Data Warehouses and Data Analysis Techniques
PMS 2024
Γεώργιος Βώβος
Αριθμός μητρώου: M013124003

Περιεχόμενα

Εύρεση και κατανόηση του χώρου εφαρμογής της ΑΔ. (5%)

Σε αυτή την εργασία, θα ασχοληθούμε με δεδομένα οικονομικών συναλλαγών (Online, On site, etc)

Εύρεση και κατανόηση των δεδομένων που θα αποθηκευτούν στην ΑΔ. (10%)

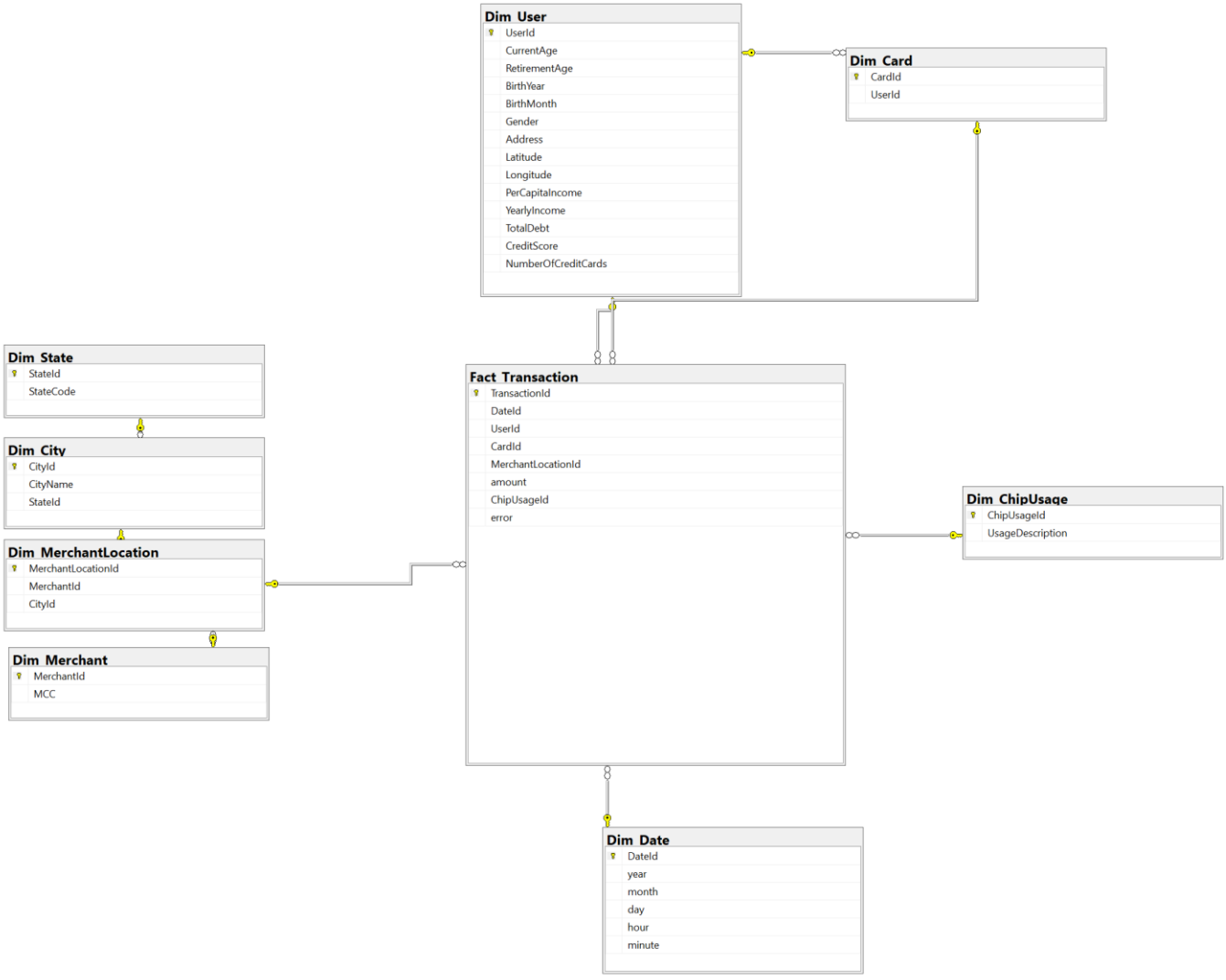
Τα δεδομένα βρίσκονται διαθέσιμα στο https://www.kaggle.com/datasets/computingvictor/transactions-fraud-datasets/data .
Αυτό το σύνολο δεδομένων συνδυάζει αρχεία συναλλαγών, πληροφορίες πελατών και δεδομένα καρτών από ένα τραπεζικό ίδρυμα, που εκτείνονται σε όλη τη δεκαετία του 2010. Το σύνολο δεδομένων έχει σχεδιαστεί για πολλαπλούς αναλυτικούς σκοπούς, συμπεριλαμβανομένου του εντοπισμού απάτης, της ανάλυσης συμπεριφοράς πελατών και της πρόβλεψης εξόδων.

Τα αποτελέσματα των ερωτημάτων και τα διαγράμματα βασίζονται στο αρχικό dataset των 1.3GB
Στην εργασία, υπάρχει συννημένο είναι μικρό υποσύνολο 60.000 εγγραγών , περίπου 6MB.

2000 users.
199 states.
18638 cities.
74831 merchants.
197205 merchant locations.
4136496 date records.
4071 cards.
3 chip usage types.
13305915 transactions.

Σχεδίαση της ΑΔ. (20%)

Για την αποθήκη δεδομένων μας, θα χρησιμοποιήσουμε την [τοπολογία Αστέρα](#).



Υλοποίηση της ΑΔ σε ένα από γνωστά συστήματα διαχείρισης βάσεων δεδομένων (SQL Server, MySQL, Postgres, Oracle). (20%)

Η βάση μας είναι υλοποιημένη σε Microsoft SQL Server 2022 και μπορεί να δημιουργηθεί εκτελώντας το αρχείο 01CreateDataWarehouse.sql

Για περισσότερες οδηγίες δείτε το README.MD

```
CREATE TABLE Dim_User (
    UserId INT PRIMARY KEY,
    CurrentAge INT,
    RetirementAge INT,
    BirthYear INT,
    BirthMonth INT,
    Gender VARCHAR(10),
    Address VARCHAR(100),
    Latitude DECIMAL(9,6),
    Longitude DECIMAL(9,6),
    PerCapitaIncome DECIMAL(10,2),
    YearlyIncome DECIMAL(10,2),
    TotalDebt DECIMAL(10,2),
    CreditScore INT,
    NumberOfCreditCards INT
);
GO

CREATE TABLE Dim_State (
    StateId INT IDENTITY(1,1) PRIMARY KEY,
    StateCode VARCHAR(50) UNIQUE
);
GO

CREATE TABLE Dim_City (
    CityId INT IDENTITY(1,1) PRIMARY KEY,
    CityName VARCHAR(100),
    StateId INT,
    CONSTRAINT FK_city_state FOREIGN KEY (StateId) REFERENCES Dim_State(StateId)
);
GO

CREATE TABLE Dim_Merchant (
    MerchantId INT PRIMARY KEY,
    MCC INT
);
GO
```

```
CREATE TABLE Dim_MerchantLocation (
    MerchantLocationId INT IDENTITY(1,1) PRIMARY KEY,
    MerchantId INT,
    CityId INT,
    CONSTRAINT FK_merchant_location_merchant FOREIGN KEY (MerchantId) REFERENCES
Dim_Merchant(MerchantId),
    CONSTRAINT FK_merchant_location_city FOREIGN KEY (CityId) REFERENCES
Dim_City(CityId),
    CONSTRAINT UQ_merchant_city UNIQUE (MerchantId, CityId)
);
GO

CREATE TABLE Dim_Date (
    DateId DATETIME PRIMARY KEY,
    year INT,
    month INT,
    day INT,
    hour INT,
    minute INT
);
GO


CREATE TABLE Dim_ChipUsage (
    ChipUsageId INT IDENTITY(1,1) PRIMARY KEY,
    UsageDescription VARCHAR(50) UNIQUE
);
GO

CREATE TABLE Dim_Card (
    CardId INT PRIMARY KEY,
    UserId INT,
    CONSTRAINT FK_dim_card_user FOREIGN KEY (UserId) REFERENCES Dim_User(UserId)
);
GO
```

```
CREATE TABLE Fact_Transaction (
    TransactionId INT PRIMARY KEY,
    DateId DATETIME,
    UserId INT,
    CardId INT,
    MerchantLocationId INT,
    amount DECIMAL(10,2),
    ChipUsageId INT,
    error VARCHAR(100),
    CONSTRAINT FK_fact_transaction_date FOREIGN KEY (DateId) REFERENCES
Dim_Date(DateId),
    CONSTRAINT FK_fact_transaction_user FOREIGN KEY (UserId) REFERENCES
Dim_User(UserId),
    CONSTRAINT FK_fact_transaction_card FOREIGN KEY (CardId) REFERENCES
Dim_Card(CardId),
    CONSTRAINT FK_fact_transaction_merchant_location FOREIGN KEY
(MerchantLocationId) REFERENCES Dim_MerchantLocation(MerchantLocationId),
    CONSTRAINT FK_fact_transaction_chip_usage FOREIGN KEY (ChipUsageId) REFERENCES
Dim_ChipUsage(ChipUsageId)
);
GO
```

Διαδικασία ETL (extract, transform, load) δηλ. «εξαγωγή, μετασχηματισμός και φόρτωση». Μαζί αυτές οι δραστηριότητες διαμορφώνουν τη διαδικασία που χρησιμοποιείται για τη λήψη δεδομένων από την πηγή και τη μετατροπή τους σε μία χρησιμοποιήσιμη μορφή και έπειτα μετακίνηση στην ΑΔ. (15%)

Έχουμε υλοποιήσει 2 stored procedures που διαβάζουν τα 2 .csv datasets (users, transactions) και εισάγουν τα δεδομένα στους πίνακες.
Είναι αρκετά μεγάλες, εκτελέστε το αρχείο **02CreateStoredProcs.sql**

Μπορούμε να τις τρέξουμε manually **03ExecureStoredProcsToLoadData.sql**

```
EXEC sp_load_user_data
    @dataFile = 'C:\dev\UTH\UTH-WAREHOUSE\users_data.csv',
    @formatFile = 'C:\dev\UTH\UTH-WAREHOUSE\users_format.fmt',
    @errorFile = 'C:\dev\UTH\UTH-WAREHOUSE\users_errors.txt';


EXEC sp_load_transaction_data
    @dataFile = 'C:\dev\UTH\UTH-WAREHOUSE\transactions_data_full.csv',
    @errorFile = 'C:\dev\UTH\UTH-WAREHOUSE\transactions_errors.txt';
```

Είτε να φτιάξουμε ένα job το οποίο τρέχει κάθε βράδυ και εισάγει τα δεδομένα της ημέρες όπως στο **05CreateETLJob.sql**

Διατύπωση (τουλάχιστον 6) ερωτημάτων SQL που θα χρησιμοποιούν συγκεντρωτικές συναρτήσεις και συνδυασμό πινάκων. (15%)

```sql
-- 1. Top 10 Users by Total Transaction Amount
-- Identifies your highest value customers
SELECT TOP 10
    u.UserId,
    COUNT(ft.TransactionId) AS TransactionCount,
    SUM(ft.amount) AS TotalAmount
FROM Dim_User u
JOIN Fact_Transaction ft ON u.UserId = ft.UserId
GROUP BY u.UserId
ORDER BY TotalAmount DESC;


-- 2. Transaction Volume by Month
-- Helps identify seasonal patterns in spending
SELECT
    d.year as Year,
    d.month as Month,
    COUNT(ft.TransactionId) AS TransactionCount,
    SUM(ft.amount) AS TotalAmount
FROM Fact_Transaction ft
JOIN Dim_Date d ON ft.DateId = d.DateId
--WHERE YEAR= 2018
GROUP BY d.year, d.month
ORDER BY d.year, d.month;


-- 3. Merchant Categories with Highest Error Rates
-- Helps identify potential technical issues with specific merchant types
SELECT
    m.MCC,
    COUNT(ft.TransactionId) AS TotalTransactions,
    SUM(CASE WHEN ft.error IS NOT NULL AND ft.error != '' THEN 1 ELSE 0 END) AS
ErrorCount,
    (SUM(CASE WHEN ft.error IS NOT NULL AND ft.error != '' THEN 1 ELSE 0 END) *
100.0 / COUNT(ft.TransactionId)) AS ErrorPercentage
FROM Fact_Transaction ft
JOIN Dim_MerchantLocation ml ON ft.MerchantLocationId = ml.MerchantLocationId
JOIN Dim_Merchant m ON ml.MerchantId = m.MerchantId
GROUP BY m.MCC
HAVING COUNT(ft.TransactionId) > 100
ORDER BY ErrorPercentage DESC;
```

```sql
-- 4. Average Transaction Amount by Age Group
-- Helps understand spending patterns across different demographics
SELECT
    CASE
        WHEN u.CurrentAge < 25 THEN 'Under 25'
        WHEN u.CurrentAge BETWEEN 25 AND 34 THEN '25-34'
        WHEN u.CurrentAge BETWEEN 35 AND 44 THEN '35-44'
        WHEN u.CurrentAge BETWEEN 45 AND 54 THEN '45-54'
        WHEN u.CurrentAge BETWEEN 55 AND 64 THEN '55-64'
        ELSE '65 and older'
    END AS AgeGroup,
    COUNT(DISTINCT u.UserId) AS UserCount,
    COUNT(ft.TransactionId) AS TransactionCount,
    AVG(ft.amount) AS AvgTransactionAmount,
    SUM(ft.amount) / COUNT(DISTINCT u.UserId) AS AvgSpendPerUser
FROM Dim_User u
JOIN Fact_Transaction ft ON u.UserId = ft.UserId
GROUP BY
    CASE
        WHEN u.CurrentAge < 25 THEN 'Under 25'
        WHEN u.CurrentAge BETWEEN 25 AND 34 THEN '25-34'
        WHEN u.CurrentAge BETWEEN 35 AND 44 THEN '35-44'
        WHEN u.CurrentAge BETWEEN 45 AND 54 THEN '45-54'
        WHEN u.CurrentAge BETWEEN 55 AND 64 THEN '55-64'
        ELSE '65 and older'
    END
ORDER BY AgeGroup;

-- 5. Chip Usage Analysis
-- Tracks adoption of different card technologies
SELECT
    cu.UsageDescription,
    COUNT(ft.TransactionId) AS TransactionCount,
    SUM(ft.amount) AS TotalAmount,
    AVG(ft.amount) AS AvgAmount
FROM Fact_Transaction ft
JOIN Dim_ChipUsage cu ON ft.ChipUsageId = cu.ChipUsageId
GROUP BY cu.UsageDescription
ORDER BY TransactionCount DESC;
```

```sql
-- 6. Top Cities by Transaction Volume
-- Identifies geographic hotspots for transactions
SELECT TOP 20
    c.CityName,
    s.StateCode,
    COUNT(ft.TransactionId) AS TransactionCount,
    SUM(ft.amount) AS TotalAmount,
    COUNT(DISTINCT ft.UserId) AS UniqueUsers
FROM Fact_Transaction ft
JOIN Dim_MerchantLocation ml ON ft.MerchantLocationId = ml.MerchantLocationId
JOIN Dim_City c ON ml.CityId = c.CityId
JOIN Dim_State s ON c.StateId = s.StateId
GROUP BY c.CityName, s.StateCode
ORDER BY TotalAmount DESC;

-- 7. Credit Score Impact on Spending Patterns
-- Analyzes how credit scores correlate with spending
SELECT
    CASE
        WHEN u.CreditScore < 580 THEN 'Poor (< 580)'
        WHEN u.CreditScore BETWEEN 580 AND 669 THEN 'Fair (580-669)'
        WHEN u.CreditScore BETWEEN 670 AND 739 THEN 'Good (670-739)'
        WHEN u.CreditScore BETWEEN 740 AND 799 THEN 'Very Good (740-799)'
        ELSE 'Excellent (800+)'
    END AS CreditScoreRange,
    COUNT(DISTINCT u.UserId) AS NumberOfUsers,
    AVG(u.YearlyIncome) AS AvgIncome,
    AVG(u.TotalDebt) AS AvgDebt,
    COUNT(ft.TransactionId) / COUNT(DISTINCT u.UserId) AS AvgTransactionsPerUser,
    SUM(ft.amount) / COUNT(DISTINCT u.UserId) AS AvgSpendPerUser
FROM Dim_User u
JOIN Fact_Transaction ft ON u.UserId = ft.UserId
GROUP BY
    CASE
        WHEN u.CreditScore < 580 THEN 'Poor (< 580)'
        WHEN u.CreditScore BETWEEN 580 AND 669 THEN 'Fair (580-669)'
        WHEN u.CreditScore BETWEEN 670 AND 739 THEN 'Good (670-739)'
        WHEN u.CreditScore BETWEEN 740 AND 799 THEN 'Very Good (740-799)'
        ELSE 'Excellent (800+)'
    END
ORDER BY AvgSpendPerUser DESC;
```

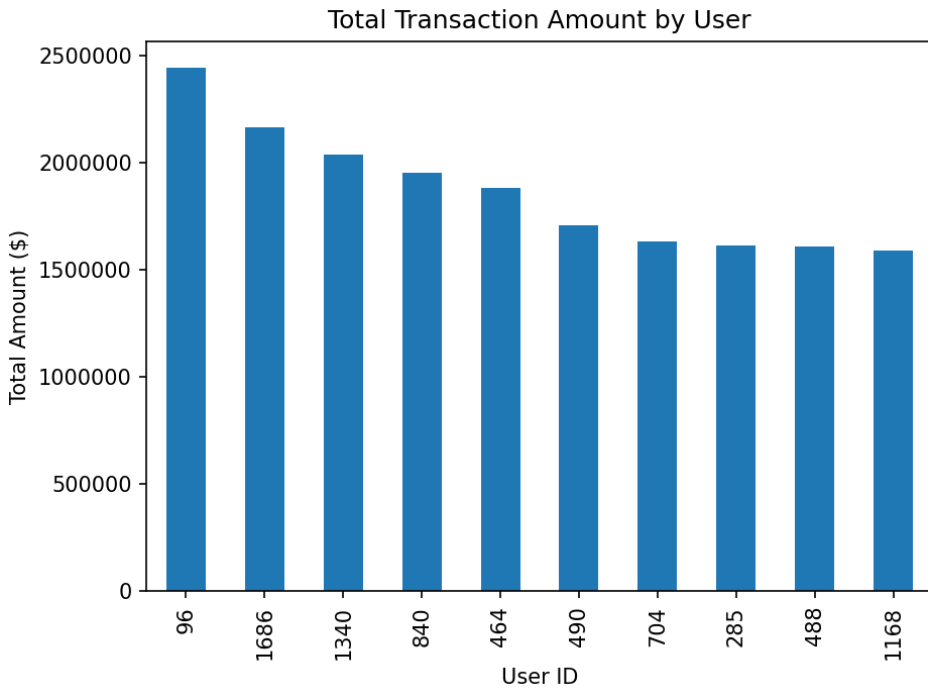
```sql
-- 8. Hourly Transaction Patterns
-- Identifies peak transaction times throughout the day
SELECT
    d.hour as Hour,
    COUNT(ft.TransactionId) AS TransactionCount,
    SUM(ft.amount) AS TotalAmount,
    AVG(ft.amount) AS AvgTransactionAmount
FROM Fact_Transaction ft
JOIN Dim_Date d ON ft.DateId = d.DateId
GROUP BY d.hour
ORDER BY d.hour;
```

Εκτέλεση των ερωτημάτων και εμφάνιση αποτελεσμάτων με οπτικοποίηση. (10%)

1. **Top 10 Users by Total Transaction Amount**

Results | Messages

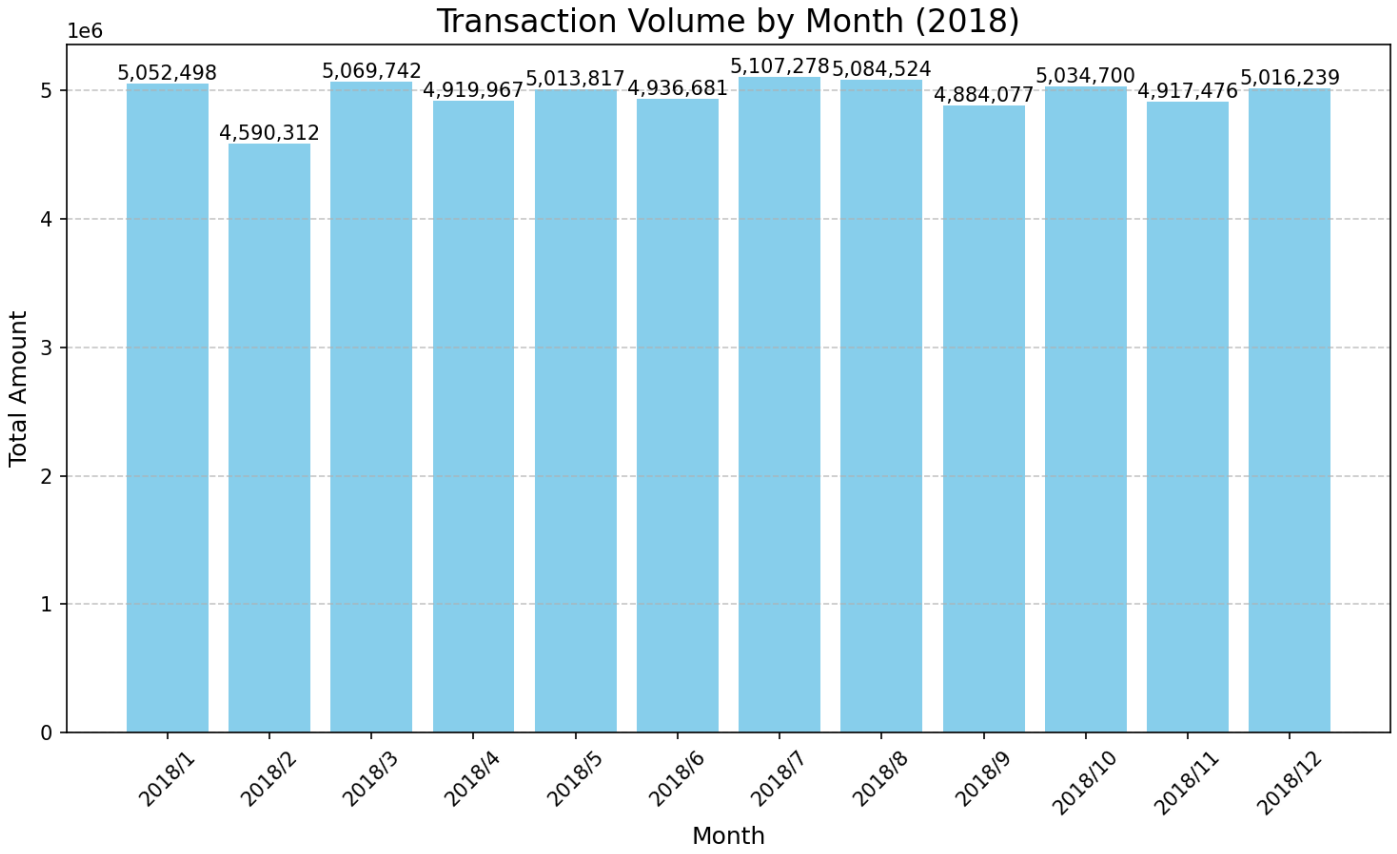
	UserId	TransactionCount	TotalAmount
1	96	38617	2445773.25
2	1686	19810	2167880.90
3	1340	22023	2039921.23
4	840	15095	1956340.84
5	464	27619	1882901.35
6	490	21831	1711482.69
7	704	20748	1635022.05
8	285	32032	1615458.99
9	488	23990	1611114.42
10	1168	30520	1590822.75



2 Transaction Volume by Month (2018)

Results | Messages

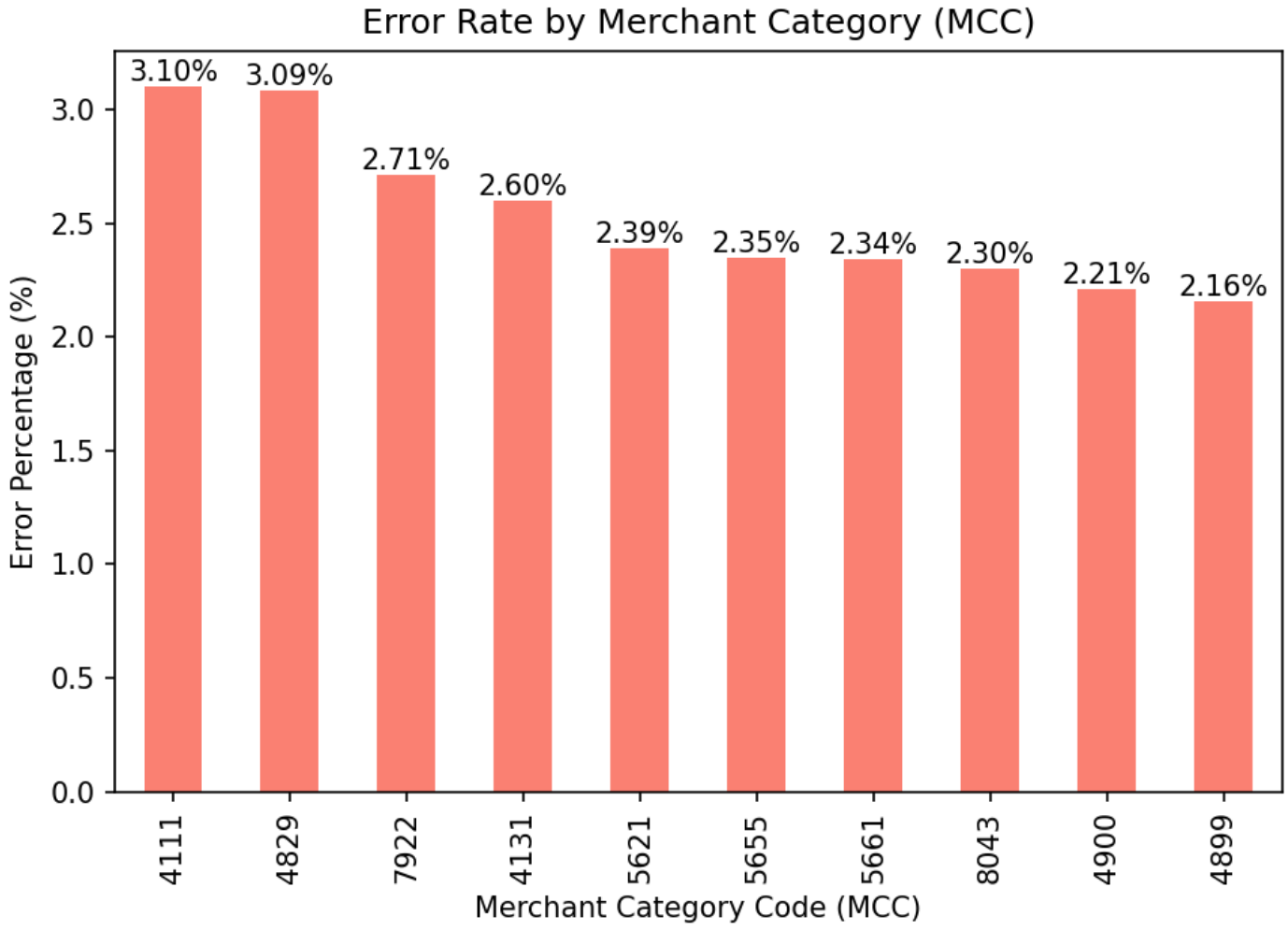
	Year	Month	TotalAmount
1	2018	1	5052498.81
2	2018	2	4590312.41
3	2018	3	5069742.21
4	2018	4	4919967.23
5	2018	5	5013817.88
6	2018	6	4936681.74
7	2018	7	5107278.77
8	2018	8	5084524.65
9	2018	9	4884077.23
10	2018	10	5034700.64
11	2018	11	4917476.53
12	2018	12	5016239.84



3. Merchant Categories with Highest Error Rates

Results | Messages

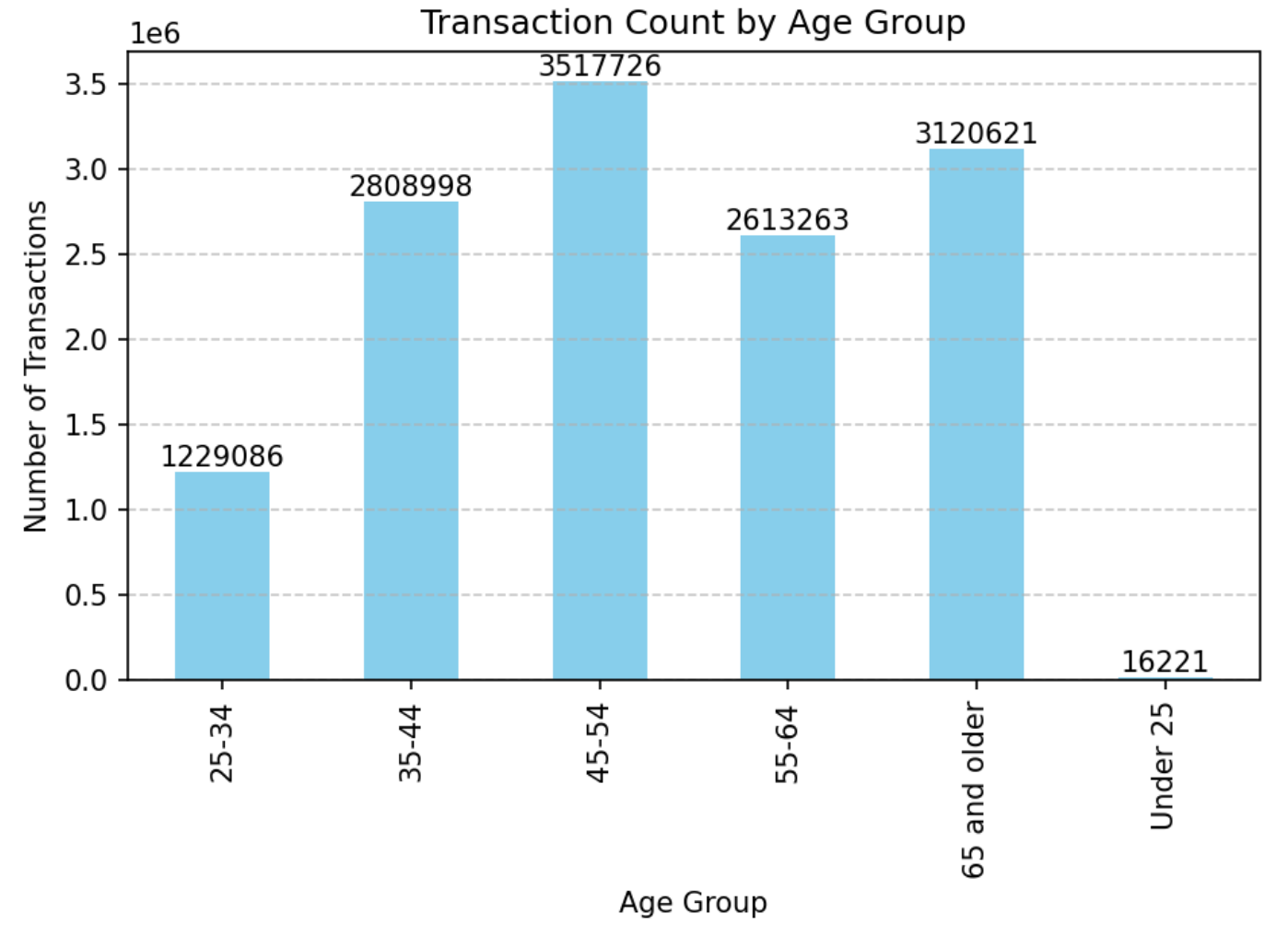
	MCC	ErrorPercentage
1	4111	3.102074787468
2	4829	3.086118503277
3	7922	2.713218510064
4	4131	2.601021830004
5	5621	2.389509470616
6	5655	2.350879313470
7	5661	2.343072027769
8	8043	2.302869544248
9	4900	2.210797540047
10	4899	2.158273381294



4. Transaction Count(and more) by Age Group

Results | Messages

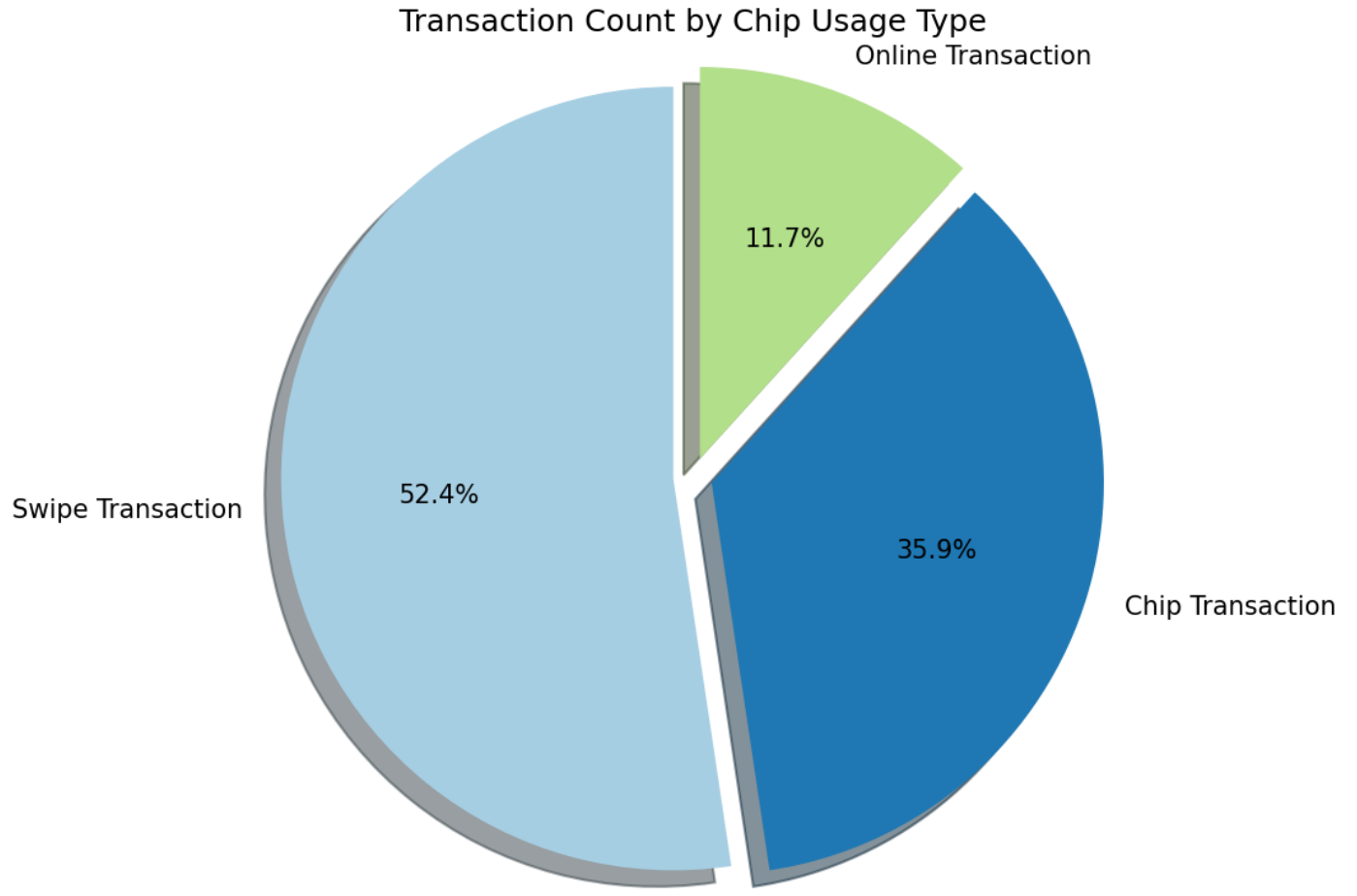
	AgeGroup	UserCount	TransactionCount	AvgTransactionAmount	AvgSpendPerUser
1	25-34	140	1229086	47.076975	413297.508928
2	35-44	257	2808998	40.786907	445798.990778
3	45-54	308	3517726	43.486432	496666.738116
4	55-64	241	2613263	42.537147	461247.937966
5	65 and older	269	3120621	43.098921	499982.896059
6	Under 25	4	16221	47.717339	193505.742500



5. Chip Usage Analysis

Results | Messages

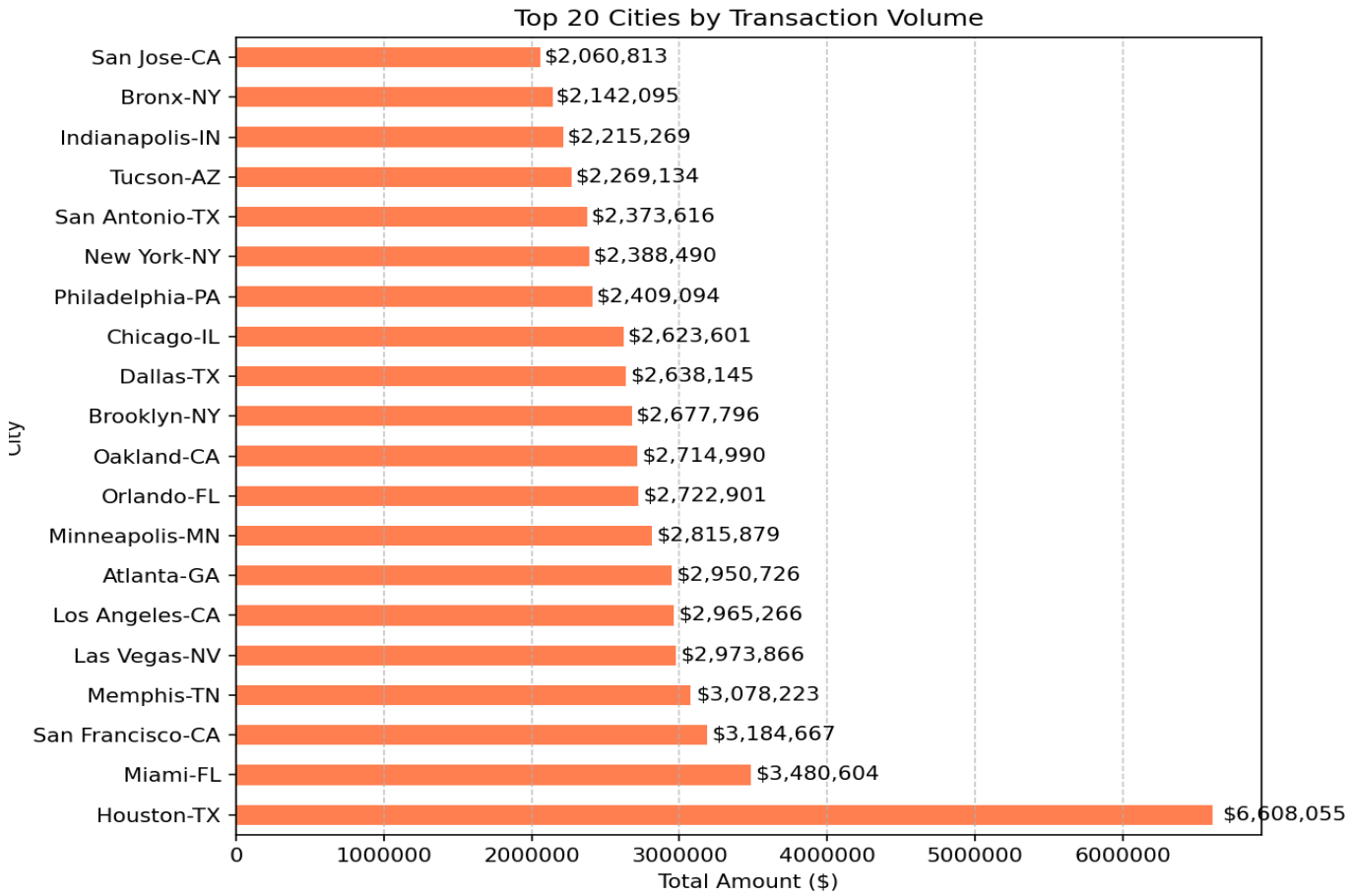
	UsageDescription	TransactionCount
1	Swipe Transaction	6967185
2	Chip Transaction	4780818
3	Online Transaction	1557912



6. Top Cities by Total Amount

Results | Messages

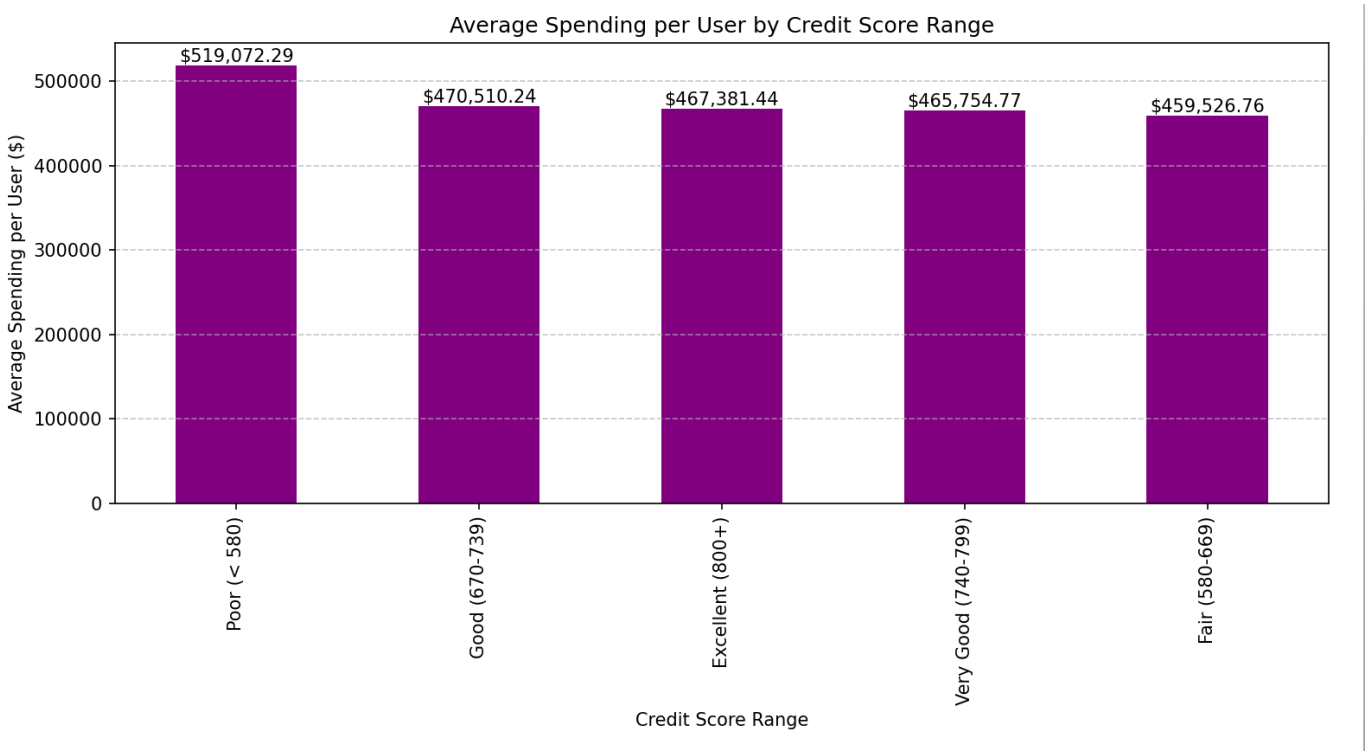
	City	TotalAmount
1	Houston-TX	6608055.27
2	Miami-FL	3480604.43
3	San Francisco-CA	3184666.93
4	Memphis-TN	3078223.20
5	Las Vegas-NV	2973865.78
6	Los Angeles-CA	2965266.39
7	Atlanta-GA	2950725.51
8	Minneapolis-MN	2815878.94
9	Orlando-FL	2722901.46
10	Oakland-CA	2714990.23
11	Brooklyn-NY	2677796.23
12	Dallas-TX	2638144.51
13	Chicago-IL	2623600.71
14	Philadelphia-PA	2409094.47
15	New York-NY	2388489.75
16	San Antonio-TX	2373615.74
17	Tucson-AZ	2269133.84
18	Indianapolis-IN	2215269.12
19	Bronx-NY	2142095.02
20	San Jose-CA	2060813.47



7. Credit Score Impact on Spending Patterns

Results | Messages

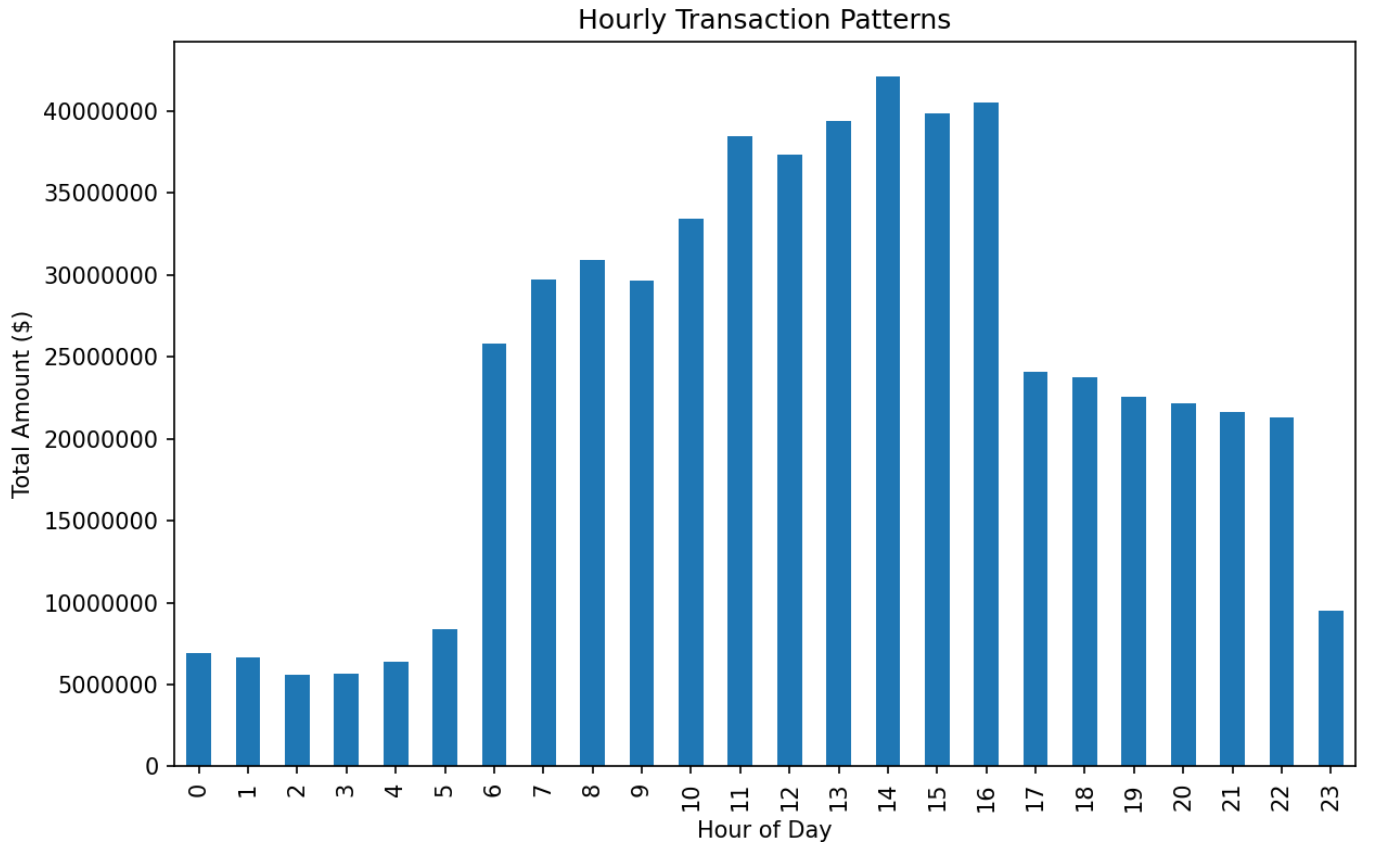
	CreditScoreRange	NumberOfUsers	AvgIncome	AvgDebt	AvgTransactionsPerUser	AvgSpendPerUser
1	Poor (< 580)	42	54119.339103	74201.446586	11369	519072.286428
2	Good (670-739)	590	46141.546385	54485.368529	10694	470510.236728
3	Excellent (800+)	100	45794.105006	51152.756080	11377	467381.444300
4	Very Good (740-799)	306	46612.171126	55902.192151	11290	465754.766797
5	Fair (580-669)	181	47270.014422	73456.970524	10641	459526.759723



8. Hourly Transaction Patterns

Results | Messages

	Hour	TotalAmount
1	0	6937962.75
2	1	6627351.42
3	2	5565786.37
4	3	5673053.87
5	4	6367185.47
6	5	8363710.32
7	6	25789299.50
8	7	29703723.51
9	8	30926930.79
10	9	29677877.04
11	10	33468050.41
12	11	38462432.80
13	12	37317780.00
14	13	39432095.90
15	14	42112924.61
16	15	39870059.48
17	16	40506365.76
18	17	24074636.69
19	18	23788324.25
20	19	22579384.00
21	20	22163906.95
22	21	21617380.60
23	22	21308226.06
24	23	9501073.73



Καταγραφή συμπερασμάτων. (5%)

Βλέπουμε ότι μπορούμε να εισάγουμε δεδομένα προς ανάλυση πολύ εύκολα κι ότι με σχετικά απλά ερωτήματα, μπορούμε να βγάλουμε χρήσιμα συμπεράσματα για την συμπεριφορά τον καταναλωτών. πχ

- Οι κινήσεις με τον μεγαλύτερο όγκο συναλλαγών γίνονται συνήθως πρωί προς μεσημέρι
- Οι κάτοικοι του Huston,Texas σπαταλούν περισσότερα χρήματα
- Άτομα >35 ετών κάνουν τις περισσότερες αγορές