"),consumption_kwh:3.5},
    {device_id: 1,timestamp: new Date("2025-02-17T12:12:36Z"),consumption_kwh:2.1},
    {device_id: 2,timestamp: new Date("2025-02-18T10:35:30Z"),consumption_kwh:1.8},
    {device_id: 3,timestamp: new Date("2025-02-18T14:45:00Z"),consumption_kwh:4.2},
    {device_id: 4,timestamp: new Date("2025-02-18T15:10:30Z"),consumption_kwh:5.3},
    {device_id: 5,timestamp: new Date("2025-02-19T16:30:15Z"),consumption_kwh:2.9},
    {device_id: 6,timestamp: new Date("2025-02-19T04:40:00Z"),consumption_kwh:3.7},
    {device_id: 7,timestamp: new Date("2025-02-20T22:20:56Z"),consumption_kwh:1.3},
    {device_id: 8,timestamp: new Date("2025-02-20T08:00:50Z"),consumption_kwh:2.8},
    {device_id: 9,timestamp: new Date("2025-02-20T20:30:00Z"),consumption_kwh:4.9},
]);

// Checking for dataset

db.energy_logs.find().pretty();
```

- We created the smart_energy_nosql database via MongoDB playground. Then , we created the energy_logs collection and added 10 test data , after than we verified that the data was added. You can see results in above document.

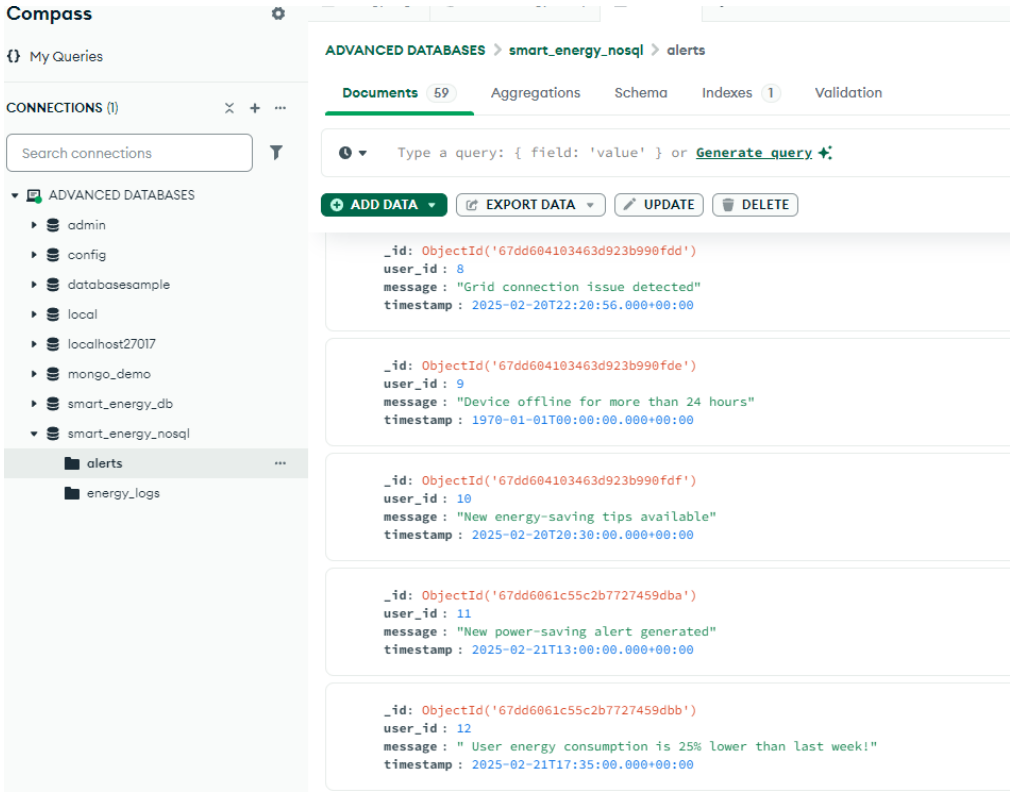


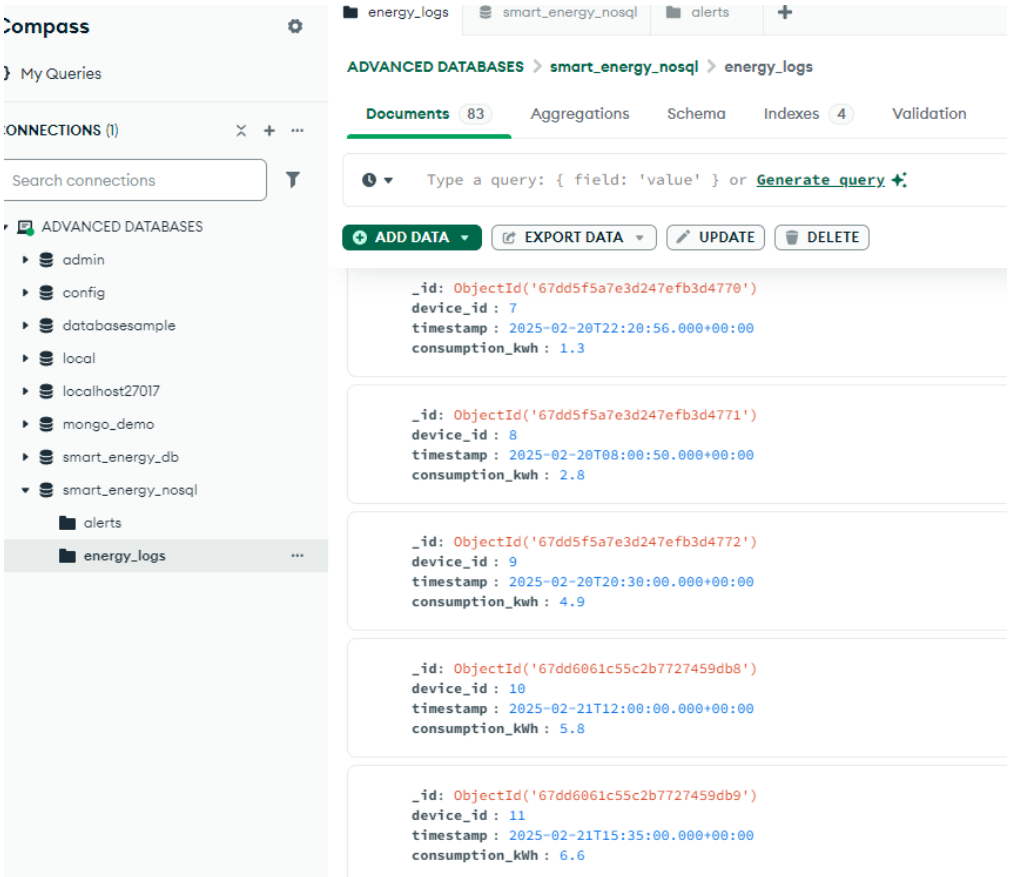


-We created the smart_energy_nosql database from MongoDB Playground when first step .Next, we created the alerts collection and added 10 test data like energy_logs, then verified that data was added. You can see the results in the results document above. We have enabled users to follow possible critical situations in their energy consumption systems and arranged to send notification messages to the user accordingly.

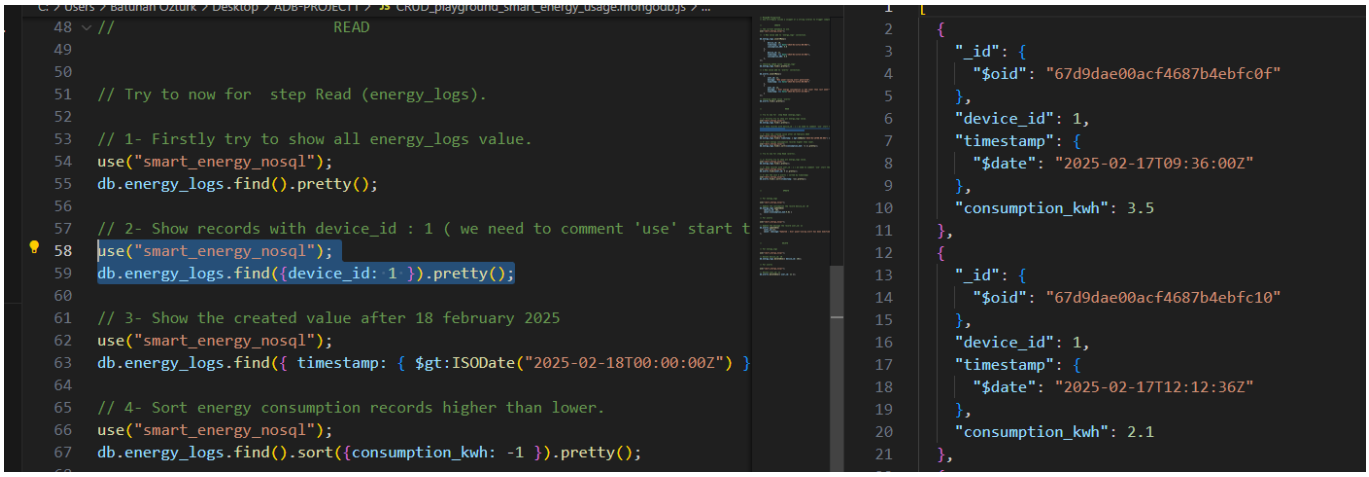


After running my code through MongoDB Playground, new data was added to the nergy logs and alerts collection in the smart energy NoSQL database.





When I checked later with MongoDB compass ,I saw that the data ia dded was saved successfully.The consumption values of the devices were added to the energy_logs collection, and the user warning messages were added to the alerts collection and displayed correctly.



```
48  //                          READ
49
50
51  // Try to now for  step Read (energy_logs).
52
53  // 1- Firstly try to show all energy_logs value.
54  use("smart_energy_nosql");
55  db.energy_logs.find().pretty();
56
57  // 2- Show records with device_id : 1 ( we need to comment 'use' start t
58  use("smart_energy_nosql");
59  db.energy_logs.find({device_id: 1 }).pretty();
60
61  // 3- Show the created value after 18 february 2025
62  use("smart_energy_nosql");
63  db.energy_logs.find({ timestamp: { $gt:ISODate("2025-02-18T00:00:00Z") }
64
65  // 4- Sort energy consumption records higher than lower.
66  use("smart_energy_nosql");
67  db.energy_logs.find().sort({consumption_kwh: -1 }).pretty();
```

```
1  [
2    {
3      "_id": {
4        "$oid": "67d9dae00acf4687b4ebfc0f"
5      },
6      "device_id": 1,
7      "timestamp": {
8        "$date": "2025-02-17T09:36:00Z"
9      },
10     "consumption_kwh": 3.5
11   },
12   {
13     "_id": {
14       "$oid": "67d9dae00acf4687b4ebfc10"
15     },
16     "device_id": 1,
17     "timestamp": {
18       "$date": "2025-02-17T12:12:36Z"
19     },
20     "consumption_kwh": 2.1
21   },
```

-It brings the energy consumption records of the device with device_id ; 1

```
60
61  // 3- Show the created value after 18 february 2025
62  use("smart_energy_nosql");
63  db.energy_logs.find({ timestamp: { $gt:ISODate("2025-02-18T00:00:00Z") } }).pretty();
64
65  // 4- Sort energy consumption records higher than lower.
66  use("smart_energy_nosql");
67  db.energy_logs.find().sort({consumption_kwh: -1 }).pretty();
68
69
```

```
22   {
23     "_id": {
24       "$oid": "67dbfab968258eaafc94b1fa"
25     },
26     "device_id": 2,
27     "timestamp": {
28       "$date": "2025-02-18T10:35:30Z"
29     },
30     "consumption_kwh": 1.8
31   },
```

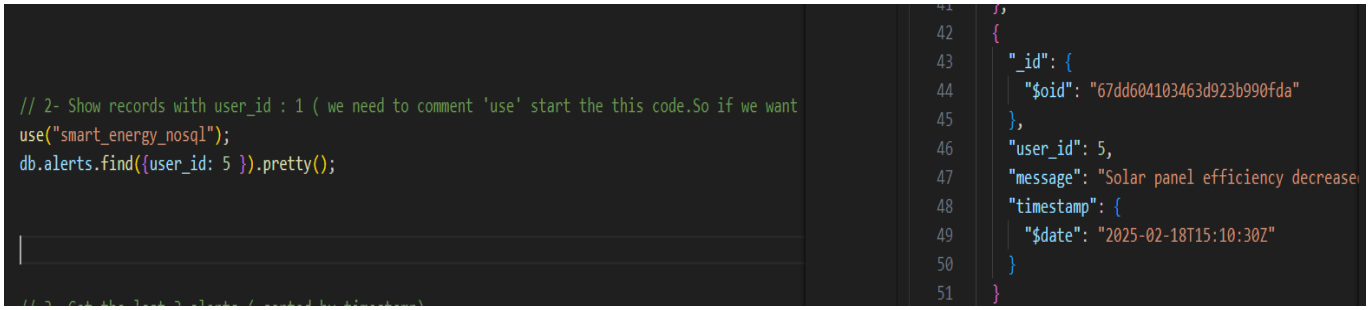
-Lists energy consumption records after 2025-02-18 00:00:00: UTC . The $gt (greater than) operator returns records greater than this date.

```
64
65  // 4- Sort energy consumption records higher than lower.
66  use("smart_energy_nosql");
67  db.energy_logs.find().sort({consumption_kwh: -1 }).pretty();
68
69
70
71  // Try to now for step Read (alerts).
72
73
74  // 1- Firstly try to show all energy_logs value.
```

```
31   },
32   {
33     "_id": {
34       "$oid": "67d9e1c592f1dac9226b3f0f"
35     },
36     "device_id": 4,
37     "timestamp": {
38       "$date": "2025-02-18T15:10:30Z"
39     },
40     "consumption_kwh": 5.3
41   },
```

It pulls all records from the energy_logs collection.It sorts from largest to smallest according to consumption value. -1 = largest to smallest , 1= from smallest to largest.

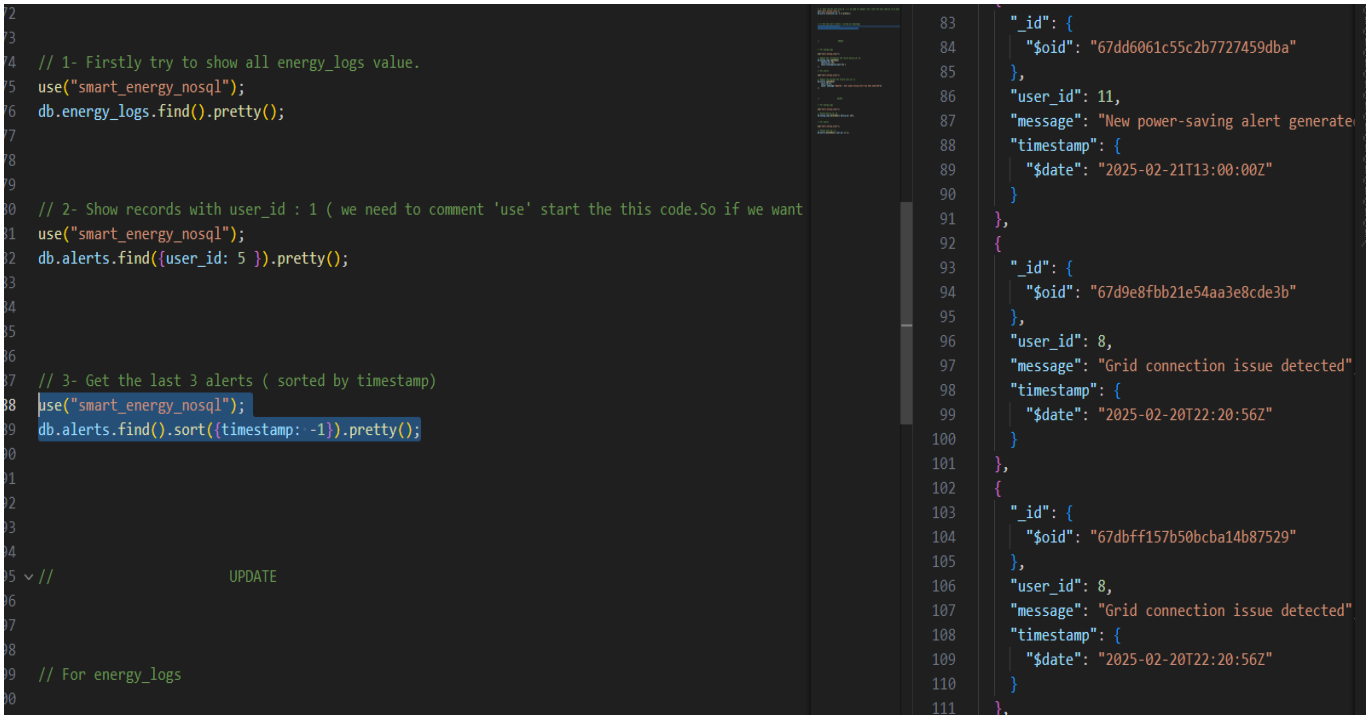
We try to same function for alerts collection .



```
// 2- Show records with user_id : 1 ( we need to comment 'use' start the this code.So if we want
use("smart_energy_nosql");
db.alerts.find({user_id: 5 }).pretty();
```

```
42   {
43     "_id": {
44       "$oid": "67dd604103463d923b990fda"
45     },
46     "user_id": 5,
47     "message": "Solar panel efficiency decrease
48     "timestamp": {
49       "$date": "2025-02-18T15:10:30Z"
50     }
51   }
```

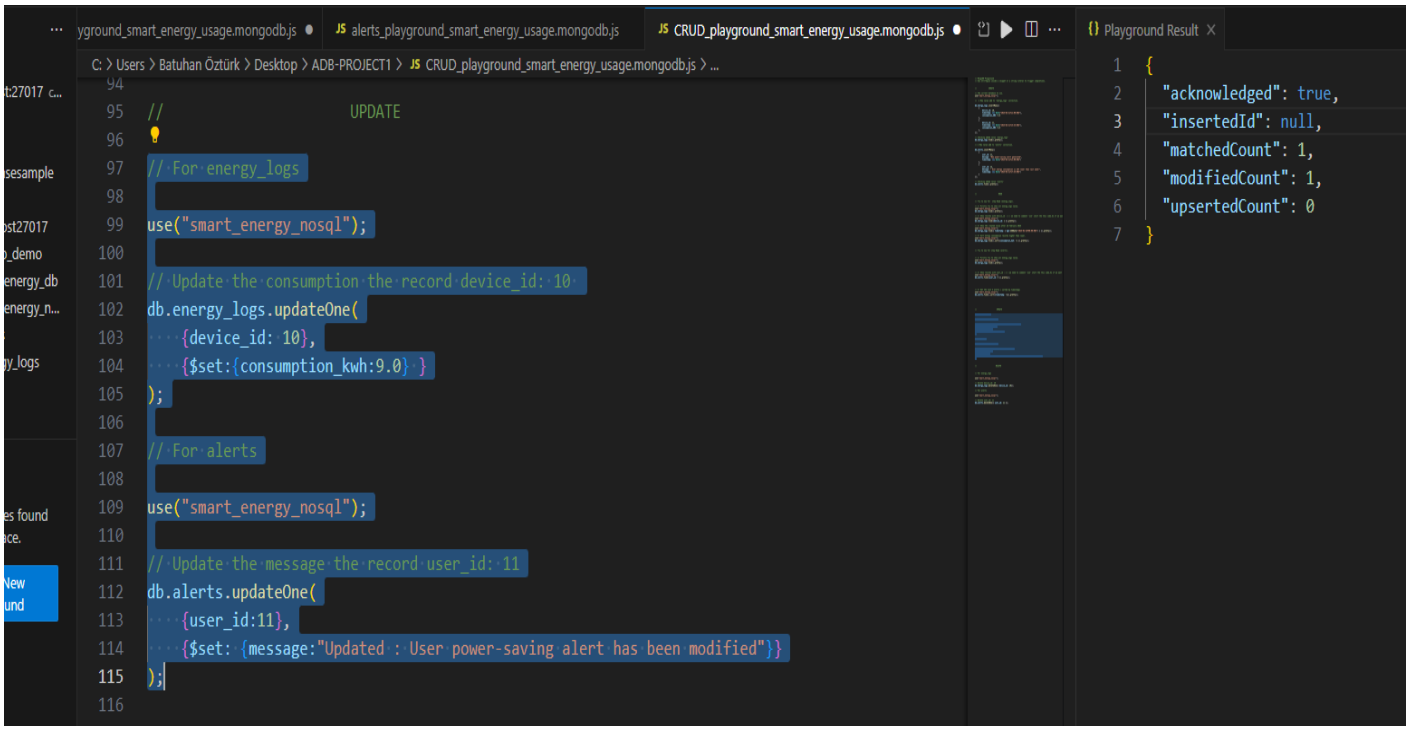
It brings all warning messages of the user whose user_id= 5



```
// 1- Firstly try to show all energy_logs value.
use("smart_energy_nosql");
db.energy_logs.find().pretty();

// 2- Show records with user_id : 1 ( we need to comment 'use' start the this code.So if we want
use("smart_energy_nosql");
db.alerts.find({user_id: 5 }).pretty();

// 3- Get the last 3 alerts ( sorted by timestamp)
use("smart_energy_nosql");
db.alerts.find().sort({timestamp: -1}).pretty();

//                    UPDATE

// For energy_logs
```

```
  "_id": {
    "$oid": "67dd6061c55c2b7727459dba"
  },
  "user_id": 11,
  "message": "New power-saving alert generate
  "timestamp": {
    "$date": "2025-02-21T13:00:00Z"
  }
},
{
  "_id": {
    "$oid": "67d9e8fbb21e54aa3e8cde3b"
  },
  "user_id": 8,
  "message": "Grid connection issue detected"
  "timestamp": {
    "$date": "2025-02-20T22:20:56Z"
  }
},
{
  "_id": {
    "$oid": "67dbff157b50bcba14b87529"
  },
  "user_id": 8,
  "message": "Grid connection issue detected"
  "timestamp": {
    "$date": "2025-02-20T22:20:56Z"
  }
},
```

This step , sorts the data in the alerts collection by the timestmap field “-1” shows the most recent records at the top.



```
yground_smart_energy_usage.mongodb.js   alerts_playground_smart_energy_usage.mongodb.js   CRUD_playground_smart_energy_usage.mongodb.js
C: > Users > Batuhan Öztürk > Desktop > ADB-PROJECT1 > CRUD_playground_smart_energy_usage.mongodb.js > ...

//                    UPDATE

// For energy_logs

use("smart_energy_nosql");

// Update the consumption the record device_id: 10
db.energy_logs.updateOne(
    {device_id: 10},
    {$set:{consumption_kwh:9.0} }
);

// For alerts

use("smart_energy_nosql");

// Update the message the record user_id: 11
db.alerts.updateOne(
    {user_id:11},
    {$set: {message:"Updated : User power-saving alert has been modified"}}
);
```

```
Playground Result
{
  "acknowledged": true,
  "insertedId": null,
  "matchedCount": 1,
  "modifiedCount": 1,
  "upsertedCount": 0
}
```

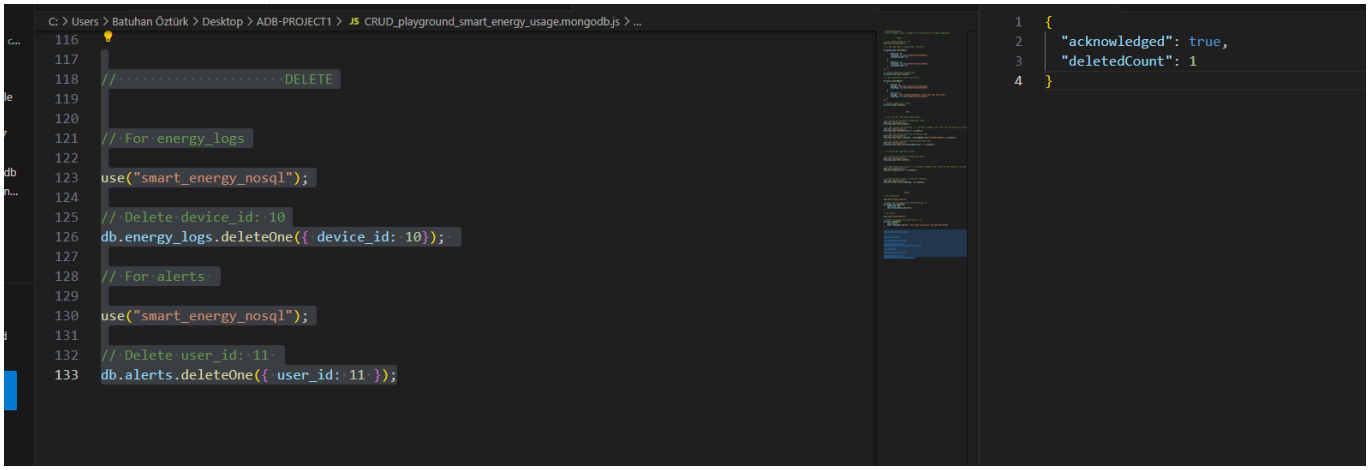
The consumption kwh value of the record with device_id: 10 in the energy_logs collection has been updated to 9.0 . The message field of the record with user_id:11 in the alerts collection has been updated to “ Updated: User power saving alert has been modified”.

Acknowledget : True = True MongoDB successfully received and processed query.

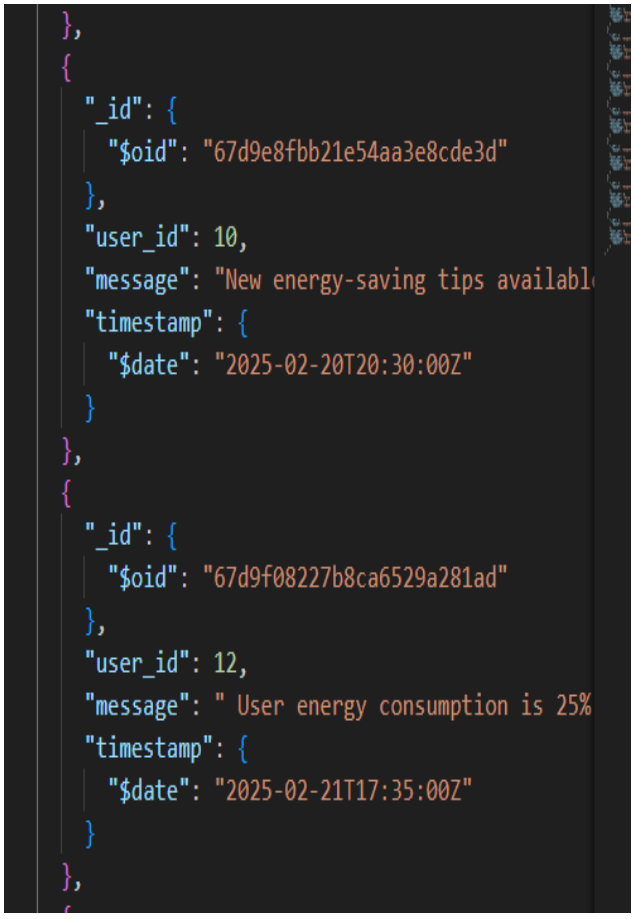
inserteId: null = No new data was added

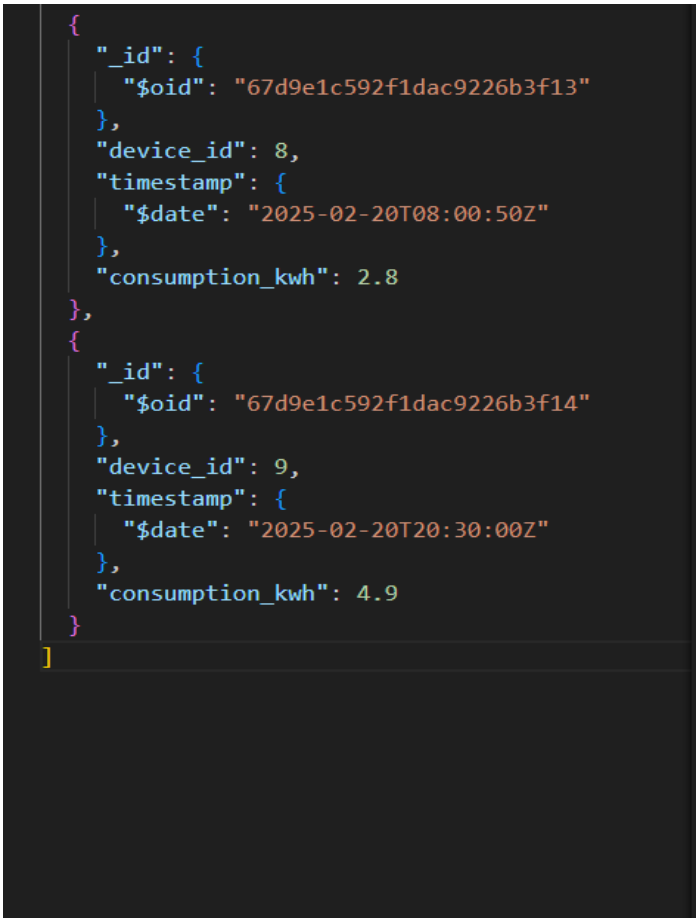
matchedCount : 1 = record matched for updated.

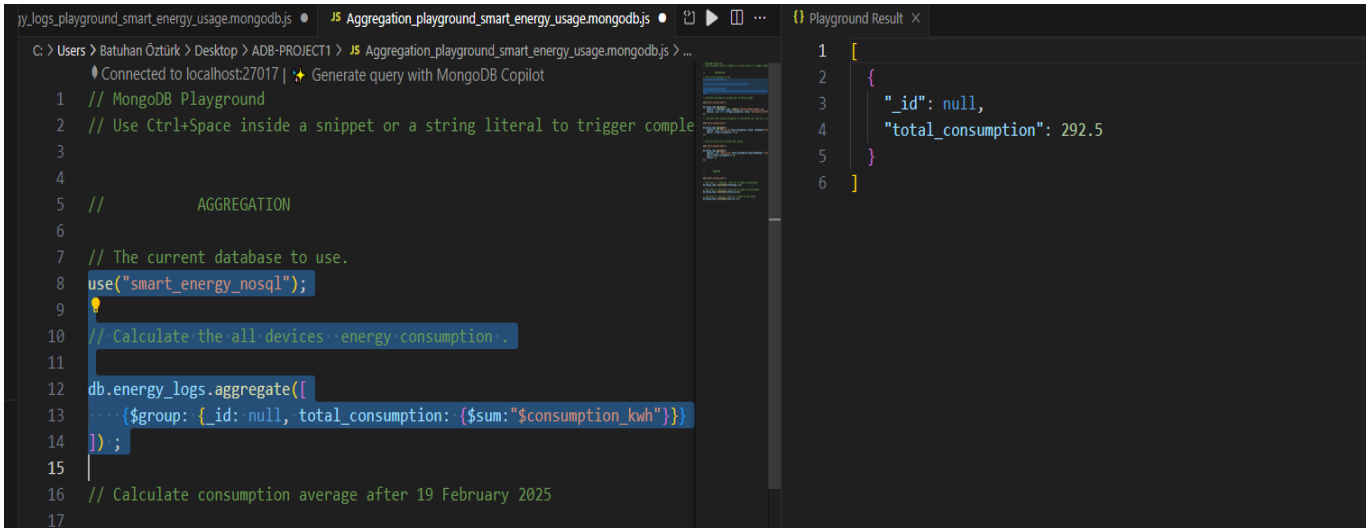
modifiedCount:1 = Number of records updated 1



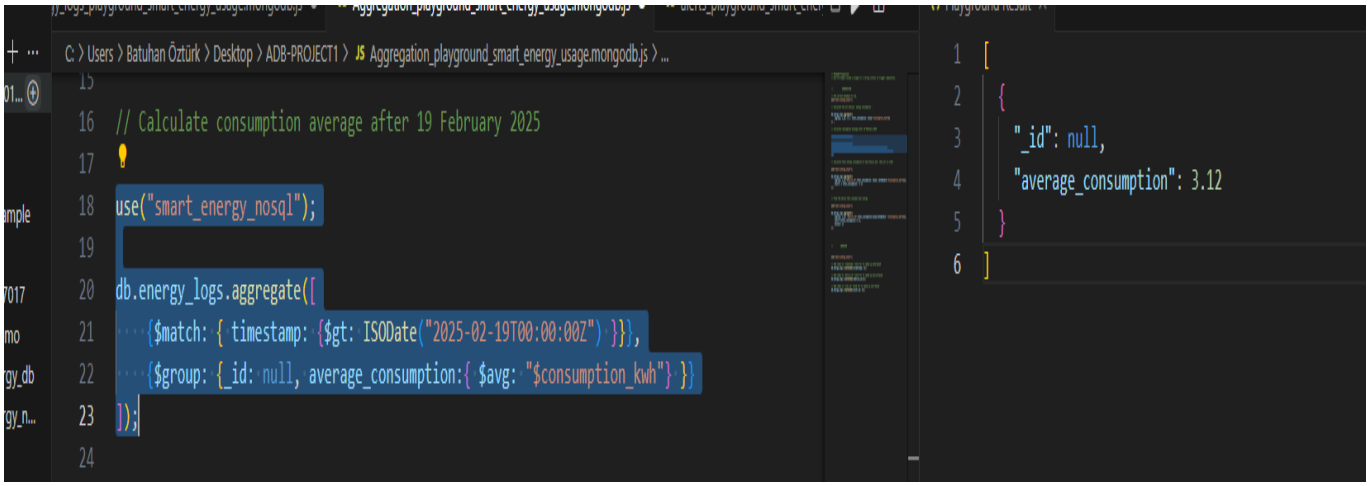
Device_id ; 10 deleted from energy_logs and user_id : 11 deleted from user_id 11.



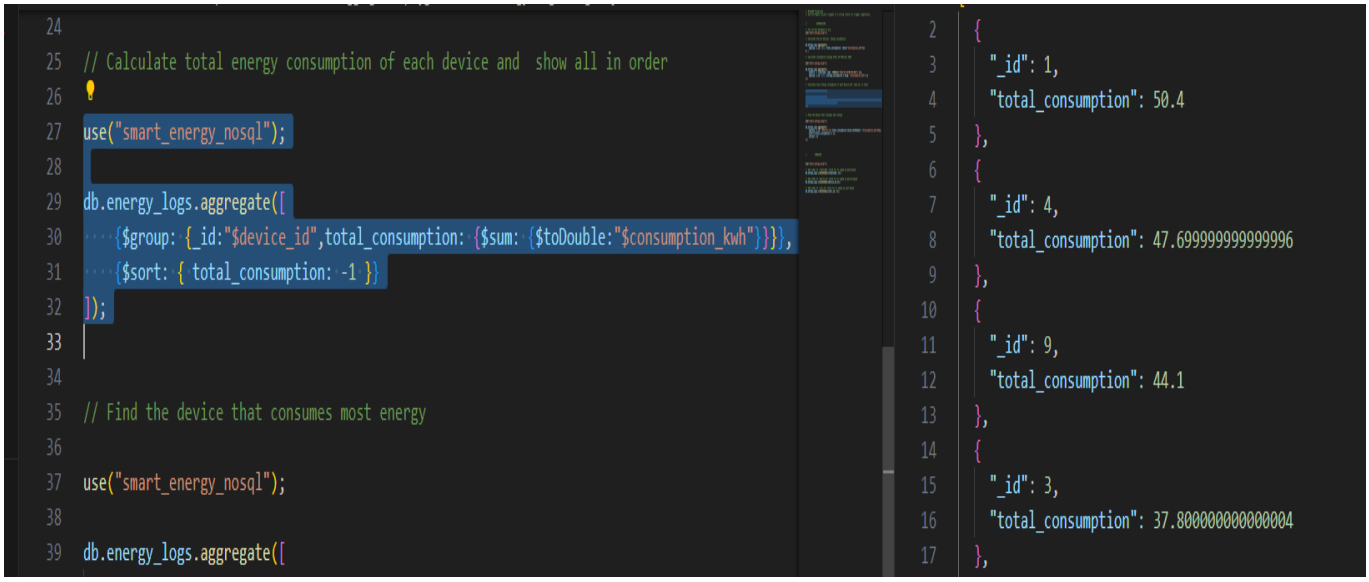


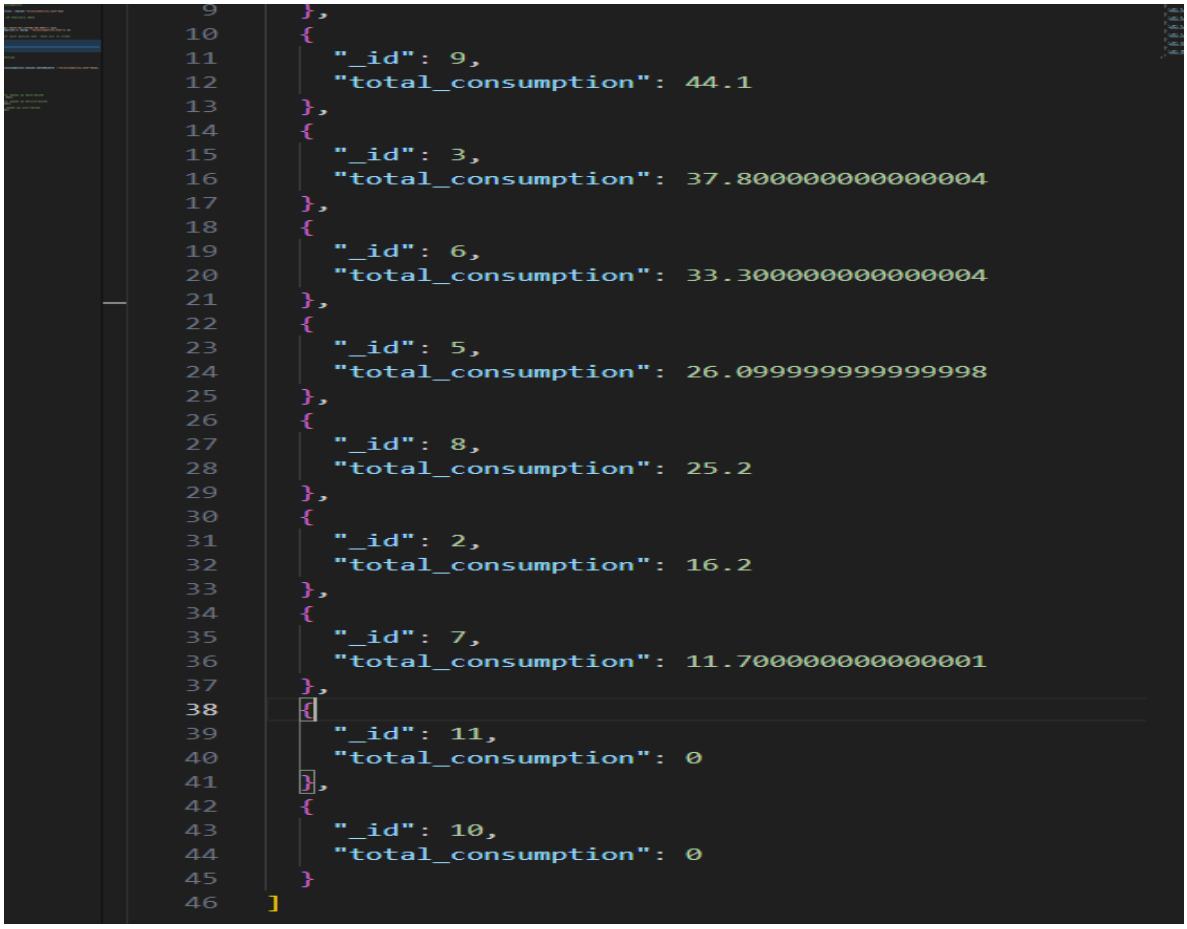


A total consumption of 292.5 kwh was calculated . This gives the sum of the energy consumption data.



An average consumption value of 3.12 kwh was calculated.This only represents the average of data recorded after February 19,2025.





In this code we analyze the total energy consumption data of the devices . Devices with the same device_id are grouped and the total consumption is calculated.

CONCLUSION

Within the scope of this project, i have successfully developed a Smart Energy Management System that integrates SQL and NoSQL databases. During the project process, we designed and created a scalable and performance optimized hybrid database system that can effectively manage energy consumption data. Thanks to the combined use of MySQL and MongooDB , we have created a flexible and efficient data management model by taking advantage of the strengths of relational and non-relational databases.

The main points we have achieved with this project are as followers:

- Successfull creation and management of SQL tables and NoSQL collections
- Implementation of CRUD operations for SQL and NoSQL databases
- Development of SQL queries such as JOIN, GROUP BY , HAVING, SUM , AVG that provide analytical insights.
- Production of accurate and comprehensive analysis reports based on energy consumption data .

As a result of the tests we condructed, we saw that the system confirmed that it can efficiently process energy consumption records, user alerts and device logs.The designed database structure ensures fast and efficient queries, while the created indexing mechanisms significantly improve data access time.In this project , we wanted to demostrate the effectiveness of the hybrid SQL-NoSQL approach in energy management systems and provide a strong infrastructure for such applications.

FUTURE WORK

Although this project currently meets the basic requierements, it can be made much more powerful with some improvements in the future. Here are some forward looking suggestions:

- Forecasting with Machine Learning = Models that predict future energy consumption can be developed using current energy consumption.

- Optimization for Larger-Scale Data = We can enable the system to manage much larger data with methods such as sharding and replication.

- A Better User Interface = A dashboard that can track real-time data via web or mobile can be developed. In my opinion this designed look to be the biggest deficiency in the project right now.

- Security Improvement = We can add stronger authenticantion mechanisms for user authorization and data security . A password can be a simple security measure.