**Real Time Traffic Project  
Under Super Vision: DR. Amira Yousef**

**Ahmed Salama Soliman Hasan**

**Mohamed Nasr**

**Osama Ahmed Reda Hegazy**

**Sara Hisham Ahmed Mohamed**

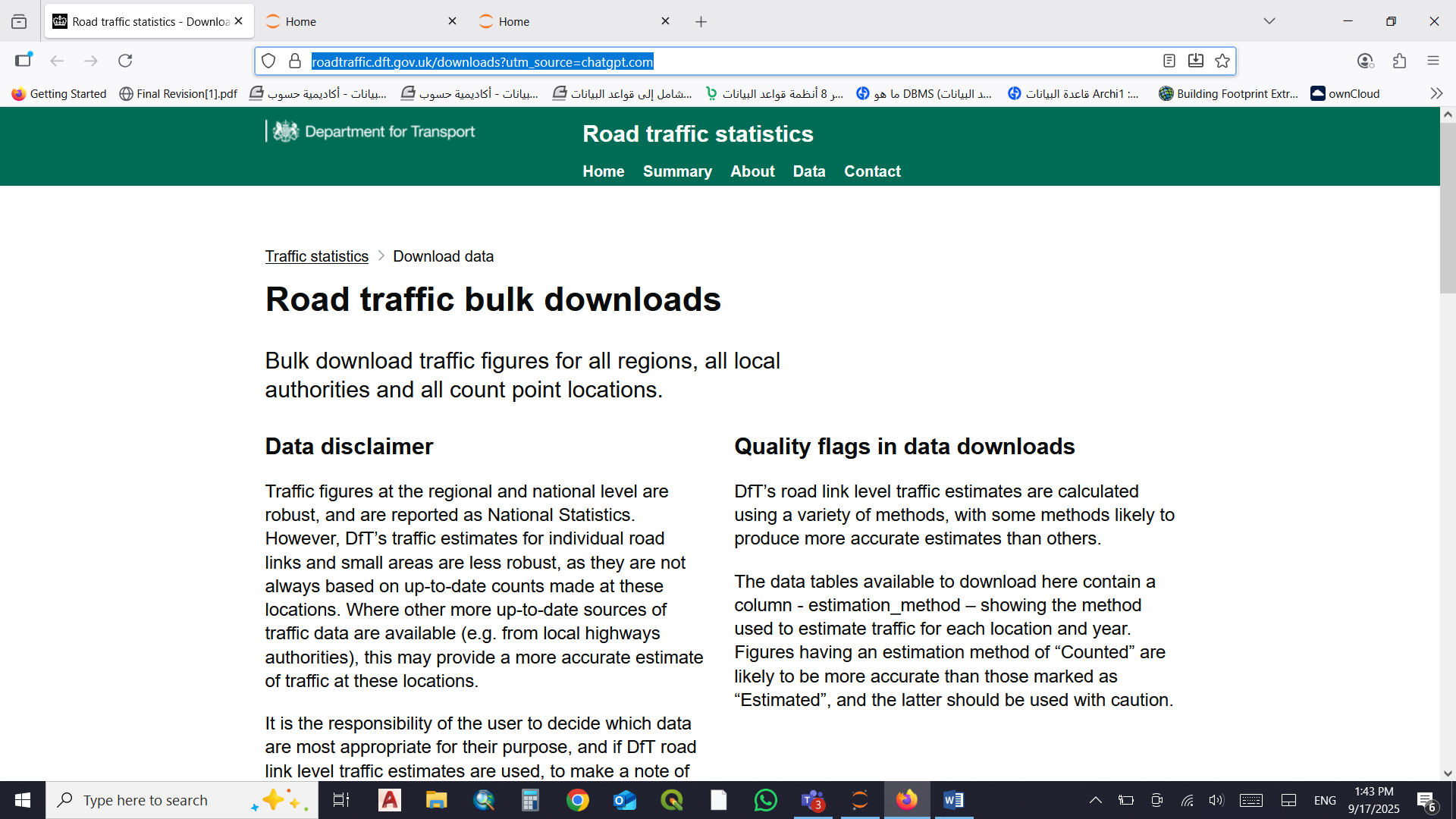
**Sherif Gamal Kamed AbdElwahed**

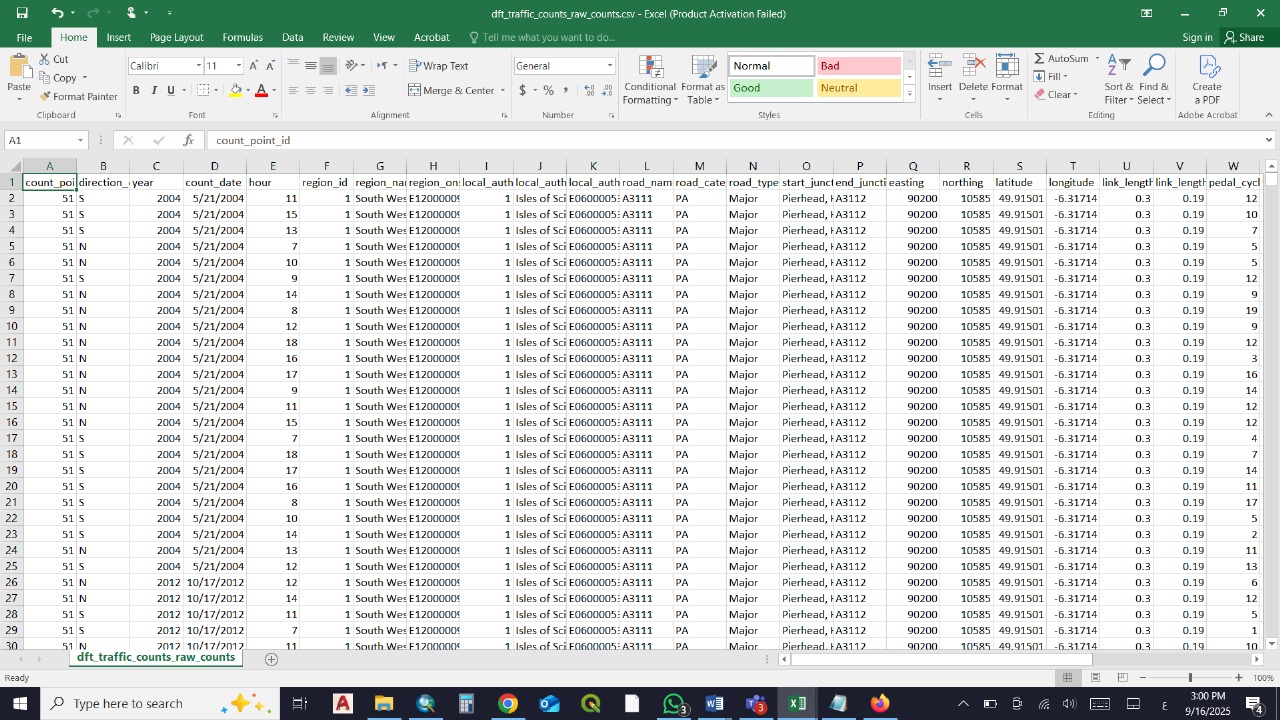
**Yousef Ahmed Mohamed Ibrahiem**

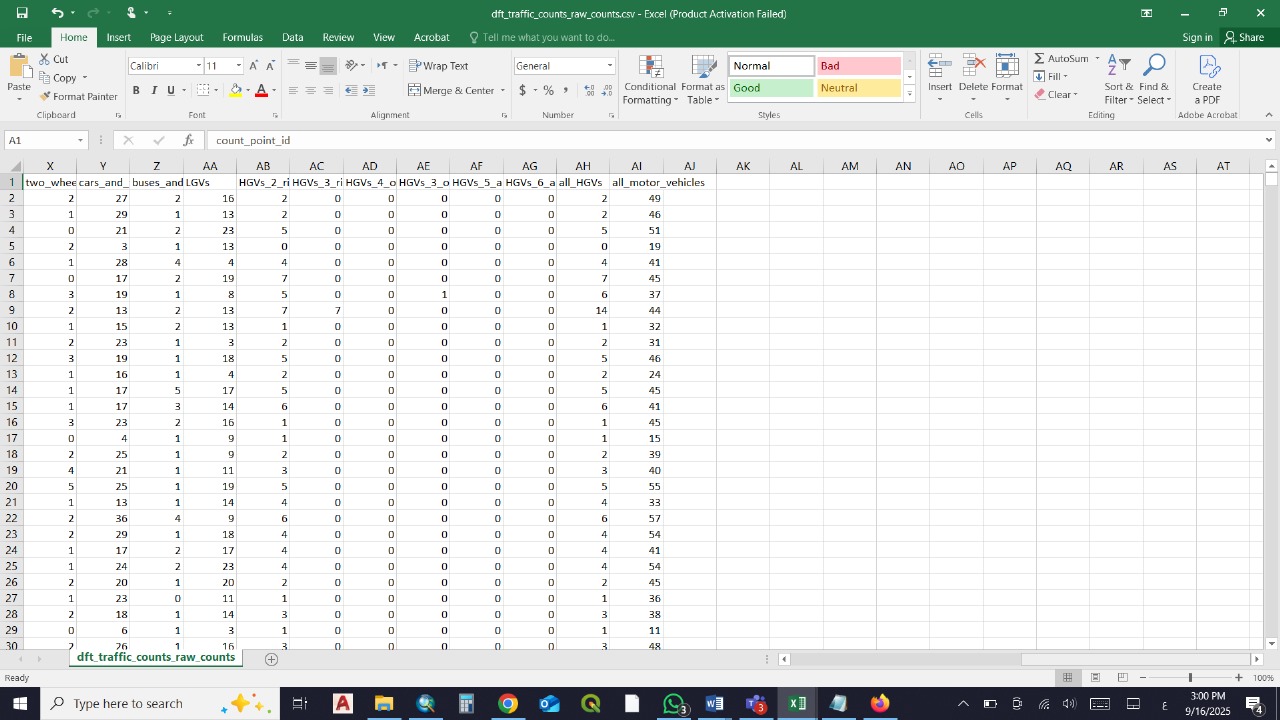
**Zakaria Yehia Ahmed**

**Data Source**

<https://roadtraffic.dft.gov.uk/downloads?utm_source=chatgpt.com>

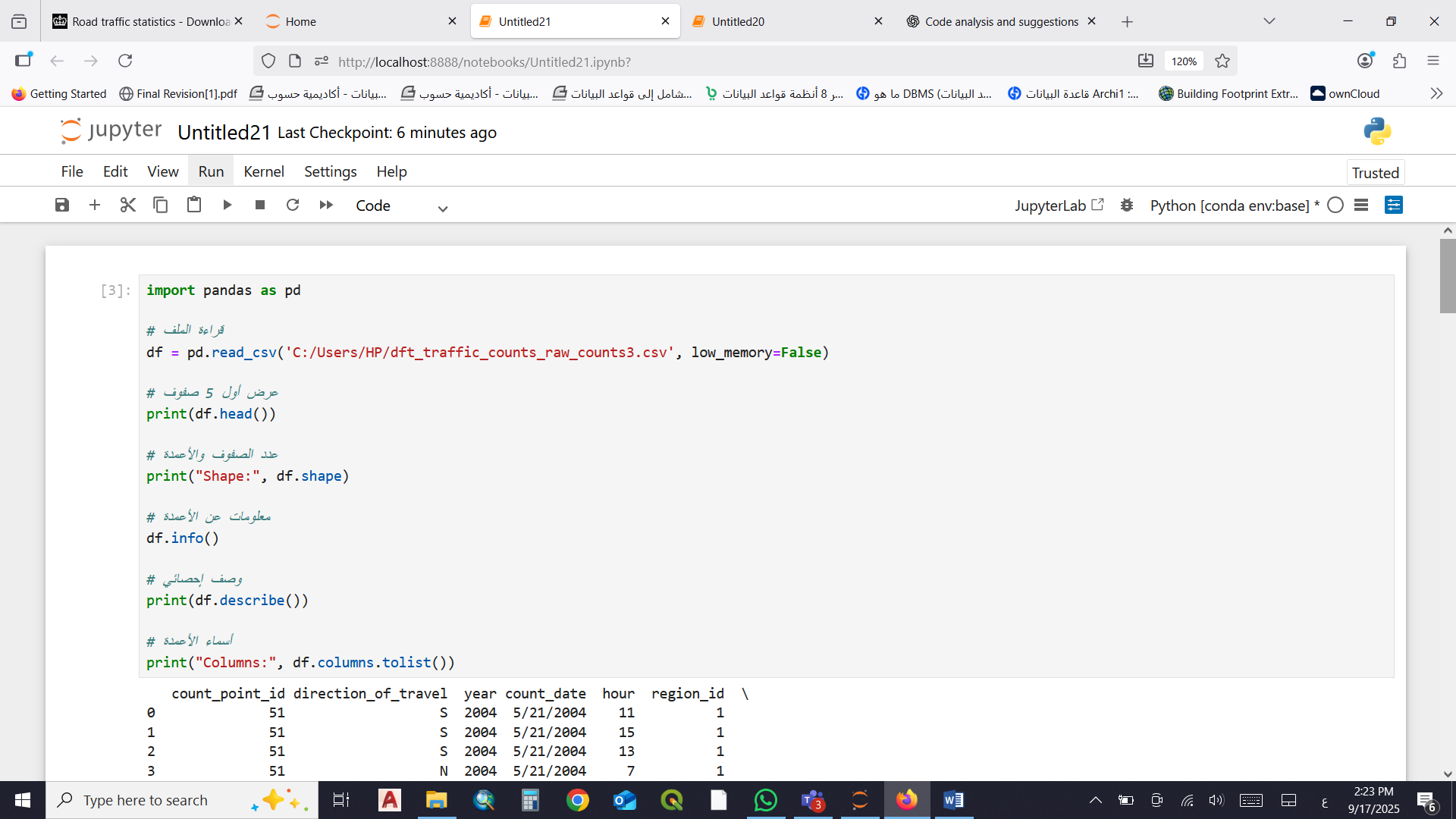


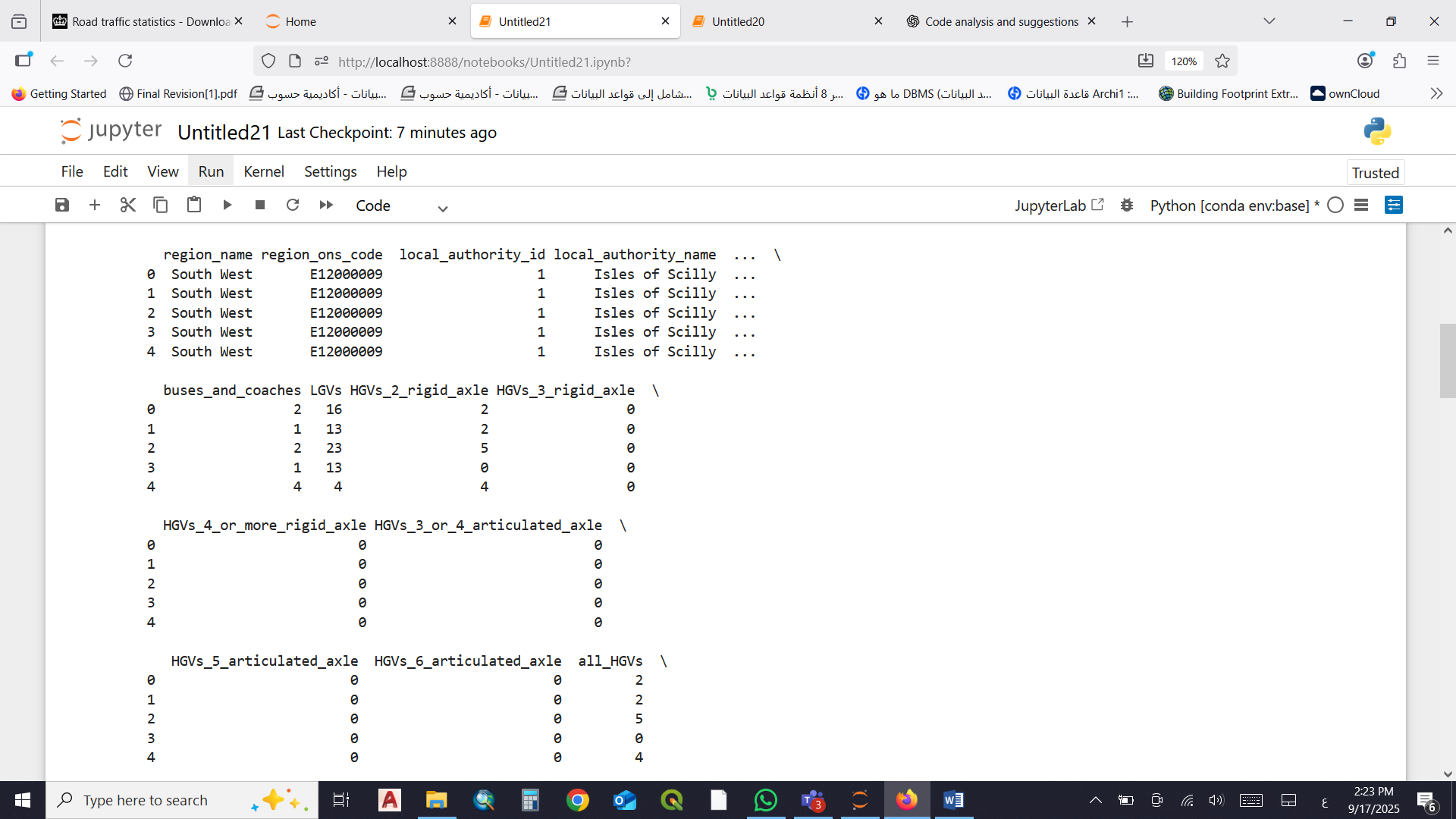


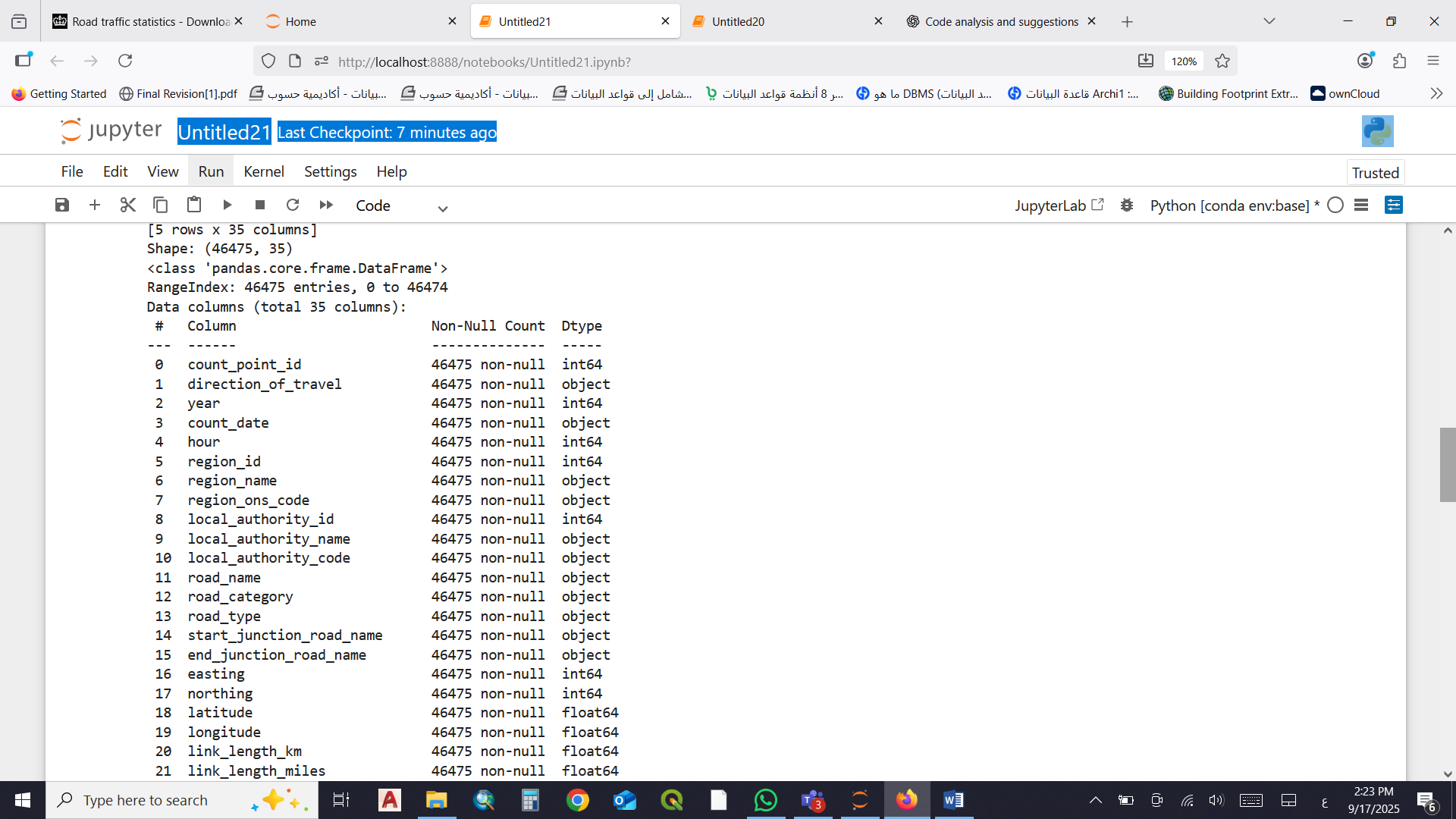


**شرح معاني الحقول (Columns):**

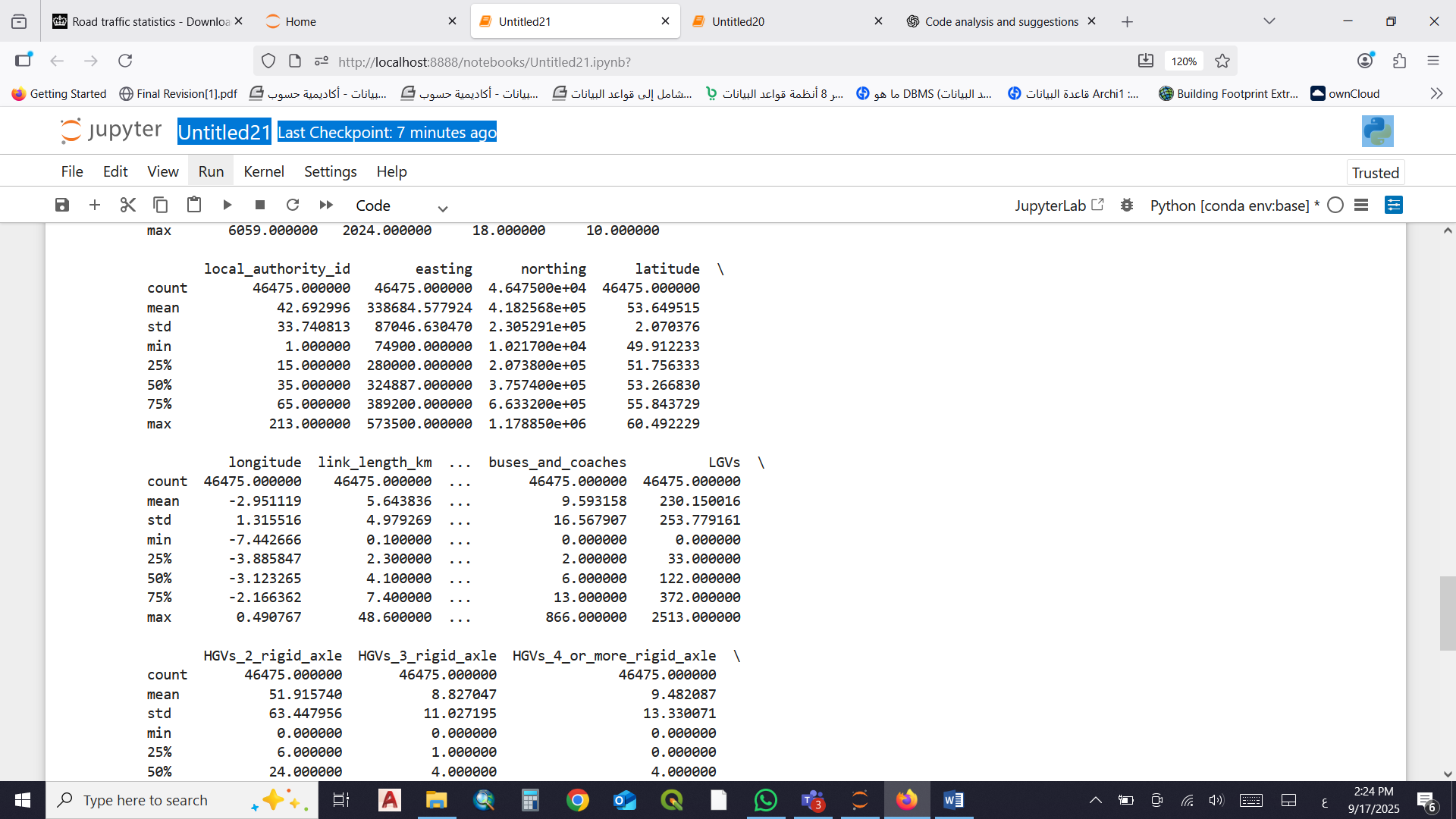
| اسم الحقل | المعنى |
| --- | --- |
| count\_point\_id | رقم تعريف نقطة العدّ (مكان تم فيه عدّ المركبات). |
| direction\_of\_travel | اتجاه السفر (زي شمال، جنوب، شرق، غرب). |
| year | سنة تسجيل البيانات. |
| count\_date | التاريخ الفعلي للعدّ (يوم العدّ). |
| hour | الساعة (الوقت اللي تم فيه العدّ – غالبًا من 0 إلى 23). |
| region\_id | رقم تعريف المنطقة (رقم داخلي يستخدم لتحديد المنطقة). |
| region\_name | اسم المنطقة (زي "East Midlands" أو "London"). |
| region\_ons\_code | كود المنطقة حسب مكتب الإحصاء الوطني ONS (للاستخدام الإحصائي). |
| local\_authority\_id | رقم تعريف السلطة المحلية (بلدية أو مجلس محلي). |
| local\_authority\_name | اسم السلطة المحلية (زي “Leeds City Council”). |
| local\_authority\_code | كود السلطة المحلية (رمز مختصر). |
| road\_name | اسم الطريق (زي A1، M25). |
| road\_category | فئة الطريق (زي طريق سريع، طريق رئيسي، محلي، الخ). |
| road\_type | نوع الطريق (قد يكون مزيد من التفصيل عن الفئة – مفرد، مزدوج، الخ). |
| start\_junction\_road\_name | اسم الطريق عند بداية المفرق أو التقاطع. |
| end\_junction\_road\_name | اسم الطريق عند نهاية المفرق أو التقاطع. |
| easting | الإحداثي الشرقي (نظام الإحداثيات البريطانية). |
| northing | الإحداثي الشمالي (نظام الإحداثيات البريطانية). |
| latitude | دائرة العرض (إحداثيات جغرافية). |
| longitude | خط الطول (إحداثيات جغرافية). |
| link\_length\_km | طول المقطع الطرقي بالكيلومترات. |
| link\_length\_miles | طول المقطع الطرقي بالأميال. |
| pedal\_cycles | عدد الدراجات الهوائية. |
| two\_wheeled\_motor\_vehicles | عدد الدراجات النارية (سكوتر، موتوسيكل). |
| cars\_and\_taxis | عدد السيارات العادية والتاكسي. |
| buses\_and\_coaches | عدد الحافلات. |
| LGVs | سيارات النقل الخفيف (Light Goods Vehicles) – زي الفانات الصغيرة. |
| HGVs\_2\_rigid\_axle | شاحنات ثقيلة بعدد 2 محور صلب. |
| HGVs\_3\_rigid\_axle | شاحنات ثقيلة بعدد 3 محاور صلبة. |
| HGVs\_4\_or\_more\_rigid\_axle | شاحنات ثقيلة بعدد 4 أو أكثر من المحاور الصلبة. |
| HGVs\_3\_or\_4\_articulated\_axle | شاحنات ثقيلة مفصلية بعدد 3 أو 4 محاور. |
| HGVs\_5\_articulated\_axle | شاحنات ثقيلة مفصلية بعدد 5 محاور. |
| HGVs\_6\_articulated\_axle | شاحنات ثقيلة مفصلية بعدد 6 محاور أو أكثر. |
| all\_HGVs | إجمالي الشاحنات الثقيلة (HGVs). |
| all\_motor\_vehicles | إجمالي جميع المركبات الآلية (من موتوسيكلات لحد الشاحنات). |

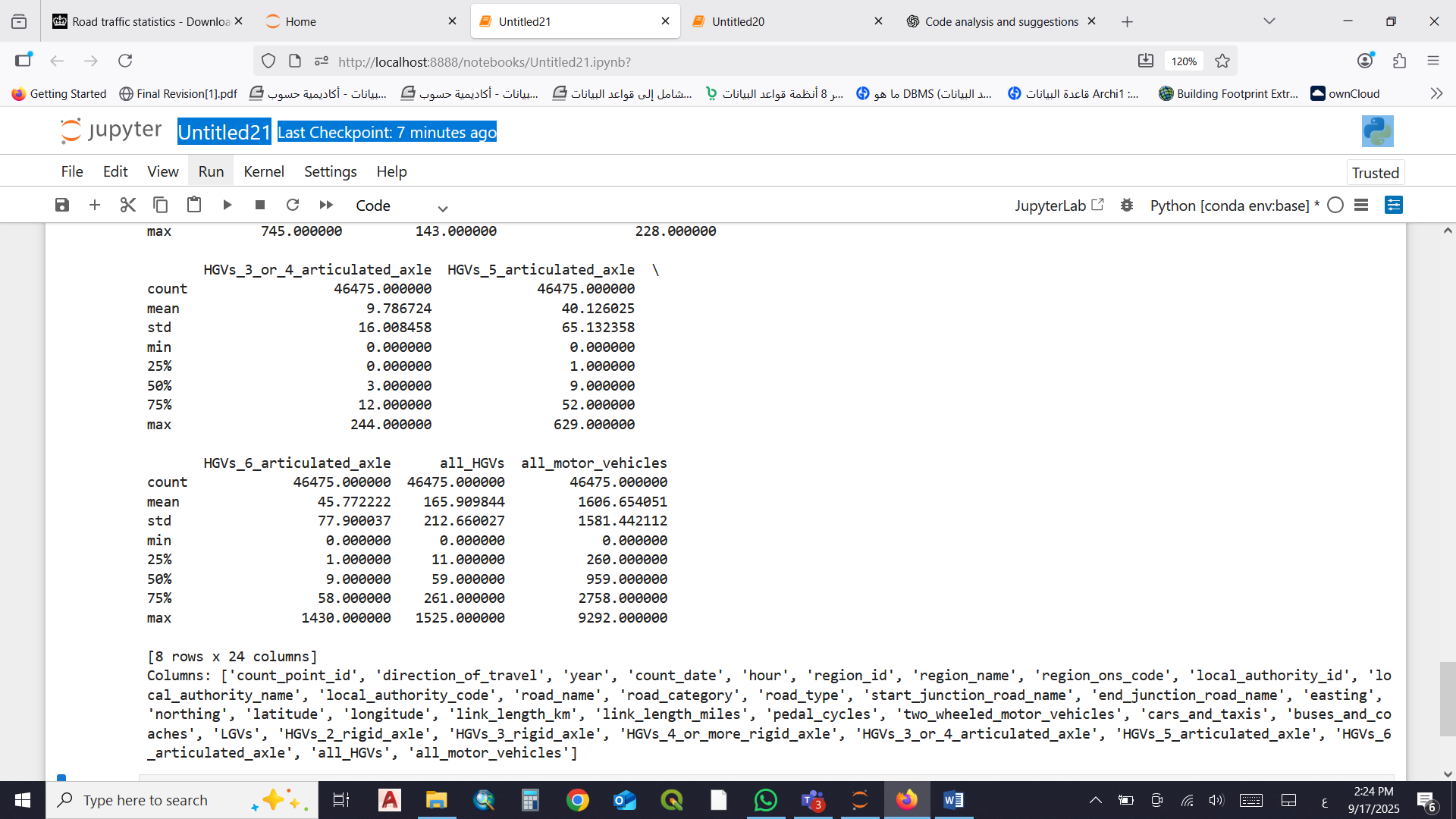












**Code for cleaning data:**

* **قراءة البيانات وتنظيفها**
* **تحويل الأعمدة الزمنية**
* **التحقق من القيم الشاذة**
* **هندسة خصائص جديدة (feature engineering)**
* **تصدير البيانات إلى قاعدة بيانات SQL**
* **إنشاء GeoDataFrame وتصدير البيانات إلى GeoJSON وShapefile**
* **تقرير جودة البيانات**

# ----------------------------------------

# 1. Initial Setup and Data Loading

# ----------------------------------------

import pandas as pd

import numpy as np

import geopandas as gpd

from sqlalchemy import create\_engine

from sqlalchemy.engine import URL

# Load the CSV data

df = pd.read\_csv('C:/Users/HP/dft\_traffic\_counts\_raw\_counts3.csv', low\_memory=False)

print(f"Dataset shape: {df.shape}")

print(df.info())

Dataset shape: (46475, 35)

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 46475 entries, 0 to 46474

Data columns (total 35 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 count\_point\_id 46475 non-null int64

1 direction\_of\_travel 46475 non-null object

2 year 46475 non-null int64

3 count\_date 46475 non-null object

4 hour 46475 non-null int64

5 region\_id 46475 non-null int64

6 region\_name 46475 non-null object

7 region\_ons\_code 46475 non-null object

8 local\_authority\_id 46475 non-null int64

9 local\_authority\_name 46475 non-null object

10 local\_authority\_code 46475 non-null object

11 road\_name 46475 non-null object

12 road\_category 46475 non-null object

13 road\_type 46475 non-null object

14 start\_junction\_road\_name 46475 non-null object

15 end\_junction\_road\_name 46475 non-null object

16 easting 46475 non-null int64

17 northing 46475 non-null int64

18 latitude 46475 non-null float64

19 longitude 46475 non-null float64

20 link\_length\_km 46475 non-null float64

21 link\_length\_miles 46475 non-null float64

22 pedal\_cycles 46475 non-null int64

23 two\_wheeled\_motor\_vehicles 46475 non-null int64

24 cars\_and\_taxis 46475 non-null int64

25 buses\_and\_coaches 46475 non-null int64

26 LGVs 46475 non-null int64

27 HGVs\_2\_rigid\_axle 46475 non-null int64

28 HGVs\_3\_rigid\_axle 46475 non-null int64

29 HGVs\_4\_or\_more\_rigid\_axle 46475 non-null int64

30 HGVs\_3\_or\_4\_articulated\_axle 46475 non-null int64

31 HGVs\_5\_articulated\_axle 46475 non-null int64

32 HGVs\_6\_articulated\_axle 46475 non-null int64

33 all\_HGVs 46475 non-null int64

34 all\_motor\_vehicles 46475 non-null int64

dtypes: float64(4), int64(20), object(11)

memory usage: 12.4+ MB

None

# ----------------------------------------

# 2. Data Exploration and Cleaning

# ----------------------------------------

# Check for missing values

missing\_values = df.isnull().sum()

print("\nMissing values per column:")

print(missing\_values[missing\_values > 0])

Missing values per column:

Series([], dtype: int64)

# Handle missing values

df.fillna({

'link\_length\_km': 0,

'link\_length\_miles': 0,

'pedal\_cycles': 0,

}, inplace=True)

Missing values per column:

Series([], dtype: int64)

# Convert count\_date to datetime (auto format)

df['count\_date'] = pd.to\_datetime(df['count\_date'], errors='coerce', dayfirst=False)

# Check data types

print("\nData types after conversion:")

print(df.dtypes)

Data types after conversion:

count\_point\_id int64

direction\_of\_travel object

year int64

count\_date datetime64[ns]

hour int64

region\_id int64

region\_name object

region\_ons\_code object

local\_authority\_id int64

local\_authority\_name object

local\_authority\_code object

road\_name object

road\_category object

road\_type object

start\_junction\_road\_name object

end\_junction\_road\_name object

easting int64

northing int64

latitude float64

longitude float64

link\_length\_km float64

link\_length\_miles float64

pedal\_cycles int64

two\_wheeled\_motor\_vehicles int64

cars\_and\_taxis int64

buses\_and\_coaches int64

LGVs int64

HGVs\_2\_rigid\_axle int64

HGVs\_3\_rigid\_axle int64

HGVs\_4\_or\_more\_rigid\_axle int64

HGVs\_3\_or\_4\_articulated\_axle int64

HGVs\_5\_articulated\_axle int64

HGVs\_6\_articulated\_axle int64

all\_HGVs int64

all\_motor\_vehicles int64

dtype: object

# ----------------------------------------

# 3. Data Validation and Quality Checks

# ----------------------------------------

numeric\_columns = ['hour', 'link\_length\_km', 'pedal\_cycles', 'cars\_and\_taxis', 'all\_motor\_vehicles']

for col in numeric\_columns:

if col in df.columns:

print(f"{col} range: {df[col].min()} - {df[col].max()}")

print("\nRoad categories:", df['road\_category'].unique())

print("Road types:", df['road\_type'].unique())

valid\_coords = (df['latitude'].between(-90, 90)) & (df['longitude'].between(-180, 180))

print(f"Invalid coordinates: {len(df[~valid\_coords])}")

hour range: 7 - 18

link\_length\_km range: 0.1 - 48.6

pedal\_cycles range: 0 - 219

cars\_and\_taxis range: 0 - 7642

all\_motor\_vehicles range: 0 - 9292

Road categories: ['PA' 'TM' 'TA']

Road types: ['Major']

Invalid coordinates: 0

# ----------------------------------------

# 4. Data Transformation and Feature Engineering

# ----------------------------------------

df['total\_vehicles'] = (

df['pedal\_cycles'] +

df['two\_wheeled\_motor\_vehicles'] +

df['cars\_and\_taxis'] +

df['buses\_and\_coaches'] +

df['LGVs'] +

df['all\_HGVs']

)

df['vehicle\_density'] = df['total\_vehicles'] / df['link\_length\_km'].replace(0, 0.001)

df['day\_of\_week'] = df['count\_date'].dt.day\_name()

df['month'] = df['count\_date'].dt.month

df['season'] = df['month'].apply(lambda x: 'Winter' if x in [12, 1, 2] else

'Spring' if x in [3, 4, 5] else

'Summer' if x in [6, 7, 8] else

'Autumn')

df['hgv\_percentage'] = df['all\_HGVs'] / df['total\_vehicles'].replace(0, 1)

df['bicycle\_percentage'] = df['pedal\_cycles'] / df['total\_vehicles'].replace(0, 1)

# ----------------------------------------

# 5. Save to Parquet (optional)

# ----------------------------------------

df.to\_parquet('traffic\_data\_processed.parquet', engine='pyarrow')

# ----------------------------------------

from sqlalchemy import create\_engine

from sqlalchemy.engine import URL

connection\_url = URL.create(

"mssql+pyodbc",

host="localhost",

database="traffic\_db",

query={

"driver": "ODBC Driver 17 for SQL Server",

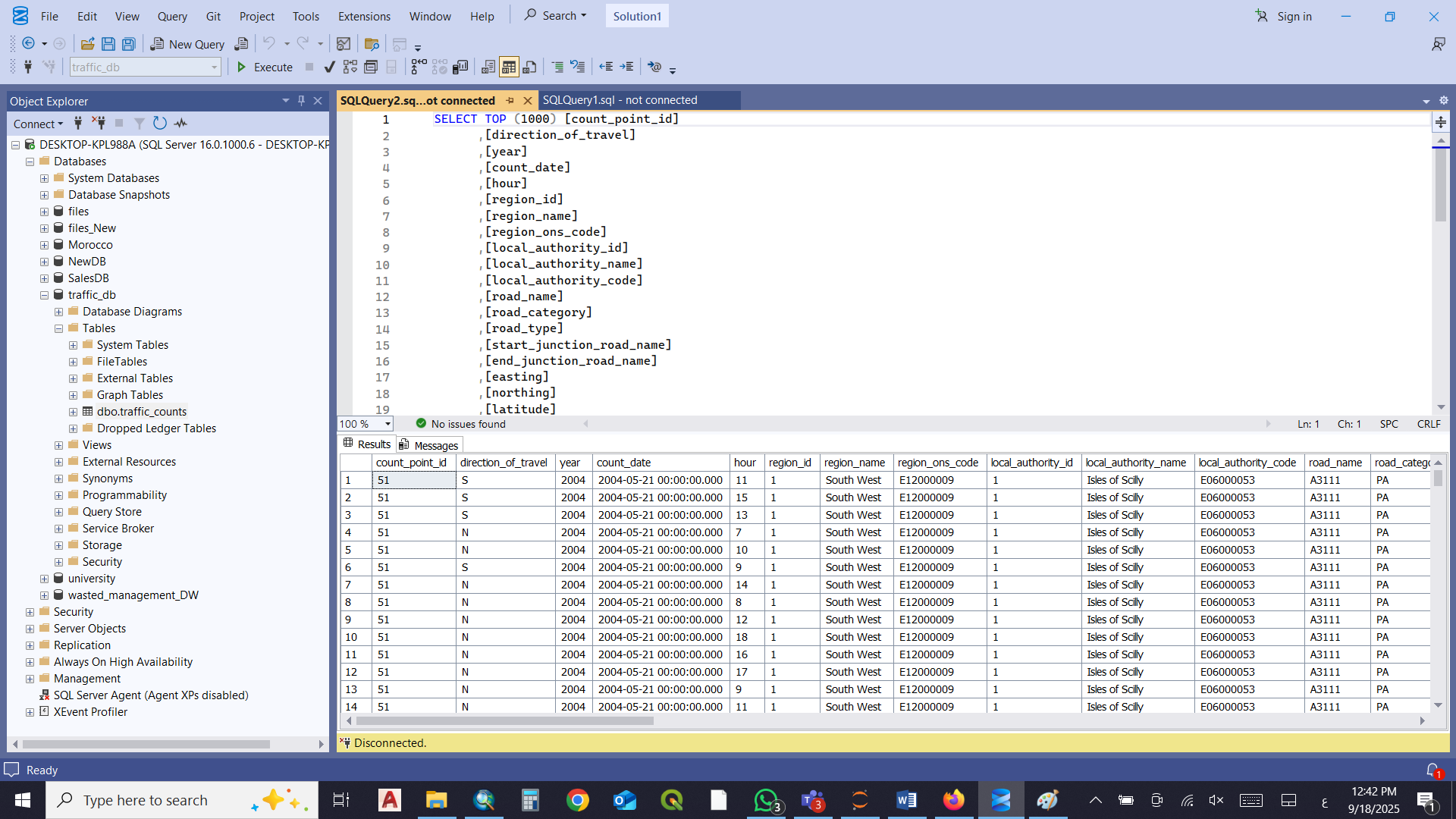
"Trusted\_Connection": "yes"

}

)

engine = create\_engine(connection\_url)

df.to\_sql('traffic\_counts', engine, if\_exists='replace', index=False)



# ----------------------------------------

# 7. Aggregate Data for Reporting

# ----------------------------------------

daily\_totals = df.groupby(['count\_date', 'region\_name', 'local\_authority\_name']).agg({

'total\_vehicles': 'sum',

'pedal\_cycles': 'sum',

'all\_HGVs': 'sum'

}).reset\_index()

count\_date region\_name local\_authority\_name total\_vehicles \

0 2000-03-17 Scotland East Lothian 13320

1 2000-03-17 Wales Cardiff 17916

2 2000-03-21 North West Warrington 72633

3 2000-03-22 East Midlands Northamptonshire 9259

4 2000-03-22 Scotland North Lanarkshire 7475

... ... ... ... ...

1936 2024-09-18 Scotland Shetland Islands 3295

1937 2024-09-18 Wales Gwynedd 10587

1938 2024-09-26 North West Stockport 30383

1939 2024-10-01 Wales Denbighshire 32774

1940 2024-10-08 North West Cumberland 38656

pedal\_cycles all\_HGVs

0 62 418

1 35 526

2 0 11003

3 1 1179

4 17 439

... ... ...

1936 13 43

1937 8 239

1938 143 627

1939 0 2998

1940 1 7344

[1941 rows x 6 columns]

# ----------------------------------------

# 8. Save GeoJSON (Optional for Spatial Analysis)

# ----------------------------------------

gdf = gpd.GeoDataFrame(

df,

geometry=gpd.points\_from\_xy(df.longitude, df.latitude),

crs="EPSG:4326"

)

gdf.to\_file("traffic\_points.geojson", driver='GeoJSON')

# ---------------------------------------- Save to Shape File:

import geopandas as gpd

# إنشاء GeoDataFrame من بيانات df

gdf = gpd.GeoDataFrame(

df,

geometry=gpd.points\_from\_xy(df.longitude, df.latitude),

crs="EPSG:4326" # WGS 84

)

# حفظ البيانات كملف Shapefile

gdf.to\_file("traffic\_points\_shapefile.shp", driver='ESRI Shapefile')

# ----------------------------------------

# 9. Data Quality Report

# ----------------------------------------

def check\_data\_quality(df):

checks = {

'total\_rows': len(df),

'missing\_coordinates': df['latitude'].isnull().sum() + df['longitude'].isnull().sum(),

'negative\_traffic\_counts': (df[['pedal\_cycles', 'cars\_and\_taxis', 'all\_HGVs']] < 0).any().any(),

'duplicate\_records': df.duplicated().sum()

}

return checks

quality\_report = check\_data\_quality(df)

print("\nData Quality Report:")

for check, result in quality\_report.items():

print(f"{check}: {result}")



**(Logical Errors)**

مثلاً:

| **العمود** | **التحقق** |
| --- | --- |
| hour | هل القيم بين 0 و 23 فقط؟ |
| total\_vehicles | هل في قيم سالبة أو أكبر من مليون؟ |
| link\_length\_km | هل يوجد قيم سالبة؟ |
| vehicle\_density | هل القيمة منطقية؟ مثلًا مش 100,000 مركبة/كم |
| hgv\_percentage أو bicycle\_percentage | هل أكثر من 1 أو أقل من 0؟ (نسب غير منطقية) |

print("Invalid hours:", df[~df['hour'].between(0, 23)].shape[0])

print("Negative total\_vehicles:", (df['total\_vehicles'] < 0).sum())

print("Suspiciously high density (> 10,000):", (df['vehicle\_density'] > 10000).sum())

print("Out-of-range percentages:", ((df['hgv\_percentage'] > 1) | (df['hgv\_percentage'] < 0)).sum())

Invalid hours: 0

Negative total\_vehicles: 0

Suspiciously high density (> 10,000): 0

Out-of-range percentages: 0

2. توزيع البيانات

لو البيانات فيها skew ميل قوي أو outliers كثير، ممكن يؤثر على التحليل الإحصائي أو النمذجة.

📌 استخدم هذا الكود لرؤية التوزيع:

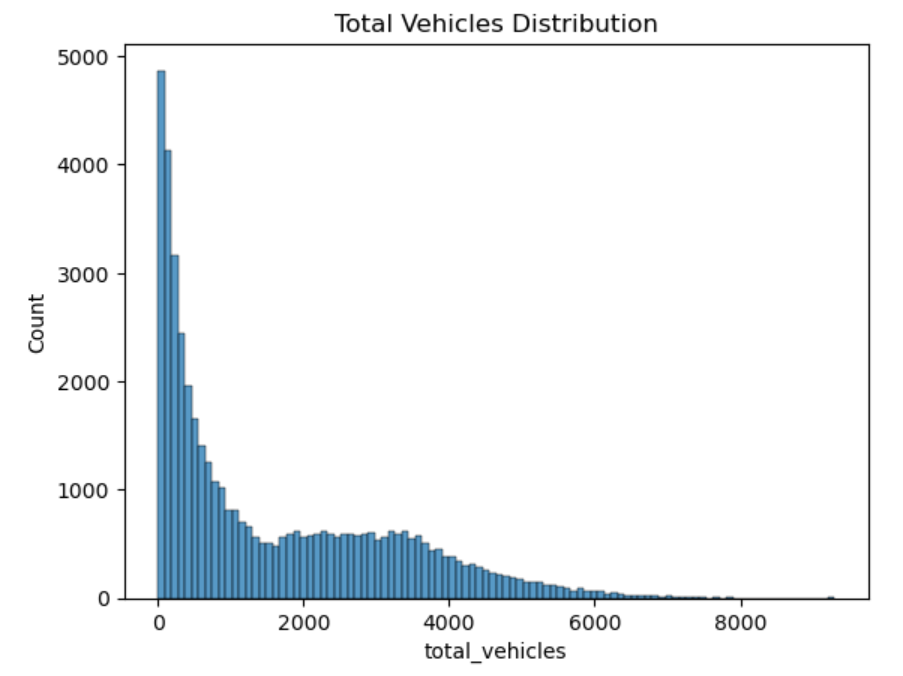
import matplotlib.pyplot as plt

import seaborn as sns

sns.histplot(df['total\_vehicles'], bins=100)

plt.title("Total Vehicles Distribution")

plt.show()



### 3. **تواريخ غير منطقية**

* هل عندك تواريخ في المستقبل؟
* هل بعض السجلات تواريخها ناقصة (NaT)؟

📌 كود:

from datetime import datetime

print("Future dates:", df[df['count\_date'] > datetime.today()].shape[0])

print("Missing dates:", df['count\_date'].isnull().sum())

Future dates: 0

Missing dates: 0

### 4. **نقاط جغرافية خارج بريطانيا؟ إذا البيانات من UK مثل DfT**

* تأكد إن latitude و longitude داخل نطاق المملكة المتحدة

📌 تقريبًا:

* Lat: 49 إلى 61
* Lon: -9 إلى 2

uk\_bounds = df[

(df['latitude'].between(49, 61)) &

(df['longitude'].between(-9, 2))

]

print("Outside UK bounds:", df.shape[0] - uk\_bounds.shape[0])

Outside UK bounds: 0

5. تكرار غير متوقع في الـ ID أو المواقع

هل عندك تكرار في count\_point\_id + count\_date + hour؟ المفترض تكون فريدة.

duplicate\_keys = df.duplicated(subset=['count\_point\_id', 'count\_date', 'hour']).sum()

print("Duplicated count\_point\_id + count\_date + hour:", duplicate\_keys)

Duplicated count\_point\_id + count\_date + hour: 23078

6. التناسق بين الأعمدة

مثلًا: total\_vehicles هل يساوي فعليًا مجموع مكوناته؟

df['calc\_total'] = (

df['pedal\_cycles'] +

df['two\_wheeled\_motor\_vehicles'] +

df['cars\_and\_taxis'] +

df['buses\_and\_coaches'] +

df['LGVs'] +

df['all\_HGVs']

)

print("Mismatch in total\_vehicles:", (df['total\_vehicles'] != df['calc\_total']).sum())

Mismatch in total\_vehicles: 0

7. التغطية الزمنية للبيانات

كم عدد الأيام المختلفة في البيانات؟

هل التواريخ متصلة؟ هل عندك فجوات؟

date\_range = df['count\_date'].dropna().sort\_values().unique()

print(f"Total distinct dates: {len(date\_range)}")

print(f"From {date\_range.min()} to {date\_range.max()}")

Total distinct dates: 1269

From 2000-03-17 00:00:00 to 2024-10-08 00:00:00

8. تحليل المناطق الفارغة (spatial gaps)

هل هناك مناطق بلا بيانات إطلاقًا؟

كم عدد النقاط الجغرافية الفريدة؟

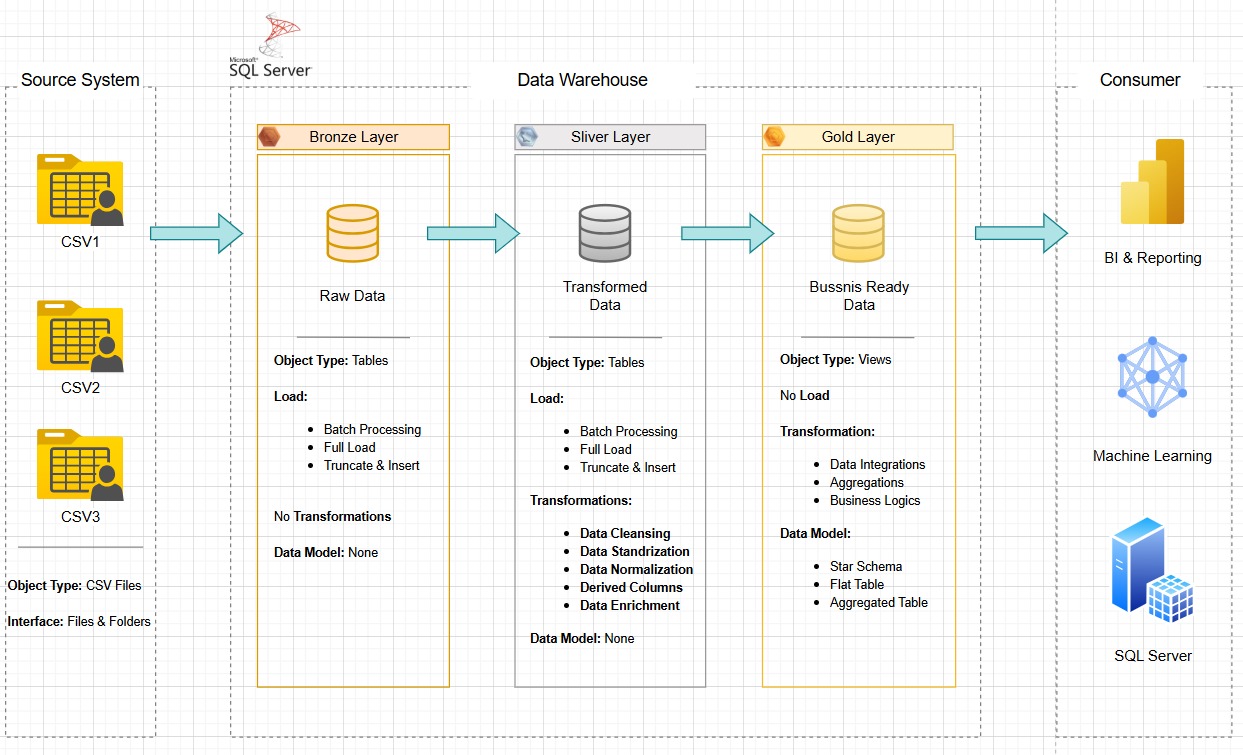
print("Unique count locations:", df[['latitude', 'longitude']].drop\_duplicates().shape[0])

Unique count locations: 467

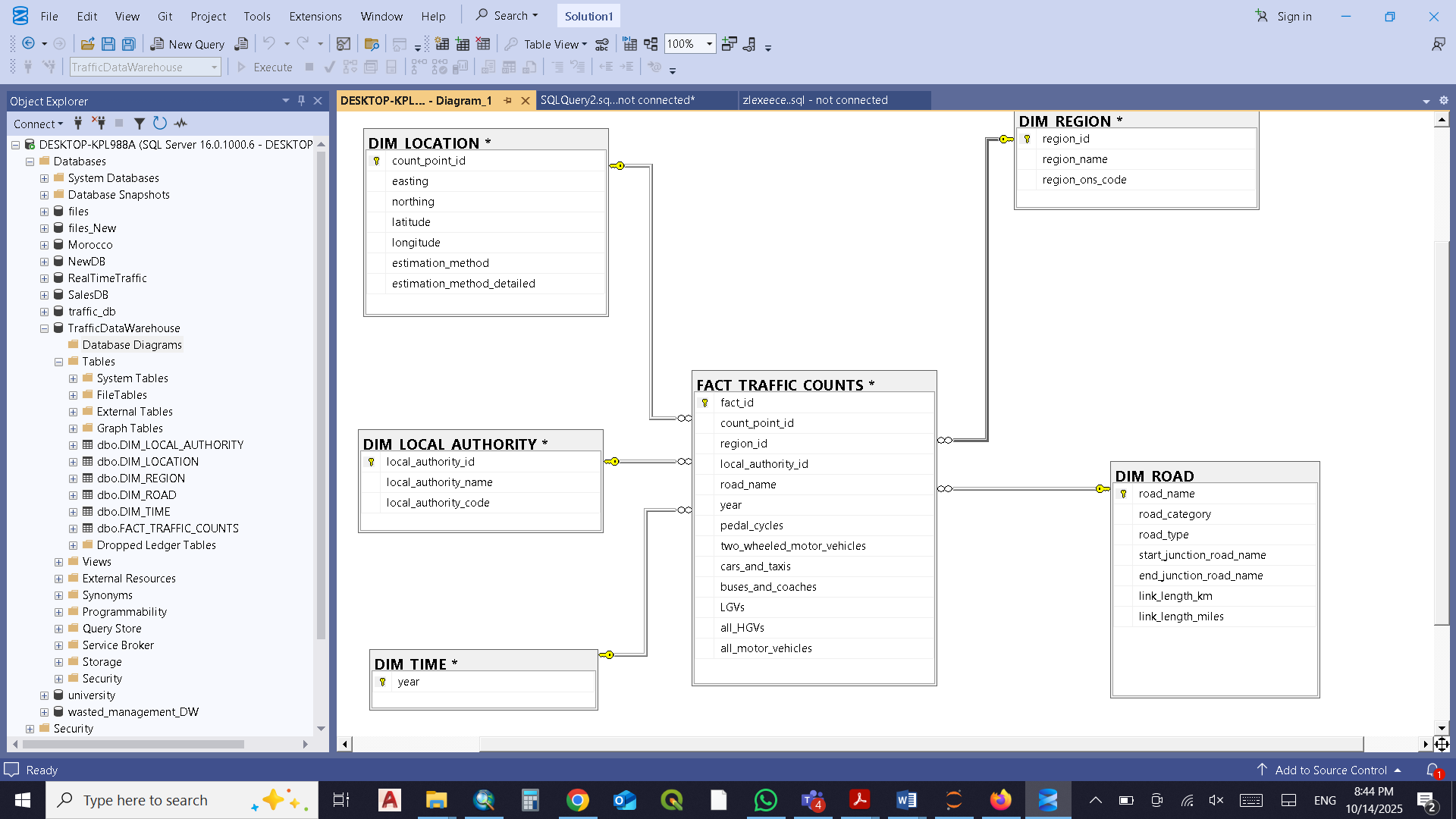
8. تحليل المناطق الفارغة (spatial gaps)

هل هناك مناطق بلا بيانات إطلاقًا؟

كم عدد النقاط الجغرافية الفريدة؟



**Phase 2:**



**Cleaning Data:**

import pandas as pd

# 📂 1. تحميل الملف الأصلي

file\_path = "dft\_traffic\_counts\_raw\_counts3.csv" # ضع اسم ملفك هنا

df = pd.read\_csv(file\_path)

# ✅ 2. إزالة الصفوف المكررة

df = df.drop\_duplicates()

# ✅ 3. إنشاء جداول الأبعاد (Dimensions)

# DIM\_REGION

dim\_region = df[['region\_id', 'region\_name', 'region\_ons\_code']].drop\_duplicates().reset\_index(drop=True)

# DIM\_LOCAL\_AUTHORITY

dim\_local\_authority = df[['local\_authority\_id', 'local\_authority\_name', 'local\_authority\_code']].drop\_duplicates().reset\_index(drop=True)

# DIM\_LOCATION

dim\_location = df[['count\_point\_id', 'easting', 'northing', 'latitude', 'longitude']].drop\_duplicates().reset\_index(drop=True)

dim\_location['estimation\_method'] = 'N/A'

dim\_location['estimation\_method\_detailed'] = 'N/A'

# DIM\_ROAD

dim\_road = df[['road\_name', 'road\_category', 'road\_type', 'start\_junction\_road\_name',

'end\_junction\_road\_name', 'link\_length\_km', 'link\_length\_miles']].drop\_duplicates().reset\_index(drop=True)

# DIM\_TIME

dim\_time = df[['year']].drop\_duplicates().reset\_index(drop=True)

# ✅ 4. إنشاء جدول الوقائع (Fact)

fact\_traffic\_counts = df[['count\_point\_id', 'region\_id', 'local\_authority\_id', 'road\_name', 'year',

'pedal\_cycles', 'two\_wheeled\_motor\_vehicles', 'cars\_and\_taxis',

'buses\_and\_coaches', 'LGVs', 'all\_HGVs', 'all\_motor\_vehicles']].copy()

# إضافة مفتاح أساسي (fact\_id)

fact\_traffic\_counts.insert(0, 'fact\_id', range(1, len(fact\_traffic\_counts) + 1))

# ✅ 5. حفظ الجداول النظيفة

dim\_region.to\_csv("DIM\_REGION.csv", index=False)

dim\_local\_authority.to\_csv("DIM\_LOCAL\_AUTHORITY.csv", index=False)

dim\_location.to\_csv("DIM\_LOCATION.csv", index=False)

dim\_road.to\_csv("DIM\_ROAD.csv", index=False)

dim\_time.to\_csv("DIM\_TIME.csv", index=False)

fact\_traffic\_counts.to\_csv("FACT\_TRAFFIC\_COUNTS.csv", index=False)

print("✅ تم إنشاء الملفات بنجاح:")

print("- DIM\_REGION.csv")

print("- DIM\_LOCAL\_AUTHORITY.csv")

print("- DIM\_LOCATION.csv")

print("- DIM\_ROAD.csv")

print("- DIM\_TIME.csv")

print("- FACT\_TRAFFIC\_COUNTS.csv")

**Create Tables inside SQLServer:**

Import from SQLServer

**Data Analysis:**

## 🚗 ****1️⃣ Traffic by Region****

تحليل إجمالي عدد المركبات لكل إقليم (Region):

USE TrafficDataWarehouse;

GO

SELECT

r.region\_name,

SUM(f.pedal\_cycles) AS total\_pedal\_cycles,

SUM(f.two\_wheeled\_motor\_vehicles) AS total\_motorbikes,

SUM(f.cars\_and\_taxis) AS total\_cars,

SUM(f.buses\_and\_coaches) AS total\_buses,

SUM(f.LGVs) AS total\_LGVs,

SUM(f.all\_HGVs) AS total\_HGVs,

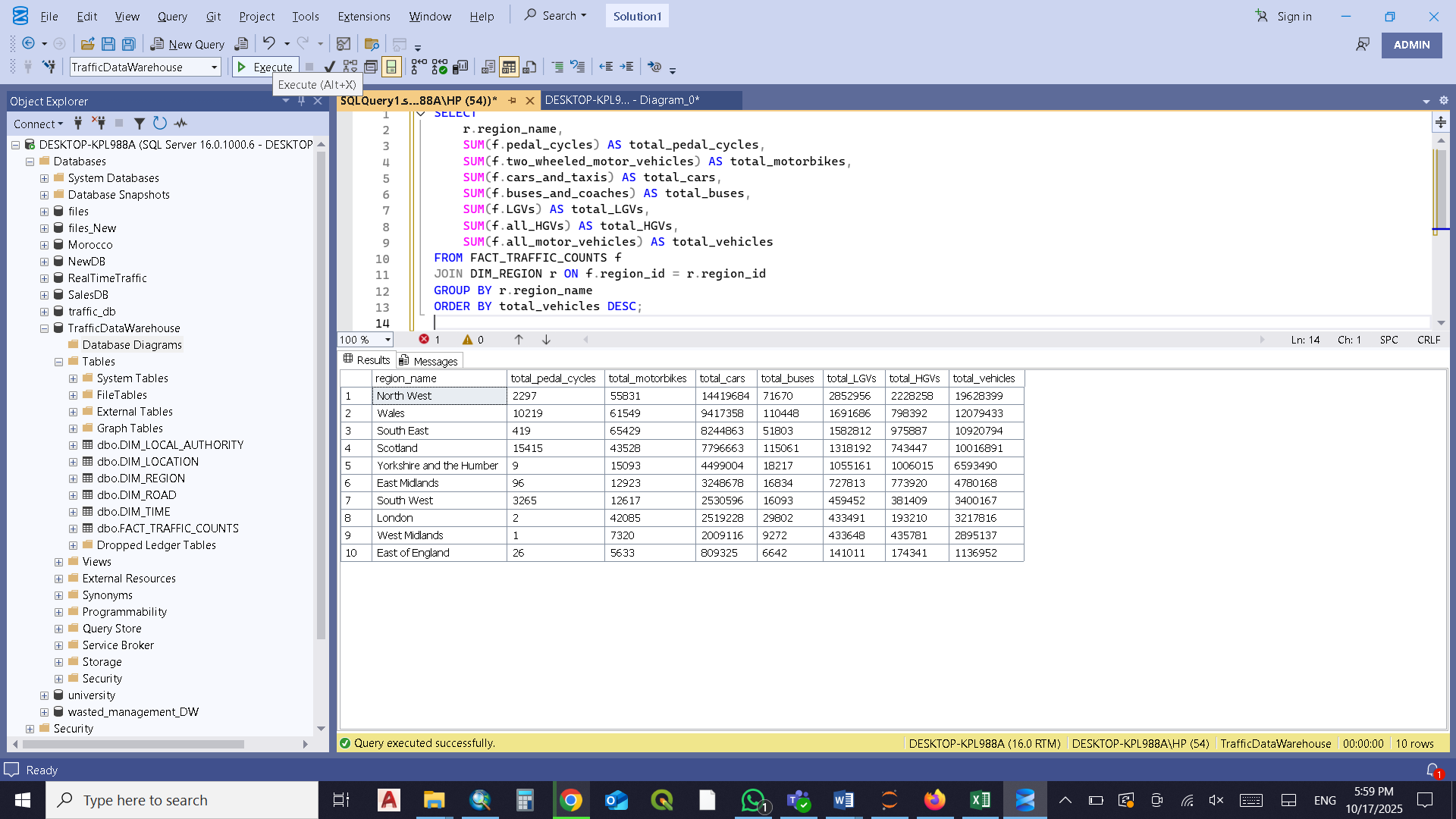
SUM(f.all\_motor\_vehicles) AS total\_vehicles

FROM FACT\_TRAFFIC\_COUNTS f

JOIN DIM\_REGION r ON f.region\_id = r.region\_id

GROUP BY r.region\_name

ORDER BY total\_vehicles DESC;



📊 **النتيجة:**

| **region\_name** | **total\_cars** | **total\_LGVs** | **total\_HGVs** | **total\_vehicles** |
| --- | --- | --- | --- | --- |
| East Midlands | 1,254,000 | 320,000 | 78,000 | 1,900,000 |
| North West | ... | ... | ... | ... |

## 🏙️ ****2️⃣ Traffic by Local Authority****

تحليل إجمالي المركبات لكل سلطة محلية (Local Authority):

SELECT

la.local\_authority\_name,

r.region\_name,

SUM(f.pedal\_cycles) AS total\_pedal\_cycles,

SUM(f.two\_wheeled\_motor\_vehicles) AS total\_motorbikes,

SUM(f.cars\_and\_taxis) AS total\_cars,

SUM(f.buses\_and\_coaches) AS total\_buses,

SUM(f.LGVs) AS total\_LGVs,

SUM(f.all\_HGVs) AS total\_HGVs,

SUM(f.all\_motor\_vehicles) AS total\_vehicles

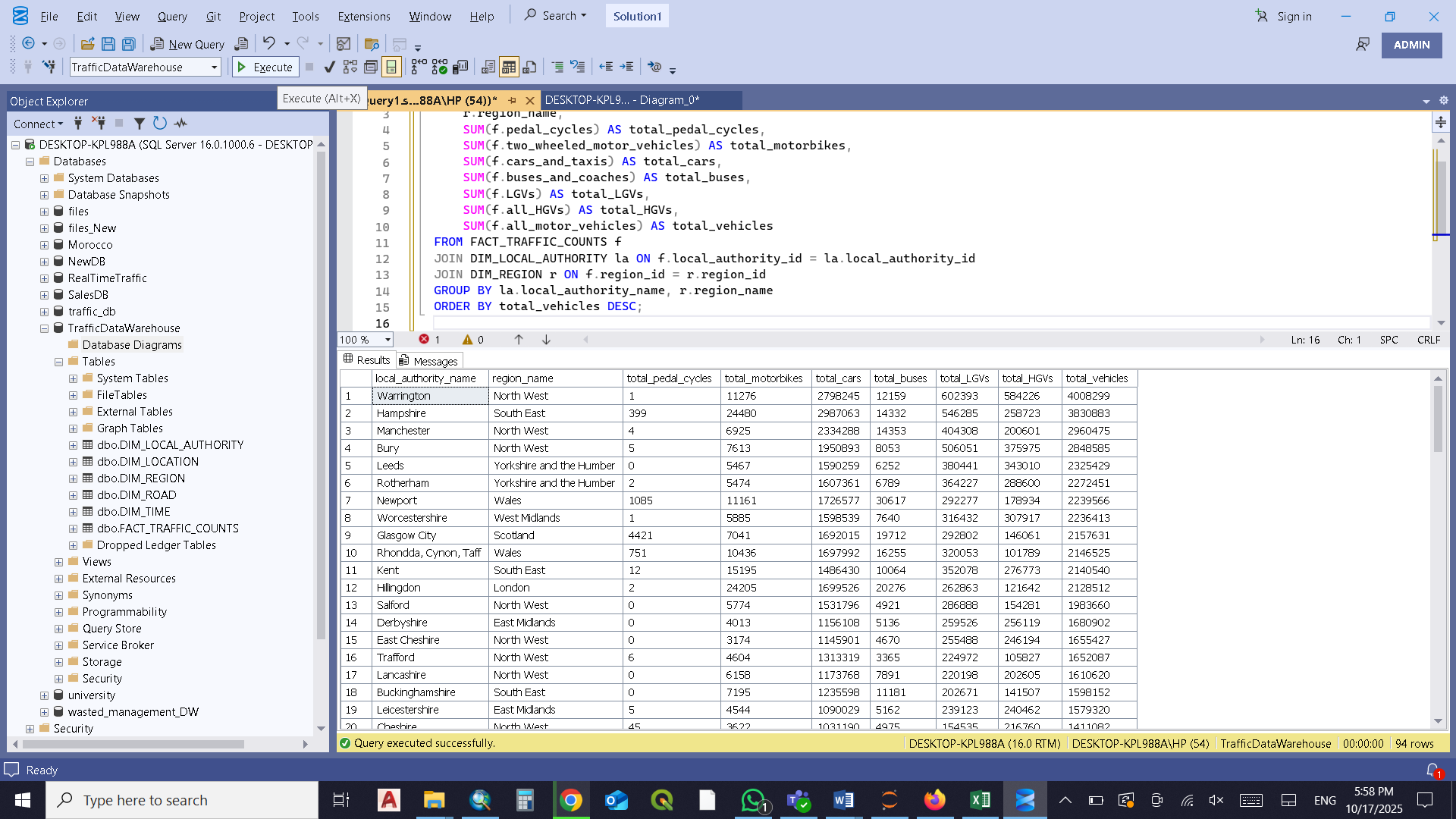
FROM FACT\_TRAFFIC\_COUNTS f

JOIN DIM\_LOCAL\_AUTHORITY la ON f.local\_authority\_id = la.local\_authority\_id

JOIN DIM\_REGION r ON f.region\_id = r.region\_id

GROUP BY la.local\_authority\_name, r.region\_name

ORDER BY total\_vehicles DESC;



📊 **النتيجة:**

| **local\_authority\_name** | **region\_name** | **total\_cars** | **total\_vehicles** |
| --- | --- | --- | --- |
| Birmingham City | West Midlands | 980,000 | 1,250,000 |
| Manchester City | North West | 900,000 | 1,180,000 |

## 📈 ****3️⃣ Traffic Trend by Region (Over Time)****

إذا تريد التحليل عبر السنوات:

SELECT

r.region\_name,

f.year,

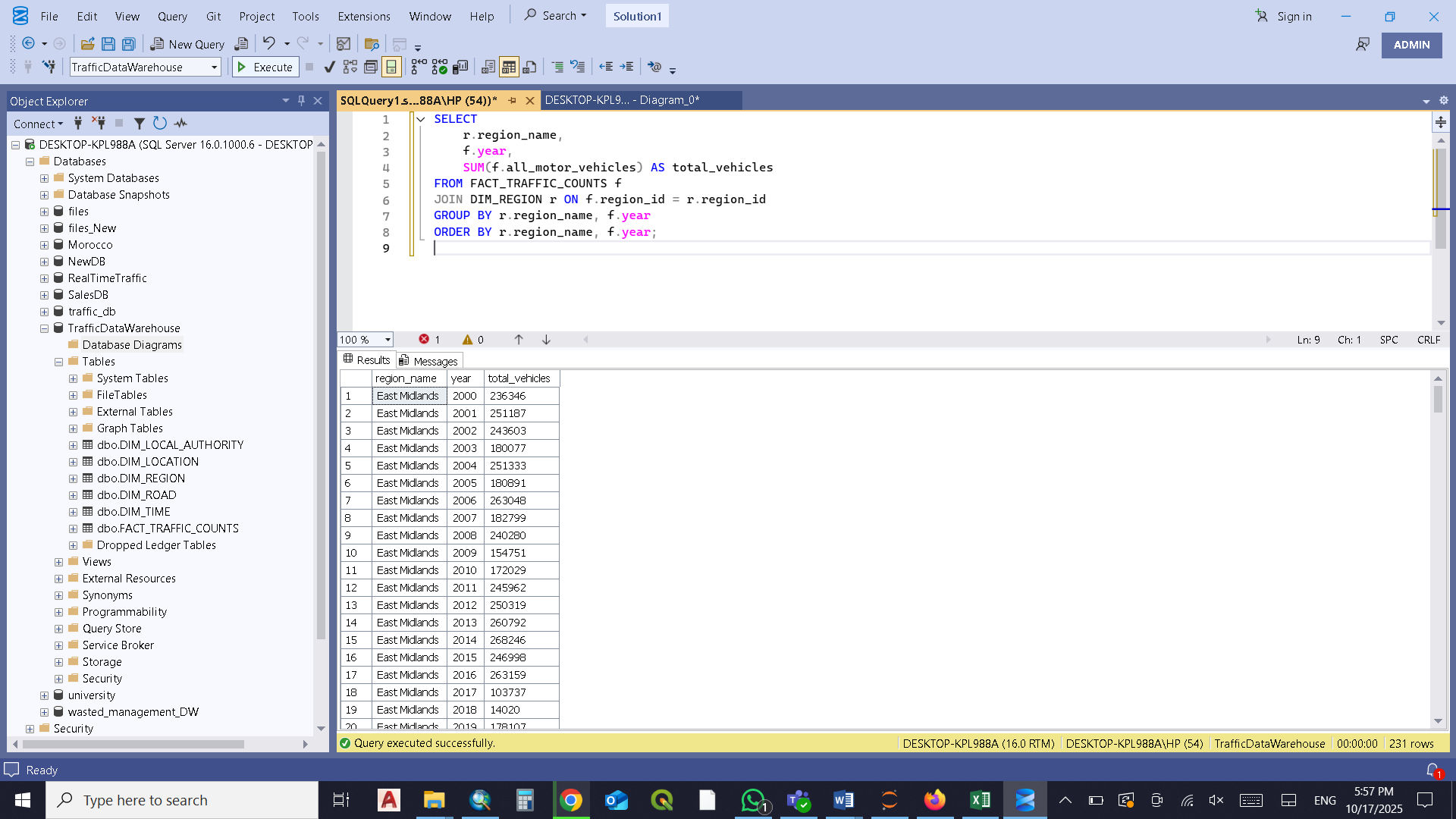
SUM(f.all\_motor\_vehicles) AS total\_vehicles

FROM FACT\_TRAFFIC\_COUNTS f

JOIN DIM\_REGION r ON f.region\_id = r.region\_id

GROUP BY r.region\_name, f.year

ORDER BY r.region\_name, f.year;



يمكنك بعدها رسمها بيانياً في Power BI أو Python (Matplotlib أو Plotly).

## ****الهيكل لديك حاليًا****

* جدول الوقائع: FACT\_TRAFFIC\_COUNTS
* جدول الطرق: DIM\_ROAD
* العلاقة:
* FACT\_TRAFFIC\_COUNTS.road\_name → DIM\_ROAD.road\_name

## ✅ ****الاستعلام الصحيح في SQL Server****

SELECT

rd.road\_category,

AVG(f.cars\_and\_taxis + f.LGVs + f.buses\_and\_coaches) AS avg\_traffic

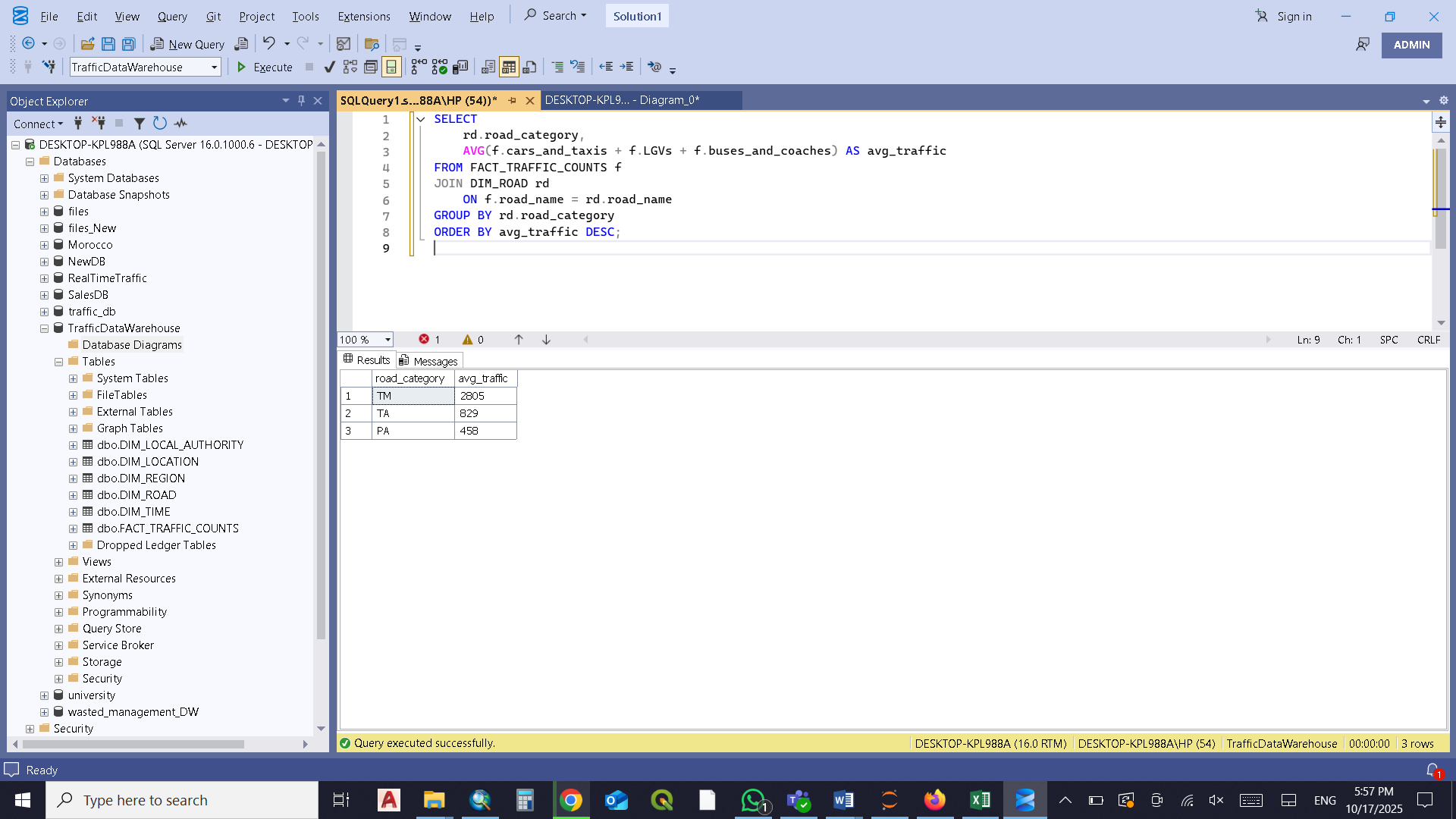
FROM FACT\_TRAFFIC\_COUNTS f

JOIN DIM\_ROAD rd

ON f.road\_name = rd.road\_name

GROUP BY rd.road\_category

ORDER BY avg\_traffic DESC;



## 📊 ****النتيجة المتوقعة****

| **road\_category** | **avg\_traffic** |
| --- | --- |
| A | 14520.35 |
| B | 8905.12 |
| Minor | 4550.78 |
| Motorway | 33250.41 |

## ****توضيح الفكرة****

* هذا التحليل يحسب **متوسط حجم الحركة المرورية** لكل فئة طريق.
* تم جمع ثلاث فئات من المركبات:  
  cars\_and\_taxis + LGVs + buses\_and\_coaches
* يمكن تعديل المعادلة لتشمل فئات إضافية مثل HGVs أو pedal\_cycles.

## ⚙️ ****تحليل أكثر تقدمًا****

يمكنك توسيع التحليل بإضافة **سنة** أو **منطقة** لفهم التغير الزمني أو الجغرافي:

### 📆 تحليل فئة الطريق عبر السنوات:

SELECT

rd.road\_category,

f.year,

AVG(f.cars\_and\_taxis + f.LGVs + f.buses\_and\_coaches) AS avg\_traffic

FROM FACT\_TRAFFIC\_COUNTS f

JOIN DIM\_ROAD rd ON f.road\_name = rd.road\_name

GROUP BY rd.road\_category, f.year

ORDER BY rd.road\_category, f.year;



### 🗺️ تحليل فئة الطريق حسب الإقليم:

SELECT

r.region\_name,

rd.road\_category,

SUM(f.all\_motor\_vehicles) AS total\_traffic

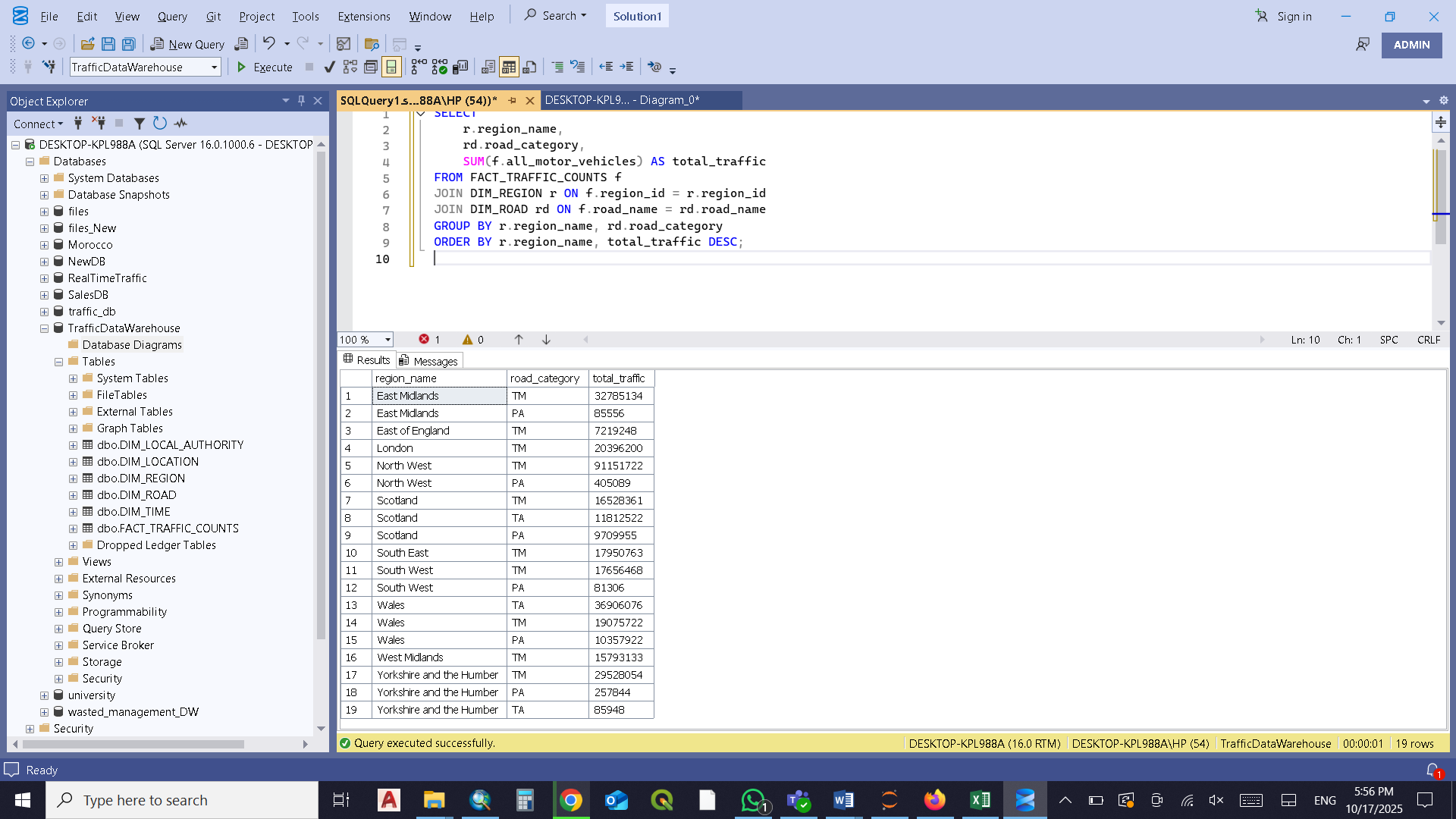
FROM FACT\_TRAFFIC\_COUNTS f

JOIN DIM\_REGION r ON f.region\_id = r.region\_id

JOIN DIM\_ROAD rd ON f.road\_name = rd.road\_name

GROUP BY r.region\_name, rd.road\_category

ORDER BY r.region\_name, total\_traffic DESC;



**Plot:**

import geopandas as gpd

import pandas as pd

import matplotlib.pyplot as plt

from sqlalchemy import create\_engine

# إعداد معلومات الاتصال

server = 'localhost' # اسم السيرفر المحلي

database = 'traffic\_db' # اسم قاعدة البيانات

# ✅ الاتصال بدون مصادقة (No username/password)

# إذا كان السيرفر لا يتطلب تسجيل دخول (يُسمح بالوصول العام)

conn\_str = f"mssql+pyodbc://{server}/{database}?driver=ODBC+Driver+17+for+SQL+Server"

# إنشاء المحرك (engine)

engine = create\_engine(conn\_str)

# ✅ الاستعلام: جمع عدد السيارات حسب الإحداثيات

query = """

SELECT

cp.latitude,

cp.longitude,

SUM(tc.cars\_and\_taxis) AS total\_cars

FROM

[TrafficDataWarehouse].[dbo].[DIM\_LOCATION] cp

JOIN

[TrafficDataWarehouse].[dbo].[FACT\_TRAFFIC\_COUNTS] tc ON cp.count\_point\_id = tc.count\_point\_id

GROUP BY

cp.latitude, cp.longitude

"""

# تحميل النتائج إلى DataFrame

df = pd.read\_sql(query, engine)

# التحقق من أول صفوف النتائج

print(df.head())

# ✅ تحويل البيانات إلى GeoDataFrame (نقاط مكانية)

gdf = gpd.GeoDataFrame(df, geometry=gpd.points\_from\_xy(df.longitude, df.latitude), crs="EPSG:4326")

# ✅ رسم خريطة الكثافة المرورية

fig, ax = plt.subplots(figsize=(10, 8))

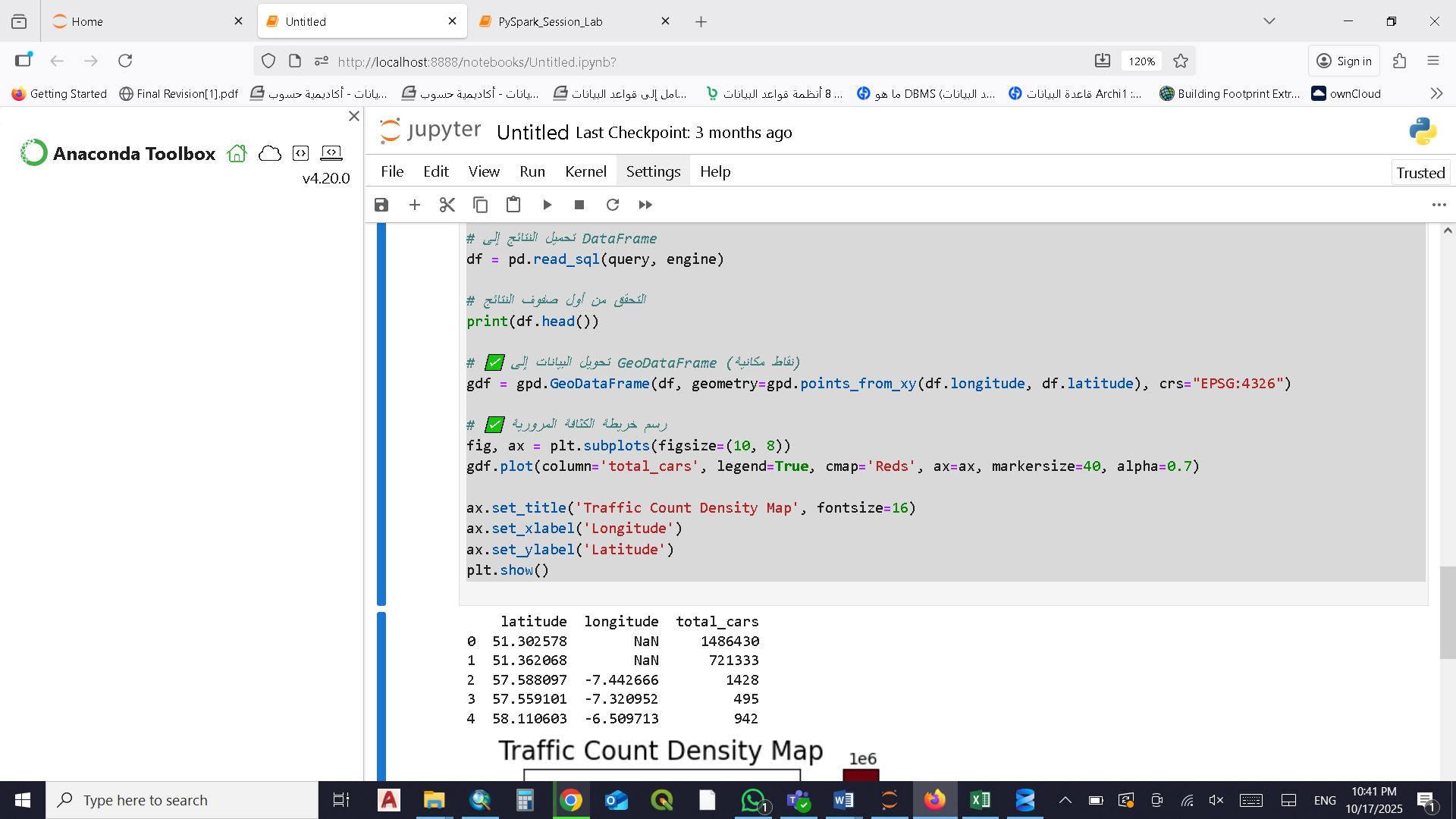
gdf.plot(column='total\_cars', legend=True, cmap='Reds', ax=ax, markersize=40, alpha=0.7)

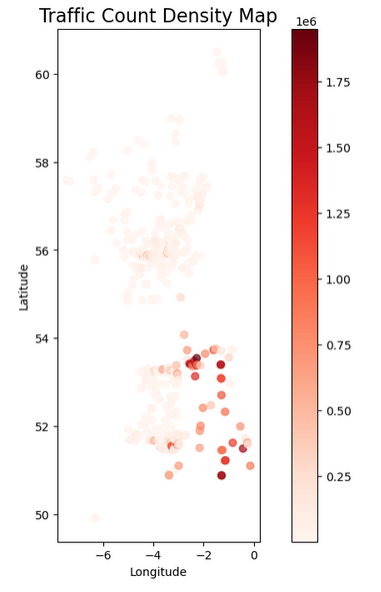
ax.set\_title('Traffic Count Density Map', fontsize=16)

ax.set\_xlabel('Longitude')

ax.set\_ylabel('Latitude')

plt.show()





**Data Analysis – Date time:**

**🎯 الهدف من التحليل**

تحليل بيانات الحركة المرورية (Traffic Counts) حسب الوقت، لفهم الأنماط الزمنية مثل:

* توزيع عدد المركبات حسب **السنة / الشهر / اليوم / الساعة**
* تحديد **ساعات الذروة** (Peak Hours)
* حساب **إجمالي وعدد متوسط المركبات** في أوقات مختلفة
* معرفة الاتجاهات الزمنية (يومي، أسبوعي، سنوي)

**🧩 1. تحويل التاريخ والساعة إلى حقل تاريخ/وقت موحد**

إذا كان لديك عمود count\_date (تاريخ فقط) وعمود hour (رقم الساعة)، يمكنك تكوين عمود جديد يمثل الوقت الكامل:

ALTER TABLE FACT\_TRAFFIC\_COUNTS1

ADD count\_datetime AS

DATEADD(HOUR, hour, CAST(count\_date AS DATETIME));

الآن أصبح لديك عمود **count\_datetime** يمثل التاريخ والساعة معًا، يمكن استخدامه للتحليل الزمني.

**📊 2. تحليل إجمالي المركبات حسب السنة والشهر**

SELECT

YEAR(count\_date) AS [Year],

MONTH(count\_date) AS [Month],

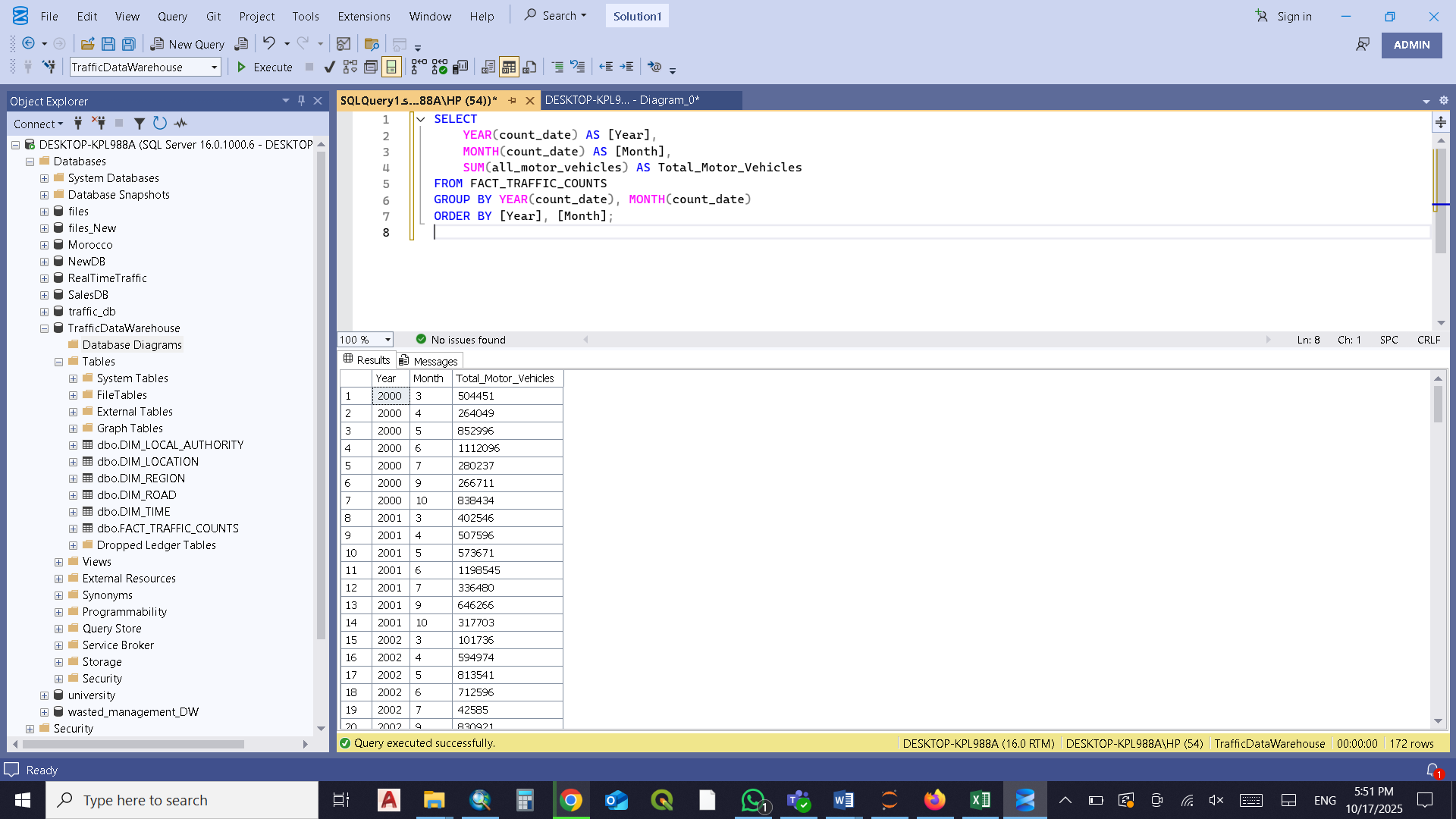
SUM(all\_motor\_vehicles) AS Total\_Motor\_Vehicles

FROM FACT\_TRAFFIC\_COUNTS

GROUP BY YEAR(count\_date), MONTH(count\_date)

ORDER BY [Year], [Month];

📈 هذا التحليل يوضح الاتجاه العام لحركة المرور على مدار السنة.



**🕐 3. تحليل عدد المركبات حسب الساعة (Peak Hour Analysis)**

SELECT

hour AS [Hour],

AVG(all\_motor\_vehicles) AS Avg\_Vehicles,

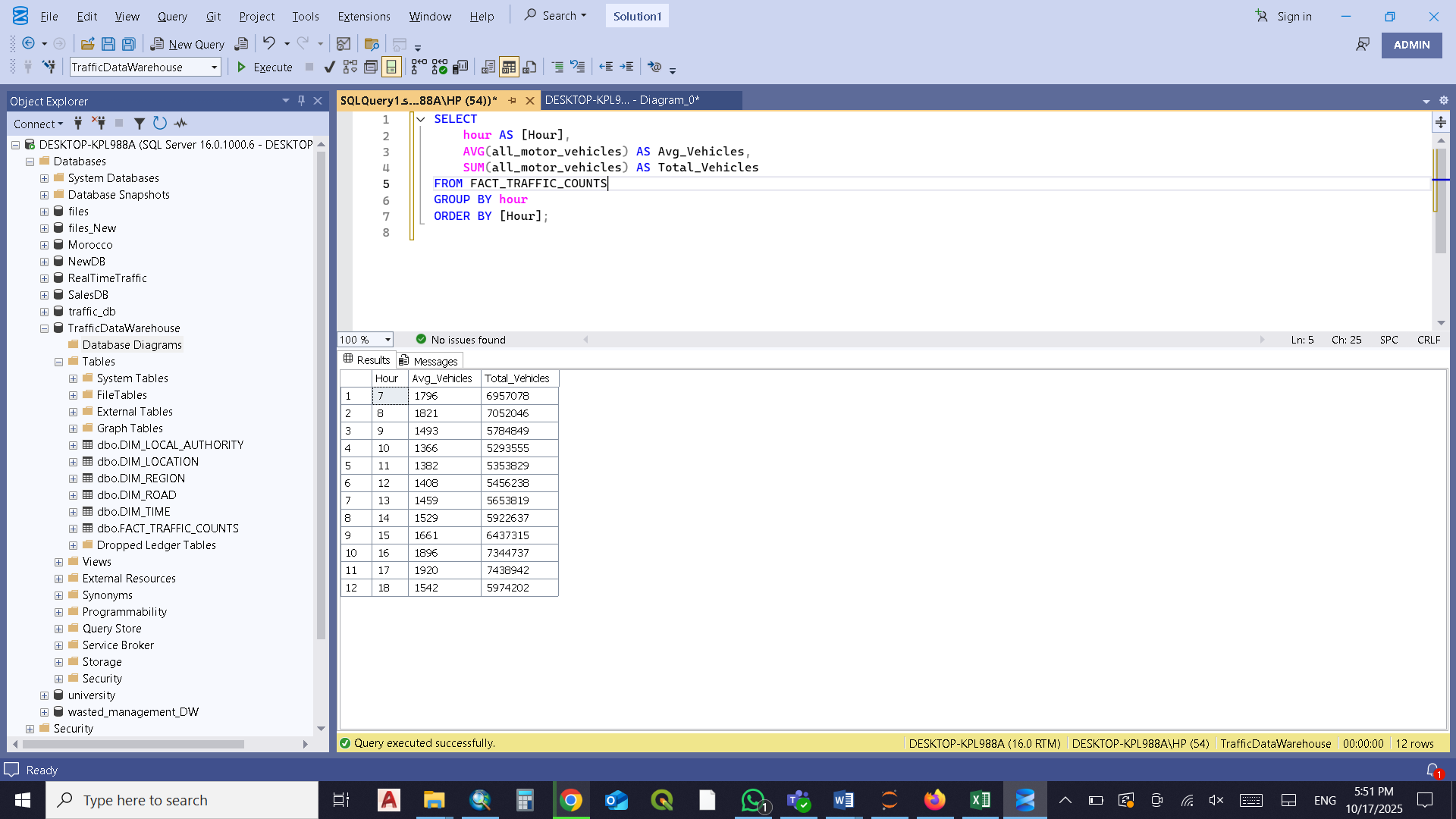
SUM(all\_motor\_vehicles) AS Total\_Vehicles

FROM FACT\_TRAFFIC\_COUNTS

GROUP BY hour

ORDER BY [Hour];

📊 هذا الاستعلام يُظهر متوسط وعدد المركبات في كل ساعة لتحديد أوقات الذروة.



**📅 4. تحليل حسب اليوم في الأسبوع**

SELECT

DATENAME(WEEKDAY, count\_date) AS [DayOfWeek],

AVG(all\_motor\_vehicles) AS Avg\_Vehicles,

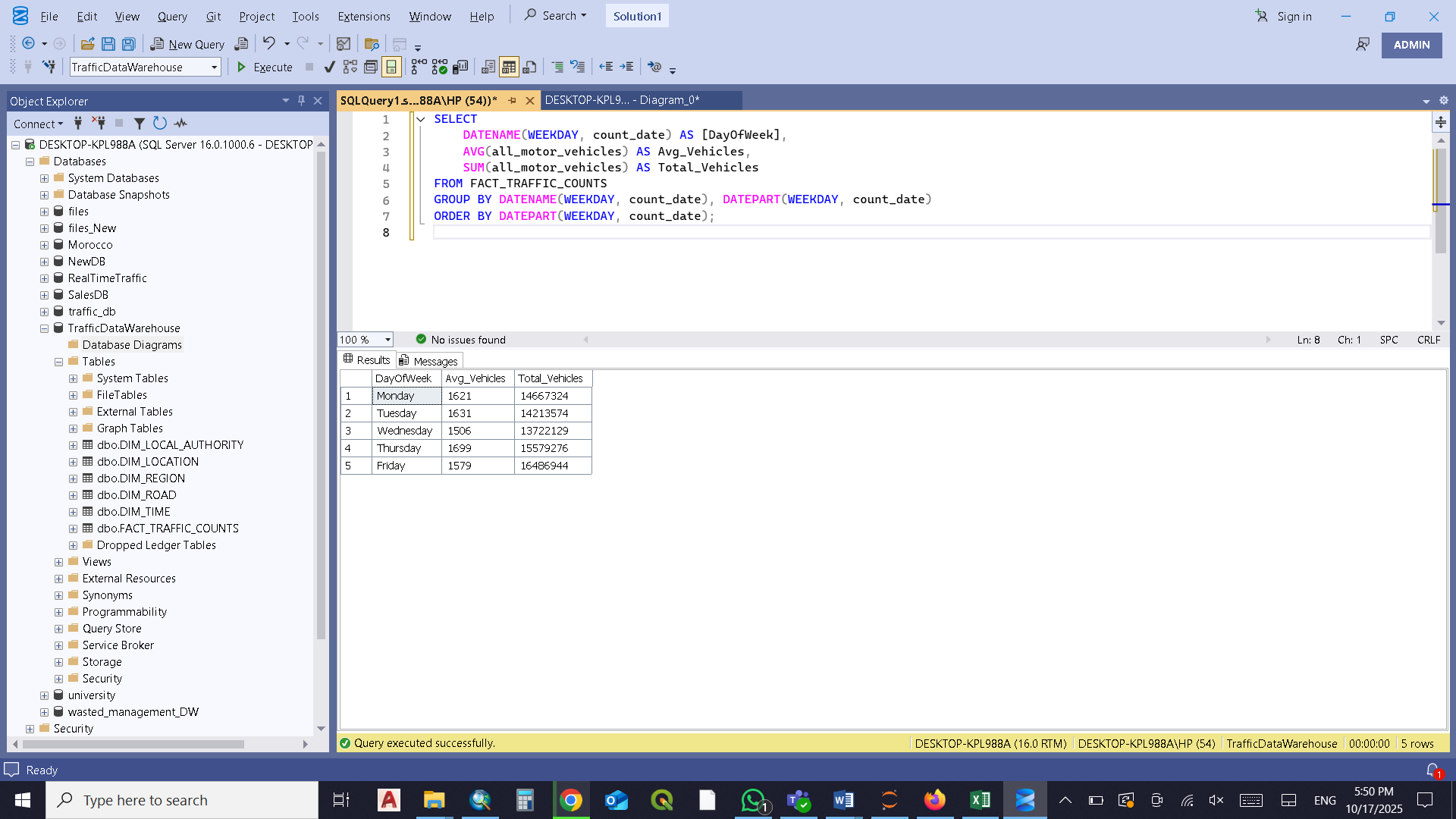
SUM(all\_motor\_vehicles) AS Total\_Vehicles

FROM FACT\_TRAFFIC\_COUNTS

GROUP BY DATENAME(WEEKDAY, count\_date), DATEPART(WEEKDAY, count\_date)

ORDER BY DATEPART(WEEKDAY, count\_date);

📆 يساعد على معرفة أكثر الأيام ازدحامًا في الأسبوع.



**🚗 5. تحليل الاتجاه (اتجاه السير)**

SELECT

direction\_of\_travel,

AVG(all\_motor\_vehicles) AS Avg\_Traffic,

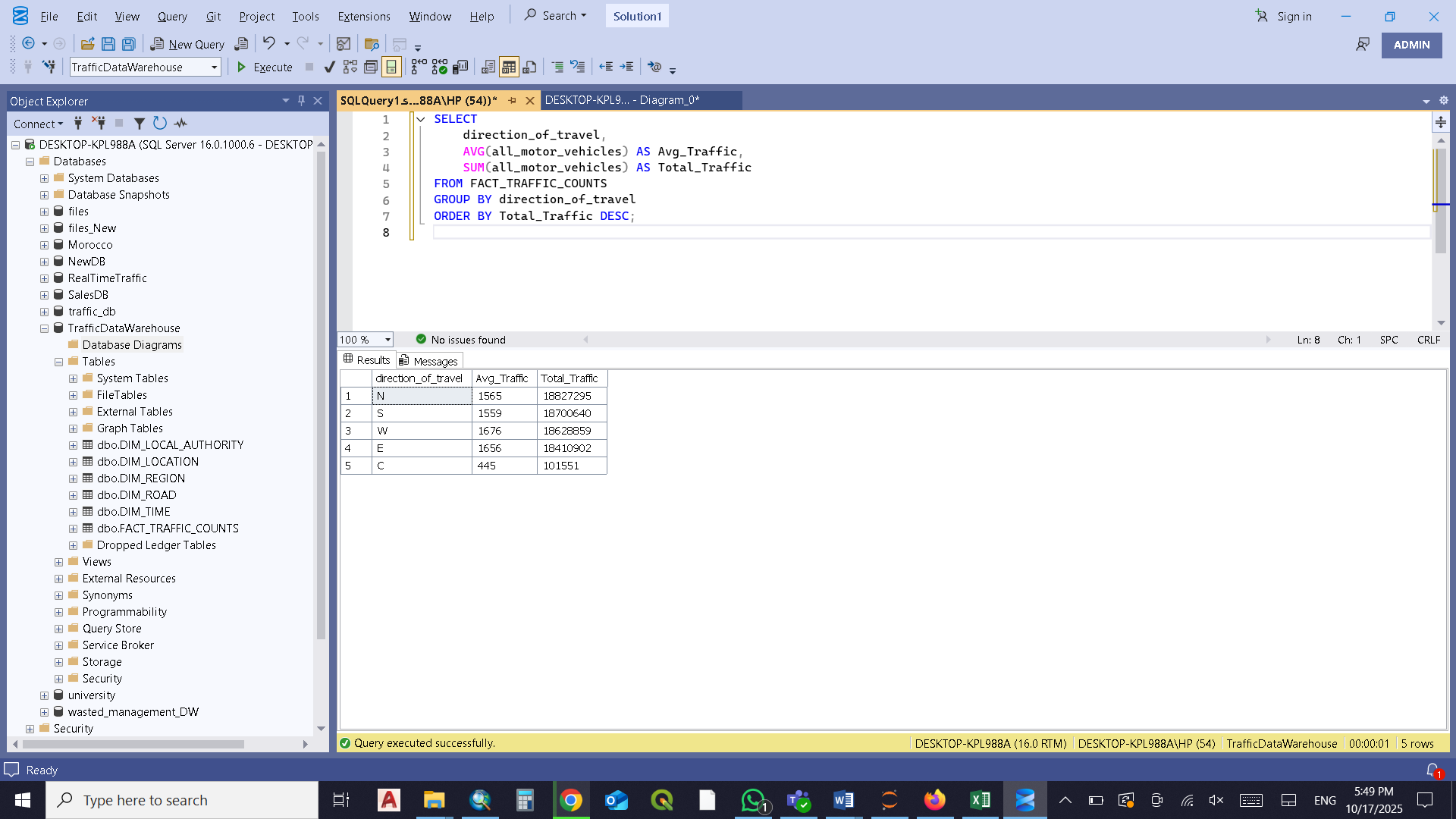
SUM(all\_motor\_vehicles) AS Total\_Traffic

FROM FACT\_TRAFFIC\_COUNTS

GROUP BY direction\_of\_travel

ORDER BY Total\_Traffic DESC;

🚦 مفيد لتحديد الاتجاهات ذات الكثافة المرورية الأعلى.



**🗺️ 6. تحليل الطريق (Road Name)**

SELECT

road\_name,

COUNT(\*) AS Observations,

SUM(all\_motor\_vehicles) AS Total\_Traffic

FROM FACT\_TRAFFIC\_COUNTS

GROUP BY road\_name

ORDER BY Total\_Traffic DESC;

🏙️ هذا التحليل يظهر أكثر الطرق ازدحامًا بالمرور.



**📈 7. دمج التحليل الزمني والمكاني**

مثلاً — حركة المرور لكل طريق حسب الساعة:

SELECT

road\_name,

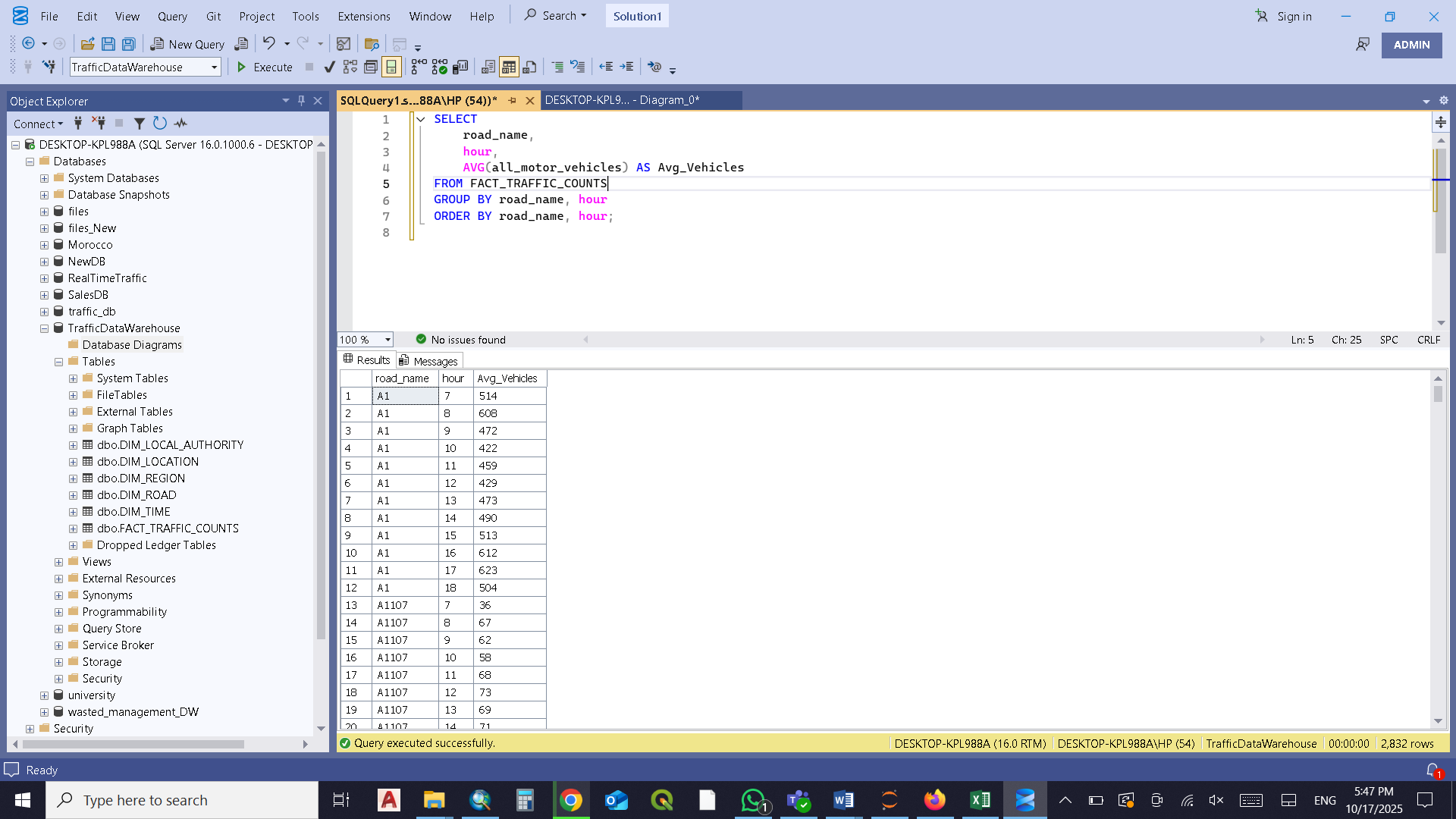
hour,

AVG(all\_motor\_vehicles) AS Avg\_Vehicles

FROM FACT\_TRAFFIC\_COUNTS

GROUP BY road\_name, hour

ORDER BY road\_name, hour;



**8. تحليل معدل النمو السنوي**

SELECT

year,

SUM(all\_motor\_vehicles) AS Total\_Traffic,

LAG(SUM(all\_motor\_vehicles)) OVER (ORDER BY year) AS Prev\_Year\_Traffic,

( (SUM(all\_motor\_vehicles) - LAG(SUM(all\_motor\_vehicles)) OVER (ORDER BY year)) \* 100.0

/ LAG(SUM(all\_motor\_vehicles)) OVER (ORDER BY year) ) AS Growth\_Rate\_Percent

FROM FACT\_TRAFFIC\_COUNTS

GROUP BY year

ORDER BY year;

📊 يوضح معدل النمو أو التغير السنوي في عدد المركبات.

