```
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
```

```
Mounted at /content/drive
```

```
import pandas as pd
import numpy as np

# ============================
# 1. Load the Dataset
# ============================
df = pd.read_csv("/adult.csv")
```

```
df
```

	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relationship	race	sex	capital_gain	capit
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	
...	...	...	...	...	...	...	...	...	...	...	...	
32556	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Female	0	
32557	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	
32558	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Female	0	
32559	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-child	White	Male	0	
32560	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	White	Female	15024	

32561 rows × 15 columns

Next steps: Generate code with df View recommended plots New interactive sheet

```
# ============================
# 2. Generalization of Quasi-Identifiers
# ============================

# Generalize Age into bins
def generalize_age(value):
    if value < 25:
        return "<25"
    elif value < 40:
        return "25-39"
    elif value < 60:
        return "40-59"
    else:
        return "60+"

df["age_generalized"] = df["age"].apply(generalize_age)

# Generalize Education
df["education_generalized"] = df["education"].replace({
    "Bachelors": "HigherEd",
    "Masters": "HigherEd",
    "HS-grad": "Secondary",
    "Some-college": "Secondary"
})

# Generalize Occupation
df["occupation_generalized"] = df["occupation"].replace({
    "Exec-managerial": "White-collar",
    "Tech-support": "White-collar",
    "Tech-support": "White-collar",
    "Craft-repair": "Blue-collar",
    "Other-service": "Service"
})

# Generalize Workclass
df["workclass_generalized"] = df["workclass"].replace({
    "Private": "Private",
    "Self-emp-not-inc": "Self-Employed",
    "Self-emp-inc": "Self-Employed",
    "State-gov": "Government",
    "Federal-gov": "Government"
})

# Generalize Marital Status
df["marital_generalized"] = df["marital_status"].replace({
    "Never-married": "Single",
    "Married-civ-spouse": "Married",
    "Divorced": "Separated"
})

# Generalize Native Country: Map "United-States" to "USA", others to "Other"
df["native_generalized"] = df["native_country"].apply(lambda x: "USA" if x == "United-States" else "Other")


# =============================
# 3. Set Privacy Parameters
# =============================
K = 3                     # Minimum group size for K-Anonymity
L = 2                     # Minimum unique sensitive attribute values for L-Diversity
C_threshold = 50 / 100     # Maximum allowed difference (50%) in sensitive attribute distribution (C-Closeness)

sensitive_attribute = "income"  # Sensitive attribute (e.g., ">50K" or "<=50K")


# =============================
# 4. Compute Overall Sensitive Distribution
# =============================
overall_distribution = df[sensitive_attribute].value_counts(normalize=True)



# =============================
# 5. Apply LKC-Privacy Constraints
# =============================

# Define the list of generalized quasi-identifier columns
qi_cols = [
    "age_generalized",
    "education_generalized",
    "occupation_generalized",
    "workclass_generalized",
    "marital_generalized",
    "native_generalized"
]

# Initialize a "suppressed" flag for all records
df["suppressed"] = False

# For storing C-distance values for groups that pass the checks
c_distances = []

# Process each equivalence class (group) based on the generalized QIs
grouped = df.groupby(qi_cols)
for group_key, group in grouped:
    group_size = len(group)
    # 4a. Check K-Anonymity
    if group_size < K:
        df.loc[group.index, "suppressed"] = True
        continue
    # 4b. Check L-Diversity
    if group[sensitive_attribute].nunique() < L:
        df.loc[group.index, "suppressed"] = True
        continue
    # 4c. Check C-Closeness:
    # Compute the group distribution for the sensitive attribute
    group_distribution = group[sensitive_attribute].value_counts(normalize=True)
    # For each category in overall distribution, compute absolute difference (assume 0 if missing)
    max_diff = 0
    for category, overall_prop in overall_distribution.items():
        group_prop = group_distribution.get(category, 0)
        diff = abs(group_prop - overall_prop)
        if diff > max_diff:
            max_diff = diff
    c_distances.append(max_diff)
    # If maximum difference exceeds threshold, mark group as suppressed
    if max_diff > C_threshold:
        df.loc[group.index, "suppressed"] = True


# ============================
# 6. Prepare Final Anonymized Dataset & Statistical Info
# ============================
# Final valid dataset: records not suppressed
df_valid = df.loc[~df["suppressed"]]
num_suppressed = len(df) - len(df_valid)

# Compute equivalence class size distribution among valid records
eq_class_sizes = df_valid.groupby(qi_cols).size()

# Compute C-Closeness statistics for groups that passed (if any groups passed)
if c_distances:
    avg_c_distance = np.mean(c_distances)
    max_c_distance = np.max(c_distances)
else:
    avg_c_distance = 0
    max_c_distance = 0


# ============================
# 7. Print Statistical Information
# ============================
print("=== Statistical Information ===")
print(f"Original Dataset Size: {len(df)}")
print(f"Anonymized (Valid) Dataset Size: {len(df_valid)}")
print(f"Suppressed Records: {num_suppressed}\n")
print("Equivalence Class Size Distribution (Valid Groups):")
print(eq_class_sizes.value_counts(), "\n")
print(f"C-Closeness Statistics: Average Distance = {avg_c_distance:.4f}, Maximum Distance = {max_c_distance:.4f}")
```

```
=== Statistical Information ===
Original Dataset Size: 32561
Anonymized (Valid) Dataset Size: 19707
Suppressed Records: 12854

Equivalence Class Size Distribution (Valid Groups):
3      97
4      62
5      57
6      44
7      37
       ..
106     1
60      1
78      1
156     1
511     1
Name: count, Length: 115, dtype: int64

C-Closeness Statistics: Average Distance = 0.1987, Maximum Distance = 0.7259
```

```
# ============================
# 8. Save Final Anonymized Dataset
# ============================
# Optionally, drop original QI columns if only generalized values are needed
df_final = df_valid.drop(columns=["age", "education", "occupation", "workclass", "marital_status", "native_country"])
df_final.to_csv("adult_lkc_privacy_merged.csv", index=False)
print("\nFinal anonymized dataset saved as 'adult_lkc_privacy_merged.csv'")
```

```

Final anonymized dataset saved as 'adult_lkc_privacy_merged.csv'
```