BQ1
- Tables: job, location, time, sales class
- Aggregate functions: sum of qty ordered and job amount
- Grouping: W_Location_D.Location_Id, Location_Name,
  W_Sales_Class_D.Sales_Class_Id, Sales_Class_Desc,
  Base_Price, Time_Year, Time_Month;

BQ2

- Tables: subjob, job, location, time, invoice, shipment
- Aggregate functions: sum of invoice quantity and amount
- Grouping: 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
- WITH statement containing CTE and query

```
        -- Need to join job and invoice through subjob and shipment (-2)

WITH 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 )
SELECT * FROM LocRevenueSummary;
```

BQ3
- Tables: subjob, job, location, time, machine type
- Aggregate functions: sum of labor cost, material cost, overhead,
  machine hours * rate per hour, total cost, quantity produced
- Grouping: 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
- WITH statement containing CTE and query

BQ4
- Tables: invoice line, time, location, sales class
- Aggregate functions: sum of return qty and return amount
- Condition on quantity_shipped > invoice_quantity
- Grouping: W_Location_D.Location_Id, Location_Name,
  W_Sales_Class_D.Sales_Class_Id, Sales_Class_Desc, Time_Year, Time_Month
- WITH statement not required

BQ5
- Tables: job, location, sales class, nested query (X1)
- Aggregate functions: none in outer query
- Columns: last shipment date, GetBusDaysDiff function, SumDelayShipQty
- Join Conditions: W_JOB_F.JOB_ID = X1.Job_Id, W_Job_F.Location_Id =
W_Location_D.Location_Id,
   W_Job_F.Sales_Class_Id = W_Sales_Class_D.Sales_Class_Id
- Grouping: none in outer query
- WITH statement containing CTE and query; can use more than one CTE

```
        -- missing join of CTEs in the final SELECT statement -1.5

WITH MaxShipDates AS
( 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 )

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 (Last_Shipment_Date, date_promised ) AS BusDaysDiff
FROM W_JOB_F , W_Location_D, W_Sales_Class_D, MaxShipDates X1
WHERE 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;
```

BQ6
- Tables: job, location, sales class, nested query (X1)
- Aggregate functions: none in outer query
- Columns: first shipment date, GetBusDaysDiff function, date ship by
- Grouping: none in outer query

- Conditions: date_ship_By < X1.FirstShipDate, W_JOB_F.JOB_ID = X1.Job_Id
- WITH statement containing CTE and query; can use more than one CTE

Analytic queries for 60% (5 points each)

AQ1
- Tables: W_JOB_F, W_Location_D, W_TIME_D
- Columns: SUM (QUANTITY_ORDERED * Unit_Price) AS SumJobAmt,
  SUM (SUM (QUANTITY_ORDERED * Unit_Price))
    OVER (PARTITION BY Location_Name, Time_Year
      ORDER BY Time_Month
      ROWS UNBOUNDED PRECEDING) AS CumSumAmt
- Grouping: Location_Name, Time_Year, Time_Month

AQ2
- Tables: W_JOB_F, W_Location_D, W_TIME_D
- Columns: AVG(QUANTITY_ORDERED * Unit_Price) AS AvgJobAmount ,
  AVG(AVG(QUANTITY_ORDERED * Unit_Price))
   OVER (PARTITION BY Location_Name
   ORDER BY Time_Year, Time_Month
   ROWS BETWEEN 11 PRECEDING AND CURRENT ROW)
Grouping: Location_Name, Time_Year, Time_Month

AQ3
- Use CTEs in FROM: LocCostSummary X1, LocRevenueSummary X2
- Columns: X1.Location_Name, X1.Time_Year, SUM(SumInvoiceAmt - TotalCosts) AS SumLocProfit
- Join condition: X1.Job_Id = X2.Job_Id
- Grouping: X1.Location_Name, X1.Time_Year

      RANK() OVER (PARTITION BY X1.Time_Year
       ORDER BY (SUM(SumInvoiceAmt - TotalCosts)) DESC) AS RankProfitSum

AQ4
- Use CTEs in FROM: LocCostSummary X1, LocRevenueSummary X2
- Columns: X1.Location_Name, X1.Time_Year,
      SUM (SumInvoiceAmt - TotalCosts) / SUM(SumInvoiceAmt) AS ProfitMargin
- Join condition: X1.Job_Id = X2.Job_Id
- Grouping: X1.Location_Name, X1.Time_Year

      RANK() OVER (PARTITION BY X1.Time_Year
       ORDER BY (SUM (SumInvoiceAmt - TotalCosts) / SUM(SumInvoiceAmt)) DESC) AS RankProfitMargin

AQ5
- Use CTEs in FROM: LocCostSummary X1, LocRevenueSummary X2
- Columns: X1.Job_Id, X1.Location_Name, X1.Time_Year, X1.Time_Year,

(SumInvoiceAmt - TotalCosts) / SumInvoiceAmt AS ProfitMargin
- Join condition: X1.Job_Id = X2.Job_Id
- Grouping: none

PERCENT_RANK() OVER (
ORDER BY ((SumInvoiceAmt - TotalCosts) / SumInvoiceAmt))
AS PercentRankProfitMargin

AQ6
- Tables: Nested query in FROM clause or CTE
- Can use 1 to 3 CTEs

```
-- Nested query in FROM
SELECT Job_Id, Location_Name, Time_Year, Time_Month,
      ProfitMargin, PercentRankProfitMargin
 FROM (
  SELECT X1.Job_Id, X1.Location_Name, X1.Time_Year, X1.Time_Month,
        (SumInvoiceAmt - TotalCosts) / SumInvoiceAmt AS ProfitMargin,
        PERCENT_RANK() OVER (
        ORDER BY ( (SumInvoiceAmt - TotalCosts) / SumInvoiceAmt ) )
         AS PercentRankProfitMargin
  FROM LocCostSummary X1, LocRevenueSummary X2
  WHERE X1.Job_Id = X2.Job_Id )
 WHERE PercentRankProfitMargin > 0.95;

-- CTE instead of nested query
-- See solution with 2 other CTEs
WITH ...

PercentRankCTE AS (
  SELECT X1.Job_Id, X1.Location_Name, X1.Time_Year, X1.Time_Month,
        (SumInvoiceAmt - TotalCosts) / SumInvoiceAmt AS ProfitMargin,
        PERCENT_RANK() OVER (
        ORDER BY ( (SumInvoiceAmt - TotalCosts) / SumInvoiceAmt ) )
         AS PercentRankProfitMargin
  FROM LocCostSummary X1, LocRevenueSummary X2
  WHERE X1.Job_Id = X2.Job_Id )

SELECT Job_Id, Location_Name, Time_Year, Time_Month,
      ProfitMargin, PercentRankProfitMargin
 FROM PercentRankCTE X
WHERE X.PercentRankProfitMargin > 0.95;
```

- Conditions: PercentRankProfitMargin > 0.95

AQ7
- Tables: invoice fact, time, sales class

- Columns: Sales_Class_Desc, Time_Year,
  SUM (quantity_shipped - invoice_quantity) as ReturnSum
- Conditions: quantity_shipped > invoice_quantity
- Grouping: Sales_Class_Desc, Time_Year
- Analytic functions

  RANK() over (PARTITION BY Time_Year
    ORDER BY SUM (quantity_shipped - invoice_quantity) DESC)

AQ8
- Tables: invoice fact, time, sales class
- Columns: Time_Year, Sales_Class_Desc,
  SUM (quantity_shipped - invoice_quantity) as SumReturnQty
- Conditions: quantity_shipped > invoice_quantity
- Grouping: Sales_Class_Desc, Time_Year
- Analytic functions

  SUM (quantity_shipped - invoice_quantity) /
  SUM(SUM (quantity_shipped - invoice_quantity))
    OVER (PARTITION BY Time_Year) AS RatioReturnSum

```
SELECT Time_Year, Sales_Class_Desc,
  SUM ( quantity_shipped - invoice_quantity ) as SumReturnQty,
  SUM ( quantity_shipped - invoice_quantity ) /
  SUM( SUM ( quantity_shipped - invoice_quantity ) )
    OVER ( PARTITION BY Time_Year ) AS RatioReturnSum
 FROM W_INVOICELINE_F INNER JOIN W_TIME_D
      ON W_INVOICELINE_F.INVOICE_SENT_DATE = W_TIME_D.TIME_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 Sales_Class_Desc, Time_Year
 ORDER BY Time_Year, SUM( quantity_shipped - invoice_quantity );
```

AQ9
- Tables: FirstShipmentDelays (CTE), W_Time_D
- Can use 1 to 2 CTEs
- Columns: Location_Name, W_Time_D.Time_Year, SUM(BusDaysDiff) as SumDelayDays
- Grouping: Location_Name, W_Time_D.Time_Year
- Analytic functions:

  RANK() OVER (PARTITION BY W_Time_D.Time_Year
    ORDER BY SUM(BusDaysDiff) DESC) AS RankSumDelayDays,
  DENSE_RANK() OVER (PARTITION BY W_Time_D.Time_Year
    ORDER BY SUM(BusDaysDiff) DESC) AS RankSumDelayDays

AQ10
- Tables: LastShipmentDelays (CTE), W_Time_D
- Can use 1 to 2 CTEs
- Columns: Location_Name, W_Time_D.Time_Year, COUNT(*) AS NumJobs
      SUM(BusDaysDiff) as SumDelayDays,
      SUM(Quantity_Ordered - SumDelayShipQty) / SUM(Quantity_Ordered)
- Grouping: Location_Name, W_Time_D.Time_Year
- Analytic functions:

```
WITH MaxShipDates AS
( 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 ),
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 (Last_Shipment_Date, date_promised ) AS BusDaysDiff
FROM W_JOB_F , W_Location_D, W_Sales_Class_D, MaxShipDates X1
WHERE 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
)

SELECT Location_Name, W_Time_D.Time_Year,
      COUNT(*) AS NumJobs,
      SUM(BusDaysDiff) as SumDelayDays,
      SUM(Quantity_Ordered - SumDelayShipQty) / SUM(Quantity_Ordered)
       AS PromisedDelayRate,
  RANK() OVER ( PARTITION BY W_Time_D.Time_Year
    ORDER BY SUM(Quantity_Ordered - SumDelayShipQty) /
          SUM(Quantity_Ordered) DESC) AS RankDelayRate
 FROM LastShipmentDelays, W_Time_D
 WHERE W_Time_D.Time_Id = LastShipmentDelays.Date_Promised
 GROUP BY Location_Name, W_Time_D.Time_Year;
```