Supply Chain Analysis

In [3]:
```
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.io as pio
import plotly.figure_objects as go
pio.templates.default = "plotly_white"
```

In [4]:
```
data = pd.read_csv(r"B:\DATA ANALYST\Project\Supply_chain.csv")
data.head()
```

Out[6]:

	Product type	SKU	Price	Availability	Number of products sold	Revenue generated	Customer demographics	Stock levels	Lead times	Order quantities	...	Location	Lead time	Production volumes	Manufacturing lead time	Manufacturing costs	Inspection results	Defect rates	Transportation modes	Routes	Costs
0	haircare	SKU0	69.808006		55	802	8661.996792	Non-binary	58	7	96	...	Mumbai	29	215	29	46.279879	Pending	0.226410	Road	Route B	187.752075
1	skincare	SKU1	14.843523		95	736	7460.900065	Female	53	30	37	...	Mumbai	23	517	30	33.616769	Pending	4.854068	Road	Route B	503.065579
2	haircare	SKU2	11.319683		34	8	9577.749626	Unknown	1	10	88	...	Mumbai	12	971	27	30.688019	Pending	4.580593	Air	Route C	141.920282
3	skincare	SKU3	61.163343		68	83	7756.836426	Non-binary	23	13	59	...	Kolkata	24	937	18	35.624741	Fail	4.746649	Rail	Route A	254.776159
4	skincare	SKU4	4.805496		26	871	2686.505152	Non-binary	5	3	56	...	Delhi	5	414	3	92.065161	Fail	3.145580	Air	Route A	923.440632

5 rows × 24 columns

In [7]:
```
data.info()
```

```
<class "pandas.core.frame.DataFrame">
RangeIndex: 100 entries, 0 to 99
Data columns (total 24 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Product type            100 non-null    object
 1   SKU                     100 non-null    object
 2   Price                   100 non-null    float64
 3   Availability            100 non-null    int64
 4   Number of products sold 100 non-null    int64
 5   Revenue generated       100 non-null    float64
 6   Customer demographics   100 non-null    object
 7   Stock levels            100 non-null    int64
 8   Lead times              100 non-null    int64
 9   Order quantities        100 non-null    int64
 10  Shipping times          100 non-null    int64
 11  Shipping carriers       100 non-null    object
 12  Shipping costs          100 non-null    float64
 13  Supplier name           100 non-null    object
 14  Location                100 non-null    object
 15  Lead time               100 non-null    int64
 16  Production volumes      100 non-null    int64
 17  Manufacturing lead time 100 non-null    int64
 18  Manufacturing costs     100 non-null    float64
 19  Inspection results      100 non-null    object
 20  Defect rates            100 non-null    float64
 21  Transportation modes    100 non-null    object
 22  Routes                  100 non-null    object
 23  Costs                   100 non-null    float64
dtypes: float64(8), int64(9), object(9)
memory usage: 18.9+ KB
```

In [8]:
```
data.drop('Routes', axis=1, inplace=True)
```

In [9]:
```
data.rename(columns={'Location': 'Shipping Location'}, inplace=True)
```

In [10]:
```
data.columns
```

Out[10]:
```
Index(['Product type', 'SKU', 'Price', 'Availability',
       'Number of products sold', 'Revenue generated',
       'Customer demographics', 'Stock levels', 'Lead times',
       'Order quantities', 'Shipping times', 'Shipping carriers',
       'Shipping costs', 'Supplier name', 'Shipping Location',
       'Lead time', 'Production volumes', 'Manufacturing lead time',
       'Manufacturing costs', 'Inspection results', 'Defect rates',
       'Transportation modes', 'Costs'],
      dtype='object')
```

In [11]:
```
data[['Price', 'Number of products sold', 'Revenue generated', 'Order quantities', 'Shipping costs', 'Manufacturing costs', 'Costs']].describe()
```

Out[11]:

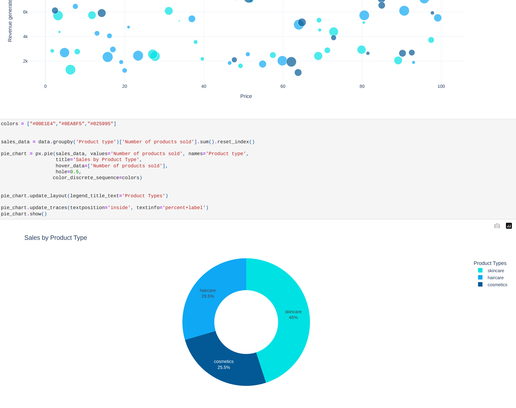
	Price	Number of products sold	Revenue generated	Order quantities	Shipping costs	Manufacturing costs	Costs
count	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000
mean	49.462461	460.990000	5776.048187	49.220000	5.544519	47.266693	529.291782
std	31.169193	303.780074	2732.841744	26.784429	2.851376	28.962641	258.301696
min	1.699976	8.000000	1001.618523	1.000000	1.013487	1.035069	103.918248
25%	19.597823	184.250000	2812.847151	26.000000	3.540248	22.983299	318.778455
50%	51.239930	392.500000	6006.352023	52.000000	5.320534	45.905622	520.438444
75%	77.198228	704.250000	8253.976935	71.250000	7.601695	68.623155	763.078231
max	99.171329	996.000000	9866.465458	96.000000	9.929816	99.466109	997.413450

In [12]:
```
product_type_data = data.groupby('Product type')[['Number of products sold', 'Revenue generated',
                                                  'Manufacturing costs', 'Shipping costs','Costs']].sum().reset_index()

product_type_data
```

Out[12]:

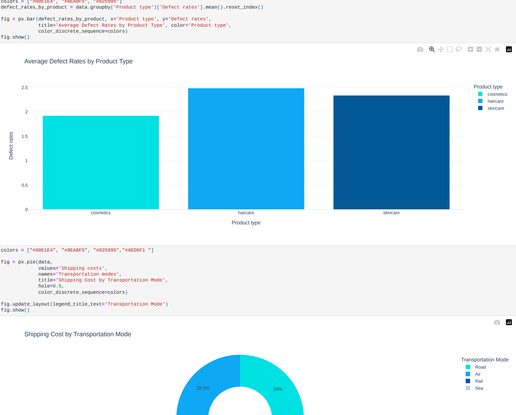
	Product type	Number of products sold	Revenue generated	Manufacturing costs	Shipping costs	Costs
0	cosmetics	11757	161521.266001	1119.371253	157.563663	13366.397283
1	haircare	13611	174455.390806	1647.571776	200.863735	17328.882865
2	skincare	20731	241628.162133	1959.726295	196.307510	22229.318668

In [13]:
```
colors = ["#00E1E4","#0EA0F5","#025995"]

fig = px.scatter(data, x='Price',
                 y='Revenue generated',
                 title='Comparing Price and Revenue across Product Types',
                 color='Product type',
                 hover_data=['Number of products sold','SKU'], size='Number of products sold',color_discrete_sequence=colors)

fig.show()
```



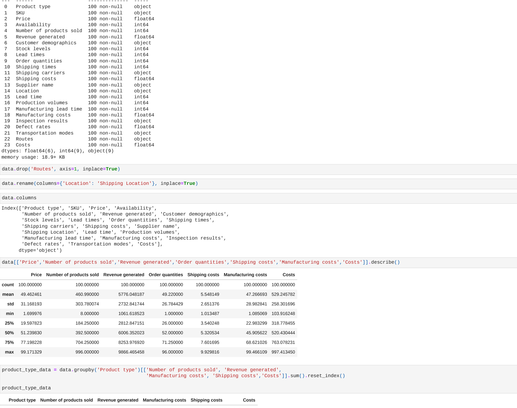
In [14]:
```
colors = ["#00E1E4","#0EA0F5","#025995"]

sales_data = data.groupby('Product type')['Number of products sold'].sum().reset_index()

pie_chart = px.pie(sales_data, values='Number of products sold', names='Product type',
                   title='Sales by Product Type',
                   hover_data=['Number of products sold'],
                   hole=0.5,
                   color_discrete_sequence=colors)


pie_chart.update_layout(legend_title_text='Product Types')

pie_chart.update_traces(textposition='inside', textinfo='percent+label')
pie_chart.show()
```

In [15]:
```
revenue_chart = px.line(data, x='SKU',
                        y='Revenue generated',
                        title='Revenue Generated by SKU')

revenue_chart.update_traces(line_color='deepskyblue')
revenue_chart.show()
```

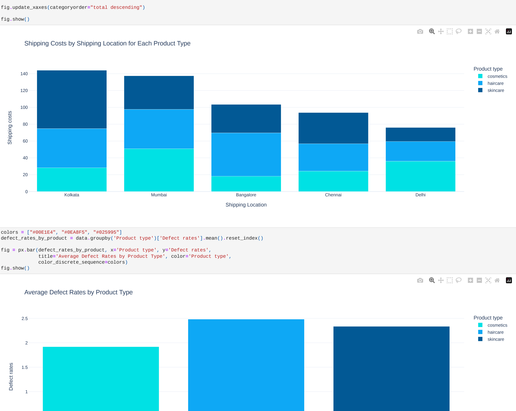


SKU stands for Stock Keeping Unit. " it is a unique identifier or code assigned to a specific product or item to track its inventory and sales."

In [16]:
```
stock_chart = px.line(data, x='SKU',
                      y='Stock levels',
                      title='Stock Levels by SKU')

stock_chart.update_traces(line_color='blue')
stock_chart.show()
```



In [21]:
```
colors = ["#00E1E4","#0EA0F5","#025995"]

product_location_shipping_costs = data.groupby(['Product type', 'Shipping Location'])['Shipping costs'].sum().reset_index()

fig = px.bar(product_location_shipping_costs, x='Shipping Location', y='Shipping costs',
             title='Shipping Costs by Shipping Location for Each Product Type', color='Product type',
             color_discrete_sequence=colors)

fig.update_xaxes(categoryorder="total descending")

fig.show()
```



In [18]:
```
colors = ["#00E1E4", "#0EA0F5", "#025995"]
defect_rates_by_product = data.groupby('Product type')['Defect rates'].mean().reset_index()

fig = px.bar(defect_rates_by_product, x='Product type', y='Defect rates',
             title='Average Defect Rates by Product Type', color='Product type',
             color_discrete_sequence=colors)

fig.show()
```


In [19]:
```
colors = ["#00E1E4", "#0EA0F5", "#025995","#AEDEF1 "]

fig = px.pie(data,
             values='Shipping costs',
             names='Transportation modes',
             title='Shipping Cost by Transportation Mode',
             hole=0.5,
             color_discrete_sequence=colors)

fig.update_layout(legend_title_text='Transportation Mode')
fig.show()
```