Credit Card Transaction Analysis

EDULYT INDIA

NANDIPATI KUSUMA

Intern Id: EI-3431

June 8, 2025

1 Project Overview

This project involves performing structured analysis on credit card transaction data using MySQL to extract meaningful insights into customer behavior and spending patterns. The analysis is based on two datasets:

- **cb2**: Contains detailed credit card transaction records including transaction IDs, product information, pricing, merchants, payment methods, and return indicators.
- ci: Contains customer demographic details such as customer ID, age, gender, and state.

The goal is to clean, organize, and analyze this data to support banking and retail decision-making. Key areas of focus include:

- ✓ **Data Cleaning**: Identifying and removing duplicate transaction records while preserving one valid entry per transaction ID.
- ✓ Customer Segmentation: Classifying customers into groups such as Young Males, Mid-age Females, etc., based on age and gender.
- ✓ **Spending Analysis**: Aggregating total spend across product categories, states, and payment methods to identify high-value segments.
- ✓ **SQL-Based Exploration**: Executing complex queries using MySQL to join datasets, group data, apply filters, and calculate summaries.

This project demonstrates how structured querying with SQL can be used effectively to transform raw banking data into actionable insights, enabling better strategic decisions around customer engagement, product offerings, and fraud detection.

2 Logging into MySQL

To start working, log into MySQL using the following command:

mysql -u root -p

```
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 15
Server version: 8.0.35-0ubuntu0.23.04.1 (Ubuntu)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Figure 1: MYSQL setup

3 Creating two tables cb2 and ci:

1) Table for Credit_Banking Transaction Information(cb2):

```
CREATE TABLE cb2 (
   Credit_card VARCHAR(50),
   Product_ID VARCHAR(50),
   P_CATEGORY VARCHAR(100),
   CONDTION VARCHAR (100),
   Brand VARCHAR(100),
   Price DECIMAL(10,2),
   Selling_price DECIMAL(10,2),
   Coupon_ID VARCHAR(100),
   Date DATE,
   Time TIME,
   GTIN VARCHAR (100),
   MPN VARCHAR(100),
   Merchant_name VARCHAR(100),
   M_ID VARCHAR(50),
   'Payment Method' VARCHAR(50),
   'Transaction ID' VARCHAR(50),
   Return_ind VARCHAR(50),
   Return_date DATE
);
```

2) Table for Customer Information(ci):

```
CREATE TABLE ci (
   C_ID INT,
   Email VARCHAR(100),
   Name VARCHAR(20),
   Mobile_number BIGINT,
   Gender VARCHAR(1),
   Age VARCHAR(20),
   City VARCHAR(20),
   State VARCHAR(20),
   Address VARCHAR(20));
```

+ Field	Туре	+ Null	Key	Default	Extra
Credit_card	varchar(50)	YES		NULL	
Product ID	varchar(50)	YES İ		NULL	i i
P_CATEGORY	varchar(100)	YES	i	NULL	i i
CONDTION	varchar(100)	YES	į	NULL	i i
Brand	varchar(100)	YES	į	NULL	i i
Price	decimal(10,2)	YES	ĺ	NULL	i i
Selling_price	decimal(10,2)	YES		NULL	
Coupon_ID	varchar(100)	YES		NULL	l l
Date	date	YES		NULL	l l
Time	time	YES		NULL	l l
GTIN	varchar(100)	YES		NULL	l l
MPN	varchar(100)	YES		NULL	
Merchant_name	varchar(100)	YES		NULL	l l
M_ID	varchar(50)	YES		NULL	l l
Payment Method	varchar(50)	YES		NULL	<u> </u>
Transaction ID	varchar(50)	YES		NULL	l <u> </u>
Return_ind	varchar(50)	YES		NULL	
Return_date	date	YES		NULL	
+					

Figure 2: Description of table cb2

Figure 3: Description of table ci

4 Loading data from csv files

```
LOAD DATA INFILE '/var/lib/mysql-files/cb2.csv'
INTO TABLE cb2
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
(@Credit_card, @Product_ID, @P_CATEGORY, @CONDTION, @Brand, @Price,
    @Selling_price, @Coupon_ID, @Date, @Time, @GTIN, @MPN, @Merchant_name, @M_ID,
    @Payment_Method, @Transaction_ID, @Return_ind, @Return_date)
```

```
SET
   Credit_card = NULLIF(@Credit_card, ''),
   Product_ID = NULLIF(@Product_ID, ''),
   P_CATEGORY = NULLIF(@P_CATEGORY, ''),
   CONDTION = NULLIF(@CONDTION, ''),
   Brand = NULLIF(@Brand, ''),
   Price = NULLIF(@Price, ''),
   Selling_price = NULLIF(@Selling_price, ''),
   Coupon_ID = NULLIF(@Coupon_ID, ''),
   Date = NULLIF(@Date, ''),
   Time = NULLIF(@Time, ''),
   GTIN = NULLIF(@GTIN, ''),
   MPN = NULLIF(@MPN, ''),
   Merchant_name = NULLIF(@Merchant_name, ''),
   M_ID = NULLIF(@M_ID, ''),
   'Payment Method' = NULLIF(@Payment_Method, ''),
   'Transaction ID' = NULLIF(@Transaction_ID, ''),
   Return_ind = NULLIF(@Return_ind, ''),
   Return_date = NULLIF(@Return_date, '');
```

EXPLANATION:

- ✓ @colname: Loads data into a user variable to preprocess it
- ✓ NULLIF(@colname, "): Converts empty strings to SQL NULL

OUTPUT:

```
Query OK, 9999 rows affected, 1 warning (5.58 sec)
Records: 9999 Deleted: 0 Skipped: 0 Warnings: 1
```

5 Sanity checks -Data Cleaning

i) Provide a meaningful treatment where the Credit Card entries are blank.

```
UPDATE cb2
SET Credit_card = '0'
WHERE Credit_card = '';
```

OUTPUT:

```
mysql> UPDATE cb2 SET Credit_card = '0' WHERE Credit_card IS NULL;
Query OK, 5 rows affected (0.18 sec)
Rows matched: 5 Changed: 5 Warnings: 0
```

ii) Apply a 5% discount where Price = Selling_Price despite having a Coupon_Code.

```
mysql> UPDATE cb2 SET Selling_price = Selling_price * 0.95 WHERE Price = Selling
_price AND Coupon_ID IS NOT NULL;
Query OK, 96 rows affected, 94 warnings (0.26 sec)
Rows matched: 96 Changed: 96 Warnings: 94
```

iii).Ensure Return_date is after Purchase_date.

```
ALTER TABLE c
ADD CONSTRAINT check_Return_date
CHECK (Return_date > Dates);
```

iv) Ensure No disount is given ,if there is no Coupon_ID (i.e Selling_Price=Price)

```
mysql> update cb2 set Selling_price =Price where Coupon_ID IS NULL;
Query OK, 20 rows affected (0.27 sec)
Rows matched: 20 Changed: 20 Warnings: 0
```

v) Age should be greater than 18 for all the CC holders.

```
ALTER TABLE ci
ADD CONSTRAINT check_Age
CHECK (Age > 18);
```

Drop the rows where, Age column ,violates check_Age constraint

vi) Ensure unique Transaction ID:

```
ALTER TABLE cb2
ADD CONSTRAINT unique_t_id
UNIQUE ('Transaction ID');
```

NOTE: Drop the rows where transaction ids are duplicated.

- ✗ Dropping out all duplicate columns, by keeping only one unique Transaction_ID repectively.
- ✓ Remove duplicate rows based on all columns. Keep the unique ones for further processing.
 - Check all rows with a specific Transaction ID.
 - Find fully duplicate rows among those, and delete the
 - duplicates, keeping only one unique row per duplicate group.

Partitioning the similar rows and deleting by using "WINDOW FUNCTIONS"

```
ALTER TABLE cb2 ADD COLUMN id INT AUTO_INCREMENT PRIMARY KEY;
```

```
WITH CTE AS (
SELECT *,
     ROW_NUMBER() OVER (
     PARTITION BY Transaction_ID,.....*
     ORDER BY id
     ) AS rn
FROM cb2
)
SELECT * FROM CTE WHERE rn > 1;
```

```
WITH CTE AS (
SELECT *,
    ROW_NUMBER() OVER (
        PARTITION BY
        Credit_card, Product_ID, P_CATEGORY, CONDTION, Brand,
        Price, Selling_price, Coupon_ID, Date, Time, GTIN, MPN,
        Merchant_name, M_ID, 'Payment Method', 'Transaction ID',
        Return_ind, Return_date
        ORDER BY id
    ) AS rn
FROM cb2
)
DELETE FROM cb2
WHERE id IN (
    SELECT id FROM CTE WHERE rn > 1
);
```

6.TASKS

1. Customer segmentation based on Age, Gender:

```
SELECT
   Customer_ID,
   Age,
   Gender,
   CASE
   WHEN Gender IN ('F', 'Female') AND Age BETWEEN 18 AND 30 THEN 'Young Females'
   WHEN Gender IN ('F', 'Female') AND Age BETWEEN 31 AND 50 THEN 'Mid age
        Females'
   WHEN Gender IN ('F', 'Female') AND Age > 50 THEN 'Old Females'
   WHEN Gender IN ('M', 'Male') AND Age BETWEEN 18 AND 30 THEN 'Young Males'
   WHEN Gender IN ('M', 'Male') AND Age BETWEEN 31 AND 50 THEN 'Mid age Males'
   WHEN Gender IN ('M', 'Male') AND Age > 50 THEN 'Old Males'
   ELSE 'Other'
   END AS Customer_Segment
FROM cb2;
```

OUTPUT:

```
2200
                         Old Females
 9174 i
                         Old Males
        80
 7589 |
                         Mid age Females
        33
                         Mid age Females
 1779 |
 7576
        78
              1 F
                         Old Females
 7735
        39
                F
                         Mid age Females
                         Young Males
 3705 I
        29
 9029
        50
                         Mid age Females
 5260
        89
                         Old Females
 3942
        26
                         Young Males
                         Old Females
 5229
        60
 4672 I
                         Old Males
        53
 7496
                         Old Females
 6042
                         Old Males
        88
 4575 |
        56
                         Old Females
 3958
        87
                         Old Males
 8609 |
        93
                         Old Females
 4542
        60
                         Old Females
 9532
        18
              M
                         Young Males
187 rows in set (0.00 sec)
```

2. Calculate the spend in terms of Product, State and Payment method.

```
SELECT
    'P_CATEGORY',
    ci.State,
    'Payment Method',
    SUM(Price) AS TotalSpend
FROM cb2
JOIN ci ON ci.C_ID = cb2.Credit_card
GROUP BY P_CATEGORY, ci.State, 'Payment Method';
```

+	+	+	++
P CATEGORY	State	Payment Method	TotalSpend
+	+	+	
SHOES	Illinois	Credit card	8904.83
DECOR	Illinois	Debit card	1023.49
GAMES	Nevada	Mobile carrier Billing	2421.99
KITCHEN & DINING	Missouri	Credit card	10303.33
GAMES	California	Mobile carrier Billing	33375.59
ELECTRONICS	California	Credit card	50651.34
OFFICE SUPPLIES	Massachusetts	Credit card	10143.63
KITCHEN & DINING	California	Credit card	29751.47
OFFICE SUPPLIES	Washington	Credit card	11256.45
APPLIANCES	Texas	Debit card	4447.54
CLOTHING	Massachusetts	Mobile carrier Billing	7390.20
SHOES	Massachusetts	Credit card	11205.70
APPLIANCES	California	Prepaid card	9402.81
KITCHEN & DINING	Texas	Mobile carrier Billing	17966.08
BABY CLOTHING	California	Mobile carrier Billing	18692.64
COMPUTERS	California	Credit card	38536.42

3. Calculate the highest 5 spending in all above categories.

```
SELECT
P_CATEGORY,
ci.State,
'Payment Method',
SUM(Price) AS TotalSpend
FROM cb2
JOIN ci ON ci.C_ID = cb2.Credit_card
GROUP BY P_CATEGORY, ci.State, 'Payment Method'
ORDER BY TotalSpend DESC
LIMIT 5;
```

P_CATEGORY	State	+ Payment Method +	TotalSpend
DECOR SHOES ELECTRONICS DECOR OFFICE SUPPLIES	California California California California California	Mobile carrier Billing Credit card Credit card Credit card	57608.35 53867.46 50651.34 46905.49 45803.56
5 rows in set (0.1		+	

- 4. Return Analysis
- 1) Return Analysis Based on "Return_ind = 1"

```
SELECT ci.State, COUNT(*) AS ReturnCount
FROM cb2
JOIN ci ON cb2.Credit_card = ci.C_ID
WHERE cb2.Return_ind = '1'
GROUP BY ci.State
ORDER BY ReturnCount DESC;
```

```
State
              | ReturnCount
 California
                        34
 Texas
                        17
 Kentucky
                       12
 Massachusetts |
                       10
 Washington
                        8
 Missouri
                        6
 Arizona
                        6
                         5
 Nevada
 Illinois
                         5
 Ohio
10 rows in set (0.01 sec)
```

2) return by age group

```
SELECT

CASE

WHEN ci.Age <= 18 THEN 'Under 18'

WHEN ci.Age <= 25 THEN '18-25'

WHEN ci.Age <= 35 THEN '26-35'

ELSE 'Over 35'

END AS AgeGroup,

COUNT(*) AS ReturnCount

FROM cb2

JOIN ci ON cb2.Credit_card = ci.C_ID

WHERE cb2.Return_ind = '1'

GROUP BY AgeGroup

ORDER BY ReturnCount DESC;
```

3.return by condition

```
SELECT CONDTION, COUNT(*) AS ReturnCount
FROM cb2
WHERE Return_ind = '1'
GROUP BY CONDTION
ORDER BY ReturnCount DESC;
```

4. return by product category

```
SELECT P_CATEGORY AS ProductCategory, COUNT(*) AS ReturnCount
FROM cb2
WHERE Return_ind = '1'
GROUP BY P_CATEGORY
ORDER BY ReturnCount DESC;
```

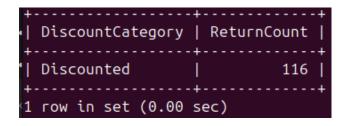
```
ProductCategory | ReturnCount
 SHOES
                              14
 COMPUTERS
                              14
                              13 |
 BEDDING
 KITCHEN & DINING |
                              12
 APPLIANCES
                              11
 ELECTRONICS
                              10
 CLOTHING
                              10
 DECOR
                              9 |
 OFFICE SUPPLIES
                               8
 BABY CLOTHING
                               4
 GAMES
                               4
 BABY TOYS
                               4
 LUGGAGE
                               3 |
13 rows in set (0.00 sec)
```

5. return by discount.

```
SELECT
  CASE
    WHEN Price > Selling_price THEN 'Discounted'
    ELSE 'Non-Discounted'
  END AS DiscountCategory,
  COUNT(*) AS ReturnCount
FROM cb2
WHERE Return_ind = '1'
```

```
GROUP BY DiscountCategory
ORDER BY ReturnCount DESC;
```

OUTPUT:



OPINION ON RETURN PATTERN

1. BY STATE:

California has higher return count of 34.

2. BY AgeGroup:

Over 35 are tend to return more.

3. By Product Condition:

Instead of used and refurbished products, new conditioned products are returned more.

4. By Product Category:

SHOES and COMPUTERS are returned more compared with other products.

5. BY Discount:

Discounted products often have more returns. This is usually because people buy them impulsively or expect less quality.

5) Create a profile of customers in terms of timing of their order.

```
SELECT DAYOFWEEK(Date) AS DayOfWeek, COUNT(*) AS OrderCount
FROM cb2
GROUP BY DayOfWeek ORDER BY DayOfWeek;
SELECT HOUR(Time) AS HourOfDay, COUNT(*) AS OrderCount
FROM cb2
GROUP BY HourOfDay ORDER BY HourOfDay;
SELECT MONTH(Date) AS Month, AVG(Price) AS AverageOrderValue
FROM cb2
GROUP BY Month ORDER BY Month;
```

OUTPUT:

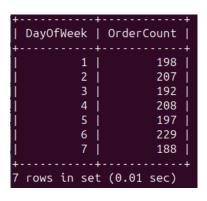


Figure 4: 1

+ Month	++ AverageOrderValue
1 2 3 4 5 6 7 8 9	2696.128773 2660.304104 2658.781307 2394.531959 2740.166218 2602.352821 2433.498750 2883.386022 2749.819886
10 11 12	2749.813666 2764.948796 2703.718288 2569.368614

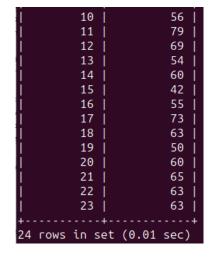


Figure 5: 2

Figure 6: 3

Which payment method is providing more discount for customers?

```
SELECT Payment_Method, SUM(Price - Selling_price) AS TotalDiscount
FROM cb2
GROUP BY Payment_Method
ORDER BY TotalDiscount DESC
LIMIT 1;
```

OUTPUT:

7) Create a profile for high value items vs low value items and relate that wrt to their number of orders.

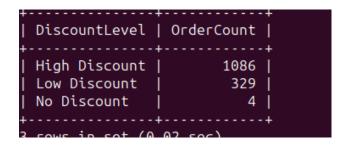
```
SELECT
  CASE
    WHEN Price >= 100 THEN 'High Value'
    ELSE 'Low Value'
  END AS ItemCategory,
  COUNT(DISTINCT 'Transaction ID') AS OrderCount
FROM
  cb2
GROUP BY
  ItemCategory;
```

OUTPUT:

8) Does offering higher discounts lead to an increase in order volume?

```
SELECT
   CASE
   WHEN (Price - Selling_price) >= 20 THEN 'High Discount'
   WHEN (Price - Selling_price) > 0 THEN 'Low Discount'
   ELSE 'No Discount'
   END AS DiscountLevel,
   COUNT(DISTINCT 'Transaction ID') AS OrderCount
FROM cb2
GROUP BY DiscountLevel
ORDER BY OrderCount DESC;
```

OUTPUT:



Yes, higher discounts can lead to an increase in the number of orders.