In [1]:
```python
import plotly.io as pio
pio.renderers.default = "notebook_connected"
```

🧭 Project Context & Dataset Overview

This notebook presents a diagnostic analysis of a full operational year of retail sales and returns using a synthetic dataset.

The data was generated using a modular scenario engine designed to simulate realistic e-commerce behavior(see; `scenario_01_vp_request.md`), including:

- Customer segmentation (loyalty tiers, sign-up cohorts)
- Multi-channel ordering and returns
- Refund patterns by product, reason, and return type

🔍 **Purpose:** Demonstrate SQL-driven analysis, customer cohort insights, return rate diagnostics, and product loss detection — all within a structured, executive-ready format.

> 📦 **Note:** All data is simulated but grounded in real-world retail logic. This enables storytelling, insight generation, and analysis.

📊 Sales vs. Returns Diagnostic Report and Analysis

Prepared for: VP of Sales
Author: Garrett Schumacher
Date: July 22, 2025
Source: Synthetic ecommerce data (`story_01_upstart_retailer.db`)

📌 Executive Summary: Q3 2025 Sales & Returns Diagnostic

This diagnostic report evaluates the company's post-launch commercial trajectory, spotlighting the forces shaping sales growth, refund pressure, and operational risk from **Q3 2024 through Q2 2025**. It combines performance benchmarking with targeted risk detection, culminating in a product-level quality framework designed to drive immediate action.

> The underlying story is clear: while growth remains strong, unchecked return behavior and operational blind spots are putting margin at risk. This report delivers a roadmap for both **immediate stabilization** and **long-term profitability**.

▼ ✅ Key Achievements & Strengths

- **Strong Initial Sales Momentum:** Sales ramped up rapidly post-launch, peaking during the Q4 2024 holiday period with total Q4 sales reaching **~$12.9M**—our strongest quarter to date.
- **Healthy Conversion Trends in New Cohorts:** While average CLV is still maturing for newer cohorts like Q2–Q3 2025, return rates are declining and customer acquisition is growing—suggesting improved onboarding efficiency and lower refund risk over time.
- **Channel Reliability: Web and Phone orders** drove nearly **$45M in sales**, with refund rates below **21.5%**, reinforcing them as scalable, cost-efficient channels.
- **Payment Mix Is Balanced:** Refund rates across payment methods ranged from **20.1% to 22.1%**, indicating healthy customer trust and no standout refund risks.

▼ ⚠️ Core Challenges & Emerging Risks

- **Eroding Net Revenue:** Total refunds across all channels surpassed **$4.67M**, with return rates **exceeding 20% of revenue** in several months during 2025.
- **Shipping Speed → Return Risk:** Expedited shipping introduces volatility. **Overnight shipping** alone generated **~$323K in refunds** on **$1.5M in sales**—a **21.5% refund rate**.
- **Regional Return Hotspots:**
  - The **Midwest and West** show **both high sales and high return rates**, requiring localized intervention.
- **High-Risk Customer Behavior:**
  - Top returners include **loyal high-CLV customers** and suspected **resellers**. The **military customer segment** posted the **highest return rate overall**, despite relatively low sales volume.
- **Product Quality Blind Spots:** Refund reasons tied to **defects, misdescription, or damage** account for a growing share of losses—often tied to high-volume SKUs.

▼ 🧠 Strategic Solutions & Next Steps

- **Lower Refund Rates Below 20%:** Target reductions through clearer product detail pages, customer education, and SKU-level QA flags.
- **Targeted Customer Risk Mitigation:** Review top 20 returners—particularly **military and reseller segments**—and assess potential loyalty or return policy adjustments.

- **Deinvest in Volatile Channels:** Reevaluate **NewEgg**, which generated **~$450K in refunds** on just **$1.24M in sales**—a margin-eroding **36% refund rate**.
- **Recalibrate Fulfillment Strategy:** Realign expectations on expedited shipping offerings to reduce mismatch and returns.
- **Activate SKU-Level Quality Controls:** Leverage the new **Product Quality Risk Flag System** to prioritize high-impact fixes and empower CX, Ops, and Product teams with visibility.

📚 Table of Contents

📁 Data Overview

▼ 🧼 Cleaned Tables

Table	Description
`orders`	Transaction-level data with dates, totals, channel, and payment method
`order_items`	Product-level detail for each order
`returns`	Return events including refund amounts and reasons
`return_items`	Line-level product returns tied to return IDs
`customers`	Customer demographics, loyalty tier, signup date
`product_catalog`	Product metadata including name, category, and price

▼ 🔍 Engineered Views Used in This Report

View Name	Description
`monthly_signup_channel_sales_returns`	Return and sales trends by signup cohort and channel

View Name	Description
`top_customers_by_returns`	Highest refund customers by volume
`monthly_payment_sales_returns`	MoM performance by payment method
`return_reason_summary`	Aggregated reasons for returns
`region_summary_by_state`	Sales and return breakdown by state/region
`monthly_sales_returns_summary`	High-level monthly trends of orders, returns, refunds
`monthly_clv_sales_returns`	CLV trends across time, including return-adjusted value
`monthly_loyalty_sales_returns`	Return behavior by loyalty tier
`customer_segment_sales_returns`	Return behavior by key customer segments
`return_rate_by_product`	Return count and percentage by product
`shipping_return_impact_summary`	Return likelihood based on shipping type
`category_monthly_sales_returns`	Sales/return trends across product categories
`monthly_channel_sales_returns`	Return trends grouped by order channel

📥 Setup & Load Data

Import cleaned views and prep notebook context.

In [2]:
```
# Import core packages
import os
import pandas as pd
import numpy as np
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
from collections import Counter
```

In [3]:
```
# Dynamically add views and tables

# Set pandas options
pd.set_option('display.max_columns', None)

# --- Path Configuration ---
```

```
# This logic makes the notebook runnable from the repo root (Jupyter)
# or from the notebook's own directory (Voila).
BASE = 'story_05_vp_request/output_data'
if not os.path.isdir(BASE):
    # If the path doesn't exist, assume we're running via Voila,
    # where the working directory is the notebook's directory.
    # Adjust the path to be relative to the story's root.
    BASE = '../output_data'

CLEANED_DIR = os.path.join(BASE, 'cleaned_tables')
VIEWS_DIR = os.path.join(BASE, 'views')

# Helper function to load all CSVs from a folder into a dict
def load_csvs_from_directory(directory_path):
    return {
        os.path.splitext(filename)[0].replace("cleaned_", ""):
pd.read_csv(os.path.join(directory_path, filename))
        for filename in os.listdir(directory_path)
        if filename.endswith('.csv')
    }

# Load cleaned tables and views
cleaned = load_csvs_from_directory(CLEANED_DIR)
views = load_csvs_from_directory(VIEWS_DIR)
```

In [4]:

```
# See available tables
list(cleaned.keys())
```

Out[4]:

```
['returns',
 'customers',
 'orders',
 'return_items',
 'order_items',
 'product_catalog']
```

In [5]:

```
# See available modeled views
list(views.keys())
```

Out[5]:

```
['monthly_signup_channel_sales_returns',
 'top_customers_by_returns',
 'monthly_payment_sales_returns',
 'return_reason_summary',
 'region_summary_by_state',
 'monthly_sales_returns_summary',
 'monthly_clv_sales_returns',
 'monthly_loyalty_sales_returns',
 'customer_segment_sales_returns',
 'return_rate_by_product',
 'shipping_return_impact_summary',
 'category_monthly_sales_returns',
 'monthly_channel_sales_returns']
```

📊 Overall Sales and Refund Health: Initial Performance Snapshot

This section provides a high-level snapshot of our **post-launch commercial performance**, capturing the dual narrative of **strong sales momentum** and rising **refund pressure** since July 2024.

Since launch, **gross sales have totaled $50.1M**, but **$10.6M has been refunded**, resulting in an effective **net revenue erosion of 21.2%**. By placing revenue growth alongside these emerging return trends, we establish a clear baseline for the deeper diagnostics that follow—focused on identifying what's impacting **net revenue, customer satisfaction, and operational efficiency**.

> This performance snapshot sets the stage for the critical questions that drive the rest of the analysis.

▼ 🔍 Observations

- **Strong Initial Momentum**: Since launching in July 2024, our sales experienced a significant ramp-up, peaking robustly during the Q4 2024 holiday season.
- **Predictable Seasonality in Sales**: Following the holiday surge, sales saw a typical post-January decline and have since plateaued in Q1 and Q2 2025, reflecting expected retail seasonality.
- **Concerning Trend in Refund Rate**: While total sales volume has stabilized in 2025, the percentage of revenue being refunded has shown a steady increase throughout Q1 and Q2 2025. This indicates a growing challenge in retaining revenue from completed sales.
- **Concerning Trend in Refund Rate**: While total sales volume has stabilized in 2025, the percentage of revenue being refunded has shown a steady increase throughout Q1 and Q2 2025.
    - In Q4 2024, refunds totaled $2.3M (~18.5% of sales)
    - By Q2 2025, that figure rose to $3.1M (~24.1% of sales)

> *Note: July 2025 data represents an incomplete reporting period and should be interpreted with this in mind.*

▼ 🧠 Key Insights & Business Implications

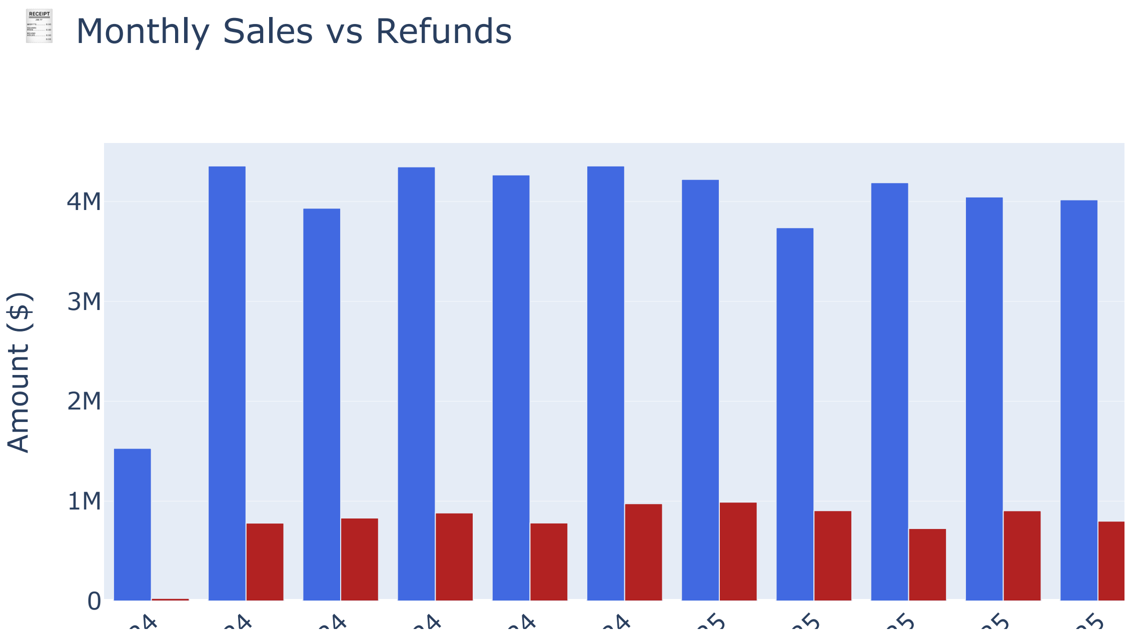
- Despite healthy top-line sales, the rising refund rate is directly eroding profitability and impacting revenue retention, aligning with the VP of Sales' core concerns. This trend suggests that even as customer acquisition remains healthy, the value of each transaction is diminishing due to higher return activity.
- Understanding the drivers behind this increasing refund rate is critical to optimizing our bottom line and improving customer lifetime value. Our subsequent deep dives

will investigate which products, customer segments, channels, and operational practices contribute most to this trend, aiming to identify specific areas for **strategic intervention.**

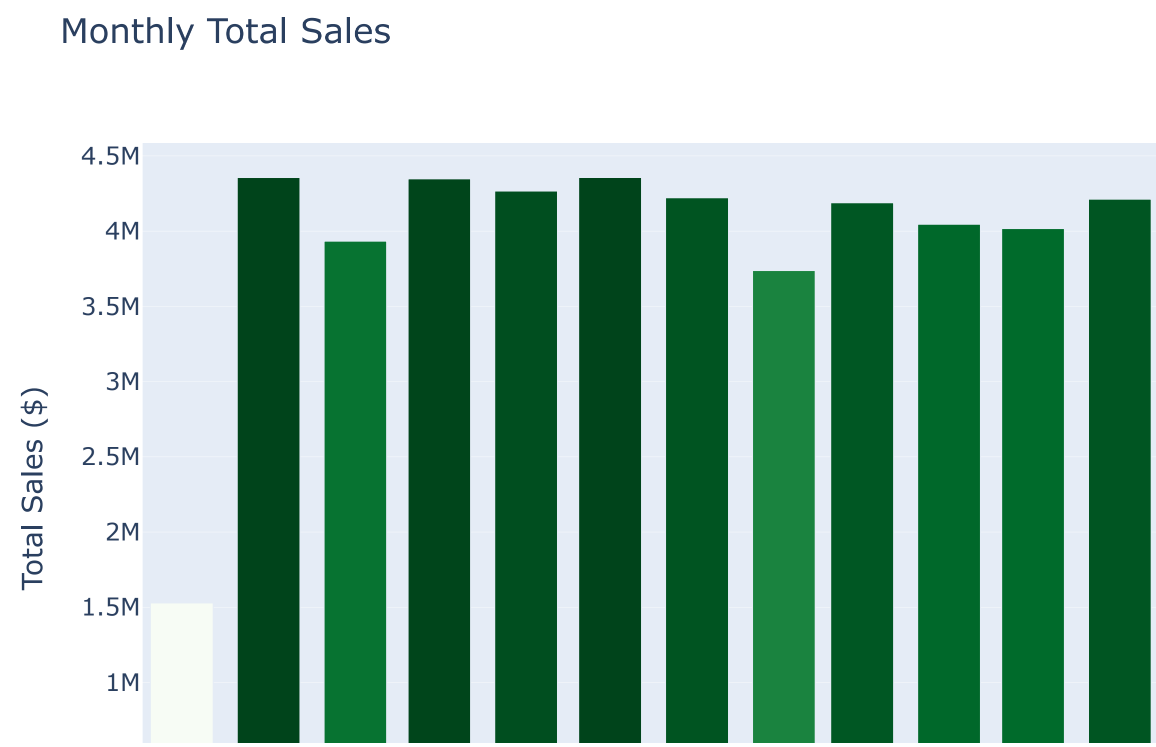
▼ 🛠 **Recommended Actions**

- **Quantify refund pressure** across cohorts, channels, and products to isolate top contributors to margin erosion.
- **Prioritize deeper diagnostics** into the source of increasing return rates—particularly during the post-holiday plateau.
- **Develop hypotheses** for what's driving refund friction (e.g., quality, mismatch, fulfillment delays) to guide the next phase of analysis.
- **Align teams early**: Share refund trends with Product, CX, and Ops to prepare them for upcoming findings and needed collaboration.

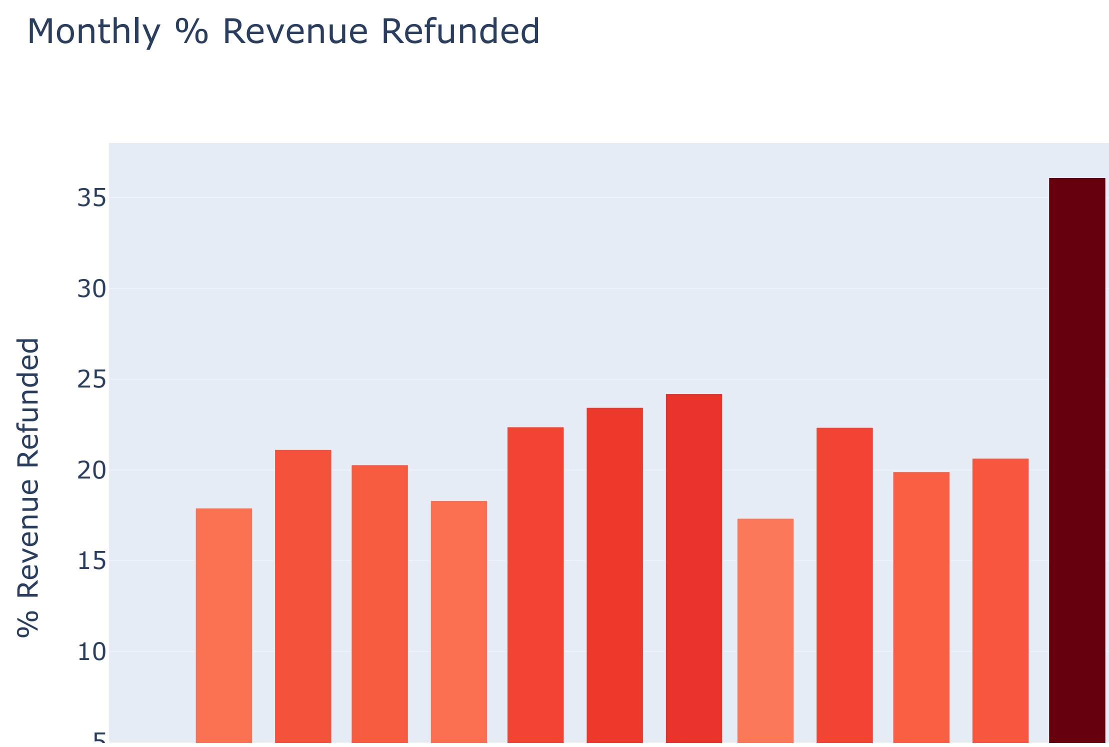
In [6]:
```
# Monthly Sales vs. Refunds (Grouped Bar Chart)

df = views["monthly_sales_returns_summary"].copy()
df["month"] = pd.to_datetime(df["month"])
df["month_label"] = df["month"].dt.strftime("%B %Y")

fig = go.Figure()

fig.add_trace(go.Bar(
    x=df["month_label"],
    y=df["total_sales"],
    name="Total Sales",
    marker_color="royalblue"
))

fig.add_trace(go.Bar(
    x=df["month_label"],
    y=df["total_refunds"],
    name="Total Refunds",
    marker_color="firebrick"
))

fig.update_layout(
    title="🧾 Monthly Sales vs Refunds",
    xaxis_title="Month",
    yaxis_title="Amount ($)",
    barmode="group",
    hovermode="x unified",
    xaxis_tickangle=-45,
    legend=dict(orientation="h", y=-0.2, x=0.5, xanchor="center"),
    height=450
)

fig.show()
```



In [7]:
```
# Build visual of MoM sales

# Load and prep
df = views["monthly_sales_returns_summary"].copy()
df["month"] = pd.to_datetime(df["month"])

# Plot monthly sales
fig = px.bar(
    df,
    x="month",
    y="total_sales",
    color="total_sales",
    color_continuous_scale="Greens",
    title="Monthly Total Sales",
    labels={"total_sales": "Total Sales ($)"}
)

fig.update_layout(hovermode="x unified")
fig.show()
```



In [8]:
```
# Build visual of Total MoM refunded

df = views["monthly_sales_returns_summary"].copy()
df["month"] = pd.to_datetime(df["month"])

fig = px.bar(
    df,
    x="month",
    y="percent_revenue_returned",
    color="percent_revenue_returned",
    color_continuous_scale="Reds",
    title="Monthly % Revenue Refunded",
    labels={"percent_revenue_returned": "% Revenue Refunded"}
)

fig.update_layout(hovermode="x unified")
fig.show()
```



🗺️ Regional Performance: Sales vs. Return Rate Analysis

This section presents a comparative analysis of **regional sales performance and refund behavior**, offering a dual-lens view into both **market strength** (via total sales volume) and **revenue risk exposure** (via return rates).

By examining these two metrics in tandem, we can clearly identify:

- Which regions are **driving top-line growth**
- Where revenue is being **eroded through returns**
- And where strategic **operational or customer experience interventions** could yield the greatest impact on profitability and retention

---

▼ 🔍 **Observations**

- **Significant Revenue Contributors:** The **South and Midwest** are the two largest contributors to total sales, followed by the West. These regions represent our

strongest market penetration.
- **Widespread Return Rate Challenge:** Refund rates across all regions fall within a narrow band of **20–24%**, confirming the VP of Sales' concern—**no region is currently operating below a 20% return rate**.
- **Dual-Impact Zones:** The **Midwest and West** both combine **high total sales** with **elevated return rates**, amplifying their financial impact.
- **Outlier Alert – Military Region:** Despite lower sales volume, the **Military region** exhibits the **highest refund rate**, flagging it as a critical segment for further review.

▼ 🧠 Key Insights & Business Implications

- **Systemic Profitability Risk:** The fact that all regions fall above the 20% refund rate mark suggests a **company-wide retention challenge**, not just isolated inefficiencies.
- **Midwest and West = Leverage Zones:** Small improvements in return rates in these high-volume regions could translate into **millions in preserved revenue**.
- **"Returns Know No Borders":** Since this trend spans all regions, the underlying causes are likely **non-geographic**—potentially related to product mix, fulfillment experience, or customer expectations.
- **Territories as Hidden Margin Leaks:** Low-volume U.S. territories (e.g., Micronesia, Palau) show high refund rates, indicating potential unprofitable fulfillment zones.

▼ 🛠️ Recommended Strategic Actions

- **Launch a Cross-Regional Return Rate Reduction Initiative**
  Aim to bring all regions **below 20%** by targeting shared return drivers like product fit, shipping expectations, and communication clarity.

- **Conduct Region-Specific Deep Dives**
  Especially in the **Midwest and West**, investigate product-level performance, return reasons, and any fulfillment anomalies (e.g., delivery delays or damage rates).

- **Prioritize a Military Segment Review**
  With the **highest refund rate and relatively low revenue contribution**, the **Military customer segment** may require tailored outreach, return education, or a fulfillment protocol review.

- **Assess Territory-Level Viability** Analyze sales and unseen costs of servicing U.S. territories. If margins are negative, consider restricting fulfillment, adding special delivery fees, or adjusting return policies.

In [9]:
```
# Catagorize Unknown Region by Code

raw_region_df = views["region_summary_by_state"]
```

```
unknowns = raw_region_df[raw_region_df["region"].str.lower() == "unknown"]
unknowns
```

Out[9]:

	region	state_code	order_month	total_orders	total_sales	total_returns	tota
598	Unknown	_a	NaN	1	0.00	0	
599	Unknown	fm	2024-07	6	23245.47	3	
600	Unknown	fm	2024-08	10	38923.01	3	
601	Unknown	fm	2024-09	10	32672.72	3	
602	Unknown	fm	2024-10	11	64688.24	4	
603	Unknown	fm	2024-11	11	55842.13	3	
604	Unknown	fm	2024-12	12	69668.19	2	
605	Unknown	fm	2025-01	6	31708.38	1	
606	Unknown	fm	2025-02	11	42821.84	1	
607	Unknown	fm	2025-03	10	51007.58	3	
608	Unknown	fm	2025-04	8	36091.94	3	
609	Unknown	fm	2025-05	14	40610.38	1	
610	Unknown	fm	2025-06	10	38075.27	2	
611	Unknown	fm	2025-07	5	25930.39	2	
612	Unknown	mh	2024-07	5	21009.59	2	
613	Unknown	mh	2024-08	20	105970.73	9	
614	Unknown	mh	2024-09	12	68076.97	3	
615	Unknown	mh	2024-10	19	85992.51	6	
616	Unknown	mh	2024-11	20	97181.22	5	
617	Unknown	mh	2024-12	23	85770.07	7	
618	Unknown	mh	2025-01	16	61111.36	5	
619	Unknown	mh	2025-02	14	53441.38	5	
620	Unknown	mh	2025-03	17	77491.05	6	
621	Unknown	mh	2025-04	15	55310.71	2	
622	Unknown	mh	2025-05	20	84579.56	8	
623	Unknown	mh	2025-06	14	64608.99	3	
624	Unknown	mh	2025-07	8	33000.44	4	
625	Unknown	pw	2024-07	7	37090.83	2	
626	Unknown	pw	2024-08	21	98575.06	6	
627	Unknown	pw	2024-09	19	67206.41	8	
628	Unknown	pw	2024-10	16	74622.84	4	
629	Unknown	pw	2024-11	21	89165.59	4	

	region	state_code	order_month	total_orders	total_sales	total_returns	tota
630	Unknown	pw	2024-12	19	77254.06	4	
631	Unknown	pw	2025-01	19	95637.61	7	
632	Unknown	pw	2025-02	19	86149.69	9	
633	Unknown	pw	2025-03	21	79741.49	7	
634	Unknown	pw	2025-04	9	38194.35	2	
635	Unknown	pw	2025-05	19	77705.94	7	
636	Unknown	pw	2025-06	21	79692.01	7	
637	Unknown	pw	2025-07	8	37667.11	1	

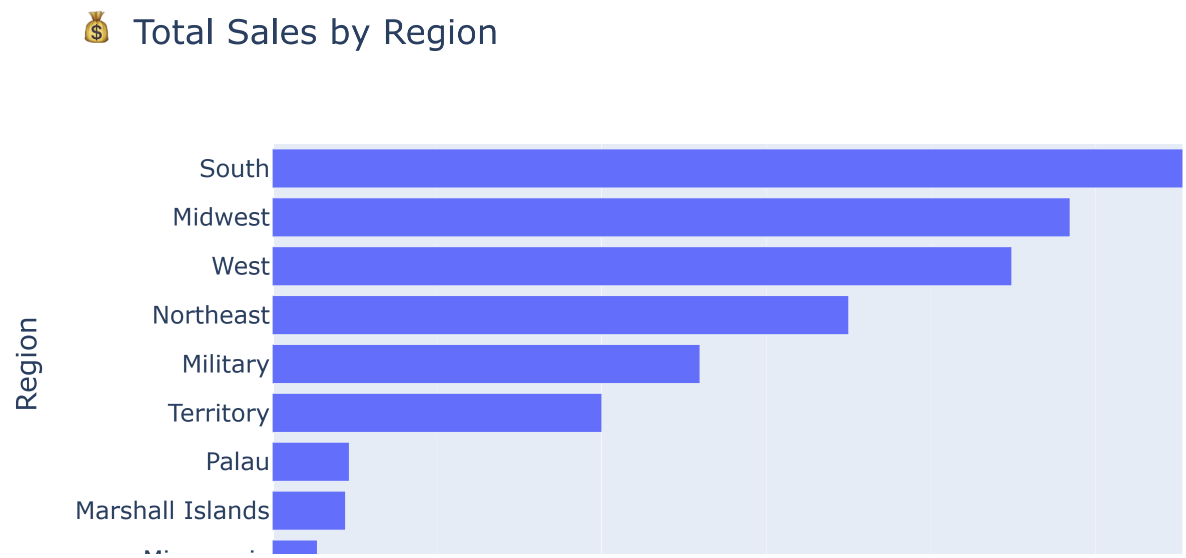
In [10]:

```
# Standarize Outlier Terrirories

territory_map = {
    "fm": "Micronesia",
    "mh": "Marshall Islands",
    "pw": "Palau",
    "_a": "Unassigned"
}

# Apply mapping to create a better label
raw_region_df["region"] = raw_region_df.apply(
    lambda row: territory_map.get(row["state_code"].lower(),
row["region"]),
    axis=1
)
```

In [11]:

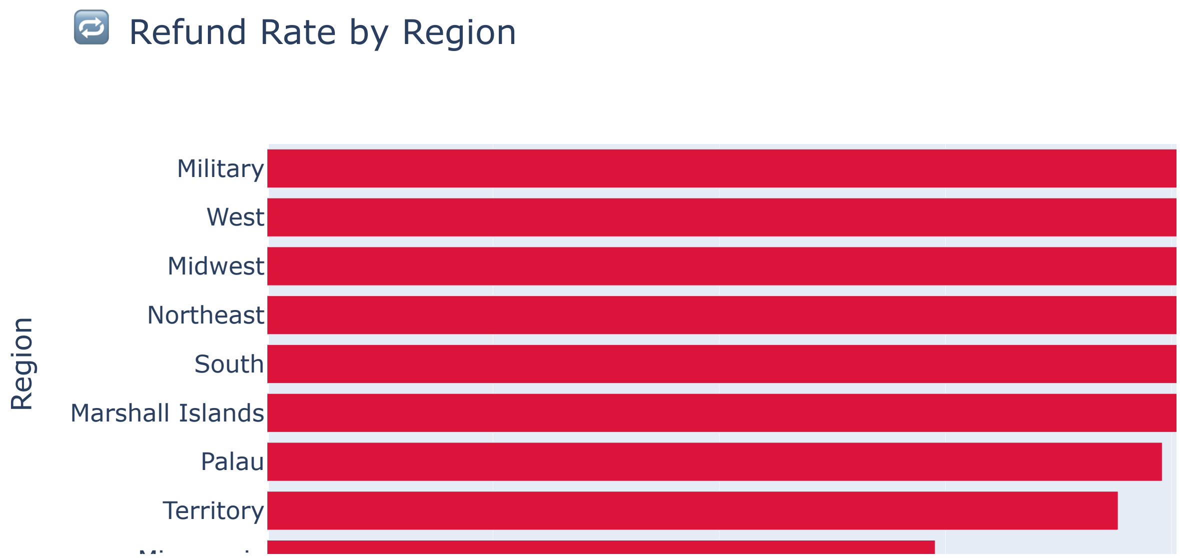
```
# Total sales by region

region_df = views["region_summary_by_state"].copy()
region_df["order_month"] = pd.to_datetime(region_df["order_month"],
errors="coerce")
region_df = region_df.dropna(subset=["order_month", "region",
"total_sales"]).copy()

region_sales = (
    region_df.groupby("region", as_index=False)["total_sales"]
    .sum()
    .sort_values("total_sales", ascending=True)
)

fig_sales = px.bar(
    region_sales,
    x="total_sales",
    y="region",
    orientation="h",
    title="💰 Total Sales by Region",
    labels={"total_sales": "Total Sales ($)", "region": "Region"}
)
```

```
fig_sales.update_layout(xaxis_tickformat="$,.0f", height=400)
fig_sales.show()
```



In [12]:
```
# Refund rate by region

region_returns = (
    region_df.groupby("region", as_index=False)
    .agg(
        total_sales=("total_sales", "sum"),
        total_refunds=("total_refunds", "sum")
    )
)

region_returns["return_rate"] = region_returns["total_refunds"] /
region_returns["total_sales"]
region_returns = region_returns.sort_values("return_rate", ascending=True)

# Plot
import plotly.express as px

fig_returns = px.bar(
    region_returns,
    x="return_rate",
    y="region",
    orientation="h",
    title="🔄 Refund Rate by Region",
    labels={"return_rate": "Refund Rate", "region": "Region"},
    color_discrete_sequence=["crimson"]
)
```

```
fig_returns.update_layout(xaxis_tickformat=".1%", height=400)
fig_returns.show()
```



🚚 Shipping Speed Impact: Evaluating Return Likelihood

Expedited shipping is typically framed as a value-added service—designed to meet urgency and enhance satisfaction.
But this analysis suggests a hidden tradeoff: **faster fulfillment may correlate with higher return rates**.

While expedited shipping aims to enhance perceived service value, these options appear to introduce **greater volatility**, potentially undermining long-term **revenue retention** and **customer trust**.

▼ 🔍 **Observations**

- **Standard Dominance:** The majority of orders are fulfilled via Standard shipping, reaffirming its role as the default method for most customers.
- **Higher Return Rates for Expedited Shipping:** Our data reveals a clear pattern: return rates increase with shipping speed.
  - **Standard shipping**: $32.8 Minsales,$6.84M refunded (**20.87%** of revenue)
  - **Two-Day shipping**: $9.9 Minsales,$2.16M refunded (**21.77%** of revenue)
  - **Overnight shipping**: $7.5 Minsales,$1.61M refunded (**21.50%** of revenue)

- **Volatility in Expedited Returns:** Two-Day and Overnight shipping show greater month-over-month variability in return rates—particularly sharp spikes in Overnight returns during Q1–Q2 2025.
- **Geographic Concentration:** *The Overnight Shipping by State* map indicates that a significant portion of Overnight orders originate from states within well-developed interstate corridors and less rural areas. This suggests that high demand for our fastest shipping isn't necessarily due to remote locations, but rather from customers in logistically "easy" shipping zones who likely have heightened expectations for rapid delivery, possibly influenced by competitor offerings.

▼ 📊 Refund Impact by Shipping Speed

Shipping Speed	Total Sales	Refunds Issued	% Revenue Refunded	Order Count
Standard	$32,792,079.09	$6,844,296.08	**20.87%**	7,775
Two-Day	$9,922,715.29	$2,159,967.54	**21.77%**	2,376
Overnight	$7,464,937.81	$1,605,040.99	**21.50%**	1,794

▼ 🧠 Key Insights & Business Implications

- **Increased Return Likelihood with Expedited Shipping:**
  - Expedited orders are more likely to be returned. This could stem from **higher expectations**, **impulse-driven purchases**, or **need-it-now urgency**.
  - Customers may be **returning expedited orders if competitors offer faster delivery**, even if we meet our promised timeline. This introduces a **perceived slowness gap** that erodes satisfaction.
- **Compounded Profitability Risks:**
  - Expedited shipping incurs higher fulfillment costs.
  - When paired with increased return rates, it creates **a double drag on margins**—higher costs, lower retained revenue.
- **Operational Considerations:**
  - The volatility in returns suggests **potential fulfillment flaws** in our expedited pipeline—rushed packaging, poor handling, or lack of pre-purchase consideration time may drive dissatisfaction.

▼ 🛠 Recommended Strategic Actions

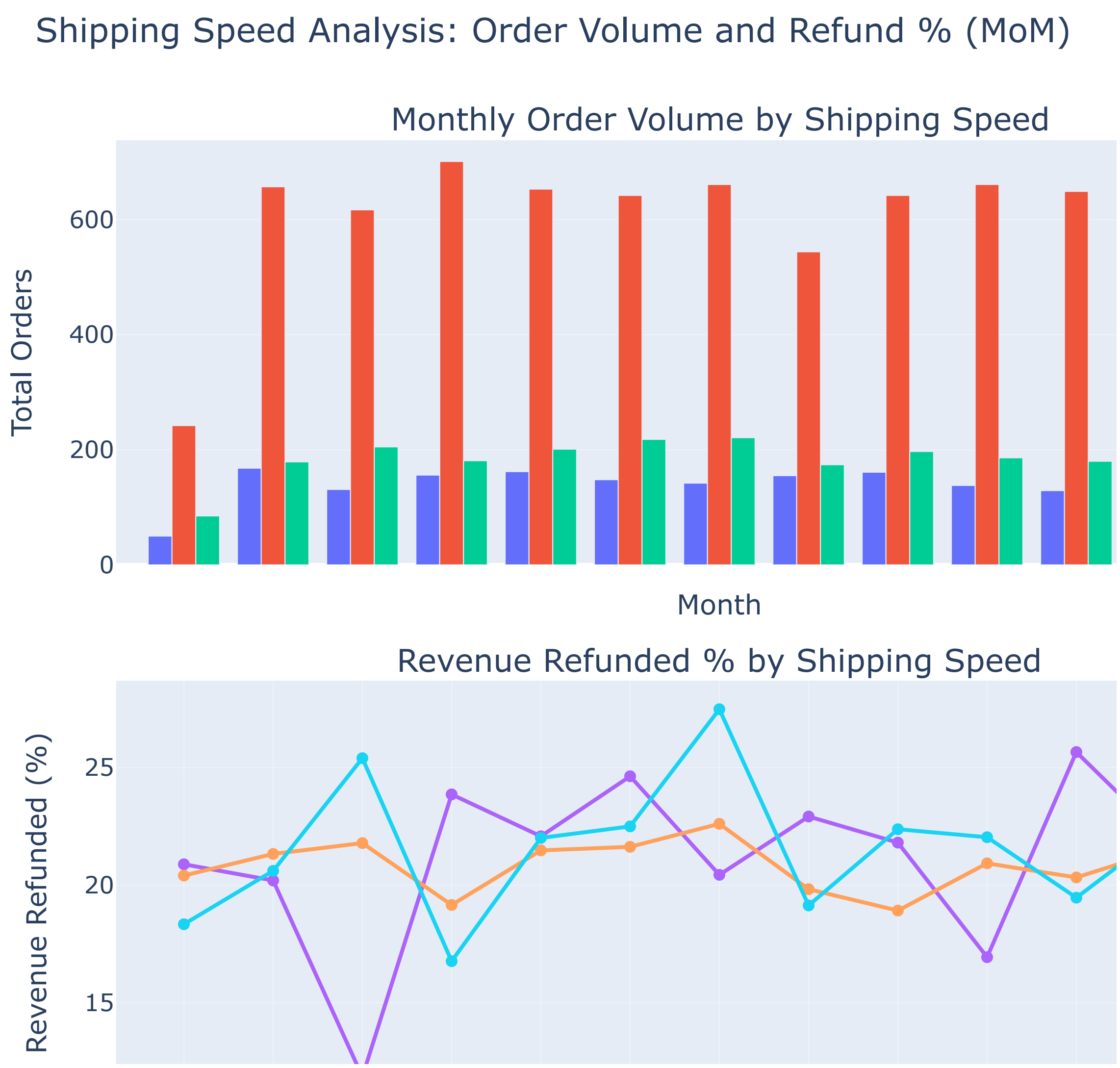
- **Review Fulfillment Quality:** Audit packaging, handling, and delivery partners for Two-Day and Overnight orders to identify root causes of higher returns.
- **Manage Expectations Proactively:**
  - Improve product visuals and descriptions for items frequently expedited.

    - For high-value or high-return items, consider proactive order follow-ups or confirmations before shipping.
- **Survey for Root Cause:** Launch **targeted post-purchase surveys** for expedited orders—especially ones returned—to identify dissatisfaction drivers (e.g., delay vs. product mismatch).
- **Analyze Competitive Gaps:** For metro regions with high Overnight demand, benchmark **competitor delivery speed** and surface any regional lag in our own network.

In [13]:

```
# MoM order Volume and Return Rate by Shipping Speed

# Load and prepare data
df = views["shipping_return_impact_summary"].copy()
df["order_month"] = pd.to_datetime(df["order_month"], errors="coerce")
df = df.dropna(subset=["order_month", "shipping_speed", "total_orders",
"revenue_refunded_pct"]).copy()
df["month_label"] = df["order_month"].dt.strftime("%B %Y")

# Set up subplots
fig = make_subplots(
    rows=2,
    cols=1,
    shared_xaxes=True,
    vertical_spacing=0.12,
    subplot_titles=[
        "Monthly Order Volume by Shipping Speed",
        "Revenue Refunded % by Shipping Speed"
    ]
)

# Plot 1: Total Orders (bar)
for speed in df["shipping_speed"].unique():
    subset = df[df["shipping_speed"] == speed]
    fig.add_trace(
        go.Bar(
            x=subset["month_label"],
            y=subset["total_orders"],
            name=f"{speed.title()} Orders"
        ),
        row=1,
        col=1
    )

# Plot 2: Revenue Refunded % (line)
for speed in df["shipping_speed"].unique():
    subset = df[df["shipping_speed"] == speed]
    fig.add_trace(
        go.Scatter(
            x=subset["month_label"],
            y=subset["revenue_refunded_pct"],
            mode="lines+markers",
            name=f"{speed.title()} Refund %",
```

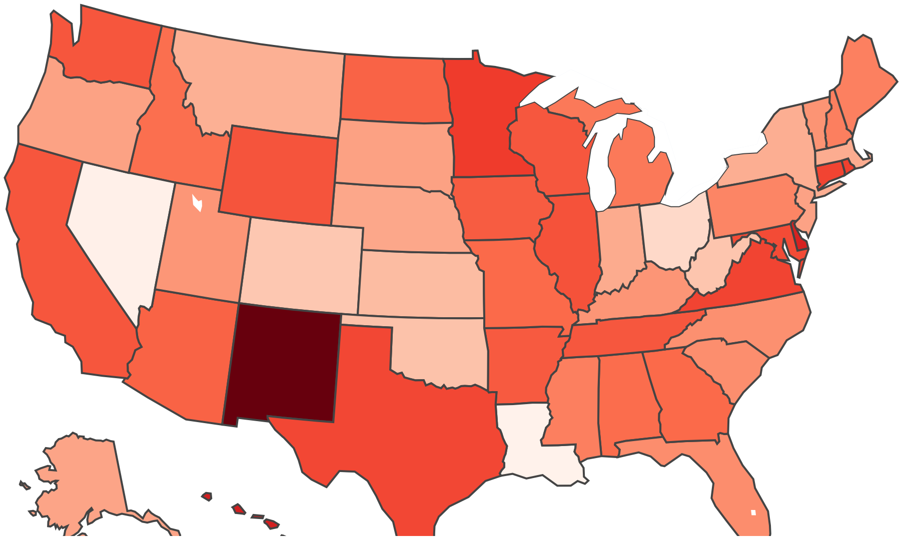
```
            hovertemplate="%{x}<br>%{y:.1f}%<extra></extra>"
        ),
        row=2,
        col=1
    )

# Final layout
fig.update_layout(
    height=750,
    title_text="Shipping Speed Analysis: Order Volume and Refund % (MoM)",
    barmode="group",
    xaxis=dict(title="Month"),
    yaxis=dict(title="Total Orders"),
    yaxis2=dict(title="Revenue Refunded (%)"),
    legend=dict(orientation="h", y=-0.2, x=0.5, xanchor="center"),
    hovermode="x unified"
)

fig.show()
```



In [14]:
```
# Percentage of Overnight shipping orders by State

# Load orders and extract state
orders_df = cleaned["orders"]
orders_df["state"] = orders_df["shipping_address"].str.extract(r",\s([A-Z]
{2})\s\d{5}")
orders_df = orders_df.dropna(subset=["state", "shipping_speed"])

# Calculate % of overnight shipping per state
state_shipping = (
```

```
    orders_df.groupby("state")
    .shipping_speed.value_counts(normalize=True)
    .rename("share")
    .reset_index()
)
overnight_share =
state_shipping[state_shipping["shipping_speed"].str.lower() ==
"overnight"]

# Plot the map
fig = px.choropleth(
    overnight_share,
    locations="state",
    locationmode="USA-states",
    color="share",
    color_continuous_scale="Reds",
    scope="usa",
    title="📦 Percent of Orders Using Overnight Shipping by State",
    labels={"share": "% Overnight Shipping"}
)
fig.update_layout(
    geo=dict(bgcolor="rgba(0,0,0,0)"),
    coloraxis_colorbar=dict(title="% Overnight")
)
fig.show()
```

📦 Percent of Orders Using Overnight Shipping by State



🔀 Segment Performance: Loyalty Tiers and CLV Buckets

This section examines how **customer segmentation—by loyalty tier and lifetime value (CLV)**—shapes both **monthly sales** and **return behavior**.

By layering financial contribution against refund patterns, we can distinguish between **high-value customers** who drive growth and **high-cost segments** that erode margins.

> These insights are essential for refining our acquisition strategy, focusing retention efforts, and designing more targeted return mitigation initiatives.

▼ 🔍 Observations

Loyalty Tier Trends

- **Platinum** members generated the highest revenue (~$15.97M), with a stable average return rate of ~21.6%.
- **Gold** customers delivered slightly less revenue (~$13.59M) but had the **lowest return rate** among all known tiers (~20.8%).

- **Silver** and **Bronze** tiers contributed lower sales and had **elevated return rates** (~21.5% and ~21.4%, respectively).
- The **Unknown** tier had surprisingly strong revenue (~$9.1M) and the **lowest return rate** (~20.2%), though data quality should be validated.

> 🔎 **Unknown** includes both guest shoppers without accounts and registered users who have not completed the signup process.

CLV Bucket Trends

- The **High CLV** segment dominated total sales (~$28M), with a return rate comparable to other buckets (~21.7%).
- **Low** and **Medium CLV** segments contributed far less revenue (under $6M each) and did not show meaningfully better return behavior.
- Return rates remained fairly consistent across CLV groups (~21%+), suggesting CLV does not strongly predict refund behavior.

▼ 🧠 Key Insights & Business Implications

- **High-Value Customers Still Return at a Cost:** Even our most valuable customers—Platinum and High CLV—are refunding over 1 in 5 dollars. This erodes margin and challenges the assumption that top-tier customers are always "safe bets."
- **Gold Tier is the Most Efficient Segment:** Gold customers offer a strong blend of sales and the lowest refund rates—potentially our “sweet spot” for retention focus.
- **CLV and Loyalty Are Correlated, But Not Aligned:** While Platinum overlaps with High CLV in spend, return stability differs slightly—suggesting other behavioral or experience factors may play a role.
- **"Unknown" Segments Represent Untapped Potential**: The “Unknown” loyalty and CLV tiers, which include both non-account shoppers (guests) and account holders who haven't yet joined the loyalty program, show promising refund performance. These customers may be currently undervalued or under-mapped and represent a significant opportunity for deeper engagement and segmentation.

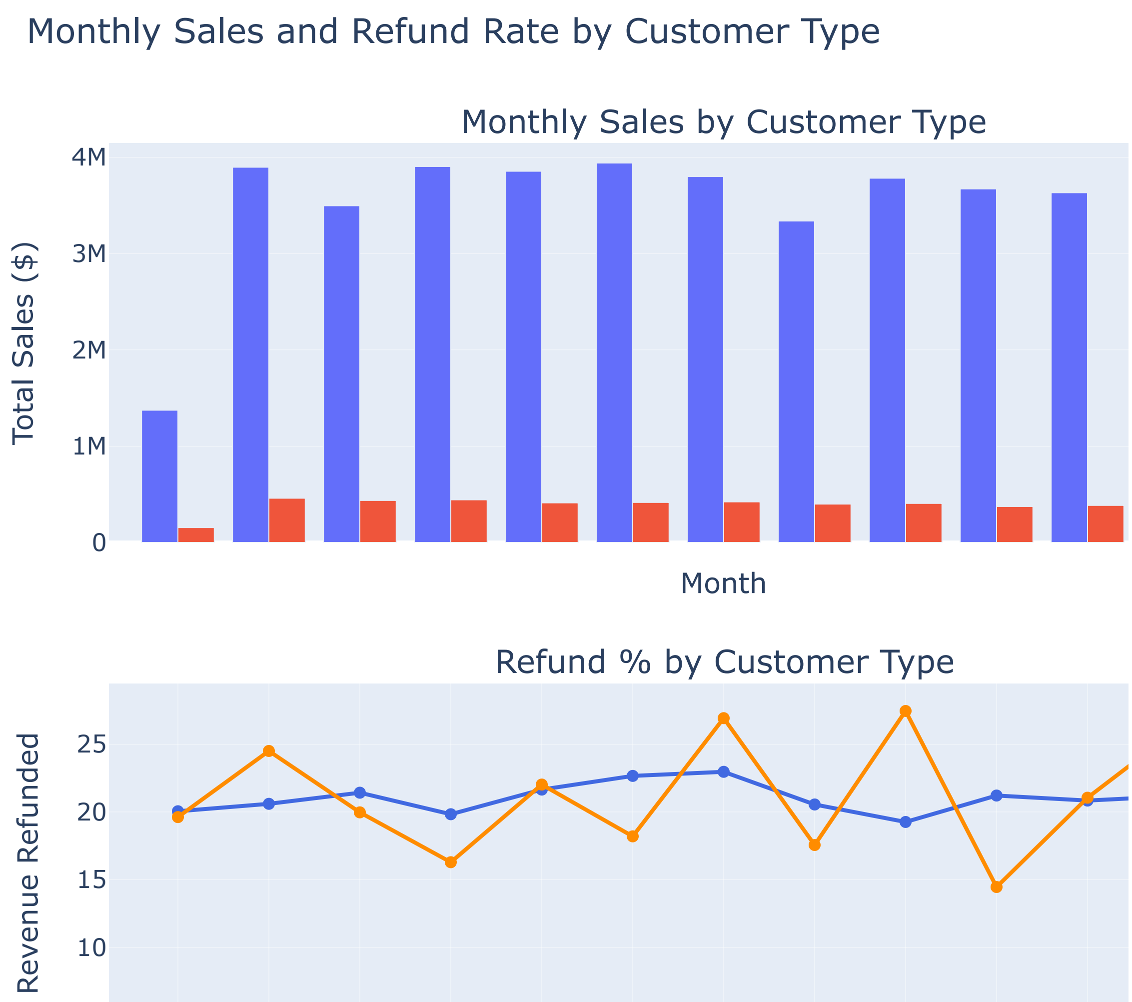
▼ 🛠️ Recommended Strategic Actions

- **Double Down on Gold Tier:** Expand lifecycle marketing and tailored retention campaigns targeting Gold members. This segment delivers efficient revenue with the lowest refund risk.
- **Analyze Platinum Return Patterns:** Investigate refund reasons among Platinum customers. Are their expectations different? Are certain products or campaigns driving dissatisfaction?
- **Drive Conversions & Engagement:**
  - **Guest to Account:** Encourage account creation for guest shoppers using incentives, messaging, and post-purchase nudges.

- **Account to Loyalty:** Nudge existing accounts into the loyalty program to move them from "Unknown" to "Known & Engaged."
- **Audit the "Unknowns":** Run a segmentation audit to better profile and classify customers currently labeled as "Unknown." This supports better personalization and future campaign targeting.
- **Align Loyalty Programs to CLV:** Explore ways to evolve loyalty tiering criteria to better match lifetime value—reward consistency, not just recent spend.

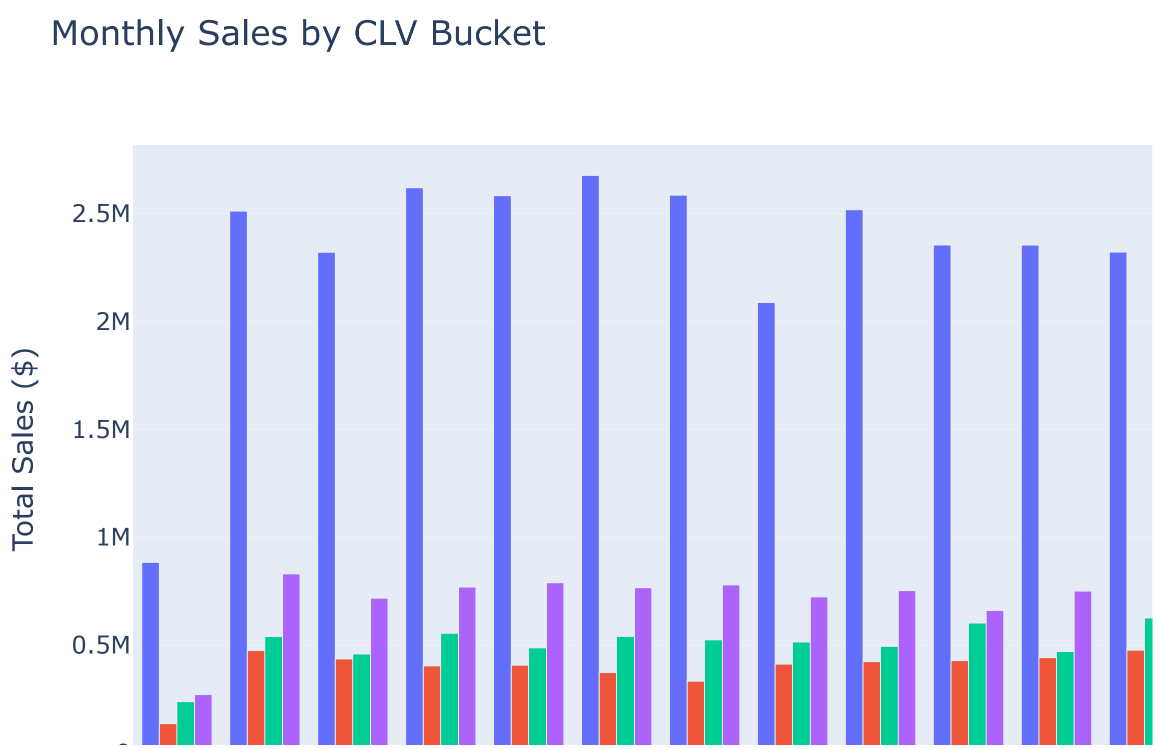
In [15]:
```
# Build visual of Sales and Refunds by Customer Type

# --- Prepare data ---
df_sales = views["monthly_channel_sales_returns"].copy()
df_segments = views["customer_segment_sales_returns"].copy()

# Clean and format
df_segments["is_guest"] = df_segments["is_guest"].astype(str).str.lower()
df_segments["customer_type"] = df_segments["is_guest"].map({
    "true": "Guest",
    "false": "Registered"
}).fillna("Registered")
df_segments["month"] = pd.to_datetime(df_segments["month"])
df_segments["month_label"] = df_segments["month"].dt.strftime("%b %Y")
df_segments = df_segments.dropna(subset=["total_sales",
"percent_revenue_returned"])

# Month ordering
valid_months = df_segments["month_label"].dropna().unique()
ordered_months = sorted(valid_months, key=lambda x: pd.to_datetime(x))
df_segments =
df_segments[df_segments["month_label"].isin(ordered_months)].copy()
df_segments["month_label"] = pd.Categorical(df_segments["month_label"],
categories=ordered_months, ordered=True)

# --- Build subplots ---
fig = make_subplots(
    rows=2,
    cols=1,
    shared_xaxes=True,
    vertical_spacing=0.15,
    subplot_titles=(
        "Monthly Sales by Customer Type",
        "Refund % by Customer Type"
    )
)

# First subplot:
# Grouped bar chart for total sales by customer type
for cust_type in df_segments["customer_type"].unique():
    cust_df = df_segments[df_segments["customer_type"] ==
cust_type].sort_values("month")
    fig.add_trace(
        go.Bar(
```

```
            x=cust_df["month_label"],
            y=cust_df["total_sales"],
            name=f"{cust_type} Sales",
            hovertemplate="%{x}<br>Total Sales: $%{y:,.0f}<br>Customer
Type: " + cust_type + "<extra></extra>"
        ),
        row=1, col=1
    )

# Update layout for grouped bars
fig.update_layout(
    barmode="group",
    title="🛍️ Monthly Sales by Customer Type",
    xaxis=dict(title="Month", tickangle=-45),
    yaxis=dict(title="Total Sales ($)"),
    hovermode="x unified"
)

# Second subplot: refund % by customer type
line_colors = {
    "Guest": "darkorange",
    "Registered": "royalblue"
}

# Second subplot: refund % by customer type
for cust_type in df_segments["customer_type"].unique():
    cust_df = df_segments[df_segments["customer_type"] ==
cust_type].sort_values("month")
    fig.add_trace(
        go.Scatter(
            x=cust_df["month_label"],
            y=cust_df["percent_revenue_returned"],
            name=f"{cust_type} (% Refunded)",
            mode="lines+markers",
            line=dict(color=line_colors.get(cust_type, None), width=2),
            hovertemplate="%{x}<br>Refund %: %{y:.1f}%<br>Customer Type: "
+ cust_type + "<extra></extra>"
        ),
        row=2, col=1
    )

# Layout
fig.update_layout(
    title="Monthly Sales and Refund Rate by Customer Type",
    xaxis=dict(title="Month", tickangle=-45),
    yaxis=dict(title="Total Sales ($)", rangemode="tozero"),
    yaxis2=dict(title="% Revenue Refunded", rangemode="tozero"),
    height=700,
    hovermode="x unified",
    legend=dict(orientation="h", x=0.5, xanchor="center", y=-0.15)
)

fig.show()
```



In [16]:
```
# Build visual of MoM sales by CLV bucket

df = views["monthly_clv_sales_returns"].copy()
df["month"] = pd.to_datetime(df["month"], errors="coerce")
df = df.dropna(subset=["month", "clv_bucket", "total_sales"])
df["month_label"] = df["month"].dt.strftime("%B %Y")

fig = px.bar(
    df,
    x="month_label",
    y="total_sales",
    color="clv_bucket",
    barmode="group",
```

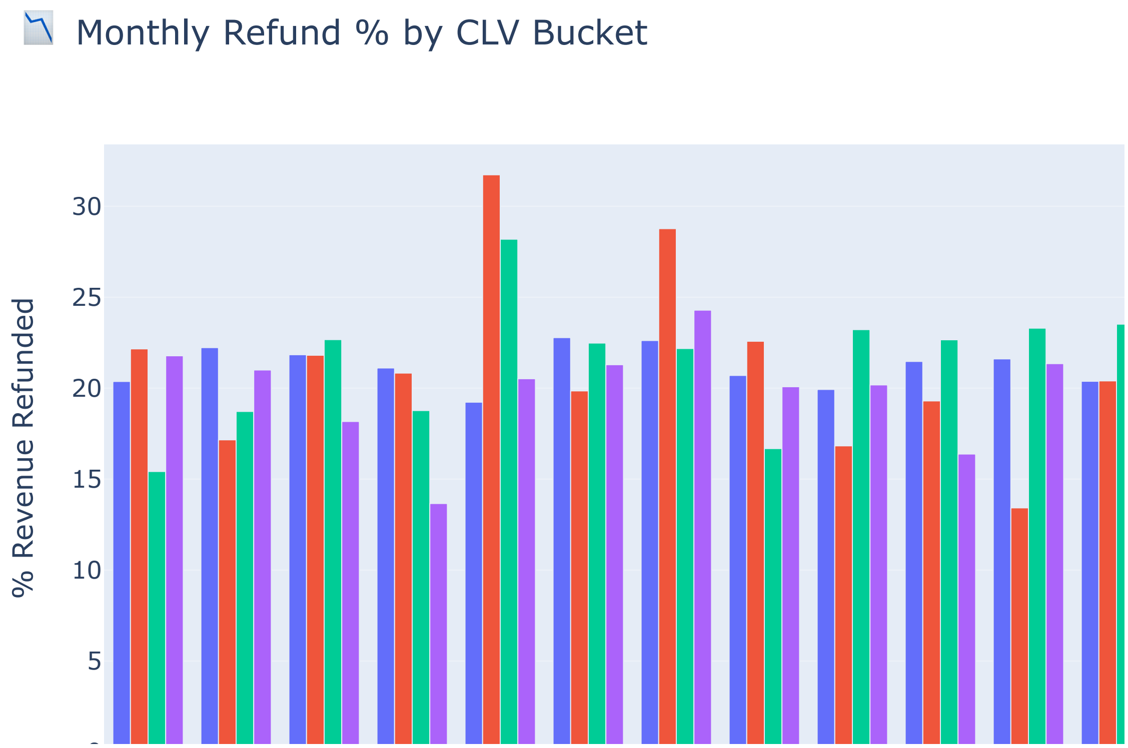
```
    title="Monthly Sales by CLV Bucket",
    labels={"month_label": "Month", "total_sales": "Total Sales ($)"}
)
fig.update_layout(xaxis_tickangle=-45)
fig.show()
```



In [17]:

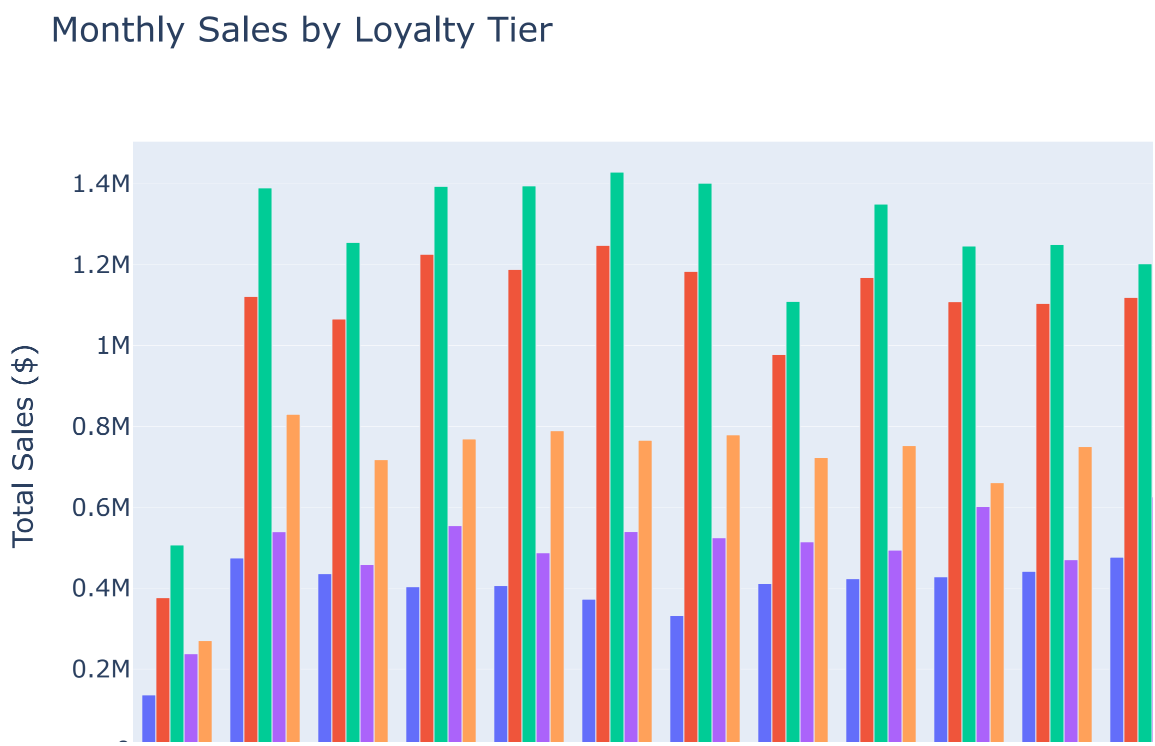
```
# Build visual of MoM refunds by CLV bucket (as grouped bar chart)

df = views["monthly_clv_sales_returns"].copy()
df["month"] = pd.to_datetime(df["month"], errors="coerce")
df = df.dropna(subset=["month", "clv_bucket", "percent_revenue_returned"])
df["month_label"] = df["month"].dt.strftime("%B %Y")

fig = px.bar(
    df,
    x="month_label",
    y="percent_revenue_returned",
    color="clv_bucket",
    barmode="group",
    title="📉 Monthly Refund % by CLV Bucket",
    labels={
        "month_label": "Month",
        "percent_revenue_returned": "% Revenue Refunded",
        "clv_bucket": "CLV Bucket"
```

```
    }
)
fig.update_layout(xaxis_tickangle=-45, hovermode="x unified")
fig.show()
```



In [18]:
```
# Build visual of MoM Sales by loyalty tier

df = views["monthly_loyalty_sales_returns"].copy()

# Ensure month is parsed properly
df["month"] = pd.to_datetime(df["month"], errors="coerce")

# Drop rows with bad or missing data
df = df.dropna(subset=["month", "loyalty_tier", "total_sales"])

# Format month label
df["month_label"] = df["month"].dt.strftime("%B %Y")

# Now plot
import plotly.express as px
fig = px.bar(
    df,
    x="month_label",
    y="total_sales",
```

```
    color="loyalty_tier",
    barmode="group",
    title="Monthly Sales by Loyalty Tier",
    labels={"month_label": "Month", "total_sales": "Total Sales ($)"}
)

fig.update_layout(xaxis_tickangle=-45)
fig.show()
```

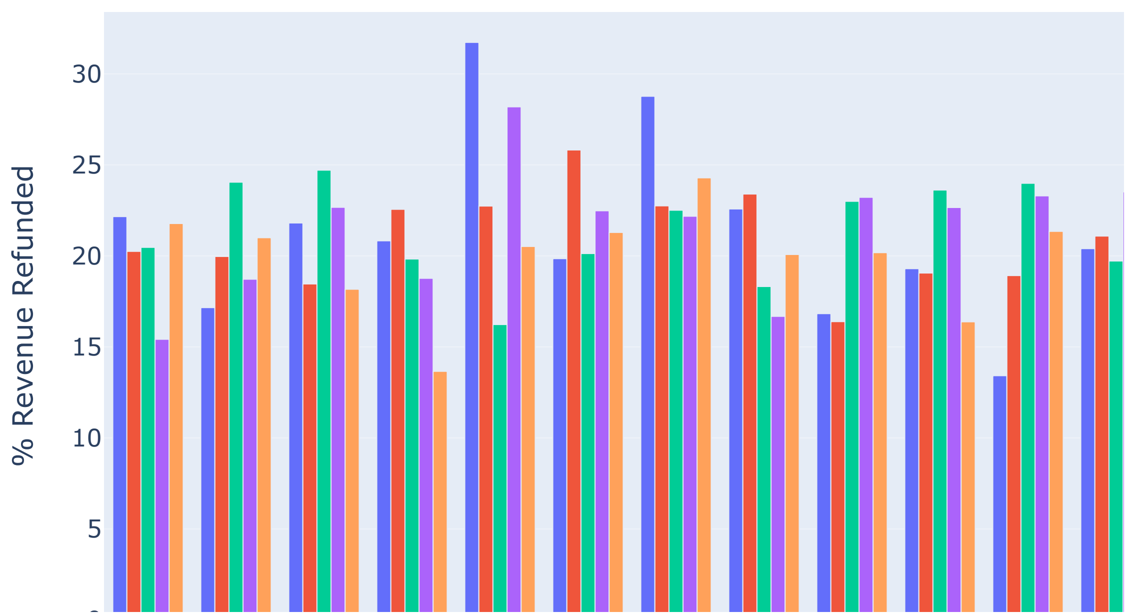


In [19]:

```
# Build visual of MoM loyalty tier returns (as grouped bar chart)

df = views["monthly_loyalty_sales_returns"].copy()
df["month"] = pd.to_datetime(df["month"])
df = df.dropna(subset=["loyalty_tier", "percent_revenue_returned"])
df["month_label"] = df["month"].dt.strftime("%B %Y")

fig = px.bar(
    df,
    x="month_label",
    y="percent_revenue_returned",
    color="loyalty_tier",
    barmode="group",
    title="📉 Monthly Refund % by Loyalty Tier",
    labels={
```

```
        "month_label": "Month",
        "percent_revenue_returned": "% Revenue Refunded",
        "loyalty_tier": "Loyalty Tier"
    }
)
fig.update_layout(xaxis_tickangle=-45, hovermode="x unified")
fig.show()
```

📉 Monthly Refund % by Loyalty Tier



🧬 Signup Cohort Analysis: Growth, Value, and Return Behavior

This section evaluates *customer behavior* based on the quarter in which account holders initially signed up.

For a young and growing e-commerce company, understanding signup cohorts is essential:

- It reveals the long-term effectiveness of acquisition strategies
- It shows how customer value and return patterns evolve over time
- It helps identify which periods brought in our most valuable—or riskiest—customers

> 📈 This analysis helps us determine whether our growth efforts are bringing in **engaged, profitable users** who contribute to long-term business value.

▼ 🔍 Observations

- **Strong Growth in Key Quarters:** Customer acquisition ramped up significantly from **Q4 2024 through Q2 2025**, peaking at nearly 500 new signups per quarter.
- **Return Rates Remain Stable:** All cohorts maintain average return rates between **18–22%**, with Q3 2024 showing the highest (~22.6%).
-
  - **Sales Track with Signup Volume:** Total revenue contributions closely follow signup volume—especially strong in **Q4 2024 ($12.96M)** and **Q2 2025 ($10.1M)**.

▼ 🧠 Key Insights & Business Implications

- **Newer Cohorts Show Promising Conversion Behavior:** Q2 and Q3 2025 cohorts exhibit **declining return rates** and steady signup growth — early signals that acquisition efforts are attracting better-fit customers.
- **Marketing Strategy Is Driving Volume:** The sustained rise in signups from Q4 2024 through Q2 2025 reflects successful promotional and outreach campaigns.
- **Sustaining Growth Requires Targeted Retention:** To preserve this momentum, the business should invest in **guest-to-account conversion tracking**, **lifecycle marketing programs**, and **ongoing cohort health monitoring**.

▼ 🛠️ Recommended Strategic Actions

- **Double Down on Q3 2025 Onboarding:** Investigate what's working—then replicate the most effective campaigns, product mixes, or flows.
- **Scale What Worked in Q2:** Reinforce the strategies behind Q2's acquisition surge—test if those playbooks scale into Q4.
- **Launch Cohort-Based Retention Tactics:** Use signup timing to tailor lifecycle messaging—e.g., loyalty nudges for Q4 2024, education for Q3 2025.

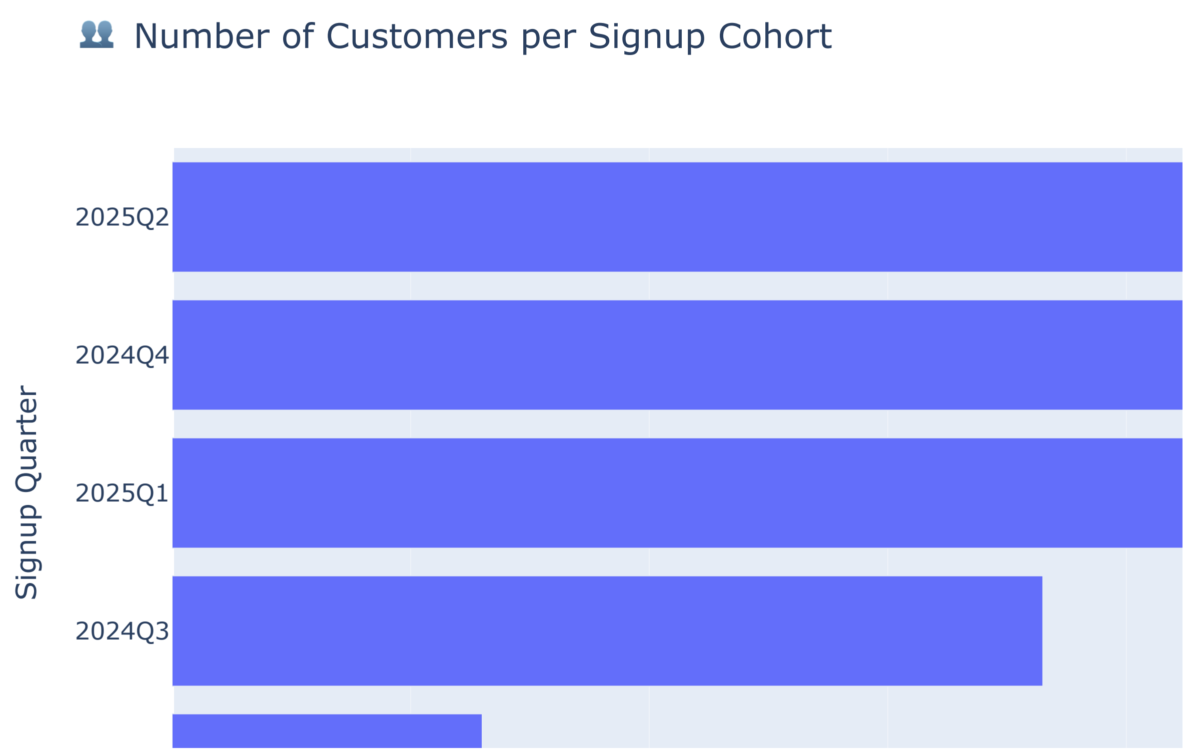
▼ 🎯 Category Preferences by Cohort

While core product interests are consistent across signup cohorts, a few patterns emerge:

- **Electronics, books, and clothing** rank as top categories for nearly all cohorts.
- **Toys and home goods** gained share among **Q2 and Q3 2025** signups—potentially influenced by seasonality or merchandising shifts.
- These findings can help tailor **personalized onboarding flows or promotional offers** by cohort.

In [20]:
```
# Cohort Size

# AVG return rate by Signup cohort

# Load and prepare cleaned data
customers = cleaned["customers"].copy()
orders = cleaned["orders"].copy()
returns = cleaned["returns"].copy()

# Standardize customer_id formats
customers["customer_id"] =
customers["customer_id"].astype(str).str.strip()
orders["customer_id"] = orders["customer_id"].astype(str).str.strip()
returns["customer_id"] = returns["customer_id"].astype(str).str.strip()

# 📅 Extract signup quarter (drop missing dates = guests)
customers["signup_date"] = pd.to_datetime(customers["signup_date"],
errors="coerce")
customers = customers.dropna(subset=["signup_date"])
customers["signup_quarter"] =
customers["signup_date"].dt.to_period("Q").astype(str)

# 💰 Clean and aggregate total order value (fix string column)
orders["order_total"] = pd.to_numeric(orders["order_total"],
errors="coerce")
clv = (
    orders.groupby("customer_id", as_index=False)["order_total"]
    .sum()
    .rename(columns={"order_total": "total_sales"})
)

# 💸 Aggregate total refunds per customer
returns["refunded_amount"] = pd.to_numeric(returns["refunded_amount"],
errors="coerce")
refunds = (
    returns.groupby("customer_id", as_index=False)["refunded_amount"]
    .sum()
    .rename(columns={"refunded_amount": "total_refunds"})
)

# 🧬 Merge cohort with CLV and refunds
cohort_df = (
    customers[["customer_id", "signup_quarter"]]
    .merge(clv, on="customer_id", how="left")
    .merge(refunds, on="customer_id", how="left")
)

# 👥 Customer count by signup cohort
cohort_counts = (
    cohort_df.groupby("signup_quarter", as_index=False)["customer_id"]
    .nunique()
    .rename(columns={"customer_id": "num_customers"})
)

# 📊 Horizontal bar plot of cohort size
```

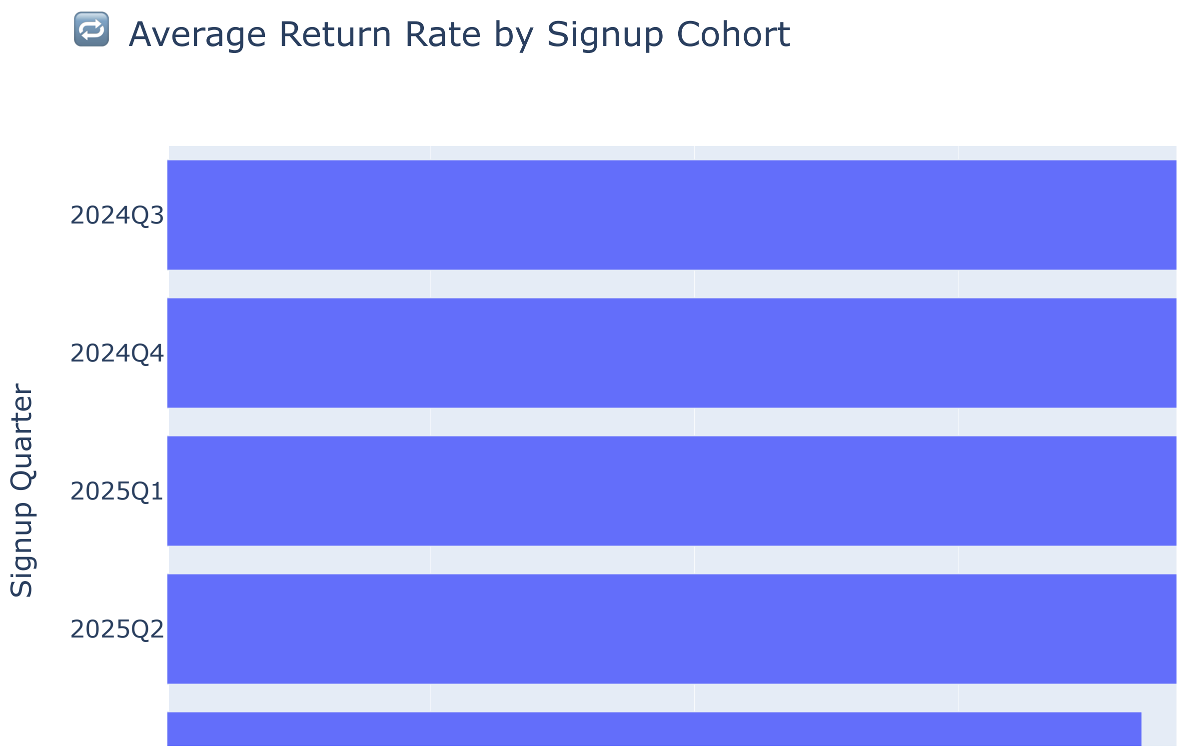
```
fig = px.bar(
    cohort_counts,
    x="num_customers",
    y="signup_quarter",
    orientation="h",
    title="👥 Number of Customers per Signup Cohort",
    labels={"signup_quarter": "Signup Quarter", "num_customers": "Customer Count"}
)
fig.update_layout(yaxis=dict(categoryorder="total ascending"))
fig.show()
```



In [21]:

```
# AVG return rate by Signup cohort

# 🧹 Clean nulls and calculate return rate
cohort_df["total_sales"] = cohort_df["total_sales"].fillna(0)
cohort_df["total_refunds"] = cohort_df["total_refunds"].fillna(0)
cohort_df["return_rate"] = cohort_df["total_refunds"] / cohort_df["total_sales"]
cohort_df.loc[~np.isfinite(cohort_df["return_rate"]), "return_rate"] = pd.NA

# 📊 Average return rate per signup cohort
summary = cohort_df.groupby("signup_quarter", as_index=False).agg(
```

```
    avg_return_rate=("return_rate", "mean")
)

# 📐 Horizontal bar plot of average return rate by signup cohort
fig = px.bar(
    summary,
    x="avg_return_rate",
    y="signup_quarter",
    orientation="h",
    title="🔁 Average Return Rate by Signup Cohort",
    labels={
        "signup_quarter": "Signup Quarter",
        "avg_return_rate": "Avg Return Rate"
    }
)
fig.update_layout(yaxis=dict(categoryorder="total ascending"),
xaxis_tickformat=".1%")
fig.show()
```



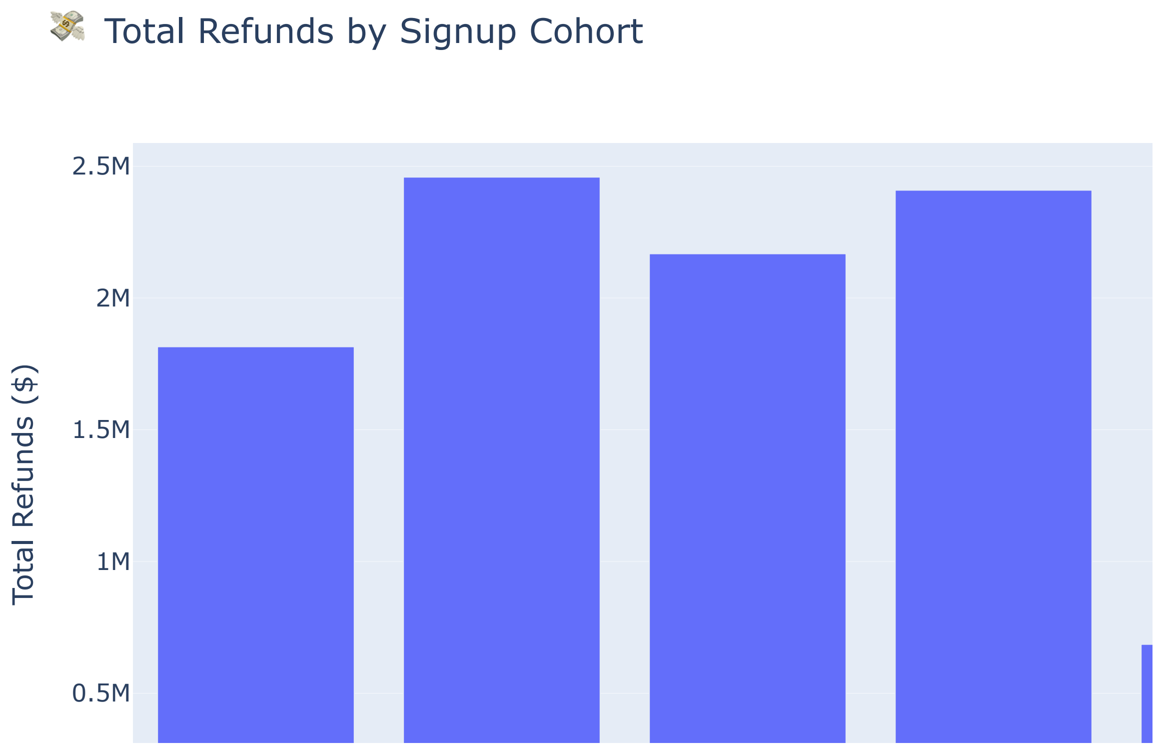
In [22]:
```
# Sales by Signup Cohort

# 📦 Total sales by signup cohort
sales_by_cohort = (
    cohort_df.groupby("signup_quarter", as_index=False)["total_sales"]
```

```
    .sum()
    .rename(columns={"total_sales": "total_sales_usd"})
)

# 📊 Bar plot of sales by cohort
fig = px.bar(
    sales_by_cohort,
    x="signup_quarter",
    y="total_sales_usd",
    title="💵 Total Sales by Signup Cohort",
    labels={"signup_quarter": "Signup Quarter", "total_sales_usd": "Total
Sales ($)"}
)
fig.update_layout(xaxis_tickangle=-45)
fig.show()
```

In [23]:

```
# Total Refunds by Signup Cohort

# Ensure correct column types
returns["refunded_amount"] = pd.to_numeric(returns["refunded_amount"],
errors="coerce")
customers["customer_id"] =
customers["customer_id"].astype(str).str.strip()
returns["customer_id"] = returns["customer_id"].astype(str).str.strip()
```

```python
customers["signup_date"] = pd.to_datetime(customers["signup_date"],
errors="coerce")
customers["signup_quarter"] =
customers["signup_date"].dt.to_period("Q").astype(str)

# Merge signup_quarter into returns
returns = returns.merge(customers[["customer_id", "signup_quarter"]],
on="customer_id", how="left")

# Drop rows with missing or invalid signup cohort
returns_clean = returns.dropna(subset=["signup_quarter"])
returns_clean = returns_clean[returns_clean["signup_quarter"] != "NaT"]

# Group total refunds by signup cohort
refund_summary = (
    returns_clean.groupby("signup_quarter", as_index=False)
["refunded_amount"]
    .sum()
    .rename(columns={"refunded_amount": "total_refunds_usd"})
)

# 📊 Plot total refunds by signup cohort
fig = px.bar(
    refund_summary,
    x="signup_quarter",
    y="total_refunds_usd",
    title="💸 Total Refunds by Signup Cohort",
    labels={"signup_quarter": "Signup Quarter", "total_refunds_usd":
"Total Refunds ($)"}
)
fig.update_layout(xaxis_tickangle=-45)
fig.show()
```

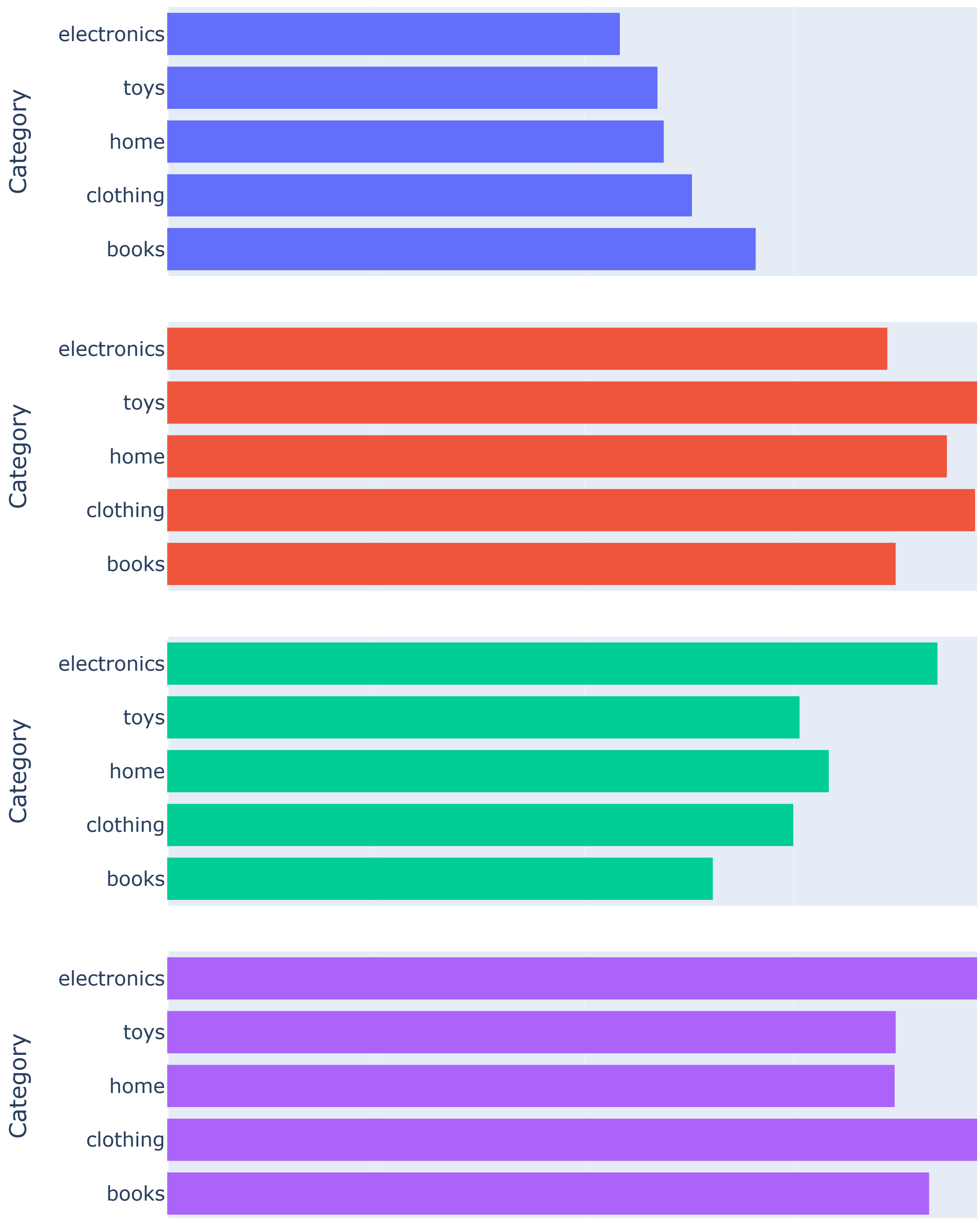


In [24]:
```
# 📦 Cohort Category Preferences: What Each Signup Cohort Bought Most in
Their First Month

import pandas as pd
import plotly.express as px

# --- Load Cleaned Data ---
orders = cleaned["orders"].copy()
order_items = cleaned["order_items"].copy()
products = cleaned["product_catalog"].copy()
customers = cleaned["customers"].copy()

# --- Standardize IDs ---
for df in [orders, order_items, customers, products]:
    for col in df.columns:
        if "id" in col or col.endswith("_id"):
            df[col] = df[col].astype(str).str.strip()

# --- Convert Dates ---
orders["order_date"] = pd.to_datetime(orders["order_date"],
errors="coerce")
customers["signup_date"] = pd.to_datetime(customers["signup_date"],
errors="coerce")
```

```
orders["order_month"] = orders["order_date"].dt.to_period("M").astype(str)
customers["signup_month"] =
customers["signup_date"].dt.to_period("M").astype(str)
customers["signup_quarter"] =
customers["signup_date"].dt.to_period("Q").astype(str)

# --- Merge Orders, Items, Customers ---
merged = (
    order_items
    .merge(orders[["order_id", "customer_id", "order_month"]],
on="order_id", how="left")
    .merge(customers[["customer_id", "signup_month", "signup_quarter"]],
on="customer_id", how="left")
    .merge(products[["product_id", "category"]].rename(columns=
{"category": "product_category"}), on="product_id", how="left")
)

# --- Filter to Orders Placed in Signup Month ---
merged["quantity"] = pd.to_numeric(merged["quantity"], errors="coerce")

# Drop rows with missing signup cohort BEFORE filtering
merged = merged.dropna(subset=["signup_quarter"])

# Now filter to only orders made in the signup month
cohort_orders = merged[merged["order_month"] ==
merged["signup_month"]].copy()

# --- Aggregate + Rank ---
top_cats = (
    cohort_orders.groupby(["signup_quarter", "product_category"],
as_index=False)["quantity"]
    .sum()
    .sort_values(["signup_quarter", "quantity"], ascending=[True, False])
)
top_cats["rank"] = top_cats.groupby("signup_quarter")
["quantity"].rank(method="first", ascending=False)

# --- Filter Top 5 per Cohort ---
top5 = top_cats[top_cats["rank"] <= 5]
top5 = top5.dropna(subset=["signup_quarter"])

top5 = top5.dropna(subset=["signup_quarter"])
top5 = top5[top5["signup_quarter"] != "NaT"]

fig = px.bar(
    top5,
    x="quantity",
    y="product_category",  # <-- FIXED HERE
    color="signup_quarter",
    facet_row="signup_quarter",
    title="🏆 Top 5 Product Categories Purchased in Signup Month by
Cohort",
    labels={"product_category": "Category", "quantity": "Units Sold"},
    orientation="h",
    height=1100
)
```

```
fig.update_layout(
    xaxis_title="Units Sold",
    yaxis_title=None,
    xaxis_tickformat=",",
    hovermode="closest"
)

fig.show()
```

🏆 Top 5 Product Categories Purchased in Signup Month by Coho



📦 Channel & Payment Method Diagnostic

This section evaluates monthly **sales and return trends** across both **order channels** and **payment methods**, offering insight into which platforms are driving **sustainable, profitable growth**—and which may be introducing **margin risk or operational overhead**.

While established channels like the website and phone show stability, emerging acquisition paths—such as eBay, Newegg, and social media—highlight our efforts to expand market share.

> However, return volatility and inconsistent performance across these channels suggest the need for **deeper cost-benefit analysis** to assess whether these investments are yielding acceptable returns—or warrant strategic reevaluation.

▼ 🔍 **Observations**

- **Our Website is the Core Driver**: Web consistently accounts for over two-thirds of total sales—driving **$38.4M** in revenue—with the **lowest refund rate (~20.3%)** across all channels.
- **Phone Orders Lead Tier 2 Channels**: With **$6.3M** in sales and a **~21.3%** refund rate, phone orders offer a reliable, commission-free stream of revenue.
- **NewEgg is Highly Volatile**: Despite only **$1.24M** in sales, NewEgg generated over **$450K in refunds**, reflecting a **~36% refund rate**—the highest among all channels.
- **eBay is Smaller but More Predictable**: eBay contributed **$1.5M** in sales with a **22.5% refund rate**, showing more stability than Newegg.
- **Social Media is Low Volume but Holds Potential**: Social Media drove **~$1.1M** in sales with a **~21.3% refund rate**, matching core channels and showing promise as a growth platform.
- **Payment Methods Are Not a Problem Driver**: Refund rates by payment method ranged from **20.1% to 22.1%**, with no standout risk factors. Sales volumes remained balanced across Credit, PayPal, ACH, and other options.

▼ 🧠 Key Insights & Business Implications

- **Web and Phone Are Core to Retention and Revenue Stability**: Together contributing nearly **$45M** in sales, these channels offer **cost-efficient, scalable growth** with stable refund behavior.
- **NewEgg Is Not Just Volatile—It's Costly**: With **36% of its sales refunded**, NewEgg represents a disproportionate margin risk that may outweigh its revenue contribution.
- **Social Media May Be an Emerging Opportunity**: Return behavior is stable despite lower volume—making this channel a candidate for low-risk, high-reward expansion.
- **Healthy Payment Method Mix**: Minimal refund variance across payment methods reinforces **strong customer trust and satisfaction** across the checkout experience.

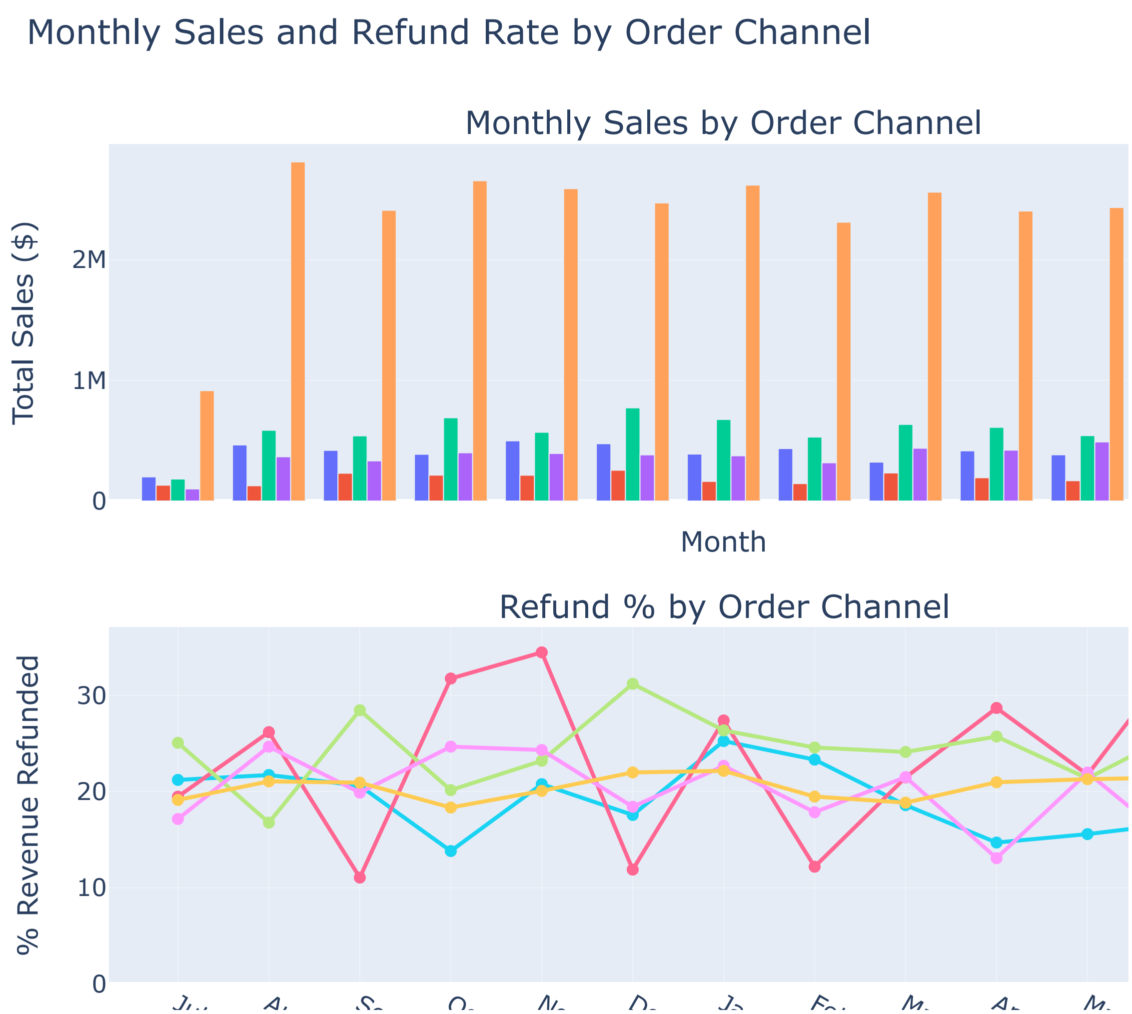
▼ 🛠 Recommended Strategic Actions

- **Double Down on Direct Channels**: Prioritize **web and phone**—which together drove nearly **$45M** in sales at refund rates below **21.5%**.
- **Reassess NewEgg's ROI**: Evaluate whether the **~$450K refund loss** on **$1.24M in sales** is sustainable, and consider **channel exit or tighter controls**.
- **Experiment with Social Spend**: With low return volatility, consider **incremental testing of paid campaigns** to grow Social Media as a cost-effective acquisition path.
- **Preserve Payment Flexibility**: No method shows red flags—maintain current diversity to continue supporting **trust and conversion**.

In [25]:
```
# Build visual of Sales and Refunds by order channel

# Prepare data
channel_df = views["monthly_channel_sales_returns"].copy()
channel_df = channel_df.dropna(subset=["order_channel", "month",
"total_sales", "percent_revenue_returned"]).copy()
channel_df["month"] = pd.to_datetime(channel_df["month"])
channel_df["month_label"] = channel_df["month"].dt.strftime("%b %Y")

# Order months for consistent x-axis
valid_months = channel_df["month_label"].dropna().unique()
ordered_months = sorted(valid_months, key=lambda x: pd.to_datetime(x))
channel_df["month_label"] = pd.Categorical(channel_df["month_label"],
categories=ordered_months, ordered=True)

# Build subplots
```

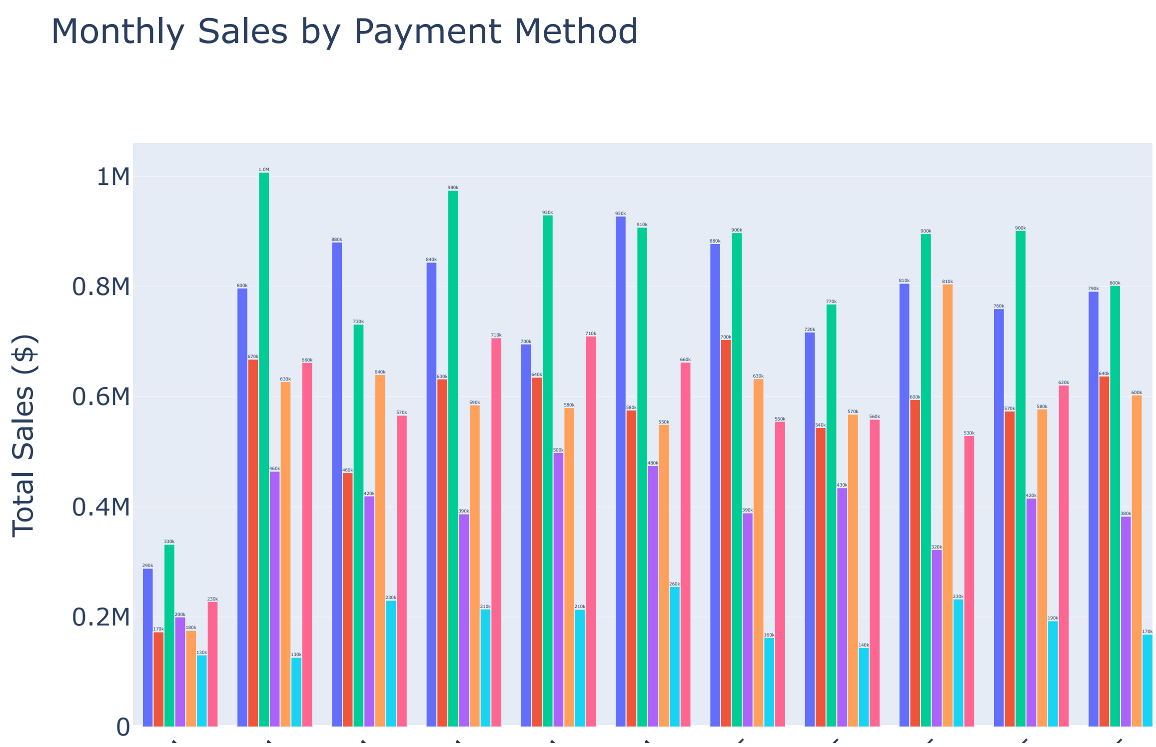
```
fig = make_subplots(
    rows=2,
    cols=1,
    shared_xaxes=True,
    vertical_spacing=0.15,
    subplot_titles=(
        "Monthly Sales by Order Channel",
        "Refund % by Order Channel"
    )
)

# First subplot: total sales by order_channel (grouped bar)
for ch in channel_df["order_channel"].unique():
    ch_df = channel_df[channel_df["order_channel"] ==
ch].sort_values("month")
    fig.add_trace(
        go.Bar(
            x=ch_df["month_label"],
            y=ch_df["total_sales"],
            name=f"{ch.title()} Sales",
            hovertemplate="%{x}<br>Total Sales: $%{y:,.0f}<br>Channel: " +
ch.title() + "<extra></extra>"
        ),
        row=1, col=1
    )

# Don't forget to set the barmode layout if not already defined:
fig.update_layout(
    barmode="group"  # Group bars side-by-side by month
)

# Second subplot: refund % by order_channel
for ch in channel_df["order_channel"].unique():
    ch_df = channel_df[channel_df["order_channel"] ==
ch].sort_values("month")
    fig.add_trace(
        go.Scatter(
            x=ch_df["month_label"],
            y=ch_df["percent_revenue_returned"],
            name=f"{ch.title()} (% Refunded)",
            mode="lines+markers",
            hovertemplate="%{x}<br>Refund %: %{y:.1f}%<br>Channel: " +
ch.title() + "<extra></extra>"
        ),
        row=2, col=1
    )

# Final layout
fig.update_layout(
    title="Monthly Sales and Refund Rate by Order Channel",
    xaxis=dict(title="Month", tickangle=-45),
    yaxis=dict(title="Total Sales ($)", rangemode="tozero"),
    yaxis2=dict(title="% Revenue Refunded", rangemode="tozero"),
    height=700,
    hovermode="x unified",
    legend=dict(orientation="h", x=0.5, xanchor="center", y=-0.15)
```

```
)

fig.show()
```



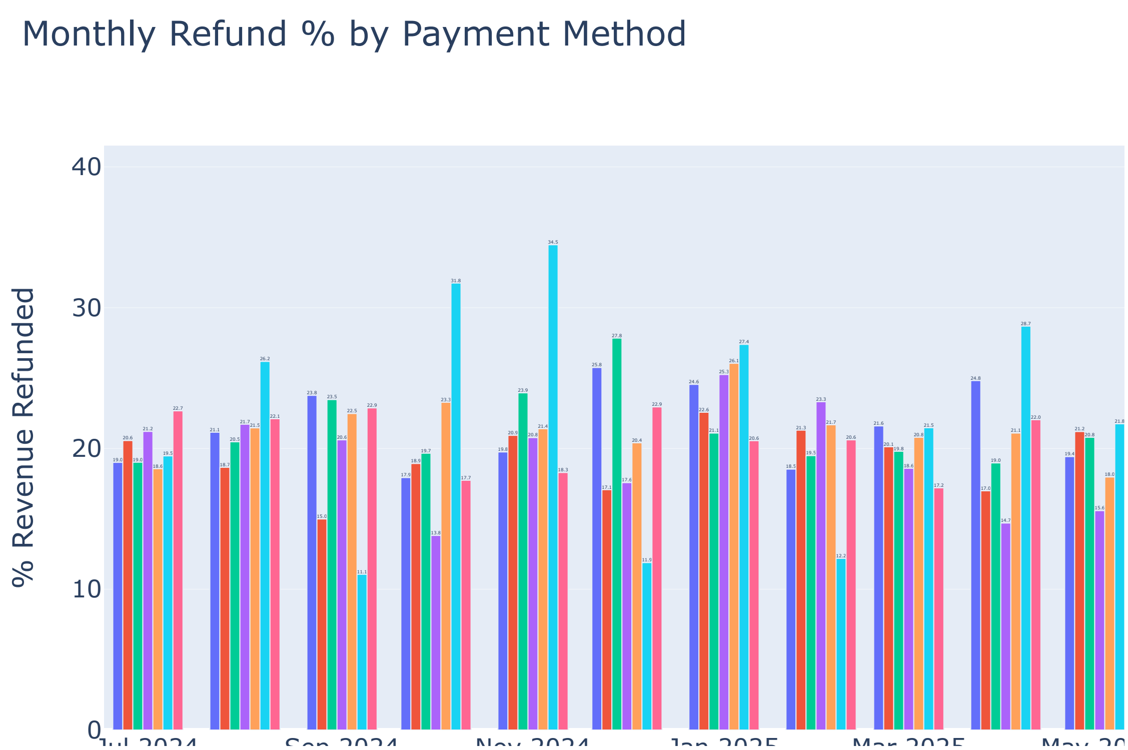
In [26]:
```
# Build visual of MoM sales by payment method

# Load and prep data
df = views["monthly_payment_sales_returns"].copy()
df["month"] = pd.to_datetime(df["month"], errors="coerce")
df = df.dropna(subset=["payment_method", "total_sales", "month"]).copy()

# Format month label
df["month_label"] = df["month"].dt.strftime("%B %Y")  # e.g., July 2024
```

```
# Create bar chart
fig = px.bar(
    df,
    x="month_label",
    y="total_sales",
    color="payment_method",
    barmode="group",
    title="Monthly Sales by Payment Method",
    labels={
        "month_label": "Month",
        "total_sales": "Total Sales ($)",
        "payment_method": "Payment Method"
    },
    text_auto=".2s"
)

# Update layout
fig.update_layout(
    xaxis_tickangle=-45,
    hovermode="x unified",
    legend=dict(title=None, orientation="h", x=0.5, xanchor="center",
y=-0.25)
)

fig.show()
```



In [27]:
```
# Build visual of return rate by payment method

# Load and prep data
df = views["monthly_payment_sales_returns"].copy()
df["month"] = pd.to_datetime(df["month"])
df = df.dropna(subset=["payment_method", "percent_revenue_returned"])

# Create bar chart
fig = px.bar(
    df,
    x="month",
    y="percent_revenue_returned",
    color="payment_method",
    barmode="group",
    title="Monthly Refund % by Payment Method",
    labels={
        "month": "Month",
        "percent_revenue_returned": "% Revenue Refunded",
        "payment_method": "Payment Method"
    },
    text_auto=".1f"
)

fig.update_layout(
```

```
    hovermode="x unified",
    legend=dict(title=None, orientation="h", x=0.5, xanchor="center",
y=-0.25)
)

fig.show()
```



🚨 High-Risk & Emerging Reseller Behavior

This section explores refund behavior at the customer level, identifying individuals who may be responsible for **disproportionate return volume**, as well as those potentially operating as **resellers**—making large, frequent purchases followed by large, frequent returns.

▼ 🔍 **Observations**

- **Top 10 Refund Customers Skew High Value:** All customers in the top 10 by total refund volume fall into the **high CLV bucket** and are members of either the **gold or platinum loyalty tier**.

- **Return Rates Suggest Risk Concentration:** Several customers maintain return rates above 40%, including **CUST-4220** and **CUST-3691**, who also show **high average return values**—over $4,300 per return.
- **Wide Distribution of Signup Channels:** Top refunders originated across all signup channels (web, phone, social, email), indicating this is not isolated to any specific source.
- **Possible Reseller Signatures:** Multiple customers (e.g., CUST-4220, CUST-4451) placed 25+ orders with **large ticket sizes and high-volume returns**, which may indicate **reseller-like behavior**—particularly if purchase types align with resale categories.

▼ 🧠 Key Insights & Business Implications

- **Not All High Refunders Are Low Value:** These customers generate high revenue and are top-tier loyalty members. But unchecked return behavior could **erode net revenue** and impact forecasting accuracy.
- **Resale-Like Behavior May Be Emerging:** High purchase volume, repeat returns, and large order values signal potential **non-personal use purchasing behavior**.
- **Current Loyalty Model May Be Too Generous:** If loyalty rewards are being unlocked and refunded quickly, it could encourage **gaming the system** among high-volume buyers.

▼ 🛠️ Recommended Strategic Actions

- **Flag and Monitor High-Risk Buyers:** Create a rule-based flag for accounts with **40%+ return rate and high order frequency** to enable proactive monitoring.
- **Conduct Manual Audit of Top Accounts:** Review items and reasons tied to top refunders—determine if they align with legitimate customer needs or potential resale behavior.
- **Reevaluate Loyalty Tier Rules:** Consider requiring minimum *net* spend (after returns) to maintain loyalty tier status.
- **Introduce Soft Limits or Friction Points:** Add return friction (e.g., re-stocking fees, manual review) for certain behaviors without blocking valid returns for all users.

In [28]:
```
# Build a view of the Top 10 Customers by Total Refunded

df = views["top_customers_by_returns"].copy()
df = df.sort_values("total_refunded", ascending=False)
```

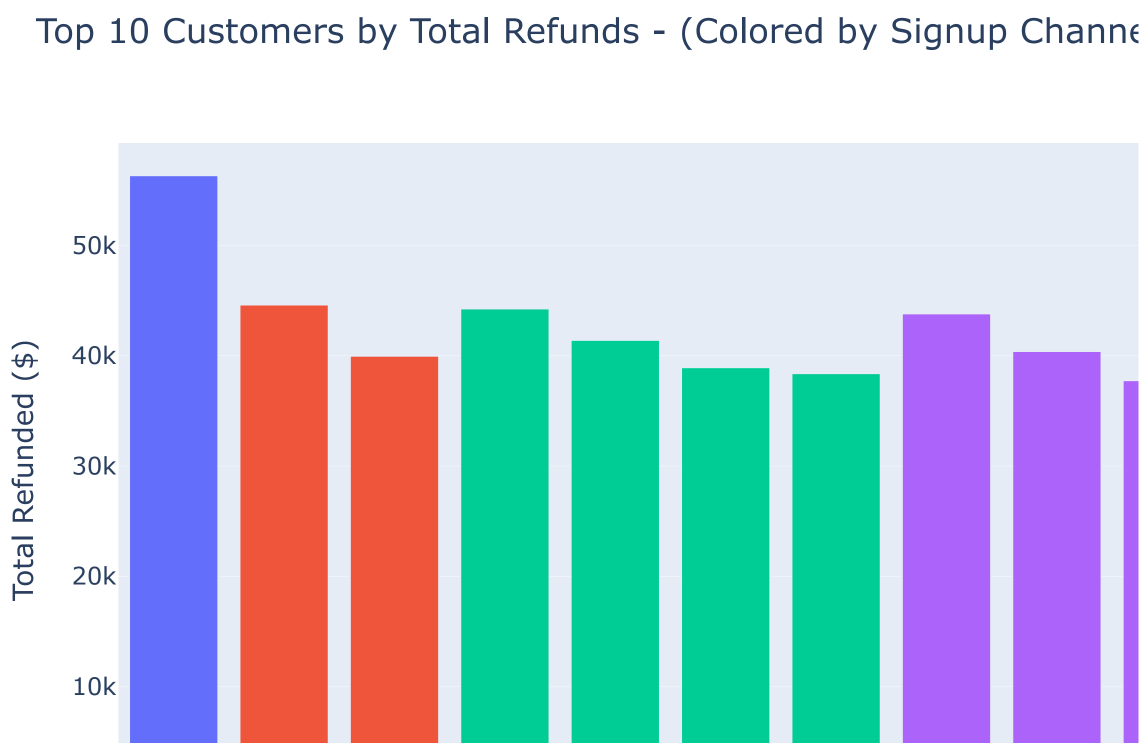
```
# Optional: add abbreviated return rate formatting
df["return_rate_pct"] = df["return_rate"].round(1).astype(str) + "%"

# Display top 10 with selected columns
cols = [
    "customer_id", "clv_bucket", "loyalty_tier", "signup_channel",
    "total_orders", "total_sales", "total_returns", "total_refunded",
"avg_return_value", "return_rate_pct"
]
df[cols].head(10)
```

Out[28]:

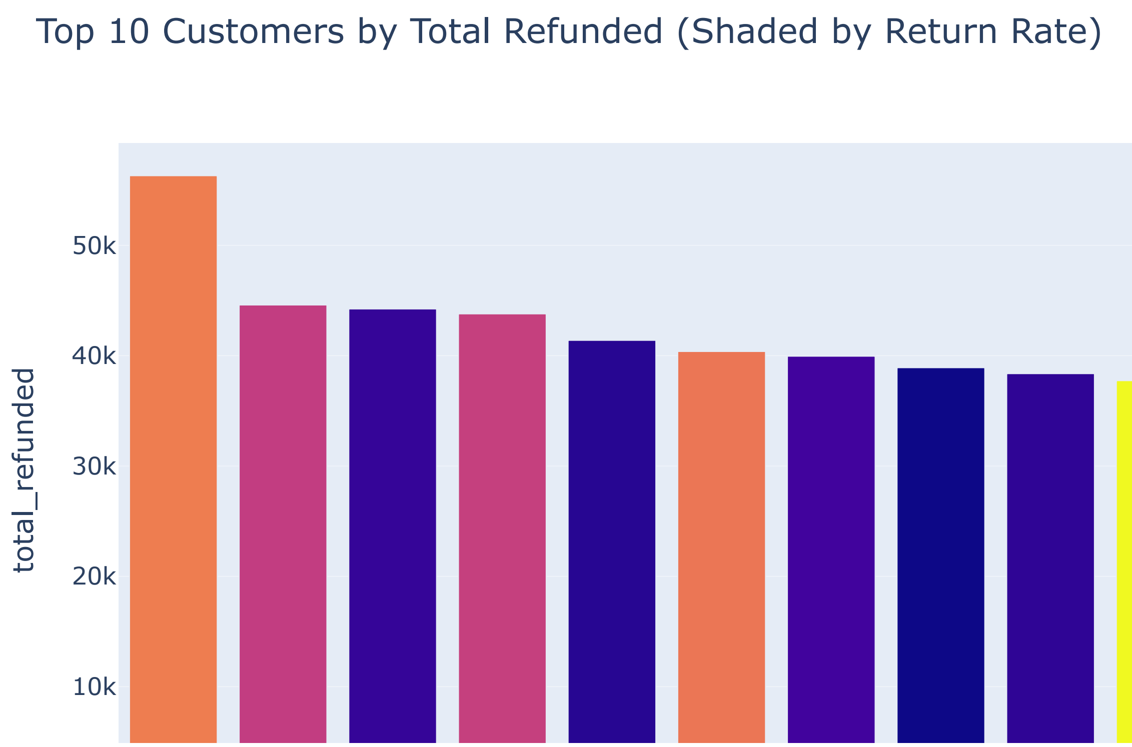
	customer_id	clv_bucket	loyalty_tier	signup_channel	total_orders	total_sales	to
0	CUST-4220	high	platinum	email	27	117084.17	
1	CUST-3691	high	platinum	phone	19	105383.54	
2	CUST-4451	high	gold	website	31	138863.44	
3	CUST-4993	high	platinum	social media	24	102667.34	
4	CUST-4897	high	platinum	website	31	132980.89	
5	CUST-4053	high	platinum	social media	17	85005.68	
6	CUST-3786	high	platinum	phone	25	122933.45	
7	CUST-4552	high	gold	website	29	130651.28	
8	CUST-3847	high	gold	website	30	121577.93	
9	CUST-5206	high	platinum	social media	14	66463.12	

In [29]:

```
# Build visual of Top 10 by Singup Channel

fig = px.bar(
    df.head(10),
    x="customer_id",
    y="total_refunded",
    color="signup_channel",
    hover_data=["clv_bucket", "total_sales", "total_returns",
"return_rate"],
    title="Top 10 Customers by Total Refunds – (Colored by Signup
Channel)"
)
fig.update_layout(xaxis_title="Customer ID", yaxis_title="Total Refunded
($)", xaxis_tickangle=-45)
fig.show()
```



In [30]:
```
# Build visual of Top 10 by with mapped colors by return rate %

fig = px.bar(
    df.head(10),
    x="customer_id",
    y="total_refunded",
    color="return_rate",  # heatmap-style color
    hover_data=["clv_bucket", "loyalty_tier", "signup_channel",
"total_sales", "total_returns"],
    title="Top 10 Customers by Total Refunded (Shaded by Return Rate)"
)
fig.update_layout(coloraxis_colorbar_title="% Return Rate",
xaxis_tickangle=-45)
fig.show()
```



📈 Return Trends by Category and Reason

This section focuses on return behavior patterns across product categories and customer-reported return reasons. Understanding these trends is essential for mitigating revenue leakage and addressing common causes of dissatisfaction.

▼ 🔍 **Observations**

- Return rates across all categories (`books` , `electronics` , `toys` , `home` , `clothing`) generally range from **18% to 25%**, but **monthly fluctuations are significant**.
- **Electronics** and **toys** show the most volatility — with sharp return rate increases during **Q4 holiday sales** and again in **early Q1**.
- **Books** and **clothing** exhibit steadier return rates, with smaller peaks and troughs.
- Among all return reasons, **"changed mind"** and **"defective"** drive the **most refund dollars**, each topping **$1.6M** in losses.
- **"Arrived damaged"** and **"product did not match description"** are also prominent — often linked to **fulfillment issues** or **expectation mismatch**.

▼ 🧠 Key Insights & Business Implications

- **Seasonality matters:** Return spikes in **January and February** are tied to **post-holiday gifting behavior**, especially for **electronics** and **toys**.
- **Durable categories return better margins:** `Clothing` and `books` perform more consistently, suggesting they may benefit from **higher-margin strategies** or **bundled offerings**.
- **Top return reasons reflect dual risk sources:**
  - *Changed mind* → **buyer behavior / impulse purchases**
  - *Defective, damaged, mismatch* → **product quality or fulfillment issues**

▼ 🛠 Recommended Strategic Actions

- **Forecast return risk by category + season:** Use return trends to predict and prepare for **seasonal surges**, especially for high-risk categories.
- **Run pre-holiday QA sweeps** for at-risk SKUs in toys and electronics — focusing on supplier consistency and packaging durability.
- **Audit listings for top-mismatch items** (e.g. “did not match description”) and improve product images or copy.
- **Quantify avoidable returns:** Segment by reason to identify **preventable return burden** driven by quality vs. customer preference.

In [31]:
```
# Sort by return rate and show top 15
views["return_rate_by_product"].sort_values(by="return_rate_percent",
ascending=False).head(15)
```

Out[31]:

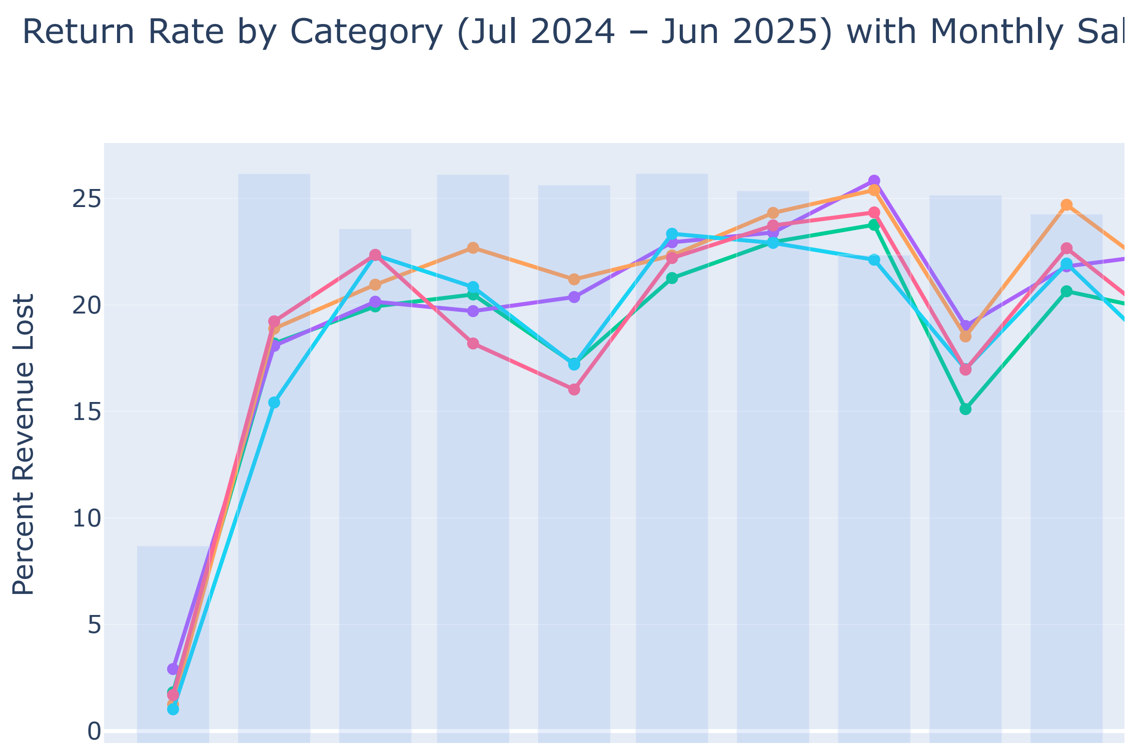
	product_id	product_name	return_reason	product_category	order_count	retu
1572	product_id	product_name	reason	category	1	
4813	697	cozy table	found a better price	home	37	
0	652	classic anthology	changed mind	books	36	
6384	1221	colorful blocks	changed mind	toys	50	
1	1095	classic memoir	changed mind	books	51	
1573	1007	durable jacket	changed mind	clothing	58	
1574	197	stylish sweater	changed mind	clothing	59	
3181	149	portable monitor	changed mind	electronics	53	
1575	526	classic sweater	changed mind	clothing	67	
1576	364	classic sweater	changed mind	clothing	47	
1577	565	durable shirt	wrong item	clothing	48	
1578	1042	classic jeans	changed mind	clothing	48	
6385	964	fun blocks	found a better price	toys	49	
1579	79	durable sweater	changed mind	clothing	42	
4814	116	rustic chair	changed mind	home	42	

In [32]:

```
# Visualize return reasons breakdown

df_reasons = views["return_reason_summary"].copy()
fig = px.bar(df_reasons.sort_values("total_refunded", ascending=False),
             x="normalized_reason", y="total_refunded",
             title="Total Refunded Amount by Return Reason")
fig.show()
```

In [33]:
```
# Visualize MoM sales vs returns by catagory

# Filter to 12-month range
df = views["category_monthly_sales_returns"].copy()
df["month"] = pd.to_datetime(df["month"], format="%Y-%m")
df = df[(df["month"] >= "2024-07-01") & (df["month"] <= "2025-06-30")]

# Group by month for total sales (to layer behind)
monthly_sales = df.groupby("month")["total_sales"].sum().reset_index()

# Create base figure
fig = go.Figure()

# Add transparent sales bar chart
fig.add_trace(go.Bar(
    x=monthly_sales["month"],
    y=monthly_sales["total_sales"],
    name="Total Sales (right axis)",
    opacity=0.4,
    marker_color="rgba(100, 149, 237, 0.4)",  # cornflowerblue with
transparency
    yaxis="y2"
))
```

```
# Add line chart traces per product category
for category in df["product_category"].unique():
    cat_data = df[df["product_category"] == category]
    fig.add_trace(go.Scatter(
        x=cat_data["month"],
        y=cat_data["percent_revenue_lost"],
        mode="lines+markers",
        name=f"{category} (return %)",
    ))

# Add secondary y-axis for sales
fig.update_layout(
    title="Return Rate by Category (Jul 2024 – Jun 2025) with Monthly
Sales Volume",
    xaxis_title="Month",
    yaxis_title="Percent Revenue Lost",
    yaxis2=dict(
        title="Total Sales ($)",
        overlaying="y",
        side="right",
        showgrid=False
    ),
    barmode="overlay",
    legend=dict(
    x=0.5,
    y=-0.2,
    xanchor='center',
    yanchor='top',
    orientation="h",
    bgcolor='rgba(255,255,255,0.8)',
    bordercolor='lightgray',
    borderwidth=1
),
    hovermode="x unified"
)

fig.show()
```



🚩 Product Return Risk: Quality Flag System Deep Dive

This section introduces a **product-level quality risk framework**—engineered to diagnose and prioritize SKUs driving disproportionate refund impact.

Built around return reason tagging, refund value thresholds, and rate-based heuristics, the system flags products by **risk tier** (⛔ High / ⚠️ Moderate / 🟢 Low) and surfaces the underlying quality signals (e.g. damage, defect, mismatch).

> More than a diagnostic tool, this framework directly answers one of the VP's core asks:
> **Which products are hurting us the most—and how do we fix it?**

This model not only pinpoints where margin and satisfaction are being eroded, but also provides a scalable mechanism to:

- Prioritize cross-functional follow-ups (Product, Fulfillment, CX)
- Monitor risk trends over time
- Integrate seamlessly into our dashboard architecture for live tracking and watchlisting

---

▼ 🔍 Observations

- The **top 20 products by total refunds** each exceed **$100K in return losses**, with 6 of them surpassing $150K.
- Quality-linked reasons like **"defective"**, **"arrived damaged"**, and **"not as described"** account for over **$4.5M in refunds** across all products.
- Items like `"compact speaker"`, `"comfortable shirt"`, and `"smart speaker"` show **high dollar impact** *and* **elevated quality return rates**.
- Several items have over **40% of their returns** linked to quality issues, flagging them for **urgent SKU-level attention**.

▼ 🧠 Key Insights & Business Implications

- **Quality failures drive substantial losses:** Quality-related returns aren't just frequent — they're **expensive**, often involving high-ticket items.
- The **risk classification system** (✅ High, ⚠️ Moderate, 🟢 Low) highlights which SKUs have a **disproportionate quality issue burden**.
- Flagged products share common themes:
  - Electronics with fragile components
  - Apparel items with **fit or durability complaints**
  - Home goods with **packaging or assembly issues**

▼ 🛠 Recommended Strategic Actions

- **Create a "Watchlist" of High-Risk SKUs:** Monitor top refund drivers monthly with flags for % of returns tied to quality.
- **Launch SKU-specific QA initiatives:** Prioritize factory and fulfillment audits for items flagged ✅ High Risk.
- **Feed return reason intelligence to merchandising:** Improve copy, sizing guidance, or packaging on items with high `"not as described"` or `"damaged"` return reasons.
- **Integrate this system into dashboards:** Add `quality_flag`, `quality_return_pct`, and `risk_tier` to internal reports to support **CX, product, and ops teams**.

In [34]:

```
# 'Build Quality Flags' and Aggragate buy "return_reason"

# Extract the view
df = views["return_rate_by_product"].copy()

# Normalize return_reason just in case
df["return_reason"] = df["return_reason"].str.lower().str.strip()
```

```
# Define quality-related reasons
quality_reasons = [
    "defective",
    "arrived damaged",
    "product did not match description",
    "damaged in transit",
    "missing parts"
]

# Tag rows as quality-related
df["quality_flag"] = df["return_reason"].isin(quality_reasons)

# Filter to only quality-related return reasons
quality_issues = df[df["quality_flag"]].sort_values("return_rate_percent",
ascending=False)

# Show top 15 products
quality_issues.head(15)
```

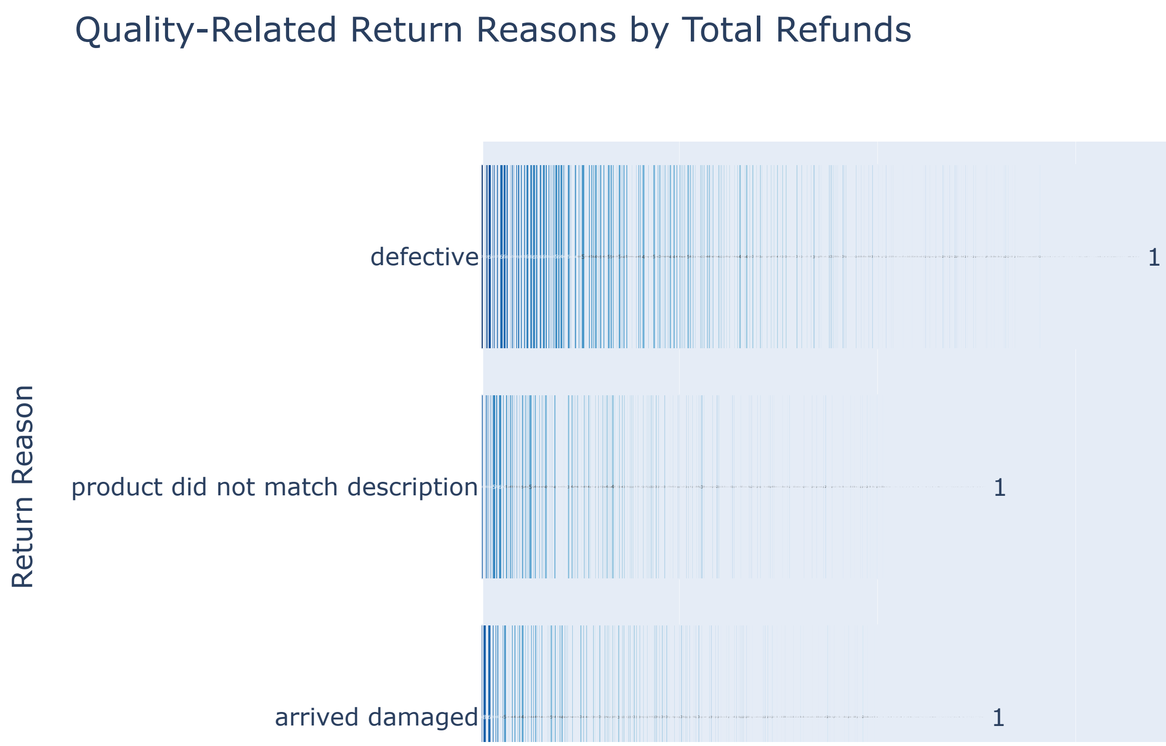
Out[34]:

	product_id	product_name	return_reason	product_category	order_count	retu
3182	583	portable headphones	defective	electronics	50	
4815	350	cozy lamp	product did not match description	home	43	
4816	703	modern chair	defective	home	44	
3187	729	smart monitor	defective	electronics	61	
3188	866	wireless camera	defective	electronics	46	
3189	1251	smart camera	defective	electronics	54	
4818	849	elegant chair	product did not match description	home	47	
3190	923	wireless monitor	defective	electronics	47	
4	886	illustrated guide	arrived damaged	books	47	
3191	692	portable camera	arrived damaged	electronics	48	
6387	1198	interactive puzzle	defective	toys	48	
1588	378	classic sweater	defective	clothing	49	
3193	570	wireless speaker	defective	electronics	49	
3192	342	smart camera	product did not match description	electronics	49	
3195	1208	portable camera	defective	electronics	58	

In [35]:

```
# Heatmap of Quality-Flagged Returns by Total Refund Value

fig = px.bar(
    quality_issues,
    x="total_refunded",
    y="return_reason",
    orientation="h",
    color="return_rate_percent",
    text="return_count",
    title="Quality-Related Return Reasons by Total Refunds",
    labels={
        "total_refunded": "Total Refunded ($)",
        "return_reason": "Return Reason",
```

```
        "return_rate_percent": "Avg Return Rate (%)",
        "return_count": "Returned Items"
    },
    color_continuous_scale="blues"
)

fig.update_layout(yaxis=dict(categoryorder="total ascending"))
fig.show()
```

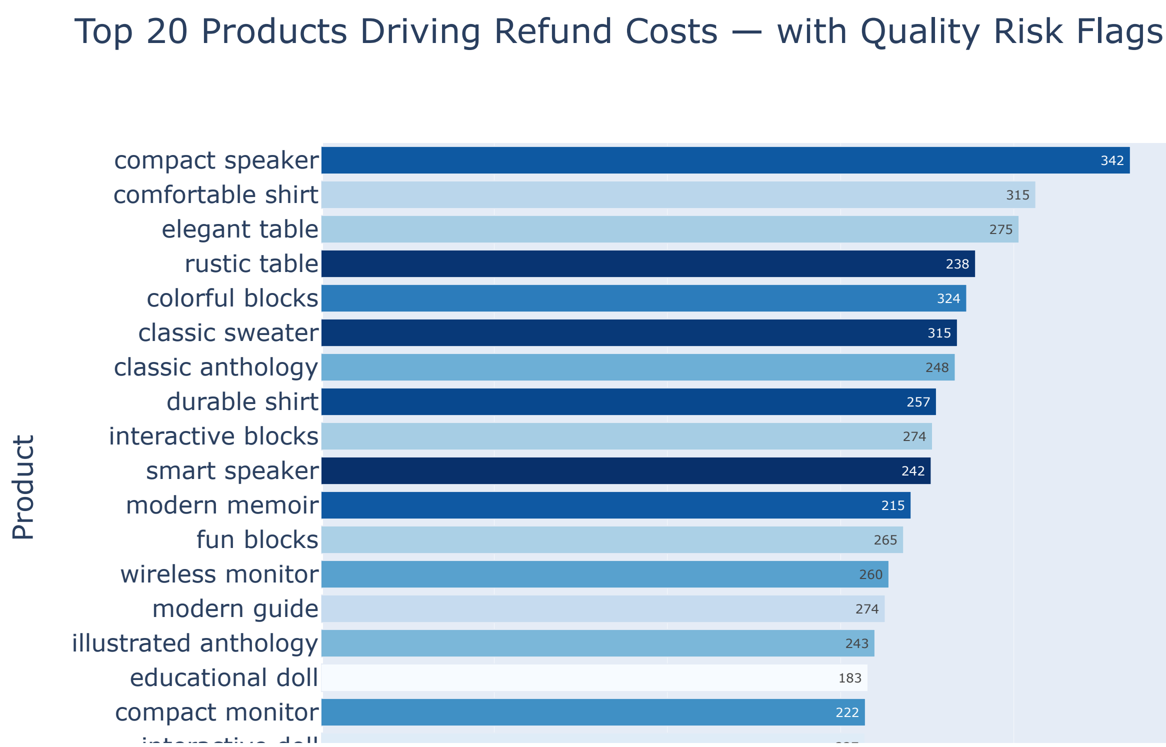


▶ 📊 **Plot Metrics and Use**

In [36]:
```
# Top 20 quality-flagged products by refund amount

# Define quality-related reasons
quality_reasons = [
    "defective",
    "arrived damaged",
    "product did not match description",
    "damaged in transit",
    "missing parts"
]

# Normalize and flag
df["return_reason"] = df["return_reason"].str.lower().str.strip()
```

```
df["quality_flag"] = df["return_reason"].isin(quality_reasons)

# Group summary stats (from full data)
grouped = (
    df.groupby("product_name")
    .agg(
        total_refunded=("total_refunded", "sum"),
        total_returns=("return_count", "sum"),
        avg_return_rate=("return_rate_percent", "mean"),
    )
)

# Compute top return reason per product (quality flagged only)
def top_reason_info(sub_df):
    reasons = sub_df["return_reason"].tolist()
    counts = Counter(reasons)
    top_reason, top_count = counts.most_common(1)[0]
    total = sub_df["return_count"].sum()
    ratio = top_count / total if total > 0 else 0
    label = f"{top_reason} ({top_count}/{total}, {ratio:.0%})"
    dominant = "⛔" if ratio > 0.5 else "✅"
    return pd.Series({
        "top_return_reason": f"{label} {dominant}",
        "dominant_reason_ratio": ratio,
        "is_strongly_dominant": ratio > 0.5
    })

# Apply top reason logic on quality-flagged returns only
reason_info = (
    df[df["quality_flag"]]
    .groupby("product_name", group_keys=False)
    .apply(top_reason_info, include_groups=False)
    .reset_index()
)

# Compute quality dominance over total returns
def quality_dominance_info(sub_df):
    total = sub_df["return_count"].sum()
    quality_count = sub_df[sub_df["quality_flag"]]["return_count"].sum()
    ratio = quality_count / total if total > 0 else 0

    # Tiered flag logic
    if ratio > 0.5:
        flag = "⛔ High Risk"
    elif ratio > 0.33:
        flag = "⚠️ Moderate Risk"
    else:
        flag = "🟢 Low Risk"

    return pd.Series({
        "quality_return_pct": ratio,
        "quality_flag_label": flag,
        "is_quality_dominant": ratio > 0.5  # Retain original binary if
needed
    })
```

```
# Apply to full dataset
dominance_info = (
    df.groupby("product_name", group_keys=False)
    .apply(quality_dominance_info, include_groups=False)
    .reset_index()
)

# Final merge and top 20 slice
top_products = (
    reason_info
    .merge(dominance_info, on="product_name")
    .merge(grouped.reset_index(), on="product_name")
    .sort_values("total_refunded", ascending=False)
    .head(20)
)

# Plot
fig = px.bar(
    top_products,
    x="total_refunded",
    y="product_name",
    orientation="h",
    color="avg_return_rate",
    text="total_returns",
    title="Top 20 Products Driving Refund Costs — with Quality Risk
Flags",
    labels={
        "total_refunded": "Total Refunded ($)",
        "product_name": "Product",
        "avg_return_rate": "Avg Return Rate (%)",
        "total_returns": "Return Count",
        "top_return_reason": "Top Quality Flag",
        "quality_flag_label": "Return Type Summary"
    },
    color_continuous_scale="blues",
    hover_data=["top_return_reason", "quality_return_pct",
"quality_flag_label"]
)

fig.update_layout(
    yaxis=dict(
        categoryorder="total ascending",
        tickmode="linear"  # Ensures all ticks are shown
    )
)
fig.show()
```



▶ 📊 **Plot Metrics and Use**

📤 Dataframe Export

The following key data tables are available for operational handoff or further analysis:

- `top_20_quality_quality_risk_products.csv`
  Contains the top 20 products by total refund value, annotated with quality risk tiers and return reasons. Ideal for product, CX, or fulfillment teams to prioritize remediation.

- `product_quality_risk_summary.csv`
  A comprehensive summary of SKUs with return rates, reason distributions, and quality flags. Useful for ongoing monitoring and dashboard integration.

In [37]:
```
# Export revised dataset and additonal data

EXPORT_DIR = "exports/vp_req_analysis"
os.makedirs(EXPORT_DIR, exist_ok=True)
```

```
# Example: export top risk products
top_products.to_csv(os.path.join(EXPORT_DIR,
"top_20_quality_risk_products.csv"), index=False)

# Optional: export final modeling table
df_product_risk = views["return_rate_by_product"].copy()
df_product_risk.to_csv(os.path.join(EXPORT_DIR,
"product_quality_risk_summary.csv"), index=False)
```

✅ Closing Note

This diagnostic provides a comprehensive, data-driven foundation for understanding both the **momentum** and **friction points** in our commercial performance. While topline sales and acquisition trends are encouraging, refund dynamics reveal clear pressure points on **profitability and retention**.

Our quality risk system and customer segmentation analyses offer actionable paths forward. As we move into Q4, aligning product, CX, and operations around these insights will be critical to maintaining growth while improving revenue efficiency.

> Note: All data in this analysis is simulated using a modular generator. Metrics are representative of realistic business scenarios but do not reflect real company performance.