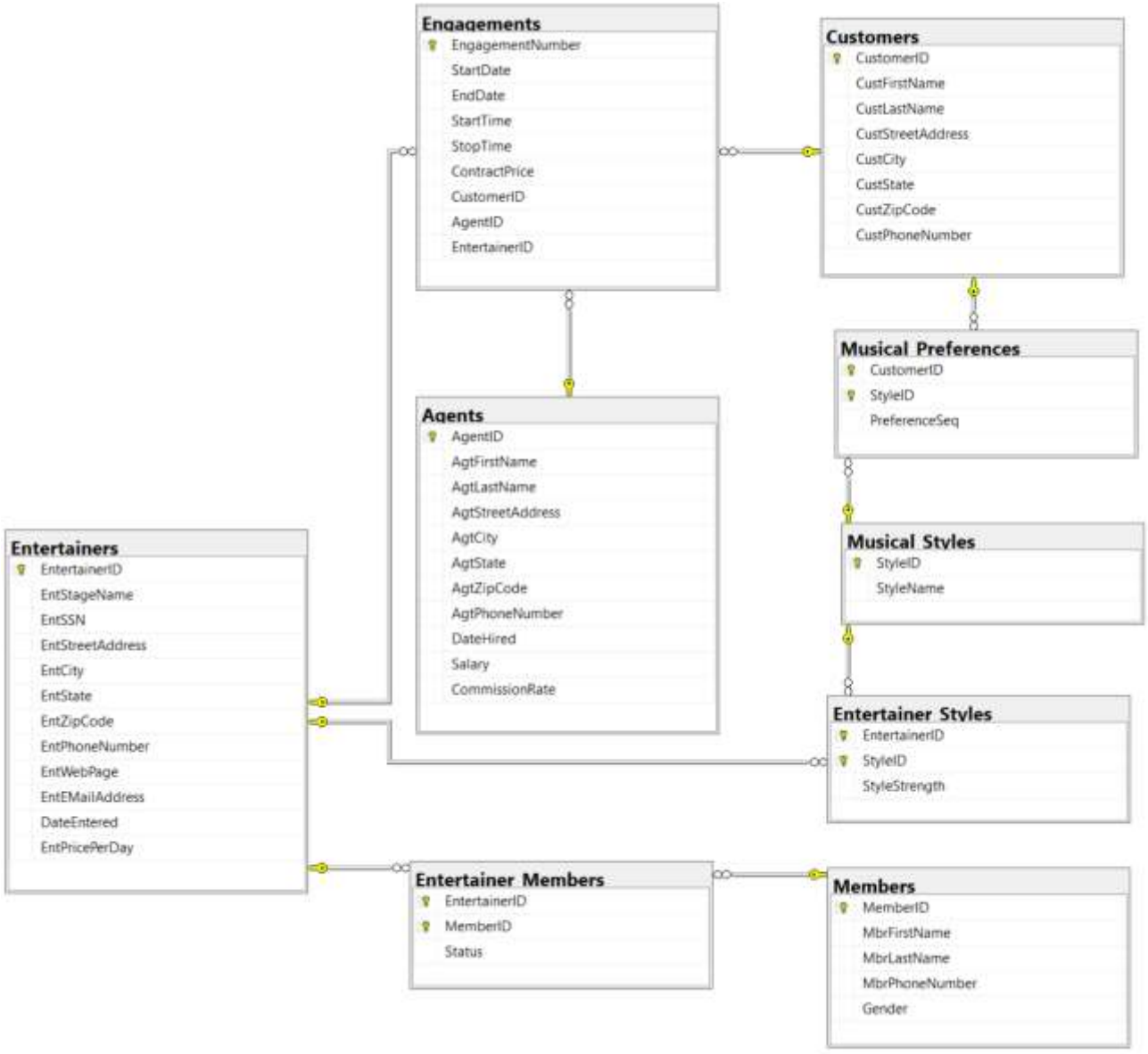
Introduction EntertainmentAgency

This database is designed to manage entertainers, agents, customers, and bookings. You would use a similar design to handle event bookings or hotel reservations.

The database handles scheduling of entertainers with customers. We list all the styles of music that each entertainer plays. We also have a table that contains the musical preferences of each customer.

You can see that the Musical_Preferences table contains a column to rank the customer preferences using a sequence number. In this database, a 1 indicates the customer's first preference, a 2 the second preference, and so on. There is also a column in the Entertainer_Styles table that lists for each style that an entertainer can play the relative strength of that style. For example, customer Zachary Johnson has specified a preference for Rhythm and Blues, Jazz, and Salsa in that order. Entertainer Jazz Persuasion says they focus on Rhythm and Blues, Salsa, and Jazz in that order.

- Download from Chamilo and execute the scripts EntertainmentAgencyStructure.sql and EntertainmentAgencyData.sql
- Create the Database Diagram



SQL Review

1. There are how many different musical Styles?

	NumberOfMusicalStyles
1	25

```
SELECT COUNT(DISTINCT StyleID) As NumberOfMusicalStyles
FROM Musical_Styles
```

2. What is the number of available entertainers per style?

	StyleID	styleName	NumberOfEntertainers
1	3	60's Music	2
2	4	70's Music	1
3	6	Country	2
4	7	Classical	3
5	8	Classic Rock & Roll	1
6	10	Contemporary	2
7	11	Country Rock	1
8	13	Folk	2
9	14	Chamber Music	2
10	15	Jazz	2
11	17	Motown	1
12	19	Rhythm and Blues	2
13	20	Show Tunes	2
14	21	Standards	3
15	22	Top 40 Hits	2
16	23	Variety	2
17	24	Salsa	2

```
SELECT es.StyleID, ms.styleName, COUNT(EntertainerID) As NumberOfEntertainers
FROM Entertainer_Styles es JOIN Musical_Styles ms ON es.StyleID = ms.StyleID
GROUP BY es.StyleID, ms.styleName
```

3. What are the members that belong to more than 1 entertainer?

	memberID	MbrName	NumberOfEntertainers
1	102	Suzanne Viescas	2
2	103	Gary Hallmark	2
3	104	Jeffrey Davidson	2
4	105	Robert Brown	2
5	107	Sara Sheskey	2
6	112	Kim Smith	2
7	114	George Johnson	2
8	117	Luke Patterson	2
9	118	Janice Davidson	2
10	120	Michael Hernandez	3
11	121	Katherine Smith	3
12	123	Susan Davidson	2
13	124	Caroline Viescas	2

```
SELECT m.memberID, m.MbrFirstName + ' ' + m.MbrLastName As MbrName, COUNT(DISTINCT
EntertainerID) As NumberOfEntertainers
FROM Entertainer_Members em JOIN members m ON em.MemberID = m.MemberID
GROUP BY m.memberID, m.MbrFirstName + ' ' + m.MbrLastName
HAVING COUNT(DISTINCT EntertainerID) > 1
```

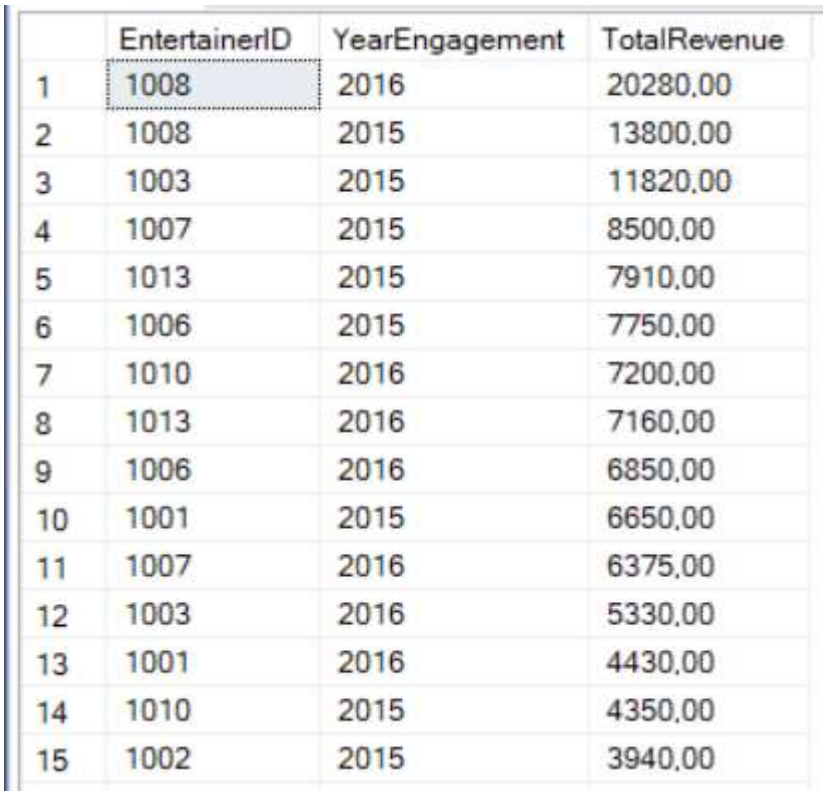
4. What is the number of engagements per year and per entertainer? Use StartDate to determine the year. Order on the number of engagements in descending way. The image below shows only a part of the resultset.

	EntertainerID	YearEngagement	NumberOfEngagements
1	1008	2015	9
2	1001	2015	7
3	1013	2015	7
4	1003	2015	6
5	1008	2016	6
6	1010	2016	6
7	1002	2015	5
8	1006	2015	5
9	1007	2015	5
10	1011	2015	5
11	1004	2016	5
12	1006	2016	5
13	1001	2016	4
14	1004	2015	4
15	1005	2015	4

```
SELECT EntertainerID, YEAR(StartDate) As YearEngagement, COUNT(EngagementNumber) As
NumberOfEngagements
FROM Engagements
```

```
GROUP BY EntertainerID, YEAR(StartDate)
ORDER BY 3 DESC
```

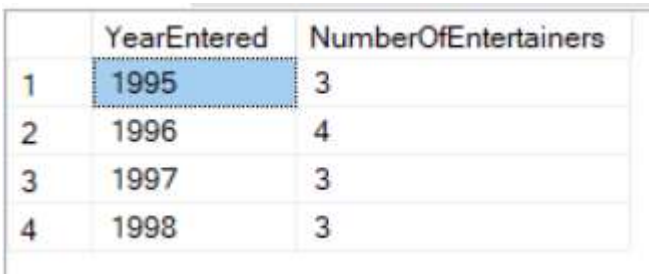
5. What is the total revenue for each entertainer per year? Use contractprice to calculate the revenue. Order on the total revenue in descending way. The image below shows only a part of the resultset.



	EntertainerID	YearEngagement	TotalRevenue
1	1008	2016	20280.00
2	1008	2015	13800.00
3	1003	2015	11820.00
4	1007	2015	8500.00
5	1013	2015	7910.00
6	1006	2015	7750.00
7	1010	2016	7200.00
8	1013	2016	7160.00
9	1006	2016	6850.00
10	1001	2015	6650.00
11	1007	2016	6375.00
12	1003	2016	5330.00
13	1001	2016	4430.00
14	1010	2015	4350.00
15	1002	2015	3940.00

```
SELECT EntertainerID, YEAR(StartDate) As YearEngagement, SUM(contractprice) As TotalRevenue
FROM Engagements
GROUP BY EntertainerID, YEAR(StartDate)
ORDER BY 3 DESC
```

6. How many entertainers were entered per year? Order by year in ascending way.



	YearEntered	NumberOfEntertainers
1	1995	3
2	1996	4
3	1997	3
4	1998	3

```
SELECT YEAR(DateEntered) As YearEntered, COUNT(EntertainerID) As NumberOfEntertainers
FROM Entertainers
GROUP BY YEAR(DateEntered)
ORDER BY YEAR(DateEntered)
```

7. Give for each agent the total income per year. An agent has a salary and a commission on the contractprice. Round the result to 2 decimals.

	AgentID	AgtName	YearEngagement	TotalSalary
1	1	William Thompson	2015	35510
2	1	William Thompson	2016	35285,8
3	2	Scott Johnson	2015	27105,6
4	2	Scott Johnson	2016	27163,2
5	3	Carol Viescas	2015	30912,5
6	3	Carol Viescas	2016	30327,5
7	4	Karen Smith	2015	22664,12
8	4	Karen Smith	2016	22358,6
9	5	Marianne Davidson	2015	25005,13
10	5	Marianne Davidson	2016	25013,45
11	6	John Kennedy	2015	33099
12	6	John Kennedy	2016	34367,1
13	7	Caleb Viescas	2015	22336,6
14	7	Caleb Viescas	2016	22235,98
15	8	Maria Patterson	2015	30337,4
16	8	Maria Patterson	2016	30175,6

```
SELECT a.AgentID, a.AgtFirstName + ' ' + a.AgtLastName As AgtName, YEAR(StartDate) As
YearEngagement, ROUND(SUM(a.CommissionRate * e.contractprice) + Salary, 2) As TotalSalary
FROM Agents a JOIN Engagements e
ON a.AgentID = e.AgentID
GROUP BY a.AgentID,  a.AgtFirstName + ' ' + a.AgtLastName, YEAR(StartDate), a.Salary
```

8. What is the number of male and female entertainers.

	Gender	NumberOfMembers
1	NULL	1
2	F	12
3	M	12

```
SELECT Gender, COUNT(memberID) As NumberOfMembers
FROM Members
GROUP BY Gender
```

Extension: Gender = NULL is perhaps an entertainer that didn't want to reveal his gender. Change the previous solution into the following solution.

	Gender	NumberOfMembers
1	X	1
2	F	12
3	M	12

```
SELECT ISNULL(Gender, 'X') As Gender, COUNT(memberID) As NumberOfMembers
FROM Members
GROUP BY Gender
```

9. What is the average number of members per entertainer per musical style. Only take into account StyleStrength = 1. Order by the average number of members in descending way.

	StyleID	StyleName	AverageNumberOfMembers
1	3	60's Music	6
2	6	Country	5
3	11	Country Rock	5
4	15	Jazz	4
5	22	Top 40 Hits	4
6	23	Variety	4
7	20	Show Tunes	3
8	19	Rhythm and Blues	2
9	13	Folk	1
10	14	Chamber Music	1

```
SELECT es.StyleID, ms.StyleName, COUNT(em.MemberID) / COUNT(DISTINCT es.EntertainerID) As
AverageNumberOfMembers
FROM Entertainer_Styles es JOIN Entertainer_Members em ON es.EntertainerID =
em.EntertainerID
JOIN Musical_Styles ms ON es.StyleID = ms.StyleID
WHERE es.StyleStrength = 1
GROUP BY es.StyleID, ms.StyleName
ORDER BY 3 DESC
```

10. What is number of entertainers each agent already worked with?

	AgentID	AgtName	NumberOfEntertainers
1	1	William Thompson	11
2	2	Scott Johnson	5
3	3	Carol Viescas	10
4	4	Karen Smith	10
5	5	Marianne Davidson	10
6	6	John Kennedy	8
7	7	Caleb Viescas	6
8	8	Maria Patterson	8

```
SELECT a.AgentID,a.AgtFirstName + ' ' + a.AgtLastName As AgtName, COUNT(DISTINCT
EntertainerID) As NumberOfEntertainers
FROM Agents a JOIN Engagements e
ON a.AgentID = e.AgentID
GROUP BY a.AgentID, a.AgtFirstName + ' ' + a.AgtLastName
```

11. What is number of agents each entertainer already worked with?

	EntertainerID	EntStageName	NumberOfAgents
1	1001	Carol Peacock Trio	5
2	1002	Topazz	6
3	1003	JV & the Deep Six	6
4	1004	Jim Glynn	5
5	1005	Jazz Persuasion	6
6	1006	Modern Dance	7
7	1007	Coldwater Cattle Company	6
8	1008	Country Feeling	6
9	1010	Saturday Revue	5
10	1011	Julia Schnebly	5
11	1012	Susan McLain	5
12	1013	Caroline Coie Cuartet	6

```
SELECT et.EntertainerID, et.EntStageName, COUNT(DISTINCT AgentID) As NumberOfAgents
FROM Entertainers et JOIN Engagements e
ON et.EntertainerID = e.EntertainerID
GROUP BY  et.EntertainerID, et.EntStageName
```

12. What are the engagements for which the contractprice is 50% more expensive than the number of days * EntPricePerDay?

	EngagementNumber	AgentID	ContractPrice	CalculatedPrice
1	6	7	2300,00	1400,00
2	11	4	950,00	560,00
3	14	1	2750,00	1680,00
4	24	4	1940,00	1225,00
5	48	1	950,00	550,00
6	58	2	770,00	480,00
7	62	2	500,00	250,00
8	68	1	1670,00	750,00
9	82	8	950,00	550,00
10	90	5	320,00	175,00
11	91	3	770,00	480,00
12	97	8	110,00	60,00
13	99	6	14105,00	8960,00
14	107	4	200,00	120,00

```
SELECT e.EngagementNumber, e.AgentID, e.ContractPrice, (DATEDIFF(DAY, StartDate, EndDate) +
1) * EntPricePerDay As CalculatedPrice
FROM Engagements e JOIN Entertainers et ON e.EntertainerID = et.EntertainerID
WHERE e.ContractPrice > 1.5 * (DATEDIFF(DAY, StartDate, EndDate) + 1) * EntPricePerDay
```

13. What is the average price per musical style based on EntPricePerDay. Only take into account StyleStrength = 1. Order by average price in descending way.

	StyleID	StyleName	AveragePrice
1	6	Country	280,00
2	11	Country Rock	275,00
3	3	60's Music	275,00
4	15	Jazz	250,00
5	22	Top 40 Hits	250,00
6	23	Variety	250,00
7	20	Show Tunes	175,00
8	19	Rhythm and Blues	122,50
9	14	Chamber Music	117,50
10	13	Folk	67,50

```
SELECT ms.StyleID, ms.StyleName, AVG(e.EntPricePerDay) As AveragePrice
FROM Musical_Styles ms JOIN Entertainer_Styles es ON ms.StyleID = es.StyleID
JOIN Entertainers e ON e.EntertainerID = es.EntertainerID
WHERE StyleStrength = 1
GROUP BY ms.StyleID, ms.StyleName
ORDER BY 3 DESC
```

14. What are the music styles for which we don't have any entertainer available in the database?

	StyleName
1	40's Ballroom Music
2	50's Music
3	80's Music
4	90's Music
5	Elvis
6	Karaoke
7	Modern Rock
8	Rap

```
SELECT DISTINCT ms.StyleName
FROM Musical_Styles ms LEFT JOIN Entertainer_Styles es ON ms.StyleID = es.StyleID
WHERE es.StyleStrength IS NULL
```

SQL 2 TIN

1. What are the entertainers without any engagements up till now and which are available in the database.

	EntertainerID	EntStageName
1	1009	Katherine Ehrlich

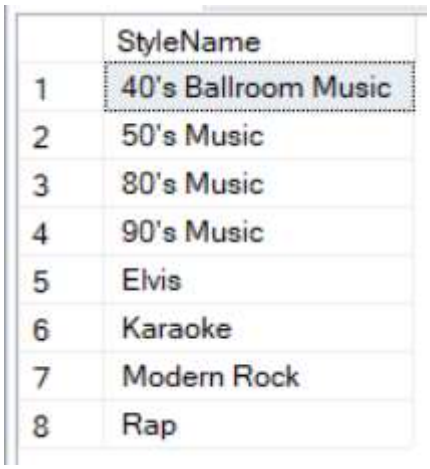
```
SELECT EntertainerID, EntStageName
FROM Entertainers
WHERE EntertainerID NOT IN (SELECT EntertainerID FROM Engagements)
```

2. What are the entertainers without any engagements for 2016

	EntertainerID	EntStageName
1	1009	Katherine Ehrlich

```
SELECT EntertainerID, EntStageName
FROM Entertainers
WHERE EntertainerID NOT IN (SELECT EntertainerID FROM Engagements WHERE YEAR(StartDate) =
2016)
```

3. What are the music styles for which we don't have any entertainer available in the database?



	StyleName
1	40's Ballroom Music
2	50's Music
3	80's Music
4	90's Music
5	Elvis
6	Karaoke
7	Modern Rock
8	Rap

```
SELECT DISTINCT StyleName
FROM Musical_Styles WHERE StyleID NOT IN (SELECT DISTINCT StyleID FROM Entertainer_Styles)
```

4. What are the music styles for which we don't have any entertainer available in the database with StyleStrength = 1?

	StyleName
1	40's Ballroom Music
2	50's Music
3	70's Music
4	80's Music
5	90's Music
6	Classic Rock & Roll
7	Classical
8	Contemporary
9	Elvis
10	Karaoke
11	Modern Rock
12	Motown
13	Rap
14	Salsa
15	Standards

```
SELECT DISTINCT StyleName
FROM Musical_Styles WHERE StyleID NOT IN (SELECT DISTINCT StyleID FROM Entertainer_Styles
WHERE StyleStrength = 1)
```

5. What is the musical style for which we have most entertainers available in the database? Don't take into account the style strength.

	StyleID	StyleName	NumberOfEntertainers
1	7	Classical	3
2	21	Standards	3

```
SELECT ms.StyleID, ms.StyleName, count(es.EntertainerID) As NumberOfEntertainers
FROM Musical_Styles ms JOIN Entertainer_Styles es ON ms.StyleID = es.StyleID
GROUP BY ms.StyleID, ms.StyleName
HAVING count(es.EntertainerID) = (SELECT TOP 1 count(EntertainerID) FROM Entertainer_Styles
GROUP BY StyleID ORDER BY 1 DESC)
```

6. Who is the most important customer based on the number of engagements?

	CustomerID	CustName	NumberOfEngagements
1	10001	Doug Steele	8
2	10005	Elizabeth Hallmark	8
3	10009	Sarah Thompson	8

```
SELECT c.CustomerID, c.CustFirstName + ' ' + c.CustLastName As CustName,
COUNT(e.EngagementNumber) As NumberOfEngagements
FROM Customers c JOIN Engagements e ON c.CustomerID = e.CustomerID
GROUP BY c.CustomerID, c.CustFirstName + ' ' + c.CustLastName
HAVING COUNT(e.EngagementNumber) = (SELECT TOP 1 COUNT(engagementNumber) FROM Engagements
GROUP BY customerid)
```

7. What are the customers that have booked the same entertainer every year.

	CustomerID	EntertainerID	NumberOfYears
1	10002	1008	2
2	10002	1013	2
3	10003	1006	2
4	10004	1005	2
5	10004	1012	2
6	10005	1008	2
7	10006	1008	2
8	10007	1004	2
9	10007	1013	2
10	10009	1004	2
11	10009	1005	2
12	10010	1006	2
13	10010	1007	2
14	10010	1010	2
15	10013	1002	2
16	10015	1011	2

Solution 1 → with CTE

```
WITH cte_NumberOfYears(NumberOfYears)
AS
(SELECT COUNT(DISTINCT YEAR(StartDate))
FROM Engagements)

SELECT CustomerID, EntertainerID, COUNT(DISTINCT YEAR(StartDate)) As NumberOfYear
FROM Engagements
GROUP BY  CustomerID, EntertainerID
HAVING COUNT(DISTINCT YEAR(StartDate)) = (SELECT NumberofYears FROM cte_NumberOfYears)
ORDER BY CustomerID ASC
```

Solution 2 → with subquery

```
SELECT CustomerID, EntertainerID, COUNT(DISTINCT YEAR(StartDate)) As NumberOfYears
FROM Engagements
GROUP BY CustomerID, EntertainerID
HAVING COUNT(DISTINCT YEAR(StartDate)) = (SELECT COUNT(DISTINCT YEAR(StartDate)) FROM
Engagements)
ORDER BY CustomerID
```

8. For each customer, provide a list of entertainers that they have booked, but whose music styles do not belong to their preferences. The image below shows only a part of the resultset.

	CustomerID	EntertainerID
1	10001	1002
2	10001	1003
3	10001	1005
4	10001	1007
5	10001	1008
6	10002	1007
7	10002	1010
8	10002	1011
9	10002	1013
10	10003	1006

```
SELECT DISTINCT e.CustomerID, e.EntertainerID
FROM  Engagements e
WHERE (NOT EXISTS(SELECT StyleID FROM Musical_Preferences WHERE CustomerID = e.CustomerID
INTERSECT SELECT StyleID FROM Entertainer_Styles WHERE EntertainerID = e.EntertainerID))
ORDER BY e.CustomerID
```

More elaborate

	CustomerID	CustName	EntertainerID	EntStageName
1	10001	Doug Steele	1002	Topazz
2	10001	Doug Steele	1003	JV & the Deep Six
3	10001	Doug Steele	1005	Jazz Persuasion
4	10001	Doug Steele	1007	Coldwater Cattle Company
5	10001	Doug Steele	1008	Country Feeling
6	10002	Deb Smith	1007	Coldwater Cattle Company
7	10002	Deb Smith	1010	Saturday Revue
8	10002	Deb Smith	1011	Julia Schnebly
9	10002	Deb Smith	1013	Caroline Coie Cuartet
10	10003	Ben Clothier	1006	Modern Dance

```
SELECT DISTINCT e.CustomerID, c.CustFirstName + ' ' + c.CustLastName As CustName,
e.EntertainerID, et.EntStageName
FROM  Engagements e JOIN Customers c ON c.CustomerID = e.CustomerID
JOIN Entertainers et ON e.EntertainerID = et.EntertainerID
WHERE (NOT EXISTS(SELECT StyleID FROM Musical_Preferences WHERE CustomerID = e.CustomerID
INTERSECT SELECT StyleID FROM Entertainer_Styles WHERE EntertainerID = e.EntertainerID))
ORDER BY e.CustomerID
```

9. What is the number of engagements per season? Use Startdate to determine the season.
   Winter : december + january + february / Spring = march + april + may / Summer = …
   Order by the number of engagements in ascending way.

	Season	NumberOfEngagements
1	Spring	1
2	Autumn	45
3	Winter	65

```
WITH EngagementsPerSeason(Season, EngagementNumber)
AS (select
case
when MONTH(StartDate) IN (1, 2, 12) then 'Winter'
when MONTH(StartDate) IN (3, 4, 5) then 'Spring'
when MONTH(StartDate) IN (6, 7, 8) then 'Summer'
else 'Autumn'
end, EngagementNumber
from Engagements
)

SELECT Season, COUNT(EngagementNumber) As NumberOfEngagements
FROM EngagementsPerSeason
GROUP BY Season
ORDER BY 2
```

10. We are looking for a Salsa group. Which entertainer is the cheapest one?

	EntertainerID	EntStageName	EntPricePerDay
1	1005	Jazz Persuasion	125,00

```
WITH cte AS
(SELECT e.EntertainerID, e. EntStageName, e.EntPricePerDay
FROM Musical_Styles ms JOIN Entertainer_Styles es ON ms.StyleID = es.StyleID
JOIN Entertainers e ON e.EntertainerID = es.EntertainerID
WHERE ms.StyleName = 'Salsa')

SELECT *
FROM cte
WHERE EntPricePerDay = (SELECT MIN(entPricePerDay) FROM cte)
```

11. We are looking for a Salsa group. Which entertainer is the most popular one?

	EntertainerID	EntStageName
1	1006	Modern Dance

```
WITH cte AS
(SELECT e.EntertainerID, e. EntStageName, COUNT(en.EngagementNumber) AS NumberOfEngagements
FROM Musical_Styles ms JOIN Entertainer_Styles es ON ms.StyleID = es.StyleID
JOIN Entertainers e ON e.EntertainerID = es.EntertainerID
JOIN Engagements en ON e.EntertainerID = en.EntertainerID
WHERE ms.StyleName = 'Salsa'
GROUP BY e.EntertainerID, e.EntStageName)

SELECT * FROM cte
WHERE NumberOfEngagements = (SELECT MAX(NumberOfEngagements) FROM cte)
```

12. A member can be part of more than 1 entertainer. We assume that the contractprice per engagement is distributed fairly among all members of an entertainer. Calculate the total revenue per member. Order by the total revenue per member in descending way.
    The image below shows only a part of the resultset.
    Tip: calculate the number of members per entertainer first.

	MemberID	MbrName	IncomePerMember
1	114	George Johnson	10583,50
2	105	Robert Brown	9791,00
3	103	Gary Hallmark	9674,333
4	120	Michael Hernandez	8786,6662
5	118	Janice Davidson	7343,3326
6	115	Joe Smith	6816,00
7	111	Kathryn Patterson	6816,00
8	107	Sara Sheskey	6668,3326
9	112	Kim Smith	6655,00
10	124	Caroline Viescas	6655,00

```
WITH NumberOfMembersPerEntertainer(EntertainerID, NumberOfMembers) AS
(SELECT e.EntertainerID, COUNT(DISTINCT em.MemberID)
FROM Entertainers e JOIN Entertainer_Members em ON e.EntertainerID = em.EntertainerID
GROUP BY e.EntertainerID)

SELECT  em.MemberID, m.MbrFirstName + ' ' + m.MbrLastName As 'MbrName', SUM(e.contractprice
/ n.NumberOfMembers) As IncomePerMember
FROM Engagements e JOIN NumberOfMembersPerEntertainer n ON e.EntertainerID = n.EntertainerID
JOIN Entertainer_Members em ON e.EntertainerID = em.EntertainerID JOIN Members m ON
em.MemberID = m.MemberID
GROUP BY em.MemberID, m.MbrFirstName + ' ' + m.MbrLastName
ORDER BY IncomePerMember DESC;
```

13. We are receiving signals that prices in the sector have risen exuberantly. We want to verify this with data.
    Therefore, make the overview below where the price for an engagement is compared between 2015 and 2016 only for engagements with the same customer, the same entertainer and the same number of days

	CustomerID	EntertainerID	NumberOfDays	ContractPrice_2015	ContractPrice_2016
1	10003	1006	4	770,00	770,00
2	10013	1002	9	860,00	1130,00
3	10010	1007	4	1550,00	1550,00

```
WITH Engagements_2015(CustomerID, EntertainerID, NumberOfDays, ContractPrice) AS
(SELECT CustomerID, EntertainerId, (DATEDIFF(DAY, StartDate, EndDate) + 1), ContractPrice
FROM Engagements
WHERE YEAR(StartDate) = 2015),

Engagements_2016(CustomerID, EntertainerID, NumberOfDays, ContractPrice) AS
(SELECT CustomerID, EntertainerId, (DATEDIFF(DAY, StartDate, EndDate) + 1), ContractPrice
FROM Engagements
WHERE YEAR(StartDate) = 2016)

SELECT fifteen.CustomerID, fifteen.EntertainerID, fifteen.NumberOfDays,
fifteen.ContractPrice As ContractPrice_2015, sixteen.ContractPrice As ContractPrice_2016
FROM Engagements_2015 fifteen JOIN Engagements_2016 sixteen
ON fifteen.CustomerID = sixteen.CustomerID AND fifteen.EntertainerID = sixteen.EntertainerID
AND fifteen.NumberOfDays = sixteen.NumberOfDays
```

Extension: How often have we noted a price increase, a price decrease and a tie?

	Evolution	NumberOfTimes
1	Increase	1
2	Tie	2

```
WITH Engagements_2015(CustomerID, EntertainerID, NumberOfDays, ContractPrice) AS
(SELECT CustomerID, EntertainerId, (DATEDIFF(DAY, StartDate, EndDate) + 1), ContractPrice
FROM Engagements
WHERE YEAR(StartDate) = 2015),

Engagements_2016(CustomerID, EntertainerID, NumberOfDays, ContractPrice) AS
(SELECT CustomerID, EntertainerId, (DATEDIFF(DAY, StartDate, EndDate) + 1), ContractPrice
FROM Engagements
WHERE YEAR(StartDate) = 2016),

Engagements_2015_2016(CustomerID, EntertainerID, NumberOfDays, ContractPrice_2015,
ContractPrice_2016) AS
(SELECT fifteen.CustomerID, fifteen.EntertainerID, fifteen.NumberOfDays,
fifteen.ContractPrice As ContractPrice_2015, sixteen.ContractPrice As ContractPrice_2016
FROM Engagements_2015 fifteen, Engagements_2016 sixteen
WHERE fifteen.CustomerID = sixteen.CustomerID AND fifteen.EntertainerID =
sixteen.EntertainerID AND fifteen.NumberOfDays = sixteen.NumberOfDays),

PriceEvolution(Evolution) AS
(SELECT
CASE
WHEN ContractPrice_2015 < ContractPrice_2016 THEN 'Increase'
WHEN ContractPrice_2015 > ContractPrice_2016 THEN 'Decrease'
ELSE 'Tie'
END
FROM Engagements_2015_2016)

SELECT Evolution, COUNT(Evolution) As NumberOfTimes
FROM PriceEvolution
GROUP BY Evolution;
```

14. Give for each year the top 3 of most popular entertainers (= entertainers with most engagements for that year)

	entertainerid	year	numberofengagements	dense_ranking
1	1008	2015	9	1
2	1001	2015	7	2
3	1013	2015	7	2
4	1003	2015	6	3
5	1008	2016	6	1
6	1010	2016	6	1
7	1003	2016	5	2
8	1004	2016	5	2
9	1006	2016	5	2
10	1001	2016	4	3
11	1013	2016	4	3

```
WITH NumberOfEngagementsPerYear(YearEngagement, EntertainerID, NumberOfEngagements) AS
```

```
WITH NumberOfEngagementsPerYear(EntertainerID, YearEngagement,NumberOfEngagements) AS
(SELECT entertainerid, year(startdate), COUNT(engagementnumber)
FROM engagements
GROUP BY entertainerid, year(startdate)),

RankingNumberOfEngagements(YearEngagement, EntertainerID,
NumberOfEngagements,DenseRankEngagements) AS
(SELECT YearEngagement, EntertainerID, NumberOfEngagements, dense_rank() OVER (partition by
YearEngagement order by NumberOfEngagements DESC) AS DenseRankEngagements
FROM NumberOfEngagementsPerYear)

SELECT * FROM RankingNumberOfEngagements
WHERE DenseRankEngagements <= 3
```

15. We are wondering if an entertainer was more popular in 2015 compared to 2016 (= if an entertainer had more engagements in 2015 than in 2016).
    Calculate the number of engagements per entertainer per year. The image below shows only a part of the resultset.

	EntertainerID	YearEngagement	(No column name)
1	1001	2015	7
2	1002	2015	5
3	1003	2015	6
4	1004	2015	4
5	1005	2015	4
6	1006	2015	5
7	1007	2015	5
8	1008	2015	9
9	1010	2015	3
10	1011	2015	5
11	1012	2015	3
12	1013	2015	7
13	1001	2016	4
14	1002	2016	2
15	1003	2016	4
16	1004	2016	5
17	1005	2016	3
18	1006	2016	5
19	1007	2016	3
20	1008	2016	6

```
SELECT DISTINCT  EntertainerID, YEAR(StartDate) as YearEngagement,COUNT(EntertainerID)
FROM Engagements
GROUP BY EntertainerID, YEAR(StartDate)
```

Extension: Change the previous overview into the following overview

	EntertainerID	YearEngagement	NumberOfEngagements2015	NumberOfEngagements2016
1	1001	2015	7	4
2	1001	2016	4	NULL
3	1002	2015	5	2
4	1002	2016	2	NULL
5	1003	2015	6	4
6	1003	2016	4	NULL
7	1004	2015	4	5
8	1004	2016	5	NULL
9	1005	2015	4	3
10	1005	2016	3	NULL
11	1006	2015	5	5
12	1006	2016	5	NULL
13	1007	2015	5	3
14	1007	2016	3	NULL
15	1008	2015	9	6
16	1008	2016	6	NULL
17	1010	2015	3	6
18	1010	2016	6	NULL
19	1011	2015	5	3
20	1011	2016	3	NULL

```
WITH NumberOfEngagementsPerYear(EntertainerID, YearEngagement, NumberOfEngagements) AS
(SELECT DISTINCT  EntertainerID, YEAR(StartDate) as YearEngagement,COUNT(EntertainerID)
FROM Engagements
GROUP BY EntertainerID, YEAR(StartDate))

SELECT EntertainerID, YearEngagement, NumberOfEngagements As NumberOfEngagements2015,
LEAD(NumberOfEngagements) OVER (Partition BY EntertainerID ORDER BY EntertainerID)  As
NumberOfEngagements2016
FROM NumberOfEngagementsPerYear
```

Extension: Change the previous overview into the following overview

	EntertainerID	NumberOfEngagements2015	NumberOfEngagements2016	RelativeDifference
1	1001	7	4	-42.86%
2	1002	5	2	-60.00%
3	1003	6	4	-33.33%
4	1004	4	5	25.00%
5	1005	4	3	-25.00%
6	1006	5	5	0.00%
7	1007	5	3	-40.00%
8	1008	9	6	-33.33%
9	1010	3	6	100.00%
10	1011	5	3	-40.00%
11	1012	3	3	0.00%
12	1013	7	4	-42.86%

```
WITH NumberOfEngagementsPerYear(EntertainerID, YearEngagement, NumberOfEngagements) AS
```

```
(SELECT DISTINCT  EntertainerID, YEAR(StartDate) as YearEngagement,COUNT(EntertainerID)
FROM Engagements
GROUP BY EntertainerID, YEAR(StartDate)),

NumberOfEngagements2015_2016(EntertainerID, YearEngagement, NumberOfEngagements2015,
NumberOfEngagements2016) AS
(SELECT EntertainerID, YearEngagement, NumberOfEngagements As NumberOfEngagements2015,
LEAD(NumberOfEngagements) OVER (Partition BY EntertainerID ORDER BY EntertainerID)  As
NumberOfEngagements2016
FROM NumberOfEngagementsPerYear)

SELECT EntertainerID, NumberOfEngagements2015, NumberOfEngagements2016,
FORMAT((NumberOfEngagements2016 - NumberOfEngagements2015) * 1.0 / NumberOfEngagements2015,
'P') As RelativeDifference
FROM NumberOfEngagements2015_2016
WHERE YearEngagement = 2015
```

16. Create the following overview for each entertainer: the total revenue per month en the running total per year. The image below shows only a part of the resultset.

	EntertainerID	YearContract	MonthContract	ContractPricePerMonth	ContractPriceTotalPerYear
1	1001	2015	9	1670,00	1670,00
2	1001	2015	10	2490,00	4160,00
3	1001	2015	11	680,00	4840,00
4	1001	2015	12	1810,00	6650,00
5	1001	2016	1	1990,00	1990,00
6	1001	2016	2	2440,00	4430,00
7	1002	2015	9	1360,00	1360,00
8	1002	2015	10	1810,00	3170,00
9	1002	2015	12	770,00	3940,00
10	1002	2016	1	1130,00	1130,00
11	1002	2016	2	1550,00	2680,00
12	1003	2015	9	6270,00	6270,00
13	1003	2015	10	2210,00	8480,00
14	1003	2015	12	3340,00	11820,00
15	1003	2016	1	2350,00	2350,00
16	1003	2016	2	1130,00	3480,00
17	1003	2016	3	1850,00	5330,00
18	1004	2015	9	1035,00	1035,00
19	1004	2015	12	230,00	1265,00
20	1004	2016	1	1080,00	1080,00
21	1004	2016	2	685,00	1765,00

```
WITH ContractPerYearAndMonths(EntertainerID, YearContract, MonthContract,
ContractPricePerMonth) As
(SELECT DISTINCT EntertainerID, YEAR(StartDate) As YearContract, MONTH(StartDate) As
MonthContract, SUM(ContractPrice)
FROM Engagements
GROUP BY EntertainerID, YEAR(StartDate), MONTH(StartDate))

SELECT EntertainerID, YearContract, MonthContract, ContractPricePerMonth,
SUM(ContractPricePerMonth) OVER (PARTITION BY EntertainerID, YearContract ORDER BY
MonthContract
RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) As ContractPriceTotalPerYear
FROM ContractPerYearAndMonths;
```

17. We are wondering if there is somehow a correlation between the EntPricePerDay of an entertainer and the popularity of the entertainer: is the most expensive entertainer the one with the fewest or the most engagements?
    Create the following overview to get an idea. Order the resultset by EntPricePerDay in a descending way.
    Tip: calculate the number of engagements per entertainer first. The column RelativePart is calculated based on the number of engagements of each entertainer.

	EntertainerID	EntPricePerDay	RelativePart
1	1008	280,00	13.51%
2	1003	275,00	29.73%
3	1007	275,00	29.73%
4	1013	250,00	56.76%
5	1006	250,00	56.76%
6	1010	250,00	56.76%
7	1001	175,00	66.67%
8	1005	125,00	72.97%
9	1002	120,00	79.28%
10	1011	90,00	86.49%
11	1012	75,00	91.89%
12	1004	60,00	100.00%

```
WITH EngagementsPerEntertainer (EntertainerID, EntPricePerDay, NumberOfEngagements) AS
(SELECT e.EntertainerID, e.EntPricePerDay, COUNT(EngagementNumber) As NumberOfEngagements
FROM Entertainers e JOIN Engagements eg ON e.EntertainerID = eg.EntertainerID
GROUP BY e.EntertainerID, e.EntPricePerDay)

SELECT EntertainerID, EntPricePerDay, FORMAT(1.0 * SUM(NumberOfEngagements) OVER (ORDER BY
EntPricePerDay DESC RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) /
SUM(NumberOfEngagements) OVER (ORDER BY EntPricePerDay DESC RANGE BETWEEN UNBOUNDED
PRECEDING AND UNBOUNDED FOLLOWING), 'P') As RelativePart
FROM EngagementsPerEntertainer;
```

18. We want to know for each entertainer if they got theirselves a raise between 2015 and 2016.
    Calculate for each entertainer the averge dayprice per engagement (based on the contractprice and the number of days) per year. The image below shows only a part of the resultset.

	EntertainerID	YearEngagement	AverageCalculatedDayPrice
1	1001	2015	163,9058
2	1002	2015	128,1944
3	1003	2015	324,1666
4	1004	2015	58,988
5	1005	2015	180,9523
6	1006	2015	312,75
7	1007	2015	329,75
8	1008	2015	339,2812
9	1010	2015	167,3484
10	1011	2015	76,6514
11	1012	2015	79,5833
12	1013	2015	230,068
13	1001	2016	220,9027
14	1002	2016	140,2777
15	1003	2016	282,0833
16	1004	2016	83,5833
17	1005	2016	97,7777
18	1006	2016	234,373
19	1007	2016	282,3484
20	1008	2016	292,0283
21	1010	2016	290,0694
22	1011	2016	85,7142

```
SELECT EntertainerID, YEAR(StartDate) As YearEngagement, AVG(ContractPrice / (DATEDIFF(DAY,
StartDate, EndDate) + 1)) As AverageCalculatedDayPrice
FROM Engagements
GROUP BY EntertainerID, YEAR(StartDate)
```

Extension: Change the previous overview into the following overview

	EntertainerID	YearEngagement	AverageCalculatedDayPrice2015	AverageCalculatedDayPrice2016
1	1001	2015	163,9058	220,9027
2	1001	2016	220,9027	NULL
3	1002	2015	128,1944	140,2777
4	1002	2016	140,2777	NULL
5	1003	2015	324,1666	282,0833
6	1003	2016	282,0833	NULL
7	1004	2015	58,988	83,5833
8	1004	2016	83,5833	NULL
9	1005	2015	180,9523	97,7777
10	1005	2016	97,7777	NULL
11	1006	2015	312,75	234,373
12	1006	2016	234,373	NULL
13	1007	2015	329,75	282,3484
14	1007	2016	282,3484	NULL
15	1008	2015	339,2812	292,0283
16	1008	2016	292,0283	NULL
17	1010	2015	167,3484	290,0694
18	1010	2016	290,0694	NULL
19	1011	2015	76,6514	85,7142
20	1011	2016	85,7142	NULL
21	1012	2015	79,5833	74,9999
22	1012	2016	74,9999	NULL

```
WITH AverageCalculatedPricePerEntertainer(EntertainerID, YearEngagement,
AverageCalculatedDayPrice) AS (
SELECT EntertainerID, YEAR(StartDate) As YearEngagement, AVG(ContractPrice / (DATEDIFF(DAY,
StartDate, EndDate) + 1)) As AverageCalculatedDayPrice
FROM Engagements
GROUP BY EntertainerID, YEAR(StartDate))
SELECT EntertainerID, YearEngagement, AverageCalculatedDayPrice As
AverageCalculatedDayPrice2015, LEAD(AverageCalculatedDayPrice) OVER (Partition BY
EntertainerID ORDER BY EntertainerID)  As AverageCalculatedDayPrice2016
FROM AverageCalculatedPricePerEntertainer
```

Extension: Change the previous overview into the following overview to get the relative raise per entertainer

	EntertainerID	AverageCalculatedDayPrice2015	AverageCalculatedDayPrice2016	RelativeDifference
1	1001	163,9058	220,9027	34.77%
2	1002	128,1944	140,2777	9.43%
3	1003	324,1666	282,0833	-12.98%
4	1004	58,988	83,5833	41.70%
5	1005	180,9523	97,7777	-45.96%
6	1006	312,75	234,373	-25.06%
7	1007	329,75	282,3484	-14.38%
8	1008	339,2812	292,0283	-13.93%
9	1010	167,3484	290,0694	73.33%
10	1011	76,6514	85,7142	11.82%
11	1012	79,5833	74,9999	-5.76%
12	1013	230,068	257,0264	11.72%

```
WITH AverageCalculatedPricePerEntertainer(EntertainerID, YearEngagement,
AverageCalculatedDayPrice) AS (
SELECT EntertainerID, YEAR(StartDate) As YearEngagement, AVG(ContractPrice / (DATEDIFF(DAY,
StartDate, EndDate) + 1)) As AverageCalculatedDayPrice
FROM Engagements
GROUP BY EntertainerID, YEAR(StartDate)),

AverageCalculatedDayPrice2015_2016(EntertainerID, YearEngagement,
AverageCalculatedDayPrice2015, AverageCalculatedDayPrice2016) AS
(SELECT EntertainerID, YearEngagement, AverageCalculatedDayPrice As
AverageCalculatedDayPrice2015, LEAD(AverageCalculatedDayPrice) OVER (Partition BY
EntertainerID ORDER BY EntertainerID)  As AverageCalculatedDayPrice2016
FROM AverageCalculatedPricePerEntertainer)

SELECT EntertainerID, AverageCalculatedDayPrice2015, AverageCalculatedDayPrice2016,
FORMAT((AverageCalculatedDayPrice2016 - AverageCalculatedDayPrice2015) * 1.0 /
AverageCalculatedDayPrice2015, 'P') As RelativeDifference
FROM AverageCalculatedDayPrice2015_2016
WHERE YearEngagement = 2015
```

19. What is total number of males and females per music style. Only take into account StyleStrength = 1. Create the following overview. The image below shows only a part of the resultset.

	StyleID	StyleName	NumberOfFemales	NumberOfMales
1	1	40's Ballroom Music	0	0
2	2	50's Music	0	0
3	3	60's Music	1	5
4	4	70's Music	0	0
5	5	80's Music	0	0
6	6	Country	1	4
7	7	Classical	0	0
8	8	Classic Rock & Roll	0	0
9	9	Rap	0	0
10	10	Contemporary	0	0
11	11	Country Rock	3	2
12	12	Elvis	0	0
13	13	Folk	1	0
14	14	Chamber Music	2	0
15	15	Jazz	2	2

```
(SELECT StyleID, StyleName,
(SELECT COUNT(m.MemberID)
FROM Entertainer_Styles es JOIN Entertainer_Members em ON es.EntertainerID =
em.EntertainerID JOIN Members m ON em.MemberID = m.MemberID
WHERE m.Gender = 'F' AND es.StyleStrength = 1 AND es.StyleID = ms.StyleID) As
NumberOfFemales,
(SELECT COUNT(m.MemberID)
FROM Entertainer_Styles es JOIN Entertainer_Members em ON es.EntertainerID =
em.EntertainerID JOIN Members m ON em.MemberID = m.MemberID
WHERE m.Gender = 'M' AND es.StyleStrength = 1 AND es.StyleID = ms.StyleID) As NumberOfMales
FROM Musical_Styles ms)
```

Extension: Calculate the percentage of females and males per music style

	StyleID	StyleName	RelativeNumberOfFemales	RelativeNumberOfMales
1	3	60's Music	16.67%	83.33%
2	6	Country	20.00%	80.00%
3	11	Country Rock	60.00%	40.00%
4	13	Folk	100.00%	0.00%
5	14	Chamber Music	100.00%	0.00%
6	15	Jazz	50.00%	50.00%
7	19	Rhythm and Blues	40.00%	60.00%
8	20	Show Tunes	100.00%	0.00%
9	22	Top 40 Hits	100.00%	0.00%
10	23	Variety	25.00%	75.00%

```
WITH NumberOfMembersPerGenre (StyleID, StyleName, NumberOfFemales, NumberOfMales) AS
(SELECT StyleID, StyleName,
(SELECT COUNT(m.MemberID)
FROM Entertainer_Styles es JOIN Entertainer_Members em ON es.EntertainerID =
em.EntertainerID JOIN Members m ON em.MemberID = m.MemberID
WHERE m.Gender = 'F' AND es.StyleStrength = 1 AND es.StyleID = ms.StyleID) As
NumberOfFemales,
(SELECT COUNT(m.MemberID)
FROM Entertainer_Styles es JOIN Entertainer_Members em ON es.EntertainerID =
em.EntertainerID JOIN Members m ON em.MemberID = m.MemberID
WHERE m.Gender = 'M' AND es.StyleStrength = 1 AND es.StyleID = ms.StyleID) As NumberOfMales
FROM Musical_Styles ms)

SELECT StyleID, StyleName, FORMAT(NumberOfFemales * 1.0 / (NumberOfFemales + NumberOfMales),
'P') As RelativeNumberOfFemales, FORMAT(NumberOfMales * 1.0 / (NumberOfFemales +
NumberOfMales), 'P') As RelativeNumberOfMales
FROM NumberOfMembersPerGenre
WHERE NumberOfFemales + NumberOfMales > 0
```

Database Programming

Exercise 1

Write a SP add_musical_style to add a new musical style, given the stylename of the new musical style (= parameter).

First check if the stylename doesn't already exist. If so throw an exception.

Use an output parameter to pass the result back to the user: 1 if the insert succeeded and 0 if the insert didn't succeed

Use the following testcode

```
-- Testcode 1: add Dance music => no problem
begin try
	begin transaction
	DECLARE @result tinyint
	EXEC add_musical_style 'Dance music', @result
	print 'Dance music inserted or not: ' + str(@result)
	select * from Musical_Styles
	rollback;
end try
begin catch
	DECLARE @e int;
	SET @e = ERROR_NUMBER();
	PRINT N'Error Procedure = ' + ERROR_PROCEDURE();
	PRINT N'Error Message = ' + ERROR_MESSAGE();

end catch

-- Testcode 2: add Classical => Error because Classical already exists

begin try
	begin transaction
	DECLARE @result tinyint
	EXEC add_musical_style 'Classical', @result
	print 'Classical inserted or not: ' + str(@result)
	select * from Musical_Styles
	rollback
end try
begin catch
	DECLARE @e int;
	SET @e = ERROR_NUMBER();
```

```
	PRINT N'Error Procedure = ' + ERROR_PROCEDURE();
	PRINT N'Error Message = ' + ERROR_MESSAGE();
end catch
```

Solution

```
ALTER PROCEDURE add_musical_style (@stylename varchar(75), @succeeded tinyint out)
AS
BEGIN
-- Check if the stylename already exists
IF EXISTS (SELECT NULL FROM Musical_Styles WHERE StyleName = @stylename)
BEGIN
	RAISERROR('There is already such a musical style', 18, 1)
END

INSERT INTO Musical_Styles
VALUES (@stylename)

SET @succeeded = @@ROWCOUNT

END
```

Exercise 2

Write a SP delete_musical_style to delete a new musical style, given the stylename of the new musical style (= parameter).

First check if the stylename exists. If not so throw an error.

Then check if there are Entertainers with this musical style. If so, throw an error

Then check if there are Customers with this musical style. If so, throw an error

Delete the musical style

Use an output parameter to pass the result back to the user: 1 if the delete succeeded and 0 if the delete didn't succeed

Write testcode to try to delete the following styles: Chamber Music, Elvis, Variety

```
CREATE PROCEDURE delete_musical_style (@stylename varchar(75), @succeeded tinyint out)
AS
BEGIN
-- Check if the stylename exists
IF NOT EXISTS (SELECT NULL FROM Musical_Styles WHERE StyleName = @stylename)
```

```
BEGIN
	RAISERROR('There is no such a musical style', 18, 1)
END

-- What is the id of this musical_style
DECLARE @styleid int
SELECT @styleid = styleid from Musical_Styles WHERE stylename = @stylename

-- Check if there are any entertainers with this musical_style.
-- If so, you can't delete the musical_style
IF EXISTS (SELECT NULL FROM Entertainer_Styles WHERE styleid = @styleid)
BEGIN
	RAISERROR('There are entertainers with this musical style', 18, 1)
END

-- Check if there are any customers with this musical_style.
-- If so, you can't delete the musical_style
IF EXISTS (SELECT NULL FROM Musical_Preferences WHERE styleid = @styleid)
BEGIN
	RAISERROR('There are customers with this musical style', 18, 1)
END

DELETE FROM Musical_Styles
WHERE styleid = @styleid

SET @succeeded = @@ROWCOUNT

END

-- Testcode 1
begin try
	begin transaction
	DECLARE @result tinyint
	EXEC delete_musical_style 'Chamber Music', @result
	print 'Chamber Music deleted or not: ' + str(@result)
	select * from Musical_Styles
	rollback;
end try
begin catch
	DECLARE @e int;
	SET @e = ERROR_NUMBER();
	PRINT N'Error Procedure = ' + ERROR_PROCEDURE();
	PRINT N'Error Message = ' + ERROR_MESSAGE();
```

```
end catch

-- Testcode 2
begin try
	begin transaction
	DECLARE @result tinyint
	EXEC delete_musical_style 'Elvis', @result
	print 'Elvis deleted or not: ' + str(@result)
	select * from Musical_Styles
	rollback;
end try
begin catch
	DECLARE @e int;
	SET @e = ERROR_NUMBER();
	PRINT N'Error Procedure = ' + ERROR_PROCEDURE();
	PRINT N'Error Message = ' + ERROR_MESSAGE();
end catch

-- Testcode 3
begin try
	begin transaction
	DECLARE @result tinyint
	EXEC delete_musical_style 'Variety', @result
	print 'Variety deleted or not: ' + str(@result)
	select * from Musical_Styles
	rollback;
end try
begin catch
	DECLARE @e int;
	SET @e = ERROR_NUMBER();
	PRINT N'Error Procedure = ' + ERROR_PROCEDURE();
	PRINT N'Error Message = ' + ERROR_MESSAGE();
end catch
```

Exercise 3

Write a SP add_musical_preference to add a new musical preference given a customerid, a stylename and a preferenceSeq (= parameters)

Check if the customer exists. If not, throw an error

Check if the stylename exists. If not, throw an error

Check the preferenceSeq: if the customer already has a stylename with this preferenceSeq: this stylename and the subsequent styles get preferenceSeq + 1

Use an output parameter to pass the result back to the user: 1 if the insert succeeded and 0 if the insert didn't succeed

Write testcode for the following cases:

- for customer 10005 add Variety with PreferenceSeq = 3
- for customer 10007 add Variety with PreferenceSeq = 1

```
CREATE PROCEDURE add_musical_preference_2 (@customerid int, @stylename varchar(75), @preferenceSeq smallint, @succeeded tinyint out)
AS
BEGIN
-- Check if the entertainer exists
IF NOT EXISTS (SELECT NULL FROM Customers WHERE customerid = @customerid)
BEGIN
	RAISERROR('There is no such customer', 1, 1)
END

-- Check if the stylename exists
IF NOT EXISTS (SELECT NULL FROM Musical_Styles WHERE StyleName = @stylename)
BEGIN
	RAISERROR('There is no such stylename', 1, 1)
END

-- What is the styleid of the stylename?
DECLARE @styleid int
SELECT @styleid = styleid FROM musical_styles WHERE StyleName = @stylename

-- Update the preferenceSeq of the following styles
UPDATE Musical_Preferences
SET PreferenceSeq = PreferenceSeq + 1
WHERE CustomerID = @customerid and PreferenceSeq >= @preferenceSeq


INSERT INTO Musical_Preferences
VALUES(@customerid, @styleid, @preferenceSeq)

SET @succeeded = @@ROWCOUNT

END

-- Testcode 1
begin try
	begin transaction
```

```
	DECLARE @result tinyint
	EXEC add_musical_preference_2 10005, 'Variety', 3, @result
	print 'Preference added: ' + str(@result)
	select * from Musical_Preferences where customerid = 10005
	rollback;
end try
begin catch
	DECLARE @e int;
	SET @e = ERROR_NUMBER();
	PRINT N'Error Procedure = ' + ERROR_PROCEDURE();
	PRINT N'Error Message = ' + ERROR_MESSAGE();

end catch

-- Testcode 2

begin try
	begin transaction
	DECLARE @result tinyint
	EXEC add_musical_preference_2 10007, 'Variety', 1, @result
	print 'Preference added: ' + str(@result)
	select * from Musical_Preferences where customerid = 10007
	rollback;
end try
begin catch
	DECLARE @e int;
	SET @e = ERROR_NUMBER();
	PRINT N'Error Procedure = ' + ERROR_PROCEDURE();
	PRINT N'Error Message = ' + ERROR_MESSAGE();
end catch
```

Exercise 4

Write a SP add_entertainer_style to add a new Entertainer_Style, given the entertainerid, the stylename and the stylestrength (= parameters)

Check if the given entertainer exists. If not, throw an error

Check if the given stylename exists. If not, throw an error

Check if the entertainer not already has this style. If so, throw an error

Check if the given stylestrength is a value between 1 and 3

If the given stylestrength is already taken for this entertainerid: the other stylestrengths are getting a new value = old value + 1. Make sure there are no more than 3 stylestrengths per entertainer

Use an output parameter to pass the result back to the user: 1 if the insert succeeded and 0 if the insert didn't succeed

Write testcode for the following cases:

- Add entertainer met entertainerid = 1020 + stylename = chamber music + stylestrength = 1
- Add entertainer met entertainerid = 1003 + stylename = disco + stylestrength = 3
- Add entertainer met entertainerid = 1005 + stylename = Jazz + stylestrength = 1
- Add entertainer met entertainerid = 1005 + stylename = Jazz + stylestrength = 4
- Add entertainer met entertainerid = 1005 + stylename = Folk + stylestrength = 2
- Add entertainer met entertainerid = 1003 + stylename = Folk + stylestrength = 3

```
CREATE PROCEDURE add_entertainer_style (@entertainerid int, @stylename varchar(75), @stylestrength smallint, @succeeded tinyint out)
AS
BEGIN
-- Check if the entertainer exists
IF NOT EXISTS (SELECT NULL FROM Entertainers WHERE entertainerid = @entertainerid)
BEGIN
	RAISERROR('There is no such entertainer', 18, 1)
END

-- Check if the stylename exists
IF NOT EXISTS (SELECT NULL FROM Musical_Styles WHERE StyleName = @stylename)
BEGIN
	RAISERROR('There is no such stylename', 18, 1)
END

-- What is the styleid of the stylename?
DECLARE @styleid int
SELECT @styleid = styleid FROM musical_styles WHERE StyleName = @stylename

-- Check if the entertainer not already has this style?
IF EXISTS (SELECT NULL FROM Entertainer_Styles
WHERE EntertainerID = @entertainerid and styleid = @styleid)

BEGIN
	RAISERROR('This style already exists for this entertainer', 18, 1)
```

```
END

-- Check if the stylestrength is a value between 1 and 3
IF @stylestrength NOT BETWEEN 1 and 3
	RAISERROR('The value for stylestrength is not correct', 18, 1)

-- Check if the entertainer already has a style with this stylestrength?
IF EXISTS (SELECT NULL FROM Entertainer_Styles
WHERE EntertainerID = @entertainerid and StyleStrength = @stylestrength)
BEGIN
	UPDATE Entertainer_Styles
	SET StyleStrength = StyleStrength + 1
	WHERE EntertainerID = @entertainerid and Stylestrength >= @stylestrength

	DELETE FROM Entertainer_Styles
	WHERE entertainerid = @entertainerid and StyleStrength > 3
END

INSERT INTO entertainer_styles
VALUES(@entertainerid, @styleid, @stylestrength)


SET @succeeded = @@ROWCOUNT

END

-- Testcode 1: try to add an entertainer that doesn't exist
begin try
	begin transaction
	DECLARE @result tinyint
	EXEC add_entertainer_style 1020, 'Chamber Music', 1, @result
	print 'Tried to add entertainerid = 1020 + stylename = chamber music + stylestrength = 1: ' + str(@result)
	select * from Entertainer_Styles where EntertainerID = 1020
	rollback;
end try
begin catch
	DECLARE @e int;
	SET @e = ERROR_NUMBER();
	PRINT N'Error Procedure = ' + ERROR_PROCEDURE();
	PRINT N'Error Message = ' + ERROR_MESSAGE();
end catch

-- Testcode 2: try to add an entertainerstyle that doesn't exist
```

```
begin try
	begin transaction
	DECLARE @result tinyint
	EXEC add_entertainer_style 1003, 'Disco', 3, @result
	print 'Tried to add entertainerid = 1003 + stylename = disco + stylestrength = 3: ' + str(@result)
	select * from Entertainer_Styles where EntertainerID = 1003
	rollback;
end try
begin catch
	DECLARE @e int;
	SET @e = ERROR_NUMBER();
	PRINT N'Error Procedure = ' + ERROR_PROCEDURE();
	PRINT N'Error Message = ' + ERROR_MESSAGE();
end catch

-- Testcode 3: try to add an entertainerstyle that already exists for this entertainer
begin try
	begin transaction
	DECLARE @result tinyint
	EXEC add_entertainer_style 1005, 'Jazz', 3, @result
	print 'Tried to add entertainerid = 1005 + stylename = Jazz + stylestrength = 1: ' + str(@result)
	select * from Entertainer_Styles where EntertainerID = 1005
	rollback;
end try
begin catch
	DECLARE @e int;
	SET @e = ERROR_NUMBER();
	PRINT N'Error Procedure = ' + ERROR_PROCEDURE();
	PRINT N'Error Message = ' + ERROR_MESSAGE();
end catch

-- Testcode 4: stylestrength is not a value between 1 and 3
begin try
	begin transaction
	DECLARE @result tinyint
	EXEC add_entertainer_style 1005, 'Jazz', 4, @result
	print 'Tried to add entertainerid = 1005 + stylename = Jazz + stylestrength = 4: ' + str(@result)
	select * from Entertainer_Styles where EntertainerID = 1005
	rollback;
end try
begin catch
	DECLARE @e int;
	SET @e = ERROR_NUMBER();
```

```
	PRINT N'Error Procedure = ' + ERROR_PROCEDURE();
	PRINT N'Error Message = ' + ERROR_MESSAGE();
end catch

-- Testcode 5: adding a style with a stylestrength that already exists
begin try
	begin transaction
	DECLARE @result tinyint
	EXEC add_entertainer_style 1005, 'Folk', 2, @result
	print 'Tried to add entertainerid = 1005 + stylename = Folk + stylestrength = 2: ' + str(@result)
	select * from Entertainer_Styles where EntertainerID = 1005
	rollback;
end try
begin catch
	DECLARE @e int;
	SET @e = ERROR_NUMBER();
	PRINT N'Error Procedure = ' + ERROR_PROCEDURE();
	PRINT N'Error Message = ' + ERROR_MESSAGE();
end catch

-- Testcode 6: adding a style with a stylestrength that not already exists
begin try
	begin transaction
	DECLARE @result tinyint
	EXEC add_entertainer_style 1003, 'Folk', 3, @result
	print 'Tried to add entertainerid = 1003 + stylename = Folk + stylestrength = 3: ' + str(@result)
	select * from Entertainer_Styles where EntertainerID = 1003
	rollback;
end try
begin catch
	DECLARE @e int;
	SET @e = ERROR_NUMBER();
	PRINT N'Error Procedure = ' + ERROR_PROCEDURE();
	PRINT N'Error Message = ' + ERROR_MESSAGE();
end catch
```

Exercise 5

List all customers per city, sorted in alphabetical order.

The image below shows only a part of the result.

Messages

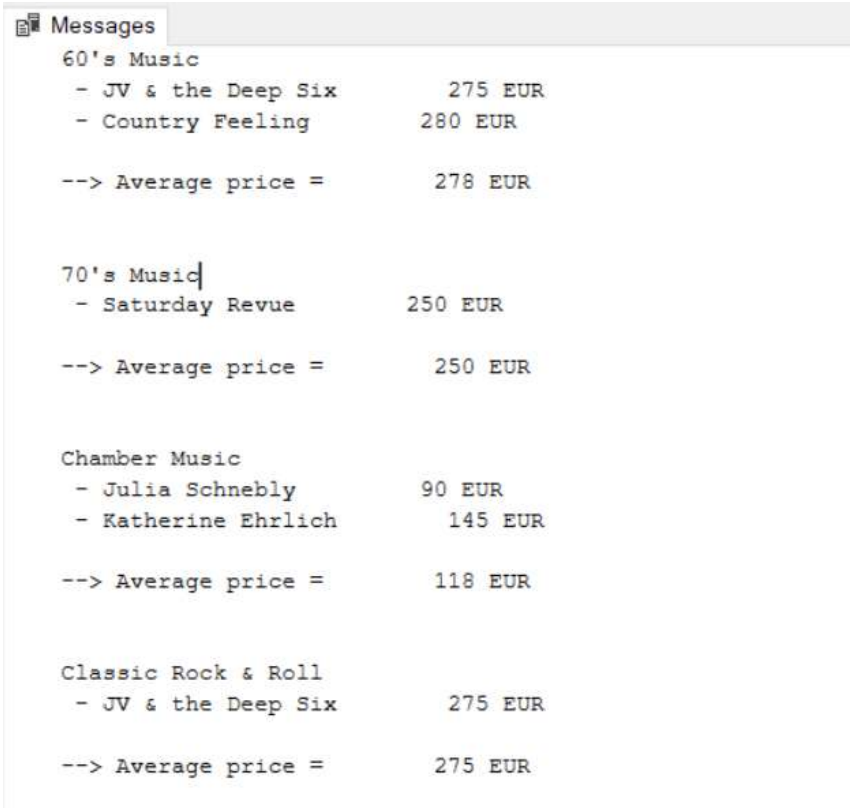
```
- Auburn
    - Elizabeth Hallmark
- Bellevue
    - Liz Smith
    - Sarah Thompson
    - Joyce Smith
    - Louise Johnson
    - Mark Davison
- Kirkland
    - Zachary Johnson
    - Darren Davidson
    - Ben Clothier
- Redmond
    - Tom Wickerath
    - Kerry Patterson
```

```
DECLARE @CustFirstName varchar(25), @CustLastName varchar(25), @CustCity varchar(30), @CustCityPrevious varchar(30)

SET @CustCityPrevious = ''

DECLARE cursor_1 CURSOR FOR
SELECT custFirstName, custLastName, CustCity
FROM Customers
ORDER BY CustCity ASC

OPEN cursor_1
FETCH NEXT FROM cursor_1 INTO @CustFirstName, @CustLastName, @CustCity
WHILE @@FETCH_STATUS = 0
BEGIN
       IF @CustCity <> @CustCityPrevious
       BEGIN
              PRINT ' - ' + @CustCity
       END
       PRINT '    - ' + @CustFirstName + ' ' + @CustLastName
       SET @CustCityPrevious = @CustCity
```

```
      FETCH NEXT FROM cursor_1 INTO @CustFirstName, @CustLastName, @CustCity
END

CLOSE cursor_1
DEALLOCATE cursor_1
```

Exercise 6

List all artists per Entertainer_Styles, sorted from cheaper to more expensive.
If there is an Entertainer_Style for which there are no artists, it should not appear in the overview
Also calculate the average price at the end for each Entertainer_Style
The image below shows only a part of the result.



```
DECLARE @StyleID int, @StyleName varchar(75), @NumberOfEntertainers int
DECLARE @EntertainerID int, @EntStageName varchar(50), @EntPricePerDay money
DECLARE @avgPrice money

SET @avgPrice = 0

DECLARE cursor_1 CURSOR FOR
SELECT ms.StyleID, ms.StyleName, count(DISTINCT es.EntertainerID) As NumberOfEntertainers
FROM Musical_Styles ms JOIN Entertainer_Styles es ON ms.StyleID = es.StyleID
GROUP BY ms.StyleID, ms.StyleName
ORDER BY ms.StyleName ASC

OPEN cursor_1
FETCH NEXT FROM cursor_1 INTO @StyleID, @StyleName, @NumberOfEntertainers
WHILE @@FETCH_STATUS = 0
BEGIN
	IF @NumberOfEntertainers >= 1
		BEGIN
			PRINT @StyleName
			DECLARE cursor_2 CURSOR FOR
			SELECT e.EntertainerID, e.EntStageName, e.EntPricePerDay
			FROM Entertainers e JOIN Entertainer_Styles es ON e.EntertainerID = es.EntertainerID
			WHERE es.StyleID = @StyleID
			ORDER BY e.EntPricePerDay ASC

			SET @avgPrice = 0

			OPEN cursor_2
			FETCH NEXT FROM cursor_2 INTO @EntertainerID, @EntStageName, @EntPricePerDay
			WHILE @@FETCH_STATUS = 0
				BEGIN
					PRINT ' - ' + @EntStageName + ' ' + str(@EntPricePerDay) + ' EUR'
					SET @avgPrice = @avgPrice + @EntPricePerDay
					FETCH NEXT FROM cursor_2 INTO @EntertainerID, @EntStageName, @EntPricePerDay
				END
			CLOSE cursor_2

			DEALLOCATE cursor_2

			PRINT ''
			PRINT '--> Average price = ' + str(@avgPrice / @NumberOfEntertainers) + ' EUR'
			PRINT ''
			PRINT ''
```

```
                FETCH NEXT FROM cursor_1 INTO @StyleID, @StyleName, @NumberOfEntertainers
            END
END

CLOSE cursor_1

DEALLOCATE cursor_1
```

Exercise 7

Give per Member the number of Entertainers a Member is part of and the list of the Entertainers of which a Member is a part of
Notice the difference between e.g. Robert Brown is part of 2 Entertainer**s** and Steve Davidson is part of 1 Entertainer
The image below shows only a part of the result.

Messages

```
Robert Brown is part of          2 Entertainers
 - Coldwater Cattle Company
 - Country Feeling
Janice Davidson is part of         2 Entertainers
 - Carol Peacock Trio
 - Modern Dance
Jeffrey Davidson is part of         2 Entertainers
 - JV & the Deep Six
 - Modern Dance
Steve Davidson is part of          1 Entertainer
 - Modern Dance
Susan Davidson is part of          2 Entertainers
 - Saturday Revue
 - Susan McLain
Gary Hallmark is part of          2 Entertainers
 - Country Feeling
 - JV & the Deep Six
Michael Hernandez is part of          3 Entertainers
 - Jazz Persuasion
 - Modern Dance
 - Topazz
George Johnson is part of          2 Entertainers
 - Caroline Coie Cuartet
 - Country Feeling
Julia Johnson is part of          1 Entertainer
 - Julia Schnebly
Megan Johnson is part of          1 Entertainer
 - Coldwater Cattle Company
Angel Kennedy is part of          1 Entertainer
 - Jazz Persuasion
```

```
DECLARE @MemberID int, @MbrFirstName varchar(25), @MbrLastName varchar(25), @NumberOfEntertainers int
DECLARE @EntertainerID int, @EntStageName varchar(50)


DECLARE cursor_1 CURSOR FOR
SELECT m.MemberID, m.MbrFirstName, m.MbrLastName, count(DISTINCT em.EntertainerID) As NumberOfEntertainers
FROM Members m JOIN Entertainer_Members em ON m.MemberID = em.MemberID
GROUP BY m.MemberID, m.MbrFirstName, m.MbrLastName
ORDER BY MbrLastName ASC, MbrFirstName ASC

OPEN cursor_1
FETCH NEXT FROM cursor_1 INTO @MemberID, @MbrFirstName, @MbrLastName, @NumberOfEntertainers
WHILE @@FETCH_STATUS = 0
BEGIN
	IF @NumberOfEntertainers > 1
		PRINT @MbrFirstName + ' ' + @MbrLastName + ' is part of ' + str(@NumberOfEntertainers) + ' Entertainers'
	ELSE
		PRINT @MbrFirstName + ' ' + @MbrLastName + ' is part of ' + str(@NumberOfEntertainers) + ' Entertainer'
	DECLARE cursor_2 CURSOR FOR
	SELECT e.EntertainerID, e.EntStageName
	FROM Entertainers e JOIN Entertainer_Members em ON e.EntertainerID = em.EntertainerID
	WHERE em.MemberID = @MemberID
	ORDER BY e.EntStageName

	OPEN cursor_2
	FETCH NEXT FROM cursor_2 INTO @EntertainerID, @EntStageName
	WHILE @@FETCH_STATUS = 0
	BEGIN
		PRINT ' - ' + @EntStageName
		FETCH NEXT FROM cursor_2 INTO @EntertainerID, @EntStageName
	END

	CLOSE cursor_2

	DEALLOCATE cursor_2

	FETCH NEXT FROM cursor_1 INTO @MemberID, @MbrFirstName, @MbrLastName, @NumberOfEntertainers
END

CLOSE cursor_1

DEALLOCATE cursor_1
```

Exercise 8

List the Engagements per Agent and what the Agent has earned per year
To add the the small sentences like 'Total salary for 2015 is 35510', you have to add an IF statement after `FETCH NEXT FROM cursor_2 INTO …`

The image below shows only a part of the result.

Messages

```
        1 William Thompson Yearly salary =      35000
 - year =       2015
 * EngagementNumber =         13 StartDate = 17 Sep 2015 ContractPrice =        770 CustomerID =      10003 EntertainerID =       1006 Commission = EUR         31

 * EngagementNumber =         14 StartDate = 24 Sep 2015 ContractPrice =       2750 CustomerID =      10001 EntertainerID =       1008 Commission = EUR        110

 * EngagementNumber =         15 StartDate = 24 Sep 2015 ContractPrice =        770 CustomerID =      10007 EntertainerID =       1013 Commission = EUR         31

 * EngagementNumber =         21 StartDate = 30 Sep 2015 ContractPrice =       1490 CustomerID =      10005 EntertainerID =       1003 Commission = EUR         60

 * EngagementNumber =         42 StartDate = 20 Oct 2015 ContractPrice =       2150 CustomerID =      10002 EntertainerID =       1013 Commission = EUR         86

 * EngagementNumber =         45 StartDate = 21 Oct 2015 ContractPrice =        530 CustomerID =      10015 EntertainerID =       1012 Commission = EUR         21

 * EngagementNumber =         48 StartDate = 05 Nov 2015 ContractPrice =        950 CustomerID =      10002 EntertainerID =       1007 Commission = EUR         38

 * EngagementNumber =         71 StartDate = 22 Dec 2015 ContractPrice =       1670 CustomerID =      10002 EntertainerID =       1003 Commission = EUR         67

 * EngagementNumber =         68 StartDate = 24 Dec 2015 ContractPrice =       1670 CustomerID =      10009 EntertainerID =       1005 Commission = EUR         67
Total salary for       2015 is      35510

 - year =       2016
 * EngagementNumber =         74 StartDate = 01 Jan 2016 ContractPrice =        590 CustomerID =      10004 EntertainerID =       1005 Commission = EUR         24

 * EngagementNumber =        111 StartDate = 12 Feb 2016 ContractPrice =        185 CustomerID =      10012 EntertainerID =       1004 Commission = EUR          7

 * EngagementNumber =        118 StartDate = 18 Feb 2016 ContractPrice =        350 CustomerID =      10014 EntertainerID =       1010 Commission = EUR         14

 * EngagementNumber =        114 StartDate = 19 Feb 2016 ContractPrice =       1550 CustomerID =      10005 EntertainerID =       1002 Commission = EUR         62

 * EngagementNumber =        124 StartDate = 23 Feb 2016 ContractPrice =       1850 CustomerID =      10006 EntertainerID =       1008 Commission = EUR         74

 * EngagementNumber =        123 StartDate = 25 Feb 2016 ContractPrice =        770 CustomerID =      10013 EntertainerID =       1001 Commission = EUR         31

 * EngagementNumber =        131 StartDate = 03 Mar 2016 ContractPrice =       1850 CustomerID =      10014 EntertainerID =       1003 Commission = EUR         74
Total salary for       2016 is      35286


        2 Scott Johnson Yearly salary =      27000
 - year =       2015
 * EngagementNumber =          9 StartDate = 16 Sep 2015 ContractPrice =       1370 CustomerID =      10010 EntertainerID =       1010 Commission = EUR         55

 * EngagementNumber =         58 StartDate = 01 Dec 2015 ContractPrice =        770 CustomerID =      10001 EntertainerID =       1002 Commission = EUR         31

 * EngagementNumber =         62 StartDate = 09 Dec 2015 ContractPrice =        500 CustomerID =      10003 EntertainerID =       1005 Commission = EUR         20
Total salary for       2015 is      27106

 - year =       2016
 * EngagementNumber =         83 StartDate = 06 Jan 2016 ContractPrice =        650 CustomerID =      10010 EntertainerID =       1006 Commission = EUR         26

 * EngagementNumber =         98 StartDate = 20 Jan 2016 ContractPrice =       2930 CustomerID =      10012 EntertainerID =       1010 Commission = EUR        117

 * EngagementNumber =        119 StartDate = 19 Feb 2016 ContractPrice =        500 CustomerID =      10012 EntertainerID =       1004 Commission = EUR         20
Total salary for       2016 is      27163
```

```
DECLARE @AgentID int, @AgtFirstName varchar(25), @AgtLastName varchar(25), @Salary money,  @CommissionRate real
DECLARE @EngagementNumber int, @StartDate date, @ContractPrice money, @CustomerID int, @EntertainerID int
DECLARE @year int, @previousYear int, @yearSalary money

SET @yearSalary = 0
SET @year = 0
SET @previousYear = 0


DECLARE cursor_1 CURSOR FOR
SELECT AgentID, AgtFirstName, AgtLastName, Salary,  CommissionRate
FROM Agents
ORDER BY AgentID ASC

OPEN cursor_1
FETCH NEXT FROM cursor_1 INTO @AgentID, @AgtFirstName, @AgtLastName, @Salary,  @CommissionRate
WHILE @@FETCH_STATUS = 0
BEGIN
	PRINT str(@AgentID) + ' ' + @AgtFirstName + ' ' + @AgtLastName + ' Yearly salary = ' + str(@Salary)



	DECLARE cursor_2 CURSOR FOR
	SELECT EngagementNumber, StartDate, ContractPrice, CustomerID, EntertainerID
	FROM Engagements
	WHERE AgentID = @AgentID
	ORDER BY StartDate ASC

	OPEN cursor_2
	FETCH NEXT FROM cursor_2 INTO @EngagementNumber, @StartDate, @ContractPrice, @CustomerID, @EntertainerID
	WHILE @@FETCH_STATUS = 0
	BEGIN

		IF @Year != YEAR(@StartDate)
			BEGIN
				PRINT ' - year = ' + str(Year(@StartDate))
				SET @Year = Year(@StartDate)
				SET @yearSalary = @Salary
			END
```

```
                    PRINT ' * EngagementNumber = ' + str(@EngagementNumber) + ' StartDate = ' + CONVERT(VARCHAR, @StartDate, 113) + '
ContractPrice = ' + str(@ContractPrice) + ' CustomerID = ' + str(@CustomerID) + ' EntertainerID = ' + str(@EntertainerID) + ' Commission =
EUR ' + str(@ContractPrice * @CommissionRate)
                    SET @yearSalary = @yearSalary + @ContractPrice * @CommissionRate
                    FETCH NEXT FROM cursor_2 INTO @EngagementNumber, @StartDate, @ContractPrice, @CustomerID, @EntertainerID

                    IF @@FETCH_STATUS != 0 OR @Year != YEAR(@StartDate)
                           PRINT 'Total salary for ' + str(@Year) + ' is ' + str(@yearSalary)
                           PRINT ' '

       END

       CLOSE cursor_2

       DEALLOCATE cursor_2

       PRINT ''


       FETCH NEXT FROM cursor_1 INTO @AgentID, @AgtFirstName, @AgtLastName, @Salary,  @CommissionRate


END

CLOSE cursor_1

DEALLOCATE cursor_1
```

Exercise 9

If there is an update of the Musical_Preferences table where the PreferenceSeq is adjusted, it must be checked whether the new PreferenceSeq is an allowed value (between 1 and 3)

If not, the transaction must be rolled back and an error should be thrown

In the testcode below, CustomerID = 10001 and StyleID = 10 get's PreferenceSeq = 10

```
begin try
       begin transaction

       UPDATE Musical_Preferences
       SET PreferenceSeq = 10
       WHERE CustomerID = 10001 and StyleID = 10
```

```
	print 'Update PreferenceSeq = 10 for CustomerID = 10001 and styleID = 10'

	select * from Musical_Preferences
	WHERE CustomerID = 10001

	rollback;
end try
begin catch
	DECLARE @e int;
	SET @e = ERROR_NUMBER();
	PRINT N'Error Procedure = ' + ERROR_PROCEDURE();
	PRINT N'Error Message = ' + ERROR_MESSAGE();
end catch

CREATE TRIGGER trigger1 ON Musical_Preferences FOR update
AS
DECLARE @PreferenceSeq AS int
SELECT @PreferenceSeq = PreferenceSeq FROM inserted
IF @PreferenceSeq < 1 or @PreferenceSeq > 3
	BEGIN
		ROLLBACK TRANSACTION
		RAISERROR ('The PreferenceSeq has to be a value between 1 and 3', 18,1)
	END

-- Testcode: Update CustomerID = 10001 and StyleID = 10 to PreferenceSeq = 10
-- Testcode: Update CustomerID = 10001 and StyleID = 10 to PreferenceSeq = 3

begin try
	begin transaction

	UPDATE Musical_Preferences
	SET PreferenceSeq = 3
	WHERE CustomerID = 10001 and StyleID = 10

	print 'Update PreferenceSeq = 10 for CustomerID = 10001 and styleID = 10'

	select * from Musical_Preferences
	WHERE CustomerID = 10001

	rollback;
end try
begin catch
```

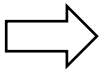
```
        DECLARE @e int;
        SET @e = ERROR_NUMBER();
        PRINT N'Error Procedure = ' + ERROR_PROCEDURE();
        PRINT N'Error Message = ' + ERROR_MESSAGE();
end catch
```

Exercise 10

If a record is removed from the Musical_Preferences table for a specific customer (for example CustomerID = 10007 + StyleID = 4 + PreferenceSeq = 2), then the PreferenceSeq (for example with value 3) of the subsequent records should be decreased by 1

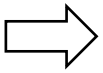
Write testcode for the following examples

CustomerID	StyleID	PreferenceSeq
~~10007~~	~~4~~	~~2~~
10007	8	1
10007	19	3



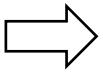
CustomerID	StyleID	PreferenceSeq
10007	8	1
10007	19	2

CustomerID	StyleID	PreferenceSeq
10007	4	2
~~10007~~	~~8~~	~~1~~
10007	19	3



CustomerID	StyleID	PreferenceSeq
10007	4	1
10007	19	2

CustomerID	StyleID	PreferenceSeq
10007	4	2
10007	8	1
~~10007~~	~~19~~	~~3~~



CustomerID	StyleID	PreferenceSeq
10007	4	2
10007	19	1

```
CREATE TRIGGER trigger3 ON Musical_Preferences FOR delete
```

```
AS
DECLARE @CustomerID int, @PreferenceSeq int
SELECT @CustomerID = CustomerID, @PreferenceSeq = PreferenceSeq FROM deleted
UPDATE Musical_Preferences SET PreferenceSeq = PreferenceSeq - 1
WHERE CustomerID = @CustomerID AND PreferenceSeq > @PreferenceSeq

-- Testcode: Update CustomerID = 10007 and StyleID = 8
begin try
	begin transaction

	delete from Musical_Preferences
	WHERE CustomerID = 10007 and StyleID = 19

	select * from Musical_Preferences
	WHERE CustomerID = 10007

	rollback;
end try
begin catch
	DECLARE @e int;
	SET @e = ERROR_NUMBER();
	PRINT N'Error Procedure = ' + ERROR_PROCEDURE();
	PRINT N'Error Message = ' + ERROR_MESSAGE();
end catch
```

Exercise 11

If a record is inserted into the Musical_Preferences table for a specific customer (for example CustomerID = 10007 + StyleID = 5 + PreferenceSeq = 2), then the PreferenceSeq (for example with value 2 and 3) of the subsequent records should be increased by 1. All records with PreferenceSeq > 3 should be removed.

Is it useful to check whether the new combination of CustomerID and StyleID already exists in the table?

Write testcode for the following examples

CustomerID	StyleID	PreferenceSeq		CustomerID	StyleID	PreferenceSeq
10007	4	2		10007	4	3
10007	8	1	⇨	10007	8	1
10007	19	3		~~10007~~	~~19~~	~~3~~

10007	5	2	10007	5	2

CustomerID	StyleID	PreferenceSeq	CustomerID	StyleID	PreferenceSeq
10007	4	2	10007	4	2
10007	8	1	10007	8	1
10007	19	3	~~10007~~	~~19~~	~~3~~
10007	5	3	10007	5	3

CustomerID	StyleID	PreferenceSeq	CustomerID	StyleID	PreferenceSeq
10007	4	2	10007	4	3
10007	8	1	10007	8	2
10007	19	3	~~10007~~	~~19~~	~~3~~
10007	5	1	10007	5	1

```
CREATE TRIGGER trigger5 ON Musical_Preferences FOR INSERT
AS
DECLARE @CustomerID int, @StyleID int, @PreferenceSeq int
SELECT @CustomerID = CustomerID, @StyleID = StyleID, @PreferenceSeq = PreferenceSeq FROM inserted

UPDATE Musical_Preferences
SET PreferenceSeq = PreferenceSeq + 1
WHERE CustomerID = @CustomerID and PreferenceSeq >= @PreferenceSeq and (StyleID <> @StyleID)

DELETE FROM Musical_Preferences
WHERE CustomerID = @CustomerID and PreferenceSeq > 3

-- Testcode

begin try
       begin transaction
```

```
	INSERT INTO Musical_Preferences
	VALUES (10007, 5, 1)

	print 'INSERT CustomerID = 10007 + StyleID = 5 + PreferenceSeq = 1'

	select * from Musical_Preferences
	WHERE CustomerID = 10007

	rollback;
end try
begin catch
	DECLARE @e int;
	SET @e = ERROR_NUMBER();
	PRINT N'Error Procedure = ' + ERROR_PROCEDURE();
	PRINT N'Error Message = ' + ERROR_MESSAGE();
end catch
```

Exercise 12

When a Member is added to an Entertainer, then it must be ensured that the EntPricePerDay is adjusted accordingly so that the price per member remains the same.

E.g.: Entertainer 1002 now has 2 members and the EntPricePerDay is 120 EUR. If an extra member is added, the EntPricePerDay should become 180 EUR.

```
CREATE TRIGGER trigger4 ON Entertainer_Members FOR insert
AS
DECLARE @EntertainerID int
SELECT @EntertainerID = EntertainerID FROM inserted

DECLARE @NewNumberOfMembers int
SELECT @NewNumberOfMembers = COUNT(distinct memberid)
FROM Entertainer_Members
WHERE EntertainerID = @EntertainerID

print '@NewNumberOfMembers = ' + str(@NewNumberOfMembers)

DECLARE @EntPricePerDay money
SELECT @EntPricePerDay = EntPricePerday
FROM Entertainers
WHERE EntertainerID = @EntertainerID

print '@EntPricePerDay = ' + str(@EntPricePerDay)
```

```
DECLARE @NewEntPricePerDay money
SET @NewEntPricePerDay = @EntPricePerDay / (@NewNumberOfMembers - 1) * @NewNumberOfMembers

print '@NewEntPricePerDay = ' + str(@NewEntPricePerDay)

UPDATE Entertainers
SET EntPricePerDay = @NewEntPricePerDay
WHERE EntertainerID = @EntertainerID

-- Testcode: Try to add MemberID 106 to EntertainerID 1002
begin try
	begin transaction

	insert into Entertainer_Members(EntertainerID, MemberID, Status)
	VALUES(1002, 106, 2)

	select * from Entertainer_Members
	WHERE EntertainerID = 1002

	rollback;
end try
begin catch
	DECLARE @e int;
	SET @e = ERROR_NUMBER();
	PRINT N'Error Procedure = ' + ERROR_PROCEDURE();
	PRINT N'Error Message = ' + ERROR_MESSAGE();
end catch
```