Register and visualize dataset

Introduction

In this lab you will ingest and transform the customer product reviews dataset. Then you will use AWS data stack services such as AWS Glue and Amazon Athena for ingesting and querying the dataset. Finally you will use AWS Data Wrangler to analyze the dataset and plot some visuals extracting insights.

Table of Contents

- 1. Ingest and transform the public dataset
  - 1.1. List the dataset files in the public S3 bucket
    - Exercise 1
  - 1.2. Copy the data locally to the notebook
  - 1.3. Transform the data
  - 1.4 Write the data to a CSV file
- 2. Register the public dataset for querying and visualizing
  - 2.1. Register S3 dataset files as a table for querying
    - Exercise 2
  - 2.2. Create default S3 bucket for Amazon Athena
- 3. Visualize data
  - 3.1. Preparation for data visualization
  - 3.2. How many reviews per sentiment?
    - Exercise 3
  - 3.3. Which product categories are highest rated by average sentiment?
  - 3.4. Which product categories have the most reviews?
    - Exercise 4
  - 3.5. What is the breakdown of sentiments per product category?
  - 3.6. Analyze the distribution of review word counts

Let's install the required modules first.

[57]:

```
!pip install --disable-pip-version-check -q sagemaker
!pip install --disable-pip-version-check -q pandas
!pip install --disable-pip-version-check -q awswrangler
!pip install --disable-pip-version-check -q numpy
!pip install --disable-pip-version-check -q matplotlib
!pip install -q protobuf
```

```
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviou
r with the system package manager. It is recommended to use a virtual environment instead: http
s://pip.pypa.io/warnings/venv
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviou
r with the system package manager. It is recommended to use a virtual environment instead: http
s://pip.pypa.io/warnings/venv
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviou
r with the system package manager. It is recommended to use a virtual environment instead: http
s://pip.pypa.io/warnings/venv
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviou
r with the system package manager. It is recommended to use a virtual environment instead: http
s://pip.pypa.io/warnings/venv
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviou
r with the system package manager. It is recommended to use a virtual environment instead: http
s://pip.pypa.io/warnings/venv
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviou
r with the system package manager. It is recommended to use a virtual environment instead: http
s://pip.pypa.io/warnings/venv

[notice] A new release of pip is available: 23.3.1 -> 23.3.2
[notice] To update, run: pip install --upgrade pip
```

1. Ingest and transform the public dataset

The dataset Women's Clothing Reviews has been chosen as the main dataset.

It is shared in a public Amazon S3 bucket, and is available as a comma-separated value (CSV) text format:

`s3://dlai-practical-data-science/data/raw/womens_clothing_ecommerce_reviews.csv`

1.1. List the dataset files in the public S3 bucket

The AWS Command Line Interface (CLI) is a unified tool to manage your AWS services. With just one tool, you can control multiple AWS services from the command line and automate them through scripts. You will use it to list the dataset files.

View dataset files in CSV format

`aws s3 ls [bucket_name]` function lists all objects in the S3 bucket. Let's use it to view the reviews data files in CSV format:

Exercise 1

View the list of the files available in the public bucket `s3://dlai-practical-data-science/data/raw/`.

Instructions: Use `aws s3 ls [bucket_name]` function. To run the AWS CLI command from the notebook you will need to put an exclamation mark in front of it: `!aws`. You should see the data file `womens_clothing_ecommerce_reviews.csv` in the list.

[58]:

```
### BEGIN SOLUTION - DO NOT delete this comment for grading purposes
!aws s3 ls dlai-practical-data-science/data/raw/ # Replace None
### END SOLUTION - DO NOT delete this comment for grading purposes

# EXPECTED OUTPUT
# ... womens_clothing_ecommerce_reviews.csv
```

```
2021-04-30 02:21:06    8457214 womens_clothing_ecommerce_reviews.csv
```

1.2. Copy the data locally to the notebook

`aws s3 cp [bucket_name/file_name] [file_name]` 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.

[59]:

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

```
download: s3://dlai-practical-data-science/data/raw/womens_clothing_ecommerce_reviews.csv to ./wo
mens_clothing_ecommerce_reviews.csv
```

Now use the Pandas dataframe to load and preview the data.

[60]:

```
import pandas as pd
import csv

df = pd.read_csv('./womens_clothing_ecommerce_reviews.csv',
                 index_col=0)

df.shape
```

[60]:

```
(23486, 10)
```

[61]:

```
df
```

[61]:

	Clothing ID	Age	Title	Review Text	Rating	Recommended IND	Positive Feedback Count	Division Name	Department Name	Class Name
0	847	33	Cute, crisp shirt	If this product was in petite i would get the...	4	1	2	General	Tops	Blouses
1	1080	34	NaN	Love this dress! it's sooo pretty. i happene...	5	1	4	General	Dresses	Dresses
2	1077	60	Some major design flaws	I had such high hopes for this dress and reall...	3	0	0	General	Dresses	Dresses
3	1049	50	My favorite buy!	I love love love this jumpsuit. it's fun fl...	5	1	0	General Petite	Bottoms	Pants
4	847	47	Flattering shirt	This shirt is very flattering to all due to th...	5	1	6	General	Tops	Blouses
...	...	...	...	...	...	...	...	...	...	...
23481	1104	34	Great dress for many occasions	I was very happy to snag this dress at such a ...	5	1	0	General Petite	Dresses	Dresses
23482	862	48	Wish it was made of cotton	It reminds me of maternity clothes. soft stre...	3	1	0	General Petite	Tops	Knits
23483	1104	31	Cute, but see through	This fit well but the top was very see throug...	3	0	1	General Petite	Dresses	Dresses
23484	1084	28	Very cute dress, perfect for summer parties an...	I bought this dress for a wedding i have this ...	3	1	2	General	Dresses	Dresses
23485	1104	52	Please make more like this one!	This dress in a lovely platinum is feminine an...	5	1	22	General Petite	Dresses	Dresses

23486 rows × 10 columns

1.3. Transform the data

To simplify the task, you will transform the data into a comma-separated value (CSV) file that contains only a `review_body`, `product_category`, and `sentiment` derived from the original data.

[62]:

```
df_transformed = df.rename(columns={'Review Text': 'review_body',
                                    'Rating': 'star_rating',
                                    'Class Name': 'product_category'})
df_transformed.drop(columns=['Clothing ID', 'Age', 'Title', 'Recommended IND', 'Positive Feedback
                    inplace=True)

df_transformed.dropna(inplace=True)

df_transformed.shape
```

[62]:

```
(22628, 3)
```

Now convert the `star_rating` into the `sentiment` (positive, neutral, negative), which later on will be for the prediction.

[63]:

```
def to_sentiment(star_rating):
    if star_rating in {1, 2}: # negative
        return -1
    if star_rating == 3:      # neutral
        return 0
    if star_rating in {4, 5}: # positive
        return 1

# transform star_rating into the sentiment
df_transformed['sentiment'] = df_transformed['star_rating'].apply(lambda star_rating:
    to_sentiment(star_rating=star_rating)
)

# drop the star rating column
df_transformed.drop(columns=['star_rating'],
                    inplace=True)

# remove reviews for product_categories with < 10 reviews
df_transformed = df_transformed.groupby('product_category').filter(lambda reviews : len(reviews) >

df_transformed.shape
```

[63]:

```
(22626, 3)
```

[64]:

```
# preview the results
df_transformed
```

[64]:

	sentiment	review_body	product_category
0	1	If this product was in petite i would get the...	Blouses
1	1	Love this dress! it's sooo pretty. i happene...	Dresses
2	0	I had such high hopes for this dress and reall...	Dresses
3	1	I love love love this jumpsuit. it's fun fl...	Pants
4	1	This shirt is very flattering to all due to th...	Blouses
...	...	...	...
23481	1	I was very happy to snag this dress at such a ...	Dresses
23482	0	It reminds me of maternity clothes. soft stre...	Knits
23483	0	This fit well but the top was very see throug...	Dresses
23484	0	I bought this dress for a wedding i have this ...	Dresses
23485	1	This dress in a lovely platinum is feminine an...	Dresses

22626 rows × 3 columns

1.4 Write the data to a CSV file

[65]:

```
df_transformed.to_csv('./womens_clothing_ecommerce_reviews_transformed.csv',
                      index=False)
```

[66]:

```
!head -n 5 ./womens_clothing_ecommerce_reviews_transformed.csv
```

```
sentiment,review_body,product_category
1,If this product was in petite  i would get the petite. the regular is a little long on me but a
tailor can do a simple fix on that.     fits nicely! i'm 5'4  130lb and pregnant so i bough t med
ium to grow into.     the tie can be front or back so provides for some nice flexibility on form
fitting.,Blouses
1,"Love this dress!  it's sooo pretty.  i happened to find it in a store  and i'm glad i did bc i
never would have ordered it online bc it's petite.  i bought a petite and am 5'8"".  i love the l
ength on me- hits just a little below the knee.  would definitely be a true midi on someone who i
s truly petite.",Dresses
0,I had such high hopes for this dress and really wanted it to work for me. i initially ordered t
he petite small (my usual size) but i found this to be outrageously small. so small in fact that
i could not zip it up! i reordered it in petite medium  which was just ok. overall  the top half
was comfortable and fit nicely  but the bottom half had a very tight under layer and several some
what cheap (net) over layers. imo  a major design flaw was the net over layer sewn directly into
the zipper - it c,Dresses
1,I love  love  love this jumpsuit. it's fun  flirty  and fabulous! every time i wear it  i get n
othing but great compliments!,Pants
```

2. Register the public dataset for querying and visualizing

You will register the public dataset into an S3-backed database table so you can query and visualize our dataset at scale.

2.1. Register S3 dataset files as a table for querying

Let's import required modules.

`boto3` is the AWS SDK for Python to create, configure, and manage AWS services, such as Amazon Elastic Compute Cloud (Amazon EC2) and Amazon Simple Storage Service (Amazon S3). The SDK provides an object-oriented API as well as low-level access to AWS services.

`sagemaker` is the SageMaker Python SDK which provides several high-level abstractions for working with the Amazon SageMaker.

[67]:

```
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))
```

```
S3 Bucket: sagemaker-us-east-1-165544655018
Region: us-east-1
Account ID: <bound method Session.account_id of <sagemaker.session.Session object at 0x7fdde3d617
20>>
```

```
Review the empty bucket which was created automatically for this account.

**Instructions**:
- open the link
- click on the S3 bucket name `sagemaker-us-east-1-ACCOUNT`
- check that it is empty at this stage
```

[68]:

```
from IPython.core.display import display, HTML

display(HTML('<b>Review <a target="top" href="https://s3.console.aws.amazon.com/s3/home?region={}#
```

```
/tmp/ipykernel_2572/2094588680.py:1: DeprecationWarning: Importing display from IPython.core.disp
lay is deprecated since IPython 7.14, please import from IPython display
  from IPython.core.display import display, HTML
```

Review Amazon S3 buckets

Copy the file into the S3 bucket.

[69]:

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

```
upload: ./womens_clothing_ecommerce_reviews_transformed.csv to s3://sagemaker-us-east-1-165544655
018/data/transformed/womens_clothing_ecommerce_reviews_transformed.csv
```

```
Review the bucket with the file we uploaded above.

**Instructions**:
- open the link
- check that the CSV file is located in the S3 bucket
- check the location directory structure is the same as in the CLI command above
- click on the file name and see the available information about the file (region, size, S3 URI,
Amazon Resource Name (ARN))
```

[70]:

```
from IPython.core.display import display, HTML

display(HTML('<b>Review <a target="top" href="https://s3.console.aws.amazon.com/s3/buckets/{}?regi
```

```
/tmp/ipykernel_2572/1628230508.py:1: DeprecationWarning: Importing display from IPython.core.disp
lay is deprecated since IPython 7.14, please import from IPython display
  from IPython.core.display import display, HTML
```

Review Amazon S3 buckets

Import AWS Data Wrangler

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/

[71]:

```
import awswrangler as wr
```

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").

[72]:

```
wr.catalog.create_database(
    name='dsoaws_deep_learning',
    exist_ok=True
)
```

[73]:

```
dbs = wr.catalog.get_databases()

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

```
Database name: dsoaws_deep_learning
```

```
Review the created database in the AWS Glue Catalog.

**Instructions**:
- open the link
- on the left side panel notice that you are in the AWS Glue -> Data Catalog -> Databases
- check that the database `dsoaws_deep_learning` has been created
- click on the name of the database
- click on the `Tables in dsoaws_deep_learning` link to see that there are no tables
```

[74]:

```
from IPython.core.display import display, HTML

display(HTML('<b>Review <a target="top" href="https://console.aws.amazon.com/glue/home?region={}#c
```

```
/tmp/ipykernel_2572/2996986658.py:1: DeprecationWarning: Importing display from IPython.core.disp
lay is deprecated since IPython 7.14, please import from IPython display
  from IPython.core.display import display, HTML
```

Review AWS Glue Databases

Register CSV data with AWS Glue Catalog

Exercise 2

Register CSV data with AWS Glue Catalog.

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=','
)
```

[75]:

```
wr.catalog.create_csv_table(
    ### BEGIN SOLUTION - DO NOT delete this comment for grading purposes
    database='dsoaws_deep_learning', # Replace None
    ### END SOLUTION - DO NOT delete this comment for grading purposes
    path='s3://{}/data/transformed/'.format(bucket),
    table="reviews",
    columns_types={
        'sentiment': 'int',
        'review_body': 'string',
        'product_category': 'string'
    },
    mode='overwrite',
    skip_header_line_count=1,
    sep=','
)
```

Review the registered table in the AWS Glue Catalog.

Instructions:

- open the link
- on the left side panel notice that you are in the AWS Glue -> Data Catalog -> Databases -> Tables
- check that you can see the table `reviews` from the database `dsoaws_deep_learning` in the list
- click on the name of the table
- explore the available information about the table (name, database, classification, location, schema etc.)

[76]:

```
from IPython.core.display import display, HTML

display(HTML('<b>Review <a target="top" href="https://console.aws.amazon.com/glue/home?region={}#"
```

```
/tmp/ipykernel_2572/19572356.py:1: DeprecationWarning: Importing display from IPython.core.displa
y is deprecated since IPython 7.14, please import from IPython display
  from IPython.core.display import display, HTML
```

Review AWS Glue Catalog

Review the table shape:

[77]:

```
table = wr.catalog.table(database='dsoaws_deep_learning',
                         table='reviews')
table
```

[77]:

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

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.

[78]:

```
# S3 bucket name
wr.athena.create_athena_bucket()

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

[78]:

```
's3://aws-athena-query-results-165544655018-us-east-1/'
```

3. Visualize data

Reviews dataset - column descriptions

- `sentiment`: The review's sentiment (-1, 0, 1).
- `product_category`: Broad product category that can be used to group reviews (in this case digital videos).
- `review_body`: The text of the review.

3.1. Preparation for data visualization

Imports

[79]:

```
import numpy as np
import seaborn as sns

import matplotlib.pyplot as plt
%matplotlib inline
%config InlineBackend.figure_format='retina'
```

Settings

Set AWS Glue database and table name.

[80]:

```
# Do not change the database and table names - they are used for grading purposes!
database_name = 'dsoaws_deep_learning'
table_name = 'reviews'
```

```
Set seaborn parameters. You can review seaborn documentation following the [link]
(https://seaborn.pydata.org/index.html).
```

[81]:

```
sns.set_style = 'seaborn-whitegrid'

sns.set(rc={"font.style":"normal",
            "axes.facecolor":"white",
            'grid.color': '.8',
            'grid.linestyle': '-',
            "figure.facecolor":"white",
            "figure.titlesize":20,
            "text.color":"black",
            "xtick.color":"black",
            "ytick.color":"black",
            "axes.labelcolor":"black",
            "axes.grid":True,
            'axes.labelsize':10,
            'xtick.labelsize':10,
            'font.size':10,
            'ytick.labelsize':10})
```

Helper code to display values on barplots:

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:

```sql
SELECT column_name, COUNT(column_name) as new_column_name
FROM table_name
GROUP BY column_name
ORDER BY column_name
```

If you are not familiar with the SQL query statements, you can review some tutorials following
the [link](https://www.w3schools.com/sql/default.asp).
~~~

3.2. How many reviews per sentiment?

Set the SQL statement to find the count of sentiments:

[82]:

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

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:

[83]:

```
df_count_by_sentiment = wr.athena.read_sql_query(
    sql=statement_count_by_sentiment,
    database=database_name
)

print(df_count_by_sentiment)
```

```
/opt/conda/lib/python3.10/site-packages/awswrangler/_distributed.py:105: FutureWarning: promote h
as been superseded by mode='default'.
  return cls.dispatch_func(func)(*args, **kw)
   sentiment  count_sentiment
0         -1             2370
1          0             2823
2          1            17433
```

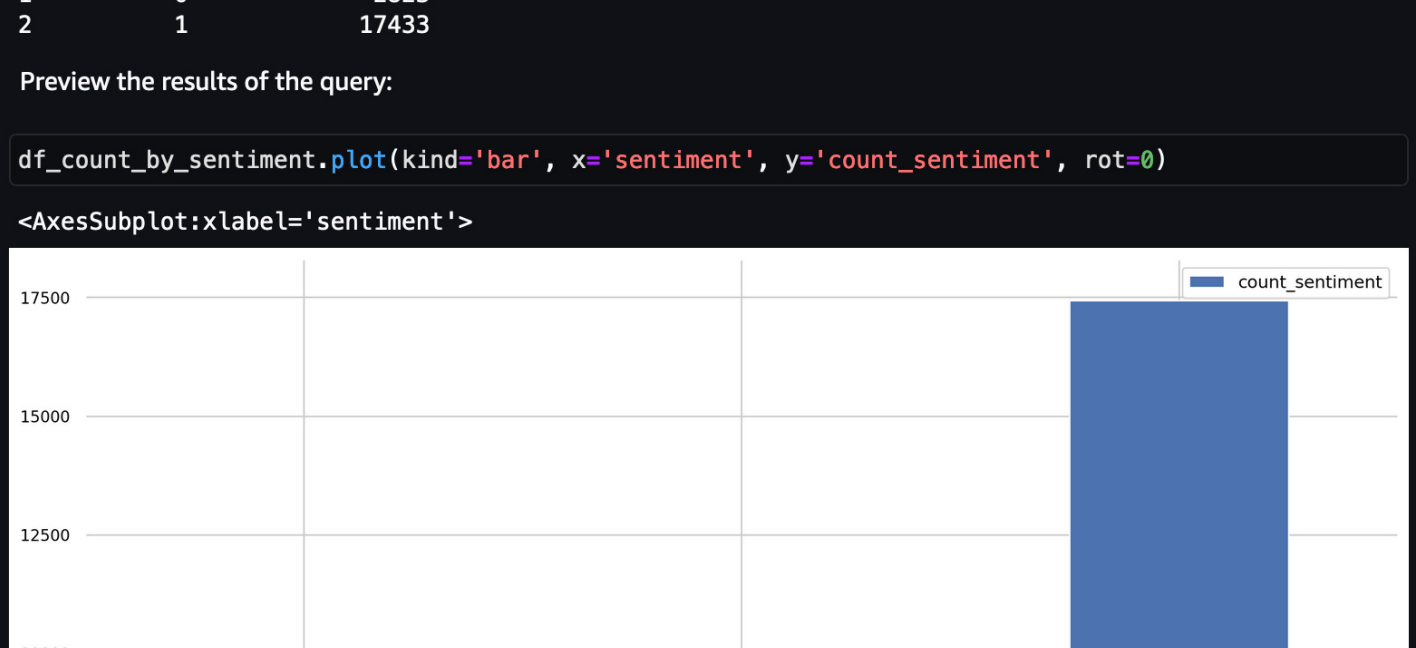
Preview the results of the query:

[84]:

```
df_count_by_sentiment.plot(kind='bar', x='sentiment', y='count_sentiment', rot=0)
```

[84]:

```
<AxesSubplot:xlabel='sentiment'>
```



Exercise 3

Use Amazon Athena query with the standard SQL statement passed as a parameter, to calculate the total number of reviews per `product_category` in the table `reviews`.

Instructions: Pass the SQL statement of the form

```
SELECT category_column, COUNT(column_name) AS new_column_name
FROM table_name
GROUP BY category_column
ORDER BY new_column_name DESC
```

as a triple quote string into the variable `statement_count_by_category`. Please use the column `sentiment` in the `COUNT` function and give it a new name `count_sentiment`.

[85]:

```
# Replace all None
### BEGIN SOLUTION - DO NOT delete this comment for grading purposes
statement_count_by_category = """
SELECT product_category, COUNT(sentiment) AS count_sentiment
FROM reviews
GROUP BY product_category
ORDER BY count_sentiment DESC
"""
### END SOLUTION - DO NOT delete this comment for grading purposes
print(statement_count_by_category)
```

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

Query data in Amazon Athena database passing the prepared SQL statement:

[86]:

```
%%time
df_count_by_category = wr.athena.read_sql_query(
    sql=statement_count_by_category,
    database=database_name
)

df_count_by_category

# EXPECTED OUTPUT
# Dresses: 6145
# Knits: 4626
# Blouses: 2983
# Sweaters: 1380
# Pants: 1350
# ...
```

```
/opt/conda/lib/python3.10/site-packages/awswrangler/_distributed.py:105: FutureWarning: promote h
as been superseded by mode='default'.
  return cls.dispatch_func(func)(*args, **kw)
CPU times: user 203 ms, sys: 6.28 ms, total: 210 ms
Wall time: 4.39 s
```

[86]:

	product_category	count_sentiment
0	Dresses	6145
1	Knits	4626
2	Blouses	2983
3	Sweaters	1380
4	Pants	1350
5	Jeans	1104
6	Fine gauge	1059
7	Skirts	903
8	Jackets	683
9	Lounge	669
10	Swim	332
11	Outerwear	319
12	Shorts	304
13	Sleep	214
14	Legwear	158
15	Intimates	147
16	Layering	132
17	Trend	118

3.3. Which product categories are highest rated by average sentiment?

Set the SQL statement to find the average sentiment per product category, showing the results in the descending order:

[87]:

```
statement_avg_by_category = """
SELECT product_category, AVG(sentiment) AS avg_sentiment
FROM {}
GROUP BY product_category
ORDER BY avg_sentiment DESC
""".format(table_name)

print(statement_avg_by_category)
```

```
SELECT product_category, AVG(sentiment) AS avg_sentiment
FROM reviews
GROUP BY product_category
ORDER BY avg_sentiment DESC
```

Query data in Amazon Athena database passing the prepared SQL statement:

[88]:

```
%%time
df_avg_by_category = wr.athena.read_sql_query(
    sql=statement_avg_by_category,
    database=database_name
)
```

```
/opt/conda/lib/python3.10/site-packages/awswrangler/_distributed.py:105: FutureWarning: promote h
as been superseded by mode='default'.
  return cls.dispatch_func(func)(*args, **kw)
CPU times: user 188 ms, sys: 6.56 ms, total: 195 ms
Wall time: 3.33 s
```

Preview the query results in the temporary S3 bucket: `s3://aws-athena-query-results-ACCOUNT-REGION/`

Instructions:

- open the link
- check the name of the S3 bucket
- briefly check the content of it

[89]:

```
from IPython.core.display import display, HTML

display(HTML('<b>Review <a target="top" href="https://s3.console.aws.amazon.com/s3/buckets/aws-ath
```

```
/tmp/ipykernel_2572/920550784.py:1: DeprecationWarning: Importing display from IPython.core.displ
ay is deprecated since IPython 7.14, please import from IPython display
  from IPython.core.display import display, HTML
```

Review Amazon S3 buckets

Preview the results of the query:

[90]:

```
df_avg_by_category
```

[90]:

	product_category	avg_sentiment
0	Layering	0.780303
1	Jeans	0.746377
2	Lounge	0.745889
3	Sleep	0.710280
4	Shorts	0.707237
5	Pants	0.705185
6	Intimates	0.700680
7	Jackets	0.699854
8	Skirts	0.696567
9	Legwear	0.696203
10	Fine gauge	0.692162
11	Outerwear	0.683386
12	Knits	0.653913
13	Swim	0.644578
14	Dresses	0.643287
15	Sweaters	0.641304
16	Blouses	0.641301
17	Trend	0.483051

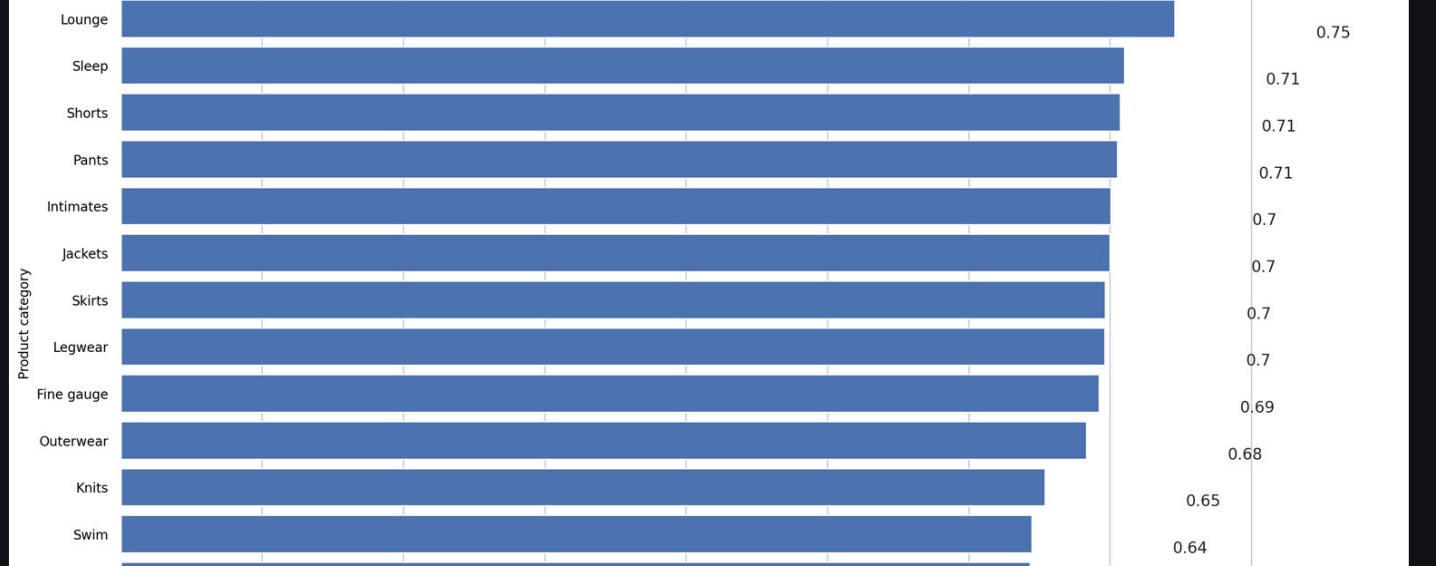
Visualization

[91]:

```
def show_values_barplot(axs, space):
    def _show_on_plot(ax):
        for p in ax.patches:
            _x = p.get_x() + p.get_width() + float(space)
            _y = p.get_y() + p.get_height()
            value = round(float(p.get_width()),2)
            ax.text(_x, _y, value, ha="left")

    if isinstance(axs, np.ndarray):
        for idx, ax in np.ndenumerate(axs):
            _show_on_plot(ax)
    else:
        _show_on_plot(axs)
```

[92]:

```
# Create plot
barplot = sns.barplot(
    data = df_avg_by_category,
    y='product_category',
    x='avg_sentiment',
    color="b",
    saturation=1
)

# Set the size of the figure
sns.set(rc={'figure.figsize':(15.0, 10.0)})

# Set title and x-axis ticks
plt.title('Average sentiment by product category')
#plt.xticks([-1, 0, 1], ['Negative', 'Neutral', 'Positive'])

# Helper code to show actual values afters bars
show_values_barplot(barplot, 0.1)

plt.xlabel("Average sentiment")
plt.ylabel("Product category")

plt.tight_layout()
# Do not change the figure name - it is used for grading purposes!
plt.savefig('avg_sentiment_per_category.png', dpi=300)

# Show graphic
plt.show(barplot)
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/_core.py:1225: FutureWarning: is_categorical_dtyp
e is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype)
instead
  if pd.api.types.is_categorical_dtype(vector):
/opt/conda/lib/python3.10/site-packages/seaborn/_core.py:1225: FutureWarning: is_categorical_dtyp
e is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype)
instead
  if pd.api.types.is_categorical_dtype(vector):
```