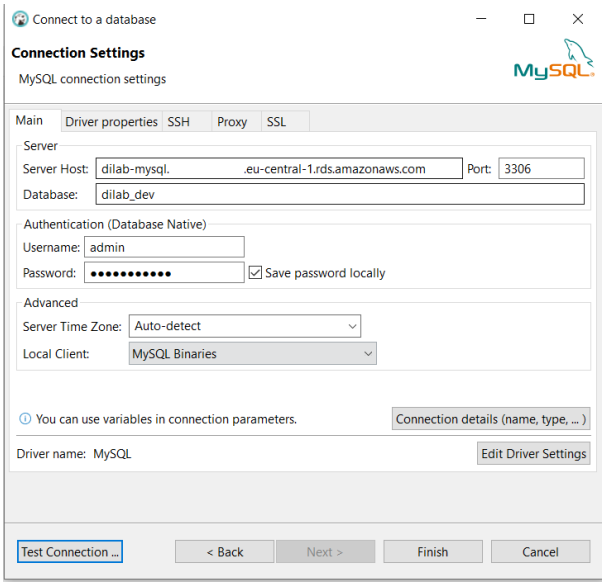
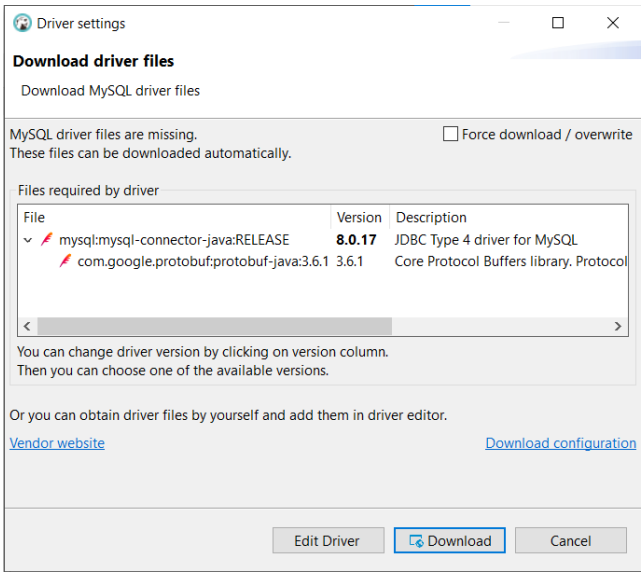
Task_04

1. RDS MYSQL

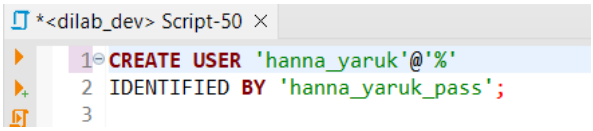
Connection to DB through DBvear

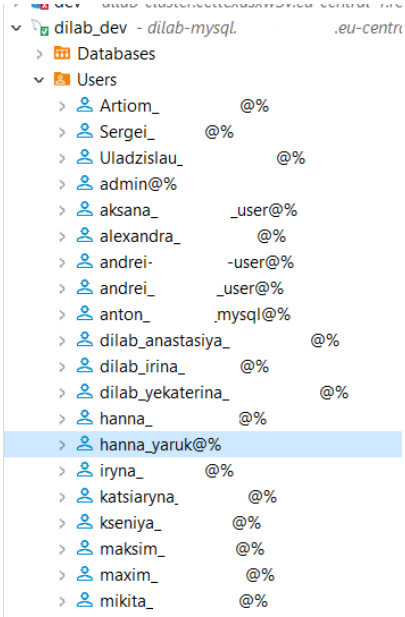
First I need to download driver files





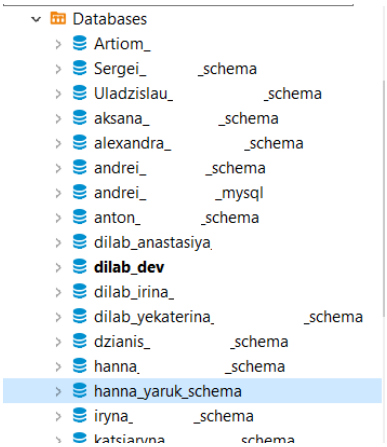
Create a new user





Create my personal schema and grant all privileges to my user

```sql
create schema hanna_yaruk_schema;

GRANT ALL PRIVILEGES ON hanna_yaruk_schema.* TO 'hanna_yaruk'@'%' WITH GRANT OPTION;
FLUSH PRIVILEGES;
```



To use Python to work with MySQL install connector Python MySQL

pip install mysql-connector-python

```
PS C:\Users\Hanna> pip install mysql-connector-python
Collecting mysql-connector-python
  Downloading mysql_connector_python-8.0.28-cp310-cp310-win_amd64.whl (7.2 MB)
     ---------------------------------------- 7.2/7.2 MB 5.2 MB/s eta 0:00:00
Collecting protobuf>=3.0.0
  Downloading protobuf-3.20.0-cp310-cp310-win_amd64.whl (903 kB)
     ---------------------------------------- 903.8/903.8 KB 7.1 MB/s eta 0:00:00
Installing collected packages: protobuf, mysql-connector-python
Successfully installed mysql-connector-python-8.0.28 protobuf-3.20.0
PS C:\Users\Hanna>
```

Connect to MySQL DataBase with my user

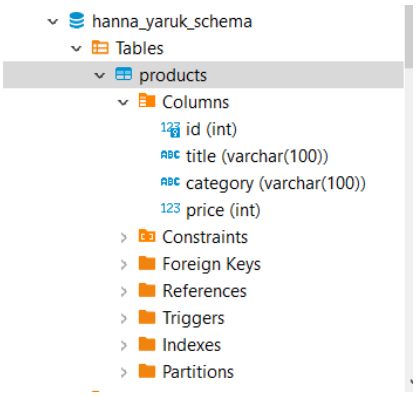
task_01.py qqq.py

```python
import mysql.connector
import sys
import boto3
import os

ENDPOINT="dilab-mysql.            .eu-central-1.rds.amazonaws.com"
PORT="3306"
USER="hanna_yaruk"
PASSWD="hanna_yaruk_pass"
REGION="eu-central-1"
DBNAME="dilab_dev"
os.environ['LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN'] = '1'

#gets the credentials from .aws/credentials
session = boto3.Session(profile_name='default')
client = session.client('rds')

#token = client.generate_db_auth_token(DBHostname=ENDPOINT, Port=PORT, DBUsername=USER, Region=REGION)

try:
    conn = mysql.connector.connect(host=ENDPOINT, user=USER, passwd=PASSWD, port=PORT, database=DBNAME, ssl_ca='SSLCERTIFICATE')
    cur = conn.cursor()
    cur.execute("""SELECT now()""")
    query_results = cur.fetchall()
    print(query_results)
except Exception as e:
    print("Database connection failed due to {}".format(e))
```

```
C:\Users\Hanna\AppData\Local\Programs\Python\Python310\python.exe C:/Users/Hanna/AppData/Roaming/JetBrains/PyCharmCE2021.3/scratches/task_01.py
[(datetime.datetime(2022, 4, 6, 15, 39, 17),)]

Process finished with exit code 0
```

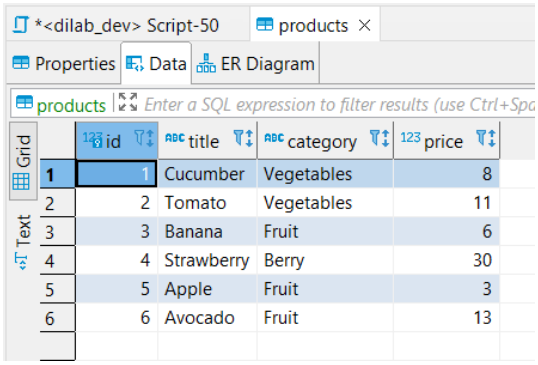
Create table products. Table is created

```python
#create tables for the database
create_products_query = """
CREATE TABLE IF NOT EXISTS products(
    id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(100),
    category VARCHAR(100),
    price INT
);
"""
with conn.cursor() as cursor:
    cursor.execute(create_products_query)
    conn.commit()
```



Truncate data from the table before inserting (for restarting). Insert data.

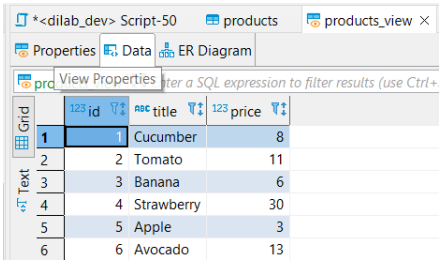
```python
#truncate table before inserting new data
truncate_products_query = """
Truncate table products;
"""
with conn.cursor() as cursor:
    cursor.execute(truncate_products_query)

#insert data into the table
insert_products_query = """
INSERT INTO products (id, title, category, price)
VALUES
    (1, "Cucumber", "Vegetables", 8),
    (2, "Tomato", "Vegetables", 11),
    (3, "Banana", "Fruit", 6),
    (4, "Strawberry", "Berry", 30),
    (5, "Apple", "Fruit", 3),
    (6, "Avocado", "Fruit", 13);
"""
with conn.cursor() as cursor:
    cursor.execute(insert_products_query)
    conn.commit()
```



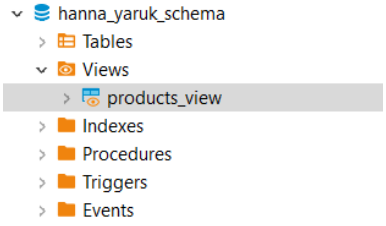
	id	title	category	price
1	1	Cucumber	Vegetables	8
2	2	Tomato	Vegetables	11
3	3	Banana	Fruit	6
4	4	Strawberry	Berry	30
5	5	Apple	Fruit	3
6	6	Avocado	Fruit	13

Create view

```python
#create view
create_view_products_query = """
CREATE OR REPLACE VIEW products_view as
SELECT id, title, price
FROM products
"""
with conn.cursor() as cursor:
    cursor.execute(create_view_products_query)
    conn.commit()
```

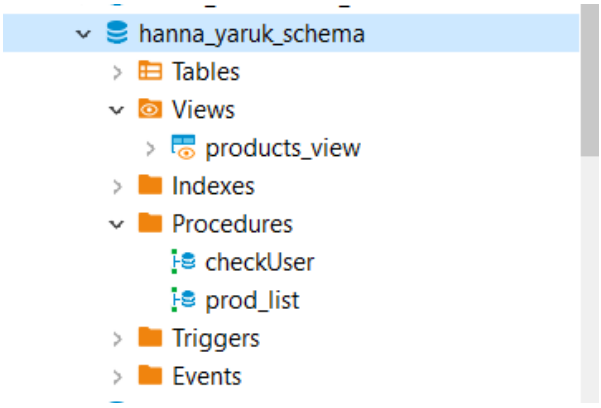


	id	title	price
1	1	Cucumber	8
2	2	Tomato	11
3	3	Banana	6
4	4	Strawberry	30
5	5	Apple	3
6	6	Avocado	13



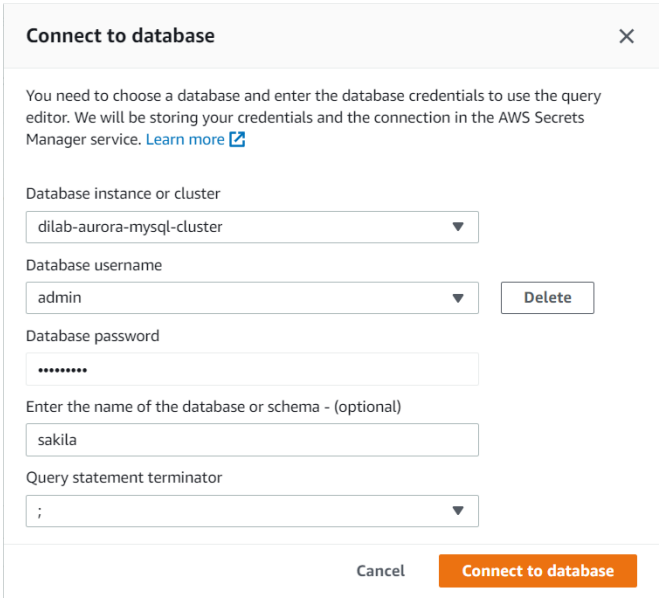
Create procedure for counting products of a category

```
#create procedure
create_procedure_query = """
CREATE PROCEDURE prod_list (category_insert varchar(64))
BEGIN
   SELECT COUNT(*) FROM products
   WHERE products.category = category_insert;
END;
"""
with conn.cursor() as cursor:
    cursor.execute(drop_procedure_query)
    cursor.execute(create_procedure_query)
    conn.commit()
```

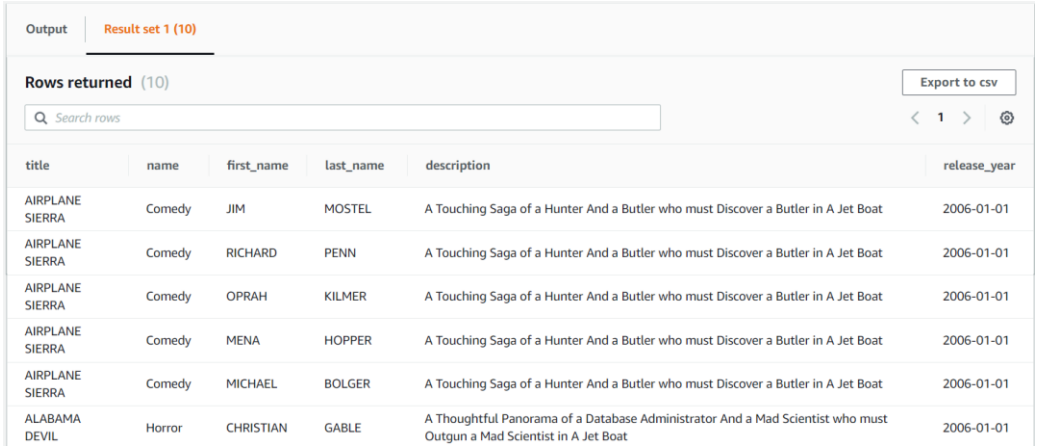


2. RDS AURORA

Connection to the schema Sakila from AWS Console

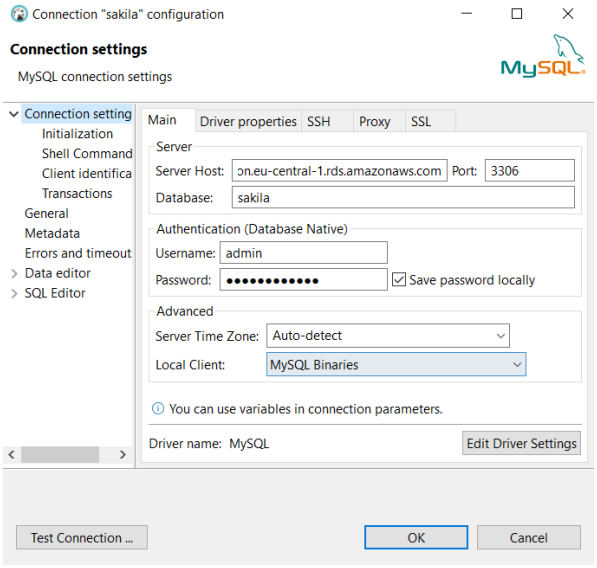


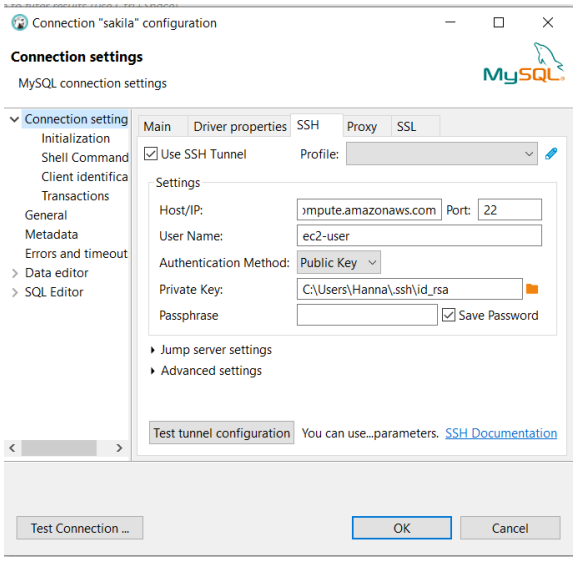
Write the query



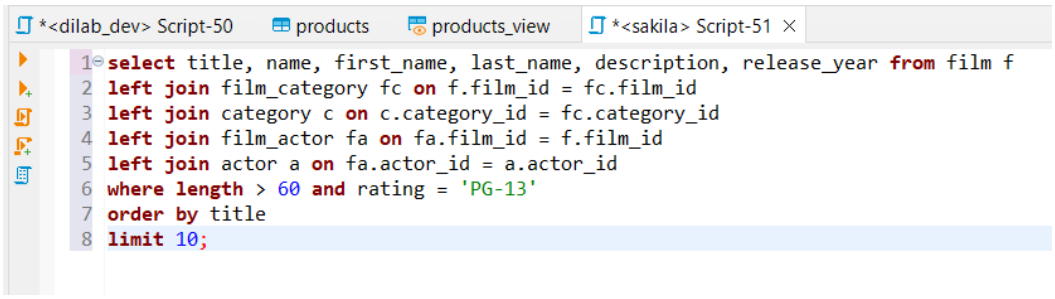
title	name	first_name	last_name	description	release_year
AIRPLANE SIERRA	Comedy	JIM	MOSTEL	A Touching Saga of a Hunter And a Butler who must Discover a Butler in A Jet Boat	2006-01-01
AIRPLANE SIERRA	Comedy	RICHARD	PENN	A Touching Saga of a Hunter And a Butler who must Discover a Butler in A Jet Boat	2006-01-01
AIRPLANE SIERRA	Comedy	OPRAH	KILMER	A Touching Saga of a Hunter And a Butler who must Discover a Butler in A Jet Boat	2006-01-01
AIRPLANE SIERRA	Comedy	MENA	HOPPER	A Touching Saga of a Hunter And a Butler who must Discover a Butler in A Jet Boat	2006-01-01
AIRPLANE SIERRA	Comedy	MICHAEL	BOLGER	A Touching Saga of a Hunter And a Butler who must Discover a Butler in A Jet Boat	2006-01-01
ALABAMA DEVIL	Horror	CHRISTIAN	GABLE	A Thoughtful Panorama of a Database Administrator And a Mad Scientist who must Outgun a Mad Scientist in A Jet Boat	2006-01-01

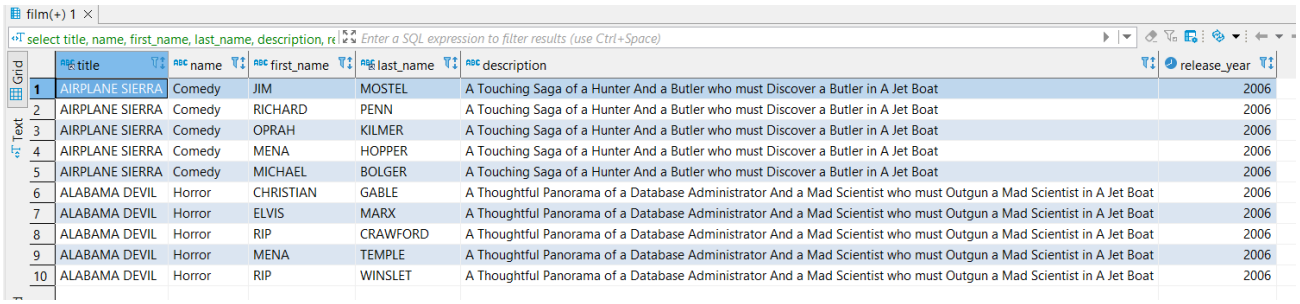
Connect to Aurora in DBeaver





Check the connection – use the same select statement

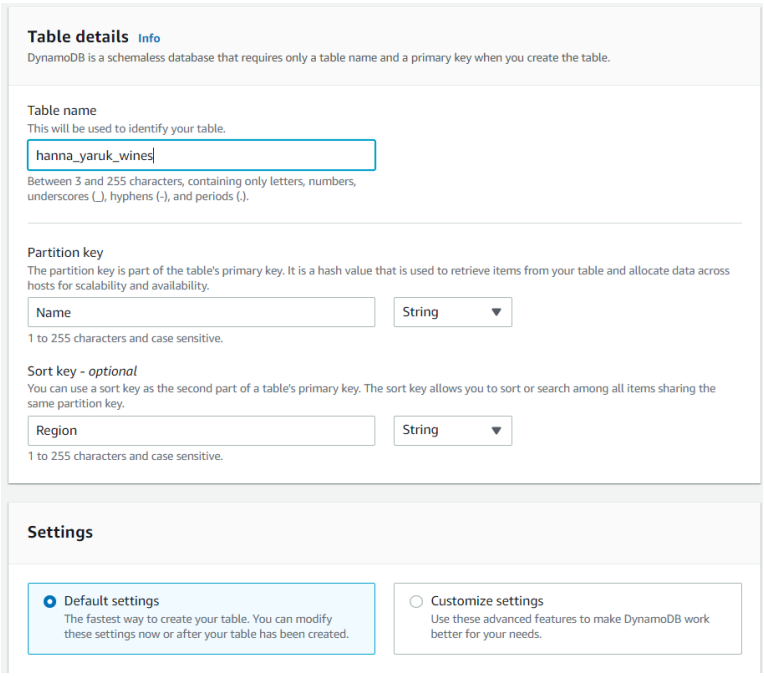




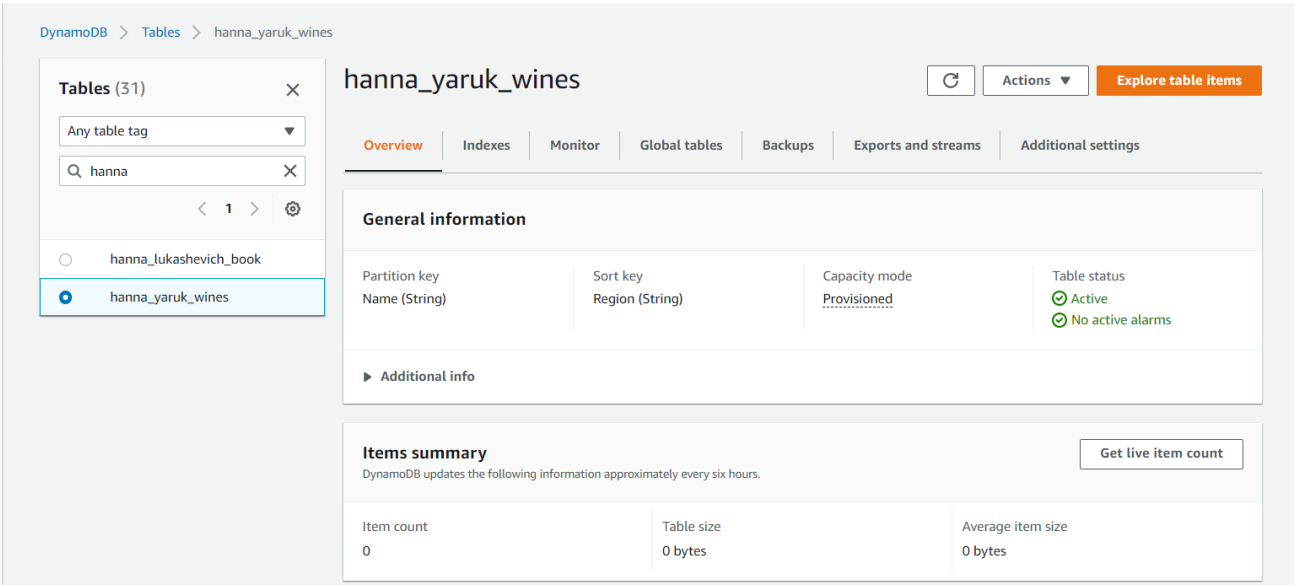
Aurora is serverless service without public IP or public DNS. We cannot work with it from PC. But we have EC2 instance that has public IP. This EC2 is running in same VPC with our Aurora DB cluster, so we should use this EC2 as a jump host for connection with Aurora (or another service in this VPC).

3. DYNAMODB

Create table with AWS Console



Partition Key is Name of wine – it is unique column. Sort key is Region – it helps to select information from the table.



Check the existed table with AWS Cli

aws dynamodb describe-table --table-name hanna_yaruk_wines

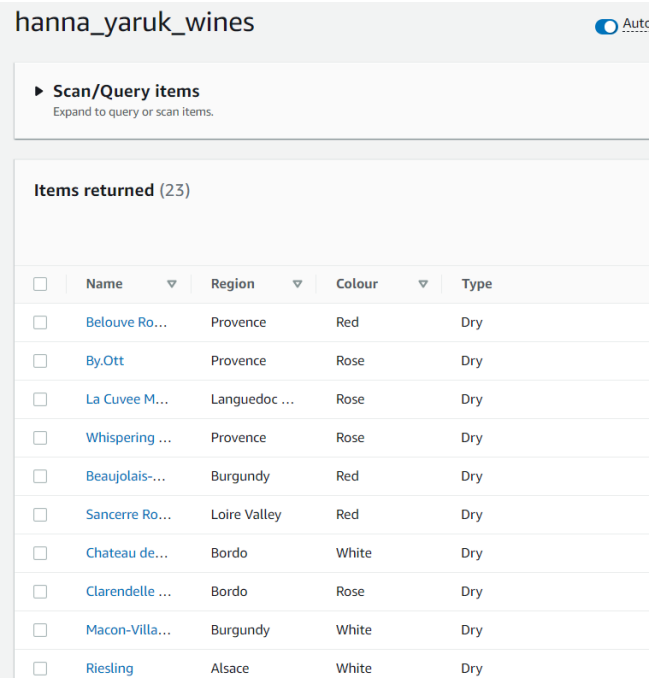
```
PS C:\Users\Hanna> aws dynamodb describe-table --table-name hanna_yaruk_wines
{
    "Table": {
        "AttributeDefinitions": [
            {
                "AttributeName": "Name",
                "AttributeType": "S"
            },
            {
                "AttributeName": "Region",
                "AttributeType": "S"
            }
        ],
        "TableName": "hanna_yaruk_wines",
        "KeySchema": [
            {
                "AttributeName": "Name",
                "KeyType": "HASH"
            },
            {
                "AttributeName": "Region",
                "KeyType": "RANGE"
            }
        ],
        "TableStatus": "ACTIVE",
        "CreationDateTime": "2022-04-06T22:32:23.411000+03:00",
        "ProvisionedThroughput": {
            "NumberOfDecreasesToday": 0,
            "ReadCapacityUnits": 5,
            "WriteCapacityUnits": 5
        },
        "TableSizeBytes": 0,
        "ItemCount": 0,
        "TableArn": "arn:aws:dynamodb:eu-central-1:260586643565:table/hanna_yaruk_wines",
        "TableId": "5a5d77d5-e989-4e4e-9b07-2929b1beb652"
    }
}

PS C:\Users\Hanna>
```

Insert the data from wines.json file

aws dynamodb batch-write-item --request-items file://wines.json

```
PS C:\Users\Hanna> aws dynamodb batch-write-item --request-items file://wines.json
{
    "UnprocessedItems": {}
}

PS C:\Users\Hanna>
```

Check the data in the table from AWS Console



Get the item (unique lines, cannot group)

aws dynamodb get-item --table-name hanna_yaruk_wines --key file://key.json --return-consumed-capacity TOTAL

```
PS C:\Users\Hanna> aws dynamodb get-item --table-name hanna_yaruk_wines --key file://key.json --return-consumed-capacity TOTAL
{
    "Item": {
        "Colour": {
            "S": "Rose"
        },
        "Type": {
            "S": "Dry"
        },
        "Region": {
            "S": "Languedoc and Roussillon"
        },
        "Name": {
            "S": "La Cuvee Mythique Rose"
        }
    },
    "ConsumedCapacity": {
        "TableName": "hanna_yaruk_wines",
        "CapacityUnits": 0.5
    }
}


PS C:\Users\Hanna>
```

Get the select statement – gives the error. Not fixed

aws dynamodb query --table-name hanna_yaruk_wines --key-condition-expression "Name = :n and Region = :r" --expression-attribute-values file://query.json

```
PS C:\Users\Hanna> aws dynamodb query --table-name hanna_yaruk_wines --key-condition-expression "Name = :n and Region = :r" --e
xpression-attribute-values file://query.json

An error occurred (ValidationException) when calling the Query operation: Invalid KeyConditionExpression: Attribute name is a r
eserved keyword; reserved keyword: Name
PS C:\Users\Hanna>
```

Delete items

```
PS C:\Users\Hanna> aws dynamodb delete-item --table-name hanna_yaruk_wines --key file://delete_key1.json
PS C:\Users\Hanna> aws dynamodb delete-item --table-name hanna_yaruk_wines --key file://delete_key2.json
PS C:\Users\Hanna>
```

Items returned (23)

Name	Region	Colour	Type
Riesling	Alsace	White	Dry
Pinot Gris Les Elements	Alsace	White	Semi-dry
Gewurztraminer Jules Geyl	Alsace	White	Sweet
Chateau des Graves Blanc	Bordo	White	Dry
Clarendelle a par Haut-Brion Rose	Bordo	Rose	Dry
Jean-Pierre Moueix Saint-Emilion	Bordo	Red	Dry
Grand Bateau Rouge	Bordo	Red	Dry
Beaujolais-Villages	Burgundy	Red	Dry
Macon-Villages	Burgundy	White	Dry
Bourgogne Pinot Noir La Vignee	Burgundy	Red	Dry
Sauvignon Saint-Bris	Burgundy	White	Dry
La Cuvee Mythique Rose	Languedoc ...	Rose	Dry
Le Grand Noir Grenache-Syrah-Mo...	Languedoc ...	Red	Semi-dry

hanna_yaruk_wines

Autopreview

▸ **Scan/Query items**
Expand to query or scan items.

Items returned (21)

Name	Region	Colour	Type
Riesling	Alsace	White	Dry
Pinot Gris Les Elements	Alsace	White	Semi-dry
Gewurztraminer Jules Geyl	Alsace	White	Sweet
Clarendelle a par Haut-Brion ...	Bordo	Rose	Dry
Grand Bateau Rouge	Bordo	Red	Dry
Beaujolais-Villages	Burgundy	Red	Dry
Macon-Villages	Burgundy	White	Dry
Bourgogne Pinot Noir La Vig...	Burgundy	Red	Dry
Sauvignon Saint-Bris	Burgundy	White	Dry
La Cuvee Mythique Rose	Languedoc ...	Rose	Dry