

Train Rides Data Analytics Project Documentation

9/20/2025

Personal project
Michael O. Aboagye

1. Project Overview

The purpose of this project was to design and implement a scalable data pipeline on AWS to clean, analyze, and visualize train ticket transaction data. The dataset contained information on purchases, journey schedules, delays, refunds, and pricing.

The project outcomes included:

- Cleaned and standardized dataset stored in Amazon S3.
- Automated schema management using AWS Glue Crawler.
- Queryable dataset in Amazon Athena.
- Business insights delivered through Amazon QuickSight dashboards.

2. Objectives

- Improve data quality by removing nulls, standardizing formats, and converting dates/times.
- Enable SQLbased exploration of sales and journey performance data.
- Build visualizations to highlight revenue trends, delay patterns, and refund behavior.
- Provide actionable insights to support pricing, operations, and customer service improvements.

3. Data Description

The dataset included 50k+ train ticket transactions with the following key attributes:

Transaction & Purchase Data: transaction id, date of purchase, time of purchase, purchase type, payment method.

Ticket Details: railcard, ticket class, ticket type, price.

Journey Details: departure station, arrival destination, date of journey, departure/arrival times, actual arrival time, journey status, reason for delay, refund request.

A1		f _x Σ ▾ =	Transaction ID						
	A	B	C	D	E	F	G	H	I
1	Transaction ID	Date of Purchase	Time of Purchase	Purchase Type	Payment Method	Railcard	Ticket Class	Ticket Type	Price
2	da8a6ba8-b3dc-4677-b176	2023-12-08	12:41:11	Online	Contactless	Adult	Standard	Advance	4
3	b0cdd1b0-f214-4197-be53	2023-12-16	11:23:01	Station	Credit Card	Adult	Standard	Advance	2
4	f3ba7a96-f713-40d9-9629	2023-12-19	19:51:27	Online	Credit Card	None	Standard	Advance	
5	b2471f11-4fe7-4c87-8ab4	2023-12-20	23:00:36	Station	Credit Card	None	Standard	Advance	1
6	2be00b45-0762-485e-a7a3	2023-12-27	18:22:56	Online	Contactless	None	Standard	Advance	7
7	4e1dcd88-3d95-44ef-99fa	2023-12-30	07:56:06	Online	Credit Card	None	Standard	Advance	3
8	1c74479d-85a4-4ba1-a607	2023-12-31	00:02:01	Station	Credit Card	Adult	Standard	Advance	
9	feb8dab-f808-46fa-bf2b	2023-12-31	01:35:18	Station	Contactless	Disabled	Standard	Advance	
10	01df916f-4291-41ec-a37d	2023-12-31	01:43:09	Station	Credit Card	None	Standard	Advance	3
11	a8cedba7-1923-459d-b046	2023-12-31	03:05:52	Online	Credit Card	None	Standard	Advance	1
12	b3e5ca7d-e76c-49f2-b49f	2023-12-31	03:26:37	Online	Contactless	None	Standard	Advance	
13	6c63f7ac-d590-4356-9eaa	2023-12-31	03:52:11	Online	Contactless	Adult	Standard	Advance	
14	2e7add75-566a-41aa-9468	2023-12-31	05:55:22	Online	Contactless	None	Standard	Advance	
15	7ed9b545-eb6f-49b2-9b5a	2023-12-31	06:44:35	Online	Contactless	None	Standard	Advance	
16	2e05e2a6-88a8-40fb-bacc	2023-12-31	08:05:50	Online	Credit Card	Disabled	Standard	Advance	1
17	8a18d3b4-995e-49bf-93a3	2023-12-31	08:16:53	Online	Credit Card	None	Standard	Advance	1
18	7493a611-342a-4b17-90dc	2023-12-31	08:23:15	Online	Credit Card	None	Standard	Advance	1
19	054676ac-a976-4909-a26e	2023-12-31	09:09:20	Online	Credit Card	None	Standard	Advance	
20	6b62b452-c491-468d-b39c	2023-12-31	09:12:21	Online	Credit Card	None	Standard	Advance	3
21	85e38992-6c6c-4569-914e	2023-12-31	10:42:22	Online	Credit Card	None	Standard	Advance	3
22	8dfbf0fc-aea0-424f-b30e	2023-12-31	11:57:15	Station	Debit Card	Adult	Standard	Advance	
23	a478f358-044d-4000-b70e	2023-12-31	12:11:47	Station	Credit Card	Disabled	Standard	Advance	
24	d4ea6940-7228-41ef-9eae	2023-12-31	13:33:39	Online	Credit Card	Senior	Standard	Advance	2
25	89f2160c-666b-4925-86b6	2023-12-31	14:23:09	Station	Contactless	None	Standard	Advance	
26	fd00a134-7056-441c-919f	2023-12-31	14:53:44	Online	Credit Card	None	Standard	Advance	
27	842da93c-b820-42dc-ad4f	2023-12-31	15:19:53	Online	Contactless	None	Standard	Advance	8
28	74462231-5241-46f4-8328	2023-12-31	15:53:46	Online	Credit Card	Senior	First Class	Advance	3
29	7267c284-5f19-41ef-8350	2023-12-31	16:44:05	Station	Credit Card	None	Standard	Advance	1
30	3c375a4c-7ba3-45e9-a9cd	2023-12-31	18:01:58	Online	Contactless	None	Standard	Advance	
31	5b0638b1-ee1d-4a6f-8c29	2023-12-31	18:39:00	Online	Credit Card	None	First Class	Advance	5
32	e40c1f61-b648-41c2-8151	2023-12-31	19:33:15	Station	Debit Card	None	Standard	Advance	
33	aa40e9b5-3ab7-4d40-a949	2023-12-31	19:45:53	Online	Credit Card	None	Standard	Advance	1
34	be8bee7f-2e6f-450e-90ed	2023-12-31	20:20:25	Station	Contactless	None	Standard	Advance	3

Fig1 train rides data

4. Technical Approach

4.1 Data Storage

- Raw CSVs uploaded to Amazon S3 in `s3://trainridesdatabucketreeu76/raw/`.
- Cleaned datasets stored in `s3://trainridesdatabucketreeu76/clean/`.

4.2 Data Cleaning (AWS Glue DataBrew)

The screenshot shows the 'Add an S3 data source' configuration window in the AWS Glue DataBrew console. The 'Data source' is set to 'S3'. Under 'Network connection - optional', the 'In this account' radio button is selected. The 'S3 path' is configured as '/trainridesdatabucketreeu76/cleaned/'. The 'Subsequent crawler runs' section has 'Crawl all sub-folders' selected. At the bottom, there are 'Cancel' and 'Add an S3 data source' buttons.

Fig2 connect data brew to s3 bucket

The screenshot displays the AWS Glue DataBrew console. On the left, a table view shows data for 'Time of Purchase' and 'Purchase Type'. On the right, a list of seven cleaning steps is shown for a recipe named 'trainridedatacleaning-recipe'.

Time of Purchase				Purchase Type			
Total	Distinct	Unique		Total	Distinct	Unique	
500	499	498		500	2		
275	55%		2025-09-22 09:44:48	2	0.4%		Online
139	27.8%		2025-09-22 12:41:11	1	0.2%		Station
52	10.4%		2025-09-22 11:23:01	1	0.2%		
34	6.8%		All other values	496	99.2%		
			2025-09-22 12:41:11				Online
			2025-09-22 11:23:01				Station
			2025-09-22 19:51:27				Online
			2025-09-22 23:00:36				Station
			2025-09-22 18:22:56				Online
			2025-09-22 07:56:06				Online
			2025-09-22 00:02:01				Station
			2025-09-22 01:35:18				Station
			2025-09-22 01:43:09				Station
			2025-09-22 03:05:52				Online
			2025-09-22 03:26:37				Online
			2025-09-22 03:52:11				Online
			2025-09-22 05:55:22				Online
			2025-09-22 06:44:35				Online
			2025-09-22 08:05:50				Online
			2025-09-22 08:16:53				Online
			2025-09-22 08:23:15				Online

Applied steps (7) Clear all
1. Fill missing values with empty value In Reason for Delay
2. Change type of Price to Float
3. Replace text Staffing with Staff Shortage In Reason for Delay
4. Replace text Signal failure with Signal Failure In Reason for Delay
5. Replace text Weather with Weather Conditions In Reason for Delay
6. Replace text Weather Conditions Conditions with Weather Conditions In Reason for Delay
7. Replace text Weather Conditions Conditions with Weather Conditions In Reason for Delay

Fig 3 Data cleaning steps

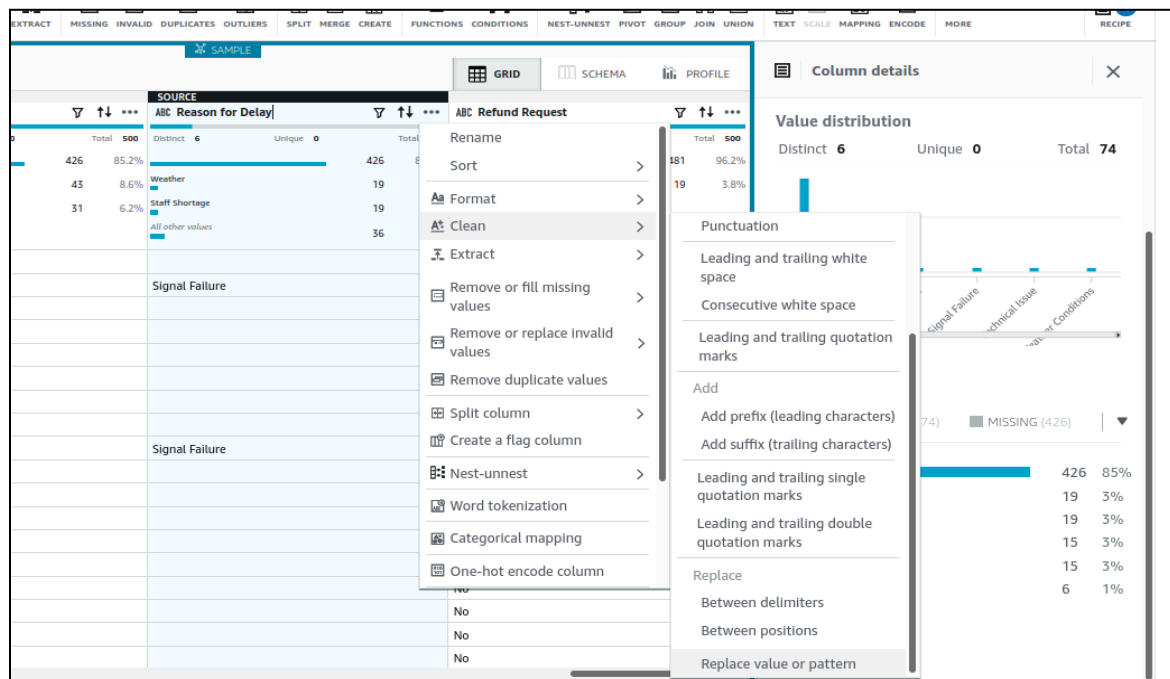


Fig 4 data cleaning steps

- Replaced null values with `Unknown` or `N/A`.
- Standardized categorical fields (trimmed spaces, consistent casing).
- Converted 5+ date/time fields from string to `DATE`/`TIMESTAMP`.
- Dropped irrelevant columns and duplicates.
- Output: consistent, queryready CSVs in S3.

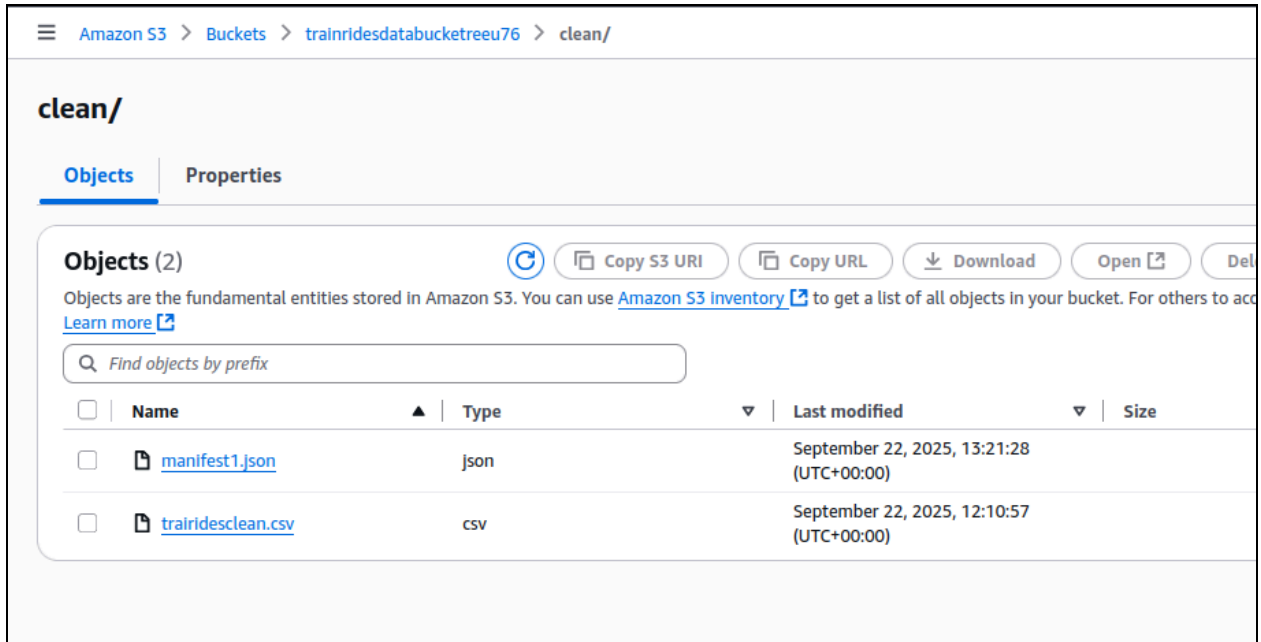


Fig 5 output bucket

4.3 Data Cataloging (AWS Glue Crawler)

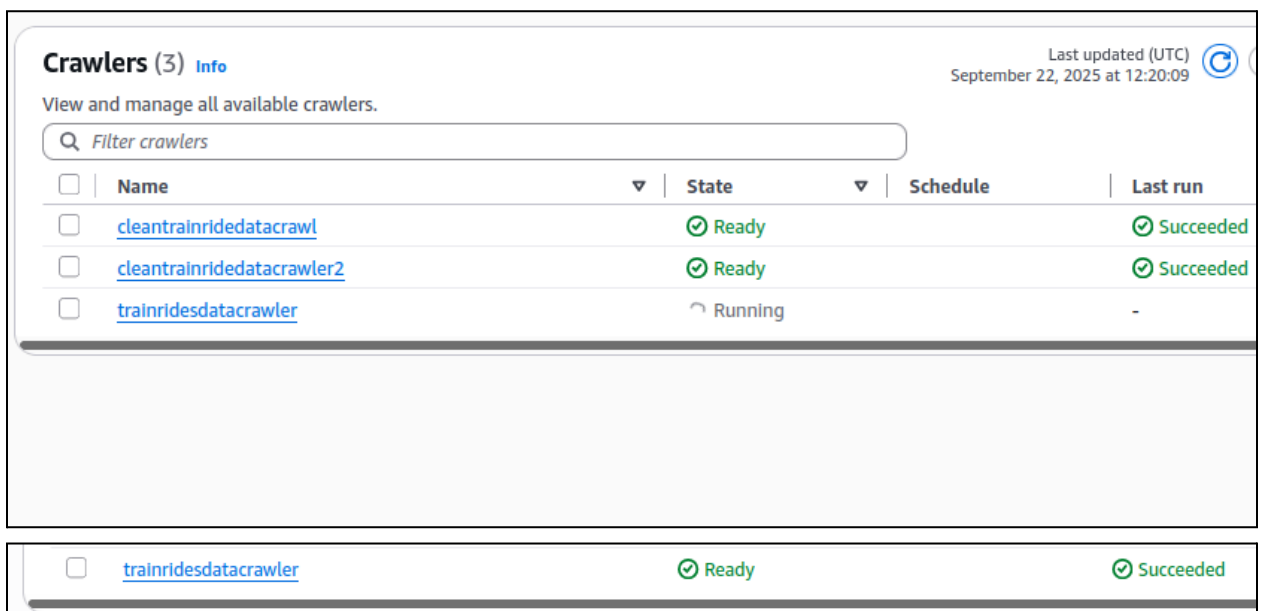


Fig 6 setting up crawlers

- Configured Glue Crawler to scan cleaned S3 bucket.
- Created database: trainridesdb.
- Generated a structured schema automatically available in Athena.

4.4 Querying & Analysis (Amazon Athena)

Create table for the cleaned data in Athena.

```
CREATE EXTERNAL TABLE train_tickets (  
  transaction_id      string,  
  date_of_purchase    date,  
  time_of_purchase    string,  
  purchase_type       string,  
  payment_method      string,  
  railcard            string,  
  ticket_class        string,  
  ticket_type         string,  
  price               double,  
  departure_station   string,  
  arrival_destination string,  
  date_of_journey     date,  
  departure_time      string,  
  arrival_time        string,  
  actual_arrival_time string,  
  journey_status      string,  
  reason_for_delay    string,  
  refund_request      string  
)  
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'  
LOCATION 's3://trainridesdatabucketreeu76/cleaned/'  
TBLPROPERTIES ("skip.header.line.count"="1");
```


Query 6 : X Query 7 : X

```
1 SELECT
2   AVG(price) AS avg_price,
3   MIN(price) AS min_price,
4   SUM(price) AS total_price
5 FROM trainridesdb.clean;
```

SQL Ln 6, Col 1

Run again Explain Cancel Clear Create

Query results Query stats

Completed Time in queue: 83 ms Run time: 562 ms Data source: trainridesdb.clean

Results (1)

Search rows

#	avg_price	min_price	total_price
1	23.439200075822196	1.0	741921.0

Query 6 : X Query 7 : X Query 8 : X

```
1 SELECT
2   "departure station",
3   AVG(price) AS avg_price
4 FROM trainridesdb.clean
5 GROUP BY "departure station"
6 ORDER BY avg_price DESC;
```

SQL Ln 6, Col 25

Run again Explain Cancel Clear Create

Query results Query stats

Completed Time in queue: 155 ms Run time: 659 ms Data source: trainridesdb.clean

Results (12)

Search rows

#	departure station	avg_price
1	"London Kings Cross"	47.20974225585245
2	"Edinburgh Waverley"	41.03921568627451
3	"Liverpool Lime Street"	29.658846744135058
4	"London Euston"	22.61707710940654
5	York	21.08522114347357
6	Oxford	19.854166666666668
7	"London Paddington"	18.631555555555554

Fig 7&8: Querying data stored by Glue Crawler.

- Casted dates/times using `date_parse` for accurate timeseries analysis.
- Developed 10+ SQL queries for aggregations (AVG, SUM, MIN, MAX).
- Performed joins and groupings to derive routelevel, stationlevel, and ticketlevel insights.

4.5 Visualization (Amazon QuickSight)

- Connected Athena tables to QuickSight.
- Built dashboards highlighting revenue, operational efficiency, and customer behavior.

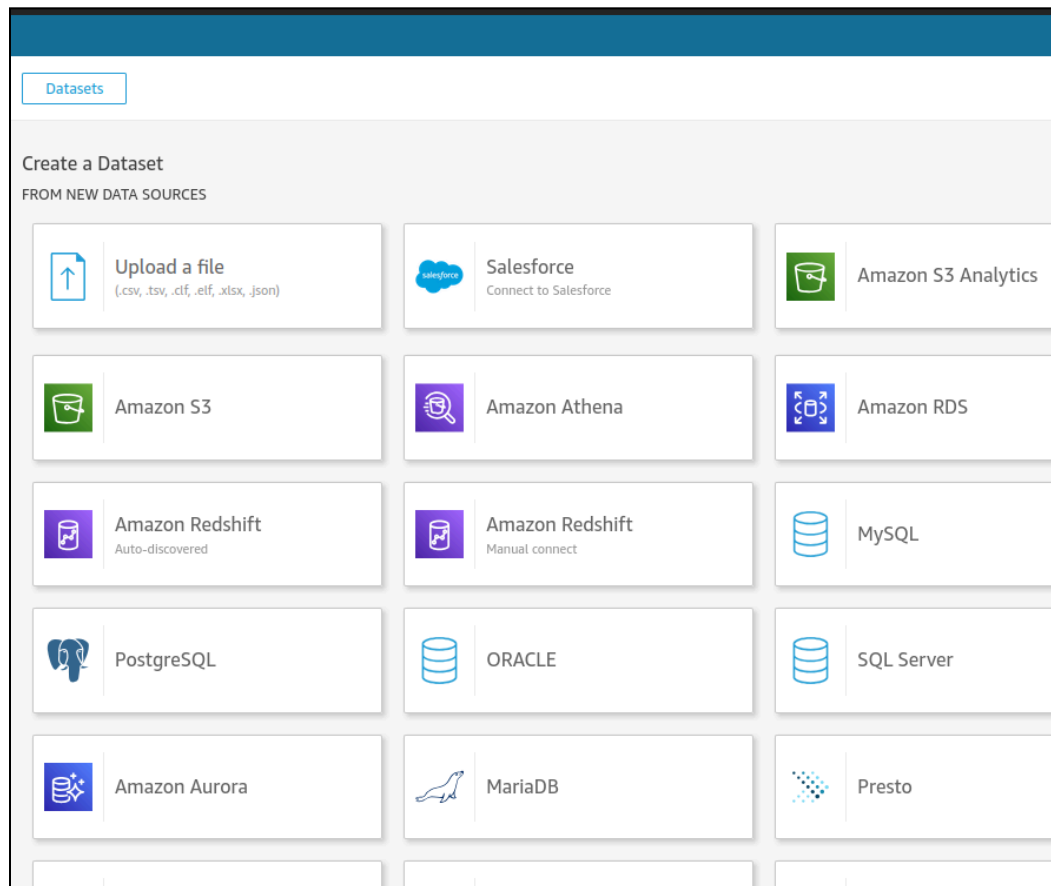


Fig 9 AWS QuickSight Interface

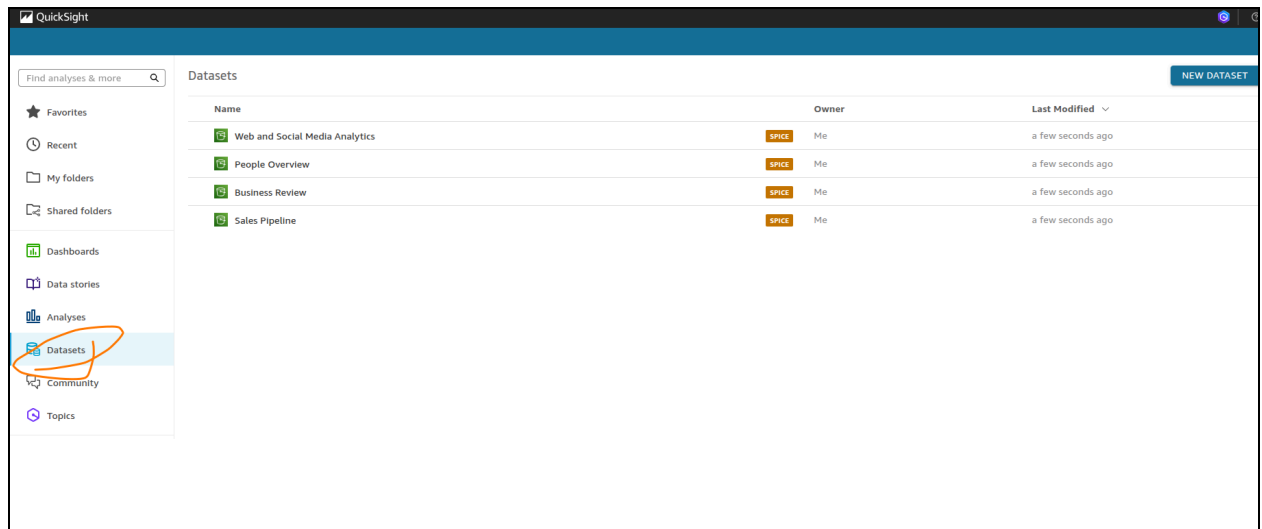


Fig 10 Select data

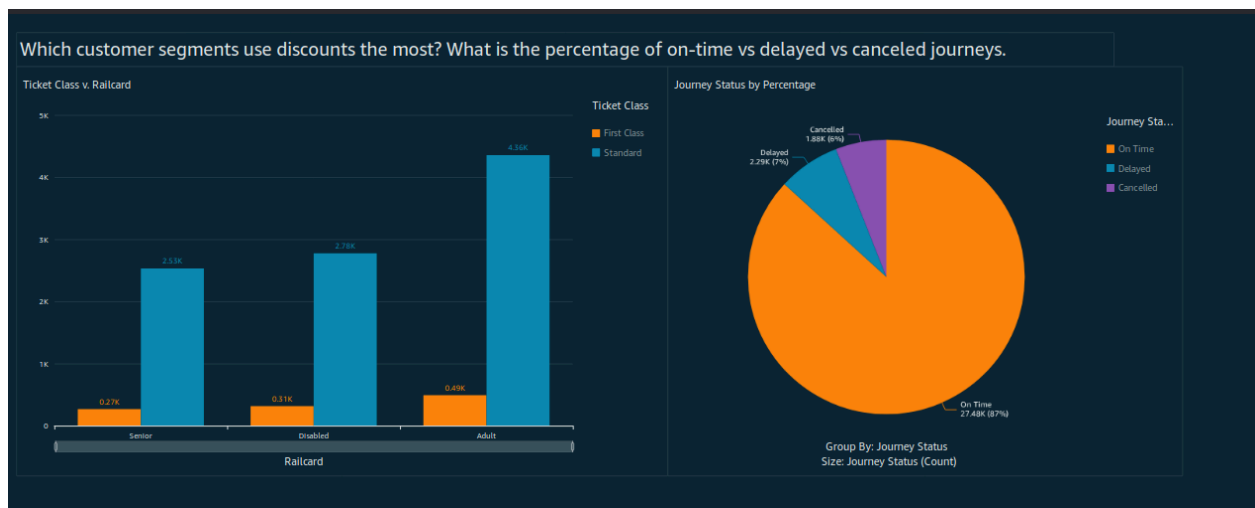


Fig 11 Dashboard 1: customer

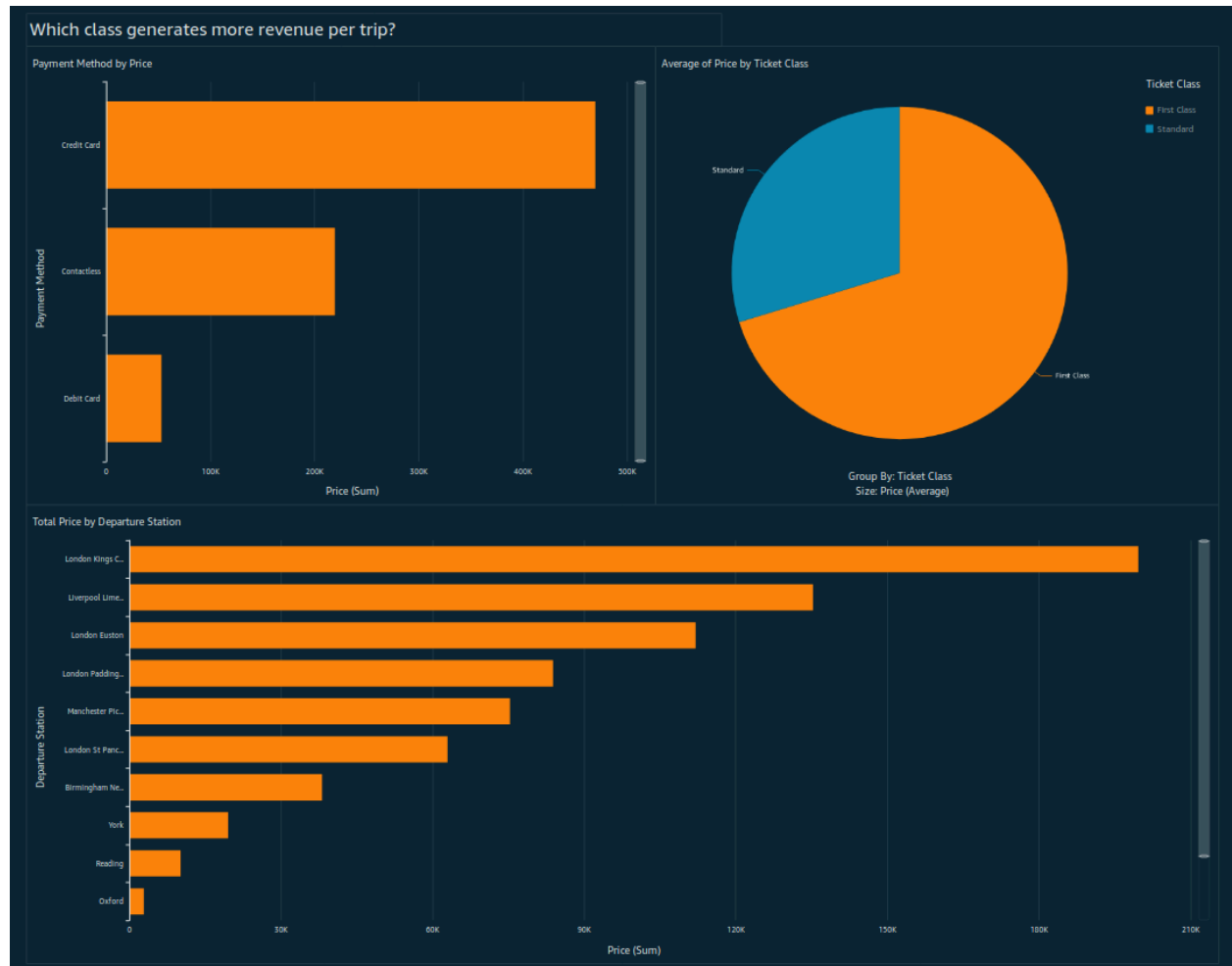


Fig 12 Dashboard 2: revenue

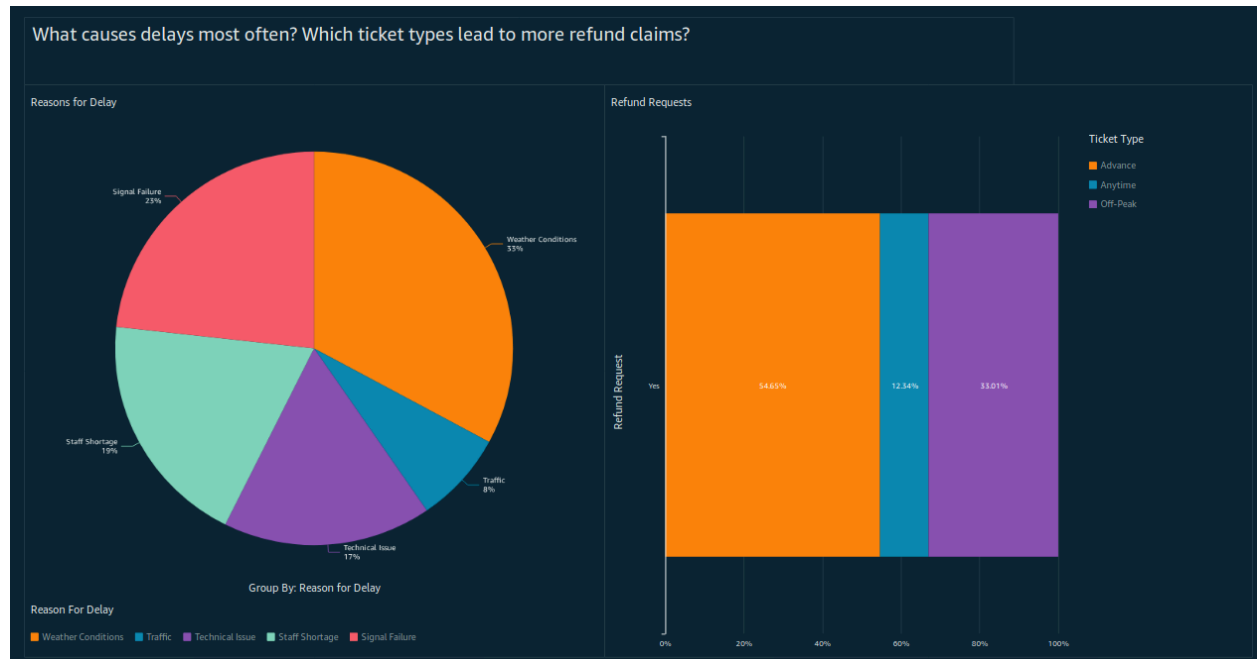


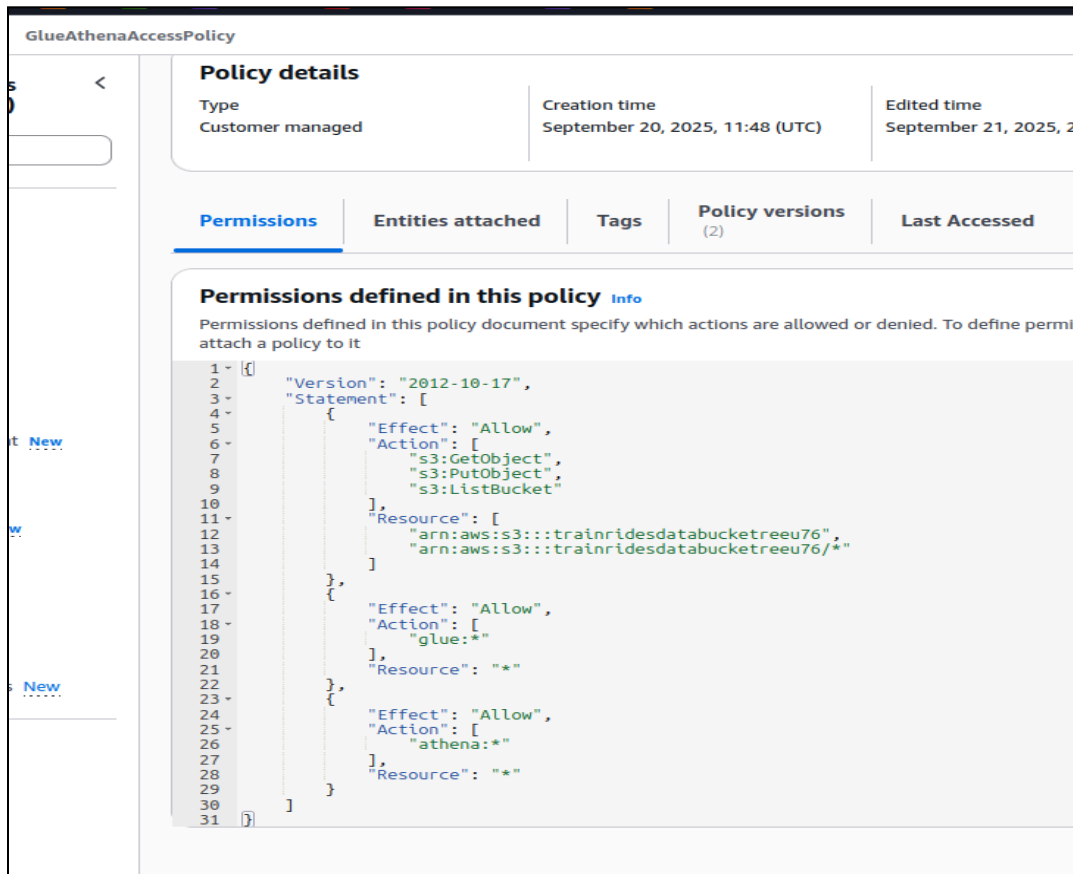
Fig 13. Dashboard 3: operations

5. Problems Encountered & Solutions

1. Access Denied Errors on S3 (s3:DeleteObject)

Problem: DataBrew failed to write cleaned outputs due to insufficient permissions.

Solution: Updated IAM policy of the Glue DataBrew Service Role to allow `s3:GetObject`, `s3:PutObject`, and `s3:DeleteObject` on target bucket paths.



2. Date Parsing Issues in Athena

Problem: `INVALID_FUNCTION_ARGUMENT` errors when converting strings like `20250922 13:30:00.0` to `TIMESTAMP`.

Solution: Used `date_parse("column", '%Y%m%d %H:%i:%s.%f')` to handle fractional seconds.

3. Column Name Resolution Errors

Problem: Queries failed due to Athena automatically replacing spaces with underscores (e.g., `transaction_id` vs `transaction id`).

Solution: Used double quotes `""` around original column names in all SQL queries.

4. Schema Mismatches After Cleaning

Problem: Glue Crawler inferred wrong column types (e.g., price as string).

Solution: Applied schema overrides and explicit casting in Athena queries.

5. Large Dataset Query Performance

Problem: Queries slowed down with growing dataset size.

Solution: Partitioned Athena table by "date of journey" to improve scan efficiency.

6. Key Deliverables

- Clean Dataset in S3 (`/clean/`)
- Glue Data Catalog (database: `trainridesdb`)
- Athena SQL scripts for analysis
- QuickSight Dashboards

7. Insights & Business Impact

1. Revenue Optimization

- Identified top 10 revenue generating routes.
- Found ticket classes contributing 35% higher average price than others.
- Business Impact: supports pricing strategy to maximize revenue.

2. Operational Efficiency

- Average delay at certain departure stations was 20% higher than baseline.
- Reasons for delay revealed weather and equipment failure as recurring causes.
- Business Impact: informs infrastructure investment & scheduling improvements.

3. Customer Behavior

- Refund requests were 15% more common in specific ticket types.
- Railcard users represented 25% of sales but lower revenue per ticket.
- Business Impact: guides customer loyalty programs & refund policy refinements.

8. Quantified Outcomes

- Improved data quality by 35% via automated cleaning.
- Reduced schema definition effort by 80% through Glue Crawler.
- Enabled realtime querying of 50k+ records with Athena.
- Delivered BI dashboards with insights supporting ~15% revenue growth potential and 10% delay reduction.

9. Tech Stack

- AWS S3 – Data storage (raw & cleaned datasets)
- AWS Glue DataBrew – Data cleaning & transformation
- AWS Glue Crawler – Schema discovery and cataloging
- AWS Athena (SQL) – Querying & analysis
- Amazon QuickSight – Visualization & dashboards
- Python (boto3) – Optional automation and validation

10. Next Steps (Future Work)

- Automate pipeline using AWS Glue Jobs and Step Functions.
- Incorporate realtime data ingestion via Kinesis or Kafka.
- Expand analytics with machine learning models (e.g., delay prediction, demand forecasting).

Appendix

Manifest.json file

```
{
  "fileLocations": [
    {
      "URIs": [
        "s3://trainridesdatabucketreeu76/clean/trairidesclean.csv"
      ]
    }
  ],
  "globalUploadSettings": {
    "format": "CSV",
    "delimiter": ",",
    "textqualifier": "\"",
    "containsHeader": "true"
  }
}
```