Sure, here's a cheat sheet that provides SQL equivalent queries for MongoDB:

SQL Command	MongoDB Command	Description
SELECT * FROM collection	db.collection.find()	Retrieve all documents from a collection
SELECT * FROM collection WHERE condition	db.collection.find({ field: value })	Retrieve documents that match a condition
SELECT field1, field2 FROM collection	db.collection.find({}, { field1: 1, field2: 1 })	Retrieve specific fields from documents
SELECT * FROM collection LIMIT n	db.collection.find().limit(n)	Limit the number of documents returned
SELECT * FROM collection WHERE condition LIMIT n	db.collection.find({ field: value }).limit(n)	Limit and filter documents
SELECT * FROM collection ORDER BY field	db.collection.find().sort({ field: 1 })	Sort documents in ascending order by a field
SELECT * FROM collection ORDER BY field DESC	db.collection.find().sort({ field: -1 })	Sort documents in descending order by a field
SELECT * FROM collection WHERE condition ORDER BY field LIMIT n	db.collection.find({ field1: value }).sort({ field2: 1 }).limit(n)	Filter, sort, and limit documents
INSERT INTO collection (field1, field2) VALUES (value1, value2)	db.collection.insertOne({ field1: value1, field2: value2 })	Insert a single document into a collection
UPDATE collection SET field = value WHERE condition	db.collection.updateOne({ field: value }, { $set: { newField: newValue } })	Update a single document that matches a condition
DELETE FROM collection WHERE condition	db.collection.deleteOne({ field: value })	Delete a single document that matches a condition

These are some basic SQL equivalents for MongoDB queries. However, it's important to note that MongoDB's query language is different from SQL, and MongoDB offers many powerful features that are not directly translatable to SQL. It's essential to understand MongoDB's query syntax and features to

effectively work with MongoDB databases. Additionally, consider using MongoDB's aggregation framework for more complex queries and data processing tasks.