Leveraging Time-Based Partitioning

All imported data is automatically partitioned into hourly buckets, based on the **'time'** field within each data record.

By specifying the time range in your query, you can avoid reading unnecessary data and can thus speed up your query significantly.

Specify Time as an Integer

When the 'time' field is specified within a WHERE clause, the query parser automatically detects which partitions should be processed. This auto-detection does **not** work if you specify the time with `float` instead of `int`.

```
[GOOD]: SELECT field1, field2, field3 FROM tbl WHERE time > 1349393020
[GOOD]: SELECT field1, field2, field3 FROM tbl WHERE time > 1349393020 + 3600
[GOOD]: SELECT field1, field2, field3 FROM tbl WHERE time > 1349393020 - 3600
[BAD]:  SELECT field1, field2, field3 FROM tbl WHERE time > 13493930200 / 10
[BAD]:  SELECT field1, field2, field3 FROM tbl WHERE time > 1349393020.00
[BAD]:  SELECT field1, field2, field3 FROM tbl WHERE time BETWEEN 1349392000 AND 1349394000
```

Use TD_TIME_RANGE

You can use TD_TIME_RANGE to partition data.

```
[GOOD]: SELECT ... WHERE TD_TIME_RANGE(time, '2013-01-01 PDT')
[GOOD]: SELECT ... WHERE TD_TIME_RANGE(time, '2013-01-01','PDT', NULL)
[GOOD]: SELECT ... WHERE TD_TIME_RANGE(time, '2013-01-01',
                                    TD_TIME_ADD('2013-01-01', '1day', 'PDT'))
```

However, if you use division in TD_TIME_RANGE, the time partition optimization doesn't work. For instance, the following conditions disable optimization.

Bad SQL

```
SELECT ... WHERE TD_TIME_RANGE(time, TD_SCHEDULED_TIME() / 86400 * 86400))
```

```
SELECT ... WHERE TD_TIME_RANGE(time, 1356998401 / 86400 * 86400))
```

The TD_INTERVAL user defined function is also available for partitioning. TD_INTERVAL provides an intuitive way to specify the time range. For example, to select the last 7 days:

```
SELECT ... WHERE TD_INTERVAL(time, '-7d')
```