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

<table>
<thead>
<tr>
<th>OPERATION</th>
<th>OBJECT_NAME</th>
<th>CARDINALITY</th>
<th>COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT STATEMENT</td>
<td>GROUPBYMonthYear</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>WINDOW (SORT)</td>
<td></td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>MAT_VIEW ACCESS (FULL)</td>
<td></td>
<td>6</td>
<td>3</td>
</tr>
</tbody>
</table>

Query 3

SELECT DateMonth, NumCalls, RANK() over (ORDER BY NumCalls DESC) as RankCalls
FROM GROUPBYMonthYear
WHERE DateYear = 2003;
Query 5

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