1. Problem Statement:

An airline plans to launch five new round-trip routes between medium and large U.S. airports, each operated by a $90M aircraft. With a brand focus on punctuality (“On time, for you”), the goal is to use 1Q2019 data to identify the most profitable and operationally efficient routes that align with their on-time performance strategy.

The main objectives include:

1. Identifying the 10 busiest round-trip routes by the number of round-trip flights in the quarter (excluding canceled flights).
2. Determining the 10 most profitable round-trip routes by calculating total revenue and total cost per route (excluding canceled flights).
3. Recommending the 5 optimal round-trip routes to invest in, based on profitability, operational factors, and business alignment.
4. Calculating the breakeven point, in terms of the number of round-trip flights required to recover the $90 million upfront investment per airplane.
5. Recommending Key Performance Indicators (KPIs) to track the ongoing success and operational health of the selected routes.
6. Dataset Overview:

* Flights dataset: Contains data about available routes from origin to destination. For occupancy, use the data provided in this dataset.
* Tickets dataset: Ticket prices data (sample data only as the data is huge). Consider only round trips in your analysis.
* Airport Codes dataset: Identifies whether an airport is considered medium or large sized. Consider only medium and large airports in your analysis.

1. Assumptions made for analysis:
   1. Each airplane is dedicated to a single round trip route (e.g., JFK -> ORD -> JFK).
   2. The upfront aircraft acquisition cost is $90 million per airplane.
   3. Airport fees:
      1. $5,000 for each medium airport landing
      2. $10,000 for each large airport landing  
         (Charged on both legs of the round trip = total of 2 charges per round trip flight).
   4. Fuel, Oil, Maintenance, and Crew costs = $8 per mile per leg.
   5. Depreciation, Insurance, and other costs = $1.18 per mile per leg.
   6. Delay costs:
      1. First 15 minutes are free for both departure and arrival.
      2. Beyond 15 minutes, it’s $75 per minute for both departure and arrival delays.
   7. Each airplane has a seating capacity of 200 passengers.
   8. Occupancy Rate:  
      Use OCCUPANCY\_RATE from the Flights dataset to estimate actual passenger counts (ignore passenger counts from Tickets dataset for this purpose).
   9. Baggage Fees:
      1. 50% of passengers check one bag per leg.
      2. Fee = $35 per bag per leg ($70 total per passenger round trip).
   10. Ticket Revenue:  
       The Tickets dataset provides round trip fare per passenger (ITIN\_FARE). Assume this fare applies uniformly across all flights for a given route and quarter.
   11. Filtered out only medium and large airports for analysis (based on the problem statement.)
2. Data Quality Insights / Issues:

For Airport Code Dataset:

* Overall missing: ~87,000 cells missing, ~19.6% of the entire dataset.
* Duplicates: 31 exact duplicate rows (0.1%).
* IATA\_CODE: Over 83% missing (~46k rows). Way too many blanks for a key field.
* CONTINENT: 50.3% missing. Half the rows don’t have continent info.
* MUNICIPALITY: 10.3% missing (~5.7k rows).
* ELEVATION\_FT: 12.7% missing (~7k rows). Also has 48 negative values, min is -1266.
* ISO\_COUNTRY: 0.4% missing (247 rows). Not much, but still incomplete.
* TYPE column’s fine structurally — no missing values. Mostly small\_airport (34k), then heliport (11k). Only 4,532 are medium\_airport, which is the main focus here.
* NAME field is clean — no missing values, and 52k+ unique names. Covers ~95% of rows, a few duplicates might exist but nothing major.
* ELEVATION\_FT has weird outliers — lowest is -1266 ft, highest is 22,000 ft. That top end looks sketchy, probably data entry or unit issues.
* That -1266 min value in ELEVATION\_FT is also unrealistic for an airport — definitely worth flagging as invalid.
* After filtering for US-based medium/large airports, only 858 rows remain out of 55k+. So most airports are either small or non-US.
* Even in the filtered set, a lot of IATA\_CODEs are still missing — unexpected for medium/large US airports. Could be a data completeness issue or mislabeling.
* Used median for numeric imputation and mode/constant for categorical — makes sense given the outliers in elevation.
* CONTINENT column got dropped since it was completely null. Nothing else crossed the 99% null mark, so rest were kept.
* After filtering for U.S. medium and large airports, ended up with 858 rows and 7 columns — a much more focused slice.
* No fully duplicated rows found — so at least it’s clean in that regard.
* Remaining nulls are where you'd expect:  
  • ELEVATION\_FT is numeric  
  • MUNICIPALITY and IATA\_CODE are categorical

Column-wise uniqueness analysis shows:

* ISO\_COUNTRY is just "US" for all rows now — doesn’t add value anymore, could be dropped or treated as constant.
* ELEVATION\_FT has 651 unique values, COORDINATES has 858 — both are high-cardinality fields.
* NAME, MUNICIPALITY, and IATA\_CODE also have solid variety — 857, 716, and 821 unique values respectively.

The abnormality report flags the following:

* ELEVATION\_FT has both negative values and extreme outliers — could be legit (like below-sea-level airports) or just bad entries.
* NAME and COORDINATES have some weird or rare patterns — probably formatting issues or noise.
* ELEVATION\_FT was cleaned by replacing bad values with the mean (1170.56). No nulls left after that.
* Outlier check using IQR showed a bunch of high-end outliers. KDE plot backs it — long-tailed distribution with some extreme elevations.
  + ELEVATION\_FT shows considerable deviation in its distribution:
  + It has a high **skewness of 2.0**, indicating a right-skewed distribution where a few values are significantly larger than the rest.
  + The **kurtosis is 3.44**, suggesting a sharper peak with heavier tails than a normal distribution.
  + There are **106 outliers** detected using the IQR method, which reinforces the presence of extreme values in elevation data.
  + **Outlier detection using IQR method**:  
    Outlier detection flagged 106 rows in ELEVATION\_FT using the 1.5×IQR rule.
  + The column is right-skewed (skewness = 2.0) with moderate kurtosis (3.44), so it's got a few big elevation values pulling the tail.

For Flights Dataset:

* The dataset has 1,915,886 rows and 16 columns — large with good flight coverage.
* Overall missingness is ~0.6% (≈176k cells), mostly in ARR\_DELAY, AIR\_TIME, DISTANCE, and OCCUPANCY\_RATE.
* Found 558 fully duplicated rows before filtering; 4,900 exact duplicates were dropped during cleanup (~0.26%).
* Columns like OP\_CARRIER\_FL\_NUM, AIR\_TIME, and DISTANCE are typed as object instead of numeric — due to mixed formatting or rogue strings.
* DEP\_DELAY (~2.6%) and ARR\_DELAY (~2.9%) have moderate missingness; AIR\_TIME missing in ~3% of rows.
* FL\_DATE, OP\_CARRIER, and ORIGIN\_AIRPORT\_ID are fully clean, with no nulls or formatting issues.
* Zero-delay flags: DEP\_DELAY has ~91k zeros (~4.8%), ARR\_DELAY has ~35k (~1.8%) — not necessarily wrong, but need context.
* DEP\_DELAY and ARR\_DELAY have a high % of negative values (early departures/arrivals); DEP\_DELAY goes down to -63 mins, and ARR\_DELAY to -94 mins.
* DEP\_DELAY maxes out at 2,941 mins, ARR\_DELAY at 2,923 mins — likely outliers needing capping or flagging.
* OP\_CARRIER\_FL\_NUM is object-typed but holds mostly numeric values — flagged for type conversion.
* AIR\_TIME ranges from 56 to 16,985 minutes — upper values are clear outliers.
* DISTANCE spans from 337 to 13,490 miles — mostly reasonable, but top-end should be checked.
* OCCUPANCY\_RATE ranges from 0.3 to 1.0, mean ≈ 0.65. Only 310 missing values, skewed slightly right.
* FL\_DATE has 90 unique entries (Jan–Mar 2019); clean and correctly typed as datetime.
* ORIGIN, DESTINATION, ORIGIN\_CITY\_NAME, and DEST\_CITY\_NAME are 100% complete and well-distributed (361 airport codes, 355 cities).
* ORIGIN\_AIRPORT\_ID and DEST\_AIRPORT\_ID are numeric, complete, and fall in a valid range (10,135 to 16,218).
* Correlation heatmap shows strong link between ARR\_DELAY and DEP\_DELAY (r = 0.649); OCCUPANCY\_RATE is uncorrelated with delays or airport IDs.
* Missing value heatmap revealed that DEP\_DELAY, ARR\_DELAY, and AIR\_TIME often go missing together.
* 10 out of 16 columns are object-typed — includes fields like TAIL\_NUM, OP\_CARRIER\_FL\_NUM, and DISTANCE, which ideally should be numeric.
* Abnormality checks flagged:
  + FL\_DATE: mixed format in some rows.
  + TAIL\_NUM, OP\_CARRIER\_FL\_NUM, AIR\_TIME, DISTANCE: non-numeric values due to text contamination.
* Imputation strategy:
  + ARR\_DELAY negatives replaced with mean of valid values.
  + Numeric fields imputed using mean, object-like fields (AIR\_TIME, DISTANCE) filled with 'Unknown'.
* Post-imputation check confirmed 0 missing values remaining.
  + No columns were dropped due to high nulls — all had enough data to retain.
  + Dataset is now clean and ready for modeling or visualization.

For Tickets dataset:

* Dataset has 1.17M rows and 12 columns — smaller than Flights but still good for ticket-level analysis.
* Only 2,937 missing cells (<0.1%) — not a big issue.
* 71,880 duplicate rows (~6.2%) — that’s high and should probably be dropped.
* ITIN\_FARE is object type but holds numeric data — need to be converted to float.
* ITIN\_ID and REPORTING\_CARRIER are highly correlated.
* ORIGIN and DESTINATION have 419 unique codes — enough variety for route-level analysis.
* ITIN\_ID is ~94% unique — reused IDs suggest roundtrip or multi-leg bookings.
* ORIGIN\_STATE\_ABR and ORIGIN\_STATE\_NM both have 52 clean, consistent entries — no issues there.
* ROUNDTRIP is a clean binary column with no nulls — about 60.7% of tickets are roundtrip.
* REPORTING\_CARRIER has 21 distinct airlines, no missing values — overlaps heavily with ITIN\_ID.
* PASSENGERS is clean but skewed — ~0.2% missing, most values are small but some large group bookings exist.
* ITIN\_FARE is a text column holding numeric data — 960 missing entries, needs conversion to float.
* DESTINATION has 410 unique airport codes, no missing values — good for route-level analysis.
* Only PASSENGERS and ITIN\_FARE have missing values (<0.2%) — overall very complete dataset.
* ORIGIN\_STATE\_ABR and ORIGIN\_STATE\_NM are clean with 52 unique entries each — full US state coverage.
* REPORTING\_CARRIER has 21 carriers, no missing values — top 5 airlines dominate most of the entries.
* DESTINATION has 410 unique airports, no nulls — solid variety across U.S. locations.
* Only 3 columns have missing values — PASSENGERS, ITIN\_FARE, DESTINATION — and all are <0.2%.
* Duplicate check flagged repeated ITIN\_ID + location combos — could be valid multi-leg trips or actual duplicates.
* Dataset has 1.16M rows, and missingness is minimal across just a couple of columns — generally very clean.
* ROUNDTRIP is a clean binary field (0/1), with 60.7% round trips — no issues here.
* DESTINATION has 410 unique codes, fully populated — solid for route-level analysis.
* 71,880 duplicate rows (~6.2%) found, mostly with same ITIN\_ID, YEAR, and ORIGIN.
* After filtering for ROUNDTRIP == 1.0, all missing values were imputed — PASSENGERS (median), ITIN\_FARE (median).
* No columns dropped for high nulls - none crossed 99% threshold, so data coverage is strong.
* After dropping 13.4% duplicate rows, remaining nulls fully handled post-imputation.
* Abnormalities detected:
  + Extreme or dirty values in ITIN\_ID, PASSENGERS, ITIN\_FARE, and strange strings in ORIGIN fields and REPORTING\_CARRIER.
* Imputation Strategy:
  + ITIN\_FARE had non-numeric junk like '820$$$' and '$’ ‘100.00' — cleaned and filled using mean strategy.
  + Final dataset has no missing values.

1. Data Munging and Data Transformation:
   1. Reading and Selecting Relevant Columns from Cleaned Datasets**:** To streamline the merging process and improve memory efficiency, only the relevant columns from each dataset were selected for further analysis.

* **Airport Codes Dataset**:  
  Retained the essential fields: type, name, elevation\_ft, muncipality, iata\_code and coordinates. these columns were used primarily for filtering airport sizes and enriching route-level insights.
* **Flights Dataset**:  
  Selected key operational flight attributes including fl\_date, op\_carrier, tail\_num, op\_carrier\_fl\_num, origin and destination airport identifiers, delay metrics, cancellation flag, airtime, distance, and occupancy rate.
* **Tickets Dataset**:  
  Focused on itinerary-level information relevant for financial analysis: itin\_id, year, quarter, origin, destination, roundtrip, reporting\_carrier, passengers and itin\_fare.
  1. Extracting Temporal Features from Flight Date
     1. The fl\_date column was converted to datetime format with errors = ‘coerce’.
     2. After converting the date column, two new columns were added:
        1. Year: Extracted using .dt.year from the parsed fl\_date.
        2. Month: Extracted using .dt.month from the parsed fl\_date.
  2. Handling Round-Trip Route Logic: To aggregate flight-level data into route-level summaries, flights were grouped using a combination of op\_carrier, origin and destination.

The following aggregations were applied:

* Distance, occupancy\_rate, dep\_delay and arr\_delay: Mean values were calculated for each.
* Tail\_num: Used as a proxy for flight count and renamed to TOTAL\_FLIGHTS.

To normalize directionality and treat the routes (eg: ATL–ORD and ORD–ATL) as the same round trip, a new route\_key was created. This key was generated by alphabetically sorting and joining the origin and destination airport codes using a dash.

Finally, the dataset was regrouped using route\_key and op\_carrier, summing up total\_flights and recalculating the average metrics. This ensured each round-trip route is uniquely represented regardless of flight direction.

* 1. Merging Tickets and Flights Data

The tickets\_sample dataframe was grouped by ROUTE\_KEY and REPORTING\_CARRIER to compute:

* + 1. Total passengers on a given route (using sum)
    2. Average fare for each round-trip itinerary (using mean on ITIN\_FARE)

A ROUTE\_KEY was created for each row by alphabetically sorting and joining the ORIGIN and DESTINATION airport codes. This ensured consistency with the previously processed flights\_grouped dataframe.

The grouped tickets data was then merged with the flight-level aggregation using:

* + 1. ROUTE\_KEY
    2. OP\_CARRIER from the flights data matched to REPORTING\_CARRIER in the tickets data

This resulted in a single merged dataset containing route-level operational metrics along with passenger and fare data.

* 1. Enriching Final Dataset with Airport Information
     1. To attach metadata for each endpoint of a round-trip route:
        1. The ROUTE\_KEY string was split into two components: AIRPORT\_A and AIRPORT\_B
        2. These were used to join with the airport codes dataset twice:
           1. First on AIRPORT\_A, renamed as IATA\_CODE
           2. Then on AIRPORT\_B, similarly renamed

This allowed airport attributes like TYPE, MUNICIPALITY, ELEVATION\_FT, and COORDINATES to be added for both origin and destination airports. The enriched dataset supports further geographic or categorical analysis of route performance.

* 1. Final Field Renaming for Clarity

After merging flight, ticket, and airport datasets, column names in the combined dataframe were renamed to improve clarity and readability. The renaming focused on:

* + 1. Making field names self-explanatory (e.g., DISTANCE- ONE\_WAY\_DISTANCE\_MILES)
    2. Aligning terminology across datasets (e.g., OP\_CARRIER and REPORTING\_CARRIER both unified as CARRIER)
    3. Labeling airport-level attributes clearly for both endpoints of a route (e.g., AIRPORT\_A\_NAME, AIRPORT\_B\_ELEVATION\_FT, etc.)

1. Metadata: Several new columns were created to estimate passenger volume, calculate revenue streams, compute total operational costs, and derive profit for each route. All calculations were based on business assumptions and route-level summaries.

|  |  |
| --- | --- |
| ESTIMATED\_PASSENGERS | Estimated total passengers = AVG\_OCCUPANCY\_RATE × 200 × TOTAL\_FLIGHTS |
| TICKET\_REVENUE | Revenue from tickets = ESTIMATED\_PASSENGERS × AVG\_ROUNDTRIP\_FARE\_PER\_PASSENGER |
| BAGGAGE\_REVENUE | Revenue from baggage fees = ESTIMATED\_PASSENGERS × 0.5 × $70 |
| TOTAL\_REVENUE | Sum of TICKET\_REVENUE and BAGGAGE\_REVENUE |
| ROUND\_TRIP\_DISTANCE\_MILES | Total round-trip distance = ONE\_WAY\_DISTANCE\_MILES × 2 |
| MILEAGE\_COST | Mileage cost = ROUND\_TRIP\_DISTANCE × TOTAL\_FLIGHTS × $9.18 |
| AIRPORT\_FEES | Total airport fees based on sizes of AIRPORT\_A and AIRPORT\_B |
| DELAY\_COST | Total cost of delays beyond 15 minutes, for both departures and arrivals |
| TOTAL\_COST | Combined cost from mileage, airport fees, and delays |
| PROFIT | Net profit = TOTAL\_REVENUE - TOTAL\_COST |
| PROFIT\_MARGIN\_PCT | Profit as % of total revenue |
| COST\_PER\_PASSENGER | Cost per estimated passenger |
| REVENUE\_PER\_PASSENGER | Revenue per estimated passenger |
| PROFIT\_PER\_PASSENGER | Profit per estimated passenger |
| FLIGHTS\_PER\_1000\_MILES | Total flights per 1,000 miles of route |
| MILEAGE\_COST\_PCT | % of total cost due to mileage |
| AIRPORT\_FEES\_PCT | % of total cost due to airport landing fees |
| DELAY\_COST\_PCT | % of total cost caused by delays |
| PASSENGERS\_PER\_FLIGHT | Average number of passengers per flight |
| REVENUE\_PER\_FLIGHT | Revenue per flight |
| PROFIT\_PER\_FLIGHT | Profit per flight |
| COST\_PER\_FLIGHT | Cost per flight |
| DELAY\_MINUTES\_PER\_FLIGHT | Sum of average departure and arrival delay per flight |
| BAGGAGE\_REVENUE\_PCT | Baggage revenue as % of total revenue |
| TOTAL\_DELAY\_MINUTES | Total average delay per flight (departure + arrival) |
| AVG\_TOTAL\_DELAY\_MINUTES | AVG\_DEP\_DELAY\_MINUTES + AVG\_ARR\_DELAY\_MINUTES |
| REV\_TO\_COST\_RATIO | TOTAL\_REVENUE / TOTAL\_COST |

* 1. Final Recommendation:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Route No. | Origin Airport | Destination Airport | Profit Margin | Delay Performance | Total Passengers | Breakeven % Occupancy |
| 1 | Honolulu, HI (HNL) | Kahului, HI (OGG) | Extremely High | Excellent (low delay cost) | 620,949+ | ~43% |
| 2 | Honolulu, HI (HNL) | Kona, HI (KOA) | Very High | Excellent | 391,000+ | ~47% |
| 3 | Honolulu, HI (HNL) | Lihue, HI (LIH) | Very High | Excellent | 410,000+ | ~49% |
| 4 | Charlotte, NC (CLT) | Wilmington, NC (ILM) | High | Good | 110,000+ | ~50% |
| 5 | Dallas/Fort Worth, TX (DFW) | Fayetteville, AR (XNA) | High | Very Good | 120,000+ | ~52% |

* 1. What’s next?

Honestly, 8 hours flew by. I prioritized building a clean, modular pipeline that delivers reliable KPIs—but if I had more time, here’s what I’d do next:

* **Add Temporal Granularity**: Right now, route performance is aggregated. I'd break it down by month or quarter to detect seasonality, demand swings, or delay patterns that vary over time.
* **ML-Based Profitability Forecasts**: Use regression models to predict profitability of routes under different fare or capacity assumptions—basically simulate “what-if” scenarios for new routes or pricing strategies.
* **Risk Metrics**: Not all profitable routes are stable. I’d add metrics like variance in delay, standard deviation of occupancy, or volatility in ticket fares to better assess route consistency.
* **Geo Visualizations**: Map-based visuals would be a great way to show route density, profitability hot zones, or underutilized large airports.
* **Benchmarking**: Compare carriers on similar routes (e.g., Delta vs. United on ATL-JFK) to identify performance leaders and laggards.
* **API Wrapper for Live Updates**: Package the final logic into a function that accepts new monthly datasets and returns updated route rankings automatically.
* **Route Expansion Model**: I’d build a model that suggests where a new airline should launch next based on unmet demand, airport congestion, and competitor presence.