**Base Query 1 :**

**BQ1 involves the job fact, location dimension, and time dimension tables. In the following SELECT statement, the cross product join style is used with 3 tables in the FROM clause and 2 join conditions in the WHERE clause. Alternatively, the join operator style can be used with 2 join operations in the FROM clause. The GROUP BY clause must contain all non-aggregate columns (Location\_Id, Location\_Name, Sales\_Class\_Id, Sales\_Class\_Desc, Base\_Price, Time\_Year, and Time\_Month).**

-- **Base query BG1 in the revenue/costs area**

**-- Location and sales class summary of job quantity and amount**

**SELECT W\_Location\_D.Location\_Id, Location\_Name,**

**W\_Sales\_Class\_D.Sales\_Class\_Id, Sales\_Class\_Desc,**

**Base\_Price, Time\_Year, Time\_Month,**

**SUM ( QUANTITY\_ORDERED ) AS Sum\_Job\_Qty,**

**SUM ( QUANTITY\_ORDERED \* Unit\_Price ) AS Sum\_Job\_Amount**

**FROM W\_JOB\_F, W\_Location\_D, W\_TIME\_D, W\_Sales\_Class\_D**

**WHERE W\_Location\_D.Location\_ID = W\_Job\_F.Location\_Id**

**AND W\_JOB\_F.CONTRACT\_DATE = W\_TIME\_D.Time\_ID**

**AND W\_Job\_F.Sales\_Class\_Id = W\_Sales\_Class\_D.Sales\_Class\_Id**

**GROUP BY W\_Location\_D.Location\_Id, Location\_Name,**

**W\_Sales\_Class\_D.Sales\_Class\_Id, Sales\_Class\_Desc,**

**Base\_Price, Time\_Year, Time\_Month;**

**Base Query 2**

BQ2 involves 4 fact tables (job, subjob, shipment, and invoice line) and 2 dimension tables, location and time. In the following SELECT statement, the cross product join style is used with 6 tables in the FROM clause and 5 join conditions in the WHERE clause. Alternatively, the join operator style can be used with 5 join operations in the FROM clause. The GROUP BY clause must contain all non-aggregate columns (Job\_Id, Location\_Id, Location\_Name, Quantity\_Ordered, Unit-Price, Time\_Year, and Time\_Month). To facilitate formulation of analytic queries, the base query should be placed in a CREATE VIEW statement.

**-- BQ2 in the revenue/costs area**

**-- Location invoice revenue summary**

**-- Use contract year and month**

**SELECT W\_Sub\_Job\_F.Job\_Id,**

**W\_Location\_D.LOCATION\_ID, W\_LOCATION\_D.LOCATION\_NAME,**

**Quantity\_Ordered, Unit\_Price,**

**W\_TIME\_D.TIME\_YEAR, W\_TIME\_D.TIME\_MONTH,**

**SUM (Invoice\_Quantity) AS SumInvoiceQty,**

**SUM (Invoice\_Amount) AS SumInvoiceAmt**

**FROM W\_Job\_Shipment\_F, W\_Sub\_Job\_F, W\_Location\_D, W\_Time\_D,**

**W\_InvoiceLine\_F, W\_Job\_F**

**WHERE W\_Sub\_Job\_F.Sub\_Job\_Id = W\_Job\_Shipment\_F.Sub\_Job\_Id**

**AND W\_Job\_Shipment\_F.Invoice\_Id = W\_InvoiceLine\_F.Invoice\_Id**

**AND W\_Time\_D.Time\_Id = Contract\_Date**

**AND W\_Location\_D.Location\_Id = W\_InvoiceLine\_F.Location\_Id**

**AND W\_Job\_F.Job\_Id = W\_Sub\_Job\_F.Job\_Id**

**GROUP BY W\_Sub\_Job\_F.Job\_Id, W\_Location\_D.Location\_Id,**

**W\_LOCATION\_D.LOCATION\_NAME, Quantity\_Ordered, Unit\_Price,**

**W\_Time\_D.Time\_Year, W\_Time\_D.Time\_Month;**

**-- CREATE VIEW statement**

**CREATE VIEW LocRevenueSummary AS**

**SELECT W\_Sub\_Job\_F.Job\_Id,**

**W\_Location\_D.LOCATION\_ID, W\_LOCATION\_D.LOCATION\_NAME,**

**Quantity\_Ordered, Unit\_Price,**

**W\_TIME\_D.TIME\_YEAR, W\_TIME\_D.TIME\_MONTH,**

**SUM (Invoice\_Quantity) AS SumInvoiceQty,**

**SUM (Invoice\_Amount) AS SumInvoiceAmt**

**FROM W\_Job\_Shipment\_F, W\_Sub\_Job\_F, W\_Location\_D, W\_Time\_D,**

**W\_InvoiceLine\_F, W\_Job\_F**

**WHERE W\_Sub\_Job\_F.Sub\_Job\_Id = W\_Job\_Shipment\_F.Sub\_Job\_Id**

**AND W\_Job\_Shipment\_F.Invoice\_Id = W\_InvoiceLine\_F.Invoice\_Id**

**AND W\_Time\_D.Time\_Id = Contract\_Date**

**AND W\_Location\_D.Location\_Id = W\_InvoiceLine\_F.Location\_Id**

**AND W\_Job\_F.Job\_Id = W\_Sub\_Job\_F.Job\_Id**

**GROUP BY W\_Sub\_Job\_F.Job\_Id, W\_Location\_D.Location\_Id,**

**W\_LOCATION\_D.LOCATION\_NAME, Quantity\_Ordered, Unit\_Price,**

**W\_Time\_D.Time\_Year, W\_Time\_D.Time\_Month;**

**Base Query 3**

BQ3 involves 2 fact tables (job and subjob) and 3 dimension tables (location, time, and machine type). In the following SELECT statement, the cross product join style is used with 5 tables in the FROM clause and 4 join conditions in the WHERE clause. Alternatively, the join operator style can be used with 4 join operations in the FROM clause. The GROUP BY clause must contain all non-aggregate columns (Job\_Id, Location\_Id, Location\_Name, Time\_Year, and Time\_Month). To facilitate formulation of analytic queries, the base query should be placed in a CREATE VIEW statement.

**-- BQ3 in the revenue/costs area**

**-- Location subjob cost summary**

**-- Use contract year and month to match revenues/costs**

**SELECT W\_Sub\_Job\_F.Job\_Id,**

**W\_Location\_D.LOCATION\_ID ,W\_LOCATION\_D.LOCATION\_NAME,**

**W\_TIME\_D.TIME\_YEAR, W\_TIME\_D.TIME\_MONTH,**

**SUM(Cost\_Labor) AS SumLaborCosts,**

**SUM(Cost\_Material) AS SumMaterialCosts,**

**SUM(Cost\_Overhead) AS SumOvrhdCosts,**

**SUM(Machine\_Hours \* Rate\_Per\_Hour) AS SumMachineCosts,**

**SUM(Quantity\_Produced) AS SumQtyProduced,**

**SUM(Cost\_Labor + Cost\_Material + Cost\_Overhead +**

**(Machine\_Hours \* Rate\_Per\_Hour) ) AS TotalCosts,**

**SUM( Cost\_Labor + Cost\_Material + Cost\_Overhead + (Machine\_Hours \***

**Rate\_Per\_Hour) ) / SUM(Quantity\_Produced) AS UnitCosts**

**FROM W\_Job\_F, W\_Sub\_Job\_F, W\_Location\_D, W\_Time\_D, W\_Machine\_Type\_D**

**WHERE W\_Job\_F.Location\_Id = W\_Location\_D.Location\_Id**

**AND W\_Sub\_Job\_F.Machine\_Type\_Id = W\_Machine\_Type\_D.Machine\_Type\_Id**

**AND W\_Time\_D.Time\_Id = Contract\_Date**

**AND W\_Job\_F.Job\_Id = W\_Sub\_Job\_F.Job\_Id**

**GROUP BY W\_Sub\_Job\_F.Job\_Id, W\_Location\_D.LOCATION\_ID,**

**W\_LOCATION\_D.LOCATION\_NAME, W\_TIME\_D.TIME\_YEAR,**

**W\_TIME\_D.TIME\_MONTH;**

**CREATE VIEW LocCostSummary AS**

**SELECT W\_Sub\_Job\_F.Job\_Id,**

**W\_Location\_D.LOCATION\_ID ,W\_LOCATION\_D.LOCATION\_NAME,**

**W\_TIME\_D.TIME\_YEAR, W\_TIME\_D.TIME\_MONTH,**

**SUM(Cost\_Labor) AS SumLaborCosts,**

**SUM(Cost\_Material) AS SumMaterialCosts,**

**SUM(Cost\_Overhead) AS SumOvrhdCosts,**

**SUM(Machine\_Hours \* Rate\_Per\_Hour) AS SumMachineCosts,**

**SUM(Quantity\_Produced) AS SumQtyProduced,**

**SUM(Cost\_Labor + Cost\_Material + Cost\_Overhead +**

**(Machine\_Hours \* Rate\_Per\_Hour) ) AS TotalCosts,**

**SUM( Cost\_Labor + Cost\_Material + Cost\_Overhead + (Machine\_Hours \***

**Rate\_Per\_Hour) ) / SUM(Quantity\_Produced) AS UnitCosts**

**FROM W\_Job\_F, W\_Sub\_Job\_F, W\_Location\_D, W\_Time\_D, W\_Machine\_Type\_D**

**WHERE W\_Job\_F.Location\_Id = W\_Location\_D.Location\_Id**

**AND W\_Sub\_Job\_F.Machine\_Type\_Id = W\_Machine\_Type\_D.Machine\_Type\_Id**

**AND W\_Time\_D.Time\_Id = Contract\_Date**

**AND W\_Job\_F.Job\_Id = W\_Sub\_Job\_F.Job\_Id**

**GROUP BY W\_Sub\_Job\_F.Job\_Id, W\_Location\_D.LOCATION\_ID,**

**W\_LOCATION\_D.LOCATION\_NAME, W\_TIME\_D.TIME\_YEAR,**

**W\_TIME\_D.TIME\_MONTH;**

**Base Query 4**

BQ4 involves 1 fact table (invoice line) and 3 dimension tables (location, sales class, and time). In the following SELECT statement, the join operator style is used with 3 join operations in the FROM clause. Alternatively, the cross product style could be used with 4 tables in the FROM clause and 3 join conditions in the WHERE clause. The GROUP BY clause must contain all non-aggregate columns (Location\_Id, Location\_Name, Sales\_Class\_Id, Sales\_Class\_Desc, Time\_Year, and Time\_Month). The WHERE clause must contain the condition that the quantity shipped is larger than the quantity invoiced. Note the calculation of return amount in the computed column *SumReturnAmt* involves a calculation of unit price (invoice\_amount / invoice\_quantity).

**-- BQ4 in the quality control area**

**-- Return quantity and amount by location and sales class**

**-- Calculate unit price as invoice\_amount/invoice\_quantity**

**SELECT**

**W\_Location\_D.Location\_Id, Location\_Name,**

**W\_Sales\_Class\_D.Sales\_Class\_Id, Sales\_Class\_Desc,**

**Time\_Year, Time\_Month,**

**SUM ( quantity\_shipped - invoice\_quantity ) as SumReturnQty,**

**SUM ( (quantity\_shipped - invoice\_quantity) \***

**(invoice\_amount/invoice\_quantity) ) AS SumReturnAmt**

**FROM W\_INVOICELINE\_F INNER JOIN W\_TIME\_D**

**ON W\_INVOICELINE\_F.INVOICE\_SENT\_DATE = W\_TIME\_D.TIME\_ID**

**INNER JOIN W\_Location\_D**

**ON W\_INVOICELINE\_F.Location\_Id = W\_Location\_D.Location\_Id**

**INNER JOIN W\_Sales\_Class\_D**

**ON W\_INVOICELINE\_F.Sales\_Class\_Id = W\_Sales\_Class\_D.Sales\_Class\_Id**

**WHERE quantity\_shipped > invoice\_quantity**

**GROUP BY W\_Location\_D.Location\_Id, Location\_Name,**

**W\_Sales\_Class\_D.Sales\_Class\_Id, Sales\_Class\_Desc, Time\_Year, Time\_Month;**

**Base Query 5**

BQ5 involves a nested query in the FROM clause as shown in the assignment on page 4. The outer query contains 3 base tables (job fact table along with location and sales class dimension tables) and a nested query in the FROM clause. The WHERE clause contains 2 join conditions for the tables in the outer query, a join condition with the nested query, and a condition comparing the date promised to the last shipment date. The SELECT clause in the outer query should use the *GetBusDaysDiff* function to calculate the difference in business days. The outer query should not contain a GROUP BY clause. To facilitate formulation of analytic queries, the base query should be placed in a CREATE VIEW statement.

-- **BQ5 in the quality control area**

**-- Jobs with delays in the last shipment date (Date\_Promised)**

**-- Nested query in the FROM clause to determine last shipment date**

**SELECT W\_JOB\_F.job\_ID,**

**W\_JOB\_F.SALES\_CLASS\_ID, Sales\_Class\_Desc,**

**W\_JOB\_F.LOCATION\_ID, Location\_Name,**

**Date\_Promised, Last\_Shipment\_Date,**

**QUANTITY\_ORDERED, SumDelayShipQty,**

**GetBusDaysDiff ( date\_promised, Last\_Shipment\_Date ) AS BusDaysDiff**

**FROM W\_JOB\_F , W\_Location\_D, W\_Sales\_Class\_D,**

**(SELECT W\_SUB\_JOB\_F.JOB\_ID,**

**MAX(actual\_ship\_Date) AS Last\_Shipment\_Date,**

**SUM ( actual\_Quantity ) AS SumDelayShipQty**

**FROM W\_JOB\_SHIPMENT\_F, W\_SUB\_JOB\_F, W\_Job\_F**

**WHERE W\_SUB\_JOB\_F.SUB\_JOB\_ID = W\_JOB\_SHIPMENT\_F.SUB\_JOB\_ID**

**AND W\_Job\_F.Job\_Id = W\_SUB\_JOB\_F.JOB\_ID**

**AND Actual\_Ship\_Date > Date\_Promised**

**GROUP BY W\_SUB\_JOB\_F.JOB\_ID**

**) X1**

**WHERE date\_promised < X1.Last\_Shipment\_Date**

**AND W\_JOB\_F.JOB\_ID = X1.Job\_Id**

**AND W\_Job\_F.Location\_Id = W\_Location\_D.Location\_Id**

**AND W\_Job\_F.Sales\_Class\_Id = W\_Sales\_Class\_D.Sales\_Class\_Id;**

**-- CREATE VIEW statement using the base query**

**CREATE VIEW LastShipmentDelays AS**

**SELECT W\_JOB\_F.job\_ID ,**

**W\_JOB\_F.SALES\_CLASS\_ID, Sales\_Class\_Desc,**

**W\_JOB\_F.LOCATION\_ID, Location\_Name,**

**Date\_Promised, Last\_Shipment\_Date,**

**QUANTITY\_ORDERED, SumDelayShipQty,**

**GetBusDaysDiff ( date\_promised, Last\_Shipment\_Date ) AS BusDaysDiff**

**FROM W\_JOB\_F , W\_Location\_D, W\_Sales\_Class\_D,**

**(SELECT W\_SUB\_JOB\_F.JOB\_ID,**

**MAX(actual\_ship\_Date) AS Last\_Shipment\_Date,**

**SUM ( actual\_Quantity ) AS SumDelayShipQty**

**FROM W\_JOB\_SHIPMENT\_F, W\_SUB\_JOB\_F, W\_Job\_F**

**WHERE W\_SUB\_JOB\_F.SUB\_JOB\_ID = W\_JOB\_SHIPMENT\_F.SUB\_JOB\_ID**

**AND W\_Job\_F.Job\_Id = W\_SUB\_JOB\_F.JOB\_ID**

**AND Actual\_Ship\_Date > Date\_Promised**

**GROUP BY W\_SUB\_JOB\_F.JOB\_ID**

**) X1**

**WHERE date\_promised < X1.Last\_Shipment\_Date**

**AND W\_JOB\_F.JOB\_ID = X1.Job\_Id**

**AND W\_Job\_F.Location\_Id = W\_Location\_D.Location\_Id**

**AND W\_Job\_F.Sales\_Class\_Id = W\_Sales\_Class\_D.Sales\_Class\_Id;**

**Base Query 6**

**BQ6 involves a nested query in the FROM clause as shown in the assignment on page 4. The outer query contains 3 base tables (job fact table and location and sales class dimension tables) and a nested query in the FROM clause. The WHERE clause contains 2 join conditions**

**for the tables in the outer query, a join condition with the nested query, and a condition comparing the shipped by date to the first shipment date. The SELECT clause in the outer query should use the *GetBusDaysDiff* function to calculate the difference in business days. The outer query should not contain a GROUP BY clause. To facilitate formulation of analytic queries, the base query should be placed in a CREATE VIEW statement.**

**-- BQ6 in the quality control area**

**-- Jobs with delays in the first shipment date (Date\_Ship\_By)**

**-- Requires a nested query in the FROM clause to determine first shipment** date

**SELECT W\_JOB\_F.job\_ID,**

**W\_JOB\_F.SALES\_CLASS\_ID, Sales\_Class\_Desc,**

**W\_JOB\_F.LOCATION\_ID, Location\_Name,**

**Date\_Ship\_By,**

**FirstShipDate,**

**GetBusDaysDiff ( date\_ship\_By, FirstShipDate ) AS BusDaysDiff**

**FROM W\_JOB\_F, W\_Location\_D, W\_Sales\_Class\_D,**

**(SELECT W\_SUB\_JOB\_F.JOB\_ID, MIN(actual\_ship\_Date) as FirstShipDate**

**FROM W\_JOB\_SHIPMENT\_F, W\_SUB\_JOB\_F**

**WHERE W\_SUB\_JOB\_F.SUB\_JOB\_ID = W\_JOB\_SHIPMENT\_F.SUB\_JOB\_ID**

**GROUP BY W\_SUB\_JOB\_F.JOB\_ID**

**) X1**

**WHERE date\_ship\_By < X1.FirstShipDate**

**AND W\_JOB\_F.JOB\_ID = X1.Job\_Id**

**AND W\_Job\_F.Location\_Id = W\_Location\_D.Location\_Id**

**AND W\_Job\_F.Sales\_Class\_Id = W\_Sales\_Class\_D.Sales\_Class\_Id;**

**-- CREATE VIEW statement using the base query**

**CREATE VIEW FirstShipmentDelays AS**

**SELECT W\_JOB\_F.job\_ID,**

**W\_JOB\_F.SALES\_CLASS\_ID, Sales\_Class\_Desc,**

**W\_JOB\_F.LOCATION\_ID, Location\_Name,**

**Date\_Ship\_By,**

**FirstShipDate,**

**GetBusDaysDiff ( date\_ship\_By, FirstShipDate ) AS BusDaysDiff**

**FROM W\_JOB\_F , W\_Location\_D, W\_Sales\_Class\_D,**

**(SELECT W\_SUB\_JOB\_F.JOB\_ID, MIN(actual\_ship\_Date) as FirstShipDate**

**FROM W\_JOB\_SHIPMENT\_F, W\_SUB\_JOB\_F**

**WHERE W\_SUB\_JOB\_F.SUB\_JOB\_ID = W\_JOB\_SHIPMENT\_F.SUB\_JOB\_ID**

**GROUP BY W\_SUB\_JOB\_F.JOB\_ID**

**) X1**

**WHERE date\_ship\_By < X1.FirstShipDate AND W\_JOB\_F.JOB\_ID = X1.Job\_Id**

**AND W\_Job\_F.Location\_Id = W\_Location\_D.Location\_Id**

**AND W\_Job\_F.Sales\_Class\_Id = W\_Sales\_Class\_D.Sales\_Class\_Id;**