| **SQL Terms, Functions, and Concepts** | **MongoDB Aggregation Operators** |
| --- | --- |
| WHERE | [$match](https://docs.mongodb.com/manual/reference/operator/aggregation/match/#pipe._S_match) |
| GROUP BY | [$group](https://docs.mongodb.com/manual/reference/operator/aggregation/group/#pipe._S_group) |
| HAVING | [$match](https://docs.mongodb.com/manual/reference/operator/aggregation/match/#pipe._S_match) |
| SELECT | [$project](https://docs.mongodb.com/manual/reference/operator/aggregation/project/#pipe._S_project) |
| ORDER BY | [$sort](https://docs.mongodb.com/manual/reference/operator/aggregation/sort/#pipe._S_sort) |
| LIMIT | [$limit](https://docs.mongodb.com/manual/reference/operator/aggregation/limit/#pipe._S_limit) |
| SUM() | [$sum](https://docs.mongodb.com/manual/reference/operator/aggregation/sum/#grp._S_sum) |
| COUNT() | [$sum](https://docs.mongodb.com/manual/reference/operator/aggregation/sum/#grp._S_sum) |
| join | [$lookup](https://docs.mongodb.com/manual/reference/operator/aggregation/lookup/#pipe._S_lookup)  *New in version 3.2.* |

Examples

The following table presents a quick reference of SQL aggregation statements and the corresponding MongoDB statements. The examples in the table assume the following conditions:

* The SQL examples assume *two* tables, orders and order\_lineitem that join by the order\_lineitem.order\_id and the orders.id columns.
* The MongoDB examples assume *one* collection orders that contain documents of the following prototype:
* {
* cust\_id: "abc123",
* ord\_date: ISODate("2012-11-02T17:04:11.102Z"),
* status: 'A',
* price: 50,
* items: [ { sku: "xxx", qty: 25, price: 1 },
* { sku: "yyy", qty: 25, price: 1 } ]
* }

| **SQL Example** | **MongoDB Example** | **Description** |
| --- | --- | --- |
| **SELECT** **COUNT**(\*) **AS** **count**  **FROM** orders | db.orders.aggregate( [  {  $group: {  \_id: **null**,  count: { $sum: 1 }  }  }  ] ) | Count all records from orders |
| **SELECT** **SUM**(price) **AS** total  **FROM** orders | db.orders.aggregate( [  {  $group: {  \_id: **null**,  total: { $sum: "$price" }  }  }  ] ) | Sum the price field from orders |
| **SELECT** cust\_id,  **SUM**(price) **AS** total  **FROM** orders  **GROUP** **BY** cust\_id | db.orders.aggregate( [  {  $group: {  \_id: "$cust\_id",  total: { $sum: "$price" }  }  }  ] ) | For each unique cust\_id, sum the price field. |
| **SELECT** cust\_id,  **SUM**(price) **AS** total  **FROM** orders  **GROUP** **BY** cust\_id  **ORDER** **BY** total | db.orders.aggregate( [  {  $group: {  \_id: "$cust\_id",  total: { $sum: "$price" }  }  },  { $sort: { total: 1 } }  ] ) | For each unique cust\_id, sum the price field, results sorted by sum. |
| **SELECT** cust\_id,  ord\_date,  **SUM**(price) **AS** total  **FROM** orders  **GROUP** **BY** cust\_id,  ord\_date | db.orders.aggregate( [  {  $group: {  \_id: {  cust\_id: "$cust\_id",  ord\_date: {  month: { $month: "$ord\_date" },  day: { $dayOfMonth: "$ord\_date" },  year: { $year: "$ord\_date"}  }  },  total: { $sum: "$price" }  }  }  ] ) | For each uniquecust\_id, ord\_dategrouping, sum the pricefield. Excludes the time portion of the date. |
| **SELECT** cust\_id,  **count**(\*)  **FROM** orders  **GROUP** **BY** cust\_id  **HAVING** **count**(\*) > 1 | db.orders.aggregate( [  {  $group: {  \_id: "$cust\_id",  count: { $sum: 1 }  }  },  { $match: { count: { $gt: 1 } } }  ] ) | For cust\_idwith multiple records, return the cust\_idand the corresponding record count. |
| **SELECT** cust\_id,  ord\_date,  **SUM**(price) **AS** total  **FROM** orders  **GROUP** **BY** cust\_id,  ord\_date  **HAVING** total > 250 | db.orders.aggregate( [  {  $group: {  \_id: {  cust\_id: "$cust\_id",  ord\_date: {  month: { $month: "$ord\_date" },  day: { $dayOfMonth: "$ord\_date" },  year: { $year: "$ord\_date"}  }  },  total: { $sum: "$price" }  }  },  { $match: { total: { $gt: 250 } } }  ] ) | For each unique cust\_id, ord\_dategrouping, sum the pricefield and return only where the sum is greater than 250. Excludes the time portion of the date. |
| **SELECT** cust\_id,  **SUM**(price) **as** total  **FROM** orders  **WHERE** status = 'A'  **GROUP** **BY** cust\_id | db.orders.aggregate( [  { $match: { status: 'A' } },  {  $group: {  \_id: "$cust\_id",  total: { $sum: "$price" }  }  }  ] ) | For each unique cust\_id with status A, sum the pricefield. |
| **SELECT** cust\_id,  **SUM**(price) **as** total  **FROM** orders  **WHERE** status = 'A'  **GROUP** **BY** cust\_id  **HAVING** total > 250 | db.orders.aggregate( [  { $match: { status: 'A' } },  {  $group: {  \_id: "$cust\_id",  total: { $sum: "$price" }  }  },  { $match: { total: { $gt: 250 } } }  ] ) | For each unique cust\_id with status A, sum the pricefield and return only where the sum is greater than 250. |
| **SELECT** cust\_id,  **SUM**(li.qty) **as** qty  **FROM** orders o,  order\_lineitem li  **WHERE** li.order\_id = o.id  **GROUP** **BY** cust\_id | db.orders.aggregate( [  { $unwind: "$items" },  {  $group: {  \_id: "$cust\_id",  qty: { $sum: "$items.qty" }  }  }  ] ) | For each unique cust\_id, sum the corresponding line item qtyfields associated with the orders. |
| **SELECT** **COUNT**(\*)  **FROM** (**SELECT** cust\_id,  ord\_date  **FROM** orders  **GROUP** **BY** cust\_id,  ord\_date)  **as** DerivedTable | db.orders.aggregate( [  {  $group: {  \_id: {  cust\_id: "$cust\_id",  ord\_date: {  month: { $month: "$ord\_date" },  day: { $dayOfMonth: "$ord\_date" },  year: { $year: "$ord\_date"}  }  }  }  },  {  $group: {  \_id: **null**,  count: { $sum: 1 }  }  }  ] ) | Count the number of distinctcust\_id, ord\_dategroupings. Excludes the time portion of the date. |

Additional Resources

* [MongoDB and MySQL Compared](https://www.mongodb.com/scale/mongodb-mysql-compared)