Project Statement for Milestone 4

Group 6

Group member: Jinyang Ruan, Rusu Wu, Yi Yao, Brian Chan, Junqiao Mou

In [1]:

```python
from pyspark.sql import SparkSession
from sedona.register import SedonaRegistrator
from sedona.utils import SedonaKryoRegistrator, KryoSerializer
import geopandas as gpd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

spark = SparkSession.\
    builder.\
    appName('Python Spark Apache Sedona').\
    master('local[*]').\
    config("spark.serializer", KryoSerializer.getName).\
    config("spark.kryo.registrator", SedonaKryoRegistrator.getName).\
    config('spark.jars.packages','org.apache.sedona:sedona-python-adapter-3.0_2.12:1.0.1-incubating,'
           'org.datasyslab:geotools-wrapper:geotools-24.1').\
    getOrCreate()

SedonaRegistrator.registerAll(spark)
```

```
Ivy Default Cache set to: /root/.ivy2/cache
The jars for the packages stored in: /root/.ivy2/jars
:: loading settings :: url = jar:file:/home/ubuntu/spark-3.0.3-bin-hadoop3.2/jars/ivy-2.4.0.jar!/org/apache/ivy/core/settings/ivysettings.xml
org.apache.sedona#sedona-python-adapter-3.0_2.12 added as a dependency
org.datasyslab#geotools-wrapper added as a dependency
:: resolving dependencies :: org.apache.spark#spark-submit-parent-7f20cc1f-9313-4965-bd9f-54d02e90380e;1.0
        confs: [default]
        found org.apache.sedona#sedona-python-adapter-3.0_2.12;1.0.1-incubating in central
        found org.locationtech.jts#jts-core;1.18.0 in central
        found org.wololo#jts2geojson;0.16.1 in central
        found com.fasterxml.jackson.core#jackson-databind;2.12.2 in central
        found com.fasterxml.jackson.core#jackson-annotations;2.12.2 in central
        found com.fasterxml.jackson.core#jackson-core;2.12.2 in central
        found org.apache.sedona#sedona-core-3.0_2.12;1.0.1-incubating in central
        found org.apache.sedona#sedona-sql-3.0_2.12;1.0.1-incubating in central
        found org.datasyslab#geotools-wrapper;geotools-24.1 in central
:: resolution report :: resolve 423ms :: artifacts dl 9ms
        :: modules in use:
        com.fasterxml.jackson.core#jackson-annotations;2.12.2 from central in [default]
        com.fasterxml.jackson.core#jackson-core;2.12.2 from central in [default]
        com.fasterxml.jackson.core#jackson-databind;2.12.2 from central in [default]
        org.apache.sedona#sedona-core-3.0_2.12;1.0.1-incubating from central in [default]
        org.apache.sedona#sedona-python-adapter-3.0_2.12;1.0.1-incubating from central in [default]
        org.apache.sedona#sedona-sql-3.0_2.12;1.0.1-incubating from central in [default]
        org.datasyslab#geotools-wrapper;geotools-24.1 from central in [default]
        org.locationtech.jts#jts-core;1.18.0 from central in [default]
        org.wololo#jts2geojson;0.16.1 from central in [default]
        :: evicted modules:
        org.locationtech.jts#jts-core;1.18.1 by [org.locationtech.jts#jts-core;1.18.0] in [default]
        ---------------------------------------------------------------------
        |                  |            modules            ||   artifacts   |
        |       conf       | number| search|dwnlded|evicted|| number|dwnlded|
        ---------------------------------------------------------------------
        |      default     |   10  |   0   |   0   |   1   ||   9   |   0   |
        ---------------------------------------------------------------------
:: retrieving :: org.apache.spark#spark-submit-parent-7f20cc1f-9313-4965-bd9f-54d02e90380e
        confs: [default]
        0 artifacts copied, 9 already retrieved (0kB/13ms)
21/11/25 01:27:49 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
```

Out[1]: True

In [2]:

```python
# data preparation

# airports
airports_dat = spark.read.option("delimiter", ",").option("header", "false").csv("airports.dat").\
                                toDF("AirportID","Name","City","Country","IATA","ICAO","Latitude",
                                     "Longitude","Altitude","Timezone","DST","Tz","Type","Source")
airports = airports_dat.selectExpr("AirportID","Name","City","Country","IATA","ICAO","Latitude",
                                   "Longitude","Altitude","Timezone","DST","Tz","Type","Source")
airports.createOrReplaceTempView("airports")

# routes
routes_dat = spark.read.option("delimiter", ",").option("header","false").csv("routes.dat").\
                                toDF("Airline","AirlineID","SourceAirport","SourceAirportID",
                                     "DestinationAirport","DestinationAirportID","Codeshare","Stops","Equipment")
routes = routes_dat.selectExpr("Airline","AirlineID","SourceAirport","SourceAirportID","DestinationAirport",
                               "DestinationAirportID","Codeshare","Stops","Equipment")
routes.createOrReplaceTempView("routes")

# airlines
airlines_dat = spark.read.option("delimiter", ",").option("header","false").csv("airlines.dat").\
                                toDF("AirlineID","Name","Alias","ITAT","ICAO","Callsign","Country", "Active")
airlines = airlines_dat.selectExpr("AirlineID","Name","Alias","ITAT","ICAO","Callsign","Country","Active")
airlines.createOrReplaceTempView("airlines")

# states
states_wkt = spark.read.option("delimiter", "\t").option("header", "false").csv("boundary-each-state.tsv").\
             toDF("s_name","s_bound")
states = states_wkt.selectExpr("s_name", "ST_GeomFromWKT(s_bound) as s_bound")
states.createOrReplaceTempView("states")
```

Find the list of airports operating in the Country X

e.g. Find the list of aiports operating in the United States (only showing top 10)

In [3]:

```python
airports_US = spark.sql("""
                        SELECT *
                        FROM airports
                        WHERE Country = 'United States'""")
airports_US.toPandas().head(10).style
```

Out[3]:

	AirportID	Name	City	Country	IATA	ICAO	Latitude	Longitude	Altitude	Timezone	DST	Tz	Type	Source
0	3411	Barter Island LRRS Airport	Barter Island	United States	BTI	PABA	70.1340026855	-143.582000732	2	-9	A	America/Anchorage	airport	OurAirports
1	3412	Wainwright Air Station	Fort Wainwright	United States	\N	PAWT	70.61340332	-159.8600006	35	-9	A	America/Anchorage	airport	OurAirports
2	3413	Cape Lisburne LRRS Airport	Cape Lisburne	United States	LUR	PALU	68.87509918	-166.1100006	16	-9	A	America/Anchorage	airport	OurAirports
3	3414	Point Lay LRRS Airport	Point Lay	United States	PIZ	PPIZ	69.73290253	-163.0050049	22	-9	A	America/Anchorage	airport	OurAirports
4	3415	Hilo International Airport	Hilo	United States	ITO	PHTO	19.721399307250977	-155.04800415039062	38	-10	N	Pacific/Honolulu	airport	OurAirports
5	3416	Orlando Executive Airport	Orlando	United States	ORL	KORL	28.5455	-81.332901	113	-5	A	America/New_York	airport	OurAirports
6	3417	Bettles Airport	Bettles	United States	BTT	PABT	66.91390228	-151.529007	647	-9	A	America/Anchorage	airport	OurAirports
7	3418	Clear Airport	Clear Mews	United States	\N	PACL	64.301201	-149.119995	552	-9	A	America/Anchorage	airport	OurAirports
8	3419	Indian Mountain LRRS Airport	Indian Mountains	United States	UTO	PAIM	65.99279785	-153.7039948	1273	-9	A	America/Anchorage	airport	OurAirports
9	3420	Fort Yukon Airport	Fort Yukon	United States	FYU	PFYU	66.57150268554688	-145.25	433	-9	A	America/Anchorage	airport	OurAirports

Find the list of airlines having X stops

e.g. Find the list of Airlines having 1 stop (only showing top 10)
There are two categories for "Stops" listed as "0" or "1"

In [4]:

```python
routes_1Stop = spark.sql("""
                       SELECT *
                       FROM routes
                       WHERE Stops = 1""")
routes_1Stop.toPandas().head(10).style
```

Out[4]:

	Airline	AirlineID	SourceAirport	SourceAirportID	DestinationAirport	DestinationAirportID	Codeshare	Stops	Equipment
0	5T	1623	YRT	132	YEK	50	None	1	ATR
1	AC	330	ABJ	253	BRU	302	None	1	333
2	AC	330	YVR	156	YBL	30	None	1	BEH
3	CU	1936	FCO	1555	HAV	1909	None	1	767
4	FL	1316	HOU	3566	SAT	3621	None	1	735
5	FL	1316	MCO	3878	HOU	3566	None	1	73W
6	FL	1316	MCO	3878	ORF	3611	None	1	717
7	SK	4319	ARN	737	GEV	715	None	1	ATP
8	WN	4547	BOS	3448	MCO	3878	None	1	73W
9	WN	4547	MCO	3878	BOS	3448	None	1	73W

Find list of airlines operating with code share (only showing top 10)

There are two categories for "Codeshare" listed as NULL or "Y"

In [5]:

```python
routes_Codeshare = spark.sql("""
                           SELECT *
                           FROM routes
                           WHERE Codeshare = 'Y'""")
routes_Codeshare.toPandas().head(10).style
```

Out[5]:

	Airline	AirlineID	SourceAirport	SourceAirportID	DestinationAirport	DestinationAirportID	Codeshare	Stops	Equipment
0	2P	897	GES	2402	MNL	2397	Y	0	320
1	2P	897	MNL	2397	GES	2402	Y	0	320
2	4M	3201	DFW	3670	EZE	3988	Y	0	777
3	4M	3201	EZE	3988	DFW	3670	Y	0	777
4	4M	3201	EZE	3988	JFK	3797	Y	0	777
5	4M	3201	JFK	3797	EZE	3988	Y	0	777
6	5N	503	ARH	4362	CSH	6110	Y	0	AN4
7	5N	503	ARH	4362	MMK	2949	Y	0	AN4
8	5N	503	ARH	4362	USK	4369	Y	0	AN4
9	5N	503	CSH	6110	ARH	4362	Y	0	AN4

Find the list of active airlines in the United Sates - Airline aggregation (only showing top 10)

In [6]:

```python
airlines_Active = spark.sql("""
                            SELECT *
                            FROM airlines
                            WHERE Country = 'United States'
                                AND Active = 'Y'""")
airlines_Active.toPandas().head(10).style
```

Out[6]:

	AirlineID	Name	Alias	ITAT	ICAO	Callsign	Country	Active
0	10	40-Mile Air	\N	Q5	MLA	MILE-AIR	United States	Y
1	22	Aloha Airlines	\N	AQ	AAH	ALOHA	United States	Y
2	24	American Airlines	\N	AA	AAL	AMERICAN	United States	Y
3	35	Allegiant Air	\N	G4	AAY	ALLEGIANT	United States	Y
4	109	Alaska Central Express	\N	KO	AER	ACE AIR	United States	Y
5	149	Air Cargo Carriers	\N	2Q	SNC	NIGHT CARGO	United States	Y
6	210	Airlift International	\N	None	AIR	AIRLIFT	United States	Y
7	281	America West Airlines	\N	HP	AWE	CACTUS	United States	Y
8	282	Air Wisconsin	\N	ZW	AWI	AIR WISCONSIN	United States	Y
9	287	Allegheny Commuter Airlines	\N	None	ALO	ALLEGHENY	United States	Y

Find the country (or) territory has the highest number of Airports

e.g. The table below shows the list of countries have the top 10 highest number of Airports

In [7]:

```python
airportsByCountry = spark.sql("""
                            SELECT Country,
                                COUNT(DISTINCT AirportID) AS Airport_cnt
                            FROM airports
                            GROUP BY 1
                            ORDER BY 2 DESC""")
airportsByCountry_top10 = airportsByCountry.toPandas().head(10)
airportsByCountry_top10.style
```

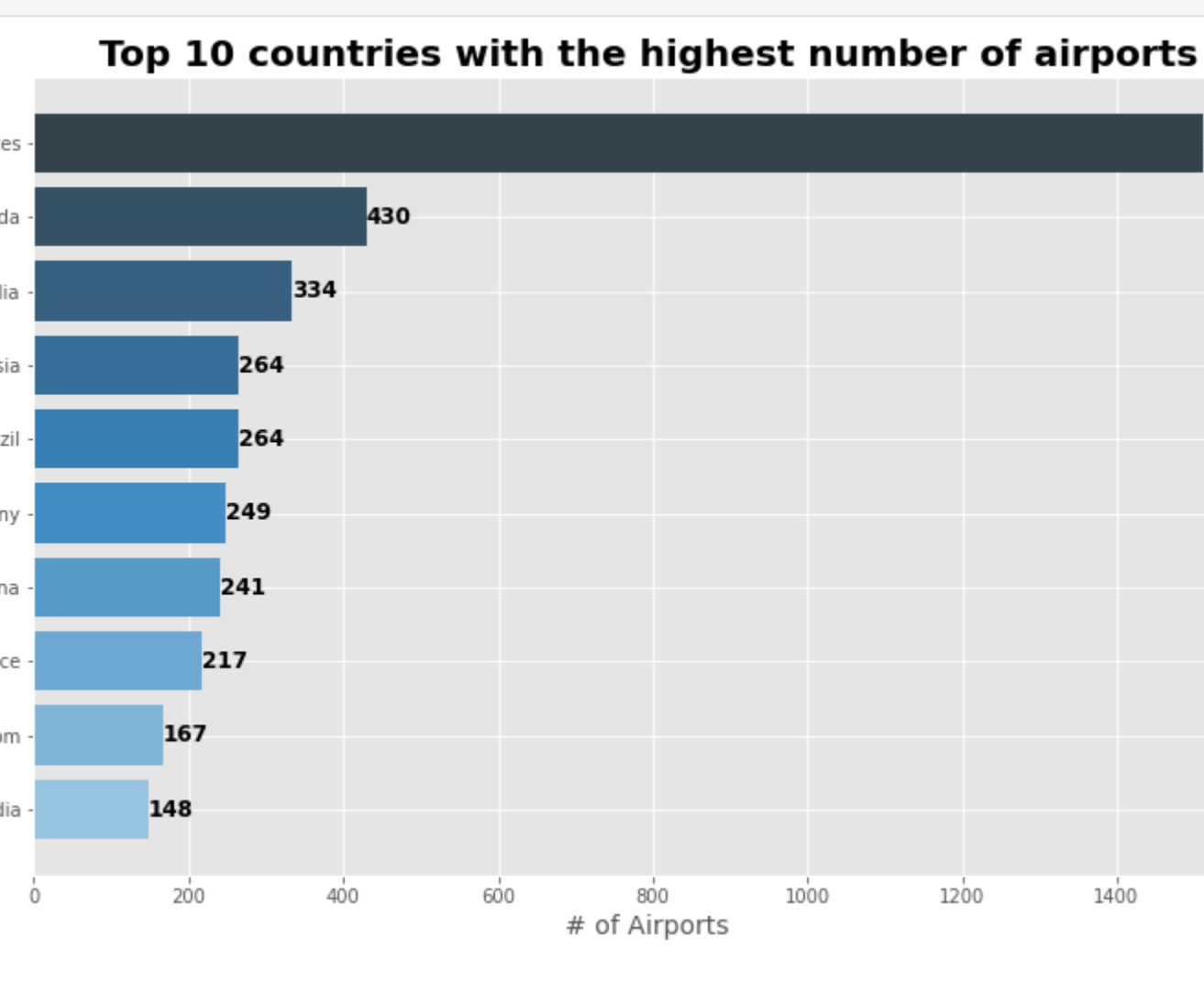
Out[7]:

	Country	Airport_cnt
0	United States	1512
1	Canada	430
2	Australia	334
3	Russia	264
4	Brazil	264
5	Germany	249
6	China	241
7	France	217
8	United Kingdom	167
9	India	148

The horizontal barplot below can be used to visualize the "top 10 countries with the highest number of airports".
It is easy to tell from the chart using either the width of the bars or the color of the bars.
The country with the most airports, which is the United States, has the longest bar in the darkest color.

In [8]:

```python
plt.style.use("ggplot")
plt.figure(figsize = (12,8))
pal = sns.color_palette("Blues_d", len(airportsByCountry_top10["Airport_cnt"]))
hbars = plt.barh(width="Airport_cnt", y="Country", data=airportsByCountry_top10, color=np.array(pal[::-1]))
plt.gca().invert_yaxis()
plt.title("Top 10 countries with the highest number of airports", fontsize = 20, weight = 'bold')
plt.xlabel("# of Airports", size = 14)
plt.ylabel("Country", size = 14)
plt.bar_label(hbars, size = 12, weight = "bold")
plt.show()
```



Find the top K cities with most incoming airlines

e.g. Find the top 10 cities with most incoming airlines
Airlines with the same airlineIDs coming from different source airports are considered as one airline

In [9]:

```python
IncomeAirlinesByCity = spark.sql("""
                                SELECT
                                    a.City,
                                    COUNT(DISTINCT r.AirlineID) AS IncomeAirline_cnt
                                FROM routes r
                                LEFT JOIN airports a
                                    ON r.DestinationAirportID = a.AirportID
                                GROUP BY 1
                                HAVING a.City IS NOT NULL
                                ORDER BY 2 DESC""")
IncomeAirlinesByCity_top10 = IncomeAirlinesByCity.toPandas().head(10)
IncomeAirlinesByCity_top10.style
```

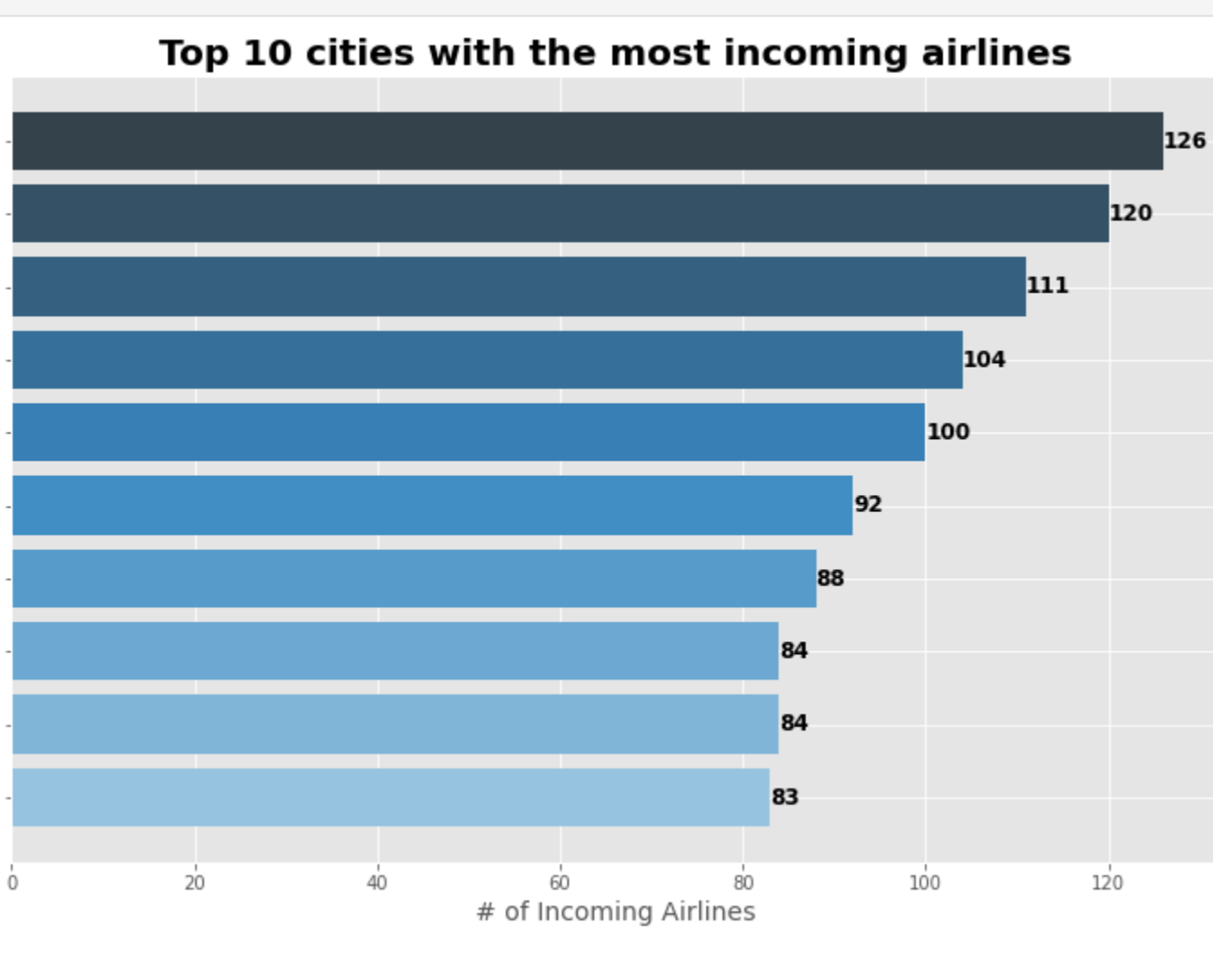
Out[9]:

	City	IncomeAirline_cnt
0	Paris	126
1	London	120
2	Moscow	111
3	Bangkok	104
4	Frankfurt	100
5	Rome	92
6	Dubai	88
7	Istanbul	84
8	Hong Kong	84
9	Singapore	83

The horizontal barplot below can be used to visualize the "top 10 cities with most incoming airlines".
It is easy to tell from the chart using either the width of the bars or the color of the bars.
The city with the most incoming airlines has the bar with the greatest length or the one with the darkest color.

In [10]:

```python
plt.figure(figsize = (12,8))
pal = sns.color_palette("Blues_d", len(IncomeAirlinesByCity_top10["IncomeAirline_cnt"]))
hbars = plt.barh(width="IncomeAirline_cnt", y="City", data=IncomeAirlinesByCity_top10, color=np.array(pal[::-1]))
plt.gca().invert_yaxis()
plt.title("Top 10 cities with the most incoming airlines", fontsize = 20, weight = 'bold')
plt.xlabel("# of Incoming Airlines", size = 14)
plt.ylabel("City", size = 14)
plt.bar_label(hbars, size = 12, weight = "bold")
plt.show()
```



Find the closest airport to a city X's geospatial coordinate

Example 1: Find the top 10 closest airport to the point with the coordinate (37.794076, -101.332631)

In [11]:

```python
airports_point = spark.sql("""
                            SELECT *,
                                ST_Point(CAST(Longitude as Double), CAST(Latitude as Double)) as Point
                            FROM airports""")
airports_point.createOrReplaceTempView("airports_point")
```

In [12]:

```python
dist_to_Pullman = spark.sql("""
                            SELECT
                                Name,
                                City,
                                Latitude,
                                Longitude,
                                Point as Coordinate,
                                ST_Distance(Point, ST_Point(-101.332631, 37.794076)) as Distance
                            FROM airports_point
                            ORDER BY Distance""")
dist_to_Pullman_pd = dist_to_Pullman.toPandas()
dist_to_Pullman_pd.head(10).style
```

Out[12]:

	Name	City	Latitude	Longitude	Coordinate	Distance
0	Ulysses Airport	Ulysses	37.60400009	-101.3740005	POINT (-101.3740005 37.60400009)	0.194526
1	Sublette Municipal Airport	Sublette	37.49140167	-100.8300018	POINT (-100.8300018 37.49140167)	0.586726
2	Garden City Regional Airport	Garden City	37.9275016785	-100.723999023	POINT (-100.723999023 37.9275016785)	0.623085
3	Scott City Municipal Airport	Scott City	38.474300384521484	-100.88500213623047	POINT (-100.8850021362305 38.47430038452148)	0.814295
4	Liberal Mid-America Regional Airport	Liberal	37.0442009	-100.9599991	POINT (-100.9599991 37.0442009)	0.837357
5	Elkhart Morton County Airport	Elkhart	37.000702	-101.879997	POINT (-101.879997 37.000702)	0.963873
6	Meade Municipal Airport	Meade	37.27690124511719	-100.35600280761719	POINT (-100.3560028076172 37.27690124511719)	1.105112
7	Guymon Municipal Airport	Guymon	36.6851005554	-101.508003235	POINT (-101.508003235 36.6851005554)	1.122756
8	Dodge City Regional Airport	Dodge City	37.76340103149414	-99.9655990600586	POINT (-99.96559906005859 37.76340103149414)	1.367376
9	Lamar Municipal Airport	Lamar	38.069698333699996	-102.68800354	POINT (-102.68800354 38.0696983337)	1.383113

Example 2: Find the top 10 closest airport to the point with the coordinate (34.04743, -118.24903)

In [13]:

```python
dist_to_LA = spark.sql("""
                        SELECT
                            Name,
                            City,
                            Latitude,
                            Longitude,
                            Point as Coordinate,
                            ST_Distance(Point, ST_Point(-118.24903,34.04743)) as Distance
                        FROM airports_point
                        ORDER BY Distance""")
dist_to_LA.toPandas().head(10).style
```

Out[13]:

	Name	City	Latitude	Longitude	Coordinate	Distance
0	Jack Northrop Field Hawthorne Municipal Airport	Hawthorne	33.922798	-118.334999	POINT (-118.334999 33.922798)	0.151406
1	Bob Hope Airport	Burbank	34.20069885253906	-118.35900115966797	POINT (-118.359001159668 34.20069885253906)	0.188640
2	Los Angeles International Airport	Los Angeles	33.94250107	-118.4079971	POINT (-118.4079971 33.94250107)	0.190475
3	Santa Monica Municipal Airport	Santa Monica	34.015800476100004	-118.450996399	POINT (-118.450996399 34.0158004761)	0.204428
4	San Gabriel Valley Airport	El Monte	34.086102	-118.035004	POINT (-118.035004 34.086102)	0.217492
5	Long Beach /Daugherty Field/ Airport	Long Beach	33.81769943	-118.1520004	POINT (-118.1520004 33.81769943)	0.249381
6	Zamperini Field	Torrance	33.803398132324	-118.33999633789	POINT (-118.33999633789 33.803398132324)	0.260435
7	Whiteman Airport	Los Angeles	34.2593002319	-118.413002014	POINT (-118.413002014 34.2593002319)	0.267910
8	Van Nuys Airport	Van Nuys	34.209800720215	-118.48999786377	POINT (-118.48999786377 34.209800720215)	0.290568
9	Fullerton Municipal Airport	Fullerton	33.8720016479	-117.980003357	POINT (-117.980003357 33.8720016479)	0.321170

Find the airport in each US state's geospatial boundary

In [14]:

```python
airports_point = spark.sql("""
                            SELECT *,
                                ST_Point(CAST(Longitude as Double), CAST(Latitude as Double)) as Point
                            FROM airports""")
airports_point.createOrReplaceTempView("airports_point")
```

In [15]:

```python
airport_per_state = spark.sql("""
                            SELECT states.s_name,
                                states.s_bound,
                                airports_point.AirportID,
                                airports_point.Name,
                                airports_point.City,
                                airports_point.Point
                            FROM states, airports_point
                            WHERE ST_Contains(states.s_bound, airports_point.Point)
                            ORDER BY states.s_name""")
airport_per_state.createOrReplaceTempView("airport_per_state")
```

We can count the number of airports in each state and visualize using choropleth maps.
The state having the most number of airports has the darkest color.

In [16]:

```python
airport_cnt_per_state = spark.sql("""
                                    SELECT s_name as State,
                                        s_bound,
                                        COUNT(DISTINCT AirportID) as airport_cnt
                                    FROM airport_per_state
                                    GROUP BY 1,2
                                    ORDER BY 3 DESC""")
airport_cnt_per_state_pd = airport_cnt_per_state.toPandas()
airport_cnt_per_state_gpd = gpd.GeoDataFrame(airport_cnt_per_state_pd, geometry='s_bound')
airport_cnt_per_state_gpd.head(10)
```

```
21/11/25 01:28:19 WARN JoinQuery: UseIndex is true, but no index exists. Will build index on the fly.
```

Out[16]:

	State	s_bound	airport_cnt
0	Alaska	POLYGON ((-141.02050 70.01870, -141.72910 70.1...	180
1	California	POLYGON ((-124.40090 41.99830, -123.62370 42.0...	114
2	Texas	POLYGON ((-106.57150 31.86590, -106.50420 31.7...	100
3	Florida	POLYGON ((-87.60500 30.99880, -86.56130 30.996...	87
4	Georgia	POLYGON ((-85.60820 34.99740, -84.72660 34.990...	57
5	Michigan	POLYGON ((-88.37130 48.30330, -87.60500 48.010...	42
6	North Carolina	POLYGON ((-78.48500 33.79630, -79.67420 34.803...	42
7	Kansas	POLYGON ((-102.05060 40.00340, -102.05060 40.0...	39
8	New York	POLYGON ((-79.76240 42.51420, -79.06720 42.778...	39
9	Wisconsin	POLYGON ((-90.63030 42.51160, -87.02410 42.495...	38

In [17]:

```python
airport_cnt_per_state_gpd.plot("airport_cnt", cmap = 'OrRd', figsize = (18,15), edgecolor = "0.8",
                               legend = True,
                               legend_kwds = {"label": "# of Airports","shrink": 0.4})
plt.axis("off")
plt.title("# of airports in each US state", fontsize = 20, weight = 'bold')
airport_cnt_per_state_gpd.apply(lambda x:
                                plt.annotate(
                                    text = x["State"],
                                    xy = x.s_bound.centroid.coords[0],
                                    ha="center",
                                    fontsize = 14) if x["airport_cnt"] > 50 else False, axis = 1)
plt.show()
```