Phase 3- OLAP Queries, and BI Dashboard
Phase 4 - Data Mining

CSI 4142 - Fundamentals of Data Science
Winter 2023

Faculté de génie
Faculty of Engineering

School of Electrical Engineering and Computer Science
University of Ottawa

Yazan Otoum, Ph.D

Group #30 : Deliverable #3

Ishanveer Gobin 300135454

Andie Samadoulougou 300209487

Kate Sin Yan Chun 300144923

Table of Content

Phase 3- OLAP Queries, and BI Dashboard

Part A.1. Standard OLAP operations

For each part of this section, we created a view for each query to save the data retrieved.

Drill down and roll up

a) Roll-up data for payment type.

This query is to aggregate the data by payment type, providing a higher-level summary of the data. It counts the number of transactions, the number of fraudulent transactions, the average name_email_similarity, the average velocity_6h, the average velocity_24h and the average velocity_4w for each payment type.

```
banking_transactions=# select * from payment_type_totalFraud;
 payment_type | num_transactions | num_fraud_transactions | avg_name_email_similarity |  avg_velocity_6h   | avg_velocity_24h  |  avg_velocity_4w
--------------+------------------+------------------------+---------------------------+--------------------+-------------------+-------------------
 AA           |           251116 |                   1299 |        0.5062369516880528 |  5873.947259812476 | 4949.561097351668 | 4944.701874919875
 AB           |           359562 |                   3969 |       0.48035679233994283 | 5274.0677337043535 |  4602.53973276819 | 4755.131827576077
 AC           |           246779 |                   4048 |       0.49863524099808243 |   5391.55929410425 | 4644.281496943974 | 4774.584219449472
 AD           |           115331 |                   1229 |         0.500514889339166 |  5626.761334069413 |   4890.7109741748 | 5041.071465683198
 AE           |              280 |                      0 |       0.46063428706467463 |  5892.876745837381 | 5101.506703275988 | 4977.958443779203
(5 rows)
```

b) Roll-up data for employment status

This query is to aggregate the data by employment_status, providing a higher-level summary of the data. It counts the number of transactions and the number of fraudulent transactions for each employment status.

```
banking_transactions=# select * from Employment_status_fraud;
 employment_status | num_transactions | num_fraud_transactions
-------------------+------------------+------------------------
 CA                |           714243 |                   8625
 CB                |           133734 |                    900
 CC                |            33652 |                    783
 CD                |            25876 |                     94
 CE                |            22158 |                     52
 CF                |            42970 |                     84
 CG                |              435 |                      7
(7 rows)
```

c) Drill-down data for customer age group

This query aims to break down the data into more detailed levels by analyzing fraud cases based on age groups. The customer age in the dataset was in bins per decade (e.g, 20-29 is represented as 20). The query counts the number of transactions and the number of fraudulent transactions for each age group.

```
banking_transactions=# select * from age_group_fraud;
 age_group | num_transactions | num_fraud_transactions
-----------+------------------+------------------------
 10        |            20452 |                     73
 20        |           240726 |                   1181
 30        |           306098 |                   2553
 40        |           234165 |                   2835
 50        |           137403 |                   2766
 60        |            34224 |                   1137
(6 rows)
```

From the age_group_fraud view, we selected all where the customer age group was between 20-29 (age_group = 20).

```
banking_transactions=# select * from age_group_fraud where age_group = '20';
 age_group | num_transactions | num_fraud_transactions
-----------+------------------+------------------------
 20        |           240726 |                   1181
(1 row)
```

d) Drill-down data for customer income

This query drills down the data to analyze the number of transactions and fraudulent transactions by different income groups (high, medium and low).

```
banking_transactions=# select * from fraud_transactions_by_income;
 income_group | num_transactions | num_fraud_transactions
--------------+------------------+------------------------
 High         |           571190 |                   7955
 Low          |           219334 |                   1271
 Medium       |           182544 |                   1319
(3 rows)
```

Slice

a) Slice data by payment_type “AB”

The query selects a single payment type from the transaction dimension and analyzes the data based on customer age and device OS. It counts the number of transactions and the number of fraudulent transactions for each combination of customer_age and device_os.

```
banking_transactions=# select * from payment_type_slice;
 customer_age | device_os | num_transactions | num_fraud_transactions
--------------+-----------+------------------+------------------------
           10 | linux     |             1154 |                      5
           10 | macintosh |              740 |                      0
           10 | other     |             2173 |                      4
           10 | windows   |             1653 |                      9
           10 | x11       |               29 |                      0
           20 | linux     |            24659 |                     63
           20 | macintosh |             5705 |                     39
           20 | other     |            28942 |                     86
           20 | windows   |            22770 |                    230
           20 | x11       |              365 |                      6
           30 | linux     |            37296 |                    155
           30 | macintosh |             4896 |                     60
           30 | other     |            35649 |                    177
           30 | windows   |            28416 |                    545
           30 | x11       |              425 |                      5
           40 | linux     |            30381 |                    149
           40 | macintosh |             4187 |                     75
           40 | other     |            27806 |                    155
           40 | windows   |            28498 |                    603
           40 | x11       |              520 |                      4
           50 | linux     |            20057 |                    164
           50 | macintosh |             2232 |                     36
           50 | other     |            12284 |                    126
           50 | windows   |            23512 |                    780
           50 | x11       |              414 |                      9
           60 | linux     |             5172 |                     78
           60 | macintosh |              566 |                     24
           60 | other     |             2777 |                     45
           60 | windows   |             6154 |                    335
           60 | x11       |              130 |                      2
(30 rows)
```

b) Slice data by housing_status “BA”

Ths query slices the data to show the number of transactions and the number of fraudulent transactions for customers with or without other cards where their housing_status is “BA”.

```
banking_transactions=# select * from fraud_by_has_other_cards;
 has_other_cards | num_transactions | num_fraud_transactions
-----------------+------------------+------------------------
 f               |           122074 |                   5539
 t               |            42865 |                    553
(2 rows)
```

Dice

a) Dice data by customer age and source

The dice operation selects multiple values from multiple dimensions to create a sub-cube. In this query, we analyze the transactions with payment_type “AB” and device_os “windows” based on the customer age and source. The query counts the number of transactions and the number of fraudulent transactions for each combination of customer age and source, considering only transactions with both payment type “AB” and device os “windows”.

```
banking_transactions=# select * from payment_type_dice;
```

customer_age	source	num_transactions	num_fraud_transactions
10	INTERNET	1651	9
10	TELEAPP	2	0
20	INTERNET	22755	230
20	TELEAPP	15	0
30	INTERNET	28393	544
30	TELEAPP	23	1
40	INTERNET	28455	602
40	TELEAPP	43	1
50	INTERNET	23491	779
50	TELEAPP	21	1
60	INTERNET	6142	334
60	TELEAPP	12	1

(12 rows)

b) Dice by customer age and proposed credit limit

This query dices the data to show the number of transactions and fraudulent transactions by customer age and proposed credit limit. Specially for transactions made through the “INTERNET” source and with employment status “CA”

```
banking_transactions=# select * from fraud_by_age_and_proposed_credit;
```

customer_age	proposed_credit_group	num_transactions	num_fraud_transactions
10	High	1294	9
10	Low	14798	54
10	Medium	710	2
20	High	17713	210
20	Low	158026	733
20	Medium	11850	77
30	High	34920	760
30	Low	173643	1247
30	Medium	22721	185
40	High	32680	979
40	Low	119256	1225
40	Medium	17309	236
50	High	24566	1064
50	Low	55471	925
50	Medium	10391	192
60	High	3793	295
60	Low	9548	299
60	Medium	1528	55

(18 rows)

Combining OLAP operations

a) Roll up and Dice on customer income and age

This query creates income and age groups within the applicant dimension and combines them to analyze the data across different income and age segments. The results will provide insights on the number of transactions, the number of fraud transactions and the average values for the different velocities.
Roll up: The data is aggregated at a higher level by creating income and age groups, which comes from the original dataset.
Slice : The query segments the data by income and age groups which focuses on a specific subset of the data.

```
banking_transactions=# select * from fraud_by_income_and_age;
```

income_group	age_group	num_transactions	num_fraud_transactions	avg_name_email_similarity	avg_velocity_6h	avg_velocity_24h	avg_velocity_4w
High	Middle-aged	275571	3682	0.4763876936801599	5122.975703642318	4528.536660803435	4709.881286146359
High	Old	89443	2636	0.47775108889836204	5403.680042935066	4696.012237961892	4826.415295691649
High	Young	97306	702	0.5124472635235728	5510.880580955264	4716.333902126021	4806.6721673904485
Low	Middle-aged	132989	767	0.4955922877322936	5666.7184220158915	4845.492256805441	4940.610051511348
Low	Old	41162	564	0.4936649857466489	5963.735745661074	5036.081418852088	5041.232039287066
Low	Young	94483	267	0.5369209132982579	5987.2275927596	4962.37116529623	4968.3672694067345
Medium	Middle-aged	131703	939	0.48380959206052226	5463.784207862637	4726.471905114854	4845.502420688189
Medium	Old	41022	703	0.484562600390691	5733.800154730882	4915.3323521790935	4947.688415968863
Medium	Young	69389	285	0.5234134419070297	5787.84850451448	4868.736864217662	4901.15030030404

(9 rows)

b) Roll-up and Dice

This query creates a view that combines roll-up (customer age groups) and slice (device OS) operations to analyze transaction data. It groups the transactions by applicant age group, employment status and device os and calculates the number of transactions, the number of fraudulent transactions and the average values for name email similarity and the different velocities.

```
banking_transactions=# select * from age_group_and_employment_status;
```

age_group	employment_status	device_os	num_transactions	num_fraud_transactions	avg_name_email_similarity	avg_velocity_6h	avg_velocity_24h	avg_velocity_4w
Middle-aged	CA	windows	103297	2674	0.506433924798973	5158.367879751904	4564.61947155868	4763.5685254922755
Middle-aged	CB	windows	17329	234	0.47843073504783135	5640.9159933901765	4890.811492004134	4991.7200911955515
Middle-aged	CC	windows	1886	44	0.45731546187630034	5541.1548723923825	4762.238826963038	4875.49966927501
Middle-aged	CD	windows	3052	18	0.49079605614248395	5678.962398817357	4840.241047690078	4906.035093501143
Middle-aged	CE	windows	1893	12	0.5344073260030243	5557.840157157265	4710.865332051689	4794.746576357007
Middle-aged	CF	windows	4352	18	0.49609493442018704	5508.171214619408	4686.233746920689	4839.879478681333
Middle-aged	CG	windows	54	1	0.44818600758830945	5268.057275964034	4679.688657533909	4636.6202024112
Old	CA	windows	41262	1923	0.5096812773431435	5347.663720510034	4708.550834599446	4858.065654095939
Old	CB	windows	7541	156	0.4718262278463452	5901.779969164384	5007.740376090326	5060.692835566509
Old	CC	windows	10067	476	0.46446633954656147	5692.441318389392	4894.543917032721	4924.300054992354
Old	CD	windows	1531	19	0.49645262240109994	6107.8115193036665	5048.739807469836	5002.914246051141
Old	CE	windows	803	9	0.5346691154856188	5741.539547345206	4944.287797768758	4931.908415906208
Old	CF	windows	1806	11	0.49742541254457506	5775.186440916514	4807.49149419145	4904.123303525108
Old	CG	windows	31	3	0.4440751743974987	5056.855364531509	4678.718693765619	4635.447466524059
Young	CA	windows	49365	557	0.5445161156857662	5591.481715005607	4759.27915553064	4840.870521719393
Young	CB	windows	5504	34	0.5034222592783223	6161.540820779252	5126.08520247216	5160.983547037545
Young	CC	windows	416	6	0.4880712009132055	5993.37234368413	4862.830622004122	4910.849291165115
Young	CD	windows	1414	6	0.5171032488958148	6056.29777795497	5009.056010856299	4992.068100373701
Young	CE	windows	3514	15	0.5781679208758079	5935.663182942863	4899.445967214499	4854.160171044794
Young	CF	windows	1279	9	0.513302300941851	5967.085154068395	4890.014923883538	4913.15709421935
Young	CG	windows	25	0	0.5277174645001783	5702.76411176591	4822.630541215627	4939.118705724453

(21 rows)

c) Rollup and Slice

This query analyzes the total number of transactions and fraud transactions for each employment_status and device_os, where the payment type is 'AC'. First, the slice operation is applied to filter the dataset for the desired payment type 'AC', then we apply the rollup operation to aggregate data at different levels of the employment_status and device_os dimensions.

```
banking_transactions=# select * from employment_status_device_os_transactions_ac_payment;
 employment_status | device_os | total_transactions | total_fraud_transactions
-------------------+-----------+--------------------+--------------------------
                   |           |             246779 |                     4048
 CB                | linux     |               9215 |                       49
 CD                | macintosh |                253 |                        4
 CD                | windows   |               1952 |                       19
 CF                | other     |               4063 |                        7
 CF                | linux     |               6819 |                        9
 CC                | macintosh |                286 |                       19
 CC                | linux     |               3480 |                       49
 CE                | windows   |               1071 |                       13
 CF                | macintosh |                278 |                        3
 CC                | windows   |               3346 |                      207
 CG                | x11       |                  1 |                        0
 CC                | other     |               2057 |                       34
 CD                | linux     |               4222 |                        8
 CB                | other     |              10112 |                       72
 CG                | linux     |                 27 |                        1
 CE                | other     |               1584 |                        1
 CC                | x11       |                 72 |                        3
 CG                | macintosh |                  8 |                        1
 CB                | x11       |                150 |                        3
 CG                | windows   |                 22 |                        1
 CF                | x11       |                 87 |                        0
 CE                | linux     |               1295 |                        1
 CG                | other     |                 38 |                        1
 CD                | x11       |                 56 |                        0
 CA                | windows   |              46061 |                     1947
 CA                | x11       |               1124 |                       19
 CD                | other     |               2850 |                        7
 CE                | x11       |                 20 |                        0
 CA                | linux     |              63632 |                      526
 CB                | macintosh |               1430 |                       30
 CF                | windows   |               1937 |                        7
 CB                | windows   |               5041 |                      142
 CE                | macintosh |                462 |                        0
 CA                | macintosh |               9078 |                      239
 CA                | other     |              64650 |                      626
 CF                |           |              13184 |                       26
 CB                |           |              25948 |                      296
 CA                |           |             184545 |                     3357
 CC                |           |               9241 |                      312
 CE                |           |               4432 |                       15
 CD                |           |               9333 |                       38
 CG                |           |                 96 |                        4
(43 rows)
```

d) Drill-Down and Dice

This query is to analyze the average days since request for each employment_status, housing_status, and device_os, for customers who used the source 'INTERNET' and payment_type ('AB'). First, the Dice operation is applied to filter the dataset based on the desired source and payment_type. Then, the Drill-Down operation is applied to break down the data into detailed levels of employment_status, housing_status, and device_os.

```
banking_transactions=# select * from employment_housing_device_avg_days_internet_ab_payment;
 employment_status | housing_status | device_os | avg_days_since_request
-------------------+----------------+-----------+------------------------
 CA                | BA             | linux     |     0.2464936169347841
 CA                | BA             | macintosh |    0.09745578389587012
 CA                | BA             | other     |    0.22498819441800327
 CA                | BA             | windows   |    0.11262955312791796
 CA                | BA             | x11       |     0.4773190567520449
 CA                | BB             | linux     |    0.42748014688907365
 CA                | BB             | macintosh |     0.4944697970708965
 CA                | BB             | other     |      0.311367567703699
 CA                | BB             | windows   |    0.28041484932130406
 CA                | BB             | x11       |     0.7877697275497146
 CA                | BC             | linux     |     0.6575467472279073
 CA                | BC             | macintosh |     0.5090624167882737
 CA                | BC             | other     |      0.468592413788198
 CA                | BC             | windows   |    0.43671329901198447
 CA                | BC             | x11       |     0.9442157601082662
 CA                | BD             | linux     |     0.4721708893109467
 CA                | BD             | macintosh |     0.4178353950747432
 CA                | BD             | other     |    0.25472400427470443
 CA                | BD             | windows   |     0.4770914414042337
 CA                | BD             | x11       |     0.6710514836215015
 CA                | BE             | linux     |    0.29719815558480545
 CA                | BE             | macintosh |    0.23686628959657785
 CA                | BE             | other     |     0.2484479317844653
 CA                | BE             | windows   |     0.2262581918972045
 CA                | BE             | x11       |     0.3416274494596606
 CA                | BF             | linux     |     0.1232989046935462
 CA                | BF             | macintosh |    0.01638331740059516
 CA                | BF             | other     |    0.28319094470331446
 CA                | BF             | windows   |    0.49466640093906256
 CA                | BF             | x11       |     0.9222999892825455
```

Part A.2. Explorative operation

Iceberg queries

For the iceberg query, we find the five age groups with the highest number of fraudulent transactions. Here, we can see that the customer age groups 40, 50, 30, 20 and 60 (in descending order) have the most number of fraudulent transactions.

customer_age	num_fraud_transactions
40	2835
50	2766
30	2553
20	1181
60	1137

(5 rows)

Windowing queries

For the windowing query, we compared the number of fraudulent transactions with the average fraudulent transactions for each customer age group in the last 6 hours (where velocity_6h > 0)

customer_age	num_fraud_transactions	avg_fraud_transactions	fraud_rank
40	2835	1757.5000000000000000	1
50	2766	1757.5000000000000000	2
30	2553	1757.5000000000000000	3
20	1181	1757.5000000000000000	4
60	1137	1757.5000000000000000	5
10	73	1757.5000000000000000	6

(6 rows)

(Alternative approach)

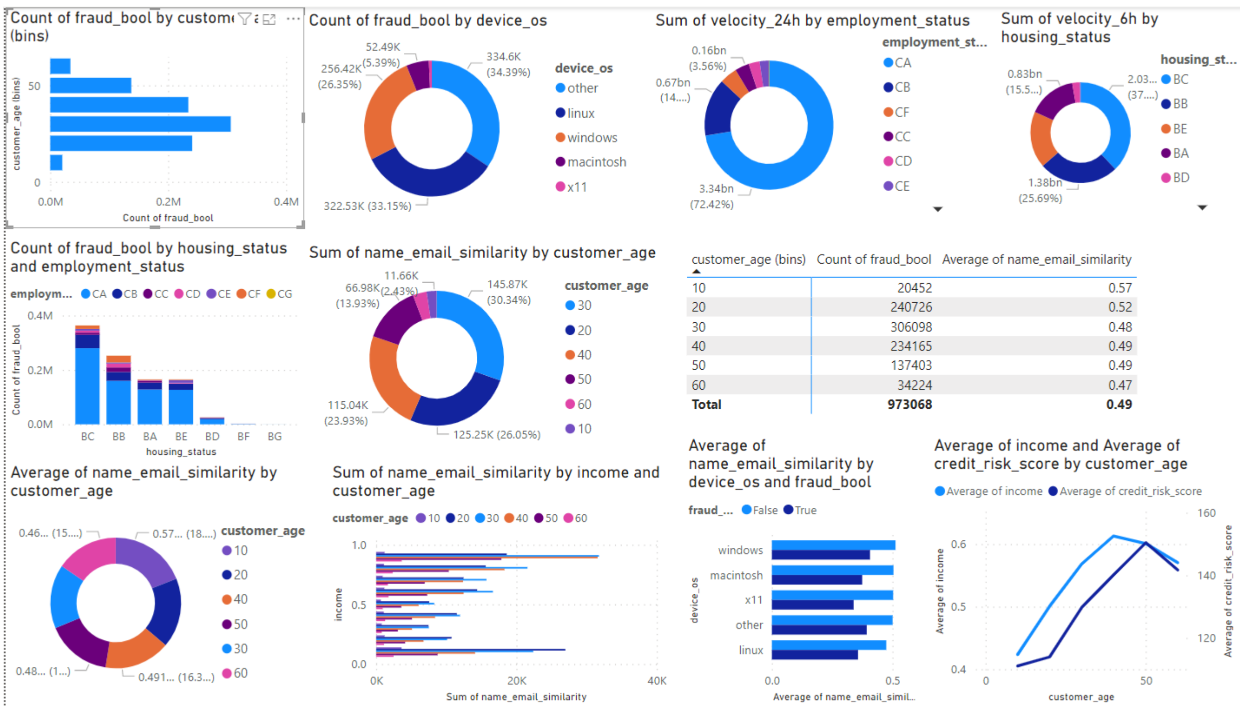
customer_age	num_fraud_transactions	num_transactions	avg_fraud_transactions_by_age	fraud_rank
40	2835	234153	0.012107468193873237	1
50	2766	137396	0.0201315904393141	2
30	2553	306081	0.008340929361835593	3
20	1181	240721	0.004906094607450119	4
60	1137	34222	0.0332242417158553	5
10	73	20452	0.003569333072560l407	6

(6 rows)

Using the Window clause

```
 customer_age | num_fraud_transactions | previous_age_fraud_transactions | next_age_fraud_transactions
--------------+------------------------+---------------------------------+----------------------------
           10 |                     73 |                                 |                        1181
           20 |                   1181 |                              73 |                        2553
           30 |                   2553 |                            1181 |                        2835
           40 |                   2835 |                            2553 |                        2766
           50 |                   2766 |                            2835 |                        1137
           60 |                   1137 |                            2766 |
(6 rows)
```

Part B. BI dashboard and Information Visualization



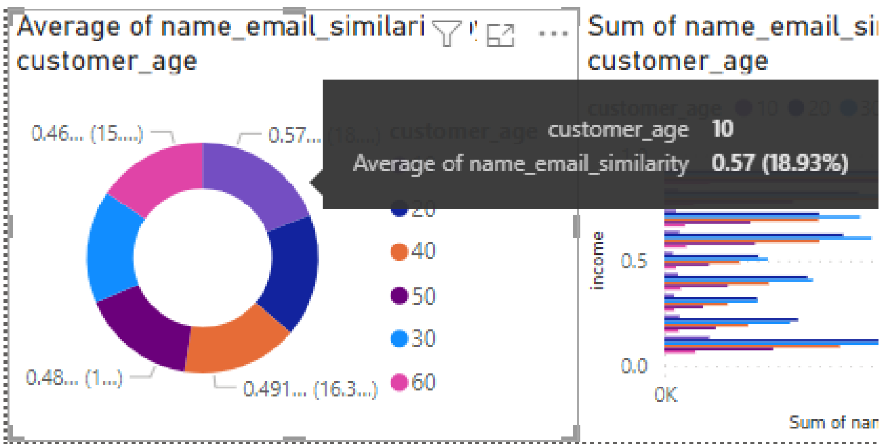
For the BI dashboard, we used donuts charts, stacked bar charts, line graphs, tables and stacked columns charts to visualize the data.

The visualization included :

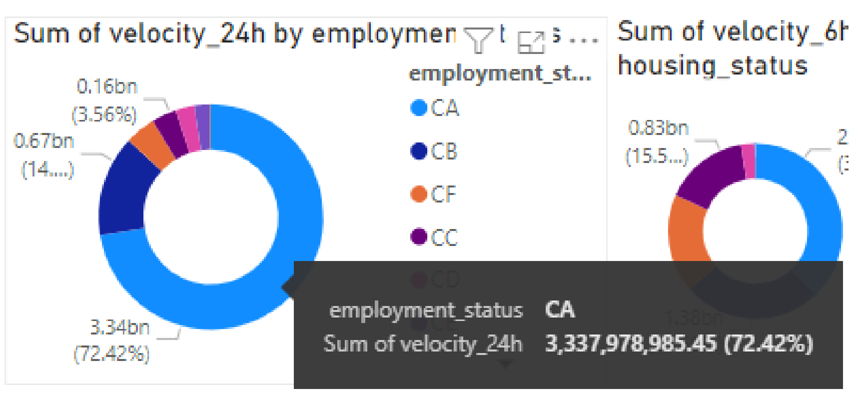
1. Count of fraud_bool by customer_age (in bins)
2. Count of fraud_bool by device_os
3. Sum of velocity_24h by employment_status
4. Sum of velocity_6h by housing_status
5. Count of fraud_bool by housing_status and employment status

6. Sum of name_email_similarity by customer_age
7. Table that shows the count of fraud_bool and average of name_email_similarity for each customer_age (in bins)
8. Average of name_email_similarity by customer_age
9. Sum of name_email_similarity by income and customer_age
10. Average of name_email_similarity by device os and fraud_bool
11. Average of income and Average of credit_risk_score by customer_age

From this visualization, we can see that, for example, the average of name_email_similarity is higher for the customer_age bin of 10 with a score of 0.57.



We can also see that the sum of velocity_24h is higher for customers with employment status "CA" which consists of 72.42% of the dataset.



Phase 4 - Data Mining

Part A. Data summarization, data preprocessing and feature selection

In order to effectively exploit the information contained within our dataset, we undertook a series of preprocessing steps to refine the raw data, address data quality issues, and conduct data summarization.

Regarding data summarization, we computed descriptive metrics encompassing both statistical measures and visualization techniques. These statistical measures included the mean, the mode, the standard deviation, the range, and the median.
Histograms were employed to visualize the distribution of each numerical attribute and identify anomalies like outliers within the data. Bar plots were also utilized to display the frequency distribution of categorical attributes, enabling the identification of prevalent categories and potential imbalances in the dataset. Additionally, pair plots were constructed for the attributes income, customer_age, credit_risk_score, intended_balcon_amount, and proposed_credit_limit, offering a rapid overview of their interrelationships and facilitating the exploration of patterns or correlations that may reveal trends and potential factors related to banking fraud.

Throughout the data preprocessing phase, we applied data cleaning and data transformation techniques. In terms of data cleaning, we pruned the dataset for missing values, null values, duplicate entries, and inconsistent data. Fortunately, our dataset was devoid of such issues. Subsequently, we conducted data transformation, converting certain attributes from int64 to boolean data types for more accurate representation within our system. We then engaged in feature engineering, merging the "phone_mobile_valid" and "home_phone_valid" columns to eliminate redundancy. Following this, we visually represented each column to identify outliers, assessing their relevance and determining whether to remove them. We also performed one-hot encoding for categorical attributes utilizing the OneHotEncoder package from the sklearn library. Numerical data was then normalized to ensure equal importance for each attribute during the learning process.
For feature selection, we initially removed several columns deemed irrelevant based on our exploration of visual representations executed in prior steps. We then employed sklearn packages, including ExtraTreesClassifier, SelectFromModel, and LinearSVC, to perform feature selection on the remaining attributes. These packages utilized tree-based and L1-based feature selection techniques.

Given the high quality nature of our dataset, we encountered minimal data quality issues. One such issue involved the data type of certain columns containing binary values (1 or 0), which were initially stored as int64 instead of boolean. As mentioned previously, we resolved this issue by converting the data type to boolean for the relevant columns. Additionally, the 'prev_address_months_count' column exhibited missing data (value = -1) in more than 71% of the dataset; consequently, we opted to remove these rows. We also merged the 'phone_mobile_valid' and 'phone_home_valid' columns into a single 'phone_valid' column using the OR operator. Finally, in relation to outliers present in the 'customer_age' and 'velocity_6h' columns, we decided to retain only the records where the customer age was less than 70 and the velocity value was below 13,000.

Part B. Classification (Supervised Learning)

Based on our first run:

Tree based feature selection

Classification Model	Accuracy	Precision	Recall	Time
Decision Trees	0.9769	False: 0.99 True: 0.05	False: 0.99 True: 0.06	16 seconds
Gradient Boosting	0.9887	False: 0.99 True:0.47	False: 1.00 True: 0.02	6m 29s
Random Forest Algorithms	0.9888	False: 0.99 True: 0.55	False: 1.00 True: 0.00	3m 5s

L1- based feature selection

Classification Model	Accuracy	Precision	Recall	Time
Decision Trees	0.9775	False: 0.99 True: 0.07	False: 0.99 True: 0.08	15s
Gradient Boosting	0.9887	False: 0.99 True: 0.47	False: 1.00 True: 0.02	6m 14s
Random Forest Algorithms	0.9888	False: 0.99 True: 0.55	False: 1.00 True: 0.00	3m 26s

A - Comparison of the results of the three learning algorithms:

(i) Accuracy:

Both Gradient Boosting and Random Forest Algorithms have very similar accuracies of around 0.9887 and 0.9888, respectively for both feature selection algorithms, which are slightly higher than the Decision Trees accuracy of 0.9769 (Tree-based) and 0.9775 (L1-based).

(ii) Precision:

For the False class, all models show similar precision values of approximately 0.99. However, when considering the True class, Random Forest Algorithms exhibit the highest precision (0.55) followed by Gradient Boosting (0.47) for both feature selection algorithms. Decision Trees have the lowest precision for the True class (0.05 for Tree-based and 0.07 for L1-based).

(iii) Recall:

For the False class, all models have similar recall values, close to 1.00. For the True class, Decision Trees have the highest recall (0.08 for L1-based and 0.06 for Tree-based), while Gradient Boosting and Random Forest Algorithms have lower recall values (0.02 and 0.00, respectively).

(iv) Time to construct the models:

From our experiment, we found out that the Decision Tree model is the fastest to construct, taking around 15 to 16 seconds. However, the Gradient Boosting has the longest construction time of over 6 minutes. As for the Random Forest Algorithm, its construction takes around 3 to 4 minutes.

B - Summary of actionable knowledge nuggets

Our team applied various data processing, data summarization, and feature selection techniques to the banking fraud dataset and trained multiple models with different algorithms. Based on the results obtained, we discovered several actionable knowledge nuggets that can help us better understand and mitigate potential banking fraud cases.

- Significance of certain features: The feature selection process highlighted key attributes, such as income, customer_age, credit_risk_score, intended_balcon_amount, and proposed_credit_limit, which played a crucial role in detecting potential fraud.

- Algorithm performance: The Gradient Boosting and Random Forest Algorithms achieved similar and better accuracies compared to Decision Trees. However, the recall for True cases was relatively low in all models, which can be an area for improvement in future iterations.

- Model construction time: Decision Trees were the fastest, but the trade-off was lower accuracy compared to Gradient Boosting and Random Forest Algorithms. Considering the critical nature of fraud detection, investing time in more accurate models might be worthwhile.

- Data quality: The high quality of the dataset allowed for minimal data cleaning, which positively impacted the overall model performance.

Regarding the fact that the recall for True cases was relatively low in all models, this could indicate that the models struggle to correctly identify positive instances in the dataset, which might be due to class imbalance or other issues. To address this, we believe that using techniques such as oversampling, undersampling, or using different performance metrics could be useful.

Overall, the insights obtained from the models can be used to develop more robust and accurate fraud detection systems. The choice of the algorithm should be made considering the trade-off between accuracy, recall, and model construction time. Additionally, it is crucial to focus on improving the recall of True cases to better capture actual fraud instances.

Part C. Detecting Outliers (Bonus)

We decided to detect the outliers using the One-class SVM algorithm available in the OneClassSVM package from the Sklearn library.
We noticed that the time it takes to run is considerable given the size of our dataset but we think the output should be correct.

References

https://www.postgresql.org/docs/current/tutorial-window.html
https://docs.aws.amazon.com/kinesisanalytics/latest/sqlref/sql-reference-window-clause.html
https://hevodata.com/learn/data-summarization-in-data-mining/#intro
https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html
https://scikit-learn.org/stable/modules/feature_selection.html
https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html
https://chat.openai.com/chat
https://scikit-learn.org/stable/modules/generated/sklearn.svm.OneClassSVM.html