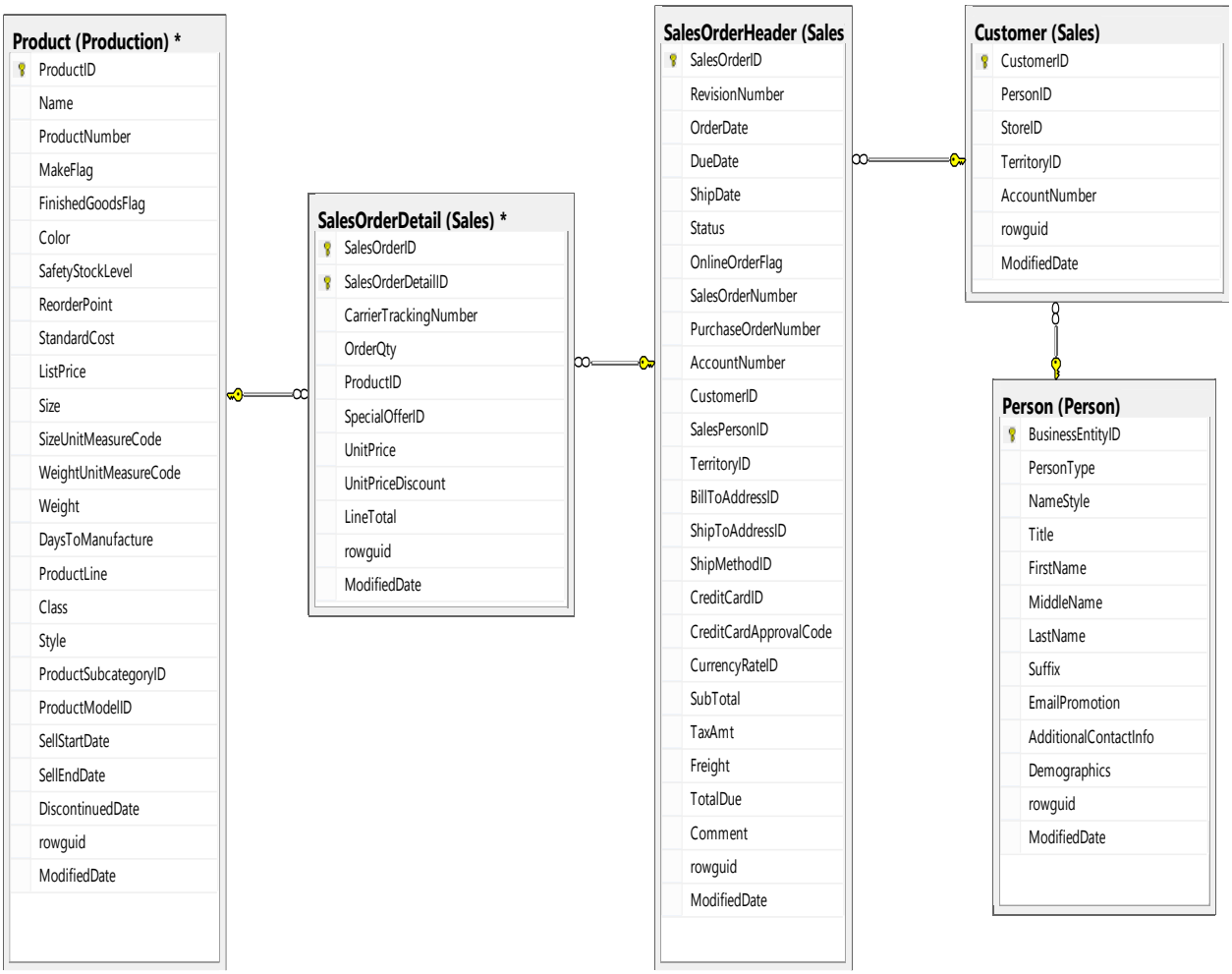
Lab 3 Exercises

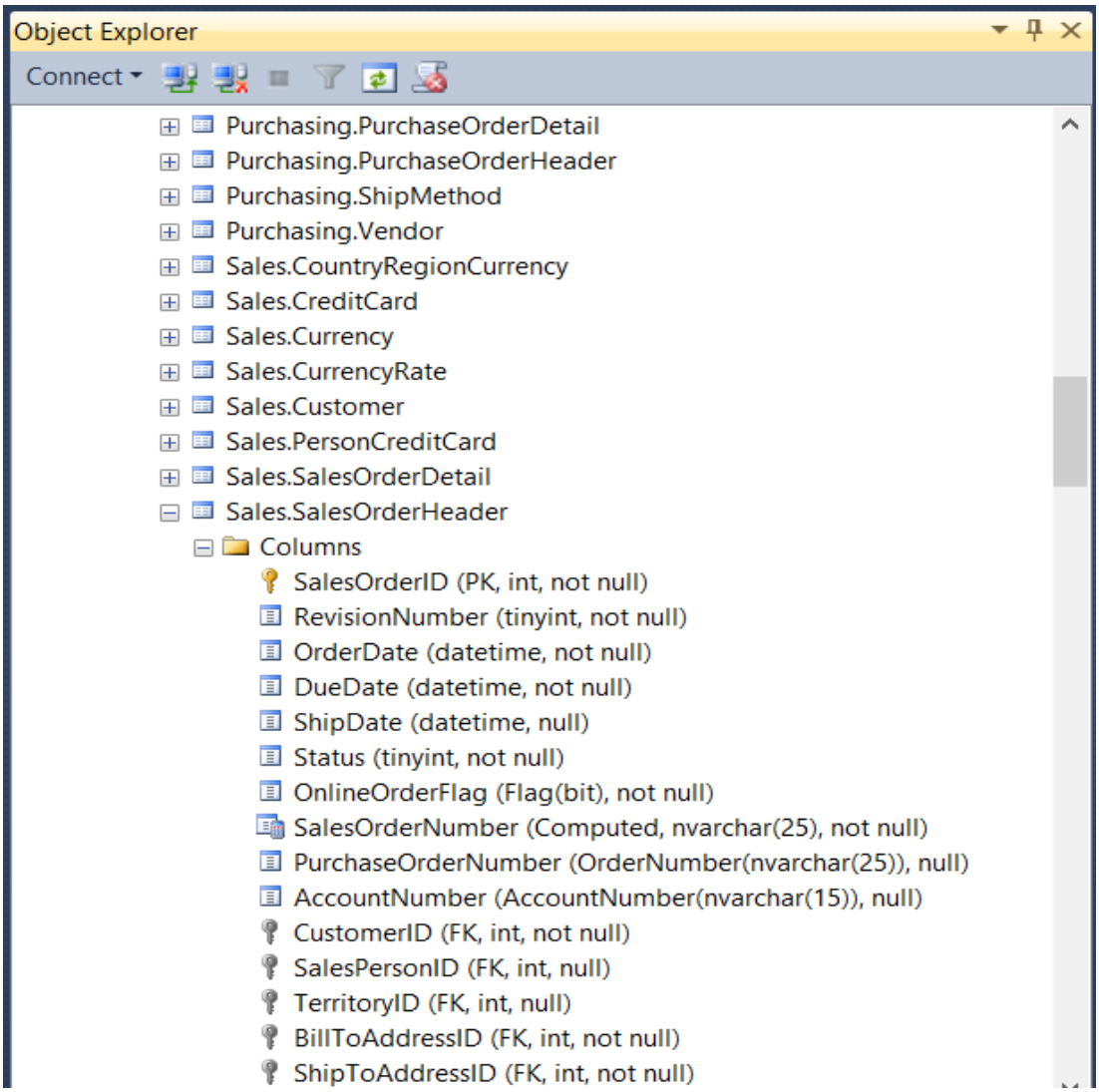
These exercise questions are for self-practice. No submission is needed.

Notes: The following partial ERD for AdventureWorks2008R2 was generated by SQL Server Management Studio. Use it to locate data when writing SQL queries.



Notes: If we don't have an ERD, we can also use the Object Explorer in SQL Server Management Studio to locate data by following the steps listed below.

1) Under Object Explorer in SQL Server Management Studio, expand Databases
2) Expand the database we want to work with, such as AdventureWorks2008R2
3) Expand Tables
4) Expand the table we want to work with, such as Sales.Customer
5) Expand Columns
6) Then we'll see all columns contained in a table



```
USE AdventureWorks2008R2;
```

Exercise 1

```
/*
   SELECT ProductID, Name, ListPrice FROM Production.Product.
   Use the CASE function to display "Expensive" if ListPrice > 3000
                                    "Medium" if ListPrice > 1000
                                    "Low" if ListPrice <= 1000.
   ORDER the results DESC BY ListPrice.
*/
```

Exercise 2

```
/*
   SELECT SalesOrderID, CustomerID, TotalDue
   FROM Sales.SalesOrderHeader
   WHERE TotalDue > 10000
   and RANK them with gaps in the desc order of TotalDue
*/
```

Exercise 3

```
/*
   SELECT SalesOrderID, CustomerID, TotalDue
   FROM Sales.SalesOrderHeader
   WHERE TotalDue > 10000
   and RANK them with gaps in the desc order of TotalDue
   also PARTITION BY CustomerID

*/
```

Exercise 4

```
/*
   SELECT SalesOrderID, CustomerID, TotalDue
   FROM Sales.SalesOrderHeader
   WHERE TotalDue > 10000
   and RANK them with gaps in the desc order of TotalDue
   also PARTITION BY CustomerID

   Display only the highest total due amount for each customer.

   Hints: For this exercise, we need to create a derived table
          using a subquery. Then SELECT FROM the derived table.
*/
```

Exercise 5

```
/* List the product id, product name, and order date of each
   product sold in 2008.

   Tables needed: Production.Product
                  Sales.SalesOrderDetail
                  Sales.SalesOrderHeader
*/
```

Exercise 6

```
/* What is the name and average rating for the product with
   ProductID = 937? */
```

Exercise 7

```
/* Use the SubTotal value in SalesOrderHeader to calculate
   total value. What is the total value of products sold to
   an address in 'Seattle'? */
```

```
/* Some report formatting examples */


/* Use system functions to format dates and make reports
   look better. */


/*  CONVERT is NOT ANSI standard. */
-- Display date portion of the system date and time.
SELECT CONVERT(CHAR(20), GETDATE(), 1) AS [Current Date];

/*  CAST is ANSI standard. */
-- Display date portion of the system date and time.
SELECT CAST(GETDATE() AS CHAR(11)) AS [Current Date];

-- Display both date and time of the system date and time.
SELECT CAST(GETDATE() AS CHAR(20)) AS [Current Date and Time];

-- Display only the time portion of the system date and time.
SELECT RIGHT(CAST(GETDATE() AS CHAR(20)), 8) AS [Current Time];



/* Use system functions to format numbers and make reports
   look better. */


SELECT CAST((ROUND(AVG(OrderQty+ .0), 2)) AS decimal(5,2)) AS [Average
Sales]
FROM Sales.SalesOrderDetail
WHERE UnitPrice BETWEEN 30 AND 50;


SELECT STR(ROUND(AVG(OrderQty + .0), 2), 8, 2) AS [Average Sales]
FROM Sales.SalesOrderDetail
WHERE UnitPrice BETWEEN 30 AND 50;
```

Don’t look at the solution until you have completed an exercise question.

```
-- Solutions


USE AdventureWorks2008R2;


-- Exercise 1 Solution

/*
   SELECT ProductID, Name, ListPrice FROM Production.Product.
   Use the CASE function to display "Expensive" if ListPrice > 3000
                                    "Medium" if ListPrice > 1000
                                    "Low" if ListPrice <= 1000.
   ORDER the results DESC BY ListPrice.
*/


SELECT ProductID, Name, ListPrice,
   CASE
       WHEN LISTPRICE > 3000
            THEN 'Expensive'
       WHEN LISTPRICE <= 1000
            THEN 'Low'
       ELSE 'Medium'
   END AS PriceRange
FROM Production.Product
ORDER BY ListPrice DESC;
```

```sql
-- Exercise 2 Solution

/*
   SELECT SalesOrderID, CustomerID, TotalDue
   FROM Sales.SalesOrderHeader
   WHERE TotalDue > 10000
   and RANK them with gaps in the desc order of TotalDue
*/


SELECT RANK() OVER (ORDER BY TotalDue DESC) as [Rank],
SalesOrderID, CustomerID, Totaldue
FROM Sales.SalesOrderHeader
WHERE TotalDue >10000;




-- Exercise 3 Solution

/*
   SELECT SalesOrderID, CustomerID, TotalDue
   FROM Sales.SalesOrderHeader
   WHERE TotalDue > 10000
   and RANK them with gaps in the desc order of TotalDue
   also PARTITION BY CustomerID

*/


SELECT RANK() OVER (PARTITION BY CustomerID ORDER BY TotalDue DESC) as
[Rank],
SalesOrderID, CustomerID, Totaldue
FROM Sales.SalesOrderHeader
WHERE TotalDue >10000;
```

```
-- Exercise 4 Solution

/*
   SELECT SalesOrderID, CustomerID, TotalDue
   FROM Sales.SalesOrderHeader
   WHERE TotalDue > 10000
   and RANK them with gaps in the desc order of TotalDue
   also PARTITION BY CustomerID

   Display only the highest total due amount for each customer.

*/


SELECT * FROM
   (SELECT RANK() OVER (PARTITION BY CustomerID
           ORDER BY TotalDue DESC) as [Rank],
           SalesOrderID, CustomerID, Totaldue
    FROM Sales.SalesOrderHeader
    WHERE TotalDue >10000) a
WHERE a.[Rank] = 1;


-- For this exercise, we need to create a derived table using a subquery.
-- Then SELECT FROM the derived table.



-- Exercise 5 Solution

/* List the product id, product name, and order date of each
   product sold in 2008.

   Tables needed: Production.Product
                  Sales.SalesOrderDetail
                  Sales.SalesOrderHeader
*/

SELECT DISTINCT p.ProductID, Name, OrderDate
FROM Production.Product p
INNER JOIN Sales.SalesOrderDetail od
ON p.ProductID = od.ProductID
INNER JOIN Sales.SalesOrderHeader oh
ON oh.SalesOrderID = od.SalesOrderID
WHERE DATEPART(YEAR, OrderDate) = 2008;
```

```sql
-- Exercise 6 Solution

/* What is the name and average rating for the product with
   ProductID = 937? */

select pdt.name, avg(pr.rating) as rating
from Production.Product pdt
join Production.ProductReview pr
     on pdt.ProductID = pr.ProductID
where pr.ProductID = 937
group by pdt.Name;



-- Exercise 7 Solution

/* Use the SubTotal value in SalesOrderHeader to calculate
   total value. What is the total value of products sold to
   an address in 'Seattle'? */

select ad.city, sum(SubTotal) as total_value
from Sales.SalesOrderHeader soh
join Person.Address ad on soh.ShipToAddressID = ad.AddressID
where ad.city = 'Seattle'
group by ad.city;
```