In [1]:
```
import os

os.environ['KAGGLE_USERNAME'] = "xxxxxx"
os.environ['KAGGLE_KEY'] = "xxxxxx"

!kaggle datasets download -d mohamedbakhet/amazon-books-reviews
```

```
Dataset URL: https://www.kaggle.com/datasets/mohamedbakhet/amazon-books-reviews
License(s): CC0-1.0
amazon-books-reviews.zip: Skipping, found more recently modified local copy (use --force to force download)
```

In [2]:
```
import zipfile

with zipfile.ZipFile("amazon-books-reviews.zip", "r") as zip_ref:
    zip_ref.extractall("dataset")
```

In [3]:
```
import pandas as pd
df_r = pd.read_csv("dataset/Books_rating.csv")
df_r.head()
```

Out[3]:

	Id	Title	Price	User_id	profileName	review/helpfulness	review/score	review/time	review/summary	review/te
0	1882931173	Its Only Art If It's Well Hung!	NaN	AVCGYZL8FQQTD	Jim of Oz "jim-of-oz"	7/7	4.0	940636800	Nice collection of Julie Strain images	This is on for Jul Strain fan It's a col
1	0826414346	Dr. Seuss: American Icon	NaN	A30TK6U7DNS82R	Kevin Killian	10/10	5.0	1095724800	Really Enjoyed It	I don't ca much h Dr. Seu but aft read
2	0826414346	Dr. Seuss: American Icon	NaN	A3UH4UZ4RSVO82	John Granger	10/11	5.0	1078790400	Essential for every personal and Public Library	If peop become t books th read and "t
3	0826414346	Dr. Seuss: American Icon	NaN	A2MVUWT453QH61	Roy E. Perry "amateur philosopher"	7/7	4.0	1090713600	Philip Nel gives silly Seuss a serious treatment	Theodo Seus Geis (1904- 1991), ak "D
4	0826414346	Dr. Seuss: American Icon	NaN	A22X4XUPKF66MR	D. H. Richards "ninthwavestore"	3/3	4.0	1107993600	Good academic overview	Philip Nel Dr. Seus Americ IconThis b

In [4]:
```
#----
# Dataset Composition
#----

print(df_r.shape)

# Dataset sample
df_r_sampled = df_r.sample(n=15000, random_state=42)

# General Overview
import pandas as pd

# NaN for columns
nan_counts_r = df_r_sampled.isnull().sum()
nan_table_r = pd.DataFrame({
'Columns': nan_counts_r.index, 'Number of NaN': nan_counts_r.values
})
print("\nNan Table for Books_rating")
print(nan_table_r)
```

```
(3000000, 10)

Nan Table for Books_rating
             Columns  Number of NaN
0                 Id              0
1              Title              0
2              Price          12663
3            User_id           2787
4        profileName           2788
5  review/helpfulness              0
6       review/score              0
7        review/time              0
8     review/summary              0
9        review/text              0
```

In [5]:
```
#----
# Dataset clean
#----

# Remove row -> user_id NaN
df_cleaned = df_r_sampled.dropna(subset=["User_id"]).copy()

# NaN profileName -> "Unnamed"
df_cleaned["profileName"] = df_cleaned["profileName"].fillna("Unnamed")

duplicate = df_cleaned[df_cleaned.duplicated(subset=["User_id", "Title"], keep=False)]
print(f"Number of duplicate pre: {len(duplicate)}")
duplicate.head()

df = df_cleaned.drop_duplicates(subset=["User_id", "Title"])
duplicate_post = df[df.duplicated(subset=["User_id", "Title"], keep=False)]
print(f"Number of duplicate post: {len(duplicate_post)}")
```

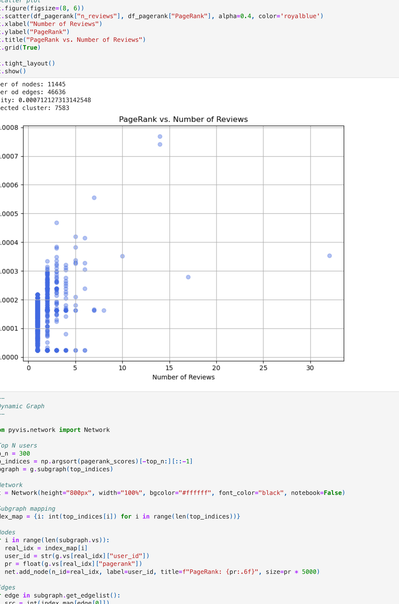
```
Number of duplicate pre: 44
Number of duplicate post: 0
```

In [6]:
```
#----
# PageRank & Graph
#----

import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from scipy.sparse import csr_matrix
import igraph as ig
import matplotlib.pyplot as plt

# Encoding
user_enc = LabelEncoder()
book_enc = LabelEncoder()
df_cleaned['user_idx'] = user_enc.fit_transform(df_cleaned['User_id'])
df_cleaned['book_idx'] = book_enc.fit_transform(df_cleaned['Title'])

# Matrix  user-book
rows = df_cleaned['user_idx'].values
cols = df_cleaned['book_idx'].values
data = np.ones(len(df_cleaned), dtype=np.uint8)
user_book_matrix = csr_matrix((data, (rows, cols)))

# Matrix user-user (A · A')
user_user_matrix = user_book_matrix.dot(user_book_matrix.T)
user_user_matrix.setdiag(0)
user_user_matrix.eliminate_zeros()

# Igraph
coo = user_user_matrix.tocoo()
edges = list(zip(coo.row, coo.col))
g = ig.Graph(edges=edges, directed=False)

# PageRank
pagerank_scores = g.pagerank()
g.vs["pagerank"] = pagerank_scores
g.vs["user_id"] = user_enc.inverse_transform(range(g.vcount()))

# PageRank in CSV
df_pagerank = pd.DataFrame({
    "User ID": g.vs["user_id"],
    "PageRank": g.vs["pagerank"]
}).sort_values(by="PageRank", ascending=False)
df_pagerank.to_csv("pagerank_user_results.csv", index=False)
print("pagerank_user_results.csv – saved")

# Graph
top_n = 300
top_indices = np.argsort(pagerank_scores)[-top_n:][::-1]
subgraph = g.subgraph(top_indices)
layout = subgraph.layout("fr")  # Fruchterman-Reingold
coords = np.array(layout.coords)

# Plot
fig, ax = plt.subplots(figsize=(12, 10))
for edge in subgraph.get_edgelist():
    x = [coords[edge[0]][0], coords[edge[1]][0]]
    y = [coords[edge[0]][1], coords[edge[1]][1]]
    ax.plot(x, y, color="lightgray", alpha=0.5, linewidth=0.5)

ax.scatter(coords[:, 0], coords[:, 1], s=15, color='skyblue', edgecolors='gray')
ax.set_title("User-User Graph – Top Users by PageRank", fontsize=14)
ax.axis('off')
plt.tight_layout()
plt.savefig("pagerank_top_users_graph.png", dpi=300)
plt.show()
```

```
pagerank_user_results.csv – saved
```

In [7]:
```
#----
# Graph composition
#----

print("Number of nodes:", g.vcount())
print("Number od edges:", g.ecount())
print("Density:", g.density())
print("Connected cluster:", len(g.connected_components()))


import matplotlib.pyplot as plt

df_pagerank["n_reviews"] = df_cleaned.groupby("User_id")["Title"].count().reindex(df_pagerank["User ID"]).values

# Scatter plot
plt.figure(figsize=(8, 6))
plt.scatter(df_pagerank["n_reviews"], df_pagerank["PageRank"], alpha=0.4, color='royalblue')
plt.xlabel("Number of Reviews")
plt.ylabel("PageRank")
plt.title("PageRank vs. Number of Reviews")
plt.grid(True)

plt.tight_layout()
plt.show()
```

```
Number of nodes: 11445
Number od edges: 46636
Density: 0.0007172737313142548
Connected cluster: 7583
```



In [8]:
```
#----
# Dynamic Graph
#----

from pyvis.network import Network

# Top N users
top_n = 300
top_indices = np.argsort(pagerank_scores)[-top_n:][::-1]
subgraph = g.subgraph(top_indices)

# Network
net = Network(height="800px", width="100%", bgcolor="#ffffff", font_color="black", notebook=False)

# Subgraph mapping
index_map = {i: int(top_indices[i]) for i in range(len(top_indices))}

# Nodes
for i in range(len(subgraph.vs)):
    real_idx = index_map[i]
    user_id = str(g.vs[real_idx]["user_id"])
    pr = float(g.vs[real_idx]["pagerank"])
    net.add_node(i, id=real_idx, label=user_id, title=f"PageRank: {pr:.6f}", size=pr * 5000)

# Edges
for edge in subgraph.get_edgelist():
    src = int(index_map[edge[0]])
    tgt = int(index_map[edge[1]])
    net.add_edge(src, tgt, color='gray', width=0.2)

# Layout
net.force_atlas_2based(gravity=-30, central_gravity=0.01, spring_length=100)
net.save_graph("pagerank_top_users_graph.html")
print("pagerank_top_users_graph.html – saved")
```

```
pagerank_top_users_graph.html – saved
```