

# Microsoft Azure Application Insights SQL to Analytics language cheat sheet

Download this document at <https://aka.ms/sql-analytics>

	SQL Query	Analytics Query
Select data from table	<code>SELECT * FROM dependencies</code>	dependencies
	<code>SELECT name, resultCode FROM dependencies</code>	dependencies   <code>project</code> name, resultCode
	<code>SELECT TOP 100 * FROM dependencies</code>	dependencies   <code>take 100</code>
Null evaluation	<code>SELECT * FROM dependencies WHERE resultCode IS NOT NULL</code>	dependencies   <code>where isnotnull(resultCode)</code>
Comparison operators (date)	<code>SELECT * FROM dependencies WHERE timestamp &gt; getdate()-1</code>	dependencies   <code>where timestamp &gt; ago(1d)</code>
	<code>SELECT * FROM dependencies WHERE timestamp BETWEEN '2016-10-01' AND '2016-11-01'</code>	dependencies   <code>where timestamp &gt; datetime(2016-10-01) and timestamp &lt;= datetime(2016-11-01)</code>
Comparison Operators (string)	<code>SELECT * FROM dependencies WHERE type = "Azure blob"</code>	dependencies   <code>where type == "Azure blob"</code>
	<code>--substring SELECT * FROM dependencies WHERE type like "%blob%"</code>	<code>//substring dependencies   where type contains "blob"</code>
	<code>--wildcard SELECT * FROM dependencies WHERE type like "Azure%"</code>	dependencies   <code>where type startswith "Azure"</code>  dependencies   <code>where type matches regex "^Azure.*"</code>
Comparison (boolean)	<code>SELECT * FROM dependencies WHERE !(success)</code>	dependencies   <code>where success == "False"</code>
Distinct	<code>SELECT DISTINCT name, type FROM dependencies</code>	dependencies   <code>summarize by name, type</code>
Grouping, Aggregation	<code>SELECT name, AVG(duration) FROM dependencies GROUP BY name</code>	dependencies   <code>summarize avg(duration) by name</code>
Column aliases, Extending	<code>SELECT operation_Name as Name, AVG(duration) as AvgD FROM dependencies GROUP BY name</code>	dependencies   <code>summarize AvgD=avg(duration) by operation_Name   project Name=operation_Name, AvgD</code>
Ordering	<code>SELECT name, timestamp FROM dependencies ORDER BY timestamp asc</code>	dependencies   <code>project name, timestamp   order by timestamp asc nulls last</code>
Top n by measure	<code>SELECT TOP 100 name, COUNT(*) as Count FROM dependencies GROUP BY name ORDER BY Count desc</code>	dependencies   <code>summarize Count=count() by name   top 100 by Count desc</code>
Union	<code>SELECT * FROM dependencies UNION SELECT * FROM exceptions</code>	<code>union dependencies, exceptions</code>
	<code>SELECT * FROM dependencies WHERE timestamp&gt;.. UNION SELECT * FROM exceptions WHERE timestamp&gt;..</code>	dependencies   <code>where timestamp &gt; ago(1d)</code>   <code>union (exceptions   where timestamp &gt; ago(1d) )</code>
Join	<code>SELECT * FROM dependencies LEFT OUTER JOIN exception ON dependencies.operation_Id = exceptions.operation_Id</code>	dependencies   <code>join kind=leftouter (exceptions) on \$left.operation_Id == \$right.operation_Id</code>

These are just subset of the operators available. Please refer to <https://aka.ms/AIAnalyticsReference> for a complete reference.

Try Analytics yourself by instrumenting with Azure Application Insights, or in the Analytics demo environment: <https://aka.ms/AIAnalyticsDemo>!

## Application Insights Analytics – useful operators

Category	Relevant Analytics functions
Selection and Column aliases	project, project-away, extend
Temporary tables and constants	let scalar_alias_name = ...; let table_alias_name = () { ...   ...   ... };
Comparison and String Operators	startswith, !startswith has*, !has contains, !contains, containscs hasprefix, !hasprefix, hassuffix, !hassuffix in, !in matches regex ==, =~, !=, !~  *has is more performant than contains
Common string functions	strcat(), replace() tolower(*), toupper(*) substring(), strlen()  *for a more performant solution than converting case when comparing strings use: "aBc" =~ "abc"
Common math functions	sqrt(), abs() exp(), exp2(), exp10(), log(), log2(), log10() pow() gamma(), gammaln()
Parsing text	extract(), extractjson(), parse*, split()  *parse is more performant
Limiting output	take, limit, top, sample  hash
Date functions	now(), ago() datetime(), datepart(), timespan startofday(), startofweek(), startofmonth(), startofyear() endofday(), endofweek(), endofmonth(), endofyear() dayofweek(), dayofmonth(), dayofyear() getmonth(), getyear(), weekofyear(), monthofyear()
Grouping and aggregation  by top, count(), min(), max(), bin()	summarize by max(), min(), count(), dcount(), avg(), sum(), stddev() countif(), dcountif() argmax(), argmin() percentiles(), percentile_array()  top, top-nested
Joins and Unions	join kind=leftouter, inner, rightouter, fullouter, leftanti  union
Sort, order	sort, order
Dynamic object (JSON and array) operators and functions	parsejson() makeset(), makelist() split(), arraylength() zip(), pack()
Logical operators	iff(condition, value_t, value_f)  binary_and(), binary_or(), binary_not(), binary_xor()
Machine learning	evaluate  autocluster, basket, diffpatterns, extractcolumns

More info about these and other functions and operators is available on our language reference: <https://aka.ms/AIAnalyticsReference>