PROJECT PHASE 1

AVIATION ACCIDENT DATA INTEGRATION

Ciências ULisboa

Information Integration and Analytic Data Processing

17-03-2025

Group 3:

- Tommaso Tragno [fc64699]
- Manuel Cardoso [fc56274]
- Chen Cheng [fc64872]
- Cristian Tedesco [fc65149]

Agenda

- Project Motivation & Questions
- Data Sources & Rationale
- Data Profiling & Cleaning
- Integrated Schema
- Identity Resolution & Blocking
- Preliminary Insights & Challenges
- Conclusion & Next Steps

1. Project Motivation & Questions

Explore links between accidents, passenger volume, weather, and aircraft models.

Assess if higher traffic numbers lead to more accidents.
Examine if bad weather increases the likelihood of accidents.
Identify if certain regions or airports have unique accident patterns.
Determine if the era or model of aircraft is linked to higher accident rates.

Goal of this Project

Provide actionable insights for aviation stakeholders—airlines, airports, and regulators—seeking data-driven strategies to enhance safety protocols and resource allocation under conditions of fluctuating demand and variable weather.

2. Data Sources & Rationale

2.1 NTSB Aviation Accident Database (Filtered)

- Contains detailed records of accidents, including location, date/time, flight phase, and possible contributing factor

2.2 Open-Meteo API

- Returns hourly or daily weather data (temperature, precipitation, wind speed, wind direction, etc.) for specified coordinates and date

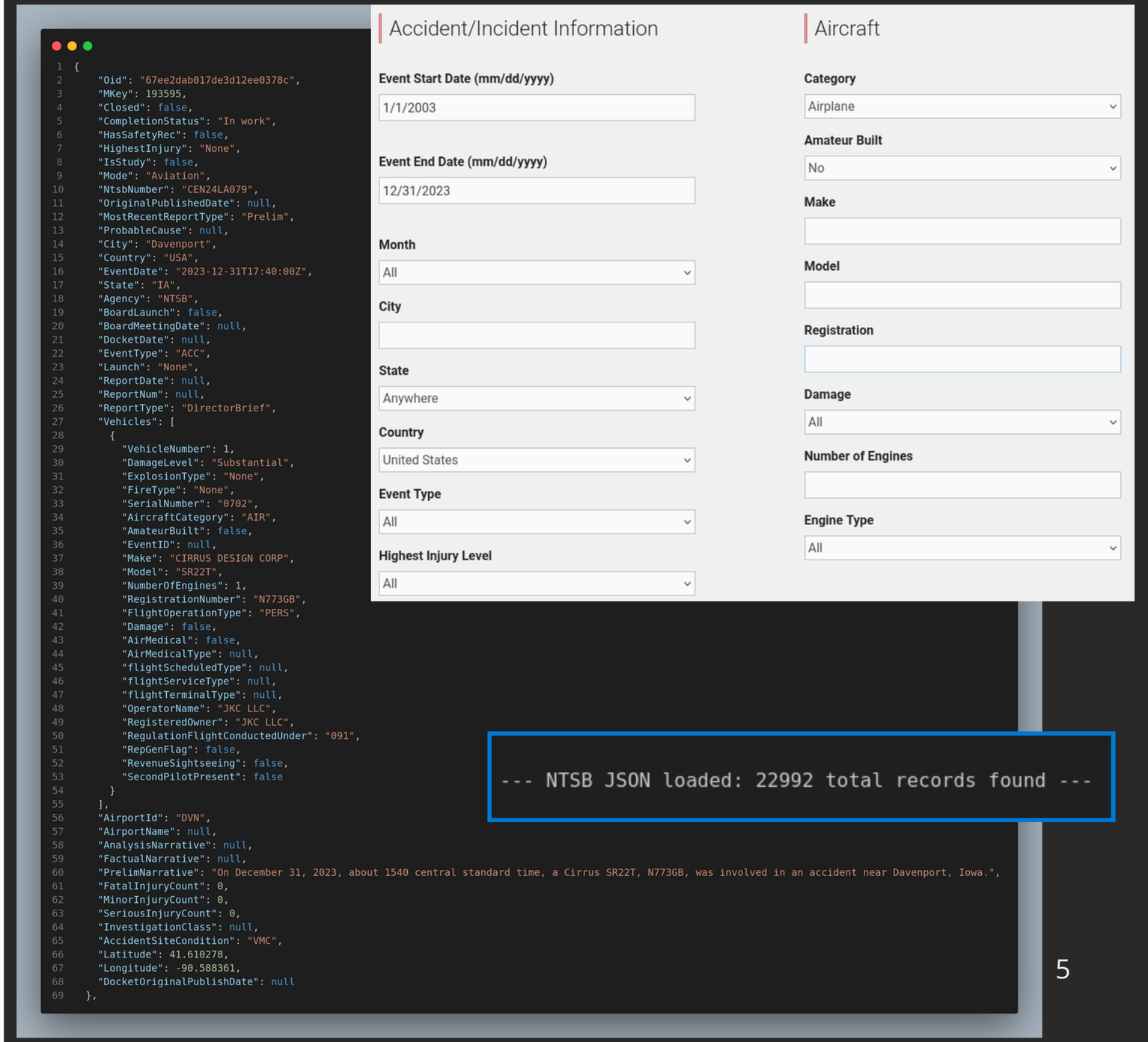
2.3 U.S. Airline Traffic Data (Kaggle)

- Provides monthly flight volumes, passenger counts, and other tourism-related indicators for both domestic and international flights

2.4 Aircraft Production Data (Kaggle)

- Contains details on various aircraft manufacturers, models, and production volume

2.1 NTSB Aviation Accident Database



2.2 Open-Meteo API

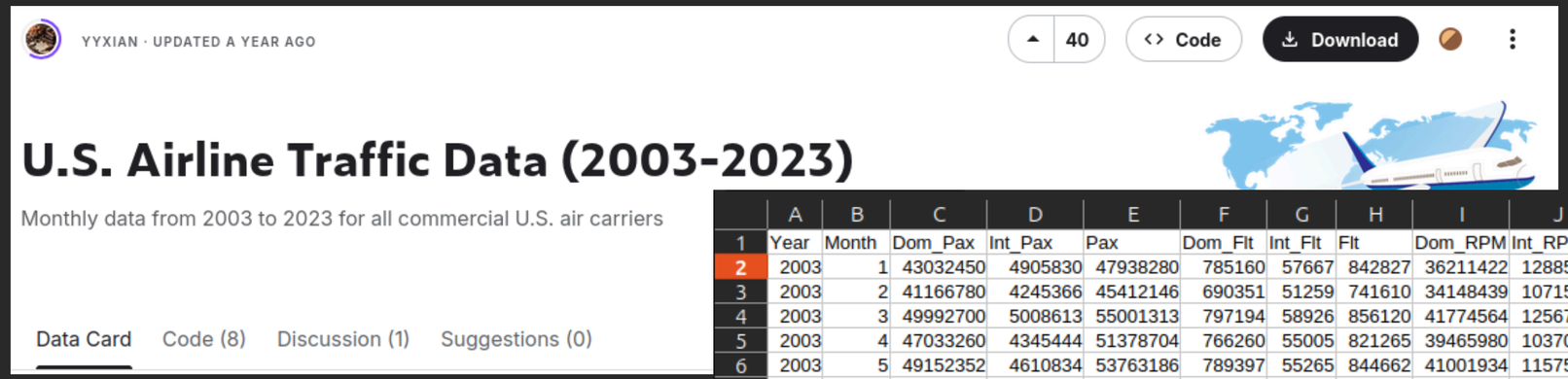
```
--- Weather DataFrame sample ---
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 491592 entries, 0 to 491591
Data columns (total 21 columns):
```

```python
def fetch_weather_data(lat, lon, date_str, cache_key):

    endpoint = "https://archive-api.open-meteo.com/v1/archive"
    params = {
        "latitude": lat,
        "longitude": lon,
        "start_date": date_str,
        "end_date": date_str,
        "hourly": ",".join([
            "temperature_2m","relative_humidity_2m","dew_point_2m","pressure_msl",
            "surface_pressure","precipitation","rain","snowfall","cloud_cover",
            "cloud_cover_low","cloud_cover_mid","cloud_cover_high","wind_speed_10m",
            "wind_speed_100m","wind_direction_10m","wind_direction_100m",
            "wind_gusts_10m","weather_code","snow_depth"
        ]),
        "timezone": "GMT"
    }

    timeout_attempts = 0

    while timeout_attempts < MAX_RETRIES_TIMEOUT:
        # We'll also keep track of 429 attempts
        status429_attempts = 0
        while True:
            try:
                # Perform the request with a timeout
                response = requests.get(endpoint, params=params, timeout=REQUEST_TIMEOUT)
                # Check HTTP status
                if response.status_code == 200:
                    return response.json().get("hourly", {})
                elif response.status_code == 429:
                    status429_attempts += 1
                    if status429_attempts <= RETRIES_ON_429:
                        print(f"[{cache_key}] Got 429. Waiting {SLEEP_ON_429_SECS}s then retrying (attempt {status429_attempts}/{RETRIES_ON_429}).")
                        time.sleep(SLEEP_ON_429_SECS)
                        continue  # Retry the same request
                    else:
                        print(f"[{cache_key}] Too many 429 responses. Giving up this accident.")
                        return {}
                else:
                    # Some other error status code
                    print(f"[{cache_key}] Request failed: {response.status_code}. Skipping.")
                    return {}
            except requests.exceptions.Timeout:
```

```
ts}/{MAX_RETRIES_TIMEOUT}. Retrying.")

imes. Giving up.")
c in the main loop
```


```
"cen24la079_2023-12-31_41.610278_-90.588361": {"time": ["2023-12-31T00:00", "2023-12-31T01:00", "2023-12-31T02:00", "2023-12-31T03:00", "2023-12-31T04:00", "2023-12-31T05:00", "2023-12
-31T06:00", "2023-12-31T07:00", "2023-12-31T08:00", "2023-12-31T09:00", "2023-12-31T10:00", "2023-12-31T11:00", "2023-12-31T12:00", "2023-12-31T13:00", "2023-12-31T14:00", "2023-12-31T1
5:00", "2023-12-31T16:00", "2023-12-31T17:00", "2023-12-31T18:00", "2023-12-31T19:00", "2023-12-31T20:00", "2023-12-31T21:00", "2023-12-31T22:00", "2023-12-31T23:00"], "temperature_2m":
 [1.1, -0.6, -1.0, -0.4, -1.3, -1.8, -2.2, -2.7, -2.7, -2.9, -3.2, -3.1, -2.9, -2.6, -2.4, -1.6, -0.7, -0.1, 0.6, 1.2, 1.0, 1.1, 0.8, 0.1], "relative_humidity_2m": [85, 89, 89, 80, 71,
69, 70, 74, 73, 71, 72, 76, 78, 77, 77, 74, 70, 73, 74, 73, 72, 68, 67, 70], "dew_point_2m": [-1.1, -2.2, -2.6, -3.5, -5.8, -6.7, -6.8, -6.7, -6.9, -7.4, -7.5, -6.8, -6.2, -6.1, -5.8, -
5.7, -5.6, -4.4, -3.4, -3.1, -3.5, -4.2, -4.5, -4.8], "pressure_msl": [1011.4, 1012.5, 1012.8, 1013.3, 1013.5, 1013.9, 1014.2, 1014.0, 1014.1, 1014.6, 1014.6, 1014.8, 1015.1, 1015.8, 10
16.1, 1016.6, 1017.3, 1017.7, 1017.8, 1018.2, 1018.8, 1019.7, 1020.8, 1021.8], "surface_pressure": [983.2, 984.0, 984.3, 984.8, 984.9, 985.3, 985.5, 985.3, 985.4, 985.9, 985.8, 986.0, 9
86.3, 987.1, 987.4, 987.9, 988.7, 989.2, 989.3, 989.8, 990.3, 991.2, 992.3, 993.2], "precipitation": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
, 0.3, 0.1, 0.1, 0.0, 0.0, 0.0, 0.0, 0.0], "rain": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0], "snowfall": [0.0,
 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.21, 0.07, 0.07, 0.0, 0.0, 0.0, 0.0], "cloud_cover": [100, 87, 100, 100, 100, 100, 100, 100, 100, 100,
100, 100, 100, 100, 100, 99, 100, 100, 100, 100, 100, 100, 99, 100], "cloud_cover_low": [0, 46, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 98, 96, 100, 100, 97, 100, 10
0, 92, 98, 100], "cloud_cover_mid": [88, 0, 0, 0, 0, 0, 0, 25, 12, 37, 45, 98, 100, 100, 96, 85, 100, 100, 100, 100, 100, 100, 42, 0], "cloud_cover_high": [100, 76, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 50, 99, 18, 0, 0, 0, 0, 0, 0], "wind_speed_10m": [12.0, 11.2, 13.2, 16.6, 17.4, 16.7, 15.8, 16.7, 17.8, 18.1, 18.5, 17.7, 17.4, 16.9, 15.9, 16.8, 18.4, 20.4, 22.6, 25
.1, 21.5, 20.1, 18.1, 18.0], "wind_speed_100m": [26.0, 25.5, 27.7, 27.6, 27.5, 26.1, 25.6, 27.9, 28.5, 28.7, 29.3, 27.5, 27.4, 26.7, 24.8, 23.8, 25.5, 28.0, 31.5, 35.4, 30.8, 29.6, 28.6
, 30.1], "wind_direction_10m": [261, 272, 292, 319, 310, 303, 297, 284, 288, 293, 294, 294, 292, 294, 295, 305, 310, 315, 326, 331, 329, 327, 325, 323], "wind_direction_100m": [266, 279
, 300, 320, 312, 305, 299, 287, 291, 297, 298, 297, 295, 297, 300, 308, 312, 315, 327, 332, 330, 328, 326, 326], "wind_gusts_10m": [19.1, 19.1, 21.2, 26.6, 29.2, 28.8, 28.1, 27.4, 29.9,
 30.2, 31.0, 31.7, 29.5, 28.8, 27.7, 29.2, 32.8, 33.8, 39.6, 44.3, 43.2, 37.1, 33.5, 29.5], "weather_code": [3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 73, 71, 71, 3, 3, 3, 3],
"snow_depth": [0.02, 0.02, 0.02, 0.02, 0.02, 0.02, 0.02, 0.02, 0.02, 0.02, 0.02, 0.02, 0.02, 0.02, 0.02, 0.02, 0.02, 0.02, 0.02, 0.02, 0.02, 0.02, 0.02, 0.02, 0.02]}
```

2.3 U.S. Airline Traffic Data



- **Size:**
  - *249 rows (20 years X 12 months + 9)*
  - *17 columns*

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	Year	Month	Dom_Pax	Int_Pax	Pax	Dom_Flt	Int_Flt	Flt	Dom_RPM	Int_RPM	RPM	Dom_ASM	Int_ASM	ASM	Dom_LF	Int_LF	LF
2	2003	1	43032450	4905830	47938280	785160	57667	842827	36211422	12885980	49097402	56191300	17968572	74159872	64.44	71.71	66.2
3	2003	2	41166780	4245366	45412146	690351	51259	741610	34148439	10715468	44863907	50088434	15587880	65676314	68.18	68.74	68.31
4	2003	3	49992700	5008613	55001313	797194	58926	856120	41774564	12567068	54341633	57592901	17753174	75346075	72.53	70.79	72.12
5	2003	4	47033260	4345444	51378704	766260	55005	821265	39465980	10370592	49836572	54639679	15528761	70168440	72.23	66.78	71.02
6	2003	5	49152352	4610834	53763186	789397	55265	844662	41001934	11575026	52576960	55349897	15629821	70979718	74.08	74.06	74.07
7	2003	6	52209516	5411504	57621020	798351	58225	856576	44492972	13918185	58411157	56555517	17191579	73747096	78.67	80.96	79.2
8	2003	7	55810773	6191120	62001893	831619	62957	894576	48321924	15516794	63838718	59617048	18701759	78318807	81.05	82.97	81.51
9	2003	8	53920973	6272332	60193305	830737	63760	894497	46982527	15906430	62888957	59634190	19144885	78779075	78.78	83.08	79.83
10	2003	9	44213408	4824596	49038004	781804	54017	835821	36819820	13570889	50390709	54973852	17820478	72794329	66.98	76.15	69.22
11	2003	10	49944931	4920822	54865753	818308	54272	872580	41480412	13447446	54927858	58001020	17694417	75695437	71.52	76	72.56
12	2003	11	47059495	4845759	51905254	765842	53817	819659	39333528	12817176	52150704	55785892	16801272	72587163	70.51	76.29	71.85
13	2003	12	49757124	5358991	55116115	798392	57578	855970	42699596	13929565	56629161	58842551	17648530	76491081	72.57	78.93	74.03
14	2004	1	43815481	5255246	49070727	787237	58293	845530	38114607	13515639	51630246	58306593	17988081	76294673	65.37	75.14	67.67
15	2004	2	45306597	4941629	50248226	761618	55822	817440	38575084	12153432	50728516	56180696	16923112	73103808	68.66	71.82	69.39
16	2004	3	54147227	5805703	59952930	834476	60649	895125	46507739	14735428	61243168	61833579	18362220	80195799	75.21	80.25	76.37
17	2004	4	53253194	5664027	58917221	817899	60455	878354	45795662	14338191	60133853	60067863	18457525	78525388	76.24	77.68	76.58
18	2004	5	53030873	5767517	58798390	833350	62596	895946	45350262	15388457	60738719	60923099	19844360	80767459	74.44	77.55	75.2
19	2004	6	56959142	6405771	63364913	836916	64256	901172	49774219	17006831	66781050	61629864	20355959	81985822	80.76	83.55	81.45
20	2004	7	59614287	7183275	66797562	871049	69890	940939	52730292	18182782	70913074	64095669	21731193	85826863	82.27	83.67	82.62
21	2004	8	57380873	6986797	64367670	882979	68332	951311	50816430	17760431	68576861	64479721	21640665	86120386	78.81	82.07	79.63
22	2004	9	47671785	5396245	53068030	819294	56541	875835	40644161	15247819	55891980	57694884	19502122	77197006	70.45	78.19	72.4
23	2004	10	54167489	5629964	59797453	861291	58659	919950	45765588	15384002	61149590	61604719	19883889	81488607	74.29	77.37	75.04
24	2004	11	51781700	5384214	57165914	820048	58312	878360	43561708	14118725	57680434	59431353	18848375	78279728	73.3	74.91	73.69
25	2004	12	52639838	6041853	58681691	836232	63469	899701	45181080	15523403	60704483	62054841	20254013	82308853	72.81	76.64	73.75

2.4 Aircraft Production Data

ALVARO · UPDATED 5 YEARS AGO

6 Code Download

Aircraft Production Data

Production details of lots of aircraft models

Data Card Code (2) Discussion (0) Suggestions (0)

- **Size:**
  - *1266 rows (aircraft models)*
  - *6 columns*

	A	B	C	D	E	F
1		aircraft	nbBuilt	startDate	endDate	retired
2	0	Gotha G.III	25	1916	1916	
3	1	Macchi C.205	262	1942	1944	1950
4	2	Antonov An-30	123	1971	1980	
5	3	Blohm & Voss Ha 137	6	1935	1937	
6	4	Panavia Tornado	992	1979	1998	
7	5	Fairchild Republic A-10 Thunderbolt II	716	1972		
8	6	AD Navyplane	1	1916	1916	
9	7	Yakovlev Yak-9	16769	1942	1948	
10	8	Progressive Aerodyne SeaRey	480	1992		
11	9	Sukhoi Su-24	1400	1967	1993	
12	10	Lockwood Aircam	160	1995		
13	11	RWD 17	30	1938	1939	1939
14	12	Martin 4-0-4	103	1951	1953	
15	13	Blohm & Voss Ha 139	3	1936	1938	
16	14	Hughes OH-6 Cayuse	1420	1965		
17	15	PZL-104 Wilga	1000	1962	2006	
18	16	PR-5 Wiewior plus	2	2010		
19	17	Boeing 737 Classic	1988	1981	2000	
20	18	PZL.26	5	1934	1934	1939
21	19	Blohm & Voss BV 138	297	1938	1938	
22	20	Fleet Finch	606	1939	1941	1947

3. Data Profiling & Cleaning

3.1 Table Flattening & Duplicates

3.2 Type Conversion

3.3 Distribution, Missingness, Uniqueness

3.4 Data Issues

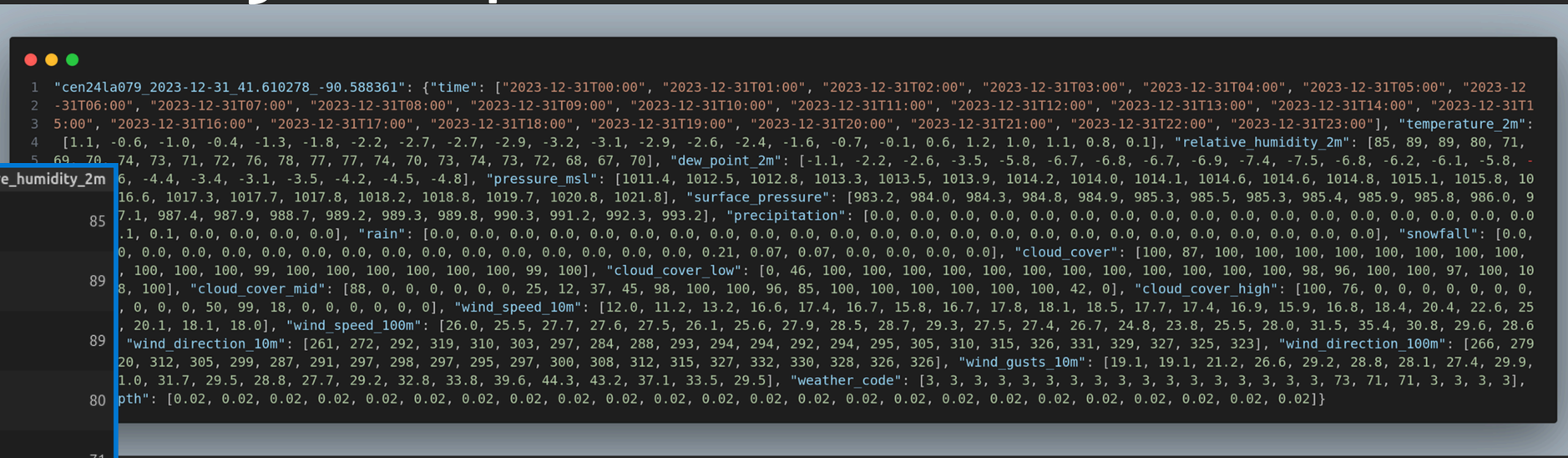
3.1 Table Flattening & Duplicates

- **NTSB Database**
- **Flattening Nested Information**
  - *Convert hierarchical data (like JSON) into a flat, tabular structure for analysis.*
  - *Record will appear multiple times in the resulting DataFrame. The top-level fields get repeated in each row, while any fields from the nested array become columns.*
  - *May introduce key duplication and data redundancy.*


```
"Vehicles": [
    {
      "VehicleNumber": 1,
      "DamageLevel": "None",
      "ExplosionType": "None",
      "FireType": "None",
      "SerialNumber": "C0218",
      "AircraftCategory": "AIR",
      "AmateurBuilt": false,
      "EventID": null,
      "Make": "DIAMOND AIRCRAFT IND INC",
      "Model": "DA20-C1",
      "NumberOfEngines": 1,
      "RegistrationNumber": "N857PA",
      "FlightOperationType": null,
      "Damage": false,
      "AirMedical": false,
      "AirMedicalType": null,
      "flightScheduledType": null,
      "flightServiceType": null,
      "flightTerminalType": null,
      "OperatorName": "DIAMOND AIRCRAFT SALES OF KENTUCKY LLC",
      "RegisteredOwner": "DIAMOND AIRCRAFT SALES OF KENTUCKY LLC",
      "RegulationFlightConductedUnder": "UNK",
      "RepGenFlag": false,
      "RevenueSightseeing": false,
      "SecondPilotPresent": false
    },
    {
      "VehicleNumber": 2,
      "DamageLevel": "None",
      "ExplosionType": "None",
      "FireType": "None",
      "SerialNumber": "1955",
      "AircraftCategory": "HELI",
      "AmateurBuilt": false,
      "EventID": null,
      "Make": "ROBINSON HELICOPTER",
      "Model": "R44",
      "NumberOfEngines": 1,
      "RegistrationNumber": "N744AF",
      "FlightOperationType": null,
      "Damage": false,
      "AirMedical": false,
      "AirMedicalType": null,
      "flightScheduledType": null,
      "flightServiceType": null,
      "flightTerminalType": null,
      "OperatorName": "SKYLINE HELICOPTER TOURS LLC",
```


	Vehicles.VehicleNumber	Vehicles.SerialNumber	Vehicles.Make	Vehicles.Model	NtsbNumber	ProbableCause	City	Country	EventDate	State	Agency	EventType	AirportId	AirportName	Latitude	Longitude
39	1	c0218	diamond aircraft ind inc	da20-c1	ops24la011	None	north las vegas	usa	2023-12-09 13:06:00	nv	ntsb	occ	vgt	north las vegas	36.211268	-115.19968
40	2	1955	robinson helicopter	r44	ops24la011	None	north las vegas	usa	2023-12-09 13:06:00	nv	ntsb	occ	vgt	north las vegas	36.211268	-115.19968

3.1 Table Flattening & Duplicates



	AccidentID	time	temperature_2m	relative_humidity_2m
0	cen24la079_2023-12-31_41.610278_-90.588361	2023-12-31 00:00:00	1.1	85
1	cen24la079_2023-12-31_41.610278_-90.588361	2023-12-31 01:00:00	-0.6	89
2	cen24la079_2023-12-31_41.610278_-90.588361	2023-12-31 02:00:00	-1.0	89
3	cen24la079_2023-12-31_41.610278_-90.588361	2023-12-31 03:00:00	-0.4	80
4	cen24la079_2023-12-31_41.610278_-90.588361	2023-12-31 04:00:00	-1.3	71
5	cen24la079_2023-12-31_41.610278_-90.588361	2023-12-31 05:00:00	-1.8	69
6	cen24la079_2023-12-31_41.610278_-90.588361	2023-12-31 06:00:00	-2.2	70
7	cen24la079_2023-12-31_41.610278_-90.588361	2023-12-31 07:00:00	-2.7	74
8	cen24la079_2023-12-31_41.610278_-90.588361	2023-12-31 08:00:00	-2.7	73
9	cen24la079_2023-12-31_41.610278_-90.588361	2023-12-31 09:00:00	-2.9	71
10	cen24la079_2023-12-31_41.610278_-90.588361	2023-12-31 10:00:00	-3.2	72
11	cen24la079_2023-12-31_41.610278_-90.588361	2023-12-31 11:00:00	-3.1	76

- Open-Meteo Database

3.2 Type Conversion

- **Ensuring Consistent Data Formats**
  - *Convert columns to appropriate data types (e.g., strings to integers, dates to datetime).*
  - *Standardized dates using ISO 8601 format (YYYY-MM-DD) for consistency and compatibility.*
  - *Enables accurate sorting, filtering, and merging across datasets.*

```python
# Type Conversion
df_ntsb['EventDate'] = pd.to_datetime(df_ntsb['EventDate']).dt.tz_localize(None)
df_ntsb['Vehicles.VehicleNumber'] = pd.to_numeric(df_ntsb['Vehicles.VehicleNumber'], errors='coerce').astype(int)
df_ntsb['MKey'] = pd.to_numeric(df_ntsb['MKey'], errors='coerce').astype(int)
df_ntsb['Vehicles.NumberOfEngines'] = pd.to_numeric(df_ntsb['Vehicles.NumberOfEngines'], errors='coerce').fillna(0).astype(int)
df_ntsb['Latitude'] = pd.to_numeric(df_ntsb['Latitude'], errors='coerce').astype(float)
df_ntsb['Longitude'] = pd.to_numeric(df_ntsb['Longitude'], errors='coerce').astype(float)
df_ntsb['TotalInjuryCount'] = pd.to_numeric(df_ntsb['TotalInjuryCount'], errors='coerce').astype(int)
```

3.3 Distribution, Missingness, Uniqueness

- **Data Profiling – Understanding the Dataset**
  - *Distribution: Understand how data is spread — look at min, max, mean, standard deviation, and quartiles to detect skewness or outliers.*
  - *Missingness: Identify how many values are missing per column (count and %) to assess data quality and plan cleaning or imputation.*
  - *Uniqueness: Measure cardinality (number of unique values) to spot identifiers, repeated patterns, or categorical vs. continuous variables.*
  - *Mode & Frequency: Reveal dominant values, helpful for detecting defaults or common entries.*

Column	DataType	TotalCount	NonNullCount	NumMissing	MissingPerc	Cardinality	Mode	ModeFreq	Mean	Min	Q25	Q50	Q75	Max	Std

3.4 Data Issues

- **Invalid Values in Aircraft Models Data Columns**
  - *Some entries in startDate contain values unrelated to actual years.*
  - *Affects temporal analysis and integration with time-based datasets.*
  - *Manually corrected a few anomalies using reliable sources from the web.*

	Column	DataType	Min	Q25	Q50	Q75	Max	Std
0	aircraft	object	NaN	NaN	NaN	NaN	NaN	NaN
1	nbBuilt	int64	0.0	32.25	185.0	703.00	43400.0	3618.899938
2	startDate	int64	1.0	1937.00	1951.0	1974.75	2015.0	224.918816
3	endDate	Int64	1.0	1938.00	1949.0	1979.00	2016.0	227.826756

```python
df_filtered = df_aircraft[(df_aircraft['startDate'] < 1000) | (df_aircraft['endDate'] < 1000)]
df_filtered.style.map(
    lambda val: 'background-color: red' if val < 1000 else '',
    subset=['startDate', 'endDate']
)
```

	aircraft	nbBuilt	startDate	endDate
82	lockheed c-5 galaxy	131	5	5
86	british aerospace nimrod aew3	8	11	11
171	schneider es-57 kingfisher	11	2	
190	bell 222	230	222	1991
284	flitfire	49	10	10
308	grumman c-2 greyhound	58	2	2
498	chu hummingbird	2	2	2
514	embraer legacy 500	500	500	
518	lockheed martin f-22 raptor	195	22	22
536	gallaudet d-4	2	2	2
637	fleet canuck	225	198	198
668	bell ah-1 supercobra	1271	1	1
688	chu cjc-3	1	1	1
896	dallach sunrise	0	5	5
951	sukhoi su-30mki	200	30	30
1049	boeing kb-29 superfortress	282	92	
1089	myasishchev m-4	2	93	93
1200	yakovlev yak-100	2	115	115

3.4 Data Issues

- **Removing Formatting for Type Conversion in Airline Traffic Data**
  - *Numbers stored as strings with commas (e.g., "1,200") can't be converted to integers.*
  - *Strip commas before converting to numeric types.*
  - *Ensures correct data types for computation and analysis.*

```
Year,Month,Dom Pax,Int Pax,Pax,Dom Flt,
2003,1,"43,032,450","4,905,830","47,938,280","785,160",
2003,2,"41,166,780","4,245,366","45,412,146","690,351",
2003,3,"49,992,700","5,008,613","55,001,313","797,194",
```

```
# Remove commas from all columns and then convert
df_airline_traffic = df_airline_traffic.replace(',', '', regex=True)

# Now convert each column to numeric. If everything converts well, no rows become NaN.
df_airline_traffic = df_airline_traffic.apply(pd.to_numeric, errors='coerce').astype(int)
```

4. Integrated Schema

4.1 Data Selection

- Focused on relevant attributes to answer the research question
- Removed unnecessary data for clarity and efficiency

4.2 Data Integration

- Combined multiple tables into a single unified schema
- Maintained data consistency, integrity, and efficiency

4.3 Schema Design

- Used high-cardinality relationships for optimal data linkage
- Supported different types of correspondences (e.g., many-to-one)

4.4 Binding Methods

- Applied techniques discussed in “Identity Resolution”

4. Integrated Schema

Origin Dataset	From	Target	Type Corresp.	Description
NTSB	NtsbNumber	AccidentNumber	1-1	
NTSB	EventDate	DateTime	1-1	
NTSB	City	City	1-1	
NTSB	State	State	1-1	
NTSB	Longitude	Longitude	1-1	
NTSB	Latitude	Latitude	1-1	
NTSB	AirportName	AirportName	1-1	
NTSB	Operator	Operator	1-1	
NTSB	Aircraft Damage	Aircraft Damage	1-1	
NTSB	FatalInjuryCount; SeriousInjuryCount; MinorInjuryCount	TotalInjuryCount	N-1	Sum all the value
NTSB	HighestInjury	HighestInjury	1-1	
NTSB	Model, Make	Aircraft	N-1	Blocking with Aircraft Data
Aircraft Data	StartDate	ProductionStartDate	1-1	
Aircraft Data	EndDate	ProductionEndDate	1-1	
Weather API	Temperature_2m	Temperature	1-1	
Weather API	Precipitation	Precipitation	1-1	
Weather API	Wind_Speed_10m	WindSpeed	1-1	
Weather API	Weather code	Weather code	1-1	
Weather API	other weather info	other weather info	1-1	
Airline Traffic	Pax	PassengersPerMonth	1-1	
Airline Traffic	Flt	FlightsPerMonth	1-1	
Airline Traffic	LF	LoadFactorPerMonth	1-1	

TABLE V
INTEGRATED MODEL

5. Identity Resolution & Blocking

1. Blocking with Q-grams (q=3) and Substring Matching
2. Conditional Numeric Filter
3. Combined Textual Similarity (Threshold 0.75)

	startDate	Vehicles.Model	Matched_AircraftI	JW	LEV	JAC	FinalScore
2.	1965	concorde	concorde	1.0	1.0	1.0	1.0
3.	1938	quad city challenger	quad city challenger	1.0	1.0	1.0	1.0
8.	1967	bac 167 strikemaster	bac strikemaster	0.96	0.8	0.66	0.82
9.	1962	pzl 104m wilga	pzl 104 wilga	0.98	0.92	0.5	0.82
33.	1985	gulfstream	gulfstream iv	0.95	0.76	0.5	0.76

Final Score = 0.4 × Jaro-Winkler + 0.3 × Levenshtein + 0.3 × Jaccard

6. Preliminary Insights & Challenges

- **Passenger Volume Limitation:**

  Passenger data is monthly, while accidents occur daily—this 1-to-many cardinality reduces correlation accuracy

- **Aircraft Data Limitation:**

  The aircraft dataset lacks detailed specs restricting deeper analysis.

- **Low Match Rate:**

  From ~23,000 accident records, only 38 aircraft matched due to identity resolution limits

7. Conclusion

- Conclusion
  - *We performed data cleaning & schema integration for four data sources*
  - *Implemented identity resolution to match aircraft models*
  - *Preliminary results show plausible correlations but more analysis needed*

8. References

- National Transportation Safety Board (NTSB). url: https://www.ntsb.gov/
- Yyxian. U.S. Airline Traffic Data. Kaggle Dataset. url: https://www.kaggle.com/datasets/yyxian/u-s-airline-traffic-data/
- Alvaroibrain. Aircraft Production Data. Kaggle Dataset. url: https://www.kaggle.com/datasets/alvaroibrain/aircraft-production-data
- Open-Meteo Historical Weather API Documentation. url: https://open-meteo.com/en/docs/historical-weather-api?

Thanks for the attention!