Prof. Dr.-Ing. Thomas Kudraß
HTWK Leipzig, F IMN

Lösungshinweise zu den SQL-Auswertung auf der DB *iw_shop* (Übungsblatt 6 – 9)

Übungsblatt 6

1. Erste Selektionen zur Ermittlung der wichtigsten Kennzahlen

a) Ermitteln Sie die Anzahl der Kunden.
b) Ermitteln Sie die Anzahl der unterschiedlichen Kunden.
c) Ermitteln Sie die Anzahl der Bestellungen.
d) Ermitteln Sie die Gesamtanzahl aller bestellten Artikel.
e) Ermitteln Sie den Warenbruttowert aller bestellten Artikel.
f) Ermitteln Sie den Warennettowert (ohne Mehrwertsteuer) aller bestellten Artikel.

2. Erstellung von Reports auf Jahres- und Monatsbasis

a) Stellen Sie die in 2 b) bis d) sowie 2 f) berechneten Kennzahlen in einem Report zusammen.

```
SELECT COUNT (distinct customerNo) Kunden,
       COUNT (distinct orderNo) Bestellungen,
       SUM (s.quantity) Artikelmenge,
       SUM (s.amount * s.quantity) Nettosumme
FROM [dbo].[iw_customer]
WHERE s.type = 2
```

b) Geben Sie die Nettosumme dabei mit zwei Stellen nach dem Komma aus.
Hinweis: Funktion `cast ... as <type>` verwenden.

```
... cast (SUM (s.amount * s.quantity) as decimal(10,2)) Nettosumme
...
```

c) Erzeugen Sie einen detaillierteren Bericht auf Jahresbasis. Legen Sie dabei das Verarbeitungsdatum der Bestellungen (*postingdate*) zugrunde.
Hinweis: Funktion `datepart(yyyy,<date_attr>)` verwenden.

```
SELECT DATEPART (yyyy,s.postingDate) Jahr,
       COUNT (distinct customerNo) Kunden,
       COUNT (distinct orderNo) Bestellungen,
       SUM (s.quantity) Artikelmenge,
       SUM (s.amount * s.quantity) Nettosumme
FROM [dbo].[iw_customer]
WHERE s.type = 2
GROUP BY DATEPART (yyyy,s.postingDate)
```

d) Verfeinern Sie den Bericht aus 3 c), indem Sie die Monate hinzunehmen. Geben Sie die Ergebnisse geordnet nach Jahren und Monaten aus.
Hinweis: Funktion `datepart(mm,<date_attr>)` verwenden.

e) Erweitern Sie den Bericht um zwei weitere Kennzahlen: durchschnittlicher Warenkorbwert und Anzahl der Artikel pro Warenkorb. Der durchschnittliche Warenkorbwert

berechnet sich aus der Nettosumme dividiert durch die Anzahl der Bestellungen. Die Warenkorbgröße berechnet sich aus der Artikelmenge dividiert durch die Anzahl der Bestellungen.
Hinweis: Anfrage schachteln.

```
SELECT a.Jahr, a.Monat, a.Kunden, a.Bestellungen,
a.Artikelmenge, a.Nettosumme,
(a.Nettosumme/a.Bestellungen) Warenkorb,
(a.Artikelmenge/a.Bestellungen) Artikel_WK
FROM
-- Inhalte/Berechnungen aus der Unterabfrage
(SELECT DATEPART (yyyy,s.postingDate) Jahr,
DATEPART (mm,s.postingDate) Monat,
COUNT (distinct s.customerNo) Kunden, -- Kundenkonten
COUNT (distinct s.orderNo) Bestellungen,
SUM (s.quantity) Artikelmenge,
cast (SUM (s.amount * s.quantity) as decimal(10,2)) Nettosumme
FROM [dbo].[iw_sales]s
WHERE s.type = 2
GROUP BY DATEPART (yyyy,s.postingDate),DATEPART (mm,s.postingDate))a
GROUP BY a.Jahr, a.Monat, a.Kunden, a.Bestellungen
-- Alle Werte außer den neuen Berechnungen in die GROUP BY-KLausel
ORDER BY a.Jahr, a.Monat  xxx
```

3. Erstellung von Reports auf Wochenbasis

a) Modifizieren Sie die Anfrage 2 e), indem Sie die Kennzahlen auf Wochenbasis ausgeben.
Hinweis: Funktion `datepart (week,<data_attr>)` verwenden.

b) Verwenden Sie anstelle des Parameters `week` den Parameter `isowk`, so dass die Wochennummern auf Basis des ISO-Wochensystems gebildet werden. Überprüfen Sie die Wirkungsweise für Datensätze mit einem *postingdate* zwischen 28.12.2010 und 31.01.2011

Bei einer ISO-Week-Numerierung hat das Jahr entweder 52 oder 53 Wochen, somit 364 oder 371 Tage. ISO-Wochen beginnen immer am Montag. Jede Woche wird dem (gregorianischen) Kalenderjahr zugeordnet, auf das der Donnerstag fällt. Die ISO-Week 1 enthält immer den 04. Januar.

4. Retouren-Daten

a) Erzeugen Sie eine Auswertung der reinen Retouren-Daten auf Monatsbasis (beginnend mit Januar 2011) mit folgenden Informationen: Anzahl der Retourenpakete, Anzahl der Kunden, Anzahl der retournierten Artikel, Bruttowert der Retouren (inkl. MWSt.), Nettowert der Retouren (ohne MWSt.)

b) Die kaufmännische Retourenquote wird auf Basis eines Abrechnungszeitraums betrachtet. Hier wird gegenübergestellt, wie hoch der monatliche Bestellwert im Vergleich zum Wert der im Monat eingegangenen Retouren ist. Führen Sie eine entsprechende Auswertung durch für die ersten sechs Monate des Jahres 2011.

c) Ermitteln Sie die echte Retourenquote, die sich auf den Bestellmonat bezieht. Hierzu ist ein Join erforderlich der Retourendaten mit den Bestelldaten, der auch die Artikelnummer betrifft. Berechnen Sie wiederum die Retourenquoten und vergleichen Sie kaufmännische und echte Retourenquote.

```sql
SELECT b.Jahr, b.Monat,
cast (r.r_bruttowert/b.bruttowert * 100 as decimal(10,2)) RQ_Bruttowert,
cast (r.r_nettowert/b.nettowert * 100 as decimal(10,2)) RQ_Nettowert
FROM
-- Erzeugung einer temp. Tabelle b mit Bestelldaten pro Jahr-Monat
(SELECT DATEPART (yyyy,s.postingdate) Jahr, DATEPART(mm,s.postingdate)
Monat,
SUM (s.line_amount) Bruttowert,
SUM (s.amount * s.quantity) Nettowert
FROM iw_sales s
WHERE s.type = 2 and s.postingDate >= '01.01.2011'
GROUP BY DATEPART(yyyy,s.postingdate), DATEPART(mm,s.postingdate)) b,
-- Erzeugung einer temp. Tabelle r mit Retourendaten pro Jahr-Monat
-- Join über orderNo und IWAN (Artikel) erforderlich
(SELECT DATEPART(yyyy,s.postingdate) Jahr, DATEPART(mm,s.postingDate)Monat,
SUM (rl.vat_line_amount) R_Bruttowert,
SUM (rl.line_amount) R_Nettowert
FROM [dbo].[iw_sales]s,[dbo].[iw_return_line]rl,[dbo].[iw_return_header]rh
WHERE s.type = 2 and s.postingDate >= '01.01.2011'
and s.orderNo = rh.orderNo
and rh.returnNo = rl.returnNo
and s.IWAN= rl.IWAN
and rl.type = 2
GROUP BY DATEPART(yyyy,s.postingdate), DATEPART(mm,s.postingDate)) r
WHERE b.Jahr = r.Jahr and b.Monat = r.Monat
ORDER BY b.Jahr, b.Monat
```

Übungsblatt 7

1. Analyse von Unique Kunden

Um unique Kunden zu erkennen, könnte man die persönlichen Daten, wie Name, Geschlecht, Geburtsdatum und Wohnort, miteinander abgleichen. In der Praxis eines Online-Shops gibt es dafür schon eine Funktionalität und zwar die Bonitätsprüfung. Dabei wird jede neue Kundenadresse geprüft und die Bonität ermittelt, die Einfluss hat auf mögliche Zahlarten (Vorkasse, Rechnung etc.). Somit haben auch Gastkäufer mit unterschiedlichen Kundennummern nur eine *riskID*.

a) Vergleichen Sie die Anzahl der Kundennummern mit der Anzahl der unterschiedlichen riskIDs.

```
SELECT COUNT (distinct customerNo)
FROM [dbo].[iw_customer]

SELECT COUNT (distinct riskID)
FROM [dbo].[iw_customer]
```

247 065 vs. 107 677

b) Ermitteln Sie die durchschnittliche Anzahl der Konten pro Kunde.

```
SELECT cast(a.Konten/a.Kunden as decimal(10,2))
-- Ausgabe als 2-stelliger Dezimalwert
FROM
(SELECT
cast(count(distinct [customerNo])as numeric) Konten,
-- Ergebnis als numerischer Wert
cast(count(distinct [riskID])as numeric) Kunden
-- Ergebnis als numerischer Wert
FROM [dbo].[iw_customer])a
```

Ergebnis: 2.29
Typkonvertierung notwendig für Rückgabewerte der count-Funktion

c) Erzeugen Sie einen KPI-Report auf der Basis uniquer Kunden mit folgenden Informationen:
Jahr, Monat, Kunden, Kundenkonten, Bestellungen, Artikelmenge, Nettosumme, Warenkorb, Artikel_WK

Vergleiche Übungsblatt 6, Lösung Aufgabe 2 e)

d) Geben Sie im Report Zwischenzeilen pro Jahr (für die verdichtete Auswertung auf Jahresbasis) aus sowie eine Zeile für die Ergebnisse im gesamten Zeitraum (Roll-Up).

Wie c) aber mit rollup

```
GROUP BY ROLLUP (DATEPART (yyyy,s.postingDate),DATEPART ...
```

Ersetzen der Ausgabe NULL für Zwischensummen (pro Jahr) bzw. Gesamtsumme:

```
...
-- Inhalte/Berechnungen aus der Unterabfrage
(SELECT
```

```
 CASE WHEN grouping(DATEPART (yyyy,s.postingDate)) = 1
  THEN cast('Alle' AS varchar)
  ELSE cast(DATEPART (yyyy,s.postingDate) as varchar)
 END Jahr,
 CASE WHEN grouping(DATEPART (mm,s.postingDate)) = 1
  THEN cast('Alle' AS varchar)
  ELSE cast(DATEPART (mm,s.postingDate) as varchar)
 END Monat,
...
GROUP BY ROLLUP (DATEPART (yyyy,s.postingDate),DATEPART ...
ORDER BY a.Jahr, right('0' + cast(a.Monat as varchar(2)),2)
```

e) Erweitern Sie den Report, so dass Sie auch die Ergebnisse auf Wochenbasis ausgeben können (Drill-Down).

```
...
CASE WHEN grouping(DATEPART (isowk,s.postingDate)) = 1
...
```

2. Analyse von Neu- und Bestandskunden

Ein Kunde ist in dem Monat Neukunde, wenn es sich um seine erste Bestellung handelt. Ein Bestandskunde besitzt mindestens einen weiteren Eintrag mit einem jüngeren Bestelldatum in der Datenbank.

a) Ermitteln Sie die Anzahl der hinzugekommenen Neukunden pro Monat.

Identifizierung von Neukunden

```
SELECT c.riskID, -- Eindeutige Kundennummer (riskID)
MIN(s.orderDate)firstOrder -- Erstes Datum (Alias = firstOrder)
FROM [dbo].[iw_sales]s,[dbo].[iw_customer]c
WHERE s.customerNo = c.customerNo --  Join der Tabellen
GROUP BY c.riskID -- Gruppierung nach riskID
```

Neukunden pro Monat

```
SELECT
DATEPART (yyyy,a.firstOrder) Jahr,
DATEPART (mm,a.firstOrder) Monat,
COUNT (distinct a.riskID)Neukunden
FROM ...
-- Berechnung der Neukunden in Tabelle a und Gruppierung nach firstOrder
```

b) Erstellen Sie einen Bericht für Neukunden. Darin sollten enthalten sein:
Jahr, Monat, Neukunden, Bestellungen, Artikelmenge, Gesamtsumme (netto)

```
SELECT
DATEPART (yyyy,a.[Datum])Jahr,DATEPART (mm,a.[Datum])Monat,
COUNT (distinct a.riskID)Neukunden,
COUNT (distinct a.[Bestellung])Bestellungen,
cast (SUM (a.Artikel)as decimal(10,0))Artikelmenge,
cast (SUM (a.Nettobetrag)as decimal(10,2))Gesamtsumme
FROM ...
-- Subquery: Bestellungen eines Kunden a
-- Subquery: Erstbestellungen eines Kunden b
WHERE a.riskID = b.riskID  -- Join von Abfrage a und Abfrage b
and a.Datum = b.firstOrder
```

```
GROUP BY DATEPART (yyyy,a.[Datum]),DATEPART (mm,a.[Datum])
ORDER BY DATEPART (yyyy,a.[Datum]),DATEPART (mm,a.[Datum])
```

c) Ermitteln Sie die Anzahl der Bestandskunden pro Monat.

Analog zur Lösung 2 b) mit Join der Subqueries a und b
allerdings mit Bedingung `a.postingDate > b.firstOrder`

d) Wie kann man die Anzahl der Gesamtkunden ermitteln.

Die Bedingung für die Neukunden lautete: `and a.Datum = b.Erstbestellung`
Die Bedingung für die Bestandskunden lautete: `and a.Datum > b.Erstbestellung`
Um diese Bedingungen außer Kraft zu setzen, werden sie auskommentiert
Siehe Abfrage 1.c)
Addition Neu- und Bestandskunden stimmt für Bestellungen und Artikel zu 100%, nicht aber für die Gesamtanzahl an Kunden.
Wenn Neukunden in ersten Bestell-Monat eine weitere Bestellung aufgeben haben, werden sie in diesem Monat einmal als Neukunde und einmal als Bestandskunde gezählt.

3. Kundenmonitor

Die Lebensdauer einer Kunden berechnet sich aus der Differenz des ersten und letzten Einkaufs.

a) Bestimmen Sie die Lebensdauer der Kunden nach
   - Jahren
   - Monaten
   - Tagen

   Hinweis: Funktion `datediff({day,month,year},<firstdate>,<lastdate>)` verwenden.

```
SELECT ld.Kunde, -- Kundennumer
DATEDIFF (day,ld.first_order,ld.last_order) Tage,
-- Berechnung der Differenz in Tagen
DATEDIFF (month,ld.first_order,ld.last_order) Monate,
-- Berechnung der Differenz in Monaten
DATEDIFF (year,ld.first_order,ld.last_order) Jahre
-- Berechnung der Differenz in Jahren
FROM
(SELECT c.riskID Kunde,
MIN(postingDate) first_order,MAX(postingDate)last_order
FROM [dbo].[iw_sales]s,[dbo].[iw_customer]c
WHERE s.customerNo = c.customerNo
GROUP BY c.riskID)ld -- Unterabfrage in Klammern mit Alias
```

b) Betrachten Sie die Probleme, die durch Rundungsungenauigkeiten dabei entstehen. Suchen Sie nach einer genaueren Möglichkeit für die Bestimmung der Differenz in Monaten und Jahren.

Differenz in Jahren wird ganzzahlig dargestellt, dies ist zu ungenau, sinnvoll 1 oder 2 Nachkommastellen (evtl. auch bei Monaten)

```
cast ((cast(DATEDIFF (day,ld.first_order,ld.last_order)as numeric)
```

```
/ 365)as DECIMAL(10,2)) Jahre
```

c) Ermitteln Sie die Anzahl der Kunden pro Monatsergebnis.

```
SELECT ldm.Monate, COUNT (*) Anzahl FROM
(SELECT ld.Kunde, DATEDIFF (month ,ld.first_order ,ld.last_order ) Monate
FROM
(SELECT c.riskID Kunde,
MIN(postingDate) first_order,MAX(postingDate)last_order
FROM [dbo].[iw_sales]s,[dbo].[iw_customer]c
WHERE s.customerNo = c.customerNo
GROUP BY c.riskID) ld) ldm
GROUP BY ldm.Monate
ORDER BY ldm.Monate
```

d) Bilden Sie Gruppen von Kunden auf der Basis ihrer Lebensdauer in Monaten. Bilden Sie dabei einfache Gruppen:

- 0 Monate
- Bis zu 3 Monate
- Bis zu 6 Monaten
- Bis zu 9 Monaten
- Bis zu 12 Monaten
- Über 12 Monate

Hinweis: Verwenden Sie die `CASE WHEN`-Klausel in Verbindung mit Summenbildung.

```
SELECT
SUM(CASE WHEN (Monate = 0) THEN 1 ELSE 0 END) '0 Monate',
SUM(CASE WHEN (Monate > 0 AND Monate <= 3) THEN 1 ELSE 0 END)
'bis 3 Monate',
...
sum(CASE WHEN (Monate > 12) THEN 1 ELSE 0 END) '> 12 Monate'
FROM
(SELECT ld.Kunde,
-- Division durch 30.42 zur Berechnung Tages- in Monatsdifferenz
cast ((cast(DATEDIFF (day ,ld.first_order ,ld.last_order )as numeric)/
30.42) as decimal(10,2))Monate
FROM
(SELECT c.riskID Kunde,
MIN(postingDate) first_order,MAX(postingDate)last_order
FROM [dbo].[iw_sales]s,[dbo].[iw_customer]c
WHERE s.customerNo = c.customerNo
GROUP BY c.riskID)ld)ldm
```

Übungsblatt 8

1. Analyse des Kundenwertes (Customer Value)

Um den Kundenwert zu berechnen, müssen zunächst die Aktivitätsdaten des Kunden berechnet werden. Dies sind die bereits bekannten Daten aus dem KPI-Bericht und dem Kundenmonitor.

a) Ermitteln Sie die Bestelldaten des Kunden (gruppiert nach der eindeutigen Kunden-ID riskID): Anzahl Konten, Anzahl Bestellungen, Anzahl bestellter Artikel, Nettosumme.

b) Die Aktivitätsbericht des Kunden soll um die Retourendaten erweitert werden: Anzahl Retouren, Anzahl retournierter Artikel, Retouren-Nettowert.
Hinweis: Left Outer Join mit der Retourentabelle (*iw_return_line*) über *die customerNo*.

c) Nehmen Sie einige Korrekturen an dem in b) erzeugten Bericht vor: In der Nettosumme müssen die Dezimalwerte richtig eingestellt werden. Artikel- oder Versandkosten werden anhand der Spalte type in den Tabelle iw_sales und iw_return gekennzeichnet. Im Bericht sind die Bestell- und Retourendaten ohne Versandkosten auszuweisen.

Eine Lösung Bestelldaten `left outer join` Retourendaten mit jeweils der Bedingung type = 2 (zum Ausschluss von Versandkosten) ist fehlerhaft, weil die Bedingung type = 2 den Outer Join aufhebt, so dass Kunden ohne Retouren aus dem Ergebnis verschwinden.
Aufbau des Ergebnisses in mehreren Schritten:
Bestelldaten a, Retourendaten b – jeweils mit type = 2, a `left outer join` b

```
SELECT a.riskID, COUNT (distinct a.customerNo) Konten,
SUM (a.Bestellungen)Bestellungen, SUM (a.Artikel) Artikel,
SUM (a.Nettosumme)Nettosumme, SUM (b.Retouren)Retouren,
SUM (b.Ret_Artikel)Ret_Artikel, SUM (b.Ret_Nettowert)Ret_Nettowert
FROM
(SELECT c.riskID, s.customerNo,
COUNT (distinct s.orderNo)Bestellungen, sum (s.quantity)Artikel,
cast (SUM (s.amount *s.quantity)as decimal(10,2))Nettosumme
FROM [dbo].[iw_customer]c,[dbo].[iw_sales]s
WHERE c.customerNo = s.customerNo
and s.type = 2
GROUP BY c.riskID, s.customerNo)a
left outer join
(SELECT rl.customerNo,
COUNT (distinct rl.returnNo)Retouren, SUM (rl.quantity)Ret_Artikel,
SUM (rl.line_amount)Ret_Nettowert
FROM
[dbo].[iw_return_line]rl
WHERE rl.type = 2
GROUP BY rl.customerNo)b
on a.customerNo = b.customerNo
GROUP BY a.riskID
```

d) Auf dem Weg zur Berechnung des Kundenwertes müssen wir für jeden Kunden errechnen, welche Kosten er durch Bestellungen und Retouren verursacht. In unserem Beispiel-Shop berechnen wir pro Bestellung 9,50 € und pro Retoure 5,80 €.

Lösung ist analog zu Aufgabe 2 c), zusätzlich in SELECT-Liste:

```
...
(SUM (a.Bestellungen)*9.5)Bestellkosten,
(SUM (b.Retouren)*5.8)Retourkosten
```

e) Somit lässt sich jetzt der individuelle Wert eines Kunden berechnen. Man rechnet für jeden Kunden den Nettowerte aller Bestellungen aus und subtrahiert davon:
   - den Nettowert all seiner Retouren
   - die Handling-Kosten all seiner Bestellungen (9,50 € pro Bestellung)
   - die Handling-Kosten all seiner Retouren (5,80 € pro Retoure)

   Hinweis:
   Zu Behandlung von NULL-Werten in den Retourenspalten verwenden Sie die Funktion `ISNULL (<Ergebnis>, 0)`, so dass sich immer ein Nettoertrag errechnen lässt.

Mit Behandlung der NULL-Werte

```
SELECT ...
ISNULL((SUM (b.Retouren)*5.8),0)Retourkosten,
SUM(a.Nettosumme)- ISNULL(SUM (b.Ret_Nettowert),0)-
(SUM(a.Bestellungen)*9.5)- ISNULL((SUM(b.Retouren)*5.8),0)Nettoertrag
...
```

2. Bestellhistorie von Kunden

Für jeden Kunden sollen die Bestellungen nach Bestelldatum geordnet untereinander geschrieben werden, so dass die Bestellungen zueinander in Beziehung gesetzt werden können. Die einzelnen Bestellungen erhalten eine laufende Nummer zur Kennzeichnung der ersten, zweiten, …, n-ten Bestellung eines Kunden.
Die Ergebnistabelle soll folgende Informationen beinhalten: lfdNr pro Kunde, riskID, Datum, Bestell-Nr., Artikelmenge, Gesamtsumme, Anzahl_R_Artikel, R_Nettowert
Hinweis:
Die Syntax für die Ausgabe der Werte in Partitionen pro Kunde sieht wie folgt aus:
`ROW_NUMBER() OVER (PARTITION BY <Spaltenname> ORDER BY <Spaltenname>)`

```
SELECT ROW_NUMBER() OVER (PARTITION BY bd.riskID ORDER BY bd.Datum) as
lfdNr,
-- Laufende Nummer
bd.riskID, bd.Datum, bd.Bestellung, bd.Artikelmenge,
bd.Gesamtsumme,rd.Anzahl_R_Artikel, rd.R_Nettowert
FROM
(SELECT a.riskID, a.Datum, a.Bestellung, a.Artikelmenge, a.Gesamtsumme
FROM
(SELECT c.riskID,s.postingDate Datum,s.orderNo Bestellung,
cast (SUM (s.quantity)as decimal(10,0))Artikelmenge,
cast (SUM (s.amount)as decimal(10,2))Gesamtsumme
FROM [dbo].[iw_sales]s,[dbo].[iw_customer]c
WHERE s.customerNo = c.customerNo
and s.type = 2 -- ohne Frachtkosten
and s.quantity > 0
GROUP BY c.riskID,s.postingDate,s.orderNo)a) bd
-- Das sind die Bestelldaten
```

```
LEFT OUTER JOIN
(SELECT c.riskID IRID,s.postingDate Datum,
rh.orderNo Bestellung,
cast(SUM( rl.quantity)as decimal(10,0)) Anzahl_R_Artikel,
cast(SUM (rl.line_amount)as decimal(10,2))R_Nettowert
FROM [dbo].[iw_sales]s,[dbo].[iw_customer]c,[dbo].[iw_return_line]rl,[dbo].
[iw_return_header]rh
WHERE s.type = 2
and s.orderNo = rh.orderNo
and rh.returnNo = rl.returnNo
and s.IWAN = rl.IWAN -- Artikelnummer hinzu!
and rl.type = 2
and s.customerNo = c.customerNo
GROUP BY c.riskID, s.postingDate, rh.orderNo)as rd
-- Das sind die Retourendaten
on bd.Bestellung = rd.Bestellung
```

Übungsblatt 9

1. Machen Sie sich mit der Bezeichnung von DB-Objekten aus anderen Datenbanken oder Schemas vertraut. Setzen Sie sich dabei auch mit dem Begriff des Schemas in einer Datenbank auseinander.

Es gibt eine Namenshierarchie: Servername.DBName.SchemaName.Objektname
Innerhalb jeder DB gibt es ein Standardschema dbo (DB Owner), dieses muss nicht explizit angegeben werden.

2. Definition von Auswertungstabellen

a) Definieren Sie in Ihrem Schema dwh17<xx> (mit <xx> = 01..14) eine neue Tabelle *KPI_Kunden_Report* auf Basis der Anfrage zur Ermittlung der Kennzahlen zur Auswertung des Kundenverhaltens auf Basis der eindeutigen riskID (vgl. Aufgabe 1.c, Übungsblatt 7).
Hinweis: CREATE TABLE AS .. (wie in Oracle) funktioniert nicht.
Alternative zum Kopieren von Tabellen: SELECT … INTO …
Testen Sie die Funktionsweise der neuen Tabelle.

```
Select a.Jahr, a.Monat, a.Kunden, a.Kundenkonten, a.Bestellungen,
a.Artikelmenge, a.Nettosumme,
cast(a.Nettosumme/a.Bestellungen as DECIMAL(10,2))Warenkorb,
cast(a.Artikelmenge/a.Bestellungen as DECIMAL(10,2))Artikel_WK
into KPI_Kunden_Report
from
-- Inhalte/Berechnungen aus der Unterabfrage
…
```

b) Testen Sie die Funktionsweise der neuen Tabelle.

3. Definition von Sichten

a) Definieren Sie eine Sicht in Ihrem Schema dwh17[xx] (mit xx = 01..14), die eine monatliche Auswertung für jede Produktgruppe über den erzielten Nettoumsatz enthält.

```
create view dwh17xx.testview with schemabinding as
select p.productgroup, DATEPART (yyyy,s.postingdate) Jahr,
       DATEPART (mm,s.postingdate) Monat,
       cast (SUM (s.quantity * s.amount) as decimal(10,2)) umsatz,
       count_big(*) tmp
from [dbo].iw_sales s, [dbo].iw_article p
where s.IWAN = p.IWAN
group by p.productgroup, DATEPART (yyyy,s.postingdate), DATEPART
    (mm,s.postingdate)
```

b) Testen Sie die Funktionsweise der Sicht.

4. Materialisierte Sicht

Erzeugen Sie aus der in Aufgabe 3 angelegten Sicht eine materialisierte Sicht.

```
create unique clustered index testindex on
dwh17xx.testview(productgroup, jahr, monat)
```

Fehlermeldung:

```
Der gruppierte Index 'testindex' kann nicht für die iw_shop.dwh17xx.testview-Sicht
erstellt werden, da die Auswahlliste der Sicht einen Ausdruck für das Ergebnis
einer Aggregatfunktion oder Gruppierungsspalte enthält. Entfernen Sie den
betreffenden Ausdruck aus der Auswahlliste.
```

Alternative: Erzeugen einer neuen Tabelle `'test_table'` mittels SELECT .. INTO

```
create unique clustered index testindex on
dwh17xx.test_table(productgroup, jahr, monat)
```

oder unter Verwendung einer Sicht:

```
create view dwh17xx.testview with schemabinding as
select *
from dwh17xx.test_table

create unique clustered index testindex on
dwh17xx.testview(productgroup, jahr, monat)
```

5. Verwendung von Sichten

a) Formulieren Sie die folgende Anfrage:
Ermitteln Sie den jährlichen Umsatz für Artikel pro Produktgruppe im Bereich der Produktgruppen-Codes 150 .. 225.

b) Formulieren Sie diese Anfrage unter Verwendung der bereits in Aufgabe 3 angelegten materialisierten Sicht.

```
select tv.productgroup, tv.Jahr, SUM(tv.Umsatz) Jahresumsatz
from dwh17xx.testview tv, [dbo].iw_article p
where tv.productgroup = p.productgroup and
      p.productgroup between 150 and 225
group by tv.productgroup, tv.Jahr
order by productgroup
```

c) Untersuchen Sie die Möglichkeiten einer automatischen Umschreibung der Anfrage durch das Datenbanksystem (Query Rewrite) bei Vorhandensein materialisierter Sichten, wobei in der Sicht vorberechnete (Teil-)Ergebnisse genutzt werden. Nutzen Sie hierzu ein Tool zur Anzeige von Anfrageplänen zur Analyse der durchgeführten Query (z.B. in Visual Studio).

Im Transact-SQL-Fenster Anfrage eingeben.
Rechte Maustaste: Anzeige der Anfrage-Operation sowie weiterer statistischer Größen.

Ein mögliches Query Rewrite unter Nutzung einer vorhandenen materialisierten Sicht setzt voraus, dass sich diese im gleichen Schema befindet.

6. PowerPivot

a) Stellen Sie aus PowerPivot eine Verbindung zur MS SQL Server Datenbank *iw_shop* her.

Menü Daten -> Externe Daten …
Analog zu Visual Studio (Server, Datenbank, User/Password)
Auswahl von Tabellen oder Sichten, die importiert werden

b) Testen Sie die Funktionsweise von PowerPivot, indem Sie die Ergebnisse der Anfrage, die der in Aufgabe 2.a) definierten Sicht zugrundeliegt, visualisieren.

Einfügen -> Empfohlene Pivot Tables
Auswahl der Sicht
Definition der Achsen (z.B. Produktgruppe, Umsatz)

c) Setzen Sie sich mit den Möglichkeiten von Data Analysis Expressions (DAX) in Power-Pivot auseinander. Finden Sie sinnvolle Anwendungsmöglichkeiten für Ihre Beispiel-anfrage.

Zusätzliche Filter.
Berechnung von Zwischen und Gesamtsummen (Alternative zu SQL-OLAP)