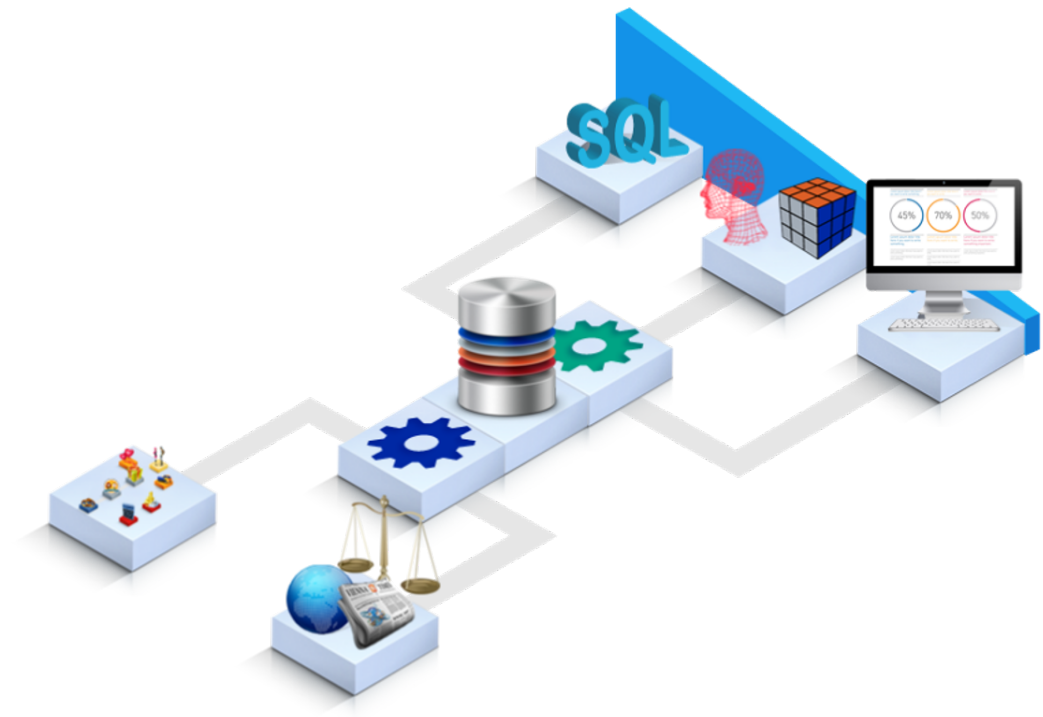
Vorlesung Informatik und Wirtschaft

Woche 4: Arten von Informationssystemen

Heutige Agenda

- Fragen / Hinweise
- Arten von Informationssystemen
  - Weshalb Informationssysteme?
  - Einsatzgebiet
  - **Wertschöpfungskette**
  - Morphologischer Kasten
- Neues aus der Forschung: Blockchains

Wertschöpfungskette: Theorie



- vorderer Bereich: ***erfassen; Dateneingang***
  - vorne links: interne Quellen (z.B. POS)
  - vorne rechts: externe Quellen
- mittlerer Bereich: ***verarbeiten und speichern***
  - "reinigen"
  - speichern

    - verwalten
- hinterer Bereich: ***analysieren und bereitstellen (mit abnehmender Flexibilität)***
    - hinten links: SQL
    - hinten mitte: CUBE, Abfragemaske (parametrisierbare Abfrage), Modelle
    - hinten rechts: Dashboard

Wertschöpfungskette: Praktisch in Python

1. Wir laden zwei Datensätze:
    - einen intern
    - einen extern
2. Wir reinigen und speichern diesen in einer Datenbank
3. Wir stellen ihn in drei Arten zur Verfügung:
    - Abfrage (SQL)
    - Explorativ / parametrisierte Suche
    - Dashboard

Das Ganze basiert auf: (https://github.com/PetraLee2019/Python-Sales-Data-Analysis/tree/master)

Daten Laden

In [5]:
```
import pandas as pd

all_data = pd.read_csv("Woche4_Daten/all_data.csv")

all_data.info()

all_data.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 186495 entries, 0 to 186494
Data columns (total 6 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   Order ID          185950 non-null  float64
 1   Product           185950 non-null  object 
 2   Quantity Ordered  185950 non-null  float64
 3   Price Each        185950 non-null  float64
 4   Order Date        185950 non-null  object 
 5   Purchase Address  185950 non-null  object 
dtypes: float64(3), object(3)
memory usage: 8.5+ MB
```

Out[5]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	176558.0	USB-C Charging Cable	2.0	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001
1	NaN	NaN	NaN	NaN	NaN	NaN
2	176559.0	Bose SoundSport Headphones	1.0	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215
3	176560.0	Google Phone	1.0	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001
4	176560.0	Wired Headphones	1.0	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001

Reinigen

Wir sehen, dass diese Daten fehler drin haben; es fehlen zum Beispiel Angaben ("NaN") und wir müssen sicherstellen, dass Zahlen wirklich als Zahlen verstanden werden.

In [7]:
```
# Drop rows of NAN
all_data = all_data.dropna(how="all")

#Convert columns to the correct type

#to int
all_data["Quantity Ordered"] = pd.to_numeric(all_data["Quantity Ord
#to float
all_data["Price Each"] = pd.to_numeric(all_data["Price Each"])

all_data.info()

all_data.head()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 185950 entries, 0 to 186494
Data columns (total 6 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   Order ID          185950 non-null  float64
 1   Product           185950 non-null  object 
 2   Quantity Ordered  185950 non-null  float64
 3   Price Each        185950 non-null  float64
 4   Order Date        185950 non-null  object 
 5   Purchase Address  185950 non-null  object 
dtypes: float64(3), object(3)
memory usage: 9.9+ MB
```

Out[7]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	176558.0	USB-C Charging Cable	2.0	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001
2	176559.0	Bose SoundSport Headphones	1.0	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215
3	176560.0	Google Phone	1.0	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001
4	176560.0	Wired Headphones	1.0	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001
5	176561.0	Wired Headphones	1.0	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001

Erweitern

Wir werden noch je eine Spalte für den Monat, die Gesamtsumme des Verkaufs, und die Stadt hinzufügen.

In [9]:
```
# Transforming "order date" column
all_data['Month'] = all_data['Order Date'].str[0:2]
all_data['Month'] = all_data['Month'].astype('int32')
all_data.head()
```

Out[9]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month
0	176558.0	USB-C Charging Cable	2.0	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001	4
2	176559.0	Bose SoundSport Headphones	1.0	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	4
3	176560.0	Google Phone	1.0	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4
4	176560.0	Wired Headphones	1.0	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4
5	176561.0	Wired Headphones	1.0	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	4

In [10]:
```
all_data['Sales'] = all_data['Quantity Ordered'] * all_data['Price
all_data.head()
```

Out[10]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sa
0	176558.0	USB-C Charging Cable	2.0	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001	4	23
2	176559.0	Bose SoundSport Headphones	1.0	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	4	99
3	176560.0	Google Phone	1.0	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	600
4	176560.0	Wired Headphones	1.0	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	11
5	176561.0	Wired Headphones	1.0	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	4	11

In [11]:
```
all_data['City'] = all_data['Purchase Address'].str.split(',').str[
all_data.head()
```

Out[11]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sa
0	176558.0	USB-C Charging Cable	2.0	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001	4	23
2	176559.0	Bose SoundSport Headphones	1.0	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	4	99
3	176560.0	Google Phone	1.0	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	600
4	176560.0	Wired Headphones	1.0	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	11
5	176561.0	Wired Headphones	1.0	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	4	11

Das Ganze in Datenbank speichern

In [13]:
```
from sqlalchemy import create_engine
engine = create_engine('sqlite://', echo=False)
```

In [14]:
```
all_data.to_sql(name='sales', con=engine)
```

Out[14]: 185950

Abfragen

- Abfrage (SQL)
- Explorativ / parametrisierte Suche
- Dashboard

SQL

In [16]:
```
from sqlalchemy import text

sql = text('SELECT * FROM sales')
```

```
with engine.connect() as connection:
    results = connection.execute(sql)

    i = 1
    for record in results:
        i = i + 1
        print("\n", record)
        if i > 10:
            break
```

```
 (0, 176558.0, 'USB-C Charging Cable', 2.0, 11.95, '04/19/19 08:46',
'917 1st St, Dallas, TX 75001', 4, 23.9, ' Dallas')

 (2, 176559.0, 'Bose SoundSport Headphones', 1.0, 99.99, '04/07/19 2
2:30', '682 Chestnut St, Boston, MA 02215', 4, 99.99, ' Boston')

 (3, 176560.0, 'Google Phone', 1.0, 600.0, '04/12/19 14:38', '669 Sp
ruce St, Los Angeles, CA 90001', 4, 600.0, ' Los Angeles')

 (4, 176560.0, 'Wired Headphones', 1.0, 11.99, '04/12/19 14:38', '66
9 Spruce St, Los Angeles, CA 90001', 4, 11.99, ' Los Angeles')

 (5, 176561.0, 'Wired Headphones', 1.0, 11.99, '04/30/19 09:27', '33
3 8th St, Los Angeles, CA 90001', 4, 11.99, ' Los Angeles')

 (6, 176562.0, 'USB-C Charging Cable', 1.0, 11.95, '04/29/19 13:03',
'381 Wilson St, San Francisco, CA 94016', 4, 11.95, ' San Francisc
o')

 (7, 176563.0, 'Bose SoundSport Headphones', 1.0, 99.99, '04/02/19 0
7:46', '668 Center St, Seattle, WA 98101', 4, 99.99, ' Seattle')

 (8, 176564.0, 'USB-C Charging Cable', 1.0, 11.95, '04/12/19 10:58',
'790 Ridge St, Atlanta, GA 30301', 4, 11.95, ' Atlanta')

 (9, 176565.0, 'Macbook Pro Laptop', 1.0, 1700.0, '04/24/19 10:38',
'915 Willow St, San Francisco, CA 94016', 4, 1700.0, ' San Francisc
o')

 (10, 176566.0, 'Wired Headphones', 1.0, 11.99, '04/08/19 14:05', '8
3 7th St, Boston, MA 02215', 4, 11.99, ' Boston')
```

Parametrisiert

In [18]:
```
sql = text("SELECT * FROM sales WHERE Product = 'Google Phone'")


with engine.connect() as connection:
    results = connection.execute(sql)

    i = 1
    for record in results:
        i = i + 1
        print("\n", record)
        if i > 10:
```

```
            break
```

```
(3, 176560.0, 'Google Phone', 1.0, 600.0, '04/12/19 14:38', '669 Sp
ruce St, Los Angeles, CA 90001', 4, 600.0, ' Los Angeles')

(11, 176567.0, 'Google Phone', 1.0, 600.0, '04/18/19 17:18', '444 7
th St, Los Angeles, CA 90001', 4, 600.0, ' Los Angeles')

(18, 176574.0, 'Google Phone', 1.0, 600.0, '04/03/19 19:42', '20 Hi
ll St, Los Angeles, CA 90001', 4, 600.0, ' Los Angeles')

(33, 176586.0, 'Google Phone', 1.0, 600.0, '04/10/19 17:00', '365 C
enter St, San Francisco, CA 94016', 4, 600.0, ' San Francisco')

(37, 176590.0, 'Google Phone', 1.0, 600.0, '04/11/19 11:46', '873 6
th St, New York City, NY 10001', 4, 600.0, ' New York City')

(77, 176630.0, 'Google Phone', 1.0, 600.0, '04/10/19 05:01', '770 M
aple St, San Francisco, CA 94016', 4, 600.0, ' San Francisco')

(94, 176647.0, 'Google Phone', 1.0, 600.0, '04/21/19 19:40', '273 S
outh St, Los Angeles, CA 90001', 4, 600.0, ' Los Angeles')

(103, 176656.0, 'Google Phone', 1.0, 600.0, '04/12/19 16:42', '12 L
akeview St, New York City, NY 10001', 4, 600.0, ' New York City')

(162, 176712.0, 'Google Phone', 1.0, 600.0, '04/20/19 05:21', '910
Washington St, San Francisco, CA 94016', 4, 600.0, ' San Francisco')

(190, 176739.0, 'Google Phone', 1.0, 600.0, '04/05/19 17:38', '730
6th St, Austin, TX 73301', 4, 600.0, ' Austin')
```

"Dashboad"

Zunächst mal einige einfache Abfragen

Was ist der Beste Monat für den Verkauf?

In [20]:
```
results = all_data.groupby('Month').sum(numeric_only = True)
results
```

Out[20]:

Month	Order ID	Quantity Ordered	Price Each	Sales
1	1.421631e+09	10903.0	1811768.38	1822256.73
2	1.871053e+09	13449.0	2188884.72	2202022.42
3	2.564811e+09	17005.0	2791207.83	2807100.38
4	3.387347e+09	20558.0	3367671.02	3390670.24
5	3.345872e+09	18667.0	3135125.13	3152606.75
6	2.932976e+09	15253.0	2562025.61	2577802.26
7	3.284140e+09	16072.0	2632539.56	2647775.76
8	2.899374e+09	13448.0	2230345.42	2244467.88
9	2.948727e+09	13109.0	2084992.09	2097560.13
10	5.457110e+09	22703.0	3715554.83	3736726.88
11	5.047203e+09	19798.0	3180600.68	3199603.20
12	7.685905e+09	28114.0	4588415.41	4613443.34

Welche Stadt hat die besten Resultate

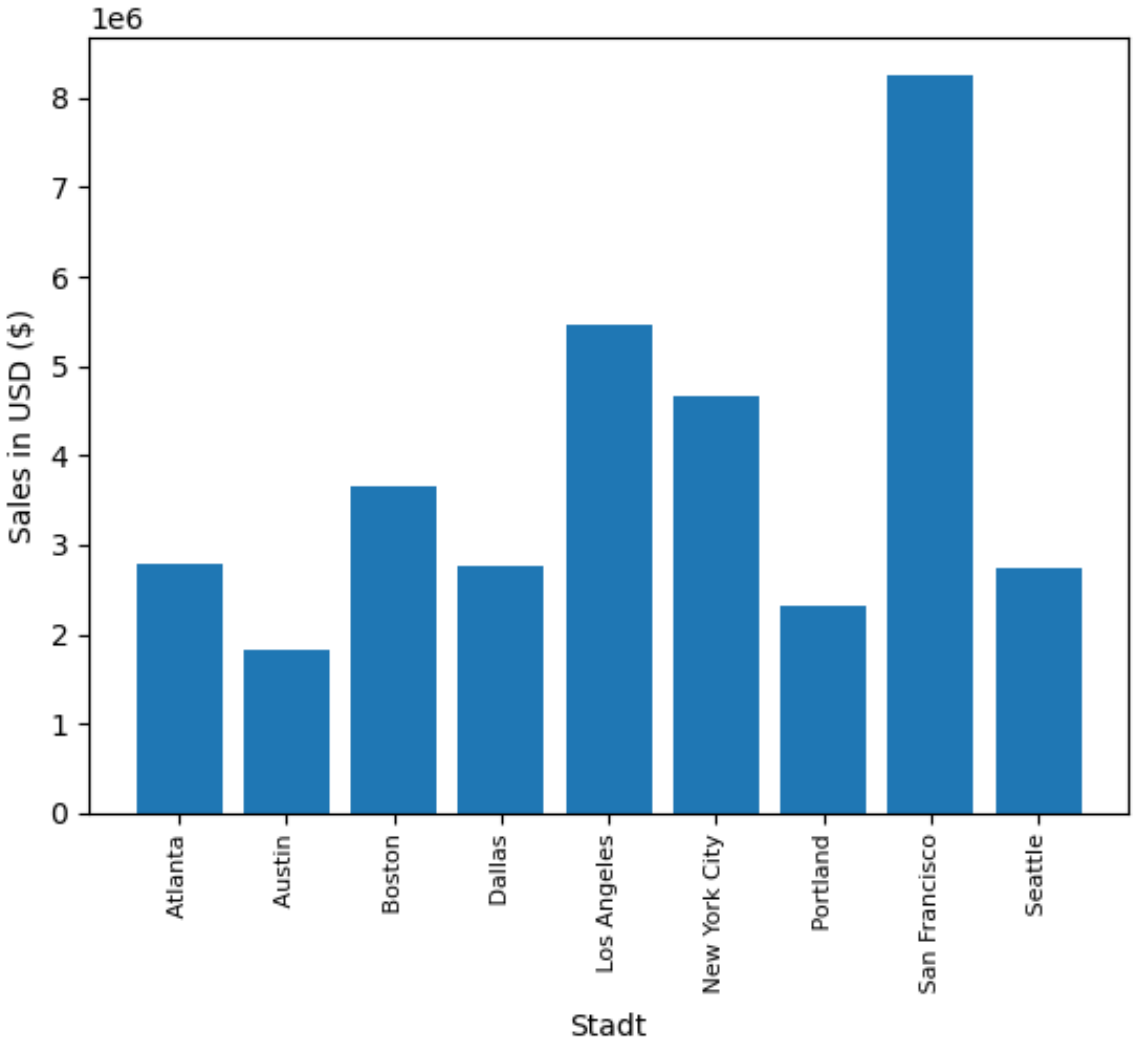
In [22]:
```
results = all_data.groupby('City').sum(numeric_only = True)
results
```

Out[22]:

City	Order ID	Quantity Ordered	Price Each	Month	Sales
Atlanta	3.423838e+09	16602.0	2779908.20	104794	2795498.58
Austin	2.280982e+09	11153.0	1809873.61	69829	1819581.75
Boston	4.598265e+09	22528.0	3637409.77	141112	3661642.01
Dallas	3.415644e+09	16730.0	2752627.82	104620	2767975.40
Los Angeles	6.811085e+09	33289.0	5421435.23	208325	5452570.80
New York City	5.736334e+09	27932.0	4635370.83	175741	4664317.43
Portland	2.868861e+09	14053.0	2307747.47	87765	2320490.61
San Francisco	1.030444e+10	50239.0	8211461.74	315520	8262203.91
Seattle	3.406694e+09	16553.0	2733296.01	104941	2747755.48

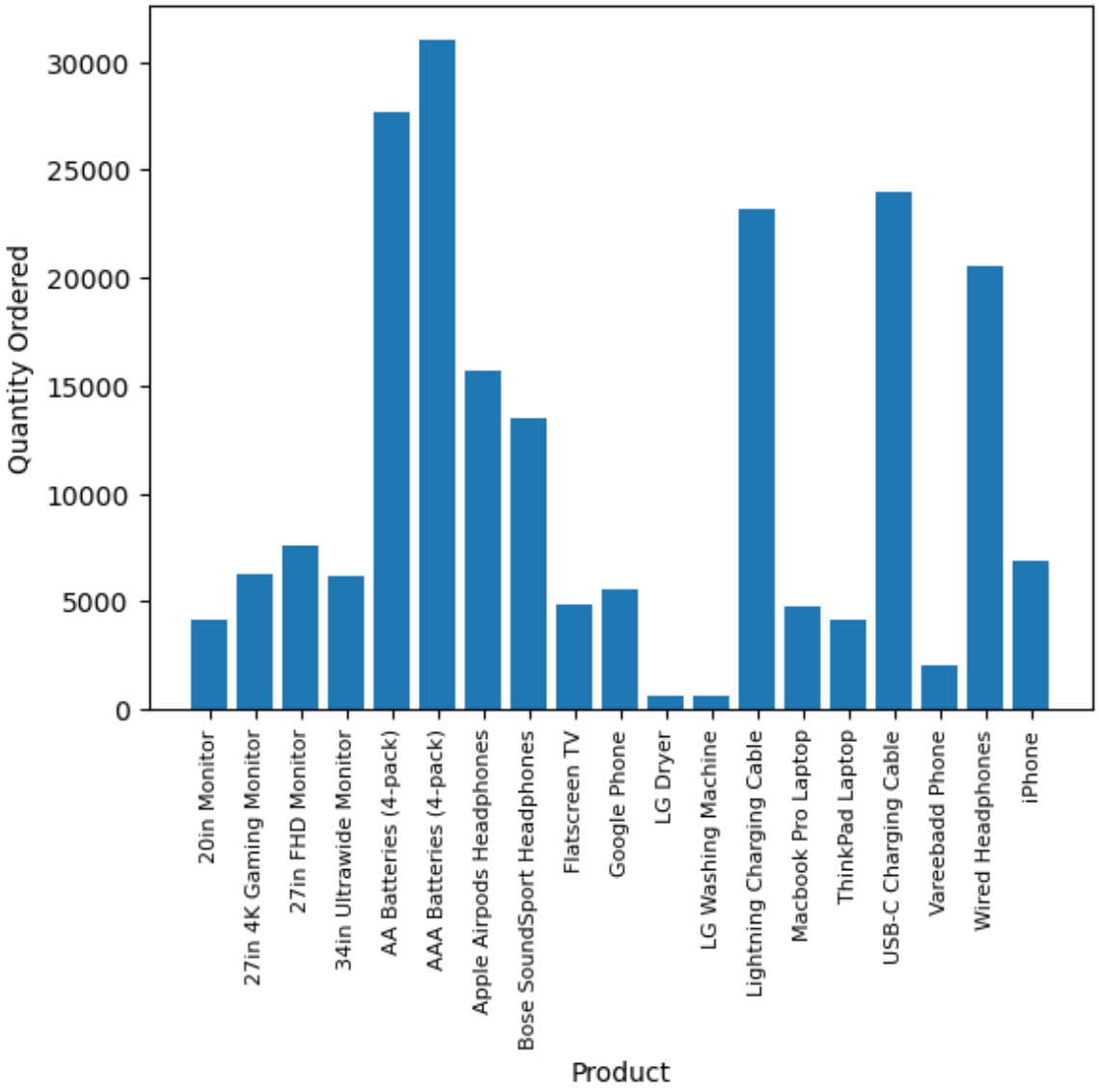
und jetzt noch als Grafik:

In [24]:
```
import matplotlib.pyplot as plt

cities = [city for city, df in all_data.groupby('City')]

plt.bar(cities, results['Sales'])
plt.xticks(cities, rotation ='vertical', size=8)
plt.xlabel('Stadt')
plt.ylabel('Sales in USD ($)')
plt.show()
```



Und jetzt das noch nach Produkt:

In [26]:
```
product_group = all_data.groupby('Product')
quantity_ordered = product_group.sum(numeric_only = True)['Quantity
products = [product for product, df in product_group]

plt.bar(products, quantity_ordered)
plt.xticks(products, rotation ='vertical', size=8)
plt.xlabel('Product')
plt.ylabel('Quantity Ordered')
plt.show()
```


Quantity Ordered
30000
25000
20000
15000
10000
5000
0
20in Monitor
27in 4K Gaming Monitor
27in FHD Monitor
34in Ultrawide Monitor
AA Batteries (4-pack)
AAA Batteries (4-pack)
Apple Airpods Headphones
Bose SoundSport Headphones
Flatscreen TV
Google Phone
LG Dryer
LG Washing Machine
Lightning Charging Cable
Macbook Pro Laptop
ThinkPad Laptop
USB-C Charging Cable
Vareebadd Phone
Wired Headphones
iPhone
Product