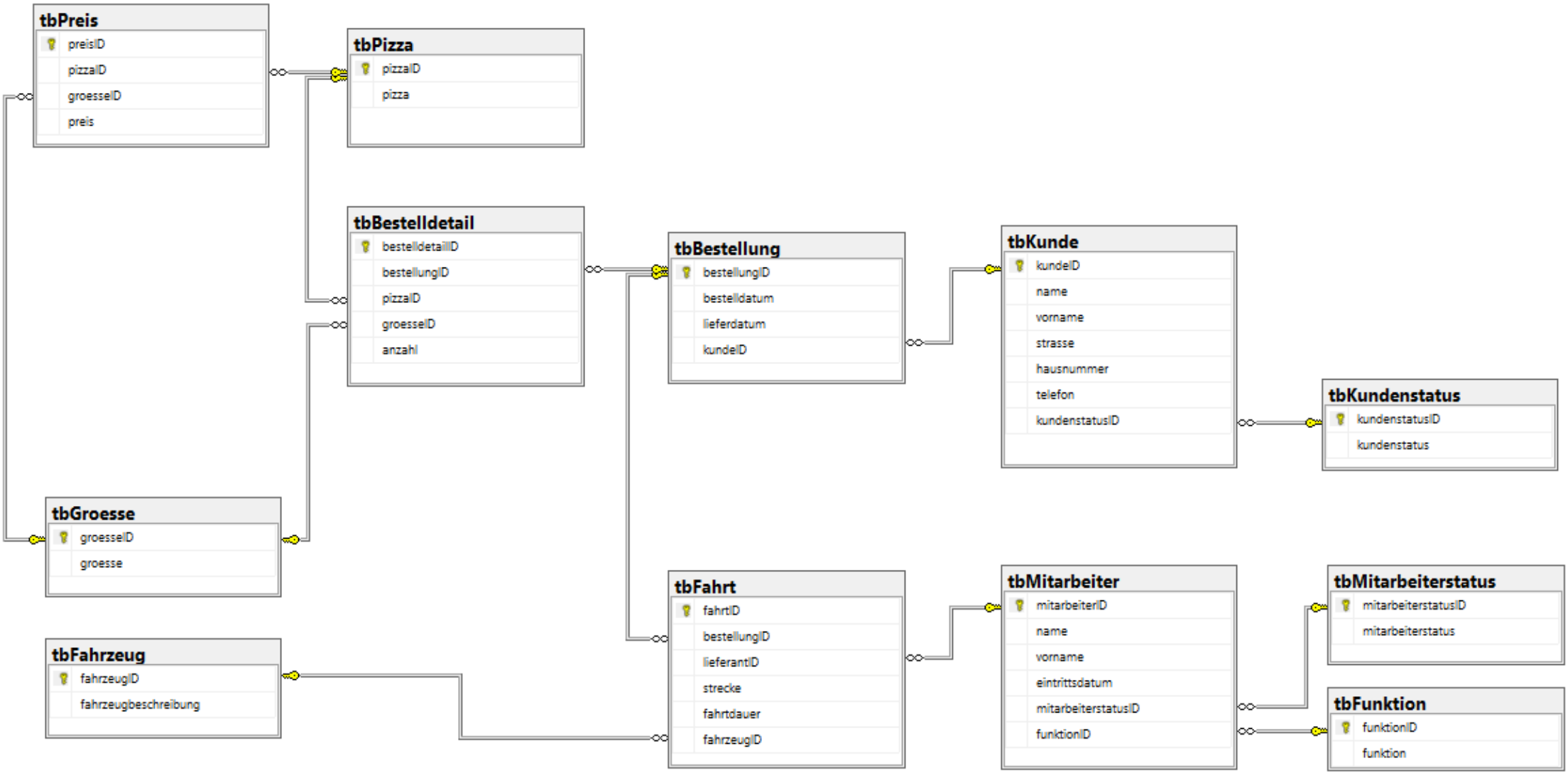
Projektbeschreibung Gruppe B: „Performance-Pizzeria“ April 2020

- Andrea ***
- Annina ***
- Benjamin ***
- Claudia ***

Businesslogik: (blau = eigene Tabelle)

- Eine Münchener Pizzeria bietet ihren Kunden einen Lieferservice an.
- Folgende **Kunde**ndaten werden erfasst: **Name, Vorname, Straße, Hausnummer, Telefon, Kundenstatus** (Normal, Premium, Gold, Platin).
  Dieser **Kundenstatus** ist abhängig von der Anzahl der Pizzen, die ein Kunde bisher insgesamt bestellt hat – und wird bei neuen Bestellungen ggf. angepasst (*Trigger + Prozedur mit Cursor*).
- Die Pizzeria bietet unterschiedliche **Pizzen** in verschiedenen **Größen** an – mit dem jeweiligen **Preis**.
- Für jede **Bestellung**, die von einem Kunden getätigt wird, werden **Bestelldatum** und **Lieferdatum** erfasst (inkl. Uhrzeit). Außerdem wird die **Anzahl** der bestellten Pizzen (+ Größe + Sorte) vermerkt (**Bestelldetails**).
- In der Pizzeria sind **Mitarbeiter** in verschiedenen **Funktionen** beschäftigt (Küche, Reinigung, Lieferant…). Neben ihrer **Funktion** werden auch **Name, Vorname** und **Eintrittsdatum** erfasst. Außerdem hat jeder Mitarbeiter einen **Status** (underperformer, performer, overperformer)
- Besonderes Interesse besteht an der Leistung der **Lieferanten**. Daher werden alle **Fahrten**, die sie machen, genau dokumentiert (**Bestellung, gefahrene Strecke**, **benötigte Zeit** und verwendetes **Fahrzeug**). Daraus wird die Geschwindigkeit errechnet, mit der die Fahrer durchschnittlich unterwegs waren (*Funktion*).
- Außerdem wird der Umsatz berechnet, den jeder Lieferant erzielt (monatlich, jährlich, vorjährlich → *table function multi statement*)


tbPreis
preisID
pizzaID
groesseID
preis
tbPizza
pizzaID
pizza
tbBestelldetail
bestelldetailID
bestellungID
pizzaID
groesseID
anzahl
tbBestellung
bestellungID
bestelldatum
lieferdatum
kundeID
tbKunde
kundeID
name
vorname
strasse
hausnummer
telefon
kundenstatusID
tbKundenstatus
kundenstatusID
kundenstatus
tbGroesse
groesseID
groesse
tbFahrzeug
fahrzeugID
fahrzeugbeschreibung
tbFahrt
fahrtID
bestellungID
lieferantID
strecke
fahrtdauer
fahrzeugID
tbMitarbeiter
mitarbeiterID
name
vorname
eintrittsdatum
mitarbeiterstatusID
funktionID
tbMitarbeiterstatus
mitarbeiterstatusID
mitarbeiterstatus
tbFunktion
funktionID
funktion

Verzeichnis	Dateinamen des Skripts	Beschreibung	verwendete Technik
001-DB-Create-Scripts	001-001-createDatabase-dbPizzeria.sql	Erstellung der Datenbank dbPizzeria	
	001-002-createTable-tbFahrt.sql	Erstellung der Tabelle tbFahrt, welche die Fahrten der Pizzalieferanten enthält	
	001-003-pk-tbBestelldetail.sql	Erstellung des Primärschlüssels für die Tabelle tbBestelldetail	
	001-004-idxPizzaIDgroesseID.sql	Erstellung eines eindeutigen Index in der Preistabelle, so dass sichergestellt ist, dass zu einer Pizzaart und einer Größe nur höchstens ein Preis definiert wird	
	001-005-foreignKeyBestellungKunde.sql	Erstellung eines Fremdschlüssels für die Tabelle tbBestellung, der auf den Primärschlüssel 'KundeID' der Tabelle tbKunde referenziert	
	001-006-defaultAnzahlPizzen.sql	Default festgelegt für Anzahl Pizzen pro Bestellung pro Pizza, dieser ist 1	
	001-007-checkDatumsabgleich.sql	Check hinzugefügt, so dass sichergestellt ist, dass das Lieferdatum immer größer ist als das Bestelldatum. Die Datumsattribute beinhalten auch die Uhrzeit.	
002-Abfragen und Sichten	002-001-queryUebersichtFahrleistung.sql 002-001-viewUebersichtFahrleistung.sql	Abfrage der Lieferanten und ihrer Fahrten als Übersicht	Mit Abfrage-Elementen von INNER JOIN, LEFT OUTER JOIN, WHERE
	002-002-queryDurchschnittsgeschwindigkeit.sql 002-002-viewDurchschnittsgeschwindigkeit.sql	Abfrage der Lieferanten mit mehr als einer Fahrt und deren Gesamtfahrtdauer, Gesamtstrecke und Durchschnittsgeschwindigkeit	Mit Abfrage-Elementen von GROUP BY, HAVING, SUM, COUNT Und Verwendung einer selbst definierten Skalarwertfunktion (sfFahrtdauerStunden())
003-StoredFunctions	003-001-sfFahrtdauerStunden.sql 003-001-sfFahrtdauerStunden_Testskript.sql	Skalarfunktion, welche die Fahrtdauer (Datentyp Time) in Stunden (Datentyp Decimal) umrechnet	Systemfunktion DATEPART und CONVERT verwendet

	003-002-sfAnzahlPizzenProKunde.sql 003-002- sfAnzahlPizzenProKunde_Testskript.sql	Skalarfunktion zur Feststellung, wieviele Pizzen ein gegebener Kunde bis zu einem gegebenem Datum bestellt hat	Mit Abfrage-Elementen von GROUP BY und SUM
	003-003-tfDurchschnittLieferantImMonat.sql 003-003- tfDurchschnittLieferantImMonat_Testskript.sql	Inline-Tabellenfunktion, welche zu einem eingegebenen Monat/ Jahr eine Tabelle ausgibt mit alle Lieferanten, die in dem Monat mind. eine Fahrt gemacht haben und deren Gesamtfahrtdauer, Gesamtstrecke und Durchschnittsgeschwindigkeit in diesem Monat	Mit Abfrage-Elementen von GROUP BY, SUM Und Verwendung einer selbst definierten Skalarwertfunktion (sfFahrtdauerStunden() siehe Dateinamenpräfix 003-001)
	003-004-sfMonatsUmsatzLieferant.sql 003-004- sfMonatsUmsatzLieferant_Testskript.sql	Skalarfunktion, welche den Monatsumsatz für einen Lieferanten berechnet	Mit Abfrage-Elementen von GROUP BY und SUM
	003-005-tfmsReportUmsatz.sql 003-005-tfmsReportUmsatz_Testskript.sql	Tabellenwertfunktion Multi-Statement , die für einen vorgegebenen Monat/ Jahr einen Report ausgibt, in dem für den eingegebenen Monat/ Jahr alle Lieferanten gelistet sind mit ihrem Vorjahresumsatz, Jahresumsatz, und Monatsumsatz	Unter Verwendung eines CURSORS, dreier WHILE-Schleifen und einer selbst definierten Funktion (sfMonatsUmsatzLieferant() (siehe Dateinamenpräfix 003-004)
004-StoredProcedures	004-001-spKundenstatusUpdate.sql 004-001-spKundenstatusUpdate_Testskript.sql	Prozedur mit OUTPUT-Parameter, welche den Kundenstatus aktualisiert aufgrund der Anzahl bestellter Pizzen bis zum gegebenen Datum und ausgibt, bei wievielen Kunden der Status verändert wurde	Unter Verwendung eines CURSORS, einer WHILE-Schleife, IF-Anweisung und einer selbst definierten Funktion (sfAnzahlPizzenProKunde siehe Dateinamenpräfix 003-002)
005-DML-Trigger	005-001-trKundenstatusUpdate.sql 005-001-trKundenstatusUpdate_Testskript.sql	Trigger für das Update des Kundenstatus (Aufruf Prozedur spKundenstatusUpdate) im Falle eines INSERT in die Tabelle tbBestelldetail	Unter Verwendung einer selbst definierten Prozedur spKundenstatusUpdate() (siehe Dateinamenpräfix 004-001)

006-LOGIN-USER-ROLE	006-001-Create-Login-Buchhaltung.sql 006-001-Create-databaseUser-BuchhaltungLesen.sql	DB-Benutzer mit nur Leserechten	
	006-002-Create-databaseUser-BuchhaltungSchreiben.sql 006-002-Create-Login-BuchhaltungSchreiben.sql	DB-Benutzer mit Schreibrechten	
007-BackUp	007-001-spBackupMitFehlermeldung.sql 007-001-spBackupMitFehlermeldung_Testskript.sql	Prozedur, um ein Backup für eine vorgegebene Datenbank in einem vorgegebenen Pfad mit Zeitstempel zu speichern mit Erfolgsmeldung 'Backup erfolgreich erstellt. Molto bene!' oder Meldung bei Fehler	Abwandlung und Anpassung der Kursinhalte aus '002-07-BACKUP-alle-DB-CURSOR-ENDVERSION.sql' vom 22.04.2020
	007-002-dbPizzeria-20200429-163202423.bak	Mit der Prozedur erstelltes Backup.	Ausführung der Prozedur spBackupMitFehlermeldung() (siehe Dateinamenpräfix 007-001)
	007-003-superscript-dbPizzeria.sql	Gesamtskript für Datenbank dbPizzeria incl aller DB-Objekte alle Tabellen, Sichten, Funktionen, Prozeduren, Trigger alle Logins, User, Rollen alle Daten	