# Exploritory Analysis II

Exploring Lightcurve and pixel data through the Lightkurve API

This exploritory analysis will look at:

- Downloading Target Pixel Images
- Apply Appereture Masks
- Converting to Lightcurves
- Removing instrument noise
- Folding Lightcurves
- Identifying and visualising Exoplant Lightcurves

In [1]:
```
#imports
import pandas as pd
import numpy as np
import lightkurve as lk
from lightkurve import search_targetpixelfile
import matplotlib.pyplot as plt
```

In [2]:
```
# import the candidates file
file = ('../data/candidates/kepler_candidates.csv')
kep_df = pd.read_csv(file, low_memory = False)
kep_df.head()
```

Out[2]:

| | kepid | kepoi_name | kepler_name | koi_disposition | koi_pdisposition | koi_score | koi_fpflag_nt |
|---|---|---|---|---|---|---|---|
| **0** | 10797460 | K00752.01 | Kepler-227 b | CONFIRMED | CANDIDATE | 1.000 | 0 |
| **1** | 10797460 | K00752.02 | Kepler-227 c | CONFIRMED | CANDIDATE | 0.969 | 0 |
| **2** | 10811496 | K00753.01 | NaN | CANDIDATE | CANDIDATE | 0.000 | 0 |
| **3** | 10848459 | K00754.01 | NaN | FALSE POSITIVE | FALSE POSITIVE | 0.000 | 0 |
| **4** | 10854555 | K00755.01 | Kepler-664 b | CONFIRMED | CANDIDATE | 1.000 | 0 |

5 rows × 50 columns

This file contains all the kepler confirmed exoplanets and possible candidates
to be used for classification

## Using the Kepler ID to download Target Pixel Files

By using the kepid column to search the API the target pixel file can be downloaded
other feature information that will be useful in this file are:

**kepid** : int
arget identification number, as listed in the Kepler Input Catalog (KIC).
The KIC was derived from a ground-based imaging survey of the Kepler field
conducted prior to launch.

**koi_score** : float
A value between 0 and 1 that indicates the confidence in the KOI disposition.
For CANDIDATEs, a higher value indicates more confidence in its disposition,
while for FALSE POSITIVEs, a higher value indicates less confidence in that disposition.

**koi_disposition** : Char
The category of this KOI from the Exoplanet Archive. Current values are CANDIDATE,
FALSE POSITIVE, NOT DISPOSITIONED or CONFIRMED. All KOIs marked as CONFIRMED are also
listed in the Exoplanet Archive Confirmed Planet table

**koi_period** : double
The interval between consecutive planetary transits (days)

Using a combination of the above, to gather and label the data for the machine learning models. First using the **kepid** and the **koi_disposition** to investigate
the target pixel files of confirmed exoplanets.

## Investigate a known exoplanet

kepid_6922244 kepler name Kepler-8b is a known exoplanet in the dataset
with a KOI score of 0.998 and a transit period of 3.522498 days

### Downloading and Visulizing the Target Pixel File

In [3]:
```
# check the dataset for the above entry
exo_df = kep_df.copy()
exo_df = exo_df[exo_df['kepid']==6922244]
exo_df = exo_df.reindex(columns=['kepid', 'koi_disposition', 'koi_score', '
exo_df
```

Out[3]:

| | kepid | koi_disposition | koi_score | koi_period |
|---|---|---|---|---|
| **12** | 6922244 | CONFIRMED | 0.998 | 3.522498 |

In [4]:
```
search_result = lk.search_targetpixelfile('KIC 6922244', author='Kepler')
search_result
```

Out[4]: SearchResult containing 49 data products.

| # | mission | year | author | exptime | target_name | distance |
|---|---|---|---|---|---|---|
| | | | | s | | arcsec |
| 0 | Kepler Quarter 00 | 2009 | Kepler | 1800 | kplr006922244 | 0.0 |
| 1 | Kepler Quarter 01 | 2009 | Kepler | 1800 | kplr006922244 | 0.0 |
| 2 | Kepler Quarter 02 | 2009 | Kepler | 60 | kplr006922244 | 0.0 |

| # | mission | year | author | exptime | target_name | distance |
|---|---|---|---|---|---|---|
| 3 | Kepler Quarter 02 | 2009 | Kepler | 60 | kplr006922244 | 0.0 |
| 4 | Kepler Quarter 02 | 2009 | Kepler | 60 | kplr006922244 | 0.0 |
| 5 | Kepler Quarter 02 | 2009 | Kepler | 1800 | kplr006922244 | 0.0 |
| 6 | Kepler Quarter 03 | 2009 | Kepler | 60 | kplr006922244 | 0.0 |
| 7 | Kepler Quarter 03 | 2009 | Kepler | 60 | kplr006922244 | 0.0 |
| 8 | Kepler Quarter 03 | 2009 | Kepler | 60 | kplr006922244 | 0.0 |
| 9 | Kepler Quarter 03 | 2009 | Kepler | 1800 | kplr006922244 | 0.0 |
| ... | ... | ... | ... | ... | ... | ... |
| 39 | Kepler Quarter 12 | 2012 | Kepler | 60 | kplr006922244 | 0.0 |
| 40 | Kepler Quarter 12 | 2012 | Kepler | 1800 | kplr006922244 | 0.0 |
| 41 | Kepler Quarter 13 | 2012 | Kepler | 60 | kplr006922244 | 0.0 |
| 42 | Kepler Quarter 13 | 2012 | Kepler | 60 | kplr006922244 | 0.0 |
| 43 | Kepler Quarter 13 | 2012 | Kepler | 60 | kplr006922244 | 0.0 |
| 44 | Kepler Quarter 13 | 2012 | Kepler | 1800 | kplr006922244 | 0.0 |
| 45 | Kepler Quarter 14 | 2012 | Kepler | 1800 | kplr006922244 | 0.0 |
| 46 | Kepler Quarter 15 | 2013 | Kepler | 1800 | kplr006922244 | 0.0 |
| 47 | Kepler Quarter 16 | 2013 | Kepler | 1800 | kplr006922244 | 0.0 |
| 48 | Kepler Quarter 17 | 2013 | Kepler | 1800 | kplr006922244 | 0.0 |

There are 49 results for when this star was individually imaged

In [5]:
```
# get the most recent quarter of when the starr was imaged from the mission
most_recent = len(search_result.mission)
details = search_result.mission[most_recent-1]
details
```

Out[5]:
```
'Kepler Quarter 17'
```

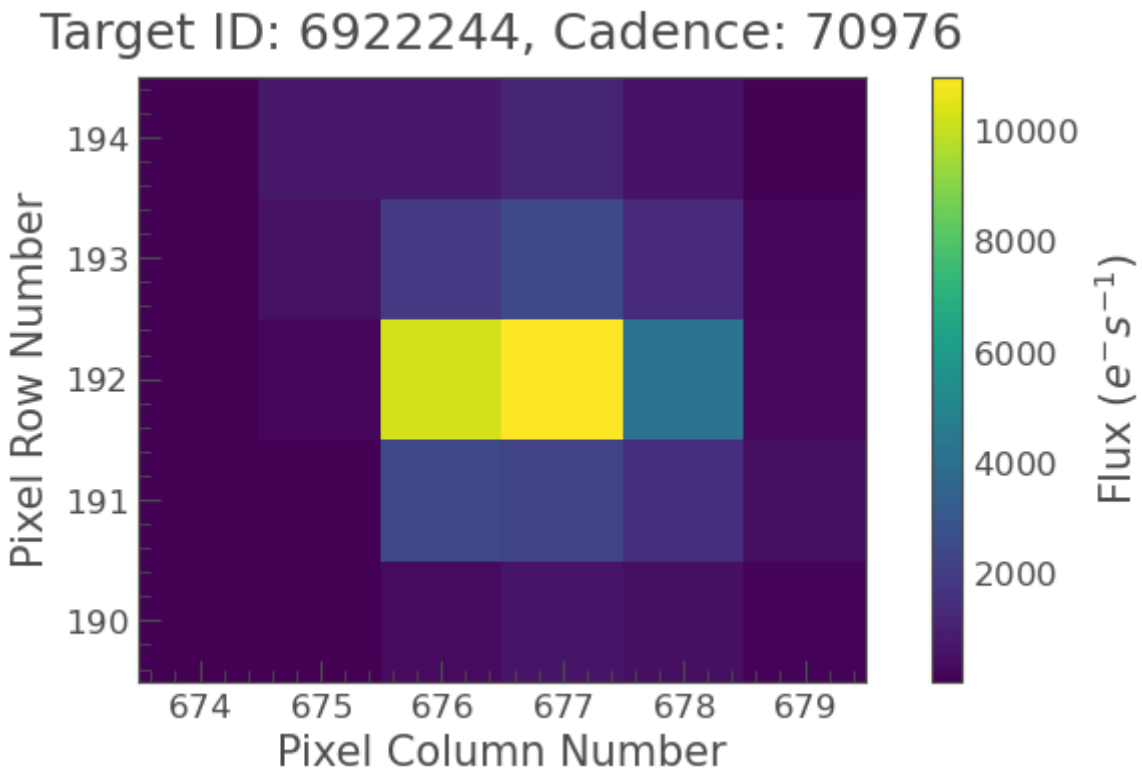
In [6]:
```
# get the quarter to use for search properties
qtr = details[-2:]
qtr = int(qtr)
qtr
```

Out[6]:
```
17
```

In [7]:
```
# search for the collection of target pixel files for this quarter
tpf = search_targetpixelfile('KIC 6922244', author="Kepler",quarter=qtr, ca
tpf
```

Out[7]:
```
KeplerTargetPixelFile Object (ID: 6922244)
```

In [8]:
```
# plot the first of the images
tpf.plot(frame=0)
plt.savefig('./graphs/target_pixel_6922244.jpg', bbox_inches="tight", dpi=3
plt.show()
```



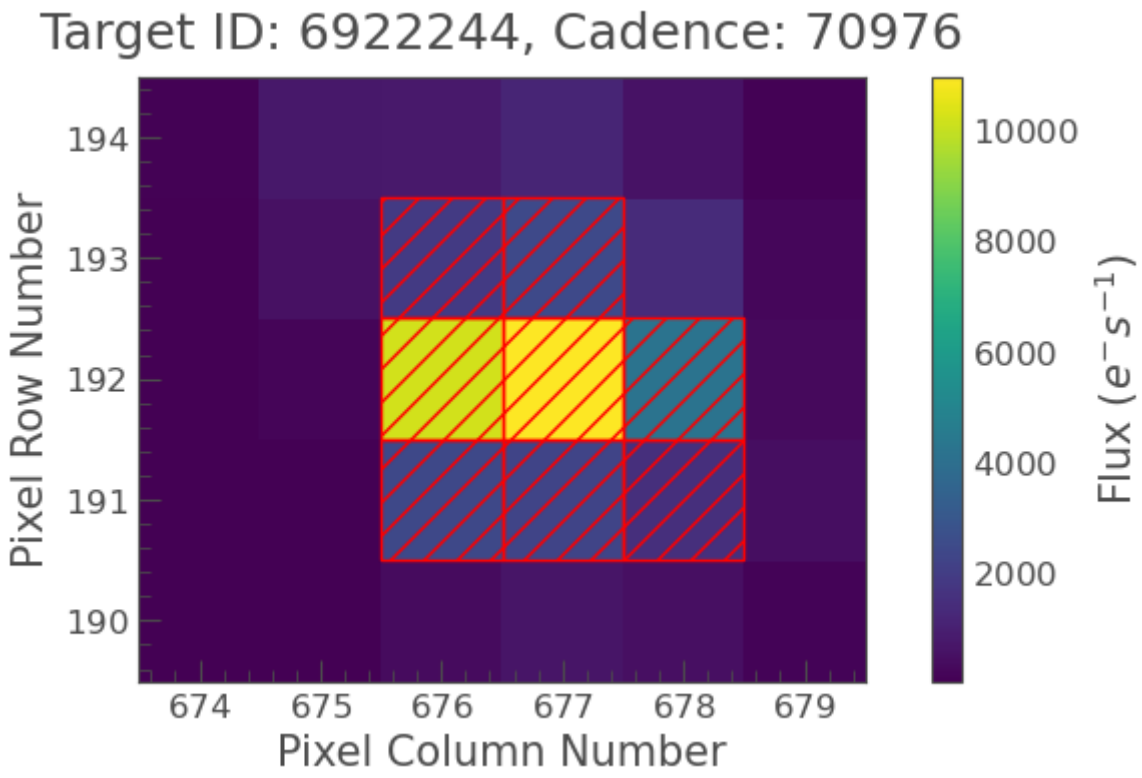

Pixel images contain the flux data needed to identify the exoplanet as well as create the dataset needed for the machine learning models

In [9]:
```
# print the flux values for this image
tpf.flux[0]
```

Out[9]: $[[7.255898,\ 4.8024983,\ 308.1806,\ 615.95422,\ 437.19757,\ 126.24071],\ [29.922783,\ 48.18$
$[16.620655,\ 146.23305,\ 10224.45,\ 12988.257,\ 4188.5552,\ 296.48846],\ [23.350971,\ 486.$
$[65.593941,\ 717.82678,\ 756.27301,\ 1123.9242,\ 521.66217,\ 70.44799]]\ \frac{e^-}{s}$

An apereture mask can be applied to the image to visualise the pixel sources of the Flux readings

In [10]:
```
# plot an apereture mask overlayed on the image
tpf.plot(aperture_mask=tpf.pipeline_mask);
plt.savefig('./graphs/target_pixel_ap_mask_6922244.jpg', bbox_inches="tight
plt.show()
```




With the LightKurve package it is possible to interact with the pixel files

In [11]:
```
# run interataaction
tpf.interact()
```

## Converting the Target Pixel File to useable Lightcurve

In [12]:
```
# convert the collection of targte pixel files to a lightcurve
lc =  tpf.to_lightcurve(aperture_mask='all')
lc
```

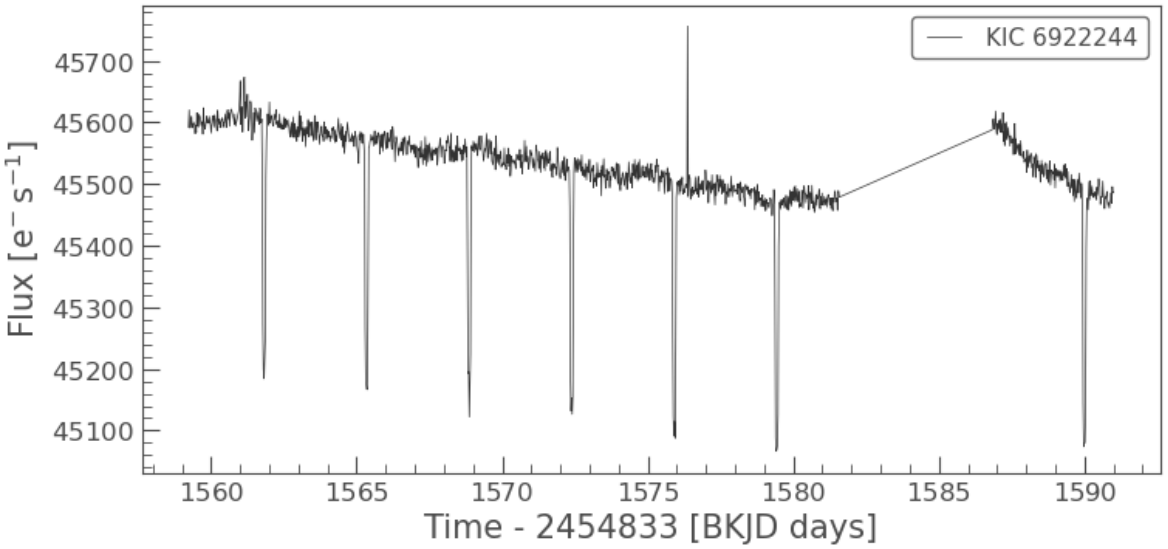
Out[12]: *KeplerLightCurve length=1286 LABEL="KIC 6922244" QUARTER=17 CAMPAIGN=None*

| time | flux | flux_err | centroid_col | centroid_row |
|---|---|---|---|---|
| | electron / s | electron / s | pix | pix |
| object | float32 | float32 | float64 | float64 |
| 1559.2266409377262 | 45592.14453125 | 10.228622436523438 | 676.8112972450118 | 192.07152290905165 |
| 1559.2470752079025 | 45620.26171875 | 10.229371070861816 | 676.8119858259928 | 192.07197517008095 |
| 1559.267509578196 | 45591.25390625 | 10.22878646850586 | 676.810995831429 | 192.07181150973543 |
| 1559.2879440486286 | 45591.44921875 | 10.22887897491455 | 676.8105453842174 | 192.07154026954547 |
| 1559.3083783191832 | 45591.8125 | 10.22874927520752 | 676.8112823546404 | 192.0720303438852 |
| 1559.328812689855 | 45609.59765625 | 10.229267120361328 | 676.8105852046192 | 192.07231446325775 |
| 1559.3492471606369 | 45596.43359375 | 10.228636741638184 | 676.8114198720602 | 192.07225536266174 |
| 1559.3696814315408 | 45591.1796875 | 10.22861385345459 | 676.8110931580934 | 192.07222875247868 |
| 1559.390115902621 | 45598.77734375 | 10.22860050201416 | 676.8109345072082 | 192.0730532169704 |

| time | flux | flux_err | centroid_col | centroid_row |
|---|---|---|---|---|
| ... | ... | ... | ... | ... |
| 1590.81826498 | 45464.90234375 | 10.200891494750977 | 676.7967535477575 | 192.10641386773565 |
| 1590.8386995243636 | 45470.796875 | 10.20124626159668 | 676.7965635448386 | 192.1065058851837 |
| 1590.8591338688348 | 45460.77734375 | 10.200848579406738 | 676.7969871858706 | 192.10592627354504 |
| 1590.8795683131975 | 45471.359375 | 10.202775955200195 | 676.7970662125382 | 192.10609285870547 |
| 1590.900002857561 | 45473.8984375 | 10.200918197631836 | 676.7969214609819 | 192.10586317009077 |
| 1590.9204372017994 | 45485.6953125 | 10.201351165771484 | 676.7972761320502 | 192.10626165848558 |
| 1590.9408717461629 | 45468.94921875 | 10.200958251953125 | 676.796997873776 | 192.10637932781802 |
| 1590.9613061904092 | 45495.74609375 | 10.201493263244629 | 676.7964642664083 | 192.1059193466083 |
| 1590.9817405346475 | 45485.9609375 | 10.201516151428223 | 676.7977503580191 | 192.10672300229498 |

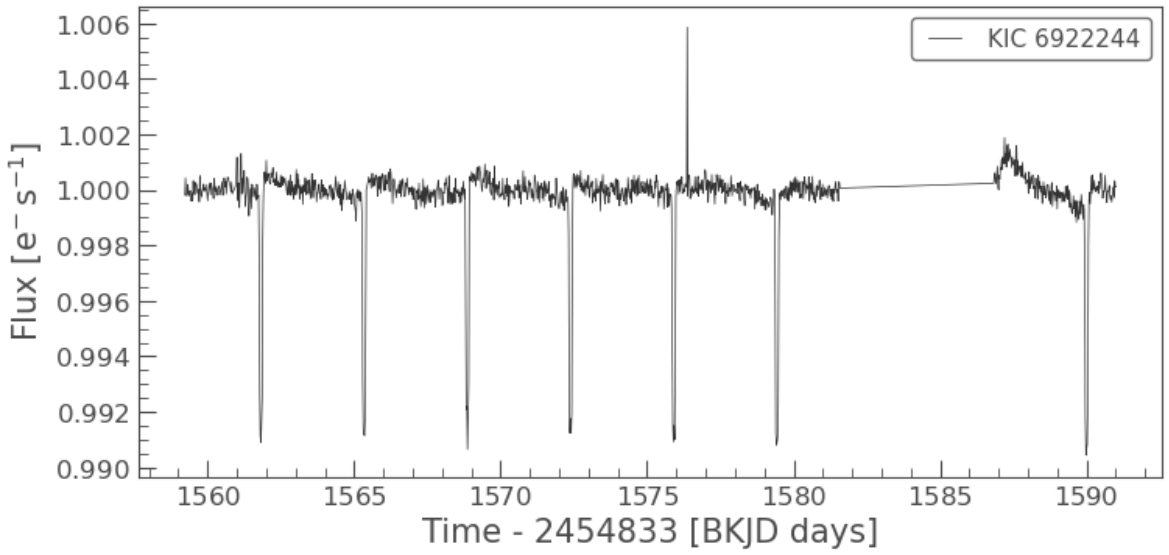
In [13]:

```
# visualisng the light curve
lc.plot()
plt.savefig('./graphs/lc_unclean_6922244.jpg', bbox_inches="tight", dpi=300
plt.show()
```




In [14]:

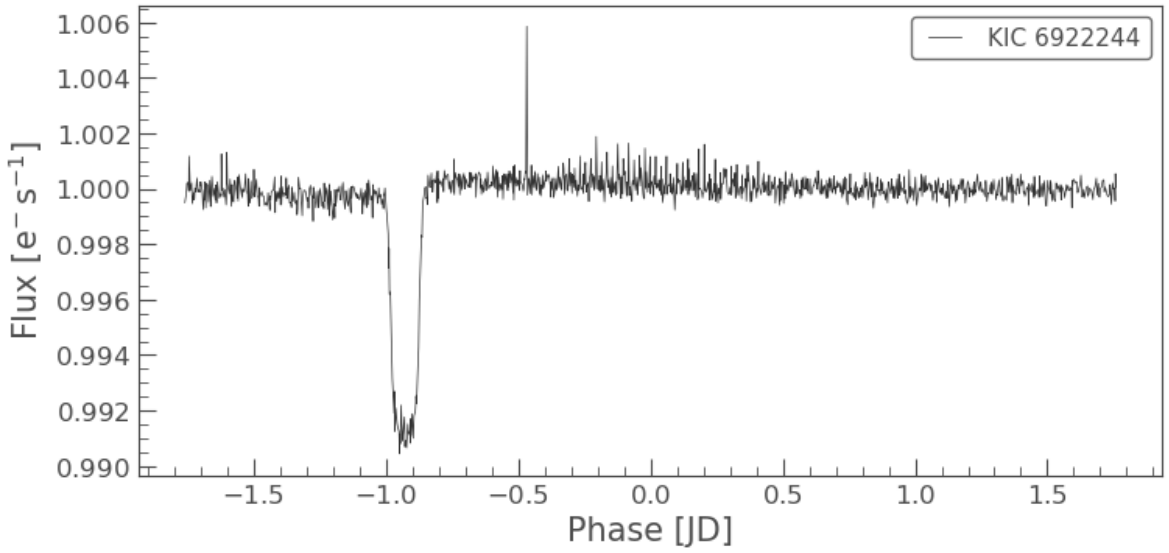
```
# removing the nans and flattening the lightcurve and replot
lc = lc.remove_nans().flatten(window_length=401)
lc.plot()
plt.savefig('./graphs/lc_nan_removed_6922244.jpg', bbox_inches="tight", dpi
plt.show()
```




There are visual dips in the lightcurve every 3 days approx.
The period time from the candidates file can be used to select the time period to fold the lightcurve to better represent this dip.

In [15]:

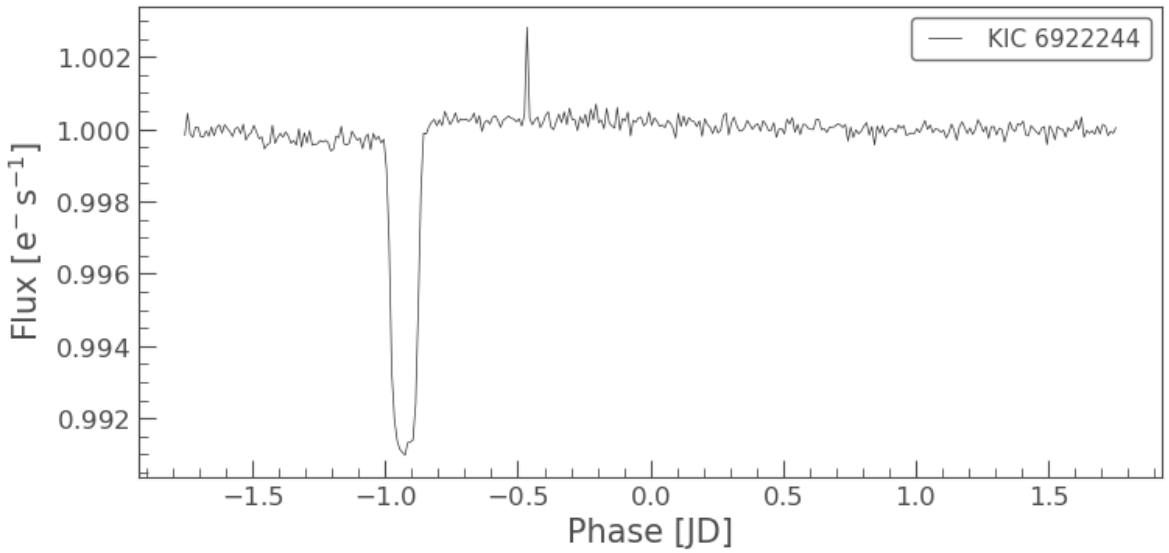
```
# using the koi period to set the folding anchor
fold_period = float(exo_df['koi_period'])

folded_lc = lc.fold(period=fold_period)
folded_lc.plot()
plt.savefig('./graphs/lc_folded_6922244.jpg', bbox_inches="tight", dpi=300)
plt.show()
```




The folded and flattened light curve can be normalised further to show a smoother line

In [16]:

```
# normalise the folded lightcurve
normalised_lc = folded_lc.bin(time_bin_size=0.01)
normalised_lc.plot();
plt.savefig('./graphs/lc_normalised_6922244.jpg', bbox_inches="tight", dpi=
plt.show()
```




## Converting the Normalised lightcurve to pandas dataframe for export

the normalised light curve can be converted to a pandas dataframe
and then can be transposed where the flux is a column of its own
to be used in the classification dataset

In [17]:
```
# convert to df
df = normalised_lc.to_pandas()
df
```

Out[17]:

| time | flux | flux_err | time_bin_start | time_bin_size | centroid_col | centroid_row | cad |
|---|---|---|---|---|---|---|---|
| **-1.755269** | 0.999830 | 0.000130 | -2 days +05:45:12.738832635 | 864.0 | 676.802260 | 192.088872 | |
| **-1.745269** | 1.000434 | 0.000130 | -2 days +05:59:36.738832635 | 864.0 | 676.800804 | 192.093229 | |
| **-1.735269** | 0.999868 | 0.000130 | -2 days +06:14:00.738832635 | 864.0 | 676.802927 | 192.088854 | |
| **-1.725269** | 0.999774 | 0.000112 | -2 days +06:28:24.738832635 | 864.0 | 676.801443 | 192.093874 | |
| **-1.715269** | 1.000051 | 0.000130 | -2 days +06:42:48.738832635 | 864.0 | 676.802476 | 192.088987 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **1.714731** | 1.000086 | 0.000129 | 1 days 17:02:00.738832634 | 864.0 | 676.798175 | 192.099280 | |
| **1.724731** | 0.999817 | 0.000130 | 1 days 17:16:24.738832634 | 864.0 | 676.802645 | 192.088611 | |
| **1.734731** | 0.999939 | 0.000112 | 1 days 17:30:48.738832634 | 864.0 | 676.800795 | 192.093972 | |
| **1.744731** | 0.999883 | 0.000129 | 1 days 17:45:12.738832634 | 864.0 | 676.802249 | 192.088768 | |

| time | flux | flux_err | time_bin_start | time_bin_size | centroid_col | centroid_row | cad |
|---|---|---|---|---|---|---|---|

In [18]:

```
# insert an incrimented column and convert to string
df.insert(0, 'Flux_', range(1, 1+len(df)))
df['Flux_'] = df['Flux_'].astype(str)
# add increment to combined column
df['Flux'] = 'Flux ' + df['Flux_']
df
```

Out[18]:

| time | Flux_ | flux | flux_err | time_bin_start | time_bin_size | centroid_col | centroid_ro |
|---|---|---|---|---|---|---|---|
| **-1.755269** | 1 | 0.999830 | 0.000130 | -2 days +05:45:12.738832635 | 864.0 | 676.802260 | 192.08887 |
| **-1.745269** | 2 | 1.000434 | 0.000130 | -2 days +05:59:36.738832635 | 864.0 | 676.800804 | 192.09322 |
| **-1.735269** | 3 | 0.999868 | 0.000130 | -2 days +06:14:00.738832635 | 864.0 | 676.802927 | 192.08885 |
| **-1.725269** | 4 | 0.999774 | 0.000112 | -2 days +06:28:24.738832635 | 864.0 | 676.801443 | 192.09387 |
| **-1.715269** | 5 | 1.000051 | 0.000130 | -2 days +06:42:48.738832635 | 864.0 | 676.802476 | 192.08898 |
| **...** | ... | ... | ... | ... | ... | ... | |
| **1.714731** | 348 | 1.000086 | 0.000129 | 1 days 17:02:00.738832634 | 864.0 | 676.798175 | 192.09928 |
| **1.724731** | 349 | 0.999817 | 0.000130 | 1 days 17:16:24.738832634 | 864.0 | 676.802645 | 192.08861 |
| **1.734731** | 350 | 0.999939 | 0.000112 | 1 days 17:30:48.738832634 | 864.0 | 676.800795 | 192.09397 |
| **1.744731** | 351 | 0.999883 | 0.000129 | 1 days 17:45:12.738832634 | 864.0 | 676.802249 | 192.08876 |
| **1.754731** | 352 | 1.000038 | 0.000112 | 1 days 17:59:36.738832634 | 864.0 | 676.800537 | 192.09436 |

352 rows × 10 columns

In [19]:

```
# strip out only the flux reading
df = df.reindex(columns=['Flux', 'flux'])
df
```

Out[19]:

| time | Flux | flux |
|---|---|---|
| **-1.755269** | Flux 1 | 0.999830 |
| **-1.745269** | Flux 2 | 1.000434 |
| **-1.735269** | Flux 3 | 0.999868 |

| time | Flux | flux |
|---|---|---|
| **-1.725269** | Flux 4 | 0.999774 |
| **-1.715269** | Flux 5 | 1.000051 |
| **...** | ... | ... |
| **1.714731** | Flux 348 | 1.000086 |
| **1.724731** | Flux 349 | 0.999817 |
| **1.734731** | Flux 350 | 0.999939 |
| **1.744731** | Flux 351 | 0.999883 |

In [20]:

```
# transpose the column to a row and reset the index
df = df.set_index('Flux').T
df.reset_index(inplace=True)
df
```

Out[20]:

| Flux | index | Flux 1 | Flux 2 | Flux 3 | Flux 4 | Flux 5 | Flux 6 | Flux 7 | Flux 8 | Flux |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | flux | 0.99983 | 1.000434 | 0.999868 | 0.999774 | 1.000051 | 1.000045 | 0.999787 | 0.999778 | 0.9999 |

1 rows × 353 columns

In [21]:

```
# rename the column for the Kepler ID
df.rename(columns={'index':'ID'}, inplace=True)
df['ID'] = lc.meta['LABEL']
df
```

Out[21]:

| Flux | ID | Flux 1 | Flux 2 | Flux 3 | Flux 4 | Flux 5 | Flux 6 | Flux 7 | Flux 8 | Fl |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | KIC 6922244 | 0.99983 | 1.000434 | 0.999868 | 0.999774 | 1.000051 | 1.000045 | 0.999787 | 0.999778 | 0.99 |

1 rows × 353 columns

In [22]:

```
df.index.rename('Index', inplace=True)
df
```

Out[22]:

| Flux | ID | Flux 1 | Flux 2 | Flux 3 | Flux 4 | Flux 5 | Flux 6 | Flux 7 | Flux 8 | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Index** | | | | | | | | | | |
| **0** | KIC 6922244 | 0.99983 | 1.000434 | 0.999868 | 0.999774 | 1.000051 | 1.000045 | 0.999787 | 0.999778 | 0.9 |

1 rows × 353 columns

In [23]:

```
# add the disposition for labeling
disposition = exo_df['koi_disposition'].values[0]
df.insert(1, 'disposition',disposition  )
df
```

Out[23]:

| Flux Index | ID | disposition | Flux 1 | Flux 2 | Flux 3 | Flux 4 | Flux 5 | Flux 6 | Flux 7 |
|---|---|---|---|---|---|---|---|---|---|
| **0** | KIC 6922244 | CONFIRMED | 0.99983 | 1.000434 | 0.999868 | 0.999774 | 1.000051 | 1.000045 | 0.999787 |

1 rows × 354 columns

## Dataset creation

This process can be done programatically (without) the visualisations to create the dataset for applying the classification models

In [ ]: