# Cap One Airline

## Recommended Tools and Code

Python *– open source, convenient, easy to manipulate and visualize large data set.*

Jupyter Lab *–open-source IDE and easy to use.*

MS Power BI *– User friendly, cost-effective and optimize use within MS Azure cloud.*

## Source code (CapOneAirline.py)

### -Import libraries

import pandas as pd *- for data frame, manipulation and analysis*

import matplotlib.pyplot as plt *- for visualization*

import seaborn as sns *- for visualization*

import numpy as np *- for calculation*

from matplotlib.ticker import FuncFormatter *- formatt the y axis for top 10 profitable*

### -Import/read the Airline data files

```
flights = pd.read_csv("Flights.csv")
tickets = pd.read_csv("Tickets.csv")
airports = pd.read_csv("Airport_Codes.csv")
```

### -Static variables declaration based on scenario

airplane_price = 90000000 *- 90M dollars price per plane*

ticket_price = 150 *- Average ticket price*

fomc_cost_per_mile = 8 *- 8 dollars Fuel, Oil, Maintenance and Crew*

dio_cost_per_mile = 1.18 *- Depreciation, Insurance and Other*

large_airport_cost = 10000 *- 10k dollars for large airport*

medium_airport_cost = 5000 *- 5k dollars for medium airport*

delay_per_minute_cost = 75 *- 75 dollars delay per minute if over 15 minutes*

delay_minutes = 15 *- 15 minutes delay only allowed for free*

passenger_cap = 200 *- 200 count of passenger per plane*

baggage_fee = 70 *- 70 for round trip*

# Cap One Airline

### -Cleaning the data based only on what is needed

clean_flights = flights[flights['CANCELLED'] == 0].copy() *– only flights that are not cancelled*

clean_flights['route_code'] = clean_flights.apply(lambda x: '-'.join(sorted([x['ORIGIN'], x['DESTINATION']])), axis=1) *- new column route code and sorted into like JFK-ORD*

airports = airports[airports['TYPE'].isin(['medium_airport', 'large_airport'])] *- this will filter the Airport type to medium or large airport only*

airports = airports.dropna(subset=['IATA_CODE']) *- this will remove not applicable values*

airports['IATA_CODE'] = airports['IATA_CODE'].str.strip() *- this will clean spaces*

clean_flights = clean_flights[clean_flights['ORIGIN'].isin(airports['IATA_CODE']) & clean_flights['DESTINATION'].isin(airports['IATA_CODE'])] *- Check the route code to match from Flights data and Airpot data*

### -Get the 10 busiest round-trip routes

route_counts = clean_flights['route_code'].value_counts().reset_index() *- Count how many flights per route*

route_counts.columns = ['route_code', 'num_flights'] *- create a column for route count*

route_counts['num_round_trips'] = route_counts['num_flights'] // 2 *- create new column for route count converted to round trips*

ten_busiest = route_counts.sort_values(by='num_round_trips', ascending=False).head(10) *- Sort from the highest number of round trips and get top 10*

ten_busiest.to_csv("10 Busiest Roundtrip Routes/10_busiest_roundtrip_routes.csv", index=False) *- create a file for 10 busiest round trip route use for external visual like tableau*

### -Create a bar chart for the 10 busiest round-tip routes using motplot.pyplot

plt.figure(figsize=(12, 6)) *- Set figure size*

plt.barh(ten_busiest['route_code'], ten_busiest['num_round_trips'], color='skyblue', edgecolor='black') *- input the values*

plt.title('The 10 Busiest Round Trip Routes for Q1 2019', fontsize=16, fontweight='bold') *- Title*

plt.xlabel('Number of Round Trips', fontsize=12) *- x label*

plt.ylabel('Route Code', fontsize=12) *- y label*

plt.gca().invert_yaxis() *- invert position*

plt.grid(axis='y', linestyle='--', alpha=0.7) *- Grid*

for i, value in enumerate(ten_busiest['num_round_trips']):

plt.text(value, i, f"{value:,.0f}", va='center') - *display exact number of roundtrips*

plt.tight_layout() - *tighten layout*

plt.savefig("10 Busiest Roundtrip Routes/10_busiest_roundtrip_routes.png") - *save and create a png file for 10 busiest roundtrip routes for visual*

### -Get the 10 most profitable round-trip routes

clean_flights['passengers'] = clean_flights['OCCUPANCY_RATE'] * passenger_cap - *calculate passenger*

clean_flights['ticket_revenue'] = clean_flights['passengers'] * ticket_price - *calculate the revenue*

clean_flights['baggage_revenue'] = clean_flights['passengers'] * 0.5 * baggage_fee - *calculate the baggage revenue*

clean_flights['total_revenue'] = clean_flights['ticket_revenue'] + clean_flights['baggage_revenue'] - *total revenue*

clean_flights['DISTANCE'] = pd.to_numeric(clean_flights['DISTANCE'], errors='coerce') - *convert to numeric*

clean_flights['fuel_cost'] = clean_flights['DISTANCE'] * 2 * fomc_cost_per_mile - *fuel cost per fllight*

clean_flights['other_cost'] = clean_flights['DISTANCE'] * 2 * dio_cost_per_mile - *other cost per flight*

clean_flights['ARR_DELAY'] = pd.to_numeric(clean_flights['ARR_DELAY'], errors='coerce') - *clean the arrival delay*

clean_flights['ARR_DELAY'].fillna(0, inplace=True) - *fill arrival delay*

clean_flights['dep_delay_cost'] = clean_flights['DEP_DELAY'].apply(lambda x: max(0, x - delay_minutes) * delay_per_minute_cost) - *calculate departure delay penalty*

clean_flights['arr_delay_cost'] = clean_flights['ARR_DELAY'].apply(lambda x: max(0, x - delay_minutes) * delay_per_minute_cost) - *calculate arrival delay penalty*

clean_flights['delay_cost'] = clean_flights['dep_delay_cost'] + clean_flights['arr_delay_cost'] - *calculate the delay cost*

origin_size_map = airports.set_index('IATA_CODE')['TYPE'].to_dict() - *change index to IATA_CODE and change TYPE into series*

def get_airport_fee(code): - *convert the airport size into airport cost*

size = origin_size_map.get(code, '')

if size == 'large_airport':

return large_airport_cost

```
    elif size == 'medium_airport':
        return medium_airport_cost
    else:
    return 0
clean_flights['origin_fee'] = clean_flights['ORIGIN'].apply(get_airport_fee) - create origin fee for the airport fee
clean_flights['dest_fee'] = clean_flights['DESTINATION'].apply(get_airport_fee) - create destination fee for the airport fee
clean_flights['airport_fees'] = clean_flights['origin_fee'] + clean_flights['dest_fee'] - total airport fee
clean_flights['total_cost'] = clean_flights['fuel_cost'] + clean_flights['other_cost'] + clean_flights['delay_cost'] + clean_flights['airport_fees'] - calculate the total cost
clean_flights['profit'] = clean_flights['total_revenue'] - clean_flights['total_cost'] - calculate the profit
```

### -Group by route code and get total profit, revenue, cost and flight count

```
profit_summary = clean_flights.groupby('route_code').agg(
    total_revenue=('total_revenue', 'sum'),
    total_cost=('total_cost', 'sum'),
    total_profit=('profit', 'sum'),
    num_flights=('route_code', 'count')
).reset_index()
profit_summary['round_trips'] = profit_summary['num_flights'] // 2 - convert flights to round trips
ten_profitable = profit_summary.sort_values(by='total_profit', ascending=False).head(10) - the top 10 profitable roundtrip routes
ten_profitable.to_csv("10 Most Profitable Roundtrip Routes/10_most_profitable_roundtrip_routes.csv", index=False) - create a file for 10 most profitable round trip route use for external visual like tableau
```

### -Create bar chart for top 10 most profitable round-trip routes using motplot.pyplot

```
plt.figure(figsize=(12, 6))  - figure size
plt.barh(ten_profitable['route_code'], ten_profitable['total_profit'], color='skyblue', edgecolor='black') - input the values
```

plt.title('The 10 Most Profitable Round Trip Routes for Q1 2019', fontsize=16, fontweight='bold') - *title*

plt.xlabel('Total Profit $', fontsize=12) - *x label*

plt.ylabel('Route Code', fontsize=12) - *y label*

plt.gca().invert_yaxis() - *invert position*

formatter = FuncFormatter(lambda x, pos: f'${x:,.0f}') - *formatt the value of x to exact $*

plt.gca().xaxis.set_major_formatter(formatter)

plt.grid(axis='y', linestyle='--', alpha=0.7) - *grid*

```
for i, value in enumerate(ten_profitable['total_profit']):
    plt.text(value, i, f"${value:,.0f}", va='center') - display exact total profits
```

plt.tight_layout() - *tighten layout*

plt.savefig("10 Most Profitable Roundtrip Routes/10_most_profitable_roundtrip_routes.png") - *save and create a png file for 10 most profitable roundtrip routes for visual*

### -Get the to 5 recommended round-trip routes

ten_profitable_sorted = ten_profitable.sort_values(by='total_profit', ascending=False) - *sort the 10 profitable routes*

ten_profitable_sorted['avg_dep_delay'] = clean_flights.groupby('route_code')['DEP_DELAY'].transform('mean') - *get the average departure delay*

ten_profitable_sorted = ten_profitable_sorted[ten_profitable_sorted['avg_dep_delay'] <= 15] - *assumed average departure delay 15 minutes*

top_5_recommended = ten_profitable_sorted.head(5) - *get the top 5 recommended routes based on profit and delays*

top_5_recommended.to_csv("5 Recommended Roundtrip Routes/5_recommended_roundtrip_routes.csv", index=False) - *create a file for 5 recommended round trip route use for external visual like tableau*

### -Create a bar chart for 5 recommended round-trip routes using motplot.pyplot

fig, ax1 = plt.subplots(figsize=(12, 6)) - *Create figure and axis*

ax1.barh(top_5_recommended['route_code'], top_5_recommended['total_profit'], color='skyblue', edgecolor='black') - *input the values*

ax1.set_xlabel('Total Profit (USD)') - *x label*

ax1.set_ylabel('Route Code') - *y label*

ax1.set_title('The 5 Recommended Round Trip Routes for Q1 2019') - *title*

plt.gca().invert_yaxis() - *invert position*

formatter = FuncFormatter(lambda x, pos: f'${x:,.0f}') - *formatt the value of x to exact $*

plt.gca().xaxis.set_major_formatter(formatter)

plt.grid(axis='y', linestyle='--', alpha=0.7) - *grid*

for i, value in enumerate(top_5_recommended['total_profit']):

    plt.text(value, i, f"${value:,.0f}", va='center') - *display exact total profits*

ax2 = ax1.twiny() - *create second axis for average departure delay*

ax2.plot(top_5_recommended['avg_dep_delay'], top_5_recommended['route_code'], color='red', marker='o', linestyle='dashed') - *input the values*

ax2.set_xlabel('Average Departure Delay (Minutes)', color='red') - *x label*

for i, value in enumerate(top_5_recommended['avg_dep_delay']):

    plt.text(value, i, f"{value:,.0f}", va='bottom') - *display the exact average delay*

plt.tight_layout() - *tighten layout*

plt.savefig("5 Recommended Roundtrip Routes/5_recommended_roundtrip_routes.png") - *save and create a png file for 5 recommended roundtrip routes for visual*

### -Get the 5 recommended breakeven round-trip routes

ten_profitable_sorted['profit_per_flight'] = ten_profitable_sorted['total_profit'] / ten_profitable_sorted['num_flights'] - *profit per flight is already available in the 'profit' column*

ten_profitable_sorted['breakeven_flights'] = ten_profitable_sorted['total_cost'] / ten_profitable_sorted['profit_per_flight'] - *calculate breakeven flights*

top_5_breakeven = ten_profitable_sorted[['route_code', 'breakeven_flights']].sort_values(by='breakeven_flights').head(5) - *sort to get the top 5 routes with the least breakeven flights*

top_5_breakeven.to_csv("5 Recommended Breakeven Roundtrip Routes/5_recommended_breakeven_roundtrip_routes.csv", index=False) - *create a file for 5 recommended breakeven round trip route use for external visual*

### -Create a bar chart for the 5 recommended breakeven round-trip routes

plt.figure(figsize=(10, 6)) - *figure size*

plt.barh(top_5_breakeven['route_code'], top_5_breakeven['breakeven_flights'], color='skyblue', edgecolor='black') - *input the values*

plt.ylabel('Route Code') - *y label*

plt.xlabel('Breakeven Flights') *- x label*

plt.title('Top 5 Routes with Least Breakeven Flights for Q1 2019') *- title*

plt.gca().invert_yaxis() *- invert position*

for i, value in enumerate(top_5_breakeven['breakeven_flights']):

plt.text(value, i, f"{value:,.0f}", va='center') *- display exact breakeven flights*

plt.tight_layout() *- tighten layout*

plt.savefig("5 Recommended Breakeven Roundtrip Routes/5_recommended_breakeven_roundtrip_routes.png") *- save and create a png file for 5 recommended breakeven roundtrip routes for visual*

## **KPI's** (KPIs.pbix)

**-On time performance**

OGG to HNL – 7.2 minutes average departure delay

HNL to KOA – 7.5 minutes average departure delay

HNL to LIH – 8.2 minutes average departure delay

KOA to HNL – 9.5 minutes average departure delay

LIH to HNL – 9.8 minutes average departure delay

**-Revenue**

HNL to OGG – 58 million total revenues

OGG to HNL - 57 million total revenues

HNL to LIH - 38 million total revenues

LIH to HNL - 38 million total revenues

KOA to HNL - 36 million total revenues

**-Profit**

HNL to OGG – 16.8 million total profits

OGG to HNL – 16.5 million total profits

HNL to LIH – 11.4 million total profits

LIH to HNL – 11.2 million total profits

# Cap One Airline

KOA to HNL – 8.8 million total profits

## Recommendation

**Recommended Round Trip Routes for CAP ONE AIRLINE (On time, for you)**

- **OGG to HNL -** *Kahului Airport to Daniel K Inouye International Airport*
- **HNL to KOA -**
  *Daniel K Inouye International Airport to Ellison Onizuka Kona International At Keahole A irport*
- **HNL to LIH -** *Daniel K Inouye International Airport to Lihue Airport*
- **KOA to HNL -**
  *Ellison Onizuka Kona International At Keahole Airport to Daniel K Inouye International*
- **LIH to HNL -** *Lihue Airport to Keahole Airport to Daniel K Inouye International*