In [1]:
```
%load_ext watermark
```

In [2]:
```
%watermark
```

```
Last updated: 2023-07-06T16:14:19.677262+03:00

Python implementation: CPython
Python version       : 3.7.16
IPython version      : 7.34.0

Compiler    : Clang 14.0.0 (clang-1400.0.29.202)
OS          : Darwin
Release     : 22.5.0
Machine     : x86_64
Processor   : i386
CPU cores   : 8
Architecture: 64bit
```

In [3]:
```
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

In [4]:
```
import pandas as pd
import single_gen_func as sgf
from sdv.metadata import SingleTableMetadata
from sdv.evaluation.single_table import get_column_plot
```

Базовая конфигурация

В данной секции необходимо определить базовую конфигурацию данных, с которыми идет работа. Это включает в себя:

1. Создание списка из имен полей, содержащих значения типа дата. Если такие отсутствуют, можно пропустить данный пункт
2. Создание pandas DataFrame через вызов функции read_csv
3. Создания словаря, определяющего поля типа sdtype=id и их шаблоны регулярных выражений. При отсутствии шаблона передается None, в таком случае генерируется числовая или символьная инкрементальная последовательность. Подробнее обо всем в документации: https://docs.sdv.dev/sdv/reference/metadata-spec/sdtypes
4. Создание строки, содержащей название первичного ключа

In [5]:
```
dates = ['scheduled_departure',
         'scheduled_arrival',
         'actual_departure',
         'actual_arrival',
         'constraint_date']
```

In [6]:
```
df = sgf.read_csv('csvs/flights.csv', 5000, dates)
df.head(5)
```

Out[6]:

	flight_id	flight_no	scheduled_departure	scheduled_arrival	departure_airport	ari
0	22020	PG0569	2017-07-27 12:05:00	2017-07-27 13:10:00	SVX	
1	20760	PG0574	2017-08-20 08:45:00	2017-08-20 12:10:00	OVB	
2	24555	PG0191	2017-05-20 08:40:00	2017-05-20 11:50:00	CEK	
3	1212	PG0416	2017-06-09 16:20:00	2017-06-09 16:55:00	DME	
4	26123	PG0307	2017-05-22 13:00:00	2017-05-22 13:30:00	ROV	

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 11 columns):
 #   Column               Non-Null Count  Dtype         
---  ------               --------------  -----         
 0   flight_id            5000 non-null   int64         
 1   flight_no            5000 non-null   object        
 2   scheduled_departure  5000 non-null   datetime64[ns]
 3   scheduled_arrival    5000 non-null   datetime64[ns]
 4   departure_airport    5000 non-null   object        
 5   arrival_airport      5000 non-null   object        
 6   status               5000 non-null   object        
 7   aircraft_code        5000 non-null   object        
 8   actual_departure     3773 non-null   datetime64[ns]
 9   actual_arrival       3768 non-null   datetime64[ns]
 10  constraint_date      5000 non-null   datetime64[ns]
dtypes: datetime64[ns](5), int64(1), object(5)
memory usage: 429.8+ KB
```

In [8]:
```
id_and_regex = {'flight_no': 'PG[0-9]{5}',
                'aircraft_code': None,
                'departure_airport': '[A-Z]{3}',
                'arrival_airport': '[A-Z]{3}'}
```

In [9]:
```
pkey = 'flight_id'
```

Метаданные

В этом блоке:

1. Создается объект метаданных
2. Производится базовое автоматическое заполнение объекта на основе исходного DataFrame
3. Вызывается функция для основных дополнений объекта, а именно: а. Добавляется тип id и регулярки в соответствии с ранее заданными данными б. Добавляется тип datetime и формат для полей с датами при их наличии в. Добавляется первичный ключ
4. Кастомные апдейты метаданных по желанию

In [10]:
```
metadata = SingleTableMetadata()
```

In [11]:
```
metadata.detect_from_dataframe(data=df)
```

При отсутствии дат в таблице функция вызывается с 3 аргументами

In [12]:
```
sgf.update_metadata(metadata, pkey, id_and_regex, dates)
```

In [13]:
```
metadata
```

Out[13]:

```
{
    "primary_key": "flight_id",
    "columns": {
        "flight_id": {
            "sdtype": "id"
        },
        "flight_no": {
            "sdtype": "id",
            "regex_format": "PG[0-9]{5}"
        },
        "scheduled_departure": {
            "sdtype": "datetime",
            "datetime_format": "%Y-%m-%d %H:%M:%S"
        },
        "scheduled_arrival": {
            "sdtype": "datetime",
            "datetime_format": "%Y-%m-%d %H:%M:%S"
        },
        "departure_airport": {
            "sdtype": "id",
            "regex_format": "[A-Z]{3}"
        },
        "arrival_airport": {
            "sdtype": "id",
            "regex_format": "[A-Z]{3}"
        },
        "status": {
            "sdtype": "categorical"
        },
        "aircraft_code": {
            "sdtype": "id"
        },
        "actual_departure": {
            "sdtype": "datetime",
            "datetime_format": "%Y-%m-%d %H:%M:%S"
        },
        "actual_arrival": {
            "sdtype": "datetime",
            "datetime_format": "%Y-%m-%d %H:%M:%S"
        },
        "constraint_date": {
            "sdtype": "datetime",
            "datetime_format": "%Y-%m-%d %H:%M:%S"
        }
    },
    "METADATA_SPEC_VERSION": "SINGLE_TABLE_V1"
}
```


Образец для обновления метаданных хардкодом. Подробнее https://docs.sdv.dev/sdv/single-table-data/data-preparation/single-table-metadata-api

In [14]:
```
metadata.update_column(
    column_name='aircraft_code',
    sdtype='id',
```

```
    regex_format='[0-9A-Z]{4}'
)
```

Генерация

Далее находится код для генерации данных с использованием всех доступных моделей.

В результате будет получен сгенерированный DataFrame, сам объект генератора (модели), а также объект отчета о качестве. Все это дополняется понятной визуализацией из основной функции. Аргументы задаются в соответствии с **документацией**, которую можно увидеть в **следующей ячейке**. Дополнительно можно построить график распределения числовых и категориальных полей, а также полей с датами

In [15]:
```
help(sgf.generate_fake_data)
```

```
Help on function generate_fake_data in module single_gen_func:

generate_fake_data(model: 'str', metadata: 'sdv.metadata.multi_table.MultiTab
leMetadata', df: 'pandas.core.frame.DataFrame', fake_df_size: 'int')
    Назначение:

    Генерация фейковых данных. В теле функции модель создается
    и обучается. Далее производится выборка синтетических данных,
    создается отчет и визуализация, оценивающие качество
    сгенерированных данных

    Аргументы:

    model: Модель, использующаяся для генерации синтетических данных
    Доступно 4 варианта значения аргумента (не зависит от регистра):
        Copula - быстрая статистическая модель
        TVAE - относительно быстрая variational autoencoder-based нейронка
        CTGAN - генеративная(generative adversarial) нейронка, средняя скорос
ть
        CopGAN - комбинация Copula и CTGAN, средняя скорость

    metadata: sdv single_table_metadata

    df: pandas dataframe с исходными данными

    fake_df_size: количество строк в синтетическом наборе данных

    Результат:

    Сгенерированный набор синтетических данных, а также отчет о качестве
    и соответствующая визуализация

    Возвращаемое значение:

    generator: обученная модель

    df_result: синтетический pandas dataframe

    report: отчет о качестве
```

GaussianCopula model

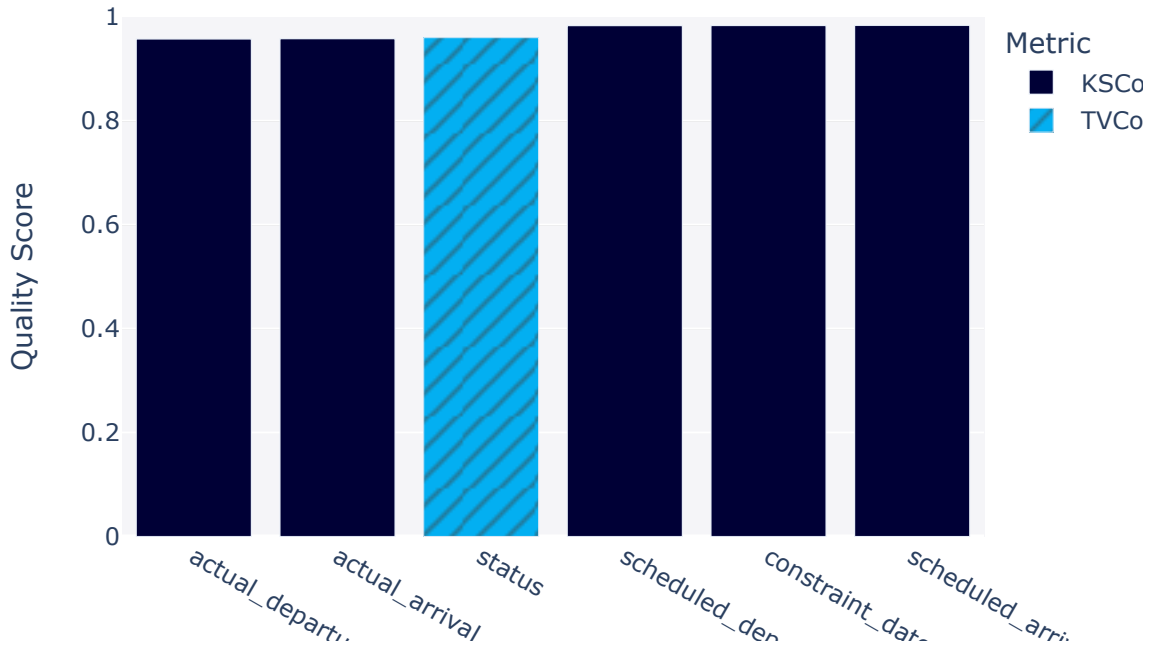
https://docs.sdv.dev/sdv/single-table-data/modeling/synthesizers/gaussiancopulasynthesizer

In [16]:
```
copula_generator, copula_sample, copula_report = sgf.generate_fake_data('cop
```

```
Время обучения  0.918 секунд
/Users/lalogin/sdv/lib/python3.7/site-packages/scipy/stats/_continuous_distn
s.py:624: RuntimeWarning: overflow encountered in _beta_ppf
  return _boost._beta_ppf(q, a, b)
Creating report: 100%|████████████████████████| 4/4 [00:00<00:00, 32.12it/s]
```

```
Overall Quality Score: 89.43%

Properties:
Column Shapes: 97.06%
Column Pair Trends: 81.79%
```



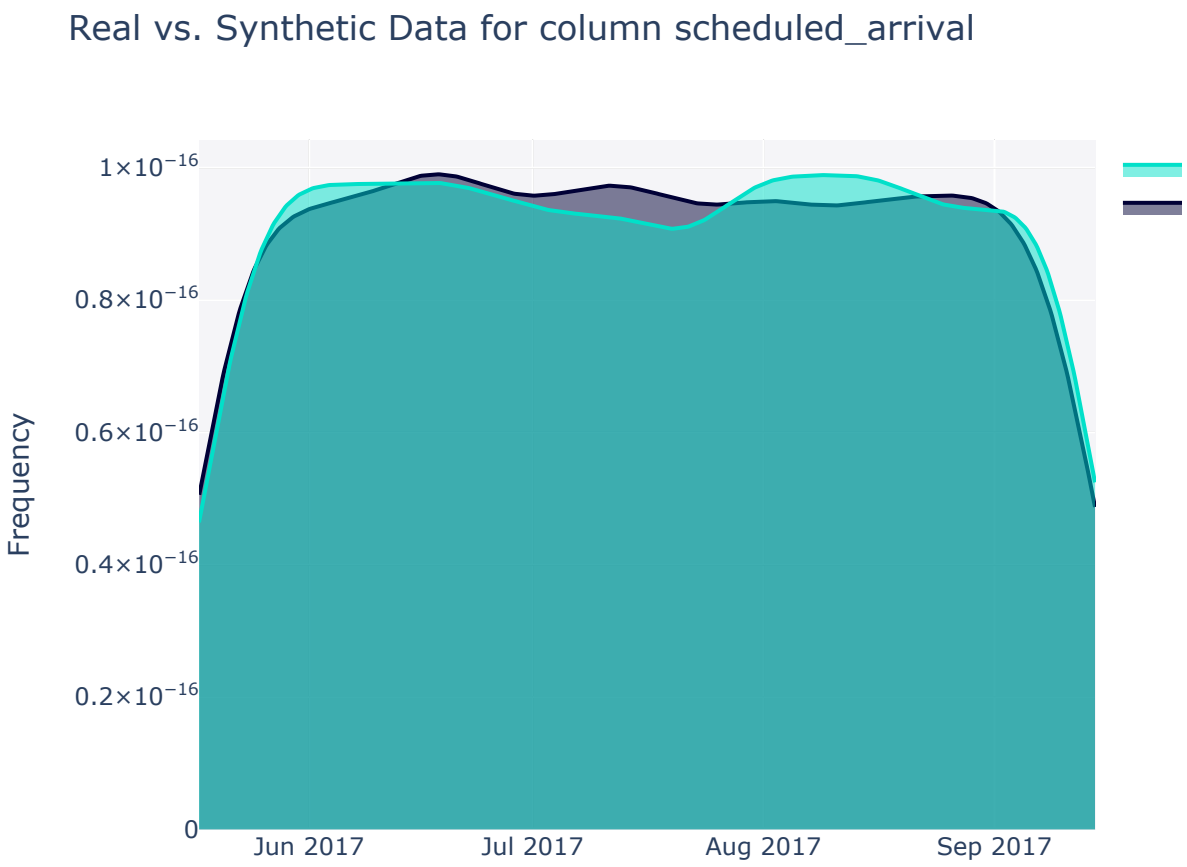
In [17]:
```
copula_sample.head(5)
```

Out[17]:

	flight_id	flight_no	scheduled_departure	scheduled_arrival	departure_airport	arı
0	0	PG00000	2017-08-17 05:47:22	2017-08-15 21:05:27	AAA	
1	1	PG00001	2017-07-16 01:29:26	2017-07-15 20:02:37	AAB	
2	2	PG00002	2017-08-26 15:12:35	2017-08-25 21:53:53	AAC	
3	3	PG00003	2017-08-07 23:47:52	2017-08-10 19:55:35	AAD	
4	4	PG00004	2017-09-13 21:45:33	2017-09-13 21:54:34	AAE	

Дополнительная визуализация распределения значений выбранного поля

In [18]:
```
fig = get_column_plot(
    real_data=df,
    synthetic_data=copula_sample,
    column_name='scheduled_arrival',
    metadata=metadata
)
fig.show()
```



CTGAN model

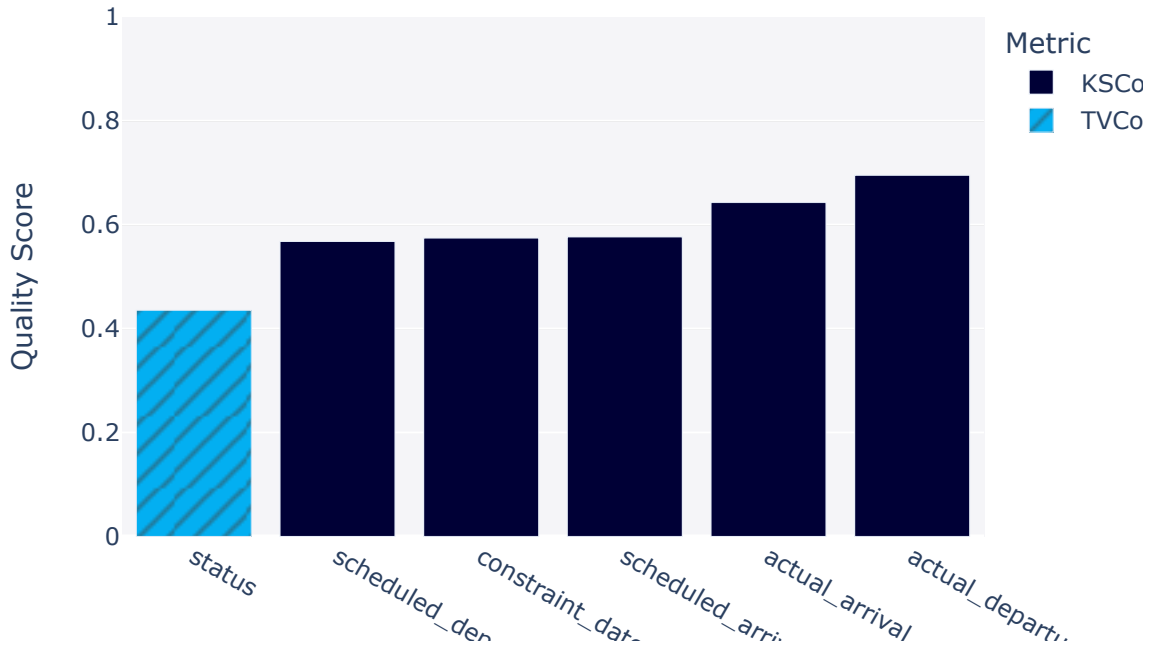
https://docs.sdv.dev/sdv/single-table-data/modeling/synthesizers/ctgansynthesizer

In [19]:
```
ctgan_generator, ctgan_sample, ctgan_report = sgf.generate_fake_data('ctgan'
```

```
Время обучения  198.798 секунд
Creating report: 100%|██████████| 4/4 [00:00<00:00, 22.63it/s]
```

```
Overall Quality Score: 57.63%

Properties:
Column Shapes: 58.18%
Column Pair Trends: 57.08%
```



In [20]:
```
ctgan_sample.head(5)
```

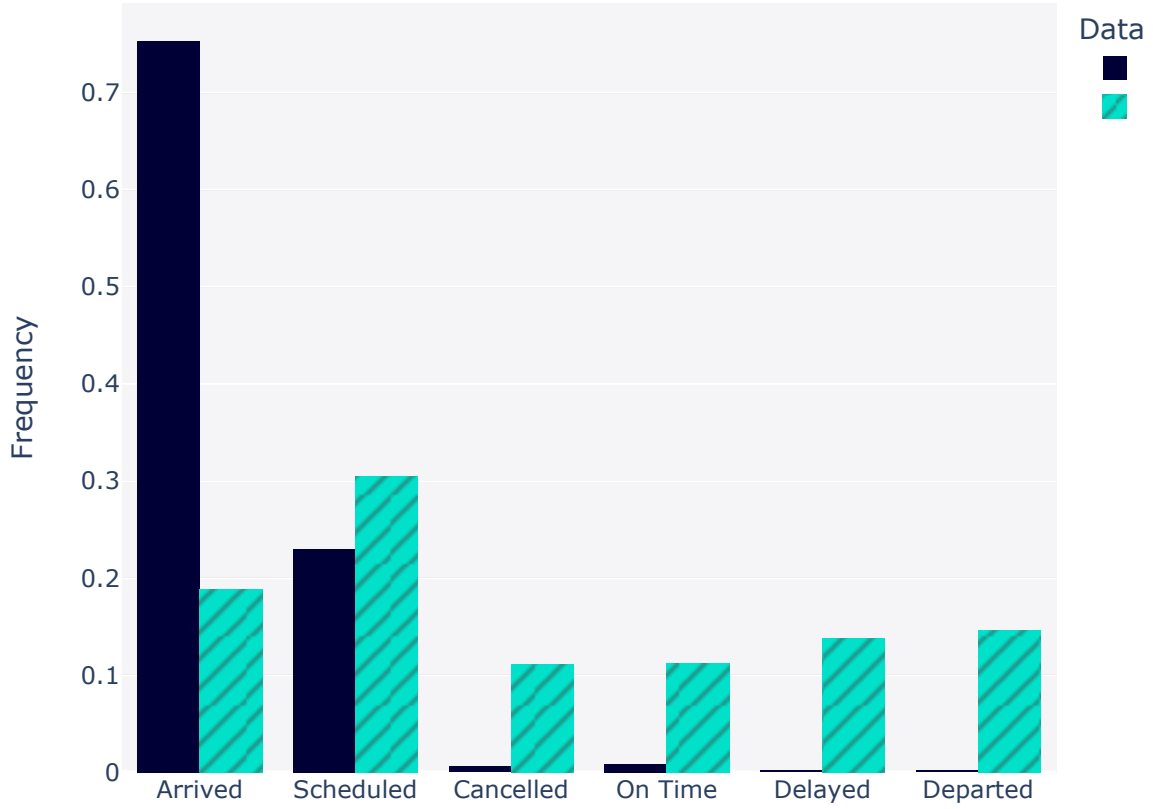
Out[20]:

	flight_id	flight_no	scheduled_departure	scheduled_arrival	departure_airport	arı
0	0	PG00000	2017-06-18 15:17:08	2017-08-14 22:25:15	AAA	
1	1	PG00001	2017-08-11 22:49:08	2017-08-12 10:19:57	AAB	
2	2	PG00002	2017-08-12 19:31:26	2017-08-08 21:12:43	AAC	
3	3	PG00003	2017-08-31 10:53:54	2017-08-16 16:05:26	AAD	
4	4	PG00004	2017-09-17 08:03:15	2017-09-11 08:52:05	AAE	

In [21]:
```
fig = get_column_plot(
```

```
    real_data=df,
    synthetic_data=ctgan_sample,
    column_name='status',
    metadata=metadata
)

fig.show()
```



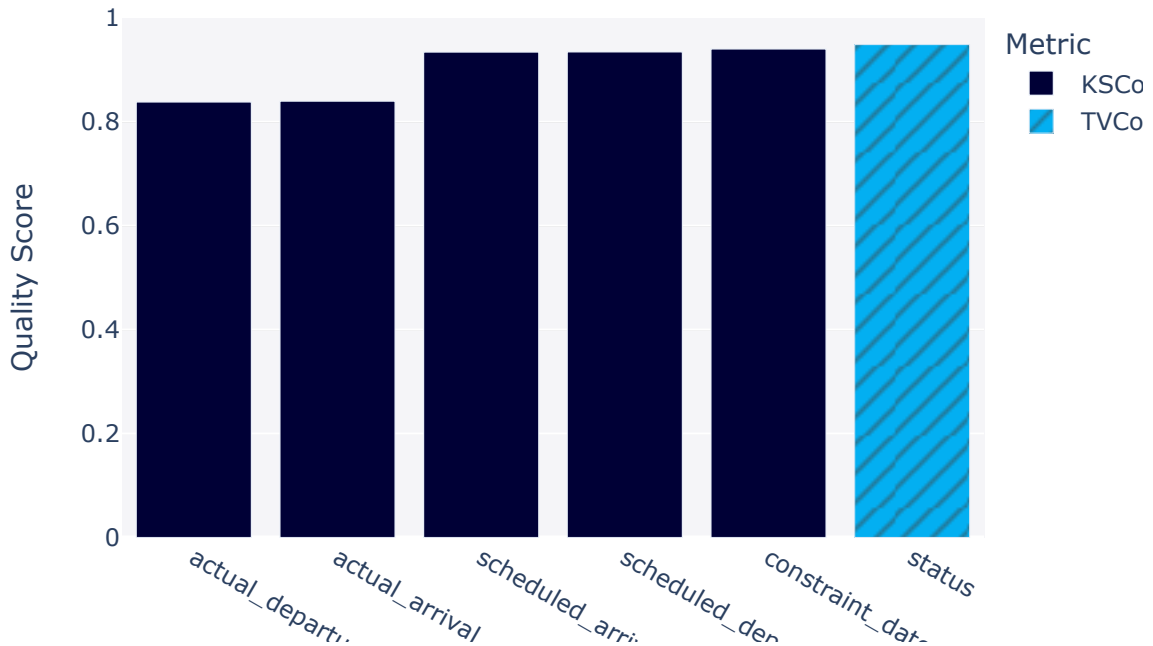
TVAE model

https://docs.sdv.dev/sdv/single-table-data/modeling/synthesizers/tvaesynthesizer

In [22]:
```
tvae_generator, tvae_sample, tvae_report = sgf.generate_fake_data('tvae', me
```

```
Время обучения  48.049 секунд
Creating report: 100%|████████████████████████| 4/4 [00:00<00:00, 27.58it/s]
Overall Quality Score: 84.45%

Properties:
Column Shapes: 90.56%
Column Pair Trends: 78.35%
```



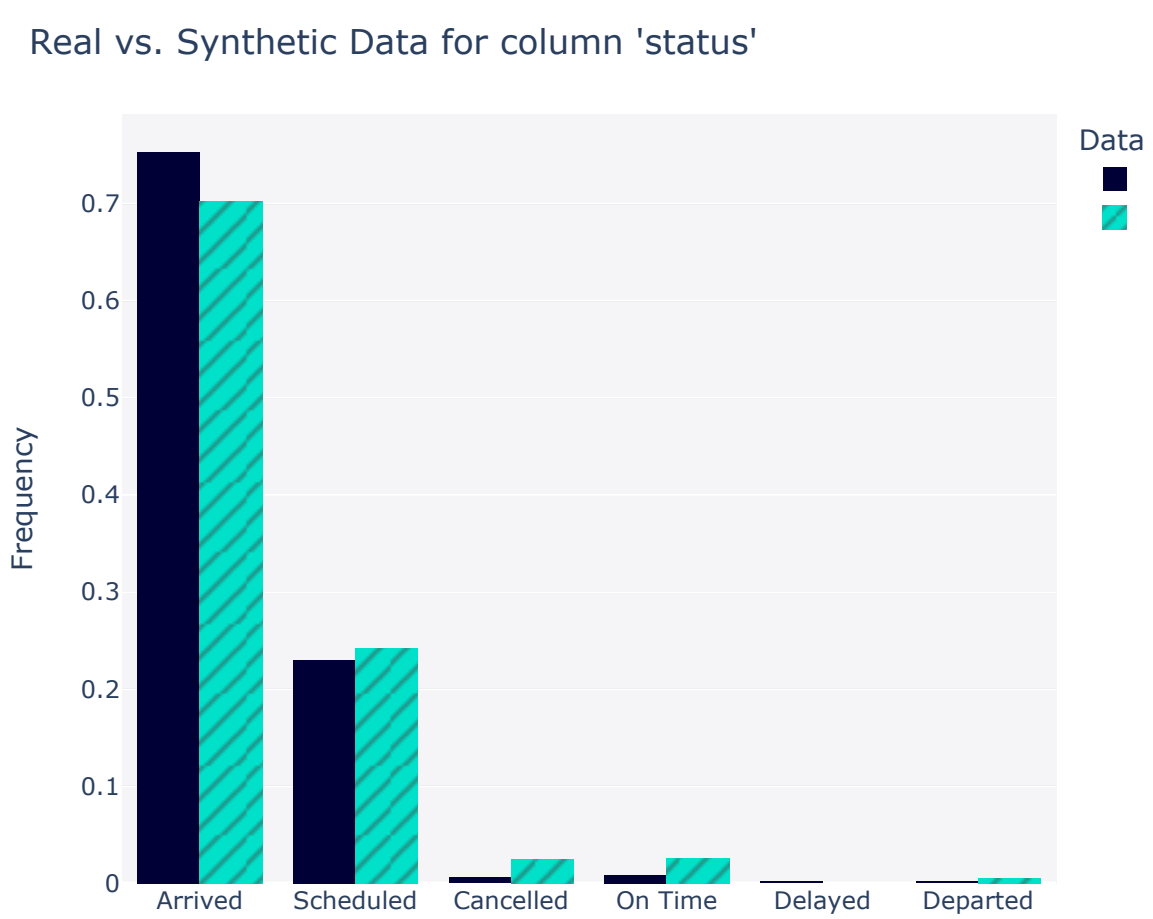
In [23]:
```
tvae_sample.head(5)
```

Out[23]:

	flight_id	flight_no	scheduled_departure	scheduled_arrival	departure_airport	ari
0	0	PG00000	2017-06-05 08:25:01	2017-07-03 17:39:34	AAA	
1	1	PG00001	2017-07-06 13:24:40	2017-06-28 02:14:25	AAB	
2	2	PG00002	2017-05-23 08:40:27	2017-06-04 05:35:13	AAC	
3	3	PG00003	2017-08-29 03:14:14	2017-06-01 01:16:24	AAD	
4	4	PG00004	2017-09-07 09:20:30	2017-09-08 02:48:45	AAE	

In [24]:
```
fig = get_column_plot(
    real_data=df,
    synthetic_data=tvae_sample,
    column_name='status',
    metadata=metadata
)
```

```
fig.show()
```



CopulaGAN model

https://docs.sdv.dev/sdv/single-table-data/modeling/synthesizers/copulagansynthesizer

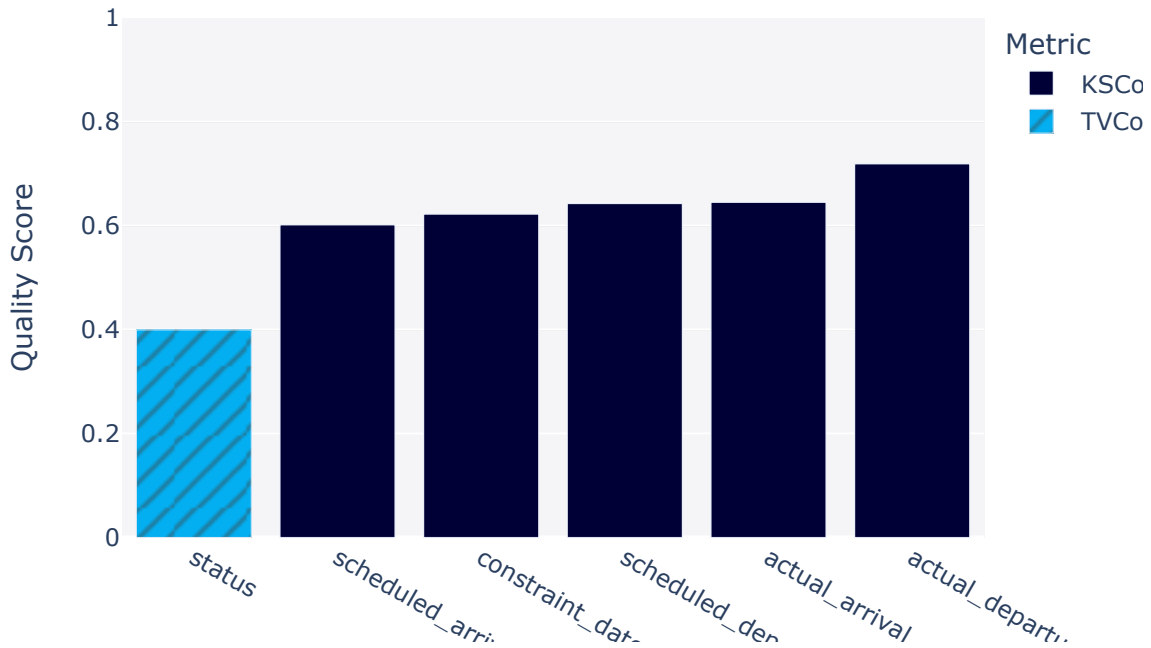
In [25]:
```
copgan_generator, copgan_sample, copgan_report = sgf.generate_fake_data('cop
```

```
Время обучения  197.147 секунд
/Users/lalogin/sdv/lib/python3.7/site-packages/scipy/stats/_continuous_distn
s.py:624: RuntimeWarning:

overflow encountered in _beta_ppf

Creating report: 100%|████████████████████████| 4/4 [00:00<00:00, 23.24it/s]
Overall Quality Score: 58.8%

Properties:
Column Shapes: 60.48%
Column Pair Trends: 57.12%
```



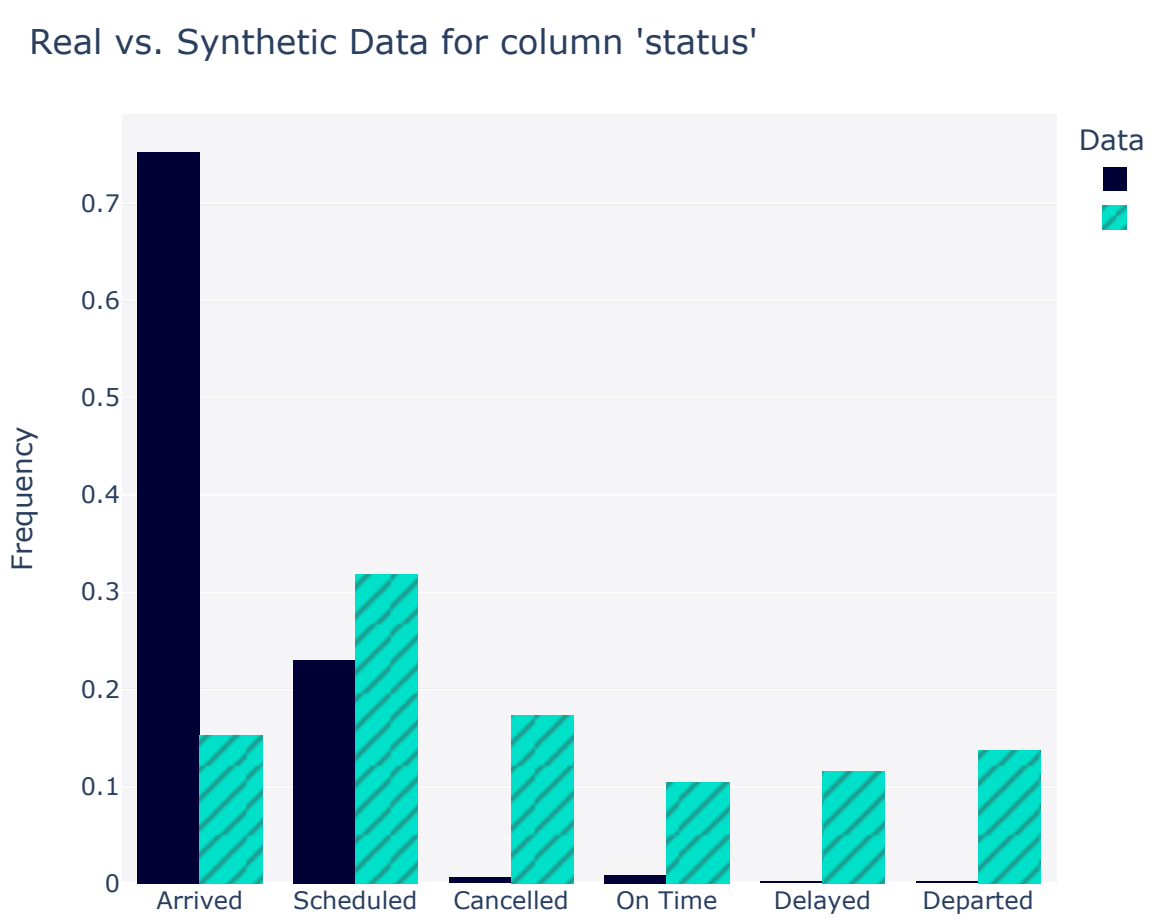
In [26]:
```
copgan_sample.head(5)
```

Out[26]:

	flight_id	flight_no	scheduled_departure	scheduled_arrival	departure_airport	ar
0	0	PG00000	2017-08-30 14:34:30	2017-09-06 01:10:12	AAA	
1	1	PG00001	2017-09-06 10:55:15	2017-08-01 02:01:52	AAB	
2	2	PG00002	2017-09-02 16:46:42	2017-09-09 21:19:14	AAC	
3	3	PG00003	2017-06-01 05:52:01	2017-06-04 07:30:54	AAD	
4	4	PG00004	2017-08-24 03:14:06	2017-08-24 12:36:09	AAE	

In [27]:
```
fig = get_column_plot(
    real_data=df,
    synthetic_data=copgan_sample,
    column_name='status',
    metadata=metadata
)
```

```
fig.show()
```


Real vs. Synthetic Data for column 'status'
Data
Frequency
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Arrived
Scheduled
Cancelled
On Time
Delayed
Departed