**Physical interval definition**

• Between a lower bound and the current row

ROWS 2 PRECEDING

• Between lower and upper bounds

ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING ROWS BETWEEN 3 PRECEDING AND 1 PRECEDING

• Between the beginning (or the end) of a partition and the current row

ROWS UNBOUNDED PRECEDING (o FOLLOWING)

**Logical interval definition**

RANGE 2 MONTH PRECEDING

**Show, for each city and month – sale amount – average on the current month and the two previous months, separately for each city**

SELECT City, Month, Amount, AVG(Amount) OVER (PARTITION BY City ORDER BY Month

ROWS 2 PRECEDING) AS MovingAvg FROM Sales

**Show, for each city and month – sale amount – cumulative sale amount for increasing months, separately for each city**

SELECT City, Month, Amount, SUM(Amount) OVER (PARTITION BY City ORDER BY Month

ROWS UNBOUNDED PRECEDING) AS CumeTot

FROM Sales

**Show, for each city and month – sale amount – total sale amount on the whole time period for the current city**

SELECT City, Month, Amount, SUM(Amount)

OVER (PARTITION BY City) AS TotalAmount

FROM Sales

**Show, for each city and month – sale amount – ratio between current row amount and grand total – ratio between current row amount and total amount by city – ratio between current row amount and total amount by month**

SELECT City, Month, Amount Amount/SUM(Amount) OVER () AS TotalFract

Amount/SUM(Amount) OVER (PARTITION BY City) AS CityFract

Amount/SUM(Amount) OVER (PARTITION BY Month) AS MonthFract FROM Sales

**Show, for each city and month – sale amount – average sale with respect to the current month and the two preceding months, separately for each city**

SELECT City, Month, SUM(Amount) AS TotMonth, AVG(SUM(Amount)) OVER (PARTITION BY City ORDER BY Month ROWS 2 PRECEDING) AS MovingAvg

FROM Sales WHERE GROUP BY City, Month

**Show, for each city in december – sale amount – rank on amount (sort the result in the former example by increasing city)**

SELECT City, Amount, RANK() OVER (ORDER BY Amount DESC) AS Ranking FROM Sales WHERE Month = 12 ORDER BY City

**Compute total sales in the year 2000 for the following groups – month – month, city, product**

SELECT City, Month, Pkey, SUM(Amount) AS TotSales FROM Time T, Shop S, Sales S

WHERE T.Tkey = S.Tkey AND S.Skey = S.Skey AND Year = 2000

GROUP BY GROUPING SETS (Month, (City,Month,Pkey))

**Select, separately for each city and for each date, the amount and the average amount over the current and the previous two rows (((**Physical aggregation)))

SELECT Date, Amount, AVG(Amount)

OVER ( PARTITION BY City ORDER BY Date

ROWS 2 PRECEDING ) AS MovingAverage

FROM Sales ORDER BY Date

**(((**Logical aggregation)))

SELECT Date, Amount, AVG(Amount)

OVER ( PARTITION BY City ORDER BY Date

RANGE BETWEEN INTERVAL ‘2’

DAY PRECEDING AND CURRENT ROW ) AS Last3DaysAverage FROM Sales ORDER BY Date;

**Select for each item the total amount sold and the ranking according to the total amount sold**

SELECT COD\_I, SUM(SoldAmount),

RANK() OVER ( ORDER BY SUM(SoldAmount) ) AS SalesRank FROM Facts GROUP BY COD\_I

**Select for each item the code, the weight, the total amount sold, the ranking according to the weight and the ranking according to the total amount sold**

SELECT Item.COD\_I, Item.Weight,

RANK() OVER (ORDER BY Item.Weight ) AS WeightRank RANK() OVER (ORDER BY SUM(SoldAmount) ) AS SalesRank

FROM Facts, Item WHERE Facts.COD\_I = Item.COD\_I GROUP BY Item.COD\_I, Item.Weight ORDER BY WeightRank

**Select ◼ the top two most sold items ◼ their code ◼ their weight ◼ the total amount sold ◼ and their ranking according to the total amount sold**

SELECT \*

FROM (SELECT COD\_I, SUM(SoldAmount),

RANK() OVER (ORDER BY SUM(SoldAmount)) AS SalesRank

FROM Facts GROUP BY COD\_I)

WHERE SalesRank<=2

**Consider the orders with type “Partner order”. Separately for mean of transport and trimester, analyze: the average delivery time, the average number of deliveries per time slot (the average number of deliveries made in an hour), assign a rank to the trimesters based on decreasing number of kilometers run on average in a minute, separately by mean of transport.**

SELECT transport, trimester, SUM(#time)/SUM(#deliveries), SUM(#deliveries)/COUNT(DISTINCT timeSlot)

RANK() OVER (ORDER BY SUM(#kilometers)/SUM(#time) DESC

PARTITION BY transport)

FROM Time T, Deliveries D, Restaurant Rs

WHERE d.rid = rs.rid and d.did = t.did AND orderType=`Partner order' GROUP BY transport, trimester

**Consider the restaurants which have “pizza” among the associated categories. Carry out the analysis separately for payment method, delivery city and month. Analyze: the cumulative monthly revenue from the beginning of each trimester, the average revenue per delivery, the percentage of revenue with respect to the total revenue considering all the payment methods**

SELECT paymentM, dCity, month, trimester SUM(SUM(revenue))

OVER(PARTITION BY trimester, paymentM, dCity ORDER BY month ROWS UNBOUNDED PRECEDING)

SUM(revenue)/SUM(#deliveries)

100\*SUM(revenue)/SUM(SUM(revenue)) OVER(PARTITION BY month, dCity)

FROM Time T, Deliveries D, Restaurant R, Location L WHERE d.did = t.did and d.rid = rs.rid and d.lid = l.lid and r.pizza=true

GROUP BY month, trimester, paymentMethod, deliveryCity

**Separately for each purchase mode and for each purchase month, analyze: the average daily revenue, the cumulative revenue from the beginning of the year, the percentage of tickets related to the considered purchase mode over the total number of tickets of the month**

SELECT purchasemode, pmonth, sum(revenue)/count(distinct pdate),

sum(sum(revenue)) over (partition by purchasemode, pyear order by pmonth rows unbounded preceding)

100\*sum(numtickets)/sum(sum(numtickets)) over (partition by pmonth)

FROM musical\_events me, purchasedate pd, payment p WHERE me.pdid = pd.pdid and me.pid = p.pid

GROUP BY purchasemode, pmonth, pyear

**Considering the events that took place in 2017, separately for each singer/band nationality and for each city, analyze: the average revenue for a ticket, the percentage of revenue over the total revenue for the corresponding region**

SELECT nationality, city, sum(revenue)/sum(numtickets), 100\*sum(revenue)/sum(sum(revenue)) over (partition by nationality, region)

FROM musical\_events me, eventdate ed, eventlocation el, tour t

WHERE me.edid = ed.edid and me.elid = el.elid and me.tid = t.tid and eyear=2017 GROUP BY nationality, city, region

**For each song and month, compute the following metrics: the total number of streamings the cumulative total number of streamings since the beginning of the year assign a rank to each song, separately for each album, based on the monthly number of streamings (rank 1st the most streamed song of the album for each month)**

SELECT month, song, SUM(NumberOfStreamings), SUM(SUM(NumberOfStreamings))

OVER (PARTITION BY songId, year ORDER BY month ROWS UNBOUNDED PRECEDING)

RANK() OVER (PARTITION BY album, month ORDER BY SUM(NumberOfStreamings) DESC),

FROM Song S, Time T, MusicStreaming MS WHERE S.SongId=MS.SongId AND T.Timeid=MS.Timeid

GROUP BY song, songId, month, year, album

**Separately for each song and province of the user, compute the following metrics: the average number of monthly likes the percentage of the number of likes with respect to the total number of likes received by users in the same country the number of likes of the album in the user province**

SELECT province, song, SUM(NumberOfLikes)/ COUNT(DISTINCT month), 100\*SUM(NumberOfLikes)/SUM(SUM(NumberOfLikes)) OVER (PARTITION BY songId, country), SUM(SUM(NumberOfLikes)) OVER (PARTITION BY album, province)

FROM Song S, MusicStreaming MS, UserLocation Lm, Time T WHERE S.SongId=MS.SongId AND L.UserLocationId=MS.PlatformId AND T.TimeId=MS.TimeId

GROUP BY song, songId, province, country, album

**Separately for each videogame and store city, compute the following metrics: A. the percentage of copies of the videogame sold with respect to the total copies sold in the store province B. assign a rank to each videogame separately for video game company and city, based on its sales (rank 1st the video game with the highest number of sold copies for each city)**

SELECT VideoGameName, S.city, (S.province), (S.company) 100\*SUM(total\_sold\_videogames)/SUM(SUM(total\_sold\_videogames))

OVER (PARTITION BY S.province, V.CodV) as A,

RANK() OVER (PARTITION BY S.city, V.company ORDER BY SUM(#total\_sold\_videogames) DESC) as B

FROM VideoGame V, Fact F, Store S WHERE V.CodV=F.CodV and S.StoreS=F.CodV GROUP BY V.CodV, S.city, S.province, V.Company, VideoGameName

**Separately for each store city and bimester, compute the following metrics, only for the videogames appropriate for children: A. the cumulative revenues since the beginning of the semester B. the daily average revenues C. the percentage of revenues in the bimester with respect to the revenues in the semester, for each city**

SELECT city, bimester, (semester), SUM(SUM(total\_revenues))

OVER ( PARTITION BY city, semester ORDER BY bimester ROWS UNBOUNDED PRECEDING) as A,

SUM(total\_revenues)/COUNT(distinct date) as B, 100\*SUM(total\_revenues)/SUM(SUM(total\_revenues)) OVER (PARTITION BY city, semester) as C,

FROM VideoGame V, Fact F, Store S

WHERE T.CodT=F.CodT and S.StoreS=F.CodV and V.CodV=F.CodV and forChildren=1

GROUP BY city, bimester, semester

**Separately for each plant species and month, compute the following metrics: A. the daily average number of plants B. the monthly percentage of the number of plants of the species with respect to the number of plants of the genus C. the cumulative total number of plants since the beginning of the year**

SELECT plantSpecies, month, SUM(numberOfPlants)/COUNT(DISTINCT date) as A,

100\*SUM(numberOfPlants)/SUM(SUM(numberOfPlants)) OVER (PARTITION BY genus, month) as B,

SUM(SUM(numberOfPlants)) OVER ( PARTITION BY plantId, plantSpecies, year ORDER BY month ROWS UNBOUNDED PRECEDING) as C

FROM Plant P, Time T, Gardens G

WHERE P.PlantId=G.PlantId AND T.Timeid=G.Timeid

GROUP BY plantId, plantSpecies, month, year, genus

**Consider only the garden centers having the “parking” service. Separately for each garden center and plant genus, compute the following metrics: A. the average revenue per plant B. the total revenues of the plant family, for each garden center C. assign a rank to each garden center within its province, based on its total revenues (rank 1st the garden center with the highest revenue in its province for each plant genus)**

SELECT gardenCenter, genus, SUM(revenue)/SUM(numberOfPlants) as A, SUM(SUM(revenue)) OVER (PARTITION BY family, gardenCenter) as B,

RANK() OVER (PARTITION BY province, genus ORDER BY SUM(revenue) DESC) as C

FROM Plant P, Gardens G, GardenCenter GC WHERE P.PlantId=G.PlantId AND G.GardenCenterId=GC.GardenCenterId AND parking=True GROUP BY gardenCenter, genus, family, province

**Separately for each courier agency and departure city, compute the following metrics: A. the percentage of packages with respect to the total number of packages of the agency for the departure region B. the average weight per package C. assign a rank to each courier agency within its corporate group, based on its total number of packages (rank 1st the courier agency with the highest number of shipped packages in its corporate group for each departure city)**

SELECT CourierAgencyName, L.city 100\*SUM(#packages)/SUM(SUM(#packages)) OVER (PARTITION BY L.region, CourierAgencyId) as B,

SUM(total\_weight)/SUM(#packages) as A,

RANK() OVER (PARTITION BY L.city, CorporateGroup ORDER BY SUM(#packages) DESC) as C

FROM CourierAgency CA, Location L, Shippings S

WHERE CA.CourierAgencyId=s.CourierAgencyId and S.DepartureLocationId=L.LocationId

GROUP BY CourierAgencyId, CourierAgencyName, L.city, L.region, CorporateGroup

**Separately for each month, departure province and arrival province, compute the following metrics: A. the daily average number of shipped packages B. the cumulative total weight of delivered packets since the beginning of the semester**

SELECT month, L1.province, L2.province, (6M), SUM(#packages)/COUNT(distinct date) as A,

SUM(SUM(total\_weight)) OVER ( PARTITION BY L1.province, L2.province, 6M ORDER BY month ROWS UNBOUNDED PRECEDING) as B,

FROM Location L1, Location L2, Shippings S, Time T WHERE S.DepartureLocationId=L1.LocationId and S.ArrivalLocationId=L2.LocationId and T.TimeId=S.TimeId GROUP BY month, L1.province, L2.province, 6M

**Consider beers distributed in packs with the format: 6 bottles. Separately for production month and factory, analyze: the cumulative monthly number of produced liters from the beginning of the year, the monthly percentage of liters produced with respect to the total annual liters for the considered factory, assign a rank to the factories according to the total number of produced liters in decreasing order, separately for each production month.**

SELECT sum(sum(#liters)) over (partition by pyear, fid order by pmonth rows unbounded preceding),

100\* sum(#liters)/sum(sum(#liters)) over (partition by pyear, fid),

rank() over (partition by pmonth order by sum(#liters) desc )

FROM productiontime pt, factory f, beersupply bs WHERE bs.tprodid = pt.tprodid and bs.fid = f.fid and packformat=6

GROUP BY pmonth, pyear, fid

**Consider beers produced in 2018. Run the analysis separately for beer type, storage and month of arrival at the storage. Analyze: the average revenue for a bottle, the average daily (consider the day of arrival at the storage) number of produced bottles, the percentage of bottles of the considered beer type with respect to the total considering all the beer types.**

SELECT sum(revenue)/sum(#bottles), sum(#bottles)/count(distinct sdate), 100\*sum(#bottles)/sum(sum(#bottles)) over(partition by sid, smonth)

FROM beersupply bs, productiontime pt, beer b, storage s, supplytime st WHERE bs.tprodid = pt.tprodid and bs.bid = b.bid and bs.sid = s.sid and bs.tsupid = st.tsupdid and pyear=2018

GROUP BY beertype, sid, smonth

**Consider only private repositories. Separately for month and repository name, analyze: the number of commits made on average in a day, the number of commits made on average in a branch, the monthly cumulative number of commits from the beginning of the year.**

SELECT repositoryname, month, sum(#commits)/count(distinct date), sum(#commits)/count(distinct bid), sum(sum(#commits)) over (partition by repositoryname, year order by month rows unbounded preceding)

FROM git g, time t, branch b WHERE g.bid = b.bid and g.tid = t.tid AND private = True

GROUP BY repositoryname, month, year

**Consider data related to repositories which include the ‘Scala’ language. Separately for branch and work team, analyze: the ratio between the number of additions and the number of deletions, the percentage of additions with respect to the total of the repository the branch belongs to, assign a rank to work teams based on the ratio between the number of additions and the number of deletions, separately for each branch**.

SELECT bid, branchname, repositoryname, workteam, sum(#additions)/sum(#deletions), sum(#additions)/sum(sum(#deletions)) over (partition by repository, workteam),

rank() over (partition by bid order by sum(#additions)/sum(#deletions))

FROM git g, branch b, collaborator c WHERE g.bid = b.bid and g.cid = c.cid and scala\_language = true

GROUP BY bid, branchname, repositoryname, workteam

**Consider the rentals payed with credit card. Separately for each month, client type, province of the client’s domicile, analyze: the average number of kilometers for rental, the cumulative number of kilometers from the beginning of the year, the average daily kilometers.**

SELECT clienttype, clientprovince, month, year, sum(#kilometers)/sum(#rentals), sum(sum(#kilometers) over (partition by clienttype, clientprovince, year order by month rows unbounded preceding) sum(#kilometers)/count(distinct date)

FROM carrental cr, rentaltime rt, clientprofile cp WHERE cr.rtid = rt.rtid and cr.cid = cp.cid and paymentmode=’credit card’

GROUP BY clienttype, clientprovince, month, year

**Consider the rentals of “truck” vehicle type. Separately for each rental start month and for each departure province, analyze: the average revenue per kilometer, the percentage of revenue over the total revenue of the year, the percentage of revenue over the total revenue for the corresponding departure region**

SELECT month, year, d.province, d.region, sum(revenue)/sum(#kilometers), 100\*sum(revenue)/sum(sum(revenue)) over (partition by year, d.province)

100\*sum(revenue)/sum(sum(revenue)) over (partition by month, d.region)

FROM carrental cr, rentaltime rt, location d, vehicle v WHERE cr.rtid = rt.rtid and cr.arrlid = d.lid and cr.vid = v.vid and v.type=’truck’

GROUP BY month, year, d.province, d.region

**Show for each category of items, • The category • The total sale amount for the category • The total number of sold items within the category • The rank of the category according to its total sale amount • The rank of the category according to its total number of sold items • Sort the result for increasing value of total number of sold items within the category**

SELECT CategoryName, SUM(TotAmount), SUM(NumSoldItems)

RANK() over (ORDER BY SUM(TotAmount) DESC) as RANK\_ TotAmount,

RANK() over (ORDER BY SUM(NumSoldItems) DESC) as RANK\_ NumSoldItems,

FROM Sales S, Category C WHERE S.ItemCategoryID = C.CategoryID GROUP BY C.CategoryID, CategoryName ORDER BY RANK\_ NumSoldItems;

**Show, for each province • The province • The region of the province • The total sale amount for the province • The rank of the province according to its total sale amount, separately for each region**

SELECT Province, Region, SUM(TotAmount),

RANK() over (PARTITION by Region ORDER BY SUM(TotAmount) DESC) as RANK\_TotAmountRegion

FROM SALES S, CUSTOMER C WHERE S.CustomerID = C.CustomerID GROUP BY Province, Region;

**Show, for each province and month • The province • The region of the province • The month • The total sale amount for the province in the current month • The rank of the province according to its total sale amount, separately for each moth**

SELECT Province, Region, Month, SUM(TotAmount), RANK() over (PARTITION by Month ORDER BY SUM(TotAmount) DESC) as CumulativeAmount

FROM SALES S, CUSTOMER C, TIME T WHERE S.CustomerID=C.CustomerID AND S.TimeID=T.TimeID GROUP BY Province, Region, Month;

**Show, for each region and month • The region • The month • The total sale amount for the region in the current month • The cumulative sale amount for increasing months, separately for each region**

SELECT Region, Month, SUM(TotAmount), SUM(SUM(TotAmount)) over (PARTITION by Region ORDER BY Month ROWS UNBOUNDED PRECEDING) as CumulativeAmount

FROM SALES S, CUSTOMER C, TIME T WHERE S.CustomerID=C.CustomerID AND S.TimeID=T.TimeID GROUP BY Region, Month

**Show, for each object • The object code • The object type • The object total rentals (number of times the object has been rented) • The object total income (SUM(price)) • The object rank according to its total rentals • The object rank according to its total income**

SELECT O.ObjectID, O.ObjectType, COUNT(\*) as RentalNumber, SUM(Price) as TotalIncome,

RANK() over (ORDER BY COUNT(\*) DESC) as RentalRank, RANK() over (ORDER BY SUM(Price) DESC) as TotalIncomeRank

FROM Objects O, Rentals R WHERE O.ObjectID = R.ObjectID GROUP BY O.ObjectID, O.ObjectType

**Show, for each object and month • the object code • the object type • the month • the total income for the object in the current month (SUM(price)) • the object rank according to its total monthly income, separately for each month**

SELECT O.ObjectID, O.ObjectType, Month SUM(Price) as TotalIncome,

RANK() over (PARTITION BY Month ORDER BY SUM(Price) DESC) as TotalIncomeRank

FROM Objects O, Rentals R WHERE O.ObjectID = R.ObjectID GROUP BY O.ObjectID, O.ObjectType, Month

**Show, for each province • The province • The region of the province • The total sale amount for the province • The rank of the province according to its total sale amount, separately for each region**

SELECT Province, Region, SUM(TotAmount),

RANK() over (PARTITION by Region ORDER BY SUM(TotAmount) DESC) as RANK\_TotAmountRegion FROM SALES S, CUSTOMER C WHERE S.CustomerID = C.CustomerID GROUP BY Province, Region;

**Show, for each region and month • The region • The month • The total sale amount for the region in the current month • The cumulative sale amount for increasing months, separately for each region**

SELECT Region, Month, SUM(TotAmount), SUM(SUM(TotAmount)) over (PARTITION by Region ORDER BY Month ROWS UNBOUNDED PRECEDING) as CumulativeAmount

FROM SALES S, CUSTOMER C, TIME T WHERE S.CustomerID=C.CustomerID AND S.TimeID=T.TimeID GROUP BY Region, Month;

**In 2005, for each state and month, analyze the portion of rooms which are reserved, free, and unavailable.**

SELECT state, month, round(SUM(reserved)\*100/SUM(total),2) as Reserved, round(SUM(free)\*100/SUM(total),2) as Free, round(SUM(unavailable)\*100/SUM(total),2) as Unavailable FROM Rooms r, Time t, Hotel h WHERE r.hotel\_id=h.hotel\_id AND r.time\_id = t.time\_id AND t.year = 2005 GROUP BY h.state, t.month ORDER BY month ASC,state ASC;

**In 2005, for each state, analyze the portion of rooms which are reserved. Associate a rank to each state according to the portion of reserved rooms for that state in 2005 with respect to all the rooms for that state. The state with the highest ratio of reserved rooms in 2005 must rank first**

B) SELECT state, round(SUM(reserved)\*100/SUM(total),2) as Reserved, RANK() OVER ( ORDER BY SUM(reserved)/SUM(total) DESC ) as RankState FROM Rooms r, Time t, Hotel h WHERE r.hotel\_id=h.hotel\_id AND r.time\_id = t.time\_id AND year = 2005 GROUP BY state ORDER BY RankState ASC;

**In 2005, for each state and month, analyze the income of 4-star hotels and the cumulative income of 4-star hotels**

C) SELECT state, month, SUM(income) as TotIncome, SUM(SUM(income)) OVER ( PARTITION BY state ORDER BY month ROWS UNBOUNDED PRECEDING ) as CumIncome FROM Rooms r, Time t, Hotel h WHERE r.hotel\_id=h.hotel\_id AND r.time\_id = t.time\_id AND category = 4 AND year=2005 GROUP BY state, month

**For each state and year, analyze the total income of public holidays.**

D) SELECT state, year, SUM(income) as TotIncome FROM Rooms r, Time t, Hotel h WHERE r.hotel\_id=h.hotel\_id AND r.time\_id = t.time\_id AND holiday = 'y' GROUP BY state, year ORDER BY year, state

**In 2005, for each hotel, analyze the total income of the rooms with satellite TV and whirlpool bath**

E) SELECT r.hotel\_id, hotel\_name, SUM(income) as TotIncome

FROM Rooms r, Features f, Hotel h, Time t WHERE r.hotel\_id=h.hotel\_id AND r.time\_id = t.time\_id AND r.features\_id = f.features\_id AND year=2005 AND satellite\_TV = 'y' AND whirpool\_bath = 'y' GROUP BY r.hotel\_id, hotel\_name

**a) In the first trimester of 2003, regarding the storehouses in Turin, select the total value of the products stored in each storehouse at any given date, and select the average daily total value of the products in each storehouse during the previous week (including the current date).**

select storehouse, date, sum(totValue),

avg( sum(totValue) ) over (partition by storehouse order by date range between interval ‘6’ day preceding and current row)

from products p, storehouses sh, time t where p.storehouseID=sh.storehouseID and p.timeID=t.timeID and t.trimester=1/2003 and sh.city=’Turin’ group by storehouseID, storehouse, date;

Card: 5 x (30 x 3) = 450 << 7300k ◊ a materialized view on this query is convenient. Removing the constraints on trimester and city, the view would be useful to answer query d and e too.

NB: averaging the daily total value over the last week could be done using the sum(sum(totValue)/7) expression, which handles missing days as if their totValue were 0, while the proposed solution fills missing values with the week average; furthermore note that totValue is a level measure, thus there should be no missing values in the data warehouse

**b) In 2004, for each city and date, select the percentage of daily free surface of the storehouses. Give a rank to the results (rank 1 is the lowest percentage).**

select city, date, sum(m2free)/sum(m2tot)\*100,

rank() over (order by sum(m2free)/sum(m2tot) asc)

from surface s, storehouses sh, time t where s.storehouseID=sh.storehouseID and s.timeID=t.timeID and t.year=2004 group by city, date;

Card: 90 x 365 = 32850 ≈ 73000 ◊ a materialized view on this query is NOT convenient

**c) In the first 6 months of 2004, select the percentage of free surface for each storehouse and date.**

select storehouse, date, (m2free/m2tot)\*100,

from products p, storehouses sh, time t where p.storehouseID=sh.storehouseID and p.timeID=t.timeID and t.month>=1/2004 and t.month<=6/2004 group by storehouseID, storehouse, date;

Card: 100 x (30 x 6) = 18000 ≈ 73000 ◊ a materialized view on this query is NOT convenient

**a) In 2004, including only the properties in cities where universities are present, select the average monthly rent cost for each city and month and the average monthly rent cost since the beginning of the year for each city and month.**

select city, month, sum(totPrice)/sum(numProperties),

( sum(sum(totPrice)) / sum(sum(numProperties)) ) over (partition by city order by month rows unbounded preceding)

from properties p, location l, month m where p.locationID=l. locationID and p.monthID=m.monthID and year=2004 and university=’y’ group by city, month;

**d) In 2003, select the average daily total value of products for each storehouse and month.**

select storehouse, month, sum(totValue)/count(distinct date)

from products p, storehouses sh, time t where p.storehouseID=sh.storehouseID and p.timeID=t.timeID and t.year=2003

group by storehouseID, storehouse, month;

\*\*Alternative solution:

select distinct storehouse, month, avg( sum(totValue) ) over (partition by storehouse, month) from products p, storehouses sh, time t where p.storehouseID=sh.storehouseID and p.timeID=t.timeID and t.year=2003 group by storehouseID, storehouse, date, month;

Card: 100 x 12 = 1200 << 7300k ◊ a materialized view on this query is convenient and it helps to answer query e too.

NB: the DISTINCT command does not remove rows with the same storehouse; it removes duplicate rows considering all attribute values of each row.

**e) In 2003, select the average daily total value of products for each region.**

select region, sum(totValue)/count(distinct date)

from products p, storehouses sh, time t

where p.storehouseID=sh.storehouseID and p.timeID=t.timeID and t.year=2003 group by region;

\*\*Alternative solution:

select distinct region, avg(sum(totValue)) over (partition by region) from products p, storehouses sh, time t where p.storehouseID=sh.storehouseID and p.timeID=t.timeID and t.year=2003 group by region, date;

Card: 40 << 7300k ◊ a materialized view on this query is convenient.

**f) In 2004, select the average percentage of daily free surface for each month and region.**

select distinct region, month, avg(sum(m2free)/sum(m2tot)\*100) over (partition by region, month)

from surface s, storehouses sh, time t where s.storehouseID=sh.storehouseID and s.timeID=t.timeID and t.year=2004

group by region, month, date;

Card: 40 x 12 = 480 << 7300k ◊ a materialized view on this query is convenient

**b) In September 2004, including only the properties in the province of Turin, select the total number of free properties for each city and week, the ratio of the total number of free properties for each city and week and the total number of free properties for the same week. Give a rank according to the total number of free properties (rank 1st the highest number). Sort the data according to the rank.**

select city, week, sum(numProperties), sum(numProperties) / ( sum(sum(numProperties)) over (partition by week) ), rank() over (order by sum(numProperties) desc) as position from properties p, location l, month m, week w where p.locationID=l. locationID and p.monthID=m.monthID and p.weekID=w.weekID and month=’September/2004’ and province=’Turin’ group by city, week order by position;

**c) In summer 2005, including only the attics with bed, fridge, and table in Rome, for each city area (district) and for each range of the monthly rent price, select the average number of users per each property who have added the property in their favorite list, and select the average number of users per each property of the city area who have added the property in their favorite list.** **Order the results according to the city area and the average number of users.**

select district, surfaceMin, surfaceMax, sum(numUsers) / sum(numProperties) as avgInterestedUsers,

( sum(sum(numUsers)) / sum(sum(numProperties)) ) over (partition by district)

from favorites f, location l, season s, year y, furniture fu, type t, price\_range pr where f.locationID=l.locationID and f.seasonID=s.seasonID and f.furnitureID=fu.furnitureID and f.typeID=t.typeID and f.priceID=pr.priceID and season=’summer’ and year=2005 and type=’attic’ and city=’Rome’ and bed=’y’ and fridge=’y’ and table=’y’

group by district, surfaceMin, surfaceMax order by district, avgInterestedUsers;

**d) Including only the properties with bed and table in cities where universities are present, select the average monthly rent cost per property for each city, month, and year, select the average monthly rent cost per square meter for each city, month, and year, and select the average monthly rent cost per property of the city since the beginning of the year for each city, month, and year.**

select city, month, year, sum(totPrice) / sum(numProperties), sum(totPrice) / sum(totSurface), (sum(sum(totPrice)) / sum(sum(numProperties)) ) over (partition by city, year order by month rows unbounded preceding)

from properties p, location l, month m, furniture f where p.locationID=l.locationID and p.monthID=m.monthID and p.furnitureID=f.furnitureID and bed=’y’ and table=’y’ and university=’y’ group by city, month, year

**e) Including only the properties in Piedmont (region), in September, October, and November 2004, select for each city the average monthly rent cost per property and the average monthly rent cost per property of the province in which the city is located.**

select city, sum(totPrice) / sum(numProperties), (sum(sum(totPrice)) / sum(sum(numProperties)) ) over (partition by province)

from properties p, location l, month m where p.locationID=l.locationID and p.monthID=m.monthID and month>=9/2004 and month<=11/2004 and region=’Piedmont’ group by city

**f) In 2004, including only the properties with bed and table in cities where universities are present, select the average monthly rent cost per property for each city and month, and the average monthly rent cost per square meter for each city and month.**

select city, month, sum(totPrice) / sum(numProperties), sum(totPrice) / sum(totSurface),

from properties p, location l, month m, furniture f where p.locationID=l.locationID and p.monthID=m.monthID and p.furnitureID=f.furnitureID and year=2004 and university=’y’ and bed=’y’ and table=’y’ group by city, month

**Separately for city, region, and year, compute:  
- the daily average number of transitions**

**-the percentage of transitions in each city, with respect to the total of the region**

**-inside each region assign a rank to to the cities based on the decreasing number of transitions**

SELECT city, region, year

    SUM(#transitions)/COUNT(DISTINCT Date)

    SUM(#transitions)/SUM(SUM(#transitions)) OVER (PARTITION BY region, year)

    RANK() OVER (PARTITION BY region, year ORDER BY SUM(#transitions) DESC),

FROM CellSite, Date, Activity

WHERE CellSite.CellSiteId=Activity.CellSiteId AND Activity.DateId=Date.DateId

GROUP BY city, region, year

**Separately for each city, month and virtual network operator, compute:  
-the total number of call minutes**

**-the percentage of call minutes of each city with respect to the monthly total over all the cities, for the considered operator**

**-the cumulative total of call minutes from the beginning of the year**

select city, month, virtualnetworkoperator ,sum(minuteofcall)

100\*(sum(minutesofcall)/sum(sum(minuteofcall)) over(partition by month, virtualnetooperator )

sum(sum(minuteofcall)) over (partition by city,year, virtualnetworkoperator

      order by month

     rows unbounded preceding)

from activity A, date D, cellsite C

where A.dateid = D.dateid and A.cellsiteid = C.cellsiteid

group by city, month, virtualnetworkoperator

**Separately for each city and month, compute:**

**-the total income**

**-the percentage of income in each month, with respect to the total of the year**

**- inside each region, assign a rank to the cities based on the income (rank 1st the highest income city), separately for each month**

SELECT city, month, SUM(Income),

100\*SUM(Income)/SUM(SUM(Income)) OVER (PARTITION BY city, year),

RANK() OVER (PARTITION BY region, month ORDER BY SUM(Income) DESC),

FROM Store S, Date D, Purchase P

GROUP BY city, month, year, region

**Separately for each month and brand, compute:**

* **the average income per item**
* **the daily average number of items**
* **the cumulative total number of items since the beginning of the year**

SELECT Brand, month, year,

SUM(income)/Sum(number\_of\_items\_sold),

SUM(number\_of\_items\_sold)/COUNT(DISTINCT Date),

SUM(SUM(number\_of\_items\_sold)) OVER (PARTITION BY Brand, year

ORDER BY month ROWS UNBOUNDED PRECEDING)

FROM Store S, Date D, Purchase P

WHERE P.StoreId=S.StoreId AND P.Timeid=D.Timeid

GROUP BY Brand, month, year

**Separately for each restaurant and bimester, compute:**

**\*the percentage of income of the restaurant, with respect to the total income of its restaurant district**

**\*the average income for delivery**

**\*for each restaurant district, assign a rank to to the restaurants based on the income (rank 1st the highest income restaurant), separately for each bimester**

SELECT 2M, restaurant, D.district,

100\*SUM(Income)/SUM(SUM(Income)) OVER (PARTITION BY 2M, D.district),

SUM(Income)/SUM(NumberDeliveries),

      RANK() OVER (PARTITION BY D.district, 2M ORDER BY SUM(Income) DESC),

FROM Delivery Del, Time T, Restaurant R, District D

WHERE R.RestaurantId=Del.RestaurantId AND T.Timeid=Del.Timeid and R.RestaurantDistrict=D.RestaurantDistrict

GROUP BY 2M, restaurant, D.district

**Separately for each month and destination district, compute:**

* **the percentage of number of deliveries  in each month, with respect to the total of the year**
* **the percentage of number of deliveries  for destination district, with respect to the total of the destination city**
* **the daily average number of deliveries**

Select month, D.district

     100\*SUM(NumberDeliveries)/SUM(SUM(NumberDeliveries)) OVER (PARTITION BY D.district, year),

     100\*SUM(NumberDeliveries)/SUM(SUM(NumberDeliveries)) OVER (PARTITION BY month, D.city),

     SUM(NumberDeliveries)/COUNT(DISTINCT Date),

FROM Delivery Del, Time T, District D

WHERE T.Timeid=Del.Timeid and Del.DestinationDistrictId=D.DistrictId

GROUP BY month, year, D.district, D.city

**The name of the renting point is unique. The ChildrenBike is True if the bike is for children, False otherwise. Separately for each Renting Point and trimester**, **compute:**

**● the average number of rented bikes per renting**

**● the cumulative total number of rentings since the beginning of the year**

**● for each region, assign a rank to the renting points based on the total rentings (rank 1st the highest number), separately for each trimester**

SELECT 3M, Name

SUM(total\_rented\_bikes) / SUM(total\_rentings),

SUM(SUM(total\_rentings)) OVER (PARTITION BY R.Name, year

ORDER BY 3M ROWS UNBOUNDED PRECEDING),

RANK() OVER (PARTITION BY region, 3M

ORDER BY SUM(total\_rentings) DESC),

FROM Fact F, Time T, Bike B, RentingPoint R

WHERE T.CodT=F.CodT and B.CodB=F.CodB and R.CodP=F.CodP

GROUP BY 3M, R.Name, R.Region, year

Alternative solution: […] GROUP BY 3M, R.CodP, R.Region, year

**The name of the renting point is unique. The ChildrenBike is True if the bike is for children, False otherwise. Separately for each city, producer, and month, compute:**

**●the percentage of incomes in each month and city of each producer, with respect to the producer yearly total for the city**

**●the average income per rented bike**

**●for each city, assign a rank to the producer based on the income (rank 1st the highest income), separately for each month**

Select city, B.Producer, month,

100 \* SUM(total\_income) / SUM(SUM(total\_income))

OVER (PARTITION BY producer, city, year), SUM(total\_income)/SUM(total\_rented\_bikes),

RANK() OVER (PARTITION BY city, month

ORDER BY SUM(total\_income) DESC),

FROM Fact F, Time T, Bike B, RentingPoint R

WHERE T.CodT=F.CodT and B.CodB=F.CodB and R.CodP=F.CodP

GROUP BY month, year, City, B.Producer