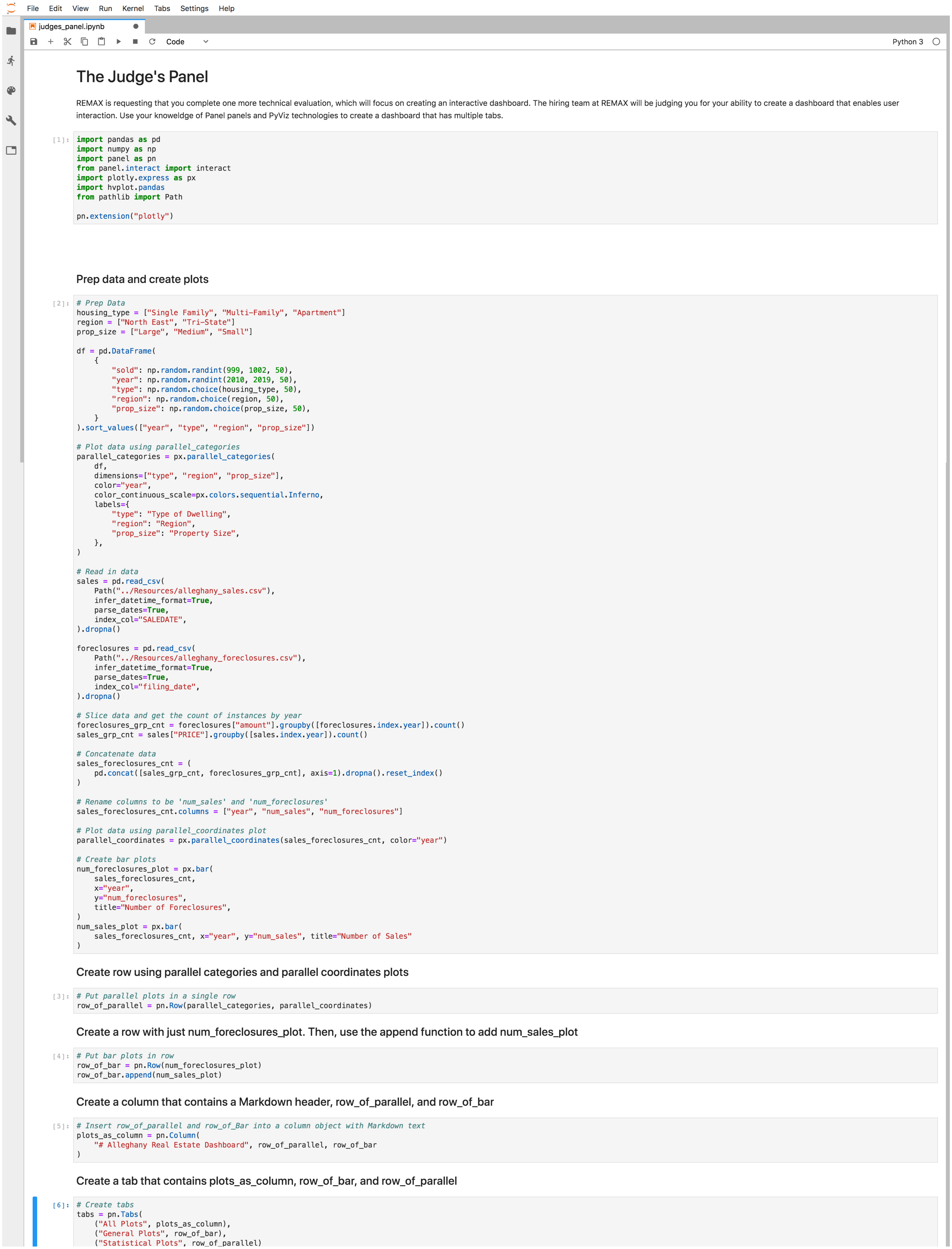


The Judge's Panel

REMAX is requesting that you complete one more technical evaluation, which will focus on creating an interactive dashboard. The hiring team at REMAX will be judging you for your ability to create a dashboard that enables user interaction. Use your knoweldge of Panel panels and PyViz technologies to create a dashboard that has multiple tabs.

[1]:

```python
import pandas as pd
import numpy as np
import panel as pn
from panel.interact import interact
import plotly.express as px
import hvplot.pandas
from pathlib import Path

pn.extension("plotly")
```

Prep data and create plots

[2]:

```python
# Prep Data
housing_type = ["Single Family", "Multi-Family", "Apartment"]
region = ["North East", "Tri-State"]
prop_size = ["Large", "Medium", "Small"]

df = pd.DataFrame(
    {
        "sold": np.random.randint(999, 1002, 50),
        "year": np.random.randint(2010, 2019, 50),
        "type": np.random.choice(housing_type, 50),
        "region": np.random.choice(region, 50),
        "prop_size": np.random.choice(prop_size, 50),
    }
).sort_values(["year", "type", "region", "prop_size"])

# Plot data using parallel_categories
parallel_categories = px.parallel_categories(
    df,
    dimensions=["type", "region", "prop_size"],
    color="year",
    color_continuous_scale=px.colors.sequential.Inferno,
    labels={
        "type": "Type of Dwelling",
        "region": "Region",
        "prop_size": "Property Size",
    },
)

# Read in data
sales = pd.read_csv(
    Path("../Resources/alleghany_sales.csv"),
    infer_datetime_format=True,
    parse_dates=True,
    index_col="SALEDATE",
).dropna()

foreclosures = pd.read_csv(
    Path("../Resources/alleghany_foreclosures.csv"),
    infer_datetime_format=True,
    parse_dates=True,
    index_col="filing_date",
).dropna()

# Slice data and get the count of instances by year
foreclosures_grp_cnt = foreclosures["amount"].groupby([foreclosures.index.year]).count()
sales_grp_cnt = sales["PRICE"].groupby([sales.index.year]).count()

# Concatenate data
sales_foreclosures_cnt = (
    pd.concat([sales_grp_cnt, foreclosures_grp_cnt], axis=1).dropna().reset_index()
)

# Rename columns to be 'num_sales' and 'num_foreclosures'
sales_foreclosures_cnt.columns = ["year", "num_sales", "num_foreclosures"]

# Plot data using parallel_coordinates plot
parallel_coordinates = px.parallel_coordinates(sales_foreclosures_cnt, color="year")

# Create bar plots
num_foreclosures_plot = px.bar(
    sales_foreclosures_cnt,
    x="year",
    y="num_foreclosures",
    title="Number of Foreclosures",
)
num_sales_plot = px.bar(
    sales_foreclosures_cnt, x="year", y="num_sales", title="Number of Sales"
)
```

Create row using parallel categories and parallel coordinates plots

[3]:

```python
# Put parallel plots in a single row
row_of_parallel = pn.Row(parallel_categories, parallel_coordinates)
```

Create a row with just num_foreclosures_plot. Then, use the append function to add num_sales_plot

[4]:

```python
# Put bar plots in row
row_of_bar = pn.Row(num_foreclosures_plot)
row_of_bar.append(num_sales_plot)
```

Create a column that contains a Markdown header, row_of_parallel, and row_of_bar

[5]:

```python
# Insert row_of_parallel and row_of_Bar into a column object with Markdown text
plots_as_column = pn.Column(
    "# Alleghany Real Estate Dashboard", row_of_parallel, row_of_bar
)
```

Create a tab that contains plots_as_column, row_of_bar, and row_of_parallel

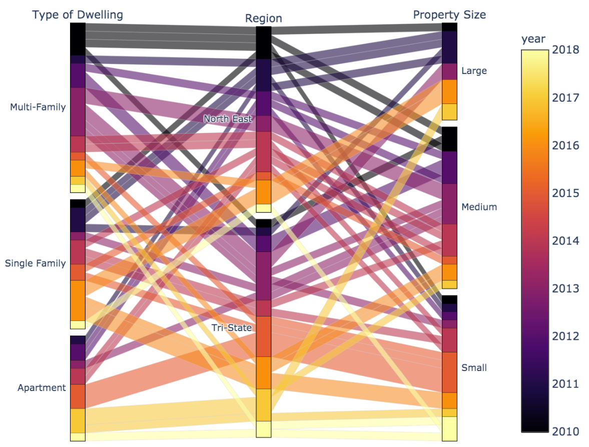
[6]:

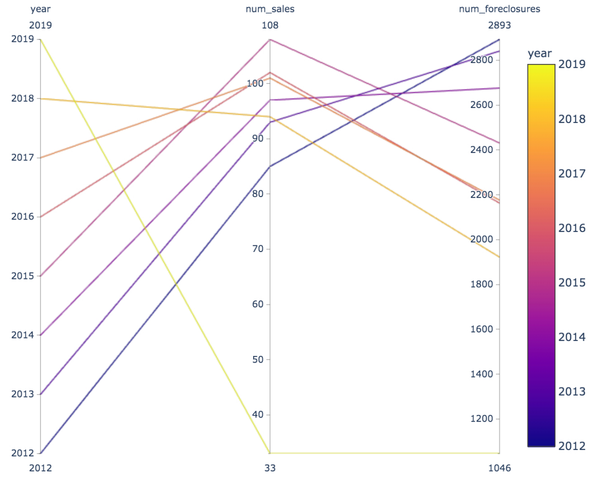
```python
# Create tabs
tabs = pn.Tabs(
    ("All Plots", plots_as_column),
    ("General Plots", row_of_bar),
    ("Statistical Plots", row_of_parallel)
)
tabs
```

[6]:

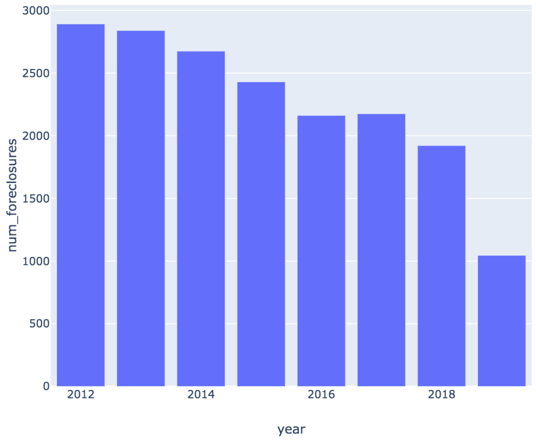
All Plots | General Plots | Statistical Plots

Alleghany Real Estate Dashboard

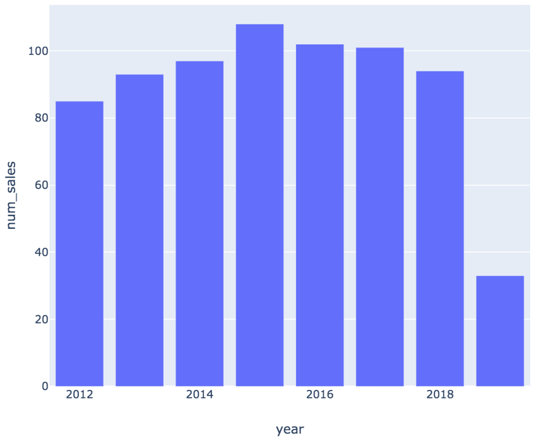




Number of Foreclosures



Number of Sales


num_sales
year