

Below screenshots which help to understand the completed test assignment:

1. Creating keyspace in Cassandra

```
C:\Python27\python.exe
WARNING: console codepage must be set to cp65001 to support utf-8 encoding on Windows platforms.
If you experience encoding problems, change your console codepage with 'chcp 65001' before starting cqlsh.

Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.3 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
WARNING: pyreadline dependency missing. Install to enable tab completion.
cqlsh> CREATE KEYSPACE test_keyspace WITH replication = {'class': 'SimpleStrategy','replication_factor': '1'} AND durable_writes = 'true';
cqlsh> DESCRIBE KEYSPACES

system_schema system test_keyspace
system_auth system_distributed system_traces

cqlsh>
```

2. Created tables

```
CREATE TABLE test_keyspace.orders (  
    id text PRIMARY KEY,  
    customerid text,  
    itemid text,  
    note text,  
    ordertotal double,  
    quantity int,  
    shippingcost double,  
    status int,  
    timestamp bigint  
) WITH bloom_filter_fp_chance = 0.01  
AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}  
AND comment = ''  
AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', 'min_threshold': '4'}  
AND compression = {'chunk_length_in_kb': '64', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}  
AND crc_check_chance = 1.0  
AND dclocal_read_repair_chance = 0.1  
AND default_time_to_live = 0  
AND gc_grace_seconds = 864000  
AND max_index_interval = 2048  
AND memtable_flush_period_in_ms = 0  
AND min_index_interval = 128  
AND read_repair_chance = 0.0  
AND speculative_retry = '99PERCENTILE';
```

```
cqlsh:test_keyspace> select * from orders;
```

id	customerid	itemid	note	ordertotal	quantity	shippingcost	status	timestamp
-----+-----+-----+-----+-----+-----+-----+-----+-----								

```
(0 rows)  
cqlsh:test_keyspace>
```

3. Data inserted by Create Order API

```
mysqlsh:test_keyspace>
mysqlsh:test_keyspace>
mysqlsh:test_keyspace> select * from orders;

id | timestamp | customerid | itemid | note | ordertotal | quantity | shippingcost | status
-----+-----+-----+-----+-----+-----+-----+-----+-----
401a85e2-f5da-494b-9f3f-756a91982f00 | 2021-02-25T16:27:45.322+0000 | 0 | 1612487464353 | test | 10.23 | 1 | 3.45 | 0
401a85e2-f5da-494b-9f3f-756a91982f10 | 2021-02-25T16:27:45.322+0000 | 0 | 1612487464353 | test | 13.23 | 1 | 4.45 | 0
(2 rows)
mysqlsh:test_keyspace> select * from orders;

id | timestamp | customerid | itemid | note | ordertotal | quantity | shippingcost | status
-----+-----+-----+-----+-----+-----+-----+-----+-----
401a85e2-f5da-494b-9f3f-756a91982f22 | 2021-02-25T16:27:45.322+0000 | 0 | 1612487464353 | test value | 17.23 | 1 | 4.45 | 0
401a85e2-f5da-494b-9f3f-756a91982f55 | 2021-02-25T16:27:45.322+0000 | 0 | 1612487464353 | test value third | 17.23 | 1 | 4.45 | 0
401a85e2-f5da-494b-9f3f-756a91982f00 | 2021-02-25T16:27:45.322+0000 | 0 | 1612487464353 | test | 10.23 | 1 | 3.45 | 0
401a85e2-f5da-494b-9f3f-756a91982f99 | 2021-02-25T16:27:45.322+0000 | 0 | 1612487464353 | test value | 17.23 | 1 | 4.45 | 0
401a85e2-f5da-494b-9f3f-756a91982f10 | 2021-02-25T16:27:45.322+0000 | 0 | 1612487464353 | test | 13.23 | 1 | 4.45 | 0
(5 rows)
mysqlsh:test_keyspace>
```

4. Testing GetOrderDetails API (404 Not Found Exception)

The image shows the Postman application interface. The left sidebar displays the 'My Workspace' with a collection named 'Order API' containing a request named 'GET Get Order Details'. The main panel shows the details of this request, which is a GET request to the endpoint `http://localhost:9001/api/orders/401a85e2-f5da-494b-9f3f-756a91982f00`. The 'Body' tab is selected, showing the response in JSON format. The response status is '404 Not Found', and the message is 'Not Found'. The response body is as follows:

```
{
  "timestamp": "2021-02-25T16:27:45.322+0000",
  "path": "/api/orders/401a85e2-f5da-494b-9f3f-756a91982f00",
  "status": 404,
  "error": "Not Found",
  "message": "",
  "requestId": "6fb3db8d"
}
```

5. Testing GetOrderDetails API (400 Bad Request)

The screenshot shows the Postman interface with a REST client request configured for the **GET** method to the endpoint `http://localhost:9001/api/orders/401a85e2-f5da-494b-9f3f-`. The **Headers** tab is active, showing 7 headers. The **Body** tab is also active, displaying a JSON response in the **Pretty** view:

```
1 {
2   "timestamp": "2021-02-25T18:08:19.513+0000",
3   "path": "/api/orders/401a85e2-f5da-494b-9f3f-",
4   "status": 400,
5   "error": "Bad Request",
6   "message": "Type mismatch.",
7   "requestId": "32c9ee24"
8 }
```

The status bar at the bottom indicates a **Status: 400 Bad Request** with a time of 11 ms and a size of 256 B. The **Save Response** button is visible.

6. Testing GetOrderDetails API (200 OK) : get Data as per OrderId

The screenshot shows the Postman interface with a REST client request configured for the **GET** method to the endpoint `http://localhost:9001/api/orders/401a85e2-f5da-494b-9f3f-756a91982f00`. The **Headers** tab is active, showing 7 headers. The **Body** tab is also active, displaying a JSON response in the **Pretty** view:

```
1 {
2   "id": "401a85e2-f5da-494b-9f3f-756a91982f00",
3   "timestamp": 1612487464353,
4   "status": 0,
5   "orderTotal": 10.29,
6   "shippingCost": 3.45,
7   "customerId": "401a85e2-f5da-494b-9f3f-756a91982f01",
8   "itemId": "401a85e2-f5da-494b-9f3f-756a91982f02",
9   "quantity": 1,
10  "note": "test"
11 }
```

The status bar at the bottom indicates a **Status: 200 OK** with a time of 38 ms and a size of 320 B. The **Save Response** button is visible.

7. Testing CreateOrders (500 Internal Server Error)

The screenshot shows the Postman interface with a POST request to `http://localhost:9001/api/orders`. The request body is a JSON object with the following fields:

```
{  "id": "",  "timestamp": 1612487464353,  "status": 0,  "orderTotal": 11.23,  "shippingCost": 4.45,}
```

The response status is **500 Internal Server Error** (Time: 81 ms, Size: 614 B). The response body is a JSON object with the following fields:

```
{  "timestamp": "2021-02-25T17:38:06.708+0000",  "path": "/api/orders",  "status": 500,  "error": "Internal Server Error",  "message": "ReactiveSessionCallback; CQL [INSERT INTO orders (customerid,id,itemid,note,ordertotal,quantity,shippingcost,status,timestamp) VALUES ('401a85e2-f5da-494b-9f3f-756a91982f11','401a85e2-f5da-494b-9f3f-756a91982f04','test',11.23,1,4.45,0,1612487464353)]; Key may not be empty; nested exception is com.datastax.driver.core.exceptions.InvalidQueryException: Key may not be empty;",  "requestId": "000098c1"}
```

8. Testing CreateOrders (400 Bad Request)

The screenshot shows the Postman interface with a POST request to `http://localhost:9001/api/orders`. The request body is a JSON object with the following fields:

```
{  "id": "401a85e2-f5da-494b-9f3f-756a91982f10",  "timestamp": 2020-10-29T15:49:21.581+00:00,  "status": 0,  "orderTotal": 11.23,  "shippingCost": 4.45,}
```

The response status is **400 Bad Request** (Time: 11 ms, Size: 244 B). The response body is a JSON object with the following fields:

```
{  "timestamp": "2021-02-25T17:39:44.297+0000",  "path": "/api/orders",  "status": 400,  "error": "Bad Request",  "message": "Failed to read HTTP message",  "requestId": "000098c1"}
```

9. Testing CreateOrders (200 OK) : create 2 orders

The image shows the Postman application interface. On the left sidebar, under 'My Workspace', there is a collection named 'Order API' with two items: 'GET Get Order Details' and 'POST Create Orders'. The 'POST Create Orders' item is selected. The main panel shows the request details for 'POST http://localhost:9001/api/orders'. The request body is a JSON array with two objects, each representing an order item. The status bar at the bottom indicates a successful response with status 201 Created, time 21 ms, and size 599 B.

```
POST http://localhost:9001/api/orders

{
  "id": "481a85e2-f5da-494b-9f3f-756a91982f",
  "timestamp": 1612487464353,
  "status": 0,
  "orderTotal": 17.23,
  "shippingCost": 6.45,
  "customerId": "481a85e2-f5da-494b-9f3f-756a91982f33",
  "itemId": "481a85e2-f5da-494b-9f3f-756a91982f44",
  "quantity": 1,
  "note": "test value"
},
{
  "id": "481a85e2-f5da-494b-9f3f-756a91982f55",
  "timestamp": 1612487464353,
  "status": 0,
  "orderTotal": 17.23,
  "shippingCost": 6.45,
  "customerId": "481a85e2-f5da-494b-9f3f-756a91982f33",
  "itemId": "481a85e2-f5da-494b-9f3f-756a91982f44",
  "quantity": 1,
  "note": "test value"
}
```

Status: 201 Created Time: 21 ms Size: 599 B Save Response