Assignment 2

1. Aggregate Functions (COUNT, SUM, AVG, MIN, MAX)

a. Count the total number of users

```
SELECT COUNT(*) AS total_users FROM hc.users;
```

	total_users bigint
1	10

b. Count the number of active users.

```
SELECT COUNT(*) AS total_active_users FROM hc.users
    WHERE is_active = 'true';
```

	total_active_users bigint
1	8

c. Count the number of users per role

```
SELECT r.role_name, COUNT(*) AS user_count
    FROM hc.users u
    JOIN hc.roles r
    ON u.role_id = r.role_id
    GROUP BY r.role_name
    ORDER BY user_count DESC;
```

	role_name character varying (50)	user_count bigint
1	Manager	3
2	Admin	2
3	Staff	2
4	Patient	2
5	Doctor	1

d. Find the minimum and maximum birth_date of users

```
SELECT MAX(birth_date) AS maximum_date,MIN(birth_date) AS minimum_date FROM
hc.users;
```

	maximum_date date	minimum_date date
1	2005-02-26	1997-08-07

e. Calculate the average age of users

```
SELECT AVG((CURRENT_DATE - birth_date)/365) AS Average_age
    FROM hc.users;
```

	average_age numeric
1	24.0000000000000000

f. Count total categories per service type

```
SELECT s.service_type_name,COUNT(c.category_id) AS total_categories
    FROM hc.categories c
    JOIN hc.service_types s
    ON c.service_type_id = s.service_type_id
    GROUP BY s.service_type_name
    ORDER BY total_categories DESC;
```

	service_type_name character varying (100)	total_categories bigint
1	Healthcare	3
2	Laboratory	2

g. Count total sub-categories per category

SELECT

c.category_name,

COUNT(s.sub_category_id) AS total_sub_categories

FROM hc.categories c

JOIN hc.sub_categories s

ON c.category_id = s.category_id

GROUP BY c.category_name

ORDER BY total_sub_categories DESC;

	category_name character varying (100)	total_sub_categories bigint
1	General Medicine	2
2	Blood Test	2
3	Cardiology	1

2 GROUP BY & HAVING

a. Fetch the number of users grouped by role

	role_name character varying (50)	total_users bigint
1	Manager	3
2	Admin	2
3	Staff	2
4	Patient	2
5	Doctor	1

b. Fetch roles having more than 2 users

```
SELECT r.role_name,
COUNT(u.user_id) AS total_users
        FROM hc.users u
        JOIN hc.roles r
ON u.role_id = r.role_id
        GROUP BY r.role_name
        HAVING COUNT(u.user_id)>2;
```

	role_name character varying (50)	total_users bigint
1	Manager	3

c. Fetch service types having more than 3 categories

```
SELECT
s.service_type_name,
COUNT(c.category_id) AS total_categories
        FROM hc.service_types s
        JOIN hc.categories c
        ON s.service_type_id = c.service_type_id
        GROUP BY s.service_type_id
        HAVING COUNT(c.category_id)>3
```

	service_type_name character varying (100)	total_categories bigint
1	Healthcare	5

d. Fetch categories having at least 2 sub-categories

	category_name character varying (100)	total_sub_categories bigint
1	General Medicine	2
2	Blood Test	2

e. Fetch users grouped by birth year

```
SELECT EXTRACT(YEAR FROM birth_date) AS birth_year,
        COUNT(*) AS total_users
        FROM hc.users
        GROUP BY birth_year
```

	birth_year numeric	total_users bigint
1	2000	1
2	2001	1
3	1999	2
4	2004	6

f. Fetch birth years having more than 5 users

```
SELECT EXTRACT(YEAR FROM birth_date) AS birth_year
    FROM hc.users
    GROUP BY birth_year
    HAVING COUNT(*) > 5;
```

	birth_year numeric
1	2004

3. Joins

a. INNER JOIN:

i. Fetch categories with service type names

```
SELECT s.service_type_name,c.category_name
    FROM hc.categories c
    INNER JOIN hc.service_types s
    ON c.service_type_id = s.service_type_id;
```

	service_type_name character varying (100)	category_name character varying (100)
1	Healthcare	General Medicine
2	Healthcare	Cardiology
3	Healthcare	Neurology
4	Laboratory	Blood Test
5	Laboratory	X-Ray
6	Healthcare	Dermatology
7	Healthcare	Orthopedics

b. LEFT JOIN:

i. Fetch all service types including those without categories

```
SELECT s.service_type_name, c.category_name
    FROM hc.service_types s
    LEFT JOIN hc.categories c
    ON s.service_type_id = c.service_type_id;
```

	service_type_name character varying (100)	category_name character varying (100)
1	Healthcare	General Medicine
2	Healthcare	Cardiology
3	Healthcare	Neurology
4	Laboratory	Blood Test
5	Laboratory	X-Ray
6	Healthcare	Dermatology
7	Healthcare	Orthopedics
8	Wellness	[null]
9	Pharmacy	[null]
10	Insurance	[null]

c. RIGHT JOIN:

i. Fetch all categories even if they have no sub-categories

```
SELECT c.category_name,s.sub_category_name
FROM hc.sub_categories s
RIGHT JOIN hc.categories c
ON c.category_id = s.category_id;
```

	category_name character varying (100)	sub_category_name character varying (100)
1	General Medicine	Diabetes
2	General Medicine	Hypertension
3	Cardiology	ECG
4	Blood Test	CBC
5	Blood Test	Lipid Profile
6	Neurology	[null]
7	X-Ray	[null]
8	Dermatology	[null]
9	Orthopedics	[null]

4. EXISTS

a. Fetch roles that have at least one user

```
SELECT r.role_name
    FROM hc.roles r
    WHERE EXISTS (
```

```
        SELECT 1
         FROM hc.users u
        WHERE u.role_id = r.role_id
);
```

	role_name character varying (50)
1	Admin
2	Doctor
3	Patient
4	Manager
5	Staff

b. Fetch service types that have categories

	service_type_name character varying (100)
1	Healthcare
2	Laboratory

c. Fetch categories that have sub-categories

```
SELECT c.category_name
        FROM hc.categories c
        WHERE EXISTS (
         SELECT 1
                FROM hc.sub_categories s
                WHERE c.category_id = s.category_id
        );
```

	category_name character varying (100)
1	General Medicine
2	Cardiology
3	Blood Test

d. Fetch users whose role exists

```
SELECT u.first_name,u.last_name
        FROM hc.users u
        WHERE EXISTS (
                SELECT 1
                FROM hc.roles r
                WHERE u.role_id = r.role_id
        );
```

	first_name character varying (100)	last_name character varying (100)
1	jaimin	vaghasiya
2	axi	chovatiya
3	Axay	kapadiya
4	bhargav	mesiya
5	akash	kapadiya
6	Anni	tanna
7	Aradhy	raghuvanshi
8	kishan	bhanderi
9	prit	gopani
10	prince	vaghasiya

5. Common Table Expression (CTE)

a. Use a CTE to count the number of categories per service type

```
WITH category_counts AS (
        SELECT c.service_type_id,
        COUNT(c.category_id) AS total_categories
        FROM hc.categories c
        GROUP BY c.service_type_id
        )
                SELECT s.service_type_name,cc.total_categories
```

FROM hc.service_types s
LEFT JOIN category_counts cc
ON s.service_type_id = cc.service_type_id;

	service_type_name character varying (100)	total_categories bigint
1	Healthcare	5
2	Laboratory	2
3	Pharmacy	[null]
4	Insurance	[null]
5	Wellness	[null]

6. Constraints
   a. UNIQUE Constraint: i. Ensure users.email is unique
   Already Done initially.

   b. FOREIGN KEY Constraints:
      i. users → roles
      ii. categories → service_types
      iii. sub_categories → categories

      Above three of instruction had done initially.

   c. CHECK Constraints:
      i. Mobile number must have a valid length
      ALTER TABLE hc.users
         ADD CONSTRAINT check_mobile_length
         CHECK (char_length(mobile_number) <= 15);

      ii. birth_date must be a past date
      ALTER TABLE hc.users
         ADD CONSTRAINT check_birth_date
         CHECK (birth_date < CURRENT_DATE);

7. Indexes a. Create an index on users.email
   CREATE INDEX email_index

ON hc.users(email);

8. Date & Time Handling

a. Fetch users created today

SELECT * FROM hc.users

WHERE created_date= CURRENT_DATE;

user_id [PK] uuid	first_name character varying (100)	last_name character varying (100)	email character varying (50)	password text	created_by uuid	created_date date
1 32b9670e-6c56-4916-b559-4d7c81673822	jaimin	vaghasiya	jaimin@google.com	$2a$06$hwmX8eaBK5dwzUfCqcrUtuO3n/C7KNeEY6R3cgOQ6Adnjr.m6BRa...	2773f677-de5d-44dd-b6e4-d765710a0245	2026-01-19

modified_by uuid	modified_date date	is_active boolean	is_deleted boolean	birth_date date	address character varying (200)	mobile_number character varying (15)	role_id uuid
2773f677-de5d-44dd-b6e4-d765710a0245	2025-01-05	false	true	2001-03-16	A 124,Taxshil Apartment	9812165065	8ce272c6-8a07-4a55-88ca-0c8fa05708f0

b. Fetch users created in the last 30 days

SELECT * FROM hc.users

WHERE created_date >= CURRENT_DATE - 30;

user_id [PK] uuid	first_name character varying (100)	last_name character varying (100)	email character varying (50)	password text	created_by uuid	created_date date
1 32b9670e-6c56-4916-b559-4d7c816738...	jaimin	vaghasiya	jaimin@google.com	$2a$06$hwmX8eaBK5dwzUfCqcrUtuO3n/C7KNeEY6R3cgOQ6Adnjr.m6B...	2773f677-de5d-44dd-b6e4-d765710a0...	2026-01-19
2 880eb8bb-e10d-4a5b-bdf1-d076e9b23f7e	Anni	tanna	anni@outlook.com	$2a$06$RI0yKDB5UnOVmsfV4M.b2e.XdzbdnSn2zEfzZeyBJgmyzPVZVBc...	cbdb1f67-5ce9-4f2a-a756-182b3f6df012	2026-01-12

modified_by uuid	modified_date date	is_active boolean	is_deleted boolean	birth_date date	address character varying (200)	mobile_number character varying (15)	role_id uuid
2773f677-de5d-44dd-b6e4-d765710a0...	2025-01-05	false	true	2001-03-16	A 124,Taxshil Apartment	9812165065	8ce272c6-8a07-4a55-88ca-0c8fa05708f0
cefd56fc-5f98-4fdf-98d4-38ab0b562e54	2024-02-25	true	false	2004-02-26	Aradhana Apartment	7523693136	c085b691-ed01-44d5-9872-1df3bfb39f82

c. Extract year from users.birth_date

SELECT first_name,last_name,EXTRACT(YEAR FROM birth_date) AS year FROM hc.users;

	first_name character varying (100)	last_name character varying (100)	year numeric
1	Axay	kapadiya	1999
2	akash	kapadiya	2000
3	axi	chovatiya	2004
4	prince	vaghasiya	2004
5	prit	gopani	1999
6	jaimin	vaghasiya	2001
7	bhargav	mesiya	2004
8	kishan	bhanderi	2004
9	Aradhy	raghuvanshi	2004
10	Anni	tanna	2004

d. Calculate age of each user

SELECT first_name,last_name,((CURRENT_DATE - birth_date)/365) AS age
FROM hc.users;

	first_name character varying (100)	last_name character varying (100)	age integer
1	Axay	kapadiya	26
2	akash	kapadiya	25
3	axi	chovatiya	21
4	prince	vaghasiya	21
5	prit	gopani	26
6	jaimin	vaghasiya	24
7	bhargav	mesiya	21
8	kishan	bhanderi	21
9	Aradhy	raghuvanshi	21
10	Anni	tanna	21

e. Group users by birth year

SELECT EXTRACT(YEAR FROM birth_date),count(*) as year
FROM hc.users
GROUP BY EXTRACT(YEAR FROM birth_date);

	extract numeric	year bigint
1	2000	1
2	2001	1
3	1999	2
4	2004	6

9. Window Functions (ROW_NUMBER, RANK, LAG, LEAD)

a. Assign row numbers to users within each role based on created_date

WITH name_role AS (
SELECT u.first_name,u.last_name,r.role_name,u.created_date
FROM hc.roles r
JOIN hc.users u
ON r.role_id = u.role_id

)
SELECT first_name,last_name,ROW_NUMBER() OVER (PARTITION BY role_name
ORDER BY created_date),created_date,role_name
FROM name_role;

	first_name character varying (100)	last_name character varying (100)	row_number bigint	created_date date	role_name character varying (50)
1	axi	chovatiya	1	2025-04-20	Admin
2	jaimin	vaghasiya	2	2026-01-19	Admin
3	Axay	kapadiya	1	2025-01-01	Doctor
4	Aradhy	raghuvanshi	1	2021-01-23	Manager
5	kishan	bhanderi	2	2022-10-22	Manager
6	Anni	tanna	3	2026-01-12	Manager
7	bhargav	mesiya	1	2023-09-29	Patient
8	akash	kapadiya	2	2025-02-01	Patient
9	prince	vaghasiya	1	2025-03-27	Staff
10	prit	gopani	2	2025-06-21	Staff

b. Rank roles based on number of users
WITH rol_based_rank AS(
SELECT r.role_name,COUNT(*) AS total_users
FROM hc.roles r
JOIN hc.users u
ON r.role_id = u.role_id
GROUP BY r.role_name

)
SELECT role_name,total_users,RANK() OVER(ORDER BY total_users)
AS rank
FROM rol_based_rank;

	role_name character varying (50)	total_users bigint	rank bigint
1	Doctor	1	1
2	Admin	2	2
3	Staff	2	2
4	Patient	2	2
5	Manager	3	5

c. Use LAG to fetch previous user creation date per role

```
WITH lag_created_date AS(
SELECT r.role_name,u.created_date,u.first_name
FROM hc.roles r
JOIN hc.users u
ON r.role_id = u.role_id
)
SELECT role_name,first_name,created_date,LAG(created_date) OVER(PARTITION
BY role_name)
FROM lag_created_date;
```

	role_name character varying (50)	first_name character varying (100)	created_date date	lag date
1	Admin	jaimin	2026-01-19	[null]
2	Admin	axi	2025-04-20	2026-01-19
3	Doctor	Axay	2025-01-01	[null]
4	Manager	Anni	2026-01-12	[null]
5	Manager	Aradhy	2021-01-23	2026-01-12
6	Manager	kishan	2022-10-22	2021-01-23
7	Patient	akash	2025-02-01	[null]
8	Patient	bhargav	2023-09-29	2025-02-01
9	Staff	prit	2025-06-21	[null]
10	Staff	prince	2025-03-27	2025-06-21

d. Use LEAD to fetch next user creation date per role

```
WITH lead_created_date AS(
SELECT r.role_name,u.created_date,u.first_name
FROM hc.roles r
```

```
JOIN hc.users u
ON r.role_id = u.role_id
)
SELECT role_name,first_name,created_date,LEAD(created_date) OVER(PARTITION
BY role_name)
FROM lead_created_date;
```

	role_name character varying (50)	first_name character varying (100)	created_date date	lead date
1	Admin	jaimin	2026-01-19	2025-04-20
2	Admin	axi	2025-04-20	[null]
3	Doctor	Axay	2025-01-01	[null]
4	Manager	Anni	2026-01-12	2021-01-23
5	Manager	Aradhy	2021-01-23	2022-10-22
6	Manager	kishan	2022-10-22	[null]
7	Patient	akash	2025-02-01	2023-09-29
8	Patient	bhargav	2023-09-29	[null]
9	Staff	prit	2025-06-21	2025-03-27
10	Staff	prince	2025-03-27	[null]

e. Fetch the second oldest user per role

```
WITH sec_old_user AS (
SELECT u.first_name,u.last_name,r.role_name,u.created_date
FROM hc.roles r
JOIN hc.users u
ON r.role_id = u.role_id

),
ranked_user AS(
 SELECT first_name,last_name,role_name,ROW_NUMBER() OVER       (PARTITION
BY role_name ORDER BY created_date) AS second_oldest
 FROM sec_old_user

)
SELECT first_name,last_name,role_name,second_oldest
FROM ranked_user
WHERE second_oldest = 2;
```

	first_name character varying (100)	last_name character varying (100)	role_name character varying (50)	second_oldest bigint
1	jaimin	vaghasiya	Admin	2
2	kishan	bhanderi	Manager	2
3	akash	kapadiya	Patient	2
4	prit	gopani	Staff	2