

## Queries

### Query 1

```
SELECT SUM(Price), dateYear, phoneRateType
FROM DWABD.Facts F, DWABD.TimeDim T, DWABD.PhoneRate P
WHERE F.Id_time = T.Id_time and F.Id_phoneRate = P.Id_phoneRate
GROUP BY cube(phoneRateType, dateYear)
```

```
SELECT dateYear, phoneRateType, SUM(Price),
SUM(SUM(Price)) OVER (PARTITION BY phoneRateType, SUM(SUM(Price)) OVER
(PARTITION BY dateYear), SUM(SUM(Price)) OVER ()
FROM DWABD.Facts F, DWABD.TimeDim T, DWABD.PhoneRate P
WHERE F.Id_time = T.Id_time and F.Id_phoneRate = P.Id_phoneRate
GROUP BY phoneRateType, dateYear
```

### Query 2

```
SELECT DateMonth, DateYear, SUM(NumberOfCalls) as TotNumOfCalls, SUM(price) as
totalIncome, RANK() over (ORDER BY SUM(price) DESC) as RankIncome
FROM DWABD.FACTS F, DWABD.TIMEDim Te
WHERE F.id_time=Te.id_time
GROUP BY DateMonth, DateYear;
```

### Query 3

```
SELECT DateMonth, SUM(NumberOfCalls) as TotNumOfCalls,
RANK() over (ORDER BY SUM(NumberOfCalls) DESC) as RankNumOfCalls
FROM dwabd.FACTS F, dwabd.TIMEDIM Te
WHERE F.id_time=Te.id_time
AND DateYear=2003
GROUP BY DateMonth;
```

### Query 4

```
SELECT DayOfMonth , SUM(Price), AVG(SUM(Price)) OVER ( ORDER BY DayOfMonth
RANGE BETWEEN INTERVAL '2' day preceding and current row) as avglast3days
FROM DWABD.FACTS F, DWABD.TIMEDIM Te
WHERE F.ID_time=Te.ID_time AND DateYear=2003 AND DateMonth= '7-2003'
GROUP BY DayOfMonth ;
```

```
SELECT DayOfMonth , SUM(Price),
AVG(SUM(Price)) OVER ( ORDER BY DayOfMonth ROWS 2 preceding) as avglast3days
FROM DWABD.FACTS F, DWABD.TIMEDIM Te
WHERE F.ID_time=Te.ID_time AND DateYear=2003 AND DateMonth= '7-2003'
GROUP BY DayOfMonth
ORDER BY DayOfMonth;
```

### Query 5

```
SELECT DateYear, DateMonth, SUM(Price) AS TOTINCOME,
SUM(SUM(PRICE)) OVER( PARTITION BY DateYear ORDER BY DateMonth ROWS UNBOUNDED
PRECEDING) AS CUMULATIVEINCOME
FROM DWABD.FACTS F, DWABD.TIMEDIM Te
WHERE F.ID_time=Te.ID_time
GROUP BY DateMonth, DateYear;
```

## Materialized views

The cardinality of all queries is at least one order of magnitude lower than those of the fact table. Hence, for each query it may be potentially useful to create a materialized view.

Queries 2, 3, and 5 are pretty similar. To answer these queries efficiently we can create a single materialized view, which is reported below.

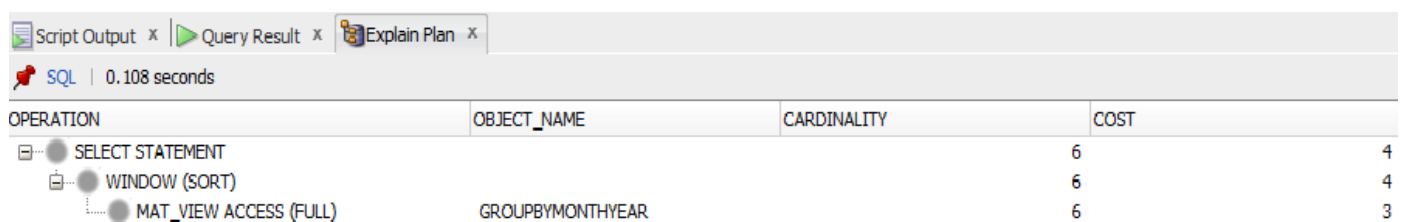
```
create materialized view GROUPBYMonthYear
  build immediate
  refresh on demand
  --enable query rewrite
as
SELECT DateMonth, DateYear, SUM(NumberOfCalls) as NumCalls, SUM(Price) as
TotPrice
FROM DWABD.FACTS F, DWABD.TIMEDIM T
WHERE F.ID_time = T.ID_time
GROUP BY DateMonth, DateYear;
```

## Queries using materialized view

### Query 2

```
SELECT DateMonth, DateYear, NumCalls, TotPrice,
RANK() over (ORDER BY TotPrice DESC) as RankPrice
FROM GROUPBYMonthYear;
```

### Explain plan



The screenshot shows a SQL client interface with three tabs: 'Script Output', 'Query Result', and 'Explain Plan'. The 'Explain Plan' tab is active, displaying the execution plan for the query. The plan shows a 'SELECT STATEMENT' operation with a cardinality of 6 and a cost of 4. This operation involves a 'WINDOW (SORT)' operation with a cardinality of 6 and a cost of 4. The window operation is implemented using 'MAT\_VIEW ACCESS (FULL)' on the 'GROUPBYMONTHYEAR' materialized view, with a cardinality of 6 and a cost of 3.

OPERATION	OBJECT_NAME	CARDINALITY	COST
SELECT STATEMENT		6	4
WINDOW (SORT)		6	4
MAT_VIEW ACCESS (FULL)	GROUPBYMONTHYEAR	6	3

### Query 3

```
SELECT DateMonth, NumCalls, RANK() over (ORDER BY NumCalls DESC) as RankCalls
FROM GROUPBYMonthYear
WHERE DateYear =2003;
```

## Explain plan

Script Output x Query Result x Autotrace x Explain Plan x  
SQL | 0.06 seconds

OPERATION	OBJECT_NAME	CARDINALITY	COST
SELECT STATEMENT		3	4
WINDOW (SORT)		3	4
MAT_VIEW ACCESS (FULL)	GROUPBYMONTHYEAR	3	3
Filter Predicates			
DATEYEAR=2003			

## Query 5

```
SELECT DateMonth, DateYear, TotPrice,  
SUM(TotPrice) over (PARTITION BY DateYear ORDER BY DateMonth rows unbounded  
preceding) as CumulativePrice  
FROM GROUPBYMonthYear;
```

## Explain plan

Script Output x Autotrace x Query Result x Explain Plan x  
SQL | 0.021 seconds

OPERATION	OBJECT_NAME	CARDINALITY	COST
SELECT STATEMENT		6	4
WINDOW (SORT)		6	4
MAT_VIEW ACCESS (FULL)	GROUPBYMONTHYEAR	6	3