**Data Warehouses and Data Analysis Techniques  
PMS 2024**

**Γεώργιος Βώβος**

**Αριθμός μητρώου: Μ013124003**

Contents

[Εύρεση και κατανόηση του χώρου εφαρμογής της ΑΔ. (5%) 2](#_Toc196148594)

[Εύρεση και κατανόηση των δεδομένων που θα αποθηκευτούν στην ΑΔ. (10%) 2](#_Toc196148595)

[Σχεδίαση της ΑΔ. (20%) 3](#_Toc196148596)

[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%) 7](#_Toc196148597)

[Υλοποίηση της ΑΔ σε ένα από γνωστά συστήματα διαχείρισης βάσεων δεδομένων (SQL Server, MySQL, Postgres, Oracle). (20%) 4](#_Toc196148598)

[Διατύπωση (τουλάχιστον 6) ερωτημάτων SQL που θα χρησιμοποιούν συγκεντρωτικές συναρτήσεις και συνδυασμό πινάκων. (15%) 7](#_Toc196148599)

[Εκτέλεση των ερωτημάτων και εμφάνιση αποτελεσμάτων με οπτικοποίηση. (10%) 7](#_Toc196148600)

[Καταγραφή συμπερασμάτων. (5%) 8](#_Toc196148601)

# 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 datataset των 1.3GB  
Στην εργασία, υπάρχει συννημένο έιναι μικρό υποσύνολο 60.000 εγγραγών , περίπου 6ΜΒ.  
  
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](https://en.wikipedia.org/wiki/Star_schema).

A screenshot of a computer

AI-generated content may be incorrect.

# 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%)

-- 1. Top 10 Users by Total Transaction Amount

-- Identifies your highest value customers

SELECT TOP 10

    u.UserId,

    u.YearlyIncome,

    u.CreditScore,

    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, u.YearlyIncome, u.CreditScore

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= 2019

GROUP BY d.year, d.month

ORDER BY d.year, d.month;

-- 3. Merchant Categories with Highest Error Rates

-- Helps identify potential fraud or 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;

-- 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;

-- 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 credit\_score\_range,

    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;

-- 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**  
   A screenshot of a computer screen

   AI-generated content may be incorrect.

**2 Transaction Volume by Month (2018)**

A screenshot of a computer

AI-generated content may be incorrect.

# Καταγραφή συμπερασμάτων. (5%)