Università Di PISA

Department of Computer Science

Master's in data science & business informatics

Laboratory of Data Science

Part 1

Submitted To:

Prof. Anna Monreale

Prof. Roberto Pellungrini

Asst. Cristiano Lan

Submitted By:

Amir Hassan (683185)

Attia Maqbool (682009)

Group_ID_07

Academic Year 2023/24

CONTEN

Assignment:1 *Data Understanding*

In the First Assignment we did the analysis of the three datasets—**Crashes**, **People**, and **Vehicles**—revealed several key insights. The **Crashes dataset** contains information about accident locations, including latitude and longitude, which we validated to ensure accuracy. Missing data was observed in multiple columns across all datasets, with some visualized through heatmaps. The **People dataset** provides details about individuals involved in accidents, while the **Vehicles dataset** includes information about the vehicles involved. A notable relationship among the datasets is the presence of shared keys, such as incident or case identifiers, which enable linking and cross-referencing records across the three datasets.

Interesting findings include patterns in accident locations, potential outliers in numerical data, and variations in the completeness of data among the datasets. Visualizations like histograms revealed distributions of attributes such as vehicle types and demographic factors. These datasets collectively offer a comprehensive view of accidents, allowing for detailed analysis of causes, demographics, and vehicle involvement. Further cleaning and alignment of data can enhance their utility for cross-analysis and insights.

Assignment: 2 *Data Cleaning*

The data cleaning process involved handling missing values across three datasets: crashes, people, and vehicles data. To address missing values effectively, we used a combination of statistical imputation techniques tailored to the nature of each column. For numerical fields like **AGE**, **LATITUDE**, and **VEHICLE_YEAR**, missing values were filled using the **mean** or **median**, ensuring that the central tendency of the data remained intact. This approach is suitable for numerical data as it minimizes the impact of missing entries without distorting the overall distribution. For categorical fields such as **MOST_SEVERE_INJURY**, **MAKE**, and **SEX**, the **mode** was used to fill missing values, as it ensures the most common category represents missing entries, maintaining consistency with the majority of the data.

Other than the above, grouped statistics were utilized for more context-specific imputation, such as filling **DAMAGE** values based on **INJURY_CLASSIFICATION** using grouped medians. This strategy provided meaningful and realistic replacements for missing values, especially when related categorical data was available. Placeholder values like "Unknown" were used sparingly for non-recoverable missing entries, such as missing IDs, to retain data integrity. The cleaning techniques were chosen based on the nature of the data and its distribution, ensuring a balance between accuracy and bias prevention. After cleaning, the data was validated to confirm that missing values were completely addressed while preserving the original trends and patterns.

Assignment: 3 *DW* Schema

In this step, we first create tables using SSMS, referred in the database schema provided (see Figure 1).

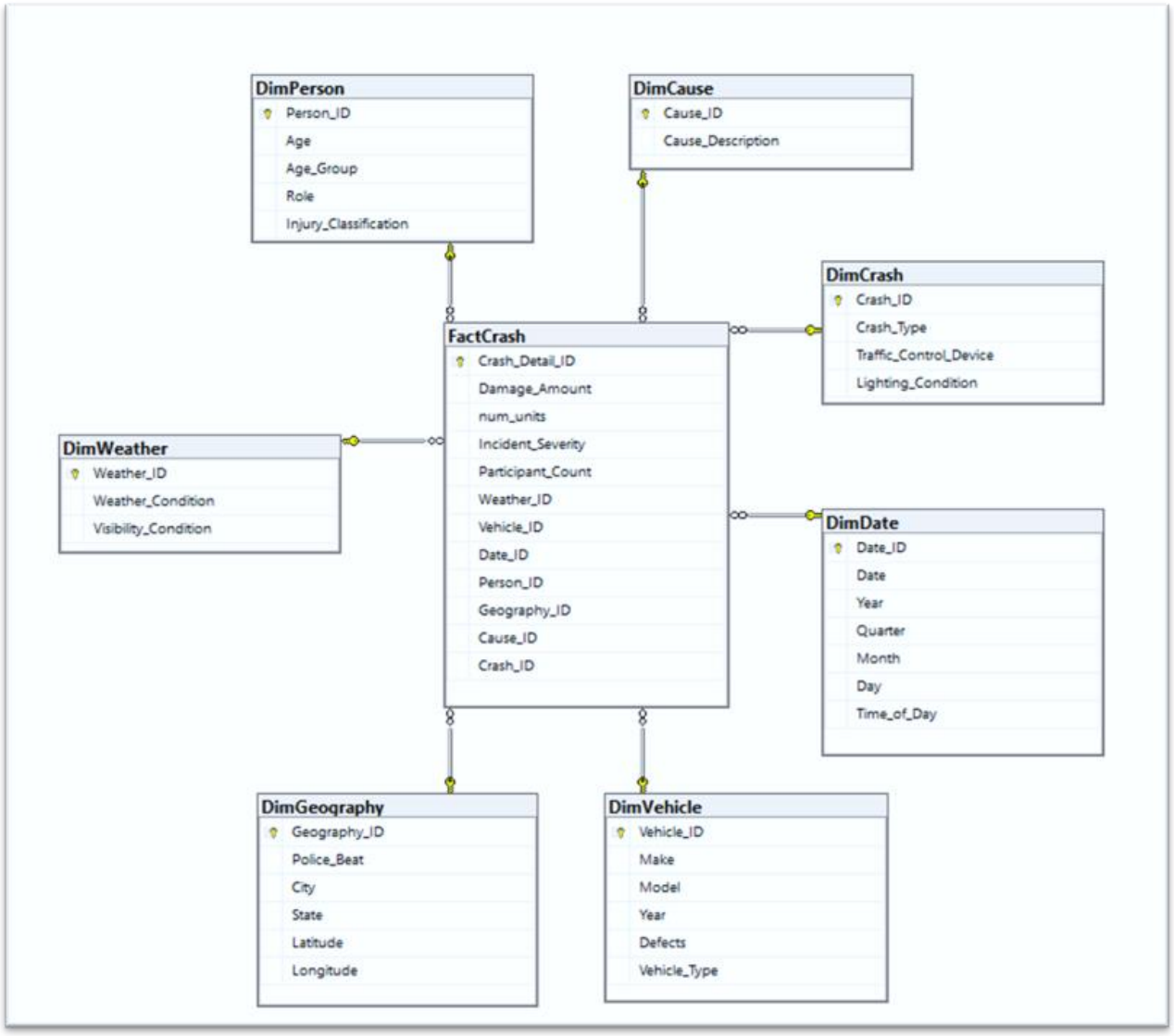

Figure 1: Database Schema

Assignment: 4 *Data preparation*

Before populating the database, we created separate files for all the tables. We also made sure that, in each file, the id columns do not have duplicate values. Otherwise, when populating the database, it would generate an error because it would be violating the uniqueness constraint of the primary key. We perform all these steps by calling the python scripts according to schema tables csv files and then write data accordingly from our cleaned datasets (Crashes, People. Vehicles).

Assignment: 5 *Data* uploading with python

To populate the **DB**, we utilized Python with the pyodbc library to establish a connection with the SQL Server database and upload the prepared data efficiently. The CSV files from Assignment 4 were processed using Python to read the data, validate fields, and upload it into the database.

We implemented the executemany() function to optimize batch data insertion, significantly reducing execution time for large datasets. Additionally, for tables containing identity columns, we leveraged the IDENTITY_INSERT option to manage explicit key insertion while preserving the table schema's integrity. This systematic and performance-focused approach ensured accurate and efficient population of the database.

Assignment: 6 *Data uploading*

***Assignment 6**: Data uploading (10%)*

The objective of this task was to duplicate the existing database tables while excluding their records, rename the duplicated tables with the suffix _SSIS, and then create an SSIS (SQL Server Integration Services) project to populate these new tables with 10% of the data from the original tables.

Steps Performed

1. Duplicating Tables Without Records

We duplicated each table from the original schema (e.g., *DimDate*, *FactCrash*) into a new schema with the same structure but without data. The duplicated tables were renamed with the _SSIS suffix (e.g., *DimDate_SSIS*, *FactCrash_SSIS*). The process involved the following steps:

1. **Extract the Table Definition Script:**
    - In SQL Server Management Studio (SSMS), for each table:
        - Right-click the table.
        - Select Script Table as > CREATE To > New Query Editor Window.
    - This generated the CREATE TABLE script for the table.
2. **Edit the Table Name:**
    - Renamed the table in the CREATE TABLE script by appending _SSIS to the original table name.
        - Example: Original table: DimDate New table: DimDate_SSIS.
3. **Remove Foreign Key Constraints:**
    - Foreign key constraints were removed from the duplicated tables to prevent dependency issues during data insertion via SSIS.
4. **Verify Column Definitions:**

- Ensured all columns in the new _SSIS tables matched the data types and constraints (e.g., NOT NULL) of the original tables.
- Retained only primary key constraints for uniqueness.

5. **Execute the Scripts:**
   - Ran the modified scripts in SSMS to create the new _SSIS tables.
   - Verified that the new tables were created successfully in the database without any data.

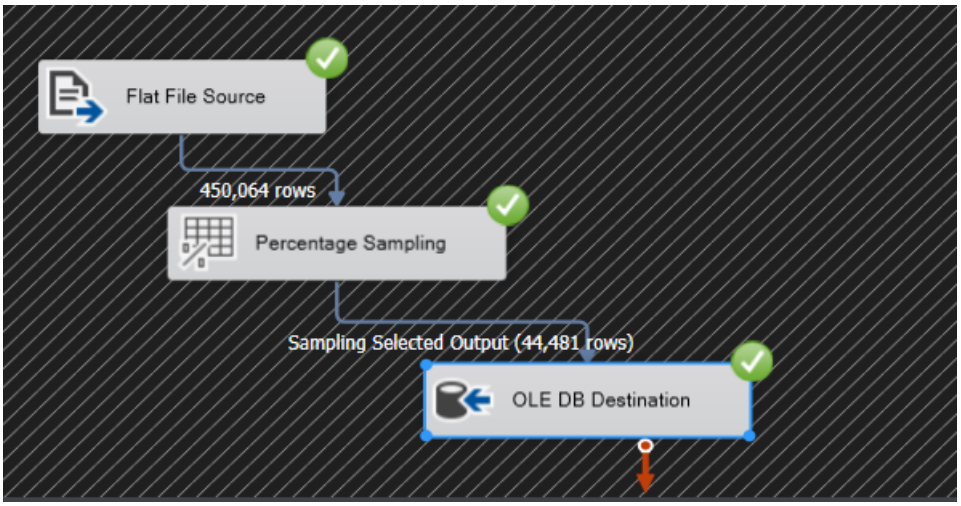

Figure 2: SSIS Implementation Snapshot

SSIS IMPLEMENTATION VS STUDIO ASSIGNMENTS

This part of the project includes performing four assignments on the existing database created in the previous stage using SQL Server Integration Services (SSIS) with computation on the client side.

To run the project, perform the following steps:

1. Go to SSIS_Implementation_Assignments folder

2. Open the solution file *Assignment6c_SSIS.sln* for tasks **Assignment6a**, **Assignment7a** and **Assignment8a** and task 9a.sln for **Assignment9a** in Visual Studio.

2. Open the corresponding packages in the Solution Explorer window on the right. The packages are named *Assignment6a.dtsx*, *Assignment7a.dtsx*, *Assignment8a.dtsx* and *Assignment9a.dtsx* with respect to each task.

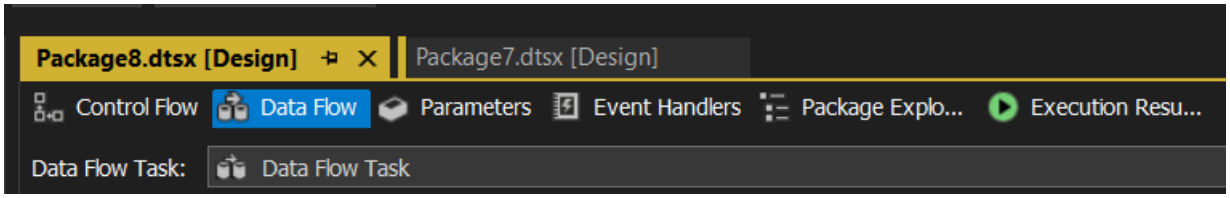

Figure 3: Basic Flow

4. After changing all the file paths according to your system, start each package one by one and view the output file when it opens.

Let us explain the Control Flow and Data Flow now.

Each package has a similar control flow (as shown below) with two tasks, a Data Flow task, and an Execute Process task.

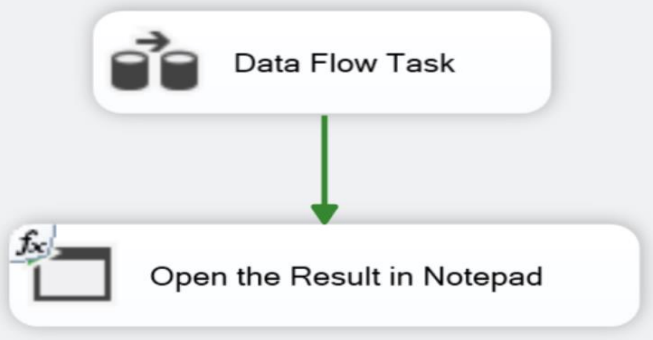

Figure 4: Control Flow of Each Assignment

Assignment 6a – For every year, show all participants ordered by the total number of crashes

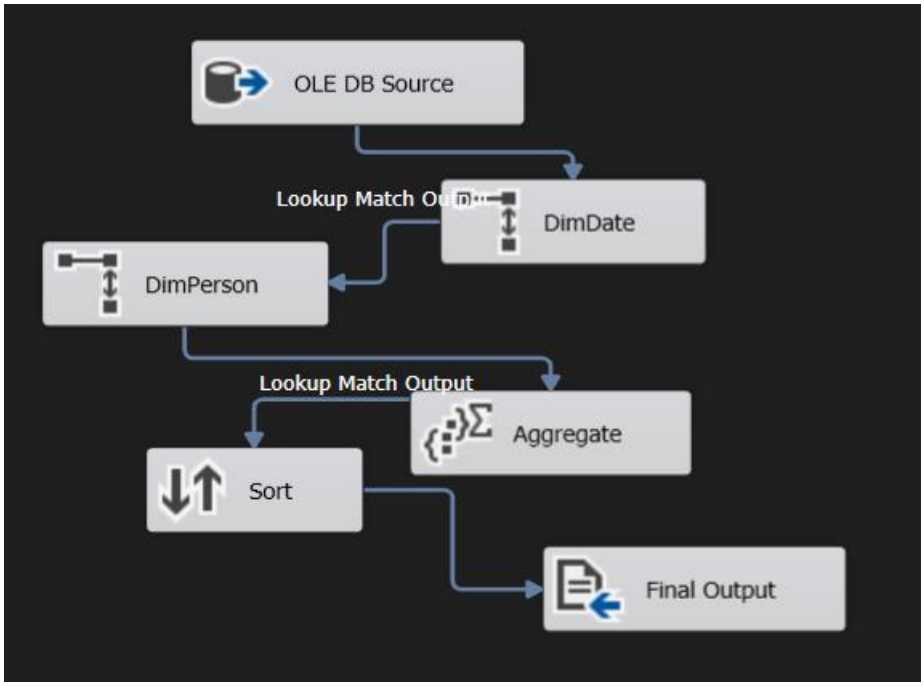

Figure 5: Data Flow of Assignment 6a

Workflow Steps:

1. **OLE DB Source**:
    - Extracts FactCrash data from Database

Lookup with DimDate:

- Matches the Date_ID from the source data with the DimDate table to retrieve the corresponding year.

2. **Lookup with DimPerson**:
   - Matches the Person_ID from the source data with the DimPerson table to retrieve participant details.
3. **Aggregate Transformation**:
   - Groups the data by:
     - Year (from DimDate).
     - Person_ID (from DimPerson).
   - Counts the total crashes (Crash_ID) for each participant and year.
4. **Sort Transformation**:
   - Sorts the data by the total number of crashes (calculated in the Aggregate transformation) in descending order.
5. **Final Output**:
   - Written the processed, aggregated, and sorted data to the destination Final Output file.

Year	Person_ID	Total_Crashes
2014	O70308	1
2014	O12093	1
2014	O30750	1
2014	O481321	1
2014	O70309	1
2014	P108071	1
2014	O30751	1
2014	O24495	1
2014	O24496	1
2015	O2369	2
2015	O18874	2
2015	O18873	2
2015	O17742	2

Figure 6: Output 6a

Assignment: 7a: For every police beat, compute the *day-night crash index*, defined as the ratio between the number of vehicles (NUM UNITS) involved in an incident between 9 pm and 8 am, and the number of vehicles involved in an incident between 8 am and 9 pm.

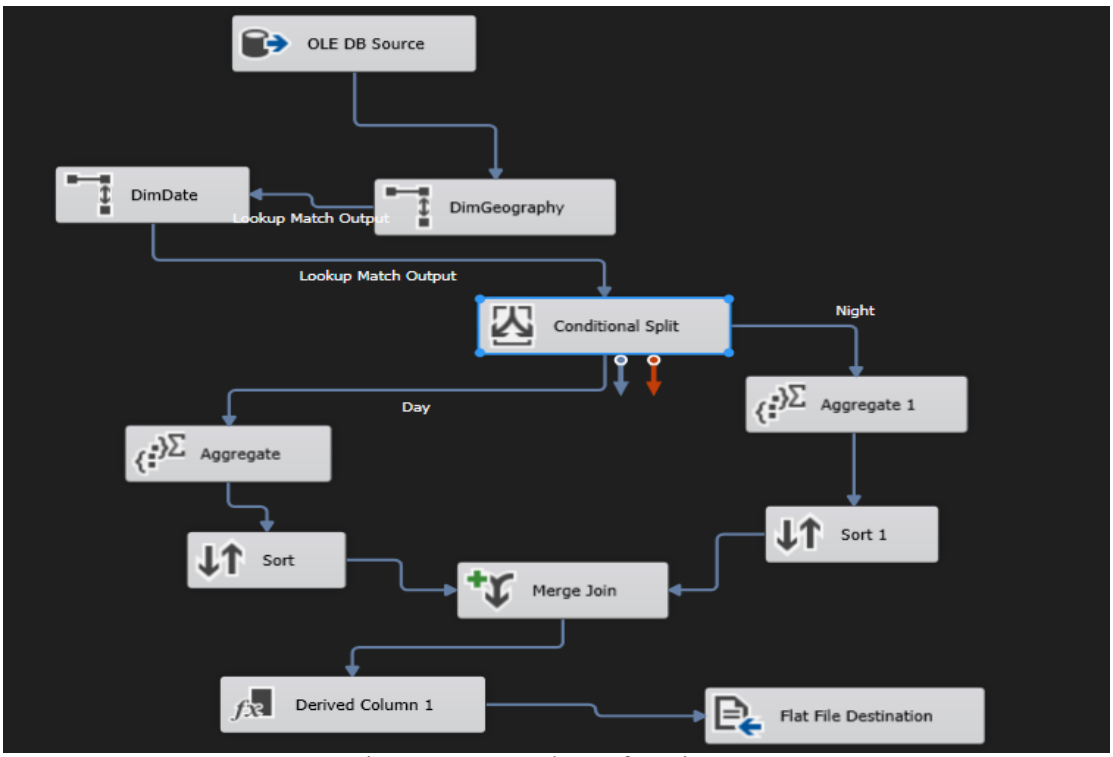

Figure 7: Data Flow of Assignment 7a

Workflow Review:

1. **OLE DB Source**:
    - Extracts the FactCrash data from the database.
2. **Lookups (DimDate and DimGeography)**:
    - Joins the fact data with the DimDate and DimGeography dimensions to add relevant attributes such as Time_of_Day and Police_Beat.
3. **Conditional Split**:
    - Splits the data into two streams: **Day** and **Night**, based on the Time_of_Day conditions.

    By Using below conditions:

    Day: SUBSTRING(Time_of_Day, 1, 2) >= "08" && SUBSTRING(Time_of_Day, 1, 2) < "21"
    Night: SUBSTRING(Time_of_Day, 1, 2) >= "21" || SUBSTRING(Time_of_Day, 1, 2) < "08"

4. **Aggregate Transformations**:
    - **Day Aggregate**: Groups by Police_Beat and sums up num_units for incidents in the Day_Period.
    - **Night Aggregate**: Groups by Police_Beat and sums up num_units for incidents at Night_Period.
5. **Sort Transformations**:
    - Prepares the aggregated data for a **Merge Join** by sorting it based on Police_Beat.
6. **Merge Join**:
    - Combines the results from the **Day** and **Night** streams using Police_Beat as the key.
7. **Derived Column**:
    - Computes the **Day-Night Crash Index** using:

      [Night_Sum] / ([Day_Sum] == 0 ? 1 : [Day_Sum])

        - We ensures there is no division by zero.

8. **Flat File Destination**:
    - Outputs the final result, including Police_Beat, Day_Period, Night_Period, and the **Day-Night_CrashIndex**, to a flat file.

Police_Beat	Day_Period	Night_Period	Day_Night_CrashIndex
1012	4	2476	619
1013	4	5074	1268
1014	10	2215	221
1022	8	3766	470
1031	4	5370	1342
1032	8	2630	328
1033	4	4136	1034
1034	4	6373	1593
111	10	6844	684
1111	6	4592	765
1112	8	3524	440
1113	2	4849	2424
1114	10	1911	191
1115	12	2828	235
112	10	5187	518
1125	2	1224	612
113	2	3425	1712
1131	12	4350	362
1132	2	3104	1552

Figure 8: Output 7a

Assignment 8a – For each quarter, weather condition, and beat, show the average ratio of people under 21 years old to people over 21 years old involved in crashes.

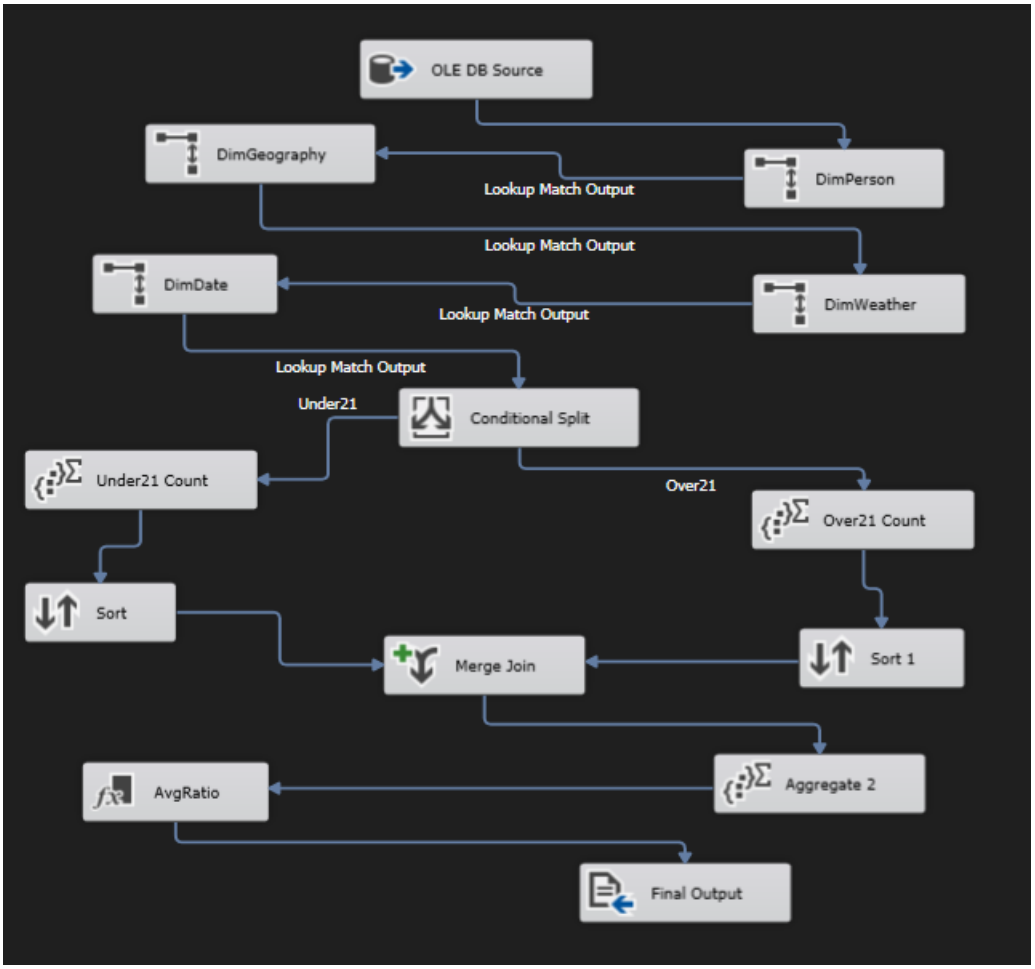

Figure 9: Data Flow of Assignment 8a

Workflow Steps:

1. **Data Source (OLE DB Source):**
    - Extracts FactCrash data from Database.
2. **Lookups:**
    - Joins the data with dimension tables (DimGeography, DimDate, DimPerson, and DimWeather) to get attributes (e.g., Age, Quarter, Weather_Condition, and Police_Beat).
3. **Conditional Split:**
    - Splits the data into two streams based on age:
        - Under21: Data for individuals under 21 years old.
        - Over21: Data for individuals over 21 years old.
4. **Aggregations (Under21 Count and Over21 Count):**
    - Calculates the count of individuals under 21 and over 21 for each quarter, weather condition, and beat.
5. **Sorting (Sort and Sort 1):**
    - Sorts both streams of data for merging purposes.
6. **Merge Join:**
    - Combines the Under21 and Over21 data streams based on common keys (e.g., quarter, weather condition, and beat).
7. **Additional Aggregation (AvgRatio):**
    - Computes the average ratio of Under21 Count to Over21 Count for each group (quarter, weather condition, and beat).
8. **Final Output:**
    - Loaded the processed data into the Final Output file.

Quarter	Weather_Condition	Police_Beat	Under21	Over21	AgeRatio
4	CLEAR	621	54188	162966	0.332511076
3	CLEAR	725	36797	135366	0.2718334
4	CLEAR	821	129352	162966	0.793736117
3	CLEAR	825	39171	135366	0.289371039
4	CLEAR	921	80408	162966	0.493403532
3	CLEAR	925	26114	135366	0.192914026
4	CLEAR	1812	19228	162966	0.117987801
3	RAIN	2535	1187	135366	8.77E-03
4	RAIN	1611	27968	162966	0.17161862
3	CLEAR	1811	46293	135366	0.341983955
4	SNOW	2523	3496	162966	0.021452327
1	CLOUDY/OVERCAST	124	1791	103289	1.73E-02

Figure 10: Output 8a

Assignment 9a – Based on the analytical results of the previous queries and the insights gathered during the data understanding phase, define an **interesting** query and answer it using SSIS.

Determine the average crash damage cost for different weather conditions, categorized by time of day (day vs. night) and vehicle type, to identify patterns in cost impact.

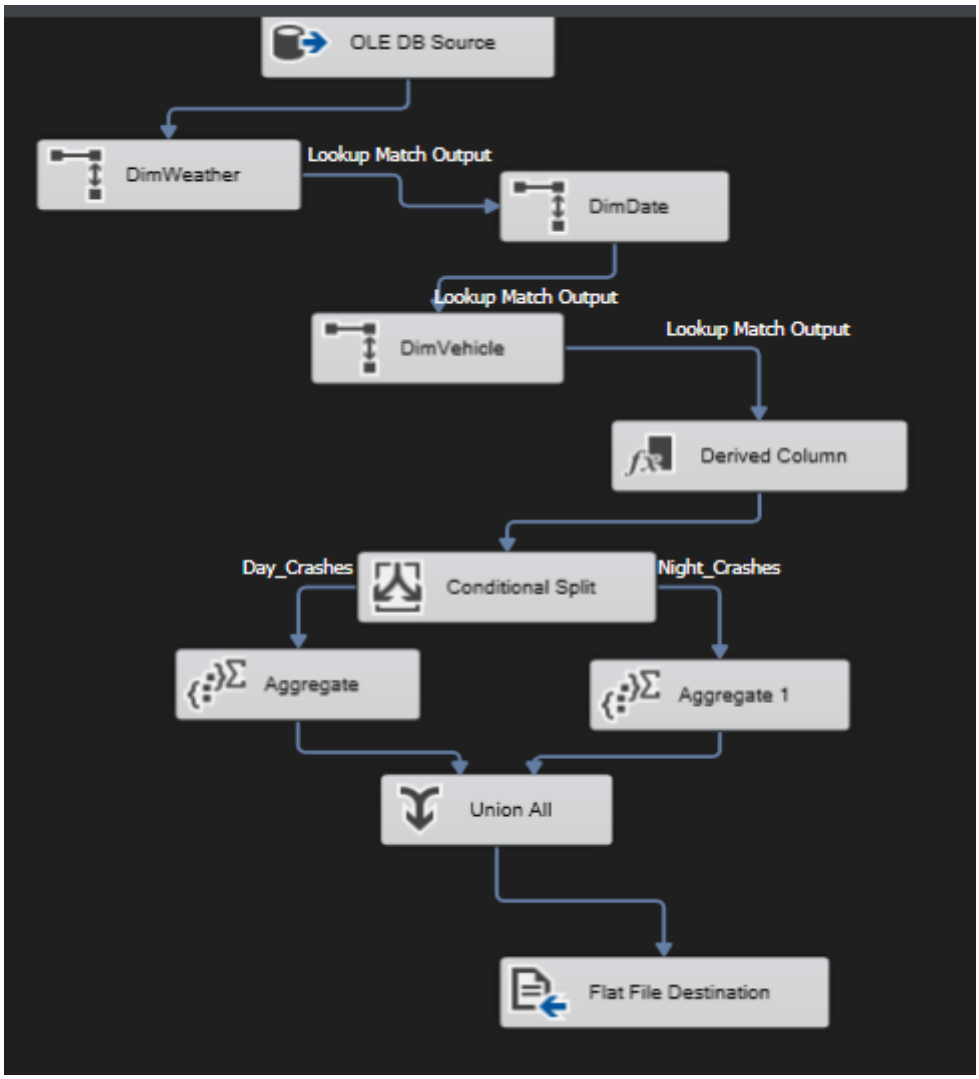

Figure 11: Data Flow of Assignment 9a

Workflow Steps

1. **Data Source (OLE DB Source)**: Extracts crash data from the FactCrash table.
2. **Lookups**: Joins with DimWeather to get Weather_Condition.
    - Joins with DimDate to get Time_of_Day.
    - Joins with DimVehicle to get Vehicle_Type.

3. **Derived Column**: Extracts the hour from Time_of_Day and creates Hour_of_Day.
4. **Conditional Split**: Splits data into:

   Day Crashes: Hour_of_Day between 8 and 20.

   Night Crashes: Hour_of_Day between 21 and 7.
5. **Aggregations**: Calculates SUM(Damage_Amount) grouped by Weather_Condition and Vehicle_Type for both Day and Night Crashes.
6. **Union All**: Merges Day and Night Crashes.
7. **Flat File Destination**: Exports the final data with Weather_Condition, Vehicle_Type, and aggregated Damage_Amount.

Weather_Condition	Vehicle_Type	Damage_Amount
RAIN	ALL-TERRAIN VEHICLE (ATV)	9496
SEVERE CROSS WIND GATE	SPORT UTILITY VEHICLE (SUV)	46938
SNOW	BUS UP TO 15 PASS.	136166
SLEET/HAIL	SPORT UTILITY VEHICLE (SUV)	307832
SLEET/HAIL	UNKNOWN/NA	133993
CLOUDY/OVERCAST	MOTOR DRIVEN CYCLE	8719
CLEAR	BUS OVER 15 PASS.	22201137
CLOUDY/OVERCAST	VAN/MINI-VAN	2896209
CLEAR	UNKNOWN/NA	94169336
CLOUDY/OVERCAST	PICKUP	1456455
SNOW	SNOWMOBILE	4183
RAIN	TRUCK - SINGLE UNIT	2254676
CLEAR	OTHER	13631654
SNOW	AUTOCYCLE	67088
FOG/SMOKE/HAZE	UNKNOWN/NA	216158
CLOUDY/OVERCAST	TRACTOR W/O SEMI-TRAILER	80308
FOG/SMOKE/HAZE	OTHER VEHICLE WITH TRAILER	3662
RAIN	PASSENGER	105227520
SEVERE CROSS WIND GATE	BUS OVER 15 PASS.	3069
UNKNOWN	AUTOCYCLE	143993
RAIN	PICKUP	4125656
FOG/SMOKE/HAZE	TRUCK - SINGLE UNIT	12707
SLEET/HAIL	TRACTOR W/ SEMI-TRAILER	14100
SEVERE CROSS WIND GATE	VAN/MINI-VAN	29241
CLOUDY/OVERCAST	FARM EQUIPMENT	1800
SLEET/HAIL	TRUCK - SINGLE UNIT	25428
CLEAR	ALL-TERRAIN VEHICLE (ATV)	70853
CLOUDY/OVERCAST	ALL-TERRAIN VEHICLE (ATV)	2332
CLOUDY/OVERCAST	AUTOCYCLE	31803
UNKNOWN	TRACTOR W/O SEMI-TRAILER	27854

Figure 12: Output of Assignment 9a