Introduction

Frauds happen in many forms, which are identity, investment, health care, etc. One of the most common frauds that experienced by 50% of Americans is payment card fraud that occurred multiple times to more than 1/3 of the card holders (to, 2004).

Although the implication of fraud can be serious, the fraud rate is incredibly low. According to Clearly Payment, the rate for CNP (Card-Not-Present) fraud in E-commerce industry is only 0.93% (Kalle Radage, 2024). Therefore, it can be tedious to spot fraudulent activities from the big volume of transaction data.

Merchants and financial institutions can prevent payment fraud by using anti-fraud services or software, where big data technologies are involved to detect certain patterns in fraudulent transactions. To further enhance accuracy, large datasets are needed for training and testing in machine learning.

Data collection

The dataset, fraud_data.csv, contains 15 variables and 14447 records. Data cleaning is required due to inconsistent values under several columns: trans_date_trans_time, job, merchant, dob, is_fraud. The highlighted values required standardization.

trans_date_trans_t	merchant	category	amt	city	stat	lat	long	city_pc	job	dob	trans_num	merch_la	merch_long	is_fraud
2020-10-12 2:12	Harber Inc	gas_transport	10.97	Lakeport	CA	39.047	-122.9328	11256	Podiatrist	18-10-1972	29b86851da4ca18a	39.217949	-122.002342	1
2020-10-12 3:06	Gutmann-Upton	misc_pos	8.76	Lakeport	CA	39.047	-122.9328	11256	Podiatrist	18-10-1972	ecae650463320906	38.596368	-123.141124	1
2020-10-12 17:30	"Schroeder, Wolff and Hermisto	travel	8.96	Lakeport	CA	39.047	-122.9328	11256	Podiatrist	18-10-1972	5dd2b06961eb8ec5	39.050171	-122.692423	1
2020-10-12 23:47	"Wuckert, Wintheiser and Friese	home	261.03	Lakeport	CA	39.047	-122.9328	11256	Podiatrist	18-10-1972	e20e4793d1662f7e	39.873689	-122.26241	1
2020-11-12 1:51	Huel-Langworth	misc_net	890.64	Lakeport	CA	39.047	-122.9328	11256	Podiatrist	18-10-1972	b1b4b5dd6a24ce4b	39.91141	-122.528368	1
2020-11-12 2:10	Kiehn Inc	grocery_pos	343.37	Lakeport	CA	39.047	-122.9328	11256	Podiatrist	18-10-1972	3bbec2bcebdc5986	38.729526	-122.571755	1
2020-11-12 2:14	Stiedemann Lt	food_dining	101.04	Lakeport	CA	39.047	-122.9328	11256	Podiatrist	18-10-1972	9e0a1109a6797b2c	38.674714	-123.544141	1
2020-11-12 3:51	"Yost, Block and Koepp"	misc_pos	8.33	Lakeport	CA	39.047	-122.9328	11256	Podiatrist	18-10-1972	858f16a84b02a891	38.10709	-121.933423	1
2020-11-12 3:52	"Bahringer, Schoen and Corkery	shopping_pos	770.65	Lakeport	CA	39.047	-122.9328	11256	Podiatrist	18-10-1972	65c4a80dabf9b440	39.721476	-122.611438	1
2020-11-12 12:36	Kris-Kertzmann	travel	7.88	Lakeport	CA	39.047	-122.9328	11256	Podiatrist	18-10-1972	6b7742d1412057c6	39.139593	-122.688798	1
2020-11-12 22:29	"Conroy, Balistreri and Gorczan	health_fitness	18.46	Lakeport	CA	39.047	-122.9328	11256	Podiatrist	18-10-1972	0563199dc605dc06	39.868656	-123.337295	1
2020-11-12 23:05	Ratke and Sons	health_fitness	17.35	Lakeport	CA	39.047	-122.9328	11256	Podiatrist	18-10-1972	7e48fbbbc83523b3	38.552146	-122.519483	1
2020-11-12 23:19	Thompson-Gleason	health_fitness	19.45	Lakeport	CA	39.047	-122.9328	11256	Podiatrist	18-10-1972	bfde75d978bb9905	39.25188	-122.490946	1"2020-12-24 16:56:24"
24-12-2020 16:58	"Hahn, Douglas and Schowalte"	travel	440.56	Meadville	MO	39.7795	-93.3014	964	Tourist information centre m	23-12-1974	68a845f709866a0f	39.419072	-93.9479	0
24-12-2020 16:59	Erdman-Durgan	health_fitness	60.39	Crownpoint	NM	35.7206	-108.0271	5662	IT consultant	1989-08-04	17075780fd7851b2	36.123857	-107.164356	0
24-12-2020 16:59	"Prosacco, Kreiger and Kovacek'	home	16.13	Mound City	MO	40.1362	-95.2138	1631	Architect	20-01-1953	a38ea67ec77fe7e0	39.984044	-96.203203	0
24-12-2020 16:59	"Romaguera, Cruickshank and G	shopping_net	6.7	Napa	CA	38.4549	-122.2564	94014	Airline pilot	21-08-1985	32455d6fefc982ae2	38.229234	-122.499378	0
24-12-2020 17:00	Spencer-Runolfsson	misc_pos	55.61	Hawthorne	CA	33.9143	-118.3493	93193	"Editor, magazine features"	19-04-1995	ca3d23dda0ff4ec42	33.66463	-117.730522	0
24-12-2020 17:11	Kuhn LLC	shopping_pos	2.58	Moab	UT	38.5677	-109.5271	9772	Location manager	24-11-1989	67a249c83b7adad9	38.956696	-109.612304	0
24-12-2020 17:16	"Willms, Kris and Bergnaum"	shopping_pos	53.52	Kansas City	MO	38.9621	-94.5959	545147	Counsellor	18-11-1987	ee30fa8d87497775	38.432755	-93.981096	0
24-12-2020 17:18	"Roberts, Ryan and Smith"	personal_care	81.53	Newhall	CA	34.3795	-118.523	34882	Health physicist	25-04-1971	abbf11c821a67b61	33.91462	-118.660667	0
24-12-2020 17:18	Goyette-Gerhol	kids_pets	44.61	Huntington Be	CA	33.6773	-118.0051	190249	"Therapist, horticultural"	17-09-1976	7a5cb8e0529501db	33.406105	-117.630637	0

The details of the variables and ideal data type are listed as below:

Variable	Description	Data type (ideal)	Current Values / example
trans_date_trans_time	Transaction datetime	VARCHAR(30)	4-1-2019 12:58:00 AM 14-01-2019 02:27
merchant	Name of the merchant	VARCHAR(255)	Predovic Inc

category	Merchant category	VARCHAR(255)	shopping_net grocery_pos
amt	Transaction amount	DECIMAL(10, 2)	367.29
city	City where credit card holder is located	VARCHAR(255)	Browning
state	State where credit card holder is located	CHAR(2)	MO
lat	Latitude of the purchase	DECIMAL(9,6)	40.029
long	Logitude of the purchase	DECIMAL(9,6)	-93.1607
city_pop	Card holder's city population	INT	602
job	Card holder's job	VARCHAR(255)	Cytogeneticist "Administrator, education"
dob	Card holder's birthday	CHAR(10)	1939-09-11 18-11-1987
trans_num	Transaction number	VARCHAR(40)	8e2d2fae5319d31c887dddbc70627ac4
merch_lat	Merchant's latitude	DECIMAL(9,6)	63.917785
merch_long	Merchant's logitude	DECIMAL(9,6)	-165.827621
is_fraud	Indication if the transaction is fraud • 0: Not fraud • 1: fraud	BOOLEAN	0 1

(Choudhury, 2024)

Problems and results

The aim of the study is to spot fraudulent transaction patterns with the information given in the dataset. However, pre-processing works need to be done before the analysis and study begin. Steps, problems faced during every process and analysis are recorded. The database engine used to store and process the data is SQLite.

Load data into database

1. CREATE TABLE fraud: “trans_date_trans_time” and “is_fraud” are created as VARCHAR for data cleaning to be conducted.

Query History

```
CREATE TABLE fraud(
    trans_date_trans_time VARCHAR(30),
    merchant VARCHAR(255),
    category VARCHAR(255),
    amt DECIMAL(10, 2),
    city VARCHAR(255),
    state CHAR(2),
    lat DECIMAL(9,6),
    long DECIMAL(9,6),
    city_pop INT,
    job VARCHAR(255),
    dob CHAR(10),
    trans_num VARCHAR(40),
    merch_lat DECIMAL(9,6),
    merch_long DECIMAL(9,6),
    is_fraud VARCHAR(255)
);
```

Structure Data Constraints Indexes Triggers DDL

fraud_data Table name: fraud WITHOUT ROWID STRICT

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated	
1	trans_date_trans_time	VARCHAR (30)								NULL
2	merchant	VARCHAR (255)								NULL
3	category	VARCHAR (255)								NULL
4	amt	DECIMAL (10,...								NULL
5	city	VARCHAR (255)								NULL
6	state	CHAR (2)								NULL
7	lat	DECIMAL (9, 6)								NULL
8	long	DECIMAL (9, 6)								NULL
9	city_pop	INT								NULL
10	job	VARCHAR (255)								NULL
11	dob	CHAR (10)								NULL
12	trans_num	VARCHAR (40)								NULL
13	merch_lat	DECIMAL (9, 6)								NULL
14	merch_long	DECIMAL (9, 6)								NULL
15	is_fraud	VARCHAR (255)								NULL

2. Import the fraud_data.csv into the table in SQLite CLI after setting the mode to CSV.

```
SQLite version 3.47.0 2024-10-21 16:30:22
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .open fraud_db.db
sqlite> .tables
fraud
sqlite> .mode csv
sqlite> .import fraud_data.csv fraud
sqlite>
```

3. SELECT * FROM fraud: Get a glimpse of the imported data

Query History

```
SELECT * FROM fraud;
```

Grid view Form view

Total rows loaded: 14447

	trans_date_trans_time	merchant	category	amt	city	state	lat	long	city_pop	job	dob	trans_nu
1	trans_date_trans_time	merchant	category	amt	city	state	lat	long	city_pop	job	dob	trans_nu
2	04-01-2019 00:58	"Stokes, Christiansen and Sipes"	grocery_net	14.37	Wales	AK	64.7556	-165.6723	145	"Administrator, education"	09-11-1939	a3806e9
3	04-01-2019 15:06	Predovic Inc	shopping_net	966.11	Wales	AK	64.7556	-165.6723	145	"Administrator, education"	09-11-1939	a59185fe
4	04-01-2019 22:37	Wisozk and Sons	misc_pos	49.61	Wales	AK	64.7556	-165.6723	145	"Administrator, education"	09-11-1939	86ba3a8
5	04-01-2019 23:06	Murray-Smitham	grocery_pos	295.26	Wales	AK	64.7556	-165.6723	145	"Administrator, education"	09-11-1939	3a068fe1
6	04-01-2019 23:59	Friesen Lt	health_fitness	18.17	Wales	AK	64.7556	-165.6723	145	"Administrator, education"	09-11-1939	891cdd1
7	05-01-2019 03:15	"Raynor, Reinger and Hagenes"	gas_transport	20.45	Wales	AK	64.7556	-165.6723	145	"Administrator, education"	09-11-1939	ef010a5f
8	05-01-2019 03:21	Heller-Langosh	gas_transport	18.19	Wales	AK	64.7556	-165.6723	145	"Administrator, education"	09-11-1939	8e2d2fae
9	05-01-2019 11:31	Padberg-Welch	grocery_pos	367.29	Browning	MO	40.029	-93.1607	602	Cytogeneticist	14-07-1954	5fbe827
10	05-01-2019 18:03	McGlynn-Heathcote	misc_net	768.15	Wales	AK	64.7556	-165.6723	145	"Administrator, education"	09-11-1939	fba83e0a
11	05-01-2019 22:02	Dooley-Thompson	misc_net	849.49	Wales	AK	64.7556	-165.6723	145	"Administrator, education"	09-11-1939	b87c92d
12	05-01-2019 22:05	"Gottlieb, Considine and Schultz"	shopping_net	1177.79	Browning	MO	40.029	-93.1607	602	Cytogeneticist	14-07-1954	f1c51701
13	05-01-2019 22:12	"Moen, Reinger and Murphy"	grocery_pos	307.09	Wales	AK	64.7556	-165.6723	145	"Administrator, education"	09-11-1939	755e4e8
14	05-01-2019 22:18	"Hauck, Dietrich and Funk"	kids_pets	4.58	Wales	AK	64.7556	-165.6723	145	"Administrator, education"	09-11-1939	8fa7880c
15	05-01-2019 22:32	Pouros-Haag	shopping_pos	730.78	Wales	AK	64.7556	-165.6723	145	"Administrator, education"	09-11-1939	2396a5b
16	05-01-2019 22:33	Goyette Inc	shopping_net	1006.4	Wales	AK	64.7556	-165.6723	145	"Administrator, education"	09-11-1939	4d7e567
17	05-01-2019 22:38	"Baumbach, Strosin and Nicolas"	shopping_pos	830.72	Wales	AK	64.7556	-165.6723	145	"Administrator, education"	09-11-1939	773a330

Data Cleaning

1. COUNT(*): Find empty cells for all columns and filter with WHERE clause. The result shown 0 null values.

```
SELECT COUNT(*) FROM fraud
WHERE trans_date_trans_time IS NULL
    OR merchant IS NULL
    OR category IS NULL
    OR amt IS NULL
    OR city IS NULL
    OR state IS NULL
    OR lat IS NULL
    OR long IS NULL
    OR city_pop IS NULL
    OR job IS NULL
    OR dob IS NULL
    OR trans_num IS NULL
    OR merch_lat IS NULL
    OR merch_long IS NULL
    OR is_fraud IS NULL;
```

Total rows loaded: 1

	COUNT(*)
1	0

2. DELETE FROM fraud WHERE amt = ‘amt’: Remove replicated header record. COUNT(*): Applied to get the total records, where 14446 is obtained.

```
DELETE FROM fraud
WHERE amt = 'amt';

SELECT COUNT(*)
FROM fraud;
```

Total rows loaded: 1

	COUNT(*)
1	14446

3. REPLACE: Remove the double quote by updating the table columns.

```
UPDATE fraud
SET
    merchant = REPLACE(merchant, '"', ''),
    job = REPLACE(job, '"', '');
```

4. Standardize “is_fraud” to 0 and 1 by editing the odd values:

Steps	Query	Result
Find and count distinct values	`SELECT DISTINCT is_fraud FROM fraud;` `SELECT DISTINCT is_fraud, COUNT(is_fraud) FROM fraud GROUP BY is_fraud;`	is_fraud 1 1 2 1"2020-12-24 16:56:24" 3 0 4 0"2019-01-01 00:00:44" is_fraud / COUNT(is_fraud) 1 0 / 12600 2 0"2019-01-01 00:00:44" / 1 3 1 / 1844 4 1"2020-12-24 16:56:24" / 1

Filter "trans_num" with WHERE clause	```sql
	SELECT trans_num, is_fraud
	FROM fraud
	WHERE is_fraud != 0 AND is_fraud != 1;
	``
	The "trans_num" are unique to lead to odd "is_fraud" value
		trans_num, is_fraud
		1 bfde75d978bb9905a4a8c87440692a4c, 1"2020-12-24 16:56:24"
		2 14392d723bb7737606b2700ac791b7aa, 0"2019-01-01 00:00:44"

Update the values to 0 and 1	```sql
	UPDATE fraud
	SET is_fraud = CASE
	WHEN trans_num = 'bfde75d978bb9905a4a8c87440692a4c' THEN 1
	WHEN trans_num = '14392d723bb7737606b2700ac791b7aa' THEN 0
	ELSE is_fraud
	END;
	
	SELECT DISTINCT is_fraud, COUNT(is_fraud)
	FROM fraud
	GROUP BY is_fraud;
	``
	Change the "is_fraud" values and remain the rest accordingly
		is_fraud, COUNT(is_fraud)
		1 0, 12601
		2 1, 1845

5. Date and time values are auto-format into "DD-MM-YYYY" upon import into SQLite, hence no cleaning nor editing required. Samples in CSV and table are filtered for comparison.
   a) "trans_date_trans_time" in "DD-MM-YYYY HH:MM"
   b) "dob" in "DD-MM-YYYY"

Field	CSV	Database "fraud" table

trans_date_trans_time	trans_date_trans_t, merchant
	2019-1-1 9:29, "Raynor, Reinger and Hagenes"
	2019-1-1 11:56, "Raynor, Reinger and Hagenes"
	2019-3-1 3:17, "Raynor, Reinger and Hagenes"
	2019-4-1 2:03, "Raynor, Reinger and Hagenes"
	2019-4-1 2:13, "Raynor, Reinger and Hagenes"
	2019-5-1 3:15, "Raynor, Reinger and Hagenes"
	2019-5-1 5:10, "Raynor, Reinger and Hagenes"
	2019-11-1 3:19, "Raynor, Reinger and Hagenes"
	13-01-2019 00:36, "Raynor, Reinger and Hagenes"
	13-01-2019 02:29, "Raynor, Reinger and Hagenes"
	14-01-2019 11:02, "Raynor, Reinger and Hagenes"
	20-01-2019 03:23, "Raynor, Reinger and Hagenes"
		``sql
		SELECT trans_date_trans_time, merchant
		FROM fraud
		WHERE merchant = 'Raynor, Reinger and Hagenes';
		``
		Grid view, Form view, Total rows loaded: 36
		trans_date_trans_time, merchant
		1 01-01-2019 09:29, Raynor, Reinger and Hagenes
		2 01-01-2019 11:56, Raynor, Reinger and Hagenes
		3 03-01-2019 03:17, Raynor, Reinger and Hagenes
		4 04-01-2019 02:03, Raynor, Reinger and Hagenes
		5 04-01-2019 02:13, Raynor, Reinger and Hagenes
		6 05-01-2019 03:15, Raynor, Reinger and Hagenes
		7 05-01-2019 03:15, Raynor, Reinger and Hagenes
		8 05-01-2019 05:10, Raynor, Reinger and Hagenes
		9 11-01-2019 03:19, Raynor, Reinger and Hagenes

dob	Cytogeneticist, 14-07-1954
	Cytogeneticist, 14-07-1954
	Cytogeneticist, 14-07-1954
	Cytogeneticist, 14-07-1954
	Occupational psychologist, 1954-05-07
	Occupational psychologist, 1954-05-07
	Occupational psychologist, 1954-05-07
	Occupational psychologist, 1954-05-07
	Occupational psychologist, 1954-05-07
	Occupational psychologist, 1954-05-07
	Occupational psychologist, 1954-05-07
		``sql
		SELECT job,dob
		FROM fraud
		WHERE dob LIKE '%1954%';
		``
		Grid view, Form view, Total rows loaded: 219
		job, dob
		95 Cytogeneticist, 14-07-1954
		96 Cytogeneticist, 14-07-1954
		97 Cytogeneticist, 14-07-1954
		98 Cytogeneticist, 14-07-1954
		99 Occupational psychologist, 05-07-1954
		100 Occupational psychologist, 05-07-1954
		101 Occupational psychologist, 05-07-1954
		102 Occupational psychologist, 05-07-1954
		103 Occupational psychologist, 05-07-1954
		104 Occupational psychologist, 05-07-1954

Analysis

1. Aggregation:
    a) Split “trans_date_trans_time” and “dob” into separate columns.
    b) Find age of card holder by taking “trans_year” to minus “dob_year”
    c) Calculate distance between location of merchants and card holder by using Harversine formula

Field: trans_date_trans_time

Queries

```
-- create new columns to split trans_date and time
ALTER TABLE fraud ADD COLUMN trans_day INT;
ALTER TABLE fraud ADD COLUMN trans_month INT;
ALTER TABLE fraud ADD COLUMN trans_year INT;
ALTER TABLE fraud ADD COLUMN trans_hour INT;
ALTER TABLE fraud ADD COLUMN trans_minute INT;

UPDATE fraud
SET
    trans_day = CAST(substr(trans_date_trans_time, 1, 2) AS INTEGER),
    trans_month = CAST(substr(trans_date_trans_time, 4, 2) AS INTEGER),
    trans_year = CAST(substr(trans_date_trans_time, 7, 4) AS INTEGER),
    trans_hour = CAST(substr(trans_date_trans_time, 12, 2) AS INTEGER),
    trans_minute = CAST(substr(trans_date_trans_time, 15, 2) AS INTEGER);

SELECT trans_date_trans_time,
    trans_day, trans_month,
    trans_year,
    trans_hour,
    trans_minute
FROM fraud;
```

Result

Grid view | Form view

Total rows loaded: 14446

	trans_date_trans_time	trans_day	trans_month	trans_year	trans_hour	trans_minute
1	01-01-2019 00:00	1	1	2019	0	0
2	01-01-2019 00:07	1	1	2019	0	7
3	01-01-2019 00:09	1	1	2019	0	9
4	01-01-2019 00:21	1	1	2019	0	21
5	01-01-2019 00:22	1	1	2019	0	22
6	01-01-2019 00:22	1	1	2019	0	22
7	01-01-2019 00:22	1	1	2019	0	22
8	01-01-2019 00:31	1	1	2019	0	31
9	01-01-2019 00:34	1	1	2019	0	34
10	01-01-2019 00:40	1	1	2019	0	40
11	01-01-2019 00:41	1	1	2019	0	41

Field: dob

Queries

```
ALTER TABLE fraud ADD COLUMN dob_year INT;
UPDATE fraud
    SET dob_year = CAST(substr(dob, 7, 4) AS INTEGER);

SELECT dob, dob_year
FROM fraud;
```

Result

Grid view | Form view

	dob	dob_year
1	01-03-1960	1960
2	01-03-1960	1960
3	01-03-1960	1960
4	01-03-1960	1960
5	01-03-1960	1960
6	01-03-1960	1960
7	01-03-1960	1960
8	01-03-1960	1960

Field: age

Queries

```
ALTER TABLE fraud ADD COLUMN age INT;
UPDATE fraud SET age = trans_year - dob_year;

SELECT trans_date_trans_time, dob, age
FROM fraud;
```

Result

	trans_date_trans_time	dob	age
1	04-01-2019 00:58	09-11-1939	80
2	04-01-2019 15:06	09-11-1939	80
3	04-01-2019 22:37	09-11-1939	80
4	04-01-2019 23:06	09-11-1939	80
5	04-01-2019 23:59	09-11-1939	80

Field: Distance_km

Queries

```
ALTER TABLE fraud ADD COLUMN distance_km FLOAT;
UPDATE fraud
    SET distance_km = printf("%.4f", 6371 * 2 * ASIN(SQRT(
        POWER(SIN((RADIANS(merch_lat) - RADIANS(lat)) / 2), 2) +
        COS(RADIANS(lat)) * COS(RADIANS(merch_lat)) *
        POWER(SIN((RADIANS(merch_long) - RADIANS(long)) / 2), 2)
    )));

SELECT merch_lat, merch_long, lat, long, distance_km
FROM fraud;
```

Result

	merch_lat	merch_long	lat	long	distance_km
1	65.654142	-164.722603	64.7556	-165.6723	109.2856
2	65.468863	-165.473127	64.7556	-165.6723	79.8569
3	65.347667	-165.914542	64.7556	-165.6723	66.8079
4	64.445035	-166.080207	64.7556	-165.6723	39.6362
5	65.447094	-165.446843	64.7556	-165.6723	77.6115
6	64.088838	-165.104078	64.7556	-165.6723	78.9988

2. Views are created to store the analysis for studying factors affecting frauds. CASE and WHEN expressions are heavily used. Percentage fraud are calculated for better comparison with other classes.
    a) age_fraud: fraud by age groups

```sql
-- AGE FRAUD
CREATE VIEW age_fraud AS
    SELECT
        is_fraud,
        COUNT(is_fraud) AS total_fraud,
        SUM(CASE WHEN age>=60 THEN 1 ELSE 0 END) AS over_60,
        SUM(CASE WHEN age<60 AND age >= 40 THEN 1 ELSE 0 END) AS "40-60",
        SUM(CASE WHEN age<40 AND age >= 20 THEN 1 ELSE 0 END) AS "20-40",
        SUM(CASE WHEN age<20 THEN 1 ELSE 0 END) AS below_20
    FROM fraud_agg
    GROUP BY is_fraud;

SELECT * FROM age_fraud;
```

Grid view | Form view

Total rows loaded: 2

	is_fraud	total_fraud	over_60	40-60	20-40	below_20
1	0	12601	2955	4862	4660	124
2	1	1845	587	699	559	0

b) amt_fraud: Fraud by transaction amount, the average fraudulent amount is $517.96.

```sql
-- AMOUNT FRAUD
CREATE VIEW amt_fraud AS
    SELECT
        rounded_avg.avg_fraud_amt,
        SUM(CASE WHEN amt >= rounded_avg.avg_fraud_amt AND is_fraud = 1 THEN 1 ELSE 0 END) AS fraud_case,
        (SELECT COUNT(*) FROM fraud_agg WHERE is_fraud = 1) AS total_fraud,
        ((SUM(CASE
            WHEN amt >= rounded_avg.avg_fraud_amt
            AND is_fraud = 1
                THEN 1 ELSE 0 END)*100)/
            (SELECT COUNT(*) FROM fraud_agg WHERE is_fraud = 1))AS fraud_pct
    FROM fraud_agg,
        (SELECT ROUND(AVG(amt), 2) AS avg_fraud_amt FROM fraud_agg WHERE is_fraud=1) AS rounded_avg;

SELECT * FROM amt_fraud;
```

Grid view | Form view

Total rows loaded: 1

	avg_fraud_amt	fraud_case	total_fraud	fraud_pct
1	517.96	863	1845	46

c) cat_fraud: Fraud by categories

```sql
-- category fraud
CREATE VIEW cat_fraud AS
    SELECT
        category,
        COUNT(category) AS total_trans,
        SUM(CASE WHEN is_fraud = 1 THEN 1 ELSE 0 END) AS fraud_cases,
        ROUND((SUM(CASE WHEN is_fraud = 1 THEN 1 ELSE 0 END) * 100.0) / COUNT(category), 2) AS fraud_pct
    FROM fraud_agg
    GROUP BY category
    ORDER BY fraud_pct DESC;

SELECT * FROM cat_fraud;
```

	category	total_trans	fraud_cases	fraud_pct
1	shopping_net	1408	396	28.13
2	grocery_pos	1602	444	27.72
3	misc_net	821	223	27.16
4	shopping_pos	1354	194	14.33
5	gas_transport	1430	159	11.12
6	travel	385	34	8.83
7	misc_pos	823	64	7.78
8	grocery_net	474	32	6.75
9	entertainment	953	59	6.19
10	personal_care	990	57	5.76
11	kids_pets	1141	56	4.91
12	food_dining	870	39	4.48
13	health_fitness	891	37	4.15
14	home	1304	51	3.91

d) distance_fraud: Relationship between fraudulent records and distance (KM) between the purchaser and merchant's location.

```sql
-- distance fraud
CREATE VIEW distance_fraud AS
    SELECT
        rounded_avg.avg_fraud_km,
        SUM(CASE
                WHEN distance_km >= rounded_avg.avg_fraud_km AND is_fraud = 1
        THEN 1 ELSE 0 END) AS fraud_case,
        (SELECT COUNT(*) FROM fraud_agg WHERE is_fraud = 1) AS total_fraud,
        ((SUM(CASE
            WHEN distance_km >= rounded_avg.avg_fraud_km AND is_fraud = 1
                THEN 1 ELSE 0 END)*100)/(
                SELECT COUNT(*)
                    FROM fraud_agg
                    WHERE is_fraud = 1))AS fraud_pct
    FROM fraud_agg,
        (SELECT ROUND(AVG(distance_km), 2) AS avg_fraud_km
            FROM fraud_agg
            WHERE is_fraud=1) AS rounded_avg;

SELECT * FROM distance_fraud;
```

Grid view | Form view | Total rows loaded: 1

	avg_fraud_km	fraud_case	total_fraud	fraud_pct
1	75.47	968	1845	52

e) hour_fraud: Frequency of fraud by in different hours, order by highest percentage.

```sql
-- transaction hour fraud
CREATE VIEW hour_fraud AS
    SELECT
        trans_hour,
        COUNT(trans_hour) AS total_trans,
        SUM(CASE
            WHEN is_fraud = 1
            THEN 1 ELSE 0 END) AS fraud,
        (SUM(CASE
            WHEN is_fraud = 1
                THEN 1 ELSE 0 END) * 100 /
            COUNT(trans_hour)) AS fraud_pct
    FROM fraud_agg
    GROUP BY trans_hour
    ORDER BY fraud DESC;

SELECT * FROM hour_fraud;
```

Grid view | Form view | Total rows loaded: 24

	trans_hou	total_trans	fraud	fraud_pct
1	23	1127	477	42
2	22	1120	468	41
3	1	613	174	28
4	0	587	174	29
5	3	564	147	26
6	2	521	142	27
7	16	653	24	3
8	15	671	24	3
9	20	702	23	3
10	18	664	20	3
11	17	666	20	3
12	21	692	18	2
13	12	624	17	2
14	14	630	16	2
15	13	638	14	2
16	8	418	14	3
17	9	397	12	3
18	19	667	11	1
19	10	418	10	2
20	5	406	10	2
21	7	438	9	2
22	11	402	7	1
23	6	415	7	1

f) state_fraud: Fraud cases in different states.

```sql
CREATE VIEW state_fraud AS
    SELECT
        state,
        COUNT(state) AS total_trans,
        SUM(CASE
            WHEN is_fraud = 1
                THEN 1 ELSE 0 END) AS fraud_cases,
        ROUND((SUM(CASE
                    WHEN is_fraud = 1
                    THEN 1 ELSE 0 END) * 100.0) /
                    COUNT(state), 2) AS fraud_pct
    FROM fraud_agg
    GROUP BY state
    ORDER BY fraud_pct DESC;

SELECT * FROM state_fraud;
```

	state	total_trans	fraud_cases	fraud_pct
1	AK	173	65	37.57
2	NE	1460	238	16.3
3	OR	1211	197	16.27
4	CO	856	115	13.43
5	UT	597	73	12.23
6	CA	3375	411	12.18
7	NM	1003	121	12.06
8	MO	2329	267	11.46
9	WA	1150	126	10.96
10	WY	1100	119	10.82
11	ID	347	33	9.51
12	AZ	673	64	9.51
13	HI	172	16	9.3

Results

From the analysis, some patterns of the fraud are obvious, such as the transaction hours where fraud is most active between 10PM to 3AM, highest at 11PM; occurred frequently in “shopping_net” of 28.13%, “grocery_pos” and “misc_net”, lowest in “home” of 3.91%. There is 46% chance of fraud when the transaction amount passed $517.9, and 52% likelihood at average distance between purchaser and merchant of 75.47KM. Not to mention, it is saddening to see that the age group which has the highest number of fraud victims is between 40 to 60.

(762 words)

Reference

to, C. (2004, September 16). *inclusive term for fraud committed using a payment card, such as a credit card or debit card*. Wikipedia.org; Wikimedia Foundation, Inc. https://en.wikipedia.org/wiki/Credit_card_fraud#:~:text=A%20few%20examples%20of%20credit,fraudsters%20attempting%20to%20steal%20money.

Kalle Radage. (2024, July 16). *Fraud Risk by Industry in Payments*. Credit Card Processing and Merchant Account. https://www.clearlypayments.com/blog/fraud-risk-by-industry-in-payments/

Shah, J. (2022). *Online Payment Fraud Detection*. Kaggle.com. https://www.kaggle.com/datasets/jainilcoder/online-payment-fraud-detection

Choudhury, N. R. (2024). *Credit Card Fraud data*. Kaggle.com. https://www.kaggle.com/datasets/neharoychoudhury/credit-card-fraud-data