```
---------------------------------------------------------------------------
SSLCertVerificationError                  Traceback (most recent call last)
File ~\anaconda3\lib\urllib\request.py:1348, in AbstractHTTPHandler.do_open(self, http_class, req, **http_conn_a
rgs)
   1347 try:
-> 1348     h.request(req.get_method(), req.selector, req.data, headers,
   1349               encode_chunked=req.has_header('Transfer-encoding'))
   1350 except OSError as err: # timeout error

File ~\anaconda3\lib\http\client.py:1282, in HTTPConnection.request(self, method, url, body, headers, encode_chu
nked)
   1281 """Send a complete request to the server."""
-> 1282 self._send_request(method, url, body, headers, encode_chunked)

File ~\anaconda3\lib\http\client.py:1328, in HTTPConnection._send_request(self, method, url, body, headers, enco
de_chunked)
   1327     body = _encode(body, 'body')
-> 1328 self.endheaders(body, encode_chunked=encode_chunked)

File ~\anaconda3\lib\http\client.py:1277, in HTTPConnection.endheaders(self, message_body, encode_chunked)
   1276     raise CannotSendHeader()
-> 1277 self._send_output(message_body, encode_chunked=encode_chunked)

File ~\anaconda3\lib\http\client.py:1037, in HTTPConnection._send_output(self, message_body, encode_chunked)
   1036 del self._buffer[:]
-> 1037 self.send(msg)
   1039 if message_body is not None:
   1040 
   1041     # create a consistent interface to message_body

File ~\anaconda3\lib\http\client.py:975, in HTTPConnection.send(self, data)
    974 if self.auto_open:
--> 975     self.connect()
    976 else:

File ~\anaconda3\lib\http\client.py:1454, in HTTPSConnection.connect(self)
   1452     server_hostname = self.host
-> 1454 self.sock = self._context.wrap_socket(self.sock,
   1455                                       server_hostname=server_hostname)

File ~\anaconda3\lib\ssl.py:513, in SSLContext.wrap_socket(self, sock, server_side, do_handshake_on_connect, sup
press_ragged_eofs, server_hostname, session)
    507 def wrap_socket(self, sock, server_side=False,
    508                 do_handshake_on_connect=True,
    509                 suppress_ragged_eofs=True,
    510                 server_hostname=None, session=None):
    511     # SSLSocket class handles server_hostname encoding before it calls
    512     # ctx._wrap_socket()
--> 513     return self.sslsocket_class._create(
    514         sock=sock,
    515         server_side=server_side,
    516         do_handshake_on_connect=do_handshake_on_connect,
    517         suppress_ragged_eofs=suppress_ragged_eofs,
    518         server_hostname=server_hostname,
    519         context=self,
    520         session=session
    521     )

File ~\anaconda3\lib\ssl.py:1071, in SSLSocket._create(cls, sock, server_side, do_handshake_on_connect, suppress
_ragged_eofs, server_hostname, context, session)
   1070             raise ValueError("do_handshake_on_connect should not be specified for non-blocking sockets")
-> 1071         self.do_handshake()
   1072 except (OSError, ValueError):

File ~\anaconda3\lib\ssl.py:1342, in SSLSocket.do_handshake(self, block)
   1341         self.settimeout(None)
-> 1342     self._sslobj.do_handshake()
   1343 finally:

SSLCertVerificationError: [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed: unable to get local issuer
certificate (_ssl.c:997)

During handling of the above exception, another exception occurred:

URLError                                  Traceback (most recent call last)
Cell In[27], line 1
----> 1 pd.read_csv('https://s3.sa-east-1.amazonaws.com/ckan.saude.gov.br/SIPNI/COVID/uf/uf%3DAC/part-00000-b1ec
5973-367f-4204-9eb0-4fc92f4edfe0.c000.csv')

File ~\anaconda3\lib\site-packages\pandas\util\_decorators.py:211, in deprecate_kwarg.<locals>._deprecate_kwarg.
<locals>.wrapper(*args, **kwargs)
    209     else:
    210         kwargs[new_arg_name] = new_arg_value
--> 211 return func(*args, **kwargs)
```

```
File ~\anaconda3\lib\site-packages\pandas\util\_decorators.py:331, in deprecate_nonkeyword_arguments.<locals>.de
corate.<locals>.wrapper(*args, **kwargs)
    325 if len(args) > num_allow_args:
    326     warnings.warn(
    327         msg.format(arguments=_format_argument_list(allow_args)),
    328         FutureWarning,
    329         stacklevel=find_stack_level(),
    330     )
--> 331 return func(*args, **kwargs)

File ~\anaconda3\lib\site-packages\pandas\io\parsers\readers.py:950, in read_csv(filepath_or_buffer, sep, delimi
ter, header, names, index_col, usecols, squeeze, prefix, mangle_dupe_cols, dtype, engine, converters, true_value
s, false_values, skipinitialspace, skiprows, skipfooter, nrows, na_values, keep_default_na, na_filter, verbose,
skip_blank_lines, parse_dates, infer_datetime_format, keep_date_col, date_parser, dayfirst, cache_dates, iterato
r, chunksize, compression, thousands, decimal, lineterminator, quotechar, quoting, doublequote, escapechar, comm
ent, encoding, encoding_errors, dialect, error_bad_lines, warn_bad_lines, on_bad_lines, delim_whitespace, low_me
mory, memory_map, float_precision, storage_options)
    935 kwds_defaults = _refine_defaults_read(
    936     dialect,
    937     delimiter,
   (...)
    946     defaults={"delimiter": ","},
    947 )
    948 kwds.update(kwds_defaults)
--> 950 return _read(filepath_or_buffer, kwds)

File ~\anaconda3\lib\site-packages\pandas\io\parsers\readers.py:605, in _read(filepath_or_buffer, kwds)
    602 _validate_names(kwds.get("names", None))
    604 # Create the parser.
--> 605 parser = TextFileReader(filepath_or_buffer, **kwds)
    607 if chunksize or iterator:
    608     return parser

File ~\anaconda3\lib\site-packages\pandas\io\parsers\readers.py:1442, in TextFileReader.__init__(self, f, engin
e, **kwds)
   1439     self.options["has_index_names"] = kwds["has_index_names"]
   1441 self.handles: IOHandles | None = None
-> 1442 self._engine = self._make_engine(f, self.engine)

File ~\anaconda3\lib\site-packages\pandas\io\parsers\readers.py:1735, in TextFileReader._make_engine(self, f, en
gine)
   1733     if "b" not in mode:
   1734         mode += "b"
-> 1735 self.handles = get_handle(
   1736     f,
   1737     mode,
   1738     encoding=self.options.get("encoding", None),
   1739     compression=self.options.get("compression", None),
   1740     memory_map=self.options.get("memory_map", False),
   1741     is_text=is_text,
   1742     errors=self.options.get("encoding_errors", "strict"),
   1743     storage_options=self.options.get("storage_options", None),
   1744 )
   1745 assert self.handles is not None
   1746 f = self.handles.handle

File ~\anaconda3\lib\site-packages\pandas\io\common.py:713, in get_handle(path_or_buf, mode, encoding, compressi
on, memory_map, is_text, errors, storage_options)
    710     codecs.lookup_error(errors)
    712 # open URLs
--> 713 ioargs = _get_filepath_or_buffer(
    714     path_or_buf,
    715     encoding=encoding,
    716     compression=compression,
    717     mode=mode,
    718     storage_options=storage_options,
    719 )
    721 handle = ioargs.filepath_or_buffer
    722 handles: list[BaseBuffer]

File ~\anaconda3\lib\site-packages\pandas\io\common.py:363, in _get_filepath_or_buffer(filepath_or_buffer, encod
ing, compression, mode, storage_options)
    361 # assuming storage_options is to be interpreted as headers
    362 req_info = urllib.request.Request(filepath_or_buffer, headers=storage_options)
--> 363 with urlopen(req_info) as req:
    364     content_encoding = req.headers.get("Content-Encoding", None)
    365     if content_encoding == "gzip":
    366         # Override compression based on Content-Encoding header

File ~\anaconda3\lib\site-packages\pandas\io\common.py:265, in urlopen(*args, **kwargs)
    259 """
    260 Lazy-import wrapper for stdlib urlopen, as that imports a big chunk of
    261 the stdlib.
    262 """
    263 import urllib.request
--> 265 return urllib.request.urlopen(*args, **kwargs)

File ~\anaconda3\lib\urllib\request.py:216, in urlopen(url, data, timeout, cafile, capath, cadefault, context)
    214 else:
    215     opener = _opener
--> 216 return opener.open(url, data, timeout)

File ~\anaconda3\lib\urllib\request.py:519, in OpenerDirector.open(self, fullurl, data, timeout)
    516     req = meth(req)
    518 sys.audit('urllib.Request', req.full_url, req.data, req.headers, req.get_method())
--> 519 response = self._open(req, data)
    521 # post-process response
    522 meth_name = protocol+"_response"

File ~\anaconda3\lib\urllib\request.py:536, in OpenerDirector._open(self, req, data)
    533     return result
    535 protocol = req.type
--> 536 result = self._call_chain(self.handle_open, protocol, protocol +
    537                           '_open', req)
    538 if result:
    539     return result

File ~\anaconda3\lib\urllib\request.py:496, in OpenerDirector._call_chain(self, chain, kind, meth_name, *args)
    494 for handler in handlers:
    495     func = getattr(handler, meth_name)
--> 496     result = func(*args)
    497     if result is not None:
    498         return result

File ~\anaconda3\lib\urllib\request.py:1391, in HTTPSHandler.https_open(self, req)
   1390 def https_open(self, req):
-> 1391     return self.do_open(http.client.HTTPSConnection, req,
   1392         context=self._context, check_hostname=self._check_hostname)

File ~\anaconda3\lib\urllib\request.py:1351, in AbstractHTTPHandler.do_open(self, http_class, req, **http_conn_a
rgs)
   1348         h.request(req.get_method(), req.selector, req.data, headers,
   1349                   encode_chunked=req.has_header('Transfer-encoding'))
   1350     except OSError as err: # timeout error
-> 1351         raise URLError(err)
   1352     r = h.getresponse()
   1353 except:

URLError: <urlopen error [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed: unable to get local issuer
certificate (_ssl.c:997)>
```