Project: Building Ingest-Transform-Load Streaming Pipeline

#### link for short youtube video:

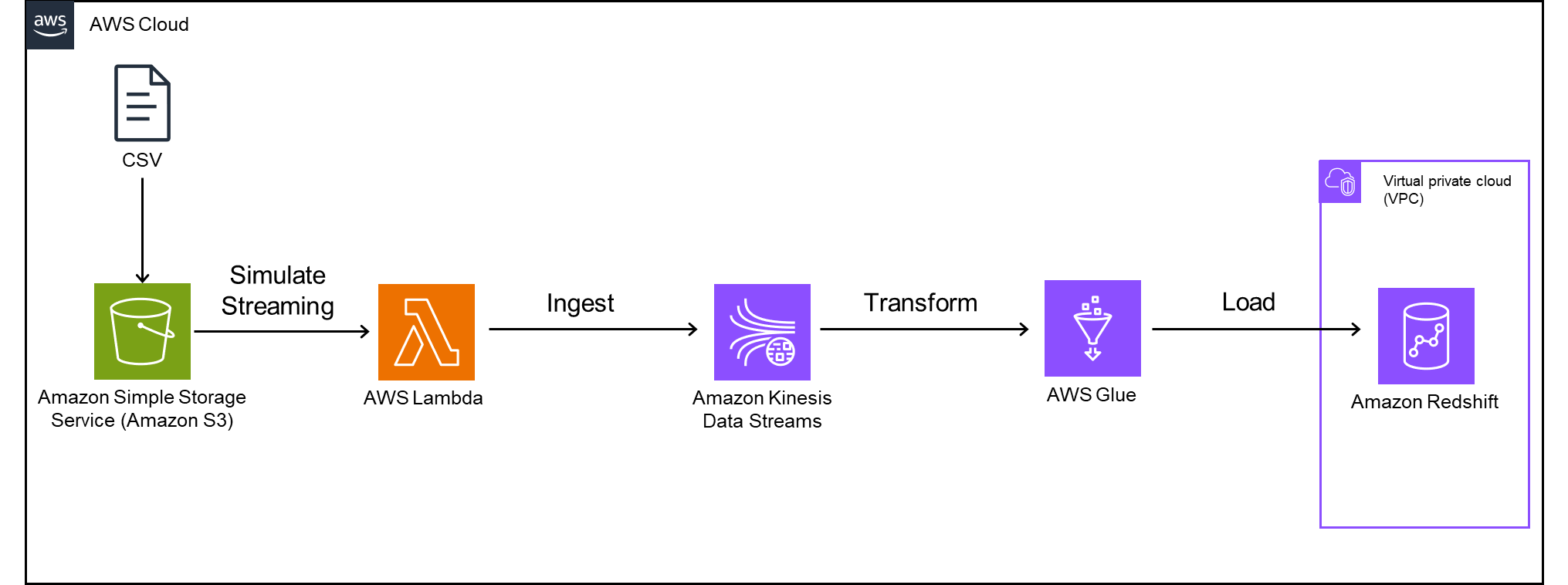
<https://youtu.be/B6mv_ffa0Pc>

#### link for long youtube videos:

<https://youtu.be/p9z2FUVVbIU>

#### Project Overview

Kinesis Data Stream is a service that offers to process data ingestion that is high in volume and velocity from various data sources and deliver to the desired end point. Kinesis Data Stream also offers fault-tolerant and data distribution across shards based on a partition key, which makes it suitable to be used in ETL process for streaming data. In this demo, we are going to demonstrate the use of Kinesis Data Stream in the ingest-transform-load streaming pipeline of three baseball players three months game statistics, including Shohei Ohtani, Betts Mookie, and Freddie Freeman. The statistics CSV file will be saved in S3 bucket. Lambda is used to stream each record to the Kinesis Data Streaming. After ingestion, the data will be transformed in Glue by selecting only the important statistics. After that, data will be loaded into a warehouse, Amazon redshift.

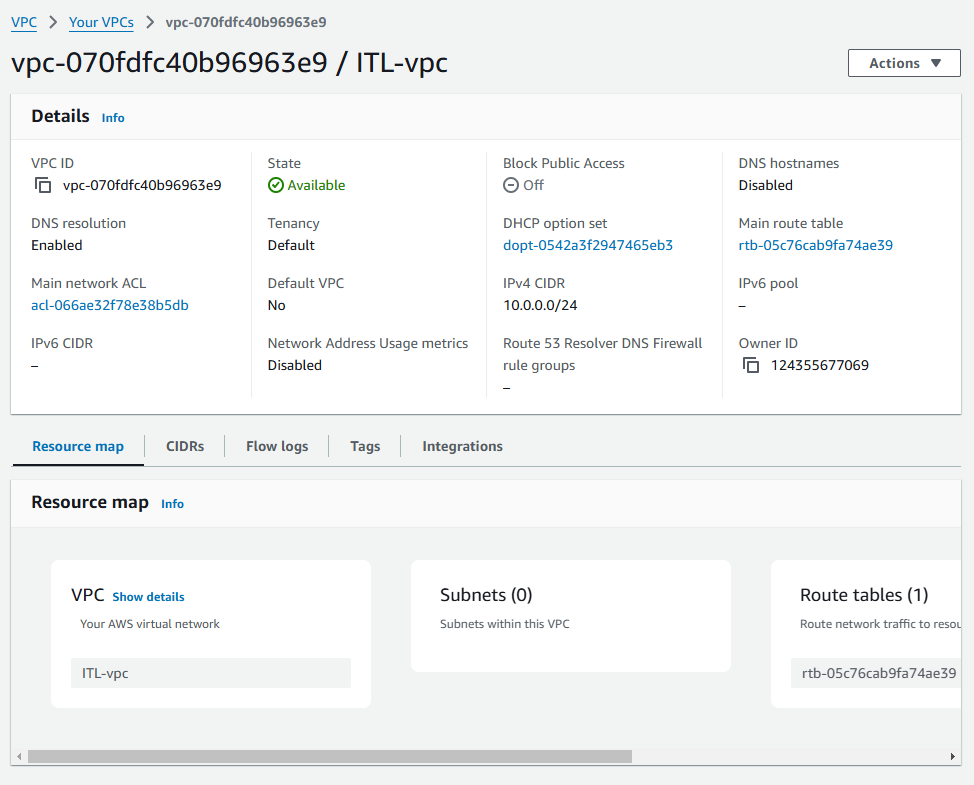


#### Create a Virtual Private Cloud and Private Subnets

The VPC and subnets are created to only host Amazon Redshift

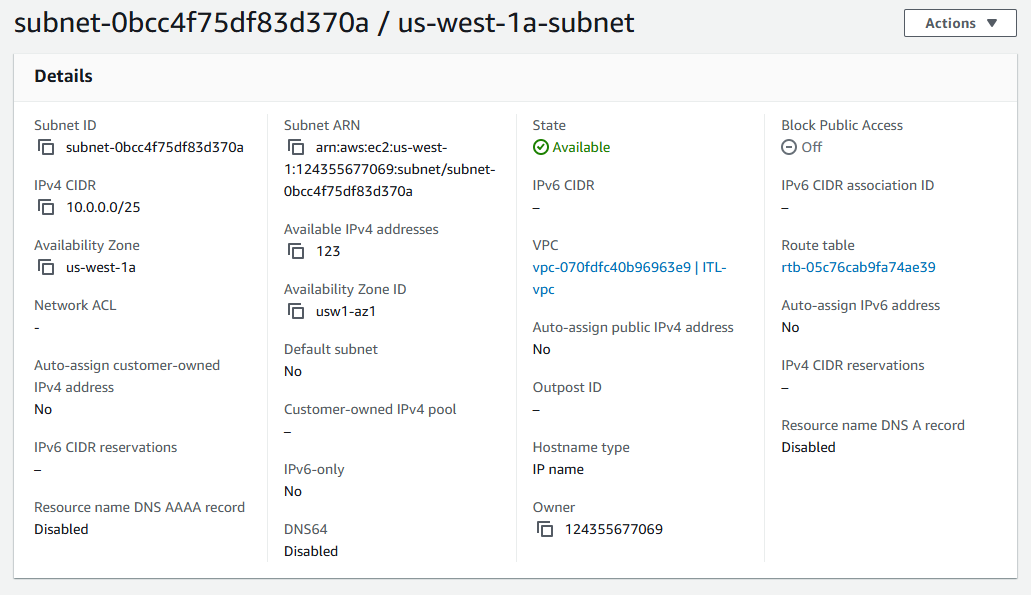
* Go to VPC service in AWS
* Click on Create VPC
* Name it ITL-vpc
* Choose IPv4 CIDR manual input
* For IPv4 CIDR, use 10.0.0.0/24 for amount of 4,096 IPs
* Select No IPv6 CIDR block
* Click create VPC

Your VPC list should look as the following:



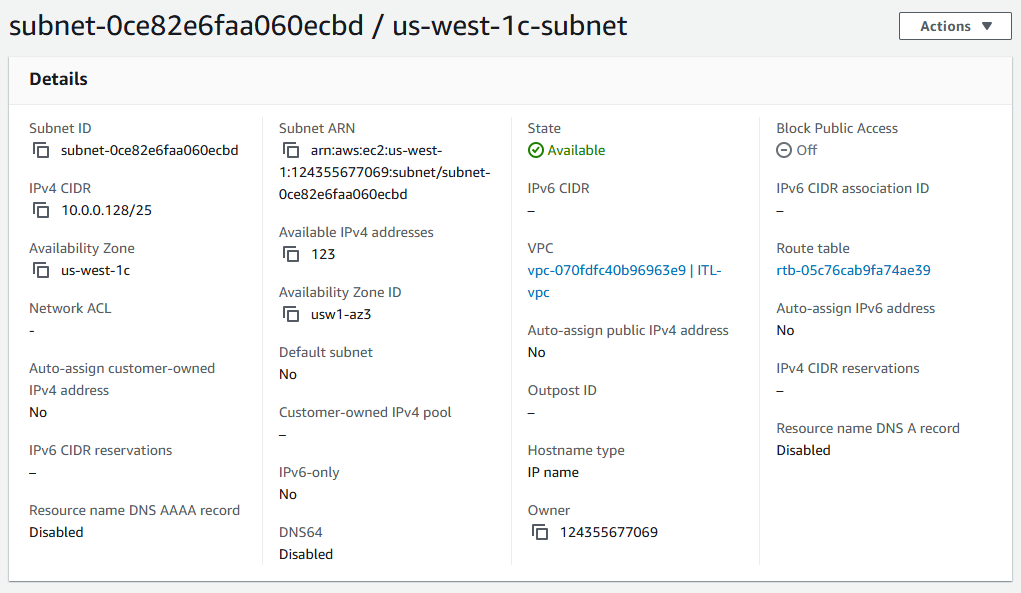
* Go to subnets
* Click create subnet
* Name subnet us-west-1a-subnet
* Choose the Availability Zone in your area. I choose us-west-1a
* Choose 10.0.0.0/24 for IPv4 VPC CIDR block
* Choose 10.0.0.0/25 for IPv4 subnet CIDR block with 128 IP addresses
* Create subnet

Your subnet should look as the following:

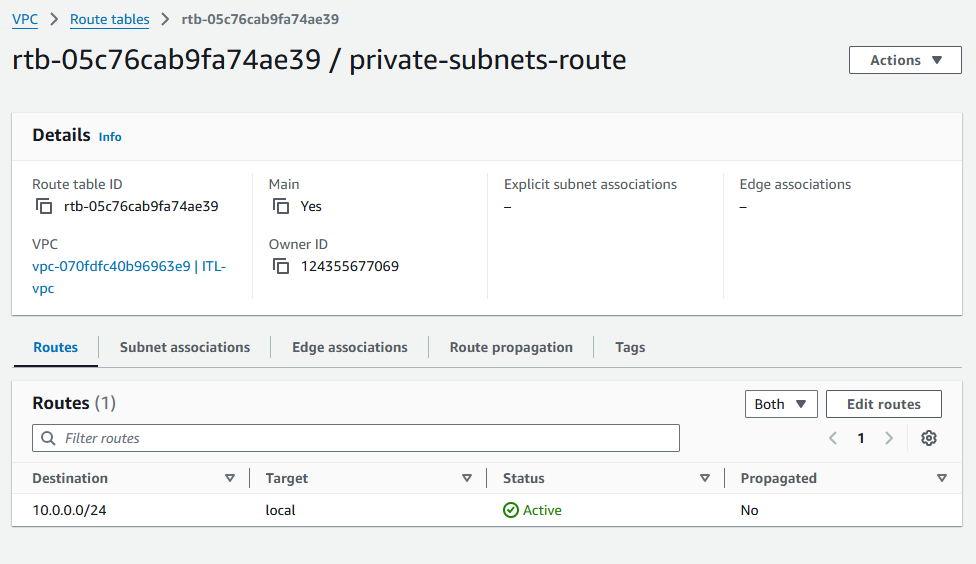


* Create another subnet name us-west-1c-subnet
* Choose the Availability Zone in your area. I choose us-west-1c
* Choose 10.0.0.0/24 for IPv4 VPC CIDR block
* Choose 10.0.0.128/25 for IPv4 subnet CIDR block with 128 IP address

Your subnet should look as the following:



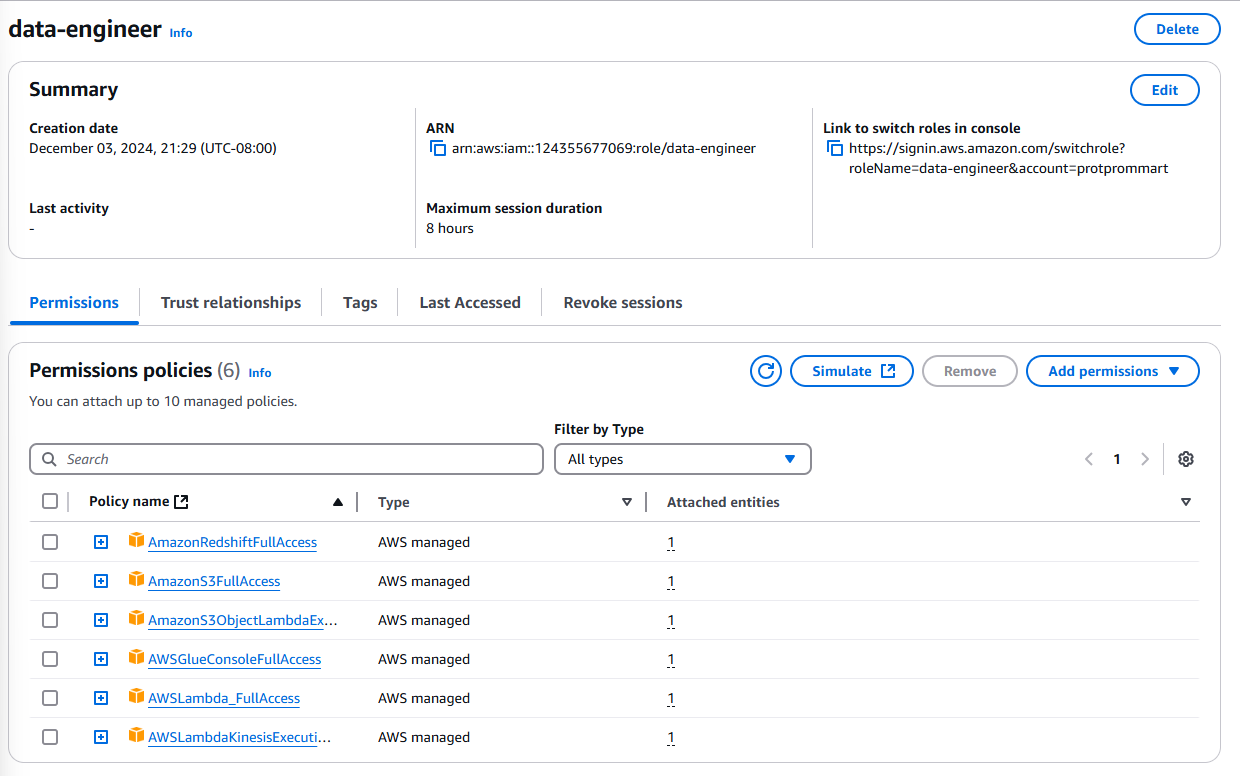
* Check in route tables, there should be a default route tables created
* Name it private-subnets-route
* 10.0.0.0/24 allows routing inside the VPC



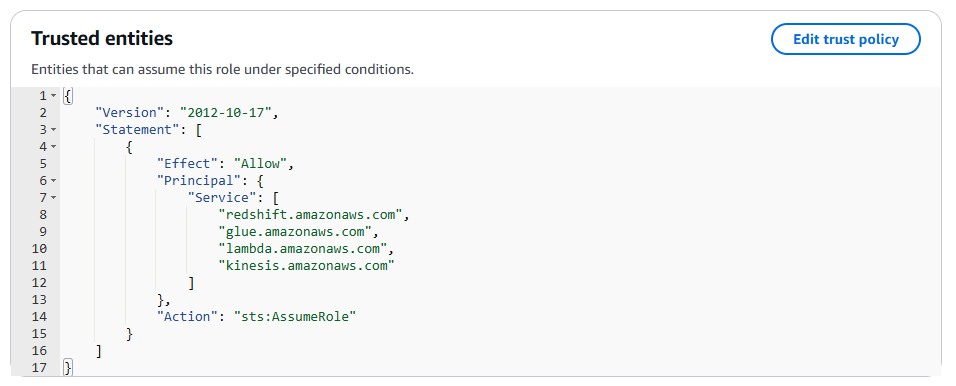
#### Create IAM role for the demo

* Go to IAM and choose roles
* Click create role
* Choose AWS account for select trusted entity and choose this account
* Attach the following policies:
  + AmazonS3ObjectLambdaExecutionRolePolicy for S3 to interact with Lambda
  + AWS Lambda\_fullAccess
  + AWSS3FullAccess
  + AWSLambdaKinesisExecutionRole
  + AmazonKinesisFullAccess
  + AmazonRedshiftFullAccess
  + AWSGlueConsoleFullAccess
* Name Role Name data-engineer
* Create role

You should be able to see your role as this:



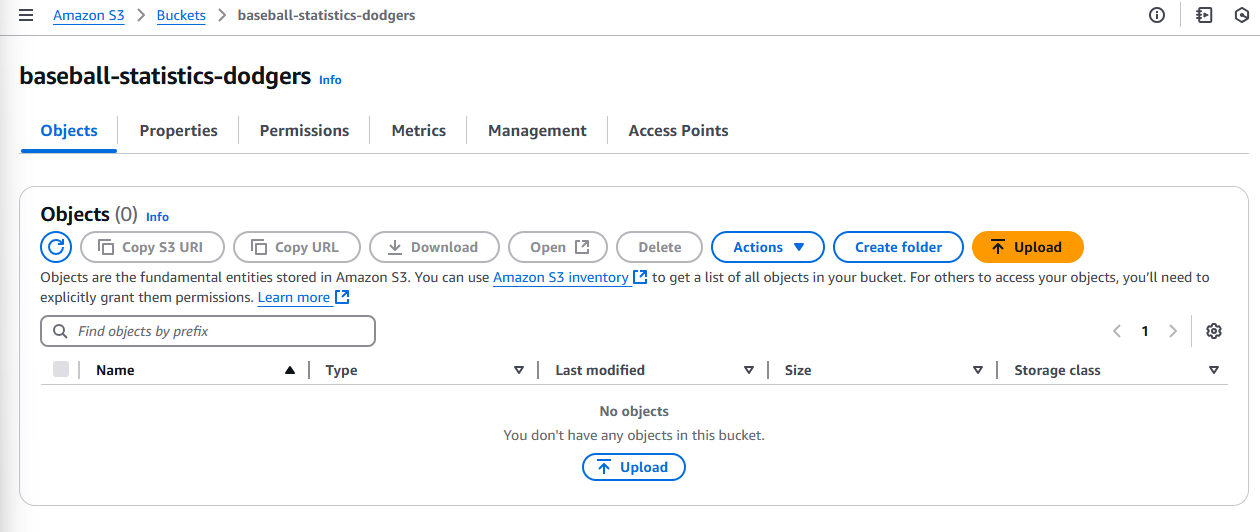
* Once done, go to the trust relationship tab of the data engineer role we created.
* We are going to add services that we want to use this role with including redshift, lambda, glue, kinesis.
* In the json documents, use the following json to establish service trust



#### Create a S3 bucket

* Go to Amazon S3 bucket in AWS
* Click on Create bucket
* Name the bucket baseball-statistics-dodgers
* Choose ACLs disabled for object ownership
* Select block all public access
* Choose disable bucket Versioning
* Create bucket

You should have a blank bucket like this:



##### Input Baseball Statistics into S3 bucket

I extracted game log statistics from March to May 2024 of LA Dodgers players Shohei Ohtani, Betts Mookie, and Freddie Freeman from LA Dodgers websites:

<https://www.mlb.com/player/freddie-freeman-518692?season=2024&team=119&stats=gamelogs-r-hitting-mlb&year=2024>

<https://www.mlb.com/player/shohei-ohtani-660271?stats=gamelogs-r-hitting-mlb&year=2024>

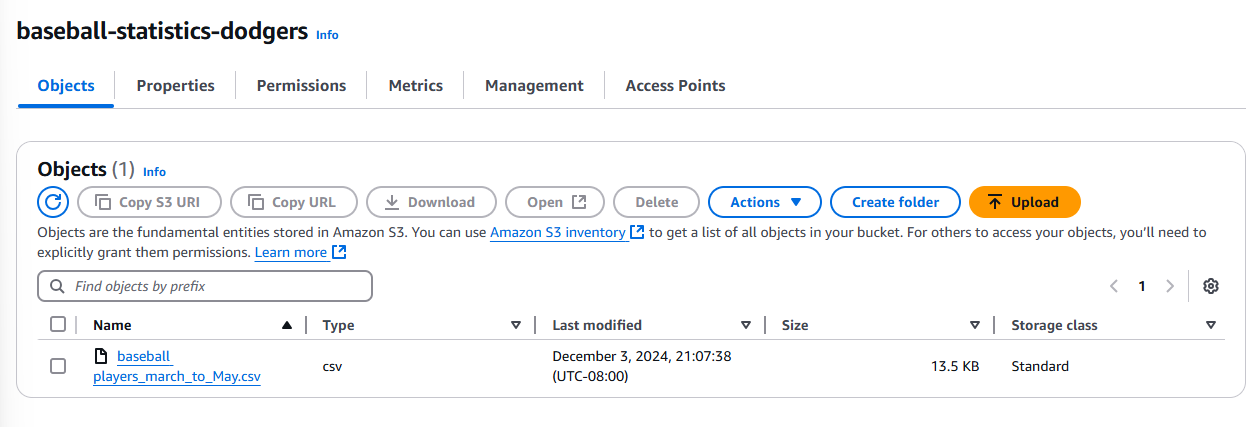
<https://www.mlb.com/player/mookie-betts-605141?season=2024&team=119&stats=gamelogs-r-hitting-mlb&year=2024>

The csv file of the game log statistics from March to May 2024 can be found in my github repository link:

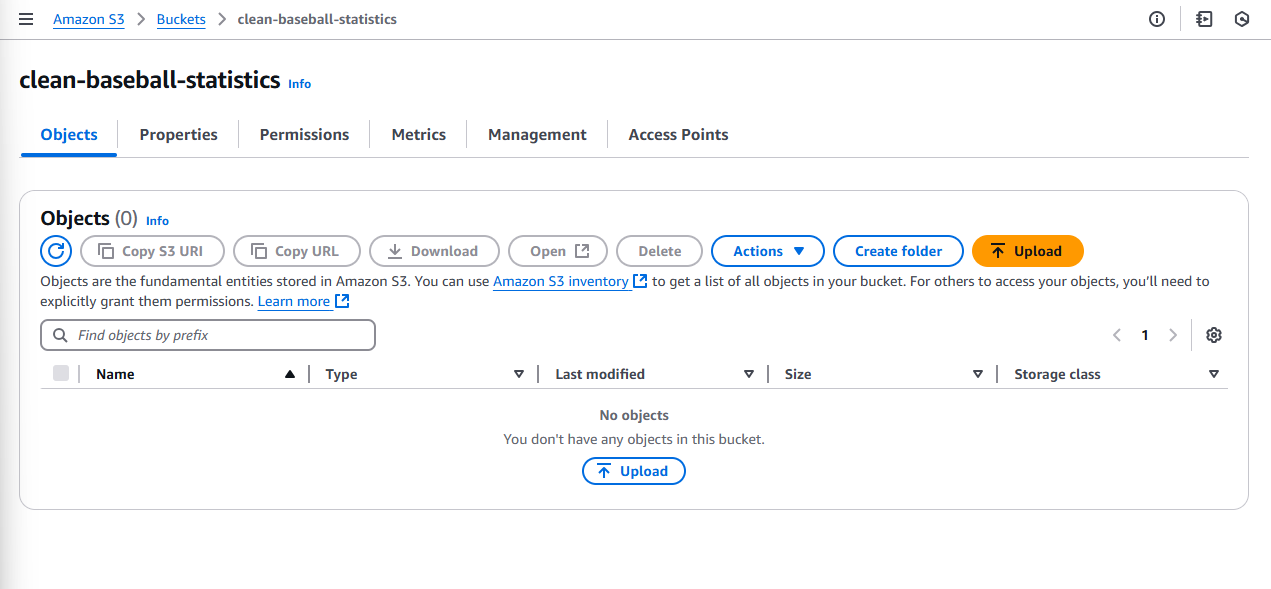
<https://github.com/Protprommart/LA-Dodgers-Cloud-ETL/blob/main/baseball_players_march_to_may.csv>

* Download csv file from the github link
* Upload this file into the baseball-statistics-dodgers S3 bucket

You should be able to see the file in the bucket like this:



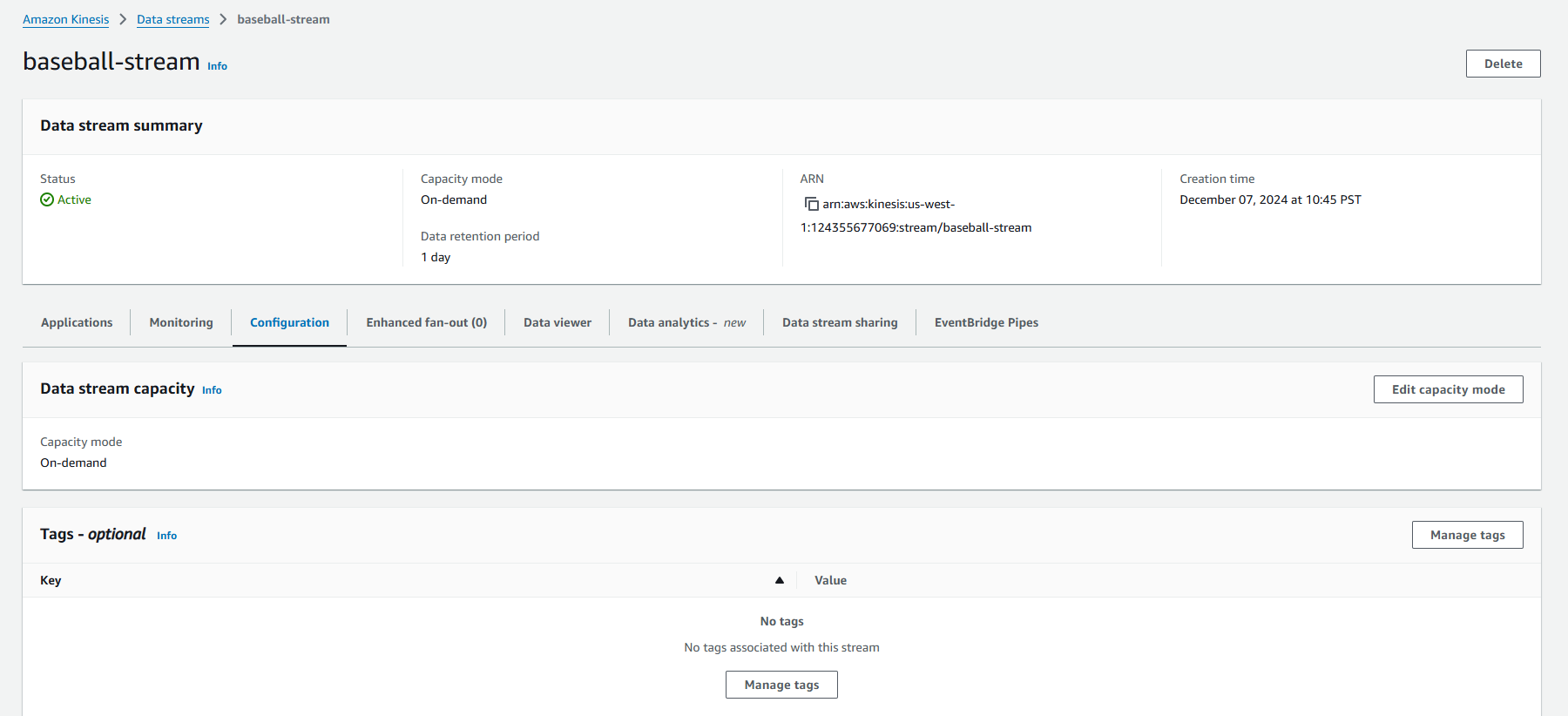
* Create another S3 bucket called clean-baseball-statistics with the same instruction from before. This is going to be a bucket where we put in our transformed dataset after running the data with visual ETL. You should have the bucket as follow:



#### Set Up Kinesis Data Stream and Streaming through Lambda

##### Create a Kinesis Data Stream

* Go to the Amazon Kinesis Console
* Go to Data Streams.
* Create a new stream name baseball-stream
* Leave other setting as default
* Your Kinesis Data Stream should look as following:



##### Create a Lambda Function

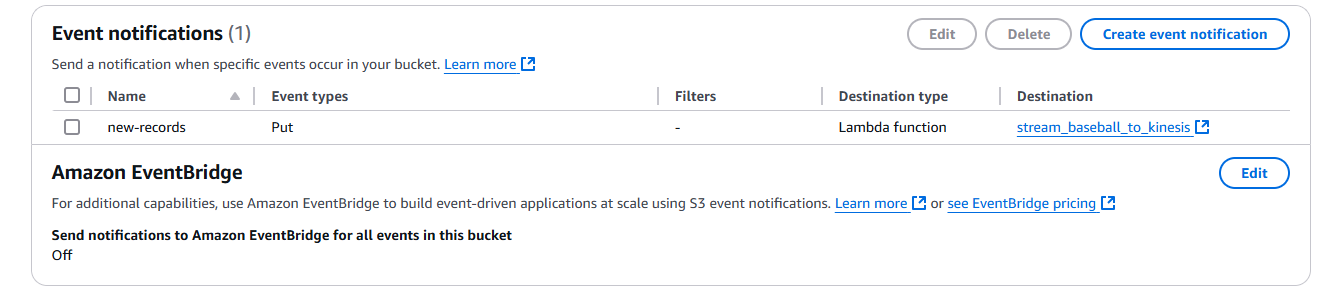
* Navigate to the Lambda Console
* Create Function.
* Select "Author from scratch".
  + Function Name: stream\_baseball\_to\_kinesis.
  + Runtime: Python 3.13.
  + Execution Role: Use the IAM role data engineer
  + Create function
  + Write the Lambda Function Code:  
    Below is the Python example for the Lambda function:
  + Click Deploy button after you finish inputting code

| import boto3  import csv  import json  import os  # Initialize the Kinesis client  kinesis\_client = boto3.client('kinesis')  def lambda\_handler(event, context):  try:  # Get S3 bucket and object key from the event  bucket\_name = event['Records'][0]['s3']['bucket']['name']  object\_key = event['Records'][0]['s3']['object']['key']  print(f"Bucket: {bucket\_name}, Key: {object\_key}")  # Download the file from S3  s3\_client = boto3.client('s3')  response = s3\_client.get\_object(Bucket=bucket\_name, Key=object\_key)  data = response['Body'].read().decode('utf-8-sig').splitlines()  # Parse the CSV file  reader = csv.DictReader(data)  # Stream records to Kinesis  for row in reader:  #skip row for no player and print it out for audit  if not row['Player']:  print(f"Skipping empty row: {row}")  continue  # Convert each row to JSON  record = json.dumps(row)  print(f"Sending record: {record}")  # Send record to Kinesis Data Stream  partition\_key = row['Player']  kinesis\_client.put\_record(  StreamName='baseball-stream',  Data=record,  PartitionKey=partition\_key  )  #verify stream complete  print("Streaming to Kinesis completed.")  #stream fail and reason for failing  except Exception as e:  print(f"Streaming to Kinesis failed: {str(e)}") |
| --- |

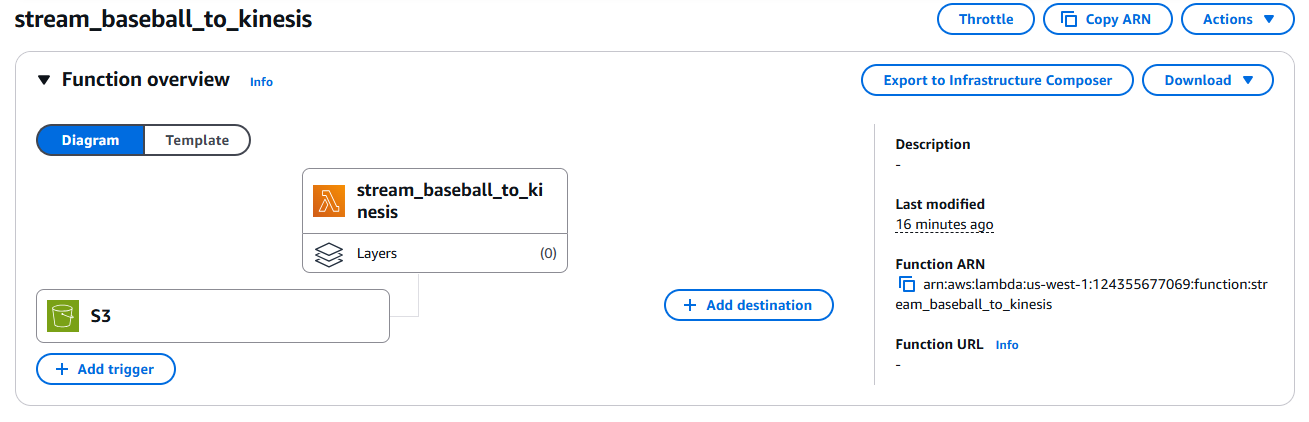
#### 

##### Configure S3 Event Notifications

* Navigate to the S3 bucket in the AWS Management Console.
* Go to the Properties tab and Event Notifications.
* Create a notification as following:
  + Event name: new-records
  + Event Type: PUT (for new file uploads).
  + Destination: Lambda Function (created in the next step).
  + Lambda function: stream\_baseball\_to\_kinesis.
* Your Kinesis S3 Event Notifications should look as following:



* Your Lambda connected to S3 Notification should look as following:

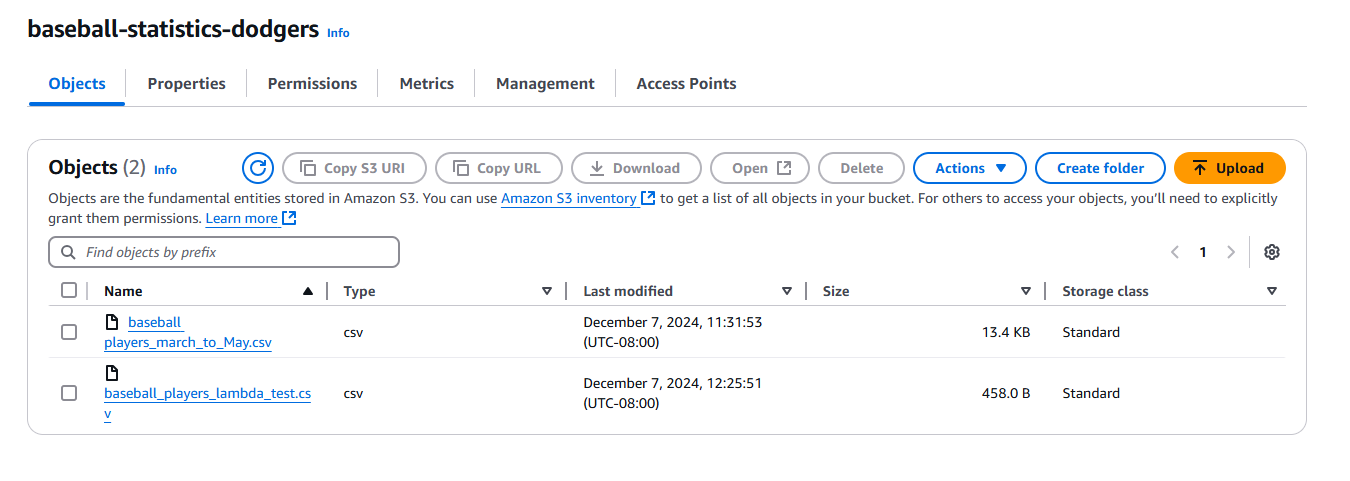


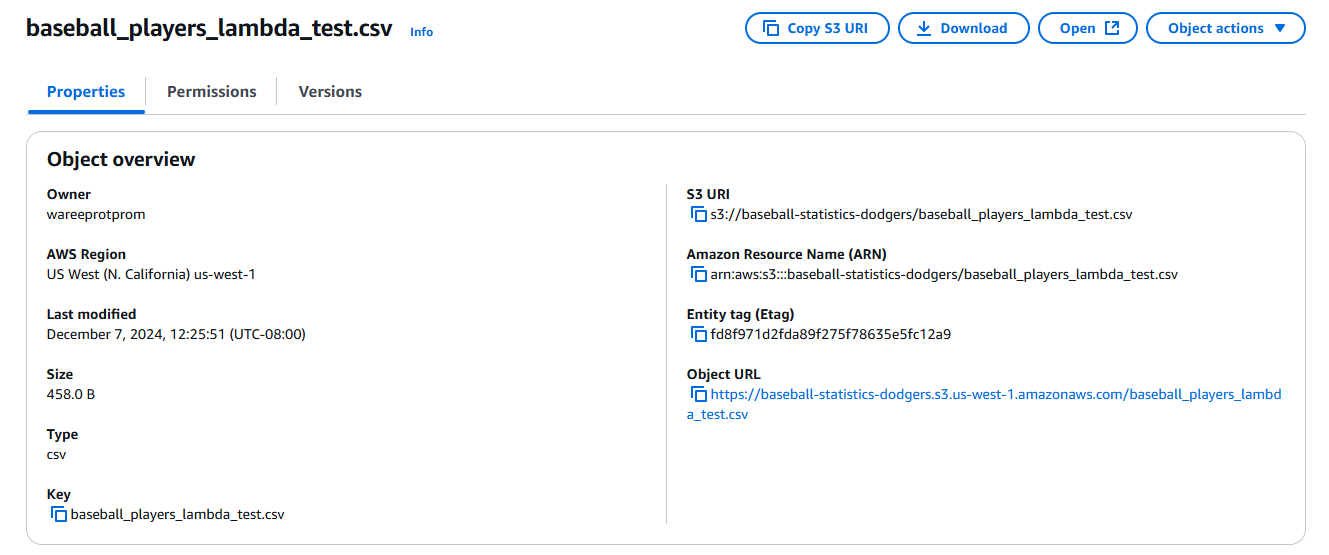
##### Create a Lambda Test Function

The csv file of the game log statistics for lambda testing can be found in my github repository link:

<https://github.com/Protprommart/LA-Dodgers-Cloud-ETL/blob/main/baseball_players_lambda_test.csv>

Upload this csv file into baseball-statistics-dodgers S3 bucket as following:



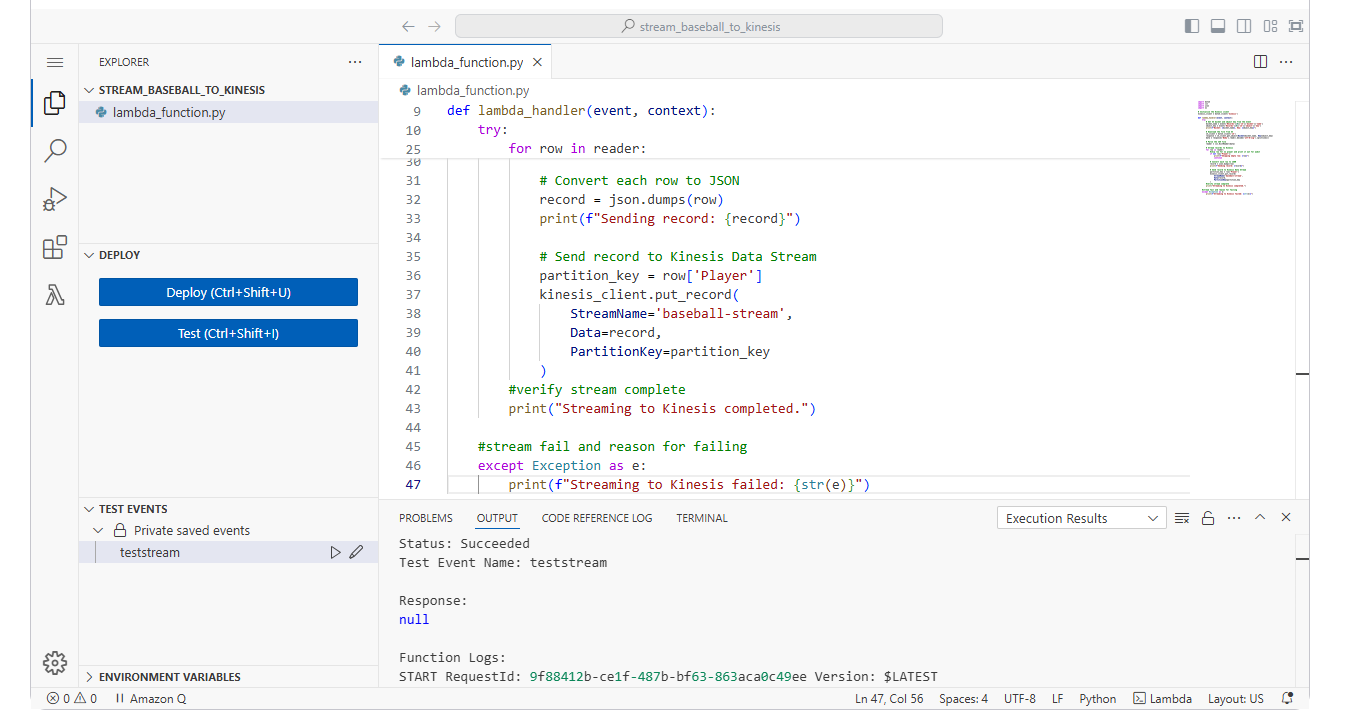


* Click on test button in stream\_baseball\_to\_kinesis lambda function
* Name the test as teststream
* Use the S3 PUT event template.
* Update the event JSON to include:
  + Correct bucket name: baseball-statistics-dodgers.
  + File name (key): baseball\_players\_lambda\_test.csv
  + Put in your awsRegion you are operating from
  + Put in the correct Etag of the object by getting it from S3 bucket object.
* Your test event JSON should look as following:

| {  "Records": [  {  "eventVersion": "2.1",  "eventSource": "aws:s3",  "awsRegion": "us-west-1",  "eventTime": "2023-03-10T15:21:30.000Z",  "eventName": "ObjectCreated:Put",  "s3": {  "s3SchemaVersion": "1.0",  "bucket": {  "name": "baseball-statistics-dodgers",  "arn": "arn:aws:s3:::baseball-statistics-dodgers"  },  "object": {  "key": "baseball\_players\_lambda\_test.csv",  "size": 801.0,  "eTag": "6ced031204f54ad963f10e25556936d0",  "sequencer": "00123456789abcdef"  }  }  }  ]  } |
| --- |

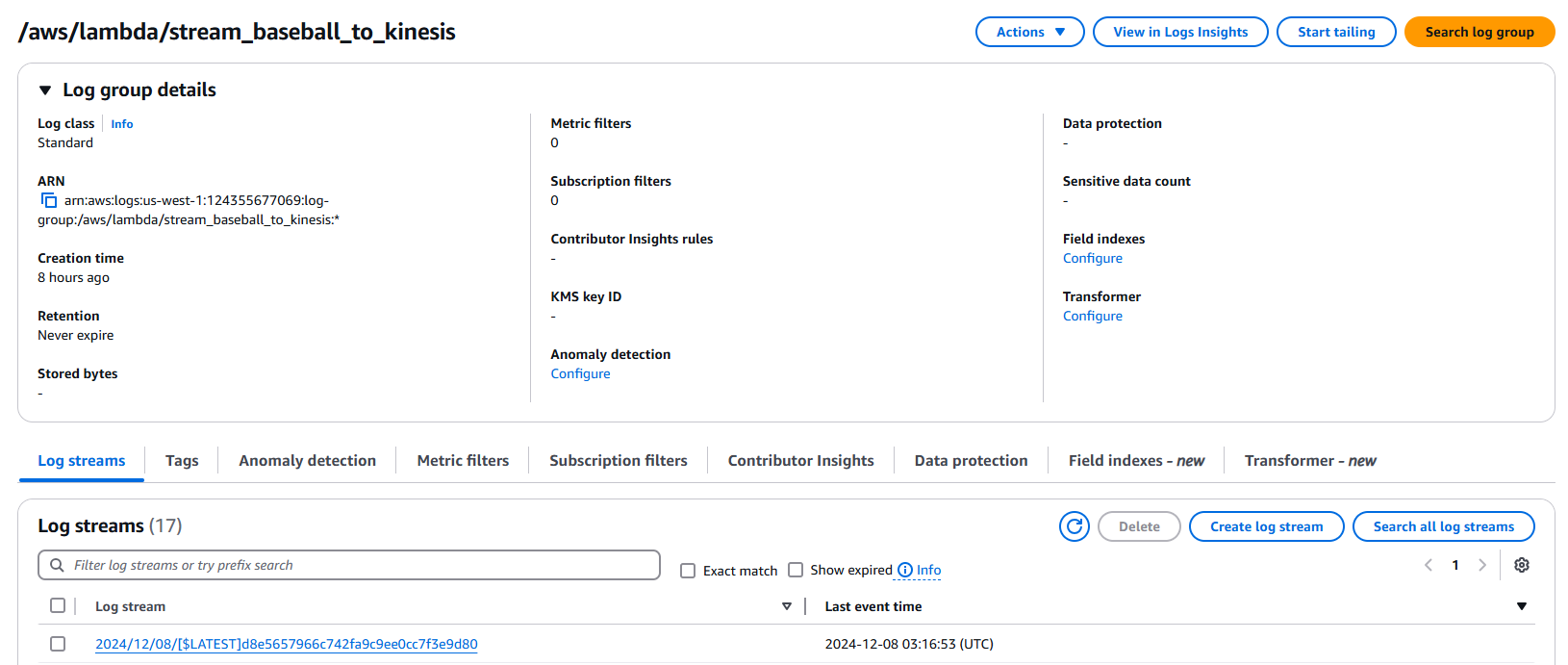
* Run the test code. Your output should be looking as following:

| Status: Succeeded  Test Event Name: teststream  Response:  null  Function Logs:  START RequestId: 9f88412b-ce1f-487b-bf63-863aca0c49ee Version: $LATEST  Bucket: baseball-statistics-dodgers, Key: baseball\_players\_lambda\_test.csv  Sending record: {"Player": "Shohei Ohtani", "Team": "20-Mar", "OPP": "@ SD", "At Bats": "5", "Runs": "0", "Hits": "2", "Total Bases": "2", "2B": "0", "3B": "0", "Home Runs": "0", "RunsBattedIn": "1", "Bases on Balls": "0", "Intentional Walks": "0", "StrikeOut": "0", "StolenBases": "1", "CaughtStealing": "0", "AVG": "0.4", "On-Base Percentage": "0.4", "Slugging Percentage": "0.4", "HitByPitch": "0", "sacrifice bunt": "0", "Sacrifice Flies": "0"}  Sending record: {"Player": "Shohei Ohtani", "Team": "21-Mar", "OPP": "vs SD", "At Bats": "5", "Runs": "1", "Hits": "1", "Total Bases": "1", "2B": "0", "3B": "0", "Home Runs": "0", "RunsBattedIn": "1", "Bases on Balls": "0", "Intentional Walks": "0", "StrikeOut": "0", "StolenBases": "0", "CaughtStealing": "0", "AVG": "0.3", "On-Base Percentage": "0.273", "Slugging Percentage": "0.3", "HitByPitch": "0", "sacrifice bunt": "0", "Sacrifice Flies": "1"}  Sending record: {"Player": "Shohei Ohtani", "Team": "28-Mar", "OPP": "vs STL", "At Bats": "3", "Runs": "1", "Hits": "2", "Total Bases": "3", "2B": "1", "3B": "0", "Home Runs": "0", "RunsBattedIn": "0", "Bases on Balls": "1", "Intentional Walks": "0", "StrikeOut": "1", "StolenBases": "0", "CaughtStealing": "0", "AVG": "0.385", "On-Base Percentage": "0.4", "Slugging Percentage": "0.462", "HitByPitch": "0", "sacrifice bunt": "0", "Sacrifice Flies": "0"}  Streaming to Kinesis completed.  END RequestId: 9f88412b-ce1f-487b-bf63-863aca0c49ee  REPORT RequestId: 9f88412b-ce1f-487b-bf63-863aca0c49ee Duration: 2125.70 ms Billed Duration: 2126 ms Memory Size: 128 MB Max Memory Used: 85 MB Init Duration: 538.16 ms  Request ID: 9f88412b-ce1f-487b-bf63-863aca0c49ee |
| --- |



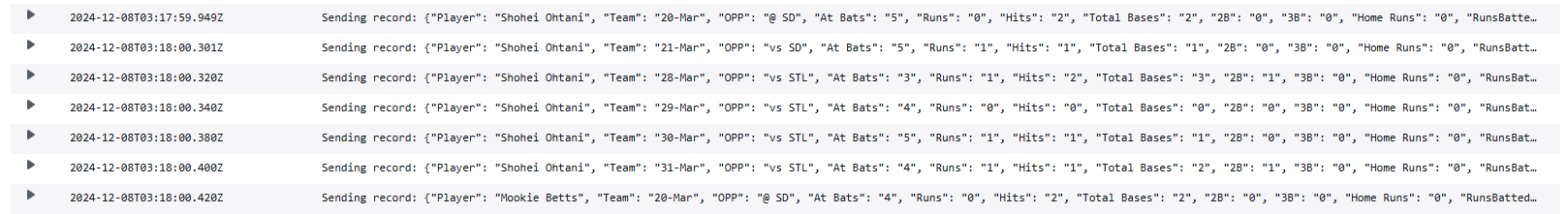
##### Testing Streaming Integration Process

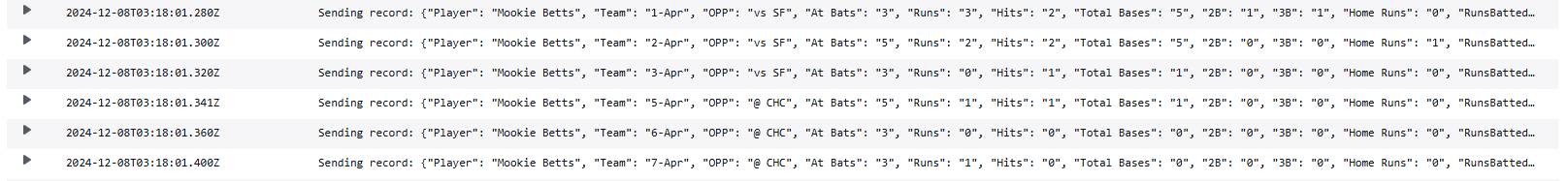
* Reupload baseball\_players\_march\_to\_may.csv data in baseball-statistics-dodgers S3 bucket
* Go to CloudWatch and navigate to Log groups under Logs tab
* Under Log groups find /aws/lambda/stream\_baseball\_to\_kinesis to look at lambda streaming records
* In the log streams, click on the latest log stream



* Check if the stream went successfully. Your log streams should look as following:



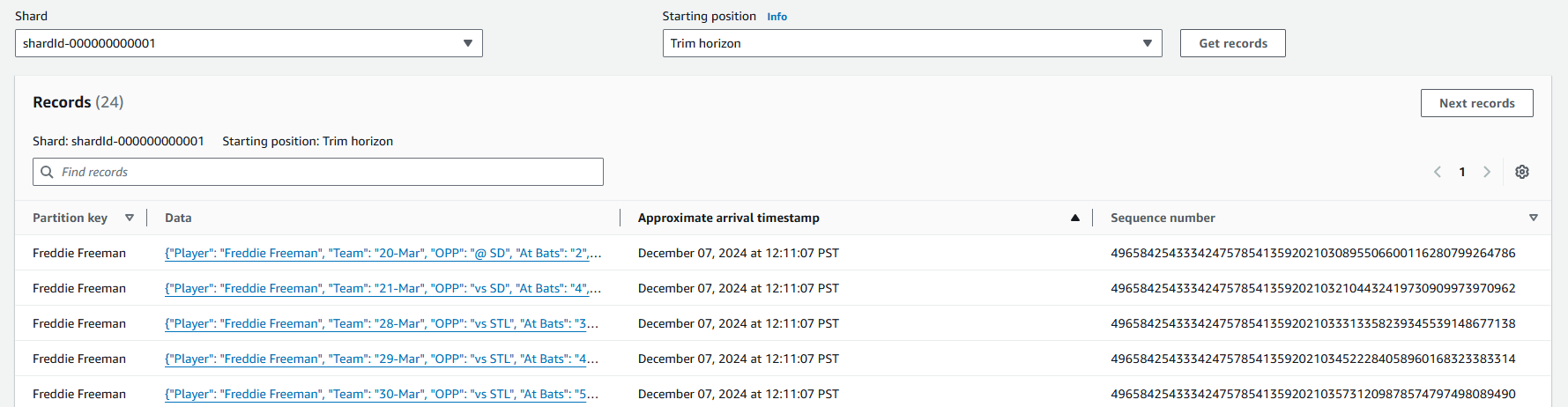




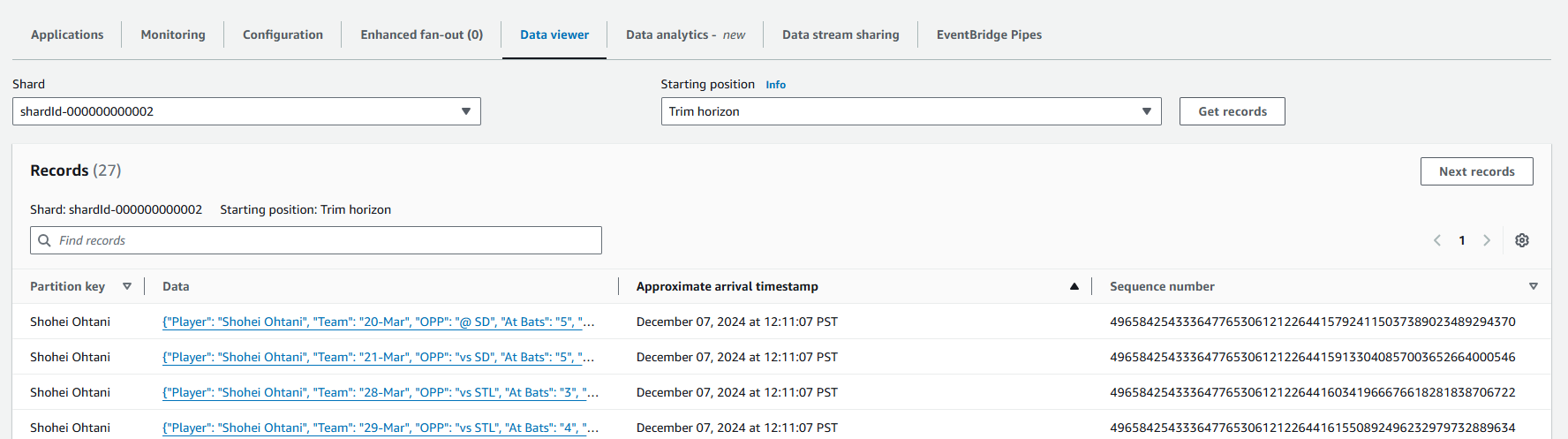
You should be able to see how lambda is streaming to kinesis by organizing according to the baseball player name. This is because we specify in the code that we are going to use the player column as a partition key.

* Go to baseball-stream Kinesis data stream and you can check records coming in to Kinesis under the data viewer tab.
* View Shardid-000000000001 for Freddie Freeman, 000000000002 for Shohei Ohtani, 000000000003 for Mookie Betts

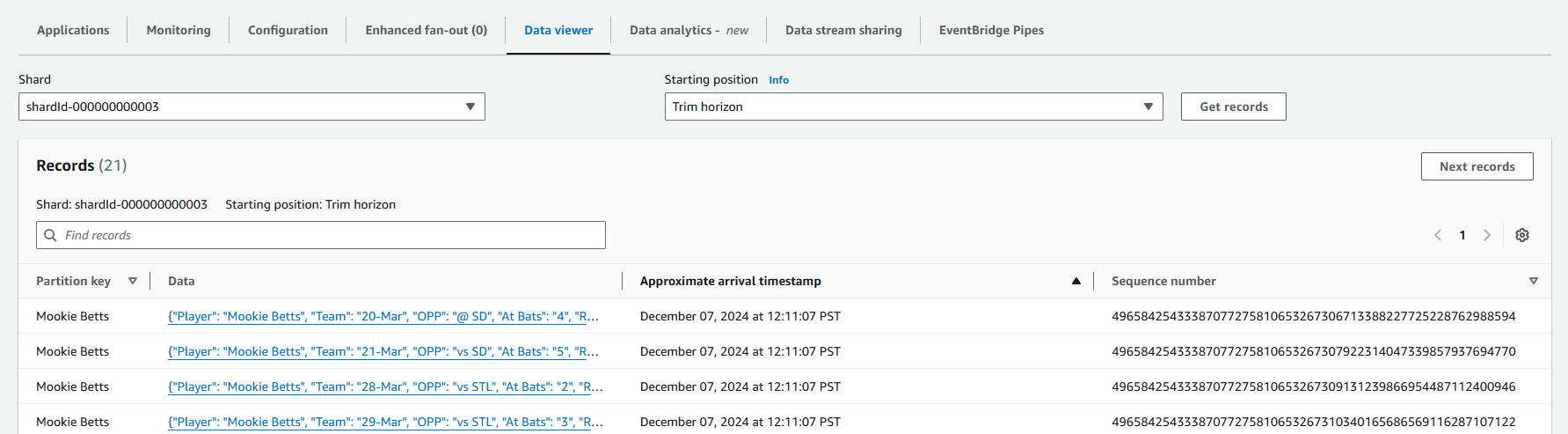
Shardid000000000001



Shardid000000000002



Shardid000000000003

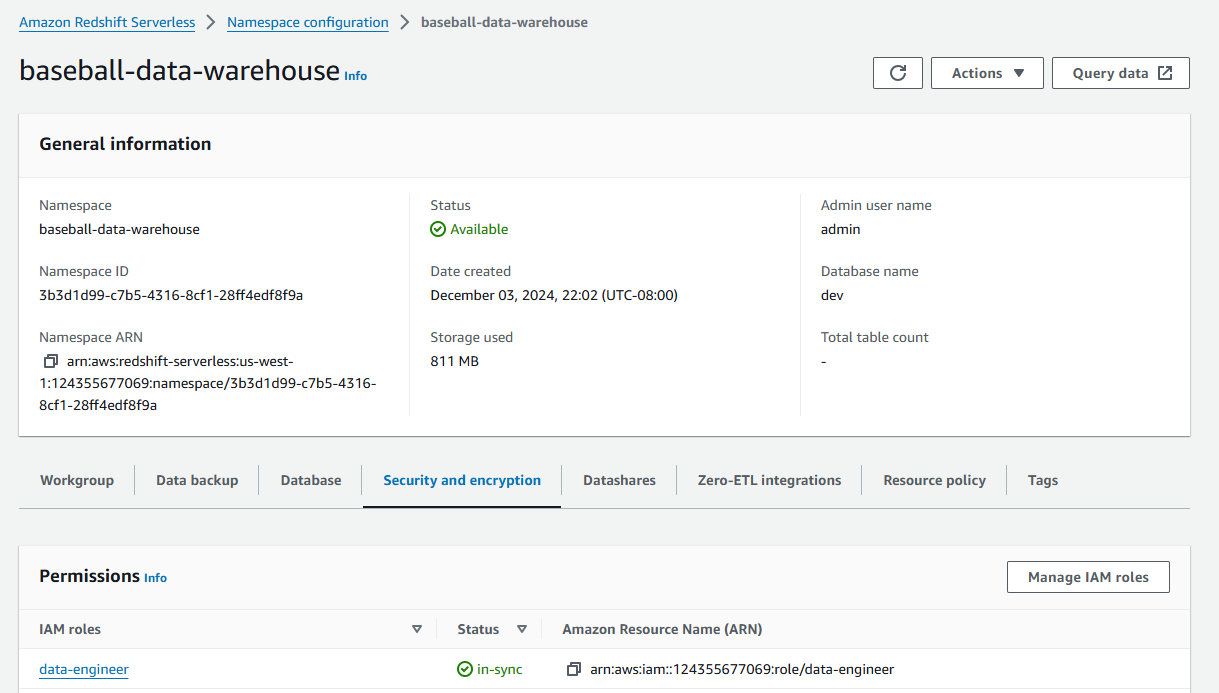


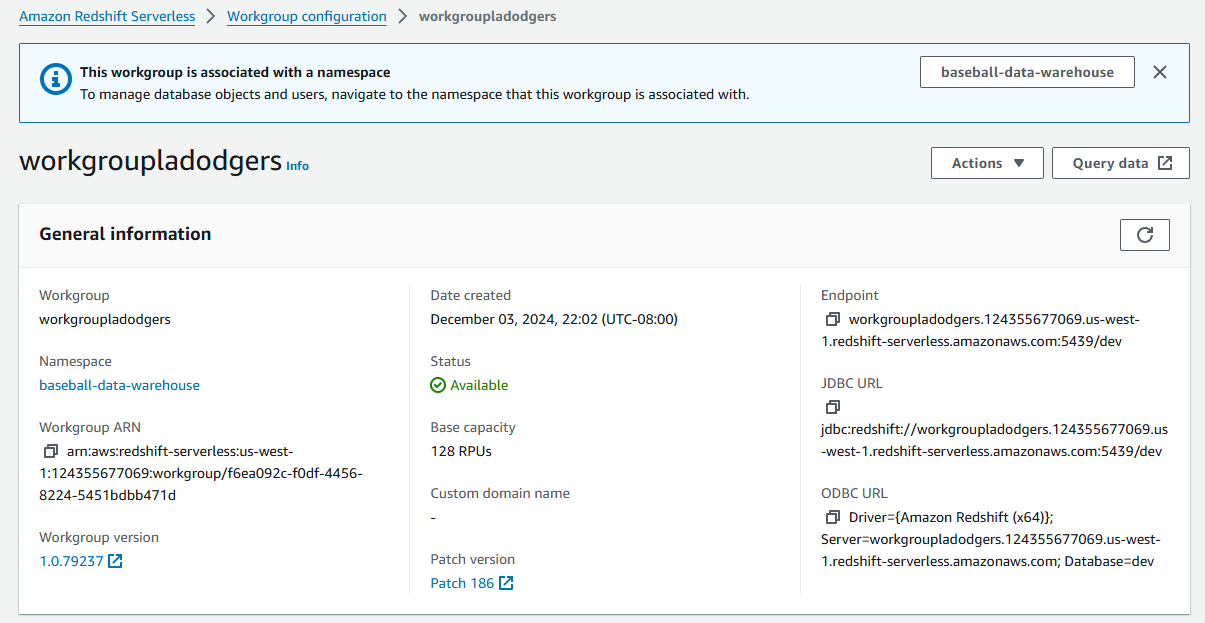
As you can see here that each player's statistics are being separated into each shards.

#### Create a Data Warehouse in Redshift

* Go to Redshift Serverless
* Name target namespace as baseball-data-warehouse. The namespace will host the database
* Name workgroup as workgroupladodgers. Workgroup will host the server
* In VPC choose ITL-vpc that we created
* Choose security group that create as a default with the vpc we created
* Choose private subnets we created for subnet
* Note: there needs to be at least 37 free IP addresses in 2 subnets. Each subnet should be in a different Availability Zone.
* Add data-engineer IAM roles to it.

You should have the following Namespace and workgroup.



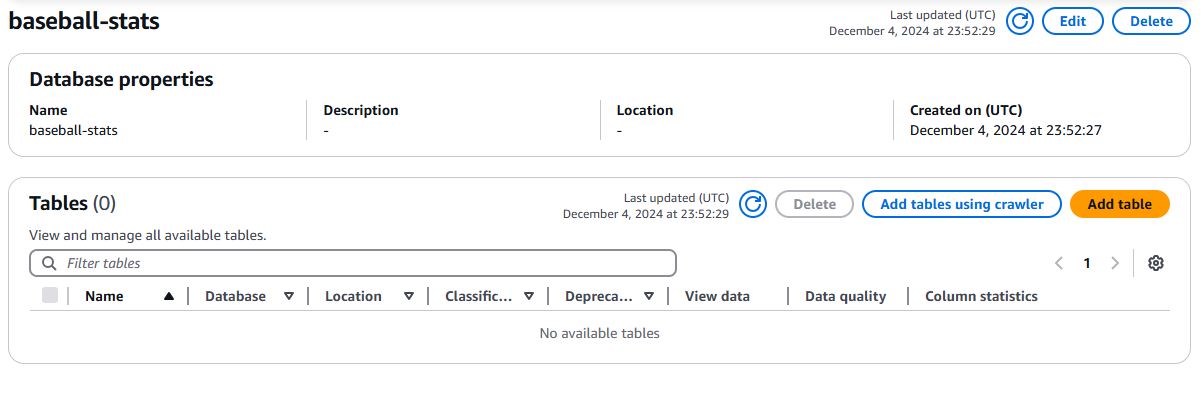


#### Transforming Data with AWS Glue

##### Create Database

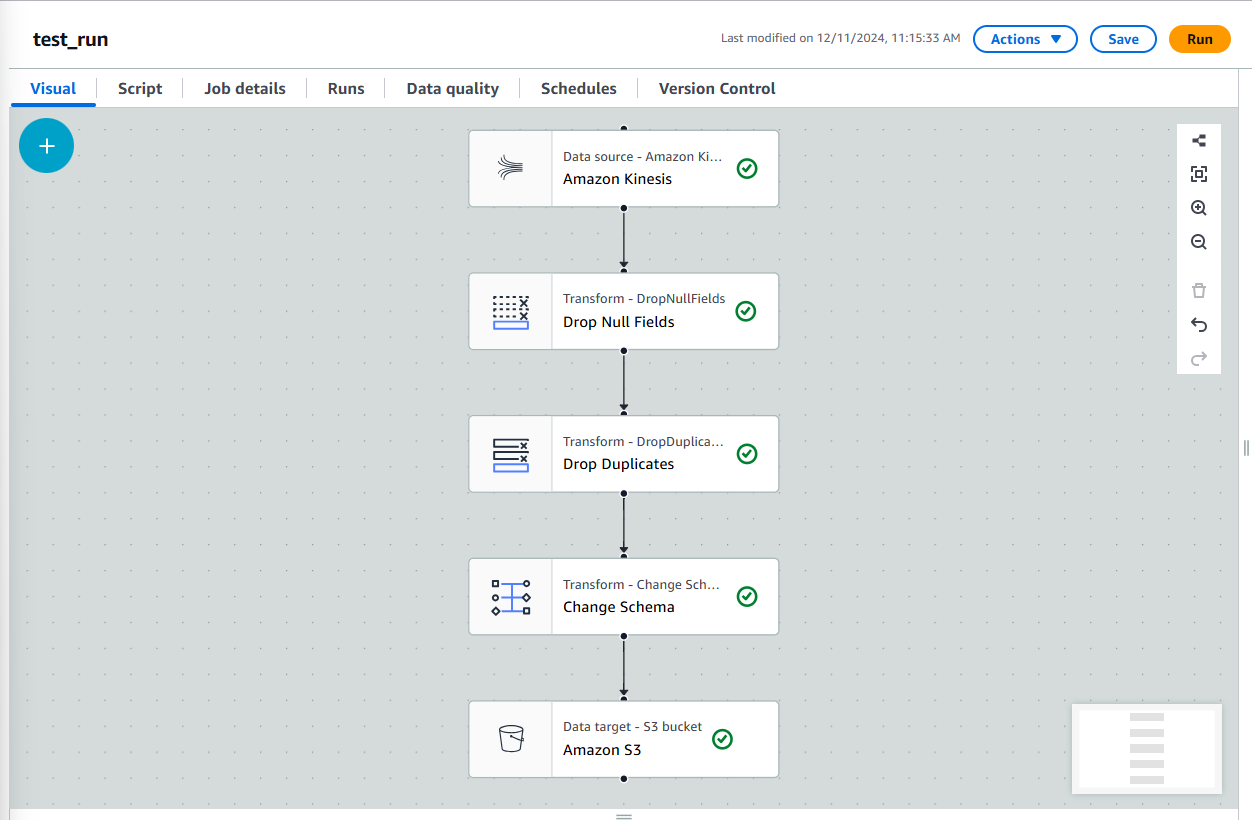
* Go to Amazon Glue and click on Databases
* Create new database by click add database
* Enter the name of database baseball-stats
* Click create database

Your Database should look as the following:



##### Use Visual ETL to transform data

* Go to AWS Glue and to Visual ETL
* Choose the source as baseball-stream Kinesis stream where we have untransformed data
  + Choose stream details
  + Starting position: Earliest
  + Window size:100
  + Choose dataformat JSON
* Choose Drop Null Fields
  + For dropnullfields, click check for empty string, “null” string, and -1 Integer
* Choose Drop Duplicates
* Choose change schema for transformation
  + keep only player, team change to date, homeruns, runbattedin, avg, and on-base percentage
  + Cast homeruns and runbattedin to integer while cast avg and on-base percentage for double
* Choose target as S3 bucket clean-baseball-statistics to store transformed data
  + Choose JSON format, uncompressed
  + Uncheck the box for data quality because it will failed the run job for not importing aws data quality library.
* Go to job detail tab
  + Make sure the IAM role is data engineer
  + Make sure type is spark streaming
  + Choose worker type G.25X which is suitable for streaming
  + Check automatically scale the number of workers
  + Input maximum number of workers you want to use. I chose 5 workers



Refs:

<https://www.youtube.com/watch?v=6ggTFOtfUxU>

<https://www.youtube.com/watch?v=ziEEdL1egPk>

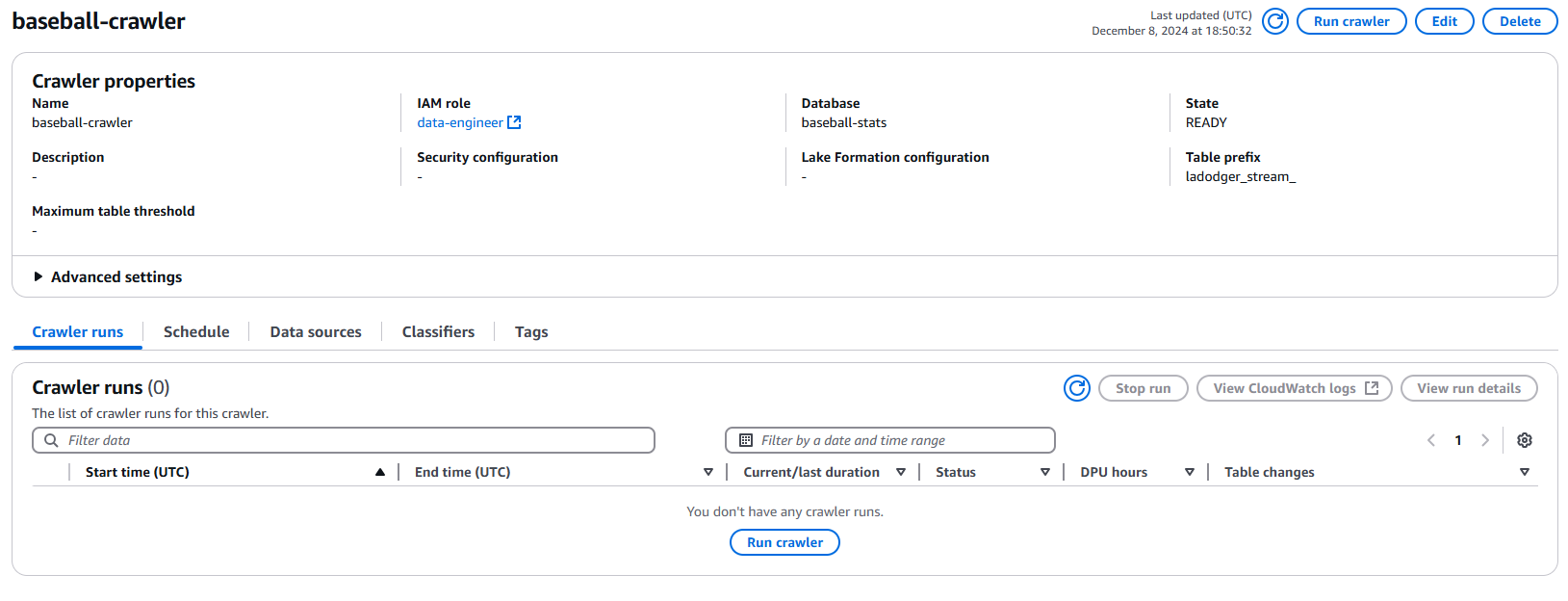
##### Check in clean-baseball-statistics S3 bucket

* You should see folder inside called ingest\_year=2024/, ingest\_month=12/, ingest\_day=8/, and hour of ingestion

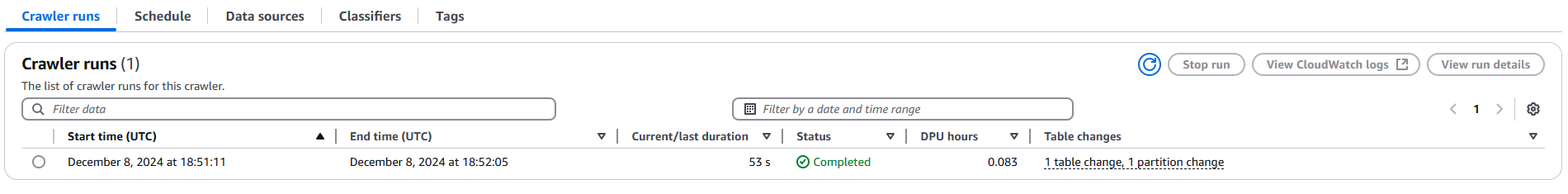
##### 

##### Use crawler to extract the schema and metadata of transformed data

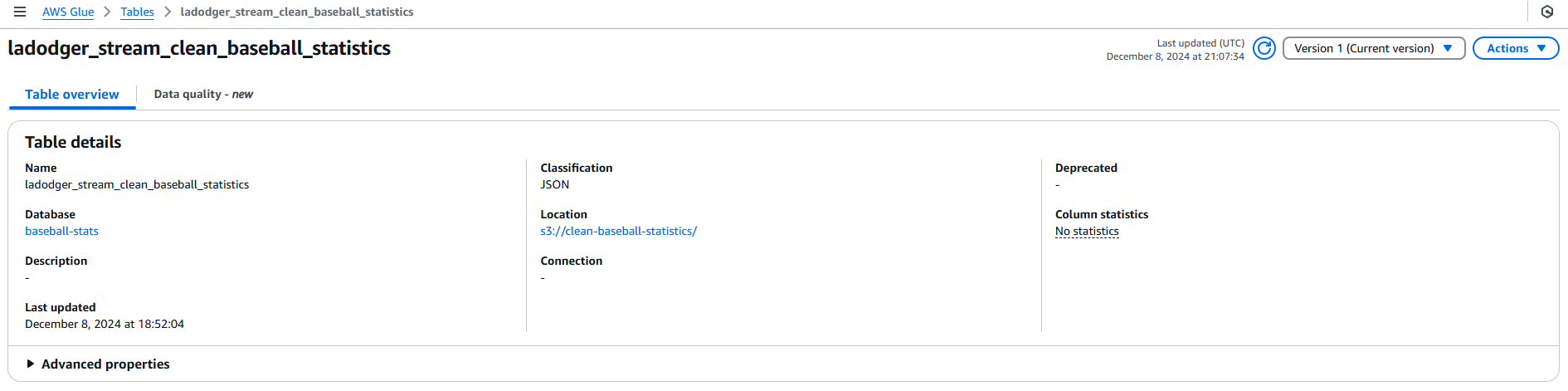
* Go to AWS Glue and to crawler
* Click create crawler
* Name the crawler baseball-crawler
* Add datasource and choose S3 bucket clean-baseball-statistics
* choose data engineer IAM role
* Choose baseball database
* Choose prefix ladodger\_stream\_
* Choose on demand for crawler schedule

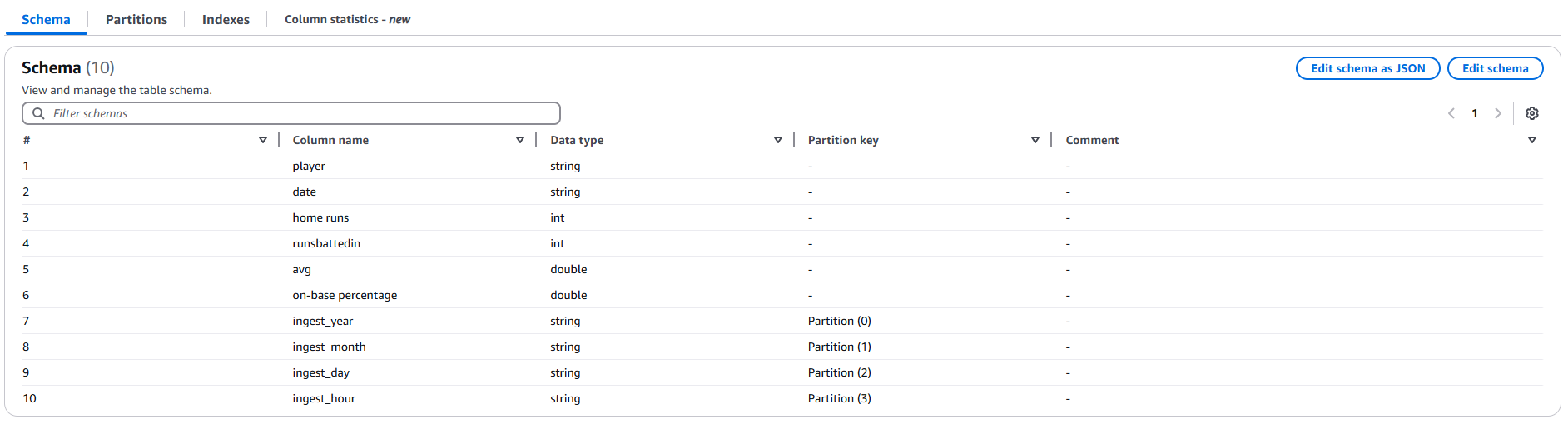


After that you should be able to see a table in the baseball-stream database once the crawler is done.



Go to database baseball-stats and to ladodger\_stream\_clean\_baseball\_statistics that we made



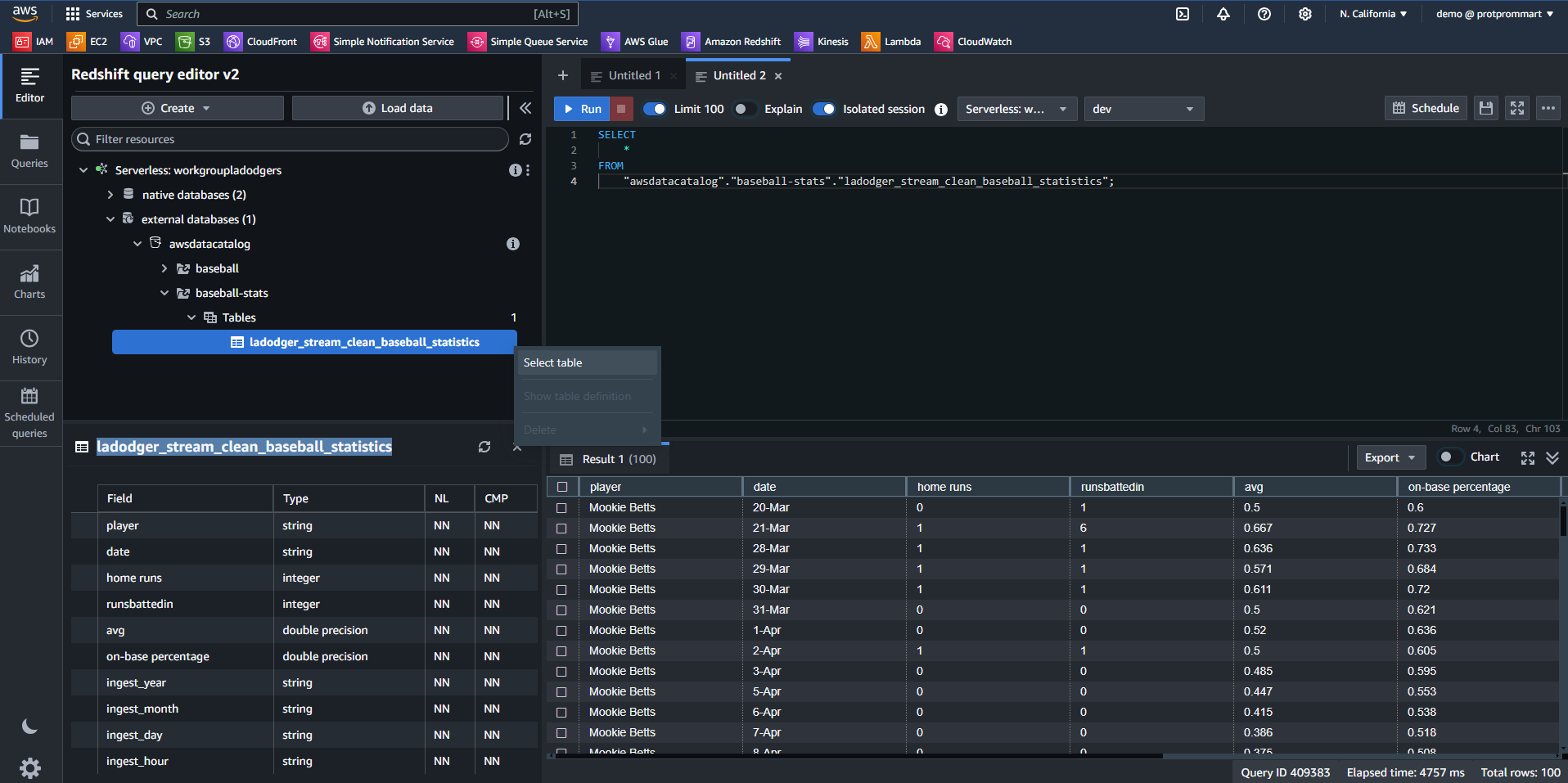


We see that ingest year, month, day, and hour column were added to save when records were brought in by Kinesis

#### Use Redshift Query Editor to visualize and query transformed data

##### Query Transformed Data

* Go to workgroup on redshift
* Click on query data
* In redshift query editor go to external databases, and into baseball-stats database, and into the ladodger\_stream\_clean\_baseball\_statistics table created by crawler
* Right click on the table, choose select table



Your table should display as this.

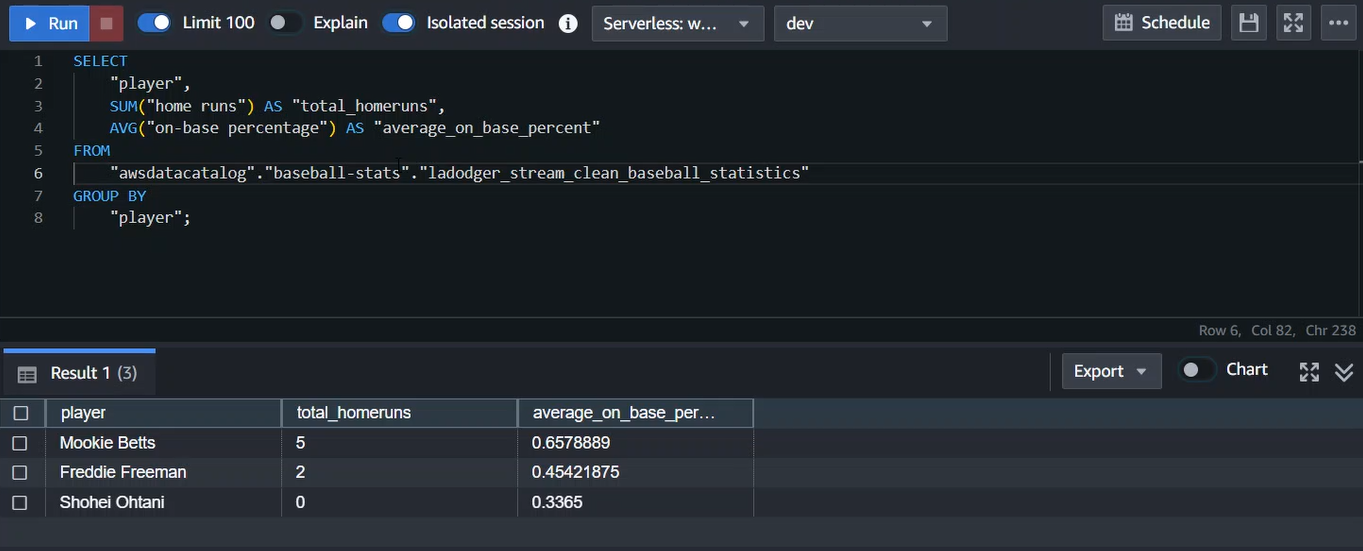
##### Visualize Data

Our goal in querying and visualizing is, let's say that you have to deliver best players statistics to the company that carries out a sports betting service, for the company to give statistics to users whether they are going to bet on among three batters. We are going to be looking especially at 2 statistics for batters position including an average of, on-base percentage, on average what is the percentage that the player hit the ball and make it safe to a base, and total home runs they made in this season so far.

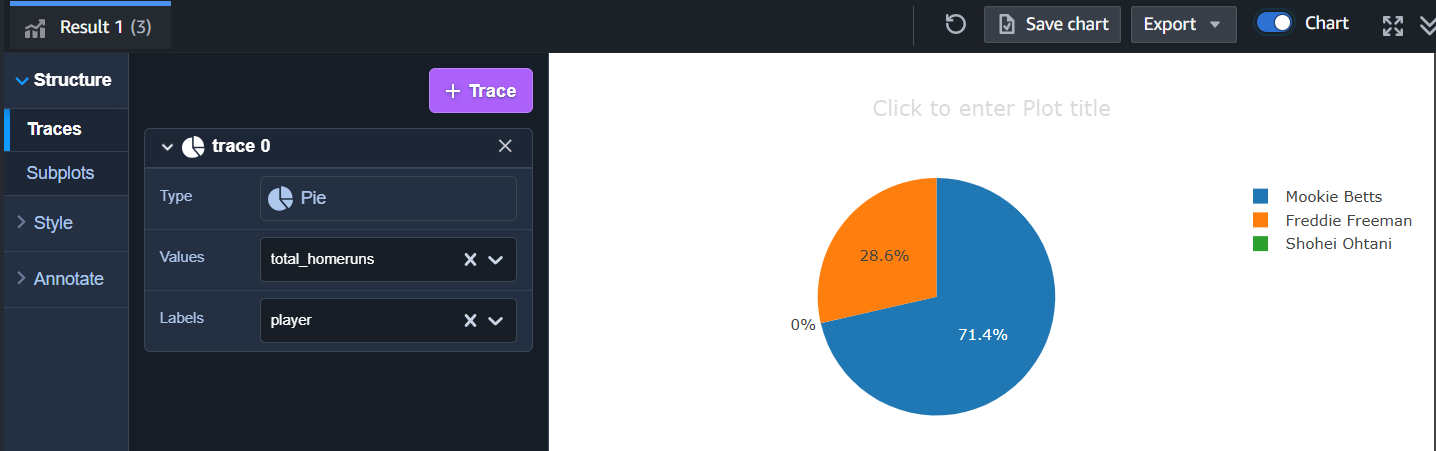
* Use the following query in redshift query editor to calculate statistics

| SELECT  "player",  SUM("home runs") AS "total\_homeruns",  AVG("on-base percentage") AS "average\_on\_base\_percent"  FROM  "awsdatacatalog"."baseball-stats"."ladodger\_stream\_clean\_baseball\_statistics"  GROUP BY  "player"; |
| --- |

* You should be able to see the following query result



* Click on chart button to view query result in chart
* Choose the following:
  + type: barchart
  + Values: total homeruns
  + Labels: player
* Choose values as average\_on\_base\_percent to view horizontal bar-chart of average on-base percentage
* You can optionally export this chart by click on export button



* Based on these statistics, a batter that you should bet on is Mookie Betts because he has the highest total home runs and average on base percentage.

