



Microsoft Azure DocumentDB Query Cheat Sheet

This cheat sheet helps you quickly write DocumentDB queries by showing some common SQL queries used to retrieve data from two simple JSON documents.

Example Family JSON Documents		SQL Query	Sample Queries
<pre>{ "id": "AndersenFamily", "lastName": "Andersen", "parents": [{ "firstName": "Thomas" }, { "firstName": "Mary Kay" }], "children": [{ "firstName": "Henriette Thaulow", "gender": "female", "grade": 5, "pets": [{ "givenName": "Fluffy" }] }], "address": { "state": "WA", "county": "King", "city": "seattle" }, "creationDate": "2015-01-03T12:00Z", "isRegistered": true, "location": { "type": "Point", "coordinates": [31.9, -4.8] } } </pre>	1	<pre>-- Find families by ID SELECT * FROM Families f WHERE f.id = "AndersenFamily"</pre> <pre>[{ "id": "AndersenFamily", "lastName": "Andersen", ... }]</pre>	Comparison (range) operators <pre>SELECT * FROM Families.children[0] c WHERE c.grade >= 5</pre>
<pre>{ "id": "WakefieldFamily", "parents": [{ "familyName": "Wakefield", "givenName": "Robin" }, { "familyName": "Miller", "givenName": "Ben" }], "children": [{ "familyName": "Merriam", "givenName": "Jesse", "gender": "female", "grade": 1, "pets": [{ "givenName": "Goofy" }, { "givenName": "Shadow" }] }, { "familyName": "Miller", "givenName": "Lisa", "gender": "female", "grade": 8 }], "address": { "state": "NY", "county": "Manhattan", "city": "NY" }, "creationDate": "2015-07-20T12:00Z", "isRegistered": false }</pre>	2	<pre>-- Find families where City equals State and return Name and City SELECT {"Name":f.id, "City":f.address.city} AS Family FROM Families f WHERE f.address.city = f.address.state</pre> <pre>[{ "Family": { "Name": "WakefieldFamily", "City": "NY" } }]</pre>	Logical operators <pre>SELECT * FROM Families.children[0] c WHERE c.grade >= 5 AND c.isRegistered = true</pre>
		<pre>-- Get the child names using an intra-document JOIN SELECT c.givenName FROM Families f JOIN c IN f.children WHERE f.id = 'WakefieldFamily' ORDER BY f.address.city ASC</pre> <pre>[{"givenName": "Jesse"}, {"givenName": "Lisa"}]</pre>	ORDER BY keyword <pre>SELECT f.id, f.address.city FROM Families f ORDER BY f.address.city</pre>
		<pre>-- Register UDF for REGEX_MATCH with this code function (input, pattern) { return input.match(pattern) != null; } -- Use JavaScript SELECT udf.REGEX_MATCH(Families.address.city, ".*eattle")</pre> <pre>[{"\$1": true}, {"\$1": false}]</pre>	IN keyword <pre>SELECT * FROM Families WHERE Families.address.state IN ("NY", "WA", "CA", "PA", "OH", "OR", "MI", "WI")</pre>
			Ternary (?) and Coalesce (??) operators <pre>SELECT (c.grade < 5)? "elementary": ((c.grade < 9)? "junior": "high") AS gradeLevel FROM Families.children[0] c</pre>
			Escape/quoted accessor <pre>SELECT f["lastName"] FROM Families f WHERE f["id"] = "AndersenFamily"</pre>
			Object/Array Creation <pre>SELECT [f.address.city, f.address.state] FROM Families f</pre>
			Value keyword <pre>SELECT VALUE "Hello World"</pre>
			Intra-document JOINS <pre>SELECT f.id AS familyName, c.givenName AS childGivenName, c.firstName AS childFirstName, p.givenName AS petName FROM Families f JOIN c IN f.children JOIN p IN f.pets</pre>
			Parameterized SQL <pre>SELECT * FROM Families f WHERE f.lastName = @lastName AND f.address.state = @addressState</pre>
			String Built-in functions <pre>SELECT Families.id, Families.address.city FROM Families WHERE STARTSWITH(Families.id, "Wakefield")</pre>
			Array Built-in functions <pre>SELECT Families.id FROM Families WHERE ARRAY_CONTAINS(Families.parents, {givenName: "Robin", familyName: "Wakefield" })</pre>
			Math Built-in functions <pre>SELECT VALUE ABS(-4)</pre>
			Type Built-in functions <pre>SELECT IS_DEFINED(f.lastName), IS_NUMBER(4) FROM Families f</pre>
			BETWEEN keyword <pre>SELECT * FROM Families.children[0] c WHERE c.grade BETWEEN 1 AND 5</pre>
			TOP keyword <pre>SELECT TOP 100 * FROM Families f</pre>
			Geospatial functions <pre>SELECT * FROM Families f WHERE ST_Distance(f.location, {"type": "Point", "coordinates": [31.9, -4.8]}) < 30000</pre>