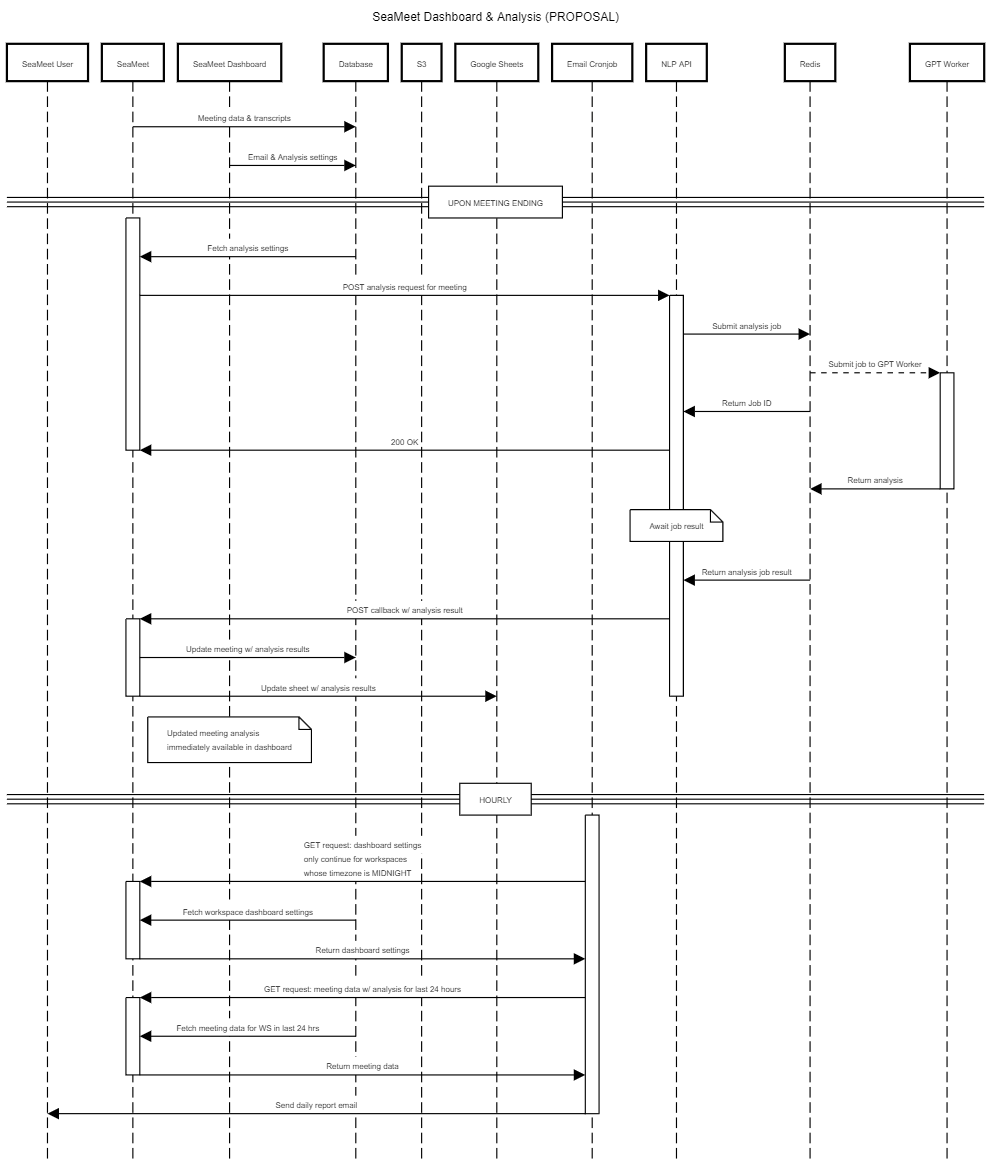
SeaMeet Dashboard Proposal to Replace Google Sheets

# Overview

We have developed a PoC feature for SeaMeet called SeaMeet Dashboard. The dashboard allows customers to create custom analytics for their business, analyze all their SeaMeet meetings in real-time, and then view the aggregated data via Looker Studio embedded in SeaMeet. The architecture of the analytics processing can be seen below. For an example, see the [Orting SeaMeet Dashboard](https://meet.seasalt.ai/workspace/5397f35a-1bb7-4f73-ae54-cbb55f6d96cd/dashboard) .



See the original proposal here: [SeaMeet Dashboard & Analaysis Pipeline Improvements](https://docs.google.com/document/d/1uaODLf9BnFZhuy_7wrs_8n_LaWsDa9Eekul1fyKUDVQ/edit)

# Issues with Google Sheets

Google Sheets, while convenient for quick prototyping and small-scale data management, has proven to be unsuitable for the long-term needs of our SeaMeet Dashboard application. Several key limitations have emerged:

* Reliability Issues: The Google Sheets API is not robust enough for production use. It often fails silently, leading to operations not being executed without providing error messages or alerts. This lack of reliability undermines the integrity of our data management processes and can result in data loss or inconsistency.
* Limited Scalability: Google Sheets was not designed to handle large volumes of data or high loads typical of production environments. It imposes upper limits on the number of rows that can be added, and the API does not provide robust handling for concurrent calls, which inhibits our ability to scale our analytics capabilities to meet the growing needs of our customers.
* Poor Performance: Accessing data in Google Sheets is inefficient, especially when dealing with sizable datasets. Fetching entire sheets and iterating through rows is slow and resource-intensive, leading to suboptimal user experiences and sluggish performance in real-time analytics scenarios.
* Security concerns
* Manual Work: A lot of manual work required for setup of each customer.

In light of these challenges, it is evident that Google Sheets is no longer a suitable choice for our application. We must explore alternative data storage and management solutions that offer greater reliability, scalability, performance, and advanced functionality to support the continued growth and success of SeaMeet Dashboard.

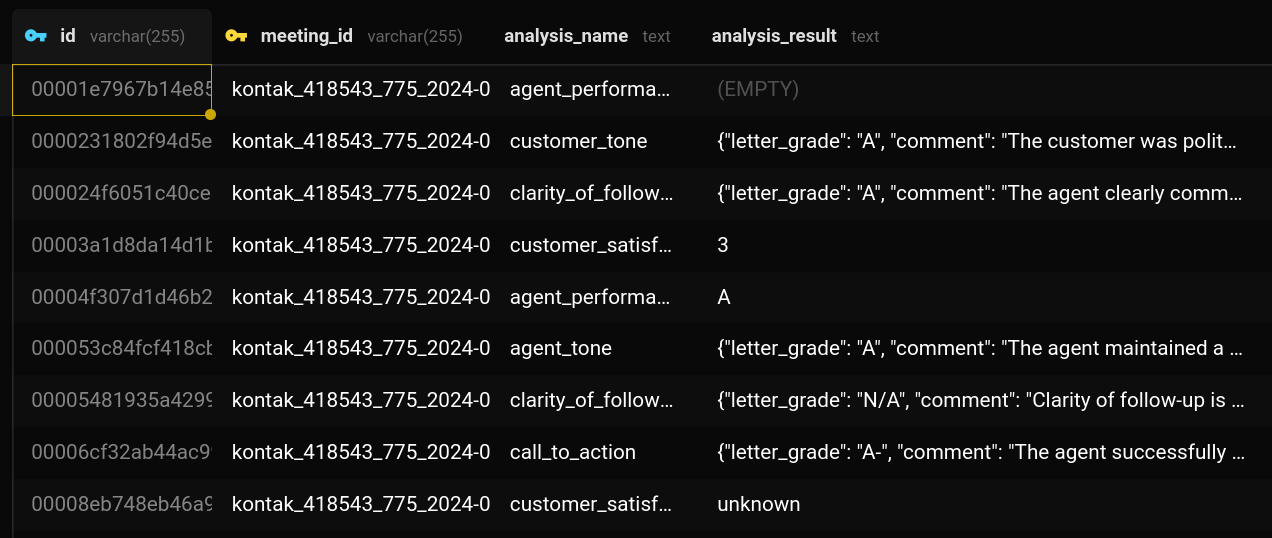
# Proposal

Here are the existing Looker Studio Data Connectors that we could possibly utilize: <https://lookerstudio.google.com/data>

## Option 1: Postgres DB

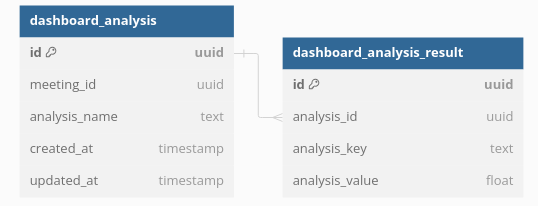
Suggested by [Kim Dodds](mailto:kim@seasalt.ai)

We currently use PostgreSQL to store our data when we get the analysis results from the NLP server. So you may think: “Why can’t we just directly connect Looker Studio to the existing database?”. The current design of the database table *will not work directly with Looker Studio*. Originally, we had some difficulty deciding how to store the data, since each client may have different requirements and different analysis results. Here is an example of the current structure of the data in the database:



Originally, we weren’t sure what the structure of the analysis would be for each client, so we simply stored the analysis result as a string. As you can see in the example, some `analysis\_result` values are individual, while others are json strings of more or more key-value pairs.

However, I think it is possible to coerce all the analysis results into key-value pairs, and then break the analysis down further by making each key-value pair in the analysis\_result its own row in the DB.



So for example, two sample entries in the current DB structure look like this:

| id | meeting\_id | analysis\_name | analysis\_result |
| --- | --- | --- | --- |
| 000024f6051c40ce8a37b2d34df57059 | kontak\_418543\_775\_2024-05-22-14:05:10.979000 | clarity\_of\_follow\_up | {"letter\_grade": "A", "comment": "The agent clearly communicated the new appointment time and confirmed the customer's understanding."} |
| 00003a1d8da14d1b8df2152fb9b81de1 | kontak\_418543\_775\_2024-05-22-14:05:10.979000 | customer\_satisfaction | 3 |

We could further break down the analysis\_result of the first item, and then ensure that the second item has an additional label/name for the specific type of analysis. The final result could look like this:

| id | meeting\_id | analysis\_name |
| --- | --- | --- |
| 000024f6051c40ce8a37b2d34df57059 | kontak\_418543\_775\_2024-05-22-14:05:10.979000 | clarity\_of\_follow\_up |
| 00003a1d8da14d1b8df2152fb9b81de1 | kontak\_418543\_775\_2024-05-22-14:05:10.979000 | customer\_satisfaction |

| id | analysis\_id | analysis\_key | analysis\_result |
| --- | --- | --- | --- |
| 111111111111111 | 000024f6051c40ce8a37b2d34df57059 | letter \_grade | A |
| 22222222222222 | 000024f6051c40ce8a37b2d34df57059 | comment | The agent clearly communicated the new appointment time and confirmed the customer's understanding. |
| 33333333333333 | 00003a1d8da14d1b8df2152fb9b81de1 | score | 3 |

Now, Looker Studio has the ability to directly execute complex SQL queries onto the database, no matter the differences between clients (each looker studio dashboard and queries would still have to be customized for each client).

* **Postgres Pros**:
  + **Reliability**: PostgreSQL is a mature, stable, and highly reliable relational database system
  + **Fits with Existing Systems**: We’re already using Postgres to store this data, we just need to refactor the data structure. No new tools to learn
  + **No 3rd Party:** Keep it in-house, no paying for 3rd party tools
* **Postgres Cons**:
  + **Refactoring**: Setting this up initially will require refactoring to the existing data pipeline & database tables
  + **Load**: It’s possible if we have a lot of clients, or clients with a lot of data, that this add load stress to our database and possibly impact production services (such as SeaMeet, since they share a database).

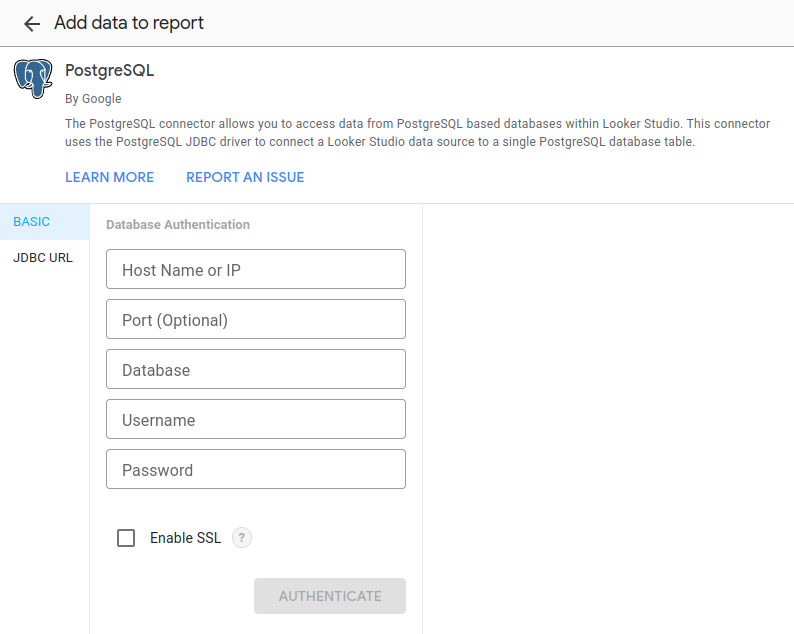
The only remaining question for this implementation is *how to connect looker studio to our postgres DB* - how can it perform the queries?

### 1a: Use the Existing Postgres Connector

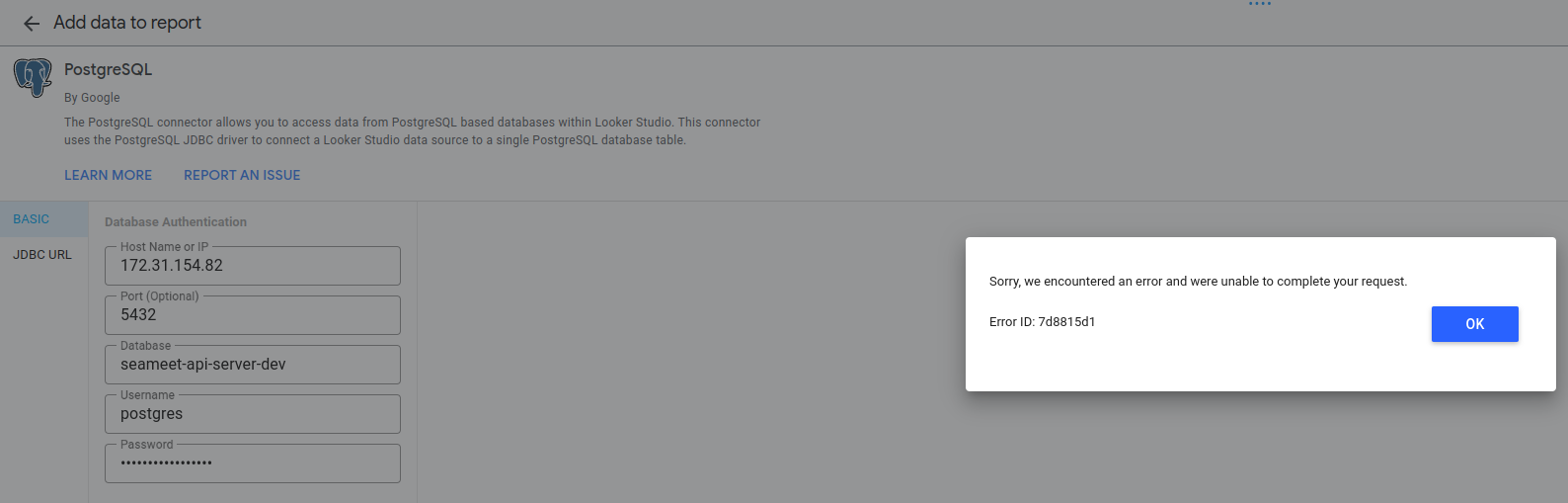
[Connect to PostgreSQL - Looker Studio Help](https://support.google.com/looker-studio/answer/7288010?hl=en#zippy=%2Cin-this-article)

[Enabling secure database access | Looker | Google Cloud](https://cloud.google.com/looker/docs/enabling-secure-db-access)

We can leverage the existing PostgreSQL connector provided by Looker Studio. This connector allows us to directly link our PostgreSQL database with Looker Studio, enabling seamless data visualization and analytics.



It should be very easy to connect to the DB, just drop in the credentials. However, of course it didn’t work for me… (on second thought, this is probably because it needs VPN access - not sure the best way to solve this)



* **Benefits**:
  + **Minimal Engineering Work**: This connector should be ready to go out of the bot (as long as it works…), and should require essentially no additional engineering time. It’s maintained by Google.
* **Challenges**:
  + **Relying on 3rd Party**: It’s nice that we don’t have to maintain it, but then we’re also relying on Google to implement new features or bugfixes

### 1b: Build Our Own Looker Studio Connector

[Looker Studio Community Connectors | Google for Developers](https://developers.google.com/looker-studio/connector)

Develop a custom Looker Studio connector tailored to our specific needs, providing greater flexibility and control over data integration and presentation. I think this is overkill at the moment, but it’s something to keep in mind for the future.

* **Benefits**:
  + **Customization**: Fully customized connector that can be optimized for our specific data structures and use cases.
  + **Integration**: Ability to integrate with other systems and data sources as required.
* **Challenges**:
  + **Development Effort**: Requires significant development resources to build and maintain the custom connector.
  + **Time to Market**: Longer implementation timeline compared to using an existing connector.

## Option 2: 3rd Party ELT Pipeline (ex/ BigQuery)

Suggested by [Block Chen](mailto:block@seasalt.ai)

While our data currently resides in a PostgreSQL database, integrating Google BigQuery can provide additional benefits for large-scale data analysis and real-time analytics. We could continuously sync data between PostgreSQL and BigQuery (or some other data warehouse). This ensures that data in BigQuery is up-to-date and reflective of the PostgreSQL database. BigQuery is part of the Google ecosystem, so it should integrate with Looker Studio easily. It also offers tools like analytics and reporting.

* **Benefits:**
  + **Advanced Analytics**: BigQuery offers powerful analytical capabilities, including built-in machine learning tools, advanced SQL functions, and support for complex queries that can enhance our data analysis processes.
  + **Scalability**: BigQuery is designed to handle petabyte-scale datasets, providing near-infinite scalability. This can be advantageous if our data volume grows significantly or if we need to perform large-scale data processing tasks.
  + **Performance**: Optimized for high-speed data querying, BigQuery can significantly reduce query times for large datasets, ensuring faster insights and better real-time analytics performance.
  + **Managed Service**: As a fully managed data warehouse, BigQuery reduces the operational burden associated with managing and scaling a large database infrastructure. This allows our team to focus on data analysis rather than database maintenance.
  + **Seamless Integration with Google Ecosystem**: BigQuery integrates seamlessly with other Google services, including Looker Studio, enabling streamlined data visualization and reporting.
  + **Future-Proofing**: Prepare for future growth in data volume and complexity by integrating a scalable and robust data warehousing solution.
* **Challenges:**
  + **Costs**: This is a paid 3rd party service. There are costs associated with data transfer between PostgreSQL and BigQuery. Optimize data sync processes to minimize unnecessary data movement.
  + **Data Security**: Ensure that data transfer and storage comply with security and privacy regulations. Implement appropriate encryption and access controls.
  + **Learning Curve**: Training may be required for team members to effectively use BigQuery’s features and tools.
  + **Overkill:** This is a robust product that offers a lot more features than we need - and some features which we’re trying to implement ourselves. At the end of the day, will we just end up reselling BigQuery instead of our own product?

## Option 3: AirTable or Zapier Table

Suggested by [Xuchen Yao](mailto:xuchen@seasalt.ai)

Use AirTable or Zapier Table as an alternative data management solution. These platforms offer user-friendly interfaces and automation capabilities, simplifying data handling and integration with other tools.

After some research, I feel these products are fully fledged customer facing solutions. We are looking for a simple data storage solution to integration with a visualization dashboard. Using AirTable or Zapier Table would basically consist of reselling their product. If we had customers interested in these products (say they already are users). It would make more sense to create a customer connector to funnel our data into these services (instead of offering our own dashboard to the client).  
[Integrating with Airtable - Overview](https://support.airtable.com/docs/options-for-integrating-with-airtable)

# Discussion

## Kim’s Thoughts

It seems to me that a data warehouse like BigQuery is a full solution that we would basically be reselling to the client. We'd probably end up dumping all of the meeting data and analysis data into BigQuery, and then we'd be selling the client the visualization tools built into that like LookerStudio. That's nice because maybe the client has other data sources that can also be connected to bigquery and they could see it all combined. But at this point, it's not really our product - we're just being contracted to set up and manage bigquery for the client. Unless there are a lot more features that would be useful to us as a company to start using BigQuery, I think it is a bit overkill for this application.

Similarly, things like AirTable or Zapier Table are already fully-fledged SaS products. We could certainly integrate with these products so customers can use them with their data from Seasalt, but the dashboard wouldn’t be our product anymore.

For these reasons, the database seems like the best choice to me. I think if we restructure the data, we can connect our DB directly with a visualization tool like Looker Studio. All the data would still remain under our control and we would not be incurring any extra cost from 3rd party platforms.

## Block’s Thoughts

I feel that if people can pay for the dashboard we can use mature tools to achieve to goal.

for example, data ==> databases ==> bigquery (data warehouse solution) ==> looker studio. the dashboard feature is a kind of ETL implementation (<https://www.databricks.com/glossary/extract-transform-load>). A key point here is that Google sheet doesn't extract data but accepts the updates, and we can replace Google sheet with BigQuery, so that we can pickup(extract) the data from database. BigQuery is an example here, we can compare many mature Data warehouse tools to choose from; BI/ETL/data warehouse tools were not usually free, so I would check if we can have some budget for it and if our user can pay for it

After careful consideration, my viewpoint is:

Since it is confirmed that customers are willing to pay for this new feature, as long as we can ensure that using a paid tool still allows us to retain profitability, we should choose the most convenient and efficient way to implement this new feature. This will allow the developers to quickly refocus on optimizing our existing products and tackling major challenges. We should not allocate excessive manpower to a new feature that can be achieved with already mature and out-of-the-box tools.

For example, instead of incurring additional database costs for this new feature, why not allocate this expense, along with the subsequent maintenance labor costs, towards those reliable and out-of-the-box tools?

## Guoguo’s Thoughts

Yeah I’m a little bit worried that things like BigQuery might be a overkill. At this point we really just need to save the data for presentation. And if we are going to use that (SeaMeet will be the user), I feel we will be the user of that, instead of reselling them to the users.

Tom’s Thoughts  
  
I would choose 1a in the context.

I am inclined to support using the existing PostgreSQL connector for the following reasons:

* Minimal Learning Curve: Our team is already familiar with PostgreSQL, and using the existing connector does not require learning new tools or technologies. This minimizes training time and costs, allowing the team to quickly get up to speed and start working.
* Fastest Implementation: The existing PostgreSQL connector can be integrated quickly, reducing the overall implementation timeline. This ensures that we can deliver the new feature to our customers faster, meeting their needs promptly.

Addressing VPN Access Concerns

If there are concerns about VPN access, we could consider setting up a dedicated database that only contains the meeting\_analysis table. This dedicated database can be configured to allow access from Looker Studio, ensuring that our primary database remains secure and performance is not impacted.