A project report

On

HORIZONTAL AGGREGATION IN DBMS BY USING K-MEANS CLUSTERING

By

Ashif Ali – BE/15204/12

Ritesh Ranjan – BE/15180/12

Under the guidance of

Dr. Kanhaiya Lal

Submitted in partial fulfillment of the requirement for the award of the Degree of Bachelor of Engineering in computer science

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
BIRLA INSTITUTE OF TECHNOLOGY MESRA,
PATNA CAMPUS
2016

BIRLA INSTITUTE OF TECHNOLOGY

(A Deemed University U/S 3 of UGC Act 1956)

MESRA, PATNA CAMPUS

TO WHOME IT MAY CONCERN

This is to certify that the following students of BE are doing project under me.

SI. No.	Name	Roll No.	Branch
1.	Ashif Ali	BE/15204/12	CSE
2.	Ritesh Ranjan	BE/15180/12	CSE

Project Title: Horizontal Aggregation in DBMS by using K-Mean Clustering

- Their level of interaction was ________ %
- Their frequency of interaction was:
  - Once a Week
  - Twice a Week
  - Once a Month
  - Less than two days in a Semester
- His/Their day to day work and progress is
  - Satisfactory
  - Unsatisfactory

Guide(S)

Name : Dr. Kanhaiya Lal

Dept : Computer Science

CERTIFICATE

This is to certify that the work titled “Horizontal Aggregation in DBMS by using K-Means Clustering” submitted by “Mr. Ashif Ali & Mr. Ritesh Ranjan” in partial fulfillment for the award of degree of B.E. in COMPUTER SCIENCE AND ENGINEERING of Birla Institute of Technology, Mesra has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Dr. Kanhaiya Lal, In-charge of CSE & IT
Dept. of Computer Science and Engineering
BIRLA INSTITUTE OF TECHNOLOGY, MESRA, PATNA CAMPUS

CERTIFICATE OF APPROVAL

The foregoing Project work entitled “Horizontal Aggregation in DBMS by using K-Means Clustering” hereby approved as credible study that it has been presented in satisfactory manner and is recommended for its acceptance in partial fulfillment of the requirement for the award of Bachelor of Engineering in Computer Science & Engineering of Birla Institute of Technology, Mesra, Patna Campus.

INTERNAL EXAMINER EXTERNAL EXAMINER

Co-ordinator/HOD
Department of Computer Science & Engineering

ACKNOWLEDGEMENT

First and foremost, I would like to express my sincere gratitude to my project guide Dr. Kanhaiya Lal, In-charge of CSE & IT and Dr. Upendra Kumar, Assistant Professor, Department of Computer Science, Birla Institute of Technology, Patna, for there guidance and support throughout my project work.

I also sincerely thank to Samant Saurabh, Assistant Professor, Department of Computer Science, for all the help he has extended to me.

I would like to thank the faculties of department of Computer Science, BIT Patna who extended all kinds of co-operation for the completion of this project.

Last but not the least, I would like to thank all my friends for their moral support and help during the course of this project.

Ashif Ali – BE/15204/12

Ritesh Ranjan – BE/15180/12

Abstract

Preparing a data set for analysis is generally the most time consuming task in a data mining project, requiring many complex SQL queries, joining tables, and aggregating columns. Existing SQL aggregations have limitations to prepare data sets because they return one column per aggregated group. In general, a significant manual effort is required to build data sets, where a horizontal layout is required. We propose simple, yet powerful, methods to generate SQL code to return aggregated columns in a horizontal tabular layout, returning a set of numbers instead of one number per row. This new class of functions is called horizontal aggregations. Horizontal aggregations build data sets with a horizontal denormalized layout, which is the standard layout required by most data mining algorithms. We propose three fundamental methods to evaluate horizontal aggregations: CASE: Exploiting the programming CASE construct; SPJ: Based on standard relational algebra operators (SPJ queries); PIVOT: Using the PIVOT operator, which is offered by some DBMSs. Experiments with large tables compare the proposed query evaluation methods. Our CASE method has similar speed to the PIVOT operator and it is much faster than the SPJ method. In general, the CASE and PIVOT methods exhibit linear scalability, whereas the SPJ method does not. Horizontal aggregations results in large volumes of data sets which are then partitioned into homogeneous clusters is important in the system. This can be performed by K Means Clustering Algorithm.

Contents

Chapter 1

Introduction

In a relational database, especially with normalized tables, a significant effort is required to prepare a summary data set that can be used as input for a data mining or statistical algorithm. Most algorithms require as input a data set with a horizontal layout, with several records and one variable or dimension per column. That is the case with models like clustering, classification, regression, and PCA. Each research discipline uses different terminology to describe the data set. In data mining the common terms are point-dimension. Statistics literature generally uses observation-variable. Machine learning research uses instance-feature. This project introduces a new class of aggregate functions that can be used to build data sets in a horizontal layout, automating SQL query writing and extending SQL capabilities. We show evaluating horizontal aggregations is a challenging and interesting problem and we introduce alternative methods and optimizations for their efficient evaluation. The K-means clustering algorithm is used to partition data sets after horizontal aggregations. It is one of the simplest unsupervised learning algorithms used to optimize the horizontal aggregation in SQL. For faster data retrieval, K-means clustering algorithm is used to cluster the classified data.

1.1 Motivation

As mentioned above, building a suitable data set for data mining purposes is a time-consuming task. This task generally requires writing long SQL statements or customizing SQL code if it is automatically generated by some tool. There are two main ingredients in such SQL code: joins and aggregations; we focus on the second one. The most widely known aggregation is the sum of a column over groups of rows. Some other aggregations return the average, maximum, minimum, or row count over groups of rows. Unfortunately, all these aggregations have limitations to build data sets for data mining purposes. The main reason is that, in general, data sets that are stored in a relational database come from Online Transaction Processing (OLTP) systems where database schemas are highly normalized. But data mining, statistical, or machine learning algorithms generally require aggregated data in summarized form. Based on current available functions and clauses in SQL, a significant effort is required to compute aggregations when they are desired in a cross-tabular (horizontal) form, suitable to be used by a data mining algorithm. There are further practical reasons to return aggregation results in a horizontal (cross-tabular) layout. Standard aggregations are hard to interpret when there are many result rows. To perform analysis of exported tables into spreadsheets it may be more convenient to have aggregations on the same group in one row. We propose a new class of aggregate functions that aggregate numeric expressions and transpose results to produce a data set with a horizontal layout. Functions belonging to this class are called horizontal aggregations. Horizontal aggregations represent an extended form of traditional SQL aggregations, which return a set of values in a horizontal layout instead of a single value per row.

1.2 Advantages

Our proposed horizontal aggregations provide several unique features and advantages. First, they represent a template to generate SQL code from a data mining tool. Such SQL code automates writing SQL queries, optimizing them, and testing them for correctness. This SQL code reduces manual work in the data preparation phase in a data mining project. Second, since SQL code is automatically generated it is likely to be more efficient than SQL code written by an end user. For instance, a person who does not know SQL well or some who is not familiar with the database schema. Therefore, data sets can be created in less time. Third, the data set can be created entirely inside the DBMS. In modern database environments, it is common to export denormalized data sets to be further cleaned and transformed outside a DBMS in external tools (e.g., statistical packages). Unfortunately, exporting large tables outside a DBMS is slow, creates inconsistent copies of the same data and compromises database security. Therefore, we provide a more efficient, better integrated and more secure solution compared to external data mining tools. Horizontal aggregations just require a small syntax extension to aggregate functions called in a SELECT statement. Alternatively, horizontal aggregations can be used to generate SQL code from a data mining tool to build data sets for data mining analysis.

Chapter 2

Problem Definition

This section defines the table that will be used to explain SQL queries throughout this work. Let F be a table having a simple primary key K represented by an integer, p discrete attributes, and one numeric attribute: F(K, D_1, D_2,..,D_p, A). Our definitions can be easily generalized to multiple numeric attributes. In OLAP terms, F is a fact table with one column used as primary key, p dimensions and one measure column passed to standard SQL aggregations. Subsets of dimension columns are used to group rows to aggregate the measure column. Column K will not be used to compute aggregations. Input table F size is called N. That is, |F| = N.

We now explain tables F_V (vertical) and F_H (horizontal) that are used in this project. Consider a standard SQL aggregation (e.g., sum()) with the GROUP BY clause, which returns results in a vertical layout. Assume there are j + k GROUP BY columns and the aggregated attribute is A. The results are stored on table F_V having j + k columns making up the primary key and A as a nonkey attribute. Table F_V has a vertical layout. The goal of a horizontal aggregation is to transform F_V into a table F_H with a horizontal layout having n rows and j + d columns, where each of the d columns represents a unique combination of the k grouping columns. The n rows represent records for analysis and the d columns represent dimensions or features for analysis. Therefore, n is data set size and d is dimensionality.

F

K	D_1	D_2	A
1	3	X	9
2	2	Y	6
3	1	Y	10
4	1	Y	0
5	2	X	1
6	1	X	null
7	3	X	8
8	2	X	7

F_V

D_1	D_2	A
1	X	null
1	Y	10
2	X	8
2	Y	6
3	X	17

F_H

D_1	D_2X	D_2Y
1	null	10
2	8	6
3	17	null

Fig. 1. Example of F, F_V, and F_H.

2.1 Examples

Fig. 1 gives an example showing the input table F, a traditional vertical sum () aggregation stored in F_V, and a horizontal aggregation stored in F_H. The basic SQL aggregation query is:

SELECT D_1, D_2, sum (A)
FROM F
GROUP BY D_1, D_2
ORDER BY D_1, D_2;

Notice table F_V has only five rows because D1 = 3 and D2 = Y do not appear together. Also, the first row in F_V has null in A following SQL evaluation semantics. On the other hand, table F_H has three rows and two (d = 2) nonkey columns, effectively storing six aggregated values. In F_H it is necessary to populate the last row with null. Therefore, nulls may come from F or may be introduced by the horizontal layout.

We now give other examples with a store (retail) database that requires data mining analysis. To give examples of F, we will use a table transactionLine that represents the transaction table from a store. Table transactionLine has dimensions grouped in three taxonomies (product hierarchy, location, time), used to group rows, and three measures represented by itemQty, costAmt, and salesAmt, to pass as arguments to aggregate functions.

We want to compute queries like “summarize sales for each store by each day of the week”; “compute the total number of items sold by department for each store.” These queries can be answered with standard SQL, but additional code needs to be written or generated to return results in tabular (horizontal) form. Consider the following two queries:

```
SELECT storeId,dayofweekNo,sum(salesAmt)
FROM transactionLine
GROUP BY storeId,dayweekNo
ORDER BY storeId,dayweekNo;
```

```
SELECT storeId,deptId,sum(itemqty)
FROM transactionLine
GROUP BY storeId,deptId
ORDER BY storeId,deptId;
```

Assume there are 200 stores, 30 store departments, and stores are open 7 days a week. The first query returns 1,400 rows which may be time consuming to compare with each other each day of the week to get trends. The second query returns 6,000 rows, which in a similar manner, makes difficult to compare store performance across departments. Even further, if we want to build a data mining model by store (e.g., clustering, regression), most algorithms require store id as primary key and the remaining aggregated columns as nonkey columns. That is, data mining algorithms expect a horizontal layout. In addition, a horizontal layout is generally more I/O efficient than a vertical layout for analysis. Notice these queries have ORDER BY clauses to make output easier to understand, but such order is irrelevant for data mining algorithms. In general, we omit ORDER BY clauses.

2.2 Typical Data Mining Problems

Let us consider data mining problems that may be solved by typical data mining or statistical algorithms, which assume each nonkey column represents a dimension, variable (statistics), or feature (machine learning). Stores can be clustered based on sales for each day of the week. On the other hand, we can predict sales per store department based on the sales in other departments using decision trees or regression. PCA analysis on department sales can reveal which departments tend to sell together. We can find out potential correlation of number of employees by gender within each department. Most data mining algorithms (e.g., clustering, decision trees, regression, and correlation analysis) require result tables from these queries to be transformed into a horizontal layout. There exist data mining algorithms that can directly analyze data sets having a vertical layout (e.g., in transaction format), but they require reprogramming the algorithm to have a better I/O pattern and they are efficient only when there many zero values (i.e., sparse matrices).

Chapter 3
Horizontal Aggregation

We introduce a new class of aggregations that have similar behaviour to SQL standard aggregations, but which produce tables with a horizontal layout. In contrast, we call standard SQL aggregations vertical aggregations since they produce tables with a vertical layout. Horizontal aggregations just require a small syntax extension to aggregate functions called in a SELECT statement. Alternatively, horizontal aggregations can be used to generate SQL code from a data mining tool to build data sets for data mining analysis. We start by explaining how to automatically generate SQL code.

3.1 SQL Code Generation

Our main goal is to define a template to generate SQL code combining aggregation and transposition (pivoting). A second goal is to extend the SELECT statement with a clause that combines transposition with aggregation. Consider the following GROUP BY query in standard SQL that takes a subset L_1,…., Lm from D_1,…..,D_p:

SELECT L_1, …, L_m, sum (A)
FROM F
GROUP BY L1, …, Lm;

This aggregation query will produce a wide table with m + 1 columns with one group for each unique combination of values L_1, …, L_m and one aggregated value per group (sum(A) in this case). In order to evaluate this query the query optimizer takes three input parameters: the input table F, the list of grouping columns L_1, …, L_m, the column to aggregate (A).
The basic goal of a horizontal aggregation is to transpose (pivot) the aggregated column A by a column subset of L_1, …, L_m ; for simplicity assume such subset is R_1, …, R_k where k < m. In other words, we partition the GROUP BY list into two sublists: one list to produce each group (j columns L_1, …, L_j) and another list (k columns R_1, …, R_k) to transpose aggregated values.

Each distinct combination of { R_1, …, R_k } will automatically produce an output column. In particular, if k = 1 then each value in R_1 will become a column storing one aggregation. Therefore, in a horizontal aggregation there are four input parameters to generate SQL code:

1. the input table F,
2. the list of GROUP BY columns L_1, …, L_j,
3. the column to aggregate (A),
4. the list of transposing columns R_1, …, R_k.

Horizontal aggregations preserve evaluation semantics of standard (vertical) SQL aggregations. The main difference will be returning a table with a horizontal layout, possibly having extra nulls.

3.2 Proposed Syntax in Extended SQL

We now turn our attention to a small syntax extension to the SELECT statement. We must point out the proposed extension represents nonstandard SQL because the columns in the output table are not known when the query is parsed. We assume F does not change while a horizontal aggregation is evaluated because new values may create new result columns. Conceptually, we extend standard SQL aggregate functions with a "transposing" BY clause followed by a list of columns (i.e., R_1, …, R_k), to produce a horizontal set of numbers instead of one number. Our proposed syntax is as follows:

SELECT L_1, …, L_j, H(A BY R_1, R_2, …, R_K)
FROM F
GROUP BY L_1, …, L_j;

In the context of our work, H() represents some SQL aggregation (e.g., sum(), count(), min(), max(), avg()). The function H() must have at least one argument represented by A, followed by a list of columns.
We intend to preserve standard SQL evaluation semantics as much as possible. Our goal is to develop sound and efficient evaluation mechanisms. Thus, we propose the following rules:

1. The GROUP BY clause is optional, like a vertical aggregation. That is, the list $L_1, \ldots, L_j$ may be empty. When the GROUP BY clause is not present then there is only one result row.
2. When the clause GROUP BY is present there should not be a HAVING clause that may produce crosstabulation of the same group.
3. The transposing BY clause is optional. When BY is not present then a horizontal aggregation reduces to a vertical aggregation.
4. When the BY clause is present the list $R_1, R_2, \ldots, R_K$ is required, where k = 1 is the default.
5. Horizontal aggregations can be combined with vertical aggregations or other horizontal aggregations on the same query, provided all use the same GROUP BY columns { $L_1, \ldots, L_j$ }.
6. The argument to aggregate represented by A is required. A can be a column name or an arithmetic expression.
7. When H() is used more than once, in different terms, it should be used with different sets of BY columns.

3.2.1 Examples

In a data mining project, most of the effort is spent in preparing and cleaning a data set. A big part of this effort involves deriving metrics and coding categorical attributes from the data set in question and storing them in a tabular (observation, record) form for analysis so that they can be used by a data mining algorithm.

Assume we want to summarize sales information with one store per row for one year of sales. In more detail, we need the sales amount broken down by day of the week, the number of transactions by store per month, the number of items sold by department and total sales. The following query in our extended SELECT syntax provides the desired data set, by calling three horizontal aggregations.

	salesAmt				countTransactions				countItems				total
storeId	Mon	Tue	..	Sun	Jan	Feb	..	Dec	dairy	meat	produce	..	salesAmt
10	120	111		200	2011	1807		4200	34	57	101		25025
32	70	65		98	802	912		1632	32	65	204		14022
⋮													

TABLE 1:A Multidimensional Data Set in Horizontal Layout, Suitable for Data Mining

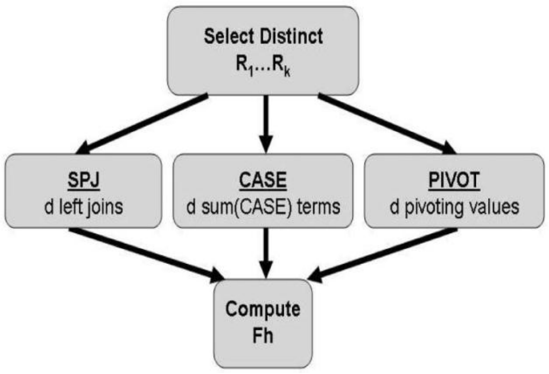
```
SELECT
        storeId,
        sum(salesAmt BY dayofweekName),
        count(distinct transactionid BY salesMonth),
        sum(1 BY deptName),
        sum(salesAmt)
FROM transactionLine
        ,DimDayOfWeek, DimDepartment, DimMonth
WHERE salesYea = 2009
AND transactionLine.dayOfWeekNo = DimDayOfWeek.dayOfWeekNo
AND transactionLine.deptId = DimDepartment.deptId
AND transactionLine.MonthId = DimTime.MonthId
GROUP BY storeId;
```

This query produces a result table like the one shown in Table 1. Observe each horizontal aggregation effectively returns a set of columns as result and there is call to a standard vertical aggregation with no subgrouping columns.

For the first horizontal aggregation, we show day names and for the second one we show the number of day of the week. These columns can be used for linear regression, clustering, or factor analysis. We can analyse correlation of sales based on daily sales. Total sales can be predicted based on volume of items sold each day of the week. Stores can be clustered based on similar sales for each day of the week or similar sales in the same department.

3.3 SQL Code Generation: Query Evaluation Methods

We propose three methods to evaluate horizontal aggregations. The first method relies only on relational operations. That is, only doing select, project, join, and aggregation queries; we call it the SPJ method. The second form relies on the SQL "case" construct; we call it the CASE method. The third method uses the built-in PIVOT operator, which transforms rows to columns (e.g., transposing). Figs. 2 and 3 show an overview of the main steps to be explained below (for a sum() aggregation).
Given the same input table F and horizontal aggregation query, the SPJ, CASE, and PIVOT methods produce the same result.



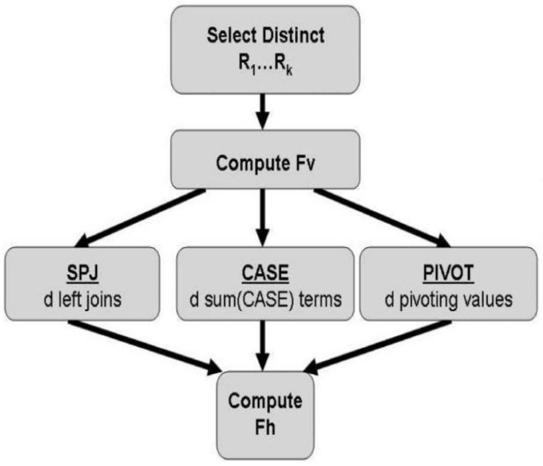


Fig. 2. Main steps of methods based on F (unoptimized).

Fig. 3. Main steps of methods based on F_V (optimized).

3.3.1 SPJ Method

The SPJ method is interesting from a theoretical point of view because it is based on relational operators only. The basic idea is to create one table with a vertical aggregation for each result column, and then join all those tables to produce F_H. We aggregate from F into d projected tables with d Select-Project-Join-Aggregation queries. Each table FI corresponds to one subgrouping combination and has $\{ L_1, \ldots, L_j \}$ as primary key and an aggregation on A as the only nonkey column. It is necessary to introduce an additional table F0, that will be outer joined with projected tables to get a complete result set. We propose two basic substrategies to compute F_H. The first one directly aggregates from F. The second one computes the equivalent vertical aggregation in a temporary table F_V grouping by L_1, …, Lj , R_1, …, Rk. Then horizontal aggregations can be instead computed from F_V, which is a compressed version of F, since standard aggregations are distributive. Let F_V be a table containing the vertical aggregation, based on L_1, …, Lj , R_1, …, Rk. The statement to compute F_V is:

```
INSERT INTO F_V
SELECT L_1, …, Lj , R_1, …, R_k, V(A)
FROM F
GROUP BY L_1, …, Lj , R_1, …, R_k;
```

Table F0 defines the number of result rows, and builds the primary key. F0 is populated so that it contains every existing combination of L_1, …, Lj. Table F0 does not have any nonkey column.

```
INSERT INTO F0
SELECT DISTINCT L_1, …, Lj
FROM { F/F_V };
```

Tables F_1, …, F_d contain individual aggregations for each combination of R_1, …, R_k. The primary key of table F_I is { L_1, …, Lj }.

```
INSERT INTO F_I
SELECT L1, . . . , Lj, V(A)
FROM {Fj | F_V}
WHERE R_1 = v_1I AND .. AND R_k = v_kI
GROUP BY L_1, . . . , L_j;
```

A possible optimization is synchronizing table scans to compute the d tables in one pass. Finally, to get F_H we need d left outer joins with the d+1 tables so that all individual aggregations are properly assembled as a set of d dimensions for each group. Outer joins set result columns to null for missing combinations for the given group.

```
INSERT INTO F_H
SELECT
F0.L_1, F0.L_2, . . . , F0.L_j,
F_1.A, F_2.A, . . . , F_d.A
```

```
FROM F0
LEFT OUTER JOIN $F_1$
ON F0.$L_1$ = $F_1$.$L_1$ and . . . and F0.Lj = $F_1$.$L_j$
LEFT OUTER JOIN $F_2$
ON F0.$L_1$ = $F_2$.$L_1$ and . . . and F0.$L_j$ = $F_2$.$L_j$
…..
…..
LEFT OUTER JOIN $F_d$
ON F0.$L_1$ = $F_d$.$L_1$ and . . . and F0.$L_j$ = $F_d$.$L_j$;
```

Since F0 has n rows each left outer join produces a partial table with n rows and one additional column. Then at the end, F_H will have n rows and d aggregation columns.

3.3.2 CASE Method

For this method, we use the “case” programming construct available in SQL. The case statement returns a value selected from a set of values based on Boolean expressions. We propose two basic strategies to compute F_H. In a similar manner to SPJ, the first one directly aggregates from F and the second one computes the vertical aggregation in a temporary table F_V and then horizontal aggregations are indirectly computed from F_V.

We now present the direct aggregation method. Horizontal aggregation queries can be evaluated by directly aggregating from F and transposing rows at the same time to produce F_H. First, we need to get the unique combinations of R1,. . ., Rk that define the matching Boolean expression for result columns. The SQL code to compute horizontal aggregations directly from F is as follows:

```
SELECT DISTINCT $R_1$, . . ., $R_k$
FROM F;

INSERT INTO $F_H$
SELECT $L_1$, . . ., $L_J$
        , V (CASE WHEN $R_1$ = $v_{11}$ and . . . and $R_k$ = $v_{k1}$
                THEN A ELSE null END)
        ……
```

```
          , V (CASE WHEN $R_1$ = $v_{1d}$ and . . . and $R_k$ = $v_{kd}$
                    THEN A ELSE null END)
FROM F
GROUP BY $L_1$, $L_2$, . . . , $L_j$;
```

This statement computes aggregations in only one scan on F. The main difficulty is that there must be a feedback process to produce the "case" Boolean expressions.

We now consider an optimized version using F_V. Based on F_V, we need to transpose rows to get groups based on L_1, . . ., L_J. Query evaluation needs to combine the desired aggregation with "CASE" statements for each distinct combination of values of R_1, . . ., R_k. As explained above, horizontal aggregations must set the result to null when there are no qualifying rows for the specific horizontal group. The following statements compute F_H:

```
SELECT DISTINCT $R_1$, . . ., $R_k$
FROM $F_V$;

INSERT INTO $F_H$
SELECT L1, . . ., LJ
          , sum(CASE WHEN $R_1$ = $v_{11}$ and . . . and $R_k$ = $v_{k1}$
                    THEN A ELSE null END)
          …..
          , sum(CASE WHEN $R_1$ = $v_{1d}$ and . . . and $R_k$ = $v_{kd}$
                    THEN A ELSE null END)
FROM $F_V$
GROUP BY $L_1$, $L_2$, . . . , $L_j$;
```

It has the disadvantage of using two tables instead of one as required by the direct computation from F. For very large tables F computing FV first, may be more efficient than computing directly from F.

3.3.3 PIVOT Method

We consider the PIVOT operator which is a built-in operator in a commercial DBMS. Since this operator can perform transposition it can help evaluating horizontal aggregations. The PIVOT method internally needs to determine how many columns are needed to store the transposed table and it can be combined with the GROUP BY clause.

The basic syntax to exploit the PIVOT operator to compute a horizontal aggregation assuming one BY column for the right key columns (i.e., k = 1) is as follows:

```
SELECT DISTINCT R1
FROM F; /* produces v1, . . . , vd */

SELECT L1, L2, . . . , Lj,
          v1, v2, . . . , vd
INTO Ft
FROM F
PIVOT(
        V(A) FOR R1 in (v1, v2, . . . , vd)
) AS P;

SELECT
        L1, L2, . . . , Lj
        ,V (v1), V (v2), . . . , V (vd)
INTO FH
FROM Ft
GROUP BY L1, L2, . . . , Lj;
```

This set of queries may be inefficient because F_t can be a large intermediate table. We introduce the following optimized set of queries which reduces of the intermediate table:

```
SELECT DISTINCT R1
FROM F; /* produces v1, . . . , vd */

SELECT L1, L2, . . . , Lj,
          v1, v2, . . . , vd
```

```
INTO $F_H$
FROM (
          SELECT $L_1$, $L_2$, . . . , $L_j$, $R_1$, A
          FROM F) $F_t$
          PIVOT(
                    V(A) FOR $R_1$ in ($v_1$, $v_2$, . . . , $v_d$)
          ) AS P;
```

Here the first and second query can also be computed from F_V for optimization.

3.3.4 Example of Generated SQL Queries

We now show actual SQL code for our small example. This SQL code produces FH in Fig. 1. Notice the three methods can compute from either F or F_V, but we use F to make code more compact.

The SPJ method code is as follows (computed from F):

```
/* SPJ method */
INSERT INTO F1
SELECT D1, sum (A) AS A
FROM F
WHERE D2='X'
GROUP BY D1;

INSERT INTO F2
SELECT D1, sum (A) AS A
FROM F
WHERE D2='Y'
GROUP BY D1;

INSERT INTO FH
SELECT F0.D1, F1.A AS D2_X, F2.A AS D2_Y
FROM F0 LEFT OUTER JOIN F1 on F0.D1=F1.D1
LEFT OUTER JOIN F2 on F0.D1=F2.D1;
```

The CASE method code is as follows (computed from F):

```
/* CASE method */
```

```
INSERT INTO FH
SELECT
          D1
          , SUM (CASE WHEN D2='X' THEN A
          ELSE null END) as D2_X
          , SUM (CASE WHEN D2='Y' THEN A
          ELSE null END) as D2_Y
FROM F
GROUP BY D1;
```

Finally, the PIVOT method SQL is as follows (computed from F):

```
/* PIVOT method */
INSERT INTO FH
SELECT
          D1
          , [X] as D2_X
          , [Y] as D2_Y
FROM (
          SELECT D1, D2, A FROM F
) as p
PIVOT (
SUM (A)
FOR D2 IN ([X], [Y])
) as pvt;
```

3.4 Properties of Horizontal Aggregation

A horizontal aggregation exhibits the following properties:

1. $n = |F_H|$ matches the number of rows in a vertical aggregation grouped by $L_1, \ldots, L_j$.
2. $d = | \pi_{R1, \ldots, Rk} (F) |$.
3. Table F_H may potentially store more aggregated values than F_V due to nulls. That is, $|F_V| <= nd$.

3.5 DBMS Limitations

There exist two DBMS limitations with horizontal aggregations: reaching the maximum number of columns in one table and reaching the maximum column name length when columns are automatically named. To elaborate on this, a horizontal aggregation can return a table that goes beyond the maximum number of columns in the DBMS when the set of columns { R_1, …, R_K } has a large number of distinct combinations of values, or when there are multiple horizontal aggregations in the same query. On the other hand, the second important issue is automatically generating unique column names. If there are many subgrouping columns { R_1, …, R_K } or columns are of string data types, this may lead to generate very long column names, which may exceed DBMS limits.

The problem of d going beyond the maximum number of columns can be solved by vertically partitioning F_H so that each partition table does not exceed the maximum number of columns allowed by the DBMS. The column name length issue can be solved by generating column identifiers with integers and creating a “dimension” description table that maps identifiers to full descriptions, but the meaning of each dimension is lost.

Chapter 4

Integrating K-Means with Horizontal Aggregation

Clustering methods partition a set of objects into clusters such that objects in the same cluster are more similar to each other than objects in different clusters according to some defined criteria. Data mining applications frequently involve categorical data. The biggest advantage of these clustering algorithms is that it is scalable to very large data sets.

Even though the existing system presented the computation of the values for different attributes, it has some drawbacks. In the research of the horizontal aggregation, the existing systems are not well defined for the different fact tables that need better indexing and extraction.

Multiple fact tables: Constructing new data sets within the range of a discrete set of known data points we need different attributes from different fact tables. In many applications one often has a number of data values, obtained by experimentation, which stored on limited number of databases. It is often required to extract the particular useful attributes from the different fact tables and perform aggregation.

4.1 K-Means Algorithm

K-means is one of the simplest unsupervised learning algorithms that solve the well-known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed apriori. The main idea is to define k centers, one for each cluster. These centers should be placed in a cunning way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest center. When no point is pending, the first step is completed and an early group age is done. At this point we need to re-calculate k new centroids as barycenter of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new center. A loop has been generated. As a result of this loop we may notice that the k centers change their location step by step until no more changes are done or in other words centers do not move any more. Finally, this algorithm aims at minimizing an objective function know as squared error function given by:

where,

$$J(V) = \sum_{i=1}^{c} \sum_{j=1}^{c_i} (\|x_i - v_j\|)^2$$

'||xi - vj||' is the Euclidean distance between xi and vj.

'ci' is the number of data points in ith cluster.

'c' is the number of cluster centers.

Algorithmic steps for k-means clustering:

Let X = {x1,x2,x3,……..,xn} be the set of data points and V = {v1,v2,…….,vc} be the set of centers.

1) Randomly select 'c' cluster centers.

2) Calculate the distance between each data point and cluster centers.

3) Assign the data point to the cluster center whose distance from the cluster center is minimum of all the cluster centers..

4) Recalculate the new cluster center using:

$$v_i = (1/c_i) \sum_{j=1}^{c_i} x_i$$

where, 'ci' represents the number of data points in ith cluster.

5) Recalculate the distance between each data point and new obtained cluster centers.

6) If no data point was reassigned then stop, otherwise repeat from step 3).

4.2 Integrating with Horizontal Aggregation

The k means algorithm is executed on huge data sets that are resulted after the horizontal aggregation with a specific end goal to segment into diverse clusters. A sample with step by step procedure applied on a data set is given below. As a basic illustration of a k-means algorithm, think about the accompanying data set comprising of the scores of two variables on each of seven people:

Subject	A	B
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

This data set which is given above in the table is to be gathered into two clusters. As a first stage in uncovering a sensible initial partition, let the A & B values of the two people furthest apart (utilizing the Euclidean distance measure), characterize the initial cluster:

	Individual	Mean Vector(centroid)
Group1	1	(1.0,1.0)
Group2	4	(5.0,5.0)

The remaining data sets are presently analyzed in an arrangement and are distributed to distinctive cluster to which they are closest, which is calculated terms of Euclidean distance to the cluster mean. The point when mean vector is recalculated, each time another new member is included.

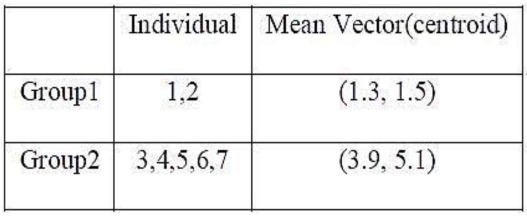
Now the initial partition is changed, and the two clusters at this stage have the following characteristics:

	Individual	Mean Vector(centroid)
Group1	1,2,3	(1.8, 2.3)
Group2	4,5,6,7	(4.1, 5.4)

Anyway we are not certain that every individual has been allocated to the right cluster. To realize that every individual is allocated to the right cluster, we contrast every individual's distance with its own particular cluster mean and to that of the inverse cluster.

Here just individual 3 is closer to the mean of the inverse cluster (Cluster 2) than its own (Cluster 1). At the end of the day, every individual's distance to its own cluster mean ought to be smaller than the distance to the other cluster's mean (which is not the situation with individual 3).

Therefore, individual 3 is relocated to Cluster 2 which results in the new partition:



	Individual	Mean Vector(centroid)
Group1	1,2	(1.3, 1.5)
Group2	3,4,5,6,7	(3.9, 5.1)

The iterative relocation might now proceed from this new segment until no more relocation occur. On the other hand, in the case given above every individual is closer its own particular cluster mean than that of the other cluster and the iteration steps, picking the most recent partitioning as the last cluster result. In this way the k means algorithm is performed on the data sets that are assessed from horizontal aggregations.

Chapter 5

Experimental Evaluation

In this section, we present our experimental evaluation on a commercial DBMS. We evaluate query optimizations, compare the three query evaluation methods, and analyse time complexity varying table sizes and output data set dimensionality.

5.1 Setup: Computer Configuration and Data Sets

We used Microsoft SQL Server 2012, running on a DBMS server running at 2.66 GHz, Core 2 Duo processor, 4 GB of RAM and 500 GB of hard disk. The SQL code generator was programmed in the C# language and connected to the server via .NET Framework Data Provider for SQL Server. The PIVOT operator was used as available in the SQL language implementation provided by the DBMS.

We used large synthetic data sets. We analyzed queries having only one horizontal aggregation, with different grouping and horizontalization columns. Each experiment was repeated many times and we report the average time in seconds. We cleared cache memory before each method started in order to evaluate query optimization under pessimistic conditions.

L_I	R_I	n	d
Transaction_id	dweek	1K	7
Transaction_id	month	1K	12
Transaction_id	department	1K	15
Transaction_id	dweek	10K	7
Transaction_id	month	10K	12
Transaction_id	department	10K	15
Transaction_id	dweek	50K	7
Transaction_id	month	50K	12
Transaction_id	department	50K	15

TABLE 2: Summary of grouping columns in database

We evaluated optimization strategies for aggregation queries with pseudo data sets generated by a C# Program written by us. This C# Program is used to insert a large number of rows with random generated values for each of the tuple into the database in less amount of time. We picked different value of d(number of distinct value in a transposing column) and n(number of rows after the group by clause) to store the data accordingly and analyze the performance of all the three methods used in this project. In order to get meaningful data sets for data mining we picked high selectivity columns for the left key and low selectivity columns for the right key. In this manner d << n, which is the most common scenario in data mining.

5.2 Comparing Evaluation Methods

Our goal is to assess the acceleration obtained by precomputing a cube and storing it on F_V. Table 3 shows that this optimization uniformly accelerates all methods. This optimization provides a different gain, depending on the method: for SPJ the optimization is best for small n, for PIVOT for large n and for CASE there is all across n. It is noteworthy PIVOT is accelerated by our optimization, despite the fact it is handled by the query optimizer. Since this optimization produces significant acceleration for the three methods (at least 2 X faster) we will use it by default. Since precomputing of F_V takes the same time within each method. Therefore, comparisons are fair. Removing columns that will not appear in F_H is significantly, accelerating evaluation time from two to three times. All our experiments incorporate this optimization by default.

n	**d**	**SPJ**		**CASE**		**PIVOT**	
		F	**F_V**	**F**	**F_V**	**F**	**F_V**
1K	**7**	0.26	0.15	0.29	0.18	0.28	0.16
	12	32	17	0.41	0.26	0.45	0.25
	15	41	24	0.51	0.30	0.49	0.27
10K	**7**	3.50	0.95	0.95	0.90	0.90	0.80
	12	43	33	1.26	1.15	1.22	1.10
	15	61	42	1.91	1.47	1.87	1.40
50K	**7**	25	3.46	6.41	3.10	6.52	3.20
	12	122	4.20	7.55	3.62	7.40	3.75
	15	243	4.85	8.10	3.95	7.75	3.89

TABLE 3: Query Optimization and comparing evaluation methods
(N = 10^5)
Mean Time in Seconds

In the time complexity and I/O cost analysis, the two main factors influencing query evaluation time are data set size and grouping columns. The three methods use the optimization precomputing F_V in order to make a uniform and fair comparison. In general, SPJ is the slowest method, as expected. On the other hand, PIVOT and CASE have similar time performance with slightly bigger differences in some cases. Time grows as n grows for all methods, but much more for SPJ. On the other hand, there is a small time growth when d increases for SPJ, but not a clear trend for PIVOT and CASE. n is the main performance factor for PIVOT and CASE methods, whereas both d and n impact the SPJ method.

Chapter 6

Source Code

```
<%--
      /PROJECT/add-transaction.aspx
--%>

<%@ Page Title="Add Transaction" Language="C#" MasterPageFile="~/storeuser.master" AutoEventWireup="true" CodeFile="add_data.aspx.cs" Inherits="add_data" %>
<%@ Register Assembly="AjaxControlToolkit" Namespace="AjaxControlToolkit" TagPrefix="ajaxtoolkit" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder3" runat="Server">
    <div style="width: 586px; height: 490px; background-image: url('img/addtransbg.jpg'); margin-bottom: 30px; padding-top: 8px; margin-top: 10px;">
        <h2 style="color: #008000; text-align: center; font-family: 'Lucida Sans'">Transactions</h2>
        <div style="width: 458px; margin: 0 auto;">
            <table style="width: 448px; margin-left: 35px;">
                <tr>
                    <td style="width: 142px">Transaction Id:</td>
                    <td>
                        <asp:TextBox ID="transactionid" runat="server" ValidationGroup="trans" Height="22px" Width="180px" BorderColor="#FFCC66" BorderStyle="Solid" BorderWidth="1px" Font-Size="17px"></asp:TextBox>
                    </td>
                    <td style="width: 116px">
                        <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server" ControlToValidate="transactionid" ErrorMessage="*" ForeColor="Red" ValidationGroup="trans">*</asp:RequiredFieldValidator>
                    </td>
                </tr>
                <tr>
                    <td style="width: 142px">Date:</td>
                    <td>
                        <ajaxtoolkit:ToolkitScriptManager ID="scrptmngr" runat="server"></ajaxtoolkit:ToolkitScriptManager>
                        <asp:TextBox ID="txtdate" runat="server" Height="22px" Width="180px" BorderColor="#FFCC66" BorderStyle="Solid" BorderWidth="1px" Font-Size="17px" ValidationGroup="trans"></asp:TextBox>
                    </td>
                    <td style="width: 116px">
                        <asp:ImageButton ID="imgcal" runat="server" Height="25px" Width="25px" ImageUrl="~/img/calicon.png" />
                        <ajaxtoolkit:CalendarExtender ID="txtdate_CalendarExtender" runat="server" BehaviorID="txtdate_CalendarExtender" TargetControlID="txtdate" PopupButtonID="imgcal" Format="dd/MM/yyyy">
                        </ajaxtoolkit:CalendarExtender>
                        <asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server" ControlToValidate="txtdate" ErrorMessage="*" ForeColor="Red" ValidationGroup="trans">*</asp:RequiredFieldValidator>
                    </td>
                </tr>
                <tr>
                    <td style="width: 142px">Weekday:</td>
                    <td>
                        <asp:DropDownList ID="dropdownday" runat="server" Height="28px" Width="120px" BorderColor="#FFCC66" BorderStyle="Solid" BorderWidth="1px" Font-Size="17px">
                            <asp:ListItem Selected="True">Monday</asp:ListItem>
                            <asp:ListItem>Tuesday</asp:ListItem>
                            <asp:ListItem>Wednesday</asp:ListItem>
                            <asp:ListItem>Thursday</asp:ListItem>
                            <asp:ListItem>Friday</asp:ListItem>
                            <asp:ListItem>Saturday</asp:ListItem>
                            <asp:ListItem>Sunday</asp:ListItem>
                        </asp:DropDownList>
                    </td>
                    <td style="width: 116px"></td>
                </tr>
                <tr>
                    <td style="width: 142px">Department:</td>
                    <td>
                        <asp:DropDownList ID="dropdowndept" runat="server" Height="28px" Width="167px" BorderColor="#FFCC66" BorderStyle="Solid" BorderWidth="1px" Font-Size="17px">
                            <asp:ListItem Selected="True">Electronics</asp:ListItem>
                            <asp:ListItem>Books</asp:ListItem>
                            <asp:ListItem>Clothing</asp:ListItem>
                            <asp:ListItem>Home_Appliances</asp:ListItem>
                        <asp:ListItem>Cosmetics</asp:ListItem>
                        <asp:ListItem>Food</asp:ListItem>
                        <asp:ListItem>Gardening</asp:ListItem>
                        <asp:ListItem>Sports</asp:ListItem>
                        <asp:ListItem>Paint</asp:ListItem>
                        <asp:ListItem>Movie_Games</asp:ListItem>
                        <asp:ListItem>Hardware</asp:ListItem>
                        <asp:ListItem>Jewelry</asp:ListItem>
                        <asp:ListItem>Baby_Toys</asp:ListItem>
                        <asp:ListItem>Health_Gourmet</asp:ListItem>
                    </asp:DropDownList>
                </td>
                <td style="width: 116px"></td>
            </tr>

            <tr>
                <td style="width: 142px">Number of Items:</td>
                <td>
                    <asp:TextBox ID="txtnumitems" runat="server" Height="22px" Width="120px" BorderColor="#FFCC66" BorderStyle="Solid" BorderWidth="1px" Font-Size="17px" ValidationGroup="trans"></asp:TextBox>
                </td>
                <td style="width: 116px">
                    <asp:RequiredFieldValidator ID="RequiredFieldValidator3" runat="server" ControlToValidate="txtnumitems" ErrorMessage="*" ForeColor="Red" ValidationGroup="trans">*</asp:RequiredFieldValidator>
                    <asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server" ControlToValidate="txtnumitems" ErrorMessage="*Integer" Font-Size="13px" ForeColor="Red" ValidationExpression="^[1-9]\d*$" ValidationGroup="trans"></asp:RegularExpressionValidator>
                </td>
            </tr>
            <tr>
                <td style="width: 142px">Sale Amount:</td>
                <td>
                    <asp:TextBox ID="saleamt" runat="server" ValidationGroup="trans" Height="22px" Width="180px" BorderColor="#FFCC66" BorderStyle="Solid" BorderWidth="1px" Font-Size="17px"></asp:TextBox>
                </td>
                <td style="width: 116px">
                    <asp:RequiredFieldValidator ID="RequiredFieldValidator4" runat="server" ControlToValidate="saleamt" ErrorMessage="*" ForeColor="Red" ValidationGroup="trans">*</asp:RequiredFieldValidator>
                    <asp:RegularExpressionValidator ID="RegularExpressionValidator2" runat="server" ControlToValidate="saleamt" ErrorMessage="Invalid Amount" Font-Size="13px" ForeColor="Red" ValidationExpression="^\d*\.?\d*$" ValidationGroup="trans"></asp:RegularExpressionValidator>
                </td>
            </tr>

            <tr>
                <td style="width: 142px; height: 15px;">-------------------------</td>
                <td style="height: 15px"></td>
                <td style="width: 116px; height: 15px;"></td>
            </tr>

            <tr>
                <td style="width: 142px">Customer Name:</td>
                <td>
                    <asp:TextBox ID="custname" runat="server" ValidationGroup="trans" Height="22px" Width="180px" BorderColor="#FFCC66" BorderStyle="Solid" BorderWidth="1px" Font-Size="17px"></asp:TextBox>
                </td>
                <td style="width: 116px">
                    <asp:RequiredFieldValidator ID="RequiredFieldValidator5" runat="server" ControlToValidate="custname" ErrorMessage="*" ForeColor="Red" ValidationGroup="trans">*</asp:RequiredFieldValidator>
                </td>
            </tr>
            <tr>
                <td style="width: 142px">Contact Number:</td>
                <td>
                    <asp:TextBox ID="txtmobile" runat="server" ValidationGroup="trans" Height="22px" Width="180px" BorderColor="#FFCC66" BorderStyle="Solid" BorderWidth="1px" Font-Size="17px"></asp:TextBox>
                    </td>
                    <td style="width: 116px">
                        <asp:RequiredFieldValidator ID="RequiredFieldValidator6" runat="server" ControlToValidate="txtmobile" ErrorMessage="*" ForeColor="Red" ValidationGroup="trans">*</asp:RequiredFieldValidator>
                    </td>
                </tr>
                <tr>
                    <td style="width: 142px">Pincode:</td>
                    <td>
                        <asp:TextBox ID="txtpin" runat="server" ValidationGroup="trans" Height="22px" Width="180px" BorderColor="#FFCC66" BorderStyle="Solid" BorderWidth="1px" Font-Size="17px"></asp:TextBox>
                    </td>
                    <td style="width: 116px">
                        <asp:RequiredFieldValidator ID="RequiredFieldValidator7" runat="server" ControlToValidate="txtpin" ErrorMessage="*" ForeColor="Red" ValidationGroup="trans">*</asp:RequiredFieldValidator>
                    </td>
                </tr>
                <tr>
                    <td colspan='2' align='center' class="auto-style9" style="height: 51px">
                        <asp:Button ID="btnsubmit" runat="server" Text="Submit" Font-Bold="False" Font-Size="15px" Width="105px" OnClick="btnsubmit_Click" ValidationGroup="trans" />
                    </td>
                    <td class="auto-style10" style="height: 51px; width: 116px"></td>
                </tr>
                <tr>
                    <td colspan='2' align='center' style="height: 27px">
                        <asp:Label ID="labelresult" runat="server"></asp:Label>
                    </td>
                    <td></td>
                </tr>
            </table>
        </div>
    </div>

</asp:Content>
```

```csharp
/*
 *    PROJECT/admin/add-transaction.aspx.cs
 */
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.IO;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;

public partial class add_data : System.Web.UI.Page
{
    public static string constring = ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString;
    public static SqlConnection con = new SqlConnection(constring);
    protected void Page_Load(object sender, EventArgs e)
    {
        if (Session["User"] == "NotAvailable")
        {
            Response.Redirect("log-in.aspx");
        }
    }
    protected void btnsubmit_Click(object sender, EventArgs e)
    {
        string storeid = Session["User"].ToString();
        int numitem = Int32.Parse(txtnumitems.Text);
        double saleamount = Convert.ToDouble(saleamt.Text);
        string caldate = txtdate.Text;
        string[] splitdate = caldate.Split('/');
        string[] montharr = { "Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec" };
        int kk = Int32.Parse(splitdate[1]);
        string month = montharr[kk-1];
        labelresult.Text = "";
        try
        {
            SqlCommand cmd = new SqlCommand("insert into querytable (storeid, transid, day, month, year, date, weekday, department, numitems, saleamt, custname, contnum, pincode) values ('" + storeid + "', '" + transactionid.Text + "', '" + splitdate[0] + "', '" + month + "', '" + splitdate[2] + "', '" + caldate + "', '" + dropdownday.SelectedItem.Text + "', '" + dropdowndept.SelectedItem.Text + "', '" + numitem + "', '" + saleamount + "', '" + custname.Text + "', '" + txtmobile.Text + "', '" + txtpin.Text + "')", con);
            if (con.State == ConnectionState.Closed)
                con.Open();

            if (con.State == ConnectionState.Open)
                cmd.ExecuteNonQuery();

            if (con.State == ConnectionState.Open)
                con.Close();

            labelresult.ForeColor = System.Drawing.Color.Green;
            labelresult.Text = "Successfully Added";

        }
        catch (Exception ex)
        {
            labelresult.ForeColor = System.Drawing.Color.Red;
            labelresult.Text = "Some Error Occured Try Again";
        }

    }
}
```

```
<%--
      /PROJECT/execute-query.aspx
--%>

<%@ Page Title="Execute Query" Language="C#" MasterPageFile="~/storeuser.master" AutoEventWireup="true" CodeFile="executequery.aspx.cs" Inherits="executequery" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder3" Runat="Server">
    <asp:Label ID="Label1" runat="server" Text="Aggregation Type  "></asp:Label>
    <asp:DropDownList ID="dropdownaggregation" runat="server">
        <asp:ListItem Value="sum">SUM</asp:ListItem>
        <asp:ListItem Value="min">Minimum</asp:ListItem>
        <asp:ListItem Value="max">Maximum</asp:ListItem>
        <asp:ListItem Value="avg">Average</asp:ListItem>
    </asp:DropDownList>
    <asp:Label ID="Label2" runat="server" Text="Group By  "></asp:Label>
    <asp:DropDownList ID="dropdowngroupby" runat="server" AutoPostBack="True" OnSelectedIndexChanged="dropdowngroupby_SelectedIndexChanged" CausesValidation="True" ValidationGroup="query">
        <asp:ListItem Selected="True">--Select--</asp:ListItem>
        <asp:ListItem Value="month">Month</asp:ListItem>
        <asp:ListItem Value="year">Year</asp:ListItem>
        <asp:ListItem Value="department">Department</asp:ListItem>
    </asp:DropDownList>
    <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server" ErrorMessage="*" Text="*" Font-Strikeout="False" ControlToValidate="dropdowngroupby" ForeColor="Red" InitialValue="--Select--" ValidationGroup="query"></asp:RequiredFieldValidator>
    <asp:Label ID="Label3" runat="server" Text="Year  "></asp:Label>
    <asp:DropDownList ID="dropdownyear" runat="server" DataSourceID="SqlDataSource1" DataTextField="year" DataValueField="year" AutoPostBack="True">
    </asp:DropDownList>
    <asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:ConnectionString %>" SelectCommand="SELECT DISTINCT [year] FROM [querytable] WHERE ([storeid] = @storeid)">
        <SelectParameters>
            <asp:SessionParameter Name="storeid" SessionField="User" Type="String" />
        </SelectParameters>
    </asp:SqlDataSource>
    <div style="margin-top:20px;">
        <asp:Button ID="Button1" runat="server" Text="Execute" OnClick="Button1_Click" ValidationGroup="query" />
    </div>
    <div style="margin-top:20px; margin-left:120px; ">
        <asp:GridView ID="GridView1" runat="server" CellPadding="4" ForeColor="#333333">
            <AlternatingRowStyle BackColor="White" />
            <EditRowStyle BackColor="#2461BF" />
            <FooterStyle BackColor="#507CD1" Font-Bold="True" ForeColor="White" />
            <HeaderStyle BackColor="#507CD1" Font-Bold="True" ForeColor="White" />
            <PagerStyle BackColor="#2461BF" ForeColor="White" HorizontalAlign="Center" />
            <RowStyle BackColor="#EFF3FB" HorizontalAlign="Center" Width="100px" />
            <SelectedRowStyle BackColor="#D1DDF1" Font-Bold="True" ForeColor="#333333" />
            <SortedAscendingCellStyle BackColor="#F5F7FB" />
            <SortedAscendingHeaderStyle BackColor="#6D95E1" />
            <SortedDescendingCellStyle BackColor="#E9EBEF" />
            <SortedDescendingHeaderStyle BackColor="#4870BE" />
        </asp:GridView>
    </div>
</asp:Content>
```

```csharp
/*
 *    PROJECT/admin/execute-query.aspx.cs
 */
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.IO;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;

public partial class executequery : System.Web.UI.Page
{
    public static string constring = ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString;
    public static SqlConnection con = new SqlConnection(constring);
    protected void Page_Load(object sender, EventArgs e)
    {
        if (Session["User"] == "NotAvailable")
        {
            Response.Redirect("log-in.aspx");
        }
    }
    protected void dropdowngroupby_SelectedIndexChanged(object sender, System.EventArgs e)
    {
        if (dropdowngroupby.SelectedItem.Text.ToString() == "Month")
        {
            dropdownyear.Visible = true; Label3.Visible = true;
        }
        else
        {
            dropdownyear.Visible = false; Label3.Visible = false;
        }
    }
    protected void Button1_Click(object sender, System.EventArgs e)
    {
        string storeid = Session["User"].ToString();
        DataSet ds = new DataSet();
        con.Open(); SqlCommand cmd;
        if (dropdownyear.Visible == false)
        {
            if (dropdownaggregation.SelectedItem.Text.ToString() == "SUM" && dropdowngroupby.SelectedItem.Text.ToString() == "Year")
            {
                cmd = new SqlCommand("select year as Year, sum(numitems) as Total_Items, sum(saleamt) as Total_Sale_Amt from querytable where storeid='" + storeid + "' group by year", con);
            }
            else if (dropdownaggregation.SelectedItem.Text.ToString() == "Minimum" && dropdowngroupby.SelectedItem.Text.ToString() == "Year")
            {
                cmd = new SqlCommand("select year as Year, min(saleamt) as Minimum_Sale_Amt from querytable where storeid='" + storeid + "' group by year", con);
            }
            else if (dropdownaggregation.SelectedItem.Text.ToString() == "Maximum" && dropdowngroupby.SelectedItem.Text.ToString() == "Year")
            {
                cmd = new SqlCommand("select year as Year, max(saleamt) as Maximum_Sale_Amt from querytable where storeid='" + storeid + "' group by year", con);
            }
            else if (dropdownaggregation.SelectedItem.Text.ToString() == "Average" && dropdowngroupby.SelectedItem.Text.ToString() == "Year")
            {
                cmd = new SqlCommand("select year as Year, avg(saleamt) as Average_Sale_Amt from querytable where storeid='" + storeid + "' group by year", con);
            }
            else if (dropdownaggregation.SelectedItem.Text.ToString() == "SUM" && dropdowngroupby.SelectedItem.Text.ToString() == "Department")
            {
                cmd = new SqlCommand("select department as Department, sum(numitems) as Total_Items, sum(saleamt) as Total_Sale_Amt from querytable where storeid='" + storeid + "' group by department", con);
            }
            else if (dropdownaggregation.SelectedItem.Text.ToString() == "Minimum" && dropdowngroupby.SelectedItem.Text.ToString() == "Department")
            {
                cmd = new SqlCommand("select department as Department, min(saleamt) as Minimum_Sale_Amt from querytable where storeid='" + storeid + "' group by department", con);
            }
            else if (dropdownaggregation.SelectedItem.Text.ToString() == "Maximum" && dropdowngroupby.SelectedItem.Text.ToString() == "Department")
            {
                cmd = new SqlCommand("select department as Department, max(saleamt) as Maximum_Sale_Amt from querytable where storeid='" + storeid + "' group by department", con);
            }
            else
            {
                cmd = new SqlCommand("select department as Department, avg(saleamt) as Average_Sale_Amt from querytable where storeid='" + storeid + "' group by department", con);
            }

        }
        else
        {
            string year = dropdownyear.SelectedItem.Text.ToString();
            if (dropdownaggregation.SelectedItem.Text.ToString() == "SUM")
            {
                cmd = new SqlCommand("select month as Month, sum(numitems) as Total_Items, sum(saleamt) as Total_Sale_Amt from querytable where storeid='" + storeid + "' and year='" + year + "' group by month", con);
            }
            else if (dropdownaggregation.SelectedItem.Text.ToString() == "Minimum")
            {
                cmd = new SqlCommand("select month as Month, min(saleamt) as Minimum_Sale_Amt from querytable where storeid='" + storeid + "' and year='" + year + "' group by month", con);
            }
            else if (dropdownaggregation.SelectedItem.Text.ToString() == "Maximum")
            {
                cmd = new SqlCommand("select month as Month, max(saleamt) as Maximum_Sale_Amt from querytable where storeid='" + storeid + "' and year='" + year + "' group by month", con);
            }
            else
            {
                cmd = new SqlCommand("select month as Month, avg(saleamt) as Average_Sale_Amt from querytable where storeid='" + storeid + "' and year='" + year + "' group by month", con);
            }
        }
        SqlDataAdapter da = new SqlDataAdapter(cmd);
        da.Fill(ds);
        con.Close();
        GridView1.DataSource = ds;
        GridView1.DataBind();
    }
}
```

```
/*
 * PROJECT/admin/adminlogin.aspx.cs
 */
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.IO;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;
using System.Drawing;

public partial class admin_adminlogin : System.Web.UI.Page
{
    public static string constring = ConfigurationManager.ConnectionStrings["ConnectionString"].   ↙
    ConnectionString;
    public static SqlConnection con = new SqlConnection(constring);
    protected void Page_Load(object sender, EventArgs e)
    {
        if (Session["AdminUser"] != "NotAvailable")
        {
            Response.Redirect("Default.aspx");
        }
    }
    protected void ImageButton1_Click(object sender, ImageClickEventArgs e)
    {
        Response.Redirect("~/Default.aspx");
    }
    private Boolean CheckUser(string Name)
    {
        SqlParameter p1;
        SqlCommand cmd = new SqlCommand("select name from login where name=@name", con);
        p1 = new SqlParameter();
        p1.ParameterName = "@name";
        p1.Value = Name;
        p1.SqlDbType = SqlDbType.VarChar;
        p1.Size = 50;
        p1.Direction = ParameterDirection.InputOutput;
        cmd.Parameters.Add(p1);
        string str = "";
        try
        {
            if (cmd.Connection.State != ConnectionState.Open)
            {
                cmd.Connection.Open();
            }
            str = cmd.ExecuteScalar().ToString();
        }
        catch (Exception ex)
        {
            return false;
        }
        finally
        {
            if (cmd.Connection.State != ConnectionState.Closed)
            {
                cmd.Connection.Close();
            }
        }

        if (str != "")
            return true;

        else
            return false;
    }
    public string CheckAuthentication(string Name, string Password)
    {
        if (CheckUser(Name))
        {
            SqlParameter p1, p2;
            SqlCommand cmd = new SqlCommand("select name, pass from login where name=@name AND   ↙
    pass=@pass", con);
```

```
            p1 = new SqlParameter();
            p1.ParameterName = "@name";
            p1.Value = Name;
            p1.SqlDbType = SqlDbType.VarChar;
            p1.Size = 50;
            p1.Direction = ParameterDirection.InputOutput;
            cmd.Parameters.Add(p1);

            p2 = new SqlParameter();
            p2.ParameterName = "@pass";
            p2.Value = Password;
            p2.SqlDbType = SqlDbType.VarChar;
            p2.Size = 50;
            p2.Direction = ParameterDirection.InputOutput;
            cmd.Parameters.Add(p2);

            DataTable dt = new DataTable();
            try
            {
                if (cmd.Connection.State != ConnectionState.Open)
                {
                    cmd.Connection.Open();
                }
                SqlDataReader reader = cmd.ExecuteReader();
                dt.Load(reader);
            }
            catch (Exception ex)
            {
                throw new Exception("Invalid Command !" + ex.Message);
            }
            finally
            {
                if (cmd.Connection.State != ConnectionState.Closed)
                {
                    cmd.Connection.Close();
                }
            }

            if (dt.Rows.Count > 0)
            {
                string sUser = dt.Rows[0]["name"].ToString();
                return sUser;
            }
            else
            {
                return "Fail";
            }

        }
        else
        {
            return "Fail";
        }
    }
    protected void btnLogin_Click(object sender, EventArgs e)
    {
        string strUserName = txtUserName.Text;
        string strPassword = txtPassword.Text;
        string sResult = CheckAuthentication(strUserName, strPassword);
        if (sResult == "Fail")
        {
            lblError.Text = "<li>Either UserName or Password is incorrect!</li>";
        }
        else
        {
            Session["AdminUser"] = sResult;
            Response.Redirect("home.aspx");
        }
    }
}
```

```
<%--
    /PROJECT/admin/add-store.aspx
--%>

<%@ Page Title="Add Store" Language="C#" MasterPageFile="~/admin/admin1.master" AutoEventWireup="true" CodeFile="add-store.aspx.cs" Inherits="admin_add_store" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="Server">
    <div style="width: 500px; background-image: url('../img/registerbg.png'); margin-left: auto; margin-right: auto; margin-bottom: 30px; padding-top: 8px; overflow: auto">
        <h2 style="color: #FFFFFF; text-align: center; font-family: 'Lucida Sans'">Add Store</h2>
        <div style="width: 430px; margin: 0 auto;">
            <table>
                <tr>
                    <td style="color: white; font-family: 'Lucida Sans'">Store Id:
                    </td>
                    <td>
                        <asp:TextBox ID="storeid" runat="server" ValidationGroup="profile" Height="22px" Width="180px" BorderColor="#FFCC66" BorderStyle="Solid" BorderWidth="1px" Font-Size="17px"></asp:TextBox>
                    </td>
                    <td>
                        <asp:RequiredFieldValidator ID="RequiredFieldValidator11" runat="server" ControlToValidate="storeid" ErrorMessage="*" ForeColor="Red" ValidationGroup="reg">*</asp:RequiredFieldValidator>
                    </td>
                </tr>
                <tr>
                    <td style="color: white; font-family: 'Lucida Sans'">Store Name:
                    </td>
                    <td>
                        <asp:TextBox ID="storename" runat="server" ValidationGroup="profile" Height="22px" Width="180px" BorderColor="#FFCC66" BorderStyle="Solid" BorderWidth="1px" Font-Size="17px"></asp:TextBox>
                    </td>
                    <td>
                        <asp:RequiredFieldValidator ID="RequiredFieldValidator12" runat="server" ControlToValidate="storename" ErrorMessage="*" ForeColor="Red" ValidationGroup="reg">*</asp:RequiredFieldValidator>
                    </td>
                </tr>
                <tr>
                    <td style="color: white; font-family: 'Lucida Sans'">Email ID:
                    </td>
                    <td>
                        <asp:TextBox ID="txtemail" runat="server" ValidationGroup="profile" Height="22px" Width="180px" BorderColor="#FFCC66" BorderStyle="Solid" BorderWidth="1px" Font-Size="17px"></asp:TextBox>
                    </td>
                    <td>
                        <asp:RequiredFieldValidator ID="RequiredFieldValidator6" runat="server" ControlToValidate="txtemail"
                            ErrorMessage="*" ForeColor="Red" ValidationGroup="reg"></asp:RequiredFieldValidator>
                        <asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server" ControlToValidate="txtemail"
                            ErrorMessage="Invalid Email" ForeColor="Red" ValidationExpression="\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*" ValidationGroup="reg" Font-Size="13px"></asp:RegularExpressionValidator>
                    </td>
                </tr>
                <tr>
                    <td style="color: white; font-family: 'Lucida Sans'">Mobile No:
                    </td>
                    <td>
                        <asp:TextBox ID="txtmob" runat="server" ValidationGroup="profile" Height="22px" Width="180px" BorderColor="#FFCC66" BorderStyle="Solid" BorderWidth="1px" Font-Size="17px"></asp:TextBox>
                    </td>
                    <td>
                        <asp:RequiredFieldValidator ID="RequiredFieldValidator13" runat="server" ControlToValidate="txtmob" ErrorMessage="*" ForeColor="Red" ValidationGroup="reg">*</asp:RequiredFieldValidator>
                        <asp:RegularExpressionValidator ID="RegularExpressionValidator2" runat="server" ErrorMessage="*" ForeColor="Red" ValidationExpression="\d{10}" ControlToValidate="txtmob" ValidationGroup="reg" Font-Size="13px">Invalid Number</asp:
```

```
RegularExpressionValidator>
                </td>
            </tr>
            <tr>
                <td style="color: white; font-family: 'Lucida Sans'">Password:
                </td>
                <td>
                    <asp:TextBox ID="txtpass" runat="server" TextMode="Password" Height="22px" Width="180px" BorderColor="#FFCC66" BorderStyle="Solid" BorderWidth="1px" Font-Size="17px"></asp:TextBox>
                </td>
                <td>
                    <asp:RequiredFieldValidator ID="RequiredFieldValidator9" runat="server" ControlToValidate="txtpass"
                        ErrorMessage="*" ForeColor="Red" ValidationGroup="reg">*</asp:RequiredFieldValidator>
                </td>
            </tr>
            <tr>
                <td style="color: white; font-family: 'Lucida Sans'; width: 140px;">Retype Password:
                </td>
                <td>
                    <asp:TextBox ID="txtrepass" runat="server" TextMode="Password" Height="22px" Width="180px" BorderColor="#FFCC66" BorderStyle="Solid" BorderWidth="1px" Font-Size="17px"></asp:TextBox>
                </td>
                <td>
                    <asp:RequiredFieldValidator ID="RequiredFieldValidator10" runat="server" ControlToValidate="txtrepass"
                        ErrorMessage="*" ForeColor="Red" ValidationGroup="reg">*</asp:RequiredFieldValidator>
                </td>
            </tr>
            <tr>
                <td style="color: white; font-family: 'Lucida Sans'">Address:
                </td>
                <td>
                    <asp:TextBox ID="txtaddress" runat="server" TextMode="MultiLine" Height="41px" Width="180px" BorderColor="#FFCC66" BorderStyle="Solid" BorderWidth="1px" Font-Size="17px"></asp:TextBox>
                </td>
                <td>
                    <asp:RequiredFieldValidator ID="RequiredFieldValidator3" runat="server" ControlToValidate="txtaddress"
                        ErrorMessage="*" ForeColor="Red" ValidationGroup="reg">*</asp:RequiredFieldValidator>
                </td>
            </tr>
            <tr>
                <td style="color: white; font-family: 'Lucida Sans'">City:
                </td>
                <td>
                    <asp:TextBox ID="city" runat="server" ValidationGroup="profile" Height="22px" Width="180px" BorderColor="#FFCC66" BorderStyle="Solid" BorderWidth="1px" Font-Size="17px"></asp:TextBox>
                </td>
                <td>
                    <asp:RequiredFieldValidator ID="RequiredFieldValidator5" runat="server" ControlToValidate="city"
                        ErrorMessage="*" ForeColor="Red" ValidationGroup="reg">*</asp:RequiredFieldValidator>
                </td>
            </tr>
            <tr>
                <td style="color: white; font-family: 'Lucida Sans'">State:
                </td>
                <td>
                    <asp:TextBox ID="state" runat="server" ValidationGroup="profile" Height="22px" Width="180px" BorderColor="#FFCC66" BorderStyle="Solid" BorderWidth="1px" Font-Size="17px"></asp:TextBox>
                </td>
                <td>
                    <asp:RequiredFieldValidator ID="RequiredFieldValidator4" runat="server" ControlToValidate="state"
```

```
                            ErrorMessage="*" ForeColor="Red" ValidationGroup="reg">*</asp:RequiredFieldValidator>
                    </td>
                </tr>
                <tr>
                    <td style="color: white; font-family: 'Lucida Sans'">Zip Code:
                    </td>
                    <td>
                        <asp:TextBox ID="zipcode" runat="server" ValidationGroup="profile" Height="22px" Width="180px" BorderColor="#FFCC66" BorderStyle="Solid" BorderWidth="1px" Font-Size="17px"></asp:TextBox>
                    </td>
                    <td>
                        <asp:RequiredFieldValidator ID="RequiredFieldValidator7" runat="server" ControlToValidate="zipcode"
                            ErrorMessage="*" ForeColor="Red" ValidationGroup="reg">*</asp:RequiredFieldValidator>
                        <asp:RegularExpressionValidator ID="RegularExpressionValidator3" runat="server" ErrorMessage="zipcode invalid" ForeColor="Red" ValidationExpression="\d{6}" ControlToValidate="zipcode" ValidationGroup="reg" Font-Size="13px"></asp:RegularExpressionValidator>
                    </td>
                </tr>
                <tr>
                    <td style="color: white; font-family: 'Lucida Sans'">Photo Id:
                    </td>
                    <td>
                        <asp:FileUpload ID="photoupload" runat="server" Height="22px" Width="180px" BorderColor="#FFCC66" BorderStyle="Solid" BorderWidth="1px" />
                    </td>
                </tr>
                <tr>
                    <td colspan='2' align='center' class="auto-style9" style="height: 50px">
                        <asp:Button ID="submit" runat="server" Text="Submit" ValidationGroup="reg" Font-Size="15px" Width="105px" OnClick="submit_Click" />
                    </td>
                    <td style="height: 50px"></td>
                </tr>
                <tr>
                    <td colspan='3' align='center'>
                        <asp:Label ID="Label1" runat="server" Text=""></asp:Label>
                    </td>
                </tr>
            </table>
        </div>
    </div>
</asp:Content>
```

```
/*
 * PROJECT/admin/add-store.aspx.cs
 */
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.IO;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;

public partial class admin_add_store : System.Web.UI.Page
{
    public static string constring = ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString;
    public static SqlConnection con = new SqlConnection(constring);
    protected void Page_Load(object sender, EventArgs e)
    {
        if (Session["AdminUser"] == "NotAvailable")
        {
            Response.Redirect("adminlogin.aspx");
        }
    }

    protected void submit_Click(object sender, EventArgs e)
    {
        if (txtpass.Text == txtrepass.Text)
        {
            Label1.Text = "";
            if (checkid(storeid.Text) == 0)
            {
                if (checkemail(txtemail.Text) == 0)
                {
                    if (photoupload.HasFile)
                    {
                        try
                        {
                            string strname = photoupload.FileName.ToString();
                            photoupload.PostedFile.SaveAs(Server.MapPath("~/photoid/") + strname);
                            SqlCommand cmd = new SqlCommand("insert into store (storeid, sname, email, mobile, password, address, city, state, zipcode, photoid) values ('" + storeid.Text + "', '" + storename.Text + "', '" + txtemail.Text + "', '" + txtmob.Text + "', '" + txtpass.Text + "', '" + txtaddress.Text + "', '" + city.Text + "', '" + state.Text + "', '" + zipcode.Text + "' , '" + strname + "')", con);
                            if (con.State == ConnectionState.Closed)
                            {
                                con.Open();
                            }
                            if (con.State == ConnectionState.Open)
                            {
                                cmd.ExecuteNonQuery();
                            }
                            if (con.State == ConnectionState.Open)
                            {
                                con.Close();
                            }
                            Label1.ForeColor = System.Drawing.Color.Green;
                            Label1.Text = "Store Successfully Added";
                        }
                        catch (Exception ex)
                        {
                            Label1.ForeColor = System.Drawing.Color.Red;
                            Label1.Text = "Some Error Occured Try Again";
                        }
                    }
                    else
                    {
                        Label1.ForeColor = System.Drawing.Color.Red;
                        Label1.Text = "Please Upload The Photo Id Proof";
                    }

                }
                else
                {
                    Label1.ForeColor = System.Drawing.Color.Red;
                    Label1.Text = "Email already registered";
                }
            }
            else
            {
                Label1.ForeColor = System.Drawing.Color.Red;
                Label1.Text = "Id already Registered";
            }
        }
        else
        {
            Label1.ForeColor = System.Drawing.Color.Red;
            Label1.Text = "Password and Retype Password must be same";
        }
    }
    public static int checkid(string r)
    {
        int chk = 0;
        SqlCommand cmd = new SqlCommand("select storeid from store", con);
        if (con.State == ConnectionState.Closed)
        {
            con.Open();
        }
        if (con.State == ConnectionState.Open)
        {
            SqlDataReader dr = cmd.ExecuteReader();
            while (dr.Read())
            {
                if (r == dr["storeid"].ToString().Trim())
                    chk = 1;
            }
            dr.Close();
        }
        if (con.State == ConnectionState.Open)
        {
            con.Close();
        }
        return chk;
    }
    public static int checkemail(string r)
    {
        int chk = 0;
        SqlCommand cmd = new SqlCommand("select email from store", con);
        if (con.State == ConnectionState.Closed)
        {
            con.Open();
        }
        if (con.State == ConnectionState.Open)
        {
            SqlDataReader dr = cmd.ExecuteReader();
            while (dr.Read())
            {
                if (r == dr["email"].ToString().Trim())
                    chk = 1;
            }
            dr.Close();
        }
        if (con.State == ConnectionState.Open)
        {
            con.Close();
        }
        return chk;
    }
}
```

```aspx
<%--
    /PROJECT/admin/store-transaction.aspx
--%>

<%@ Page Title="Store Transaction" Language="C#" MasterPageFile="~/admin/admin1.master" AutoEventWireup="true" CodeFile="store-transaction.aspx.cs" Inherits="admin_store_transaction" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="Server">
    <div style="margin-left: 290px;">
        <asp:Label ID="Label1" runat="server" Text="Select Store Id:  "></asp:Label>
        <asp:DropDownList ID="dropdownstore" runat="server" AutoPostBack="True" OnSelectedIndexChanged="dropdownstore_SelectedIndexChanged"></asp:DropDownList>
    </div>
    <div style="text-align: center; color: #008000; font-size: 18px; font-weight: bold; margin-top: 20px">
        <asp:Label ID="Label2" runat="server" Text="Transactions"></asp:Label>
    </div>
    <div style="margin-top: 20px; float: right; width: 810px; margin-bottom: 20px; overflow: auto">
        <asp:GridView ID="gvtransaction" runat="server" BackColor="White" BorderColor="#E7E7FF" BorderStyle="None" BorderWidth="1px" CellPadding="3" GridLines="Vertical" Font-Size="15px" HorizontalAlign="Center" Width="650px">
            <AlternatingRowStyle BackColor="#F7F7F7" />
            <FooterStyle BackColor="#B5C7DE" ForeColor="#4A3C8C" />
            <HeaderStyle BackColor="#4A3C8C" Font-Bold="True" ForeColor="#F7F7F7" HorizontalAlign="Center" />
            <PagerStyle BackColor="#E7E7FF" ForeColor="#4A3C8C" HorizontalAlign="Right" />
            <RowStyle BackColor="#E7E7FF" ForeColor="#4A3C8C" HorizontalAlign="Center" />
            <SelectedRowStyle BackColor="#738A9C" Font-Bold="True" ForeColor="#F7F7F7" />
            <SortedAscendingCellStyle BackColor="#F4F4FD" />
            <SortedAscendingHeaderStyle BackColor="#5A4C9D" />
            <SortedDescendingCellStyle BackColor="#D8D8F0" />
            <SortedDescendingHeaderStyle BackColor="#3E3277" />
        </asp:GridView>
    </div>
    <div>
        <p>.</p>
    </div>
</asp:Content>
```

```csharp
/*
 * PROJECT/admin/store-transaction.aspx.cs
 */
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.IO;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;

public partial class admin_store_transaction : System.Web.UI.Page
{
    public static string constring = ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString;
    public static SqlConnection con = new SqlConnection(constring);
    protected void Page_Load(object sender, EventArgs e)
    {
        if (Session["AdminUser"] == "NotAvailable")
        {
            Response.Redirect("adminlogin.aspx");
        }
        Label2.Visible = false;
        if (!this.IsPostBack)
        {
            SqlCommand cmd = new SqlCommand("select storeid, sname from store", con);
            if (con.State == ConnectionState.Closed) con.Open();
            if (con.State == ConnectionState.Open)
            {
                dropdownstore.DataSource = cmd.ExecuteReader();
                dropdownstore.DataTextField = "storeid";
                dropdownstore.DataValueField = "storeid";
                dropdownstore.DataBind();
            }
            if (con.State == ConnectionState.Open) con.Close();
            dropdownstore.Items.Insert(0, new ListItem("--Select--", "0"));
        }
    }

    protected void dropdownstore_SelectedIndexChanged(object sender, EventArgs e)
    {
        if (dropdownstore.SelectedItem.Text.ToString().Equals("--Select--"))
        {
            gvtransaction.DataSource = null; gvtransaction.DataBind(); Label2.Visible = false;
        }
        else
        {
            string storeid = dropdownstore.SelectedItem.Text.ToString();
            Label2.Visible = true;
            DataSet ds = new DataSet();
            SqlCommand cmd = new SqlCommand("select transid as Transaction_Id,date as Date, department as Department,numitems as Items,saleamt as Sale_Amount from querytable where storeid ='"+ storeid +"'", con);
            if (con.State == ConnectionState.Closed)
            {
                con.Open();
            }
            SqlDataAdapter da = new SqlDataAdapter(cmd);
            da.Fill(ds);
            if (con.State == ConnectionState.Open)
            {
                con.Close();
            }
            gvtransaction.DataSource = ds;
            gvtransaction.DataBind();
        }
    }
}
```

```
--Method : SPJ; Transposing Column: weekday; Aggregation Column: Number of Item; Use of Fv : NO
-----------------------------------------------------------------------------------------------
create view spj_day_num as
select f1.storeid, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday
from
(select storeid, SUM(numitems) as Monday from querytable where weekday='Monday'
group by storeid) f1 ,
(select storeid, SUM(numitems) as Tuesday from querytable where weekday='Tuesday'
group by storeid) f2 ,
(select storeid, SUM(numitems) as Wednesday from querytable where weekday='Wednesday'
group by storeid) f3 ,
(select storeid, SUM(numitems) as Thursday from querytable where weekday='Thursday'
group by storeid) f4 ,
(select storeid, SUM(numitems) as Friday from querytable where weekday='Friday'
group by storeid) f5 ,
(select storeid, SUM(numitems) as Saturday from querytable where weekday='Saturday'
group by storeid) f6 ,
(select storeid, SUM(numitems) as Sunday from querytable where weekday='Sunday'
group by storeid) f7 where
f1.storeid=f2.storeid and f2.storeid=f3.storeid and f3.storeid=f4.storeid and f4.storeid=f5.storeid ↙
     and f5.storeid=f6.storeid and f6.storeid=f7.storeid;


--Method: SPJ; Transposing Column:Month; Aggregation Column:Number of Item; Use of Fv:NO
----------------------------------------------------------------------------------------------
create view spj_month_num as
select f1.storeid, Jan,Feb,Mar,Apr,May, Jun, Jul, Aug, Sep, Oct, Nov, Dec
from
(select storeid, SUM(numitems) as Jan from querytable where month='Jan' group by storeid) f1 ,
(select storeid, SUM(numitems) as Feb from querytable where month='Feb' group by storeid) f2 ,
(select storeid, SUM(numitems) as Mar from querytable where month='Mar' group by storeid) f3 ,
(select storeid, SUM(numitems) as Apr from querytable where month='Apr' group by storeid) f4 ,
(select storeid, SUM(numitems) as May from querytable where month='May' group by storeid) f5 ,
(select storeid, SUM(numitems) as Jun from querytable where month='Jun' group by storeid) f6 ,
(select storeid, SUM(numitems) as Jul from querytable where month='Jul' group by storeid) f7 ,
(select storeid, SUM(numitems) as Aug from querytable where month='Aug' group by storeid) f8 ,
(select storeid, SUM(numitems) as Sep from querytable where month='Sep' group by storeid) f9 ,
(select storeid, SUM(numitems) as Oct from querytable where month='Oct' group by storeid) f10 ,
(select storeid, SUM(numitems) as Nov from querytable where month='Nov' group by storeid) f11 ,
(select storeid, SUM(numitems) as Dec from querytable where month='Dec' group by storeid) f12
where
f1.storeid=f2.storeid and f2.storeid=f3.storeid and f3.storeid=f4.storeid and
f4.storeid=f5.storeid and f5.storeid=f6.storeid and f6.storeid=f7.storeid and
f7.storeid=f8.storeid and f8.storeid=f9.storeid and f9.storeid=f10.storeid and
f10.storeid=f11.storeid and f11.storeid=f12.storeid;
```

```sql
--Method : CASE; Transposing Column: weekday; Aggregation Column: Number of Item; Use of Fv : NO
------------------------------------------------------------------------------------------------
CREATE view case_day_num as
select storeid,
SUM(CASE WHEN weekday='Monday' THEN numitems ELSE null END) as Monday,
SUM(CASE WHEN weekday='Tuesday' THEN numitems ELSE null END) as Tuesday,
SUM(CASE WHEN weekday='Wednesday' THEN numitems ELSE null END) as Wednesday,
SUM(CASE WHEN weekday='Thursday' THEN numitems ELSE null END) as Thursday,
SUM(CASE WHEN weekday='Friday' THEN numitems ELSE null END) as Friday,
SUM(CASE WHEN weekday='Saturday' THEN numitems ELSE null END) as Saturday,
SUM(CASE WHEN weekday='Sunday' THEN numitems ELSE null END) as Sunday
from querytable
group by storeid;

--Method: CASE; Transposing Column:department; Aggregation Column:Number of Item; Use of Fv:NO
-----------------------------------------------------------------------------------------------
CREATE view case_department_num as
select storeid,
SUM(CASE WHEN department='Electronics' THEN numitems ELSE null END) as Electronics,
SUM(CASE WHEN department='Books' THEN numitems ELSE null END) as Books,
SUM(CASE WHEN department='Clothing' THEN numitems ELSE null END) as Clothing,
SUM(CASE WHEN department='Home_Appliances' THEN numitems ELSE null END) as Home_Appliances,
SUM(CASE WHEN department='Cosmetics' THEN numitems ELSE null END) as Cosmetics,
SUM(CASE WHEN department='Food' THEN numitems ELSE null END) as Food,
SUM(CASE WHEN department='Gardening' THEN numitems ELSE null END) as Gardening,
SUM(CASE WHEN department='Sports' THEN numitems ELSE null END) as Sports,
SUM(CASE WHEN department='Paint' THEN numitems ELSE null END) as Paint,
SUM(CASE WHEN department='Movie_Games' THEN numitems ELSE null END) as Movie_Games,
SUM(CASE WHEN department='Hardware' THEN numitems ELSE null END) as Hardware,
SUM(CASE WHEN department='Jewelry' THEN numitems ELSE null END) as Jewelry,
SUM(CASE WHEN department='Baby_Toys' THEN numitems ELSE null END) as Baby_Toys,
SUM(CASE WHEN department='Health_Gourmet' THEN numitems ELSE null END) as Health_Gourmet
from querytable
group by storeid;

--Method: CASE;Transposing Column: weekday; Aggregation Column: Number of Item; Use of Fv: YES
-----------------------------------------------------------------------------------------------
CREATE view case_day_num_fv as
select storeid,
SUM(CASE WHEN weekday='Monday' THEN numitems ELSE null END) as Monday,
SUM(CASE WHEN weekday='Tuesday' THEN numitems ELSE null END) as Tuesday,
SUM(CASE WHEN weekday='Wednesday' THEN numitems ELSE null END) as Wednesday,
SUM(CASE WHEN weekday='Thursday' THEN numitems ELSE null END) as Thursday,
SUM(CASE WHEN weekday='Friday' THEN numitems ELSE null END) as Friday,
SUM(CASE WHEN weekday='Saturday' THEN numitems ELSE null END) as Saturday,
SUM(CASE WHEN weekday='Sunday' THEN numitems ELSE null END) as Sunday
from
(select storeid,weekday,SUM(numitems) as numitems
 from querytable
 group by storeid, weekday) as newtable
group by storeid;

--Method: CASE; Transposing Column: department; Aggregation Column: Number of Item; Use of Fv: YES
---------------------------------------------------------------------------------------------------
CREATE view case_department_num_fv as
select storeid,
SUM(CASE WHEN department='Electronics' THEN numitems ELSE null END) as Electronics,
SUM(CASE WHEN department='Books' THEN numitems ELSE null END) as Books,
SUM(CASE WHEN department='Clothing' THEN numitems ELSE null END) as Clothing,
SUM(CASE WHEN department='Home_Appliances' THEN numitems ELSE null END) as Home_Appliances,
SUM(CASE WHEN department='Cosmetics' THEN numitems ELSE null END) as Cosmetics,
SUM(CASE WHEN department='Food' THEN numitems ELSE null END) as Food,
SUM(CASE WHEN department='Gardening' THEN numitems ELSE null END) as Gardening,
SUM(CASE WHEN department='Sports' THEN numitems ELSE null END) as Sports,
SUM(CASE WHEN department='Paint' THEN numitems ELSE null END) as Paint,
SUM(CASE WHEN department='Movie_Games' THEN numitems ELSE null END) as Movie_Games,
SUM(CASE WHEN department='Hardware' THEN numitems ELSE null END) as Hardware,
SUM(CASE WHEN department='Jewelry' THEN numitems ELSE null END) as Jewelry,
SUM(CASE WHEN department='Baby_Toys' THEN numitems ELSE null END) as Baby_Toys,
SUM(CASE WHEN department='Health_Gourmet' THEN numitems ELSE null END) as Health_Gourmet
from
(select storeid,department,SUM(numitems) as numitems
 from querytable
 group by storeid, department) as newtable
group by storeid;
```

```sql
--Method : PIVOT ; Transposing Column: Month ; Aggregation Column: Number of Item ; Use of Fv : NO
-----------------------------------------------------------------------------------------------------
create view v_month_num as
select storeid, Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec
from
(
    select storeid, month, numitems
    from querytable
    ) as querytablenew
pivot
(
    Sum(numitems)
    for month in (Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec)
    ) as Pivottable;


--Method: PIVOT; Transposing Column:department; Aggregation Column:Number of Item; Use of Fv:NO
--------------------------------------------------------------------------------------------------
create view v_department_num as
select storeid, Electronics, Books, Clothing, Home_Appliances, Cosmetics, Food, Gardening, Sports, Paint, Movie_Games, Hardware, Jewelry, Baby_Toys, Health_Gourmet
from
(
    select storeid, department, numitems
    from querytable
    ) as querytablenew
pivot
(
    Sum(numitems)
    for department in (Electronics, Books, Clothing, Home_Appliances, Cosmetics, Food, Gardening, Sports, Paint, Movie_Games, Hardware, Jewelry, Baby_Toys, Health_Gourmet)
    ) as Pivottable;


--Method: PIVOT; Transposing Column: Month; Aggregation Column: Number of Item; Use of Fv: YES
-------------------------------------------------------------------------------------------------
create view v_month_num_fv as
select storeid, Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec
from
(
    select storeid, month, Sum(numitems) as Number_of_Items
    from querytable
    group by storeid, month
    ) as querytablenew
pivot
(
    Sum(Number_of_Items)
    for month in (Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec)
    ) as Pivottable;


--Method: PIVOT; Transposing Column: department; Aggregation Column: Number of Item; Use of Fv: YES
------------------------------------------------------------------------------------------------------
create view v_department_num_fv as
select storeid, Electronics, Books, Clothing, Home_Appliances, Cosmetics, Food, Gardening, Sports, Paint, Movie_Games, Hardware, Jewelry, Baby_Toys, Health_Gourmet
from
(
    select storeid, department, Sum(numitems) as Number_of_Items
    from querytable
    group by storeid, department
    ) as querytablenew
pivot
(
    Sum(Number_of_Items)
    for department in (Electronics, Books, Clothing, Home_Appliances, Cosmetics, Food, Gardening, Sports, Paint, Movie_Games, Hardware, Jewelry, Baby_Toys, Health_Gourmet)
    ) as Pivottable;


```

```
<%--
    /PROJECT/admin/horizontal-view.aspx
--%>

<%@ Page Title="Horizontal View" Language="C#" MasterPageFile="~/admin/admin1.master" AutoEventWireup="true" CodeFile="horizontal-view.aspx.cs" Inherits="admin_horizontal_view" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="Server">

    <p style="text-align: center; color: #3E02DB; font-size: 18px;">Horizontal View</p>
    <div style="margin-left: 20px;">
        <table>
            <tr>
                <td>
                    <asp:Label ID="Label1" runat="server" Text="Transposing Column: "></asp:Label>
                </td>
                <td>
                    <asp:DropDownList ID="DropDownList1" runat="server" ValidationGroup="view">
                        <asp:ListItem>--Select--</asp:ListItem>
                        <asp:ListItem Value="day">Day</asp:ListItem>
                        <asp:ListItem Value="month">Month</asp:ListItem>
                        <asp:ListItem Value="department">Department</asp:ListItem>
                    </asp:DropDownList></td>
                <td style="width: 40px;">
                    <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server" ControlToValidate="DropDownList1" ErrorMessage="*" ForeColor="Red" InitialValue="--Select--" ValidationGroup="view">*</asp:RequiredFieldValidator>
                </td>
                <td>
                    <asp:Label ID="Label2" runat="server" Text="Aggregation Column: "></asp:Label> </td>
                <td>
                    <asp:DropDownList ID="DropDownList2" runat="server" ValidationGroup="view">
                        <asp:ListItem>--Select--</asp:ListItem>
                        <asp:ListItem Value="numitems">Number of Items</asp:ListItem>
                        <asp:ListItem Value="saleamt">Sale Amount</asp:ListItem>
                    </asp:DropDownList></td>
                <td style="width: 30px">
                    <asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server" ErrorMessage="*" ControlToValidate="DropDownList2" ForeColor="Red" InitialValue="--Select--" ValidationGroup="view">*</asp:RequiredFieldValidator></td>
                <td>
                    <asp:Label ID="Label4" runat="server" Text="Compute Fv  "></asp:Label>
                    <asp:CheckBox ID="checkFv" runat="server" />
                </td>
            </tr>

        </table>

    </div>

    <div style="margin-top: 10px; margin-left: 160px;">
        <table>
            <tr>
                <td style="width: 120px;">
                    <asp:Button ID="btnSpj" runat="server" Text="SPJ" Width="80px" OnClick="btnSpj_Click" ValidationGroup="view" />
                </td>
                <td style="width: 120px;">
                    <asp:Button ID="btnCase" runat="server" Text="CASE" Width="80px" OnClick="btnCase_Click" ValidationGroup="view" />
                </td>
                <td style="width: 120px;">
                    <asp:Button ID="btnPivot" runat="server" Text="PIVOT" Width="80px" OnClick="btnPivot_Click" ValidationGroup="view" />
                </td>
                <td style="width: 250px;">
                    <asp:Label ID="Label5" runat="server" Text="Execution Time:"></asp:Label>
                    <asp:Label ID="Label3" runat="server" Text="Label"></asp:Label>
                </td>
            </tr>
        </table>
    </div>
    <div style="margin-top: 15px; float: right; width: 810px; margin-bottom: 20px; overflow: auto;">
```

```
        <asp:GridView ID="gridhorizontal" runat="server" CellPadding="4" ForeColor="#333333" GridLines="Vertical" Font-Size="15px" HorizontalAlign="Center" Width="800px">
            <AlternatingRowStyle BackColor="White" ForeColor="#284775" />
            <EditRowStyle BackColor="#999999" />
            <FooterStyle BackColor="#5D7B9D" Font-Bold="True" ForeColor="White" />
            <HeaderStyle BackColor="#5D7B9D" Font-Bold="True" ForeColor="White" HorizontalAlign="Center" />
            <PagerStyle BackColor="#284775" ForeColor="White" HorizontalAlign="Center" />
            <RowStyle BackColor="#F7F6F3" ForeColor="#333333" HorizontalAlign="Center" />
            <SelectedRowStyle BackColor="#E2DED6" Font-Bold="True" ForeColor="#333333" />
            <SortedAscendingCellStyle BackColor="#E9E7E2" />
            <SortedAscendingHeaderStyle BackColor="#506C8C" />
            <SortedDescendingCellStyle BackColor="#FFFDF8" />
            <SortedDescendingHeaderStyle BackColor="#6F8DAE" />
        </asp:GridView>
    </div>

    <div style="margin-left: 20px; margin-bottom: 20px;">
        <asp:Button ID="btnexcel" runat="server" Text="Export To Excel" OnClick="btnexcel_Click" />

    </div>

</asp:Content>
```

```csharp
/*
 * PROJECT/admin/horizontal-view.aspx.cs
 */
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.IO;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;

public partial class admin_horizontal_view : System.Web.UI.Page
{
    public static string constring = ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString;
    public static SqlConnection con = new SqlConnection(constring);
    protected void Page_Load(object sender, EventArgs e)
    {
        if (Session["AdminUser"] == "NotAvailable")
        {
            Response.Redirect("adminlogin.aspx");
        }
        labeltime.Text = "0 ms"; btnexcel.Visible = false;
    }
    protected void btnSpj_Click(object sender, EventArgs e)
    {
        btnexcel.Visible = true;
        var watch = System.Diagnostics.Stopwatch.StartNew();
        string transcolumnval = DropDownList1.SelectedValue.ToString();
        string aggcolumnval = DropDownList2.SelectedValue.ToString();
        DataSet ds = new DataSet();
        if (con.State == ConnectionState.Closed)
        {
            con.Open();
        }
        SqlCommand cmd = new SqlCommand("select * from v_day_num_fv;", con);
        if (checkFv.Checked)
        {
            if (transcolumnval == "day" && aggcolumnval == "numitems")
            {
                cmd = new SqlCommand("select v_day_num_fv.storeid, Monday,Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday, Total_Items from v_day_num_fv,v_total_number where v_day_num_fv.storeid=v_total_number.storeid;", con);
            }
            else if (transcolumnval == "month" && aggcolumnval == "numitems")
            {
                cmd = new SqlCommand("select spj_month_num_fv.storeid, Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec, Total_Items from spj_month_num_fv,v_total_number where spj_month_num_fv.storeid=v_total_number.storeid order by spj_month_num_fv.storeid;", con);
            }
            else if (transcolumnval == "department" && aggcolumnval == "numitems")
            {
                cmd = new SqlCommand("select spj_department_num_fv.storeid, Electronics, Books, Clothing, Home_Appliances, Cosmetics, Food, Gardening, Sports, Paint, Movie_Games, Hardware, Jewelry, Baby_Toys, Health_Gourmet, Total_Items from spj_department_num_fv,v_total_number where spj_department_num_fv.storeid=v_total_number.storeid order by spj_department_num_fv.storeid;", con);
            }
        }
        else
        {
            if (transcolumnval == "day" && aggcolumnval == "numitems")
            {
                cmd = new SqlCommand("select spj_day_num.storeid, Monday,Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday, Total_Items from spj_day_num,v_total_number where spj_day_num.storeid=v_total_number.storeid order by spj_day_num.storeid;", con);
            }
            else if (transcolumnval == "month" && aggcolumnval == "numitems")
            {
                cmd = new SqlCommand("select spj_month_num.storeid, Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec, Total_Items from spj_month_num,v_total_number where spj_month_num.storeid=v_total_number.storeid order by spj_month_num.storeid;", con);
        }
        else if (transcolumnval == "department" && aggcolumnval == "numitems")
        {
            cmd = new SqlCommand("select spj_department_num.storeid, Electronics, Books, Clothing, Home_Appliances, Cosmetics, Food, Gardening, Sports, Paint, Movie_Games, Hardware, Jewelry, Baby_Toys, Health_Gourmet, Total_Items from spj_department_num,v_total_number where spj_department_num.storeid=v_total_number.storeid order by spj_department_num.storeid;", con);

        }
    }
    SqlDataAdapter da = new SqlDataAdapter(cmd);
    da.Fill(ds);
    if (con.State == ConnectionState.Open)
    {
        con.Close();
    }
    gridhorizontal.DataSource = ds;
    gridhorizontal.DataBind();
    watch.Stop();
    var elapsedMs = watch.ElapsedMilliseconds;
    labeltime.Text = elapsedMs.ToString() + " ms";

}
protected void btnCase_Click(object sender, EventArgs e)
{
    btnexcel.Visible = true;
    var watch = System.Diagnostics.Stopwatch.StartNew();
    string transcolumnval = DropDownList1.SelectedValue.ToString();
    string aggcolumnval = DropDownList2.SelectedValue.ToString();
    DataSet ds = new DataSet();
    if (con.State == ConnectionState.Closed)
    {
        con.Open();
    }
    SqlCommand cmd = new SqlCommand("select * from v_day_num_fv;", con);
    if (checkFv.Checked)
    {
        if (transcolumnval == "day" && aggcolumnval == "numitems")
        {
            cmd = new SqlCommand("select case_day_num_fv.storeid, Monday,Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday, Total_Items from case_day_num_fv,v_total_number where case_day_num_fv.storeid=v_total_number.storeid order by case_day_num_fv.storeid;", con);
        }
        else if (transcolumnval == "month" && aggcolumnval == "numitems")
        {
            cmd = new SqlCommand("select case_month_num_fv.storeid, Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec, Total_Items from case_month_num_fv,v_total_number where case_month_num_fv.storeid=v_total_number.storeid order by case_month_num_fv.storeid;", con);
        }
        else if (transcolumnval == "department" && aggcolumnval == "numitems")
        {
            cmd = new SqlCommand("select case_department_num_fv.storeid, Electronics, Books, Clothing, Home_Appliances, Cosmetics, Food, Gardening, Sports, Paint, Movie_Games, Hardware, Jewelry, Baby_Toys, Health_Gourmet, Total_Items from case_department_num_fv,v_total_number where case_department_num_fv.storeid=v_total_number.storeid order by case_department_num_fv.storeid;", con);
        }
    }
    else
    {
        if (transcolumnval == "day" && aggcolumnval == "numitems")
        {
            cmd = new SqlCommand("select case_day_num.storeid, Monday,Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday, Total_Items from case_day_num,v_total_number where case_day_num.storeid=v_total_number.storeid order by case_day_num.storeid", con);
        }
        else if (transcolumnval == "month" && aggcolumnval == "numitems")
        {
            cmd = new SqlCommand("select case_month_num.storeid, Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec, Total_Items from case_month_num,v_total_number where case_month_num.storeid=v_total_number.storeid order by case_month_num.storeid;", con);
        }
        else if (transcolumnval == "department" && aggcolumnval == "numitems")
        {
            cmd = new SqlCommand("select case_department_num.storeid, Electronics, Books, Clothing, Home_Appliances, Cosmetics, Food, Gardening, Sports, Paint, Movie_Games, Hardware,
Jewelry, Baby_Toys, Health_Gourmet, Total_Items from case_department_num,v_total_number where ↙
case_department_num.storeid=v_total_number.storeid order by case_department_num.storeid;", ↙
con);
        }
    }
    SqlDataAdapter da = new SqlDataAdapter(cmd);
    da.Fill(ds);
    if (con.State == ConnectionState.Open)
    {
        con.Close();
    }
    gridhorizontal.DataSource = ds;
    gridhorizontal.DataBind();
    watch.Stop();
    var elapsedMs = watch.ElapsedMilliseconds;
    labeltime = elapsedMs.ToString()+" ms";

}
protected void btnPivot_Click(object sender, EventArgs e)
{
    btnexcel.Visible = true;
    var watch = System.Diagnostics.Stopwatch.StartNew();
    string transcolumnval = DropDownList1.SelectedValue.ToString();
    string aggcolumnval = DropDownList2.SelectedValue.ToString();
    DataSet ds = new DataSet();
    if (con.State == ConnectionState.Closed)
    {
        con.Open();
    }
    SqlCommand cmd = new SqlCommand("select * from v_day_num_fv;", con);
    if (checkFv.Checked)
    {
        if (transcolumnval == "day" && aggcolumnval == "numitems")
        {
            cmd = new SqlCommand("select v_day_num_fv.storeid, Monday,Tuesday, Wednesday, ↙
Thursday, Friday, Saturday, Sunday, Total_Items from v_day_num_fv,v_total_number where ↙
v_day_num_fv.storeid=v_total_number.storeid;", con);
        }
        else if (transcolumnval == "month" && aggcolumnval == "numitems")
        {
            cmd = new SqlCommand("select v_month_num_fv.storeid, Jan, Feb, Mar, Apr, May, Jun, ↙
 Jul, Aug, Sep, Oct, Nov, Dec, Total_Items from v_month_num_fv,v_total_number where ↙
v_month_num_fv.storeid=v_total_number.storeid;", con);
        }
        else if (transcolumnval == "department" && aggcolumnval == "numitems")
        {
            cmd = new SqlCommand("select v_department_num_fv.storeid, Electronics, Books, ↙
Clothing, Home_Appliances, Cosmetics, Food, Gardening, Sports, Paint, Movie_Games, Hardware, ↙
Jewelry, Baby_Toys, Health_Gourmet, Total_Items from v_department_num_fv,v_total_number where ↙
v_department_num_fv.storeid=v_total_number.storeid;", con);
        }
    }
    else
    {
        if (transcolumnval == "day" && aggcolumnval == "numitems")
        {
            cmd = new SqlCommand("select v_day_num.storeid, Monday,Tuesday, Wednesday,Thursday ↙
, Friday, Saturday, Sunday, Total_Items from v_day_num,v_total_number where v_day_num.storeid= ↙
v_total_number.storeid;", con);
        }
        else if (transcolumnval == "month" && aggcolumnval == "numitems")
        {
            cmd = new SqlCommand("select v_month_num.storeid, Jan, Feb, Mar, Apr, May, Jun, ↙
Jul, Aug, Sep, Oct, Nov, Dec, Total_Items from v_month_num,v_total_number where v_month_num. ↙
storeid=v_total_number.storeid;", con);
        }
        else if (transcolumnval == "department" && aggcolumnval == "numitems")
        {
            cmd = new SqlCommand("select v_department_num.storeid, Electronics, Books, ↙
Clothing, Home_Appliances, Cosmetics, Food, Gardening, Sports, Paint, Movie_Games, Hardware, ↙
Jewelry, Baby_Toys, Health_Gourmet, Total_Items from v_department_num,v_total_number where ↙
v_department_num.storeid=v_total_number.storeid;", con);
        }
    }
    SqlDataAdapter da = new SqlDataAdapter(cmd);
```

```
        da.Fill(ds);
        if (con.State == ConnectionState.Open)
        {
            con.Close();
        }
        gridhorizontal.DataSource = ds;
        gridhorizontal.DataBind();
        watch.Stop();
        var elapsedMs = watch.ElapsedMilliseconds;
        labeltime = elapsedMs.ToString() + " ms";

    }
    protected void btnexcel_Click(object sender, EventArgs e)
    {
        Response.ClearContent();
        Response.AppendHeader("content-disposition", "attachment; filename=storedata.xls");
        Response.ContentType = "application/excel";

        StringWriter stringWriter = new StringWriter();
        HtmlTextWriter htmlTextWriter = new HtmlTextWriter(stringWriter);

        gridhorizontal.RenderControl(htmlTextWriter);
        Response.Write(stringWriter.ToString());
        Response.End();
    }
    public override void VerifyRenderingInServerForm(Control control)
    {

    }
}
```

```
<%--
    /PROJECT/admin/K-Mean.aspx
--%>

<%@ Page Title="K-Mean Clustering" Language="C#" MasterPageFile="~/admin/admin1.master" AutoEventWireup="true" CodeFile="K-Mean.aspx.cs" Inherits="admin_K_Mean" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
    <div style="margin-left:330px; margin-top:20px;">
        <asp:Label ID="Label1" runat="server" Text="Value of K "></asp:Label>
        <asp:TextBox ID="txtK" runat="server" Height="20px" Width="65px" BackColor="#99CCFF" CausesValidation="True"  ValidationGroup="kmean" TextMode="Number"></asp:TextBox>
        <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server" ControlToValidate="txtK" ErrorMessage="*" ForeColor="Red" ValidationGroup="kmean">*</asp:RequiredFieldValidator>
        <asp:RangeValidator ID="RangeValidator1" runat="server" ControlToValidate="txtK" ErrorMessage="&gt;0 and &lt;=200" Font-Size="14px" ForeColor="Red" MaximumValue="200" MinimumValue="1" Type="Integer" ValidationGroup="kmean"></asp:RangeValidator>
    </div>
    <div style="margin-top:10px; margin-left:360px">
        <asp:Button ID="btnMain" runat="server" Text="Perform K-Mean" OnClick="btnMain_Click" ValidationGroup="kmean" />
    </div>
    <div style="margin-left:380px; margin-top:20px">
        <asp:Label ID="Label2" runat="server" Text="Clusters"></asp:Label>
    </div>
    <div style="float: right; width: 810px; margin-bottom: 20px; overflow: auto;">
        <asp:GridView ID="gvclusters" runat="server" BackColor="White" BorderColor="#999999" BorderStyle="Solid" BorderWidth="1px" CellPadding="3" ForeColor="Black" GridLines="Vertical" Font-Size="15px" HorizontalAlign="Center" Width="800px">
            <AlternatingRowStyle BackColor="#CCCCCC" />
            <FooterStyle BackColor="#CCCCCC" />
            <HeaderStyle BackColor="Black" Font-Bold="True" ForeColor="White" HorizontalAlign="Center"/>
            <PagerStyle BackColor="#999999" ForeColor="Black" HorizontalAlign="Center" />
            <RowStyle BackColor="#F7F6F3" ForeColor="#333333" HorizontalAlign="Center" />
            <SelectedRowStyle BackColor="#000099" Font-Bold="True" ForeColor="White" />
            <SortedAscendingCellStyle BackColor="#F1F1F1" />
            <SortedAscendingHeaderStyle BackColor="#808080" />
            <SortedDescendingCellStyle BackColor="#CAC9C9" />
            <SortedDescendingHeaderStyle BackColor="#383838" />
        </asp:GridView>
    </div>

    <div style="margin-top: 15px; float: right; width: 810px; margin-bottom: 20px; overflow: auto;">
        <asp:GridView ID="gvstorecluster" runat="server" BackColor="White" BorderColor="#999999" BorderStyle="Solid" BorderWidth="1px" CellPadding="3" ForeColor="Black" GridLines="Vertical" Font-Size="15px" HorizontalAlign="Center" Width="800px" Caption="Cluster Number And Their Corresponding Stores">
            <AlternatingRowStyle BackColor="#CCCCCC" />
            <FooterStyle BackColor="#CCCCCC" />
            <HeaderStyle BackColor="Black" Font-Bold="True" ForeColor="White" HorizontalAlign="Center"/>
            <PagerStyle BackColor="#999999" ForeColor="Black" HorizontalAlign="Center" />
            <RowStyle BackColor="#F7F6F3" ForeColor="#333333" HorizontalAlign="Center" />
            <SelectedRowStyle BackColor="#000099" Font-Bold="True" ForeColor="White" />
            <SortedAscendingCellStyle BackColor="#F1F1F1" />
            <SortedAscendingHeaderStyle BackColor="#808080" />
            <SortedDescendingCellStyle BackColor="#CAC9C9" />
            <SortedDescendingHeaderStyle BackColor="#383838" />
        </asp:GridView>
    </div>

    <div>
        <p>.</p>
    </div>

</asp:Content>
```

```csharp
/*
 * PROJECT/admin/K-Mean.aspx.cs
 */
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.IO;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;

public partial class admin_K_Mean : System.Web.UI.Page
{
    public static string constring = ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString;
    public static SqlConnection con = new SqlConnection(constring);
    protected void Page_Load(object sender, EventArgs e)
    {
        if (Session["AdminUser"] == "NotAvailable")
        {
            Response.Redirect("adminlogin.aspx");
        }
        Label2.Visible = false;
    }
    protected void btnMain_Click(object sender, EventArgs e)
    {
        int k = Convert.ToInt32(txtK.Text.ToString());
        SqlCommand cmd = new SqlCommand("select * from v_department_num_fv;", con);
        DataTable dt = new DataTable();
        if (cmd.Connection.State != ConnectionState.Open)
        {
            cmd.Connection.Open();
        }
        SqlDataReader reader = cmd.ExecuteReader();
        dt.Load(reader);
        if (cmd.Connection.State != ConnectionState.Closed)
        {
            cmd.Connection.Close();
        }
        Label2.Visible = true;
        string[] store= new string[dt.Rows.Count];
        int[] clusternumber = new int[dt.Rows.Count];
        double[,] points = new double[dt.Rows.Count, 14];
        for (int i = 0; i < dt.Rows.Count; i++)
        {
            store[i] = dt.Rows[i]["storeid"].ToString();
            if (dt.Rows[i]["Electronics"].ToString() == "") points[i, 0] = 0.0;
            else points[i, 0] = Convert.ToDouble(dt.Rows[i]["Electronics"].ToString());

            if (dt.Rows[i]["Books"].ToString() == "") points[i, 1] = 0.0;
            else points[i, 1] = Convert.ToDouble(dt.Rows[i]["Books"].ToString());

            if (dt.Rows[i]["Clothing"].ToString() == "") points[i, 2] = 0.0;
            else points[i, 2] = Convert.ToDouble(dt.Rows[i]["Clothing"].ToString());

            if (dt.Rows[i]["Home_Appliances"].ToString() == "") points[i, 3] = 0.0;
            else points[i, 3] = Convert.ToDouble(dt.Rows[i]["Home_Appliances"].ToString());

            if (dt.Rows[i]["Cosmetics"].ToString() == "") points[i, 4] = 0.0;
            else points[i, 4] = Convert.ToDouble(dt.Rows[i]["Cosmetics"].ToString());

            if (dt.Rows[i]["Food"].ToString() == "") points[i, 5] = 0.0;
            else points[i, 5] = Convert.ToDouble(dt.Rows[i]["Food"].ToString());

            if (dt.Rows[i]["Gardening"].ToString() == "") points[i, 6] = 0.0;
            else points[i, 6] = Convert.ToDouble(dt.Rows[i]["Gardening"].ToString());

            if (dt.Rows[i]["Sports"].ToString() == "") points[i, 7] = 0.0;
            else points[i, 7] = Convert.ToDouble(dt.Rows[i]["Sports"].ToString());

            if (dt.Rows[i]["Paint"].ToString() == "") points[i, 8] = 0.0;
            else points[i, 8] = Convert.ToDouble(dt.Rows[i]["Paint"].ToString());
```

```
                if (dt.Rows[i]["Movie_Games"].ToString() == "") points[i, 9] = 0.0;
                else points[i, 9] = Convert.ToDouble(dt.Rows[i]["Movie_Games"].ToString());

                if (dt.Rows[i]["Hardware"].ToString() == "") points[i, 10] = 0.0;
                else points[i, 10] = Convert.ToDouble(dt.Rows[i]["Hardware"].ToString());

                if (dt.Rows[i]["Jewelry"].ToString() == "") points[i, 11] = 0.0;
                else points[i, 11] = Convert.ToDouble(dt.Rows[i]["Jewelry"].ToString());

                if (dt.Rows[i]["Baby_Toys"].ToString() == "") points[i, 12] = 0.0;
                else points[i, 12] = Convert.ToDouble(dt.Rows[i]["Baby_Toys"].ToString());

                if (dt.Rows[i]["Health_Gourmet"].ToString() == "") points[i, 13] = 0.0;
                else points[i, 13] = Convert.ToDouble(dt.Rows[i]["Health_Gourmet"].ToString());
            }

            double[,] clusters = new double[k,14];

            for (int i = 0; i < k; i++)
            {
                for (int j = 0; j < 14; j++)
                {
                    clusters[i, j] = points[i, j];
                }
            }

            for (int i = 0; i < dt.Rows.Count; i++)
            {
                if (i < k) clusternumber[i] = i;
                else
                {
                    clusternumber[i] = 0; double distance = 0;
                    for (int j = 0; j < 14; j++)
                    {
                        distance += Math.Abs(points[i,j]-clusters[0,j]);
                    }
                    for (int j = 1; j < k; j++)
                    {
                        double distance1 = 0;
                        for (int w = 0; w < 14; w++)
                        {
                            distance1 += Math.Abs(points[i, w] - clusters[j, w]);
                        }
                        if (distance1 <= distance)
                        {
                            clusternumber[i] = j; distance = distance1;
                        }
                    }
                }
            }

            while (1>0)
            {
                int flag = 0;
                for (int i = 0; i < k; i++)
                {
                    for (int j = 0; j < 14; j++)
                    {
                        clusters[i, j] = 0;
                    }
                }
                int[] count = new int[k]; for (int p = 0; p < k; p++) count[p] = 0;
                for (int i = 0; i < clusternumber.Length; i++)
                {
                    int currentcluster = clusternumber[i];
                    count[currentcluster] += 1;
                    for (int j = 0; j < 14; j++)
                    {
                        clusters[currentcluster, j] += points[i, j];
                    }

                }
                for (int i = 0; i < k; i++)
                {
```

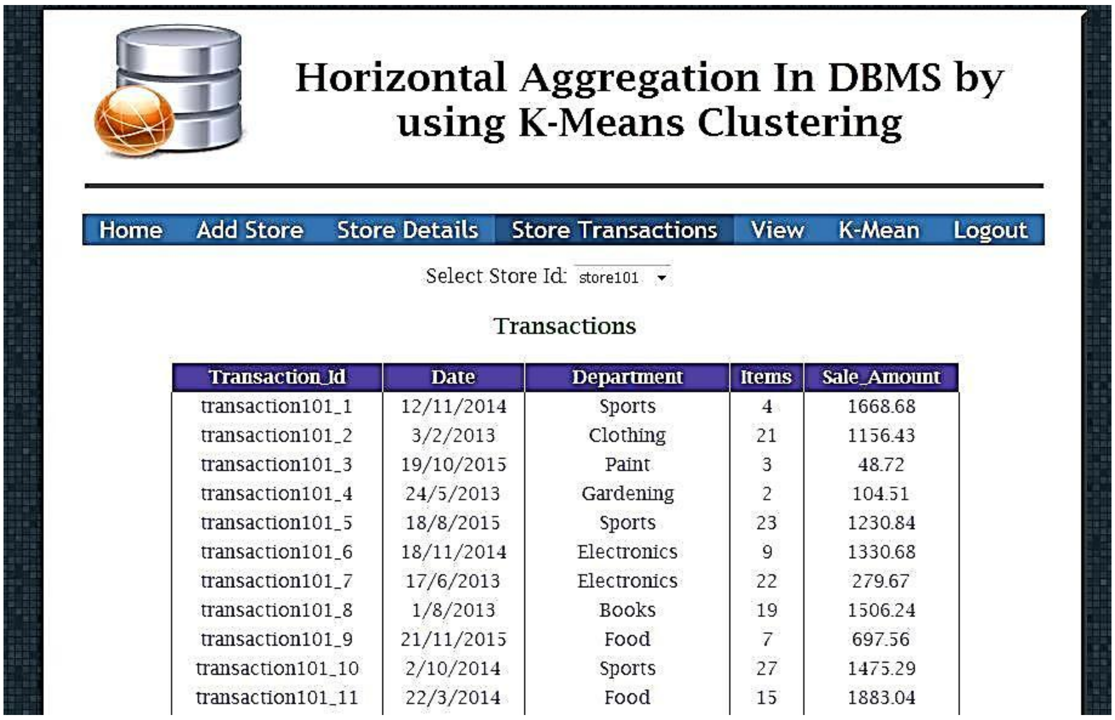
```
                int cnt = count[i];
                if (cnt > 0)
                {
                    for (int j = 0; j < 14; j++)
                    {
                        clusters[i, j] /= cnt;
                    }
                }
            }
            for (int i = 0; i < dt.Rows.Count; i++)
            {
                int currentcluster = clusternumber[i];
                double distance = 0;
                for (int j = 0; j < 14; j++)
                {
                    distance += Math.Abs(points[i, j] - clusters[currentcluster, j]);
                }

                for (int j = 0; j < k; j++)
                {
                    double distance1 = 0;
                    for (int w = 0; w < 14; w++)
                    {
                        distance1 += Math.Abs(points[i, w] - clusters[j, w]);
                    }
                    if (distance1 < distance)
                    {
                        flag = 1;
                        clusternumber[i] = j;
                        distance = distance1;
                    }
                }
            }
            if (flag == 0) break;

        }

        DataTable dtcluster = new DataTable();
        dtcluster.Columns.Add("Cluster_Number"); dtcluster.Columns.Add("Electronics");
        dtcluster.Columns.Add("Books"); dtcluster.Columns.Add("Clothing");
        dtcluster.Columns.Add("Home_Appliances"); dtcluster.Columns.Add("Cosmetics");
        dtcluster.Columns.Add("Food"); dtcluster.Columns.Add("Gardening");
        dtcluster.Columns.Add("Sports"); dtcluster.Columns.Add("Paint");
        dtcluster.Columns.Add("Movie_Games"); dtcluster.Columns.Add("Hardware");
        dtcluster.Columns.Add("Jewelry"); dtcluster.Columns.Add("Baby_Toys"); dtcluster.Columns.Add("Health_Gourmet");
        for (int i = 0; i < k; i++)
        {
            dtcluster.Rows.Add();
            dtcluster.Rows[i]["Cluster_Number"] = (i+1).ToString();
            dtcluster.Rows[i]["Electronics"] = Convert.ToInt32(clusters[i, 0]).ToString();
            dtcluster.Rows[i]["Books"] = Convert.ToInt32(clusters[i, 1]).ToString();
            dtcluster.Rows[i]["Clothing"] = Convert.ToInt32(clusters[i, 2]).ToString();
            dtcluster.Rows[i]["Home_Appliances"] = Convert.ToInt32(clusters[i, 3]).ToString();
            dtcluster.Rows[i]["Cosmetics"] = Convert.ToInt32(clusters[i, 4]).ToString();
            dtcluster.Rows[i]["Food"] = Convert.ToInt32(clusters[i, 5]).ToString();
            dtcluster.Rows[i]["Gardening"] = Convert.ToInt32(clusters[i, 6]).ToString();
            dtcluster.Rows[i]["Sports"] = Convert.ToInt32(clusters[i, 7]).ToString();
            dtcluster.Rows[i]["Paint"] = Convert.ToInt32(clusters[i, 8]).ToString();
            dtcluster.Rows[i]["Movie_Games"] = Convert.ToInt32(clusters[i, 9]).ToString();
            dtcluster.Rows[i]["Hardware"] = Convert.ToInt32(clusters[i, 10]).ToString();
            dtcluster.Rows[i]["Jewelry"] = Convert.ToInt32(clusters[i, 11]).ToString();
            dtcluster.Rows[i]["Baby_Toys"] = Convert.ToInt32(clusters[i, 12]).ToString();
            dtcluster.Rows[i]["Health_Gourmet"] = Convert.ToInt32(clusters[i, 13]).ToString();
        }
        gvclusters.DataSource = dtcluster;
        gvclusters.DataBind();

        string[,] storecluster = new string[k, 3];
        for (int i = 0; i < k; i++)
        {
            storecluster[i, 0] = (i + 1).ToString();
            storecluster[i, 1] = ""; storecluster[i, 2] = "";
        }
        int[] countstore = new int[k]; for (int i = 0; i < k; i++) countstore[i] = 0;
        for (int i = 0; i < clusternumber.Length; i++)
        {
            int num = clusternumber[i]; countstore[num] += 1;
            storecluster[num, 1] += store[i] + ", ";
        }
        for (int i = 0; i < k; i++)
        {
            storecluster[i, 2] = countstore[i].ToString();
        }

        DataTable dtstore = new DataTable();
        dtstore.Columns.Add("Cluster_Number"); dtstore.Columns.Add("Store_Id");
        dtstore.Columns.Add("Count");
        for (int i = 0; i < k; i++)
        {
            dtstore.Rows.Add();
            dtstore.Rows[i]["Cluster_Number"] = storecluster[i,0];
            dtstore.Rows[i]["Store_Id"] = storecluster[i, 1];
            dtstore.Rows[i]["Count"] = storecluster[i, 2];
        }

        gvstorecluster.DataSource = dtstore;
        gvstorecluster.DataBind();

    }
}
```

```
<%--
      /PROJECT/Global.asax
--%>

<%@ Application Language="C#" %>

<script runat="server">

    void Application_Start(object sender, EventArgs e)
    {
        // Code that runs on application startup
    }

    void Application_End(object sender, EventArgs e)
    {
        //  Code that runs on application shutdown
    }

    void Application_Error(object sender, EventArgs e)
    {
        // Code that runs when an unhandled error occurs
    }

    void Session_Start(object sender, EventArgs e)
    {
        // Code that runs when a new session is started
        Session.Add("AdminUser", "NotAvailable");
        Session.Add("User", "NotAvailable");
    }

    void Session_End(object sender, EventArgs e)
    {
        // Code that runs when a session ends.
        // Note: The Session_End event is raised only when the sessionstate mode
        // is set to InProc in the Web.config file. If session mode is set to StateServer
        // or SQLServer, the event is not raised.

    }

</script>
```

```xml
<!--
       /PROJECT/Web.sitemap
-->

<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
  <siteMapNode url="" title=""  description="">
    <siteMapNode url="~/admin/home.aspx" title="Home"  description="" />
    <siteMapNode url="~/admin/add-store.aspx" title="Add Store"  description="" />
    <siteMapNode url="~/admin/store-detail.aspx" title="Store Details"  description="" />
    <siteMapNode url="~/admin/store-transaction.aspx" title="Store Transactions"  description="" />
    <siteMapNode url="~/admin/horizontal-view.aspx" title="View"  description="" />
    <siteMapNode url="~/admin/K-Mean.aspx" title="K-Mean"  description="" />
    <siteMapNode url="~/admin/logout.aspx" title="Logout"  description="" />

  </siteMapNode>
</siteMap>

```

Chapter 7

Screenshots



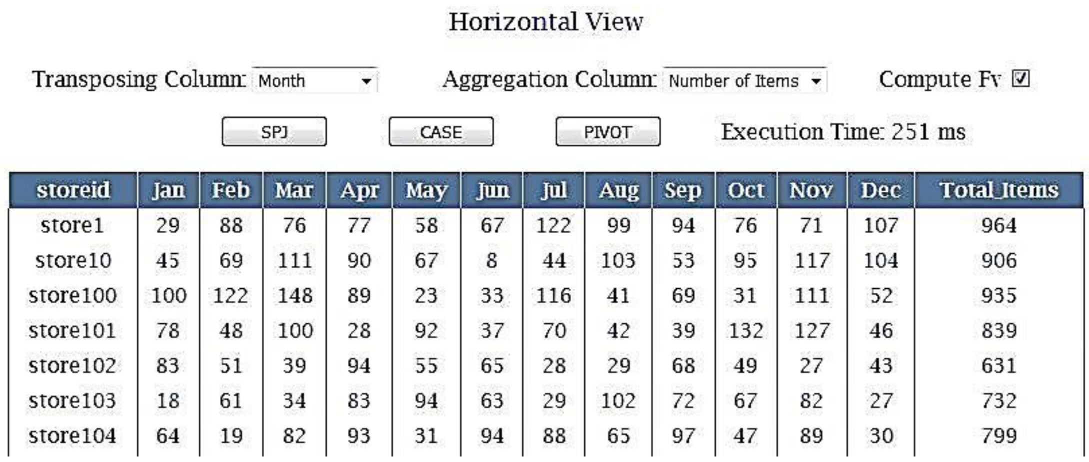
Horizontal Aggregation In DBMS by using K-Means Clustering

Select Store Id: store101

Transactions

Transaction_Id	Date	Department	Items	Sale_Amount
transaction101_1	12/11/2014	Sports	4	1668.68
transaction101_2	3/2/2013	Clothing	21	1156.43
transaction101_3	19/10/2015	Paint	3	48.72
transaction101_4	24/5/2013	Gardening	2	104.51
transaction101_5	18/8/2015	Sports	23	1230.84
transaction101_6	18/11/2014	Electronics	9	1330.68
transaction101_7	17/6/2013	Electronics	22	279.67
transaction101_8	1/8/2013	Books	19	1506.24
transaction101_9	21/11/2015	Food	7	697.56
transaction101_10	2/10/2014	Sports	27	1475.29
transaction101_11	22/3/2014	Food	15	1883.04

Fig. 4: All transactions of a particular store



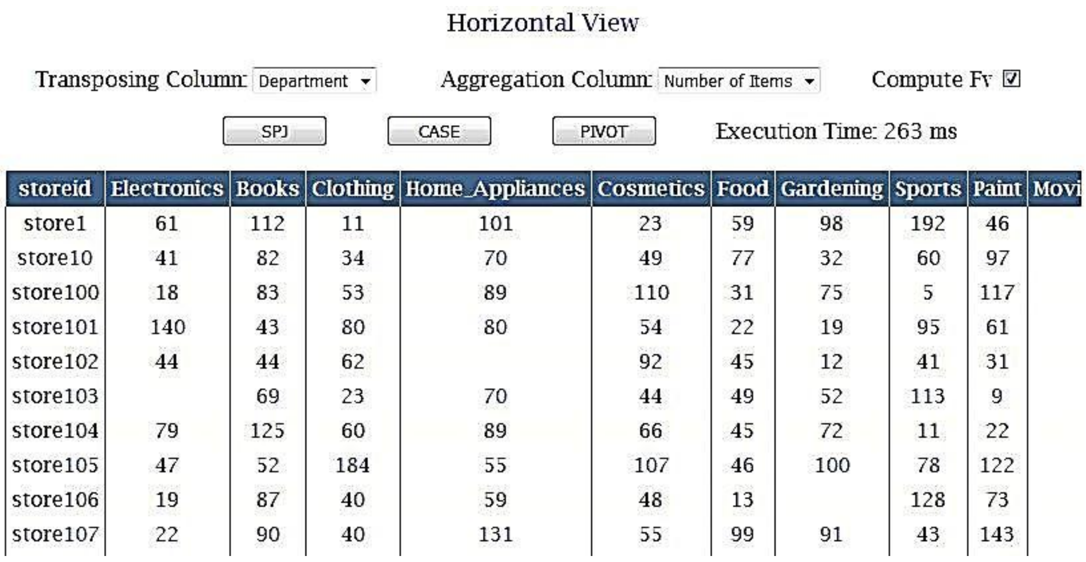
Horizontal View

Transposing Column: Month   Aggregation Column: Number of Items   Compute Fv ☑

Execution Time: 251 ms

storeid	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Total_Items
store1	29	88	76	77	58	67	122	99	94	76	71	107	964
store10	45	69	111	90	67	8	44	103	53	95	117	104	906
store100	100	122	148	89	23	33	116	41	69	31	111	52	935
store101	78	48	100	28	92	37	70	42	39	132	127	46	839
store102	83	51	39	94	55	65	28	29	68	49	27	43	631
store103	18	61	34	83	94	63	29	102	72	67	82	27	732
store104	64	19	82	93	31	94	88	65	97	47	89	30	799

Fig. 5: Horizontal Aggregation with transposing column ‘month’ with vertical aggregation as an intermediate table



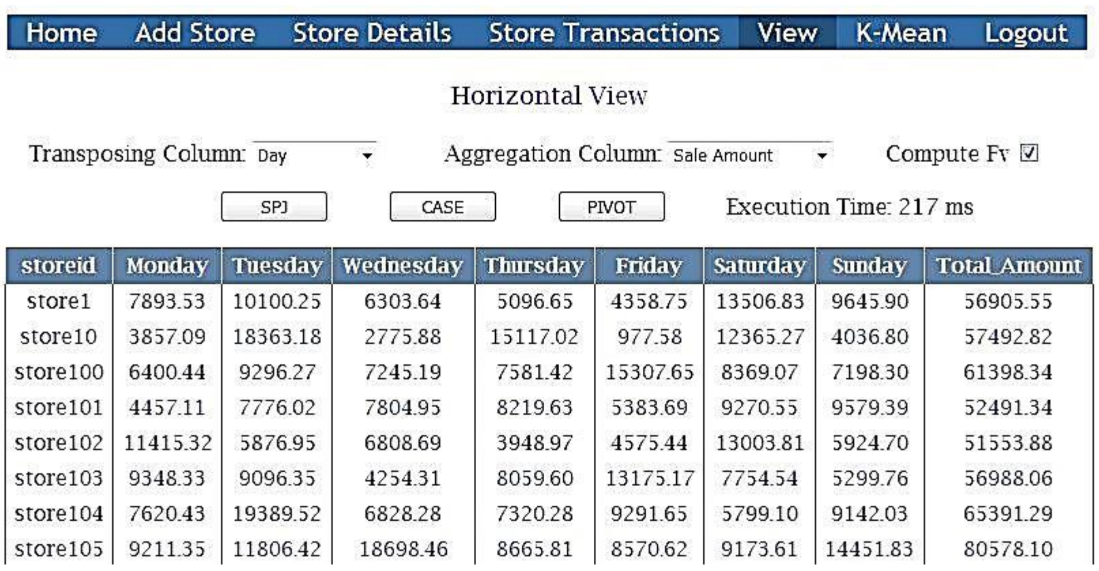
Horizontal View

Transposing Column: Department Aggregation Column: Number of Items Compute Fv ☑

Execution Time: 263 ms

storeid	Electronics	Books	Clothing	Home_Appliances	Cosmetics	Food	Gardening	Sports	Paint	Movi
store1	61	112	11	101	23	59	98	192	46	
store10	41	82	34	70	49	77	32	60	97	
store100	18	83	53	89	110	31	75	5	117	
store101	140	43	80	80	54	22	19	95	61	
store102	44	44	62		92	45	12	41	31	
store103		69	23	70	44	49	52	113	9	
store104	79	125	60	89	66	45	72	11	22	
store105	47	52	184	55	107	46	100	78	122	
store106	19	87	40	59	48	13		128	73	
store107	22	90	40	131	55	99	91	43	143	

Fig. 6: Horizontal View with transposing column ‘department’ and execution time



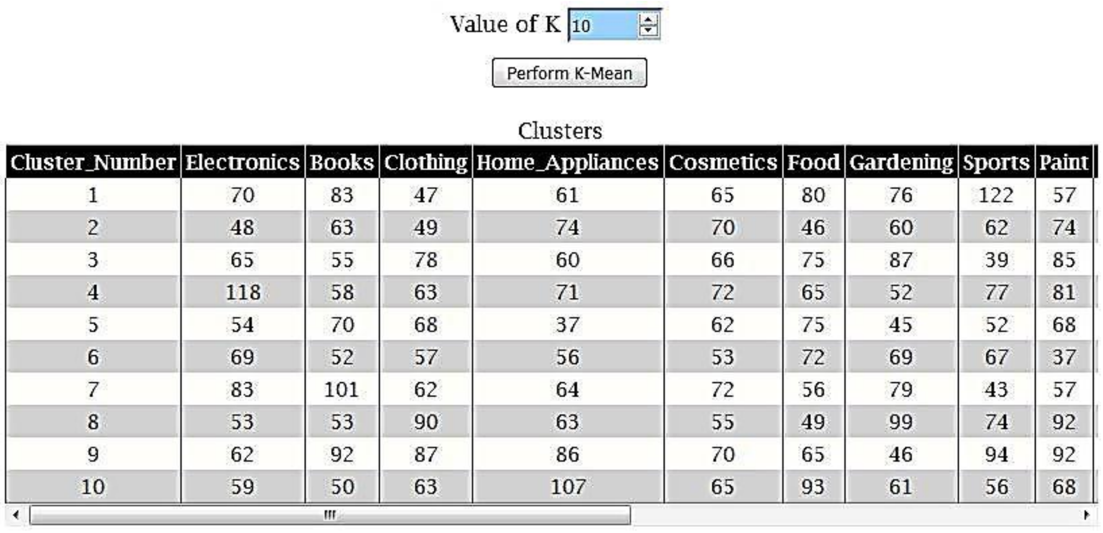
Horizontal View

Transposing Column: Day Aggregation Column: Sale Amount Compute Fv ☑

Execution Time: 217 ms

storeid	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday	Total_Amount
store1	7893.53	10100.25	6303.64	5096.65	4358.75	13506.83	9645.90	56905.55
store10	3857.09	18363.18	2775.88	15117.02	977.58	12365.27	4036.80	57492.82
store100	6400.44	9296.27	7245.19	7581.42	15307.65	8369.07	7198.30	61398.34
store101	4457.11	7776.02	7804.95	8219.63	5383.69	9270.55	9579.39	52491.34
store102	11415.32	5876.95	6808.69	3948.97	4575.44	13003.81	5924.70	51553.88
store103	9348.33	9096.35	4254.31	8059.60	13175.17	7754.54	5299.76	56988.06
store104	7620.43	19389.52	6828.28	7320.28	9291.65	5799.10	9142.03	65391.29
store105	9211.35	11806.42	18698.46	8665.81	8570.62	9173.61	14451.83	80578.10

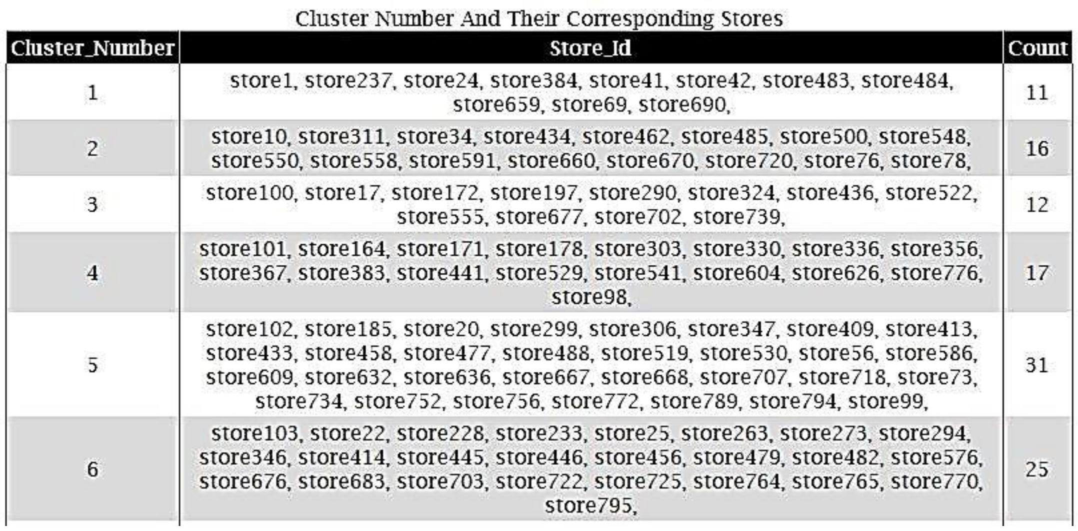
Fig. 7: Horizontal aggregation with Day as transposing column and sale amount as aggregation column



Clusters

Cluster_Number	Electronics	Books	Clothing	Home_Appliances	Cosmetics	Food	Gardening	Sports	Paint
1	70	83	47	61	65	80	76	122	57
2	48	63	49	74	70	46	60	62	74
3	65	55	78	60	66	75	87	39	85
4	118	58	63	71	72	65	52	77	81
5	54	70	68	37	62	75	45	52	68
6	69	52	57	56	53	72	69	67	37
7	83	101	62	64	72	56	79	43	57
8	53	53	90	63	55	49	99	74	92
9	62	92	87	86	70	65	46	94	92
10	59	50	63	107	65	93	61	56	68

Fig. 8: Classification of Horizontally aggregated data into K clusters by using K-Means algorithm.



Cluster Number And Their Corresponding Stores

Cluster_Number	Store_Id	Count
1	store1, store237, store24, store384, store41, store42, store483, store484, store659, store69, store690,	11
2	store10, store311, store34, store434, store462, store485, store500, store548, store550, store558, store591, store660, store670, store720, store76, store78,	16
3	store100, store17, store172, store197, store290, store324, store436, store522, store555, store677, store702, store739,	12
4	store101, store164, store171, store178, store303, store330, store336, store356, store367, store383, store441, store529, store541, store604, store626, store776, store98,	17
5	store102, store185, store20, store299, store306, store347, store409, store413, store433, store458, store477, store488, store519, store530, store56, store586, store609, store632, store636, store667, store668, store707, store718, store73, store734, store752, store756, store772, store789, store794, store99,	31
6	store103, store22, store228, store233, store25, store263, store273, store294, store346, store414, store445, store446, store456, store479, store482, store576, store676, store683, store703, store722, store725, store764, store765, store770, store795,	25

Fig. 9: Cluster Number and their corresponding stores with number of stores in a particular cluster.

Chapter 8
Conclusion

We introduced a new class of extended aggregate functions, called horizontal aggregations which help preparing data sets for data mining which is commonly required by data mining algorithms. Basically, a horizontal aggregation returns a set of numbers instead of a single number for each group, resembling a multidimensional vector. From a query optimization perspective, we proposed three query evaluation methods. The first one (SPJ) relies on standard relational operators. The second one (CASE) relies on the SQL CASE construct. The third (PIVOT) uses a builtin operator in a commercial DBMS that is not widely available. The SPJ method is important from a theoretical point of view because it is based on select, project, and join (SPJ) queries. The CASE method is our most important contribution. It is in general the most efficient evaluation method and it has wide applicability since it can be programmed combining GROUP-BY and CASE statements. All these three methods produce the same result. We have explained it is not possible to evaluate horizontal aggregations using standard SQL without either joins or "case" constructs using standard SQL operators.

Our experiments with large tables show our proposed horizontal aggregations evaluated with the CASE method have similar performance to the built-in PIVOT operator. Both CASE and PIVOT evaluation methods are significantly faster than the SPJ method. Precomputing a cube on selected dimensions produced acceleration on all methods.

Horizontal aggregations results in large volumes of data sets which are then partitioned into homogeneous clusters. This can be performed by K Means Clustering Algorithm. K-means clustering algorithm with query evaluation method and aggregation function is used for optimizing horizontal aggregation.

Chapter 9

Future Scope of Work

Efficiently evaluating horizontal aggregations using left outer joins presents opportunities for query optimization. Secondary indexes on common grouping columns, besides indexes on primary keys, can accelerate computation. Our proposed horizontal aggregations do not introduce conflicts with vertical aggregations, but we need to develop a more formal model of evaluation. In particular, we want to study the possibility of extending SQL OLAP aggregations with horizontal layout capabilities. Horizontal aggregations produce tables with fewer rows, but with more columns. Thus query optimization techniques used for standard (vertical) aggregations are inappropriate for horizontal aggregations.

We want to study optimization of horizontal aggregations processed in parallel in a sharednothing DBMS architecture. Cube properties can be generalized to multivalued aggregation results produced by a horizontal aggregation. Optimizing a workload of horizontal aggregation queries is another challenging problem.

References

[1] C. Cunningham, G. Graefe, and C.A. Galindo-Legaria. “PIVOT and UNPIVOT: Optimization and execution strategies in an RDBMS”, In Proc. VLDB Conference, pages 998–1009, 2004.

[2] J. Gray, A. Bosworth, A. Layman, and H. Pirahesh. “Data cube: A relational aggregation operator generalizing group-by, cross-tab and subtotal”, In ICDE Conference, pages 152–159, 1996.

[3] C. Ordonez. “Horizontal aggregations for building tabular data sets”, In Proc. ACM SIGMOD Data Mining and Knowledge Discovery Workshop, pages 35–42, 2004.

[4] C. Ordonez. “Vertical and horizontal percentage aggregations”, In Proc. ACM SIGMOD Conference, pages 866–871, 2004.

[5] C. Ordonez. “Integrating K-means clustering with a relational DBMS using SQL”, IEEE Transactions on Knowledge and Data Engineering (TKDE), 18(2):188–201, 2006.

[6] C.Ordonez.“Data set preprocessing and transformation in a database system”, Intelligent Data Analysis (IDA), 15(4), 2011.

[7] Sarawagi, Sunita, Shiby Thomas, and Rakesh Agrawal. Integrating association rule mining with relational database systems: Alternatives and implications. Vol. 27, no. 2. ACM, 1998.

[8] Jens-Peter Dittrich, Donald Kossmann and Alexander “Bridging the Gap between OLAP and SQL”, Proceedings of the 31st VLDB Conference, Trondheim, Norway, 2005.

[9] G. Graefe, U. Fayyad, and S. Chaudhuri. “On the Efficient Gathering of Sufficient Statistics for Classification from Large SQL Databases”, Proceedings of The Fourth International Conference on Knowledge Discovery and Data Mining, 1998, pages 204-208.