Final Project 2 Report

Group: **B**

Participants: **Atera Alam & Priska Mohunsingh**

Project Title: **Cinematic Climate: How Weather Influences Movie Release Success**

Milestone 5:

Movie Release and Box Office Data

- The Movie Database (TMDb): This source provides extensive metadata on films, including release dates, genres, and other attributes, accessible through an open API for non-commercial academic projects. TMDb's data will serve as a foundational dataset, allowing us to analyze genre and release timing with consistency.

- Box Office Mojo by IMDb Pro: Box Office Mojo provides downloadable box office performance reports directly from their website. Although the data isn't as flexible as an API, these reports include historical revenue data segmented by genre, location, and release timing.

Weather Data by Location and Date

- NOAA Climate Data Online (CDO): Historical daily weather data from NOAA, covering temperature, precipitation, and weather conditions across the U.S., will allow us to track environmental variables across release periods. NOAA data will facilitate a location-based examination of weather effects on attendance.

Audience Engagement & Trend Data

There is a IMDb ratings and reviews dataset on Kaggle for download. This includes detailed information about user ratings and, in some cases, reviews by users and critics.

Event and Holiday Data

Holiday and Event Data: Public holiday datasets allow us to consider whether attendance spikes around holiday weekends influence weather-related attendance trends, providing insight into additional temporal and social factors.

Analysis Dimensions

1. Weather Conditions
    - **Temperature**: We'll segment temperature data on release dates into categorized ranges (e.g., below freezing, mild, hot) to examine whether temperature extremes correlate with box office performance.
    - **Precipitation**: By capturing both precipitation type and intensity on release days, we can explore if adverse weather (e.g., rain, snow) suppresses or encourages attendance across various regions.

Final Project 2 Report

    - **Seasonal Influence:** A season-based analysis will allow us to identify patterns that may emerge due to seasonal weather variations, such as winter storms or summer heat, and their potential impact on movie attendance.
2. Movie-Specific Factors
    - **Genre**: We plan to classify movies by genre (e.g., horror, comedy, action) to see if weather conditions correlate with varying performance trends. For instance, we can assess whether certain genres appeal more under certain environmental conditions.
    - **Release Timing**: Release timing is critical; we'll differentiate between weekday, weekend, and holiday releases to isolate potential differences in attendance and genre preferences.
    - **Box Office Metrics**: We will use opening weekend revenue and cumulative box office gross as primary indicators of success, providing quantitative benchmarks to measure against weather conditions.
3. Location-Based Analysis
    - **Regional Weather Patterns**: We'll examine box office performance across major urban centers, using regional weather data to explore if local weather patterns differentially affect movie attendance.
    - **Demographic Data:** Where possible, we will incorporate demographic data by region to further assess if specific audience groups may be more affected by weather conditions in their decision to attend.
4. Audience Engagement
    - **Google Trends Interest:** By tracking search interest leading up to release dates, we hope to determine if heightened audience anticipation correlates with release-day weather, potentially influencing overall turnout.
    - **IMDb Ratings and Reviews:** Post-release ratings and reviews will allow us to explore correlations between release-day weather and audience sentiment, particularly if adverse weather influences viewing perception.

Project Intentions & Summary

Through this multidimensional approach, we aim to understand the influence of weather conditions on box office performance and audience engagement, examining factors such as genre resilience, seasonal effects, and regional behaviors. By analyzing these dimensions in tandem, our project seeks to provide a comprehensive assessment of weather as a contributing factor to movie success. This study could potentially offer insights for studios in strategizing release dates, adjusting marketing efforts, and anticipating audience behaviors under varied conditions.

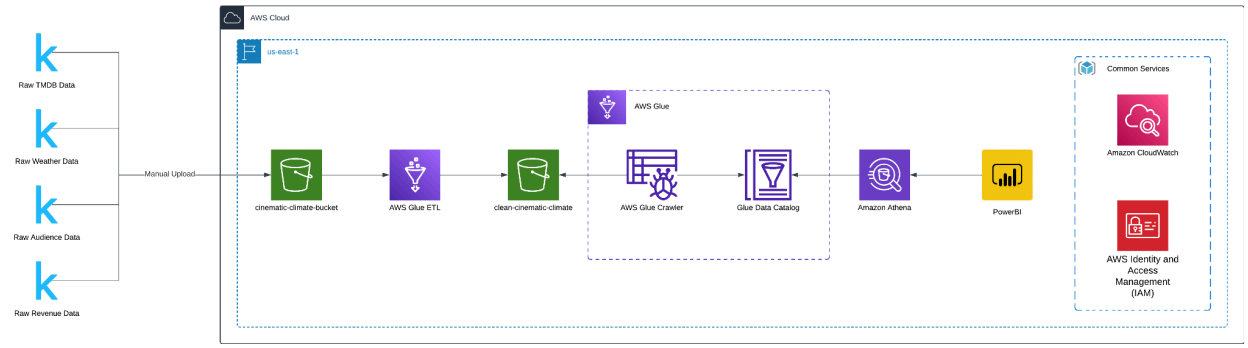
Final Project 2 Report

Milestone 6:

I. Overview

Architecture Diagram

In this architecture diagram, we have outlined each step that needs to be taken to build a data pipeline. At first, we uploaded our data that was retrieved from Kaggle. Then, these datasets were dropped into an S3 bucket.



The pipeline will include the following stages:

1. Extract and Load: Storing Raw Data in S3
    - Raw datasets (e.g., TMDb metadata, weather, revenue, and audience data) are uploaded to S3 in their original format.
    - These datasets are stored in specific folders for organization: Raw-TMDB, Raw-NOAA, Raw-Revenue, and Raw-Audience-Data.
2. Crawling: Cataloging Data with Glue
    - AWS Glue Crawlers scan the raw data in S3 and create tables in the Glue Data Catalog.
    - This makes the raw data queryable and sets up the foundation for transformation.
3. Transformation: Cleaning Data with Glue ETL and Lambda
    - Glue ETL:
        - Processes raw data into a cleaner format (e.g., merging datasets, filtering, and handling missing values).
        - Writes the intermediate cleaned data to the clean-cinematic-climate S3 bucket.
    - Lambda:
        - Automatically triggered after Glue ETL writes data to the cleaned S3 bucket.
        - Performs additional validation and final transformations.
4. Load: Storing Cleaned Data in S3
    - The final cleaned and transformed dataset (e.g., final_movies_dataset.csv) is saved in the clean-cinematic-climate S3 bucket.
    - This dataset combines information from all sources (e.g., movie metadata, weather, revenue, and audience sentiment).
5. Query and Analysis: Using Athena
    - Purpose: Athena is used to query the cleaned data directly from S3.
    - Steps:

Final Project 2 Report

- Query the final_movies_dataset.csv using SQL via Athena.
- Analyze relationships like weather conditions and movie performance or audience sentiment and popularity.
- Save query results back to S3 or connect Athena to QuickSight for dashboard creation.

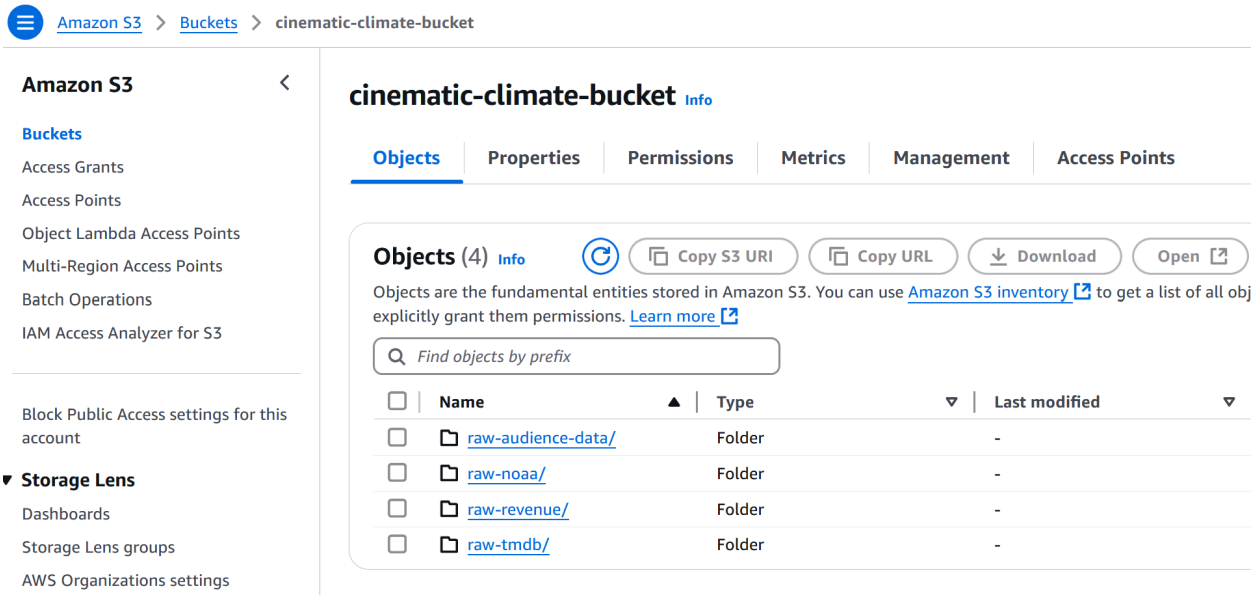
Data Sources

Four types of files and data sources are used for this data pipeline. The first data source is the Movie metadata (TMDb API) which is accessed using Python and a provided API key from TMDB. The second source is the weather data (NOAA CSV files), which was manually uploaded to Amazon S3. The third data source is audience data from Kaggle. The fourth data source is revenue data from Kaggle as well.

II. ELT Process

1. Set up S3 bucket for Raw Data Storage

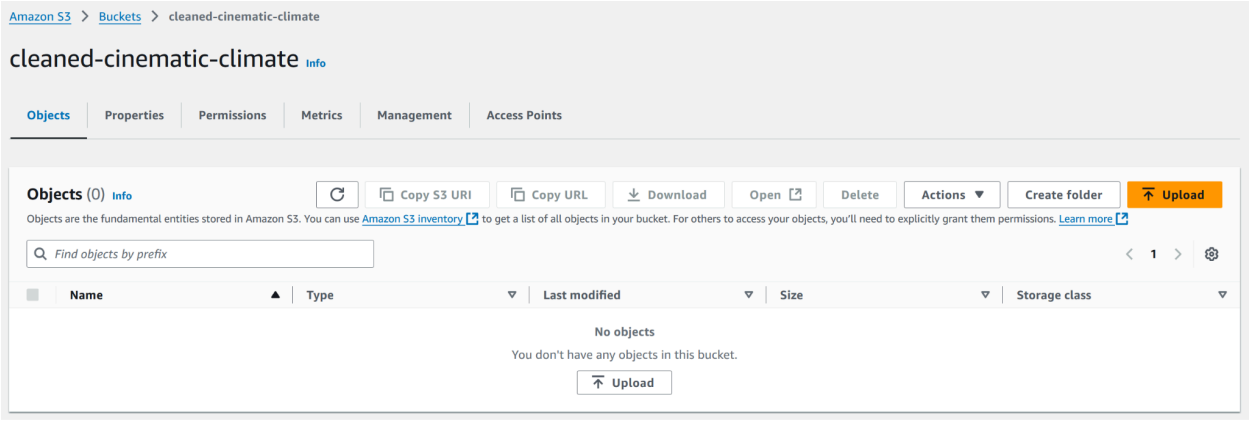
We first created an S3 bucket in **us-east-1** called cinematic-climate-bucket, and created 3 folders within it, namely, Raw-TMDB, Raw-NOAA, Raw-Revenue, and Raw-Audience-Data.



2. Set up S3 Bucket for Cleaned/Transformed Data Storage

A bucket for cleaned data was created so that transformed and matched data between all datasets exists for coherent storytelling.

Final Project 2 Report



3. Load Appropriate Data into Each Raw Folder

For this part of the project, we systematically processed and uploaded raw datasets sourced from Kaggle into the designated folders within our S3 bucket (cinematic-climate-bucket). These datasets represent key components of the project, each contributing to our analysis of movie metadata, weather conditions, revenue trends, and audience behavior. Below is a detailed breakdown of how the data was added to the respective folders:

1. Raw TMDB Folder:
    - Dataset: movies.csv (sourced from Kaggle's TMDB Data.
    - Purpose: To store movie metadata such as titles, release dates, and popularity scores.
2. Raw NOAA Folder:
    - Dataset: global-weather-repository.csv (from Kaggle, representing global weather summaries).
    - Purpose: To correlate weather conditions with movie release dates and their performance.
3. Raw Revenue Folder:
    - Dataset: revenue.csv (sourced from Kaggle and curated manually for completeness).
    - Purpose: To analyze box office performance and revenue trends for the movies in the TMDb dataset.
4. Raw Audience Data Folder:
    - Dataset: rotten-tomatoes-movies.csv (sourced from Kaggle's Rotten Tomatoes).
    - Purpose: To measure audience sentiment, ratings, and review counts for the movies in the dataset.

By organizing and uploading these raw datasets into the designated folders, we established a structured foundation for subsequent data transformations and analysis. This organization ensures seamless integration with serverless applications in AWS.

Final Project 2 Report

4. Streamlining Data Processing with AWS Glue and Lambda

Our next step was to set up a Glue ETL job. This step was critical for processing and transforming the raw data stored in S3 into a format suitable for further analysis. The Glue ETL job reads the raw data, processes and organizes it, and writes the cleaned data into a separate S3 bucket designated for sanitized datasets. This, in turn, triggers a Lambda function, which performs additional cleaning and refinement tasks to ensure the data is ready for analysis. A Glue Crawler then scans through the sanitized data and updates the Glue Data Catalog, making the transformed datasets readily available for querying.

By centralizing and organizing the data through this workflow, we enable efficient querying and joining of data from different sources—such as TMDb and NOAA. This structured approach lays the groundwork for detailed analysis, such as studying correlations between weather conditions and movie performance. Glue provides the scalability and automation necessary to handle increasing data volume and complexity, while the integration with Lambda enhances the cleaning process, ensuring our ELT pipeline remains robust and efficient.

Glue Job:

```
import sys
from pyspark.context import SparkContext
from pyspark.sql import functions as F
from pyspark.sql.types import StringType, IntegerType
from awsglue.context import GlueContext
from awsglue.job import Job
from awsglue.utils import getResolvedOptions

# Initialize Glue context and job
args = getResolvedOptions(sys.argv, ['JOB_NAME'])
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args['JOB_NAME'], args)

# Input and output S3 paths
RAW_BUCKET = "s3://cinematic-climate-bucket"
CLEAN_BUCKET = "s3://cleaned-cinematic-climate"
RAW_FILES = {
      "tmdb": f"{RAW_BUCKET}/raw-tmdb/movies.csv",
      "weather": f"{RAW_BUCKET}/raw-noaa/global-weather-repository.csv",
      "audience":
f"{RAW_BUCKET}/raw-audience-data/rotten-tomatoes-movies.csv",
```

Final Project 2 Report

```
      "revenue": f"{RAW_BUCKET}/raw-revenue/revenue.csv"
}
OUTPUT_FILE = f"{CLEAN_BUCKET}/final-movies-dataset.csv"

# Load datasets with proper encoding and parsing options
read_options = {
      "header": True,
      "inferSchema": True,
      "encoding": "UTF-8",  # Ensure correct handling of special characters
      "multiLine": True,
      "quote": '"',
      "escape": '"'
}

tmdb_df = spark.read.csv(RAW_FILES["tmdb"], **read_options)
weather_df = spark.read.csv(RAW_FILES["weather"], **read_options)
audience_df = spark.read.csv(RAW_FILES["audience"], **read_options)
revenue_df = spark.read.csv(RAW_FILES["revenue"], **read_options)

# Helper function to clean excessive quotes in string columns
from pyspark.sql.functions import regexp_replace, col

def clean_excessive_quotes(df):
      string_columns = [field.name for field in df.schema.fields if
isinstance(field.dataType, StringType)]
      for column in string_columns:
      # Remove leading and trailing quotes (if they are not part of the
actual data)
      df = df.withColumn(column, reg12exp_replace(col(column),
r'^"+(.*?[^"])"*$', r'\1'))
      # Replace multiple consecutive quotes inside the field with a single
quote
      df = df.withColumn(column, regexp_replace(col(column), r'""+', '"'))
      return df

# Apply the cleaning function to all datasets
tmdb_df = clean_excessive_quotes(tmdb_df)
weather_df = clean_excessive_quotes(weather_df)
audience_df = clean_excessive_quotes(audience_df)
revenue_df = clean_excessive_quotes(revenue_df)
```

Final Project 2 Report

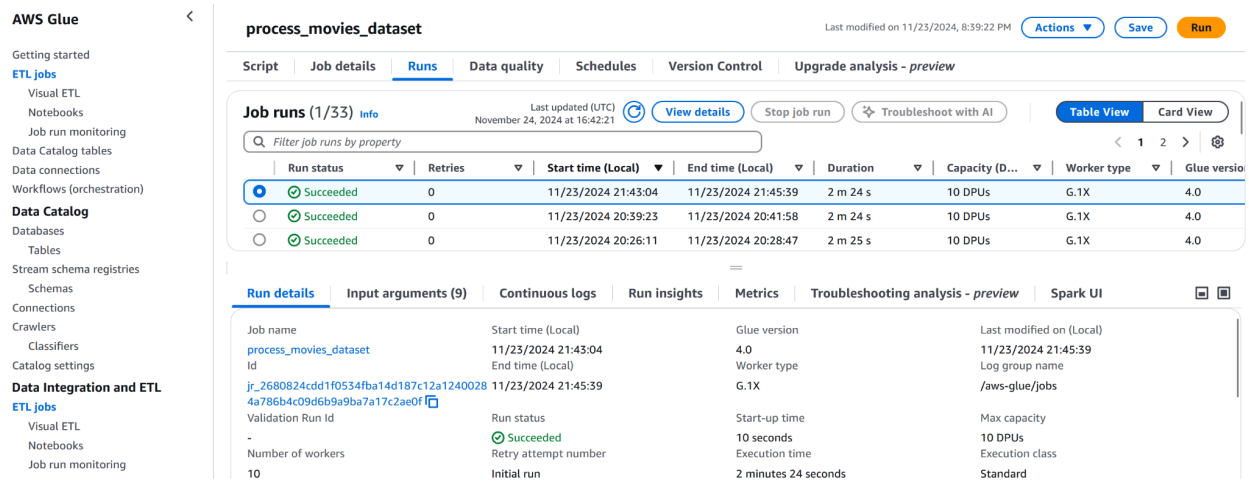
```
# Rename 'title' columns for consistency
tmdb_df = tmdb_df.withColumnRenamed('title', 'tmdb_title')
revenue_df = revenue_df.withColumnRenamed('title', 'revenue_title')
audience_df = audience_df.withColumnRenamed('movie_title',
'audience_title')

# Parse date columns into proper date format
tmdb_df = tmdb_df.withColumn('release_date', F.to_date('release_date',
'yyyy-MM-dd'))
revenue_df = revenue_df.withColumn('release_date',
F.to_date('release_date', 'yyyy-MM-dd'))

# Parse 'in_theaters_date' in 'audience_df' to 'release_date'
audience_df = audience_df.withColumn('release_date',
F.to_date('in_theaters_date', 'MMM d, yyyy'))

# Extract date from 'last_updated' in weather_df
weather_df = weather_df.withColumn('release_date',
F.to_date(F.col('last_updated').substr(1, 10), 'yyyy-MM-dd'))

# Ensure date columns are not null
tmdb_df = tmdb_df.filter(F.col("release_date").isNotNull())
weather_df = weather_df.filter(F.col("release_date").isNotNull())
revenue_df = revenue_df.filter(F.col("release_date").isNotNull())
audience_df = audience_df.filter(F.col("release_date").isNotNull())

# Select necessary columns to avoid duplicates
tmdb_df = tmdb_df.select(
      'tmdb_title', 'release_date', 'genres', 'original_language', 'id',
'overview',
      'popularity', 'budget', 'revenue', 'runtime', 'status', 'tagline',
'vote_average', 'vote_count'
)

# Rename 'genres' to 'movie_genres'
tmdb_df = tmdb_df.withColumnRenamed('genres', 'movie_genres')

weather_columns = ['release_date', 'temperature_celsius', 'humidity',
'wind_mph', 'condition_text']
weather_df = weather_df.select(weather_columns)
```

Final Project 2 Report

```
revenue_columns = ['revenue_title', 'release_date', 'adult', 'homepage',
'imdb_id',
                   'original_title', 'production_countries',
'spoken_languages']
revenue_df = revenue_df.select(revenue_columns)

audience_columns = ['audience_title', 'release_date', 'movie_info',
'critics_consensus', 'rating',
                    'genre', 'directors', 'writers', 'cast',
'runtime_in_minutes',
                    'studio_name', 'tomatometer_status',
'tomatometer_rating',
                    'tomatometer_count', 'audience_rating', 'audience_count']
audience_df = audience_df.select(audience_columns)

# Perform joins
merged_weather = tmdb_df.join(weather_df, on='release_date', how='left')

merged_revenue = merged_weather.join(
      revenue_df,
      (merged_weather.tmdb_title == revenue_df.revenue_title) &
(merged_weather.release_date == revenue_df.release_date),
      how='left'
).drop(revenue_df.revenue_title).drop(revenue_df.release_date)

final_df = merged_revenue.join(
      audience_df,
      (merged_revenue.tmdb_title == audience_df.audience_title) &
(merged_revenue.release_date == audience_df.release_date),
      how='left'
).drop(audience_df.audience_title).drop(audience_df.release_date)

# Remove duplicate columns if any
final_df = final_df.select(*{col for col in final_df.columns})

# Ensure 'id' is formatted as integer without decimal point
final_df = final_df.withColumn('id', final_df['id'].cast(IntegerType()))

# Clean up quotes in the final DataFrame
final_df = clean_excessive_quotes(final_df)
```

Final Project 2 Report

```
# Write to S3 with proper quoting and escaping
final_df.write.csv(
      OUTPUT_FILE,
      mode="overwrite",
      header=True,
      quote='"',
      escape='"',  # Set escape character to double quote
      sep=","
)

# Commit job
job.commit()
```



Lambda Function:

```
import boto3
import csv
import io

s3 = boto3.client('s3')

# Input and output S3 bucket and prefixes
CLEANED_BUCKET = 'cleaned-cinematic-climate'
INPUT_PREFIX = 'final-movies-dataset.csv/'
OUTPUT_PREFIX = 'final-movies-dataset-cleaned.csv/'

def sanitize_row(row, fieldnames):
```

Final Project 2 Report

```
    """Sanitize a row by cleaning strings and ensuring fields match
fieldnames."""
    sanitized_row = {}
    for key in fieldnames:  # Process only expected fieldnames
        value = row.get(key, "")  # Default to empty string if key is
missing
        if value is not None and isinstance(value, str):
            # Remove leading/trailing quotes and escape internal quotes
            sanitized_value = value.strip().replace('"', '"')
            sanitized_row[key] = sanitized_value  # Keep as is (no
enclosing quotes added here)
        else:
            sanitized_row[key] = value  # Keep other data types as is
    return sanitized_row

def lambda_handler(event, context):
    try:
        # List all part files in the input directory
        response = s3.list_objects_v2(Bucket=CLEANED_BUCKET,
Prefix=INPUT_PREFIX)
        if 'Contents' not in response:
            raise Exception(f"No files found in
{CLEANED_BUCKET}/{INPUT_PREFIX}")

        for obj in response['Contents']:
            key = obj['Key']
            if key.endswith('/'):  # Skip directory markers
                continue

            # Read the part file from the input prefix
            response = s3.get_object(Bucket=CLEANED_BUCKET, Key=key)
            data = response['Body'].read().decode('utf-8')

            # Sanitize the CSV
            input_csv = io.StringIO(data)
            reader = csv.DictReader(input_csv)
            fieldnames = reader.fieldnames  # Capture fieldnames from the
input CSV

            output_csv = io.StringIO()
            writer = csv.DictWriter(
```

Final Project 2 Report

```
                output_csv,
                fieldnames=fieldnames,
                quoting=csv.QUOTE_MINIMAL,  # Quote only fields with
special characters
                quotechar='"',              # Use standard double quotes
                escapechar='\\'             # Escape special characters
            )

            # Write the header without quotes
            output_csv.write(",".join(fieldnames) + "\n")

            # Write the sanitized rows
            for row in reader:
                sanitized_row = sanitize_row(row, fieldnames)
                writer.writerow(sanitized_row)

            # Write the sanitized part file to the output prefix
            sanitized_key = key.replace(INPUT_PREFIX, OUTPUT_PREFIX)
            s3.put_object(Bucket=CLEANED_BUCKET, Key=sanitized_key,
Body=output_csv.getvalue())

            print(f"Sanitized file written to
s3://{CLEANED_BUCKET}/{sanitized_key}")

        return {
            'statusCode': 200,
            'body': f"Sanitization complete. Files written to
{CLEANED_BUCKET}/{OUTPUT_PREFIX}"
        }
    except Exception as e:
        print(f"Error: {str(e)}")
        return {
            'statusCode': 500,
            'body': f"Error occurred: {str(e)}"
        }
```

5. Future Directions: Leveraging Athena and QuickSight for Insights

Moving forward, Amazon Athena will be a key tool for querying the sanitized data stored in the clean S3 bucket. Since Athena is serverless and integrates directly with the AWS Glue Data Catalog, it'll let us run SQL queries on the data without worrying about setting up extra infrastructure. For example, we can use it to calculate the average revenue for movies released

during specific weather conditions or to explore how audience sentiment correlates with a movie's popularity. The results from these queries can then be connected to Amazon QuickSight, which we'll use to build interactive dashboards. These dashboards will visualize insights like how weather impacts box office performance, the relationship between audience sentiment and popularity, and trends in movie revenue over time. This setup will help us turn raw data into meaningful, actionable insights while keeping the process scalable and efficient as the project grows.

III. Conclusion

Importance of Deployment Package in Lambda Function

One of the challenges we came across while using lambda was the size restrictions. To upload our data and merge them into one file, we needed to zip the file and upload it onto Lambda, but soon we ran into the size limit restrictions in lambda. We needed to carefully choose our dependencies. We also needed to ensure that the versions of the libraries in our package were compatible with each other so that the code to merge our datafiles would successfully execute. The deployment package is crucial in the lambda function as it contains necessary materials such as third party libraries, the raw data, etc.

Choosing Athena Over Redshift for Cost-Effective Data Analysis

We decided to use Amazon Athena instead of Redshift for this project due to cost considerations and the nature of our data analysis needs. Athena allows us to run SQL queries directly on the cleaned data stored in S3, making it a cost-effective, serverless solution that eliminates the need for managing and maintaining a data warehouse. Since our primary goal is to analyze and visualize data rather than run continuous complex queries, Athena provides the flexibility and scalability we need without incurring the higher costs associated with Redshift. Additionally, its seamless integration with AWS Glue and QuickSight ensures that our pipeline remains streamlined and efficient, making it the ideal choice for our use case.

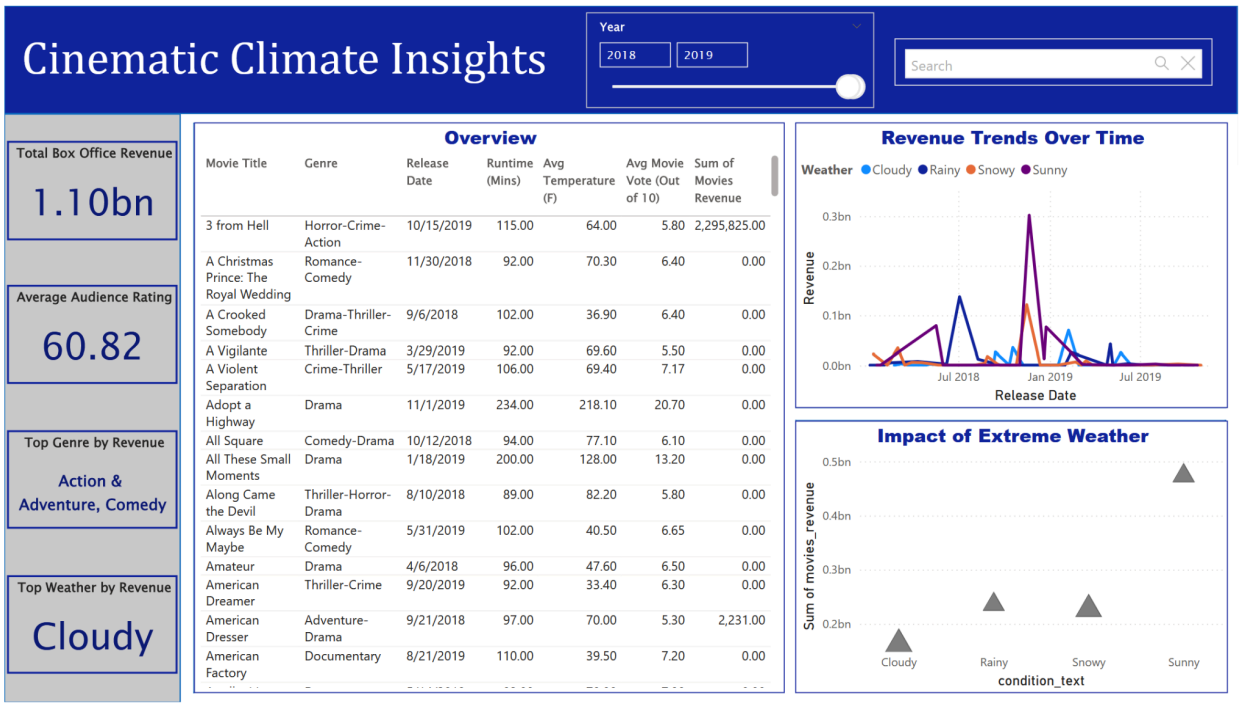
Concluding Thoughts

In conclusion, this project has demonstrated how a robust ELT pipeline can transform raw datasets into actionable insights using AWS services. By leveraging S3 for storage, Glue for cataloging and primary transformations, and Lambda for fine-tuned processing, we've created a scalable and efficient system for handling complex datasets like movie metadata, weather patterns, revenue, and audience sentiment. With the cleaned and structured data query-able from Athena, and visualization capabilities enabled through QuickSight, the pipeline provides a seamless workflow for in-depth analysis. This setup not only highlights the power of AWS tools in data engineering but also lays the groundwork for exploring meaningful correlations—like the impact of weather on movie performance or audience behavior trends. Moving forward, this pipeline can adapt to larger datasets and new analysis goals, making it a valuable framework for data-driven decision-making in the entertainment industry.

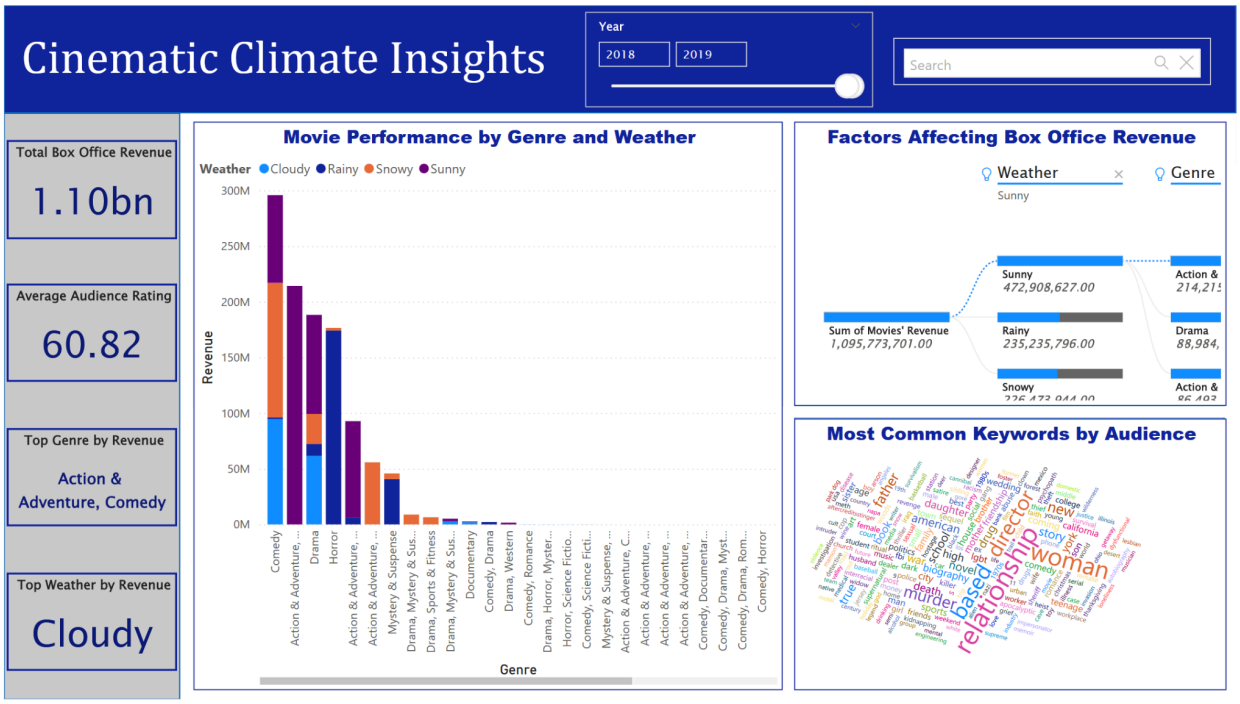
Final Project 2 Report

Cloud Project:

Overview of Dashboard: This first page reflects the main goal of the data in this project – to assess the effect of weather on film revenue and participation by audience. The second page delves into deeper insights using numerous visualizations to provide a better understanding of the data and how it relates to the weather and success of the film release.



Final Project 2 Report



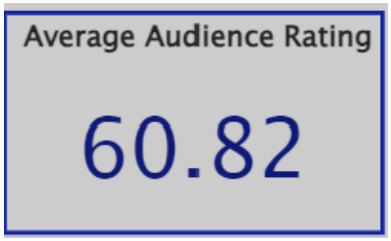
Year Filter: Allows users to select specific years (e.g., 2018, 2019) to filter all visuals and focus on data relevant to the chosen timeframe.

Search Bar: Enables keyword searches (e.g., movie titles) to quickly locate specific entries in the dataset.

Total Box Office Revenue KPI Card: Displays the total cumulative revenue generated by all movies, providing a high-level financial performance overview.

Average Audience Rating KPI Card: Shows the average audience rating across all movies, offering a quick summary of audience sentiment.

Final Project 2 Report



Top Genre by Revenue KPI Card: Highlights the movie genre (e.g., Action, Comedy) that contributed the highest revenue, giving insights into the most profitable category.

Top Weather by Revenue KPI Card: Identifies the weather condition (e.g., sunny, cloudy) that generated the highest total revenue, linking environmental factors to financial performance.

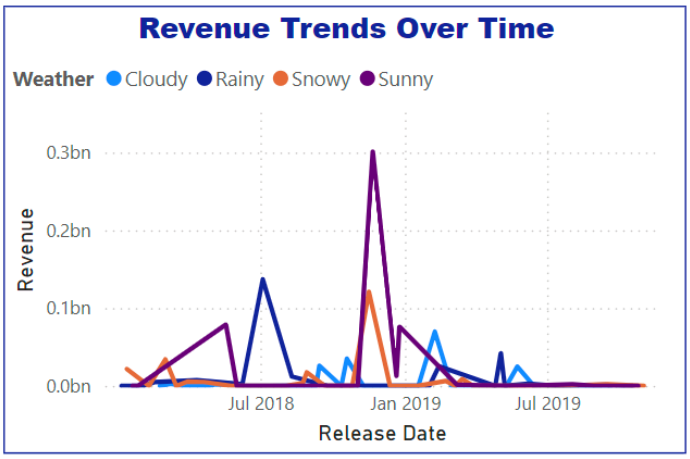
Overview Table Chart: Displays a detailed table with movie-specific data such as title, genre, release date, runtime, average temperature, audience rating, and revenue, allowing granular analysis.

Final Project 2 Report

Overview

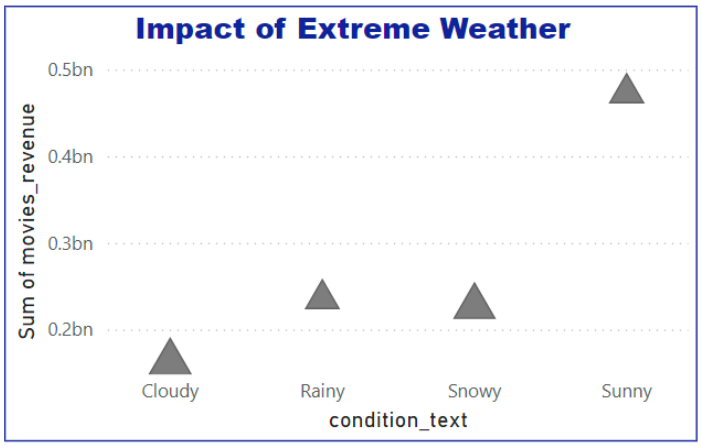
Movie Title	Genre	Release Date	Runtime (Mins)	Avg Temperature (F)	Avg Movie Vote (Out of 10)	Sum of Movies Revenue
3 from Hell	Horror-Crime-Action	10/15/2019	115.00	64.00	5.80	2,295,825.00
A Christmas Prince: The Royal Wedding	Romance-Comedy	11/30/2018	92.00	70.30	6.40	0.00
A Crooked Somebody	Drama-Thriller-Crime	9/6/2018	102.00	36.90	6.40	0.00
A Vigilante	Thriller-Drama	3/29/2019	92.00	69.60	5.50	0.00
A Violent Separation	Crime-Thriller	5/17/2019	106.00	69.40	7.17	0.00
Adopt a Highway	Drama	11/1/2019	234.00	218.10	20.70	0.00
All Square	Comedy-Drama	10/12/2018	94.00	77.10	6.10	0.00
All These Small Moments	Drama	1/18/2019	200.00	128.00	13.20	0.00
Along Came the Devil	Thriller-Horror-Drama	8/10/2018	89.00	82.20	5.80	0.00
Always Be My Maybe	Romance-Comedy	5/31/2019	102.00	40.50	6.65	0.00
Amateur	Drama	4/6/2018	96.00	47.60	6.50	0.00
American Dreamer	Thriller-Crime	9/20/2019	92.00	33.40	6.30	0.00
American Dresser	Adventure-Drama	9/21/2018	97.00	70.00	5.30	2,231.00
American Factory	Documentary	8/21/2019	110.00	39.50	7.20	0.00

Revenue Trends Over Time Line Chart: Visualizes movie revenue trends across release dates, segmented by weather conditions, to analyze seasonal or temporary impacts on performance.

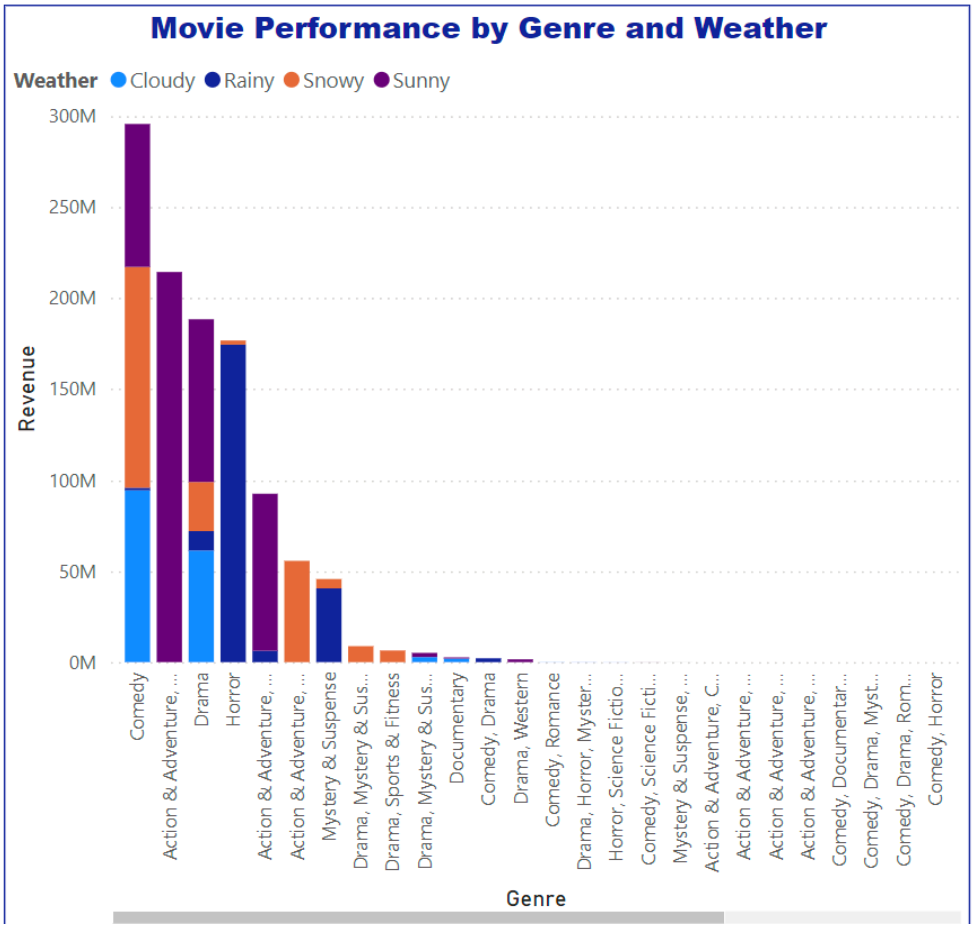


Final Project 2 Report

Impact of Extreme Weather Scatter Plot: Plots revenue against weather conditions (e.g., sunny, rainy) to reveal correlations between extreme weather and financial outcomes.

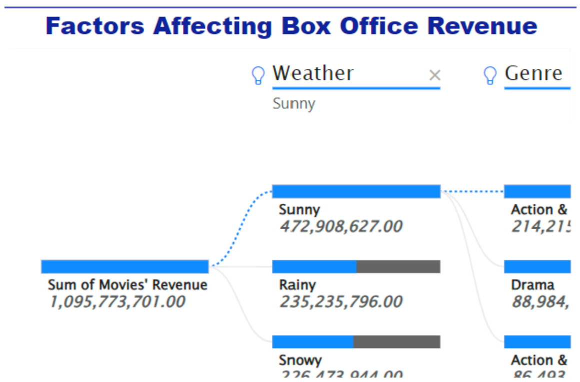


Movie Performance by Weather Stacked Column Chart: Breaks down movie revenues by genres and weather conditions, illustrating how different weather types affect various genres.



Final Project 2 Report

Factors Affecting Box Office Revenues Decomposition Tree: An interactive breakdown of revenue influenced by factors such as weather, genre, and audience ratings, helping to identify key contributors.



Most Common Keywords by Audience Word Cloud: Displays a word cloud of frequently used keywords in audience reviews, providing insights into themes or sentiments that resonate with viewers.

Link to PowerBI File and Presentation: [MILESTONE 7 CLOUD]