

Precision in chaos

Leveraging GenAI Multi-Agent Systems for Trade Analytics

STAC Tech Briefing | December 3, 2024

Jean-Philippe Rezler | Global Head of AI & Analytics

The Promise vs. Reality of GenAI in Trading

- GenAI and LLMs have revolutionized the way to **interact with data** and how to **get insights across sectors**.
- In trading analytics, however, the promise often falls short.

CHALLENGE:

Translate complex data into actionable insights using GenAI

CHALLENGE #1

Generate the right SQL queries on large datasets

Text to SQL: From basic tests to real-world enterprise use cases

Successive benchmarks assess SQL generation from LLMs.

How OpenAI o1 performs with best agentic framework

Increase in complexity to reach real-world use case

Benchmark	Success Rate
Spider 1.0 (2017)	92%
BIRD (2023)	74%
Spider 2.0 (current)	15%

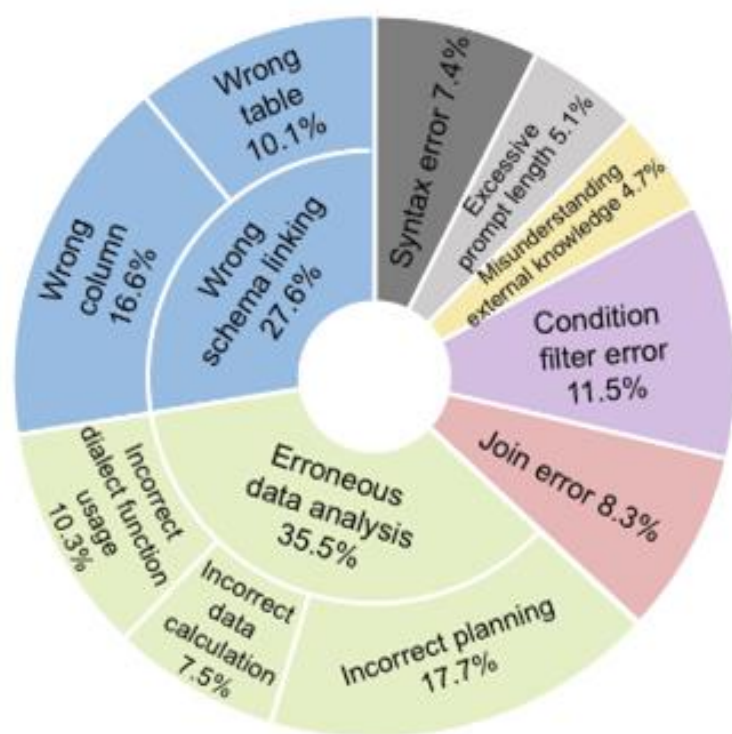
Spider 2.0 is trying to reflect real-world enterprise so is much more complex

To get to the best results, state-of-the-art LLMs, like OpenAI o1 or Claude Sonnet 3.5, need to be **used and augmented with agentic, RAG framework.**

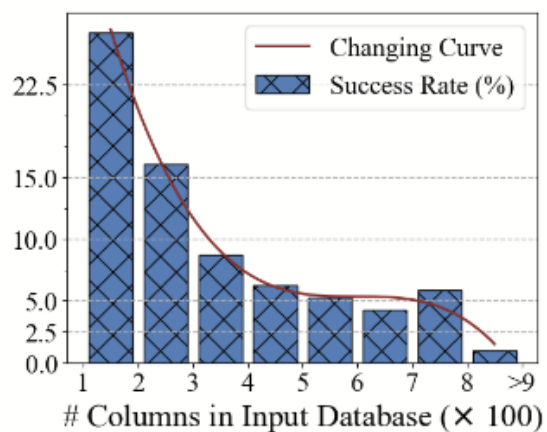
NOT STAC BENCHMARKS

CHALLENGE #1 Generate the right SQL queries on large datasets

Text to SQL: Why it makes this so difficult



Statistics of errors



Why poor results?

- Project level with multiple queries
- Nested schemas
- Schema size

CHALLENGE #1 Generate the right SQL queries on large datasets

- GenAI struggles with **complex, multi-step SQL queries** needed in trading.
- LLMs often generate **inaccurate or overly simplistic queries**.

➔ Time wasted **troubleshooting** or writing **queries manually**.

BIRD (**BI**g Bench for La**R**ge-scale **D**atabase Grounded Text-to-SQL Evaluation) benchmark:

	Model	Code	Size	Oracle Knowledge	Dev (%)	Test (%)
	Human Performance <i>Data Engineers + DB Students</i>			✓		92.96
1 Nov 11, 2024	DSAIR + GPT-4o <i>AT&T - CDO</i>		UNK	✓	74.32	74.12
2 Nov 3, 2024	CHASE-SQL + Gemini <i>Google Cloud</i> [Pourreza et al. '24]		UNK	✓	73.14	74.06
3 Oct 27, 2024	ExSL + granite-34b-code <i>IBM Research AI</i>		34B	✓	72.43	73.17
39 Jul 01, 2023	GPT-4 <i>Baseline</i>	[link]	UNK	✓	46.35	54.89

NOT STAC BENCHMARKS

CHALLENGE #2

Get reliable insights while LLMs struggle with logic and math & hallucinate

Key reasons:

- 1 Trained to always give an answer** It will never say it doesn't know, leading to hallucinations.
- 2 Simulating human text** Logic is not always the only thing that matters in text.
- 3 Tokenisation** Can lead to errors in digits

Example of tokenisation

10012394  100 | 123 | 9 | 4  

CHALLENGE #2

Get reliable insights while LLMs struggle with logic and math & hallucinate

ChatGPT without agent is (very) bad at math

Euclidean division of 1765 by 123



The Euclidean division of 1765 by 123 gives:

- Quotient: 14
- Remainder: 53

So, 1765 divided by 123 equals 14 with a remainder of 53.

Wrong!

$$14 \times 123 + 53 = 1775$$

- LLMs are **not built for advanced mathematical reasoning**.
- They “hallucinate” – generating **plausible-sounding but incorrect** information.

➤ High risk for **error-prone, unreliable** insights.

NEW APPROACH

Multi-Agent Structure and Semantic Layer

- Make **complex data model simply accessible** through a semantic layer
- **Simplify LLMs** and leverage a **multi-agent structure with chain of thoughts** to get relevant insight on your data

➤ **Don't rely heavily on LLM for advanced logic and accurate answers**

The Semantic Layer

How it works



What end user would like to see: **simple pivot tables**

Dimensions / decoration	Dimensions	Dimensions
<ul style="list-style-type: none">RiskFactorId (String)Regulation (String)RiskClass (String) declared as EnumFRTBucket (String)CreditQuality (String)CurrentName (String)RiskType (String)RiskClassMkt (String)LiquidityHorizon (Float64)Reflexivity (String)Desk (String)Trade (String)TenorAndMaturity (Array[String])RiskValue (Array[Float])Scenario (String)Currency (String)Maturity (String)UnderlyingName (String)Book (String)ProductID (String)AssetClass (String)ProductCode (String)Desk (String)BusinessLine (String)Country (String)Region (String)	<ul style="list-style-type: none">Date (Date)Period (String) declared as EnumRiskFactorId (String)Trade (String)Scenario (String)Currency (String)TenorAndMaturity (String)ProductCode (String)Book (String)Desk (String)ProductID (String)AssetClass (String)ProductCode (String)ProductClass (String)ProductCode (String)Desk (String)BusinessLine (String)Country (String)Region (String)POVector (String) declared as EnumEquity (String)LiquidityHorizon (Float64)RiskClassMkt (String) declared as EnumScenarioPeriod (String) declared as Enum	<ul style="list-style-type: none">Date (Date)Period (String) declared as EnumRiskFactorId (String)Trade (String)Scenario (String)Currency (String)TenorAndMaturity (String)ProductCode (String)Book (String)Desk (String)ProductID (String)AssetClass (String)ProductCode (String)ProductClass (String)ProductCode (String)Desk (String)BusinessLine (String)Country (String)Region (String)POVector (String) declared as EnumEquity (String)LiquidityHorizon (Float64)RiskClassMkt (String) declared as EnumScenarioPeriod (String) declared as Enum
Metrics	Metrics	Metrics
<ul style="list-style-type: none">Sum Risk ValuesCount Risk FactorsAny bespoke	<ul style="list-style-type: none">CountSumAny bespoke	<ul style="list-style-type: none">CountSumAny bespoke

Pivot representation of data including all facts, dimensions, decoration, versioning etc.

Semantic layer

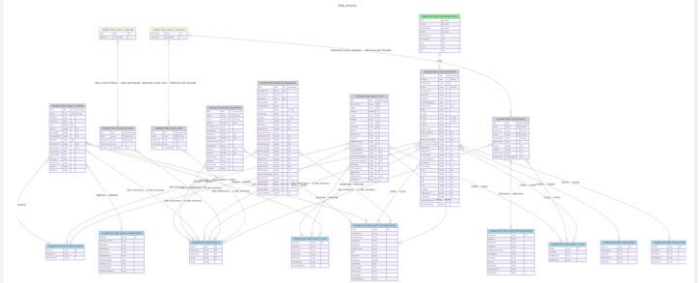
Use of relational algebra

Build of Abstract Syntax Tree (AST) on pivots

Transform AST to physical model

Optimise AST

The real world



Physical reality with sharding, multiple tables and zip to join, materialized views etc.

Request based on **pivots representation**

LLM like any user

Result of the **SQL query**

Seamless handling of...

- Fact fields vs Decoration fields
- User-defined fields
- Vector-based dimensions and metrics
- Result conversion (FX)
- Materialised Views
- Sharding
- Data versions...etc.

Semantic layer transforms
to optimised SQL query

```
POST /module/mtm/query {  
  "metrics": [ "MtmAggreg" ],  
  "pivot": "MTM",  
  "by": [ "country", "dates", "scenario" ],  
  "where": { "csa": { "values": [ "BANK01" ] },  
             "country": { "values": [ "CANADA" ] }  
},  
  "versioning": [ { "commit": "", "asOf": "2023-06-01" } ],  
}
```

```
SELECT  
  dates,  
  MtmAggreg,  
  scenario,  
  country  
FROM  
  (  
    SELECT  
      country,  
      scenario,  
      MtmAggreg,  
      mtms_mtms_datearray_dates  
    FROM  
      (  
        SELECT  
          dictGetString('csald',  
            'country',  
            toUInt64(csald)) AS country,  
          scenario,  
          sumForEach(arrayMap(x -> x * factor,  
            mtms)) AS MtmAggreg  
        FROM  
          mtm_facet_csa  
        WHERE  
          (toUInt64(csald) IN (  
            SELECT  
              csald  
            FROM  
              dict.csald  
            WHERE  
              (csa IN ('BANK01')  
                AND country IN ('CANADA'))  
              AND ((Date = '2023-06-01'  
                AND commit IN (1,  
                  xyz))))  
          GROUP BY  
            country,  
            scenario) ARRAY  
        JOIN MtmAggreg,  
          arrayEnumerate(MtmAggreg) as  
          mtms_mtms_datearray_dates) ALL  
      INNER JOIN (  
        SELECT  
          dates,  
          mtms_mtms_datearray_dates  
        FROM  
          datearray ARRAY  
        JOIN dates,  
          arrayEnumerate(dates) as mtms_mtms_datearray_dates )  
          USING mtms_mtms_datearray_dates)
```



Benchmark	Success Rate
Spider 1.0 (2017)	92%
BIRD (2023)	74%
Spider 2.0 (current)	15%

Thanks to the semantic layer, we can simplify our use case to a use case like BIRD / Spider 1.0.

NOT STAC BENCHMARKS

Define clear responsibilities, context and controls

MASTER AGENT

Based on a state-of-the-art LLM with ability to plan, execute those plans, evaluate outcomes, and adapt accordingly, but **should not substitute to specialized agents**

SPECIALIZED AGENTS

Provided with context, rules along with benchmarks and quality controls. Using the right LLM, if any, for the right task

Provide LLM the necessary information for successful reasoning in your particular knowledge domain
i.e. trading analytics

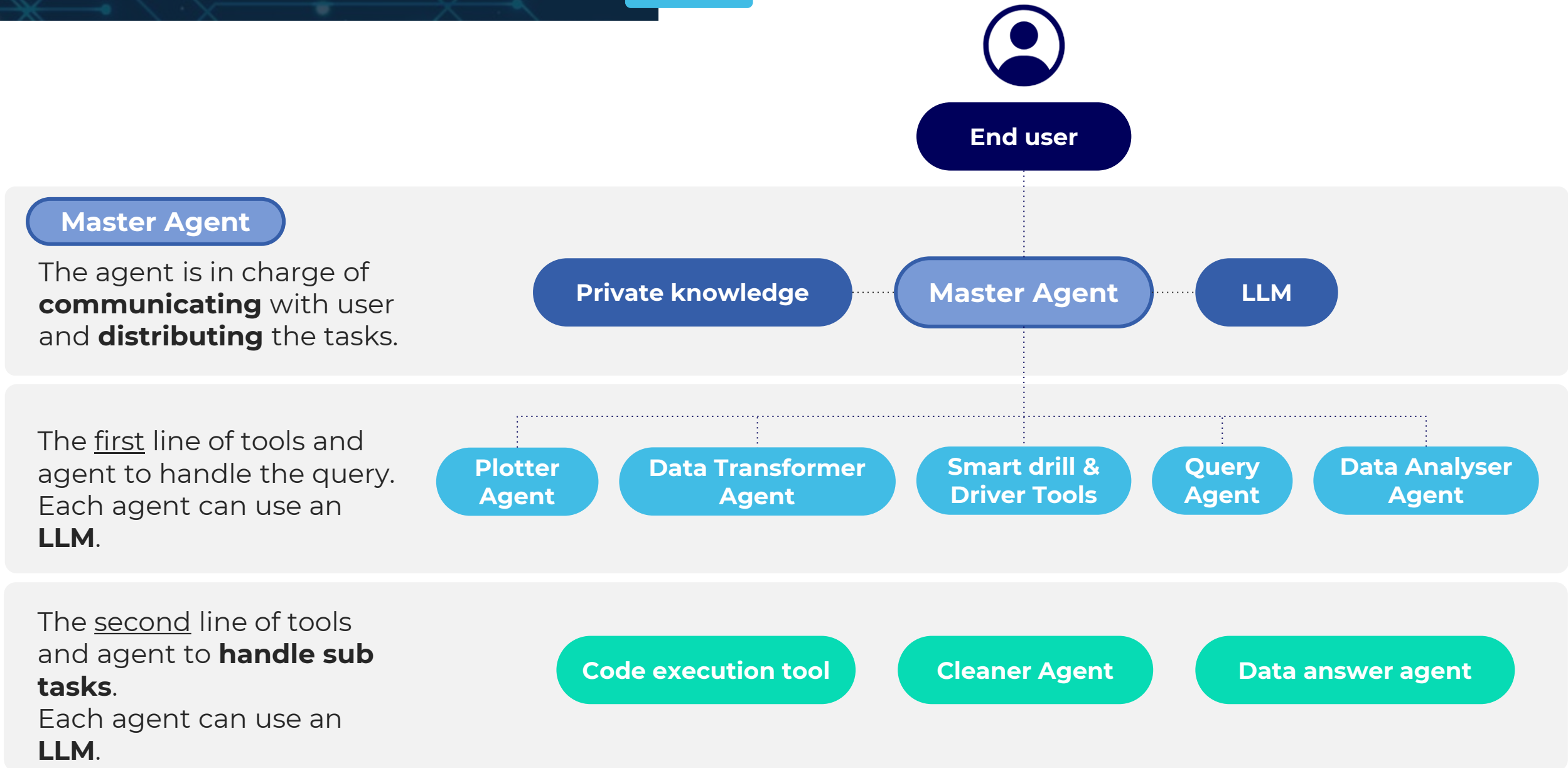
Make the LLM an expert in the specific domain with...

Prompt engineering

RAG and private knowledge

To adapt to specific data and semantic used by the user

Fine tuning for generic knowledge of the task



BENEFITS

Multi-Agent Structure and Semantic Layer

- **Accelerate build** of Analytics solutions for Risk, Finance, and Trade, while maintaining **full control**
- Reduced risk of errors and hallucinations
- Provide users an intuitive interface to get accurate, actionable insights from complex data

THE 'EQUITY ANALYTICS' DATASET

OPENSEE DEMO

17,000,000 market data per day

on **600** stocks (STOXX600)

250 days of history

2,700 parent orders per day

5 billion data points

**Simulation of a large AM
business hierarchy:**

- 20 Funds
- 10 trades

How many orders has been executed and the total volume?

Okay, let's start the analysis for the 22nd April 2022. To get the number of orders executed and the total volume, I will make the following tool call:

```
{"tool": "query_tool", "args": {"metrics": ["count_ParentOrderId", "sum_OrderExecQuantity"]}}
```

This will query the `TCA_ParentOrders` table, and return the count of `ParentOrderId` (which represents the number of orders) and the sum of `OrderExecQuantity` (which represents the total volume).

count_ParentOrderId	sum_OrderExecQuantity
2,705	440,920,731

Okay, let's get the 10 most executed instruments by volume on 2022-04-22.

First, I'll make a query to get the top 10 instruments by total volume:

```
{"tool": "query_tool", "args": {"metrics": ["sum_MktVolume"], "by": ["Instr:"
```

This will give me a dataframe with the instrument symbol and the total market volume for each. I can then sort this and take the top 10 rows:

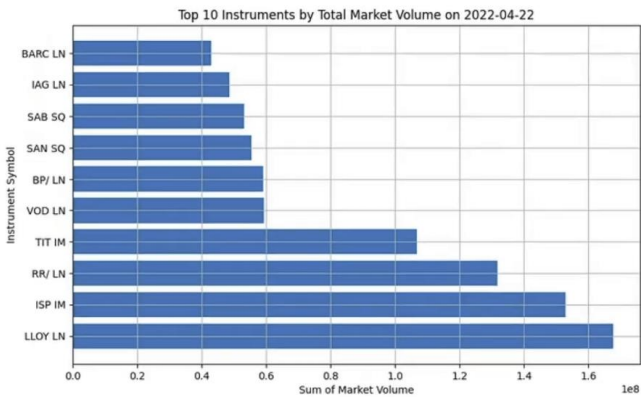
```
top_instruments = df.sort_values('sum_MktVolume', ascending=False).head(10)
```

Now I can create a table and a bar chart visualization:

```
{"tool": "smart_coding_plotter_tool", "args": {"description": "Create a bar
```

The resulting table and chart will show the 10 most executed instruments by total market volume on 2022-04-22.

InstrSymbol	sum_MktVolume
LLOY LN	167,675,767
ISP IM	152,780,451
RR/ LN	131,709,320
TIT IM	106,678,925
VOD LN	59,252,220
BP/ LN	59,095,067
SAN SQ	55,495,936
SAB SQ	53,198,918
IAG LN	48,570,257
BARC LN	42,957,502



Solution extends to many use cases

DATA MANAGEMENT AT SCALE AND REAL-TIME ANALYTICS



**MARKET
RISK**



**MARGIN
MANAGEMENT**



**MARKET
DATA**



**PRE & POST
TRADE ANALYTICS**

THANK YOU!

Get in touch with us



Jean-Philippe Rezler
Global Head of AI & Analytics
jean-philippe.rezler@opensee.io

Demo request
[**info@opensee.io**](mailto:info@opensee.io)

Visit our website
[**opensee.io**](https://opensee.io)