Foundations

Problem - (link to presentation)

Velocity of Decision making where data insights or data validation is needed

- Cost of Analysts
- Back and forth on standard views created
- Intellectual gaps and context trasnfer
- Hiring timelines

Customer

Decision maker in a medium/large organisation with large amount of complex data

Oragnisations to target

- Working with technology products - have complex system generated data
- Relatively big in scale - more data to analyse
    - While number of employees is not always a relevant proxy for the scale of the org, the organisations that we intend to work with (e-commerce/Retail/D2C) would scale employees as their scale grows, thus orgs above 50 employees atleast
- Have heavy business analysis need as a delegated task
    - Younger orgs - 50 to 200 employees - The number of delegated resources to run analysis are less, but the need for analysis is high
    - Relatively more mature orgs - 200 to 1000 employees - The number of analysts compared to overall emplyees is between 1-5%, and on the upper side for retail orgs compared to tech or service orgs. So we're talking about orgs with more than 10 dedicated analysts who would be using a dedicated tool already to write and run SQL queries.

- E-commerce, retail, D2C brands - similar data, faimiliarity with the segment, better connects for initial GTM, leverage of building common foundational models as add-on services to optimise cost

Decision Makers and Influencers

- Decision Maker - CTO, Head of Technology, Head of Product, Head of Data Engineering, Director of Analytics, Data Platform Owners
- Influencers
  - Organisational Influence
    - Founder, CEO, COO, VP, SVP
  - Tech Influence
    - Architects, Data Scientist, Principal Devs, Lead Data Engineers
  - Business Influence
    - Business head, Category lead, Product Lead

Competition

Organisational work-arounds/solves

- Business Analysts
- In-house Consoles/Reporting dashboards/SAP Solutions
- Intuition

Platforms/Tools

- BlazeSQL
- Hex AI
- Julius AI
- Wren AI
- Databricks

    - AI/BI Genie
    - Mosaic AI
- Snowflakes
    - Cortex AI
- Amazon Q
- Microsoft co-pilot
- OpenAI for Businesses
- Google Data Platform
- text2sql.ai
- MindsDB
- Alteryx
- DataRobot
- Devrev, Whatfix, Glean, coworker.ai (bunch of other agent building orgs, very crowded market)

Insight

- Either too surface area, or too deep into the end goal of building multiple AI agents for people
    - Databricks and Snowflake are giving ADKs that can be used by devs
        - Bigger orgs in-general focus of core capabilities, not smaller nuances for enterprises
    - DevRev, Whatfix, Glean are deliveirng end agents that can be used by users, it's case by case basis, and pretty expensive
- No tool/platform is solving for the last mile of how to ensure that apart from the context gained by column names, sample data and historical queries, how do we ensure that for an enterprise level solution, we have 100% accuracy of the data.

    - They rely on dev teams to build the agents, and for analysts to add context using queries (reactive vs pro-active)
- For a business leader to use a solution, they need 100% trust on the results
    - Something nobody in the industry has been able to solve for
    - The current personna is that of a robot built by the central technology team that will help everyone (Point of trust/failure)
        - Or something that a very specific team has built for themselves and has a limited scope
        - Then there are solution providers who will build everything end to end - this is the ideal one, and needs a lot of investment; if the org can afford that, we are not the right platform for them
    - We need to acknowledge hallucinations and build with them, not around that
- Absence of ecosystem of tools that support the agents to be extended to enterprise end-users
    - Not focusing on the way an analyst extends context to AI - prioritising the art of asking the right questions, building the scope of the project and focusing on just that
    - Fail-safe flows for analysts to step in when the response needs validation

Motivation

- I've spent most of my career as a decision maker who has to struggle getting data quickly, and taking calls. With AI, the basic roadblocks of going through each column of data to extend context has been removed, and there is an opportunity to redefine how business analytics fundamentally works at organisations as of today.

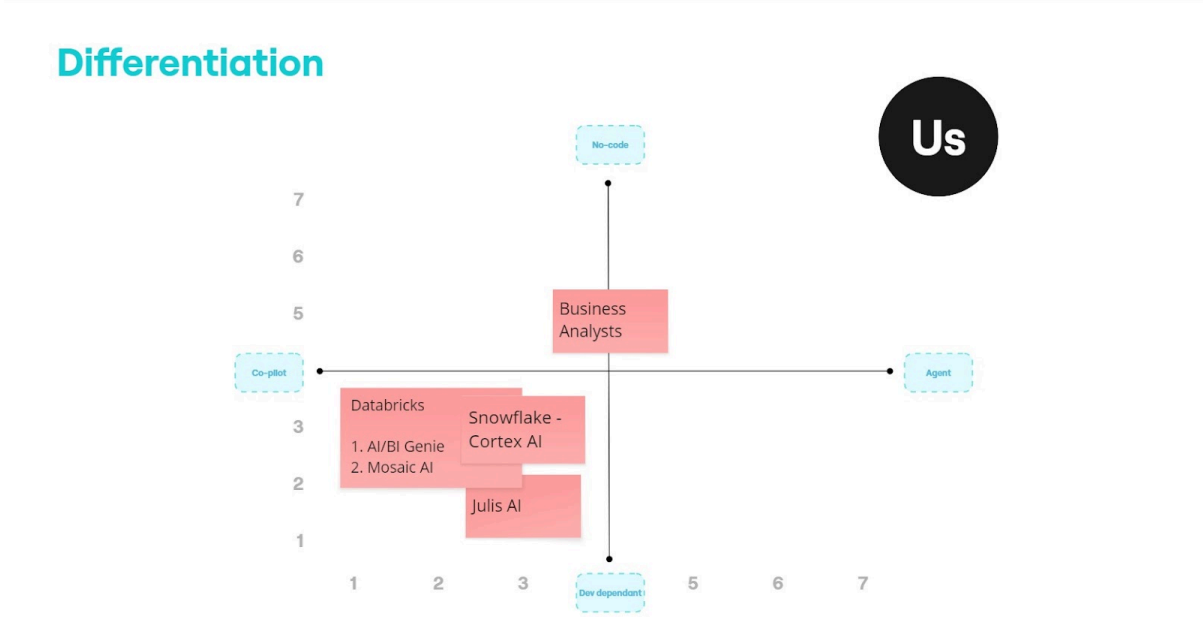
Differentiation

Agent vs Co-pilot

- We intend to deliver agents as the end product; they can be powered by a co-pilot, but for our end-users, agent is the right solution. Co-pilots are the most ideal choice if we build for today, as we expect the user to make changes in the code and then run it. We want to build a tool that gets better with better inputs; inputs here being context extended by analysts, and LLMs being able to work with larger context windows and not hallucinating with higher tokens. We build it with hallucinations, not around them, and expect organisations to adopt overtime with some parts of analysis being completely automated, vs some part being looked up by data architects (what data analyst role will evolve to, in future)

Open ended last mile vs No-code last mile platform

- Most of the platforms are industry mammoths that thrive on their horizontal capabilities and massive infrastructures, they have left it to devs and analysts to build agents and give them context. This means
    - Friction in how devs build the agents and how they interface with analysts to gather the context and add that to agents
    - Need of additional tools for the best experience
        - Editing/Adding context to data
        - Console to retrieve conversations/data-fetch across multiple integrated platforms (essentially a homepage for your agent)
    - Dev team needed everytime a new agent has to be built
    - Analysts having to go off the platform to audit/validate correctness of certain data points



Approach

It is clear that we will be developing multiple layers of context that will eventually lead to the end-users being able to trust the data

1. Automated/System Generated - Use table structures, column names, sample data, existing queries and code repos (optional) to populate the first layer of context to your data
2. For the second layer of context, we can have two ways
    a. Solution approach (for the first 4-5 design partners) - Work like a consulting firm. You do an initial effort on building the second layer of context manually with the analytics team, and provide them with an agent and a co-pilot (the reason I've added co-pilot is because it will soon enough become a standard offering, and we don't want our users to have multiple platforms to jump to)
    b. Co-pilot + Agent Builder + Agents - Let Analysts become agent builders where they make data models (that contribute to our agent) as easy as a powerBI dashboards using a no-code tool
        i. Link (this is a very crude prototype, to give a basic idea)

ii. Point of trust/failure moves to analysts rather than dev teams; as a business leader, you are more likely to trust an agent built by an analyst in your team rather than a developer building it across the org

iii. It will be a one-time effort to build the data models, post which you dont need to dedicate time in writing queries on these tables mostly

1. So the expectation is not to remove analysts, but rather replace the redundant part of their work, and shift the expectations of their role more towards monitoring the quality of data, maintaining data models, and debugging + training agent outputs for complex queries.

We place the above product as the upgraded way of working for business analysts across all organisations, make it available as an open source tool with a minimal (ballpark 1000 USD/month) licensing fee for enterprises

- Given that the compute, storage and LLM are all isolated to the user's own backend, data is completely secure
- We will sit on top of the existing data warehousing solutions like Databricks and Snowflakes. Most of the orgs use them for these services, and not SQL query editing.
- There will be regular upgrades and cloud versions also, where we will host everything with us, but given that it needs a lot more capital, we want to lay that off for sometime
- There will be additional services on top of this, that will end up being key revenue generators for us
    - Token Optimisation
        - Developing simpler more focussed foundation models for less complex tasks not involving user interaction
    - Agents for Marketing/Finance and multiple other organisational functions
        - Now that you have an analytics agent, it'll be amazing to extend your internal APIs to the tool, and have it configure services using the

insights generated with the user being in loop for the approvals and context addition.

Proof of concept

1. Will AI Agents be able to understand complex business use-cases and generate queries for the same? - Let’s add context to data and see if agent is able to answer the below shared set of questions - Working Doc
2. If the users are given an agent to work with, will they start using the conversational chat or still stick to traditional methods of asking their analyst to share the data with them? - Let’s expose the agent to business teams using the consoles extended to them already
    a. This testimony by Julius AI founder is what validates this - Working Doc
3. Does adding manual context help AI agents be more accurate? - Working Doc
4. What should the manual onboarding of data look like? How do we make it more actionable for the users? - Working Doc

Product

Link to prototype - helps in visualising!

Users

1. End-users - These are business teams who are expected to have conversations and consume the data insights for decision making.
2. Agent Builders - These are analysts or data scientists who will be building these agents by creating data models. They will be montioring and debugging the conversations made by end-users as well.
3. Developers - They will be responsible for the tool's integration with data pipelines, LLM and communication tools that the end-user will the accesing the agents at.

Three pillars

1. Exisiting data platform capabilities + co-pilot
2. Context Builder - This tool is connected to your database and LLM provider. It's a tool for your analysts, to add exhaustive context to the tables and create data models that are relevant for their stream of business. These data models will be our key context for the analytics agent.
3. Analytics agent - This is the tool that will be exposed to our end users, and is responsible for extending business insights using a conversational medium. It is expected to operate like coding agents (imagine claude code) where you break down the problem statement into multiple steps, intrinsically query the data, and share the response with the user.

Key Features

1. Data Platform
    a. Schema browser + SQL Editor
    b. Co-pilot for the SQL Editor
2. Context Builder
    a. Database connector
    b. System context - using Schema, DDL, imported queries and Statiscal analysis (nullabilty, cardianlity, etc.)
    c. LLM generated context - using system context plus data sampling
    d. Human Dialogue - LLM generates hypothesis and validates deeper nuances from the agent builder. Examples are below
        i. We noticed that 535 rows have null values in this column, can you please help us understand in what case would this happen?
        ii. We noticed that this column has 35 unique event names across the rows, we have anticipated the context for them, can you please review the same?
        iii. We noticed that the three columns main_order_id, order_id and order_line_id are present in the same table, can you please help us identify the difference between the three columns?

e. Relational context - Establishing relations within the data models and also between other data models is super important. All of the above context should help us suggest most of these joins, and then the intention is to validate all of these with the agent builder.

f. Persona context - The data model's persona context will sit at the top of our context's hierarchy and help the agent maneuver amongst mutiple data models built by different users, and understanding the most relevant ones to further deep-dive into. An exaple would be - "I am a GTM sales agent that understands all the orders placed on our website. I understand the different users who placed these orders, and the products that they placed it for. ". The exact stored data would be much more compact, and might only have keywords.

    i. Jargons - We will import slack queries and outlook mails to understand commonly used metrics that are being referred uning non-traditional jargons, and check with agent builder on the definitions of the same

    ii. Commonly fetched data - Saving views or queries that have been requested for multiple times. Buyer can have an option to add these manually or import.

g. Storage - All of the context generated and the respective human feedback received is compiled using LLM and stored in the system, to be accessed by MCP and/or by the agent for RAG. (Choice to be made between Knowledge Graph database vs Vector Database)

    i. For lower cardinality columns, we will be storing their distinct values as context nodes

    ii. There will be only one single source of truth for the context against a table, column and value. Agent builders will be asked to skip pre-populated values, coulmns and tables, or be given an option to raise a conflict which can be resolved by admins. There can be as many relations for a single cell.

    iii. Any changes in data schema also need to be highlighted to the agent builders, who in turn are expected to make corrections if needed

3. Agent

a. There will be one agent that is available to all users
b. The data models built by different analysts will have restricted access, and the agent is expected to use only authroised data models for it’s context for a specific user
c. Generated queries will be saved in a separate DB, and be used for building future queries, to avoid building new queries from scratch everytime
    i. Context of these queries will not be limited at user level, but rather to all who have shared access to the involved data models (inner join, not union)
d. For new queries, the agent will start from the first hierarchy of personas to choose amongst multiple data models, then move to tables, columns and values respectively.
e. The agent is expected to break the problem statement into smaller steps and build queries for that (taking inspiration from Claude code’s agentic chat)
f. There will be an option to clip different tables, views, and metrics similar to a dashboarding tool, with an option to conversationally deep-dive into the same
g. While the generated queries will be used for global context, approach towards different problem statements and commonly used metrics by the user will be stored as local context to have a more hands-on retrieval for the same
h. Users will have the ability to map their analysis to different Agent builders, who in-turn will have the ability to mark a conversation as “Verified”.
    i. This allows the users to play around with the tool while not comprimising on their current flows. The eventual expectation is for users to figure out a sweet spot in terms of what level of analysis they can trust with the agent, vs where they need an analyst to step in.
    ii. We can also have semantic scoring, where we automatically assign the conversation to the owner of the primary data model, expecting hallucinations highlighted by the confidence score.

i. Agent will also behave as a feedback mechanism where a user can share feedback with the agent builder, who in turn can share the problem with the query by pointing the exact problem or sharing the right query.

MVP

For the MVP, we will not build the entire platform, but rather focus on the two pillars of context builder and agent in isolation. This will help us validate the points we intend to find POC against. Even within agent, we can choose to isolate to single user features not expecting teams to collaborate on the MVP.

Researches

Context Builder

Pitch Deck