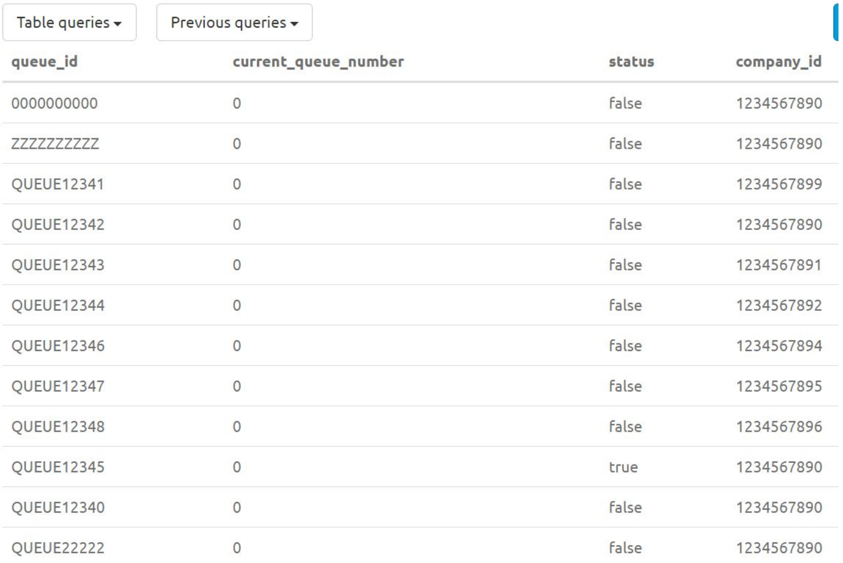
Table Of Content

1. Create Queue API

1.1 Successful Request

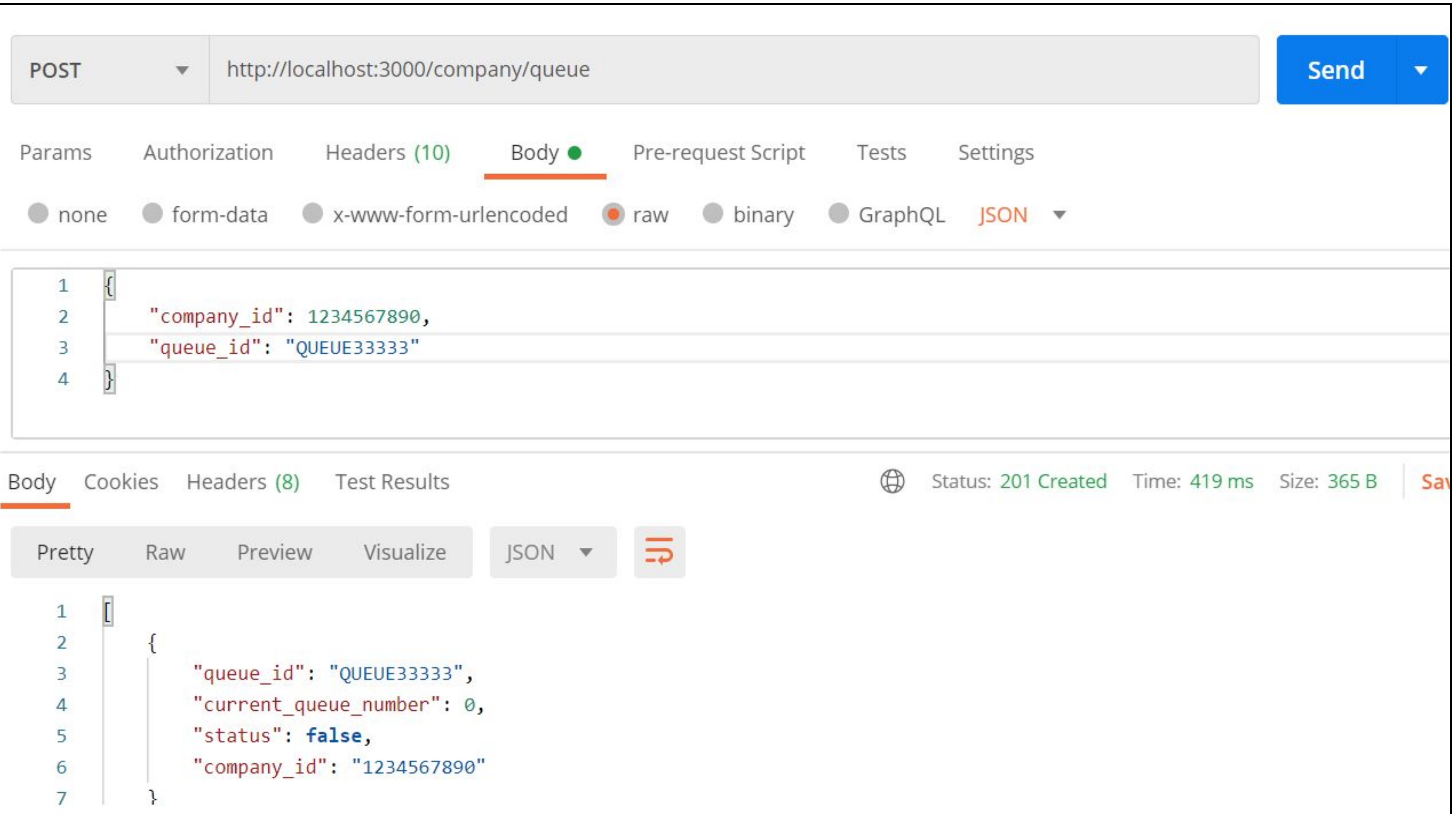
<table>
<tr><td><b>Title</b></td><td>Create Queue API - Successful request creates queue in database.</td></tr>
<tr><td><b>Description</b></td><td>A successful request made to the server should create a new entry in the queue_id with the correct parameters.</td></tr>
<tr><td><b>Precondition</b></td><td>Table should be created, queue_id (“QUEUE33333”) does not already exist.

<table>
<tr><th>queue_id</th><th>current_queue_number</th><th>status</th><th>company_id</th></tr>
<tr><td>0000000000</td><td>0</td><td>false</td><td>1234567890</td></tr>
<tr><td>ZZZZZZZZZZ</td><td>0</td><td>false</td><td>1234567890</td></tr>
<tr><td>QUEUE12341</td><td>0</td><td>false</td><td>1234567899</td></tr>
<tr><td>QUEUE12342</td><td>0</td><td>false</td><td>1234567890</td></tr>
<tr><td>QUEUE12343</td><td>0</td><td>false</td><td>1234567891</td></tr>
<tr><td>QUEUE12344</td><td>0</td><td>false</td><td>1234567892</td></tr>
<tr><td>QUEUE12346</td><td>0</td><td>false</td><td>1234567894</td></tr>
<tr><td>QUEUE12347</td><td>0</td><td>false</td><td>1234567895</td></tr>
<tr><td>QUEUE12348</td><td>0</td><td>false</td><td>1234567896</td></tr>
<tr><td>QUEUE12345</td><td>0</td><td>true</td><td>1234567890</td></tr>
<tr><td>QUEUE12340</td><td>0</td><td>false</td><td>1234567890</td></tr>
<tr><td>QUEUE22222</td><td>0</td><td>false</td><td>1234567890</td></tr>
</table></td></tr>
<tr><td><b>Test Steps</b></td><td>1. Start the Backend Server.
2. Send a Create Queue request to the backend by using Postman:

Method: POST

Path: http://localhost:3000/company/queue</td></tr>
</table>

	Body:
	{
	“company_id”: 1234567890,
	“queue_id”: “QUEUE33333”
	}
	3. Receive a 201 Created response from Postman.
	4. Go to elephantsql to inspect the queue_tab.

Expected Result	A new row with the queue_id (“QUEUE33333”) is created.
Evidence	201 Created response from Postman.



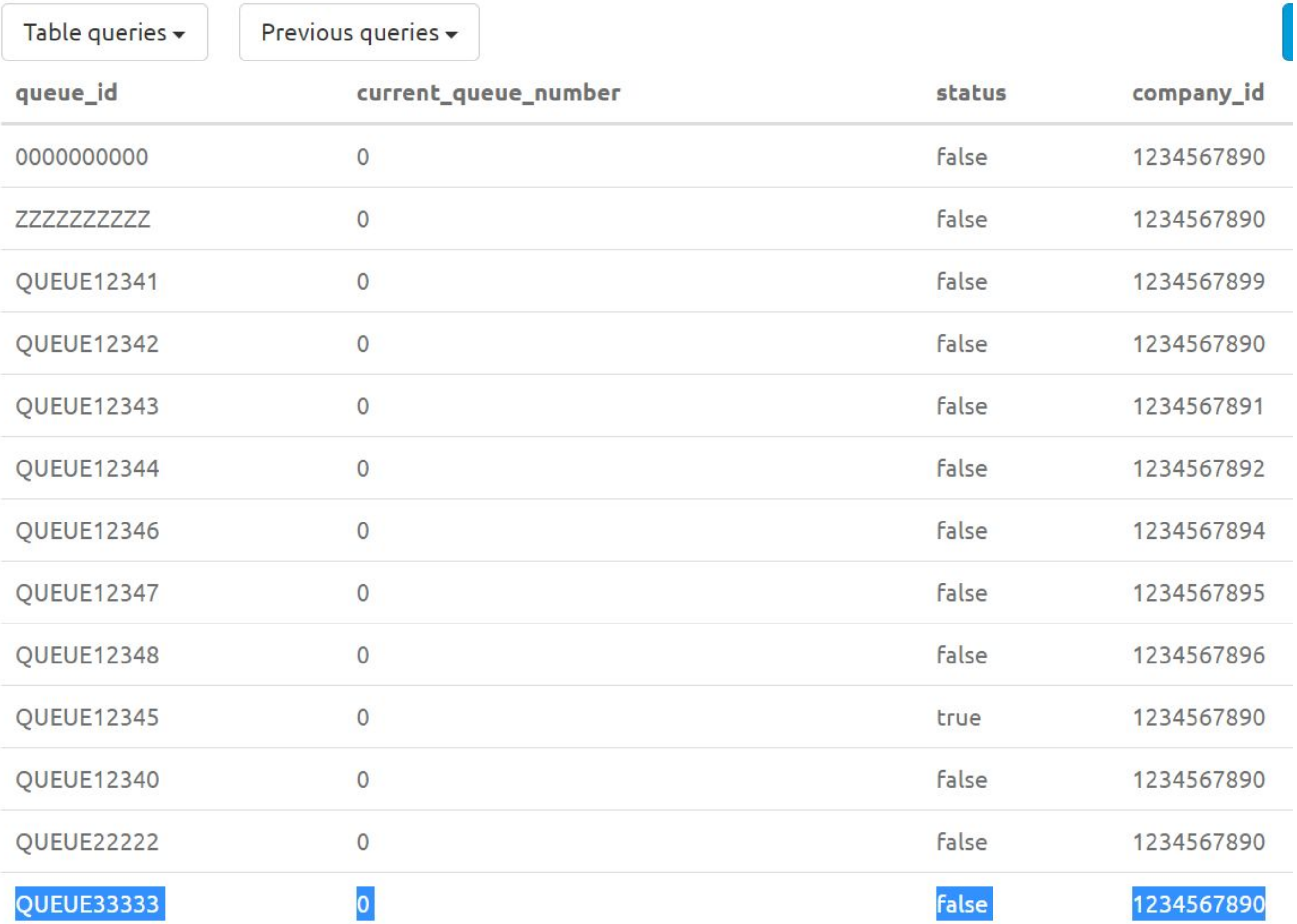
POST http://localhost:3000/company/queue

```
{
    "company_id": 1234567890,
    "queue_id": "QUEUE33333"
}
```

Status: 201 Created Time: 419 ms Size: 365 B

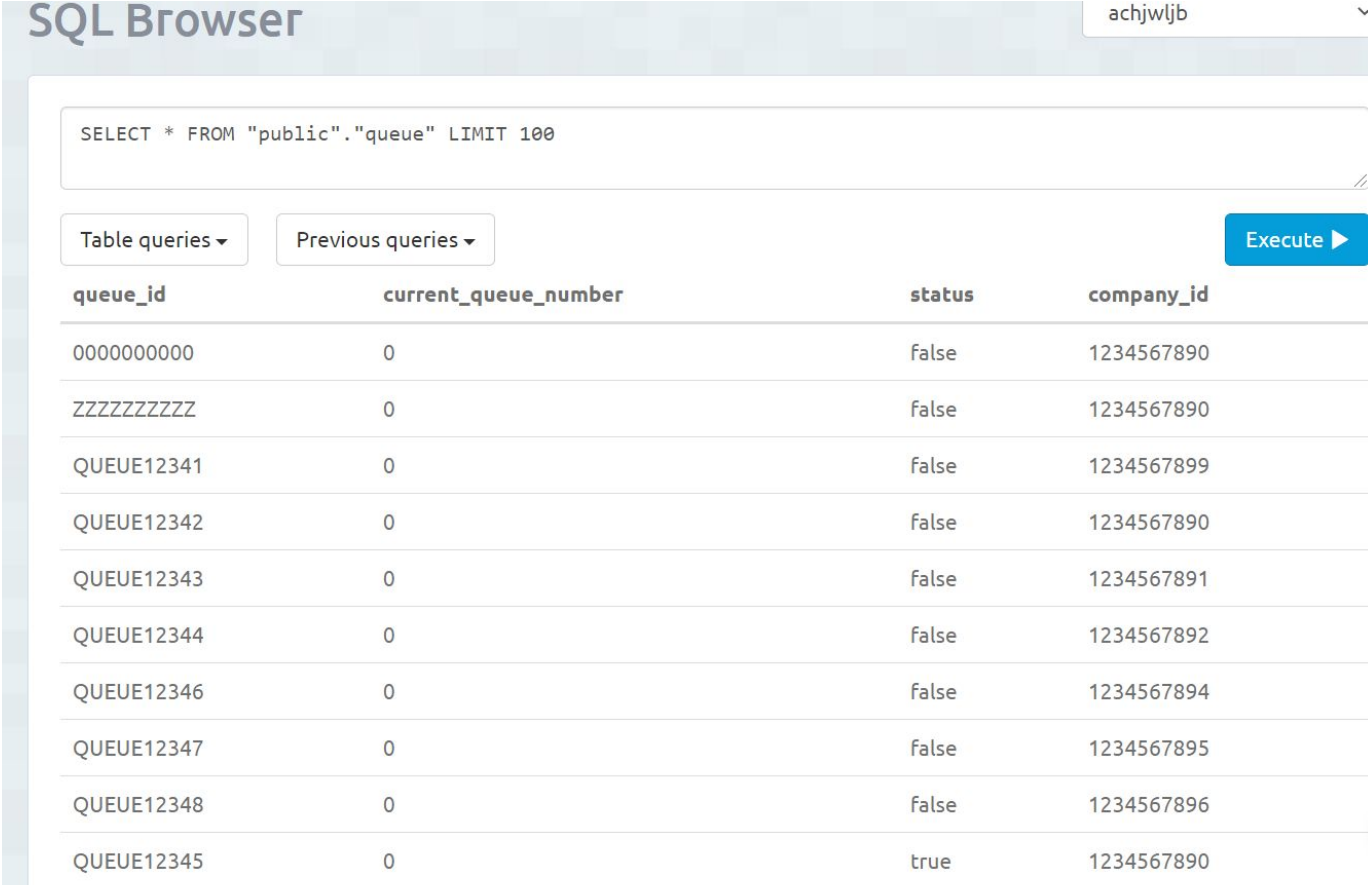
```
[
    {
        "queue_id": "QUEUE33333",
        "current_queue_number": 0,
        "status": false,
        "company_id": "1234567890"
    }
```

queue_id (“QUEUE33333”) created in table Queue on Elephant SQL.



queue_id	current_queue_number	status	company_id
0000000000	0	false	1234567890
ZZZZZZZZZZ	0	false	1234567890
QUEUE12341	0	false	1234567899
QUEUE12342	0	false	1234567890
QUEUE12343	0	false	1234567891
QUEUE12344	0	false	1234567892
QUEUE12346	0	false	1234567894
QUEUE12347	0	false	1234567895
QUEUE12348	0	false	1234567896
QUEUE12345	0	true	1234567890
QUEUE12340	0	false	1234567890
QUEUE22222	0	false	1234567890
QUEUE33333	0	false	1234567890

1.2 Successful Request - Not Case Sensitive

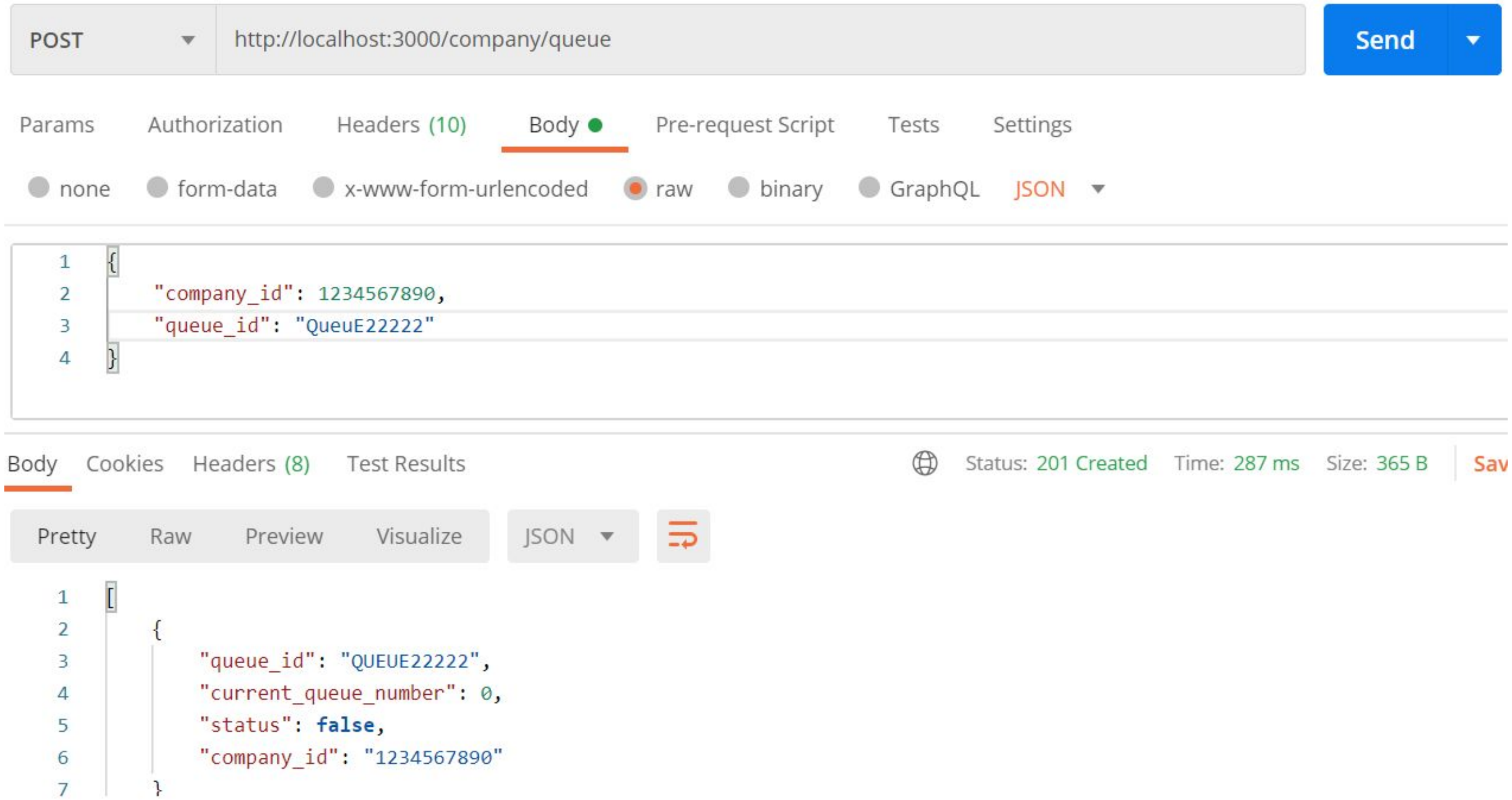
Title	Create Queue API - Not Case Sensitive - Successful request creates queue in database.
Description	A successful request made to the server should create a new entry in the queue_id with the correct parameters.
Precondition	Table should be created, queue_id (“QUEUE22222”) does not already exist. 

```
SELECT * FROM "public"."queue" LIMIT 100
```

queue_id	current_queue_number	status	company_id
0000000000	0	false	1234567890
ZZZZZZZZZZ	0	false	1234567890
QUEUE12341	0	false	1234567899
QUEUE12342	0	false	1234567890
QUEUE12343	0	false	1234567891
QUEUE12344	0	false	1234567892
QUEUE12346	0	false	1234567894
QUEUE12347	0	false	1234567895
QUEUE12348	0	false	1234567896
QUEUE12345	0	true	1234567890

Test Steps	1. Start the Backend Server.
	2. Send a Create Queue request to the backend by using Postman:
	Method: POST
	Path: http://localhost:3000/company/queue
	Body:
	{
	“company_id”: 1234567890,
	“queue_id”: “QueuE22222”
	}
	3. Receive a 201 Created response from Postman.
	4. Go to elephantsql to inspect the queue_tab.

Expected Result	A new row with the queue_id (“QUEUE22222”) is created.
Evidence	201 Created response from Postman.



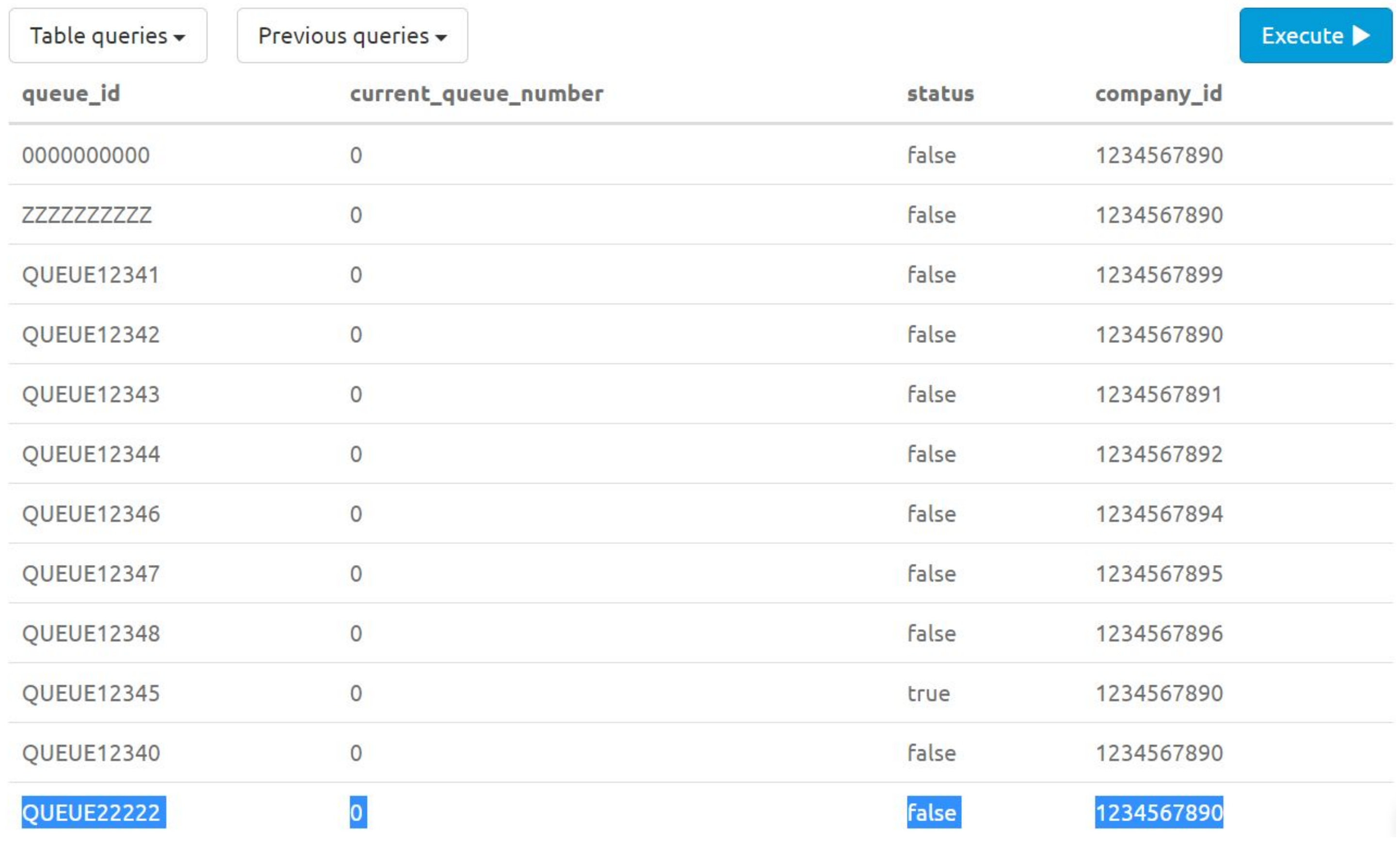
POST http://localhost:3000/company/queue

```
{
    "company_id": 1234567890,
    "queue_id": "QueuE22222"
}
```

Status: 201 Created Time: 287 ms Size: 365 B

```
[
    {
        "queue_id": "QUEUE22222",
        "current_queue_number": 0,
        "status": false,
        "company_id": "1234567890"
    }
```

queue_id (QUEUE22222) created in table Queue on Elephant SQL.



queue_id	current_queue_number	status	company_id
0000000000	0	false	1234567890
ZZZZZZZZZZ	0	false	1234567890
QUEUE12341	0	false	1234567899
QUEUE12342	0	false	1234567890
QUEUE12343	0	false	1234567891
QUEUE12344	0	false	1234567892
QUEUE12346	0	false	1234567894
QUEUE12347	0	false	1234567895
QUEUE12348	0	false	1234567896
QUEUE12345	0	true	1234567890
QUEUE12340	0	false	1234567890
QUEUE22222	0	false	1234567890

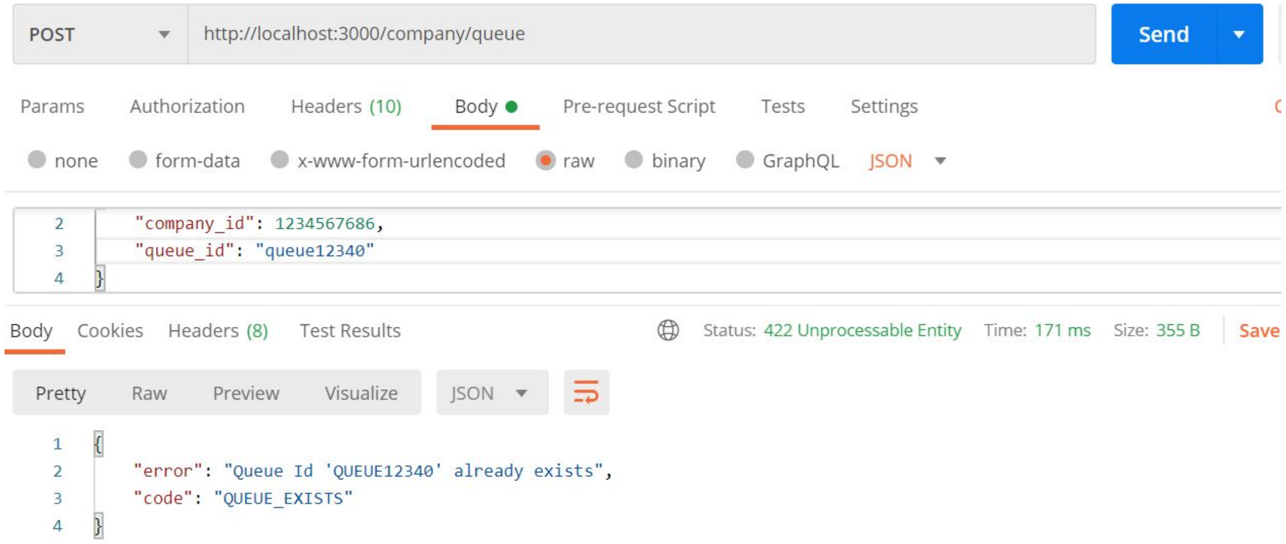
1.3 Queue Exists

Title	Create Queue API - Queue Exists - Unsuccessful request will not create Queue in Database.
Description	An unsuccessful request made to the server due to a queue already existing should return an error that indicates queue id already exists.
Precondition	Table should be created, queue_id (“QUEUE12340”) already exists.

Test Steps	1. Start the Backend Server.
	2. Send a Create Queue request to the backend by using Postman:
	Method: POST
	Path: http://localhost:3000/company/queue
	Body:
	{
	“company_id”: 1234567686,
	“queue_id”: “queue12340”
	}
	3. Receive a 422 Unprocessable Entity response from Postman that has an error message indicating queue id already exists.

Expected Result	Receive a 422 Unprocessable Entity response from Postman that has an error message with error code that indicates that queue id(“QUEUE12340”) already exists in the table.

Evidence



POST http://localhost:3000/company/queue

```
    "company_id": 1234567686,
    "queue_id": "queue12340"
}
```

Status: 422 Unprocessable Entity Time: 171 ms Size: 355 B

JSON

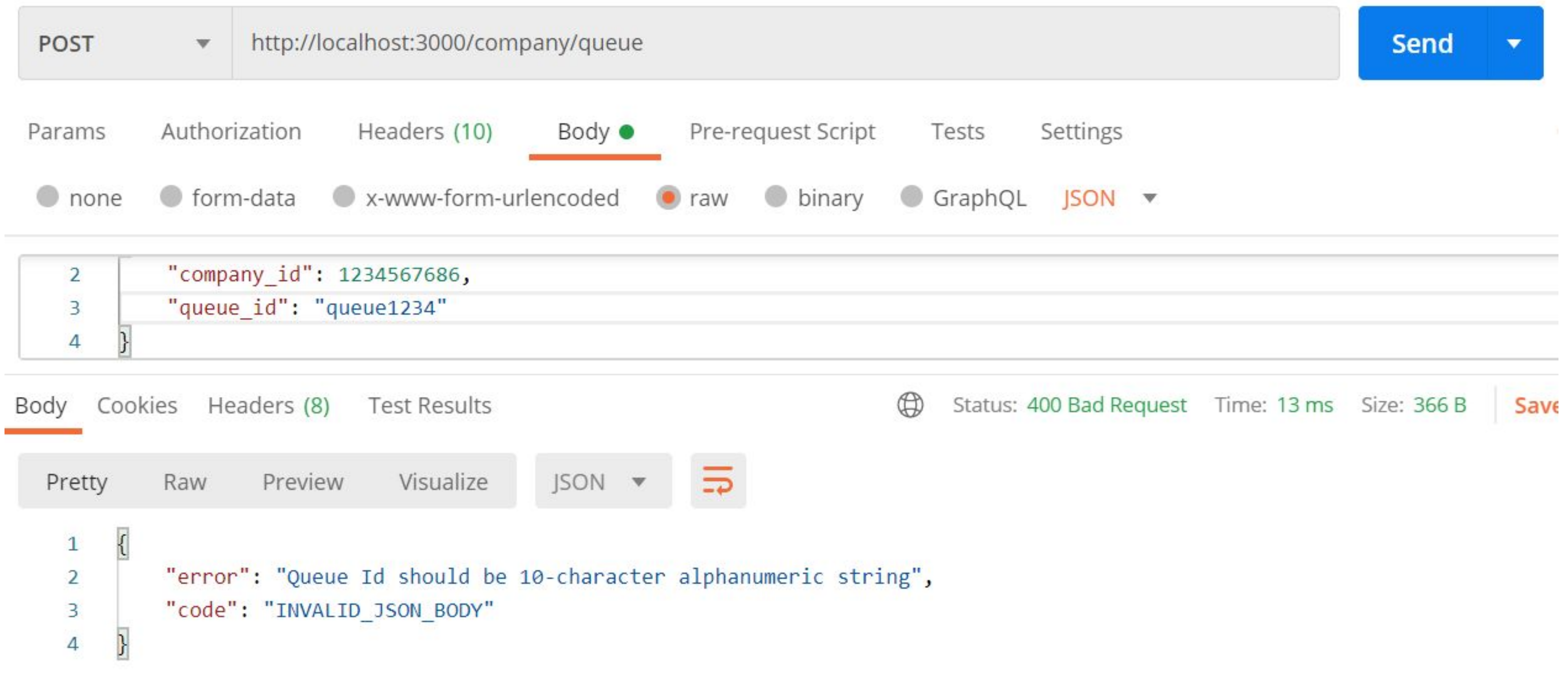
```
{
    "error": "Queue Id 'QUEUE12340' already exists",
    "code": "QUEUE_EXISTS"
}
```

1.4 Queue Id Validation Failed

Title	Create Queue API - Queue Id Validation Failed - Unsuccessful request will not create Queue in Database.
Description	An unsuccessful request made to the server due to invalid queue_id should return an error that indicates the queue_id is invalid.
Precondition	Table should be created.

Test Steps	1. Start the Backend Server.
	2. Send a Create Queue request to the backend by using Postman:
	Method: POST
	Path: http://localhost:3000/company/queue
	Body:
	{
	“company_id”: 1234567686,
	“queue_id”: “queue1234”
	}
	3. Receive a 400 Bad Request response from Postman.

Expected Result	Receive a 400 Bad Request response from Postman that has an error message with a code that indicates the queue_id is invalid.

Evidence	

http://localhost:3000/company/queue

```
    "company_id": 1234567686,
    "queue_id": "queue1234"
}
```

Status: 400 Bad Request Time: 13 ms Size: 366 B

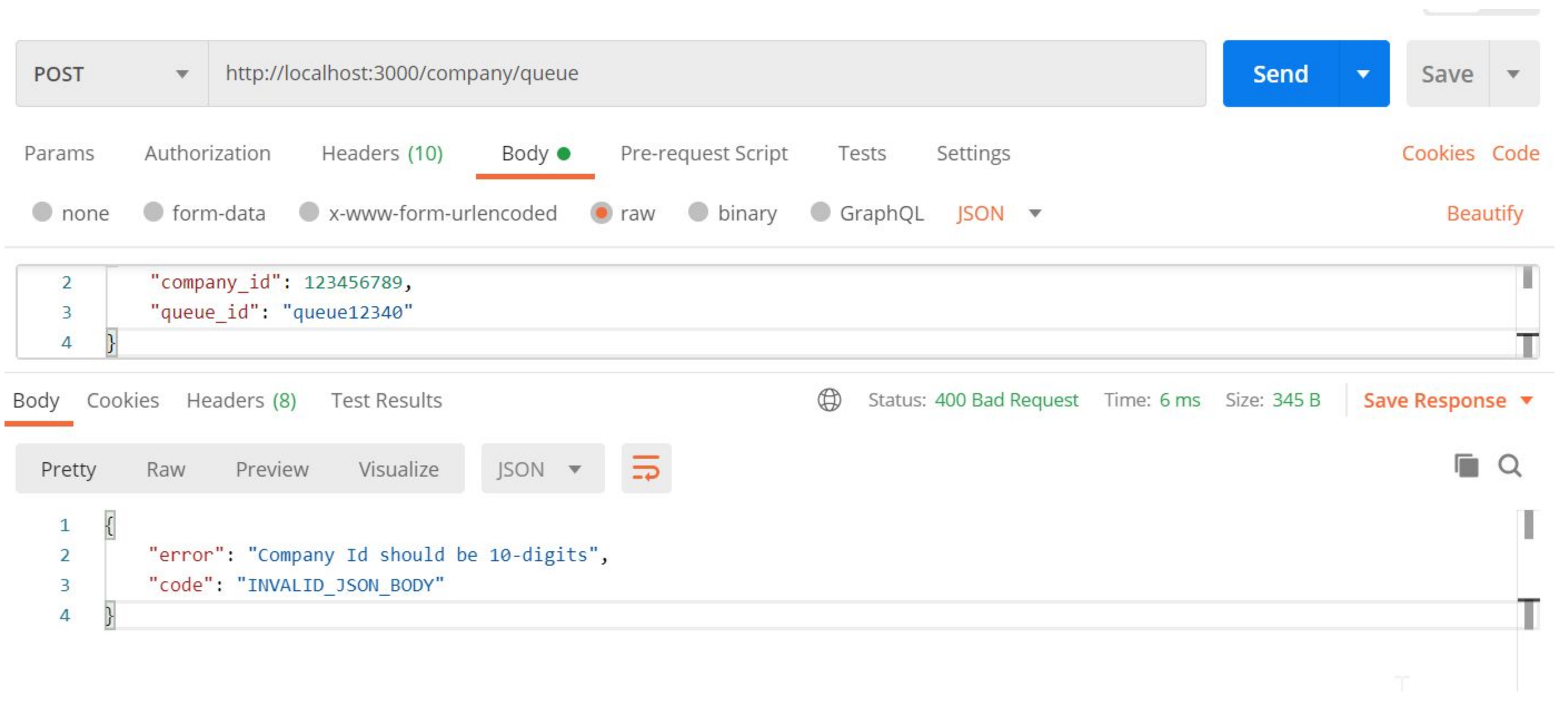
```
{
    "error": "Queue Id should be 10-character alphanumeric string",
    "code": "INVALID_JSON_BODY"
}
```

1.5 Company Id Validation Failed

Title	Create Queue API - Company Id Validation Failed - Unsuccessful request will not create Queue in Database.
Description	An unsuccessful request made to the server due to invalid company_id should return an error that indicates the company_id is invalid.
Precondition	Table should be created.

Test Steps	1. Start the Backend Server.
	2. Send a Create Queue request to the backend by using Postman:
	Method: POST
	Path: http://localhost:3000/company/queue
	Body:
	{
	“company_id”: 123456789,
	“queue_id”: “queue12340”
	}
	3. Receive a 400 Bad Request response from Postman.

Expected Result	Receive a 400 Bad Request response from Postman that has an error message with a code that indicates the company_id is invalid.

Evidence	

http://localhost:3000/company/queue

```
    "company_id": 123456789,
    "queue_id": "queue12340"
}
```

Status: 400 Bad Request Time: 6 ms Size: 345 B

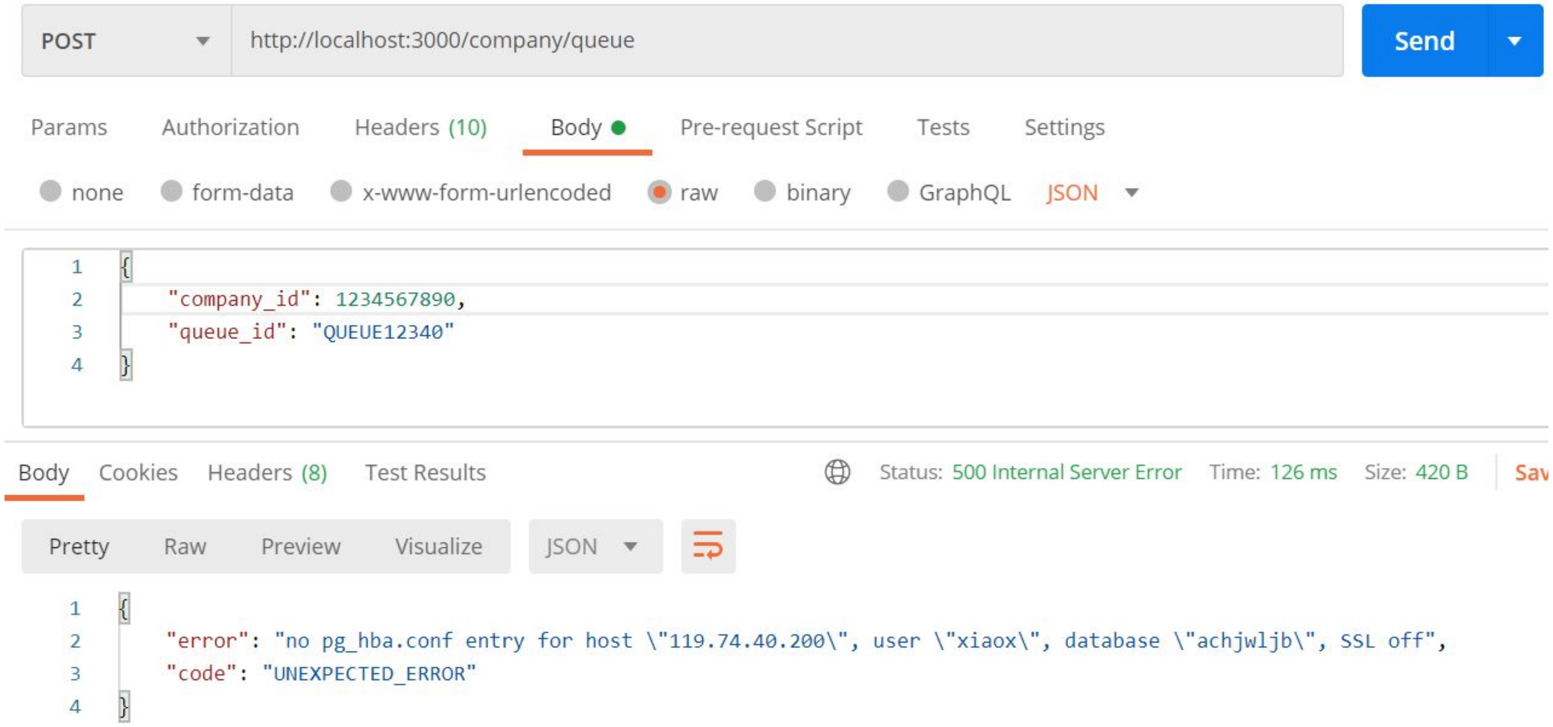
```
{
    "error": "Company Id should be 10-digits",
    "code": "INVALID_JSON_BODY"
}
```

1.6 Unexpected Server Error

Title	Create Queue API - Unexpected Server Error - Unsuccessful request will not create Queue in Database.
Description	An unsuccessful request made to the server due to server error should return an error indicating what happened.
Precondition	Tables should be created, wrong database credentials in database.js.

Test Steps	1. Start the Backend Server.
	2. Send a Create Queue request to the backend by using Postman:
	Method: POST
	Path: http://localhost:3000/company/queue
	Body:
	{
	“company_id”: 1234567890,
	“queue_id”: “queue12340”
	}
	3. Receive a 500 Internal Server response from Postman.

Expected Result	Receive a 500 Internal Server response from Postman that has an error message with a code that indicates what happened.

Evidence	

POST http://localhost:3000/company/queue

```
{
    "company_id": 1234567890,
    "queue_id": "QUEUE12340"
}
```

Status: 500 Internal Server Error Time: 126 ms Size: 420 B

```
{
    "error": "no pg_hba.conf entry for host \"119.74.40.200\", user \"xiaox\", database \"achjwljb\", SSL off",
    "code": "UNEXPECTED_ERROR"
}
```

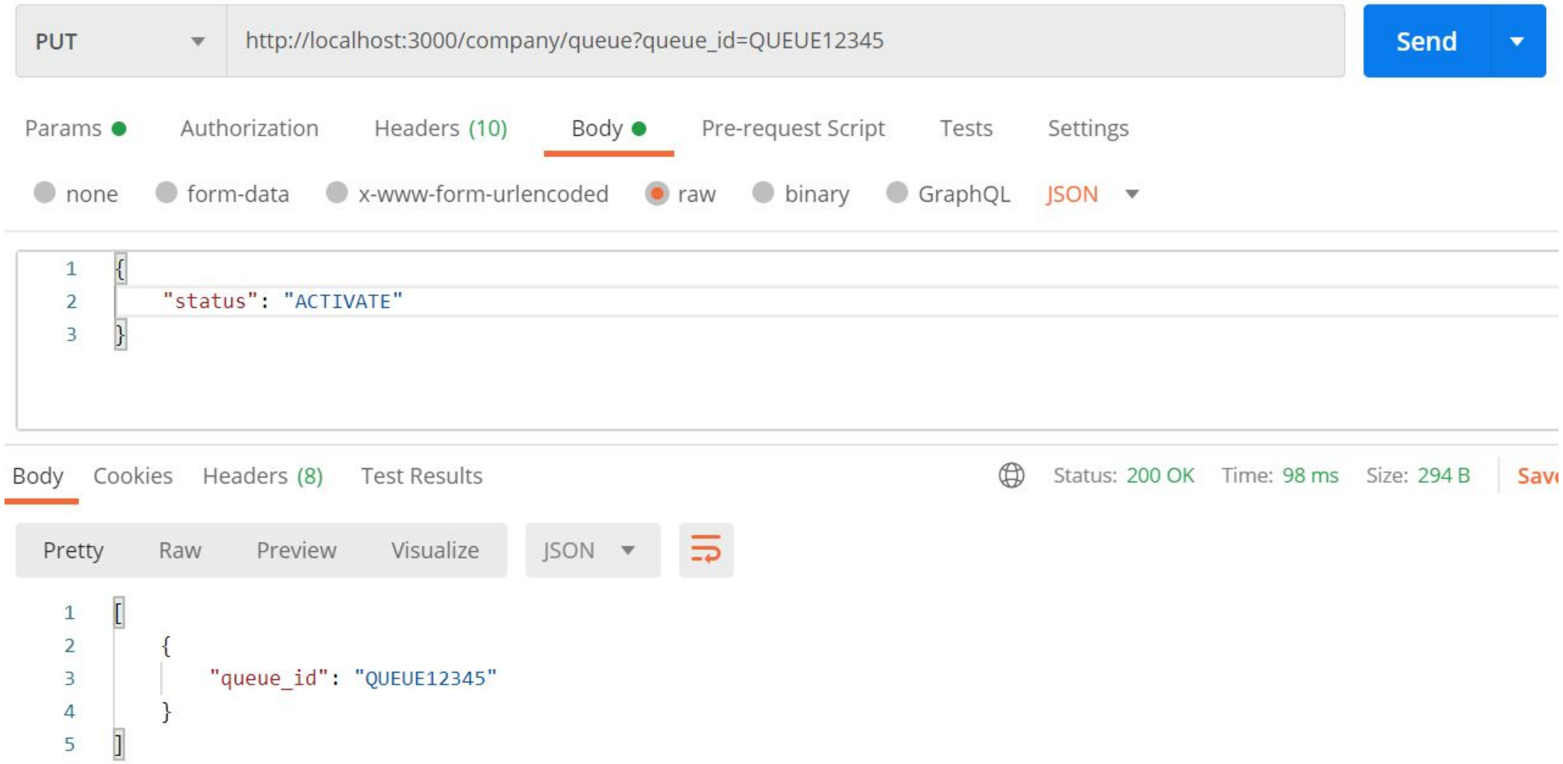
2. Update Queue API

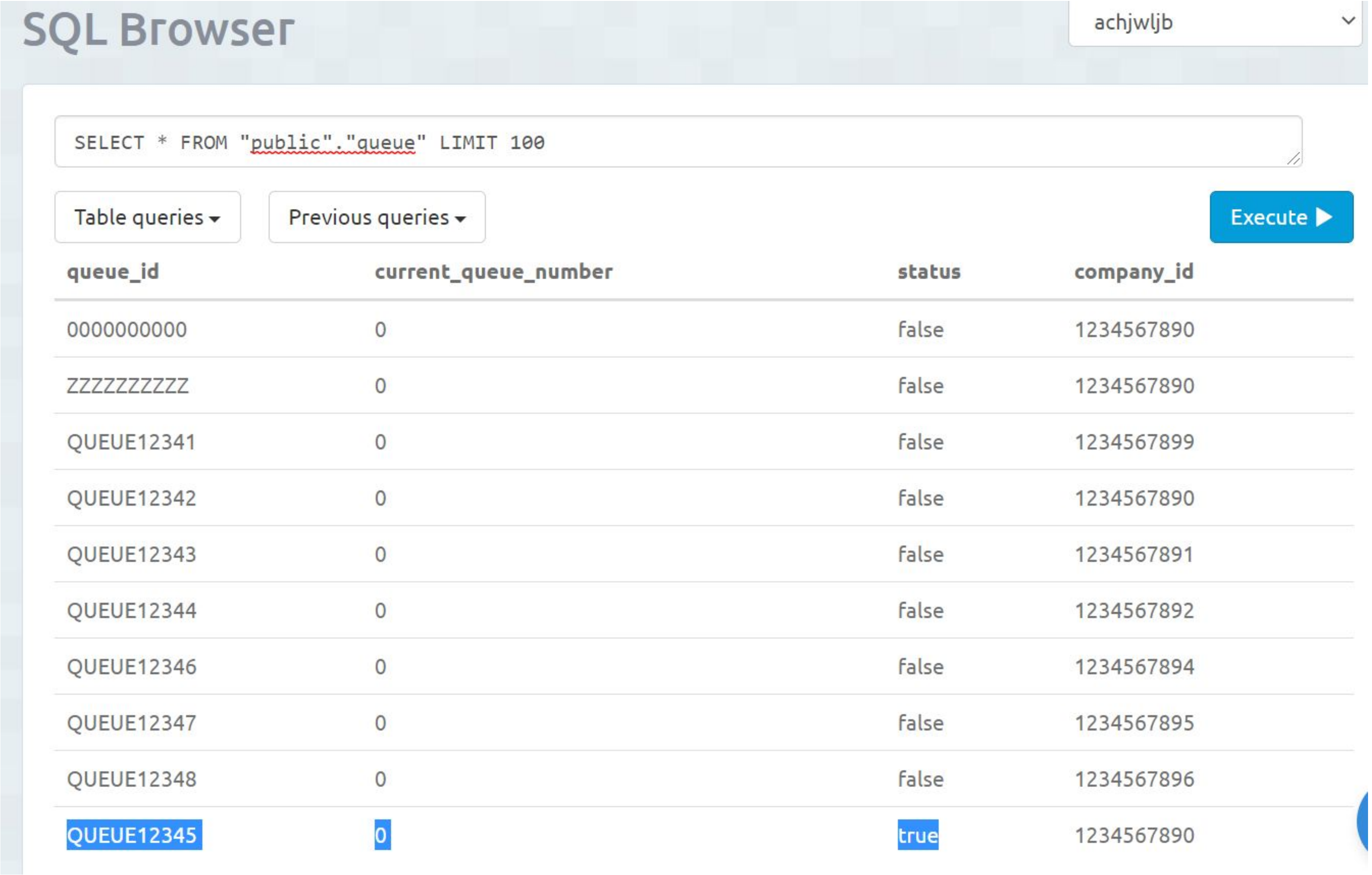
2.1 Successful request

Title	Update Queue API - Successful request update information of a queue.
Description	A successful request made to the server should update information of a queue with the correct parameters.
Precondition	Table should be created, queue_id (QUEUE12345) already exists with default status ‘false’.

Test Steps	1. Start the Backend Server
	2. Send an Update Queue request to the backend by using Postman:
	Method: PUT
	Path: http://localhost:3000/company/queue?queue_id=QUEUE12345
	Body:
	{
	“status”: “ACTIVATE”
	}
	3. Receive a 200 OK response from Postman.
	4. Go to elephantsql to inspect the queue_tab.

Expected Result	Status of queue_id (“QUEUE12345”) should be updated to ‘true’.

Evidence	200 OK responses from Postman.
	
	 Send
	Params ● Authorization Headers (10) Body ● Pre-request Script Tests Settings
	○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON
	Body Cookies Headers (8) Test Results
	Pretty Raw Preview Visualize JSON
	PUT http://localhost:3000/company/queue?queue_id=QUEUE12345
	`{`
	`    "status": "ACTIVATE"`
	`}`
	Status: 200 OK Time: 98 ms Size: 294 B Sav
	`[`
	`    {`
	`        "queue_id": "QUEUE12345"`
	`    }`
	`]`
	Status of queue_id (“QUEUE12345”) updated to ‘true’ in table Queue on Elephant SQL.



```
SELECT * FROM "public"."queue" LIMIT 100
```

queue_id	current_queue_number	status	company_id
0000000000	0	false	1234567890
ZZZZZZZZZZ	0	false	1234567890
QUEUE12341	0	false	1234567899
QUEUE12342	0	false	1234567890
QUEUE12343	0	false	1234567891
QUEUE12344	0	false	1234567892
QUEUE12346	0	false	1234567894
QUEUE12347	0	false	1234567895
QUEUE12348	0	false	1234567896
QUEUE12345	0	true	1234567890

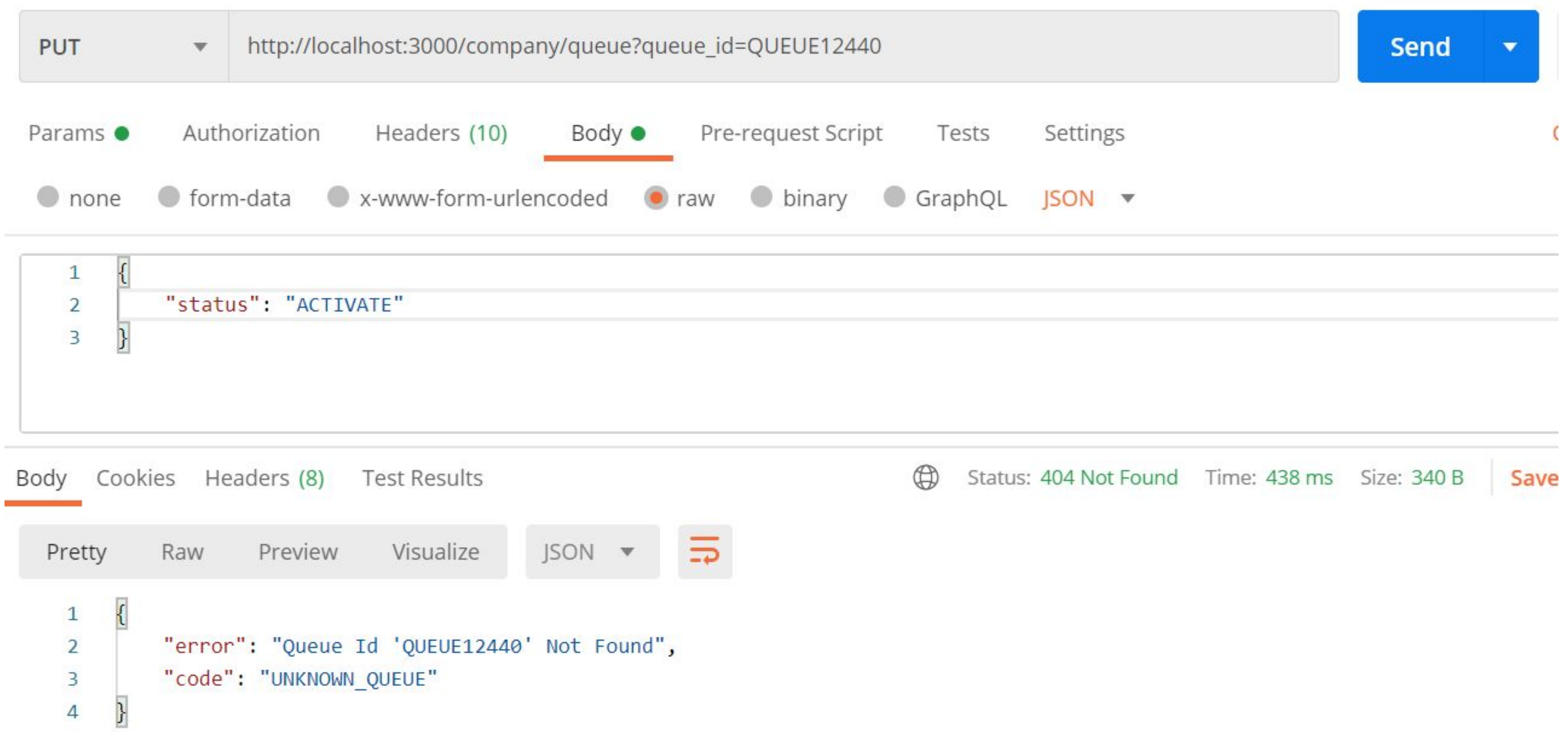
2.2 Non-existence Queue Id

Title	Update Queue API - Non-existence Queue Id.
Description	An unsuccessful request made to the server due to non-existence queue_id should return an error that indicates that the queue_id does not exist.
Precondition	Table should be created, queue_id ("QUEUE12440") does not already exist.

Test Steps	1. Start the Backend Server.
	2. Send an Update Queue request to the backend by using Postman:
	Method: PUT
	Path: http://localhost:3000/company/queue?queue_id=QUEUE12440
	Body:
	{
	"status": "ACTIVATE"
	}
	3. Receive a 404 Not Found response from Postman.

Expected Result	Receive a 404 Not Found response from Postman that has an error message with error code that indicates the queue_id ("QUEUE12440") not found in the table.

Evidence



PUT http://localhost:3000/company/queue?queue_id=QUEUE12440

Body: ● raw JSON

```
{
    "status": "ACTIVATE"
}
```

Status: 404 Not Found Time: 438 ms Size: 340 B

```
{
    "error": "Queue Id 'QUEUE12440' Not Found",
    "code": "UNKNOWN_QUEUE"
}
```

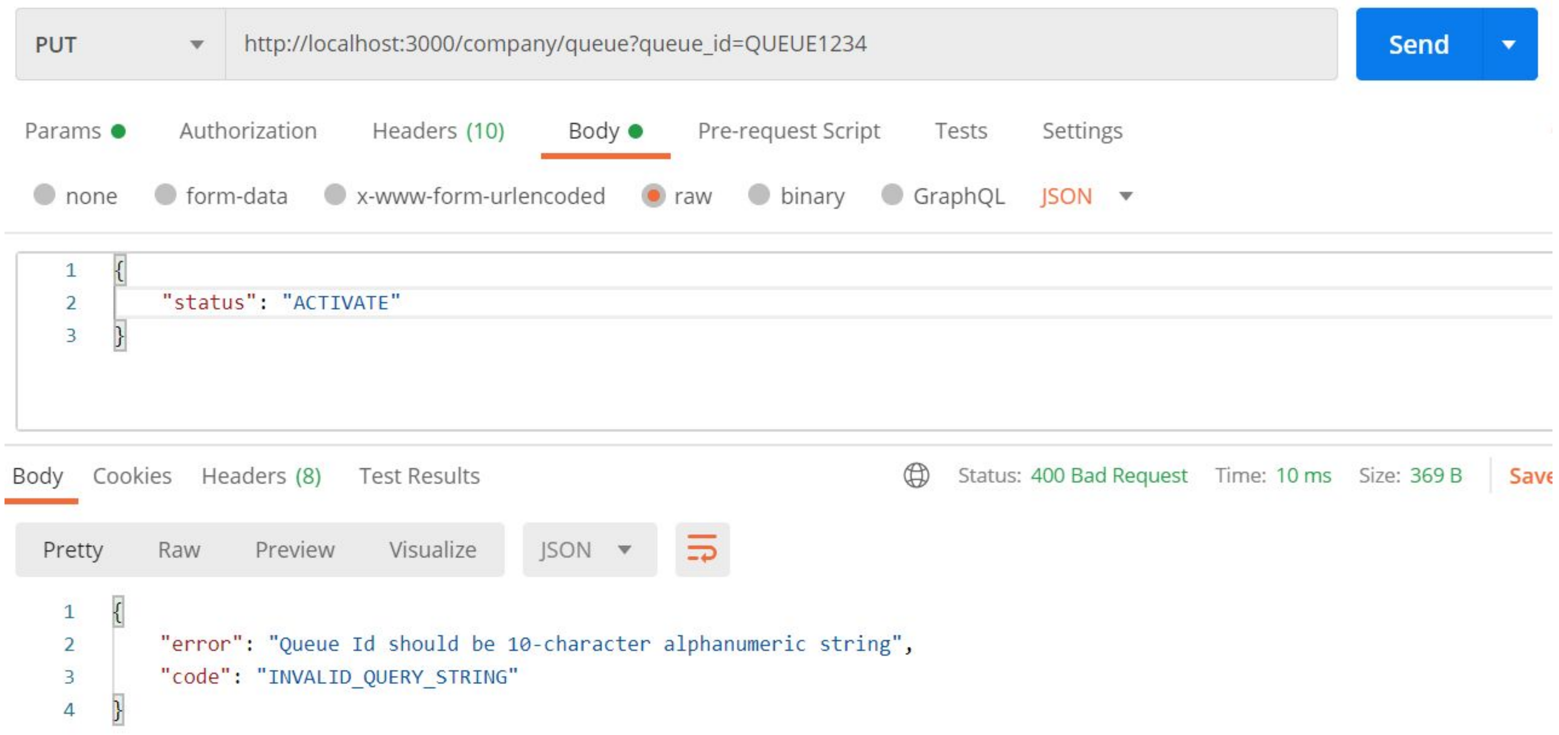
2.3 Queue Id Validation Failed

Title	Update Queue API - Queue Id Validation Failed - Unsuccessful request will not update the table Queue.
Description	An unsuccessful request made to the server due to invalid queue_id should return an error that indicates the queue_id is invalid.
Precondition	Table should be created.

Test Steps	1. Start the Backend Server.
	2. Send an Update Queue request to the backend by using Postman:
	Method: PUT
	Path: http://localhost:3000/company/queue?queue_id=QUEUE1234
	Body:
	{
	“status”: “ACTIVATE”
	}
	3. Receive a 400 Bad Request response from Postman that has an error message indicating the queue_id is invalid.

Expected Result	Receive a 400 Bad Request response from Postman that has an error message with a code that indicates the queue_id is invalid.

Evidence



PUT http://localhost:3000/company/queue?queue_id=QUEUE1234

○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
    "status": "ACTIVATE"
}
```

Status: 400 Bad Request Time: 10 ms Size: 369 B

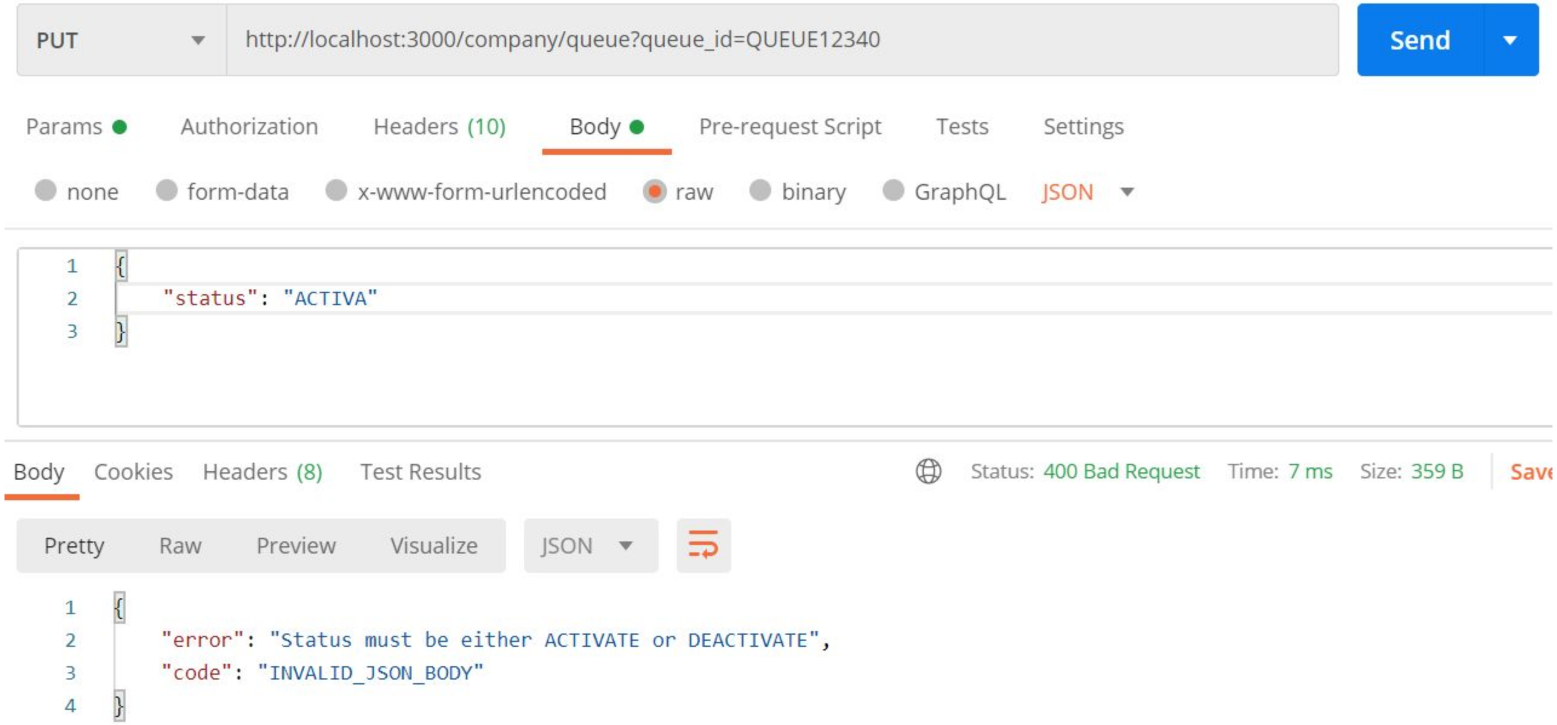
```
{
    "error": "Queue Id should be 10-character alphanumeric string",
    "code": "INVALID_QUERY_STRING"
}
```

2.4 Status Validation Failed

Title	Update Queue API - Status Validation Failed - Unsuccessful request will not update status of a queue.
Description	An unsuccessful request made to the server due to invalid status should return an error that indicates the status is invalid.
Precondition	Table should be created, queue_id (“QUEUE12340”) already exists in the table Queue.

Test Steps	1. Start the Backend Server.
	2. Send an Update Queue request to the backend by using Postman:
	Method: PUT
	Path: http://localhost:3000/company/queue?queue_id=QUEUE12340
	Body:
	{
	“status”: “ACTIVA”
	}
	3. Receive a 400 Bad Request response from Postman that has an error message indicating the status is invalid.

Expected Result	Receive a 400 Bad Request response from Postman that has an error message with a code that indicates the status is invalid.

Evidence	

PUT http://localhost:3000/company/queue?queue_id=QUEUE12340

```
{
    "status": "ACTIVA"
}
```

Status: 400 Bad Request Time: 7 ms Size: 359 B

```
{
    "error": "Status must be either ACTIVATE or DEACTIVATE",
    "code": "INVALID_JSON_BODY"
}
```

2.5 Unexpected Server Error

Title	Update Queue API - Unexpected Server Error - Unsuccessful request will not update Queue in Database.
Description	An unsuccessful request made to the server due to server error should return an error indicating what happened.
Precondition	Tables should be created, wrong database credentials in database.js.

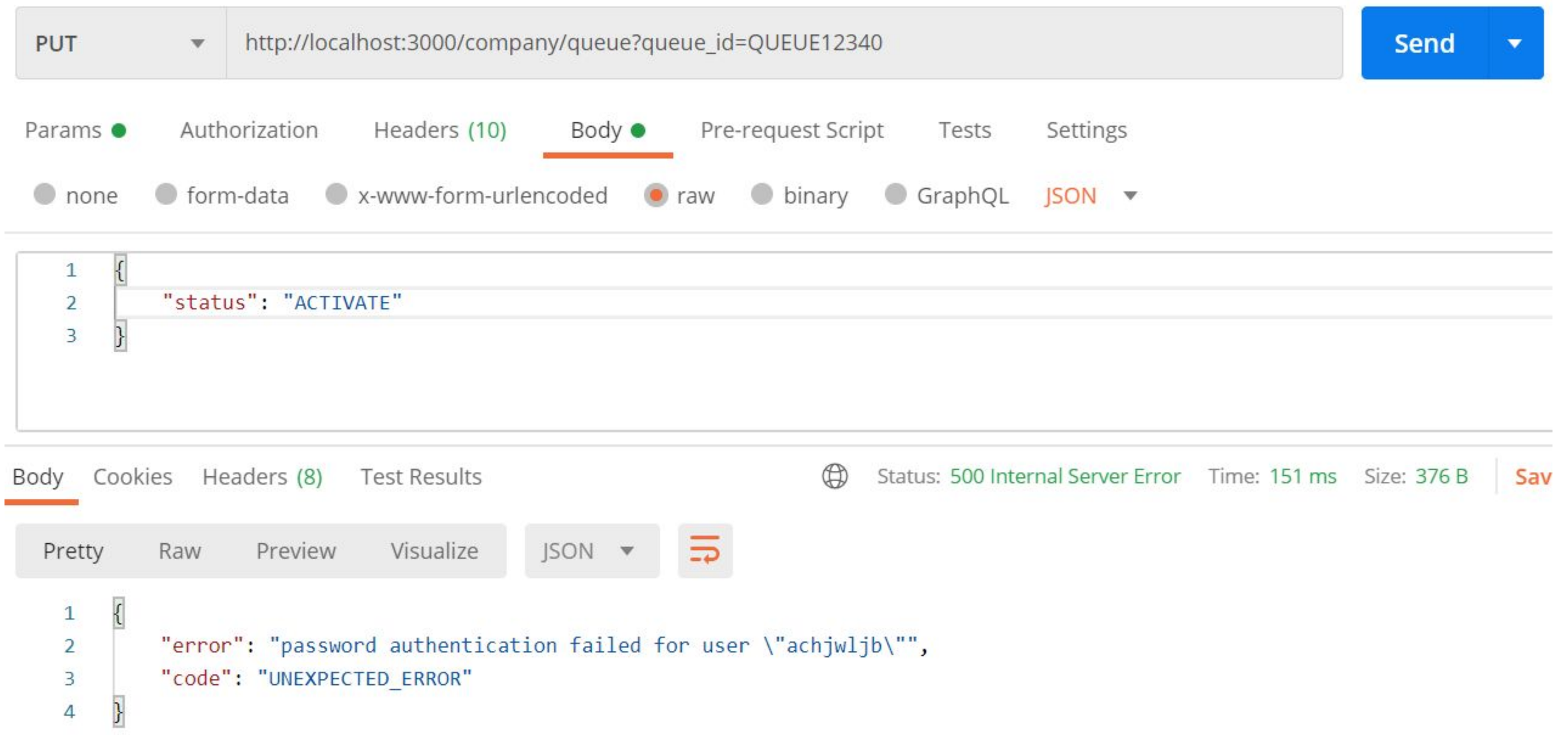
Test Steps	1. Start the Backend Server.
	2. Send an Update Queue request to the backend by using Postman:
	Method: PUT
	Path: http://localhost:3000/company/queue?queue_id=QUEUE12340
	Body:
	{
	“status”: “ACTIVATE”
	}
	3. Receive a 500 Internal Server response from Postman.

Expected Result	Receive a 500 Internal Server response from Postman that has an error message with a code that indicates what happened.

<table>
<tr>
<td>

Evidence

</td>
<td>



PUT http://localhost:3000/company/queue?queue_id=QUEUE12340

○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
    "status": "ACTIVATE"
}
```

Status: 500 Internal Server Error Time: 151 ms Size: 376 B

```
{
    "error": "password authentication failed for user \"achjwljb\"",
    "code": "UNEXPECTED_ERROR"
}
```

</td>
</tr>
</table>

3. Server Available API

3.1 Successful request

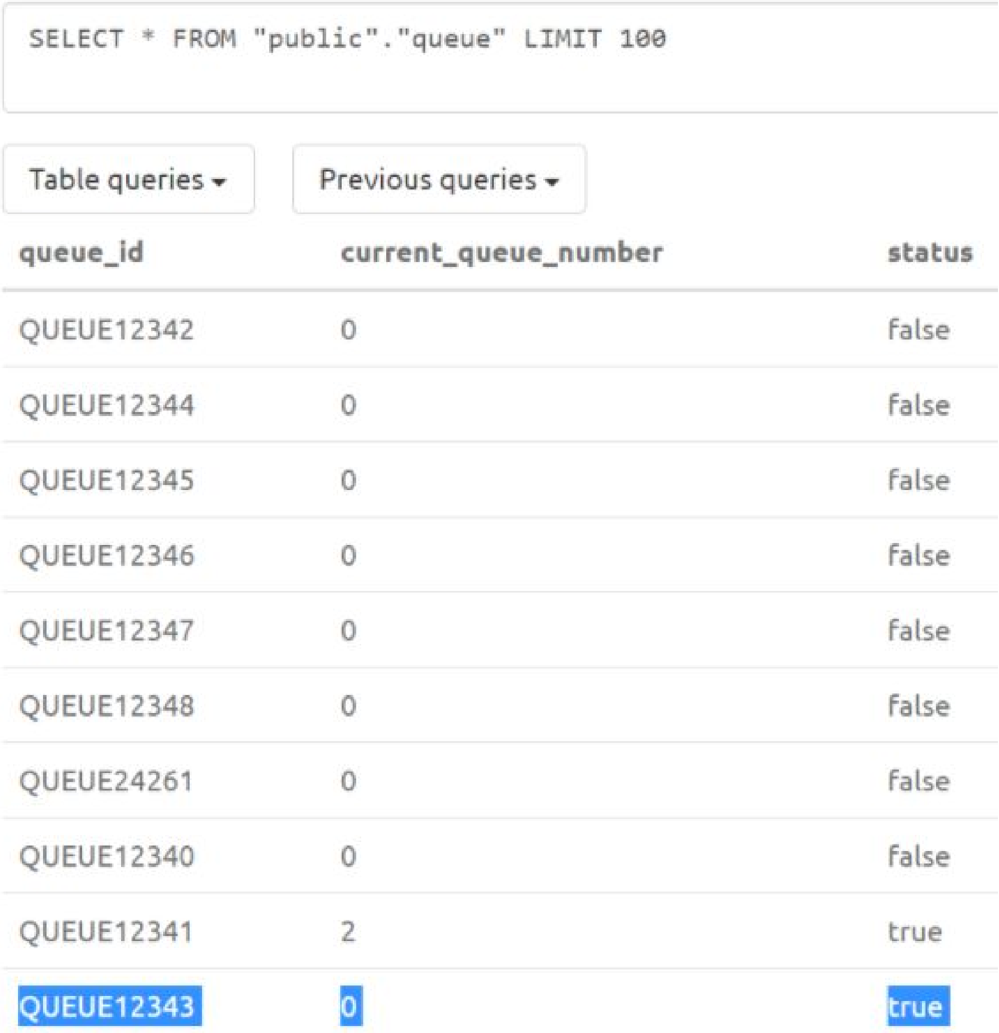
Title	Server Available API - Successful request.
Description	A successful request made to the server should allow the company to indicate that they have a server that is available to serve the next customer in the queue if any.
Precondition	Tables should be created with meaningful records(Queue table and CustomerQueueNumber table).

Test Steps	1. Start the Backend Server.
	2. Send an Server Available request to the backend by using Postman:
	Method: PUT
	Path: http://localhost:3000/company/server
	Body:
	{
	“queue_id”: “QUEUE12343”
	}
	3. Receive a 200 OK response from Postman.
	4. Go to elephantsql to check the Queue and CustomerQueueNumber table.

Expected Result	Show the customer_id of the next customer in the queue to the server if any.

Evidence

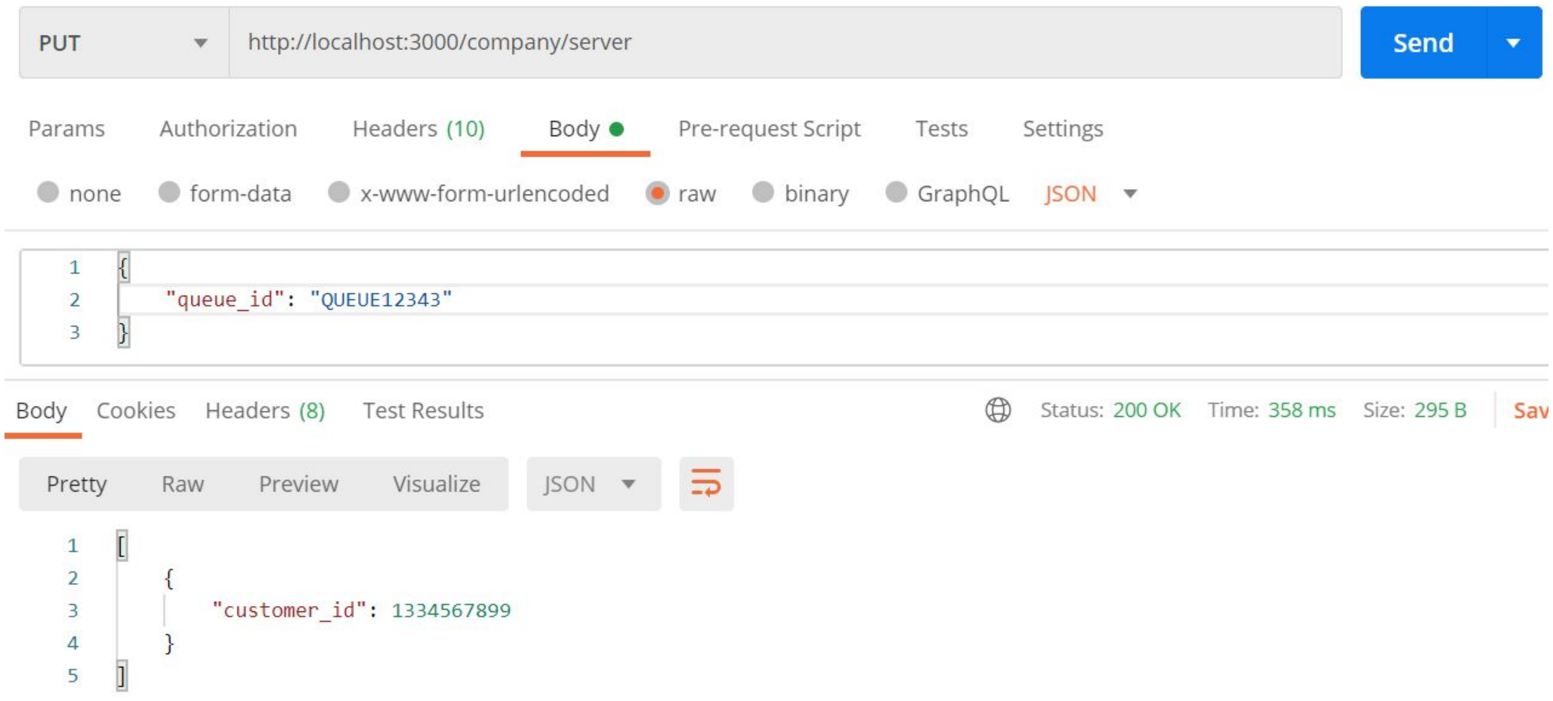
Initially, the current queue number of QUEUE12343 is 0.

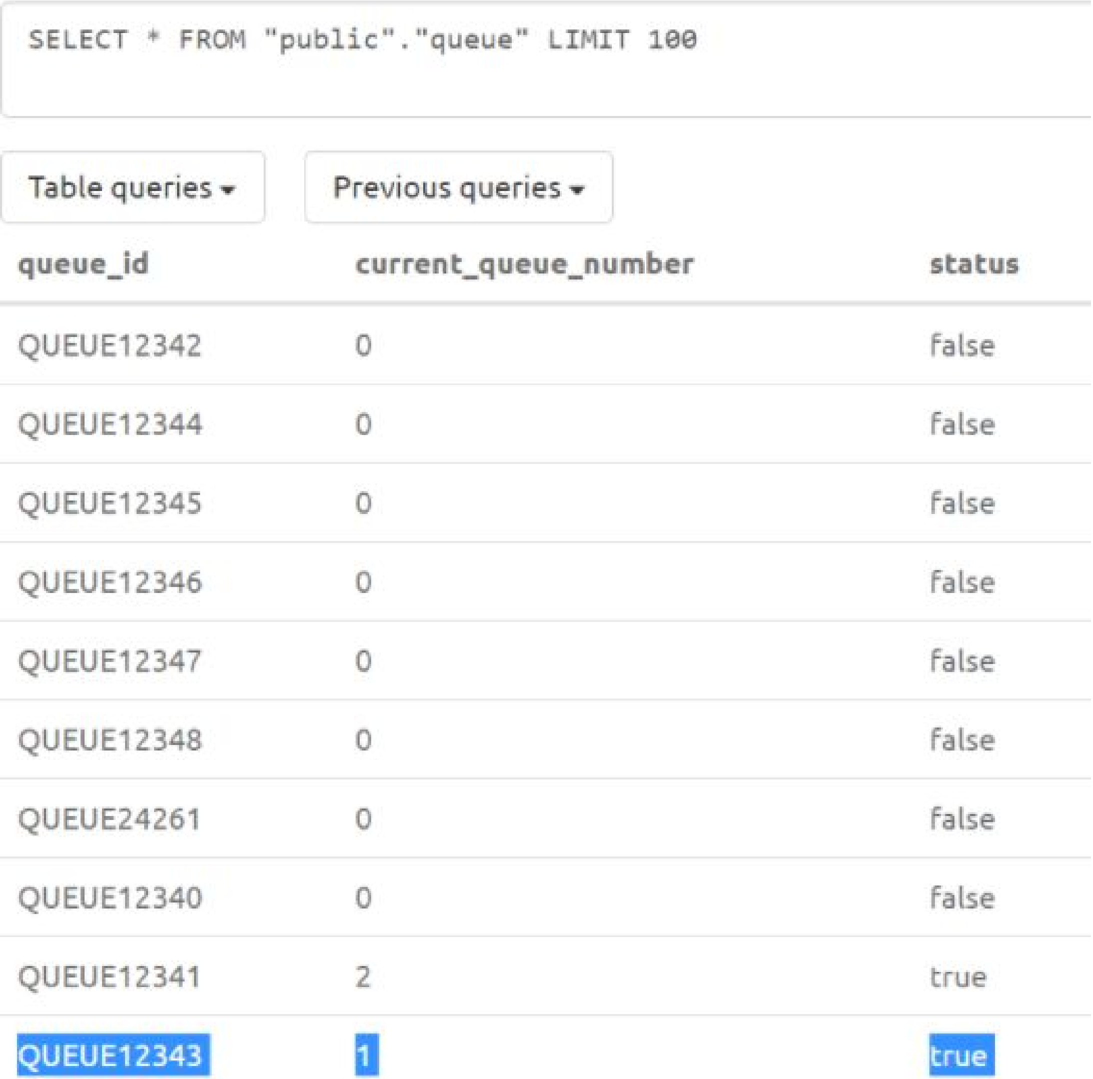


```
SELECT * FROM "public"."queue" LIMIT 100
```

queue_id	current_queue_number	status
QUEUE12342	0	false
QUEUE12344	0	false
QUEUE12345	0	false
QUEUE12346	0	false
QUEUE12347	0	false
QUEUE12348	0	false
QUEUE24261	0	false
QUEUE12340	0	false
QUEUE12341	2	true
QUEUE12343	0	true

After sending a Server Available request to the backend with a valid queue_id ("QUEUE12343") as body by using Postman, will then receive 200 OK responses from Postman, it shows the customer_id of the next customer to be served.

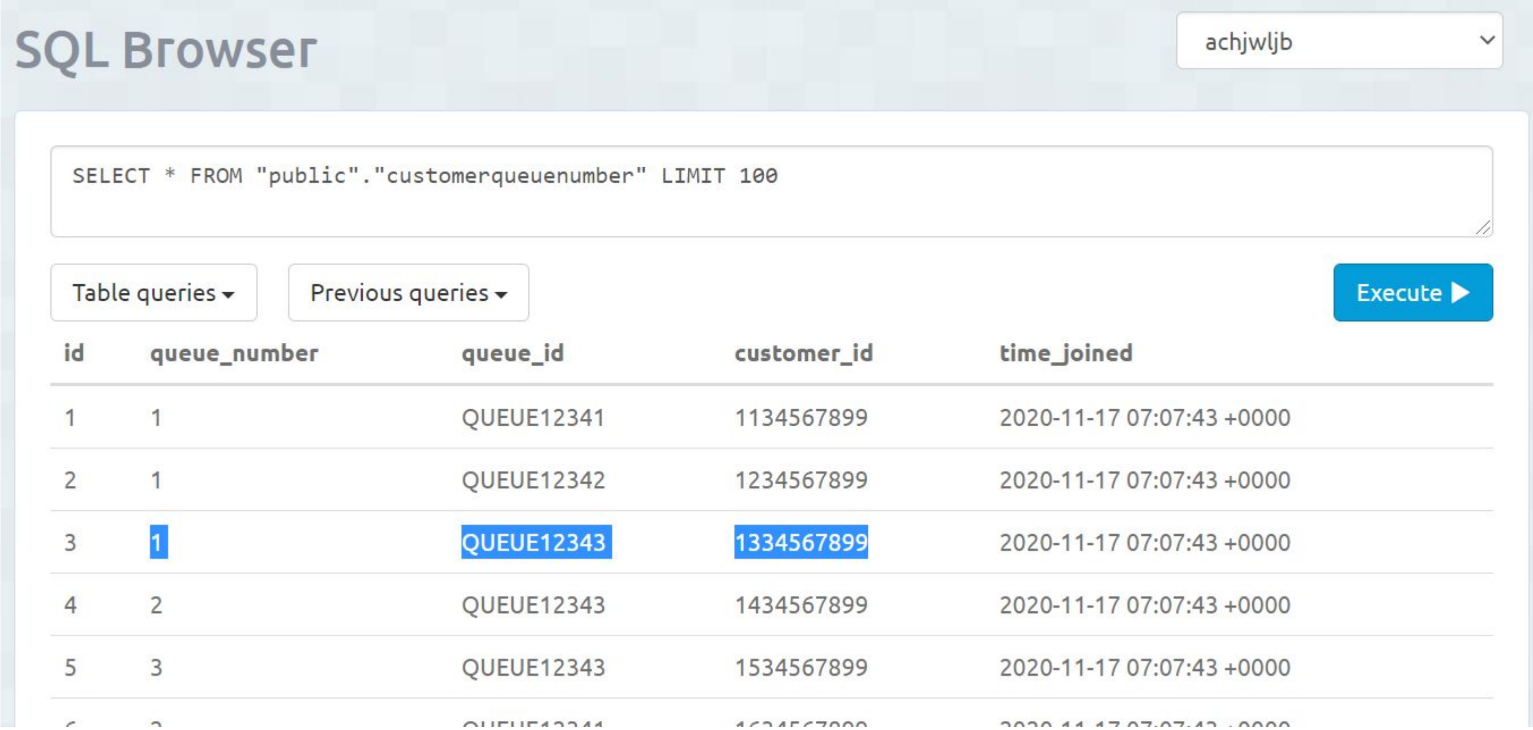
	
	PUT http://localhost:3000/company/server
	Body: raw JSON
	`{ "queue_id": "QUEUE12343" }`
	Status: 200 OK Time: 358 ms Size: 295 B
	`[ { "customer_id": 1334567899 } ]`
	 Send
	Params
	Authorization
	Headers (10)
	Body
	Pre-request Script
	Tests
	Settings
	none
	form-data
	x-www-form-urlencoded
	raw
	binary
	GraphQL
	JSON
	Body
	Cookies
	Headers (8)
	Test Results
	Sav
	Pretty
	Raw
	Preview
	Visualize
	JSON
	For queue_id (“QUEUE12343”) in table Queue, the current_queue_number updated from 0 to 1.



```
SELECT * FROM "public"."queue" LIMIT 100
```

queue_id	current_queue_number	status
QUEUE12342	0	false
QUEUE12344	0	false
QUEUE12345	0	false
QUEUE12346	0	false
QUEUE12347	0	false
QUEUE12348	0	false
QUEUE24261	0	false
QUEUE12340	0	false
QUEUE12341	2	true
QUEUE12343	1	true

In table CustomerQueueNumber, the customer_id that holds queue_number 1 in QUEUE12343 is 1334567899 which is shown in Postman.



```
SELECT * FROM "public"."customerqueuenumber" LIMIT 100
```

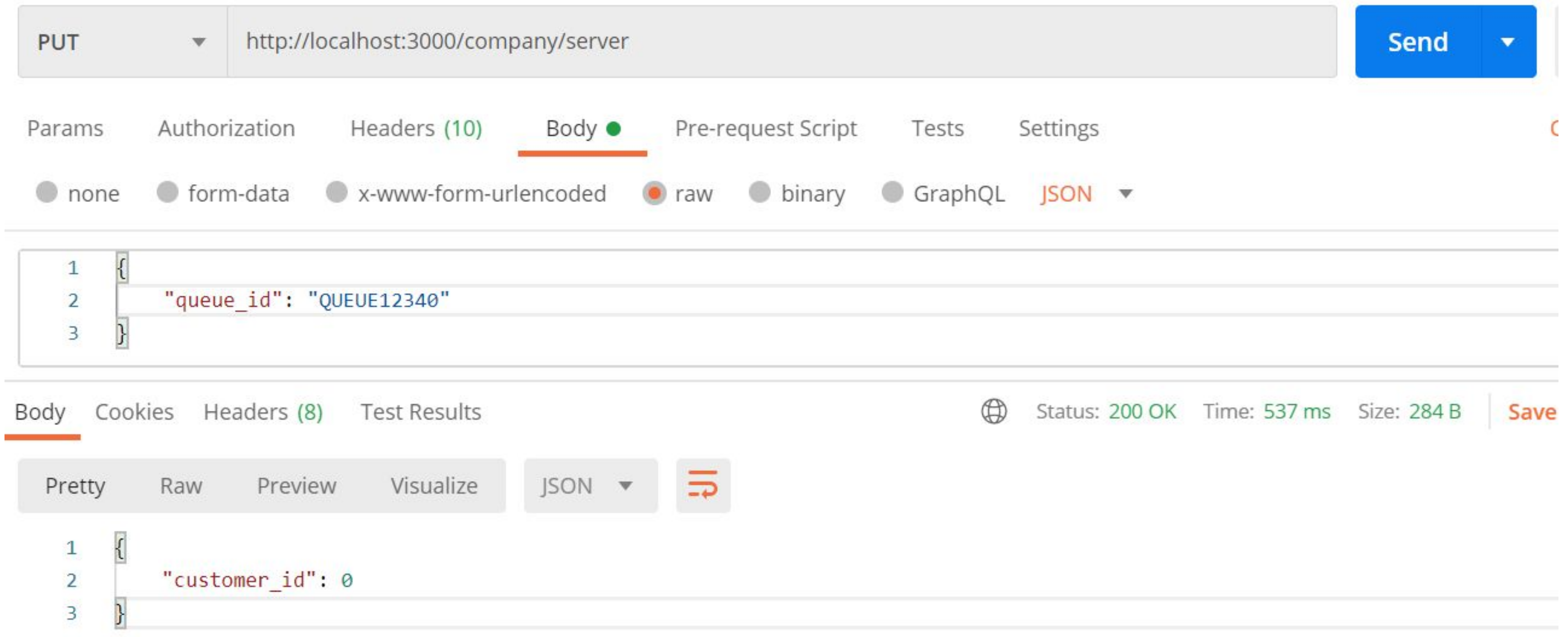
id	queue_number	queue_id	customer_id	time_joined
1	1	QUEUE12341	1134567899	2020-11-17 07:07:43 +0000
2	1	QUEUE12342	1234567899	2020-11-17 07:07:43 +0000
3	1	QUEUE12343	1334567899	2020-11-17 07:07:43 +0000
4	2	QUEUE12343	1434567899	2020-11-17 07:07:43 +0000
5	3	QUEUE12343	1534567899	2020-11-17 07:07:43 +0000
6	2	QUEUE12341	1634567899	2020-11-17 07:07:43 +0000

3.2 Successful request

Title	Server Available API - Successful request but no customer in Queue.
Description	A successful request made to the server should allow the company to indicate that they have a server that is available to serve the next customer in the queue if any.
Precondition	Tables should be created, queue_id (“QUEUE12340”) with no customer in the queue.

Test Steps	1. Start the Backend Server.
	2. Send an Server Available request to the backend by using Postman:
	Method: PUT
	Path: http://localhost:3000/company/server
	Body:
	{
	“queue_id”: “QUEUE12340”
	}
	3. Receive a 200 OK response from Postman.

Expected Result	queue_id (“QUEUE12340”) should show customer_id=0 in Postman when don't have the next customer in the queue to the server.

Evidence	Since there is no next customer to serve in QUEUE12340, ‘customer_id’:0 will be shown in Postman
	
	 Send
	Params
	Authorization
	Headers (10)
	Body
	Pre-request Script
	Tests
	Settings
	none
	form-data
	x-www-form-urlencoded
	binary
	GraphQL
	Cookies
	Headers (8)
	Test Results
	Save
	Raw
	Preview
	Visualize
	PUT http://localhost:3000/company/server
	Body: raw JSON
	`{`
	`    "queue_id": "QUEUE12340"`
	`}`
	Body — Status: 200 OK Time: 537 ms Size: 284 B
	Pretty JSON
	`{`
	`    "customer_id": 0`
	`}`

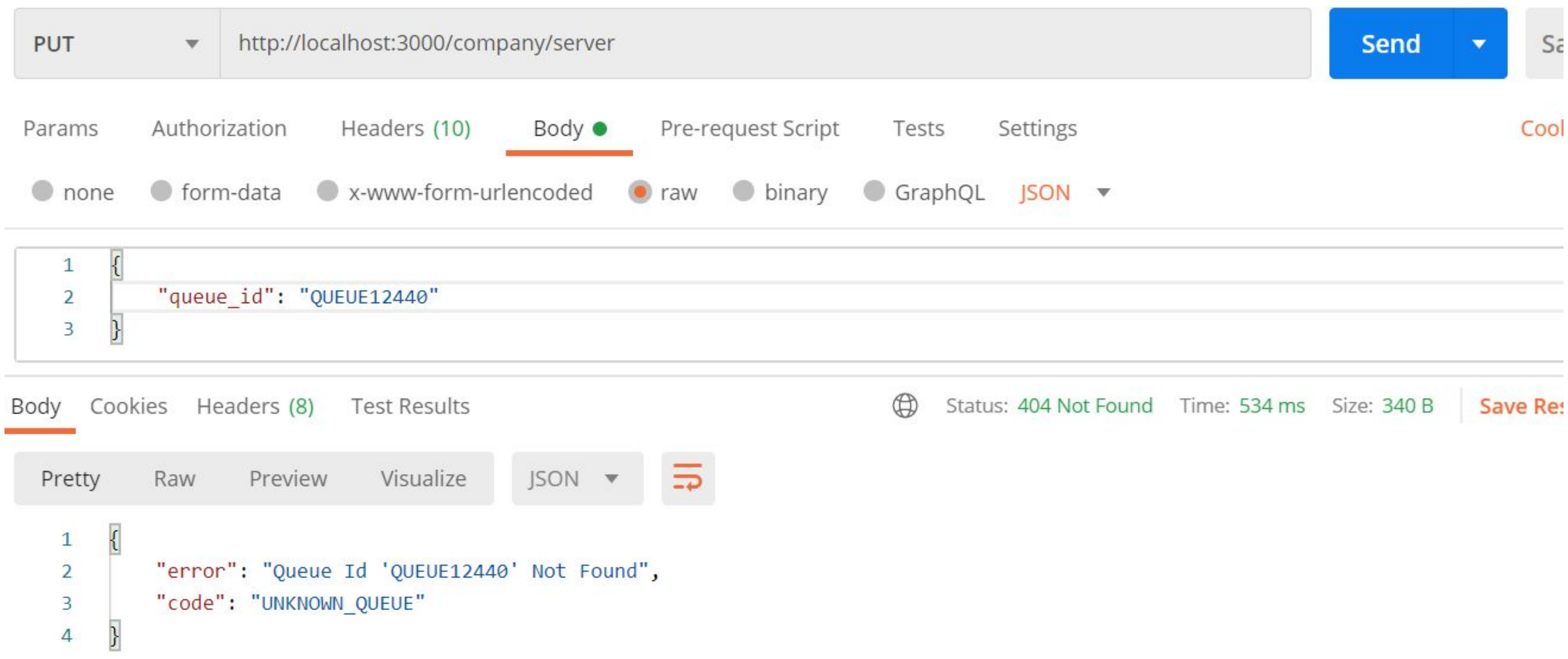
3.3 Non-existence Queue Id

Title	Server Available Queue API - Non-existence Queue Id
Description	An unsuccessful request made to the server due to non-existence queue_id should return an error that indicates that the queue id does not exist.
Precondition	Table should be created, queue_id (“QUEUE12440”) does not already exist.

Test Steps	1. Start the Backend Server.
	2. Send an Server Available request to the backend by using Postman:
	Method: PUT
	Path: http://localhost:3000/company/server
	Body:
	{
	“queue_id”: “QUEUE12440”
	}
	3. Receive a 404 Not Found response from Postman.

Expected Result	Receive a 404 Not Found response from Postman that has an error message with error code that indicates the queue_id (“QUEUE12440”) not found in the table.

Evidence



http://localhost:3000/company/server

○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
    "queue_id": "QUEUE12440"
}
```

Status: 404 Not Found Time: 534 ms Size: 340 B

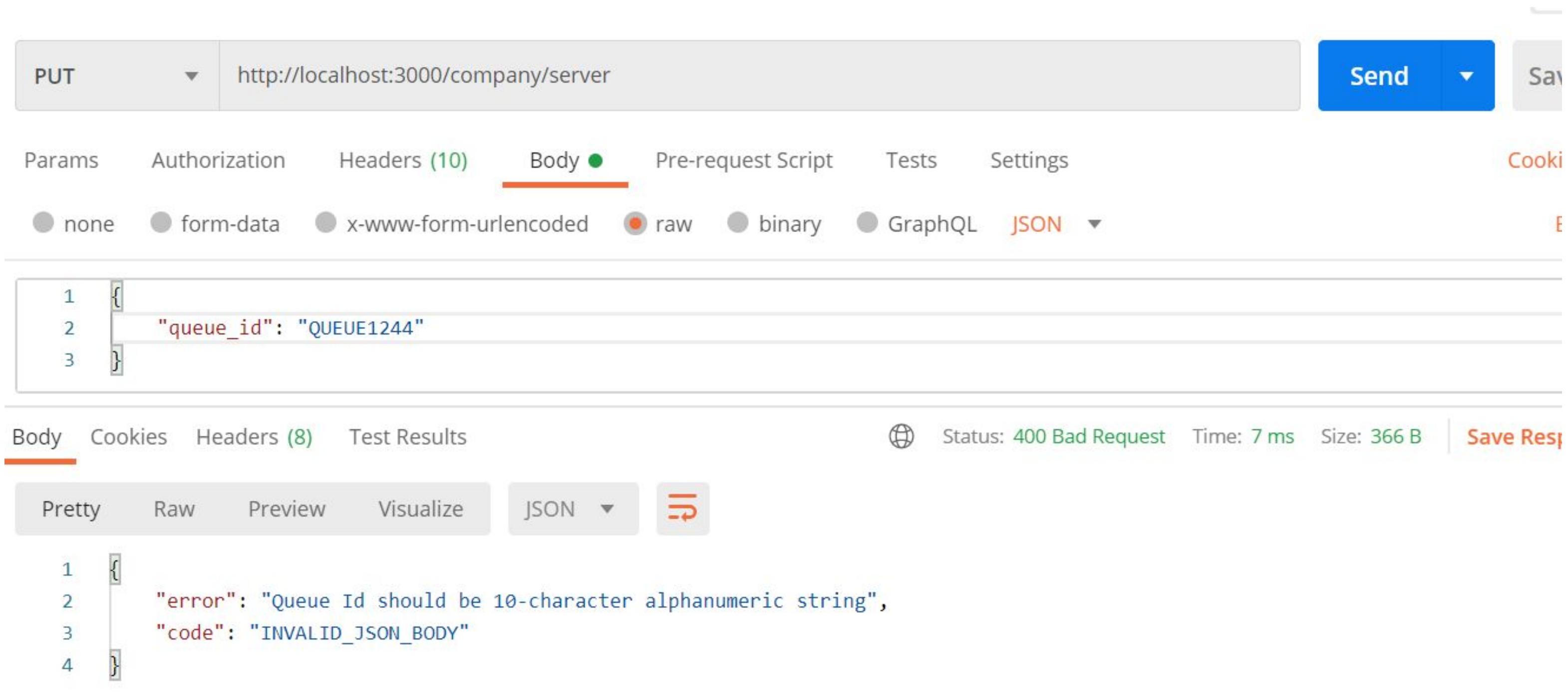
```
{
    "error": "Queue Id 'QUEUE12440' Not Found",
    "code": "UNKNOWN_QUEUE"
}
```

3.4 Queue Id Validation Failed

Title	Server Available API - Queue Id Validation Failed.
Description	An unsuccessful request made to the server due to invalid queue_id should return an error that indicates the queue_id is invalid.
Precondition	Table should be created.

Test Steps	1. Start the Backend Server.
	2. Send an Server Available request to the backend by using Postman:
	Method: PUT
	Path: http://localhost:3000/company/server
	Body:
	{
	“queue_id”: “QUEUE1244”
	}
	3. Receive a 400 Bad Request response from Postman that has an error message indicating the queue_id is invalid.

Expected Result	Receive a 400 Bad Request response from Postman that has an error message with a code that indicates the queue_id is invalid.

Evidence	

PUT http://localhost:3000/company/server

Body: ● raw JSON

```
{
    "queue_id": "QUEUE1244"
}
```

Status: 400 Bad Request Time: 7 ms Size: 366 B

```
{
    "error": "Queue Id should be 10-character alphanumeric string",
    "code": "INVALID_JSON_BODY"
}
```

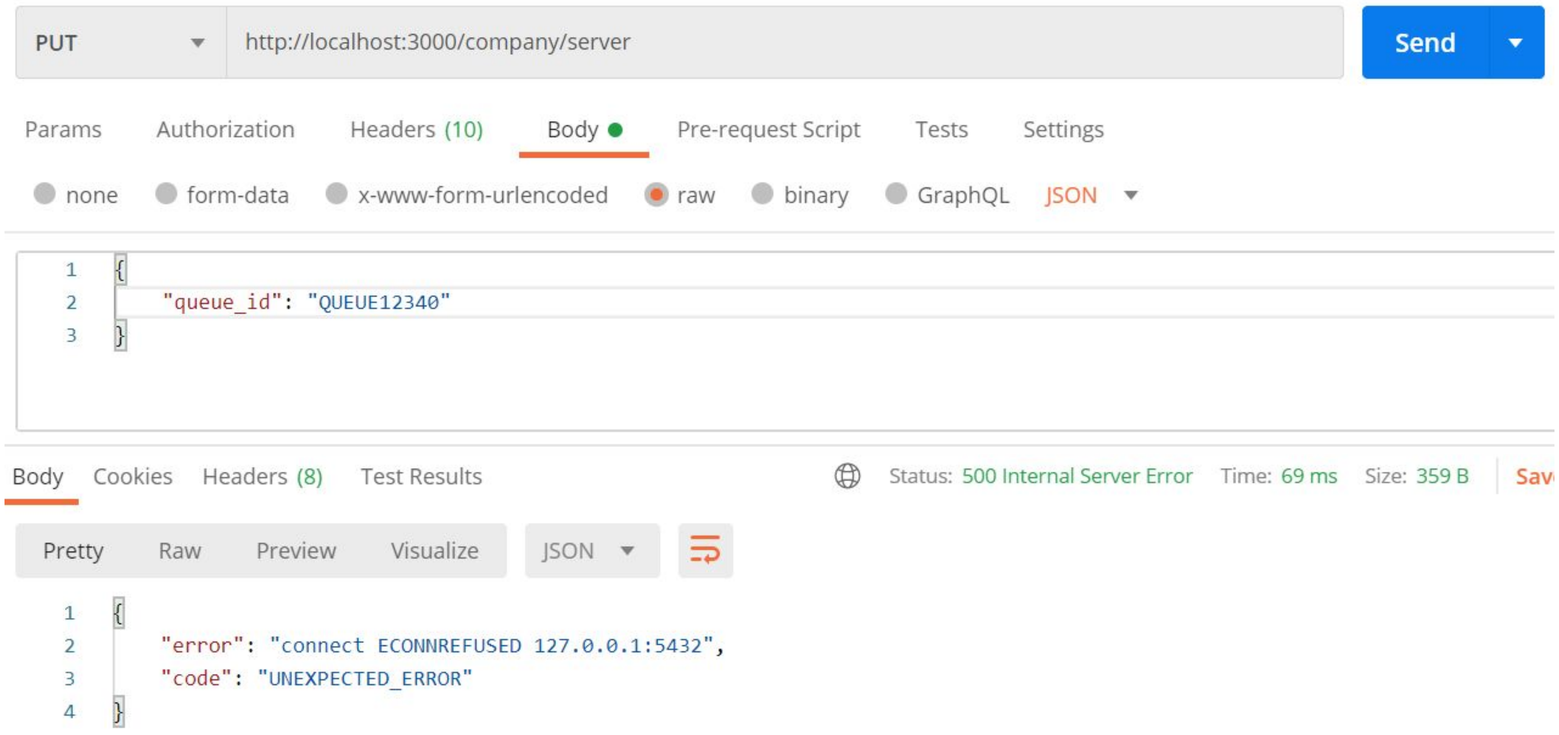
3.5 Unexpected Server Error

Title	Server Available API - Unexpected Server Error.
Description	An unsuccessful request made to the server due to server error should return an error indicating what happened.
Precondition	Tables should be created, wrong database credentials in database.js.

Test Steps	1. Start the Backend Server.
	2. Send an Server Available request to the backend by using Postman:
	Method: PUT
	Path: http://localhost:3000/company/server
	Body:
	{
	“queue_id”: “QUEUE12440”
	}
	3. Receive a 500 Internal Server response from Postman.

Expected Result	Receive a 500 Internal Server response from Postman that has an error message with a code that indicates what happened.

Evidence



PUT http://localhost:3000/company/server

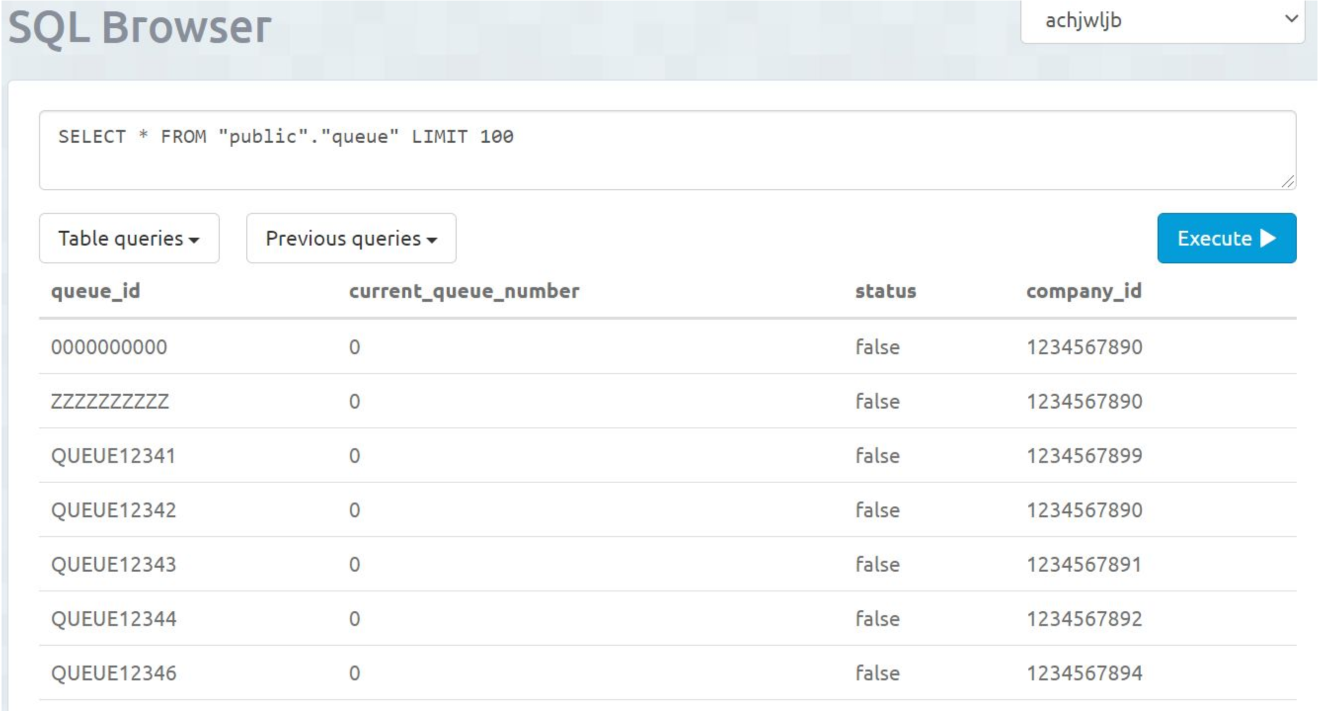
```
{
    "queue_id": "QUEUE12340"
}
```

Status: 500 Internal Server Error Time: 69 ms Size: 359 B

```
{
    "error": "connect ECONNREFUSED 127.0.0.1:5432",
    "code": "UNEXPECTED_ERROR"
}
```

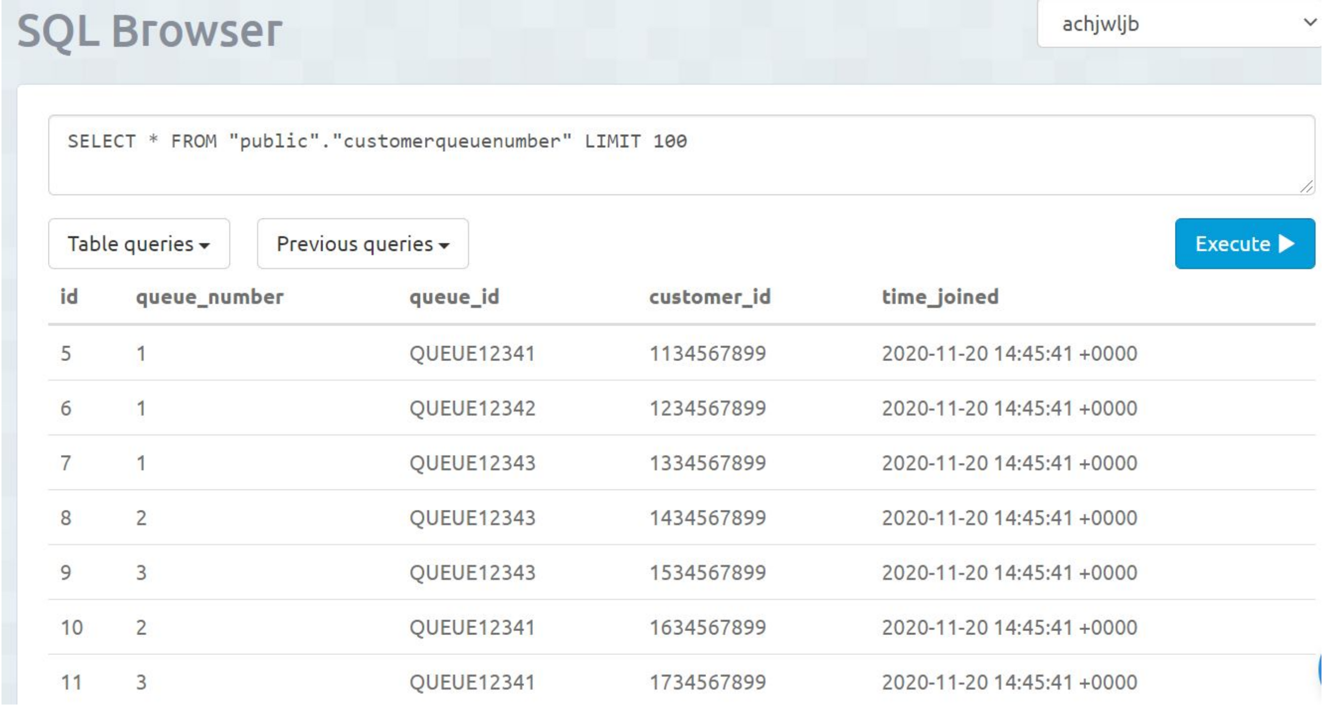
4. Reset API

4.1 Successful request

Title	Reset API - Successful request reset table in database.
Description	A successful request made to the server should reset the database to its fresh state with no data inserted.
Precondition	Tables should be created with data inserted. Table - Queue 

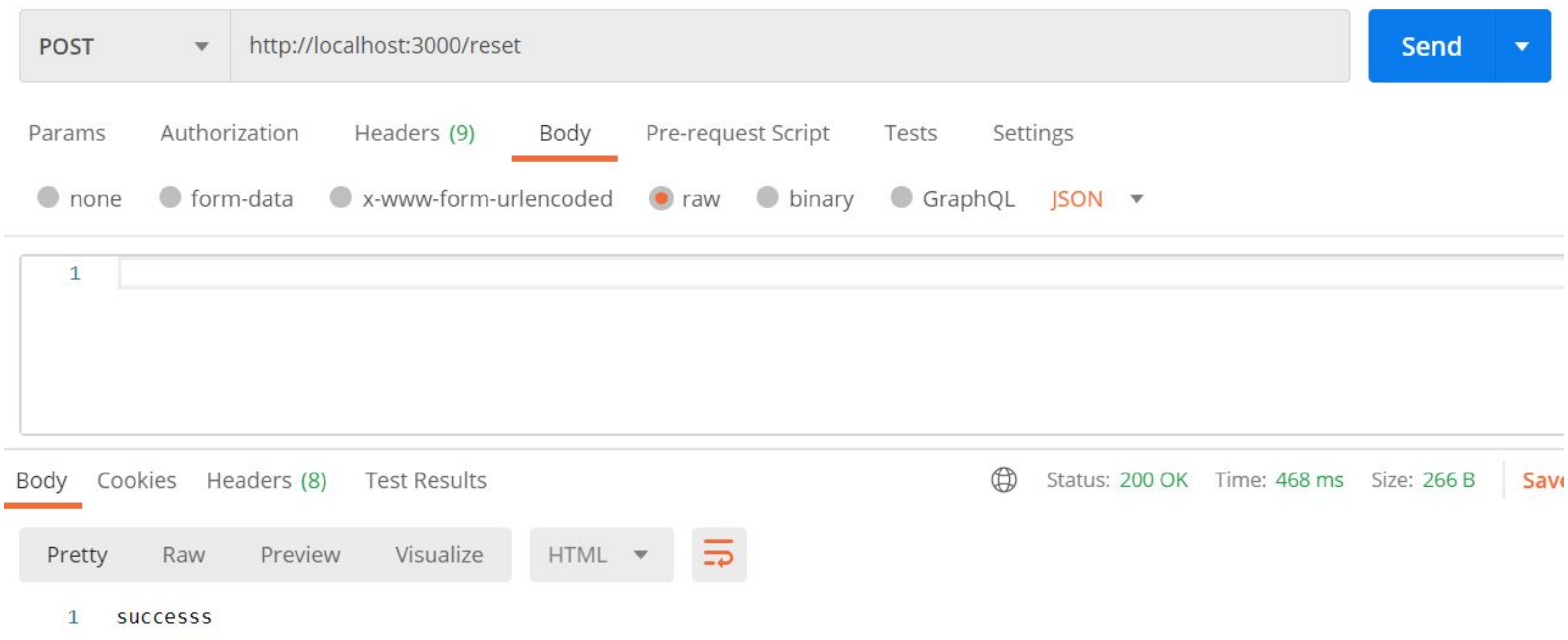
```
SELECT * FROM "public"."queue" LIMIT 100
```

queue_id	current_queue_number	status	company_id
0000000000	0	false	1234567890
ZZZZZZZZZZ	0	false	1234567890
QUEUE12341	0	false	1234567899
QUEUE12342	0	false	1234567890
QUEUE12343	0	false	1234567891
QUEUE12344	0	false	1234567892
QUEUE12346	0	false	1234567894

<table>
<tr><td></td><td>Table - CustomerQueueNumber

SQL Browser


```
SELECT * FROM "public"."customerqueuenumber" LIMIT 100
```

<table>
<tr><th>id</th><th>queue_number</th><th>queue_id</th><th>customer_id</th><th>time_joined</th></tr>
<tr><td>5</td><td>1</td><td>QUEUE12341</td><td>1134567899</td><td>2020-11-20 14:45:41 +0000</td></tr>
<tr><td>6</td><td>1</td><td>QUEUE12342</td><td>1234567899</td><td>2020-11-20 14:45:41 +0000</td></tr>
<tr><td>7</td><td>1</td><td>QUEUE12343</td><td>1334567899</td><td>2020-11-20 14:45:41 +0000</td></tr>
<tr><td>8</td><td>2</td><td>QUEUE12343</td><td>1434567899</td><td>2020-11-20 14:45:41 +0000</td></tr>
<tr><td>9</td><td>3</td><td>QUEUE12343</td><td>1534567899</td><td>2020-11-20 14:45:41 +0000</td></tr>
<tr><td>10</td><td>2</td><td>QUEUE12341</td><td>1634567899</td><td>2020-11-20 14:45:41 +0000</td></tr>
<tr><td>11</td><td>3</td><td>QUEUE12341</td><td>1734567899</td><td>2020-11-20 14:45:41 +0000</td></tr>
</table></td></tr>
<tr><td><b>Test Steps</b></td><td>1. Start the Backend Server.
2. Send a Reset request to the backend by using Postman.
Method: POST
Path: http://localhost:3000/reset
3. Receive a 200 OK response from Postman.
4. Go to elephantsql to check the Queue and CustomerQueueNumber table.</td></tr>
<tr><td><b>Expected Result</b></td><td>Data in the tables are reset to its fresh state with no data inserted.</td></tr>
<tr><td><b>Evidence</b></td><td>200 OK response from Postman.</td></tr>
</table>



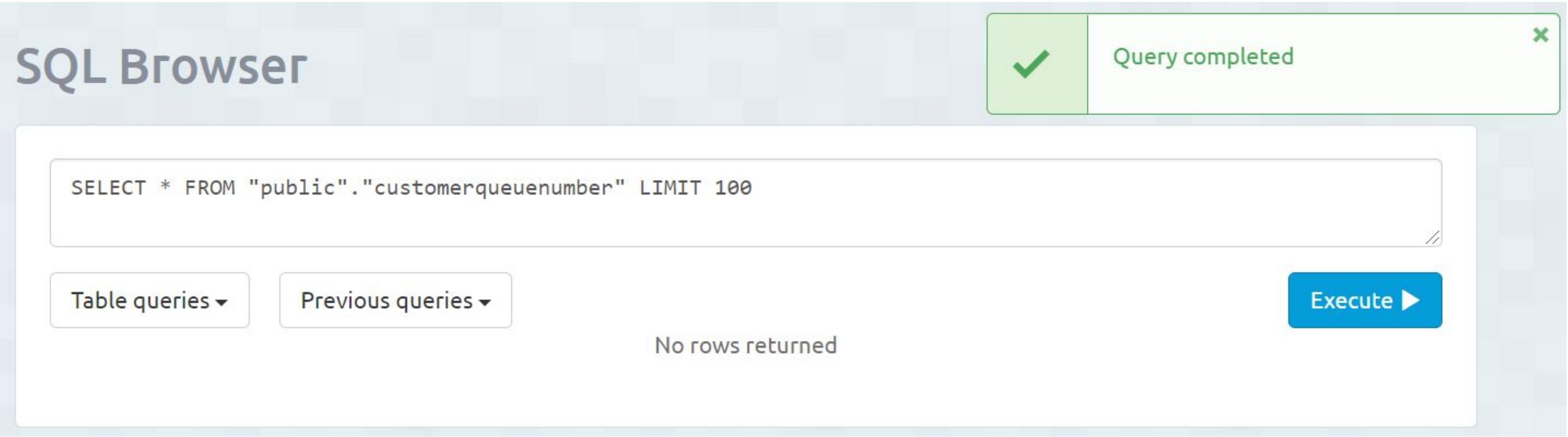
POST http://localhost:3000/reset

Body: ● raw JSON

Body Status: 200 OK Time: 468 ms Size: 266 B

```
successs
```

Table Queue in ElephantSQL

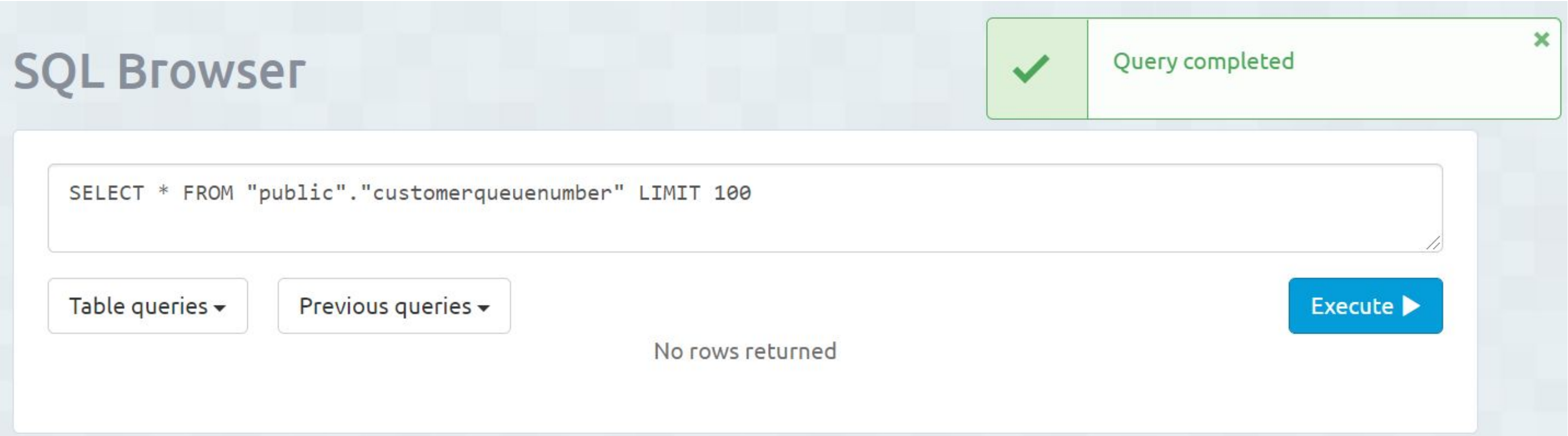


✓ Query completed

```
SELECT * FROM "public"."customerqueuenumber" LIMIT 100
```

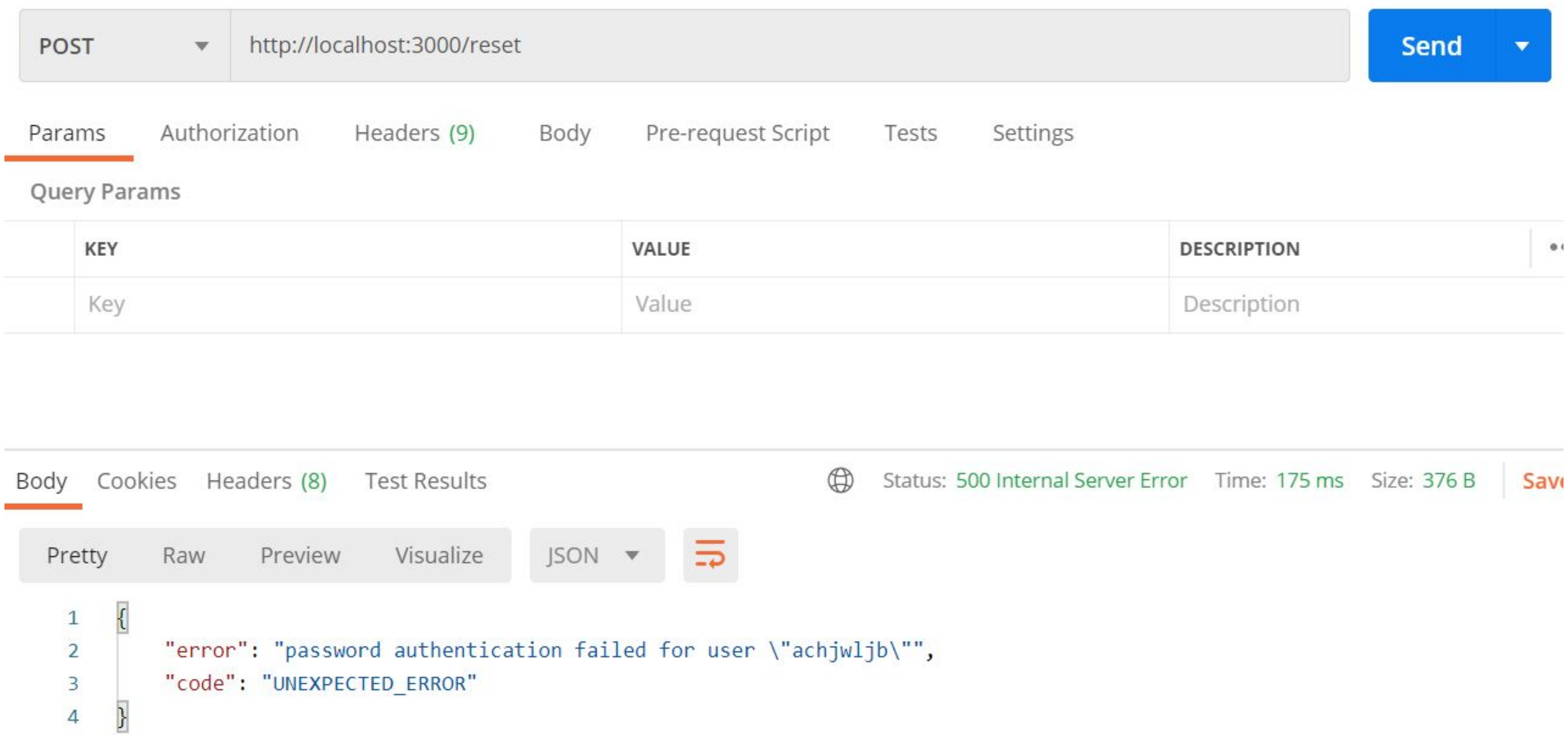
No rows returned

Table CustomerQueueNumber in ElephantSQL

	 SQL Browser Query completed `SELECT * FROM "public"."customerqueuenumber" LIMIT 100` No rows returned

4.2 Unexpected Server Error

Title	Reset API - Unexpected Server Error -Unsuccessful requests will not reset tables in the database.
Description	An unsuccessful request made to the server due to server error should return an error indicating what happened.
Precondition	Tables should be created, wrong database credentials in database.js.
Test Steps	1. Start the Backend Server. 2. Send a Reset request to the backend by using Postman. Method: POST Path: http://localhost:3000/reset 3. Receive a 500 Internal Server response from Postman. 4. Go to elephantsql to check the Queue and CustomerQueueNumber table.
Expected Result	Receive a 500 Internal Server response from Postman that has an error message with a code that indicates what happened. Unsuccessful requests won't reset tables in the database.
Evidence	500 Internal Server response from Postman.

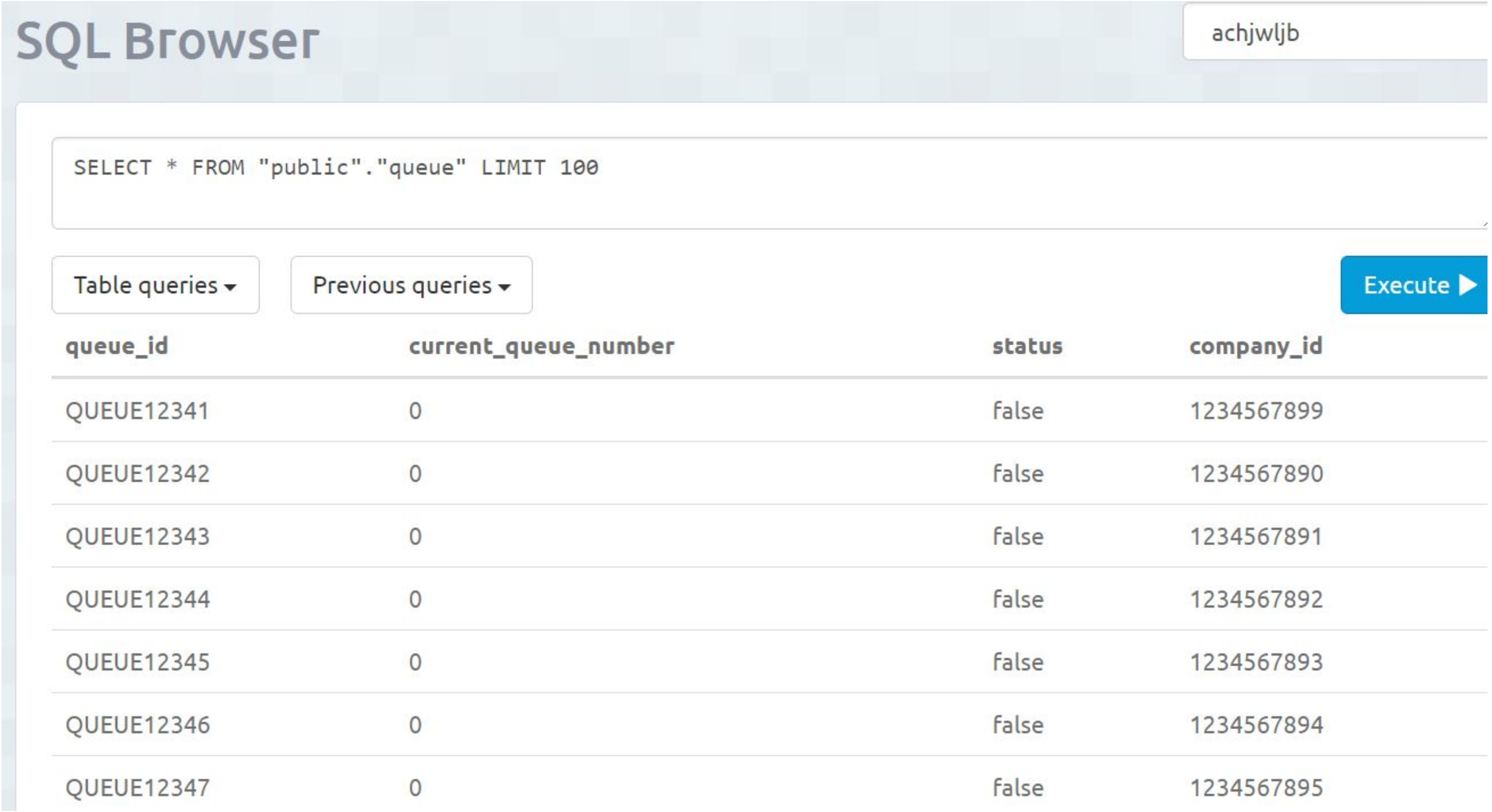


POST http://localhost:3000/reset

Status: 500 Internal Server Error Time: 175 ms Size: 376 B

```
{
    "error": "password authentication failed for user \"achjwljb\"",
    "code": "UNEXPECTED_ERROR"
}
```

Below is the table in ElephantSQL. Unsuccessful requests won't reset tables in the database.

```
SELECT * FROM "public"."queue" LIMIT 100
```

queue_id	current_queue_number	status	company_id
QUEUE12341	0	false	1234567899
QUEUE12342	0	false	1234567890
QUEUE12343	0	false	1234567891
QUEUE12344	0	false	1234567892
QUEUE12345	0	false	1234567893
QUEUE12346	0	false	1234567894
QUEUE12347	0	false	1234567895

5. Join Queue API

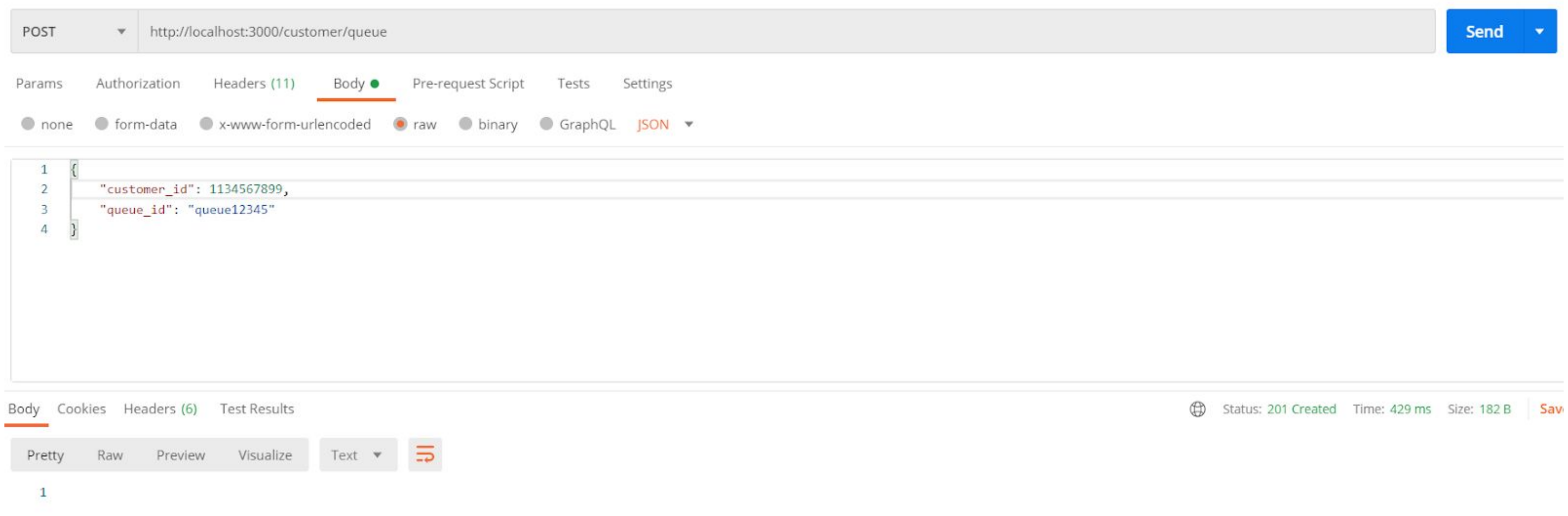
5.1 Successful request - Queue Id Not Case Sensitive

Title	Join Queue API - Queue Id not case sensitive.
Description	Successful request indicating that the customer has joined the queue.
Precondition	queue_id(“QUEUE12345”) must exist in table CustomerQueueNumber and customer_id(1134567899) must not be inside.

Test Steps	1. Start the Backend Server.
	2. Send a Join Queue request to the backend by using Postman:
	Method: POST
	Path: http://localhost:3000/customer/queue
	Body:
	{
	“customer_id”: 1134567899,
	“queue_id”: “queue12345”
	}
	3. Receive a 201 Created response code from Postman.

Expected Result	Receive a 201 Created response indicating that the customer joined the queue successfully.

Evidence

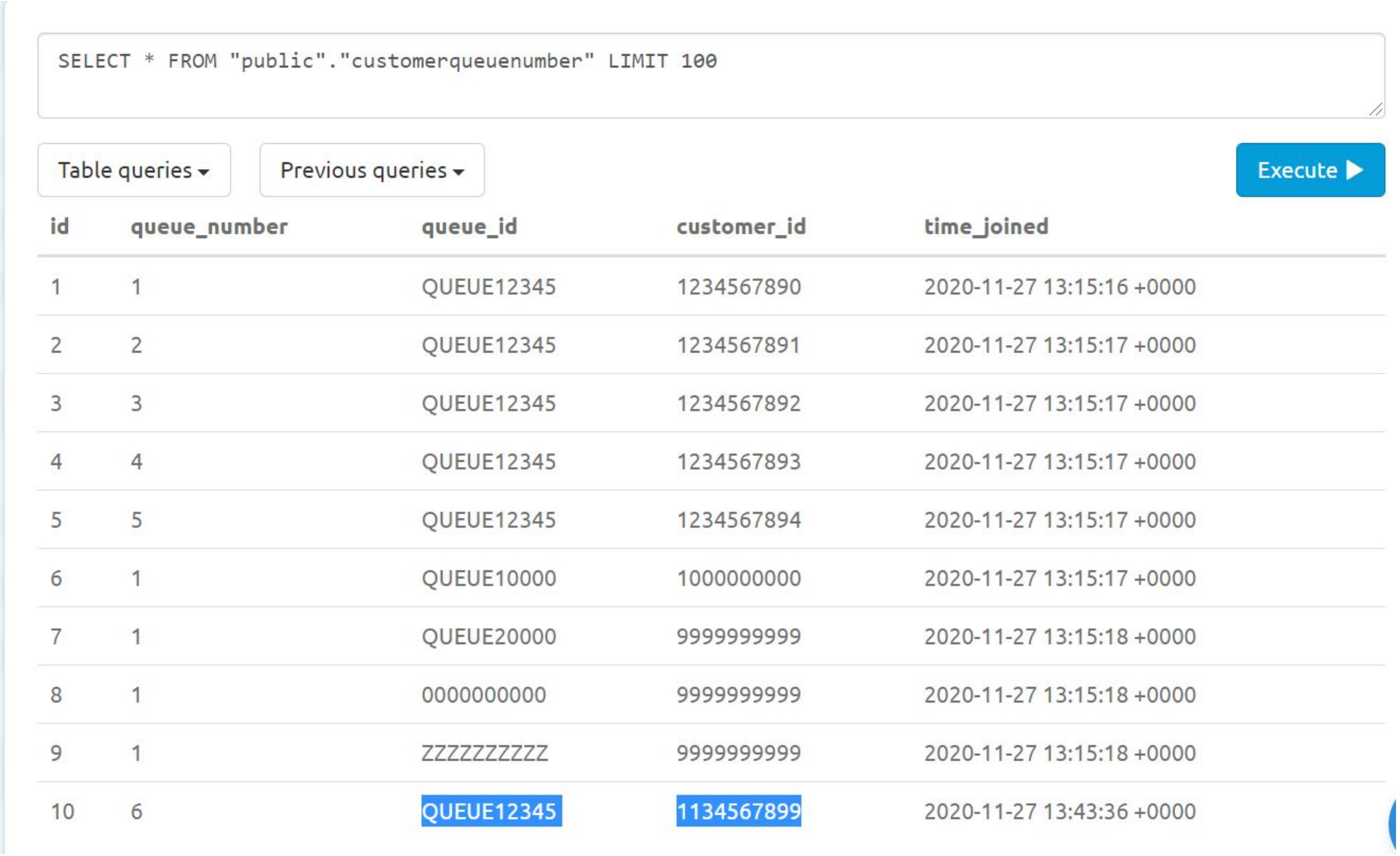


POST http://localhost:3000/customer/queue

```
{
    "customer_id": 1134567899,
    "queue_id": "queue12345"
}
```

Status: 201 Created Time: 429 ms Size: 182 B

The customer with customer_id(1134567899) joined the queue with queue_id("QUEUE12345) successfully.



```
SELECT * FROM "public"."customerqueuenumber" LIMIT 100
```

id	queue_number	queue_id	customer_id	time_joined
1	1	QUEUE12345	1234567890	2020-11-27 13:15:16 +0000
2	2	QUEUE12345	1234567891	2020-11-27 13:15:17 +0000
3	3	QUEUE12345	1234567892	2020-11-27 13:15:17 +0000
4	4	QUEUE12345	1234567893	2020-11-27 13:15:17 +0000
5	5	QUEUE12345	1234567894	2020-11-27 13:15:17 +0000
6	1	QUEUE10000	1000000000	2020-11-27 13:15:17 +0000
7	1	QUEUE20000	9999999999	2020-11-27 13:15:18 +0000
8	1	0000000000	9999999999	2020-11-27 13:15:18 +0000
9	1	ZZZZZZZZZZ	9999999999	2020-11-27 13:15:18 +0000
10	6	QUEUE12345	1134567899	2020-11-27 13:43:36 +0000

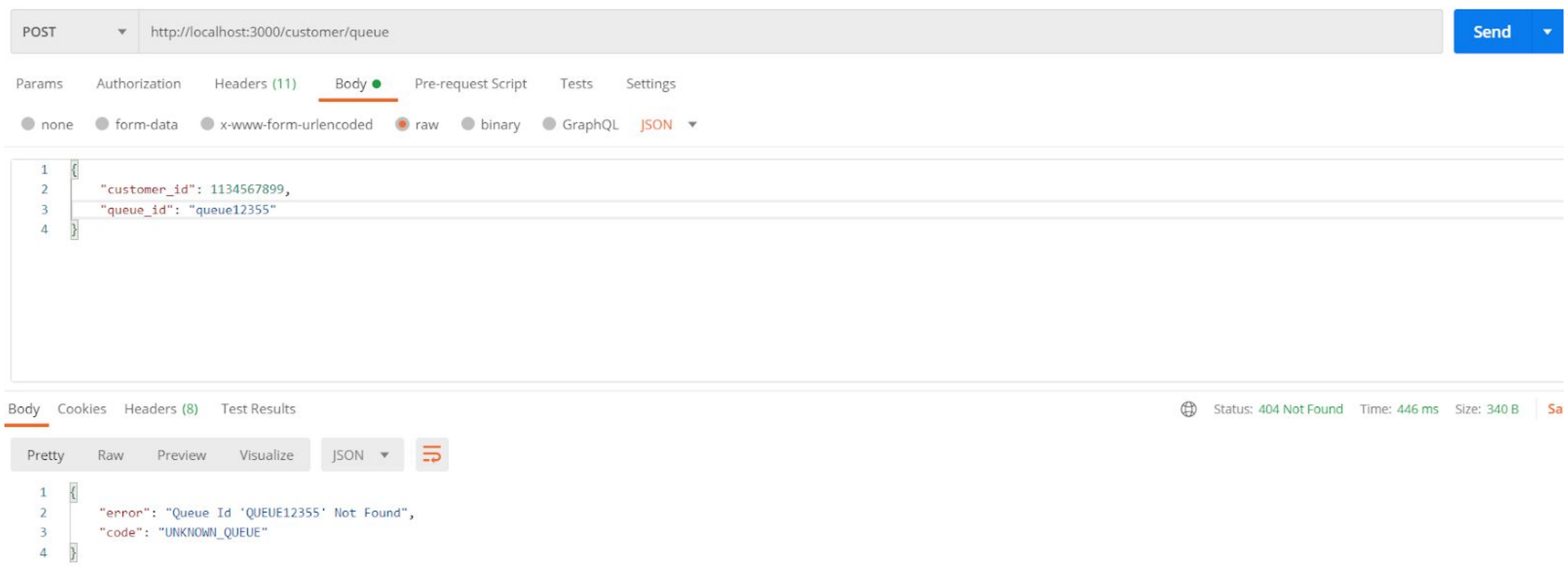
5.2 Non-existence Queue Id

Title	Join Queue API - Non-existence Queue Id.
Description	Failed request indicating that the queue id doesn't exist.
Precondition	queue_id(“QUEUE12355”) must not be in the table.

Test Steps	1. Start the Backend Server.
	2. Send a Join Queue request to the backend by using Postman:
	Method: POST
	Path: http://localhost:3000/customer/queue
	Body:
	{
	“customer_id”: 1134567899,
	“queue_id”: “queue12355”
	}
	3. Receive a 404 Not Found response from Postman.

Expected Result	Receive a 404 Not Found response indicating that the queue does not exist.

Evidence



POST http://localhost:3000/customer/queue

○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
    "customer_id": 1134567899,
    "queue_id": "queue12355"
}
```

Status: 404 Not Found Time: 446 ms Size: 340 B Sa

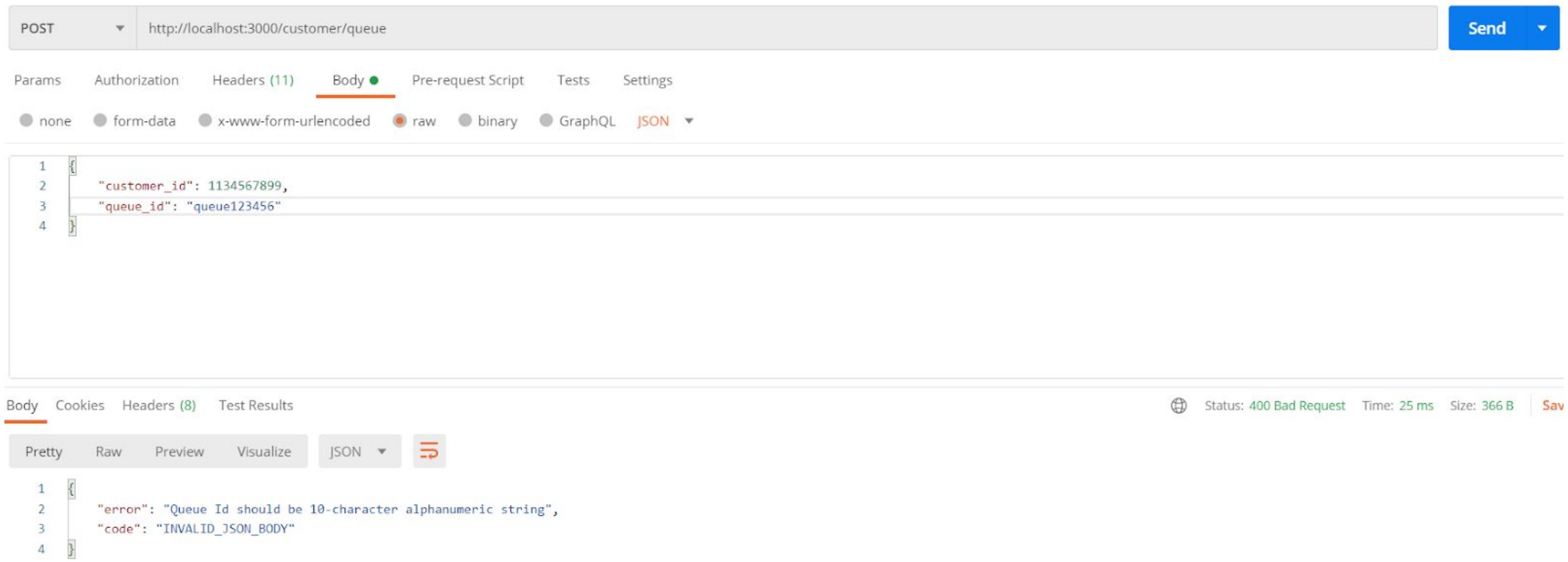
```
{
    "error": "Queue Id 'QUEUE12355' Not Found",
    "code": "UNKNOWN_QUEUE"
}
```

5.3 Queue Id Validation Failed

Title	Join Queue API - Queue Id Validation Failed.
Description	Failed request indicating that the queue id is not valid.
Precondition	Tables must be created.

Test Steps	1. Start the Backend Server.
	2. Send a Join Queue request to the backend by using Postman:
	Method: POST
	Path: http://localhost:3000/customer/queue
	Body:
	{
	“customer_id”: 1134567899,
	“queue_id”: “queue123456”
	}
	3. Receive a 400 Bad request from Postman.

Expected Result	Receive a 400 Bad Request indicating that the queue is invalid.

Evidence	

http://localhost:3000/customer/queue

```
{
    "customer_id": 1134567899,
    "queue_id": "queue123456"
}
```

Status: 400 Bad Request Time: 25 ms Size: 366 B

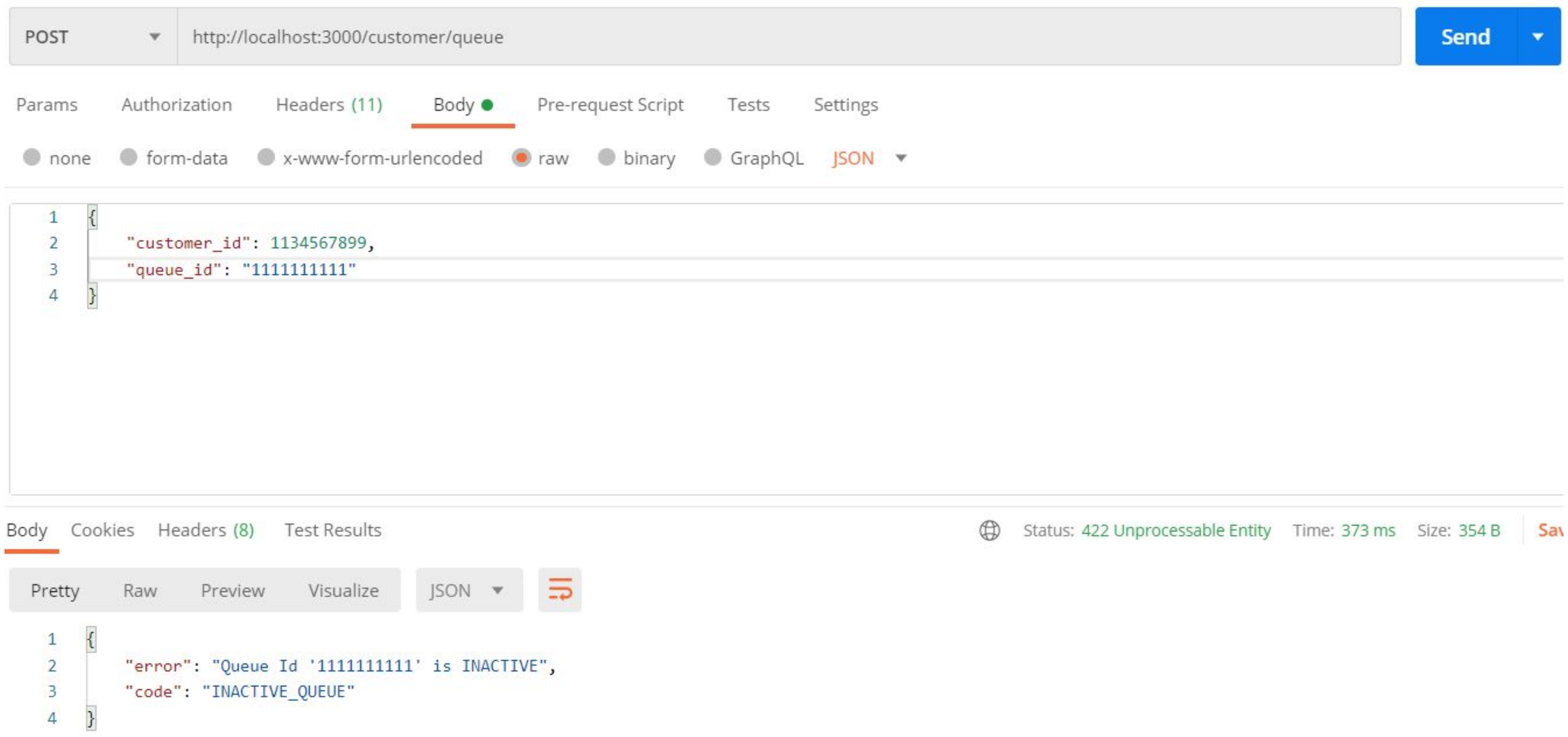
```
{
    "error": "Queue Id should be 10-character alphanumeric string",
    "code": "INVALID_JSON_BODY"
}
```

5.4 Queue Id is Inactive

Title	Join Queue API - Queue Id is Inactive.
Description	Failed request indicating that the queue id is not active.
Precondition	Status of queue_id(“111111111”) is false.

Test Steps	1. Start the Backend Server.
	2. Send a Join Queue request to the backend by using Postman:
	Method: POST
	Path: http://localhost:3000/customer/queue
	Body:
	{
	“customer_id”: 1134567899,
	“queue_id”: “1111111111”
	}
	3. Receive a 422 Unprocessable Entity response from Postman.

Expected Result	Receive a 422 Unprocessable Entity response indicating that the queue is inactive.

<table>
<tr>
<td><strong>Evidence</strong></td>
<td>


http://localhost:3000/customer/queue

```
{
    "customer_id": 1134567899,
    "queue_id": "1111111111"
}
```

Status: 422 Unprocessable Entity Time: 373 ms Size: 354 B

```
{
    "error": "Queue Id '1111111111' is INACTIVE",
    "code": "INACTIVE_QUEUE"
}
```
</td>
</tr>
</table>

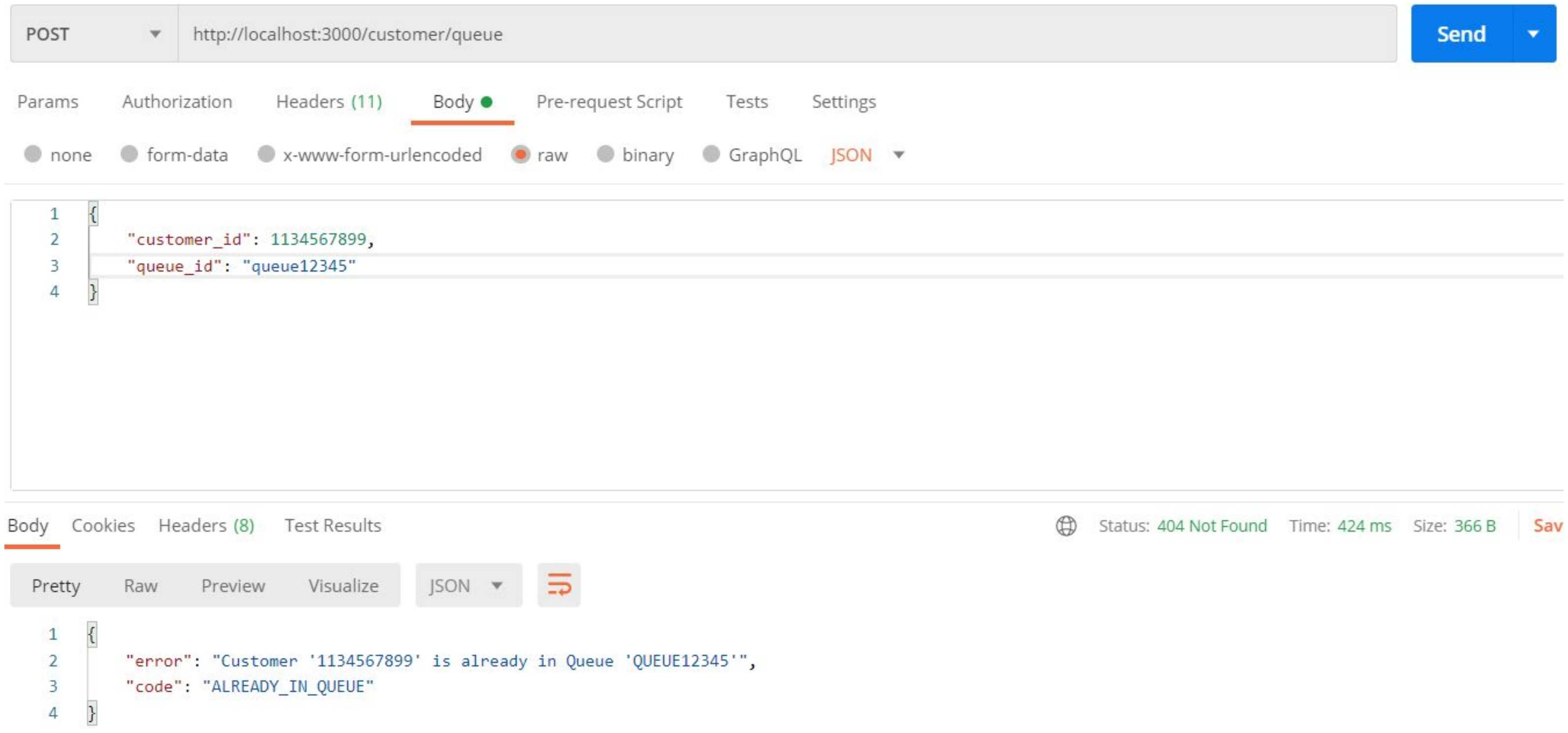
5.5 Customer Already In Queue

Title	Join Queue API - Customer already in queue.
Description	Failed request indicating that the customer was already in the queue.
Precondition	customer_id(1134567899) inside the queue already.

Test Steps	1. Start the Backend Server.
	2. Send a Join Queue request to the backend by using Postman:
	Method: POST
	Path: http://localhost:3000/customer/queue
	Body:
	{
	“customer_id”: 1134567899,
	“queue_id”: “queue123456”
	}
	3. Receive a 404 Not Found response from Postman.

Expected Result	Receive a 404 Not Found response indicating that the customer was already in the queue from Postman.

Evidence



http://localhost:3000/customer/queue

```
{
    "customer_id": 1134567899,
    "queue_id": "queue12345"
}
```

Status: 404 Not Found Time: 424 ms Size: 366 B

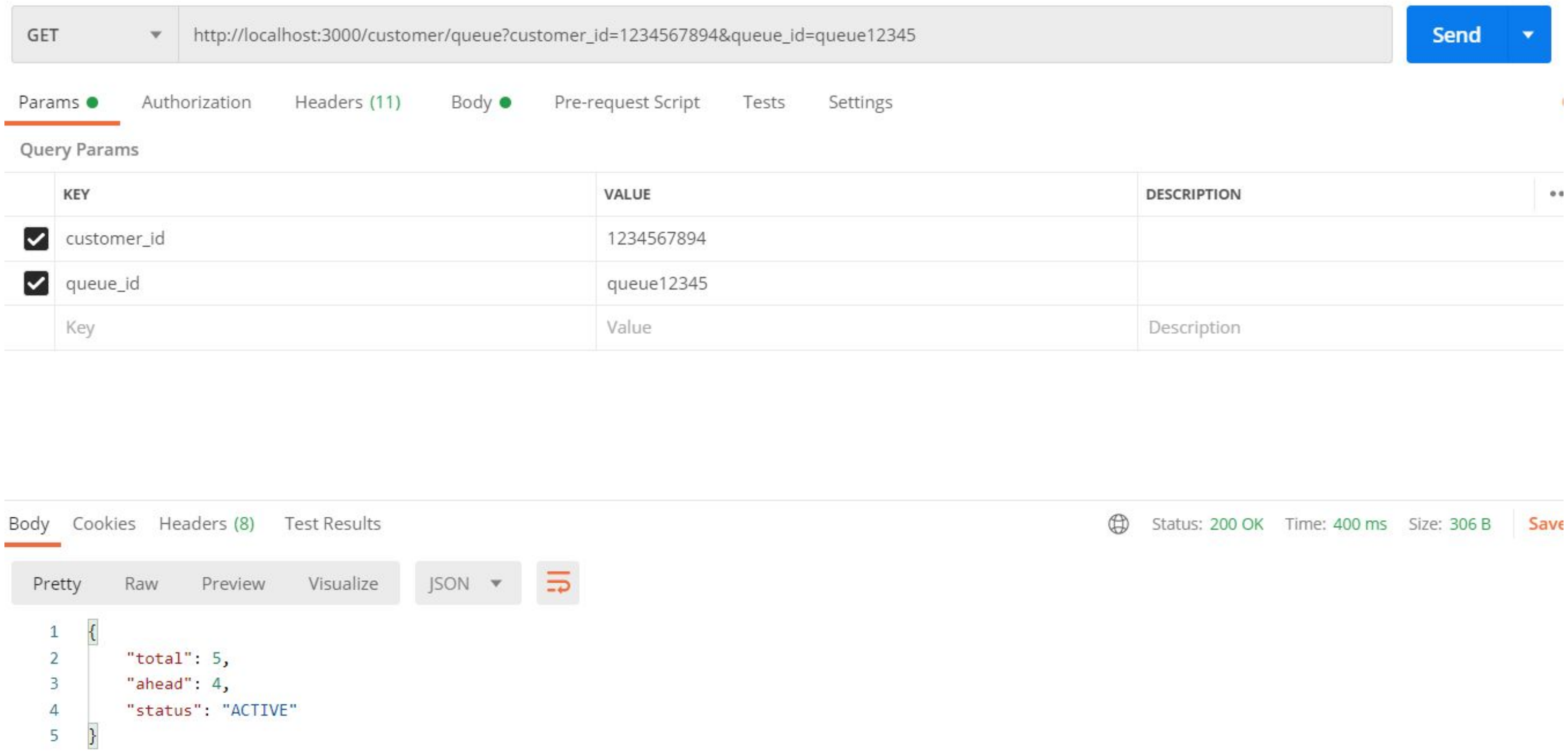
```
{
    "error": "Customer '1134567899' is already in Queue 'QUEUE12345'",
    "code": "ALREADY_IN_QUEUE"
}
```

6. Check Queue API

6.1 Successful Request

Title	Check Queue API - Successful Request.
Description	Success request showing the customer position in the queue.
Precondition	customer_id(1234567894) and queue_id("QUEUE12345") already exist.
Test Steps	1. Start the Backend Server. 2. Send a Check Queue request to the backend by using Postman: Method: GET Path: http://localhost:3000/customer/queue?customer_id=1234567894&queue_id=queue12345 3. Receive a 200 OK response from Postman and json body.
Expected Result	Receive a 200 OK response and show the position of the customer in the queue.

Evidence



GET http://localhost:3000/customer/queue?customer_id=1234567894&queue_id=queue12345

	KEY	VALUE	DESCRIPTION
☑	customer_id	1234567894	
☑	queue_id	queue12345	
	Key	Value	Description

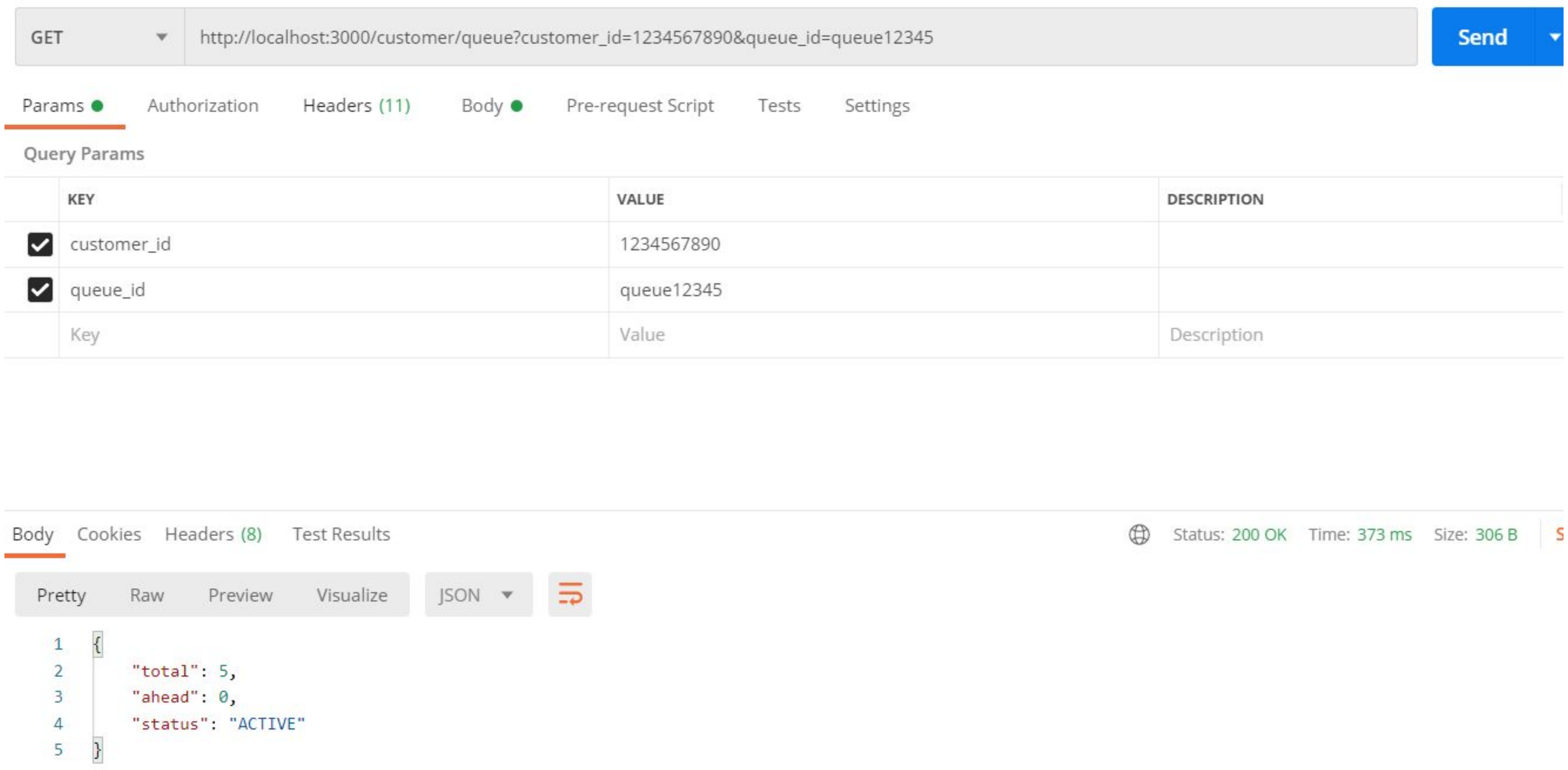
Status: 200 OK Time: 400 ms Size: 306 B

```
{
    "total": 5,
    "ahead": 4,
    "status": "ACTIVE"
}
```

6.2 Successful Request - Customer to be served next

Title	Check Queue API - Successful Request - Customer to be served next.
Description	Success request showing the customer position in the queue.
Precondition	customer_id(1234567890) and queue_id("QUEUE12345") already exist.
Test Steps	1. Start the Backend Server. 2. Send a Check Queue request to the backend by using Postman: Method: GET Path: http://localhost:3000/customer/queue?customer_id=1234567890&queue_id=queue12345 3. Receive a 200 OK code and json body from Postman.
Expected Result	Receive a 200 OK response and json body that show the position of the customer in the queue from Postman.

Evidence



http://localhost:3000/customer/queue?customer_id=1234567890&queue_id=queue12345

Query Params

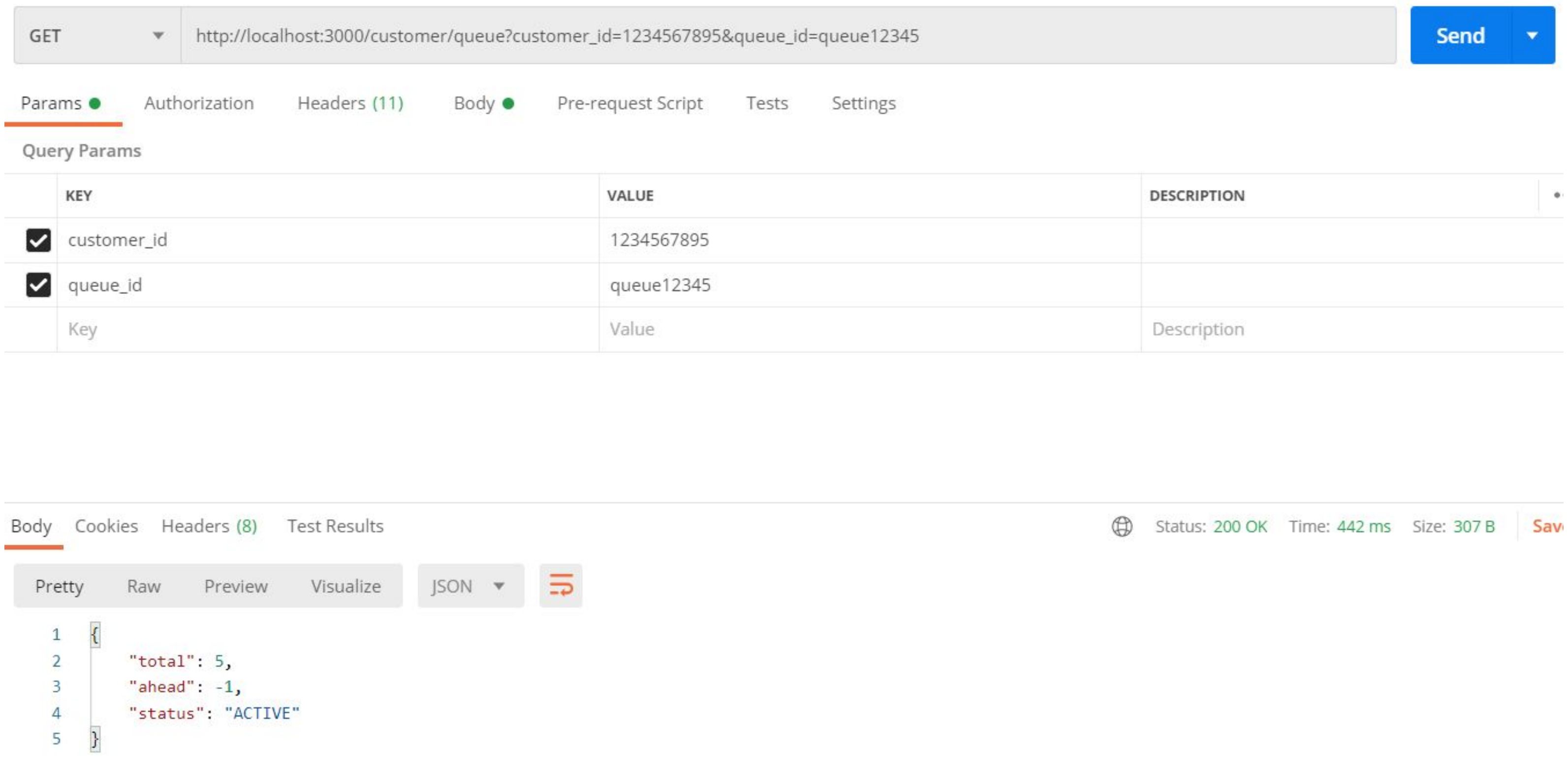
	KEY	VALUE	DESCRIPTION
☑	customer_id	1234567890	
☑	queue_id	queue12345	
	Key	Value	Description

Status: 200 OK Time: 373 ms Size: 306 B

```
{
    "total": 5,
    "ahead": 0,
    "status": "ACTIVE"
}
```

6.3 Successful Request - Customer Not in Queue

Title	Check Queue API - Successful Request - Customer Not in Queue.
Description	Success request showing the customer position in the queue.
Precondition	customer_id(1234567895) and queue_id(“QUEUE12345”) already exist.
Test Steps	1. Start the Backend Server. 2. Send an Check Queue request to the backend by using Postman: Method: GET Path: http://localhost:3000/customer/queue?customer_id=1234567895&queue_id=queue12345 3. Receive a 200 OK code and json body from Postman .
Expected Result	Receive a 200 OK response and json body that show the position of the customer in the queue from Postman.

Evidence	

GET http://localhost:3000/customer/queue?customer_id=1234567895&queue_id=queue12345

Query Params

	KEY	VALUE	DESCRIPTION
☑	customer_id	1234567895	
☑	queue_id	queue12345	
	Key	Value	Description

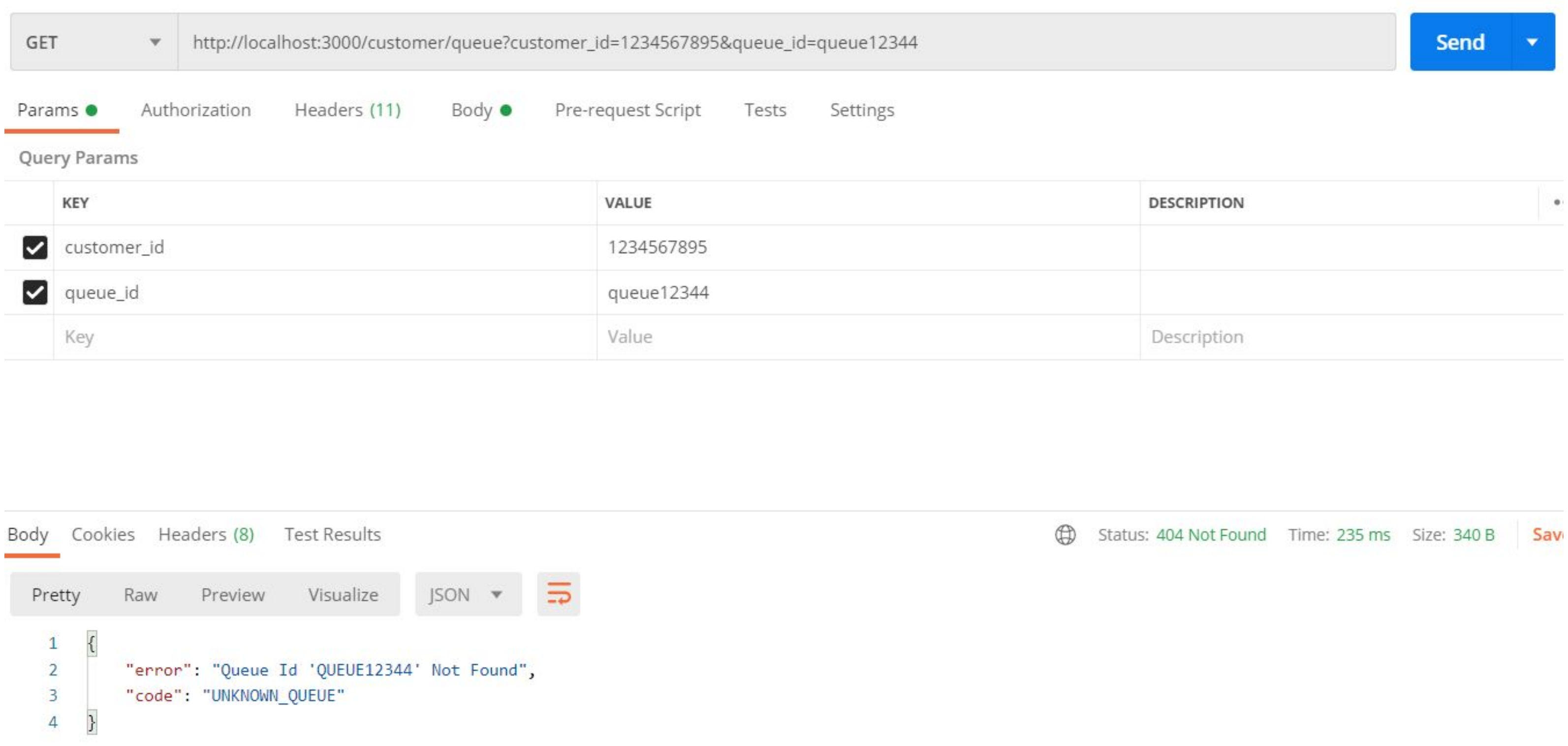
Status: 200 OK Time: 442 ms Size: 307 B

```
{
    "total": 5,
    "ahead": -1,
    "status": "ACTIVE"
}
```

6.4 Queue Id Not Found

Title	Check Queue API - Queue Id Not Found
Description	Error response indicating queue id not found.
Precondition	queue_id(“QUEUE12344”) does not already exist in the table.
Test Steps	1. Start the Backend Server. 2. Send an Check Queue request to the backend by using Postman: Method: GET Path: http://localhost:3000/customer/queue?customer_id=1234567895&queue_id=queue12344 3. Receive a 404 Not Found code and json body from Postman.
Expected Result	Receive a 404 Not Found code and the error message from Postman.

Evidence



GET http://localhost:3000/customer/queue?customer_id=1234567895&queue_id=queue12344

Send
Params ● Authorization Headers (11) Body ● Pre-request Script Tests Settings

Query Params

	KEY	VALUE	DESCRIPTION
☑	customer_id	1234567895	
☑	queue_id	queue12344	
	Key	Value	Description

Body Cookies Headers (8) Test Results

Status: 404 Not Found Time: 235 ms Size: 340 B

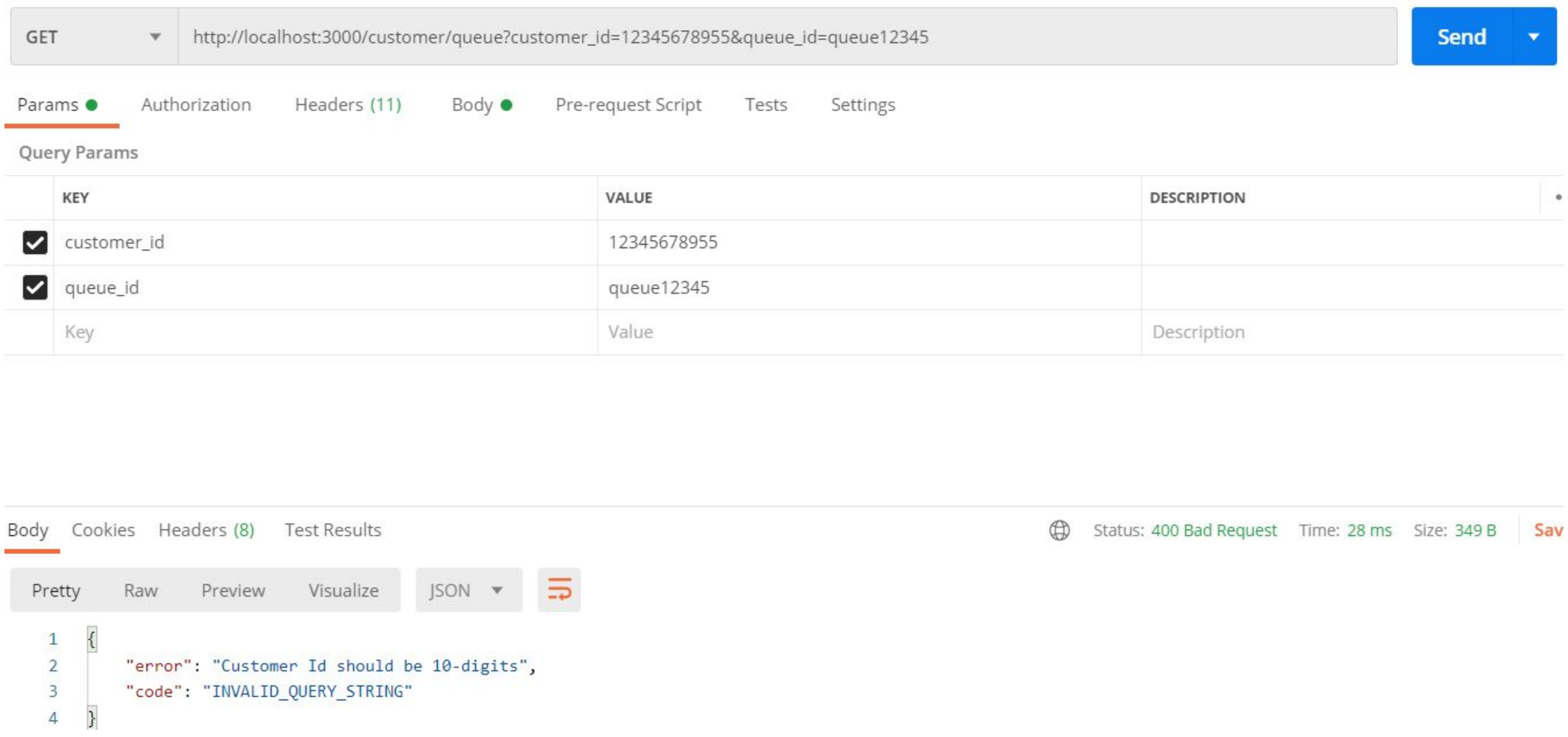
Sav
Pretty Raw Preview Visualize JSON

```
{
    "error": "Queue Id 'QUEUE12344' Not Found",
    "code": "UNKNOWN_QUEUE"
}
```

6.5 Invalid Customer Id

Title	Check Queue API - Invalid Customer Id.
Description	Error response indicating customer id is invalid.
Precondition	Invalid customer_id(12345678955).
Test Steps	1. Start the Backend Server. 2. Send an Check Queue request to the backend by using Postman: Method: GET Path: http://localhost:3000/customer/queue?customer_id=12345678955&queue_id=queue12345 3. Receive a 400 Bad Request and json body from Postman.
Expected Result	Receive a 400 Bad Request and the error message from Postman.

Evidence



GET http://localhost:3000/customer/queue?customer_id=12345678955&queue_id=queue12345

Query Params

	KEY	VALUE	DESCRIPTION
☑	customer_id	12345678955	
☑	queue_id	queue12345	
	Key	Value	Description

Status: 400 Bad Request Time: 28 ms Size: 349 B

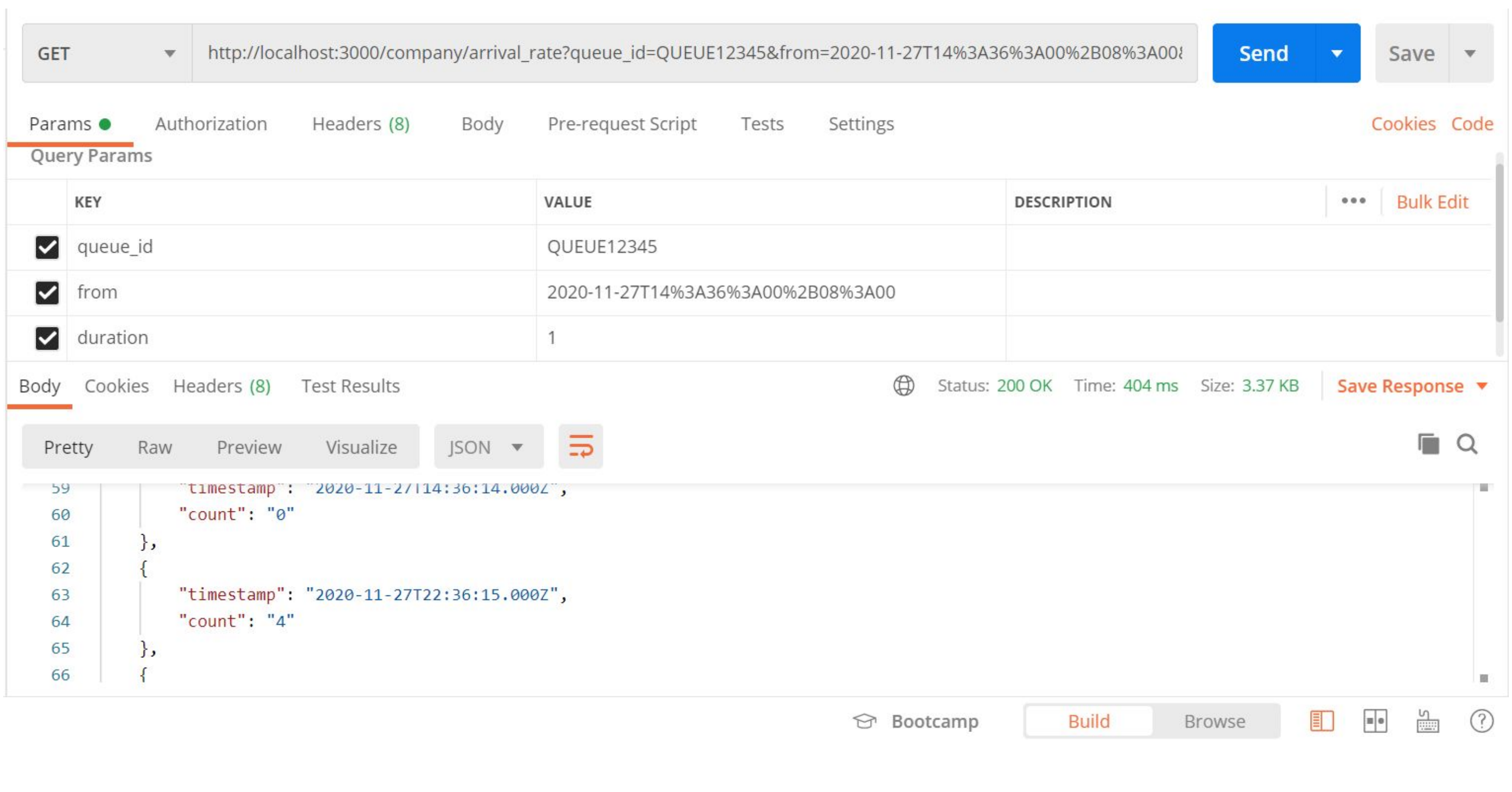
```
{
    "error": "Customer Id should be 10-digits",
    "code": "INVALID_QUERY_STRING"
}
```

7. Arrival Rate API

7.1 Successful response

Title	Arrival Rate API .
Description	Success Response should show the number of arrivals within a certain period of time.
Precondition	Tables should be created with meaningful records.
Test Steps	1. Start the Backend Server. 2. Send an Arrival Rate request to the backend by using Postman: Method: GET Path:http://localhost:3000/company/arrival_rate?queue_id=QUEUE12345&from=2020-11-27T14%3A36%3A00%2B08%3A00&duration=1 3. Receive a 200 OK from Postman.
Expected Result	Receive a 200 OK from Postman and json body that show the number of arrivals within a certain period of time.

Evidence



GET http://localhost:3000/company/arrival_rate?queue_id=QUEUE12345&from=2020-11-27T14%3A36%3A00%2B08%3A00&

Query Params

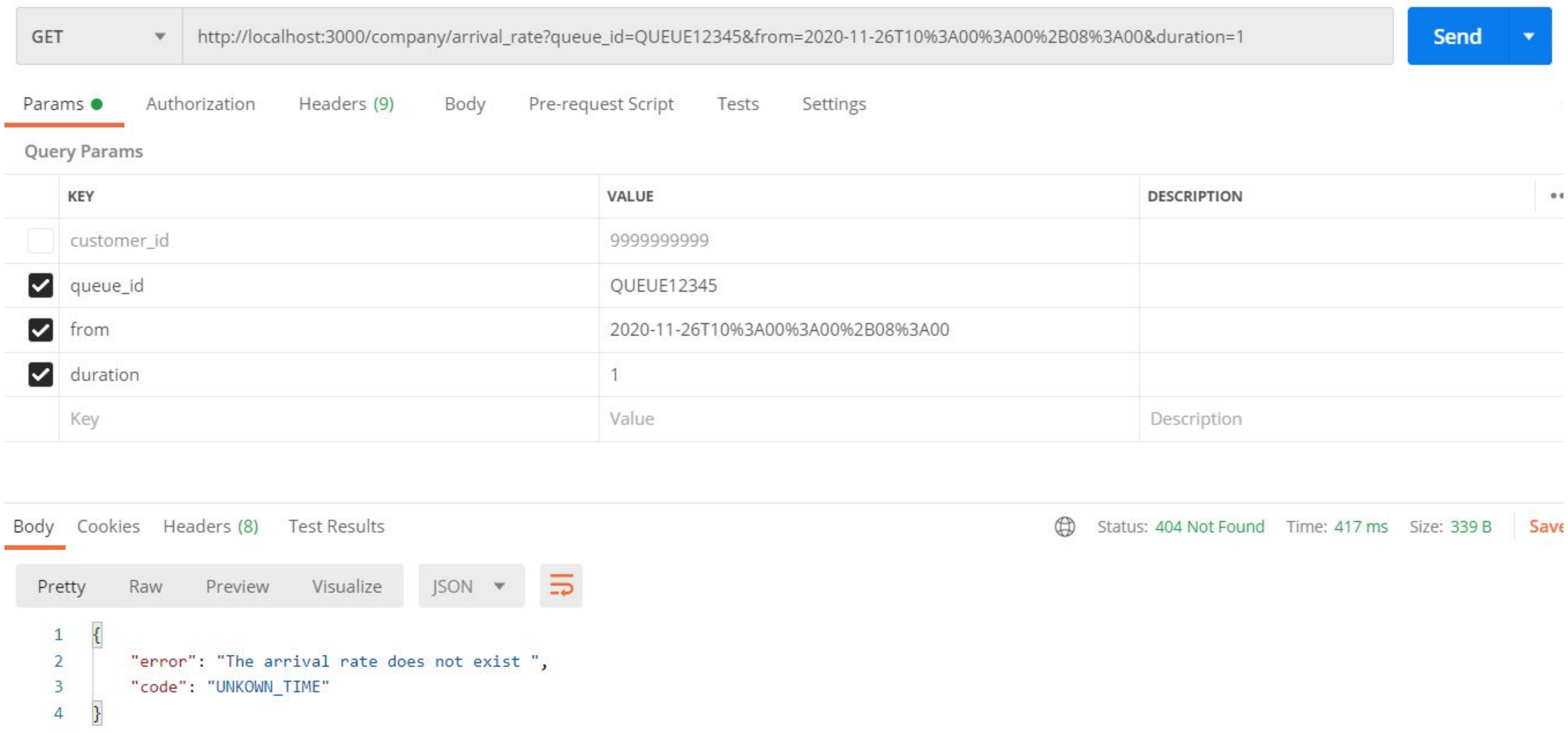
	KEY	VALUE	DESCRIPTION
☑	queue_id	QUEUE12345	
☑	from	2020-11-27T14%3A36%3A00%2B08%3A00	
☑	duration	1	

Status: 200 OK Time: 404 ms Size: 3.37 KB

```
        "timestamp": "2020-11-27T14:36:14.000Z",
        "count": "0"
    },
    {
        "timestamp": "2020-11-27T22:36:15.000Z",
        "count": "4"
    },
    {
```

7.2 Error Response - Arrival Rate Range Does Not Exist

Title	Arrival Rate API - Arrival Rate Range Does Not Exist.
Description	A request sent containing an arrival rate range that does not exist should return an error response.
Precondition	Tables should be created with meaningful records.
Test Steps	1. Start the Backend Server. 2. Send an Arrival Rate request to the backend by using Postman: Method: GET Path: http://localhost:3000/company/arrival_rate?queue_id=QUEUE12345&from=2020-11-26T10%3A00%3A00%2B08%3A00&duration=1 3. Receive a 404 Not Found from Postman.
Expected Result	Receive a 404 Not Found from Postman and error message indicating the arrival rate does not exist.

Evidence	

GET http://localhost:3000/company/arrival_rate?queue_id=QUEUE12345&from=2020-11-26T10%3A00%3A00%2B08%3A00&duration=1

Query Params

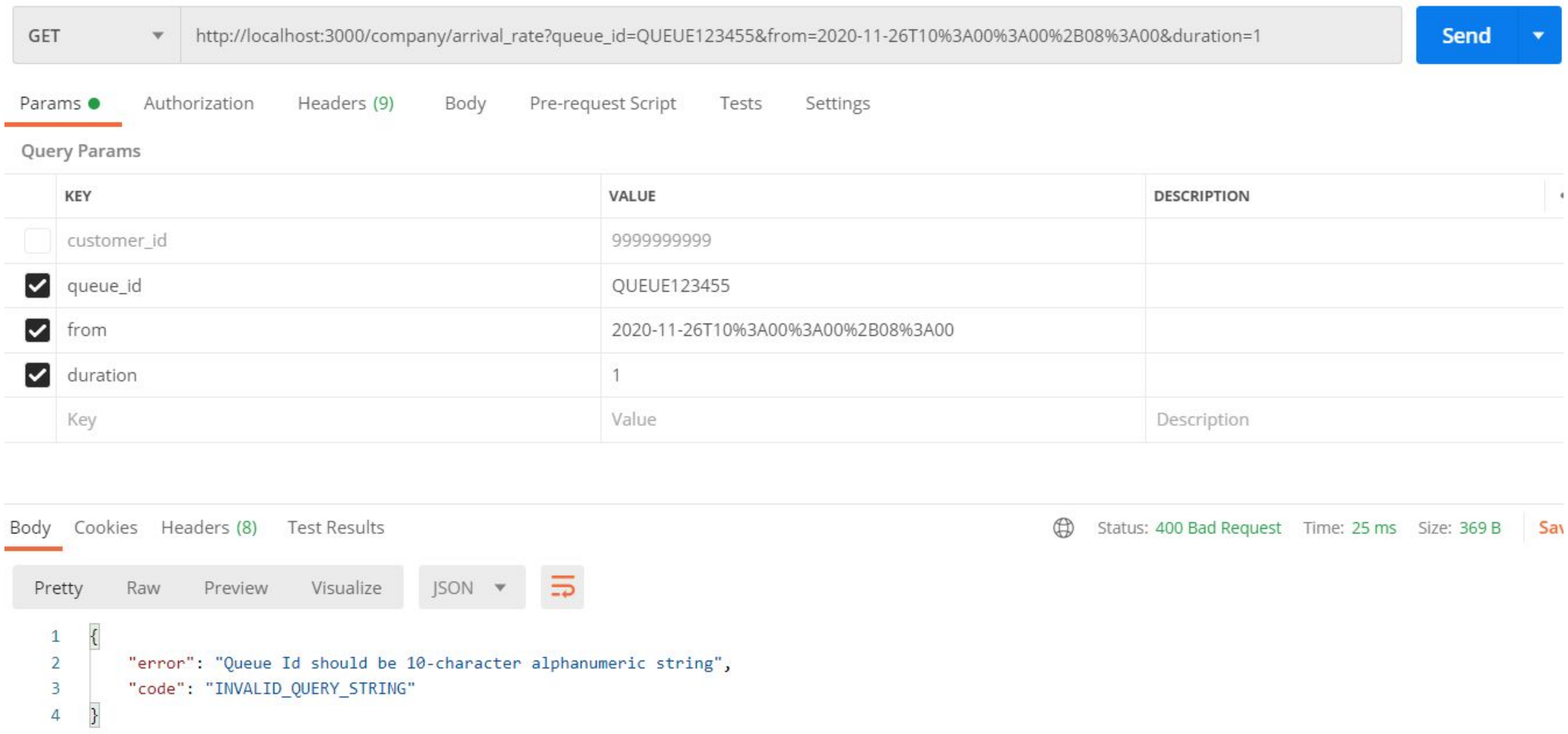
	KEY	VALUE	DESCRIPTION
□	customer_id	9999999999	
☑	queue_id	QUEUE12345	
☑	from	2020-11-26T10%3A00%3A00%2B08%3A00	
☑	duration	1	
	Key	Value	Description

Status: 404 Not Found Time: 417 ms Size: 339 B

```
{
    "error": "The arrival rate does not exist ",
    "code": "UNKOWN_TIME"
}
```

7.3 Error Response - Queue Id Invalid

Title	Arrival Rate API - Queue Id Invalid
Description	A request sent containing an invalid Queue Id should return an error response.
Precondition	Tables should be created with meaningful records.
Test Steps	1. Start the Backend Server. 2. Send an Arrival Rate request to the backend by using Postman: Method: GET Path: http://localhost:3000/company/arrival_rate?queue_id=QUEUE123455&from=2020-11-26T10%3A00%3A00%2B08%3A00&duration=1 3. Receive a 400 Bad Request from Postman.
Expected Result	Receive a 400 Bad Request from Postman and error message indicating the Queue Id is invalid.

Evidence	

GET http://localhost:3000/company/arrival_rate?queue_id=QUEUE123455&from=2020-11-26T10%3A00%3A00%2B08%3A00&duration=1

Query Params

	KEY	VALUE	DESCRIPTION
□	customer_id	9999999999	
☑	queue_id	QUEUE123455	
☑	from	2020-11-26T10%3A00%3A00%2B08%3A00	
☑	duration	1	
	Key	Value	Description

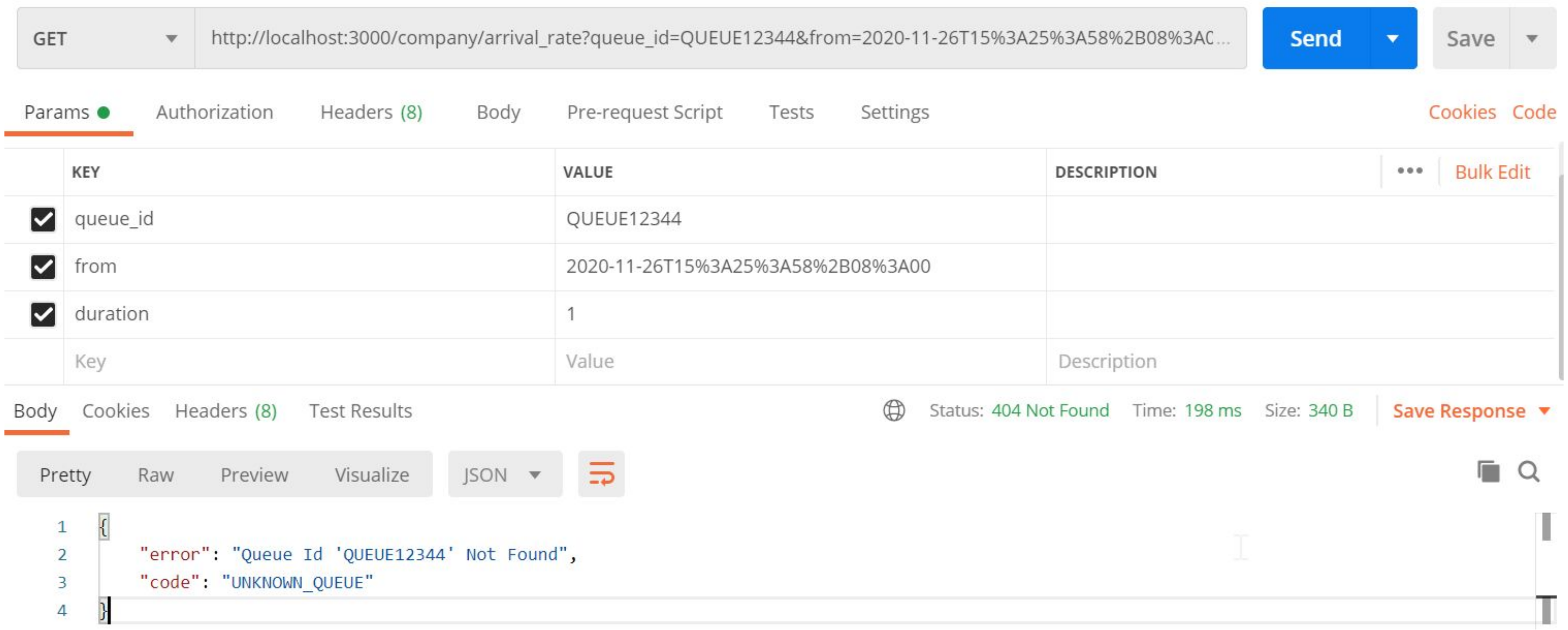
Status: 400 Bad Request Time: 25 ms Size: 369 B

```
{
    "error": "Queue Id should be 10-character alphanumeric string",
    "code": "INVALID_QUERY_STRING"
}
```

7.4 Error Response - Queue Id Not Found

Title	Arrival Rate API - Queue Id Not Found.
Description	An unsuccessful request due to non-existence Queue Id should return an error response.
Precondition	Tables should be created with meaningful records, queue_id(“QUEUE12344”) does not already exist.
Test Steps	1. Start the Backend Server. 2. Send an Arrival Rate request to the backend by using Postman: Method: GET Path: http://localhost:3000/company/arrival_rate?queue_id=QUEUE12344&from=2020-11-26T15%3A25%3A58%2B08%3A00&duration=1 3. Receive a 404 Not Found from Postman.
Expected Result	Receive a 404 Not Found from Postman and error message indicating the Queue Id is not found.

Evidence



GET http://localhost:3000/company/arrival_rate?queue_id=QUEUE12344&from=2020-11-26T15%3A25%3A58%2B08%3AC...

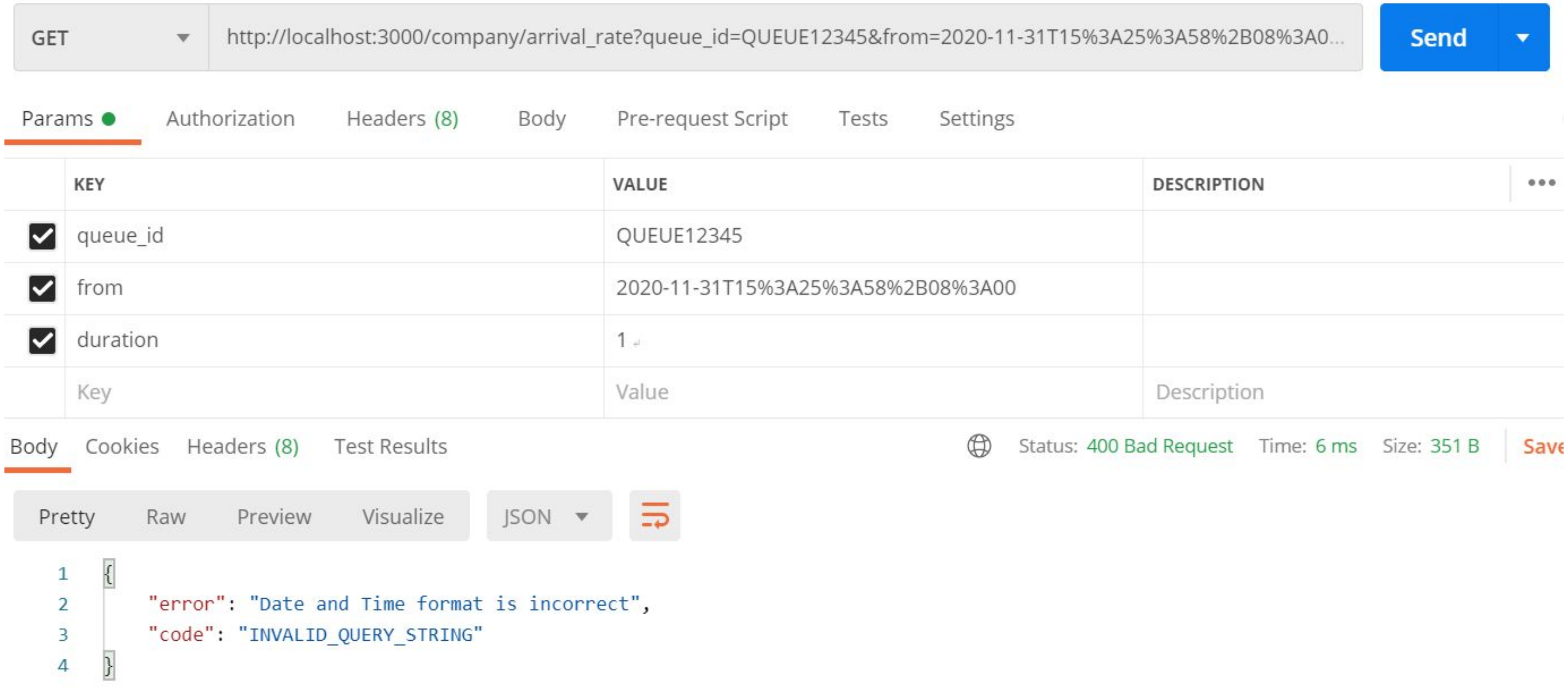
	KEY	VALUE	DESCRIPTION
☑	queue_id	QUEUE12344	
☑	from	2020-11-26T15%3A25%3A58%2B08%3A00	
☑	duration	1	
	Key	Value	Description

Status: 404 Not Found Time: 198 ms Size: 340 B

```
{
    "error": "Queue Id 'QUEUE12344' Not Found",
    "code": "UNKNOWN_QUEUE"
}
```

7.5 Error Response - Date and Time Format is Incorrect

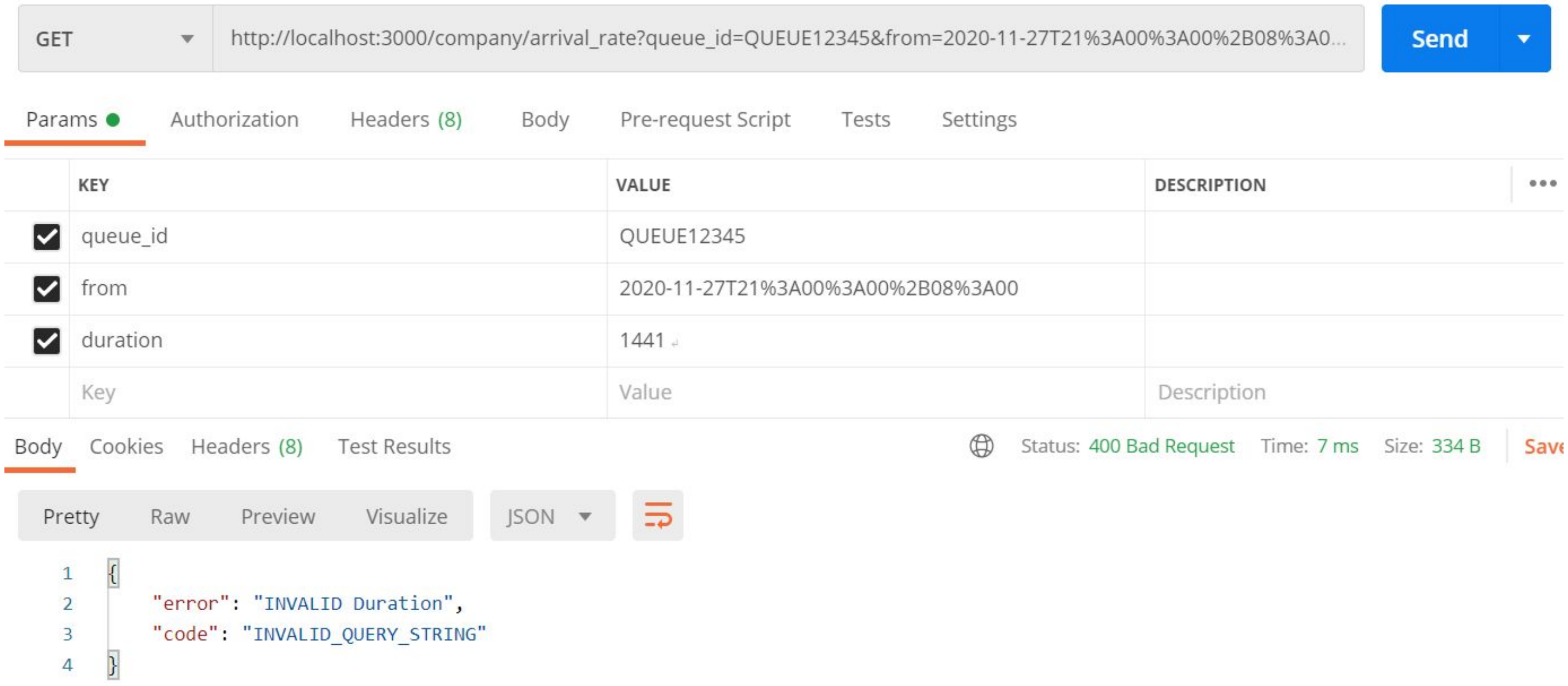
Title	Arrival Rate API - Date and Time format is Incorrect.
Description	An unsuccessful request due to incorrect datetime format should return an error response.
Precondition	Tables should be created with meaningful records.
Test Steps	1. Start the Backend Server. 2. Send an Arrival Rate request to the backend by using Postman: Method: GET Path: http://localhost:3000/company/arrival_rate?queue_id=QUEUE12345&from=2020-11-31T15%3A25%3A58%2B08%3A00&duration=1 3. Receive a 400 Bad Request from Postman.
Expected Result	Receive a 400 Bad Request from Postman and error message indicating the date and time format is invalid.(2020/11/31 is not a valid date)
Evidence	

	
	GET http://localhost:3000/company/arrival_rate?queue_id=QUEUE12345&from=2020-11-31T15%3A25%3A58%2B08%3A0...
	 Send
	Params
	Authorization
	Headers (8)
	Body
	Pre-request Script
	Tests
	Settings
	KEY | VALUE | DESCRIPTION
	☑ queue_id | QUEUE12345
	☑ from | 2020-11-31T15%3A25%3A58%2B08%3A00
	☑ duration | 1
	Key | Value | Description
	 Body
	Cookies
	Headers (8)
	Test Results
	Status: 400 Bad Request Time: 6 ms Size: 351 B
	 Save
	Pretty
	Raw
	Preview
	Visualize
	JSON
	{
	"error": "Date and Time format is incorrect",
	"code": "INVALID_QUERY_STRING"
	}

7.6 Error Response - Duration Invalid

Title	Arrival Rate API - Duration Invalid.
Description	A request sent containing an invalid duration should return an error response.
Precondition	Tables should be created with meaningful records.
Test Steps	1. Start the Backend Server. 2. Send an Arrival Rate request to the backend by using Postman: Method: GET Path: http://localhost:3000/company/arrival_rate?queue_id=QUEUE12345&from=2020-11-27T21%3A00%3A00%2B08%3A00&duration=1441 3. Receive a 400 Bad Request from Postman.
Expected Result	Receive a 400 Bad Request from Postman and error message indicating the duration is invalid.

Evidence



http://localhost:3000/company/arrival_rate?queue_id=QUEUE12345&from=2020-11-27T21%3A00%3A00%2B08%3A0...

	KEY	VALUE	DESCRIPTION
☑	queue_id	QUEUE12345	
☑	from	2020-11-27T21%3A00%3A00%2B08%3A00	
☑	duration	1441	
	Key	Value	Description

Status: 400 Bad Request Time: 7 ms Size: 334 B

```
{
    "error": "INVALID Duration",
    "code": "INVALID_QUERY_STRING"
}
```