SQL VS NoSQL DATABASES

What is SQL ?

Structure Query Language(SQL) is a database query language used for storing and managing data in Relational DBMS. SQL was the first commercial language introduced for E.F Codd's **Relational** model of database. Today almost all RDBMS(MySql, Oracle, Infomix, Sybase, MS Access) use **SQL** as the standard database query language. SQL is used to perform all types of data operations in RDBMS.

Why SQL ?

- Allows users to access data in the relational database management systems.
- Allows users to describe the data.
- Allows users to define the data in a database and manipulate that data.
- Allows to embed within other languages using SQL modules, libraries & pre-compilers.
- Allows users to create and drop databases and tables.
- Allows users to create view, stored procedure, functions in a database.
- Allows users to set permissions on tables, procedures and views.

Advantages of SQL

- High Speed: SQL Queries can be used to retrieve large amounts of records from a database quickly and efficiently.
- Well Defined Standards Exist: SQL databases use long-established standard, which is being adopted by ANSI & ISO.
- Portable: SQL runs on mainframes, PCs, laptops, servers, tablets and smartphones. It runs in local systems, intranet and internet. Databases using SQL can be moved from device to another without problems.
- Client/Server language: SQL is used to connect front end computers (clients) and back end databases (servers). Thus, supporting the client-server architecture.
- Open Source: Free databases from MySQL, MariaDB and PostGreSQL means, SQL databases can be used at low cost and with large communities behind them.

Disadvantages of SQL

- Partial Control –Due to hidden business rules, complete control is not given to the database
- Difficulty in Interfacing: Interfacing an SQL database is more complex than adding a few lines of code.
- Security: Regardless of the SQL version, databases in SQL is constantly under threat as it holds huge amounts of sensitive data.

Application of SQL

- Data Definition Language (DDL) : This allows users to create a database on their own that may be structured, used, and then deleted once the purpose has been fulfilled.
- Data Control Language (DCL) : DCL commands are used to grant and take back authority from any user.
- Data Manipulation Language (DML) : These commands are used to modify the databases and is responsible for all form of modifications in the database. It is not auto-committed which means the changes made to the databases are not permanently saved.
- Transaction Control Language (TCL) : TCL commands can only be used with DML commands like INSERT, DELETE and UPDATE. These operations can't be used while creating tables because they are automatically committed to the database.

Common SQL Databases

- Oracle : Oracle is the most widely used corporate standard database. It is built to store massive amounts of data, supports PLSQL, includes extra commands such as tablespace, synonyms, and packages, and includes numerous backup techniques. It's supported by a wide range of programming languages.
- Postgresql : It has a limited set of applications and, as a result, is slower than other SQL databases. It is, however, particularly good at connecting several tables and looking through vast volumes of data. Unlike MySQL and MariaDB, it also provides support for PLSQL. For complex data models with many tables and complex queries, PostgreSQL should be used. When PLSQL support is required, it can be used as a free alternative to Oracle.

NOSQL Databases(MongoDB)

- MongoDB is open source NoSQL document store database which is written in C++
- MongoDB stores records in Collections.
- Collections are similar to tables in a relational DB
- Each collection consist of documents
- Documents are stored in BSON format (Binary form of JSON)

Feature Highlights

- Rich Object Model: MongoDB supports a rich and expressive object model. Objects can have properties and objects can be nested in one another (for multiple levels).
- Secondary Indexes: Indexes speed up the queries significantly, but they also slow down writes. Secondary indexes are a first-class construct in MongoDB. This makes it easy to index any property of an object stored in MongoDB even if it is nested.
- Replication and high availability: MongoDB supports a "single master" model. This means you have a master node and a number of slave nodes. In case the master goes down, one of the slaves is elected as master. During the time of new leader election, your replica set is down and cannot take writes.
- Native Aggregation: MongoDB has a built-in Aggregation framework to run an ETL(Extract, transform and load) pipeline to transform the data stored in the database.

Why Choose MongoDB

- The document data model is a powerful way to store and retrieve data that allows developers to move fast.
- MongoDB’s horizontal, scale-out architecture can support huge volumes of both data and traffic.
- MongoDB has a great user experience for developers who can install MongoDB and start writing code immediately.
- MongoDB can be used everywhere by anyone.
- MongoDB has developed a large and mature platform ecosystem.

When to choose MongoDB

- Integrating large amounts of diverse data: If you are bringing together tens or hundreds of data sources, the flexibility and power of the document model can create a unified single view in ways that other databases cannot.
- Describing complex data structures that evolve: Document databases allow embedding of documents to describe nested structures and easily tolerate variations in data in generations of documents.
- Delivering data in high-performance applications: MongoDB’s scale-out architecture can support huge numbers of transactions on humongous databases.
- Supporting hybrid and multi-cloud applications: MongoDB can be deployed and run on a desktop, a huge cluster of computers in a data center, or in a public cloud, either as installed software or through MongoDB Atlas, a database as a service product.

Advantages of MongoDB

- MongoDB is schema less. It is a document database in which one collection holds different documents.
- There may be difference between number of fields, content and size of the document from one to other.
- Structure of a single object is clear in MongoDB.
- There are no complex joins in MongoDB.
- MongoDB provides the facility of deep query because it supports a powerful dynamic query on documents.
- It is very easy to scale.
- It uses internal memory for storing working sets and this is the reason of its fast access.

Implementation

- We conducted a couple of experiments to measure performance of SQL and NoSQL database.
- For SQL we have chosen Postgres DB and for NoSQL we chose Mongodb.
- Editors for DB
  - In mongo we can either use mongo shell or mongo Compass to perform queries.
  - For Postgres we used pgadmin 4.
- We took a sales record dataset of 5 million records.
- We used CRUD operations and some advance queries to measure performance between both databases.
- Performance was measured on the basis of time taken for a query to complete.

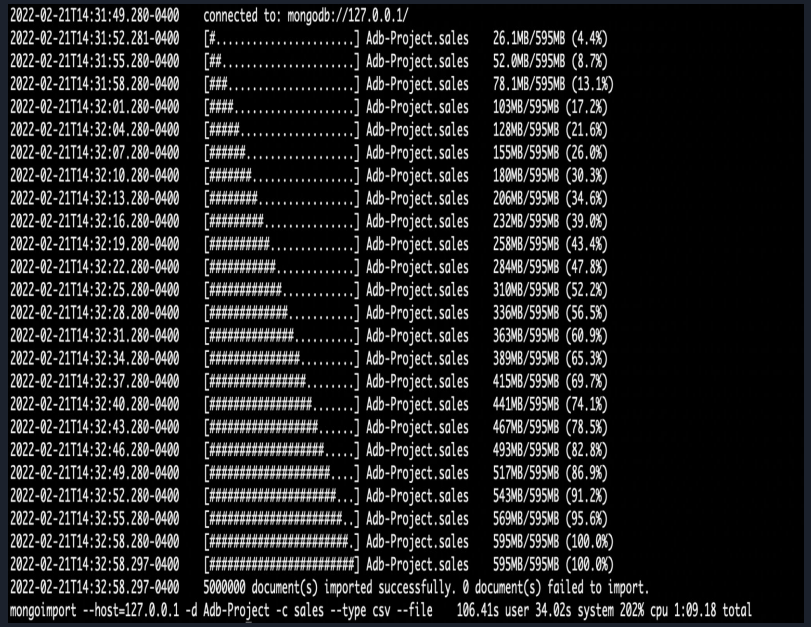
Brief On Data

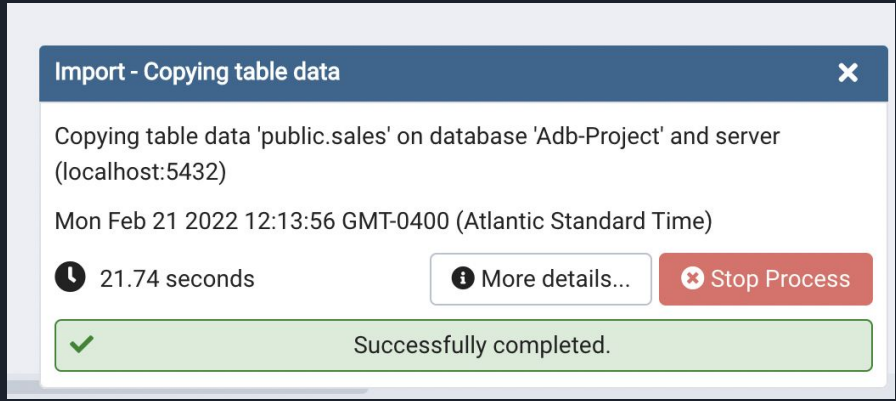
5m Sales Records

Region	Country	Item Type	Sales Channel	Order Priority	Order Date	Order ID	Ship Date	Units Sold	Unit Price	Unit Cost	Total Revenue	Total Cost	Total Profit
Australia and Oceania	Palau	Office Supplies	Online	H	3/6/2016	517073523	3/26/2016	2401	651.21	524.96	1563555.21	1260428.96	303126.25
Europe	Poland	Beverages	Online	L	4/18/2010	380507028	5/26/2010	9340	47.45	31.79	443183.00	296918.60	146264.40
North America	Canada	Cereal	Online	M	1/8/2015	504055583	1/31/2015	103	205.70	117.11	21187.10	12062.33	9124.77
Europe	Belarus	Snacks	Online	C	1/19/2014	954955518	2/27/2014	1414	152.58	97.44	215748.12	137780.16	77967.96
Middle East and North Africa	Oman	Cereal	Offline	H	4/26/2019	970755660	6/2/2019	7027	205.70	117.11	1445453.90	822931.97	622521.93
Sub-Saharan Africa	Burkina Faso	Office Supplies	Online	C	3/3/2012	309317338	4/5/2012	2729	651.21	524.96	1777152.09	1432615.84	344536.25
Europe	Montenegro	Personal Care	Online	H	11/24/2012	598814380	12/25/2012	1337	81.73	56.67	109273.01	75767.79	33505.22
Middle East and North Africa	Azerbaijan	Cosmetics	Offline	M	3/18/2011	387733113	5/5/2011	7699	437.20	263.33	3366002.80	2027377.67	1338625.13
Sub-Saharan Africa	South Sudan	Clothes	Offline	C	5/10/2014	994872367	6/17/2014	3696	109.28	35.84	403898.88	132464.64	271434.24
North America	Greenland	Personal Care	Online	C	5/25/2020	659343469	6/14/2020	3239	81.73	56.67	264723.47	183554.13	81169.34
Europe	Portugal	Cosmetics	Online	M	11/12/2012	390570247	11/13/2012	7270	437.20	263.33	3178444.00	1914409.10	1264034.90
Sub-Saharan Africa	Sierra Leone	Clothes	Offline	C	7/25/2019	818289029	7/27/2019	8763	109.28	35.84	957620.64	314065.92	643554.72
Europe	Germany	Personal Care	Offline	H	12/25/2010	842548644	1/24/2011	7722	81.73	56.67	631119.06	437605.74	193513.32
Europe	Montenegro	Personal Care	Offline	L	8/3/2012	783710420	8/10/2012	6184	81.73	56.67	505418.32	350447.28	154971.04
Asia	Bangladesh	Cereal	Offline	L	12/3/2013	132448184	1/15/2014	3216	205.70	117.11	661531.20	376625.76	284905.44
Sub-Saharan Africa	Mauritius	Snacks	Online	M	9/22/2016	699748265	10/3/2016	9917	152.58	97.44	1513135.86	966312.48	546823.38
Asia	Kyrgyzstan	Snacks	Offline	H	11/22/2018	766909492	12/15/2018	4416	152.58	97.44	673793.28	430295.04	243498.24
Sub-Saharan Africa	Madagascar	Clothes	Online	C	10/27/2019	935543191	10/29/2019	6875	109.28	35.84	751300.00	246400.00	504900.00
Europe	Romania	Meat	Online	M	1/25/2015	977573192	2/7/2015	4486	421.89	364.69	1892598.54	1635999.34	256599.20
Europe	Malta	Personal Care	Offline	H	5/21/2012	285235822	6/22/2012	6473	81.73	56.67	529038.29	366824.91	162213.38
Europe	Latvia	Clothes	Online	H	9/10/2015	121079909	10/2/2015	9091	109.28	35.84	993464.48	325821.44	667643.04
Asia	Bangladesh	Fruits	Online	L	9/14/2019	467966783	9/27/2019	625	9.33	6.92	5831.25	4325.00	1506.25
Middle East and North Africa	Yemen	Household	Online	C	4/18/2015	479070270	5/8/2015	6394	668.27	502.54	4272918.38	3213240.76	1059677.62
Asia	Myanmar	Vegetables	Offline	H	3/28/2016	727876698	4/19/2016	3745	154.06	90.93	576954.70	340532.85	236421.85
Sub-Saharan Africa	The Gambia	Office Supplies	Online	L	1/28/2011	958184897	3/7/2011	7059	651.21	524.96	4596891.39	3705692.64	891198.75
Europe	Russia	Fruits	Offline	H	3/26/2011	534106194	5/13/2011	2747	9.33	6.92	25629.51	19009.24	6620.27
Sub-Saharan Africa	Malawi	Snacks	Online	M	8/8/2013	690064418	8/29/2013	4251	152.58	97.44	648617.58	414217.44	234400.14
Middle East and North Africa	Oman	Cosmetics	Offline	M	8/1/2016	480795896	9/10/2016	7045	437.20	263.33	3080074.00	1855159.85	1224914.15
Central America and the Caribbean	Belize	Fruits	Offline	H	9/24/2014	831349456	10/20/2014	3633	9.33	6.92	33895.89	25140.36	8755.53
Central America and the Caribbean	Haiti	Baby Food	Online	M	10/17/2014	487086427	10/30/2014	8553	255.28	159.42	2183409.84	1363519.26	819890.58
Europe	Slovakia	Cosmetics	Offline	L	9/7/2019	963680636	10/1/2019	3370	437.20	263.33	1473364.00	887422.10	585941.90
Middle East and North Africa	Oman	Vegetables	Online	L	6/2/2012	697118413	7/16/2012	1685	154.06	90.93	259591.10	153217.05	106374.05
Middle East and North Africa	Qatar	Office Supplies	Offline	M	12/18/2014	893698585	2/2/2015	4126	651.21	524.96	2686892.46	2165984.96	520907.50
Sub-Saharan Africa	Namibia	Household	Offline	M	9/11/2018	580032479	9/29/2018	3355	668.27	502.54	2242045.85	1686021.70	556024.15
Sub-Saharan Africa	Democratic Republic of the Congo	Snacks	Online	L	2/22/2011	653043925	3/31/2011	9064	152.58	97.44	1382985.12	883196.16	499788.96
Europe	Slovenia	Cosmetics	Online	H	6/9/2013	729969251	7/19/2013	2978	437.20	263.33	1301981.60	784196.74	517784.86
Asia	Uzbekistan	Baby Food	Online	L	10/9/2010	498357284	10/12/2010	3850	255.28	159.42	982828.00	613767.00	369061.00
Europe	United Kingdom	Personal Care	Offline	H	3/27/2011	166069352	4/16/2011	2467	81.73	56.67	201627.91	139804.89	61823.02
Sub-Saharan Africa	Zambia	Cosmetics	Online	C	1/26/2019	911279284	2/12/2019	6646	437.20	263.33	2905631.20	1750091.18	1155540.02
Europe	Poland	Clothes	Online	H	8/21/2018	832473409	9/27/2018	6236	109.28	35.84	681470.08	223498.24	457971.84

Experiment 1
Importing the CSV file

- File size is around 600mb.
- Mongo took around 1 minute 10 seconds to complete the import.
- Postgres took only 20 seconds to complete the operation.
- In mongo we had to install mongo dev tools to import CSV.
- Postgres has in built functionality to import CSV.


2022-02-21T14:31:49.280-0400 connected to: mongodb://127.0.0.1/
2022-02-21T14:31:52.281-0400 [#.....] Adb-Project.sales 26.1MB/595MB (4.4%)
2022-02-21T14:31:55.280-0400 [##.....] Adb-Project.sales 52.0MB/595MB (8.7%)
2022-02-21T14:31:58.280-0400 [###.....] Adb-Project.sales 78.1MB/595MB (13.1%)
2022-02-21T14:32:01.280-0400 [###.....] Adb-Project.sales 103MB/595MB (17.2%)
2022-02-21T14:32:04.280-0400 [#####.....] Adb-Project.sales 128MB/595MB (21.6%)
2022-02-21T14:32:07.280-0400 [#####.....] Adb-Project.sales 155MB/595MB (26.0%)
2022-02-21T14:32:10.280-0400 [######.....] Adb-Project.sales 180MB/595MB (30.3%)
2022-02-21T14:32:13.280-0400 [#######.....] Adb-Project.sales 206MB/595MB (34.6%)
2022-02-21T14:32:16.280-0400 [########.....] Adb-Project.sales 232MB/595MB (39.0%)
2022-02-21T14:32:19.280-0400 [########.....] Adb-Project.sales 258MB/595MB (43.4%)
2022-02-21T14:32:22.280-0400 [#########.....] Adb-Project.sales 284MB/595MB (47.8%)
2022-02-21T14:32:25.280-0400 [##########.....] Adb-Project.sales 310MB/595MB (52.2%)
2022-02-21T14:32:28.280-0400 [##########.....] Adb-Project.sales 336MB/595MB (56.5%)
2022-02-21T14:32:31.280-0400 [###########.....] Adb-Project.sales 363MB/595MB (60.9%)
2022-02-21T14:32:34.280-0400 [############.....] Adb-Project.sales 389MB/595MB (65.3%)
2022-02-21T14:32:37.280-0400 [############.....] Adb-Project.sales 415MB/595MB (69.7%)
2022-02-21T14:32:40.280-0400 [#############.....] Adb-Project.sales 441MB/595MB (74.1%)
2022-02-21T14:32:43.280-0400 [#############.....] Adb-Project.sales 467MB/595MB (78.5%)
2022-02-21T14:32:46.280-0400 [##############.....] Adb-Project.sales 493MB/595MB (82.8%)
2022-02-21T14:32:49.280-0400 [##############....] Adb-Project.sales 517MB/595MB (86.9%)
2022-02-21T14:32:52.280-0400 [###############...] Adb-Project.sales 543MB/595MB (91.2%)
2022-02-21T14:32:55.280-0400 [###############..] Adb-Project.sales 569MB/595MB (95.6%)
2022-02-21T14:32:58.280-0400 [###############.] Adb-Project.sales 595MB/595MB (100.0%)
2022-02-21T14:32:58.297-0400 [################] Adb-Project.sales 595MB/595MB (100.0%)
2022-02-21T14:32:58.297-0400 5000000 document(s) imported successfully. 0 document(s) failed to import.
mongoimport --host=127.0.0.1 -d Adb-Project -c sales --type csv --file 106.41s user 34.02s system 202% cpu 1:09.18 total


Import - Copying table data
Copying table data 'public.sales' on database 'Adb-Project' and server (localhost:5432)
Mon Feb 21 2022 12:13:56 GMT-0400 (Atlantic Standard Time)
21.74 seconds
More details...
Stop Process
Successfully completed.

Experiment 2
Creating A backup table

- Generally, we need backup table for testing purposes.
- Mongo took around 1 minute 46 seconds to create backup
- Postgres took only 7 seconds to complete the operation.
- No command in mongo, had to use export and import from dev tools.
- SQL command for backup -

```
3  -- Experiment 2 (Creating Backup Table)
4
5  explain analyze create table sales_bckp as select * from sales;
6
```

```
2022-02-21T18:41:22.686-0400    [########################.]  Adb-Project.sales  4808000/5000000  (96.2%)
2022-02-21T18:41:23.686-0400    [########################.]  Adb-Project.sales  4856000/5000000  (97.1%)
2022-02-21T18:41:24.686-0400    [########################.]  Adb-Project.sales  4904000/5000000  (98.1%)
2022-02-21T18:41:24.984-0400    Adb-Project.sales_bckp  1.68GB
2022-02-21T18:41:25.686-0400    [########################.]  Adb-Project.sales  4952000/5000000  (99.0%)
2022-02-21T18:41:26.686-0400    [########################.]  Adb-Project.sales  4992000/5000000  (99.8%)
2022-02-21T18:41:26.729-0400    [########################]  Adb-Project.sales  5000000/5000000  (100.0%)
2022-02-21T18:41:26.729-0400    exported 5000000 records
2022-02-21T18:41:26.738-0400    Adb-Project.sales_bckp  1.70GB
2022-02-21T18:41:26.739-0400    5000000 document(s) imported successfully. 0 document(s) failed to import.
mongoexport -d Adb-Project -c sales  77.20s user 12.13s system 83% cpu 1:46.90 total
mongoimport -d Adb-Project -c sales_bckp --drop  274.52s user 53.43s system 306% cpu 1:46.90 total
```

Data Output | Explain | Messages | Notifications

	QUERY PLAN text
1	Seq Scan on sales (cost=0.00..136738.26 rows=5000426 width=101) (actual time=0.027..668.346 rows=5000000 loops=1)
2	Planning Time: 0.072 ms
3	Execution Time: 7263.572 ms

Experiment 3
Some various Selects

- Simple Select Query
  - Mongo took around only 10 seconds to fetch 5 million records.
  - Postgres took 11 seconds to complete the operation.
  - Finally Mongo wins 🚀
- Select with condition
  - Mongo took 2 seconds to look up through records
  - Postgres took only 800 milliseconds.
  - This is how the query looks like -

```
15  select * from sales where region='Asia' and sales_channel='Offline';
16
17  -- db.sales.find({Region: "Asia", "Sales Channel": "Offline"}).explain('executionStats') (1863ms)
```

Experiment 4
SQL Aggregate vs Mongo Aggregation Pipeline

- In SQL we can easily perform aggregate function.
  - We found the averages of sold units and even grouped them by region.
  - It took only 2 seconds.
  - This is how simple the query is -

```
11  select region, avg(units_sold) from sales group by region;
```

- In mongo we have to use aggregation pipelines. We have to create pipelines to perform any kind of aggregate function.
- It took 8 seconds to execute the pipeline.
- This is how query looks -

```
23  -- db.sales.aggregate( [ { $project: { _id: 1, Region: 1, "Units Sold": 1 }},
24  --{ $group : { _id: "$Region", averageUnitsSold : { $avg: "$Units Sold" }}}]) (8 Seconds)
```

Experiment 4
Nested Queries

- In this experiment we had to find records where revenue was greater than average revenue for Asia region.
- Time taken was around 1.5 seconds.
- We used the below query -

```
28
29  select avg(total_revenue) from sales where region='Asia';
30
31  select * from sales where total_revenue > (select avg(total_revenue) from sales where region='Asia') and region='Asia';
32
33
34
35
```

Data Output | Explain | Messages | Notifications

```
Successfully run. Total query runtime: 1 secs 425 msec.
247666 rows affected.
```

Experiment 4
Nested Queries

- In mongo the query get more complex when we are writing subqueries.
- We used two queries to get the same result.
- Time taken was around 2 seconds.

```
> const avgRevenue = db.sales.aggregate([{$project : {_id:0, Region:1, "Total Revenue": 1}},{$match : {Region:"Asia"}}, {$group : { _id: "$Region", averageRevenue : { $avg : "$Total
> avgRevenue
< { _id: 'Asia', averageRevenue: 1331297.1457965458 }
> db.sales.find({Region:"Asia","Total Revenue" : { $gt : 1331297.1457965458}}).explain('executionStats')
< { queryPlanner:
    { plannerVersion: 1,
      namespace: 'Adb-Project.sales',
      indexFilterSet: false,
      parsedQuery:
       { '$and':
          [ { Region: { '$eq': 'Asia' } },
            { 'Total Revenue': { '$gt': 1331297.1457965458 } } ] },
      winningPlan:
       { stage: 'COLLSCAN',
         filter:
          { '$and':
             [ { Region: { '$eq': 'Asia' } },
               { 'Total Revenue': { '$gt': 1331297.1457965458 } } ] },
         direction: 'forward' },
      rejectedPlans: [] },
   executionStats:
    { executionSuccess: true,
      nReturned: 247666,
      executionTimeMillis: 1906,
      totalKeysExamined: 0,
      totalDocsExamined: 5000000,
      executionStages:
```

Experiment 5
Update with nested select

- In this experiment we are updating total revenue of those columns where it is less than average by selecting the minimum value from the revenue above average.
- Time taken was around 18.5 seconds

```
39  update sales_bckp set total_revenue = (select min(total_revenue) from sales where total_revenue > (select avg(total_revenue)
40  from sales where region='Asia') and region='Asia') where region='Asia';
41
42
```

Data Output | Explain | Messages | Notifications

```
UPDATE 729864

Query returned successfully in 18 secs 431 msec.
```

- In mongo this query cannot be written. We have to use some server side language to perform such operation.

Experiment 6
Truncating tables

- SQL is fast as usual. It took only 65 milliseconds to remove all records.
- On the other hand Mongo for the same operation took 45 seconds.

```
51  truncate table sales_bckp
52
53
```

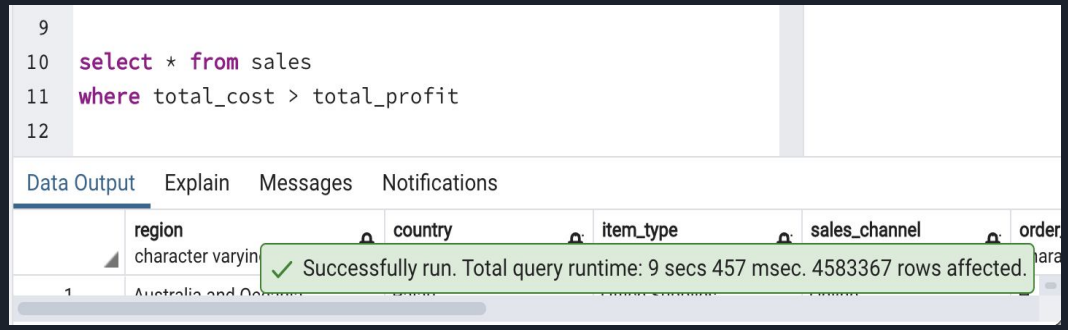
Data Output | Explain | Messages | Notifications

```
TRUNCATE TABLE

Query returned successfully in 65 msec.
```

Experiment 7
Comparison Condition

- Extracted records based on the comparison of two columns
- All the records are extracted having total cost > total profit
- SQL took 9.4 seconds



Experiment 7
Comparison Condition

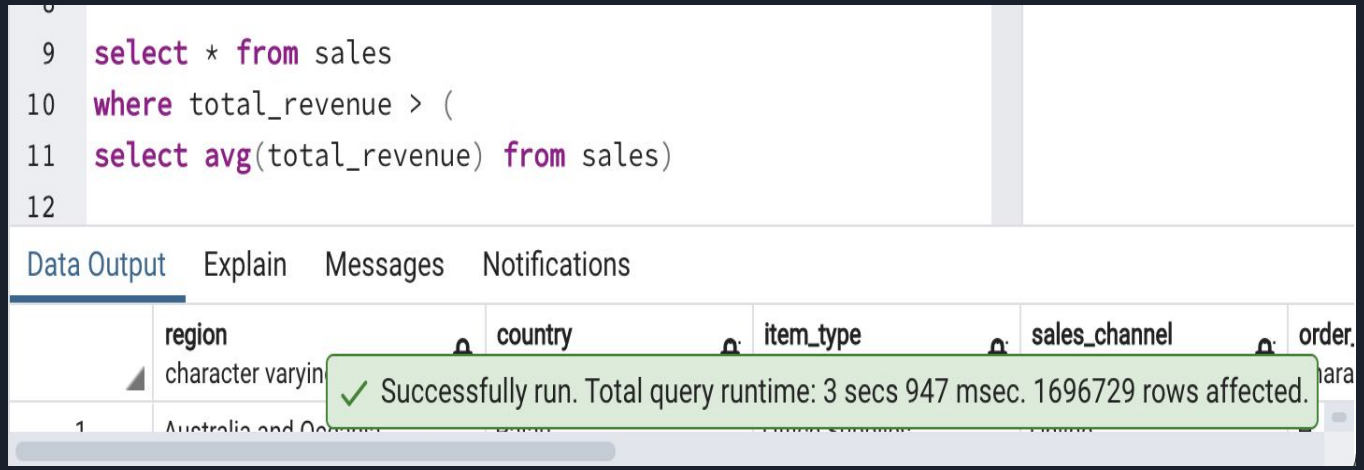
- MongoDB took 5.8 seconds

```
In [36]: collection.find({'$expr': {'$gt': ['$total_cost', '$total_profit']}}).explain()['executionStats']

Out[36]: {'executionSuccess': True,
          'nReturned': 4583367,
          'executionTimeMillis': 5808,
          'totalKeysExamined': 0,
          'totalDocsExamined': 5000000,
          'executionStages': {'stage': 'COLLSCAN',
           'filter': {'$expr': {'$gt': ['$total_cost', '$total_profit']}},
           'nReturned': 4583367,
           'executionTimeMillisEstimate': 122,
           'works': 5000002,
           'advanced': 4583367,
           'needTime': 416634,
           'needYield': 0,
           'saveState': 5000,
           'restoreState': 5000,
           'isEOF': 1,
           'direction': 'forward',
           'docsExamined': 5000000},
          'allPlansExecution': []}
```

Experiment 8
Nested Query with Aggregation

- Extract all sales having total revenue greater than the average revenue\
- SQL took 3.9 seconds



Experiment 8
Nested Query with Aggregation

ongoDB took 5.1 seconds

```
In [41]: collection.find({'$expr': {'$gt': ['$total_revenue', avg_]}}).explain()['executionStats']

Out[41]: {'executionSuccess': True,
          'nReturned': 1696729,
          'executionTimeMillis': 5113,
          'totalKeysExamined': 0,
          'totalDocsExamined': 5000000,
          'executionStages': {'stage': 'COLLSCAN',
           'filter': {'$and': [{'$expr': {'$gt': ['$total_revenue',
                {'$const': 1331058.049277108}]}},
             {'total_revenue': {'$_internalExprGt': 1331058.049277108}}]},
           'nReturned': 1696729,
           'executionTimeMillisEstimate': 103,
           'works': 5000002,
           'advanced': 1696729,
           'needTime': 3303272,
           'needYield': 0,
           'saveState': 5000,
           'restoreState': 5000,
           'isEOF': 1,
           'direction': 'forward',
           'docsExamined': 5000000},
          'allPlansExecution': []}
```

SQL VS MongoDB

- SQL databases are used to store structured data while NoSQL databases like MongoDB are used to save unstructured data in json format. In MySQL, the data is stored in tables. In MongoDB, data is stored in collections.
- SQL databases, have a predefined schema to which the data should comply. On the other hand, in MongoDB, there is no need to predefine any schema, it can store different types of documents.
- MongoDB does not support advanced analytics and joins like SQL databases support.
- The architecture of SQL databases like MySQL is governed by the principles of ACID property.These properties focus on the consistency and reliability of the transaction done in the database. MongoDB is built on the principles of CAP Theorem which focuses on availability of data

SQL VS mongoDB

- MySQL database or the SQL databases, in general, can be scaled only vertically by increasing memory size, disk space or computing power of the server. Vertical scaling can be expensive with costs growing rapidly for large databases with high query volume.
- NoSQL databases like MongoDB support horizontal scaling, also known as sharding. In this case, instead of increasing the server configuration a new server is added for the purpose of scalability. This approach is usually less expensive because a cluster of low-cost commodity hardware can together meet the requirements to support high query volume in a cost-effective manner

PostgreSQL VS MongoDB

- MongoDB vs PostgreSQL ScalabilIty : MongoDB comes with scaling built-in, however PostgreSQL requires an extension to do so. To achieve scalability with PostgreSQL, there are several extensions to choose from. In a sharded cluster, MongoDB allows you to have as many nodes as you need, while PostgreSQL has no database capacity limitations.

- MongoDB vs PostgreSQL Performance : For data warehousing and data analytical workloads, PostgreSQL outperforms MongoDB, according to many benchmarks. However, there are benchmarks that suggest that both PostgreSQL and MongoDB have an advantage when it comes to JSON operations. MongoDB is well-suited to web applications with enormous data stores that must be retained for years.

PostgreSQL VS MongoDB

- MongoDB Vs. PostgreSQL Security : Both MongoDB and PostgreSQL allow role-based access management as well as LDAP and Kerberos authentication. Unfortunately, MongoDB’s enterprise features include LDAP and Kerberos. Both solutions can encrypt communications using TLS, which is an industry standard. With PostgreSQL, there are a few choices for encryption at rest, but encryption at rest is a MongoDB enterprise capability.

Conclusion

- SQL and MongoDB perform differently based on the queries and datasets
- The environment may affect the performance of both SQL and MongoDB
- In some cases SQL performs better while in some cases MongoDB performs better
- It is hard to say that one is better than other

References

[1] Seyyed Hamid Aboutorabi[a], Mehdi Rezapour, Milad Moradi, and Nasser Ghadiri. Performance evaluation of sql and mongodb databases for big ecommerce data. In 2015 International Symposium on Computer Science and Software Engineering (CSSE), pages 1–7. IEEE, 2015.

[2] Irina Astrova, Arne Koschel, Chris Eickemeyer, Jan Kersten, and Norman Offel. Dbaas comparison: Amazon vs. microsoft. In 2017 International Conference on Information Society (i-Society), pages 15–21. IEEE, 2017.

[3] Yishan Li and Sathiamoorthy Manoharan. A performance comparison of sql and nosql databases. In 2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM), pages 15–19, 2013.

[4] Daniel Pereira, Paulo Oliveira, and F´atima Rodrigues. Data warehouses in mongodb vs sql server: A comparative analysis of the querie performance. In 2015 10th Iberian Conference on Information Systems and Technologies (CISTI), pages 1–7. IEEE, 2015.

[5] Ricardo S´anchez-de Madariaga, Adolfo Mu˜noz, Raimundo Lozano-Rub´ı, Pablo Serrano-Balazote, Antonio L Castro, Oscar Moreno, and Mario Pascual. Examining database persistence of iso/en 13606 standardized electronic health record extracts: relational vs. nosql approaches. BMC medical informatics and decision making, 17(1):1–14, 2017.

Thank you!