SQL Assignment Solution

```
--Exercise-1
--1
/*     Display the number of records in the [SalesPerson] table.
(Schema(s) involved: Sales)
*/
SELECT COUNT(*) AS 'Number of Records'
FROM Sales.SalesPerson;
```

where Title is one of Design Engine

98 %

Results | Messages

	Number of Records
1	17

```
--2
/*     Select both the FirstName and LastName of records from
the Person table where the FirstName begins with the letter 'B'
(Schema(s) involved: Person)
*/
SELECT FirstName
       , LastName
FROM Person.Person
WHERE FirstName LIKE 'B%'
```

Results | Messages

	FirstName	LastName
1	Bailey	ME
2	Bailey	ME
3	Bailey	ME
4	Bailey	ME
5	Bailey	ME
6	Bailey	ME
7	Bailey	ME
8	Bailey	ME
9	Bailey	ME
10	Bailey	ME
11	Bailey	ME
12	Bailey	ME
13	Bailey	ME
14	Bailey	ME
15	Bailey	ME
16	Bailey	ME
17	Bailey	ME
18	Bailey	ME
19	Bailey	ME
20	Bailey	ME
21	Bailey	ME

```sql
--3
/*     Select a list of FirstName and LastName for employees
where Title is one of Design Engineer, Tool Designer
or Marketing Assistant. (Schema(s) involved: HumanResources, Person)
*/
SELECT p.FirstName
       , p.LastName
FROM Person.Person AS p
INNER JOIN HumanResources.Employee AS e
ON p.BusinessEntityID = e.BusinessEntityID
WHERE e.JobTitle IN ('Design Engineer', 'Tool Designer', 'Marketing Assistant');
```

98 %

Results | Messages

	FirstName	LastName
1	Gail	ME
2	Jossef	ME
3	Thierry	ME
4	Janice	ME
5	Sharon	ME
6	Kevin	ME
7	Mary	ME
8	Wanida	ME

```sql
--4
/*     Display the Name and Color of the Product with the
maximum weight. (Schema(s) involved: Production)
*/
DECLARE @MaxWeight int =
       (SELECT MAX(Weight) FROM Production.Product);
SELECT Name
       , Color
       , Weight
FROM Production.product
WHERE weight = @MaxWeight;
```

98 %

Results | Messages

	Name	Color	Weight
1	LL Road Rear Wheel	Black	1050.00

```sql
--5
/*     Display Description and MaxQty fields from the SpecialOffer table.
Some of the MaxQty values are NULL, in this case display
the value 0.00 instead. (Schema(s) involved: Sales)
*/


SELECT COALESCE(CAST(MaxQty AS VARCHAR),'0.00') AS 'Weight'
       , Description
FROM Sales.SpecialOffer
```

98 %

Results | Messages

	Weight	Description
1	0.00	No Discount
2	14	Volume Discount 11 to 14
3	24	Volume Discount 15 to 24
4	40	Volume Discount 25 to 40
5	60	Volume Discount 41 to 60
6	0.00	Volume Discount over 60
7	0.00	Mountain-100 Clearance Sale
8	0.00	Sport Helmet Discount-2002
9	0.00	Road-650 Overstock
10	0.00	Mountain Tire Sale
11	0.00	Sport Helmet Discount-2003
12	0.00	LL Road Frame Sale
13	0.00	Touring-3000 Promotion
14	0.00	Touring-1000 Promotion
15	0.00	Half-Price Pedal Sale
16	0.00	Mountain-500 Silver Clearan...

```sql
--6
/*     Display the overall Average of the [CurrencyRate].[AverageRate]
values for the exchange rate ‘USD’ to ‘GBP’ for the year 2005
i.e. FromCurrencyCode = ‘USD’ and ToCurrencyCode = ‘GBP’.
Note: The field [CurrencyRate].[AverageRate] is defined as
'Average exchange rate for the day.' (Schema(s) involved: Sales)
*/
SELECT AVG(AverageRate) AS 'Average exchange rate for the day'
FROM Sales.CurrencyRate
WHERE datepart(year,CurrencyRateDate)=2005
       AND FromCurrencyCode='USD'
       AND ToCurrencyCode='GBP';
```

98 %

Results | Messages

	Average exchange rate for the day
1	0.6261

```sql
--7
/*     Display the FirstName and LastName of records from the
Person table where FirstName contains the letters ‘ss’.
Display an additional column with sequential numbers for each
row returned beginning at integer 1. (Schema(s) involved: Person)
*/

SELECT ROW_NUMBER() OVER(ORDER BY FirstName) AS 'Row Number'
       , FirstName
       , LastName
FROM Person.Person
WHERE FirstName like '%ss%'
```

98 %

Results | Messages

	Row Number	FirstName	LastName
1	1	Alyssa	ME
2	2	Alyssa	ME
3	3	Alyssa	ME
4	4	Alyssa	ME
5	5	Alyssa	ME
6	6	Alyssa	ME
7	7	Alyssa	ME
8	8	Alyssa	ME
9	9	Alyssa	ME
10	10	Alyssa	ME
11	11	Alyssa	ME
12	12	Alyssa	ME
13	13	Alyssa	ME
14	14	Alyssa	ME
15	15	Alyssa	ME
16	16	Alyssa	ME
17	17	Alyssa	ME

Query executed successfully.

Solution - 7

98 %

Results | Messages

	SalesPersonId	Commison Band
1	274	band 0
2	285	band 0
3	287	band 0
4	278	band 1
5	279	band 1
6	280	band 1
7	281	band 1
8	275	band 2
9	283	band 2
10	276	band 2
11	277	band 2
12	282	band 2
13	290	band 3
14	288	band 3
15	286	band 3
16	284	band 3
17	289	band 3

Query executed successfully.

Solution - 8

```sql
--8
/*     Sales people receive various commission rates that
belong to 1 of 4 bands. (Schema(s) involved: Sales)
CommissionPct Commission Band
0.00                  Band 0
Up To 1%              Band 1
Up To 1.5%            Band 2
Greater 1.5%  Band 3
Display the [SalesPersonID] with an additional
column entitled ‘Commission Band’ indicating the appropriate band as above.
*/
SELECT BusinessEntityID As 'SalesPersonId'
       , CASE
       WHEN CommissionPct = 0 THEN 'band 0'
       WHEN CommissionPct > 0 and CommissionPct <= 0.01 THEN 'band 1'
       WHEN CommissionPct > 0.01 and CommissionPct <= 0.015 THEN 'band 2'
       WHEN CommissionPct > 0.015 THEN 'band 3'
       END AS 'Commison Band'
FROM Sales.SalesPerson
ORDER BY CommissionPct
```

```sql
--9
/*     Display the managerial hierarchy from
Ruth Ellerbrock (person type – EM) up to CEO Ken Sanchez.
Hint: use [uspGetEmployeeManagers]
(Schema(s) involved: [Person], [HumanResources])
*/
DECLARE @RuthEllerbrockID int =
       (
       SELECT BusinessEntityID
       FROM Person.Person
       WHERE PersonType = 'EM'
              AND FirstName = 'Ruth'
              AND LastName = 'Ellerbrock'
       );

EXEC dbo.uspGetEmployeeManagers @RuthEllerbrockID;
GO
```

98 %

Results | Messages

RecursionLevel	BusinessEntityID	FirstName	LastName	OrganizationNode	ManagerFirstName	ManagerLastName

```sql
--10
/*     Display the ProductId of the product with the largest stock level.
Hint: Use the Scalar-valued function [dbo].
[UfnGetStock]. (Schema(s) involved: Production)
*/
DECLARE @MaxStock INT =
(
       SELECT MAX(dbo.ufnGetStock(ProductID))
       FROM Production.Product
);
SELECT ProductID
FROM Production.Product
WHERE dbo.ufnGetStock(ProductID) = @MaxStock;
```

98 %

Results | Messages

	ProductID
1	528

```sql
--Exercise-2
/*     Write separate queries using a join,
a subquery, a CTE, and then an EXISTS to list all
AdventureWorks customers who have not placed an order.
*/
-- Using JOIN
SELECT c.CustomerID
FROM Sales.Customer AS c
WHERE c.CustomerID NOT IN(
SELECT c.CustomerID
FROM Sales.Customer AS c
INNER JOIN Sales.SalesOrderHeader AS soh
ON c.CustomerID = soh.CustomerID);

-- Using Subquery
SELECT CustomerID
FROM Sales.Customer
WHERE CustomerID NOT IN
       (SELECT CustomerID
       FROM Sales.SalesOrderHeader);

-- Using CTE
WITH CustomersWithOrders (CustomerID)
AS
       (
       SELECT CustomerID
       FROM Sales.SalesOrderHeader
       )

SELECT CustomerID
FROM Sales.Customer AS c
WHERE CustomerID NOT IN (SELECT * FROM CustomersWithOrders);

-- Using EXISTS
SELECT CustomerID
FROM Sales.Customer AS c
WHERE NOT EXISTS
(
       SELECT CustomerID
       FROM Sales.SalesOrderHeader AS soh
       WHERE c.CustomerID = soh.CustomerID
);
```

98 %

Results | Messages

	CustomerID
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16
17	17
18	18
19	19
20	20
21	21

Query executed successfully.

```sql
--Exercise-3
/*     Show the most recent five orders that were purchased from account
numbers that have spent more than $70,000 with AdventureWorks.
*/
SELECT TOP 5 SalesOrderID
                 , AccountNumber
                 , OrderDate
                 , TotalDue
FROM Sales.SalesOrderHeader
WHERE AccountNumber IN (SELECT AccountNumber
       FROM Sales.SalesOrderHeader
       WHERE TotalDue>70000)
ORDER BY OrderDate DESC;
GO
```

98 %

Results | Messages

	SalesOrderID	AccountNumber	OrderDate	TotalDue
1	71783	10-4020-000024	2008-06-01 00:00:00.000	74488.5028
2	71784	10-4020-000448	2008-06-01 00:00:00.000	101268.2686
3	71792	10-4020-000155	2008-06-01 00:00:00.000	57368.8253
4	71794	10-4020-000678	2008-06-01 00:00:00.000	73045.0853
5	71797	10-4020-000142	2008-06-01 00:00:00.000	73316.5808

```sql
--Exercise-4
/*     Create a function that takes as inputs a SalesOrderID, a Currency Code,
and a date, and returns a table of all the SalesOrderDetail
rows for that Sales Order including Quantity, ProductID,
UnitPrice, and the unit price converted to the target currency based on
the end of day rate for the date provided.
Exchange rates can be found in the Sales.CurrencyRate table. (Use AdventureWorks)
*/
CREATE FUNCTION dbo.ufOrderDetails(@SalesOrderID int, @CurrencyCode nchar(3), @Date
datetime)
RETURNS @Result TABLE (SalesOrderID int, QrderQty int,ProductID int, UnitPrice
money,TargetCurrencyPrice money)
AS
BEGIN
       DECLARE @ConversionRate money =
       (
              SELECT EndOfDayRate
              FROM Sales.CurrencyRate
              WHERE ToCurrencyCode = @CurrencyCode
                     AND CurrencyRateDate = @Date
       )

       INSERT INTO @Result
       SELECT SalesOrderID
              , OrderQty
               , ProductID
               , UnitPrice
               , UnitPrice * @ConversionRate AS 'TargetCurrencyPrice'
       FROM Sales.SalesOrderDetail
       WHERE SalesOrderID = @SalesOrderID
       RETURN;
END;
GO
```

```sql
-- For Testing Function
DECLARE @SalesOrderID int = 54199;
DECLARE @CurrencyCode nchar(3) = 'AUD'
DECLARE @Date datetime = '2005-07-01 00:00:00.000';
SELECT * FROM ufOrderDetails(@SalesOrderID, @CurrencyCode, @Date);
GO
```

98 %

Results | Messages

	SalesOrderID	QrderQty	ProductID	UnitPrice	TargetCurrencyPrice
1	54199	1	921	4.99	7.7345
2	54199	1	873	2.29	3.5495

```sql
--Exercise-5
/*     Write a Procedure supplying name information from the
Person.Person table and accepting a filter for the first name.
Alter the above Store Procedure to supply Default Values if user
does not enter any value.( Use AdventureWorks)
*/
CREATE PROCEDURE upfilterByFirstName
       @FirstName varchar(50)
AS
SELECT FirstName
FROM Person.Person
WHERE FirstName LIKE '%' + @FirstName + '%';
GO
--For Test Filter by Name
EXEC upfilterByFirstName @FirstName = 'ss'
GO
```

98 %

Results | Messages

	FirstName
1	Alyssa
2	Alyssa
3	Alyssa
4	Alyssa
5	Alyssa
6	Alyssa
7	Alyssa
8	Alyssa
9	Alyssa
10	Alyssa
11	Alyssa

Query executed successfully.

```sql
--Alter Method
ALTER PROCEDURE upfilterByFirstName
       @FirstName varchar(50) = ''
AS
SELECT FirstName
FROM Person.Person
WHERE FirstName LIKE '%' + @FirstName + '%';
GO
```

```sql
--For Test Alter method
EXEC upfilterByFirstName
GO
```

98 %

Results | Messages

	FirstName
1	A.
2	A.
3	A. Scott
4	Aaron
5	Aaron
6	Aaron
7	Aaron
8	Aaron
9	Aaron
10	Aaron
11	Aaron

Query executed successfully.

```sql
--Exercise-6
/*     Write a trigger for the Product table to ensure the list price
can never be raised more than 15 Percent in a single change.
Modify the above trigger to execute its check code only if the
ListPrice column is updated (Use AdventureWorks Database).
*/

CREATE TRIGGER [Production].[PriceChangesLimit]
ON [Production].[Product]
FOR UPDATE
AS
    IF EXISTS
        (
        SELECT *
        FROM inserted i
        JOIN deleted d
            ON i.ProductID = d.ProductID
        WHERE i.ListPrice > (d.ListPrice * 1.15)
        )
    BEGIN
        RAISERROR('Price increase may not be greater than 15 percent.Transaction
Failed.',16,1)
        ROLLBACK TRAN
    END
GO
-- Check Value
SELECT ListPrice FROM Production.Product Order By ListPrice DESC;
```

98 %

Results | Messages

	ListPrice
1	19894.8857
2	19894.8857
3	19894.8857
4	19894.8857
5	19894.8857
6	18903.6638
7	18903.6638
8	18903.6638
9	18903.6638
10	18764.6657
11	18764.6657

Query executed successfully.

```sql
--Trigger Test 1
UPDATE Production.Product
SET ListPrice = ListPrice * 1.10
```

98 %

Messages

```
(504 rows affected)

Completion time: 2021-03-09T18:00:36.0605436+05:30
```

```sql
--Trigger Test 2
UPDATE Production.Product
SET ListPrice = ListPrice * 1.50
```

117 %

Messages

```
Msg 50000, Level 16, State 1, Procedure PriceChangesLimit, Line 16 [Batch Start Line 297]
Price increase may not be greater than 15 percent.Transaction Failed.
Msg 3609, Level 16, State 1, Line 298
The transaction ended in the trigger. The batch has been aborted.

Completion time: 2021-03-10T09:57:50.4147684+05:30
```

```sql
--UPDATED TRIGGER
ALTER TRIGGER [Production].[PriceChangesLimit]
ON [Production].[Product]
FOR UPDATE
AS
    IF UPDATE(ListPrice)
    BEGIN
        IF EXISTS
            (
            SELECT *
            FROM inserted i
            JOIN deleted d
                ON i.ProductID = d.ProductID
            WHERE i.ListPrice > (d.ListPrice * 1.15)
            )
        BEGIN
            RAISERROR('List Price cannot be raised more than 15 precent',16,1)
            ROLLBACK TRAN
        END
              ELSE
              BEGIN
                     PRINT 'SUCCESS'
              END
END
```

```sql
--Trigger Test 1
UPDATE Production.Product
SET ListPrice = ListPrice * 1.10
```

117 %

Messages

```
SUCCESS

(504 rows affected)

Completion time: 2021-03-10T09:59:49.3989117+05:30
```

```sql
--Trigger Test 2
UPDATE Production.Product
SET ListPrice = ListPrice * 1.50
```

á

117 %

Messages

```
Msg 50000, Level 16, State 1, Procedure PriceChangesLimit, Line 16 [Batch Start Line 297]
List Price cannot be raised more than 15 precent
Msg 3609, Level 16, State 1, Line 298
The transaction ended in the trigger. The batch has been aborted.

Completion time: 2021-03-10T10:00:33.9798349+05:30
```