

Capstone



Pep Three

The S3 Musketeers

ETL Pipeline

Finn Mikkola, Justin Quinn, Alex Daughters




Scenario

- Global Corp Ltd. employs a diverse workforce in IT services around the world. They are starting a new initiative to help promote multicultural awareness among its staff. S3 Musketeer Solutions has been tasked with creating a cloud based service that will identify local holidays from around the world using the Public Holiday API

User Stories



- As a Global Corp Ltd. employee, I want to be able to easily identify holidays from around the world.
 - As a software developer at S3 Musketeer Solutions, I want to integrate the Public Holiday API into the cloud-based service effectively so that it accurately retrieves and displays local holidays from various countries.
- 

Overview and goal

Our goal was to architect and implement a data pipeline, handling data through stages such as extraction, transformation, loading (ETL), and analysis, simulating real-world challenges by sourcing and analyzing semi-structured data.



Local vs Cloud-Based

While doing this project locally would probably have been easier we determined that it would be best to go with a cloud-based pipeline mostly for the practice it would give us.



Our Task

01

Extract

Extract the raw data from various sources

03

Load

Load the data into a data warehouse for safe keeping

02

Transform

Transform the data into a clean consistent format

04

Analysis

Perform statistical analysis on the loaded data





01

The Data

Holidays from around the world!

Upcoming Holidays

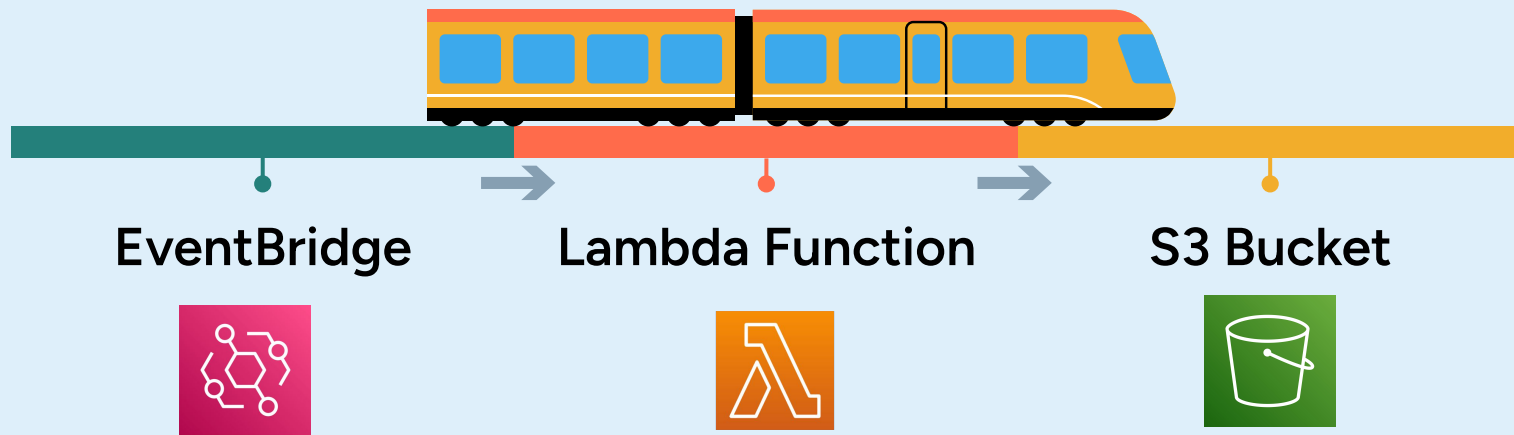
We accessed the nager.date api which gave us access to information about holidays from all over the world.

AWS Redshift did not like the arrays of strings so for simplicity's sake we dropped those

```
{
  "date": "2024-06-03",
  "localName": "King's Birthday",
  "name": "King's Birthday",
  "countryCode": "NZ",
  "fixed": false,
  "global": true,
  "counties": null,
  "launchYear": null,
  "types": [
    "Public"
  ]
},
```

| | |
|--------------------|---|
| date | string, The date of the holiday |
| localName | string, Local name |
| name | string, English name |
| countryCode | Text, ISO 3166-1 alpha-2 |
| fixed | Bool, Is this public holiday every year on the same date |
| global | Bool, Is this public holiday in every county (federal state) |
| counties | Array of strings, If it is not global you found here the Federal states (ISO-3166-2) |
| launchYear | Integer, The launch year of the public holiday |
| types | Array of strings, The types of the public holiday, several possible <ul style="list-style-type: none">• Public• Bank (Bank holiday, banks and offices are closed)• School (School holiday, schools are closed)• Authorities (Authorities are closed)• Optional (Majority of people take a day off)• Observance (Optional festivity, no paid day off) |

Obtaining the Data



EventBridge



Using Amazon EventBridge, we set up a scheduled event to invoke the lambda function everyday at a certain time.

Challenges

- Before I went to EventBridge I tried to use Amazon API Gateway to connect to the lambda but it seems that that way was a bit more confusing and may have not worked at all as it seems that API Gateway might be more for setting up your own endpoint instead of using a public one.
- Permissions were thankfully not too much an issue

Amazon EventBridge > Schedules > CapRule

CapRule

[Disable](#) [Edit](#) [Delete](#)

Schedule detail

| | | | |
|---------------------|---|------------------------------------|------------------------------------|
| Schedule name | Status | Schedule start time | Flexible time window |
| CapRule | Enabled | Feb 20, 2024, 16:00:00 (UTC-06:00) | - |
| Description | Schedule ARN | Schedule end time | Created date |
| - | arn:aws:scheduler:us-east- | Feb 24, 2024, 12:00:00 (UTC-06:00) | Feb 20, 2024, 14:30:35 (UTC-06:00) |
| Schedule group name | 1:891377269970:schedule/default/CapRule | Execution time zone | Last modified date |
| default | Action after completion | America/Chicago | Feb 20, 2024, 15:56:31 (UTC-06:00) |
| | NONE | | |

- Amazon EventBridge's timing matters and all events need to have enough time to set up in order for the event to trigger. This caused quite a bit of confusion as sometimes it would appear to not be working when in reality it just didn't have time to set up before it was supposed to trigger.

Lambda

Upon invocation from the scheduled event in EventBridge, Lambda would request the latest data from the api endpoint and save that as a .json into the s3 bucket as holidays.json



Challenges

We tried to import modules such as “responses” through the layers and also directly into the lambda files and nothing worked so we had to find a workaround.

The work around ended up being figuring out how to do the same thing but using a module that is already part of AWS Lambda. Instead we used “urllib3” to get the api endpoint information



S3 Bucket

In our architecture, data flows into the S3 bucket during the extraction phase. S3 serves as the initial landing point for our raw data.

The S3 bucket that we had set up would have a holidays.json that we put a smaller slice of data into to test that everything was working. It would be overwritten by the lambda everyday at the scheduled time.

Amazon S3 > Buckets > capstone-bucket-g2

capstone-bucket-g2 Info

Objects | Properties | Permissions | Metrics | Management | Access Points

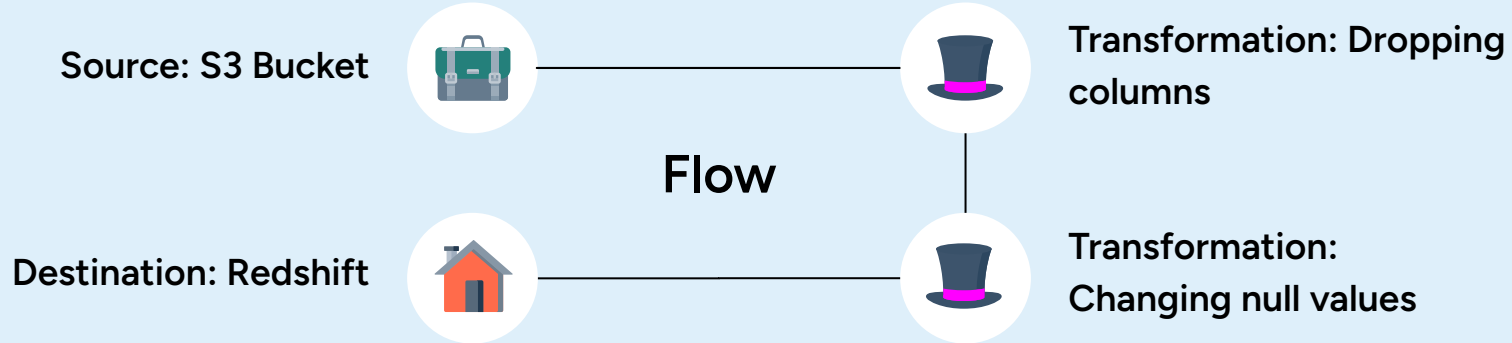
Objects (4) Info 🔄 📄 Copy S3 URI 📄 Copy URL 📄 Download 🔗 Open 🗑 Delete ⌵ Actions 📁 Create folder 📤 Upload

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

| <input type="checkbox"/> | Name | Type | Last modified | Size | Storage class |
|--------------------------|------------------------|--------|---|----------|---------------|
| <input type="checkbox"/> | all_holidays_2024.csv | csv | February 21, 2024, 10:25:23 (UTC-06:00) | 115.0 KB | Standard |
| <input type="checkbox"/> | all_holidays_2024.json | json | February 20, 2024, 10:47:05 (UTC-06:00) | 572.3 KB | Standard |
| <input type="checkbox"/> | holidays.json | json | February 21, 2024, 16:03:13 (UTC-06:00) | 2.0 KB | Standard |
| <input type="checkbox"/> | imports/ | Folder | - | - | - |

AWS Glue

Amazon's ETL Service. A wonderful and albeit slightly expensive service that can facilitate extracting, transforming, and loading it all in one place



Redshift

Storing The Data

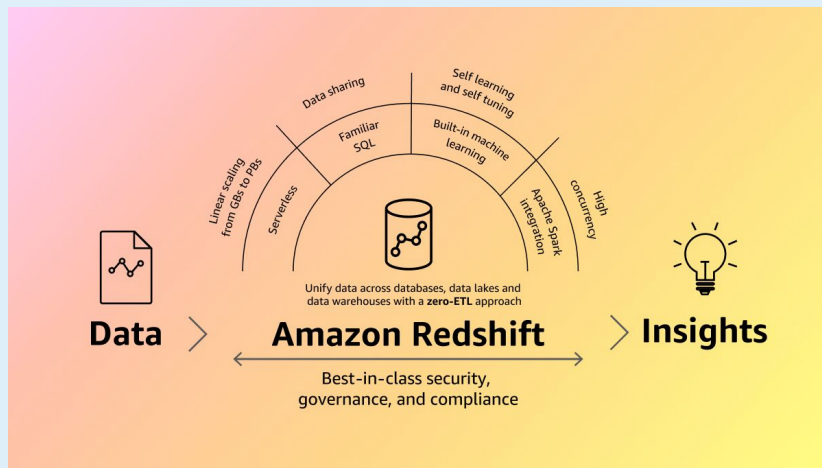
We used 2 tables to store the data, one for upcoming holidays, one for all holidays

Querying the Data

Now that the data is in the warehouse we can analyze it!

Showing Results

We can connect to Redshift via our local machines, or use Redshift's query editor v2



Amazon
Redshift

Analysis Questions



Which countries have the most holidays?



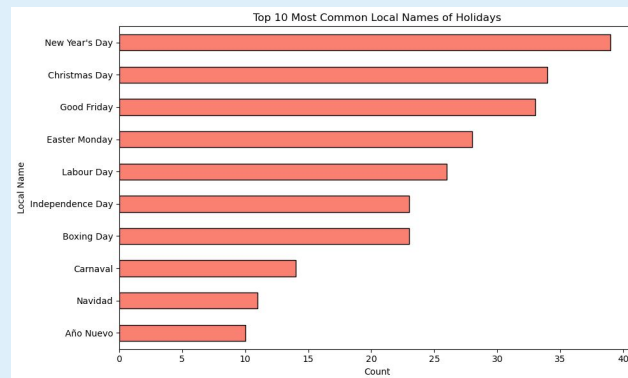
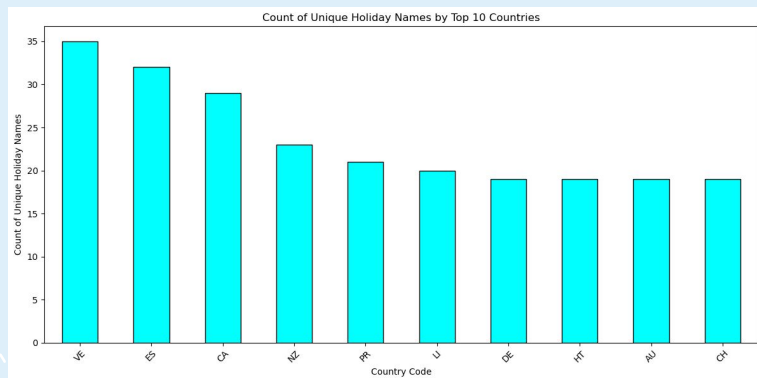
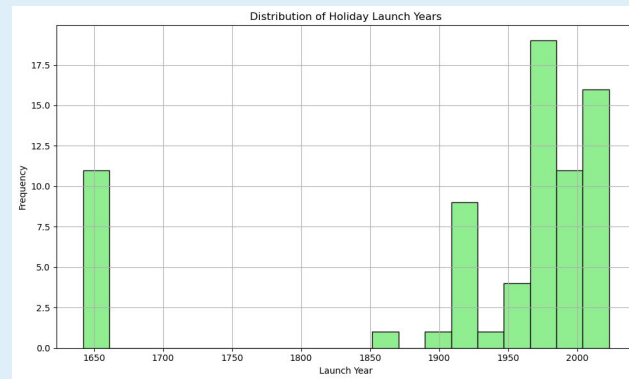
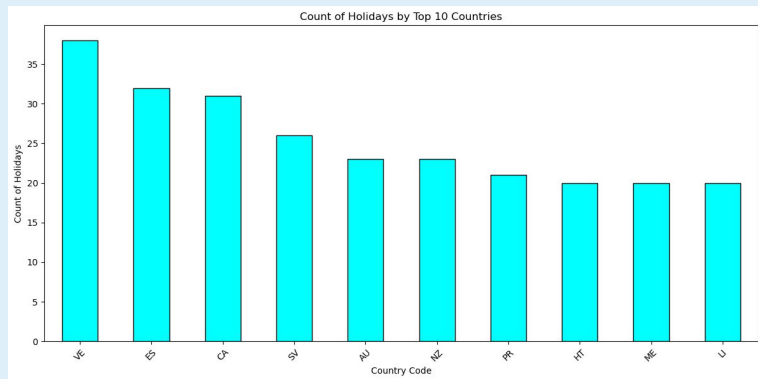
What does the distribution of launch years look like?



What are the most common local holidays across all countries?



Analysis





Analysis of Variance

Analysis of Variance (ANOVA) test:

- The F-statistic value indicates the ratio of the variability in the number of fixed holidays among countries compared to the variability within each country. ~4.92 suggests that there may be significant differences in the means of the fixed holidays among the countries being compared.
- The p-value represents the probability of observing the data under the assumption that the null hypothesis is true. 1.73e-46 suggests that the observed differences in the means of the groups are unlikely to be due to random chance alone.

```
from scipy.stats import f_oneway
```

```
# 'fixed' column contains the number of fixed holidays for each country  
#Regression Analysis...]
```

```
# Perform ANOVA test
```

```
f_statistic, p_value = f_oneway(df[df['countryCode'] == 'AD']['fixed'],  
                                df[df['countryCode'] == 'AL']['fixed'],  
                                df[df['countryCode'] == 'AR']['fixed'],  
                                df[df['countryCode'] == 'AT']['fixed'],  
                                df[df['countryCode'] == 'AU']['fixed'],  
                                df[df['countryCode'] == 'AX']['fixed'],  
                                df[df['countryCode'] == 'BA']['fixed'],  
                                df[df['countryCode'] == 'BB']['fixed'],  
                                df[df['countryCode'] == 'BE']['fixed'],  
                                df[df['countryCode'] == 'BG']['fixed'],  
                                df[df['countryCode'] == 'BJ']['fixed'],  
                                df[df['countryCode'] == 'BO']['fixed'],
```

```
                                df[df['countryCode'] == 'VA']['fixed'],  
                                df[df['countryCode'] == 'VE']['fixed'],  
                                df[df['countryCode'] == 'VN']['fixed'],  
                                df[df['countryCode'] == 'ZA']['fixed'],  
                                df[df['countryCode'] == 'ZW']['fixed'])
```

```
# Print results
```

```
print("F-statistic:", f_statistic)
```

```
print("p-value:", p_value)
```

```
# Interpret results
```

```
alpha = 0.05
```

```
if p_value < alpha:
```

```
    print("Reject the null hypothesis. There is a significant difference in the average number of fixed holidays between countries.")
```


```
else:
```

```
    print("Fail to reject the null hypothesis. There is no significant difference.")
```

F-statistic: 4.917863280287758

p-value: 1.725069229210718e-46

Reject the null hypothesis. There is a significant difference in the average number of fixed holidays between countries.



Chi-Square Test

- Since the p-value, $\sim 3.52e-131$, is significantly smaller than the chosen significance level ($\alpha = 0.05$), the distribution of holidays across different types and countries is unlikely to be due to random chance alone.

```
# Flatten lists in 'types' column
df['types'] = df['types'].apply(lambda x: ', '.join(x) if isinstance(x, list) else x)

# Perform chi-square test
contingency_table = pd.crosstab(df['types'], df['countryCode'])
chi2_statistic, p_value, _, _ = chi2_contingency(contingency_table)

# Print results
print("Chi-square statistic:", chi2_statistic)
print("p-value:", p_value)

# Interpret results
alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis. There is a significant association between types of")
else:
    print("Fail to reject the null hypothesis. There is no significant association.")
```



```
Chi-square statistic: 1965.3425532967663
p-value: 3.5185817497425714e-131
Reject the null hypothesis. There is a significant association between types of holidays and countries.
```


Challenges:

- Remote Access to Redshift:
- The official Amazon Documentation referred to using a Redshift Cluster for remote access. However, it was no longer in Free Tier, but thousands of dollars a month, depending on the configuration.
- Amazon did offer a \$300 credit, but it was decided that this would be an uneconomical decision.
- Since the parameters of the project did not technically require remote retrieval of the data prior to analysis, the decision was made to not move forward with this feature, even though the infrastructure was already set up in S3, Glue, and the VPC.
- Lesson Learned: Although this did not require significant additional cost, it would be prudent in the future to price out all pieces of a cloud architecture before project start so as not to waste time and money.
- Chi-Square Test:
- There were some issues with the data types (particularly, the 'type' column). This echoes the issues earlier in the project moving the data into Redshift.
- In this case, a lambda function to flatten the data was used, rather than change the parameters of the test.

```
# Flatten lists in 'types' column
df['types'] = df['types'].apply(lambda x: ', '.join(x) if isinstance(x, list) else x)
```

Conclusion

We were able to create this pipeline rather fast and where we did have problems arise, we were able to find solutions and workarounds

Having a clear idea from the start of what the architecture and data flow would look like aided in the speedy development.

We were able to demonstrate our ability to work within a cloud-based environment with different types of data and sources



Any Questions?

Thank you for listening!