How to use this website

There are 10 units under UGC NET Paper 1. You click the dropdown, select the unit, then click the other dropdown for the year and select the year. In about 10 seconds, around 25 questions from that year will load. You can click the labelled options to check your answer. For units that have more than 25 questions, as you scroll down, you'll find the option to click 'next page' for the remaining questions.

Old question paper vs New question paper

Each year, UGC NET conducts two exams, and each exam session lasts about 5-6 days, with two shifts per day. For each shift, there is a different Paper 1 question paper; thus, for one exam session, there are a total of 12 unique question papers. Each question paper has 5 questions from each of the 10 sections. Because of this arrangement, it is better to have the preparation of all versions of the last 3 years' PYQ than to have the preparation of one question paper from each of the last 10 years. This website, following this approach, will periodically update questions in the above-mentioned order (and this is also why one year, e.g., 2025 June, can have 25+ questions in any one unit, even though one paper has only 5 questions from that unit).

Please use these questions as a guiding platform to understand the paper and pinpoint the repeating/unknown topics, rather than just solving the questions. This website is more for understanding what to read than actually solving the questions.

Why this website?

Higher Education/Research is a field that shouldn't be gatekept. While pyqs of paper 1 are available online for "free", they are always either incomplete, ex, missing shift 2, or available only upon registration, making them inaccessible to many. Even in places where they are available truly freely, like NDLI: UGC-NET General Paper on Teaching & Research Aptitude (Paper I) year-wise Question Papers and Syllabus, they are either non-downloadable or available in static form.

The point of this website is to make the resource available for bite-sized consumption, for either working professionals who want to read on the go or students who are stuck on one or two units and don't want to subscribe to whole Paper I modules.

Is this website free?

Yes. And I plan to keep it that way. While coaching institutes that have these pyqs have dedicated teams for their online module creation and maintenance, this is entirely handled

by a single person (me) with a non-technical/non-CS educational background.

As a scholar who has extensively relied on and used open-source products, from Anna's Archive to Wikipedia to Way Back Machine to free YouTube videos to get most of my education, I really am glad to contribute something meaningful to the open-source community myself. However, each of these open-source projects has real operational costs; failing to cover them, they face the threat of forever deletion or security attacks. To be part of the open source ecosystem, whether consumer or creator, is to acknowledge this part of the transaction as well.

While this is a passion project, I cannot skip over the fact that this project uses a live database (NeonDB), a live front-end and backend server (Vercel), both of which have operational costs, along with my countless days (around 6 months to be precise) spent creating this (from acquiring the pyqs to scanning them to making the website for displaying them). If you are a consumer of open source, would like to see open source thrive more, and would yourself like to be a part of it someday, you can start by supporting this or similar open source projects. Thanks!

How is this website made?

This section is for people who really want to know how this was made, probably DH students, or general enthusiasts. This section actually goes into the project's pipeline rather than the actual code for brevity. However, broadly, I have used the following tools - Python, Tesseract, Regex library, SQL, JSON, TSX, CSS, and SkLearn.

A short pipeline -

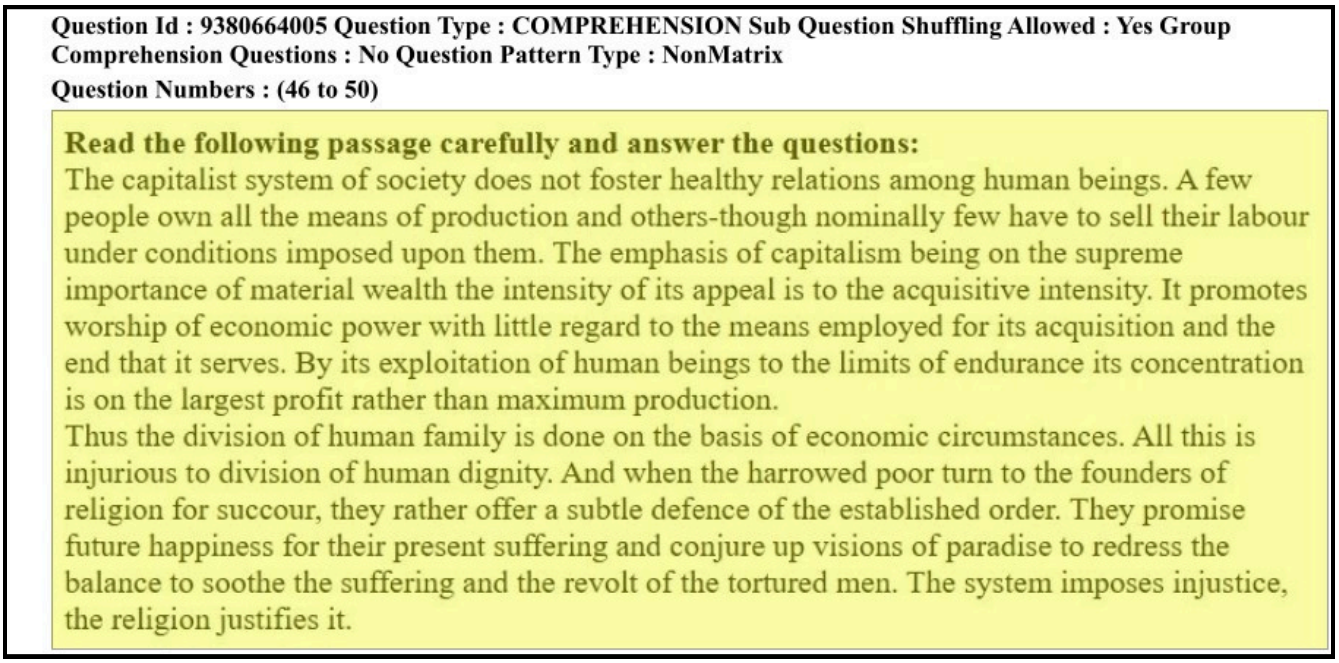
Get the PYQs → Do OCR on PYQs to extract questions and answers → Store the data in a displayable format (JSON) → Upload the data to a live database (NeonDB) → Write script for

the front-end and backend that talks to the database and displays the information on screen → Deploy on GitHub and Front-end server (Vercel).

A detailed synopsis -

Get the PYQs - Do a lot of digging on the internet and accumulate all the PYQs.

OCR and Extraction - This is the part that really is interesting. The biggest problem with the available PYQs (from ANY online source) is that the questions are embedded in the PDFs as images, not text, so you can't just copy and paste. For example, here, the entire highlighted part is an image, not a text.



Question Id : 9380664005 Question Type : COMPREHENSION Sub Question Shuffling Allowed : Yes Group Comprehension Questions : No Question Pattern Type : NonMatrix

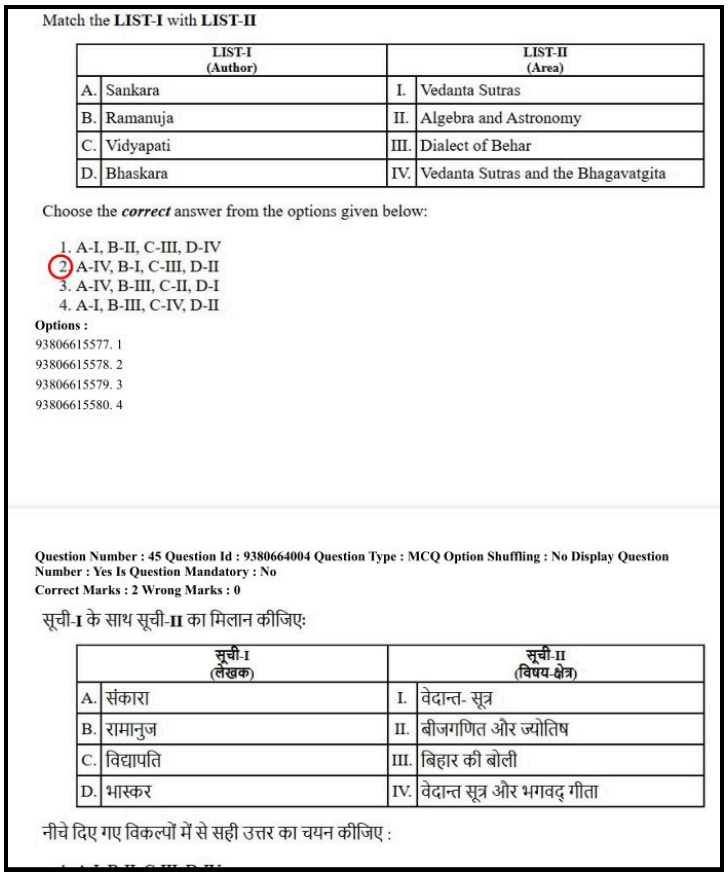
Question Numbers : (46 to 50)

Read the following passage carefully and answer the questions:

The capitalist system of society does not foster healthy relations among human beings. A few people own all the means of production and others-though nominally few have to sell their labour under conditions imposed upon them. The emphasis of capitalism being on the supreme importance of material wealth the intensity of its appeal is to the acquisitive intensity. It promotes worship of economic power with little regard to the means employed for its acquisition and the end that it serves. By its exploitation of human beings to the limits of endurance its concentration is on the largest profit rather than maximum production.

Thus the division of human family is done on the basis of economic circumstances. All this is injurious to division of human dignity. And when the harrowed poor turn to the founders of religion for succour, they rather offer a subtle defence of the established order. They promise future happiness for their present suffering and conjure up visions of paradise to redress the balance to soothe the suffering and the revolt of the tortured men. The system imposes injustice, the religion justifies it.

So I had to use an OCR engine, Tesseract, to extract the text from these images. But, two problems, first, OCR isn't perfect, so even after scanning, there is a lot of error correction to do, for example, OCR often confuses i for l and vice versa; second, the PYQ source I was using had alternate Hindi texts of each questions,



Match the **LIST-I** with **LIST-II**

	LIST-I (Author)		LIST-II (Area)
A.	Sankara	I.	Vedanta Sutras
B.	Ramanuja	II.	Algebra and Astronomy
C.	Vidyapati	III.	Dialect of Behar
D.	Bhaskara	IV.	Vedanta Sutras and the Bhagavatgita

Choose the ***correct*** answer from the options given below:

1. A-I, B-II, C-III, D-IV
2. A-IV, B-I, C-III, D-II
3. A-IV, B-III, C-II, D-I
4. A-I, B-III, C-IV, D-II

Options :

93806615577. 1
93806615578. 2
93806615579. 3
93806615580. 4

Question Number : 45 Question Id : 9380664004 Question Type : MCQ Option Shuffling : No Display Question Number : Yes Is Question Mandatory : No
Correct Marks : 2 Wrong Marks : 0

सूची-I के साथ सूची-II का मिलान कीजिए:

	सूची-I (लेखक)		सूची-II (विषय-क्षेत्र)
A.	संकारा	I.	वेदान्त- सूत्र
B.	रामानुज	II.	बीजगणित और ज्योतिष
C.	विद्यापति	III.	बिहार की बोली
D.	भास्कर	IV.	वेदान्त सूत्र और भगवद् गीता

नीचे दिए गए विकल्पों में से सही उत्तर का चयन कीजिए :

and running OCR on the whole file, the engine would try to approximate devnagri scripts to their closest english alphabet and give a lot of useless noisy data which I would have to manually sort again. So imported the devnagri OCR engine as well, and now my program would scan and recognize devnagri script for as it is and drop them, giving me clean English text from the images.

The English text however would still not be perfect, containing lot of noise, looking like this -

```
----- PAGE 1: page001_img000.png -----
The following table shows the number of students (Boys and Girls) and the percentage of girls
who passed in a 10+2 examination from six different schools (A-F) in the year 2023 and 2024.
Based on the data in the table, answer the questions that follow.
School-wise Details of Students Passing in a 10+2 examination
Year
schoo 2023 2024
umber of Passed|Percentage of Number of Passed|Percentage of
[Students [Passed Girls [Students [Passed Girls
lA | 500 46% [550 44%
IB | 500 | [640 [40%
IC 650 56% 560 [45%
Dp | [720 800 48%
IE | [600 840 55%

----- PAGE 2: page001_img001.png -----
What is the average number of boys who passed in the 10+2 examination in the year 2024 from
all the six schools (A-F) together?
1.376
2.377
3.378
4. 379

----- PAGE 3: page001_img002.png -----
What is the ratio of the number of boys who passed from School A in 2023 to the number of girls
who passed from School E in the year 2023?

----- PAGE 4: page002_img000.png -----
For School F, the number of boys who passed in the year 2024 is approximately % more
than that in 2023.
1.42.5
2.47.7
3. 52.8
4.57.5
```

so next was writing specific regex code that targets trails left by the OCR engine like image metadata when converting the image to text (which I myself had put to track in case of any problematic/unfinished image translation) and giving me a cleaner file

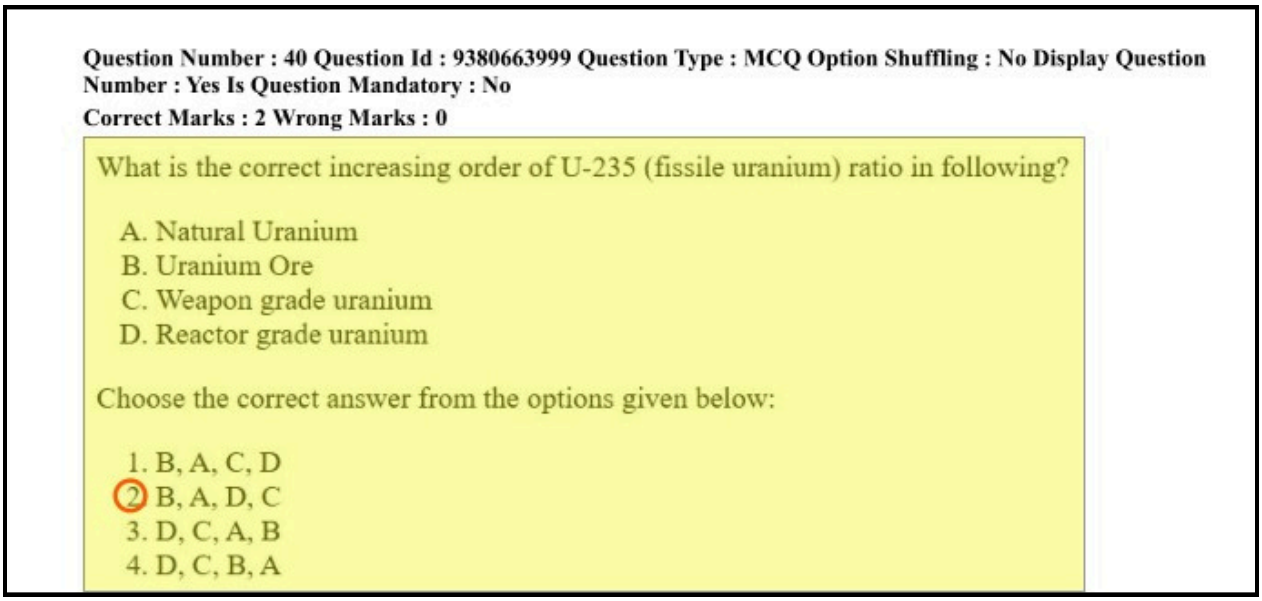
```python
def clean_raw_text(input_path, output_path):
    with open(input_path, "r", encoding="utf-8") as f:
        text = f.read()

    # appling specific cleaning rules
    patterns_to_remove = [
        #r"-----",
        r"PAGE",
        r"page",
        r"_img001\.png",
        r"_img000\.png"
    ]

    for pattern in patterns_to_remove:
        text = re.sub(pattern, "", text)

    # reduce whitespace
```

Now that the question text was extracted, it was time to extract the answers to each questions and map both the answer and question to their particular question id. The PYQ bundle I had got, instead of having answers in a seperate file list wise, had answers marked on the pdf image itself -



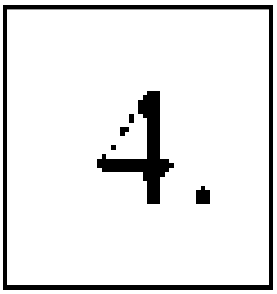
in the form of a red circle. The problem is, OCR cannot extract this information from the image because the circle is not any known alphabet. Each question was also of variable length, so it was also not possible to divide an OCR image into say blocks for example, and only search bottom portion of the image, which also would need coordinate mapping, entirely seperate field from text extraction. Two things were needed to extract the data of the correct answer - identifying a red circle, and identifying the letter/number within the red circle. I knew I could do the second step, so I focused on isolating the red circle.

So now, I run the OCR script, it extracts all the images from the pdf and stores them in a folder, then goes through each image with its english and hindi engine and stores the relevant text, another script cleans the extracted text, and then a new script goes through the images again and using Python Image Library (PIL), to scan any pixel that might have the red value,

```
for y in range(h):
    for x in range(w):
        r, g, b = px[x, y]
        if r > 150 and g < 130 and b < 130:
            px[x, y] = (255, 255, 255)
        elif r < 140 and g < 140 and b < 140:
            px[x, y] = (0, 0, 0)
        else:
            px[x, y] = (255, 255, 255)

return crop.convert("L").point(lambda x: 0 if x < 128 else 255, "1")
```

by calculating the rgb value of the page, and then cropping only that section. When done for each image, this gives us another file of images, this only contained pictures of numbers within red circles. However, before doing OCR to these images to get the numbers out, we have to convert the image to black-and-white and desaturate the red to zero, so that the OCR can concentrate on the number and not get distracted by the red noise around it. When done the ocr has images like these -



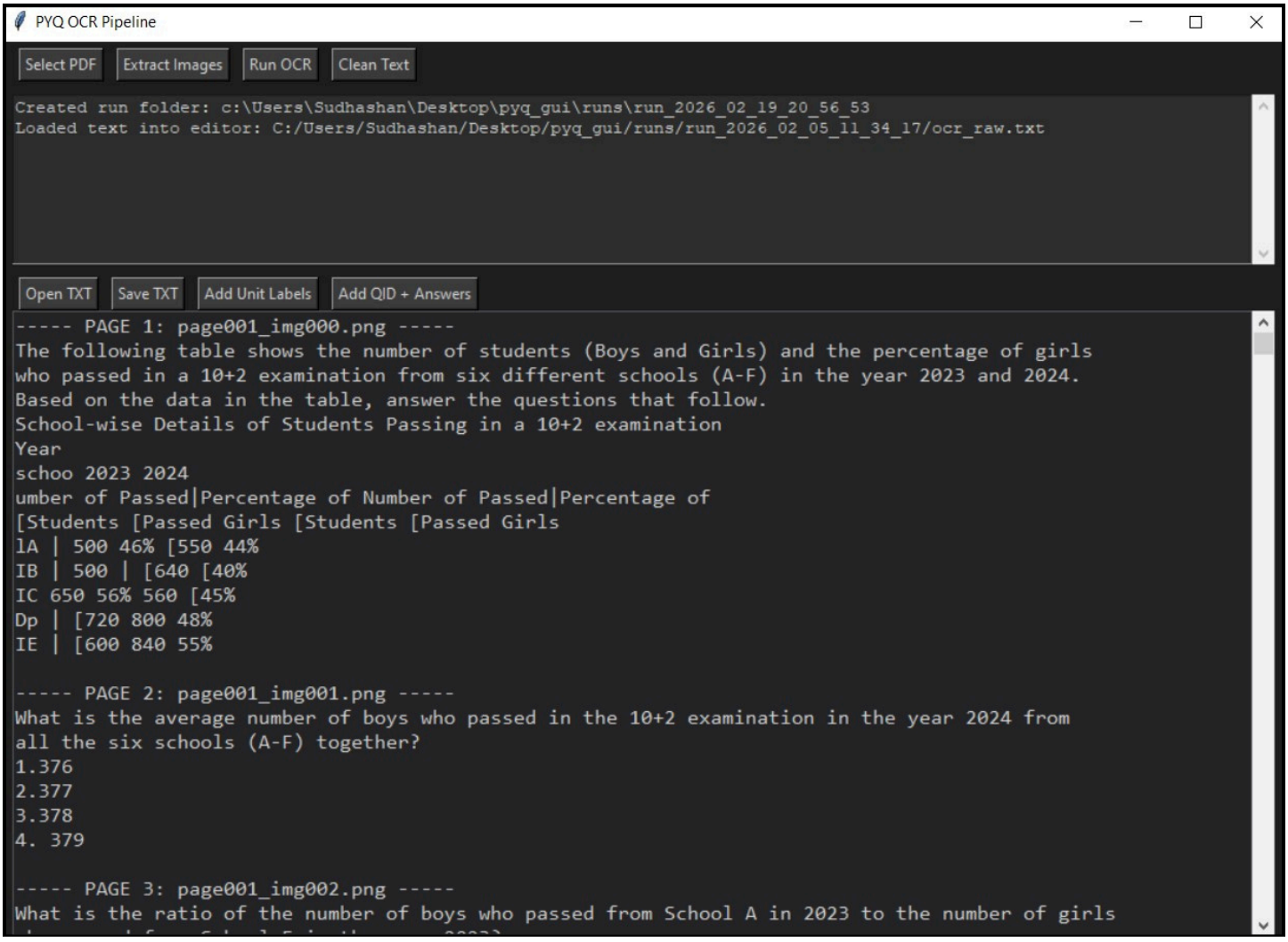
from which it extracts the number and stores it as the ‘correct option’ and then maps the extracted question, question number and now the correct option onto each other.

However, now we do have the question, answer and question id, for the website to function as a place to filter specific units, the questions themselves have to have their units mentioned, which is a data not available in the PYQ file. Though each question paper only has 50 questions and it is possible to manually tag them, I knew I would later do multiple PYQs, so manually tagging was out of option. I needed a script that tags the question’s units for me.

I tried some language models like BART and other zero-shot-classification models, but all of them either were too heavy (2.5GB) to download or were under paid license. However, even with smaller accessible models, the problem was that all of them were trained on general data, and neither knew UGC NET’s syllabus, so it would often classify as question like ‘What is the communication method used by Teachers in classroom’ under Communication Unit than the correct Teaching Aptitude unit. Because of this closed domain knowledge, even with bigger models, the problem wouldn’t be solved. So I had to opt for making a small transformer model myself using SkLearn’s Test/Train Split and TfidfVector, however, making a

model myself meant it had zero data to train on, unless I gave it data, so I spent some days just manually typing the units of different questions beside them, so that the model can see the data and train to predict future questions. I still am at 80% accuracy with this model, so for now, I still have to manually type out the question tags, but I aim to automate that once the model reaches 90% accuracy.

Running all these codes individually, one by one, on different files, and tracking which version is saved where, especially after multiple revisions was getting tiresome, so I also had to make a GUI with tkinter in python that allows me to do all these in one single place and have version control



Store the data in a displayable format (JSON) - Now, when all this is done, and the txt file looks clean like this

ocr_cleaned - Notepad

File Edit Format View Help

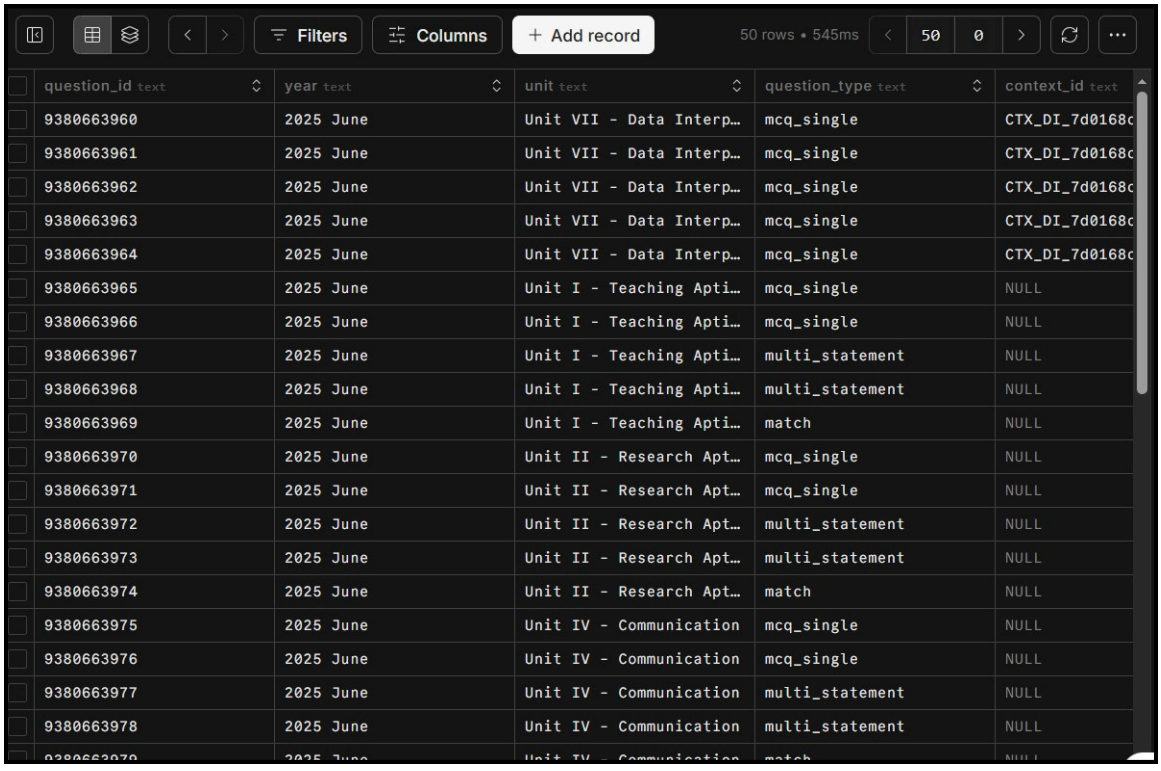
```
[YEAR: 2025 June]
[UNIT: Unit VII - Data Interpretation]
For School F, the number of boys who passed in the year 2024 is approximately % more than that in 2(
1. 42.5
2. 47.7
3. 52.8
4. 57.5

----- 5: 002 -----
[QID: 9380663963 | ANS: 2]
[YEAR: 2025 June]
[UNIT: Unit VII - Data Interpretation]
The average number of girls who passed in the year 2024 from all the six schools is ___ % of the nun
1. 61.1
2. 62.2
3. 63.3
4. 64.4

----- 6: 002_img002.png -----
[QID: 9380663964 | ANS: 1]
[YEAR: 2025 June]
[UNIT: Unit VII - Data Interpretation]
Number of boys who passed in the examination is more than 300 in the year 2023 as well as in 2024 ir
1. two
2. three
3. four
4. five
```

I wrote another script that takes this txt data and converts it into JSON, a format that the front-end and the database would store thing in for smooth display on websites across mobiles and desktops.

Upload the data to a live database (NeonDB) - Once the data was in JSON format, it was quite easy to transfer to NeonDB, and the account making in Neon was easy too.



Filters | Columns | + Add record | 50 rows • 545ms | 50 | 0

question_id text	year text	unit text	question_type text	context_id text
9380663960	2025 June	Unit VII - Data Interp…	mcq_single	CTX_DI_7d0168c
9380663961	2025 June	Unit VII - Data Interp…	mcq_single	CTX_DI_7d0168c
9380663962	2025 June	Unit VII - Data Interp…	mcq_single	CTX_DI_7d0168c
9380663963	2025 June	Unit VII - Data Interp…	mcq_single	CTX_DI_7d0168c
9380663964	2025 June	Unit VII - Data Interp…	mcq_single	CTX_DI_7d0168c
9380663965	2025 June	Unit I - Teaching Apti…	mcq_single	NULL
9380663966	2025 June	Unit I - Teaching Apti…	mcq_single	NULL
9380663967	2025 June	Unit I - Teaching Apti…	multi_statement	NULL
9380663968	2025 June	Unit I - Teaching Apti…	multi_statement	NULL
9380663969	2025 June	Unit I - Teaching Apti…	match	NULL
9380663970	2025 June	Unit II - Research Apt…	mcq_single	NULL
9380663971	2025 June	Unit II - Research Apt…	mcq_single	NULL
9380663972	2025 June	Unit II - Research Apt…	multi_statement	NULL
9380663973	2025 June	Unit II - Research Apt…	multi_statement	NULL
9380663974	2025 June	Unit II - Research Apt…	match	NULL
9380663975	2025 June	Unit IV - Communication	mcq_single	NULL
9380663976	2025 June	Unit IV - Communication	mcq_single	NULL
9380663977	2025 June	Unit IV - Communication	multi_statement	NULL
9380663978	2025 June	Unit IV - Communication	multi_statement	NULL

In the database, it stays like this, and the front end can call the data by SQL queries. Because the database charges by the amount of data stored in it, I had to learn about proper database techniques, one of which I implemented was storing the data of the Comprehension Passage and Data Interpretation tables only once, instead of storing five times for five comprehension passage questions, and then the once stored pasaage with a context_id, that I then distribute to the five questions, and they render the once stored passage for five times, instead of five stored passages for five times.

Write script for the front-end and backend that talks to the database and displays the information on screen - This was another tedious part, designing the website, calling the data, making the tables, getting used to React and API calling. However, except of the logic of 'click the correct option and green lights up and wrong option, red lights up', there was no operational logic used. A code would contact the database, Neon would send the data, the website would show the data.

However, i did implement pagination and moving the filters to backend and database, than on browser. Earlier, with limited questions, I would call all the data from the database to the user's browser, and then the browser would only show the data that the user clicked for, but as data increased, dumping all the data from the database to the user's mobile/PC would use up a lot of RAM, so now with moving filters to backend, the database only send the specific data that the user clicks for. So earlier, on opening the website, the database would send all data across all units to the browser, which would store the data in RAM, and then only show, say the Comprehension unit when the user clicked the dropdown, but now, when the user clicks the Comprehension unit, only the data for the comprehension unit goes to the user's browser, decreasing RAM usage.

Deploy on GitHub and Front-end server (Vercel) - Once I had deployed and checked everything from my localhost server, I pushed the files to Gihub, and made an account of Vercel, which used the Github repo to display all frontend, and contact and process the backend.