- Steps taken to generate LinkLab Heatmap:

  1. Created a function inside Influx class in influxdb_interface.py that gets total count of data points grouped by device_id from a table.

```
def get_deviceid_datapoint_count(self, fieldname, add_param):
    q_str = 'SELECT COUNT(value) FROM "%s" WHERE %s GROUP BY device_id' % (fieldname, add_param)
    client = self.client
    result_set = client.query(q_str)
    return result_set
```

  2. Created a function in utility.py that makes use of above function to generate a device_id to data_points count map from all the tables with name inside fieldnames list.

```
def get_deviceid_datapoint_countmap(fieldnames, start_time, end_time, device_ids=None,
  timezone='US/Eastern'):
    x = inf.Influx()
    a = x.get_time_query_from_datetime(start_time, end_time)

    if device_ids is not None:
        a += x.get_device_query_adds(device_ids)

    count_map = {}

    for field in fieldnames:
        result_set = x.get_deviceid_datapoint_count(field, a)
        for item in result_set.items():
            device_id = str(item[0][1]["device_id"])
            data_points = next(item[1])["count"]
            if device_id in count_map:
                count_map[device_id] += data_points
            else:
                count_map[device_id] = data_points

    return count_map
```

  3. In heatmap.py inside generate_linklab_heatmap, first, grid to device_ids map is created. Next, making use of grid to device_ids map, device_id to data_points count map returned by above function, and seaborn, heatmap is generated.

```
import utility as util
from datetime import datetime
from dateutil.relativedelta import relativedelta
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt, image as mpimg

def generate_linklab_heatmap(start_datetime, end_datetime, fields, export_filepath):
  try:
    df = pd.read_csv("book_with_grids.csv")
    grid_deviceids_map = {}
    for row in df.itertuples():
      if row.grid in grid_deviceids_map:
        grid_deviceids_map[row.grid].append(row.device_id)
      else:
        grid_deviceids_map[row.grid] = [row.device_id]

    deviceid_datapoint_countmap = util.get_deviceid_datapoint_countmap(fields, start_datetime,
        end_datetime, list(df["device_id"]))

    data = []
    row = 0
    column = 'A'
    for grid in range(200):
      data_points = 0
      if grid in grid_deviceids_map:
        for device_id in grid_deviceids_map[grid]:
          if device_id in deviceid_datapoint_countmap:
            data_points += deviceid_datapoint_countmap[device_id]
      data.append([column, row, grid, data_points])
      column = chr(ord(column) + 1)
      if column == "U":
        row += 1
        column = "A"

    grid_df = pd.DataFrame(data, columns=["Column", "Row", "grid", "data_points"])
    grid_df = grid_df.pivot(index="Row", columns="Column", values="data_points")
```

```
        plt.figure(figsize=(25, 10))
        map_img = mpimg.imread('./img/lll_grid.png')
        heatmap = sns.heatmap(grid_df, linewidths=1, cmap="Reds", annot=True, fmt="d", alpha=0.8,
             zorder=2)
        heatmap.imshow(map_img, aspect=heatmap.get_aspect(),
             extent=heatmap.get_xlim()+heatmap.get_ylim(), zorder=1)
        plt.title(f"LinkLab Heatmap [{start_datetime.date()} - {end_datetime.date()}]")
        filepath = f"{export_filepath}/LinkLab_Heatmap_{start_datetime.date()}_{end_datetime.date()}.png"
        plt.savefig(filepath)
        return filepath
      except Exception as e:
        print(e)
        return None
```

- GitHub link to the repo https://github.com/Kalvinci/lll-book