A common problem when creating models to generate business value from data is that the datasets can be so large that it can take days for the model to generate predictions. Ensuring that your dataset is stored as efficiently as possible is crucial for allowing these models to run on a more reasonable timescale without having to reduce the size of the dataset.

You've been hired by a major online data science training provider called *Training Data Ltd.* to clean up one of their largest customer datasets. This dataset will eventually be used to predict whether their students are looking for a new job or not, information that they will then use to direct them to prospective recruiters.

You've been given access to `customer_train.csv`, which is a subset of their entire customer dataset, so you can create a proof-of-concept of a much more efficient storage solution. The dataset contains anonymized student information, and whether they were looking for a new job or not during training:

Column	Description
`student_id`	A unique ID for each student.
`city`	A code for the city the student lives in.
`city_development_index`	A scaled development index for the city.
`gender`	The student's gender.
`relevant_experience`	An indicator of the student's work relevant experience.
`enrolled_university`	The type of university course enrolled in (if any).
`education_level`	The student's education level.
`major_discipline`	The educational discipline of the student.
`experience`	The student's total work experience (in years).
`company_size`	The number of employees at the student's current employer.
`company_type`	The type of company employing the student.
`last_new_job`	The number of years between the student's current and previous jobs.
`training_hours`	The number of hours of training completed.
`job_change`	An indicator of whether the student is looking for a new job (`1`) or not (`0`).

```python
# Import necessary libraries
import pandas as pd

# Load the dataset
ds_jobs = pd.read_csv("customer_train.csv")

# View the dataset
ds_jobs.head(20)
```

	s...		city_development_ind...		relevant_experience	enrolled_universi...	education...
0	8949	city_103	0.92	Male	Has relevant experience	no_enrollment	Graduate
1	29725	city_40	0.776	Male	No relevant experience	no_enrollment	Graduate
2	11561	city_21	0.624	null	No relevant experience	Full time course	Graduate
3	33241	city_115	0.789	null	No relevant experience	null	Graduate
4	666	city_162	0.767	Male	Has relevant experience	no_enrollment	Masters
5	21651	city_176	0.764	null	Has relevant experience	Part time course	Graduate
6	28806	city_160	0.92	Male	Has relevant experience	no_enrollment	High Scho
7	402	city_46	0.762	Male	Has relevant experience	no_enrollment	Graduate
8	27107	city_103	0.92	Male	Has relevant experience	no_enrollment	Graduate
9	699	city_103	0.92	null	Has relevant experience	no_enrollment	Graduate
10	29452	city_21	0.624	null	No relevant experience	Full time course	High Scho
11	23853	city_103	0.92	Male	Has relevant experience	no_enrollment	Graduate
12	25619	city_61	0.913	Male	Has relevant experience	no_enrollment	Graduate
13	5826	city_21	0.624	Male	No relevant experience	null	null
14	8722	city_21	0.624	null	No relevant experience	Full time course	High Scho
15	6588	city_114	0.926	Male	Has relevant experience	no_enrollment	Graduate

Rows: 20 Expand

```
# Create a copy of ds_jobs for transforming
ds_jobs_transformed = ds_jobs.copy()

# Start coding here. Use as many cells as you like!



# convert columns with in64 -> int32
ds_jobs_transformed["student_id"] = ds_jobs_transformed["student_id"].astype("int32")

ds_jobs_transformed["training_hours"] = ds_jobs_transformed["training_hours"].astype("int32")

# convert columns with float64 -> float16

ds_jobs_transformed = ds_jobs_transformed.astype({
    "city_development_index":"float16",
    "job_change":"float16"

})

# two-factor categories -> bool:


relevent_map = {"Has relevant experience":1,
               "No relevant experience":0}

ds_jobs_transformed["gender"].replace(gender_map, inplace = True)
ds_jobs_transformed["relevant_experience"].replace(relevent_map, inplace = True)

ds_jobs_transformed = ds_jobs_transformed.astype({
    "relevant_experience":"bool",
    "job_change":"bool",
})


# convert nominal columns(doesn't have natural order eg: orange,blue,... or it's not comparable) -> category

ds_jobs_transformed = ds_jobs_transformed.astype({
    "city":"category",
    "gender":"category",
   # "relevant_experience":"category",  2 factor category -> bool
    "major_discipline":"category",
    "company_type":"category"
})

# convert ordinal columns -> ordered categories.
enrolled_university_order = ["no_enrollment", "Part time course", "Full time course"]
education_level_order = ["High School", "Graduate", "Masters"]
experience_order = ["<1"] + [str(i) for i in range(1,21)] + [">20"]
company_size_order = ["<10","10-49", "50-99", "100-499","500-999","1000-4999","5000-9999","10000+"]
last_new_job_order = ["never","1","2","3","4",">4"]

ds_jobs_transformed["enrolled_university"] = pd.Categorical(ds_jobs_transformed["enrolled_university"],
                                                           categories = enrolled_university_order,
                                                           ordered = True)

ds_jobs_transformed["education_level"] = pd.Categorical(ds_jobs_transformed["education_level"],
                                                       categories = education_level_order,
                                                       ordered = True)

ds_jobs_transformed["experience"] = pd.Categorical(ds_jobs_transformed["experience"],
                                                  categories = experience_order,
                                                  ordered= True)

ds_jobs_transformed["company_size"] = pd.Categorical(ds_jobs_transformed["company_size"],
                                                    categories = company_size_order,
                                                    ordered= True)

ds_jobs_transformed["last_new_job"] = pd.Categorical(ds_jobs_transformed["last_new_job"],
                                                    categories= last_new_job_order,
                                                    ordered=True)
```

```
# filtring the dataframe: (only contain students with 10 or more years of experience at companies with at least 1000 employees, as
their recruiter base is suited to more experienced professionals at enterprise companies.)

is_experience_sup_than10 = ds_jobs_transformed["experience"] >= "10"
is_comp_size_1000 = ds_jobs_transformed["company_size"] >= "1000-4999"

ds_jobs_transformed = ds_jobs_transformed[is_experience_sup_than10 & is_comp_size_1000]



# View the dtypes
print(ds_jobs_transformed.dtypes)
print("!---------------------------------------------------------------------------!")
# memory usage check:
print("Our initial data:")
print(ds_jobs.info())
print("---------------------------------------------------------------------------")
print("After processing the data:")
print(ds_jobs_transformed.info())
```

```
student_id                  int32
city                     category
city_development_index    float16
gender                   category
relevant_experience          bool
enrolled_university      category
education_level          category
major_discipline         category
experience               category
company_size             category
company_type             category
last_new_job             category
training_hours              int32
job_change                   bool
dtype: object
!---------------------------------------------------------------------------!
Our initial data:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19158 entries, 0 to 19157
Data columns (total 14 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   student_id              19158 non-null  int64
 1   city                    19158 non-null  object
 2   city_development_index  19158 non-null  float64
 3   gender                  14650 non-null  object
 4   relevant_experience     19158 non-null  object
 5   enrolled_university     18772 non-null  object
 6   education_level         18698 non-null  object
```