MID-COURSE PROJECT

PROJECT DATA: US & CANADA RESORTS

```
resorts.head()
```

	ID	Resort	Latitude	Longitude	Country	Continent	Price	Season	Highest point	Lowest point	...	Snow cannons	Surface lifts	Chair lifts	Gondola lifts	Total lifts	Lift capacity
0	4	Red Mountain Resort-Rossland	49.105520	-117.846280	Canada	North America	60	December - April	2075	1185	...	0	2	5	1	8	9200
1	11	Fernie	49.504175	-115.062867	Canada	North America	67	December - April	2134	1052	...	11	3	7	0	10	14514
2	12	Sun Peaks	50.884468	-119.882329	Canada	North America	62	November - April	2082	1198	...	0	6	6	0	12	13895
3	13	Panorama	50.736999	-119.120561	Canada	North America	62	December - April	2365	1140	...	0	3	6	4	13	11890
4	22	Steamboat	35.754022	-109.853751	United States	North America	120	November - April	3221	2103	...	0	1	14	2	17	32720

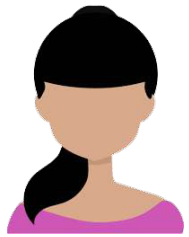
5 rows × 25 columns

```
resorts.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 98 entries, 0 to 97
Data columns (total 25 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   ID                    98 non-null     int64
 1   Resort                98 non-null     object
 2   Latitude              98 non-null     float64
 3   Longitude             98 non-null     float64
 4   Country               98 non-null     object
 5   Continent             98 non-null     object
 6   Price                 98 non-null     int64
 7   Season                98 non-null     object
 8   Highest point         98 non-null     int64
 9   Lowest point          98 non-null     int64
 10  Beginner slopes       98 non-null     int64
 11  Intermediate slopes   98 non-null     int64
 12  Difficult slopes      98 non-null     int64
 13  Total slopes          98 non-null     int64
 14  Longest run           98 non-null     int64
 15  Snow cannons          98 non-null     int64
 16  Surface lifts         98 non-null     int64
 17  Chair lifts           98 non-null     int64
 18  Gondola lifts         98 non-null     int64
 19  Total lifts           98 non-null     int64
 20  Lift capacity         98 non-null     int64
 21  Child friendly        98 non-null     object
 22  Snowparks             98 non-null     object
 23  Nightskiing           98 non-null     object
 24  Summer skiing         98 non-null     object
dtypes: float64(2), int64(15), object(8)
memory usage: 19.3+ KB
```

MAVELUXE

ASSIGNMENT: MIDCOURSE PROJECT



NEW MESSAGE

March 28, 2024

From: **Deepthi Downhill** *(VP of Analytics)*

Subject: **More Ambitious Ski Resort App**

Hello,

The work you've been doing with Leonard is very exciting. This type of application can save our agents hundreds of hours annually! I want to applaud you both on this amazing initiative.

That said, it's time to think a bit bigger. While Europe is a solid market, it's behind the US and Canada for us given our customers are almost exclusively from North America. Can you create two apps that will help us with these markets?

Thanks!

section04_midcourse_project.ipynb

Reply Forward

Key Objectives

1. Build two working Dash Applications
2. Add multiple chart types and interactive elements
3. Connect them with callback functions capable of taking multiple inputs and returning multiple outputs

MAVELUXE

FINAL PROJECT

PROJECT DATA: WORLDWIDE RESORTS

```
resorts.head()
```

	ID	Resort	Latitude	Longitude	Country	Continent	Price	Season	Highest point	Lowest point
0	1	Hemsedal	60.928244	8.383487	Norway	Europe	46	November - May	1450	620
1	2	Geilosiden Geilo	60.534526	8.206372	Norway	Europe	44	November - April	1178	800
2	3	Golm	47.057810	9.828167	Austria	Europe	48	December - April	2110	650
3	4	Red Mountain Resort-Rossland	49.105520	-117.846280	Canada	North America	60	December - April	2075	1185
4	5	Hafjell	61.230369	10.529014	Norway	Europe	45	November - April	1030	195

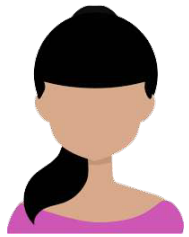
5 rows × 25 columns

MAVELUXE

```
resorts.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 499 entries, 0 to 498
Data columns (total 25 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   ID                    499 non-null    int64
 1   Resort                499 non-null    object
 2   Latitude              499 non-null    float64
 3   Longitude             499 non-null    float64
 4   Country               499 non-null    object
 5   Continent             499 non-null    object
 6   Price                 499 non-null    int64
 7   Season                499 non-null    object
 8   Highest point         499 non-null    int64
 9   Lowest point          499 non-null    int64
 10  Beginner slopes       499 non-null    int64
 11  Intermediate slopes   499 non-null    int64
 12  Difficult slopes      499 non-null    int64
 13  Total slopes          499 non-null    int64
 14  Longest run           499 non-null    int64
 15  Snow cannons          499 non-null    int64
 16  Surface lifts         499 non-null    int64
 17  Chair lifts           499 non-null    int64
 18  Gondola lifts         499 non-null    int64
 19  Total lifts           499 non-null    int64
 20  Lift capacity         499 non-null    int64
 21  Child friendly        499 non-null    object
 22  Snowparks             499 non-null    object
 23  Nightskiing           499 non-null    object
 24  Summer skiing         499 non-null    object
dtypes: float64(2), int64(15), object(8)
memory usage: 97.6+ KB
```

ASSIGNMENT: FINAL PROJECT



NEW MESSAGE

May 1, 2023

From: **Deepthi Downhill** *(VP of Analytics)*

Subject: **Even MORE Ambitious Resort App**

Hey, thanks for the great work on the two dashboards.

However, I'm getting some feedback that having two separate dashboards is challenging to navigate. Can you make this a single app, with each view on its own tab? Try to improve the design a bit as well.

We also want to think EVEN BIGGER. The US and Canada were a great start, but we have access to data on ski resorts world-wide, and we should be able to leverage much of our existing code to include all of them.

Thanks!

section07_final_project.ipynb

Reply Forward

Key Objectives

1. Build a multi-tab dashboard with a grid-based layout
2. Add multiple chart types and interactive elements
3. Write standard callback functions to connect them
4. Include a chained callback function and (if you're daring) a cross-filtering callback function

MAVELUXE