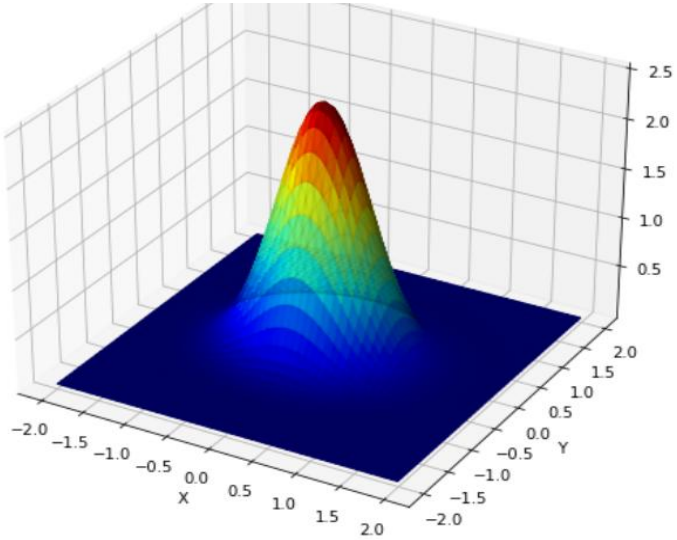
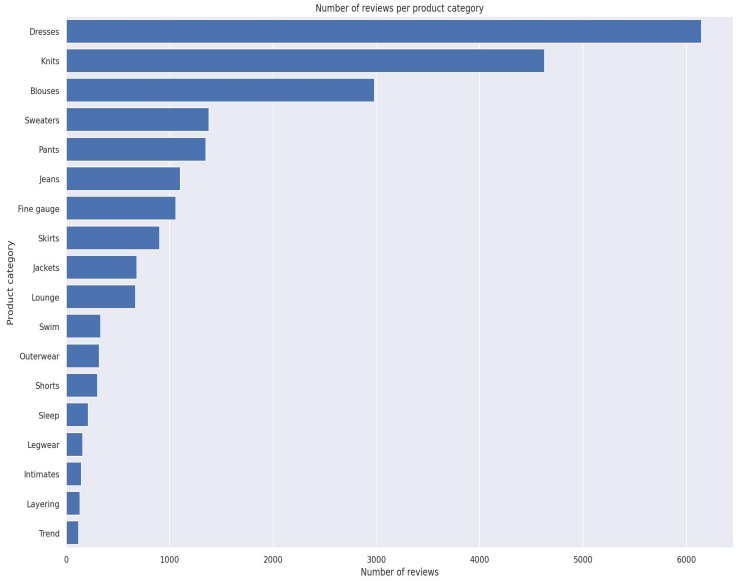
Practical Data Science with AWS Cloud

Divyanshu Vyas
Energy Data Scientist, Petroleum From Scratch

Day 2





Code and explanation

- !aws s3 ls s3://dlai-practical-data-science/data/raw/
- aws s3 cp [bucket_name/file_name] [file_name]

- lists all the files in bucket given in yellow shade.
- function copies the file from the S3 bucket into the local environment or into another S3 bucket. Let's use it to copy the file with the dataset locally.

Example : !aws s3 cp s3://dlai-practical-data-science/data/raw/womens_clothing_ecommerce_reviews.csv ./womens_clothing_ecommerce_reviews.csv

Brings the dataset to local

Code and explanation

- !head -n 5 ./womens_clothing_ecommerce_reviews_transformed.csv

```
import boto3
import sagemaker
import pandas as pd
import numpy as np
import botocore

config = botocore.config.Config(user_agent_extra='dlai-pds/c1/w1')

# low-level service client of the boto3 session
sm = boto3.client(service_name='sagemaker',
                  config=config)

sess = sagemaker.Session(sagemaker_client=sm)

bucket = sess.default_bucket()
role = sagemaker.get_execution_role()
region = sess.boto_region_name
account_id = sess.account_id

print('S3 Bucket: {}'.format(bucket))
print('Region: {}'.format(region))
print('Account ID: {}'.format(account_id))
```

- Just a faster way of seeing df.head() using AWS CLI. !head -1... prints the column headers

- Registering the Public DF

Code and explanation

- !aws s3 cp ./womens_clothing_ecommerce_reviews_transformed.csv s3://$bucket/data/transformed/womens_clothing_ecommerce_reviews_transformed.csv

- upload the local file to S3 Bucket

AWS Data Wrangler is an AWS Professional Service open source python initiative that extends the power of Pandas library to AWS connecting dataframes and AWS data related services (Amazon Redshift, AWS Glue, Amazon Athena, Amazon EMR, Amazon QuickSight, etc).

Built on top of other open-source projects like Pandas, Apache Arrow, Boto3, SQLAlchemy, Psycopg2 and PyMySQL, it offers abstracted functions to execute usual ETL tasks like load/unload data from data lakes, data warehouses and databases.

Review the AWS Data Wrangler documentation: https://aws-data-wrangler.readthedocs.io/en/stable/

```
>>>import awswrangler as wr
```

About AWS Glue Catalog

- Create AWS Glue Catalog database

- The data catalog features of AWS Glue and the inbuilt integration to Amazon S3 simplify the process of identifying data and deriving the schema definition out of the discovered data. Using AWS Glue crawlers within your data catalog, you can traverse your data stored in Amazon S3 and build out the metadata tables that are defined in your data catalog.

- Here you will use wr.catalog.create_database function to create a database with the name dsoaws_deep_learning ("dsoaws" stands for "Data Science on AWS").

- Navigation : Search AWS Glue in search bar --> Data Catalog --> Databases ---> You can see the list of DBs that you've created (snippet).
- If you click on the DB names youll see that there's no tables yet.

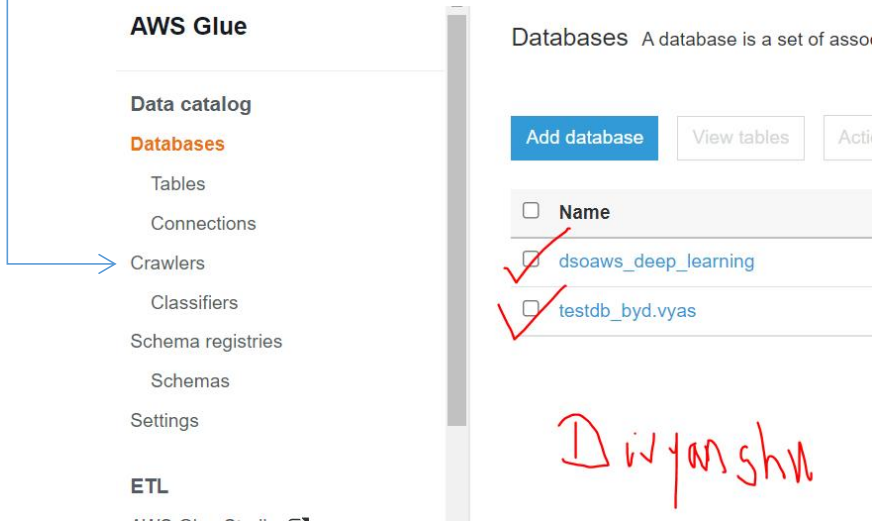
```
[19]: wr.catalog.create_database(
          name='dsoaws_deep_learning',
          exist_ok=True
      )

[21]: wr.catalog.create_database(
          name='testDB_byD.Vyas',
          exist_ok=True
      )

[22]: dbs = wr.catalog.get_databases()

      for db in dbs:
          print("Database name: " + db['Name'])

      Database name: dsoaws_deep_learning
      Database name: testdb_byd.vyas
```



Registering CSV Table

- Register CSV data with AWS Glue Catalog.
- Registering is nothing but picking a table from a S3-Bucket (path) and storing it in a given DB created by AWS glue.

Review the table shape:

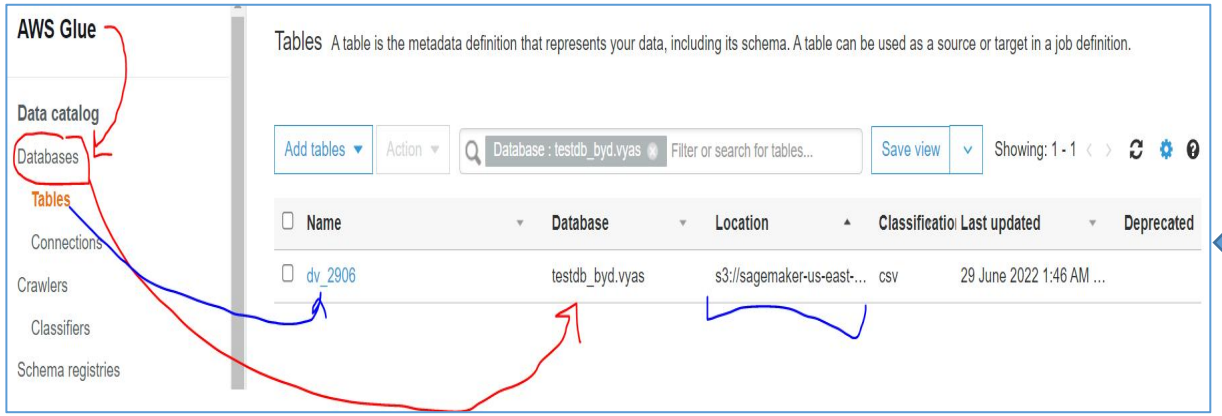
```
[32]: table = wr.catalog.table(database='dsoaws_deep_learning',
                               table='reviews')
      table
```

[32]:

	Column Name	Type	Partition	Comment
0	sentiment	int	False	
1	review_body	string	False	
2	product_category	string	False	

Instructions: Use `wr.catalog.create_csv_table` function with the following parameters

```
res = wr.catalog.create_csv_table(
    database='...', # AWS Glue Catalog database name
    path='s3://{}/data/transformed/'.format(bucket), # S3 object path for the data
    table='reviews', # registered table name
    columns_types={
        'sentiment': 'int',
        'review_body': 'string',
        'product_category': 'string'
    },
    mode='overwrite',
    skip_header_line_count=1,
    sep=','
)
```




```
[27]: def store_table_in_DB(table_name, db_name, bucket_name, schema):
          wr.catalog.create_csv_table(
          ### BEGIN SOLUTION - DO NOT delete this comment for grading purposes
          database=db_name, # Replace None
          ### END SOLUTION - DO NOT delete this comment for grading purposes
          path='s3://{}/data/transformed/'.format(bucket_name),
          table=table_name,
          columns_types=schema,
          mode='overwrite',
          skip_header_line_count=1,
          sep=','
                                  )

          print('Table picked from S3-Bucket path and stored/registered in DB')
```

```
[28]: schema = {
            'sentiment': 'int',
            'review_body': 'string',
            'product_category': 'string'
        }

      store_table_in_DB(table_name='DV_2906', db_name='testdb_byd.vyas', bucket_name=bucket, schema=schema)

      Table picked from S3-Bucket path and stored/registered in DB
```

SQL Queries using Amzon AWS Athena

- Run SQL queries using Amazon Athena

- Amazon Athena lets you query data in Amazon S3 using a standard SQL interface. It reflects the databases and tables in the AWS Glue Catalog. You can create interactive queries and perform any data manipulations required for further downstream processing.

- Standard SQL query can be saved as a string and then passed as a parameter into the Athena query. Run the following cells as an example to count the total number of reviews by sentiment. The SQL query here will take the following form:

- Athena Creates a TEMPORARY S3 Bucket in which the queried tables are stored and can be accessed.

SELECT column_name, COUNT(column_name) as new_column_name

FROM table_name

GROUP BY column_name

ORDER BY column_name

2.2. Create default S3 bucket for Amazon Athena

Amazon Athena requires this S3 bucket to store temporary query results and improve performance of subsequent queries.

The contents of this bucket are mostly binary and human-unreadable.

[33]:
```
# S3 bucket name
wr.athena.create_athena_bucket()

# EXPECTED OUTPUT
# 's3://aws-athena-query-results-ACCOUNT-REGION/'
```

[33]: 's3://aws-athena-query-results-426108123359-us-east-1/'

```
print(statement_count_by_sentiment)
```

```
SELECT sentiment, COUNT(sentiment) AS count_sentiment
FROM reviews
GROUP BY sentiment
ORDER BY sentiment
```

Query data in Amazon Athena database cluster using the prepared SQL statement:

[42]:
```
df_count_by_sentiment = wr.athena.read_sql_query(
    sql=statement_count_by_sentiment,
    database=database_name
)

print(df_count_by_sentiment)
```

```
   sentiment  count_sentiment
0         -1             2370
1          0             2823
2          1            17433
```

Preview the results of the query:

SQL Queries using Amzon AWS Athena contd.

- On the right you can see the temporary S3 Bucket created when you're querying using AWS Athena

- *Note you can also save image data from local to an S3 bucket.*