In [417]:
```
#Importing packages
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [418]:
```
#Reading and Studying the Datasets
sales=pd.read_csv(r"C:\Users\HP\OneDrive\Desktop\RISE internship\DataScience and Anal
sales
```

Out[418]:

	Transaction ID	Date	Product Category	Product Name	Units Sold(Quantity)	Unit Price	Total Revenue	Reg
0	10001	1/1/2024	Electronics	iPhone 14 Pro	2	999.99	1999.98	N Ame
1	10002	1/2/2024	Home Appliances	Dyson V11 Vacuum	1	499.99	499.99	Eur
2	10003	1/3/2024	Clothing	Levi's 501 Jeans	3	69.99	209.97	
3	10004	1/4/2024	Books	The Da Vinci Code	4	15.99	63.96	N Ame
4	10005	1/5/2024	Beauty Products	Neutrogena Skincare Set	1	89.99	89.99	Eur
...	...	...	...	...	...	...	...	
235	10236	8/23/2024	Home Appliances	Nespresso Vertuo Next Coffee and Espresso Maker	1	159.99	159.99	Eur
236	10237	8/24/2024	Clothing	Nike Air Force 1 Sneakers	3	90.00	270.00	
237	10238	8/25/2024	Books	The Handmaid's Tale by Margaret Atwood	3	10.99	32.97	N Ame
238	10239	8/26/2024	Beauty Products	Sunday Riley Luna Sleeping Night Oil	1	55.00	55.00	Eur
239	10240	8/27/2024	Sports	Yeti Rambler 20 oz Tumbler	2	29.99	59.98	

240 rows × 9 columns

In [419]:
```
#Data Cleaning and Analysis
sales.isnull().sum()
```

Out[419]:
```
Transaction ID          0
Date                    0
Product Category        0
Product Name            0
Units Sold(Quantity)    0
Unit Price              0
Total Revenue           0
Region                  0
Payment Method          0
dtype: int64
```

In [420]:
```
sales.dtypes
```

Out[420]:
```
Transaction ID          int64
Date                   object
Product Category       object
Product Name           object
Units Sold(Quantity)    int64
Unit Price            float64
Total Revenue         float64
Region                 object
Payment Method         object
dtype: object
```

In [421]:
```
for i in sales.columns:
    print(i,":","\n",sales[i].unique(),"\n")
```

```
Transaction ID :
 [10001 10002 10003 10004 10005 10006 10007 10008 10009 10010 10011 10012
 10013 10014 10015 10016 10017 10018 10019 10020 10021 10022 10023 10024
 10025 10026 10027 10028 10029 10030 10031 10032 10033 10034 10035 10036
 10037 10038 10039 10040 10041 10042 10043 10044 10045 10046 10047 10048
 10049 10050 10051 10052 10053 10054 10055 10056 10057 10058 10059 10060
 10061 10062 10063 10064 10065 10066 10067 10068 10069 10070 10071 10072
 10073 10074 10075 10076 10077 10078 10079 10080 10081 10082 10083 10084
 10085 10086 10087 10088 10089 10090 10091 10092 10093 10094 10095 10096
 10097 10098 10099 10100 10101 10102 10103 10104 10105 10106 10107 10108
 10109 10110 10111 10112 10113 10114 10115 10116 10117 10118 10119 10120
 10121 10122 10123 10124 10125 10126 10127 10128 10129 10130 10131 10132
 10133 10134 10135 10136 10137 10138 10139 10140 10141 10142 10143 10144
 10145 10146 10147 10148 10149 10150 10151 10152 10153 10154 10155 10156
 10157 10158 10159 10160 10161 10162 10163 10164 10165 10166 10167 10168
 10169 10170 10171 10172 10173 10174 10175 10176 10177 10178 10179 10180
 10181 10182 10183 10184 10185 10186 10187 10188 10189 10190 10191 10192
 10193 10194 10195 10196 10197 10198 10199 10200 10201 10202 10203 10204
 10205 10206 10207 10208 10209 10210 10211 10212 10213 10214 10215 10216
 10217 10218 10219 10220 10221 10222 10223 10224 10225 10226 10227 10228
 10229 10230 10231 10232 10233 10234 10235 10236 10237 10238 10239 10240]

Date :
 ['1/1/2024' '1/2/2024' '1/3/2024' '1/4/2024' '1/5/2024' '1/6/2024'
 '1/7/2024' '1/8/2024' '1/9/2024' '1/10/2024' '1/11/2024' '1/12/2024'
 '1/13/2024' '1/14/2024' '1/15/2024' '1/16/2024' '1/17/2024' '1/18/2024'
 '1/19/2024' '1/20/2024' '1/21/2024' '1/22/2024' '1/23/2024' '1/24/2024'
 '1/25/2024' '1/26/2024' '1/27/2024' '1/28/2024' '1/29/2024' '1/30/2024'
 '1/31/2024' '2/1/2024' '2/2/2024' '2/3/2024' '2/4/2024' '2/5/2024'
 '2/6/2024' '2/7/2024' '2/8/2024' '2/9/2024' '2/10/2024' '2/11/2024'
 '2/12/2024' '2/13/2024' '2/14/2024' '2/15/2024' '2/16/2024' '2/17/2024'
 '2/18/2024' '2/19/2024' '2/20/2024' '2/21/2024' '2/22/2024' '2/23/2024'
 '2/24/2024' '2/25/2024' '2/26/2024' '2/27/2024' '2/28/2024' '2/29/2024'
 '3/1/2024' '3/2/2024' '3/3/2024' '3/4/2024' '3/5/2024' '3/6/2024'
 '3/7/2024' '3/8/2024' '3/9/2024' '3/10/2024' '3/11/2024' '3/12/2024'
 '3/13/2024' '3/14/2024' '3/15/2024' '3/16/2024' '3/17/2024' '3/18/2024'
 '3/19/2024' '3/20/2024' '3/21/2024' '3/22/2024' '3/23/2024' '3/24/2024'
 '3/25/2024' '3/26/2024' '3/27/2024' '3/28/2024' '3/29/2024' '3/30/2024'
 '3/31/2024' '4/1/2024' '4/2/2024' '4/3/2024' '4/4/2024' '4/5/2024'
 '4/6/2024' '4/7/2024' '4/8/2024' '4/9/2024' '4/10/2024' '4/11/2024'
 '4/12/2024' '4/13/2024' '4/14/2024' '4/15/2024' '4/16/2024' '4/17/2024'
 '4/18/2024' '4/19/2024' '4/20/2024' '4/21/2024' '4/22/2024' '4/23/2024'
 '4/24/2024' '4/25/2024' '4/26/2024' '4/27/2024' '4/28/2024' '4/29/2024'
 '4/30/2024' '5/1/2024' '5/2/2024' '5/3/2024' '5/4/2024' '5/5/2024'
 '5/6/2024' '5/7/2024' '5/8/2024' '5/9/2024' '5/10/2024' '5/11/2024'
 '5/12/2024' '5/13/2024' '5/14/2024' '5/15/2024' '5/16/2024' '5/17/2024'
 '5/18/2024' '5/19/2024' '5/20/2024' '5/21/2024' '5/22/2024' '5/23/2024'
 '5/24/2024' '5/25/2024' '5/26/2024' '5/27/2024' '5/28/2024' '5/29/2024'
 '5/30/2024' '5/31/2024' '6/1/2024' '6/2/2024' '6/3/2024' '6/4/2024'
 '6/5/2024' '6/6/2024' '6/7/2024' '6/8/2024' '6/9/2024' '6/10/2024'
 '6/11/2024' '6/12/2024' '6/13/2024' '6/14/2024' '6/15/2024' '6/16/2024'
 '6/17/2024' '6/18/2024' '6/19/2024' '6/20/2024' '6/21/2024' '6/22/2024'
 '6/23/2024' '6/24/2024' '6/25/2024' '6/26/2024' '6/27/2024' '6/28/2024'
 '6/29/2024' '6/30/2024' '7/1/2024' '7/2/2024' '7/3/2024' '7/4/2024'
 '7/5/2024' '7/6/2024' '7/7/2024' '7/8/2024' '7/9/2024' '7/10/2024'
 '7/11/2024' '7/12/2024' '7/13/2024' '7/14/2024' '7/15/2024' '7/16/2024'
```

```
 '7/17/2024' '7/18/2024' '7/19/2024' '7/20/2024' '7/21/2024' '7/22/2024'
 '7/23/2024' '7/24/2024' '7/25/2024' '7/26/2024' '7/27/2024' '7/28/2024'
 '7/29/2024' '7/30/2024' '7/31/2024' '8/1/2024' '8/2/2024' '8/3/2024'
 '8/4/2024' '8/5/2024' '8/6/2024' '8/7/2024' '8/8/2024' '8/9/2024'
 '8/10/2024' '8/11/2024' '8/12/2024' '8/13/2024' '8/14/2024' '8/15/2024'
 '8/16/2024' '8/17/2024' '8/18/2024' '8/19/2024' '8/20/2024' '8/21/2024'
 '8/22/2024' '8/23/2024' '8/24/2024' '8/25/2024' '8/26/2024' '8/27/2024']

Product Category :
 ['Electronics' 'Home Appliances' 'Clothing' 'Books' 'Beauty Products'
 'Sports']

Product Name :
 ['iPhone 14 Pro' 'Dyson V11 Vacuum' "Levi's 501 Jeans" 'The Da Vinci Code'
 'Neutrogena Skincare Set' 'Wilson Evolution Basketball'
 'MacBook Pro 16-inch' 'Blueair Classic 480i' 'Nike Air Force 1'
 'Dune by Frank Herbert' 'Chanel No. 5 Perfume'
 'Babolat Pure Drive Tennis Racket' 'Samsung Galaxy Tab S8'
 'Keurig K-Elite Coffee Maker' 'North Face Down Jacket'
 'Salt, Fat, Acid, Heat by Samin Nosrat' 'Dyson Supersonic Hair Dryer'
 'Manduka PRO Yoga Mat' 'Garmin Forerunner 945'
 'Ninja Professional Blender' 'Zara Summer Dress'
 'Gone Girl by Gillian Flynn' 'Olay Regenerist Face Cream'
 'Adidas FIFA World Cup Football' 'Bose QuietComfort 35 Headphones'
 'Panasonic NN-SN966S Microwave' 'Adidas Ultraboost Shoes'
 'Pride and Prejudice by Jane Austen' 'MAC Ruby Woo Lipstick'
 'Nike Air Zoom Pegasus 37' 'Sony WH-1000XM4 Headphones' 'Instant Pot Duo'
 'Under Armour HeatGear T-Shirt' '1984 by George Orwell'
 "L'Oreal Revitalift Serum" 'Peloton Bike' 'Apple Watch Series 8'
 'Roomba i7+' 'Columbia Fleece Jacket'
 "Harry Potter and the Sorcerer's Stone"
 'Estee Lauder Advanced Night Repair' 'Fitbit Charge 5'
 'GoPro HERO10 Black' 'Nespresso VertuoPlus' 'Patagonia Better Sweater'
 'Becoming by Michelle Obama' 'Clinique Moisture Surge'
 'Yeti Rambler Tumbler' 'Kindle Paperwhite' 'Breville Smart Oven'
 'Ray-Ban Aviator Sunglasses' 'The Silent Patient by Alex Michaelides'
 'Shiseido Ultimate Sun Protector' 'Titleist Pro V1 Golf Balls'
 'Anker PowerCore Portable Charger' 'KitchenAid Artisan Stand Mixer'
 'Calvin Klein Boxer Briefs' 'Educated by Tara Westover'
 'Anastasia Beverly Hills Brow Wiz' 'Hyperice Hypervolt Massager'
 'Nintendo Switch' 'Philips Airfryer XXL' 'Hanes ComfortSoft T-Shirt'
 'Where the Crawdads Sing by Delia Owens' 'Lancome La Vie Est Belle'
 'Garmin Edge 530' 'Samsung QLED 4K TV' 'Eufy RoboVac 11S'
 'Puma Suede Classic Sneakers' 'The Great Gatsby by F. Scott Fitzgerald'
 'Drunk Elephant C-Firma Day Serum' 'Nike Metcon 6'
 'HP Spectre x360 Laptop' "De'Longhi Magnifica Espresso Machine"
 'Tommy Hilfiger Polo Shirt' 'To Kill a Mockingbird by Harper Lee'
 'Glossier Boy Brow' 'Rogue Fitness Kettlebell' 'Apple AirPods Pro'
 'Dyson Pure Cool Link' "Levi's Trucker Jacket"
 'The Hobbit by J.R.R. Tolkien' 'Charlotte Tilbury Magic Cream'
 'Spalding NBA Street Basketball' 'Ring Video Doorbell' 'LG OLED TV'
 'Uniqlo Ultra Light Down Jacket'
 'The Catcher in the Rye by J.D. Salinger' 'Sunday Riley Good Genes'
 'On Running Cloud Shoes' 'Logitech MX Master 3 Mouse'
 'Instant Pot Duo Crisp' 'Adidas Originals Superstar Sneakers'
 'The Alchemist by Paulo Coelho' 'Tatcha The Water Cream'
```

'Garmin Fenix 6X Pro' 'Bose SoundLink Revolve+ Speaker'
'Vitamix Explorian Blender' 'Gap Essential Crewneck T-Shirt'
'The Power of Now by Eckhart Tolle'
"Kiehl's Midnight Recovery Concentrate" 'Under Armour HOVR Sonic 4 Shoes'
'Canon EOS R5 Camera' 'Shark IQ Robot Vacuum' 'H&M Slim Fit Jeans'
'The Girl on the Train by Paula Hawkins' 'The Ordinary Niacinamide Serum'
'Bowflex SelectTech 552 Dumbbells' 'Google Nest Hub Max'
'Cuisinart Griddler Deluxe' 'Old Navy Relaxed-Fit T-Shirt'
'Sapiens: A Brief History of Humankind by Yuval Noah Harari'
'Biore UV Aqua Rich Watery Essence Sunscreen' 'Fitbit Versa 3'
'Amazon Echo Show 10' 'Breville Smart Grill' 'Gap High Rise Skinny Jeans'
'Atomic Habits by James Clear' 'CeraVe Hydrating Facial Cleanser'
'YETI Hopper Flip Portable Cooler' 'Apple iPad Air'
'Hamilton Beach FlexBrew Coffee Maker' 'Forever 21 Graphic Tee'
'The Subtle Art of Not Giving a F*ck by Mark Manson'
'NARS Radiant Creamy Concealer' 'Yeti Roadie 24 Cooler'
'Sony PlayStation 5' 'Lululemon Align Leggings'
'The Four Agreements by Don Miguel Ruiz'
'Fenty Beauty Killawatt Highlighter'
'Hydro Flask Wide Mouth Water Bottle' 'Microsoft Surface Laptop 4'
'Keurig K-Mini Coffee Maker' 'Gap Crewneck Sweatshirt'
'Think and Grow Rich by Napoleon Hill'
'The Ordinary Hyaluronic Acid Serum' 'Fitbit Inspire 2'
'Samsung Odyssey G9 Gaming Monitor' 'Instant Pot Ultra'
'Adidas Essential Track Pants' 'The Power of Habit by Charles Duhigg'
'Clinique Dramatically Different Moisturizing Lotion'
'YETI Tundra 45 Cooler' 'Apple AirPods Max' 'Cuisinart Coffee Center'
"Levi's Sherpa Trucker Jacket" 'The Outsiders by S.E. Hinton'
'Laneige Water Sleeping Mask' 'Bose SoundSport Wireless Earbuds'
'Ninja Foodi Pressure Cooker' 'Nike Sportswear Club Fleece Hoodie'
'The Night Circus by Erin Morgenstern'
'GlamGlow Supermud Clearing Treatment' 'Garmin Forerunner 245'
'Google Pixel 6 Pro' 'Breville Nespresso Creatista Plus'
'Under Armour Tech 2.0 T-Shirt' 'The Art of War by Sun Tzu'
'Youth to the People Superfood Antioxidant Cleanser'
'TriggerPoint GRID Foam Roller' 'Apple MacBook Air'
'Cuisinart Custom 14-Cup Food Processor' 'Adidas 3-Stripes Shorts'
'The Hunger Games by Suzanne Collins' 'Neutrogena Hydro Boost Water Gel'
'Yeti Rambler Bottle' 'Samsung Odyssey G7 Gaming Monitor'
'Instant Pot Duo Evo Plus' 'Nike Tempo Running Shorts'
'The Girl with the Dragon Tattoo by Stieg Larsson'
"Paula's Choice Skin Perfecting 2% BHA Liquid Exfoliant"
'Bowflex SelectTech 1090 Adjustable Dumbbells' 'Amazon Fire TV Stick 4K'
'Crock-Pot 6-Quart Slow Cooker' 'Uniqlo Airism Mesh Boxer Briefs'
'The Sun Also Rises by Ernest Hemingway'
'First Aid Beauty Ultra Repair Cream' 'Oakley Holbrook Sunglasses'
'Google Pixelbook Go' 'Dyson V8 Absolute' "Levi's 511 Slim Fit Jeans"
'The Martian by Andy Weir' 'La Mer Crème de la Mer Moisturizer'
'Polar Vantage V2' 'Sonos Beam Soundbar' 'Anova Precision Cooker'
'Nike Dri-FIT Training Shorts' 'Glossier Cloud Paint'
'TRX All-in-One Suspension Training System'
'Logitech G Pro X Wireless Gaming Headset'
'Breville Smart Coffee Grinder Pro' 'Adidas Ultraboost Running Shoes'
'The Road by Cormac McCarthy' 'Tom Ford Black Orchid Perfume'
'GoPro HERO9 Black' 'Apple TV 4K' 'Instant Pot Duo Nova'
'Gap 1969 Original Fit Jeans' 'The Goldfinch by Donna Tartt'

```
 'Dr. Jart+ Cicapair Tiger Grass Color Correcting Treatment'
 'Yeti Tundra Haul Portable Wheeled Cooler' 'Samsung Galaxy Watch 4'
 'KitchenAid Stand Mixer' 'Lululemon Wunder Under High-Rise Leggings'
 'The Great Alone by Kristin Hannah' 'Caudalie Vinoperfect Radiance Serum'
 'Bose SoundLink Color Bluetooth Speaker II'
 'Canon EOS Rebel T7i DSLR Camera' 'Uniqlo Airism Seamless Boxer Briefs'
 "L'Occitane Shea Butter Hand Cream" 'YETI Tundra 65 Cooler'
 'Apple MacBook Pro 16-inch' 'iRobot Braava Jet M6'
 'Champion Reverse Weave Hoodie' 'The Nightingale by Kristin Hannah'
 'Tarte Shape Tape Concealer' 'Amazon Echo Dot (4th Gen)'
 'Philips Sonicare DiamondClean Toothbrush'
 'Old Navy Mid-Rise Rockstar Super Skinny Jeans'
 'The Ordinary Caffeine Solution 5% + EGCG' 'Fitbit Luxe'
 'Google Nest Wifi Router' 'Anova Precision Oven'
 'Adidas Originals Trefoil Hoodie' 'Fresh Sugar Lip Treatment'
 'Hydro Flask Standard Mouth Water Bottle'
 'Bose QuietComfort 35 II Wireless Headphones'
 'Nespresso Vertuo Next Coffee and Espresso Maker'
 'Nike Air Force 1 Sneakers' "The Handmaid's Tale by Margaret Atwood"
 'Sunday Riley Luna Sleeping Night Oil' 'Yeti Rambler 20 oz Tumbler']

Units Sold(Quantity) :
 [ 2  1  3  4  5  6 10]

Unit Price :
 [ 999.99  499.99   69.99   15.99   89.99   29.99 2499.99  599.99   25.99
  129.99  199.99  749.99  189.99  249.99   35.99  399.99  119.99   99.99
   59.99   22.99   49.99  299.99  179.99   12.99  349.99   19.99   39.99
 1895.    799.99   24.99  105.    139.99   32.5    52.    154.99   26.99
   49.     28.     23.    349.      9.99   18.99  102.   1199.99  219.99
   10.99   78.   1599.99  899.99   14.99   16.    100.   1299.99   79.99
   13.99   68.     82.    109.99 3899.99    6.5   229.99  159.99   15.
  229.95  299.95   16.99   30.     98.      8.99   36.     39.95   34.99
    6.8    99.95 1499.99   44.99   11.99   29.5   549.    199.95   25.
  149.99   54.99   59.    499.95    7.99  699.99   14.9    34.    146.
  649.99  190.    399.    199.     18.    169.95  125.    449.99  179.
  379.99   79.    129.    169.99    9.9    29.   2399.     27.      6.7
  149.95  169.    599.     64.99   24.     32.95  299.     90.     55.  ]

Total Revenue :
 [1999.98  499.99  209.97   63.96   89.99  149.95 2499.99 1199.98  539.94
   51.98  129.99  599.97 1499.98  189.99  499.98  107.97  399.99  479.96
  999.98   99.99  179.97   45.98   49.99   89.97  299.99  179.99  359.98
   38.97   29.99  259.98  699.98  269.97   79.96   79.98 1895.   1199.97
 1599.98  239.96   74.97  105.    199.99  279.98  130.     52.    239.94
  464.97   53.98   49.    249.95   84.     46.    349.    899.97  399.98
   99.9    75.96  102.    599.98 1199.99  659.97   21.98   78.    389.97
 1599.99  899.99   59.96   32.    179.98  100.    149.94 2599.98  239.97
   55.96  199.98  319.96   44.97   68.    999.99  349.99  119.94   25.98
   82.    219.98 3899.99  119.97   43.96    6.5   459.98  159.99   37.98
   15.    689.85  249.99  599.9   149.97   67.96   29.98   64.95   30.
  799.98  294.     17.98   36.    159.8  1299.99  159.98  139.96   29.97
    6.8   199.9  1499.99  139.99  134.97   23.98   29.5   549.    399.9
  196.     32.97   25.    299.98  164.97   33.98   59.    499.95   99.96
   23.97   69.98   35.96   16.99  699.99  104.97   19.98   99.98   59.6
   35.97  146.    649.99  190.    399.    398.     18.    169.95  199.95
```

```
 125.    449.99  358.    99.95  379.99   50.97   79.    129.    749.99
 339.98   39.6    58.   2399.    27.    599.99  199.96  89.98   80.97
   6.7   299.9   169.    599.   259.96   24.     98.85  299.    270.
  55.    59.98]

Region :
 ['North America' 'Europe' 'Asia']

Payment Method :
 ['Credit Card' 'PayPal' 'Debit Card']
```

In [422]:
```
sales.describe(include="all")
```

Out[422]:

	Transaction ID	Date	Product Category	Product Name	Units Sold(Quantity)	Unit Price	Rev
count	240.00000	240	240	240	240.000000	240.000000	240.00
unique	NaN	240	6	232	NaN	NaN	
top	NaN	1/1/2024	Electronics	Dyson Supersonic Hair Dryer	NaN	NaN	
freq	NaN	1	40	2	NaN	NaN	
mean	10120.50000	NaN	NaN	NaN	2.158333	236.395583	335.69
std	69.42622	NaN	NaN	NaN	1.322454	429.446695	485.80
min	10001.00000	NaN	NaN	NaN	1.000000	6.500000	6.50
25%	10060.75000	NaN	NaN	NaN	1.000000	29.500000	62.96
50%	10120.50000	NaN	NaN	NaN	2.000000	89.990000	179.97
75%	10180.25000	NaN	NaN	NaN	3.000000	249.990000	399.22
max	10240.00000	NaN	NaN	NaN	10.000000	3899.990000	3899.99

In [423]:
```
#Data Visualization or EDA (Exploratory Data Analysis)
sales=sales.drop(["Transaction ID","Product Name","Region","Unit Price","Payment Meth
```

In [424]:
```
sales
```

Out[424]:

	Date	Product Category	Units Sold(Quantity)	Total Revenue
0	1/1/2024	Electronics	2	1999.98
1	1/2/2024	Home Appliances	1	499.99
2	1/3/2024	Clothing	3	209.97
3	1/4/2024	Books	4	63.96
4	1/5/2024	Beauty Products	1	89.99
...	...	...	...	...
235	8/23/2024	Home Appliances	1	159.99
236	8/24/2024	Clothing	3	270.00
237	8/25/2024	Books	3	32.97
238	8/26/2024	Beauty Products	1	55.00
239	8/27/2024	Sports	2	59.98

240 rows × 4 columns

In [425]:
```
import pandas as pd
import numpy as np

# Basic conversion - ensure the Date column is in datetime format
sales['Date'] = pd.to_datetime(sales['Date'])

# Linear time feature - days since first date in dataset
sales['days_since_start'] = (sales['Date'] - sales['Date'].min()).dt.days

# Cyclical features - extract date components
sales['month'] = sales['Date'].dt.month
sales['day_of_week'] = sales['Date'].dt.dayofweek  # Monday=0, Sunday=6
sales['day_of_month'] = sales['Date'].dt.day

# Verify the transformations
print(sales[['Date', 'days_since_start', 'month', 'day_of_week']].head())
```

```
        Date  days_since_start  month  day_of_week
0 2024-01-01                 0      1            0
1 2024-01-02                 1      1            1
2 2024-01-03                 2      1            2
3 2024-01-04                 3      1            3
4 2024-01-05                 4      1            4
```

In [426]:
```
#Encoding
from sklearn.preprocessing import LabelEncoder

# Create encoder
le = LabelEncoder()

# Fit and transform
```

```
sales["Product Category"] = le.fit_transform(sales["Product Category"])

sales["Product Category"].unique()
```

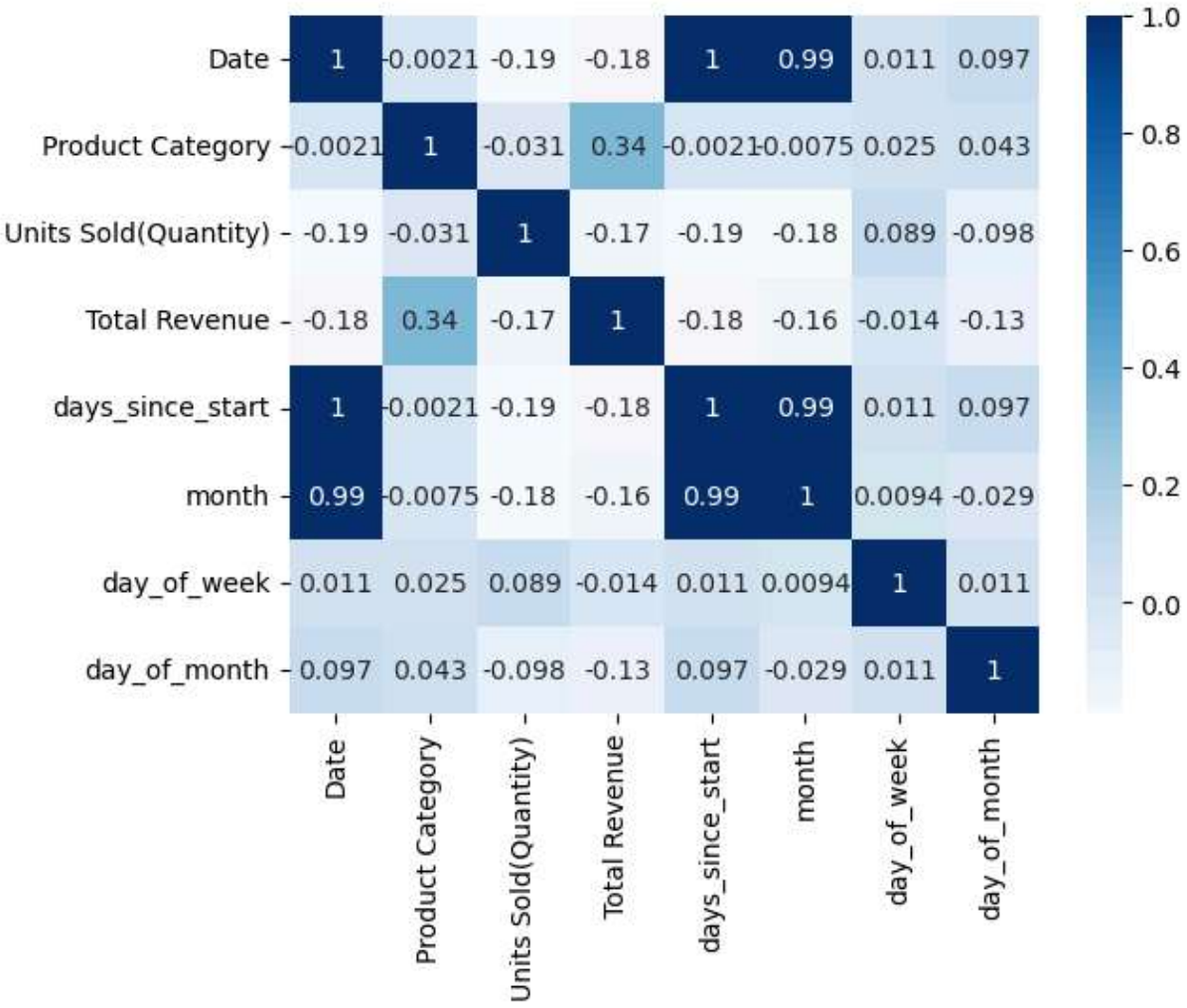
Out[426]:
```
array([3, 4, 2, 1, 0, 5])
```

In [427]:
```
c=sales.corr()
c
```

Out[427]:

	Date	Product Category	Units Sold(Quantity)	Total Revenue	days_since_start	month
Date	1.000000	-0.002113	-0.189944	-0.175622	1.000000	0.992106
Product Category	-0.002113	1.000000	-0.031428	0.341534	-0.002113	-0.007521
Units Sold(Quantity)	-0.189944	-0.031428	1.000000	-0.171151	-0.189944	-0.178548
Total Revenue	-0.175622	0.341534	-0.171151	1.000000	-0.175622	-0.160099
days_since_start	1.000000	-0.002113	-0.189944	-0.175622	1.000000	0.992106
month	0.992106	-0.007521	-0.178548	-0.160099	0.992106	1.000000
day_of_week	0.010723	0.024944	0.089426	-0.013518	0.010723	0.009418
day_of_month	0.096719	0.043307	-0.097618	-0.128593	0.096719	-0.028714

In [428]:
```
sns.heatmap(c,annot=True,cmap="Blues")
plt.show()
```



In [429]:
```
ip=sales.drop(["Total Revenue","Date"],axis=1)
op=sales["Total Revenue"]
```

In [399]:
```
ip.head()
```

Out[399]:

	Product Category	Units Sold(Quantity)	days_since_start	month	day_of_week	day_of_month
0	3	2	0	1	0	1
1	4	1	1	1	1	2
2	2	3	2	1	2	3
3	1	4	3	1	3	4
4	0	1	4	1	4	5

In [400]:
```
op.head()
```

Out[400]:
```
0    1999.98
1     499.99
2     209.97
3      63.96
4      89.99
Name: Total Revenue, dtype: float64
```

In [401]:
```
#Train Test Split
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(ip,op,test_size=0.2,random_state=5)
```

In [402]:
```
print(x_train,x_test,y_train,y_test)
```

```
     Product Category  Units Sold(Quantity)  days_since_start  month  \
169                 4                     2               169      6
104                 2                     3               104      4
238                 0                     1               238      8
136                 0                     1               136      5
226                 0                     1               226      8
..                ...                   ...               ...    ...
118                 0                     2               118      4
230                 2                     4               230      8
189                 1                     3               189      7
206                 2                     2               206      7
99                  1                     2                99      4

     day_of_week  day_of_month
169            1            18
104            6            14
238            0            26
136            3            16
226            2            14
..           ...           ...
118            6            28
230            6            18
189            0             8
206            3            25
99             1             9

[192 rows x 6 columns]      Product Category  Units Sold(Quantity)  days_since_start
month  \
180                 3                     1               180      6
162                 3                     1               162      6
119                 5                     1               119      4
187                 4                     2               187      7
46                  0                     1                46      2
161                 5                     2               161      6
192                 3                     1               192      7
60                  3                     3                60      3
177                 1                     3               177      6
55                  4                     1                55      2
56                  2                     5                56      2
102                 3                     1               102      4
237                 1                     3               237      8
96                  3                     3                96      4
122                 2                     5               122      5
89                  5                     2                89      3
225                 1                     3               225      8
223                 4                     2               223      8
6                   3                     1                 6      1
148                 0                     1               148      5
53                  5                     5                53      2
233                 5                     3               233      8
151                 4                     2               151      5
209                 5                     1               209      7
3                   1                     4                 3      1
178                 0                     2               178      6
84                  3                     1                84      3
221                 5                     1               221      8
```

42	3	3	42	2
79	4	1	79	3
61	4	2	61	3
140	2	3	140	5
220	0	1	220	8
51	1	2	51	2
98	2	6	98	4
28	0	1	28	1
216	3	1	216	8
48	3	2	48	2
17	5	4	17	1
37	4	2	37	2
25	4	1	25	1
21	1	2	21	1
26	2	2	26	1
75	1	4	75	3
88	0	1	88	3
200	2	3	200	7
231	1	2	231	8
40	0	1	40	2

	day_of_week	day_of_month
180	5	29
162	1	11
119	0	29
187	5	6
46	4	16
161	0	10
192	3	11
60	4	1
177	2	26
55	6	25
56	0	26
102	4	12
237	6	25
96	5	6
122	3	2
89	5	30
225	1	13
223	6	11
6	6	7
148	1	28
53	4	23
233	2	21
151	4	31
209	6	28
3	3	4
178	3	27
84	0	25
221	4	9
42	0	12
79	2	20
61	5	2
140	0	20
220	3	8
51	2	21

```
98            0            8
28            0           29
216           6            4
48            6           18
17            3           18
37            2            7
25            4           26
21            0           22
26            5           27
75            5           16
88            4           29
200           4           19
231           0           19
40            5           10   169    279.98
104    119.97
238     55.00
136      6.80
226      6.70
        ...
118     29.98
230    259.96
189     32.97
206    196.00
99      25.98
Name: Total Revenue, Length: 192, dtype: float64 180     649.99
162    1199.99
119     249.99
187     398.00
46       52.00
161      69.98
192     199.99
60      899.97
177      35.97
55      499.99
56      149.95
102    3899.99
237      32.97
96      899.97
122      64.95
89      259.98
225      80.97
223     459.98
6      2499.99
148      25.00
53      249.95
233      98.85
151     399.98
209     129.00
3        63.96
178      68.00
84       99.99
221     599.99
42     1199.97
79      499.99
61      399.98
140     134.97
```

```
220     27.00
51      53.98
98     119.94
28      29.99
216   2399.00
48     259.98
17     479.96
37    1599.98
25     179.99
21      45.98
26     359.98
75      59.96
88     105.00
200    179.97
231     19.98
40     105.00
Name: Total Revenue, dtype: float64
```

In [403]:
```
print(x_train.shape,x_test.shape,y_train.shape,y_test.shape)
```

```
(192, 6) (48, 6) (192,) (48,)
```

In [404]:
```
#Standard Scaler Transform
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
```

In [405]:
```
x_train=sc.fit_transform(x_train)
x_test=sc.fit_transform(x_test)
```

In [406]:
```
#Linear Regression
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
```

In [407]:
```
lr.fit(x_train,y_train)
```

Out[407]:

▾ LinearRegression

```
LinearRegression()
```

In [408]:
```
#Prediction
pred=lr.predict(x_test)
pred
```

Out[408]:
```
array([ 219.92611157,  320.62911526,  669.42410461,  238.47275652,
        218.64782683,  476.37022376,  235.04762847,  434.07836473,
         -7.26444776,  541.9777913 ,  267.78475023,  415.14364555,
       -248.0722845 ,  294.04878237,   96.54652597,  625.40810031,
       -132.17251323,   98.44317415,  580.04289506,  138.59020807,
        467.30081295,  213.00687439,  393.77435715,  343.03478108,
        299.90328792,  -69.58518968,  600.76353067,  288.51143482,
        480.40852353,  657.13770652,  558.59835597,  248.03492432,
       -156.74097493,  296.16533633,  144.37573254,  404.26771194,
         64.62451094,  402.26103247,  660.27944312,  589.73174718,
        697.20414124,  451.39168628,  435.56493714,  159.13384951,
        233.10473836,    7.22708758,  -64.02669784,  203.45105924])
```

In [409]:
```
y_test
```

Out[409]:
```
180     649.99
162    1199.99
119     249.99
187     398.00
46       52.00
161      69.98
192     199.99
60      899.97
177      35.97
55      499.99
56      149.95
102    3899.99
237      32.97
96      899.97
122      64.95
89      259.98
225      80.97
223     459.98
6      2499.99
148      25.00
53      249.95
233      98.85
151     399.98
209     129.00
3        63.96
178      68.00
84       99.99
221     599.99
42     1199.97
79      499.99
61      399.98
140     134.97
220      27.00
51       53.98
98      119.94
28       29.99
216    2399.00
48      259.98
17      479.96
37     1599.98
25      179.99
21       45.98
26      359.98
75       59.96
88      105.00
200     179.97
231      19.98
40      105.00
Name: Total Revenue, dtype: float64
```

In [410]:
```
#Accuracy
from sklearn.metrics import mean_squared_error,r2_score

mse=mean_squared_error(pred,y_test)
r2=r2_score(pred,y_test)
```

```
print("MSE :",mse)
print("R2 Score :",r2)
```

```
MSE : 553406.0233966399
R2 Score : -9.096359180143581
```

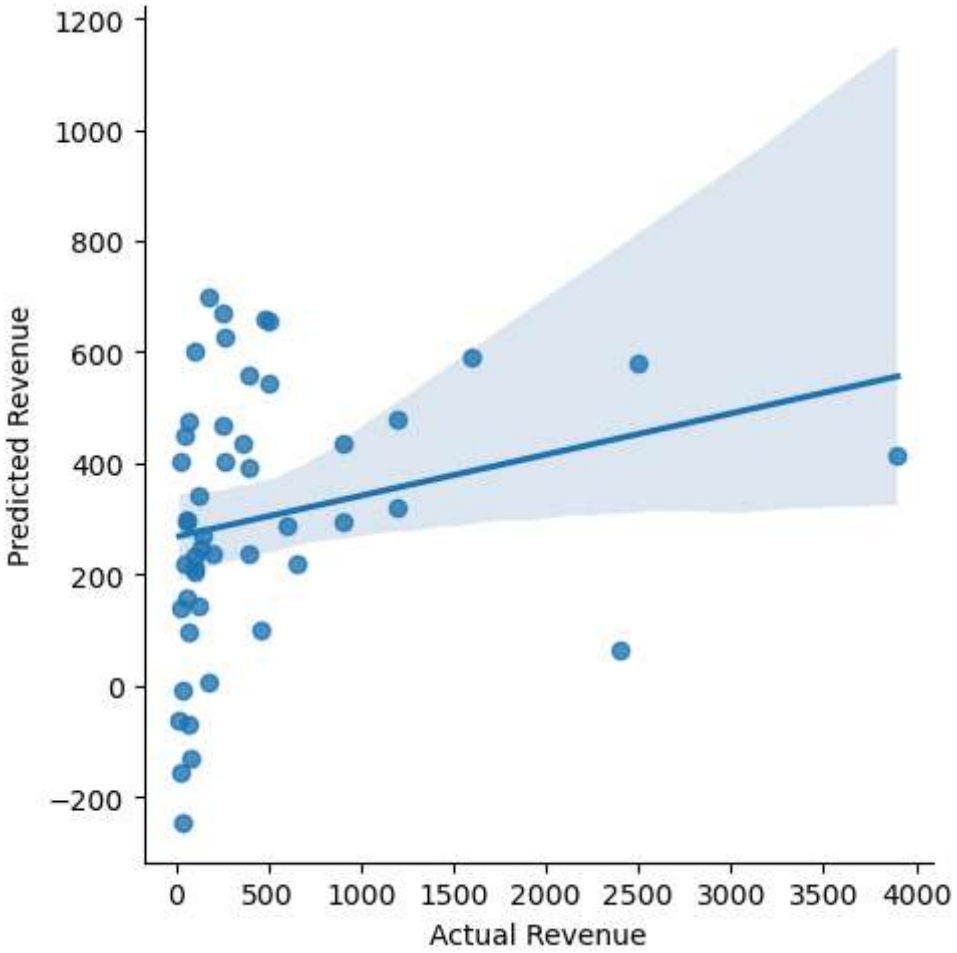
In [411]:
```
#Lmplot -Linear Model Plot
df=pd.DataFrame({"Actual Revenue":list(y_test),"Predicted Revenue":pred})
df
```

Out[411]:

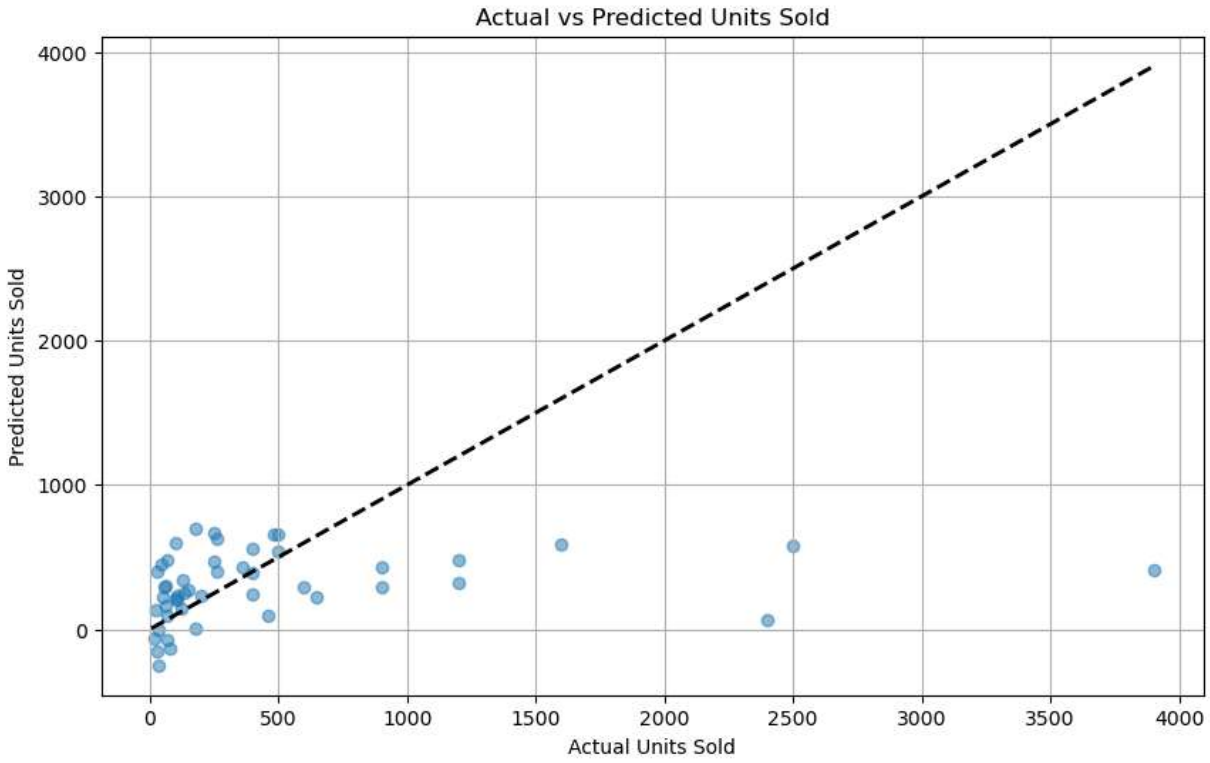
	Actual Revenue	Predicted Revenue
0	649.99	219.926112
1	1199.99	320.629115
2	249.99	669.424105
3	398.00	238.472757
4	52.00	218.647827
5	69.98	476.370224
6	199.99	235.047628
7	899.97	434.078365
8	35.97	-7.264448
9	499.99	541.977791
10	149.95	267.784750
11	3899.99	415.143646
12	32.97	-248.072285
13	899.97	294.048782
14	64.95	96.546526
15	259.98	625.408100
16	80.97	-132.172513
17	459.98	98.443174
18	2499.99	580.042895
19	25.00	138.590208
20	249.95	467.300813
21	98.85	213.006874
22	399.98	393.774357
23	129.00	343.034781
24	63.96	299.903288
25	68.00	-69.585190
26	99.99	600.763531
27	599.99	288.511435
28	1199.97	480.408524
29	499.99	657.137707

	Actual Revenue	Predicted Revenue
30	399.98	558.598356
31	134.97	248.034924
32	27.00	-156.740975
33	53.98	296.165336
34	119.94	144.375733
35	29.99	404.267712
36	2399.00	64.624511
37	259.98	402.261032
38	479.96	660.279443
39	1599.98	589.731747
40	179.99	697.204141
41	45.98	451.391686
42	359.98	435.564937
43	59.96	159.133850
44	105.00	233.104738
45	179.97	7.227088
46	19.98	-64.026698
47	105.00	203.451059

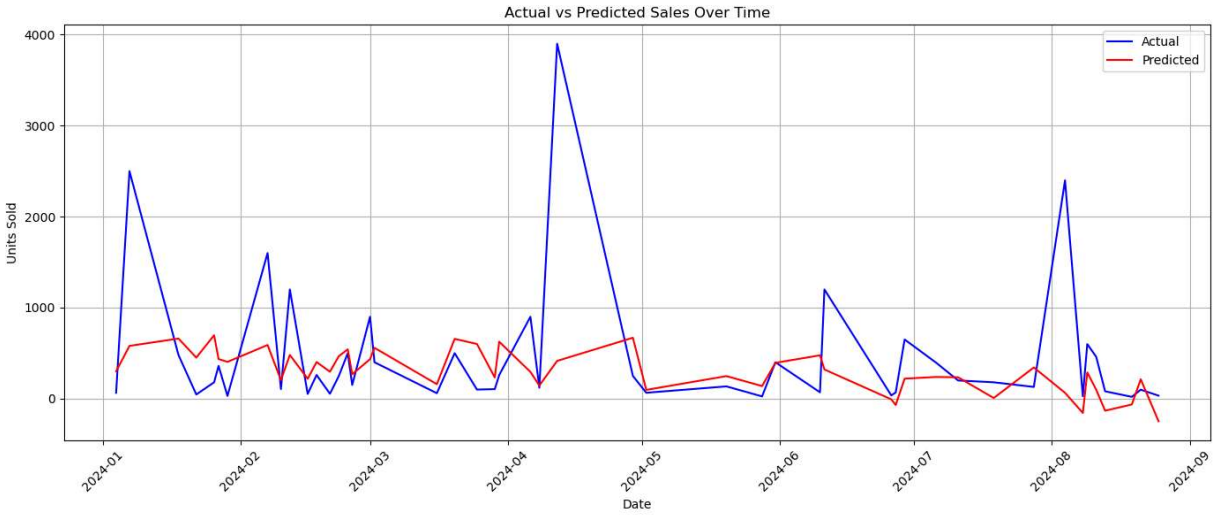
In [412]:
```
sns.lmplot(x="Actual Revenue",y="Predicted Revenue",data=df)
plt.show()
```



In [413]:
```
# Actual vs Predicted plot
plt.figure(figsize=(10, 6))
plt.scatter(y_test,pred, alpha=0.5)
plt.plot([op.min(), op.max()], [op.min(), op.max()], 'k--', lw=2)
plt.xlabel('Actual Units Sold')
plt.ylabel('Predicted Units Sold')
plt.title('Actual vs Predicted Units Sold')
plt.grid(True)
plt.show()
```



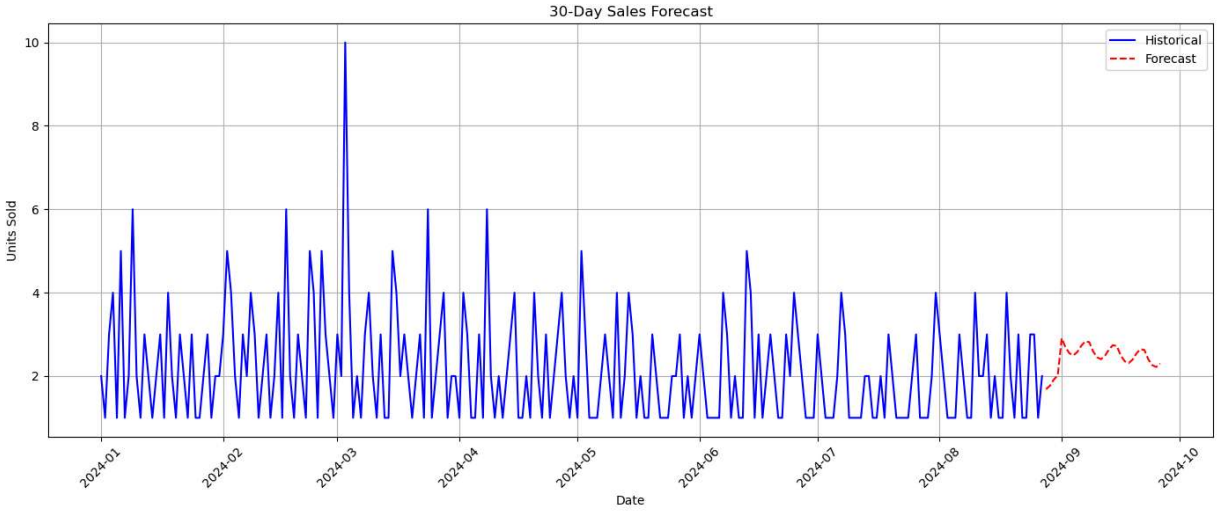
In [450]:
```
# Time series comparison
results = pd.DataFrame({
    'Date': sales.loc[y_test.index, 'Date'].values,  # Use y_test.index which should
    'Actual': y_test,
    'Predicted': pred
}).sort_values('Date')

plt.figure(figsize=(14, 6))
plt.plot(results['Date'], results['Actual'], 'b-', label='Actual')
plt.plot(results['Date'], results['Predicted'], 'r-', label='Predicted')
plt.xlabel('Date')
plt.ylabel('Units Sold')
plt.title('Actual vs Predicted Sales Over Time')
plt.legend()
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```



In [446]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from datetime import datetime, timedelta
from sklearn.linear_model import LinearRegression
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.pipeline import Pipeline

## Step 1: Load and Prepare Data ##
# Load your data (replace with your actual data loading code)
# sales = pd.read_csv('Online Sales Data.csv')

# Convert Date to datetime and create features
sales['Date'] = pd.to_datetime(sales['Date'])

# Correct way to calculate days since start
sales['days_since_start'] = (sales['Date'] - sales['Date'].min()).dt.days  # Fixed: c

# Create cyclical date features
sales['month'] = sales['Date'].dt.month
sales['day_of_week'] = sales['Date'].dt.dayofweek
sales['day_of_month'] = sales['Date'].dt.day
sales['month_sin'] = np.sin(2 * np.pi * sales['month']/12)
sales['month_cos'] = np.cos(2 * np.pi * sales['month']/12)
sales['dow_sin'] = np.sin(2 * np.pi * sales['day_of_week']/7)
sales['dow_cos'] = np.cos(2 * np.pi * sales['day_of_week']/7)

## Step 2: Define Preprocessor and Model ##
# Define which columns are categorical vs numerical
categorical_cols = ['Product Category']
numerical_cols = ['days_since_start', 'month', 'day_of_week', 'day_of_month',
                  'month_sin', 'month_cos', 'dow_sin', 'dow_cos',
                  'Total Revenue']  # Removed 'Units Sold' as it's our target

# Create preprocessor
preprocessor = ColumnTransformer(
    transformers=[
        ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_cols),
        ('num', 'passthrough', numerical_cols)
```

```
    ])

# Create model pipeline
model = Pipeline([
    ('preprocessor', preprocessor),
    ('regressor', LinearRegression())
])

## Step 3: Train the Model ##
X = sales.drop('Units Sold(Quantity)', axis=1)  # Features
y = sales['Units Sold(Quantity)']  # Target

model.fit(X, y)

## Step 4: Prepare Future Data ##
last_date = sales['Date'].max()
future_dates = pd.date_range(start=last_date + pd.Timedelta(days=1), periods=30, freq

future_df = pd.DataFrame({
    'Product Category': ['Sports']*30,  # Most frequent category
    'Date': future_dates,
    'days_since_start': (future_dates - sales['Date'].min()).days.values,  # Fixed: u
    'month': future_dates.month,
    'day_of_week': future_dates.dayofweek,
    'day_of_month': future_dates.day,
    'month_sin': np.sin(2 * np.pi * future_dates.month/12),
    'month_cos': np.cos(2 * np.pi * future_dates.month/12),
    'dow_sin': np.sin(2 * np.pi * future_dates.dayofweek/7),
    'dow_cos': np.cos(2 * np.pi * future_dates.dayofweek/7),
    'Total Revenue': [0]*30  # Placeholder
})

## Step 5: Make Predictions ##
future_predictions = model.predict(future_df)

## Step 6: Visualize Results ##
plt.figure(figsize=(14, 6))
plt.plot(sales['Date'], sales['Units Sold(Quantity)'], 'b-', label='Historical')
plt.plot(future_dates, future_predictions, 'r--', label='Forecast')
plt.xlabel('Date')
plt.ylabel('Units Sold')
plt.title('30-Day Sales Forecast')
plt.legend()
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()

## Step 7: Create Forecast Table ##
forecast_table = pd.DataFrame({
    'Date': future_dates,
    'Predicted Units Sold': future_predictions.round(1)
})
print("\n30-Day Sales Forecast:")
print(forecast_table)
```



30-Day Sales Forecast:

	Date	Predicted Units Sold
0	2024-08-28	1.7
1	2024-08-29	1.8
2	2024-08-30	1.9
3	2024-08-31	2.0
4	2024-09-01	2.9
5	2024-09-02	2.7
6	2024-09-03	2.5
7	2024-09-04	2.5
8	2024-09-05	2.6
9	2024-09-06	2.7
10	2024-09-07	2.8
11	2024-09-08	2.8
12	2024-09-09	2.6
13	2024-09-10	2.4
14	2024-09-11	2.4
15	2024-09-12	2.5
16	2024-09-13	2.6
17	2024-09-14	2.7
18	2024-09-15	2.7
19	2024-09-16	2.5
20	2024-09-17	2.4
21	2024-09-18	2.3
22	2024-09-19	2.4
23	2024-09-20	2.5
24	2024-09-21	2.6
25	2024-09-22	2.6
26	2024-09-23	2.4
27	2024-09-24	2.3
28	2024-09-25	2.2
29	2024-09-26	2.3