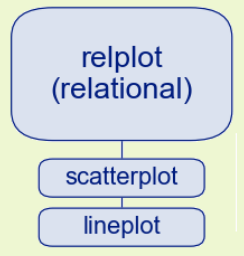
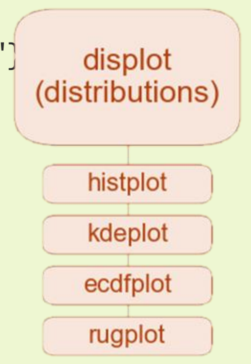
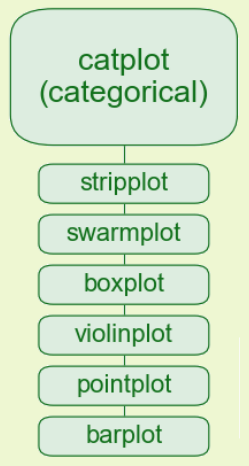
		SQL		SEABORN		PANDAS	
	Steps	**code**	**args**	**code**	**args**	**code**	**args/attributes**
1	**Import & export**			***set seaborn style***		***load from file***	
				sns.set_style()	whitegrid	pd.read_csv-excel-html	
				sns.set_context()	notebook	pd.read_json()	orient='split'
				palette		pd.read_pickle()	
				sns.set_palette()			
				close & clear			
				plt.cla()	clear axis	***load from db***	
				plt.clf()	clear figure	pd.read_sql(query, conn)	
				save			
				plt.savefig()	foo.png	***load from web/API***	
2	**Describe & Subset**	***subset***		***correlation***		***basic descriptive stats***	
		SELECT		sns.heatmap()		df.describe()/info/shape	
				sns.pairplot()		df.value_counts()	
						df.nunique()	
		SELECT DISTINCT		***relation plot***		df[[c1,c2,c3]].duplicated()	
				sns.relplot()			
						slice	
				usual plots -> 1 plot		df.iloc()	
				sns.scatterplot()		df.loc()	
				sns.histplot()		***correlate***	
				sns.countplot()		df1.corr(df2)	
						plotting	
						df.plot	kind, ax
						.hist()	
						.scatter()	x, y
3	**Filter & Clean**	***filter***		***set facetgrid***		***using logical criteria***	
		WHERE	a = b	g = sns.FacetGrid()		df[df.col1 >0]	
			a IN …			***using query***	
		within a group statement	a IS (NOT) NULL	***map a plot to facetgrid***		df.query()	query
		HAVING		g = g.map(plt.hist, 'age')		***using regex***	
						df.filter()	regex
						rename/drop columns	
						df.rename()	columns={}
						df.drop()	columns=[]
						drop missing or duplicated data	
						df.dropna()	
						df.drop_duplicates()	[c1, c2…], keep='first'
						change data type & operate on it	
						s.astype('int')	
						s.astype('category').cat.rename_categories()	
4	**Impute**			***distr plots onto FacetGrid***		***impute missing data***	
	(num & categ)			sns.displot()	kind={'hist','kde','ecdf'}	df.fillna()	method = {'ffill', 'bfill'}
					x=		
				reg plot onto FacetGrid		***stats for imputation***	
				sns.lmplot()		s.median()/mean()/mode()	# check for average
						s.var()/std()	# check for dispersion
						stats for distribution	
						s.quantile()	[0,25, 0,75]





		SQL		SEABORN		PANDAS	
	Steps	**code**	**args**	**code**	**args**	**code**	**args/attributes**
4bis	**Impute**	***parse dates***				***dates : parse (pd.Timestamp / pd.Period)***	
	(dates & text)	DATE(string)				datetime = pd.to_datetime('2017-01')	
		TO_DATE(string, format)				timestamp = pd.Timestamp('date_ts')	attr : {year, month_name()…}
		TO_TIMESTAMP(string, format)				timestamp.strftime()	# -> string
		EXTRACT('year' FROM 'date_ts' :: timestamp)				period = pd.Period('2017-01')	
		DATE(string) + INTERVAL '3 days'				period/DatetimeIndex.asfreq('D')	# define frequency
						text data : clean & extract	
						s.str.strip()/lstrip()/split()	pat='/', expand=True
		clean & extract text data				s.str.lower()/upper()	
						s.str.title()/capitalize()	
		TRIM/LTRIM(string)				***text data : match pattern***	
		LOWER/UPPER(string)				s.str.contains()	# False True True
5	**Summarize & Scale**	***group***		***cat plot -> FacetGrid (1 cat + 1 discr/contin)***		***group statistics***	
		GROUP BY	COUNT()	sns.catplot()	kind=	df.groupby().size()	# size of each group
			MEDIAN()		{'point','bar','count'}	df.groupby()[col].count()	# non-NA count of each obj
			AVG()	***cat plot -> FacetGrid (1 cat + 1 discr-var dist)***		df.groupby()[col].rank()	method, pct
				sns.catplot()	kind=		
		window functions			{'box','violin','boxen'}	***multiple aggregate***	
		LAG(value, offset) OVER()	# call to last value	***cat plot -> FacetGrid (1 cat + 1 contin-var dist)***		df.groupby().agg(func)	np.mean
		LEAD(value, offset) OVER()	# call to following value	sns.catplot()	kind=		np.cumsum
					{'strip','swarm'}	***window functions***	
		window functions (rolling summary)				df.expending()	# -> expending object
		SUM() OVER(				df.rolling()	# -> rolling object for summary funcs
		ROWS BETWEEN CURRENT AND 1 FOLLOWING)					# to be applied to windows of length n
6	**Reshape & Combine**	***reshape***		***multiple plots***		***reshape***	
		ORDER BY		f, ax1, ax2 = plt.subplots	nrows, ncols	df.sort_values()	ascending=
					figsize=(5,6)	pd.melt()	id_vars, value_vars
				add several plots on a same fig		pd.pivot()	columns, values
				ax1 = sns.boxplot()			
		combine by rows				***combine by rows***	
		UNION / UNION ALL		***handle grid facets***		pd.concat()	[df1, df2], axis=0/1
		INTERSECT / EXCEPT		g = sns.catplot()	col = 'catcol1'	s.str.cat(sep=' ')	# series string concat
					row= 'catcol2'		
		combine by columns		g.set_[xy]ticklabels()	[men, women]	***combine by columns***	
		INNER/LEFT JOIN … ON … (ids)		g.set_axis_labels()	xlabel, ylabel	pd.merge()	how={'left', 'outer'}
		FULL OUTER JOIN … ON … (ids)				df1.merge(df2)	how={'left', 'outer'}
7	**Customize & Transform**			***customize labels***		***add/transform columns***	
				ax.set_[xy]label()		df.assign()	new=lambda df: …
						df.insert()	# insert col at specific location
				customize title		pd.qcut(df.col, n)	# bin col into n buckets
				ax.set_title()			
				other			
				ax.set_[xy]lim()			
				ax.set_[xy]ticks()			
				ax.set_[xy]ticklabels()			
				ax.bar_label()			