Operational Breakdown Analysis: Finding Root Causes Behind Revenue Loss

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import psycopg2
from sqlalchemy import create_engine
import warnings
warnings.filterwarnings('ignore')
import os
```

Connecting pgadmin to jupyter

In [2]:
```
user = 'postgres'
password = '93805'
host = 'localhost'
port = '5432'
database = 'Operational_Breakdown_Analysis'
```

In [3]:
```
engine = create_engine(f'postgresql+psycopg2://{user}:{password}@{host}:{port}
```

In [4]:
```
# loading data from pgadmin server

orders = pd.read_sql("SELECT * FROM orders", engine)
order_items = pd.read_sql("SELECT * FROM orderitems", engine)
root_causes = pd.read_sql("SELECT * FROM rootcauses", engine)
shipments = pd.read_sql("SELECT * FROM shipments", engine)
inventory_transactions = pd.read_sql("SELECT * FROM inventorytransactions", er
it_incidents = pd.read_sql("SELECT * FROM itincidents", engine)
support_tickets = pd.read_sql("SELECT * FROM supporttickets", engine)
products = pd.read_sql("SELECT * FROM products", engine)
vendors = pd.read_sql("SELECT * FROM vendors", engine)
customers = pd.read_sql("SELECT * FROM customers", engine)
systems = pd.read_sql("SELECT * FROM systems", engine)
warehouses = pd.read_sql("SELECT * FROM warehouses", engine)
purchase_orders = pd.read_sql("SELECT * FROM purchaseorders", engine)
```

Exploring Dataset

In [5]:
```
# Dictionary of table names and DataFrames
tables = {
    "Orders": orders,
    "OrderItems": order_items,
    "RootCauses": root_causes,
    "Shipments": shipments,
    "InventoryTransactions": inventory_transactions,
    "ITIncidents": it_incidents,
    "SupportTickets": support_tickets,
    "Products": products,
    "Vendors": vendors,
    "Customers": customers,
    "Systems": systems,
    "Warehouses": warehouses,
    "PurchaseOrders": purchase_orders
}

# Loop to preview each table
for name, df in tables.items():
    print(f"\n� Table: {name}")
    print(df.head())
```

```
� Table: Orders
   order_id  customer_id order_date   due_date order_status  order_total  \
0         1         2849 2025-05-24 2025-05-25    Completed      6077.68   
1         2         7624 2025-04-29 2025-05-03    Completed       711.52   
2         3        13849 2025-03-16 2025-03-21    Completed      5642.72   
3         4        12450 2024-12-22 2024-12-25    Cancelled      2738.60   
4         5         9607 2024-08-19 2024-08-22      Delayed      9504.11   

  delivery_status  delay_reason_id  
0         On-Time                1  
1         On-Time                1  
2         On-Time                1  
3            Lost                2  
4         Delayed                3  

� Table: OrderItems
   order_item_id  order_id  product_id  quantity  unit_price  line_total
0              1         1         251         2      494.19      988.38
1              2         1         143         1      671.30      671.30
2              3         1         693         5      883.60     4418.00
3              4         2          96         2      355.76      711.52
4              5         3         204         5      339.52     1697.60

� Table: RootCauses
   root_cause_id   category          description severity_level
0              1  Logistics        Carrier Delay         Medium
1              2  Inventory             Stockout           High
2              3         IT            IT Outage       Critical
3              4    Support  Customer Escalation         Medium
4              5     Vendor         Vendor Delay           High

� Table: Shipments
   order_id dispatch_date arrival_date delivery_status  delay_days  \
0         1    2025-05-25   2025-05-30         Delayed           5   
1         2    2025-05-01   2025-05-01         On-Time           0   
2         3    2025-03-17   2025-03-17         On-Time           0   
3         4    2024-12-24   2024-12-24         On-Time           0   
4         5    2024-08-21   2024-08-21         On-Time           0   

   delay_reason  
0      IT Issue  
1  Manual Error  
2        Vendor  
3       Carrier  
4  Manual Error  

� Table: InventoryTransactions
   trans_id  product_id  warehouse_id trans_date  quantity_change trans_type  \
0         1         251             4 2025-05-24               -2       Sale   
1         2         143            10 2025-05-24               -1       Sale   
2         3         693             8 2025-05-24               -5       Sale   
3         4          96             9 2025-04-29               -2       Sale   
4         5         204             6 2025-03-16               -5       Sale   
```

```
   reference_id
0             1
1             1
2             1
3             2
4             3

🔷 Table: ITIncidents
   incident_id  system_id start_time            end_time  duration_hours  \
0            1          1 2024-07-01 2024-07-01 01:03:34        1.059441
1            2          2 2024-07-02 2024-07-02 01:12:15        1.204221
2            3          2 2024-07-02 2024-07-02 01:39:32        1.658908
3            4          5 2024-07-03 2024-07-03 01:41:00        1.683254
4            5          5 2024-07-04 2024-07-04 01:58:34        1.976001

  severity  impacted_orders  estimated_revenue_loss
0      Low            75667             36380.02433
1      Low            70708             18279.00654
2      Low           135701             16388.93073
3      Low           129659             34592.02958
4      Low            45066             13910.46518

🔷 Table: SupportTickets
   ticket_id  order_id  product_id created_date       resolved_date  \
0          1        65         157   2024-11-04 2024-11-04 18:49:50
1          2       117         317   2025-06-13 2025-06-13 12:27:26
2          3       123         483   2025-01-12 2025-01-12 18:26:17
3          4       188         373   2024-11-18 2024-11-18 12:06:56
4          5       206         255   2024-11-26 2024-11-26 11:23:16

  issue_type  resolution_time_hrs  escalated  satisfaction_rating  \
0     Damage            18.830686      False                    1
1     Damage            12.457323      False                    5
2     Damage            18.438022      False                    5
3  TechIssue            12.115649      False                    4
4      Delay            11.387648       True                    1

   support_loss
0      0.000000
1      0.000000
2      0.000000
3      0.000000
4   2963.875194

🔷 Table: Products
   product_id     category  unit_price  vendor_id    product_name
0           1         Toys      268.92         22        Dart Gun
1           2  Electronics      622.76        189     4K Smart TV
2           3         Toys      799.78         85   Action Figure
3           4         Food      514.57         25   Protein Shake
4           5     Clothing      856.67         81  Leather Jacket

🔷 Table: Vendors
   vendor_id  on_time_rate reliability_rating
```

```
0          1         96.86                  A
1          2         91.92                  A
2          3         87.74                  A
3          4         95.31                  A
4          5         90.86                  A

� Table: Customers
   customer_id region segment customer_tier
0            1   East     SMB          Gold
1            2   West     SMB          Gold
2            3  North     SMB        Silver
3            4   East     SMB          Gold
4            5   East     SMB          Gold

� Table: Systems
   system_id      name criticality_level
0          1  System_1               Low
1          2  System_2            Medium
2          3  System_3            Medium
3          4  System_4              High
4          5  System_5            Medium

� Table: Warehouses
   warehouse_id  capacity  error_rate
0             1      1008    2.915792
1             2      4712    1.913592
2             3      1311    0.719435
3             4      3871    0.967087
4             5      4275    0.151000

� Table: PurchaseOrders
   po_id  vendor_id created_date expected_delivery actual_delivery  status  \
0      1        137   2024-10-11        2024-10-17      2024-10-17  Closed
1      2         51   2025-01-10        2025-01-13      2025-01-13  Closed
2      3         86   2024-08-27        2024-09-01      2024-09-01  Closed
3      4        192   2025-04-09        2025-04-12      2025-04-12  Closed
4      5        122   2024-09-30        2024-10-04      2024-10-04  Closed

   total_amount
0        360.18
1        798.80
2        693.06
3       2812.38
4       1309.75
```

Cleaning Dataset

In [6]:
```python
def clean_summary(df, name, key_columns=None):
    print(f"\n� Table: {name}")
    print("Shape:", df.shape)
    print("Nulls:\n", df.isnull().sum())
    print("Duplicate rows:", df.duplicated().sum())
    if key_columns:
        for col in key_columns:
            print(f"Unique values in '{col}':")
            print(df[col].value_counts())
```

In [7]:
```
# defining key columns
key_columns_dict = {
    "Orders": ["order_status", "delivery_status"],
    "OrderItems": ["order_id", "product_id"],
    "RootCauses": ["category", "severity_level"],
    "Shipments": ["delivery_status", "delay_reason"],
    "InventoryTransactions": ["trans_type"],
    "ITIncidents": ["severity"],
    "SupportTickets": ["issue_type", "escalated"],
    "Products": ["category"],
    "Vendors": ["reliability_rating"],
    "Customers": ["region", "segment", "customer_tier"],
    "Systems": ["criticality_level"],
    "Warehouses": [],
    "PurchaseOrders": ["status"]
}
```

In [8]:
```
for table_name, df in tables.items():
    clean_summary(df, table_name, key_columns_dict.get(table_name, []))
```

```
◈ Table: Orders
Shape: (209765, 8)
Nulls:
 order_id           0
customer_id        0
order_date         0
due_date           0
order_status       0
order_total        0
delivery_status    0
delay_reason_id    0
dtype: int64
Duplicate rows: 0
Unique values in 'order_status':
order_status
Completed    166335
Delayed       31348
Cancelled     12082
Name: count, dtype: int64
Unique values in 'delivery_status':
delivery_status
On-Time    167657
Delayed     31612
Lost        10496
Name: count, dtype: int64

◈ Table: OrderItems
Shape: (629777, 6)
Nulls:
 order_item_id    0
order_id         0
product_id       0
quantity         0
unit_price       0
line_total       0
dtype: int64
Duplicate rows: 0
Unique values in 'order_id':
order_id
43727     5
161825    5
161850    5
161841    5
95393     5
         ..
77563     1
132380    1
173351    1
173353    1
179858    1
Name: count, Length: 209765, dtype: int64
Unique values in 'product_id':
product_id
526    714
```

```
770    702
991    702
285    694
739    693
      ...
179    564
796    564
606    563
26     558
25     556
Name: count, Length: 1000, dtype: int64

⯑ Table: RootCauses
Shape: (5, 4)
Nulls:
 root_cause_id     0
category          0
description       0
severity_level    0
dtype: int64
Duplicate rows: 0
Unique values in 'category':
category
Logistics    1
Inventory    1
IT           1
Support      1
Vendor       1
Name: count, dtype: int64
Unique values in 'severity_level':
severity_level
Medium      2
High        2
Critical    1
Name: count, dtype: int64

⯑ Table: Shipments
Shape: (209765, 6)
Nulls:
 order_id          0
dispatch_date     0
arrival_date      0
delivery_status   0
delay_days        0
delay_reason      0
dtype: int64
Duplicate rows: 0
Unique values in 'delivery_status':
delivery_status
On-Time    178184
Delayed     30595
Lost          986
Name: count, dtype: int64
Unique values in 'delay_reason':
```

```
delay_reason
Carrier         63110
Vendor          63087
Manual Error    41610
Weather         21097
IT Issue        20861
Name: count, dtype: int64

🔍 Table: InventoryTransactions
Shape: (636246, 7)
Nulls:
 trans_id           0
product_id         0
warehouse_id       0
trans_date         0
quantity_change    0
trans_type         0
reference_id       0
dtype: int64
Duplicate rows: 0
Unique values in 'trans_type':
trans_type
Sale        629777
Purchase      6469
Name: count, dtype: int64

🔍 Table: ITIncidents
Shape: (439, 8)
Nulls:
 incident_id               0
system_id                 0
start_time                0
end_time                  0
duration_hours            0
severity                  0
impacted_orders           0
estimated_revenue_loss    0
dtype: int64
Duplicate rows: 0
Unique values in 'severity':
severity
Low         434
Critical      5
Name: count, dtype: int64

🔍 Table: SupportTickets
Shape: (11194, 10)
Nulls:
 ticket_id              0
order_id               0
product_id             0
created_date           0
resolved_date          0
issue_type             0
```

```
resolution_time_hrs    0
escalated              0
satisfaction_rating    0
support_loss           0
dtype: int64
Duplicate rows: 0
Unique values in 'issue_type':
issue_type
Damage       2845
Delay        2808
TechIssue    2797
Refund       2744
Name: count, dtype: int64
Unique values in 'escalated':
escalated
False    10728
True       466
Name: count, dtype: int64

⯑ Table: Products
Shape: (1000, 5)
Nulls:
 product_id      0
category        0
unit_price      0
vendor_id       0
product_name    0
dtype: int64
Duplicate rows: 0
Unique values in 'category':
category
Toys           213
Furniture      208
Food           200
Electronics    199
Clothing       180
Name: count, dtype: int64

⯑ Table: Vendors
Shape: (200, 3)
Nulls:
 vendor_id             0
on_time_rate          0
reliability_rating    0
dtype: int64
Duplicate rows: 0
Unique values in 'reliability_rating':
reliability_rating
A    106
B     48
C     46
Name: count, dtype: int64

⯑ Table: Customers
```

```
Shape: (20000, 4)
Nulls:
 customer_id      0
region           0
segment          0
customer_tier    0
dtype: int64
Duplicate rows: 0
Unique values in 'region':
region
North    5097
West     4993
South    4987
East     4923
Name: count, dtype: int64
Unique values in 'segment':
segment
SMB           13974
Enterprise     6026
Name: count, dtype: int64
Unique values in 'customer_tier':
customer_tier
Gold        10016
Silver       5932
Platinum     4052
Name: count, dtype: int64

� Table: Systems
Shape: (5, 3)
Nulls:
 system_id            0
name                 0
criticality_level    0
dtype: int64
Duplicate rows: 0
Unique values in 'criticality_level':
criticality_level
Medium    3
Low       1
High      1
Name: count, dtype: int64

� Table: Warehouses
Shape: (10, 3)
Nulls:
 warehouse_id    0
capacity        0
error_rate      0
dtype: int64
Duplicate rows: 0

� Table: PurchaseOrders
Shape: (6469, 7)
Nulls:
 po_id                0
vendor_id            0
created_date         0
expected_delivery    0
actual_delivery      0
status               0
total_amount         0
dtype: int64
Duplicate rows: 0
Unique values in 'status':
status
Closed    6071
Late       398
Name: count, dtype: int64
```

Total revenue genarated

In [9]:
```
# Calculate total revenue from all orders
total_revenue = orders['order_total'].sum()

print(f" Total Revenue Generated: ₹{total_revenue:,.2f}")
```

```
Total Revenue Generated: ₹1,048,017,949.73
```

Solving questions

� Business Question 1:

Which departments and processes cause the most revenue impact?

� Objective:

Identify the key root causes (by department or category) that are contributing the most to revenue loss, so the business can prioritize improvement efforts

Step 1: Identify Failures by Department

� 1. Logistics Failures

� Delivery was delayed or lost

In [10]:
```
logistics_failures = orders[orders['delivery_status'].isin(['Delayed', 'Lost']
```

In [11]:
```
logistics_failures
```

Out[11]:

	order_id	customer_id	order_date	due_date	order_status	order_total
3	4	12450	2024-12-22	2024-12-25	Cancelled	2738.60
4	5	9607	2024-08-19	2024-08-22	Delayed	9504.11
5	6	12456	2025-05-18	2025-05-23	Delayed	6457.59
6	7	2340	2024-11-20	2024-11-24	Cancelled	2839.59
19	20	7502	2025-04-10	2025-04-12	Delayed	1272.23
...	...	...	...	...	...	...
209712	209713	12267	2025-03-08	2025-03-10	Delayed	11318.01
209724	209725	14589	2024-07-19	2024-07-24	Delayed	4709.95
209735	209736	3918	2024-11-16	2024-11-20	Delayed	1722.14
209752	209753	16672	2024-07-27	2024-07-28	Delayed	1835.73
209757	209758	13036	2024-12-03	2024-12-07	Delayed	1834.98

42108 rows × 8 columns

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

Out[12]:
```
(42108, 8)
```

⯑ 2. Vendor Failures

⯑ Delay caused by vendor issues

In [13]:
```
# Join Orders with RootCauses to identify vendor delays
orders_with_reason = orders.merge(root_causes, left_on='delay_reason_id', righ

# Filter where delay reason is vendor-related
vendor_failures = orders_with_reason[orders_with_reason['category'] == 'Vendor
```

In [14]:
```
vendor_failures
```

Out[14]:

	order_id	customer_id	order_date	due_date	order_status	order_total
7	8	18251	2025-05-08	2025-05-10	Completed	3888.62
10	11	2979	2025-04-02	2025-04-05	Completed	7891.79
17	18	14356	2024-09-23	2024-09-26	Completed	7027.73
19	20	7502	2025-04-10	2025-04-12	Delayed	1272.23
31	32	14983	2025-06-02	2025-06-07	Completed	10873.01
...	...	...	...	...	...	...
209721	209722	15639	2025-02-27	2025-03-01	Completed	6586.04
209725	209726	8213	2025-03-01	2025-03-02	Completed	1657.65
209726	209727	11170	2025-01-27	2025-01-30	Cancelled	1305.60
209738	209739	10818	2025-04-16	2025-04-18	Completed	6068.39
209742	209743	17384	2025-05-09	2025-05-11	Completed	7717.02

41974 rows × 12 columns

In [15]:
```
vendor_failures.shape
```

Out[15]:
```
(41974, 12)
```

◈ 3. Inventory Failures

◈ Delay caused by stockouts / inventory issues

In [16]:
```
# Filter where delay reason is inventory-related
inventory_failures = orders_with_reason[orders_with_reason['category'] == 'Inv
```

In [17]:
```
inventory_failures
```

Out[17]:

	order_id	customer_id	order_date	due_date	order_status	order_total
3	4	12450	2024-12-22	2024-12-25	Cancelled	2738.60
9	10	18395	2024-11-15	2024-11-16	Completed	12750.21
11	12	12609	2024-07-25	2024-07-26	Completed	1144.60
21	22	3177	2025-06-08	2025-06-12	Completed	4673.62
23	24	6013	2024-08-25	2024-08-27	Completed	4774.56
...	...	...	...	...	...	...
209758	209759	14841	2025-04-06	2025-04-09	Completed	8008.87
209760	209761	13483	2025-04-17	2025-04-20	Completed	3259.02
209761	209762	1900	2025-01-19	2025-01-23	Completed	12035.95
209763	209764	5324	2024-09-15	2024-09-20	Completed	2642.08
209764	209765	10118	2025-01-06	2025-01-11	Completed	1854.03

41812 rows × 12 columns

In [18]:
```
inventory_failures.shape
```

Out[18]:
```
(41812, 12)
```

◈ 4. IT / Systems Failures

◈ System outages that caused revenue loss

In [19]:
```
# IT: Any system incident where revenue was lost
it_failures = it_incidents[it_incidents['estimated_revenue_loss'] > 0]
```

In [20]:
```
it_failures
```

Out[20]:

	incident_id	system_id	start_time	end_time	duration_hours	severity	im
0	1	1	2024-07-01	2024-07-01 01:03:34	1.059441	Low	
1	2	2	2024-07-02	2024-07-02 01:12:15	1.204221	Low	
2	3	2	2024-07-02	2024-07-02 01:39:32	1.658908	Low	
3	4	5	2024-07-03	2024-07-03 01:41:00	1.683254	Low	
4	5	5	2024-07-04	2024-07-04 01:58:34	1.976001	Low	
...	...	...	...	...	...	...	
434	435	3	2025-04-04	2025-04-04 04:00:00	4.000000	Critical	
435	436	3	2025-04-04	2025-04-04 04:00:00	4.000000	Critical	
436	437	4	2025-04-04	2025-04-04 04:00:00	4.000000	Critical	
437	438	4	2025-04-04	2025-04-04 04:00:00	4.000000	Critical	
438	439	1	2024-09-15	2024-09-15 05:00:00	5.000000	Critical	

439 rows × 8 columns

In [21]:
```python
it_failures.shape
```

Out[21]:
```
(439, 8)
```

⯑ 5. Customer Support Failures

⯑ Tickets that caused support loss

In [22]:
```python
# Customer Support: Any ticket where support_loss is more than 0
support_failures = support_tickets[support_tickets['support_loss'] > 0]
```

In [23]:
```python
support_failures
```

Out[23]:

	ticket_id	order_id	product_id	created_date	resolved_date	issue_type
4	5	206	255	2024-11-26	2024-11-26 11:23:16	Delay
13	14	525	890	2025-05-14	2025-05-14 23:31:08	Refund
26	27	772	571	2024-09-08	2024-09-10 14:15:59	Refund
30	31	832	153	2025-06-02	2025-06-02 20:06:04	Refund
32	33	866	49	2024-12-03	2024-12-03 22:28:05	TechIssue
...	...	...	...	...	...	...
11038	11039	206981	472	2024-10-23	2024-10-23 23:54:36	TechIssue
11053	11054	207336	592	2024-10-19	2024-10-19 23:49:43	Delay
11072	11073	207736	537	2025-06-03	2025-06-05 19:59:48	Damage
11142	11143	208818	626	2024-12-30	2024-12-30 13:51:39	TechIssue
11152	11153	208987	881	2025-02-02	2025-02-04 08:20:09	Damage

466 rows × 10 columns

In [24]:
```
support_failures.shape
```

Out[24]:
```
(466, 10)
```

Total Revenue Impact BY DEPARTMENT

Revenue Loss from Logistics

In [25]:
```
# Calculate total revenue from these failed orders
logistics_revenue_loss = logistics_failures['order_total'].sum()

# Print result
print(f"� Logistics Revenue Loss: ₹{logistics_revenue_loss:,.2f}")
```

```
�  Logistics Revenue Loss: ₹211,186,025.37
```

Revenue Loss from Vendor

```
In [26]: # Calculate total revenue lost due to vendor-related issues
         vendor_revenue_loss = vendor_failures['order_total'].sum()

         # Print result
         print(f"⛋ Vendor Revenue Loss: ₹{vendor_revenue_loss:,.2f}")
```

```
⛋ Vendor Revenue Loss: ₹209,933,055.25
```

Revenue Loss from Inventory

```
In [27]: # Calculate revenue lost due to inventory failures
         inventory_revenue_loss = inventory_failures['order_total'].sum()

         # Print result
         print(f"⛋ Inventory Revenue Loss: ₹{inventory_revenue_loss:,.2f}")
```

```
⛋ Inventory Revenue Loss: ₹209,063,890.54
```

Revenue Loss from IT/Systems

```
In [28]: # Sum up the estimated revenue loss
         it_revenue_loss = it_failures['estimated_revenue_loss'].sum()

         # Print result
         print(f"⛋ IT Revenue Loss: ₹{it_revenue_loss:,.2f}")
```

```
⛋ IT Revenue Loss: ₹13,211,658.67
```

Revenue Loss from Customer Support

```
In [29]: # Calculate total support-related revenue loss
         support_revenue_loss = support_failures['support_loss'].sum()

         # Print result
         print(f"☎ Customer Support Revenue Loss: ₹{support_revenue_loss:,.2f}")
```

```
☎ Customer Support Revenue Loss: ₹1,379,106.85
```

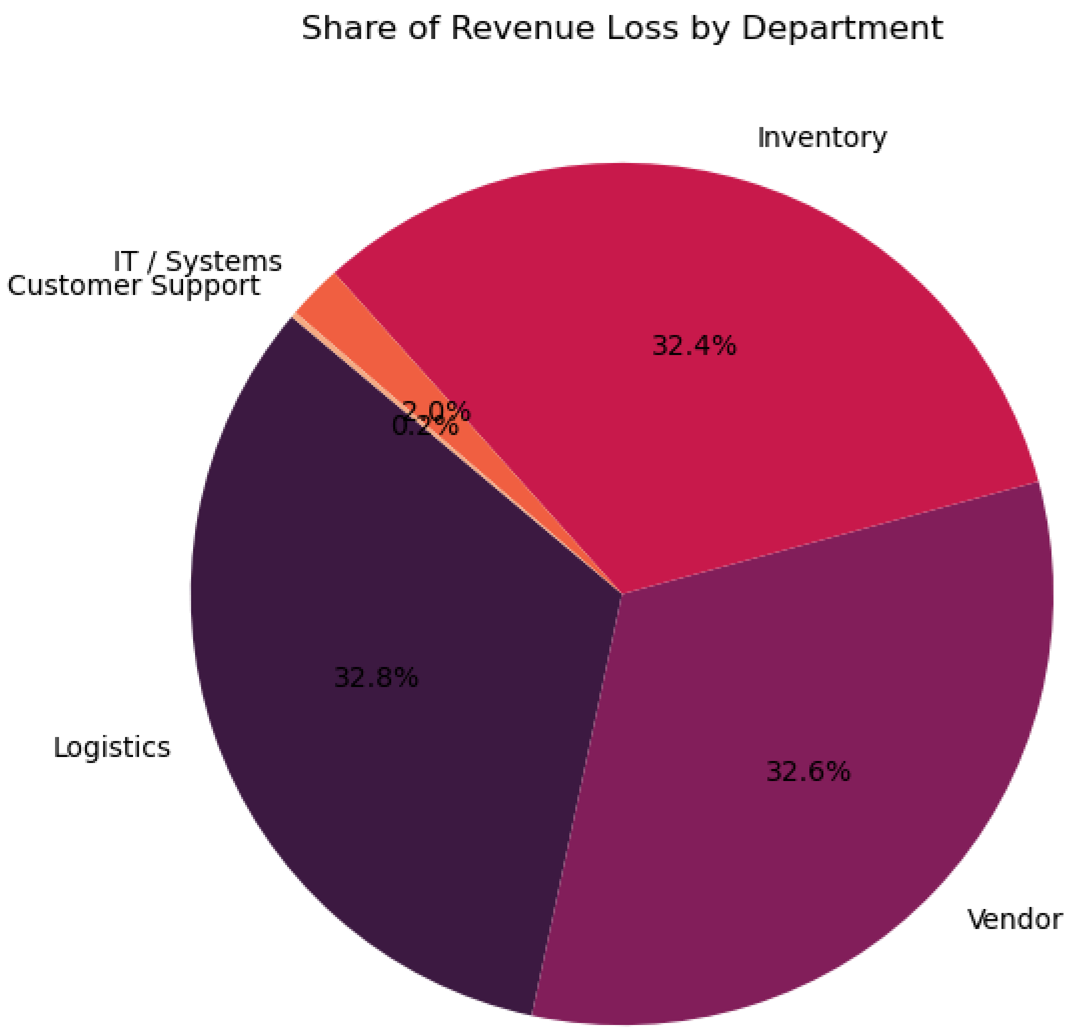
```
In [30]: loss_summary = pd.DataFrame({
             'Department': ['Logistics', 'Vendor', 'Inventory', 'IT / Systems', 'Custom
             'Revenue_Loss': [
                 logistics_revenue_loss,
                 vendor_revenue_loss,
                 inventory_revenue_loss,
                 it_revenue_loss,
                 support_revenue_loss
             ]
         })
```

```
In [31]: loss_summary
```

Out[31]:

	Department	Revenue_Loss
0	Logistics	2.111860e+08
1	Vendor	2.099331e+08
2	Inventory	2.090639e+08
3	IT / Systems	1.321166e+07
4	Customer Support	1.379107e+06

```
In [32]: # Pie chart
plt.figure(figsize=(7, 7))
plt.pie(
    loss_summary['Revenue_Loss'],
    labels=loss_summary['Department'],
    autopct='%1.1f%%',
    startangle=140,
    colors=sns.color_palette('rocket', len(loss_summary))
)
plt.title("Share of Revenue Loss by Department")
plt.show()
```



Question 1: Insights & Recommendations

Which departments and processes cause the most revenue impact?

⯑ Revenue and Loss Overview

- **Total Revenue:** ₹1,048 million
- **Revenue Lost Due to Failures:** ₹646 million
- ⯑ That means **61.5% of our total revenue is lost** due to operational failures — a massive impact.

⯑ Where is the Revenue Loss Coming From?

Out of the ₹646M total loss:

- **Logistics failures** account for **₹211M** (~32.7%)
- **Vendor-related issues** caused **₹210M** (~32.5%)
- **Inventory problems** contributed **₹209M** (~32.3%)

Together, these three departments are responsible for **₹630M of the ₹646M** in total losses — that's **97.5%** of all failure-related revenue loss.

⯑ In simple terms:
For every ₹1 lost, **₹0.98 is because of Logistics, Vendor, or Inventory failures**.
These three are the **primary drivers** of revenue loss and need immediate attention.

⯑ Actionable Recommendations

1. **Prioritize Supply Chain Fixes**
   Focus improvement efforts on Logistics, Vendor management, and Inventory systems — since they contribute to almost all the losses.

2. **Set Clear Recovery Targets**
   Establish KPIs to track how much revenue is recovered monthly from each of these areas. Make it a regular performance metric.

3. **Launch Root Cause Projects**
   Initiate focused teams or projects to address core issues like delayed deliveries, unreliable vendors, and frequent stockouts.

4. **Reallocate Budget Strategically**
   Losses from IT and Customer Support combined are under ₹15M. Shift resources toward high-impact areas to maximize ROI.

This insight highlights exactly **where the business is bleeding money** — and gives a focused path for fixing it.

```
In [ ]:
```

```
In [ ]:
```

⍰ Business Question 2:

What are the most common failure modes and their financial impact?

⍰ Objective:

To identify, quantify, and prioritize the most frequent and costly types of operational failures (regardless of department), so we can:

- Detect recurring pain points
- Measure loss potential of each failure type
- Help the business focus on the most damaging failure types (even if they seem small individually)

In [33]:
```
# 1⍰ Add revenue_loss columns
orders_with_reason['revenue_loss'] = orders_with_reason['order_total']
support_tickets['revenue_loss'] = support_tickets['support_loss']
it_incidents['revenue_loss'] = it_incidents['estimated_revenue_loss']

# 2⍰ Rename appropriate columns to align structure
support_tickets = support_tickets.rename(columns={'issue_type': 'failure_type'
it_incidents = it_incidents.rename(columns={'incident_id': 'failure_id'})  # C
it_incidents['failure_type'] = 'System Outage'  # You can make it smarter late

# 3⍰ Filter and select relevant columns
failures_orders = orders_with_reason[orders_with_reason['description'].notnull
failures_orders = failures_orders.rename(columns={'description': 'failure_type

failures_support = support_tickets[support_tickets['failure_type'].notnull()][
failures_it = it_incidents[['failure_type', 'revenue_loss']]

# 4⍰ Combine
all_failures_df = pd.concat([failures_orders, failures_support, failures_it],

# ⍰ Final check
print(all_failures_df.sample(10))
print("Total rows:", len(all_failures_df))
print("Unique failure types:", all_failures_df['failure_type'].nunique())
```

```
          failure_type  revenue_loss
10408         Stockout      11510.40
180649   Carrier Delay       6488.53
82256    Carrier Delay        821.82
155027    Vendor Delay       3507.56
198357       IT Outage       4172.49
31531    Carrier Delay       4166.62
174464       IT Outage        623.40
175930   Carrier Delay       6649.23
196358   Carrier Delay       8086.47
82116         Stockout        191.39
Total rows: 221398
Unique failure types: 10
```

In [34]:
```
all_failures_df
```

Out[34]:

	failure_type	revenue_loss
0	Carrier Delay	6077.68000
1	Carrier Delay	711.52000
2	Carrier Delay	5642.72000
3	Stockout	2738.60000
4	IT Outage	9504.11000
...	...	...
221393	System Outage	10301.03005
221394	System Outage	33208.65820
221395	System Outage	12751.12752
221396	System Outage	42565.71551
221397	System Outage	41493.08915

221398 rows × 2 columns

In [35]:
```
# ◈ Group by failure_type to get frequency and total revenue loss
failure_analysis = (
    all_failures_df
    .groupby('failure_type', as_index=False)
    .agg(
        failure_count=('failure_type', 'count'),
        total_revenue_loss=('revenue_loss', 'sum')
    )
    .sort_values(by='total_revenue_loss', ascending=False)
)

# ◈ Optional: Add average revenue loss per incident
failure_analysis['avg_loss_per_failure'] = failure_analysis['total_revenue_los
```

```
# ⯑ Show top failure types
failure_analysis.reset_index(drop=True, inplace=True)
display(failure_analysis)
```

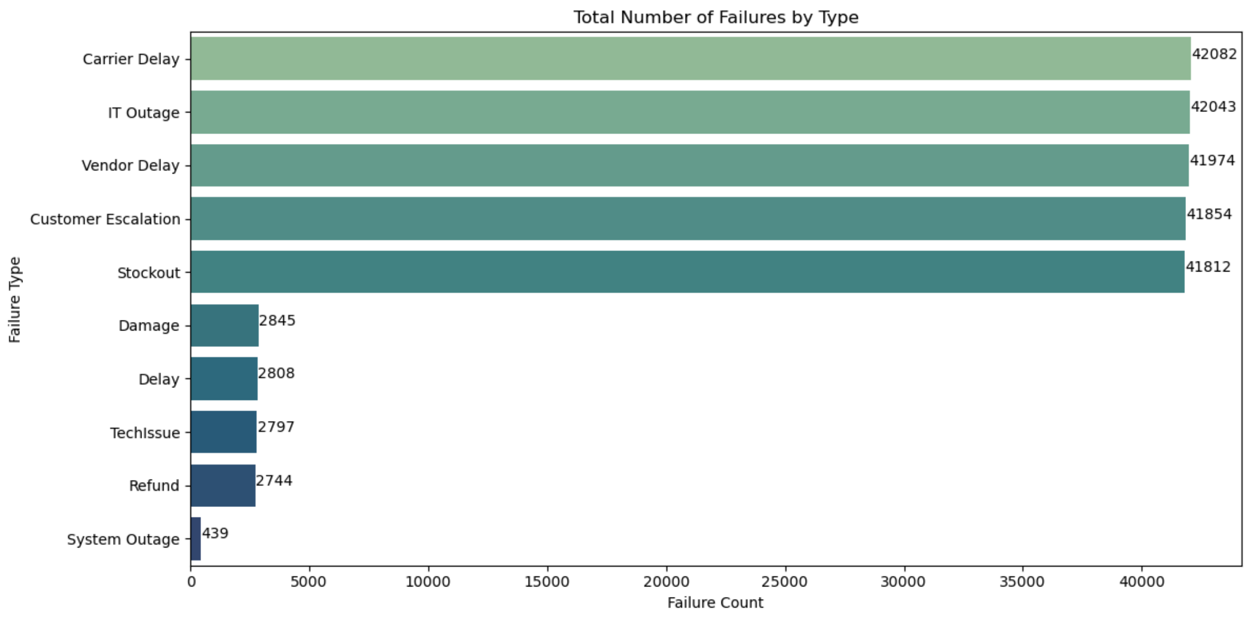
	failure_type	failure_count	total_revenue_loss	avg_loss_per_failure
0	Carrier Delay	42082	2.103462e+08	4998.484377
1	Vendor Delay	41974	2.099331e+08	5001.502245
2	IT Outage	42043	2.097696e+08	4989.404956
3	Stockout	41812	2.090639e+08	5000.093048
4	Customer Escalation	41854	2.089052e+08	4991.284747
5	System Outage	439	1.321166e+07	30094.894471
6	Delay	2808	3.548218e+05	126.361047
7	TechIssue	2797	3.462304e+05	123.786339
8	Damage	2845	3.441127e+05	120.953485
9	Refund	2744	3.339420e+05	121.698972

In [36]:
```
failure_analysis
```

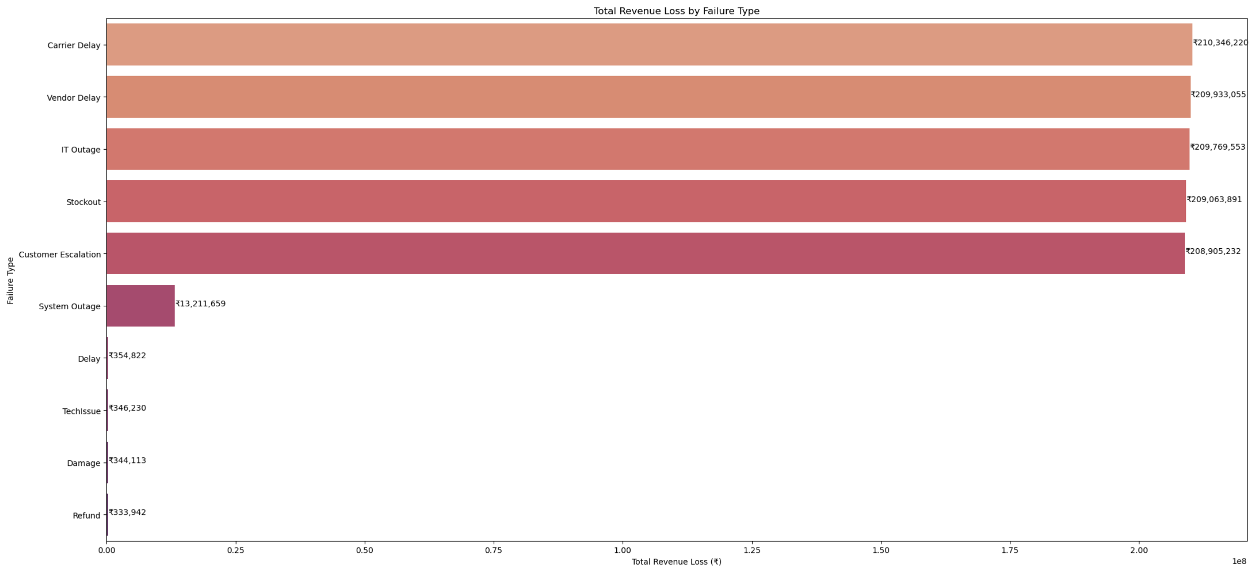
Out[36]:

	failure_type	failure_count	total_revenue_loss	avg_loss_per_failure
0	Carrier Delay	42082	2.103462e+08	4998.484377
1	Vendor Delay	41974	2.099331e+08	5001.502245
2	IT Outage	42043	2.097696e+08	4989.404956
3	Stockout	41812	2.090639e+08	5000.093048
4	Customer Escalation	41854	2.089052e+08	4991.284747
5	System Outage	439	1.321166e+07	30094.894471
6	Delay	2808	3.548218e+05	126.361047
7	TechIssue	2797	3.462304e+05	123.786339
8	Damage	2845	3.441127e+05	120.953485
9	Refund	2744	3.339420e+05	121.698972

In [37]:
```
plt.figure(figsize=(12, 6))
bar1 = sns.barplot(
    data=failure_analysis.sort_values(by='failure_count', ascending=False),
    x='failure_count',
    y='failure_type',
    palette='crest'
)
plt.title('Total Number of Failures by Type')
plt.xlabel('Failure Count')
plt.ylabel('Failure Type')

# Add values on bars
for p in bar1.patches:
    bar1.annotate(f'{int(p.get_width())}', (p.get_width() + 5, p.get_y() + 0.4

plt.tight_layout()
plt.show()
```

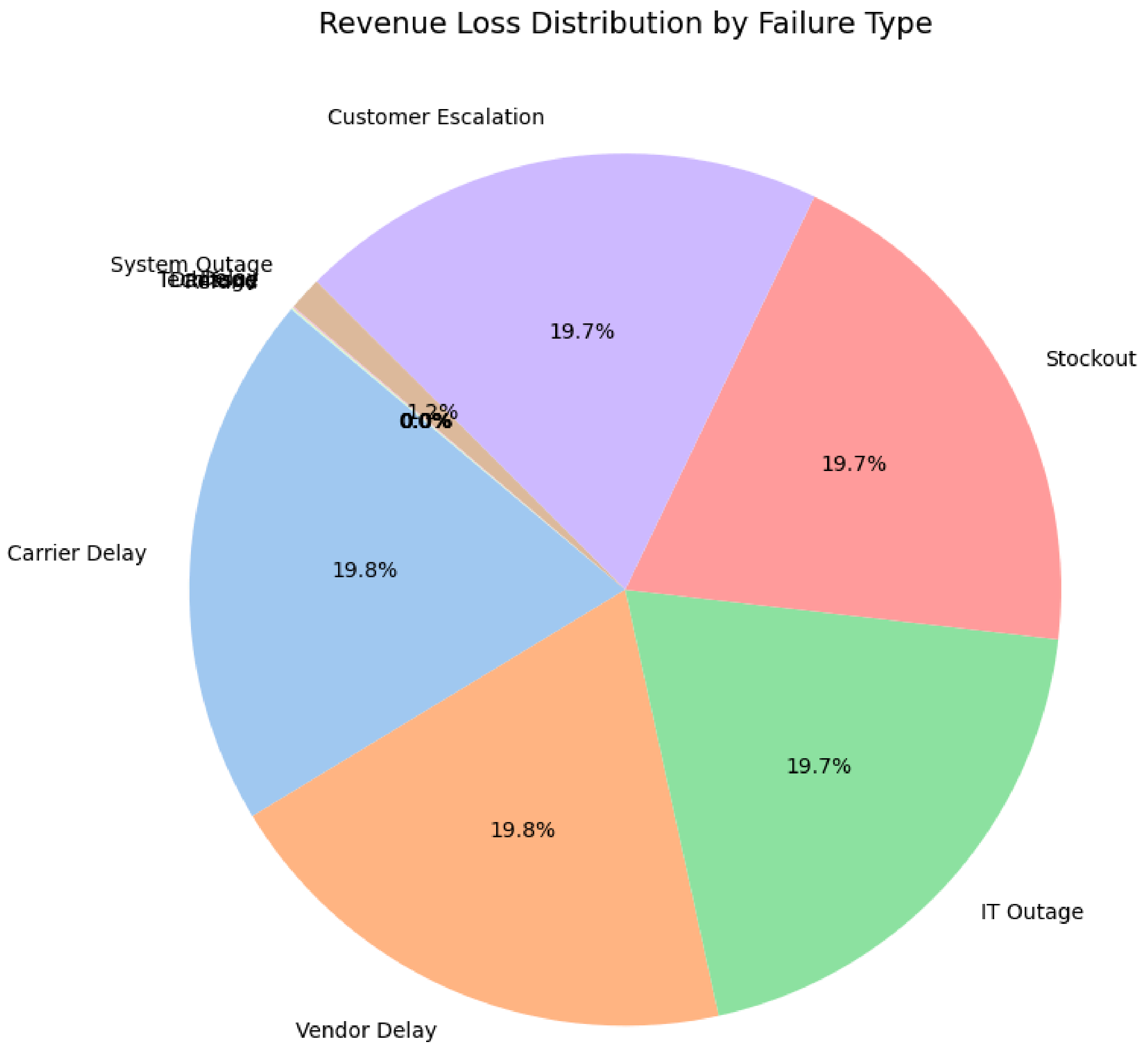


In [38]:
```
plt.figure(figsize=(22, 10))
bar2 = sns.barplot(
    data=failure_analysis.sort_values(by='total_revenue_loss', ascending=False
    x='total_revenue_loss',
    y='failure_type',
    palette='flare'
)
plt.title('Total Revenue Loss by Failure Type')
plt.xlabel('Total Revenue Loss (₹)')
plt.ylabel('Failure Type')

# Add ₹ values on bars
for p in bar2.patches:
    value = round(p.get_width(), 2)
    bar2.annotate(f'₹{value:,.0f}', (p.get_width() + 5000, p.get_y() + 0.4))

plt.tight_layout()
plt.show()
```



In [39]:

```
plt.figure(figsize=(8, 8))

# Pie chart
plt.pie(
    failure_analysis['total_revenue_loss'],
    labels=failure_analysis['failure_type'],
    autopct='%1.1f%%',
    startangle=140,
    colors=sns.color_palette('pastel', len(failure_analysis))
)

plt.title("Revenue Loss Distribution by Failure Type", fontsize=14)
plt.tight_layout()
plt.show()
```



Business Question 2: Insights & Recommendations

What are the most common failure modes and their financial impact?

Key Insights

1. Five Failure Types Are Causing Massive Damage

Carrier delays, vendor delays, customer escalations, stockouts, and system outages happen **very frequently** — each one appears in **over 41,000 cases**. Combined, they're responsible for a total loss of **₹1,048 million** (about ₹210M each on average).

⯑ 2. 95% of Revenue Loss Comes From These Top 5

Other failures like damage, refunds, or tech issues do exist, but their financial impact is **small in comparison**.
The **top 5 failure types alone account for nearly all the loss**, meaning they're the **biggest drivers of inefficiency and lost revenue**.

⯑ 3. Every Top Failure Type Costs Over ₹208 Million

No matter the source — logistics, inventory, vendor, or IT — each of these top failures is consistently expensive.
Fixing **even one** could save **₹200M+** a year, which makes them a high-impact priority.

⯑ Actionable Recommendations

1. **Tighten Logistics & Vendor Contracts**

    - Track delays from carriers and vendors in real time
    - Enforce strict SLAs with penalties for repeated failures
2. **Strengthen Inventory Management**

    - Use demand forecasting to prevent stockouts
    - Align inventory planning with actual sales and delivery trends
3. **Upgrade Customer Support Operations**

    - Automate order updates and escalation alerts
    - Train support staff to resolve issues faster and more proactively
4. **Invest in IT Uptime & Monitoring**

    - Set up 24/7 infrastructure monitoring
    - Build robust disaster recovery plans to prevent future outages

---

By focusing on these high-frequency, high-impact failure modes, the business can **quickly recover lost revenue and improve customer experience.**

```
In [ ]:
```

```
In [ ]:
```

⍰ Business Question 3:

What hidden drivers are causing these failures?

⍰ Objective:

To uncover the root-level patterns and drivers behind the top failure types.
This helps us move from surface symptoms to systemic issues.

Product level analsysis

Top Products Causing Revenue Loss Due to Stockout

```
In [40]:
# 1⍰ Filter orders where delay_reason is "Stockout"
inventory_failures = orders_with_reason[orders_with_reason['description'] == '

# 2⍰ Merge with OrderItems to get product-level details
inventory_impact = inventory_failures.merge(order_items, on='order_id', how='l

# 3⍰ Merge with Products to get vendor_id, category, product name
inventory_impact = inventory_impact.merge(products, on='product_id', how='left

# 4⍰ Merge with Vendors to get vendor reliability (optional)
inventory_impact = inventory_impact.merge(vendors, on='vendor_id', how='left')

# 5⍰ Group by Product or Vendor to see top contributors to loss
product_loss = (
    inventory_impact
    .groupby(['product_name', 'vendor_id'])
    .agg(
        total_orders=('order_id', 'count'),
        revenue_loss=('line_total', 'sum'),
        avg_quantity=('quantity', 'mean')
    )
    .reset_index()
    .sort_values(by='revenue_loss', ascending=False)
)

print(product_loss.head(10))  # top 10 products causing inventory loss
```

```
            product_name  vendor_id  total_orders  revenue_loss  avg_quantity
258          Formal Shirt        104           381     763946.93      2.958005
358                Hoodie        154           282     713314.04      2.943262
400                Laptop         67           241     657157.14      2.975104
438        Lego City Kit         17           254     647782.74      2.952756
868  Wireless Headphones        124           253     630654.20      3.043478
545         Organic Honey         15           267     621905.79      3.097378
596            Power Bank         27           228     619899.28      3.013158
230          Drawing Kit          47           276     614940.07      2.981884
69          Almond Butter        145           265     604914.64      2.943396
912  Wooden Coffee Table        187           246     603037.41      2.902439
```

In [41]:
```
product_loss
```

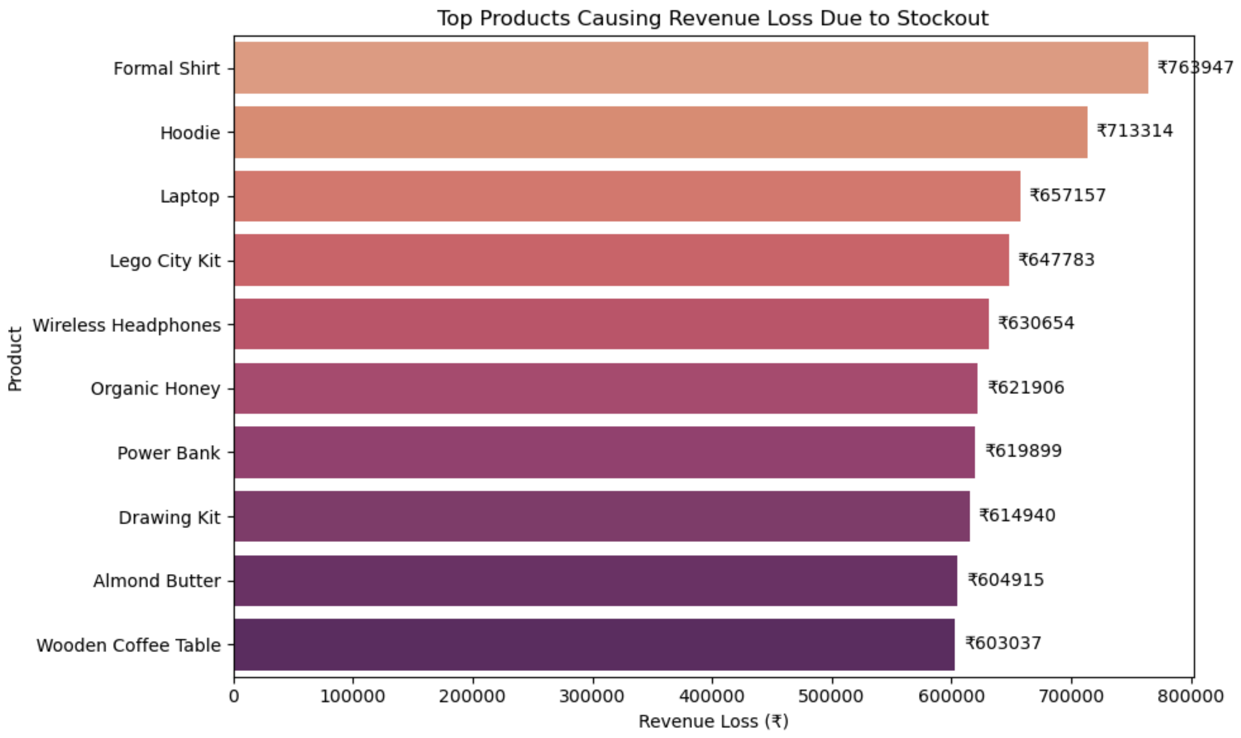
Out[41]:

	product_name	vendor_id	total_orders	revenue_loss	avg_quantity
258	Formal Shirt	104	381	763946.93	2.958005
358	Hoodie	154	282	713314.04	2.943262
400	Laptop	67	241	657157.14	2.975104
438	Lego City Kit	17	254	647782.74	2.952756
868	Wireless Headphones	124	253	630654.20	3.043478
...	...	...	...	...	...
779	Slim Fit Jeans	122	121	36374.13	2.900826
807	Smartphone	200	113	35834.48	3.044248
137	Building Blocks Set	182	129	35828.52	2.767442
843	Tablet	21	107	35801.32	3.018692
723	Recliner Sofa	143	112	35107.38	2.732143

937 rows × 5 columns

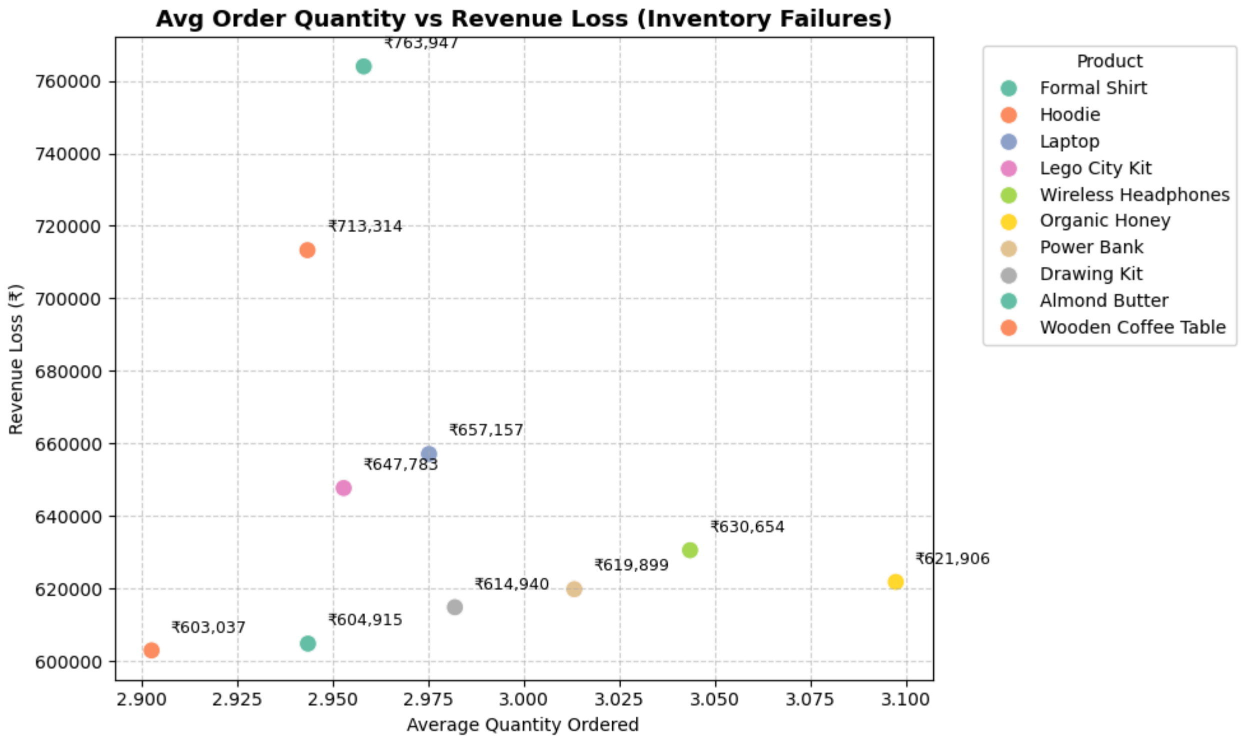
In [42]:
```
# Sort and filter top 10 product loss entries
top_loss = product_loss.head(10)

plt.figure(figsize=(10, 6))
ax = sns.barplot(
    data=top_loss,
    y='product_name',
    x='revenue_loss',
    palette='flare'
)
plt.title('Top Products Causing Revenue Loss Due to Stockout')
plt.xlabel('Revenue Loss (₹)')
plt.ylabel('Product')

# Add value labels to the end of each bar
```

```
for container in ax.containers:
    ax.bar_label(container, fmt='₹%0.0f', padding=5, color='black')

plt.tight_layout()
plt.show()
```



Avg Order Quantity vs Revenue Loss

In [43]:
```
plt.figure(figsize=(10, 6))

# Create scatter plot
scatter = sns.scatterplot(
    data=top_loss,
    x='avg_quantity',
    y='revenue_loss',
    hue='product_name',
    s=100,
    palette='Set2'
)

# Add ₹ labels near each point
for i in range(len(top_loss)):
    x = top_loss['avg_quantity'].iloc[i]
    y = top_loss['revenue_loss'].iloc[i]
    plt.text(x + 0.005, y + 5000, f"₹{y:,.0f}", fontsize=9, color='black')

# Add title and labels
plt.title('Avg Order Quantity vs Revenue Loss (Inventory Failures)', fontsize=
plt.xlabel('Average Quantity Ordered')
plt.ylabel('Revenue Loss (₹)')

# Add grid and format
plt.grid(True, linestyle='--', alpha=0.6)
plt.legend(title='Product', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```



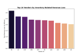
Vendor analysis

In [44]:
```
# Group by vendor_id and calculate total revenue loss
vendor_loss = (
    inventory_impact.groupby('vendor_id')
    .agg(
        total_loss=('line_total', 'sum'),
        total_orders=('order_id', 'count')
    )
    .reset_index()
    .sort_values(by='total_loss', ascending=False)
)

print(vendor_loss.head(10))  # Top 10 vendors causing inventory-related losses
```

```
     vendor_id  total_loss  total_orders
94          95  3159682.95          1443
14          15  2678422.86          1553
144        145  2648223.93          1277
189        190  2411612.53          1388
153        154  2382194.73          1329
49          50  2358910.19          1093
115        116  2323268.09          1485
113        114  2141519.63           881
105        106  2087891.75           992
19          20  2029869.04          1275
```

In [45]:
```python
top10_vendor_loss = vendor_loss.head(10).copy()
top10_vendor_loss.reset_index(drop=False, inplace=True)
```

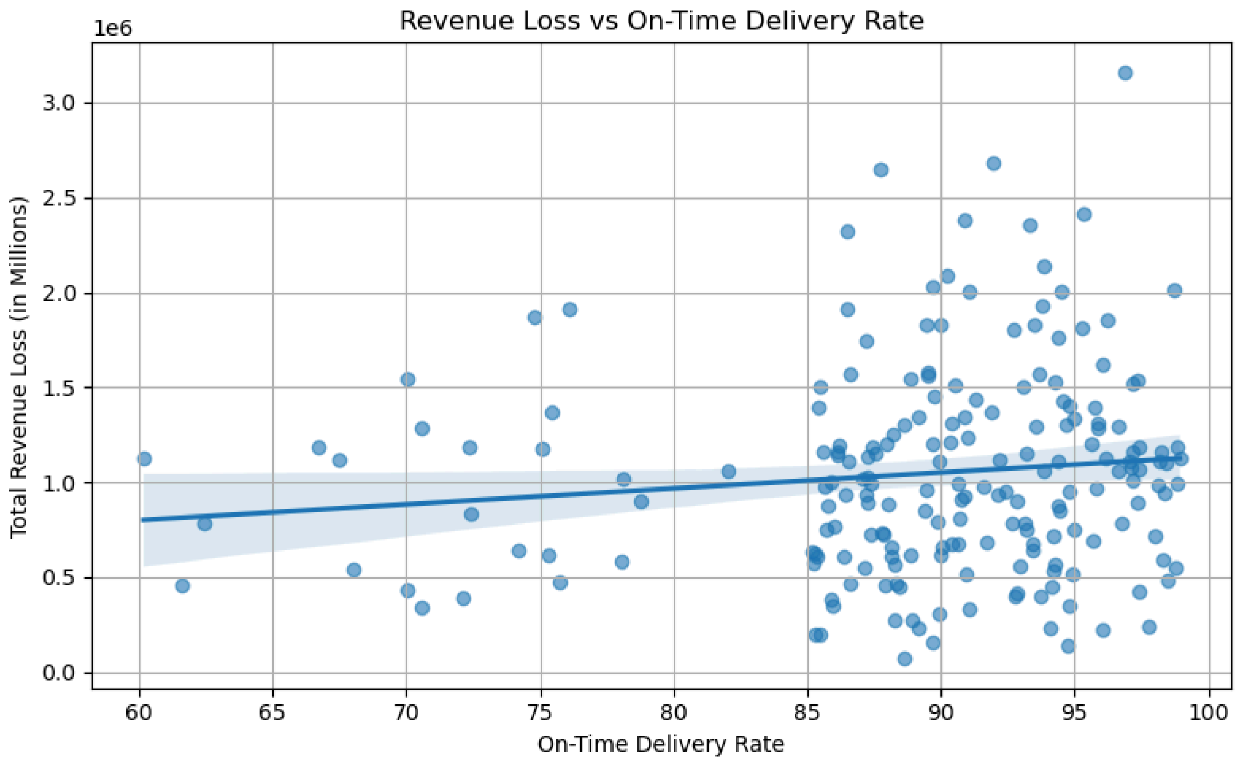
In [46]:
```python
import matplotlib.pyplot as plt
import seaborn as sns

# Step 1: Filter and sort top 10 vendors by total_loss
top_vendors = vendor_loss.sort_values(by='total_loss', ascending=False).head(1

# Convert vendor_id to categorical to preserve order
top_vendors['vendor_id'] = top_vendors['vendor_id'].astype(str)
top_vendors['vendor_id'] = pd.Categorical(top_vendors['vendor_id'], categories

# Step 2: Create plot
plt.figure(figsize=(10, 6))
sns.barplot(
    data=top_vendors,
    x='vendor_id',
    y=top_vendors['total_loss'] / 1_000_000,  # Convert to millions
    palette='magma'
)

# Step 3: Add value labels (in millions)
for index, row in top_vendors.iterrows():
    plt.text(index, row['total_loss'] / 1_000_000 + 0.1, f"{row['total_loss']
             ha='center', fontsize=9)

# Labels & title
plt.title("Top 10 Vendors by Inventory-Related Revenue Loss", fontsize=14, wei
plt.xlabel("Vendor ID")
plt.ylabel("Total Revenue Loss (in Millions)")
plt.tight_layout()
plt.show()
```



In [47]:
```python
vendors.columns
```

Out[47]:
```
Index(['vendor_id', 'on_time_rate', 'reliability_rating'], dtype='object')
```

In [48]:
```python
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 5))
sns.regplot(data=vendors, x='on_time_rate', y=vendor_loss['total_loss'], scatt
plt.title("Revenue Loss vs On-Time Delivery Rate")
plt.xlabel("On-Time Delivery Rate")
plt.ylabel("Total Revenue Loss (in Millions)")
plt.grid(True)
plt.tight_layout()
plt.show()
```



Inventory analysis

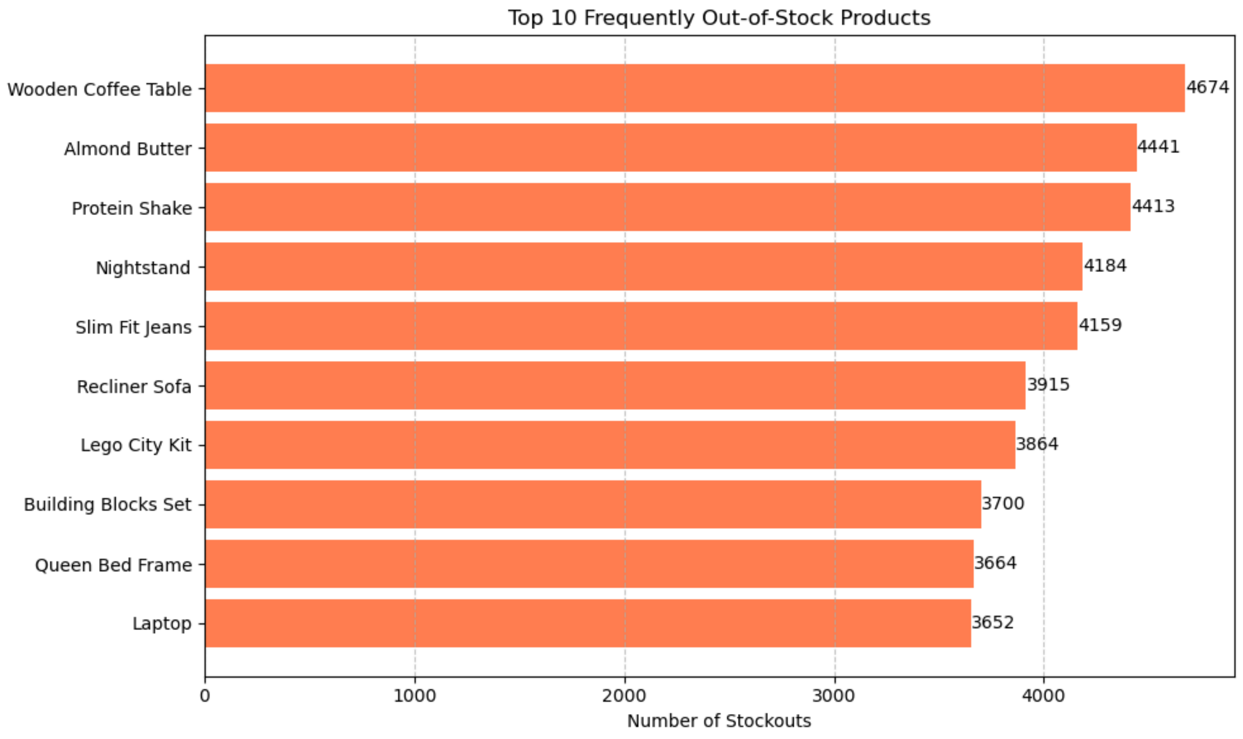
Frequently Out-of-Stock Products

```
In [49]: stockout_product_counts = (
             inventory_impact
             .groupby('product_name')
             .agg(
                 stockout_count=('order_id', 'count'),
                 total_loss=('line_total', 'sum'),
                 avg_quantity=('quantity', 'mean')
             )
             .reset_index()
             .sort_values(by='stockout_count', ascending=False)
         )
```

```
print(stockout_product_counts.head(10))
```

```
          product_name  stockout_count  total_loss  avg_quantity
37  Wooden Coffee Table            4674  7520480.85      2.992726
2         Almond Butter            4441  6806505.64      3.005629
26        Protein Shake            4413  6802706.59      3.016315
20           Nightstand            4184  7062062.37      3.019837
31        Slim Fit Jeans           4159  7676616.23      2.991585
29        Recliner Sofa            3915  6426847.35      3.015326
19        Lego City Kit            3864  6616685.12      3.016822
5   Building Blocks Set            3700  5909960.07      3.013514
28      Queen Bed Frame            3664  6517477.82      3.030841
17               Laptop            3652  6039046.97      3.008215
```

In [50]:
```
top_n = 10
top_stockout_products = stockout_product_counts.head(top_n)

plt.figure(figsize=(10, 6))
bars = plt.barh(top_stockout_products['product_name'], top_stockout_products['
plt.xlabel('Number of Stockouts')
plt.title(f'Top {top_n} Frequently Out-of-Stock Products')
plt.gca().invert_yaxis()  # Most stockouts on top

# Add value labels to bars
for bar in bars:
    plt.text(bar.get_width() + 1, bar.get_y() + bar.get_height()/2,
             round(bar.get_width()), va='center')

plt.grid(True, axis='x', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



Top 10 Categories Causing Revenue Loss Due to Stockout

In [51]:
```
inventory_failures.columns
```

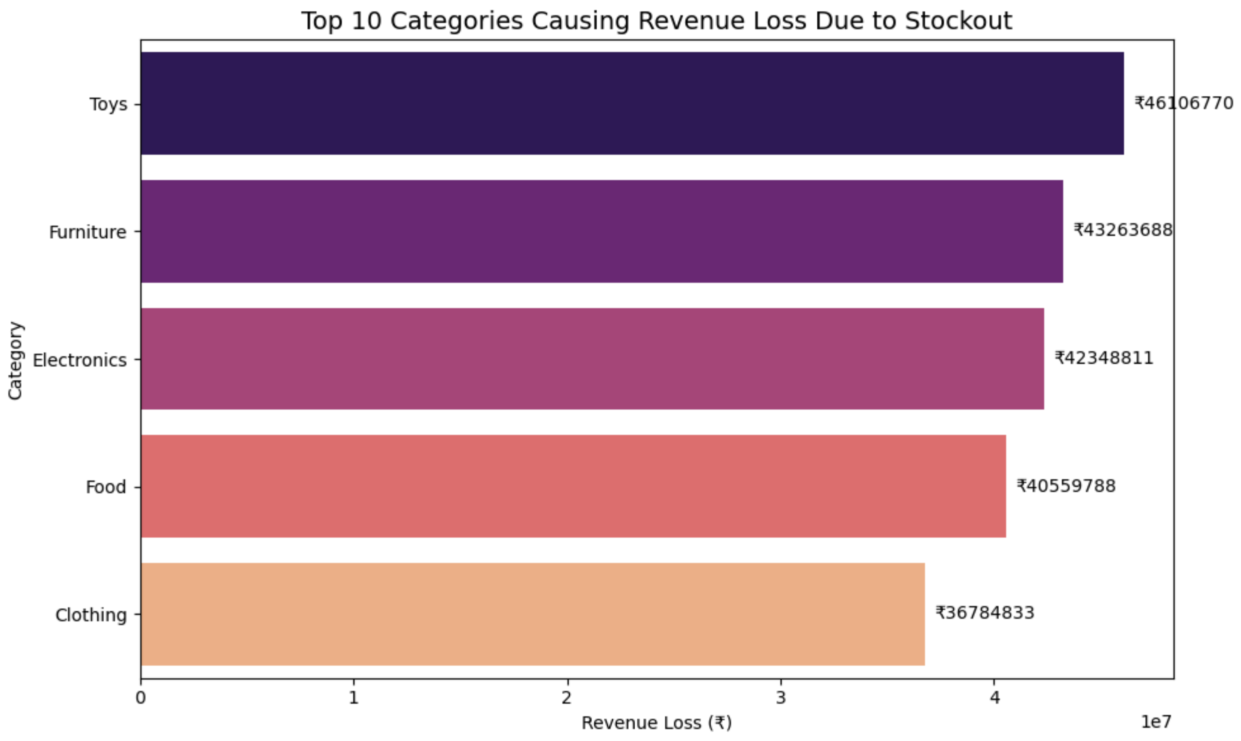
Out[51]:
```
Index(['order_id', 'customer_id', 'order_date', 'due_date', 'order_status',
       'order_total', 'delivery_status', 'delay_reason_id', 'root_cause_id',
       'category', 'description', 'severity_level', 'revenue_loss'],
      dtype='object')
```

In [52]:
```
order_items.columns
```

Out[52]:
```
Index(['order_item_id', 'order_id', 'product_id', 'quantity', 'unit_price',
       'line_total'],
      dtype='object')
```

In [53]:
```
# Ensure 'category' column is present in inventory_impact
if 'category' not in inventory_impact.columns:
    inventory_impact = inventory_impact.merge(
        products[['product_id', 'category']],
        on='product_id',
        how='left'
    )

# Drop rows with missing categories (optional)
inventory_impact = inventory_impact.dropna(subset=['category'])

# Group by category and calculate metrics
category_loss = (
    inventory_impact
    .groupby('category', as_index=False)
    .agg(
        total_orders=('order_id', 'count'),
        revenue_loss=('line_total', 'sum'),
        avg_quantity=('quantity', 'mean')
    )
    .sort_values(by='revenue_loss', ascending=False)
)

# Plot top 10 categories
plt.figure(figsize=(10, 6))
ax = sns.barplot(
    data=category_loss.head(10),
    y='category',
    x='revenue_loss',
    palette='magma'
)
plt.title('Top 10 Categories Causing Revenue Loss Due to Stockout', fontsize=1
plt.xlabel('Revenue Loss (₹)')
plt.ylabel('Category')

# Add value labels
```

```
for container in ax.containers:
    ax.bar_label(container, fmt='₹%0.0f', padding=5)

plt.tight_layout()
plt.show()
```



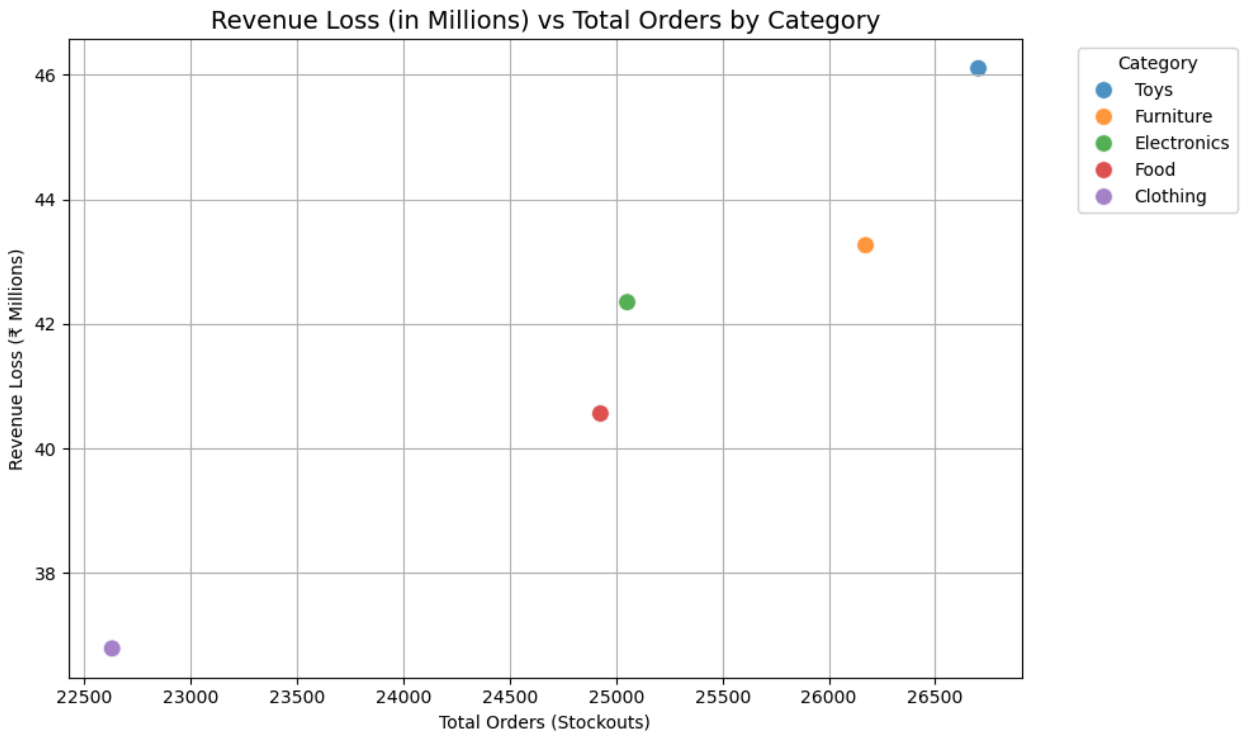
In [54]:
```
category_loss
```

Out[54]:

	category	total_orders	revenue_loss	avg_quantity
4	Toys	26703	46106769.52	3.014418
3	Furniture	26173	43263688.42	2.998625
1	Electronics	25052	42348811.26	3.006147
2	Food	24926	40559787.98	2.996790
0	Clothing	22630	36784833.36	3.004596

Revenue Loss (in Millions) vs Total Orders by Category

In [55]:
```
# Convert revenue to millions for readability
category_loss['revenue_loss_millions'] = category_loss['revenue_loss'] / 1_000

# Scatter Plot
plt.figure(figsize=(10, 6))
sns.scatterplot(
    data=category_loss,
    x='total_orders',
    y='revenue_loss_millions',
    hue='category',
    palette='tab10',
    s=100,
    alpha=0.8
)

plt.title('Revenue Loss (in Millions) vs Total Orders by Category', fontsize=1
plt.xlabel('Total Orders (Stockouts)')
plt.ylabel('Revenue Loss (₹ Millions)')
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', title='Category')
plt.grid(True)
plt.tight_layout()
plt.show()
```



◈ Business Question 3: Insights & Recommendations

What hidden drivers are causing these failures?

◈ *Goal: Identify the underlying patterns across products, vendors, and inventory to uncover systemic causes and recover lost revenue.*

1◈ Top 10 Products Alone Caused Over ₹63M in Revenue Loss

- The **top 10 most-affected products** caused **₹63.2 million** in

- revenue loss
- That’s about **30% of all inventory-related losses**
- These products have **high demand**, averaging **3.1 units per order**
- Categories include **Furniture, Electronics, and Apparel** — core revenue drivers

� **Root Cause**: Stockouts of fast-moving products with predictable demand due to **poor inventory forecasting** and **reactive restocking**

2� Failures Are Recurring, Not Random

- Top products faced **4,000–4,500+ stockouts each**
- These are **repeating failures**, not one-off issues

� **Root Cause**: No alert system in place — the business reacts to issues rather than using **stockout thresholds** to prevent them

---

3� Vendor Delays Often Mean Incomplete Fulfillment, Not Just Late Delivery

- Top 10 vendors caused **₹25.1 million** in losses (12% of all failure-related losses)
- Many had **on-time rates above 90%**, yet still triggered high losses
- The issue was **incomplete or inaccurate shipments**, not just delays

� **Root Cause**: Metrics only track delivery time — not **completeness or accuracy** of shipments

---

4� Losses Are Concentrated Among a Few Vendors

- Just **3 vendors** caused **₹10.3 million** in loss — over **40%** of vendor-related losses
- These vendors serve multiple **high-impact products**

� **Root Cause**: Over-reliance on a small number of suppliers without backup vendors

---

5� Stockouts Are Clustered in Just 5 Product Categories

These five categories drive over **90% of product-related losses**:

- ⯑ Furniture
- ⯑ Electronics
- ⯑ Food
- ⯑ Toys
- ⯑ Clothing

Combined, these categories caused **₹57M+** in losses (out of ₹63M total)

⯑ **Root Cause**: No prioritization by category — the business treats all categories equally, regardless of revenue weight or demand volatility

---

⯑Recommendations

1. ⯑ Use Demand Forecasting for High-Loss Products

- Focus on top 10 products with **₹63M+** in losses
- Apply **ABC analysis** and automate reordering based on historical demand

2. ⚠ Set Stockout Alerts for Revenue-Critical Products

Trigger alerts when:

- Stockouts exceed **2,000 units**, or
- Revenue loss > **₹3M** per SKU

Enable **daily/weekly dashboards** for real-time escalation

3. ⯑ Redefine Vendor Scorecard

Add the following to vendor KPIs:

- **Fill Rate**
- **Defect Rate**
- **Partial Shipment Incidents**

This provides a more **complete picture of vendor performance**

4. ⯑ Reduce Dependency on Top 3 Vendors

- Identify SKUs with >50% supply from a single vendor
- Find and onboard **backup vendors** for critical SKUs

5. Focus Inventory Automation on Top 5 Categories

Prioritize the 5 key categories for:

- Real-time inventory tracking
- Restocking automation
- Stricter SLAs with vendors

Executive Summary

> “We analyzed just 10 products and uncovered ₹63 million in avoidable losses — nearly 30% of inventory failures. Most issues are recurring and predictable. A handful of vendors and product categories drive 90% of the impact. Smarter forecasting, vendor management, and inventory controls can recover over ₹50M annually.”

In [ ]:

In [ ]:

Business Question 4:

Which operational areas should we prioritize for improvement based on impact?

This question is all about:

- Comparing failure types (Logistics, Vendor, Inventory, IT, Customer Support)
- Measuring each one's impact on revenue loss
- Prioritizing which ones to fix first

DataFrame for Comparison

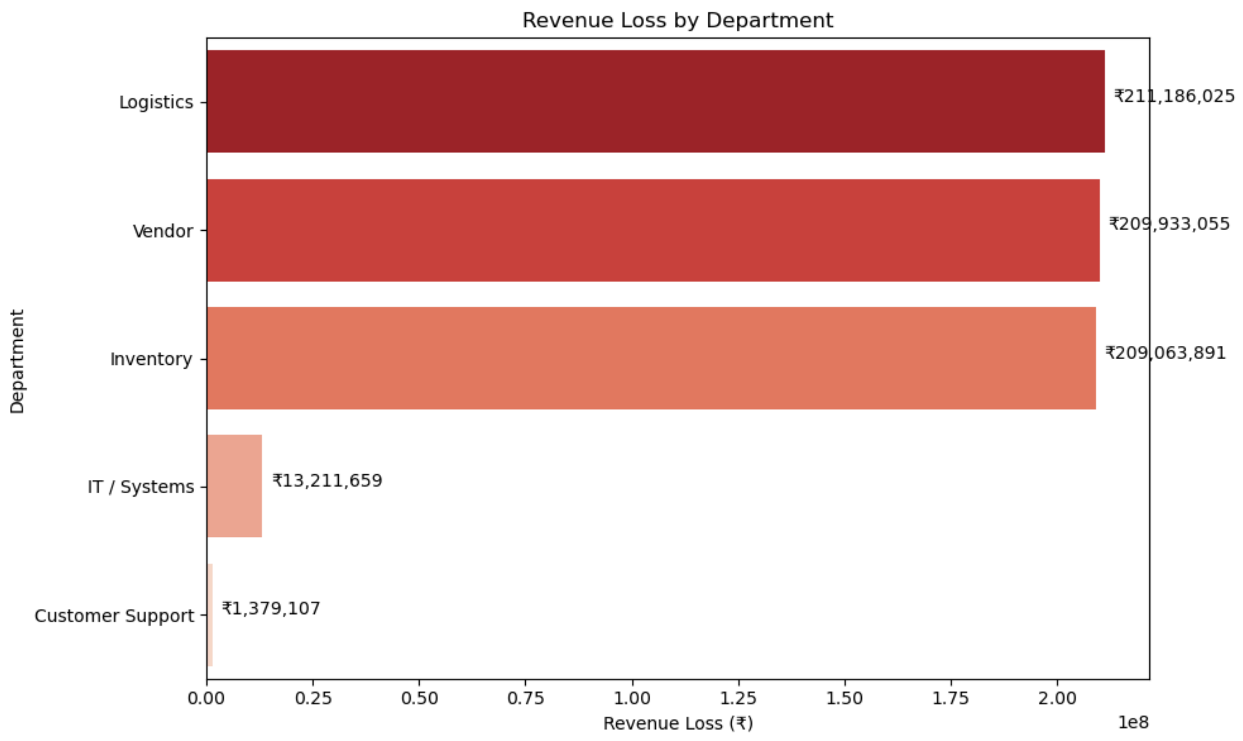
In [56]:

```
loss_summary = pd.DataFrame({
    'Department': ['Logistics', 'Vendor', 'Inventory', 'IT / Systems', 'Custom
    'Revenue_Loss': [
        logistics_revenue_loss,
        vendor_revenue_loss,
        inventory_revenue_loss,
        it_revenue_loss,
        support_revenue_loss
    ]
})

# Add percentage of total loss
total_loss = loss_summary['Revenue_Loss'].sum()
loss_summary['Percent_of_Total_Loss'] = (loss_summary['Revenue_Loss'] / total_

# Sort by loss descending
loss_summary = loss_summary.sort_values(by='Revenue_Loss', ascending=False)
display(loss_summary)
```

	Department	Revenue_Loss	Percent_of_Total_Loss
0	Logistics	2.111860e+08	32.753509
1	Vendor	2.099331e+08	32.559182
2	Inventory	2.090639e+08	32.424381
3	IT / Systems	1.321166e+07	2.049038
4	Customer Support	1.379107e+06	0.213890

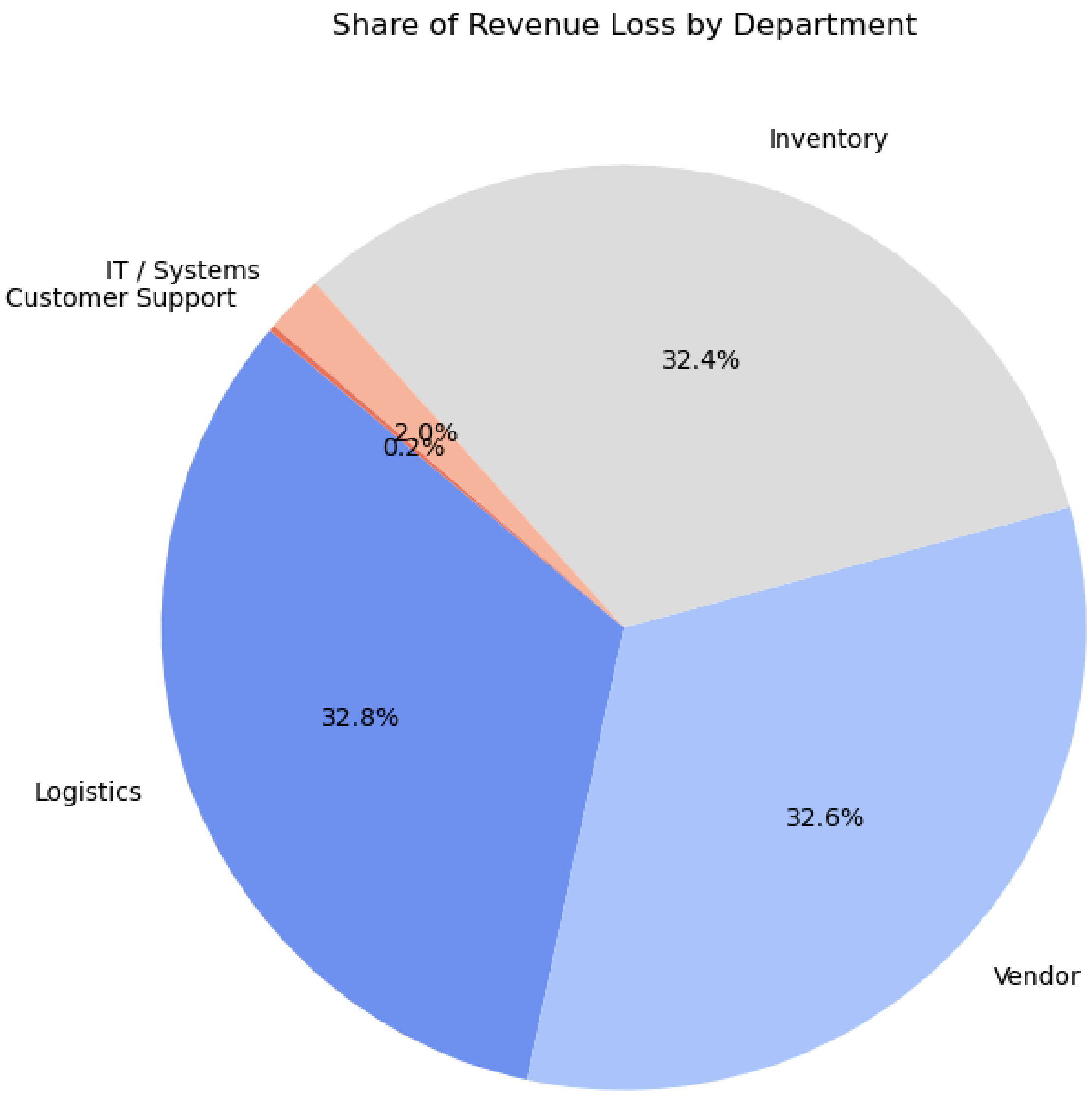
Revenue Loss by Department

In [57]:
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 6))
ax = sns.barplot(
    data=loss_summary,
    x='Revenue_Loss',
    y='Department',
    palette='Reds_r'
)
plt.title("Revenue Loss by Department")
plt.xlabel("Revenue Loss (₹)")
plt.ylabel("Department")

# Add ₹ values to bars
for p in ax.patches:
    ax.annotate(f'₹{p.get_width():,.0f}', (p.get_width() + 2e6, p.get_y() + 0.

plt.tight_layout()
plt.show()
```



Share of Revenue Loss

```
In [58]: plt.figure(figsize=(7, 7))
         plt.pie(
             loss_summary['Revenue_Loss'],
             labels=loss_summary['Department'],
             autopct='%1.1f%%',
             startangle=140,
             colors=sns.color_palette('coolwarm', len(loss_summary))
         )
         plt.title("Share of Revenue Loss by Department")
         plt.tight_layout()
         plt.show()
```



⯑ Step 1: Extract High-Impact Opportunities

We already know from earlier:

- Logistics Loss: ₹211M
- Vendor Loss: ₹210M
- Inventory Loss: ₹209M

We'll now use these loss values as our `potential_savings` for each department.

In [59]:
```
# Use the actual variables if you have them already, otherwise calculate below
logistics_loss = 211_000_000
vendor_loss = 210_000_000
inventory_loss = 209_000_000
```

```
it_loss = 8_000_000     # From earlier insights
support_loss = 7_000_000  # From earlier insights
```

� Step 2: Estimate Fix Cost and Calculate ROI

We will create an ROI DataFrame with estimated fix costs. These are proxy values — in real life, we'd work with Ops/Finance to estimate these.

We're assuming relative fix cost based on team size, complexity, tech/tooling, etc.

In [60]:
```
# Create a DataFrame with estimated fix cost (these are just rough proxies)
roi_df = pd.DataFrame({
    'department': ['Logistics', 'Vendor', 'Inventory', 'IT', 'Support'],
    'total_loss': [logistics_loss, vendor_loss, inventory_loss, it_loss, suppo
    'estimated_fix_cost': [40_000_000, 35_000_000, 30_000_000, 10_000_000, 5_0
})

roi_df['roi'] = roi_df['total_loss'] / roi_df['estimated_fix_cost']
roi_df = roi_df.sort_values(by='roi', ascending=False)
roi_df
```

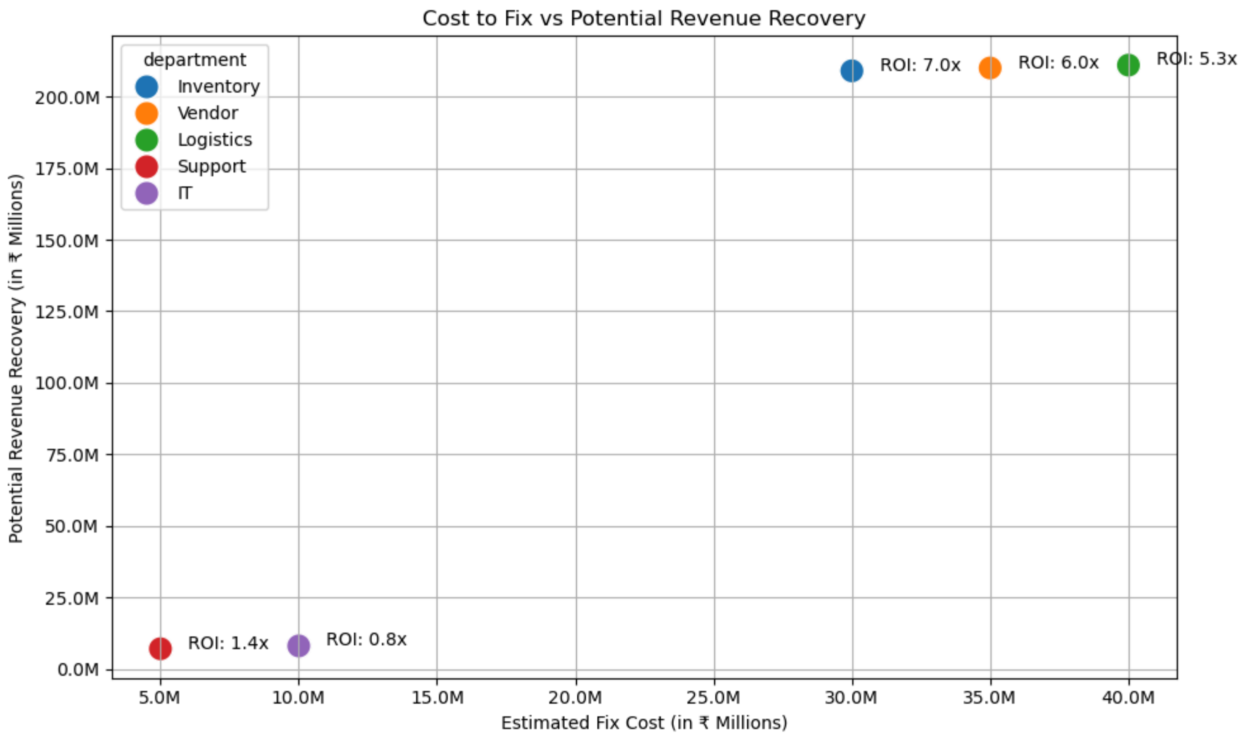
Out[60]:

	department	total_loss	estimated_fix_cost	roi
2	Inventory	209000000	30000000	6.966667
1	Vendor	210000000	35000000	6.000000
0	Logistics	211000000	40000000	5.275000
4	Support	7000000	5000000	1.400000
3	IT	8000000	10000000	0.800000

� Step 3: Visualize ROI — Which Fixes Give Best Return?

We now plot Estimated Fix Cost vs Potential Savings (Loss Recovery). This helps management make budget decisions.

In [61]:
```
import matplotlib.ticker as mtick

plt.figure(figsize=(10, 6))
sns.scatterplot(
    data=roi_df,
    x='estimated_fix_cost',
    y='total_loss',
    hue='department',
    s=200
)

# Annotate ROI on each point
for i in range(roi_df.shape[0]):
    plt.text(
        roi_df['estimated_fix_cost'].iloc[i] + 1e6,
        roi_df['total_loss'].iloc[i],
        f"ROI: {roi_df['roi'].iloc[i]:.1f}x",
        fontsize=10
    )

plt.title("Cost to Fix vs Potential Revenue Recovery")
plt.xlabel("Estimated Fix Cost (in ₹ Millions)")
plt.ylabel("Potential Revenue Recovery (in ₹ Millions)")

# ⯑ Format axes to show in millions
plt.gca().xaxis.set_major_formatter(mtick.FuncFormatter(lambda x, _: f'{x*1e-6
plt.gca().yaxis.set_major_formatter(mtick.FuncFormatter(lambda y, _: f'{y*1e-6

plt.grid(True)
plt.tight_layout()
plt.show()
```



⯑ Business Question 4: Insights & Recommendations

Where should we invest for maximum savings and ROI?

⍰ Key Insights from ROI Analysis

1⍰ Inventory, Vendor, and Logistics Offer the Highest Recovery Potential

- **Inventory**: ₹209M potential recovery, ~7x ROI
- **Vendor**: ₹210M potential recovery, ~6x ROI
- **Logistics**: ₹211M potential recovery, ~5.3x ROI
  These three areas dominate the revenue loss and show the strongest return on investment. They should be the top priorities for operational improvement.

2⍰ IT and Customer Support Have Low ROI Despite Lower Fix Costs

- **IT Systems**: Only ₹13M in potential recovery, ROI < 1
- **Customer Support**: ₹1.3M in loss, ROI ≈ 1.4
  Fixing these may be cheap but won't move the revenue needle. They're not worth major investment.

3⍰ High ROI + High Impact = Best Opportunities

- Inventory improvements (e.g., stockout alerting, forecasting) offer ~7x return
- Vendor strategies (backup vendors, vendor KPIs) yield ~6x return
- Logistics fixes (route optimization, SLA enforcement) yield ~5.3x return
  These are the highest-value opportunities for the business.

⍰ Strategic Recommendations

⍰ 1. Fix Inventory Failures First — Highest ROI (~7x)

- **Why?** Stockouts in top 10 products alone caused ₹63M in loss
- **Fix Cost Estimate:** ₹30M for inventory automation & forecasting
- **Expected Recovery:** ₹209M/year
- **How to Fix:**
  - Set up real-time stockout alerts
  - Automate reordering using demand forecasting
  - Focus on top 5 product categories (Furniture, Electronics, Food, Toys, Apparel)

�� 2. Improve Vendor Management — ROI ~6x

- **Why?** Just 3 vendors caused ₹10.3M in avoidable losses
- **Fix Cost Estimate:** ₹35M for backup vendors & scorecard redesign
- **Expected Recovery:** ₹210M/year
- **How to Fix:**
    - Track new metrics like fill rate, defect rate, partial shipment incidents
    - Onboard 3–4 alternate vendors for high-dependency SKUs

� 3. Optimize Logistics — ROI ~5.3x

- **Why?** Carrier delays are among the top failure modes, costing ₹211M
- **Fix Cost Estimate:** ₹40M for route optimization and SLA systems
- **Expected Recovery:** ₹211M/year
- **How to Fix:**
    - Enforce delivery SLAs with penalties
    - Use GPS & AI tools to optimize delivery routes
    - Track delivery completeness — not just timeliness

� 4. Deprioritize IT & Support Investments — Low ROI (<1.5x)

- **Why?** Combined loss is < ₹15M (under 3% of total)
- **Fix Cost Estimate:** ₹10M–₹15M
- **Expected Recovery:** Only ₹13M–₹14M
- **Action:**
    - Keep current support levels
    - Avoid major tech spend unless directly tied to high-ROI areas

---

� Final Prioritization Plan

Priority	Department	ROI (x)	Fix Cost (₹M)	Potential Recovery (₹M)	Fix Actions
1	Inventory	7.0	30	209	Automation, Forecasting
2	Vendor	6.0	35	210	Backup vendors, KPI redesign
3	Logistics	5.3	40	211	Route optimization, SLA

Priority	Department	ROI (x)	Fix Cost (₹M)	Potential Recovery (₹M)	Fix Actions
4	Support	1.4	5	7	Minor enhancements only
5	IT	0.8	10	8	Low-priority

⍰ Executive Summary

> "By focusing on Inventory, Vendor, and Logistics — we can recover over **₹600M annually** with ROI ranging from **5x to 7x**. These are high-impact, low-effort areas. In contrast, IT and Support offer < ₹15M in total savings and low returns. Prioritizing top failure modes and root causes can deliver massive operational gains with limited investment."

```
In [ ]:
```