

End to End Data Engineering Project in AWS using Spark (Pyspark)

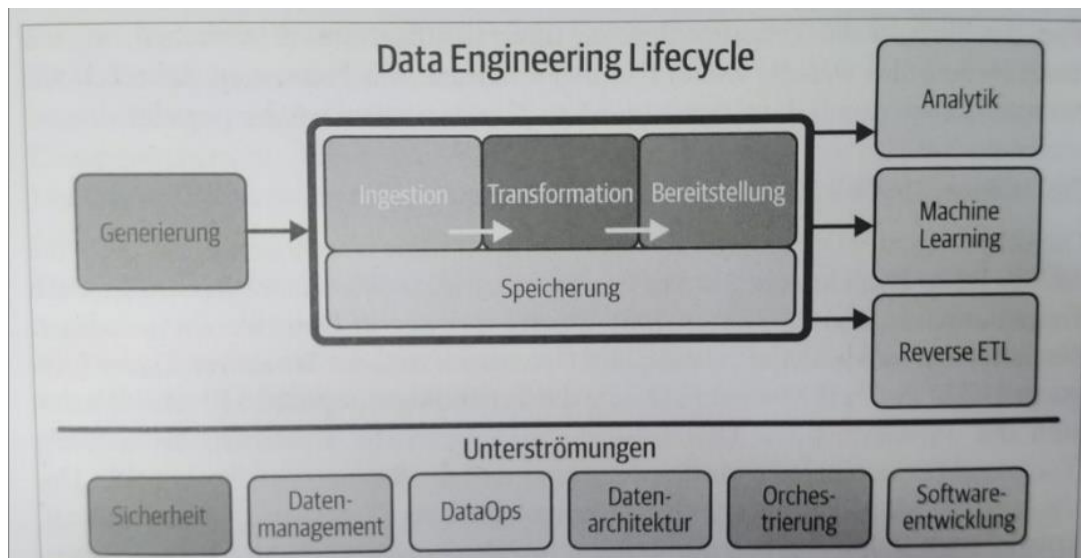
Used Technologies:

- AWS Cloud
- Infrastructure as a Code
- AWS S3 Data Lake
- AWS Glue:
 - AWS Glue Data Catalog
 - AWS Glue ETL Data Pipeline (CSV File → AWS Redshift)
 - Apache Spark
 - Jupyter Notebook, Pyspark (Data Cleaning, Transformation, Aggregation...)
- AWS Redshift Data Warehouse

1 Inhalt

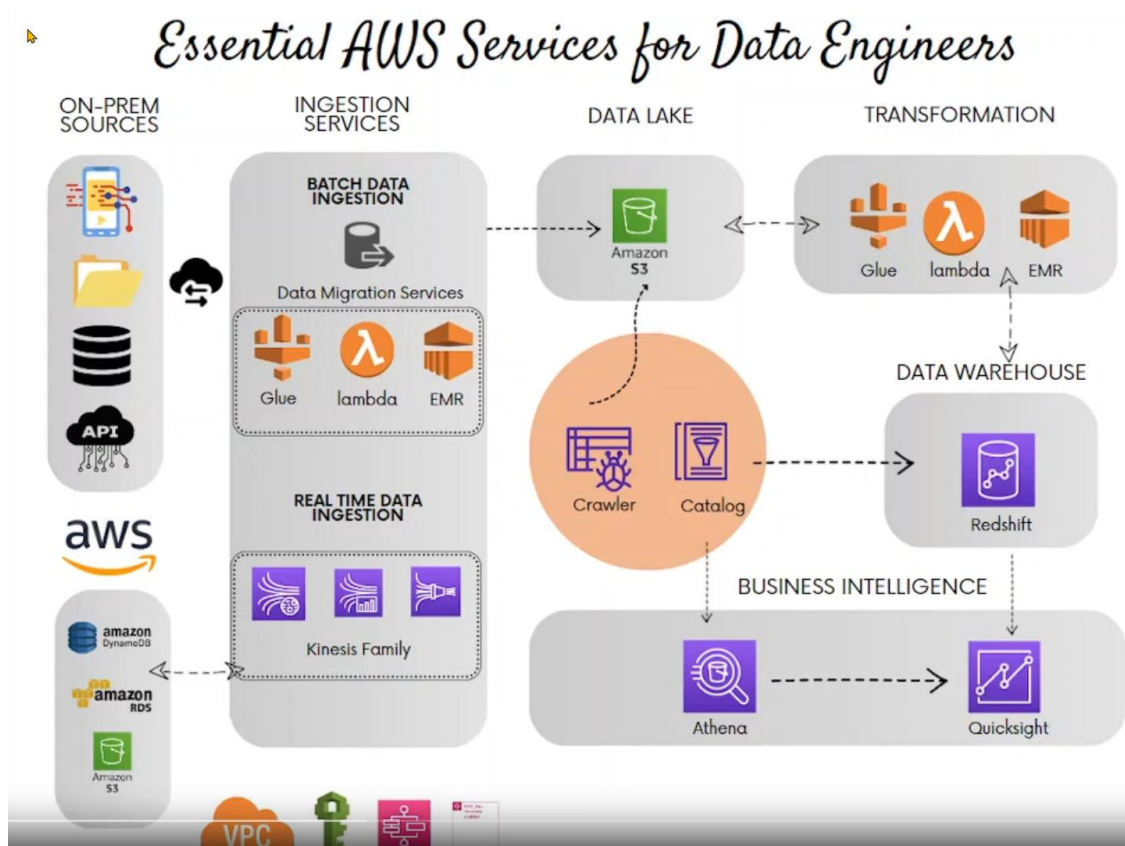
2	Overview: Data Engineering Lifecycle	2
3	AWS ETL architecture for Data Engineering.....	2
3.1	Project description	3
3.2	Project Workflow.....	3
4	Screenshots and explanation during development.....	4
4.1	Create Data Engineering System in AWS using „Infrastructure as Code“	4
4.2	Upload csv in S3.....	7
4.3	Create database	8
4.4	Create Crawler for tables with data source sales_record.csv	8
4.5	Run the crawler	13
4.6	ETL Job as interactive jupyter Notebook using pyspark.....	14
4.6.1	Jupyter Notebook for ETL Process using Spark / Pyspark – Step by Step as interactive Notebook.....	15
4.6.2	Jupyter Notebook as Pyspark Script that could be scheduled	21
5	AWS Redshift	22

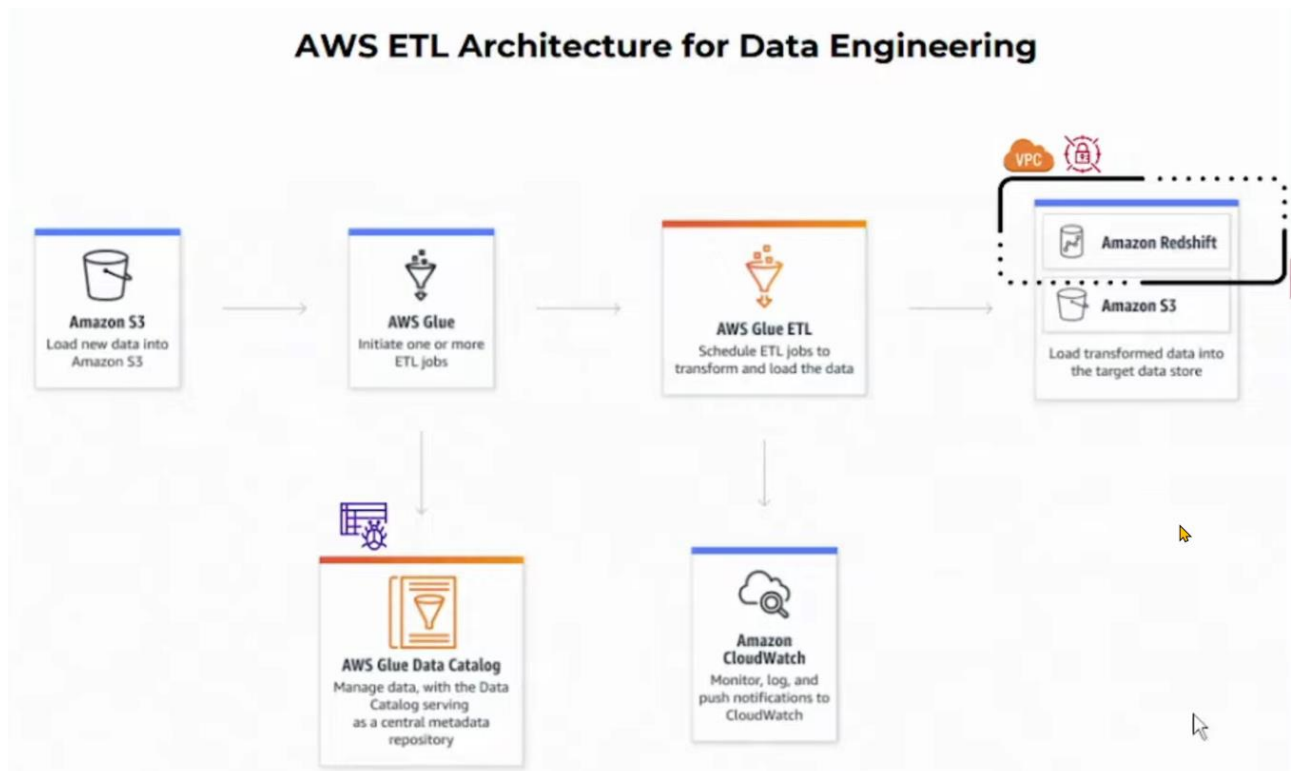
2 Overview: Data Engineering Lifecycle



Overview of the general Data Engineering Lifecycle. Data Pipeline from Source on the left to the destination on the right

3 AWS ETL architecture for Data Engineering





Overview of the AWS architecture for Data Engineering. Explanation see below Chapters.

3.1 Project description

Data-Engineer end to end project using AWS Cloud.

Using of infrastructure as a Code for creation of necessary AWS infrastructure.

Uploading raw data into S3 Data Lake. Building an end to end Data Pipeline using Pyspark for loading Data from S3 Source, Transformation, Aggregation, Data Quality, writing in AWS Redshift. Data Warehouse

3.2 Project Workflow

- Create Data Engineering System in AWS using „Infrastructure as Code“
- Test the created infrastructure
- Develop the necessary components (crawler, database..)
- Create an End to End Data Pipeline
 - Upload raw data (Source File: sales_records.csv) into S3 Data Lake (Data Source)
 - Destination: AWS Redshift database
 - S3 Data Lake Storage & Source of Data Pipeline
 - AWS Glue – using crawler to catalog data
 - Processing data using Pyspark within interactive Jupyter Notebook in Glue
 - Build Data Pipeline in Pyspark using Glue Jupyter interactive Notebook (Data Quality, Transformation, Aggregation...)
 - Reading data from s3 storage, processing it via Data Pipeline in Spark and then loading it into Redshift Data Warehouse (using dynamic frames and spark data frames)

4 Screenshots and explanation during development

4.1 Create Data Engineering System in AWS using „Infrastructure as Code“

This Chapter shows screenshots of the above explained development workflow in AWS.

CloudFormation > Stacks > Stack erstellen

Schritt 1
Stack erstellen

Schritt 2
Stack-Details abgeben

Schritt 3
Stack-Optionen konfigurieren

Schritt 4
Überprüfen und erstellen

Stack erstellen

Voraussetzung – Vorlage vorbereiten

Vorlage vorbereiten
Jeder Stack basiert auf einer Vorlage. Eine Vorlage ist eine JSON- oder YAML-Datei, die Konfigurationsinformationen über die AWS-Ressourcen enthält, die Sie in den Stack aufnehmen möchten.

☒ Vorlage ist bereit ☐ Eine Beispieldvorlage verwenden ☐ Vorlage in Designer erstellen

Vorlage angeben
Eine Vorlage ist eine JSON- oder YAML-Datei, die die Ressourcen und Eigenschaften Ihres Stacks beschreibt.

Vorlagenquelle
Wenn Sie eine Vorlage auswählen, wird eine Amazon-S3-URL generiert, in der sie gespeichert wird.

☐ Amazon-S3-URL
Geben Sie eine Amazon-S3-URL zu Ihrer Vorlage ein.

☒ Eine Vorlagendatei hochladen
Laden Sie Ihre Vorlage direkt auf die Konsole hoch.

☐ Von Git aus synchronisieren - *neu*
Synchronisieren Sie eine Vorlage aus Ihrem Git-Repository.

Eine Vorlagendatei hochladen

s3-glue-redshift-iam.yaml

JSON- oder YAML-formatierte Datei

S3-URL: <https://s3.us-east-1.amazonaws.com/cf-templates-vvf9gl0qt8t-us-east-1/2024-02-10T165012.190Z2gc-s3-glue-redshift-iam.yaml>

☰ Löschen für `arn:aws:cloudformation:us-east-1:131795324646:stack/etl/01020f50-c5ec-11ee-973b-0a073586d959` gestartet

CloudFormation > Stacks > etl

Stacks (1)

Status filtern: Aktiv

☒ Verschachtelte anzeigen

< 1 >

Stacks
etl 2024-02-10 17:52:11 UTC+0100 CREATE_IN_PROGRESS

etl

Stack-Aktionen Stack erstellen

< Stack-Info Ereignisse Ressourcen Ausgaben Parameter Vorlage Änderungssätze Git >

Ereignisse (1)

Zeitstempel	Logische ID	Status	Statusgrund
2024-02-10 17:52:11 UTC+0100	etl	CREATE_IN_PROGRESS	User Initiated

aws

Services

Suche

[Alt+S]

🔍 🔔 ⓘ ⚙️ Global ▾

Amazon S3

Buckets

Access Grants

Access Points

Objekt-Lambda-Zugriffspunkte

Multi-Region-Zugriffspunkte

Batch-Operationen

IAM Access Analyzer für S3

Einstellungen "Öffentlichen Zugriff beschränken" für dieses Konto

▼ Storage Lens

Dashboards

Storage-Lens-Gruppen

AWS Organizations-Einstellungen

Vorgestellte Funktion 7

▶ AWS Marketplace für S3

Amazon S3

► Konto-Snapshot

Die Speicherlinse bietet Einblicke in die Speichernutzung und Aktivitätstrends. [Weitere Informationen](#)

Storage-Lens-Dashboard anzeigen

Allzweck-Buckets

Verzeichnis-Buckets

Allzweck-Buckets (3) Info

🔄

📄 ARN kopieren

Leer

Löschen

Bucket erstellen

Buckets sind Container für in S3 gespeicherte Daten. [Weitere Informationen](#)

🔍 Buckets nach Namen suchen

< 1 > ⚙️

	Name ▲	AWS-Region ▼	Zugriff ▼	Erstellungsdatum ▼
<input type="radio"/>	cf-templates-vvf9gl0qot8t-us-east-1	USA Ost (Nord-Virginia) us-east-1	Bucket und Objekte nicht öffentlich	07.02.2024 07:28:48 PM CET
<input type="radio"/>	etl-databucket-jytroakfoopf	USA Ost (Nord-Virginia) us-east-1	Bucket und Objekte nicht öffentlich	07.02.2024 08:06:44 PM CET
<input type="radio"/>	etl-sourcedatabucket-zycqso8sqzqq	USA Ost (Nord-Virginia) us-east-1	Bucket und Objekte nicht öffentlich	07.02.2024 08:06:43 PM CET

aws

Services

Suche

[Alt+S]

🔍 🔔 ⓘ ⚙️ Nord-Virginia ▾

NEW: Amazon Redshift now supports zero-ETL integration with Amazon Aurora MySQL. Learn how you can get started applying near-real time analytics and machine learning on your transactional data today. [Learn more about Zero-ETL integrations](#)

🔍 ⚠️ 0 🔔 0 🔄 0 📄 4 🗑️ 0 ▼

Amazon Redshift > Dashboard für bereitgestellte Cluster

Dashboard für bereitgestellte Cluster Info

Testen Sie Amazon Redshift Serverless

Reservierte Knoten kaufen

Cluster erstellen

Ressourcenübersicht

Ressourcendaten für US East (N. Virginia) Region.

Gesamtzahl Knoten	On-Demand-Knoten	Reservierte Knoten	Reservierte Knoten verfügbar (0 von 0 genutzt)	Automatisierte Snapshots	Manuelle Snapshots
2	2	0	0	1	0

Cluster-Übersicht (1)

Beliebiger Status ▼

Cluster	Status
etl-redshift-cluster	🟢 Available

Alle Cluster anzeigen

Cluster-Metriken

🔄

Beliebige Cluster ▼

Letzte Stunde ▼

In CloudWatch anzeigen

Anzahl der Abfragen

Datenbankverbindungen

Verwendeter Festplattenspeicher

CPU-Auslastung

0

19:30 19:40 19:50 20:00 20:10 20:20

■ etl-redshift-cluster

Abragenübersicht

🔄

Beliebige Cluster ▼

Letzte Stunde ▼

Datenfreigaben

Autorisieren Sie andere AWS-Konten für den Zugriff auf Datenfreigaben, die in diesem AWS-Konto erstellt wurden. Ordnen Sie Datenfreigaben von anderen AWS-Konten zu oder lehnen Sie sie ab.

⚠️ Muss autorisiert werden

0

⚠️ Zuordnung anfordern

0

Alarme (0)

In CloudWatch anzeigen

Alarmname

Keine laufenden Alarme

Ereignisse (5)

Letzte Woche ▼

Datum	Ereignis
14 minutes ago 8:09 PM	Amazon Redshift cluster 'etl-redshift-cluster' has been created at 2024-02-07 19:09 UTC and is ready for use.

aws

Services

Suche

[Alt+S]

Nord-Virginia

AWS Glue

×

Getting started

ETL jobs

Visual ETL

Notebooks

Job run monitoring

Data Catalog tables

Data connections

▼ Data Catalog

Databases

Tables

Stream schema registries

Schemas

Connections

Crawlers

Classifiers

Catalog settings

► Data Integration and ETL

► Legacy pages

What's New

Documentation

AWS Marketplace

Enable compact mode

Enable new navigation

AWS Glue

>

Connectors

Connectors

info

Marketplace connectors

Subscribe to connectors from AWS partners to expand your data sources.

Go to AWS Marketplace

Custom connectors

Provide your own connector to expand your data sources. [Creating custom connectors](#)

Create custom connector

Connectors (0)

info

Actions

You can manage your connectors or use them to create connections.

Filter connections by property

< 1 >

Name	Type	Last modified
No connectors		
No connectors to display. You can create a custom connector or get a Marketplace connector .		
Create custom connector		

Connections (1)

info

Actions

Create connection

Create job

You can manage your connections or use a connection in a job.

Filter connections by property

< 1 >

Name	Type	Last modified
redshift-demo-connection	JDBC	Feb 07, 2024

4.2 Upload csv in S3 Data Lake

Amazon S3 > Buckets > etl-sourcedatabucket-zycqso8sqzqq > Hochladen

Hochladen Info

Fügen Sie die Dateien und Ordner hinzu, die Sie in S3 hochladen möchten. Verwenden Sie zum Hochladen von Dateien, die größer als 160 GB sind, die AWS-CLI, das AWS-SDK oder die Amazon S3 REST API. [Weitere Informationen](#)

Legen Sie Dateien und Ordner, die Sie hochladen möchten, per Drag-and-Drop hier ab oder wählen Sie **Dateien hinzufügen** oder **Ordner hinzufügen** aus.

Dateien und Ordner (1 Gesamt, 626.6 KB)

Entfernen

Dateien hinzufügen

Ordner hinzufügen

Alle Dateien und Ordner in dieser Tabelle werden hochgeladen.

< 1 >

<input type="checkbox"/>	Name	Ordner	Typ
<input type="checkbox"/>	sales_records.csv	-	applicatio

Ziel Info

Ziel
s3://etl-sourcedatabucket-zycqso8sqzqq

► Zieldetails
Die Bucket-Einstellungen, die sich auf neue Objekte auswirken, die im angegebenen Ziel gespeichert sind.

► Berechtigungen
Gewähren Sie öffentlichen Zugriff und Zugriff auf andere AWS-Konten.

► Eigenschaften
Geben Sie Speicherklasse, Verschlüsselungseinstellungen, Tags und mehr an.

Abbrechen

Hochladen

Upload erfolgreich

Sehen Sie sich die folgenden Details an.

Upload: Status Schließen

Die folgenden Informationen sind nicht mehr verfügbar, nachdem Sie diese Seite verlassen haben.

Zusammenfassung

Ziel
s3://etl-sourcedatabucket-zycqso8sqzqq

Erfolgreich
1 Datei, 626.6 KB (100.00%)

Fehler
0 Dateien, 0 B (0%)

Dateien und Ordner

Konfiguration

Dateien und Ordner (1 Gesamt, 626.6 KB)

< 1 >

Name	Ordner	Typ	Größe	Status	Fehler
sales_record...	-	application/...	626.6 KB	Erfolgreich	-

4.3 Create database

The screenshot shows the AWS Glue console with the 'Create a database' page. The left sidebar contains navigation links for 'Getting started', 'ETL jobs', 'Visual ETL', 'Notebooks', 'Job run monitoring', 'Data Catalog tables', 'Data connections', 'Workflows (orchestration)', 'Data Catalog' (expanded), 'Databases', 'Tables', 'Stream schema registries', 'Schemas', 'Connections', 'Crawlers', 'Classifiers', 'Catalog settings', 'Data Integration and ETL', and 'Legacy pages'. The main content area is titled 'Create a database' and includes a sub-header 'Database details'. It features three input fields: 'Name' (required, unique, lowercase, max 255 characters), 'Location - optional' (URI for clients), and 'Description - optional' (max 2048 characters). At the bottom right are 'Cancel' and 'Create database' buttons.

The screenshot shows the 'Databases' list in the AWS Glue console. It includes a search bar, a table with columns 'Name' and 'Description', and a single entry 'salesdb'.

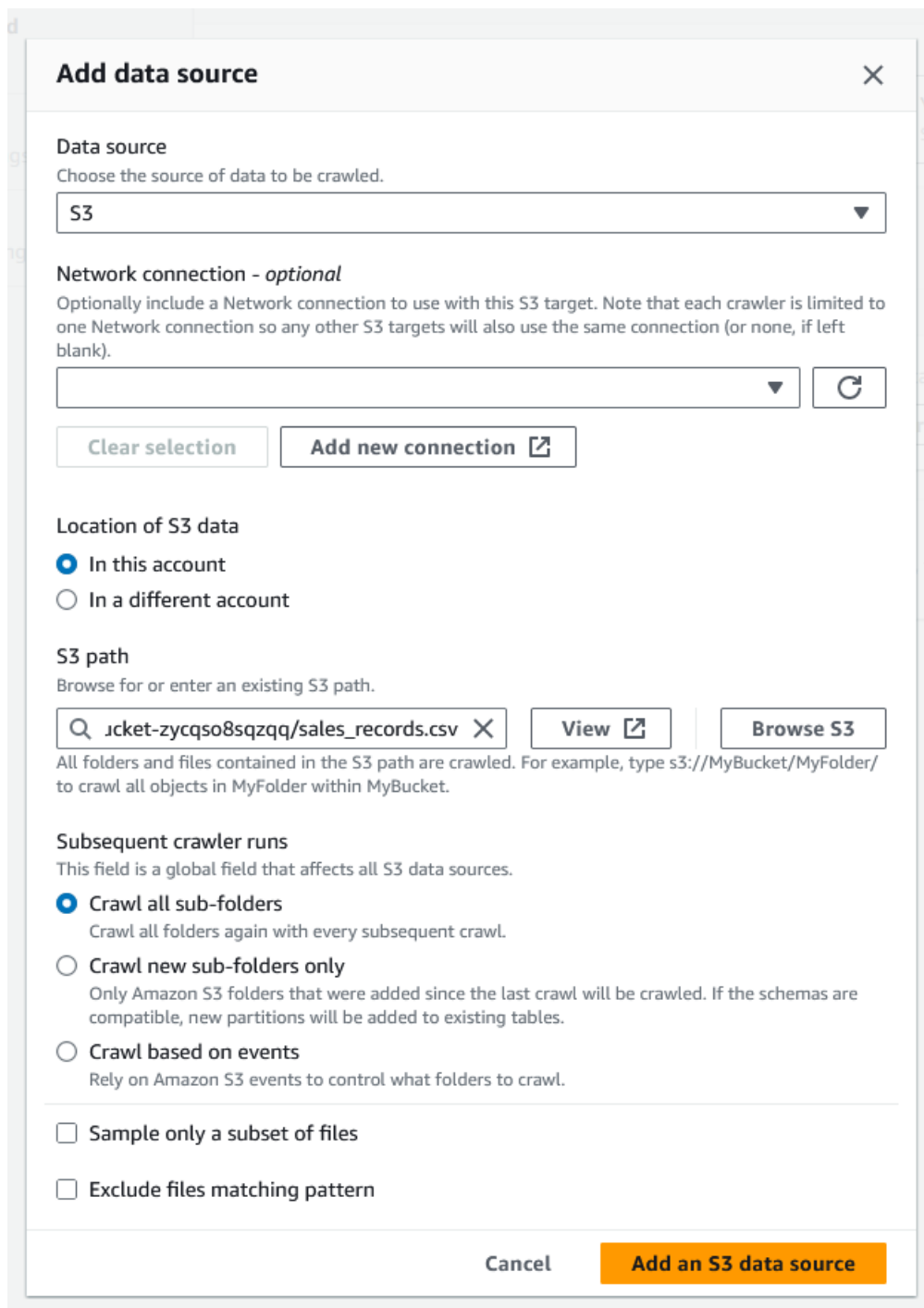
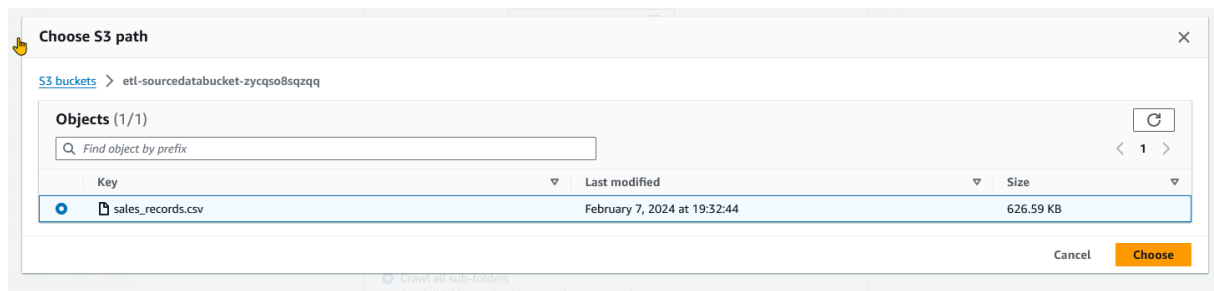
Name	Description
salesdb	

4.4 Create Crawler for tables with data source sales_record.csv

The screenshot shows the AWS Glue console 'Choose data sources and classifiers' page. The left sidebar is identical to the previous screenshot. The main content area is titled 'Choose data sources and classifiers' and includes a sub-header 'Data source configuration'. It features a radio button selection for 'Is your data already mapped to Glue tables?' (Not yet selected). Below this is a section for 'Data sources (0)' with an 'Add a data source' button. At the bottom is a section for 'Custom classifiers - optional' with a description. At the bottom right are 'Cancel', 'Previous', and 'Next' buttons.

The screenshot shows the 'Choose S3 path' dialog in the AWS Glue console. It includes a search bar, a table with columns 'Name' and 'Creation date', and three entries: 'cf-templates-vv9glQot8t-us-east-1', 'etl-databucket-jytr0akfoopf', and 'etl-sourcedatabucket-jycqso8sqzqg'. At the bottom right are 'Cancel' and 'Choose' buttons.

Name	Creation date
cf-templates-vv9glQot8t-us-east-1	February 7, 2024 at 18:28:48
etl-databucket-jytr0akfoopf	February 7, 2024 at 19:06:44
etl-sourcedatabucket-jycqso8sqzqg	February 7, 2024 at 19:06:43



[AWS Glue](#)

>

[Crawlers](#)

>

Add crawler

Step 1

[Set crawler properties](#)

Step 2

Choose data sources and classifiers

Step 3

Configure security settings

Step 4

Set output and scheduling

Step 5

Review and create

Choose data sources and classifiers

Data source configuration

Is your data already mapped to Glue tables?

☒ Not yet
Select one or more data sources to be crawled.

☐ Yes
Select existing tables from your Glue Data Catalog.

Data sources (1) [Info](#)

[Edit](#) [Remove](#) [Add a data source](#)

The list of data sources to be scanned by the crawler.

Type	Data source	Parameters
<input type="radio"/> S3	s3://etl-sourcedatabucket-zycqso8s...	Recrawl all

Custom classifiers - optional

A classifier checks whether a given file is in a format the crawler can handle. If it is, the classifier creates a schema in the form of a StructType object that matches that data format.

[Cancel](#) [Previous](#) [Next](#)

[AWS Glue](#)

>

[Crawlers](#)

>

Add crawler

Step 1

[Set crawler properties](#)

Step 2

[Choose data sources and classifiers](#)

Step 3

Configure security settings

Step 4

Set output and scheduling

Step 5

Review and create

Configure security settings

IAM role [Info](#)

Existing IAM role

Choose an IAM role

Q

etl-RedshiftIAMRole-pHSTcqtPNuao

[View](#)

etl-RedshiftIAMRole-pHSTcqtPNuao can be updated.

Lake Formation configuration - optional

Allow the crawler to use Lake Formation credentials for crawling the data source. [Learn more.](#)

☐ Use Lake Formation credentials for crawling S3 data source
Checking this box will allow the crawler to use Lake Formation credentials for crawling the data source. If the data source is registered in another account, you must provide the registered account ID. Otherwise, the crawler will crawl only those data sources associated to the account. Only applicable to S3, Glue Catalog, Iceberg, and Hudi data sources.

Security configuration - optional

Enable at-rest encryption with a security configuration.

[Cancel](#) [Previous](#) [Next](#)

[AWS Glue](#) > [Crawlers](#) > Add crawler

Step 1

[Set crawler properties](#)

Step 2

[Choose data sources and classifiers](#)

Step 3

[Configure security settings](#)

Step 4

Set output and scheduling

Step 5

Review and create

Set output and scheduling

Output configuration [Info](#)

Target database

Choose a database

Q |

salesdb

table name prefix - optional

Type a prefix added to table names

salesdb

Maximum table threshold - optional

This field sets the maximum number of tables the crawler is allowed to generate. In the event that this number is surpassed, the crawl will fail with an error. If not set, the crawler will automatically generate the number of tables depending on the data schema.

Type a number greater than 0

► Advanced options

Crawler schedule

You can define a time-based schedule for your crawlers and jobs in AWS Glue. The definition of these schedules uses the Unix-like [cron](#) syntax. [Learn more](#)

Frequency

On demand

Cancel

Previous

Next

[AWS Glue](#) > [Crawlers](#) > Add crawler

Step 1

[Set crawler properties](#)

Step 2

[Choose data sources and classifiers](#)

Step 3

[Configure security settings](#)

Step 4

[Set output and scheduling](#)

Step 5

Review and create

Review and create

Step 1: Set crawler properties [Edit](#)

Set crawler properties

Name	Description	Tags
sales	-	-

Step 2: Choose data sources and classifiers [Edit](#)

Data sources (1) [Info](#)

The list of data sources to be scanned by the crawler.

Type	Data source	Parameters
S3	s3://etl-sourcedatabucket-zycqso8sqzq...	Recrawl all

Step 3: Configure security settings [Edit](#)

Configure security settings

IAM role	Security configuration	Lake Formation configuration
etl-RedshiftIamRole-pHSTcqtPNuao	-	-

Step 4: Set output and scheduling [Edit](#)

Set output and scheduling

Database	Table prefix - optional	Maximum table threshold - optional	Schedule
salesdb	-	-	On demand

Cancel

Previous

Create crawler

One crawler successfully created
The following crawler is now created: "sales"

[AWS Glue](#) > [Crawlers](#) > sales

sales

Last updated (UTC)
February 7, 2024 at 19:48:03

Run crawlerEditDelete

Crawler properties

Name

sales

Description

-

Maximum table threshold

-

IAM role

etl-RedshiftIamRole-pHSTcqtPNuao

Security configuration

-

Database

salesdb

Lake Formation configuration

-

State

READY

Table prefix

-

Advanced settings

Crawler runs

ScheduleData sourcesClassifiersTags

Crawler runs (0)

The list of crawler runs for this crawler.

Filter data

Filter by a date and time range

< 1 >

Start time (UTC)

End time (UTC)

Current/last duration

Status

DPU hours

Table changes

You don't have any crawler runs.

Run crawler

One crawler successfully created
The following crawler is now created: "sales"

[AWS Glue](#) > Crawlers

Crawlers

A crawler connects to a data store, progresses through a prioritized list of classifiers to determine the schema for your data, and then creates metadata tables in your data catalog.

Crawlers (1) Info

Last updated (UTC)
February 7, 2024 at 19:49:23

ActionRunCreate crawler

Filter crawlers

< 1 >

☐

Name

▼

☐

sales

State

▼

Ready

Schedule

Last run

▼

-

Last run times...

▼

-

Log

-

Table changes fr...

-

4.5 Run the crawler

One crawler successfully created
The following crawler is now created: "sales"

[AWS Glue](#) > [Crawlers](#) > sales

sales

Last updated (UTC)
February 7, 2024 at 19:48:03

Refresh

Run crawler

Edit

Delete

Crawler properties

Name sales	IAM role etl-RedshiftIamRole-pHSTcqtPNuao	Database salesdb	State READY
Description -	Security configuration -	Lake Formation configuration -	Table prefix -
Maximum table threshold -			

▶ Advanced settings

Crawler runs

Schedule | Data sources | Classifiers | Tags

Crawler runs (0)

The list of crawler runs for this crawler.

< 1 > ⚙

Start time (UTC)	End time (UTC)	Current/last duration	Status	DPU hours	Table changes
You don't have any crawler runs.					
<div>Run crawler</div>					

Crawler successfully starting
The following crawler is now starting: "sales"

[AWS Glue](#) > [Crawlers](#) > sales

sales

Last updated (UTC)
February 7, 2024 at 19:48:03

Refresh

Run crawler

Edit

Delete

Crawler properties

Name sales	IAM role etl-RedshiftIamRole-pHSTcqtPNuao	Database salesdb	State READY
Description -	Security configuration -	Lake Formation configuration -	Table prefix -
Maximum table threshold -			

▶ Advanced settings

Crawler runs

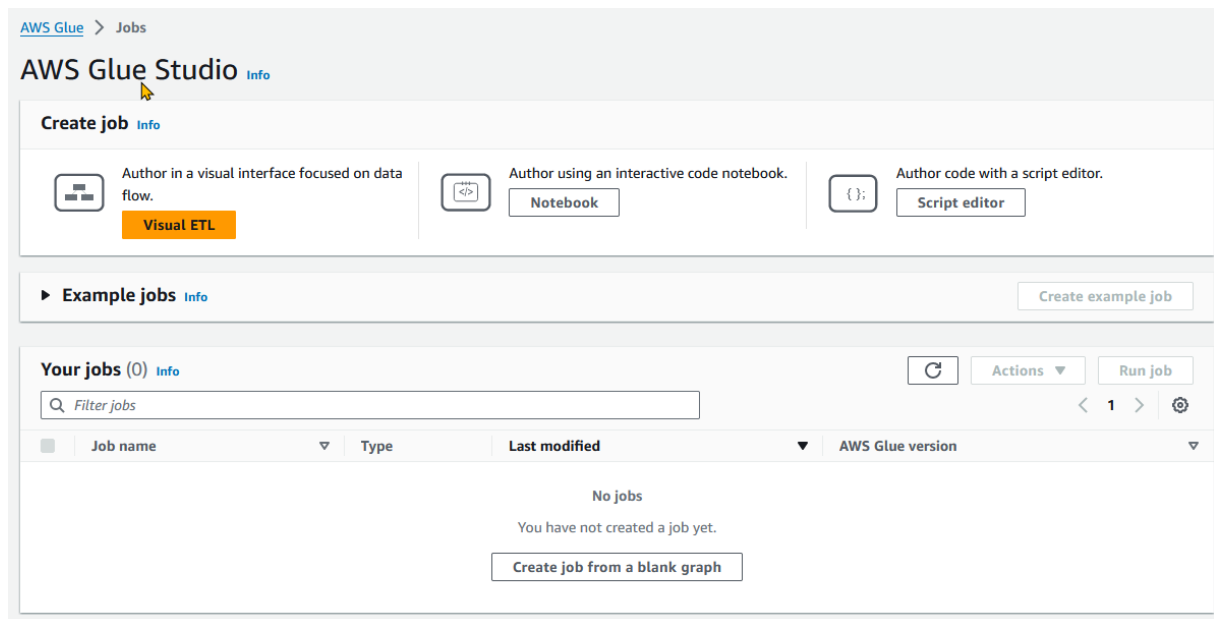
Schedule | Data sources | Classifiers | Tags

Crawler runs (1C)

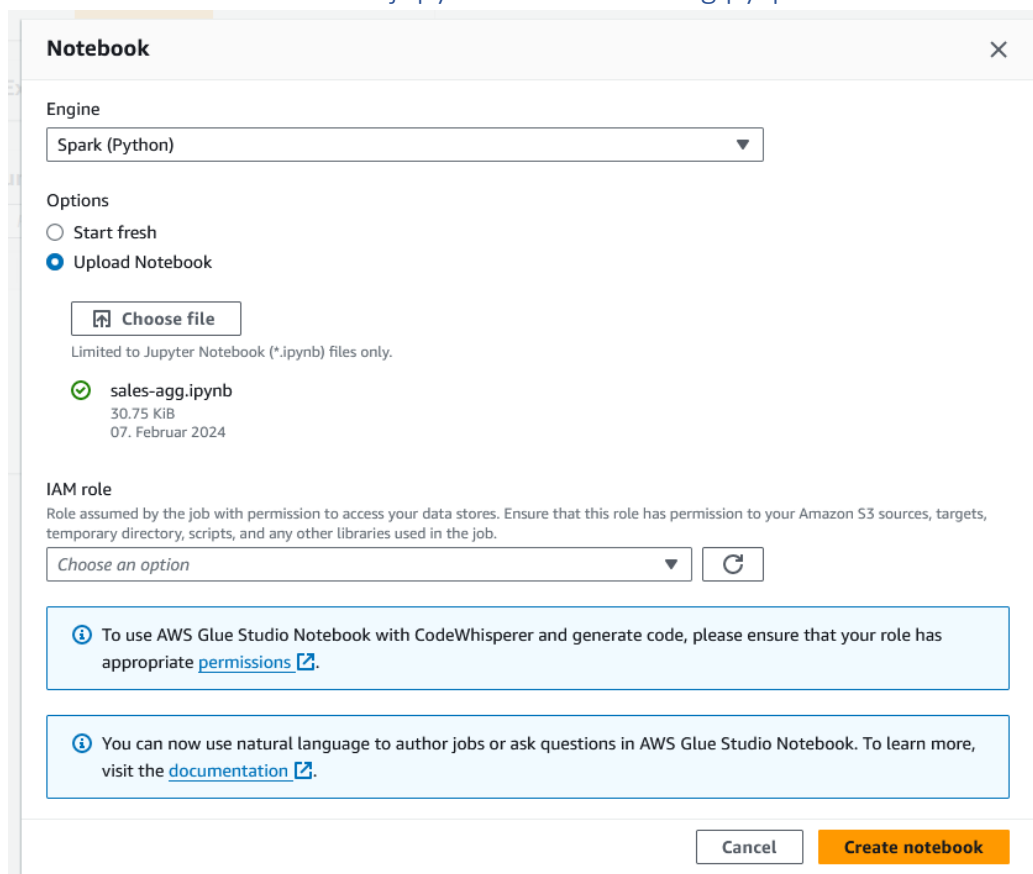
The list of crawler runs for this crawler.

< 1 > ⚙

Start time (UTC)	End time (UTC)	Current/last duration	Status	DPU hours	Table changes
○ February 7, 2024 at 19:50:30	-	05 s	Running	-	-



4.6 ETL Job as interactive jupyter Notebook using pyspark



Untitled job

Stop notebookDownload NotebookActionsSaveRun

NotebookScriptJob detailsRunsData quality - updatedSchedulesVersion Control

+

✂️📄🗑️▶️⏮️⏭️▶️▶️

Markdown

Glue PySpark

AWS Glue Studio Notebook

📄↑↓🖨️🔍🗑️

You are now running a AWS Glue Studio notebook: To start using your notebook you need to start an AWS Glue Interactive Session.

Optional: Run this cell to see available notebook commands ("magics").

```
[2]: %help

Welcome to the Glue Interactive Sessions Kernel
For more information on available magic commands, please type %help in any new cell.

Please view our Getting Started page to access the most up-to-date information on the Interactive Sessions kernel: https://docs.aws.amazon.com/glue/latest/dg/interactive-sessions.html
Installed kernel version: 0.30.1
```

Available Magic Commands

Sessions Magic

%help		Return a list of descriptions and input types for all magic commands.
%profile	String	Specify a profile in your aws configuration to use as the credentials provider.
%region	String	Specify the AWS region in which to initialize a session. Default from ~/.aws/config on Linux or macOS, or C:\Users\ USERNAME \.aws\config" on Windows.
%idle_timeout	Int	The number of minutes of inactivity after which a session will timeout. Default: 2880 minutes (48 hours).
%session_id_prefix	String	Define a String that will precede all session IDs in the format [session_id_prefix]-[session_id]. If a session ID is not provided, a random UUID will be generated.
%status		Returns the status of the current Glue session including its duration, configuration and executing user / role.
%session_id		Returns the session ID for the running session.
%list_sessions		Lists all currently running sessions by ID.
%stop_session		Stops the current session.
%glue_version	String	The version of Glue to be used by this session. Currently, the only valid options are 2.0, 3.0 and 4.0. Default: 2.0.

Selecting Job Types

%streaming	String	Sets the session type to Glue Streaming.
%etl	String	Sets the session type to Glue ETL.
%glue_ray	String	Sets the session type to Glue Ray.


Glue Config Magic

[AWS Glue](#) > Crawlers

Crawlers

A crawler connects to a data store, progresses through a prioritized list of classifiers to determine the schema for your data, and then creates metadata tables in your data catalog.

Crawlers (1) [Info](#)

Last updated (UTC)
 February 7, 2024 at 20:02:50
 

[Action](#) ▼

[Run](#)

[Create crawler](#)

View and manage all available crawlers.

<input type="checkbox"/>	Name ▼	State ▼	Schedule	Last run ▼	Last run timest... ▼	Log	Table chan...
<input type="checkbox"/>	sales	Ready		Succeeded	February 7, 2024 a...	View log	1 created

Workflow steps:

Import Libraries & initializing Spark and glue context (main entry point for AWS glue ETL)

```
[1]: %idle_timeout 2880
      %glue_version 3.0
      %worker_type G.1X
      %number_of_workers 5
      %connections redshift-demo-connection

import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
from pyspark.sql.functions import *
from awsglue.dynatmicframe import DynamicFrame

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)

Current idle_timeout is None minutes.
idle_timeout has been set to 2880 minutes.
Setting Glue version to: 3.0
Previous worker type: None
Setting new worker type to: G.1X
Previous number of workers: None
Setting new number of workers to: 5
Connections to be included:
redshift-demo-connection
Trying to create a Glue session for the kernel.
Session Type: glueetl
Worker Type: G.1X
Number of Workers: 5
Session ID: db9dda3f-c514-47dd-9f9e-3bb59152b943
Applying the following default arguments:
--glue_kernel_version 1.0.2
--enable-glue-datacatalog true
Waiting for session db9dda3f-c514-47dd-9f9e-3bb59152b943 to get into ready status...
Session db9dda3f-c514-47dd-9f9e-3bb59152b943 has been created.
```

Example: Create a DynamicFrame from a table in the AWS Glue Data Catalog , dropping null records and display its schema

```
: dyf = glueContext.create_dynamic_frame.from_catalog(database='salesdb', table_name='sales_records_csv')
  dyf = DropNullFields.apply(frame=dyf)
  dyf.printSchema()

null_fields []
root
|-- id: long
|-- region: string
|-- country: string
|-- item_type: string
|-- sales_channel: string
|-- order_priority: string
|-- order_date: string
|-- order_id: long
|-- ship_date: string
|-- units_sold: long
|-- unit_price: double
|-- unit_cost: double
|-- total_revenue: double
|-- total_cost: double
|-- total_profit: double
```

Dynamic DataFrame similar to spark dataframe but different syntax to perform action

👉 Example: Convert the DynamicFrame to a Spark DataFrame and display a sample of the data

```
[3]: df = dyf.toDF()
df.show()
```

	id	region	country	item_type	sales_channel	order_priority	order_date	order_id	ship_date	units_sold	unit_price	unit_cost	total_revenue	total_cost	total_profit
1	Central America a...	Antigua and Barbuda	Baby Food	Online	M	12/20/2013	957081544	1/11/2014	552	255.28	159.42	140914.56	87999.84	52914.72	
2	Central America a...	Panama	Snacks	Offline	C	7/5/2010	301644504	7/26/2010	2167	152.58	97.44	330640.86	211152.48	119488.38	
3	Europe	Czech Republic	Beverages	Offline	C	9/12/2011	478051030	9/29/2011	4778	47.45	31.79	226716.1	151892.62	74823.48	
4	Asia	North Korea	Cereal	Offline	L	5/13/2010	892599952	6/15/2010	9016	205.7	117.11	1854591.2	1055863.76	798727.44	
5	Asia	Sri Lanka	Snacks	Offline	C	7/20/2015	571902596	7/27/2015	7542	152.58	97.44	1150758.36	734892.48	415865.88	
6	Middle East and N...	Morocco	Personal Care	Offline	L	11/8/2010	412882792	11/22/2010	48	81.73	56.67	3923.04	2720.16	1202.88	
7	Australia and Oce...	Federated States ...	Clothes	Offline	H	3/28/2011	932776868	5/10/2011	8258	109.28	35.84	902434.24	295966.72	606467.52	
8	Europe	Bosnia and Herzeg...	Clothes	Online	M	10/14/2013	919133651	11/4/2013	927	109.28	35.84	101302.56	33223.68	68078.88	
9	Middle East and N...	Afghanistan	Clothes	Offline	M	8/27/2016	579814469	10/5/2016	8841	109.28	35.84	966144.48	316861.44	649283.04	
10	Sub-Saharan Africa	Ethiopia	Baby Food	Online	M	4/13/2015	192993152	5/7/2015	9817	255.28	159.42	2506083.76	1565026.14	941057.62	
11	Middle East and N...	Turkey	Office Supplies	Offline	C	9/25/2013	557156026	10/15/2013	3704	651.21					

Date Transformation

👉 Example: Perform data transformations

```
[4]: spark.sql("set spark.sql.legacy.timeParserPolicy=LEGACY")
sales_df = df.withColumn("Order_Date", to_date(unix_timestamp(col('order_date'), 'MM/dd/yyyy').cast('timestamp'))) \
    .withColumn("Ship_Date", to_date(unix_timestamp(col('ship_date'), 'MM/dd/yyyy').cast('timestamp')))

sales_df.show(10, True)
```

	id	region	country	item_type	sales_channel	order_priority	Order_Date	order_id	Ship_Date	units_sold	unit_price	unit_cost	total_revenue	total_cost	total_profit
1	Central America a...	Antigua and Barbuda	Baby Food	Online	M	2013-12-20	957081544	2014-01-11	552	255.28	159.42	140914.56	87999.84	52914.72	
2	Central America a...	Panama	Snacks	Offline	C	2010-07-05	301644504	2010-07-26	2167	152.58	97.44	330640.86	211152.48	119488.38	
3	Europe	Czech Republic	Beverages	Offline	C	2011-09-12	478051030	2011-09-29	4778	47.45	31.79	226716.1	151892.62	74823.48	
4	Asia	North Korea	Cereal	Offline	L	2010-05-13	892599952	2010-06-15	9016	205.7	117.11	1854591.2	1055863.76	798727.44	
5	Asia	Sri Lanka	Snacks	Offline	C	2015-07-20	571902596	2015-07-27	7542	152.58	97.44	1150758.36	734892.48	415865.88	
6	Middle East and N...	Morocco	Personal Care	Offline	L	2010-11-08	412882792	2010-11-22	48	81.73	56.67	3923.04	2720.16	1202.88	
7	Australia and Oce...	Federated States ...	Clothes	Offline	H	2011-03-28	932776868	2011-05-10	8258	109.28	35.84	902434.24	295966.72	606467.52	

👉 Group by Region and Country and calculate aggregate metrics

```
[5]: aggregate_df = sales_df.groupBy("Region", "Country", year("order_date").alias('year'), quarter("order_date").alias('quarter')).agg(
    sum("Total_Revenue").alias("Total_Revenue_By_Region_Country"),
    sum("Total_Cost").alias("Total_Cost_By_Region_Country"),
    sum("Total_Profit").alias("Total_Profit_By_Region_Country")
)
```

👉 Show the aggregated data (for demonstration purposes) ⓘ

```
[6]: aggregate_df.orderBy("year","quarter").show()
      aggregate_df.count()
```

Region	Country	year	quarter	Total_Revenue_By_Region_Country	Total_Cost_By_Region_Country	Total_Profit_By_Region_Country
Asia	South Korea	2010	1	44700.03	33153.72	11546.31
Middle East and N...	Iran	2010	1	2931671.66	2239089.38	692582.28
Central America a...	El Salvador	2010	1	1886886.1	1074250.03	812636.07
Middle East and N...	Algeria	2010	1	229050.88	75120.64	153930.24
Europe	Switzerland	2010	1	255802.95	171379.89	84423.06
Central America a...	Jamaica	2010	1	1573974.57	1268828.32	305146.25
Europe	Luxembourg	2010	1	1123251.46	662970.63	460280.83
Europe	Sweden	2010	1	4149902.4	2499528.36	1650374.04
Asia	Sri Lanka	2010	1	200419.52	65730.56	134688.96
Europe	Andorra	2010	1	3348578.1900000004	2008080.1500000001	1340498.04
Middle East and N...	Oman	2010	1	2680430.97	2015687.94	664743.03
Europe	Kosovo	2010	1	677247.76	399728.28	277519.48
Middle East and N...	Somalia	2010	1	2485916.64	1552431.96	933484.68
Central America a...	Saint Kitts and N...	2010	1	245126.7	164227.14	80899.56
Australia and Oce...	East Timor	2010	1	1271998.35	1099540.35	172458.0
Europe	Italy	2010	1	438322.08	143754.24	294567.84
Asia	Cambodia	2010	1	5682062.319999999	4130533.3400000003	1551528.9800000002
Europe	Bulgaria	2010	1	1829789.5	839380.1499999999	990409.35
Sub-Saharan Africa	Zambia	2010	1	5837205.6	4453417.91	1383787.69
North America	Mexico	2010	1	538028.59	373058.61	164969.98

only showing top 20 rows

👉 Renaming the cloumns and displaying the content in a sorted manner.

```
[7]: aggregate_df= aggregate_df.withColumnRenamed("Total_Revenue_By_Region_Country","Total_Revenue")\
      .withColumnRenamed("Total_Cost_By_Region_Country","Total_Cost")\
      .withColumnRenamed("Total_Profit_By_Region_Country","Total_Profit")
      aggregate_df.orderBy("year","quarter").show()
```

Region	Country	year	quarter	Total_Revenue	Total_Cost	Total_Profit
Europe	Serbia	2010	1	627485.76	205793.28	421692.48
Central America a...	Jamaica	2010	1	1573974.57	1268828.32	305146.25
Central America a...	El Salvador	2010	1	1886886.1	1074250.03	812636.07
Europe	Andorra	2010	1	3348578.1900000004	2008080.1500000001	1340498.04
Middle East and N...	Iran	2010	1	2931671.66	2239089.38	692582.28
Europe	Bulgaria	2010	1	1829789.5	839380.1499999999	990409.35
Europe	Luxembourg	2010	1	1123251.46	662970.63	460280.83
North America	Mexico	2010	1	538028.59	373058.61	164969.98
Middle East and N...	Algeria	2010	1	229050.88	75120.64	153930.24
Middle East and N...	Oman	2010	1	2680430.97	2015687.94	664743.03
Europe	Switzerland	2010	1	255802.95	171379.89	84423.06
Asia	Cambodia	2010	1	5682062.319999999	4130533.3400000003	1551528.9800000002
Europe	Kosovo	2010	1	677247.76	399728.28	277519.48
Australia and Oce...	East Timor	2010	1	1271998.35	1099540.35	172458.0
Europe	Italy	2010	1	438322.08	143754.24	294567.84
Europe	Sweden	2010	1	4149902.4	2499528.36	1650374.04
Central America a...	Saint Kitts and N...	2010	1	245126.7	164227.14	80899.56
Middle East and N...	Somalia	2010	1	2485916.64	1552431.96	933484.68
Asia	Sri Lanka	2010	1	200419.52	65730.56	134688.96
Sub-Saharan Africa	Zambia	2010	1	5837205.6	4453417.91	1383787.69

only showing top 20 rows



Example: Convert the Spark DataFrame to a DynamicFrame and display a sample of the data

```
[8]: dyf = DynamicFrame.fromDF(aggregate_df, glueContext, "dynamic_frame")
```

Example: load the dynamic frame into our Amazon Redshift cluster

```
[9]: dyf.show()

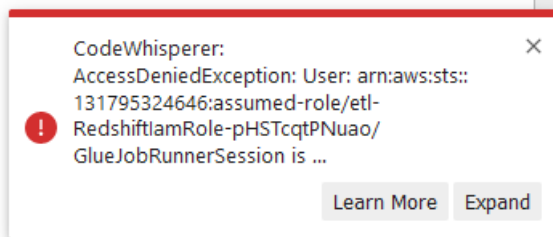
{"Region": "Europe", "Country": "Luxembourg", "year": 2010, "quarter": 1, "Total_Revenue": 1123251.46, "Total_Cost": 662970.63, "Total_Profit": 460280.83}
{"Region": "Europe", "Country": "Switzerland", "year": 2014, "quarter": 1, "Total_Revenue": 4429651.8, "Total_Cost": 2873705.61, "Total_Profit": 1555946.19}
{"Region": "Central America and the Caribbean", "Country": "Dominica", "year": 2010, "quarter": 2, "Total_Revenue": 1255966.53, "Total_Cost": 1085682.13, "Total_Profit": 170284.4}
{"Region": "Australia and Oceania", "Country": "Federated States of Micronesia", "year": 2012, "quarter": 2, "Total_Revenue": 5588354.33, "Total_Cost": 4029198.36, "Total_Profit": 1559155.97}
{"Region": "Europe", "Country": "Poland", "year": 2015, "quarter": 4, "Total_Revenue": 5740416.15, "Total_Cost": 4627522.4, "Total_Profit": 1112893.75}
{"Region": "Sub-Saharan Africa", "Country": "Namibia", "year": 2016, "quarter": 1, "Total_Revenue": 1861809.39, "Total_Cost": 1500860.64, "Total_Profit": 360948.75}
{"Region": "Europe", "Country": "Estonia", "year": 2011, "quarter": 2, "Total_Revenue": 657986.0, "Total_Cost": 396311.65, "Total_Profit": 261674.35}
{"Region": "Europe", "Country": "Armenia", "year": 2011, "quarter": 1, "Total_Revenue": 1000641.06, "Total_Cost": 647551.76, "Total_Profit": 353089.3}
{"Region": "Europe", "Country": "Denmark", "year": 2016, "quarter": 2, "Total_Revenue": 27948.05, "Total_Cost": 18724.31, "Total_Profit": 9223.74}
{"Region": "Middle East and North Africa", "Country": "Lebanon", "year": 2011, "quarter": 2, "Total_Revenue": 3937826.65, "Total_Cost": 2884792.9, "Total_Profit": 1053033.75}
{"Region": "Central America and the Caribbean", "Country": "Jamaica", "year": 2013, "quarter": 3, "Total_Revenue": 792397.3799999999, "Total_Cost": 629472.22, "Total_Profit": 162925.16}
{"Region": "Middle East and North Africa", "Country": "Afghanistan", "year": 2013, "quarter": 2, "Total_Revenue": 278255.2, "Total_Cost": 173767.8, "Total_Profit": 104487.4}
{"Region": "Europe", "Country": "Spain", "year": 2011, "quarter": 4, "Total_Revenue": 4544537.69, "Total_Cost": 3633328.41, "Total_Profit": 911209.28}
{"Region": "Asia", "Country": "Taiwan", "year": 2014, "quarter": 4, "Total_Revenue": 3204577.16, "Total_Cost": 1694851.36, "Total_Profit": 1509725.7999999998}
{"Region": "Europe", "Country": "Macedonia", "year": 2014, "quarter": 3, "Total_Revenue": 4672951.7, "Total_Cost": 3482977.09, "Total_Profit": 1189974.61}
{"Region": "Sub-Saharan Africa", "Country": "Liberia", "year": 2015, "quarter": 4, "Total_Revenue": 871139.5, "Total_Cost": 495960.85, "Total_Profit": 375178.65}
{"Region": "Central America and the Caribbean", "Country": "Jamaica", "year": 2017, "quarter": 2, "Total_Revenue": 6577657.34, "Total_Cost": 4587400.02, "Total_Profit": 1990257.3199999998}
{"Region": "Sub-Saharan Africa", "Country": "Senegal", "year": 2011, "quarter": 3, "Total_Revenue": 80527.23, "Total_Cost": 59726.52, "Total_Profit": 20800.71}
{"Region": "Asia", "Country": "Turkmenistan", "year": 2010, "quarter": 4, "Total_Revenue": 3508859.13, "Total_Cost": 3033126.73, "Total_Profit": 475732.4}
{"Region": "Middle East and North Africa", "Country": "Saudi Arabia", "year": 2013, "quarter": 4, "Total_Revenue": 9850970.7, "Total_Cost": 7127364.799999999, "Total_Profit": 2723605.92}
```

Writing dynamicFrame to Redshift

Authorization access problem

```
[10]: redshift_output = glueContext.write_dynamic_frame.from_jdbc_conf(
    frame=dyf,
    catalog_connection="redshift-demo-connection",
    connection_options={"dbtable": "public.Regionalsales", "database": "dev"},
    redshift_tmp_dir = "s3://aws-glue-assets-262136919150-us-east-1/temporary/",
    transformation_ctx = "redshift_output"
)
```

```
Py4JJavaError: An error occurred while calling o140.pyWriteDynamicFrame.
: java.io.IOException: com.amazon.ws.emr.hadoop.fs.shaded.com.amazonaws.services.s3.model.AmazonS3Exception: Access Denied (Service: Amazon S3; Status Code: 403; Error Code: AccessDenied; Request ID: A6IX3A6X22NX7BK4; S3 Extended Request ID: NDKfHB8P8vpeTqgQqn0AIT8nLWESsLUrDt9MgNfYZ/VptwFBkx9bVdat0sQzXpjoGazn2T1Dcus=; Proxy: null), S3 Extended Request ID: NDKfHB8P8vpeTqgQqn0AIT8nLWESsLUrDt9MgNfYZ/VptwFBkx9bVdat0sQzXpjoGazn2T1Dcus=
at com.amazon.ws.emr.hadoop.fs.s3n.Jets3tNativeFileSystemStore.list(Jets3tNativeFileSystemStore.java:303)
at com.amazon.ws.emr.hadoop.fs.s3n.S3NativeFileSystem.getFileStatus(S3NativeFileSystem.java:510)
at org.apache.hadoop.fs.FileSystem.exists(FileSystem.java:1690)
at com.amazon.ws.emr.hadoop.fs.EmrFileSystem.exists(EmrFileSystem.java:436)
at org.apache.spark.sql.execution.datasources.InsertIntoHadoopFsRelationCommand.run(InsertIntoHadoopFsRelationCommand.scala:124)
at org.apache.spark.sql.execution.command.DataWritingCommandExec.sideEffectResult$lzycompute(commands.scala:108)
at org.apache.spark.sql.execution.command.DataWritingCommandExec.sideEffectResult(commands.scala:106)
at org.apache.spark.sql.execution.command.DataWritingCommandExec.doExecute(commands.scala:131)
at org.apache.spark.sql.execution.SparkPlan.$anonfun$execute$1(SparkPlan.scala:185)
at org.apache.spark.sql.execution.SparkPlan.$anonfun$executeQuery$1(SparkPlan.scala:223)
at org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.scala:151)
at org.apache.spark.sql.execution.SparkPlan.executeQuery(SparkPlan.scala:220)
at org.apache.spark.sql.execution.SparkPlan.execute(SparkPlan.scala:181)
at org.apache.spark.sql.execution.QueryExecution.toRdd$lzycompute(QueryExecution.scala:134)
at org.apache.spark.sql.execution.QueryExecution.toRdd(QueryExecution.scala:133)
at org.apache.spark.sql.DataFrameWriter.$anonfun$runCommand$1(DataFrameWriter.scala:989)
at org.apache.spark.sql.catalyst.QueryPlanningTracker$.withTracker(QueryPlanningTracker.scala:107)
at org.apache.spark.sql.execution.SQLExecution$.withTracker(SQLExecution.scala:232)
```



CodeWhisperer: AccessDeniedException: User: arn:aws:sts::131795324646:assumed-role/etl-RedshiftIamRole-pHSTcqtPNuao/GlueJobRunnerSession is not authorized to perform: codewhisperer:GenerateRecommendations because no identity-based policy allows the codewhisperer:GenerateRecommendations action

OK

Solution:

First a lot of trial & error concerning role authorizations...but the solution was much simpler...
The adress for the redshift temp directory was wrong that caused the autohorization problem.

```
[10]: redshift_output = glueContext.write_dynamic_frame.from_jdbc_conf(  
    frame=dyf,  
    catalog_connection="redshift-demo-connection",  
    connection_options={"dbtable": "public.Regionalsales", "database": "dev"},  
    redshift_tmp_dir = "s3://aws-glue-assets-131795324646-us-east-1/temporary/",  
    transformation_ctx = "redshift_output"  
)
```

```
[ ]:
```

4.6.2 Jupyter Notebook as Pyspark Script that could be sheduled

NotebookScriptJob details2RunsData quality - updatedSchedulesVersion Control

Script Info

```
1
2 import sys
3 from awsglue.transforms import *
4 from awsglue.utils import getResolvedOptions
5 from pyspark.context import SparkContext
6 from awsglue.context import GlueContext
7 from awsglue.job import Job
8 from pyspark.sql.functions import *
9 from awsglue.dynamicframe import DynamicFrame
10
11 sc = SparkContext.getOrCreate()
12 glueContext = GlueContext(sc)
13 spark = glueContext.spark_session
14 job = Job(glueContext)
15 dyf = glueContext.create_dynamic_frame.from_catalog(database='salesdb', table_name='sales_records_csv')
16 dyf = DropNullFields.apply(frame=dyf)
17 dyf.printSchema()
18 df = dyf.toDF()
19 df.show()
20 spark.sql("set spark.sql.legacy.timeParserPolicy=LEGACY")
21 sales_df = df.withColumn("Order_Date", to_date(unix_timestamp(col('order_date'), 'MM/dd/yyyy').cast('timestamp'))) \
22             .withColumn("Ship_Date", to_date(unix_timestamp(col('ship_date'), 'MM/dd/yyyy').cast('timestamp')))
23
24 sales_df.show(10, True)
25
26 aggregate_df = sales_df.groupBy("Region", "Country", year("order_date").alias('year'), quarter("order_date").alias('quarter')).agg(
27     sum("Total_Revenue").alias("Total_Revenue_By_Region_Country"),
28     sum("Total_Cost").alias("Total_Cost_By_Region_Country"),
29     sum("Total_Profit").alias("Total_Profit_By_Region_Country")
30 )
31
32
33 aggregate_df.orderBy("year", "quarter").show()
34 aggregate_df.count()
35 aggregate_df= aggregate_df.withColumnRenamed("Total_Revenue_By_Region_Country", "Total_Revenue")\
36                             .withColumnRenamed("Total_Cost_By_Region_Country", "Total_Cost")\
37                             .withColumnRenamed("Total_Profit_By_Region_Country", "Total_Profit")
38 aggregate_df.orderBy("year", "quarter").show()
39 dyf = DynamicFrame.fromDF(aggregate_df, glueContext, "dynamic_frame")
40 dyf.show()
41 redshift_output = glueContext.write_dynamic_frame.from_jdbc_conf(
42     frame=dyf,
43     catalog_connection="redshift-demo-connection",
44     connection_options={"dbtable": "public.Regionalsales", "database": "dev"},
45     redshift_tmp_dir = "s3://aws-glue-assets-131795324646-us-east-1/temporary/",
46     transformation_ctx = "redshift_output"
47 )
48
49
50 job.commit()
```

5 AWS Redshift Data Warehouse

Here we see the result of our working Data Pipeline. All data are processed (transformed, aggregated) and then written to AWS Redshift Data Warehouse. Now further data analytics for ex Business Intelligence, Machine Learning is possible in this basis.

The screenshot shows the AWS Redshift query editor v2 interface. The left sidebar contains navigation options: Editor, Queries, Notebooks, Charts, History, and Scheduled queries. The main panel displays a tree view of resources under 'etl-redshift-cluster', including 'awsdatacatalog', 'dev', 'public', 'Tables', 'Views', 'Functions', 'Stored procedures', and 'sample_data_dev'. The 'regional_sales' table is selected, showing its schema with fields: region, country, year, quarter, total_revenue, total_cost, and total_profit. The right panel shows the SQL query 'select count(*) from regional_sales;' and the result '3317'.

Field	Type	NL	CMP	
A	region	character varying(65535)	NULL	lzo
A	country	character varying(65535)	NULL	lzo
#	year	integer	NULL	az64
#	quarter	integer	NULL	az64
#	total_revenue	double precision	NULL	none
#	total_cost	double precision	NULL	none
#	total_profit	double precision	NULL	none

only showing top 20 rows

3317

Amount is the same as in interactive Jupyter Notebook

aws

Services

Suche

[Alt+S]

Editor

Queries

Notebooks

Charts

History

Scheduled queries

Redshift query editor v2

CreateLoad data

Filter resources

etl-redshift-cluster

awsdatacatalog

dev

public

Tables

regionalsales

Views

Functions

Stored procedures

sample_data_dev

regional

Field

Type

NL

C

Aregion

character varying(65535)

NULL

lzc

Acountry

character varying(65535)

NULL

lzc

#year

integer

NULL

az

#quarter

integer

NULL

az

#total_revenue

double precision

NULL

nc

#total_cost

double precision

NULL

nc

#total_profit

double precision

NULL

nc

Untitled 1

RunLimit 100ExplainIsolated sessionetl-redshift-cl...dev

1select * from regional

Result 1 (100)

region	country	year	quar...	total_revenue	total_cost	total_profit
Australia and Oceania	Australia	2012	1	1995325.22	1161763.1	833562.12
Sub-Saharan Africa	Mauritius	2012	4	1247896.9	739215.19	508681.70999999996
Sub-Saharan Africa	Equatorial Guinea	2016	3	1559994.38	929873.66999999999	630120.71000000001
Sub-Saharan Africa	Republic of the Congo	2013	3	411813.42	262990.56	148822.86
Middle East and North Afr...	Somalia	2015	1	172433.3	115524.86	56908.44
Asia	India	2012	3	6410028.5200000005	4812548.66	1597479.85999999999
Asia	Bangladesh	2014	2	1718066.8299999998	1262595.38000000001	455471.45
Middle East and North Afr...	Egypt	2017	2	120385.62	76880.16	43505.46
Sub-Saharan Africa	Equatorial Guinea	2015	4	655964.98	454833.42	201131.56
Asia	Taiwan	2015	3	2455620.88	1281893.93	1173726.95
Sub-Saharan Africa	Rwanda	2013	2	353809.17	245324.43	108484.74
Middle East and North Afr...	Tunisia	2012	3	994229.44	326072.32	668157.12
Australia and Oceania	Nauru	2013	1	6397795.3000000001	5085001.86	1312793.44
Sub-Saharan Africa	Comoros	2010	3	146193.45	97944.99	48248.46
Australia and Oceania	Samoa	2016	2	1588689.45	1179858.4300000002	408831.02
Asia	Uzbekistan	2014	1	4081117.33	3129775.62	951341.71
Sub-Saharan Africa	Ethiopia	2016	3	5036301.2299999995	3984452.3899999997	1051848.84
Central America and the ...	The Bahamas	2014	2	604878.37	363025.35000000003	241853.02
Middle East and North Afr...	Pakistan	2013	3	2005224.4	1252244.1	752980.3
Europe	Hungary	2014	4	636268.05	441175.95	195092.1
Asia	Singapore	2011	4	551410.23	476649.83	74760.4
Middle East and North Afr...	Afghanistan	2012	4	2593777.2399999998	1624544.2799999998	969232.96
Asia	Tajikistan	2011	1	284483.1	161963.13	122519.97
Middle East and North Afr...	Qatar	2016	2	943661.4	805687.79	137973.61000000002
Asia	Cambodia	2010	1	5682062.3199999999	4130533.34000000003	1551528.98000000002
Europe	Netherlands	2010	4	1545804.96	1336224.16	209580.8
Middle East and North Afr...	Somalia	2013	1	691563.4	393723.82	297839.58
Central America and the ...	Dominican Republic	2013	1	2561551.31	1621054.89000000001	940496.41999999999
Middle East and North Afr...	Iran	2015	2	5858723.09	4405768.18	1452954.91
Asia	Malaysia	2012	2	1533287.8	872937.94	660349.86
Europe	Belgium	2012	4	4291025.49	3278650.79	1012374.7
Central America and the ...	El Salvador	2017	2	775411.56	495190.08	280221.48
Central America and the ...	Saint Kitts and Nevis	2015	1	1995539.7	1724983.7	270556