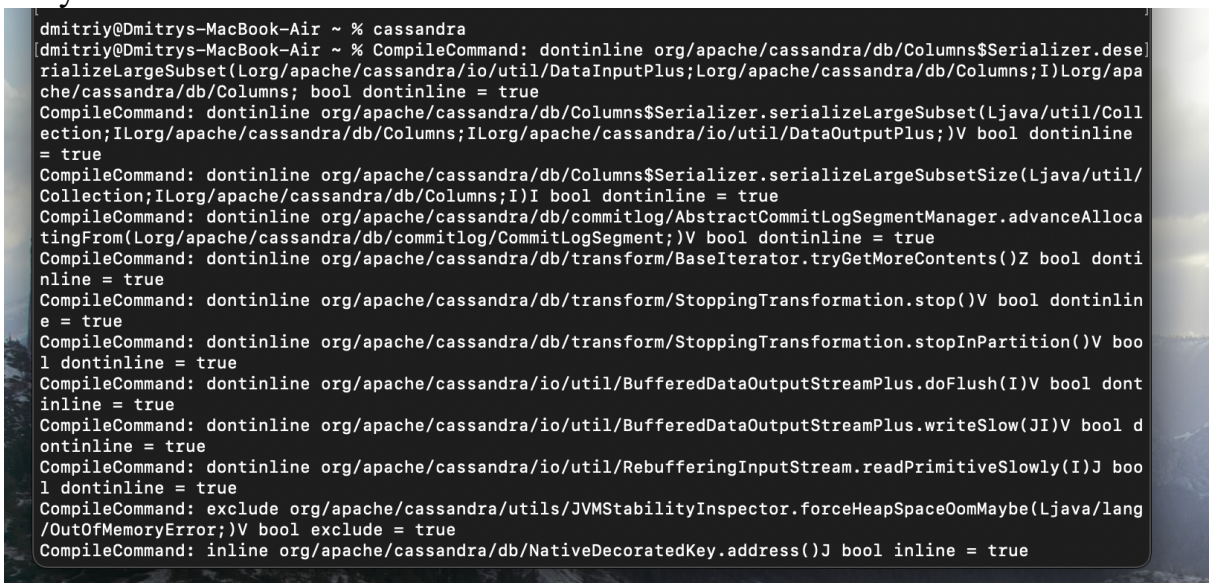
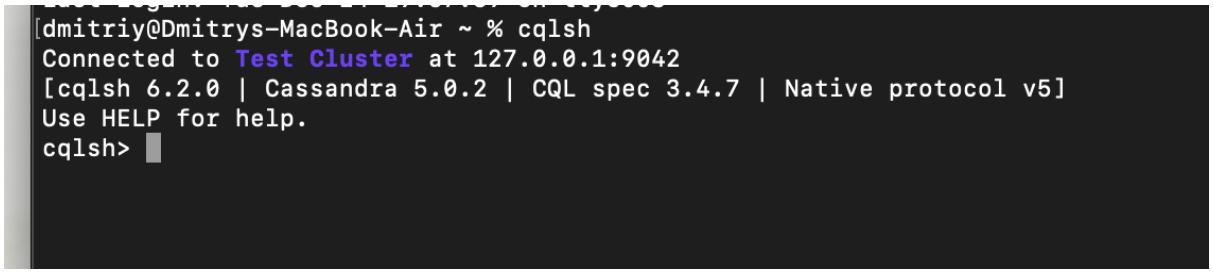
Рябко Дмитро ФБ-42мп

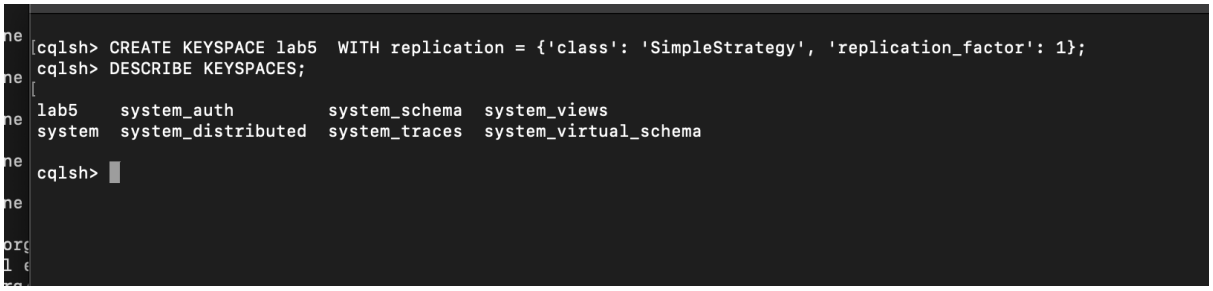
Запуск Cassandra:





Частина 1. Робота зі структурами даних у Cassandra

Створення keyspace:



Створення таблиці:

```
Table 'items' not found in keyspace 'lab5'
cqlsh:lab5> CREATE TABLE lab5.items (
        ...     category TEXT,
        ...     price DECIMAL,
        ...     manufacturer TEXT,
        ...     item_id UUID,
        ...     name TEXT,
        ...     attributes MAP<TEXT, TEXT>,
        ...     PRIMARY KEY ((category), price, manufacturer, item_id)
        ... );
```

Заповнення даними таблиці:

```
cqlsh> CREATE INDEX ON lab5.items(attributes);
cqlsh> INSERT INTO lab5.items (category, item_id, name, price, manufacturer, attributes)
   ... VALUES ('electronics', uuid(), 'iPad', 700.00, 'Apple', {'Screen': '10.2 inches', 'Storage': '256GB'});
cqlsh>
cqlsh> INSERT INTO lab5.items (category, item_id, name, price, manufacturer, attributes)
   ... VALUES ('electronics', uuid(), 'Gaming Laptop', 1800.00, 'Asus', {'RAM': '32GB', 'GPU': 'NVIDIA RTX 3060'});
cqlsh>
cqlsh> INSERT INTO lab5.items (category, item_id, name, price, manufacturer, attributes)
   ... VALUES ('electronics', uuid(), 'Headphones', 300.00, 'Sony', {'Type': 'Wireless', 'Battery': '30h'});
cqlsh>
cqlsh> INSERT INTO lab5.items (category, item_id, name, price, manufacturer, attributes)
   ... VALUES ('appliances', uuid(), 'Refrigerator', 1200.00, 'Samsung', {'Capacity': '300L', 'Energy Rating': 'A+'});
cqlsh>
cqlsh> INSERT INTO lab5.items (category, item_id, name, price, manufacturer, attributes)
   ... VALUES ('appliances', uuid(), 'Microwave', 200.00, 'LG', {'Power': '1000W', 'Capacity': '25L'});
cqlsh>
cqlsh> INSERT INTO lab5.items (category, item_id, name, price, manufacturer, attributes)
   ... VALUES ('appliances', uuid(), 'Dishwasher', 800.00, 'Bosch', {'Capacity': '12 settings', 'Energy Rating': 'A+'});
cqlsh>
cqlsh> INSERT INTO lab5.items (category, item_id, name, price, manufacturer, attributes)
   ... VALUES ('furniture', uuid(), 'Office Chair', 150.00, 'Ikea', {'Color': 'Black', 'Material': 'Mesh'});
cqlsh>
cqlsh> INSERT INTO lab5.items (category, item_id, name, price, manufacturer, attributes)
   ... VALUES ('furniture', uuid(), 'Dining Table', 400.00, 'Ikea', {'Seats': '6', 'Material': 'Wood'});
cqlsh>
cqlsh> INSERT INTO lab5.items (category, item_id, name, price, manufacturer, attributes)
   ... VALUES ('furniture', uuid(), 'Wardrobe', 600.00, 'HomeCenter', {'Doors': '3', 'Material': 'Particleboard'});
cqlsh>
```

```
cqlsh> SELECT * FROM lab5.items;

 category    | item_id                              | attributes                                                | manufacturer | name          | price
-------------+--------------------------------------+-----------------------------------------------------------+--------------+---------------+---------
  appliances | 1605a350-4d2d-4d0c-9135-9fdfa6884902 |        {'Capacity': '300L', 'Energy Rating': 'A+'} |      Samsung |  Refrigerator | 1200.00
  appliances | c14312d4-4bfe-4b19-8d42-571e24c96495 | {'Capacity': '12 settings', 'Energy Rating': 'A+'} |        Bosch |    Dishwasher |  800.00
  appliances | ed26443c-f6f0-4f2e-9e7b-fbe0879c9db9 |            {'Capacity': '25L', 'Power': '1000W'} |           LG |     Microwave |  200.00
 electronics | 412b8600-0d03-4b9f-8829-5b71cf314adc |      {'Screen': '10.2 inches', 'Storage': '256GB'} |        Apple |          iPad |  700.00
 electronics | 4f6915da-55cc-4843-a3cc-01ce62a402b8 |          {'GPU': 'NVIDIA RTX 3060', 'RAM': '32GB'} |         Asus | Gaming Laptop | 1800.00
 electronics | ad0e25b3-e0f3-481c-aed6-c2afa4b97a50 |            {'Battery': '30h', 'Type': 'Wireless'} |         Sony |    Headphones |  300.00
   furniture | 9af11488-aacb-4562-83ed-56545ff54f7f |        {'Doors': '3', 'Material': 'Particleboard'} |   HomeCenter |      Wardrobe |  600.00
   furniture | a597163d-392d-4619-b31e-5769413e3562 |             {'Color': 'Black', 'Material': 'Mesh'} |          Ikea |  Office Chair |  150.00
   furniture | af80613e-5777-414c-b487-ceab7ce1f2e8 |                {'Material': 'Wood', 'Seats': '6'} |          Ikea |  Dining Table |  400.00

(9 rows)
cqlsh>
```

1. Напишіть запит, який показує структуру створеної таблиці (команда *DESCRIBE*)

```
cqlsh> DESCRIBE TABLE lab5.items;

CREATE TABLE lab5.items (
    category text,
    item_id uuid,
    manufacturer text,
    name text,
    price decimal,
    attributes map<text, text>,
    PRIMARY KEY (category, item_id)
) WITH CLUSTERING ORDER BY (item_id ASC)
    AND additional_write_policy = '99p'
    AND allow_auto_snapshot = true
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND cdc = false
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '16', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND memtable = 'default'
    AND crc_check_chance = 1.0
    AND default_time_to_live = 0
    AND extensions = {}
    AND gc_grace_seconds = 864000
    AND incremental_backups = true
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair = 'BLOCKING'
    AND speculative_retry = '99p';

CREATE INDEX items_attributes_idx ON lab5.items (values(attributes));

CREATE INDEX items_manufacturer_idx ON lab5.items (manufacturer);
```

2. Напишіть запит, який виводить усі товари в певній категорії відсортовані за ціною

```
cqlsh:lab5> CREATE MATERIALIZED VIEW items_by_price AS
        ... SELECT * FROM lab5.items
        ... WHERE category IS NOT NULL AND price IS NOT NULL AND item_id IS NOT NULL
        ... PRIMARY KEY (category, price, item_id);
^[[C
Warnings :
Materialized views are experimental and are not recommended for production use.
```

```
cqlsh:lab5> SELECT * FROM items_by_price WHERE category = 'electronics' ORDER BY price ASC;

 category    | price   | item_id                              | attributes                                                | manufacturer | name
-------------+---------+--------------------------------------+-----------------------------------------------------------+--------------+----------------
 electronics |  300.00 | ad0e25b3-e0f3-481c-aed6-c2afa4b97a50 |                {'Battery': '30h', 'Type': 'Wireless'} |         Sony |    Headphones
 electronics |  700.00 | 412b8600-0d03-4b9f-8829-5b71cf314adc | {'Screen': '10.2 inches', 'Storage': '256GB'} |        Apple |           iPad
 electronics | 1800.00 | 4f6915da-55cc-4843-a3cc-01ce62a402b8 |     {'GPU': 'NVIDIA RTX 3060', 'RAM': '32GB'} |         Asus | Gaming Laptop

(3 rows)
```

3. Напишіть запити, які вибирають товари за різними критеріями в межах певної категорії (тут де треба замість індексу використайте Matirialized view):
назва,

```
(3 rows)
cqlsh:lab5> CREATE MATERIALIZED VIEW items_by_name AS
        ... SELECT * FROM lab5.items
        ... WHERE category IS NOT NULL AND name IS NOT NULL AND item_id IS NOT NULL
        ... PRIMARY KEY (category, name, item_id);

Warnings :
Materialized views are experimental and are not recommended for production use.

cqlsh:lab5>
```

```
(0 rows)
cqlsh:lab5> SELECT * FROM items_by_name WHERE category = 'electronics' AND name = 'iPad';

 category    | name | item_id                              | attributes                                    | manufacturer | price
-------------+------+--------------------------------------+-----------------------------------------------+--------------+--------
 electronics | iPad | 412b8600-0d03-4b9f-8829-5b71cf314adc | {'Screen': '10.2 inches', 'Storage': '256GB'} |        Apple | 700.00

(1 rows)
cqlsh:lab5>
```

ціна (в проміжку),

```
AlreadyExists: Table 'lab5.items_by_price' already exists
cqlsh:lab5> SELECT * FROM items_by_price
        ... WHERE category = 'electronics' AND price >= 700 AND price <= 1500;

 category    | price  | item_id                              | attributes                                                | manufacturer | name
-------------+--------+--------------------------------------+-----------------------------------------------------------+--------------+------
 electronics | 700.00 | 412b8600-0d03-4b9f-8829-5b71cf314adc | {'Screen': '10.2 inches', 'Storage': '256GB'} |        Apple | iPad

(1 rows)
cqlsh:lab5>
```

ціна та виробник

```
cqlsh:lab5> CREATE MATERIALIZED VIEW items_by_price_manufacturer AS
        ... SELECT * FROM lab5.items
        ... WHERE category IS NOT NULL AND price IS NOT NULL AND manufacturer IS NOT NULL AND item_id IS NOT NULL
        ... PRIMARY KEY ((category), price, manufacturer, item_id);

Warnings :
Materialized views are experimental and are not recommended for production use.

cqlsh:lab5>
```

```
cqlsh:lab5> SELECT * FROM items_by_price_manufacturer
        ... WHERE category = 'electronics' AND price = 800 AND manufacturer = 'Samsung';

 category    | price  | manufacturer | item_id                              | attributes                                                | name
-------------+--------+--------------+--------------------------------------+-----------------------------------------------------------+------------
 electronics | 800.00 |      Samsung | 5daca501-ffe3-4fb4-a34a-bdeefe3a008b | {'Battery': '4000mAh', 'Screen': '6.5 inches'} | Smartphone

(1 rows)
cqlsh:lab5>
```

Таблиця Orders:

```
cqlsh:lab5> CREATE TABLE lab5.orders (
        ...     customer_name TEXT,
        ...     order_id UUID,
        ...     item_ids LIST<UUID>,
        ...     total_price DECIMAL,
        ...     order_date TIMESTAMP,
        ...     PRIMARY KEY (customer_name, order_date, order_id)
        ... );
```

```
        ... );
cqlsh:lab5> INSERT INTO lab5.orders (customer_name, order_id, item_ids, total_price, order_date)
        ... VALUES ('John Doe', uuid(), [uuid(), uuid()], 2000.00, '2024-12-24 10:00:00');
cqlsh:lab5>
cqlsh:lab5> INSERT INTO lab5.orders (customer_name, order_id, item_ids, total_price, order_date)
        ... VALUES ('Jane Smith', uuid(), [uuid()], 800.00, '2024-12-23 15:30:00');
cqlsh:lab5>
cqlsh:lab5> INSERT INTO lab5.orders (customer_name, order_id, item_ids, total_price, order_date)
        ... VALUES ('John Doe', uuid(), [uuid(), uuid()], 1200.00, '2024-12-23 09:45:00');
cqlsh:lab5>
```

1. Напишіть запит, який показує структуру створеної таблиці (команда *DESCRIBE*)

```
CREATE TABLE lab5.orders (
    customer_name text,
    order_date timestamp,
    order_id uuid,
    total_price decimal,
    item_ids list<uuid>,
    PRIMARY KEY (customer_name, order_date, order_id)
) WITH CLUSTERING ORDER BY (order_date ASC, order_id ASC)
    AND additional_write_policy = '99p'
    AND allow_auto_snapshot = true
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND cdc = false
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '16', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND memtable = 'default'
    AND crc_check_chance = 1.0
    AND default_time_to_live = 0
    AND extensions = {}
    AND gc_grace_seconds = 864000
    AND incremental_backups = true
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair = 'BLOCKING'
    AND speculative_retry = '99p';
```

2. Для замовника виведіть всі його замовлення відсортовані за часом коли вони були зроблені

```
cqlsh:lab5> SELECT * FROM lab5.orders
        ... WHERE customer_name = 'John Doe'
        ... ORDER BY order_date DESC;
```

```
 customer_name | order_date                      | order_id                             | item_ids                                                                     | total_price
---------------+---------------------------------+--------------------------------------+------------------------------------------------------------------------------+-------------
      John Doe | 2024-12-24 08:00:00.000000+0000 | bd57034a-b950-41a0-af63-a5bac29341a5 | [0bf26bc6-6f6f-4f9a-9571-9151792eec1c, 33a811f2-b21d-4fe3-b861-b0da491fc0fe] |     2000.00
      John Doe | 2024-12-23 07:45:00.000000+0000 | a9dc76b3-7630-46da-be4a-1090b45055a8 | [fa0382ac-6005-47b5-8742-cde0c1d029fd, dd4a0834-6d70-4063-9f35-39896b7f0074] |     1200.00

(2 rows)
cqlsh:lab5>
```

3. Для кожного замовників визначте суму на яку були зроблені усі його замовлення

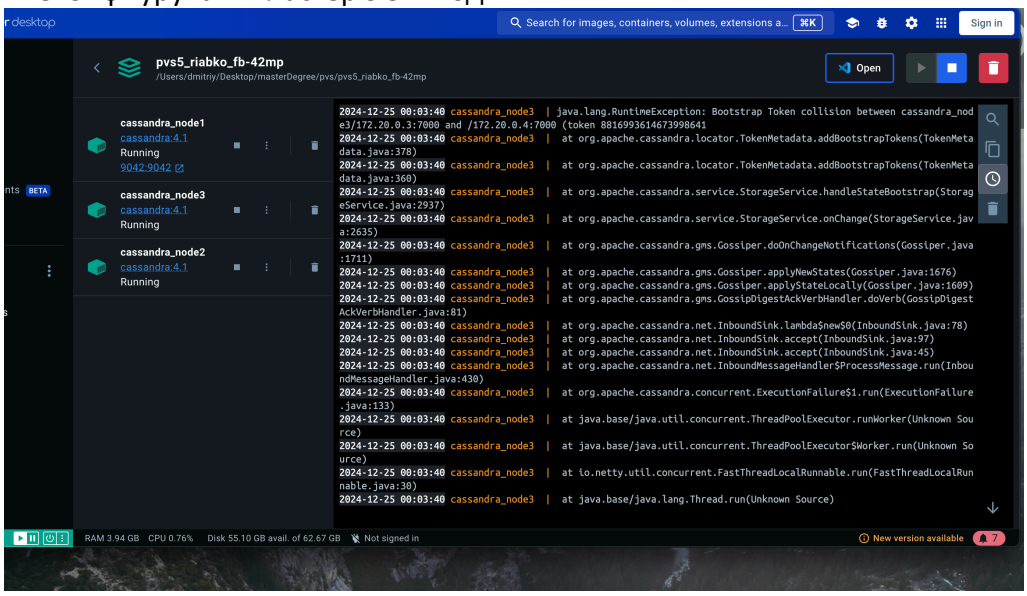
```
cqlsh:lab5> SELECT customer_name, SUM(total_price) AS total_spent
        ... FROM lab5.orders
        ... GROUP BY customer_name;

 customer_name | total_spent
---------------+-------------
    Jane Smith |      800.00
      John Doe |     3200.00

(2 rows)

Warnings :
Aggregation query used without partition key

cqlsh:lab5>
```

4. Для кожного замовлення виведіть час коли його ціна були занесена в базу (SELECT WRITETIME)

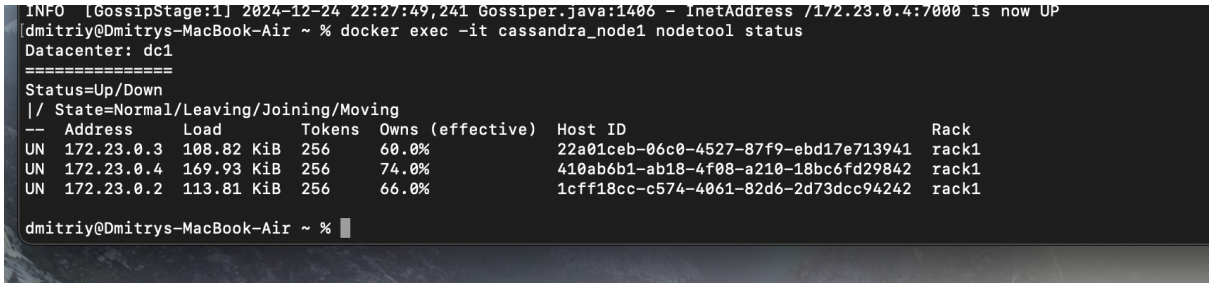
```
cqlsh:lab5> SELECT order_id, WRITETIME(total_price) AS write_time
        ... FROM lab5.orders
        ... WHERE customer_name = 'John Doe';

 order_id                             | write_time
--------------------------------------+------------------
 a9dc76b3-7630-46da-be4a-1090b45055a8 | 1735071630753511
 bd57034a-b950-41a0-af63-a5bac29341a5 | 1735071630641531

(2 rows)
```

Завдання

1. Сконфігурувати кластер з 3-х нод:



2. Перевірити правильність конфігурації за допомогою *nodetool status*



Дуже багато часу заняло те, щоб 3 нода змогла законектитись до інших.

Потрібно було декілька раз обмежувати ресурси для нод.

```
    CASSANDRA_ENDPOINT_SNITCH: "Goss
    MAX_HEAP_SIZE: "512M"
    HEAP_NEWSIZE: "100M"
  depends_on:
```

3. Використовуючи *cqlsh*, створити три *Keyspace* з replication factor 1, 2, 3 з SimpleStrategy

```
dmitriy@Dmitrys-MacBook-Air ~ % docker exec -it cassandra_node1 cqlsh
Connected to MyCluster at 127.0.0.1:9042
[cqlsh 6.1.0 | Cassandra 4.1.7 | CQL spec 3.4.6 | Native protocol v5]
Use HELP for help.
cqlsh> CREATE KEYSPACE keyspace1 WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 1};
cqlsh>
cqlsh> CREATE KEYSPACE keyspace2 WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 2};
cqlsh>
cqlsh> CREATE KEYSPACE keyspace3 WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 3};
cqlsh>
```

4. В кожному з кейспейсів створити прості таблиці

```
cqlsh> USE keyspace1;
cqlsh:keyspace1> CREATE TABLE test_table (id UUID PRIMARY KEY, value text);
cqlsh:keyspace1> USE keyspace2;
cqlsh:keyspace2> CREATE TABLE test_table (id UUID PRIMARY KEY, value text);
cqlsh:keyspace2> USE keyspace3;
cqlsh:keyspace3> CREATE TABLE test_table (id UUID PRIMARY KEY, value text);
cqlsh:keyspace3>
```

5. Спробуйте писати і читати в ці таблиці підключаюсь на різні ноди.

```
cqlsh:keyspace3>
cqlsh:keyspace3> USE keyspace1;
cqlsh:keyspace1> INSERT INTO test_table (id, value) VALUES (uuid(), 'Data in keyspace1');
cqlsh:keyspace1> USE keyspace2;
cqlsh:keyspace2> INSERT INTO test_table (id, value) VALUES (uuid(), 'Data in keyspace2');
cqlsh:keyspace2> USE keyspace3;
cqlsh:keyspace3> INSERT INTO test_table (id, value) VALUES (uuid(), 'Data in keyspace3');
cqlsh:keyspace3>
```

```
cqlsh:keyspace3> SELECT * FROM test_table;

 id                                   | value
--------------------------------------+-------------------
 37550b59-d1af-4362-b0e7-d387c6ad4e23 | Data in keyspace3

(1 rows)
cqlsh:keyspace3>
```

```
cqlsh:keyspace3> use keyspace1;
cqlsh:keyspace1> SELECT * FROM test_table;

 id                                   | value
--------------------------------------+-------------------
 da08a0c6-3cfd-4faa-8ea6-3c028f5b7bdd | Data in keyspace1

(1 rows)
cqlsh:keyspace1>
```

6. Вставте дані в створені таблиці і подивіться на їх розподіл по вузлах кластера для кожного з кейспесов (команда *nodetool status*)

```
Last login: Wed Dec 25 00:28:14 on ttys003
dmitriy@Dmitrys-MacBook-Air ~ % docker exec -it cassandra_node1 nodetool status
Datacenter: dc1
===============
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load        Tokens  Owns  Host ID                               Rack
UN  172.23.0.3  109.71 KiB  256     ?     22a01ceb-06c0-4527-87f9-ebd17e713941  rack1
UN  172.23.0.4  166.61 KiB  256     ?     410ab6b1-ab18-4f08-a210-18bc6fd29842  rack1
UN  172.23.0.2  114.69 KiB  256     ?     1cff18cc-c574-4061-82d6-2d73dcc94242  rack1

Note: Non-system keyspaces don't have the same replication settings, effective ownership information is meaningless
dmitriy@Dmitrys-MacBook-Air ~ %
```

7. Для якогось запису з кожного з кейспейсу виведіть ноди на яких зберігаються дані

```
cqlsh:keyspace1> USE keyspace3;
cqlsh:keyspace3> SELECT id, value FROM test_table;

 id                                   | value
--------------------------------------+-------------------
 37550b59-d1af-4362-b0e7-d387c6ad4e23 | Data in keyspace3

(1 rows)
cqlsh:keyspace3>
```

```
dmitriy@Dmitrys-MacBook-Air ~ % docker exec -it cassandra_node1 nodetool getendpoints keyspace3 test_table  37550b59-d1af-4362-b0e7-d387c6ad4e23

172.23.0.3
172.23.0.2
172.23.0.4
dmitriy@Dmitrys-MacBook-Air ~ %
```

```
cqlsh:keyspace3> use keyspace1;
cqlsh:keyspace1> SELECT id, value FROM test_table;

 id                                   | value
--------------------------------------+-------------------
 da08a0c6-3cfd-4faa-8ea6-3c028f5b7bdd | Data in keyspace1

(1 rows)
cqlsh:keyspace1>
```

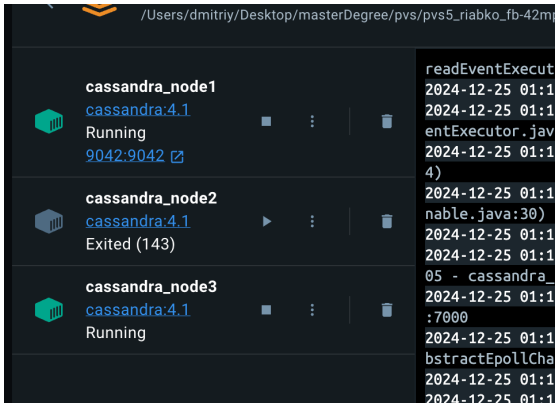
```
dmitriy@Dmitrys-MacBook-Air ~ % docker exec -it cassandra_node1 nodetool getendpoints keyspace3 test_table da08a0c6-3cfd-4faa-8ea6-3c028f5b7bdd
172.23.0.2
172.23.0.4
172.23.0.3
dmitriy@Dmitrys-MacBook-Air ~ %
```

```
cqlsh:keyspace1> use keyspace2;
cqlsh:keyspace2> SELECT id, value FROM test_table;

 id                                   | value
--------------------------------------+--------------------
 93b9888f-f339-4da1-8e5c-34f9a755b782 | Data in keyspace2

(1 rows)
cqlsh:keyspace2>
```

```
dmitriy@Dmitrys-MacBook-Air ~ % docker exec -it cassandra_node1 nodetool getendpoints keyspace3 test_table 93b9888f-f339-4da1-8e5c-34f9a755b782
172.23.0.2
172.23.0.4
172.23.0.3
dmitriy@Dmitrys-MacBook-Air ~ %
```

8. Відключити одну з нод. Для кожного з кейспейсів перевірити з якими рівнями *consistency* можемо читати та писати

- для *Keyspace* з replication factor 1 - **CONSISTENCY ONE**
- для *Keyspace* з replication factor 2 - **CONSISTENCY ONE/TWO**
- для *Keyspace* з replication factor 3 - **CONSISTENCY ONE/TWO/THREE**




```
Note: Non-system keyspaces don't have the same replication settings, effective ownership information is meaningless
dmitriy@Dmitrys-MacBook-Air ~ % docker exec -it cassandra_node1 nodetool status
Datacenter: dc1
===============
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load        Tokens  Owns  Host ID                               Rack
DN  172.23.0.3  230.21 KiB  256     ?     22a01ceb-06c0-4527-87f9-ebd17e713941  rack1
UN  172.23.0.4  276.93 KiB  256     ?     410ab6b1-ab18-4f08-a210-18bc6fd29842  rack1
UN  172.23.0.2  177.44 KiB  256     ?     1cff18cc-c574-4061-82d6-2d73dcc94242  rack1

Note: Non-system keyspaces don't have the same replication settings, effective ownership information is meaningless
```

Бачимо, що одна нода в статусі DN

```
cqlsh> CONSISTENCY ONE;
Consistency level set to ONE.
cqlsh> USE keyspace1;
cqlsh:keyspace1> CONSISTENCY ONE;
Consistency level set to ONE.
cqlsh:keyspace1> INSERT INTO test_table (id, value) VALUES (uuid(), 'Test for RF=1');
cqlsh:keyspace1> SELECT * FROM test_table;
NoHostAvailable: ('Unable to complete the operation against any hosts', {<Host: 127.0.0.1:9042 dc1>: Unavailable('Error from server: code=1000 [Unavailable exception] message="Cannot achieve consistency level ONE" info={\'consistency\': \'ONE\', \'required_replicas\': 1, \'alive_replicas\': 0}')})
cqlsh:keyspace1>
```

```
cqlsh:keyspace1> USE keyspace2;
cqlsh:keyspace2> CONSISTENCY ONE;
Consistency level set to ONE.
cqlsh:keyspace2> INSERT INTO test_table (id, value) VALUES (uuid(), 'Test for RF=2 consistency ONE');
cqlsh:keyspace2> SELECT * FROM test_table;

 id                                   | value
--------------------------------------+-----------------------------
 80729315-8d88-40e4-ba98-6bf4f58893ef | Test for RF=2 consistency ONE
 93b9888f-f339-4da1-8e5c-34f9a755b782 |             Data in keyspace2

(2 rows)
cqlsh:keyspace2>
```

```
cqlsh:keyspace2> CONSISTENCY TWO;
Consistency level set to TWO.
cqlsh:keyspace2> INSERT INTO test_table (id, value) VALUES (uuid(), 'Test for RF=2 consistency TWO');
NoHostAvailable: ('Unable to complete the operation against any hosts', {<Host: 127.0.0.1:9042 dc1>: Unavailable('Error from server: code=1000 [Un
available exception] message="Cannot achieve consistency level TWO" info={\'consistency\': \'TWO\', \'required_replicas\': 2, \'alive_replicas\':
1}')})
cqlsh:keyspace2> SELECT * FROM test_table;
NoHostAvailable: ('Unable to complete the operation against any hosts', {<Host: 127.0.0.1:9042 dc1>: Unavailable('Error from server: code=1000 [Un
available exception] message="Cannot achieve consistency level TWO" info={\'consistency\': \'TWO\', \'required_replicas\': 2, \'alive_replicas\':
1}')})
cqlsh:keyspace2>
```

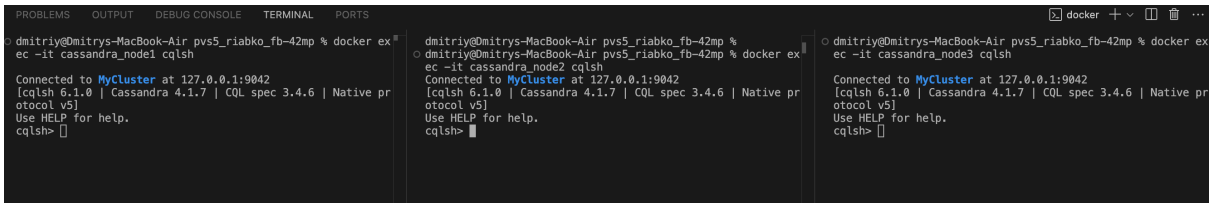
```
cqlsh:keyspace2> USE keyspace3;
cqlsh:keyspace3> CONSISTENCY ONE;
Consistency level set to ONE.
cqlsh:keyspace3> INSERT INTO test_table (id, value) VALUES (uuid(), 'Test for RF=3 consistency ONE');
cqlsh:keyspace3> CONSISTENCY TWO;
Consistency level set to TWO.
cqlsh:keyspace3> INSERT INTO test_table (id, value) VALUES (uuid(), 'Test for RF=3 consistency TWO');
cqlsh:keyspace3> CONSISTENCY THREE;
Consistency level set to THREE.
cqlsh:keyspace3> INSERT INTO test_table (id, value) VALUES (uuid(), 'Test for RF=3 consistency THREE')
;
NoHostAvailable: ('Unable to complete the operation against any hosts', {<Host: 127.0.0.1:9042 dc1>: U
navailable('Error from server: code=1000 [Unavailable exception] message="Cannot achieve consistency l
evel THREE" info={\'consistency\': \'THREE\', \'required_replicas\': 3, \'alive_replicas\': 2}')})
cqlsh:keyspace3> SELECT * FROM test_table;
NoHostAvailable: ('Unable to complete the operation against any hosts', {<Host: 127.0.0.1:9042 dc1>: U
navailable('Error from server: code=1000 [Unavailable exception] message="Cannot achieve consistency l
evel THREE" info={\'consistency\': \'THREE\', \'required_replicas\': 3, \'alive_replicas\': 2}')})
cqlsh:keyspace3>
```

9. Зробить так щоб три ноди працювали, але не бачили одна одну по мережі (заблокуйте чи відключити зв'язок між ними)

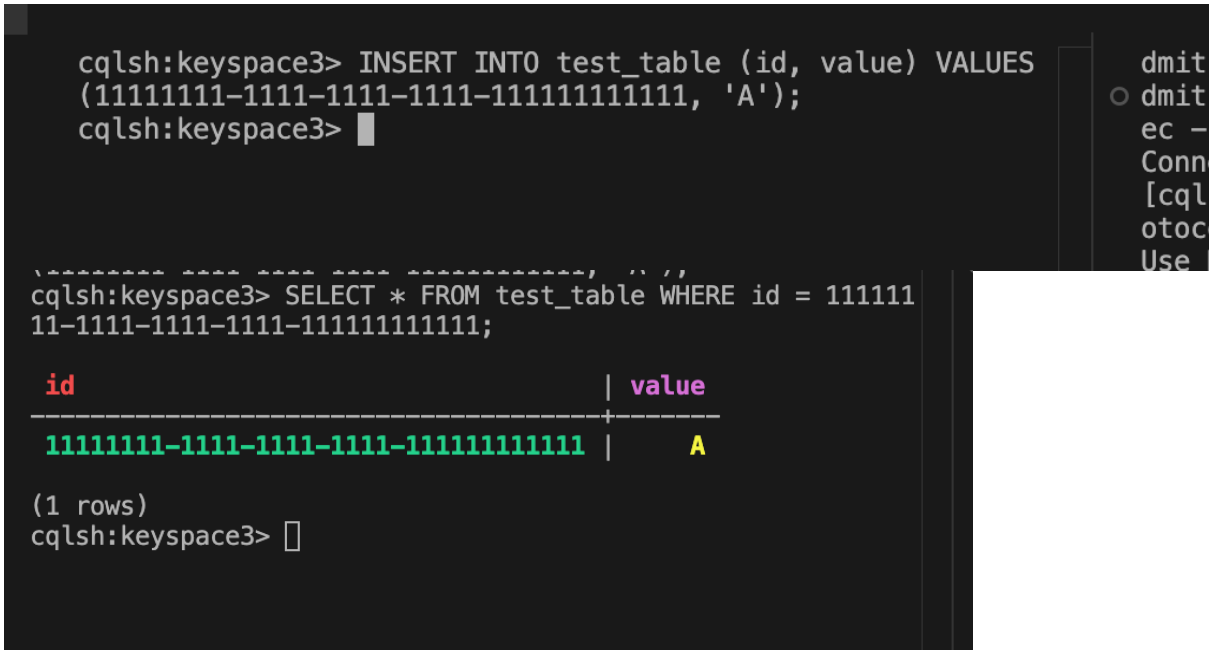
```
dmitriy@Dmitrys-MacBook-Air ~ % docker network disconnect pvs5_riabko_fb-42mp_ca
ssandra_network cassandra_node1
docker network disconnect pvs5_riabko_fb-42mp_cassandra_network cassandra_node2
docker network disconnect pvs5_riabko_fb-42mp_cassandra_network cassandra_node3

dmitriy@Dmitrys-MacBook-Air ~ %
```

```
dmitriy@Dmitrys-MacBook-Air ~ % docker network inspect pvs5_riabko_fb-42mp_cassandra_network
[
    {
        "Name": "pvs5_riabko_fb-42mp_cassandra_network",
        "Id": "232e2bd66e8c35c1f76fcd91d92780b7a80a66bf5e2ae86c8cea7f582ddfa15a",
        "Created": "2024-12-24T22:19:43.952308625Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.23.0.0/16",
                    "Gateway": "172.23.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {},
        "Options": {},
        "Labels": {
            "com.docker.compose.network": "cassandra_network",
            "com.docker.compose.project": "pvs5_riabko_fb-42mp",
            "com.docker.compose.version": "2.24.6"
        }
    }
]
```

10. Для кейспейсу з *replication factor* 3 задайте рівень consistency рівним 1. Виконайте по черзі запис значення з однаковим primary key, але різними іншими значенням окремо на кожну з нод (тобто створіть конфлікт)



Node 1:



Node2:

```
Consistency level set to ONE.
cqlsh:keyspace3> INSERT INTO test_table (id, value) VALUES
(11111111-1111-1111-1111-111111111111, 'B');
cqlsh:keyspace3>
cqlsh:keyspace3> SELECT * FROM test_table WHERE id = 111111
11-1111-1111-1111-111111111111;

 id                                   | value
--------------------------------------+-------
 11111111-1111-1111-1111-111111111111 |     B

(1 rows)
cqlsh:keyspace3>
```

Node 3:

```
 cqlsh:keyspace3> CONSISTENCY ONE;
 Consistency level set to ONE.
 cqlsh:keyspace3> INSERT INTO test_table (id, value) VALUES
 (11111111-1111-1111-1111-111111111111, 'C');
 cqlsh:keyspace3>
```

```
cqlsh:keyspace3> SELECT * FROM test_table WHERE id = 111111
11-1111-1111-1111-111111111111;

 id                                   | value
--------------------------------------+-------
 11111111-1111-1111-1111-111111111111 |     C

(1 rows)
cqlsh:keyspace3>
```

11. Відновіть зв'язок між нодами, і перевірте що вони знову об'єдналися у кластер. Визначте яким чином була вирішений конфлікт даних та яке значення було прийнято кластером та за яким принципом

```
dmitriy@Dmitrys-MacBook-Air ~ % docker network connect pvs5_riabko_fb-42mp_cassandra_
network cassandra_node1
docker network connect pvs5_riabko_fb-42mp_cassandra_network cassandra_node2
docker network connect pvs5_riabko_fb-42mp_cassandra_network cassandra_node3

dmitriy@Dmitrys-MacBook-Air ~ % docker exec -it cassandra_node1 nodetool status

Datacenter: dc1
===============
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load        Tokens  Owns  Host ID                               Rack
UN  172.23.0.3  245.74 KiB  256     ?     22a01ceb-06c0-4527-87f9-ebd17e713941  rack1
UN  172.23.0.4  256.36 KiB  256     ?     410ab6b1-ab18-4f08-a210-18bc6fd29842  rack1
UN  172.23.0.2  177.44 KiB  256     ?     1cff18cc-c574-4061-82d6-2d73dcc94242  rack1

Note: Non-system keyspaces don't have the same replication settings, effective owners
hip information is meaningless
dmitriy@Dmitrys-MacBook-Air ~ %
```

```
dmitriy@Dmitrys-MacBook-Air ~ % docker exec -it cassandra_node1 cqlsh
```

```
cqlsh:keyspace3> USE keyspace3;
cqlsh:keyspace3> SELECT * FROM test_table WHERE id = 11111111-1111-1111-1111-111111111111;

 id                                   | value
--------------------------------------+-------
 11111111-1111-1111-1111-111111111111 |     C

(1 rows)
cqlsh:keyspace3>
```

Після відновлення зв'язку між нодами Cassandra автоматично синхронізувала дані між усіма нодами. Конфлікт був вирішений за допомогою принципу **"Last Write Wins"**: значення з найбільшим timestamp було обрано як фінальне. Фінальне значення (value) було репліковано на всі ноди.