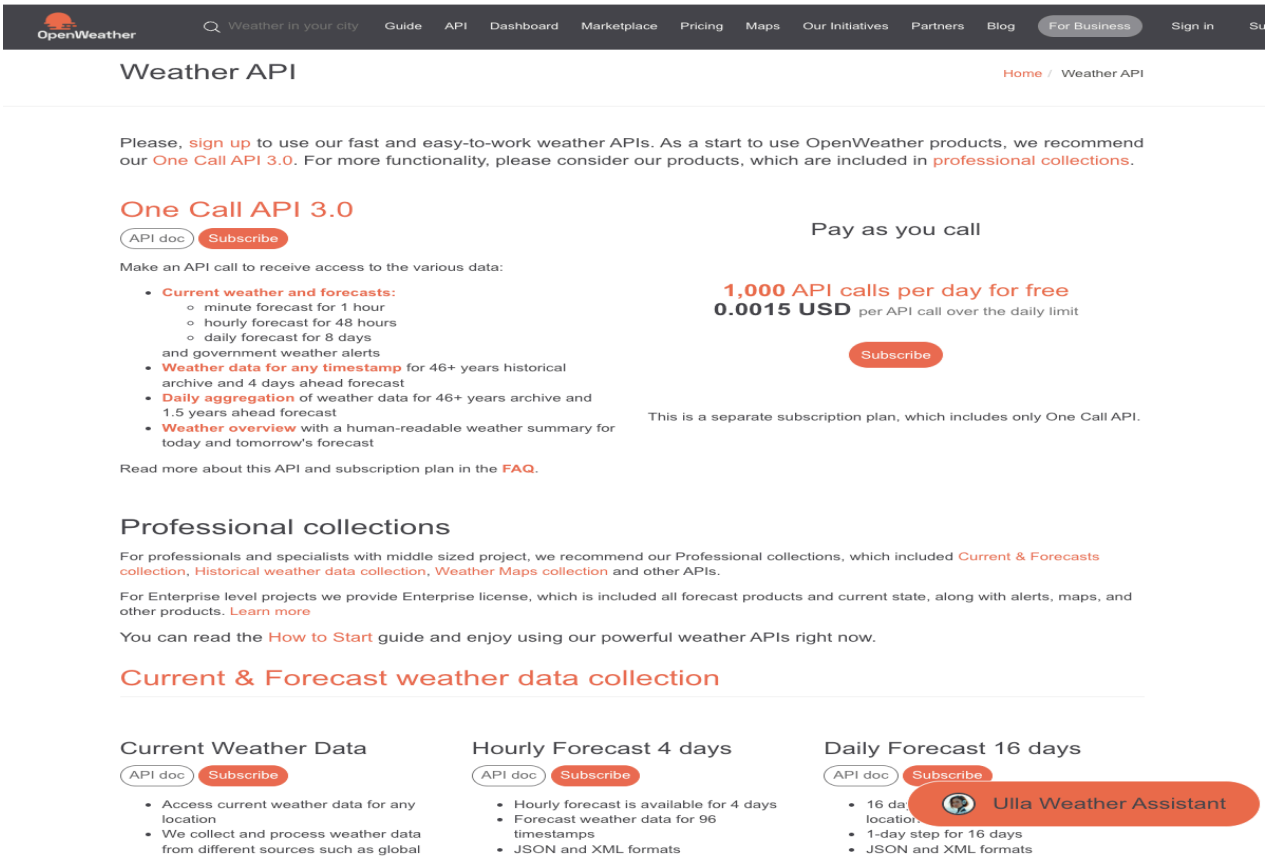
QUESTION 1:

JSON: https://openweathermap.org/api



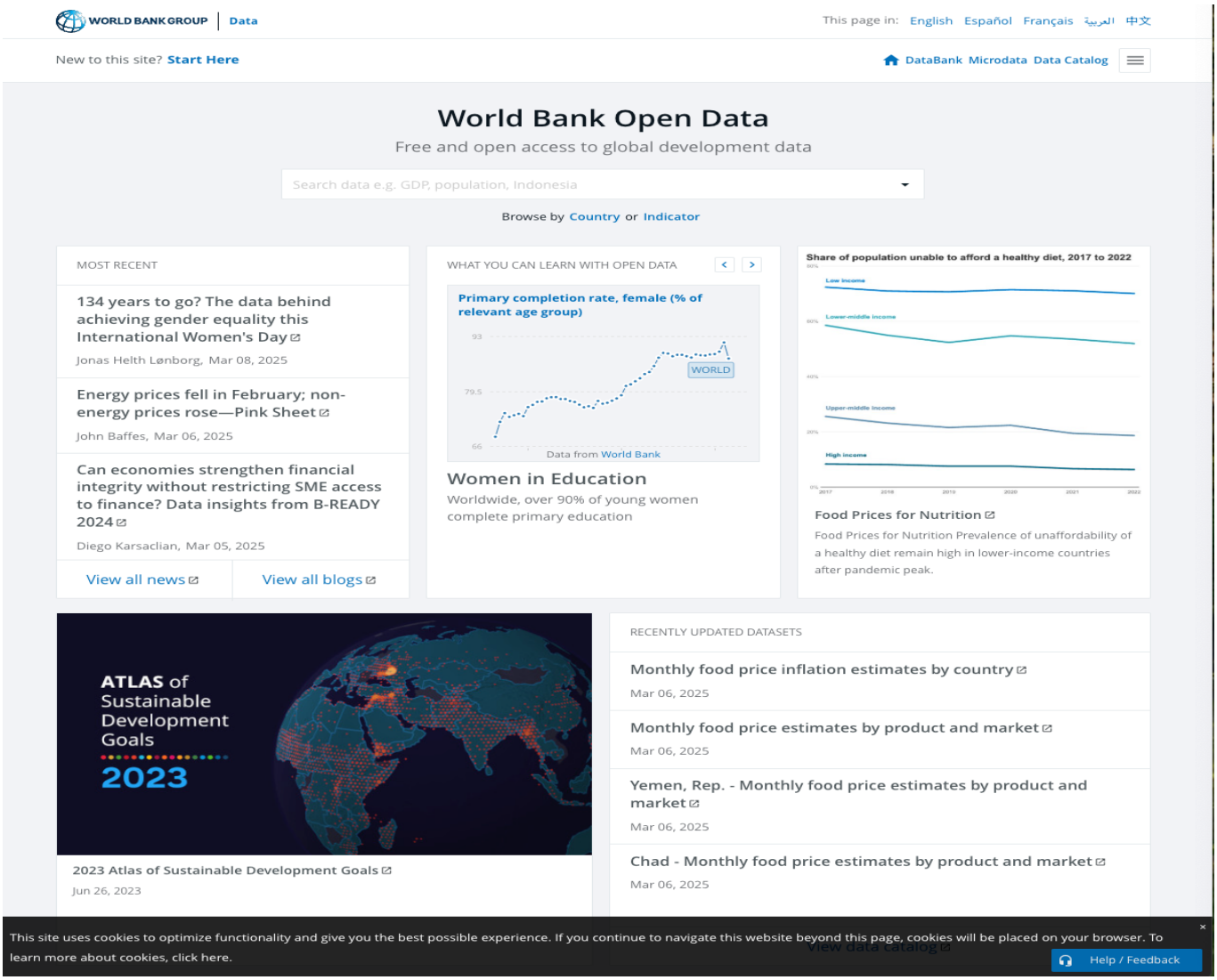
Structure and Composition:

1. **Clear Navigation**: The site is well-organized, with sections for API descriptions, pricing, and support, making it easy for users to find information.
2. **Pricing Transparency**: Free and paid tiers are clearly outlined, helping users choose the right plan.
3. **Detailed Documentation**: APIs like One Call 3.0 are explained with JSON response examples, ensuring developers can integrate them easily.

Technology/Methods for Web Database (JSON Focus):

1. **RESTful APIs**: JSON is delivered via RESTful APIs, using HTTP methods (GET, POST) for data retrieval.
2. **Database Backend**: A relational database (e.g., PostgreSQL or MySQL) is used to store weather data, with JSON responses generated dynamically.
3. **Cloud Infrastructure**: Scalable cloud platforms (e.g., AWS, Google Cloud) handle data storage and API hosting, ensuring reliability.
4. **Data Aggregation**: Weather data from global sources is collected, processed, and served in JSON format using ETL pipelines.

https://data.worldbank.org/



Structure and Composition:

1. **User-Friendly Navigation**:
   - The site offers clear navigation options, such as Search, Browse by Country or Indicator, and Most Recent updates, making it easy for users to find relevant data.
2. **Data Accessibility**:
   - The homepage highlights key datasets, news, and blogs, ensuring users can quickly access important information like gender equality, energy prices, and financial insights.
3. **Visual Data Representation**:
   - Charts and graphs (e.g., primary completion rates for females, food price inflation) are used to present data visually, enhancing user understanding.
4. **Recent Updates**:
   - A section for Recently Updated Datasets keeps users informed about the latest data additions, such as monthly food price estimates.

Technology/Methods for Web Database:

1. **Database Management**:
   - A robust relational database system like **PostgreSQL** or **MySQL** is likely used to store and manage large-scale global development data.
2. **Data Aggregation and ETL**:
   - Data from various sources (e.g., countries, indicators) is collected, cleaned, and transformed using **ETL (Extract, Transform, Load)** processes before being stored.
3. **RESTful APIs**:
   - The site likely provides RESTful APIs for programmatic access to datasets, allowing developers to retrieve data in XML or other formats.
4. **Cloud Infrastructure**:
   - Cloud platforms like **AWS** or **Azure** are likely used to host the database and ensure scalability, reliability, and global accessibility.
5. **Visualization Tools**:
   - Tools like **D3.js** or **Chart.js** may be used for creating interactive charts and graphs on the website.

QUESTION 2

```
SELECT s.ID
FROM student s
LEFT OUTER JOIN advisor a ON s.ID = a.s_ID
WHERE a.i_ID IS NULL;
```

Enter SQL commands here

```
1 SELECT s.ID
2 FROM student s
3 LEFT OUTER JOIN advisor a ON s.ID = a.s_ID
4 WHERE a.i_ID IS NULL;
5
6 -- enter your commands here
```

Execute | Save the db | Load an SQLite database file:

Choose File | No file chosen

ID
19991
54321
55739
70557

ii.

Enter SQL commands here

```
1 select i.id, i.name
2 from instructor as i
3 join course as c on i.dept_name = c.dept_name
4 left join teaches as t on i.id = t.id and c.course_id = t.course_id
5 group by i.id, i.name
6 having count(distinct c.course_id) = count(distinct t.course_id)
7 order by i.name-- enter your commands here
```

Execute | Save the db | Load an SQLite database file:

Choose File | No file chosen

ID	name
22222	Einstein
32343	El Said
98345	Kim
15151	Mozart
12121	Wu

Using Posrgres for the same question:

```
query_partial <- "
SELECT i.ID, i.name, i.dept_name, COUNT(DISTINCT t.course_id) AS
courses_taught
FROM instructor i
JOIN teaches t ON i.ID = t.ID
JOIN course c ON t.course_id = c.course_id
WHERE i.dept_name = c.dept_name
GROUP BY i.ID, i.name, i.dept_name
ORDER BY courses_taught DESC;
"
result_partial <- dbGetQuery(con, query_partial)
print(result_partial)
write.csv(result_partial, file =
'instructors_teaching_most_courses.csv', row.names = FALSE)
```

instructors_teaching_most_courses

id	name	dept_name	courses_taught
22591	DAgostino	Psychology	10
6569	Mingoz	Finance	8
99052	Dale	Cybernetics	8
74420	Voronina	Physics	5
79081	Ullman	Accounting	5
36897	Morris	Marketing	5
77346	Mahmoud	Geology	5
43779	Romero	Astronomy	3
3199	Gustafsson	Elec. Eng.	3
34175	Bondi	Comp. Sci.	3
41930	Tung	Athletics	3
19368	Wieland	Pol. Sci.	2
63287	Jaekel	Athletics	2
28400	Atanassov	Statistics	2
28097	Kean	English	2
81991	Valtchev	Biology	2
90376	Bietzk	Cybernetics	1
25946	Liley	Languages	1
80759	Queiroz	Biology	1
3335	Bourrier	Comp. Sci.	1
65931	Pimenta	Cybernetics	1
14365	Lembr	Accounting	1

QESTION 3

(a) Simple query: fetch all rows/columns in 'instructor' and create a data object

```
instructor_data <- dbGetQuery(con, "SELECT * FROM instructor")
```

```
head(instructor_data)
```

```
     id            name   dept_name    salary
1 63395       McKinnon Cybernetics  94333.99
2 78699          Pingr  Statistics  59303.62
3 96895           Mird   Marketing 119921.41
4  4233            Luo     English  88791.45
5  4034         Murata   Athletics  61387.56
6 50885 Konstantinides   Languages  32570.50
>
```

(b) Another query: fetch instructors in 'Comp. Sci.' department with a salary > 60000 (example condition)

```
comp_sci_instructors <- dbGetQuery(
  con,
  "SELECT * FROM instructor
   WHERE dept_name = 'Comp. Sci.' AND salary > 60000;"
)
```

```
comp_sci_instructors
```

```
     id     name  dept_name    salary
1 34175    Bondi Comp. Sci. 115469.11
2  3335 Bourrier Comp. Sci.  80797.83
>
```

(c) Query a different table, e.g., 'student', and store in an R dataframe

```
student_data <- dbGetQuery(con, "SELECT * FROM student WHERE
tot_cred >= 50")
```

```
head(student_data)
```

```
     id       name  dept_name tot_cred
1 79352      Rumat    Finance      100
2 76672     Miliko Statistics      116
3 14182 Moszkowski Civil Eng.       73
4 44985     Prieto    Biology       91
5 44271    Sowerby    English      108
6 40897    Coppens       Math       58
>
```

id	name	dept_name	salary
63395	McKinnon	Cybernetics	94333.99
78699	Pingr	Statistics	59303.62
96895	Mird	Marketing	119921.41
4233	Luo	English	88791.45
4034	Murata	Athletics	61387.56
50885	Konstantinides	Languages	32570.5
79653	Levine	Elec. Eng.	89805.83
50330	Shuming	Physics	108011.81
80759	Queiroz	Biology	45538.32
73623	Sullivan	Elec. Eng.	90038.09
97302	Bertolino	Mech. Eng.	51647.57
57180	Hau	Accounting	43966.29
74420	Voronina	Physics	121141.99
35579	Soisalon-Soininen	Psychology	62579.61
31955	Moreira	Accounting	71351.42
37687	Arias	Statistics	104563.38
6569	Mingoz	Finance	105311.38
16807	Yazdi	Athletics	98333.65
14365	Lembr	Accounting	32241.56
90643	Choll	Statistics	57807.09
81991	Valtchev	Biology	77036.18
95030	Arinb	Statistics	54805.11
15347	Bawa	Athletics	72140.88
74426	Kenje	Marketing	106554.73
42782	Vicentino	Elec. Eng.	34272.67
58558	Dusserre	Marketing	66143.25
63287	Jaekel	Athletics	103146.87
59795	Desyl	Languages	48803.38
22591	DAgostino	Psychology	59706.49
48570	Sarkar	Pol. Sci.	87549.8
79081	Ullman	Accounting	47307.1
52647	Bancilhon	Pol. Sci.	87958.01
25946	Liley	Languages	90891.69
36897	Morris	Marketing	43770.36
72553	Yin	English	46397.59
3199	Gustafsson	Elec. Eng.	82534.37
34175	Bondi	Comp. Sci.	115469.11
48507	Lent	Mech. Eng.	107978.47
65931	Pimenta	Cybernetics	79866.95
3335	Bourrier	Comp. Sci.	80797.83
64871	Gutierrez	Statistics	45310.53
95709	Sakurai	English	118143.98
43779	Romero	Astronomy	79070.08
77346	Mahmoud	Geology	99382.59
28097	Kean	English	35023.18
90376	Bietzk	Cybernetics	117836.5
28400	Atanassov	Statistics	84982.92
41930	Tung	Athletics	50482.03
19368	Wieland	Pol. Sci.	124651.41
99052	Dale	Cybernetics	93348.83