Login2Xplore JsonPowerDB

1.JsonPowerDB and its features

JsonPowerDB is a high-performance, real-time database management system designed for modern applications. It focuses on providing speed, security, and flexibility.

Features of JsonPowerDB

1. \*\*High Performance\*\*: Optimized for fast processing and querying.

2. \*\*Real-Time\*\*: Allows immediate data management and updates.

3. \*\*Ease of Use\*\*: Simple setup with minimal configuration required.

4. \*\*RESTful API\*\*: Supports seamless integration with other applications.

5. \*\*In-Memory Indexing\*\*: Facilitates quick data access.

6. \*\*Flexibility\*\*: Accommodates multiple data types and structures.

7. \*\*Scalability\*\*: Efficiently handles increasing data volumes.

8. \*\*Security\*\*: Incorporates multi-layer security protocols.

2. JsonPowerDB use case and its benefits

JsonPowerDB Use Cases

1. \*\*Web and Mobile Applications\*\*: Ideal for applications requiring fast and real-time data handling.

2. \*\*IoT Devices\*\*: Suitable for managing and querying large volumes of data from connected devices.

3. \*\*Analytics and Reporting\*\*: Efficient for high-speed data analysis and generating reports.

4. \*\*Document Storage\*\*: Supports flexible document-oriented storage and retrieval.

### Benefits of JsonPowerDB

1. \*\*Speed\*\*: High performance ensures quick data processing.

2. \*\*Scalability\*\*: Can handle growing data volumes efficiently.

3. \*\*Ease of Integration\*\*: RESTful API facilitates easy integration with other systems.

4. \*\*Flexibility\*\*: Supports various data types and structures.

5. \*\*Security\*\*: Multi-layer security protects data integrity.

3. PUT Command - Creating (Inserting) RecordPage

The PUT command in JsonPowerDB is used to create or insert a new record into the database. Here’s how you can use it:

Syntax:

```plaintext

PUT /<DB-Name>/<Collection-Name>/{Primary Key}

Content-Type: application/json

{

"field1": "value1",

"field2": "value2",

...

}

```

### Example:

#### Request:

```http

PUT /exampleDB/employee/123

Content-Type: application/json

{

"name": "John Doe",

"age": 30,

"department": "Sales"

}

```

#### Response:

```json

{

"status": "success",

"message": "Record inserted successfully",

"data": {

"name": "John Doe",

"age": 30,

"department": "Sales"

}

}

```

### Steps:

1. \*\*Specify the Database and Collection\*\*: Use the format `/DB-Name/Collection-Name/{Primary Key}`.

2. \*\*Define the Data\*\*: Provide the data to be inserted in JSON format.

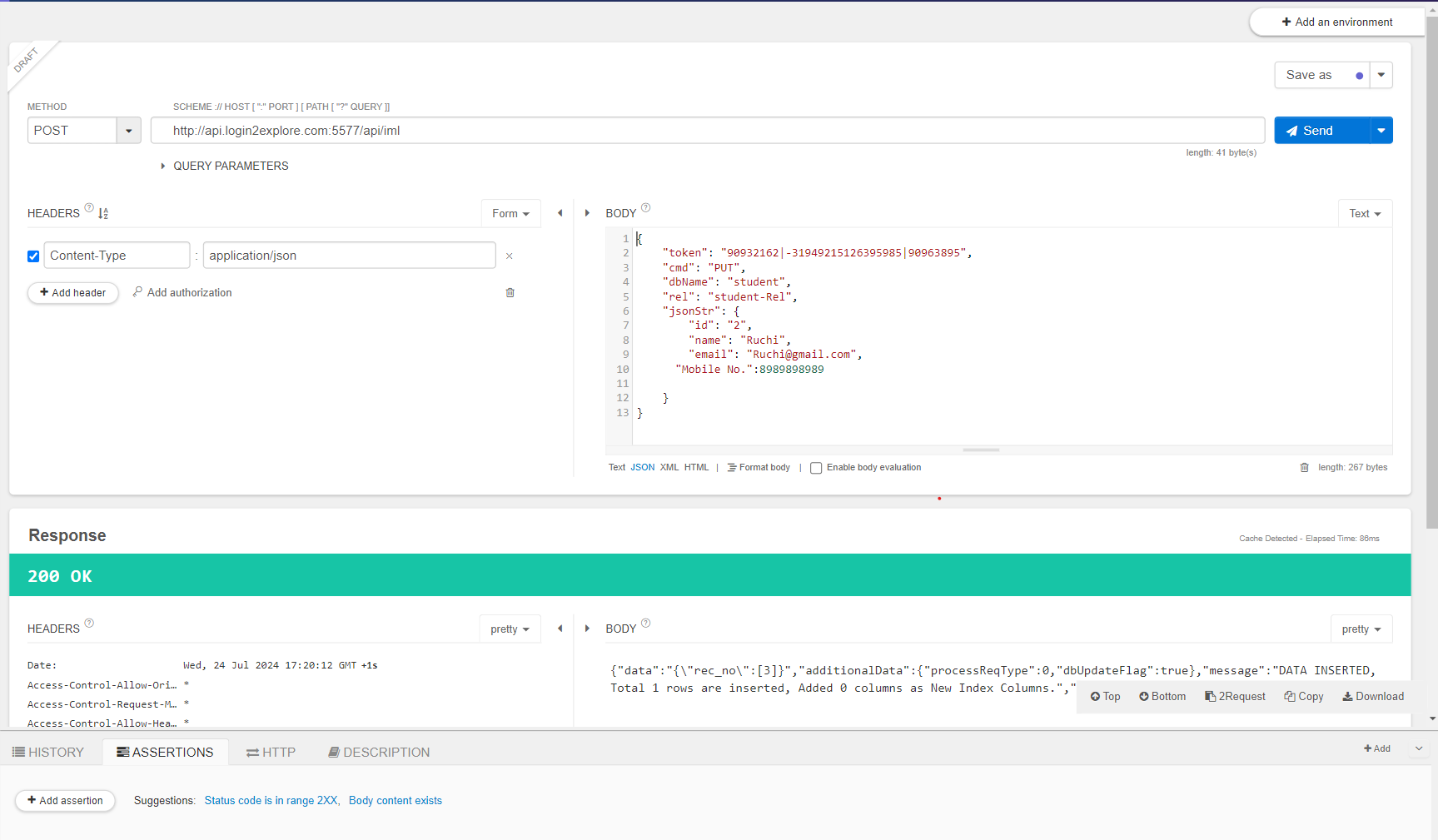
3. \*\*Send the Request\*\*: Make an HTTP PUT request with the data.

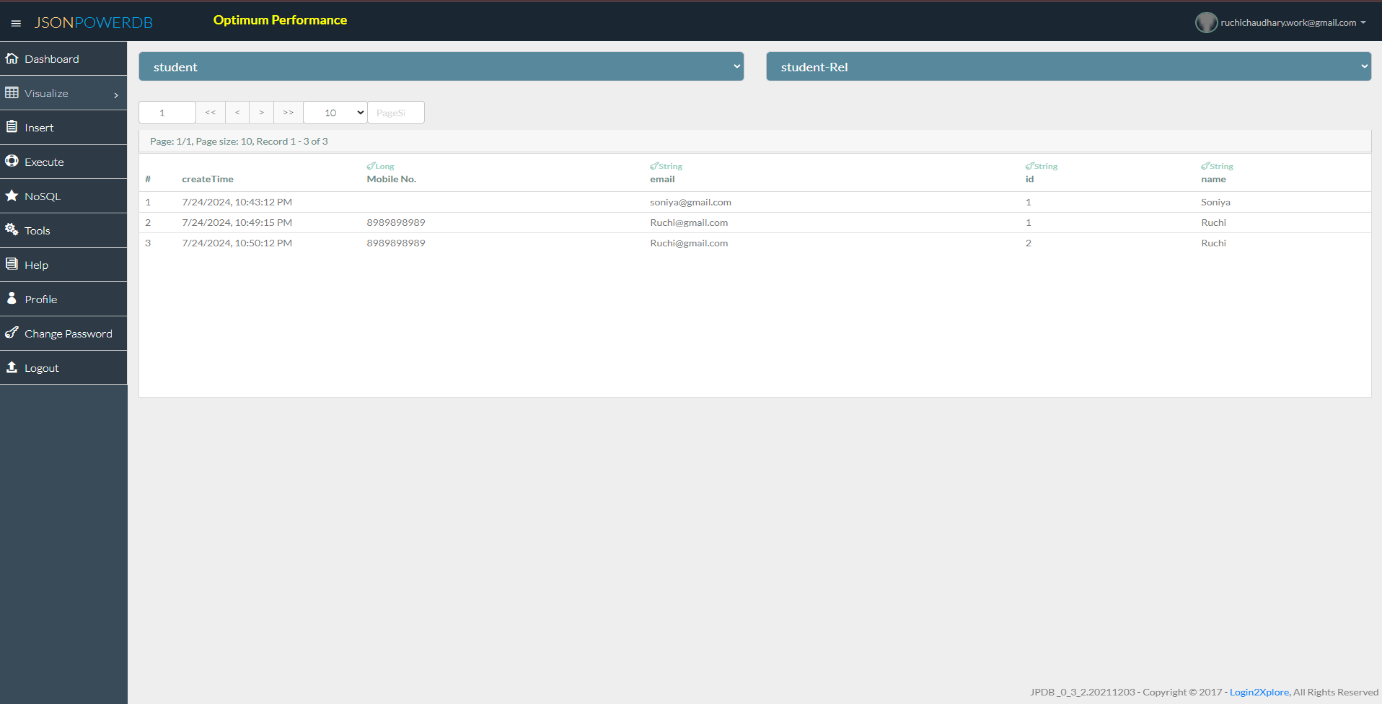
### Benefits:

- \*\*Efficiency\*\*: Quick insertion of records.

- \*\*Flexibility\*\*: Easily insert different data structures.

- \*\*Real-Time\*\*: Immediate reflection of inserted data.





4.GET Command - Retrieving a Record in JsonPowerDB

The GET command in JsonPowerDB is used to retrieve a specific record from the database.

#### Syntax:

```plaintext

GET /<DB-Name>/<Collection-Name>/{Primary Key}

```

### Example:

#### Request:

```http

GET /exampleDB/employee/123

```

#### Response:

```json

{

"status": "success",

"data": {

"name": "John Doe",

"age": 30,

"department": "Sales"

}

}

```

### Steps:

1. \*\*Specify the Database and Collection\*\*: Use the format `/DB-Name/Collection-Name/{Primary Key}`.

2. \*\*Send the Request\*\*: Make an HTTP GET request.

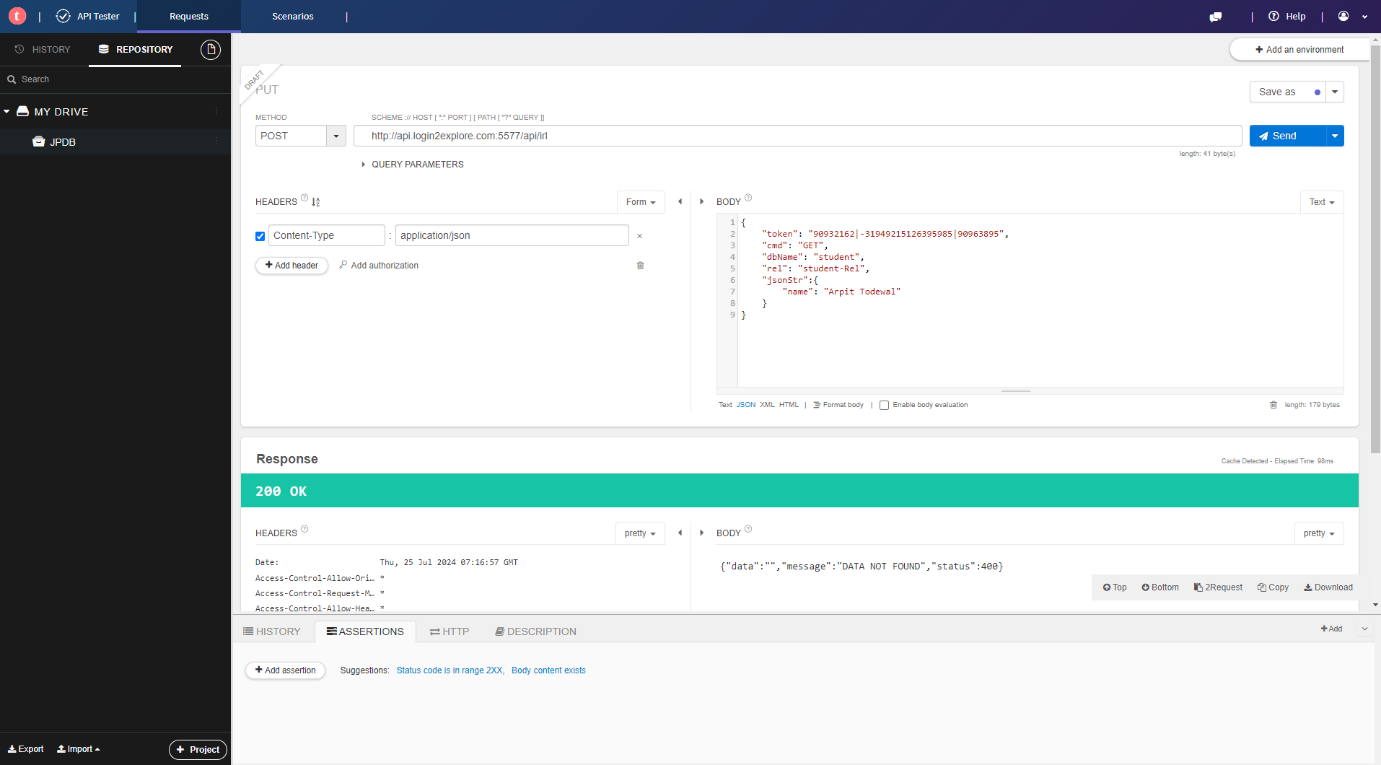
3. \*\*Receive the Response\*\*: The response contains the requested record.

### Benefits:

- \*\*Quick Access\*\*: Fast retrieval of records.

- \*\*Real-Time Data\*\*: Immediate access to up-to-date information.

- \*\*Simplicity\*\*: Easy to use with minimal configuration.



5.UPDATE Command - Updating a Record in JsonPowerDB

The UPDATE command in JsonPowerDB allows you to modify an existing record, requiring authentication via a token.

#### Syntax:

```plaintext

PUT /<DB-Name>/<Collection-Name>/{Primary Key}

x-access-token: <your-token>

Content-Type: application/json

{

"field1": "new\_value1",

"field2": "new\_value2",

...

}

```

### Example:

#### Request:

```http

PUT /exampleDB/employee/123

x-access-token: your-token

Content-Type: application/json

{

"age": 31,

"department": "Marketing"

}

```

#### Response:

```json

{

"status": "success",

"message": "Record updated successfully",

"data": {

"name": "John Doe",

"age": 31,

"department": "Marketing"

}

}

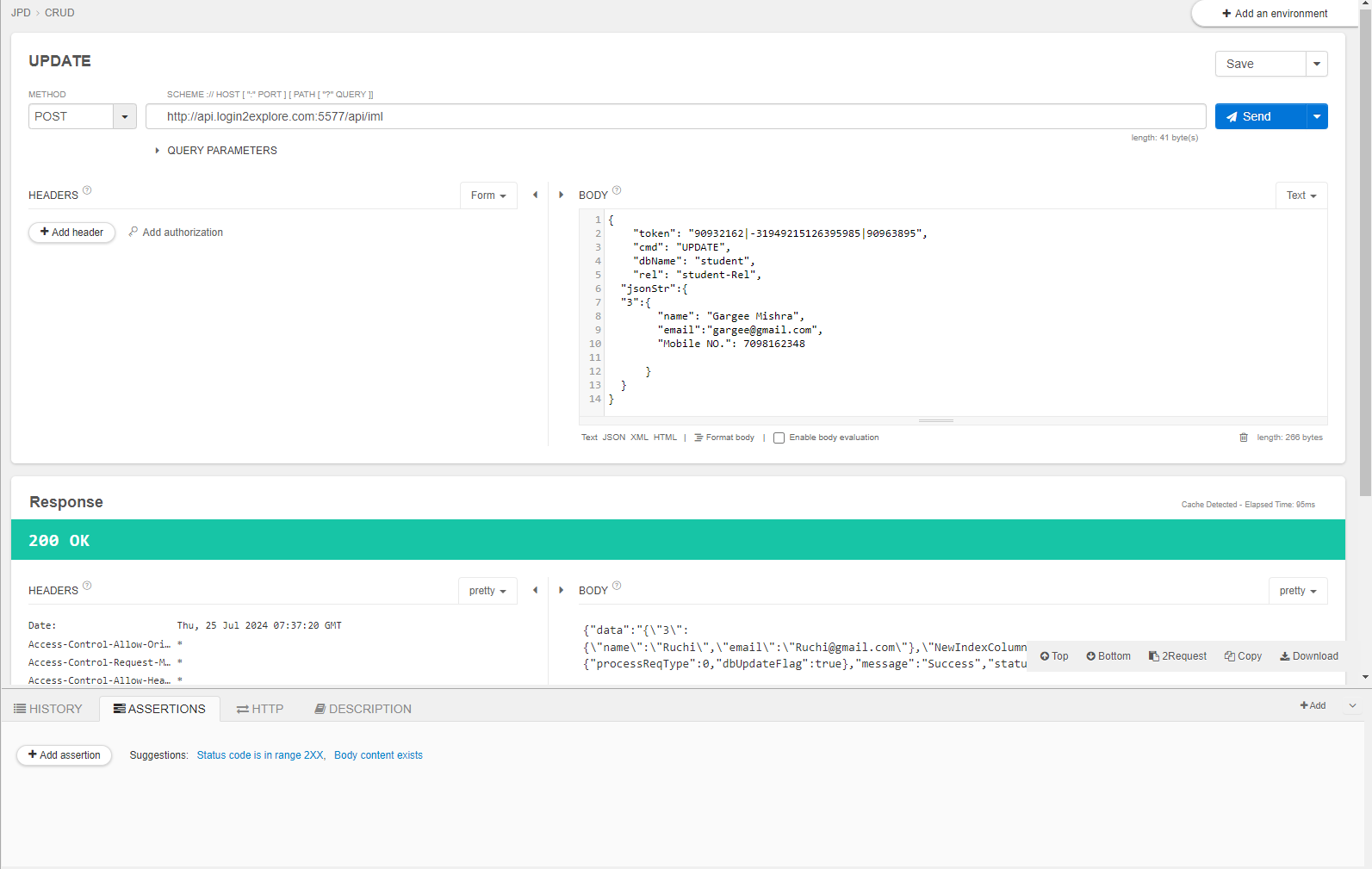
```

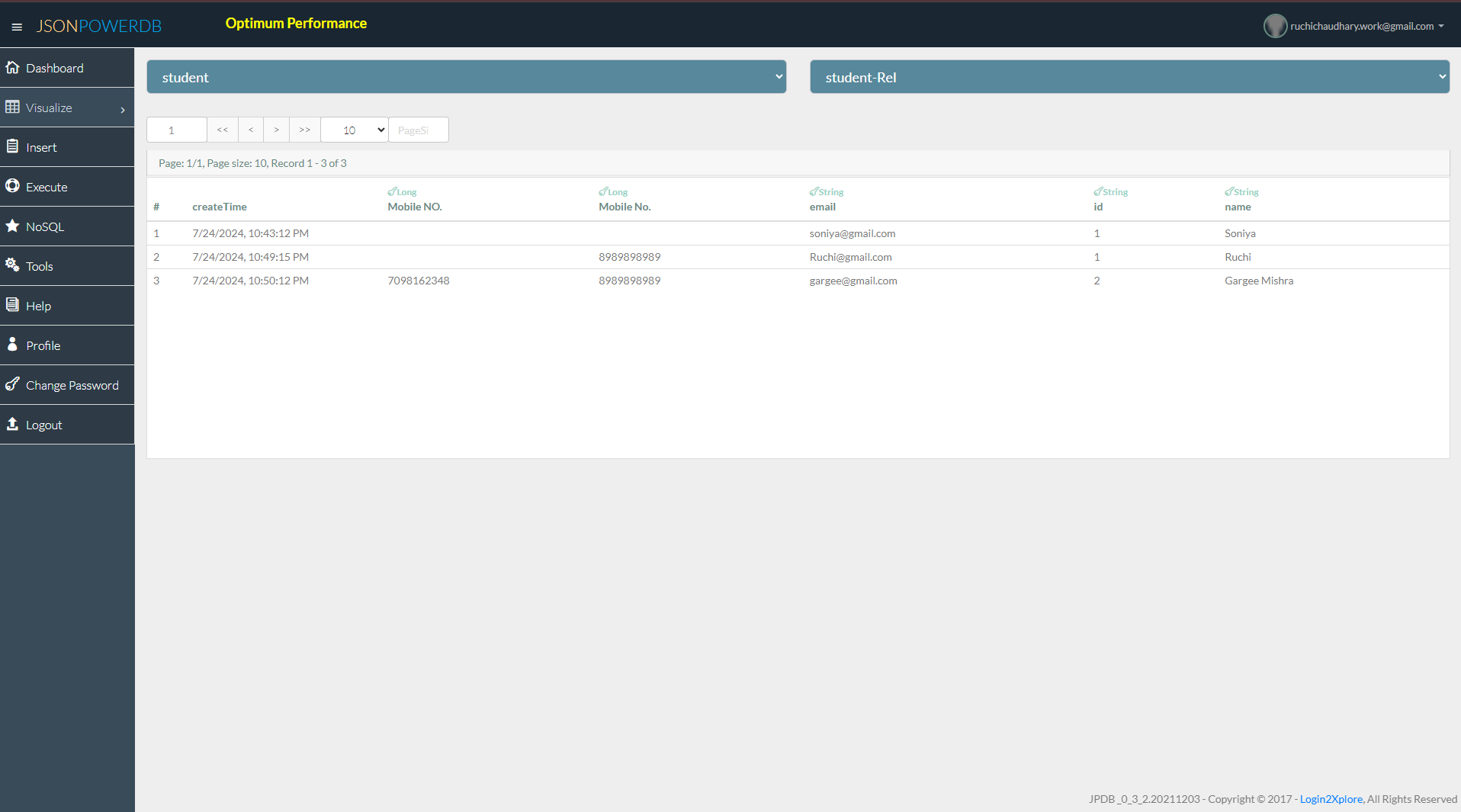
### Steps:

1. \*\*Specify the Database and Collection\*\*: Format as `/DB-Name/Collection-Name/{Primary Key}`.

2. \*\*Include Token\*\*: Add `x-access-token` header with your token.

3. \*\*Define the Updated Data\*\*: Provide the new data in JSON format.

4. \*\*Send the Request\*\*: Make an HTTP PUT request with the updated data and token.



6. REMOVE Command - Removing a Record in JsonPowerDB

The REMOVE command in JsonPowerDB is used to delete a specific record from the database, and it requires an authentication token.

#### Syntax:

```plaintext

DELETE /<DB-Name>/<Collection-Name>/{Primary Key}

x-access-token: <your-token>

```

### Example:

#### Request:

```http

DELETE /exampleDB/employee/123

x-access-token: your-token

```

#### Response:

```json

{

"status": "success",

"message": "Record removed successfully"

}

```

### Steps:

1. \*\*Specify the Database and Collection\*\*: Use the format `/DB-Name/Collection-Name/{Primary Key}`.

2. \*\*Include Token\*\*: Add `x-access-token` header with your token.

3. \*\*Send the Request\*\*: Make an HTTP DELETE request with the token.

