In [2]:

```
!python --version
!pip list
# Pip
!pip install --disable-pip-version-check -q pip --upgrade > /dev/null
!pip install --disable-pip-version-check -q wrapt --upgrade > /dev/null
# AWS CLI and AWS Python SDK (boto3)
!pip install --disable-pip-version-check -q awscli==1.18.216 boto3==1.16.56 botocore==1.19.56
# SageMaker
!pip install --disable-pip-version-check -q sagemaker==2.29.0
!pip install --disable-pip-version-check -q smdebug==1.0.1
!pip install --disable-pip-version-check -q sagemaker-experiments==0.1.26
# PyTorch
!conda install -y pytorch==1.6.0 -c pytorch
# TensorFlow
!pip install --disable-pip-version-check -q tensorflow==2.3.1
# Hugging Face Transformers (BERT)
!pip install --disable-pip-version-check -q transformers==3.5.1
# TorchServe
!pip install --disable-pip-version-check -q torchserve==0.3.0
!pip install --disable-pip-version-check -q torch-model-archiver==0.3.0
# PyAthena
!pip install --disable-pip-version-check -q PyAthena==2.1.0
# Redshift
!pip install --disable-pip-version-check -q SQLAlchemy==1.3.22
# AWS Data Wrangler
!pip install --disable-pip-version-check -q awswrangler==2.3.0
# StepFunctions
!pip install --disable-pip-version-check -q stepfunctions==2.0.0rc1
# Zip
!conda install -y zip
# Matplotlib
!pip install --disable-pip-version-check -q matplotlib==3.1.3
# Seaborn
!pip install --disable-pip-version-check -q seaborn==0.10.0


# Summarize
!python --version
# !pip list
setup_dependencies_passed = True
%store setup_dependencies_passed
%store
```

Python 3.7.10

Package	Version
aiobotocore	2.4.2
aiohttp	3.8.4
aioitertools	0.11.0
aiosignal	1.3.1
alabaster	0.7.12
anaconda-client	1.7.2
anaconda-project	0.8.3
ansi2html	1.8.0
anyio	3.6.2
argh	0.26.2
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
asn1crypto	1.3.0
astroid	2.15.2
astropy	4.3.1
async-timeout	4.0.2
asynctest	0.13.0
atomicwrites	1.3.0
attrs	22.2.0
autopep8	1.4.4
autovizwidget	0.20.5
awscli	1.27.111
Babel	2.12.1
backcall	0.1.0
backports.shutil-get-terminal-size	1.0.0
beautifulsoup4	4.8.2
bitarray	1.2.1
bkcharts	0.2
bleach	6.0.0
bokeh	1.4.0
boto	2.49.0
boto3	1.26.111
botocore	1.29.111
Bottleneck	1.3.2
brotlipy	0.7.0
cached-property	1.5.2
certifi	2022.12.7
cffi	1.15.0
chardet	3.0.4
charset-normalizer	2.0.4
Click	7.0
cloudpickle	2.2.1

```
clyent                          1.2.2
colorama                        0.4.3
conda                           22.9.0
conda-package-handling          1.8.1
contextlib2                     0.6.0.post1
cryptography                    40.0.1
cycler                          0.10.0
Cython                          0.29.15
cytoolz                         0.10.1
dash                            2.9.2
dash-core-components            2.0.0
dash-html-components            2.0.0
dash-table                      5.0.0
dask                            2022.2.0
decorator                       4.4.1
defusedxml                      0.6.0
diff-match-patch                20181111
dill                            0.3.6
distributed                     2022.2.0
distro                          1.8.0
docker                          6.0.1
docker-compose                  1.29.2
dockerpty                       0.4.1
docopt                          0.6.2
docutils                        0.16
dparse                          0.6.2
entrypoints                     0.3
et-xmlfile                      1.0.1
fastcache                       1.1.0
fastjsonschema                  2.16.3
filelock                        3.0.12
flake8                          3.7.9
Flask                           1.1.1
frozenlist                      1.3.3
fsspec                          2023.1.0
future                          0.18.2
gevent                          1.4.0
glob2                           0.7
gmpy2                           2.0.8
google-pasta                    0.2.0
greenlet                        0.4.15
h5py                            2.10.0
hdijupyterutils                 0.20.5
HeapDict                        1.0.1
html5lib                        1.0.1
```

hypothesis	5.5.4
idna	2.8
imageio	2.6.1
imagesize	1.2.0
importlib-metadata	6.3.0
intervaltree	3.0.2
ipykernel	5.1.4
ipython	7.34.0
ipython_genutils	0.2.0
ipywidgets	7.5.1
isort	4.3.21
itsdangerous	1.1.0
jdcal	1.4.1
jedi	0.18.2
jeepney	0.4.2
Jinja2	3.1.2
jmespath	1.0.1
joblib	1.2.0
json5	0.9.1
jsonschema	3.2.0
jupyter	1.0.0
jupyter_client	7.4.9
jupyter-console	6.1.0
jupyter_core	4.12.0
jupyter-dash	0.4.2
jupyter-server	1.23.6
jupyterlab	1.2.21
jupyterlab-pygments	0.2.2
jupyterlab-server	1.0.6
keyring	21.1.0
kiwisolver	1.1.0
lazy-object-proxy	1.4.3
libarchive-c	2.8
lief	0.9.0
llvmlite	0.39.1
locket	0.2.0
lxml	4.9.2
MarkupSafe	2.1.2
matplotlib	3.1.3
matplotlib-inline	0.1.6
mccabe	0.6.1
mistune	0.8.4
mkl-fft	1.0.15
mkl-random	1.1.0
mkl-service	2.3.0

mock	4.0.1
more-itertools	8.2.0
mpmath	1.1.0
msgpack	0.6.1
multidict	6.0.4
multipledispatch	0.6.0
multiprocess	0.70.14
nbclassic	0.5.5
nbclient	0.7.3
nbconvert	6.5.4
nbformat	5.8.0
nest-asyncio	1.5.6
networkx	2.4
nltk	3.8.1
nose	1.3.7
notebook	6.5.4
notebook_shim	0.2.2
numba	0.56.4
numexpr	2.7.1
numpy	1.21.6
numpydoc	0.9.2
olefile	0.46
openpyxl	3.0.3
packaging	20.1
pandas	1.3.5
pandocfilters	1.4.2
parso	0.8.3
partd	1.1.0
path	13.1.0
pathlib2	2.3.5
pathos	0.3.0
pathtools	0.1.2
patsy	0.5.1
pep8	1.7.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.5.0
pip	23.0.1
pkginfo	1.5.0.1
platformdirs	3.2.0
plotly	5.8.2
pluggy	0.13.1
ply	3.11
pox	0.3.2
ppft	1.7.6.6

prometheus-client	0.7.1
prompt-toolkit	3.0.3
protobuf	3.20.3
protobuf3-to-dict	0.1.5
psutil	5.6.7
ptyprocess	0.6.0
pure-sasl	0.6.2
py	1.11.0
pyarrow	11.0.0
pyasn1	0.4.8
pycodestyle	2.5.0
pycosat	0.6.3
pycparser	2.19
pycryptodome	3.17
pycurl	7.43.0.5
pydocstyle	4.0.1
pyerfa	2.0.0.3
pyflakes	2.1.1
pyfunctional	1.4.3
Pygments	2.15.0
PyHive	0.6.5
pykerberos	1.2.1
pylint	2.17.2
pyodbc	4.0.0-unsupported
pyOpenSSL	23.1.1
pyparsing	2.4.6
pyrsistent	0.15.7
PySocks	1.7.1
pytest	5.3.5
pytest-arraydiff	0.3
pytest-astropy	0.8.0
pytest-astropy-header	0.1.2
pytest-doctestplus	0.5.0
pytest-openfiles	0.4.0
pytest-remotedata	0.3.2
python-dateutil	2.8.2
python-dotenv	0.21.1
python-jsonrpc-server	0.3.4
python-language-server	0.31.7
pytz	2019.3
PyWavelets	1.1.1
pyxdg	0.26
PyYAML	6.0
pyzmq	25.0.2
QDarkStyle	2.8

```
QtAwesome                           0.6.1
qtconsole                           4.6.0
QtPy                                1.9.0
regex                               2022.10.31
requests                            2.28.2
requests-kerberos                   0.12.0
retrying                            1.3.4
rope                                0.16.0
rsa                                 4.9
Rtree                               0.9.3
ruamel_yaml                         0.15.87
s3fs                                0.4.2
s3transfer                          0.6.0
sagemaker                           2.145.0
sagemaker-data-insights             0.3.3
sagemaker-datawrangler              0.3.9
sagemaker-scikit-learn-extension    2.5.0
sagemaker-studio-analytics-extension 0.0.18
sagemaker-studio-sparkmagic-lib     0.1.4
sasl                                0.3a1
schema                              0.7.5
scikit-image                        0.16.2
scikit-learn                        0.22.1
scipy                               1.4.1
seaborn                             0.10.0
SecretStorage                       3.1.2
Send2Trash                          1.8.0
setuptools                          59.3.0
simplegeneric                       0.8.1
singledispatch                      3.4.0.3
six                                 1.14.0
smclarify                           0.5
smdebug-rulesconfig                 1.0.1
sniffio                             1.3.0
snowballstemmer                     2.0.0
sortedcollections                   1.1.2
sortedcontainers                    2.1.0
soupsieve                           1.9.5
sparkmagic                          0.20.4
Sphinx                              2.4.0
sphinxcontrib-applehelp             1.0.1
sphinxcontrib-devhelp               1.0.1
sphinxcontrib-htmlhelp              1.0.2
sphinxcontrib-jsmath                1.0.1
sphinxcontrib-qthelp                1.0.2
```

sphinxcontrib-serializinghtml	1.1.3
sphinxcontrib-websupport	1.2.0
spyder	4.0.1
spyder-kernels	1.8.1
SQLAlchemy	1.3.13
statsmodels	0.11.0
sympy	1.5.1
tables	3.6.1
tabulate	0.9.0
tblib	1.6.0
tenacity	8.2.2
terminado	0.8.3
testpath	0.4.4
texttable	1.6.7
thrift	0.13.0
thrift-sasl	0.4.3
tinycss2	1.2.1
toml	0.10.2
tomli	2.0.1
tomlkit	0.11.7
toolz	0.10.0
tornado	6.2
tqdm	4.42.1
traitlets	5.9.0
typed-ast	1.5.4
typing_extensions	4.5.0
ujson	5.7.0
unicodecsv	0.14.1
urllib3	1.26.15
watchdog	0.10.2
wcwidth	0.1.8
webencodings	0.5.1
websocket-client	0.59.0
Werkzeug	2.2.3
wheel	0.40.0
widgetsnbextension	3.5.1
wrapt	1.11.2
wurlitzer	2.0.0
xlrd	1.2.0
XlsxWriter	1.2.7
xlwt	1.3.0
yapf	0.28.0
yarl	1.8.2
zict	1.0.0
zipp	3.15.0

```
[notice] A new release of pip is available: 23.0.1 -> 23.1
[notice] To update, run: pip install --upgrade pip
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the s
ystem package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/v
env
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. Th
is behaviour is the source of the following dependency conflicts.
spyder 4.0.1 requires pyqt5<5.13; python_version >= "3", which is not installed.
spyder 4.0.1 requires pyqtwebengine<5.13; python_version >= "3", which is not installed.
aiobotocore 2.4.2 requires botocore<1.27.60,>=1.27.59, but you have botocore 1.29.111 which is incompatible.
spyder 4.0.1 requires jedi==0.14.1, but you have jedi 0.18.2 which is incompatible.
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the s
ystem package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/v
env
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. Th
is behaviour is the source of the following dependency conflicts.
spyder 4.0.1 requires pyqt5<5.13; python_version >= "3", which is not installed.
spyder 4.0.1 requires pyqtwebengine<5.13; python_version >= "3", which is not installed.
aiobotocore 2.4.2 requires botocore<1.27.60,>=1.27.59, but you have botocore 1.19.56 which is incompatible.
sagemaker 2.145.0 requires boto3<2.0,>=1.26.28, but you have boto3 1.16.56 which is incompatible.
sagemaker 2.145.0 requires PyYAML==5.4.1, but you have pyyaml 5.3 which is incompatible.
sagemaker-studio-analytics-extension 0.0.18 requires boto3<2.0,>=1.26.49, but you have boto3 1.16.56 which is
incompatible.
spyder 4.0.1 requires jedi==0.14.1, but you have jedi 0.18.2 which is incompatible.
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the s
ystem package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/v
env
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the s
ystem package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/v
env
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the s
ystem package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/v
env
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the s
ystem package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/v
env
Collecting package metadata (current_repodata.json): done
Solving environment: failed with initial frozen solve. Retrying with flexible solve.
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /opt/conda
```

```
  added / updated specs:
    - pytorch==1.6.0


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    ca-certificates-2023.01.10 |       h06a4308_0         120 KB
    certifi-2022.12.7          |   py37h06a4308_0         150 KB
    cudatoolkit-10.2.89        |       hfd86e86_1       365.1 MB
    ninja-1.10.2               |       h06a4308_5           8 KB
    ninja-base-1.10.2          |       hd09550d_5         109 KB
    pytorch-1.6.0              |py3.7_cuda10.2.89_cudnn7.6.5_0       537.7 MB  pytorch
    ------------------------------------------------------------
                                           Total:       903.1 MB

The following NEW packages will be INSTALLED:

  cudatoolkit        pkgs/main/linux-64::cudatoolkit-10.2.89-hfd86e86_1 None
  ninja              pkgs/main/linux-64::ninja-1.10.2-h06a4308_5 None
  ninja-base         pkgs/main/linux-64::ninja-base-1.10.2-hd09550d_5 None
  pytorch            pytorch/linux-64::pytorch-1.6.0-py3.7_cuda10.2.89_cudnn7.6.5_0 None

The following packages will be UPDATED:

  ca-certificates    conda-forge::ca-certificates-2022.12.~ --> pkgs/main::ca-certificates-2023.01.10-h06a430
8_0 None

The following packages will be SUPERSEDED by a higher-priority channel:

  certifi            conda-forge/noarch::certifi-2022.12.7~ --> pkgs/main/linux-64::certifi-2022.12.7-py37h06
a4308_0 None



Downloading and Extracting Packages
certifi-2022.12.7    | 150 KB    | ##################################### | 100%
cudatoolkit-10.2.89  | 365.1 MB  | ##################################### | 100%
ca-certificates-2023 | 120 KB    | ##################################### | 100%
ninja-1.10.2         | 8 KB      | ##################################### | 100%
pytorch-1.6.0        | 537.7 MB  | ##################################### | 100%
ninja-base-1.10.2    | 109 KB    | ##################################### | 100%
Preparing transaction: done
```

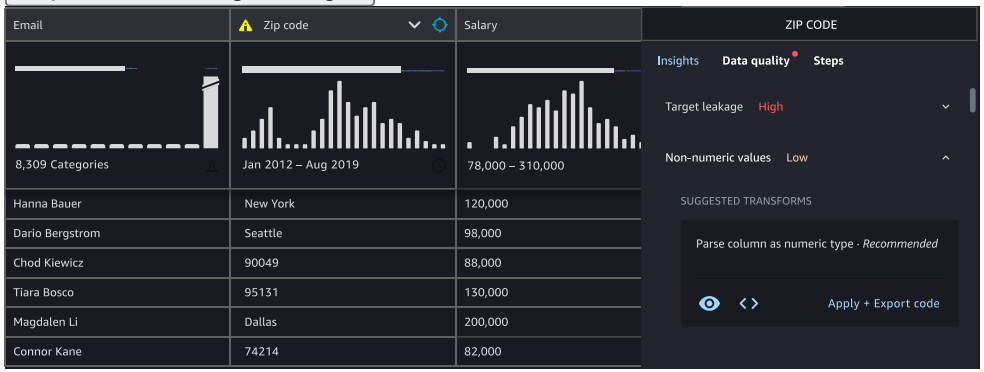
```
Verifying transaction: done
Executing transaction: done
Retrieving notices: ...working... done
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. Th
is behaviour is the source of the following dependency conflicts.
pytest-astropy 0.8.0 requires pytest-cov>=2.0, which is not installed.
pytest-astropy 0.8.0 requires pytest-filter-subpackage>=0.1, which is not installed.
spyder 4.0.1 requires pyqt5<5.13; python_version >= "3", which is not installed.
spyder 4.0.1 requires pyqtwebengine<5.13; python_version >= "3", which is not installed.
python-language-server 0.31.7 requires jedi<0.16,>=0.14.1, but you have jedi 0.18.2 which is incompatible.
python-language-server 0.31.7 requires ujson<=1.35; platform_system != "Windows", but you have ujson 5.7.0 wh
ich is incompatible.
sparkmagic 0.20.4 requires nest-asyncio==1.5.5, but you have nest-asyncio 1.5.6 which is incompatible.
spyder 4.0.1 requires jedi==0.14.1, but you have jedi 0.18.2 which is incompatible.
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the s
ystem package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/v
env
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the s
ystem package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/v
env
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the s
ystem package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/v
env
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the s
ystem package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/v
env
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the s
ystem package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/v
env
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the s
ystem package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/v
env
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. Th
is behaviour is the source of the following dependency conflicts.
spyder 4.0.1 requires pyqt5<5.13; python_version >= "3", which is not installed.
spyder 4.0.1 requires pyqtwebengine<5.13; python_version >= "3", which is not installed.
sagemaker-data-insights 0.3.3 requires numpy>=1.21.6, but you have numpy 1.18.1 which is incompatible.
sparkmagic 0.20.4 requires nest-asyncio==1.5.5, but you have nest-asyncio 1.5.6 which is incompatible.
spyder 4.0.1 requires jedi==0.14.1, but you have jedi 0.18.2 which is incompatible.
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the s
ystem package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/v
env
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the s
ystem package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/v
env
```

```
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /opt/conda

  added / updated specs:
    - zip


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    zip-3.0                    |       h7f8727e_1         111 KB
    ------------------------------------------------------------
                                           Total:         111 KB

The following NEW packages will be INSTALLED:

  zip                pkgs/main/linux-64::zip-3.0-h7f8727e_1 None



Downloading and Extracting Packages
zip-3.0              | 111 KB    | ##################################### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
Retrieving notices: ...working... done
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the s
ystem package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/v
env
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the s
ystem package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/v
env
Python 3.7.10
Stored 'setup_dependencies_passed' (bool)
Stored variables and their in-db values:
```

Clean data in minutes

Automatically visualize data, and improve data quality in a few clicks. Learn more

Import Data Wrangler widget



Remind me later

Don't show again

```
df4
-> Unnamed: 0  68.7000.119 62.6000.104 63.30
ingest_create_athena_db_passed
-> True
s3_public_goldendrive
-> 's3://goldevdrive_data'
setup_dependencies_passed
-> T
```

```
rue
setu
p_ia
m_ro
les_
pass
ed
-> T
rue
setu
p_in
stan
ce_c
heck
_pas
sed
-> T
rue
setu
p_s3
_buc
ket_
pass
ed
-> T
rue
```

In [4]:
```
import boto3, re, sys, math, json, os, sagemaker, urllib.request
import io
import sagemaker
from sagemaker import get_execution_role
from IPython.display import Image
from IPython.display import display
from time import gmtime, strftime
from sagemaker.predictor import csv_serializer
from pyathena import connect
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split, \
RepeatedStratifiedKFold, RandomizedSearchCV
from sklearn.metrics import roc_curve, auc, mean_squared_error,\
```

```
precision_score, recall_score, f1_score, accuracy_score,\
confusion_matrix, plot_confusion_matrix, classification_report
from sagemaker.tuner import HyperparameterTuner
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from scipy.stats import loguniform
import warnings
warnings.filterwarnings('ignore')
```

In [5]:
```
import boto3

region = boto3.Session().region_name
session = boto3.session.Session()

ec2 = boto3.Session().client(service_name="ec2", region_name=region)
sm = boto3.Session().client(service_name="sagemaker", region_name=region)
```

In [6]:
```
import json

notebook_instance_name = None

try:
    with open("/opt/ml/metadata/resource-metadata.json") as notebook_info:
        data = json.load(notebook_info)
        domain_id = data["DomainId"]
        resource_arn = data["ResourceArn"]
        region = resource_arn.split(":")[3]
        name = data["ResourceName"]
    print("DomainId: {}".format(domain_id))
    print("Name: {}".format(name))
except:
    print("+++++++++++++++++++++++++++++++++++++++++")
    print("[ERROR]: COULD NOT RETRIEVE THE METADATA.")
    print("+++++++++++++++++++++++++++++++++++++++++")
```

```
DomainId: d-x9g2xqzh9zoe
Name: datascience-1-0-ml-t3-medium-1abf3407f667f989be9d86559395
```

In [7]:
```
describe_domain_response = sm.describe_domain(DomainId=domain_id)
print(describe_domain_response["Status"])
```

```
InService
```

In [8]:
```
try:
    get_status_response = sm.get_sagemaker_servicecatalog_portfolio_status()
```

```
    print(get_status_response["Status"])
except:
    pass
```

```
Enabled
```

In [9]:
```
if (
    describe_domain_response["Status"] == "InService"
    and get_status_response["Status"] == "Enabled"
    and "datascience" in name
):
    setup_instance_check_passed = True
    print("[OK] Checks passed!  Great Job!!  Please Continue.")
else:
    setup_instance_check_passed = False
    print("++++++++++++++++++++++++++++++++++++++++++++++++")
    print("[ERROR]: WE HAVE IDENTIFIED A MISCONFIGURATION.")
    print(describe_domain_response["Status"])
    print(get_status_response["Status"])
    print(name)
    print("++++++++++++++++++++++++++++++++++++++++++++++++")
```

```
[OK] Checks passed!  Great Job!!  Please Continue.
```

In [10]:
```
import boto3
import sagemaker
import time
from time import gmtime, strftime

sagemaker_session = sagemaker.Session()
role = sagemaker.get_execution_role()
bucket = sagemaker_session.default_bucket()
region = boto3.Session().region_name

from botocore.config import Config

config = Config(retries={"max_attempts": 10, "mode": "adaptive"})

iam = boto3.client("iam", config=config)
```

In [11]:
```
role_name = role.split("/")[-1]

print("Role name: {}".format(role_name))
```

```
Role name: LabRole
```

In [12]:
```
admin = False
post_policies = iam.list_attached_role_policies(RoleName=role_name)["AttachedPolicies"]
for post_policy in post_policies:
    if post_policy["PolicyName"] == "AdministratorAccess":
        admin = True
        break

setup_iam_roles_passed = True
print("[OK] You are all set up to continue with this workshop!")
```

```
[OK] You are all set up to continue with this workshop!
```

In [13]:
```
if not setup_instance_check_passed:
    print("++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")
    print("[ERROR] YOU HAVE TO RUN ALL NOTEBOOKS IN THE SETUP FOLDER FIRST. You are missing Instance Check."
    print("++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")
if not setup_dependencies_passed:
    print("++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")
    print("[ERROR] YOU HAVE TO RUN ALL NOTEBOOKS IN THE SETUP FOLDER FIRST. You are missing Setup Dependenci
    print("++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")
if not setup_iam_roles_passed:
    print("++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")
    print("[ERROR] YOU HAVE TO RUN ALL NOTEBOOKS IN THE SETUP FOLDER FIRST. You are missing Setup IAM Roles.
    print("++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")
```

In [14]:
```
import boto3
import sagemaker

sess = sagemaker.Session()
bucket = sess.default_bucket()
role = sagemaker.get_execution_role()
region = boto3.Session().region_name
```

In [15]:
```
!pip install --disable-pip-version-check -q PyAthena==2.1.0
from pyathena import connect
```

```
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the
system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warning
s/venv
```

In [16]:
```
database_name = "CA_Traffic"
```

In [17]:
```
bucket = sess.default_bucket()
# Set S3 staging directory -- this is a temporary directory used for Athena queries
```

```
s3_staging_dir = "s3://{0}/athena/staging".format(bucket)
```

In [18]:
```
conn = connect(region_name=region, s3_staging_dir=s3_staging_dir)
```

In [19]:
```
statement = "CREATE DATABASE IF NOT EXISTS {}".format(database_name)
print(statement)
```

```
CREATE DATABASE IF NOT EXISTS CA_Traffic
```

In [20]:
```
import pandas as pd
```

In [21]:
```
pd.read_sql(statement, conn)
```

Out[21]: —

In [ ]:
```

```

In [22]:
```
statement = "SHOW DATABASES"

df_show = pd.read_sql(statement, conn)
df_show.head(5)
```

Out[22]:

	database_name
0	ca_traffic
1	default
2	goldendrive

In [23]:
```
if database_name in df_show.values:
    ingest_create_athena_db_passed = True
```

In [24]:
```
# Set Athena parameters
database_name = "ca_traffic"
Ex_Ln_table = "Ex_Ln"
HOV_table = "HOV"
vol_table = "vol"
Route77_table = 'Route77'
s3_path_Ex_Ln = "s3://{}/Express_Lanes.csv".format(bucket)
s3_path_HOV = "s3://{}/HOV.csv".format(bucket)
s3_path_vol = "s3://{}/Traffic_Volumes_AADT.csv".format(bucket)
s3_path_Route77 = "s3://{}/Route_77.csv".format(bucket)
```

In [25]:
```
statement1 = """CREATE EXTERNAL TABLE IF NOT EXISTS {}.{}(
         OBJECTID int,
         District int,
         Route string,
         Direction string,
         Begin_County string,
         Begin_Post_Mile string,
         End_County string,
         End_Post_Mile string,
         Begin_Abs_Post_Mile int,
         End_Abs_Post_Mile int,
         Description string,
         Length__Lane_Miles_ int,
         Opening_Date string,
         Occupancy_Requirement_for_Toll_ string,
         Hours_of_Operation string,
         Comments string,
         Shape_Length int
) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LINES TERMINATED BY '\\n' LOCATION '{}'
TBLPROPERTIES ('skip.header.line.count'='1')""".format(
    database_name, Ex_Ln_table, s3_path_Ex_Ln
)

pd.read_sql(statement1, conn)
```

Out[25]: —

In [26]:
```
statement2 = """CREATE EXTERNAL TABLE IF NOT EXISTS {}.{}(
         OBJECTID int,
         District int,
         Route string,
         Direction string,
         Begin_County string,
```

```
        Begin_Post_Mile string,
        End_County string,
        End_Post_Mile string,
        Begin_Abs_Post_Mile int,
        End_Abs_Post_Mile int,
        Description string,
        Length__Lane_Miles_ int,
        Occupancy_Requirement string,
        Hours_of_Operation string,
        Segment_Opening_Date string,
        Comments string,
        Shape_Length int
) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LINES TERMINATED BY '\\n' LOCATION '{}'
TBLPROPERTIES ('skip.header.line.count'='1')""".format(
    database_name, HOV_table, s3_path_HOV
)

pd.read_sql(statement2, conn)
```

Out[26]: —

In [27]:
```
statement3 = """CREATE EXTERNAL TABLE IF NOT EXISTS {}.{}(
        OBJECTID int,
        District int,
        Route int,
        RTE_SFX string,
        COUNTY string,
        PM_PFX string,
        PM int,
        PM_SFX string,
        DESCRIPTION string,
        BACK_PEAK_HOUR int,
        BACK_PEAK_MADT int,
        BACK_AADT int,
        AHEAD_PEAK_HOUR int,
        AHEAD_PEAK_MADT int,
        AHEAD_AADT int
) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LINES TERMINATED BY '\\n' LOCATION '{}'
TBLPROPERTIES ('skip.header.line.count'='1')""".format(
    database_name, vol_table, s3_path_vol
)

pd.read_sql(statement3, conn)
```

Out[27]: —

In [28]:
```
statement4 = """CREATE EXTERNAL TABLE IF NOT EXISTS {}.{}(
         Date DATE,
         Day string,
         Lane string,
         Date_Time TIMESTAMP,
         Time TIMESTAMP,
         Minimum int,
         Mean int,
         Maximum int,
         Lane_Pts_num int,
         Oberserved_percent int
) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LINES TERMINATED BY '\\n' LOCATION '{}'
TBLPROPERTIES ('skip.header.line.count'='1')""".format(
    database_name, Route77_table, s3_path_Route77
)

pd.read_sql(statement4, conn)
```

Out[28]: —

In [29]:
```
statement = "SHOW TABLES in {}".format(database_name)

df_show = pd.read_sql(statement, conn)
df_show.head(5)
```

Out[29]:

	tab_name
0	ex_ln
1	hov
2	route77
3	vol

In [30]:
```
#HOV


HOV_df = pd.read_csv("s3://sagemaker-us-east-1-938981654669/GoldenDrive/Data/HOV.csv")
HOV_df.info()
HOV_df.head(5)
```

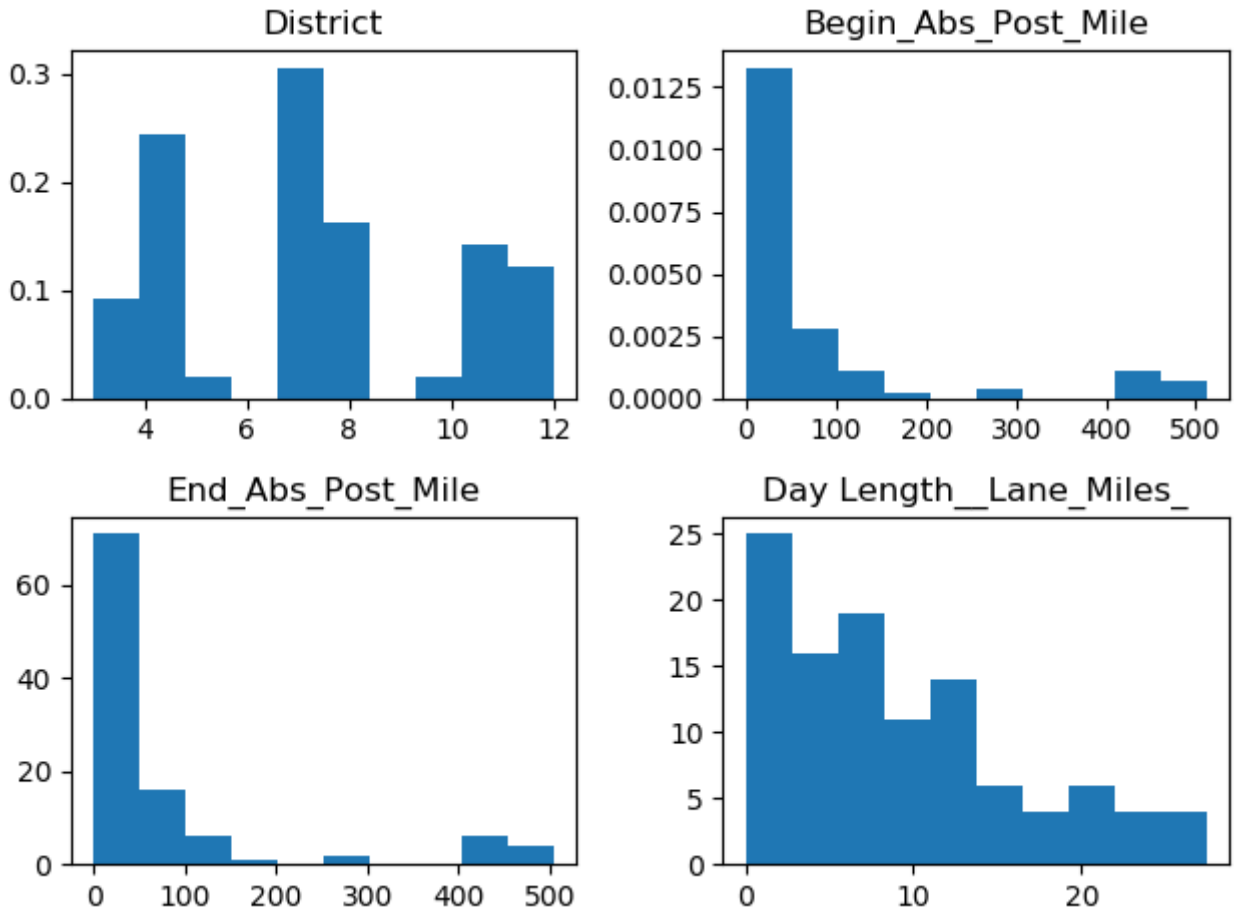
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 109 entries, 0 to 108
Data columns (total 17 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   OBJECTID               109 non-null    int64
 1   District               109 non-null    int64
 2   Route                  109 non-null    object
 3   Direction              109 non-null    object
 4   Begin_County           109 non-null    object
 5   Begin_Post_Mile        109 non-null    object
 6   End_County             109 non-null    object
 7   End_Post_Mile          109 non-null    object
 8   Begin_Abs_Post_Mile    106 non-null    float64
 9   End_Abs_Post_Mile      106 non-null    float64
 10  Description            109 non-null    object
 11  Length__Lane_Miles_    109 non-null    float64
 12  Occupancy_Requirement  109 non-null    object
 13  Hours_of_Operation     109 non-null    object
 14  Segment_Opening_Date   105 non-null    object
 15  Comments               83 non-null     object
 16  Shape_Length           109 non-null    float64
dtypes: float64(4), int64(2), object(11)
memory usage: 14.6+ KB
```

Out[30]:

	OBJECTID	District	Route	Direction	Begin_County	Begin_Post_Mile	End_County	End_Post_Mile	Begin_Abs_Post_Mile	End_A
0	1	3	50	EB	SAC	R5.371	ED	5.834	11.005	
1	2	3	80	EB	SAC	M0.767	PLA	4.718	84.691	
2	3	4	80	EB	SOL	0.504	SOL	0.673	27.995	
3	4	4	80	EB	SOL	R11.485	SOL	19.594	38.976	
4	5	7	10	EB	LA	30.995	LA	48.265	29.495	

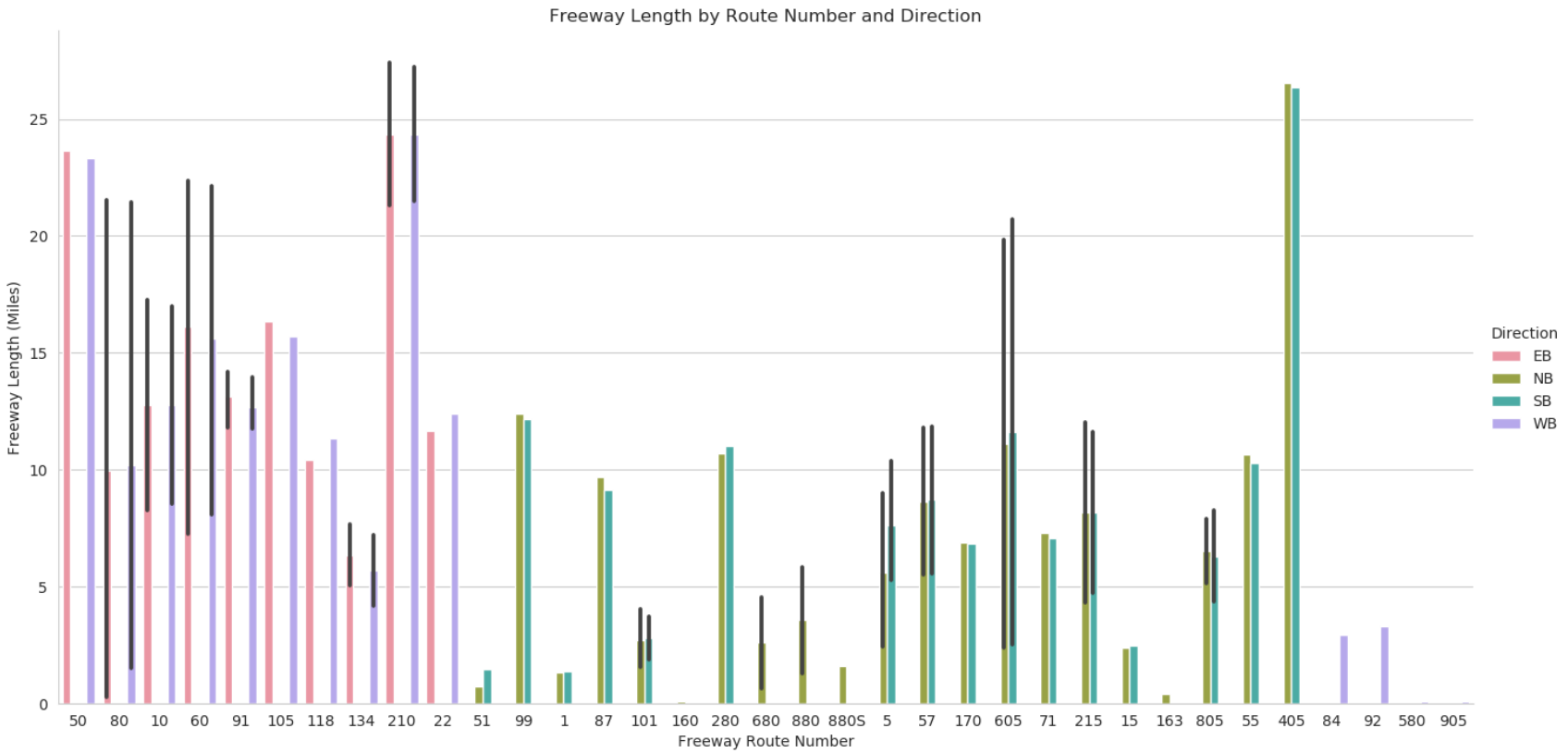
In [31]:

```
n_bins = 10
fig, ((ax0, ax1), (ax2, ax3)) = plt.subplots(nrows=2, ncols=2)

ax0.hist(HOV_df['District'], n_bins, density=True, histtype='bar', stacked = True)
ax0.set_title('District')

ax1.hist(HOV_df['Begin_Abs_Post_Mile'], n_bins, density=True, histtype='bar', stacked=True)
ax1.set_title('Begin_Abs_Post_Mile')

ax2.hist(HOV_df['End_Abs_Post_Mile'], n_bins, histtype='bar', stacked=True)
ax2.set_title('End_Abs_Post_Mile')

ax3.hist(HOV_df['Length__Lane_Miles_'], n_bins, histtype='bar', stacked=True)
ax3.set_title('Day Length__Lane_Miles_')

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



In [32]:

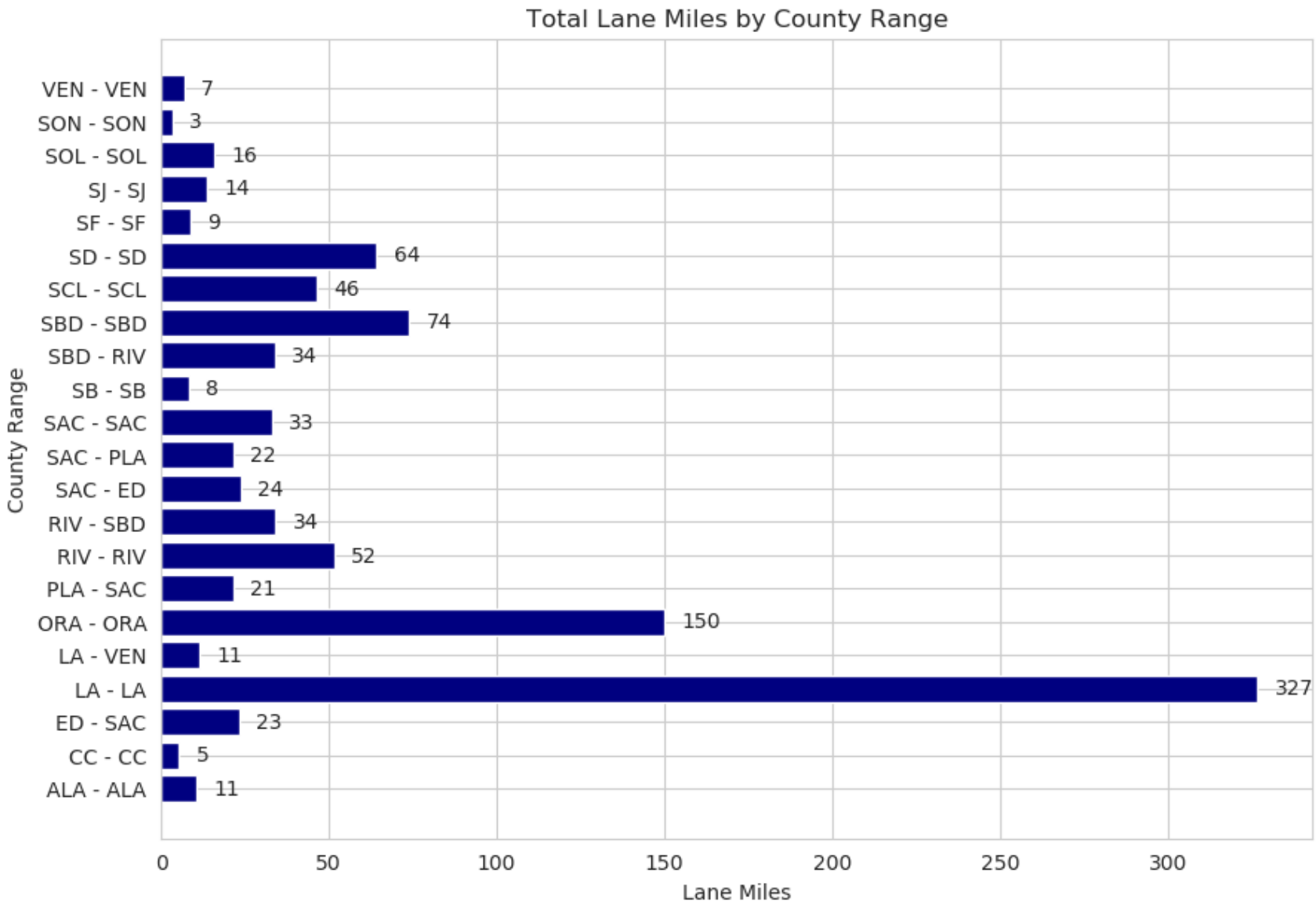
```
# Create a grouped bar chart using seaborn
sns.set_style("whitegrid")
sns.set_palette("husl", 2)
ax = sns.catplot(x='Route', y='Length__Lane_Miles_', hue='Direction',
                 kind='bar', data=HOV_df, height = 7, aspect=2)

# Set the axis labels and title
ax.set_xlabels('Freeway Route Number')
ax.set_ylabels('Freeway Length (Miles)')
ax.set(title='Freeway Length by Route Number and Direction')

# Show the plot
plt.show()
```



In [33]:

```
# Group the data by county and sum the lane miles
grouped_data = HOV_df.groupby(['Begin_County', 'End_County']).agg({'Length__Lane_Miles_': 'sum'}).reset_inde

# Create a horizontal bar chart
fig, ax = plt.subplots(figsize=(10, 7))
bars = ax.barh(y=grouped_data['Begin_County'] + ' - ' + grouped_data['End_County'], width=grouped_data['Leng

# Set the axis labels and title
ax.set_xlabel('Lane Miles')
ax.set_ylabel('County Range')
ax.set_title('Total Lane Miles by County Range')

# Add labels to the bars
for i, bar in enumerate(bars):
    value = bar.get_width()
    x_pos = value + 5
    y_pos = i
    ax.text(x_pos, y_pos, str(round(value)), va='center')
```

```
# Show the plot
plt.show()
```



In [34]:
```
#Traffic volume table

# vol_df = pd.read_csv("s3://sagemaker-us-east-1-938981654669/GoldenDrive/Data/Traffic_Volumes_AADT.csv")
vol_df = pd.read_csv("s3://sagemaker-us-east-1-938981654669/GoldenDrive/Data/HOV2.csv")
#"s3://sagemaker-us-east-1-938981654669/GoldenDrive/Data/HOV.csv"
```

```
#vol_df.info()
#vol_df.head(5)
```

In [ ]:

In [35]:
```
grouped_data = vol_df.groupby('ROUTE').mean().reset_index()
grouped_data_back = grouped_data[['ROUTE', 'BACK_PEAK_HOUR','BACK_AADT']]
grouped_data_back['Rate'] = (grouped_data_back['BACK_PEAK_HOUR'] / grouped_data_back['BACK_AADT']) * 100
grouped_data_back = grouped_data_back.sort_values( "Rate", ascending = False).head(10)
grouped_data_back.style.background_gradient(cmap="Blues")
grouped_data_back
```

Out[35]:

	ROUTE	BACK_PEAK_HOUR	BACK_AADT	Rate
187	220	474.000000	322.000000	147.204969
178	207	300.000000	740.000000	40.540541
120	136	80.000000	280.000000	28.571429
225	284	100.000000	370.000000	27.027027
217	270	120.000000	450.000000	26.666667
214	266	40.000000	160.000000	25.000000
206	254	263.333333	1106.666667	23.795181
146	167	20.000000	90.000000	22.222222
131	150	1967.333333	9116.666667	21.579525
138	158	255.000000	1187.500000	21.473684

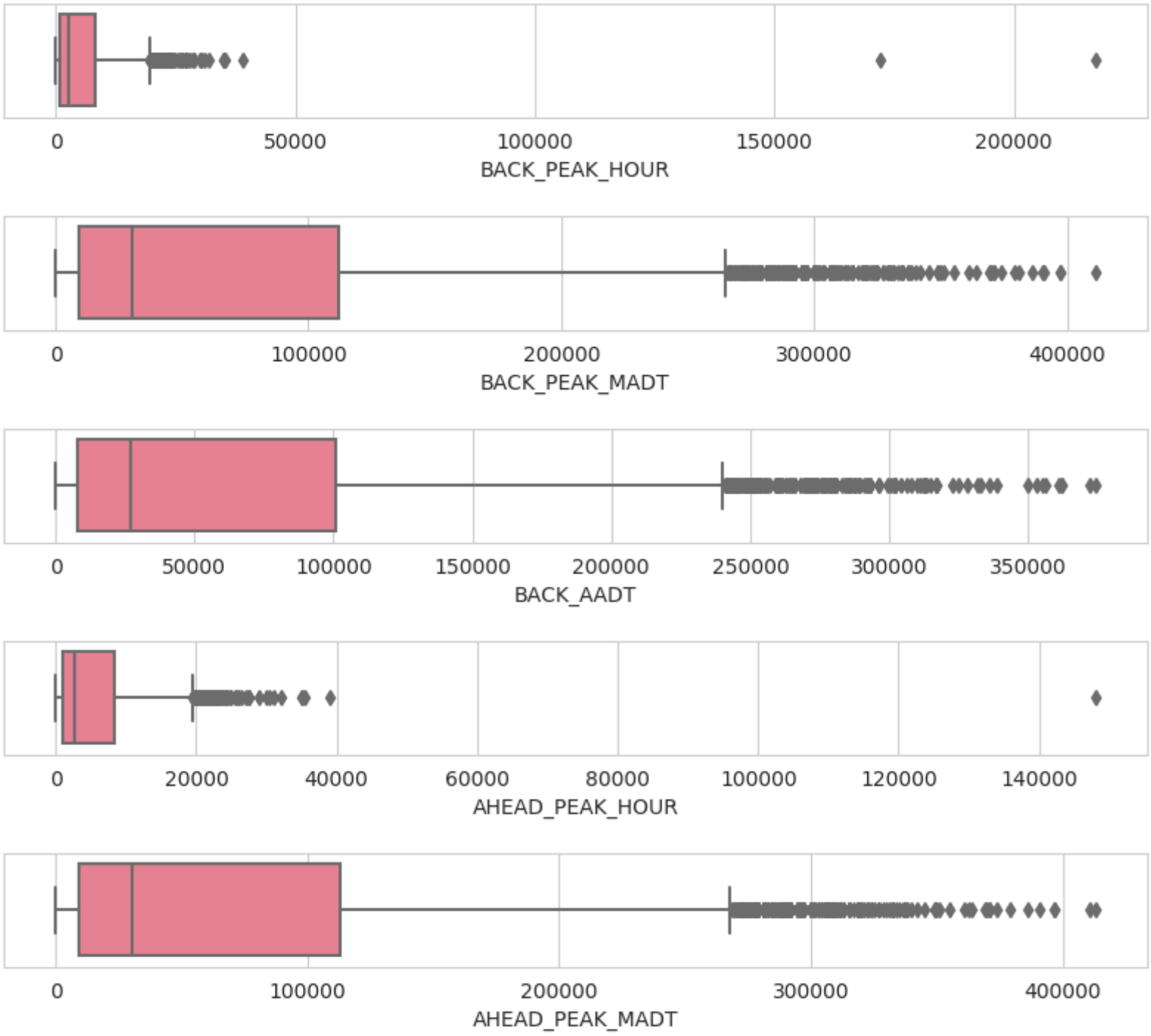
In [36]:
```
grouped_data_ahead = grouped_data[['ROUTE', 'AHEAD_PEAK_HOUR','AHEAD_AADT']]
grouped_data_ahead['Rate'] = (grouped_data_ahead['AHEAD_PEAK_HOUR'] / grouped_data_ahead['AHEAD_AADT']) * 10
grouped_data_ahead = grouped_data_ahead.sort_values( "Rate", ascending = False).head(10)
grouped_data_ahead.style.background_gradient(cmap="Blues")
grouped_data_ahead
```

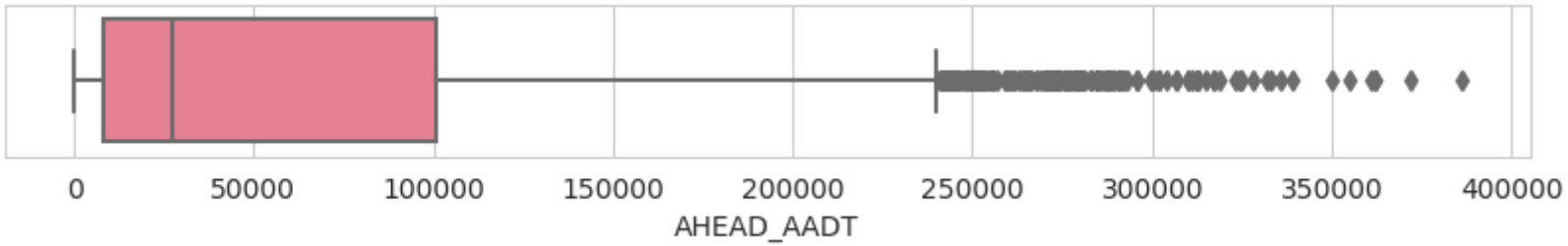
Out[36]:

	ROUTE	AHEAD_PEAK_HOUR	AHEAD_AADT	Rate
217	270	90.000000	210.000000	42.857143
178	207	300.000000	750.000000	40.000000
192	229	60.000000	170.000000	35.294118
214	266	75.000000	215.000000	34.883721
84	95	718.000000	2685.000000	26.741155
187	220	76.000000	322.000000	23.602484
85	96	272.727273	1155.681818	23.598820
138	158	315.000000	1342.500000	23.463687
225	284	150.000000	660.000000	22.727273
218	271	51.428571	226.428571	22.712934

In [37]:
```
fig,ax = plt.subplots(figsize = (10,1))
sns.boxplot(vol_df.BACK_PEAK_HOUR)

fig,ax = plt.subplots(figsize = (10,1))
sns.boxplot(vol_df.BACK_PEAK_MADT)

fig,ax = plt.subplots(figsize = (10,1))
sns.boxplot(vol_df.BACK_AADT)

fig,ax = plt.subplots(figsize = (10,1))
sns.boxplot(vol_df.AHEAD_PEAK_HOUR)

fig,ax = plt.subplots(figsize = (10,1))
sns.boxplot(vol_df.AHEAD_PEAK_MADT)

fig,ax = plt.subplots(figsize = (10,1))
sns.boxplot(vol_df.AHEAD_AADT)
```

Out[37]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff1db67fd50>
```


0
50000
100000
150000
200000
BACK_PEAK_HOUR
0
100000
200000
300000
400000
BACK_PEAK_MADT
0
50000
100000
150000
200000
250000
300000
350000
BACK_AADT
0
20000
40000
60000
80000
100000
120000
140000
AHEAD_PEAK_HOUR
0
100000
200000
300000
400000
AHEAD_PEAK_MADT



In [38]:
```
#route77


route77_df = pd.read_csv("s3://sagemaker-us-east-1-938981654669/GoldenDrive/Data/Route_77.csv")
route77_df = pd.DataFrame(route77_df)
route77_df.info()
route77_df.head(5)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10080 entries, 0 to 10079
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Date                10080 non-null  object
 1   Day                 10080 non-null  object
 2   Lane                10080 non-null  object
 3   Date_Time           10080 non-null  object
 4   Time                10080 non-null  object
 5   Minimum             10080 non-null  float64
 6   Mean                10080 non-null  float64
 7   Maximum             10080 non-null  float64
 8   Lane_Pts_num        10080 non-null  int64
 9   Oberserved_percent  10080 non-null  float64
dtypes: float64(4), int64(1), object(5)
memory usage: 787.6+ KB
```

Out[38]:

	Date	Day	Lane	Date_Time	Time	Minimum	Mean	Maximum	Lane_Pts_num	Oberserved_percent
0	3/5/23	Sunday	Lane1	3/5/23 0:00	0:00:00	30.37	30.37	30.37	318	38.7
1	3/5/23	Sunday	Lane1	3/5/23 0:05	0:05:00	30.40	30.40	30.40	318	39.3
2	3/5/23	Sunday	Lane1	3/5/23 0:10	0:10:00	30.45	30.45	30.45	317	39.1
3	3/5/23	Sunday	Lane1	3/5/23 0:15	0:15:00	30.52	30.52	30.52	317	38.8
4	3/5/23	Sunday	Lane1	3/5/23 0:20	0:20:00	30.40	30.40	30.40	318	39.3

In [39]:
```
route77_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10080 entries, 0 to 10079
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Date                10080 non-null  object 
 1   Day                 10080 non-null  object 
 2   Lane                10080 non-null  object 
 3   Date_Time           10080 non-null  object 
 4   Time                10080 non-null  object 
 5   Minimum             10080 non-null  float64
 6   Mean                10080 non-null  float64
 7   Maximum             10080 non-null  float64
 8   Lane_Pts_num        10080 non-null  int64  
 9   Oberserved_percent  10080 non-null  float64
dtypes: float64(4), int64(1), object(5)
memory usage: 787.6+ KB
```

In [40]:
```
# Add the needed columns for better visualization, and transform the data type.
route77_df['Date_Time'] = pd.to_datetime(route77_df['Date_Time'])
route77_df['dt'] = pd.to_datetime(route77_df['Date_Time'])
route77_df['dt'] = route77_df['dt'].dt.date
route77_df['time_s']  = route77_df['Date_Time'].dt.time
route77_df
```

Out[40]:

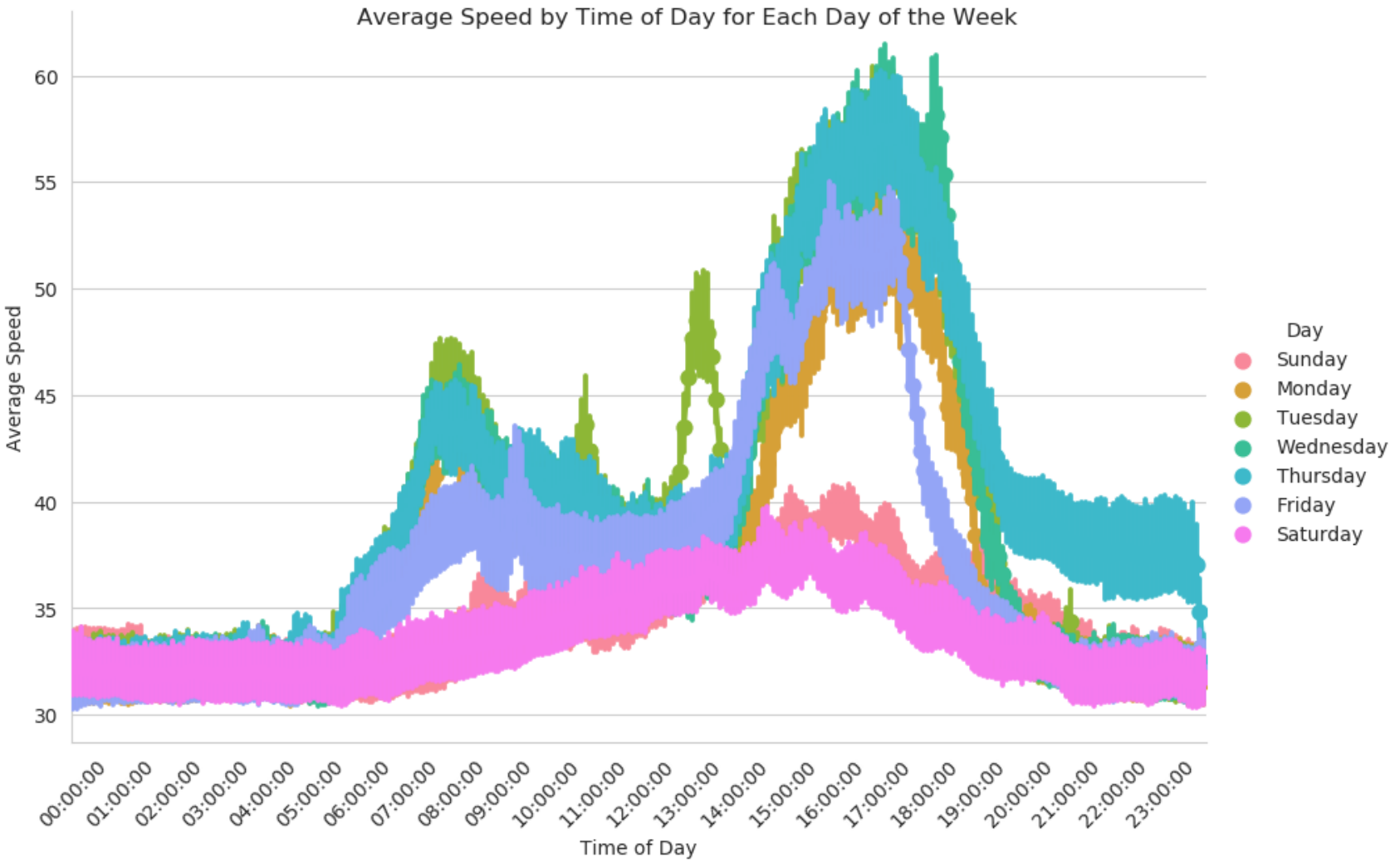
	Date	Day	Lane	Date_Time	Time	Minimum	Mean	Maximum	Lane_Pts_num	Oberserved_percent	dt	tir
0	3/5/23	Sunday	Lane1	2023-03-05 00:00:00	0:00:00	30.37	30.37	30.37	318	38.7	2023-03-05	00:0
1	3/5/23	Sunday	Lane1	2023-03-05 00:05:00	0:05:00	30.40	30.40	30.40	318	39.3	2023-03-05	00:0
2	3/5/23	Sunday	Lane1	2023-03-05 00:10:00	0:10:00	30.45	30.45	30.45	317	39.1	2023-03-05	00:1
3	3/5/23	Sunday	Lane1	2023-03-05 00:15:00	0:15:00	30.52	30.52	30.52	317	38.8	2023-03-05	00:1
4	3/5/23	Sunday	Lane1	2023-03-05 00:20:00	0:20:00	30.40	30.40	30.40	318	39.3	2023-03-05	00:2
...	...	...	...	...	...	...	...	...	...	...	...	
10075	3/11/23	Saturday	HOV	2023-03-11 23:35:00	23:35:00	31.30	31.30	31.30	316	5.1	2023-03-11	23:3
10076	3/11/23	Saturday	HOV	2023-03-11 23:40:00	23:40:00	31.25	31.25	31.25	317	5.0	2023-03-11	23:4
10077	3/11/23	Saturday	HOV	2023-03-11 23:45:00	23:45:00	31.23	31.23	31.23	316	5.1	2023-03-11	23:4
10078	3/11/23	Saturday	HOV	2023-03-11 23:50:00	23:50:00	31.27	31.27	31.27	317	5.0	2023-03-11	23:5
10079	3/11/23	Saturday	HOV	2023-03-11 23:55:00	23:55:00	31.32	31.32	31.32	317	5.0	2023-03-11	23:5

10080 rows × 12 columns

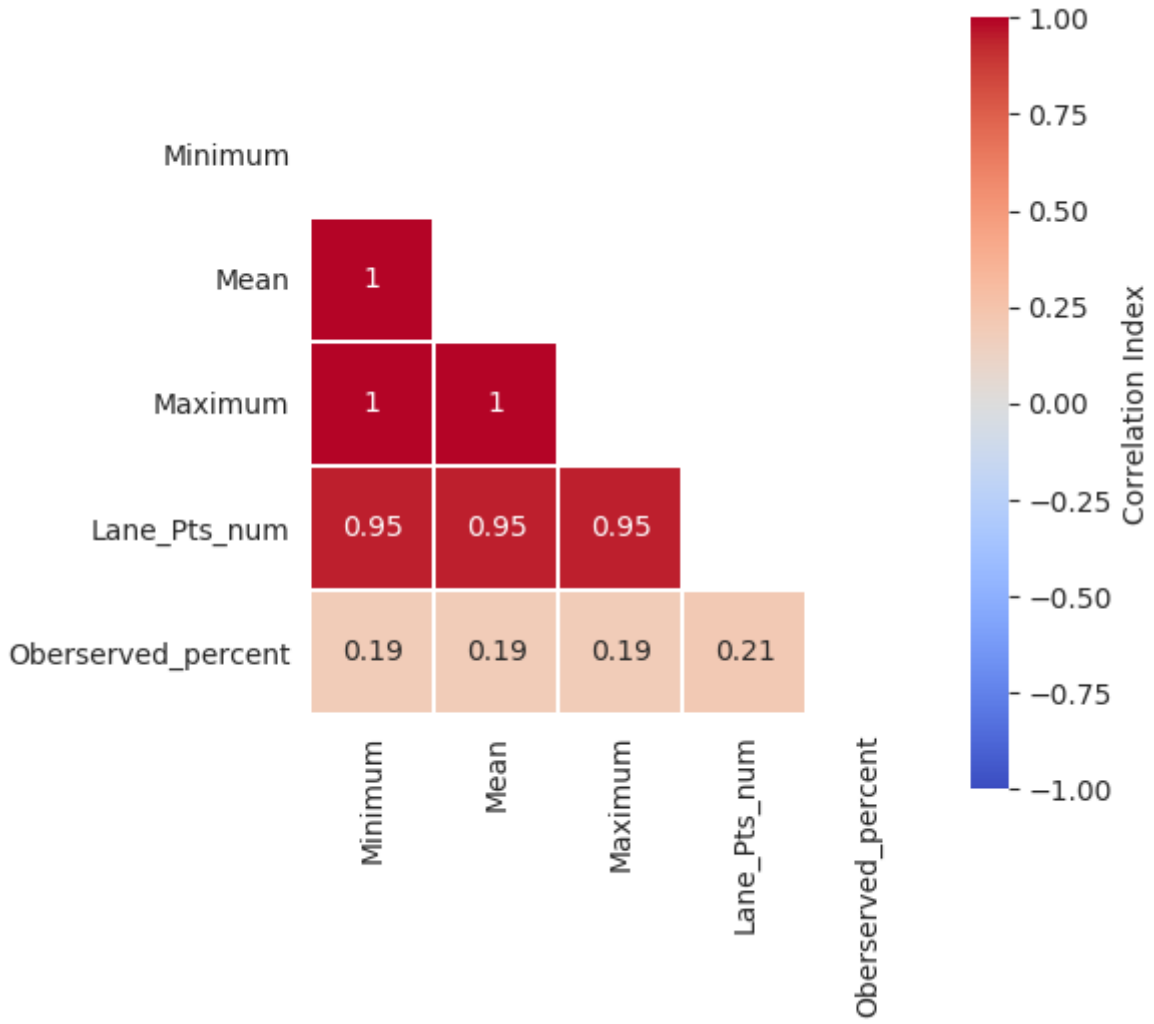
In [41]:

```
import matplotlib.ticker as ticker

sns.set_style("whitegrid")
```

```
g = sns.catplot(x="time_s", y="Mean", hue="Day", data=route77_df, kind="point", height=6, aspect=1.5)

g.set_xlabels("Time of Day")
g.set_ylabels("Average Speed")
plt.xticks(rotation=45)
g.fig.suptitle("Average Speed by Time of Day for Each Day of the Week")

ax = g.facet_axis(0,0)
every_nth = 12
for i, label in enumerate(ax.get_xticklabels()):
    if i % every_nth != 0:
        label.set_visible(False)

plt.show()
```



In [42]:
```
# assign correlation function to new variable
corr = route77_df.corr()
matrix = np.triu(corr) # for triangular matrix
plt.figure(figsize=(5,5))
# parse corr variable intro triangular matrix
sns.heatmap(route77_df.corr(method='pearson'),
            annot=True, linewidths=.5,
            cmap="coolwarm", mask=matrix,
            square = True,
            cbar_kws={'label': 'Correlation Index'},
            vmin=-1, vmax=1)
plt.show()
```



In [43]:
```
# create a new table to store the needed data
route = pd.DataFrame([route77_df.dt, route77_df.Day,route77_df.Lane,route77_df.Mean,route77_df.time_s]).tran
route.head(5)
```

Out[43]:

	dt	Day	Lane	Mean	time_s
0	2023-03-05	Sunday	Lane1	30.37	00:00:00
1	2023-03-05	Sunday	Lane1	30.40	00:05:00
2	2023-03-05	Sunday	Lane1	30.45	00:10:00
3	2023-03-05	Sunday	Lane1	30.52	00:15:00
4	2023-03-05	Sunday	Lane1	30.40	00:20:00

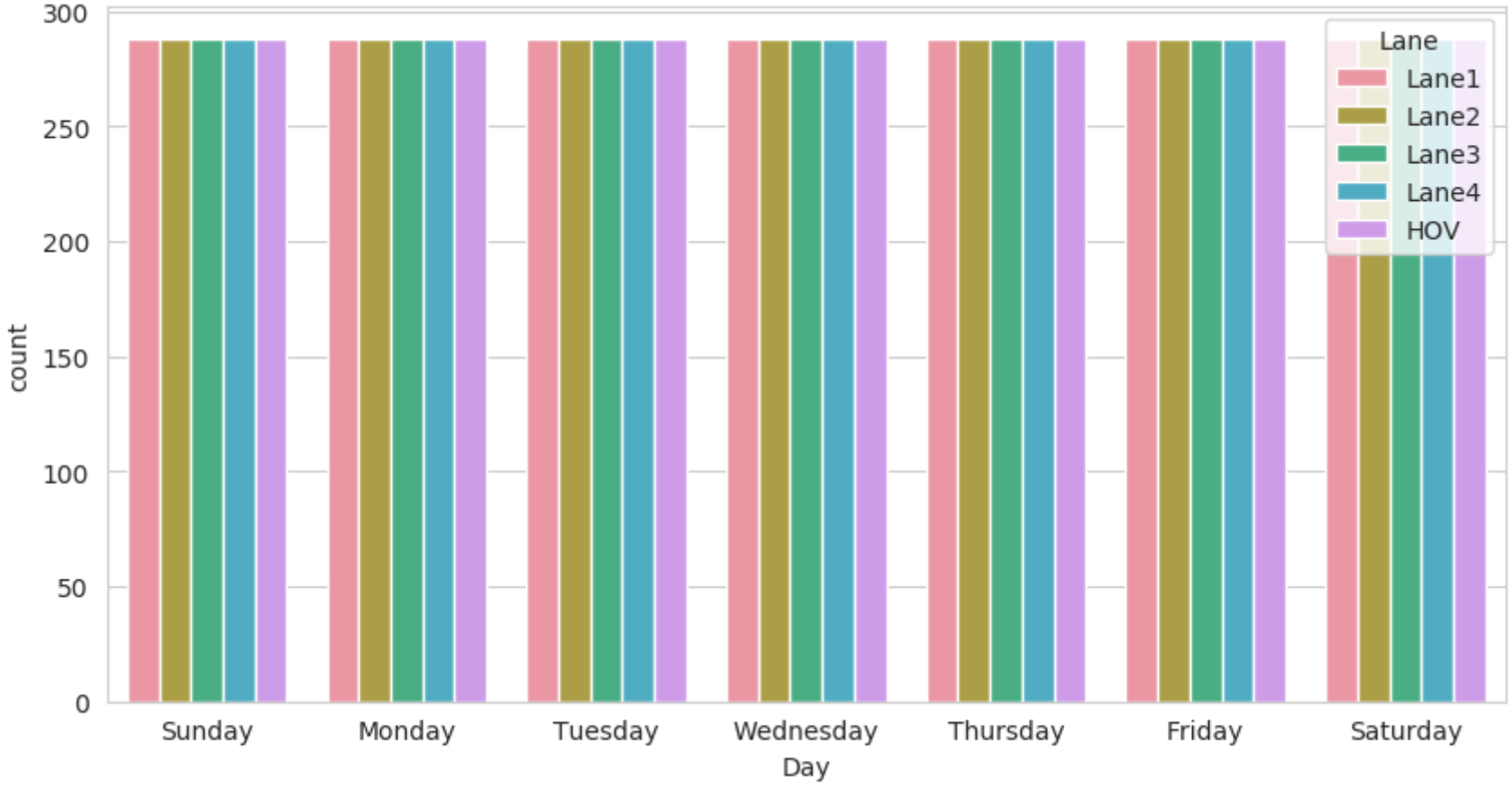
In [44]:
```
route77_df.columns
```

Out[44]:
```
Index(['Date', 'Day', 'Lane', 'Date_Time', 'Time', 'Minimum', 'Mean',
       'Maximum', 'Lane_Pts_num', 'Oberserved_percent', 'dt', 'time_s'],
      dtype='object')
```

In [45]:
```
route['dt'] = pd.to_datetime(route['dt'])
# route['time_s'] = pd.to_datetime(route['time_s'])
route.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10080 entries, 0 to 10079
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype         
---  ------  --------------  -----         
 0   dt      10080 non-null  datetime64[ns]
 1   Day     10080 non-null  object        
 2   Lane    10080 non-null  object        
 3   Mean    10080 non-null  object        
 4   time_s  10080 non-null  object        
dtypes: datetime64[ns](1), object(4)
memory usage: 393.9+ KB
```

In [46]:
```
plt.figure(figsize=(10,5))
sns.countplot(data=route, x="Day", hue="Lane")
plt.show()
```



In [47]:
```
# split hov


hov_df = route[route['Lane'] == 'HOV']
hov_df
```

Out[47]:

	dt	Day	Lane	Mean	time_s
1152	2023-03-05	Sunday	HOV	31.95	00:00:00
1153	2023-03-05	Sunday	HOV	31.97	00:05:00
1154	2023-03-05	Sunday	HOV	32.00	00:10:00
1155	2023-03-05	Sunday	HOV	32.03	00:15:00
1156	2023-03-05	Sunday	HOV	31.93	00:20:00
...	...	...	...	...	...
10075	2023-03-11	Saturday	HOV	31.30	23:35:00
10076	2023-03-11	Saturday	HOV	31.25	23:40:00
10077	2023-03-11	Saturday	HOV	31.23	23:45:00
10078	2023-03-11	Saturday	HOV	31.27	23:50:00
10079	2023-03-11	Saturday	HOV	31.32	23:55:00

2016 rows × 5 columns

In [48]:

```
lan_df = route[route['Lane'] != 'HOV']
lan_df
```

Out[48]:

	dt	Day	Lane	Mean	time_s
0	2023-03-05	Sunday	Lane1	30.37	00:00:00
1	2023-03-05	Sunday	Lane1	30.40	00:05:00
2	2023-03-05	Sunday	Lane1	30.45	00:10:00
3	2023-03-05	Sunday	Lane1	30.52	00:15:00
4	2023-03-05	Sunday	Lane1	30.40	00:20:00
...	...	...	...	...	...
9787	2023-03-11	Saturday	Lane4	34.02	23:35:00
9788	2023-03-11	Saturday	Lane4	34.05	23:40:00
9789	2023-03-11	Saturday	Lane4	34.05	23:45:00
9790	2023-03-11	Saturday	Lane4	34.10	23:50:00
9791	2023-03-11	Saturday	Lane4	34.28	23:55:00

8064 rows × 5 columns

In [49]:

```
merged_df = pd.merge(hov_df, lan_df, on=[ 'Day','time_s'])
merged_df
```

Out[49]:

	dt_x	Day	Lane_x	Mean_x	time_s	dt_y	Lane_y	Mean_y
0	2023-03-05	Sunday	HOV	31.95	00:00:00	2023-03-05	Lane1	30.37
1	2023-03-05	Sunday	HOV	31.95	00:00:00	2023-03-05	Lane2	31.43
2	2023-03-05	Sunday	HOV	31.95	00:00:00	2023-03-05	Lane3	34.08
3	2023-03-05	Sunday	HOV	31.95	00:00:00	2023-03-05	Lane4	34.85
4	2023-03-05	Sunday	HOV	31.97	00:05:00	2023-03-05	Lane1	30.40
...	...	...	...	...	...	...	...	...
8059	2023-03-11	Saturday	HOV	31.27	23:50:00	2023-03-11	Lane4	34.10
8060	2023-03-11	Saturday	HOV	31.32	23:55:00	2023-03-11	Lane1	29.60
8061	2023-03-11	Saturday	HOV	31.32	23:55:00	2023-03-11	Lane2	30.95
8062	2023-03-11	Saturday	HOV	31.32	23:55:00	2023-03-11	Lane3	32.57
8063	2023-03-11	Saturday	HOV	31.32	23:55:00	2023-03-11	Lane4	34.28

8064 rows × 8 columns

In [50]:

```
dummies = pd.get_dummies(merged_df[['Lane_y', 'Day', 'time_s']])
new_dum = pd.concat([merged_df, dummies], axis=1)

# drop the original 'Lane' and 'Day_of_Week' columns
new_dum.drop(['Lane_y', 'Day','time_s'], axis=1, inplace=True)

# print the resulting dataframe
print(new_dum)
```

```
           dt_x Lane_x Mean_x       dt_y Mean_y  Lane_y_Lane1  Lane_y_Lane2  \
0    2023-03-05    HOV  31.95 2023-03-05  30.37             1             0
1    2023-03-05    HOV  31.95 2023-03-05  31.43             0             1
2    2023-03-05    HOV  31.95 2023-03-05  34.08             0             0
3    2023-03-05    HOV  31.95 2023-03-05  34.85             0             0
4    2023-03-05    HOV  31.97 2023-03-05  30.40             1             0
...         ...    ...    ...        ...    ...           ...           ...
8059 2023-03-11    HOV  31.27 2023-03-11  34.10             0             0
8060 2023-03-11    HOV  31.32 2023-03-11  29.60             1             0
8061 2023-03-11    HOV  31.32 2023-03-11  30.95             0             1
8062 2023-03-11    HOV  31.32 2023-03-11  32.57             0             0
8063 2023-03-11    HOV  31.32 2023-03-11  34.28             0             0

      Lane_y_Lane3  Lane_y_Lane4  Day_Friday  ...  time_s_23:10:00  \
0                0             0           0  ...                0
1                0             0           0  ...                0
2                1             0           0  ...                0
3                0             1           0  ...                0
4                0             0           0  ...                0
...            ...           ...         ...  ...              ...
8059             0             1           0  ...                0
8060             0             0           0  ...                0
8061             0             0           0  ...                0
8062             1             0           0  ...                0
8063             0             1           0  ...                0

      time_s_23:15:00  time_s_23:20:00  time_s_23:25:00  time_s_23:30:00  \
0                   0                0                0                0
1                   0                0                0                0
2                   0                0                0                0
3                   0                0                0                0
4                   0                0                0                0
...               ...              ...              ...              ...
8059                0                0                0                0
8060                0                0                0                0
8061                0                0                0                0
8062                0                0                0                0
8063                0                0                0                0

      time_s_23:35:00  time_s_23:40:00  time_s_23:45:00  time_s_23:50:00  \
0                   0                0                0                0
1                   0                0                0                0
2                   0                0                0                0
3                   0                0                0                0
4                   0                0                0                0
```

```
...                ...                ...                ...                ...
8059                 0                  0                  0                  1
8060                 0                  0                  0                  0
8061                 0                  0                  0                  0
8062                 0                  0                  0                  0
8063                 0                  0                  0                  0

      time_s_23:55:00
0                    0
1                    0
2                    0
3                    0
4                    0
...                ...
8059                 0
8060                 1
8061                 1
8062                 1
8063                 1

[8064 rows x 304 columns]
```

In [ ]:

Modeling

In [51]:
```
# Get the variable list and remove the target variable and irrelevant columns
column_names = list(new_dum.columns)
column_names.remove('dt_x')
column_names.remove('Lane_x')
column_names.remove('Mean_x')
column_names.remove('dt_y')
# Print the list of column names
print(column_names)
```

['Mean_y', 'Lane_y_Lane1', 'Lane_y_Lane2', 'Lane_y_Lane3', 'Lane_y_Lane4', 'Day_Friday', 'Day_Monday', 'Day_
Saturday', 'Day_Sunday', 'Day_Thursday', 'Day_Tuesday', 'Day_Wednesday', 'time_s_00:00:00', 'time_s_00:05:0
0', 'time_s_00:10:00', 'time_s_00:15:00', 'time_s_00:20:00', 'time_s_00:25:00', 'time_s_00:30:00', 'time_s_0
0:35:00', 'time_s_00:40:00', 'time_s_00:45:00', 'time_s_00:50:00', 'time_s_00:55:00', 'time_s_01:00:00', 'ti
me_s_01:05:00', 'time_s_01:10:00', 'time_s_01:15:00', 'time_s_01:20:00', 'time_s_01:25:00', 'time_s_01:30:0
0', 'time_s_01:35:00', 'time_s_01:40:00', 'time_s_01:45:00', 'time_s_01:50:00', 'time_s_01:55:00', 'time_s_0
2:00:00', 'time_s_02:05:00', 'time_s_02:10:00', 'time_s_02:15:00', 'time_s_02:20:00', 'time_s_02:25:00', 'ti
me_s_02:30:00', 'time_s_02:35:00', 'time_s_02:40:00', 'time_s_02:45:00', 'time_s_02:50:00', 'time_s_02:55:0
0', 'time_s_03:00:00', 'time_s_03:05:00', 'time_s_03:10:00', 'time_s_03:15:00', 'time_s_03:20:00', 'time_s_0
3:25:00', 'time_s_03:30:00', 'time_s_03:35:00', 'time_s_03:40:00', 'time_s_03:45:00', 'time_s_03:50:00', 'ti
me_s_03:55:00', 'time_s_04:00:00', 'time_s_04:05:00', 'time_s_04:10:00', 'time_s_04:15:00', 'time_s_04:20:0
0', 'time_s_04:25:00', 'time_s_04:30:00', 'time_s_04:35:00', 'time_s_04:40:00', 'time_s_04:45:00', 'time_s_0
4:50:00', 'time_s_04:55:00', 'time_s_05:00:00', 'time_s_05:05:00', 'time_s_05:10:00', 'time_s_05:15:00', 'ti
me_s_05:20:00', 'time_s_05:25:00', 'time_s_05:30:00', 'time_s_05:35:00', 'time_s_05:40:00', 'time_s_05:45:0
0', 'time_s_05:50:00', 'time_s_05:55:00', 'time_s_06:00:00', 'time_s_06:05:00', 'time_s_06:10:00', 'time_s_0
6:15:00', 'time_s_06:20:00', 'time_s_06:25:00', 'time_s_06:30:00', 'time_s_06:35:00', 'time_s_06:40:00', 'ti
me_s_06:45:00', 'time_s_06:50:00', 'time_s_06:55:00', 'time_s_07:00:00', 'time_s_07:05:00', 'time_s_07:10:0
0', 'time_s_07:15:00', 'time_s_07:20:00', 'time_s_07:25:00', 'time_s_07:30:00', 'time_s_07:35:00', 'time_s_0
7:40:00', 'time_s_07:45:00', 'time_s_07:50:00', 'time_s_07:55:00', 'time_s_08:00:00', 'time_s_08:05:00', 'ti
me_s_08:10:00', 'time_s_08:15:00', 'time_s_08:20:00', 'time_s_08:25:00', 'time_s_08:30:00', 'time_s_08:35:0
0', 'time_s_08:40:00', 'time_s_08:45:00', 'time_s_08:50:00', 'time_s_08:55:00', 'time_s_09:00:00', 'time_s_0
9:05:00', 'time_s_09:10:00', 'time_s_09:15:00', 'time_s_09:20:00', 'time_s_09:25:00', 'time_s_09:30:00', 'ti
me_s_09:35:00', 'time_s_09:40:00', 'time_s_09:45:00', 'time_s_09:50:00', 'time_s_09:55:00', 'time_s_10:00:0
0', 'time_s_10:05:00', 'time_s_10:10:00', 'time_s_10:15:00', 'time_s_10:20:00', 'time_s_10:25:00', 'time_s_1
0:30:00', 'time_s_10:35:00', 'time_s_10:40:00', 'time_s_10:45:00', 'time_s_10:50:00', 'time_s_10:55:00', 'ti
me_s_11:00:00', 'time_s_11:05:00', 'time_s_11:10:00', 'time_s_11:15:00', 'time_s_11:20:00', 'time_s_11:25:0
0', 'time_s_11:30:00', 'time_s_11:35:00', 'time_s_11:40:00', 'time_s_11:45:00', 'time_s_11:50:00', 'time_s_1
1:55:00', 'time_s_12:00:00', 'time_s_12:05:00', 'time_s_12:10:00', 'time_s_12:15:00', 'time_s_12:20:00', 'ti
me_s_12:25:00', 'time_s_12:30:00', 'time_s_12:35:00', 'time_s_12:40:00', 'time_s_12:45:00', 'time_s_12:50:0
0', 'time_s_12:55:00', 'time_s_13:00:00', 'time_s_13:05:00', 'time_s_13:10:00', 'time_s_13:15:00', 'time_s_1
3:20:00', 'time_s_13:25:00', 'time_s_13:30:00', 'time_s_13:35:00', 'time_s_13:40:00', 'time_s_13:45:00', 'ti
me_s_13:50:00', 'time_s_13:55:00', 'time_s_14:00:00', 'time_s_14:05:00', 'time_s_14:10:00', 'time_s_14:15:0
0', 'time_s_14:20:00', 'time_s_14:25:00', 'time_s_14:30:00', 'time_s_14:35:00', 'time_s_14:40:00', 'time_s_1
4:45:00', 'time_s_14:50:00', 'time_s_14:55:00', 'time_s_15:00:00', 'time_s_15:05:00', 'time_s_15:10:00', 'ti
me_s_15:15:00', 'time_s_15:20:00', 'time_s_15:25:00', 'time_s_15:30:00', 'time_s_15:35:00', 'time_s_15:40:0
0', 'time_s_15:45:00', 'time_s_15:50:00', 'time_s_15:55:00', 'time_s_16:00:00', 'time_s_16:05:00', 'time_s_1
6:10:00', 'time_s_16:15:00', 'time_s_16:20:00', 'time_s_16:25:00', 'time_s_16:30:00', 'time_s_16:35:00', 'ti
me_s_16:40:00', 'time_s_16:45:00', 'time_s_16:50:00', 'time_s_16:55:00', 'time_s_17:00:00', 'time_s_17:05:0
0', 'time_s_17:10:00', 'time_s_17:15:00', 'time_s_17:20:00', 'time_s_17:25:00', 'time_s_17:30:00', 'time_s_1
7:35:00', 'time_s_17:40:00', 'time_s_17:45:00', 'time_s_17:50:00', 'time_s_17:55:00', 'time_s_18:00:00', 'ti
me_s_18:05:00', 'time_s_18:10:00', 'time_s_18:15:00', 'time_s_18:20:00', 'time_s_18:25:00', 'time_s_18:30:0
0', 'time_s_18:35:00', 'time_s_18:40:00', 'time_s_18:45:00', 'time_s_18:50:00', 'time_s_18:55:00', 'time_s_1
9:00:00', 'time_s_19:05:00', 'time_s_19:10:00', 'time_s_19:15:00', 'time_s_19:20:00', 'time_s_19:25:00', 'ti
me_s_19:30:00', 'time_s_19:35:00', 'time_s_19:40:00', 'time_s_19:45:00', 'time_s_19:50:00', 'time_s_19:55:0
0', 'time_s_20:00:00', 'time_s_20:05:00', 'time_s_20:10:00', 'time_s_20:15:00', 'time_s_20:20:00', 'time_s_2

```
0:25:00', 'time_s_20:30:00', 'time_s_20:35:00', 'time_s_20:40:00', 'time_s_20:45:00', 'time_s_20:50:00', 'ti
me_s_20:55:00', 'time_s_21:00:00', 'time_s_21:05:00', 'time_s_21:10:00', 'time_s_21:15:00', 'time_s_21:20:0
0', 'time_s_21:25:00', 'time_s_21:30:00', 'time_s_21:35:00', 'time_s_21:40:00', 'time_s_21:45:00', 'time_s_2
1:50:00', 'time_s_21:55:00', 'time_s_22:00:00', 'time_s_22:05:00', 'time_s_22:10:00', 'time_s_22:15:00', 'ti
me_s_22:20:00', 'time_s_22:25:00', 'time_s_22:30:00', 'time_s_22:35:00', 'time_s_22:40:00', 'time_s_22:45:0
0', 'time_s_22:50:00', 'time_s_22:55:00', 'time_s_23:00:00', 'time_s_23:05:00', 'time_s_23:10:00', 'time_s_2
3:15:00', 'time_s_23:20:00', 'time_s_23:25:00', 'time_s_23:30:00', 'time_s_23:35:00', 'time_s_23:40:00', 'ti
me_s_23:45:00', 'time_s_23:50:00', 'time_s_23:55:00']
```

In [52]:
```python
from sklearn.preprocessing import StandardScaler,PolynomialFeatures

X = new_dum[column_names]
Y = new_dum['Mean_x']
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=0)

sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

In [53]:
```python
print("x_train:",x_train.shape)
print("x_test:",x_test.shape)
print("y_train:",y_train.shape[0])
print("y_test:",y_test.shape[0])
```

```
x_train: (6451, 300)
x_test: (1613, 300)
y_train: 6451
y_test: 1613
```

Linear Regression Model

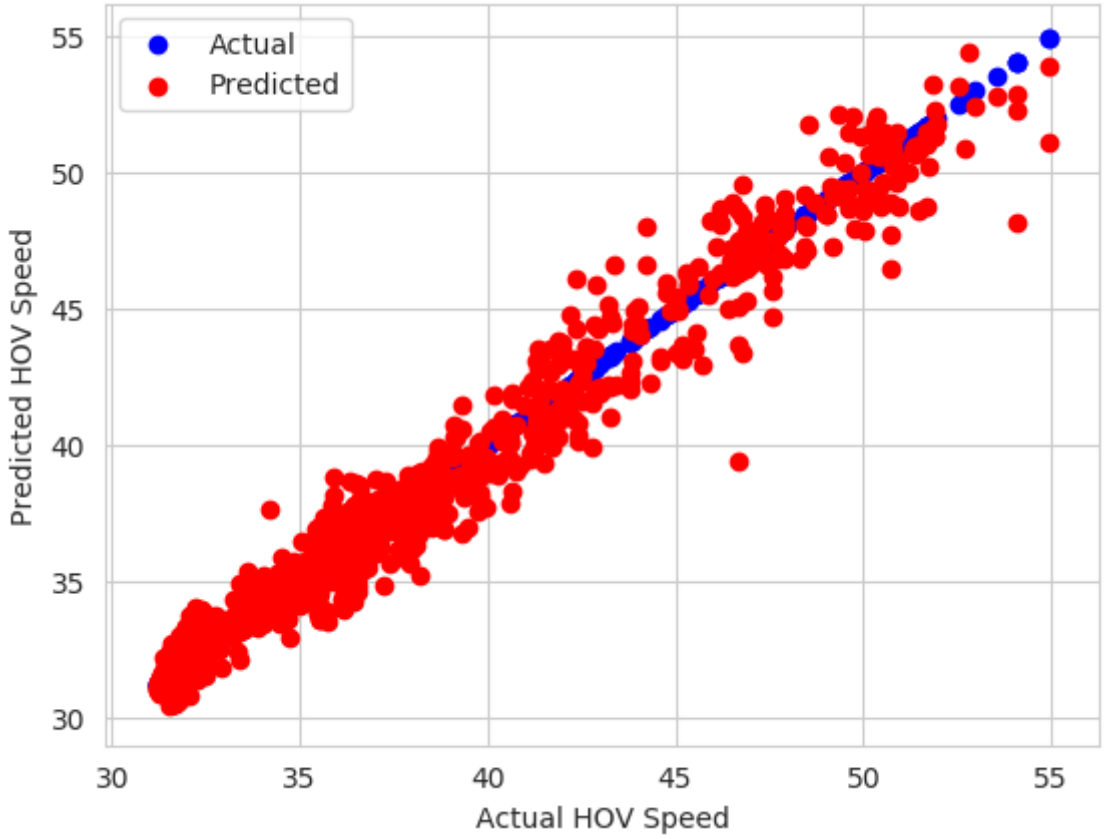
In [54]:
```python
#LinearRegression model
from sklearn.linear_model import LinearRegression
# Train a linear regression model on the training set
regressor = LinearRegression()
# Fit the regressor with the training data
regressor.fit(x_train, y_train)
# Predict the values for the test set
y_pred = regressor.predict(x_test)

rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print("RMSE:", rmse)
```

```
RMSE: 0.8525795828205489
```

In [55]:
```
from sklearn.metrics import r2_score
Accuracy_lr=r2_score(y_test,y_pred)*100
print(" Accuracy of the model is %.2f" %Accuracy_lr)
```

```
 Accuracy of the model is 97.51
```

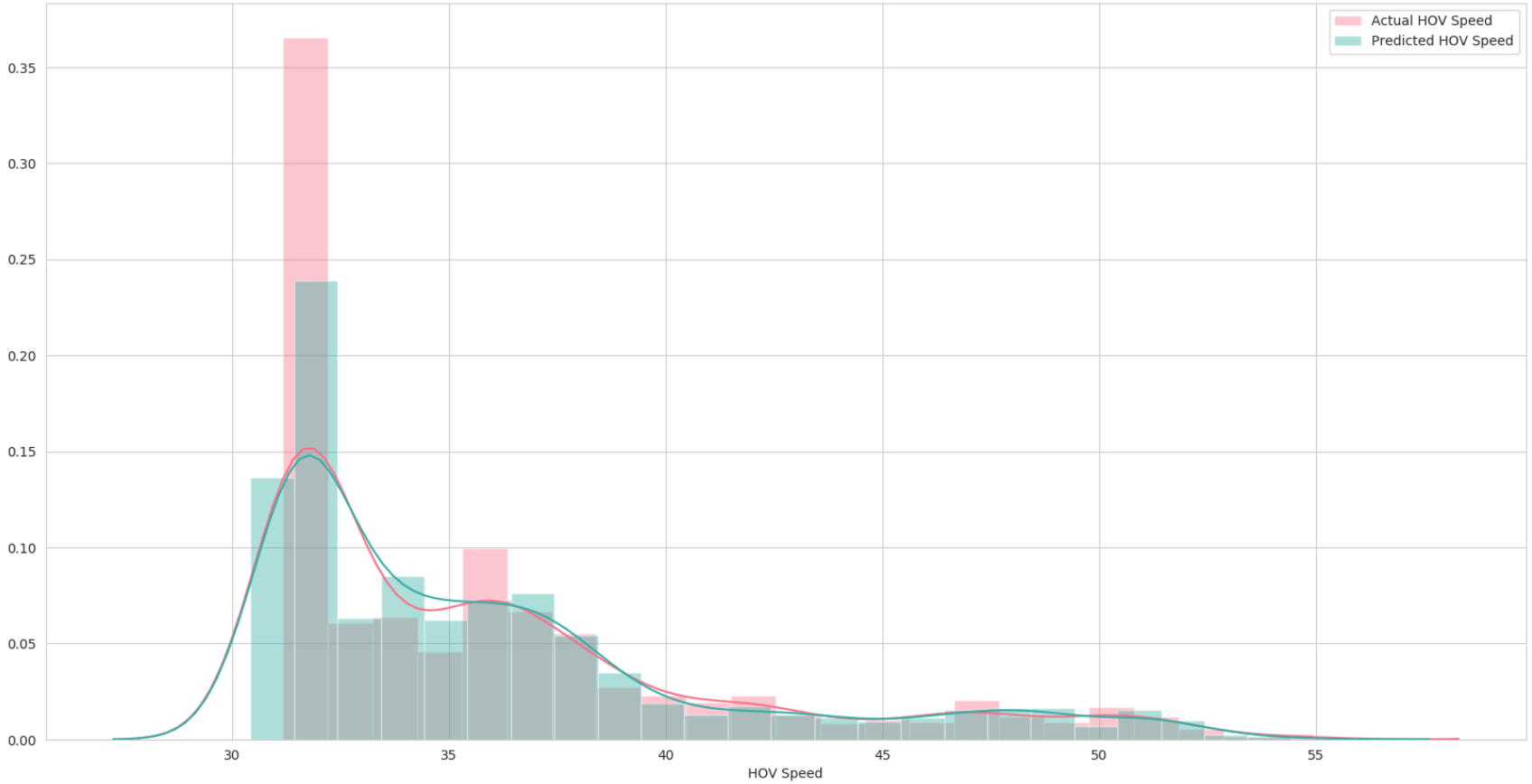
In [56]:
```
# Plot the actual values
plt.scatter(y_test, y_test, color='blue', label='Actual')

# Plot the predicted values
plt.scatter(y_test, y_pred, color='red', label='Predicted')

# Set the axis labels and legend
plt.xlabel('Actual HOV Speed')
plt.ylabel('Predicted HOV Speed')
plt.legend()

# Show the plot
plt.show()
```



In [58]:
```
from sklearn.metrics import mean_squared_error
import math

print('RMSE for linear regression model is: ', math.sqrt(mean_squared_error(y_test, y_pred)))
```

```
RMSE for linear regression model is:  0.8525795828205489
```

In [59]:
```
f, ax = plt.subplots(figsize=(20, 10))
sns.distplot(y_test, hist=True, label="Actual HOV Speed")
sns.distplot(y_pred, hist=True, label="Predicted HOV Speed")
plt.xlabel("HOV Speed")
plt.legend()
plt.show()
```



In [ ]:

In [ ]:

SVM Model

In [60]:
```
import pandas as pd
from sklearn import svm
RegModel = svm.SVR(C=2, kernel='linear')
```

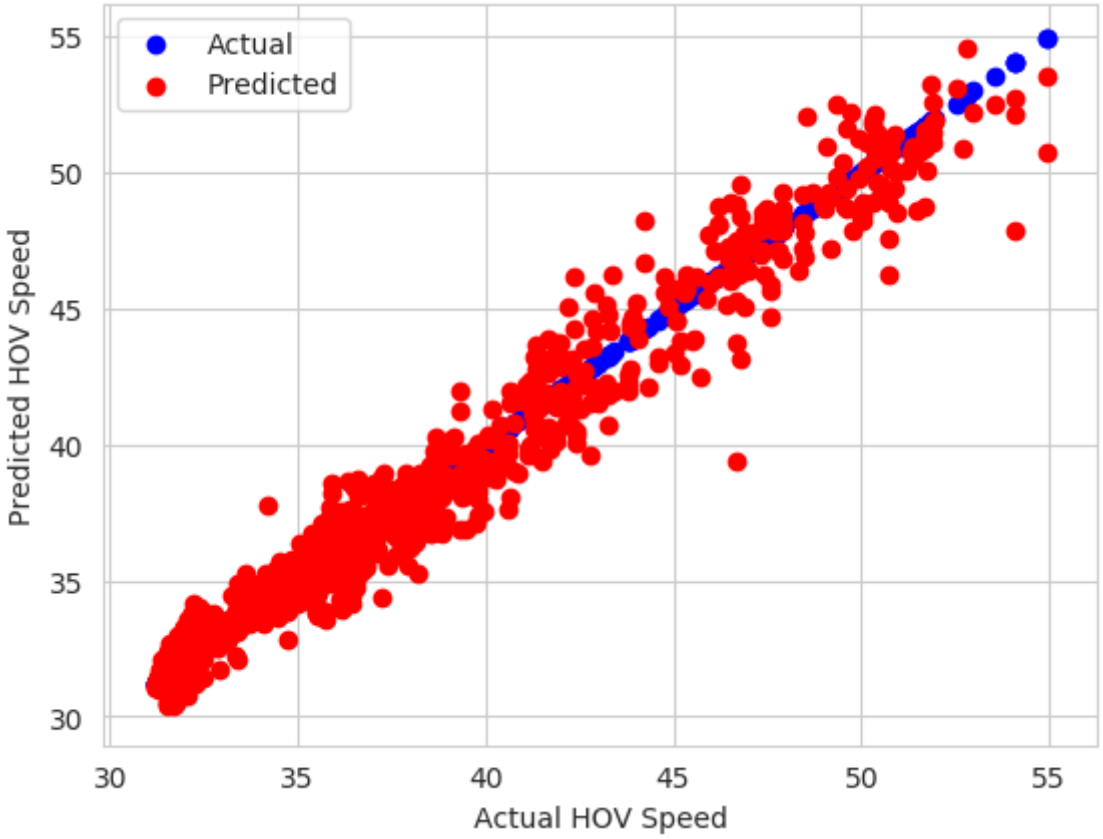
In [61]:
```
##Creating the model on Training Data
```

```
SVM=RegModel.fit(x_train,y_train)
y_pred_SVM = SVM.predict(x_test)
```

In [62]:
```
print('RMSE for SVM model is: ', math.sqrt(mean_squared_error(y_test, y_pred_SVM)))
```

```
RMSE for SVM model is:  0.8688802309925242
```

In [63]:
```
# Plot the actual values
plt.scatter(y_test, y_test, color='blue', label='Actual')

# Plot the predicted values
plt.scatter(y_test, y_pred_SVM, color='red', label='Predicted')

# Set the axis labels and legend
plt.xlabel('Actual HOV Speed')
plt.ylabel('Predicted HOV Speed')
plt.legend()

# Show the plot
plt.show()
```



In [92]:
```
from sklearn.metrics import r2_score
Accuracy_lr=r2_score(y_test,y_pred_SVM)*100
print(" Accuracy of the model is %.2f" %Accuracy_lr)
```

```
 Accuracy of the model is 97.41
```

In [64]:
```
from tensorflow import keras
```

Neural Nerwork model

In [82]:
```
# Neural Nerwork model
import numpy as np
#from tensorflow import keras
from tensorflow import keras
from tensorflow.keras.models import Sequential
```

```
from tensorflow.keras.layers import Dense
from sklearn.model_selection import train_test_split


# Define the model architecture
model = Sequential()
model.add(Dense(300, input_dim = 300))
model.add(Dense(50, input_dim = 300, activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dense(1))
# Compile the model
model.compile(loss='mean_squared_error', optimizer='adam', metrics = 'accuracy')

# Train the model
#model.fit(x_train, y_train, epochs=50, batch_size=32)

# Evaluate the model on the test set
#mse = model.evaluate(x_test, y_test)
#print("MSE:", mse)
```

In [83]:
```
x_train = pd.DataFrame(x_train)
x_train = np.array(x_train)
x_train = np.asarray(x_train).astype('float32')
```

In [84]:
```
type(y_train)
```

Out[84]:
```
numpy.ndarray
```

In [85]:
```
# Train the model
x_train = np.asarray(x_train).astype(np.float32)
y_train = np.asarray(y_train).astype(np.float32)
model.fit(x_train, y_train, epochs=50, batch_size=32)
```

```
Epoch 1/50
202/202 [==============================] - 0s 2ms/step - loss: 462.1585 - accuracy: 0.0000e+00
Epoch 2/50
202/202 [==============================] - 0s 2ms/step - loss: 4.6521 - accuracy: 0.0000e+00
Epoch 3/50
202/202 [==============================] - 0s 2ms/step - loss: 3.1202 - accuracy: 0.0000e+00
Epoch 4/50
202/202 [==============================] - 0s 2ms/step - loss: 2.8186 - accuracy: 0.0000e+00
Epoch 5/50
202/202 [==============================] - 0s 2ms/step - loss: 2.6316 - accuracy: 0.0000e+00
Epoch 6/50
202/202 [==============================] - 0s 2ms/step - loss: 2.3765 - accuracy: 0.0000e+00
Epoch 7/50
202/202 [==============================] - 0s 2ms/step - loss: 2.3017 - accuracy: 0.0000e+00
Epoch 8/50
202/202 [==============================] - 0s 2ms/step - loss: 2.1646 - accuracy: 0.0000e+00
Epoch 9/50
202/202 [==============================] - 0s 2ms/step - loss: 2.0906 - accuracy: 0.0000e+00
Epoch 10/50
202/202 [==============================] - 0s 2ms/step - loss: 1.8287 - accuracy: 0.0000e+00
Epoch 11/50
202/202 [==============================] - 0s 2ms/step - loss: 1.9565 - accuracy: 0.0000e+00
Epoch 12/50
202/202 [==============================] - 0s 2ms/step - loss: 1.8530 - accuracy: 0.0000e+00
Epoch 13/50
202/202 [==============================] - 0s 2ms/step - loss: 1.7350 - accuracy: 0.0000e+00
Epoch 14/50
202/202 [==============================] - 0s 2ms/step - loss: 1.9131 - accuracy: 0.0000e+00
Epoch 15/50
202/202 [==============================] - 0s 2ms/step - loss: 1.7579 - accuracy: 0.0000e+00
Epoch 16/50
202/202 [==============================] - 0s 2ms/step - loss: 1.6373 - accuracy: 0.0000e+00
Epoch 17/50
202/202 [==============================] - 0s 2ms/step - loss: 1.7475 - accuracy: 0.0000e+00
Epoch 18/50
202/202 [==============================] - 0s 2ms/step - loss: 1.7558 - accuracy: 0.0000e+00
Epoch 19/50
202/202 [==============================] - 0s 2ms/step - loss: 1.7463 - accuracy: 0.0000e+00
Epoch 20/50
202/202 [==============================] - 0s 2ms/step - loss: 1.6281 - accuracy: 0.0000e+00
Epoch 21/50
202/202 [==============================] - 0s 2ms/step - loss: 1.5262 - accuracy: 0.0000e+00
Epoch 22/50
202/202 [==============================] - 0s 2ms/step - loss: 1.4528 - accuracy: 0.0000e+00
Epoch 23/50
```

```
202/202 [==============================] - 0s 2ms/step - loss: 1.4330 - accuracy: 0.0000e+00
Epoch 24/50
202/202 [==============================] - 0s 2ms/step - loss: 1.5562 - accuracy: 0.0000e+00
Epoch 25/50
202/202 [==============================] - 0s 2ms/step - loss: 1.7090 - accuracy: 0.0000e+00
Epoch 26/50
202/202 [==============================] - 0s 2ms/step - loss: 1.3617 - accuracy: 0.0000e+00
Epoch 27/50
202/202 [==============================] - 0s 2ms/step - loss: 1.3349 - accuracy: 0.0000e+00
Epoch 28/50
202/202 [==============================] - 0s 2ms/step - loss: 1.2997 - accuracy: 0.0000e+00
Epoch 29/50
202/202 [==============================] - 0s 2ms/step - loss: 1.3033 - accuracy: 0.0000e+00
Epoch 30/50
202/202 [==============================] - 0s 2ms/step - loss: 1.3254 - accuracy: 0.0000e+00
Epoch 31/50
202/202 [==============================] - 0s 2ms/step - loss: 1.1196 - accuracy: 0.0000e+00
Epoch 32/50
202/202 [==============================] - 0s 2ms/step - loss: 1.3479 - accuracy: 0.0000e+00
Epoch 33/50
202/202 [==============================] - 0s 2ms/step - loss: 1.1987 - accuracy: 0.0000e+00
Epoch 34/50
202/202 [==============================] - 0s 2ms/step - loss: 1.0543 - accuracy: 0.0000e+00
Epoch 35/50
202/202 [==============================] - 0s 2ms/step - loss: 1.2121 - accuracy: 0.0000e+00
Epoch 36/50
202/202 [==============================] - 0s 2ms/step - loss: 1.1628 - accuracy: 0.0000e+00
Epoch 37/50
202/202 [==============================] - 0s 2ms/step - loss: 1.0927 - accuracy: 0.0000e+00
Epoch 38/50
202/202 [==============================] - 0s 2ms/step - loss: 0.9654 - accuracy: 0.0000e+00
Epoch 39/50
202/202 [==============================] - 0s 2ms/step - loss: 0.9495 - accuracy: 0.0000e+00
Epoch 40/50
202/202 [==============================] - 0s 2ms/step - loss: 1.1171 - accuracy: 0.0000e+00
Epoch 41/50
202/202 [==============================] - 0s 2ms/step - loss: 0.8583 - accuracy: 0.0000e+00
Epoch 42/50
202/202 [==============================] - 0s 2ms/step - loss: 0.9600 - accuracy: 0.0000e+00
Epoch 43/50
202/202 [==============================] - 0s 2ms/step - loss: 0.9806 - accuracy: 0.0000e+00
Epoch 44/50
202/202 [==============================] - 0s 2ms/step - loss: 0.9992 - accuracy: 0.0000e+00
Epoch 45/50
202/202 [==============================] - 0s 2ms/step - loss: 0.9341 - accuracy: 0.0000e+00
```

```
Epoch 46/50
202/202 [==============================] - 0s 2ms/step - loss: 0.9449 - accuracy: 0.0000e+00
Epoch 47/50
202/202 [==============================] - 0s 2ms/step - loss: 0.8162 - accuracy: 0.0000e+00
Epoch 48/50
202/202 [==============================] - 0s 2ms/step - loss: 0.8919 - accuracy: 0.0000e+00
Epoch 49/50
202/202 [==============================] - 0s 2ms/step - loss: 0.8606 - accuracy: 0.0000e+00
Epoch 50/50
202/202 [==============================] - 0s 2ms/step - loss: 0.8000 - accuracy: 0.0000e+00
```

Out[85]:
```
<tensorflow.python.keras.callbacks.History at 0x7ff19076af90>
```

In [86]:
```
# Evaluate the model on the test set
x_test = np.asarray(x_test).astype(np.float32)
y_test = np.asarray(y_test).astype(np.float32)
a,b = model.evaluate(x_test, y_test)

print("RMSE for the NN model",math.sqrt(a))
```

```
51/51 [==============================] - 0s 1ms/step - loss: 0.9472 - accuracy: 0.0000e+00
RMSE for the NN model 0.9732396471211264
```

In [87]:
```
y_pred_NN = model.predict(x_test)
print('RMSE for NN model is: ', math.sqrt(mean_squared_error(y_test, y_pred_NN)))
```

```
RMSE for NN model is:  0.9732396471211264
```

In [93]:
```
from sklearn.metrics import r2_score
Accuracy_lr=r2_score(y_test,y_pred_NN)*100
print(" Accuracy of the model is %.2f" %Accuracy_lr)
```

```
 Accuracy of the model is 96.75
```

XGBoost Model

In [88]:
```
# evaluate an xgboost regression model
import sys

!{sys.executable} -m pip install xgboost

from numpy import absolute
from pandas import read_csv
from sklearn.model_selection import cross_val_score
```

```
from sklearn.model_selection import RepeatedKFold
from xgboost import XGBRegressor
#load the dataset
#model = XGBRegressor()
# define model evaluation method
#cv = RepeatedKFold(n_splits=10, n_repeats=3, random_state=1)
# evaluate model
#scores = cross_val_score(model, x_train, y_train, scoring='neg_mean_absolute_error', cv=cv, n_jobs=-1)
# force scores to be positive
#scores = absolute(scores)
#print('Mean MAE: %.3f (%.3f)' % (scores.mean(), scores.std()) )
```

```
Requirement already satisfied: xgboost in /opt/conda/lib/python3.7/site-packages (1.6.2)
Requirement already satisfied: numpy in /opt/conda/lib/python3.7/site-packages (from xgboost) (1.18.1)
Requirement already satisfied: scipy in /opt/conda/lib/python3.7/site-packages (from xgboost) (1.4.1)
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
```

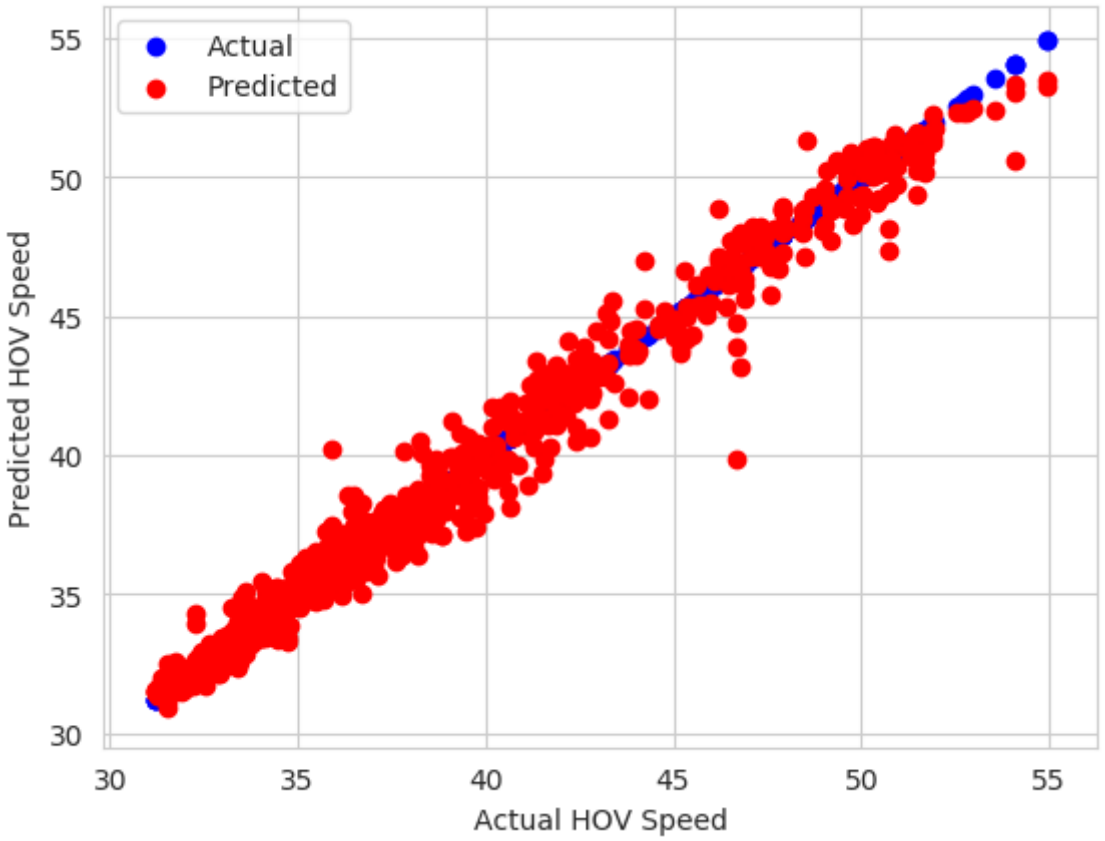
In [89]:
```
#load the dataset
model = XGBRegressor()
# define model evaluation method
cv = RepeatedKFold(n_splits=10, n_repeats=3, random_state=1)

model.fit(x_train,y_train)
y_pred_XGB = model.predict(x_test)
# evaluate model
# scores = cross_val_score(model, x_train, y_train, scoring='mean_squared_error', cv=cv, n_jobs=-1)
```

In [90]:
```
print('RMSE for XGB model is: ', math.sqrt(mean_squared_error(y_test, y_pred_XGB)))
```

```
RMSE for XGB model is:  0.6188560818758817
```

In [91]:
```
# Plot the actual values
plt.scatter(y_test, y_test, color='blue', label='Actual')

# Plot the predicted values
plt.scatter(y_test, y_pred_XGB, color='red', label='Predicted')

# Set the axis labels and legend
plt.xlabel('Actual HOV Speed')
plt.ylabel('Predicted HOV Speed')
plt.legend()
```

```
# Show the plot
plt.show()
```



In [94]:
```
from sklearn.metrics import r2_score
Accuracy_lr=r2_score(y_test,y_pred_XGB)*100
print(" Accuracy of the model is %.2f" %Accuracy_lr)
```

```
 Accuracy of the model is 98.69
```

In [ ]:

In [ ]: