INSIGHT 3 — Generational Price Resistance

Year-over-Year Price Ceiling Detection

```
# ■■ BLOCK 3: GENERATIONAL PRICE RESISTANCE ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■

# Year-over-Year price and review volume by category
yoy = df.groupby(['category', 'release_year']).agg(
    avg_price=('price', 'mean'),
    total_reviews=('review_count', 'sum'),
    avg_reviews=('review_count', 'mean'),
    sku_count=('product_id', 'count')
).reset_index()

# Compute correlation per category
corr_results = []
for cat in sorted(df['category'].unique()):
    cat_data = yoy[yoy['category'] == cat].sort_values('release_year')
    if len(cat_data) >= 3:
        r, p = corr_test(cat_data['avg_price'].values, cat_data['total_reviews'].values)
        corr_results.append({
            'category': cat,
            'price_review_corr': r,
            'p_value': p,
            'price_change_pct': ((cat_data['avg_price'].iloc[-1] / cat_data['avg_price'].iloc[0]) - 1) *
                100,
            'review_change_pct': ((cat_data['total_reviews'].iloc[-1] / cat_data['total_reviews'].iloc[0]
                ) - 1) * 100
        })
corr_df = pd.DataFrame(corr_results).sort_values('price_review_corr')

# Find the worst offender (most negative correlation)
worst_cat = corr_df.iloc[0]['category']

# ■■ VISUALIZATION: Dual-axis for each category (2×3 grid) ■■■■■■■■■■■■■■■■■■■■
fig, axes = plt.subplots(2, 3, figsize=(16, 9))
axes_flat = axes.flatten()

for idx, cat in enumerate(sorted(df['category'].unique())):
    ax1 = axes_flat[idx]
    cat_data = yoy[yoy['category'] == cat].sort_values('release_year')
    years = cat_data['release_year']

    # Price line (left axis)
    color1 = PALETTE[0]
    ax1.plot(years, cat_data['avg_price'], 'o-', color=color1, linewidth=2, markersize=5)
    ax1.set_ylabel('Avg Price ($)', color=color1, fontsize=9)
    ax1.tick_params(axis='y', labelcolor=color1)
    ax1.set_xlabel('Release Year')

    # Review volume (right axis)
    ax2 = ax1.twinx()
    color2 = PALETTE[3]
    ax2.plot(years, cat_data['total_reviews'], 's--', color=color2, linewidth=2, markersize=5)
    ax2.set_ylabel('Total Reviews', color=color2, fontsize=9)
    ax2.tick_params(axis='y', labelcolor=color2)

    # Correlation annotation
    r_val = corr_df[corr_df['category'] == cat]['price_review_corr'].values[0]
    bg_color = '#FFCCCC' if r_val < -0.3 else '#CCFFCC' if r_val > 0.3 else '#FFFFCC'
    ax1.set_title(f'{cat}\nr = {r_val:.3f}', fontweight='bold',
                  fontsize=10, bbox=dict(boxstyle='round,pad=0.3', facecolor=bg_color, alpha=0.7))
    ax1.set_xticks(sorted(years.unique()))
    ax1.tick_params(axis='x', rotation=45)

plt.suptitle('Generational Price Resistance: Price (blue) vs Review Volume (red)\n'
             'Negative r = Price Ceiling Hit', fontweight='bold', fontsize=13, y=1.02)
plt.tight_layout()
plt.savefig('block3_price_resistance.png', dpi=150, bbox_inches='tight')
plt.show()

print(f"\n{'='*70}")
print(f"WORST PRICE CEILING: {worst_cat}")
```

INSIGHT 3 — Generational Price Resistance

Year-over-Year Price Ceiling Detection

```
print(f"  Pearson r (price vs. review volume) = {corr_df.iloc[0]['price_review_corr']:.4f}")
print(f"  p-value = {corr_df.iloc[0]['p_value']:.6f}")
print(f"{'='*70}")

# ■■ BLOCK 3: SUMMARY TABLE FOR EXPORT ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■

print("■ TABLE 3: Price-Review Correlation by Category (Sorted: Worst Ceiling First)")
print("="*95)
export_corr = corr_df.copy()
export_corr['price_review_corr'] = export_corr['price_review_corr'].round(4)
export_corr['p_value'] = export_corr['p_value'].map('{:.6f}'.format)
export_corr['price_change_pct'] = export_corr['price_change_pct'].map('{:+.1f}%'.format)
export_corr['review_change_pct'] = export_corr['review_change_pct'].map('{:+.1f}%'.format)
export_corr['verdict'] = corr_df['price_review_corr'].apply(
    lambda r: 'CEILING HIT' if r < -0.3 else ('Watch' if r < 0 else 'Pricing Power')
)
export_corr.columns = ['Category', 'Price-Review r', 'p-value', 'Price Chg (2015-2024)',
                       'Review Chg (2015-2024)', 'Verdict']
print(export_corr.to_string(index=False))
```

Block 3 Complete | 88 lines of executable Python | Statistical tests + Visualization + Export table | 2 page(s)