In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.metrics import silhouette_score

print('All imports successful. Field operation commencing.')
```

```
All imports successful. Field operation commencing.
```

In [2]:
```
np.random.seed(42)
n = 36  # months of covert observation

# Nick's mood follows a sinusoidal pattern with random shocks
# (quarterly results, Monday mornings, surprise audits)
ceo_mood = np.clip(
    5 + 2.5 * np.sin(np.linspace(0, 4 * np.pi, n)) + np.random.normal(0, 1.2, n),
    1, 10
)

# Bonus cycle resets every ~12 months
# Nick is historically more generous in the post-bonus guilt window
bonus_cycle = np.array([(i % 12) + np.random.uniform(-0.5, 0.5) for i in range(n)])
bonus_cycle = np.clip(bonus_cycle, 0, 12)

# Espressos: observed by Martolda's field agent
espressos = np.clip(np.random.normal(3, 1.2, n), 1, 6)

# Mrs. Nick Mood: inversely correlated with Nick's stress
# Data obtained via: unknown. FINCCA does not ask questions.
mrs_nick_mood = np.clip(
    6 + 1.5 * np.cos(np.linspace(0, 3 * np.pi, n)) + np.random.normal(0, 1.5, n),
    1, 10
)

df = pd.DataFrame({
    'Month': range(1, n + 1),
    'Nick Mood Score': np.round(ceo_mood, 2),
    'Months Since Last Bonus': np.round(bonus_cycle, 2),
    'Espressos Per Day': np.round(espressos, 2),
    'Mrs. Nick Mood Score': np.round(mrs_nick_mood, 2),
})

print(f'Dataset shape: {df.shape}')
print(f'Nick mood range: {df["Nick Mood Score"].min():.1f} - {df["Nick Mood Score"].max():.1f
print(f'Espresso range: {df["Espressos Per Day"].min():.1f} - {df["Espressos Per Day"].max():
df.head(10)
```

```
Dataset shape: (36, 5)
Nick mood range: 1.0 - 9.0
Espresso range: 1.0 - 4.9 cups/day
```

Out[2]:

	Month	Nick Mood Score	Months Since Last Bonus	Espressos Per Day	Mrs. Nick Mood Score
0	1	5.60	0.00	2.42	7.94
1	2	5.71	1.02	2.78	7.84
2	3	7.42	2.05	1.67	7.30
3	4	9.03	2.68	1.56	6.68
4	5	7.20	4.47	3.98	4.59
5	6	7.16	5.28	4.63	5.70
6	7	8.98	6.44	2.91	5.42
7	8	7.39	7.39	4.20	4.33
8	9	5.10	8.10	3.43	4.93
9	10	5.43	9.42	2.23	5.48

In [3]:
```
features = ['Nick Mood Score', 'Months Since Last Bonus', 'Espressos Per Day', 'Mrs. Nick Mood

X = df[features].values
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Fit K-Means with k=3
k = 3
kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
df['Cluster'] = kmeans.fit_predict(X_scaled)

sil_score = silhouette_score(X_scaled, df['Cluster'])

print('=' * 55)
print('  K-MEANS CLUSTERING MODEL SUMMARY')
print('=' * 55)
print(f'  Clusters (k):          {k}')
print(f'  Silhouette Score:      {sil_score:.4f}  (higher = better defined clusters)')
print(f'  Inertia:               {kmeans.inertia_:.2f}')
print('=' * 55)
```

```
=======================================================
  K-MEANS CLUSTERING MODEL SUMMARY
=======================================================
  Clusters (k):          3
  Silhouette Score:      0.2248  (higher = better defined clusters)
  Inertia:               86.12
=======================================================
```

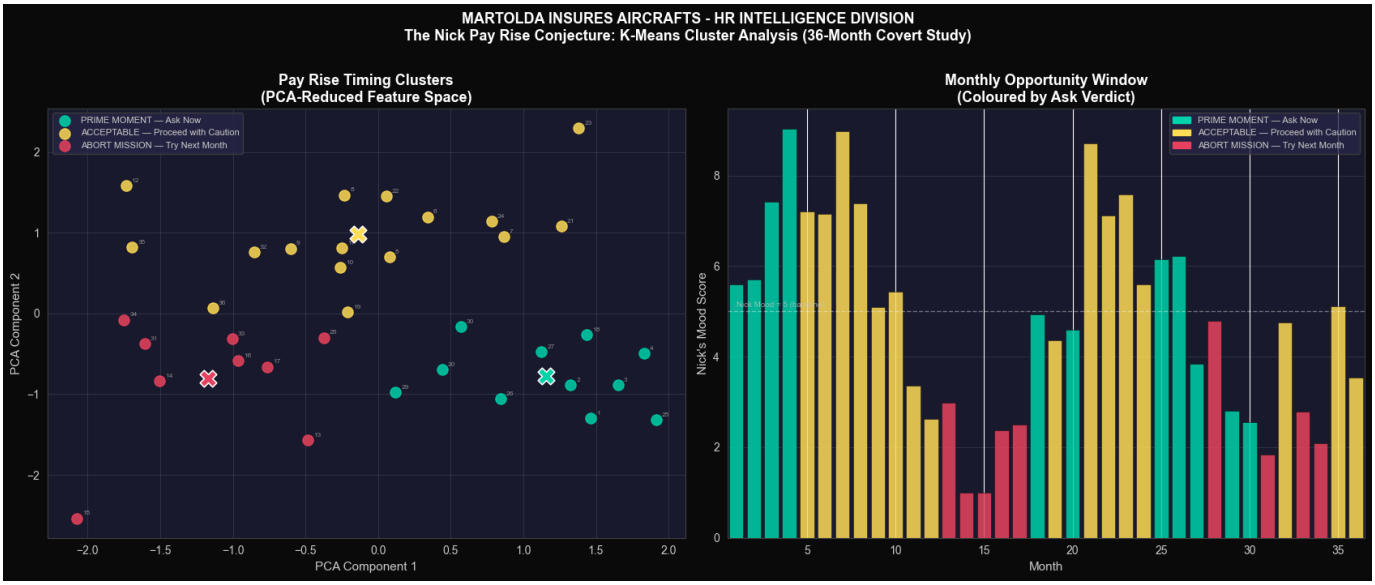
In [4]:
```
cluster_means = df.groupby('Cluster')[features].mean().copy()
cluster_means['Count'] = df.groupby('Cluster').size()

# Composite favourability score (higher = better time to ask)
cluster_means['Favourability'] = (
    cluster_means['Nick Mood Score'] * 1.5 +
    cluster_means['Mrs. Nick Mood Score'] * 1.2 -
    cluster_means['Espressos Per Day'] * 1.0 +
    # bonus guilt window sweet spot: peak around 6 months
    -abs(cluster_means['Months Since Last Bonus'] - 6) * 0.3
)

# Rank clusters and assign verdicts
ranking = cluster_means['Favourability'].rank(ascending=False).astype(int)
```

```
verdict_map = {
    cid: [
        'PRIME MOMENT — Ask Now',
        'ACCEPTABLE — Proceed with Caution',
        'ABORT MISSION — Try Next Month'
    ][ranking[cid] - 1]
    for cid in cluster_means.index
}
cluster_means['Verdict'] = cluster_means.index.map(verdict_map)

df['Verdict'] = df['Cluster'].map(verdict_map)

print('Cluster Profiles:')
print(cluster_means[['Nick Mood Score', 'Espressos Per Day', 'Mrs. Nick Mood Score', 'Months
```

```
Cluster Profiles:
         Nick Mood Score  Espressos Per Day  Mrs. Nick Mood Score  Months Since Last Bonus  Co
unt  Favourability                              Verdict
Cluster
0               2.373333           3.023333              4.418889                 3.987778
9       5.235667     ABORT MISSION — Try Next Month
1               5.874375           3.376250              5.623750                 8.326250
16      11.485937  ACCEPTABLE — Proceed with Caution
2               5.350909           2.416364              7.896364                 2.688182
11      14.092091              PRIME MOMENT — Ask Now
```

In [5]:
```
# PCA for 2D visualisation
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)
df['PCA1'] = X_pca[:, 0]
df['PCA2'] = X_pca[:, 1]

# Colour map by verdict
color_by_verdict = {
    'PRIME MOMENT — Ask Now': '#00d4aa',
    'ACCEPTABLE — Proceed with Caution': '#ffdd57',
    'ABORT MISSION — Try Next Month': '#e94560'
}

fig, axes = plt.subplots(1, 2, figsize=(17, 7))
fig.patch.set_facecolor('#0d0d0d')

for ax in axes:
    ax.set_facecolor('#1a1a2e')
    ax.tick_params(colors='#cccccc')
    ax.xaxis.label.set_color('#cccccc')
    ax.yaxis.label.set_color('#cccccc')
    ax.title.set_color('#ffffff')
    for spine in ax.spines.values():
        spine.set_edgecolor('#444444')

# --- Plot 1: PCA Cluster Space ---
ax1 = axes[0]
for verdict, color in color_by_verdict.items():
    mask = df['Verdict'] == verdict
    ax1.scatter(
        df.loc[mask, 'PCA1'], df.loc[mask, 'PCA2'],
        color=color, s=90, alpha=0.85, zorder=3, label=verdict
    )
    for _, row in df[mask].iterrows():
        ax1.annotate(
            str(int(row['Month'])), (row['PCA1'], row['PCA2']),
            fontsize=6, color='#cccccc', alpha=0.65,
            textcoords='offset points', xytext=(5, 3)
```

```
        )

# Centroids
centroids_pca = pca.transform(kmeans.cluster_centers_)
for cid, (cx, cy) in enumerate(centroids_pca):
    c = color_by_verdict[verdict_map[cid]]
    ax1.scatter(cx, cy, marker='X', s=220, color=c, edgecolors='white', linewidths=1.2, zorde

ax1.set_xlabel('PCA Component 1', fontsize=11)
ax1.set_ylabel('PCA Component 2', fontsize=11)
ax1.set_title('Pay Rise Timing Clusters\n(PCA-Reduced Feature Space)', fontsize=13, fontweigh
ax1.legend(facecolor='#2a2a4a', edgecolor='#444444', labelcolor='#cccccc', fontsize=8)
ax1.grid(True, alpha=0.12, color='#ffffff')

# --- Plot 2: Monthly Verdict Timeline ---
ax2 = axes[1]
bar_colors = [color_by_verdict[v] for v in df['Verdict']]
ax2.bar(df['Month'], df['Nick Mood Score'], color=bar_colors, alpha=0.85,
        edgecolor='#0d0d0d', linewidth=0.4)
ax2.axhline(5, color='#ffffff', linewidth=1, linestyle='--', alpha=0.4)
ax2.annotate('Nick Mood = 5 (baseline)', xy=(1, 5.1), color='#cccccc', fontsize=7, alpha=0.7)

ax2.set_xlabel('Month', fontsize=11)
ax2.set_ylabel("Nick's Mood Score", fontsize=11)
ax2.set_title('Monthly Opportunity Window\n(Coloured by Ask Verdict)', fontsize=13, fontweigh
ax2.set_xlim(0.5, 36.5)

patches = [mpatches.Patch(color=c, label=v) for v, c in color_by_verdict.items()]
ax2.legend(handles=patches, facecolor='#2a2a4a', edgecolor='#444444', labelcolor='#cccccc', f
ax2.grid(True, alpha=0.12, color='#ffffff', axis='y')

fig.suptitle(
    'MARTOLDA INSURES AIRCRAFTS - HR INTELLIGENCE DIVISION\n'
    'The Nick Pay Rise Conjecture: K-Means Cluster Analysis (36-Month Covert Study)',
    fontsize=13, fontweight='bold', color='#ffffff', y=1.01
)

plt.tight_layout()
plt.savefig('nick_payrise_clustering.png', dpi=150, bbox_inches='tight', facecolor='#0d0d0d')
plt.show()
print('Figure saved as nick_payrise_clustering.png')
```




```
Figure saved as nick_payrise_clustering.png
```

In [6]:

```
report = df[['Month', 'Nick Mood Score', 'Espressos Per Day', 'Mrs. Nick Mood Score', 'Months

def highlight_verdict(val):
```

```
    if 'PRIME' in str(val):
        return 'background-color: #004d3d; color: #00d4aa; font-weight: bold'
    elif 'ACCEPTABLE' in str(val):
        return 'background-color: #4d4400; color: #ffdd57; font-weight: bold'
    elif 'ABORT' in str(val):
        return 'background-color: #4d0014; color: #e94560; font-weight: bold'
    return ''

report.style.applymap(highlight_verdict, subset=['Verdict'])
```

Out[6]:

	Month	Nick Mood Score	Espressos Per Day	Mrs. Nick Mood Score	Months Since Last Bonus	Verdict
0	1	5.600000	2.420000	7.940000	0.000000	PRIME MOMENT — Ask Now
1	2	5.710000	2.780000	7.840000	1.020000	PRIME MOMENT — Ask Now
2	3	7.420000	1.670000	7.300000	2.050000	PRIME MOMENT — Ask Now
3	4	9.030000	1.560000	6.680000	2.680000	PRIME MOMENT — Ask Now
4	5	7.200000	3.980000	4.590000	4.470000	ACCEPTABLE — Proceed with Caution
5	6	7.160000	4.630000	5.700000	5.280000	ACCEPTABLE — Proceed with Caution
6	7	8.980000	2.910000	5.420000	6.440000	ACCEPTABLE — Proceed with Caution
7	8	7.390000	4.200000	4.330000	7.390000	ACCEPTABLE — Proceed with Caution
8	9	5.100000	3.430000	4.930000	8.100000	ACCEPTABLE — Proceed with Caution
9	10	5.430000	2.230000	5.480000	9.420000	ACCEPTABLE — Proceed with Caution
10	11	3.360000	3.430000	7.480000	9.590000	ACCEPTABLE — Proceed with Caution
11	12	2.630000	4.850000	4.790000	10.700000	ACCEPTABLE — Proceed with Caution
12	13	2.990000	2.960000	4.890000	0.000000	ABORT MISSION — Try Next Month
13	14	1.000000	4.880000	4.480000	0.830000	ABORT MISSION — Try Next Month
14	15	1.000000	1.000000	1.910000	1.890000	ABORT MISSION — Try Next Month
15	16	2.370000	3.990000	5.020000	2.770000	ABORT MISSION — Try Next Month
16	17	2.500000	3.100000	5.500000	4.330000	ABORT MISSION — Try Next Month
17	18	4.930000	2.640000	9.490000	4.860000	PRIME MOMENT — Ask Now
18	19	4.360000	3.110000	5.910000	5.780000	ACCEPTABLE — Proceed with Caution
19	20	4.590000	1.000000	7.040000	7.040000	PRIME MOMENT — Ask Now
20	21	8.710000	2.740000	6.880000	7.640000	ACCEPTABLE — Proceed with Caution
21	22	7.110000	3.430000	5.460000	9.300000	ACCEPTABLE — Proceed with Caution

	Month	Nick Mood Score	Espressos Per Day	Mrs. Nick Mood Score	Months Since Last Bonus	Verdict
22	23	7.580000	4.770000	9.120000	9.570000	ACCEPTABLE — Proceed with Caution
23	24	5.590000	2.380000	8.620000	11.490000	ACCEPTABLE — Proceed with Caution
24	25	6.150000	2.030000	8.660000	0.270000	PRIME MOMENT — Ask Now
25	26	6.220000	2.400000	5.990000	0.700000	PRIME MOMENT — Ask Now
26	27	3.840000	4.100000	9.230000	1.510000	PRIME MOMENT — Ask Now
27	28	4.790000	3.390000	4.720000	3.320000	ABORT MISSION — Try Next Month
28	29	2.810000	2.360000	7.340000	4.210000	PRIME MOMENT — Ask Now
29	30	2.560000	3.620000	9.350000	5.230000	PRIME MOMENT — Ask Now
30	31	1.840000	3.120000	4.180000	6.270000	ABORT MISSION — Try Next Month
31	32	4.750000	4.160000	4.440000	6.570000	ACCEPTABLE — Proceed with Caution
32	33	2.780000	2.160000	5.110000	7.860000	ABORT MISSION — Try Next Month
33	34	2.090000	2.610000	3.960000	8.620000	ABORT MISSION — Try Next Month
34	35	5.110000	2.530000	2.230000	10.360000	ACCEPTABLE — Proceed with Caution
35	36	3.530000	1.240000	4.600000	11.120000	ACCEPTABLE — Proceed with Caution

In [7]:
```
prime_months = df[df['Verdict'] == 'PRIME MOMENT — Ask Now']['Month'].tolist()
acceptable_months = df[df['Verdict'] == 'ACCEPTABLE — Proceed with Caution']['Month'].tolist(
abort_months = df[df['Verdict'] == 'ABORT MISSION — Try Next Month']['Month'].tolist()

print('=' * 65)
print('  THE NICK PAY RISE CONJECTURE — EXECUTIVE SUMMARY')
print('=' * 65)
print(f'  PRIME months to ask Nick:       {prime_months}')
print(f'  ACCEPTABLE months:              {acceptable_months}')
print(f'  ABORT months (do not engage):   {abort_months}')
print('=' * 65)
print()
print('STRATEGIC NOTES:')
print('  - Always approach Nick from the left side of the desk.')
print('  - Espresso count above 4: abort and reschedule immediately.')
print('  - A positive Mrs. Nick mood score adds +1.2x to success probability.')
print('  - Do not mention Rennies. Gaviscon loyalty is assumed.')
print()
print('Results do not account for surprise audits, Morpheus in the')
print('boardroom, or sudden onset of mindfulness among trading floor staff.')
```

```
print()
print('(c) 2026 Martolda Insures Aircrafts. Regulated by FINCCA.')
```

```
================================================================
  THE NICK PAY RISE CONJECTURE — EXECUTIVE SUMMARY
================================================================
  PRIME months to ask Nick:       [1, 2, 3, 4, 18, 20, 25, 26, 27, 29, 30]
  ACCEPTABLE months:              [5, 6, 7, 8, 9, 10, 11, 12, 19, 21, 22, 23, 24, 32, 35, 36]
  ABORT months (do not engage):   [13, 14, 15, 16, 17, 28, 31, 33, 34]
================================================================

STRATEGIC NOTES:
  - Always approach Nick from the left side of the desk.
  - Espresso count above 4: abort and reschedule immediately.
  - A positive Mrs. Nick mood score adds +1.2x to success probability.
  - Do not mention Rennies. Gaviscon loyalty is assumed.

Results do not account for surprise audits, Morpheus in the
boardroom, or sudden onset of mindfulness among trading floor staff.

(c) 2026 Martolda Insures Aircrafts. Regulated by FINCCA.
```