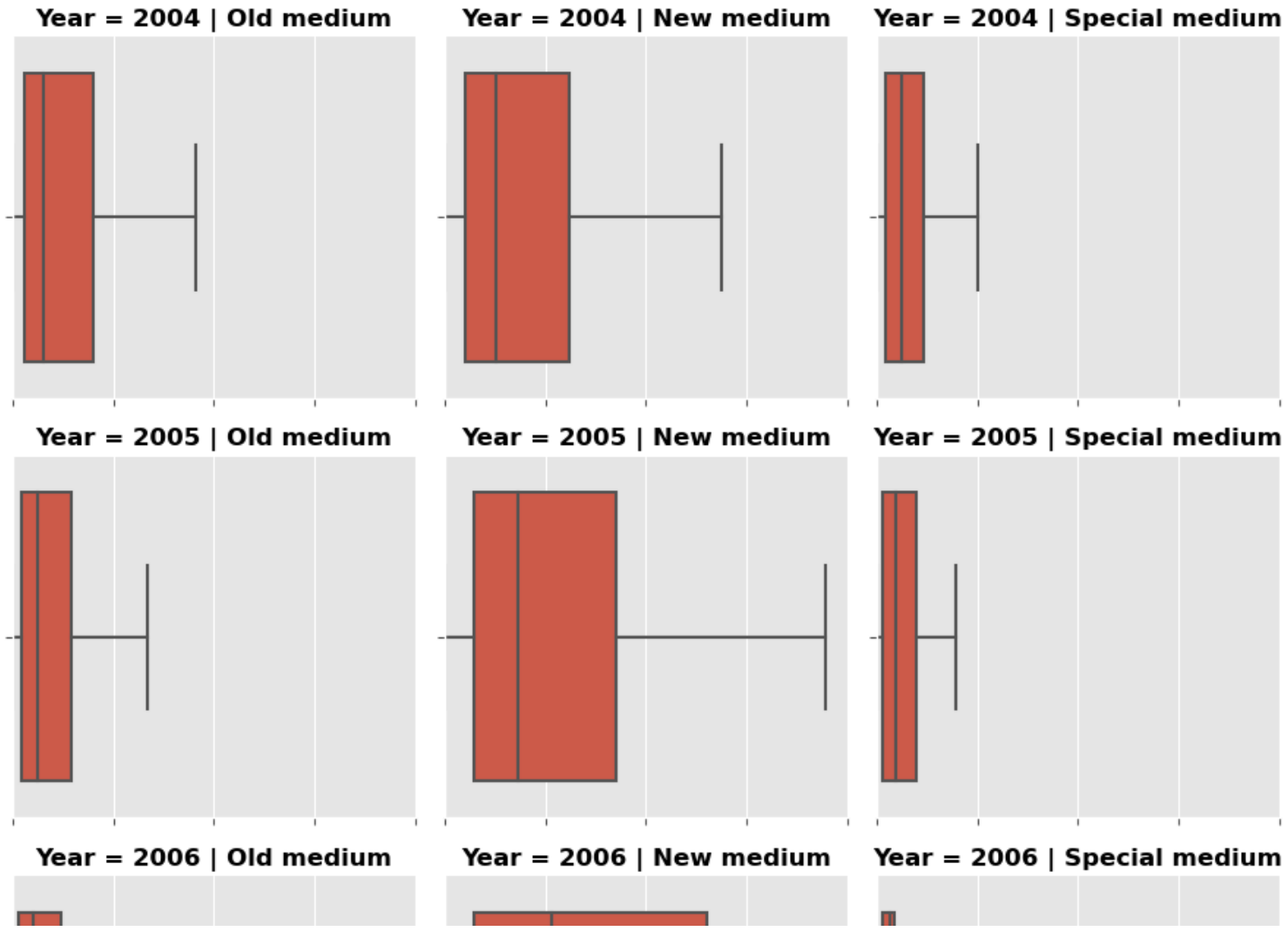
Announcements

HW#3 due 10/5 at 11:59 PM

In [65]:
```
g=sns.FacetGrid(sales,col='Medium',row='Year')
g.map(sns.boxplot,'Revenue',order=['Old medium','New medium','Special medium'],showfliers=False)
g.set(xlim=(0,400000))
g.set_titles(col_template="{col_name}", fontweight='bold', fontsize=18)
```

Out[65]: <seaborn.axisgrid.FacetGrid at 0x1c4cb68d010>



pandas.read_csv

```
pandas.read_csv(filepath_or_buffer, *, sep=_NoDefault.no_default,
delimiter=None, header='infer', names=_NoDefault.no_default, index_col=None,
usecols=None, dtype=None, engine=None, converters=None, true_values=None,
false_values=None, skipinitialspace=False, skiprows=None, skipfooter=0,
nrows=None, na_values=None, keep_default_na=True, na_filter=True, verbose=False,
skip_blank_lines=True, parse_dates=None,
infer_datetime_format=_NoDefault.no_default, keep_date_col=False,
date_parser=_NoDefault.no_default, date_format=None, dayfirst=False,
cache_dates=True, iterator=False, chunksize=None, compression='infer',
thousands=None, decimal='.', lineterminator=None, quotechar='"', quoting=0,
doublequote=True, escapechar=None, comment=None, encoding=None,
encoding_errors='strict', dialect=None, on_bad_lines='error',
delim_whitespace=False, low_memory=True, memory_map=False, float_precision=None,
storage_options=None, dtype_backend=_NoDefault.no_default) #
```

[source]

Read a comma-separated values (csv) file into DataFrame.

Also supports optionally iterating or breaking of the file into chunks.

Additional help can be found in the online docs for IO Tools.

https://pandas.pydata.org/docs/reference/api/pandas.read_csv.html

seaborn.scatterplot

```
seaborn.scatterplot(data=None, *, x=None, y=None, hue=None, size=None, style=None,
palette=None, hue_order=None, hue_norm=None, sizes=None, size_order=None, size_norm=None,
markers=True, style_order=None, legend='auto', ax=None, **kwargs)
```

Draw a scatter plot with possibility of several semantic groupings.

The relationship between `x` and `y` can be shown for different subsets of the data using the `hue`, `size`, and `style` parameters. These parameters control what visual semantics are used to identify the different subsets. It is possible to show up to three dimensions independently by using all three semantic types, but this style of plot can be hard to interpret and is often ineffective. Using redundant semantics (i.e. both `hue` and `style` for the same variable) can be helpful for making graphics more accessible.

See the tutorial for more information.

The default treatment of the `hue` (and to a lesser extent, `size`) semantic, if present, depends on whether the variable is inferred to represent "numeric" or "categorical" data. In particular, numeric variables are represented with a sequential colormap by default, and the legend entries show regular "ticks" with values that may or may not exist in the data. This behavior can be controlled through various parameters, as described and illustrated below.

Parameters: **data** : *`pandas.DataFrame`, `numpy.ndarray`, mapping, or sequence*

Input data structure. Either a long-form collection of vectors that can be assigned to named variables or a wide-form dataset that will be internally reshaped.

x, y : *vectors or keys in `data`*

Variables that specify positions on the x and y axes.

hue : *vector or key in `data`*

Grouping variable that will produce points with different colors. Can be either categorical or numeric, although color mapping will behave differently in latter case.

matplotlib.pyplot.figure

`matplotlib.pyplot.figure(num=None, figsize=None, dpi=None, *, facecolor=None, edgecolor=None, frameon=True, FigureClass=<class 'matplotlib.figure.Figure'>, clear=False, **kwargs)` [source]

Create a new figure, or activate an existing figure.

Parameters:

num : *int or str or* `Figure` *or* `SubFigure`, *optional*

A unique identifier for the figure.
If a figure with that identifier already exists, this figure is made active and returned. An integer refers to the `Figure.number` attribute, a string refers to the figure label.
If there is no figure with the identifier or *num* is not given, a new figure is created, made active and returned. If *num* is an int, it will be used for the `Figure.number` attribute, otherwise, an auto-generated integer value is used (starting at 1 and incremented for each new figure). If *num* is a string, the figure label and the window title is set to this value. If num is a `SubFigure`, its parent `Figure` is activated.

figsize : *(float, float), default:* `rcParams["figure.figsize"]` *(default:* `[6.4, 4.8]`*)*

Width, height in inches.

dpi : *float, default:* `rcParams["figure.dpi"]` *(default:* `100.0`*)*

The resolution of the figure in dots-per-inch.

facecolor : *color, default:* `rcParams["figure.facecolor"]` *(default:* `'white'`*)*

The background color.

edgecolor : *color, default:* `rcParams["figure.edgecolor"]` *(default:* `'white'`*)*

The border color.

frameon : *bool, default: True*

matplotlib.pyplot.title

matplotlib.pyplot.title(*label*, *fontdict=None*, *loc=None*, *pad=None*, *, *y=None*, ***kwargs*) [source]

Set a title for the Axes.

Set one of the three available Axes titles. The available titles are positioned above the Axes in the center, flush with the left edge, and flush with the right edge.

Parameters:

label : *str*

Text to use for the title

fontdict : *dict*

> **Discouraged**
>
> The use of *fontdict* is discouraged. Parameters should be passed as individual keyword arguments or using dictionary-unpacking `set_title(..., **fontdict)`.

loc : *{'center', 'left', 'right'}, default:* `rcParams["axes.titlelocation"]` *(default:* `'center'`*)*

Which title to set.

y : *float, default:* `rcParams["axes.titley"]` *(default:* `None`*)*

Vertical Axes location for the title (1.0 is the top). If None (the default) and `rcParams["axes.titley"]` (default: `None`) is also None, y is determined automatically to avoid decorators on the Axes.

pad : *float, default:* `rcParams["axes.titlepad"]` *(default:* `6.0`*)*

The offset of the title from the top of the Axes, in points.

matplotlib.pyplot.xlabel

matplotlib.pyplot.xlabel(*xlabel*, *fontdict=None*, *labelpad=None*, *, *loc=None*, ***kwargs*) [source]

Set the label for the x-axis.

Parameters:

xlabel : *str*

The label text.

labelpad : *float, default:* `rcParams["axes.labelpad"]` *(default:* `4.0`*)*

Spacing in points from the Axes bounding box including ticks and tick labels. If None, the previous value is left as is.

loc : *{'left', 'center', 'right'}, default:* `rcParams["xaxis.labellocation"]` *(default:* `'center'`*)*

The label position. This is a high-level alternative for passing parameters *x* and *horizontalalignment*.

Other Parameters:

****kwargs** : `Text` *properties*

`Text` properties control the appearance of the label.

matplotlib.pyplot.xticks

`matplotlib.pyplot.xticks(ticks=None, labels=None, *, minor=False, **kwargs)`

Get or set the current tick locations and labels of the x-axis. [source]

Pass no arguments to return the current values without modifying them.

Parameters:

ticks : *array-like, optional*

The list of xtick locations. Passing an empty list removes all xticks.

labels : *array-like, optional*

The labels to place at the given *ticks* locations. This argument can only be passed if *ticks* is passed as well.

minor : *bool, default: False*

If `False`, get/set the major ticks/labels; if `True`, the minor ticks/labels.

****kwargs**

`Text` properties can be used to control the appearance of the labels.

Returns:

locs

The list of xtick locations.

labels

The list of xlabel `Text` objects.

matplotlib.spines

class **matplotlib.spines.Spine**(*axes*, *spine_type*, *path*, ***kwargs*) [source]

Bases: `Patch`

An axis spine -- the line noting the data area boundaries.

Spines are the lines connecting the axis tick marks and noting the boundaries of the data area. They can be placed at arbitrary positions. See `set_position` for more information.

The default position is `('outward', 0)`.

Spines are subclasses of `Patch`, and inherit much of their behavior.

Spines draw a line, a circle, or an arc depending on if `set_patch_line`, `set_patch_circle`, or `set_patch_arc` has been called. Line-like is the default.

For examples see Spines.

Parameters:

axes : *Axes*

The `Axes` instance containing the spine.

spine_type : *str*

The spine type.

path : *Path*

```
set(*, agg_filter=<UNSET>, alpha=<UNSET>, animated=<UNSET>,
antialiased=<UNSET>, bounds=<UNSET>, capstyle=<UNSET>, clip_box=<UNSET>,
clip_on=<UNSET>, clip_path=<UNSET>, color=<UNSET>, ...
facecolor=<UNSET>, fill=<UNSET>, gid=<UNSET>, hatch=...
in_layout=<UNSET>, joinstyle=<UNSET>, label=<UNSET>...
linewidth=<UNSET>, mouseover=<UNSET>, patch_arc=<UN...
path_effects=<UNSET>, picker=<UNSET>, position=<UNS...
sketch_params=<UNSET>, snap=<UNSET>, transform=<UNS...
visible=<UNSET>, zorder=<UNSET>)
```

Set multiple properties at once.

`alpha`	scalar or None
`animated`	bool
`antialiased` or aa	bool or None
`bounds`	(low: float, high: float)
`capstyle`	`CapStyle` or {'butt', 'projecting', 'round'}
`clip_box`	`BboxBase` or None
`clip_on`	bool
`clip_path`	Patch or (Path, Transform) or None
`color`	color
`edgecolor` or ec	color or None
`facecolor` or fc	color or None
`figure`	`Figure`
`fill`	bool
`gid`	str
`hatch`	{'/', '\', '|', '-', '+', 'x', 'o', 'O', '.', '*'}
`in_layout`	bool
`joinstyle`	`JoinStyle` or {'miter', 'round', 'bevel'}
`label`	object
`linestyle` or ls	{'-', '--', '-.', ':', '', (offset, on-off-seq), ...}
`linewidth` or lw	float or None

matplotlib.pyplot.annotate

matplotlib.pyplot.annotate(*text, xy, xytext=None, xycoords='data', textcoords=None, arrowprops=None, annotation_clip=None, **kwargs*) [source]

Annotate the point *xy* with text *text*.

In the simplest form, the text is placed at *xy*.

Optionally, the text can be displayed in another position *xytext*. An arrow pointing from the text to the annotated point *xy* can then be added by defining *arrowprops*.

Parameters:

text : *str*

The text of the annotation.

xy : *(float, float)*

The point *(x, y)* to annotate. The coordinate system is determined by *xycoords*.

xytext : *(float, float), default: xy*

The position *(x, y)* to place the text at. The coordinate system is determined by *textcoords*.

xycoords : *single or two-tuple of str or* `Artist` *or* `Transform` *or callable, default: 'data'*

The coordinate system that *xy* is given in. The following types of values are supported:

```
plt.figure(figsize=(5,5))
sns.scatterplot(data=sales,x='Planned revenue',y='Revenue',color='grey')

plt.title('Revenue vs Planned revenue',loc='left',fontsize=12,fontweight='bold',pad=10,color='green')

plt.xlabel('REVENUE($)',color='green',fontsize=9,)
plt.ylabel('PLANNED REVENUE ($)',color='green',fontsize=9)

plt.xticks(tickposition,ticklabels)
plt.yticks(tickposition,ticklabels)

plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)
plt.gca().spines['left'].set_color('red')
plt.gca().spines['left'].set_linestyle('--')
```

matplotlib.pyplot.axvspan

`matplotlib.pyplot.`**axvspan**`(`*`xmin, xmax, ymin=0, ymax=1, **kwargs`*`)` [source]

Add a vertical span (rectangle) across the Axes.

The rectangle spans from *xmin* to *xmax* horizontally, and, by default, the whole y-axis vertically. The y-span can be set using *ymin* (default: 0) and *ymax* (default: 1) which are in axis units; e.g. `ymin = 0.5` always refers to the middle of the y-axis regardless of the limits set by `set_ylim`.

Parameters:

xmin : ***float***

Lower x-coordinate of the span, in data units.

xmax : ***float***

Upper x-coordinate of the span, in data units.

ymin : ***float, default: 0***

Lower y-coordinate of the span, in y-axis units (0-1).

ymax : ***float, default: 1***

Upper y-coordinate of the span, in y-axis units (0-1).

pandas.to_datetime

pandas.to_datetime(*arg, errors='raise', dayfirst=False, yearfirst=False, utc=False, format=None, exact=_NoDefault.no_default, unit=None, infer_datetime_format=_NoDefault.no_default, origin='unix', cache=True*) [source]

Convert argument to datetime.

This function converts a scalar, array-like, `Series` or `DataFrame`/dict-like to a pandas datetime object.

Parameters:

arg : *int, float, str, datetime, list, tuple, 1-d array, Series, DataFrame/dict-like*

The object to convert to a datetime. If a `DataFrame` is provided, the method expects minimally the following columns: `"year"`, `"month"`, `"day"`. The column "year" must be specified in 4-digit format.

errors : *{'ignore', 'raise', 'coerce'}, default 'raise'*

- If `'raise'`, then invalid parsing will raise an exception.
- If `'coerce'`, then invalid parsing will be set as `NaT`.

pandas.date_range

pandas.date_range(*start=None, end=None, periods=None, freq=None, tz=None, normalize=False, name=None, inclusive='both', *, unit=None, **kwargs*) [source]

Return a fixed frequency DatetimeIndex.

Returns the range of equally spaced time points (where the difference between any two adjacent points is specified by the given frequency) such that they all satisfy *start <[=] x <[=] end*, where the first one and the last one are, resp., the first and last time points in that range that fall on the boundary of `freq` (if given as a frequency string) or that are valid for `freq` (if given as a `pandas.tseries.offsets.DateOffset`). (If exactly one of `start`, `end`, or `freq` is *not* specified, this missing parameter can be computed given `periods`, the number of timesteps in the range. See the note below.)

Parameters:

start : *str or datetime-like, optional*

Left bound for generating dates.

end : *str or datetime-like, optional*

Right bound for generating dates.

periods : *int, optional*

Number of periods to generate.

freq : *str, Timedelta, datetime.timedelta, or DateOffset, default 'D'*

Frequency strings can have multiples, e.g. '5H'. See here for a list of frequency aliases.

tz : *str or tzinfo, optional*

Time zone name for returning localized DatetimeIndex, for example 'Asia/Hong_Kong'.

numpy.random.normal

`random.normal(loc=0.0, scale=1.0, size=None)`

Draw random samples from a normal (Gaussian) distribution.

The probability density function of the normal distribution, first derived by De Moivre and 200 years later by both Gauss and Laplace independently [2], is often called the bell curve because of its characteristic shape (see the example below).

The normal distributions occurs often in nature. For example, it describes the commonly occurring distribution of samples influenced by a large number of tiny, random disturbances, each with its own unique distribution [2].

> **Note**
>
> New code should use the `normal` method of a `Generator` instance instead; please see the Quick Start.

Parameters: **loc** : *float or array_like of floats*

Mean ("centre") of the distribution.

scale : *float or array_like of floats*

Standard deviation (spread or "width") of the distribution. Must be non-negative.

size : *int or tuple of ints, optional*

Output shape. If the given shape is, e.g., `(m, n, k)`, then `m * n * k` samples are drawn. If size is `None` (default), a single value is returned if `loc` and `scale` are both scalars. Otherwise, `np.broadcast(loc, scale).size` samples are drawn.

Returns: **out** : *ndarray or scalar*

Drawn samples from the parameterized normal distribution.

pandas.read_csv

```
pandas.read_csv(filepath_or_buffer, *, sep=_NoDefault.no_default,
delimiter=None, header='infer', names=_NoDefault.no_default, index_col=None,
usecols=None, dtype=None, engine=None, converters=None, true_values=None,
false_values=None, skipinitialspace=False, skiprows=None, skipfooter=0,
nrows=None, na_values=None, keep_default_na=True, na_filter=True, verbose=False,
skip_blank_lines=True, parse_dates=None,
infer_datetime_format=_NoDefault.no_default, keep_date_col=False,
date_parser=_NoDefault.no_default, date_format=None, dayfirst=False,
cache_dates=True, iterator=False, chunksize=None, compression='infer',
thousands=None, decimal='.', lineterminator=None, quotechar='"', quoting=0,
doublequote=True, escapechar=None, comment=None, encoding=None,
encoding_errors='strict', dialect=None, on_bad_lines='error',
delim_whitespace=False, low_memory=True, memory_map=False, float_precision=None,
storage_options=None, dtype_backend=_NoDefault.no_default) #                [source]
```

Read a comma-separated values (csv) file into DataFrame.