Name: Iqra Ilyas

MSIS: M00909152

Course: CST4060 - CW2

Task: To investigate the sensor readings to find possible link to the bird population deduction.

# *Importing Required Libraries and Dependencies*

In [2]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
import numpy as np
import pandas as pd

import altair as alt
```

# *Loading Dataset*

In [4]:
```
wildlife_dataset = pd.read_csv("Boonsong Lekagul waterways readings.csv")
```

In [5]:
```
wildlife_dataset.head()
```

Out[5]:

| | 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 |

# *Data Pre-Processing*

In [6]:
```
print(f"The dataset has {wildlife_dataset.shape[0]} rows and {wildlife_dataset.shape[1]} columns.")
```

```
The dataset has 136824 rows and 5 columns.
```

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

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

In [8]:
```
print(f"Duplicate rows in dataset : {wildlife_dataset.duplicated().sum()}")
```

```
Duplicate rows in dataset : 0
```

In [9]:
```
wildlife_dataset.describe()
```

Out[9]:

| | id | value |
|---|---|---|
| **count** | 1.368240e+05 | 136824.000000 |
| **mean** | 1.197736e+06 | 24.021591 |
| **std** | 1.000893e+06 | 231.254038 |
| **min** | 2.221000e+03 | 0.000000 |
| **25%** | 5.147528e+05 | 0.059950 |
| **50%** | 8.891045e+05 | 1.862100 |
| **75%** | 1.640213e+06 | 14.100000 |
| **max** | 3.448888e+06 | 37959.280000 |

**OUTLIERS**

In [10]:
```
# The IQR is used to identify outliers based on a specified multiplier (1.5 times the IQR).
# The dataset is then filtered to exclude values that fall outside the lower and upper bounds determined by the IQR
# and multiplier.

Q1 = wildlife_dataset.groupby('measure')['value'].quantile(0.25).reset_index()
Q3 = wildlife_dataset.groupby('measure')['value'].quantile(0.75).reset_index()
IQR = pd.merge(Q1, Q3, on='measure', suffixes=('_Q1', '_Q3'))
IQR['IQR'] = IQR['value_Q3'] - IQR['value_Q1']

outlier_multiplier = 1.5

dataset_with_iqr = wildlife_dataset.merge(IQR, on='measure')

filtered_dataset = dataset_with_iqr[
    (dataset_with_iqr['value'] >= dataset_with_iqr['value_Q1'] - outlier_multiplier * dataset_with_iqr['IQR']) &
    (dataset_with_iqr['value'] <= dataset_with_iqr['value_Q3'] + outlier_multiplier * dataset_with_iqr['IQR'])
]

filtered_dataset.reset_index(drop = True, inplace = True)
filtered_dataset.drop(['value_Q1', 'value_Q3', 'IQR'], axis = 1, inplace = True)
filtered_dataset
```

Out[10]:

| | id | value | location | sample date | measure |
|---|---|---|---|---|---|
| **0** | 2221 | 2.0000 | Boonsri | 11-Jan-98 | Water temperature |
| **1** | 8291 | 1.0000 | Kannika | 26-Jan-98 | Water temperature |
| **2** | 8772 | 1.0000 | Kannika | 26-Jan-98 | Water temperature |
| **3** | 9255 | 1.0000 | Kannika | 26-Jan-98 | Water temperature |
| **4** | 5049 | 2.0000 | Chai | 31-Jan-98 | Water temperature |
| **...** | ... | ... | ... | ... | ... |
| **127091** | 3279933 | 0.0460 | Boonsri | 10-Dec-16 | Total dissolved phosphorus |
| **127092** | 3280838 | 0.0520 | Boonsri | 10-Dec-16 | Total dissolved phosphorus |
| **127093** | 3305578 | 0.1144 | Kohsoom | 12-Dec-16 | Total dissolved phosphorus |
| **127094** | 3281876 | 0.1200 | Somchair | 17-Dec-16 | Total dissolved phosphorus |
| **127095** | 3282872 | 0.0510 | Busarakhan | 17-Dec-16 | Total dissolved phosphorus |

127096 rows × 5 columns

**Dataframe Statistics**

In [11]:
```
numerical_cols = filtered_dataset.select_dtypes(include=['int64', 'float64'])

numerical_stats_table = pd.DataFrame({
    'Unique': numerical_cols.nunique(),
    'Null': numerical_cols.isna().sum(),
    'Types': numerical_cols.dtypes,
    'Mean': numerical_cols.mean(),
    'Standard Deviation': numerical_cols.std(),
    'Minimum': numerical_cols.min(),
    '25th Percentile': numerical_cols.quantile(0.25),
    'Median': numerical_cols.median(),
    '75th Percentile': numerical_cols.quantile(0.75),
    'Maximum': numerical_cols.max(),
    'Variance': numerical_cols.var(),
    'Mode': numerical_cols.mode().iloc[0]
})

numerical_stats_table['Range'] = numerical_stats_table['Minimum'].astype(str)+ ' - '
+ numerical_stats_table['Maximum'].astype(str)

numerical_stats_table
```

Out[11]:

| | Unique | Null | Types | Mean | Standard Deviation | Minimum | 25th Percentile | Median | 75th Percentile | Maximum | Variance | Mode | Range |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **id** | 127096 | 0 | int64 | 1.217513e+06 | 1.007394e+06 | 2221.0 | 520080.75 | 890983.5 | 1963551.25 | 3448888.0 | 1.014842e+12 | 2221.0 | 2221.0 - |
| **value** | 12513 | 0 | float64 | 1.961154e+01 | 5.006526e+01 | 0.0 | 0.05 | 1.8 | 13.00 | 439.0 | 2.506530e+03 | 0.0 | 0.0 - |

In [12]:
```
categorical_cols = filtered_dataset.select_dtypes(include=['object'])
```

```
categorical_stats_table = pd.DataFrame({
    'Unique': categorical_cols.nunique(),
    'Null': categorical_cols.isna().sum(),
    'NullType': categorical_cols.isna().sum() / len(filtered_dataset),
    'Types': categorical_cols.dtypes,
    'Top_Category': categorical_cols.mode().iloc[0],
    'Top_Category_Count': categorical_cols.mode().apply(lambda x: categorical_cols[x.name].value_counts().max())
})

categorical_stats_table
```

Out[12]:

| | Unique | Null | NullType | Types | Top_Category | Top_Category_Count |
|---|---|---|---|---|---|---|
| **location** | 10 | 0 | 0.0 | object | Boonsri | 29948 |
| **sample date** | 1974 | 0 | 0.0 | object | 10-May-07 | 472 |
| **measure** | 106 | 0 | 0.0 | object | Water temperature | 5031 |

In [13]:
```
filtered_dataset.columns
```

Out[13]:
```
Index(['id', 'value', 'location', 'sample date', 'measure'], dtype='object')
```

### Creating Columns

In [14]:
```
filtered_dataset['sample date'] = pd.to_datetime(filtered_dataset['sample date'], format='%d-%b-%y')

filtered_dataset['year'] = filtered_dataset['sample date'].dt.year
filtered_dataset['month'] = filtered_dataset['sample date'].dt.month
filtered_dataset['day'] = filtered_dataset['sample date'].dt.day
```

In [15]:
```
filtered_dataset.head()
```

Out[15]:

| | id | value | location | sample date | measure | year | month | day |
|---|---|---|---|---|---|---|---|---|
| **0** | 2221 | 2.0 | Boonsri | 1998-01-11 | Water temperature | 1998 | 1 | 11 |
| **1** | 8291 | 1.0 | Kannika | 1998-01-26 | Water temperature | 1998 | 1 | 26 |
| **2** | 8772 | 1.0 | Kannika | 1998-01-26 | Water temperature | 1998 | 1 | 26 |
| **3** | 9255 | 1.0 | Kannika | 1998-01-26 | Water temperature | 1998 | 1 | 26 |
| **4** | 5049 | 2.0 | Chai | 1998-01-31 | Water temperature | 1998 | 1 | 31 |

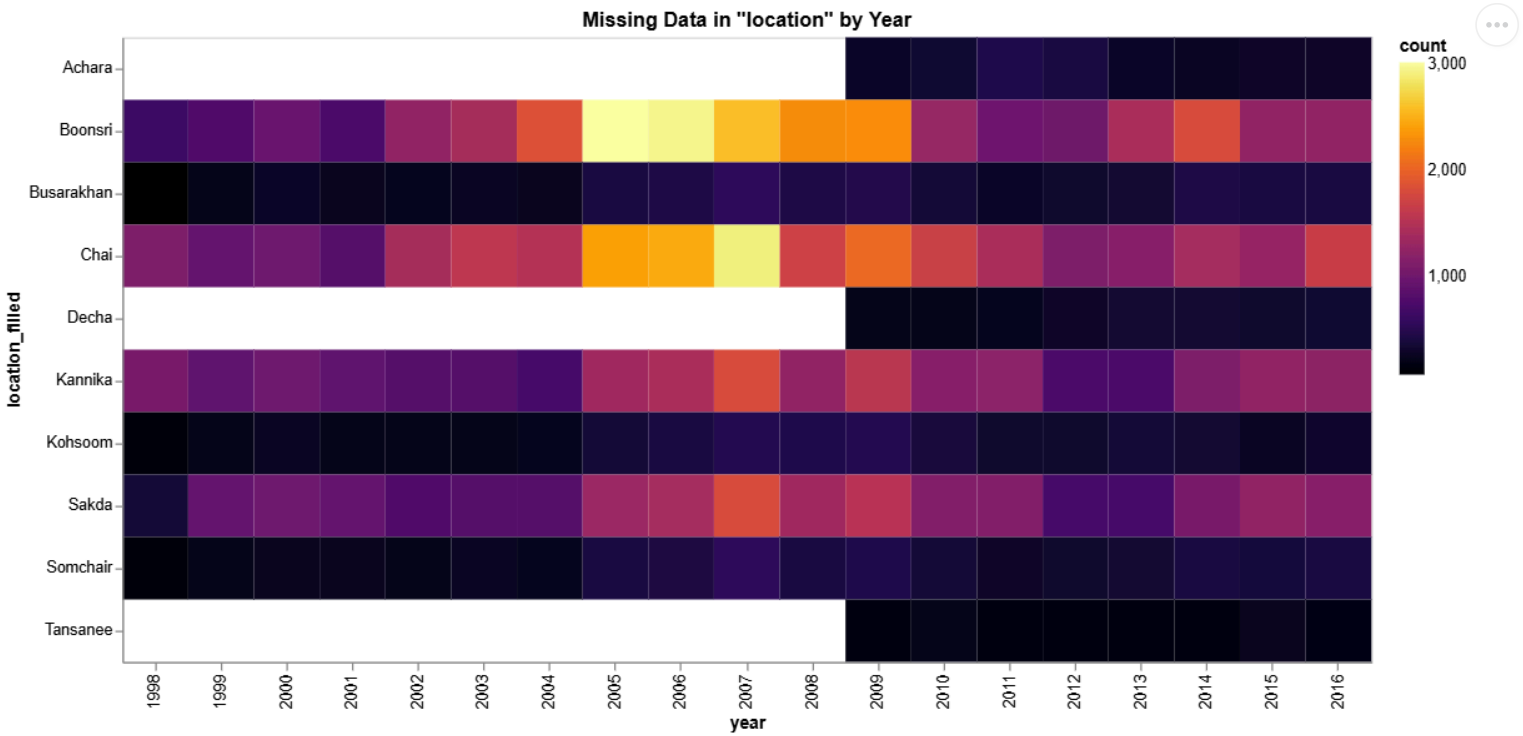
In [16]:
```
filtered_dataset.isna().sum()
```

Out[16]:
```
id             0
value          0
location       0
sample date    0
measure        0
year           0
month          0
day            0
dtype: int64
```

## Missing Data in 'Location' Over Time: A Heatmap of Counts by Year

In [17]:
```
# Generating heatmap to visualize missing data in the 'location' column of the filtered_dataset over the years.
# It first fills missing values in the 'location' column with the label 'Missing' and then creates a pivot table
# with years as the index, 'location_filled' as columns, and the count of values as the aggregated function.
# The resulting pivot table is melted to transform it into a format suitable for the heatmap.
# The color of each cell in the heatmap represents the count of values.

filtered_dataset['location_filled'] = filtered_dataset['location'].fillna('Missing')

pivot_df = filtered_dataset.pivot_table(index='year', columns='location_filled', values='value',
                                        aggfunc='count', fill_value=np.nan).reset_index()

melted_df = pd.melt(pivot_df, id_vars='year', var_name='location_filled', value_name='count')

heatmap = alt.Chart(melted_df).mark_rect().encode(
    x='year:O',
    y='location_filled:O',
    color=alt.Color('count:Q', scale=alt.Scale(scheme='inferno')),
    tooltip=['year:N', 'location_filled:N', 'count:Q']
).properties(
    width=800,
    height=400,
    title='Missing Data in "location" by Year'
)

heatmap
```

Out[17]:




# ANALYSIS

---

---

---

## Describe trends and anomalies with respect to chemical contamination

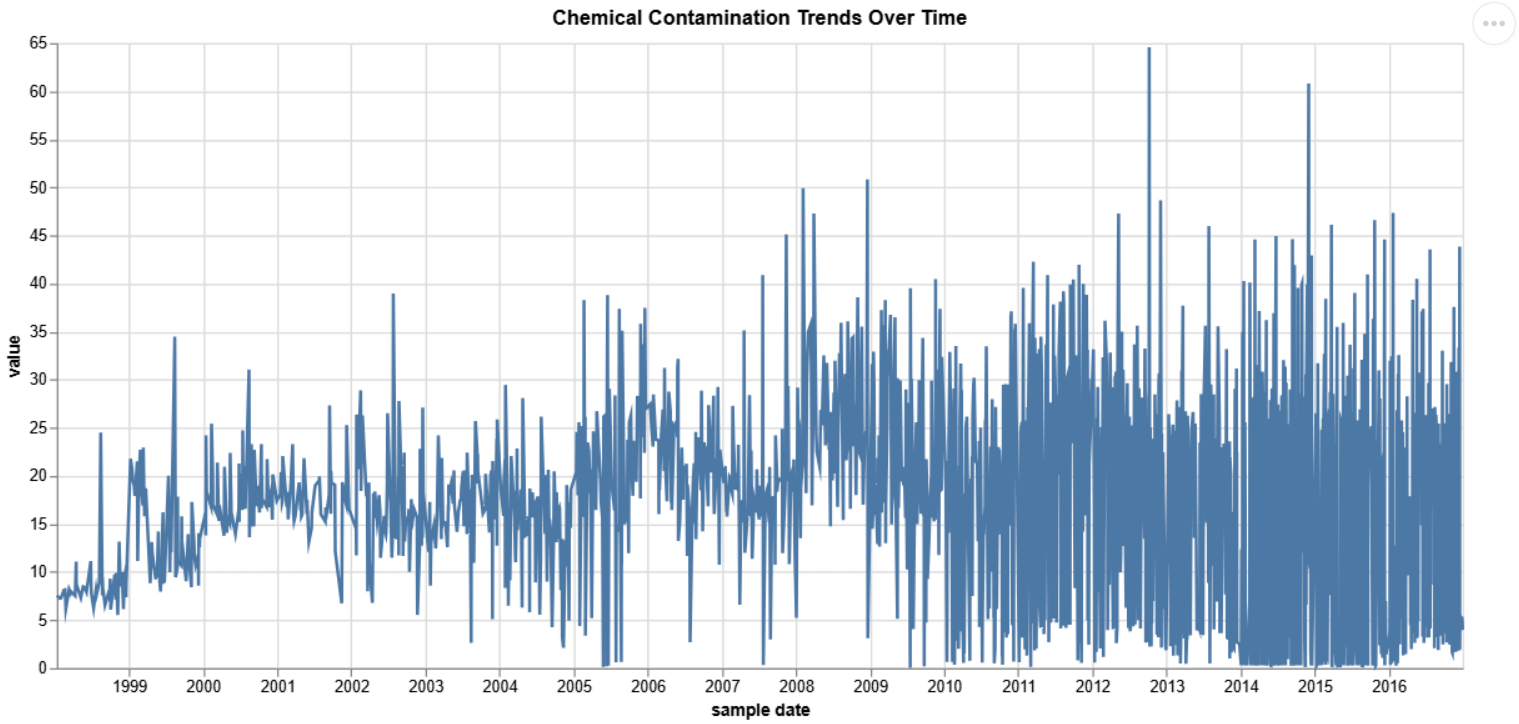
### i. Trends: changes over time and/or sensor site

In [18]:

```
# Aggregating dataset by the 'sample date' column, calculating the mean contamination value for each date

trends_over_time = filtered_dataset.groupby('sample date').agg({'value': 'mean'}).reset_index()

time_series_plot = alt.Chart(trends_over_time).mark_line().encode(
    x='sample date:T',
    y='value:Q',
    tooltip=['sample date:T', 'value:Q']
).properties(
    width=900,
    height=400,
    title='Chemical Contamination Trends Over Time'
)
```

```
time_series_plot
```

Out[18]:

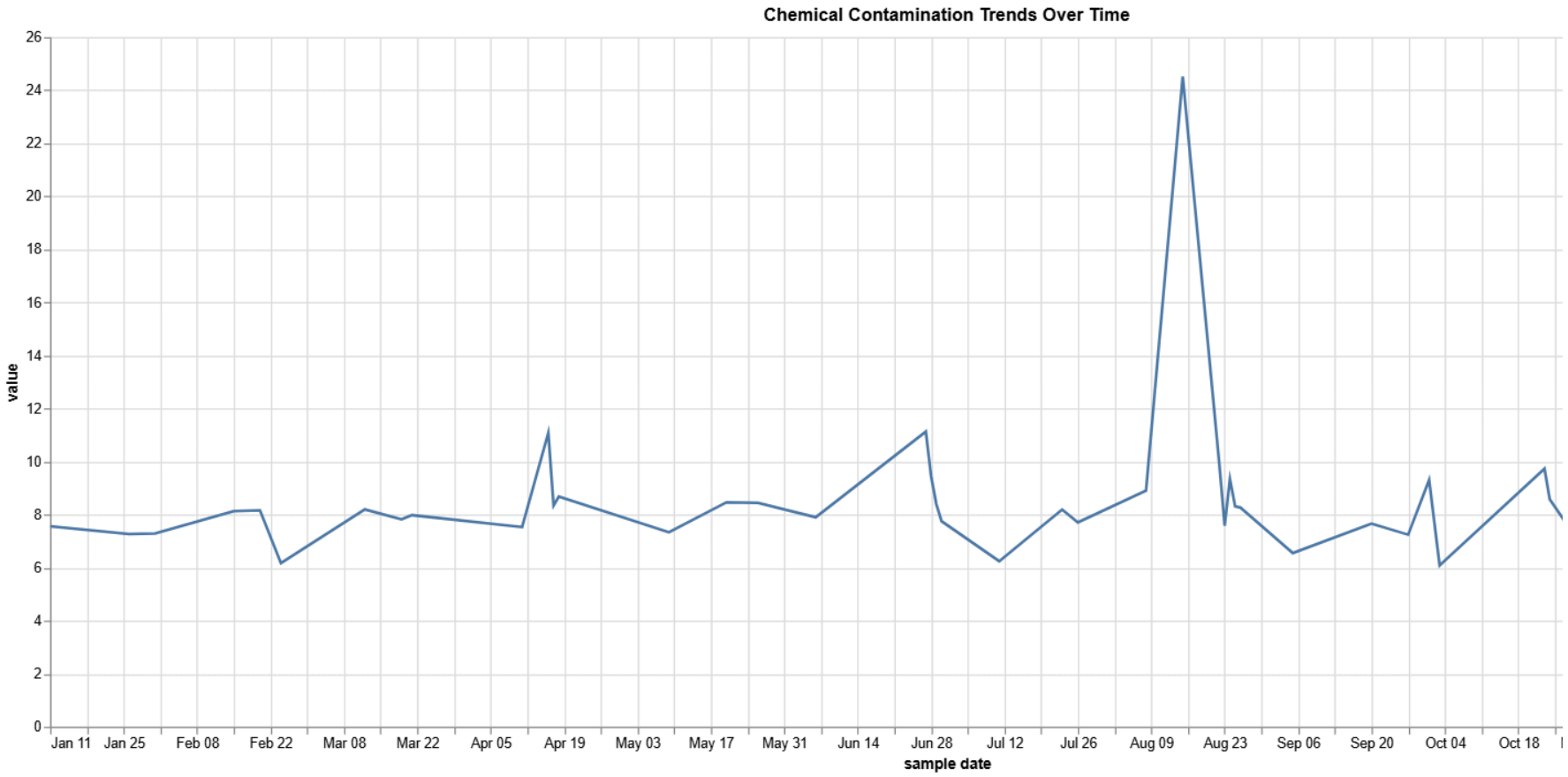



In [19]:

```
# Aggregating dataset by the 'sample date' column, calculating the mean contamination value for each date

# Group and aggregate the dataset
trends_over_time = filtered_dataset.groupby(['year', 'sample date']).agg({'value': 'mean'}).reset_index()

# Create a slider for selecting the year
year_slider = alt.binding_range(min=filtered_dataset['year'].min(),
                                max=filtered_dataset['year'].max(),
                                step=1,
                                name='Selection_Year: ')
year_value = alt.param(value=filtered_dataset['year'].min(), bind=year_slider)

# Create the time series plot
time_series_plot = alt.Chart(trends_over_time).mark_line().encode(
    x='sample date:T',
    y='value:Q',
    tooltip=['sample date:T', 'value:Q']
).add_params(
    year_value
).transform_filter(
    alt.datum.year == year_value
).properties(
    width=1300,
    height=500,
    title='Chemical Contamination Trends Over Time'
```

```
)

time_series_plot
```

Out[19]:




Selection_Year: 1998

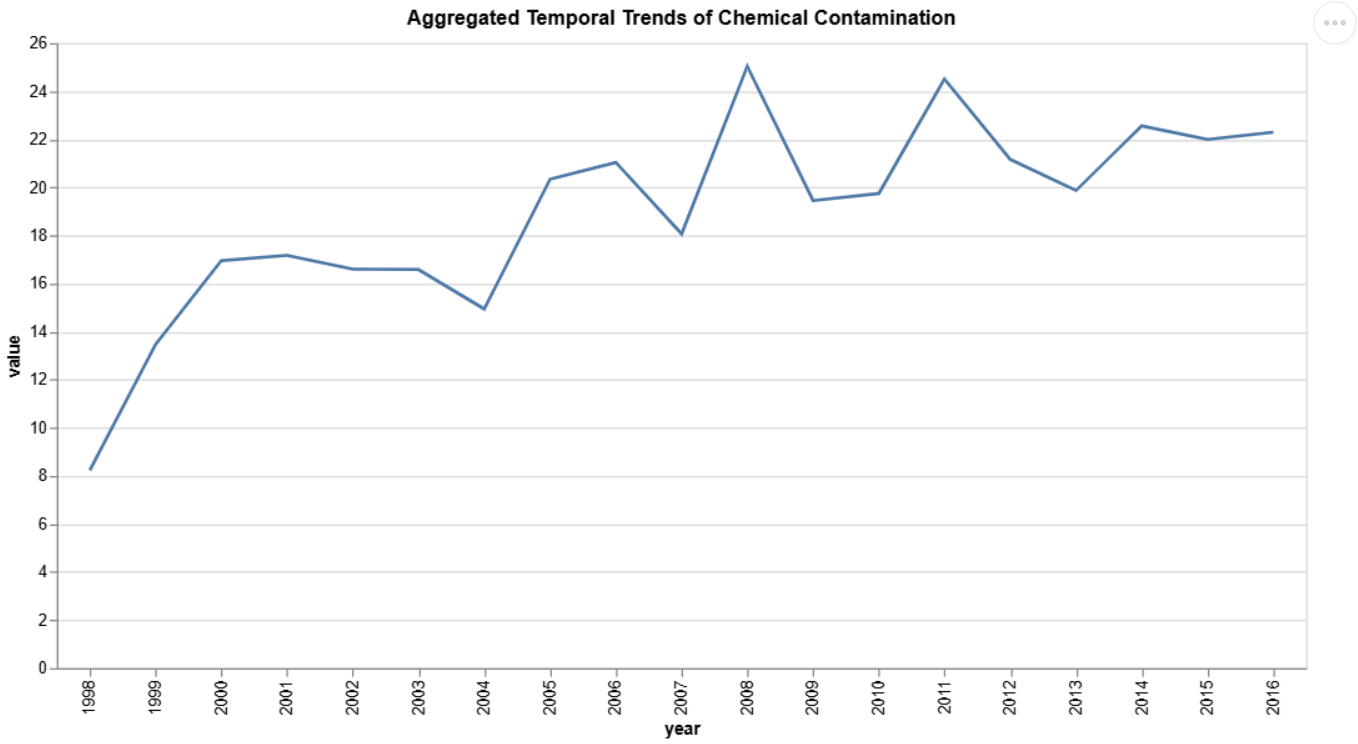
In [20]:

```
# Calculating the mean contamination value for each year and computing the proportional
# change in contamination values from one year to the next.

aggregated_df = filtered_dataset.groupby('year')['value'].mean().reset_index()
aggregated_df['proportional_change'] = aggregated_df['value'].pct_change() * 100

print("\n")
print(aggregated_df)
print("\n")

line_chart = alt.Chart(aggregated_df).mark_line().encode(
    x='year:N',
    y='value:Q',
    tooltip=['year:N', 'value:Q']
).properties(
    width=800,
```

```
    height=400,
    title='Aggregated Temporal Trends of Chemical Contamination'
)

line_chart
```

| | year | value | proportional_change |
|---|---|---|---|
| 0 | 1998 | 8.234349 | NaN |
| 1 | 1999 | 13.467198 | 63.549035 |
| 2 | 2000 | 16.950513 | 25.865184 |
| 3 | 2001 | 17.170719 | 1.299114 |
| 4 | 2002 | 16.602718 | -3.307965 |
| 5 | 2003 | 16.581720 | -0.126475 |
| 6 | 2004 | 14.944048 | -9.876371 |
| 7 | 2005 | 20.339027 | 36.101192 |
| 8 | 2006 | 21.037322 | 3.433275 |
| 9 | 2007 | 18.060527 | -14.150064 |
| 10 | 2008 | 25.038782 | 38.638160 |
| 11 | 2009 | 19.450464 | -22.318651 |
| 12 | 2010 | 19.741020 | 1.493826 |
| 13 | 2011 | 24.510030 | 24.157870 |
| 14 | 2012 | 21.167389 | -13.637849 |
| 15 | 2013 | 19.873357 | -6.113328 |
| 16 | 2014 | 22.557863 | 13.508067 |
| 17 | 2015 | 21.998330 | -2.480434 |
| 18 | 2016 | 22.298105 | 1.362717 |

Out[20]:




**OBSERVATIONS**

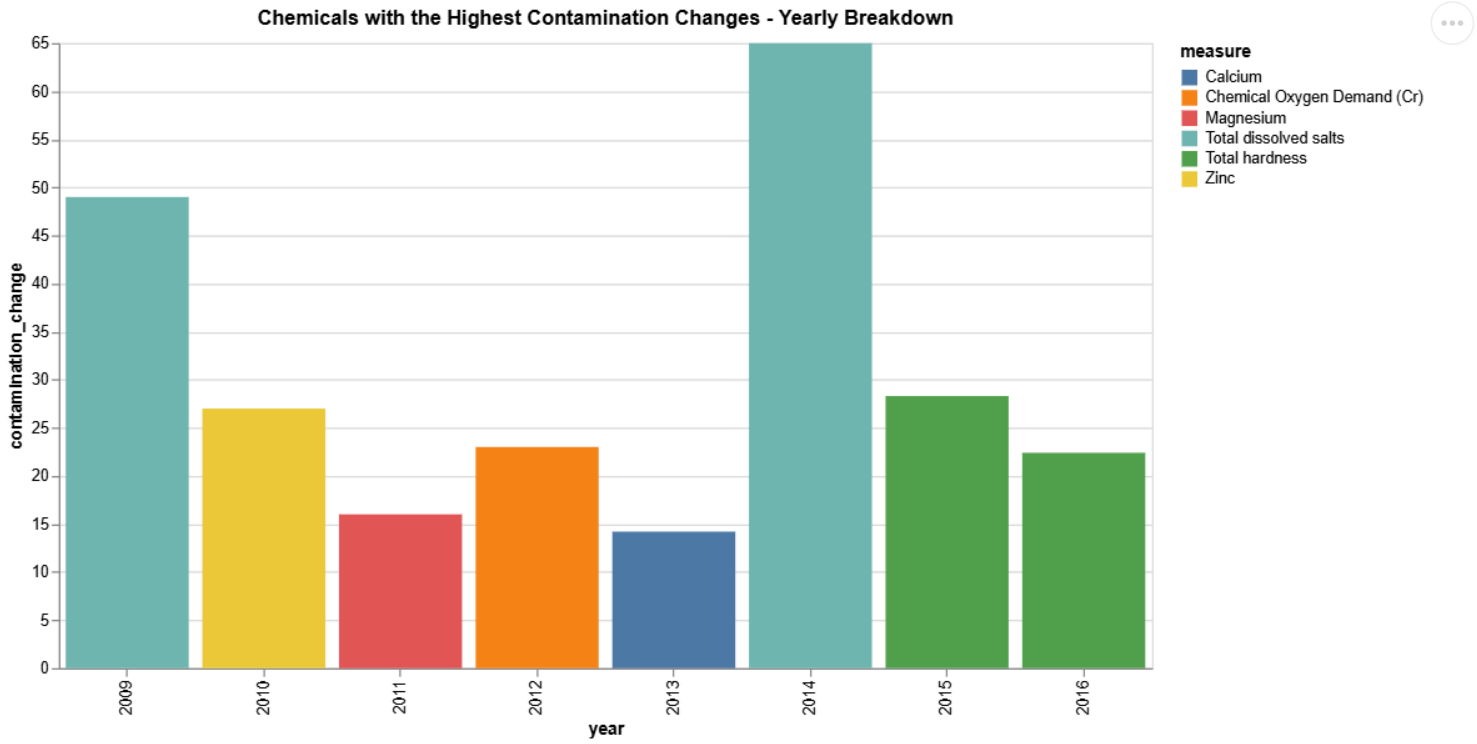
- The contamination level appears to increase from 1998 to 2005, with a significant spike in 2005.
- After 2005, there is a general trend of fluctuation, with both increases and decreases in contamination levels.
- Noticeable increases occur from 1998 to 2000 and from 2005 to 2008.
- The period between 2008 and 2011 also shows an increase.
- There is a significant decrease in contamination levels from 2008 to 2009.
- Another decrease is observed from 2011 to 2012.
- The years 2001, 2002, 2003, and 2016 show relatively stable or slightly decreasing contamination levels.
- Large proportional increases are observed in 2005, 2008, and 2011.
- The largest proportional decrease is seen in 2009.

In [21]:

```
# Calculating the change in contamination values for each measurement and location by taking difference
# between consecutive values, then identifying the top contaminant changes for each location and year.
# Organized in facets, with each column corresponding to a different location, allowing for a clear
# comparison of the highest contamination changes for each location on a yearly basis.

filtered_dataset['contamination_change'] = filtered_dataset.groupby(['location', 'measure'])['value'].diff()
top_contaminant_changes = filtered_dataset.loc[filtered_dataset.groupby(['location', 'year'])
                                               ['contamination_change'].idxmax()]

location_dropdown = alt.binding_select(options=top_contaminant_changes['location'].unique(),
                                       name='Select Location:')
location_selection = alt.selection_single(fields=['location'], bind=location_dropdown,
```

```
                                    value= top_contaminant_changes['location'].unique()[0])

bar_chart_top_changes = alt.Chart(top_contaminant_changes).mark_bar().encode(
    x='year:N',
    y='contamination_change:Q',
    color='measure:N',
    tooltip=['year:N', 'location:N', 'measure:N', 'contamination_change:Q']
).add_selection(
    location_selection
).transform_filter(
    location_selection
).properties(
    width=700,
    height=400,
    title='Chemicals with the Highest Contamination Changes - Yearly Breakdown'
)

bar_chart_top_changes
```

Out[21]:




Select Location: Achara

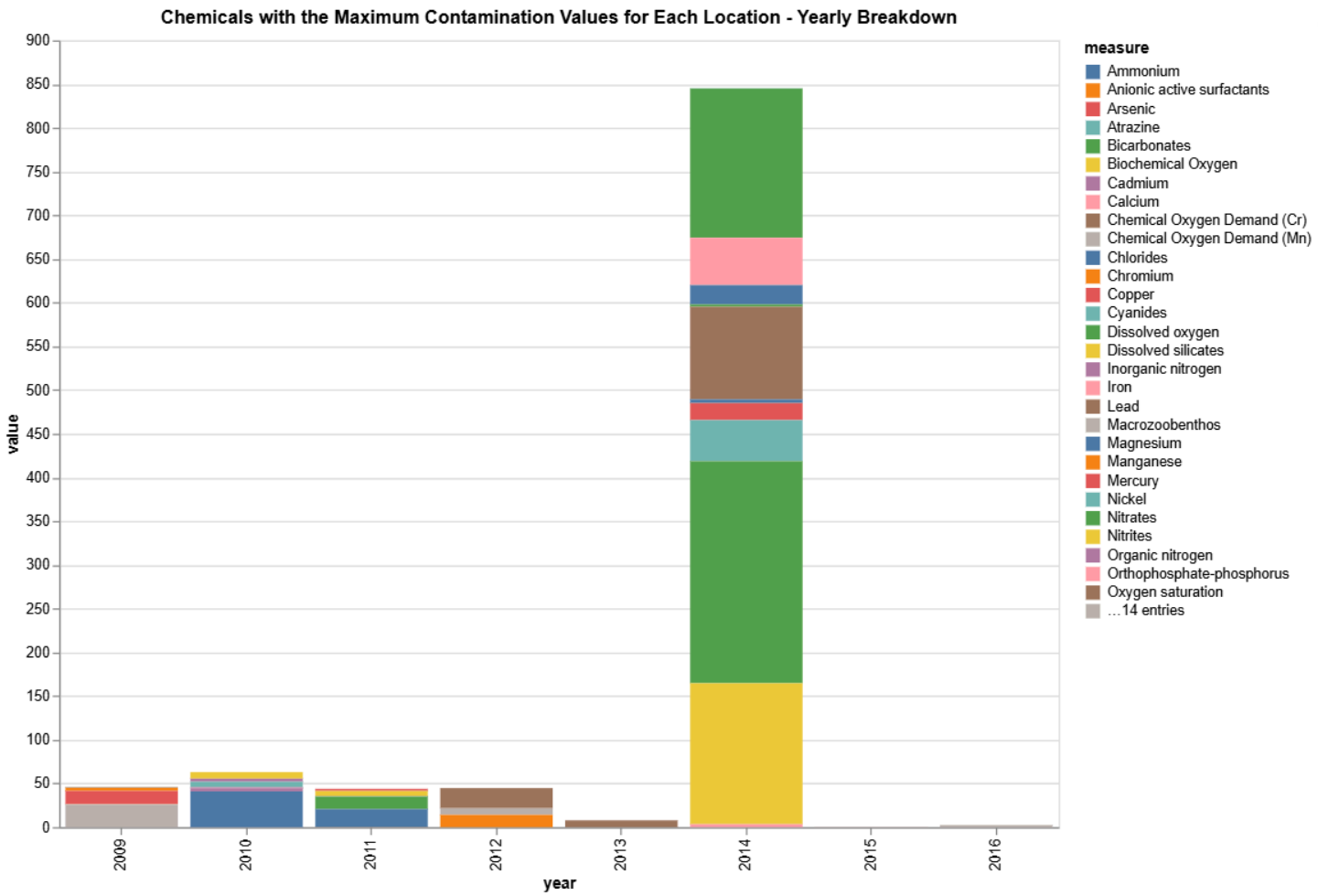
**OBSERVATIONS**

- The chemical most frequently associated with contamination changes across various locations and years is "total dissolved salt."
- Across multiple locations, a notable spike in contamination is observed in the year 2007 of total dissolved salt. This temporal pattern is consistent and may indicate a significant event or factor influencing contamination levels during that period.
- Different locations exhibit distinct contamination patterns. For instance:

- Bosarakhan, Somchair, Tonsanee: Bicarbonates contamination stands out.
- Boonsri, Kannika, Sakda: Zinc contamination shows noticeable changes.

In [22]:
```
# Calculating the change in contamination values for each measurement and location by taking difference
# between consecutive values, then identifying the top contaminant changes for each location and year.
# Additionally, finding the maximum contamination values for each measurement and location.
# This allows for a clear comparison of maximum contamination values for each location on a yearly basis.

filtered_dataset['contamination_change'] = filtered_dataset.groupby(['location', 'measure'])['value'].diff()

top_contaminant_changes = filtered_dataset.loc[filtered_dataset.groupby(['location', 'year'])
                                               ['contamination_change'].idxmax()]

max_contaminant_values = filtered_dataset.loc[filtered_dataset.groupby(['location', 'measure'])['value'].idxmax()]

location_dropdown = alt.binding_select(options=top_contaminant_changes['location'].unique(),
                                       name='Select Location:')
location_selection = alt.selection_single(fields=['location'], bind=location_dropdown,
                                          value= top_contaminant_changes['location'].unique()[0])

bar_chart_max_values = alt.Chart(max_contaminant_values).mark_bar().encode(
    x='year:N',
    y='value:Q',
    color='measure:N',
    tooltip=['year:N', 'location:N', 'measure:N', 'value:Q']
).properties(
    width=700,
    height=550,
    title='Chemicals with the Maximum Contamination Values for Each Location - Yearly Breakdown'
).add_selection(location_selection).transform_filter(location_selection)

bar_chart_max_values
```

Out[22]:




Select Location: Achara

## Anomalies: sudden change over time or one site significantly different from others.

Calculating z-scores for each observation in a dataset, considering the grouping by 'measure', 'location', and 'year'. Z-scores quantify how far each data point is from the mean within its specific group. A threshold value of 2 is set for the absolute z-scores. Observations with absolute z-scores exceeding this threshold are identified as outliers. This helps detect data points that deviate significantly from the mean within their respective groups, aiding in the identification of potential anomalies.

In [23]:

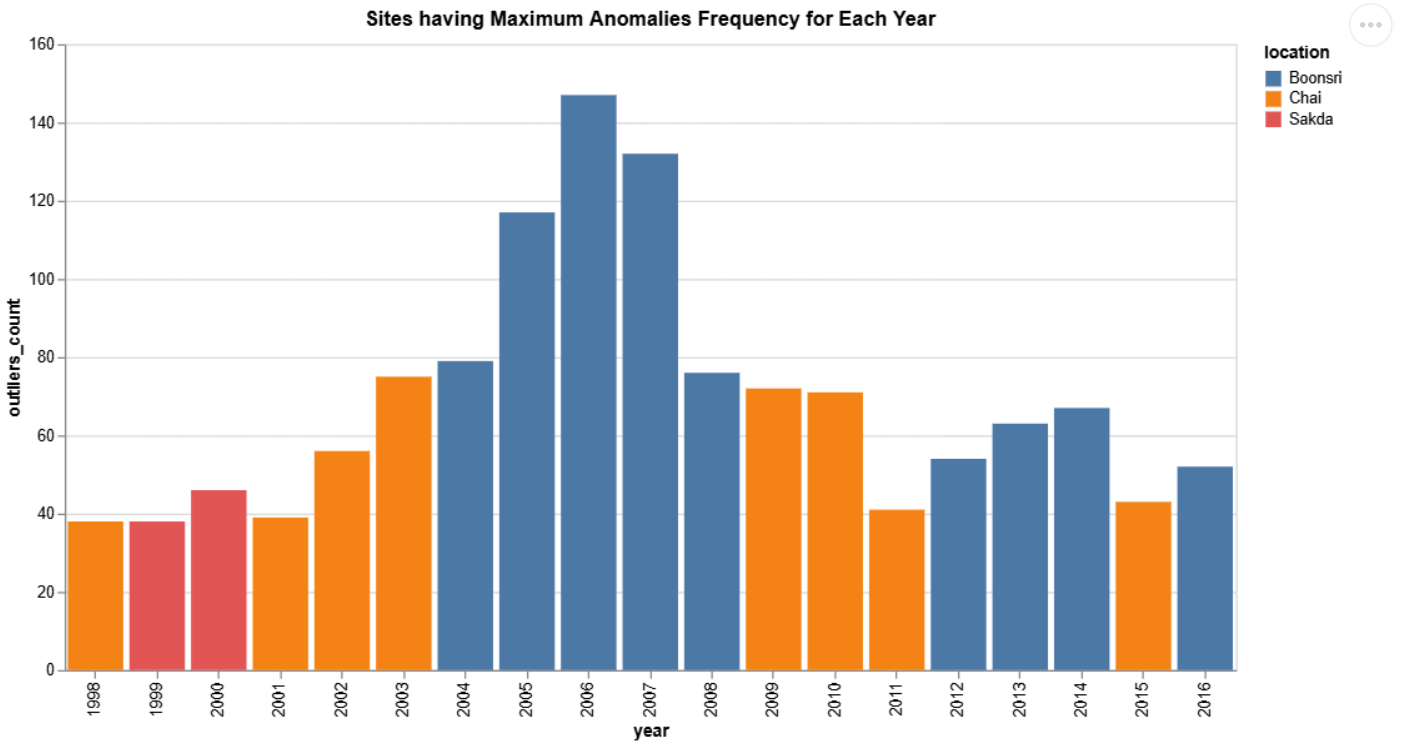
```
# Calculating the z-scores for the contamination values. Z-scores are a measure of how many standard deviations
# a data point is from the mean of a group of data points.
# The z-scores are computed separately for each combination of measurement (measure), location, and year.
# The threshold for identifying outliers is set to 2 standard deviations (z_score_threshold = 2).
# The outliers are then selected and stored
# These outliers represent contamination values that deviate significantly from the mean for their respective
#groups, indicating potential anomalies or extreme values in the data.
```

```
filtered_dataset['z_score'] = filtered_dataset.groupby(['measure', 'location', 'year']
                                                       )['value'].transform(lambda x: (x - x.mean()) / x.std())

z_score_threshold = 2

outliers_df = filtered_dataset[abs(filtered_dataset['z_score']) > z_score_threshold]
```

In [24]:

```
# Calculating the count of outliers for each combination of year and location from the outliers_df DataFrame.
# It then identifies, for each year, the location with the maximum count of outliers.
# Each bar represents the count of outliers, and the bars are grouped by year.
# The color of the bars indicates the corresponding location.
# Intended to provide a visual representation of locations with the highest anomaly frequency in each year.

outliers_count_df = outliers_df.groupby(['year', 'location']).size().reset_index(name='outliers_count')

max_outliers_location_df = outliers_count_df.loc[outliers_count_df.groupby('year')['outliers_count'].idxmax()]

bar_chart_max_outliers = alt.Chart(max_outliers_location_df).mark_bar().encode(
    x='year:N',
    y='outliers_count:Q',
    color='location:N',
    tooltip=['year:N', 'outliers_count:Q', 'location:N']
).properties(
    width=750,
    height=400,
    title='Sites having Maximum Anomalies Frequency for Each Year'
)

bar_chart_max_outliers
```

Out[24]:




**OBSERVATIONS**

1. Temporal Patterns:

- The number of outliers varies across years, indicating temporal fluctuations in contamination levels.
- Years 2006 and 2007 show a substantial increase in outliers, especially in the "Boonsri" location.

1. Location Disparities:

- Different locations exhibit varying degrees of contamination, with "Boonsri" consistently having higher outlier counts.
- The location "Chai" also experiences notable outlier counts, particularly in 2003.

1. Potential Environmental Events:

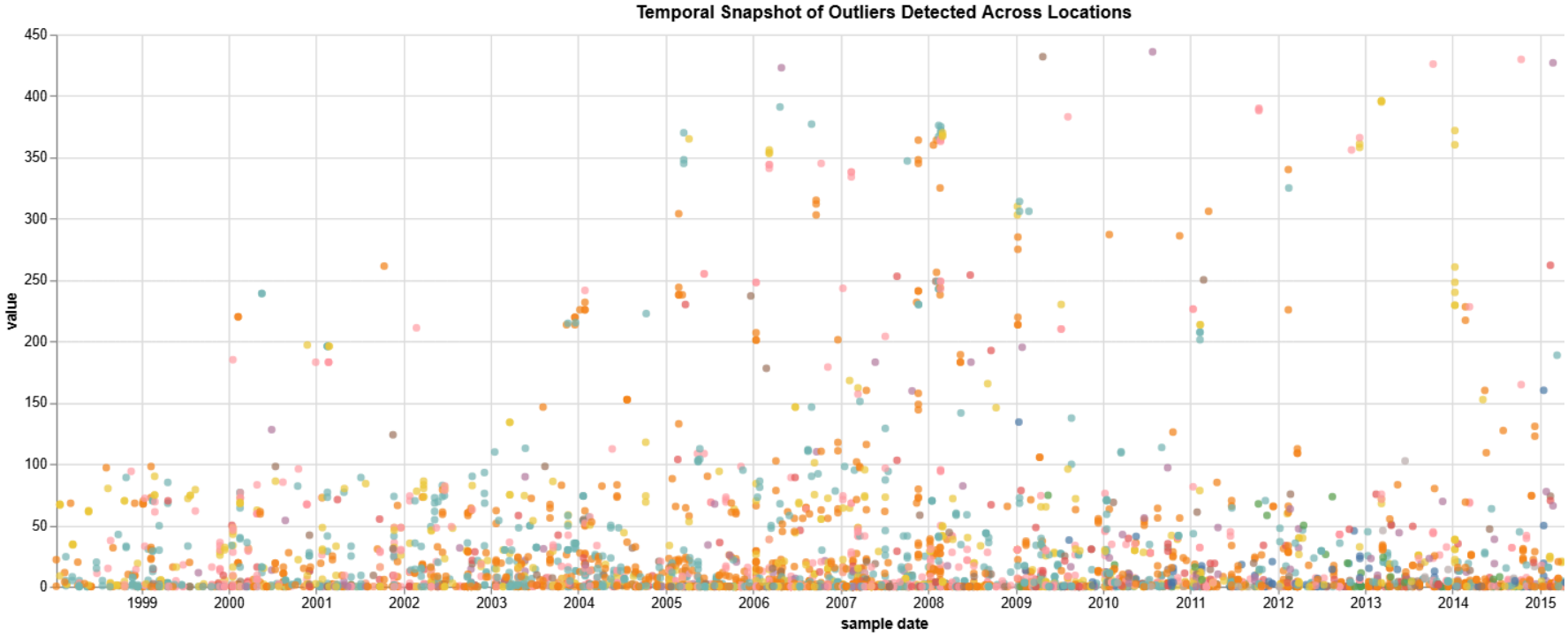
- The years with the highest outlier counts, such as 2006 and 2007, might be associated with specific environmental events or industrial activities.
- Some years witness a significant increase or decrease in outliers, suggesting dynamic environmental conditions.
- These fluctuations could be influenced by factors such as weather, land use changes, or industrial practices.

In [25]:

```
# A temporal snapshot of outliers detected across different locations. Each circle in the plot represents
# an outlier with its position determined by the sample date and the corresponding value.
# The circles are color-coded based on the location, and tooltips provide additional information about the sample
# date,value,and location. A visual overview of outlier occurrences over time and their distribution across locations.

outliers_chart = alt.Chart(outliers_df).mark_circle(color='red').encode(
```

```
    x='sample date:T',
    y='value:Q',
    color='location:N',
    tooltip=['sample date:T', 'value:Q', 'location:N']
).properties(
    width=1200,
    height=400,
    title='Temporal Snapshot of Outliers Detected Across Locations'
)

outliers_chart
```

Out[25]:




**The chart provides a visual representation of outliers detected over time (year-wise) at different locations, emphasizing the temporal aspect and the variation across sites.**

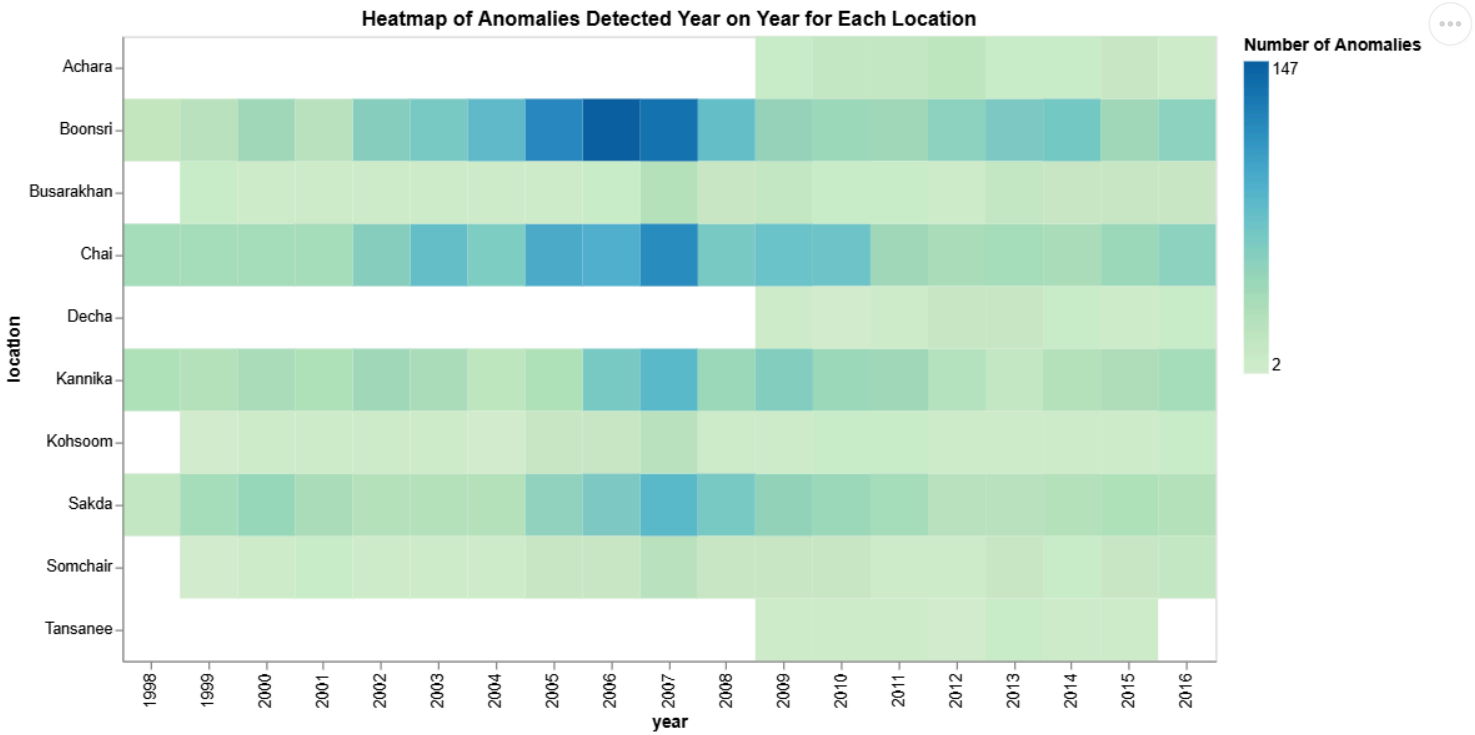
**Splitting on YoY site wise**

In [26]:

```
# Representating the number of anomalies detected year on year for each location.
# It provides insights into the annual variations in anomaly counts for a specific location.

heatmap = alt.Chart(outliers_df).mark_rect().encode(
    x='year:N',
    y='location:N',
    color=alt.Color('count():Q', scale=alt.Scale(scheme='greenblue'), title='Number of Anomalies'),
    tooltip=['year:N', 'location:N', 'count():Q']
).properties(
```

```
    width=700,
    height=400,
    title='Heatmap of Anomalies Detected Year on Year for Each Location'
)

heatmap
```

Out[26]:




**OBSERVATIONS**

- **Achara**: No anomalies were observed until 2008. From 2009 onward, anomalies started to emerge, indicating a potential shift in environmental conditions.
- **Boonsri**: This site experienced the highest number of anomalies, peaking in 2006. The anomalies increased consistently from 2001, reaching their zenith in 2006, followed by a decline until 2009.
- **Busarakhan**: Exhibits slight fluctuations over the years, with a peak in anomalies observed in 2007.
- **Chai**: Anomalies show a relatively constant pattern until 2001, after which random spikes and dips are observed. The anomalies reached their highest count in 2007, followed by continued fluctuations.
- **Decha**: No anomalies were recorded until 2009, suggesting a relatively stable environment. After 2009, anomalies appeared sporadically.
- **Kannika**: nomalies increased steadily from 2004 to reach their highest count in 2007. The pattern exhibits spikes and dips until 2004, followed by a consistent upward trend, further spikes, and dips, and a decline from 2009.

-- **Kohsoom**: Anomalies peaked in 2007, with a generally constant pattern observed over the years.

-- **Sakda**: Anomalies showed a consistent increase from 2004, reaching their highest count in 2007. Subsequently, a decline was observed until 2012.

-- **Somchair**:The highest count of anomalies was recorded in 2007, with generally uniform spikes and dips observed over the years.

-- **Tansanee**: Anomalies were detected consistently from 2009 to 2015, indicating a shift in the environmental conditions during this period.

By using z-scores, trying to identify points that deviate significantly from the mean, suggesting potential outliers in the sensor readings. The circles in the chart represent these outliers. The facetted structure of the chart allows you to compare outlier patterns across different locations.

In [27]:
```
color_scale = 'viridis'

location_dropdown = alt.binding_select(options=outliers_df['location'].unique(), name='Select Location:')
location_selection = alt.selection_single(fields=['location'], bind=location_dropdown,
                                          value=outliers_df['location'].unique()[0])

scatter_chart = alt.Chart(outliers_df).mark_circle().encode(
    x='sample date:T',
    y='value:Q',
    size='value:Q',
    color=alt.Color('value:Q', scale=alt.Scale(scheme=color_scale)),
    tooltip=['sample date:T', 'value:Q', 'location:N']
).add_selection(
    location_selection
).transform_filter(
    location_selection
).properties(
    width=850,
    height=550,
    title='Outliers Detected for Each Year and Location'
)

scatter_chart
```

Out[27]:


Select Location: Chai

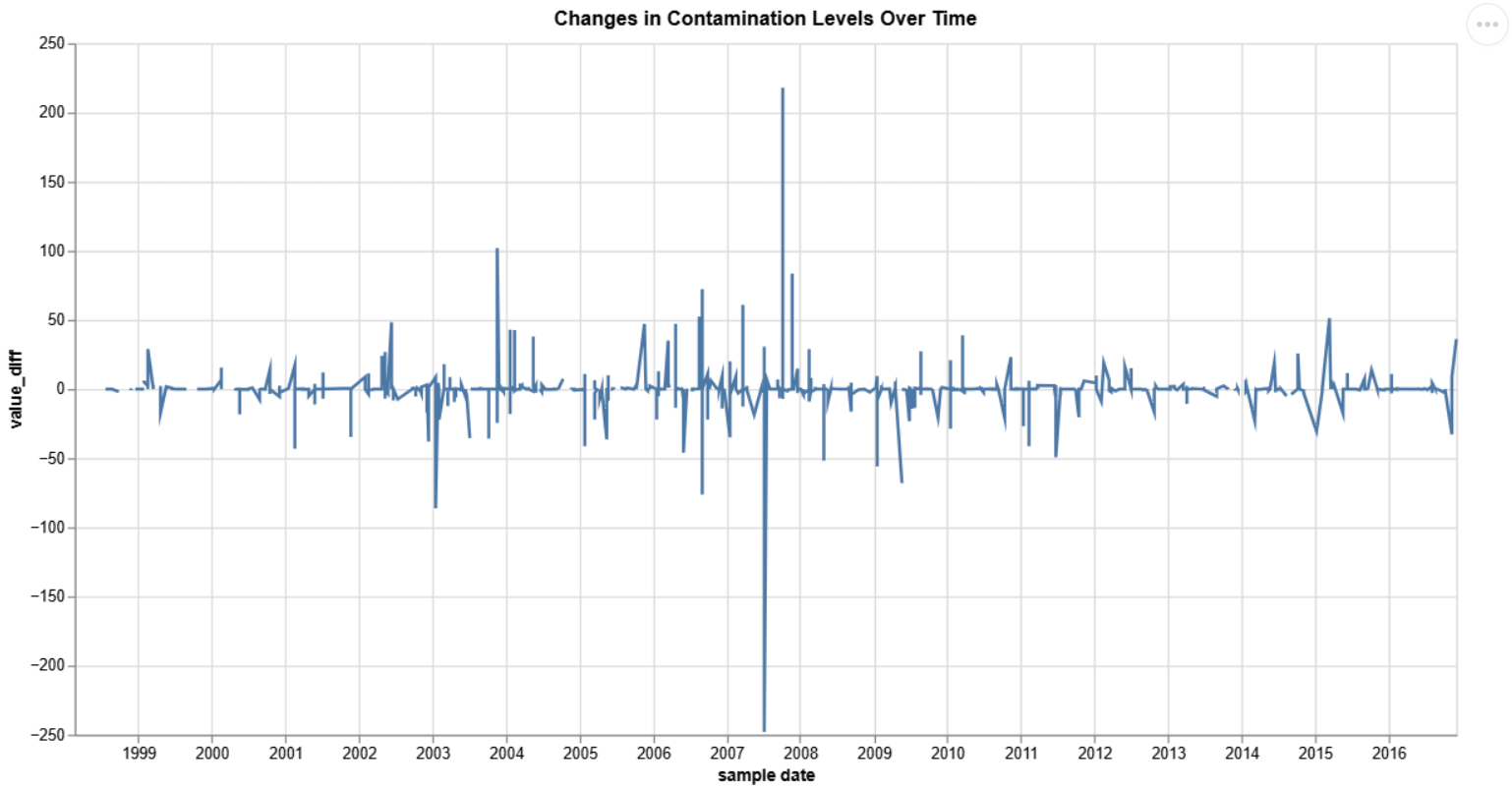
**Temporal Evolution of Contamination Levels Across Locations**

In [28]:

```
outliers_df['sample date'] = pd.to_datetime(outliers_df['sample date'])

location_dropdown = alt.binding_select(options=outliers_df['location'].unique(), name='Select Location:')
selected_location = alt.selection_single(fields=['location'], bind=location_dropdown,
                                         value=outliers_df['location'].unique()[0])

outliers_df['value_diff'] = outliers_df.groupby(['location', 'measure'])['value'].diff()

diff_chart = alt.Chart(outliers_df).mark_line().encode(
    x='sample date:T',
    y='value_diff:Q',
    tooltip=['sample date:T', 'value_diff:Q', 'location:N']
).properties(
    width=900,
    height=450,
```

```
    title='Changes in Contamination Levels Over Time'
).add_selection(
    selected_location
).transform_filter(
    selected_location
)

diff_chart
```

Out[28]:




Select Location: Chai

**OBSERVATIONS**

- Over the years, Achara, Decha, and Tansanee show relatively stable contamination levels with no drastic changes.
- Boonsri, Chai, Kannika, and Sakda, on the other hand, exhibit diverse and fluctuating contamination levels across all years, suggesting a dynamic pattern of changes in contamination over time.
- Busarakhan, Kohsoom, and Somchair experience surges and dips in contamination levels, but these fluctuations are not as drastic compared to the variations observed in the second group.

**Identification of Most Prominent Chemical Anomalies Across Locations - Frequency**

In [29]:

```
outliers_summary = outliers_df.groupby(['location', 'measure']).agg(
    frequency=('value', 'count'),
```

```
    max_value=('value', 'max')
).reset_index()

most_prominent_chemicals = outliers_summary.loc[outliers_summary.groupby('location')['frequency'].idxmax()]

most_prominent_chemicals
```

Out[29]:

| | location | measure | frequency | max_value |
|---|---|---|---|---|
| **0** | Achara | Ammonium | 8 | 0.552000 |
| **36** | Boonsri | Ammonium | 49 | 0.760000 |
| **141** | Busarakhan | Total phosphorus | 15 | 0.243000 |
| **186** | Chai | Nitrites | 62 | 0.075000 |
| **230** | Decha | Orthophosphate-phosphorus | 9 | 0.118000 |
| **251** | Kannika | Chlorides | 33 | 63.200001 |
| **324** | Kohsoom | Magnesium | 7 | 34.000000 |
| **362** | Sakda | Chlorides | 36 | 68.800003 |
| **444** | Somchair | Orthophosphate-phosphorus | 12 | 0.145000 |
| **462** | Tansanee | Ammonium | 5 | 0.755000 |

**Identification of Most Prominent Chemical Anomalies Across Locations - Max Value**

In [30]:

```
outliers_summary = outliers_df.groupby(['location', 'measure']).agg(
    frequency=('value', 'count'),
    max_value=('value', 'max')
).reset_index()

most_prominent_chemicals_max_value = outliers_summary.loc[outliers_summary.groupby('location')
                                                          ['max_value'].idxmax()]

most_prominent_chemicals_max_value
```

Out[30]:

| | location | measure | frequency | max_value |
|---|---|---|---|---|
| **27** | Achara | Total hardness | 1 | 160.10 |
| **82** | Boonsri | Total dissolved salts | 18 | 364.00 |
| **139** | Busarakhan | Total hardness | 4 | 262.00 |
| **199** | Chai | Total dissolved salts | 18 | 391.00 |
| **213** | Decha | Bicarbonates | 1 | 207.50 |
| **287** | Kannika | Total dissolved salts | 18 | 396.00 |
| **338** | Kohsoom | Total dissolved salts | 3 | 436.00 |
| **396** | Sakda | Total dissolved salts | 25 | 429.80 |
| **452** | Somchair | Total dissolved salts | 1 | 432.00 |
| **464** | Tansanee | Bicarbonates | 1 | 125.05 |

# *QUESTION 02*

---

**Describe any data quality and uncertain issues, such as**

i. missing data, outliers

ii. change in collection frequency, and

iii. unrealistic values (e.g. water temperature higher than 100 degrees).

**Part i : Missing Values**

---

In [31]:
```
# Assessing the proportion of missing values in the wildlife_dataset.
# The dataframe displays the proportion of missing values within the dataset.

missing_values = wildlife_dataset.isnull().sum()

missing_data_percentage = missing_values.sort_values(ascending=False) * 100 / len(wildlife_dataset)

missing_data_table = pd.DataFrame({
    'Attribute': missing_data_percentage.index,
    'Missing Data Percentage': missing_data_percentage.values
})

print("\n\nMissing Values Proportion in dataset\n\n")
missing_data_table
```

```
Missing Values Proportion in dataset
```

Out[31]:

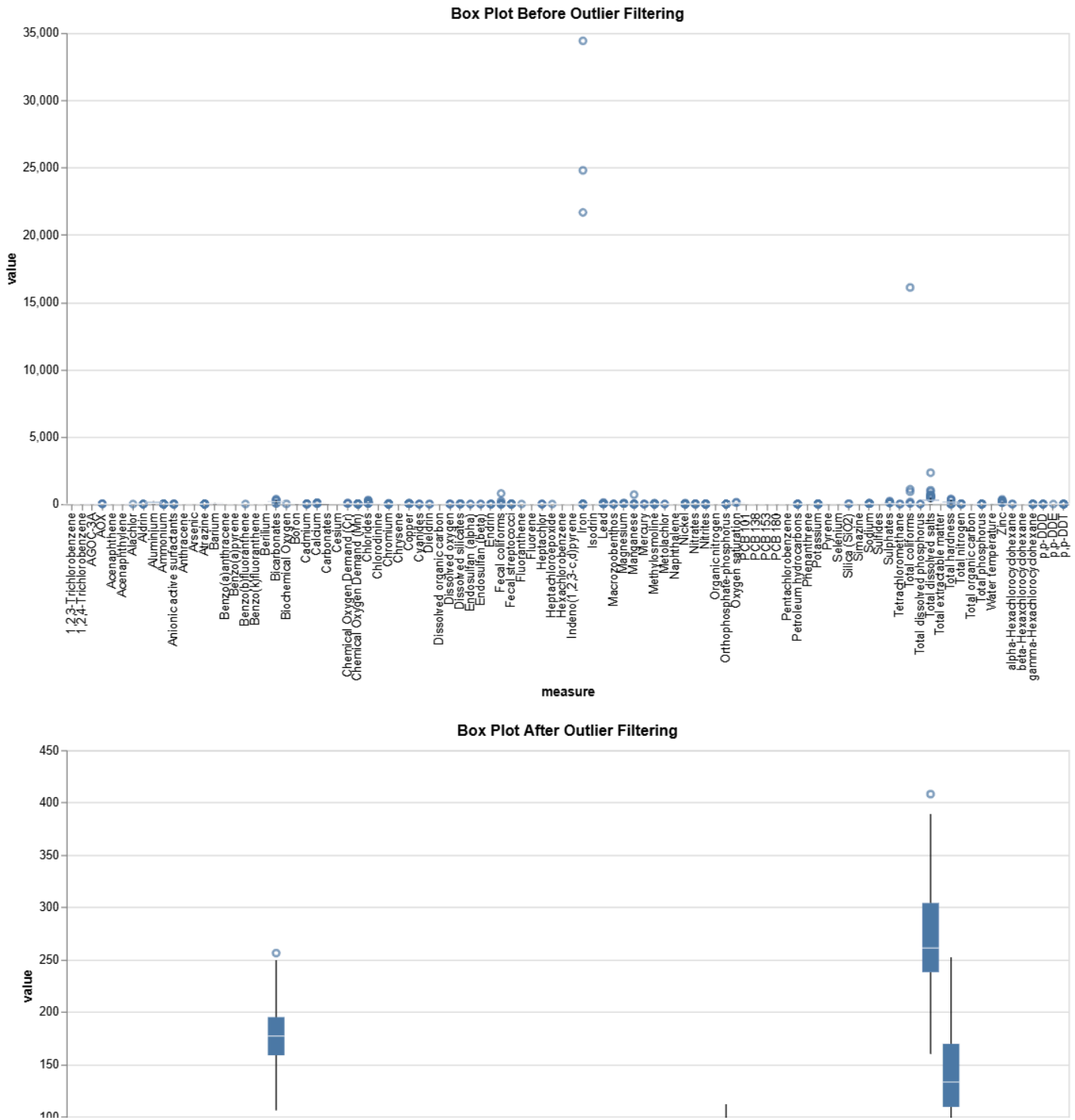
| | Attribute | Missing Data Percentage |
|---|---|---|
| **0** | id | 0.0 |
| **1** | value | 0.0 |
| **2** | location | 0.0 |
| **3** | sample date | 0.0 |
| **4** | measure | 0.0 |

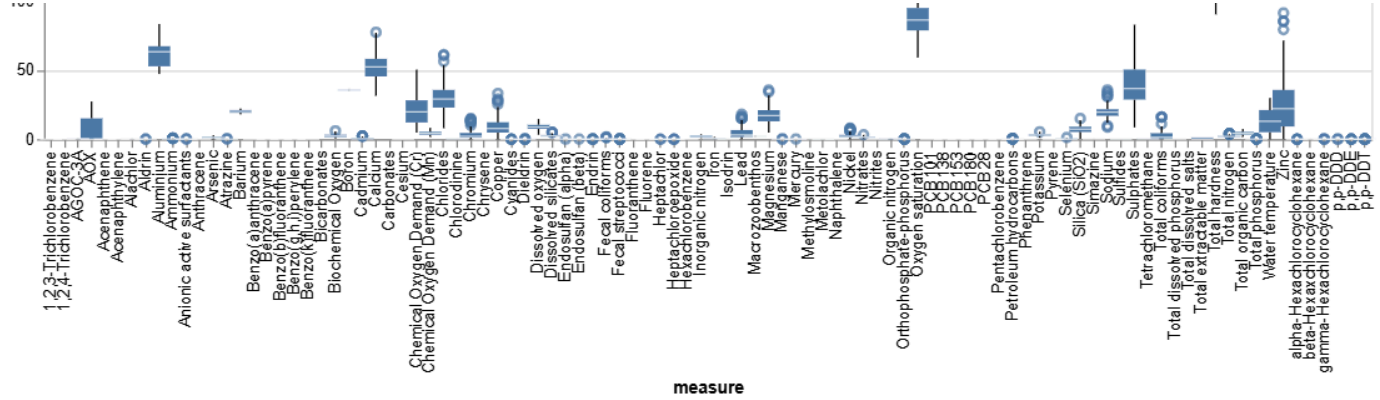
**Part i : Outliers**

In [38]:

```
wildlife_dataset['measure'] = wildlife_dataset['measure'].astype(str)
filtered_dataset['measure'] = filtered_dataset['measure'].astype(str)

# Sample the data for plotting
sampled_wildlife = wildlife_dataset.sample(n=5000, random_state=42)
sampled_filtered = filtered_dataset.sample(n=5000, random_state=42)

boxplot_before = alt.Chart(sampled_wildlife).mark_boxplot().encode(
    x='measure:N',
    y='value:Q',
    tooltip=['measure:N', 'value:Q']
).properties(
    width=850,
    height=400,
    title='Box Plot Before Outlier Filtering'
)

boxplot_after = alt.Chart(sampled_filtered).mark_boxplot().encode(
    x='measure:N',
    y='value:Q',
    tooltip=['measure:N', 'value:Q']
).properties(
    width=850,
    height=400,
    title='Box Plot After Outlier Filtering'
)

alt.vconcat(boxplot_before, boxplot_after)
```

Out[38]:


Box Plot Before Outlier Filtering
value
35,000
30,000
25,000
20,000
15,000
10,000
5,000
0
measure
1,2,3-Trichlorobenzene
1,2,4-Trichlorobenzene
AGOC-3A
AOX
Acenaphthene
Acenaphthylene
Alachlor
Aldrin
Aluminium
Ammonium
Anionic active surfactants
Anthracene
Arsenic
Atrazine
Barium
Benzo(a)anthracene
Benzo(b)fluoranthene
Benzo(k)fluoranthene
Beryllium
Bicarbonates
Biochemical Oxygen
Boron
Cadmium
Calcium
Carbonates
Cesium
Chemical Oxygen Demand (Cr)
Chemical Oxygen Demand (Mn)
Chlorides
Chlorodinine
Chrysene
Copper
Cyanides
Dieldrin
Dissolved organic carbon
Dissolved oxygen
Dissolved silicates
Endosulfan (alpha)
Endosulfan (beta)
Endrin
Fecal coliforms
Fecal streptococci
Fluoranthene
Fluorene
Heptachlor
Heptachlorepoxide
Hexachlorobenzene
Indeno(1,2,3-c,d)pyrene
Iron
Isodrin
Lead
Macrozoobenthos
Magnesium
Manganese
Mercury
Methylosmoline
Metolachlor
Naphthalene
Nickel
Nitrates
Nitrites
Organic nitrogen
Orthophosphate-phosphorus
Oxygen saturation
PCB 101
PCB 138
PCB 153
PCB 180
Pentachlorobenzene
Petroleum hydrocarbons
Phenanthrene
Potassium
Pyrene
Selenium
Silica (SiO2)
Simazine
Sodium
Sulphates
Tetrachloromethane
Total coliforms
Total dissolved phosphorus
Total dissolved salts
Total extractable matter
Total hardness
Total nitrogen
Total organic carbon
Total phosphorus
Water temperature
Zinc
alpha-Hexachlorocyclohexane
beta-Hexaxchlorocyclohexane
gamma-Hexachlorocyclohexane
p,p-DDD
p,p-DDE
p,p-DDT
Box Plot After Outlier Filtering
value
450
400
350
300
250
200
150
100



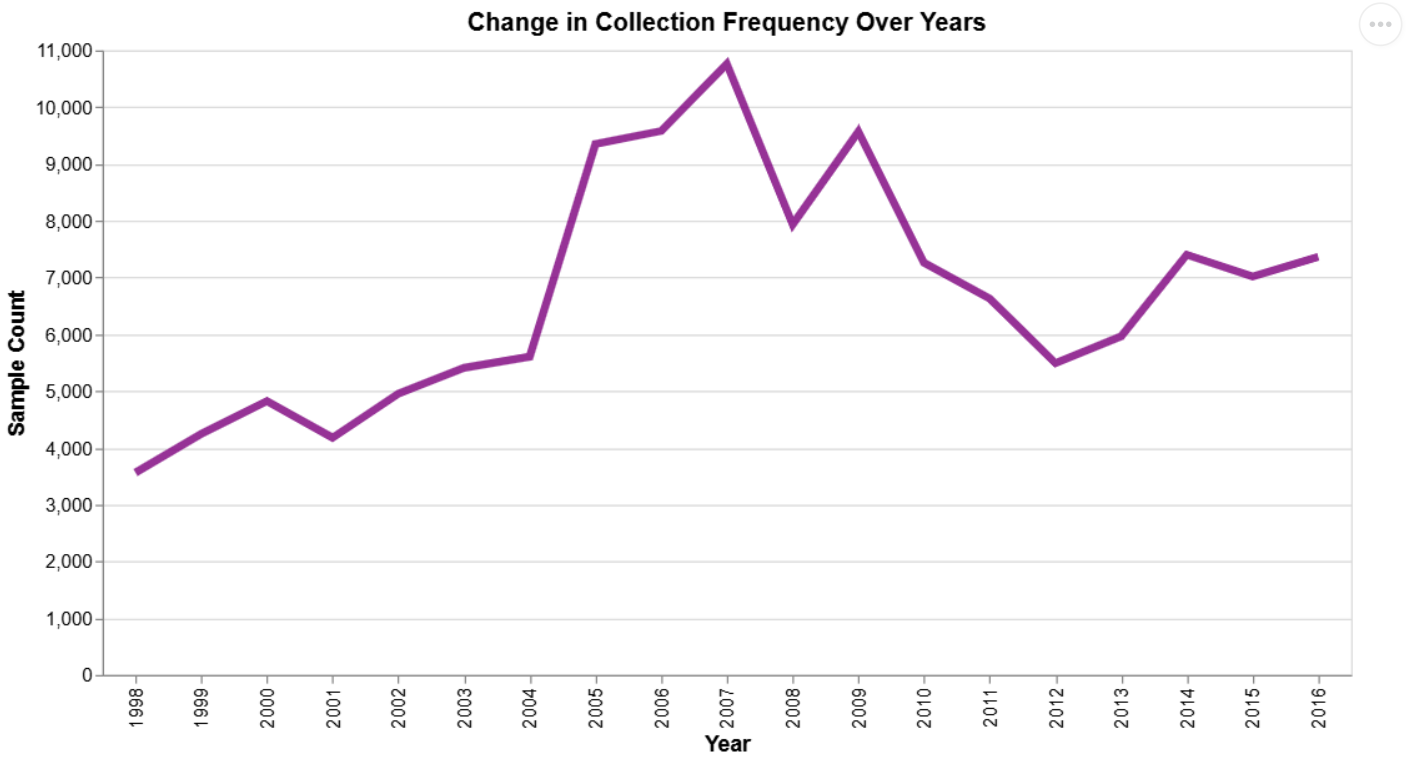

**Part ii : Change in collection Frequency**

In [33]:
```
# Counting the number of samples for each year
sample_counts_by_year = filtered_dataset.groupby('year').size().reset_index(name='count')
```

In [34]:
```
line_plot_yearly = alt.Chart(sample_counts_by_year).mark_line(
    color='purple',
    opacity=0.8,
    size=5
).encode(
    x=alt.X('year:N', title='Year'),
    y=alt.Y('count:Q', title='Sample Count'),
    tooltip=[
        alt.Tooltip('year:N', title='Year'),
        alt.Tooltip('count:Q', title='Sample Count', format=',')
    ]
).properties(
    width=800,
    height=400,
    title='Change in Collection Frequency Over Years'
).configure_title(
    fontSize=16,
    fontWeight='bold'
).configure_axis(
    labelFontSize=12,
    titleFontSize=14
)

sample_counts_by_year['YoY_change'] = sample_counts_by_year['count'].pct_change() * 100

print(sample_counts_by_year)
print("\n\n")

line_plot_yearly
```

| | year | count | YoY_change |
|---|---|---|---|
| 0 | 1998 | 3564 | NaN |
| 1 | 1999 | 4246 | 19.135802 |
| 2 | 2000 | 4822 | 13.565709 |
| 3 | 2001 | 4178 | -13.355454 |
| 4 | 2002 | 4953 | 18.549545 |
| 5 | 2003 | 5411 | 9.246921 |
| 6 | 2004 | 5606 | 3.603770 |
| 7 | 2005 | 9349 | 66.767749 |
| 8 | 2006 | 9581 | 2.481549 |
| 9 | 2007 | 10763 | 12.336917 |
| 10 | 2008 | 7935 | -26.275202 |
| 11 | 2009 | 9567 | 20.567108 |
| 12 | 2010 | 7259 | -24.124595 |
| 13 | 2011 | 6625 | -8.733985 |
| 14 | 2012 | 5490 | -17.132075 |
| 15 | 2013 | 5963 | 8.615665 |
| 16 | 2014 | 7400 | 24.098608 |
| 17 | 2015 | 7018 | -5.162162 |
| 18 | 2016 | 7366 | 4.958678 |

Out[34]:

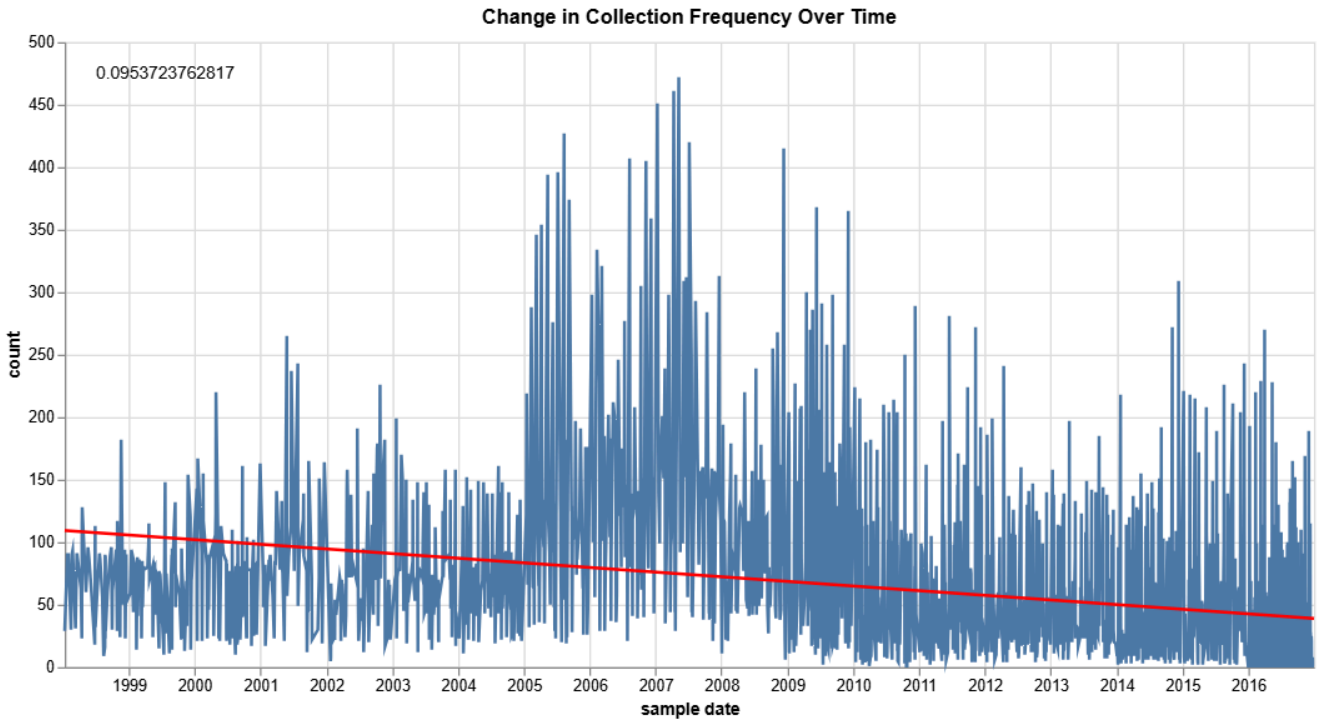



**OBSERVATIONS:**

- The frequency of sample collection exhibits a consistent upward trend from 1998 to 2005, interrupted by a 15% drop in 2001 compared to the preceding year. Throughout this period, there is a steady increase in the number of samples collected annually.
- Around 2005, there is a noticeable surge in collection frequency, peaking in 2006 and 2007. However, in the subsequent year, 2008, there is a significant decline in the number of samples collected, registering a decrease of approximately 26%, indicating a period of fluctuation.
- From 2009 onward, fluctuations in sample collection become evident. There are instances of notable increases, ranging from 22% to 24%, while in other cases, there are decreases ranging from 5% to 23%.

In [35]:
```
# Assuming wildlife_dataset is already defined and formatted correctly
filtered_dataset['sample date'] = pd.to_datetime(filtered_dataset['sample date'], format='%d-%b-%y')
sample_counts = filtered_dataset.groupby('sample date').size().reset_index(name='count')

# Create the base scatter plot
scatter_plot = alt.Chart(sample_counts).mark_line().encode(
    x='sample date:T',
    y='count:Q'
).properties(
    width=800,
    height=400,
    title='Change in Collection Frequency Over Time'
)

# Create the regression line
regression_line = scatter_plot.transform_regression(
    'sample date', 'count'
).mark_line(color='red')

# Calculate regression parameters
params = alt.Chart(sample_counts).transform_regression(
    'sample date', 'count', params=True
).mark_text(
    align='left'
).encode(
    x=alt.value(20),  # Position for the text
    y=alt.value(20),  # Position for the text
    text=alt.Text('rSquared:Q')  # Display R-squared
)

# Combine the scatter plot, regression line, and parameters text
combined_chart = scatter_plot + regression_line + params

combined_chart
```

Out[35]:




The trend line depicted in the graph signifies the overall trajectory or inclination of the data as it evolves over time. Specifically, it indicates a declining trend in the frequency of collection over the observed period. The R-squared value of 0.09 indicates a weak fit between the trend line and the actual data points in the time series, implying that only approximately 9 perrcent of the variability in the time series values can be accounted for by the trend line. This signifies a limited degree of correlation (not zero correlation) between the time variable and the observed values.

**iii. unrealistic values (e.g. water temperature higher than 100 degrees).**

In [36]:

```
def calculate_statistics(df, measure):
    subset = df[df['measure'] == measure]
    min_value = subset['value'].min()
    max_value = subset['value'].max()
    q1 = subset['value'].quantile(0.25)
    q2 = subset['value'].median()
    q3 = subset['value'].quantile(0.75)

    return pd.Series({
        'measure': measure,
        'min_value': min_value,
        'q1': q1,
        'q2': q2,
        'q3': q3,
        'max_value': max_value,
    })
unique_measures = filtered_dataset['measure'].unique()

measure_statistics = pd.DataFrame(columns=['measure', 'min_value', 'max_value', 'q1', 'q2', 'q3'])
```

```
for measure in unique_measures:
    measure_stats = calculate_statistics(filtered_dataset, measure)
    measure_statistics = measure_statistics.append(measure_stats, ignore_index=True)

measure_statistics
```

Out[36]:

| | measure | min_value | max_value | q1 | q2 | q3 |
|---|---|---|---|---|---|---|
| **0** | Water temperature | 0.000 | 36.4000 | 6.50000 | 14.0000 | 21.50000 |
| **1** | Dissolved oxygen | 3.000 | 14.8500 | 7.43250 | 8.8000 | 10.40000 |
| **2** | Ammonium | 0.000 | 0.7900 | 0.08800 | 0.1700 | 0.30555 |
| **3** | Nitrites | 0.000 | 0.0760 | 0.01530 | 0.0230 | 0.03400 |
| **4** | Nitrates | 0.000 | 3.2300 | 0.98300 | 1.3900 | 1.85700 |
| **...** | ... | ... | ... | ... | ... | ... |
| **101** | Boron | 33.000 | 41.0000 | 35.00000 | 37.0000 | 38.00000 |
| **102** | AGOC-3A | 0.000 | 0.7090 | 0.06500 | 0.2735 | 0.31600 |
| **103** | Methylosmoline | 0.250 | 0.4600 | 0.27650 | 0.3080 | 0.32800 |
| **104** | Chlorodinine | 0.000 | 0.4360 | 0.04325 | 0.2250 | 0.25800 |
| **105** | Total dissolved phosphorus | 0.008 | 0.1306 | 0.04400 | 0.0590 | 0.07700 |

106 rows × 6 columns

In [37]:
```
unrealistic_dataset = filtered_dataset[filtered_dataset['measure'] == 'Water temperature']
unrealistic_dataset = unrealistic_dataset[(filtered_dataset['value'] > 100)]

print(unrealistic_dataset)
```

```
Empty DataFrame
Columns: [id, value, location, sample date, measure, year, month, day, location_filled, contamination_change, z_score]
Index: []
```

No such unrealistic value for water temperature higher than 100 degrees

## Conclusion

Across various locations and over different years, the predominant contributors to chemical contamination consistently include total dissolved salts, bicarbonates, and zinc. Notably, the specific pattern of contribution remains somewhat consistent across the diverse locations. In Busarakhan and Kohsoom, as well as in Somchair, the primary contributors are dissolved salts and bicarbonates.For Boonsri, Kannika, and Sakda, the major contributors are dissolved salts and zinc. In the case of Decha, total dissolved salts emerge as the leading contributor. Lastly, in Tansanee, bicarbonates are identified as the primary contributors to chemical contamination. The potential link between chemical contamination (specifically from total dissolved salts, bicarbonates, and zinc) and bird population reduction could be multifaceted. Elevated levels of total dissolved salts, bicarbonates, and zinc in water sources may have impacted the availability and suitability of nesting sites, feeding grounds, and overall habitat conditions.Certain concentrations of zinc, in particular, can be toxic to birds. Exposure to high levels of zinc through contaminated water sources could have lead to detrimental health effects in birds which includes developmental issues, and overall population decline. Along with that birds often rely on specific water sources during their migration or movement patterns. If these water sources are contaminated, it had affected the availability of clean water during critical times, thus impacting the well-being and overall deduction in birds population.