```
In [1]: import pandas as pd
```

```
In [3]: df = pd.read_csv(r"D:\\Python\\Project\\Your Orders- Amazon\\Retail.OrderHis
```

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

Out[4]:

	Website	Order ID	Order Date	Purchase Order Number	Currency	Unit Price	Unit Price Tax	Shipping Charge	Tota Discount
0	Amazon.in	405-3350864-5032300	2023-06-04T04:39:00Z	Not Applicable	INR	1100.84	198.16	0.00	
1	Amazon.in	407-3158398-9699568	2023-03-02T13:04:50Z	Not Applicable	INR	338.14	60.86	0.00	
2	Amazon.in	407-7792390-7685951	2023-02-14T14:59:55Z	Not Applicable	INR	2160.16	388.84	0.00	
3	Amazon.in	403-3016875-2779536	2023-01-15T14:38:37Z	Not Applicable	INR	140.68	25.32	0.00	
4	Amazon.in	407-2079257-0509922	2023-01-05T07:21:09Z	Not Applicable	INR	846.62	152.38	0.00	
...	...	...	...	...	...	...	...	...	
70	Amazon.in	403-0675650-5133167	2019-05-22T09:42:17Z	Not Applicable	INR	592.38	106.62	6.10	'-5
71	Amazon.in	402-0445900-8806716	2019-04-21T09:01:59Z	Not Applicable	INR	7107.14	852.86	8.57	
72	Amazon.in	403-3522727-4762706	2019-03-16T09:09:13Z	Not Applicable	INR	288.14	51.86	6.10	
73	Amazon.in	171-4642161-2239500	2018-10-10T05:49:12Z	Not Applicable	INR	4321.18	777.82	15.26	'-
74	Amazon.in	405-6012944-3268316	2018-06-02T06:41:39Z	Not Applicable	INR	507.63	91.37	0.00	

75 rows × 27 columns

Loading [MathJax]/extensions/Safe.js

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

Out[5]:

	Website	Order ID	Order Date	Purchase Order Number	Currency	Unit Price	Unit Price Tax	Shipping Charge	Total Discounts
0	Amazon.in	405-3350864-5032300	2023-06-04T04:39:00Z	Not Applicable	INR	1100.84	198.16	0.0	C
1	Amazon.in	407-3158398-9699568	2023-03-02T13:04:50Z	Not Applicable	INR	338.14	60.86	0.0	C
2	Amazon.in	407-7792390-7685951	2023-02-14T14:59:55Z	Not Applicable	INR	2160.16	388.84	0.0	C
3	Amazon.in	403-3016875-2779536	2023-01-15T14:38:37Z	Not Applicable	INR	140.68	25.32	0.0	C
4	Amazon.in	407-2079257-0509922	2023-01-05T07:21:09Z	Not Applicable	INR	846.62	152.38	0.0	C

5 rows × 27 columns

In [6]:
```
df.info()
```

Loading [MathJax]/extensions/Safe.js

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 75 entries, 0 to 74
Data columns (total 27 columns):
 #   Column                            Non-Null Count  Dtype  
---  ------                            --------------  -----  
 0   Website                           75 non-null     object 
 1   Order ID                          75 non-null     object 
 2   Order Date                        75 non-null     object 
 3   Purchase Order Number             75 non-null     object 
 4   Currency                          75 non-null     object 
 5   Unit Price                        75 non-null     float64
 6   Unit Price Tax                    75 non-null     float64
 7   Shipping Charge                   75 non-null     float64
 8   Total Discounts                   75 non-null     object 
 9   Total Owed                        75 non-null     float64
 10  Shipment Item Subtotal            75 non-null     object 
 11  Shipment Item Subtotal Tax        75 non-null     object 
 12  ASIN                              75 non-null     object 
 13  Product Condition                 75 non-null     object 
 14  Quantity                          75 non-null     int64  
 15  Payment Instrument Type           75 non-null     object 
 16  Order Status                      75 non-null     object 
 17  Shipment Status                   75 non-null     object 
 18  Ship Date                         75 non-null     object 
 19  Shipping Option                   75 non-null     object 
 20  Shipping Address                  75 non-null     object 
 21  Billing Address                   75 non-null     object 
 22  Carrier Name & Tracking Number    75 non-null     object 
 23  Product Name                      75 non-null     object 
 24  Gift Message                      75 non-null     object 
 25  Gift Sender Name                  75 non-null     object 
 26  Gift Recipient Contact Details    75 non-null     object 
dtypes: float64(4), int64(1), object(22)
memory usage: 15.9+ KB
```

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

Loading [MathJax]/extensions/Safe.js

```
Out[7]: Website                               object
        Order ID                              object
        Order Date                            object
        Purchase Order Number                 object
        Currency                              object
        Unit Price                           float64
        Unit Price Tax                       float64
        Shipping Charge                      float64
        Total Discounts                       object
        Total Owed                           float64
        Shipment Item Subtotal                object
        Shipment Item Subtotal Tax            object
        ASIN                                  object
        Product Condition                     object
        Quantity                               int64
        Payment Instrument Type               object
        Order Status                          object
        Shipment Status                       object
        Ship Date                             object
        Shipping Option                       object
        Shipping Address                      object
        Billing Address                       object
        Carrier Name & Tracking Number        object
        Product Name                          object
        Gift Message                          object
        Gift Sender Name                      object
        Gift Recipient Contact Details        object
        dtype: object
```

```
In [8]: df.describe()
```

Out[8]:

	Unit Price	Unit Price Tax	Shipping Charge	Total Owed	Quantity
count	75.000000	75.000000	75.000000	75.000000	75.000000
mean	2349.042133	423.221867	3.264133	2664.373333	0.960000
std	5152.001891	1113.187708	11.833787	6224.611626	0.256799
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	283.475000	42.805000	0.000000	312.050000	1.000000
50%	592.380000	76.120000	0.000000	649.000000	1.000000
75%	2032.630000	292.510000	0.000000	1999.000000	1.000000
max	25780.460000	7218.540000	69.000000	32999.000000	2.000000

```
In [12]: df.shape
```

```
Out[12]: (75, 27)
```

```
In [10]: df['Order Date'] = pd.to_datetime(df['Order Date'])
```

```
In [11]: df['Order Date']
```

Loading [MathJax]/extensions/Safe.js

```
Out[11]: 0    2023-06-04 04:39:00+00:00
         1    2023-03-02 13:04:50+00:00
         2    2023-02-14 14:59:55+00:00
         3    2023-01-15 14:38:37+00:00
         4    2023-01-05 07:21:09+00:00
                         ...
         70   2019-05-22 09:42:17+00:00
         71   2019-04-21 09:01:59+00:00
         72   2019-03-16 09:09:13+00:00
         73   2018-10-10 05:49:12+00:00
         74   2018-06-02 06:41:39+00:00
         Name: Order Date, Length: 75, dtype: datetime64[ns, UTC]
```

```
In [13]: df['Total Owed'].sum()
```

```
Out[13]: 199828.0
```

```
In [14]: df['Total Owed'].mean()
```

```
Out[14]: 2664.3733333333334
```

```
In [15]: df['Total Owed'].median()
```

```
Out[15]: 649.0
```

```
In [16]: # max spent amount

         df['Total Owed'].max()
```

```
Out[16]: 32999.0
```

```
In [17]: df['Total Owed'].min()
```

```
Out[17]: 0.0
```

```
In [18]: df['Unit Price Tax'].sum()
```

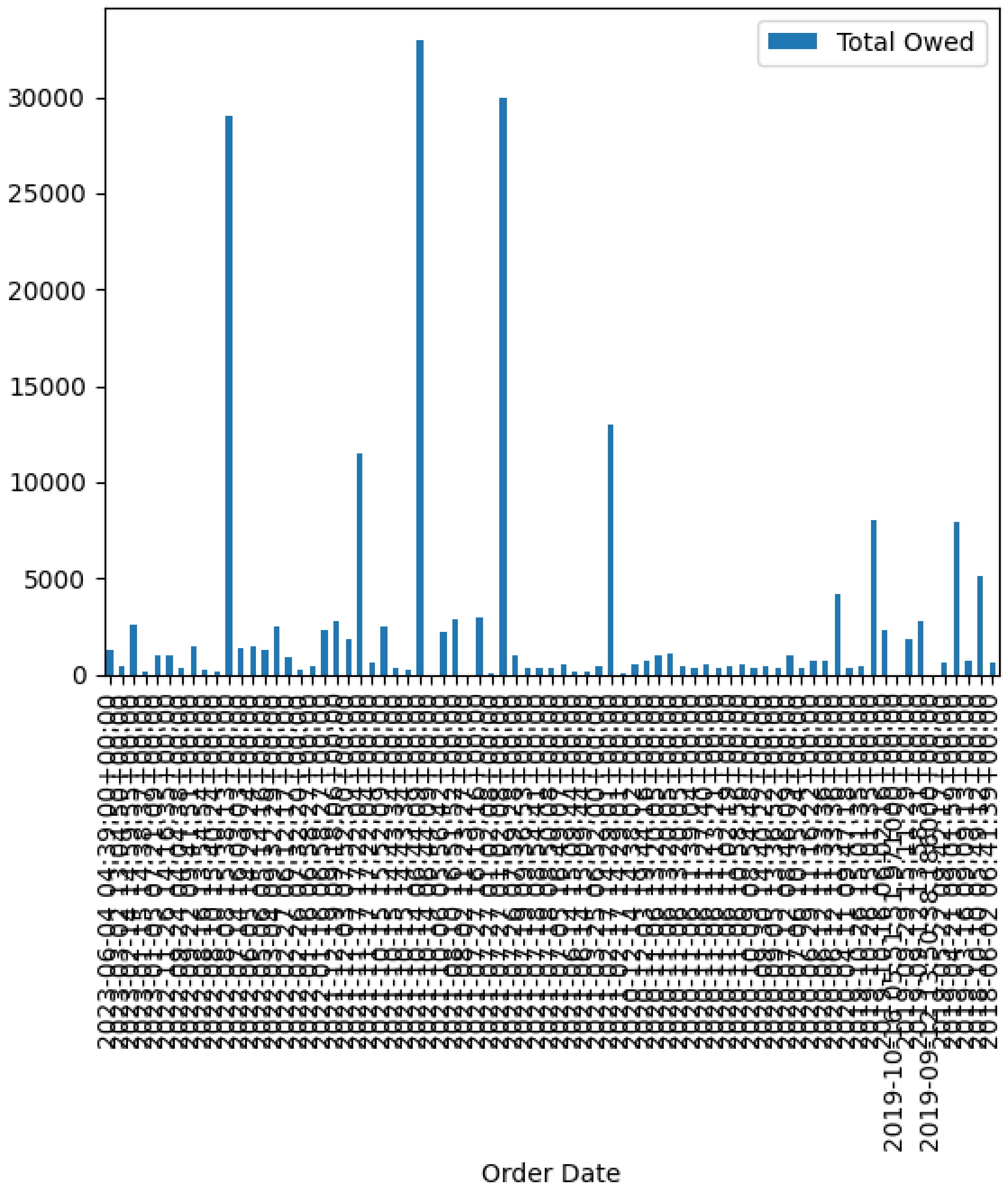
```
Out[18]: 31741.64
```

```
In [19]: import matplotlib as mpl
```

```
In [20]: df.plot.bar(x= 'Order Date', y= 'Total Owed', rot = 90)
```

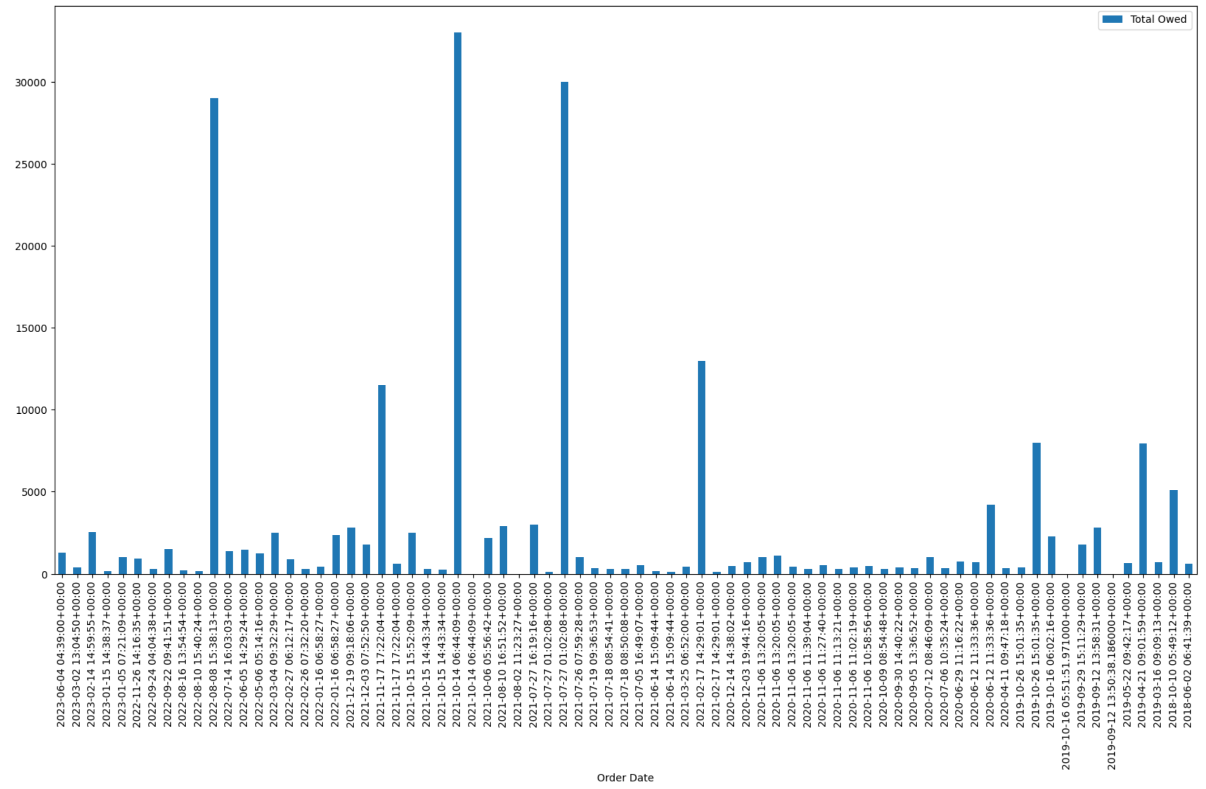
```
Out[20]: <Axes: xlabel='Order Date'>
```

Loading [MathJax]/extensions/Safe.js



```
In [21]: df.plot.bar(x= 'Order Date', y= 'Total Owed', rot = 90, figsize= (20,10))

Out[21]: <Axes: xlabel='Order Date'>
```



```
In [22]: orders_per_day= df.groupby('Order Date').sum()['Total Owed']
```

```
C:\Users\kruna\AppData\Local\Temp\ipykernel_1624\1454309424.py:1: FutureWar
ning: The default value of numeric_only in DataFrameGroupBy.sum is deprecat
ed. In a future version, numeric_only will default to False. Either specify
numeric_only or select only columns which should be valid for the function.
  orders_per_day= df.groupby('Order Date').sum()['Total Owed']
```

```
In [23]: orders_per_day.head(60)
```

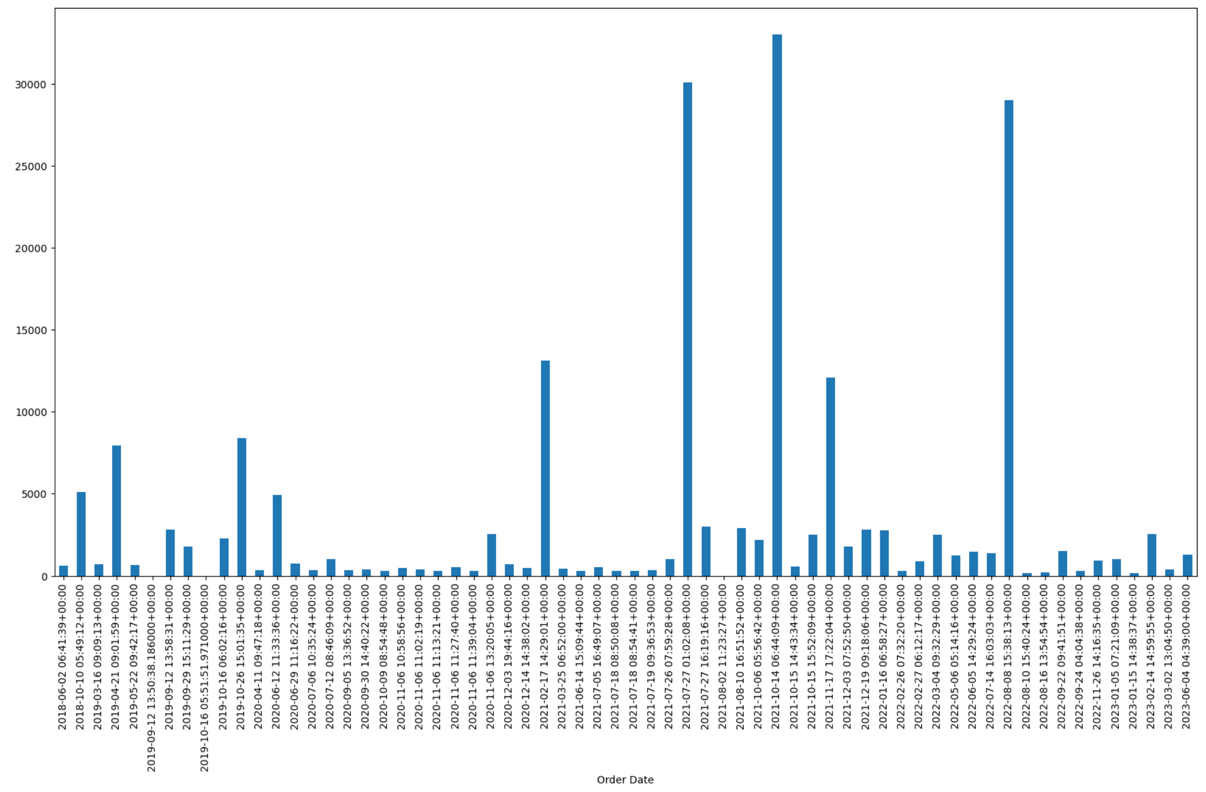
Loading [MathJax]/extensions/Safe.js

```
Out[23]: Order Date
         2018-06-02 06:41:39+00:00               599.0
         2018-10-10 05:49:12+00:00              5098.0
         2019-03-16 09:09:13+00:00               680.0
         2019-04-21 09:01:59+00:00              7960.0
         2019-05-22 09:42:17+00:00               649.0
         2019-09-12 13:50:38.186000+00:00          0.0
         2019-09-12 13:58:31+00:00              2799.0
         2019-09-29 15:11:29+00:00              1799.0
         2019-10-16 05:51:51.971000+00:00          0.0
         2019-10-16 06:02:16+00:00              2298.0
         2019-10-26 15:01:35+00:00              8399.0
         2020-04-11 09:47:18+00:00               329.0
         2020-06-12 11:33:36+00:00              4920.0
         2020-06-29 11:16:22+00:00               749.0
         2020-07-06 10:35:24+00:00               349.0
         2020-07-12 08:46:09+00:00               999.0
         2020-09-05 13:36:52+00:00               349.0
         2020-09-30 14:40:22+00:00               399.0
         2020-10-09 08:54:48+00:00               295.0
         2020-11-06 10:58:56+00:00               499.0
         2020-11-06 11:02:19+00:00               395.0
         2020-11-06 11:13:21+00:00               299.0
         2020-11-06 11:27:40+00:00               516.0
         2020-11-06 11:39:04+00:00               319.0
         2020-11-06 13:20:05+00:00              2551.0
         2020-12-03 19:44:16+00:00               699.0
         2020-12-14 14:38:02+00:00               499.0
         2021-02-17 14:29:01+00:00             13099.0
         2021-03-25 06:52:00+00:00               449.0
         2021-06-14 15:09:44+00:00               304.0
         2021-07-05 16:49:07+00:00               529.0
         2021-07-18 08:50:08+00:00               305.1
         2021-07-18 08:54:41+00:00               305.1
         2021-07-19 09:36:53+00:00               339.0
         2021-07-26 07:59:28+00:00               999.0
         2021-07-27 01:02:08+00:00             30099.0
         2021-07-27 16:19:16+00:00              2999.0
         2021-08-02 11:23:27+00:00                 0.0
         2021-08-10 16:51:52+00:00              2899.0
         2021-10-06 05:56:42+00:00              2199.0
         2021-10-14 06:44:09+00:00             32999.0
         2021-10-15 14:43:34+00:00               558.0
         2021-10-15 15:52:09+00:00              2499.0
         2021-11-17 17:22:04+00:00             12098.0
         2021-12-03 07:52:50+00:00              1799.0
         2021-12-19 09:18:06+00:00              2799.0
         2022-01-16 06:58:27+00:00              2768.0
         2022-02-26 07:32:20+00:00               288.8
         2022-02-27 06:12:17+00:00               899.0
         2022-03-04 09:32:29+00:00              2499.0
         2022-05-06 05:14:16+00:00              1249.0
         2022-06-05 14:29:24+00:00              1450.0
         2022-07-14 16:03:03+00:00              1399.0
         2022-08-08 15:38:13+00:00             28998.0
Loading [MathJax]/extensions/Safe.js :40:24+00:00               169.0
```

```
2022-08-16 13:54:54+00:00          225.0
2022-09-22 09:41:51+00:00         1499.0
2022-09-24 04:04:38+00:00          299.0
2022-11-26 14:16:35+00:00          949.0
2023-01-05 07:21:09+00:00          999.0
Name: Total Owed, dtype: float64
```

In [25]:
```
orders_per_day.plot.bar(figsize=(20,10))
```

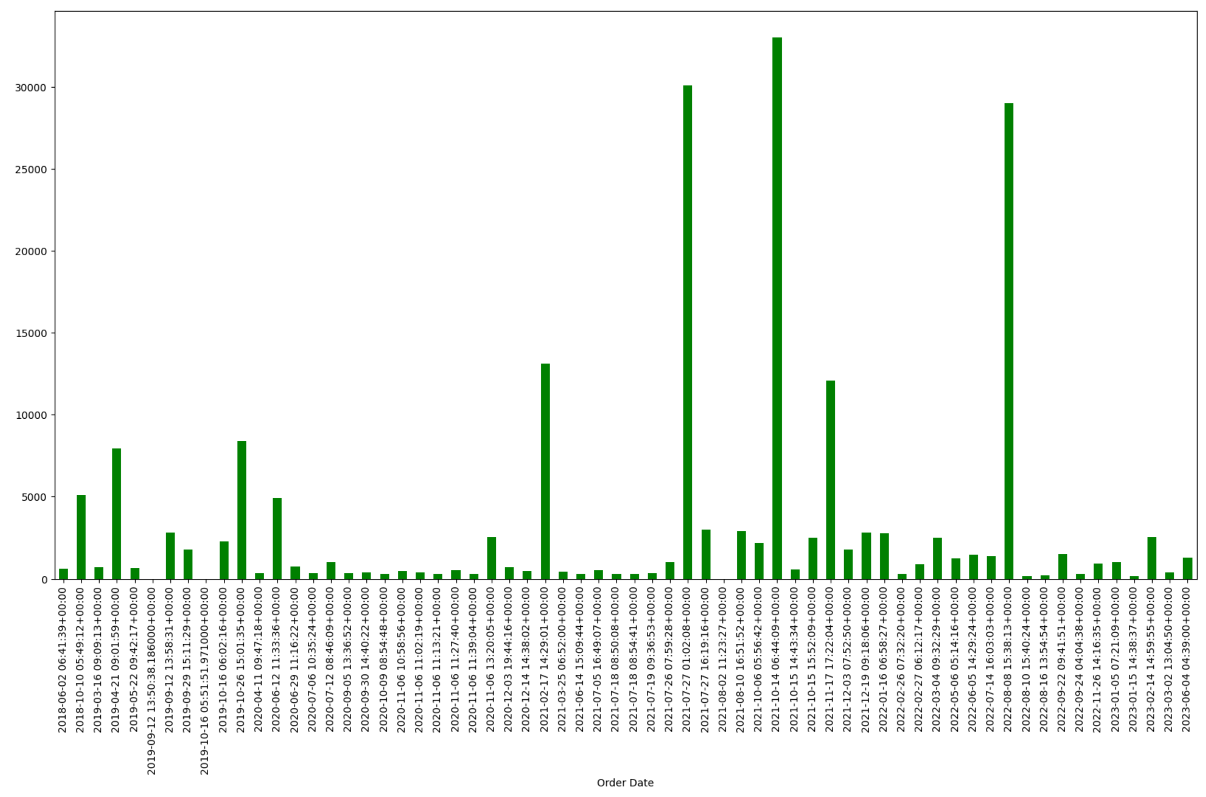
Out[25]:
```
<Axes: xlabel='Order Date'>
```



In [26]:
```
orders_per_day.plot.bar(figsize=(20,10), color = "green")
```

Out[26]:
```
<Axes: xlabel='Order Date'>
```

Loading [MathJax]/extensions/Safe.js



```
In [27]: df['Year'] = df['Order Date'].dt.year
```

```
In [28]: df['Year']
```

```
Out[28]: 0     2023
         1     2023
         2     2023
         3     2023
         4     2023
               ...
         70    2019
         71    2019
         72    2019
         73    2018
         74    2018
         Name: Year, Length: 75, dtype: int64
```

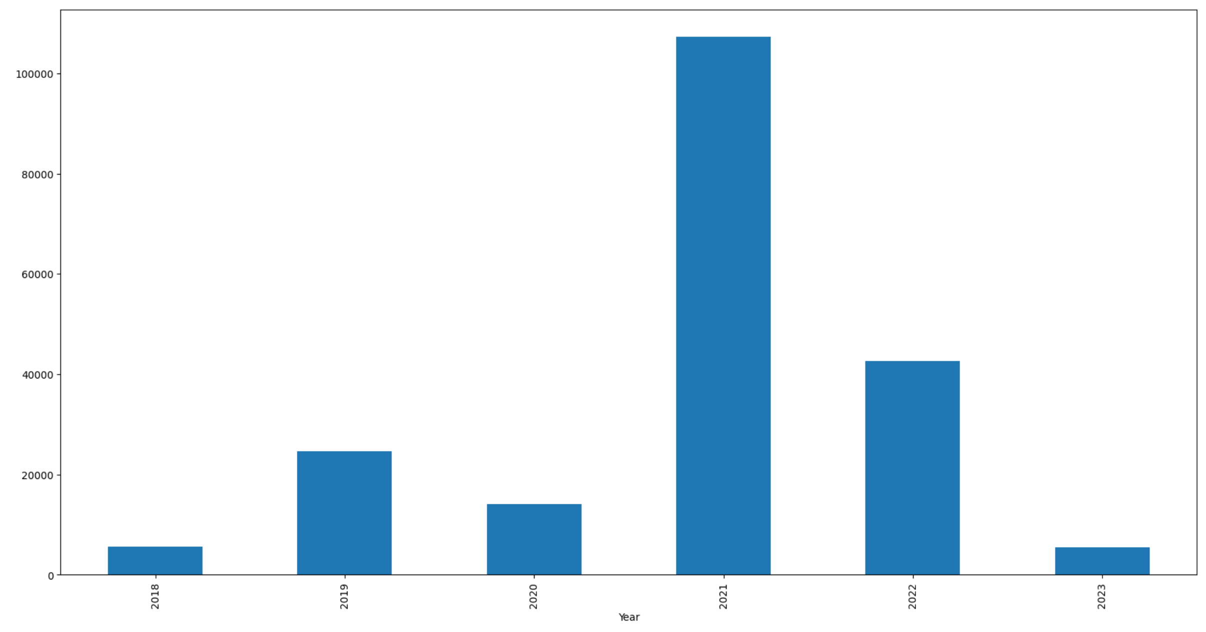
```
In [29]: yearly_spent = df.groupby('Year').sum()['Total Owed']
```

```
C:\Users\kruna\AppData\Local\Temp\ipykernel_1624\3626654573.py:1: FutureWar
ning: The default value of numeric_only in DataFrameGroupBy.sum is deprecat
ed. In a future version, numeric_only will default to False. Either specify
numeric_only or select only columns which should be valid for the function.
  yearly_spent = df.groupby('Year').sum()['Total Owed']
```

```
In [30]: yearly_spent.plot.bar(figsize=(20,10))
```

```
Out[30]: <Axes: xlabel='Year'>
```

Loading [MathJax]/extensions/Safe.js

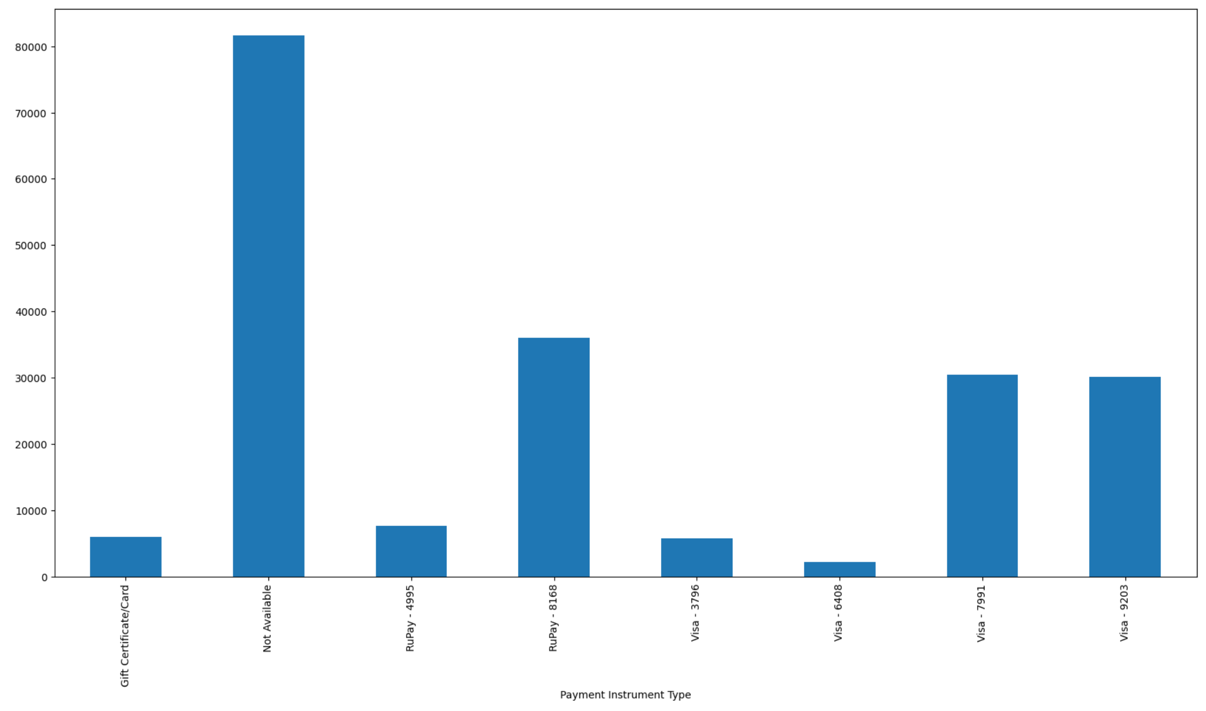


```
In [31]: payment_mode = df.groupby('Payment Instrument Type').sum()['Total Owed']
```

```
C:\Users\kruna\AppData\Local\Temp\ipykernel_1624\2568554988.py:1: FutureWar
ning: The default value of numeric_only in DataFrameGroupBy.sum is deprecat
ed. In a future version, numeric_only will default to False. Either specify
numeric_only or select only columns which should be valid for the function.
  payment_mode = df.groupby('Payment Instrument Type').sum()['Total Owed']
```

```
In [32]: payment_mode.plot.bar(figsize=(20,10))
```

```
Out[32]: <Axes: xlabel='Payment Instrument Type'>
```



```
In [ ]:
```

Loading [MathJax]/extensions/Safe.js