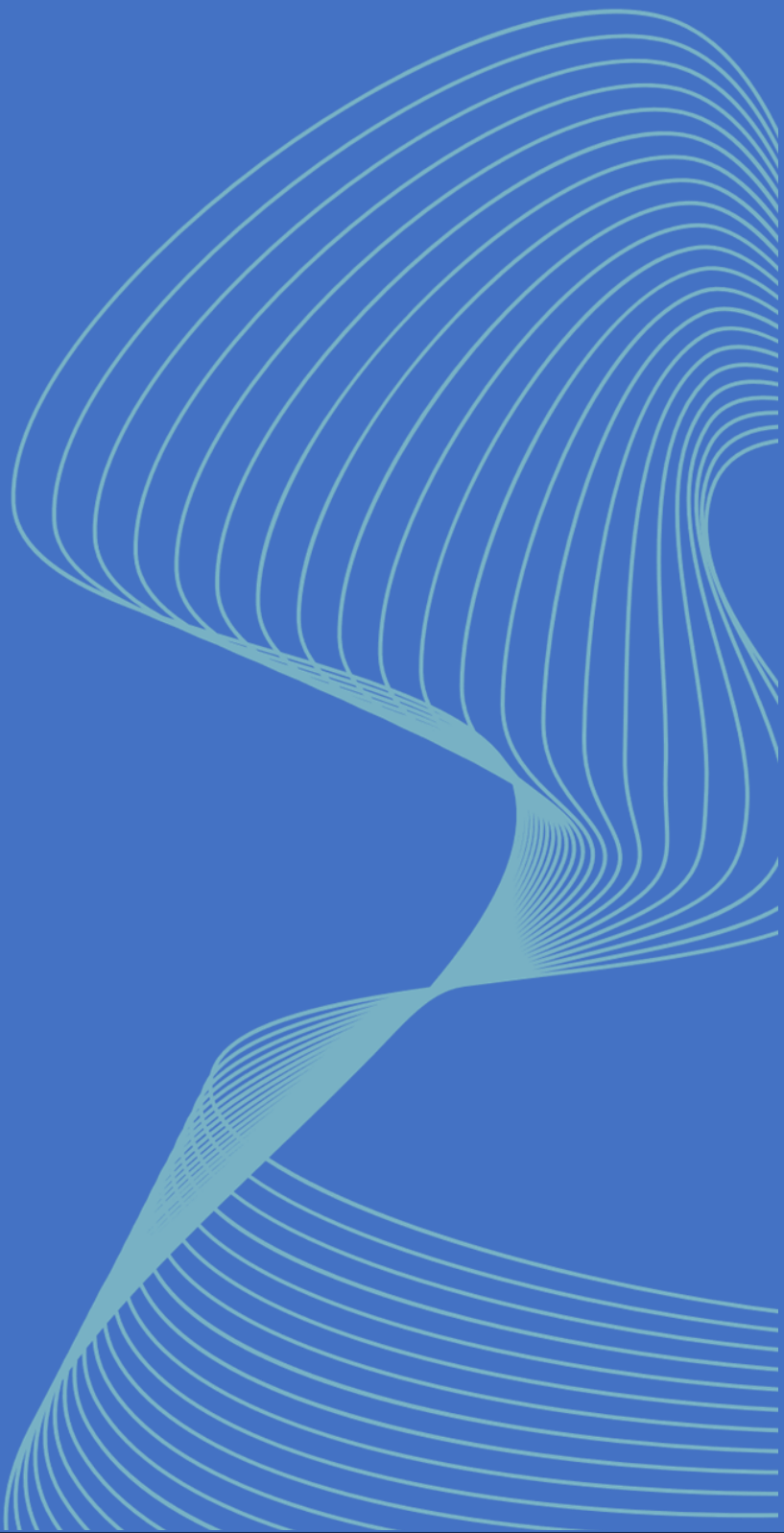
NSCC

APRIL 2024

DBAS 3035 PROJECT

Credit Card Fraud Dataset



Prepared by

OBIEME, Esther

SANTIAGO, Ma. Angelica

Contents

Introduction

Our team's final project in the Information Systems Design course DBAS3035 delves into the analysis of a dataset focusing on credit card fraud, sourced from Kaggle. Comprising 555,719 records, each with 22 attributes covering categorical and numerical data, this dataset offers a rich landscape for exploration. Our central aim is to uncover patterns of fraud, particularly concerning victim demographics, and ascertain potential correlations with factors like location, date, and time. Additionally, we seek to quantify the prevalence of fraud across different states and cities.

In the execution of our project, Python played a pivotal role in both data extraction and transformation. We utilized PostgreSQL as our database management system, with SQL statements seamlessly executed in Python through the SQL Alchemy library. To visualize our findings, we employed Python's Matplotlib for graphical analysis and Folium for interactive mapping. Throughout the process, Pandas and NumPy facilitated efficient data manipulation, ensuring a comprehensive and cohesive analysis.

Entity Relationship Diagram (ERD)

The ERD diagram presented below is based on the credit card fraud dataset. The relationships between tables reflect the associations found within the actual dataset obtained.

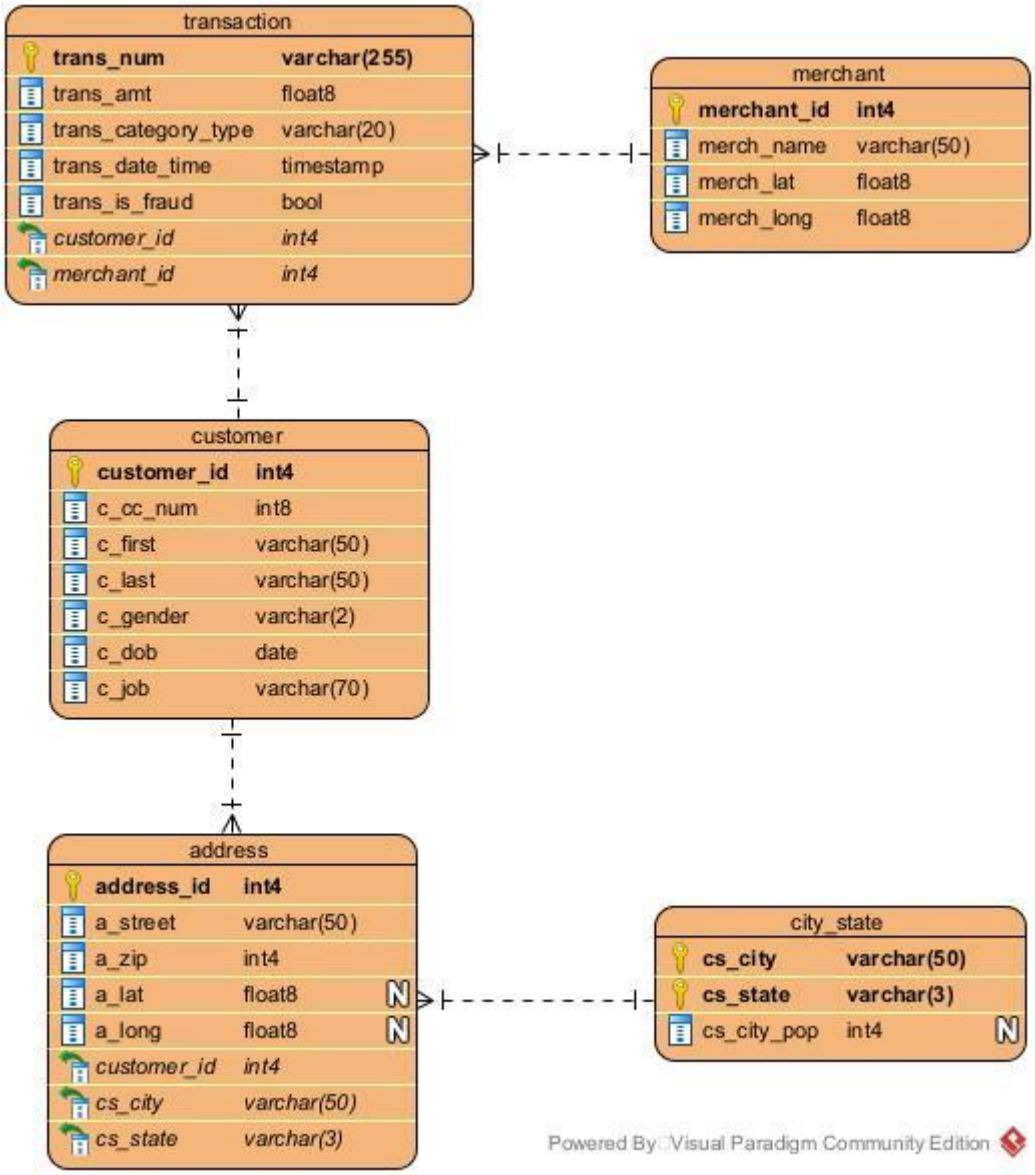


Figure 1 Entity Relationship Diagram

Data Dictionary

Entity	Description	Column	Description	Key Type	Data Type	No. of Distinct Values	Sample Values
customer	This table stores personal information about customers who are conducting transactions.	customer_id	A unique identifier for each customer who made a transaction.	Primary	int4	924	59867230, 16096630, 71467364, 4657782, 29521722
		c_cc_num	Customer credit card number used in the transaction.		int8	904	630452000000, 4561550000000, 30242900000000, 4292900000000000000, 4981130000000
		c_first	Cardholder's first name.		varchar(50)	341	Scott, Theresa, Joshua, John, Barbara
		c_last	Cardholder's last name.		varchar(50)	471	Wood, Ramirez, Henry, Johnson, Murphy
		c_gender	Cardholder's gender.		varchar(2)	2	F, F, M, F, F
		c_dob	Cardholder's date of birth.		date	910	1996-04-10, 1982-04-19, 1987-09-26, 1960-02-01, 1976-11-21
		c_job	Cardholder's job title.		varchar(70)	478	Engineer, production, Water engineer, Exhibition designer, Historic buildings inspector/conservation officer, Systems analyst
address	The Address table holds information on the location details of each customer.	address_id	A unique identifier assigned to each address entry.	Primary	int4	924	41696290, 6675694, 45188408, 18953142, 86239672
		a_street	Cardholder's street address.		varchar(50)	924	27203 Darrell Landing, 319 Wendy Fort Suite 179, 977 Rita Grove Apt. 000, 5930 Rebecca Bridge Apt. 444, 25036 Contreras Turnpike Suite 270
		a_zip	Cardholder's zip code.		int4	912	82901, 34668, 21524, 47434, 22151
		a_lat	Latitude of cardholder's location.		float8	910	-106.0654, -74.0163, -90.3359, -84.1481, -89.0933
		a_long	Longitude of cardholder's location.		float8	910	5161, 471, 3807, 198, 59744
		customer_id	A foreign key that references the customer_id in the customer table. It establishes a relationship between an address and a customer, indicating which address belongs to which customer.	Foreign	int4		
		cs_city	This field stores the name of the city for the address. It forms part of the composite foreign key in conjunction with cs_state, referencing the city_state table.	Foreign	varchar(50)		
		cs_state	The state of the address. Along with cs_city, part of the composite foreign key links the address to the corresponding city and state entry in the city_state table.	Foreign	varchar(3)		
city_state	This table provides details about cities and states, including a unique	cs_city	This field holds the name of the city, that when combined with cs_state, uniquely identifies a city within a particular state. It is part of the composite primary key for the city_state table.	Primary (Composite)	varchar(50)	849	Allentown, Matawan, Karnack, Lake Oswego, Naples

Entity	Description	Column	Description	Key Type	Data Type	No. of Distinct Values	Sample Values
	composite key combining city and state names.	cs_state	This column contains the name of the state in which the city is located. It forms the other half of the composite primary key, allowing cities with the same name but in different states to be uniquely identified.		varchar(3)	50	AL, AR, MO, MI, AL
		cs_city_pop	The population of the city.		int4	835	5161, 471, 3807, 198, 59744
merchant	The Merchant table contains information about the merchants where transactions are made.	merchant_id	A unique identifier for each merchant.	Primary	int4	693	603, 399, 246, 284, 89
		merch_name	The name of the merchant involved in the transaction.		varchar(50)	693	Greenholt, Jacobi and Gleason, Metz-Boehm, Lang, Towne and Schuppe, Bednar PLC, Baumbach, Hodkiewicz and Walsh
		merch_lat	Latitude coordinate for the merchant's location.		float8	546,490	38.653065, 38.637922, 43.57037, 40.843771, 43.949124
		merch_long	Longitude coordinate for the merchant's location.		float8	551,770	-74.872315, -80.718791, -78.808638, -116.718714, -82.195389
transaction	This table records the details of each transaction the customer made.	trans_num	A unique identifier for each transaction.	Primary	varchar(255)	555,719	4f3e05921436e3abe437ec23e06bb6bb, 2443733fab9756df150b3ab1454ce51e, 37d15a4ca75fd5617ecc57e70072f954, 9a8763fb3f2c59e1fb95c4a9682fa754, bb2cce5d370e918b8abf308b96ea163a
		trans_amt	The amount of money involved in the transaction.		float8	37,256	60.98, 83.5, 44.27, 54.73, 4.15
		trans_category_type	A categorical description of the transaction type (e.g., personal, childcare).		varchar(20)	14	entertainment, personal care, home, grocery pos, shopping net
		trans_date_time	The timestamp of when the transaction occurred (date and time).		timestamp	544,760	2013-07-10 23:25:29, 2013-07-28 05:29:34, 2013-10-06 15:23:39, 2013-10-22 18:37:58, 2013-07-24 19:59:45
		trans_is_fraud	A boolean flag indicating whether the transaction is fraudulent (1 = fraud, 0 = legitimate).		bool	2	0, 0, 0, 0, 1
		customer_id	A foreign key that links to the customer_id in the customer table. It identifies which customer conducted the transaction.	Foreign	int4		
		merchant_id	A foreign key that references the merchant_id in the merchant table. It establishes which merchant is involved in the transaction.	Foreign	int4		

(Kelue, 2024)

SQL Queries

Query 1 - Exploring Transaction Details (Multi-Table Join)

This SQL query joins information from the 'customer', 'address', 'transaction', and 'merchant' tables to create a unified view. The resulting dataset includes details such as customer ID, occupation, gender, age calculated from the date of birth and year the transaction was made, city, state, zip code, transaction date and time, merchant name, transaction category type, transaction amount, and a flag indicating whether the transaction is flagged as fraudulent. The dataset is ordered chronologically by transaction date and time, providing a structured view of credit card activities over time.

```
SELECT
    c.customer_id,
    c.c_job,
    c.c_gender,
    EXTRACT(YEAR FROM AGE('2020-12-31', c_dob)) AS age,
    a.cs_city,
    a.cs_state,
    a.a_zip,
    t.trans_date_time,
    m.merch_name,
    t.trans_category_type,
    t.trans_amt,
    t.trans_is_fraud
FROM
    customer c
    JOIN address a ON c.customer_id = a.customer_id
    JOIN transaction t ON c.customer_id = t.customer_id
    JOIN merchant m ON t.merchant_id = m.merchant_id
ORDER BY
    t.trans_date_time;
```

Output:

Sorted by trans_date_time

	customer_id	c_job	c_gender	age	cs_city	cs_state	a_zip	trans_date_time	merch_name	trans_category_type	trans_amt	trans_is_fraud
0	74078269	Mechanical engineer	M	52.0	Columbia	SC	29209	2020-06-21 12:14:00	Kirlin and Sons	personal care	2.86	False
1	18702918	Sales professional, IT	F	30.0	Altonah	UT	84002	2020-06-21 12:14:00	Sporer-Keebler	personal care	29.84	False
2	95127892	Librarian, public	F	50.0	Bellmore	NY	11710	2020-06-21 12:14:00	Swaniawski, Nitzsche and Welch	health fitness	41.28	False
3	59734903	Set designer	M	33.0	Titusville	FL	32780	2020-06-21 12:15:00	Haley Group	misc pos	60.05	False
4	81017923	Furniture designer	M	65.0	Falmouth	MI	49632	2020-06-21 12:15:00	Johnston-Casper	travel	3.19	False
...	...	...	...	...	...	...	...	...	...	...	...	...
555714	91660892	Town planner	M	54.0	Luray	MO	63453	2020-12-31 23:59:00	Reilly and Sons	health fitness	43.77	False
555715	51585285	Futures trader	M	21.0	Lake Jackson	TX	77566	2020-12-31 23:59:00	Hoppe-Parisian	kids pets	111.84	False
555716	99872181	Musician	F	39.0	Burbank	WA	99323	2020-12-31 23:59:00	Rau-Robel	kids pets	86.88	False
555717	28750882	Cartographer	M	55.0	Mesa	ID	83643	2020-12-31 23:59:00	Breitenberg LLC	travel	7.99	False
555718	58471486	Media buyer	M	27.0	Edmond	OK	73034	2020-12-31 23:59:00	Dare-Marvin	entertainment	38.13	False

555719 rows × 12 columns

Sorted by trans_amt

	customer_id	c_job	c_gender	age	cs_city	cs_state	a_zip	trans_date_time	merch_name	trans_category_type	trans_amt	trans_is_fraud
0	48909282	Metallurgist	M	50.0	Falconer	NY	14733	2020-06-30 01:52:00	Nader-Heller	misc net	1.00	False
1	1627715	Farm manager	F	28.0	Amanda	OH	43102	2020-06-22 15:57:00	Langworth, Boehm and Gulgowski	shopping net	1.00	False
2	25226820	Museum/gallery conservator	F	32.0	Clifton	SC	29324	2020-07-07 15:52:00	Baumbach, Hodkiewicz and Walsh	shopping pos	1.00	False
3	71467364	Exhibitions officer, museum/gallery	F	25.0	Elberta	MI	49628	2020-06-22 19:49:00	Kuhic, Bins and Pfeffer	shopping net	1.00	False
4	55870867	Engineer, manufacturing	M	66.0	Gainesville	TX	76240	2020-08-06 02:21:00	Huel Ltd	misc net	1.00	False
...	...	...	...	...	...	...	...	...	...	...	...	...
555714	83765334	Surgeon	M	21.0	Conway	NH	3818	2020-12-16 21:16:00	Boyer-Haley	travel	16,339.26	False
555715	89989635	Health physicist	F	49.0	Newhall	CA	91321	2020-09-21 12:02:00	Johnston-Casper	travel	16,837.08	False
555716	64061725	Hospital doctor	M	65.0	Pembroke	NC	28372	2020-11-27 14:54:00	Kovacek Ltd	travel	19,364.91	False
555717	49735544	Higher education careers adviser	F	26.0	Jay	FL	32565	2020-12-22 21:30:00	Corwin-Romaguera	travel	21,437.71	False
555718	163011	Engineer, control and instrumentation	F	40.0	Bridgeport	NJ	8014	2020-12-27 15:58:00	Kozey-McDermott	travel	22,768.11	False

555719 rows × 12 columns

Analysis:

Upon sorting the dataset by trans_date_time (transaction date and time), it becomes evident that the recorded transactions span from June 21, 2020, to December 31, 2020. Further examination by sorting the data based on trans_amount (transaction amount) reveals interesting insights. The highest transaction amount observed was $22,768.11, attributed to travel expenses, while the lowest transaction amount recorded was $1.00, designated for miscellaneous purposes.

Query 2 - Creating a View - Identifying Frequent Customers

A view is created to rank customers by their transaction frequency. This view consolidates customer IDs and transaction counts. Subsequent queries to this view identify the most frequent customers, offering crucial insights for pinpointing valuable customers or those whose transaction frequency may warrant additional scrutiny.

```
CREATE VIEW top_customers
    AS
        SELECT
            c.customer_id,
            c.c_gender,
            EXTRACT(YEAR FROM AGE('2020-12-31', c_dob)) AS age,
            c.c_job,
            a.a_zip,
            a.cs_city,
            a.cs_state,
            COUNT(t.trans_num) AS num_of_transaction,
            SUM(t.trans_amt) AS total_amount,
            COUNT(CASE WHEN trans_is_fraud = True THEN 1 ELSE NULL END ) AS
no_of_fraud
        FROM
            customer c
            JOIN transaction t ON c.customer_id = t.customer_id
            JOIN address a ON c.customer_id = a.customer_id
        GROUP BY c.customer_id, c.c_job, c.c_gender,a.a_zip, a.cs_city, a.cs_state
        LIMIT 100;
```

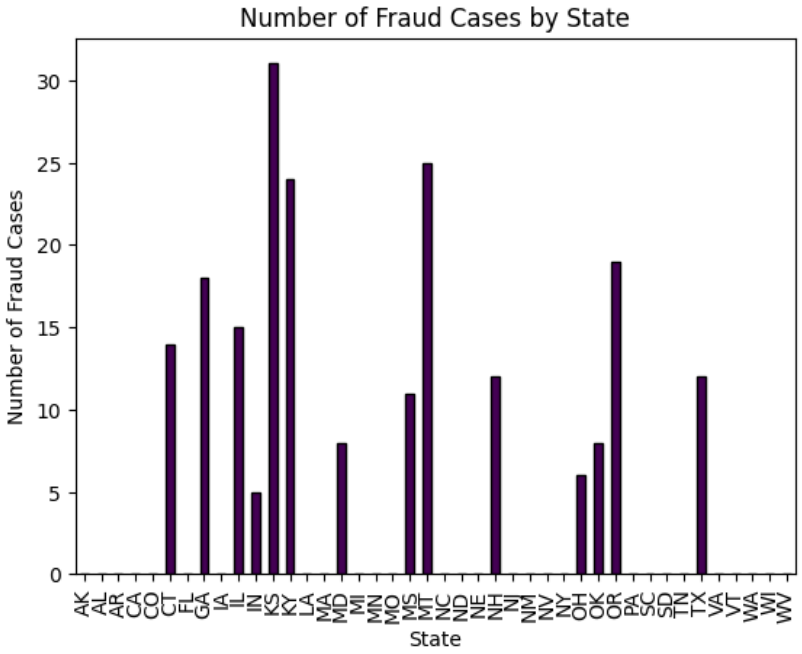
Output:

	customer_id	c_gender	age	c_job	a_zip	cs_city	cs_state	num_of_transaction	total_amount	no_of_fraud
0	2279629	M	24.0	Pensions consultant	40077	Westport	KY	1428	98991.93	0
1	916035	F	48.0	Designer, exhibition/display	92585	Sun City	CA	1352	117944.10	0
2	6779390	F	45.0	Firefighter	46254	Indianapolis	IN	1339	114631.63	0
3	9794581	F	32.0	Tax inspector	65072	Rocky Mount	MO	1334	74756.38	0
4	8548529	F	35.0	Regulatory affairs officer	21872	Whaleyville	MD	1301	66376.27	0
...	...	...	...	...	...	...	...	...	...	...
95	10359342	F	68.0	Magazine journalist	62266	New Memphis	IL	211	10807.69	0
96	4294505	F	84.0	Biochemist, clinical	25832	Daniels	WV	209	12838.53	0
97	7150172	M	33.0	Comptroller	92101	San Diego	CA	204	15149.16	0
98	10180169	M	81.0	Estate manager/land agent	50527	Curlew	IA	195	11102.84	0
99	3603551	M	18.0	Chemical engineer	66958	Morrowville	KS	12	7993.74	12

100 rows × 10 columns

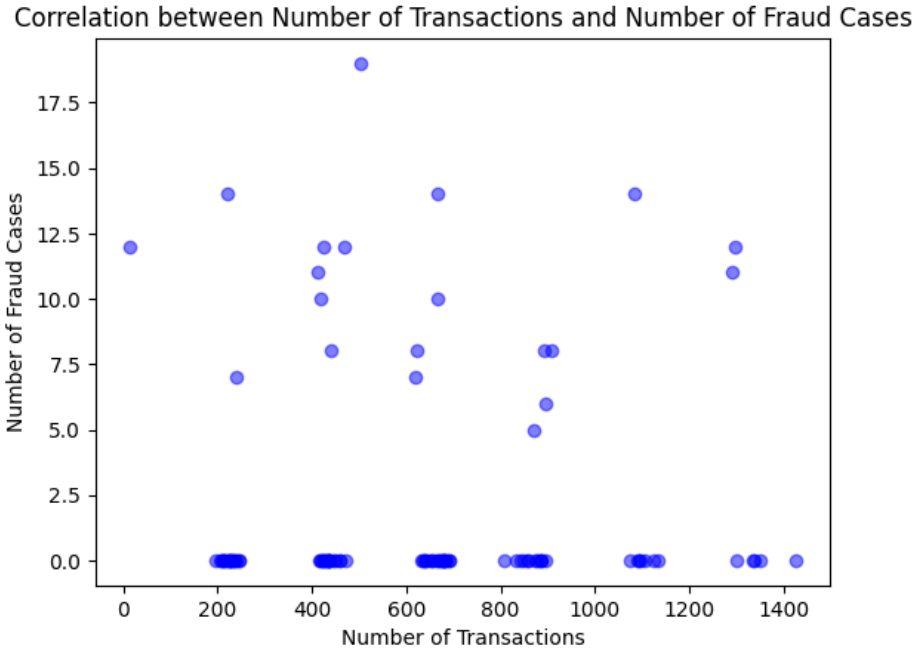
The output reveals that the customer with the highest activity has engaged in 1,428 transactions, totaling $98,991.93, while the customer with the least activity has 12 transactions, amounting to $7,993.74.

From the created view, we can generate the following grahps to draw meaningful insights:



The bar graph indicates that Kansas (KS) records the highest number of fraudulent transactions, closely followed by Montana (MT), highlighting potential hotspots for fraud.

On the other hand, the scatter plot reveals a pattern of fraud cases against the volume of transactions. It is evident that there is a clustering of data points near the bottom, suggesting that a higher number of transactions does not necessarily correlate with an increase in fraud cases. This underscores the complexity of fraud detection, where transaction volume alone is not a reliable predictor of fraudulent activity.



Query 3: Common Table Expression (CTE) - Transaction Patterns Analyzed

Using a CTE, the median transaction amount is calculated, and individual transactions are compared to identify those that significantly deviate from this average. Transactions substantially above or below the average may indicate errors or potentially fraudulent activity.

```
WITH median_transaction AS (
    SELECT
        c.customer_id,
        PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY t.trans_amt) AS
median_trans_amt
    FROM customer c
    JOIN transaction t ON t.customer_id = c.customer_id
    GROUP BY c.customer_id
)
SELECT
    c.customer_id,
    c.c_gender,
    c.c_job,
    m.merch_name,
    t.trans_category_type,
    t.trans_date_time,
    t.trans_amt,
    med.median_trans_amt,
    ((t.trans_amt - med.median_trans_amt)/ med.median_trans_amt) * 100 AS
percentage_diff,
    t.trans_is_fraud
FROM customer c
    JOIN transaction t ON c.customer_id = t.customer_id
    JOIN merchant m ON t.merchant_id = m.merchant_id
    JOIN median_transaction med ON c.customer_id = med.customer_id
ORDER BY percentage_diff DESC;
```

Output:

	customer_id	c_gender	c_job	merch_name	trans_category_type	trans_date_time	trans_amt	median_trans_amt	percentage_diff	trans_is_fraud
0	83765334	M	Surgeon	Boyer-Haley	travel	2020-12-16 21:16:00	16339.26	22.960	71064.024390	False
1	163011	F	Engineer, control and instrumentation	Kozey-McDermott	travel	2020-12-27 15:58:00	22768.11	35.460	64107.868020	False
2	49735544	F	Higher education careers adviser	Corwin-Romaguera	travel	2020-12-22 21:30:00	21437.71	50.430	42409.835415	False
3	50479522	F	Counsellor	Hagenes, Hermann and Stroman	travel	2020-07-03 18:13:00	13149.15	32.380	40508.863496	False
4	9488913	M	Biomedical engineer	Schroeder, Wolff and Hermiston	travel	2020-10-19 18:11:00	12969.90	32.895	39328.180575	False
...	...	...	...	...	...	...	...	...	...	...
555714	19654975	M	Armed forces training and education officer	Luettgen PLC	gas transport	2020-07-30 00:52:00	7.84	774.140	-98.987263	True
555715	87292974	M	Commissioning editor	Huels-Nolan	gas transport	2020-12-02 00:07:00	9.16	955.500	-99.041340	True
555716	69296734	M	Historic buildings inspector/conservation officer	Raynor, Feest and Miller	gas transport	2020-09-01 09:19:00	7.99	837.825	-99.046340	True
555717	19654975	M	Armed forces training and education officer	Prohaska-Murray	gas transport	2020-07-31 03:43:00	7.00	774.140	-99.095771	True
555718	34461572	M	Database administrator	Mraz-Herzog	gas transport	2020-12-22 03:56:00	6.60	867.050	-99.238798	True

555719 rows × 10 columns

To simplify the interpretation of percentage differences, case statements can be used to categorize them into easily understandable groups (PostgreSQL Tutorial, 2011). The computation of high and low-risk thresholds involves determining the standard deviation of the transaction amounts. Specifically, a transaction is considered low risk if it falls below the median transaction amount minus the standard deviation, and high risk if it exceeds the median transaction amount plus the standard deviation.

```
WITH median_transaction AS (
            SELECT
                c.customer_id,
                PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY t.trans_amt) AS
median_trans_amt
            FROM customer c
                JOIN transaction t ON t.customer_id = c.customer_id
            GROUP BY c.customer_id
        )
```

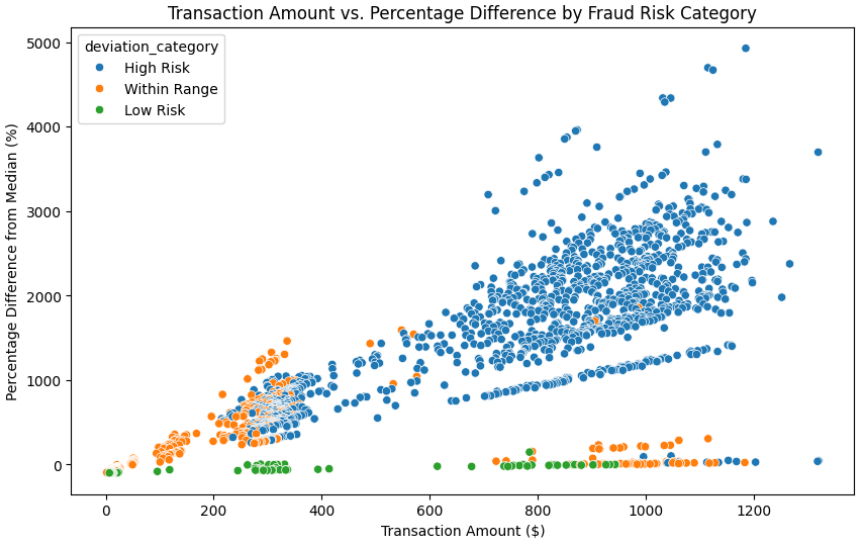
```
        SELECT
            c.customer_id,
            c.c_gender,
            c.c_job,
            m.merch_name,
            t.trans_num,
            t.trans_category_type,
            t.trans_date_time,
            t.trans_amt,
            med.median_trans_amt,
            ((t.trans_amt - med.median_trans_amt)/ med.median_trans_amt) * 100 AS
percentage_diffs,
            CASE
                WHEN ((t.trans_amt - med.median_trans_amt)/ med.median_trans_amt) *
100 < -356 THEN 'Low Risk'
                WHEN ((t.trans_amt - med.median_trans_amt)/ med.median_trans_amt) *
100 > 492 THEN 'High Risk'
                ELSE 'Within Range'
                END AS percentage_diff,
            t.trans_is_fraud
        FROM customer c
            JOIN transaction t ON c.customer_id = t.customer_id
            JOIN merchant m ON t.merchant_id = m.merchant_id
            JOIN median_transaction med ON c.customer_id = med.customer_id
        ORDER BY percentage_diffs DESC;
```

Output:

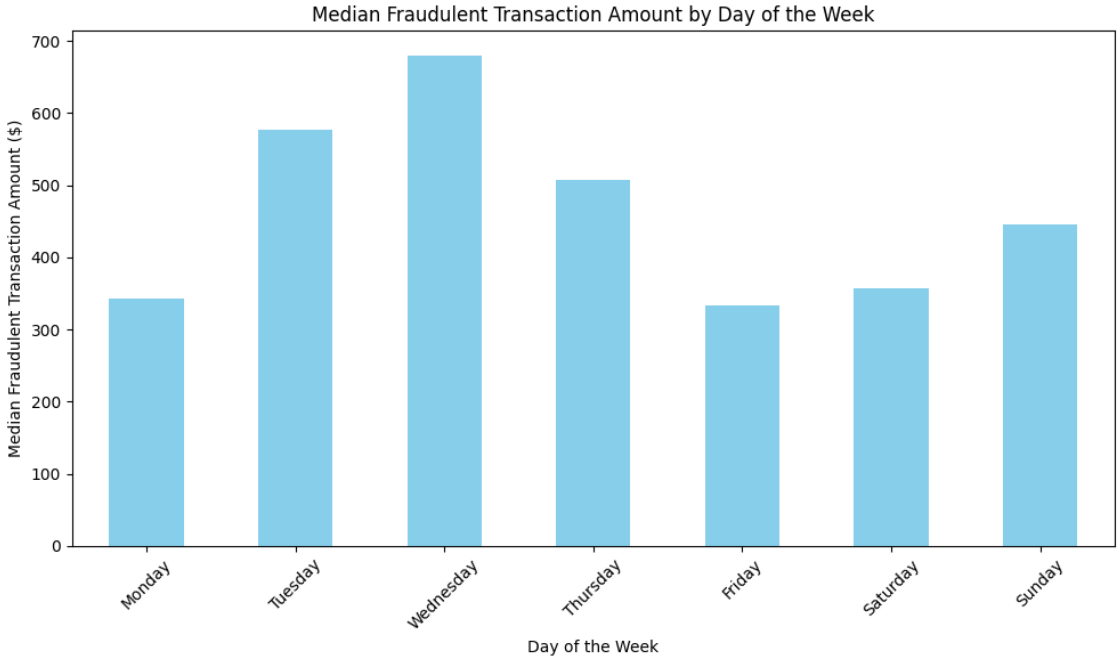
	customer_id	c_gender	c_job	merch_name	trans_num	trans_category_type	trans_date_time	trans_amt	median_trans_amt	percentage_diffs	percentage_diff	trans_is_fraud
0	83765334	M	Surgeon	Boyer-Haley	787b3a6dae2fad86d24e89ec9e13482b	travel	2020-12-16 21:16:00	$16,339.26	$22.96	71064.024390	High Risk	False
1	163011	F	Engineer, control and instrumentation	Kozey-McDermott	93dd11ae955df7ea72df575dc091ac82	travel	2020-12-27 15:58:00	$22,768.11	$35.46	64107.868020	High Risk	False
2	49735544	F	Higher education careers adviser	Corwin-Romaguera	932aef04df2c7a1f4d53127eb1bd71f2	travel	2020-12-22 21:30:00	$21,437.71	$50.43	42409.835415	High Risk	False
3	50479522	F	Counsellor	Hagenes, Hermann and Stroman	3a9d71cce25e35776395ce67aefff16f	travel	2020-07-03 18:13:00	$13,149.15	$32.38	40508.863496	High Risk	False
4	9488913	M	Biomedical engineer	Schroeder, Wolff and Hermiston	3433e46c700a366392f7e0dc96505b13	travel	2020-10-19 18:11:00	$12,969.90	$32.89	39328.180575	High Risk	False
...	...	...	...	...	...	...	...	...	...	...	...	...
555714	19654975	M	Armed forces training and education officer	Luettgen PLC	7a4349b621986d6d52a1495c71ab55d6	gas transport	2020-07-30 00:52:00	$7.84	$774.14	-98.987263	Within Range	True
555715	87292974	M	Commissioning editor	Huels-Nolan	904224770db9a338bc82fb05ff5b69cb	gas transport	2020-12-02 00:07:00	$9.16	$955.50	-99.041340	Within Range	True
555716	69296734	M	Historic buildings inspector/conservation officer	Raynor, Feest and Miller	312724184cf488a1f1aac9fe0bc97a43	gas transport	2020-09-01 09:19:00	$7.99	$837.83	-99.046340	Within Range	True
555717	19654975	M	Armed forces training and education officer	Prohaska-Murray	ac173980a20036534db37c59b9626954	gas transport	2020-07-31 03:43:00	$7.00	$774.14	-99.095771	Within Range	True
555718	34461572	M	Database administrator	Mraz-Herzog	92364a31b4d19e614bf8b6ef31b2255a	gas transport	2020-12-22 03:56:00	$6.60	$867.05	-99.238798	Within Range	True

555719 rows × 12 columns

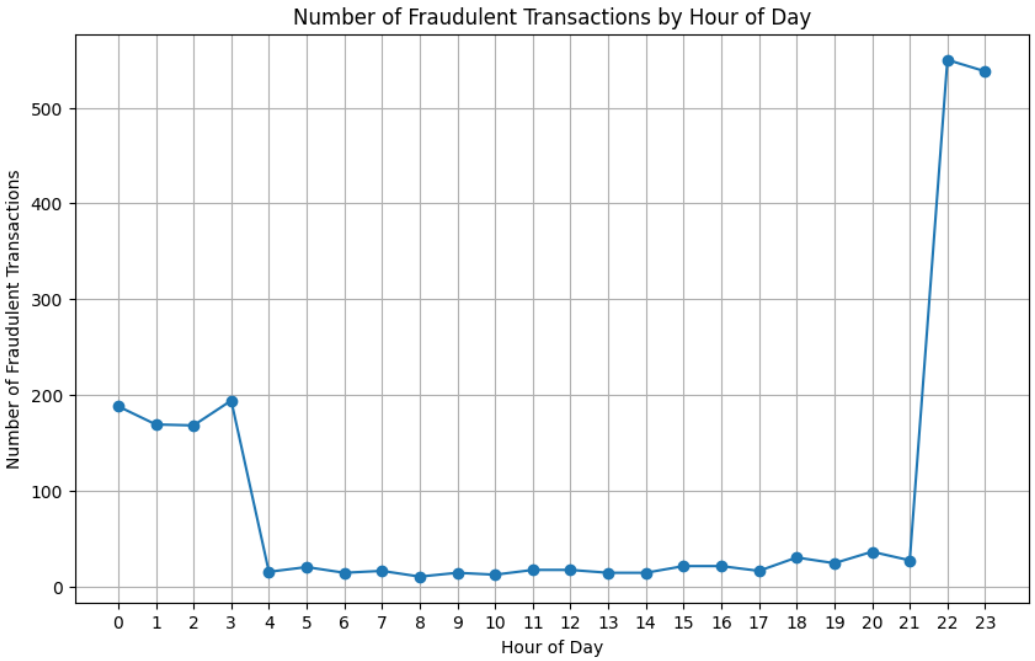
Visualizations generated from the SQL queries have provided a concentrated examination of fraudulent activities within the dataset. It is revealed by the scatterplot that many transactions, identified as high risk, are flagged as fraudulent.

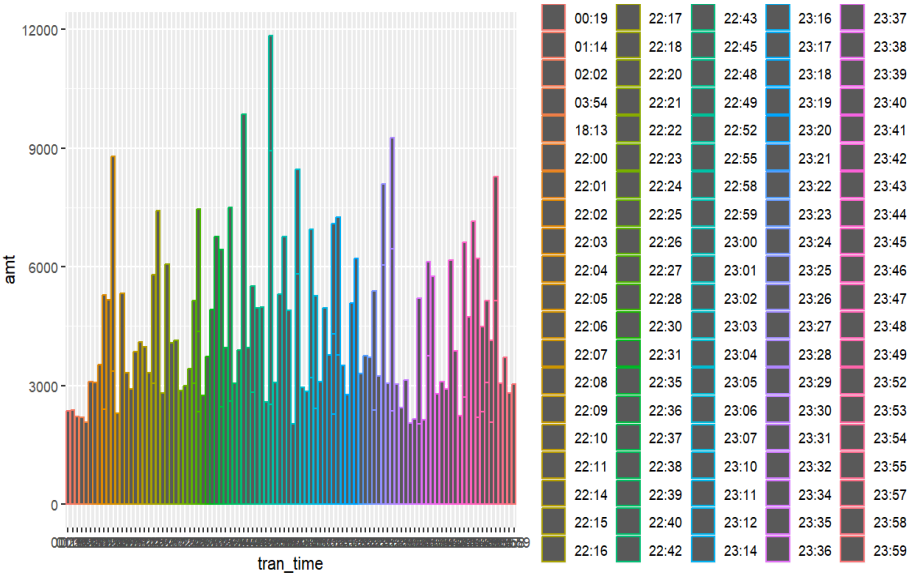


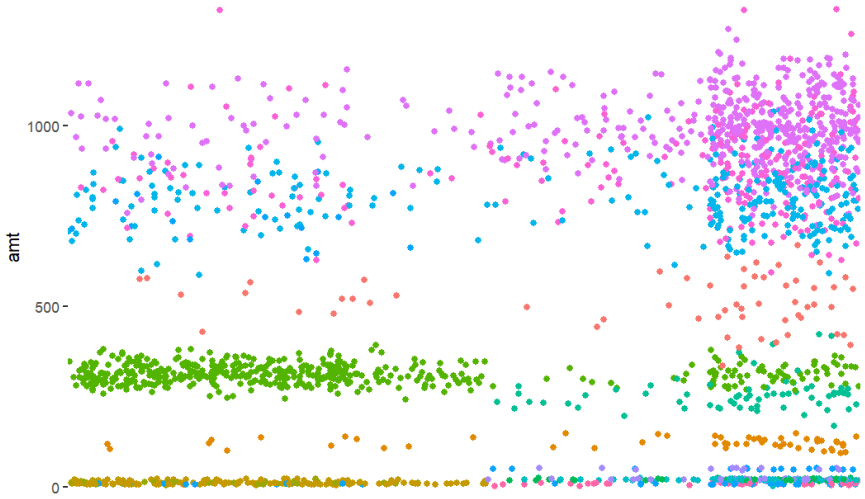
However, it is important to note that a significant number of transactions within the normal spending range are also marked as fraud. This poses a challenge for credit card companies, as fraudulent activities can often mimic typical spending patterns, making the transaction amount alone an unreliable fraud indicator.



The bar graph plotting fraudulent activities by the day of the week reveals that Wednesdays have the highest incidence of fraud. Complementing this, the use of a line chart indicates that the frequency of fraudulent activities peaks at around 10 pm. These observations emphasize the need for more sophisticated analytical approaches to effectively detect and address the complexities of fraud.


Number of Fraudulent Transactions by Hour of Day
Number of Fraudulent Transactions
0
100
200
300
400
500
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour of Day


amt
0
3000
6000
9000
12000
tran_time
00:19 01:14 02:02 03:54 18:13 22:00 22:01 22:02 22:03 22:04 22:05 22:06 22:07 22:08 22:09 22:10 22:11 22:14 22:15 22:16
22:17 22:18 22:20 22:21 22:22 22:23 22:24 22:25 22:26 22:27 22:28 22:30 22:31 22:35 22:36 22:37 22:38 22:39 22:40 22:42
22:43 22:45 22:48 22:49 22:52 22:55 22:58 22:59 23:00 23:01 23:02 23:03 23:04 23:05 23:06 23:07 23:10 23:11 23:12 23:14
23:16 23:17 23:18 23:19 23:20 23:21 23:22 23:23 23:24 23:25 23:26 23:27 23:28 23:29 23:30 23:31 23:32 23:34 23:35 23:36
23:37 23:38 23:39 23:40 23:41 23:42 23:43 23:44 23:45 23:46 23:47 23:48 23:49 23:52 23:53 23:54 23:55 23:57 23:58 23:59


amt
0
500
1000

Query 4: Aggregation - Profiling High-Risk Transactions

This query categorizes fraudulent transactions by type and merchant, extracts the month from the date for temporal analysis, and uses aggregation (PostgreSQL, 2024) to calculate the total fraud cases and involvement by gender. It also calculates the median transaction amount using percentile continuation function. Results are filtered to include only fraud cases and grouped by type, month, and merchant details, and ordered by the number of fraud cases to prioritize investigation of high-risk categories.

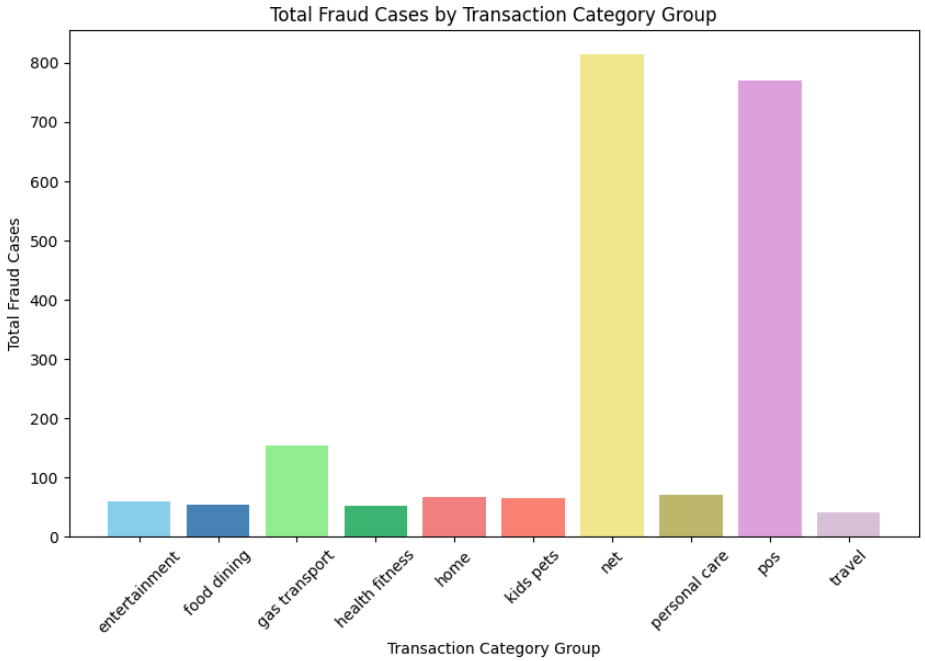
```
SELECT
    tr.trans_category_type,
    EXTRACT(MONTH FROM tr.trans_date_time) AS transaction_month,
    COUNT(CASE WHEN tr.trans_is_fraud = True THEN 1 END) AS total_fraud_cases,
    COUNT(CASE WHEN tr.c_gender = 'M' THEN 1 END) AS total_male_involved,
    COUNT(CASE WHEN tr.c_gender = 'F' THEN 1 END) AS total_female_involved,
    PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY tr.trans_amt) AS med_fraud_amt,
    tr.merch_name,
    m.merch_lat,
    m.merch_long
FROM
    trans_cust_demo tr
    JOIN transaction t ON tr.trans_num = t.trans_num
    JOIN merchant m ON t.merchant_id = m.merchant_id
WHERE
    tr.trans_is_fraud = True
GROUP BY
    tr.trans_category_type,
    transaction_month,
    tr.merch_name,
    m.merch_lat,
    m.merch_long
HAVING
    COUNT(CASE WHEN tr.trans_is_fraud = True THEN 1 END) >= 1
ORDER BY
    total_fraud_cases DESC;
```

Output:

	trans_category_type	transaction_month	total_fraud_cases	total_male_involved	total_female_involved	med_fraud_amt	merch_name	merch_lat	merch_long
0	grocery pos	8.0	6	1	5	330.490	Cole PLC	40.325204	-79.237092
1	shopping net	7.0	6	3	3	963.375	Jast Ltd	37.963992	-82.926133
2	shopping net	10.0	5	1	4	977.080	Mosciski, Ziemann and Farrell	38.709403	-104.012066
3	grocery pos	10.0	5	2	3	317.370	Padberg-Welch	44.732656	-109.444699
4	grocery pos	10.0	5	3	2	291.240	Kiehn Inc	32.988015	-80.955716
...	...	...	...	...	...	...	...	...	...
1440	kids pets	10.0	1	0	1	20.610	Nolan-Williamson	34.849935	-86.922332
1441	kids pets	10.0	1	0	1	21.230	Roberts, Daniel and Macejkovic	31.212812	-85.081825
1442	kids pets	10.0	1	0	1	20.730	Streich, Rolfson and Wilderman	39.296040	-76.368333
1443	kids pets	10.0	1	0	1	21.090	Weimann-Lockman	41.881426	-83.316455
1444	kids pets	11.0	1	1	0	21.550	Berge-Hills	38.653037	-84.419288

1445 rows × 9 columns

Based on the SQL statement outputs, visualizations have been created to delve deeper into the transaction types and their associated fraud cases.



The bar graph illustrates that Point of Sale (POS) and internet (net) transactions harbor the highest number of fraud instances. This discovery calls for heightened scrutiny of such transactions.

Furthermore, by plotting the latitude and longitude coordinates of merchants on a map, it becomes evident that POS transactions (marked in red) and net transactions (marked in blue), along with other transactions (in green), are geographically widespread. It's crucial to cross-reference these locations with local crime rates, particularly theft, as such environmental factors could potentially influence fraudulent activities:

Through these analytical steps, the complexities of transaction fraud are better understood, reinforcing the need for comprehensive and dynamic fraud detection strategies.

Query 5: Sub-query – Fraud Victims' demographics

This query utilizes a Common Table Expression (CTE) named cust_demo to first assemble a targeted dataset of customer demographics, including age, job, gender, city, and state, derived

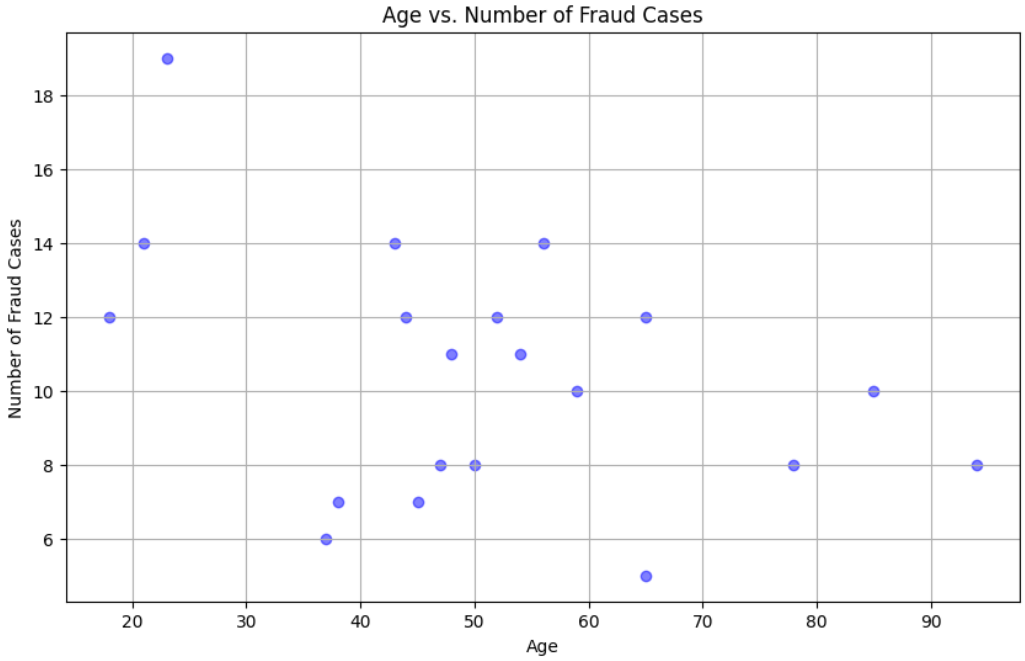
from the customer and address tables. It calculates the customer's age as of the end of the year 2020 for consistent age determination across the dataset.

```
WITH cust_demo AS (
    SELECT
        c.customer_id,
        EXTRACT(YEAR FROM AGE('2020-12-31', c_dob)) AS age,
        c.c_job,
        c.c_gender,
        a.cs_city,
        a.cs_state
    FROM
        customer c
        JOIN address a ON c.customer_id = a.customer_id
    GROUP BY
        c.customer_id, a.cs_city, a.cs_state, c.c_gender, c.c_job, age
)   SELECT
        cd.*,
        SUM(t.trans_amt) AS total_trans_amt,
        COUNT(*) AS no_of_fraud
    FROM
        cust_demo cd
        JOIN transaction t ON cd.customer_id = t.customer_id
        JOIN merchant m ON t.merchant_id = m.merchant_id
    WHERE
        t.trans_is_fraud = True
        AND cd.customer_id IN (SELECT customer_id FROM top_customers)
    GROUP BY
        cd.customer_id, cd.age, cd.c_job, cd.cs_state, cd.cs_city, cd.c_gender
    ORDER BY
        no_of_fraud DESC;
```

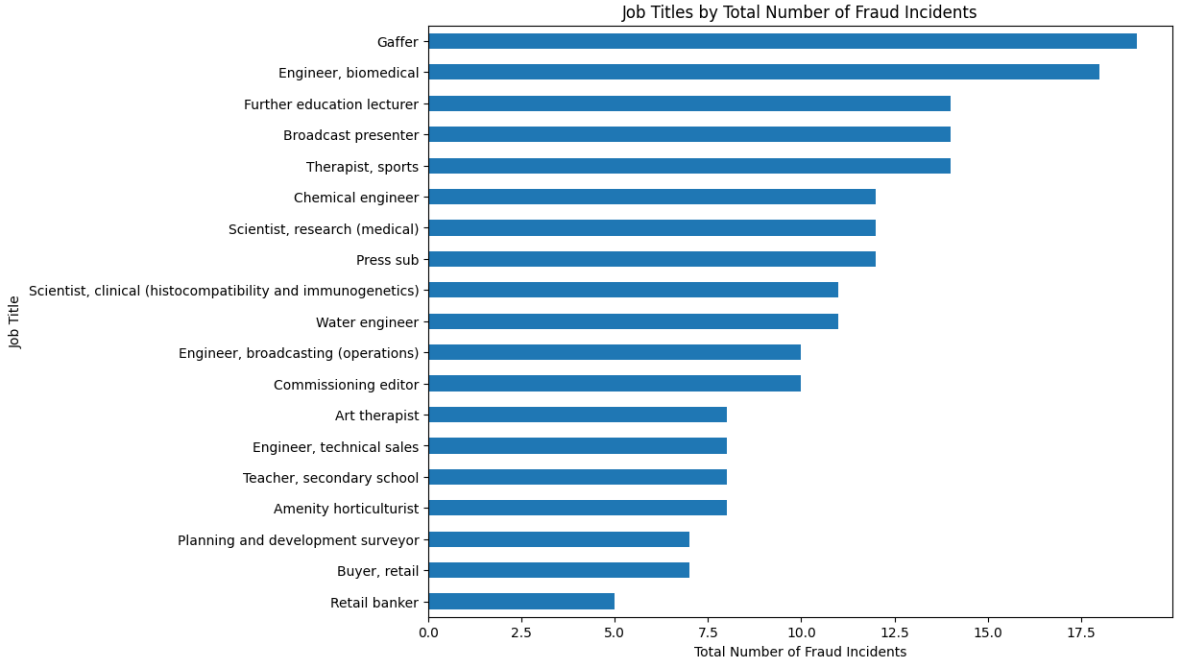
Output:

	customer_id	age	c_job	c_gender	cs_city	cs_state	total_trans_amt	no_of_fraud
0	5936811	23.0	Gaffer	F	Burrton	KS	12002.06	19
1	7303638	56.0	Further education lecturer	M	Turner	MT	6863.58	14
2	9533425	21.0	Therapist, sports	F	Smiths Grove	KY	5839.30	14
3	4279521	43.0	Broadcast presenter	F	Preston	CT	5782.20	14
4	3603551	18.0	Chemical engineer	M	Morrowville	KS	7993.74	12
5	10580441	52.0	Engineer, biomedical	F	Stayton	OR	8195.87	12
6	1529474	44.0	Press sub	F	Thrall	TX	5685.68	12
7	5077091	65.0	Scientist, research (medical)	M	Belmont	NH	5489.25	12
8	146774	54.0	Scientist, clinical (histocompatibility and im...	F	Ridgeland	MS	3950.54	11
9	3684340	48.0	Water engineer	F	Heart Butte	MT	3536.49	11
10	9984065	85.0	Engineer, broadcasting (operations)	M	Brinson	GA	4677.91	10
11	8382901	59.0	Commissioning editor	M	Cadiz	KY	7663.82	10
12	4881887	94.0	Engineer, technical sales	F	Surrency	GA	5695.97	8
13	6199553	50.0	Amenity horticulturist	F	Roosevelt	OK	4335.02	8
14	4878815	78.0	Teacher, secondary school	F	Goreville	IL	4085.26	8
15	8098672	47.0	Art therapist	M	Great Mills	MD	5277.95	8
16	277372	38.0	Planning and development surveyor	F	Lake Oswego	OR	1723.41	7
17	2541464	45.0	Buyer, retail	F	Rossville	IL	2641.64	7
18	8459658	37.0	Engineer, biomedical	F	Diamond	OH	1963.02	6
19	5654207	65.0	Retail banker	M	Andrews	IN	1002.07	5

From the SQL output, visualizations were generated to delve into fraudulent activities across different transaction categories.



When considering the relationship between age and fraud, the assumption that older individuals are wiser and therefore less susceptible to fraud is challenged by the data visualized in the scatter plot. The graph indicates no clear relationship between age and the number of fraud cases, dispelling the notion that age is a determinant factor in the likelihood of fraud. This insight demonstrates that fraud does not discriminate by age and that vigilance is necessary across all age groups.



Further analysis of the data using a bar graph sheds light on the prevalence of fraud within various professions. It reveals that the occupation titled 'Gaffer' has the highest number of reported fraudulent activities, which stands out significantly in comparison to other job titles.

This data may suggest targeted areas for fraud prevention efforts or further investigation into the reasons behind this occupation's vulnerability to fraud.

Another bar graph comparing the total number of fraud incidents by gender reveals that female customers are more frequently involved in fraud cases than male customers. While this could suggest a variety of social or behavioral factors, such as shopping habits, it's essential to approach these findings with caution. Correlation does not imply causation, and without a deeper analysis considering additional variables and data, definitive conclusions about gender and susceptibility to fraud cannot be responsibly drawn.

Materialized View

Using Query 3 to create a Materialized View named trans_cust_demo:

```
CREATE MATERIALIZED VIEW IF NOT EXISTS trans_cust_demo

AS

WITH median_transaction AS (
    SELECT
        c.customer_id,
        PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY t.trans_amt) AS
median_trans_amt                        FROM customer c
        JOIN transaction t ON t.customer_id = c.customer_id
    GROUP BY c.customer_id
    )
    SELECT
        c.customer_id,
        c.c_gender,
        c.c_job,
        m.merch_name,
        t.trans_num,
        t.trans_category_type,
        t.trans_date_time,
        t.trans_amt,
        med.median_trans_amt,
        CASE
            WHEN ((t.trans_amt - med.median_trans_amt)/ med.median_trans_amt) * 100
< -349.09 THEN 'Low Risk'
```

```
            WHEN ((t.trans_amt - med.median_trans_amt)/ med.median_trans_amt) * 100
> 467.24 THEN 'High Risk'
            ELSE 'Within Range'
            END AS percentage_diff,
        t.trans_is_fraud
    FROM customer c
        JOIN transaction t ON c.customer_id = t.customer_id
        JOIN merchant m ON t.merchant_id = m.merchant_id
        JOIN median_transaction med ON c.customer_id = med.customer_id
     ORDER BY t.trans_amt DESC
WITH NO DATA;

REFRESH MATERIALIZED VIEW trans_cust_demo;

SELECT * FROM trans_cust_demo;
```

A materialized view named "trans_cust_demo" has been created to encapsulate complex calculations and provide a static snapshot of the data. This view is suitable for scenarios like year-end reporting where the figures do not require frequent updates.

The "trans_cust_demo" view contains a Common Table Expression (CTE) named "median_transaction". This CTE computes the median transaction amount for each customer. The "median_transaction" CTE is then used in the main SELECT query to assemble a detailed view that includes customer details, transaction amounts, merchant information, and a calculated risk category.

The risk category is calculated based on the percentage difference from the median transaction amount. For example, the risk is categorized as "Low Risk" if the percentage difference is below -349.09. It is categorized as "High Risk" if the percentage difference is above 467.24. If the percentage difference falls within this range, it is categorized as "Within Range".

This design leverages the performance benefits of materialized views for heavy calculations that do not require dynamic updating. This approach aligns with the purpose of static reporting.

Index Analysis

Performance tuning using an index is demonstrated by executing an 'Explain Analyze' on Query 5.

```
SELECT
     tr.trans_category_type,
     EXTRACT(MONTH FROM tr.trans_date_time) AS transaction_month,
     COUNT(CASE WHEN tr.trans_is_fraud = True THEN 1 END) AS total_fraud_cases,
     COUNT(CASE WHEN tr.c_gender = 'M' THEN 1 END) AS total_male_involved,
     COUNT(CASE WHEN tr.c_gender = 'F' THEN 1 END) AS total_female_involved,
     PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY tr.trans_amt) AS med_fraud_amt,
     tr.merch_name,
     m.merch_lat,
     m.merch_long
FROM
     trans_cust_demo tr
     JOIN transaction t ON tr.trans_num = t.trans_num
     JOIN merchant m ON t.merchant_id = m.merchant_id
WHERE
     tr.trans_is_fraud = True
GROUP BY
     tr.trans_category_type, transaction_month, tr.merch_name, m.merch_lat,
     m.merch_long
HAVING COUNT(CASE WHEN tr.trans_is_fraud = True THEN 1 END) >= 1
ORDER BY
total_fraud_cases DESC;
```

Explain Analyze Output

	WITHOUT INDEX		WITH INDEX idx_trans_fraud
	Node		**Node**
1	Sort (cost=21901.04..21902.85 rows=722 width=109) (rows=1445 loops=1)	1	Sort (cost=13378.27..13380.07 rows=722 width=109) (rows=1445 loops=1)
2	Aggregate (cost=21048.37..21866.76 rows=722 width=109) (rows=1445 loops=1)	2	Aggregate (cost=12525.6..13343.99 rows=722 width=109) (rows=1445 loops=1)
	Filter: (count(CASE WHEN tr.trans_is_fraud THEN 1 ELSE NULL::integer END) >= 1)		**Filter**: (count(CASE WHEN tr.trans_is_fraud THEN 1 ELSE NULL::integer END) >= 1)
	Rows Removed by Filter: 0		**Rows Removed by Filter**: 0
3	Incremental Sort (cost=21048.37..21772.86 rows=2167 width=88) (rows=2145 loops=1)	3	Incremental Sort (cost=12525.6..13250.09 rows=2167 width=88) (rows=2145 loops=1)
4	Nested Loop Inner Join (cost=21048.07..21675.69 rows=2167 width=88) (rows=2145 loops=1)	4	Nested Loop Inner Join (cost=12525.3..13152.92 rows=2167 width=88) (rows=2145 loops=1)
5	Gather Merge (cost=21047.66..21300.05 rows=2167 width=52) (rows=2145 loops=1)	5	Gather Merge (cost=12524.89..12777.27 rows=2167 width=52) (rows=2145 loops=1)
6	Sort (cost=20047.64..20049.9 rows=903 width=52) (rows=715 loops=3)	6	Sort (cost=11524.87..11527.12 rows=903 width=52) (rows=715 loops=3)
7	Nested Loop Inner Join (cost=0.42..20003.31 rows=903 width=52) (rows=715 loops=3)	7	Hash Inner Join (cost=414.7..11480.53 rows=903 width=52) (rows=715 loops=3)
			Hash Cond: ((t.trans_num)::text = (tr.trans_num)::text)
8	Seq Scan on trans_cust_demo as tr (cost=0..13451.5 rows=903 width=81) (rows=715 loops=3)	8	Seq Scan on transaction as t (cost=0..10188.5 rows=231550 width=37) (rows=185240 loops=3)
	Filter: trans_is_fraud	9	Hash (cost=387.61..387.61 rows=2167 width=81) (rows=2145 loops=1)
	Rows Removed by Filter: 184525		**Buckets**: 4096 **Batches**: 1 **Memory Usage**: 290 kB
9	Index Scan using transaction_pkey on transaction as t (cost=0.42..7.26 rows=1 width=37) (rows=1 loops=2145)	10	Index Scan using idx_trans_fraud on trans_cust_demo as tr (cost=0.42..387.61 rows=2167 width=81) (rows=2145 loops=1)
	Index Cond: ((trans_num)::text = (tr.trans_num)::text)		**Index Cond**: (trans_is_fraud = true)
10	Memoize (cost=0.41..0.47 rows=1 width=20) (rows=1 loops=2145)	11	Memoize (cost=0.41..0.47 rows=1 width=20) (rows=1 loops=2145)
	Buckets: **Batches**: **Memory Usage**: 66 kB		**Buckets**: **Batches**: **Memory Usage**: 66 kB
11	Index Scan using merchant_pkey on merchant as m (cost=0.4..0.46 rows=1 width=20) (rows=1 loops=557)	12	Index Scan using merchant_pkey on merchant as m (cost=0.4..0.46 rows=1 width=20) (rows=1 loops=557)
	Index Cond: (merchant_id = t.merchant_id)		**Index Cond**: (merchant_id = t.merchant_id)

When examining the results of the 'Explain Analyze' command before and after the introduction of the index idx_trans_fraud, a noticeable change in the cost metrics can be observed. Initially, without the index, the query's cost at the point of executing a sequential scan on the trans_cust_demo materialized view was significantly high due to the usage of the trans_is_fraud filter, resulting in a total cost of 21902.85. The introduction of an index on the trans_is_fraud column was deemed necessary to reduce this cost.

After creating the idx_trans_fraud index on the materialized view and rerunning 'Explain Analyze', the total cost was effectively reduced to 13380.07. The execution plan shifted from a sequential scan to an index scan on the trans_cust_demo materialized view. This alteration cut the cost of that specific operation from 13451.5 to 387.61, clearly demonstrating the efficiency gained from indexing. The index scan utilized the newly created index, optimizing the performance by targeting indexed data rather than scanning the entire table.

Conclusion

The data exploration undertaken has shed light on apparent patterns in fraudulent activity, although it is crucial to note that causation does not imply correlation. This foundational analysis is a step towards more advanced analytics needed to identify the factors that genuinely influence fraud.

The inclusion of additional data sets could provide further clarity on fraud patterns, such as crime rates in areas where credit card fraud has occurred or specific events that may impact fraudulent activities during certain periods.

Data manipulation for visualization preparation has been facilitated by the use of Common Table Expressions (CTEs), materialized views, and CASE statements.

Python has proven to be an effective tool in streamlining the process of extracting and transforming data into meaningful results, demonstrating its utility in simplifying complex analytical tasks.

.

References

Kelue, K. (2024, March). *Credit Card Fraud Prediction.* Retrieved from Kaggle: https://www.kaggle.com/datasets/kelvinkelue/credit-card-fraud-prediction?resource=download

PostgreSQL. (2024). *Aggregate Functions.* Retrieved from PostgreSQL: https://www.postgresql.org/docs/9.4/functions-aggregate.html

PostgreSQL Tutorial. (2011). *PostgreSQL CASE.* Retrieved from PostgreSQL Tutorial: https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-case/

APPENDIX A- Work Journal

Group Members	Tasks
1. OBIEME, ESTHER	▪ Query 1: Multi-Table Join (Exploring and transaction Details) ▪ Query 5: Sub-query – Fraud Victims' Demographics ▪ Time graph analysis
2. SANTIAGO, MA. ANGELICA	▪ Query 2: Creating view-identifying frequent customers. ▪ Query 3: CTE & Materialized View – Analyzing Transaction ▪ Query 4: Aggregation – Profiling High-Risk Transactions ▪ Performance Tuning
3. COLLABORATION BETWEEN TEAM MEMBER	▪ Introduction ▪ Choosing Dataset ▪ ERD and Data-dictionary ▪ Graph Analytics ▪ Presentation ▪ Documentation