

**Data Warehousing & Integration  
IE6750  
Spring 2025**

**Customer Personality Analysis**

Milestone 6 (starts on pg. 4)  
Group 5

Anushka Jami  
Barath Keshav Sriram Kumaran

[jami.a@northeastern.edu](mailto:jami.a@northeastern.edu)  
[sriramkumaran.b@northeastern.edu](mailto:sriramkumaran.b@northeastern.edu)

**4/13/2025**

## **Problem Statement:**

Customer Personality Analysis is a critical process that enables businesses to gain a deep understanding of their customer segments. By analyzing customer data, businesses can separate their customer base into distinct groups based on various characteristics, needs, behaviors, and preferences. The goal of this analysis is to provide businesses with insights into how different types of customers interact with products and services, ultimately allowing companies to customize their offerings to meet the unique demands of each customer segment.

The key objective of customer personality analysis is to optimize marketing efforts and product development by identifying and targeting specific customer segments that are more likely to engage with and purchase particular products. Rather than investing resources in marketing a product to every customer within a business's database, customer personality analysis helps identify which customer segment is most likely to respond to a given product and marketing effort. By focusing on these high-potential segments, businesses can enhance marketing efficiency, improve customer satisfaction, and ultimately drive higher sales.

By leveraging this customer personality analysis, businesses can make more informed decisions about product development, promotional strategies, and resource allocation. This approach allows for more personalized marketing campaigns and a deeper understanding of customer behavior, leading to increased customer sales and better business outcomes.

## Datasets and Dimensions:

1. <https://www.kaggle.com/datasets/imakash3011/customer-personality-analysis>
2. Datasets extracted from:  
<https://www.bls.gov/news.release/pdf/cesan.pdf>
  - a. Tables

Expenditure_ID	Year	Category	Subcategory	Amount (USD)
1	2021	Food	Food at Home	5,259
2	2021	Food	Food Away from Home	3,030
3	2021	Food	Total (Food)	8,289
4	2022	Food	Food at Home	5,703
5	2022	Food	Food Away from Home	3,639
6	2022	Food	Total (Food)	9,343
7	2023	Food	Food at Home	6,053
8	2023	Food	Food Away from Home	3,933
9	2023	Food	Total (Food)	9,985
10	2021	Housing	Owned Dwellings	7,591
11	2021	Housing	Rented Dwellings	4,684
12	2021	Housing	Other Lodging	983
13	2021	Housing	Total (Housing)	22,624
14	2022	Housing	Owned Dwellings	8,230
15	2022	Housing	Rented Dwellings	4,990

i.

	A	B	C	D	E	F	G
	Item	All Consumer Units	Married Couple Only	Married Couple with Children	Other Married Couple Consumer Units	One Parent, at Least One Child Under 18	Single Person and Other Consumer Un
1							
2	Housing	32.9	31.2	30.6	30.7	37.3	31
3	Transportation	17	16.9	17.4	19.3	19	10
4	Food	12.9	12.2	13.6	14.5	14.3	11
5	Personal Insurance & Pensions	12.4	11.9	15.1	13.3	9	10
6	Healthcare	8	10	6.9	7.8	4.9	7
7	Entertainment	4.7	5	4.9	4.1	3.8	4
	Other	12.1	12.0	11.6	10.3	11.7	10

ii.

<b>Spending Category</b>	<b>2020</b>	<b>2021</b>	<b>2022</b>	<b>2023</b>
Food	11.9	12.4	12.8	12.9
Alcoholic Beverages	0.8	0.8	0.8	0.8
Housing	34.9	33.8	33.3	32.9
Apparel and Services	2.3	2.6	2.7	2.6
Transportation	16	16.4	16.8	17
Healthcare	8.4	8.1	8	8
Entertainment	4.7	5.3	4.7	4.7
Personal Care Products and Services	1.1	1.2	1.2	1.2
Reading	0.2	0.2	0.2	0.2
Education	2.1	1.8	1.8	2.1
Tobacco Products and Smoking Supplies	0.5	0.5	0.5	0.5

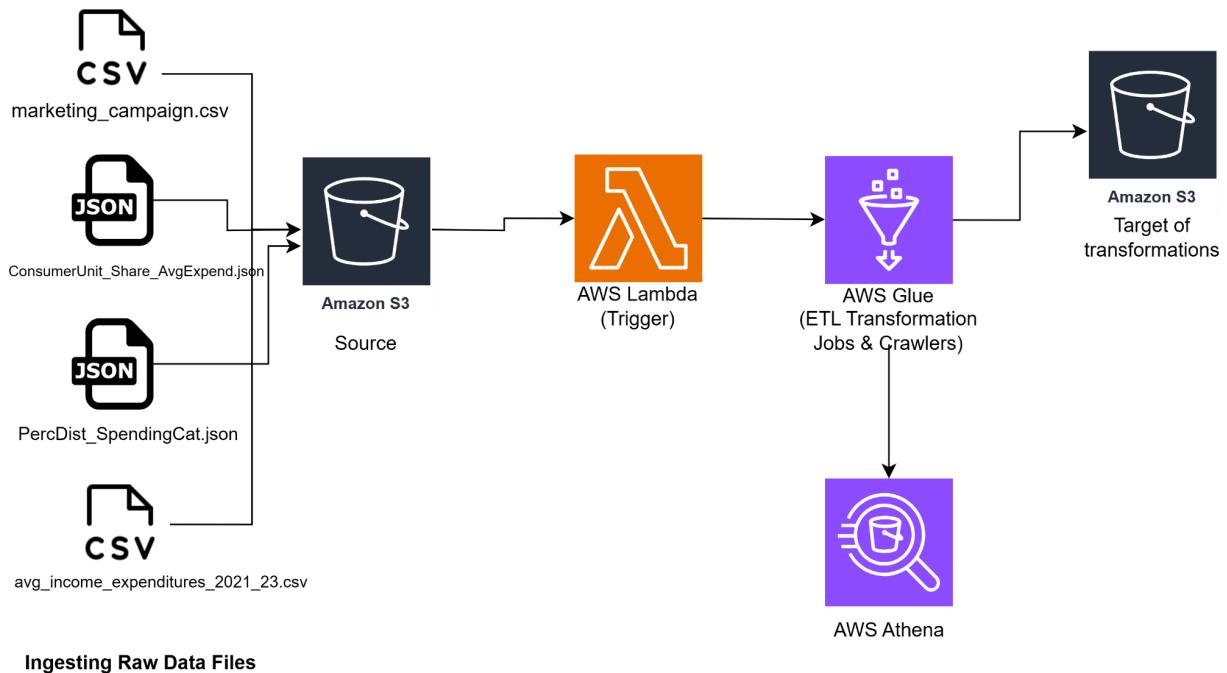
iii.

1. Education
2. Marital status
3. Children present
4. Income
5. Product Type
6. Method of Purchase

Datasets in analysis: Two csv files and two JSON files

1. marketing\_campaign.csv
2. avg\_income\_expenditures\_2021\_23.csv
3. ConsumerUnit\_Share\_AvgExpend.json
4. PercDist\_SpendingCat.json

## Architecture Diagram:



The architecture diagram above describes a cloud-based ETL pipeline that uses AWS services and components to prepare a target datasource ready for analysis. This pipeline processes raw data from both CSV and JSON files using Glue jobs to transform the raw data into a format to easily integrate multiple sources. Once the data is ready, it is loaded into a target S3 bucket to store the final analysis-ready result and is saved in a Glue Catalog data table to be queried using Athena. Below is a more detailed explanation of the architecture:

### 1. Ingesting the data files (CSV and JSON)

The raw data files, two CSV files and two JSON files, are ingested into an Amazon S3 bucket to be stored in their own folders within a cohesive project folder.

## 2. Amazon S3 source file storage

This S3 bucket serves as a place for raw data storage as well as intermediate dataset storage for data not fully processed. Files uploaded here are checked for filetype using a trigger.

## 3. AWS Lambda

### a. Trigger

A Lambda function is triggered when a file is uploaded to the source S3 bucket, warning when the file is not a CSV or JSON as expected.

## 4. AWS Glue

### a. ETL Transformation Jobs

Data was processed using ETL Notebook jobs written in Python. These were used to clean the data by dropping unnecessary columns and renaming column names for standardization and interpretability. In order to merge data from the four tables, three were merged on the "Spending Category" / "Item" / "Category" columns, the result of which was then merged to the fourth using a mapping column to map conditions of the dataset to values in the initial merged dataset.

### b. Crawlers

A crawler was created for each step in the transformation process, used to catalog the data into a Glue Database. These were then viewable and queryable using AWS Athena.

## 5. AWS Athena

Athena was used to query the data at different steps throughout the transformation process and additionally to query the final resulting data table.

## 6. Amazon S3 target file storage

This S3 bucket serves as a storage space for the output of the final Glue ETL transformation. This dataset is a clean and analysis-ready dataset that can be downloaded in a CSV file.

## Data Flow screenshots

The Source Bucket is created with a folder for each file to be able to retrieve its schema.

Amazon S3 < consumerproject-group5 Info

Objects (8)

Name	Type	Last modified	Size	Storage class
avg_spending/	Folder	-	-	-
consumer_unit_exp_clean/	Folder	-	-	-
consumer_unit_exp_json/	Folder	-	-	-
marketing_campaign_dropped/	Folder	-	-	-
marketing_campaign/	Folder	-	-	-
PercDist_SpendingCat.xlsx	xlsx	April 13, 2025, 14:47:40 (UTC-04:00)	9.0 KB	Standard
spending_cat_percent_clean/	Folder	-	-	-
spending_cat_percent_json/	Folder	-	-	-

Creating a Lambda function and adding a trigger that warns when a file is added, which is not in .csv or .json format.

Lambda > Functions > projectfunction

projectfunction

Function overview Info

Diagram | Template

projectfunction

S3

+ Add trigger

+ Add destination

Description

Last modified 2 hours ago

Function ARN arn:aws:lambda:us-east-1:651411371317:function:projectfunction

Function URL Info

## Code source for the Trigger.

The screenshot shows the AWS Lambda function editor interface. The left sidebar displays the project structure under 'PROJECTFUNCTION' with 'lambda\_function.py' selected. The main area shows the Python code for the Lambda function:

```

# coding: utf-8
import json
import logging

logger = logging.getLogger()
logger.setLevel(logging.INFO)

def lambda_handler(event, context):
    try:
        # Log the event
        print("Received event:", json.dumps(event))

        # Extract file key from the event
        s3_file_key = event["Records"][0]["s3"]["object"]["key"]
        bucket_name = event["Records"][0]["s3"]["bucket"]["name"]

        # Get file extension
        file_extension = s3_file_key.split('.')[1].lower()

        # Allowed formats
        allowed_formats = ['csv', 'json']

        if file_extension not in allowed_formats:
            logger.warning(f"File {s3_file_key} in bucket {bucket_name} is not a CSV or JSON file.")
        else:
            logger.info(f"File {s3_file_key} is a valid {file_extension} file.")

        return {
            'statusCode': 200,
            'body': f'File format check complete for {s3_file_key}.'
        }

    except Exception as e:
        logger.error(f"Error checking file format: {e}")

```

The status bar at the bottom indicates the code has 37 lines, 10 spaces, is in UTF-8 encoding, and is written in Python. It also shows the Lambda service and the layout is set to U.S.

Log Event to show that the trigger has given a warning as there was a file added that was not in a .csv or .json format.

The screenshot shows the AWS CloudWatch Log Groups interface. The left sidebar lists various monitoring services: CloudWatch, AI Operations, Alarms, Logs, Metrics, X-Ray traces, Events, Application Signals, Network Monitoring, and Insights. Under 'Logs', 'Log groups' is selected, showing 'aws/lambda/projectfunction'. The main area displays log events for the specified log group and timestamp range:

Timestamp	Message
2025-04-13T18:47:40.662Z	INIT_START Runtime Version: python:3.13.v31 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:f713ac0af982f...
2025-04-13T18:47:40.753Z	START RequestId: 16af96e9-7789-4748-b345-ec62f3a5a704 Version: \$LATEST
2025-04-13T18:47:40.754Z	Received event: {"Records": [{"eventVersion": "2.1", "eventSource": "aws:s3", "awsRegion": "us-east-1", "eventTime": "2025-04-13T18:47:40.754Z", "s3": {"region": "us-east-1", "bucket": "consumerproject-group5", "key": "PercDist_SpendingCat.xlsx"}, "approximateByteSize": 128, "receiptHandle": "AQEjwX..."}]}
2025-04-13T18:47:40.755Z	[WARNING] 2025-04-13T18:47:40.754Z 16af96e9-7789-4748-b345-ec62f3a5a704 File PercDist_SpendingCat.xlsx in bucket consumerproject-group5 is not a CSV or JSON file.
2025-04-13T18:47:40.757Z	END RequestId: 16af96e9-7789-4748-b345-ec62f3a5a704
2025-04-13T18:47:40.757Z	REPORT RequestId: 16af96e9-7789-4748-b345-ec62f3a5a704 Duration: 2.70 ms Billed Duration: 3 ms Memory Size: 128 M...

The status bar at the bottom indicates the logs have 37 lines, 10 spaces, are in UTF-8 encoding, and are in Python. It also shows the CloudWatch service and the layout is set to U.S.

## Using AWS Glue for doing the cleaning, transformations, and the joining of all the tables.

The screenshot shows the AWS Glue Studio interface. On the left, there's a sidebar with navigation links for AWS Glue, ETL jobs, Data Catalog, Data Integration and ETL, and Legacy pages. The main area is titled "AWS Glue Studio Info" and contains sections for "Create job" (with options for Visual ETL, Notebook, and Script editor), "Example jobs" (with a "Create example job" button), and "Your jobs (5)". The "Your jobs" section lists five jobs:

Job name	Type	Created by	Last modified	AWS Glue version
<a href="#">consumerunit_exp_transform_2</a>	Glue ETL	Notebook	4/12/2025, 5:32:31 PM	5.0
<a href="#">spending_cat_percent_clean_2</a>	Glue ETL	Notebook	4/12/2025, 5:28:23 PM	5.0
<a href="#">marketing_camp_drop</a>	Glue ETL	Notebook	4/12/2025, 5:23:16 PM	5.0
<a href="#">initial_join</a>	Glue ETL	Notebook	4/12/2025, 5:18:46 PM	5.0
<a href="#">final_join</a>	Glue ETL	Notebook	4/12/2025, 5:05:06 PM	5.0

## Cleaning of the table Consumer Units Average Expenditure.

The screenshot shows the AWS Glue Studio Notebook interface for the "consumerunit\_exp\_transform\_2" job. The notebook tab is selected. The code cell at the top reads "Melt data table into long format preparing for the join". Below it is a Python code snippet:

```

[11]: from pyspark.sql.functions import expr
      # Read JSON array file with multiline option
      df = spark.read.option("multiline", "true").json("s3://consumerproject-group5/consumer_unit_exp_json/ConsumerUnit_Share_AvgExp")
      # List of columns to unpivot
      id_vars = ['Item']
      value_vars = [
          'All Consumer Units',
          'Married Couple Only',
          'Married Couple with Children',
          'Other Married Couple Consumer Units',
          'One Parent, at Least One Child Under 18',
          'Single Person and Other Consumer Units'
      ]
      # Build the stack expression
      n = len(value_vars)
      expr_str = ", ".join([f'{id_var}', '{col}' for col in value_vars])
      # Perform unpivot using selectExpr and stack
      df_melted = df.selectExpr("Item", f"stack({n}, {expr_str}) as (Consumer_Unit_Type, Percent)")
      # Cast Percent to double
      df_melted = df_melted.withColumn("Percent", expr("cast(Percent as double)"))
      # Show result
      df_melted.show()
  
```

At the bottom of the code cell, there's a note: "CodeWhisperer". The footer of the page includes standard AWS links: CloudShell, Feedback, Copyright 2025, Privacy, Terms, and Cookie preferences.

## Cleaning of the table Spending Category Percentage.

The screenshot shows the AWS Glue Data Catalog interface with a notebook titled "spending\_cat\_percent\_clean\_2". The notebook contains Python code for unpivoting a DataFrame. The code uses the `pyspark.sql.functions` module to select columns, join them into a single column, and then stack the data into a long format. It also converts the year and percent columns to integers and doubles respectively. The resulting DataFrame is then displayed as a table.

```
[3]: from pyspark.sql.functions import expr

# Columns to unpivot
id_vars = ['Spending Category']
value_vars = ['2020', '2021', '2022', '2023']

n = len(value_vars)
expr_str = ", ".join([f'{col}', '{col}' for col in value_vars])

# Unpivot to long format
df2_melted = df2.selectExpr(`Spending Category`, f"stack({n}, {expr_str}) as (Year, Percent)")

# Convert columns
df2_melted = df2_melted.withColumn("Year", expr("cast(Year as int)"))
df2_melted = df2_melted.withColumn("Percent", expr("cast(Percent as double)"))

df2_melted.show()
```

	Spending Category	Year	Percent
1	Food	2020	11.9
2	Food	2021	12.4
3	Food	2022	12.8
4	Food	2023	12.9
5	Alcoholic Beverages	2020	0.8
6	Alcoholic Beverages	2021	0.8

Dropping unused columns from the Marketing Campaign table.

AWS Glue

Getting started

ETL jobs

- Visual ETL
- Notebooks
- Job run monitoring

Data Catalog tables

Data connections

Workflows (orchestration)

Zero-ETL integrations [New](#)

▼ Data Catalog

- Databases
- Tables
- Stream schema registries
- Schemas
- Connections
- Crawlers
- Classifiers
- Catalog settings

► Data Integration and ETL

► Legacy pages

What's New [New](#)

Documentation [New](#)

AWS Marketplace

Enable compact mode

Last modified on 4/12/2025, 5:23:16 PM ([Stop notebook](#)) ([Download Notebook](#)) ([Actions ▾](#)) ([Save](#))

## marketing\_camp\_drop

Run

Notebook Script Job details Runs Data quality Schedules Version Control

Glue PySpark

### Dropping columns not needed

```
[5]: cols_to_drop = ["acceptedcmp3", "acceptedcmp4", "acceptedcmp5", "acceptedcmp1", "acceptedcmp2", "complain", "z_costcontact", "z_revenue", "response"]

df4 = df3.drop(*cols_to_drop)
df4.show()
```

id year_birth  education marital_status income kidhome teenhome dt_customer recency mntwines mntfruits mntmeatproducts mntfishproducts mntsweetproducts mngoldprods numdealspurchases numwebpurchases numcatalogpurchases numstorepurchases numwebvisitsmonth
5524  1957 Graduation  Single  58138  3  0  0  04-09-2012  58  635  88  546  7
172  88    88    3    8    10    4    4    7
2174  1954 Graduation  Single  46344  1  1  0  08-03-2014  38  11  1  6  1
2    1    6    2    1    1    1    2    5
4141  1965 Graduation  Together  71613  0  0  0  21-08-2013  26  426  49  127  1
111  21    42    1    8    2    10    4
6182  1984 Graduation  Together  26646  1  0  0  10-02-2014  26  11  4  20  4
10    3    5    2    2    0    4    6
5324  1981  Phd  Married  58293  1  0  0  19-01-2014  94  173  43  118  5
46    27    15    5    5    3    6    5
7446  1967  Master  Together  62513  0  0  1  09-09-2013  16  520  42  98  6
0    42    14    2    6    4    10    6

0 8 1 ⏪ CodeWhisperer 0

Doing the Initial join of the tables avg\_income\_expenditures\_2021\_23, ConsumerUnit\_Share\_AvgExpend, and PercDist\_SpendingCat, each of which has been cleaned and transformed.

The screenshot shows the AWS Glue Data Catalog interface with a notebook titled "initial\_join". The notebook contains Python code using AWS Glue DynamicFrame API to perform joins between three datasets: "dyg\_clean", "dyz\_clean", and "Category". The code converts the first dataset to a DynamicFrame, performs a join with "dyz\_clean" using "Spending Category" as the key, and then renames columns to "Percent" and "Year". The resulting DataFrame is then converted back to a Spark DataFrame and printed to show the joined data.

```

from awsglue.dynamicframe import DynamicFrame
dyg_clean = DynamicFrame.fromDF(dyg_clean, glueContext, "dyg_clean")

joined_dyf = dyf.join(
    paths1=["Spending Category"],
    frame2=dyg_clean,
    paths2=["Item"]
)

df = joined_dyf.toDF()
df = df.withColumnRenamed("Percent", "Spending_cat_year_percent")
df = df.withColumnRenamed("Year", "Spending_cat_year")
df.show()

+-----+-----+-----+-----+
| Item|Spending_cat_year_percent| Spending Category| Consumer_Unit_Type|Spending_cat_year|consumer_unit_per
+-----+-----+-----+-----+
| cent|                         |               |                         |                         |
+-----+-----+-----+-----+

```

This screenshot shows the continuation of the notebook. It performs a second join using the "Category" table. The joined DataFrame is then converted back to a DynamicFrame and shown. The final output DataFrame includes columns for expenditure\_id, amount, subcategory, category, and various consumer unit metrics like income\_exp\_year and consumer\_unit\_percent.

```

from awsglue.dynamicframe import DynamicFrame
dyz_clean = DynamicFrame.fromDF(dyz_clean, glueContext, "dyz_clean")

final_join = joined_dyf.join(
    paths1=["Spending Category"],
    frame2=dyz_clean,
    paths2=["Category"]
)

dz = final_join.toDF()
dz.show()

+-----+-----+-----+-----+-----+-----+
|expenditure_id| Item|amount (usd)|subcategory|Spending_cat_year_percent| Spending Category|Spending
+-----+-----+-----+-----+-----+-----+
| 2022| 85|Personal Insuranc...| 473|Life & Other Pers...| 12.0|Personal Insuranc...
| 2021| 2021|One Parent, at Le...| 9.0|Personal Insurance...|
| 2022| 86|Personal Insuranc...| 7,480|Pensions & Social...| 12.0|Personal Insuranc...
| 2021| 2021|One Parent, at Le...| 9.0|Personal Insurance...|
| 2022| 87|Personal Insuranc...| 7,873|Total (Personal I...| 12.0|Personal Insuranc...
| 2021| 2021|One Parent, at Le...| 9.0|Personal Insurance...|
+-----+-----+-----+-----+-----+-----+

```

Doing the Final join, in which the initial join and the marketing campaign table are joined together. The mapping of the join is shown.

The screenshot shows the AWS Glue Data Catalog interface. On the left, there's a sidebar with navigation links like 'AWS Glue', 'Getting started', 'ETL jobs', 'Data Catalog tables', 'Data connections', 'Workflows (orchestration)', 'Zero-ETL integrations', 'Data Catalog', 'Databases', 'Tables', 'Stream schema registries', 'Schemas', 'Connections', 'Crawlers', 'Classifiers', 'Catalog settings', 'Data Integration and ETL', and 'Legacy pages'. Below these are links for 'What's New', 'Documentation', and 'AWS Marketplace'. At the bottom left is a 'Enable compact mode' button. The main area is titled 'final\_join' and has tabs for 'Notebook', 'Script', 'Job details', 'Runs', 'Data quality', 'Schedules', and 'Version Control'. A 'Run' button is highlighted. The notebook content starts with a section titled 'Creating a mapping column with the join conditions' containing a complex PySpark DataFrame construction. Below it is a section titled 'Join tables on mapping columns' with a single line of code: `df2_filtered = df2.filter(col("consumer_unit_type").isNotNull())`. The status bar at the bottom shows 'CodeWhisperer' and a progress bar.

AWS crawlers are used to load into a Glue Database to visualize the tables using Athena.

The screenshot shows the AWS Glue interface with the 'Crawlers' section selected. On the left, there's a sidebar with navigation links for AWS Glue, Data Catalog, Data Integration and ETL, and Legacy pages. The main content area displays a table of crawlers with the following columns: Name, State, Last run, Last run time..., Log, and Table changes from... . All crawlers are listed as 'Ready' and have succeeded in their last run. The table includes rows for avg\_income\_exp, consumer\_unit\_ex, consumerunit\_e, final\_join, initial\_join, marketing\_camp, marketing\_camp\_c, spending\_cat\_p, and spending\_cat\_perc.

Name	State	Last run	Last run time...	Log	Table changes from...
avg_income_exp	Ready	Succeeded	April 9, 2025 at ...	<a href="#">View log</a>	1 created
consumer_unit_ex	Ready	Succeeded	April 9, 2025 at ...	<a href="#">View log</a>	1 created
consumerunit_e	Ready	Succeeded	April 10, 2025 a...	<a href="#">View log</a>	1 created
final_join	Ready	Succeeded	April 12, 2025 a...	<a href="#">View log</a>	1 created
initial_join	Ready	Succeeded	April 12, 2025 a...	<a href="#">View log</a>	1 created
marketing_camp	Ready	Succeeded	April 9, 2025 at ...	<a href="#">View log</a>	1 created
marketing_camp_c	Ready	Succeeded	April 11, 2025 a...	<a href="#">View log</a>	1 created
spending_cat_p	Ready	Succeeded	April 9, 2025 at ...	<a href="#">View log</a>	1 created
spending_cat_perc	Ready	Succeeded	April 10, 2025 a...	<a href="#">View log</a>	1 created

All the tables are in a Glue database called consumerproj.

The screenshot shows the AWS Glue interface with the 'Databases' section selected. The main content area displays the 'Database properties' for 'consumerproj'. It shows the database name, a brief description, its location (arn:aws:s3:::target-consumerproj-group5), and the date it was created (April 9, 2025 at 22:10:25). Below this, the 'Tables' section lists nine tables: avg\_spending, consumer\_unit\_ex, consumerunit\_ex, final\_join, joined\_1, marketing\_camp, marketing\_camp\_c, spending\_cat\_perc, and spending\_cat\_perc. Each table entry includes a 'Last updated (UTC)' timestamp, a 'Delete' button, and links to 'Table data', 'View data quality', and 'View statistics'.

Name	Database	Location	Description	Created on (UTC)
consumerproj	consumerproj	s3://consumerproj	-	April 9, 2025 at 22:10:25

Name	Database	Location	Classification	Deprecated	View data	Data quality	Column stati...
avg_spending	consumerproj	s3://consumerproj	CSV	-	<a href="#">Table data</a>	<a href="#">View data quality</a>	<a href="#">View statistics</a>
consumer_unit_ex	consumerproj	s3://consumerproj	CSV	-	<a href="#">Table data</a>	<a href="#">View data quality</a>	<a href="#">View statistics</a>
consumerunit_ex	consumerproj	s3://consumerproj	Parquet	-	<a href="#">Table data</a>	<a href="#">View data quality</a>	<a href="#">View statistics</a>
final_join	consumerproj	s3://target-consum	Parquet	-	<a href="#">Table data</a>	<a href="#">View data quality</a>	<a href="#">View statistics</a>
joined_1	consumerproj	s3://target-consum	Parquet	-	<a href="#">Table data</a>	<a href="#">View data quality</a>	<a href="#">View statistics</a>
marketing_camp	consumerproj	s3://consumerproj	CSV	-	<a href="#">Table data</a>	<a href="#">View data quality</a>	<a href="#">View statistics</a>
marketing_camp_c	consumerproj	s3://consumerproj	Parquet	-	<a href="#">Table data</a>	<a href="#">View data quality</a>	<a href="#">View statistics</a>
spending_cat_perc	consumerproj	s3://consumerproj	CSV	-	<a href="#">Table data</a>	<a href="#">View data quality</a>	<a href="#">View statistics</a>
spending_cat_perc	consumerproj	s3://consumerproj	Parquet	-	<a href="#">Table data</a>	<a href="#">View data quality</a>	<a href="#">View statistics</a>

The final Target Bucket where the joined tables are stored in with a folder for each to be able to retrieve the schema.

Amazon S3 < target-consumerproj-group5

**General purpose buckets**

- Directory buckets
- Table buckets
- Access Grants
- Access Points
- Object Lambda Access Points
- Multi-Region Access Points
- Batch Operations
- IAM Access Analyzer for S3

Block Public Access settings for this account

**Storage Lens**

- Dashboards
- Storage Lens groups
- AWS Organizations settings

Feature spotlight: 11

AWS Marketplace for S3

**Objects (2)**

Name	Type	Last modified	Size	Storage class
final_join/	Folder	-	-	-
joined_1/	Folder	-	-	-

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

## Visualizing the final table using Athena.

Amazon Athena > Query editor

**Tables (9)**

- avg\_spending
- consumer\_unit\_exp
- consumer\_unit\_exp\_clean
- final\_join
- joined\_1
- marketing\_campaign
- marketing\_campaign\_dropped
- spending\_cat\_percent
- spending\_cat\_percent\_clean

**Views (0)**

**SQL** Ln 1, Col 1

Run again Explain Cancel Clear Create Reuse query results up to 60 minutes ago

**Query results** | **Query stats**

Completed Time in queue: 65 ms Run time: 969 ms Data scanned: 306.49 KB

**Results (10)**

#	consumer_unit_type	expenditure_id	amount (usd)	subcategory	spending_cat_year_percent
1	Married Couple with Children	8	3,933	Food Away from Home	12.9
2	Married Couple with Children	8	3,933	Food Away from Home	12.4
3	Married Couple with Children	21	25,436	Total (Housing)	32.9
4	Married Couple with Children	14	8,230	Owned Dwellings	33.3
5	Married Couple with Children	92	9,011	Pensions & Social Security	11.8
6	Married Couple with Children	92	9,011	Pensions & Social Security	12.0
7	Married Couple with Children	51	951	Fees and Admissions	4.7
8	Married Couple with Children	48	908	Pets, Toys, Hobbies, Playgrounds	5.3
9	Married Couple with Children	45	925	Other Entertainment Supplies	4.7

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences