Project Details

End-to-End Yelp Data Analysis with AWS & Snowflake

Dheeraj S Kulkarni

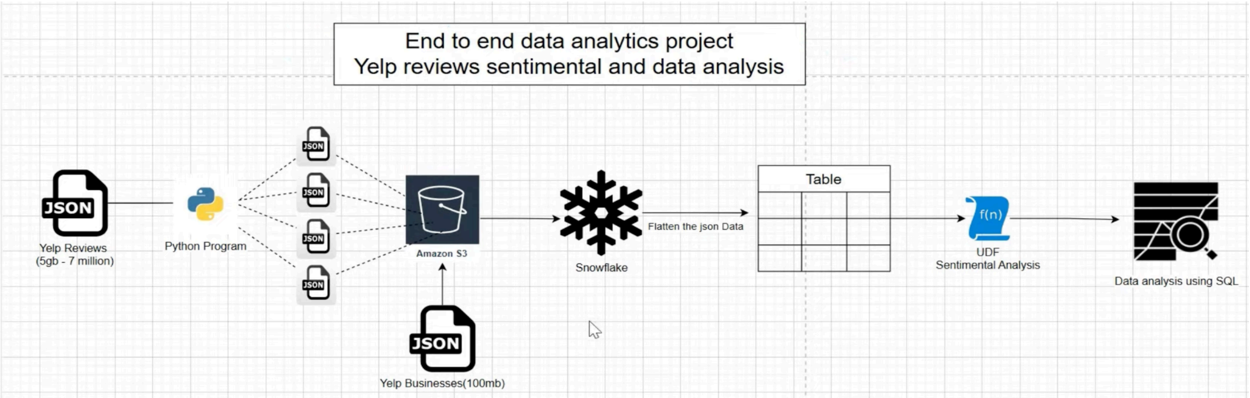
June 2025

Project Description

In this **end-to-end data analysis project**, I processed over **7 million Yelp reviews** and performed **sentiment analysis** and **EDA** using **SQL**, **AWS S3**, and **Snowflake**. This project showcases a full data pipeline — from data ingestion and storage to analysis and insights — offering a comprehensive walkthrough of building scalable analytics solutions using **AWS** and **Snowflake**.

Project Workflow

- Data Ingestion ➔ JSON files (Yelp reviews & business data)
- Cloud Storage (AWS) ➔ Amazon S3
- Data Warehousing ➔ Snowflake (structured tables)
- Sentiment Analysis ➔ SQL UDFs
- Business Insights ➔ SQL queries

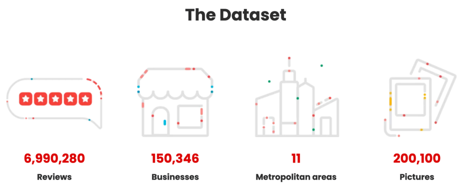


Key Takeaways:

- Efficiently handled large datasets by **splitting a massive JSON file into smaller, manageable chunks** for better performance.
- **Migrated data from AWS S3 to Snowflake** using the `COPY` command with optimized file formats for faster ingestion.
- Implemented **User-Defined Functions (UDFs)** to perform sentiment analysis.
- Wrote and executed **10+ analytical SQL queries** to uncover business insights, trends, and sentiment-driven patterns.
- Tackled and resolved performance bottlenecks in Snowflake to enable faster, large-scale data processing.

Resources

- **YT Source video:** Superstore Data Analysis | End to End AWS Data Engineering Proj...
- **Instructor:** Ankit Bansal
- **Dataset:** [Open Dataset | Yelp Data Licensing]()



IAM User

An IAM User represents an **individual person or application** with permanent credentials.

Key Features of IAM Users

- Has a username and password for AWS Console login.
- Can have access keys (Access Key ID & Secret Key) for API/CLI access.
- Permissions are controlled via IAM policies (e.g., Read/Write to S3).
- Used for long-term access to AWS resources.
- Cannot be assumed like roles; each user has dedicated credentials.

Amazon S3 (Simple Storage Service)

Amazon S3 (Simple Storage Service) is a **scalable, highly durable, and secure object storage service** that allows you to store and retrieve any amount of data from anywhere.

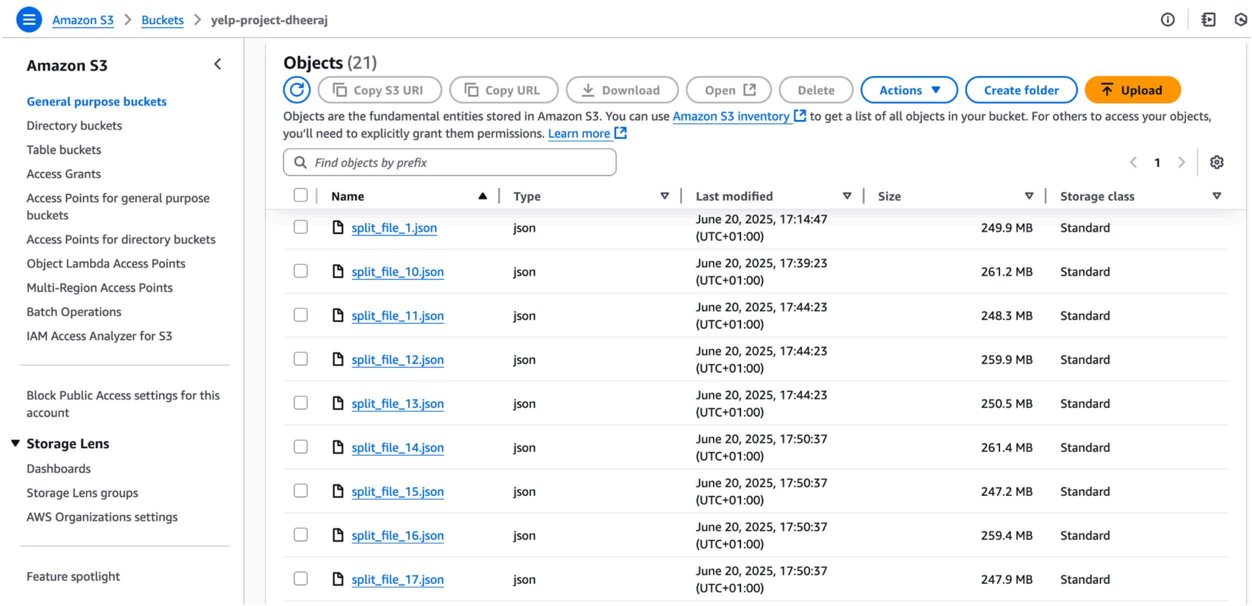
Snowflake

Snowflake is a fully managed, cloud-native **data platform** that supports **data warehousing, data lakes, data engineering, and advanced analytics** — all from a single platform. It separates **compute** and **storage**, allowing for independent scaling and better cost control.

Key Features of Snowflake

- **Cloud-native & fully managed**: No infrastructure to manage; runs on AWS, Azure, and GCP.
- **Separation of storage & compute**: Enables flexible and cost-effective scaling.
- **Supports multiple workloads**: BI, ELT/ETL, data science, and data sharing.

Screen capture of AWS S3 bucket



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

Name	Type	Last modified	Size	Storage class
split_file_1.json	json	June 20, 2025, 17:14:47 (UTC+01:00)	249.9 MB	Standard
split_file_10.json	json	June 20, 2025, 17:39:23 (UTC+01:00)	261.2 MB	Standard
split_file_11.json	json	June 20, 2025, 17:44:23 (UTC+01:00)	248.3 MB	Standard
split_file_12.json	json	June 20, 2025, 17:44:23 (UTC+01:00)	259.9 MB	Standard
split_file_13.json	json	June 20, 2025, 17:44:23 (UTC+01:00)	250.5 MB	Standard
split_file_14.json	json	June 20, 2025, 17:50:37 (UTC+01:00)	261.4 MB	Standard
split_file_15.json	json	June 20, 2025, 17:50:37 (UTC+01:00)	247.2 MB	Standard
split_file_16.json	json	June 20, 2025, 17:50:37 (UTC+01:00)	259.4 MB	Standard
split_file_17.json	json	June 20, 2025, 17:50:37 (UTC+01:00)	247.9 MB	Standard

SQL queries to analyze the data and generate insights

1) Find the number of businesses in each category

```
WITH category_cte AS (
SELECT business_id, TRIM(C.value) AS category
FROM yelp_businesses_table,
LATERAL split_to_table(categories,',') AS C
)

SELECT category, COUNT(*) AS number_of_businesses
FROM category_cte
GROUP BY category
ORDER BY number_of_businesses DESC;
```

---

2) Find the top 10 users who have reviewed the most businesses in the “Restaurants” category

```
WITH cte1 AS (
SELECT r.user_id, COUNT(DISTINCT r.business_id) AS distinct_count_reviews
FROM yelp_businesses_table AS b
INNER JOIN yelp_reviews_table AS r
ON b.business_id = r.business_id
WHERE categories ILIKE '%Restaurants%'
GROUP BY  r.user_id
ORDER BY distinct_count_reviews DESC
)

, cte2 AS (
SELECT *, RANK() OVER(ORDER BY distinct_count_reviews DESC) AS review_rank
FROM cte1
)

SELECT user_id, distinct_count_reviews
FROM cte2
WHERE review_rank <=10;
```

---

3) Find the most popular categories of businesses (based on the number of reviews)

```
WITH category_cte AS (
SELECT business_id, TRIM(C.value) AS category
FROM yelp_businesses_table,
LATERAL split_to_table(categories,',') AS C
)

SELECT c.category, COUNT(*) AS number_of_reviews
FROM yelp_reviews_table r
INNER JOIN category_cte c
ON r.business_id = c.business_id
GROUP BY category
ORDER BY number_of_reviews DESC;
```

4) Find the top 3 most recent reviews for each business

```
WITH cte1 AS (
SELECT *, ROW_NUMBER() OVER(PARTITION BY r.business_id ORDER BY r.review_date DESC)
AS rn
FROM yelp_businesses_table AS b
INNER JOIN yelp_reviews_table AS r
ON b.business_id = r.business_id
)

SELECT *
FROM cte1
WHERE rn<=3;
```

5) Find the month with the highest number of reviews

```
WITH cte AS(
SELECT EXTRACT(MONTH FROM review_date) AS review_month, COUNT(*) AS total_reviews,
RANK() OVER(ORDER BY total_reviews DESC) AS rank_review
FROM yelp_reviews_table
GROUP BY review_month
ORDER BY total_reviews DESC
)

SELECT * FROM cte
WHERE rank_review = 1;
```

6) Find the percentage of 5-star reviews for each business

```
SELECT b.business_id, b.name, COUNT(*) AS total_reviews,
COUNT(CASE WHEN r.review_stars = '5' THEN 1 ELSE NULL END) AS five_star_review,
ROUND((five_star_review/total_reviews)*100,2) AS percent_of_five_star_review
FROM yelp_reviews_table r
INNER JOIN yelp_businesses_table b
ON r.business_id = b.business_id
GROUP BY b.business_id, b.name
ORDER BY percent_of_five_star_review DESC;
```

7) Find the top 5 most reviewed businesses in each city

```
WITH cte1 AS (
SELECT b.city, b.business_id, b.name, COUNT(*) AS total_reviews
FROM yelp_businesses_table AS b
INNER JOIN yelp_reviews_table AS r
ON b.business_id = r.business_id
GROUP BY b.city, b.business_id, b.name
)

,cte2 AS(
SELECT  city, name, total_reviews, ROW_NUMBER() OVER(PARTITION BY city ORDER BY
total_reviews DESC) AS rank_review
FROM cte1
)

SELECT * FROM cte2
WHERE rank_review <=5
ORDER BY city,rank_review;
```

8) Find the average rating of businesses that have at least 100 reviews

```
SELECT b.business_id, b.name, COUNT(*) AS total_reviews,
ROUND(AVG(review_stars),2) AS avg_rating
FROM yelp_reviews_table r
INNER JOIN yelp_businesses_table b
ON r.business_id = b.business_id
GROUP BY b.business_id, b.name
HAVING total_reviews >=100
ORDER BY total_reviews DESC;
```

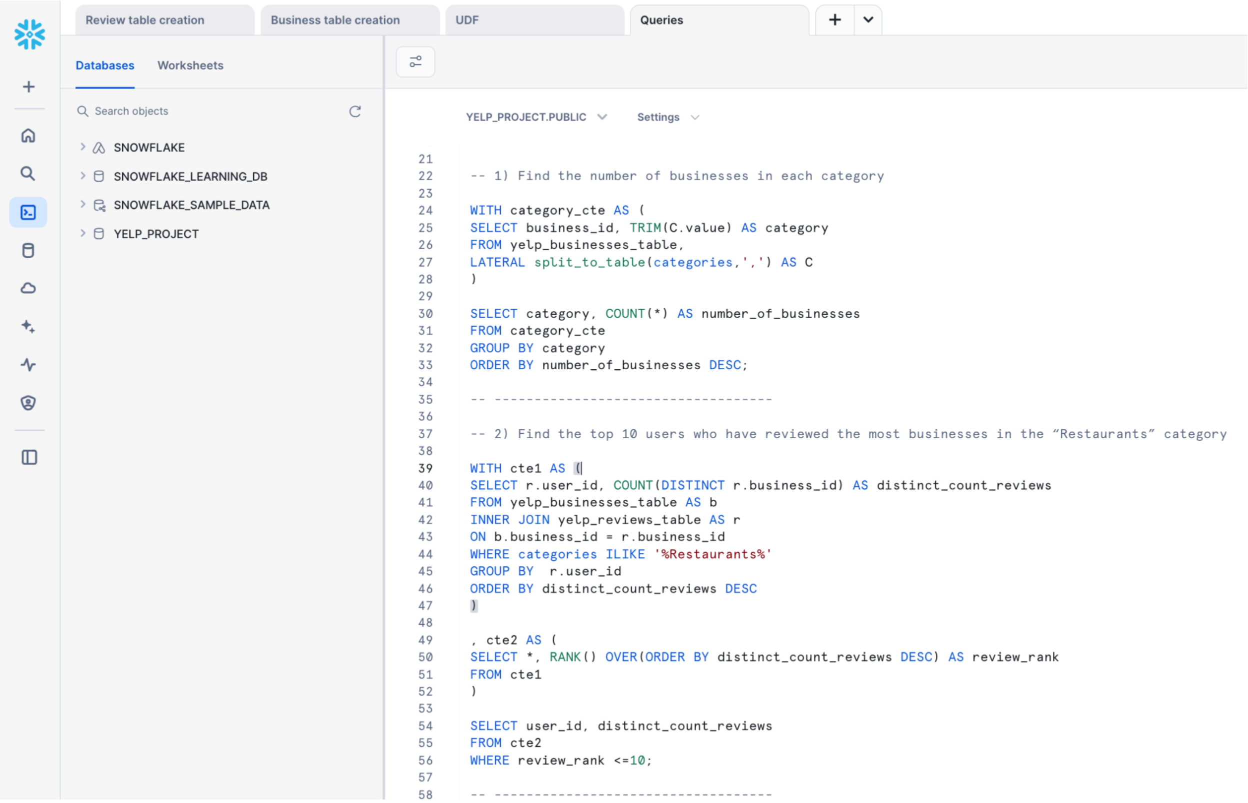
9) List the top 10 users who have written the most reviews

```
WITH cte1 AS (
SELECT r.user_id, COUNT(*) total_reviews
FROM yelp_businesses_table AS b
INNER JOIN yelp_reviews_table AS r
ON b.business_id = r.business_id
GROUP BY  r.user_id
)

, cte2 AS (
SELECT *, RANK() OVER(ORDER BY total_reviews DESC) AS review_rank
FROM cte1
)

SELECT user_id,business_id, total_reviews
FROM cte2
WHERE review_rank <= 10
ORDER BY total_reviews DESC;
```

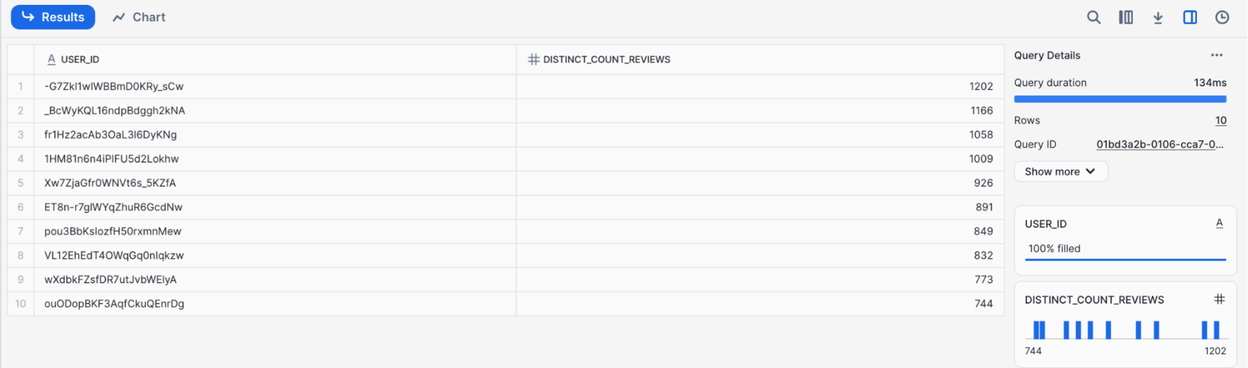
10) Find the top 10 businesses with the highest positive sentiment reviews

```
WITH cte1 AS (
SELECT b.business_id, b.name, COUNT(*) AS total_reviews
FROM yelp_reviews_table r
INNER JOIN yelp_businesses_table b
ON r.business_id = b.business_id
WHERE sentiments = 'Positive'
GROUP BY b.business_id, b.name
)

, cte2 AS (
SELECT *, RANK() OVER(ORDER BY total_reviews DESC) as rank_review
FROM cte1
)

SELECT * FROM
cte2
WHERE rank_review <=10;
```

Screen captures of some queries and results in Snowflake



```sql

-- 1) Find the number of businesses in each category

WITH category_cte AS (
SELECT business_id, TRIM(C.value) AS category
FROM yelp_businesses_table,
LATERAL split_to_table(categories,',') AS C
)

SELECT category, COUNT(*) AS number_of_businesses
FROM category_cte
GROUP BY category
ORDER BY number_of_businesses DESC;

-- -----------------------------------

-- 2) Find the top 10 users who have reviewed the most businesses in the “Restaurants” category

WITH cte1 AS (
SELECT r.user_id, COUNT(DISTINCT r.business_id) AS distinct_count_reviews
FROM yelp_businesses_table AS b
INNER JOIN yelp_reviews_table AS r
ON b.business_id = r.business_id
WHERE categories ILIKE '%Restaurants%'
GROUP BY  r.user_id
ORDER BY distinct_count_reviews DESC
)

, cte2 AS (
SELECT *, RANK() OVER(ORDER BY distinct_count_reviews DESC) AS review_rank
FROM cte1
)

SELECT user_id, distinct_count_reviews
FROM cte2
WHERE review_rank <=10;

-- -----------------------------------
```



	USER_ID	DISTINCT_COUNT_REVIEWS
1	-G7Zkl1wlWBBmD0KRy_sCw	1202
2	_BcWyKQL16ndpBdggh2kNA	1166
3	fr1Hz2acAb3OaL3l6DyKNg	1058
4	1HM81n6n4iPIFU5d2Lokhw	1009
5	Xw7ZjaGfr0WNVt6s_5KZfA	926
6	ET8n-r7glWYqZhuR6GcdNw	891
7	pou3BbKslozfH50rxmnMew	849
8	VL12EhEdT4OWqGq0nlqkzw	832
9	wXdbkFZsfDR7utJvbWElyA	773
10	ouODopBKF3AqfCkuQEnrDg	744

Query Details

Query duration 134ms

Rows 10

Query ID 01bd3a2b-0106-cca7-0...

USER_ID

100% filled

DISTINCT_COUNT_REVIEWS