

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 8

Aim : Cloudant database in IBM Cloud integration with a third-party application (Postman)

1. Install third party application postman on your system

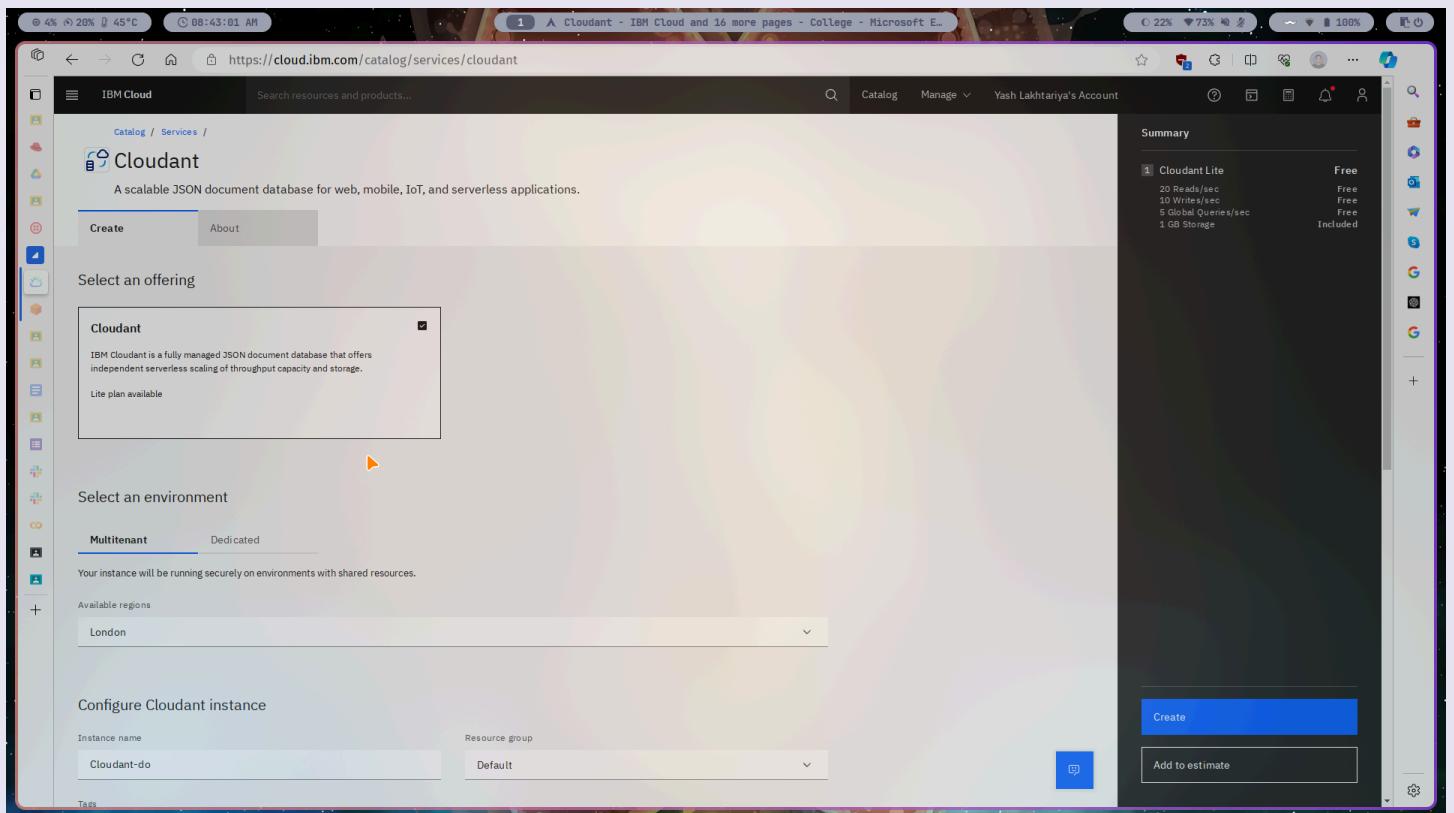
2. Create database for novel shop and document via creating an instance of Cloudant database.

4. Perform crud operations on database created using API's

5. Make PDF document for performed tasks

Steps and screenshots :

1. Create Cloudant service on IBM Cloud



Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 8

2. Select preferred region and choose **IAM and Legacy Credentials** as Authentication method and rest default options

The screenshot shows the IBM Cloud Catalog interface for creating a Cloudant instance. The URL is https://cloud.ibm.com/catalog/services/cloudant. On the left, there's a sidebar with icons for various services like Watson, Blockchain, and Cloudant. The main area has a search bar and a 'Catalog' tab. A dropdown menu shows 'Available regions' with 'London' selected. The configuration section includes fields for 'Instance name' (Cloudant-ysl), 'Resource group' (Default), and 'Tags' (Examples: env:dev, version-1). Under 'Authentication method', 'IAM and legacy credentials' is selected. The 'Plan' section offers two options: 'Lite' (selected) and 'Standard'. The 'Lite' plan is described as 'Full functionality for development and evaluation with a set capacity. Only one Lite plan instance per account.' It is marked as 'Free - Only in multi-tenant'. The 'Standard' plan is described as 'Granular control over provisioned throughput capacity allocated. Billing prorated hourly.' and starts at \$88.80/month. Capacity details at the bottom show 20 Reads per second, 10 Writes per second, 5 Queries per second, and 1 GB Storage included. On the right, a summary table shows the 'Cloudant Lite' plan with '20 Reads/sec', '10 Writes/sec', '5 Global Queries/sec', and '1 GB Storage Included'. A large blue 'Create' button is at the bottom right.

(Here, to send API requests through third party app (postman), we need to choose IAM and legacy credentials, not IAM only)

Name - Yash Lakhtariya

Enrollment number - 21162101012

Branch - CBA Batch - 61

EADC Practical 8

- Now, after the creation of service, the dashboard of cloudant can be launched via option on top right

The screenshot shows the IBM Cloud service details page for a Cloudant service named "Cloudant-ysl". The service is currently in a "Provision in progress" state. The "Overview" tab is selected, displaying deployment details such as CRN, location (London), and external endpoints. Under "Capacity", it indicates a current subscription to the "Lite" plan. The "Activity Tracker event types" dropdown is set to "Management". A prominent "Launch Dashboard" button is located in the top right corner of the main content area.

Name - Yash Lakhtariya

Enrollment number - 21162101012

Branch - CBA Batch - 61

EADC Practical 8

4. On dashboard, in databases tab, create a database and specify the name. The partitioning option should be ***Non-partitioned***

The screenshot shows a Microsoft Edge browser window displaying the Cloudant Dashboard at <https://60379ad1-b8cb-4933-bb03-b1bfa6df8a7e-bluemix.cloudant.com/dashboard.html>. The dashboard has a sidebar with various icons and a main area titled 'Databases'. In the top right corner, there is a 'Create Database' dialog box. The 'Database name' field contains 'demodb'. Under 'Partitioning', the radio button for 'Non-partitioned - recommended for most workloads' is selected. Below this, a dropdown menu 'Which should I choose?' provides information about non-partitioned databases. At the bottom right of the dialog are 'Cancel' and 'Create' buttons.

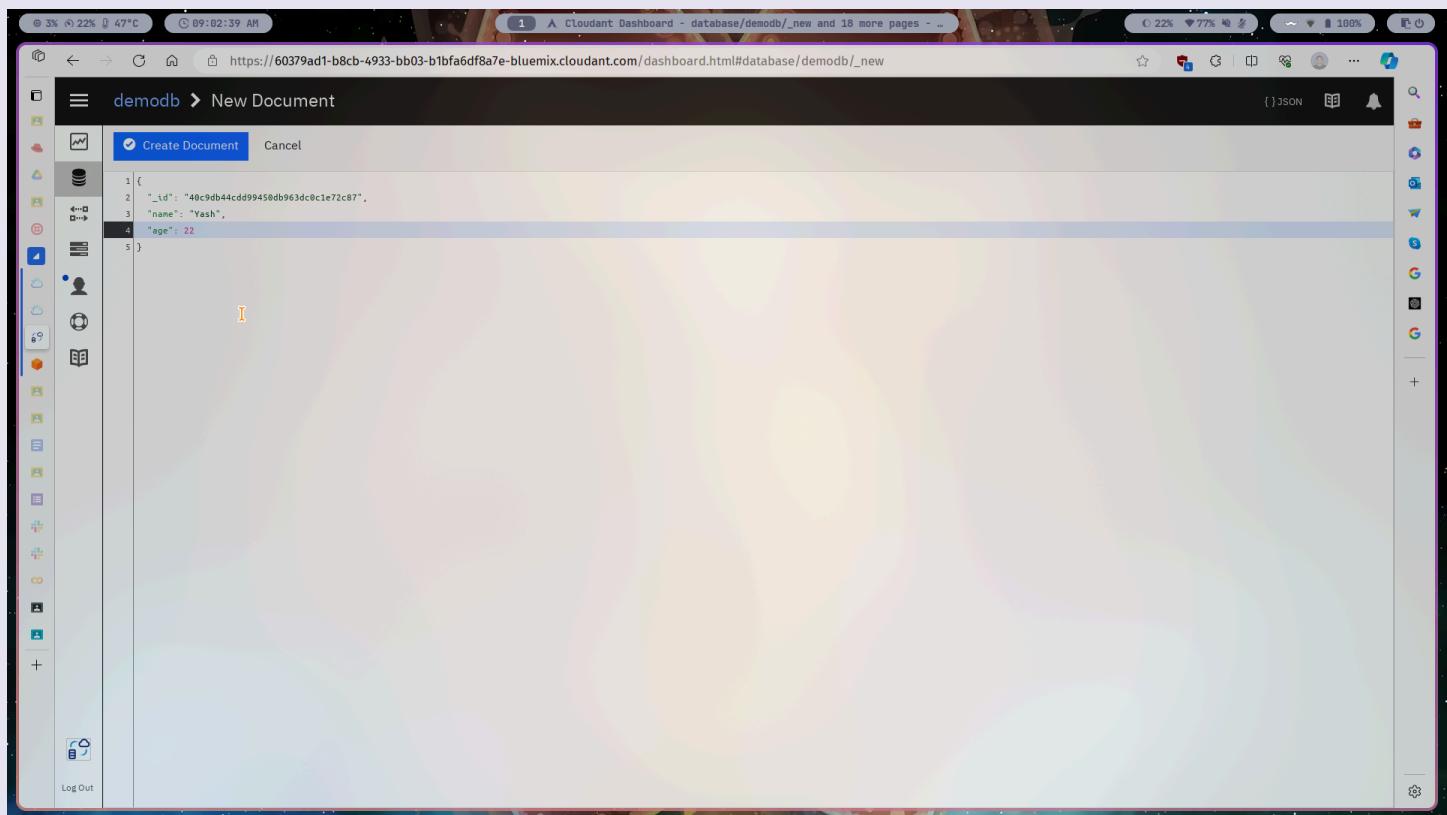
Name - Yash Lakhtariya

Enrollment number - 21162101012

Branch - CBA Batch - 61

EADC Practical 8

5. In the database created, create a document with some key-value pairs, as it is in JSON format



The screenshot shows a web browser window for the Cloudant Dashboard. The URL is https://60379ad1-b8cb-4933-bb03-b1bfa6df8a7e-bluemix.cloudant.com/dashboard.html#/database/demodb/_new. The page title is "Cloudant Dashboard - database/demodb/_new and 16 more pages -". The main content area displays a "Create Document" dialog with the following JSON code:

```
1 | {
2 |   "_id": "40c9db44cd99450db963dc8c1e72c87",
3 |   "name": "Yash",
4 |   "age": 22
5 | }
```

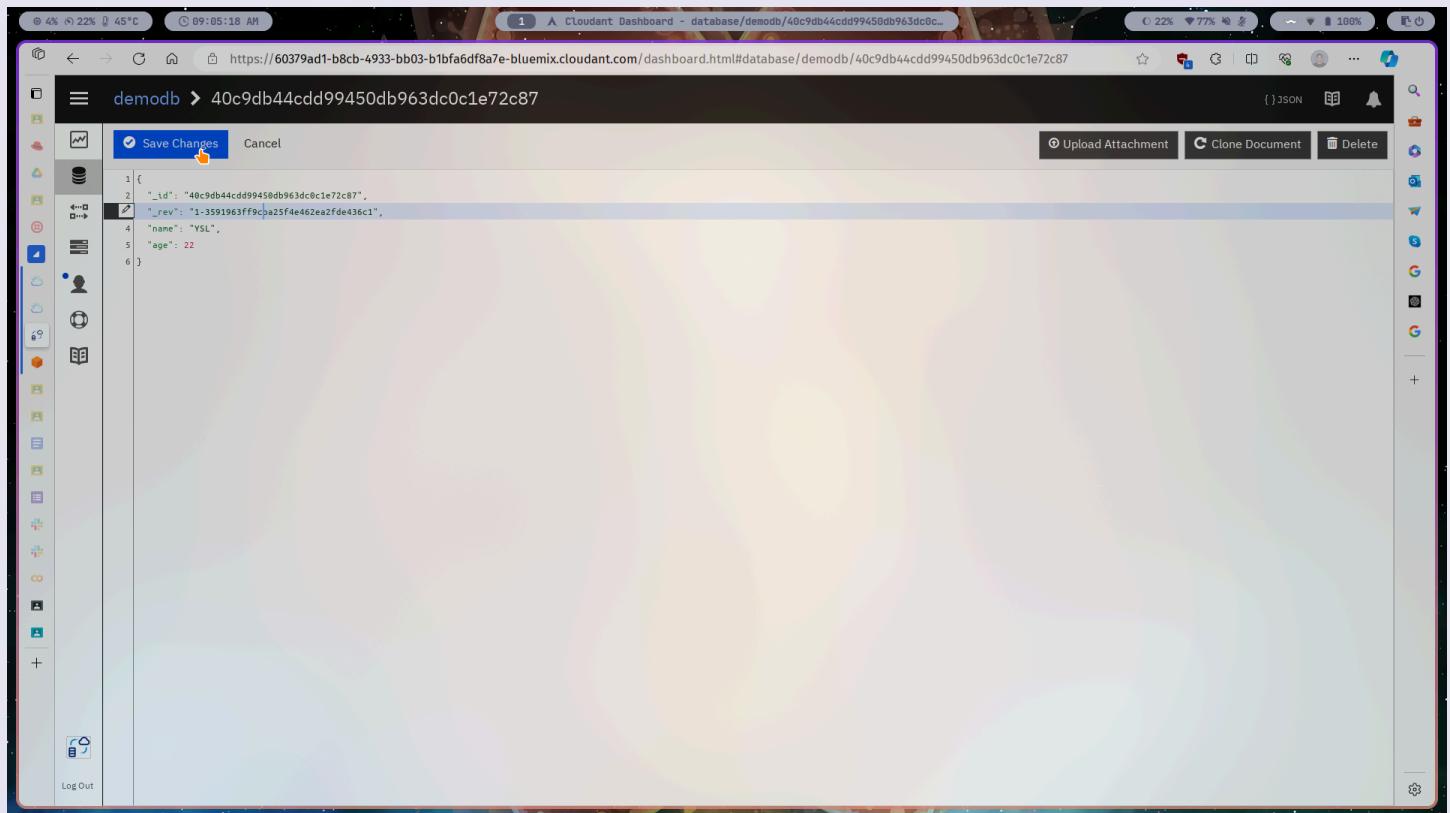
Name - Yash Lakhtariya

Enrollment number - 21162101012

Branch - CBA Batch - 61

EADC Practical 8

6. A revision ID is automatically generated which is used at many places, specifically, the first digit(s) states the number of times it is changed



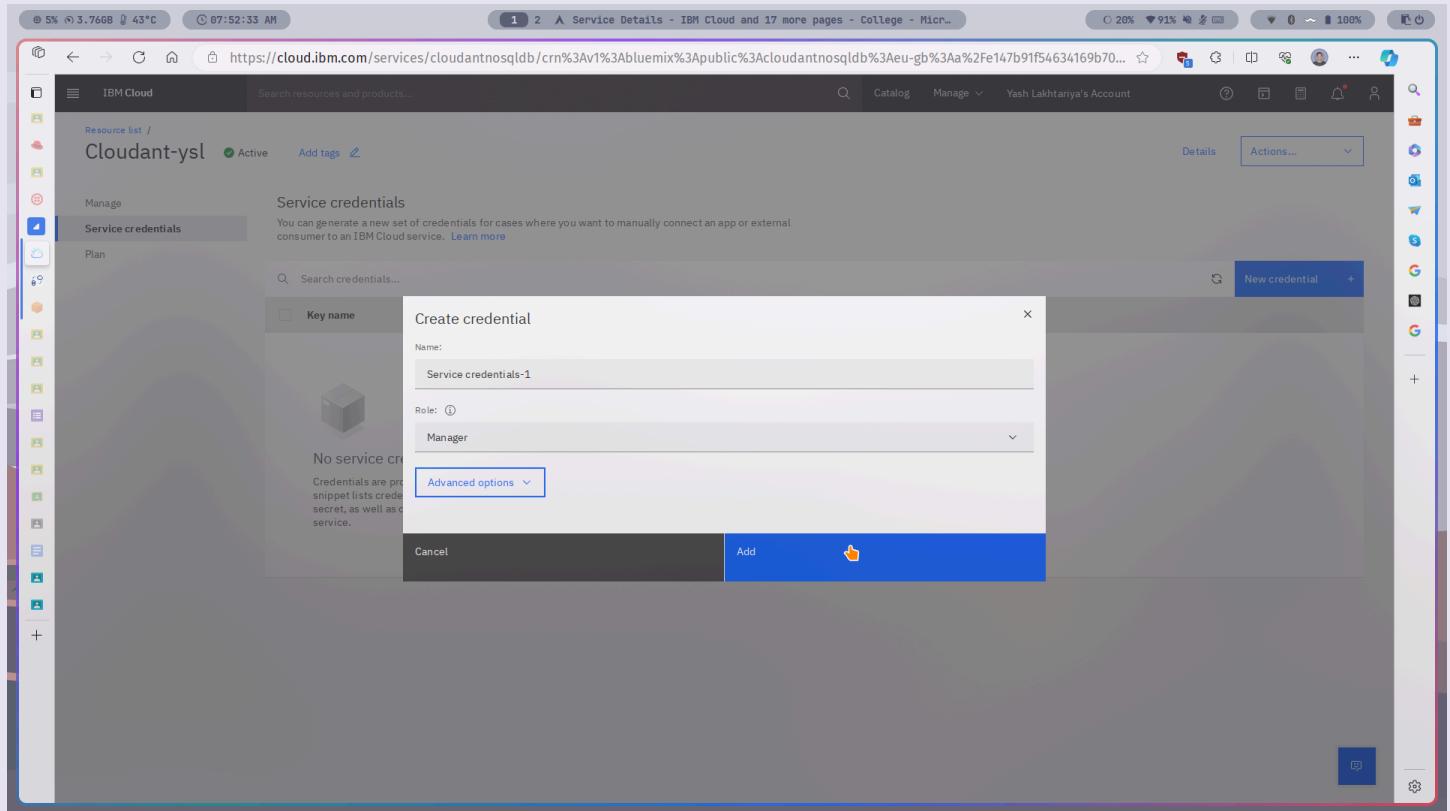
The screenshot shows a web browser window for the Cloudant Dashboard. The URL is <https://60379ad1-b8cb-4933-bb03-b1bfa6df8a7e-bluemix.cloudant.com/dashboard.html#/database/demodb/40c9db44cdd99450db963dc0c1e72c87>. The page displays a JSON document with the following content:

```
1 {
2   "_id": "40c9db44cdd99450db963dc0c1e72c87",
3   "_rev": "1-3591963ff9c3a25f4e462ea2fde436c1",
4   "name": "YSL",
5   "age": 22
6 }
```

The "Save Changes" button is highlighted with a yellow arrow. The dashboard interface includes a sidebar with various icons and a toolbar with "Upload Attachment", "Clone Document", and "Delete" buttons.

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 8

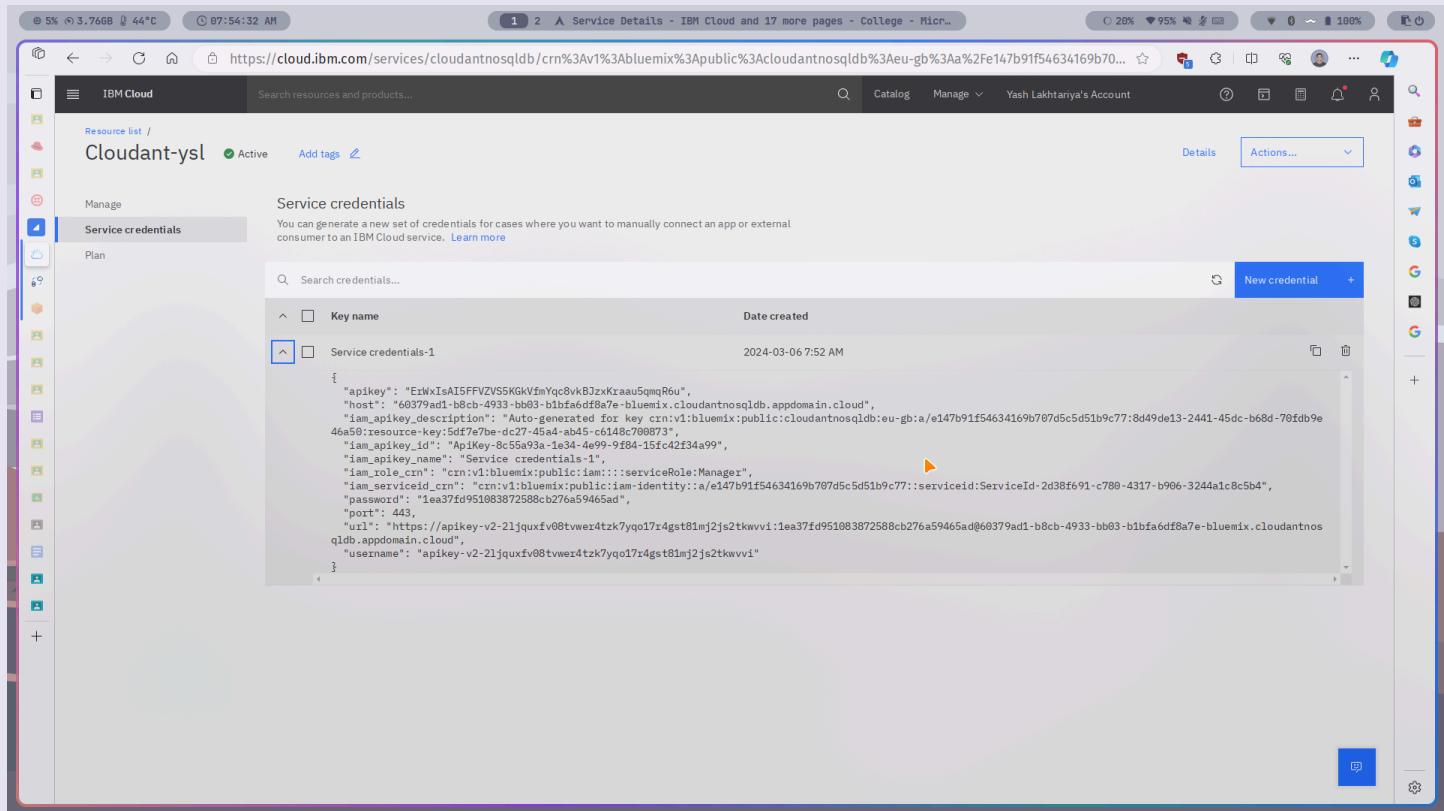
7. In IBM Cloudant service, now create credentials in Manage Credentials tab



The screenshot shows the IBM Cloudant service details page. On the left, there's a sidebar with various icons and a search bar. The main area displays the service name 'Cloudant-ysl' and its status as 'Active'. Below this, under 'Manage', is the 'Service credentials' section. A modal window titled 'Create credential' is open, prompting for a 'Name' (set to 'Service credentials-1') and a 'Role' (set to 'Manager'). At the bottom of the modal are 'Cancel' and 'Add' buttons, with a hand cursor hovering over the 'Add' button.

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 8

**8. Check the credentials and copy the same, the URL with format
username:password@host is mostly used to send requests via postman**



The screenshot shows the IBM Cloud Service Details page for a service named "Cloudant-ysl". The "Service credentials" tab is selected. A table lists a single credential entry:

Key name	Date created
Service credentials-1	2024-03-06 7:52 AM

The credential details are as follows:

```
{  
  "apikey": "EiWxIsAI5FFVZV55KGkVfmYqc8vkBJzxKraau5qmqr6u",  
  "host": "60379ad1-b8cb-4933-bb03-b1bfa6df0a7e-bluemix.cloudantnosqldb.appdomain.cloud",  
  "iam_apikey_description": "Auto-generated for key crn:v1:bluemix:public:cloudantnosqldb:eu-gb:a/e147b91f54634169b707d5c5d51b9c77:8d49de13-2441-45dc-b68d-70fdb9e46a59:resource-key:5df7e7be-dc27-45ad-ab45-c6148c708873",  
  "iam_apikey_id": "ApiKey-8c55a93a-1e34-4e99-f84-15fc42f34a99",  
  "iam_apikey_name": "Service credentials-1",  
  "iam_role_crn": "crn:v1:bluemix:public:iam::::serviceRole:Manager",  
  "iam_serviceid_crn": "crn:v1:bluemix:public:iam:identity:a/e147b91f54634169b707d5c5d51b9c77::serviceid:ServiceId-2d38f691-c780-4317-b906-3244a1c8c5b4",  
  "password": "1ea37fd951083872588c276a59465ad",  
  "port": 443,  
  "url": "https://apikey-v2-21jquxfv08tvwer4tzk7yqo17i4gst81mj2js2tkwvv1:1ea37fd951083872588cb276a59465ad@60379ad1-b8cb-4933-bb03-b1bfa6df0a7e-bluemix.cloudantnosqldb.appdomain.cloud",  
  "username": "apikey-v2-21jquxfv08tvwer4tzk7yqo17r4gst81mj2js2tkwvv1"  
}
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 8

9. Now, using Postman application, send PUT request to the URL/databasename to create a new database

The screenshot displays two windows side-by-side. On the left is the IBM Cloud Service Credentials interface, showing a list of credentials for the 'Cloudant-ysl' service. One credential, 'Service credentials-1', is listed with a creation date of 2024-03-06 7:52 AM. On the right is the Postman application, which is sending a PUT request to the URL `https://apikey-v2-2|jquxfv08tvwer4tzk7yqo17r4gst81mj2js2tkwvv1...``. The response body shows a JSON object with three fields: 1, 2, and 3, all containing the value "ok": true.

IBM Cloud Service Credentials:

- Cloudant-ysl (Active)
- Service credentials-1 (Created: 2024-03-06 7:52 AM)

Postman Response Body:

```
1: "ok": true
2: "ok": true
3: "ok": true
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 8

10. Check on the dashboard if the database is created successfully

The screenshot displays two browser windows side-by-side. The left window shows the IBM Cloudant Dashboard with a sidebar containing 'Monitoring', 'Databases', 'Replication', 'Active Tasks', 'Account', 'Support', and 'Documentation'. The main area is titled 'Databases' and lists two databases: 'demodb' (23 bytes, 1 doc, No) and 'yslnovel' (0 bytes, 0 docs, No). The right window shows a Postman API request for creating a database. The URL is `https://apikey-v2-2ljquxfv08tvwer4tzk7yqo17r4gst81mj2s2tkwvv1:ea37fd951083872588cb276a59465ad@60379ad1-b8cb-48`. The method is PUT, and the response status is 201 Created. The response body is a JSON object with a single key-value pair: `{"ok": true}`.

Name - Yash Lakhtariya

Enrollment number - 21162101012

Branch - CBA Batch - 61

EADC Practical 8

11. Now, make a GET request with URL/_all_dbs to get the list of existing databases

The screenshot displays two windows side-by-side. On the left is the IBM Cloudant web interface showing a list of databases: 'demodb' and 'yslnovel'. On the right is a Postman API client window showing a successful GET request to the URL `https://apikey-v2-2ljquxfv08tvwer4tzk7yqo17r4gst81mj2js2tkwvv1...`. The response body contains the list of databases: ["demodb", "yslnovel"].

Databases

Name	Size	# of Docs	Partitioned	Actions
demodb	23 bytes	1	No	[Edit, Lock, Delete]
yslnovel	0 bytes	0	No	[Edit, Lock, Delete]

Postman API Request:

```
GET https://apikey-v2-2ljquxfv08tvwer4tzk7yqo17r4gst81mj2js2tkwvv1:1ea37fd951083872588cb276a59465ad@60379ad1-b8cb-4933-8f3d-1a333e333333/_all_dbs
```

Body:

```
[ "demodb", "yslnovel" ]
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 8

12. Now, make a GET request to the URL/database to get the specific database's details

The screenshot displays two windows side-by-side. On the left is the IBM Cloudant interface, specifically the 'Databases' section. It shows two databases: 'demodb' (23 bytes, 1 doc, No) and 'yslnovel' (0 bytes, 0 docs, No). On the right is a Postman application window showing a successful GET request to the same database endpoint. The response body is a JSON object containing detailed database metadata, including fields like 'update_seq', 'db_name', 'purge_seq', 'size_mb', 'file', 'external', 'active', 'props', 'doc_del_count', 'doc_count', 'disk_format_version', 'compact_running', 'cluster', and various 'q', 'n', 'w', and 'r' values.

```
1 "update_seq":  
2 "2-g1AAAeDeJy11c1NwzAYgGGrReJIN4AbpwQ7cRznRDeAdSCff6iqNEGoPcMgsAFsABvABrAbbAAbuUfGMs730jgHR4qU7SH1y  
wkYtu2M12x-e9WnfuhUn_Y1b02w7f5loupi1amQe7sqGFri8-2kgUI-UPJ396ua5x1VicPy2Usaz85uR3L2MFd0kjtpIsPhy1SN  
dHLL1YNSjK88527721MCDSYY-Sdb5x862Juac5TgZls0k1BFZ1powJpuEjeHcqkOgUsKgrifHxz1i6lmgczPXNvxuXNbmfV1j  
GgkhsOrPq_o8ecI8jJpoguMKuLYK98NhQhTun8WoWw2dJUOM",  
3 "instance_start_time": "1709263715",  
4 "db_name": "demodb",  
5 "purge_seq": 0,  
6 "size_mb": {  
7     "file": 174951,  
8     "external": 23,  
9     "active": 1360  
10 },  
11 "props": {},  
12 "doc_del_count": 0,  
13 "doc_count": 1,  
14 "disk_format_version": 8,  
15 "compact_running": false,  
16 "cluster": {  
17     "q": 16,  
18     "n": 3,  
19     "w": 2,  
20     "r": 2  
21 },  
22 }
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 8

13. To add details, make a POST request to URL/database, with json details in body

The screenshot shows the Postman application interface. At the top, the URL is https://apikey-v2-2ljquxfv08tvwer4tzk7yqo17r4gst81mj2js2tkwvv1... and the status bar indicates 6% battery, 5.48GB RAM, 45°C, 88:46:48 AM, 28% battery, 95% signal, and 100% network. The main window displays a POST request to https://apikey-v2-2ljquxfv08tvwer4tzk7yqo17r4gst81mj2js2tkwvv1... with the following JSON body:

```
1 "id": "novel_004",
2 "name": "The Merry Adventures of Robin Hood",
3 "author": "Howard Pyle",
4 "year": 1883
```

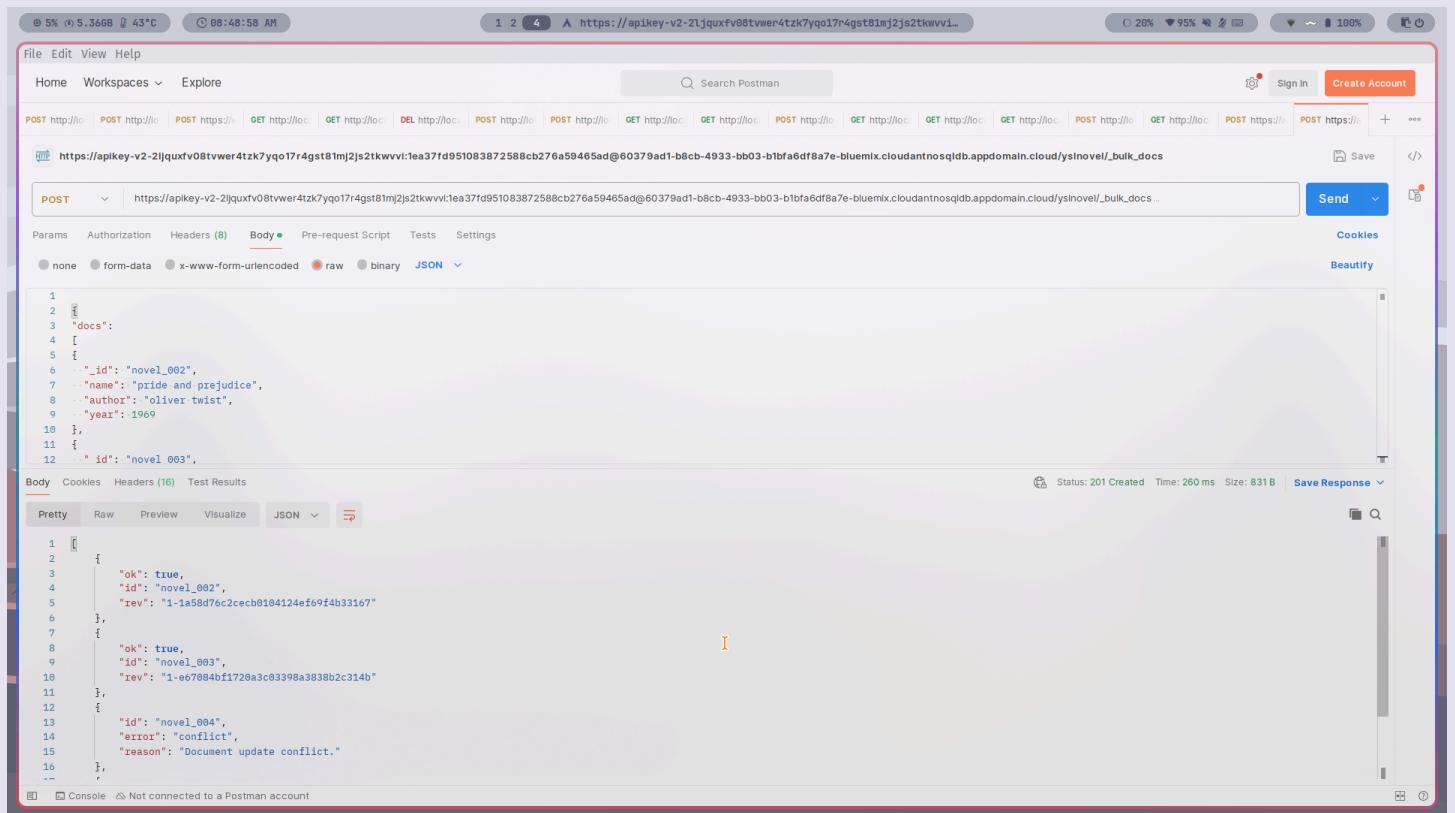
The response tab shows a 201 Created status with a response time of 872 ms and a size of 699 B. The response body is:

```
1 {
2   "ok": true,
3   "id": "novel_004",
4   "rev": "1-fd3f8e0f831970ce76dea0cc9910e222"
5 }
```

At the bottom, it says "Console" and "Not connected to a Postman account".

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 8

14. _bulk_docs can be added after database name for dealing with all docs in the database.



The screenshot shows a Postman interface with the following details:

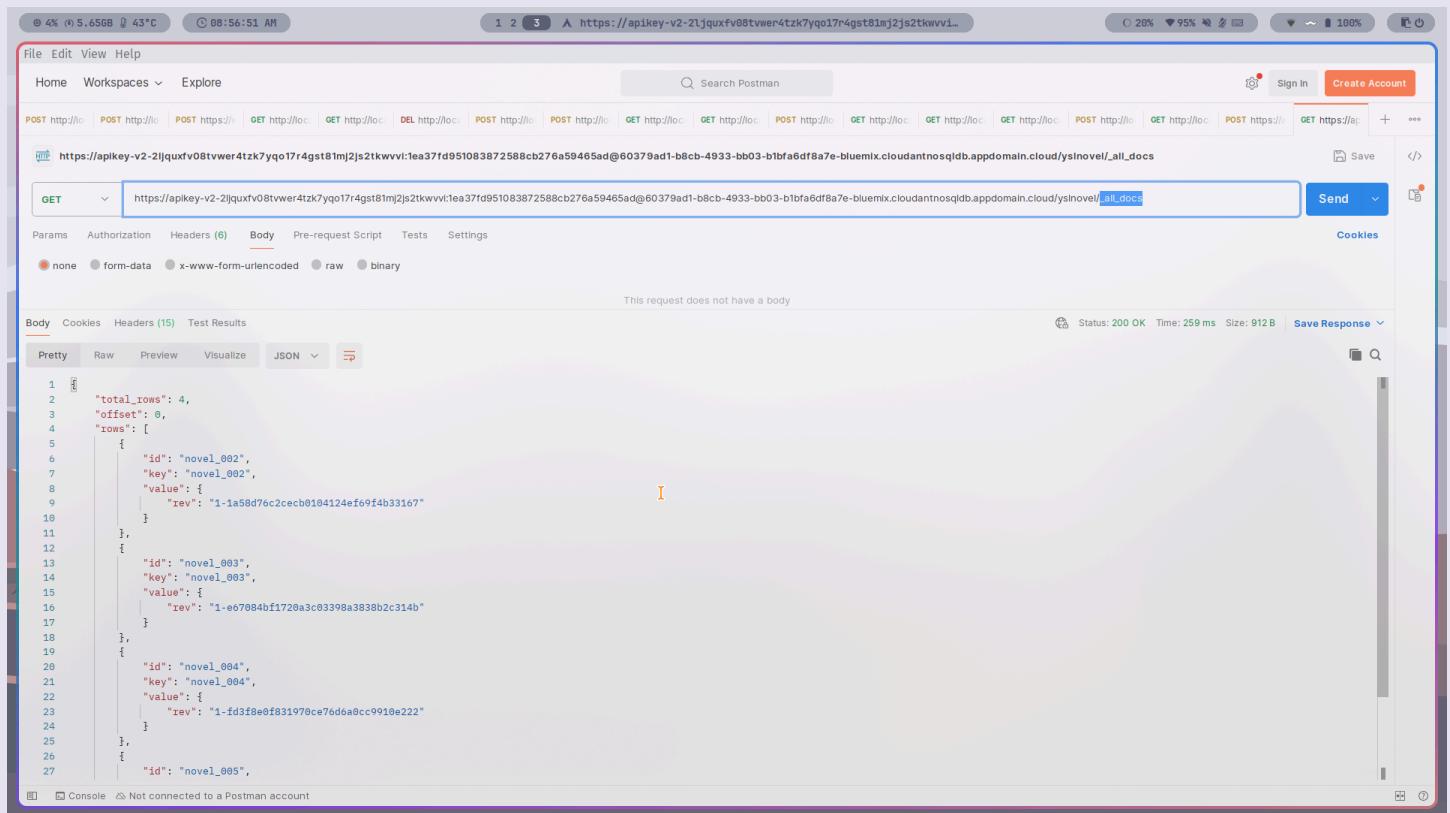
- URL:** https://apikey-v2-2ljqxv08tvwer4tzk7yqo17r4gst81mj2js2tkwvv1...
https://apikey-v2-2ljqxv08tvwer4tzk7yqo17r4gst81mj2js2tkwvv1...:1ea37fd951083872588cb276a59465ad@60379ad1-b8cb-4933-bb03-b1bfa6df8a7e-bluemix.cloudantnosqldb.appdomain.cloud/yslnovel/_bulk_docs
- Method:** POST
- Body (JSON):**

```
1
2 [
3   "docs": [
4     [
5       {
6         "_id": "novel_002",
7         "name": "pride and prejudice",
8         "author": "oliver twist",
9         "year": 1909
10      },
11      {
12        "_id": "novel_003",
13        "ok": true,
14        "id": "novel_002",
15        "rev": "1-1a58d76c2cecb0104124ef694b33167"
16      },
17      {
18        "ok": true,
19        "id": "novel_003",
20        "rev": "1-e67084bf1720a3c03398a3838b2c314b"
21      },
22      {
23        "id": "novel_004",
24        "error": "conflict",
25        "reason": "Document update conflict."
26      }
27    ]
28  ]
29 ]
```

- Response Status:** 201 Created
- Time:** 260 ms
- Size:** 831 B

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 8

15. If GET request is sent to database/_all_docs then it returns the doc's details like ID and all without including the actual document



The screenshot shows a Postman interface with the following details:

- URL:** https://apikey-v2-2ljqxv08tvver4t2k7yqo17r4gst81mj2js2tkwvv1...
- Method:** GET
- Body:** This request does not have a body.
- Headers:** Authorization (Bearer [REDACTED]), Content-Type (application/json)
- Response Status:** 200 OK
- Response Time:** 259 ms
- Response Size:** 912 B
- Response Body (Pretty JSON):**

```
1
2   "total_rows": 4,
3   "offset": 0,
4   "rows": [
5     {
6       "id": "novel_002",
7       "key": "novel_002",
8       "value": {
9         "rev": "1-1a58d76c2cecb0104124ef69f4b33167"
10      }
11    },
12    {
13      "id": "novel_003",
14      "key": "novel_003",
15      "value": {
16        "rev": "1-e67084bf1720a3c03398a3838b2c314b"
17      }
18    },
19    {
20      "id": "novel_004",
21      "key": "novel_004",
22      "value": {
23        "rev": "1-fd3f8e0f831970ce76d6a0cc9910e222"
24      }
25    },
26    {
27      "id": "novel_005",
28    }
29  ]
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 8

16. But, if `include_docs=true` is used as URN, then it also returns the actual document details

The screenshot shows a Postman interface with the following details:

- Request URL:** `https://apikey-v2-2ljquxfv08tvwer4tzk7yqo17r4gst81mj2js2tkwvv1...`
- Method:** GET
- Body:** `?all_docs?include_docs=true`
- Response Status:** 200 OK
- Response Time:** 937 ms
- Response Size:** 1.41 KB
- Response Body (Pretty JSON):**

```
1
2   "total_rows": 4,
3   "offset": 0,
4   "rows": [
5     {
6       "id": "novel_002",
7       "key": "novel_002",
8       "value": {
9         "rev": "1-1a58d76c2cecb0104124ef69f4b33167"
10      },
11      "doc": {
12        "_id": "novel_002",
13        "_rev": "1-1a58d76c2cecb0104124ef69f4b33167",
14        "name": "pride and prejudice",
15        "author": "oliver twist",
16        "year": 1969
17      }
18    },
19    {
20      "id": "novel_003",
21      "key": "novel_003",
22      "value": {
23        "rev": "1-e67084bf1720a3c03398a3838b2c314b"
24      },
25      "doc": {
26        "_id": "novel_003",
27        "_rev": "1-e67084bf1720a3c03398a3838b2c314b",
28      }
29    }
30  ]
31 }
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 8

17. To get the specific document's details, enter its ID after URL/dbname/docID and make a GET request

The screenshot shows the Postman application interface. At the top, there are various status icons and a search bar labeled "Search Postman". Below the header, a navigation bar includes "File", "Edit", "View", and "Help". A toolbar with various HTTP method buttons (POST, PUT, DELETE, etc.) is visible. The main workspace displays a GET request to the URL `https://apikey-v2-2ljqquxfv08tvwer4tzk7yqo17r4gst81mj2js2tkwvv1...`. The "Body" tab is selected, showing the response body in JSON format:

```
1
2   "_id": "novel_003",
3   "_rev": "1-e67084bf1f720a3c03398a3838b2c314b",
4   "name": "The Meiry go round",
5   "author": "Howard",
6   "year": 2000
7 }
```

At the bottom right of the response pane, it says "Status: 200 OK Time: 757 ms Size: 674 B Save Response". The bottom left of the interface shows "Console" and "Not connected to a Postman account".

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 8

18. To update details, make a PUT request on same URL, but include previous revision ID in the json in body

The screenshot displays a macOS desktop environment. On the left, a browser window shows a MongoDB interface for a collection named 'novel_003'. The document details are as follows:

```
_id: "novel_003",
_rev: "1-e67084bf1720a3c03398a3838b2c314b",
name: "The Merry go round",
author: "Howard",
year: 2000
```

On the right, a Postman application window is open, showing a PUT request to the following URL:

```
https://apikey-v2-2jquxfv08tvwer4tzk7yqo17r4gst81m[2]s2tkwwv1ea37fd95...
```

The request parameters and body are set to:

```
Params: @60379ad1-b5cb-4933-bb03-b10fa6df8a7e-blueMix.cloudantnosqldb.appdomain.cloud/yslnovel/novel_003
Auth: Basic auth (disabled)
Body: raw JSON
```

The JSON body of the request is:

```
{
  "_id": "novel_003",
  "_rev": "1-e67084bf1720a3c03398a3838b2c314b",
  "name": "Bhagavad Gita Asitish",
  "author": "Sila Prabhupada",
  "year": 1972
}
```

The response status is 409 Conflict, with the reason being:

```
{"error": "conflict", "reason": "Document update conflict."}
```

Name - Yash Lakhtariya

Enrollment number - 21162101012

Branch - CBA Batch - 61

EADC Practical 8

19. Now, the details will be updated in Cloudant database

```
1 {
2   "_id": "novel_003",
3   "_rev": "2-a1061ddadbf642226db6b4001be47c7",
4   "name": "Bhagavad Gita Asitis",
5   "author": "Sriла Прабхупада",
6   "year": 1972
7 }
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 8

20. Now, to delete a record, same URL but with rev={revID} is required for DELETE request

The screenshot displays two windows side-by-side. On the left is a MongoDB interface showing a document in a collection named 'novel_005'. The document has the following JSON structure:

```
1: {  
2:   "_id": "novel_005",  
3:   "_rev": "1-fb4f4760ae33491b08a8b71246ebe842",  
4:   "name": "great work",  
5:   "author": "anonymous",  
6:   "year": 1945  
7: }
```

On the right is a Postman API client window. It shows a DELETE request to the URL `https://apikey-v2-2jquxfv08tvwer4tzk7yqo174gst81m[2]s2tkwwvl:1ea37fd95...0@60379ad1-b5cb-4933-b003-b1fbfa6d8f8a7e-blueomicloudanthonsqldb.appdomain.cloud/novel_005?rev=1-fb4f4760ae33491b08a8b71246ebe842`. The 'Query Params' section has 'rev' checked and its value is set to the same value as the '_rev' field in the MongoDB document. The response body shows a JSON object with an 'ok' key set to true.

```
1: {  
2:   "ok": true,  
3:   "id": "novel_005",  
4:   "rev": "2-4ad86448895735b1fe9bc8f81246ebe842"  
5: }
```

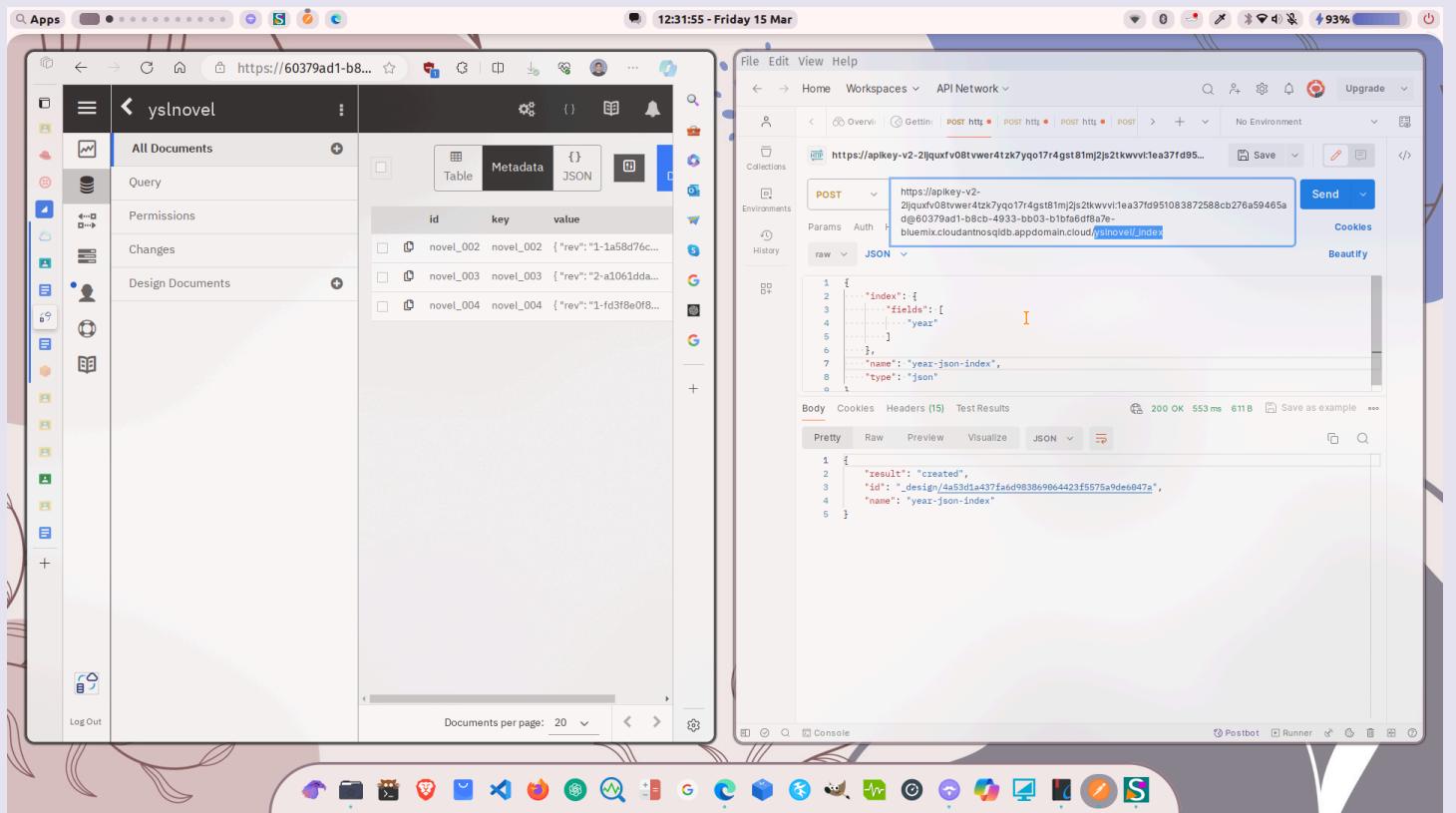
Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 8

21. The document will be deleted now from the database

The screenshot shows a desktop interface with two windows open. On the left is a web browser window titled 'yslnovel' displaying a list of documents. The table has columns 'id', 'key', and 'value'. There are four entries: novel_002, novel_003, novel_004, and novel_005. On the right is a Postman API client window showing a DELETE request to 'https://apikey-v2-2ijquxfv08tvwer4tzk7yqo174gst81mj2s2tkwvv1ea37fd95...'. The body of the request contains 'Key' and 'rev' parameters. The response shows a JSON object with 'ok': true, 'id': 'novel_005', and 'rev': '2-4a4d86448095735b1fe96c8f912543a2'.

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 8

22. Here, for performing queries, we can create an index file for the same which includes the field(s) to perform query on, while the URI will be URL/dbname/_index and POST request should be made



Name - Yash Lakhtariya

Enrollment number - 21162101012

Branch - CBA Batch - 61

EADC Practical 8

23. Now, for example if we want to filter data as per year, say year greater than 1864 and sort the data in ascending format, then the format of json body will be below with URL/dbname/_find as URI with POST request :

```
{  
    "selector": {  
        "year": {  
            "$gt": 1880  
        }  
    },  
    "fields": [  
        "_id",  
        "name",  
        "author",  
        "year"  
    ],  
    "sort": [  
        {  
            "year": "asc"  
        }  
    ]  
}
```

The index will also be visible on cloudant database

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
EADC Practical 8

The screenshot shows a dual-monitor setup. The left monitor displays a web browser window for a database named 'yslnovel'. The browser interface includes a sidebar with icons for file, database, and document management, and a main panel showing a table of documents with columns 'id' and 'key'. The table contains four rows: '_design/4a53d1a437...', 'novel_002', 'novel_003', and 'novel_004'. The right monitor displays a REST client window titled 'File Edit View Help'. It shows a POST request to 'https://apiv2-2ijquxfv08tvver4tzk7yqo174gst81m2j2s2tkwvv1ea37fd95108387...'. The request body is a JSON object:

```
1 {  
2   "selector": {  
3     "year": {  
4       "$gt": 1864  
5     }  
6   }  
7 }  
8  
9 {  
10   "docs": [  
11     {  
12       "_id": "novel_004",  
13       "name": "The Mezzy Adventures of Robin Hood",  
14       "author": "Howard Pyle",  
15       "year": 1883  
16     },  
17     {  
18       "_id": "novel_002",  
19       "name": "pride and prejudice",  
20       "author": "oliver twist",  
21       "year": 1969  
22     },  
23     {  
24       "_id": "novel_003",  
25       "name": "Bhagavad Gita Asitis",  
26       "author": "Srila Prabhupada",  
27       "year": 1972  
28     }  
29   ],  
30   "bookmark":  
31   "g2eAAACAAJkAA5zdGfydGtLeV9kb2NpZG8AAAABm922xvFMDAzaAJkAAhzdGfydGtLeWwAAABygAA87Rqag"  
32 }
```

The REST client also shows a preview of the response, which is a JSON object with 'docs' and 'bookmark' fields. The status bar at the bottom of the client window indicates a 200 OK response with 593 ms latency.