#roles table

```
CREATE TABLE IF NOT EXISTS hc.roles (
   role_id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
   role_name VARCHAR(50) UNIQUE NOT NULL
);

INSERT INTO hc.roles (role_name) VALUES
('admin'),
('customer'),
('service_provider'),
('support_staff');
```

#update users table

```
ALTER TABLE hc.users
ADD COLUMN role_id UUID;

ALTER TABLE hc.users
ADD CONSTRAINT fk_user_role
FOREIGN KEY (role_id) REFERENCES hc.roles(role_id);

UPDATE hc.users SET role_id = (SELECT role_id FROM hc.roles WHERE role_name = 'customer');
UPDATE hc.users SET role_id = (SELECT role_id FROM hc.roles WHERE role_name = 'admin')
WHERE hc.users.email = 'bhargav@gmail.com';
UPDATE hc.users SET role_id = (SELECT role_id FROM hc.roles WHERE role_name =
'service_provider')WHERE hc.users.email = 'parth@gmail.com';
UPDATE hc.users SET role_id = (SELECT role_id FROM hc.roles WHERE role_name =
'support_staff')WHERE hc.users.email IN ('jay@example.com','akash@gmail.com') ;
```

#service_type table

```
CREATE TABLE  IF NOT EXISTS hc.service_types (
   service_type_id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
   service_type_name VARCHAR(100) NOT NULL
);

INSERT INTO hc.service_types (service_type_name)
VALUES ('home_cleaning'),
('senior_care'),
('nursing_care');
```

#categories

```
CREATE TABLE  IF NOT EXISTS hc.categories (
   category_id UUID PRIMARY KEY DEFAULT  uuid_generate_v4(),
   category_name VARCHAR(100) NOT NULL,
   service_type_id UUID NOT NULL CONSTRAINT fk_service_type REFERENCES
hc.service_types(service_type_id)
);

INSERT INTO hc.categories (category_name, service_type_id)
VALUES
('regular_cleaning',
 (SELECT service_type_id FROM hc.service_types WHERE service_type_name =
'home_cleaning')),
```

```
('deep_cleaning',
 (SELECT service_type_id FROM hc.service_types WHERE service_type_name =
'home_cleaning')),
('kitchen_cleaning',
 (SELECT service_type_id FROM hc.service_types WHERE service_type_name =
'home_cleaning')),
('bathroom_cleaning',
 (SELECT service_type_id FROM hc.service_types WHERE service_type_name =
'home_cleaning')),

('daily_assistance',
 (SELECT service_type_id FROM hc.service_types WHERE service_type_name = 'senior_care')),

('icu_nursing',
 (SELECT service_type_id FROM hc.service_types WHERE service_type_name = 'nursing_care')),
('general_nursing',
 (SELECT service_type_id FROM hc.service_types WHERE service_type_name = 'nursing_care'));
```

#sub_categories

```
CREATE TABLE hc.sub_categories (
   sub_category_id UUID PRIMARY KEY DEFAULT  uuid_generate_v4(),
   sub_category_name VARCHAR(100) NOT NULL,
   category_id UUID NOT NULL CONSTRAINT fk_category REFERENCES
hc.categories(category_id)
);
```

Assignment-2

```
INSERT INTO hc.sub_categories (sub_category_name, category_id)
VALUES
('daily_cleaning',
 (SELECT category_id FROM hc.categories WHERE category_name = 'regular_cleaning')),
('weekly_cleaning',
 (SELECT category_id FROM hc.categories WHERE category_name = 'regular_cleaning')),

('full_house_cleaning',
 (SELECT category_id FROM hc.categories WHERE category_name = 'deep_cleaning')),

('chimney_cleaning',
 (SELECT category_id FROM hc.categories WHERE category_name = 'kitchen_cleaning')),
('cabinet_cleaning',
 (SELECT category_id FROM hc.categories WHERE category_name = 'kitchen_cleaning')),

('tile_cleaning',
 (SELECT category_id FROM hc.categories WHERE category_name = 'bathroom_cleaning')),

('personal_hygiene',
 (SELECT category_id FROM hc.categories WHERE category_name = 'daily_assistance')),
('meal_support',
 (SELECT category_id FROM hc.categories WHERE category_name = 'daily_assistance')),

('ventilator_support',
 (SELECT category_id FROM hc.categories WHERE category_name = 'icu_nursing')),
```

```
('critical_patient_monitoring',
 (SELECT category_id FROM hc.categories WHERE category_name = 'icu_nursing')),

('injection_service',
 (SELECT category_id FROM hc.categories WHERE category_name = 'general_nursing')),
('wound_dressing',
 (SELECT category_id FROM hc.categories WHERE category_name = 'general_nursing')),
```

#Assignment Queries

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

--a. Count the total number of users

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

--b. Count the number of active users

```
SELECT COUNT(*) FROM hc.users WHERE is_active = TRUE;
```

--c. Count the number of users per role

```
SELECT role_id, COUNT(*)  FROM hc.users
GROUP BY role_id;
```

--d. Find the minimum and maximum birth_date of users

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

--e. Calculate the average age of users

```
SELECT AVG(EXTRACT(YEAR FROM AGE(birth_date)))
FROM hc.users;
```

--f. Count total categories per service type

```
SELECT service_type_id, COUNT(*) FROM hc.categories
GROUP BY service_type_id;
```

--g. Count total sub-categories per category

```
SELECT category_id, COUNT(*) FROM hc.sub_categories
GROUP BY category_id;
```

--2. GROUP BY & HAVING

--a. Fetch the number of users grouped by role

```
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;
```

--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;
```

--c. Fetch service types having more than 3 categories

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

```
--d. Fetch categories having at least 2 sub-categories
SELECT c.category_name, COUNT(sc.sub_category_id) AS total_sub_categories
FROM hc.categories c
JOIN hc.sub_categories sc ON c.category_id = sc.category_id
GROUP BY c.category_name
HAVING COUNT(sc.sub_category_id) >= 2;

--e. Fetch users grouped by birth year
SELECT EXTRACT(YEAR FROM birth_date) AS birth_year, COUNT(user_id) AS total_users
FROM hc.users
GROUP BY EXTRACT(YEAR FROM birth_date);

--f. Fetch birth years having more than 5 users
SELECT EXTRACT(YEAR FROM birth_date) AS birth_year, COUNT(user_id) AS total_users
FROM hc.users
GROUP BY EXTRACT(YEAR FROM birth_date)
HAVING COUNT(user_id) > 5;
```

--3. Joins

```
--a. INNER JOIN: i. Fetch categories with service type names
SELECT c.category_name, st.service_type_name
FROM hc.categories c
INNER JOIN hc.service_types st ON c.service_type_id = st.service_type_id;
```

```
--b. LEFT JOIN:  i. Fetch all service types including those without categories
SELECT st.service_type_name, c.category_name
FROM hc.service_types st
LEFT JOIN hc.categories c ON st.service_type_id = c.service_type_id;

--c. RIGHT JOIN: i. Fetch all categories even if they have no sub-categories
SELECT c.category_name, sc.sub_category_name
FROM hc.sub_categories sc
RIGHT JOIN hc.categories c ON sc.category_id = c.category_id;
```

--4. EXISTS

```
--a. Fetch roles that have at least one user
SELECT r.role_id, r.role_name
FROM hc.roles r
WHERE EXISTS (
   SELECT 1
   FROM hc.users u
   WHERE u.role_id = r.role_id
);
```

Assignment-2

```
--b. Fetch service types that have categories
SELECT st.service_type_id, st.service_type_name
FROM hc.service_types st
WHERE EXISTS (
   SELECT 1
   FROM hc.categories c
   WHERE c.service_type_id = st.service_type_id
);
--c. Fetch categories that have sub-categories
SELECT c.category_id, c.category_name
FROM hc.categories c
WHERE EXISTS (
   SELECT 1
   FROM hc.sub_categories sc
   WHERE sc.category_id = c.category_id
);
--d. Fetch users whose role exists
SELECT u.user_id, u.first_name, u.email
FROM hc.users u
WHERE EXISTS (
   SELECT 1
   FROM hc.roles r
   WHERE r.role_id = u.role_id
);
```

--5. Common Table Expression (CTE)

```
--a. Use a CTE to count the number of categories per service type
WITH category_count AS (
   SELECT st.service_type_id, st.service_type_name, COUNT(c.category_id) AS total_categories
   FROM hc.service_types st
   INNER JOIN hc.categories c
      ON st.service_type_id = c.service_type_id
   GROUP BY st.service_type_id
)
SELECT
   service_type_name,
   total_categories
FROM category_count;
```

--6. Constraints

--a. UNIQUE Constraint:

i. Ensure users.email is unique

```
ALTER TABLE hc.users
ADD CONSTRAINT uq_users_email UNIQUE (email);
```

--b. FOREIGN KEY Constraints:

```
--i. users → roles
ALTER TABLE hc.users
ADD CONSTRAINT fk_user_role
FOREIGN KEY (role_id) REFERENCES hc.roles(role_id);
```

```
--ii. categories → service_types
ALTER TABLE hc.categories
ADD CONSTRAINT fk_categories_service_types
FOREIGN KEY (service_type_id) REFERENCES hc.service_types (service_type_id);

--iii. sub_categories → categories
ALTER TABLE hc.sub_categories
ADD CONSTRAINT fk_sub_categories_categories
FOREIGN KEY (category_id)
REFERENCES hc.categories (category_id);
```

--c. CHECK Constraints:

```
--i. Mobile number must have a valid length
ALTER TABLE hc.users
ADD CONSTRAINT chk_users_mobile_length
CHECK (length(mobile_number) BETWEEN 10 AND 15);

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

--7. Indexes

--a. Create an index on users.email

```
CREATE INDEX idx_users_email
ON hc.users (email);
```

--8. Date & Time Handling

--a. Fetch users created today

```
SELECT user_id, first_name, email
FROM hc.users
WHERE created_date = CURRENT_DATE;
```

--b. Fetch users created in the last 30 days

```
SELECT user_id, first_name, email
FROM hc.users
WHERE created_date >= CURRENT_DATE - INTERVAL '30 days';
```

--c. Extract year from users.birth_date

```
SELECT user_id, EXTRACT(YEAR FROM birth_date) AS Year
FROM hc.users;
```

--d. Calculate age of each user

```
SELECT user_id, EXTRACT(YEAR FROM age(birth_date)) AS age
FROM hc.users;
```

```
--e. Group users by birth year
SELECT EXTRACT(YEAR FROM birth_date) AS birth_year, COUNT(user_id) AS total_users
FROM hc.users
GROUP BY EXTRACT(YEAR FROM birth_date);
```

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

```
--a. Assign row numbers to users within each role based on created_date
SELECT u.user_id, u.first_name, u.last_name, u.role_id,
   ROW_NUMBER() OVER (
      PARTITION BY u.role_id
      ORDER BY u.created_date
   ) AS row_number
FROM hc.users u;

--b. Rank roles based on number of users
SELECT r.role_name, COUNT(u.user_id) AS total_users,
   RANK() OVER (
      ORDER BY COUNT(u.user_id) DESC
   ) AS role_rank
FROM hc.roles r
LEFT JOIN hc.users u
   ON r.role_id = u.role_id
GROUP BY r.role_name;
```

Assignment-2

```
--c. Use LAG to fetch previous user creation date per role
SELECT u.user_id, u.first_name, u.role_id, u.created_date,
   LAG(u.created_date) OVER (
      PARTITION BY u.role_id
      ORDER BY u.created_date
   ) AS previous_created_date
FROM hc.users u;

--d. Use LEAD to fetch next user creation date per role
SELECT u.user_id, u.first_name, u.role_id, u.created_date,
   LEAD(u.created_date) OVER (
      PARTITION BY u.role_id
      ORDER BY u.created_date
   ) AS next_created_date
FROM hc.users u;

--e. Fetch the second oldest user per role
SELECT user_id, first_name, last_name, role_id, birth_date
FROM (
   SELECT
      u.*,
      ROW_NUMBER() OVER (
         PARTITION BY u.role_id
         ORDER BY u.birth_date
      ) AS rn
   FROM hc.users u
```

```
) ranked_users

WHERE rn = 2;
```