

```
1 from google.colab import drive  
2 drive.mount('/content/drive')
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

```
1 import pandas as pd  
2  
3 file_path = '/content/drive/MyDrive/test_2023_public_lar.csv'  
4  
5 df = pd.read_csv(file_path)  
6  
7 df.head()
```

| | LoanAmount | AppraisedValue | LoanToValue_LTVratio | InterestRate | LoanTerm | LoanStatus | DenialReason | PropertyType | ConstructionMethod | OccupancyType | ... | Met |
|---|------------|----------------|----------------------|--------------|----------|------------|--------------|-----------------------------------------|--------------------|---------------|-----|-----|
| 0 | 305000 | 295000.0 | NaN | 5.5 | 360 | 6 | 10 | Single Family (1-4 Units):Site-Built | | 1 | 1 | ... |
| 1 | 185000 | 185000.0 | NaN | 8.0 | 360 | 6 | 10 | Single Family (1-4 Units):Site-Built | | 1 | 1 | ... |
| 2 | 205000 | 205000.0 | NaN | 7.125 | 360 | 6 | 10 | Single Family (1-4 Units):Site-Built | | 1 | 1 | ... |
| 3 | 255000 | NaN | NaN | 6.625 | 360 | 6 | 10 | Single Family (1-4 Units):Site-Built | | 1 | 1 | ... |
| 4 | 175000 | NaN | NaN | 7.125 | 360 | 6 | 10 | Single Family (1-4 Units):Site-Built | | 1 | 1 | ... |

5 rows × 25 columns

Q1. What is this dataset about?

```
1 """
2
3 In plain English, this is a home-lending (mortgage) dataset at the loan-application level. Each row describes a single loan/application with fields
4 that capture the loan's size and terms (LoanAmount, InterestRate, LoanTerm, LoanToValue_LTVratio), the collateral (AppraisedValue, PropertyType,
5 ConstructionMethod, OccupancyType, TotalUnits), limited geography (CountyCode, StateCode, CensusTract, MetroArea), and some underwriting context
6 (ApplicantIncome, DebtToIncomeRatio_DTI, CreditScoreType, DenialReason) along with lender and conforming status.
7
8 The kinds of decisions that I believe could be made using this dataset are:
9
10 - Pricing & risk: calibrate rates, LTV/DTI screens, and conforming thresholds by county/segment.
11 - Market selection: identify counties/metros with strong demand, lower delinquency risk proxies, or better margins.
12 - Product design: tune terms (e.g., LoanTerm) or emphasize products for manufactured vs. site-built homes.
13 - Fair lending monitoring: look for systematic differences across geographies/segments (and then investigate root causes).
14
15 The variables seem most important for housing dynamics are:
16
17 - LoanAmount & AppraisedValue → core to price levels and leverage; together define LTV, a key risk/affordability indicator.
18 - InterestRate & LoanTerm → affordability (monthly payment), demand elasticity, and prepayment risk.
19 - ApplicantIncome & DebtToIncomeRatio_DTI → borrower capacity; strongest signals of ability to repay.
20 - PropertyType, ConstructionMethod, OccupancyType → segment structure (SFH vs. MF, manufactured vs. site-built, owner-occupied vs. investment).
21 - CountyCode / StateCode / MetroArea / CensusTract → local market differences (prices, supply, regulation, economic conditions).
22 - CreditScoreType & DenialReason → credit box tightness and drivers of approval/denial.
23
24 """
```

'\n\nIn plain English, this is a home-lending (mortgage) dataset at the loan-application level. Each row describes a single loan/application with fields\nthat capture the loan's size and terms (LoanAmount, InterestRate, LoanTerm, LoanToValue_LTVratio), the collateral (AppraisedValue, PropertyType,\nConstructionMethod, OccupancyType, TotalUnits), limited geography (CountyCode, StateCode, CensusTract, MetroArea), and some underwriting context\n(ApplicantIncome, DebtToIncomeRatio_DTI, CreditScoreType, DenialReason) along with lender and conforming status.\n\nThe kinds of decisions that I believe could be made using this dataset are:\n- Pricing & risk: calibrate rates, LTV/DTI screens, and conforming thresholds by county/segment.\n- Market selection: identify counties/metros with strong demand, lower delinquency risk proxies, or better margins.\n- Product design: tune terms (e.g., LoanTerm) or emphasize products for manufactured vs. site-built homes.\n- Fair lending monitoring: look fo...'

Q2. Three analysis-ready questions

```

1 """
2
3 1. How does borrower affordability differ across counties?
4
5 Vars: ApplicantIncome, DebtToIncomeRatio_DTI, InterestRate, LoanAmount, CountyCode, OccupancyType.
6 Value: pinpoints where borrowers are most/least stretched → informs county-level pricing, marketing, and risk appetite.
7
8 2. Which factors most strongly explain high LTV loans?
9 Vars: LoanToValue_LTVratio (target), LoanAmount, AppraisedValue, ApplicantIncome, PropertyType, OccupancyType, CountyCode, CreditScoreType.
10 Value: clarifies drivers of leverage → guides appraisal diligence, down-payment assistance, and LTV caps.
11
12 3. Are manufactured homes priced differently than site-built after controlling for income and LTV?
13 Vars: InterestRate (target), ConstructionMethod / ManufacturedType, LoanToValue_LTVratio, ApplicantIncome, CountyCode, LoanTerm, ConformingLoanStatus.
14 Value: detects product-level pricing gaps → supports product design and fair-pricing reviews.
15
16 """

```

'\n\n1. How does borrower affordability differ across counties?\n\nVars: ApplicantIncome, DebtToIncomeRatio_DTI, InterestRate, LoanAmount, CountyCode, OccupancyType.\nValue: pinpoints where borrowers are most/least stretched → informs county-level pricing, marketing, and risk appetite.\n\n2. Which factors most strongly explain high LTV loans?\nVars: LoanToValue_LTVratio (target), LoanAmount, AppraisedValue, ApplicantIncome, PropertyType, OccupancyType, CountyCode, CreditScoreType.\nValue: clarifies drivers of leverage → guides appraisal diligence, down-payment assistance, and LTV caps.\n\n3. Are manufactured homes priced differently than site-built after controlling for income and LTV?\nVars: InterestRate (target), ConstructionMethod / ManufacturedType, LoanToValue_LTVratio, ApplicantIncome, CountyCode, LoanTerm, ConformingLoanStatus.\nValue: detects product-level pricing gaps → supports product design and fair-pricing reviews.\n\n'

Q3. Find Missing Data

```

1 # Q3a – Missing data: counts and percentages
2 import pandas as pd
3 import numpy as np
4 n_rows = len(df)
5
6 missing_counts = df.isna().sum().sort_values(ascending=False)
7 missing_pct = (missing_counts / n_rows * 100).round(2)
8
9 missing_table = (
10     pd.DataFrame({"missing_count": missing_counts, "missing_pct": missing_pct})
11     .sort_values("missing_count", ascending=False)
12 )
13
14 print(f"Dataset shape: {df.shape}")
15 print(f"Columns with any missing: {(missing_counts > 0).sum()} / {df.shape[1]}")
16 missing_table.head(25)

```

Dataset shape: (10200, 25)
 Columns with any missing: 7 / 25

| missing_count | missing_pct |
|---------------|-------------|
| 1 | 0.00% |
| 2 | 0.00% |
| 3 | 0.00% |
| 4 | 0.00% |
| 5 | 0.00% |
| 6 | 0.00% |
| 7 | 0.00% |
| 8 | 0.00% |
| 9 | 0.00% |
| 10 | 0.00% |
| 11 | 0.00% |
| 12 | 0.00% |
| 13 | 0.00% |
| 14 | 0.00% |
| 15 | 0.00% |
| 16 | 0.00% |
| 17 | 0.00% |
| 18 | 0.00% |
| 19 | 0.00% |
| 20 | 0.00% |
| 21 | 0.00% |
| 22 | 0.00% |
| 23 | 0.00% |
| 24 | 0.00% |
| 25 | 0.00% |

| | | |
|-----------------------------------|------|-------|
| LoanToValue_LTVratio | 2633 | 25.81 |
| DebtToIncomeRatio_DTI | 1969 | 19.30 |
| AppraisedValue | 1059 | 10.38 |
| InterestRate | 995 | 9.75 |
| ApplicantIncome | 624 | 6.12 |
| CensusTract | 2 | 0.02 |
| LoanTerm | 1 | 0.01 |
| LoanStatus | 0 | 0.00 |
| LoanAmount | 0 | 0.00 |
| ConstructionMethod | 0 | 0.00 |
| PropertyType | 0 | 0.00 |
| DenialReason | 0 | 0.00 |
| OccupancyType | 0 | 0.00 |
| CountyCode | 0 | 0.00 |
| StateCode | 0 | 0.00 |
| ManufacturedType | 0 | 0.00 |
| TotalUnits | 0 | 0.00 |
| Race | 0 | 0.00 |
| MetroArea | 0 | 0.00 |
| Sex | 0 | 0.00 |
| Ethnicity | 0 | 0.00 |
| CreditScoreType | 0 | 0.00 |
| CoApplicantCreditScoreType | 0 | 0.00 |
| LenderID | 0 | 0.00 |
| ConformingLoanStatus | 0 | 0.00 |

```

1 # Q3b – County list and likely state mapping (via mode of StateCode)
2
3 county_col = next((c for c in ["CountyCode","county","County","county_code"] if c in df.columns), None)
4 state_col = next((c for c in ["StateCode","state","state_code","region","msa"] if c in df.columns), None)
5
6 if not county_col:
7     print("No county-like column found.")
8 else:
9     counties = sorted(df[county_col].dropna().unique().tolist())
10    print(f"Unique counties in '{county_col}':", len(counties))
11    print("County values:", counties)
12
13    if state_col:
14        # Map county → most common state in the data
15        m = (df[[county_col, state_col]]
16              .dropna()
17              .groupby(county_col)[state_col]
18              .agg(lambda s: s.mode().iloc[0] if not s.mode().empty else None)
19              .reset_index()
20              .rename(columns={county_col:"County", state_col:"LikelyState"}))
21        print("\nCounty → LikelyState (from mode):")
22        print(m.sort_values(["LikelyState","County"]).to_string(index=False))
23    else:
24        print("No state/region column found for mapping.")

```

Unique counties in 'CountyCode': 8
 County values: [17031.0, 18097.0, 19113.0, 26163.0, 27053.0, 29095.0, 39061.0, 55025.0]

County → LikelyState (from mode):

| County | LikelyState |
|---------|-------------|
| 19113.0 | IA |
| 17031.0 | IL |
| 18097.0 | IN |
| 26163.0 | MI |
| 27053.0 | MN |
| 29095.0 | MO |
| 39061.0 | OH |
| 55025.0 | WI |

```

1 # Q3c – IQR-based outlier flags and examples
2
3 cols = [c for c in ["LoanAmount","AppraisedValue","LoanToValue_LTVratio","InterestRate","ApplicantIncome"] if c in df.columns]
4 for c in cols:
5     df[c] = pd.to_numeric(df[c], errors="coerce")
6
7 def iqr_bounds(s: pd.Series):
8     s = s.dropna()
9     if len(s) < 5:

```

```
10     return None, None
11     q1, q3 = s.quantile([0.25, 0.75])
12     iqr = q3 - q1
13     return q1 - 1.5*iqr, q3 + 1.5*iqr
14
15 for c in cols:
16     lo, hi = iqr_bounds(df[c])
17     if lo is None:
18         print(f"{c}: not enough data.")
19         continue
20     mask = (df[c] < lo) | (df[c] > hi)
21     print(f"\n{c}: outliers = {mask.sum()} | bounds = [{lo:.3g}, {hi:.3g}]")
22     print(df.loc[mask, [c]].head(5))
```

LoanAmount: outliers = 669 | bounds = [-1.35e+05, 6.65e+05]

| | LoanAmount |
|-----|------------|
| 31 | 725000 |
| 98 | 725000 |
| 148 | 755000 |
| 179 | 675000 |
| 185 | 1485000 |

AppraisedValue: outliers = 747 | bounds = [-1.55e+05, 8.05e+05]

| | AppraisedValue |
|-----|----------------|
| 94 | 855000.0 |
| 98 | 965000.0 |
| 158 | 1285000.0 |
| 163 | 1185000.0 |
| 165 | 855000.0 |

LoanToValue_LTVratio: outliers = 524 | bounds = [55.2, 121]

| | LoanToValue_LTVratio |
|-----|----------------------|
| 151 | 45.410 |
| 155 | 53.506 |
| 156 | 49.020 |
| 157 | 34.916 |
| 158 | 37.296 |

InterestRate: outliers = 266 | bounds = [4.5, 8.5]

| | InterestRate |
|-----|--------------|
| 146 | 3.250 |
| 260 | 8.750 |
| 278 | 11.490 |
| 309 | 9.250 |
| 310 | 9.375 |

ApplicantIncome: outliers = 843 | bounds = [-72, 288]

| | ApplicantIncome |
|-----|-----------------|
| 148 | 508.0 |
| 163 | 352.0 |
| 170 | 366.0 |

| | |
|-----|--------|
| 177 | 354.0 |
| 185 | 1608.0 |

```

1 # Q3c – verbal questions
2
3 ...
4 # How would you confirm if it's a data entry error or a legitimate value?
5
6 – I will confirm legitimacy by:
7 (1) Cross-checking internal consistency (e.g., AppraisedValue > 0, recompute LTV ≈ LoanAmount/AppraisedValue * 100),
8 (2) Comparing to realistic ranges from the data dictionary/course notes,
9 (3) Checking if extreme values cluster in specific counties/segments where high tail is plausible,
10 (4) Looking for obvious keying artifacts (e.g., InterestRate == 0 or 99).
11
12 # What to do with the outliers (and why)
13
14 1) LoanAmount & AppraisedValue (very big numbers)
15 Do: Keep them in the data. For charts/models, use a log transform (e.g., log1p(x)) or winsorize (cap at the 1st/99th percentiles).
16 Why: Expensive homes and big loans are real. Log/winsorizing stops a few huge numbers from skewing results.
17
18 2) LoanToValue_LTVratio (low values like 35–55)
19 Do: Keep them. Maybe cap only very high LTVs if you must.
20 Why: Low LTV just means bigger down payments—totally normal, not an error.
21
22 3) InterestRate (very low like 3.25% or high like 9–11.5%)
23 Do: Keep them. If you need to fill missing rates later, fill by similar loans (same LoanTerm and ConformingLoanStatus). For modeling, light
24 winsorizing is okay.
25 Why: Different products and times have different rates. These are likely real.
26
27 4) ApplicantIncome (very high, e.g., 352–1608)
28 Do: Keep. For modeling, log transform and/or winsorize the high tail. Also check against DTI and loan size.
29 Why: High incomes exist, especially in some areas. Log/winsorizing reduces distortion.
30
31 ...

```

```
'\n# How would you confirm if it's a data entry error or a legitimate value?\n\n- I will confirm legitimacy by:\n(1) Cross-checking internal consistency (e.\n    g., AppraisedValue > 0, recompute LTV ≈ LoanAmount/AppraisedValue * 100),\n(2) Comparing to realistic ranges from the data dictionary/course notes,\n(3) Che\ncking if extreme values cluster in specific counties/segments where high tail is plausible,\n(4) Looking for obvious keying artifacts (e.g., InterestRate ==\n0 or 99).\n\n# What to do with the outliers (and why)\n\n1) LoanAmount & AppraisedValue (very big numbers)\nDo: Keep them in the data. For charts/models, us\ne a log transform (e.g., log1p(x)) or winsorize (cap at the 1st/99th percentiles).\nWhy: Expensive homes and big loans are real. Log/winsorizing stops a few\nhuge numbers from skewing results.\n\n2) LoanToValue_LTVratio (low values like 35–55)\nDo: Keep them. Maybe cap only very high LTVs if you must.\nWhy: Low L\nTV just means bigger down payments—totally normal, not an error...'
```

Q4 – Identify missing variables

```

1 # 1) Identify variables with missing data
2 missing_pct = (missing_counts / n_rows * 100).round(2)
3 missing = (
4     pd.DataFrame({"missing_count": missing_counts, "missing_pct": missing_pct})
5     .query("missing_count > 0")
6     .sort_values("missing_count", ascending=False)
7 )
8 print("Columns with missing values:")
9 print(missing.to_string())

```

Columns with missing values:

| | missing_count | missing_pct |
|-----------------------|---------------|-------------|
| LoanToValue_LTVratio | 2714 | 26.61 |
| DebtToIncomeRatio_DTI | 1969 | 19.30 |
| InterestRate | 1076 | 10.55 |
| AppraisedValue | 1059 | 10.38 |
| ApplicantIncome | 624 | 6.12 |
| CensusTract | 2 | 0.02 |
| LoanTerm | 1 | 0.01 |

```
1 # 2) Keep/Drop policy for missing values:
2 """
3 - LoanToValue_LTVratio: KEEP; compute = LoanAmount/AppraisedValue*100 when both exist (preserves meaning).
4 - InterestRate: KEEP; impute median within product/term if available; fallback to global median.
5 - ApplicantIncome: KEEP; impute median by CountyCode (local incomes); fallback to global median.
6 - AppraisedValue: KEEP; if you must impute later, use median within (CountyCode, PropertyType); otherwise leave as NaN.
7 - DebtToIncomeRatio_DTI: KEEP; if numeric, median by (CountyCode, OccupancyType); if banded, mode.
8 - LoanTerm, CensusTract: very small missingness → reasonable to DROP those few rows instead of inventing values.
9 """

```

'\n- LoanToValue_LTVratio: KEEP; compute = LoanAmount/AppraisedValue*100 when both exist (preserves meaning).\n- InterestRate: KEEP; impute median within pr
oduct/term if available; fallback to global median.\n- ApplicantIncome: KEEP; impute median by CountyCode (local incomes); fallback to global median.\n- App
raisedValue: KEEP; if you must impute later, use median within (CountyCode, PropertyType); otherwise leave as NaN.\n- DebtToIncomeRatio_DTI: KEEP; if numeri
c, median by (CountyCode, OccupancyType); if banded, mode.\n- LoanTerm, CensusTract: very small missingness → reasonable to DROP those few rows instead of i
nventing values.\n'

```

1 # 3) Clean THREE variables (LTV, InterestRate, ApplicantIncome)
2 df_clean = df.copy()
3
4 # Ensure numeric types where needed
5 for c in ["LoanAmount","AppraisedValue","LoanToValue_LTVratio","InterestRate","ApplicantIncome"]:
6     if c in df_clean.columns:
7         df_clean[c] = pd.to_numeric(df_clean[c], errors="coerce")
8
9 report = []
10
11 # A) LTV: compute when possible
12 if {"LoanToValue_LTVratio","LoanAmount","AppraisedValue"}.issubset(df_clean.columns):

```

```

13     before = int(df_clean["LoanToValue_LTVratio"].isna().sum())
14     m = (
15         df_clean["LoanToValue_LTVratio"].isna()
16         & df_clean["LoanAmount"].notna()
17         & df_clean["AppraisedValue"].notna()
18         & (df_clean["AppraisedValue"] > 0)
19     )
20     df_clean.loc[m, "LoanToValue_LTVratio"] =
21         df_clean.loc[m, "LoanAmount"] / df_clean.loc[m, "AppraisedValue"] * 100
22     )
23     after = int(df_clean["LoanToValue_LTVratio"].isna().sum())
24     report.append(["LoanToValue_LTVratio", before, after, "computed = LoanAmount/AppraisedValue*100"])
25
26 # B) InterestRate: median by (LoanTerm, ConformingLoanStatus) if present; else global
27 if "InterestRate" in df_clean.columns:
28     before = int(df_clean["InterestRate"].isna().sum())
29     group_cols = [c for c in ["LoanTerm", "ConformingLoanStatus"] if c in df_clean.columns]
30     if group_cols:
31         med = df_clean.groupby(group_cols)["InterestRate"].transform("median")
32         df_clean["InterestRate"] = df_clean["InterestRate"].fillna(med)
33     df_clean["InterestRate"] = df_clean["InterestRate"].fillna(df_clean["InterestRate"].median())
34     after = int(df_clean["InterestRate"].isna().sum())
35     report.append(["InterestRate", before, after, f"median by {group_cols or 'global'}, then global median"])
36
37 # C) ApplicantIncome: median by CountyCode; else global
38 if "ApplicantIncome" in df_clean.columns:
39     before = int(df_clean["ApplicantIncome"].isna().sum())
40     if "CountyCode" in df_clean.columns:
41         med_cc = df_clean.groupby("CountyCode")["ApplicantIncome"].transform("median")
42         df_clean["ApplicantIncome"] = df_clean["ApplicantIncome"].fillna(med_cc)
43     df_clean["ApplicantIncome"] = df_clean["ApplicantIncome"].fillna(df_clean["ApplicantIncome"].median())
44     after = int(df_clean["ApplicantIncome"].isna().sum())
45     report.append(["ApplicantIncome", before, after, "median by CountyCode, then global median"])
46
47 # Show cleaning summary
48 clean_report = pd.DataFrame(report, columns=["variable", "missing_before", "missing_after", "strategy"])
49 print("\nCleaning summary:")
50 print(clean_report.to_string(index=False))
51
52 # Before/after snapshot for top-missing columns
53 print("\nMissingness AFTER cleaning (top 15):")
54 print(df_clean.isna().sum().sort_values(ascending=False).head(15).to_string())

```

Cleaning summary:

| | variable | missing_before | missing_after | strategy |
|----------------------|----------|----------------|---------------|--------------------------------------------------------------------|
| LoanToValue_LTVratio | | 2714 | 312 | computed = LoanAmount/AppraisedValue*100 |
| InterestRate | | 1076 | 0 | median by ['LoanTerm', 'ConformingLoanStatus'], then global median |
| ApplicantIncome | | 624 | 0 | median by CountyCode, then global median |

Missingness AFTER cleaning (top 15):

| | |
|-----------------------|------|
| DebtToIncomeRatio_DTI | 1969 |
| AppraisedValue | 1059 |
| LoanToValue_LTVratio | 312 |
| CensusTract | 2 |
| LoanTerm | 1 |
| LoanStatus | 0 |
| DenialReason | 0 |
| LoanAmount | 0 |
| InterestRate | 0 |
| ConstructionMethod | 0 |
| PropertyType | 0 |
| ManufacturedType | 0 |
| OccupancyType | 0 |
| CountyCode | 0 |
| StateCode | 0 |

1 # 3) Process Explanation:

```

2 ''
3
4 - Identify missingness: I listed every column with missing values and the % of rows affected.
5 - Policy:
6   + Keep important economic fields and impute intelligently using related info (e.g., compute LTV from LoanAmount/AppraisedValue; impute
7   InterestRate by loan term/product; impute ApplicantIncome by county to reflect local wages).
8   + Drop rows only when missingness is tiny in ID-like fields (e.g., LoanTerm, CensusTract) to avoid inventing identifiers.
9 - Cleaned three variables:
10 1) LoanToValue_LTVratio = LoanAmount / AppraisedValue × 100 when both are present (preserves meaning).
11 2) InterestRate filled by median within (LoanTerm, ConformingLoanStatus), then global median (respects product tiers).
12 3) ApplicantIncome filled by county median, then global median (captures local income levels).
13 - Why this fits: These choices use domain structure (loan terms, product status, geography) to make fills realistic, reduce bias, and keep as many
14 useful rows as possible for analysis.
15
16 ''

```

'\n\n- Identify missingness: I listed every column with missing values and the % of rows affected.\n- Policy:\n + Keep important economic fields and impute intelligently using related info (e.g., compute LTV from LoanAmount/AppraisedValue; impute InterestRate by loan term/product; impute ApplicantIncome by county to reflect local wages).\n + Drop rows only when missingness is tiny in ID-like fields (e.g., LoanTerm, CensusTract) to avoid inventing identifiers.\n- Cleaned three variables:\n 1) LoanToValue_LTVratio = LoanAmount / AppraisedValue × 100 when both are present (preserves meaning).\n 2) InterestRate filled by median within (LoanTerm, ConformingLoanStatus), then global median (respects product tiers).\n 3) ApplicantIncome filled by county median, then global median (captures local income levels).\n- Why this fits: These choices use domain structure (loan terms, product status, geography) to make fills realistic, reduce bias, and keep as many useful rows as possible for analy...'

Q5. Reflection

```

1 ***
2 1) Key steps I took are:
3 - Got oriented: skimmed column names and the data dictionary to understand entities (loan, property, borrower, geography).
4 - Scanned data quality: computed missing counts/percentages to see where problems live (e.g., LTV, DTI, AppraisedValue, InterestRate, Income).
5 - Checked geography: identified the county and state fields and mapped counties → likely states to understand coverage.
6 - Probed distributions/outliers: used a simple IQR rule on monetary/rate fields (LoanAmount, AppraisedValue, LTV, InterestRate, Income) to spot
7 unusual points.
8 - Chose practical fixes: designed minimal, domain-sensible imputations (compute LTV from fundamentals; median InterestRate within product/term;
9 county-median ApplicantIncome).
10 - Validated logic: sanity-checked with internal consistency (e.g.,  $LTV \approx LoanAmount / AppraisedValue \times 100$ ), "impossible value" rules, and small
11 before/after missingness snapshots.
12
13 2) What surprised me are:
14 - How much LTV and DTI were missing: >25% of rows lacked LTV and ~19.3% lacked DTI—higher than I expected for core risk fields. That pushed me to
15 compute LTV where possible rather than blanket imputation.
16 - A small set of counties (only 8 unique): I expected broader national coverage; the limited geography suggests results are quite regional and not
17 automatically generalizable.
18 - Outliers weren't "wrong," just real tails: Very large loans/appraisals and high incomes showed up, plus rate values outside the 4.5–8.5% IQR band.
19 These look like genuine market/product effects (e.g., jumbos, non-conforming loans, timing), reminding me that "outlier" ≠ "error."
20 - Low LTV flagged as "outliers" by IQR: Statistically unusual but economically sensible—big down payments are legitimate and typically lower risk.
21 Good reminder to pair stats with domain logic.
22 - Tiny missingness in ID-like fields (e.g., LoanTerm, CensusTract): nice win—dropping 1–2 rows can be cleaner than inventing identifiers.
23
24 3) If I continued, I would:
25 - Segment results by product/term and county to see how pricing/affordability shifts locally.
26 - Add quick log transforms / winsorization for modeling stability, and run a simple model to quantify drivers of LTV or InterestRate.
27 - Revisit DTI handling (banded vs numeric) and, if needed, impute within county/occupancy groups to keep more rows for analysis.
28
29 ***

```

'\n1) Key steps I took are:\n- Got oriented: skimmed column names and the data dictionary to understand entities (loan, property, borrower, geography).\n- Scanned data quality: computed missing counts/percentages to see where problems live (e.g., LTV, DTI, AppraisedValue, InterestRate, Income).\n- Checked geography: identified the county and state fields and mapped counties → likely states to understand coverage.\n- Probed distributions/outliers: used a simple IQR rule on monetary/rate fields (LoanAmount, AppraisedValue, LTV, InterestRate, Income) to spot\nunusual points.\n- Chose practical fixes: designed minimal, domain-sensible imputations (compute LTV from fundamentals; median InterestRate within product/term;\ncounty-median ApplicantIncome).\n- Validated logic: sanity-checked with internal consistency (e.g., $LTV \approx LoanAmount / AppraisedValue \times 100$), "impossible value" rules, and small\nbefore/after missingness snapshots.\n\n2) What surprised me are:\n- How much LTV and DTI were ...'

Q6. AI Disclosure:

```
1 '''
2 I used AI tools, but only for deeper business problem understanding: what I did was posting the entire .xlsx dictionary file to the ChatGPT and
3 asked it to explain deeper about what those variable means in the context and why are they being encoded like that (ONLY thing I did). Those helped
4 me a lot in reaching the high level of understanding of this problem that incentivized me to better understand how each code I write influences the
5 entire problem.
6
7 NOTE: All the codes are made by my own.
8
9'''
```

'\nI used AI tools, but only for deeper business problem understanding: what I did was posting the entire .xlsx dictionary file to the ChatGPT and \nasked i
t to explain deeper about what those variable means in the context and why are they being encoded like that (ONLY thing I did). Those helped \nme a lot in r
eaching the high level of understanding of this problem that incentivized me to better understand how each code I write influences the\ntire problem. \n\n
NOTE: All the codes are made by my own.\n\n'