PROJE KURULUM AŞAMALARI

1. PostgreSQL 14/ pgAdmin4

Getir adında sıfırdan bir database kurulumu yapıldı. Bu databasenin kurulması çok önemli çünkü gerek pgAdmin4 gerek Python üzeridnen sqlalchemy modülü aracılığıyla databaseden veri sorgulamaları veya doğrudan databasenin içerisine tablo aktarılması gibi işlemler çoğu python\_related klasörü içerisindeki scriptlerde mevcut.

1. Python/Anaconda

Proje boyunce ihtiyacınız olacak tüm modüllerin bir çıktısını requirements.txt adlı dosyada bulabilirsiniz. İçerisinde bir tek Spyder-Kernels ve bu modülün gereksinimleri fazlalıktır. Python 3.12.11 sürümü kullanılarak hazırlanmıştır ve modüller arası bir çakışmaya rastlanmamıştır. Sıfırdan bu proje için oluşturulan virtual environment Anaconda ile kurulmuş olup Conda 23.9.0 sürümü kullanılmıştır.

1. QGIS

Proje boyunca QGIS 3.30.0-'s-Hertogenbosch sürümü kullanılmıştır. Çalışma boyunca yoğun saatlere karşılık hiçbir çökme ile karşılaşılmamıştır. Ekstra bir plugin gereksinimi yoktur. PostgreSQL getir veritabanının Data Source Manager aracılığı ile QGIS’e kaydedilmesi gerekiyor. İlerleyen aşamalarda sıkça kullanılmasının yanı sıra projenin kapatılıp yeniden sorunsuz açılması için önemli.

1. SQL scriptleri

Getir/SQL related klasöründen bu dosyalara ulaşılabilir. İlk aşamada “/getir/python\_related/ push.py“ dosyası çalıştırılarak hem siparis hem de warehose csv dosyaları postgreSQL sistemine aktarılır. Aktarımın ardından “/getir/sql\_related/getir\_create\_geom.sql” scripti adım adım takip edilir. /getir/python\_related/getir\_data.py çalıştırılır. Hali hazırda dosya içerisinde bulacağınız bursa\_district.geojson dosyasini OSM kullanarak kendiniz üretecekseniz Multipolygon tipine dönüştürmeniz zorunlu bunun için scriptin en altındaki işlem uygulanıyor fakat bunun öncesinde OSM üzerinden elde ettğiniz geojson dosyasının “@relations” sütununu kendiniz silmeniz gerekmektedir. Geopandas veya QGIS kullanılarak bu sorun aşılır. Sorunsuz ilerlenilirse Geojson çıktısı PostgreSQL veritabanına aktarılır. Tamamlanmasını ardından “/sql\_related/pygetir\_dist.sql” scriptinin takip edilmesi gerekiyor. ./getir\_query.sql dosyası veriler üretilirken yapılan sorguları içeren dosyadır. Yalnızca sorgu dosyası olmakla beraber kullanılmaması bir aksaklık yaratmaz.

1. Karışık
   1. Bu aşamada işlemleri belli bir başlık altında toplaması zor çünkü hepsi bir arada kullanılmaya başlanacak.” /python\_related/getir\_service\_ors.py” scripti çalıştırılır. Heigit üzerinden sonraki aşamalarda rota hesaplanması için gerekli OpenRouteService API keyinın veri üretmeyi planlayacak kişiler için zorunlu olduğunu belirteyim. 24 saatte bir sorgu limitleriniz sorgu çeşitlerine göre yenilenmektedir. Tüm projenin oluşturulmasında herhangi bir limit aşımı söz konusu olmamıştır. Script ile üretilen yalnızca geojson dosyasının PostgreSQL’e aktarımı yapılmalıdır, diğer tüm dosyalar otomatik olarak içeri aktarılmaktadır. Proje QGIS üzerinden hiçbir csv açma zorunluluğu içermez, tek seferlik **elde** **edilen geojson formatındaki çıktılar PostgreSQL’e aktarıldıkları koşuluyla**.
   2. Ayni klasör içerisinden “getir\_service\_ors\_bicyle.py” çalıştırılarak aynı işlem bisiklet senaryosuna göre gerçekleştirilir, geojson çıktısının PostgreSQL’e aktarılması gerekir.
   3. “ors.compare.py” ile iki farklı araç senaryosu içerisinden zaman ve mesafeye göre hangisinin avantajlı olduğunu gösteren tabloyu oluşturur. Script sonunda oluşturulan csv opsiyoneldir, kullanımı zorunlu değil.
   4. “pipe.py” Kendi sql sorgularınızı çalıştırıp dataframeye dönüştürebileceğinz bir script, tamamen opsiyonel.
   5. “shortest.py” iki koordinat çifti arasıdaki mesafe ve süresinin hesaplayabileceğinz basir bir script, tamamen opsiyonel.
   6. “service\_point\_needed”, “getir\_service\_ors.py” çıktısı olan “sip\_war\_with\_routes.csv” dosyasındaki noktaları kullanarak istenilen servis noktasının sayısına göre Kmeans kullanarak işlem sonucunda geojson çıktısı veren bir script.
   7. “service\_point\_isochrone[median,min,half\_median].py” scripleri ORS kullanılarak isochrome haritaları oluşturmak için kullanılıyor. Farklı scriptlerin oluşturulmasındaki amaç “delivery\_duration” yani teslimat süresinden elde edilen farklı istatistiksel verilere göre farklı haritaların elde edilmesi. Hepsi veya tercih etmek istediğiniz arasından bir tanesi ile çalışılabilir hatta hiç kullanılmayada bilir.
   8. “siparis\_bursa\_valid.py” PostgreSQL’de bulunan “siparis” tablosu ile “bursa\_mahalle\_valid” tablolarını birleştirip, geojson formatında kaydediyor. Tamamen opsiyonel ilerleyen aşamalara bir katkısı yoktur.
   9. “neighbourhood\_order\_count.py” Mahalle bazlı toplam sipariş sayılarını QGIS üzerinden gösterme amaçlı iki farklı PostgreSQL sorgusunu birleştirip, “order\_counts\_by\_hood.geojson” çıktısını üreten script.
   10. “district\_order\_count.py” İlçe bazlı sipariş sayısının oluşturulması için postgreSQL sorgusu çalıştırıp, çıktıyı “district\_order\_count.geojson” olarak kaydeder.
   11. “gen\_df\_then\_to \_postgre” mahalle bazlı nüfus bilgisi bu proje içerisinde edinilmesi en güç veri oldu, yalnızca veriye ulaşım değil güncel veriye ulaştıktan sonrada standart dışı oluşturulmuş csv dosyalarını düzenleyerek, düzenleme sonrası mahalle adlarında yaşanabilecek ufak değişiklikleri yakalayarak doğru mahallelere atanmasını sağlayan PostgreSQL sorgusunu çalıştırır. Sorgu sonrası “mahalle\_pop\_matched.geojson” dosyası ilişkili mahelle nüfus değerlerini ve mahalle bölgelerini içerisinde barındırır.