



ShieldScrape

Secure & Scalable Cloud-Based
Web Data Collection

- Collect live ESPN sports data.
- Automate cleaning, storage, querying, and visualization.
- Achieve serverless end-to-end cloud architecture.

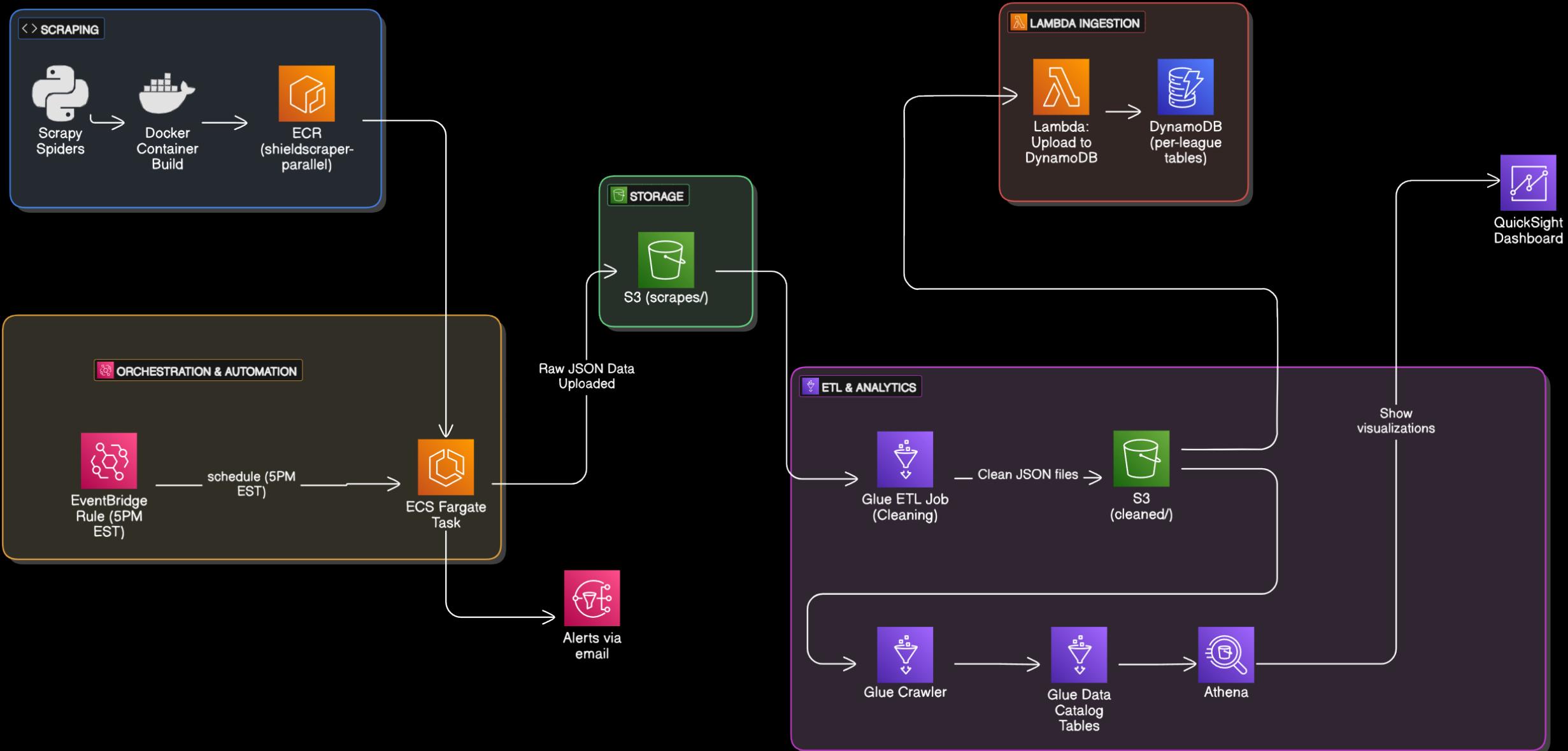


Project Goal

Team Contributions

- **Shivali Mate** - Developed the job scrapers and implemented containerization using Docker for scalable deployments.
- **Varun Sonawane** - Designed and built the ETL pipeline integrating AWS services such as Glue, S3, DynamoDB, and Athena.
- **Aryan Dhuru** - Implemented end-to-end automation, including job scheduling, orchestration with EventBridge, and scripting of cloud workflows.
- **Suhana Ambol** – Created impactful data visualizations using Amazon QuickSight and led project documentation efforts.

ShieldScraper Cloud Architecture: End-to-End Data Flow



Phase 1: Scraping and Containerization

- Developed multiple Scrapy spiders for different leagues.
- Fields: Team Name, matches played, wins, draws, losses, goals, and points.
- Created a Fargate handler to run spiders in parallel.
- Dockerized the entire scraping setup.
- Pushed Docker image to AWS ECR (**shieldscraper-parallel**) repo.

Phase 1: Scraping and Containerization

The screenshot shows a code editor interface with two files open:

- File Tree:** Shows multiple files in the directory structure: job_spider.py, job_spider_2.py, job_spider_3.py, job_spider_4.py, job_spider_5.py, and fargate_handler.py.
- JobSpider Class (job_spider.py):**

```
1 import scrapy
2
3 class JobSpider(scrapy.Spider):
4     name = "jobspider"
5     allowed_domains = []
6
7     def __init__(self, start_url=None, output_file=None, *args, **kwargs):
8         super(JobSpider, self).__init__(*args, **kwargs)
9         self.start_urls = [start_url]
10        self.output_file = output_file
11
12    def parse(self, response):
13
14        title = response.css(".Table__Title::text").get()
15        if not title:
16            title = "unknown_league"
17            formatted_title = title.replace(" ", "_")
18
19        self.formatted_title = formatted_title
20
21        team_names = response.css('span.hide-mobile a::text').getall()
22        stats = response.css('tr.Table__TR.Table__TR--sm.Table__even td.Table__Ti
23
24        for i in range(0, len(stats), 8):
25            team_stats = stats[i:i+8]
26            team_data = {
27                'Team': team_names[i // 8],
28                'Games Played': team_stats[0],
29                'Wins': team_stats[1],
30                'Draws': team_stats[2],
31                'Losses': team_stats[3],
32                'Goals For': team_stats[4],
33                'Goals Against': team_stats[5],
34                'Goal Difference': team_stats[6],
35                'Points': team_stats[7],
36                'Title': formatted_title
37            }
38            yield team_data
39
```
- fargate_handler.py:**

```
3 import boto3
4 from scrapy.crawler import CrawlerRunner
5 from twisted.internet import reactor, defer
6
7 from job_spider import JobSpider
8 from job_spider_2 import JobSpider2
9 from job_spider_3 import JobSpider3
10 from job_spider_4 import JobSpider4
11 from job_spider_5 import JobSpider5
12
13 s3 = boto3.client('s3')
14 RAW_BUCKET = os.environ['RAW_BUCKET']
15
16 output_files = [
17     "scraped_soccer_stats_league1.json",
18     "scraped_soccer_stats_league2.json",
19     "scraped_soccer_stats_league3.json",
20     "scraped_soccer_stats_league4.json",
21     "scraped_soccer_stats_league5.json",
22 ]
23
24 start_urls = [
25     "https://www.espn.com/soccer/standings/_/league/eng.1",
26     "https://www.espn.com/soccer/standings/_/league/fra.1",
27     "https://www.espn.com/soccer/standings/_/league/esp.1",
28     "https://www.espn.com/soccer/standings/_/league/ita.1",
29     "https://www.espn.com/soccer/standings/_/league/ger.1",
30 ]
31
32 def upload_to_s3(local_file, s3_key):
33     with open(local_file, "rb") as f:
34         s3.put_object(
35             Bucket=RAW_BUCKET,
36             Key=f"scrapes/{s3_key}",
37             Body=f.read(),
38             ContentType='application/json'
39         )
40         print(f"Uploaded {s3_key} to S3!")
41
```

The code editor includes a sidebar with various icons for file operations, a status bar at the bottom showing file details like 'shimate (1 day ago)', and a system tray at the very bottom.

Phase 1: Scraping and Containerization

The screenshot shows the AWS ECR (Amazon Elastic Container Registry) console. The left sidebar is titled "Amazon Elastic Container Registry" and includes sections for "Private registry" (Repositories, Summary, **Images**, Permissions, Lifecycle Policy, Repository tags, Features & Settings), "Public registry" (Repositories, Settings), and links to "ECR public gallery", "Amazon ECS", "Amazon EKS", "Getting started", and "Documentation". The main content area shows a table titled "Images (36)" with a search bar at the top. A blue banner at the top of the table area states: "Image scan overview, status, and full vulnerabilities has moved to the Image detail page. To access, click an image tag." The table columns are: Image tag, Artifact type, Pushed at, Size (MB), Image URI, Digest, and Last recorded pull time. The data in the table is as follows:

Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest	Last recorded pull time
latest	Image Index	April 28, 2025, 20:37:52 (UTC-04)	119.36	<input type="button" value="Copy URI"/>	sha256:12f0abe0301f458090fa7185ccba2d...	April 28, 2025, 20:42:47 (UTC-04)
-	Image	April 28, 2025, 20:37:52 (UTC-04)	0.00	<input type="button" value="Copy URI"/>	sha256:413d32d3e7bcfaa41978785611f57e...	-
-	Image	April 28, 2025, 20:37:52 (UTC-04)	119.36	<input type="button" value="Copy URI"/>	sha256:83fa11c336bdb71a467357028bb5d...	April 28, 2025, 20:37:52 (UTC-04)
-	Image Index	April 28, 2025, 20:31:47 (UTC-04)	119.36	<input type="button" value="Copy URI"/>	sha256:bbae548834fabaa3414f5923ae5531f...	April 28, 2025, 20:34:43 (UTC-04)
-	Image	April 28, 2025, 20:31:47 (UTC-04)	0.00	<input type="button" value="Copy URI"/>	sha256:3ff52bd71c9e321747921e6fca54ea...	-
-	Image	April 28, 2025, 20:31:47 (UTC-04)	119.36	<input type="button" value="Copy URI"/>	sha256:84360b75c3205c47dad936784744...	April 28, 2025, 20:31:47 (UTC-04)
-	Image Index	April 28, 2025, 18:26:07 (UTC-04)	119.33	<input type="button" value="Copy URI"/>	sha256:1e30d1e41b551edc2eaebbe28a6e6...	April 28, 2025, 18:26:52 (UTC-04)
-	Image	April 28, 2025, 18:26:06 (UTC-04)	119.33	<input type="button" value="Copy URI"/>	sha256:8c86c2bc9aea23a7682f606b0876eb...	April 28, 2025, 18:26:06 (UTC-04)
-	Image	April 28, 2025, 18:26:06 (UTC-04)	0.00	<input type="button" value="Copy URI"/>	sha256:166c5ddd2c678ed1b3eb6de9adc90...	-
-	Image Index	April 28, 2025, 18:06:23 (UTC-04)	119.33	<input type="button" value="Copy URI"/>	sha256:a28588ec18b01357dee62d57b1b57...	April 28, 2025, 18:07:29 (UTC-04)
-	Image	April 28, 2025, 18:06:23 (UTC-04)	0.00	<input type="button" value="Copy URI"/>	sha256:f9ef7a7d2e0793e7a4051f642cc7c25...	April 28, 2025, 18:06:23 (UTC-04)
-	Image	April 28, 2025, 18:06:23 (UTC-04)	119.33	<input type="button" value="Copy URI"/>	sha256:3d7182e1aae6e2eaec103dc002322...	April 28, 2025, 18:07:30 (UTC-04)
-	Image Index	April 28, 2025, 17:50:14 (UTC-04)	119.33	<input type="button" value="Copy URI"/>	sha256:177774e77e9cf0a3a740946cf2708...	April 28, 2025, 17:50:58 (UTC-04)

Phase 2: Scheduled Trigger using EventBridge

- Created AWS EventBridge rule with daily trigger at 05:00 pm EST.
- Event triggers ECS task in shieldscraper-cluster1.
- Task definition: **shieldscraper-parallel-task**

Phase 2: Scheduled Trigger using EventBridge

The screenshot shows the AWS EventBridge Rules page. On the left, there's a navigation sidebar with sections like Amazon EventBridge, Developer resources, Buses, Pipes, Scheduler, Integration, and Schema registry. The main area is titled "Rules" and contains a "Select event bus" section where "default" is chosen. Below this is a table titled "Rules (3)" showing three entries:

Name	Status	Type	ARN	Description
ecs-task-failure-alert	Enabled	Standard	arn:aws:events:us-east-1:85172530551:rule/ecs-task-failure-alert	Alert when ECS Task fails
RunLambdaDaily530PM	Enabled	Scheduled Standard	arn:aws:events:us-east-1:85172530551:rule/RunLambdaDaily530PM	Trigger Lambda function daily at 5:30 PM
shieldscraper-ecs-daily-trigger	Enabled	Scheduled Standard	arn:aws:events:us-east-1:85172530551:rule/shieldscraper-ecs-daily-trigger	Run ECS scraping task daily

Phase 3: Data Collection (Scraping)

- Fargate ECS Task launches Scrapy spiders.
- Fresh soccer standings are scraped.
- Output stored to Amazon S3 Bucket.

Phase 3: Data Collection (Scraping)

The screenshot shows the AWS Elastic Container Service (ECS) interface. The top navigation bar includes the AWS logo, a search bar, and a 'Clusters' section. The main content area is titled 'shieldscraper-cluster1' and displays the 'Cluster overview' and 'Tasks' sections.

Cluster overview:

- ARN:** arn:aws:ecs:us-east-1:851725305519:cluster/shieldscraper-cluster1
- Status:** Active
- CloudWatch monitoring:** Default
- Registered container instances:** -

Tasks:

Status	Pending	Running
1	-	-

Tasks (1):

Task	Last status	Desired state	Task definition	Health status	Created at	Started by	Started at	Group
c4345a2e1b0b4aa1a8d8881522973af2	⚠️ Stopped	Stopped	shieldscraper-para...	Unknown	1 minute ago	-	1 minute ago	family:shieldscrap...

Phase 4: ETL with AWS Glue

- AWS Glue job (shieldscraper-cleaning-job) starts.
- Reads raw files from S3.
- Cleans and formats the data.
- Stores the clean JSON into: s3://iu-sheild-scraper-bucket/cleaned/

Phase 4: ETL with AWS Glue

The screenshot shows the AWS Glue console interface. On the left, there is a navigation sidebar with the following sections:

- AWS Glue**
 - Getting started
 - ETL jobs**
 - Visual ETL
 - Notebooks
 - Job run monitoring
 - Data Catalog tables
 - Data connections
 - Workflows (orchestration)
 - Zero-ETL integrations [New](#)- Data Catalog**
 - Databases
 - Tables
 - Stream schema registries
 - Schemas
 - Connections
 - Crawlers
 - Classifiers
 - Catalog settings
- Data Integration and ETL**
 - Zero-ETL integrations [New](#)
 - ETL jobs**
 - Visual ETL
 - Notebooks
 - Job run monitoring
 - Interactive Sessions
 - Data classification tools
 - Sensitive data detection
 - Record Matching
 - Triggers
 - Workflows (orchestration)
 - Blueprints
 - Security configurations

The main content area displays the configuration for the job **shieldscraper-cleaning-job**. The top navigation bar includes a search bar, a [Alt+S] keyboard shortcut, and various status indicators. The job name is **shieldscraper-cleaning-job**, last modified on **4/28/2025, 11:38:58 PM**. The tabs at the top are **Script**, **Job details**, **Runs**, **Data quality**, **Schedules** (which is selected), and **Version Control**. The **Schedules** section shows a single entry:

Description	Schedule	Cron Expression	Status
DailyShieldScraperCleaner Runs daily at 5:15 PM EDT to clean soccer stats.	At 09:15 PM	15 21 * * ? *	Activated

At the bottom of the page, there are links for [CloudShell](#) and [Feedback](#), along with copyright information from 2025 and links for [Privacy](#), [Terms](#), and [Cookie preferences](#).

Phase 4: ETL with AWS Glue

The screenshot shows the AWS S3 console interface. The left sidebar contains navigation links for Amazon S3, General purpose buckets, Storage Lens, Feature spotlight, and AWS Marketplace for S3. The main content area displays the 'iu-shield-scraper-bucket' page, which includes tabs for Objects, Metadata, Properties, Permissions, Metrics, Management, and Access Points. The 'Objects' tab is selected, showing a list of four objects: 'athena-results/' (Folder), 'cleaned/' (Folder), 'scrapes/' (Folder), and 'scripts/' (Folder). Each object entry includes a checkbox, the object name, type, last modified date, size, and storage class. Above the object list, there are buttons for Copy S3 URI, Copy URL, Download, Open, Delete, Actions, Create folder, and Upload. A search bar at the top allows users to find objects by prefix.

Name	Type	Last modified	Size	Storage class
athena-results/	Folder	-	-	-
cleaned/	Folder	-	-	-
scrapes/	Folder	-	-	-
scripts/	Folder	-	-	-

Raw Data VS Cleaned Data

Raw data

```
{"Team": "Real Madrid", "Games Played": "33", "Wins": "22", "Draws": "6", "Losses": "5", "Goals For": "66", "Goals Against": "31", "Goal Difference": "+35", "Points": "72", "Title": "Spanish_LALIGA"},
```

Raw Data VS Cleaned Data

Cleaned data

```
{"Team": "Real Madrid", "Games Played": 33, "Wins": 22,  
"Draws": 6, "Losses": 5, "Goals For": 66, "Goals Against":  
31, "Goal Difference": 35, "Points": 72}
```

Phase 5: DynamoDB update via Lambda

- AWS Lambda function triggers after ETL completes.
- Pulls cleaned data from /cleaned/.
- Uploads records into separate DynamoDB tables (per league).

Phase 5: DynamoDB update via Lambda

The screenshot shows the AWS Lambda console interface for a function named 'ShieldScraperUploader'. The top navigation bar includes the AWS logo, a search bar, and account information for 'United States (N. Virginia)' and 'varun_sonawane'. The main content area displays the 'Function overview' tab for 'ShieldScraperUploader'. The function has no layers and no triggers. It has one destination, which is an S3 bucket. There are buttons for 'Throttle', 'Copy ARN', 'Actions', 'Export to Infrastructure Composer', and 'Download'. The 'Description' field is empty. The 'Last modified' field shows '2 days ago'. The 'Function ARN' is listed as 'arn:aws:lambda:us-east-1:851725305519:function:ShieldScraperUploader'. The 'Function URL' field is empty. Below the overview, there are tabs for 'Code', 'Test', 'Monitor', 'Configuration', 'Aliases', and 'Versions'. The 'Code' tab is selected, showing the code source editor for 'lambda_function.py'. The code is as follows:

```
lambda_function.py
1 import json
2 import boto3
3
4 def lambda_handler(event, context):
5     s3 = boto3.client('s3')
6     dynamodb = boto3.resource('dynamodb', region_name='us-east-1')
7
8     BUCKET_NAME = 'iu-shield-scraper-bucket'
9     PREFIX = 'cleaned/'
```

Phase 5: DynamoDB update via Lambda

The screenshot shows the AWS DynamoDB console interface. The left sidebar has a navigation menu with 'DynamoDB' selected, followed by 'Dashboard', 'Tables', 'Explore items', 'PartiQL editor', 'Backups', 'Exports to S3', 'Imports from S3', 'Integrations', 'Reserved capacity', and 'Settings'. Below this is a 'DAX' section with 'Clusters', 'Subnet groups', 'Parameter groups', and 'Events'. The main area is titled 'Soccer_Stats_English_Premier_League'. It features a 'Scan or query items' section with 'Scan' (selected) and 'Query' buttons, and dropdowns for 'Select a table or index' (set to 'Table - Soccer_Stats_English_Premier_League') and 'Select attribute projection' (set to 'All attributes'). A green status bar at the bottom indicates 'Completed · Items returned: 20 · Items scanned: 20 · Efficiency: 100% · RCU's consumed: 2'. Below this is a table titled 'Table: Soccer_Stats_English_Premier_League - Items returned (20)' with a timestamp 'Scan started on April 30, 2025, 19:12:23'. The table has columns: Team (String), Draws, Games Played, Goal Difference, Goals Against, Goals For, Losses, and Points. The data rows are:

Team (String)	Draws	Games Played	Goal Difference	Goals Against	Goals For	Losses	Points
Brentford	7	33	6	50	56	13	46
Ipswich Town	9	34	-41	74	33	21	21
Arsenal	13	34	34	29	63	3	67
Southampton	5	34	-55	80	25	27	11

Phase 6: Glue Crawler Refresh

- Lambda function triggers AWS Glue Crawler.
- Updates Glue Catalog with latest S3 data structure.

Phase 6: Glue Crawler Refresh

The screenshot shows the AWS Glue Crawler properties page for a crawler named "soccer_leagues_crawler".

Crawler properties:

- Name: soccer_leagues_crawler
- IAM role: GlueS3ShieldScraperRole
- Description: -
- Maximum table threshold: -
- Database: soccer_leagues
- Security configuration: -
- Lake Formation configuration: -
- State: READY
- Table prefix: -

Advanced settings: (button)

Crawler runs (11): The list of crawler runs for this crawler.

Start time (UTC)	End time (UTC)	Current/last duration	Status	DPU hours	Table changes
April 29, 2025 at 17:46:19	April 29, 2025 at 17:47:40	01 min 20 s	Completed	0.038	-
April 29, 2025 at 00:50:07	April 29, 2025 at 00:51:33	01 min 26 s	Completed	0.037	3 table changes, 0 partition changes
April 29, 2025 at 00:43:56	April 29, 2025 at 00:45:13	01 min 16 s	Completed	0.043	-
April 29, 2025 at 00:27:54	April 29, 2025 at 00:29:09	01 min 15 s	Completed	0.037	2 table changes, 0 partition changes
April 29, 2025 at 00:23:14	April 29, 2025 at 00:24:28	01 min 13 s	Completed	0.037	-
April 29, 2025 at 00:15:32	April 29, 2025 at 00:16:57	01 min 24 s	Completed	0.037	1 table change, 5 partition changes
April 28, 2025 at 23:05:56	April 28, 2025 at 23:07:37	01 min 41 s	Completed	0.053	11 table changes, 5 partition changes
April 28, 2025 at 20:05:24	April 28, 2025 at 20:07:18	01 min 53 s	Completed	0.043	11 table changes, 0 partition changes
April 28, 2025 at 19:51:12	April 28, 2025 at 19:52:28	01 min 15 s	Completed	0.040	1 table change, 0 partition changes
April 28, 2025 at 19:43:02	April 28, 2025 at 19:44:17	01 min 14 s	Completed	0.037	7 table changes, 0 partition changes

Phase 7: Query via Amazon Athena and Visualization with QuickSight

- Glue Tables auto-created from Crawler metadata.
- Data instantly queryable using Amazon Athena SQL.
- Build QuickSight dashboards linked to Athena tables.
- Dashboard automatically refreshes with latest soccer data.

Phase 7: Query via Amazon Athena and Visualization with QuickSight

The screenshot shows the Amazon Athena Query editor interface. On the left, the 'Data' sidebar is open, displaying the 'Data source' (AwsDataCatalog), 'Catalog' (None), and 'Database' (soccer_leagues). Below these are sections for 'Tables and views' (with a 'Create' button) and a list of tables: english_premier_league, french_ligue_1, german_bundesliga, italian_serie_a, and spanish_laliga. The main workspace shows a query titled 'Query 11' with the SQL command:

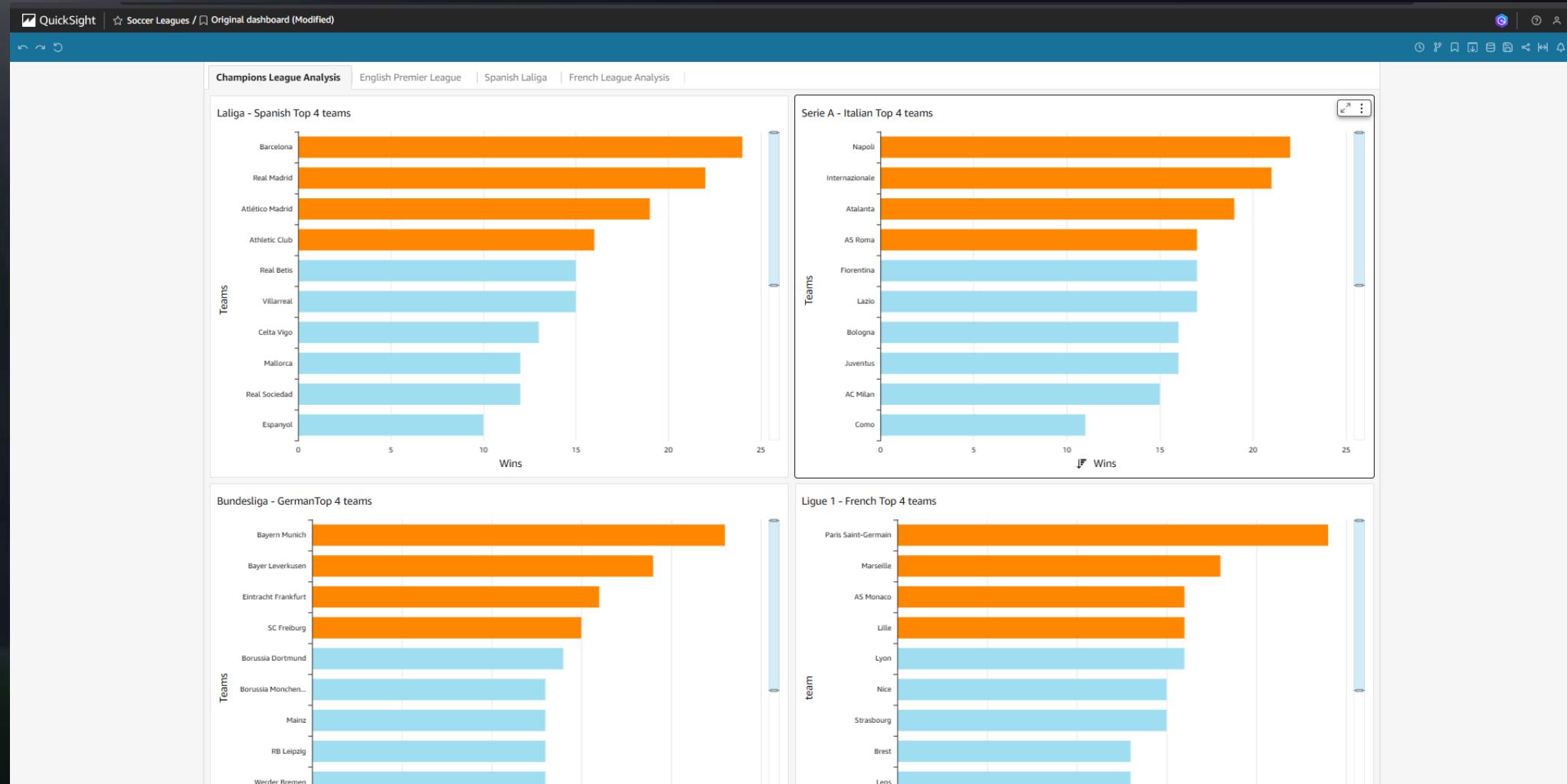
```
1 SELECT * FROM "AwsDataCatalog"."soccer_leagues"."english_premier_league" limit 10;
```

Below the query, the status bar indicates 'SQL Ln 1, Col 1'. There are buttons for 'Run again' (orange), 'Explain', 'Cancel', 'Clear', and 'Create'. A note says 'Reuse query results up to 60 minutes ago'. The 'Query results' tab is selected, showing a green bar with 'Completed' status, 'Time in queue: 86 ms', 'Run time: 564 ms', and 'Data scanned: 3.10 KB'. The 'Results (10)' table displays the top two rows of data:

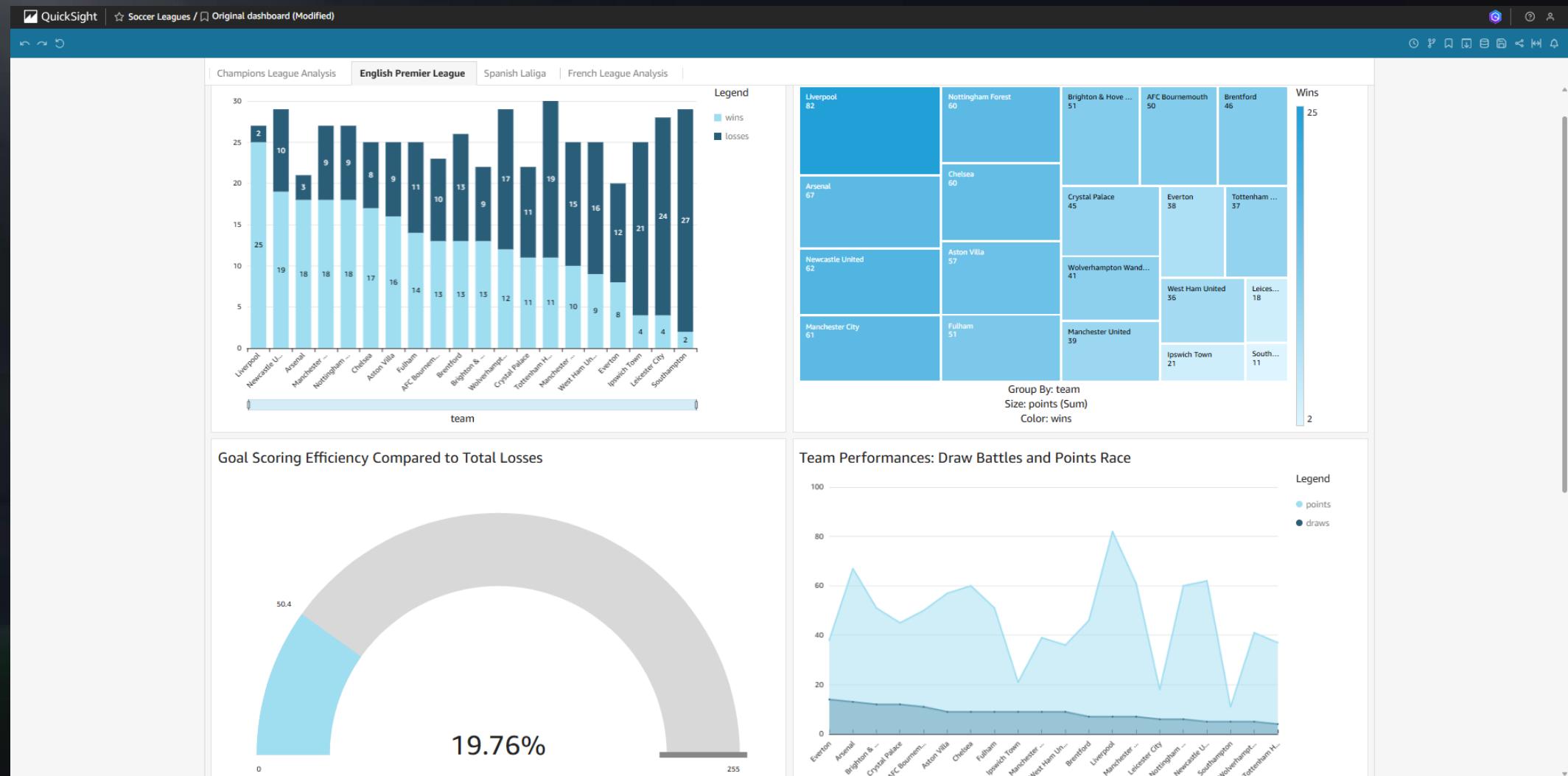
#	team	games played	wins	draws	losses	goals for	goals against	goal difference	points
1	Liverpool	34	25	7	2	80	32	48	82
2	Arsenal	34	18	13	3	63	29	34	67

Buttons for 'Copy' and 'Download results CSV' are available at the bottom right of the results table.

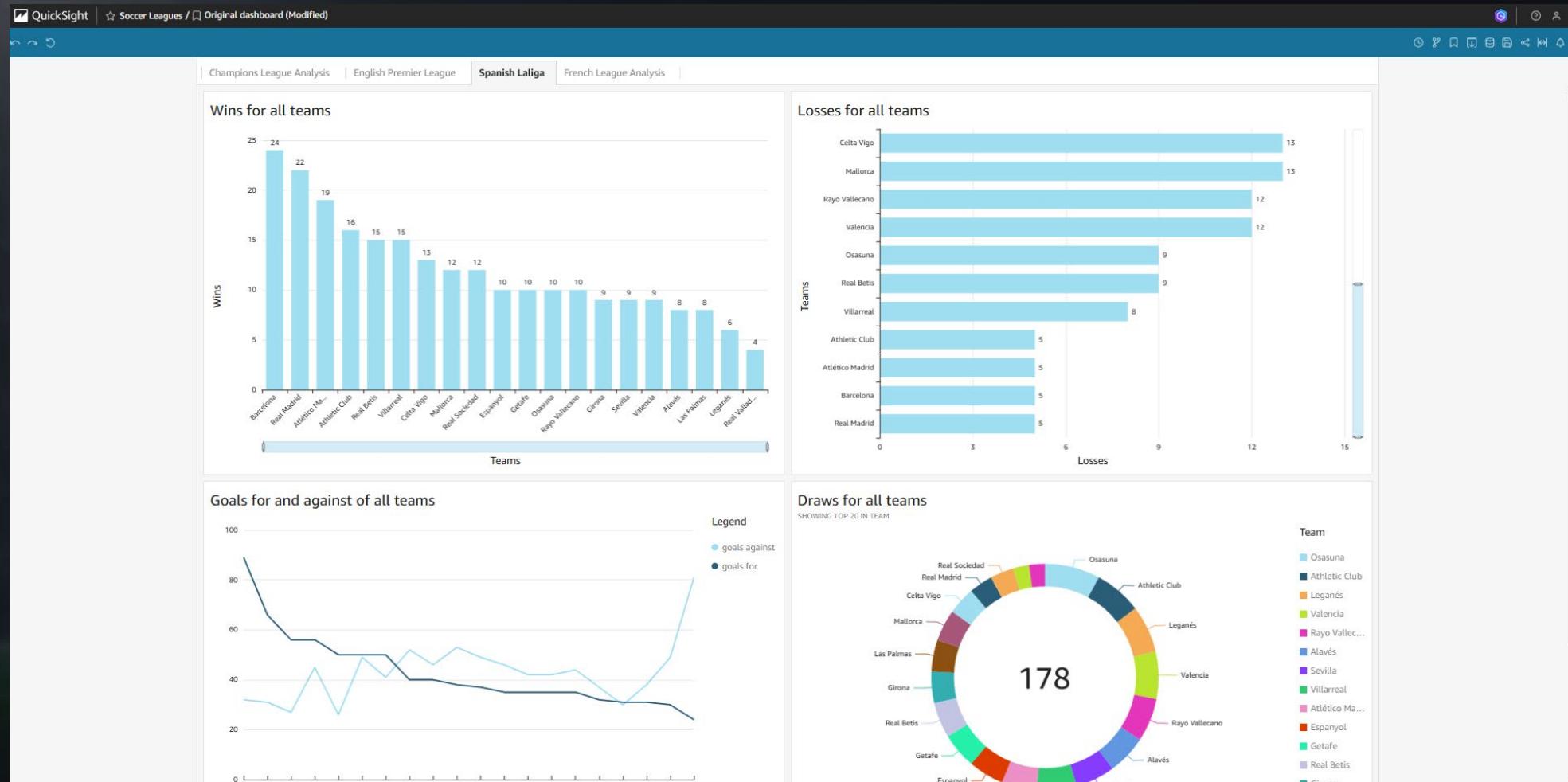
Phase 7: Query via Amazon Athena and Visualization with QuickSight



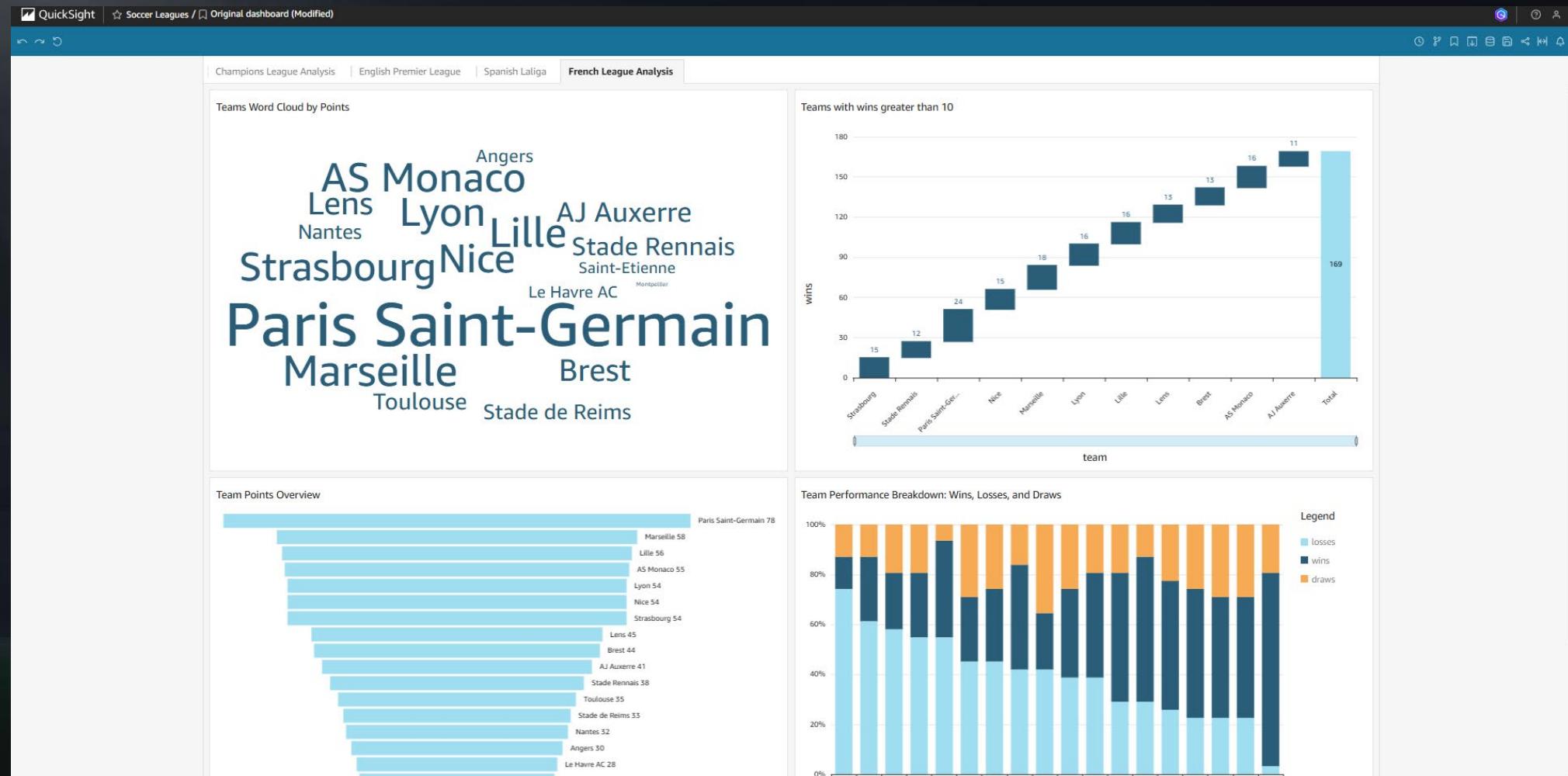
Phase 7: Query via Amazon Athena and Visualization with QuickSight



Phase 7: Query via Amazon Athena and Visualization with QuickSight



Phase 7: Query via Amazon Athena and Visualization with QuickSight



Execution Timeline

Timeline of a ShieldScraper Run

Time (EST)	Actions
05:00 PM	EventBridge triggers ECS Fargate
05:10 PM	Scraping completes, saved to S3
05:15 PM	Glue ETL cleans and stores to /cleaned/
05:25 PM	Lambda uploads to DynamoDB
05:30 PM	Glue Crawler refreshes catalog
06:00 PM	Dashboards show updated data

Conclusion

- ShieldScraper achieves fully automated, scalable sports data collection and visualization using serverless cloud technologies, delivering fresh insights daily with minimal operational effort.
- The project demonstrates strong system design, cloud automation, and scalable data engineering practices.



THANK YOU