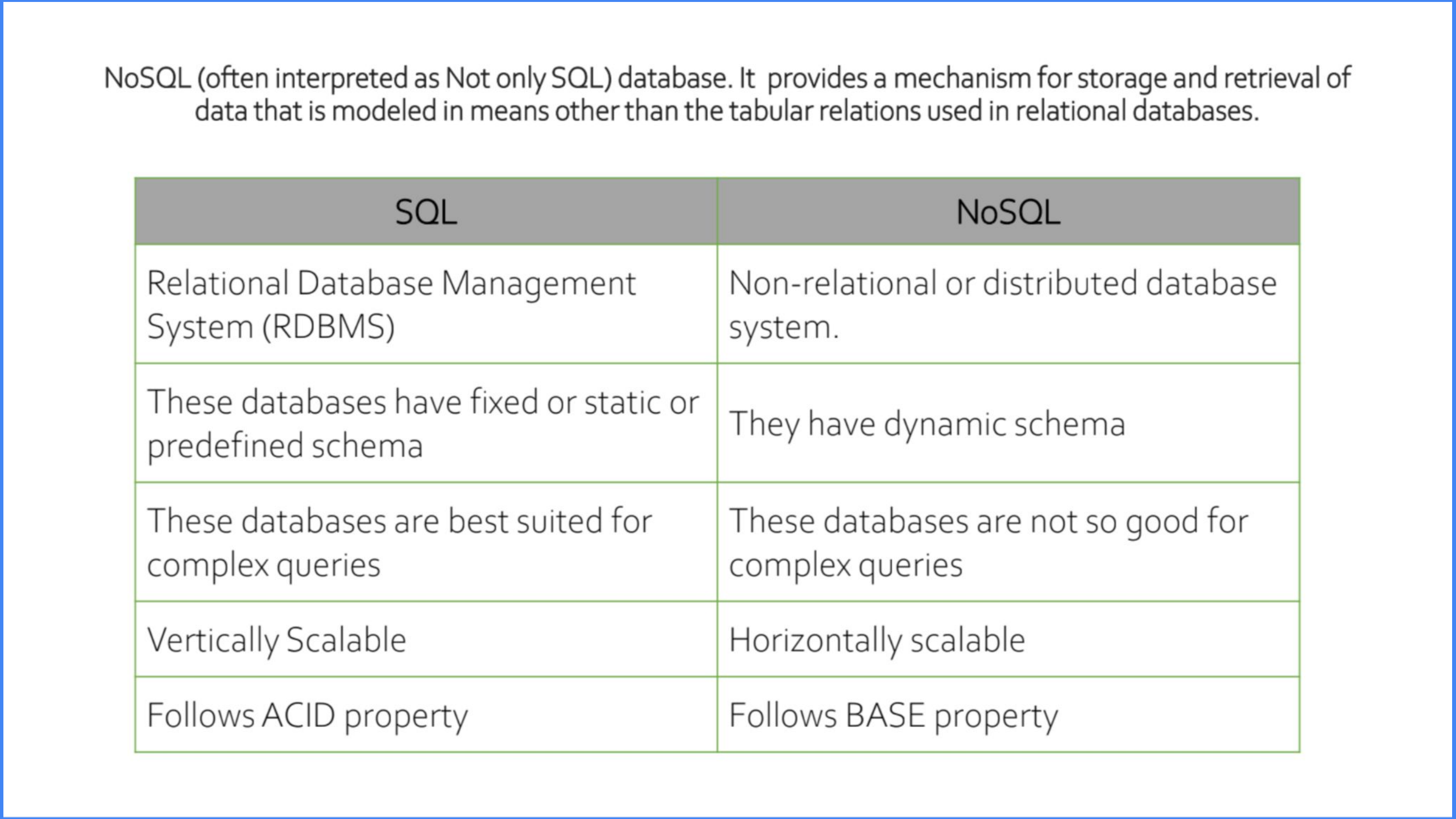
NoSQL (Lab - 3)

Tejas Cavale - Teaching Assistant , SSD M25CS6.302:

Recap - NoSQL



NoSQL (often interpreted as Not only SQL) database. It provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases.

SQL	NoSQL
Relational Database Management System (RDBMS)	Non-relational or distributed database system.
These databases have fixed or static or predefined schema	They have dynamic schema
These databases are best suited for complex queries	These databases are not so good for complex queries
Vertically Scalable	Horizontally scalable
Follows ACID property	Follows BASE property

Recap - MongoDB

- Easy to use
  - Document-oriented (more flexible than RDBMS rows)
  - Supports complex hierarchical data in a single record
  - No predefined schema ➔ quick add/remove fields
- Designed to scale out
  - Scale up = bigger server (costly)
  - Scale out = add servers (cheaper, MongoDB supports via sharding & load balancing)
- Rich features
  - CRUD operations (Insert, Update, Delete, Select)
  - Indexing, Aggregation, Custom collections/index types, File storage
- High performance
  - Scalable, flexible, and fast by design

Recap - MongoDB

- Stores data as **documents** inside **collections** inside **databases**

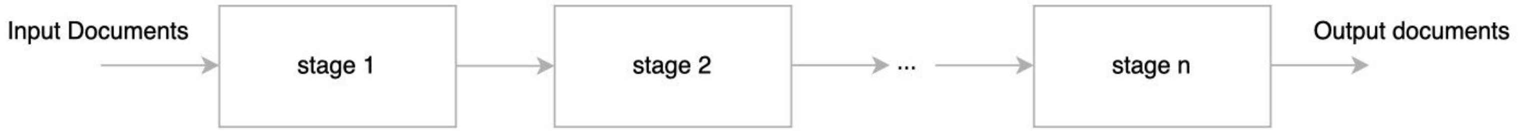
- JSON → JavaScript Object Notation
  - Human-readable
  - { "first_name": "John", "age": 22, "skills": ["Programming"] }
- BSON → Binary JSON
  - Efficient, faster to parse, supports more data types
  - MongoDB stores data internally as BSON

Recap - MongoDB CRUD

- Create → insertOne() | insertMany()
- Read → findOne() | find()
- Update → updateOne() | updateMany()
- Delete → deleteOne() | deleteMany()

MongoDB - Aggregation

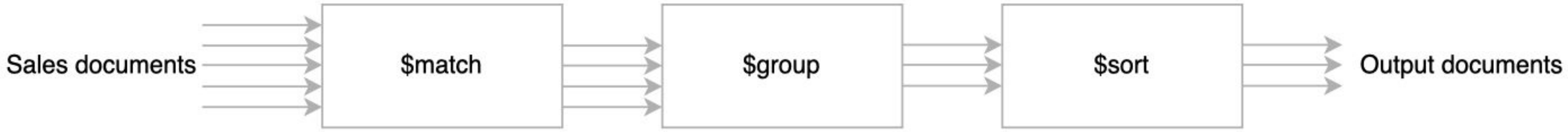
- Aggregation operations process multiple documents and return computed results
- Common use: group documents by field values and calculate summaries
  - Example: Total sales per product from sales orders
- Aggregations are performed using aggregation pipelines
- Pipeline stages:
  - Each stage processes input documents
  - Output of one stage is passed to the next stage
  - Final stage produces the aggregated result



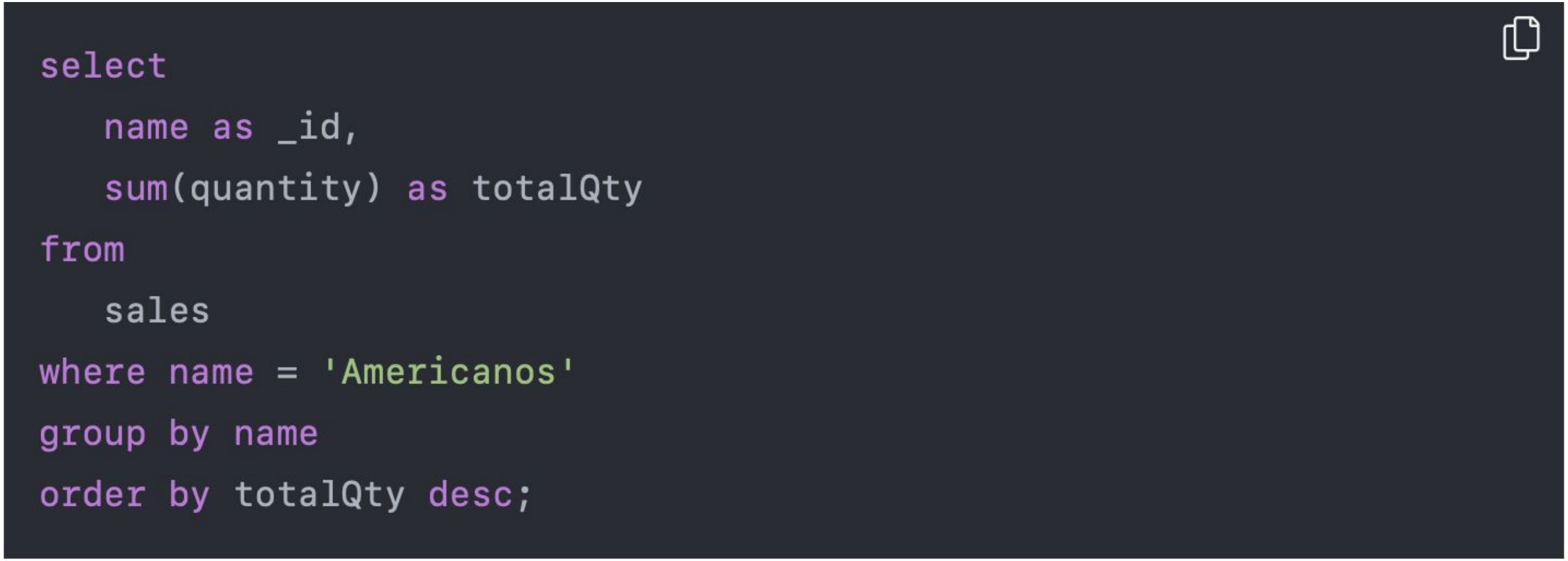
MongoDB - Aggregation Operations

- $project – select fields for the output documents.
- $match – select documents to be processed.
- $limit – limit the number of documents to be passed to the next stage.
- $sort – sort documents.
- $group – group documents by a specified key.
- $count – count the number of documents passing through the pipeline and return the total.
- $lookup – perform a join with another collection and add matching documents as an array field.

```
db.sales.aggregate([
        {
                $match: { item: "Americanos" }
        },
        {
                $group: {
                        _id: "$size",
                        totalQty: {$sum: "$quantity"}
                }
        },
        {
                $sort: { totalQty : -1}
        }
]);
```

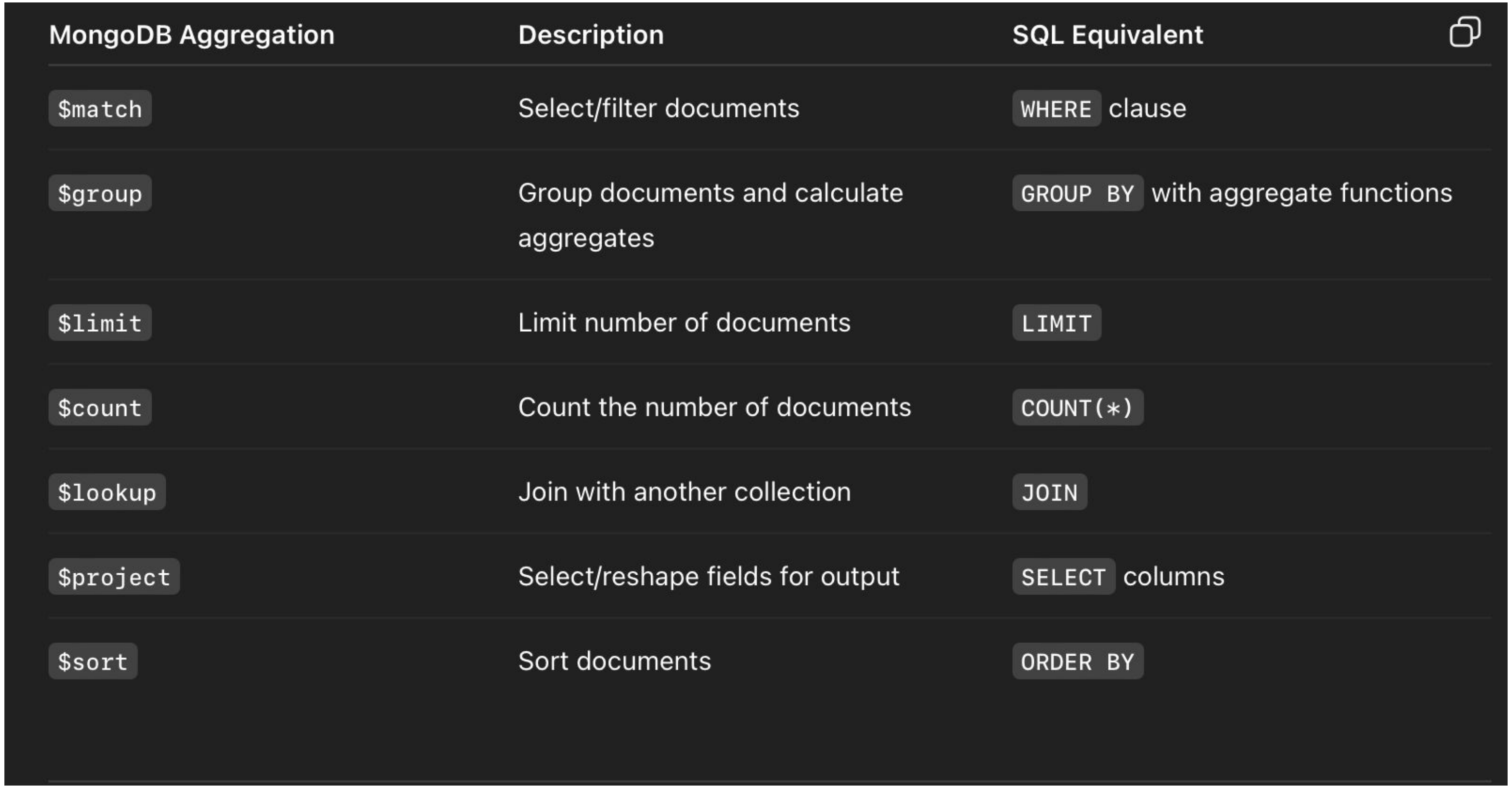


Equivalent SQL Query



```sql
select
   name as _id,
   sum(quantity) as totalQty
from
   sales
where name = 'Americanos'
group by name
order by totalQty desc;
```

Comparison with SQL



MongoDB Aggregation	Description	SQL Equivalent
`$match`	Select/filter documents	`WHERE` clause
`$group`	Group documents and calculate aggregates	`GROUP BY` with aggregate functions
`$limit`	Limit number of documents	`LIMIT`
`$count`	Count the number of documents	`COUNT(*)`
`$lookup`	Join with another collection	`JOIN`
`$project`	Select/reshape fields for output	`SELECT` columns
`$sort`	Sort documents	`ORDER BY`

Thank You!

(Refer Official Documentation for any doubts in syntax / usage)