In [2]:
```
#import pandas
import pandas as pd
```

In [3]:
```
#import csv
df = pd.read_csv('Boonsong Lekagul waterways readings.csv')
```

In [4]:
```
df
```

Out[4]:

	id	value	location	sample date	measure
0	2221	2.00	Boonsri	11-Jan-98	Water temperature
1	2223	9.10	Boonsri	11-Jan-98	Dissolved oxygen
2	2227	0.33	Boonsri	11-Jan-98	Ammonium
3	2228	0.01	Boonsri	11-Jan-98	Nitrites
4	2229	1.47	Boonsri	11-Jan-98	Nitrates
...	...	...	...	...	...
136819	3295800	5.20	Chai	27-Dec-16	Water temperature
136820	3295802	5.20	Chai	28-Dec-16	Water temperature
136821	3295804	5.00	Chai	29-Dec-16	Water temperature
136822	3295806	4.60	Chai	30-Dec-16	Water temperature
136823	3295808	4.00	Chai	31-Dec-16	Water temperature

136824 rows × 5 columns

In [5]:
```
#check null values
df.isnull().any()
```

Out[5]:
```
id             False
value          False
location       False
sample date    False
measure        False
dtype: bool
```

In [6]:
```
#drop duplicates
df.duplicated()
```

Out[6]:
```
0         False
1         False
2         False
3         False
4         False
          ...
136819    False
136820    False
136821    False
136822    False
136823    False
Length: 136824, dtype: bool
```

In [7]:
```
df.dtypes
```

Out[7]:
```
id              int64
value         float64
location       object
sample date    object
measure        object
dtype: object
```

In [8]:
```
from IPython.display import Image, Markdown, display
import altair as alt
from altair import datum
alt.data_transformers.enable("vegafusion")
```

Out[8]:
```
DataTransformerRegistry.enable('vegafusion')
```

In [9]:
```
#change dtype of sample date
df["Sample Date"] = pd.to_datetime(df["sample date"],format="mixed")
```

In [10]:
```
#remove sample date
df.drop(['sample date'], axis=1, inplace = True)
```

In [11]:
```
df.head()
```

Out[11]:

	id	value	location	measure	Sample Date
0	2221	2.00	Boonsri	Water temperature	1998-01-11
1	2223	9.10	Boonsri	Dissolved oxygen	1998-01-11
2	2227	0.33	Boonsri	Ammonium	1998-01-11
3	2228	0.01	Boonsri	Nitrites	1998-01-11
4	2229	1.47	Boonsri	Nitrates	1998-01-11

In [12]:
```
#extract day,month and year from Sample Date
df['day'] = pd.DatetimeIndex(df['Sample Date']).day
df['month'] = pd.DatetimeIndex(df['Sample Date']).strftime('%b')
df['year'] = pd.DatetimeIndex(df['Sample Date']).year
```

In [13]:
```
df.head()
```

Out[13]:

	id	value	location	measure	Sample Date	day	month	year
0	2221	2.00	Boonsri	Water temperature	1998-01-11	11	Jan	1998
1	2223	9.10	Boonsri	Dissolved oxygen	1998-01-11	11	Jan	1998
2	2227	0.33	Boonsri	Ammonium	1998-01-11	11	Jan	1998
3	2228	0.01	Boonsri	Nitrites	1998-01-11	11	Jan	1998
4	2229	1.47	Boonsri	Nitrates	1998-01-11	11	Jan	1998

In [14]:
```
#find unique locations
locations=df.location.unique().tolist()
```

```
locations.sort()
locations
```

Out[14]:
```
['Achara',
 'Boonsri',
 'Busarakhan',
 'Chai',
 'Decha',
 'Kannika',
 'Kohsoom',
 'Sakda',
 'Somchair',
 'Tansanee']
```

In [15]:
```
#find unique measures
df.measure.unique().tolist()
```

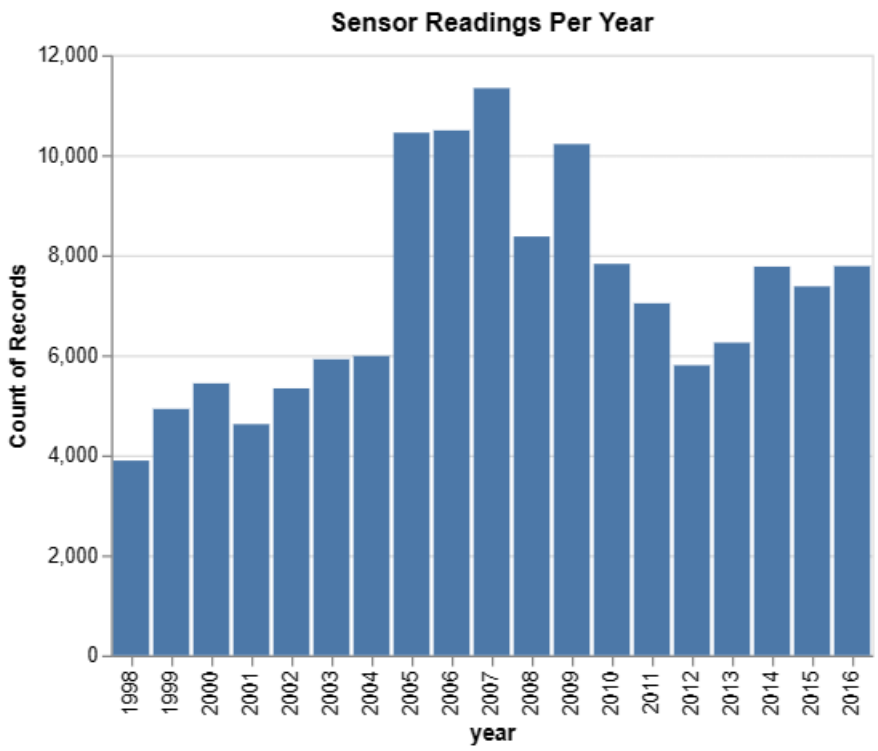
Out[15]:
```
['Water temperature',
 'Dissolved oxygen',
 'Ammonium',
 'Nitrites',
 'Nitrates',
 'Orthophosphate-phosphorus',
 'Total phosphorus',
 'Sodium',
 'Potassium',
 'Calcium',
 'Magnesium',
 'Chlorides',
 'Sulphates',
 'Iron',
 'Manganese',
 'Zinc',
 'Copper',
 'Chromium',
 'Lead',
 'Cadmium',
 'Mercury',
 'Nickel',
 'Arsenic',
 'Biochemical Oxygen',
 'Chemical Oxygen Demand (Cr)',
 'Chemical Oxygen Demand (Mn)',
 'AOX',
 'Atrazine',
 'Cesium',
 'Macrozoobenthos',
 'Total coliforms',
 'Fecal coliforms',
 'p,p-DDT',
 'gamma-Hexachlorocyclohexane',
 'Bicarbonates',
 'Anionic active surfactants',
 'Total extractable matter',
 'Fecal streptococci ',
 'Petroleum hydrocarbons',
 'PAHs',
 'Benzo(a)pyrene',
 'Benzo(g,h,i)perylene',
 'Benzo(b)fluoranthene',
 'Benzo(k)fluoranthene',
 'Fluoranthene',
 'Indeno(1,2,3-c,d)pyrene',
 'PCB 28',
 'PCB 52',
 'PCB 101',
 'PCB 138',
 'PCB 153',
 'PCB 180',
 'Silica (SiO2)',
 'Oxygen saturation',
 'Total hardness',
 'Total dissolved salts',
 'Heptachloroepoxide',
 'Heptachlor',
 'Endosulfan (alpha)',
 'Endosulfan (beta)',
```

```
 'p,p-DDD',
 'p,p-DDE',
 'alpha-Hexachlorocyclohexane',
 'beta-Hexaxchlorocyclohexane',
 'Aldrin',
 'Dieldrin',
 'Endrin',
 'Methoxychlor',
 'Simazine',
 'Metolachlor',
 'Alachlor',
 'Carbonates',
 'Total nitrogen',
 'Tetrachloromethane',
 'Barium',
 'Cyanides',
 'Sulfides',
 'Selenium',
 'Total organic carbon',
 '1,2,4-Trichlorobenzene',
 '1,2,3-Trichlorobenzene',
 'Pentachlorobenzene',
 'Acenaphthene',
 'Acenaphthylene',
 'Anthracene',
 'Benzo(a)anthracene',
 'Chrysene',
 'Naphthalene',
 'Phenanthrene',
 'Pyrene',
 'Hexachlorobenzene',
 'Isodrin',
 'Aluminium',
 'Dissolved organic carbon',
 'Dissolved silicates',
 'Organic nitrogen',
 'Fluorene',
 'PCB 118',
 'Trifluralin',
 'Inorganic nitrogen',
 'Berilium',
 'Boron',
 'AGOC-3A',
 'Methylosmoline',
 'Chlorodinine',
 'Total dissolved phosphorus']
```

In [16]:
```
year = df.year.unique().tolist()
year
month = df.month.unique().tolist()
```

In [17]:
```
# plot bar chart for sensor readings per year
sensor_reading_chart=alt.Chart(df, title='Sensor Readings Per Year').mark_bar().encode
    x='year:O',
    y='count(value)',
).interactive()
sensor_reading_chart
```

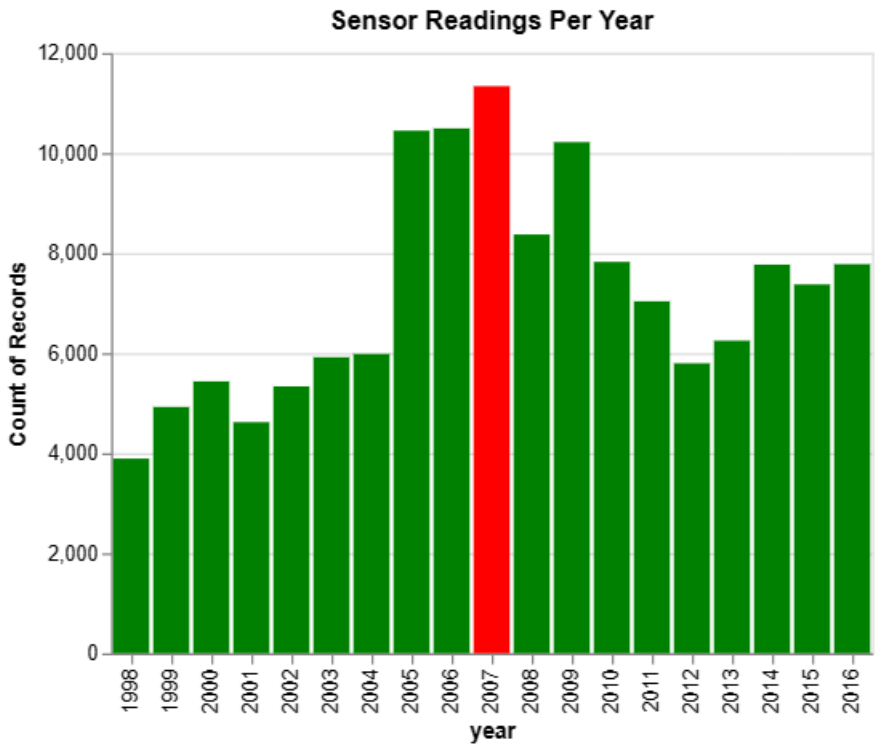
Out[17]:



In [18]:

```
#visualising the highest reading
sensor_reading_chart.encode(
    color = alt.condition(
        datum['year'] == 2007,
        alt.value('red'),
        alt.value('green')
    )
)
```

Out[18]:

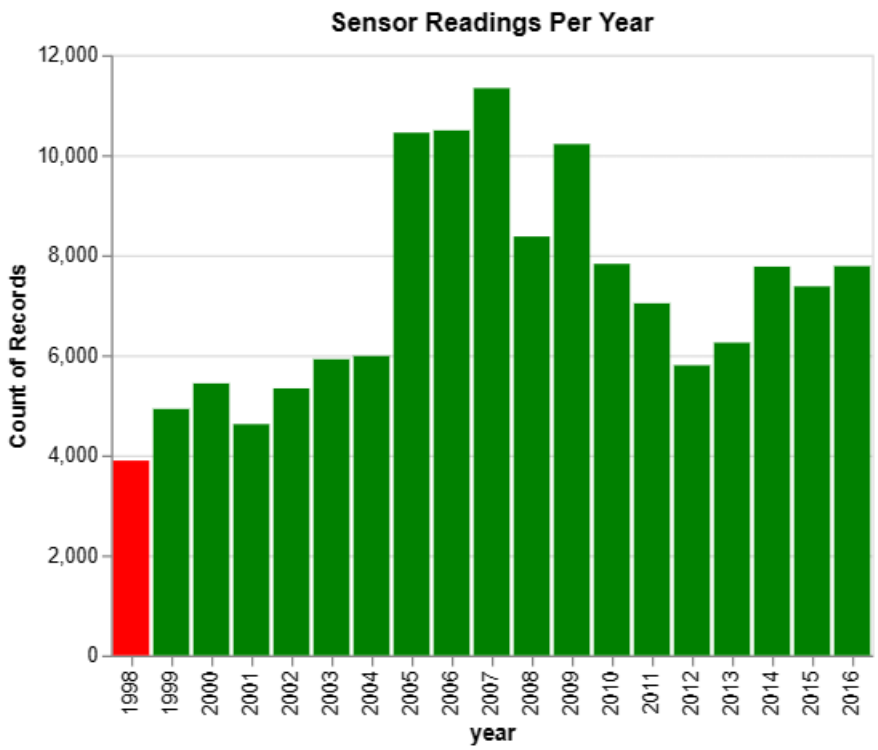


In [19]:

```
#visualising the lowest reading
sensor_reading_chart.encode(
```

```
    color = alt.condition(
        datum['year'] == 1998,
        alt.value('red'),
        alt.value('green')
    )
)
```

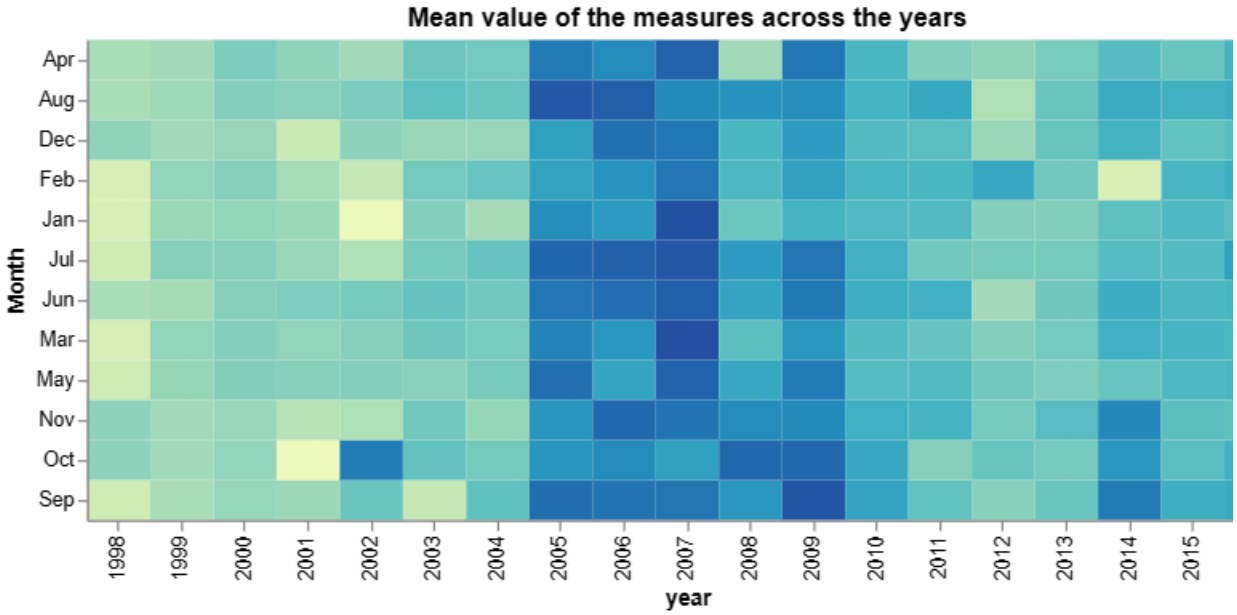
Out[19]:



In [20]:

```
#heatmap
alt.Chart(df).mark_rect().encode(
    x='year:O',
    y=alt.Y('month:N', axis=alt.Axis(title='Month'), scale=alt.Scale()),
    color=alt.Color(
        'count(value):Q', scale=alt.Scale(domain=(100, 1200))),
    tooltip=['year', 'month', 'count(value)']
).properties(
    width=600,
    title='Mean value of the measures across the years')
```

Out[20]:

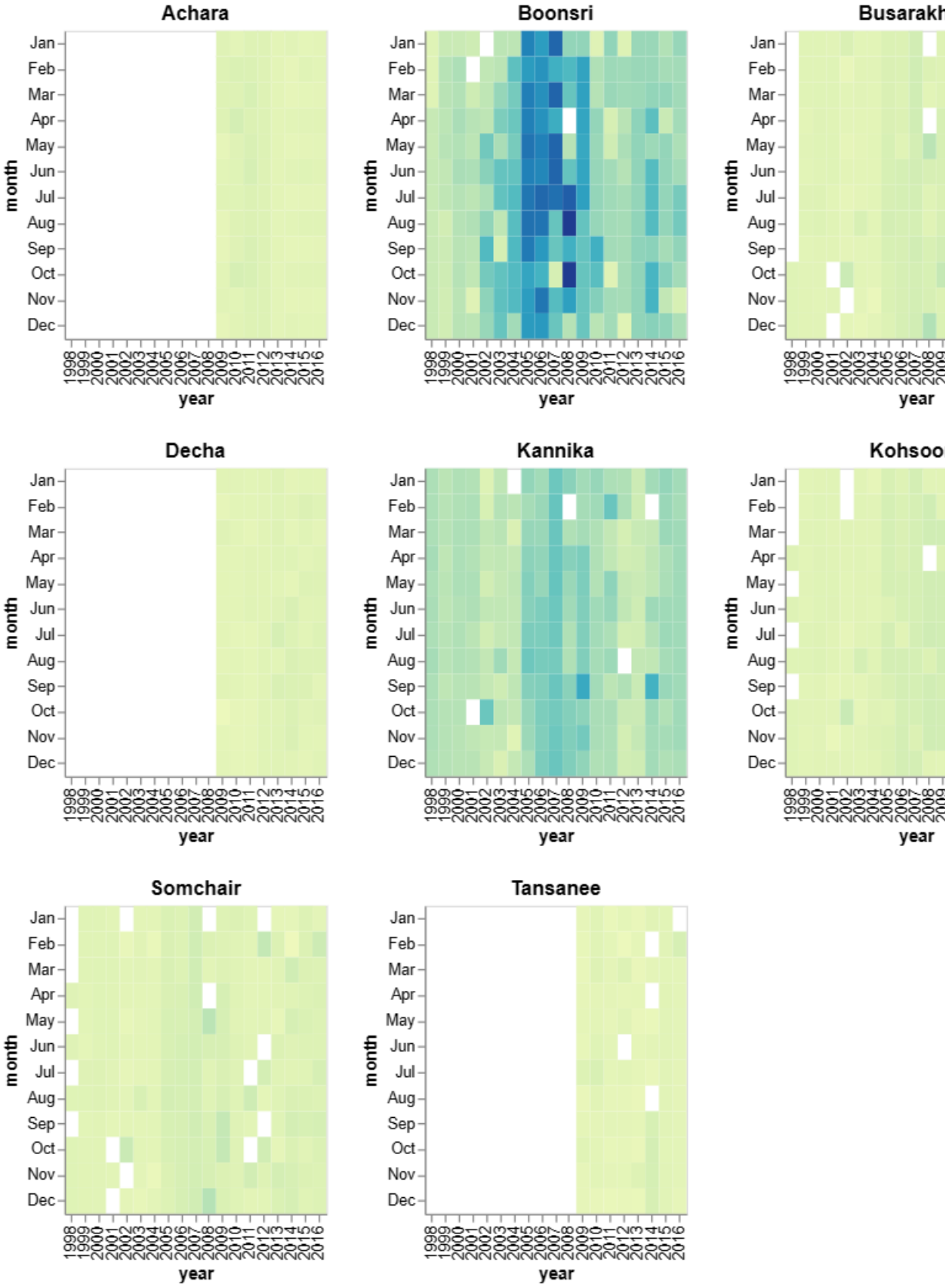


In [21]:

```
display(Markdown('### monthly count of values across all locations'))


# splitting the 10 locations into 5
ab_1 = alt.concat()
ab_2 = alt.concat()
ab_3 = alt.concat()

for index, location in enumerate(locations):
    chart = alt.Chart(df[df['location'] == location]).mark_rect().encode(
        x=alt.X('year:O', scale=alt.Scale(domain=year)),
        y=alt.Y('month:N', scale=alt.Scale(domain=month)),
        color=alt.Color(
            'count(value):Q', scale=alt.Scale( domain=(0, 400))),
        tooltip=['year', 'month', 'count(value)']
    ).properties(width=170, height=200, title=location)

    if index < 4:
        ab_1 |= chart
    elif 4 <= index <= 7:
        ab_2 |= chart
    else:
        ab_3 |= chart

# vertically concatenating all the rows
alt.vconcat(ab_1, ab_2, ab_3)
```

monthly count of values across all locations

Out[21]:



In [22]:

```
monthly_mean_value = df.groupby(
    ['measure', 'month', 'year']).agg({'value': ['mean']})
monthly_mean_value.columns = ['mean_value']
monthly_mean_value= monthly_mean_value.sort_values(
    by=['mean_value'], ascending=False)
```

```
monthly_mean_value = monthly_mean_value.reset_index()
monthly_mean_value
```

Out[22]:

	measure	month	year	mean_value
0	Iron	Aug	2003	8400.021654
1	Total coliforms	Oct	2010	3252.400000
2	Total coliforms	Jan	2009	1253.073846
3	Fecal coliforms	Apr	2011	790.000000
4	Total coliforms	Jul	2013	612.200000
...	...	...	...	...
9679	Cyanides	Jun	2011	0.000000
9680	Cyanides	Mar	2011	0.000000
9681	Cyanides	May	2011	0.000000
9682	Cyanides	Nov	2011	0.000000
9683	p,p-DDT	Sep	2016	0.000000

9684 rows × 4 columns

In [23]:

```
daily_mean_values = df.groupby(['measure', 'Sample Date']).agg({'value': ['mean']})
daily_mean_values.columns = ['mean_value']
daily_mean_values= daily_mean_values.sort_values(by=['mean_value'], ascending=False)
daily_mean_values = daily_mean_values.reset_index()
daily_mean_values
```

Out[23]:

	measure	Sample Date	mean_value
0	Iron	2003-08-15	27299.0
1	Total coliforms	2009-01-15	16090.0
2	Total coliforms	2010-10-20	13000.0
3	Total coliforms	2009-11-20	4600.0
4	Total coliforms	2009-08-20	3300.0
...	...	...	...
47305	Endosulfan (beta)	2005-01-28	0.0
47306	Endosulfan (beta)	2005-01-27	0.0
47307	Endosulfan (beta)	2005-01-22	0.0
47308	Endosulfan (beta)	2005-01-16	0.0
47309	p,p-DDT	2016-12-12	0.0

47310 rows × 3 columns

In [24]:

```
daily_mean_value = df.groupby(['measure', 'Sample Date']).agg({'value': ['mean']})
daily_mean_value.columns = ['mean_value']
```

```
daily_mean_value= daily_mean_value.sort_values(by=['mean_value'], ascending=False)
daily_mean_value = daily_mean_value.reset_index()
daily_mean_value
```

Out[24]:

	measure	Sample Date	mean_value
0	Iron	2003-08-15	27299.0
1	Total coliforms	2009-01-15	16090.0
2	Total coliforms	2010-10-20	13000.0
3	Total coliforms	2009-11-20	4600.0
4	Total coliforms	2009-08-20	3300.0
...	...	...	...
47305	Endosulfan (beta)	2005-01-28	0.0
47306	Endosulfan (beta)	2005-01-27	0.0
47307	Endosulfan (beta)	2005-01-22	0.0
47308	Endosulfan (beta)	2005-01-16	0.0
47309	p,p-DDT	2016-12-12	0.0

47310 rows × 3 columns

In [25]:
```
# List of all measures sorted in descending order of their daily mean values
sort_by_values = daily_mean_values['measure'].unique().tolist()
# Get the top 5 measures with the highest values
top_10_value = sort_by_values[0:10]
top_10_value
```

Out[25]:
```
['Iron',
 'Total coliforms',
 'Total dissolved salts',
 'Zinc',
 'Fecal coliforms',
 'Manganese',
 'Aluminium',
 'Total hardness',
 'Bicarbonates',
 'Chlorides']
```

In [26]:
```
df[df['measure'] == 'Iron']['value'].describe()
```

Out[26]:
```
count     2710.000000
mean        81.342855
std       1525.187169
min          0.000000
25%          0.250000
50%          0.480000
75%          0.871500
max      37959.280000
Name: value, dtype: float64
```

In [27]:
```
sort_by_count = df.measure.value_counts().index.tolist()
top_10_count = sort_by_count[0:10]
top_10_count
```

Out[27]:
```
['Water temperature',
 'Nitrites',
 'Ammonium',
 'Nitrates',
 'Orthophosphate-phosphorus',
 'Total phosphorus',
 'Dissolved oxygen',
 'Biochemical Oxygen',
 'Manganese',
 'Chlorides']
```

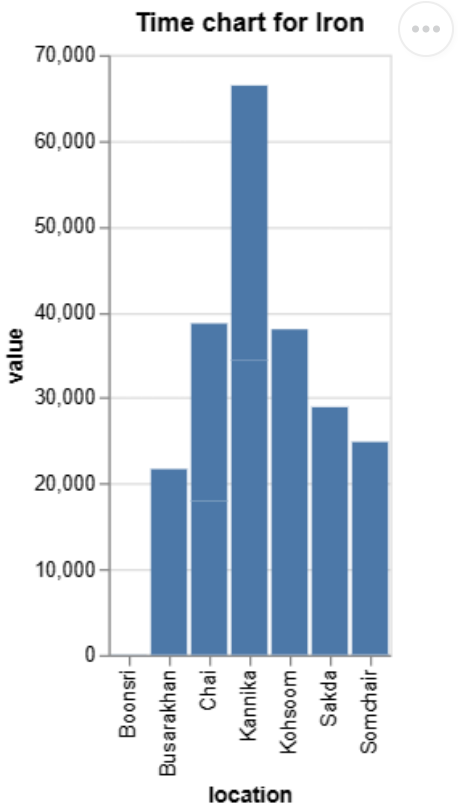
In [28]:
```
highest_iron=df[(df['measure'] == 'Iron') & (df['value'] >=8.71)]
highest_iron.sort_values(
    by=['value'], ascending=False)
```

Out[28]:

	id	value	location	measure	Sample Date	day	month	year
27876	298168	37959.28	Kohsoom	Iron	2003-08-15	15	Aug	2003
27986	303329	34413.96	Kannika	Iron	2003-08-15	15	Aug	2003
28008	304037	32002.20	Kannika	Iron	2003-08-15	15	Aug	2003
28028	305080	28901.59	Sakda	Iron	2003-08-15	15	Aug	2003
27942	300556	24790.53	Somchair	Iron	2003-08-15	15	Aug	2003
27964	300930	21665.84	Busarakhan	Iron	2003-08-15	15	Aug	2003
27920	300092	20688.60	Chai	Iron	2003-08-15	15	Aug	2003
27898	298744	17970.00	Chai	Iron	2003-08-15	15	Aug	2003
22887	227805	30.25	Boonsri	Iron	2002-10-15	15	Oct	2002
1516	6547	28.25	Somchair	Iron	1998-06-29	29	Jun	1998
22898	234893	24.25	Kohsoom	Iron	2002-10-15	15	Oct	2002
22931	239159	23.47	Busarakhan	Iron	2002-10-15	15	Oct	2002
22920	238440	20.87	Somchair	Iron	2002-10-15	15	Oct	2002
22942	242183	20.00	Kannika	Iron	2002-10-15	15	Oct	2002
22953	244276	20.00	Sakda	Iron	2002-10-15	15	Oct	2002
22909	235805	20.00	Chai	Iron	2002-10-15	15	Oct	2002
1383	4864	14.31	Kohsoom	Iron	1998-06-27	27	Jun	1998
888	6506	9.20	Somchair	Iron	1998-04-17	17	Apr	1998
15705	168299	8.73	Somchair	Iron	2001-04-29	29	Apr	2001

In [29]:
```
# iron chart
highest_iron_chart = alt.Chart(
    highest_iron, title='Time chart for Iron'
).mark_bar().encode(
    x='location',
    y='value',
)
highest_iron_chart
```

Out[29]:



In [30]:
```
df[df['measure'] == 'Water temperature']['value'].describe()
```

Out[30]:
```
count    5031.000000
mean       14.060153
std         8.162503
min         0.000000
25%         6.500000
50%        14.000000
75%        21.500000
max        36.400000
Name: value, dtype: float64
```

In [31]:
```
highest_water_temp=df[(df['measure'] == 'Water temperature') & (df['value'] >=25.0000)
highest_water_temp.sort_values(by=['value'], ascending=False)
```

Out[31]:

	id	value	location	measure	Sample Date	day	month	year
33620	510636	36.4	Kohsoom	Water temperature	2004-07-25	25	Jul	2004
133873	3291664	34.0	Tansanee	Water temperature	2016-08-12	12	Aug	2016
27459	300066	32.0	Chai	Water temperature	2003-07-10	10	Jul	2003
27415	298718	32.0	Chai	Water temperature	2003-07-10	10	Jul	2003
27437	299400	32.0	Chai	Water temperature	2003-07-10	10	Jul	2003
...	...	...	...	...	...	...	...	...
92027	1327027	25.0	Somchair	Water temperature	2010-08-28	28	Aug	2010
92059	1327623	25.0	Busarakhan	Water temperature	2010-08-28	28	Aug	2010
98358	1626307	25.0	Tansanee	Water temperature	2011-07-08	8	Jul	2011
98579	1628775	25.0	Chai	Water temperature	2011-07-17	17	Jul	2011
134290	3295351	25.0	Chai	Water temperature	2016-08-27	27	Aug	2016

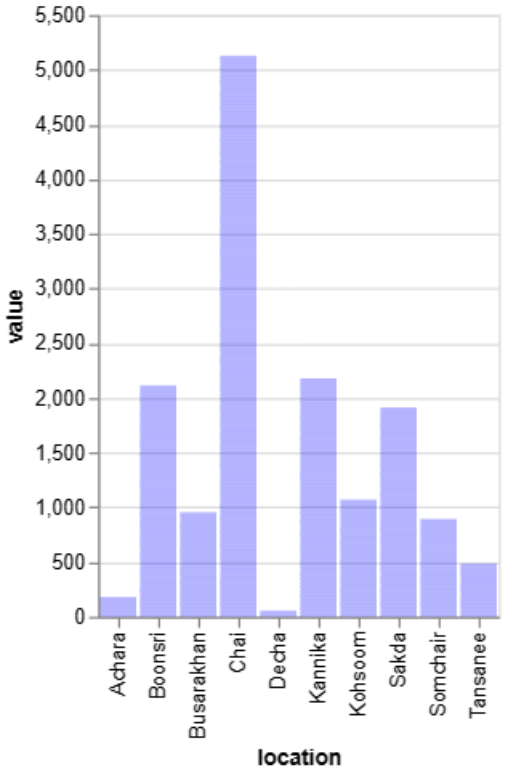
571 rows × 8 columns

In [32]:
```
highest_water_temp.sort_values(by=['year'], ascending=False)
highest_water_temp.max()
```

Out[32]:
```
id                          3448678
value                          36.4
location                   Tansanee
measure           Water temperature
Sample Date     2016-08-27 00:00:00
day                              31
month                           Sep
year                           2016
dtype: object
```

In [33]:
```
alt.Chart(highest_water_temp).mark_bar(color='blue'
, opacity=0.3).encode(
    x='location',
    y='value',
)
```

Out[33]:



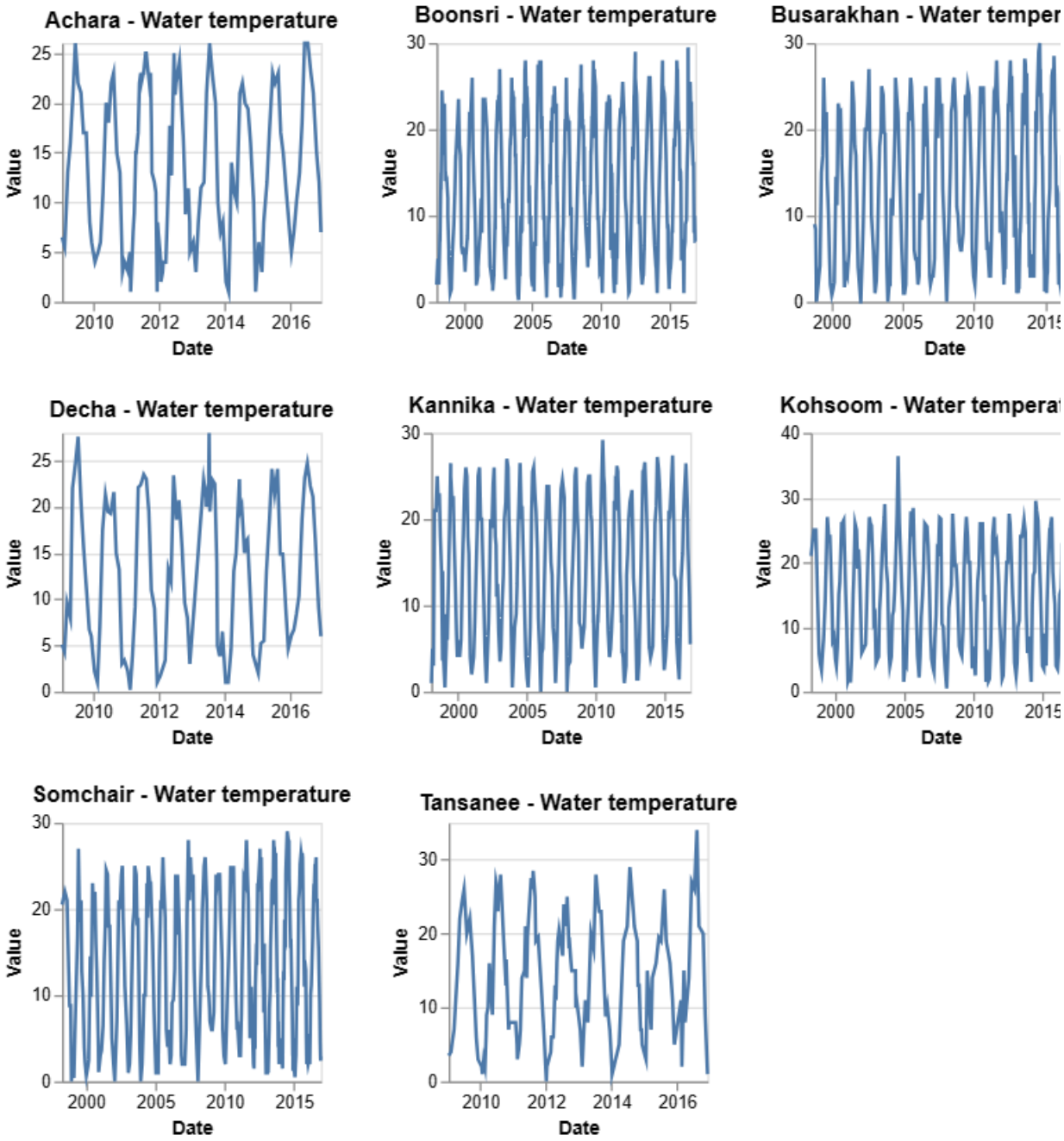
In [34]:

```
display(Markdown("Water Temperature Time Chart"))
def plot_measure_time_series(df, measure, y_scale=None):
    base_chart = alt.Chart(
        df[df['measure'] == measure]
    ).mark_line().encode(
        x=alt.X('Sample Date:T', axis=alt.Axis(title='Date', grid=False)),
        tooltip=['value', 'Sample Date']
    ).properties(
        width = 150,
        height = 150
    )
    ab_1 = alt.concat()
    ab_2 = alt.concat()
    ab_3 = alt.concat()

    for index, location in enumerate(locations):
        title = location + " - " + measure

        chart = base_chart.transform_filter(
            datum.location == location
        ).properties(
            title = title
        )

        if y_scale:
            chart = chart.encode(
                y = alt.Y('value', axis=alt.Axis(title='Value'),
                          scale=alt.Scale(domain=y_scale, clamp=True))
            )
        else:
            chart = chart.encode(
                y = alt.Y('value', axis=alt.Axis(title='Value'))
            )
        if index < 4:
```

```
            ab_1 |= chart
        elif 4 <= index <= 7:
            ab_2 |= chart
        else:
            ab_3 |= chart
    # vertically concatenating all the rows
    return alt.vconcat(ab_1, ab_2, ab_3)
plot_measure_time_series(df,'Water temperature')
```

Water Temperature Time Chart

Out[34]:



In [35]:
```
location_dropdown = alt.binding_select(options=locations)
```

In [36]:
```
from distutils.sysconfig import get_python_lib
import joblib
```

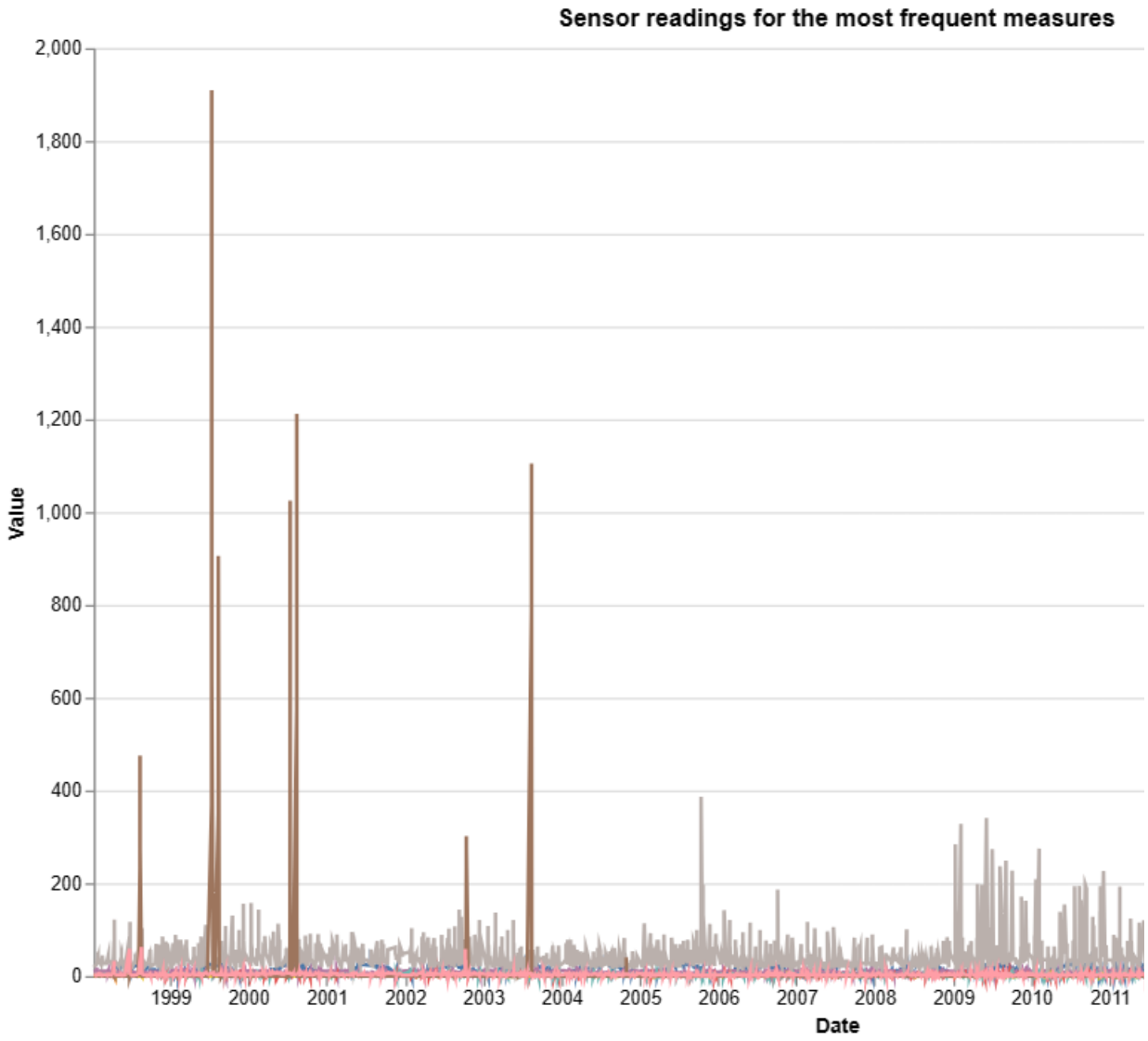
In [37]:
```
location_selection = alt.selection_point(
    fields=['location'], bind=location_dropdown, name='Select ')
```

In [38]:
```
measures_selection = alt.selection_point(fields=['measure'], bind='legend')
```

In [39]:
```
interest_df = df[df['measure'].isin(top_10_count)]
```

In [40]:
```
alt.Chart(
    interest_df, title=("Sensor readings for the most frequent measures")
).mark_line().encode(
    x=alt.X('Sample Date:T', axis=alt.Axis(title='Date', grid=False)),
    y=alt.Y('value', axis=alt.Axis(title='Value')),
    color=alt.Color('measure:N', scale=alt.Scale(domain=top_10_count)),
    tooltip=['location', 'measure', 'value', 'Sample Date']
).add_params(
    measures_selection,
    location_selection
).transform_filter(
    measures_selection
).transform_filter(
    location_selection
).properties(
    width=800,
    height=500
)
```

Out[40]:



Select_location Achara

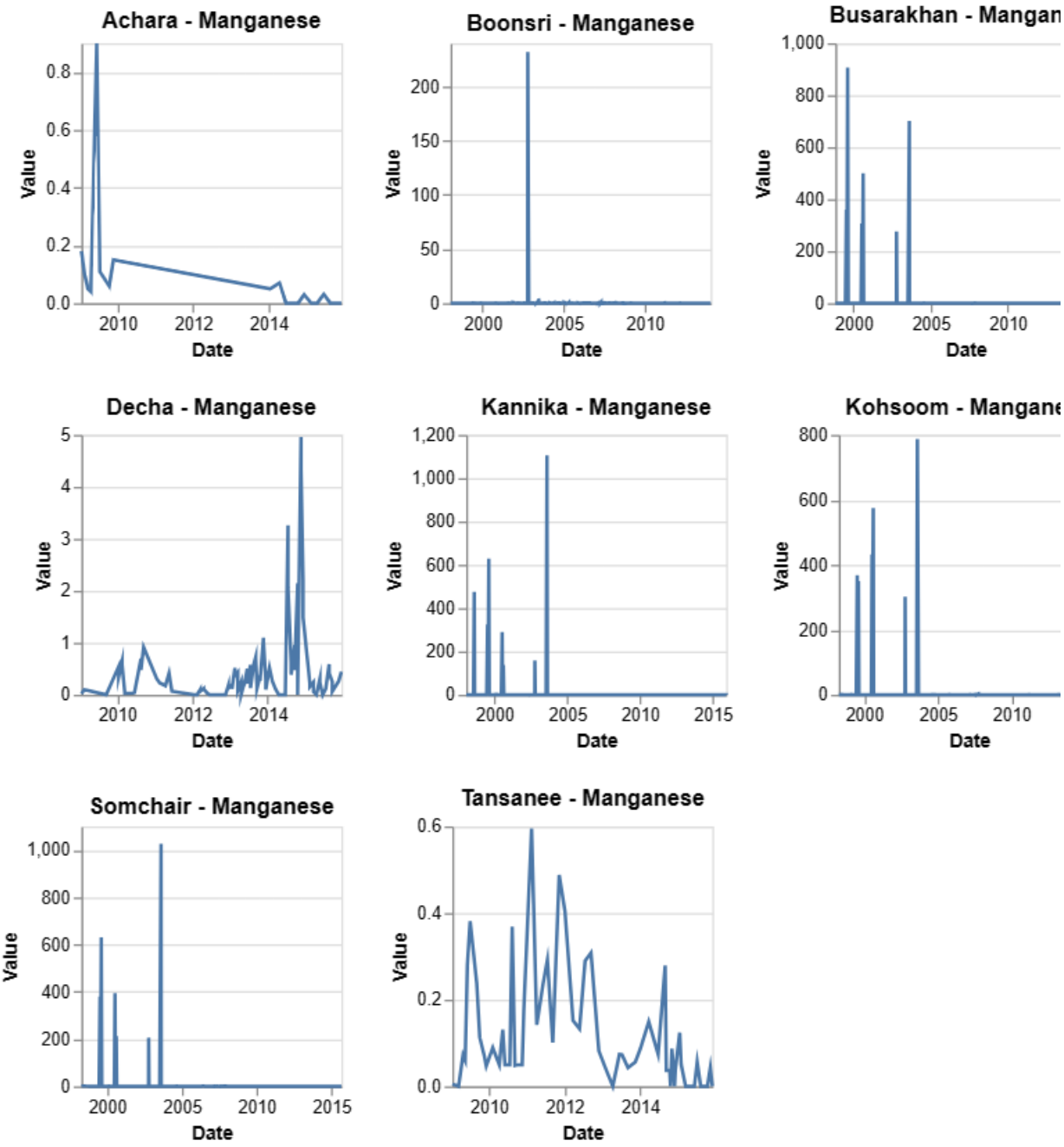
In [41]:

```
display(Markdown("Manganese Time Chart"))
def plot_measure_time_series(df, measure, y_scale=None):
    base_chart = alt.Chart(
        df[df['measure'] == measure]
    ).mark_line().encode(
        x=alt.X('Sample Date:T', axis=alt.Axis(title='Date', grid=False)),
        tooltip=['value', 'Sample Date']
    ).properties(
        width = 150,
        height = 150
    )
    ab_1 = alt.concat()
    ab_2 = alt.concat()
    ab_3 = alt.concat()

    for index, location in enumerate(locations):
        title = location + " - " + measure

        chart = base_chart.transform_filter(
            datum.location == location
        ).properties(
```

```
            title = title
        )

        if y_scale:
            chart = chart.encode(
                y = alt.Y('value', axis=alt.Axis(title='Value'),
                          scale=alt.Scale(domain=y_scale, clamp=True))
            )
        else:
            chart = chart.encode(
                y = alt.Y('value', axis=alt.Axis(title='Value'))
            )
        if index < 4:
            ab_1 |= chart
        elif 4 <= index <= 7:
            ab_2 |= chart
        else:
            ab_3 |= chart
    # vertically concatenating all the rows
    return alt.vconcat(ab_1, ab_2, ab_3)
plot_measure_time_series(df,'Manganese')
```

Manganese Time Chart

Out[41]:



In [42]:
```
df[df['measure'] == 'Manganese']['value'].describe()
```

Out[42]:
```
count    4039.000000
mean        5.526752
std        64.646204
min         0.000000
25%         0.006650
50%         0.039000
75%         0.080000
max      1910.000000
Name: value, dtype: float64
```

In [43]:
```
highest_manganese_temp=df[(df['measure'] == 'Manganese') & (df['value'] >=0.5000)]
highest_manganese_temp.sort_values(by=['value'], ascending=False)
```

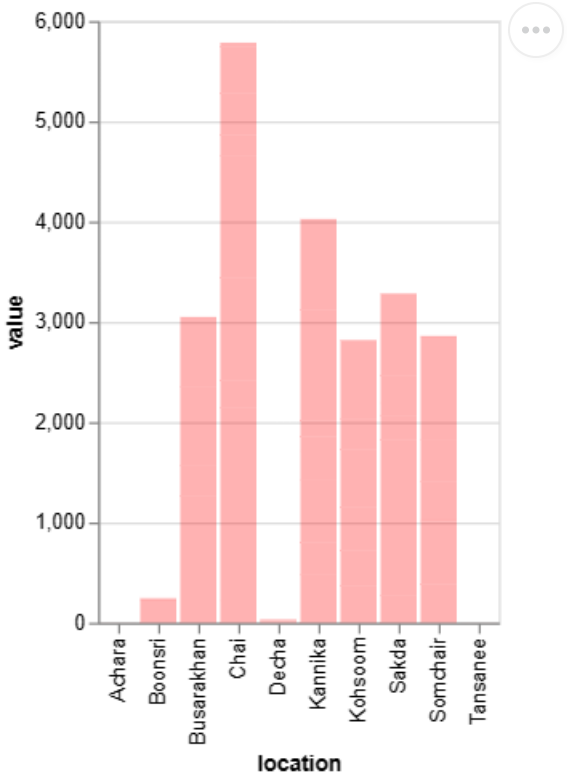
Out[43]:

	id	value	location	measure	Sample Date	day	month	year
6394	46324	1910.000	Chai	Manganese	1999-07-15	15	Jul	1999
12093	104879	1212.000	Chai	Manganese	2000-08-15	15	Aug	2000
27987	303330	1105.200	Kannika	Manganese	2003-08-15	15	Aug	2003
27943	300557	1027.100	Somchair	Manganese	2003-08-15	15	Aug	2003
11716	104838	1025.000	Chai	Manganese	2000-07-15	15	Jul	2000
...	...	...	...	...	...	...	...	...
87272	1327896	0.544	Decha	Manganese	2010-01-17	17	Jan	2010
111116	2188534	0.521	Kohsoom	Manganese	2013-07-25	25	Jul	2013
114163	2566591	0.520	Decha	Manganese	2014-01-18	18	Jan	2014
108463	2166020	0.518	Decha	Manganese	2013-02-17	17	Feb	2013
110268	2166203	0.503	Decha	Manganese	2013-06-17	17	Jun	2013

98 rows × 8 columns

In [44]:
```
alt.Chart(highest_manganese_temp).mark_bar(color='red'
, opacity=0.3).encode(
    x='location',
    y='value',
)
```

Out[44]:



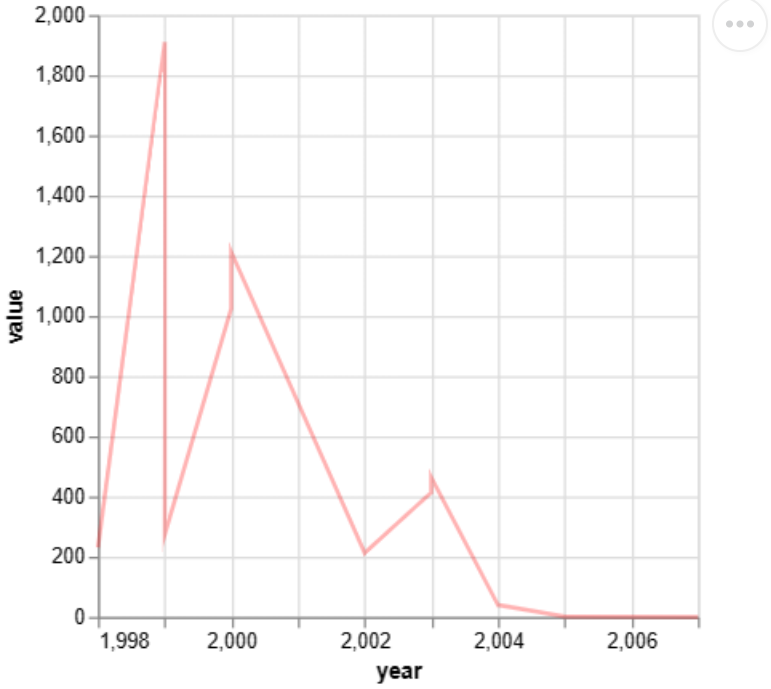
In [45]:
```
highest_manganese_temp1=df[(df['measure'] == 'Manganese') & (df['value'] >=0.5000) & (
highest_manganese_temp1.sort_values(by=['value'], ascending=False)
```

Out[45]:

	id	value	location	measure	Sample Date	day	month	year
6394	46324	1910.000	Chai	Manganese	1999-07-15	15	Jul	1999
12093	104879	1212.000	Chai	Manganese	2000-08-15	15	Aug	2000
11716	104838	1025.000	Chai	Manganese	2000-07-15	15	Jul	2000
27921	300093	463.070	Chai	Manganese	2003-08-15	15	Aug	2003
27899	298745	414.000	Chai	Manganese	2003-08-15	15	Aug	2003
6857	46363	274.000	Chai	Manganese	1999-08-15	15	Aug	1999
1990	5326	231.300	Chai	Manganese	1998-08-15	15	Aug	1998
22910	235806	213.000	Chai	Manganese	2002-10-15	15	Oct	2002
35252	512908	40.000	Chai	Manganese	2004-10-30	30	Oct	2004
37624	604788	1.135	Chai	Manganese	2005-02-28	28	Feb	2005
60313	803143	0.620	Chai	Manganese	2007-04-12	12	Apr	2007

In [46]:
```
alt.Chart(highest_manganese_temp1).mark_line(color='red'
, opacity=0.3).encode(
    x='year',
    y='value',
)
```

Out[46]:



In [ ]: