stratebi
open business intelligence

Big Data OLAP Analytics Benchmark

Apache Kylin™ vs VERTICA

PostgreSQL

1. INTRODUCTION

The user of Business Intelligence (BI) systems usually gets a very fast and interactive response when using dashboards, reports and detailed analytical queries. BI applications that meet this interactive processing requirement are known as OLAP (On-Line Analytical Processing) applications.

However, when we work with data sources with Big Data features (Volume, Variety and Velocity), our metrics tables (e.g. sales volume, units...) and those tables that describe the context (e.g. date, customer, product) could store billions of rows, making the processing requirements very high, even for the most advanced Big Data technologies.

In order to support OLAP applications with Big Data, multiple technologies that promise excellent results have emerged in recent years. Some of the best known are Apache Kylin, Vertica, Druid, Google Big Query or Amazon Red Shift.

Although these architectures are very different, each of these promise the capability of performing analytical queries with response times (query latency) ranging from milliseconds to a few seconds, using the standard SQL query language, very well known in most IT teams.

However, our experience deploying these technologies has shown us that not all deliver promised results. Apache Kylin and Vertica have achieved the best results so far in our implementations of Big Data OLAP systems. In order to test and compare the performance of Vertica against Kylin and, also, of these two against other non-Big Data technologies, such as PostgreSQL, we have carried the Big Data OLAP benchmark that we present in this whitepaper.

2. CURRENT BIG DATA OLAP TECHNOLOGIES

In this section we describe the Big Data OLAP technologies that are part of the benchmark: Apache Kylin and Vertica. Besides comparing these technologies against each other, we have also compared them with the relational database PostgreSQL. This open source technology, despite not being a Big Data database, usually offers very good results for traditional OLAP systems. Therefore, we considered worthwhile to include PostgreSQL in order to measure the differences of it against Kylin and Vertica in a Big Data OLAP scenario.

Apache Kylin

Apache Kylin is an open source tool that is defined on its website as a distributed engine for analytical processing scenarios that provides an SQL interface and supports multidimensional analysis applications (OLAP) on a Hadoop/Spark cluster and over Big Data sources. Originally developed on eBay by Luke Han and other researchers, at the end of 2014 it was released as an open source Apache project and its development has continued unceasingly until today, as demonstrated by the activity of the Github website of the project.

The main features of Kylin are listed below:

- **Interactive queries with sub-second latencies over analytical models with tables of more than 10 billion rows.**
- **Support for the standard SQL query language.**
- **Advanced data compression.**
- **Support for J/OBDC connectors to enable integration with the most popular BI tools:** Power BI, Tableau, Pentaho, Zeppelin, Superset , Microstrategy, …
- **Full scalability** by adding hosts to the Hadoop cluster and scalable storage in HDFS and Amazon S3. Also supports **high availability**.
- **Comprehensive web interface and API restful**, to manage data models, cube models, loads and updates, configuration and even to query data and retrieve results in JSON format.
- **Automatic cube optimizer** based on the gathering of statistics and the application of Machine Learning techniques.
- **Security** through LDAP and Apache Ranger integration, for security at the level of functionalities, cube schemas and data.

- **Enterprise version with added features**: In addition to the open source version we have benchmarked in this whitepaper, there is a commercial version called Kyligence. This version adds some improved features such as a higher performance storage system, intelligent wizards for cube design, improved security and technical support.

Figure 1. shows the architecture of the Apache Kylin system. The Data Warehouse (DW) that we use as the data source for the Kylin system is a set of tables that is maintained in Apache Hive or other database systems (e.g. SQL Server, MySQL or even Vertica). Additionally, Kylin also allows real-time data gathering from an Apache Kafka queue (aka topic or streaming table).

Once we have selected the data source, we have to define the data model from the Apache Kylin web interface. This model describes aspects such as the joins between the different tables, the columns we would like to use as a metrics or dimensions and the date columns used for incremental loading and data updating.

The last step is the OLAP cube definition over the previously created data model. We have to define certain aspects related to the level of pre-aggregation and pre-combination of the data. This kind of OLAP architectures based on the pre-aggregation and pre-combination of the data are called Multidimensional OLAP (M-OLAP) and, thanks to this feature, Kylin achieves excellent query times.

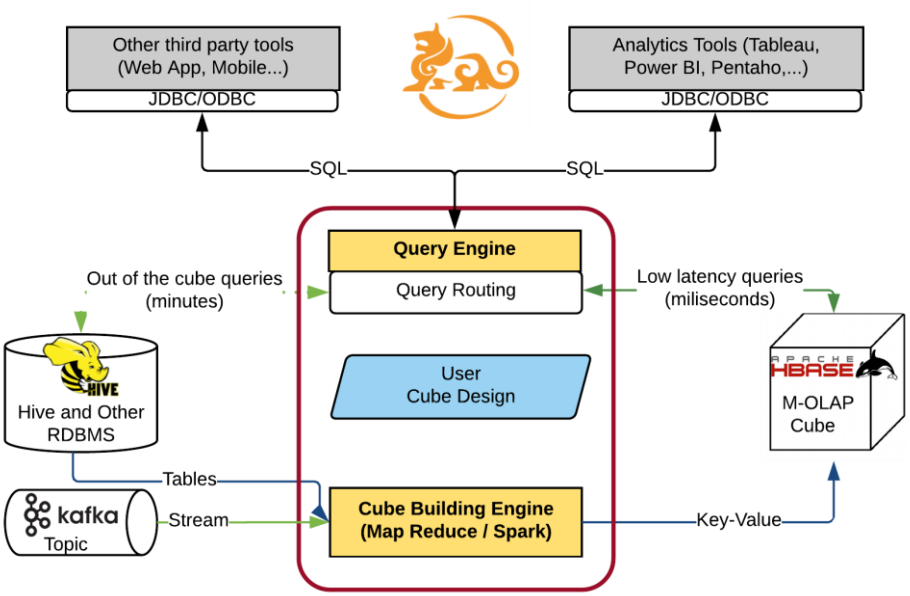


Figure 1. Apache Kylin Architecture.

Once a first data load has been done (first cube building), we can start to execute SQL queries or starting new cube building processes to add or refresh data in the OLAP cube. Queries over the cube defined in Kylin are usually resolved in less than 1 second.

More than 200 companies already use Kylin worldwide, among them StrateBI as shown in the list of companies that use and support the Kylin project.

Vertica

Vertica is an analytical database that implements a MPP (Massive Parallel Processing) architecture for distributed data processing and columnar storage. As Kylin, Vertica is aimed to support OLAP applications over Big Data sources. Its beginnings date back to 2005, when researcher Michael Sonebraker published the paper "C-Store: A Column-oriented DBMS" which is the basis of the Vertica system. It was later acquired by Hewlett Packard (HP) and later by Microfocus in 2017.

Unlike Kylin, Vertica is a technology that can be deployed independently of Hadoop technologies. The following are the main Vertica features:

- **Interactive queries over tables of billions of rows.**
- **Support for standard SQL query language.**
- **Advanced data compression.**
- **Support for J/OBDC connectors to enable integration with the most popular BI tools:** Power BI, Tableau, Pentaho, Zeppelin, Superset, Microstrategy, …
- **Full scalability** by adding machines to the Vertica cluster and **high availability**.
- **Comprehensive web interface** to manage cluster status and load (general and detailed views by node), databases, schemas, tables, wizard for automatic schema optimization or query plan analysis.
- **Semi-automatic schema optimizer** by means of a simple wizard.
- **Security** features include user authentication and authorization, schema and table level security, LDAP integration and client-server communication encryption.
- **Native Machine Learning Functions**: Includes native data preparation functions, regression algorithms, classification and clustering.
- **User Defined Functions (UDF):** Very useful feature that allow users to create ad-hoc functions with languages such as SQL, Python, C++, Java or R and use these new functions within SQL queries.
- **Enterprise** version with added features: Free community version (used in this benchmark) is limited to a cluster of 3 machines and 1 TB of storage. However, these free features are suitable for many companies. The enterprise version has no limitations.

Figure 2. shows the Vertica architecture. To create an OLAP cube in Vertica it is necessary to define a table schema and load data in it from one of the supported data sources: files (e.g. CSV), relational databases, using ETL tools such as Pentaho or Talend, or even from Kafka queues (topics), to load data in real time.

After defining the schema and loading the data into it, we can execute queries in SQL language or connect from one of the best-known BI tools, such as Tableau, Power BI or Pentaho, to create new dashboards and reports with interactive response.

In contrast to Kylin, Vertica does not pre-aggregate and pre-combine data. Due to this fact, the resolution of analytical queries in Vertica requires more computation in query time than in Kylin. However, we can run the optimization wizard to apply projections and flattened tables optimization techniques, which allow to achieve a certain level of pre-aggregation and pre-combination in order to improve query latency over one Vertica schema. This kind of hybrid approaches are called Hybrid OLAP (H-OLAP).

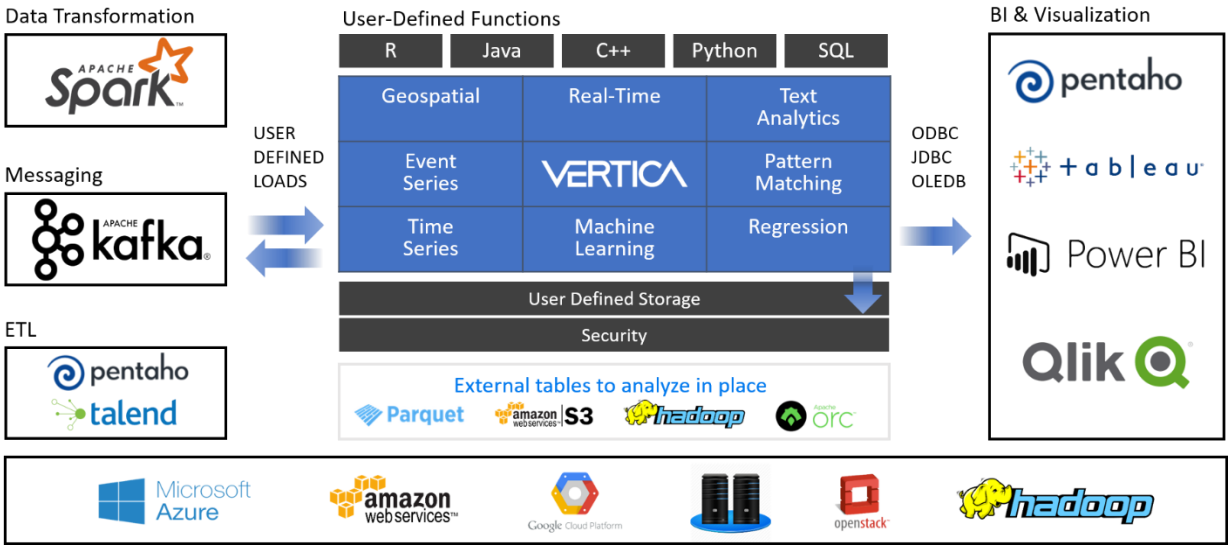


Figure 2. Vertica Architecture.

In addition, the Vertica cluster can be integrated with a Hadoop cluster with the aim to enable querying data stored in Hive or HDFS and combine it with Vertica tables to enrich the analysis or take advantage of tools such as Spark for the processing of less structured Big Data.

Vertica has been successfully implemented in a large number of companies. In StrateBI we trust on this technology, achieving its successful deployment for several scenarios and different sectors. That is why we are now Vertica partners.

3. BENCHMARK

Once the benchmarked technologies have been presented, in the following sections we will describe the details of the benchmark implementation and the setup of the several tests carried out.

Benchmark implementation

There are many database benchmarks, but the TPC-H benchmark is one of the best known and accepted by industry leaders such as Oracle or Microsoft. In our case we have used the SSB benchmark, a variant of the TPC-H benchmark that uses a "star" scheme, a more optimized and common Data Warehouse (DW) model than the “snowflake” schema used by the original TPC-H benchmark.

Figure 3 shows the star scheme proposed by SSB that describes a typical scenario of the retail sector. In this DW model, sales of products (quantity, price, discount, taxes...) are studied at a detailed level of sales line by date (year, month and day), product, customer and supplier dimensions. This benchmark selects and adapts 13 of the original SQL queries proposed by original TPC-H. These queries use SQL group by, where and joins clauses, aggregation functions and its design is scientifically based. All these queries are provided in Annex 1 of this document.

In addition, the SSB benchmark requires to adapt the original data generator provided by TPC-H. With this aim we have used one existing implementation for Kylin and Hive from Kyligence team, but applying some modifications in its code to use it also with Vertica and PostgreSQL.

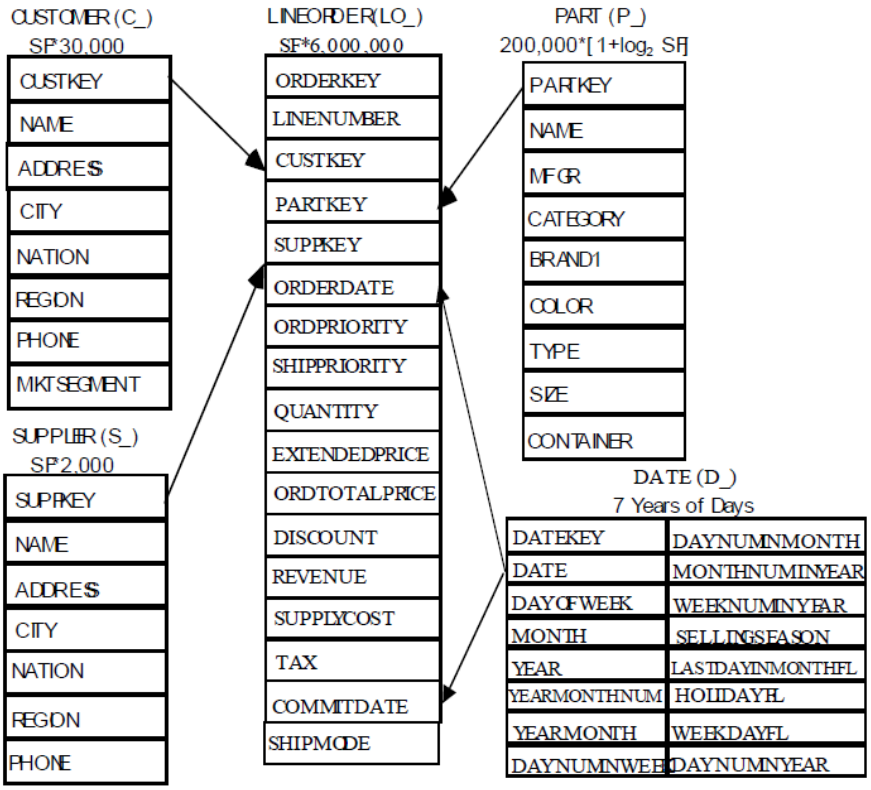


Figure 3. Star scheme of a retail scenario proposed by the SSB benchmark

Design of the tests performed

The setup of the tests performed has three key aspects, listed below in the order in which they were determined:

- **Number** (Volume) of rows in facts (LINE ORDER) and dimension tables (DATE, CUSTOMER, SUPPLIER, PART).
- Selected **hardware**.
- **System-specific common optimizations applied** to the data model.

A typical Big Data OLAP scenario with a **very large fact table (LINE_ORDER) and big dimension (CUSTOMER, PART, SUPPLIER and DATE) tables has been configured for the tests.** The **volume of rows to be loaded** in the tables can be configured for the SSB data generator by indicating the row values for each table in a configuration file. Also the cardinality can be increased by indicating a scaling factor when executing the data generator.

The number of rows in tables has been configured with the aim since the simplest test it represents a challenge for OLAP systems over traditional relational technology, such as PostgreSQL. For the most complex tests, the number of rows is high enough to pose a challenge for current Big Data OLAP systems.

Table 1. shows the configuration in number of rows per table for each of the three tests performed.

	LINEORDER	CUSTOMER	PART	SUPPLIER	DATE
Test – Table Role	Fact (KPI)	Dimension	Dimension	Dimension	Dimension
100M	100.000.000	40.000	32.000	20.000	2.556
500M	500.000.000	200.000	48.000	100.000	2.556
1.000M	1.000.000.000	400.000	56.000	200.000	2.556

Table 1. Number of rows for each table in the 3 tests performed.

In order to implement the 3 proposed tests with the OLAP systems of the benchmark, we chose the **hardware and software** versions shown in Table 2.

Tool and Version	Distributed Processing	Infrastructure	Cluster Hosts	Processor	Cores	RAM Memory
Kylin 2.4	Yes	Dedicated Cloud	3	Intel(R) Atom(TM) CPU C2750 @ 2.40GHz	8	32 Gb
Vertica 9.1	Yes	Dedicated Cloud	3	Intel(R) Atom(TM) CPU C2750 @ 2.40GHz	8	32 Gb
PostgreSQL 9.6	No	Dedicated Cloud	1	Intel(R) Atom(TM) CPU C2750 @ 2.40GHz	8	32 Gb

Table 2. Hardware and Software versions used in the benchmark.

For Kylin and Vertica we have used the same hardware resources. However, in the case of PostgreSQL only one instance has been used, as it does not support distributed processing thus cannot be deployed in a cluster mode.

Moreover, it is important to highlight that while Vertica operates in an exclusive cluster, Kylin has been installed on one of the machines of a Hadoop cluster that uses a Hortonworks distribution. Thus Kylin shares resources with other tools deployed on Hadoop cluster. In practice it is common to install Kylin on a machine separated from the Hadoop cluster or even deploy Kylin in cluster mode (Kylin cluster) to improve building (load) and query processes performance.

Finally, we have applied some **system-specific common techniques** to implement the SSB proposed data model for each of the three tools benchmarked. However, we have only applied those optimizations that are usually applied for any implementation in order to perform the benchmark in equality of conditions:

- Apache Kylin
  - Some dimension tables columns were defined as "Normal" in order to increase the pre-aggregation in the resulting OLAP cube (M-OLAP approach). Queries involving "Normal" dimension columns will require minimal computation in query time.
- Vertica
  - We have used the Vertica model auto optimizer using the benchmark queries as a source. The result is the automatic optimization of the original schema through the generation of projections. Projections are sub-sets of the model optimized (ordered data, using most appropriate encoding, ...) to provide better query latency for the most frequent types of analytical queries executed in that OLAP system.
- PostgreSQL.
  - Indexes are defined for the columns used to join fact table (Foreign Keys) and dimension tables (Surrogate Keys). It is a common technique used for the implementation of OLAP models over relational databases (aka R-OLAP).

Benchmark results and analysis

In order to obtain latency samples from each query with the highest accuracy, we have used the time measurement tools provided by Kylin, Vertica and PostgreSQL. The result of each query is obtained from the average of 20 samples. In addition, in the case of Kylin the query cache has been disabled to avoid the additional benefit of using it.

Table 3 shows the results of the tests. To better analyze the results, we have used the following color coding:

- Green: Query response time (latency) less than 5 seconds. Suitable to allow the interactive querying (OLAP) of the data.
- Blue: Query latency < 10 seconds. The user is aware of a small delay when execute such queries, but in many cases this time is acceptable.
- Orange: Query Latency < 20 seconds. Interactivity is seriously affected by query response times. In many OLAP scenarios this execution times could be considered unacceptable.
- Red: Query latency >= 20. Unacceptable times for an OLAP system, as the interactivity cannot be maintained.

Test	P1 – 100M (seconds)			P1 – 500M (seconds)			P1 – 1.000M (seconds)		
Query	Kylin	Vertica	PostgreSQL	Kylin	Vertica	PostgreSQL	Kylin	Vertica	PostgreSQL
Q1.1	0.2	0.2	22.4	0.3	0.3	+280	0.6	0.6	-
Q1.2	0.2	0.4	18.7	0.3	0.2	+280	0.5	0.3	-
Q1.3	0.2	0.4	18.5	0.3	0.3	+280	0.6	0.2	-
Q2.1	0.3	1.1	18.1	0.4	2.7	+280	0.6	9.1	-
Q2.2	0.3	0.8	16.3	0.4	2.7	+280	0.7	8.2	-
Q2.3	0.3	0.8	15.2	0.4	2.2	+280	0.6	7.4	-
Q3.1	0.3	1.4	23.9	0.4	3.7	+280	0.8	15.1	-
Q3.2	0.6	0.7	18.5	0.8	0.7	+280	0.9	9.8	-
Q3.3	0.3	0.9	15.8	0.3	0.6	+280	0.7	3.7	-
Q3.4	0.2	0.6	15.9	0.2	0.2	+280	0.2	1.0	-
Q4.1	0.3	1.4	23.7	0.4	7.3	+280	0.7	14.7	-
Q4.2	0.3	1.0	23.3	0.4	2.0	+280	0.7	3.8	-
Q4.3	2.5	0.8	17.1	2.4	1.3	+280	2.9	2.0	-

Table 3. Results of the tests performed.

According to the results obtained, **Kylin is the tool that achieves the best performance for query latency**. Kylin achieves a query latency of less than 5 seconds in all tests and all queries are executed is less than 1 second, except Q4.3.

In the case of Vertica, the results vary depending on the test. In tests with 100M and 500M of rows in the fact table, Vertica manages to execute any of the benchmark queries in less than 5 seconds and more than half of them in less than 1 second.

However, **in the most severe test 1000M**, with a fact table of 1 billons of rows, **Vertica achieves slightly worse results than Kylin**, running two of the 13 queries over 10 seconds. Even so, Vertica gets very good times considering the volume of data and also we are aware that more optimizations could be applied to the Vertica schema, such as the use of flattened tables or the RAM usage settings per query (user quota).

As expected, **PostgreSQL achieves the worst results**. Even so, in the first test PostgreSQL achieves execution times between 15 and 25 seconds that are not especially bad considering that PostgreSQL is not a Big Data technology. However, in the 500M test, queries are automatically aborted by the system after about 280 seconds without response.

Therefore, we decided not to perform the 1,000M test with PostgreSQL, since PostgreSQL failed to provide an answer in a reasonable time for the 500M test. This demonstrates that Kylin and Vertica surpass by far a traditional system like PostgreSQL, which in StrateBI we have used successfully in many implementations for OLAP systems that not use Big Data sources.

The following Figure shows the results obtained by Vertica and Kylin per query in each test performed.

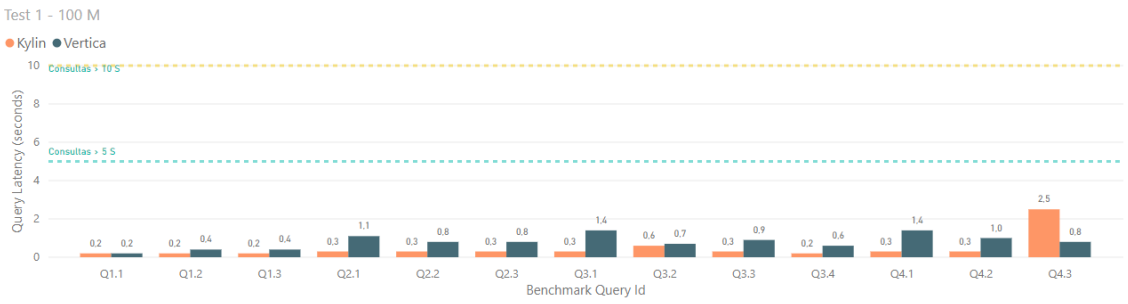


Figure 4. Query latency detailed analysis for the 3 test performed.

The comparison of results between Kylin and Vertica can be analyzed in Figure 5. **In the case of Kylin, we observe that the query latency barely increases when the size of the data set is significantly increased.**

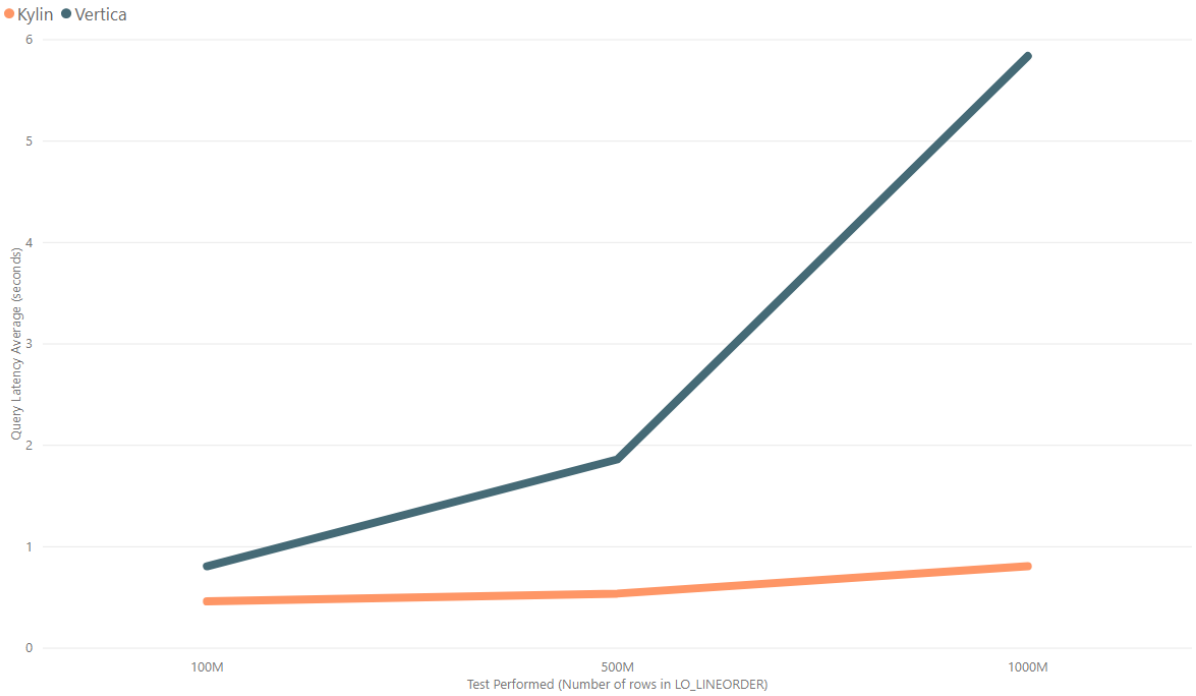


Figure 5. Relationship between number of rows and query latency between Kylin and Vertica

This is due to the fact this tool is based on data pre-aggregation so the computation and disk I/O required during query time are much lower than in the case of Vertica. For Vertica we have observed that query latency average increases considerably as the volume of data increases, especially for the most aggressive 1 Billion rows test. **However, the 6-second average query latency achieved by Vertica is still a very good and appropriate execution time for a Big Data OLAP scenario.**

Moreover, **to compare Kylin and Vertica we consider necessary to take into account other points** such as data loading time. **Schema or cube data load and refresh time is a very important feature** to consider when working with Big Data scenarios.

In the case of Kylin, most of the computation is done during load time (cube building or refresh phase), to pre-aggregate and pre-combine the data. This is performed by a Hadoop - MapReduce processes generated by Kylin that require significant hardware resources in the Hadoop cluster to achieve good loading times for historical data.

In Kylin, the load of the billions of rows for 1000M test was performed in about 16 hours by our Kylin cluster. This load time is quite elevated considering that in Vertica the same process took about 2 hours and cluster resource consumption was lower than in Kylin (80% of Hadoop cluster resources, compared to 40% in the Vertica cluster). However, Kylin is able to add or reload data from periods such as days or weeks in a few minutes, a very reasonable time for incremental data loading.

In addition, in Kylin it is possible to continue querying historical data while it is being updated, so the query service is never interrupted. Moreover, it is also important to highlight that the latest versions of Kylin already enable data loading using Spark generated processes instead of original Map Reduce, resulting in more efficient cube building and updating processes that can significantly speed up the loading process.

4. CONCLUSIONS

In this benchmark we have tested the query performance (query latency) for two of the most powerful Big Data OLAP technologies on the market, Kylin and Vertica. In addition, we compared these Big Data OLAP technologies with a non-Big Data relational database technology, PostgreSQL.

The result of the benchmark is that **both Kylin and Vertica are technologies suitable for Big Data OLAP scenarios** where is required to run interactive queries over analytical models with tables of up to several billion rows.

However, **test results have shown that Kylin is significantly faster in query performance than Vertica**, especially in the test of 1 billion rows (1000M) in the LINEORDER fact table. In addition, unlike Vertica, Kylin is a 100% open source tool with no limitations. Vertica has a free version (not open source) limited in number of cluster nodes (3) and storage (1TB). Although in many projects it is not necessary to overcome these limitations.

Vertica has achieved results very close to Kylin in the first two tests and, for the third test, we consider still possible to apply more optimizations to achieve better results. For this reason, we consider that **Vertica achieves some very good query execution times that make it an alternative to Kylin, easier to install, with lower hardware requirements and it does not require a Hadoop cluster to work.** Not needing a Hadoop cluster to operate can be an advantage in many projects where other Hadoop tools (e.g. Spark or Kafka) are not required to process any kind of Big Data source. Deploying and managing a Hadoop cluster also require greater knowledge and maintenance than an isolated Vertica cluster.

5. ANNEX 1: SSB BENCHMARK QUERIES

Q1.1

```
select sum(v_revenue) as revenue
from SSB.p_lineorder
left join SSB.dates on lo_orderdate = d_datekey
where d_year = 1993
and lo_discount between 1 and 3
and lo_quantity < 25
```

Q1.2

```
select sum(v_revenue) as revenue
from SSB.p_lineorder
left join SSB.dates on lo_orderdate = d_datekey
where d_yearmonthnum = 199401
and lo_discount between 4 and 6
and lo_quantity between 26 and 35;
```

Q1.3

```
select sum(v_revenue) as revenue
from SSB.p_lineorder
left join SSB.dates on lo_orderdate = d_datekey
where d_weeknuminyear = 6 and d_year = 1994
and lo_discount between 5 and 7
and lo_quantity between 26 and 35;
```

Q2.1

```
select sum(lo_revenue) as lo_revenue, d_year, p_brand
from SSB.p_lineorder
left join SSB.dates on lo_orderdate = d_datekey
left join SSB.part on lo_partkey = p_partkey
left join SSB.supplier on lo_suppkey = s_suppkey
```

```
where p_category = 'MFGR#12' and s_region = 'AMERICA'
group by d_year, p_brand
order by d_year, p_brand;
```

Q2.2

```
select sum(lo_revenue) as lo_revenue, d_year, p_brand
from SSB.p_lineorder
left join SSB.dates on lo_orderdate = d_datekey
left join SSB.part on lo_partkey = p_partkey
left join SSB.supplier on lo_suppkey = s_suppkey
where p_brand between 'MFGR#2221' and 'MFGR#2228' and s_region = 'ASIA'
group by d_year, p_brand
order by d_year, p_brand;
```

Q2.3

```
select sum(lo_revenue) as lo_revenue, d_year, p_brand
from SSB.p_lineorder
left join SSB.dates on lo_orderdate = d_datekey
left join SSB.part on lo_partkey = p_partkey
left join SSB.supplier on lo_suppkey = s_suppkey
where p_brand = 'MFGR#2239' and s_region = 'EUROPE'
group by d_year, p_brand
order by d_year, p_brand;
```

Q3.1

```
select c_nation, s_nation, d_year, sum(lo_revenue) as lo_revenue
from SSB.p_lineorder
left join SSB.dates on lo_orderdate = d_datekey
left join SSB.customer on lo_custkey = c_custkey
left join SSB.supplier on lo_suppkey = s_suppkey
where c_region = 'ASIA' and s_region = 'ASIA'and d_year >= 1992 and d_year <= 1997
group by c_nation, s_nation, d_year
```

```
order by d_year asc, lo_revenue desc;
```

Q3.2

```
select c_city, s_city, d_year, sum(lo_revenue) as lo_revenue
from SSB.p_lineorder
left join SSB.dates on lo_orderdate = d_datekey
left join SSB.customer on lo_custkey = c_custkey
left join SSB.supplier on lo_suppkey = s_suppkey
where c_nation = 'UNITED STATES' and s_nation = 'UNITED STATES'
and d_year >= 1992 and d_year <= 1997
group by c_city, s_city, d_year
order by d_year asc, lo_revenue desc;
```

Q3.3

```
select c_city, s_city, d_year, sum(lo_revenue) as lo_revenue
from SSB.p_lineorder
left join SSB.dates on lo_orderdate = d_datekey
left join SSB.customer on lo_custkey = c_custkey
left join SSB.supplier on lo_suppkey = s_suppkey
where (c_city='UNITED KI1' or c_city='UNITED KI5')
and (s_city='UNITED KI1' or s_city='UNITED KI5')
and d_year >= 1992 and d_year <= 1997
group by c_city, s_city, d_year
order by d_year asc, lo_revenue desc;
```

Q3.4

```
select c_city, s_city, d_year, sum(lo_revenue) as lo_revenue
from SSB.p_lineorder
left join SSB.dates on lo_orderdate = d_datekey
left join SSB.customer on lo_custkey = c_custkey
left join SSB.supplier on lo_suppkey = s_suppkey
```

where (c_city='UNITED KI1' or c_city='UNITED KI5') and (s_city='UNITED KI1' or s_city='UNITED KI5') and d_yearmonth = 'Dec1997'

group by c_city, s_city, d_year

order by d_year asc, lo_revenue desc;

Q4.1

select d_year, c_nation, sum(lo_revenue) - sum(lo_supplycost) as profit

from SSB.p_lineorder

left join SSB.dates on lo_orderdate = d_datekey

left join SSB.customer on lo_custkey = c_custkey

left join SSB.supplier on lo_suppkey = s_suppkey

left join SSB.part on lo_partkey = p_partkey

where c_region = 'AMERICA' and s_region = 'AMERICA' and (p_mfgr = 'MFGR#1' or p_mfgr = 'MFGR#2')

group by d_year, c_nation

order by d_year, c_nation;

Q4.2

select d_year, s_nation, p_category, sum(lo_revenue) - sum(lo_supplycost) as profit

from SSB.p_lineorder

left join SSB.dates on lo_orderdate = d_datekey

left join SSB.customer on lo_custkey = c_custkey

left join SSB.supplier on lo_suppkey = s_suppkey

left join SSB.part on lo_partkey = p_partkey

where c_region = 'AMERICA'and s_region = 'AMERICA'

and (d_year = 1997 or d_year = 1998)

and (p_mfgr = 'MFGR#1' or p_mfgr = 'MFGR#2')

group by d_year, s_nation, p_category

order by d_year, s_nation, p_category;

Q4.3

select d_year, s_city, p_brand, sum(lo_revenue) - sum(lo_supplycost) as profit

from SSB.p_lineorder

```
left join SSB.dates on lo_orderdate = d_datekey
left join SSB.customer on lo_custkey = c_custkey
left join SSB.supplier on lo_suppkey = s_suppkey
left join SSB.part on lo_partkey = p_partkey
where c_region = 'AMERICA'and s_nation = 'UNITED STATES'
and (d_year = 1997 or d_year = 1998)
and p_category = 'MFGR#14'
group by d_year, s_city, p_brand
order by d_year, s_city, p_brand;
```

6. ABOUT STRATEBI

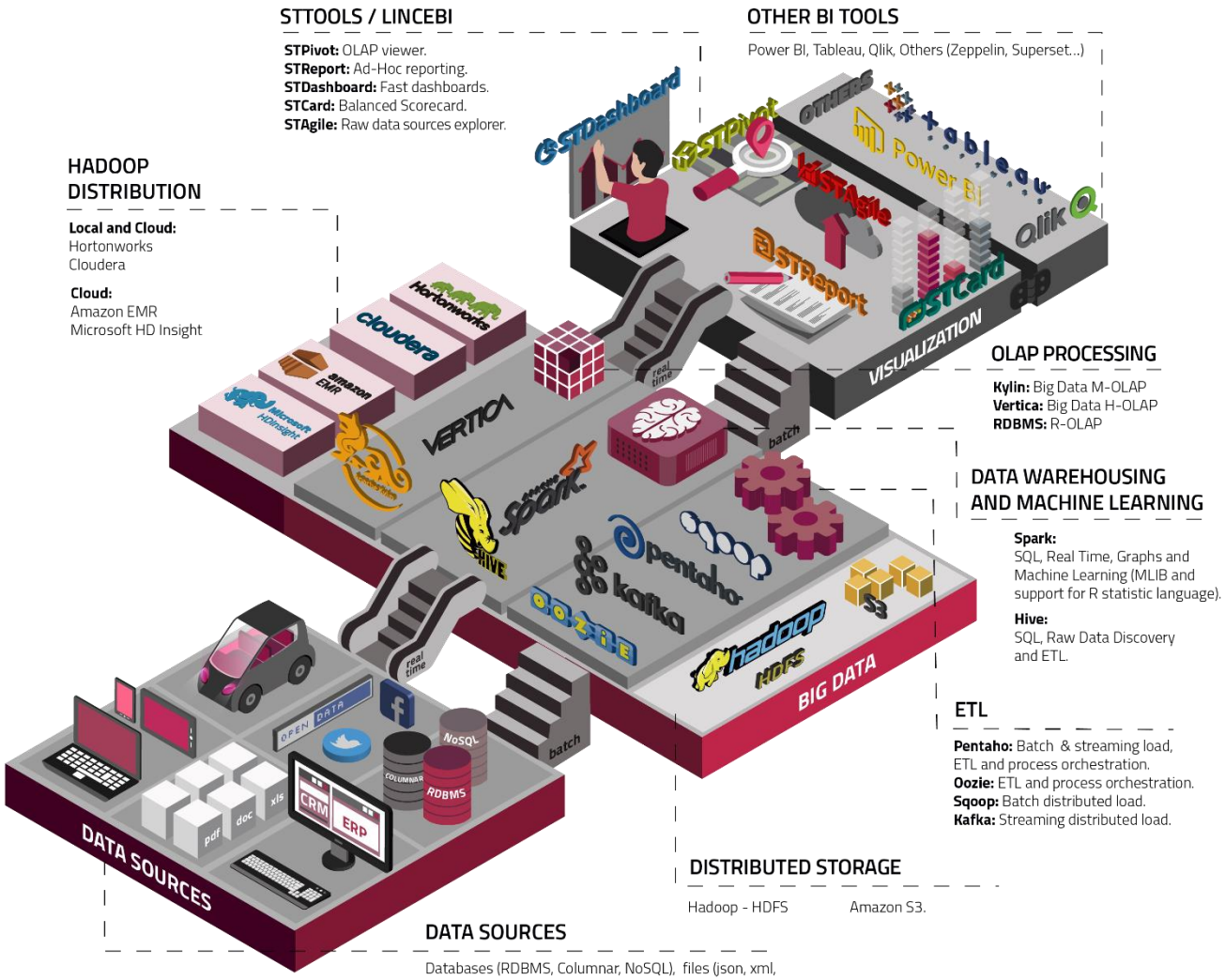
StrateBI is a Spanish based company, with offices in Madrid, Barcelona, Seville and Alicante, created by a group of professionals with extensive experience in information systems, technological solutions and processes related to Open Source solutions for Business Intelligence.

www.stratebi.com

Our experience in Big Data is demonstrated by the numerous **successful deployments carried out of Big Data systems based on Hadoop distributions using tools such as Spark, Hive or Kafka, OLAP systems such as Kylin, Vertica or Amazon Redshift or other NoSQL such as Neo4J**. These solutions have been successfully implemented in sectors as diverse as digital marketing, retail, pharmaceutical industry, tourism, education, cybersecurity or aeronautics.

In addition, we complement these Big Data solutions with a very powerful **ETL tools (Pentaho and Talend) and BI tools (Pentaho, Power BI or Tableau)**, in which we have extensive experience in projects using or not Big Data sources. We also provide **training** in all these technologies and we are official **partners of some of them such as Hortonworks, Vertica or Talend.**

Stratebi has created the solution Big Data Analytics LinceBI.com, based on Open Source technologies.

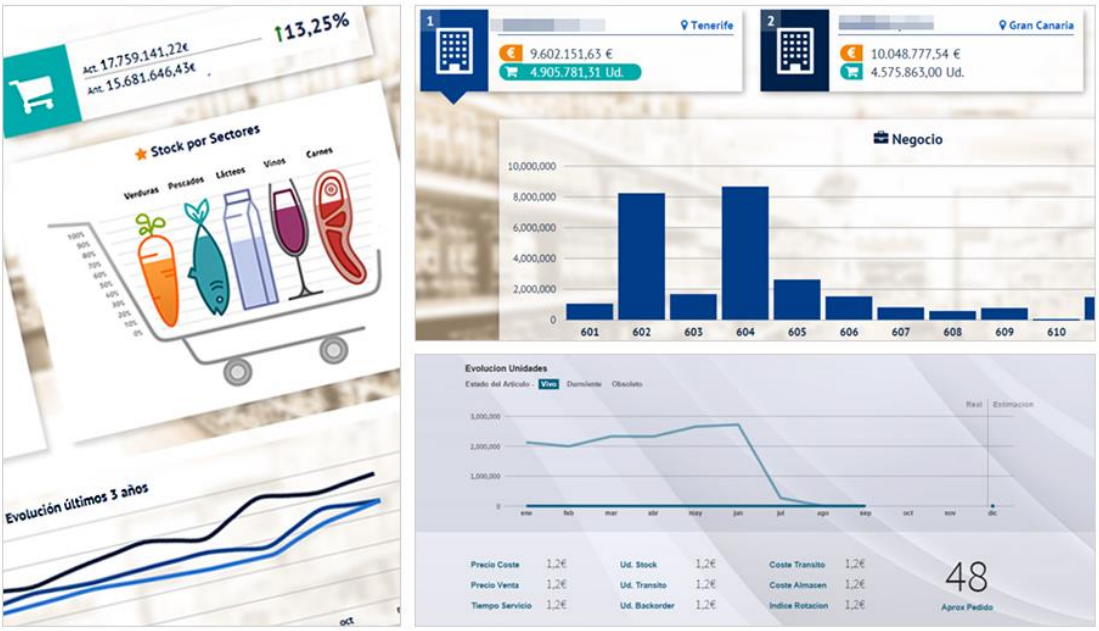

STTOOLS / LINCEBI
STPivot: OLAP viewer.
STReport: Ad-Hoc reporting.
STDashboard: Fast dashboards.
STCard: Balanced Scorecard.
STAgile: Raw data sources explorer.
OTHER BI TOOLS
Power BI, Tableau, Qlik, Others (Zeppelin, Superset...)
HADOOP DISTRIBUTION
Local and Cloud:
Hortonworks
Cloudera
Cloud:
Amazon EMR
Microsoft HD Insight
VISUALIZATION
OLAP PROCESSING
Kylin: Big Data M-OLAP
Vertica: Big Data H-OLAP
RDBMS: R-OLAP
DATA WAREHOUSING AND MACHINE LEARNING
Spark:
SQL, Real Time, Graphs and Machine Learning (MLIB and support for R statistic language).
Hive:
SQL, Raw Data Discovery and ETL.
BIG DATA
ETL
Pentaho: Batch & streaming load, ETL and process orchestration.
Oozie: ETL and process orchestration.
Sqoop: Batch distributed load.
Kafka: Streaming distributed load.
DISTRIBUTED STORAGE
Hadoop - HDFS
Amazon S3.
DATA SOURCES
Databases (RDBMS, Columnar, NoSQL), files (json, xml, txt...), internet of the things (IoT), Apps, etc.

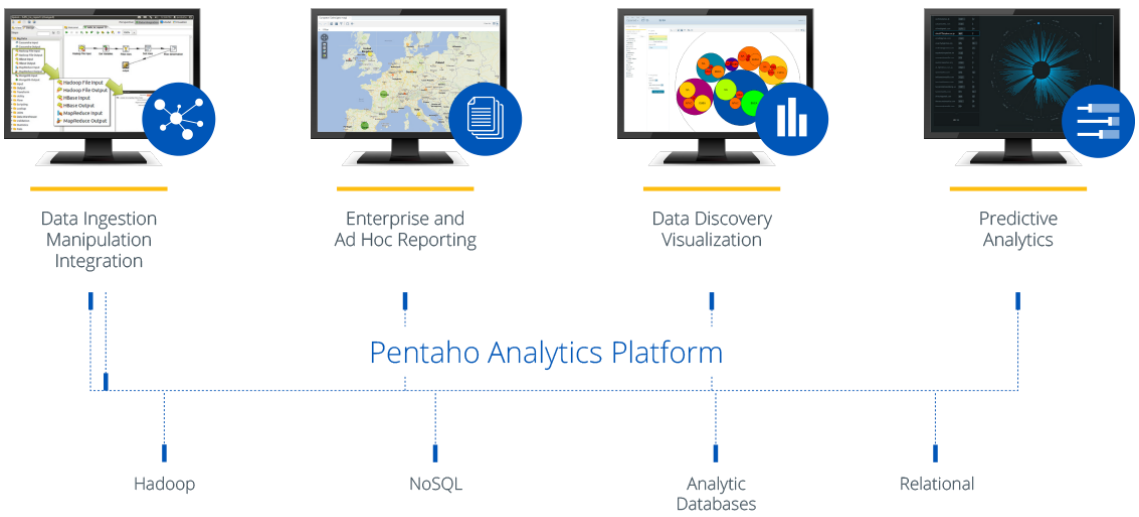
7. TECHNOLOGIES

Recently, we have been named Certified Partners of Vertica, Talend and Microsoft.

8. OUR ANALYTICS DEVELOPMENTS SAMPLES

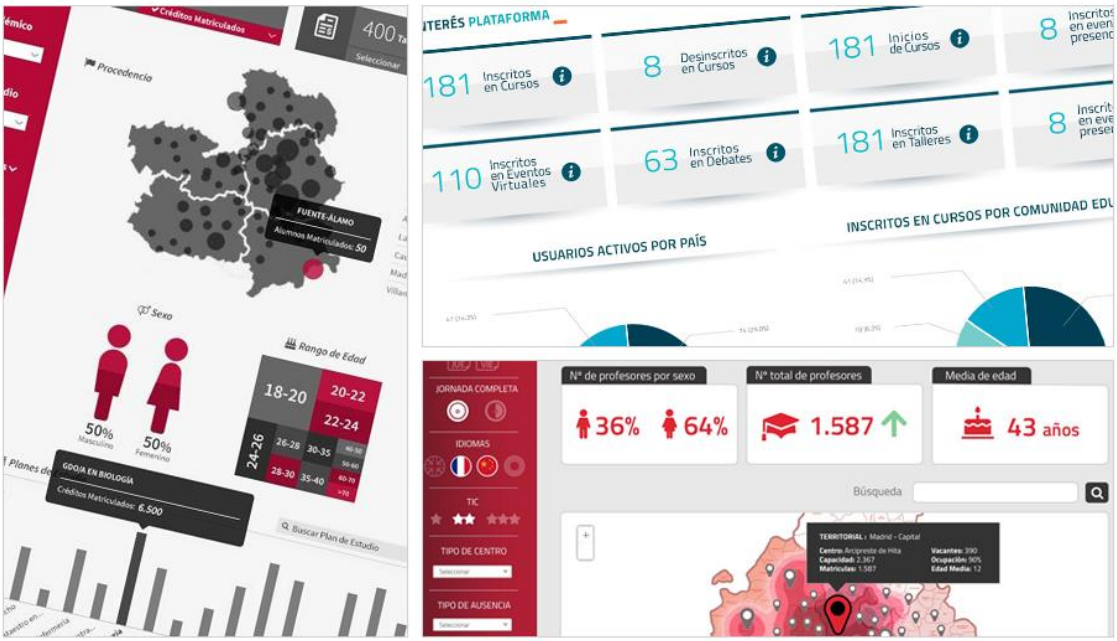
Below are some screenshots of dashboards and applications designed by StrateBI, in order to show you what you can get using our Big Data and BI solutions. Also you can try our Online Demos on the Stratebi website:

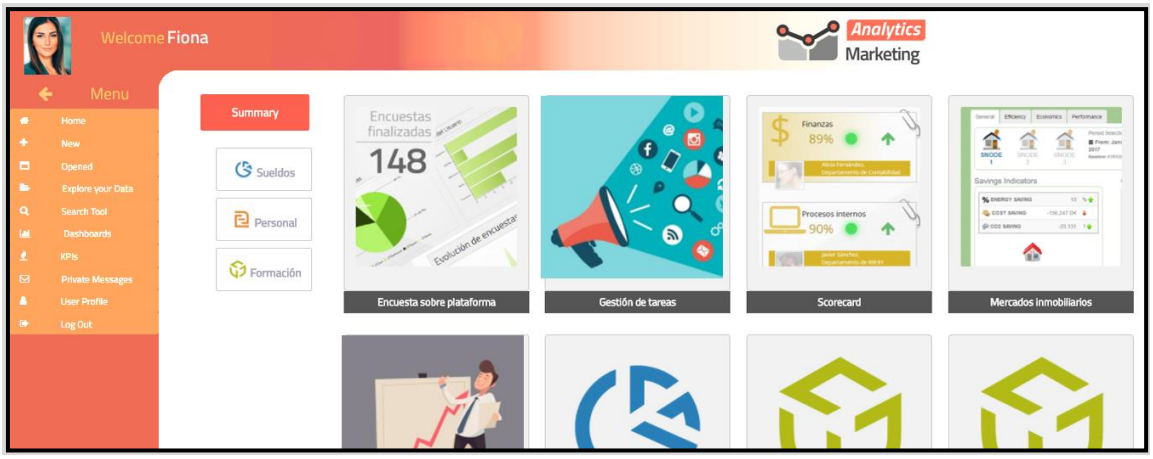





facebook DASHBOARD COMENTARIOS
Sexo
Sentimiento
Localización
65%
35%
28%
60%
12%
Tipo de recurso
Fotos
Videos
Status
Preguntas
Evolución Comentarios por días


smart MADRID
Distrito
Última Actualización
18:24
TOP 5 PUNTOS DE MEDICIÓN
PROXIMIDAD DE PUNTOS DE MEDIDA
EVOLUCIÓN INTENSIDAD DE TRÁFICO


Créditos Matriculados
Procedencia
FUENTE-ÁLAMO
Alumnos Matriculados: 50
Sexo
50% Masculino
50% Femenino
Rango de Edad
18-20
20-22
22-24
24-26
26-28
28-30
30-35
35-40
GDO/A EN BIOLOGÍA
Créditos Matriculados: 6.500
Buscar Plan de Estudio
INTERÉS PLATAFORMA
181 Inscritos en Cursos
8 Desinscritos en Cursos
181 Inicios de Cursos
110 Inscritos en Eventos Virtuales
63 Inscritos en Debates
181 Inscritos en Talleres
USUARIOS ACTIVOS POR PAÍS
INSCRITOS EN CURSOS POR COMUNIDAD EDU
JORNADA COMPLETA
IDIOMAS
TIC
TIPO DE CENTRO
TIPO DE AUSENCIA
Nº de profesores por sexo
36% 64%
Nº total de profesores
1.587
Media de edad
43 años
Búsqueda


Welcome Fiona
Analytics
Marketing
Menu
Home
New
Opened
Explore your Data
Search Tool
Dashboards
KPIs
Private Messages
User Profile
Log Out
Summary
Sueldos
Personal
Formación
Encuestas finalizadas
148
Evolución de encuestas
Finanzas
89%
Procesos internos
90%
Savings Indicators
Encuesta sobre plataforma
Gestión de tareas
Scorecard
Mercados inmobiliarios