Unit 5
Chapter 2: Organize and Report Supersteps:

Organize Superstep:

- The Organize superstep takes the complete data warehouse you built at the end of the Transform superstep and subsections it into business-specific data marts.
- A data mart is the access layer of the data warehouse environment built to expose data to the users.
- The data mart is a subset of the data warehouse and is generally oriented to a specific business group.

1) Horizontal Style:

- Performing horizontal-style slicing or subsetting of the data warehouse is achieved by applying a filter technique that forces the data warehouse to show only the data for a specific preselected set of filtered outcomes against the data population.
- The horizontal-style slicing selects the subset of rows from the population while preserving the columns.
- That is, the data science tool can see the complete record for the records in the subset of records.
- Ex:
- sSQL="SELECT * FROM [Dim-BMI] WHERE Height > 1.5 and Indicator = 1 ORDER BY Height, Weight;"
- PersonFrame1=pd.read_sql_query(sSQL, conn1)

- **2) Vertical Style:**
- Performing vertical-style slicing or subsetting of the data warehouse is achieved by applying a filter technique that forces the data warehouse to show only the data for specific preselected filtered outcomes against the data population.
- The vertical-style slicing selects the subset of columns from the population, while preserving the rows.
- That is, the data science tool can see only the preselected columns from a record for all the records in the population.
- Ex:
- sSQL="SELECT Height, Weight, Indicator FROM [Dim-BMI];"
- PersonFrame1=pd.read_sql_query(sSQL, conn1)

- **3) Island Style:**
- Performing island-style slicing or subsetting of the data warehouse is achieved by applying a combination of horizontal- and vertical-style slicing.
- This generates a subset of specific rows and specific columns reduced at the same time.
- The technique generates a set of data islands that are specific to particular requirements within the business.
- Ex:
- sSQL="SELECT Height,Weight,Indicator FROM [Dim-BMI]
- WHERE Indicator > 2 ORDER BY Height,Weight;"
- PersonFrame1=pd.read_sql_query(sSQL, conn1)

4) Secure Vault Style:

- The secure vault is a version of one of the horizontal, vertical, or island slicing techniques, but the outcome is also attached to the person who performs the query.
- This is common in multi-security environments, where different users are allowed to see different data sets.
- This process works well, if you use a role-based access control (RBAC) approach to restricting system access to authorized users.
- The security is applied against the "role," and a person can then, by the security system, simply be added or removed from the role, to enable or disable access.

- It is also possible to use a time-bound RBAC that has different access rights during office hours than after hours.
- Ex:
- sSQL="SELECT Height,Weight,Indicator,
- CASE Indicator
- WHEN 1 THEN 'Pip'
- WHEN 2 THEN 'Norman'
- WHEN 3 THEN 'Grant'
- ELSE 'Sam'
- END AS Name
- FROM [Dim-BMI]
- WHERE Indicator > 2
- ORDER BY Height, Weight;"
- PersonFrame1=pd.read_sql_query(sSQL, conn1)
- sSQL="SELECT * FROM [Dim-BMI-Secure] WHERE Name = 'Sam';"
- PersonFrame2=pd.read_sql_query(sSQL, conn2)

- **5) Association Rule Mining:**
- Association rule learning is a rule-based machine-learning method for discovering interesting relations between variables in large databases, similar to the data you will find in a data lake.
- The technique enables you to investigate the interaction between data within the same population.
- Ex: "market basket analysis."
- It will investigate the analysis of a customer's purchases during a period of time.
- The new measure you need to understand is called "lift."
- Lift is simply estimated by the ratio of the joint probability of two items x and y, divided by the product of their individual probabilities:
- $$Lift = \frac{P(x,y)}{P(x)P(y)}$$
- It proceeds by identifying the frequent individual items in the data lake and extends them to larger and larger item sets, as long as those item sets appear satisfactorily frequently in the data lake.
- The frequent item sets determined by Apriori can be used to determine association rules that highlight common trends in the overall data lake.
- Ex: formulate the basket's simulation in your model.
- basket = (df[df['Country'] =="France"]
- .groupby(['InvoiceNo', 'Description'])['Quantity']
- .sum().unstack().reset_index().fillna(0)
- .set_index('InvoiceNo'))
- def encode_units(x):
- if x <= 0:
- return 0
- if x >= 1:
- return 1
-
- basket_sets = basket.applymap(encode_units)
- # Apply the Apriori algorithm to the data model.
- frequent_itemsets = apriori(basket_sets, min_support=0.07, use_colnames=True)
- rules = association_rules(frequent_itemsets, metric="lift", min_
- threshold=1)
- print(rules.head())
- rules[(rules['lift'] >= 6) &
- (rules['confidence'] >= 0.8)]

- sProduct1='ALARM CLOCK BAKELIKE GREEN'
- print(sProduct1)
- print(basket[sProduct1].sum())
-
- sProduct2='ALARM CLOCK BAKELIKE RED'
- print(sProduct2)
- print(basket[sProduct2].sum())

- **Report Superstep:**
- The Report superstep is the step in the ecosystem that enhances the data science findings with the art of storytelling and data visualization.
- You can perform the best data science, but if you cannot execute a respectable and trustworthy Report step by turning your data science into actionable business insights, you have achieved no advantage for your business.
- **Summary of the Results:**
- Your data science techniques and algorithms can produce the most methodically, most advanced mathematical or most specific statistical results to the requirements, but if you cannot summarize those into a good story, you have not achieved your requirements.
- **1) Understand the Context:**
- What differentiates good data scientists from the best data scientists are not the algorithms or data engineering; it is the ability of the data scientist to apply the context of his findings to the customer.
- **2) Appropriate Visualization:**
- It is true that a picture tells a thousand words. But in data science, you only want your visualizations to tell one story: the findings of the data science you prepared.
- It is absolutely necessity to ensure that your audience will get your most important message clearly and without any other meanings.
- Practice with your visual tools and achieve a high level of proficiency.
- **3) Eliminate Clutter:**
- There are various algorithms to eliminate dimensions and eliminate missing values, decision trees to subdivide but the biggest contributor to eliminating clutter is good and solid feature engineering.
- On average, it takes five minutes to cover one slide.
- You should never present more than ten slides.
- If you do not need it, lose it! Applying appropriate feature engineering wins every time.
- **4) Draw Attention Where You Want It:**
- Remember: Your purpose as a data scientist is to deliver insights to your customer, so that they can implement solutions to resolve a problem they may not even know about.
- You must place the attention on the insight and not the process.
- **5) Telling a Story (Freytag's Pyramid):**
- Under Freytag's pyramid, the plot of a story consists of five parts: exposition, rising action, climax, falling action, and resolution.
- This is used by writers of books and screenplays as the basic framework of any story.
- In the same way, you must take your business through the data science process.
- Exposition is the portion of a story that introduces important background information to the audience.
- In data science, you tell the background of the investigation you performed.
- Rising action refers to a series of events that build toward the point of greatest interest.
- In data science, you point out the important findings or results.
- Keep it simple and to the point.
- The climax is the turning point that determines a good or bad outcome for the story's characters.
- In data science, you show how your solution or findings will change the outcome of the work you performed.
- During the falling action, the conflict between what occurred before and after the climax takes place.

- In data science, you prove that after your suggestion has been implemented in a pilot, the same techniques can be used to find the issues now proving that the issues can inevitably be resolved.
- Resolution is the outcome of the story.
- In data science, you produce the solution and make the improvements permanent.
- Practice your storytelling with friends and family.
- Get them to give you feedback.
- Make sure you are comfortable speaking in front of people.
-
- **Graphics:**
- **Plot Options:**
- **ecosystem:**
- **data=[**
- **['London', 29.2, 17.4],**
- **['Glasgow', 18.8, 11.3],**
- **['Cape Town', 15.3, 9.0],**
- **['Houston', 22.0, 7.8],**
- **['Perth', 18.0, 23.7],**
- **['San Francisco', 11.4, 33.3]**
- **]**
- **os_new=pd.DataFrame(data)**
- **pd.Index(['Item', 'Value', 'Value Percent', 'Conversions', 'Conversion**
- **Percent',**
- **'URL', 'Stats URL'],**
- **dtype='object')**
- **os_new.rename(columns = {0 : "Warehouse Location"}, inplace=True)**
- **os_new.rename(columns = {1 : "Profit 2016"}, inplace=True)**
- **os_new.rename(columns = {2 : "Profit 2017"}, inplace=True)**
-
- **1) Pie Graph:**
- explode = (0, 0, 0, 0, 0, 0.1)
- labels=os_new['Warehouse Location']
- colors_mine = ['yellowgreen', 'gold', 'lightskyblue', 'lightcoral', 'lightcyan','lightblue']
- os_new.plot(figsize=(10, 10),kind="pie", y="Profit 2017",autopct='%.2f%%', \ shadow=True, explode=explode, legend = False, colors = colors_mine,\ labels=labels, fontsize=20)
-
- **2) Double Pie:**
- explode = (0, 0, 0, 0, 0, 0)
- colors_mine = ['yellowgreen', 'gold', 'lightskyblue', 'lightcoral', 'lightcyan','lightblue']
- os_new.plot(figsize=(10, 5),kind="pie", y=['Profit 2016','Profit 2017'],autopct='%.2f%%', \
- shadow=True, explode=explode, legend = False, colors = colors_mine,\
- subplots=True, labels=labels, fontsize=10)
-
- **3) Line Graph:**
- os_new.iloc[:5].plot(figsize=(10, 10),kind='Line',x='Warehouse Location',\
- y=['Profit 2016','Profit 2017']);
-
- **4) Bar Graph:**
- os_new.iloc[:5].plot(figsize=(10, 10),kind='bar',x='Warehouse Location',\
- y=['Profit 2016','Profit 2017']);
-
- **5) Horizontal Bar Graph:**
- os_new.iloc[:5].plot(figsize=(10, 10),kind='barh',x='Warehouse Location',\
- y=['Profit 2016','Profit 2017']);
-

- **6) Area Graph:**
- os_new.iloc[:5].plot(figsize=(10, 10),kind='area',x='Warehouse Location',\
- y=['Profit 2016','Profit 2017'],stacked=False);
-
- **7) Scatter Graph:**
- os_new.iloc[:5].plot(figsize=(10, 10),kind='scatter',x='Profit 2016',\
- y='Profit 2017',color='DarkBlue',marker='D');
-
- **8) Hex Bin Graph:**
- if you can perform the data science, you can
- • Easily judge distances, owing to the built-in scale
- • Easily judge the density of points of interest, also owing to the built-in
- scale
- • Easily scale the map up or down and put things roughly where they
- need to be for geospatial reporting formats
- os_new.iloc[:5].plot(figsize=(13, 10),kind='hexbin',x='Profit 2016',\
- y='Profit 2017', gridsize=25);
-
- **Pictures:**
- Pictures are an interesting data science specialty.
- The processing of movies and pictures is a science on its own.
- **1)Channels of Images:**
- The interesting fact about any picture is that it is a complex data set in every image.
- Pictures are built using many layers or channels that assists the visualization tools to render the required image.
- sPicName=Base+'/01-Vermeulen/00-RawData/AudiR8.png'
- t=0
- img=mpimg.imread(sPicName)
- print('Size:', img.shape)
- plt.figure(figsize=(10, 10))
- t+=1
- sTitle= '(' + str(t) + ') Original'
- plt.title(sTitle)
- plt.imshow(img)
- plt.show()
- for c in range(img.shape[2]):
- t+=1
- plt.figure(figsize=(10, 10))
- sTitle= '(' + str(t) + ') Channel: ' + str(c)
- plt.title(sTitle)
- lum_img = img[:,:,c]
- plt.imshow(lum_img)
- plt.show()
-
- **2) Cutting the Edge:**
- One of the most common techniques that most data science projects require is the determination of the edge of an item’s image.
- This is useful in areas such as robotics object selection and face recognition.
- imageIn = Image.open(sPicNameIn)
- fig1=plt.figure(figsize=(10, 10))
- mask=imageIn.convert("L")
- th=49
- imageOut = mask.point(lambda i: i < th and 255)
-
- **3) One Size Does Not Fit All:**
- The images we get to process are mostly of different sizes and quality.
- You will have to size images to specific sizes for most of your data science.

- (what happens to an image if you reduce the pixel quality.)
- img = Image.open(sPicName)
- plt.figure(figsize=(nSize, nSize))
- sTitle='Unchanges'
- plt.title(sTitle)
- imgplot = plt.imshow(img)
-
- img.thumbnail((64, 64), Image.ANTIALIAS) # resizes image in-place
- plt.figure(figsize=(nSize, nSize))
- sTitle='Resized'
- plt.title(sTitle)
- imgplot = plt.imshow(img)

- **Showing the Difference:**
- Enables you to produce two overlaying results but still show that both exist.
- In a data science presentation showing that two sets of nodes on a graph are the same, you must make each set marginally different, in an orderly manner, to facilitate their visualization.
- Without this slight shift, the two sets will simply overlay each other.
- import numpy as np
- from matplotlib import pyplot as plt
- from matplotlib.collections import LineCollection
- from sklearn import manifold
- from sklearn.metrics import euclidean_distances
- from sklearn.decomposition import PCA
- n_samples = 25
- seed = np.random.RandomState(seed=3)
- X_true = seed.randint(0, 20, 2 * n_samples).astype(np.float)
- X_true = X_true.reshape((n_samples, 2))
- # Center the data
- X_true -= X_true.mean()
- similarities = euclidean_distances(X_true)
- You simply add noise to the similarities.
- noise = np.random.rand(n_samples, n_samples)
- noise = noise + noise.T
- noise[np.arange(noise.shape[0]), np.arange(noise.shape[0])] = 0
- similarities += noise
- mds = manifold.MDS(n_components=2, max_iter=3000, eps=1e-9, random_state=seed, dissimilarity="precomputed", n_jobs=1)
- pos = mds.fit(similarities).embedding_
- nmds = manifold.MDS(n_components=2, metric=False, max_iter=3000, eps=1e-12, dissimilarity="precomputed", random_state=seed, n_jobs=1,
- n_init=1)
- npos = nmds.fit_transform(similarities, init=pos)