Chosen coverage criteria: Input space partitioning

For every function, we basically separated the test cases into 2 sections, where 1 section contains the tests for valid inputs, and 1 section contains the tests for invalid inputs.

For the valid input section, the purpose is to test whether the function returns the expected output in the correct format given that the valid inputs are passed to the function. The purpose of invalid input section, on the other hand, is to test whether the relevant exceptions are raised by the function given that the invalid inputs are passed to the function.

Function 1: airline_stats()

Valid inputs section:

Input parameters: origin, destination, yearFrom, yearTo, month (optional), day (optional)

Characteristics used to partition: year period (5 years period per partition), distance between the origin airport and destination airport

The visualisation of our partitions and input values from each partition is shown in the image below:

Partition	Year Period	Distance	yearFrom	yearTo	Origin	Destination
1	1990 - 1994	< 1000 miles	1990	1994	SAT	PHX
2		>= 1000 miles and <= 4000 miles	1990	1994	SFO	EWR
3		> 4000 miles	1990	1994	HNL	JFK
4	1995 - 1999	< 1000 miles	1995	1999	LGA	ORD
5		>= 1000 miles and <= 4000 miles	1995	1999	IAH	SEA
6		> 4000 miles	1995	1999	HNL	ATL
7	2000 - 2004	< 1000 miles	2000	2004	DFW	MKE
8		>= 1000 miles and <= 4000 miles	2000	2004	HNL	LAX
9		> 4000 miles	2000	2004	IAS	HNL
10	2005 - 2009	< 1000 miles	2005	2009	BOS	IAD
11		>= 1000 miles and <= 4000 miles	2005	2009	LAS	JFK
12		> 4000 miles	2005	2009	OGG	ORD
13	2010 - 2014	< 1000 miles	2010	2014	ORD	DEN
14		>= 1000 miles and <= 4000 miles	2010	2014	ATL	SAN
15		> 4000 miles	2010	2014	BOS	HNL
16	2015 - 2019	< 1000 miles	2015	2019	DEN	SNA
17		>= 1000 miles and <= 4000 miles	2015	2019	PHX	EWR
18		> 4000 miles	2015	2019	EWR	HNL
19	2020 - 2021	< 1000 miles	2020	2021	SFO	SAN
20		>= 1000 miles and <= 4000 miles	2020	2021	OAK	BWI
21		> 4000 miles	2020	2021	ORD	OGG

Invalid input section:

Invalid input characteristics: invalid year, invalid airport, invalid flight, invalid month, invalid day, illegal input (missing required parameter), wrong logic input (yearFrom better than yearTo)

The visualisation of invalid input values used is shown in the image below:

Invalid Input	Characteristics	yearFrom	yearTo	Origin	Destination	Month	Day
1	Invalid Year	1990	2030	ORD	OGG		
2	Invalid Airport	1990	2021	AAA	OGG		
3	Invalid Flight	1990	2021	BOS	BOS		
4	Invalid Month	1990	2021	SAT	PHX	14	
5	Invalid Day	1990	2021	SAT	PHX	12	40
6	Illegal (Missing required parameter)			SAT	PHX		
7	Wrong Logic (yearFrom greater than yearTo)	2021	1990	SAT	PHX		

Function 2: airline_delays() & **Function 3:** delays_comparison()

**We use the same input values to test both airline_delays() and delays_comparison() because both functions have the same required input parameters*

Valid inputs section:

Input parameters: origin, destination, airline, yearFrom, yearTo, month (optional), day (optional)

Characteristics used to partition: year period (5 years period per partition), distance between the origin airport and destination airport

The visualisation of our partitions and input values from each partition is shown in the image below:

Partition	Year Period	Distance	yearFrom	yearTo	Origin	Destination	Airline
1	1990 - 1994	< 1000 miles	1990	1994	SAT	PHX	VX
2		>= 1000 miles and <= 4000 miles	1990	1994	SFO	EWR	US
3		> 4000 miles	1990	1994	HNL	JFK	DL
4	1995 - 1999	< 1000 miles	1995	1999	LGA	ORD	AX
5		>= 1000 miles and <= 4000 miles	1995	1999	IAH	SEA	NK
6		> 4000 miles	1995	1999	HNL	ATL	EM
7	2000 - 2004	< 1000 miles	2000	2004	DFW	MKE	G7
8		>= 1000 miles and <= 4000 miles	2000	2004	HNL	LAX	YX
9		> 4000 miles	2000	2004	IAH	HNL	OH
10	2005 - 2009	< 1000 miles	2005	2009	BOS	IAD	PT
11		>= 1000 miles and <= 4000 miles	2005	2009	LAS	JFK	KS
12		> 4000 miles	2005	2009	OGG	ORD	YV
13	2010 - 2014	< 1000 miles	2010	2014	ORD	DEN	B6
14		>= 1000 miles and <= 4000 miles	2010	2014	ATL	SAN	9E
15		> 4000 miles	2010	2014	BOS	HNL	HA
16	2015 - 2019	< 1000 miles	2015	2019	DEN	SNA	OO
17		>= 1000 miles and <= 4000 miles	2015	2019	PHX	EWR	F9
18		> 4000 miles	2015	2019	EWR	HNL	UA
19	2020 - 2021	< 1000 miles	2020	2021	SFO	SAN	QX
20		>= 1000 miles and <= 4000 miles	2020	2021	OAK	BWI	WN
21		> 4000 miles	2020	2021	ORD	OGG	UA

Invalid input section:

Invalid input characteristics: invalid year, invalid airport, invalid flight, invalid airline, invalid month, invalid day, illegal input (missing required parameter), wrong logic input (yearFrom better than yearTo)

The visualisation of invalid input values used is shown in the image below:

Invalid Input	Characteristics	yearFrom	yearTo	Airline	Origin	Destination	Month	Day
1	Invalid Year	1985	2030	AA	SAT	PHX		
2	Invalid Airport	1990	2021	AA	BBB	CCC		
3	Invalid Flight	1990	2021	AA	SAT	SAT		
4	Invalid Airline	1990	2021	NOT	SAT	PHX		
5	Invalid Month	1990	2021	AA	SAT	PHX	20	
6	Invalid Day	1990	2021	AA	SAT	PHX	12	-1
7	Illegal (Missing required parameter)			AA	SAT	PHX		
8	Wrong Logic (yearFrom greater than yearTo)	2021	1990	AA	SAT	PHX		

Justification for some main decisions

Case 1: Input given is valid, but the query result is empty

We set the expected output for the function equal to either empty object or empty array instead of raising exceptions for this situation. This is because we think that exceptions should not be raised if the given input is valid. If an exception is raised in this situation, the user of the function (one that makes a GET request to the functions) might think that the input he passes to the function is invalid, thus the user might be confused.

Case 2: Input given is invalid

We think that relevant exceptions should be raised by the function. By raising exceptions, the user of the function can be informed that the input he passes is invalid, thus able to handle it accordingly.

Case 3: Only one airline involved in a particular flight (for delays_comparison())

We think that an array of 2 objects should be returned instead of an array of 1 object. We would like to use the example below to explain this:

params = {'o': "EWR", 'dst': "HNL", 'a': "UA",'yf' : 2015, 'yt' : 2019}

If the above input is passed to the delays_comparison() function, the function will return only one object: [{'carrier': 6.1303, 'nas': 2.713, 'sec': 0.0, 'late_aircraft': 4.0792, 'weather': 0.8791}], this is because there is only 1 airline (UA) is involved in this flight.

But we think that an array of 2 objects: [{'carrier': 6.1303, 'nas': 2.713, 'sec': 0.0, 'late_aircraft': 4.0792, 'weather': 0.8791},{'carrier': 6.1303, 'nas': 2.713, 'sec': 0.0, 'late_aircraft': 4.0792, 'weather': 0.8791}] should be returned in this case, because if only one object is returned, the bar chart in the front end of this app won't show.

Test cases involved: testcase20, testcase20, testcase21 (test_delays_comparison.py)