【小专项】数据业务竞品设计洞察

1. 设计师做竞品研究的目标愿景

我们有两个方向可以在竞品研究这件事情上拔高设计团队对业务贡献的价值（这里竞品洞察的价值既可以是内化，也可以是向外展现）：

一、纵向发挥设计师的独特视角，深挖竞品体验维度的特征：

这里要问的问题是：我们做竞品研究与产品或商业部门做竞品分析的区别是什么？或者说有哪些是设计师可以看到的Insight而其他职能人员看不到的？那些Insight对于产品的商业成功有什么价值？

案例一：本产品只是根据查询语句生成Audience的query结果数量；而竞品Adobe Campaign的画像条件编辑工具可以实时提供可视化的交/并集展示帮助用户看到不同条件语句之间的重叠空间，从而对不同属性的受众体量有更加直觉化的理解。（这种insight属于功能层面，是PM在分析竞品时一定会注意到的，作为设计洞察价值比较低；UX可以为用户争取相关功能，但需要了解它对业务的帮助有多大。）

案例二：本产品对于查询语句提供了一些模版，方便常用的查询需求；竞品Mailchimp也把常用的查询语句作为模版，只是进一步包装成更符合用户职业的语言描述，并且放置在流程的最前端，这样做为小白用户提供了更人性化的体验。（这种insight在于体验的细节层面，PM在做竞品分析时可能注意不到；如果能明确改善用户体验的话，作为设计洞察的价值较高。）

案例三：本产品的数据流配置采用五步式的线性流程；而竞品XXX采用从summary页面分别进入不同的配置分页进行设置，同样实现了操作任务，但后者可以让用户不必按照顺序执行步骤，并且在事后编辑时可以先看到整个配置内容中的关键信息再进入分页进行编辑，而不用进入具体步骤查看。（这种insight在于具体的交互设计层面，PM在做竞品分析时会注意到但是不太在乎，UX需要分析这方面变量对于产品整体体验的影响如何，作为设计洞察的价值较高。）

二、横向探索竞品带给客户的完整服务体验

其他团队在竞品研究方面做得不足的地方我们进行补足，包括但不仅限于营销策略、服务、价格、技术能力。鉴于我们的职业优势，可以从客户体验的视角看待这些设计之外的事情。

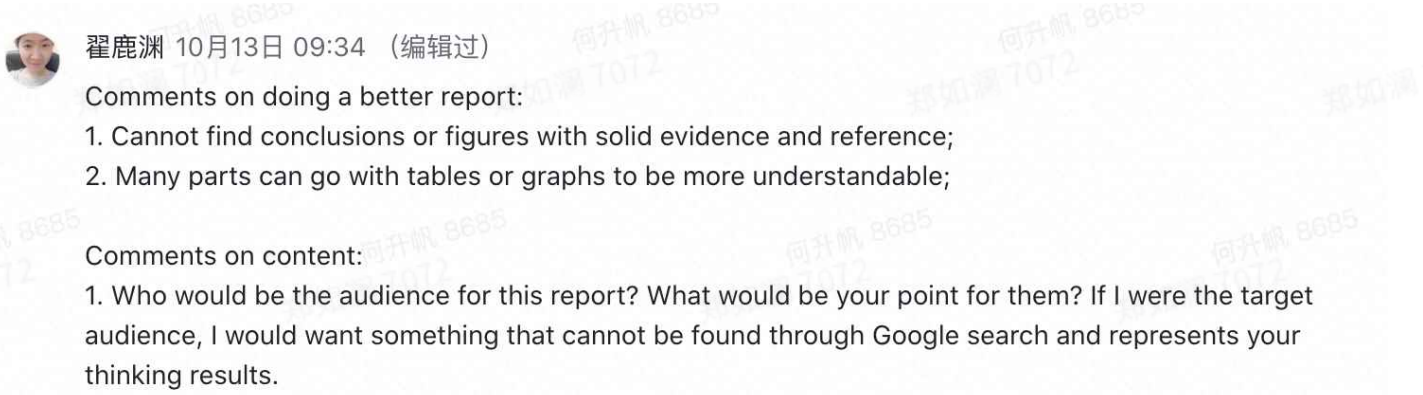
案例一：在商业策略上某公司先向小客户提供半年免费使用的优惠来创造success stories，进而采用这些营销素材来吸引大客户的加入。这样做容易入手但也有一些风险，比如一年之后大客户的市场已经被竞争对手占领，或者小客户运营不下去了拒绝续费。（设计相关度：低）

案例二：A公司提供较全面的能力覆盖跨部门的软件需求，而B公司专注于某一个领域的需求并提供丰富的上下游接口能力。（设计相关度：中，需要用研来定性分析各个行业的客户在不同场景下对这一问题的偏好如何。）

案例三：A公司有较大的服务团队来解决客户的运维需求，而B公司强调开发很多Self Service的功能来教育用户自己完成一些运维的动作。（设计相关度：高，需要用研来定性+定量分析各个行业的客户在不同场景下的痛点程度来制定优先级，哪些事情我们要做成功能，哪些事情必须要有服务部门进行人力支持。）

2. 竞品研究的预期产出是什么？

虽然产品经理未必会输出很结构化的竞品分析报告，但不意味着他们不做竞品调研。他们必然已经从互联网上摄取了大量的市场和竞品信息并进行对比分析。所以我们在做竞品分析时应该清楚，产品侧希望看到在**Google上搜索不到的内容**或者经过**从产品设计角度分析总结后的结论**。下方截图是PM对于ClickHouse竞品报告的反馈。


翟鹿渊 10月13日 09:34 （编辑过）

Comments on doing a better report:
1. Cannot find conclusions or figures with solid evidence and reference;
2. Many parts can go with tables or graphs to be more understandable;

Comments on content:
1. Who would be the audience for this report? What would be your point for them? If I were the target audience, I would want something that cannot be found through Google search and represents your thinking results.

预期产出：

1）从竞品的设计方面**收集一些亮点或创新点**，别人做得好的地方我们要学；
2）竞品研究相当于别人帮我们**试错，不好的产品思路和设计方向**我们也要避免；
3）**别人做得不够好的功能**我们要在**它的基础上**做得更极致；
4）识别出用户需求和友商产品之间存在的Gap并从中创造出**新的功能点**；
5）从竞品分析中看到**差异化的机会点**，或者找到蓝海区域；
6）获得若干能够**衡量**产品增长或成功的**基准点**（benchmark）；
7）商业和产品策略上做到知己知彼。

根据产品所处的不同阶段，我们的产出也会具有不同的意义：

0到1产品：设计师对于一个新的B端产品领域通常完全没接触过，通过学习直接/间接竞品可以快速了解问题空间、用户需求、产品逻辑以及市场状态；这些领域知识能够帮助设计师在思考设计时在大脑中有一个更加明确的产品形态。

产品发展初期：产品发展初期，迭代的思路往往没有打开，这个阶段我们可以从直接竞品的使用体验和用户反馈中获得一些已经明确的需求，并借鉴成熟的设计模式。

产品成熟期：产品中期往往会遇到一些瓶颈，这时候需要开始全面进行竞品分析，需要学习的不单单是直接竞品，更多应该去学习创新产品的思路，寻求突破的方法。

重新改版：有时候用户对产品的新鲜度会不断降低，产生审美疲劳，这时候需要重新焕发用户的新鲜感，需要进行一次产品的全新改版。这个阶段学习的不仅仅是竞品，更需要的是创新，多维度的学习各类前沿的产品思维。

3. 竞品分析流程

成功的竞品分析需要做好四件事情：

1️⃣市场分析 -> 2️⃣寻找合适的竞品 -> 3️⃣作出完整的分析 -> 4️⃣总结分析出来的结论。

3.1 市场分析

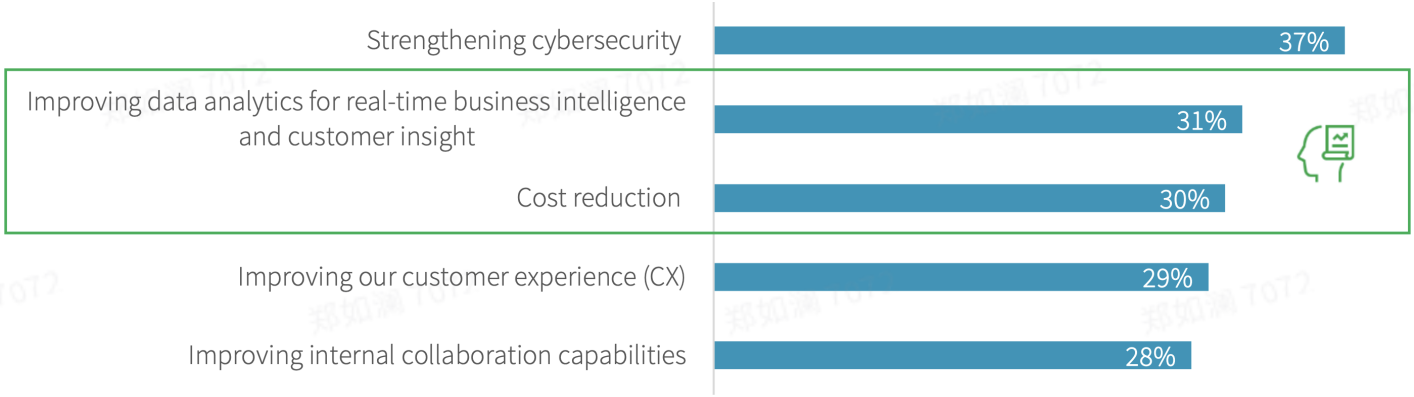
市场分析可以作为竞品研究的前期准备工作，主要目的是帮助我们自己获得足够的领域知识从而在拿到竞品的时候能对产品有一个更全面的认知。这部分的输出对于产品侧的意义不太大，但是从用户视角的观察也可以为产品侧提供一些知识补足。更重要的目的是为了**与产品侧对齐宏观层面的理解**，提前知道设计师对领域知识的理解有哪些漏洞。

3.1.1 商业视角

1）了解该领域的价值空间、竞争状况，重要的历史事件，各大厂商在该领域的布局情况；

2）了解主要客户的业务需求和这些客户企业想要达成的目标是什么，相对应有哪些相关的产品功能？

3）企业客户在选择该类产品时最关心的点是什么？（以往解决方案的问题的缺陷是什么？）例如下图是企业在对于数据分析产品的主要考量点：



4）该领域内还有哪些机会？有哪些需求还没有被挖掘出来或现有产品没有满足的？
5）我们的产品与其它同类竞品的区别是什么？在产品和技术方面有哪些已知的竞争优势？

3.1.2 用户视角

1）了解用户的种类和职业类别，他们分别在公司运作中的角色，他们在任务流程中各个环节的任务目标是什么？（参考对标企业在招聘相应职位时的job description）
2）用户群体对技术知识水平和掌握程度如何？（日常工作会接触到哪些技术？）
2）各个角色是如何一起合作来实现商业目标的？具体的分工与合作方式。。。
3）用户的主要痛点是什么？（以往解决方案对于用户来说有哪些是需要改进的？）
4）用户在使用这类产品时在上下游分别还使用哪些工具或产品？

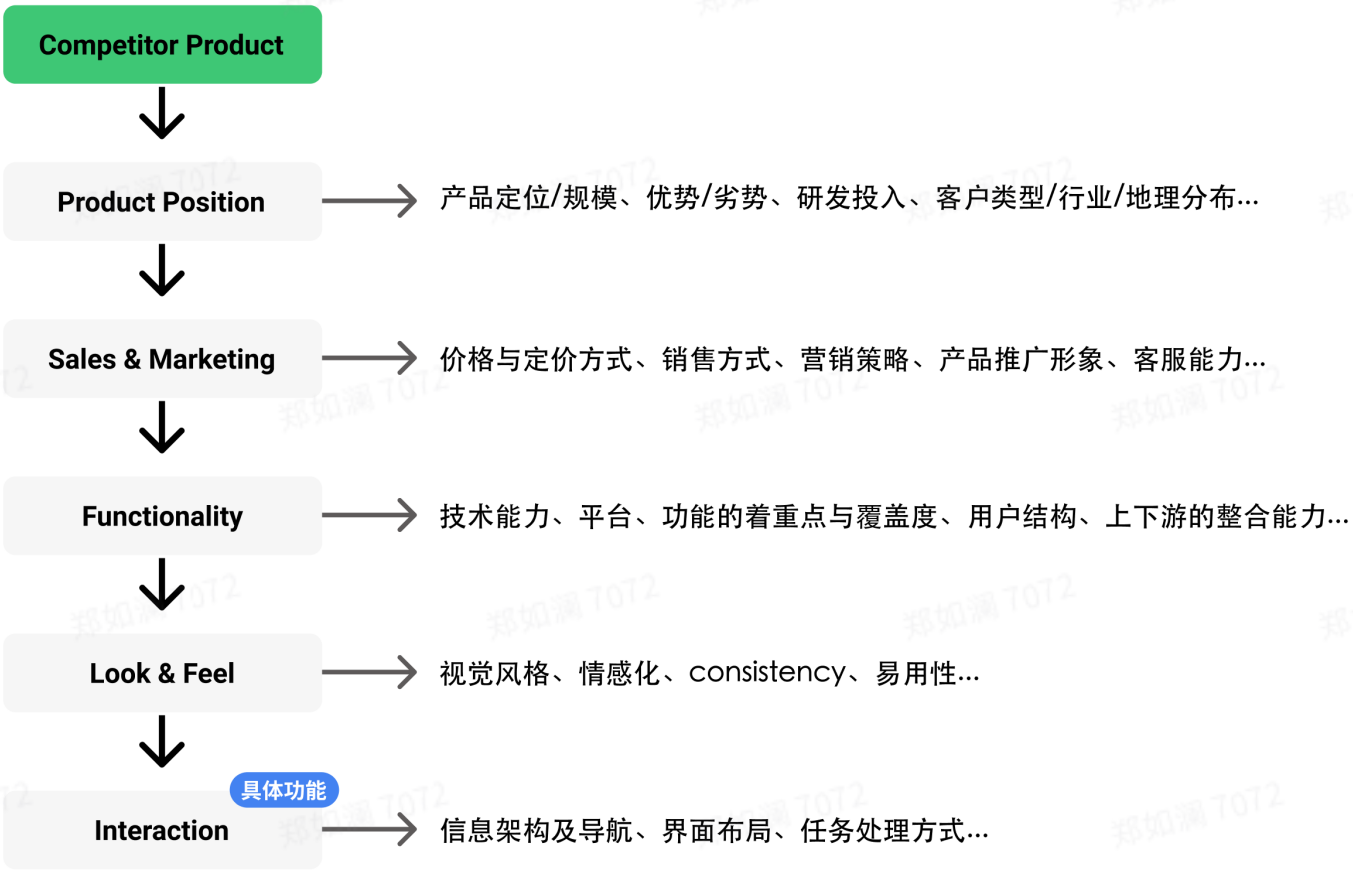
找到竞品的典型客户，再去客户公司的招聘网站上去看招聘的职位类别和描述;
直接Google搜索“What Does a xxx Do?”；知乎搜索“大数据工程师的日常工作内容是干嘛？”
https://www.rasmussen.edu/degrees/technology/blog/what-does-a-data-analyst-do/

3.2 找到竞品

直接在G2.com或TrustRadius上按照一个你了解的直接竞品来查找其他的竞品。
根据竞品所提供的功能来分为直接竞品、间接竞品、潜在竞品，我认为对于设计师做竞品研究还是主要放在直接竞品，或者那些有相同功能模块的间接竞品上。

3.3 解析竞品

设计师的竞品分析**重在对具体产品的精细化分析研究**而不是宽泛的对比分析，所以我们可以把竞品解析流程想像成炼油的步骤，每一个步骤有不同的目的并抽取出相应的产出。

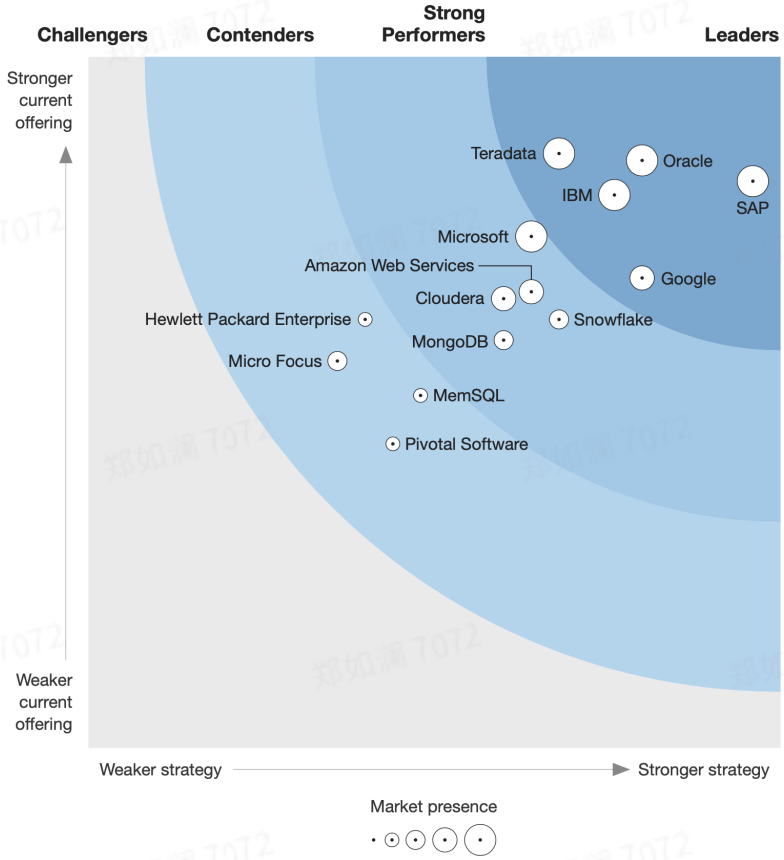


3.2.1 产品基本情况的横向对比分析

这一块的产出不太重要，目的还是为了帮助自己了解情况。可以参照下表从网上搜索信息填入；主要有价值的部分在于别人分析过的二手资料，例如优势/劣势。

产品名称	Snowflake	Google BigQuery	ClickHouse	Amazon RedShift
公司规模	M	XXXL	S	XXXL
收入	$264.7M			
运营年份	2013	2010	2016	2012
业务分布	北美、西欧、北欧、日本	All around of the world	东欧、亚洲	All around of the world
优势	计算存储彻底分离，外部接入的灵活性，	Google Cloud直接提供存储服务，完整的云设	开源，灵活性高	与AWS丰富的云服务衔接得更好并且有强大的

	各种云服务和ETL工具都可以适配；精准的按需计价；云服务免除了很多IT管理需求；非结构化数据的支持很强大；对安全和权限的颗粒度很高	施提供一站式消费；将例如VW和集群的概念全部封装起来用户仅需面对Query这件事情		内置安全能力。
劣势	依赖于第三方的存储服务，对于想要一站式服务的less technical的客户可能是一个障碍	Queries need to be optimized to avoid high costs when pulling data; 外部接入可能不够灵活	需要很多自己的运维和开发支持，不适合于小型客户	按照集群扩容，增加节点需要几分钟的时间，需要更多的人工介入运维，
价格	$$ charge by computing time	$$$ charge by storage		$$ charge by computing time
试用	注册直接免费试用	Sandbox有Google账户即可立即体验		两个月免费试用；长期订单折扣高；但需要提供信用卡信息才能注册试用
客户类型	[Large Corps] Disney, Discover, Salesforce, Nike	[Large Corps] UPS, Twitter, HomeDepot, DowJohns		[Large Corps] Yelp, Comcast, Lyft, Intuit, McDonald
用户评价	★★★★★	★★★★★	★★★	★★★★
销售特征 Voice & Tone	主打“云数仓”的概念，突出Daas；Managing Data, Not Infrastructure; "near-zero management"; Forrester reveals a **customer ROI of 612%**	采用第三方的分析作为依据：Forrester Research 将 Google Cloud 评为分析型数据管理领域的领导者；ESG分析了它的Economic Advantage最好(26%-34% TCO savings)；提供了非常方便的售前界面体验		采用有说服力的客户案例：比如“借助Redshift，Lyft 等公司已**从初创公司发展为市值高达**几十亿美元的企业。”因为AWS已经有很多大客户？



3.3.2 营销与销售等方面的商业策略分析

竞品的商业模式是什么样子的？收费方式、价格策略、客户的长期投入等等。
免费试用的方式是怎样的？注册是否需要提供信用卡信息？有没有一个可以直接体验的demo？
营销策略如何，针对什么行业、地区、层次的公司客户？
有没有其他产品、基础设施或C端业务作为支撑？
Marketing采用什么形式？视频、网站、线下活动？
Marketing素材的视觉风格怎样？传达怎样的品牌信息？

3.3.3 技术与功能分析

技术和功能是客户在选择产品时最重要的决定因素，也是在网络上能找到现成分析最丰富和最完整的。这部分的分析可以按照以下几种形式：

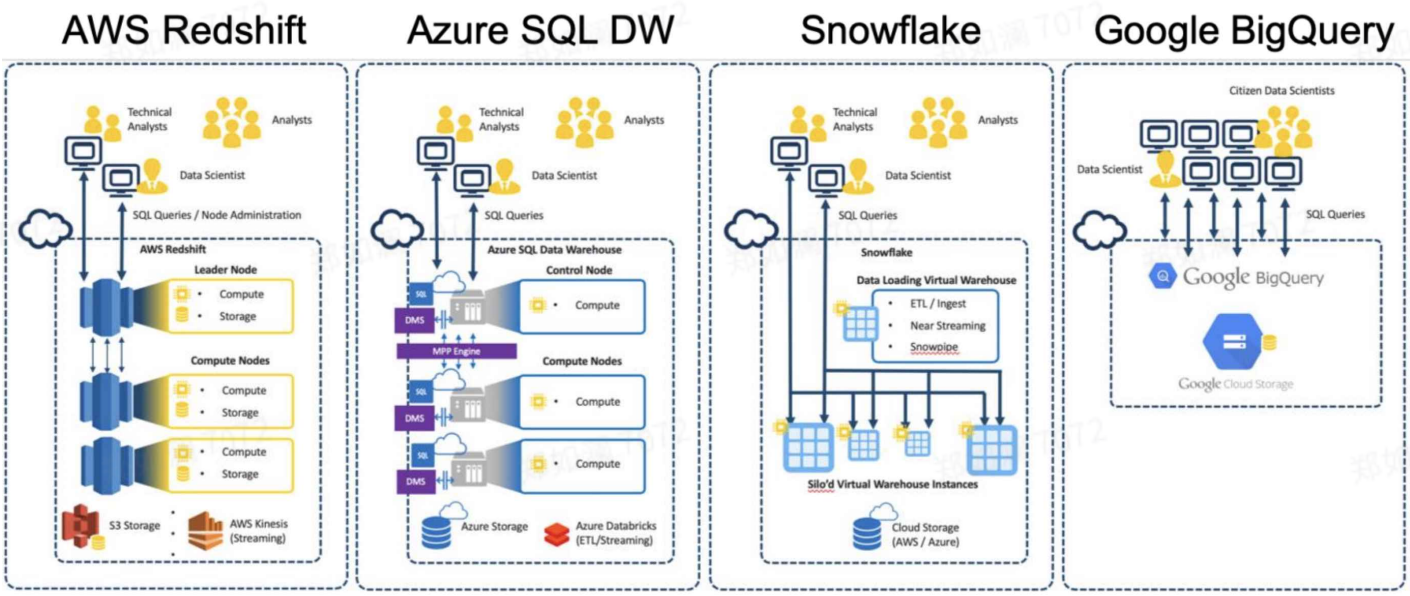
1）直接参考网上的对比文章，例如搜索“Google BigQuery VS AWS Snowflake”就可以得到现成的详细分析，往往这类对比文章都是从产品、技术、功能的维度展开的。基本上可以列出该品类产品最重要的一些功能点的差异性

Snowflake对比RedShift具有更好的Semi-Structured data支持，比如有更好的JSON based功能与query；Snowflake可以瞬间扩容，而RedShift需要花几分钟的时间增加计算节点；Snowflake有更多的自动化维护而不需要人为介入；Snowflake的计算与存储分离，这一点与Redshift和Google Query

都不一样，计费方式上Snowflake采用time-based pricing model for compute resources。Snowflake允许管理员分别对计算和存储进行扩缩。

RedShift提供更好的与AWS其他服务整合的能力，它的计费方式是计算与存储打包的。Redshift相较于另外二者需要更多的运维管理，但是在数据导入时不需要考虑staging的存储控件。

Google BigQuery强调Serverless它完全自动化控制scaling。Google BigQuery不像AWS Redshift有集群、Snowflake有VW，而是把这些东西都封装起来了直接按照Query结果的体量来计费；看上去更适合小白用户使用。



Source: Enterprise Strategy Group

2）使用TrustRadius和G2等企业级软件评价网站查看用户对产品的使用感受和评论，这些网站往往都会把评论分为“like”和“dislike”两个部分，很清晰看到在用户眼里这些竞品有哪些地方是有价值的，哪些地方是值得改进的。把关键信息记录下来之后可以采用类似于同理心地图的方式进行归纳总结。

Snowflake's user quotes:

Google BigQuery's user quotes:

RedShift's user quotes:

Advantages:

- Great UI, the initial costs were very low, and it was super easy for us to spin up databases. Also, setting up multiple instances were easily managed with just a click of a button.

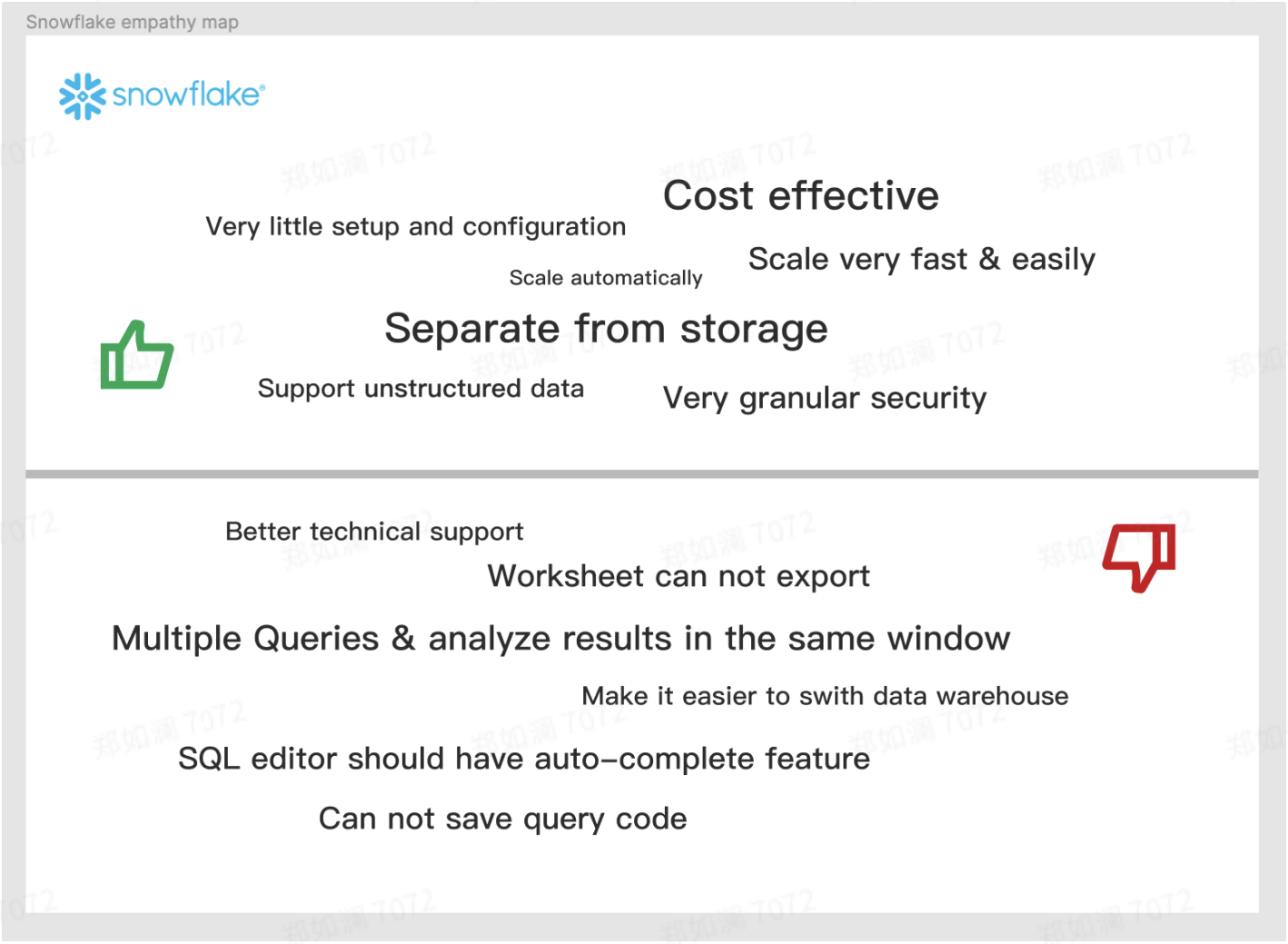
- Their auto-scaling feature came to our rescue when it came to cost management.
- Tunable table design. (待调查)
- Integrates quite well with other products within the AWS ecosystem.
- Performance tuning and database optimization can be done using the system tables and advisors. These solutions are similar to the solution available for Oracle SQL Server. It makes it easy to do the optimization for queries and databases.
- Large dataset and running fast

Can be improved:

- Implementation in **non AWS server**
- It could not separate users from using the **same infrastructure**.
- No real separation between computing and storage
- A leader node can become a bottleneck for too many **concurrent aggregate queries**.
- All users share the same infrastructure resulting in frequent 100% utilization error messages.
- The number of connections is too small, I think at around 50 are allowed in parallel. With some ETL and apps connecting all the time, this brings an undesired possibility to some users or tools being unable to connect.
- No **statistical inbuilt functions** are available within the tool.
- The concurrency and scale up based on it could be improved. It would be good if it scale-up and scale-down the memory/CPU capacity automatically based on workload.
- Time consuming process for schema design and modification
- It does not support **row-level controls**.
- There is no support for data de-duplication; meaning this has to be either accounted for upstream, or you'll have to build your own services to de-dupe your data.
- Permissions can be a pain... In some instances it's easier to drop/rebuild a table than try to navigate incremental updates/insertions, but retaining user-permissions is a pain-point.

这里面提到最多的问题是：快速按需扩容、控制成本、与其他功能整合、细致分析Query、计算存储分离、云上服务的自动化、智能化压缩、灵活易用的数据安全和权限管理

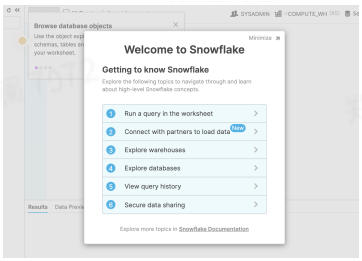
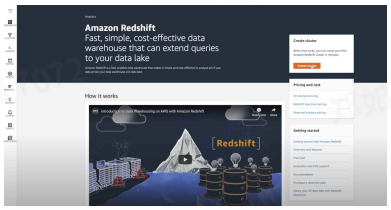
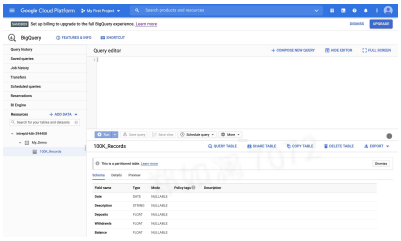
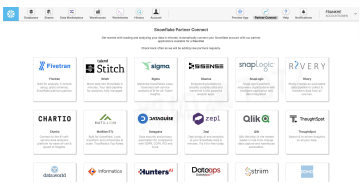
类同理心地图



3.3.4 体验流程与感受

注册竞品账户，登陆试用是最直接的体验方式；或者我们可以用YouTube来观察用户完成某项特定的任务，例如Google BigQuery和Snowflake的数据导入流程在YouTube上都能轻松找到。体验流程可以重点分析售前的网站内容、产品开箱体验、说明文档以及网络上可获得的学习资源、客服体验（这一块可能只能通过网络搜集信息）、界面视觉感官、外部系统整合能力、setup、花销管理、布局导航等方面。

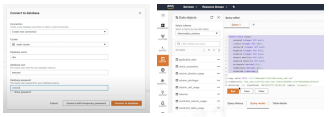
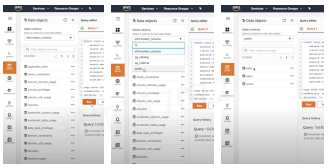
	Snowflake	BigQuery	AWS Redshift
售前的上手试用体验	邮箱注册免费使用30天	有Sandbox无需注册即可直接开始体验产品 ，但这个feature好像不是很容易发现	注册可以试用2个月，但是要提供信用卡信息用于身份验证。

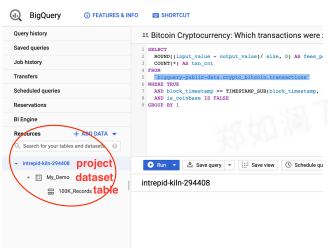
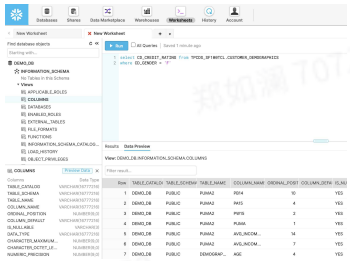
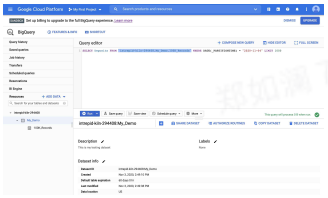
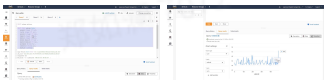
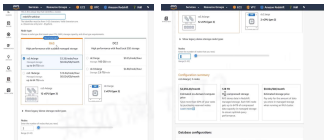
产品的开箱引导体验			 Landing page有How it works视频，然后引导用户创建集群
说明文档与相关的教学视频	有官方视频解说，也有大量的网络教学视频		
管理员工具或使用前的Setup工作			
客户服务的体验			
界面的视觉观感体验			
Integration 从外部系统转移到新环境	数据存储支持了AWS S3、Google Cloud、Azure Blob，用户可以直接建立连接。 		
Billing 如何监控花销			
布局与导航	Snowflake顶部导航突出数据库的位置，其次是Warehouse	BigQuery只有一个界面，就是查询界面，数据库的展示和管理、	Redshift采用侧栏导航，首页为dashboard其次是集群cluster用

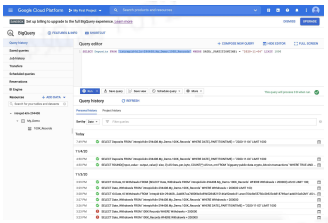
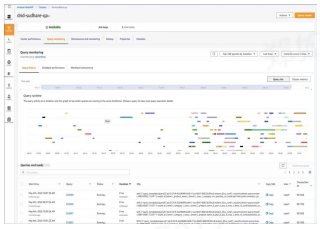
		数据导入都是放在这个界面之中的，通过抽屉、弹窗来进行分支功能的操作。	于监测使用状况，	

3.3.5 具体功能的交互设计分析

列出产品的主要功能，依次研究各个竞品是如何设计该功能的试用流程的。

	Snowflake	BigQuery	AWS Redshift	ByteHouse
数据导入	Snowflake提供了四种数据导入方式： 1）界面导入wizard（低频）；2）用SnowSQL在command line里面执行数据导入；3）SnowPipe数据导入工具；4）第三方ETL工具的数据导入。 	BigQuery通过“建表”的操作进行数据导入，并提供Auto-detect的选项来构建Schema，用户也可以手动输入schema。这一功能是放在查询页面中的 	Redshift的数据导入操作也放在了Query Editor里面，用户需要先做Connect to database的操作把  用COPY command把数据从S3 load到刚才创建的table中	ByteHouse明确把Data Import单独作为一个产品功能放置在顶部导航中，并提供丰富的数据接入方式，支持scheduled batch loading和streaming，提供一定的监控能力。
库表管理	Snowflake把数据库作为首页，用分开的两个listview展示数据库和表，用面包屑表示层级关系。数据库层级有很多功能：创建编辑表、Views、Schema、Stages... 	BigQuery**没有具体的数据库概念**，数据表是按照Project > Dataset > Table的层级来组织的。Datasets are **top-level containers** that are used to organize and control access to your tables and views, you need to create at least one dataset before loading data into BigQuery.	**Redshift在导航中没有明确的库表管理**，用户会在Query界面中直接看到数据表，表是按照schema来进行组织的。在数据导入之前有一个“**Connect to Database**”的操作，可能AWS有专门的database区域。 	

				
查询	Snowflake用Worksheet的概念来放置查询功能 	BigQuery就是以查询页面作为单一页面， 	下面的tab分别是Query History、Query Results、Table Details，在Query Results里面有Execution、Data、Visualize三个tab分别以不用的方式来描述query结果。 	
权限管理	Snowflake提供了高颗粒度的权限管理并且直接放在了库表页面 	BigQuery的权限控制是放在Dataset这个层面的，“Share Dataset”打开操作面板，分别控制permission和authorized views。 		
虚拟数仓	Snowflake把Warehouse放在了顶层导航作为入口 	**BigQuery不需要数据仓库**	RedShift需要设置cluster集群，选择节点类型 	
查询历史	Snowflake把查询历史放在了顶层导航作为入口，并提供了视觉化的Profile描述 	BigQuery把Saved Query、Query History、Job History放在一个目录里，并分有personal和project的历史记录两个tabs	RedShift在集群Cluster里面有Query Monitoring，下面有一个查询历史的可视化分析能力。这篇文章教用户使用这个功能优化query：https://aws.ama	

		zon.com/cn/blogs/big-data/monitor-and-optimize-queries-on-the-new-amazon-redshift-console/ 	

3.4 总结分析结果

以上的分析和研究主要是用探索的形式搜集信息，最后我们需要对搜集到的竞品信息进行有针对性的总结，从而得到对我们产品设计有价值的洞察。在总结阶段，我们可以按照功能、设计、策略这三个层面来进行。

3.4.1 功能层面的总结（What？）

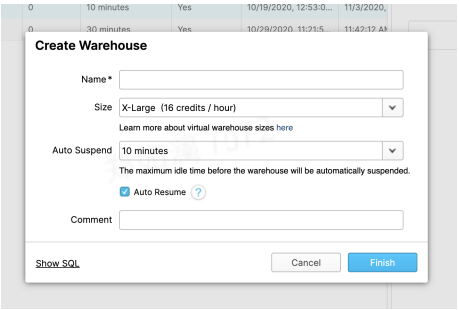
因为我们在这里是**总结**而不是**罗列**，所以要关注的不是“竞品分别有哪些功能”，而是“竞品的哪些功能特别好、或不好”。无论是亲自体验产品、或看产品使用视频、还是从软件测评网站上看真实用户的留言评价，我们需要从中总结出来竞品的哪些功能在产品问题空间中特别关键，哪些是用户特别在乎的功能，哪些功能其实没有那么重要等等。区别于下面的设计层面，功能层面的总结在回答“我们**要不要**某些功能”的问题。这里的“要不要”往往是从用户体验的角度来判断的。

例如对于数据仓库产品，很多用户提到了特别喜欢Snowflake的计算与存储分离的架构方式，以及高颗粒度的权限管理能力；也有用户提到了RedShift细致的Query分析功能；说明这些功能在数据仓库的产品使用情景中是非常关键的功能。一些细节的功能设计例如Snowflake的stage管理并没有发挥很大的作用并且造成了一定的confusion，这就是我们可以优化的设计点。

下面例举了一些数仓产品功能竞品洞察：

1）Sizing & Costing Reference

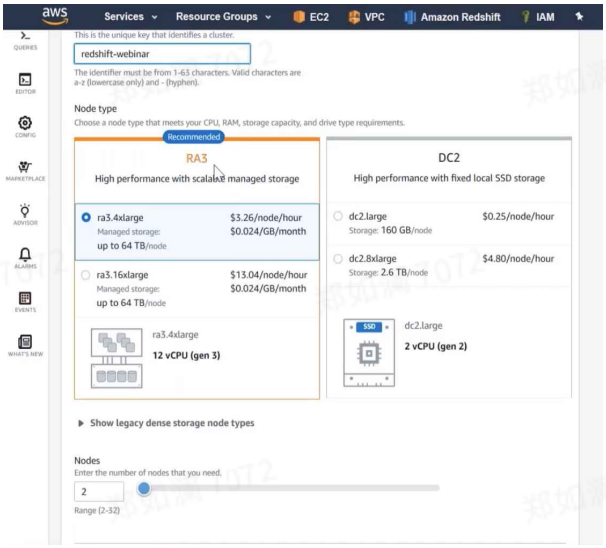
Snowflake让用户通过configure Warehouse的方式定义计算资源从而决定最终花费，它采用了T-shirt size的分类方式，但是不论在UI中还是在帮助文档里都只是告诉用户每个size代表了什么意思以及价格是怎么计算，而没有告诉用户对于多大的任务量应该采用多大的warehouse。作为用户仅仅知道XL的价格以及相对应的服务器数量可能是不够的，如果能够给用户一个例子来参考可以帮助用户更快做出决定。虽然Snowflake说用多少花多少，但是从用户的心理上还是希望更有底一些。

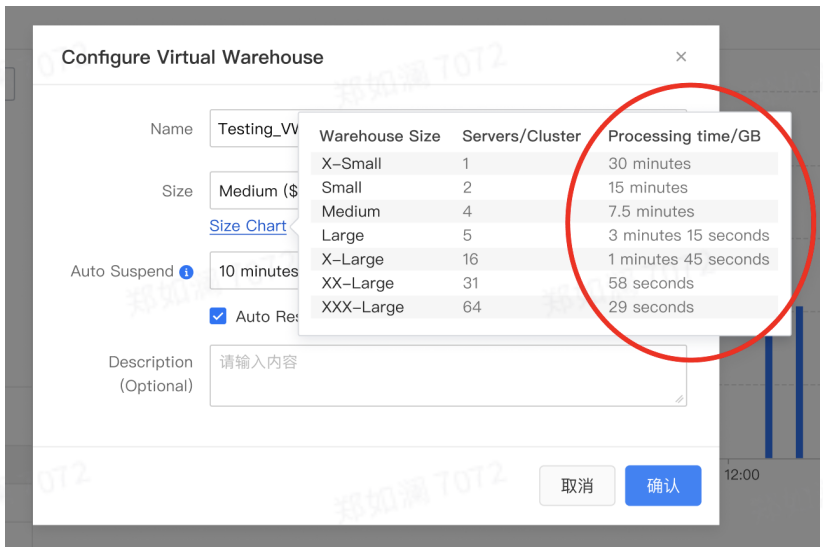


Warehouse Size	Servers / Cluster	Credits / Hour	Credits / Second	Notes
X-Small	1	1	0.0003	Default size for warehouses created using CREATE WAREHOUSE.
Small	2	2	0.0006	
Medium	4	4	0.0011	
Large	8	8	0.0022	
X-Large	16	16	0.0044	Default for warehouses created in the web interface.
2X-Large	32	32	0.0089	
3X-Large	64	64	0.0178	
4X-Large	128	128	0.0356	

AWS Redshift用了一些插图并且对计算/存贮资源进行了一定程度的描述。毕竟这是直接与钱挂钩的，所以在UI上做得细致一些有助于提高用户的获得感。

建议：提供一些可以帮助用户了解他/她应该选多大数仓的参考。





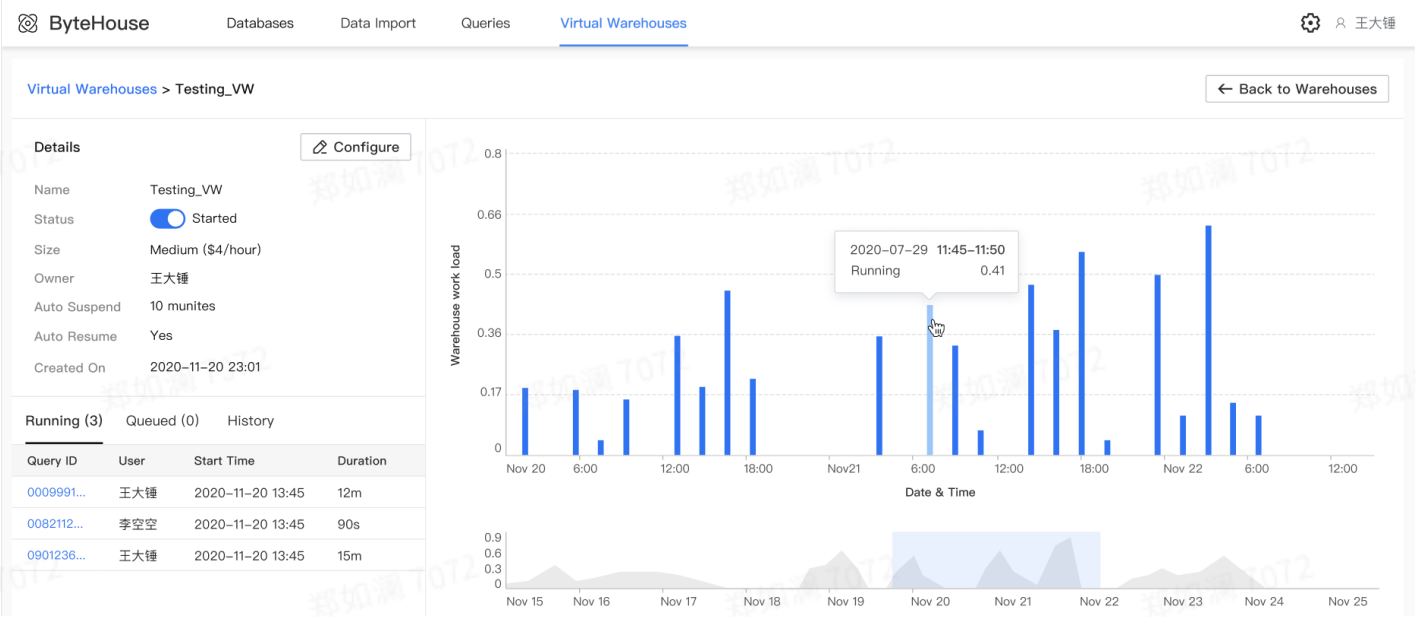
2) Data Warehouse Details & Monitoring

Snowflake的数仓详情页面非常简单，除了monitoring图表，基本没有其他任何信息。其实数据仓库作为运行数据的hub可以提供给用户更多与具体数据运行任务之间的关联。例如在AWS Redshift的Cluster里面就有关于Query的runtime在集群内部的一个分步情况的可视化分析。

Query runtime allocation in a Redshift Cluster:

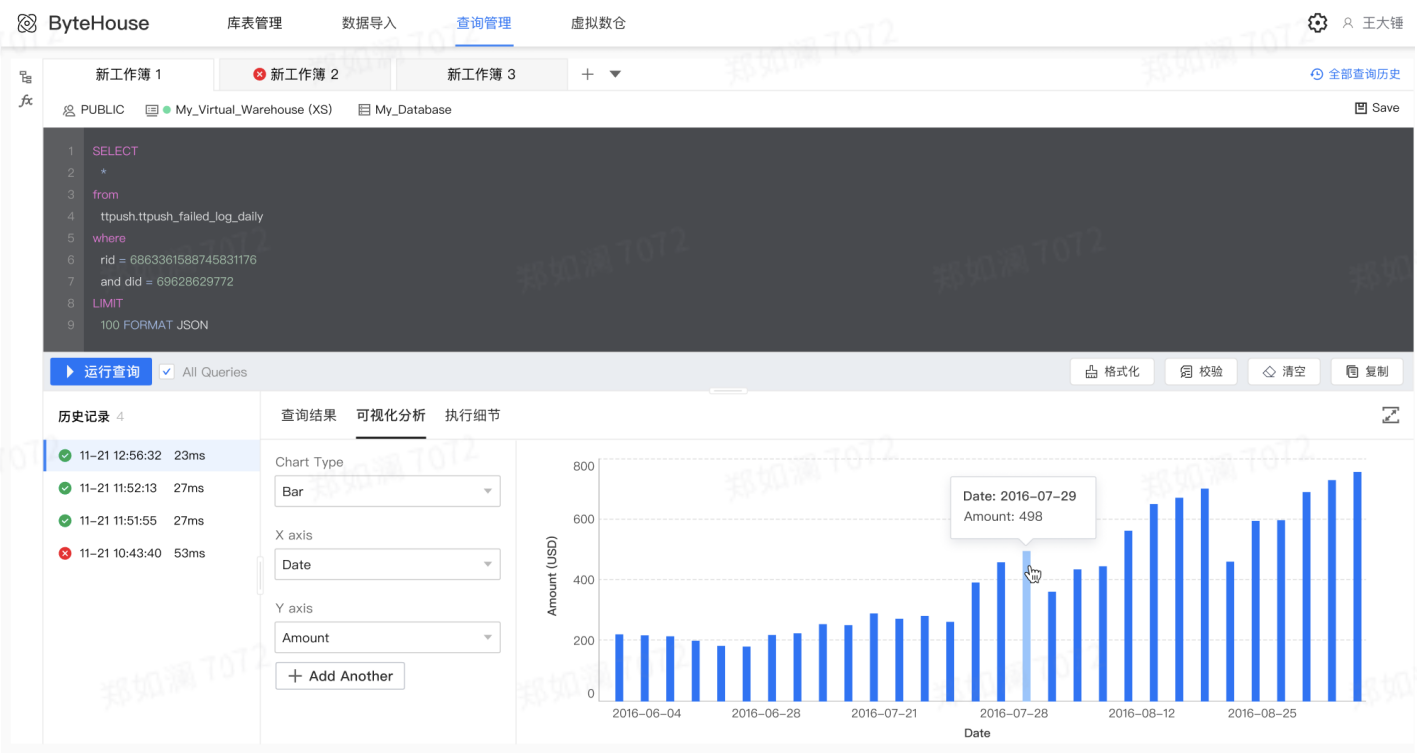
这样的功能可以给用户很直观的认知看到当前Warehouse被占用的情况，一目了然知道这个warehouse的大小是否合理。对于并发的情况用户可以看到拥堵信息，哪些任务在占用资源，从而进一步调查原因从而做出改进。

建议：在VW详情页给出更多信息，增加VW与Query任务之间的关联，帮助用户分析资源占用情况。



3) Quick data visualization

不同竞品由于其对应的产品生态不同，对于数据可视化的能力有所差异。Snowflake并没有在查询结果处设置任何数据可视化的功能，期望用户导出数据在其它产品中进行可视化；Google BigQuery可以把查询结果用内置的BI工具进行可视化，Amazon Redshift则在查询结果处提供了一个简单的可视化模块。



建议：鉴于ByteHouse相对独立的产品定位，我们应该在查询结果处设置基本的可视化功能。后续可以在用户研究中重点调查：如果在查询结果这里展示可视化图表可以给用户带来什么价值？从整个数据分析的用户体验流程中，是否能够帮助到用户？例如有些insights可能不需要去专业BI工具就能直接看到？

3.4.2 设计层面的总结（How？）

设计层面的总结在回答how it works的问题，我们希望总结不同竞品在对于几个关键功能的设计方式上的特点。从竞品用户反馈或者设计师的自行体验中分析不同设计方式的优缺点，再根据我们设计的产品定位和用户特点提出建议。最后的结论可以是跟随某一个设计方式，或者在已有的设计方式上提出更好的改进方案。区别于上面的功能层面，设计层面**在“我们要某些功能”的前提下**回答“这些功能要**如何设计能更好地为用户使用**”的问题。

下面例举了一些数仓产品设计层面的竞品洞察：

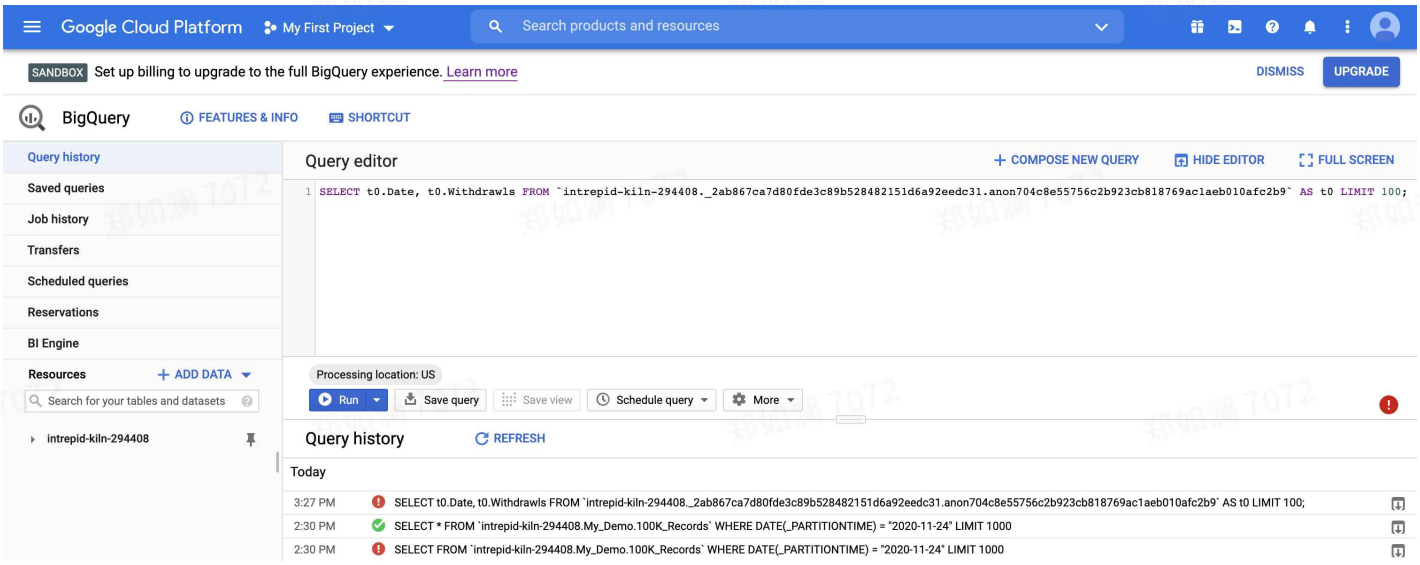
1）Permission: role-based access managment

在Snowflake的竞品分析中发现很多用户对它的权限管理功能提出了较高的评价，尤其提到了它的权限管理的颗粒度做得很人性化。这是一个典型的设计层面的竞品洞察，权限管理是数仓产品肯定会有的功能，但如何设计它很大程度上决定了用户的使用效率和整体体验。

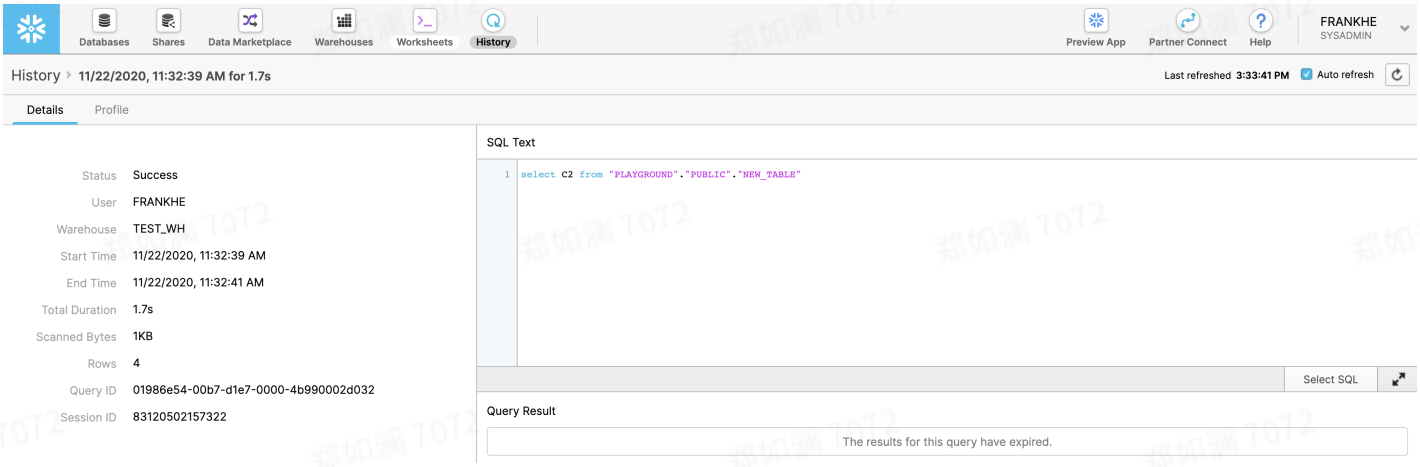
建议：在ByteHouse的设计中我们跟随这一交互设计，并在这个基础上增加一个建表完成时默认弹出的效果来测试用户的反馈。

2）查询页面布局与工作区-结果-历史记录之间的界面逻辑关系

Google BigQuery采用了单页面设计，几乎所有用户流程都在查询器的界面内发生；工作区显示和编辑查询语句，数据区显示表详情、字段、数值以及查询结果、查询历史等等。由于BigQuery是Google Cloud下面的子产品，可能是因为它希望让用户感觉这个产品尽量轻量化所以采用这种设计。



Snowflake在界面上有把一个query执行结果作为一个**实体**来展现，占据了单独的页面。这样做的缺点是产生了一些冗余(查询页面的下方也是可以展现查询结果的历史记录和详细信息而无需单独页面)，但优点是逻辑清晰：分析用户专注于当前查询任务，然后用户每运行一次SQL就会产生一个record，这些records可以给Admin用户便于分析Query performance和识别一些技术性问题。



建议：把查询结果作为一个单独实体，有单独的列表页和详情页；但是用户可以快速基于同一个SQL语句重新运行并在worksheet中得到结果。

3）存储SQL text的功能

Snowflake对于SQL code采用了默认的auto-save，这使得用户感觉worksheet就是一个工作视窗，任何一个改动都立即被保留在了编辑器里，但是如果想要存储一个写好的SQL代码但不执行，又希望在这个代码上做一些修改则需要复制粘贴到另一个worksheet里进行编辑。另一个场景是希望复用一段代码，用户希望直接调取使用，这种情况也只能依靠用户手动操作。

建议：将Worksheet作为一个工作视窗，但是具有persisting保存（auto-save）SQL内容的能力(也就是用户不关闭worksheet的话里面的SQL会一直在那儿)；另外提供一个保存代码块的功能，让用户能够快速复用之前编辑好的代码。

3.4.3 策略层面的总结（Why？）

我们要基于以上的What（竞品有哪些重要的功能）和How（这些功能是怎样设计的）来总结为什么他们有这些功能并且按照这种方式来设计这些功能。策略的总结可以为我们在功能和设计的抉择上提供更明确的理据。

产品	功能点/设计点	策略解析	映射到我们产品的意义
Snowflake	暴露Stage的概念	Snowflake是一个对技术包裹得比较浅的SaaS产品，很多技术概念是直接暴露的给予用户操作的灵活性。大量的执行未必在UI上，所以基于这个考虑他们把Stage的概念展示在UI上有助于用户在UI和IDE环境下切换时的连贯性。	我们希望提供更加简明清晰的界面逻辑，更多的操作是在界面而不是用代码完成。
Google BigQuery	单一页面设计，查看表内容和查询结果的展示共用同一个页面空间	BigQuery属于Google Cloud下面的子产品，所以它需要被进行轻量化设计；但在可用性方面其实不如Snowflake这样的完整产品。	。。。
Google BigQuery	仅能打开一个代码编辑框	它更像一个代码执行工具，**用户重视的是运行结果**；由于是单页面设计，用户从运行结果可以直接打开SQL编辑窗口，所以从查询记录作为不同worksheet的导航而不是同时打开多个tab，其实本质上是一样的。	
Redshift	。。。	。。。	

基于上面的功能/设计点的策略分析，我们已经有足够的产品认知可以按照SWOT的四个象限概述我们产品的强项、弱势、机会和威胁。目的是帮助我们为产品功能制定优先级。

Strength：我们在哪些地方可以做得比竞品更好？

Weakness：有哪些地方是我们肯定不如竞品的？（这里指的是内因）

Opportunities：我们最主要的竞品有什么致命弱点区域可以让我们发力？市场上是否有没有被填补的Gap？最近的半年内市场上有什么最新动向值得我们跟随？

Threats：当产品上线推广时会受到哪些挑战？（这是外因）