Projet MLOps – Exploration des données de revenus

Objectif

L'objectif de ce projet est de mettre en place un pipeline de data science permettant de prédire le niveau de revenu d'un individu (≤50K ou >50K) à partir de caractéristiques socio-démographiques. Dans cette partie, nous explorerons les données.

Import des bibliothèques et configuration

Entrée [88]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

pd.set_option("display.max_columns", 200)
pd.set_option("display.width", 200)
pd.set_option("display.max_rows", None)
```

Chargement des données

Entrée [53]:
```
DATA_PATH = "../data/raw/revenus.csv"

try:
    df = pd.read_csv(DATA_PATH, sep=",")
except:
    df = pd.read_csv(DATA_PATH, sep=",")
```

Aperçu du jeu de données

Entrée [54]:
```
print("Dimensions du dataset :", df.shape)
df.head()
```

```
Dimensions du dataset : (48842, 15)
```

Out[54]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain
0	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-child	Black	Male	0
1	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	Male	0
2	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White	Male	0
3	44	Private	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male	7688
4	18	?	103497	Some-college	10	Never-married	?	Own-child	White	Female	0

Structure et types des variables

Entrée [55]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48842 entries, 0 to 48841
Data columns (total 15 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   age              48842 non-null  int64 
 1   workclass        48842 non-null  object
 2   fnlwgt           48842 non-null  int64 
 3   education        48842 non-null  object
 4   educational-num  48842 non-null  int64 
 5   marital-status   48842 non-null  object
 6   occupation       48842 non-null  object
 7   relationship     48842 non-null  object
 8   race             48842 non-null  object
 9   gender           48842 non-null  object
 10  capital-gain     48842 non-null  int64 
 11  capital-loss     48842 non-null  int64 
 12  hours-per-week   48842 non-null  int64 
 13  native-country   48842 non-null  object
```

Explorations des modalités des variables

Entrée [56]:
```
# Afficher les modalités (valeurs distinctes) de chaque variable
for col in df.columns:
    print("\n" + "="*60)
    print(f"Variable : {col}")
    print(f"Type : {df[col].dtype}")
    print(f"Nombre de modalités : {df[col].nunique()}")

    # Affichage des modalités
    values = df[col].unique()

    # Couper l'affichage
    if len(values) > 30:
        print("Modalités (30 premières) :")
        print(values[:30])
        print("... (tronqué)")
    else:
        print("Modalités :")
        print(values)
```

```

============================================================
Variable : age
Type : int64
Nombre de modalités : 74
Modalités (30 premières) :
[25 38 28 44 18 34 29 63 24 55 65 36 26 58 48 43 20 37 40 72 45 22 23 54
 32 46 56 17 39 52]
... (tronqué)

============================================================
Variable : workclass
Type : object
Nombre de modalités : 9
Modalités :
['Private' 'Local-gov' '?' 'Self-emp-not-inc' 'Federal-gov' 'State-gov'
 'Self-emp-inc' 'Without-pay' 'Never-worked']

============================================================
```

Valeurs manquantes et valeurs aberrantes

Certaines variables contiennent des valeurs manquantes codées par le caractère "?" (6465 valeurs observées).
Elles seront traitées lors de l'étape de pré-traitement.

Entrée [57]:
```
# Comptage des "?" par variable
missing_counts = (df == "?").sum()

# les variables concernées
missing_counts = missing_counts[missing_counts > 0]

# Tableau récapitulatif
missing_table = pd.DataFrame({
    "Variable": missing_counts.index,
    "Effectif": missing_counts.values
})

# Calcul des pourcentages (par rapport au nombre total de lignes)
missing_table["Pourcentage (%)"] = (missing_table["Effectif"] / len(df)) * 100

# Arrondis du %
missing_table["Pourcentage (%)"] = missing_table["Pourcentage (%)"].round(2)

# Calcul des totaux
total_effectif = missing_table["Effectif"].sum()
total_pourcentage = missing_table["Pourcentage (%)"].sum()

# Ajout de la ligne Total
total_row = pd.DataFrame({
    "Variable": ["Total"],
    "Effectif": [total_effectif],
    "Pourcentage (%)": [round(total_pourcentage, 2)]
})

missing_table = pd.concat([missing_table, total_row], ignore_index=True)

missing_table
```

Out[57]:

	Variable	Effectif	Pourcentage (%)
0	workclass	2799	5.73
1	occupation	2809	5.75
2	native-country	857	1.75
3	Total	6465	13.23

Statistiques descriptives

A. Analyse de la variable cible

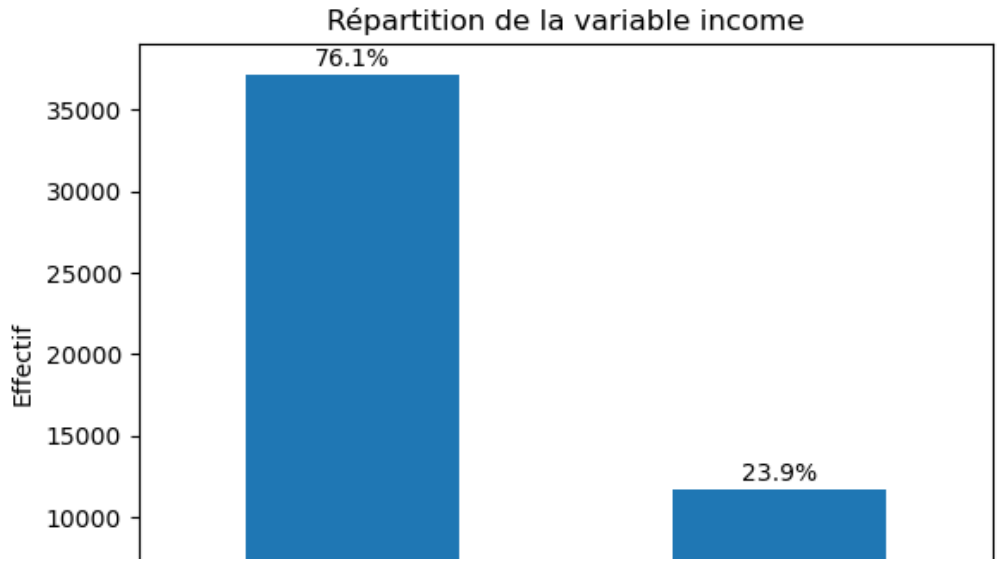
La variable cible "income" correspond au niveau de revenu annuel, avec deux modalités : ≤50K et >50K

Entrée [58]:
```
# Comptage des modalités de la variable cible
income_counts = df["income"].value_counts()

# Création du tableau
income_table = pd.DataFrame({
    "Modalité income": income_counts.index,
    "Effectif": income_counts.values
})

# Calcul des pourcentages
income_table["Pourcentage (%)"] = (income_table["Effectif"] / len(df)) * 100
income_table["Pourcentage (%)"] = income_table["Pourcentage (%)"].round(2)

# Calcul des totaux
total_effectif = income_table["Effectif"].sum()
total_pourcentage = income_table["Pourcentage (%)"].sum()

# Ajout de la ligne Total
total_row = pd.DataFrame({
    "Modalité income": ["Total"],
    "Effectif": [total_effectif],
    "Pourcentage (%)": [round(total_pourcentage, 2)]
})

income_table = pd.concat([income_table, total_row], ignore_index=True)

income_table
```

Out[58]:

	Modalité income	Effectif	Pourcentage (%)
0	<=50K	37155	76.07
1	>50K	11687	23.93
2	Total	48842	100.00

Entrée [59]:
```
counts = df["income"].value_counts()
percentages = counts / counts.sum() * 100

plt.figure()
ax = counts.plot(kind="bar")

plt.title("Répartition de la variable income")
plt.ylabel("Effectif")
plt.xlabel("Income")
plt.xticks(rotation=0)

# Ajout des pourcentages au-dessus des barres
for i, (count, pct) in enumerate(zip(counts, percentages)):
    ax.text(i, count + 500, f"{pct:.1f}%", ha="center", fontsize=10)

plt.show()
```



La variable cible *income* présente un déséquilibre de classes, avec une majorité d'individus percevant un revenu inférieur ou égal à 50K.
Ce déséquilibre devra être pris en compte lors de la phase de modélisation afin d'évaluer correctement les performances des modèles.

B. Analyse exploratoire des variables explicatives

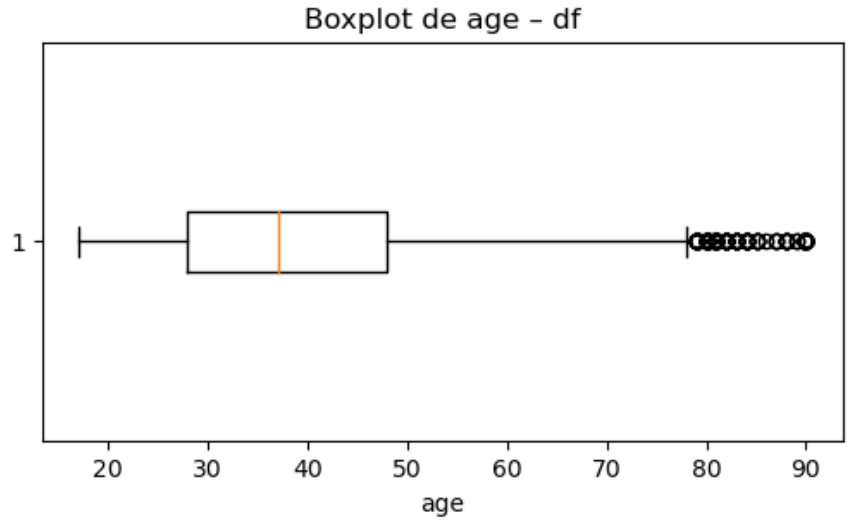
Cette section présente quelques statistiques générales et distributions de variables explicatives afin de mieux comprendre le profil des individus.

Statistiques générales

Entrée [60]:
```
# Variables quantitatives
vars_quant = [
    "age",
    "fnlwgt",
    "educational-num",
    "capital-gain",
    "capital-loss",
    "hours-per-week"
]

pd.set_option("display.max_columns", None)
pd.set_option("display.width", 200)
pd.set_option("display.float_format", "{:.2f}".format)

# Tableau de statistiques descriptives
desc = df[vars_quant].describe().T

# Ajout de la médiane (Q2)
desc["median"] = df[vars_quant].median()

# Réorganisation des colonnes
cols = ["count", "mean", "std", "min", "25%", "median", "50%", "75%", "max"]
desc = desc[cols]

print("Statistiques descriptives – df")
print(desc)
```

```
Statistiques descriptives – df
                   count      mean       std      min       25%    median       50%
75%        max
age              48842.00     38.64     13.71    17.00     28.00     37.00     37.00     4
8.00      90.00
fnlwgt           48842.00 189664.13 105604.03 12285.00 117550.50 178144.50 178144.50 23764
2.00 1490400.00
educational-num  48842.00     10.08      2.57     1.00      9.00     10.00     10.00     1
2.00      16.00
capital-gain     48842.00   1079.07   7452.02     0.00      0.00      0.00      0.00
0.00   99999.00
capital-loss     48842.00     87.50    403.00     0.00      0.00      0.00      0.00
0.00    4356.00
hours-per-week   48842.00     40.42     12.39     1.00     40.00     40.00     40.00     4
5.00      99.00
```

Entrée [61]:
```
for var in vars_quant:
    plt.figure(figsize=(6, 3))
    plt.boxplot(df[var], vert=False)
    plt.title(f"Boxplot de {var} - df")
    plt.xlabel(var)
    plt.show()
```



Commentaires : L'analyse des statistiques descriptives générales met en évidence plusieurs éléments importants.
L'âge moyen des individus est d'environ 38,6 ans, avec une médiane proche (37 ans), ce qui indique une population relativement jeune et centrée autour de l'âge actif. Les valeurs minimales et maximales (17 à 90 ans) montrent toutefois une certaine hétérogénéité.

La variable *hours-per-week* présente une moyenne d'environ 40 heures, avec une médiane identique, traduisant un temps de travail hebdomadaire standard pour la majorité des individus. Néanmoins, des valeurs extrêmes sont observées, pouvant aller jusqu'à 99 heures par semaine.

Les variables *capital-gain* et *capital-loss* sont fortement asymétriques. La majorité des individus présente des valeurs nulles, tandis que quelques observations très élevées expliquent les valeurs maximales importantes. Ces variables nécessiteront une attention particulière lors du pré-traitement afin de limiter l'influence des valeurs extrêmes sur la modélisation.

Enfin, la variable *education-num* montre une dispersion modérée autour d'une moyenne proche de 10, traduisant des niveaux d'éducation relativement variés au sein de la population.

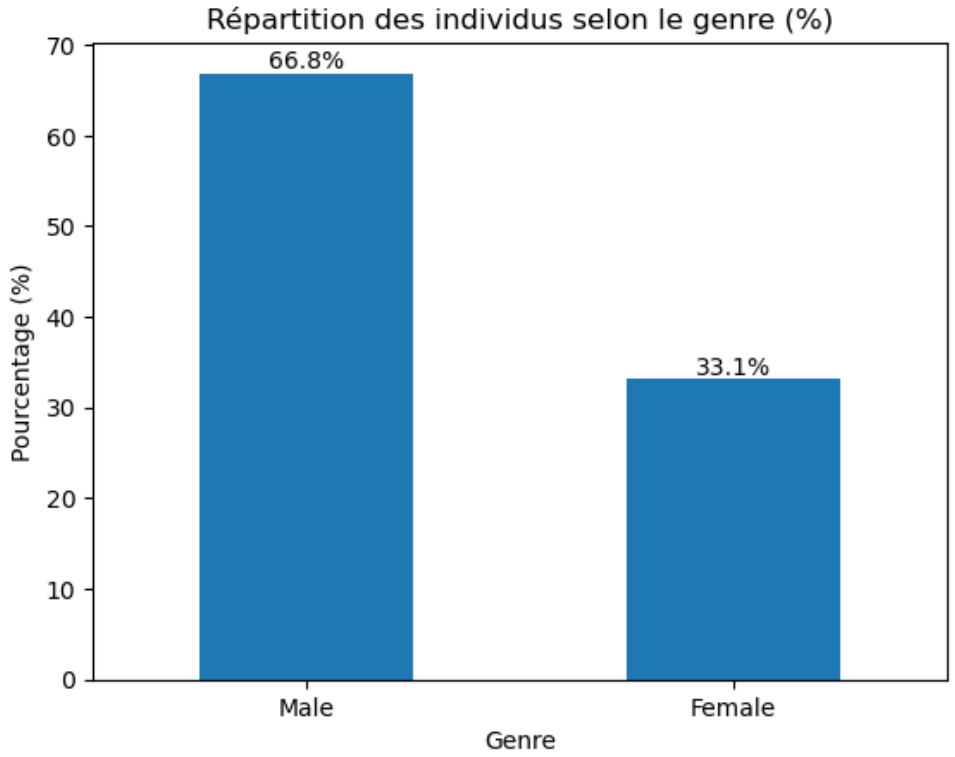
Variable gender

Entrée [62]:
```
# Comptage des modalités de gender
gender_counts = df["gender"].value_counts()

gender_table = pd.DataFrame({
    "Genre": gender_counts.index,
    "Effectif": gender_counts.values
})

# Pourcentages
gender_table["Pourcentage (%)"] = (gender_table["Effectif"] / len(df)) * 100
gender_table["Pourcentage (%)"] = gender_table["Pourcentage (%)"].round(2)

# Totaux
total_row = pd.DataFrame({
    "Genre": ["Total"],
    "Effectif": [gender_table["Effectif"].sum()],
    "Pourcentage (%)": [round(gender_table["Pourcentage (%)"].sum(), 2)]
})

gender_table = pd.concat([gender_table, total_row], ignore_index=True)

gender_table
```

Out[62]:

	Genre	Effectif	Pourcentage (%)
0	Male	32650	66.85
1	Female	16192	33.15
2	Total	48842	100.00

Entrée [63]:
```
plt.figure()
ax = gender_table.iloc[:-1].set_index("Genre")["Pourcentage (%)"].plot(kind="bar")

plt.title("Répartition des individus selon le genre (%)")
plt.ylabel("Pourcentage (%)")
plt.xlabel("Genre")
plt.xticks(rotation=0)

# Affichage des pourcentages
for i, pct in enumerate(gender_table.iloc[:-1]["Pourcentage (%)"]):
    ax.text(i, pct + 0.5, f"{pct:.1f}%", ha="center")

plt.show()
```



Commentaire : On remarque que la population étudiée est majoritairement masculine (66,8%).

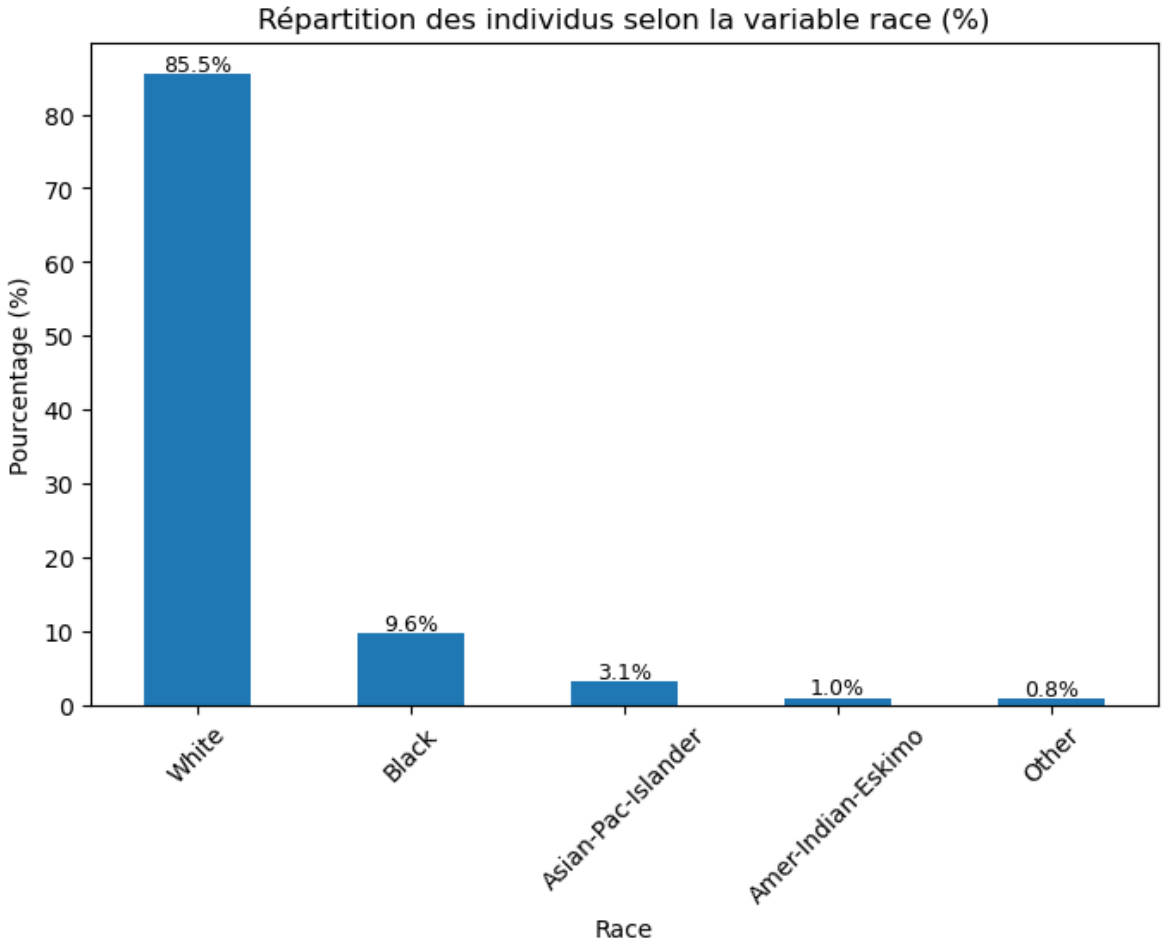
Variable race

Entrée [64]:
```
# Comptage des modalités de race
race_counts = df["race"].value_counts()

race_table = pd.DataFrame({
    "Race": race_counts.index,
    "Effectif": race_counts.values
})

# Pourcentages
race_table["Pourcentage (%)"] = (race_table["Effectif"] / len(df)) * 100
race_table["Pourcentage (%)"] = race_table["Pourcentage (%)"].round(2)

# Totaux
total_row = pd.DataFrame({
    "Race": ["Total"],
    "Effectif": [race_table["Effectif"].sum()],
    "Pourcentage (%)": [round(race_table["Pourcentage (%)"].sum(), 2)]
})

race_table = pd.concat([race_table, total_row], ignore_index=True)

race_table
```

Out[64]:

	Race	Effectif	Pourcentage (%)
0	White	41762	85.50
1	Black	4685	9.59
2	Asian-Pac-Islander	1519	3.11
3	Amer-Indian-Eskimo	470	0.96
4	Other	406	0.83
5	Total	48842	99.99

Entrée [65]:
```
plt.figure(figsize=(8,5))
ax = race_table.iloc[:-1].set_index("Race")["Pourcentage (%)"].plot(kind="bar")

plt.title("Répartition des individus selon la variable race (%)")
plt.ylabel("Pourcentage (%)")
plt.xlabel("Race")
plt.xticks(rotation=45)

# Ajout des pourcentages
for i, pct in enumerate(race_table.iloc[:-1]["Pourcentage (%)"]):
    ax.text(i, pct + 0.5, f"{pct:.1f}%", ha="center", fontsize=9)

plt.show()
```



Commentaire : La population étudiée est majoritairement composée d'individus appartenant à la catégorie *White* (85,5%).
Les autres catégories sont nettement moins représentées, ce qui traduit une distribution déséquilibrée de la variable *race* dans le jeu de données.

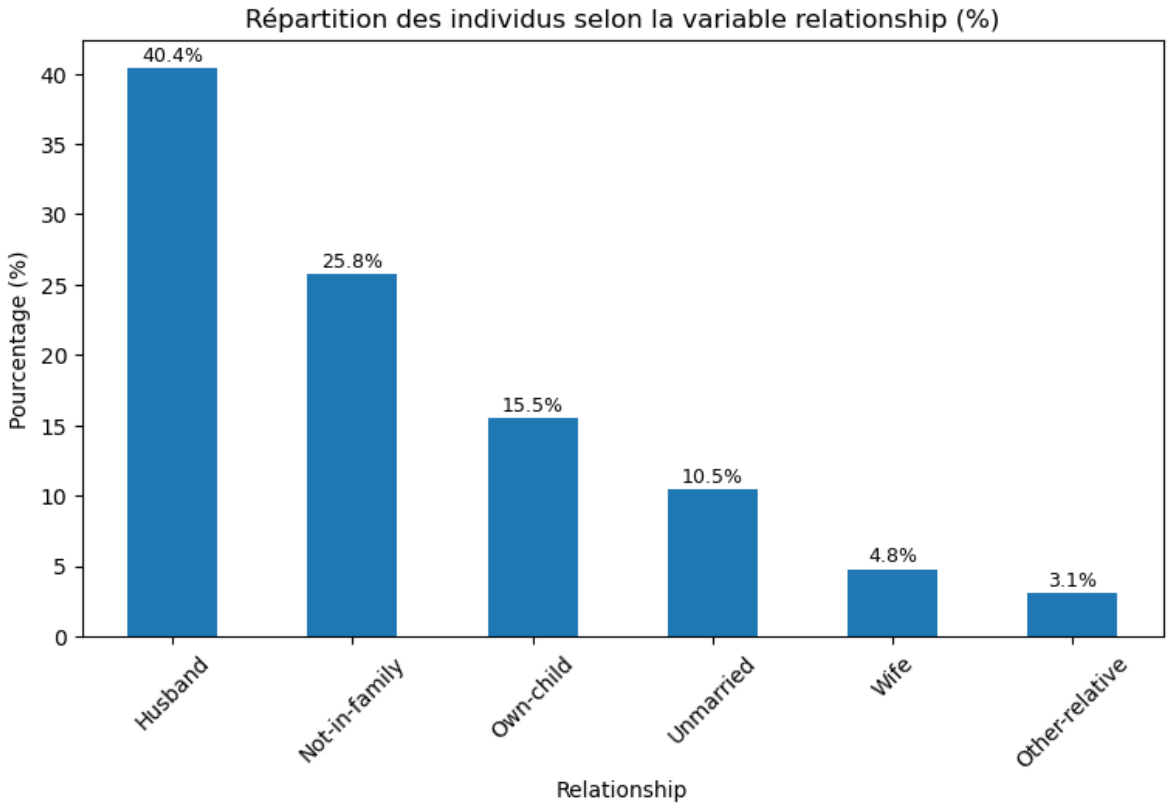
Variable relationship

Entrée [66]:
```
# Comptage des modalités de relationship
relationship_counts = df["relationship"].value_counts()

relationship_table = pd.DataFrame({
    "Relationship": relationship_counts.index,
    "Effectif": relationship_counts.values
})

# Pourcentages
relationship_table["Pourcentage (%)"] = (relationship_table["Effectif"] / len(df)) * 100
relationship_table["Pourcentage (%)"] = relationship_table["Pourcentage (%)"].round(2)

# Totaux
total_row = pd.DataFrame({
    "Relationship": ["Total"],
    "Effectif": [relationship_table["Effectif"].sum()],
    "Pourcentage (%)": [round(relationship_table["Pourcentage (%)"].sum(), 2)]
})

relationship_table = pd.concat([relationship_table, total_row], ignore_index=True)

relationship_table
```

Out[66]:

	Relationship	Effectif	Pourcentage (%)
0	Husband	19716	40.37
1	Not-in-family	12583	25.76
2	Own-child	7581	15.52
3	Unmarried	5125	10.49
4	Wife	2331	4.77
5	Other-relative	1506	3.08
6	Total	48842	99.99

Entrée [67]:
```
plt.figure(figsize=(9,5))
ax = relationship_table.iloc[:-1].set_index("Relationship")["Pourcentage (%)"].plot(kind="

plt.title("Répartition des individus selon la variable relationship (%)")
plt.ylabel("Pourcentage (%)")
plt.xlabel("Relationship")
plt.xticks(rotation=45)

# Ajout des pourcentages au-dessus des barres
for i, pct in enumerate(relationship_table.iloc[:-1]["Pourcentage (%)"]):
    ax.text(i, pct + 0.5, f"{pct:.1f}%", ha="center", fontsize=9)

plt.show()
```



Commentaire : La catégorie *Husband* est la plus représentée dans le jeu de données (40,4%), suivie des catégories *Not-in-family* (25,8%) et *Own-child* (15,5%).
Cette variable apporte une information complémentaire sur la structure familiale des individus et pourra jouer un rôle dans l'explication du niveau de revenu.

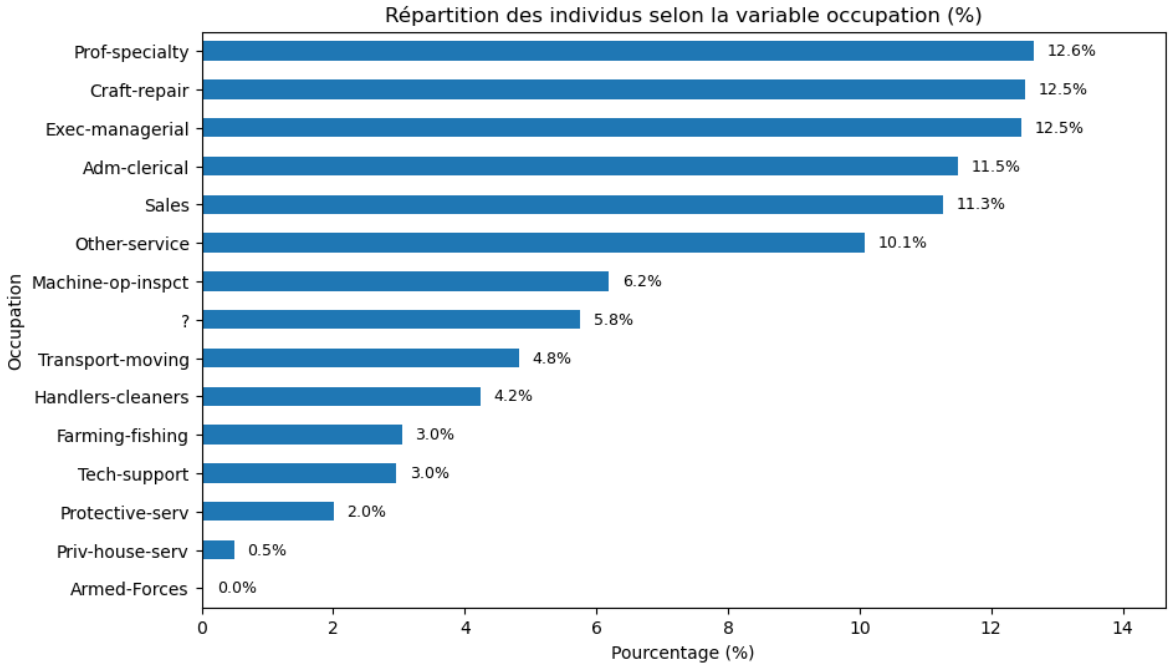
Variable occupation

Entrée [68]:
```
# Comptage des modalités de occupation
occupation_counts = df["occupation"].value_counts()

occupation_table = pd.DataFrame({
    "Occupation": occupation_counts.index,
    "Effectif": occupation_counts.values
})

# Pourcentages
occupation_table["Pourcentage (%)"] = (occupation_table["Effectif"] / len(df)) * 100
occupation_table["Pourcentage (%)"] = occupation_table["Pourcentage (%)"].round(2)

# Totaux
total_row = pd.DataFrame({
    "Occupation": ["Total"],
    "Effectif": [occupation_table["Effectif"].sum()],
    "Pourcentage (%)": [round(occupation_table["Pourcentage (%)"].sum(), 2)]
})

occupation_table = pd.concat([occupation_table, total_row], ignore_index=True)

occupation_table
```

Out[68]:

	Occupation	Effectif	Pourcentage (%)
0	Prof-specialty	6172	12.64
1	Craft-repair	6112	12.51
2	Exec-managerial	6086	12.46
3	Adm-clerical	5611	11.49
4	Sales	5504	11.27
5	Other-service	4923	10.08
6	Machine-op-inspct	3022	6.19
7	?	2809	5.75
8	Transport-moving	2355	4.82
9	Handlers-cleaners	2072	4.24
10	Farming-fishing	1490	3.05
11	Tech-support	1446	2.96

Entrée [69]:
```
plt.figure(figsize=(10,6))

occupation_plot = occupation_table.iloc[:-1].set_index("Occupation")["Pourcentage (%)"]
occupation_plot = occupation_plot.sort_values()  # ordre croissant

ax = occupation_plot.plot(kind="barh")

plt.title("Répartition des individus selon la variable occupation (%)")
plt.xlabel("Pourcentage (%)")
plt.ylabel("Occupation")

# Ajuster la limite de l'axe X
x_max = occupation_plot.max()
plt.xlim(0, x_max + 2)

# Ajout des pourcentages
for i, pct in enumerate(occupation_plot):
    ax.text(pct + 0.2, i, f"{pct:.1f}%", va="center", fontsize=9)

plt.show()
```



Commentaire : La variable *occupation* présente une distribution hétérogène, avec une concentration plus élevée dans certaines catégories professionnelles telles que *Prof-specialty* (12,6%), *Craft-repair* (12,5%) et *Exec-managerial* (12,5%).

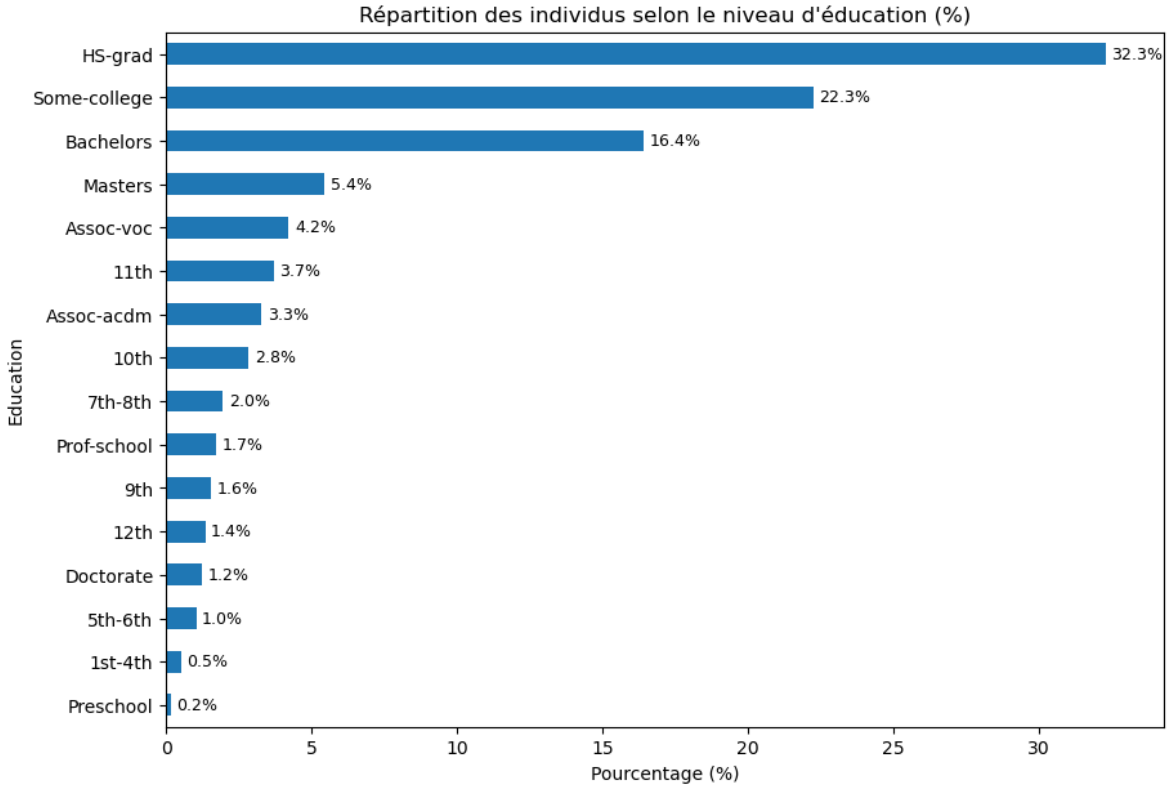
Variable education

Entrée [70]:
```
education_counts = df["education"].value_counts()

education_table = pd.DataFrame({
    "Education": education_counts.index,
    "Effectif": education_counts.values
})

education_table["Pourcentage (%)"] = (education_table["Effectif"] / len(df)) * 100
education_table["Pourcentage (%)"] = education_table["Pourcentage (%)"].round(2)

total_row = pd.DataFrame({
    "Education": ["Total"],
    "Effectif": [education_table["Effectif"].sum()],
    "Pourcentage (%)": [round(education_table["Pourcentage (%)"].sum(), 2)]
})

education_table = pd.concat([education_table, total_row], ignore_index=True)

education_table
```

Out[70]:

	Education	Effectif	Pourcentage (%)
0	HS-grad	15784	32.32
1	Some-college	10878	22.27
2	Bachelors	8025	16.43
3	Masters	2657	5.44
4	Assoc-voc	2061	4.22
5	11th	1812	3.71
6	Assoc-acdm	1601	3.28
7	10th	1389	2.84
8	7th-8th	955	1.96
9	Prof-school	834	1.71
10	9th	756	1.55
11	12th	657	1.35
12	Doctorate	594	1.22
13	5th-6th	509	1.04
14	1st-4th	247	0.51
15	Preschool	83	0.17
16	Total	48842	100.02

Entrée [71]:
```
plt.figure(figsize=(10,7))

education_plot = education_table.iloc[:-1].set_index("Education")["Pourcentage (%)"]
education_plot = education_plot.sort_values()  # ordre croissant

ax = education_plot.plot(kind="barh")

plt.title("Répartition des individus selon le niveau d'éducation (%)")
plt.xlabel("Pourcentage (%)")
plt.ylabel("Education")

x_max = education_plot.max()
plt.xlim(0, x_max + 2)

# Ajout des pourcentages
for i, pct in enumerate(education_plot):
    ax.text(pct + 0.2, i, f"{pct:.1f}%", va="center", fontsize=9)

plt.show()
```



Commentaire : La variable *education* montre une forte concentration sur certains niveaux d'éducation, notamment *HS-grad* (32,3%) et *Some-college* (22,3%).

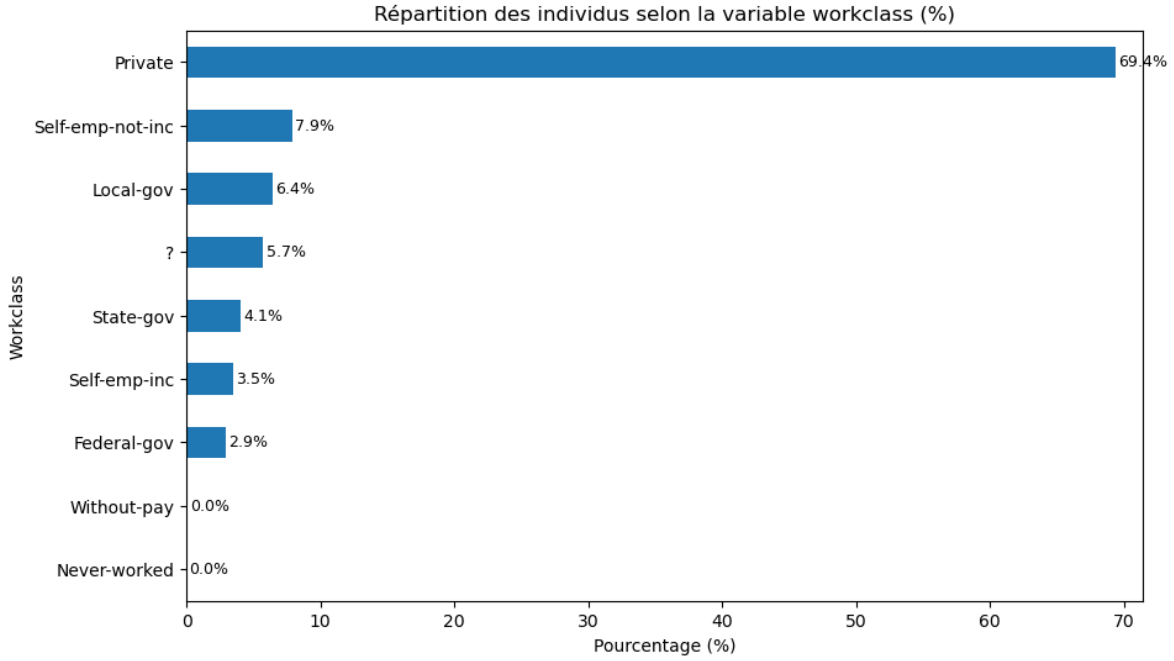
Variable Workclass

Entrée [72]:
```
workclass_counts = df["workclass"].value_counts()

workclass_table = pd.DataFrame({
    "Workclass": workclass_counts.index,
    "Effectif": workclass_counts.values
})

workclass_table["Pourcentage (%)"] = (workclass_table["Effectif"] / len(df)) * 100
workclass_table["Pourcentage (%)"] = workclass_table["Pourcentage (%)"].round(2)

total_row = pd.DataFrame({
    "Workclass": ["Total"],
    "Effectif": [workclass_table["Effectif"].sum()],
    "Pourcentage (%)": [round(workclass_table["Pourcentage (%)"].sum(), 2)]
})

workclass_table = pd.concat([workclass_table, total_row], ignore_index=True)

workclass_table
```

Out[72]:

	Workclass	Effectif	Pourcentage (%)
0	Private	33906	69.42
1	Self-emp-not-inc	3862	7.91
2	Local-gov	3136	6.42
3	?	2799	5.73
4	State-gov	1981	4.06
5	Self-emp-inc	1695	3.47
6	Federal-gov	1432	2.93
7	Without-pay	21	0.04
8	Never-worked	10	0.02
9	Total	48842	100.00

Entrée [73]:
```
plt.figure(figsize=(10,6))

workclass_plot = workclass_table.iloc[:-1].set_index("Workclass")["Pourcentage (%)"]
workclass_plot = workclass_plot.sort_values()

ax = workclass_plot.plot(kind="barh")

plt.title("Répartition des individus selon la variable workclass (%)")
plt.xlabel("Pourcentage (%)")
plt.ylabel("Workclass")

x_max = workclass_plot.max()
plt.xlim(0, x_max + 2)

for i, pct in enumerate(workclass_plot):
    ax.text(pct + 0.2, i, f"{pct:.1f}%", va="center", fontsize=9)

plt.show()
```



Commentaire : La variable *workclass* est majoritairement dominée par la catégorie *Private* (69,4%).
Les catégories *Self-emp-not-inc* (7,9%) et *Local-gov* (6,4%) représentent des parts plus modestes de la population étudiée, tandis que certaines catégories restent très marginales.

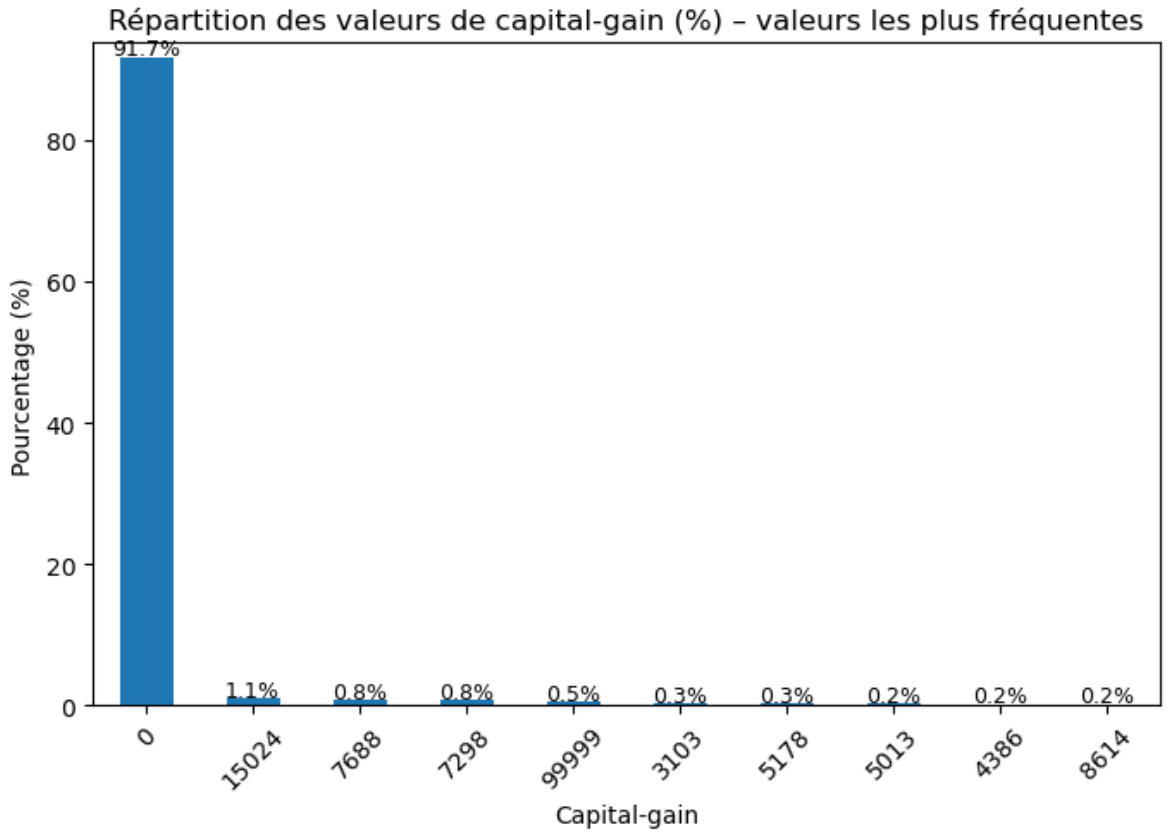
Variable capital-gain

Entrée [75]:
```
# Comptage des valeurs brutes
capital_gain_counts = df["capital-gain"].value_counts()

capital_gain_table = pd.DataFrame({
    "Capital-gain": capital_gain_counts.index,
    "Effectif": capital_gain_counts.values
})

capital_gain_table["Pourcentage (%)"] = (capital_gain_table["Effectif"] / len(df)) * 100
capital_gain_table["Pourcentage (%)"] = capital_gain_table["Pourcentage (%)"].round(2)

# Affichage des valeurs les plus fréquentes (pour lisibilité)
capital_gain_table.head(10)
```

Out[75]:

	Capital-gain	Effectif	Pourcentage (%)
0	0	44807	91.74
1	15024	513	1.05
2	7688	410	0.84
3	7298	364	0.75
4	99999	244	0.50
5	3103	152	0.31
6	5178	146	0.30
7	5013	117	0.24
8	4386	108	0.22
9	8614	82	0.17

Entrée [76]:
```
plt.figure(figsize=(8,5))

capital_gain_plot = capital_gain_table.head(10).set_index("Capital-gain")["Pourcentage (%)

ax = capital_gain_plot.plot(kind="bar")

plt.title("Répartition des valeurs de capital-gain (%) – valeurs les plus fréquentes")
plt.ylabel("Pourcentage (%)")
plt.xlabel("Capital-gain")
plt.xticks(rotation=45)

y_max = capital_gain_plot.max()
plt.ylim(0, y_max + 2)

for i, pct in enumerate(capital_gain_plot):
    ax.text(i, pct + 0.3, f"{pct:.1f}%", ha="center", fontsize=9)

plt.show()
```



Commentaire : La variable *capital-gain* est fortement concentrée sur la valeur nulle (0), qui représente une part très majoritaire des observations (91,7%).
Les autres valeurs de *capital-gain* apparaissent de manière très marginale, chacune représentant une proportion très faible de la population étudiée.

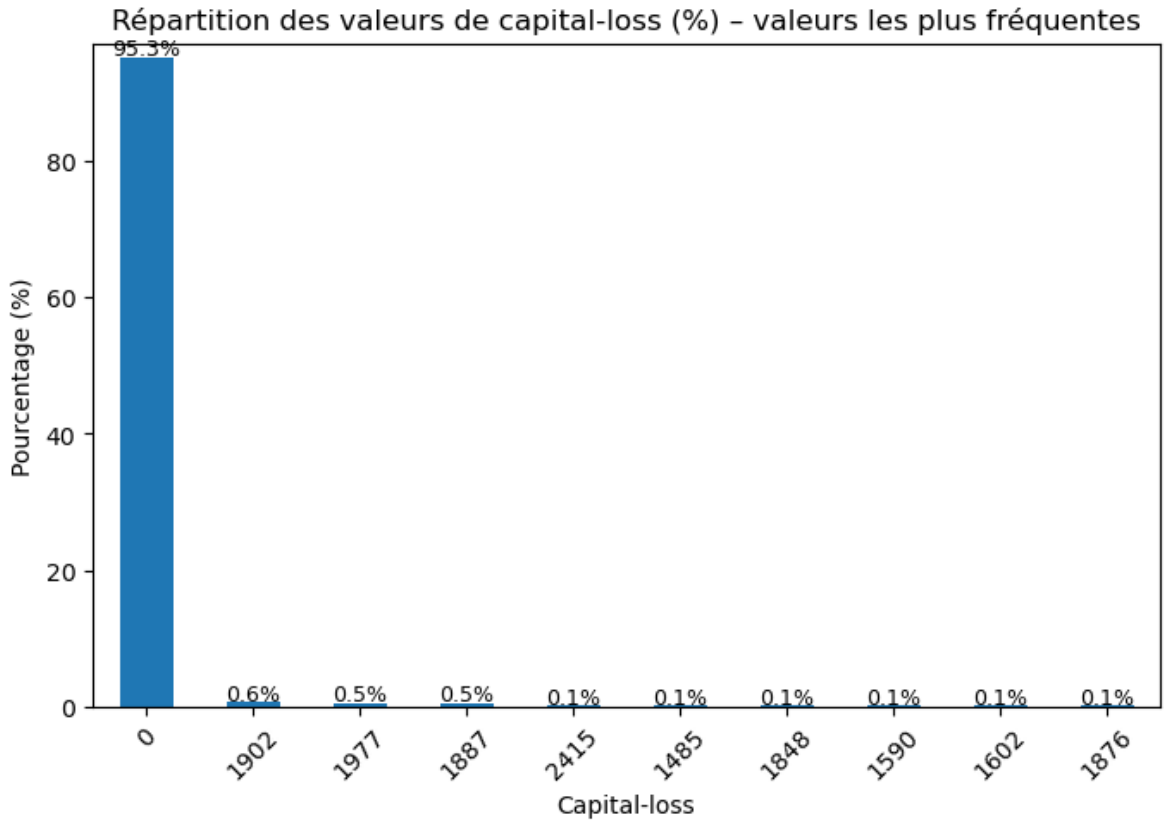
Variable capital-loss

Entrée [77]:
```
# Comptage des valeurs brutes
capital_loss_counts = df["capital-loss"].value_counts()

capital_loss_table = pd.DataFrame({
    "Capital-loss": capital_loss_counts.index,
    "Effectif": capital_loss_counts.values
})

capital_loss_table["Pourcentage (%)"] = (capital_loss_table["Effectif"] / len(df)) * 100
capital_loss_table["Pourcentage (%)"] = capital_loss_table["Pourcentage (%)"].round(2)

# Affichage des valeurs les plus fréquentes pour la lisibilité
capital_loss_table.head(10)
```

Out[77]:

	Capital-loss	Effectif	Pourcentage (%)
0	0	46560	95.33
1	1902	304	0.62
2	1977	253	0.52
3	1887	233	0.48
4	2415	72	0.15
5	1485	71	0.15
6	1848	67	0.14
7	1590	62	0.13
8	1602	62	0.13
9	1876	59	0.12

Entrée [78]:
```
plt.figure(figsize=(8,5))

capital_loss_plot = capital_loss_table.head(10).set_index("Capital-loss")["Pourcentage (%)

ax = capital_loss_plot.plot(kind="bar")

plt.title("Répartition des valeurs de capital-loss (%) – valeurs les plus fréquentes")
plt.ylabel("Pourcentage (%)")
plt.xlabel("Capital-loss")
plt.xticks(rotation=45)

y_max = capital_loss_plot.max()
plt.ylim(0, y_max + 2)

for i, pct in enumerate(capital_loss_plot):
    ax.text(i, pct + 0.3, f"{pct:.1f}%", ha="center", fontsize=9)

plt.show()
```



Commentaire : La variable *capital-loss* est également très majoritairement concentrée sur la valeur nulle (0), qui représente une part écrasante des observations (95,3%).
Les autres valeurs de *capital-loss* apparaissent de manière marginale et concernent une faible proportion des individus.

Variable age

Entrée [84]:
```
age_counts = df["age"].value_counts().sort_index()

age_table = pd.DataFrame({
    "Âge": age_counts.index,
    "Effectif": age_counts.values
})

age_table["Pourcentage (%)"] = (age_table["Effectif"] / len(df)) * 100
age_table["Pourcentage (%)"] = age_table["Pourcentage (%)"].round(2)

# Affichage des âges les plus fréquents
age_table.sort_values("Effectif", ascending=False).head(10)
```

Out[84]:

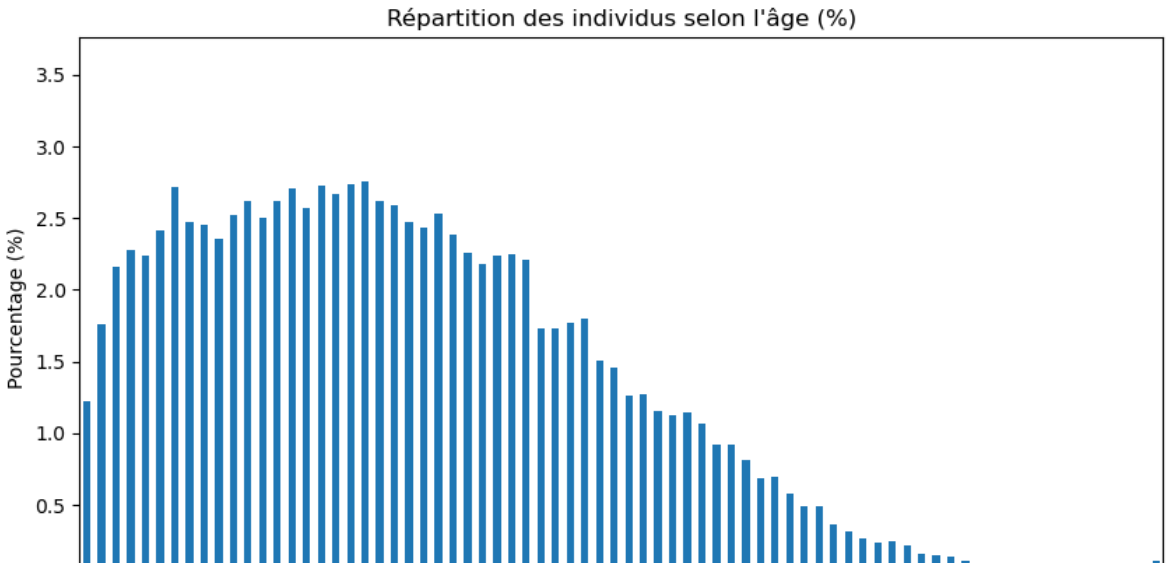
	Âge	Effectif	Pourcentage (%)
19	36	1348	2.76
18	35	1337	2.74
16	33	1335	2.73
6	23	1329	2.72
14	31	1325	2.71
17	34	1303	2.67
11	28	1280	2.62
20	37	1280	2.62
13	30	1278	2.62
21	38	1264	2.59

Entrée [89]:
```
age_counts = df["age"].value_counts().sort_index()

age_table = pd.DataFrame({
    "Âge": age_counts.index,
    "Effectif": age_counts.values
})

age_table["Pourcentage (%)"] = (age_table["Effectif"] / len(df)) * 100
age_table["Pourcentage (%)"] = age_table["Pourcentage (%)"].round(2)

age_table
```

Out[89]:

	Âge	Effectif	Pourcentage (%)
0	17	595	1.22
1	18	862	1.76
2	19	1053	2.16
3	20	1113	2.28
4	21	1096	2.24
5	22	1178	2.41
6	23	1329	2.72
7	24	1206	2.47
8	25	1195	2.45
9	26	1153	2.36
10	27	1232	2.52
11	28	1280	2.62

Entrée [80]:
```
plt.figure(figsize=(10,5))

age_plot = age_table.set_index("Âge")["Pourcentage (%)"]

ax = age_plot.plot(kind="bar")

plt.title("Répartition des individus selon l'âge (%)")
plt.ylabel("Pourcentage (%)")
plt.xlabel("Âge")

# Limiter l'affichage des âges sur l'axe X
plt.xticks(
    ticks=range(0, len(age_plot), 5),
    labels=age_plot.index[::5],
    rotation=0
)

y_max = age_plot.max()
plt.ylim(0, y_max + 1)

plt.show()
```



Commentaire : La variable *age* présente une distribution étendue allant de 17 à 90 ans, avec un âge médian de 37 ans.
La population étudiée est majoritairement composée d'individus âgés entre environ 30 et 50 ans, avec une concentration plus marquée autour de la quarantaine (2,76 % ont 36 ans).

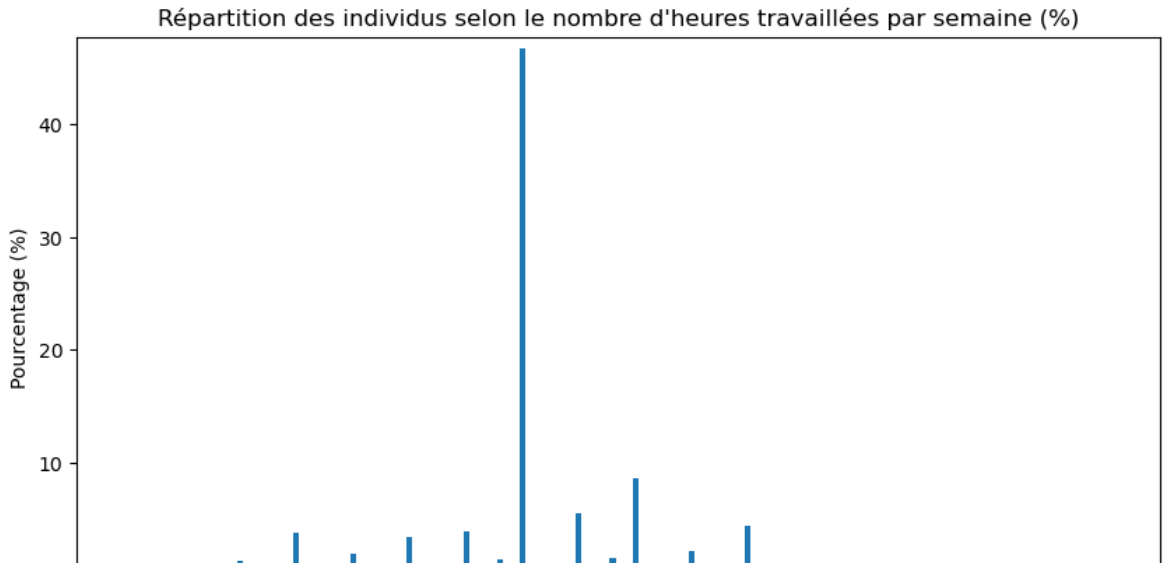
Variable hours-peer-week

Entrée [81]:
```
hpw_counts = df["hours-per-week"].value_counts().sort_index()

hpw_table = pd.DataFrame({
    "Heures par semaine": hpw_counts.index,
    "Effectif": hpw_counts.values
})

hpw_table["Pourcentage (%)"] = (hpw_table["Effectif"] / len(df)) * 100
hpw_table["Pourcentage (%)"] = hpw_table["Pourcentage (%)"].round(2)

# Affichage des modalités les plus fréquentes
hpw_table.sort_values("Effectif", ascending=False).head(10)
```

Out[81]:

	Heures par semaine	Effectif	Pourcentage (%)
39	40	22803	46.69
49	50	4246	8.69
44	45	2717	5.56
59	60	2177	4.46
34	35	1937	3.97
19	20	1862	3.81
29	30	1700	3.48
54	55	1051	2.15
24	25	958	1.96
47	48	770	1.58

Entrée [82]:
```
plt.figure(figsize=(10,5))

hpw_plot = hpw_table.set_index("Heures par semaine")["Pourcentage (%)"]

ax = hpw_plot.plot(kind="bar")

plt.title("Répartition des individus selon le nombre d'heures travaillées par semaine (%)"
plt.ylabel("Pourcentage (%)")
plt.xlabel("Heures par semaine")

# Espacement des labels sur l'axe X
plt.xticks(
    ticks=range(0, len(hpw_plot), 5),
    labels=hpw_plot.index[::5],
    rotation=0
)

y_max = hpw_plot.max()
plt.ylim(0, y_max + 1)

plt.show()
```



Commentaire : La variable *hours-per-week* est fortement concentrée autour de la valeur de 40 heures par semaine, qui représente la modalité la plus fréquente (46,69 %).
La majorité des individus travaille entre environ 35 et 45 heures par semaine, bien que des valeurs plus extrêmes soient également observées.

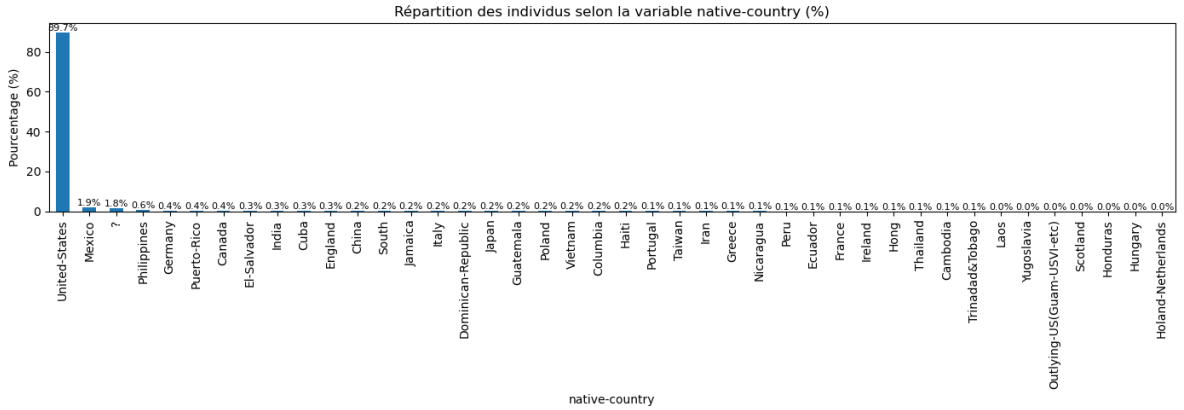
Variable native-country

Entrée [90]:
```
# Tableau effectif + pourcentage pour native-country
tab_native_country = (
    df["native-country"]
    .value_counts(dropna=False)
    .reset_index()
)

tab_native_country.columns = ["native-country", "Effectif"]

tab_native_country["Pourcentage (%)"] = (
    tab_native_country["Effectif"] / tab_native_country["Effectif"].sum() * 100
).round(2)

tab_native_country
```

Out[90]:

	native-country	Effectif	Pourcentage (%)
0	United-States	43832	89.74
1	Mexico	951	1.95
2	?	857	1.75
3	Philippines	295	0.60
4	Germany	206	0.42
5	Puerto-Rico	184	0.38
6	Canada	182	0.37
7	El-Salvador	155	0.32
8	India	151	0.31
9	Cuba	138	0.28
10	England	127	0.26
11	China	122	0.25
12	South	115	0.24
13	Jamaica	106	0.22
14	Italy	105	0.21
15	Dominican-Republic	103	0.21
16	Japan	92	0.19
17	Guatemala	88	0.18
18	Poland	87	0.18
19	Vietnam	86	0.18
20	Columbia	85	0.17
21	Haiti	75	0.15
22	Portugal	67	0.14
23	Taiwan	65	0.13
24	Iran	59	0.12
25	Greece	49	0.10
26	Nicaragua	49	0.10
27	Peru	46	0.09
28	Ecuador	45	0.09
29	France	38	0.08
30	Ireland	37	0.08
31	Hong	30	0.06
32	Thailand	30	0.06
33	Cambodia	28	0.06
34	Trinadad&Tobago	27	0.06
35	Laos	23	0.05
36	Yugoslavia	23	0.05
37	Outlying-US(Guam-USVI-etc)	23	0.05
38	Scotland	21	0.04
39	Honduras	20	0.04
40	Hungary	19	0.04
41	Holand-Netherlands	1	0.00

Entrée [91]:
```
plt.figure(figsize=(14, 5))

percentages = (
    df["native-country"]
    .value_counts(normalize=True) * 100
)

percentages.plot(kind="bar")

plt.title("Répartition des individus selon la variable native-country (%)")
plt.ylabel("Pourcentage (%)")
plt.xlabel("native-country")
plt.xticks(rotation=90)

# ajout des pourcentages au-dessus des barres
for i, v in enumerate(percentages.values):
    plt.text(i, v, f"{v:.1f}%", ha="center", va="bottom", fontsize=8)

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



Commentaire : La variable native-country est très fortement dominée par la modalité United-States, qui représente une large majorité des individus. Les autres pays apparaissent avec des effectifs et des pourcentages très faibles, ce qui traduit une forte concentration géographique de la population étudiée.

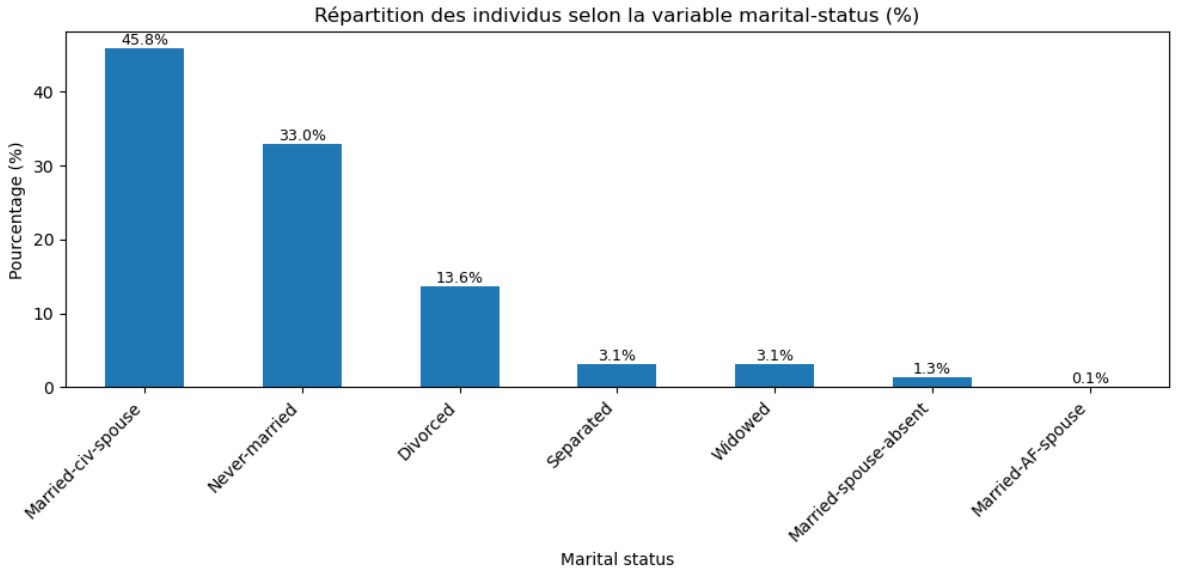
Variable marital-status

Entrée [92]:
```
# Tableau effectif + pourcentage pour marital-status
tab_marital = (
    df["marital-status"]
    .value_counts(dropna=False)
    .reset_index()
)

tab_marital.columns = ["marital-status", "Effectif"]

tab_marital["Pourcentage (%)"] = (
    tab_marital["Effectif"] / tab_marital["Effectif"].sum() * 100
).round(2)

# ajout ligne Total
total = pd.DataFrame([["Total", tab_marital["Effectif"].sum(), 100]],
                     columns=tab_marital.columns)

tab_marital = pd.concat([tab_marital, total], ignore_index=True)

tab_marital
```

Out[92]:

	marital-status	Effectif	Pourcentage (%)
0	Married-civ-spouse	22379	45.82
1	Never-married	16117	33.00
2	Divorced	6633	13.58
3	Separated	1530	3.13
4	Widowed	1518	3.11
5	Married-spouse-absent	628	1.29
6	Married-AF-spouse	37	0.08
7	Total	48842	100.00

Entrée [93]:
```
plt.figure(figsize=(10, 5))

percentages = df["marital-status"].value_counts(normalize=True) * 100
percentages.plot(kind="bar")

plt.title("Répartition des individus selon la variable marital-status (%)")
plt.xlabel("Marital status")
plt.ylabel("Pourcentage (%)")
plt.xticks(rotation=45, ha="right")

for i, v in enumerate(percentages.values):
    plt.text(i, v, f"{v:.1f}%", ha="center", va="bottom", fontsize=9)

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



Commentaire : La variable marital-status est dominée par la modalité Married-civ-spouse (45,8%), suivie par Never-married (33,0%). Les autres statuts matrimoniaux présentent des proportions plus faibles, traduisant une diversité de situations familiales au sein de la population étudiée.

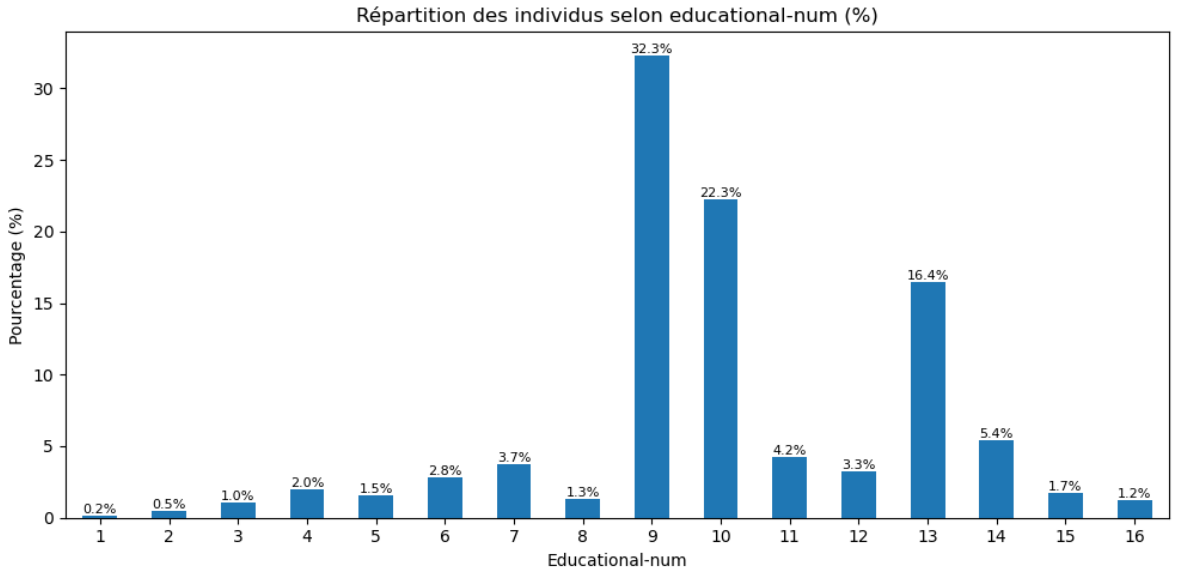
Variable educational-num

Entrée [95]:
```
tab_educ_num = (
    df["educational-num"]
    .value_counts()
    .sort_index()
    .reset_index()
)

tab_educ_num.columns = ["educational-num", "Effectif"]

tab_educ_num["Pourcentage (%)"] = (
    tab_educ_num["Effectif"] / tab_educ_num["Effectif"].sum() * 100
).round(2)

# ajout ligne Total
total = pd.DataFrame([[ "Total", tab_educ_num["Effectif"].sum(), 100 ]],
                     columns=tab_educ_num.columns)

tab_educ_num = pd.concat([tab_educ_num, total], ignore_index=True)

tab_educ_num
```

Out[95]:

	educational-num	Effectif	Pourcentage (%)
0	1	83	0.17
1	2	247	0.51
2	3	509	1.04
3	4	955	1.96
4	5	756	1.55
5	6	1389	2.84
6	7	1812	3.71
7	8	657	1.35
8	9	15784	32.32
9	10	10878	22.27
10	11	2061	4.22
11	12	1601	3.28
12	13	8025	16.43
13	14	2657	5.44
14	15	834	1.71
15	16	594	1.22
16	Total	48842	100.00

Entrée [96]:
```
plt.figure(figsize=(10, 5))

percentages = (
    df["educational-num"]
    .value_counts(normalize=True)
    .sort_index() * 100
)

percentages.plot(kind="bar")

plt.title("Répartition des individus selon educational-num (%)")
plt.xlabel("Educational-num")
plt.ylabel("Pourcentage (%)")
plt.xticks(rotation=0)

for i, v in enumerate(percentages.values):
    plt.text(i, v, f"{v:.1f}%", ha="center", va="bottom", fontsize=8)

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



Commentaire : La variable educational-num présente une concentration autour des niveaux intermédiaires, notamment les valeurs 9 (32,3%) et 10 (22,2%), correspondant aux niveaux de formation les plus représentés dans l'échantillon. Les niveaux extrêmes apparaissent avec des pourcentages plus faibles.

Conclusion

L'analyse exploratoire du jeu de données met en évidence une population majoritairement composée d'adultes en âge d'activité, avec un âge médian de 37 ans et une concentration notable des individus entre 30 et 50 ans. La population étudiée est majoritairement masculine (66,9 %), provenant des United-stated (89,74%) et appartient principalement à la catégorie raciale *White* (85,5 %), ce qui reflète un déséquilibre important de certaines variables socio-démographiques. Du point de vue de la structure familiale, la catégorie *Husband* est la plus représentée (40,4 %), suivie des individus *Not-in-family* (25,8 %). Le niveau d'éducation est également marqué par une forte concentration sur certains diplômes, notamment *HS-grad* (32,3 %) et *Some-college* (22,3 %).

Les variables liées à l'activité professionnelle montrent une forte hétérogénéité. Les catégories *Prof-specialty*, *Craft-repair* et *Exec-managerial* concentrent chacune environ 12 % des individus, tandis que d'autres catégories restent marginales. Le temps de travail hebdomadaire est fortement centré autour de 40 heures (46,7 %), traduisant une organisation du travail relativement standard pour la majorité des individus.

Enfin, l'analyse de la variable cible *income* révèle un déséquilibre de classes, avec une majorité d'individus percevant un revenu annuel inférieur ou égal à 50K (76,1 %), contre 23,9 % au-delà de ce seuil. Les variables

capital-gain et *capital-loss* présentent quant à elles des distributions fortement asymétriques, caractérisées par une prédominance de valeurs nulles (respectivement 91,7 % et 95,3 %).

Dans l'ensemble, cette phase d'exploration met en évidence un jeu de données hétérogène, comportant des variables numériques et catégorielles, des distributions déséquilibrées ainsi que des valeurs manquantes codées par le caractère « ? ».