**Oklahoma State University**

**Spears School of Business**

**MSIS-5663-24593**

**Advanced Data Wrangling**

**Dr. RATHIN SARATHY**

**Term Project**

**By**

**TEAM -08**

**TUESDAY SECTION**

**Srivarshika Gadde – A20450847**

**Kishore Madanagopal – A20450297**

**Somunath Reddy Sirasanambeti – A20446153**

**Kumari Rudraraju Subramanyam – A20441226**

**SCHEMA: OLTP**

A screenshot of a computer

Description automatically generated

**SCHEMA: OLAP**

A screenshot of a computer

Description automatically generated

**TASK-1**

**Stored procedure to create tables and drop tables**

-> **Create Tables:**

CREATE PROCEDURE [dbo].[CREATE\_TABLES]

AS

BEGIN

-- Creating the Accident\_Details table

CREATE TABLE Accident\_Details (

Accident\_ID INT IDENTITY(1,1) PRIMARY KEY,

Accident\_Index VARCHAR(500),

Accident\_Severity VARCHAR(255),

Carriageway\_Hazards VARCHAR(255) DEFAULT NULL,

Did\_Police\_Officer\_Attend\_Scene\_of\_Accident INT,

Junction\_Control VARCHAR(255),

Junction\_Detail VARCHAR(255),

Latitude FLOAT,

Light\_Conditions VARCHAR(255),

[Local\_Authority\_District] VARCHAR(255),

[Local\_Authority\_Highway] VARCHAR(255),

Location\_Easting\_OSGR INT,

Location\_Northing\_OSGR INT,

Longitude FLOAT,

LSOA\_of\_Accident\_Location VARCHAR(255),

Number\_of\_Casualties INT,

Number\_of\_Vehicles INT,

[Pedestrian\_Crossing\_Human\_Control] INT,

[Pedestrian\_Crossing\_Physical\_Facilities] INT,

Police\_Force VARCHAR(255),

Special\_Conditions\_at\_Site VARCHAR(255) DEFAULT NULL,

Speed\_limit INT DEFAULT NULL,

Urban\_or\_Rural\_Area VARCHAR(255),

Weather\_Conditions VARCHAR(255),

Year INT,

InScotland VARCHAR(225) DEFAULT NULL

);

-- Creating the Accident\_Time table

CREATE TABLE Accident\_Time (

Time\_ID INT IDENTITY(1,1) PRIMARY KEY,

Accident\_ID INT,

Accident\_Index VARCHAR(500),

Date DATE,

Day\_of\_Week VARCHAR(255),

Time TIME,

CONSTRAINT FK\_Accident\_Time\_Accident\_Details FOREIGN KEY (Accident\_ID)

REFERENCES Accident\_Details(Accident\_ID)

ON DELETE CASCADE

);

-- Creating the Accident\_Road\_Details table

CREATE TABLE Accident\_Road\_Details (

Road\_ID INT IDENTITY(1,1) PRIMARY KEY,

Accident\_Index VARCHAR(500),

First\_Road\_Class VARCHAR(255),

First\_Road\_Number INT,

Second\_Road\_Class VARCHAR(255) DEFAULT NULL,

Second\_Road\_Number INT DEFAULT NULL,

Road\_Surface\_Conditions VARCHAR(255),

Road\_Type VARCHAR(255),

Accident\_ID INT,

CONSTRAINT FK\_Accident\_Road\_Details\_Accident\_Details FOREIGN KEY (Accident\_ID) REFERENCES Accident\_Details(Accident\_ID) ON DELETE CASCADE

);

-- Creating the Vehicle\_Details table

CREATE TABLE Vehicle\_Details (

Vehicle\_ID INT IDENTITY(1,1) PRIMARY KEY,

Accident\_ID INT,

Accident\_Index VARCHAR(500),

Hit\_Object\_in\_Carriageway VARCHAR(255),

Hit\_Object\_off\_Carriageway VARCHAR(255),

Journey\_Purpose\_of\_Driver VARCHAR(255),

Junction\_Location VARCHAR(255),

make VARCHAR(255),

model VARCHAR(255),

Skidding\_and\_Overturning VARCHAR(255),

Towing\_and\_Articulation VARCHAR(255),

Vehicle\_Leaving\_Carriageway VARCHAR(255),

Vehicle\_Location\_Restricted\_Lane VARCHAR(255),

Vehicle\_Manoeuvre VARCHAR(255),

Vehicle\_Type VARCHAR(255),

Was\_Vehicle\_Left\_Hand\_Drive VARCHAR(255),

X1st\_Point\_of\_Impact VARCHAR(255),

Age\_Band\_of\_Driver VARCHAR(255),

Age\_of\_Vehicle INT,

Driver\_Home\_Area\_Type VARCHAR(255),

Driver\_IMD\_Decile INT,

Engine\_Capacity\_CC INT,

Propulsion\_Code VARCHAR(255),

Sex\_of\_Driver VARCHAR(255),

Vehicle\_Reference INT

CONSTRAINT FK\_Vehicle\_Details\_Accident\_Details FOREIGN KEY (Accident\_ID)

REFERENCES Accident\_Details(Accident\_ID)

ON DELETE CASCADE

);

END;

GO

EXEC [dbo].[CREATE\_TABLES];

-> **Drop Table:**

CREATE PROCEDURE [dbo].[DROP\_TABLES]

AS

BEGIN

-- Dropping the Vehicle\_Details table

IF EXISTS (SELECT \* FROM sys.objects WHERE object\_id = OBJECT\_ID(N'[dbo].[Vehicle\_Details]') AND type in (N'U'))

DROP TABLE Vehicle\_Details;

-- Dropping the Accident\_Road\_Details table

IF EXISTS (SELECT \* FROM sys.objects WHERE object\_id = OBJECT\_ID(N'[dbo].[Accident\_Road\_Details]') AND type in (N'U'))

DROP TABLE Accident\_Road\_Details;

-- Dropping the Accident\_Time table

IF EXISTS (SELECT \* FROM sys.objects WHERE object\_id = OBJECT\_ID(N'[dbo].[Accident\_Time]') AND type in (N'U'))

DROP TABLE Accident\_Time;

-- Dropping the Accident\_Details table

IF EXISTS (SELECT \* FROM sys.objects WHERE object\_id = OBJECT\_ID(N'[dbo].[Accident\_Details]') AND type in (N'U'))

DROP TABLE Accident\_Details;

END;

GO

EXEC [dbo].[DROP\_TABLES];

Design and implement a normalized OLTP database based on the extracted accident and vehicle data. You will then develop 2 views and 5 useful SQL queries. At least two of the 5 SQL queries must use one or more of these views. At least two of the 5 SQL queries must employ aggregations. At least three of the 5 SQL queries must involve joins or sub-queries.

**QUERIES TO CREATE VIEWS**:

-> **View 1: AccidentSummaryView:**

CREATE VIEW AccidentSummaryView AS

SELECT ad.Accident\_ID, ad.Accident\_Index, ad.Accident\_Severity, ad.Number\_of\_Vehicles, ad.Number\_of\_Casualties,

at.Date, at.Time, at.Day\_of\_Week,

ard.First\_Road\_Class, ard.Road\_Surface\_Conditions, ard.Road\_Type

FROM Accident\_Details ad

JOIN Accident\_Time at ON ad.Accident\_ID = at.Accident\_ID

JOIN Accident\_Road\_Details ard ON ad.Accident\_ID = ard.Accident\_ID;

-> **View 2: VehicleAccidentView**

CREATE VIEW VehicleAccidentView AS

SELECT vd.Accident\_ID, vd.Accident\_Index, vd.make, vd.model, vd.Vehicle\_Type, vd.Age\_of\_Vehicle,

ad.Accident\_Severity, ad.Weather\_Conditions, ad.Speed\_limit

FROM Vehicle\_Details vd

JOIN Accident\_Details ad ON vd.Accident\_ID = ad.Accident\_ID;

->**Query 1:**

**Title: Count of accidents by day of the week and severity using Views.**

This SQL query analyzes accident severity across different days of the week by selecting data from AccidentSummaryView. It groups the results by Day\_of\_Week and Accident\_Severity and uses the COUNT() function to calculate the total number of accidents for each group. The results are then ordered by the day of the week to display trends and patterns in accident frequency and severity.

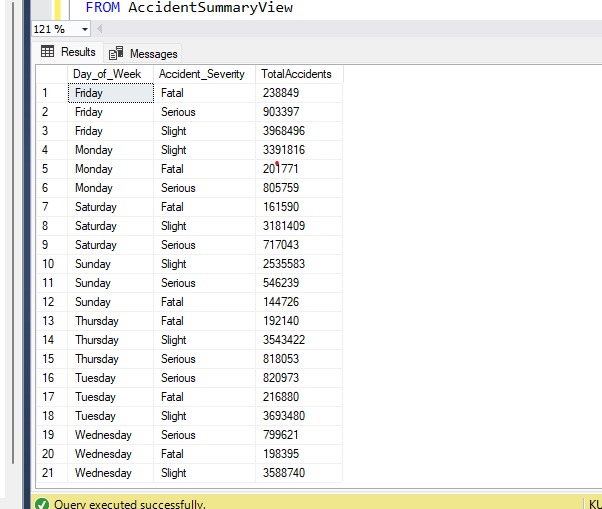
**How many accidents occurred on each day of the week, categorized by the severity of the accident?**

SELECT Day\_of\_Week, Accident\_Severity, COUNT(\*) AS TotalAccidents

FROM AccidentSummaryView

GROUP BY Day\_of\_Week, Accident\_Severity

ORDER BY Day\_of\_Week;



->**Query 2:**

**Title: Get the count of accidents by severity**

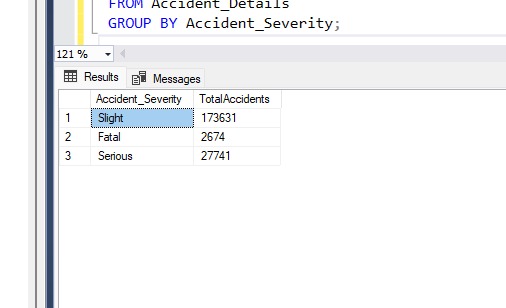
This SQL query selects data from the Accident\_Details table to compute the number of accidents grouped by their severity using the COUNT() function, which tallies entries in each category. The results are grouped by Accident\_Severity, providing a count of accidents for each severity level, useful for understanding the distribution of accident severity.

**What is the total number of accidents categorized by their severity?**

SELECT Accident\_Severity, COUNT(\*) AS TotalAccidents

FROM Accident\_Details

GROUP BY Accident\_Severity;



->**Query 3:**

**Title: Query 3: Average number of casualties in accidents per year**

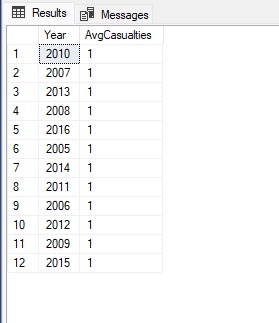
This SQL query calculates the average number of casualties per accident for each year from the Accident\_Details table. It uses the AVG() function to determine the mean value of Number\_of\_Casualties, and the results are grouped by Year to provide annual statistics. This helps in analyzing trends in accident severity over different years.

**What is the average number of casualties per accident for each year?**

SELECT Year, AVG(Number\_of\_Casualties) AS AvgCasualties

FROM Accident\_Details

GROUP BY Year;



->**Query 4:**

**Title: calculates the average number of vehicles involved in all recorded accidents**

This SQL query retrieves accidents from the Accident\_Details table where the number of vehicles involved exceeds the average across all accidents. It selects columns like Accident\_ID, Accident\_Index, Number\_of\_Vehicles, and Accident\_Severity. The average is calculated using a subquery that computes the mean number of vehicles from the same table, helping identify and analyze unusually large accidents.

**Which accidents involved more vehicles than the average number of vehicles per accident?**

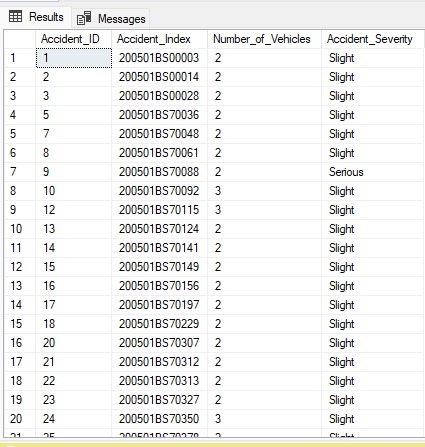
SELECT ad.Accident\_ID,ad.Accident\_Index,ad.Number\_of\_Vehicles, ad.Accident\_Severity

FROM Accident\_Details ad

WHERE ad.Number\_of\_Vehicles >

(SELECT AVG(Number\_of\_Vehicles)

FROM Accident\_Details);



->**Query 5**:

**Title: calculates the overall average number of casualties in all recorded accidents**

This SQL query selects records from the Accident\_Details table where the number of casualties exceeds the average for all accidents. It retrieves fields such as Accident\_ID, Accident\_Index, Number\_of\_Casualties, and Weather\_Conditions. The average number of casualties is computed by a subquery, focusing analysis on severe accidents and their weather conditions.

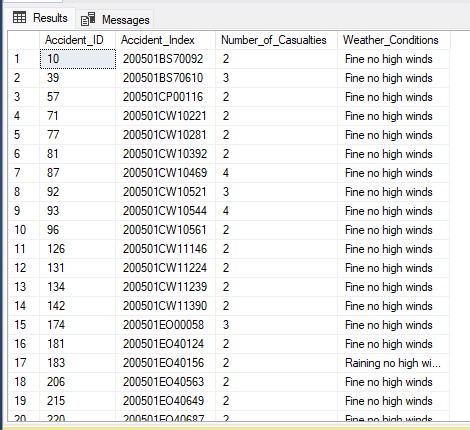
**Which accidents had more casualties than the average number of casualties per accident, and what were the weather conditions during those accidents?**

SELECT ad.Accident\_ID, ad.Accident\_Index, ad.Number\_of\_Casualties, ad.Weather\_Conditions

FROM Accident\_Details ad

WHERE ad.Number\_of\_Casualties >

(SELECT AVG(Number\_of\_Casualties) FROM Accident\_Details);



->**Query 6:**

**Title: Detailed Vehicle Information for Each Accident**

The SQL query performs an INNER JOIN between the Accident\_Details and Vehicle\_Details tables, linking them via the common Accident\_ID field. This join operation allows the query to gather comprehensive details from both tables, such as accident severity and vehicle specifics like make and model. Although no computational SQL functions are used in this query, the JOIN function is crucial for combining relevant data from the two tables, enabling detailed analysis of vehicle involvement in accidents.

**What are the details of each accident, including the severity, make, model, and type of vehicle involved?**

SELECT

ad.Accident\_ID,

ad.Accident\_Index,

ad.Accident\_Severity,

vd.make,

vd.model,

vd.Vehicle\_Type

FROM

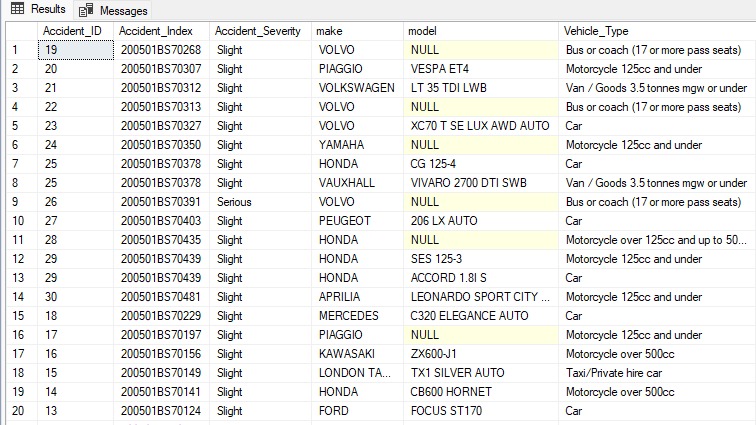
Accident\_Details ad

JOIN

Vehicle\_Details vd

ON

ad.Accident\_ID = vd.Accident\_ID;



->**Query 7:**

**Title:** **Analysis of Vehicle Accident Frequency by Type and Weather Conditions**

This SQL query selects the top 10 records from the VehicleAccidentView to analyze accident data by vehicle type and weather conditions. It employs COUNT(Accident\_ID) to determine the total number of accidents and AVG(Speed\_limit) to calculate the average speed limit for each category. The results are grouped by Vehicle\_Type and Weather\_Conditions and ordered by the number of accidents in descending order, highlighting the most prevalent conditions for accidents.

**What are the top 10 combinations of vehicle type and weather conditions that have the highest number of accidents, and what is the average speed limit reported in these conditions?**

SELECT TOP 10

Vehicle\_Type,

Weather\_Conditions,

COUNT(Accident\_ID) AS TotalAccidents,

AVG(Speed\_limit) AS AverageSpeedLimit

FROM

VehicleAccidentView

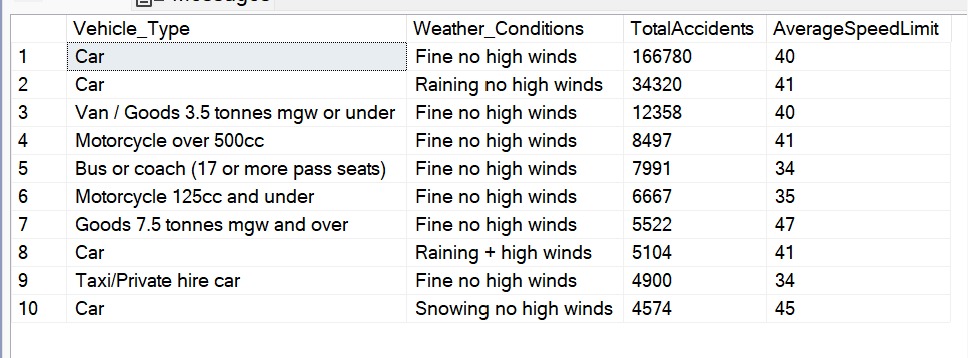
GROUP BY

Vehicle\_Type,

Weather\_Conditions

ORDER BY

TotalAccidents DESC;



**TASK-2**

**Stored procedure to create tables and drop tables**

-> **Create Tables:**

CREATE PROCEDURE [dbo].[CREATE\_TABLES]

AS

BEGIN

CREATE TABLE AccidentDetailsDim (

Accident\_ID INT IDENTITY(1,1) PRIMARY KEY,

Accident\_Index VARCHAR(500),

Accident\_Severity VARCHAR(255),

Carriageway\_Hazards VARCHAR(255) DEFAULT NULL,

Did\_Police\_Officer\_Attend\_Scene\_of\_Accident INT,

Junction\_Control VARCHAR(255),

Junction\_Detail VARCHAR(255),

Longitude FLOAT,

Light\_Conditions VARCHAR(255),

Local\_Authority\_District VARCHAR(255),

Local\_Authority\_Highway VARCHAR(255),

Location\_Easting\_OSGR INT,

Location\_Northing\_OSGR INT,

Latitude FLOAT,

LSOA\_of\_Accident\_Location VARCHAR(255),

Pedestrian\_Crossing\_Human\_Control INT,

Pedestrian\_Crossing\_Physical\_Facilities INT,

Police\_Force VARCHAR(255),

Special\_Conditions\_at\_Site VARCHAR(255) DEFAULT NULL,

Urban\_or\_Rural\_Area VARCHAR(255),

Weather\_Conditions VARCHAR(255),

Year INT,

InScotland VARCHAR(255) DEFAULT NULL

);

CREATE TABLE AccidentRoadDetailsDim (

Road\_ID INT IDENTITY(1,1) PRIMARY KEY,

First\_Road\_Class VARCHAR(255),

First\_Road\_Number INT,

Second\_Road\_Class VARCHAR(255) DEFAULT NULL,

Second\_Road\_Number INT DEFAULT NULL,

Road\_Surface\_Conditions VARCHAR(255),

Road\_Type VARCHAR(255)

);

CREATE TABLE AccidentTimeDim (

Date\_ID INT IDENTITY(1,1) PRIMARY KEY,

Date DATE,

Day\_of\_Week VARCHAR(255),

Time TIME(7)

);

CREATE TABLE VehicleDetailsDim (

Vehicle\_ID INT IDENTITY(1,1) PRIMARY KEY,

Hit\_Object\_in\_Carriageway VARCHAR(255),

Hit\_Object\_off\_Carriageway VARCHAR(255),

Journey\_Purpose\_of\_Driver VARCHAR(255),

Junction\_Location VARCHAR(255),

make VARCHAR(255),

model VARCHAR(255),

Skidding\_and\_Overturning VARCHAR(255),

Towing\_and\_Articulation VARCHAR(255),

Vehicle\_Leaving\_Carriageway VARCHAR(255),

Vehicle\_Location\_Restricted\_Lane VARCHAR(255),

Vehicle\_Manoeuvre VARCHAR(255),

Vehicle\_Type VARCHAR(255),

Was\_Vehicle\_Left\_Hand\_Drive VARCHAR(255),

X1st\_Point\_of\_Impact VARCHAR(255),

Age\_Band\_of\_Driver VARCHAR(255),

Driver\_Home\_Area\_Type VARCHAR(255),

Propulsion\_Code VARCHAR(255),

Sex\_of\_Driver VARCHAR(255),

Vehicle\_Reference INT

);

CREATE TABLE Fact\_Details (

Fact\_ID INT IDENTITY(1,1) PRIMARY KEY,

Accident\_Index VARCHAR(500),

Accident\_ID INT,

Date\_ID INT,

Road\_ID INT,

Vehicle\_ID INT,

Number\_of\_Casualties INT,

Number\_of\_Vehicles INT,

Speed\_limit INT,

Age\_of\_vehicle INT,

Driver\_IMD\_Decile INT,

Engine\_Capacity\_CC INT,

CONSTRAINT FK\_Fact\_Details\_AccidentTimeDim FOREIGN KEY (Date\_ID)

REFERENCES AccidentTimeDim(Date\_ID)

ON DELETE NO ACTION,

CONSTRAINT FK\_Fact\_Details\_AccidentDetailsDim FOREIGN KEY (Accident\_ID)

REFERENCES AccidentDetailsDim(Accident\_ID)

ON DELETE NO ACTION,

CONSTRAINT FK\_Fact\_Details\_AccidentRoadDetailsDim FOREIGN KEY (Road\_ID)

REFERENCES AccidentRoadDetailsDim(Road\_ID)

ON DELETE NO ACTION,

CONSTRAINT FK\_Fact\_Details\_VehicleDetailsDim FOREIGN KEY (Vehicle\_ID)

REFERENCES VehicleDetailsDim(Vehicle\_ID)

ON DELETE NO ACTION

);

END;

GO

EXEC [dbo].[CREATE\_TABLES];

-> **Drop Table:**

CREATE PROCEDURE [dbo].[DROP\_TABLES]

AS

BEGIN

IF EXISTS (SELECT \* FROM sys.objects WHERE object\_id = OBJECT\_ID(N'[dbo].[Fact\_Details]') AND type in (N'U'))

DROP TABLE Fact\_Details;

IF EXISTS (SELECT \* FROM sys.objects WHERE object\_id = OBJECT\_ID(N'[dbo].[VehicleDetailsDim]') AND type in (N'U'))

DROP TABLE VehicleDetailsDim;

IF EXISTS (SELECT \* FROM sys.objects WHERE object\_id = OBJECT\_ID(N'[dbo].[AccidentTimeDim]') AND type in (N'U'))

DROP TABLE AccidentTimeDim;

IF EXISTS (SELECT \* FROM sys.objects WHERE object\_id = OBJECT\_ID(N'[dbo].[AccidentRoadDetailsDim]') AND type in (N'U'))

DROP TABLE AccidentRoadDetailsDim;

IF EXISTS (SELECT \* FROM sys.objects WHERE object\_id = OBJECT\_ID(N'[dbo].[AccidentDetailsDim]') AND type in (N'U'))

DROP TABLE AccidentDetailsDim;

END;

GO

EXEC [dbo].[DROP\_TABLES];

Design and implement a dimensional OLTP data warehouse. The dimensional warehouse will contain data that has been extracted, transformed, and loaded from the OLTP database. You will develop 5 extended SQL queries that showcase the different OLAP operations, such as rollup and drill-down.

**Query 1:**

**Title: Summary of Casualties by Severity and Year(Rollup Query)**

This SQL query analyzes total casualties from accidents by severity and year, using data joined from Fact\_Details and AccidentDetailsDim on Accident\_ID. It employs the SUM() function to aggregate casualties, and the ROLLUP function in the GROUP BY clause to organize data hierarchically, allowing for the analysis of each severity and year combination as well as their subtotals and grand totals. This setup is effective for tracking casualty trends over time across different levels of accident severity.

**What is the total number of casualties for each severity level of accidents, summarized annually and cumulatively?**

SELECT

ADDim.Accident\_Severity,

ADDim.Year,

SUM(FD.Number\_of\_Casualties) AS TotalCasualties

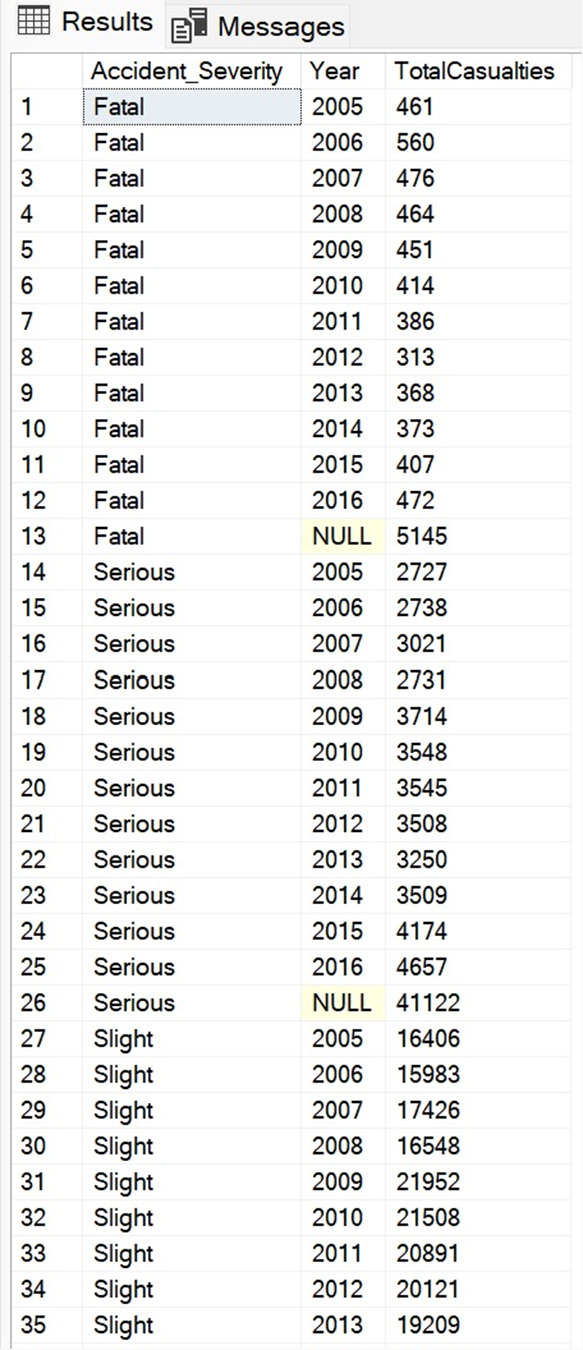
FROM

Fact\_Details FD

JOIN

AccidentDetailsDim ADDim ON FD.Accident\_ID = ADDim.Accident\_ID

GROUP BY ROLLUP (ADDim.Accident\_Severity, ADDim.Year);



**Query 2:**

**Title: Analysis of Car Accident Impacts by Make: Casualties and Vehicles Involved(Drill Down)**

This SQL query extracts and formats data about cars involved in accidents by joining Fact\_Details and VehicleDetailsDim on Vehicle\_ID. It filters for records where the Vehicle\_Type is 'Car', then groups the results by Vehicle\_Type and Make. Using the SUM() function, it calculates the total number of casualties and vehicles involved, formatted with commas for easier reading. The results offer insights into the impact of different car makes on accident severity and frequency.

**How many casualties and vehicles are involved in car accidents, categorized by the make of the car?**

SELECT

V.Vehicle\_Type,

V.Make,

FORMAT(SUM(F.Number\_of\_Casualties), 'N0') AS Total\_Casualties,

FORMAT(SUM(F.Number\_of\_Vehicles), 'N0') AS Total\_Vehicles

FROM

Fact\_Details F

JOIN

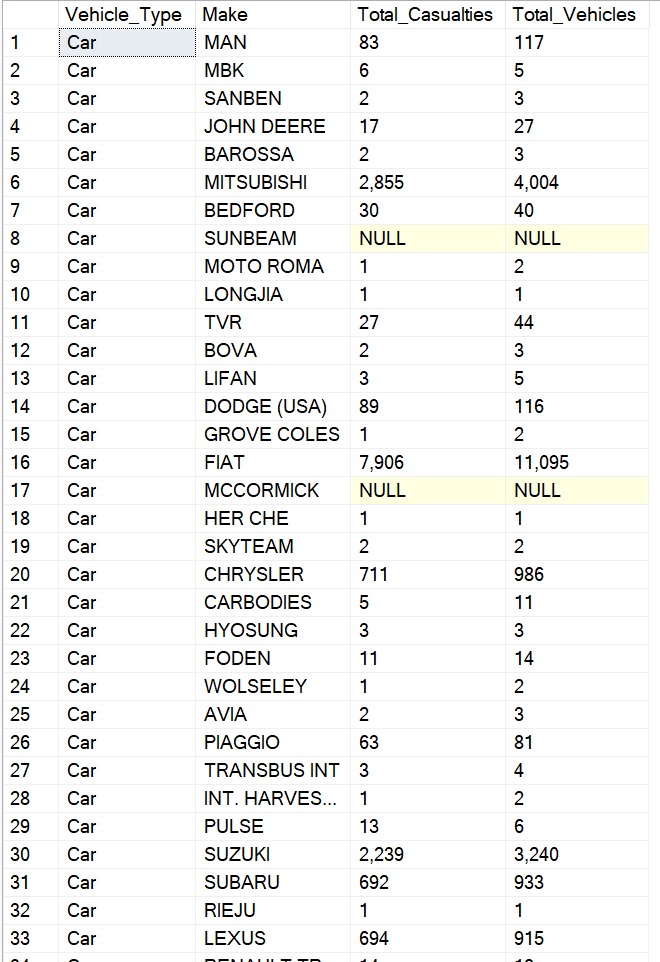
VehicleDetailsDim V ON F.Vehicle\_ID = V.Vehicle\_ID

WHERE

V.Vehicle\_Type = 'Car'

GROUP BY

V.Vehicle\_Type, V.Make;



**Query 3:**

**Title:** A**ccident Timing Details for Mondays -Top N Analysis**

This SQL query retrieves data from the AccidentTimeDim table, specifically focusing on accidents that occurred on Mondays. It selects the Date, Day\_of\_Week, and Time of each accident, providing detailed information about when these incidents took place. By filtering on Day\_of\_Week = 'Monday', the query efficiently isolates and displays only the records related to accidents on this particular day of the week, which can be useful for analyzing accident trends or patterns specific to Mondays.

**What are the specific dates and times when accidents occurred on Mondays?**

SELECT

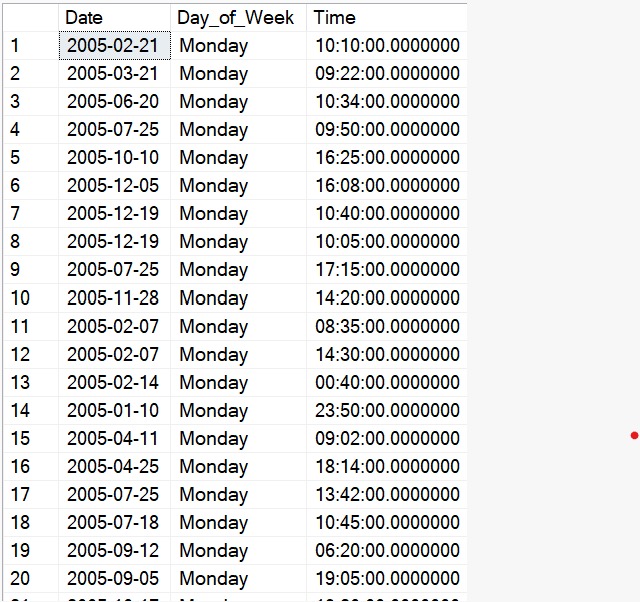
Date,

Day\_of\_Week,

Time

FROM AccidentTimeDim

WHERE Day\_of\_Week = 'Monday';



**Query 4:**

**Title:** **Analysis of Urban Area Accidents: Total Incidents and Average Casualties -SLICE**

This SQL query analyzes accident data specifically in urban areas by joining the Fact\_Details and AccidentDetailsDim tables on the Accident\_ID. It focuses on urban areas by filtering with ADDim.Urban\_or\_Rural\_Area = 'Urban'. The query calculates the total number of accidents (COUNT(FD.Fact\_ID) AS TotalAccidents) and the average number of casualties (AVG(FD.Number\_of\_Casualties) AS AverageCasualties) for urban settings. Results are grouped by Urban\_or\_Rural\_Area and ordered by the total number of accidents in descending order, which prioritizes areas with higher incidences of accidents, providing a clear view of urban accident trends and severity.

**How many total accidents occur in urban areas, and what is the average number of casualties per accident in these areas?**

SELECT

ADDim.Urban\_or\_Rural\_Area,

COUNT(FD.Fact\_ID) AS TotalAccidents,

AVG(FD.Number\_of\_Casualties) AS AverageCasualties

FROM

Fact\_Details FD

JOIN

AccidentDetailsDim ADDim ON FD.Accident\_ID = ADDim.Accident\_ID

WHERE

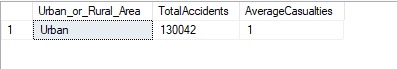
ADDim.Urban\_or\_Rural\_Area = 'Urban' -- Slicing to focus on Urban areas

GROUP BY

ADDim.Urban\_or\_Rural\_Area

ORDER BY

TotalAccidents DESC;



**QUERY 5:**

**Title: Top 10 Most Common Vehicles by Make, Model, and Type – DRILL DOWN**

This SQL query selects the top 10 most frequent combinations of vehicle make, model, and type from the VehicleDetailsDim table. It uses the COUNT(\*) function to calculate the total number of entries (vehicles) for each combination, which are then grouped by make, model, and Vehicle\_Type. The results are intended to identify the most common vehicles in the dataset, providing insights into vehicle distribution and popularity.

**What are the top 10 most common vehicle combinations of make, model, and type?**

SELECT TOP 10

make,

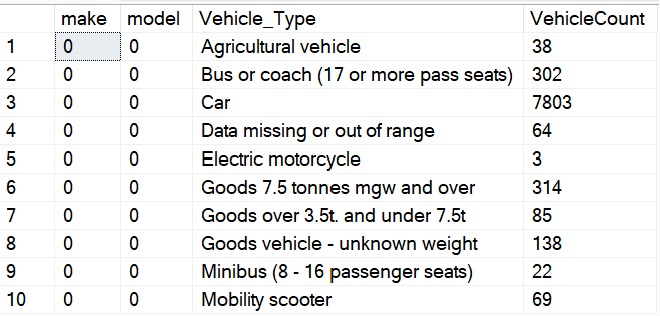
model,

Vehicle\_Type,

COUNT(\*) AS VehicleCount

FROM VehicleDetailsDim

GROUP BY make, model, Vehicle\_Type;



**QUERY 6:**

**Title:** **Vehicle Count Analysis by Type with Grand Total**

This SQL query from the VehicleDetailsDim table uses the COALESCE function to ensure that all vehicle types are included in the output, substituting 'All Vehicle Types' for any null values in the Vehicle\_Type column. It counts the total number of vehicles (COUNT(\*)) and groups the results using the ROLLUP function to aggregate data by Vehicle\_Type and also provide a grand total across all vehicle types. This approach offers a hierarchical view of vehicle counts, facilitating comprehensive analysis across different types or the entire dataset.

**How many vehicles are there for each vehicle type, including a total count for all vehicle types combined?**

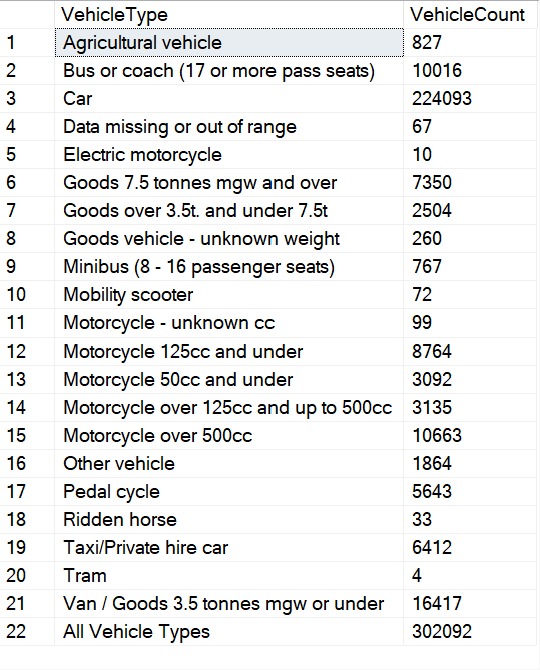
SELECT

COALESCE(Vehicle\_Type, 'All Vehicle Types') AS VehicleType,

COUNT(\*) AS VehicleCount

FROM VehicleDetailsDim

GROUP BY ROLLUP (Vehicle\_Type);



**TASK 3:**

Design and implement a NoSQL database in MongoDB containing the data from the extracted csv files in 1. You will then develop 3 NoSQL queries and two aggregations. The NoSQL queries can reflect the SQL queries in Step 1, and the aggregations can reflect the OLAP queries in Step 2, but this is not mandatory.

**QUERY 1-** What are the details of the most recent severe accidents that occurred in Scotland, including the accident index, date, location, number of casualties, and time of the accident? (NoSQL)

This MongoDB query retrieves records from the Accidents collection where the accident severity is marked as "Serious" and the accident occurred in Scotland. It selects specific fields such as Accident\_Index, Date, Year, and others while excluding the default \_id field. The results are sorted in descending order by date, presenting the most recent serious accidents in Scotland first. This setup is particularly useful for timely analysis and reporting on recent severe accidents in specific regions.

**SQL query:** db.Accidents.find({Accident\_Severity:"Serious", InScotland:"Yes"},{"\_id":0,"Accident\_Index":1,"Accident\_Severity":1,"Date":1,"Year":1,"Local\_Authority\_(District)":1,"InScotland": 1,"Number\_of\_Casualties":1, "\_id" : 0}}).sort({"Date":-1});

**OUTPUT:**

A screen shot of a computer

Description automatically generated

**QUERY 2-** What are the details of all accidents where the weather was 'Raining no high winds' and there were more than three casualties, including the accident index, date, weather conditions, number of casualties, and time of the accident, sorted by the most recent date?" (NoSQL)

This MongoDB query searches the Accidents collection for accidents occurring during "Raining no high winds" weather conditions with more than three casualties. It selects fields such as Accident\_Index, Date, Weather\_Conditions, Number\_of\_Casualties, and Time, excluding the default \_id field. The results are sorted by date in descending order, ensuring that the most recent relevant accidents are listed first, useful for analyzing the impact of specific weather conditions on accident severity.

**SQL query:** db.Accidents.find({Weather\_Conditions:"Raining no high winds",Number\_of\_Casualties:{$gt:3}},{Accident\_Index:1,Date:1,Weather\_Conditions:1,Number\_of\_Casualties:1,Time:1,\_id:0}).sort({Date:-1});

**Output:**

A black screen with white lines

Description automatically generated

**QUERY 3-** What are the details of accidents involving drivers over the age of 75, including the year, date of the accident, accident index, drivers' sex, and their home area type? (NoSQL)

This MongoDB query retrieves records from the Vehicles collection, focusing on vehicles driven by individuals either male or female in the "Over 75" age band. It selects specific fields like Year, Sex\_of\_Driver, Age\_Band\_of\_Driver, Accident\_Index, Date, and Driver\_Home\_Area\_Type, while excluding the \_id field. This setup aids in demographic studies of accidents, particularly analyzing the impact and frequency of incidents involving elderly drivers.

**SQL query:** db.Vehicles.find({"Sex\_of\_Driver":{"$in":["Male", "Female"]},"Age\_Band\_of\_Driver":"Over 75"},{"Year":1,"Sex\_of\_Driver":1,"Age\_Band\_of\_Driver":1,"Accident\_Index":1,"Date":1,"Driver\_Home\_Area\_Type": 1,"\_id":0});

**Output:**

A screen shot of a computer

Description automatically generated

**QUERY 4-** What is the distribution of vehicle types and age bands of drivers involved in accidents, along with the average engine capacity of these vehicles?

This MongoDB aggregation pipeline groups records in a collection based on combinations of Vehicle\_Type, Age\_Band\_of\_Driver, and Accident\_Index. It calculates the total number of vehicles (Number\_of\_Vehicles) and the average engine capacity (Average\_Engine\_Capacity) for each group. The results are then sorted in descending order by the number of vehicles. Finally, the project stage formats the output by excluding the default \_id field and explicitly listing the desired fields, ensuring the data is clear and focused on key attributes for analysis.

**SQL query:** [{"$group": {"\_id": {"Vehicle\_Type": "$Vehicle\_Type","Age\_Band\_of\_Driver": "$Age\_Band\_of\_Driver","Accident\_Index": "$Accident\_Index"},"Number\_of\_Vehicles": { "$sum": 1 },"Average\_Engine\_Capacity": { "$avg": "$Engine\_Capacity" }}},{"$sort": { "number\_of\_vehicles": -1 }},{"$project": {"\_id": 0,"Accident\_Index": "$\_id.Accident\_Index","Vehicle\_Type": "$\_id.Vehicle\_Type","Age\_Band\_of\_Driver": "$\_id.Age\_Band\_of\_Driver","Number\_of\_Vehicles": 1,"Average\_Engine\_Capacity": 1,}}]

**Output:**

A screen shot of a computer

Description automatically generated

**QUERY 5-** How do the total number of accidents, average number of casualties, and distribution of police attendance vary across different road classes and years?

This MongoDB aggregation pipeline groups accident records by whether police attended, road class, and year, calculating the total number of accidents and the average number of casualties for each group. The results are sorted in descending order based on the total number of accidents to prioritize the most frequent scenarios. The final output is formatted to include only the specified attributes, effectively summarizing the impact and circumstances of accidents across different conditions.  
  
**SQL Query:** {"$group":{"\_id": {"Did\_Police\_Attend": "$Did\_Police\_Attend","Road\_Class": "$Road\_Class","Year": "$Year"},"Total\_Accidents": {"$sum": 1},"Average\_Number\_of\_Casualties": { "$avg": "$Number\_of\_Casualties" },}},{"$sort": { "Total\_Accidents": -1 }},{"$project": { "\_id": 0,"Did\_Police\_Attend": "$\_id.Did\_Police\_Attend","Road\_Class": "$\_id.Road\_Class","Year": "$\_id.Year","Total\_Accidents": 1,"Average\_Number\_of\_Casualties": 1,"Accident\_Indices": 1}}

**Output:**

A screenshot of a computer

Description automatically generated