1. Sebutkan perbedaan antara sharding dan replication

Sharding adalah memisahkan data yang besar dari suatu database ke beberapa node/worker sehingga menjadi komponen-komponen yang lebih kecil, dan lebih mudah dikelola, database dipisahkan secara horizontal yaitu berdasarkan baris

Replication adalah menyimpan salinan data yang sama di beberapa node/worker, sehingga data dapat diakses dari node mana saja dan tetap dapat digunakan meski node yang lain mengalami kendala

2. Lakukan percobaan untuk membuat reference table + distributed table

```
create table users(
    user_id serial primary key,
    username text not null,
    email text not null unique
);
select create_reference_table('users');
```

Membuat tabel users dengan beberapa kolom seperti user_id, username, dan email. Setelah itu table users akan di duplikat ke 3 node berbeda dengan perintah ‘create_reference_table(‘users)’

```
create table products (
    product_id serial primary key,
    name text not null,
    price numeric(10,2) not null
);
select create_reference_tabel('products');
```

Membuat tabel products dengan beberapa kolom seperti product_id, name, dan price. Setelah itu table users akan di duplikat ke 3 node berbeda dengan perintah ‘create_reference_table(‘product’)’

```
create sequence orders_order_id_seq;

create table orders (
    order_id int default nextval ('orders_order_id_seq'),
    user_id int references users(user_id),
    total_price numeric(10,2) not null,
    created_at timestamp defult now()
);

select create_distributed_table('orders', 'order_id');
```

Membuat table orders dengan beberapa kolom seperti order_id, user_id, total_price dan created_at. sebelum membuat table tsb, kita membuat sequence terlebih dahulu, agar kolom order_id dapat terisi secara otomatis/membuat interger nya sendiri secara ototmatis sesuai dengan banyak data yang diinput ke dalam tabel orders. Setelah itu table orders

akan kita sharding ke 3 node berbeda dengan perintah ‘create_distributed_table(‘orders’, ‘order_id’)’

```
create sequence order_details_order_detail_id_seq ('order_details', 'order_detail');

create table order_details (
    order_detail_id int default netxval ('order_details', 'order_detail_id'),
    order_id int,
    product_id int,
    quantity int not null
);

select create_distributed_table('order_details', 'order_id');
```

Membuat table order_details dengan beberapa kolom seperti order_detail_id, order_id, product_id dan quantity. sebelum membuat table tsb, kita membuat sequence terlebih dahulu, agar kolom order_detail_id dapat terisi secara otomatis/membuat interger nya sendiri secara ototmatis sesuai dengan banyak data yang diinput ke dalam tabel order_details. Setelah itu table order_details akan kita sharding ke 3 node berbeda dengan perintah ‘create_distributed_table(‘order_details’, ‘order_id’)’

```
INSERT INTO users (username, email) VALUES ('JohnDoe', 'john.doe@example.com'), ('JaneSmith', 'jane.smith@exam

INSERT INTO products (name, price) VALUES ('Laptop', 1000.00), ('Phone', 500.00), ('Headphones', 200.00), ('Mo

DO $$
DECLARE
  i INTEGER := 0;
BEGIN
  WHILE i < 1000 LOOP -- insert 1000 rows
    INSERT INTO orders (user_id, total_price) VALUES (floor(random() * 2 + 1)::INT, random() * 1000);
    INSERT INTO order_details (order_id, product_id, quantity) VALUES (i + 1, floor(random() * 4 + 1)::INT, fl
    i := i + 1;
  END LOOP;
END $$;
```

Memasukkan data ke dalam table users, products, orders, dan order_details

3. Di node/worker mana saja product ‘handphone’ tersimpan

```
WITH placement AS (
    SELECT
        shardid as shard_id
        , nodename as node_name
    FROM pg_dist_shard_placement
)

, product_ids AS (
    SELECT product_id
    FROM products
    where product_id = 3
    ORDER BY product_id
)

, product_shards AS (
    SELECT
        product_id
        , get_shard_id_for_distribution_column('products', product_id) as shard_id
        , 'orders_' || get_shard_id_for_distribution_column('products', product_id) as real_table_name
    FROM product_ids
)

SELECT
    product_shards.*
    , placement.node_name
FROM product_shards
INNER JOIN placement
    ON placement.shard_id = product_shards.shard_id
;
```

Melakukan pembuktian dengan kueri Common Table Expressions (CTE), yang mana klausa WITH mendefinisikan 3 pernyataan tambahan yaitu placement, product_ids, dan product_shard. Output placement digunakan pada kueri SELECT utama, output product_ids digunakan pada product_shard dan output product_shard digunakan pada kueri SELECT utama.

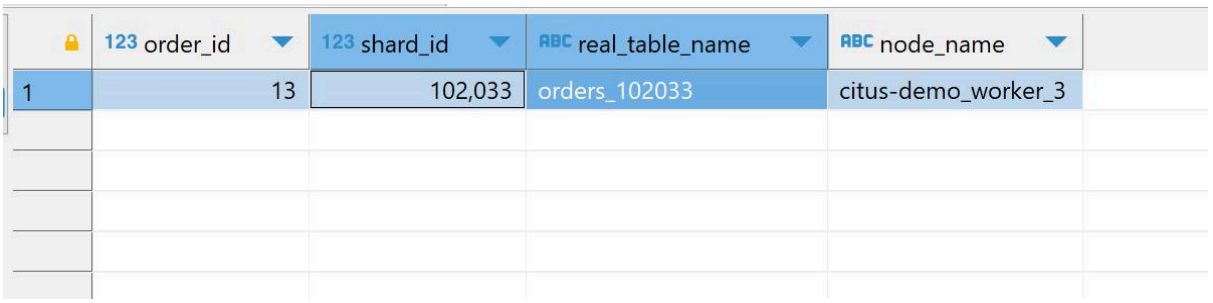
	123 product_id	123 shard_id	ABC real_table_name	ABC node_name
1	3	102,009	orders_102009	citus-demo_worker_1
2	3	102,009	orders_102009	citus-demo_worker_2
3	3	102,009	orders_102009	citus-demo_worker_3

Product ‘handphone’ ditemukan pada citus-demo worker_1, citus-demo worker_2, dan citus-demo worker_3, masing masing memiliki shard_id 102,009. Seperti yang diketahui product ‘handphone’ memiliki product_id = 3, sehingga yang ditampilkan hanya product dengan id = 3

4. Di node/worker mana saja product order dengan id 13 tersimpan

```
WITH placement AS (
    SELECT
        shardid as shard_id
        , nodename as node_name
    FROM pg_dist_shard_placement
)
, order_ids AS (
    SELECT order_id
    FROM orders
    where order_id = 13
    ORDER BY order_id
)
, order_shards AS (
    SELECT
        order_id
        , get_shard_id_for_distribution_column('orders', order_id) as shard_id
        , 'orders_' || get_shard_id_for_distribution_column('orders', order_id) as real_table_name
    FROM order_ids
)
SELECT
    order_shards.*
    , placement.node_name
FROM order_shards
INNER JOIN placement
    ON placement.shard_id = order_shards.shard_id
;
```

Melakukan pembuktian dengan kueri Common Table Expressions (CTE), yang mana klausa WITH mendefinisikan 3 pernyataan tambahan yaitu placement, order_ids, dan order_shard. Output placement digunakan pada kueri SELECT utama, output order_ids digunakan pada ordert_shard dan output order_shard digunakan pada kueri SELECT utama.



	order_id	shard_id	real_table_name	node_name
1	13	102,033	orders_102033	citus-demo_worker_3

Product order dengan id = 13 ditemukan hanya pada citus-demo worker_3, dengan memiliki shard_id 102,033.

5. Kapan sebaiknya kita menggunakan replication

Kita menggunakan replication, ketika ingin mengolah data yang bersifat kecil dan tidak bertambah setiap saat atau bisa disebut dengan data agregat, data ini cenderung akan selalu sama dan kecil kemungkinan bertambah setiap saat, sehingga data dapat digunakan di berbagai node atau worker jika dibutuhkan

6. Kapan sebaiknya kita menggunakan sharding

kita menggunakan sharding ketika ingin mengolah data yang berkukuran besar, dan cenderung bertambah setiap waktu, atau biasa disebut dengan data transaksional, data ini cenderung akan bertambah setiap waktu sesuai dengan jalannya suatu bisnis