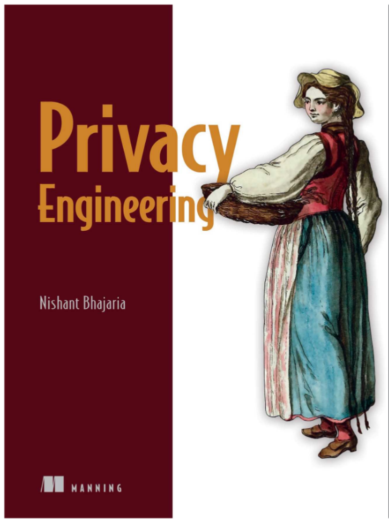
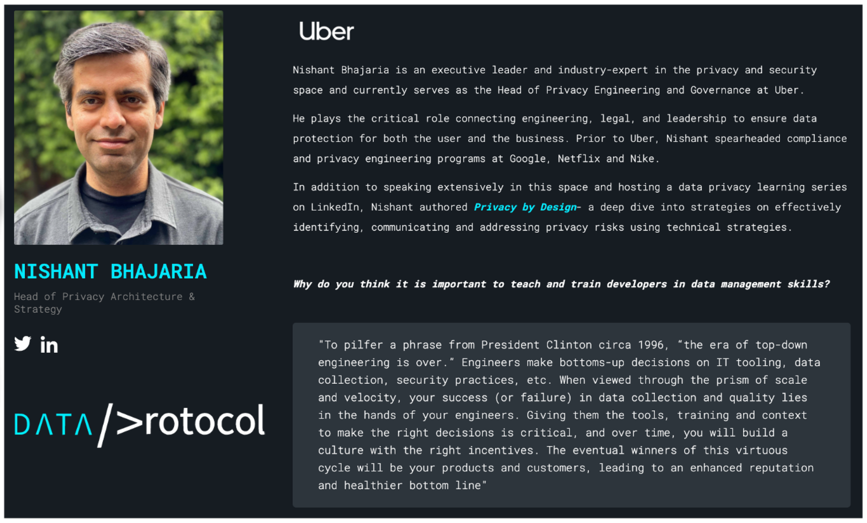
Data Protocol Privacy Engineering Certification

Module 4 - Data Sharing Live Lab





NISHANT BHAJARIA

Head of Privacy Architecture & Strategy

DATA/>rotocol

Uber

Nishant Bhajaria is an executive leader and industry-expert in the privacy and security space and currently serves as the Head of Privacy Engineering and Governance at Uber.

He plays the critical role connecting engineering, legal, and leadership to ensure data protection for both the user and the business. Prior to Uber, Nishant spearheaded compliance and privacy engineering programs at Google, Netflix and Nike.

In addition to speaking extensively in this space and hosting a data privacy learning series on LinkedIn, Nishant authored ***Privacy by Design***- a deep dive into strategies on effectively identifying, communicating and addressing privacy risks using technical strategies.

Why do you think it is important to teach and train developers in data management skills?

"To pilfer a phrase from President Clinton circa 1996, “the era of top-down engineering is over.” Engineers make bottoms-up decisions on IT tooling, data collection, security practices, etc. When viewed through the prism of scale and velocity, your success (or failure) in data collection and quality lies in the hands of your engineers. Giving them the tools, training and context to make the right decisions is critical, and over time, you will build a culture with the right incentives. The eventual winners of this virtuous cycle will be your products and customers, leading to an enhanced reputation and healthier bottom line"

This workbook is part of the Data Protocol Privacy Engineering Certification. Working with data and anonymizing data is a growing essential skill that has to be learned as well as applied. To fully understand and benefit from this workbook, take the course, read the book, and get certified!

This is a Juypter Labs workbook that supports module 4 "Data Sharing", If you are completely new to Jupyter workbooks and want to understand how to use, please watch this short video to learn the basics.

Introduction

This introduction looks at common data sharing techniques introduced in the course, applied in code and with data. One is k-anonymity, a privacy model commonly applied to protect data subjects’ privacy in data sharing scenarios, and the guarantees that k-anonymity can provide when used to anonymise data. Another is l-diversity.

Sharing data is a very important decision to take. It is impossible to get data back once shared and liabilities can be tremendously high, especially if the data being shared can be used to identify individuals and lead to a loss of privacy. You may have tried really hard to anonymize the data being released but legal organizations want to trust more than your word. K-anonymity serves that need by providing one mechanism to quantify the risk contained in any released dataset, it moves the conversation from subjective opinion to factual basis.

Our dataset comes represents trips taken in New York during the month of January 2015. We will be looking at thousands of rides with very rich data attached. This dataset in entirety should never be released externally since re-identification would be easy. Inside the dataset there are four columns that represent where each ride has a pickup location and a drop off location. For each location, we will vary the number of decimal points in their GPS coordinates so that we can provide that same location with varying degrees of precision. Based on this variation in precision, we will see how many other rides meet the same criteria and what the associated privacy values are.

Preparing the Lab

We are going to use python for this exercise since python has many libraries for supporting large datasets such as pandas and numpy, all packaged by scipy.org. We are not going to do anything advanced so we hope you can follow along and focus on the privacy aspects even if you are not a regular python coder. We have designed the code so even if you are not an expert, you should be able to play with the varying input criteria and see the effects on output.

It is necessary to execute each line of the code starting from the top of the workbook. You can re-execute individual cells after changing the contained code if wanted.

The Dataset

We will first import the main libraries we will be using - pandas and numpy. This makes working with large datasets very easy. We then load in the master trip dataset that our engineers have generated for us.

In [1]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Set for nice formatting of table
pd.options.display.max_columns = None

master_df = pd.read_csv(
    "passenger-trips.csv",
    dtype={"Sex": str},
    parse_dates=["Pickup", "Dropoff", "DOB"],
)

master_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 97950 entries, 0 to 97949
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Pickup        97950 non-null  datetime64[ns]
 1   Dropoff       97950 non-null  datetime64[ns]
 2   Pickup_long   97950 non-null  float64
 3   Pickup_lat    97950 non-null  float64
 4   Dropoff_long  97950 non-null  float64
 5   Dropoff_lat   97950 non-null  float64
 6   Sex           97950 non-null  object
 7   DOB           97950 non-null  datetime64[ns]
dtypes: datetime64[ns](3), float64(4), object(1)
memory usage: 6.0+ MB
```

In [2]:
```
print("Master Dataset size = ", len(master_df), '\n')
```

```
Master Dataset size =  97950
```

In [3]:
```
display (master_df.head(3))
```

	Pickup	Dropoff	Pickup_long	Pickup_lat	Dropoff_long	Dropoff_lat	Sex	DOB
0	2015-01-15 19:05:39	2015-01-15 19:23:42	-73.993896	40.750111	-73.974785	40.750618	M	1968-08-04
1	2015-01-10 20:33:38	2015-01-10 20:53:28	-74.001648	40.724243	-73.994415	40.759109	F	1980-07-23
2	2015-01-10 20:33:38	2015-01-10 20:43:41	-73.963341	40.802788	-73.951820	40.824413	M	1996-07-06

We have printed the top 3 rows of the data. You can see the data is incredibly detailed both from a time and location point of view. The data covers all trips taken in the New York area in the month of January, 2015. This is a data set that looks anonymous at first glance since there is no clear Personally Identifying Information (PII) visible. But it is exactly the opposite, it is rich in quasi-identifiers and location data is always sensitive. Both can be combined with other datasets and generally available knowledge to de-anonymize the data.

This workbook shall use this dataset as the master dataset to create secondary privacy centric datasets, that are "safe" to release while maintaining analytics value for the receiver.

Important Note

Privacy analysis (inventory, categorization, sharing) is not absolute, do once and never more. It depends on what data is needed, how big the dataset is, what the context of the dataset is, what specific data is in the specific dataset. This workbook teaches techniques for data sharing that should be continuously applied.

End to End Trip Analysis

The first analyses the pick up and drop off location data that can be released to enable end to end trip analysis by a third party. We use k-anonymity and l-diversity to understand and quantify the risks contained in the data. This quantified risk analysis allows objective discussions with your legal and risk management experts, to decide what should be released and when.

To help, we will create helper functions as we introduce new techniques to study our dataset. Our first analysis extracts the location data from the master dataset, adjusts the decimal point accurary and runs k-anonymity on the resulting dataset, to capture % compliance levels.

In [4]:
```
# Python iteration generator function
def gps_reduce(gps_df):
    for dp in range(5):
        gps_reduced = gps_df.round(decimals=dp)
        yield dp, gps_reduced


# run analysis of data precision versus wanted k-anonymity value
def create_k_anon_matrix(df, transform, k_anon_values):

    k_anon_matrix = pd.DataFrame({}, columns=k_anon_values)

    for row_name, gps_reduced in transform(df):
        frequencies = gps_reduced.value_counts(ascending=True)
        k_result_values = {}

        for k in k_anon_values:
            match = frequencies[frequencies >= k].sum()
            total = frequencies.sum()
            result = round(match / total * 100, 2)
            k_result_values[k] = result

        row = pd.Series(k_result_values)
        row.name = row_name
        k_anon_matrix = k_anon_matrix.append(row)

    return k_anon_matrix

# Call the functions for analysis

k_anon_values = [2, 5, 10, 50, 100, 1000]
working_df = master_df[["Pickup_long", "Pickup_lat", "Dropoff_long", "Dropoff_la
result = create_k_anon_matrix(working_df, gps_reduce, k_anon_values)

result.style.format("{:,.1f}%").set_caption(
        "% K-Anon Compliance: x = k value, rows = gps decimal points\n Dataset s
    )
```

Out[4]:

% K-Anon Compliance: x = k value, rows = gps decimal points Dataset size = 97950

	2	5	10	50	100	1000
0	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
1	99.9%	99.8%	99.7%	99.2%	98.4%	93.8%
2	95.2%	88.7%	82.9%	61.9%	41.8%	0.0%
3	6.8%	0.2%	0.0%	0.0%	0.0%	0.0%
4	0.1%	0.0%	0.0%	0.0%	0.0%	0.0%

Along the x axis are increasing k values so for example the last column is the percentage of records that have 999 other identical entries. The Y column is the precision of the data, moving from 0 decimal places of gps accurracy to 4 decimal places of gps accurary.

This table should not be surprising if we understand how the accuracy of gps locations works.

The gps coordinates have 15 decimal places. Using acuracy information on gis.stackexhange.com we see...

- The tens digit gives a position to about 1,000 kilometers. It gives us useful information about what continent or ocean we are on.
- The units digit (one decimal degree) gives a position up to 111 kilometers (60 nautical miles, about 69 miles). It can tell us roughly what large state or country we are in.
- The first decimal place is worth up to 11.1 km: it can distinguish the position of one large city from a neighboring large city.
- The second decimal place is worth up to 1.1 km: it can separate one village from the next.
- The third decimal place is worth up to 110 m: it can identify a large agricultural field or institutional campus.
- The fourth decimal place is worth up to 11 m: it can identify a parcel of land. It is comparable to the typical accuracy of an uncorrected GPS unit with no interference.
- The fifth decimal place is worth up to 1.1 m: it distinguish trees from each other. Accuracy to this level with commercial GPS units can only be achieved with differential correction.
- The sixth decimal place is worth up to 0.11 m: you can use this for laying out structures in detail, for designing landscapes, building roads. It should be more than good enough for tracking movements of glaciers and rivers. This can be achieved by taking painstaking measures with GPS, such as differentially corrected GPS.
- The seventh decimal place is worth up to 11 mm: this is good for much surveying and is near the limit of what GPS-based techniques can achieve.
- The eighth decimal place is worth up to 1.1 mm: this is good for charting motions of tectonic plates and movements of volcanoes. Permanent, corrected, constantly-running GPS base stations might be able to achieve this level of accuracy.
- The ninth decimal place is worth up to 110 microns: we are getting into the range of microscopy. For almost any conceivable application with earth positions, this is overkill and will be more precise than the accuracy of any surveying device.
- Ten or more decimal places indicates a computer or calculator was used and that no attention was paid to the fact that the extra decimals are useless. Be careful, because unless you are the one reading these numbers off the device, this can indicate low quality processing!

In this dataset if we want an industry best practice of k=5 but also want a location accuracy of 110 metres (3 decimal points) then we would have to strip out 99.9% of the records, which would make the dataset potentially useless for any meaningful analytics. Let us modify the dataset to have the property of k=5 for location data with 2 digit accuracy (1.1 km) by setting all location points that are more precise than that to "missing" which in python and pandas is represented generically by "NaN".

In [5]:

```
# Mask all data that does not meet the wanted k-anonymity value

def setMissing(master_df, df, freq_cols, threshold):

    result_df = master_df.copy()

    frequencies = df[freq_cols].value_counts()

    condition = frequencies <= threshold  # you can define it however you want
    mask_obs = frequencies[condition].index
    for row in mask_obs:
        row_filter = {}
        for i, col in enumerate(freq_cols):
            row_filter[col] = row[i]

        idx = df.loc[(df[list(row_filter)] == pd.Series(row_filter)).all(axis=1)

        result_df.loc[idx, df.columns.isin(freq_cols)] = np.nan

    return result_df


freq_cols = ["Pickup_long", "Pickup_lat", "Dropoff_long", "Dropoff_lat"]
threshold = 5

working_df = master_df.copy()
working_df[freq_cols] = working_df[freq_cols].round(decimals=2)
new_master_df = setMissing(master_df, working_df, freq_cols, threshold)

new_master_df.info()

# Call create_k_anon_matrix for re- analysis

k_anon_values = [2, 5, 10, 50, 100, 1000]
gps_loc = new_master_df[["Pickup_long", "Pickup_lat", "Dropoff_long", "Dropoff_l
result = create_k_anon_matrix(gps_loc, gps_reduce, k_anon_values)

result.style.format("{:,.1f}%").set_caption(
        "% K-Anon Compliance: x = k value, rows = gps decimal points\n Dataset s
    )
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 97950 entries, 0 to 97949
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Pickup        97950 non-null  datetime64[ns]
 1   Dropoff       97950 non-null  datetime64[ns]
 2   Pickup_long   85461 non-null  float64
 3   Pickup_lat    85461 non-null  float64
 4   Dropoff_long  85461 non-null  float64
 5   Dropoff_lat   85461 non-null  float64
 6   Sex           97950 non-null  object
 7   DOB           97950 non-null  datetime64[ns]
dtypes: datetime64[ns](3), float64(4), object(1)
memory usage: 6.0+ MB
```

Out[5]:

% K-Anon Compliance: x = k value, rows = gps decimal points Dataset size = 97950

	2	5	10	50	100	1000
0	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
1	100.0%	100.0%	100.0%	99.7%	99.6%	96.2%
2	100.0%	100.0%	95.0%	71.0%	47.9%	0.0%
3	7.6%	0.2%	0.0%	0.0%	0.0%	0.0%
4	0.1%	0.0%	0.0%	0.0%	0.0%	0.0%

We are now going to lock in on 2 decimal points in the new dataset and run another analysis to get the precise k-anonymity value.

In [6]:

```python
new_master_df[freq_cols] = new_master_df[freq_cols].round(decimals=2)

def analyse_table_k_anon(df):

    print("-----------------------")
    print("Dataset K-Anon Analysis")
    print("-----------------------")

    print("Dataset size = ", len(df))
    print("-----------------------")

    print("K-Anonymity Min to Max Values")
    groupings = df.value_counts(ascending=True).reset_index(name='freq')

    display (groupings)

    print("-----------------------")
    print(
        "Overall K-Anonymity Classification for dataset = ", groupings['freq'][0
    )
    print("-----------------------")


analyse_table_k_anon(new_master_df[freq_cols])
new_master_df.to_csv("cleaned-trips.csv", index=False)
```

```
-----------------------
Dataset K-Anon Analysis
-----------------------
Dataset size =  97950
-----------------------
K-Anonymity Min to Max Values
```

	Pickup_long	Pickup_lat	Dropoff_long	Dropoff_lat	freq
0	-73.97	40.76	-73.94	40.81	6
1	-74.01	40.74	-73.97	40.79	6
2	-73.98	40.79	-74.00	40.74	6
3	-74.01	40.74	-73.96	40.78	6
4	-73.98	40.78	-73.94	40.80	6
...	...	...	...	...	...
2058	-73.96	40.77	-73.97	40.76	447
2059	-73.99	40.75	-73.98	40.75	464
2060	-73.98	40.76	-73.99	40.75	478
2061	-73.97	40.76	-73.96	40.77	520
2062	-73.99	40.75	-73.98	40.76	562

2063 rows × 5 columns

```
-----------------------
```

```
Overall K-Anonymity Classification for dataset =  6
-----------------------
```

We are now going to look at the l-diversity in the new master dataset. For every pickup location there should be "l" dropoff locations and for every dropoff location there should be "l" pickup locations where l > 1. For example an l-diversity of 2 says there should be at least two dropoff locations for every pickup location and vice versa.

In [7]:

```python
def l_diversity_analysis(df, columns):

    lowest_l = None

    print("-----------------------")
    print("Dataset L-Diversity Analysis")
    print("-----------------------")

    print("Dataset Information")
    df.info()
    print("-----------------------")
    print("Dataset size = ", len(df))
    print("-----------------------")

    for col in columns:
        freq = working_df[col].value_counts(ascending=True)
        print(freq)

        print("\n-----------------------")
        print("L-Diversity Min to Max Values: ", col)
        groupings = df.value_counts(ascending=True)

        if lowest_l == None:
            lowest_l = freq.values[0]
        elif lowest_l > freq.values[0]:
            lowest_l = freq.values[0]

    print("-----------------------")
    print("Overall L-Diversity for dataset = ", lowest_l)
    print("-----------------------")


working_df = pd.DataFrame(
    {
        "Pickup_gps": pd.Series([], dtype="object"),
        "Dropoff_gps": pd.Series([], dtype="object"),
    }
)

working_df["Pickup_gps"] = list(
    zip(new_master_df["Pickup_long"], new_master_df["Pickup_lat"])
)
working_df["Dropoff_gps"] = list(
    zip(new_master_df["Dropoff_long"], new_master_df["Dropoff_lat"])
)

l_diversity_analysis(working_df, ["Pickup_gps", "Dropoff_gps"])
```

```
-----------------------
Dataset L-Diversity Analysis
-----------------------
```

```
Dataset Information
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 97950 entries, 0 to 97949
Data columns (total 2 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   Pickup_gps   97950 non-null  object
 1   Dropoff_gps  97950 non-null  object
dtypes: object(2)
memory usage: 1.5+ MB
-----------------------
Dataset size =  97950
-----------------------
(-73.95, 40.75)        6
(-73.91, 40.77)        6
(-73.94, 40.71)        6
(-73.93, 40.76)        7
(-73.94, 40.82)        7
                    ... 
(-73.98, 40.75)     4576
(-73.98, 40.76)     5077
(-73.99, 40.75)     5612
(-73.97, 40.76)     5989
(nan, nan)         12489
Name: Pickup_gps, Length: 87, dtype: int64

-----------------------
L-Diversity Min to Max Values:  Pickup_gps
(-73.91, 40.69)        6
(-73.83, 40.76)        6
(-73.95, 40.69)        6
(-73.99, 40.67)        6
(-73.98, 40.7)         6
                    ... 
(-73.98, 40.75)     4588
(-73.98, 40.76)     5075
(-73.99, 40.75)     5092
(-73.97, 40.76)     5283
(nan, nan)         12489
Name: Dropoff_gps, Length: 125, dtype: int64

-----------------------
L-Diversity Min to Max Values:  Dropoff_gps
-----------------------
Overall L-Diversity for dataset =  6
-----------------------
```

Pickup Analysis

We are now going to use the same orginal dataset and focus on demographics of passengers relating to pickup locations and times only.

We are going to use the columns DOB, Sex, Pickup, Pickup_long and Pickup_lat

In [8]:

```python
pickup_columns = ["DOB", "Sex", "Pickup", "Pickup_long", "Pickup_lat"]
working_df = master_df[pickup_columns].copy()

print(working_df.head(3))
```

```
         DOB Sex              Pickup  Pickup_long  Pickup_lat
0 1968-08-04   M 2015-01-15 19:05:39   -73.993896   40.750111
1 1980-07-23   F 2015-01-10 20:33:38   -74.001648   40.724243
2 1996-07-06   M 2015-01-10 20:33:38   -73.963341   40.802788
```

Let's run k-anonymity analysis on this dataset.

In [9]:

```
analyse_table_k_anon(working_df)
```

```
-----------------------
Dataset K-Anon Analysis
-----------------------
Dataset size =  97950
-----------------------
K-Anonymity Min to Max Values
```

	DOB	Sex	Pickup	Pickup_long	Pickup_lat	freq
0	1955-08-01	F	2015-01-10 01:06:28	-73.968590	40.756573	1
1	1985-08-31	F	2015-01-17 08:58:13	-73.961151	40.801994	1
2	1985-08-31	F	2015-01-09 19:27:55	-73.949570	40.777069	1
3	1985-08-31	F	2015-01-06 22:17:29	-74.008919	40.716106	1
4	1985-08-30	M	2015-01-31 19:13:11	-73.966476	40.757915	1
...	...	...	...	...	...	...
97945	1970-07-28	F	2015-01-12 12:09:32	-73.981857	40.748798	1
97946	1970-07-28	F	2015-01-06 17:46:43	-73.953224	40.767941	1
97947	1970-07-28	F	2015-01-06 08:54:38	-73.969208	40.763103	1
97948	1970-07-29	F	2015-01-05 07:37:26	-73.944969	40.774799	1
97949	2000-07-30	M	2015-01-24 04:33:17	-73.954376	40.778011	1

97950 rows × 6 columns

```
-----------------------
Overall K-Anonymity Classification for dataset =  1
-----------------------
```

Let's start reducing the specificity of the different data types.

We do not need the specific date of birth, let's change this to the year born, to allow age to be approximately calculated.

In [10]:

```
working_df['DOB'] = master_df["DOB"].dt.year

print(working_df.head(3))
```

```
    DOB Sex              Pickup  Pickup_long  Pickup_lat
0  1968   M 2015-01-15 19:05:39   -73.993896   40.750111
1  1980   F 2015-01-10 20:33:38   -74.001648   40.724243
2  1996   M 2015-01-10 20:33:38   -73.963341   40.802788
```

Let's reduce the pickup date and time to just hour of pickup.

In [11]:

```
working_df['Pickup'] = master_df["Pickup"].dt.hour

print(working_df.head(3))
```

```
    DOB Sex  Pickup  Pickup_long  Pickup_lat
0  1968   M      19   -73.993896   40.750111
1  1980   F      20   -74.001648   40.724243
2  1996   M      20   -73.963341   40.802788
```

Let's examine the k-anonymity value of DOB, Sex and Pickup fields.

In [12]:
```
analyse_table_k_anon(working_df[["DOB", "Sex", "Pickup"]])
```

```
-----------------------
Dataset K-Anon Analysis
-----------------------
Dataset size =  97950
-----------------------
K-Anonymity Min to Max Values
```

	DOB	Sex	Pickup	freq
0	1994	F	5	3
1	1955	M	4	3
2	1955	M	5	3
3	1955	F	5	3
4	1963	F	4	3
...	...	...	...	...
2203	1971	M	19	96
2204	1965	F	14	98
2205	1963	F	18	100
2206	1989	M	19	101
2207	1986	F	19	106

2208 rows × 4 columns

```
-----------------------
Overall K-Anonymity Classification for dataset =  3
-----------------------
```

In [13]:
```
# Python iteration generator function
def custom_round(x, base=5):
    return int(base * round(float(x) / base))

def dob_reduce(df):

    for year_precision in [1, 5, 10]:
        df["DOB"] = df["DOB"].apply(lambda x: custom_round(x, base=year_precisio
        print("\n-----------------------")
        print( "Adjusted dob for precision = ", year_precision)
        print("--------------------------")
        print (df)
        yield year_precision, df
```

```
# Call the functions for analysis

k_anon_values = [2, 5, 10, 50, 100, 1000]
result = create_k_anon_matrix(working_df[["DOB", "Sex", "Pickup"]].copy(), dob_r

result.style.format("{:,.1f}%").set_caption(
        "% K-Anon Compliance: x = k value, rows = age ranges\n Dataset size = "
    )
```

```
-----------------------
Adjusted dob for precision =  1
-------------------------
        DOB Sex  Pickup
0      1968   M      19
1      1980   F      20
2      1996   M      20
3      1964   F      20
4      1987   F      20
...     ...  ..     ...
97945  1966   M      18
97946  1979   M      18
97947  1978   M      18
97948  1990   M      18
97949  1992   M      18

[97950 rows x 3 columns]

-----------------------
Adjusted dob for precision =  5
-------------------------
        DOB Sex  Pickup
0      1970   M      19
1      1980   F      20
2      1995   M      20
3      1965   F      20
4      1985   F      20
...     ...  ..     ...
97945  1965   M      18
97946  1980   M      18
97947  1980   M      18
97948  1990   M      18
97949  1990   M      18

[97950 rows x 3 columns]

-----------------------
Adjusted dob for precision =  10
-------------------------
        DOB Sex  Pickup
0      1970   M      19
1      1980   F      20
2      2000   M      20
3      1960   F      20
4      1980   F      20
...     ...  ..     ...
97945  1960   M      18
97946  1980   M      18
97947  1980   M      18
97948  1990   M      18
```

```
97949  1990   M      18

[97950 rows x 3 columns]
```

Out[13]:

% K-Anon Compliance: x = k value, rows = age ranges
Dataset size = 97950

	2	5	10	50	100	1000
1	100.0%	100.0%	99.1%	64.7%	0.3%	0.0%
5	100.0%	100.0%	100.0%	98.7%	93.5%	0.0%
10	100.0%	100.0%	100.0%	99.8%	98.2%	4.5%

We are going to choose to round ages to the nearest decade, to give more optionality on location precision.

In [14]:

```
working_df["DOB"] = working_df["DOB"].apply(lambda x: custom_round(x, base=10))
analyse_table_k_anon(working_df[["DOB", "Sex", "Pickup"]])
```

```
-----------------------
Dataset K-Anon Analysis
-----------------------
Dataset size =  97950
-----------------------
K-Anonymity Min to Max Values
```

	DOB	Sex	Pickup	freq
0	2000	F	4	38
1	2000	F	5	54
2	2000	M	4	57
3	2000	M	5	57
4	1990	F	5	67
...	...	...	...	...
235	1980	F	18	807
236	1980	M	19	827
237	1960	M	19	827
238	1960	F	19	833
239	1980	F	19	842

240 rows × 4 columns

```
-----------------------
Overall K-Anonymity Classification for dataset =  38
-----------------------
```

Let us analyze our options for location precision.

In [15]:

```
# Call create_k_anon_matrix for re- analysis

# Python iteration generator function
def pickup_gps_reduce(df):
    for dp in range(5):
        gps_reduced = df.copy()
        gps_reduced["Pickup_long", "Pickup_lat"] = gps_reduced["Pickup_long", "P
        yield dp, gps_reduced

k_anon_values = [2, 5, 10, 50, 100, 1000]

result = create_k_anon_matrix(working_df.copy(), gps_reduce, k_anon_values)

result.style.format("{:,.1f}%").set_caption(
        "% K-Anon Compliance: x = k value, rows = gps decimal points\n Dataset s
    )
```

Out[15]:

% K-Anon Compliance: x = k value, rows = gps decimal points Dataset size = 97950

	2	5	10	50	100	1000
0	100.0%	100.0%	100.0%	100.0%	98.6%	0.0%
1	99.7%	99.1%	98.0%	90.1%	85.2%	0.0%
2	95.0%	84.7%	67.4%	2.0%	0.0%	0.0%
3	28.6%	1.1%	0.0%	0.0%	0.0%	0.0%
4	1.2%	0.0%	0.0%	0.0%	0.0%	0.0%

Homework

We are going to stop there. We hope you now understand the importance of continuously investigating any data you intend to share with others, both internally and externally. Data privacy is a continuous process that changes depending on specific data and sharing context. Feel free to continue to explore and decide what you would do next with the above dataset.

Welcome to being a data protector...