FINAL PROJECT REPORT – DATABASE SYSTEMS

University Admissions s Test Score Analytics System

Course: Databases
Project Title: University Admissions C Test Score Analytics System

Team Members:

- Seher Balibay (220709072)
- Arda Karahan (230709025)
- Ceren Doganay (220709065)

1. Project Description:

This project designs and implements a relational database system for analyzing U.S. universities based on admissions rates, SAT/ACT test scores, income distributions, and ranking information. The system supports analytical queries through normalized tables, stored procedures, and views, enabling flexible and efficient data analysis.

2. Changes Since Phase I

Since Phase I, the database design has been revised to improve normalization, reduce redundancy, and better support analytical queries.

a. Unification of Exam Score Tables

In Phase I, SAT and ACT scores were stored in separate tables with multiple score columns.
In Phase II, these structures were replaced by ExamScores, ExamTypes and ScoreTypes table. Exam scores are now represented using ExamType from ExamTypes table, ScoreType from ScorTypes table, Percentile, and ScoreValue.

This change eliminated repeating attributes and ensured compliance with Third Normal Form (3NF).

b. Normalization of Income Distribution Data

Previously, income distribution data was stored as multiple percentage columns. In the final design, this data was normalized into the IncomeDistributions table, where each income bracket is stored as a separate row.

This structure simplifies aggregation and comparison across institutions.

c. Central Role of the Institutions Table

The Institutions table was restructured to serve as the central entity of the database.

This approach improves referential integrity and simplifies join operations.

d. Removal of Redundant Attributes

Several redundant or derived attributes present in Phase I were removed. Derived values such as averages are now calculated using queries, views, or aggregate functions rather than being stored directly.

e. Addition of Stored Procedures and Views

Phase I included only basic table structures.
In Phase II, stored procedures and views were added to support dynamic queries and summarized outputs, as required by the project guidelines.

f. Improved Relationship Structure

The final ER diagram clearly defines one-to-many relationships between institutions and related entities such as exam scores and income distributions.

Overall, the database evolved from a partially normalized structure to a fully normalized relational design that supports efficient querying and future extension.

ER DIAGRAM:

First diagram:

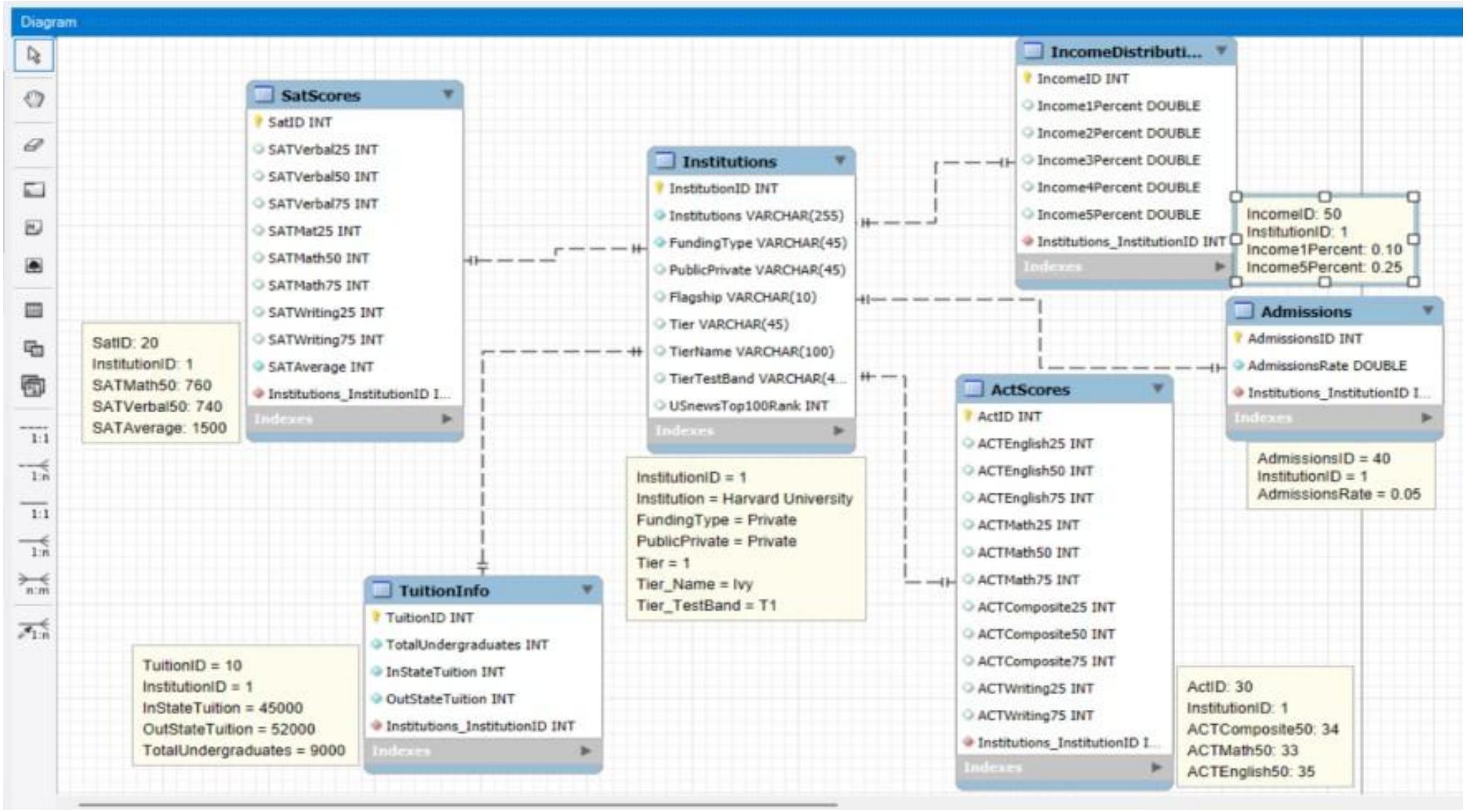
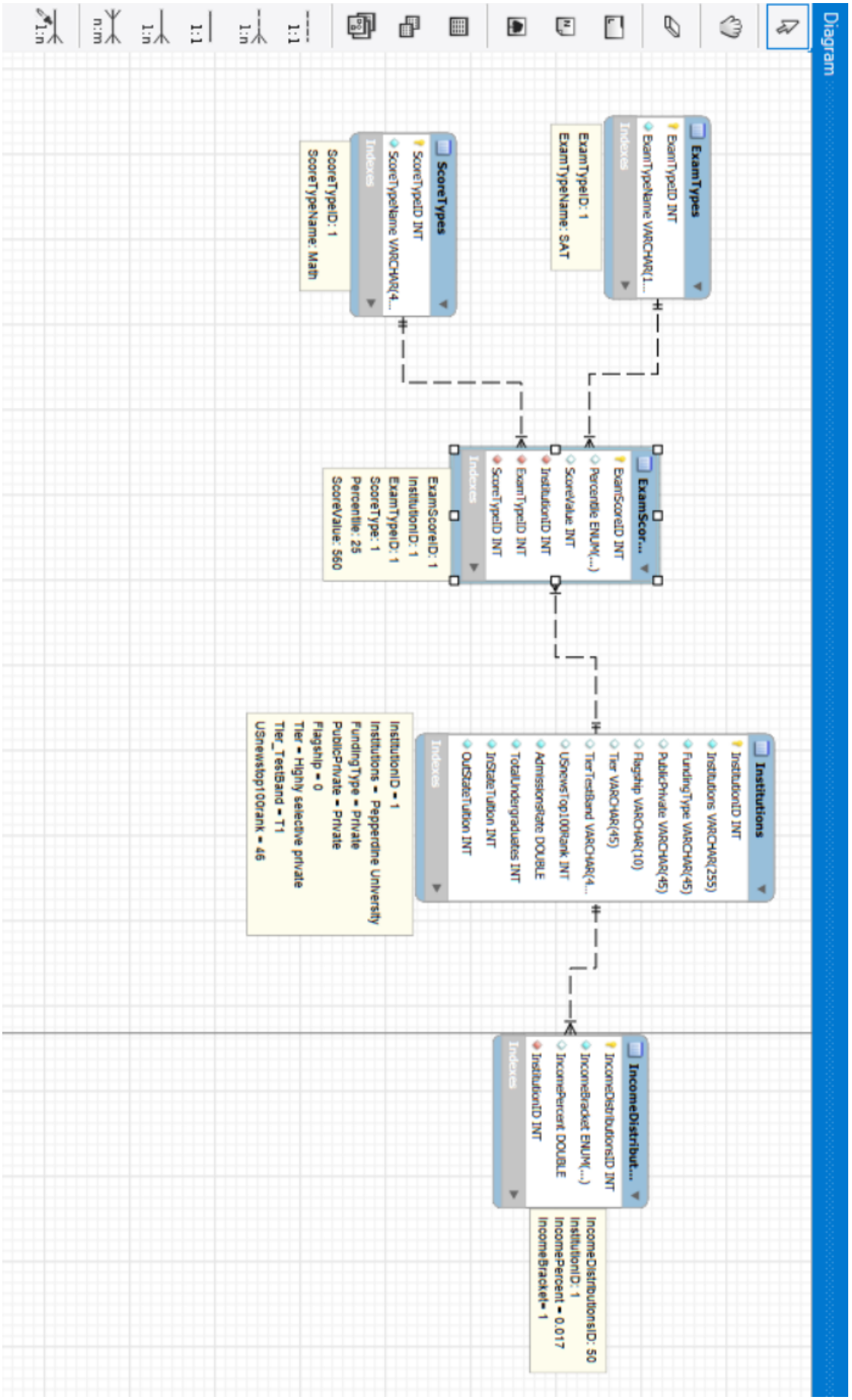


Diagram
SatScores
SatID INT
SATVerbal25 INT
SATVerbal50 INT
SATVerbal75 INT
SATMat25 INT
SATMath50 INT
SATMath75 INT
SATWriting25 INT
SATWriting75 INT
SATAverage INT
Institutions_InstitutionID I...
Indexes
SatID: 20
InstitutionID: 1
SATMath50: 760
SATVerbal50: 740
SATAverage: 1500
Institutions
InstitutionID INT
Institutions VARCHAR(255)
FundingType VARCHAR(45)
PublicPrivate VARCHAR(45)
Flagship VARCHAR(10)
Tier VARCHAR(45)
TierName VARCHAR(100)
TierTestBand VARCHAR(4...
USnewsTop100Rank INT
Indexes
InstitutionID = 1
Institution = Harvard University
FundingType = Private
PublicPrivate = Private
Tier = 1
Tier_Name = Ivy
Tier_TestBand = T1
IncomeDistributi...
IncomeID INT
Income1Percent DOUBLE
Income2Percent DOUBLE
Income3Percent DOUBLE
Income4Percent DOUBLE
Income5Percent DOUBLE
Institutions_InstitutionID INT
Indexes
IncomeID: 50
InstitutionID: 1
Income1Percent: 0.10
Income5Percent: 0.25
Admissions
AdmissionsID INT
AdmissionsRate DOUBLE
Institutions_InstitutionID I...
Indexes
AdmissionsID = 40
InstitutionID = 1
AdmissionsRate = 0.05
ActScores
ActID INT
ACTEnglish25 INT
ACTEnglish50 INT
ACTEnglish75 INT
ACTMath25 INT
ACTMath50 INT
ACTMath75 INT
ACTComposite25 INT
ACTComposite50 INT
ACTComposite75 INT
ACTWriting25 INT
ACTWriting75 INT
Institutions_InstitutionID I...
Indexes
ActID: 30
InstitutionID: 1
ACTComposite50: 34
ACTMath50: 33
ACTEnglish50: 35
TuitionInfo
TuitionID INT
TotalUndergraduates INT
InStateTuition INT
OutStateTuition INT
Institutions_InstitutionID INT
Indexes
TuitionID = 10
InstitutionID = 1
InStateTuition = 45000
OutStateTuition = 52000
TotalUndergraduates = 9000

Last Diagram :



3. Data Loading Process

The dataset used in this project was obtained from Kaggle:
https://www.kaggle.com/datasets/alexsciences/university-test-scores

The data was originally provided as a CSV file containing admissions rates, SAT/ACT test scores, income distribution percentages, and ranking information for U.S. universities.

The database was loaded in multiple steps. First, the raw CSV file was imported into MySQL using the **MySQL Workbench Table Data Import Wizard** to create a staging table (university_test_scores).
Before loading the final tables, preprocessing was performed using SQL to handle missing values, convert non-numeric fields (e.g., "NA") into NULL, and cast text-based numeric attributes into appropriate data types.

Custom **INSERT … SELECT** SQL scripts were developed to:

- Populate the Institutions table with unique institution records
- Normalize exam score data into the ExamScores table
- Transform income distribution percentages into the IncomeDistributions table

All rows from the original dataset were successfully uploaded into the database. No data was excluded.

4. If you did not use MySql Workbench on Windows, describe the exact software/hardware platform you used.

We used MySql Workbench on Windows.

5. User's Guide – Views and Stored Procedures

Stored Procedure: GetInstitutionExamScores

This stored procedure retrieves all exam score records belonging to a specified institution. The institution name is provided as an input parameter, and the procedure returns the exam type, score type, percentile, and score value. It performs JOIN operations across normalized tables and is primarily intended for analytical and reporting purposes.

Stored Procedure: GetTopInstitutionsByScore

This stored procedure retrieves the institutions with the highest exam scores for a selected exam type, score type, and percentile. The user can also specify how many institutions will be returned (Top-N query). This procedure supports comparative performance analysis across institutions and demonstrates the use of parameterized queries, sorting, and limiting in SQL.

Stored Procedure: InsertExamScore

This stored procedure inserts a new exam score record into the ExamScores table. Rather than requiring users to manually supply foreign-key identifiers, it accepts human-readable input such as institution name, exam type, and score type. These values are internally mapped to their corresponding IDs using lookup tables, improving data-entry usability while maintaining referential integrity within the normalized database schema.

View: vw_AllExamScores

This view combines data from the ExamScores, Institutions, ExamTypes, and ScoreTypes tables into a single logical dataset. It presents each exam score together with the institution name, exam type, score type, percentile, and score value. The view simplifies reporting by providing a denormalized perspective over the normalized relational structure.

View: vw_AvgExamScoresByInstitution

This view calculates the average exam score for each institution, grouped by exam type, score type, and percentile. It supports analytical reporting by allowing users to compare institutional SAT/ACT performance and demonstrates the use of aggregation over normalized relational data.

6. Outputs of Selected Stored Procedures and Views

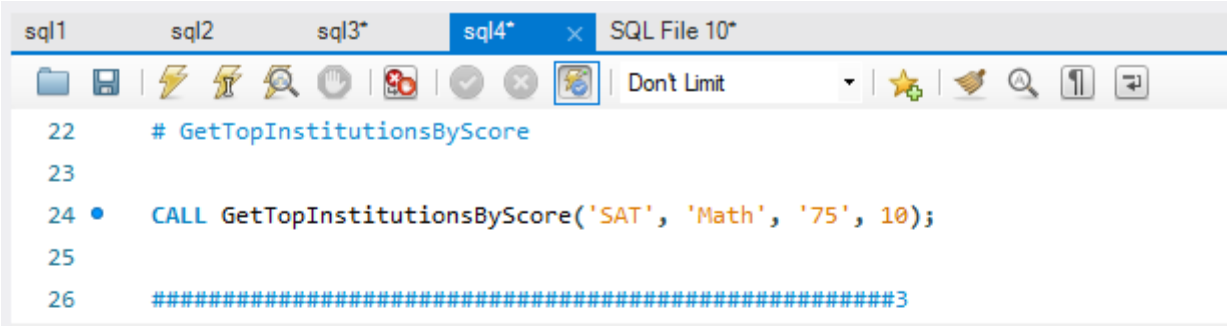
Output of Stored Procedure: GetInstitutionExamScores

```
11      #1. GetInstitution :
12
13 •    SELECT Institutions
14      FROM Institutions
15      LIMIT 5;
16
17 •    CALL GetInstitutionExamScores('Alabama State University');
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

InstitutionName	ExamType	ScoreType	Percentile	ScoreValue
Alabama State University	ACT	Composite	25	16
Alabama State University	ACT	Composite	50	18
Alabama State University	ACT	Composite	75	20
Alabama State University	ACT	English	25	15
Alabama State University	ACT	English	50	18
Alabama State University	ACT	English	75	21
Alabama State University	ACT	Math	25	15
Alabama State University	ACT	Math	50	17
Alabama State University	ACT	Math	75	19
Alabama State University	SAT	Average	50	977
Alabama State University	SAT	Math	25	421
Alabama State University	SAT	Math	50	477
Alabama State University	SAT	Math	75	531
Alabama State University	SAT	Verbal	25	444
Alabama State University	SAT	Verbal	50	489
Alabama State University	SAT	Verbal	75	538

Output of Stored Procedure: GetTopInstitutionsByScore



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

InstitutionName	ExamType	ScoreType	Percentile	ScoreValue
Alabama State University	ACT	Composite	25	16
Alabama State University	ACT	Composite	50	18
Alabama State University	ACT	Composite	75	20
Alabama State University	ACT	English	25	15
Alabama State University	ACT	English	50	18
Alabama State University	ACT	English	75	21
Alabama State University	ACT	Math	25	15
Alabama State University	ACT	Math	50	17
Alabama State University	ACT	Math	75	19
Alabama State University	SAT	Average	50	977
Alabama State University	SAT	Math	25	421
Alabama State University	SAT	Math	50	477
Alabama State University	SAT	Math	75	531
Alabama State University	SAT	Verbal	25	444
Alabama State University	SAT	Verbal	50	489
Alabama State University	SAT	Verbal	75	538

.Output of Stored Procedure: InsertExamScore

sql1 sql2 sql3* sql4* SQL File 10*

Don't Limit

```
25
26  ##########################################################3
27  #  InsertExamScore
28
29  SELECT Institutions
30  FROM Institutions
31  LIMIT 1;
32
33  CALL InsertExamScore(
34      'Alabama A & M University',
35      'SAT',
36      'Math',
37      '75',
38      780
39  );
40
41  SELECT *
42  FROM ExamScores es
43  JOIN Institutions i ON es.InstitutionID = i.InstitutionID
44  JOIN ExamTypes   et ON es.ExamTypeID    = et.ExamTypeID
45  JOIN ScoreTypes  st ON es.ScoreTypeID   = st.ScoreTypeID
46  ORDER BY es.ExamScoreID DESC
47  LIMIT 1;
```

sql1 sql2 sql3* sql4* SQL File 10*

Don't Limit

```
33  CALL InsertExamScore(
34      'Alabama A & M University',
35      'SAT',
36      'Math',
37      '75',
38      780
39  );
40
41  SELECT *
42  FROM ExamScores es
43  JOIN Institutions i ON es.InstitutionID = i.InstitutionID
44  JOIN ExamTypes   et ON es.ExamTypeID    = et.ExamTypeID
45  JOIN ScoreTypes  st ON es.ScoreTypeID   = st.ScoreTypeID
46  ORDER BY es.ExamScoreID DESC
47  LIMIT 1;
48
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ExamScoreID	Percentile	ScoreValue	InstitutionID	ExamTypeID	ScoreTypeID	InstitutionID	Institutions	FundingType	PublicPrivate	Flagship	Tier	TierTestBand
16389	75	780	1	1	1	1	Alabama A & M University	public	NA	NA	NA	NA

Output of View Procedure: vw_AllExamScores

sql1 sql2 sql3* sql4* SQL File 10*

Don't Limit

```sql
CREATE VIEW vw_AllExamScores AS
SELECT
    es.ExamScoreID,
    i.Institutions      AS InstitutionName,
    et.ExamTypeName     AS ExamType,
    st.ScoreTypeName    AS ScoreType,
    es.Percentile,
    es.ScoreValue
FROM ExamScores es
JOIN Institutions i ON es.InstitutionID = i.InstitutionID
JOIN ExamTypes   et ON es.ExamTypeID    = et.ExamTypeID
JOIN ScoreTypes  st ON es.ScoreTypeID   = st.ScoreTypeID;

SELECT * FROM vw_AllExamScores
LIMIT 10;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ExamScoreID	InstitutionName	ExamType	ScoreType	Percentile	ScoreValue
5879	Alabama A & M University	SAT	Average	50	947
5880	University of Alabama at Birmingham	SAT	Average	50	1251
5881	University of Alabama in Huntsville	SAT	Average	50	1321
5882	Alabama State University	SAT	Average	50	977
5883	The University of Alabama	SAT	Average	50	1287

Output of View Procedure: vw_AvgExamScoresByInstitution

sql1 sql2 sql3* sql4* SQL File 10*

Don't Limit

```sql
-- View that provides average exam scores per institution, exam type, score type and percentile.
CREATE VIEW vw_AvgExamScoresByInstitution AS
SELECT
    i.InstitutionID,
    i.Institutions      AS InstitutionName,
    et.ExamTypeName     AS ExamType,
    st.ScoreTypeName    AS ScoreType,
    es.Percentile,
    AVG(es.ScoreValue)  AS AvgScore
FROM ExamScores es
JOIN Institutions i ON es.InstitutionID = i.InstitutionID
JOIN ExamTypes   et ON es.ExamTypeID    = et.ExamTypeID
JOIN ScoreTypes  st ON es.ScoreTypeID   = st.ScoreTypeID
GROUP BY
    i.InstitutionID,
    i.Institutions,
    et.ExamTypeName,
```

```
41          st.ScoreTypeName,
42          es.Percentile;
43
44  •   SELECT *
45      FROM vw_AvgExamScoresByInstitution
46      ORDER BY AvgScore DESC
47      LIMIT 10;
48
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

InstitutionID	InstitutionName	ExamType	ScoreType	Percentile	AvgScore
145	University of Chicago	SAT	Average	50	1554.0000
291	Harvard University	SAT	Average	50	1553.0000
791	Stanford University	SAT	Average	50	1553.0000
298	Massachusetts Institute of Technology	SAT	Average	50	1553.0000
695	Rice University	SAT	Average	50	1553.0000
270	Johns Hopkins University	SAT	Average	50	1553.0000

vw_AvgExamScoresByInstitutio...

7. System Limitations and Suggested Improvements

Limitations

1. Single-source dataset: The database is built from one public Kaggle dataset, so the system is limited to the coverage, accuracy, and update frequency of that source.
2. Missing/NULL values: Some institutions do not provide complete SAT/ACT percentiles or income bracket percentages. These are

stored as NULL, which can reduce the completeness of certain analyses.

3. No historical tracking: The dataset represents a snapshot in time. The schema does not track changes across years (e.g., ranking or tuition trends over time).
4. Limited domain scope: The system focuses on U.S. institutions and a predefined set of attributes (admissions, test scores, income distribution, tuition). Other relevant factors (e.g., graduation rates, employment outcomes) are not included.

Suggested Improvements

1. **Add time dimension (Year/Semester): Introduce a time-based structure to store historical changes in rankings, tuition, admissions rates, and scores to enable trend analysis.**
2. **Data validation and cleansing layer: Implement preprocessing scripts (Python/ETL) to standardize naming, handle outliers, and improve data consistency before loading.**
3. **Expand the dataset: Integrate additional sources (e.g., official government/education datasets) to increase coverage and reliability.**
4. **Advanced indexing and query optimization: Add composite indexes based on common query patterns (e.g., (ExamType, ScoreType, Percentile, InstitutionID)) and analyze query plans for improvement.**

8. FULL RELATIONAL TABLE SPECIFICATION (SQL DDL)

```sql
SET FOREIGN_KEY_CHECKS = 0;

DROP TABLE IF EXISTS ExamScores;
DROP TABLE IF EXISTS IncomeDistributions;
DROP TABLE IF EXISTS ExamTypes;
DROP TABLE IF EXISTS ScoreTypes;
DROP TABLE IF EXISTS Institutions;

SET FOREIGN_KEY_CHECKS = 1;

CREATE TABLE ExamTypes (
    ExamTypeID   TINYINT PRIMARY KEY,
    ExamTypeName VARCHAR(10) NOT NULL
);

CREATE TABLE ScoreTypes (
    ScoreTypeID   TINYINT PRIMARY KEY,
    ScoreTypeName VARCHAR(45) NOT NULL UNIQUE
);

CREATE TABLE Institutions (
    InstitutionID       INT AUTO_INCREMENT PRIMARY KEY,
    Institutions        VARCHAR(255) NOT NULL,
    FundingType         VARCHAR(45),
    PublicPrivate       VARCHAR(45),
```

```
    Flagship            VARCHAR(10),
    Tier                VARCHAR(45),
    TierTestBand        VARCHAR(45),
    USNewsTop100Rank    INT,
    AdmissionsRate      DOUBLE,
    TotalUndergraduates INT,
    InStateTuition      INT,
    OutStateTuition     INT
);

CREATE TABLE IncomeDistributions (
    IncomeDistributionsID INT AUTO_INCREMENT PRIMARY KEY,
    InstitutionID         INT NOT NULL,
    IncomeBracket         TINYINT NOT NULL,  -- 1..5
    IncomePercent         DOUBLE,
    FOREIGN KEY (InstitutionID)
        REFERENCES Institutions(InstitutionID)
);

CREATE TABLE ExamScores (
    ExamScoreID   INT AUTO_INCREMENT PRIMARY KEY,
    Percentile    ENUM('25','50','75') NOT NULL,
    ScoreValue    INT,
    InstitutionID INT NOT NULL,
    ExamTypeID    TINYINT NOT NULL,
    ScoreTypeID   TINYINT NOT NULL,

    FOREIGN KEY (InstitutionID)
        REFERENCES Institutions(InstitutionID),
    FOREIGN KEY (ExamTypeID)
        REFERENCES ExamTypes(ExamTypeID),
    FOREIGN KEY (ScoreTypeID)
        REFERENCES ScoreTypes(ScoreTypeID)
);
```

9. SQL CODES OF THE SYSTEM

Insertations:

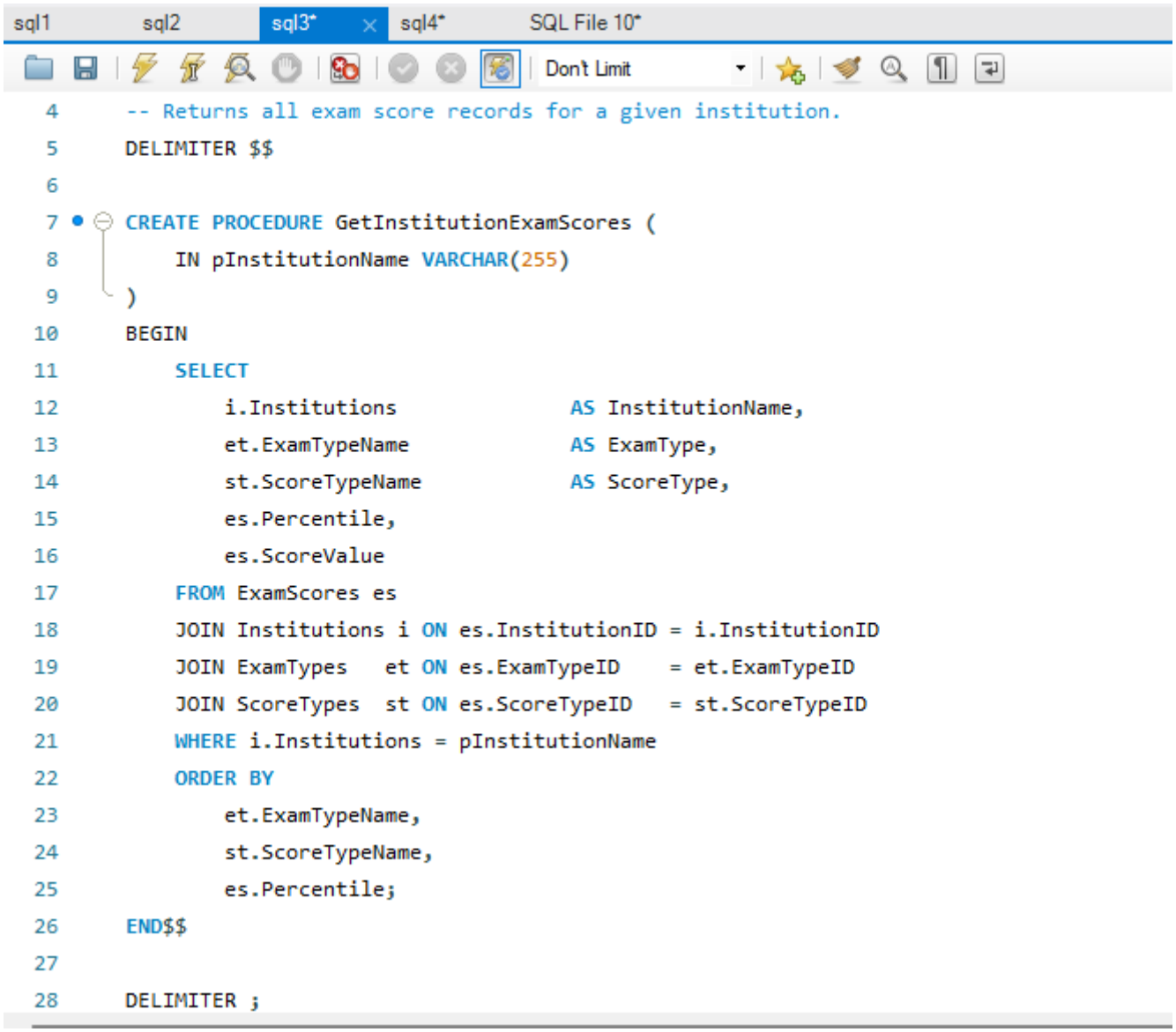
```sql
INSERT INTO ExamTypes (ExamTypeID, ExamTypeName)
VALUES
(1, 'SAT'),
(2, 'ACT');

INSERT INTO ScoreTypes (ScoreTypeID, ScoreTypeName)
VALUES
(1, 'Math'),
(2, 'Verbal'),
(3, 'Writing'),
(4, 'Composite'),
(5, 'Average'),
(6, 'English');  -- ACT English için
```

```sql
INSERT INTO Institutions (
    Institutions,
    FundingType,
    PublicPrivate,
    Flagship,
    Tier,
    TierTestBand,
    USNewsTop100Rank,
    AdmissionsRate,
    TotalUndergraduates,
    InStateTuition,
    OutStateTuition
)
SELECT
    uts.Institution,
    uts.Funding_Type,
    uts.`Public/Private`,
    uts.Flagship,
    uts.Tier,
    uts.Tier_TestBand,
    CASE
        WHEN uts.USnewsTop100Rank = 'NA' OR uts.USnewsTop100Rank IS NULL
        THEN NULL
        ELSE CAST(uts.USnewsTop100Rank AS SIGNED)
    END,
    uts.Admissions_Rate,
    uts.Total_Undergraduates,
```

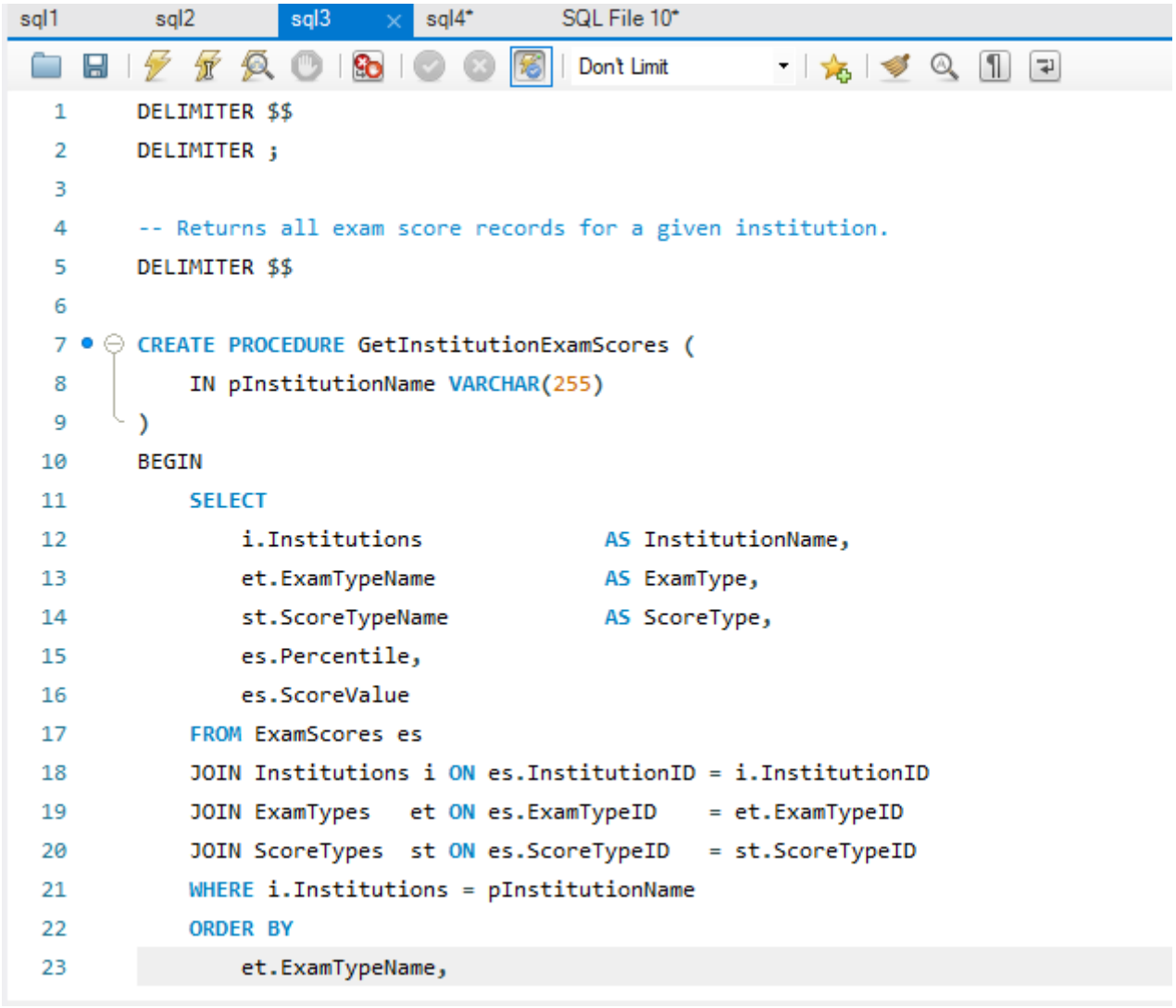
```sql
INSERT INTO IncomeDistributions (
    InstitutionID,
    IncomeBracket,
    IncomePercent
)
SELECT i.InstitutionID, 1, uts.Income_1_Percent
FROM university_test_scores uts
JOIN Institutions i ON i.Institutions = uts.Institution
WHERE uts.Income_1_Percent IS NOT NULL

UNION ALL
SELECT i.InstitutionID, 2, uts.Income_2_Percent
FROM university_test_scores uts
JOIN Institutions i ON i.Institutions = uts.Institution
WHERE uts.Income_2_Percent IS NOT NULL

UNION ALL
SELECT i.InstitutionID, 3, uts.Income_3_Percent
FROM university_test_scores uts
JOIN Institutions i ON i.Institutions = uts.Institution
WHERE uts.Income_3_Percent IS NOT NULL

UNION ALL
SELECT i.InstitutionID, 4, uts.Income_4_Percent
FROM university_test_scores uts
JOIN Institutions i ON i.Institutions = uts.Institution
WHERE uts.Income_4_Percent IS NOT NULL
```

```sql
INSERT INTO ExamScores (
    ExamTypeID,
    ScoreTypeID,
    InstitutionID,
    Percentile,
    ScoreValue
)

-- SAT VERBAL
SELECT et.ExamTypeID, st.ScoreTypeID, i.InstitutionID, '25',
       CAST(uts.SAT_Verbal_25 AS UNSIGNED)
FROM university_test_scores uts
JOIN Institutions i   ON i.Institutions   = uts.Institution
JOIN ExamTypes   et   ON et.ExamTypeName  = 'SAT'
JOIN ScoreTypes  st   ON st.ScoreTypeName = 'Verbal'
WHERE uts.SAT_Verbal_25 REGEXP '^[0-9]+$'

UNION ALL
SELECT et.ExamTypeID, st.ScoreTypeID, i.InstitutionID, '50',
       CAST(uts.SAT_Verbal_50 AS UNSIGNED)
FROM university_test_scores uts
JOIN Institutions i   ON i.Institutions   = uts.Institution
JOIN ExamTypes   et   ON et.ExamTypeName  = 'SAT'
JOIN ScoreTypes  st   ON st.ScoreTypeName = 'Verbal'
WHERE uts.SAT_Verbal_50 REGEXP '^[0-9]+$'
```

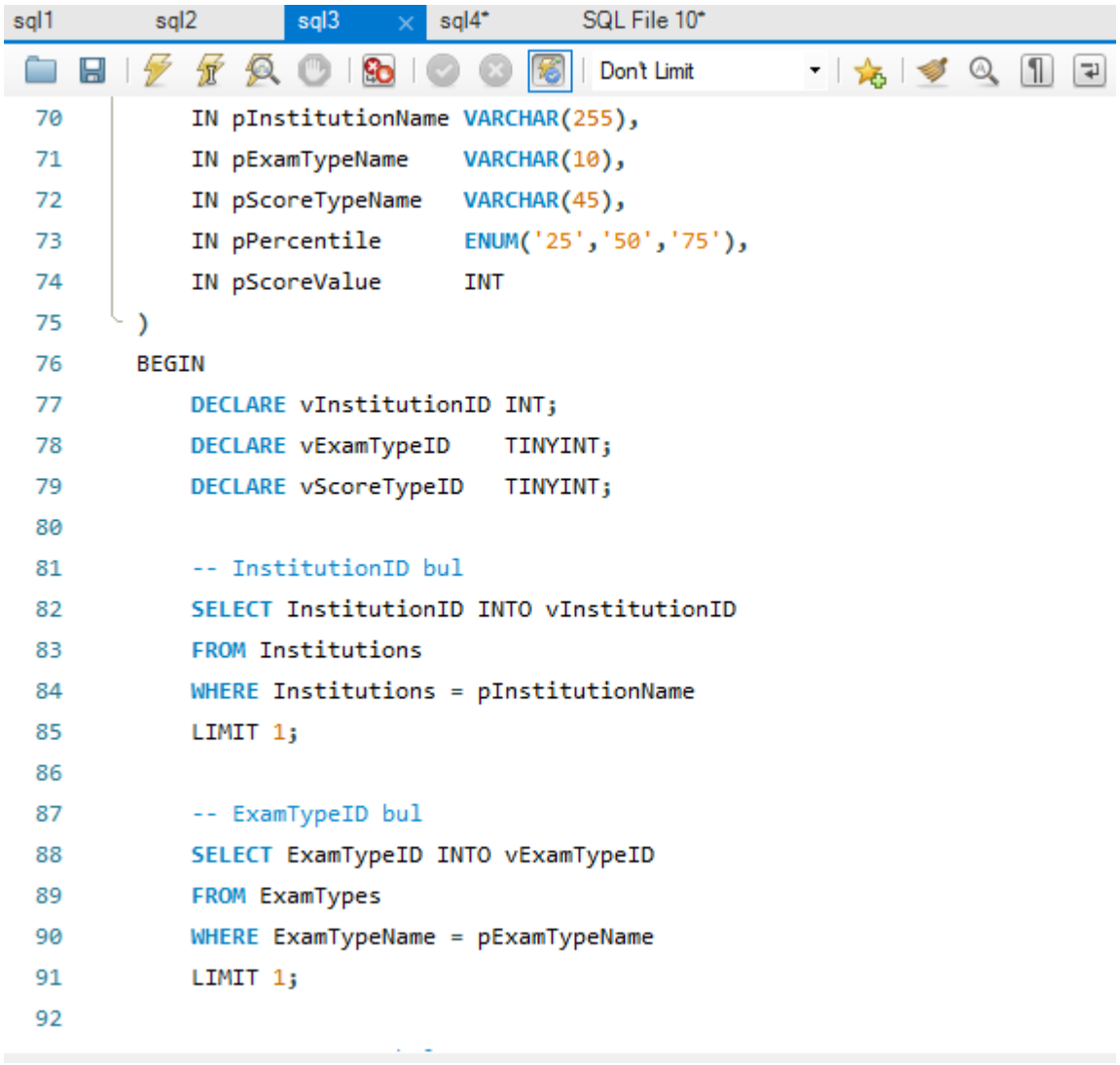
Stored Prodedures SQL Codes:

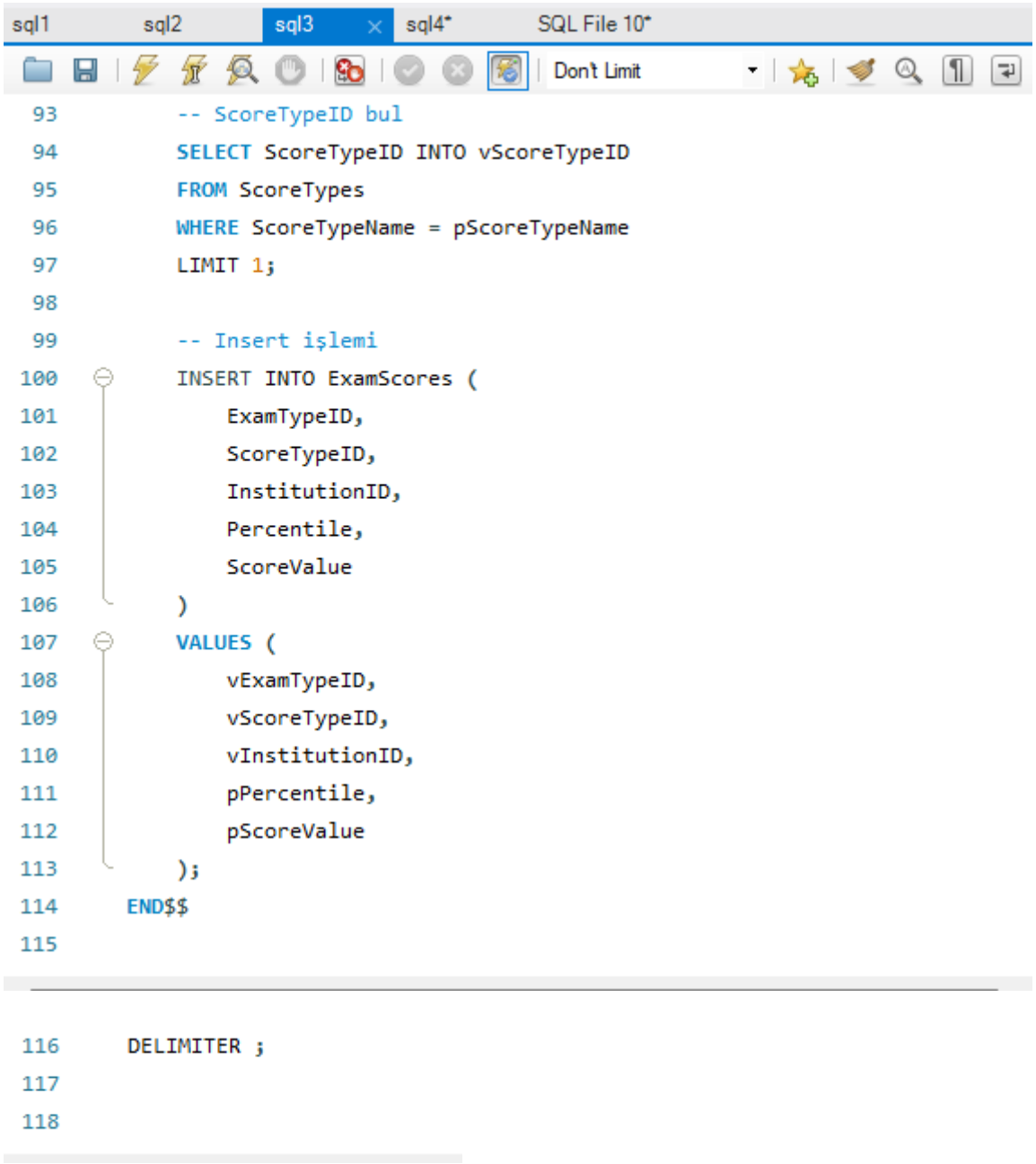



```sql
-- Returns all exam score records for a given institution.
DELIMITER $$

CREATE PROCEDURE GetInstitutionExamScores (
    IN pInstitutionName VARCHAR(255)
)
BEGIN
    SELECT
        i.Institutions              AS InstitutionName,
        et.ExamTypeName             AS ExamType,
        st.ScoreTypeName            AS ScoreType,
        es.Percentile,
        es.ScoreValue
    FROM ExamScores es
    JOIN Institutions i ON es.InstitutionID = i.InstitutionID
    JOIN ExamTypes   et ON es.ExamTypeID    = et.ExamTypeID
    JOIN ScoreTypes  st ON es.ScoreTypeID   = st.ScoreTypeID
    WHERE i.Institutions = pInstitutionName
    ORDER BY
        et.ExamTypeName,
        st.ScoreTypeName,
        es.Percentile;
END$$

DELIMITER ;
```


sql1
sql2
sql3
sql4*
SQL File 10*
Don't Limit
DELIMITER $$
DELIMITER ;
-- Returns all exam score records for a given institution.
DELIMITER $$
CREATE PROCEDURE GetInstitutionExamScores (
IN pInstitutionName VARCHAR(255)
)
BEGIN
SELECT
i.Institutions AS InstitutionName,
et.ExamTypeName AS ExamType,
st.ScoreTypeName AS ScoreType,
es.Percentile,
es.ScoreValue
FROM ExamScores es
JOIN Institutions i ON es.InstitutionID = i.InstitutionID
JOIN ExamTypes et ON es.ExamTypeID = et.ExamTypeID
JOIN ScoreTypes st ON es.ScoreTypeID = st.ScoreTypeID
WHERE i.Institutions = pInstitutionName
ORDER BY
et.ExamTypeName,


sql1
sql2
sql3
sql4*
SQL File 10*
Don't Limit
st.ScoreTypeName,
es.Percentile;
END$$
DELIMITER ;
##
-- Returns the top N institutions with the highest exam scores for a given exam type, score type, and percentile.
DELIMITER $$
CREATE PROCEDURE GetTopInstitutionsByScore (
IN pExamTypeName VARCHAR(10),
IN pScoreTypeName VARCHAR(45),
IN pPercentile ENUM('25','50','75'),
IN pLimitN INT
)
BEGIN
SELECT
i.Institutions AS InstitutionName,
et.ExamTypeName AS ExamType,
st.ScoreTypeName AS ScoreType,
es.Percentile,
es.ScoreValue
FROM ExamScores es
JOIN Institutions i ON es.InstitutionID = i.InstitutionID
JOIN ExamTypes et ON es.ExamTypeID = et.ExamTypeID
JOIN ScoreTypes st ON es.ScoreTypeID = st.ScoreTypeID
WHERE et.ExamTypeName = pExamTypeName
AND st.ScoreTypeName = pScoreTypeName
AND es.Percentile = pPercentile
ORDER BY es.ScoreValue DESC
LIMIT pLimitN;
END$$
DELIMITER ;
CALL GetTopInstitutionsByScore('SAT', 'Math', '75', 10);
##
DELIMITER $$
CREATE PROCEDURE InsertExamScore (


sql1
sql2
sql3
sql4*
SQL File 10*
Don't Limit
70 IN pInstitutionName VARCHAR(255),
71 IN pExamTypeName VARCHAR(10),
72 IN pScoreTypeName VARCHAR(45),
73 IN pPercentile ENUM('25','50','75'),
74 IN pScoreValue INT
75)
76 BEGIN
77 DECLARE vInstitutionID INT;
78 DECLARE vExamTypeID TINYINT;
79 DECLARE vScoreTypeID TINYINT;
80
81 -- InstitutionID bul
82 SELECT InstitutionID INTO vInstitutionID
83 FROM Institutions
84 WHERE Institutions = pInstitutionName
85 LIMIT 1;
86
87 -- ExamTypeID bul
88 SELECT ExamTypeID INTO vExamTypeID
89 FROM ExamTypes
90 WHERE ExamTypeName = pExamTypeName
91 LIMIT 1;
92


sql1
sql2
sql3
sql4*
SQL File 10*
Don't Limit
93 -- ScoreTypeID bul
94 SELECT ScoreTypeID INTO vScoreTypeID
95 FROM ScoreTypes
96 WHERE ScoreTypeName = pScoreTypeName
97 LIMIT 1;
98
99 -- Insert işlemi
100 INSERT INTO ExamScores (
101 ExamTypeID,
102 ScoreTypeID,
103 InstitutionID,
104 Percentile,
105 ScoreValue
106)
107 VALUES (
108 vExamTypeID,
109 vScoreTypeID,
110 vInstitutionID,
111 pPercentile,
112 pScoreValue
113);
114 END$$
115
116 DELIMITER ;
117
118