data-engineer-lgde-day3-answer

September 8, 2021

1 3.

plot.ly

[1]:
```
from pyspark.sql import *
from pyspark.sql.functions import *
from pyspark.sql.types import *
from IPython.display import display, display_pretty, clear_output, JSON

spark = (
    SparkSession
    .builder
    .config("spark.sql.session.timeZone", "Asia/Seoul")
    .getOrCreate()
)

#
spark.conf.set("spark.sql.repl.eagerEval.enabled", True) # display enabled
spark.conf.set("spark.sql.repl.eagerEval.truncate", 100) # display output␣
 ↪columns size

#
home_jovyan = "/home/jovyan"
work_data = f"{home_jovyan}/work/data"
work_dir=!pwd
work_dir = work_dir[0]
answer = "/answer"

#
spark.conf.set("spark.sql.shuffle.partitions", 5) # the number of partitions to␣
 ↪use when shuffling data for joins or aggregations.
spark.conf.set("spark.sql.streaming.forceDeleteTempCheckpointLocation", "true")
spark
```

```
21/09/08 13:54:48 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).
21/09/08 13:54:50 WARN Utils: Service 'SparkUI' could not bind on port 4040.
Attempting port 4041.
21/09/08 13:54:50 WARN Utils: Service 'SparkUI' could not bind on port 4041.
Attempting port 4042.
```

[1]:
```
<pyspark.sql.session.SparkSession at 0x7f51b86c9e50>
```

1.0.1 3

, Append .

[23]:
```python
# 2020-10-25 ~ 2020-11-03 :

today = "2020-10-27"
lgde_origin = spark.read.jdbc("jdbc:mysql://mysql:3306/testdb", "testdb.lgde",␣
↪properties={"user": "sqoop", "password": "sqoop"}).where(col("dt") <␣
↪lit(today))
lgde_today = (
    spark.createDataFrame(
        [
            ("2020-10-27", 30, 10, 10000000)
            , ("2020-10-28", 40, 25, 50000000)
            , ("2020-10-29", 100, 28, 100000000)
            , ("2020-10-30", 90, 25, 60000000)
            , ("2020-10-31", 150, 10, 160000000)
            , ("2020-11-01", 140, 13, 150000000)
            , ("2020-11-02", 180, 15, 180000000)
            , ("2020-11-03", 160, 12, 170000000)
        ], ["DT", "DAU", "PU", "DR"]
    )
)


lgde_union = lgde_origin.union(lgde_today)
lgde_local = lgde_union.collect()
lgde = spark.createDataFrame(lgde_local)
lgde.write.mode("overwrite").jdbc("jdbc:mysql://mysql:3306/testdb", "testdb.
↪lgde", properties={"user": "sqoop", "password": "sqoop"})
```

1.0.2 3-1.

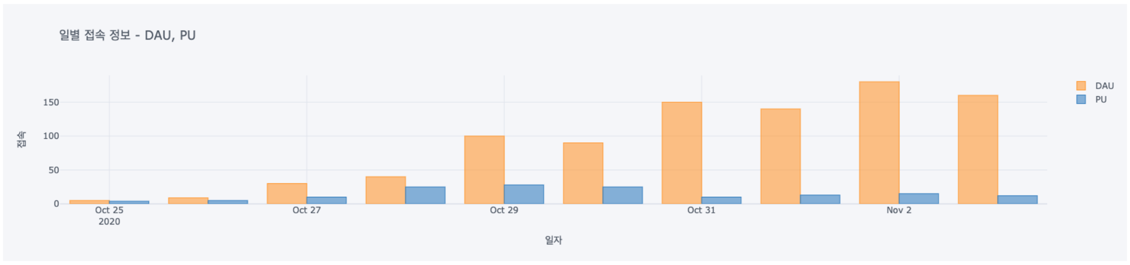
NoSQL , JDBC .

[19]:
```python
import chart_studio.plotly as py
import cufflinks as cf
cf.go_offline(connected=True)
```

```
raw = spark.read.jdbc("jdbc:mysql://mysql:3306/testdb", "testdb.lgde",␣
 ↪properties={"user": "sqoop", "password": "sqoop"}).orderBy(asc("dt"))
data = raw.withColumn("ARPU", expr("round(DR / DAU)")).withColumn("ARPPU",␣
 ↪expr("round(DR / PU)"))
```

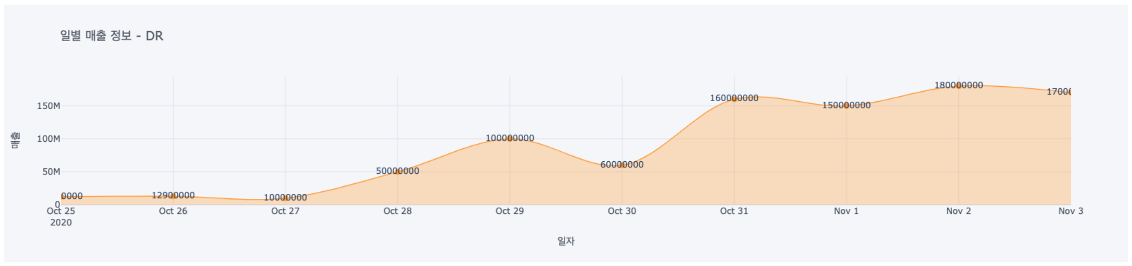
```
kine = { chart, scatter, bar, box, spread, ratio, heatmap, surface, histogram, bubble, bubble3c
dash = { key: value = { solid, dash, dashdot, dot } }
mode = { key: value = { lines, markers, lines+markers, lines+text, markers+text, lines+markers+
symbol = { key: value = { circle, circle-dot, diamond, square } } : mode
interpolation = { key: value = { linear, spline, vhv, hvh, vh, hv } }
```

[20]:
```
# DAU, PU -
_kind = 'bar'
_barmode = 'group'
_dash = {'DAU':'solid', 'PU':'solid'}
_mode = {'DAU':'lines+markers+text', 'PU':'lines+markers'}
_symbol = {'DAU':'square', 'PU':'circle'}
_interpolation = {'DAU':'spline', 'PU':'spline'}
_size = 8

users = data.withColumn("datetime", to_date(col('DT'), 'yyyy-MM-dd')).
 ↪drop("DT", "DR", "ARPU", "ARPPU")
pdUsers = users.toPandas().set_index('datetime')
pdUsers.iplot(kind=_kind, barmode=_barmode, title='       - DAU, PU',␣
 ↪xTitle=' ', yTitle=' ', fill=True, dash=_dash, mode=_mode, symbol=_symbol,␣
 ↪interpolation=_interpolation, size=_size)
```

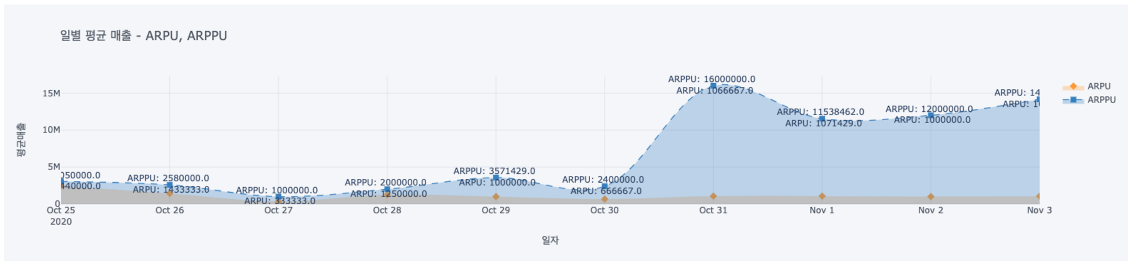


[21]:
```
@udf(returnType=StringType())
def addComma(column):
    return "{:,}".format(column)

# DR
_kind = 'line'
_dash = {'DR':'solid'}
_mode = {'DR':'lines+markers+text'}
_symbol = {'DR':'circle'}
```

```
_interpolation = {'DR':'spline'}
_size = 8
revenue = data.withColumn("datetime", to_date(col('DT'), 'yyyy-MM-dd')).
↪drop("DT", "DAU", "PU", "ARPU", "ARPPU")
pdRevenue = revenue.toPandas().set_index('datetime') # .withColumn("DRC",␣
↪addComma(col("DR")))
pdRevenue.iplot(kind=_kind, text='DR', title='      - DR', xTitle='  ',␣
↪yTitle='  ', fill=True, \
                dash=_dash, mode=_mode, symbol=_symbol,␣
↪interpolation=_interpolation, size=_size)
```



[22]:
```
# DR, ARPU, ARPPU -
_kind = 'line'
_dash = {'ARPU':'solid', 'ARPPU':'dash'}
_mode = {'ARPU':'markers', 'ARPPU':'lines+markers+text'}
_symbol = {'ARPU':'diamond', 'ARPPU':'square'}
_interpolation = {'ARPU':'spline', 'ARPPU':'spline'}
_size = 8

purchase = data.withColumn("datetime", to_date(col('DT'), 'yyyy-MM-dd')).
↪drop("DT", "DAU", "PU", "DR")
pdPurchase = purchase.toPandas().set_index('datetime')

arppu = list(pdPurchase['ARPPU'])
arpu = list(pdPurchase['ARPU'])
_text = [f'ARPPU: {x} <br> ARPU: {y}' for x,y in list(zip(arppu, arpu))]

pdPurchase.iplot(kind='line', text=_text, title='      - ARPU, ARPPU',␣
↪xTitle='  ', yTitle='   ', fill=True, \
                 dash=_dash, mode=_mode, symbol=_symbol,␣
↪interpolation=_interpolation, size=_size)
```



[13]:
```
help(pdPurchase.iplot)
```

```
Help on method _iplot in module cufflinks.plotlytools:

_iplot(kind='scatter', data=None, layout=None, filename='', sharing=None,
title='', xTitle='', yTitle='', zTitle='', theme=None, colors=None,
colorscale=None, fill=False, width=None, dash='solid', mode='',
interpolation='linear', symbol='circle', size=12, barmode='', sortbars=False,
bargap=None, bargroupgap=None, bins=None, histnorm='', histfunc='count',
orientation='v', boxpoints=False, annotations=None, keys=False, bestfit=False,
bestfit_colors=None, mean=False, mean_colors=None, categories='', x='', y='',
z='', text='', gridcolor=None, zerolinecolor=None, margin=None, labels=None,
values=None, secondary_y='', secondary_y_title='', subplots=False, shape=None,
error_x=None, error_y=None, error_type='data', locations=None, lon=None,
lat=None, asFrame=False, asDates=False, asFigure=False, asImage=False,
dimensions=None, asPlot=False, asUrl=False, online=None, **kwargs) method of
pandas.core.frame.DataFrame instance
           Returns a plotly chart either as inline chart, image of Figure object

           Parameters:
           -----------
                   kind : string
                           Kind of chart
                                   scatter
                                   bar
                                   box
                                   spread
                                   ratio
                                   heatmap
                                   surface
                                   histogram
                                   bubble
                                   bubble3d
                                   scatter3d
                                   scattergeo
                                   ohlc
                                   candle
```

```
                                pie
                                choroplet
                data : Data
                        Plotly Data Object.
                        If not entered then the Data object will be
automatically
                        generated from the DataFrame.
                layout : Layout
                        Plotly layout Object
                        If not entered then the Layout objet will be
automatically
                        generated from the DataFrame.
                filename : string
                        Filename to be saved as in plotly account
                sharing : string
                        Sets the sharing level permission
                                public - anyone can see this chart
                                private - only you can see this chart
                                secret - only people with the link can see
the chart
                title : string
                        Chart Title
                xTitle : string
                        X Axis Title
                yTitle : string
                        Y Axis Title
                                zTitle : string
                zTitle : string
                        Z Axis Title
                        Applicable only for 3d charts
                theme : string
                        Layout Theme
                                solar
                                pearl
                                white
                        see cufflinks.getThemes() for all
                        available themes
                colors : dict, list or string
                        {key:color} to specify the color for each column
                        [colors] to use the colors in the defined order
                colorscale : string
                        Color scale name
                        If the color name is preceded by a minus (-)
                        then the scale is inversed
                        Only valid if 'colors' is null
                        See cufflinks.colors.scales() for available scales
                fill : bool
                        Filled Traces
```

```
width : dict, list or int
                int : applies to all traces
                list : applies to each trace in the order
                                specified
                dict: {column:value} for each column in
                                the dataframe
        Line width
dash : dict, list or string
                string : applies to all traces
                list : applies to each trace in the order
                                specified
                dict: {column:value} for each column in
                                the dataframe
        Drawing style of lines
                solid
                dash
                dashdot
                dot
mode : dict, list or string
                string : applies to all traces
                list : applies to each trace in the order
                                specified
                dict: {column:value} for each column in
                                the dataframe
        Plotting mode for scatter trace
                lines
                markers
                lines+markers
                lines+text
                markers+text
                lines+markers+text
interpolation : dict, list, or string
                string : applies to all traces
                list : applies to each trace in the order
                                specified
                dict: {column:value} for each column in
                                the dataframe
        Positioning of the connecting lines
                linear
                spline
                vhv
                hvh
                vh
                hv
symbol : dict, list or string
                string : applies to all traces
                list : applies to each trace in the order
                                specified
                                dict: {column:value} for each column in
                                                the dataframe
                        The symbol that is drawn on the plot for each marker
                        Valid only when mode includes markers
                                circle
                                circle-dot
                                diamond
                                square
                                and many more…(see
plotly.validators.scatter.marker.SymbolValidator.values)
                size : string or int
                        Size of marker
                        Valid only if marker in mode
                barmode : string
                        Mode when displaying bars
                                group
                                stack
                                overlay
                        * Only valid when kind='bar'
                sortbars : bool
                        Sort bars in descending order
                        * Only valid when kind='bar'
                bargap : float
                        Sets the gap between bars
                                [0,1)
                        * Only valid when kind is 'histogram' or 'bar'
                bargroupgap : float
                        Set the gap between groups
                                [0,1)
                        * Only valid when kind is 'histogram' or 'bar'
                bins : int or tuple
                        if int:
                                Specifies the number of bins
                        if tuple:
                                (start, end, size)
                                start : starting value
                                end: end value
                                size: bin size
                        * Only valid when kind='histogram'

                histnorm : string
                                '' (frequency)
                                percent
                                probability
                                density
                                probability density
                        Sets the type of normalization for an histogram
trace. By default
```

```
                           the height of each bar displays the frequency of
occurrence, i.e.,
                           the number of times this value was found in the
                           corresponding bin. If set to 'percent', the height of
each bar
                           displays the percentage of total occurrences found
within the
                           corresponding bin. If set to 'probability', the
height of each bar
                           displays the probability that an event will fall into
the
                           corresponding bin. If set to 'density', the height of
each bar is
                           equal to the number of occurrences in a bin divided
by the size of
                           the bin interval such that summing the area of all
bins will yield
                           the total number of occurrences. If set to
'probability density',
                           the height of each bar is equal to the number of
probability that an
                           event will fall into the corresponding bin divided by
the size of
                           the bin interval such that summing the area of all
bins will yield
                           1.
                           * Only valid when kind='histogram'
                   histfunc : string
                                   count
                                   sum
                                   avg
                                   min
                                   max
                      Sets the binning function used for an histogram trace.
                           * Only valid when kind='histogram'
                   orientation : string
                                   h
                                   v
                           Sets the orientation of the bars. If set to 'v', the
length of each
    |          bar will run vertically. If set to 'h', the length of each bar
will
    |          run horizontally
                           * Only valid when kind is 'histogram','bar' or 'box'
                   boxpoints : string
                           Displays data points in a box plot
                                   outliers
                                   all
```

```
                suspectedoutliers
                False
annotations : dictionary
        Dictionary of annotations
        {x_point : text}
keys : list of columns
        List of columns to chart.
        Also can be used for custom sorting.
bestfit : boolean or list
        If True then a best fit line will be generated for
        all columns.
        If list then a best fit line will be generated for
        each key on the list.
bestfit_colors : list or dict
        {key:color} to specify the color for each column
        [colors] to use the colors in the defined order
categories : string
        Name of the column that contains the categories
x : string
        Name of the column that contains the x axis values
y : string
        Name of the column that contains the y axis values
z : string
        Name of the column that contains the z axis values
text : string
        Name of the column that contains the text values
gridcolor : string
        Grid color
zerolinecolor : string
        Zero line color
margin : dict or tuple
        Dictionary (l,r,b,t) or
        Tuple containing the left,
        right, bottom and top margins
labels : string
        Name of the column that contains the labels.
        * Only valid when kind='pie'
values : string
        Name of the column that contains the values.
        * Only valid when kind='pie'
secondary_y : string or list(string)
        Name(s) of the column to be charted on the
        right hand side axis
secondary_y_title : string
        Title of the secondary axis
subplots : bool
        If true then each trace is placed in
        subplot layout
```

```
                   shape : (rows,cols)
                           Tuple indicating the size of rows and columns
                           If omitted then the layout is automatically set
                           * Only valid when subplots=True
                   error_x : int or float or [int or float]
                           error values for the x axis
                   error_y : int or float or [int or float]
                           error values for the y axis
                   error_type : string
                           type of error bars
                                   'data'
                                   'constant'
                                   'percent'
                                   'sqrt'
                                   'continuous'
                                   'continuous_percent'
                   asFrame : bool
                           If true then the data component of Figure will
                           be of Pandas form (Series) otherwise they will
                           be index values
                   asDates : bool
                           If true it truncates times from a DatetimeIndex
                   asFigure : bool
                           If True returns plotly Figure
                   asImage : bool
                           If True it returns an Image (png)
                           In ONLINE mode:
                                   Image file is saved in the working directory
                                           Accepts:
                                                   filename
                                                   dimensions
                                                   scale
                                                   display_image
                           In OFFLINE mode:
                                   Image file is downloaded (downloads folder)
and a
                                   regular plotly chart is displayed in Jupyter
                                           Accepts:
                                                   filename
                                                   dimensions
                   dimensions : tuple(int,int)
                           Dimensions for image / chart
                                   (width,height)
                   asPlot : bool
                           If True the chart opens in browser
                   asUrl : bool
                           If True the chart url/path is returned. No chart is
displayed.
```

```
                        If Online : the URL is returned
                        If Offline : the local path is returned
                online : bool
                        If True then the chart/image is rendered on the
server
                        even when running in offline mode.

                Other Kwargs
                ============
                Line, Scatter
                        connectgaps : bool
                                If True, empty values are connected
                Pie charts
                        sort : bool
                                If True it sorts the labels by value
                        pull : float [0-1]
                                Pulls the slices from the centre
                        hole : float [0-1]
                                Sets the size of the inner hole
                        linecolor : string
                                Sets the color for the contour line of the
slices
                        linewidth : string
                                Sets the width for the contour line of the
slices
                        textcolor : string
                                Sets the color for the text in the slices
                        textposition : string
                                Sets the position of the legends for each
slice
                                        outside
                                        inner
                        textinfo : string
                                Sets the information to be displayed on
                                the legends
                                        label
                                        percent
                                        value
                                        * or ony combination of the above
using
                                          '+' between each item
                                          ie 'label+percent'

                Histogram
                        linecolor : string
                                specifies the line color of the histogram

                Heatmap and Surface
                center_scale : float
                        Centers the colorscale at a specific value
                        Automatically sets the (zmin,zmax) values
                zmin : float
                        Defines the minimum range for the z values.
                        This affects the range for the colorscale
                zmax : float
                        Defines the maximum range for the z values.
                        This affects the range for the colorscale

        Error Bars
                error_trace : string
                        Name of the column for which error should be
                        plotted. If omitted then errors apply to all
                        traces.
                error_values_minus : int or float or [int or float]
                        Values corresponding to the span of the error
bars
                        below the trace coordinates
                error_color : string
                        Color for error bars
                error_thickness : float
                        Sets the line thickness of the error bars
                error_width :  float
                        Sets the width (in pixels) of the cross-bar
at both
                        ends of the error bars
                error_opacity : float [0,1]
                        Opacity for the error bars

        Subplots
                horizontal_spacing : float [0,1]
                        Space between subplot columns.
                vertical_spacing : float [0,1]
                        Space between subplot rows.
                subplot_titles : bool
                        If True, chart titles are plotted
                        at the top of each subplot
                shared_xaxes : bool
                        Assign shared x axes.
                        If True, subplots in the same grid column
have one common
                        shared x-axis at the bottom of the grid.
                shared_yaxes : bool
                        Assign shared y axes.
                        If True, subplots in the same grid row have
one common
                        shared y-axis on the left-hand side of the
```

```
grid.

                   Shapes
                           hline : float, list or dict
                                   Draws a horizontal line at the
                                   indicated y position(s)
                                   Extra parameters can be passed in
                                   the form of a dictionary (see shapes)
                           vline : float, list or dict
                                   Draws a vertical line at the
                                   indicated x position(s)
                                   Extra parameters can be passed in
                                   the form of a dictionary (see shapes)
                           hpsan : (y0,y1)
                                   Draws a horizontal rectangle at the
                                   indicated (y0,y1) positions.
                                   Extra parameters can be passed in
                                   the form of a dictionary (see shapes)
                           vspan : (x0,x1)
                                   Draws a vertical rectangle at the
                                   indicated (x0,x1) positions.
                                   Extra parameters can be passed in
                                   the form of a dictionary (see shapes)
                           shapes : dict or list(dict)
                                   List of dictionaries with the
                                   specifications of a given shape.
                                   See help(cufflinks.tools.get_shape)
                                   for more information

                   Axis Ranges
                           xrange : [lower_bound,upper_bound]
                                   Sets the range for the x axis
                           yrange : [lower_bound,upper_bound]
                                   Sets the range for the y axis
                           zrange : [lower_bound,upper_bound]
                                   Sets the range for the z axis

                   Explicit Layout Updates
                           layout_update : dict
                                   The layout will be modified with all
                                   the explicit values stated in the
                                   dictionary.
                                   It will not apply if layout is passed
                                   as parameter.


                   Range Selector
                           rangeselector : dict
```

```
                                   Defines a rangeselector object
                                   see help(cf.tools.get_range_selector) for
more information
                                   Example:
                                           {'steps':['1y','2 months','5
weeks','ytd','2mtd'],
                                            'axis':'xaxis', 'bgcolor' :
('blue',.3),
                                            'x': 0.2 , 'y' : 0.9}

                   Range Slider
                           rangeslider : bool or dict
                                   Defines if a rangeslider is displayed
                                   If bool:
                                           True : Makes it visible
                                   if dict:
                                           Rangeslider object
                                   Example:
{'bgcolor':('blue',.3),'autorange':True}

                   Annotations
                           fontcolor : str
                                   Text color for annotations
                           fontsize : int
                                   Text size for annotations
                           textangle : int
                                   Text angle
                           See https://plot.ly/python/reference/#layout-
annotations
                           for a complete list of valid parameters.

                   Exports
                           display_image : bool
                                   If True then the image if displayed after
being saved
                                   ** only valid if asImage=True
                           scale : integer
                                   Increase the resolution of the image by
`scale` amount
                                   Only valid when asImage=True
```

1.0.3 3-2.

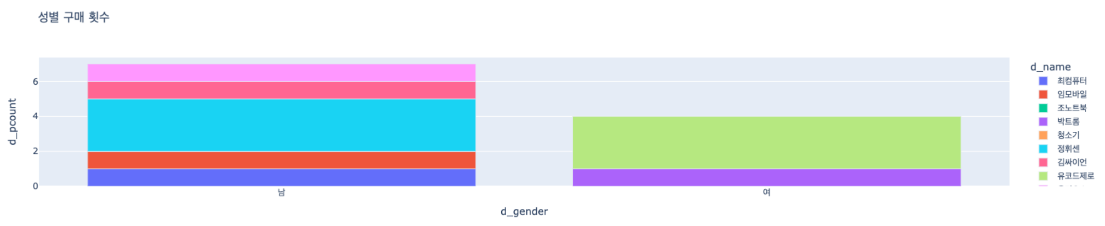
[9]:
```
dimension = spark.read.parquet(f"{work_dir}{answer}/dimension/dt=20201026")
display(dimension)
```

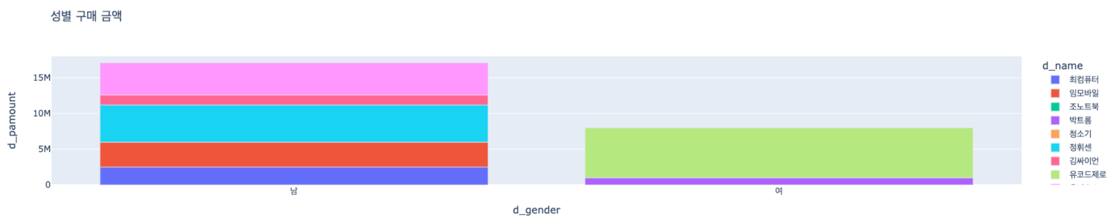
```
+-----+----------+--------+--------+---------+--------+-------------------+
|d_uid|    d_name|d_gender|d_acount|d_pamount|d_pcount|   d_first_purchase|
+-----+----------+--------+--------+---------+--------+-------------------+
|    9|     |       |       1|  2500000|       1|2020-10-26 07:49:15|
|    7|     |       |       1|  3500000|       1|2020-10-26 07:45:55|
|    8|     |       |       1|        0|       0|               null|
|    3|       |       |       3|  1000000|       1|2020-10-25 05:42:35|
|    4|       |       |       5|        0|       0|               null|
|    1|       |       |       3|  5200000|       3|2020-10-25 05:32:30|
|    2|     |       |       5|  1400000|       1|2020-10-25 11:38:20|
|    5|   |       |       3|  7000000|       3|2020-10-25 09:32:35|
|    6|     |       |       1|  4500000|       1|2020-10-26 10:08:20|
+-----+----------+--------+--------+---------+--------+-------------------+
```

[10]:
```
gender = dimension.select("d_name", "d_gender", "d_pcount", "d_pamount").
 ↪toPandas()
display(gender)
```

```
  d_name d_gender  d_pcount  d_pamount
0                         1    2500000
1                         1    3500000
2                         0          0
3                          1    1000000
4                          0          0
5                          3    5200000
6                         1    1400000
7                        3    7000000
8                         1    4500000
```

[11]:
```
import plotly.express as px

fig = px.bar(gender, x="d_gender", y="d_pcount", color="d_name", title="      ")
fig.show()

fig = px.bar(gender, x="d_gender", y="d_pamount", color="d_name", title="      ")
fig.show()
```



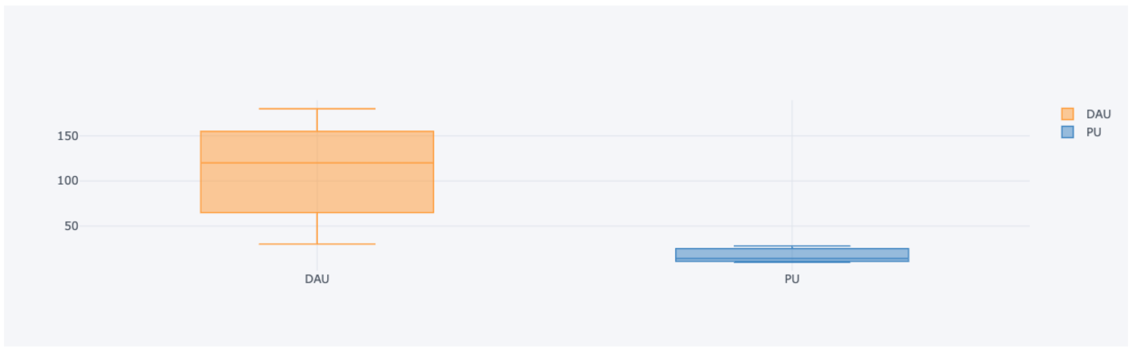


1.0.4 3-3.

1.0.5

- plot.ly bar-charts
- plot.ly line-charts
- plot.ly express
- plot.ly w/ apache spark

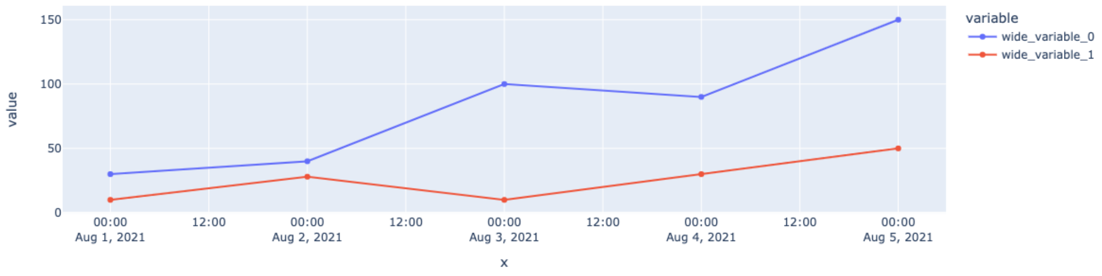
[12]:
```python
pdUsers.iplot(kind='box')
```



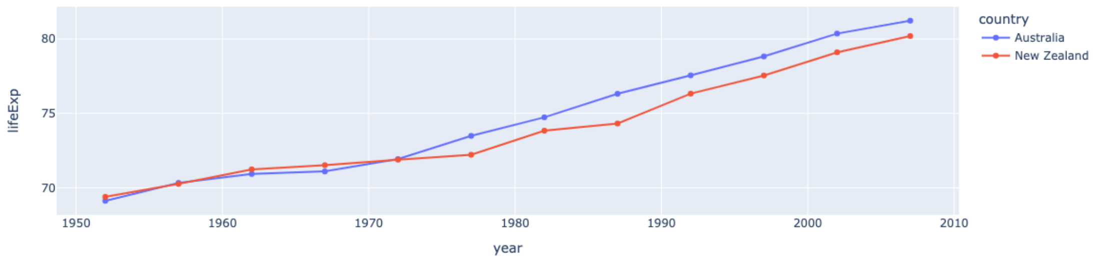
[13]:
```python
import numpy as np
users = spark.createDataFrame([("2021-08-01", 30, 10), ("2021-08-02", 40, 28),␣
↪("2021-08-03", 100, 10), ("2021-08-04", 90, 30), ("2021-08-05", 150, 50)],␣
↪["DATE", "DAU", "DPU"])
users.show(truncate=False)

dt = np.array(users.select("DATE").collect()).reshape(-1)
y1 = np.array(users.select("DAU").collect()).reshape(-1)
y2 = np.array(users.select("DPU").collect()).reshape(-1)
```

```
+----------+---+---+
|DATE      |DAU|DPU|
+----------+---+---+
|2021-08-01|30 |10 |
|2021-08-02|40 |28 |
|2021-08-03|100|10 |
|2021-08-04|90 |30 |
|2021-08-05|150|50 |
+----------+---+---+
```

[14]:
```
fig = px.line(x=dt, y=[y1, y2], markers=True)
fig.update_traces()
fig.show()
```



[15]:
```
import plotly.express as px
df = px.data.gapminder().query("continent == 'Oceania'")
fig = px.line(df, x='year', y='lifeExp', color='country', markers=True)
fig.show()
```



[16]:
```
import chart_studio.plotly as py
import cufflinks as cf
cf.go_offline(connected=True)
```

[17]:
```
pdUsers.head()
```

[17]:
```
            DAU  PU
datetime
2021-08-01   30  10
2021-08-02   40  25
2021-08-03  100  28
2021-08-04   90  25
2021-08-05  150  10
```

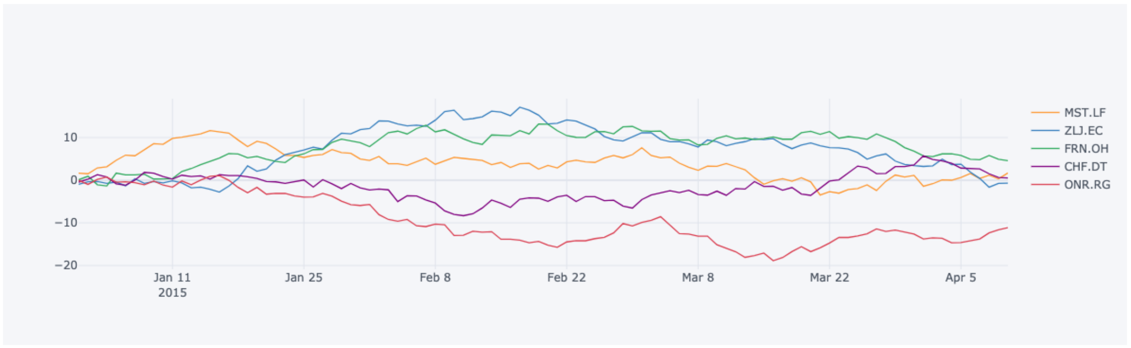
[18]:
```
df = cf.datagen.lines()
df.head()
```

[18]:
```
              MST.LF    ZLJ.EC    FRN.OH    CHF.DT    ONR.RG
2015-01-01  1.646000 -0.957035  0.109382 -0.448231 -0.037617
2015-01-02  1.509426 -0.337004  0.927018  0.271396 -0.979886
2015-01-03  2.886446 -0.319585 -1.052523  1.322693  0.181529
2015-01-04  3.157752 -0.720623 -1.347637  0.746635  0.743181
2015-01-05  4.664085 -0.446313  1.668729 -0.862617 -0.459035
```

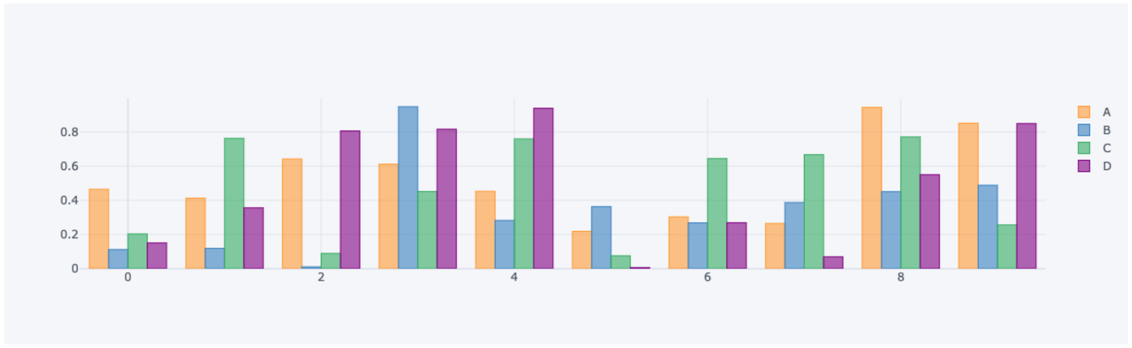
[19]:
```
print(df)
```

```
              MST.LF    ZLJ.EC    FRN.OH    CHF.DT     ONR.RG
2015-01-01  1.646000 -0.957035  0.109382 -0.448231  -0.037617
2015-01-02  1.509426 -0.337004  0.927018  0.271396  -0.979886
2015-01-03  2.886446 -0.319585 -1.052523  1.322693   0.181529
2015-01-04  3.157752 -0.720623 -1.347637  0.746635   0.743181
2015-01-05  4.664085 -0.446313  1.668729 -0.862617  -0.459035
…                …         …         …         …          …
2015-04-06  1.566699  1.875156  4.898736  2.770628 -14.276211
2015-04-07  0.413490  0.427677  4.816949  2.645884 -13.765856
2015-04-08  1.175292 -1.631982  5.784717  1.501562 -12.347416
2015-04-09  0.338442 -0.744472  4.901484  0.640499 -11.631579
2015-04-10  1.689080 -0.673680  4.580768  0.527162 -11.130009

[100 rows x 5 columns]
```

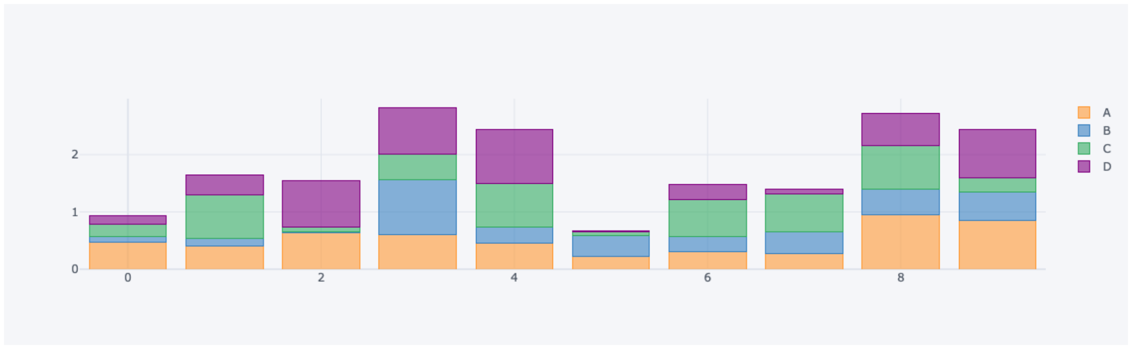
[20]:
```
df.iplot(kind='line')
```



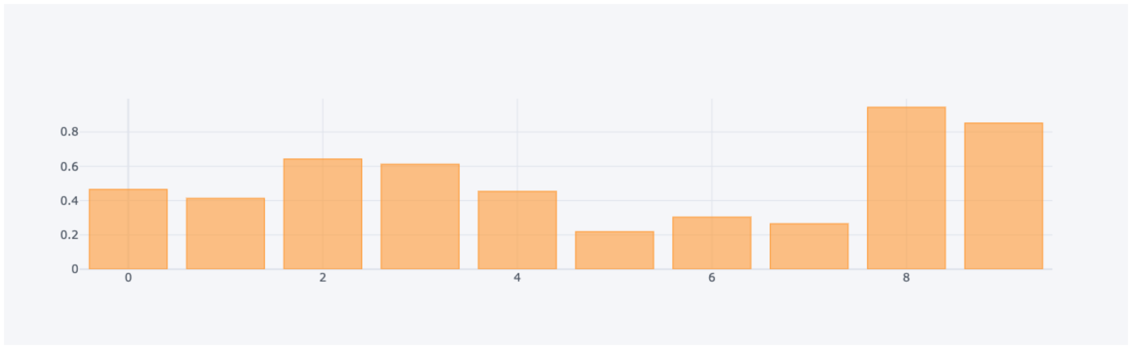
[21]:
```python
import numpy as np
import pandas as pd
df = pd.DataFrame(np.random.rand(10, 4), columns=['A', 'B', 'C', 'D'])
df.head()
df.iplot(kind='bar')
```



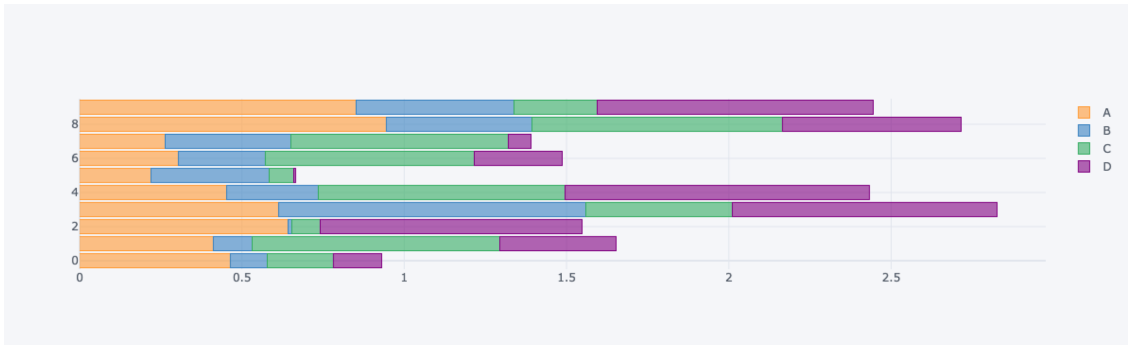
[22]:
```python
df.iplot(kind='bar', barmode='stack')
```



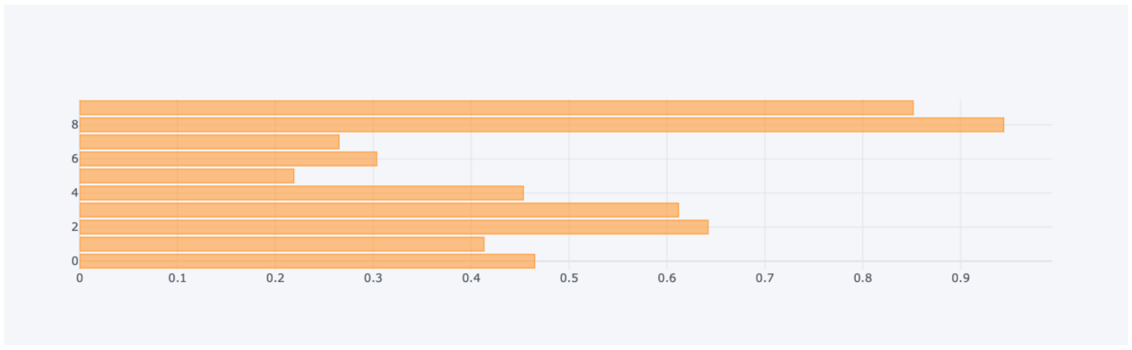
[23]:
```python
df['A'].iplot(kind='bar')
```



[24]:
```
df.iplot(kind='barh', barmode='stack')
```



[25]:
```
df['A'].iplot(kind='barh')
```



[ ]:
```

```