Name: Pradeep Medagiri

You are assigned to create a NoSQL key-value database for a product catalog using AWS DynamoDB for a new store located in Northern Virginia. As is normally done in development stage, you start with a handful of items to test if the database works. Here are the parameters:

Table name: yourlastnameProductCatalog

Primary Key:

- Partition Key: ID (attribute type: number)

Global Secondary Index

- ProductCategory (attribute type: string)

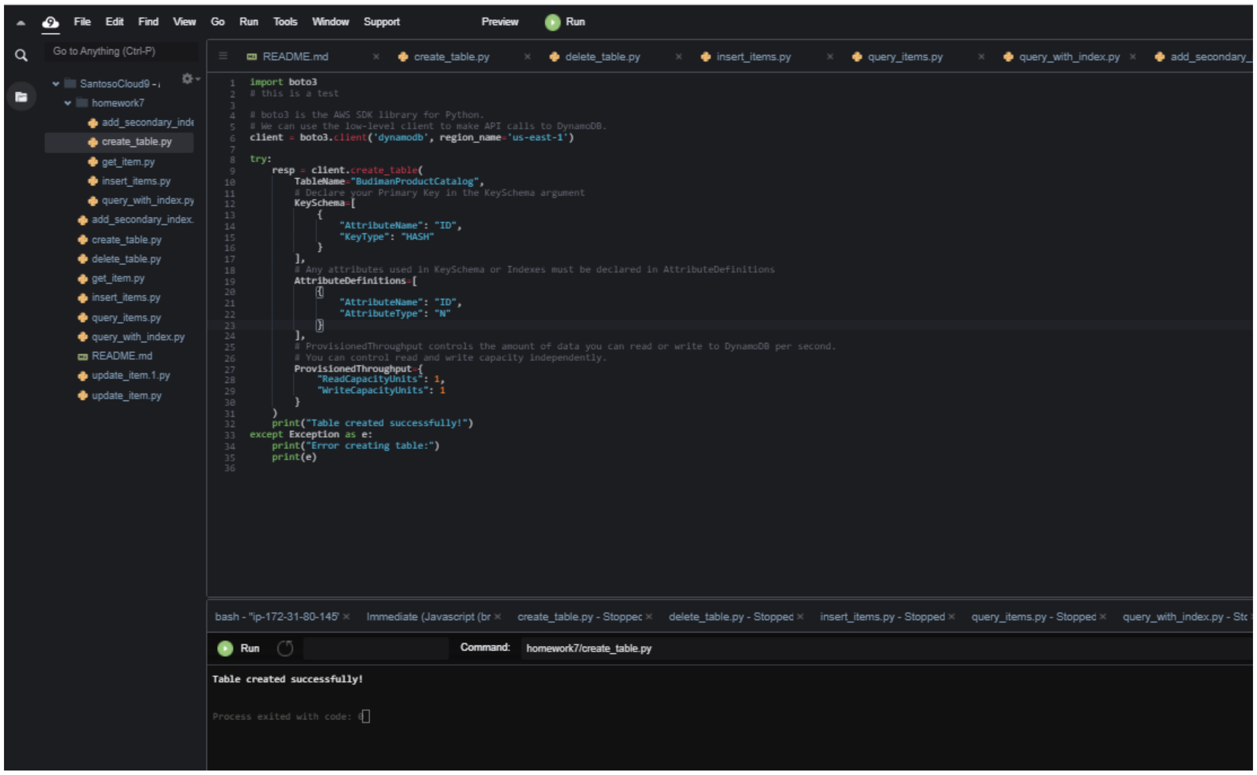
Use Cloud9 or whatever IDE you prefer and create python scripts to:

- Create the dynamoDB table
- Define the Primary Key and Global Secondary Index
- To insert the 6 items below
- To query item ID number 103
- To query all items in the Bicycle category

What to submit:

1. The screenshot of the DynamoDB table creation python script and show that it runs successfully (show the file editor and the terminal). (5)
2. The screenshot of the Global Secondary Index definition python script and show that it runs successfully (show the file editor and the terminal). (5)
3. The screenshot of the item insertion python script and show that it runs successfully (show the file editor and the terminal). (5)
4. The screenshot of the query item ID number 103 python script and show the result (file editor and terminal). (5)
5. The screenshot of the query all items in the Bicycle category python script and show the result (file editor and terminal. (5)

Example of screenshot:



```python
import boto3
# this is a test

# boto3 is the AWS SDK library for Python.
# We can use the low-level client to make API calls to DynamoDB.
client = boto3.client('dynamodb', region_name='us-east-1')

try:
    resp = client.create_table(
        TableName="BudimanProductCatalog",
        # Declare your Primary Key in the KeySchema argument
        KeySchema=[
            {
                "AttributeName": "ID",
                "KeyType": "HASH"
            }
        ],
        # Any attributes used in KeySchema or Indexes must be declared in AttributeDefinitions
        AttributeDefinitions=[
            {
                "AttributeName": "ID",
                "AttributeType": "N"
            }
        ],
        # ProvisionedThroughput controls the amount of data you can read or write to DynamoDB per second.
        # You can control read and write capacity independently.
        ProvisionedThroughput={
            "ReadCapacityUnits": 1,
            "WriteCapacityUnits": 1
        }
    )
    print("Table created successfully!")
except Exception as e:
    print("Error creating table:")
    print(e)
```

Command: homework7/create_table.py

```
Table created successfully!

Process exited with code: 0
```

Note: **if there is no " ", the value is not a string**

1st Item

"Id": 101
"Title": "Book 101 Title"
"ISBN":"111-1111111111"
"Authors": "Author1"
"Price": 2
"Dimensions": "8.5 x 11.0 x 0.5"
"PageCount": 500
"InPublication": true
"ProductCategory": "Book"

2nd item

"Id": 102
"Title": "Book 102 Title"
"ISBN": "222-2222222222"
"Authors": "Author2
"Price": 20
"Dimensions": "8.5 x 11.0 x 0.8"
"PageCount": 600
"InPublication": true
"ProductCategory": "Book"

3rd item

```
"Id": 103
"Title":  "Book 103 Title"
"ISBN": "333-3333333333"
"Authors": "Author1"
"Price": 2000
"Dimensions": "8.5 x 11.0 x 1.5"
"PageCount": 600
"InPublication": false
"ProductCategory": "Book"
```

4th item

```
"Id": 201
"Title": "18-Bike-201"
"Description": "201 Description"
"BicycleType": "Road"
"Brand": "Mountain A"
"Price": 100
"Color":  ["Red",”Black”]
"ProductCategory": "Bicycle"
```

5th item

```
"Id": 202
“Title": "21-Bike-202"
"Description": "202 Description"
"BicycleType": "Road"
"Brand":  "Brand-Company A"
"Price": 200
"Color": ["Green", "Black"]
"ProductCategory": "Bicycle"
```

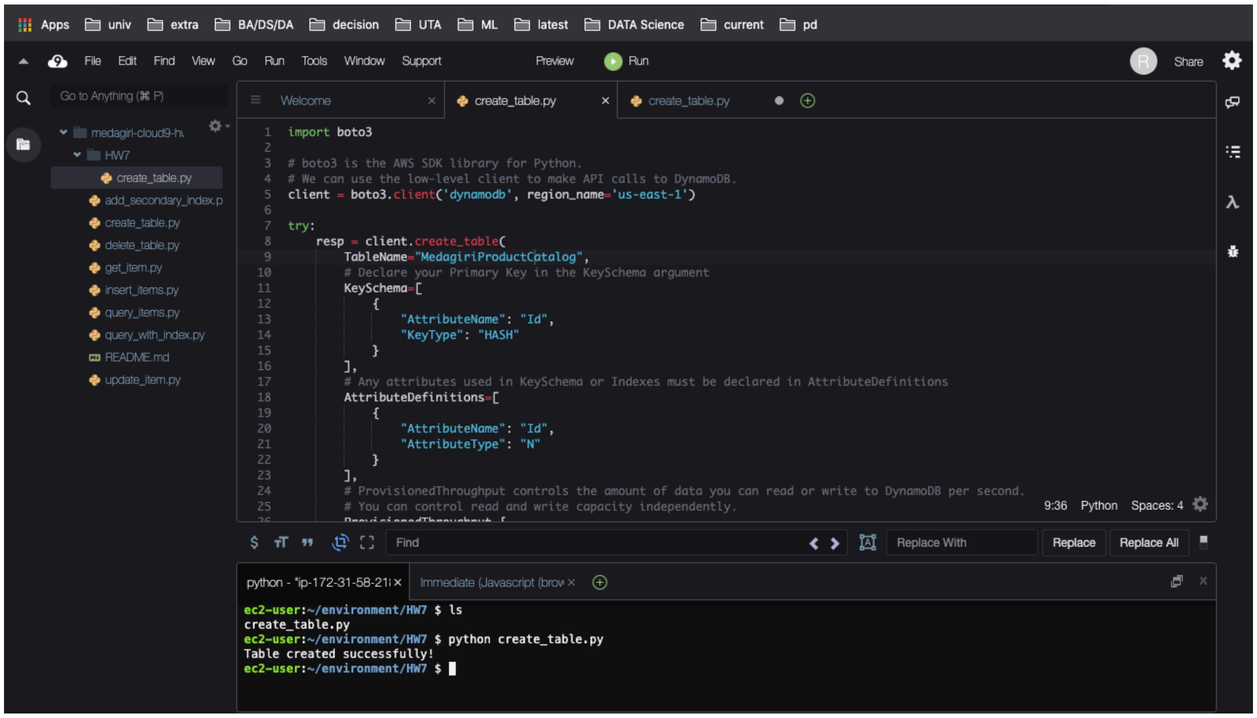
6th item

```
"Id": 203
"Title": "19-Bike-203"
"Description": "203 Description"
"BicycleType": "Road"
"Brand": "Brand-Company B"
"Price": 300
"Color": ["Red","Green","Black"]
"ProductCategory": "Bicycle"
```

Answers:

What to submit:

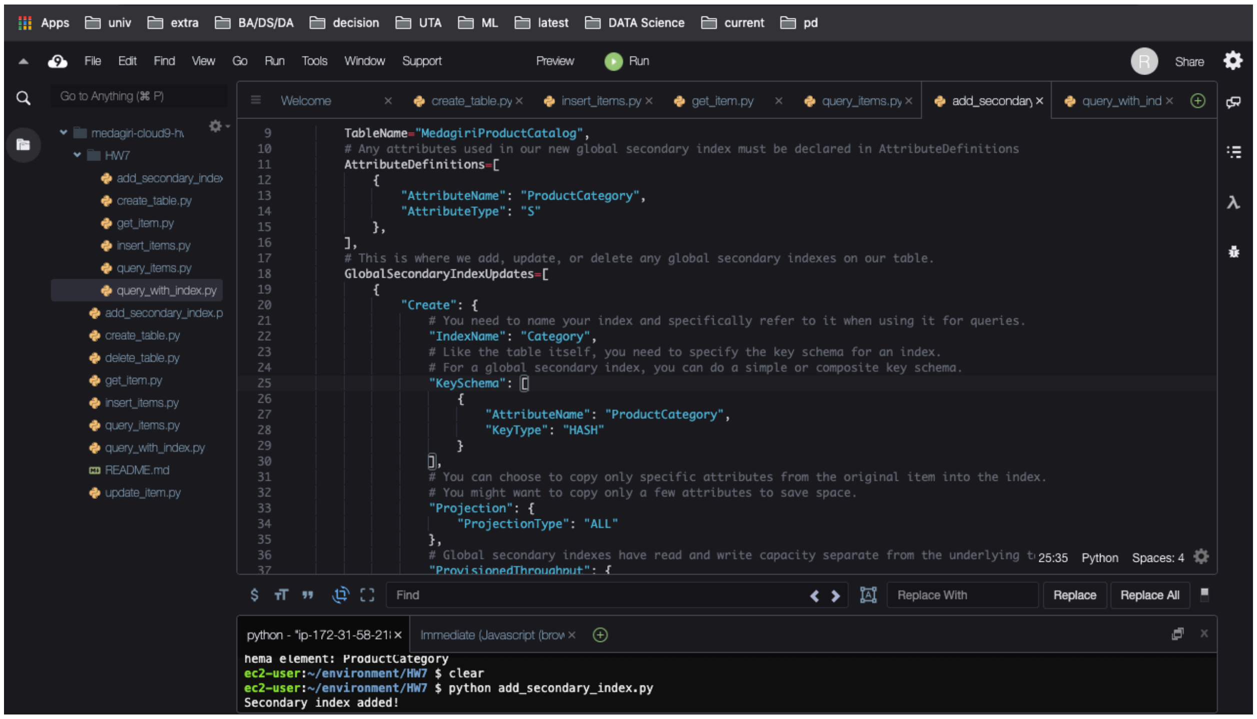
1. The screenshot of the DynamoDB table creation python script and show that it runs successfully (show the file editor and the terminal). (5)



```python
import boto3

# boto3 is the AWS SDK library for Python.
# We can use the low-level client to make API calls to DynamoDB.
client = boto3.client('dynamodb', region_name='us-east-1')

try:
    resp = client.create_table(
        TableName="MedagiriProductCatalog",
        # Declare your Primary Key in the KeySchema argument
        KeySchema=[
            {
                "AttributeName": "Id",
                "KeyType": "HASH"
            }
        ],
        # Any attributes used in KeySchema or Indexes must be declared in AttributeDefinitions
        AttributeDefinitions=[
            {
                "AttributeName": "Id",
                "AttributeType": "N"
            }
        ],
        # ProvisionedThroughput controls the amount of data you can read or write to DynamoDB per second.
        # You can control read and write capacity independently.
```

```
ec2-user:~/environment/HW7 $ ls
create_table.py
ec2-user:~/environment/HW7 $ python create_table.py
Table created successfully!
ec2-user:~/environment/HW7 $ 
```

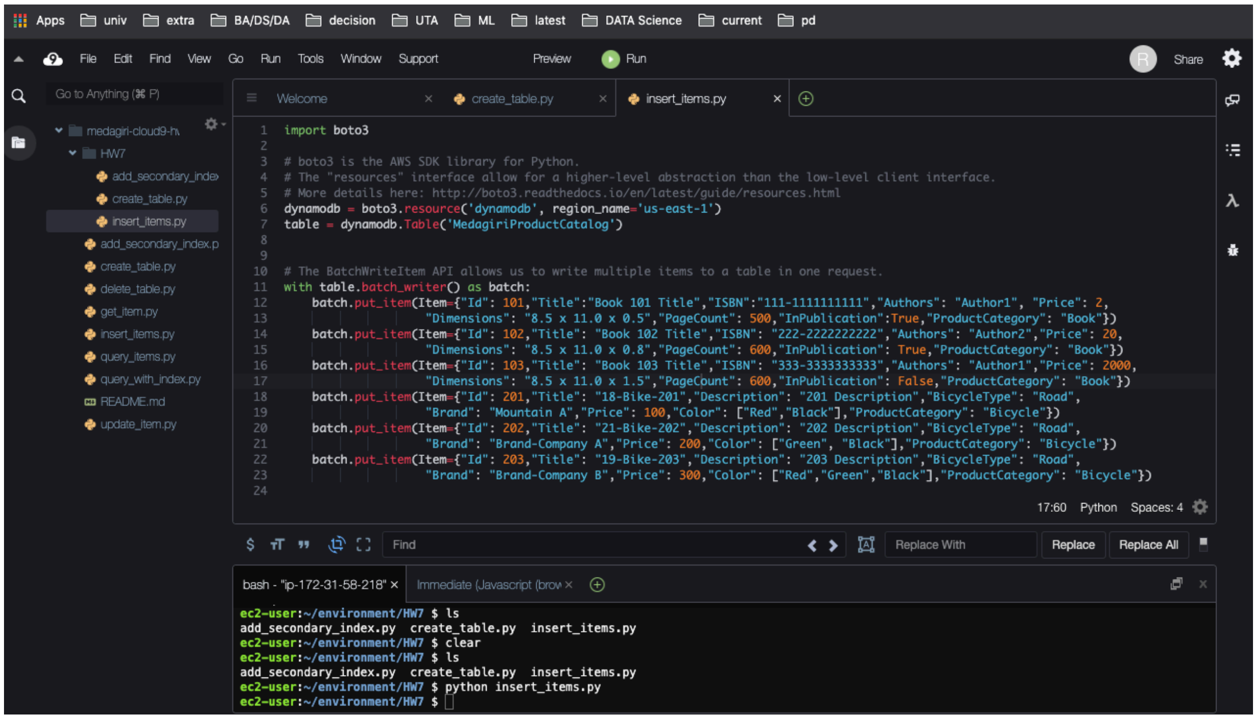
2. The screenshot of the Global Secondary Index definition python script and show that it runs successfully (show the file editor and the terminal). (5)



```python
        TableName="MedagiriProductCatalog",
        # Any attributes used in our new global secondary index must be declared in AttributeDefinitions
        AttributeDefinitions=[
            {
                "AttributeName": "ProductCategory",
                "AttributeType": "S"
            },
        ],
        # This is where we add, update, or delete any global secondary indexes on our table.
        GlobalSecondaryIndexUpdates=[
            {
                "Create": {
                    # You need to name your index and specifically refer to it when using it for queries.
                    "IndexName": "Category",
                    # Like the table itself, you need to specify the key schema for an index.
                    # For a global secondary index, you can do a simple or composite key schema.
                    "KeySchema": [
                        {
                            "AttributeName": "ProductCategory",
                            "KeyType": "HASH"
                        }
                    ],
                    # You can choose to copy only specific attributes from the original item into the index.
                    # You might want to copy only a few attributes to save space.
                    "Projection": {
                        "ProjectionType": "ALL"
                    },
                    # Global secondary indexes have read and write capacity separate from the underlying t
                    "ProvisionedThroughput": {
```

```
hema element: ProductCategory
ec2-user:~/environment/HW7 $ clear
ec2-user:~/environment/HW7 $ python add_secondary_index.py
Secondary index added!
```

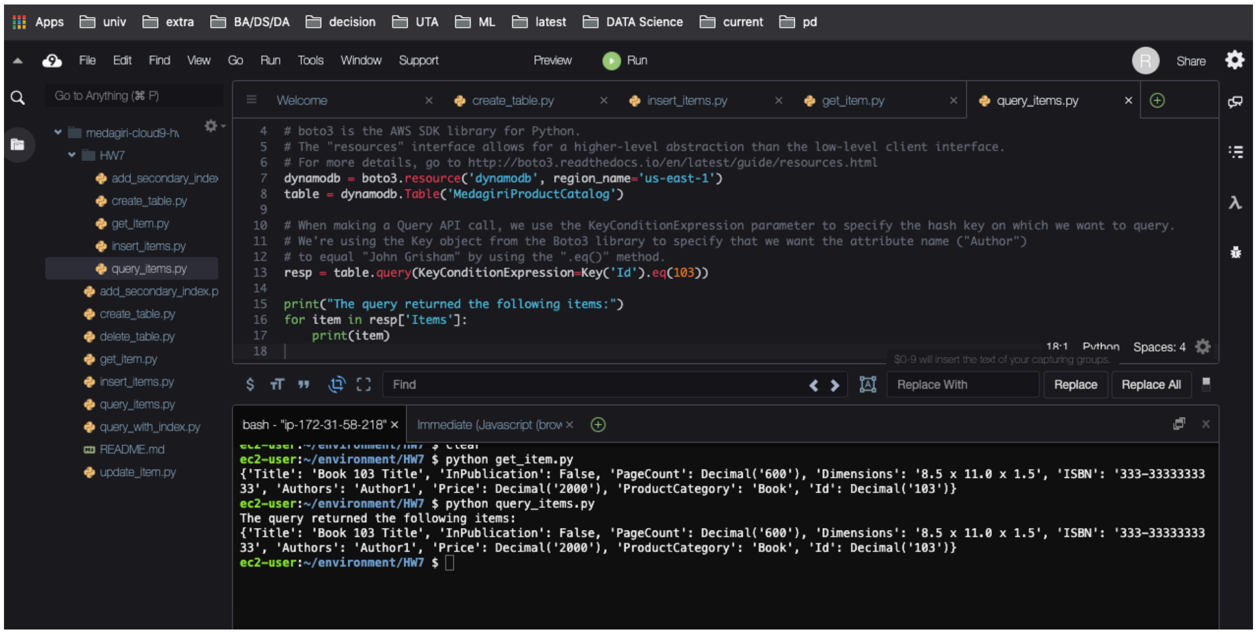
3. The screenshot of the item insertion python script and show that it runs successfully (show the file editor and the terminal). (5)



```python
import boto3

# boto3 is the AWS SDK library for Python.
# The "resources" interface allow for a higher-level abstraction than the low-level client interface.
# More details here: http://boto3.readthedocs.io/en/latest/guide/resources.html
dynamodb = boto3.resource('dynamodb', region_name='us-east-1')
table = dynamodb.Table('MedagiriProductCatalog')


# The BatchWriteItem API allows us to write multiple items to a table in one request.
with table.batch_writer() as batch:
    batch.put_item(Item={"Id": 101,"Title":"Book 101 Title","ISBN":"111-1111111111","Authors": "Author1", "Price": 2,
                    "Dimensions": "8.5 x 11.0 x 0.5","PageCount": 500,"InPublication":True,"ProductCategory": "Book"})
    batch.put_item(Item={"Id": 102,"Title": "Book 102 Title","ISBN": "222-2222222222","Authors": "Author2","Price": 20,
                    "Dimensions": "8.5 x 11.0 x 0.8","PageCount": 600,"InPublication": True,"ProductCategory": "Book"})
    batch.put_item(Item={"Id": 103,"Title": "Book 103 Title","ISBN": "333-3333333333","Authors": "Author1","Price": 2000,
                    "Dimensions": "8.5 x 11.0 x 1.5","PageCount": 600,"InPublication": False,"ProductCategory": "Book"})
    batch.put_item(Item={"Id": 201,"Title": "18-Bike-201","Description": "201 Description","BicycleType": "Road",
                    "Brand": "Mountain A","Price": 100,"Color": ["Red","Black"],"ProductCategory": "Bicycle"})
    batch.put_item(Item={"Id": 202,"Title": "21-Bike-202","Description": "202 Description","BicycleType": "Road",
                    "Brand": "Brand-Company A","Price": 200,"Color": ["Green", "Black"],"ProductCategory": "Bicycle"})
    batch.put_item(Item={"Id": 203,"Title": "19-Bike-203","Description": "203 Description","BicycleType": "Road",
                    "Brand": "Brand-Company B","Price": 300,"Color": ["Red","Green","Black"],"ProductCategory": "Bicycle"})

```

```
ec2-user:~/environment/HW7 $ ls
add_secondary_index.py  create_table.py  insert_items.py
ec2-user:~/environment/HW7 $ clear
ec2-user:~/environment/HW7 $ ls
add_secondary_index.py  create_table.py  insert_items.py
ec2-user:~/environment/HW7 $ python insert_items.py
ec2-user:~/environment/HW7 $ 
```

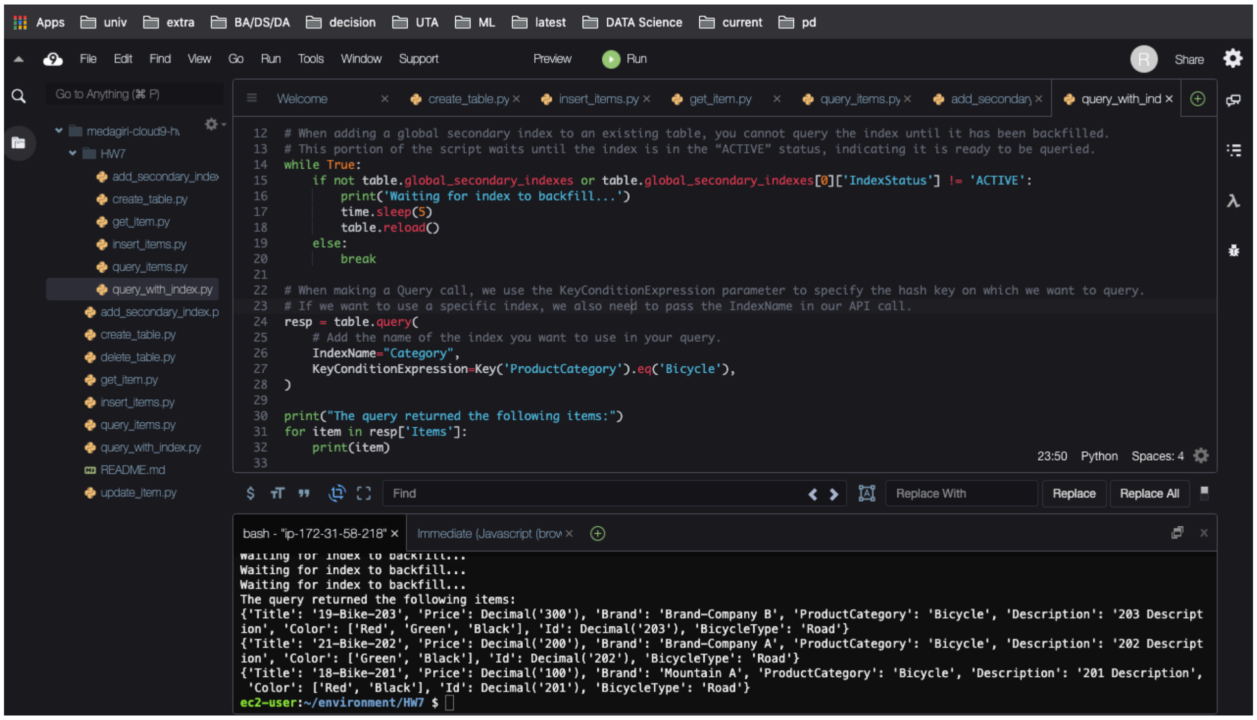
4. The screenshot of the query item ID number 103 python script and show the result (file editor and terminal). (5)



```python
# boto3 is the AWS SDK library for Python.
# The "resources" interface allows for a higher-level abstraction than the low-level client interface.
# For more details, go to http://boto3.readthedocs.io/en/latest/guide/resources.html
dynamodb = boto3.resource('dynamodb', region_name='us-east-1')
table = dynamodb.Table('MedagiriProductCatalog')

# When making a Query API call, we use the KeyConditionExpression parameter to specify the hash key on which we want to query.
# We're using the Key object from the Boto3 library to specify that we want the attribute name ("Author")
# to equal "John Grisham" by using the ".eq()" method.
resp = table.query(KeyConditionExpression=Key('Id').eq(103))

print("The query returned the following items:")
for item in resp['Items']:
    print(item)

```

```
ec2-user:~/environment/HW7 $ clear
ec2-user:~/environment/HW7 $ python get_item.py
{'Title': 'Book 103 Title', 'InPublication': False, 'PageCount': Decimal('600'), 'Dimensions': '8.5 x 11.0 x 1.5', 'ISBN': '333-33333333
33', 'Authors': 'Author1', 'Price': Decimal('2000'), 'ProductCategory': 'Book', 'Id': Decimal('103')}
ec2-user:~/environment/HW7 $ python query_items.py
The query returned the following items:
{'Title': 'Book 103 Title', 'InPublication': False, 'PageCount': Decimal('600'), 'Dimensions': '8.5 x 11.0 x 1.5', 'ISBN': '333-33333333
33', 'Authors': 'Author1', 'Price': Decimal('2000'), 'ProductCategory': 'Book', 'Id': Decimal('103')}
ec2-user:~/environment/HW7 $ 
```

5. The screenshot of the query all items in the Bicycle category python script and show the result (file editor and terminal. (5)



```python
# When adding a global secondary index to an existing table, you cannot query the index until it has been backfilled.
# This portion of the script waits until the index is in the "ACTIVE" status, indicating it is ready to be queried.
while True:
    if not table.global_secondary_indexes or table.global_secondary_indexes[0]['IndexStatus'] != 'ACTIVE':
        print('Waiting for index to backfill...')
        time.sleep(5)
        table.reload()
    else:
        break

# When making a Query call, we use the KeyConditionExpression parameter to specify the hash key on which we want to query.
# If we want to use a specific index, we also need to pass the IndexName in our API call.
resp = table.query(
    # Add the name of the index you want to use in your query.
    IndexName="Category",
    KeyConditionExpression=Key('ProductCategory').eq('Bicycle'),
)

print("The query returned the following items:")
for item in resp['Items']:
    print(item)

```

```
Waiting for index to backfill...
Waiting for index to backfill...
Waiting for index to backfill...
The query returned the following items:
{'Title': '19-Bike-203', 'Price': Decimal('300'), 'Brand': 'Brand-Company B', 'ProductCategory': 'Bicycle', 'Description': '203 Descript
ion', 'Color': ['Red', 'Green', 'Black'], 'Id': Decimal('203'), 'BicycleType': 'Road'}
{'Title': '21-Bike-202', 'Price': Decimal('200'), 'Brand': 'Brand-Company A', 'ProductCategory': 'Bicycle', 'Description': '202 Descript
ion', 'Color': ['Green', 'Black'], 'Id': Decimal('202'), 'BicycleType': 'Road'}
{'Title': '18-Bike-201', 'Price': Decimal('100'), 'Brand': 'Mountain A', 'ProductCategory': 'Bicycle', 'Description': '201 Description',
 'Color': ['Red', 'Black'], 'Id': Decimal('201'), 'BicycleType': 'Road'}
ec2-user:~/environment/HW7 $ 
```