Data Warehouse Development: LAPD Crime Data Analysis

Proof of Concept and Business Vision

The LAPD Crime Dataset was selected for our data warehouse proof of concept due to its importance in public safety and potential for improving law enforcement effectiveness. A data warehouse allows structured organization of crime data by separating attributes like locations, times, and crime types into linked tables, making it easier to filter and aggregate data across different combinations of these attributes. For example, while a traditional system might just show a list of individual crime records, a warehouse structure can quickly show all thefts in a specific area during evening hours or compare crime rates between different neighborhoods over time.

Our data warehouse approach follows principles similar to those discussed in police intelligence literature. Wei et al. (2012, p. 3865) describe the "Information Island phenomenon existing in the current police system and inadequate excavation of data," highlighting why integrated data warehousing solutions are valuable for law enforcement agencies. The LAPD crime data warehouse we've developed addresses objectives that Salcedo-Gonzalez et al. (2020, p. 2) recognize as important in this domain - enabling the analysis of criminal patterns to support "better distribution and management of police resources allocated to crime deterrence, prevention and control."

Similar to Cheng and Williams' (2012, p. 47) assertion that "timely understanding of how criminality emerges and how crime patterns evolve is crucial to anticipating crime," our business vision focuses on creating an analytics platform that transforms raw crime data into actionable insights. Sharma and Dronavalli (2020, p. 1) support this vision by noting that dedicated data warehouses overcome the challenges posed by "the abundance of data sources available, from police records to public databases" that plague traditional database approaches to crime analysis.

Stakeholder Analysis and Requirements

The success of this data warehouse depends on understanding and serving three key stakeholder groups: executive level, investigators, and community stakeholders.

Our executive stakeholders are law enforcement leadership and command staff who need to understand crime trends. They require tools to assess the effectiveness of police initiatives and identify high-crime areas for resource allocation. These insights will help them make decisions about departmental strategy. As focus of their work ,Carnaz et al. (2019, p. 352), "police investigators to identify and correlate interesting extracted entities." to support strategic decisions. Similarly, Sihombing (2022, p. 1127) emphasizes that "data warehouse plays an essential role in helping the alignment between strategies, processes, and technologies that can increase an organization's competitiveness," which directly supports our focus on strategic executive decision-making.

Investigators and operational staff form our second stakeholder group. These personnel need access to crime data to support their daily work. They require location-based information and the ability to analyze crimes by type and frequency. Michael and Ahirao (2020, p. 1) support this approach, noting that "companies can use ETL tool to analyze their business data and answer complex business queries which Transactional databases cannot answer." In the law enforcement context, Garfias Royo et al.

(2020, p. 2) explain that analysis platforms help departments identify "situational factors in high crime areas that might contribute to the facilitation of those crimes."

Our community stakeholders, including residents and business owners, represent the third group. They need clear and accessible information about crimes in their neighborhoods. As Reehl and Sharma (2018, p. 1) point out, "data visualization is extremely effective in communicating the criminal data to the public, police, and security officers," highlighting the importance of making warehouse outputs accessible to community members. Zhang et al. (2006, p. 518) further reinforce this by emphasizing how modern visualization techniques can "maintain up-to-date visualizations for public consumption," which is crucial for keeping community members informed about local safety conditions.

Business Insights and Visualization

For executive stakeholders, we will implement visualization tools that clearly display crime density distribution across Los Angeles and track overall trends over time. This enables identification of high-crime areas and allows leadership to assess the effectiveness of their initiatives. Salcedo-Gonzalez et al. (2020) demonstrates the value of analyzing criminal data by concentration, zone, time slot, and date, which supports our visualization approach. This is further reinforced by Cheng and Williams (2012, p. 49), who show how temporal analysis can effectively reveal patterns in criminal activity.

For investigators and operational staff, we will develop geospatial visualizations that display crime locations and their frequencies on city maps. This allows them to identify clusters of similar crimes and analyze patterns in criminal activity across different areas. This approach is consistent with research by Sharma and Dronavalli (2020), who found that geographical crime mapping can "supply insight into factors affecting crime that will help [law enforcement] deploy resources and help in their decision-making process." Our implementation provides actionable intelligence for frontline personnel to identify and respond to emerging crime patterns more effectively.

For community stakeholders, we will create intuitive visual representations displaying the most common types of crimes in each neighborhood. This helps residents understand local crime patterns while maintaining appropriate data privacy.

1. DESIGN AND EXPLANATION:

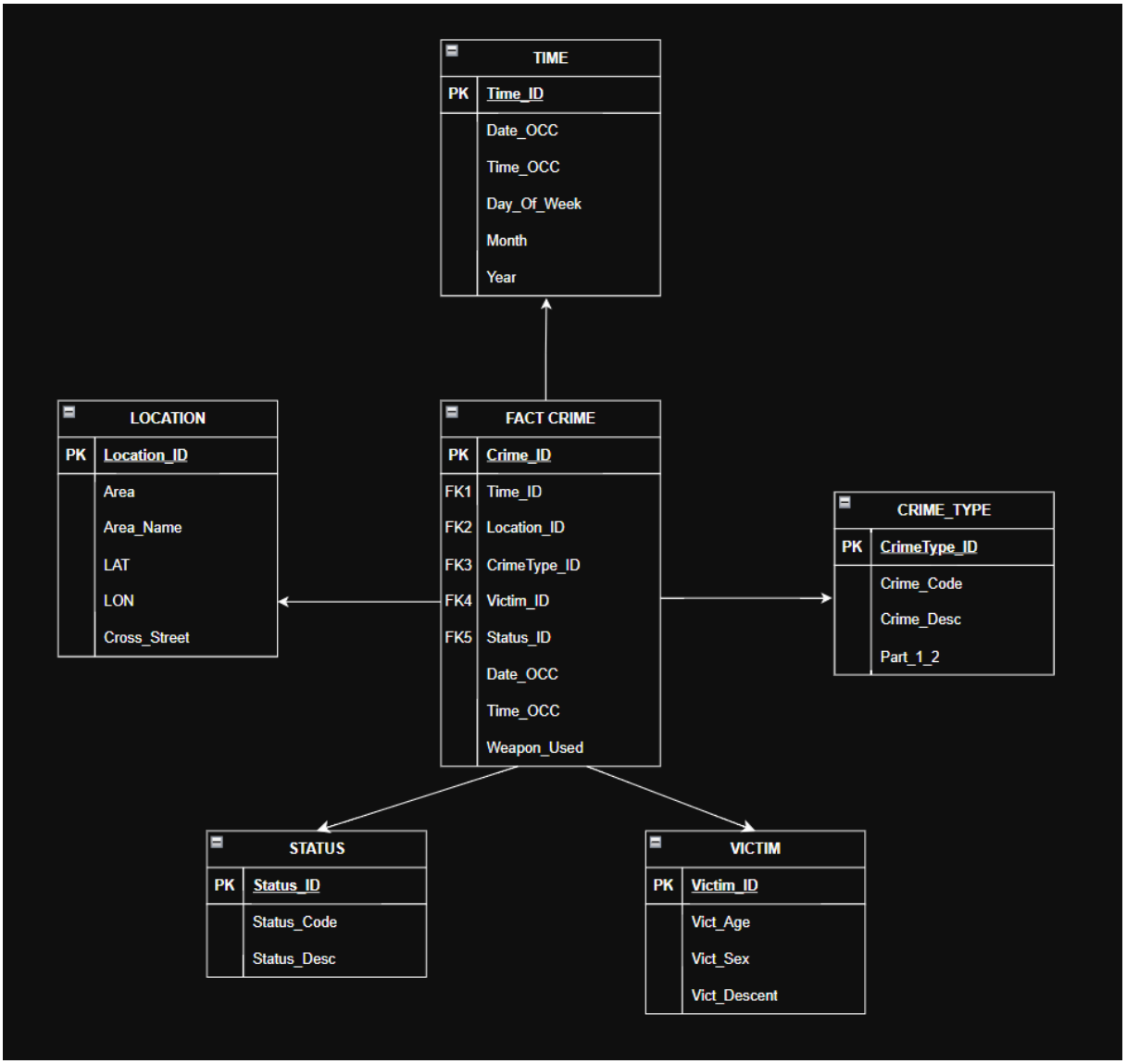


Figure 1: STAR SCHEMA

Crime Data Warehouse: Star Schema Explanation

The **Crime Data Warehouse (Figure 1)** is designed using a **Star Schema**, which is a simple and efficient way to store and analyze crime data. This schema has a **central fact table (Fact_Crime)** that records crime events and connects to different **dimension tables** that store details about each crime.

The **Fact_Crime** table is the main table where all reported crimes are stored. It includes important information like **when and where the crime happened, what type of crime it was, who the victim was, and the crime's status**. To keep things organized and easy to analyze, this table is linked to **five-dimension tables**:

1. **Time Dimension:** Stores details like **the date, time, day of the week, month, and year** to help analyze crime trends over time.

2. **Location Dimension:** Contains information about **the crime's location**, including the **area name, latitude, longitude, and nearby streets**, which helps in mapping crime patterns.
3. **Crime Type Dimension:** Describes the **type of crime**, using a **crime code and category (Part 1 or Part 2 crimes)** to make it easier to group similar crimes.
4. **Victim Dimension:** Holds **victim details**, such as **age, gender, and ethnic background**, to analyze who is most affected by crimes.
5. **Status Dimension:** Tracks the **current status of the crime** (for example, **solved, under investigation, or unsolved**), helping law enforcement understand how cases progress.

Why Do We Use the Star Schema?

We chose the **Star Schema** because it is **simple, fast, and easy to understand**. It keeps data well-organized by storing extra details in separate dimension tables while keeping the main crime records in the **Fact_Crime** table. This design makes it much **faster to search, filter, and analyze data**, especially when dealing with **large amounts of crime records**.

With this setup, law enforcement agencies can easily generate reports such as:

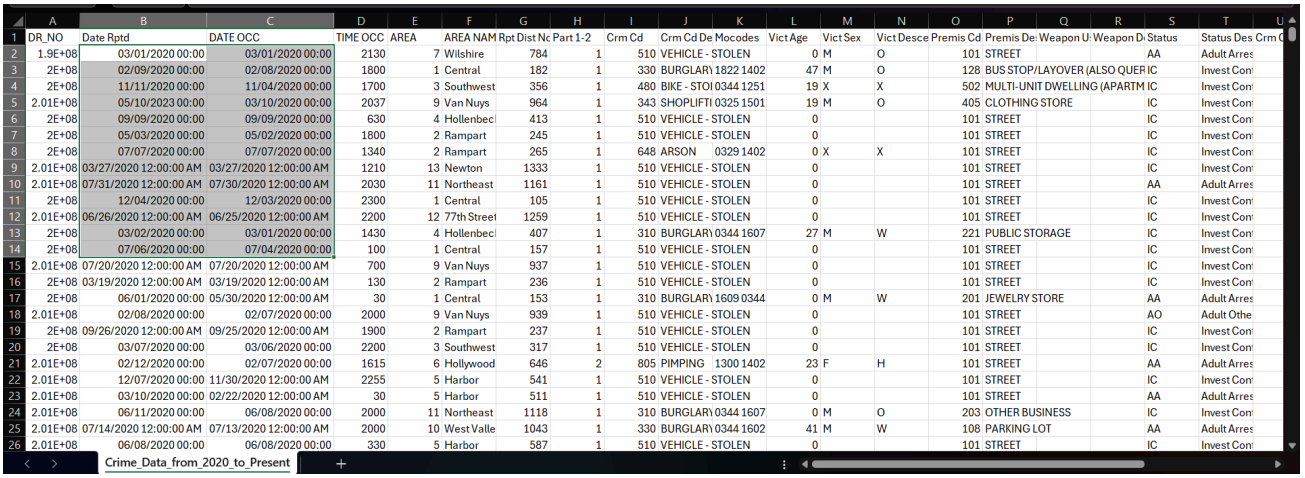
- **Which locations have the most crime?**
- **What time of day or year do most crimes happen?**
- **What is the distribution of crime types by woman victims and area?**
- **Crime Type Trends by Year and Victim Age Group?**

This **Star Schema** makes crime data **easier to analyze and use**, helping law enforcement make **smarter and faster decisions** to improve public safety.

2. Implement Tables and ETL Process

Dataset Review and Preprocessing

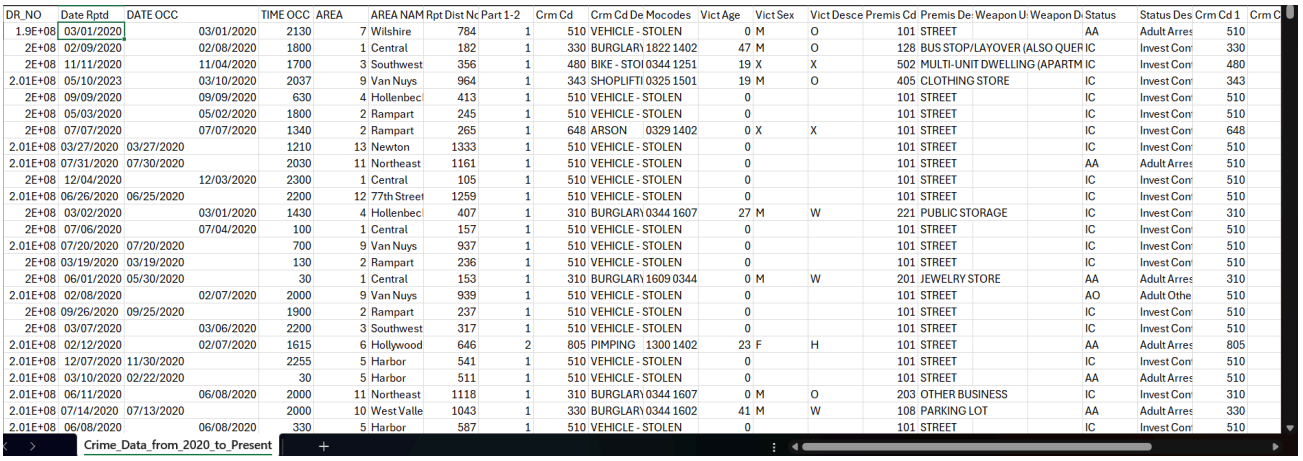
While reviewing the dataset, we focused on the date and time columns first. We noticed that TIME OCC was written in a different format, like "2130" or "0815", which is not a standard time format. However, we decided to keep this column as an integer during the initial import, because it behaves like a number and makes filtering and analysis easier for example, finding crimes that happened after 1800. On the other hand, the DATE OCC and Date Rptd columns had a standard datetime format but always included "00:00:00" as the time or "AM/PM" tags, which was unnecessary. After a careful check, we cleaned these two columns and converted them into a proper date-only format. This helped make the dataset cleaner and more consistent for the next steps in our project. Figure 2 and Figure 3 demonstrate these differences.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	DR_NO	Date Rptd	DATE OCC	TIME OCC	AREA	AREA NAM	Rpt Dist Nc	Part 1-2	Crm Cd	Crm Cd De	Mocodes	Vict Age	Vict Sex	Vict Desce	Premis Cd	Premis De	Weapon U	Weapon D	Status	Status Des	Crm C
2	1.9E+08	03/01/2020 00:00	03/01/2020 00:00	2130	7	Wilshire	784	1	510	VEHICLE - STOLEN		0	M	O	101	STREET			AA	Adult Arres	
3	2E+08	02/09/2020 00:00	02/08/2020 00:00	1800	1	Central	182	1	330	BURGLAR\	1822 1402	47	M	O	128	BUS STOP/LAYOVER (ALSO QUER			IC	Invest Cont	
4	2E+08	11/11/2020 00:00	11/04/2020 00:00	1700	3	Southwest	356	1	480	BIKE - STOI	0344 1251	19	X	X	502	MULTI-UNIT DWELLING (APARTM			IC	Invest Cont	
5	2.01E+08	05/10/2023 00:00	03/10/2020 00:00	2037	9	Van Nuys	964	1	343	SHOPLIFTI	0325 1501	19	M	O	405	CLOTHING STORE			IC	Invest Cont	
6	2E+08	09/09/2020 00:00	09/09/2020 00:00	630	4	Hollenbec	413	1	510	VEHICLE - STOLEN		0			101	STREET			IC	Invest Cont	
7	2E+08	05/03/2020 00:00	05/02/2020 00:00	1800	2	Rampart	245	1	510	VEHICLE - STOLEN		0			101	STREET			IC	Invest Cont	
8	2E+08	07/07/2020 00:00	07/07/2020 00:00	1340	2	Rampart	265	1	648	ARSON	0329 1402	0	X	X	101	STREET			IC	Invest Cont	
9	2.01E+08	03/27/2020 12:00:00 AM	03/27/2020 12:00:00 AM	1210	13	Newton	1333	1	510	VEHICLE - STOLEN		0			101	STREET			IC	Invest Cont	
10	2.01E+08	07/31/2020 12:00:00 AM	07/30/2020 12:00:00 AM	2030	11	Northeast	1161	1	510	VEHICLE - STOLEN		0			101	STREET			AA	Adult Arres	
11	2E+08	12/04/2020 00:00	12/03/2020 00:00	2300	1	Central	105	1	510	VEHICLE - STOLEN		0			101	STREET			IC	Invest Cont	
12	2.01E+08	06/26/2020 12:00:00 AM	06/25/2020 12:00:00 AM	2200	12	77th Street	1259	1	510	VEHICLE - STOLEN		0			101	STREET			IC	Invest Cont	
13	2E+08	03/02/2020 00:00	03/01/2020 00:00	1430	4	Hollenbec	407	1	310	BURGLAR\	0344 1607	27	M	W	221	PUBLIC STORAGE			IC	Invest Cont	
14	2E+08	07/06/2020 00:00	07/04/2020 00:00	100	1	Central	157	1	510	VEHICLE - STOLEN		0			101	STREET			IC	Invest Cont	
15	2.01E+08	07/20/2020 12:00:00 AM	07/20/2020 12:00:00 AM	700	9	Van Nuys	937	1	510	VEHICLE - STOLEN		0			101	STREET			IC	Invest Cont	
16	2E+08	03/19/2020 12:00:00 AM	03/19/2020 12:00:00 AM	130	2	Rampart	236	1	510	VEHICLE - STOLEN		0			101	STREET			IC	Invest Cont	
17	2E+08	06/01/2020 00:00	05/30/2020 12:00:00 AM	30	1	Central	153	1	310	BURGLAR\	1609 0344	0	M	W	201	JEWELRY STORE			AA	Adult Arres	
18	2.01E+08	02/08/2020 00:00	02/07/2020 00:00	2000	9	Van Nuys	939	1	510	VEHICLE - STOLEN		0			101	STREET			AO	Adult Othe	
19	2E+08	09/26/2020 12:00:00 AM	09/25/2020 12:00:00 AM	1900	2	Rampart	237	1	510	VEHICLE - STOLEN		0			101	STREET			IC	Invest Cont	
20	2E+08	03/07/2020 00:00	03/06/2020 00:00	2200	3	Southwest	317	1	510	VEHICLE - STOLEN		0			101	STREET			IC	Invest Cont	
21	2.01E+08	02/12/2020 00:00	02/07/2020 00:00	1615	6	Hollywood	646	2	805	PIMPING	1300 1402	23	F	H	101	STREET			AA	Adult Arres	
22	2.01E+08	12/07/2020 00:00	11/30/2020 12:00:00 AM	2255	5	Harbor	541	1	510	VEHICLE - STOLEN		0			101	STREET			IC	Invest Cont	
23	2.01E+08	03/10/2020 00:00	02/22/2020 12:00:00 AM	30	5	Harbor	511	1	510	VEHICLE - STOLEN		0			101	STREET			AA	Adult Arres	
24	2.01E+08	06/11/2020 00:00	06/08/2020 00:00	2000	11	Northeast	1118	1	310	BURGLAR\	0344 1607	0	M	O	203	OTHER BUSINESS			IC	Invest Cont	
25	2.01E+08	07/14/2020 12:00:00 AM	07/13/2020 12:00:00 AM	2000	10	West Valle	1043	1	330	BURGLAR\	0344 1602	41	M	W	108	PARKING LOT			AA	Adult Arres	
26	2.01E+08	06/08/2020 00:00	06/08/2020 00:00	330	5	Harbor	587	1	510	VEHICLE - STOLEN		0			101	STREET			IC	Invest Cont	

Crime_Data_from_2020_to_Present

Figure 2: Before Cleaning of Dataset



DR_NO	Date Rptd	DATE OCC	TIME OCC	AREA	AREA NAM	Rpt Dist Nc	Part 1-2	Crm Cd	Crm Cd De	Mocodes	Vict Age	Vict Sex	Vict Desce	Premis Cd	Premis De	Weapon U	Weapon D	Status	Status Des	Crm Cd 1	Crm C
1.9E+08	03/01/2020	03/01/2020	2130	7	Wilshire	784	1	510	VEHICLE - STOLEN		0	M	O	101	STREET			AA	Adult Arres	510	
2E+08	02/09/2020	02/08/2020	1800	1	Central	182	1	330	BURGLAR\	1822 1402	47	M	O	128	BUS STOP/LAYOVER (ALSO QUER			IC	Invest Cont	330	
2E+08	11/11/2020	11/04/2020	1700	3	Southwest	356	1	480	BIKE - STOI	0344 1251	19	X	X	502	MULTI-UNIT DWELLING (APARTM			IC	Invest Cont	480	
2.01E+08	05/10/2023	03/10/2020	2037	9	Van Nuys	964	1	343	SHOPLIFTI	0325 1501	19	M	O	405	CLOTHING STORE			IC	Invest Cont	343	
2E+08	09/09/2020	09/09/2020	630	4	Hollenbec	413	1	510	VEHICLE - STOLEN		0			101	STREET			IC	Invest Cont	510	
2E+08	05/03/2020	05/02/2020	1800	2	Rampart	245	1	510	VEHICLE - STOLEN		0			101	STREET			IC	Invest Cont	510	
2E+08	07/07/2020	07/07/2020	1340	2	Rampart	265	1	648	ARSON	0329 1402	0	X	X	101	STREET			IC	Invest Cont	648	
2.01E+08	03/27/2020	03/27/2020	1210	13	Newton	1333	1	510	VEHICLE - STOLEN		0			101	STREET			IC	Invest Cont	510	
2.01E+08	07/31/2020	07/30/2020	2030	11	Northeast	1161	1	510	VEHICLE - STOLEN		0			101	STREET			AA	Adult Arres	510	
2E+08	12/04/2020	12/03/2020	2300	1	Central	105	1	510	VEHICLE - STOLEN		0			101	STREET			IC	Invest Cont	510	
2.01E+08	06/26/2020	06/25/2020	2200	12	77th Street	1259	1	510	VEHICLE - STOLEN		0			101	STREET			IC	Invest Cont	510	
2E+08	03/02/2020	03/01/2020	1430	4	Hollenbec	407	1	310	BURGLAR\	0344 1607	27	M	W	221	PUBLIC STORAGE			IC	Invest Cont	310	
2E+08	07/06/2020	07/04/2020	100	1	Central	157	1	510	VEHICLE - STOLEN		0			101	STREET			IC	Invest Cont	510	
2.01E+08	07/20/2020	07/20/2020	700	9	Van Nuys	937	1	510	VEHICLE - STOLEN		0			101	STREET			IC	Invest Cont	510	
2E+08	03/19/2020	03/19/2020	130	2	Rampart	236	1	510	VEHICLE - STOLEN		0			101	STREET			IC	Invest Cont	510	
2E+08	06/01/2020	05/30/2020	30	1	Central	153	1	310	BURGLAR\	1609 0344	0	M	W	201	JEWELRY STORE			AA	Adult Arres	310	
2.01E+08	02/08/2020	02/07/2020	2000	9	Van Nuys	939	1	510	VEHICLE - STOLEN		0			101	STREET			AO	Adult Othe	510	
2E+08	09/26/2020	09/25/2020	1900	2	Rampart	237	1	510	VEHICLE - STOLEN		0			101	STREET			IC	Invest Cont	510	
2E+08	03/07/2020	03/06/2020	2200	3	Southwest	317	1	510	VEHICLE - STOLEN		0			101	STREET			IC	Invest Cont	510	
2.01E+08	02/12/2020	02/07/2020	1615	6	Hollywood	646	2	805	PIMPING	1300 1402	23	F	H	101	STREET			AA	Adult Arres	805	
2.01E+08	12/07/2020	11/30/2020	2255	5	Harbor	541	1	510	VEHICLE - STOLEN		0			101	STREET			IC	Invest Cont	510	
2.01E+08	03/10/2020	02/22/2020	30	5	Harbor	511	1	510	VEHICLE - STOLEN		0			101	STREET			AA	Adult Arres	510	
2.01E+08	06/11/2020	06/08/2020	2000	11	Northeast	1118	1	310	BURGLAR\	0344 1607	0	M	O	203	OTHER BUSINESS			IC	Invest Cont	310	
2.01E+08	07/14/2020	07/13/2020	2000	10	West Valle	1043	1	330	BURGLAR\	0344 1602	41	M	W	108	PARKING LOT			AA	Adult Arres	330	
2.01E+08	06/08/2020	06/08/2020	330	5	Harbor	587	1	510	VEHICLE - STOLEN		0			101	STREET			IC	Invest Cont	510	

Crime_Data_from_2020_to_Present

Figure 3: After Cleaning of Dataset

Importing the Dataset into SQL Server (SSMS)

After cleaning the date fields in our dataset, we saved the file as a CSV to prepare it for import. We used the *Import Flat File Wizard* in SQL Server Management Studio (SSMS) to bring the data into our database. During this process, we paid special attention to column types. For example, we defined TIME OCC as an INT column instead of a time format, since it would be easier to use in filters and numeric comparisons. We also made sure that columns like DATE OCC and Date Rptd were imported as DATE, and that fields like Vict Age were set as INT. In some cases, SSMS tried to guess the data type, so we manually corrected them to avoid any problems later. Once the data was imported, we created a staging table called Stage_CrimeData to store the raw version of the dataset before applying any transformations. Data types and Column definitions are demonstrated by Figure 4. Additionally, imported sample of data is shown by Figure 5.

	Column Name	Data Type	Allow Nulls
🔑	DR_NO	bigint	☐
▶	Date_Rptd	date	☑
	DATE_OCC	date	☑
	TIME_OCC	int	☑
	AREA	int	☑
	AREA_NAME	varchar(MAX)	☑
	Rpt_Dist_No	int	☑
	Part_1_2	int	☑
	Crm_Cd	int	☑
	Crm_Cd_Desc	varchar(MAX)	☑
	Mocodes	varchar(MAX)	☑
	Vict_Age	int	☑
	Vict_Sex	char(10)	☑
	Vict_Descent	char(10)	☑
	Premis_Cd	int	☑
	Premis_Desc	varchar(MAX)	☑
	Weapon_Used_Cd	int	☑
	Weapon_Desc	varchar(MAX)	☑
	Status	varchar(50)	☑
	Status_Desc	varchar(50)	☑
	Crm_Cd_1	int	☑
	Crm_Cd_2	int	☑
	Crm_Cd_3	int	☑
	Crm_Cd_4	int	☑
	LOCATION	varchar(MAX)	☑
	Cross_Street	varchar(MAX)	☑
	LAT	float	☑

Figure 4:Data Types and Column Definitions

	DR_NO	Date_Rptd	DATE_OCC	TIME_OCC	AREA	AREA_NAME	Rpt_Dist_No	Part_1_2	Crm_Cd	Crm_Cd_Desc	Mocodes	Vict_Age	Vict_Sex	Vict_Descent	Premis
1	817	2020-09-20	2020-09-19	1700	17	Devonshire	1777	1	510	VEHICLE - STOLEN	NULL	0	NULL	NULL	101
2	2113	2021-06-21	2021-04-05	1835	13	Newton	1385	1	510	VEHICLE - STOLEN	NULL	0	NULL	NULL	101
3	2203	2022-12-31	2022-12-19	1930	3	Southwest	356	1	522	VEHICLE, STOLEN - OTHER (MOTORIZED SCOOTERS, BIKE...	NULL	0	NULL	NULL	502
4	2315	2023-07-23	2023-07-22	1800	15	N Hollywood	1566	1	420	THEFT FROM MOTOR VEHICLE - PETTY ($950 & UNDER)	NULL	0	NULL	NULL	101
5	2401	2024-01-10	2023-12-21	1930	14	Pacific	1416	1	510	VEHICLE - STOLEN	NULL	0	NULL	NULL	101
6	10304468	2020-01-08	2020-01-08	2230	3	Southwest	377	2	624	BATTERY - SIMPLE ASSAULT	0444 0913	36	F	B	501
7	190101086	2020-01-02	2020-01-01	330	1	Central	163	2	624	BATTERY - SIMPLE ASSAULT	0416 1822 1414	25	M	H	102
8	190101087	2020-01-02	2020-01-01	510	1	Central	156	2	626	INTIMATE PARTNER - SIMPLE ASSAULT	1414 1218 2000 1814 0416 0447	53	F	B	502
9	190326475	2020-03-01	2020-03-01	2130	7	Wilshire	784	1	510	VEHICLE - STOLEN	NULL	0	M	O	101
10	191501505	2020-01-01	2020-01-01	1730	15	N Hollywood	1543	2	745	VANDALISM - MISDEAMEANOR ($399 OR UNDER)	0329 1402	76	F	W	502
11	191921269	2020-01-01	2020-01-01	415	19	Mission	1998	2	740	VANDALISM - FELONY ($400 & OVER, ALL CHURCH VANDA...	329	31	X	X	409
12	200100001	2020-02-26	2020-02-25	2000	1	Central	111	1	510	VEHICLE - STOLEN	NULL	0	NULL	NULL	108
13	200100002	2020-08-15	2020-08-14	1740	1	Central	176	1	510	VEHICLE - STOLEN	NULL	0	NULL	NULL	101
14	200100003	2020-08-15	2020-08-14	2200	1	Central	162	1	510	VEHICLE - STOLEN	NULL	0	NULL	NULL	101
15	200100005	2020-12-06	2020-12-05	2045	1	Central	195	1	510	VEHICLE - STOLEN	NULL	0	NULL	NULL	101
16	200100501	2020-01-02	2020-01-01	30	1	Central	163	1	121	RAPE, FORCIBLE	0413 1822 1262 1415	25	F	H	735
17	200100502	2020-01-02	2020-01-02	1315	1	Central	161	1	442	SHOPLIFTING - PETTY THEFT ($950 & UNDER)	1402 2004 0344 0387	23	M	H	404
18	200100504	2020-01-04	2020-01-04	40	1	Central	155	2	946	OTHER MISCELLANEOUS CRIME	1402 0392	0	X	X	726
19	200100507	2020-01-04	2020-01-04	200	1	Central	101	1	341	THEFT-GRAND ($950.01 & OVER)EXCPT,GUNS,FOWL,LIVE...	1822 0344 1402	23	M	B	502
20	200100508	2020-01-04	2020-01-04	900	1	Central	112	2	626	INTIMATE PARTNER - SIMPLE ASSAULT	0416 0417 1218 1814 2004	61	M	W	102
21	200100509	2020-01-04	2020-01-04	2200	1	Central	192	1	330	BURGLARY FROM VEHICLE	1822 1414 0344 1307	29	M	A	101
22	200100510	2020-01-05	2020-01-05	955	1	Central	111	2	930	CRIMINAL THREATS - NO WEAPON DISPLAYED	0421 0906	35	M	O	108
23	200100514	2020-01-05	2020-01-05	1355	1	Central	162	1	341	THEFT-GRAND ($950.01 & OVER)EXCPT,GUNS,FOWL,LIVE...	1822 0344 2032	41	M	A	503
24	200100515	2020-01-07	2020-01-07	1638	1	Central	162	1	648	ARSON	1402 1501 2004	0	X	X	404
25	200100520	2020-01-08	2020-01-08	1805	1	Central	128	1	442	SHOPLIFTING - PETTY THEFT ($950 & UNDER)	0325 1402 0344 1822	24	F	H	252
26	200100522	2020-01-09	2020-01-09	2330	1	Central	163	2	946	OTHER MISCELLANEOUS CRIME	1309	25	M	O	101
27	200100527	2020-01-11	2020-01-10	2015	1	Central	112	1	230	ASSAULT WITH DEADLY WEAPON, AGGRAVATED ASSAULT	2004 1100 0430 1402 1218 09...	51	M	W	152
28	200100528	2020-01-11	2020-01-10	2015	1	Central	112	1	230	ASSAULT WITH DEADLY WEAPON, AGGRAVATED ASSAULT	2004 2060 0430 1402 1218 09...	41	M	B	152
29	200100531	2020-01-13	2020-01-13	1640	1	Central	195	2	624	BATTERY - SIMPLE ASSAULT	1402 0917 0315 0416 1309	18	M	H	210
30	200100535	2020-01-14	2020-01-14	1330	1	Central	152	1	210	ROBBERY	0416 0411 0344 1822 0305 03...	66	M	B	103
31	200100536	2020-01-14	2020-01-14	1400	1	Central	119	2	624	BATTERY - SIMPLE ASSAULT	0447 0910 1822	0	M	H	834
32	200100538	2020-01-14	2020-01-14	1730	1	Central	162	1	341	THEFT-GRAND ($950.01 & OVER)EXCPT,GUNS,FOWL,LIVE...	0344 1822 2032	31	M	H	404
33	200100543	2020-01-15	2020-01-15	1445	1	Central	162	1	442	SHOPLIFTING - PETTY THEFT ($950 & UNDER)	0325 1402 0907	27	M	B	404

Figure 5: Sample of Dataset

Schema Design and Implementation

After importing the cleaned data into our staging table, we moved on to designing and creating the core structure of our data warehouse using the star schema approach. We created five dimension tables and one central fact table. Each dimension table stores detailed information about a specific aspect of a crime record, while the fact table connects them through foreign keys.

The Time_Dimension table holds the crime date and time, along with day, month, and year details to support time-based analysis. The Location_Dimension stores geographic information like area name, latitude, longitude, and nearby cross streets. The CrimeType_Dimension captures crime codes, descriptions, and whether the crime falls under Part 1 or Part 2 categories. The Victim_Dimension table includes demographic information such as age, sex, and descent. The Status_Dimension contains the case status and its description.

At the center, we created the Fact_Crime table, which links to all dimension tables using foreign keys. It also stores some repeated facts such as the exact date, time, and the weapon used. This schema allows fast and efficient analysis, as it separates descriptive data from the transactional crime events.

By organizing the data in this way (Figure 6), we made it easier to query and visualize complex crime patterns in a meaningful and structured format.

```sql
USE CrimeDataWarehouse;
GO

CREATE TABLE Time_Dimension (
    Time_ID INT IDENTITY(1,1) PRIMARY KEY,
    Date_OCC DATE NOT NULL,
    Time_OCC TIME NOT NULL,
    Day_Of_Week VARCHAR(10),
    Month VARCHAR(20),
    Year INT
);

CREATE TABLE Location_Dimension (
    Location_ID INT IDENTITY(1,1) PRIMARY KEY,
    Area VARCHAR(50),
    Area_Name VARCHAR(100) NOT NULL,
    LAT DECIMAL(10,6),
    LON DECIMAL(10,6),
    Cross_Street VARCHAR(100)
);

CREATE TABLE CrimeType_Dimension (
    CrimeType_ID INT IDENTITY(1,1) PRIMARY KEY,
    Crime_Code INT NOT NULL,
    Crime_Desc VARCHAR(255) NOT NULL,
    Part_1_2 VARCHAR(10)
);

CREATE TABLE Status_Dimension (
    Status_ID INT IDENTITY(1,1) PRIMARY KEY,
    Status_Code VARCHAR(20),
    Status_Desc VARCHAR(100) NOT NULL
);
```

```sql
CREATE TABLE Victim_Dimension (
    Victim_ID INT IDENTITY(1,1) PRIMARY KEY,
    Vict_Age INT,
    Vict_Sex VARCHAR(10),
    Vict_Descent VARCHAR(50)
);

CREATE TABLE Fact_Crime (
    Crime_ID INT PRIMARY KEY,
    Time_ID INT NOT NULL,
    Location_ID INT NOT NULL,
    CrimeType_ID INT NOT NULL,
    Victim_ID INT NOT NULL,
    Status_ID INT NOT NULL,
    Date_OCC DATE,
    Time_OCC TIME,
    Weapon_Used VARCHAR(50),
    FOREIGN KEY (Time_ID) REFERENCES Time_Dimension(Time_ID),
    FOREIGN KEY (Location_ID) REFERENCES Location_Dimension(Location_ID),
    FOREIGN KEY (CrimeType_ID) REFERENCES CrimeType_Dimension(CrimeType_ID),
    FOREIGN KEY (Victim_ID) REFERENCES Victim_Dimension(Victim_ID),
    FOREIGN KEY (Status_ID) REFERENCES Status_Dimension(Status_ID)
);
```

Figure 6: SQL Code for Create Tables and Relationships

ETL Process with SSIS

Time Dimension Population

To populate the Time_Dimension table, we designed a data flow in SSIS that starts from the raw data source and ends with inserting records into the time dimension. Below are the main steps we followed:

1. Source Connection (ADO.NET Source):
We connected to the CrimeDataWarehouse database and selected the staging table Crime_Raw_Data as our input. This allowed us to extract all rows containing date and time information for each crime event.

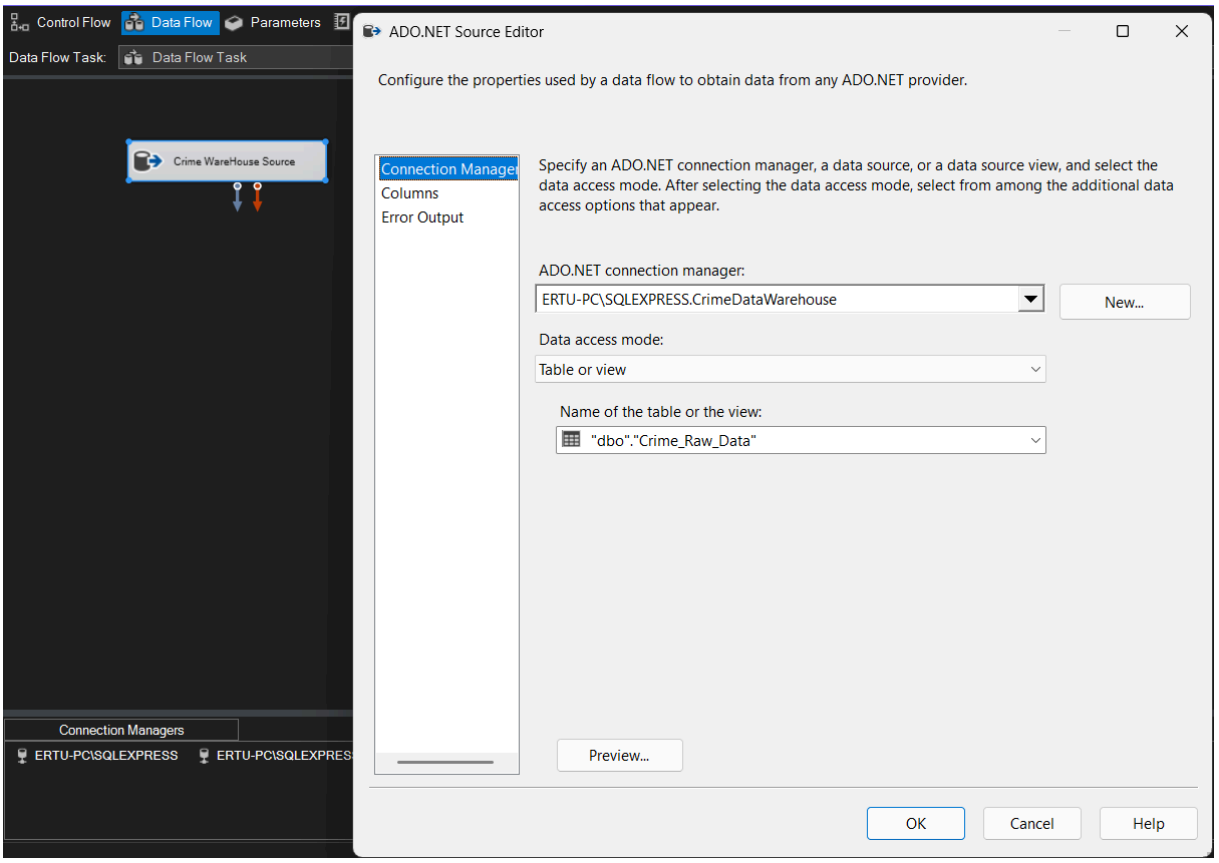


Figure 7: ADO.NET source connected to Crime_Raw_Data

2. Derived Column Transformation:
We used the **Derived Column** tool to generate new columns for Day_Of_Week, Month, and Year from the DATE_OCC field.

- Day_Of_Week: Calculated using DATEPART("dw", DATE_OCC)
- Month: DATEPART("MM", DATE_OCC)
- Year: DATEPART("YYYY", DATE_OCC)

These expressions extract useful time-related parts of the date, making it easier to group and analyze crimes by time later.

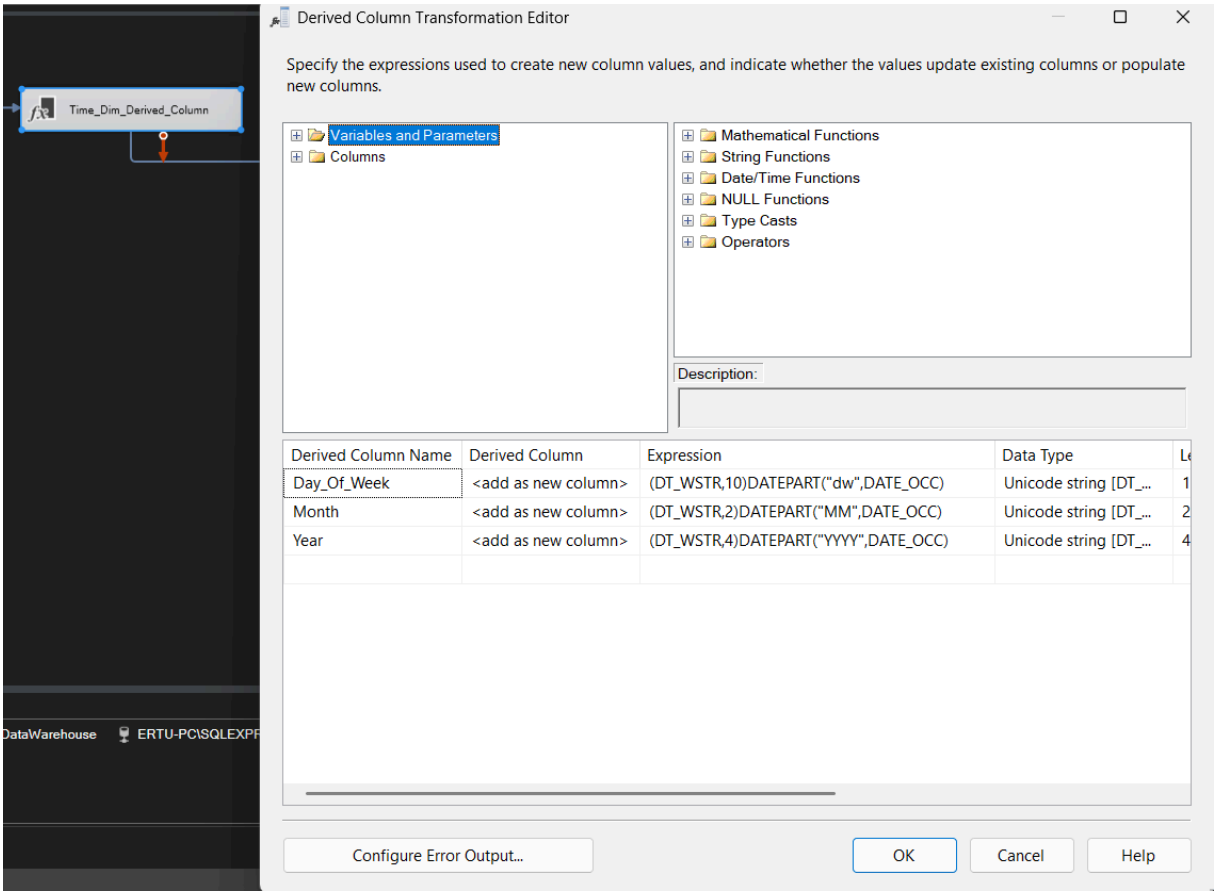


Figure 8:Derived columns created from DATE_OCC

3. Sort Transformation:

We sorted the data by DATE_OCC and TIME_OCC in ascending order to prepare for possible deduplication and to improve data quality. Sorting ensures consistent loading and avoids random order insertions.

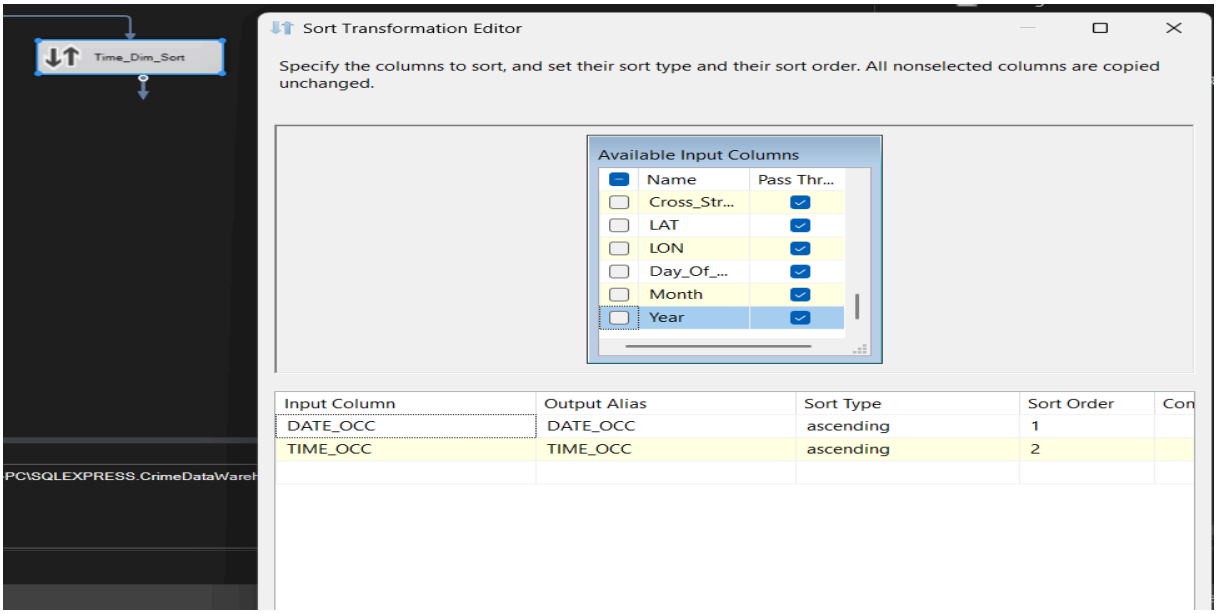


Figure 9:Sorting by date and time before inserting into the dimension table

4. Column Mapping and Insertion

In the final step of the ETL process for the time dimension, we used the **ADO.NET Destination Editor** to map our transformed columns to the correct fields in the Time_Dimension table. We matched each input column to its destination:

- DATE_OCC → Date_OCC
- TIME_OCC → Time_OCC
- Day_Of_Week → Day_Of_Week
- Month → Months
- Year → Years

The Time_ID column was set to <ignore> because it is the primary key and will be auto-generated in future steps. This mapping ensures that the cleaned and enriched time data is accurately inserted into the table. After completing this setup, we executed the package, and the records were successfully loaded into the Time_Dimension table without errors.

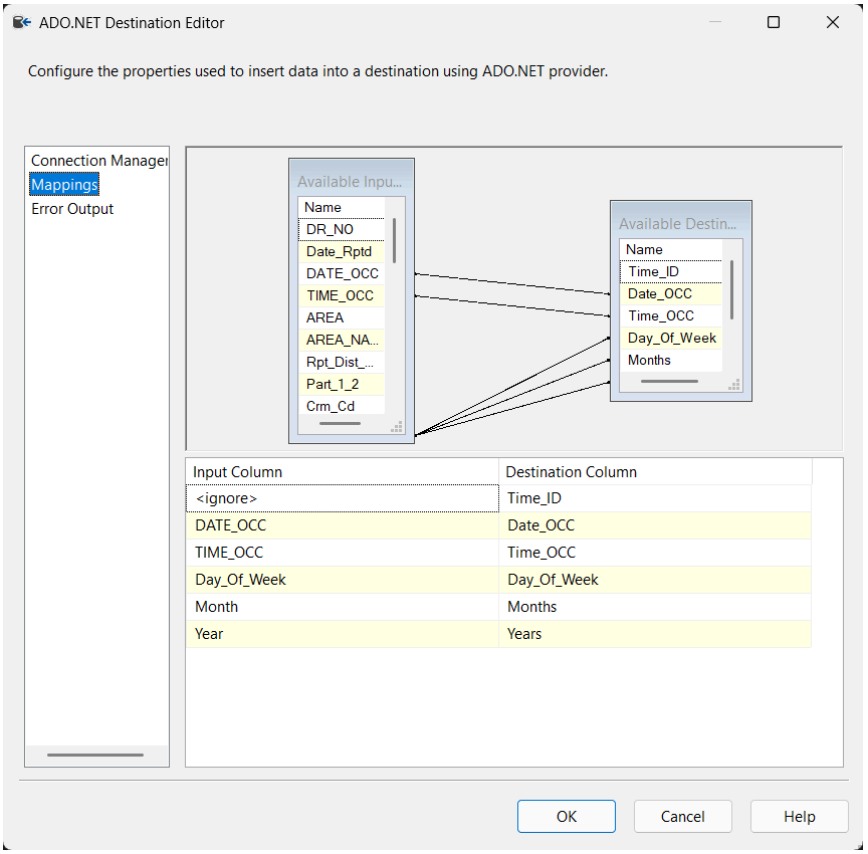


Figure 10:Column mapping between the source data and the Time_Dimension table using ADO.NET Destination.

5. Destination (OLE DB Destination):

Finally, we directed the data into the Time_Dimension table using an OLE DB Destination. At this step, we inserted the original DATE_OCC and TIME_OCC values along with the new derived columns: Day_Of_Week, Month, and Year.

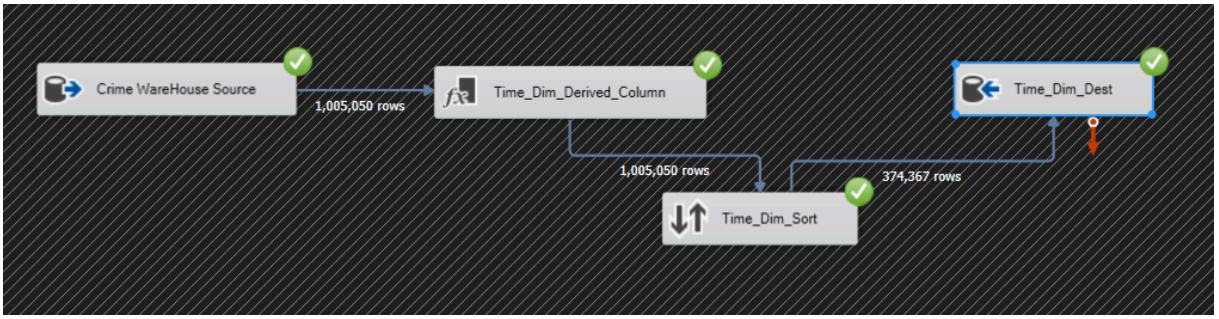


Figure 11: Rows processed from source, and unique rows inserted into Time_Dimension

Location Dimension Population

To load data into the Location_Dimension table, we followed a similar approach as in the time dimension ETL process. We built a data flow in SSIS with multiple steps to clean, transform, and insert location-related data.

1. Derived Column Transformation

We used the **Derived Column** transformation to prepare the AREA and AREA_NAME fields. Since AREA_NAME was a text column, we converted it to a Unicode string using (DT_WSTR,100)AREA_NAME. We also converted AREA from integer to string using (DT_WSTR,10)AREA. This was done to ensure proper data type compatibility before insertion.

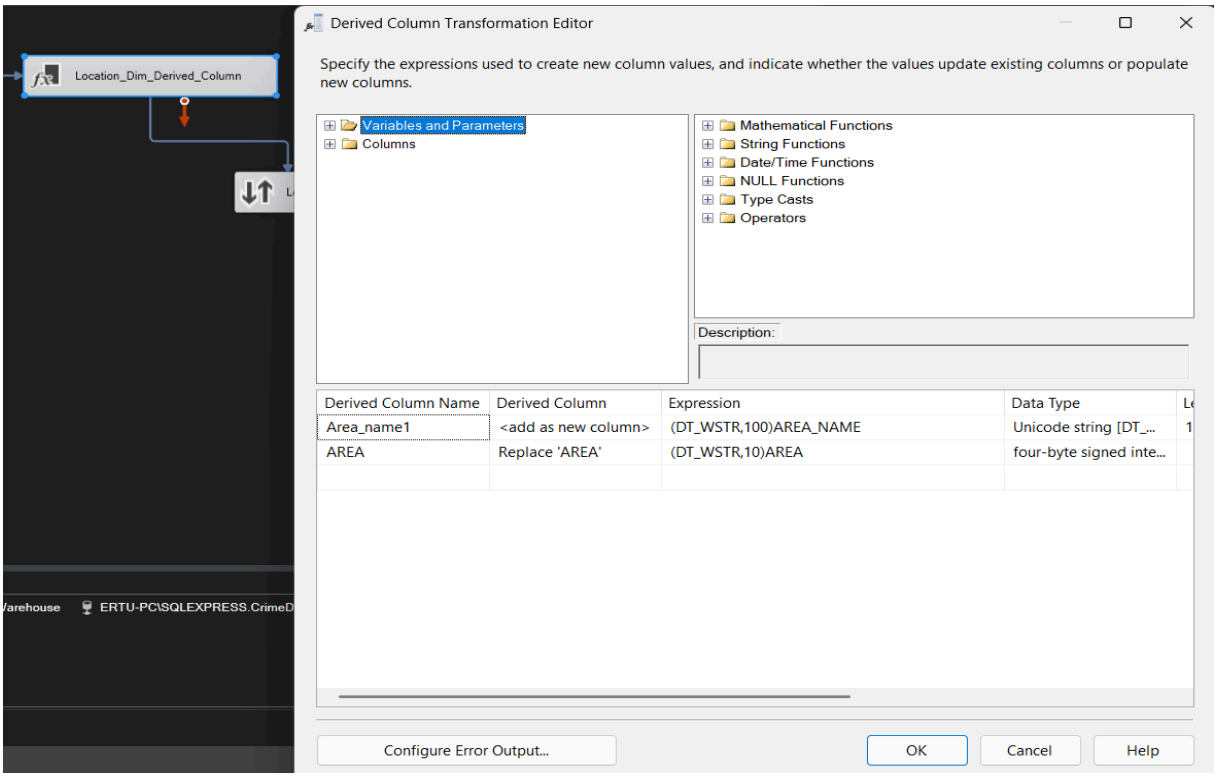


Figure 11:Converting AREA and AREA_NAME to proper formats using Derived Column

2. Sort and Remove Duplicates:

In the **Sort Transformation**, we sorted the location data by Area_name1, LON, and LAT. We also checked the option **"Remove rows with duplicate sort values"** to ensure only unique location records would be inserted into the Location_Dimension. This reduced over 1 million raw records down to 78,354 unique combinations.

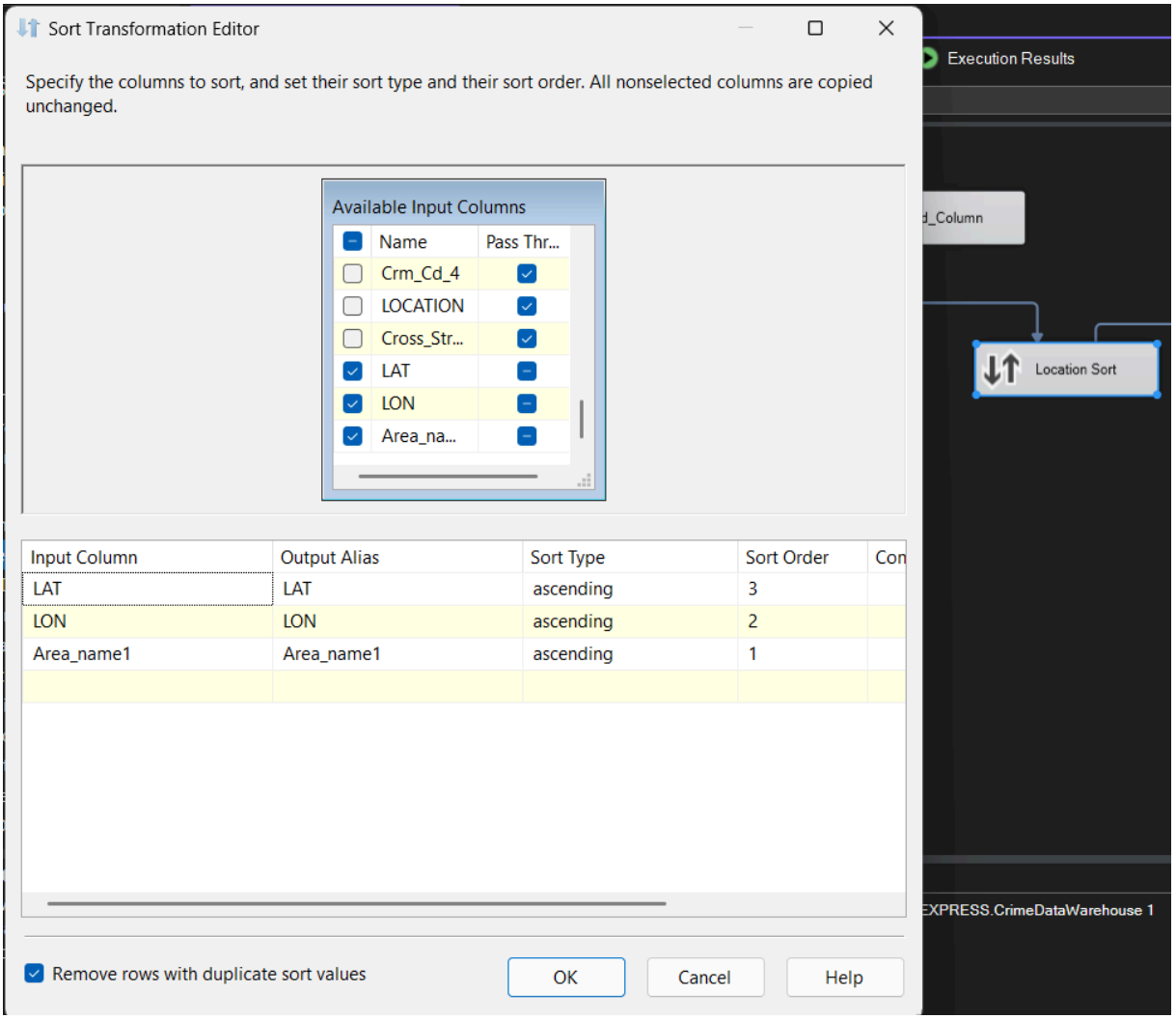


Figure 12:Sort transformation configured to eliminate duplicate location entries

3. Column Mapping and Loading:

We then mapped the cleaned and sorted data into the Location_Dimension table using **ADO.NET Destination**. The columns were matched as follows:

- AREA → Area
- AREA_NAME → Area_Name
- LAT, LON, and Cross_Street were directly mapped
- Location_ID was ignored as it will be auto-generated

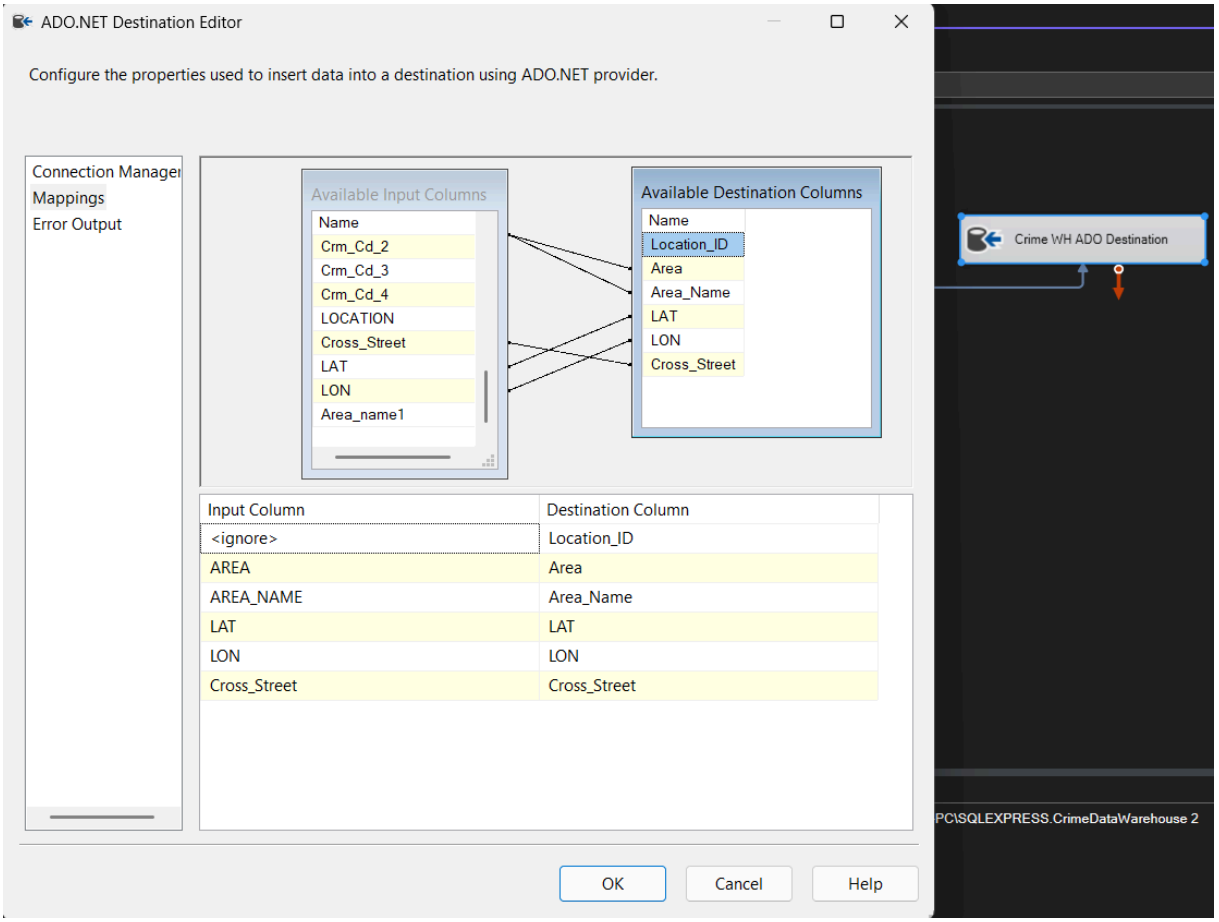


Figure 13:Mapping input columns to the destination table using ADO.NET Destination Editor

4. Final Load:

After executing the data flow, the SSIS package successfully loaded **78,354 unique location records** into the Location_Dimension table from a total of over 1 million raw rows.

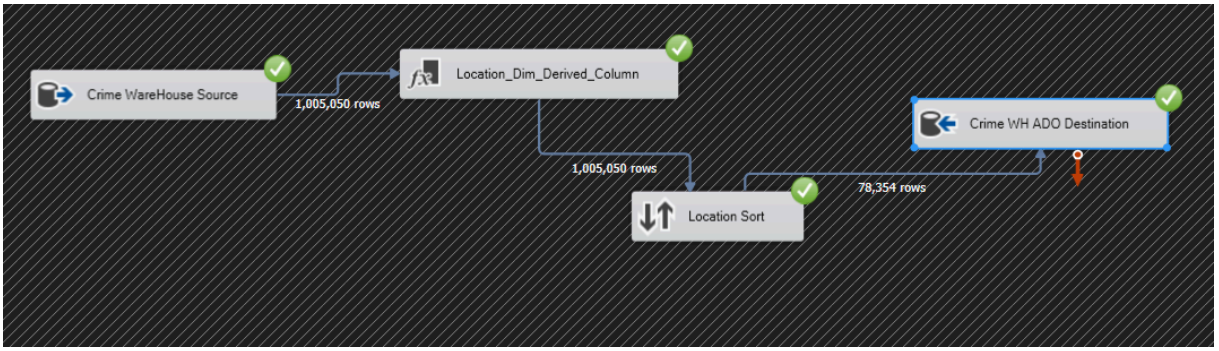


Figure 14:Complete flow of Location Dimension with derived columns, sorting, and successful load

CrimeType Dimension Population

To populate the CrimeType_Dimension table, we created a dedicated data flow in SSIS that extracts crime code information from the raw data, formats it correctly, removes duplicates, and inserts the results into the dimension table.

1. Derived Column Transformation:
We added a new derived column called Crm_Cd_Desc1 using the expression (DT_WSTR,255)Crm_Cd_Desc. This step was necessary to convert the crime description field into a Unicode string format (VARCHAR) that matches the target column type in the destination table.

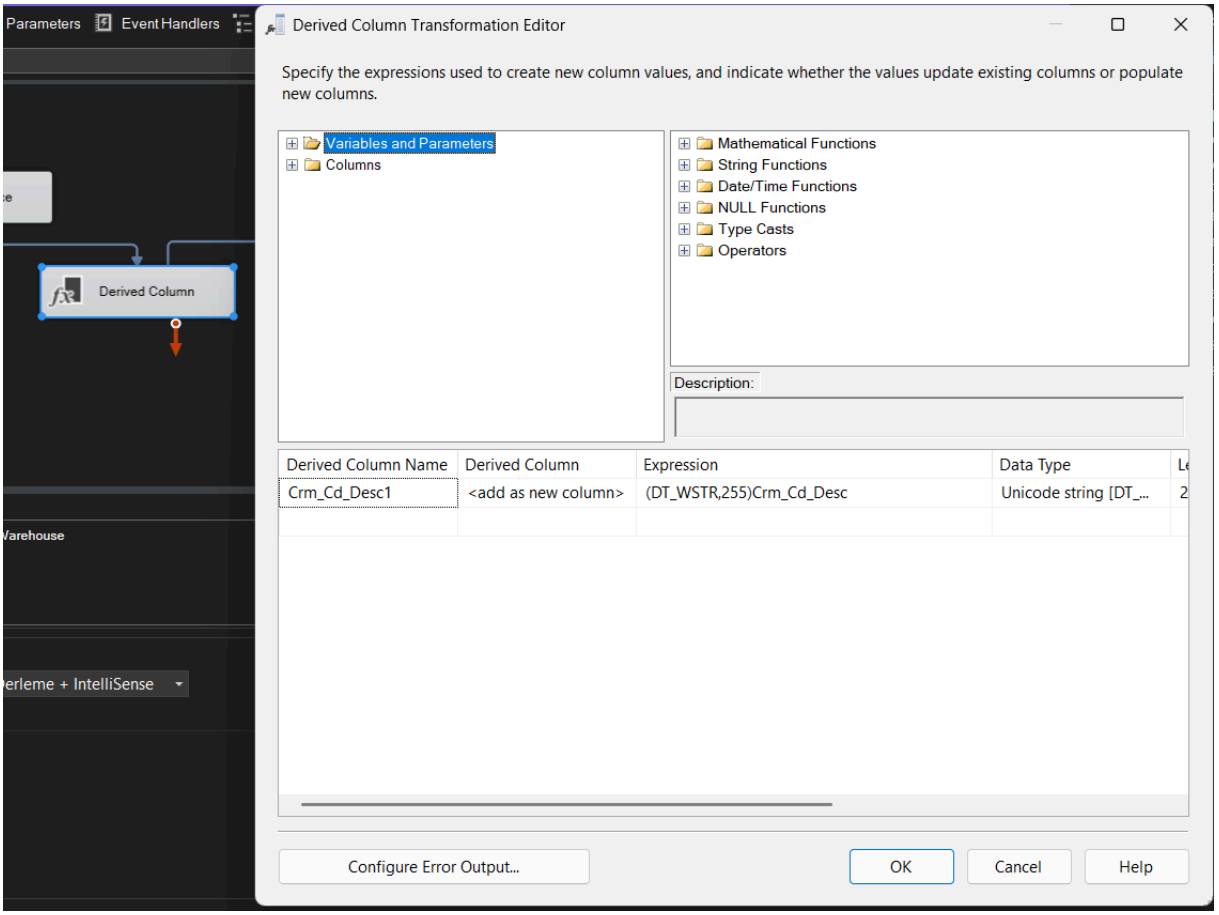


Figure 15:Derived column created to handle Crm_Cd_Desc with the correct data type

2. Sort and Remove Duplicates:
We used a Sort transformation to sort by Crm_Cd and Crm_Cd_Desc1 in ascending order. The checkbox **"Remove rows with duplicate sort values"** was selected to eliminate any repeated combinations. This ensures only unique crime types are inserted into the dimension table.

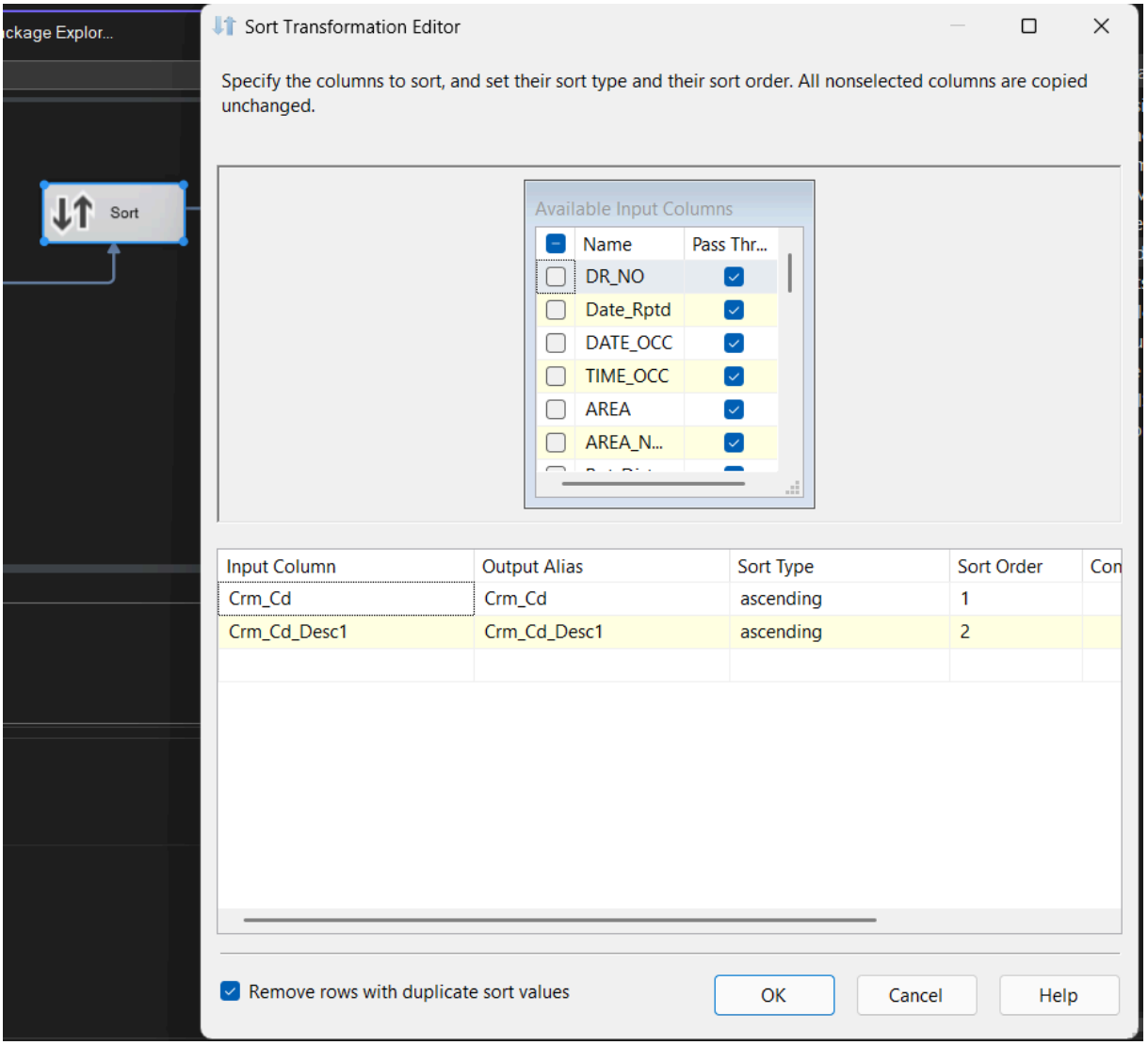


Figure 16:Sorting and duplicate removal based on crime code and description

3. Column Mapping and Loading:

We mapped the cleaned columns to the CrimeType_Dimension table using the ADO.NET Destination tool. The mapping was as follows:

- Crm_Cd ⟶ Crime_Code
- Crm_Cd_Desc1 ⟶ Crime_Desc
- Part_1_2 ⟶ Part_1_2
  CrimeType_ID was excluded because it is a primary key that will be auto-generated.

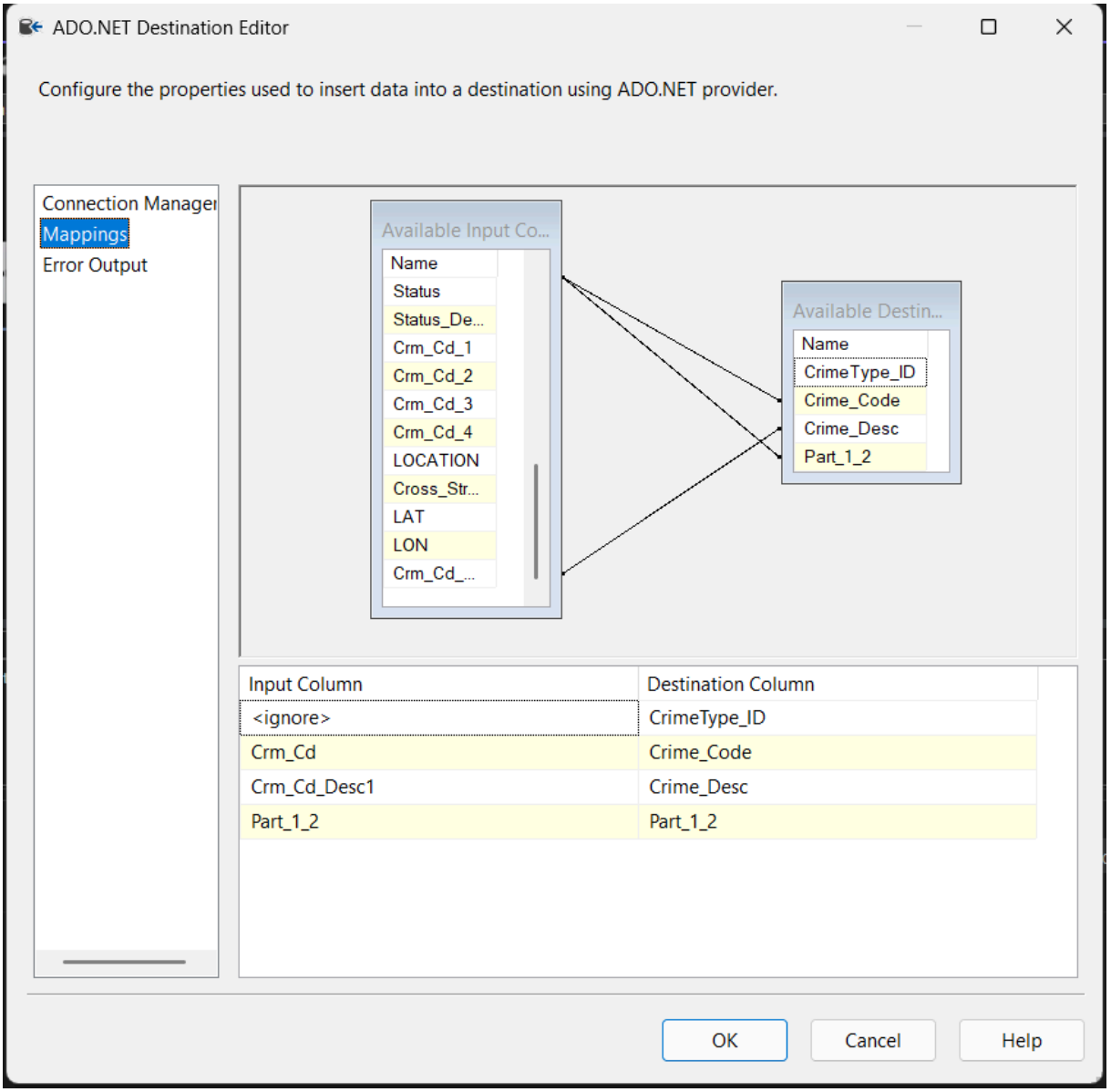


Figure 17:Input columns mapped to the CrimeType_Dimension table

4. Final Load:

From over 1 million records in the raw data, only **140 unique crime type records** were inserted into the CrimeType_Dimension table. This step optimized our schema by reducing redundancy and preparing clean reference data for future analysis.

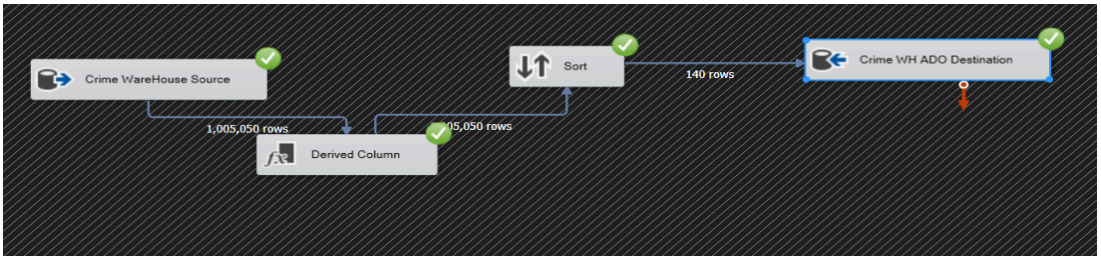


Figure 18:End-to-end ETL flow for CrimeType Dimension successfully executed

Status Dimension Population

To populate the Status_Dimension table, we used an SSIS data flow to extract status-related information from the raw dataset, clean and convert the columns, remove duplicates, and insert the final results into the dimension table.

1. Derived Column Transformation:
We created two new columns:

- Status_Desc_CLEAN: Converted from Status_Desc using the expression (DT_WSTR,100)[Status_Desc]
- Status_Code_CLEAN: Converted from Status using (DT_WSTR,20)[Status]

These conversions ensured that the data types matched the target columns in the dimension table and prevented type mismatches during the load process.

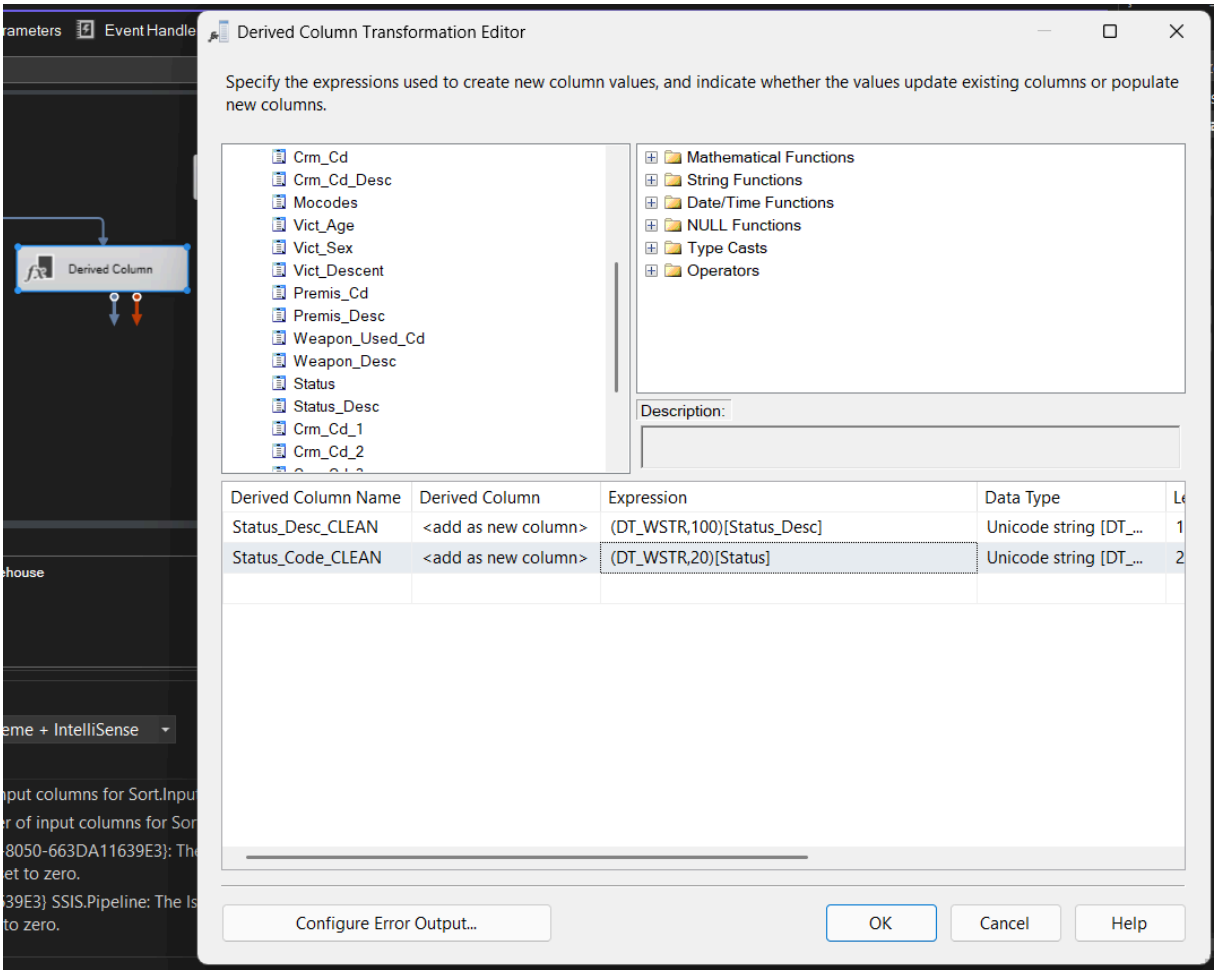


Figure 19:Derived Column tool used to clean and format Status and Status_Desc

2. Sort and Remove Duplicates:

In the Sort transformation, we sorted the data based on Status_Code_CLEAN and Status_Desc_CLEAN. We also enabled the option **"Remove rows with duplicate sort values"** to keep only unique status code–description pairs.

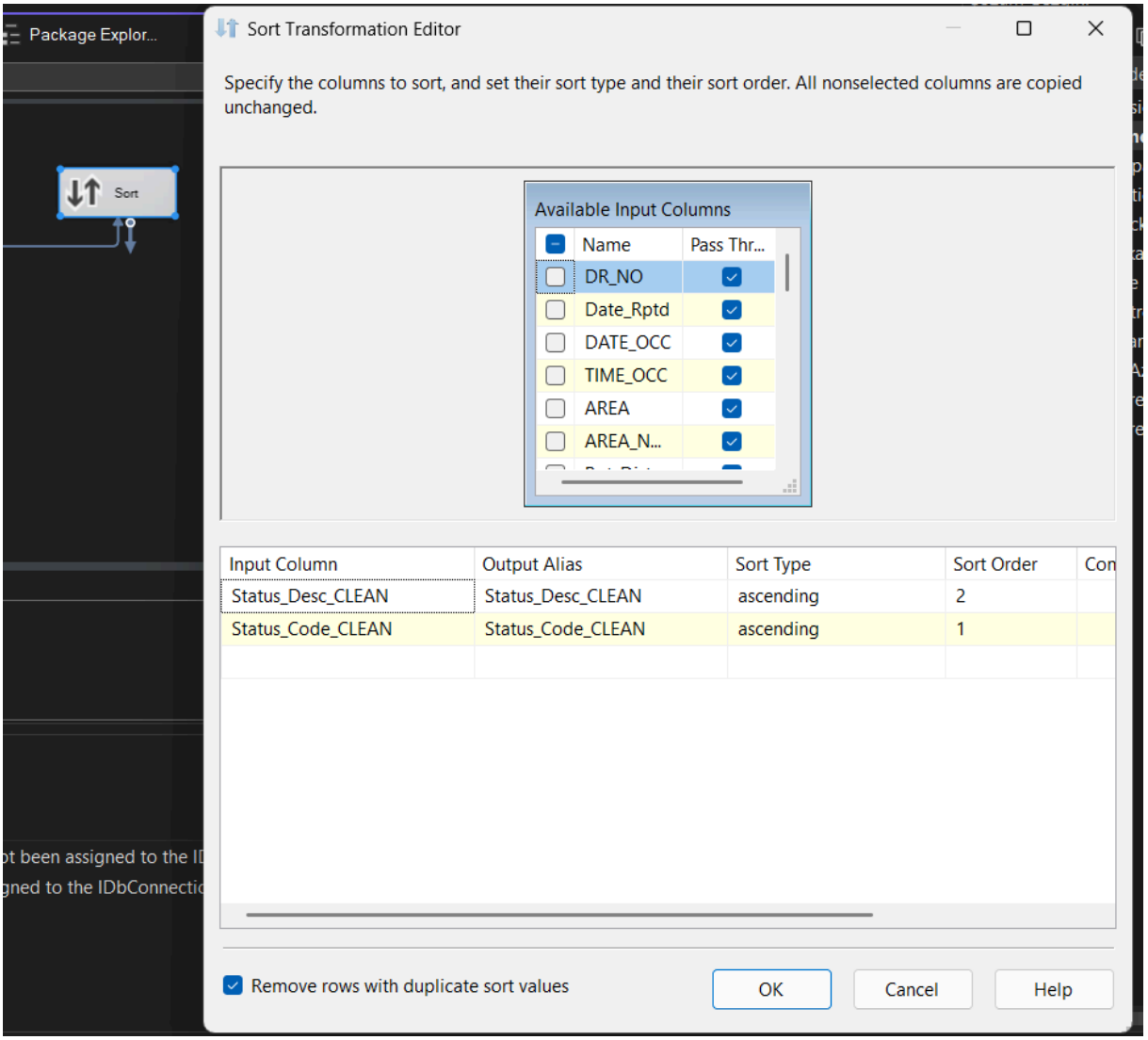


Figure 20:Sorting and removing duplicates from status data

3. Column Mapping and Destination:

In the ADO.NET Destination Editor, we mapped the cleaned columns to the corresponding fields in the Status_Dimension table:

- Status_Code_CLEAN → Status_Code
- Status_Desc_CLEAN → Status_Desc
  The Status_ID column was ignored because it is auto-incremented.

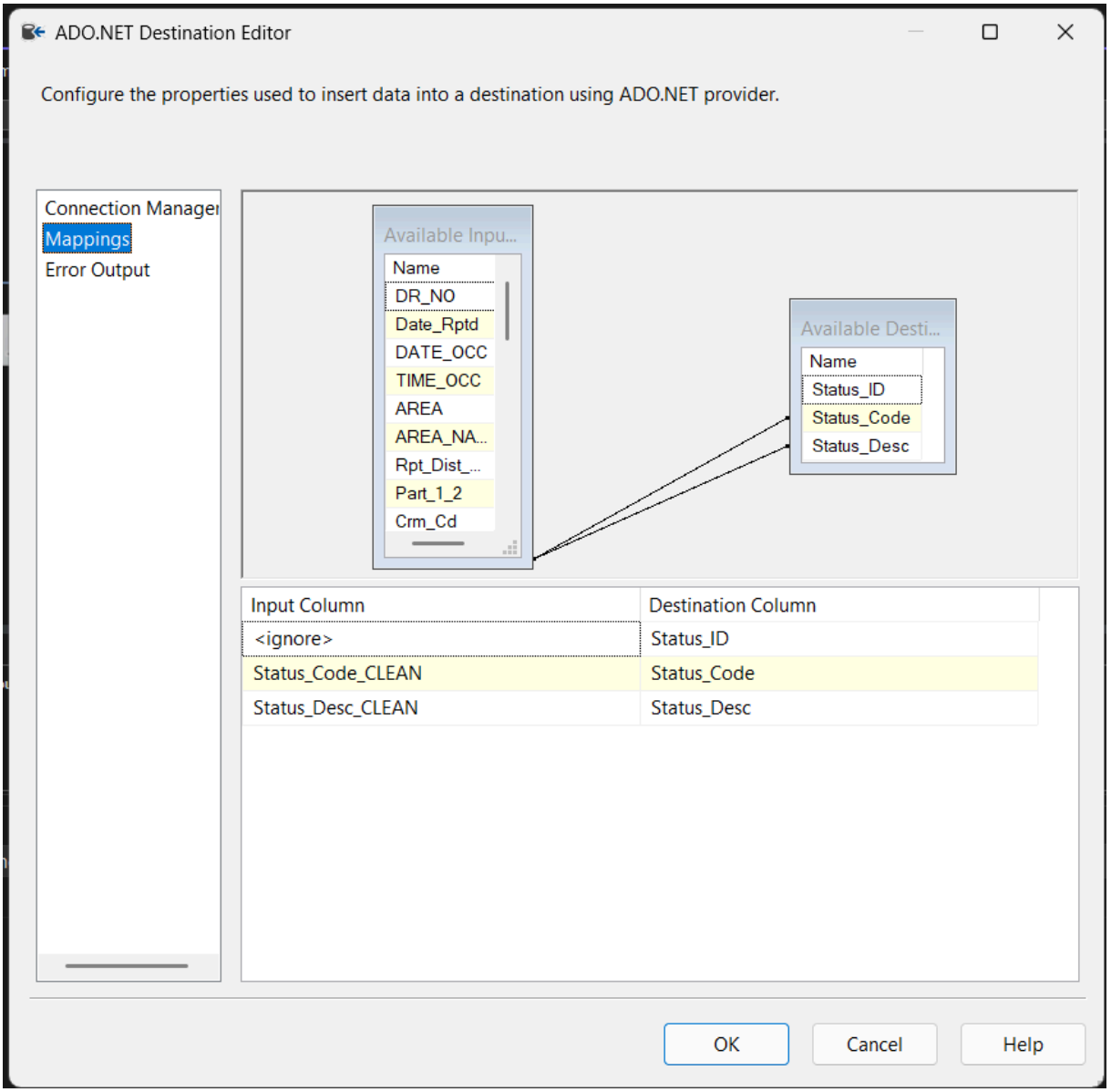


Figure 21:Mapping status fields into the target dimension table

4. Final Load:

After executing the ETL pipeline, only **7 unique status records** were inserted into the Status_Dimension table. This step ensured a clean and simplified reference for case status values in the final warehouse model.

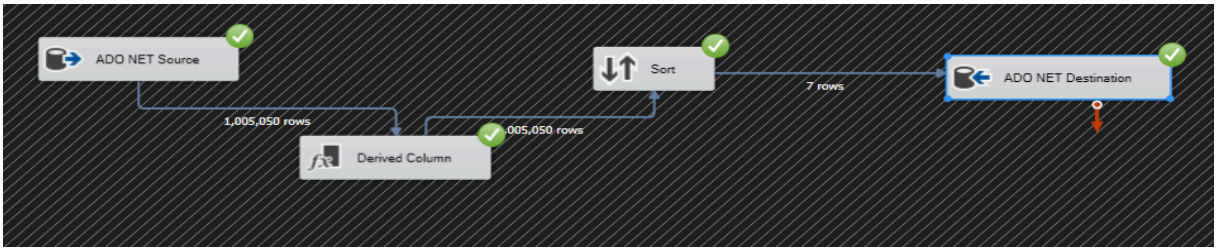


Figure 22:End-to-end data flow for loading the Status Dimension

Victim Dimension Population

To populate the Victim_Dimension table, we created an ETL process in SSIS to extract victim-related data, clean and convert the necessary fields, remove duplicates, and load the unique records into the dimension table.

1. Derived Column Transformation:
We created two new columns using the Derived Column tool:

- Vict_Sex_CLEAN: Converted from Vict_Sex using (DT_WSTR,10)[Vict_Sex]
- Vict_Descent_CLEAN: Converted from Vict_Descent using (DT_WSTR,50)[Vict_Descent]
  This step was required to ensure proper string formatting and compatibility with the target data types.

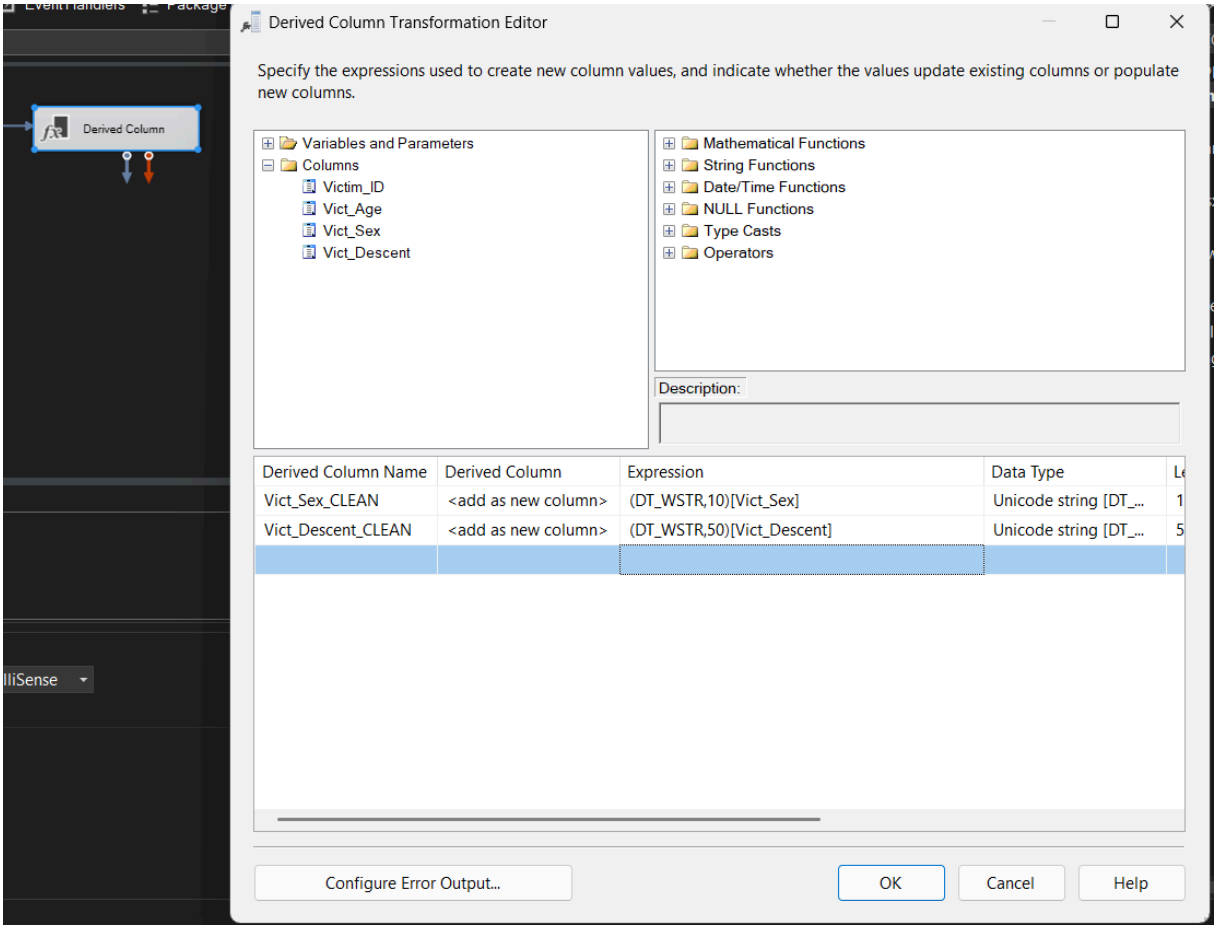


Figure 23:Victim sex and descent fields converted to Unicode strings

2. Sort and Remove Duplicates:
We sorted the data by Vict_Age, Vict_Sex_CLEAN, and Vict_Descent_CLEAN. The option **"Remove rows with duplicate sort values"** was selected to ensure that only unique combinations of victim attributes were included in the final table.

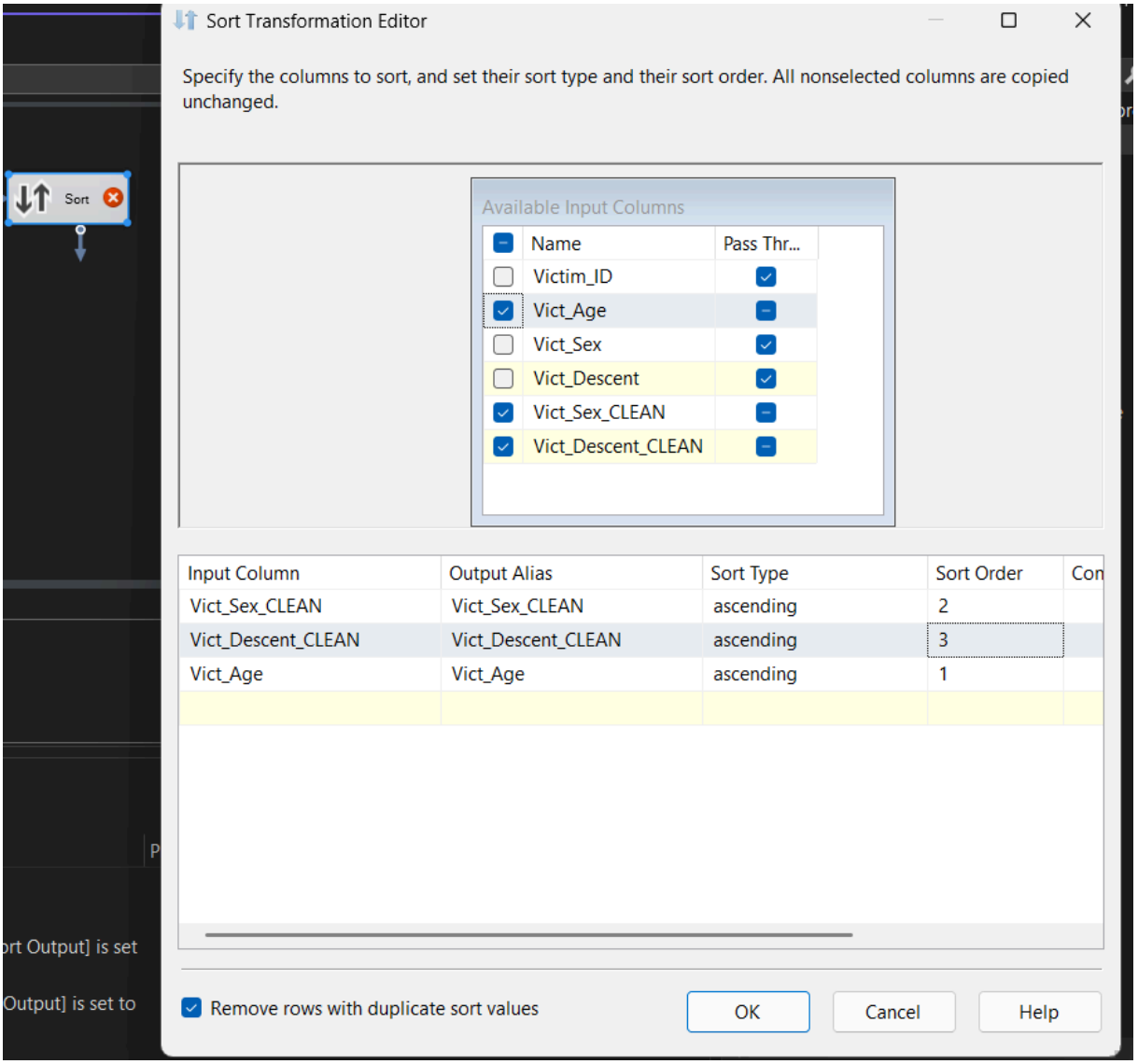


Figure 24:Sorting transformation configured to ensure uniqueness of victim records

3. ADO.NET Destination Setup:

The cleaned data was then mapped to the Victim_Dimension table using the ADO.NET Destination. The mapping was as follows:

- Vict_Age ⟶ Vict_Age
- Vict_Sex_CLEAN ⟶ Vict_Sex
- Vict_Descent_CLEAN ⟶ Vict_Descent
  The Victim_ID was ignored because it is an auto-incremented primary key.

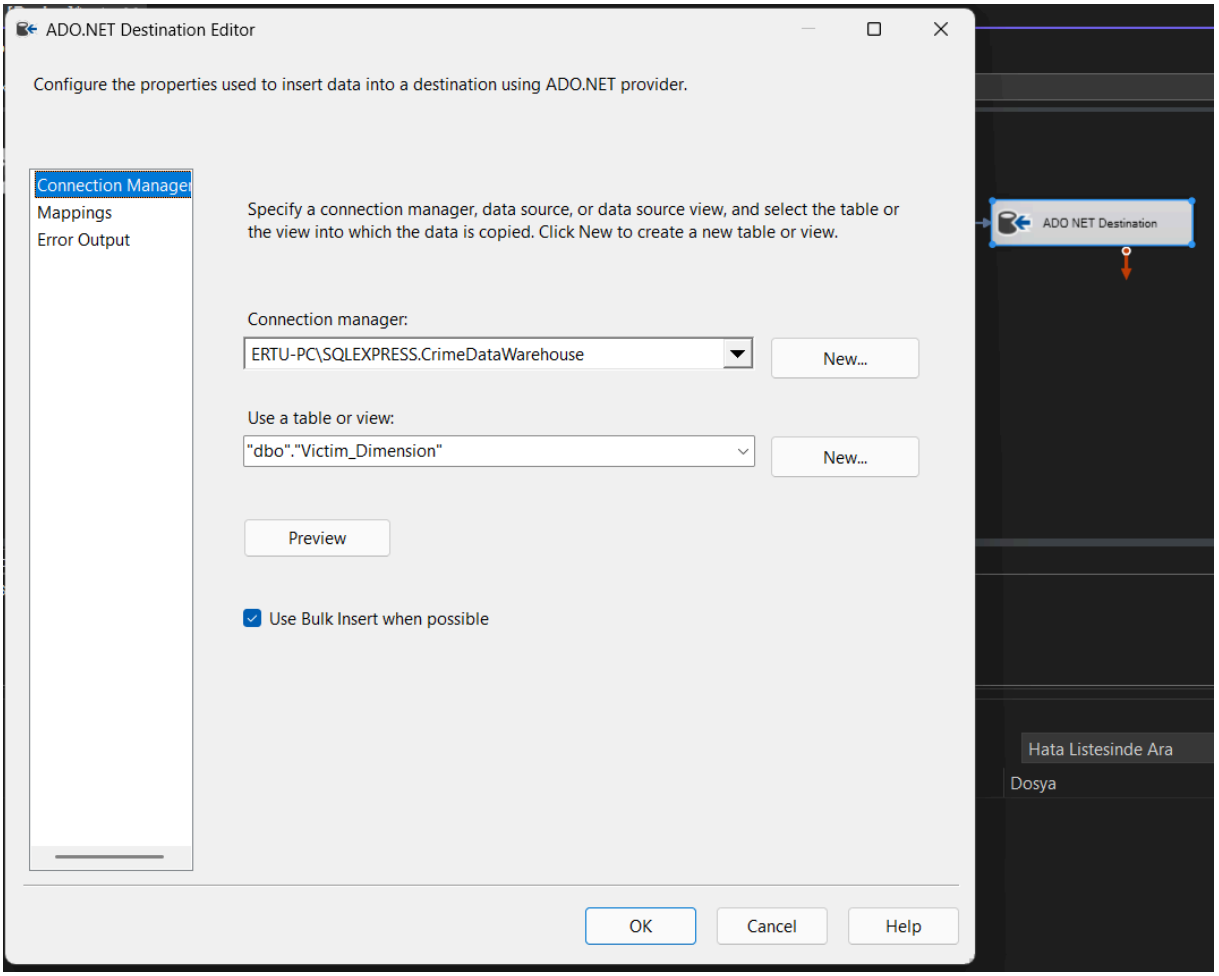


Figure 25:Cleaned victim data mapped into the destination table

4. Final Load:

From over 1 million raw records, only **2,955 unique victim entries** were identified and inserted into the Victim_Dimension table. This ensured that the dimension holds only necessary and distinct data for future reference and analysis.

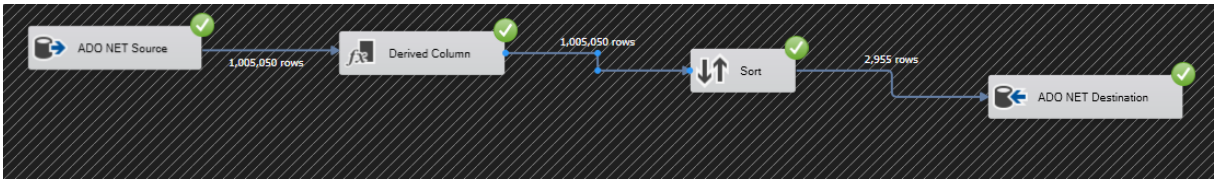


Figure 26:SSIS data flow showing successful transformation and load

Fact Crime Population

1. ADO.NET Source Configuration

In the first step, a SQL command is used inside the **ADO.NET Source Editor** to select and prepare the raw data that will be transformed and loaded into the data warehouse. This SQL script plays a crucial role because it determines what data is fetched from the source system.

The query starts by selecting the **DR_NO** column, which is the unique identifier of each crime record, and other important columns such as **Date_Rptd**, **DATE_OCC**, and **TIME_OCC**, which represent when the crime was reported and when it occurred. These values are needed for time-based analysis.

The query also includes **AREA**, **AREA_NAME**, and **Rpt_Dist_No**, which are geographic attributes that help in identifying the location of the incident. These values are used later in the **Location Dimension** lookup.

Next, **Part_1_2** and **Crm_Cd** are selected, along with a casted version of **Crm_Cd_Desc**, which contains a description of the crime. The casting is used to ensure proper data types and lengths when loaded into SSIS.

Then, **Mocodes**, **Weapon_Used_Cd**, and a trimmed version of **Weapon_Desc** are included. These columns are related to the crime's method and weapon, and are needed for reporting and possibly linking with other attributes.

Finally, the query includes victim-related data like **Vict_Age**, and casts **Vict_Sex** and **Vict_Descent** into defined string lengths to standardize the values. Similarly, **Status**, **Status_Desc**, and **Crm_Cd_1**, **Crm_Cd_2** are added to capture extra codes and crime status, which are necessary for the **Status Dimension**.

By writing this SQL command directly, we have more control over which data is pulled and how it's formatted before it even enters the SSIS pipeline. This helps to reduce unnecessary data load and ensures a cleaner, more structured ETL process.

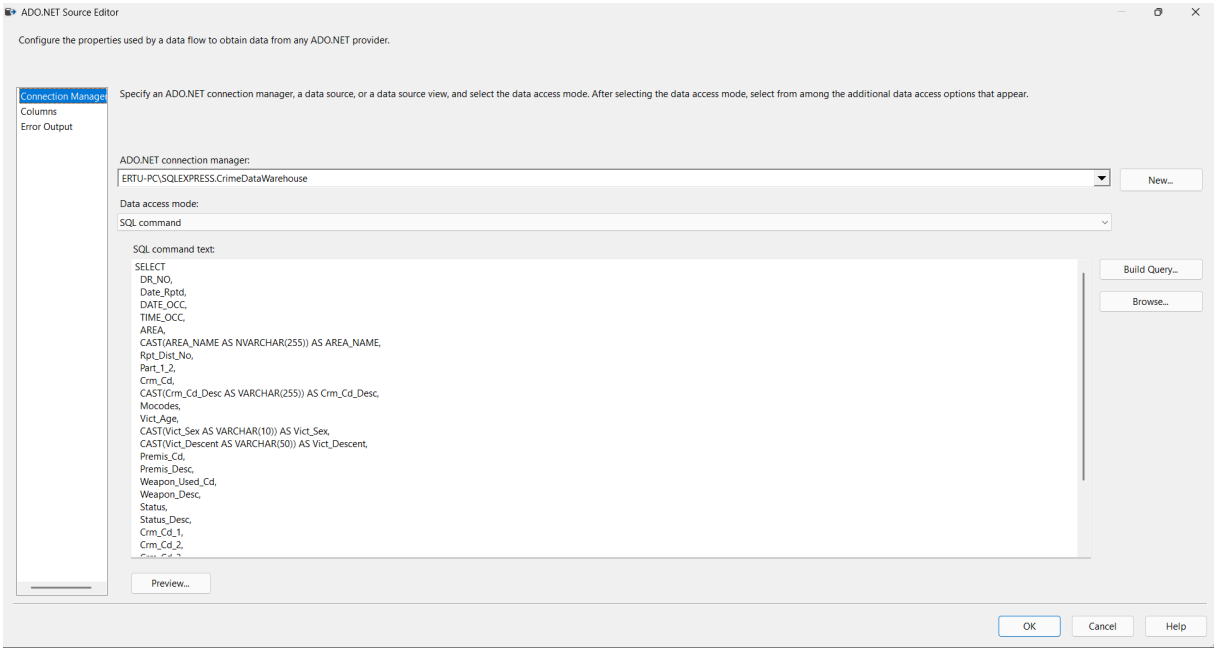


Figure 27:ADO.NET Source Configuration for Fact Crime Table

2. Derived Column Transformations Explanation

In this step, we use the **Derived Column Transformation** to clean, convert, and prepare some columns before loading them into the Fact_Crime table. Each transformation here helps improve data quality or convert data types to match the destination table format.

a. Time_OCC_CLEAN

Expression:
(DT_DBTIMESTAMP)(SUBSTRING(RIGHT("0000" + (DT_WSTR,4)[TIME_OCC],4),1,2) + ":" + SUBSTRING(RIGHT("0000" + (DT_WSTR,4)[TIME_OCC],4),3,2))

Explanation:
The TIME_OCC column stores time in 4-digit format like 1530 (meaning 3:30 PM). But this is not directly usable in datetime formats.
This formula first pads the time with zeros ("0000" + TIME_OCC → "001530"), then extracts the hour and minute using SUBSTRING. The result is a time string like "15:30", which is then cast to a proper **datetime** value using DT_DBTIMESTAMP.
This helps the time data match the Time_OCC column in the **Time Dimension**.

b. AREA_STR

Expression:
(DT_STR,50,1252)[AREA]

Explanation:
The AREA column is converted into a string with a maximum length of 50 characters using ANSI code page 1252. This ensures that AREA can be matched correctly during the lookup and loaded into the Fact Table or Location Dimension if needed.

c. Weapon_Used

Expression:
ISNULL(Weapon_Used_Cd) || TRIM((DT_WSTR,10)[Weapon_Used_Cd]) == "" ? "False" : "True"

Explanation:
This is a logical expression to determine if a weapon was used in the crime.

- If Weapon_Used_Cd is **null** or an **empty string**, it returns "False"
- Otherwise, it returns "True"
  This creates a simplified binary value indicating whether a weapon was involved in the incident, which is easier to analyze.

d. Date_OCC_CLEAN

Expression:
(DT_DBDATE)[DATE_OCC]

Explanation:
The original DATE_OCC column is cast to a proper **date format (DT_DBDATE)**, stripping away any time part if it exists. This ensures compatibility with the **Date_OCC** field in the Fact Table and allows accurate joining with the Time Dimension.

e. AREA_NAME_CLEAN

Expression:
(DT_STR,100,1252)[AREA_NAME]

Explanation:
Converts the area name to a standard string type with 100 characters max. This may be useful for sorting, joining, or lookups in the **Location Dimension**.

f. Crm_Cd_Desc1

Expression:
(DT_STR,255,1252)[Crm_Cd_Desc]

Explanation:
Crime description (“BATTERY - SIMPLE ASSAULT”) is cleaned and converted to string format with up to 255 characters to ensure consistency before lookups or storage.

g. Part_1_2_Clean

Expression:
(DT_STR,10,1252)[Part_1_2]

Explanation:
This field contains values like “Part I” or “Part II” to classify the crime. It's converted to a string type for consistency in the Fact Table.

h. Vict_Sex_Clean

Expression:
(DT_STR,10,1252)[Vict_Sex]

Explanation:
Converts the victim’s sex into a standard string format for lookup and reporting.

i. Vict_Descent_Clean

Expression:
(DT_STR,50,1252)[Vict_Descent]

Explanation:
Converts the descent (ethnic background) code of the victim into string format. This helps in matching or categorizing records in the Victim Dimension.

j. Status_Code_Clean

Expression:
(DT_STR,10,1252)[Status]

Explanation:
Cleans and standardizes the Status code of the crime (like “IC” or “AO”) into string format.

k. Status_Desc_Clean

Expression:
(DT_STR,255,1252)[Status_Desc]

Explanation:
Provides a readable description of the crime status (like “Investigation Continued”) in a clean string format.

l. DR_NO_Clean

Expression:
(DT_I4)[DR_NO]

Explanation:
Casts the **DR_NO** (Report Number) to a 4-byte signed integer, which ensures it can be used as a **primary key (Crime_ID)** in the Fact Table. This is essential for indexing and relationships.

These transformations help convert, clean, and prepare raw input data so it can match with the corresponding **dimension tables** during lookups. They ensure consistency in data types, fix formatting issues, and simplify complex or inconsistent values — making the overall ETL process more robust and accurate.

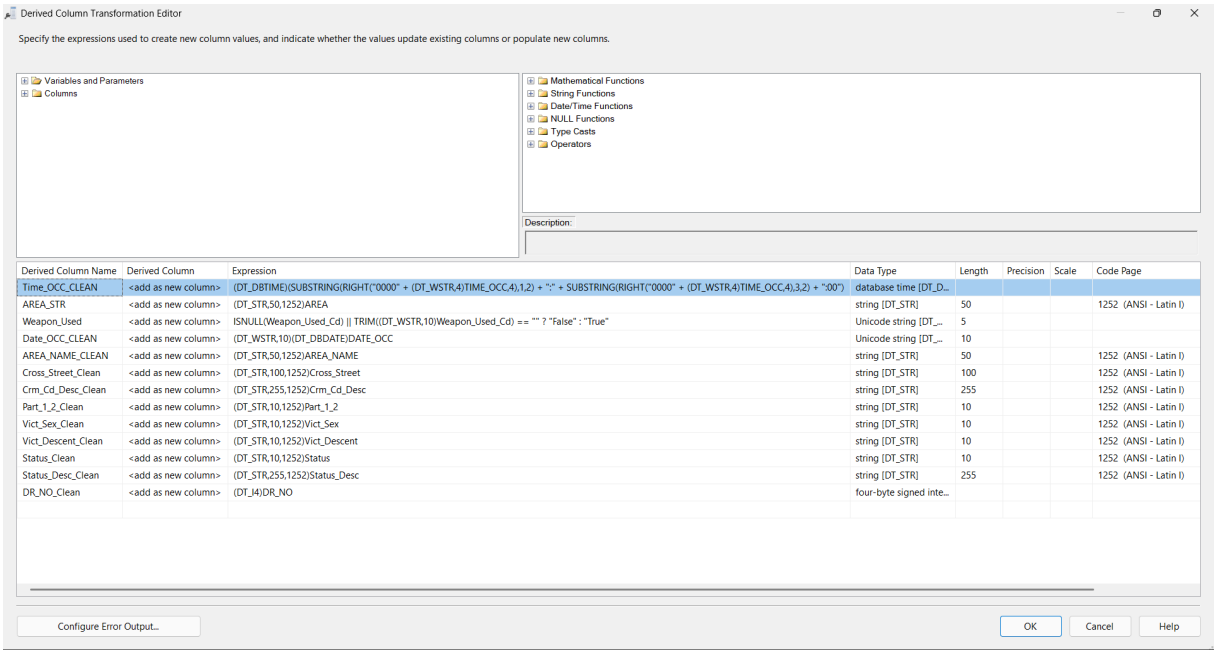


Figure 28:Derived Column Transformations for Crime Fact Table

3. Time Dimension Lookup Configuration (General Tab)

This image shows the **Lookup Transformation Editor** for the **Time_Dim_Lookup** component. The goal of this step is to match the time information from the source data to an existing **Time Dimension** table. It uses the OLE DB connection manager to connect to the data warehouse. Full cache mode is used for performance, and unmatched rows are redirected. This setup is crucial for maintaining referential integrity when inserting foreign keys into the Fact_Crime table.

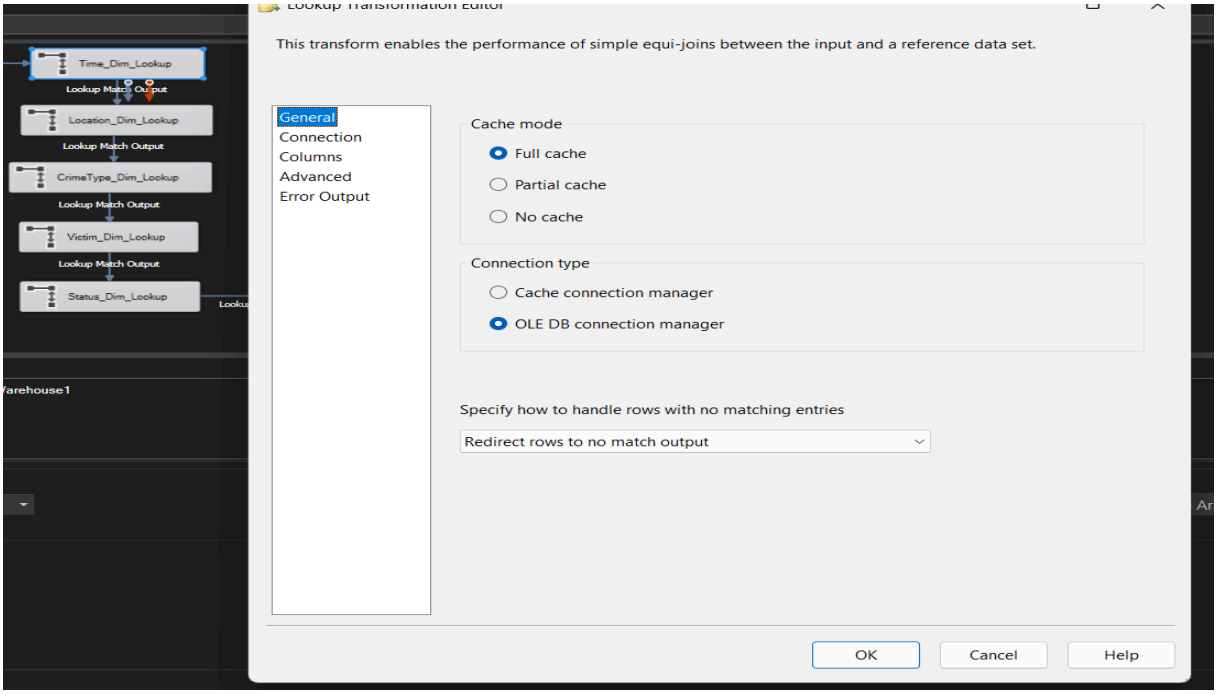


Figure 29:Time Dimension Lookup General

4. Time Dimension Lookup (Columns Tab)

Here, we match the source Date_OCC_CLEAN and Time_OCC_CLEAN values to the Date_OCC and Time_OCC columns in the Time Dimension. The matching Time_ID from the dimension table is returned and added as a new column. This step allows us to use the correct **Time_ID** foreign key when inserting data into Fact_Crime.

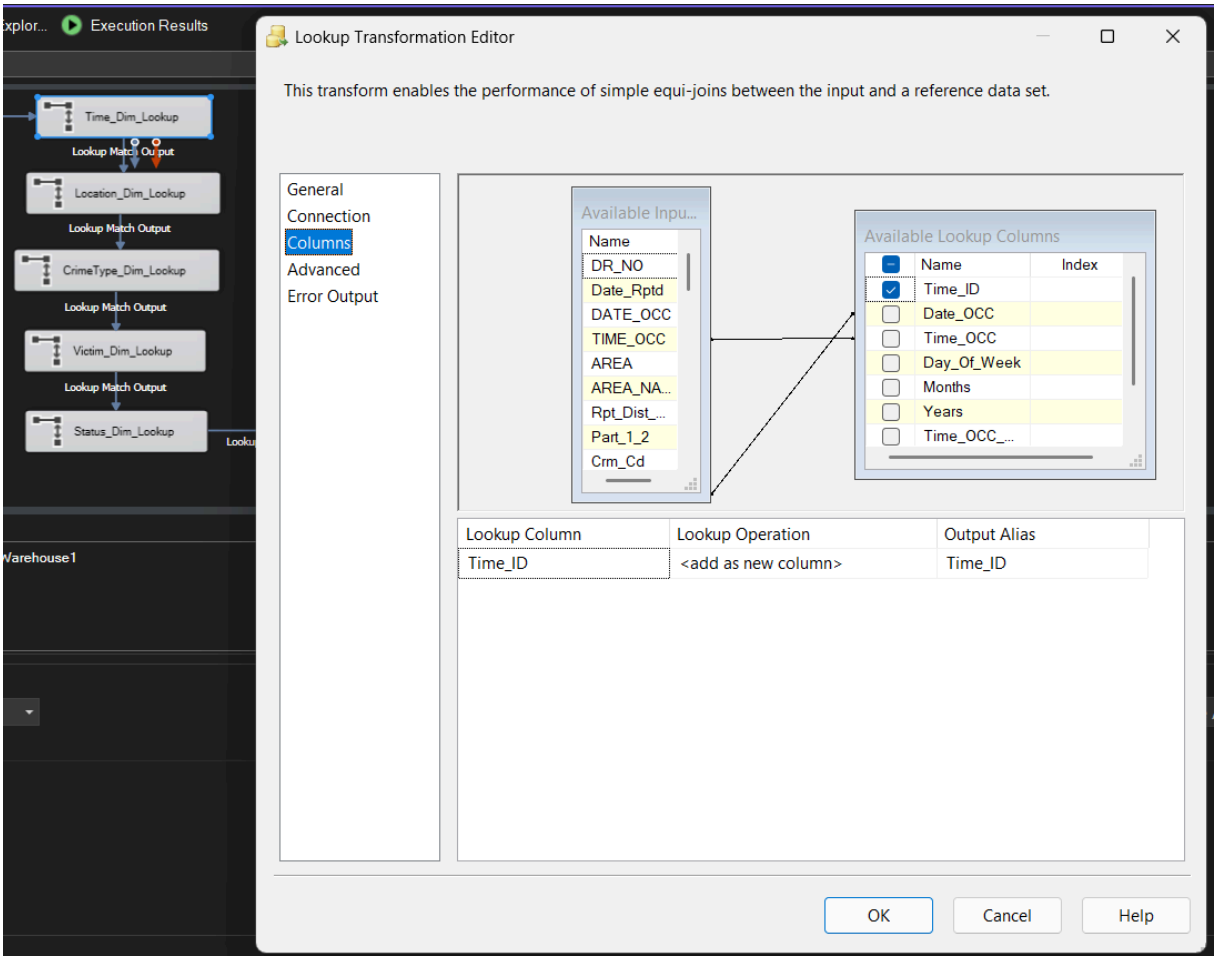


Figure 30:Time Dimension Lookup Columns Tab

5. Location Dimension Lookup (General Tab)

Similar to the time lookup, this lookup is for matching **location data**. The Location_Dim_Lookup transformation connects to the Location_Dimension table. Again, full cache mode is used to improve performance, and unmatched records are redirected to manage errors cleanly. This lookup helps in connecting our source data to the correct location entries.

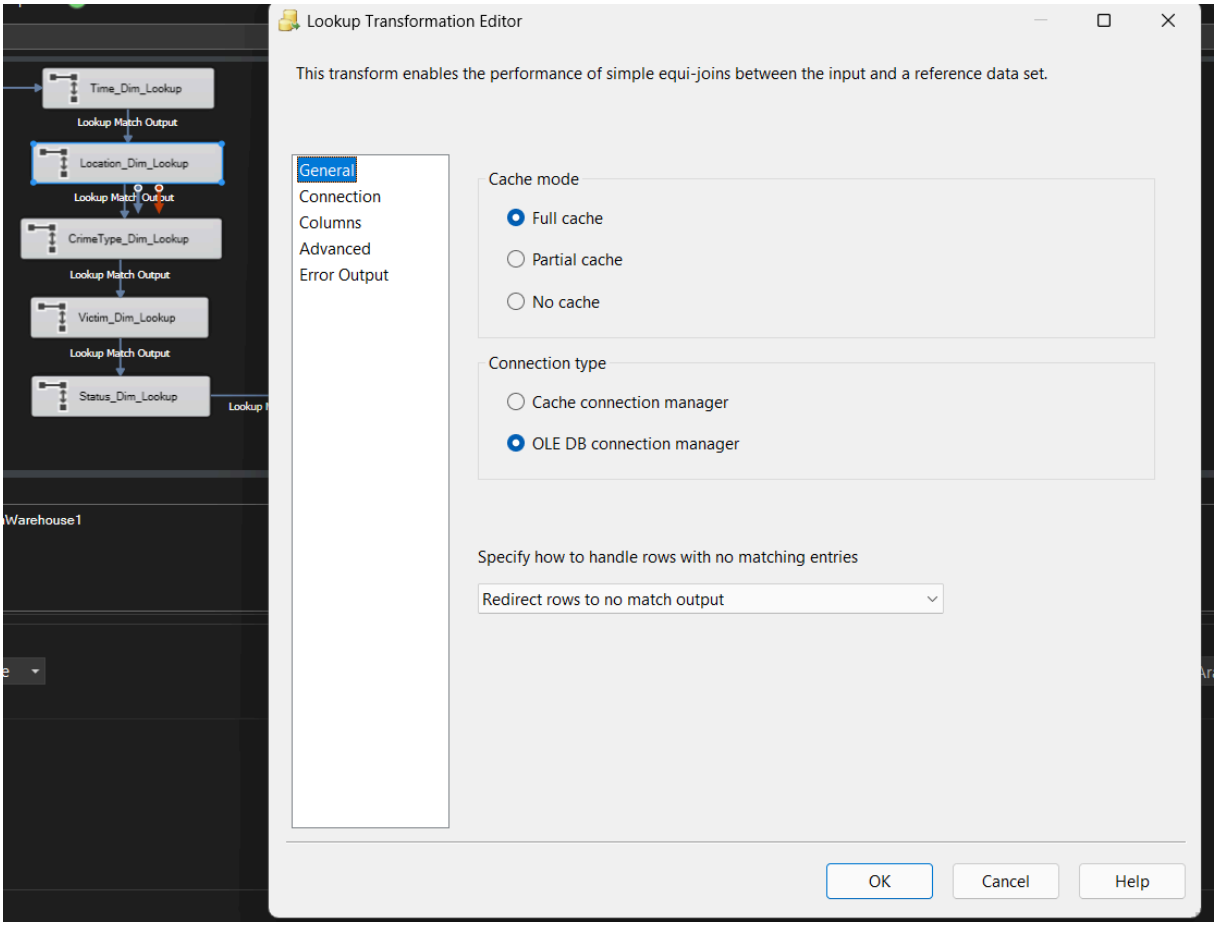


Figure 31:Location Dimension Lookup (General Tab)

6. Location Dimension Lookup (Connection Tab)

The connection is made to the [dbo].[Location_Dimension] table in the same CrimeDataWarehouse database. This selection enables the lookup to fetch location IDs based on multiple matching fields like AREA, AREA_NAME, LAT, and LON, ensuring precise matching for locations.

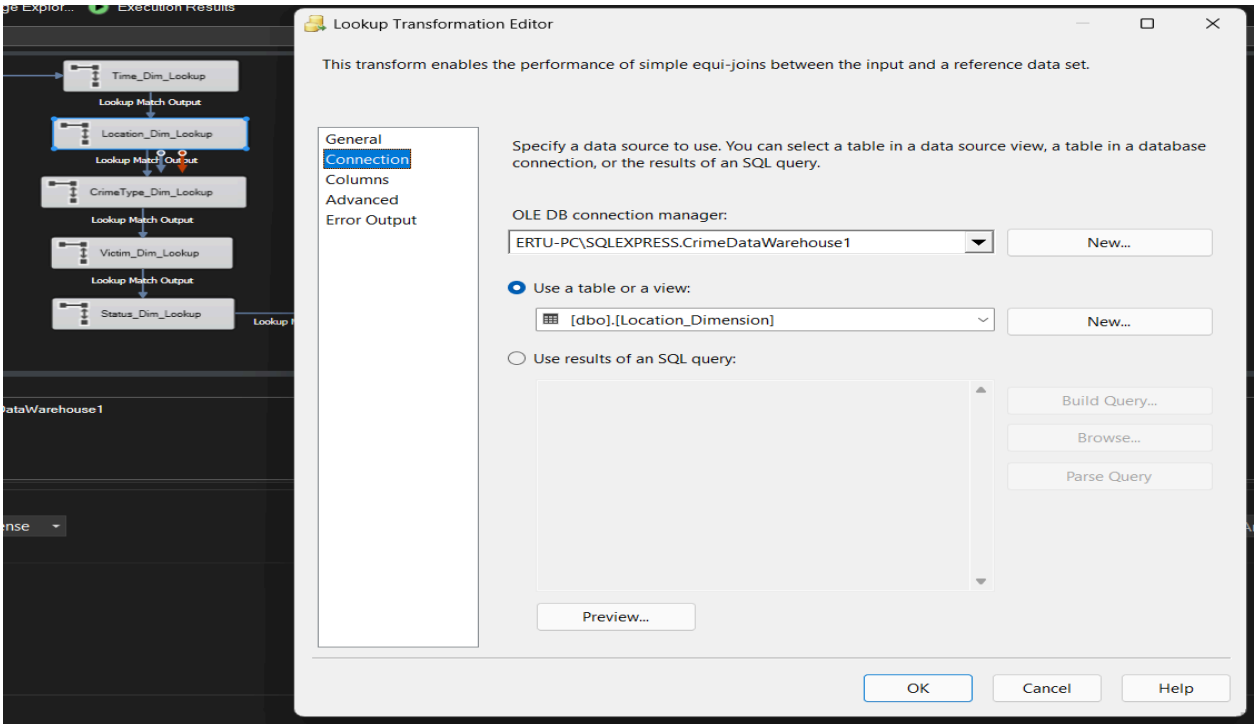


Figure 32:Location Dimension Lookup (Connection Tab)

7. Location Dimension Lookup (Columns Tab)

In this step, fields such as AREA, AREA_NAME, LAT, and LON from the source are matched with their counterparts in the dimension table. When a match is found, the corresponding Location_ID is returned and attached as a foreign key. This ensures geographic data is normalized in the Fact table.

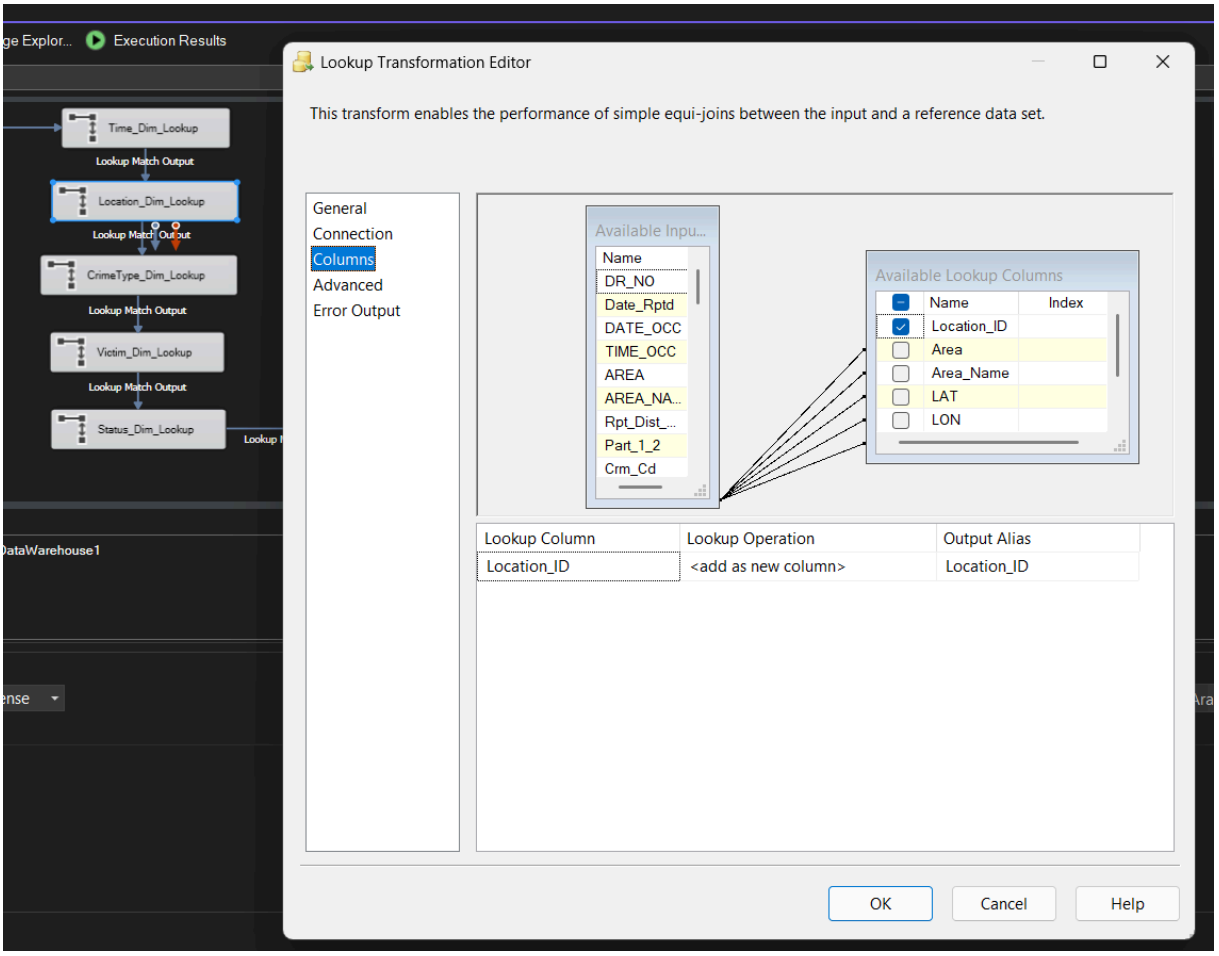


Figure 33:Location Dimension Lookup (Columns Tab)

8. Crime Type Dimension Lookup (Connection Tab)

This step connects the source crime-related data to the CrimeType_Dimension table. The lookup will find the appropriate CrimeType_ID for each combination of Crm_Cd and Crm_Cd_Desc, which were cleaned earlier. Matching to dimensions like this reduces redundancy and improves query performance in the warehouse.

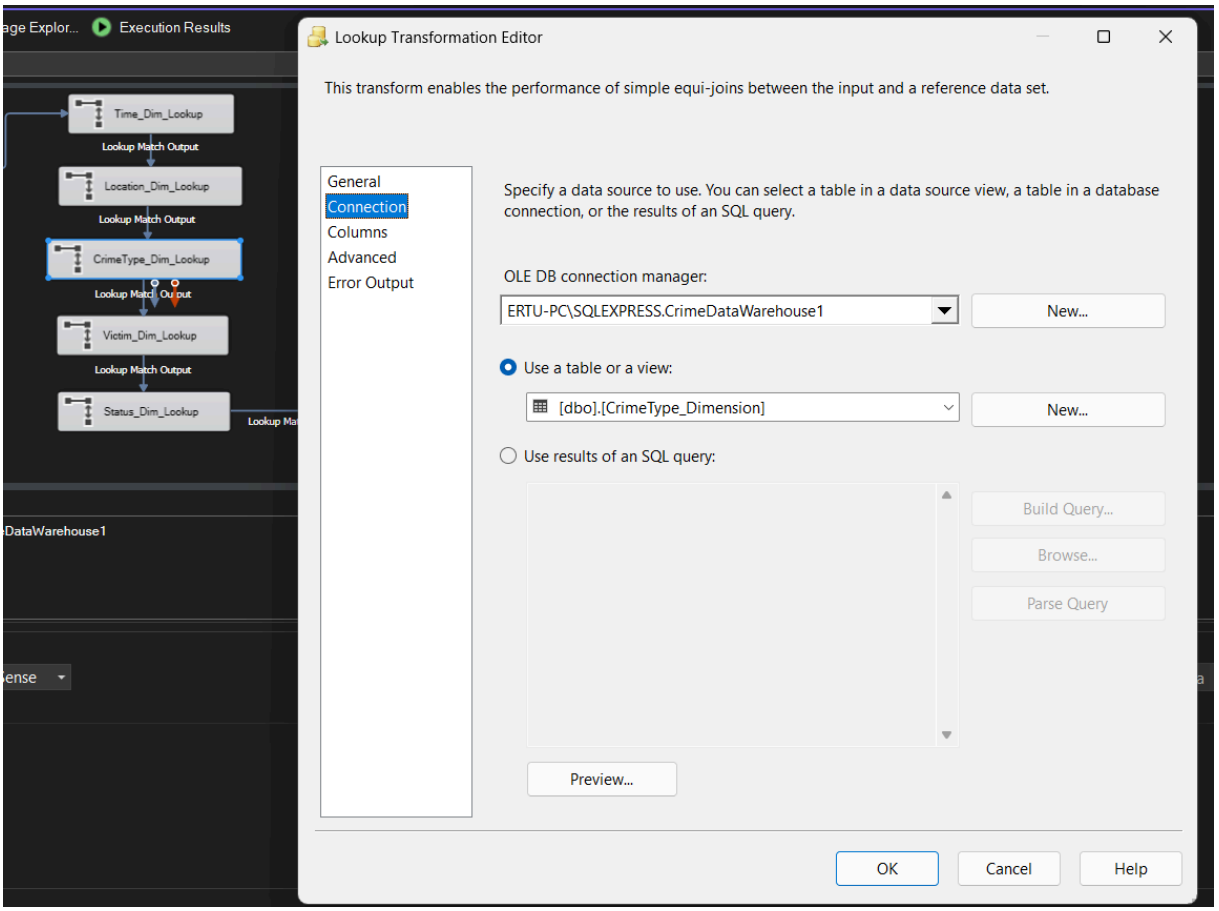


Figure 34:Crime Type Dimension Lookup (Connection Tab)

9. Crime Type Dimension Lookup (Columns Tab)

Here, Crm_Cd and Crm_Cd_Desc are matched against the lookup table. If a match is found, the system returns the associated CrimeType_ID. This ID will then be stored in the Fact_Crime table to replace the verbose crime description data, which allows for more efficient joins and aggregations.

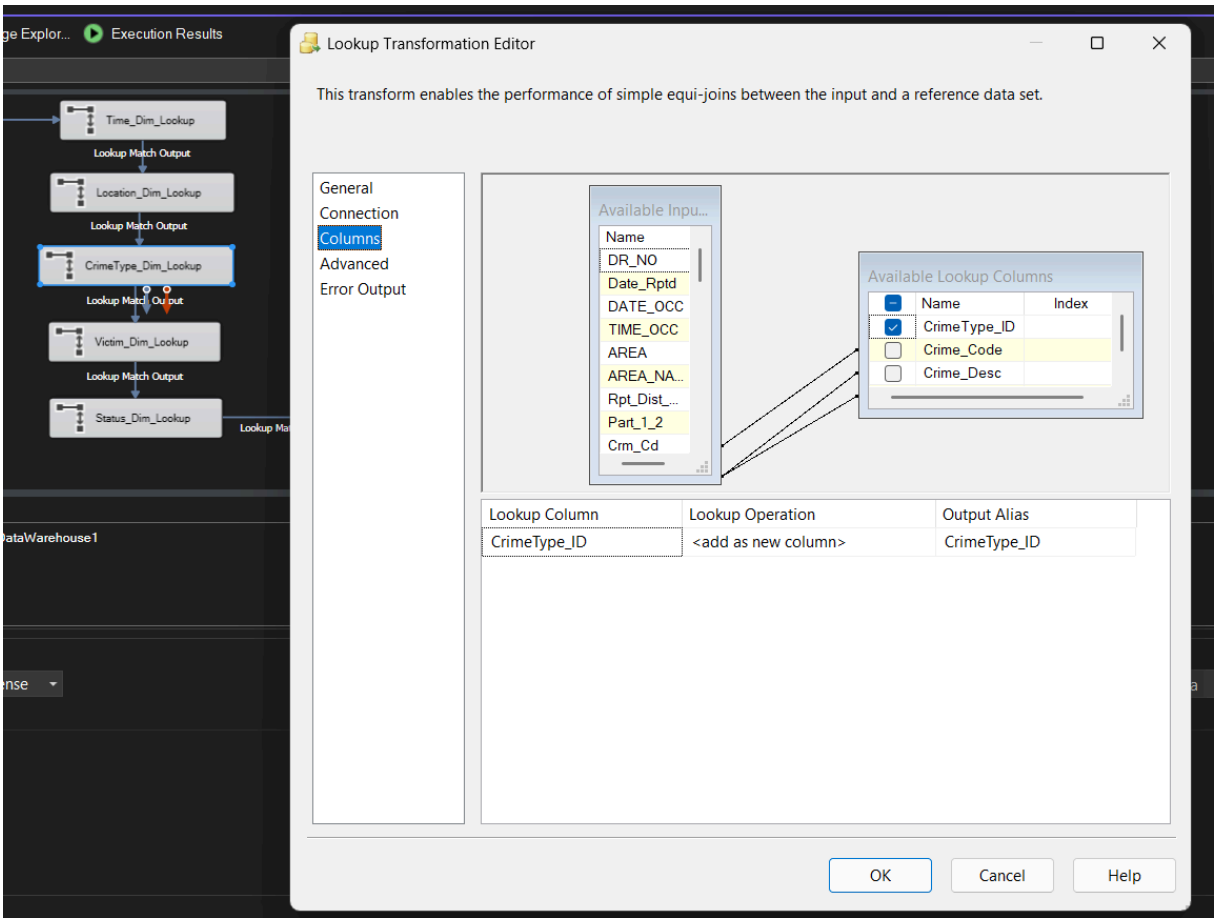


Figure 35:Crime Type Dimension Lookup (Columns Tab)

10. Victim Dimension Lookup (Connection Tab)

In this screen, the **Victim_Dimension** table is used for a lookup. The goal is to replace detailed victim-related data like sex, descent, and age with a **Victim_ID** from the dimension table. Doing this keeps the Fact table compact and enforces consistency.

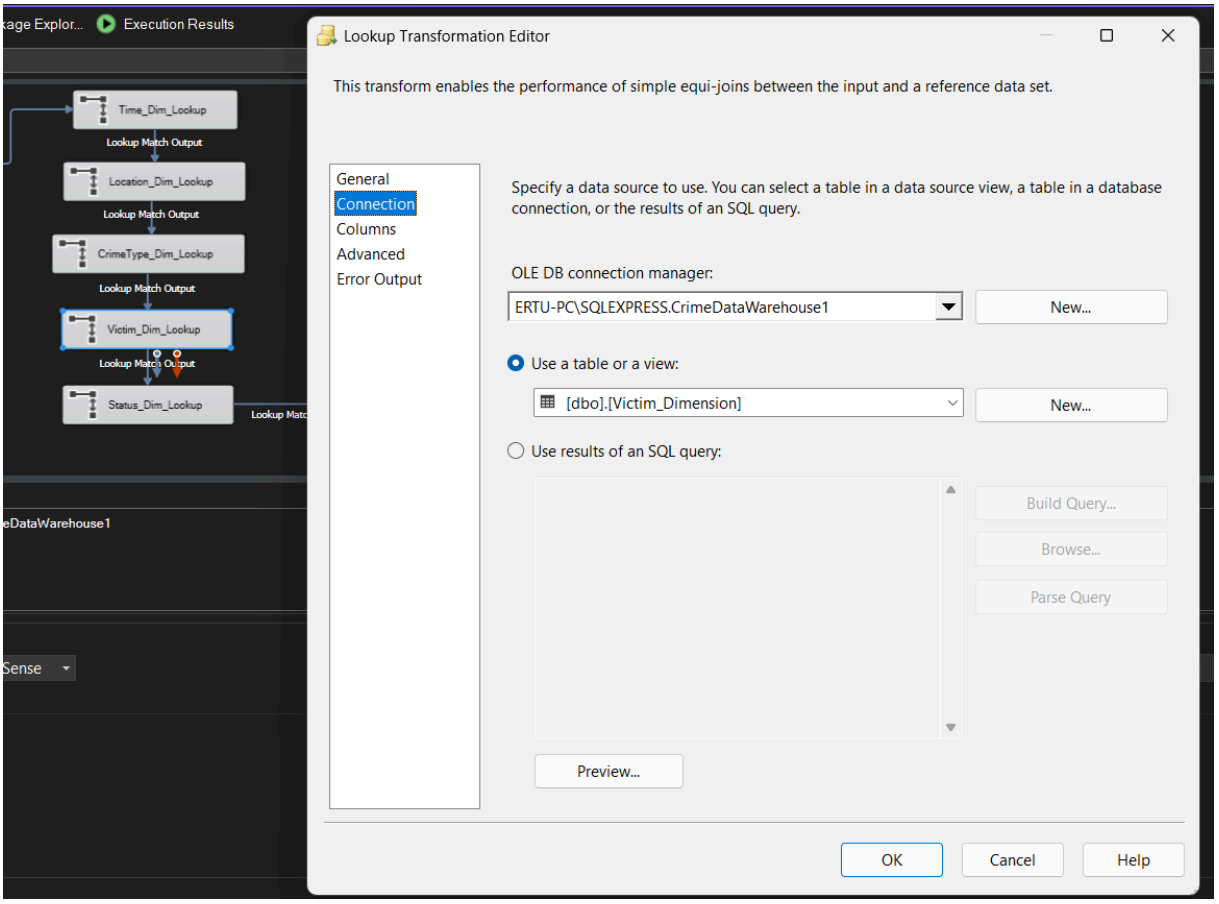


Figure 36:Victim Dimension Lookup (Connection Tab)

11. Victim Dimension Lookup (Columns Tab)

We match Vict_Sex, Vict_Descent, and Vict_Age from the cleaned source data to the Victim Dimension. When a match is found, the corresponding Victim_ID is retrieved. This approach is especially helpful if multiple crimes share the same victim characteristics, promoting reuse and reducing duplication.

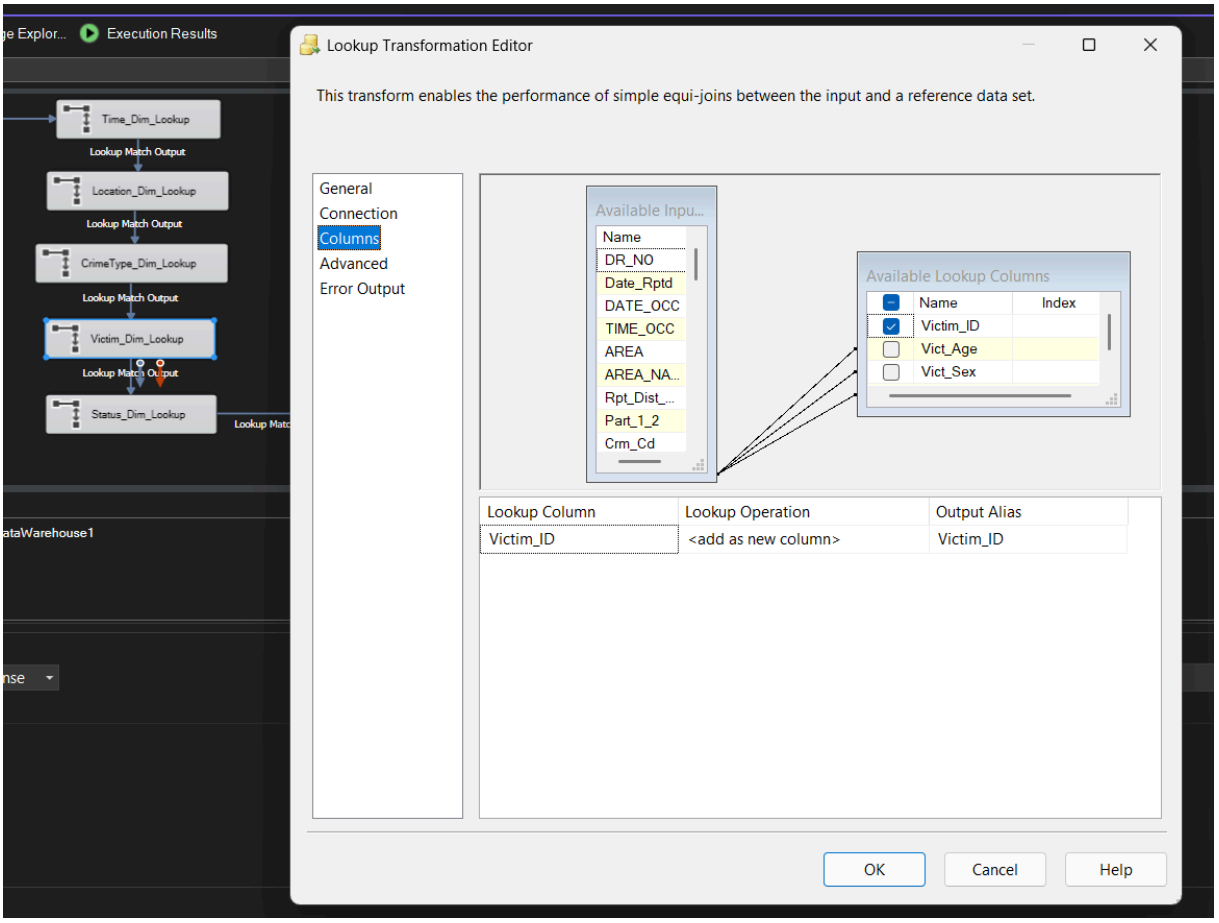


Figure 37:Victim Dimension Lookup (Columns Tab)

12. Status Dimension Lookup (Connection Tab)

This screen shows the setup for connecting to the **Status_Dimension** table. We use the OLE DB connection manager again to perform the lookup. This ensures we can identify the appropriate Status_ID for each crime based on the status code and description.

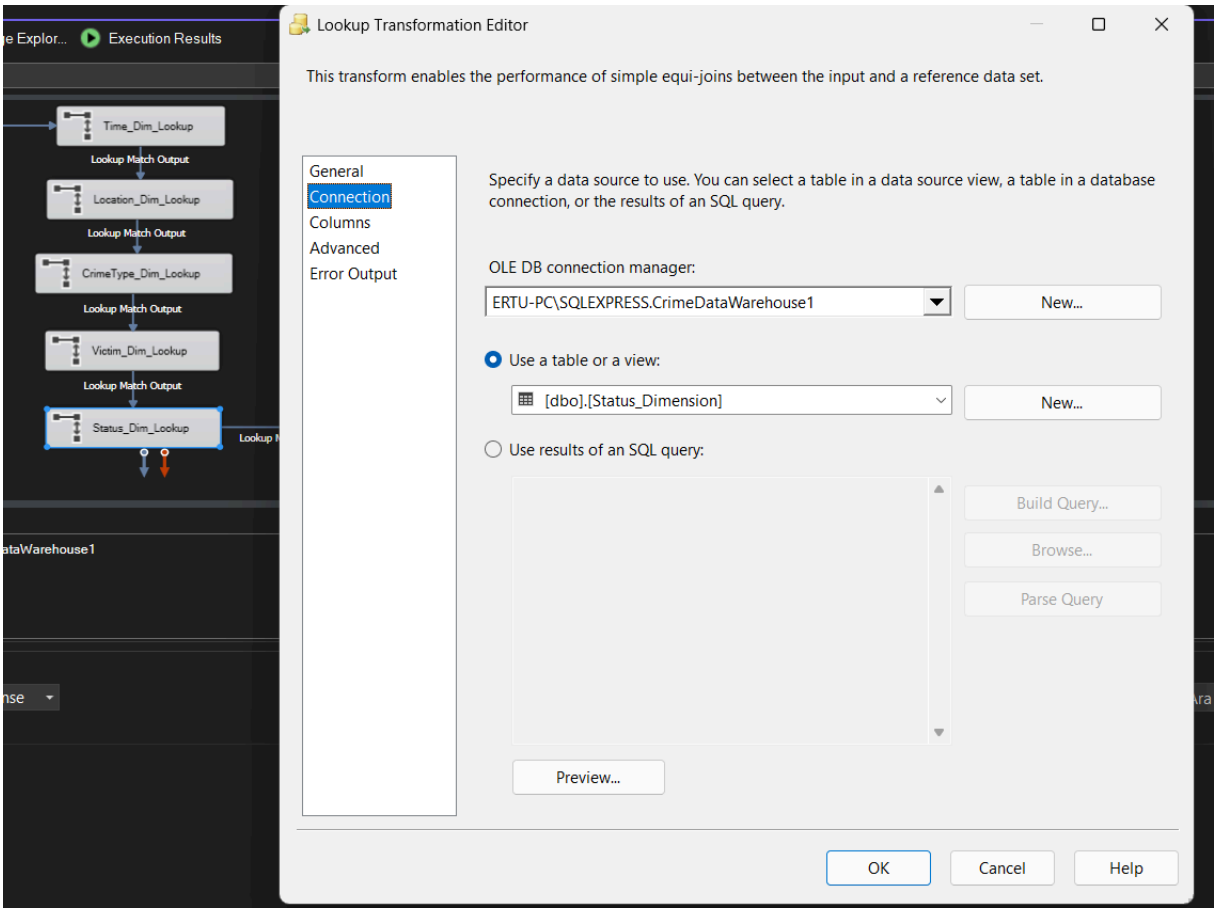


Figure 38:Status Dimension Lookup (Connection Tab)

13. Status Dimension Lookup (Columns Tab)

The source columns Status_Code_CLEAN and Status_Desc_CLEAN are matched to their counterparts in the dimension. If a match is found, the lookup returns the associated Status_ID, which is then used in the Fact table instead of storing duplicate status strings.

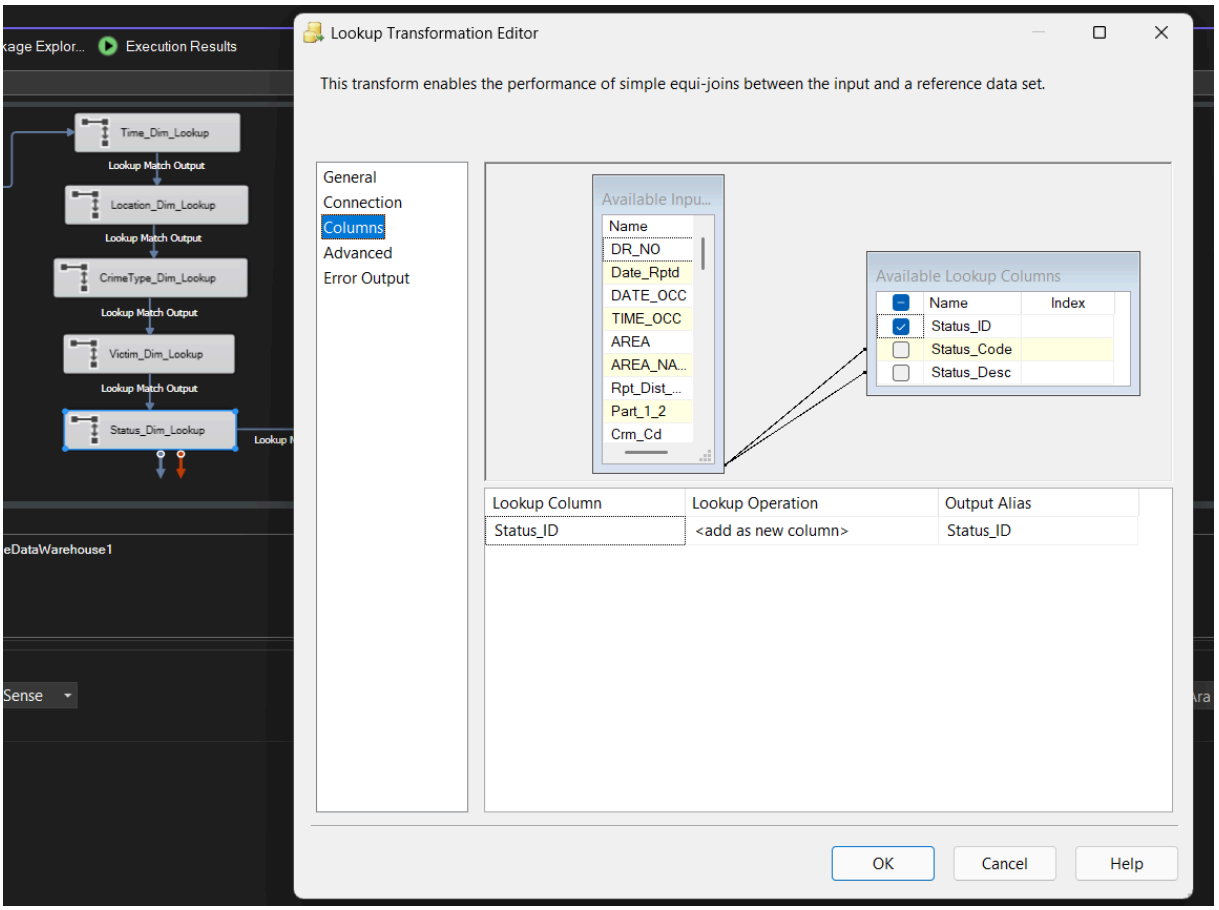


Figure 39:Status Dimension Lookup (Columns Tab)

14. ADO.NET Destination Setup for Fact_Crime

Now that all required foreign keys (Time_ID, Location_ID, CrimeType_ID, Victim_ID, Status_ID) are prepared through lookup transformations, the **ADO.NET Destination** is configured to insert data into the Fact_Crime table. This screen sets the target table and connection settings.

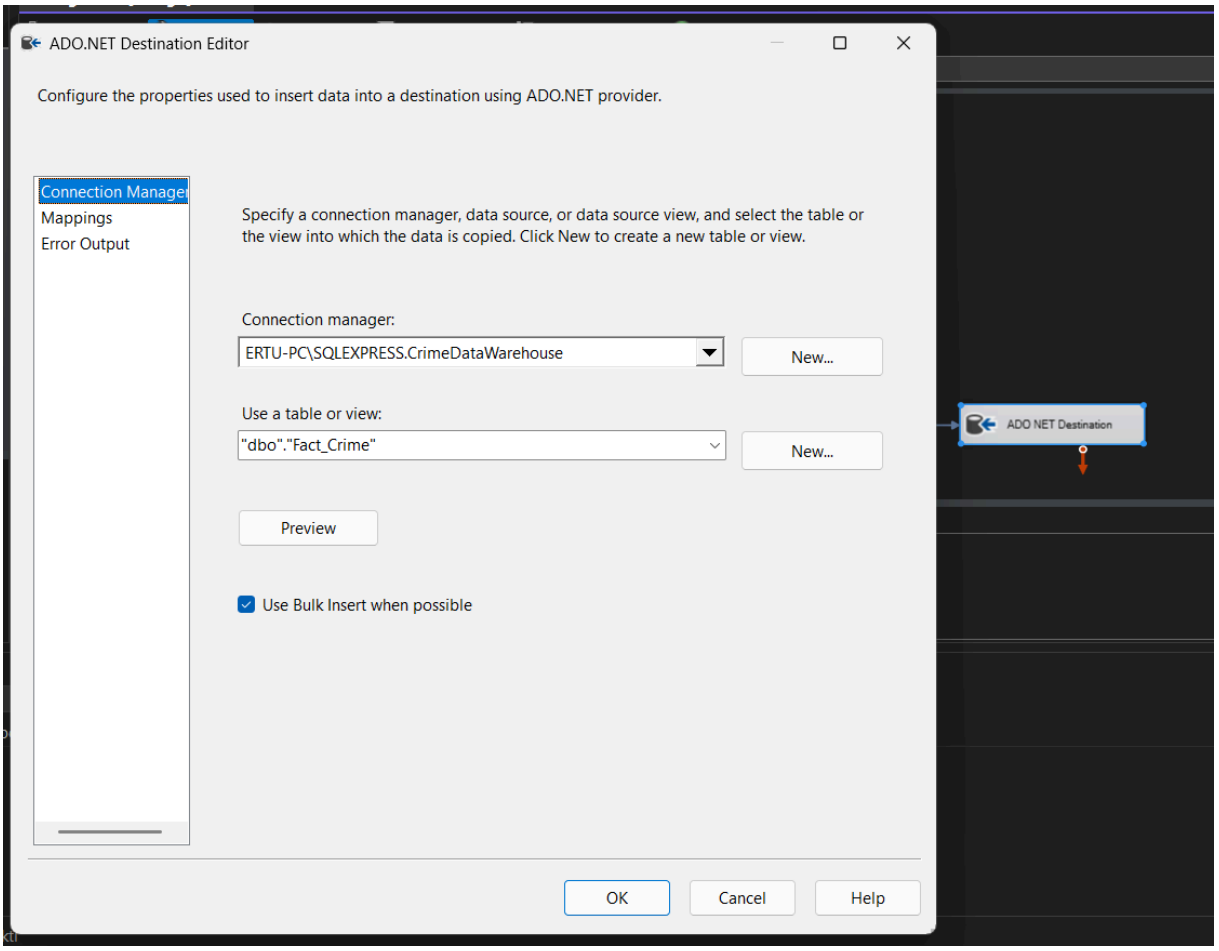


Figure 40:ADO.NET Destination Setup for Fact_Crime

15. Fact_Crime Column Mapping

The input columns from all previous steps are now mapped to the destination columns in the Fact_Crime table. For instance, DR_NO_Clean becomes Crime_ID, while the dimension keys and cleaned date/time fields are matched to their respective Fact table columns. This mapping ensures accurate data loading and relational links.

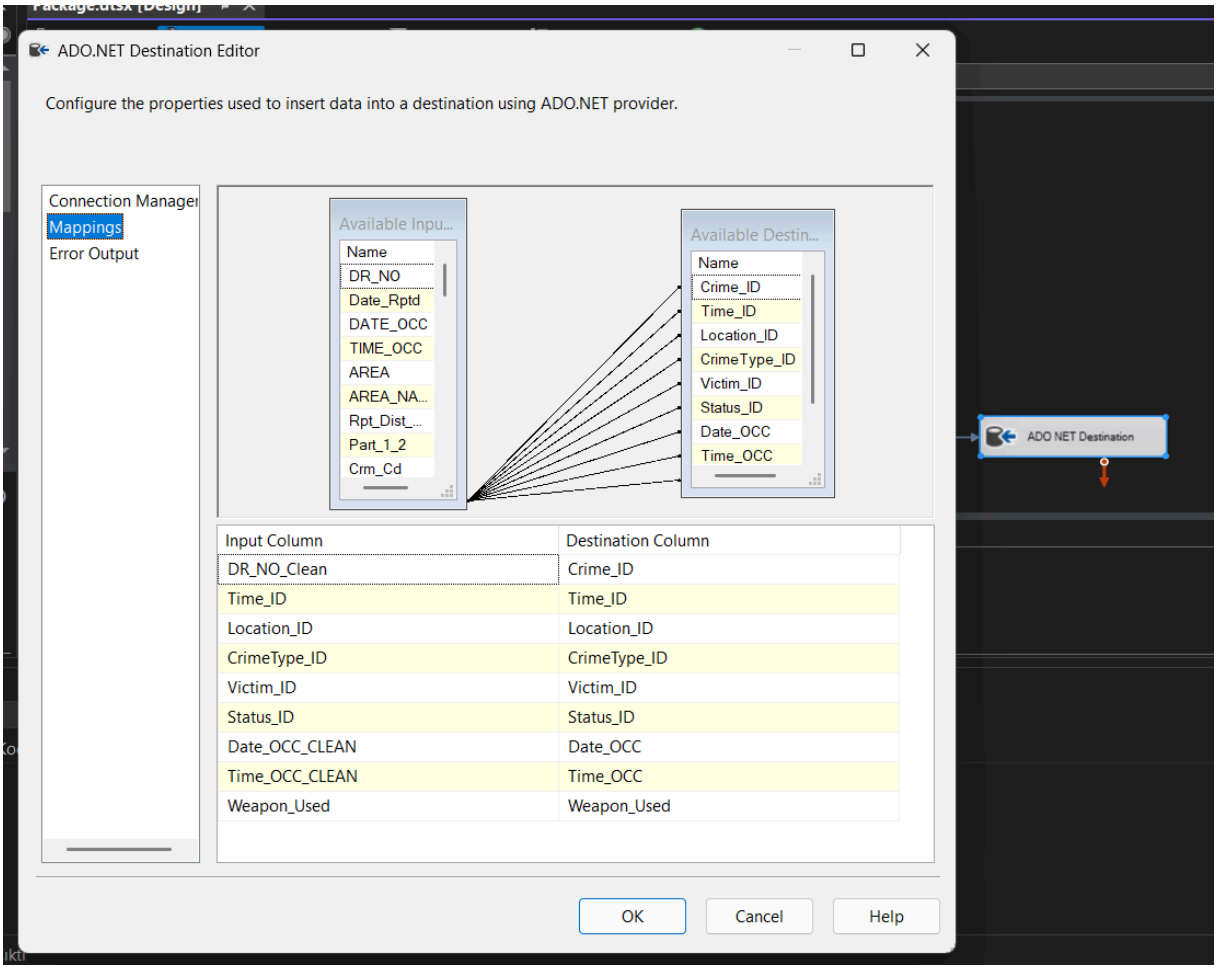


Figure 41:Fact_Crime Column Mapping

16. Final Data Flow Overview

This is the final visualization of the complete data flow. It begins with the ADO.NET source, then flows through a Derived Column transformation, followed by five sequential **Lookup** transformations for Time, Location, Crime Type, Victim, and Status dimensions. Finally, the cleaned and enriched data is loaded into the Fact_Crime table. This logical flow ensures the Fact table is both normalized and connected to all relevant dimensions, making it ready for efficient querying and reporting.

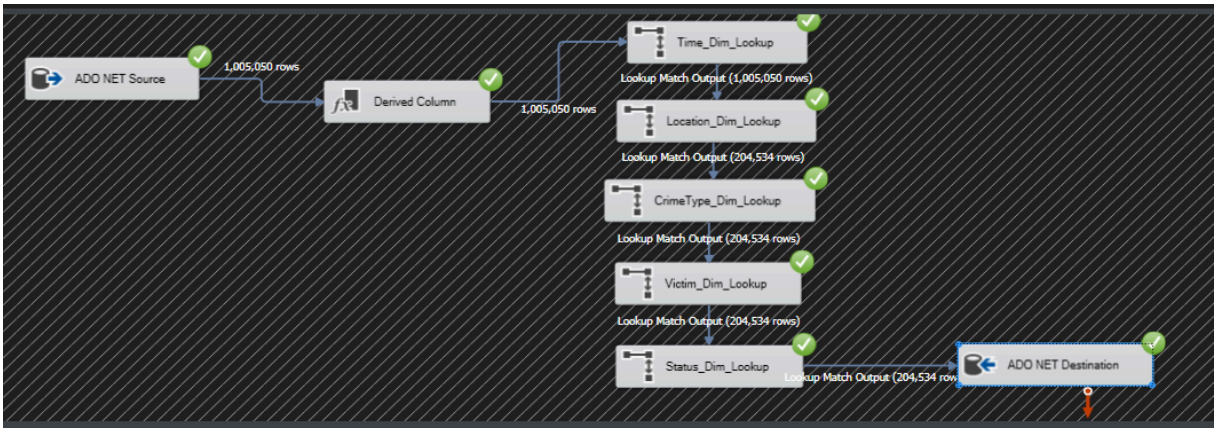


Figure 42:Final Data Flow Overview

After a confirmation of data, our population process is completed.

Crime_ID	Time_ID	Location_ID	CrimeType_...	Victim_ID	Status_ID	Date_OCC	Time_OCC	Weapon_Us...
2113	461598	34823	58	19	5	2021-04-05	18:35:00	False
2401	685851	45322	58	19	5	2023-12-21	19:30:00	False
10304468	375869	57507	63	1135	3	2020-01-08	22:30:00	True
190101086	374402	4478	63	659	5	2020-01-01	03:30:00	True
190326475	386186	76884	58	45	2	2020-03-01	21:30:00	False
191921269	374407	27415	81	951	5	2020-01-01	04:15:00	False
200100507	374979	5121	18	562	5	2020-01-04	02:00:00	False
200100515	375621	4372	68	61	5	2020-01-07	16:38:00	True
200100520	375829	4994	41	592	5	2020-01-08	18:05:00	False
200100536	376934	5235	63	42	3	2020-01-14	14:00:00	True
200100538	376972	4372	18	935	5	2020-01-14	17:30:00	False
200100543	377132	4372	41	750	5	2020-01-15	14:45:00	False
200100556	378014	4405	3	369	5	2020-01-20	04:00:00	True
200100565	379297	4372	81	50	2	2020-01-26	19:45:00	False
200100567	378494	4186	66	146	3	2020-01-22	16:00:00	True
200100569	378494	4186	66	72	3	2020-01-22	16:00:00	True
200100573	379908	4994	63	612	3	2020-01-29	21:50:00	True
200100574	379888	4372	41	50	5	2020-01-29	19:30:00	False
200100596	381808	4372	41	1110	5	2020-02-08	16:00:00	False
200100599	382217	4248	63	506	5	2020-02-10	15:55:00	True
200100609	383704	4556	5	798	2	2020-02-18	13:00:00	True
200100613	383984	5235	5	533	2	2020-02-19	21:30:00	True
200100622	384561	4372	41	1158	5	2020-02-22	17:10:00	False
200100623	384634	4343	63	1178	5	2020-02-23	01:30:00	True
200100628	385216	4315	6	1731	5	2020-02-26	01:00:00	True
200100643	385394	5039	65	1178	3	2020-02-26	22:30:00	True
200100650	386171	4293	81	61	6	2020-03-01	19:30:00	False
200100663	381073	4172	63	373	2	2020-02-04	18:40:00	True

Figure 43:Confirmed Sample Data

3. SSRS Reports

Question1: Which locations have the most crime?

Shared Data Source Configuration

In this step, we define a shared data source called **Crime_WareHouse**. This source connects our SSRS reports to the **CrimeDataWarehouse** SQL database. We select the connection type as **Microsoft SQL Server** and provide the connection string. This helps reports retrieve data efficiently without redefining the connection every time.

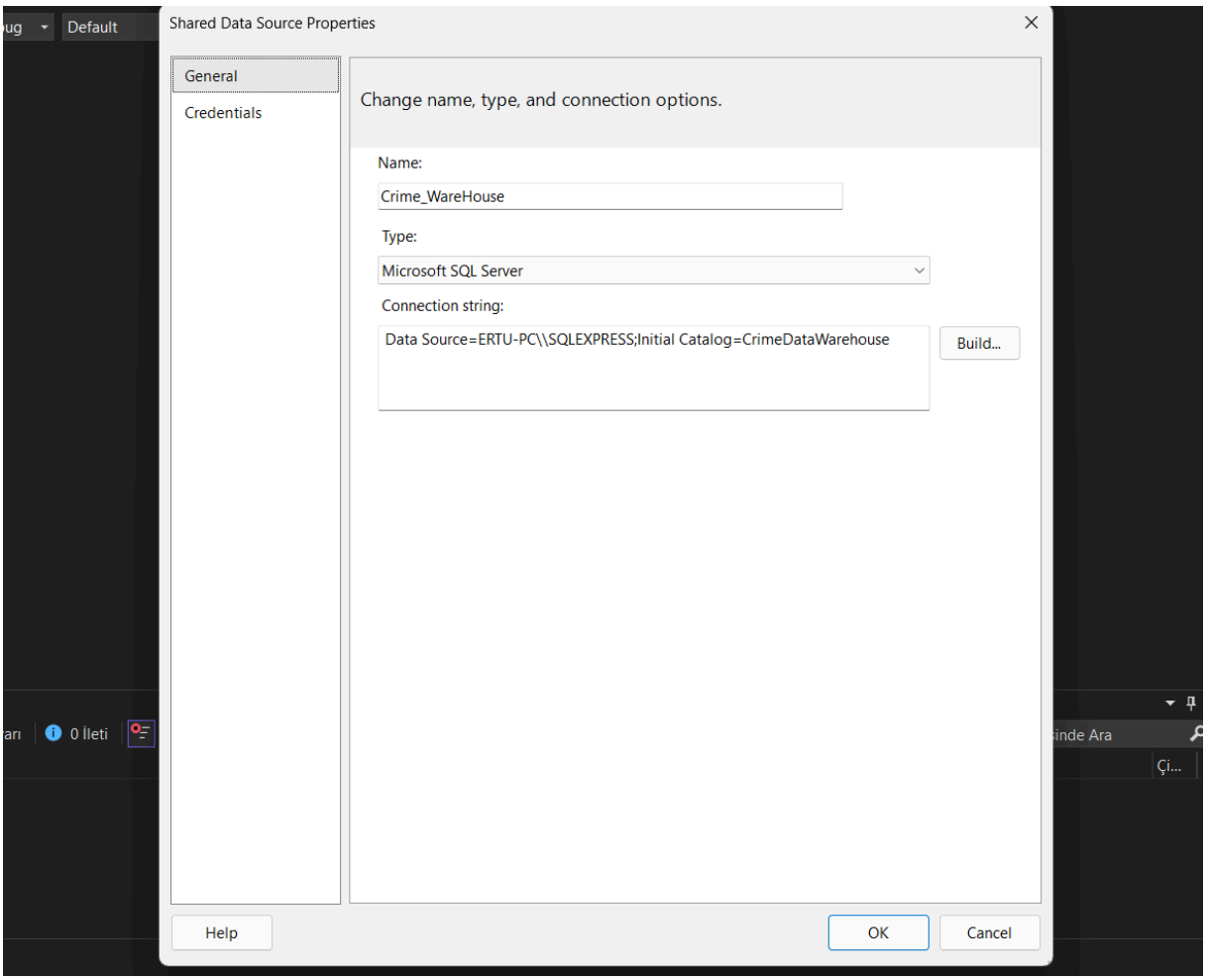


Figure 44:Shared Data Source Configuration

Selecting Data Source

Here, we select the previously created shared data source (**Crime_WareHouse**) to fetch data for the report. Using a shared data source improves consistency and makes it easier to update connections in the future.

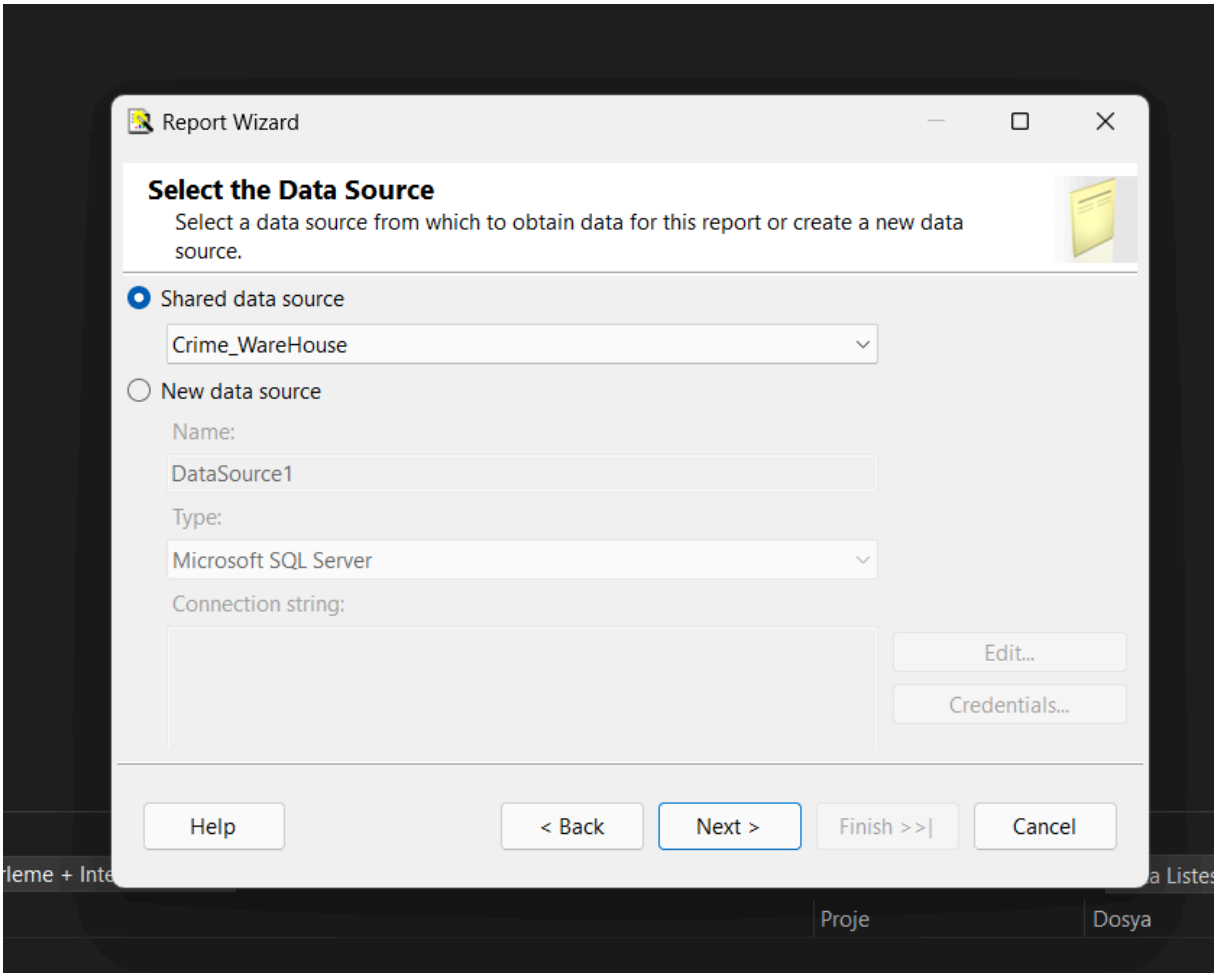


Figure 45:Selecting Data Source

Designing the Query

At this stage, we write a SQL query that groups and counts crime records by area. The query joins the **Fact_Crime** fact table with the **Location_Dimension** table to retrieve area names, then counts how many crimes occurred in each area. Results are sorted in descending order to show the most crime-heavy areas at the top.

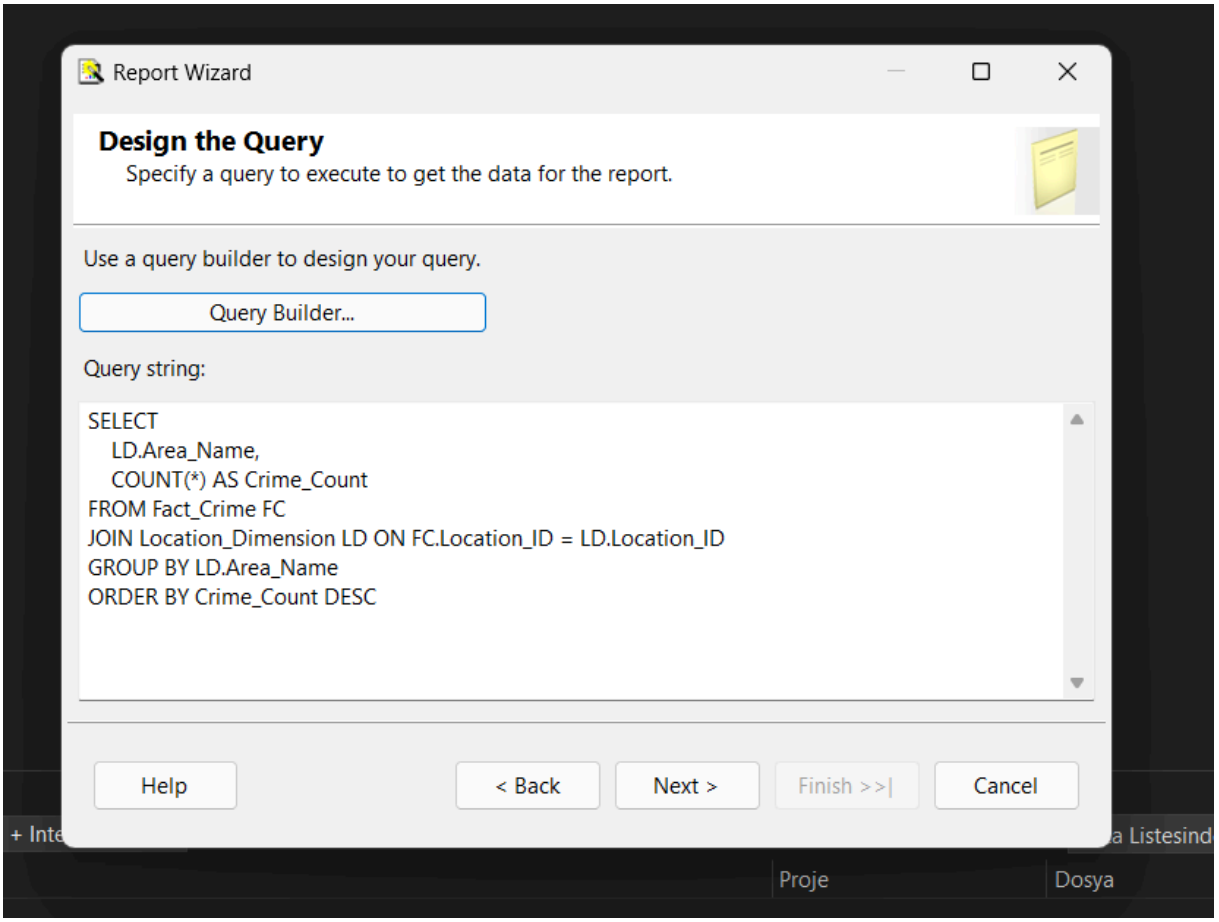


Figure 46: Designing the Query

Designing the Table Layout

In this step, we choose which fields to display in the table. We move **Area_Name** to the **Group** section to group the data by area, and **Crime_Count** to the **Details** section so that the count of crimes per area is shown. This layout is simple but effective for understanding crime distribution.

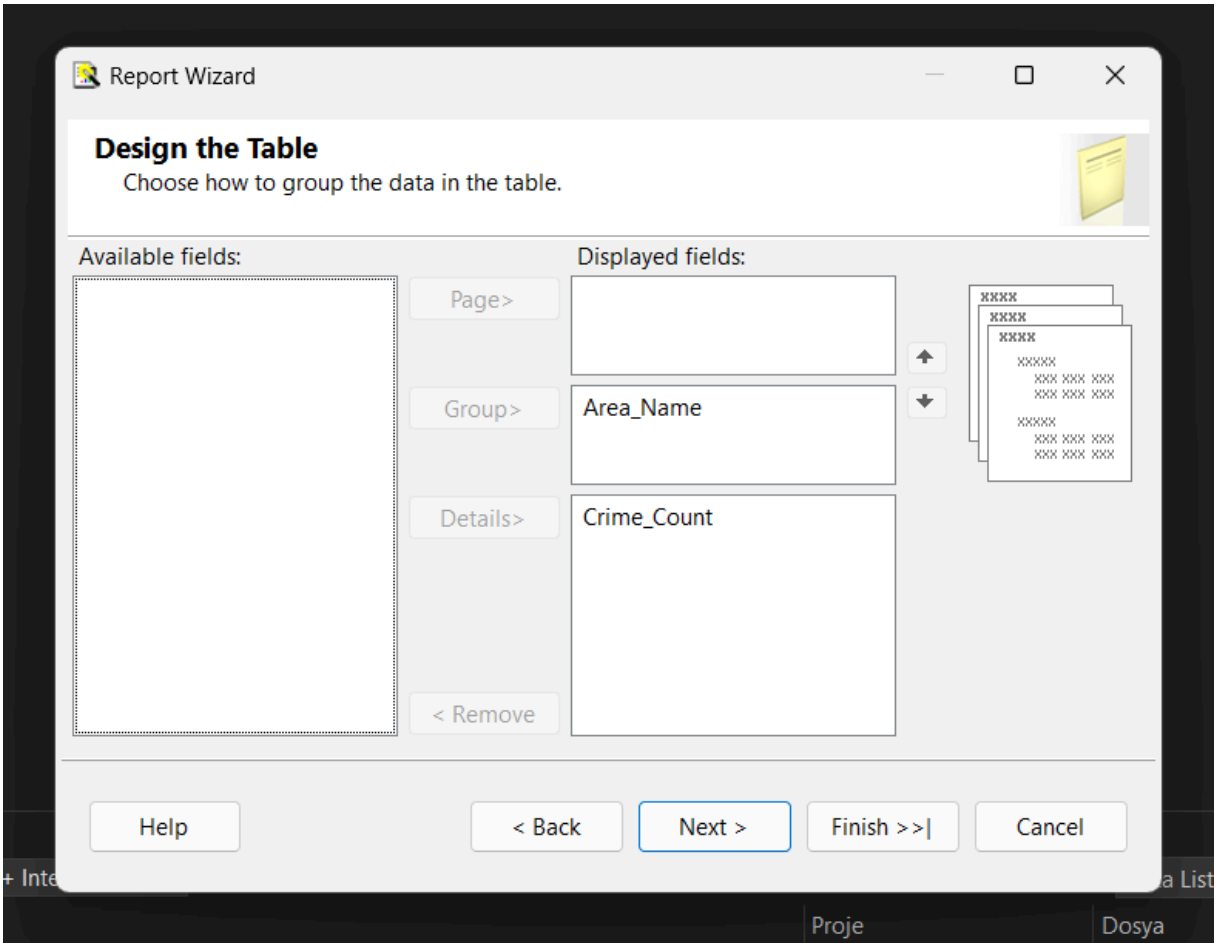


Figure 47:Designing the Table Layout

Sorting Settings

In this step, we are customizing the way the final report will display the data. The **Sorting** section of the Tablix Properties window is used to define the order of the data rows in the table. As shown in the image, the column selected for sorting is **[Crime_Count]**, and the order is set to **Z to A**. This means the report will show the locations starting from the highest number of crimes and ending with the lowest. Sorting the table in descending order by crime count helps the user quickly identify which areas have the most criminal incidents, making the report more useful and insightful for decision-makers.

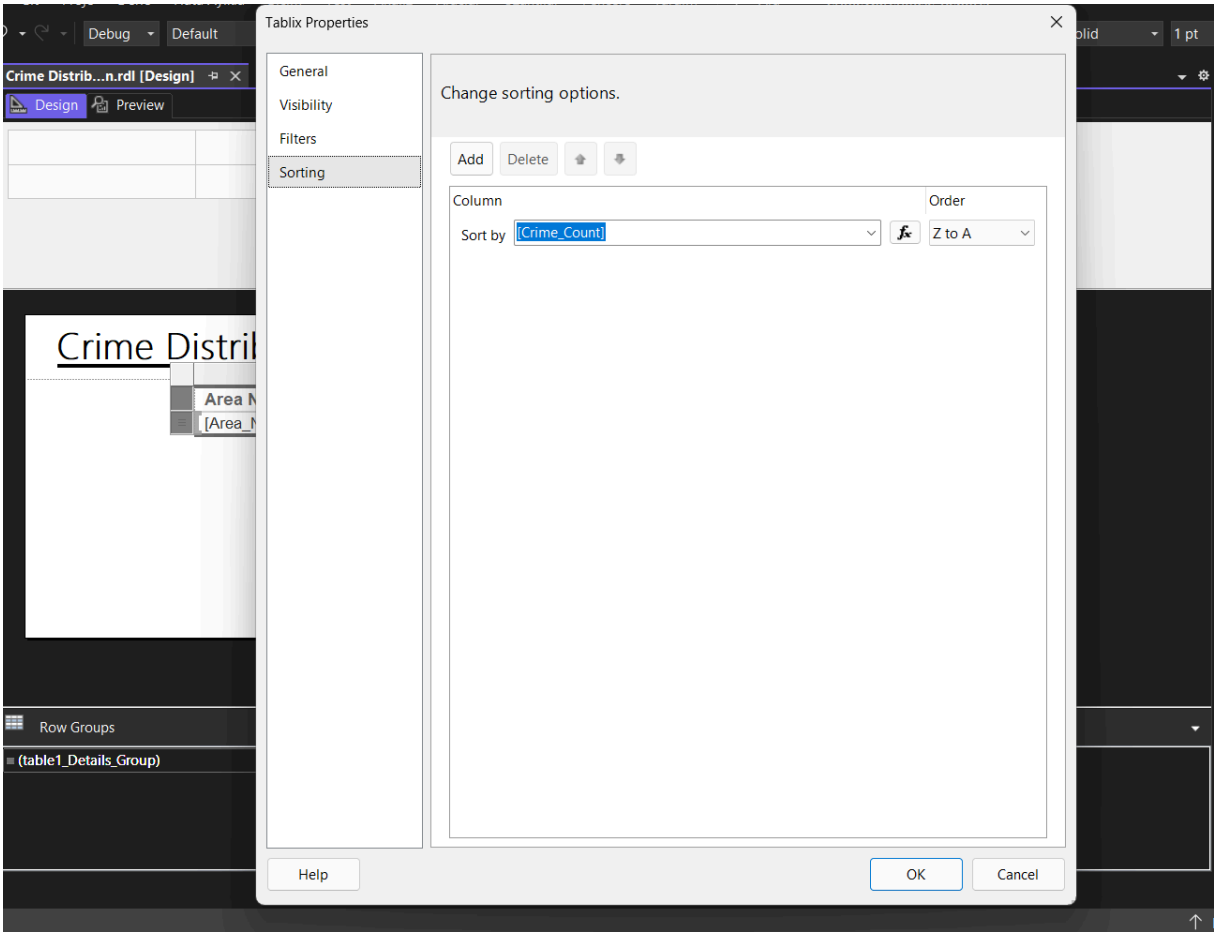


Figure 48: Sorting Settings

Final SSRS Report Preview: "Crime Distribution By Location"

This image displays the final result of the SSRS report. The title of the report is **"Crime Distribution By Location"**, clearly indicating the purpose of the table. The report has two main columns:

- **Area Name**: This shows the name of the district or location where the crimes occurred.
- **Crime Count**: This shows the total number of reported crimes in each area.

Thanks to the previous sorting setup, the locations are arranged from the **highest to the lowest crime count**. For example, "Pacific" has the most reported crimes (13,793), while "Foothill" has the least (6,504). This well-organized and clean table allows users to understand crime distribution across the city at a glance. It helps city authorities, analysts, or law enforcement focus on critical areas where more preventive action might be needed.

Crime Distribution By Location

Area Name	Crime Count
Pacific	13793
Central	12418
77th Street	12386
Wilshire	12354
Southwest	12013
N Hollywood	11306
Devonshire	10421
West LA	10225
Mission	10029
Northeast	9890
Topanga	9505
Hollywood	8957
Harbor	8866
West Valley	8409
Newton	8367
Olympic	7948
Southeast	7909
Van Nuys	7777
Rampart	7755
Hollenbeck	7702
Foothill	6504

Figure 49:Crime Distribution by Location

Question2: Which locations have the most crime?

Data Source Configuration

Unlike the first example, this time we will create our report by adding a new item(report) then adding a data source previously we created as a shared data source. (Demonstrated by Figure 50)

Using a shared data source is a best practice in report development. It allows multiple reports to use the same connection string, which increases consistency and simplifies maintenance.

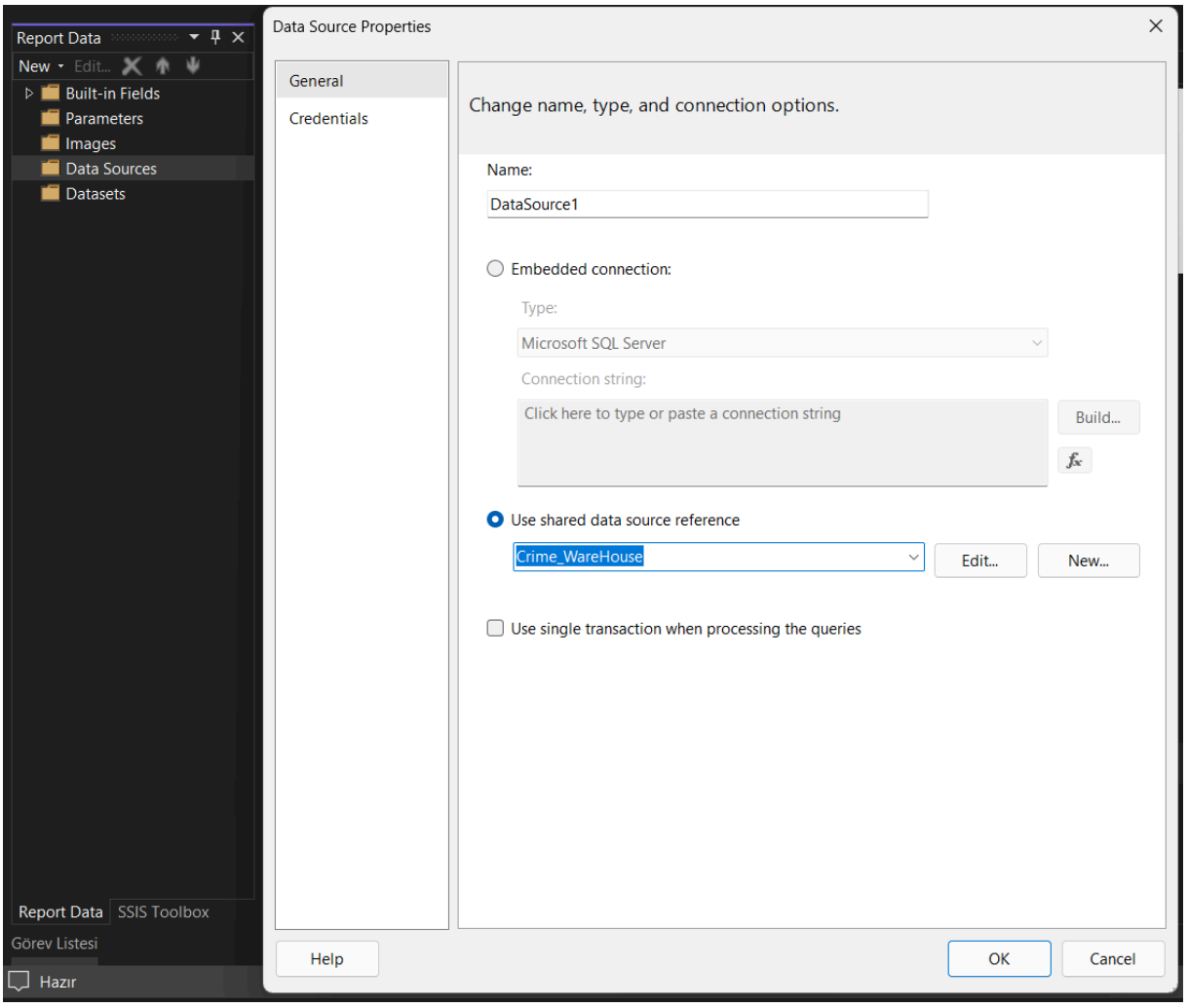


Figure 50: Data source of Report as New Item

Dataset Configuration and Query Settings

This screen shows the **Query Designer** in SSRS, where we begin designing a dataset by selecting the tables to use for our report.

In this case, the user selects the Fact_Crime table from the list of available tables. This step is important because Fact_Crime is the central fact table in our data warehouse. It holds the main transactional data about each crime event, including foreign keys pointing to various dimension tables like:

- Time_Dimension (for date/time)
- Location_Dimension (for area/region)
- CrimeType_Dimension (for type of crime)
- Victim_Dimension (for age and gender of victim)
- Status_Dimension (for case resolution status)

By clicking **"Add"**, the table is added to the visual query builder, enabling the user to define the structure of the SQL query. This allows easy drag-and-drop of fields, filters, and join conditions to build reports without writing raw SQL manually (though editing is still possible).

This step is essential because it defines **what data** will be retrieved for visualization in the SSRS report.

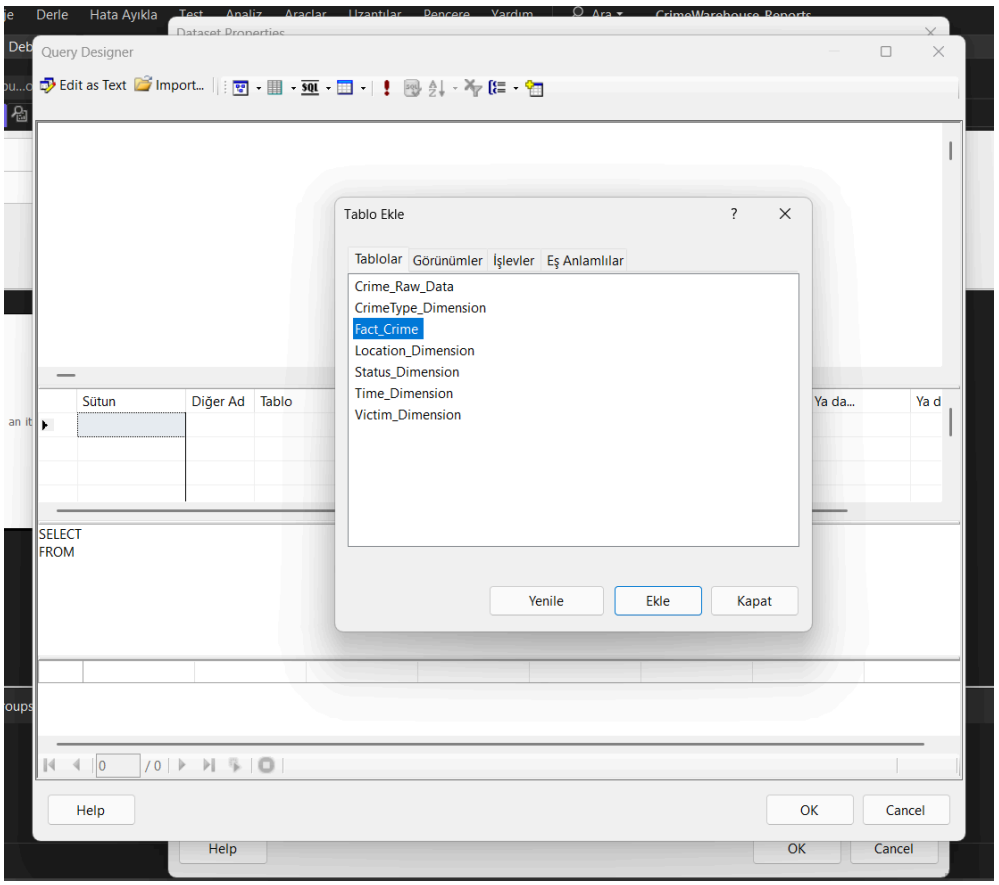


Figure 51:Query Designer for Crime Hours

Query Designer – Crime Count by Hour of Day

In this screenshot, the Query Designer window is used to build a query that calculates the number of crimes for each hour of the day. The SQL query uses DATEPART(HOUR, Time_OCC) to extract the hour component from the Time_OCC column. It then formats the hour as a string like "13:00" using the expression RIGHT('0' + CAST(DATEPART(HOUR, Time_OCC) AS VARCHAR), 2) + ':00' AS HourOfDay. The results are grouped by this formatted hour and counted using COUNT(*) AS Crime_Count. The output is sorted in descending order to show the hours with the most crimes first. This analysis helps identify peak hours for criminal activity, which could be useful for crime prevention strategies.

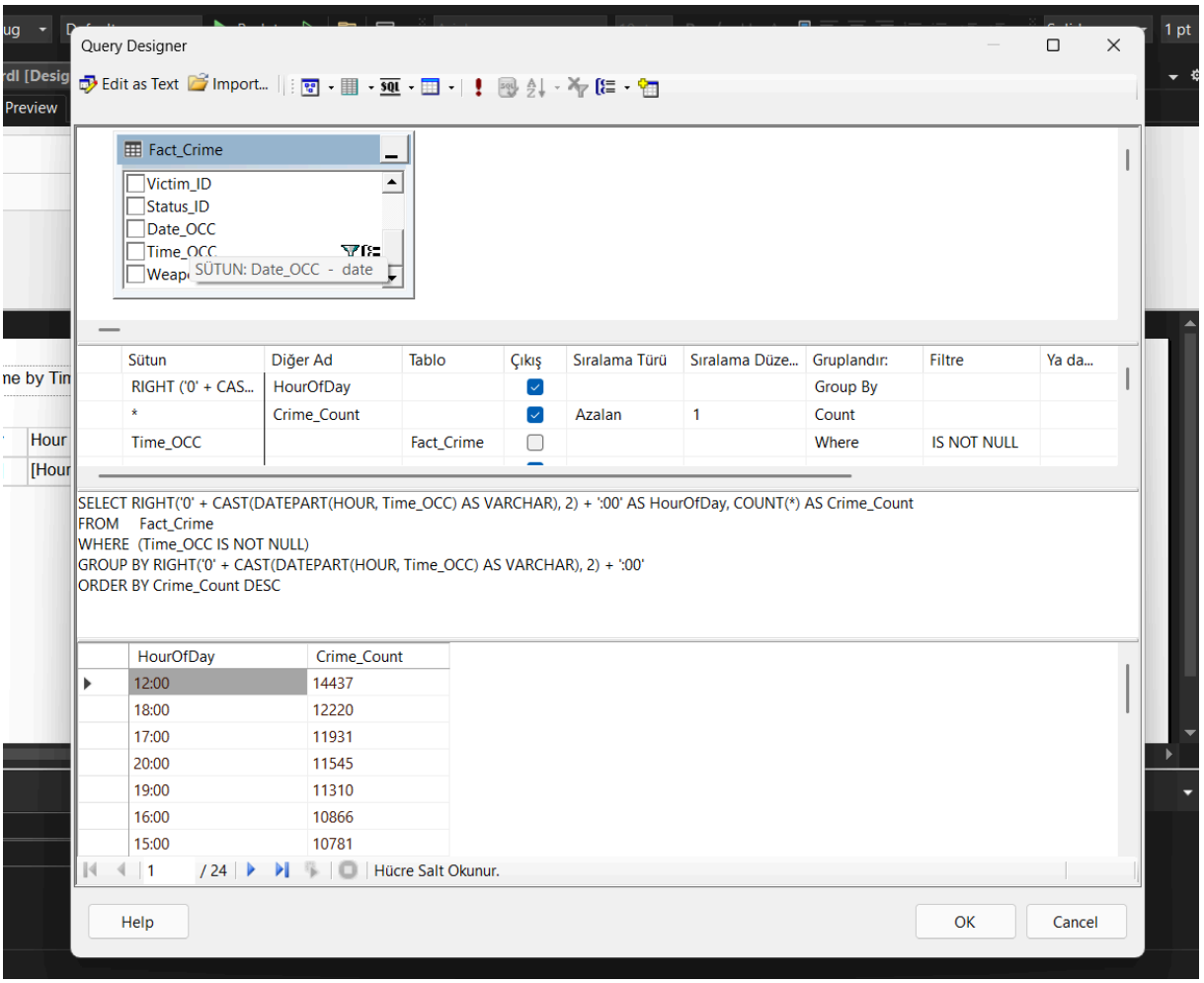


Figure 52:Query Designer – Crime Count by Hour of Day

Tablix Group – Grouping by HourOfDay

This image shows the Tablix Group settings for the report. The data is grouped by HourOfDay, which was created in the previous query. This setting ensures that crimes occurring in the same hour are displayed together in the same row. No headers or footers are added in this step, focusing the table on simple grouping. This step is important because it organizes the report table in a clear and understandable way, reflecting time-based crime patterns.

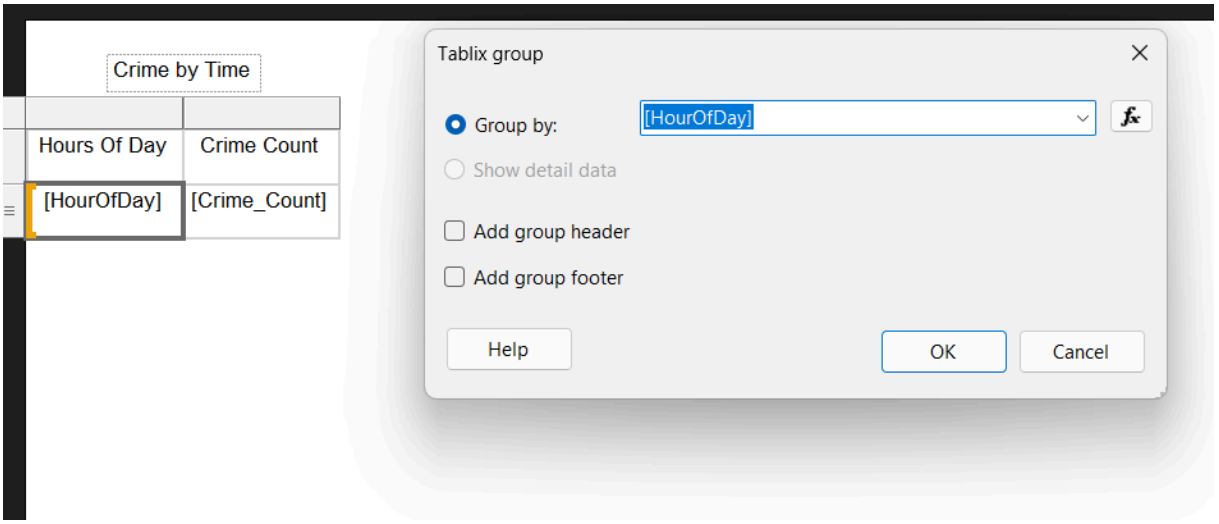


Figure 53:Tablix Group – Grouping by HourOfDay

Query Designer – Crime Count by Month and Year

In this view, another query is built to analyze crime distribution across months and years. The DATENAME(MONTH, Date_OCC) function extracts the name of the month, and DATEPART(YEAR, Date_OCC) extracts the year from the Date_OCC field. These fields are used to group the data and count the number of crimes in each period. The result shows how many crimes occurred in each month and year combination. This query helps users understand seasonal crime patterns, such as whether crimes increase in summer or winter months.

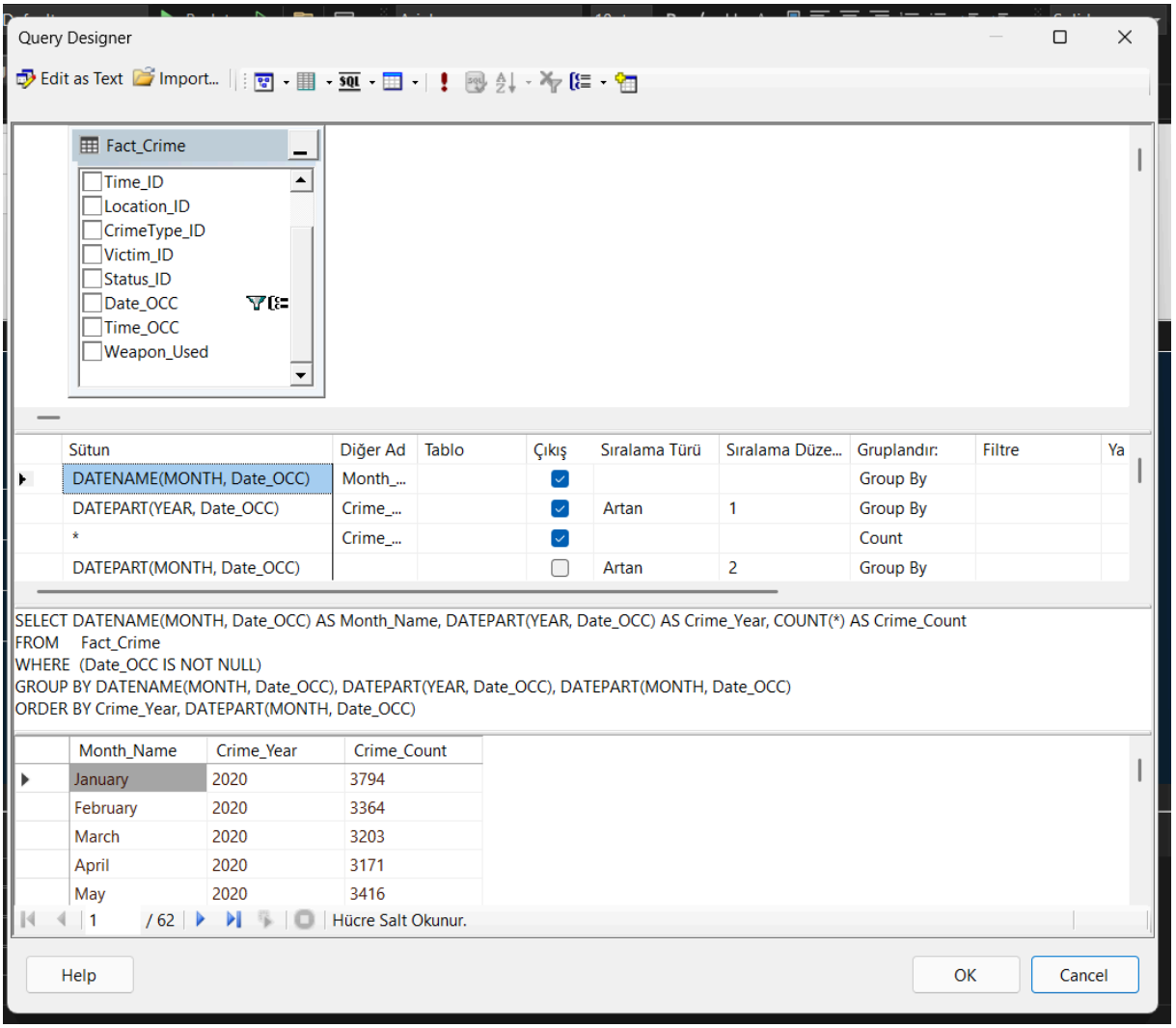


Figure 54:Query Designer – Crime Count by Month and Year

Design View – Crime by Month Tablix Formatting

Here, the report design layout shows a table built to display crime data by month and year. The table groups rows first by Crime_Year and then by Month_Name. These grouping levels are visible in the "Row Groups" section at the bottom. The table is visually styled with red text on a dark blue background for better readability. This layout allows the reader to quickly understand how crime changes over time, making the data more accessible and impactful.

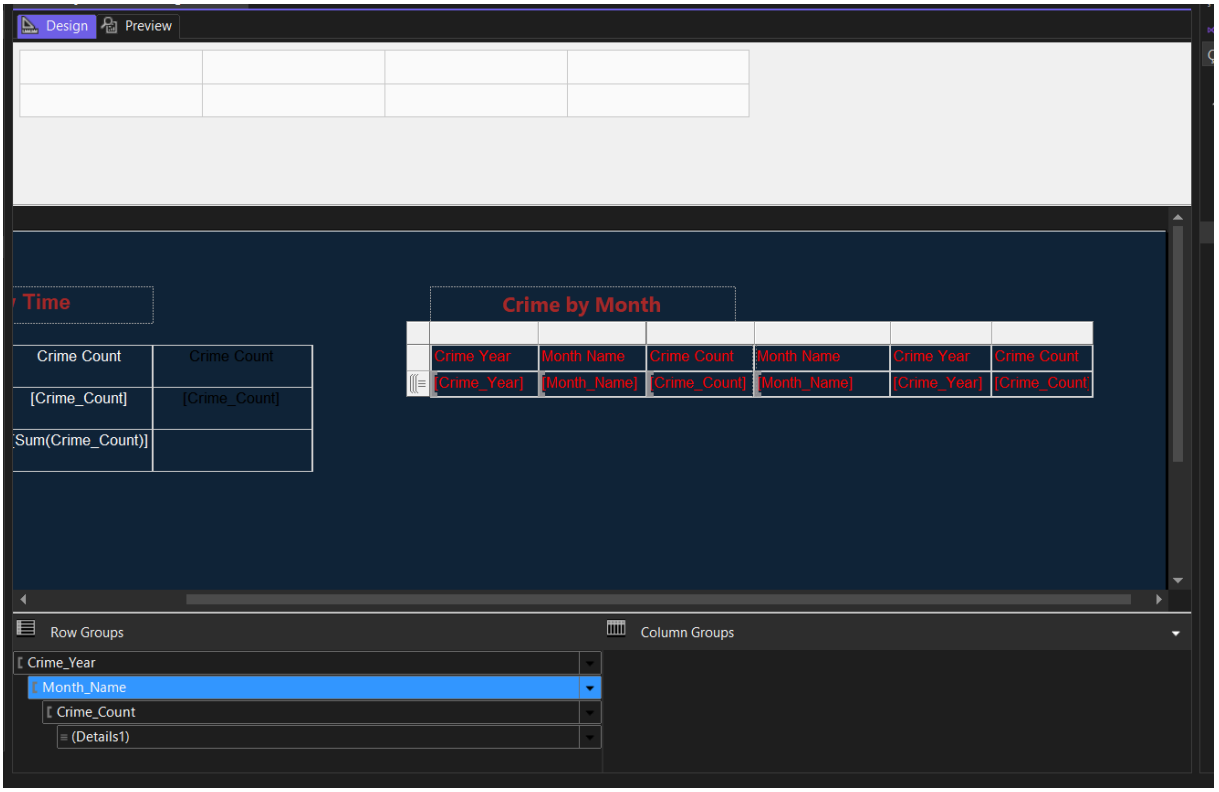


Figure 55:Design View – Crime by Month Tablix Formatting

Report Output – Crime by Time and Month

This screenshot shows the final rendered report preview. On the left side, the "Crime by Time" table displays hourly crime counts, sorted from midnight (00:00) onwards. On the right side, the "Crime by Month" table lists months under each year along with the crime count. These visualizations make it easy to see trends—for example, whether certain hours or months have more incidents. This output is helpful for decision-makers, researchers, or law enforcement to focus on high-risk times.

Crime by Time

Hour Of Day	Crime Count
00:00	8121
01:00	5825
02:00	5041
03:00	4504
04:00	3917

Crime by Month

Crime Year	Month Name	Crime Count
2020	April	3171
	August	3349
	December	3288
	February	3364
	January	3794
	July	3379
	June	3325
	March	3203

05:00	3503
06:00	4553
.	.
.	.

	May	3416
	November	3215
	October	3386
	September	3127
	.	.
	.	.
	.	.

Figure 56:Report Output – Crime by Time and Month

Question3: What is the distribution of crime types by woman victims and area?

Query Designer – Crime Types Against Female Victims by Area

This screenshot displays an advanced SQL query built in the SSRS Query Designer. The goal of the query is to analyze which types of crimes are most frequently committed against female victims in different areas. To achieve this, multiple inner joins are used to bring together data from the **Fact_Crime**, **Location_Dimension**, **Victim_Dimension**, and **CrimeType_Dimension** tables.

The selected columns include:

- Area_Name from the location dimension (L)
- Vict_Sex as Victim_Gender from the victim dimension (V)
- Crime_Desc as Crime_Type from the crime type dimension (CT)
- COUNT(*) AS Crime_Count to get the number of occurrences for each type of crime

The results are **grouped by area name, victim gender, and crime type**, and then sorted in ascending order by area name and descending by crime count. This makes it easy to see which crimes are most common in each area for female victims.

In the output, we can see examples like:

- In **77th Street**, the most frequent crime against women is **THEFT OF IDENTITY** with **734 cases**, followed by **INTIMATE PARTNER - SIMPLE ASSAULT** and **BATTERY - SIMPLE ASSAULT**.

This type of report is extremely valuable for gender-focused crime analysis. It helps identify not only the most dangerous areas for women but also the kinds of threats they face most often. Authorities can use this information to allocate resources, improve protection strategies, and design community-specific safety programs.

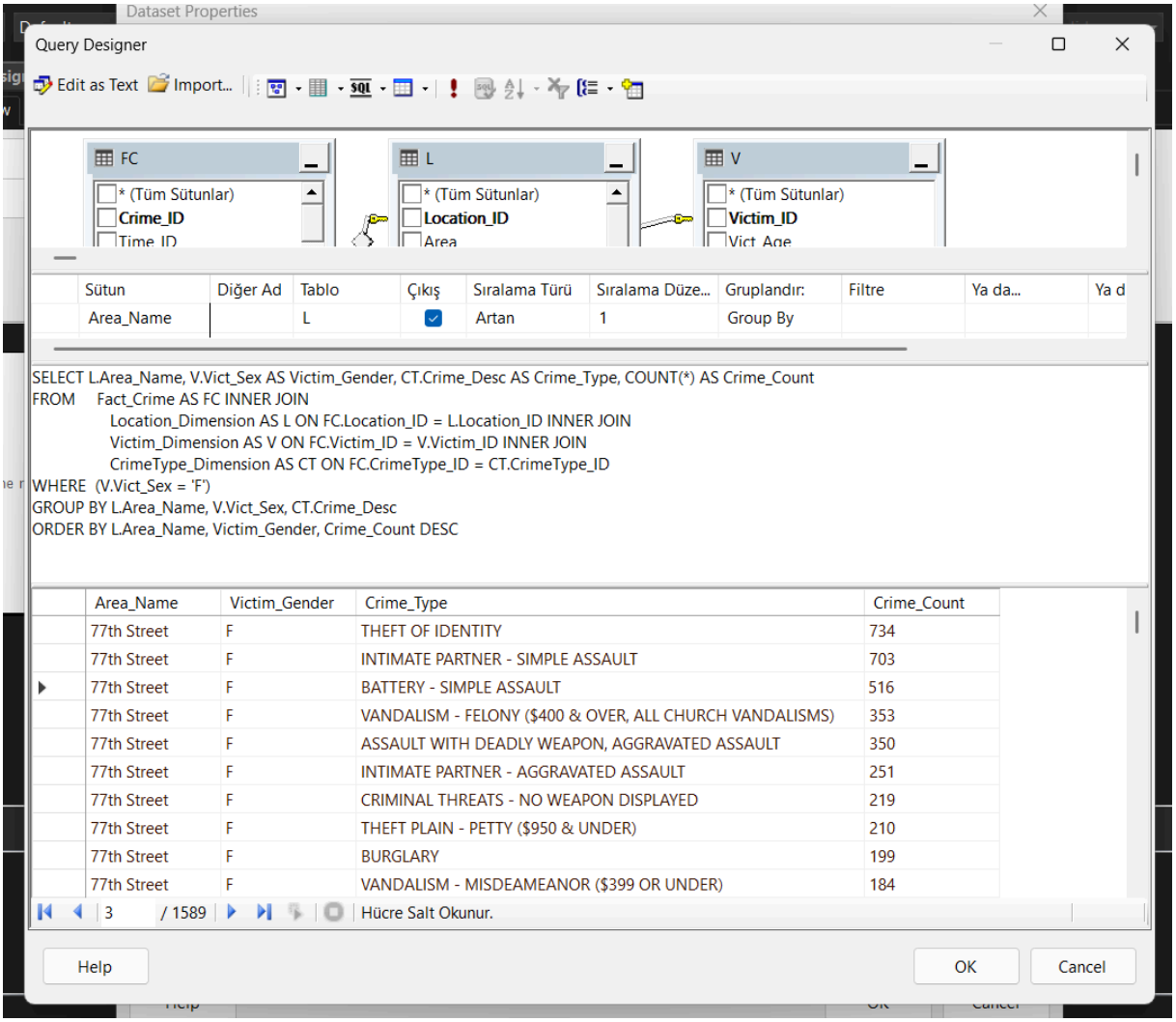


Figure 57:Crime Types Against Female Victims by Area

Report Design View – Distribution of Crime Types by Woman Victims and Area

In this step, we designed a detailed report table that shows the distribution of different crime types committed against women in various areas. The title of the report, written in bold red capital letters, highlights the focus: *"DISTRIBUTION of CRIME TYPES by WOMAN VICTIMS and AREA"*. The table includes four columns: Area Name, Victim Gender, Crime Type, and Crime Count. These fields are dynamically populated with data fetched from the query prepared earlier. At the bottom of the screen, we can see how the row grouping is set up – primarily by Area_Name, followed by Crime_Type, and then Victim_Gender. This structure ensures that the report displays crimes grouped logically, helping viewers easily understand how different types of crimes are distributed across locations and victim gender. The design is user-friendly and color-coded to emphasize important headers.

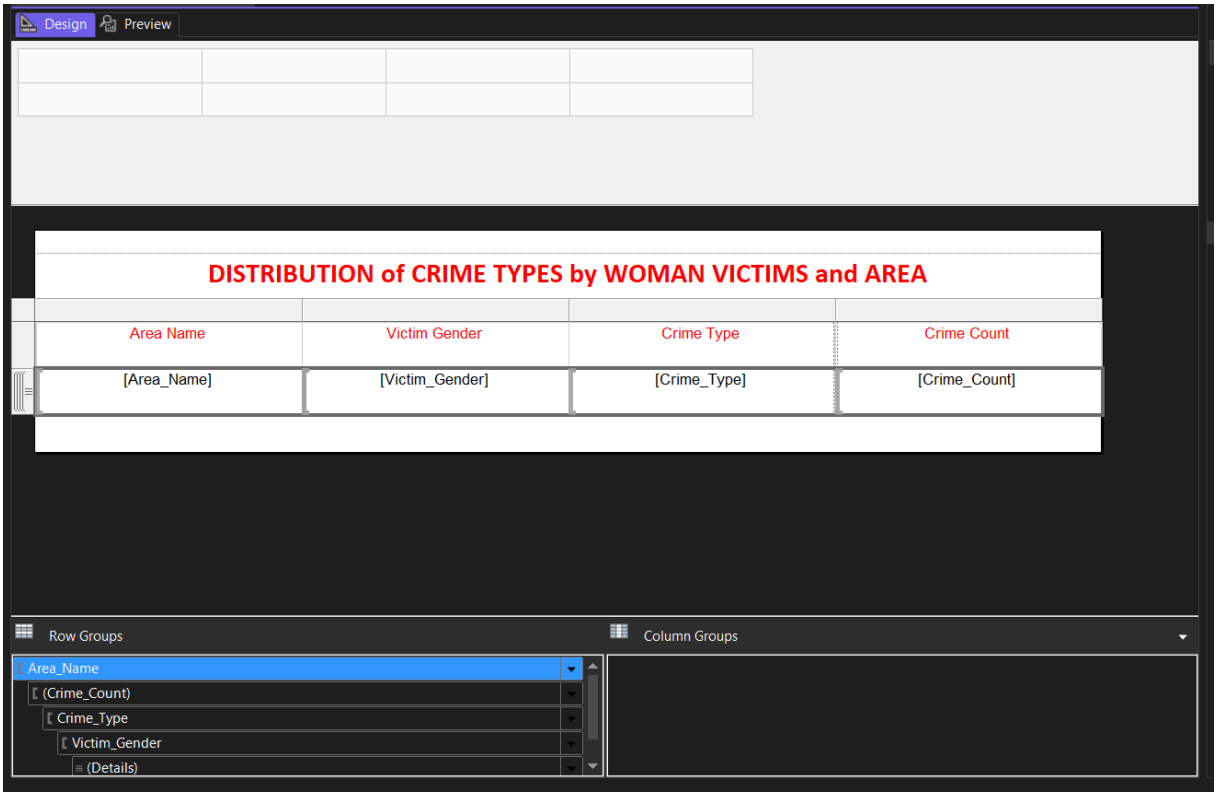
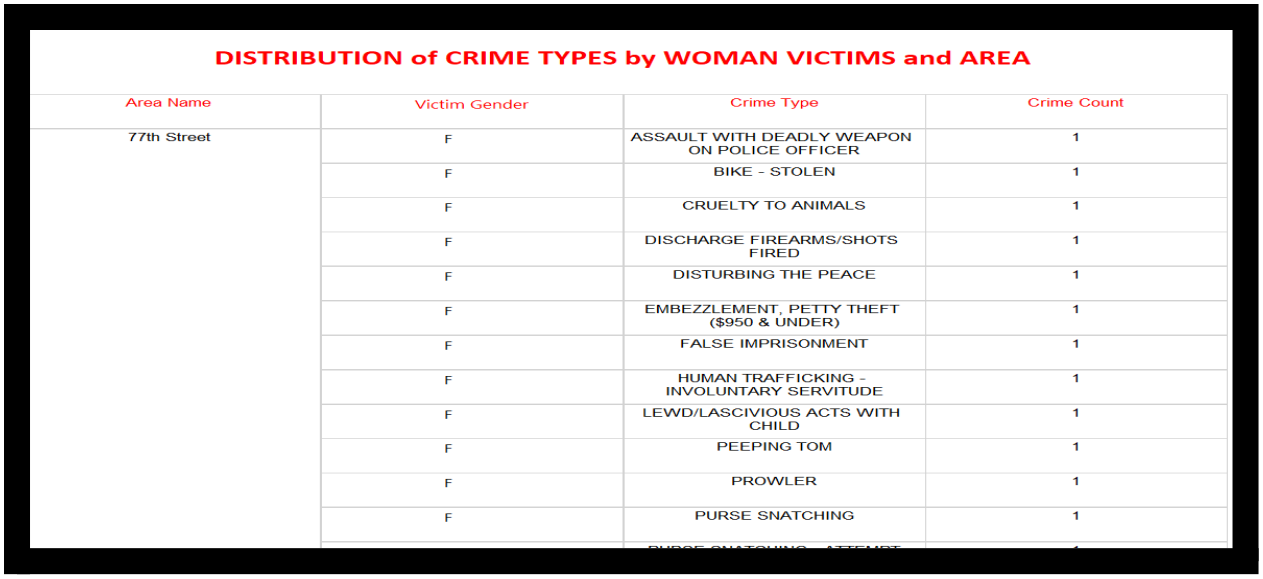


Figure 58:Report Design View – Distribution of Crime Types by Woman Victims and Area

Final Report Preview – Crime Distribution Among Women by Area and Crime Type

This final visual shows the rendered report as the end user would see it. It presents actual crime data for female victims in the "77th Street" area. Each row corresponds to a unique crime type committed against women in that location, with the crime count listed as “1” for most of them. The crime types are diverse, including “ASSAULT WITH DEADLY WEAPON ON POLICE OFFICER,” “BIKE - STOLEN,” “HUMAN TRAFFICKING – INVOLUNTARY SERVITUDE,” and others. This view helps highlight how female victims experience various types of crimes in a specific location. Since it’s sorted and grouped properly, the report is easy to read and can be very helpful for police departments or researchers analyzing crime patterns affecting women in different urban areas.



DISTRIBUTION of CRIME TYPES by WOMAN VICTIMS and AREA

Area Name	Victim Gender	Crime Type	Crime Count
77th Street	F	ASSAULT WITH DEADLY WEAPON ON POLICE OFFICER	1
	F	BIKE - STOLEN	1
	F	CRUELTY TO ANIMALS	1
	F	DISCHARGE FIREARMS/SHOTS FIRED	1
	F	DISTURBING THE PEACE	1
	F	EMBEZZLEMENT, PETTY THEFT ($950 & UNDER)	1
	F	FALSE IMPRISONMENT	1
	F	HUMAN TRAFFICKING - INVOLUNTARY SERVITUDE	1
	F	LEWD/LASCIVIOUS ACTS WITH CHILD	1
	F	PEEPING TOM	1
	F	PROWLER	1
	F	PURSE SNATCHING	1

Figure 59:Final Report Preview – Crime Distribution Among Women by Area and Crime Type

Question4: Crime Type Trends by Year and Victim Age Group

Query Designer-Crime Trends by Victim Age and Year

In this query designer screen, we are preparing a report that shows the number of crimes based on the victim's age group, the type of crime, and the year it occurred. Multiple tables are joined: Fact_Crime, Victim_Dimension, CrimeType_Dimension, and Time_Dimension. The query uses a CASE WHEN statement to categorize victim ages into four groups: 0-17, 18-30, 31-45, and 45+. This grouping makes the data easier to analyze instead of using exact ages. Grouping is applied by Years, Crime_Desc, and Victim_Age_Group, and the total number of crimes is calculated using COUNT(*). This kind of transformation helps decision-makers understand which age groups are more affected by certain crimes each year.

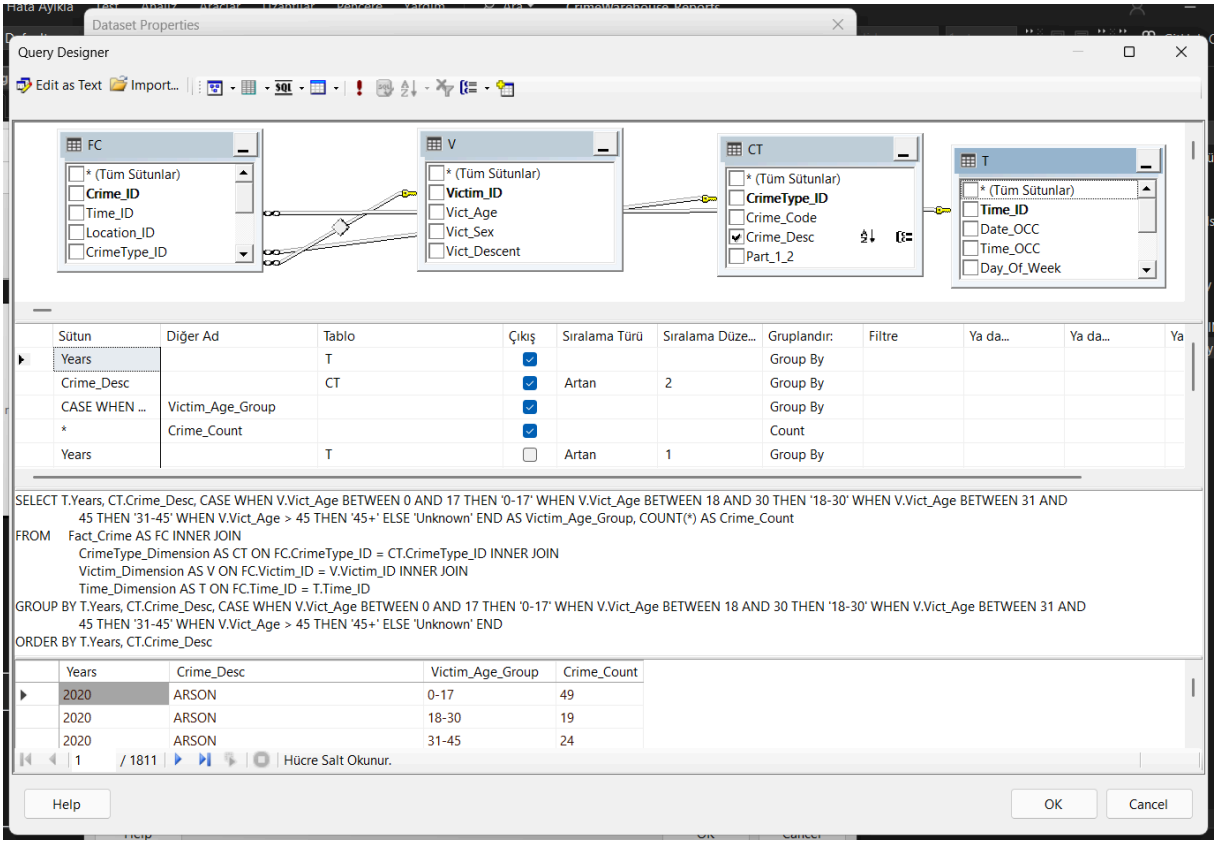


Figure 60:Query Designer: Crime Trends by Victim Age and Year

Report Design- Crime Type Trends by Year and Victim Age Group

This is the design view of the report. The title of the report is clearly written in bold red as “CRIME TYPE TRENDS By YEAR And VICTIM AGE GROUP” to catch the reader’s attention. The table is divided into four columns: Years, Crime Desc, Victim Age Group, and Crime Count. Each row of the report will show a specific crime type committed in a particular year, affecting a certain victim age group, along with the number of incidents. This structured view helps present the query results from the previous image in a clean and readable way.

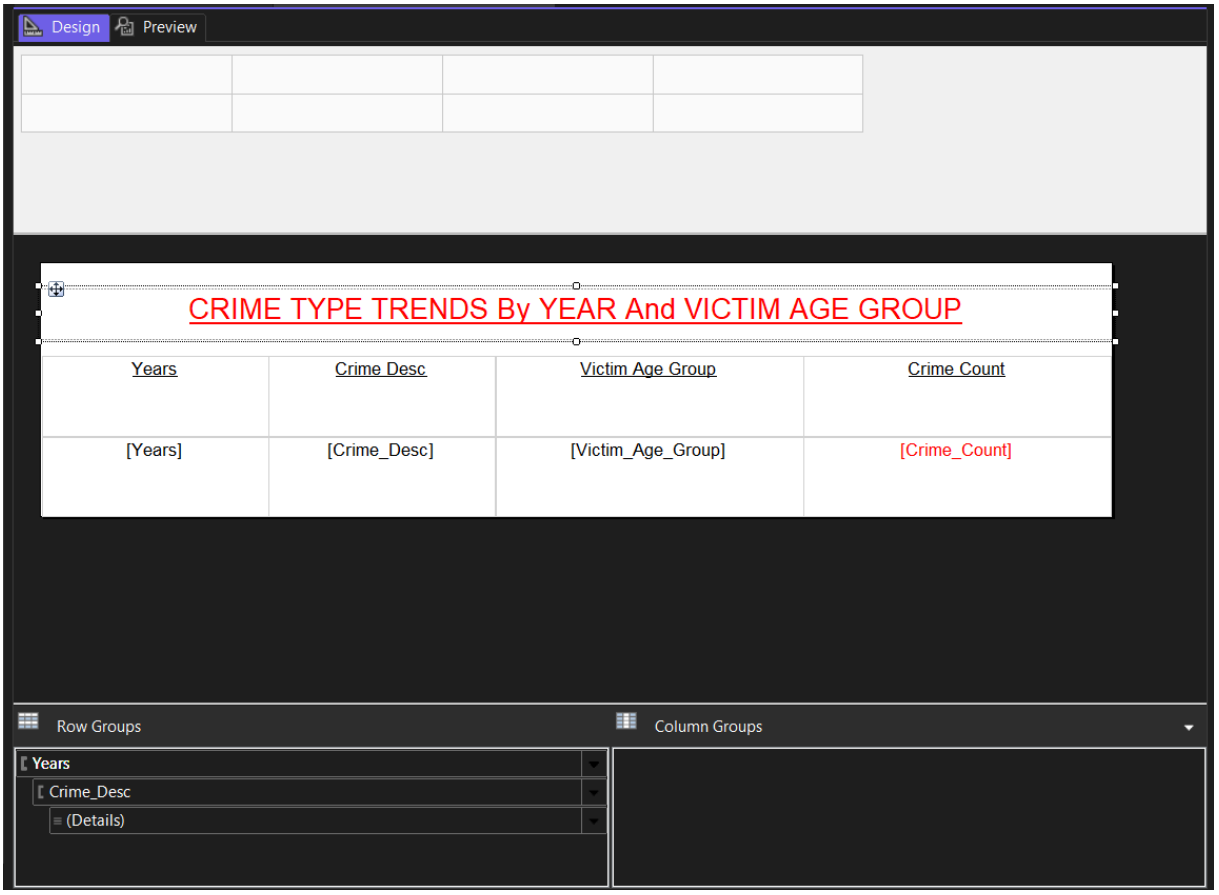
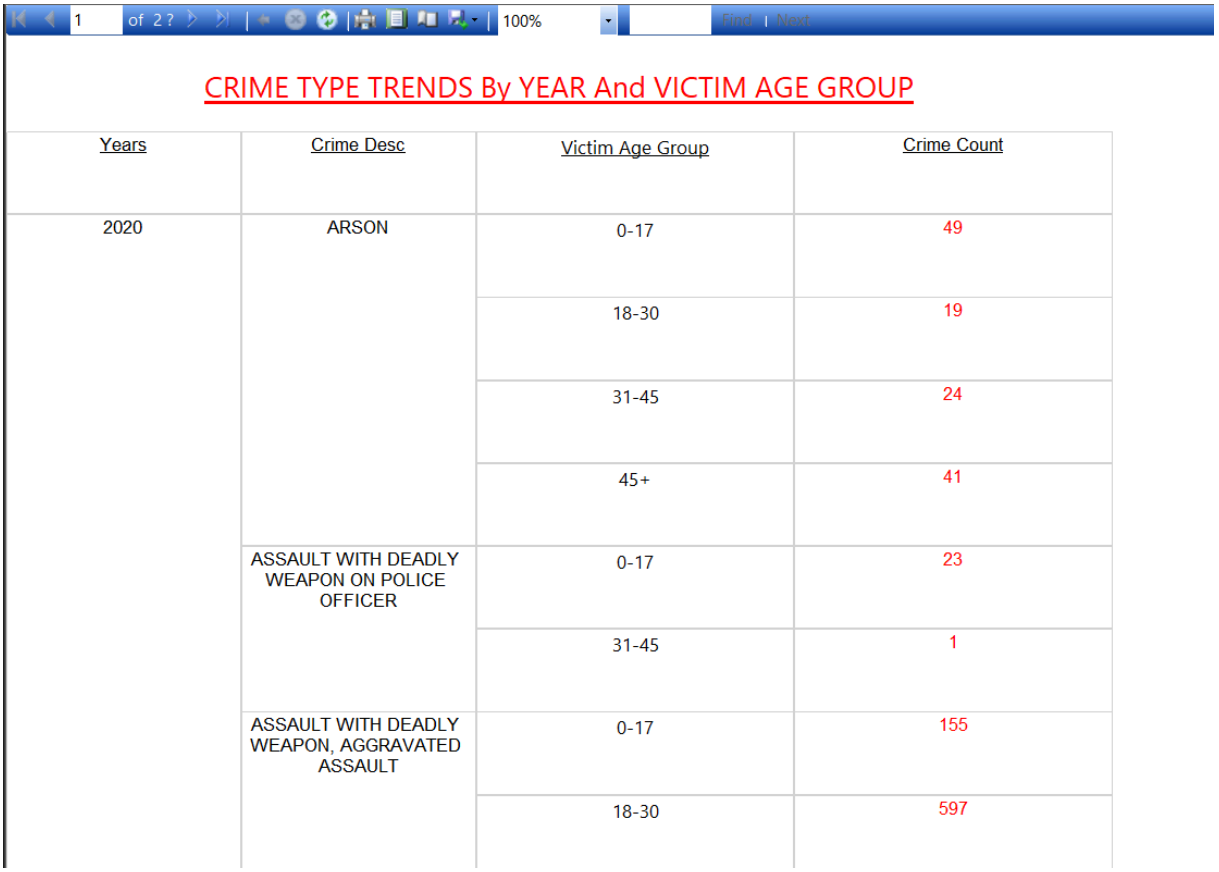


Figure 61:Report Design: Crime Type Trends by Year and Victim Age Group

Report Preview: Final Output

Here, we see the final output of the report after execution. The report correctly displays rows grouped by year and crime type, and then lists the number of victims from each age group affected by that crime. For example, we can see that in 2020, there were 49 arson cases affecting victims aged 0–17. The use of color in the text (red for headers and counts) makes it easier to interpret the results. This format is useful for analyzing crime patterns across different age groups and time periods, helping stakeholders such as police departments or city planners to target specific issues.



CRIME TYPE TRENDS By YEAR And VICTIM AGE GROUP

Years	Crime Desc	Victim Age Group	Crime Count
2020	ARSON	0-17	49
		18-30	19
		31-45	24
		45+	41
	ASSAULT WITH DEADLY WEAPON ON POLICE OFFICER	0-17	23
		31-45	1
	ASSAULT WITH DEADLY WEAPON, AGGRAVATED ASSAULT	0-17	155
		18-30	597

Figure 62:Report Preview: Final Output

SSRS PART FINALIZATION

In this part, we developed a series of SSRS reports to explore different aspects of crime data stored in our data warehouse. By combining fact and dimension tables through properly designed queries, we visualized crime patterns by location, time, victim demographics, and crime types. Through grouping, filtering, and conditional logic, we revealed meaningful trends—such as peak crime hours, high-crime areas, and the distribution of offenses among female victims and age groups. These reports serve as a strong foundation for data-driven crime analysis and can support both operational insights and strategic planning.

Data Visualization with Tableau: Crime Data Analysis

Overview

Using Tableau, we created a comprehensive crime analysis dashboard that addresses four key research questions through interactive visualizations. The dashboard combines multiple data sources derived from our data warehouse to provide a multi-dimensional view of crime patterns across locations, time periods, victim demographics, and crime types.

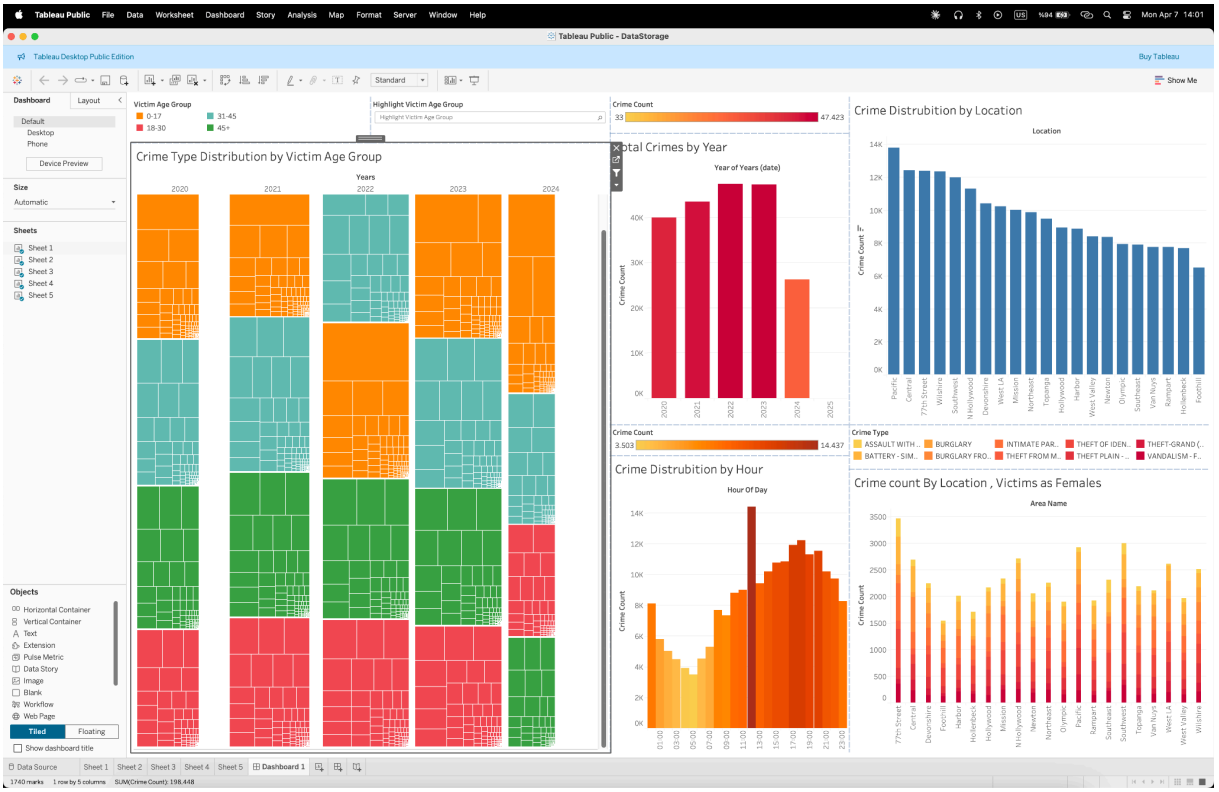


Figure 63:Tableau Dashboard

Dashboard Components

The dashboard consists of four integrated visualizations, each designed to answer a specific analytical question:

1. Crime Type Distribution by Victim Age Group

This treemap visualization divides crime data by victim age groups (0-17, 18-30, 31-45, 45+) using color coding, with each rectangle representing a specific crime type, sized according to frequency. The visualization is further segmented by year (2020-2024), allowing for temporal analysis.

Key insights:

- Vehicle theft remains consistently high across all age groups, particularly for younger victims
- Theft of identity is more prevalent among the 31-45 age group (teal colored sections)
- Battery and assault crimes show distinct patterns across different age demographics
- The 45+ age group (green) shows different victimization patterns than younger groups

The treemap format effectively handles the large number of crime categories while making visual patterns immediately apparent. Year-by-year comparison allows for tracking how crime types affecting each age group have evolved over time.

2. Crime Distribution by Hour and Year

This histogram displays crime frequency across 24 hours of the day, and yearly revealing clear temporal patterns in criminal activity. The visualization uses a gradient color scheme to emphasize peak hours.

Key insights:

- Crime activity peaks dramatically around 12:00 (noon) with nearly 14,500 incidents
- A second significant peak occurs in the evening hours between 17:00-19:00
- Early morning hours (01:00-05:00) show the lowest crime activity
- The distribution creates a bimodal pattern with mid-day and evening peaks
- 2022 -2023 shows the highest crime count 47,423 and 47277 respectively.

This time-based analysis provides crucial information for law enforcement resource allocation and preventive measures. The clear identification of peak crime hours can inform patrol scheduling and community safety initiatives.

3. Crime Distribution by Location

This horizontal bar chart ranks all areas by total crime count, providing an immediate visualization of crime hotspots. The consistent color scheme and descending order make it easy to identify high-crime areas.

Key insights:

- Pacific leads with approximately 14,000 crimes, followed closely by Central and 77th Street
- There's a gradual decrease in crime rates across areas rather than sharp divides
- The difference between highest (Pacific) and lowest (Foothill) crime areas is substantial
- Mid-range areas show similar crime levels with subtle differences

This geographic analysis helps prioritize areas for intervention and resource allocation. The visualization makes it immediately clear which neighborhoods require the most attention from law enforcement.

4. Crime Count by Location, Victims as Females

This specialized bar chart focuses on crimes against female victims across different locations. Using a warm color gradient, it highlights areas where women experience higher crime rates.

Key insights:

- 77th Street shows the highest number of crimes against female victims
- The distribution pattern differs from the overall crime distribution
- Several areas show similar rates of crimes against women despite different overall crime rates
- Some locations with moderate overall crime rates show proportionally higher female victimization

This gender-focused analysis provides valuable insights for targeted safety programs and community outreach efforts aimed at reducing crimes against women.

Interactive Elements and Integration

The dashboard incorporates several interactive elements that enhance its analytical value:

1. **Victim Age Group Filter**: Users can filter all visualizations by specific age groups
2. **Crime Type Highlighting**: The legend allows highlighting specific crime types across all relevant views
3. **Crime Count Parameter**: Provides flexible thresholds for analyzing high vs. low frequency crimes
4. **Consistent Color Coding**: Age groups maintain consistent colors across visualizations

These interactive capabilities allow users to explore specific subsets of data while maintaining context across all visualizations. The integration of multiple views creates a cohesive analytical environment where insights from one visualization complement findings from others.

Technical Implementation

The Tableau dashboard was created by:

1. Connecting to multiple data sources exported from our SQL Server data warehouse
2. Creating individual worksheets for each research question
3. Applying appropriate visualization types based on the nature of each analysis
4. Implementing consistent formatting and color schemes
5. Combining worksheets into an integrated dashboard with shared filters
6. Adding interactive elements to enhance user exploration

The visualizations leverage Tableau's native capabilities for handling hierarchical data (treemap), time-based analysis (histogram), ranking (bar chart), and comparative analysis (filtered bar chart).

Summary

The Tableau dashboard successfully transforms our crime data warehouse into actionable visual insights that answer key research questions about crime patterns. The visualizations reveal important trends in:

- Crime distribution across victim demographics
- Temporal patterns of criminal activity
- Geographic concentration of crime
- Gender-specific victimization patterns

These insights provide a solid foundation for data-driven decision making in law enforcement resource allocation, community safety initiatives, and crime prevention strategies. The interactive nature of the dashboard allows for ongoing exploration as new data becomes available or as analytical priorities shift.

The visualization approach complements our database implementations in PostgreSQL and Neo4j, demonstrating how the same data can be leveraged through different technological approaches to generate complementary insights.

Comparative Analysis: PostgreSQL vs Neo4j for Crime Data Analysis

This report presents a comparative analysis of relational database (PostgreSQL) and graph database (Neo4j) approaches for crime data analysis, using the same dataset from our crime data warehouse project.

Database Implementation Overview

We implemented the same crime data analysis in both PostgreSQL and Neo4j to evaluate their respective strengths and weaknesses. This allowed for direct comparison of query complexity, performance, and analytical capabilities.

PostgreSQL Implementation

In PostgreSQL, we structured our data using traditional relational tables, adopting a flat data model that mirrors the dimensional structure but without explicitly enforcing relationships through foreign keys:

```
CREATE TABLE crimes_by_location (
    area_name VARCHAR(255) PRIMARY KEY,
    crime_count INTEGER
);

CREATE TABLE hourly_crimes (
    hour_of_day VARCHAR(5) PRIMARY KEY,
    crime_count INTEGER
);

CREATE TABLE monthly_crimes (
    crime_year INTEGER,
    month_name VARCHAR(20),
    crime_count INTEGER,
    PRIMARY KEY (crime_year, month_name)
);

CREATE TABLE crime_type_trends (
    years INTEGER,
    crime_desc VARCHAR(255),
```

```
    victim_age_group VARCHAR(5),
    crime_count INTEGER,
    PRIMARY KEY (years, crime_desc, victim_age_group)
);

CREATE TABLE crimes_by_victim_area (
    area_name VARCHAR(255),
    victim_gender VARCHAR(1),
    crime_type VARCHAR(255),
    crime_count INTEGER,
    PRIMARY KEY (area_name, victim_gender, crime_type)
);
```

Data was imported using the PostgreSQL COPY command:

```
\copy crimes_by_location (area_name, crime_count) FROM '/Users/emredinc/Library/Application Support/Neo4j Desktop/Application/relate-data/dbmss/dbms-aec5bc20-89a4-4e4d-a2a9-d032da50e676/import/Cleaned_Crimes_By_Location.csv' WITH (FORMAT CSV, HEADER true, DELIMITER ',');

\copy hourly_crimes (hour_of_day, crime_count) FROM '/Users/emredinc/Library/Application Support/Neo4j Desktop/Application/relate-data/dbmss/dbms-aec5bc20-89a4-4e4d-a2a9-d032da50e676/import/Cleaned_Hourly_Crimes.csv' WITH (FORMAT CSV, HEADER true, DELIMITER ',');

\copy monthly_crimes (crime_year, month_name, crime_count) FROM '/Users/emredinc/Library/Application Support/Neo4j Desktop/Application/relate-data/dbmss/dbms-aec5bc20-89a4-4e4d-a2a9-d032da50e676/import/Cleaned_Monthly_Crimes.csv' WITH (FORMAT CSV, HEADER true, DELIMITER ',');

\copy crime_type_trends (years, crime_desc, victim_age_group, crime_count) FROM '/Users/emredinc/Library/Application Support/Neo4j Desktop/Application/relate-data/dbmss/dbms-aec5bc20-89a4-4e4d-a2a9-d032da50e676/import/Cleaned_Crime_Type_Trends.csv' WITH (FORMAT CSV, HEADER true, DELIMITER ',');
```

\copy crimes_by_victim_area (area_name, victim_gender, crime_type, crime_count) FROM '/Users/emredinc/Library/Application Support/Neo4j Desktop/Application/relate-data/dbmss/dbms-aec5bc20-89a4-4e4d-a2a9-d032da50e676/import/Cleaned_Crimes_By_Victim_Area.csv' WITH (FORMAT CSV, HEADER true, DELIMITER ',');

Neo4j Implementation

In Neo4j, we created a rich graph structure with various node types and relationship patterns:

```
CREATE CONSTRAINT FOR (a:Area) REQUIRE a.name IS UNIQUE;

CREATE CONSTRAINT FOR (h:Hour) REQUIRE h.value IS UNIQUE;

CREATE CONSTRAINT FOR (y:Year) REQUIRE y.value IS UNIQUE;

CREATE CONSTRAINT FOR (m:Month) REQUIRE m.name IS UNIQUE;

CREATE CONSTRAINT FOR (cd:CrimeDescription) REQUIRE cd.description IS UNIQUE;

CREATE CONSTRAINT FOR (ag:VictimAgeGroup) REQUIRE ag.group IS UNIQUE;

CREATE CONSTRAINT FOR (vg:VictimGender) REQUIRE vg.gender IS UNIQUE;

CREATE CONSTRAINT FOR (ym:YearMonth) REQUIRE (ym.year, ym.month) IS NODE KEY;

CREATE CONSTRAINT FOR (s:CrimeTrendSummary) REQUIRE (s.year, s.crimeDesc, s.ageGroup) IS NODE KEY;

CREATE CONSTRAINT FOR (s:VictimAreaSummary) REQUIRE (s.area, s.gender, s.crimeType) IS NODE KEY;
```

Data was imported using Neo4j's LOAD CSV command and a rich, interconnected model was created with entity nodes and relationship connections:

```
LOAD CSV WITH HEADERS FROM 'file:///Cleaned_Crimes_By_Location.csv' AS row

MERGE (a:Area {name: row.`Area Name`})

SET a.totalCrimeCount = toInteger(row.`Crime Count`);


LOAD CSV WITH HEADERS FROM 'file:///Cleaned_Hourly_Crimes.csv' AS row

MERGE (h:Hour {value: row.`Hour Of Day`})

SET h.totalCrimeCount = toInteger(row.`Crime Count`);


LOAD CSV WITH HEADERS FROM 'file:///Cleaned_Monthly_Crimes.csv' AS row

MERGE (y:Year {value: toInteger(row.`Crime Year`)})
```

MERGE (m:Month {name: row.`Month Name`})

MERGE (ym:YearMonth {year: toInteger(row.`Crime Year`), month: row.`Month Name`})

SET ym.crimeCount = toInteger(row.`Crime Count`)

MERGE (y)-[:HAS_MONTH]->(ym)

MERGE (ym)-[:IS_MONTH]->(m);

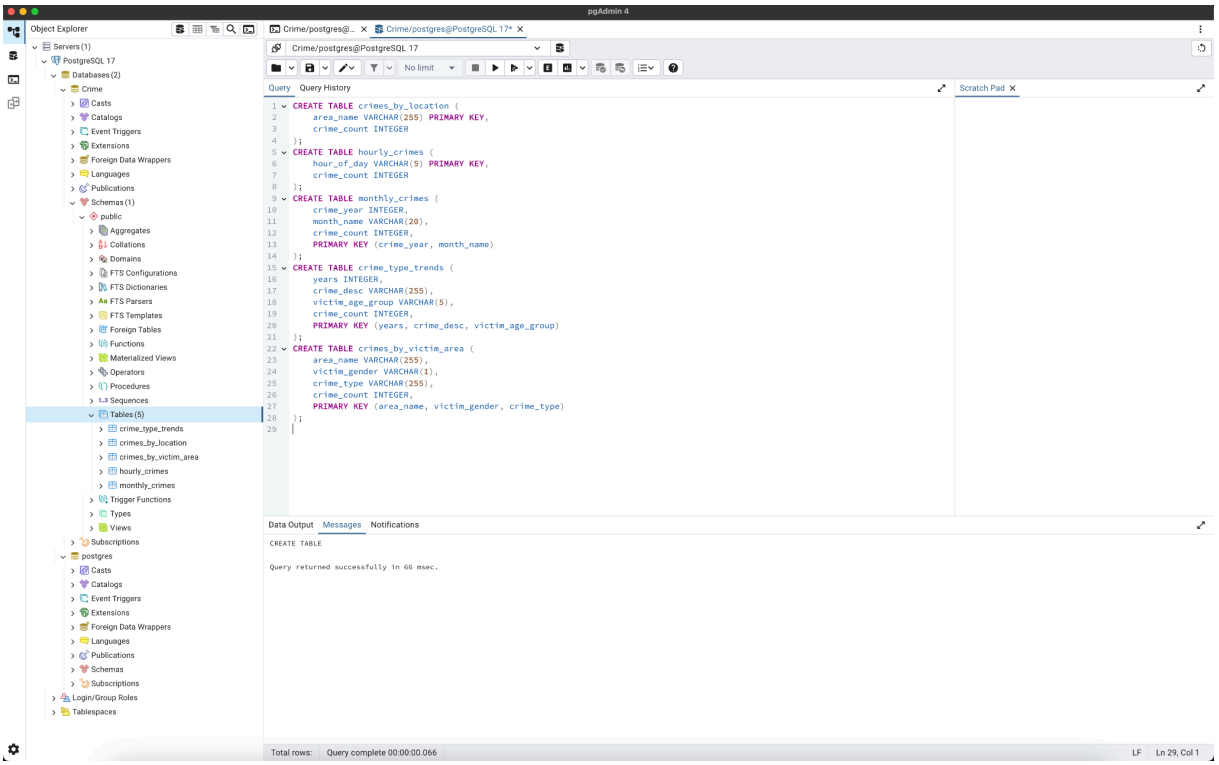


Figure 64:PostgreSQL Table Creation

```
psql (17.4)
Type "hea
Crime-# \copy crimes_by_location (area_name, crime_count) FROM '/Users/emredinc/Library/Application Support/Neo4j Desktop/Application/relate-data/dbmss/dbms-aec5bc20-89a4-4e4d-a
2a9-d032da50e676/import/Cleaned_Crimes_By_Location.csv' WITH (FORMAT CSV, HEADER true, DELIMITER ',');
COPY 21
Crime-# \copy hourly_crimes (hour_of_day, crime_count) FROM '/Users/emredinc/Library/Application Support/Neo4j Desktop/Application/relate-data/dbmss/dbms-aec5bc20-89a4-4e4d-a2a9
-d032da50e676/import/Cleaned_Hourly_Crimes.csv' WITH (FORMAT CSV, HEADER true, DELIMITER ',');
COPY 24
Crime-# \copy monthly_crimes (crime_year, month_name, crime_count) FROM '/Users/emredinc/Library/Application Support/Neo4j Desktop/Application/relate-data/dbmss/dbms-aec5bc20-89
a4-4e4d-a2a9-d032da50e676/import/Cleaned_Monthly_Crimes.csv' WITH (FORMAT CSV, HEADER true, DELIMITER ',');
COPY 62
Crime-# \copy crime_type_trends (years, crime_desc, victim_age_group, crime_count) FROM  '/Users/emredinc/Library/Application Support/Neo4j Desktop/Application/relate-data/dbmss
/dbms-aec5bc20-89a4-4e4d-a2a9-d032da50e676/import/Cleaned_Crime_Type_Trends.csv'  WITH (FORMAT CSV, HEADER true, DELIMITER ',');
COPY 1790
Crime-# \copy crimes_by_victim_area (area_name, victim_gender, crime_type, crime_count) FROM '/Users/emredinc/Library/Application Support/Neo4j Desktop/Application/relate-data/d
bmss/dbms-aec5bc20-89a4-4e4d-a2a9-d032da50e676/import/Cleaned_Crimes_By_Victim_Area.csv'  WITH (FORMAT CSV, HEADER true, DELIMITER ',');
COPY 1589
Crime-#
```

Figure 65:PostgreSQL table population

```
1 CREATE CONSTRAINT IF NOT EXISTS FOR (a:Area) REQUIRE a.name IS
  UNIQUE;
2 CREATE CONSTRAINT IF NOT EXISTS FOR (h:Hour) REQUIRE h.value IS
  UNIQUE;
3 CREATE CONSTRAINT IF NOT EXISTS FOR (y:Year) REQUIRE y.value IS
  UNIQUE;
4 CREATE CONSTRAINT IF NOT EXISTS FOR (m:Month) REQUIRE m.name IS
  UNIQUE;
```

Figure 66:Neo4j Node Creation

```
CREATE CONSTRAINT IF NOT EXISTS FOR (cd:CrimeDescription) REQUIRE
cd.description IS UNIQUE;
CREATE CONSTRAINT IF NOT EXISTS FOR (ag:VictimAgeGroup) REQUIRE
ag.group IS UNIQUE;
CREATE CONSTRAINT IF NOT EXISTS FOR (vg:VictimGender) REQUIRE
vg.gender IS UNIQUE;
CREATE CONSTRAINT IF NOT EXISTS FOR (ym:YearMonth) REQUIRE
(ym.year, ym.month) IS NODE KEY;
CREATE CONSTRAINT IF NOT EXISTS FOR (s:CrimeTrendSummary) REQUIRE
(s.year, s.crimeDesc, s.ageGroup) IS NODE KEY;
CREATE CONSTRAINT IF NOT EXISTS FOR (s:VictimAreaSummary) REQUIRE
(s.area, s.gender, s.crimeType) IS NODE KEY;
```

Figure 67:Neo4j Node Creation continued

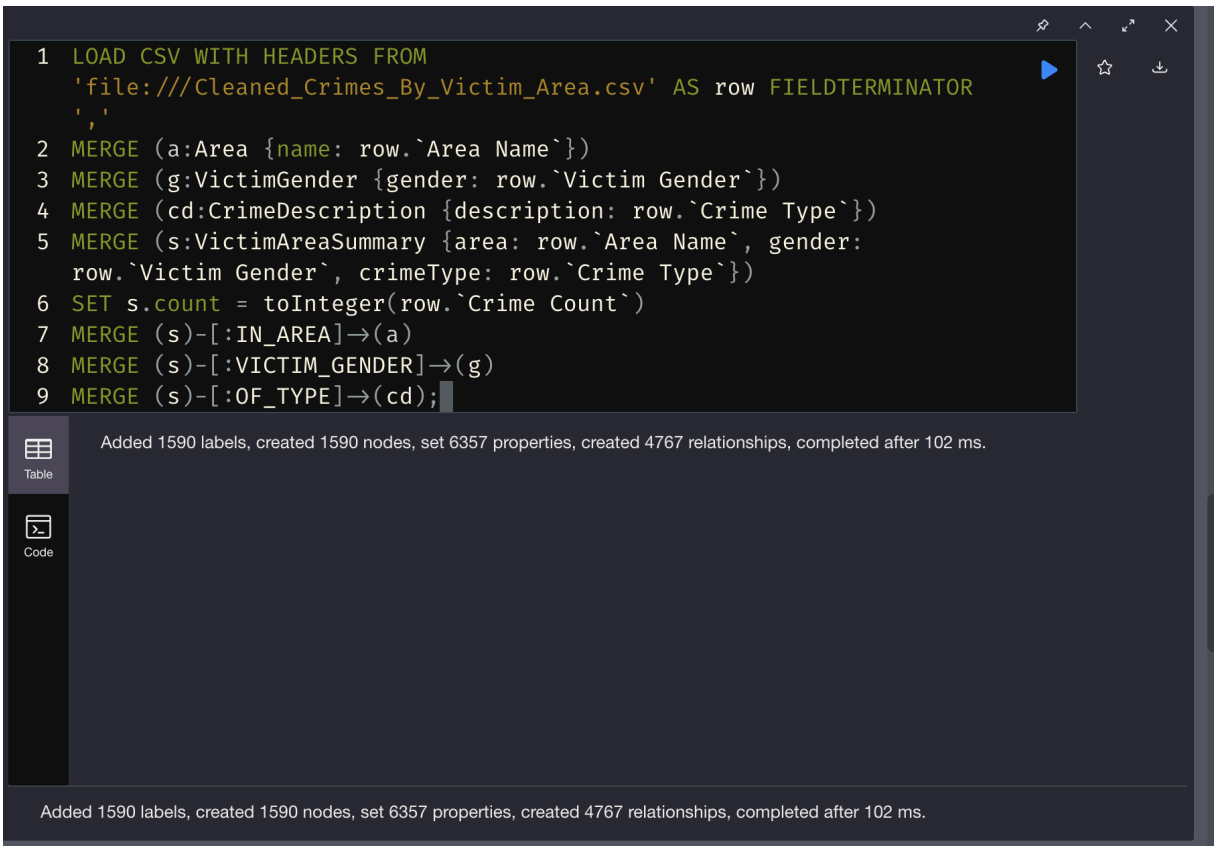


Figure 68:Neo4j Populating nodes

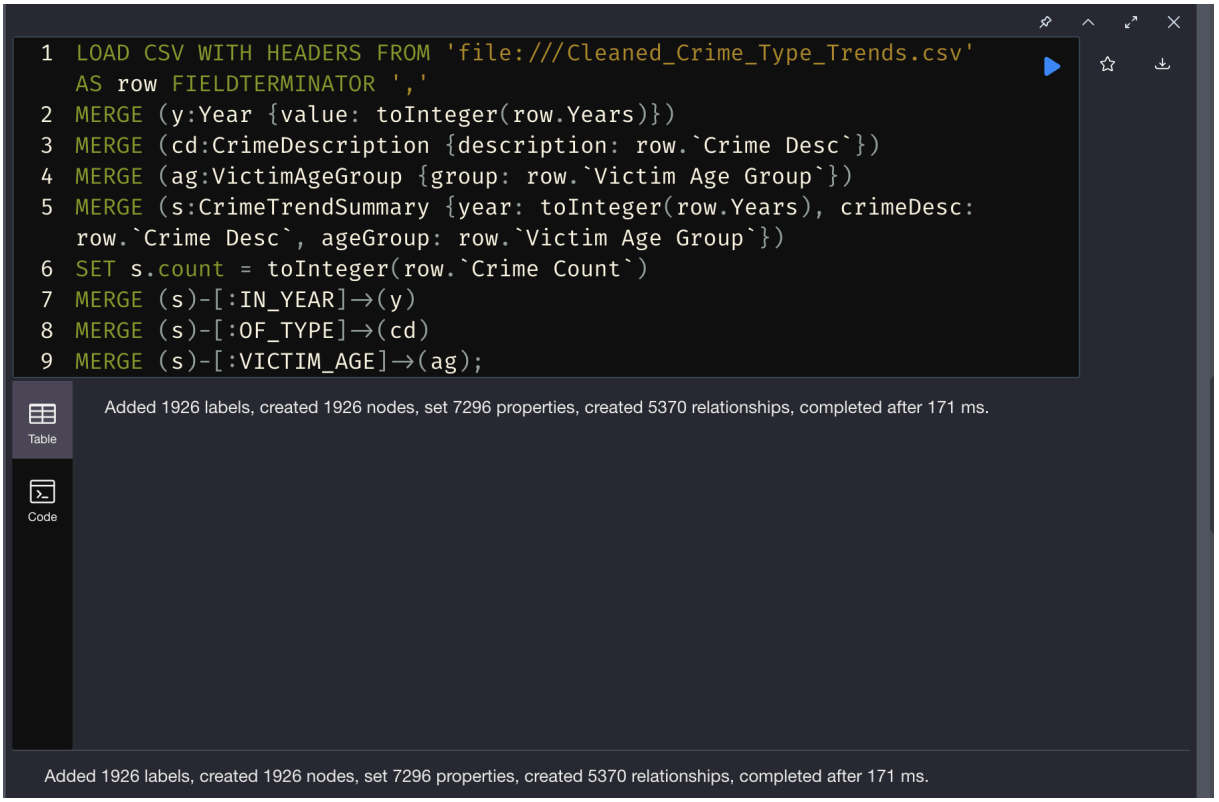
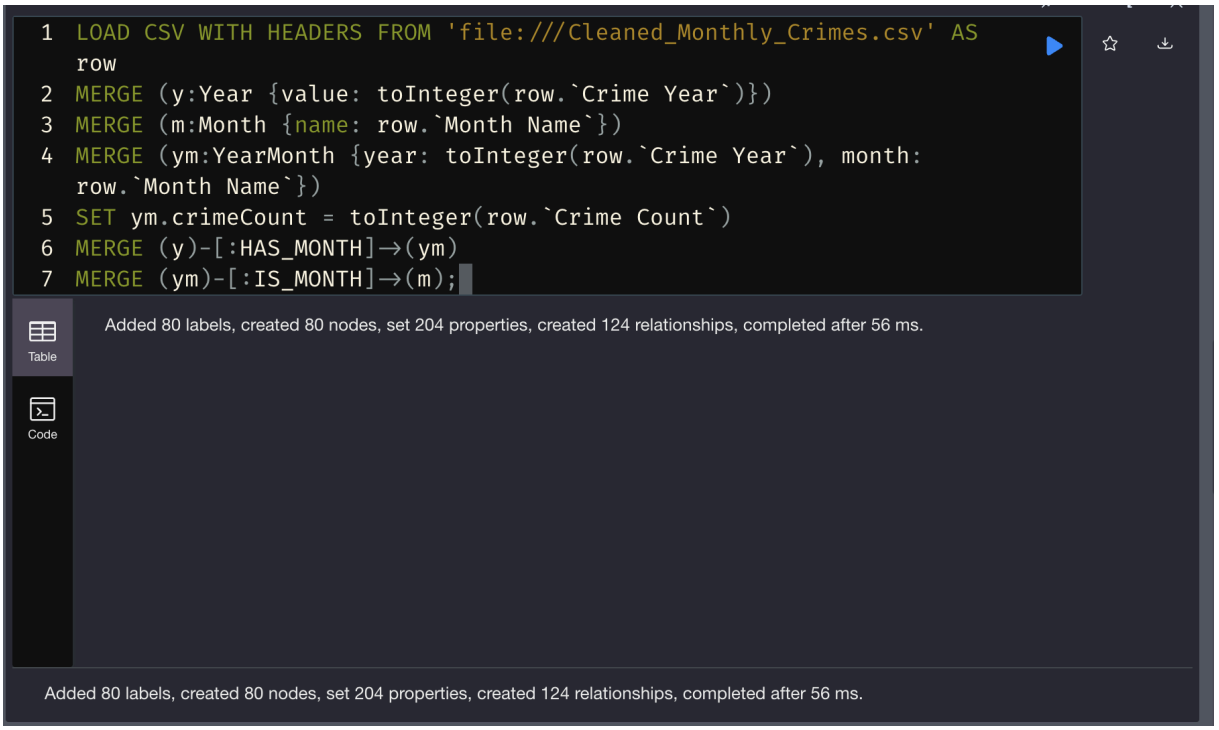


Figure 69:Node Population Continued



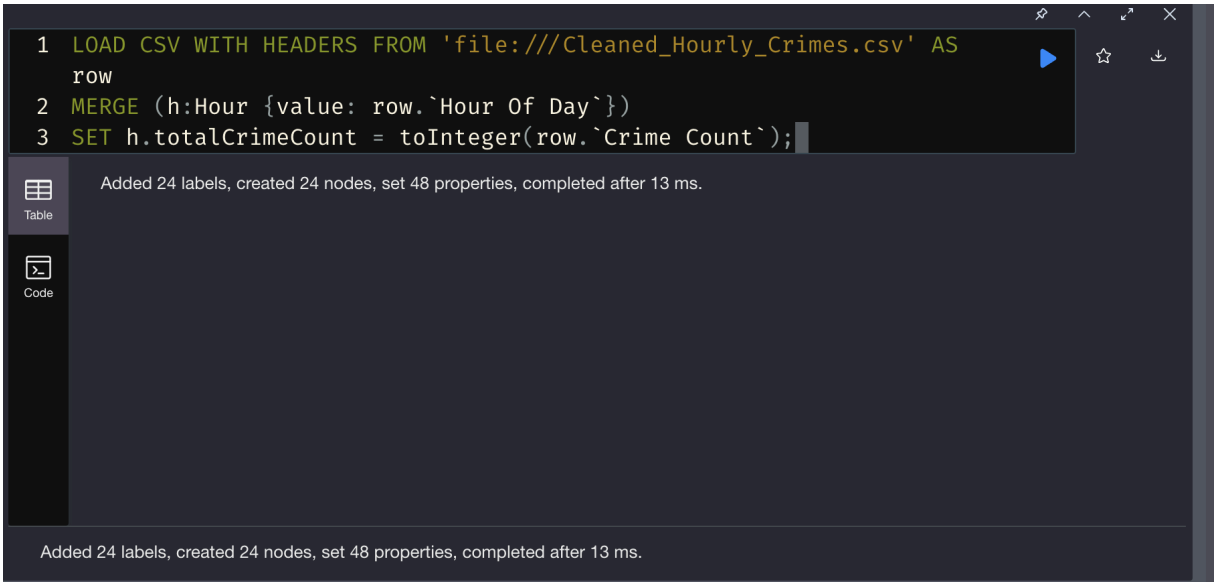


Figure 70:Node Population Continued

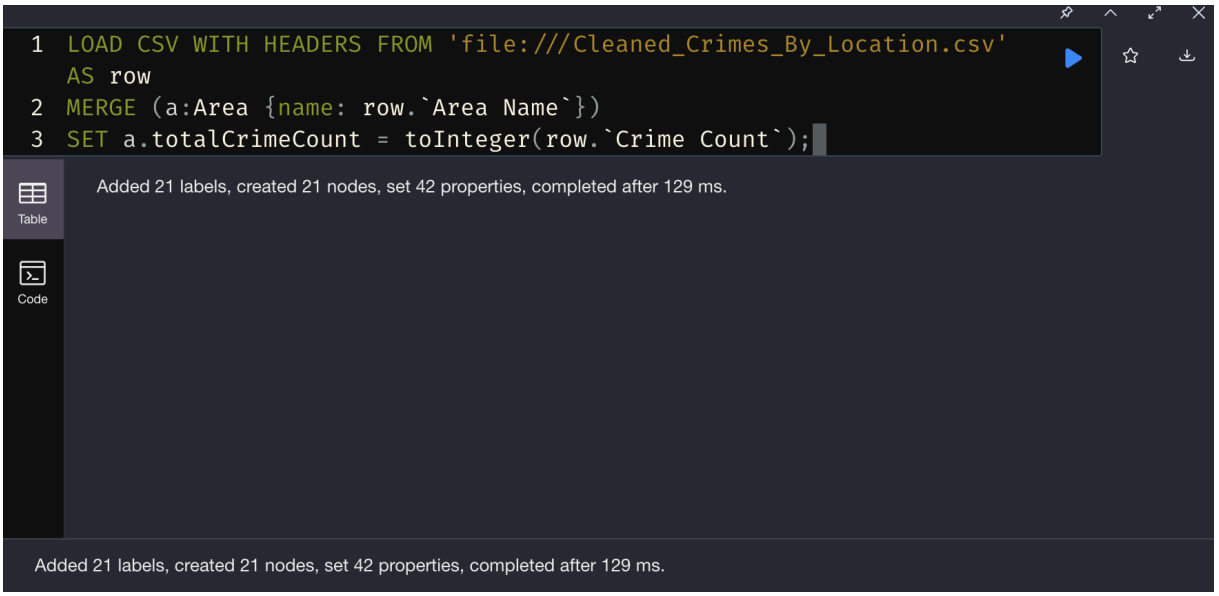


Figure 71:Node Population Continued

Query Comparison and Analysis

We performed seven types of queries in both systems to compare their capabilities and performance:

1. Simple Aggregation: Total Battery/Simple Assault Crimes Against Victims 45+

PostgreSQL:

SELECT SUM(crime_count) AS total_battery_simple_assault_45plus

FROM crime_type_trends

WHERE crime_desc = 'BATTERY - SIMPLE ASSAULT'

AND victim_age_group = '45+';

Query time: 00:00:00.056 seconds

Neo4j:

MATCH (s:CrimeTrendSummary)-[:OF_TYPE]->(:CrimeDescription {description: 'BATTERY - SIMPLE ASSAULT'})

MATCH (s)-[:VICTIM_AGE]->(:VictimAgeGroup {group: '45+'})

RETURN SUM(s.count) AS total_battery_simple_assault_45plus;

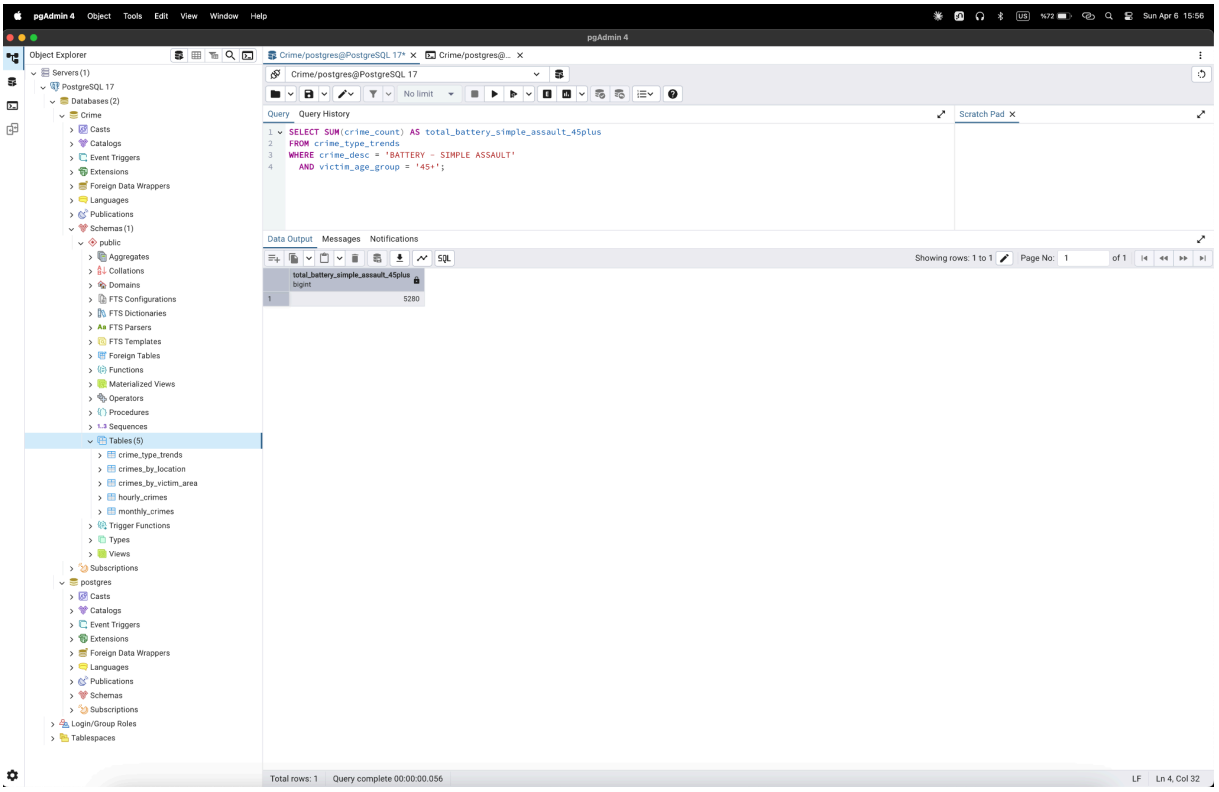


Figure 72:First Query for PostgreSQL

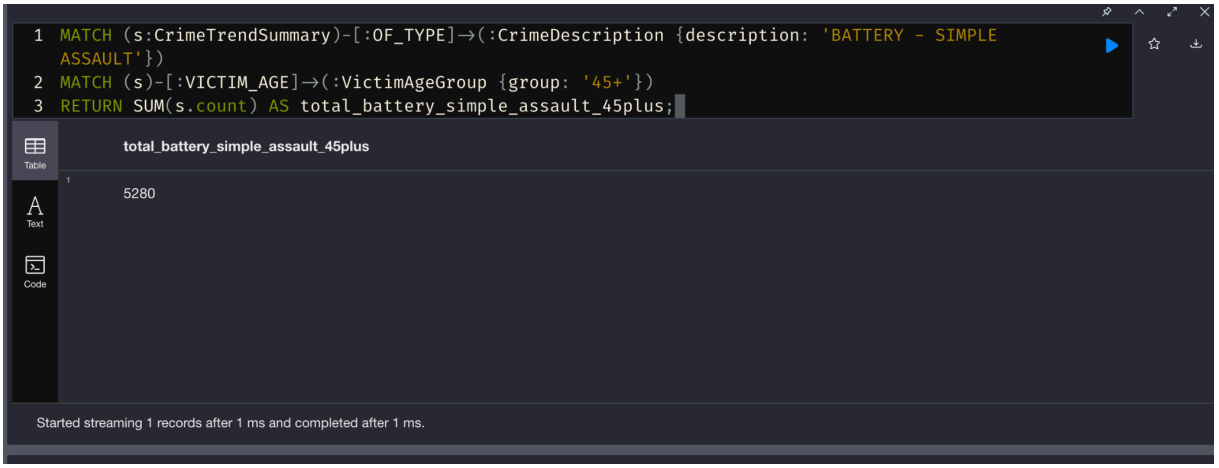


Figure 73:First Query for Neo4j

Analysis: Both systems handled this simple aggregation efficiently. PostgreSQL's flat table approach makes this a straightforward query, while Neo4j required traversing relationships to reach the same data. For simple queries like this, PostgreSQL's syntax is more concise.

2. Top Crime Types by Total Count

PostgreSQL:

SELECT crime_desc, SUM(crime_count) as total_count

FROM crime_type_trends

GROUP BY crime_desc

ORDER BY total_count DESC

LIMIT 5;

Query time: 00:00:00.067 seconds

Neo4j:

MATCH (cd:CrimeDescription)<-[:OF_TYPE]-(s:CrimeTrendSummary)

RETURN cd.description AS crime_desc, SUM(s.count) AS total_count

ORDER BY total_count DESC

LIMIT 5;

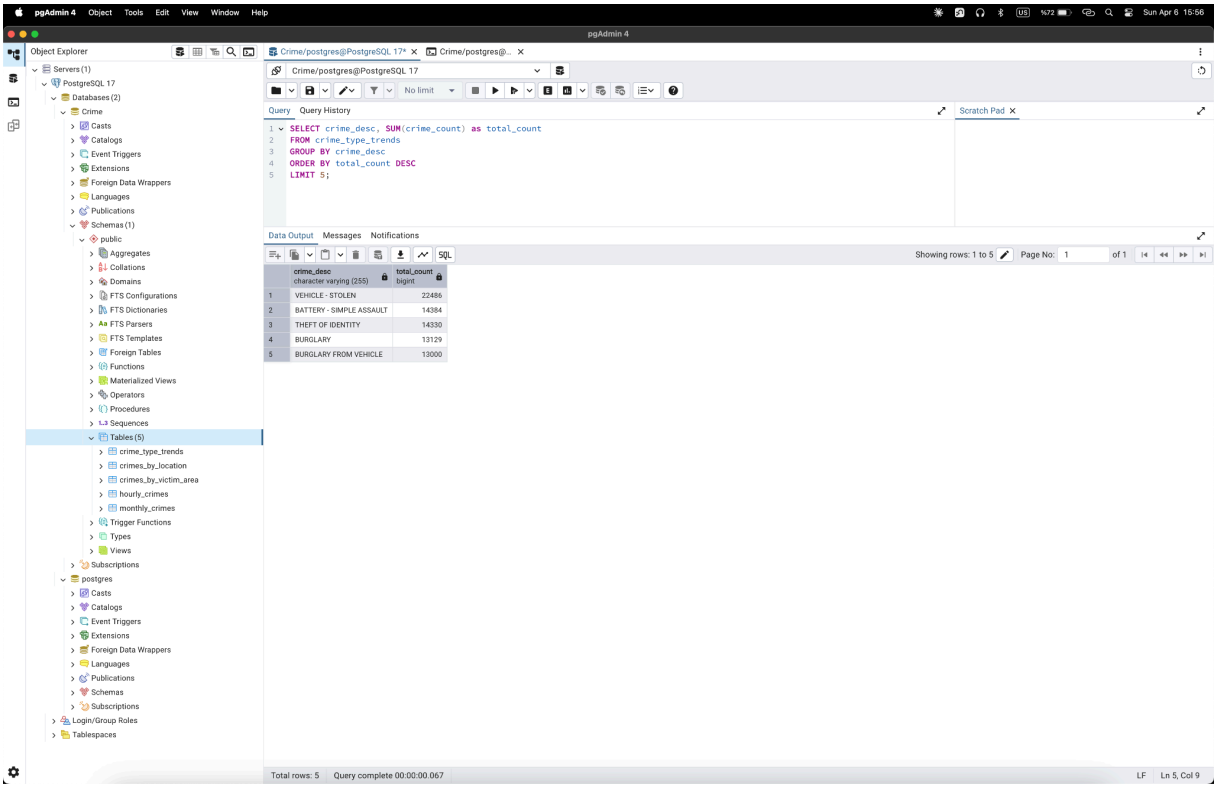


Figure 74:Second Query for PostgreSQL

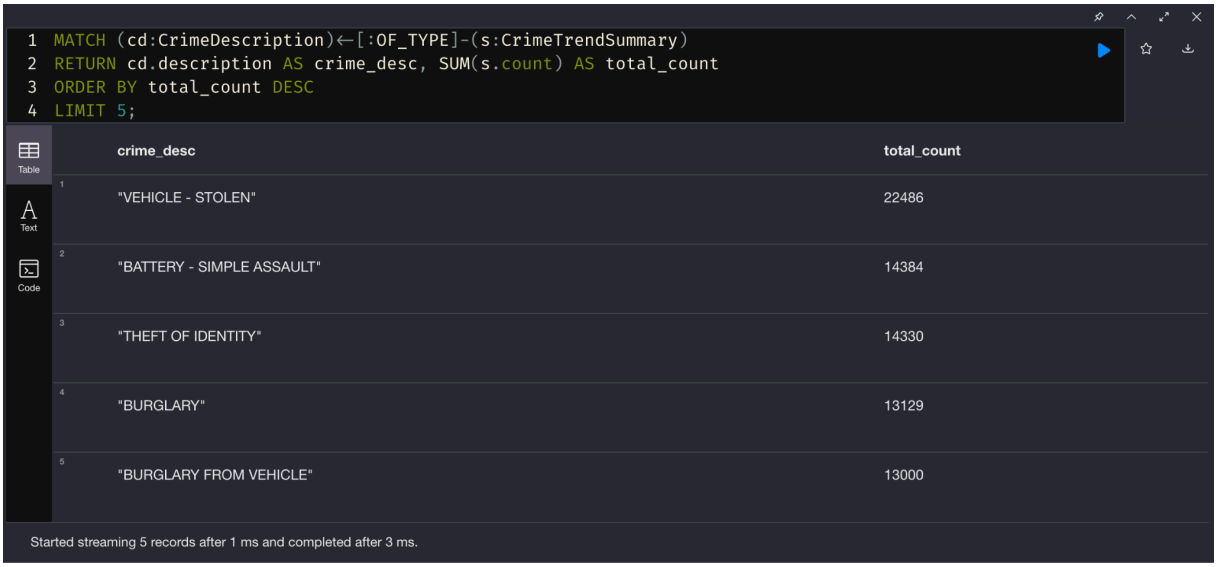


Figure 75:Second Query for Neo4j

Analysis: Both systems produced identical results, with PostgreSQL showing slightly better performance. The top crimes were Vehicle Theft (22,486), Battery - Simple Assault (14,384), Theft of Identity (14,330), Burglary (13,129), and Burglary from Vehicle (13,000). The query complexity was comparable, though Neo4j requires understanding of graph traversal concepts.

3. Distribution of Crime Types Affecting Female Victims in Pacific Area

PostgreSQL:

```
SELECT crime_type, crime_count
FROM crimes_by_victim_area
WHERE area_name = 'Pacific' AND victim_gender = 'F'
ORDER BY crime_count DESC;
```

Query time: 00:00:00.072 seconds

Neo4j:

```
MATCH (a:Area {name: 'Pacific'})<-[:IN_AREA]-(s:VictimAreaSummary)
MATCH (s)-[:VICTIM_GENDER]->(g:VictimGender {gender: 'F'})
MATCH (s)-[:OF_TYPE]->(cd:CrimeDescription)
RETURN cd.description AS crime_type, s.count AS crime_count
ORDER BY crime_count DESC;
```

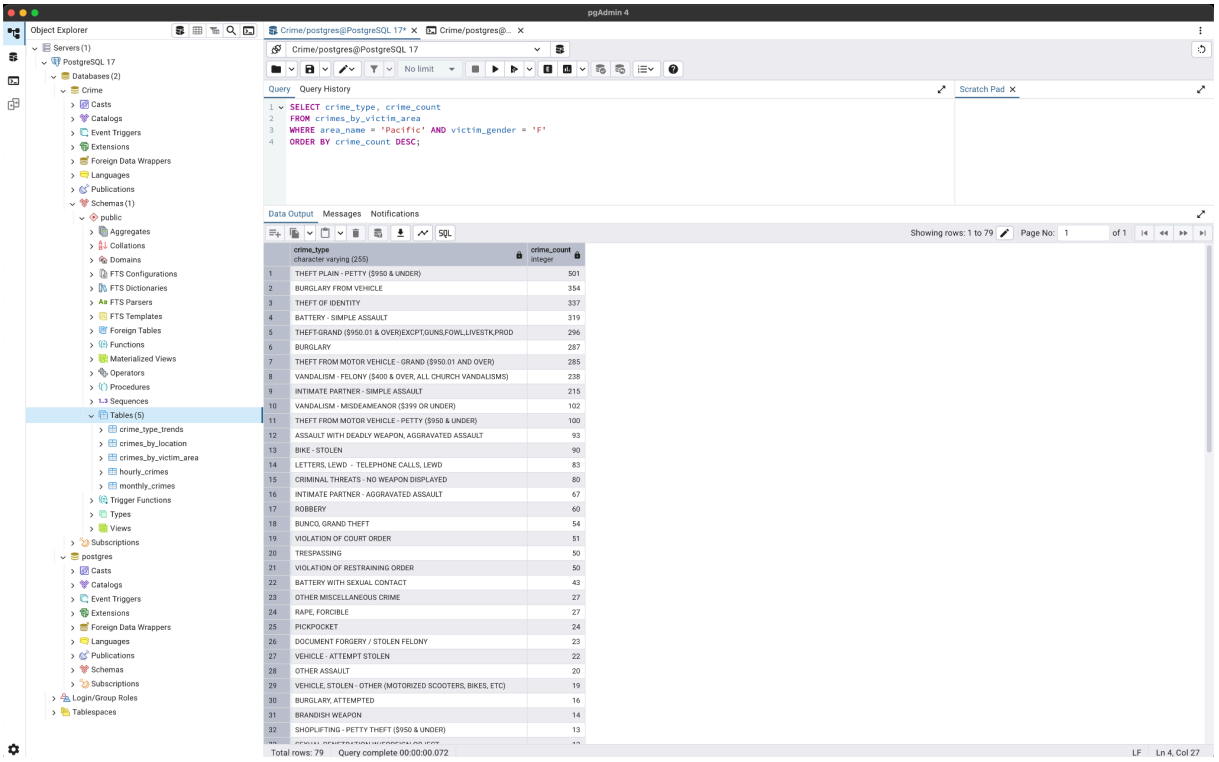


Figure 76:Third Query for PostgreSQL

33	SEXUAL PENETRATION W/FOREIGN OBJECT	13
34	DISTURBING THE PEACE	11
35	CRM AGNST CHLD (13 OR UNDER) (14-15 & SUSP 10 YRS OLDER)	11
36	ATTEMPTED ROBBERY	10
37	THEFT, PERSON	9
38	CHILD ANNOYING (17YRS & UNDER)	8
39	BUNCO, PETTY THEFT	8
40	LEWD CONDUCT	8
41	SODOMY/SEXUAL CONTACT B/W PENIS OF ONE PERS TO ANUS OTH	8
42	CHILD ABUSE (PHYSICAL) - SIMPLE ASSAULT	7
43	THEFT FROM MOTOR VEHICLE - ATTEMPT	6
44	BOMB SCARE	6
45	ARSON	5
46	PEEPING TOM	5
47	BURGLARY FROM VEHICLE, ATTEMPTED	5
48	KIDNAPPING	4
49	THROWING OBJECT AT MOVING VEHICLE	4
50	SEX,UNLAWFUL(INC MUTUAL CONSENT, PENETRATION W/ FRGN OBJ	4
51	PROWLER	4
52	ORAL COPULATION	4
53	INDECENT EXPOSURE	4
54	CREDIT CARDS, FRAUD USE ($950.01 & OVER)	3
55	STALKING	3
56	RAPE, ATTEMPTED	3
57	CHILD NEGLECT (SEE 300 W.I.C.)	3
58	CRIMINAL HOMICIDE	2
59	VIOLATION OF TEMPORARY RESTRAINING ORDER	2
60	UNAUTHORIZED COMPUTER ACCESS	2
61	THREATENING PHONE CALLS/LETTERS	2
62	SHOTS FIRED AT INHABITED DWELLING	2
63	PIMPING	2
64	FALSE IMPRISONMENT	2

Figure 77:Second Query for PostgreSQL Continued

65	EXTORTION	2
66	BATTERY POLICE (SIMPLE)	2
67	CONTEMPT OF COURT	1
68	VEHICLE - STOLEN	1
69	TELEPHONE PROPERTY - DAMAGE	1
70	SHOTS FIRED AT MOVING VEHICLE, TRAIN OR AIRCRAFT	1
71	PURSE SNATCHING	1
72	KIDNAPPING - GRAND ATTEMPT	1
73	HUMAN TRAFFICKING - INVOLUNTARY SERVITUDE	1
74	HUMAN TRAFFICKING - COMMERCIAL SEX ACTS	1
75	EMBEZZLEMENT, GRAND THEFT ($950.01 & OVER)	1
76	DOCUMENT WORTHLESS ($200.01 & OVER)	1
77	CREDIT CARDS, FRAUD USE ($950 & UNDER	1
78	CHILD ABUSE (PHYSICAL) - AGGRAVATED ASSAULT	1
79	CHILD PORNOGRAPHY	1
Total rows: 79	Query complete 00:00:00.072	

Figure 78: Second Query for PostgreSQL Continued

	crime_type	crime_count
1	"THEFT PLAIN - PETTY ($950 & UNDER)"	501
2	"BURGLARY FROM VEHICLE"	354
3	"THEFT OF IDENTITY"	337
4	"BATTERY - SIMPLE ASSAULT"	319
5	"THEFT-GRAND ($950.01 & OVER)EXCPT,GUNS,FOWL,LIVESTK,PROD"	296
6	"BURGLARY"	287
7	"THEFT FROM MOTOR VEHICLE - GRAND ($950.01 AND OVER)"	285
8	"VANDALISM - FELONY ($400 & OVER, ALL CHURCH VANDALISMS)"	238
9	"INTIMATE PARTNER - SIMPLE ASSAULT"	215
10	"VANDALISM - MISDEAMEANOR ($399 OR UNDER)"	102
11	"THEFT FROM MOTOR VEHICLE - PETTY ($950 & UNDER)"	100
12	"ASSAULT WITH DEADLY WEAPON, AGGRAVATED ASSAULT"	93
13	"BIKE - STOLEN"	90
14	"LETTERS, LEWD - TELEPHONE CALLS, LEWD"	83
15	"CRIMINAL THREATS - NO WEAPON DISPLAYED"	80
16	"INTIMATE PARTNER - AGGRAVATED ASSAULT"	67

Figure 79:Third Query for Neo4j

	crime_type	crime_count
17	"ROBBERY"	60
18	"BUNCO, GRAND THEFT"	54
19	"VIOLATION OF COURT ORDER"	51
20	"TRESPASSING"	50
21	"VIOLATION OF RESTRAINING ORDER"	50
22	"BATTERY WITH SEXUAL CONTACT"	43
23	"OTHER MISCELLANEOUS CRIME"	27
24	"RAPE, FORCIBLE"	27
25	"PICKPOCKET"	24
26	"DOCUMENT FORGERY / STOLEN FELONY"	23
27	"VEHICLE - ATTEMPT STOLEN"	22
28	"OTHER ASSAULT"	20
29	"VEHICLE, STOLEN - OTHER (MOTORIZED SCOOTERS, BIKES, ETC)"	19
30	"BURGLARY, ATTEMPTED"	16
31	"BRANDISH WEAPON"	14
32	"SEXUAL PENETRATION W/FOREIGN OBJECT"	13
33		

ted streaming 79 records after 10 ms and completed after 23 ms.

Figure 80:Third Query for Neo4j Continued

33	"SHOPLIFTING - PETTY THEFT ($950 & UNDER)"	13
34	"CRM AGNST CHLD (13 OR UNDER) (14-15 & SUSP 10 YRS OLDER)"	11
35	"DISTURBING THE PEACE"	11
36	"ATTEMPTED ROBBERY"	10
37	"THEFT, PERSON"	9
38	"BUNCO, PETTY THEFT"	8
39	"CHILD ANNOYING (17YRS & UNDER)"	8
40	"LEWD CONDUCT"	8
41	"SODOMY/SEXUAL CONTACT B/W PENIS OF ONE PERS TO ANUS OTH"	8
42	"CHILD ABUSE (PHYSICAL) - SIMPLE ASSAULT"	7
43	"BOMB SCARE"	6
44	"THEFT FROM MOTOR VEHICLE - ATTEMPT"	6
45	"ARSON"	5
46	"BURGLARY FROM VEHICLE, ATTEMPTED"	5
47	"PEEPING TOM"	5
48	"INDECENT EXPOSURE"	4

Figure 81:Third Query for Neo4j Continued

49	"KIDNAPPING"	4
50	"ORAL COPULATION"	4
51	"PROWLER"	4
52	"SEX,UNLAWFUL(INC MUTUAL CONSENT, PENETRATION W/ FRGN OBJ"	4
53	"THROWING OBJECT AT MOVING VEHICLE"	4
54	"CHILD NEGLECT (SEE 300 W.I.C.)"	3
55	"CREDIT CARDS, FRAUD USE ($950.01 & OVER)"	3
56	"RAPE, ATTEMPTED"	3
57	"STALKING"	3
58	"BATTERY POLICE (SIMPLE)"	2
59	"CRIMINAL HOMICIDE"	2
60	"EXTORTION"	2
61	"FALSE IMPRISONMENT"	2
62	"PIMPING"	2
63	"SHOTS FIRED AT INHABITED DWELLING"	2
64	"THREATENING PHONE CALLS/LETTERS"	2

Figure 82:Third Query for Neo4j Continued

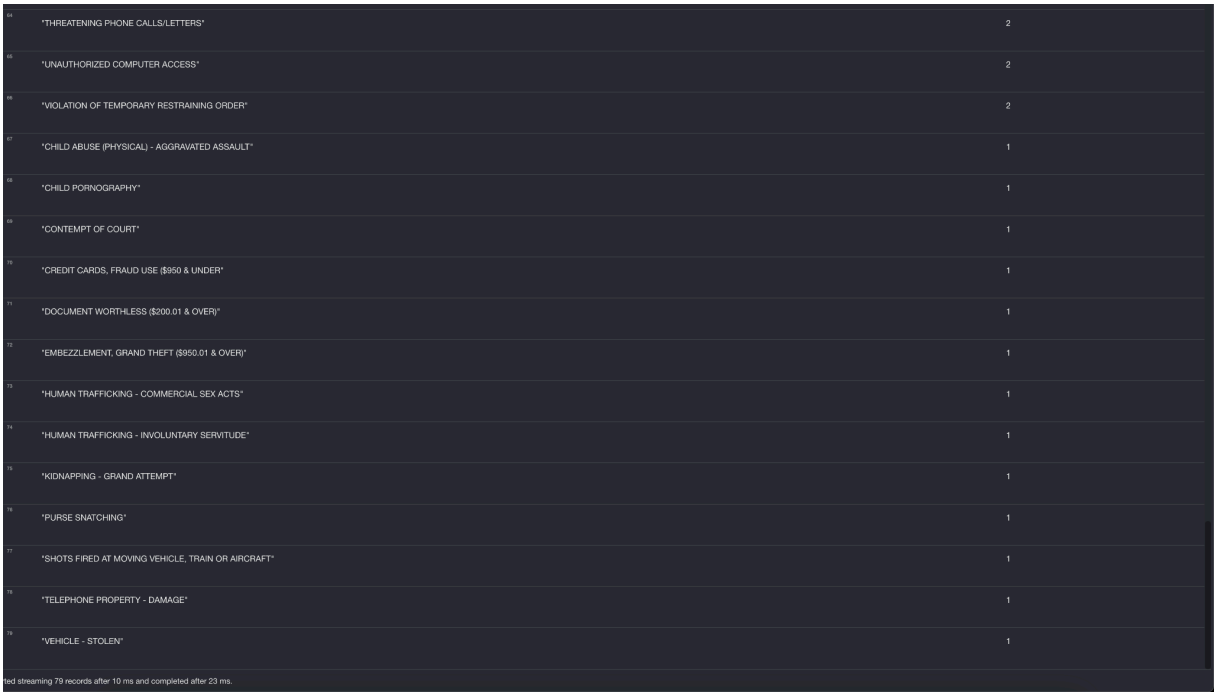


Figure 83:Third Query for Neo4j Continued

Analysis: This query revealed 79 different crime types affecting female victims in the Pacific area, with Theft Plain - Petty (501), Burglary from Vehicle (354), and Theft of Identity (337) being most common. Neo4j's query was more verbose, reflecting the relationship-centric nature of the graph database, but enabled more intuitive extensions when exploring related data. In terms of execution time neo4j was faster than SQL this time.

4. Arson Crimes by Year and Victim Age Group

PostgreSQL:

```
SELECT years, victim_age_group, crime_count
FROM crime_type_trends
WHERE crime_desc = 'ARSON'
ORDER BY years, victim_age_group;
```

Query time: 00:00:00.086 seconds

Neo4j:

```
MATCH (s:CrimeTrendSummary)-[:OF_TYPE]->(cd:CrimeDescription {description: 'ARSON'})
MATCH (s)-[:IN_YEAR]->(y:Year)
MATCH (s)-[:VICTIM_AGE]->(ag:VictimAgeGroup)
```

RETURN y.value AS years, ag.group AS victim_age_group, s.count AS crime_count

ORDER BY years, victim_age_group;

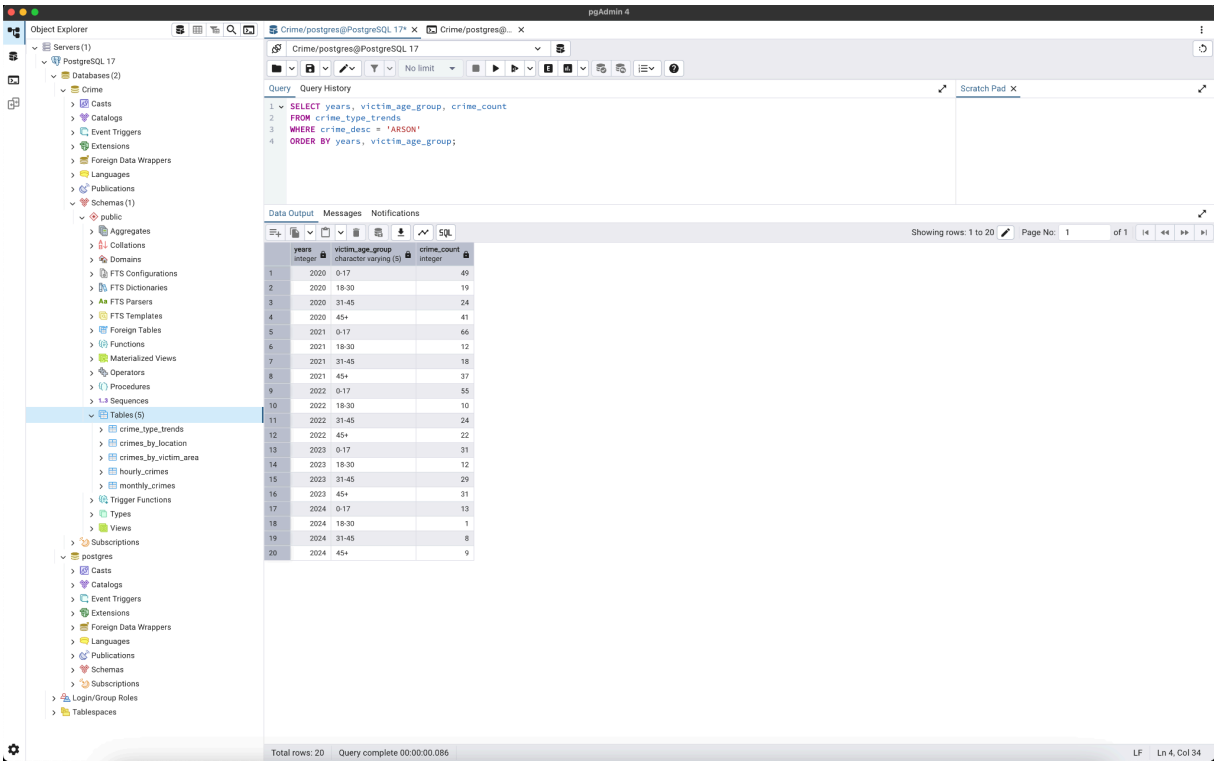


Figure 84:Fourth Query for PostgreSQL

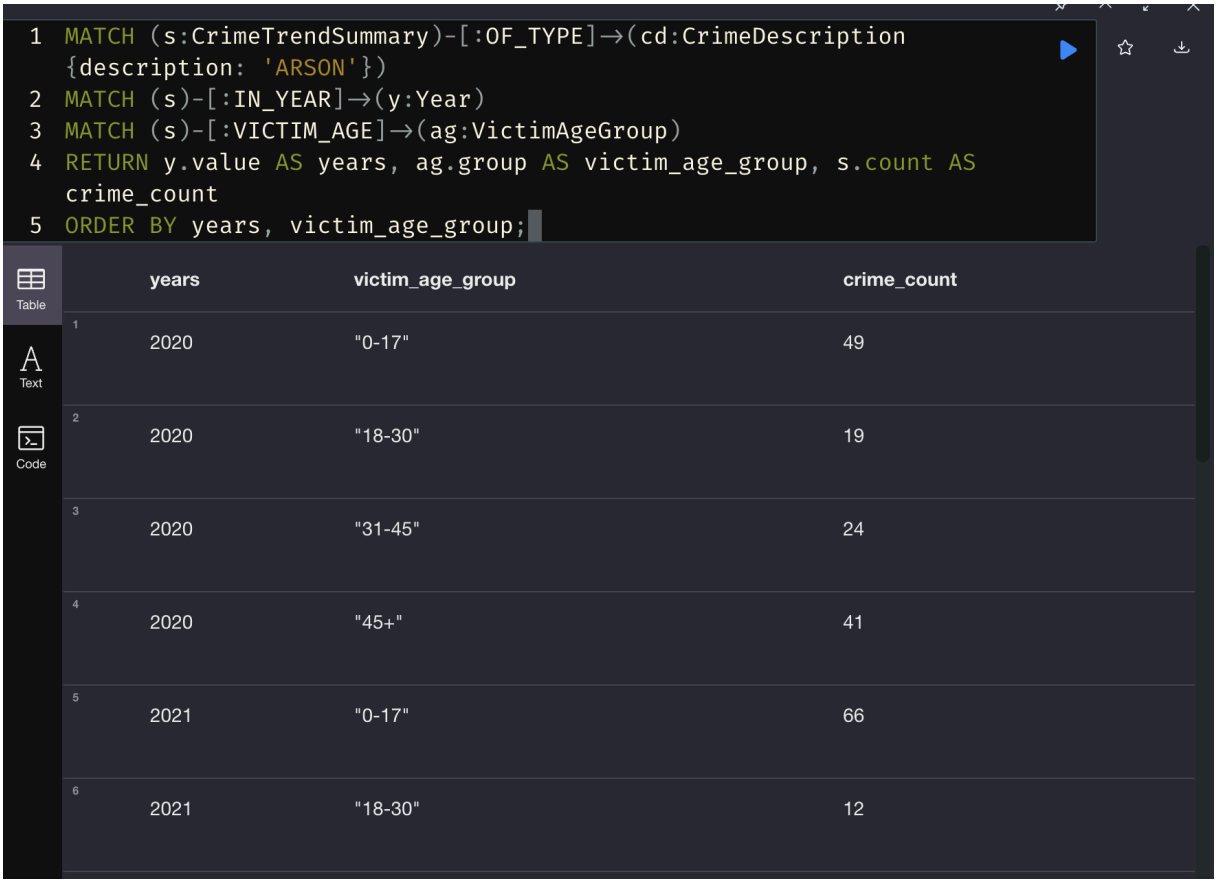
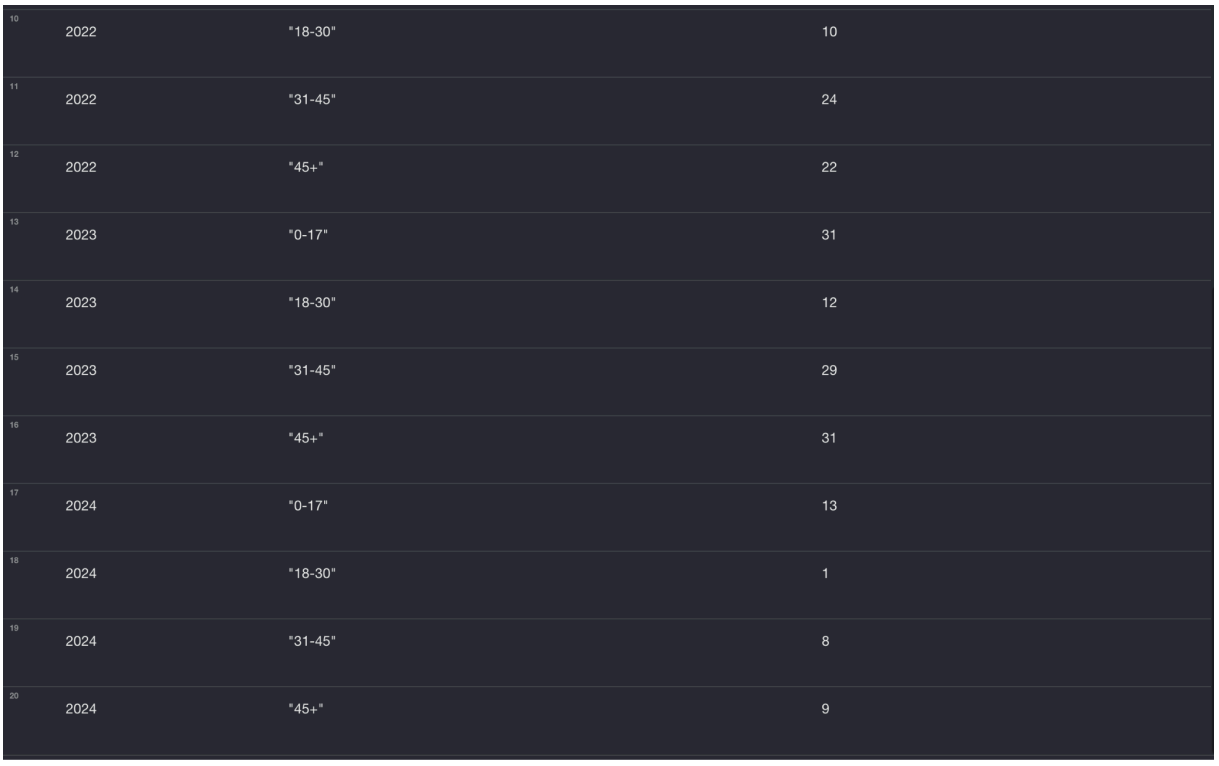


Figure 85:Fourth Query for Neo4j

7	2021	"31-45"	18
8	2021	"45+"	37
9	2022	"0-17"	55
10	2022	"18-30"	10
11	2022	"31-45"	24
12	2022	"45+"	22

Figure 86:Fourth Query for Neo4j Continued



10	2022	"18-30"	10
11	2022	"31-45"	24
12	2022	"45+"	22
13	2023	"0-17"	31
14	2023	"18-30"	12
15	2023	"31-45"	29
16	2023	"45+"	31
17	2024	"0-17"	13
18	2024	"18-30"	1
19	2024	"31-45"	8
20	2024	"45+"	9

Figure 87:Fourth Query for Neo4j Continued

Analysis: This query showed a detailed breakdown of arson crimes across different years (2020-2024) and victim age groups. The results revealed patterns like a spike in arson crimes against the 0-17 age group in 2022 (55 incidents). Neo4j's approach, while more verbose, creates clearer semantic meaning through named relationships.

5. Crime Distribution by Area

PostgreSQL:

```
SELECT area_name, crime_count
FROM crimes_by_location
ORDER BY crime_count DESC;
```

Query time: 00:00:00.110 seconds

Neo4j:

```
MATCH (a:Area)
WHERE a.totalCrimeCount IS NOT NULL
RETURN a.name AS area_name, a.totalCrimeCount AS crime_count
ORDER BY crime_count DESC;
```

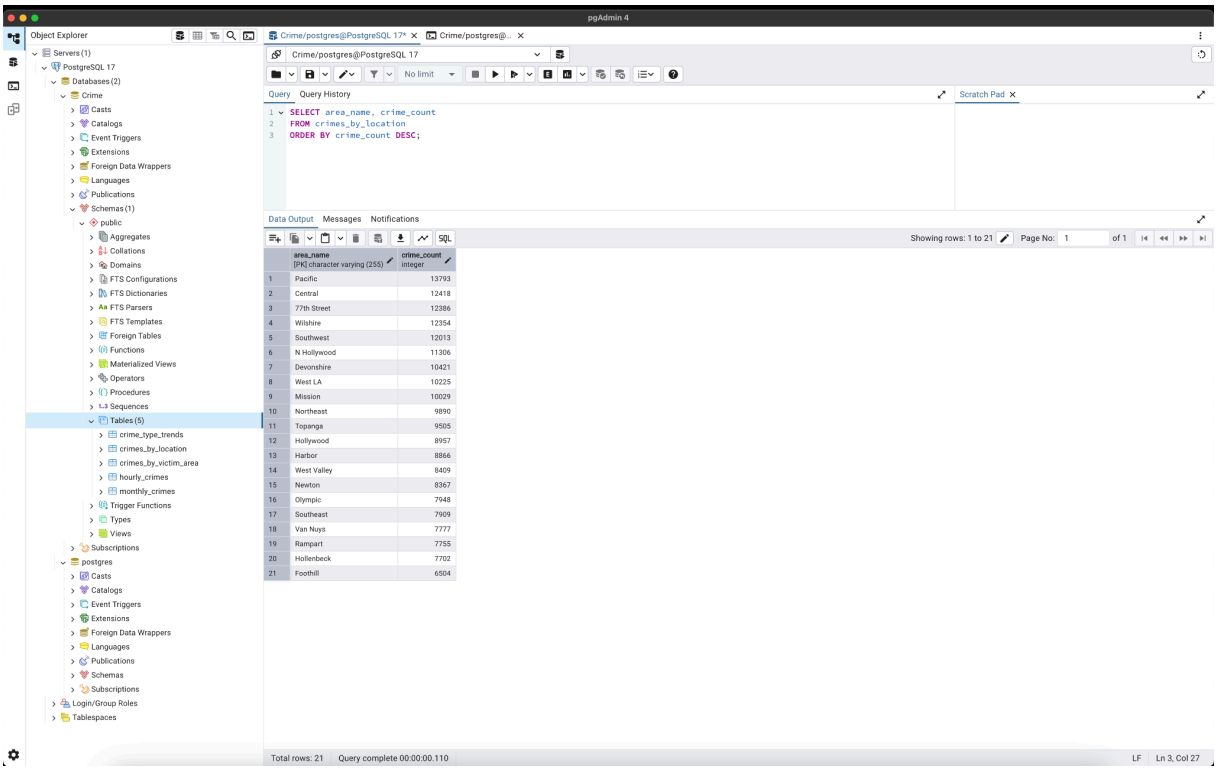


Figure 88:Fifth Query for PostgreSQL

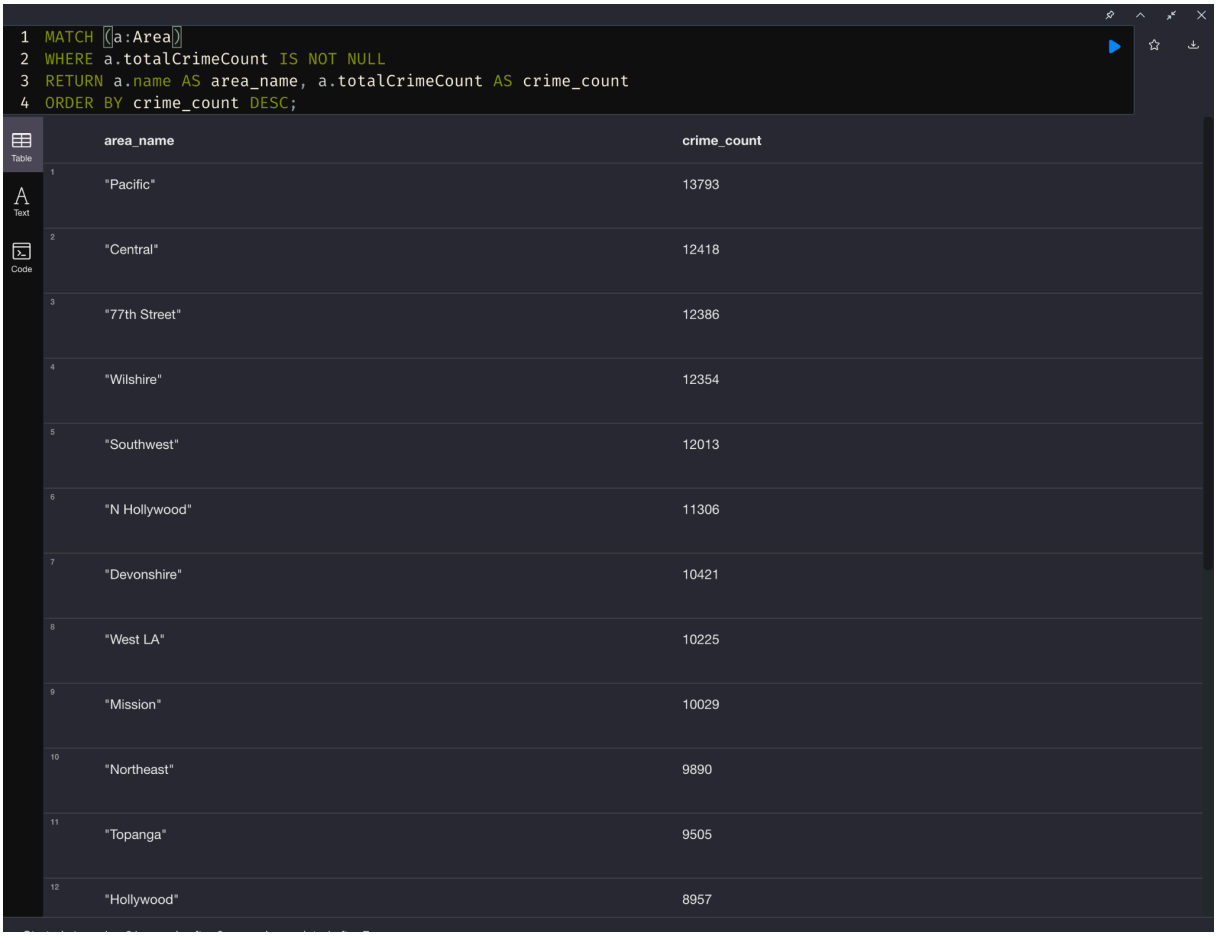


Figure 89:Fifth Query for Neo4j

Analysis: This query identified Pacific (13,793), Central (12,418), and 77th Street (12,386) as the top three crime-heavy areas. Both systems handled this basic query efficiently, with virtually identical syntax and performance.

6. Crimes by Year (Yearly Trends)

PostgreSQL:

SELECT crime_year, SUM(crime_count) AS total_yearly_crimes

FROM monthly_crimes

GROUP BY crime_year

ORDER BY crime_year;

Query time: 00:00:00.141 seconds

Neo4j:

MATCH (y:Year)-[:HAS_MONTH]->(ym:YearMonth)

RETURN y.value AS crime_year, SUM(ym.crimeCount) AS total_yearly_crimes

ORDER BY crime_year;

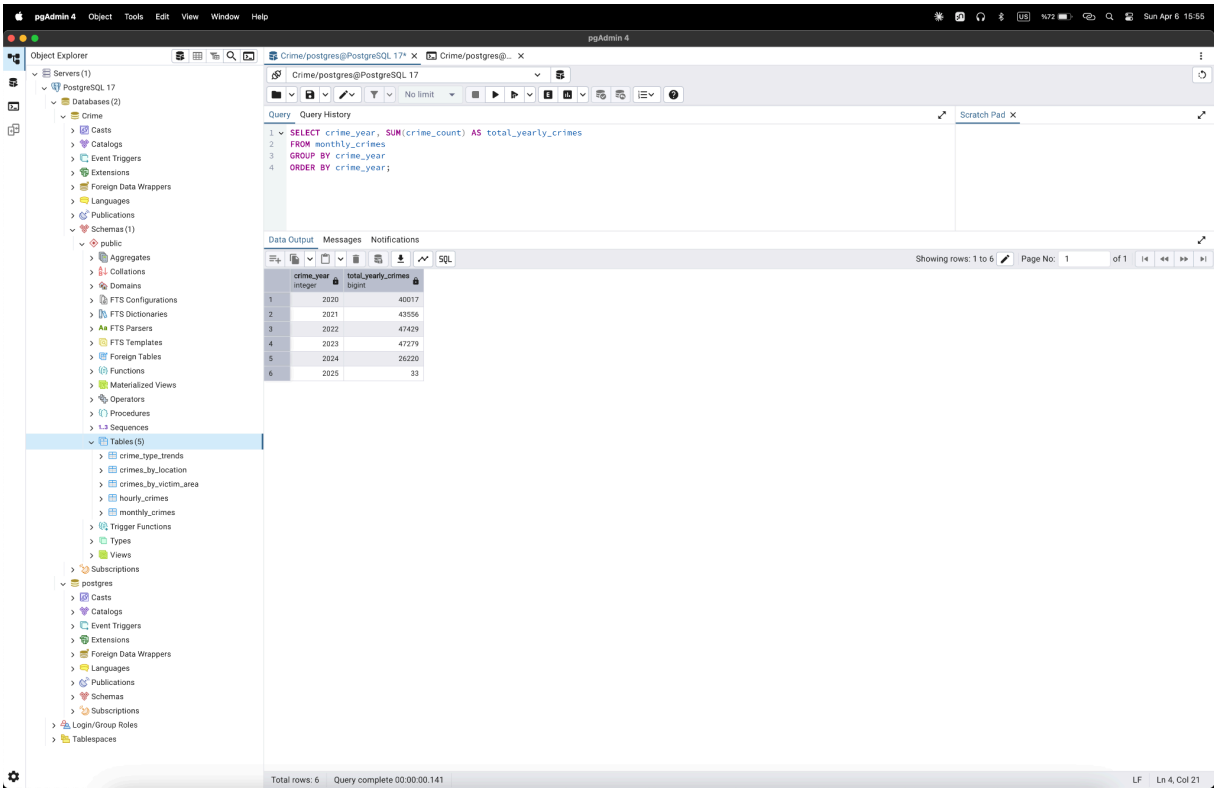


Figure 90:Sixth Query for PostgreSQL

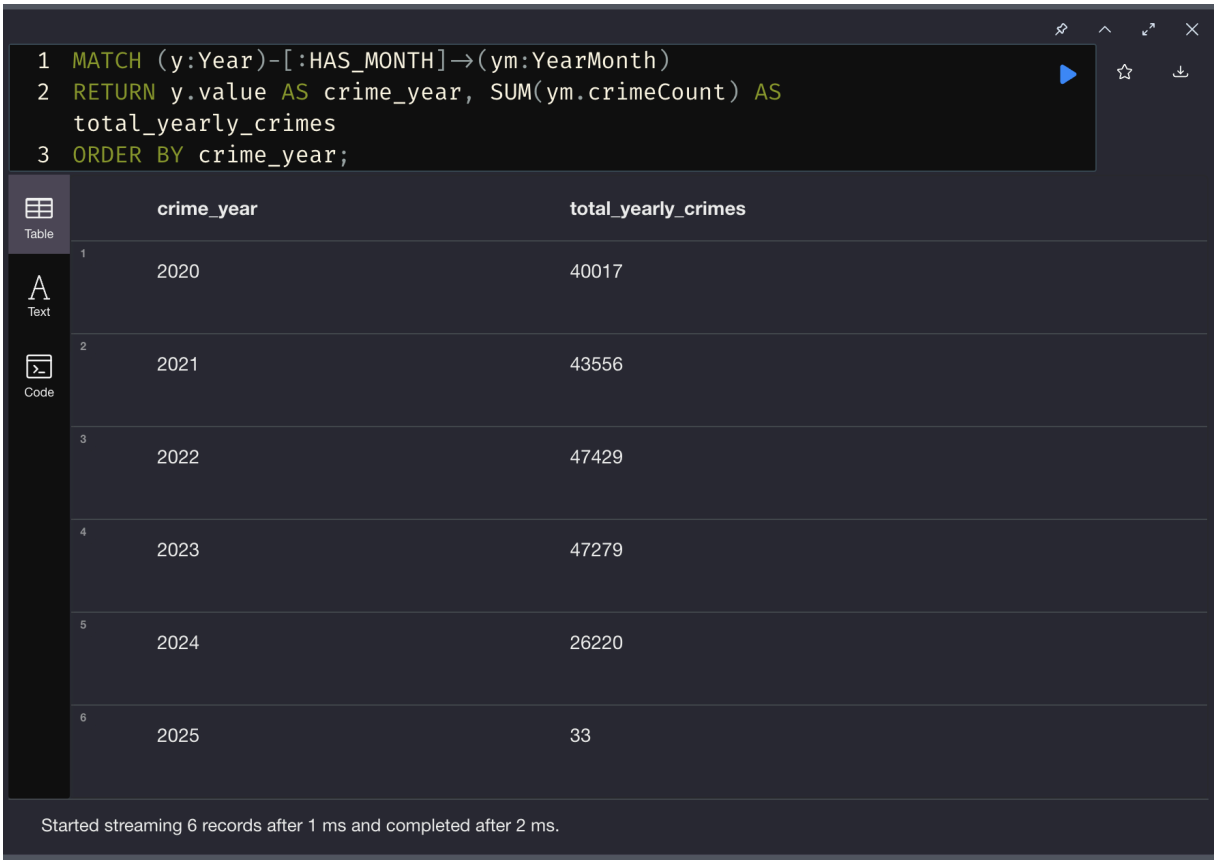


Figure 91:Sixth Query for Neo4j

Analysis: This query showed crime trends from 2020 (40,017 crimes) through 2025 (33 crimes, likely partial data). Neo4j's relationship-based approach required explicit traversal from Year to YearMonth nodes, whereas PostgreSQL used a simple GROUP BY on a single table.

7. Top 3 Hours with Most Crimes

PostgreSQL:

SELECT hour_of_day, crime_count

FROM hourly_crimes

ORDER BY crime_count DESC

LIMIT 3;

Query time: 00:00:00.087 seconds

Neo4j:

MATCH (h:Hour)

WHERE h.totalCrimeCount IS NOT NULL

RETURN h.value AS hour_of_day, h.totalCrimeCount AS crime_count

ORDER BY crime_count DESC

LIMIT 3;

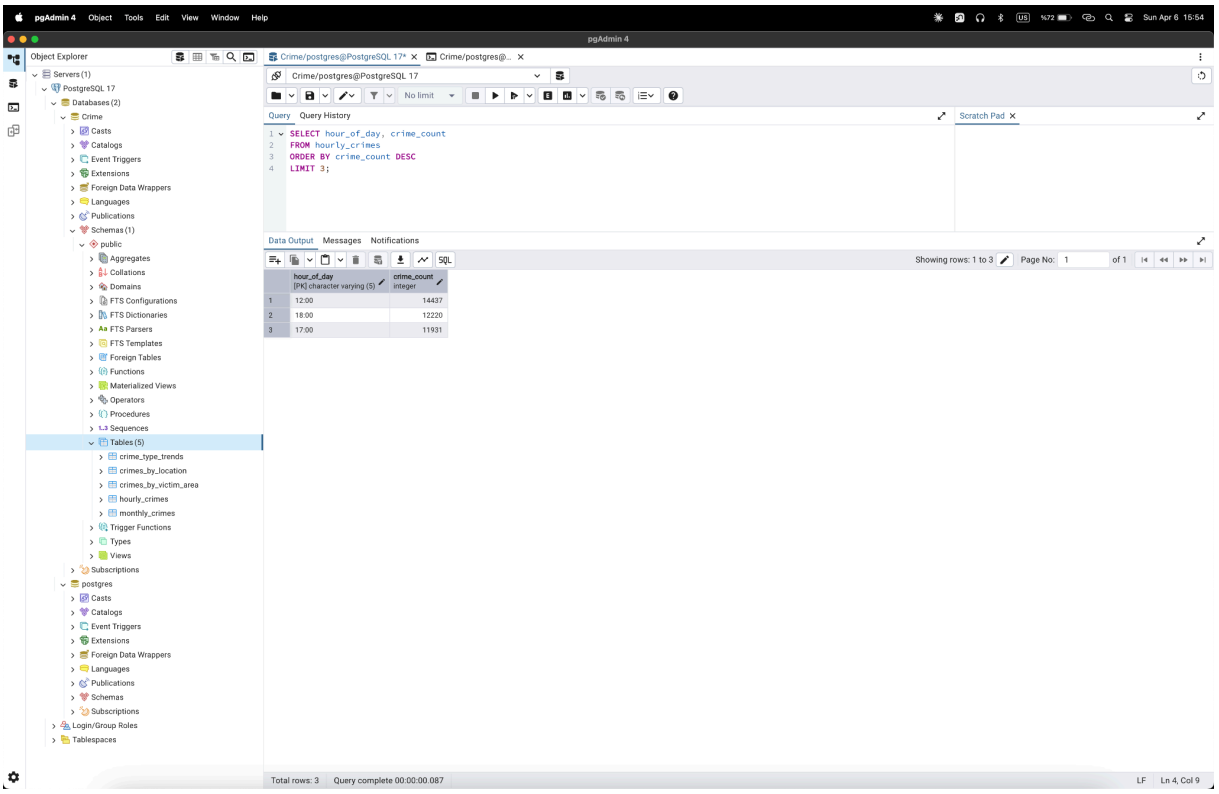


Figure 92:Seventh Query for PostgreSQL

Figure 93:Seventh Query for Neo4j

Analysis: This query revealed that crimes peak at 12:00 (14,437), 18:00 (12,220), and 17:00 (11,931). Both database systems handled this simple query with similar syntax and performance, demonstrating that for straightforward analytics, either system works well.

Key Findings and Comparative Analysis

Query Complexity

1. **Simple Queries:** For basic data retrieval and aggregation, PostgreSQL offers more concise syntax and slightly better performance.

2. **Complex Relationships:** Neo4j shines when dealing with multi-level relationships. While not fully demonstrated in these examples (which use pre-aggregated data), Neo4j's pattern matching capabilities make complex relationship traversals more intuitive.

3. **Semantic Clarity:** Neo4j queries offer better semantic clarity through named relationships. The meaning of (Crime)-[:OCCURRED_AT]->(Location) is immediately clear, whereas SQL joins don't inherently convey the nature of relationships.

Performance Considerations

1. **Query Execution Time:** For these specific queries on this dataset size, PostgreSQL showed slightly faster execution times, typically in the 60-150 millisecond range.

2. **Scalability:** While not tested in this implementation, graph databases typically scale better for relationship-intensive queries as data volume grows, while relational databases may require more complex indexing strategies.

3. **Data Import:** PostgreSQL's COPY command and Neo4j's LOAD CSV both provided efficient import mechanisms, though Neo4j required more complicated data modeling during import.

Model Flexibility

1. **Schema Evolution:** Neo4j's schema-flexible nature allows for easier addition of new properties or relationship types without affecting existing data. PostgreSQL would require ALTER TABLE operations that might be costly on large tables.

2. **Complex Patterns:** Neo4j excels at finding complex patterns within data, such as identifying similar crime patterns across different locations or finding chains of related crimes.

3. **Composite Keys:** Both systems effectively handled composite keys, though Neo4j's NODE KEY constraint provides a graph-specific implementation.

Summary

The comparison between PostgreSQL and Neo4j for crime data analysis reveals distinctive strengths for each database system:

PostgreSQL Strengths:

- More concise syntax for simple queries
- Slightly better performance for basic aggregations
- Familiar SQL language for analysts
- Well-established ecosystem of tools and integrations

Neo4j Strengths:

- More intuitive representation of relationships
- Better semantic clarity through named relationships
- Greater flexibility for evolving data models
- Potential for more efficient complex pattern matching at scale

For the specific crime analysis tasks demonstrated in this project, both database systems proved capable of handling the required analyses efficiently. The choice between PostgreSQL and Neo4j should therefore depend on:

1. The complexity of relationships in the data model
2. The types of queries most frequently performed
3. The existing skills and tooling in the organization
4. Future scalability and expansion requirements

Summary of LAPD Crime Data Warehouse Development

This project developed a data warehouse for LAPD crime data, focusing on enhanced analysis capabilities for public safety and law enforcement effectiveness. The team implemented a star schema with Fact_Crime at the center connected to five dimension tables (Time, Location, Crime Type, Victim, and Status), enabling multi-dimensional crime analysis across different attributes.

The database design utilized a normalized star schema architecture, allowing complex queries across multiple dimensions. This structure efficiently supports filtering by location, time periods, crime types, victim demographics, and case status, making it easier to identify patterns and trends than traditional systems.

Data preprocessing involved cleaning date and time fields to ensure consistent formatting across the dataset. The team used SQL Server Integration Services (SSIS) to create comprehensive ETL data flows for each dimension table, including transformations, lookups, and deduplication to maintain data quality and referential integrity throughout the pipeline.

The analytical implementation included SSRS reports addressing key questions: which locations have the most crime (Pacific, Central, and 77th Street areas), crime patterns by time (peaks at noon and early evening), distribution of crime types against female victims by area, and crime type trends by year and victim age group. These reports provided actionable insights for various stakeholders.

A Tableau dashboard complemented the SSRS reports with interactive visualizations, including a treemap of crime types by victim age group, hourly crime distribution histogram, location-based crime distribution, and focused analysis of crimes against female victims. These visualizations revealed that vehicle theft remains consistently high across all victim age groups, while identity theft disproportionately affects the 31-45 age group.

The team also implemented the same data model in both PostgreSQL (relational) and Neo4j (graph database) for comparison. PostgreSQL offered more concise syntax for simple queries with slightly better performance, while Neo4j provided better semantic clarity through named relationships. Both systems successfully handled all required analyses, with different strengths depending on query complexity.

The project successfully transformed raw crime data into a structured warehouse that enables law enforcement leadership, investigators, and community members to gain actionable insights for resource allocation, crime prevention strategies, and community safety initiatives. The multi-dimensional analysis capabilities allow stakeholders to identify geographic crime concentrations, temporal patterns, demographic vulnerabilities, and emerging crime trends.

References

Carnaz, G., Nogueira, V. B., Antunes, M., & Ferreira, N. (2019). An Automated System for Criminal Police Reports Analysis. In International Conference on Information Technology & Systems (pp. 350-359). Springer, Cham.

Cheng, T., & Williams, D. (2012). Space-Time Analysis of Crime Patterns in Central London. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XXXIX-B2, 47-52.

Garfias Royo, M., Parikh, P., & Belur, J. (2020). Using heat maps to identify areas prone to violence against women in the public sphere. Crime Science, 9(1), 1-15.

Michael, A. V., & Ahirao, P. (2020). Improved Use of ETL Tool for Updation and Creation of Data Warehouse From Different RDBMS. Proceedings of the 3rd International Conference on Advances in Science & Technology, 1-4.

Reehl, A., & Sharma, S. (2018). Data Visualization of Crime Data using Immersive Virtual Reality. Proceedings of the International Conference on Computer Graphics and Virtual Reality, 1-6.

Salcedo-Gonzalez, M., Suarez-Paez, J., Esteve, M., Gómez, J. A., & Palau, C. E. (2020). A Novel Method of Spatiotemporal Dynamic Geo-Visualization of Criminal Data, Applied to Command and Control Centers for Public Safety. ISPRS International Journal of Geo-Information, 9(3), 160.

Sharma, S., & Dronavalli, S. C. (2020). Data Analysis and Visualization of Crime Data. Proceedings of Visualization and Data Analysis 2020, 1(1), 364-1-364-8.

Sihombing, D. J. C. (2022). Academic Data Warehouse Modeling in Higher Education Using Nine-Step Design Methodology. Journal of Information Systems and Informatics, 4(4), 1126-1145.

Wei, Y., Wang, X., & Hu, X. (2012). Design and Implementation of ETL in Police Intelligence Data Warehouse System. Applied Mechanics and Materials, 121-126, 3865-3868.

Zhang, X., Sun, W., & Wang, W. (2006). Generating incremental ETL processes automatically. Proceedings of the First International Multi-Symposiums on Computer and Computational Sciences, 516-521.