

# AWS POC 4 - Load Government Debt Total % GDP via a custom Python Job from a local host

## Overview



## Process Flow

1. Push the formatted CSV file to the staging S3 bucket
2. Run the custom AWS Glue Job (Python)
3. Save the result to the World Bank Data S3 bucket
4. Create an AWS Glue Table over the S3 Bucket and query in Athena

## Glue Job Python Code

```
1 import sys
2 from awsglue.transforms import *
3 from awsglue.utils import getResolvedOptions
4 from pyspark.context import SparkContext
5 from awsglue.context import GlueContext
6 from awsglue.job import Job
7
8 args = getResolvedOptions(sys.argv, ['JOB_NAME'])
9 sc = SparkContext()
10 glueContext = GlueContext(sc)
11 spark = glueContext.spark_session
12 job = Job(glueContext)
13 job.init(args['JOB_NAME'], args)
14 print("The args: ", args)
15
16
17 # Script generated for node AWS Glue Data Catalog
18 AWSGlueDataCatalog_node1708237359808 = glueContext.create_dynamic_frame.from_catalog(database="aws-poc2-db", tab
19
20 # Script generated for node Remove Prior to 1990
21 RemovePriorTo1990_node1708238527332 = ApplyMapping.apply(frame=AWSGlueDataCatalog_node1708237359808, mappings=[(
22
23 # Script generated for node Amazon S3
24 AmazonS3_node1708238624148 = glueContext.write_dynamic_frame.from_options(frame=RemovePriorTo1990_node1708238527
25
26 job.commit()
```

[marthinusswart/aws\\_glue\\_jobs\\_poc](#)

Created by marthinusswart • Updated 3 minutes ago

local running glue jobs

# AWS Glue Table

## SQL DDL

```

1 CREATE EXTERNAL TABLE `world_data_indicators_parquet_custom_python`(
2   `country_name` string COMMENT '',
3   `country_code` string COMMENT '',
4   `indicator_name` string COMMENT '',
5   `indicator_code` string COMMENT '',
6   `1991` decimal COMMENT '',
7   `1992` decimal COMMENT '',
8   `1993` decimal COMMENT '',
9   `1994` decimal COMMENT '',
10  `1995` decimal COMMENT '',
11  `1996` decimal COMMENT '',
12  `1997` decimal COMMENT '',
13  `1998` decimal COMMENT '',
14  `1999` decimal COMMENT '',
15  `2000` decimal COMMENT '',
16  `2001` decimal COMMENT '',
17  `2002` decimal COMMENT '',
18  `2003` decimal COMMENT '',
19  `2004` decimal COMMENT '',
20  `2005` decimal COMMENT '',
21  `2006` decimal COMMENT '',
22  `2007` decimal COMMENT '',
23  `2008` decimal COMMENT '',
24  `2009` decimal COMMENT '',
25  `2010` decimal COMMENT '',
26  `2011` decimal COMMENT '',
27  `2012` decimal COMMENT '',
28  `2013` decimal COMMENT '',
29  `2014` decimal COMMENT '',
30  `2015` decimal COMMENT '',
31  `2016` decimal COMMENT '',
32  `2017` decimal COMMENT '',
33  `2018` decimal COMMENT '',
34  `2019` decimal COMMENT '',
35  `2020` decimal COMMENT '',
36  `2021` decimal COMMENT '',
37  `2022` decimal COMMENT '')
38 ROW FORMAT SERDE
39   'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
40 STORED AS INPUTFORMAT
41   'org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat'
42 OUTPUTFORMAT
43   'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat'
44 LOCATION
45   's3://poc-world-bank-data/GC.DOD.TOTL.GD.ZS_python_poc/'
46 TBLPROPERTIES (
47   'classification'='parquet')

```

## JSON Schema

```
1  [
2    {
3      "Name": "country_name",
4      "Type": "string",
5      "Comment": ""
6    },
7    {
8      "Name": "country_code",
9      "Type": "string",
10     "Comment": ""
11   },
12   {
13     "Name": "indicator_name",
14     "Type": "string",
15     "Comment": ""
16   },
17   {
18     "Name": "indicator_code",
19     "Type": "string",
20     "Comment": ""
21   },
22   {
23     "Name": "1991",
24     "Type": "decimal",
25     "Comment": ""
26   },
27   {
28     "Name": "1992",
29     "Type": "decimal",
30     "Comment": ""
31   },
32   {
33     "Name": "1993",
34     "Type": "decimal",
35     "Comment": ""
36   },
37   {
38     "Name": "1994",
39     "Type": "decimal",
40     "Comment": ""
41   },
42   {
43     "Name": "1995",
44     "Type": "decimal",
45     "Comment": ""
46   },
47   {
48     "Name": "1996",
49     "Type": "decimal",
50     "Comment": ""
51   },
52   {
53     "Name": "1997",
54     "Type": "decimal",
55     "Comment": ""
56   },
57   {
58     "Name": "1998",
```

```
59     "Type": "decimal",
60     "Comment": ""
61 },
62 {
63     "Name": "1999",
64     "Type": "decimal",
65     "Comment": ""
66 },
67 {
68     "Name": "2000",
69     "Type": "decimal",
70     "Comment": ""
71 },
72 {
73     "Name": "2001",
74     "Type": "decimal",
75     "Comment": ""
76 },
77 {
78     "Name": "2002",
79     "Type": "decimal",
80     "Comment": ""
81 },
82 {
83     "Name": "2003",
84     "Type": "decimal",
85     "Comment": ""
86 },
87 {
88     "Name": "2004",
89     "Type": "decimal",
90     "Comment": ""
91 },
92 {
93     "Name": "2005",
94     "Type": "decimal",
95     "Comment": ""
96 },
97 {
98     "Name": "2006",
99     "Type": "decimal",
100    "Comment": ""
101 },
102 {
103     "Name": "2007",
104     "Type": "decimal",
105     "Comment": ""
106 },
107 {
108     "Name": "2008",
109     "Type": "decimal",
110     "Comment": ""
111 },
112 {
113     "Name": "2009",
114     "Type": "decimal",
115     "Comment": ""
116 },
```

```
117 {
118     "Name": "2010",
119     "Type": "decimal",
120     "Comment": ""
121 },
122 {
123     "Name": "2011",
124     "Type": "decimal",
125     "Comment": ""
126 },
127 {
128     "Name": "2012",
129     "Type": "decimal",
130     "Comment": ""
131 },
132 {
133     "Name": "2013",
134     "Type": "decimal",
135     "Comment": ""
136 },
137 {
138     "Name": "2014",
139     "Type": "decimal",
140     "Comment": ""
141 },
142 {
143     "Name": "2015",
144     "Type": "decimal",
145     "Comment": ""
146 },
147 {
148     "Name": "2016",
149     "Type": "decimal",
150     "Comment": ""
151 },
152 {
153     "Name": "2017",
154     "Type": "decimal",
155     "Comment": ""
156 },
157 {
158     "Name": "2018",
159     "Type": "decimal",
160     "Comment": ""
161 },
162 {
163     "Name": "2019",
164     "Type": "decimal",
165     "Comment": ""
166 },
167 {
168     "Name": "2020",
169     "Type": "decimal",
170     "Comment": ""
171 },
172 {
173     "Name": "2021",
174     "Type": "decimal",
```

```

175     "Comment": ""
176 },
177 {
178     "Name": "2022",
179     "Type": "decimal",
180     "Comment": ""
181 }
182 ]

```

## Query Table

```

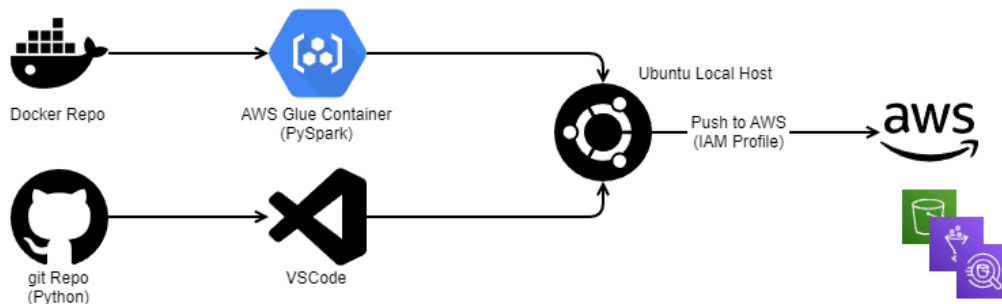
1 select * from world_data_indicators_parquet_custom_python;

```

## Result

#	country_name	country_code	indicator_name	indicator_code	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2
1	Aruba	ABW	Central government debt, total (% of GDP)	GC.DOD.TOTL.GD.ZS												
2	Africa Eastern and Southern	AFE	Central government debt, total (% of GDP)	GC.DOD.TOTL.GD.ZS												
3	Afghanistan	AFG	Central government debt, total (% of GDP)	GC.DOD.TOTL.GD.ZS												
4	Africa Western and Central	AFW	Central government debt, total (% of GDP)	GC.DOD.TOTL.GD.ZS												
5	Angola	AGO	Central government debt, total (% of GDP)	GC.DOD.TOTL.GD.ZS												
6	Albania	ALB	Central government debt, total (% of GDP)	GC.DOD.TOTL.GD.ZS					36	37	53	56				

## AWS Glue Docker Container



## Process Flow

1. Fetch AWS Glue Container from the public Docker Repo
2. Run the AWS Glue Container on a local Ubuntu host
3. Fetch the Python code from Git
4. Run the Python job using VSCode on Ubuntu
5. The Python job has full access to the AWS environment via the IAM profile setup to read and write to the S3 buckets