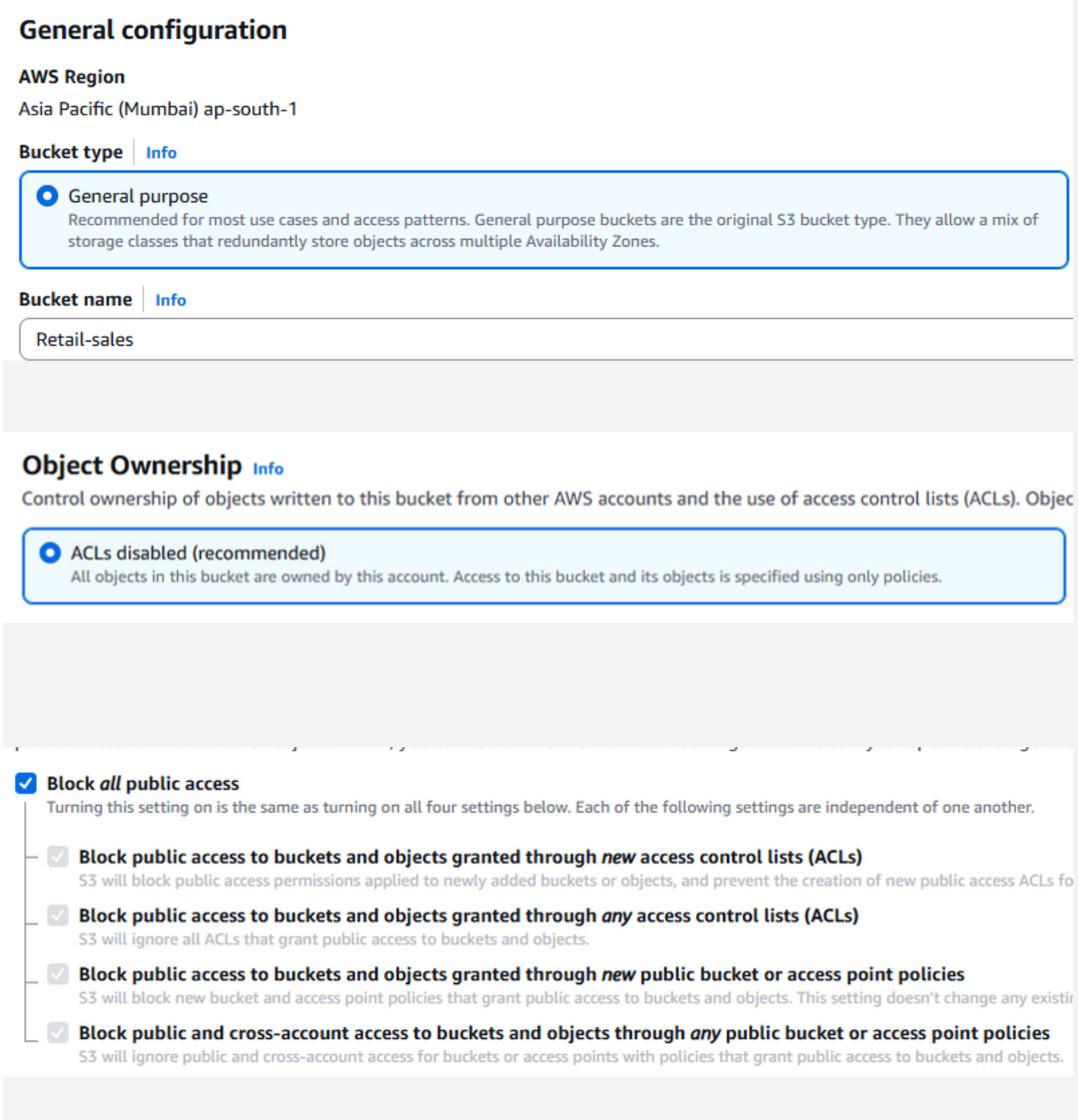
RETAIL REVENUE PREDICTION

DATA ANALYSIS USING THE AWS SERVICES

Presented By:

Himanshi Saini

Create a Bucket



We have created our bucket

Bucket Versioning

- (•) Disable
- ( ) Enable

Default encryption Info

Server-side encryption is automatically applied to new objects stored in this bucket.

Encryption type | Info

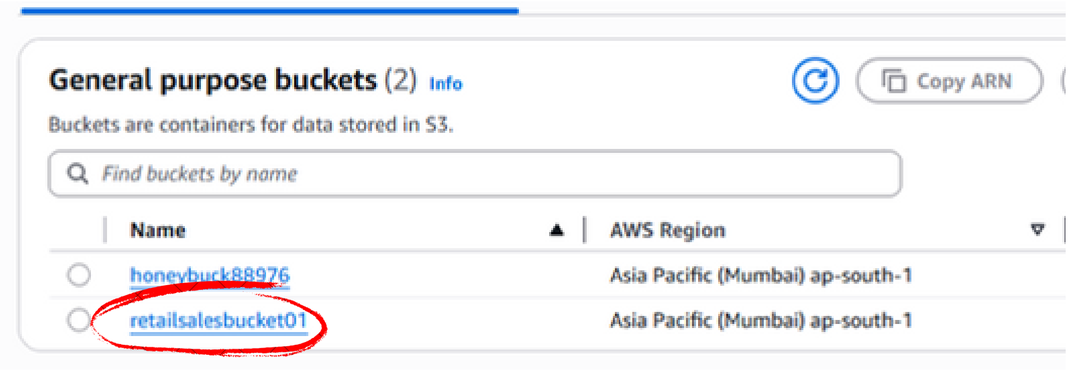
- (•) Server-side encryption with Amazon S3 managed keys (SSE-S3)
- ( ) Server-side encryption with AWS Key Management Service keys (SSE-KMS)
- ( ) Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)
  Secure your objects with two separate layers of encryption. For details on pricing, see **DSSE-KMS pricing**

Bucket Key

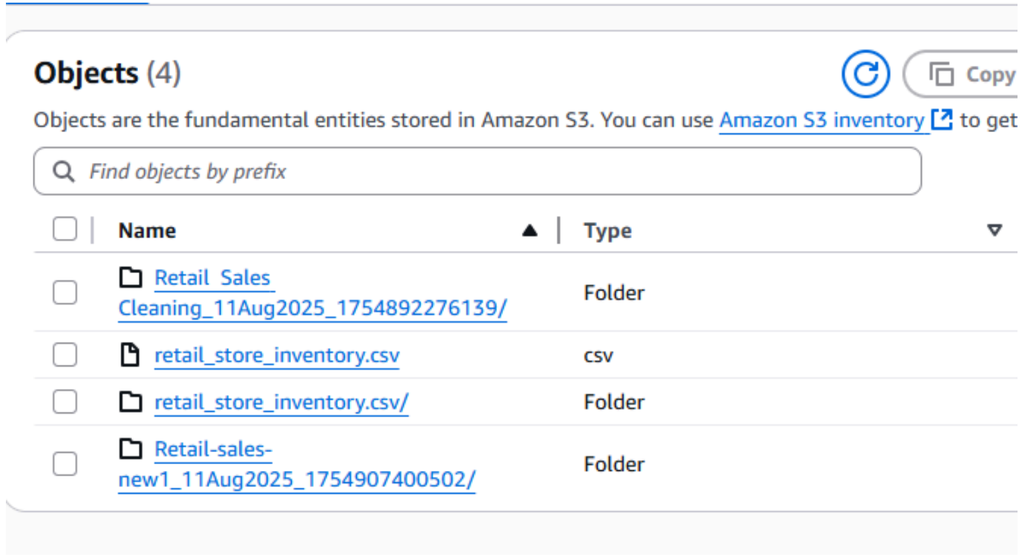
Using an S3 Bucket Key

- ( ) Disable
- (•) Enable

Cancel **Create bucket**



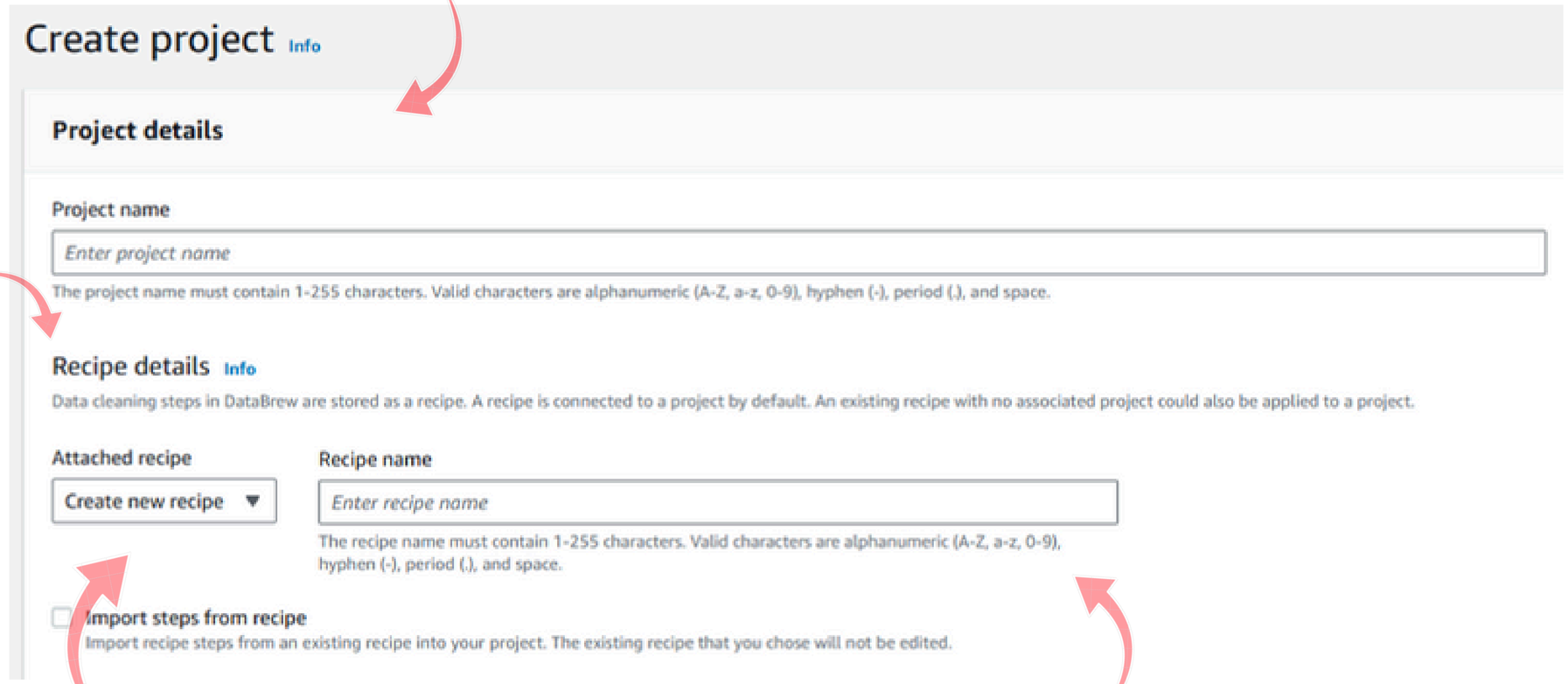
We have prepared a bucket in which we can upload our data files (CSV, Excel, etc.). Whenever we need the files, we simply mention their location and use them.



Create Project in Glue Data Brew

The unique identifier for your DataBrew project (must follow naming rules)

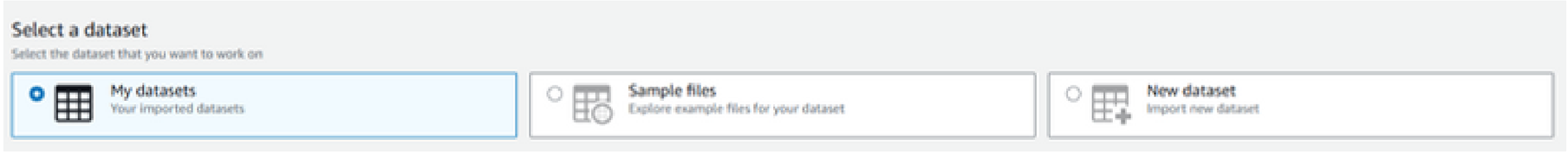
Explanation that cleaning/transform steps are stored as a recipe, which is linked to your project.



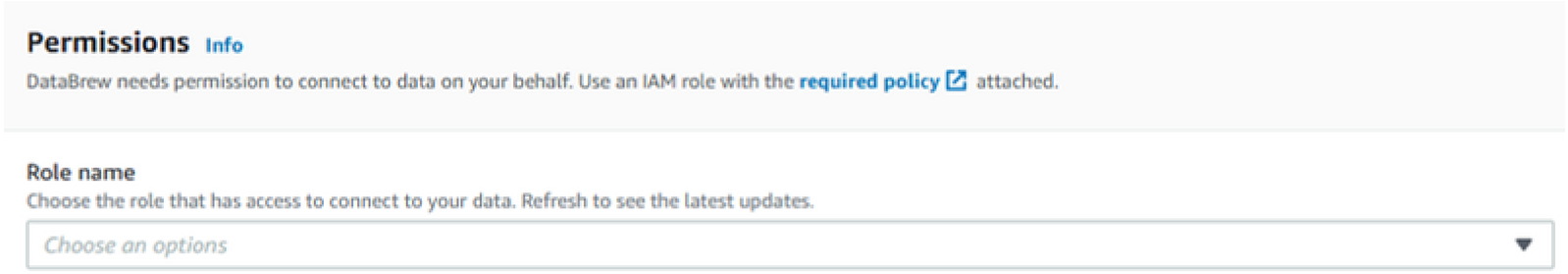
Option to build a brand-new recipe for this project.

The name for your recipe (must follow naming rules).

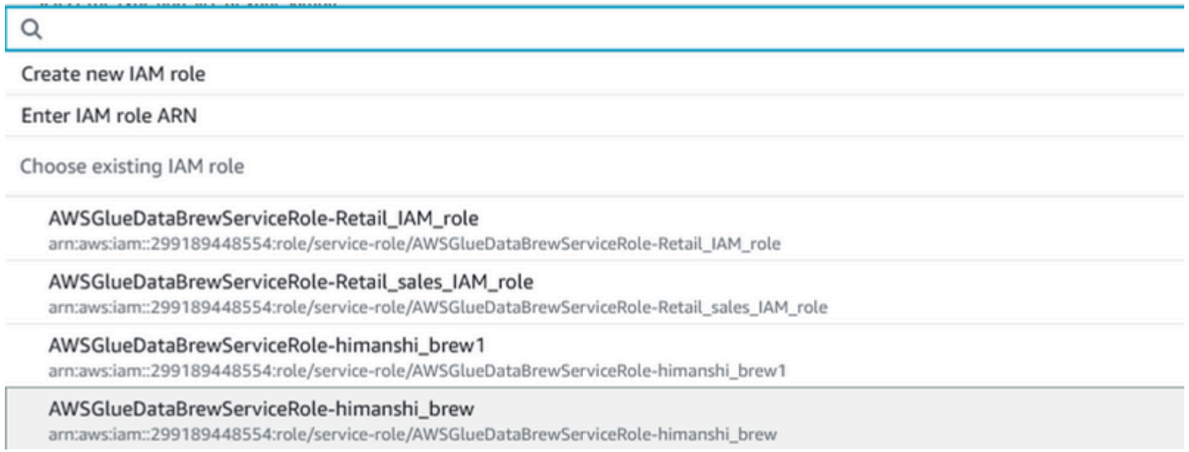
Choose a dataset option from where you want to select your dataset



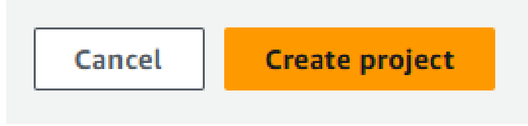
Select the IAM Role if already made, or create a new IAM Role



Like this, as shown below:

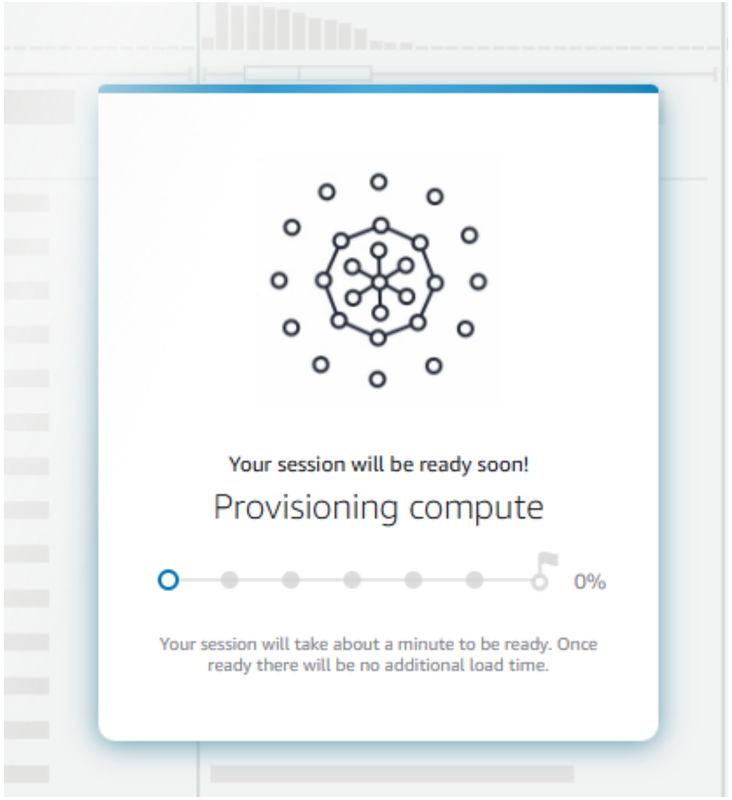


Final Click to create project

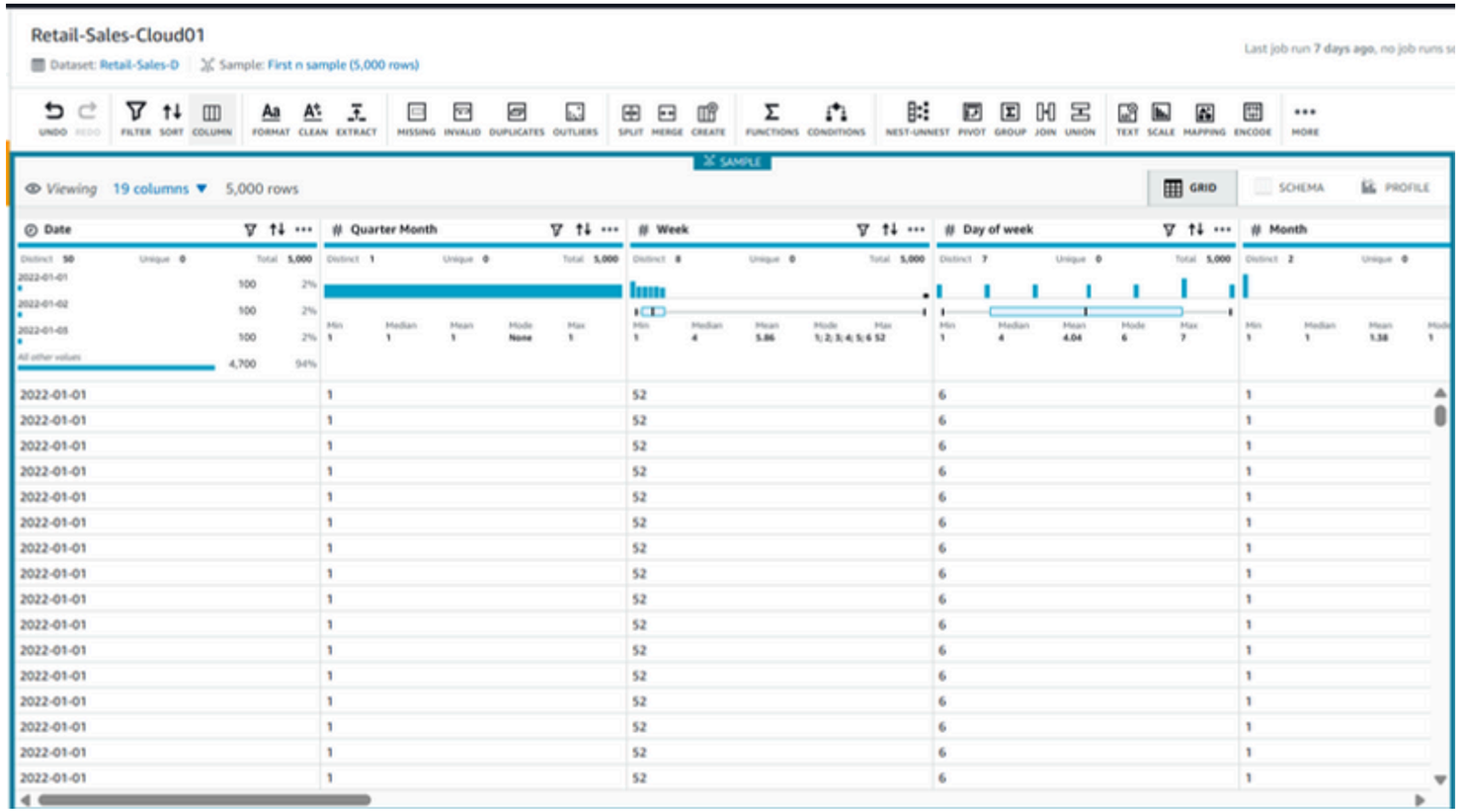


Handling Data

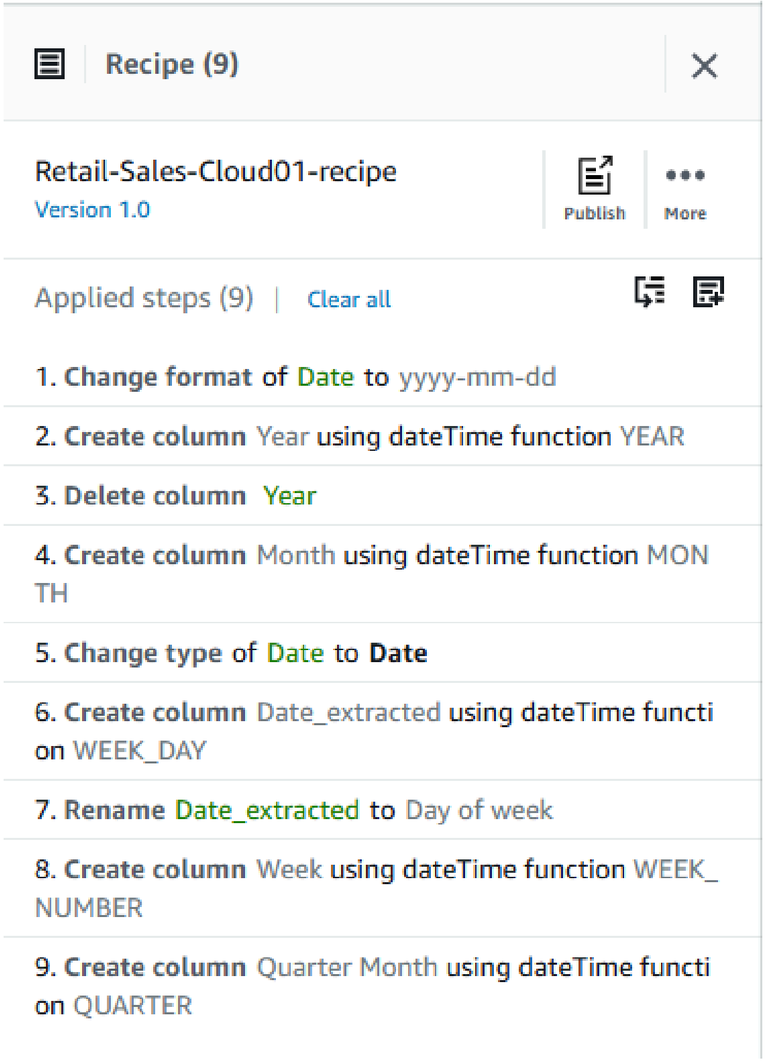
Our Project is getting ready to be launched for Handling Inconsistencies and for data cleaning



This is the final page after loading, and here we will handle our data



We have performed several steps known as recipe to clean the data, preparing it for analysis and manipulation



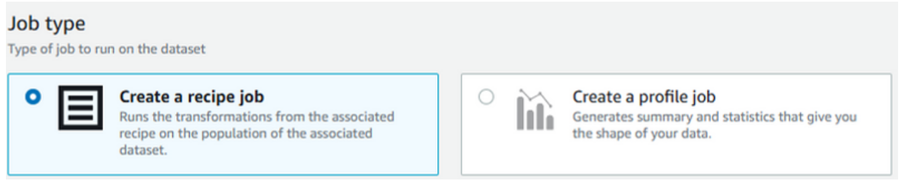
These steps will not be applied to your entire dataset; they will only be implemented on your custom sample dataset. To apply these cleaning steps to the whole dataset, you need to create a new option called "Job." This option will enable you to apply all the cleaning steps to the entire dataset.

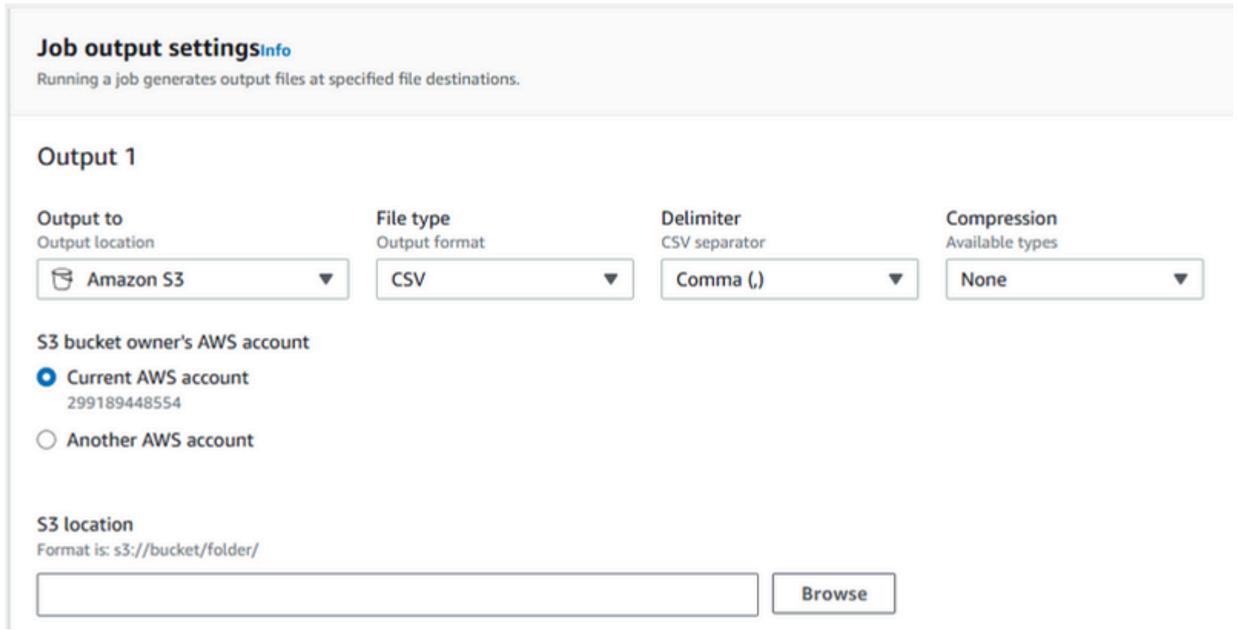
What is Job and how to Create Job

Name the “Job”

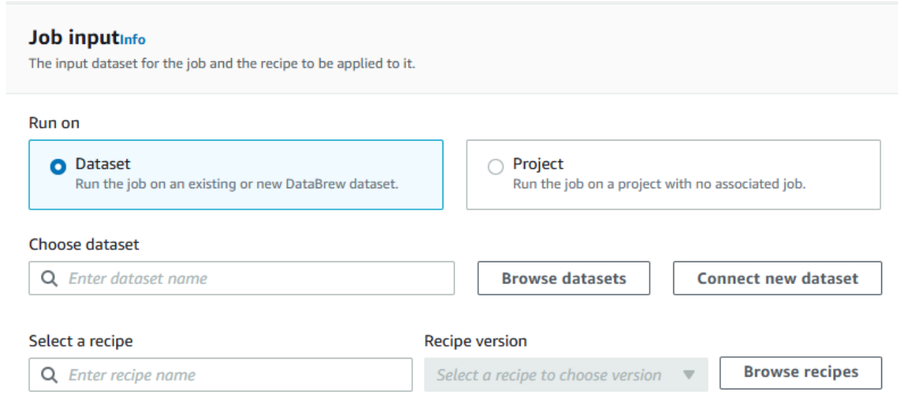
In AWS Glue DataBrew, a job function is the process that runs either data transformations (recipe job) or data profiling (profile job) on your dataset.

Applies your recipe’s cleaning/transformation steps to the full dataset for “RECIPE JOB” and”PROFILE JOB” for Analyzes your dataset to generate statistics, summaries, and data quality insights.





Choose the dataset on which you want to run all these steps of cleaning and browse the file stores in the S3



Choose the settings and options where you want your output to stored

Output After Running Job

01

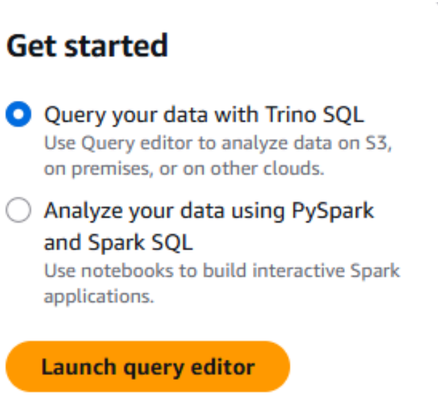
The processed output (from a recipe job) is stored in the Amazon S3 bucket you specify during job setup.

02

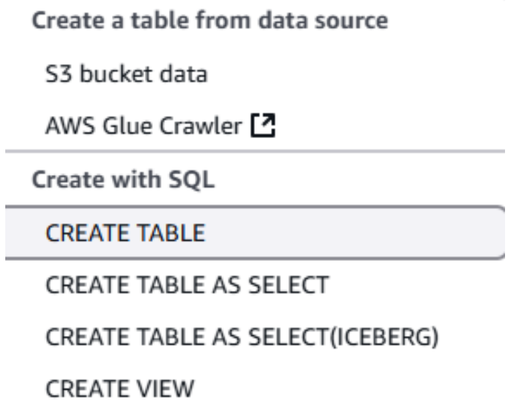
Then, you load the data into Athena for querying the database and conducting data analysis to understand it better.

Working on Athena for Querying the Data Base

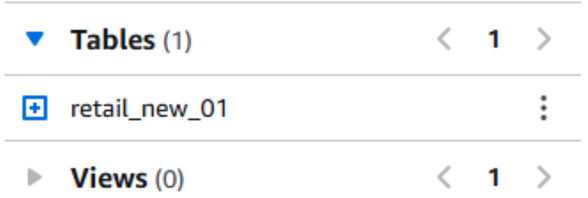
1 .Launch the query editor



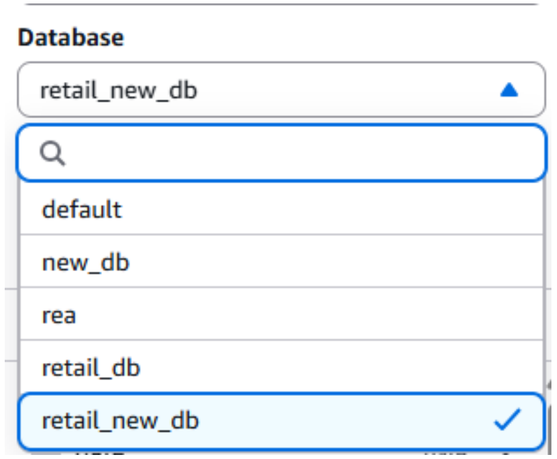
2. Then we load the data from the S3 bucket or we can create db manually by option



3. Here we can see Tables created



3. Choose the db you want to work on



Query 1 | Query 2 | Query 3 | Query 4 | Query 5 | Query 6 | Query 7 | Retail_sales_queries

```sql
-- Understand the target variable (Units Sold)


-- What is the total and average sales per day, week, month, and quarter?

-- weekly total sales:
SELECT
    "Week" AS week_num,
    SUM("Units_Sold" * "Price") AS total_sales
FROM retail_new_01
GROUP BY "Week"
ORDER BY week_num;

-- weekly avg sales
WITH cte AS (
    SELECT
        "Week" AS week_num,
        SUM("Units_Sold" * "Price") AS total_sales
    FROM retail_new_01
    GROUP BY "Week"
)
SELECT
```

ATHENA NOTEBOOK

This is the interface of the SQL Query Notebook where we run all queries to analyse the data

DATA ANALYSIS & QUERYING

Using Athena

WHAT IS THE TOTAL AND AVERAGE SALES PER DAY, WEEK, MONTH, AND QUARTER?

Weekly Total Revenue

```sql
SELECT
    "Week" AS week_num,
    SUM("Units_Sold" * "Price") AS revenue
FROM retail_new_01
GROUP BY "Week"
ORDER BY week_num;
```

Monthly Total Revenue

```sql
SELECT
    "month" AS Month_num,
    SUM("Units_Sold" * "Price") AS total_revenue
FROM retail_new_01
GROUP BY "month"
ORDER BY Month_num;
```

Weekly Avg Revenue

```sql
WITH cte AS (
    SELECT
        "Week" AS week_num,
        SUM("Units_Sold" * "Price") AS revenue
    FROM retail_new_01
    GROUP BY "Week"
)
SELECT
    week_num,
    AVG(total_sales) AS avg_revenue
FROM cte
GROUP BY week_num
ORDER BY week_num;
```

Monthly Avg Revenue

```sql
WITH cte AS (
    SELECT
        "month" AS Month_num,
        SUM("Units_Sold" * "Price") AS total_revenue
    FROM retail_new_01
    GROUP BY "month"
)
SELECT
    Month_num,
    AVG(revenue) AS avg_sales
FROM cte
GROUP BY Month_num
ORDER BY Month_num;
```

Quarter Sales

```sql
SELECT
    "quarter_month" AS quarter,
    SUM("Units_Sold" * "Price") AS revenue
FROM retail_new_01
GROUP BY "quarter_month"
ORDER BY quarter;
```

WHICH PRODUCTS GENERATE THE HIGHEST TOTAL REVENUE?

```sql
select product_id, sum(units_sold * price) as total_revenue
from retail_new_01
group by product_id
order by total_revenue desc limit 5;
```

WHICH PRODUCTS GENERATE THE HIGHEST TOTAL REVENUE?

```sql
SELECT
    DATE_FORMAT(date, '%Y-%m') AS year_month,
    SUM(units_sold * price) AS revenue
FROM
    retail_new_01
GROUP BY
    DATE_FORMAT(date, '%Y-%m')
ORDER BY
    DATE_FORMAT(date, '%Y-%m');
```

- Sales tend to be a little lower at the end of the year (November and December).
- Sales are a bit higher in the middle of the year, around June and July.

PRODUCT & STORE INSIGHTS

Which stores have the highest and lowest average daily revenue?

```sql
WITH daily_revenue AS (
  SELECT
    store_id,
    product_id,
    date,
    SUM(Units_Sold * Price) AS total_revenue_per_day
  FROM retail_new_01
  GROUP BY store_id, product_id, date
)
SELECT
  store_id,
  product_id,
  AVG(total_revenue_per_day) AS avg_daily_revenue
FROM daily_revenue
GROUP BY store_id, product_id
ORDER BY avg_daily_revenue DESC;
```

Which product categories or individual products generate the most revenue?

```sql
select category , sum(units_sold * price) as total_revenue
from retail_new_01
group by category
order by total_revenue desc ;
```

How does sales vary by region? Are some regions consistently outperforming others?

```sql
select region , sum(units_sold * price) as total_revenue
from retail_new_01
group by region
order by total_revenue desc ;
```

WHAT'S THE LOW INVENTORY CONSTRAIN REVENUE?

Step 1: See average revenue at different inventory levels

```sql
SELECT
    inventory_level,
    AVG(Units_Sold * price) AS avg_revenue,
    COUNT(*) AS records_count
FROM retail_new_01
GROUP BY inventory_level
ORDER BY inventory_level;
```

Step 2: Group inventory into buckets (if inventory has wide range)

```sql
SELECT
    CASE
        WHEN inventory_level BETWEEN 0 AND 10 THEN '0-10'
        WHEN inventory_level BETWEEN 11 AND 20 THEN '11-20'
        WHEN inventory_level BETWEEN 21 AND 50 THEN '21-50'
        ELSE '50+'
    END AS inventory_bucket,
    AVG(Units_Sold) AS avg_revenue,
    COUNT(*) AS records_count
FROM retail_new_01
GROUP BY
    CASE
        WHEN inventory_level BETWEEN 0 AND 10 THEN '0-10'
        WHEN inventory_level BETWEEN 11 AND 20 THEN '11-20'
        WHEN inventory_level BETWEEN 21 AND 50 THEN '21-50'
        ELSE '50+'
    END
ORDER BY inventory_bucket;
```

How do units ordered compare to actual units sold (sales fulfillment gap)?

```sql
select units_ordered - units_sold as sales_fulfillment_gap
from retail_new_01
order by sales_fulfillment_gap desc ;
```

PRICING & PROMOTIONS

How do price changes?

```sql
SELECT
    CASE
        WHEN Price BETWEEN 0 AND 100 THEN 'Low'
        WHEN Price BETWEEN 101 AND 500 THEN 'Medium'
        ELSE 'High'
    END AS price_bin,
    AVG(units_sold) AS avg_units_sold,
    COUNT(*) AS records_count
FROM retail_new_01
GROUP BY
    CASE
        WHEN Price BETWEEN 0 AND 100 THEN 'Low'
        WHEN Price BETWEEN 101 AND 500 THEN 'Medium'
        ELSE 'High'
    END
ORDER BY price_bin;
```

Combined effect of Price and Discount on Revenue

```sql
SELECT
    CASE
        WHEN Price BETWEEN 0 AND 100 THEN 'Low'
        WHEN Price BETWEEN 101 AND 500 THEN 'Medium'
        ELSE 'High'
    END AS price_bin,
    CASE
        WHEN Discount BETWEEN 0 AND 10 THEN 'Low'
        WHEN Discount BETWEEN 11 AND 30 THEN 'Medium'
        ELSE 'High'
    END AS discount_bin,
    AVG(units_sold) AS avg_units_sold,
    COUNT(*) AS records_count
FROM retail_new_01
GROUP BY
    CASE
        WHEN Price BETWEEN 0 AND 100 THEN 'Low'
        WHEN Price BETWEEN 101 AND 500 THEN 'Medium'
        ELSE 'High'
    END,
    CASE
        WHEN Discount BETWEEN 0 AND 10 THEN 'Low'
        WHEN Discount BETWEEN 11 AND 30 THEN 'Medium'
        ELSE 'High'
    END
ORDER BY price_bin, discount_bin;
```

EXTERNAL FACTORS

How does weather condition influence revenue?

```sql
SELECT
    Weather_Condition,
    COUNT(*) AS num_days,
    SUM(Units_Sold*price) AS total_revenue,
    AVG(Units_Sold*price) AS avg_revenue
FROM retail_new_01
GROUP BY Weather_Condition
ORDER BY avg_revenue DESC;
```

Does holiday/promotion combined with weather or seasonality boost revenue more than any factor alone?

a) Holiday & Weather Interaction

```sql
SELECT
    holiday_promotion,
    weather_condition,
    COUNT(*) AS num_days,
    SUM(units_sold * price) AS total_revenue,
    AVG(units_sold * price ) AS avg_revenue
FROM retail_new_01
GROUP BY holiday_promotion, weather_condition
ORDER BY holiday_promotion, avg_revenue DESC;
```

b) Holiday & Seasonality Interaction

```sql
SELECT
    holiday_promotion,
    Seasonality,
    COUNT(*) AS num_days,
    SUM(units_sold) AS total_revenue,
    AVG(units_sold) AS avg_revenue
FROM retail_new_01
GROUP BY holiday_promotion, Seasonality
ORDER BY holiday_promotion, avg_revenue DESC;
```

DEMAND FORECAST & INVENTORY PLANNING

Inventory Level vs Demand Forecast & Actual sales

```sql
SELECT
    Date,
    store_id,
    Product_ID,
    Inventory_Level,
    Units_Sold,
    Demand_Forecast,
    (Inventory_Level - Units_Sold) AS inventory_remaining,
    (Inventory_Level - Demand_Forecast) AS inventory_vs_forecast
FROM retail_new_01
ORDER BY inventory_remaining ASC;
```

Interaction effects on sales

```sql
SELECT
    Category,
    Region,
    AVG(units_sold) AS avg_units_sold,
    COUNT(*) AS num_records
FROM retail_new_01
GROUP BY Category, Region
ORDER BY Category, Region;
```

Holiday × Discount

```sql
SELECT
    Holiday_Promotion,
    Discount,
    AVG(units_sold) AS avg_units_sold,
    COUNT(*) AS num_records
FROM retail_new_01
GROUP BY Holiday_Promotion, Discount
ORDER BY Holiday_Promotion, Discount;
```

END

01

After querying the data, we can either store it in Amazon Redshift, or we can load the cleaned CSV file and make a model in an IPython notebook file

02

We can also create a dashboard using Amazon QuickSight.

03

But in our case we will load the data in python notebook and build the ml model there and use the power bi for visualization

THANK YOU!

Contact : himanshisaini7802@gmail.com