EXPLORATORY DATA ANALYSIS ON FASHION INVENTORY DATA

A. Importing Important Libaries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
# Define the file path
file_path = r"C:\Users\user\Downloads\Sales Fashion Inventory 1.xlsx"

# Load the dataset into a Pandas DataFrame
df = pd.read_excel(file_path)
```

In [25]:
```
df.head()
```

Out[25]:

	ID	Date	Months	Time	Product ID	Size	Size Category	Brand	Dominant Material	Dominant Color	Product Type	Variant Price	C…
0	30000	2019-08-14	Aug	05:49:46	602471	XL	Adult Size	Imara	Polyester	Black	Top	7970	
1	30001	2019-07-06	Jul	01:04:34	604241	XS	Adult Size	Global Desi	viscose	Red	Top	10190	
2	30002	2019-07-18	Jul	09:43:31	606353	XL	Adult Size	Rangriti	cotton	Beige	Printed Kurta	10490	
3	30003	2019-07-18	Jul	14:11:14	611890	5-6Y	Kids Size	Pspeaches	Cotton	Rust	Kurta with Sharara & Dupatta	5990	
4	30004	2019-04-06	Apr	08:17:28	608783	M	Adult Size	Libas	Cotton	Mustard	Straight Kurta	22900	

In [4]:
```
df.info()
df.isnull().sum()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15000 entries, 0 to 14999
Data columns (total 15 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   ID                          15000 non-null  int64
 1   Date                        15000 non-null  datetime64[ns]
 2   Months                      15000 non-null  object
 3   Time                        15000 non-null  object
 4   Product ID                  15000 non-null  int64
 5   Size                        15000 non-null  object
 6   Size Category               15000 non-null  object
 7   Brand                       15000 non-null  object
 8   Dominant Material           15000 non-null  object
 9   Dominant Color              15000 non-null  object
 10  Product Type                15000 non-null  object
 11  Variant Price               15000 non-null  int64
 12  Variant Compare At Price    15000 non-null  int64
 13  Ideal For                   15000 non-null  object
 14  Is In Stock                 15000 non-null  object
dtypes: datetime64[ns](1), int64(4), object(10)
memory usage: 1.7+ MB
```

Out[4]:
```
ID                          0
Date                        0
Months                      0
Time                        0
Product ID                  0
Size                        0
Size Category               0
Brand                       0
Dominant Material           0
Dominant Color              0
Product Type                0
Variant Price               0
Variant Compare At Price    0
Ideal For                   0
Is In Stock                 0
dtype: int64
```

B. Creating a 'Discount' column to capture how much discount was offered.

In [5]:
```
# Add a discount column
df['Discount (%)'] = ((df['Variant Compare At Price'] - df['Variant Price']) / df['Variant C
```

In [6]:
```
df.head(10)
```

Out[6]:

	ID	Date	Months	Time	Product ID	Size	Size Category	Brand	Dominant Material	Dominant Color	Product Type	Variant Price	C
0	30000	2019-08-14	Aug	05:49:46	602471	XL	Adult Size	Imara	Polyester	Black	Top	7970	
1	30001	2019-07-06	Jul	01:04:34	604241	XS	Adult Size	Global Desi	viscose	Red	Top	10190	
2	30002	2019-07-18	Jul	09:43:31	606353	XL	Adult Size	Rangriti	cotton	Beige	Printed Kurta	10490	
3	30003	2019-07-18	Jul	14:11:14	611890	5-6Y	Kids Size	Pspeaches	Cotton	Rust	Kurta with Sharara & Dupatta	5990	
4	30004	2019-04-06	Apr	08:17:28	608783	M	Adult Size	Libas	Cotton	Mustard	Straight Kurta	22900	
5	30005	2019-07-31	Jul	13:03:52	602791	XXL	Adult Size	Libas	Viscose Rayon	Navy	Kurta with Palazzos	22100	
6	30006	2019-06-20	Jun	05:21:59	600971	XL	Adult Size	Vastramay	Cotton	White	Kurta with Churidar	5590	
7	30007	2019-07-18	Jul	04:00:15	610992	40	Numeric Size	Anouk	Null	Blue	Null	18990	
8	30008	2019-06-20	Jun	00:52:43	600139	XS	Adult Size	Anouk	viscose	Green	A-Line Kurta	4990	
9	30009	2019-02-08	Feb	07:24:13	610796	S	Adult Size	Taavi	cotton	Pink	Straight Kurta	7790	

C. Summary Statistics

In [7]:
```
# Describe numerical columns
df[['Variant Price', 'Variant Compare At Price', 'Discount (%)']].describe()
```

Out[7]:

	Variant Price	Variant Compare At Price	Discount (%)
count	15000.000000	15000.000000	15000.000000
mean	13726.709333	23651.510000	39.712831
std	11458.483235	17554.184219	24.762293
min	2390.000000	3500.000000	0.000000
25%	7170.000000	13990.000000	20.013342
50%	9990.000000	18990.000000	50.022738
75%	15990.000000	27990.000000	60.024010
max	255000.000000	255000.000000	80.066722

D. Categorical Feature Distribution to understand the most popular brands and types of fashion products, ideal customers, month etc.

In [8]:
```
# Most frequent brands
df['Brand'].value_counts().head(10)
```

Out[8]:
```
Brand
Anouk          1725
Biba           1093
Libas           811
Shree           791
Global Desi     772
Fabindia        399
Imara           376
Sangria         340
Deyann          292
Gerua           289
Name: count, dtype: int64
```

In [9]:
```
# Most common product types
df['Product Type'].value_counts().head(10)
```

Out[9]:
```
Product Type
Straight Kurta                  3724
A-Line Kurta                    1590
Null                            1420
Kurta with Churidar              716
Kurta with Pyjamas               712
Kurta with Palazzos              428
Maxi Dress                       422
A-Line Dress                     346
Top                              318
Kurta with Churidar & Dupatta    238
Name: count, dtype: int64
```

In [10]:
```
# Most common ideal costomers
df['Ideal For'].value_counts()
```

Out[10]:
```
Ideal For
Women     9205
Men       3943
Girls      701
Boys       591
Unisex     560
Name: count, dtype: int64
```

In [11]:
```
# Most frequented month
df['Months'].value_counts()
```

Out[11]:
```
Months
Aug    6786
Jul    2830
Jun    2378
Apr    1112
May     613
Nov     600
Sep     417
Feb     264
Name: count, dtype: int64
```

In [12]:
```
# Stock distribution
df['Is In Stock'].value_counts()
```

Out[12]:
```
Is In Stock
In Stock        7610
Out of Stock    7322
Null              68
Name: count, dtype: int64
```

E. STATISTICAL ANALYSIS

1. Chi-Squared Test: Are Discounts Causing Stockouts?

Purpose: To test whether heavily discounted products (>= 50%) are more likely to be out of stock.

In [13]:
```
from scipy.stats import chi2_contingency

# Create "Heavy Discount" flag
df['Heavy Discount'] = np.where(df['Discount (%)'] >= 50, 'Yes', 'No')

# Crosstab of Heavy Discount vs Stock Status
contingency_table = pd.crosstab(df['Heavy Discount'], df['Is In Stock'])

# Run chi-squared test
chi2, p_value, dof, expected = chi2_contingency(contingency_table)

print("Chi-squared:", chi2)
print("p-value:", p_value)
```

```
Chi-squared: 1.0135830672345778
p-value: 0.6024253428258771
```

Interpretation

The analysis indicates that high discount levels (≥ 50%) do not have a statistically significant relationship with stockouts. In other words, products with heavy discounts are not necessarily more likely to be out of stock. This suggests that factors beyond pricing such as brand appeal, seasonal demand patterns, or limited size availability may be more influential in driving product unavailability. Therefore, while discounting is a common promotional tactic, it may not be the primary driver of inventory depletion in this dataset.

2.Correlation

Why It Matters for Business:

Comparing these variables helps you analyze pricing strategies.

You can see if:

-Some brands always sell at full price.

-Others rely heavily on discounting to move inventory.

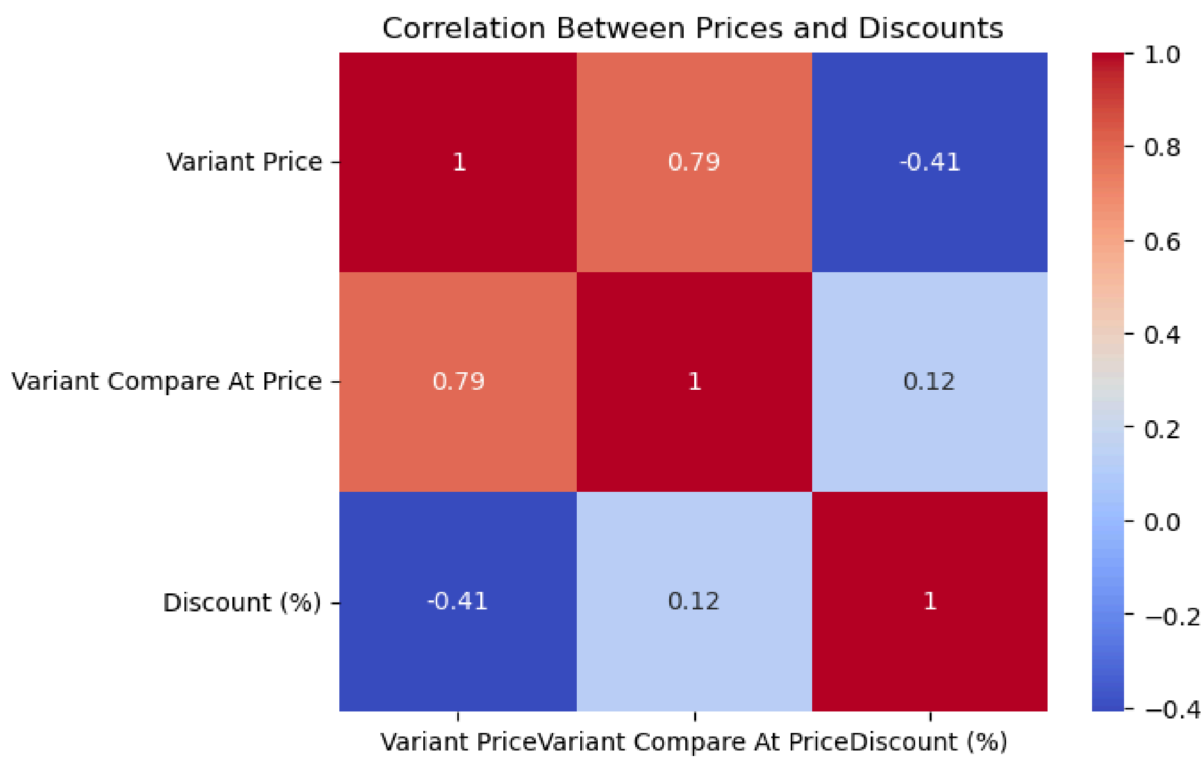
-It helps identify whether price slashing actually helps with stock clearance or not.

In [15]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Select only numerical columns
corr_data = df[['Variant Price', 'Variant Compare At Price', 'Discount (%)']]

# Create the correlation matrix
corr_matrix = corr_data.corr()

# Plot the heatmap
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title("Correlation Between Prices and Discounts")
plt.show()
```



Interpretation:

Variant Price vs. Compare At Price:

Correlation: 0.79

This is a strong positive correlation. It means that when the original (Compare At) price is high, the actual selling price is usually also high. That's expected—higher original prices usually lead to higher final prices.

Variant Price vs. Discount (%):

Correlation: -0.41

This is a moderate negative correlation. It suggests that higher-priced products tend to have lower discount percentages. Or, to say it differently, the cheaper the product, the more likely it is to have a bigger discount.

Compare At Price vs. Discount (%):

Correlation: 0.12

This is a very weak positive correlation. It means there’s almost no clear relationship between the original listed price and the discount percentage

F. Discount (%) Feature Analysis

Average discount by:

1.Month

2.Brand

3.Stock status (Is In Stock)

In [21]:
```
avg_discount_by_month = df.groupby('Months')['Discount (%)'].mean().reset_index()
avg_discount_by_month = avg_discount_by_month.sort_values(by='Discount (%)', ascending=False
print(avg_discount_by_month)
```

```
  Months  Discount (%)
2    Feb     41.488139
3    Jul     40.527357
6    Nov     39.622047
1    Aug     39.523143
4    Jun     39.508150
0    Apr     39.466968
7    Sep     39.258634
5    May     38.925601
```

In [22]:
```
avg_discount_by_brand = df.groupby('Brand')['Discount (%)'].mean().reset_index()
avg_discount_by_brand = avg_discount_by_brand.sort_values(by='Discount (%)', ascending=False
print(avg_discount_by_brand)
```

```
                   Brand  Discount (%)
40          Athom Trendz     70.043777
46   Balance By Rohit Bal     70.043777
349               Vulcan     70.043777
218              Peaches     70.043618
324         Tokyo Talkies     70.036862
..                   ...           ...
84             Cotton On      0.000000
320  The Silhouette Store      0.000000
302        Styles Closet      0.000000
313       Tales & Stories      0.000000
196                  Mbe      0.000000

[364 rows x 2 columns]
```

In [19]:
```
avg_discount_by_stock = df.groupby('Is In Stock')['Discount (%)'].mean().reset_index()
print(avg_discount_by_stock)
```

```
    Is In Stock  Discount (%)
0      In Stock     39.816873
1          Null     40.610939
2  Out of Stock     39.596354
```

In [23]:
```
avg_discount_combined = df.groupby(['Months', 'Brand', 'Is In Stock'])['Discount (%)'].mean(
avg_discount_combined = avg_discount_combined.sort_values(by='Discount (%)', ascending=False
print(avg_discount_combined)
```

```
     Months           Brand  Is In Stock  Discount (%)
1422    Jun             Noi     In Stock     80.030781
1324    Jun  Faballey Indya  Out of Stock    77.519380
1821    Nov            Manu     In Stock     76.004750
1643    May    Kuons Avenue     In Stock     75.015674
918     Jul         Bianca  Out of Stock     72.516258
...     ...             ...          ...           ...
992     Jul          Globus  Out of Stock     0.000000
981     Jul        Folklore     In Stock      0.000000
1488    Jun        Street 9     In Stock      0.000000
1849    Nov    Raymond Home     In Stock      0.000000
1566    May        Anekaant     In Stock      0.000000

[2049 rows x 4 columns]
```

In [24]:
```
avg_discount_by_ideal_for = df.groupby('Ideal For')['Discount (%)'].mean().reset_index()
avg_discount_by_ideal_for = avg_discount_by_ideal_for.sort_values(by='Discount (%)', ascendi

print(avg_discount_by_ideal_for)
```

```
  Ideal For  Discount (%)
1     Girls     40.303432
4     Women     39.819236
2       Men     39.790272
3    Unisex     38.565677
0      Boys     37.925314
```

In [26]:
```
heavy_discount_counts = df['Heavy Discount'].value_counts().reset_index()
heavy_discount_counts.columns = ['Heavy Discount', 'Count']

print(heavy_discount_counts)
```

```
  Heavy Discount  Count
0            Yes   8123
1             No   6877
```

G. Margin Segment

What Does "Margin" Mean in Retail? Margin = (Selling Price – Cost Price) / Selling Price

It tells you how much profit you're making per product sold.

But since we don’t have cost price in this dataset, I will use discount percentage as a proxy:

High Discount (%) → Likely Low Margin

Low/No Discount (%) → Likely High Margin

Why Segment Products by Margin Potential?

High Discounts (≥50%):

These are likely clearance items or overstock.

They earn little profit per sale.

You might want to stop restocking these or negotiate better wholesale prices.

Low Discounts (<20% or 0%):

These are your premium or high-margin products.

In [28]:
```
def margin_segment(discount):
    if discount >= 50:
        return 'Low Margin'
    elif discount < 20:
        return 'High Margin'
    else:
        return 'Medium Margin'

df['Margin Segment'] = df['Discount (%)'].apply(margin_segment)
df['Margin Segment'].value_counts()
```

Out[28]:
```
Margin Segment
Low Margin       8123
High Margin      3519
Medium Margin    3358
Name: count, dtype: int64
```

Interpretation:

Most of the products in your inventory are Low Margin, about 8,123 items fall into this category. This suggests that a large portion of your catalog may have limited profit per item.

You also have:

3,519 High Margin products — items likely offering better profits.

3,358 Medium Margin products — sitting in the middle ground.

What This Tells You:

1. You might be relying heavily on volume (selling more items with lower margins).
2. There may be room to focus on promoting or stocking more high-margin products to improve profitability.
3. You can also analyze whether low-margin items are getting the most discounts or going out of stock faster.

In [ ]: