Nama: Farhan Riyandi

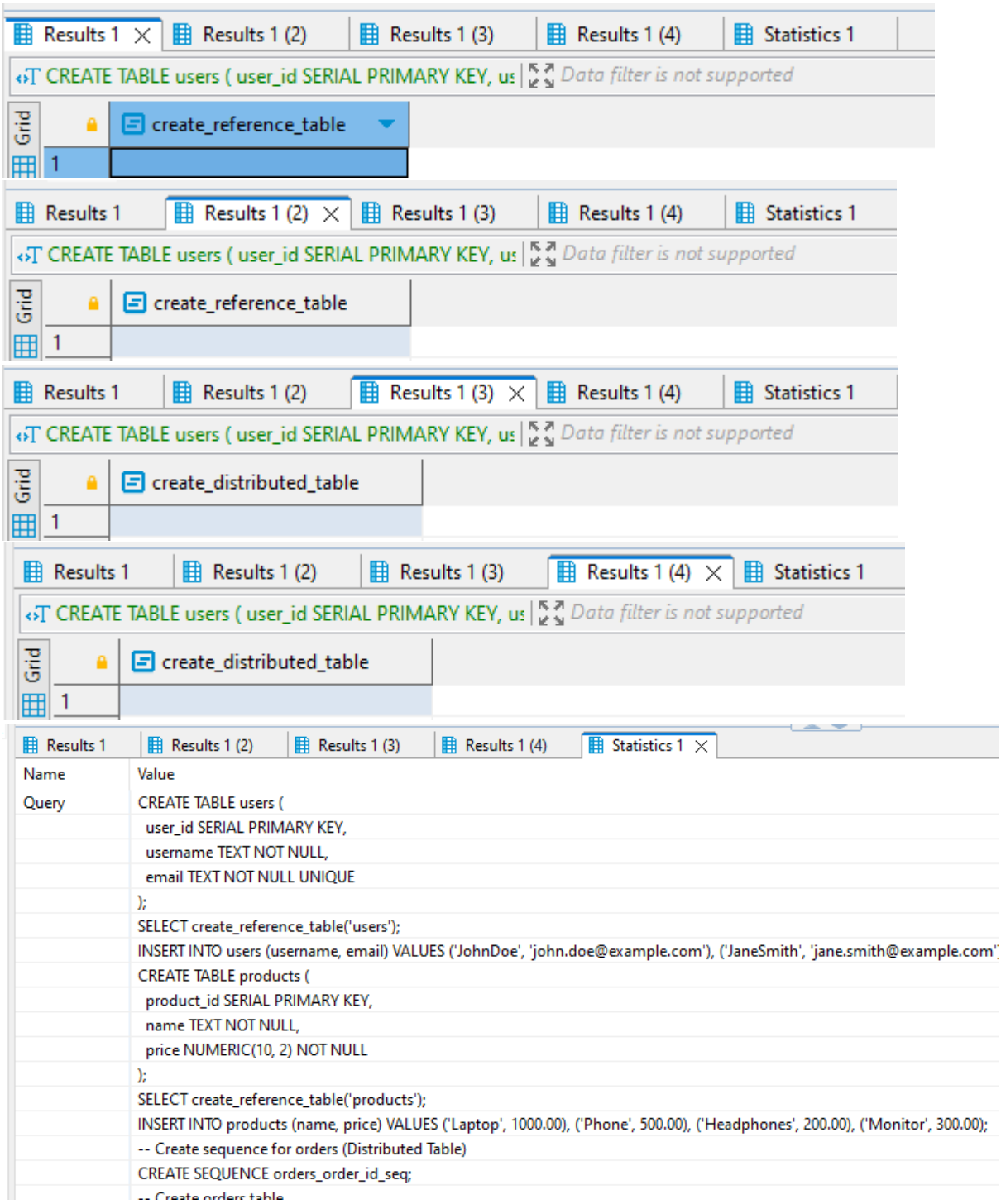
Data Engineer Batch 4

1. Jelaskan perbedaan antara replication dan sharding
   Jawab:
   Replikasi (Replication)
   - Data yang sama disalin di antara beberapa node:
     Replikasi melibatkan pembuatan salinan identik dari data di beberapa node. Ini memastikan bahwa jika salah satu node gagal, data tetap tersedia di node lain.
   - Digunakan untuk cadangan/keandalan tinggi (backup/high availability):
     Tujuan utama dari replikasi adalah untuk meningkatkan keandalan dan ketersediaan data. Dalam skenario replikasi, jika satu node gagal, data masih dapat diakses dari salinan yang ada di node lain, memberikan backup dan keandalan tinggi.
   - Biasanya digunakan pada tabel master atau dimension.

   Sharding
   - Data dibagi menjadi beberapa bagian dan didistribusikan di antara beberapa node:
     Data dibagi menjadi potongan-potongan yang lebih kecil (shard) dan didistribusikan ke beberapa node. Ini membantu dalam mendistribusikan beban dan memungkinkan pemrosesan data yang lebih efisien.
   - Digunakan untuk pemrosesan/distribusi data yang terdistribusi:
     Sharding membantu dalam menangani data yang sangat besar dengan membagi data di berbagai node. Ini juga memungkinkan pemrosesan data yang lebih cepat karena beban kerja didistribusikan di seluruh node.
   - Dapat menyimpan data yang lebih besar dari kapasitas node:
     Total kapasitas penyimpanan dan pemrosesan sistem dapat melampaui kapasitas dari satu node, karena data disimpan dan diproses di beberapa node.
   - Biasanya digunakan pada tabel transaction .

2. Lakukan percobaan untuk membuat reference table + distributed pada repo https://github.com/Immersive-DataEngineer-Resource/citus-demo
   Jawab:
   Pada file populate.sql saya copy paste ke dbeaver dan saya jalankan

   Dan begini hasilnya:



3. Di node/worker mana saja product “Headphone” tersimpan? Tunjukkan shard id nya

```sql
WITH placement AS (
    SELECT
        shardid as shard_id
        , nodename as node_name
    FROM pg_dist_shard_placement
),  product_shards as (
    select product_id, name
    , get_shard_id_for_distribution_column('products', product_id) as shard_id
    from products
    where name='Headphones'
)
select product_shards.*, placement.node_name
from product_shards
inner join placement on placement.shard_id = product_shards.shard_id
```

Results 1

WITH placement AS (SELECT shardid as shard_id , noc | Enter a SQL expression to filter res

Grid | Text

	product_id	name	shard_id	node_name
1	3	Headphones	102,112	citus-demo_worker_1
2	3	Headphones	102,112	citus-demo_worker_2
3	3	Headphones	102,112	citus-demo_worker_3

4. Di node/worker mana saja order dengan id 13 tersimpan? Tunjukkan shard id nya

```sql
WITH placement AS (
    SELECT
        shardid as shard_id
        , nodename as node_name
    FROM pg_dist_shard_placement
),  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 orders
    where order_id=13
)
select order_shards.*, placement.node_name
from order_shards
inner join placement on placement.shard_id = order_shards.shard_id
```

	order_id	shard_id	real_table_name	node_name
1	13	102,136	orders_102136	citus-demo_worker_3

5. Kapan sebaiknya kita menggunakan replication?
   Jawab:
   - Ketersediaan Tinggi (High Availability): Replikasi memungkinkan database tetap tersedia bahkan jika salah satu node mengalami kegagalan. Dengan replikasi, data tetap dapat diakses dari salinan yang lain.
   - Cadangan Data (Backup): Replikasi dapat digunakan untuk membuat salinan cadangan data yang bisa digunakan jika terjadi kehilangan data atau korupsi data.
   - Distribusi Beban (Load Balancing): Dengan beberapa salinan data yang tersebar di beberapa node, beban baca dapat didistribusikan, mengurangi tekanan pada satu server dan meningkatkan kinerja.

- Pemulihan Bencana (Disaster Recovery): Replikasi memastikan bahwa ada salinan data di lokasi yang berbeda yang dapat digunakan untuk pemulihan cepat jika terjadi bencana.

6. Kapan sebaiknya kita menggunakan sharding?
   Jawab:
   - Skalabilitas (Scalability): Sharding memungkinkan database untuk menangani volume data yang besar dan beban transaksi yang tinggi dengan membaginya ke beberapa node. Ini sangat berguna ketika data terus bertambah secara konsisten dan cepat.
   - Kinerja (Performance): Dengan membagi data ke dalam beberapa shard, beban kerja dapat didistribusikan, yang dapat meningkatkan kinerja aplikasi terutama untuk operasi tulis (write-heavy workloads).
   - Pengelolaan Data Besar (Big Data Management): Ketika ukuran data melebihi kapasitas satu server, sharding memungkinkan data tersebut disebar ke beberapa server, sehingga masing-masing server hanya menangani subset dari keseluruhan data.