Data Mining and Analytics Part-2

Implement algorithms for the following using any programming languages of your choice out of C, C++, Java, Python

NDEX

SN		
1	About Data	
	Prepare a dataset by accepting age data from the monitor. The age data are integer values representing age of people belonging to the different groups of the society with age from 5 to 95 years. Take about 50 such data values; you can change the number of data values according to the computation of the measures' requirements. (Q. No. 2.2 on page no. 80, Han book)	
	a	Arrange data in ascending order
	b	Compute mean, median and mode of the data
	c	Compute percentile of a data item from the formed dataset
	d	Compute 70 and 80 percentiles
	e	Compute 70th and 80th percentiles
	f	Compute midrange of the data
	g	Compute 1st and 3rd qualtiles of the dataset
	h	Find 1st and 3rd quartiles of the dataset
	i	Give five number summary of the data
	j	Repeat experiment from (a) to (i) at least five times by taking different data values.
	k.	Write a general program for computing percentile of a given dataset. Take the number of data values in a dataset as a variable. The program should accept all data of the dataset from the monitor.

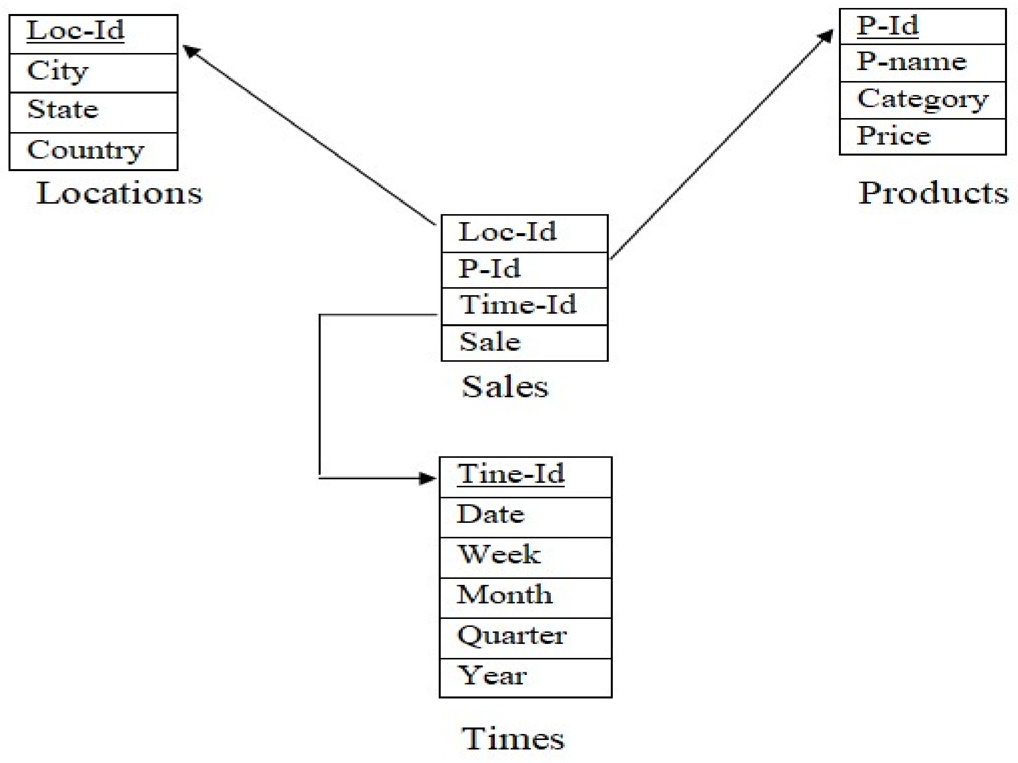
2	Dissimilarity Measures	
	a	Nominal Attributes Consider a data matrix of ten objects. Each object is made of one specific colour out of the five different colours. Let attribute colour has values red, yellow, green, blue, and black. You can consider any colour of the object of your own choice. Write a program to generate dissimilarity matrix from the data matrix. (Here, dissimilarity matrix will be of 10x10) Generalize the program for computing dissimilarity matrix for nominal attributes by accepting inputs from the monitor.
	b	Binary Attribute Consider the patient data set given in the Table 2.4 on page 71 of the Han's 3rd edition book. Ignore attribute gender as it is symmetric and consider remaining six asymmetric attributes fever, cough, test-1, test-2, test-3 and test 4. Let the table has five tuples; generate a dissimilarity matrix. Generalize the program for computing dissimilarity matrix for binary asymmetric attributes by taking inputs from the monitor.
	c	Numeric Attributes Write a program for computing the following distances between two objects described by numeric attributes (see question no. 2.6, on page no. 81 of Han's 3rd edition book): (1) Euclidean distance (2) Manhattan distance (3) Minkowski distance (4) Supermum distance The program should accept the number of attributes and their values for the objects from monitor.
	d	Ordinal attributes For patient data given in the Table 2.2 on page no. 69 of the Han's 3rd edition book, consider only test-2 ordinal data with ranks *poor, fair, good, very good*, and *excellent* for different patients/objects. Take ten patient data, normalize ranks of data on scale (0.0 - 1.0) and generate dissimilarity matrix. Generalize the program for computing dissimilarity matrix for ordinal attributes by taking inputs from the monitor. See section 2.4.5 page no. 74-75 in the Han's 3rd edition book.
	e	Cosine Similarity Write a program for computing cosine similarity between two document vectors. See Table 2.5 on page no. 77 and example 2.23on page no. 78 in Han's 3rd edition book. Generalize the program for computing cosine similarity by taking inputs from the monitor.

3	Data Preprocessing	
	a	Smoothing Consider age data as given in the question 3.3, on page no. 121, in Han's 3rd edition book. Apply the following data smoothing methods for smoothing data; use bin depth of three. (i) smoothing by bin boundaries (ii) smoothing by bin means (iii) smoothing by bin medians
	b	Write a program for performing X^2 (Chi-square) correlation tests of nominal data. See example 3.1, on page no. 95-96, in Han's 3rd edition book. You can consider other nominal attributes instead of the preferred reading material as given in the example. Generalize the program for computing value X^2, the program should accept inputs from monitor.
	c	Write a program for computing correlation coefficient of numeric data. You can take data values of variables x and y from any website on the internet. Generalize the program such that it accepts input data for the variables from the monitor.
	d	Write a program for computing covariance of numeric data. See example 3.2, on page no. 98, in Han's 3rd edition book. You can take data values of variables x and y from any website on the internet. Generalize the program such that it accepts input data for the variables from the monitor.
	e	Write a program for computing variance and standard deviation for a given numeric attribute x. You can take data values for x from any website on the internet. Generalize the program such that it accepts input data for the variables from the monitor.
	f	Write a program for transforming data by the following normalizations: (i) Min-max normalization (ii) Z-score normalization (iii) Decimal scaling normalization (iv) [0.0,-1.0] normalization (v) [-1,1] normalization See questions 3.5, 3.6, 3.7, 3.8, on page no. 121-122, in Han's 3rd edition book. Generalize the program for transforming data by all the above types of normalizations.
	g	Download publicly available data cleaning tool Potter Wheel. Study and use this tool for cleaning data. See the following article: Potter's Wheel: An Interactive Data Cleaning System by Vijayshankar Raman and Joseph M. Hellerstein, Proceedings of the 27th VLDB Conference, Roma, Italy, 2001.

4	Frequent itemsets and apriori algorithm (Association Rule Mining) See page no. 248 to 253 in Han’s 3rd edition book.	
	a	Objective: To compute maximal frequent itemset. Write a program for computing candidate 3-itemsets from frequent 2-itemsets using join $C_3 = L_2 \times L_2$. (Han’s book example) Generalize the program for generating candidate C_{i+1} itemsets from frequent L_i itemsets $C_{i+1} = L_i \times L_i$
	b	Objective: To develop prune operation using apriori property. Write a program for pruning/removing unnecessary 3-itemsets from the set of generated 3-itemsets C_3 to make C_3 to set of frequent 3-itemsets L_3. (Han book example) Generalize the program for pruning unnecessary i-itemsets from the set of generated i-itemsets C_i to make C_i to set of frequent i-itemsets L_i.
	c	Write a program for apriori algorithm using the above join and prune procedures.
5	FP Growth algorithm (Association Rule Mining) See page no. 257 to 262 in Han’s 3rd edition book.	
	a	Objective: To compute maximal frequent itemset. Write a program for computing item set of frequent items (1-itemsets) and their support counts from a given transactional dataset. Sort frequent itemsets and generate them in L order (descending order of support counts).
	b	Sort items in the transactions of the dataset in L-order (descending order of support counts).
	c	Construct FP tree using the Han’s book example. Display FP tree using appropriate notation/representation.
	d	Using FP tree, construct pattern bases and conditional FP trees.
	e	Generate frequent patterns.

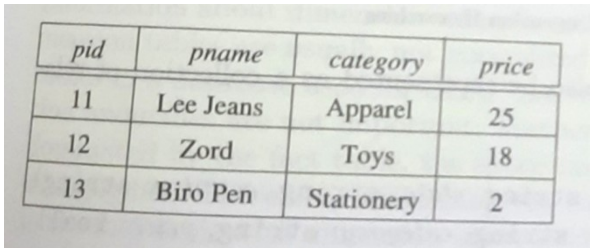
6	Data Warehousing	
	a	Create a data warehouse from different .csv files using PostgreSQL tool.

b Using MySQL/PostgreSQL RDBMS, create a data warehouse for the following star schema



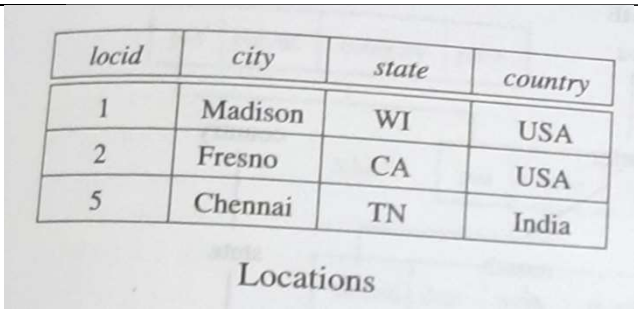
Products	(P-Id (integer), P-name (string), Category (string), Price (real))
Locations	(Loc-Id (integer), City (string), State (string), Country (string))
Times	(Time-Id (integer), Date (string), Week (integer), Month (integer), Quarter (integer), Year (integer))

The following tables exhibit the sample data for the tables Product, Locations and Times.



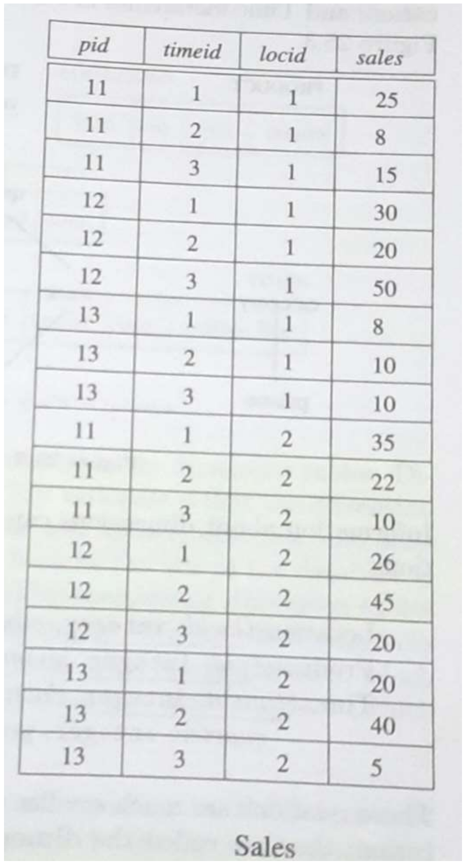
pid	*pname*	*category*	*price*
11	Lee Jeans	Apparel	25
12	Zord	Toys	18
13	Biro Pen	Stationery	2

Products



locid	city	state	country
1	Madison	WI	USA
2	Fresno	CA	USA
5	Chennai	TN	India

Locations



pid	timeid	locid	sales
11	1	1	25
11	2	1	8
11	3	1	15
12	1	1	30
12	2	1	20
12	3	1	50
13	1	1	8
13	2	1	10
13	3	1	10
11	1	2	35
11	2	2	22
11	3	2	10
12	1	2	26
12	2	2	45
12	3	2	20
13	1	2	20
13	2	2	40
13	3	2	5

Sales

Dimension hierarchies on each dimension are as under

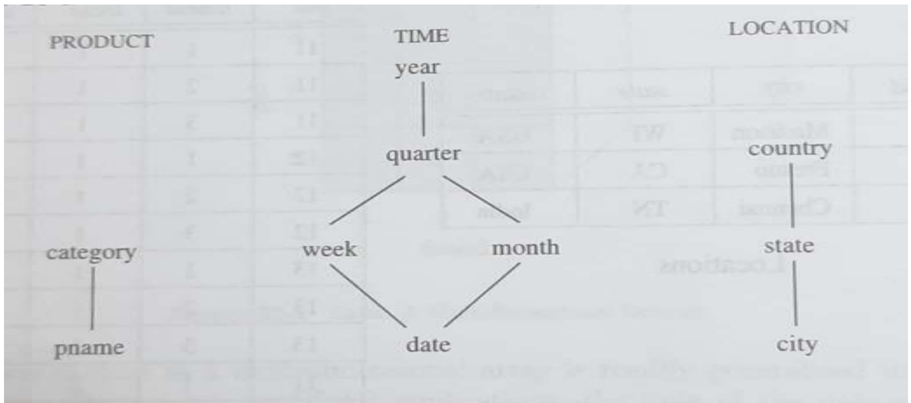


Figure 25.3 Dimension Hierarchies

Write SQL queries (both MySQL and PostgreSQL support SQL 92 standard) for the following:

1. Find the total sales.
2. Find total sales for each city.
3. Find total sales for each state.
4. Find total sales for each country.
5. Find sales of all cities of a specific state in a specific year.
6. Find year-wise total sales for each state.
7. Find year-wise total sales for each country.

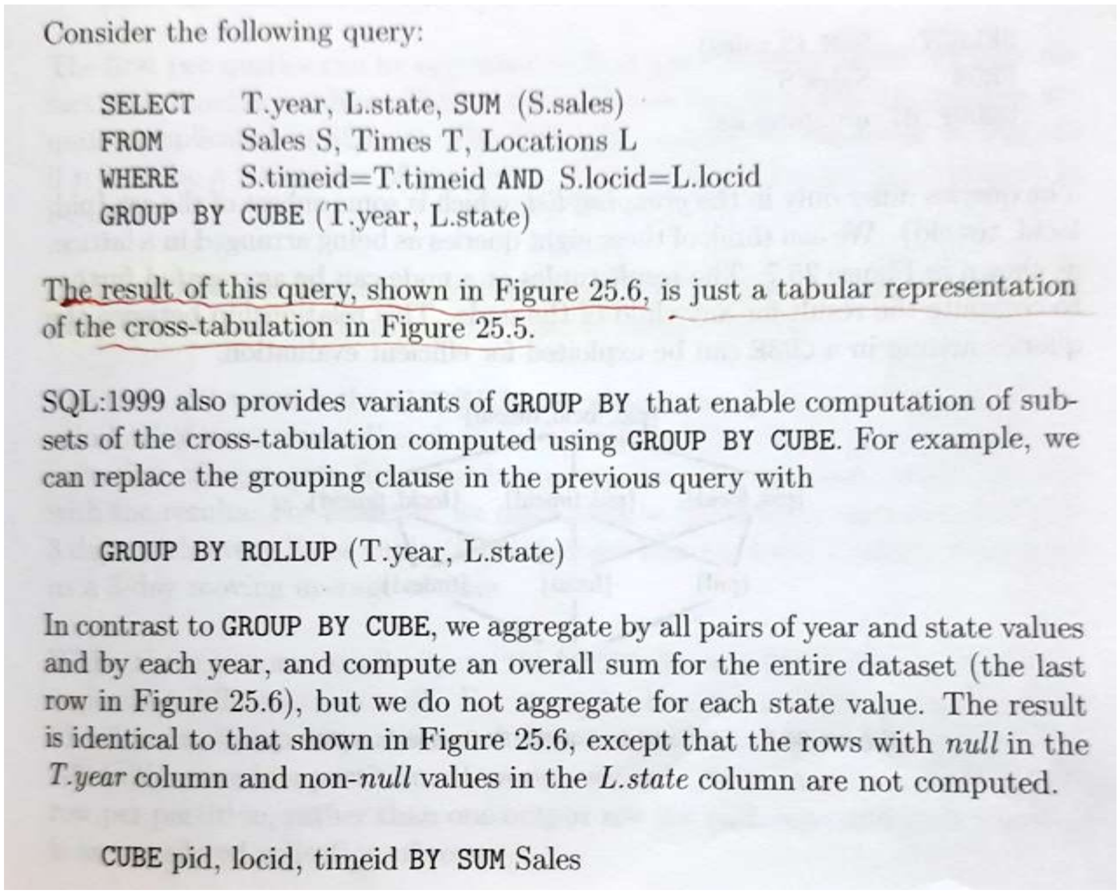
When we aggregate a given measure on a dimension, the aggregated measure will be associated with more abstracted dimension than that of the original measure.
For example,
For given total sale by city, aggregation of measure on location dimension will compute total sales by state, and further aggregation of measure on location dimension will compute total sales by country. In OLAP, it is equivalent to roll-up operation. Dril-down is a reverse operation of the roll-up.

An OL:AP operation is equivalent to several closely related SQL queries with aggregation and grouping.

SQL 1999 (supported by PostgreSQL) supports OLAP operations by extending clause GROUP BY as GROUP BY CUBE.

Students are advised to write queries (1 to 7) using OLAP operations supported by SQL 1999. For writing other queries students can take help of dimension hierarchies. Study of a tutorial on PostgreSQL for OLAP operation will be helpful.

For easy understanding, students can make observation of the following queries:



Consider the following query:

```
SELECT    T.year, L.state, SUM (S.sales)
FROM      Sales S, Times T, Locations L
WHERE     S.timeid=T.timeid AND S.locid=L.locid
GROUP BY CUBE (T.year, L.state)
```

The result of this query, shown in Figure 25.6, is just a tabular representation of the cross-tabulation in Figure 25.5.

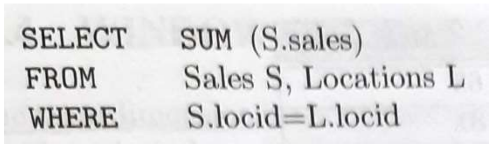
SQL:1999 also provides variants of GROUP BY that enable computation of subsets of the cross-tabulation computed using GROUP BY CUBE. For example, we can replace the grouping clause in the previous query with

```
GROUP BY ROLLUP (T.year, L.state)
```

In contrast to GROUP BY CUBE, we aggregate by all pairs of year and state values and by each year, and compute an overall sum for the entire dataset (the last row in Figure 25.6), but we do not aggregate for each state value. The result is identical to that shown in Figure 25.6, except that the rows with *null* in the *T.year* column and non-*null* values in the *L.state* column are not computed.

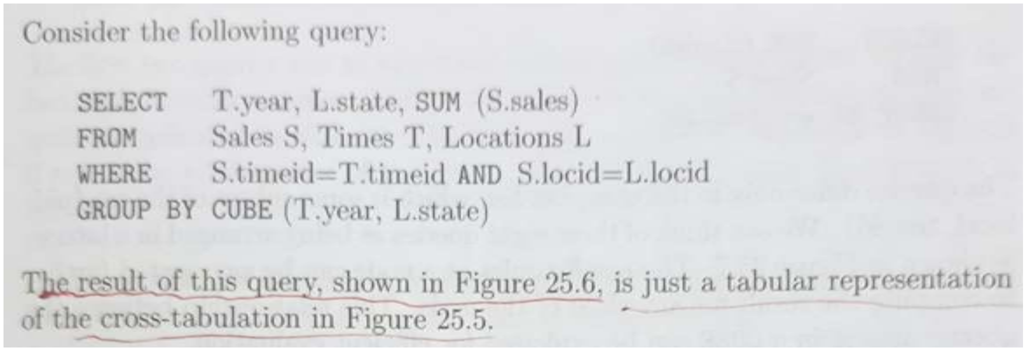
```
CUBE pid, locid, timeid BY SUM Sales
```

(Image-1)



```
SELECT    SUM (S.sales)
FROM      Sales S, Locations L
WHERE     S.locid=L.locid
```

(Image-2)

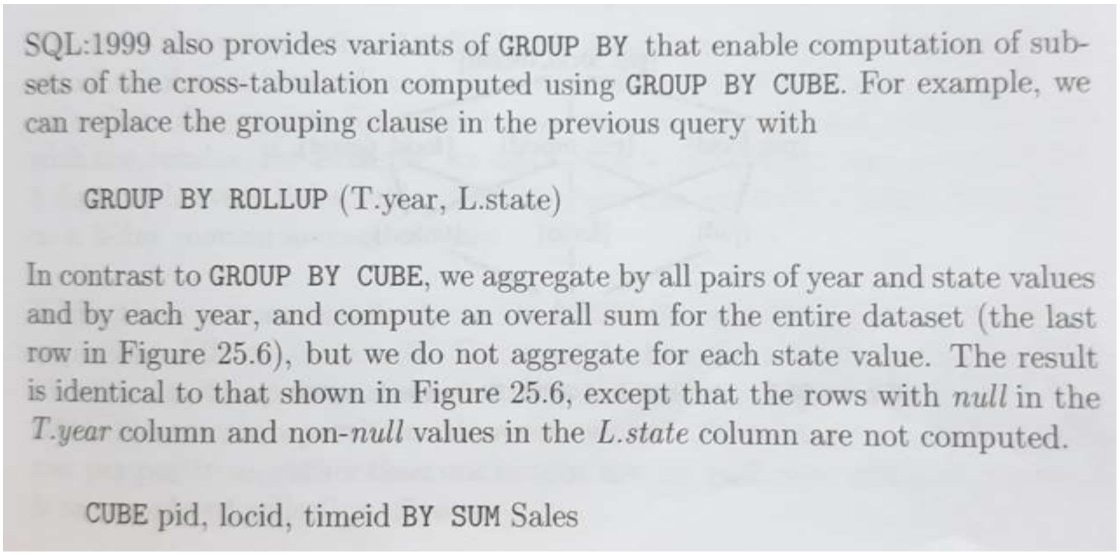


Consider the following query:

```
SELECT    T.year, L.state, SUM (S.sales)
FROM      Sales S, Times T, Locations L
WHERE     S.timeid=T.timeid AND S.locid=L.locid
GROUP BY CUBE (T.year, L.state)
```

The result of this query, shown in Figure 25.6, is just a tabular representation of the cross-tabulation in Figure 25.5.

(Image-3)



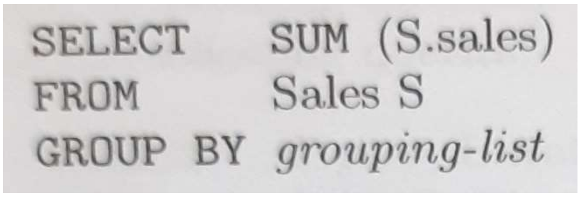
SQL:1999 also provides variants of `GROUP BY` that enable computation of subsets of the cross-tabulation computed using `GROUP BY CUBE`. For example, we can replace the grouping clause in the previous query with

```
GROUP BY ROLLUP (T.year, L.state)
```

In contrast to `GROUP BY CUBE`, we aggregate by all pairs of year and state values and by each year, and compute an overall sum for the entire dataset (the last row in Figure 25.6), but we do not aggregate for each state value. The result is identical to that shown in Figure 25.6, except that the rows with *null* in the *T.year* column and non-*null* values in the *L.state* column are not computed.

```
CUBE pid, locid, timeid BY SUM Sales
```

(Image-4)



```
SELECT     SUM (S.sales)
FROM       Sales S
GROUP BY   grouping-list
```

(Image-5)

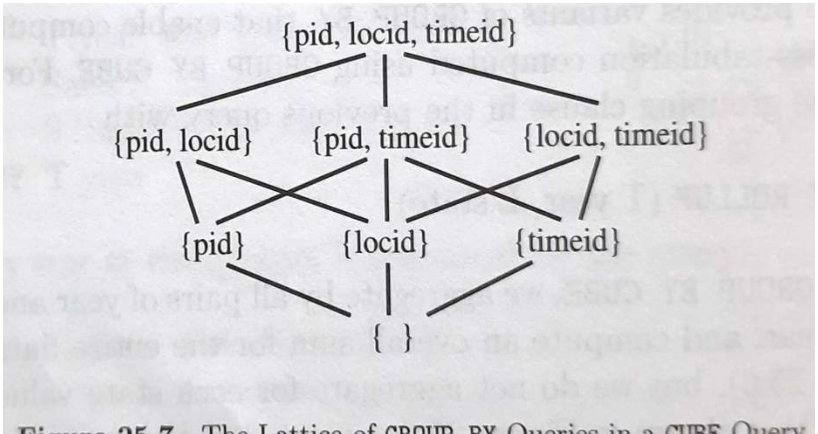
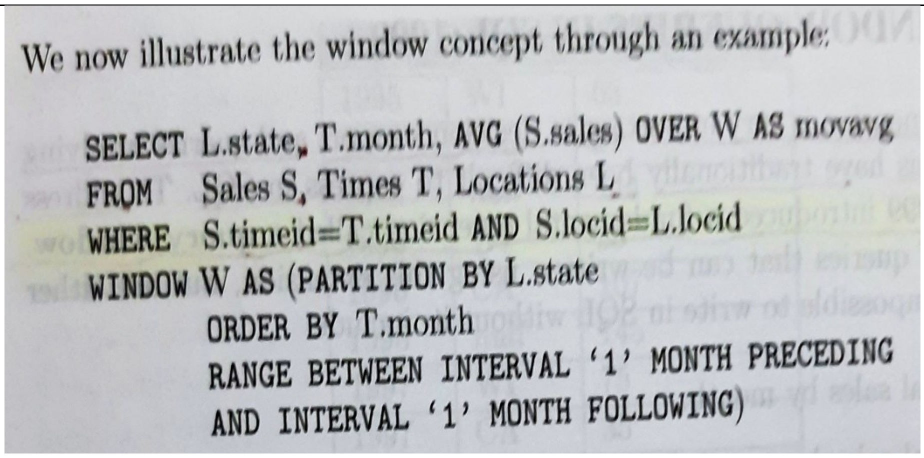


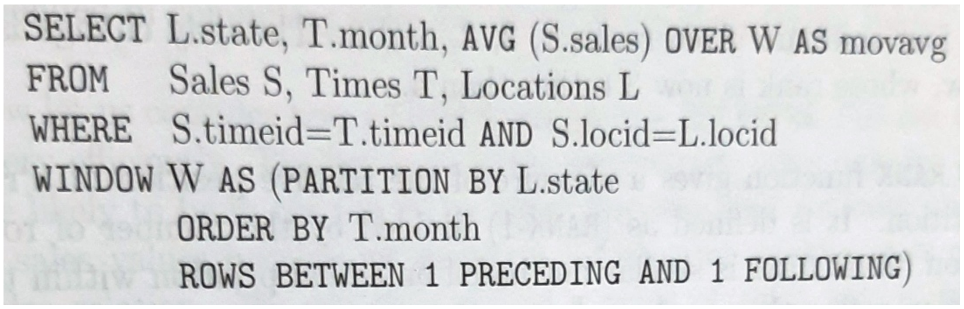
Figure 25.7 The Lattice of GROUP BY Queries in a CUBE Query

(Image-6)


We now illustrate the window concept through an example:

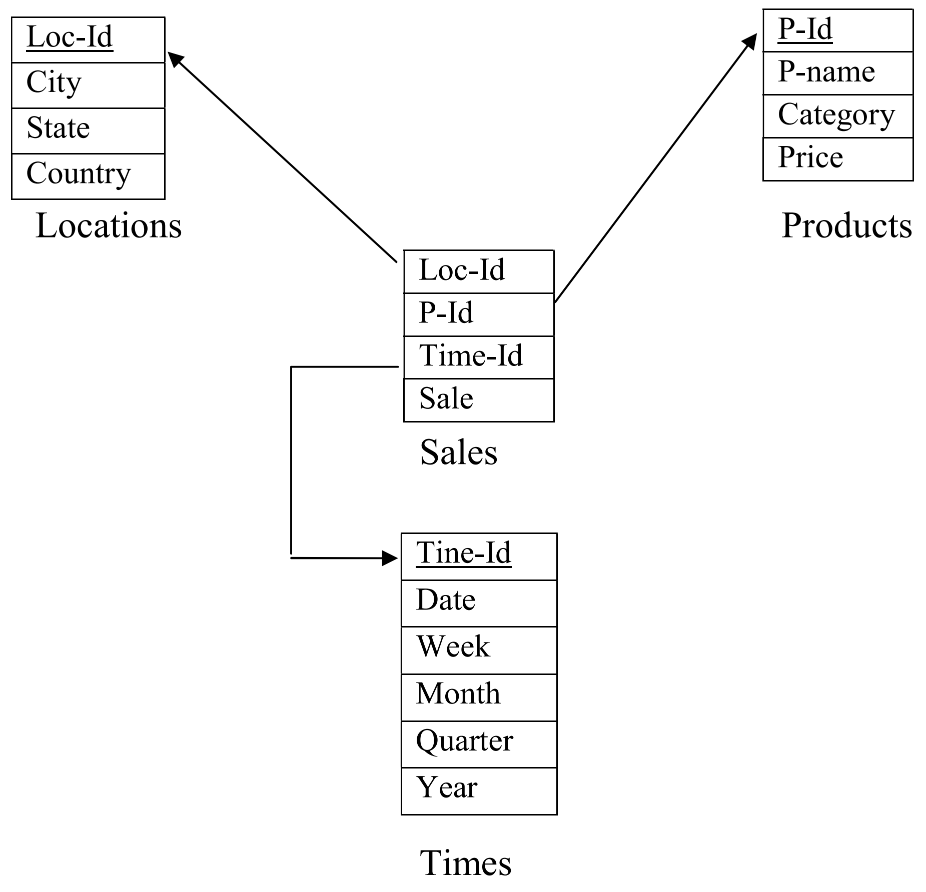
```
SELECT L.state, T.month, AVG (S.sales) OVER W AS movavg
FROM   Sales S, Times T, Locations L
WHERE  S.timeid=T.timeid AND S.locid=L.locid
WINDOW W AS (PARTITION BY L.state
        ORDER BY T.month
        RANGE BETWEEN INTERVAL '1' MONTH PRECEDING
        AND INTERVAL '1' MONTH FOLLOWING)
```

(Image-7)



```
SELECT L.state, T.month, AVG (S.sales) OVER W AS movavg
FROM   Sales S, Times T, Locations L
WHERE  S.timeid=T.timeid AND S.locid=L.locid
WINDOW W AS (PARTITION BY L.state
        ORDER BY T.month
        ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING)
```

(Image-8)


Loc-Id
City
State
Country
Locations
P-Id
P-name
Category
Price
Products
Loc-Id
P-Id
Time-Id
Sale
Sales
Tine-Id
Date
Week
Month
Quarter
Year
Times