Aufgabe Nr. 1, BETWEEN-Prädikat
Welche Fahrten finden von Heiligabend bis Neujahr 2001/2002 einschließlich statt? Rückgabe-Typ sei (fahrtNr).

```
SELECT fahrtNr FROM Fahrt
WHERE tag BETWEEN '2001-12-24' AND '2002-1-1' ;
```

Aufgabe Nr. 2, Simulation des BETWEEN-Prädikates
Welche Fahrten finden von Heiligabend bis Neujahr 2001/2002 einschließlich statt? Rückgabe-Typ sei (fahrtNr).

```
SELECT fahrtNr FROM Fahrt
WHERE '2001-12-24' <= tag AND '2002-1-1' >= tag ;
```

Aufgabe Nr. 3, IN-Prädikat
Welche Linien führen an den Haltestellen 18 oder 20 vorbei? Rückgabe-Typ sei (linieNr).
Stelle die Anfrage unter Verwendung des IN-Prädikates.

```
SELECT linieNr FROM Erreicht
WHERE haltestelleNr IN ( 18 , 20 ) ;
```

Aufgabe Nr. 4, Simulation des IN-Prädikates
Welche Linien führen an den Haltestellen 18 oder 20 vorbei? Rückgabe-Typ sei (linieNr).
Stelle die Anfrage ohne Verwendung des IN-Prädikates (ohne NOT und ohne Klammern).

```
SELECT linieNr FROM Erreicht
WHERE 18 = haltestelleNr OR 20 = haltestelleNr ;
```

Aufgabe Nr. 5, LIKE-Prädikat
Gib alle Fahrer aus, deren letzter Buchstabe des Nachnamens ein 's' ist. Ergebnistyp sei das volle Tupel.

```
SELECT * FROM Fahrer
WHERE nachname LIKE '%s' ;
```

Aufgabe Nr. 6, Wertebereichdefinition
Man benötigt einen Typen GESCHLECHT_TYP, der zur Aufnahme des Geschlechtes der Fahrer, 'Frau' oder 'Mann', geeignet ist. Dabei ist die am genauesten passende Datentypdefinition zu verwenden.
Definiere den Wertebereich.

```
CREATE DOMAIN geschlecht_typ CHAR ( 4 ) ;
```

Aufgabe Nr. 7, Arbeiten mit nutzerdefinierten Wertebereichen
Erstelle eine Tabelle Fahrer, die (vorerst nur) die Attribute nachname, vorname und geschlecht enthält. (Datentypen links unten nachsehen.)

```
CREATE TABLE Fahrer (
 nachname VARCHAR ( 31 ) ,
 vorname VARCHAR ( 31 ) ,
 geschlecht GESCHLECHT_TYP
) ;
```

Aufgabe Nr. 8, Wertebereichdeklaration mit Default
Man benötigt einen Typen GESCHLECHT_TYP, der zur Aufnahme des Geschlechtes der Fahrer, 'Frau' oder 'Mann', geeignet ist. Dabei ist die am genauesten passende Datentypdefinition zu verwenden.
Definiere den Wertebereich mit unter Verwendung des Default-Wertes 'Mann' ohne Integritätsbedingung.

```
CREATE DOMAIN GESCHLECHT_TYP CHAR(4) DEFAULT 'Mann';
```

Aufgabe Nr. 9, Wertebereichdeklaration mit Integritätsbedingung
Man benötigt einen Typen GESCHLECHT_TYP, der zur Aufnahme des Geschlechtes der Fahrer, 'Frau' oder 'Mann', geeignet ist. Dabei ist die am genauesten passende Datentypdefinition zu verwenden.
Definiere den Wertebereich ohne Defaultwert unter Verwendung der Integritätsbedingung, dass es sich nur um eines der beiden Geschlechter handeln kann. (Zwitter wurden hier nicht vorgesehen).

```
CREATE DOMAIN GESCHLECHT_TYP CHAR ( 4 )
 CHECK ( 'Frau' = VALUE OR 'Mann' = VALUE ) ;
```

Aufgabe Nr. 10, Dynamische Wertebereichfestlegung
Der Wertebereich des Attributs busNr der Tabelle Fahrt, BUS_NR_TYP, soll dynamisch so festgelegt werden, dass nur Nummern vorhandener Busse verwendet werden koennen. Die Pruefung soll bei Verlangen aufschiebbar sein, aber per Voreinstellung sofort vollstreckt werden. Die Einschraenkung (Constraint) soll explizit als busVorhanden benannt werden.

```
CREATE DOMAIN BUS_NR_TYP AS SMALLINT
 CONSTRAINT busVorhanden CHECK ( VALUE IN ( SELECT busNr FROM Bus ) )
  DEFERRABLE INITIALLY IMMEDIATE ;
```

Aufgabe Nr. 11, SET CONSTRAINTS-Klausel

Der Wertebereich des Attributs busNr der Tabelle Fahrt, BUS_NR_TYP, soll dynamisch so festgelegt werden, dass nur Nummern vorhandener Busse verwendet werden koennen. Die Pruefung soll bei Verlangen aufschiebbar sein, aber per Voreinstellung sofort vollstreckt werden. Die Einschraenkung (Constraint) soll explizit als busVorhanden benannt werden.
Durch die dynamische Wertebereichfestlegung droht ein Fehler, wenn ein neuer Bus für eine zusätzliche Fahrtroute eingekauft wurde und innerhalb einer Transaktion zunächst die neue Fahrtroute (fahrtNr = 34511, busNr = 232, linieNr = 112, uhrzeitBeginn = '23:45', wochentag = 'So', tag = '2001-11-11') und danach der neue Bus für diese Fahrt (busNr = 232, anschaffungstag = '1996-11-9') eingegeben wird. Vermeide den Fehler durch Angabe der entsprechenden SET CONSTRAINTS-Klauseln. Die ausdrückliche Zuweisung der

```
SET CONSTRAINTS busVorhanden DEFERRED ;
INSERT INTO Fahrt VALUES ( 34511 , 232 , 112 , '23:45' , 'So' , '2001-11-11' ) ;
INSERT INTO Bus VALUES ( 232 , '1996-11-09' ) ;
SET CONSTRAINTS busVorhanden IMMEDIATE ;
```

Aufgabe Nr. 12, Anlegen einer Tabelle ohne Integritätsbedingungen

Es wird eine Tabelle Erreicht(linieNr, haltestelleNr, fahrzeit) benötigt. (Siehe Hilfs-Window 'Datentypen'.) Lege die Tabelle ohne Integritätsbedingungen an.

```
CREATE TABLE Erreicht(
 linieNr SMALLINT,
 haltestelleNr SMALLINT,
 fahrzeit INTERVAL MINUTE
);
```

Aufgabe Nr. 13, NOT NULL-Spezifikation

Es wird eine Tabelle Erreicht(linieNr, haltestelleNr, fahrzeit) benötigt. (Siehe Hilfs-Window 'Datentypen'.) Lege die Tabelle an, wobei weder die fahrtNr, die haltestelleNr noch die fahrzeit eine Nullmarke aufweisen darf. Es sollen keine benannten Einschränkungen verwendet werden.

```
CREATE TABLE Erreicht(linieNr SMALLINT NOT NULL, haltestelleNr SMALLINT NOT NULL, fahrzeit
INTERVAL MINUTE NOT NULL);
```

Aufgabe Nr. 14, Wertebereicheinschränkungen und Spaltenvorbelegungen

Die Tabelle Erreicht(linieNr, haltestelleNr, fahrzeit) mit einer genau dreistelligen Liniennummer und Haltestellennummer wird benötigt. Die vorgesehene Fahrzeit seit dem Aufbruchspunkt der Linie soll öchstens 120 Minuten betragen. (Siehe Hilfs-Window 'Datentypen'.) Verwende unbenannte spaltenbezogene Constraints und BETWEEN, wo es geht.

```
CREATE TABLE Erreicht(
 linieNr SMALLINT CHECK(VALUE BETWEEN 100 AND 999),
 haltestelleNr SMALLINT CHECK(VALUE BETWEEN 100 AND 999),
 fahrzeit INTERVAL MINUTE CHECK(VALUE <= '120')
);
```

Aufgabe Nr. 15, Anfrage als Wertebereicheinschränkung

Die Tabelle Erreicht(linieNr, haltestelleNr, fahrzeit) mit einer genau dreistelligen Liniennummer und Haltestellennummer wird benötigt. Die vorgesehene Fahrzeit seit dem Aufbruchspunkt der Linie soll öchstens 120 Minuten betragen. (Siehe Hilfs-Window 'Datentypen'.) Verwende unbenannte spaltenbezogene Constraints und BETWEEN, wo es geht.
Wandle die CHECK-Klausel für linieNr so ab, dass durch eine Anfrage der Wertebereich für linieNr geeignet

```
CREATE TABLE Erreicht(
 linieNr SMALLINT CHECK(
  VALUE BETWEEN 100 AND 999 AND
  VALUE IN(SELECT linieNr FROM Linie)),
 haltestelleNr SMALLINT CHECK(
  VALUE BETWEEN 100 AND 999),
 fahrzeit INTERVAL MINUTE CHECK(
  VALUE <= '120'));
```

Aufgabe Nr. 16, CHECK-Klausel als Hilfmittel zur Formatierung
Die Tabelle Erreicht(linieNr, haltestelleNr, fahrzeit) mit einer höchstens dreistelligen Liniennummer und Haltestellennummer wird benötigt. Die vorgesehene Fahrzeit seit dem Aufbruchspunkt der Linie soll höchstens 120 Minuten betragen. Die Einschränkungen sollen 'linieNr3stellig', 'haltestelleNr3stellig' und 'hoechstens120Minuten' heissen. (Siehe Hilfs-Window 'Datentypen'.) Verwende BETWEEN, wo es geht. Lege die Tabelle mit Hilfe von spaltenbezogenen Einraenkungen an.

```
CREATE TABLE Erreicht(
 linieNr SMALLINT CONSTRAINT linieNr3stellig
  CHECK(VALUE BETWEEN 0 AND 999),
 haltestelleNr SMALLINT CONSTRAINT haltestelleNr3stellig
  CHECK(VALUE BETWEEN 0 AND 999),
 fahrzeit INTERVAL MINUTE CONSTRAINT hoechstens120Minuten
  CHECK(VALUE <= '120')
);
```

Aufgabe Nr. 17, CHECK-Klausel im Anschluss an die Deklarationen der Spalten
Die Tabelle Erreicht(linieNr, haltestelleNr, fahrzeit) mit einer höchstens dreistelligen Liniennummer und Haltestellennummer wird benötigt. Die vorgesehene Fahrzeit seit dem Aufbruchspunkt der Linie soll höchstens 120 Minuten betragen. Die Einschränkungen sollen 'linieNr3stellig', 'haltestelleNr3stellig' und 'hoechstens120Minuten' heissen. (Siehe Hilfs-Window 'Datentypen'.) Verwende BETWEEN, wo es geht. Lege die Tabelle an, wobei drei tabellenbezogene Einschränkungen verwendet werden sollen.

```
CREATE TABLE Erreicht (
 linieNr SMALLINT ,
 haltestelleNr SMALLINT ,
 fahrzeit INTERVAL MINUTE ,
 CONSTRAINT haltestelleNr3stellig CHECK ( haltestelleNr BETWEEN 0 AND 999 ) ,
 CONSTRAINT hoechstens120Minuten CHECK ( '120' >= fahrzeit ) ,
 CONSTRAINT linieNr3stellig CHECK ( linieNr BETWEEN 0 AND 999 )
) ;
```

Aufgabe Nr. 18, Tabellenbezogene CHECK-Klausel
Die Fahrer bekommen zusaetzlich zum Grundgehalt noch Kinderzulage von monatlich 500 Euro je Kind. Formuliere die zugehoerige unbenannte tabellenbezogene CHECK-Klausel. Gib den Betrag dabei in Cent

```
CHECK ( 50000 * kinderzahl = kinderzulage )
```

Aufgabe Nr. 19, Primärschlüssel und Schlüsselkandidaten
Lege die Tabelle Fahrt unter Angabe des Primaerschluessels an. Es sollen keine benannten Einschraenkungen verwendet werden. Fuer das Verstaendnis dieser Aufgabe ist die Bearbeitung von

```
CREATE TABLE Fahrt(
 fahrtNr INTEGER PRIMARY KEY,
 linieNr SMALLINT,
 busNr BUS_NR_TYP,
 uhrzeitBeginn TIME,
 wochentag WOCHENTAG_TYP,
 tag DATE
);
```

Aufgabe Nr. 20, Fremdschlüssel, ohne Behandlung von Integritätsverletzungen
Die Tabelle Fahrt unter Angabe des Primärschlüssels und des Fremdschlüssels mit Verweis auf die Tabelle Bus wird benötigt. Es sind spaltenbezogene Einschränkungen zu verwenden. (Siehe Hilfs-Windows 'Schema' und 'Datentypen'.)

```
CREATE TABLE fahrt (
 fahrtNr INTEGER PRIMARY KEY ,
 linieNr SMALLINT,
 busNr BUS_NR_TYP FOREIGN KEY REFERENCES Bus ,
 uhrzeitBeginn TIME ,
 wochentag WOCHENTAG_TYP ,
 tag DATE
) ;
```

Aufgabe Nr. 21, Fremdschlüssel Weitergabe von Änderungen
Die Tabelle Fahrt unter Angabe des Primärschlüssels und des Fremdschlüssels mit Verweis auf die Tabelle Bus wird benötigt. Es sind spaltenbezogene Einschränkungen zu verwenden. (Siehe Hilfs-Windows 'Schema' und 'Datentypen'.)
Bilde die Tabelle so, dass den Fremdschlüssel verletzende Änderungen der Werte von busNr automatisch

```
CREATE TABLE fahrt (
 fahrtNr INTEGER PRIMARY KEY ,
 linieNr SMALLINT,
 busNr BUS_NR_TYP FOREIGN KEY REFERENCES Bus ON UPDATE CASCADE ,
 uhrzeitBeginn TIME ,
 wochentag WOCHENTAG_TYP ,
 tag DATE
) ;
```

Aufgabe Nr. 22, Fremdschlüssel, Löschung aufgrund von Änderungen
Die Tabelle Fahrt unter Angabe des Primärschlüssels und des Fremdschlüssels mit Verweis auf die Tabelle Bus wird benötigt. Es sind spaltenbezogene Einschränkungen zu verwenden. (Siehe Hilfs-Windows 'Schema' und 'Datentypen'.)
Bilde die Tabelle so, dass den Fremdschlüssel verletzende Änderungen der Werte von busNr den

```
CREATE TABLE fahrt (
 fahrtNr INTEGER PRIMARY KEY ,
 linieNr SMALLINT,
 busNr BUS_NR_TYP FOREIGN KEY REFERENCES Bus ON UPDATE SET NULL ,
 uhrzeitBeginn TIME ,
 wochentag WOCHENTAG_TYP ,
 tag DATE
) ;
```

Aufgabe Nr. 23, Fremdschlüssel, Abweisung von Änderungen
Die Tabelle Fahrt unter Angabe des Primärschlüssels und des Fremdschlüssels mit Verweis auf die Tabelle Bus wird benötigt. Es sind spaltenbezogene Einschränkungen zu verwenden. (Siehe Hilfs-Windows 'Schema' und 'Datentypen'.)
Bilde die Tabelle so, dass den Fremdschlüssel verletzende Änderungen der Werte von busNr von

```
CREATE TABLE fahrt (
 fahrtNr INTEGER PRIMARY KEY ,
 linieNr SMALLINT,
 busNr BUS_NR_TYP FOREIGN KEY REFERENCES Bus ON UPDATE RESTRICT ,
 uhrzeitBeginn TIME ,
 wochentag WOCHENTAG_TYP ,
 tag DATE
) ;
```

Aufgabe Nr. 24, Fremdschlüssel,Löschung aufgrund von Löschungen
Die Tabelle Fahrt unter Angabe des Primärschlüssels und des Fremdschlüssels mit Verweis auf die Tabelle Bus wird benötigt. Es sind spaltenbezogene Einschränkungen zu verwenden. (Siehe Hilfs-Windows 'Schema' und 'Datentypen'.)
Bilde die Tabelle so, dass den Fremdschlüssel verletzende Löschungen von Zeilen in Bus automatisch die

```
CREATE TABLE fahrt (
 fahrtNr INTEGER PRIMARY KEY ,
 linieNr SMALLINT,
 busNr BUS_NR_TYP FOREIGN KEY REFERENCES Bus ON DELETE CASCADE,
 uhrzeitBeginn TIME ,
 wochentag WOCHENTAG_TYP ,
 tag DATE
) ;
```

Aufgabe Nr. 25, Tabellen ändern - Spalten
Entfernen Sie die Spalte fahrzeit aus der Tabelle Erreicht. Fügen Sie die Spalte wieder in die Tabelle ein, wobei als Default 0 Minuten verwendet werden soll. Das Schlüsselwort SET soll nicht verwendet werden.

```
ALTER TABLE Erreicht
 DROP fahrzeit;
ALTER TABLE Erreicht
 ADD fahrzeit INTERVAL MINUTE DEFAULT '0';
```

Aufgabe Nr. 26, Wertebereichsänderungen

Ändere den Default-Wert des Wertebereiches GESCHLECHT_TYP von maennlich auf weiblich. Beachte die Bezeichnungen der Geschlechter aus der Wertebereischsdefinition.

```
ALTER DOMAIN GESCHLECHT_TYP
 SET DEFAULT 'Frau' ;
```

Aufgabe Nr. 27, Löschen von Datenbankobjekten - Wertebereiche

Lösche den Wertebereich GESCHLECHT_TYP, falls dieser nicht mehr in der DB verwendet wird.

```
DROP DOMAIN GESCHLECHT_TYP RESTRICT ;
```

Aufgabe Nr. 28, Löschen von Datenbankobjekten - Tabellen

Lösche die Tabelle Fahrer aus der DB.

```
DROP TABLE Fahrer ;
```

Aufgabe Nr. 29, Einfache SQL-Anfrage - Bedingung mit Gleichung

Welche Linien (linieNr) fahren samstags ('Sa')? Ergebnistyp sei (linieNr).

```
SELECT linieNr FROM Fahrt
WHERE 'Sa' = wochentag ;
```

Aufgabe Nr. 30, Einfache SQL-Anfrage- Bedingung mit Ungleichung

Welcher Fahrer ist vor dem 1.1.1965 geboren (ohne NOT)? Ergebnistyp sei (vorname, nachname, fahrerNr)

```
SELECT vorname , nachname , fahrerNr FROM Fahrer
WHERE '1965-1-1' > geborenAm ;
```

Aufgabe Nr. 31, Einfache SQL-Anfrage - Gesamttupel

Gib alle Kunde über die FahrerInnen in der Reihenfolge der Attribute (fahrerNr, nachname, vorname, geschlecht, kinderzahl, grundgehalt, kinderzulage, unfallzahl, zustand, einstellung, geborenAm) aus.

```
SELECT * FROM Fahrer ;
```

Aufgabe Nr. 32, Ergebnistabelle mit und ohne Duplikate

Ermittle unter Verwendung und Nicht-Verwendung der Duplikateliminierung, an welchen Tagen (anschaffungstag) neue Busse angeschafft wurden; zuerst allein zur Auflistung der betroffenen Tage, danach mit Rückschluss auf die Anzahl der an den Tagen stattfindenden Kaufhandlungen. Ergebnistyp sei

```
SELECT DISTINCT anschaffungstag FROM Bus ;
SELECT anschaffungstag FROM Bus ;
```

Aufgabe Nr. 33, Benennung von abgeleiteten Spalten

Ermittle das Alter aller Fahrer mit dem Ergebnistyp (alter, vorname, nachname, fahrerNr).

```
SELECT (CURRENT_DATE - geborenAm ) AS ALTER , vorname , nachname , fahrerNr
FROM Fahrer;
```

Aufgabe Nr. 34, Umbenennung von Spalten

In die Tabelle Fahrer ist eine zusätzliche Spalte 'zustand' augenommen, die angibt, ob ein Fahrer 'aktiv', 'beurlaubt', 'krank', 'entlassen' oder 'im Ruhestand' ist. Erstelle eine Liste der monatlichen Bezüge der Ruheständler mit den Spalten vorname, nachname, fahrerNr als personalNr und grundgehalt als

```
SELECT vorname , nachname , fahrerNr AS personalNr , grundgehalt AS betriebsrente
FROM Fahrer WHERE 'im Ruhestand' = zustand ;
```

Aufgabe Nr. 35, Arithmetische Ausdrücke in der SELECT-Klausel

Gebe jeden Fahrernamen zusammen mit seinem Gesamtgehalt aus. Ergebnistyp sei (vorname, nachname, gesamtgehalt).

```
SELECT vorname , nachname , grundgehalt + kinderzulage AS gesamtgehalt
FROM Fahrer ;
```

Aufgabe Nr. 36, Anfrage mit Aliasnamen

Wie hoch ist das höchste Grundgehalt der Fahrer? Der Ergebnis-Typ sei (hoechstGehalt).

```
SELECT MAX ( grundgehalt ) AS hoechstGehalt
FROM fahrer ;
```

Aufgabe Nr. 37, COUNT-Funktion

Ermittle die Anzahl aller Linien. Benennungen von Ergebnisspalten sind nicht vorzunehme

```
SELECT COUNT ( * )
FROM Linie ;
```

Aufgabe Nr. 38, COUNT-Funktion - Verwendung mit DISTINCT
Wir betrachten die Ausdrücke COUNT(fahrerNr) und COUNT(DISTINCT fahrerNr):
Nenne jene SELECT-Anfragen an die Tabelle Fahrer, welche mit einem der beiden Ausdrücke das gleiche Ergebnis erzielen wie 'SELECT COUNT(*) FROM Fahrer;'. Benennungen von Ergebnisspalten sind nicht

```
SELECT COUNT ( DISTINCT fahrerNr ) FROM Fahrer ;
SELECT COUNT ( fahrerNr ) FROM Fahrer ;
```

Aufgabe Nr. 39, COUNT-Funktion - Verwendung mit DISTINCT
Wir betrachten die Ausdrücke COUNT(fahrerNr) und COUNT(DISTINCT fahrerNr):
Nenne jene SELECT-Anfragen an die Tabelle Leitet, welche mit einem der beiden Ausdrücke das gleiche Ergebnis erzielen wie 'SELECT COUNT(*) FROM Leitet;'. Benennungen von Ergebnisspalten sind nicht

```
SELECT COUNT ( fahrerNr )
FROM Leitet ;
```

Aufgabe Nr. 40, MAX-Funktion
Bestimme den Fahrer mit den meisten Unfaellen. Benennungen von Ergebnisspalten sind nicht

```
SELECT MAX ( unfallzahl )
FROM Fahrer ;
```

Aufgabe Nr. 41, MIN-Funktion
Bestimme den Fahrer mit den wenigsten Kindern. Benennungen von Ergebnisspalten sind nicht

```
SELECT MIN ( kinderzahl )
FROM Fahrer ;
```

Aufgabe Nr. 42, SUM-Funktion
Die Kinder aller Fahrer werden zu einem Fest eingeladen. Für wieviele Kinder muss dieses Fest geplant werden, falls jeweils nur ein Elternteil einer Familie Fahrer ist? Benennungen von Ergebnisspalten sind nicht

```
SELECT SUM ( kinderzahl )
FROM Fahrer ;
```

Aufgabe Nr. 43, AVG-Funktion
Ermittle das Durchschnittsgehalt aller Fahrer. Benennungen von Ergebnisspalten sind nicht vorzunehmen.

```
SELECT AVG ( grundgehalt )
FROM Fahrer ;
```

Aufgabe Nr. 44, Gemeinsame Verwendung von DISTINCT-Klausel und Aggregatfunktion
An wie vielen Tagen finden Fahrten statt? Benennungen von Ergebnisspalten sind nicht vorzunehmen.

```
SELECT COUNT ( DISTINCT tag ) FROM Fahrt ;
```

Aufgabe Nr. 45, Natürlicher Verbund
Für welche Linien werden die einzelnen Fahrer eingesetzt?
Erzeuge eine Ergebnistabelle vom Typ (fahrerNr, linieNr). Verwende die neue Syntax.

```
SELECT fahrerNr , linieNr
FROM Fahrt NATURAL JOIN Leitet ;
```

Aufgabe Nr. 46, Natürlicher Verbund vieler Tabellen
Fahrer Schmidt möchte die Namen aller Haltestellen wissen, die er anfährt.
Nenne eine SQL-Anfrage, die ausschleisslich NATURAL JOIN benutzt. Verwende dazu nicht mehr als 5 Tabellen. Der Ergebnistyp sei (haltestelleName).

```
SELECT haltestelleName
FROM Erreicht
 NATURAL JOIN Fahrer
 NATURAL JOIN Fahrt
 NATURAL JOIN Haltestelle
 NATURAL JOIN Leitet
WHERE 'Schmidt' = nachname ;
```

Aufgabe Nr. 47, Natürlicher Verbund - Gefahren
Fahrer Schmidt möchte die Namen aller Haltestellen wissen, die er anfährt.
Die Geschaeftsleitung moechte sich ein Bild vom Zustand aller Haltestellen machen koennen, so dass auch die Tabelle Haltestelle eine zusaetzliche Spalte 'zustand' mit den moeglichen Werten 'gut', 'mittel', 'schlecht' und 'normal' bekommt. Bei einer Anfrage wie in Aufgabe 46 musste Herr Schmidt feststellen, dass er angeblich keine Haltestellen anfaehrt. Berichtige diesen Fehler in der Schema-Definition durch Umbenennung aller betroffenen Attribute in die Gestalt [tabellenname][Alter Attributname].

```
ALTER TABLE Fahrer RENAME zustand TO fahrerZustand ;
ALTER TABLE Haltestelle RENAME zustand TO haltestelleZustand ;
```

Aufgabe Nr. 48, Natürlicher Verbund alter Art

Für welche Linien werden die einzelnen Fahrer eingesetzt?
Erzeuge eine Ergebnistabelle vom Typ (fahrerNr, linieNr). Verwende eine Anfrage nach alter Syntax.

```
SELECT fahrerNr , linieNr FROM Fahrt , Leitet
WHERE Fahrt .fahrtNr = Leitet .fahrtNr ;
```

Aufgabe Nr. 49, Wegfall übereinstimmender Spalten beim natürlichen Verbund alter Art

Für welche Linien werden die einzelnen Fahrer eingesetzt?
Stelle die Anfrage in alter Syntax so, dass der Ergebnistyp (fahrerNr, fahrtNr, linieNr) entsteht.

```
SELECT fahrerNr , Fahrt .fahrtNr , linieNr FROM Fahrt , Leitet
WHERE Fahrt .fahrtNr = Leitet .fahrtNr ;
```

Aufgabe Nr. 50, NATURAL JOIN-Klausel

Zu welchen Uhrzeiten (als 'anfahrzeit') wird die Haltestelle 'Schillerstrasse' von welcher Linie angefahren?
Formuliere die Anfrage mit dem natürlichen Verbund neuer Schreibweise.

```
SELECT fahrzeit + uhrzeitBeginn AS anfahrzeit , linieNr
FROM Erreicht NATURAL JOIN Fahrt NATURAL JOIN Haltestelle
WHERE 'Schillerstrasse' = haltestelleName ;
```

Aufgabe Nr. 51, INNER JOIN über die USING-Klausel

Zu welchen Uhrzeiten (als 'anfahrzeit') wird die Haltestelle 'Schillerstrasse' von welcher Linie angefahren?
Formuliere die Anfrage mit dem inneren Verbund in Verbindung mit der USING-Klausel.

```
SELECT fahrzeit + uhrzeitBeginn AS anfahrzeit , linieNr
FROM Erreicht JOIN Fahrt USING ( linieNr ) JOIN Haltestelle USING ( haltestelleNr )
WHERE 'Schillerstrasse' = haltestelleName ;
```

Aufgabe Nr. 52, INNER JOIN über die ON-Klausel

Zu welchen Uhrzeiten (als 'anfahrzeit') wird die Haltestelle 'Schillerstrasse' von welcher Linie angefahren?
Formuliere die Anfrage mit dem inneren Verbund in Verbindung mit der ON-Klausel.

```
SELECT fahrzeit + uhrzeitBeginn AS anfahrzeit , Fahrt .linieNr
FROM Erreicht
 JOIN Fahrt ON ( Erreicht .linieNr = Fahrt .linieNr )
 JOIN Haltestelle ON ( Erreicht .haltestelleNr = Haltestelle .haltestelleNr )
WHERE 'Schillerstrasse' = haltestelleName ;
```

Aufgabe Nr. 53, CROSS JOIN-Klausel

Zu welchen Uhrzeiten (als 'anfahrzeit') wird die Haltestelle 'Schillerstrasse' von welcher Linie angefahren?
Formuliere die Anfrage mit dem kartesischen Produkt neuer Schreibweise.

```
SELECT fahrzeit + uhrzeitBeginn AS anfahrzeit , Fahrt .linieNr
FROM Erreicht CROSS JOIN Fahrt CROSS JOIN Haltestelle
WHERE 'Schillerstrasse' = haltestelleName
 AND Erreicht .haltestelleNr = Haltestelle .haltestelleNr
 AND Fahrt .linieNr = Erreicht .linieNr ;
```

Aufgabe Nr. 54, Alte und neue Variante des kartesischen Produktes

Welche Kombinationen von Fahrer und Fahrt sind möglich? Es kann davon ausgegangen werden, dass alle Fahrer auf allen Fahrten eingesetzt werden können. Verwenden Sie bei der Anfrage erst die alte und dann die neue Variante des kartesischen Produktes. Die Ergebnistabelle soll vom Typ (fahrerNr, fahrtNr) sei aus den Tabellen fahrer und fahrt zu bilden.

```
SELECT fahrerNr , fahrtNr FROM Fahrer , Fahrt ;
SELECT fahrerNr , fahrtNr FROM Fahrer CROSS JOIN Fahrt ;
```

Aufgabe Nr. 55, Einfacher Verbund über USING-Klausel

Welche Haltestellen werden von welcher Linie angefahren? Der Ergebnistyp sei (linieNr, haltestelleName).

```
SELECT linieNr , haltestelleName
FROM Erreicht JOIN Haltestelle USING ( haltestelleNr ) ;
```

Aufgabe Nr. 56, Bedingungsverbund über nicht namensgleiche Spalten

Wie heißen die Start-Haltestellen der jeweiligen Linien? Gib die Liniennummer und den Namen der

```
SELECT linieNr , haltestelleName
FROM Haltestelle JOIN Linie ON ( haltestelleNr = linieBeginnHaltestelleNr ) ;
```

Aufgabe Nr. 57, Nicht auf Gleichheit beruhender Bedingungsverbund
Gib für alle Linien ihre Haltestellennummern aus mit Ausnahme der Starthaltestellen. (Mit ON-Klammerung.)

```
SELECT linieNr , haltestelleNr
FROM Erreicht JOIN Linie ON (
 Erreicht .linieNr = Linie .linieNr
 AND
 haltestelleNr <> linieBeginnHaltestelleNr
) ;
```

Aufgabe Nr. 58, Äußerer Verbund
Die Zentrale braucht Angaben über die Besetzung der einzelnen Fahrtnummern aller Fahrten für den Fall, dass Fahrten augenblicklich zu disponieren sind. Ergebnis-Typ sei das gesamte Fahrt-Tupel in ursprünglicher Reihenfolge, gefolgt von fahrerNr. Insbesondere sollen auch Fahrten angezeigt werden, bei welchen noch kein Fahrer fest steht.

```
SELECT Fahrt .* , fahrerNr
FROM Fahrt LEFT JOIN Leitet ON ( Fahrt .fahrtNr = Leitet .fahrtNr ) ;
```

Aufgabe Nr. 59, Äußerer Verbund
Die Zentrale braucht Angaben über die Besetzung der einzelnen Fahrtnummern aller Fahrten für den Fall, dass Fahrten augenblicklich zu disponieren sind. Ergebnis-Typ sei das gesamte Fahrt-Tupel in ursprünglicher Reihenfolge, gefolgt von fahrerNr. Insbesondere sollen auch Fahrten angezeigt werden, bei welchen noch kein Fahrer fest steht.
Gib zusätzlich zu den Angaben auch die Fahrtnummern aus, für die noch kein Fahrer feststeht. Verwenden

```
SELECT Fahrt .* , fahrerNr
FROM Fahrt NATURAL LEFT JOIN Leitet ;
```

Aufgabe Nr. 60, Einfacher Verbund über mehrere Tabellen
An welchen Tagen werden die Fahrer (Vorname, Nachname, Nummer, Tag) eingesetzt?

```
SELECT DISTINCT vorname , nachname , fahrerNr , tag
FROM Fahrer NATURAL JOIN Fahrt NATURAL JOIN Leitet ;
```

Aufgabe Nr. 61, Verbund einer Tabelle mit sich selber
Welche Fahrer haben das gleiche Grundgehalt? Verwende die Tabellen-Aliasnamen Fahrer1 und Fahrer2 sowie den Ergebnis-Typen (Fahrer1.fahrerNr, Fahrer2.fahrerNr, Fahrer1.grundgehalt).
Jedes Paar soll nur einmal (verwende Prädikat 'Fahrer1.fahrerNr < Fahrer2.fahrerNr') angezeigt werden.

```
SELECT Fahrer1 .fahrerNr , Fahrer2 .fahrerNr , Fahrer1 .grundgehalt
FROM Fahrer AS Fahrer1 JOIN Fahrer AS Fahrer2 ON (
 Fahrer1 .fahrerNr < Fahrer2 .fahrerNr
 AND
 Fahrer1 .grundgehalt = Fahrer2 .grundgehalt
) ;
```

Aufgabe Nr. 62, Vergleichsprädikat
Welcher Fahrer (Vorname, Nachname, Nummer) erhaelt weniger Grundgehalt als 200.000 Euro?

```
SELECT vorname , nachname , fahrerNr FROM Fahrer
WHERE 200000 > grundgehalt ;
```

Aufgabe Nr. 63, Komplexe Bedingung
Um welche Uhrzeit genau (als 'ankunftZeit' und an welcher Haltestelle (Name) enden Fahrten der einzelnen Linien der einzelnen Tage nach Mitternacht (bis 3:00)? Verwende den Ergebnistypen (Linie.linieNr, aufbruchstag, haltestelleName, ankunftZeit). Verwende den Verbund nach alter Art. Jede Tabelle ist so umzubenennen, dass ihr Aliasname der Anfangsbuchstabe als Grossbuchstabe ist. Beispiel: '... Fahrt AS F

```
SELECT Linie .linieNr , tag AS aufbruchstag , haltestelleName , fahrzeit + uhrzeitBeginn AS ankunftZeit
FROM Erreicht AS E , Fahrt AS F , Haltestelle AS H , Linie AS L
WHERE E .haltestelleNr = H .haltestelleNr AND E .haltestelleNr = linieEndHaltestelleNr
 AND E .linieNr = F .linieNr AND E .linieNr = L .linieNr
 AND fahrzeit + uhrzeitBeginn BETWEEN '0:00' AND '3:00' ;
```

Aufgabe Nr. 64, IS NOT NULL-Prädikat
Wie wäre die Bedingung für attr1 und attr2 getrennt auszudrücken (ohne Klammern)?
 WHERE NOT (attr1,attr2) IS NOT NULL. Es ist nur die Selektionsbedingung ohne Semikolon zu verwenden

```
WHERE attr1 IS NULL OR attr2 IS NULL
```

Aufgabe Nr. 65, GROUP BY-Klausel
Gib für jede Linie die Anzahl der Haltestellen aus. Ergebnis-Typ sei (linieNr, anzahlHaltestellen).

```
SELECT linieNr , COUNT ( haltestelleNr ) AS anzahlHaltestellen
FROM erreicht GROUP BY linieNr ;
```

Aufgabe Nr. 66, Sortieren über die ORDER BY-Klausel
Gib das Grundgehalt jedes Fahrers aufsteigend sortiert aus. Ergebnis-Typ sei (vorname, nachname,

```
SELECT vorname , nachname , fahrerNr , grundgehalt
FROM fahrer ORDER BY grundgehalt ;
```

Aufgabe Nr. 67, HAVING-Klausel
Gib für die Linien 88 und 99 die Anzahl ihrer Haltestellen aus. Ergebnis-Typ sei (linieNr, anzahlHaltestellen). Verwende dabei die IN-Klausel.

```
SELECT linieNr , COUNT ( haltestelleNr ) AS anzahlHaltestellen FROM Erreicht
GROUP BY linieNr HAVING linieNr IN ( 88 , 99 ) ;
```

Aufgabe Nr. 68, HAVING-Klausel
Fuer jeden Bus soll ermittelt werden, bei wie vielen Fahrten (als 'anzahlFahrten') er eingesetzt wird, welche genau um 8:00 bgeinnen. Die folgende Anfrage ist falsch: SELECT busNr, COUNT(fahrtNr) AS anzahlFahrten FROM Fahrt GROUP BY busNr HAVING uhrzeitBeginn = TIME '8:00'
Wie hat die Anfrage tatsächlich zu lauten? Verwende einen Typecast für die Zeit und keine Sekundenangabe sowie den Ergebnis-Typ (busNr, anzahlFahrten).

```
SELECT busNr , COUNT ( fahrtNr ) AS anzahlFahrten FROM Fahrt
WHERE '8:00' = uhrzeitBeginn GROUP BY busNr ;
```

Aufgabe Nr. 69, HAVING-Klausel
Fuer jeden Bus soll ermittelt werden, bei wie vielen Fahrten (als 'anzahlFahrten') er eingesetzt wird, welche genau um 8:00 bgeinnen. Die folgende Anfrage ist falsch: SELECT busNr, COUNT(fahrtNr) AS anzahlFahrten FROM Fahrt GROUP BY busNr HAVING uhrzeitBeginn = TIME '8:00'
Wie kann man diese Anfrage abändern, so dass die Anzahl der Fahrten jener Busse angezeigt werden, die AUCH vor 8:00 aufbrechen (diese dürfen nicht überlastet sein, da vor dieser Zeit Pannen nicht durch den hauseigenen Service behoben werden können)? Verwende einen Typecast für die Zeit und keine Sekundenangabe sowie den Ergebnis-Typ (busNr, anzahlFahrten).

```
SELECT busNr , COUNT ( fahrtNr ) AS anzahlFahrten FROM Fahrt
GROUP BY busNr HAVING '8:00' = MAX ( uhrzeitBeginn ) ;
```

Aufgabe Nr. 70, HAVING-Klausel mit Aggregatfunktionen
Gib die Namen aller Haltestellen aus, die von mehr als fünf Linien angefahren werden.

```
SELECT haltestelleName FROM Erreicht NATURAL JOIN Haltestelle
GROUP BY haltestelleNr HAVING 5 < COUNT ( linieNr ) ;
```

Aufgabe Nr. 71, Alternative Verwendung von WHERE- und HAVING-Klausel
Von wie vielen Linien wird die Haltestelle 'Konsumtempel' angefahren? Die sich ergebende Anzahl sei unbenannt (ohne Alias) zurückzugeben.
Verwende WHERE.

```
SELECT COUNT ( linieNr ) FROM Erreicht NATURAL JOIN Haltestelle
WHERE 'Konsumtempel' = haltestelleName GROUP BY haltestelleName ;
```

Aufgabe Nr. 72, Alternative Verwendung von WHERE- und HAVING-Klausel
Von wie vielen Linien wird die Haltestelle 'Konsumtempel' angefahren? Die sich ergebende Anzahl sei unbenannt (ohne Alias) zurückzugeben.
Verwende HAVING.

```
SELECT COUNT ( linieNr ) FROM Erreicht NATURAL JOIN Haltestelle
GROUP BY haltestelleName HAVING 'Konsumtempel' = haltestelleName ;
```

Aufgabe Nr. 73, HAVING-Klausel mit Aggregatfunktion
Welche Busse müssen montags ('Mo') vor 11:00 Uhr einsatzbereit sein? Nenne diese Zeit des Einsatzbeginns ('einsatzBeginn') mit, verwende den Ergebnistyp (busNr,einsatzBeginn). Setze dabei auch WHERE ein, wo möglich. Verwende einen Typecast für die Zeit, keine Sekundenangabe und den Ergebnis-

```
SELECT busNr , MIN ( uhrzeitBeginn ) AS einsatzBeginn FROM Fahrt
WHERE 'Mo' = wochentag GROUP BY busNr HAVING '11:00' > MIN ( uhrzeitBeginn ) ;
```

Aufgabe Nr. 74, Mengenwertige Spalten und Duplikate
Wie viele Haltestellen werden an den verschiedenen Wochentagen angefahren? Der Rückgabetyp sei (wochentag, anzahlHaltestellen). Verwende NATURAL JOIN.

```
SELECT wochentag , COUNT ( DISTINCT haltestelleNr ) AS anzahlHaltestellen
FROM Erreicht NATURAL JOIN Fahrt GROUP BY wochentag ;
```

Aufgabe Nr. 75, Mengenwertige Spalten und Duplikate

Auf wieviele verschiedene Fahrer ist die Leitung der verschiedenen Busse verteilt? Verwende NATURAL JOIN. Der Rückgabetyp sei (busNr, anzahlFahrer).

```
SELECT busNr , COUNT ( DISTINCT fahrerNr ) AS anzahlFahrer
FROM Fahrt NATURAL JOIN Leitet GROUP BY busNr ;
```

Aufgabe Nr. 76, Unteranfrage in der WHERE-Klausel

Welche Fahrer haben die meisten Kinder? Ergebnistyp sei (vorname, nachname, fahrerNr, kinderzahl).

```
SELECT vorname , nachname , fahrerNr , kinderzahl FROM Fahrer
WHERE ( SELECT MAX ( kinderzahl ) FROM Fahrer ) = kinderzahl ;
```

Aufgabe Nr. 77, Unteranfrage in der HAVING-Klausel

Welche Haltestellen werden von allen Linien angefahren? Ergebnis-Typ sei (haltestelleName).

```
SELECT haltestelleName FROM Erreicht NATURAL JOIN Haltestelle
GROUP BY haltestelleNr
HAVING ( SELECT COUNT ( linieNr ) FROM Linie ) = COUNT ( DISTINCT linieNr ) ;
```

Aufgabe Nr. 78, Unteranfrage in der FROM-Klausel

Wieviele Haltestellen (verwende 'anzahlHaltestellen') werden von den Linien maximal angefahren? Ergebnistyp sei (maxAnzahl).

```
SELECT MAX ( anzahlHaltestellen ) AS maxAnzahl
FROM ( SELECT COUNT ( * ) AS anzahlHaltestellen FROM Erreicht GROUP BY linieNr ) ;
```

Aufgabe Nr. 79, Sichere wertliefernde Unteranfragen

Wieviele Fahrer (Vername, Name, Fahrernummer) haben mehr Kinder (Anzahl bitte angeben) als Fahrer Meier (Fahrernummer 101)= Ergebnis-Typ sei (vorname, nachname, fahrerNr, kinderzahl).

```
SELECT vorname , nachname , fahrerNr , kinderzahl FROM Fahrer
WHERE ( SELECT kinderzahl FROM Fahrer WHERE 101 = fahrerNr ) < kinderzahl ;
```

Aufgabe Nr. 80, Einfache geschachtelte Unteranfrage

Welche Linie führt die größte Anzahl von Haltestellen an? Der Ergebnis-Typ sei (linieNr).
Verwende in der FROM Klausel eine Unterabfrage gleichen Typs, welche mit Hilfe der Ermittlung von max(anzahlHaltestellen) gleichermassen selektiert wird.

```
SELECT linieNr
FROM (
 SELECT COUNT ( haltestelleNr ) AS anzahlHaltestellen , linieNr
 FROM Erreicht GROUP BY linieNr
) WHERE (
 SELECT MAX ( anzahlHaltestellen )
 FROM (
  SELECT COUNT ( haltestelleNr ) AS anzahlHaltestellen
  FROM Erreicht GROUP BY linieNr
 )
 ) = anzahlHaltestellen ;
```

Aufgabe Nr. 81, Korrelierende Unteranfrage

Welche Fahrer wurden gleichzeitig mit genau einem weiteren Fahrer eingestellt? Verwende dazu eine korrelierende Anfage mit den entsprechenden Tabellen-Aliasnamen 'Selbst' und 'Belegschaft' ohne NOT

```
SELECT fahrerNr FROM Fahrer AS Selbst
WHERE (
 SELECT COUNT ( * ) FROM Fahrer AS Belegschaft
 WHERE Belegschaft .einstellung = Selbst .einstellung
```

Aufgabe Nr. 82, IN-Prädikat mit Unteranfrage

Welche Fahrer werden am 24.12.2001 eingesetzt? Verwende NATURAL JOIN und den Typecast DATE. Ergebnis-Typ sei (vorname, nachname, fahrerNr).

```
SELECT vorname , nachname , fahrerNr FROM Fahrer NATURAL JOIN Leitet
WHERE fahrtNr IN ( SELECT fahrtNr FROM Fahrt WHERE '2001-12-24' = tag ) ;
```

Aufgabe Nr. 83, IN-Prädikat und Zeilen

Nenne Frauen ('Frau') mit vier oder fuenf Kindern unter den Fahrern. Ergebnis-Typ sei (vorname, nachname, fahrerNr). Die WHERE-Klausel darf nur aus genau einer IN-Klausel bestehen, in der Kinderzahl

```
SELECT vorname , nachname , fahrerNr FROM Fahrer
WHERE ( geschlecht , kinderzahl ) IN ( ( 'Frau' , 4 ) , ( 'Frau' , 5 ) ) ;
```

Aufgabe Nr. 84, Simulation eines IN-Prädikates auf Zeilen

Zeige mögliche Fahrten der Busse 2 und 3 auf den Linien 77 oder 88. Ergebnis-Typ sei (busNr, linieNr).

```
SELECT busNr , linieNr FROM Fahrt WHERE busNr IN ( 2 , 3 ) AND linieNr IN ( 77 , 88 ) ;
```

Aufgabe Nr. 85, Vergleichbarkeit von ANY- und IN-Prädikat

Welche Fahrer müssen Heiligabend 2001 arbeiten? Verwende den Ergebnis-Typ (fahrerNr) und den Typecast DATE.

```
SELECT fahrerNr FROM Leitet
WHERE ANY ( SELECT fahrtNr FROM Fahrt WHERE '2001-12-24' = tag ) = fahrtNr ;
```

Aufgabe Nr. 86, Vergleichbarkeit von ANY- und IN-Prädikat

Welche Fahrer müssen Heiligabend 2001 arbeiten? Verwende den Ergebnis-Typ (fahrerNr) und den Typecast DATE.

```
SELECT fahrerNr FROM Leitet
WHERE fahrtNr IN ( SELECT fahrtNr FROM Fahrt WHERE '2001-12-24' = tag ) ;
```

Aufgabe Nr. 87, ALL-Prädikat

Welche Fahrten besuchen nur Haltestellen in gutem Zustand? Verwende ALL mit einem natürlichen Verbund 'LinienHaltestellen' und den Ergebnis-Typ (fahrtNr).

```
SELECT fahrtNr FROM Fahrt
WHERE 'gut' = ALL (
 SELECT haltestelleZustand
 FROM Erreicht NATURAL JOIN Haltestelle AS LinienHaltestellen
 WHERE Fahrt .linieNr = LinienHaltestellen .linieNr
) ;
```

Aufgabe Nr. 88, EXISTS-Prädikat

Welche Fahrer müssen am 24.12.2001 arbeiten? Verwende den Typecast DATE und den Ergebnis-Typ

```
SELECT fahrerNr FROM Leitet
WHERE EXISTS (
 SELECT * FROM Fahrt
 WHERE '2001-12-24' = tag AND Fahrt .fahrtNr = Leitet .fahrtNr
) ;
```

Aufgabe Nr. 89, Vereinigung

Welche Fahrer wurden am 22.1.2001 oder am 23.1.2001 eingestellt? Verwende UNION mit Typecast DATE und Ergebnis-Typ (vorname, nachname, fahrerNr).

```
SELECT vorname , nachname , fahrerNr FROM Fahrer WHERE '2001-1-22' = einstellung
UNION
SELECT vorname , nachname , fahrerNr FROM Fahrer WHERE '2001-1-23' = einstellung ;
```

Aufgabe Nr. 90, ORDER BY-Klausel

Gib die auf den Fahrten eingesetzten Busse sortiert erst nach Busnummer, dann Tag, aus. Ergebnis-Typ sei

```
SELECT DISTINCT busNr , tag FROM fahrt ORDER BY busNr , tag ;
```

Aufgabe Nr. 91, Beispiel einer umfassenden SQL-Anfrage

Der Fahrer Michael Showmacher (wir wissen, dass es nur einen Fahrer dieses Namens in unserem Unternehmen gibt) soll für den 13. bis 15. Juli 2001 die vollen Strecken fahren, da sein Kollege ein Wochenendseminar besucht Er möchte chronologisch aufgereiht wissen, mit welchem Bus er welche Linie fahren soll. Dazu braucht er sowohl die die Zeitvorgabe für die Abfahrt von der Starthaltestelle und jene der Ankunft an der Endhaltestelle (die Linien selbst kennt er auswendig). Verwende NATURAL JOIN, außerdem sollen alle in der SELECT-Klausel verwendeten Attribute auch in der GROUP BY-Klausel erscheinen.

```
SELECT tag , uhrzeitBeginn , MAX ( fahrzeit ) + uhrzeitBeginn AS uhrzeitEnde , linieNr , busNr
FROM Erreicht NATURAL JOIN Fahrer NATURAL JOIN Fahrt NATURAL JOIN Leitet
WHERE 'Michael' = vorname AND 'Showmacher' = nachname
GROUP BY busNr , linieNr , tag , uhrzeitBeginn
HAVING tag IN ( '2001-7-13' , '2001-7-14' , '2001-7-15' )
ORDER BY tag , uhrzeitBeginn ;
```

Aufgabe Nr. 92, Neue Zeile einfügen
Ein neuer Fahrer, Roland Wohlfahrt, wurde heute eingestellt. Er soll die Fahrernummer 33 erhalten und sofort die Fahrt 555 auf ganzer Strecke von anderen Fahrern übernehmen.
Nenne die SQL-Anfrage. Die Einfügung soll mittels einer ausdrücklichen Festlegung er zu setzenden Attribute innerhalb der INSERT-Klausel erfolgen.

```
INSERT INTO Fahrer ( einstellung , fahrerNr , geschlecht, nachname , vorname )
 VALUES ( CURRENT_DATE , 33 , 'Mann', 'Wohlfahrt' , 'Roland' ) ;
UPDATE Leitet SET fahrerNr = 33 WHERE 555 = fahrtNr ;
```

Aufgabe Nr. 93, Neue Zeilen einfügen
Die Fahrt der Nummer 888 soll durch den Einsatz eines zusätzlichen Busses mit Nummer 44 unterstützt werden. Die zusätzliche Fahrt bekommt die Fahrtnummer 889.

```
INSERT INTO Fahrt
SELECT 889 AS fahrtNr , linieNr, 44 AS busNr , uhrzeitBeginn , wochentag, tag
FROM Fahrt WHERE 888 = fahrtNr ;
```

Aufgabe Nr. 94, Zwei Anfragen, verknüpft über eine temporäre Tabelle
Welche Linie führt die größte Anzahl von Haltestellen an? Der Ergebnis-Typ sei (linieNr).
Wie wir gesehen haben, ist es nicht einfach, dies mit einer einzelnen Anfrage auszudrücken. Diese kann übersichtlicher durch Bildung und Auffüllung einer temporären Tabelle ('HaltestellenzahlEinerLinie' mit einem einzigen SMALLINT-Attribut anzahlHaltestellen) und anschließender Verwendung in einer HAVING-

```
CREATE TABLE HaltestellenzahlEinerLinie ( anzahlHaltestellen SMALLINT ) ;
INSERT INTO HaltestellenzahlEinerLinie
 SELECT COUNT ( * ) AS anzahlHaltestellen FROM Erreicht GROUP BY linieNr ;
SELECT linieNr FROM Erreicht
GROUP BY linieNr
HAVING ( SELECT MAX ( anzahlHaltestellen ) FROM HaltestellenzahlEinerLinie ) = COUNT ( * ) ;
```

Aufgabe Nr. 95, Verfeinerung von neuen Zeilen über die UPDATE-Anweisung
Ein neuer Fahrer, Roland Wohlfahrt, wurde heute eingestellt. Er soll die Fahrernummer 33 erhalten und sofort die Fahrt 555 auf ganzer Strecke von anderen Fahrern übernehmen.
Inzwischen sind nach der Aufnahme in die Datenbank das Geburtsdatum (30.5.1961) und die Kinderzahl (4) von Herrn Wohlfahrt bekannt und sollen entsprechend nachgetragen werden.

```
UPDATE Fahrer SET geborenAm = '1961-5-30' , kinderzahl = 4 WHERE 33 = fahrerNr ;
```

Aufgabe Nr. 96, Abändern von Zeilen
In der Tabelle Leitet sollen NULL-Werte in den Spalten 'leitetVonHaltestelleNr' durch die entsprechenden Anfangs-Haltestellen der Tabelle linie ersetzt werden. Dazu soll ein unbenannter natürlicher Verbund

```
UPDATE Leitet SET leitetVonHaltestelleNr = linieBeginnHaltestelleNr FROM Fahrt NATURAL JOIN Linie
WHERE Fahrt .fahrtNr = Leitet .fahrtNr AND leitetVonHaltestelleNr IS NULL ;
```

Aufgabe Nr. 97, Abänderungsoperation und Aggregatfunktionen
Die letzte Haltestelle von Linie Nummer 17 soll zwei Minuten später angefahren werden. Sie ist noch nicht unter 'linieEndHaltestelleNr' in der Tabelle Linie eingetragen.

```
UPDATE Erreicht SET fahrzeit = '2' + fahrzeit
WHERE ( SELECT MAX ( fahrzeit ) FROM Erreicht WHERE 17 = linieNr ) = fahrzeit AND 17 = linieNr ;
```

Aufgabe Nr. 98, Zeile löschen
Die Fahrt mit Nummer 1818 soll ersatzlos gestrichen werden.

```
DELETE FROM Fahrt WHERE fahrtNr = 1818 ;
```

Aufgabe Nr. 99, Löschen aller Zeilen einer Tabelle
Die temporäre Tabelle HaltestellenzahlEinerLinie beansprucht Platz auf der Festplatte. Daher sollen sämtliche Zeilen dieser Tabelle gelöscht werden.

```
DELETE FROM HaltestellenzahlEinerLinie ;
```

Aufgabe Nr. 100, Vollständiges Entfernen einer Tabelle
Die temporäre Tabelle HaltestellenzahlEinerLinie wird nicht mehr benötigt und soll daher aus der DB entfernt

```
DROP TABLE HaltestellenzahlEinerLinie ;
```

Aufgabe Nr. 101, Löschen mit komplexer Anfrage
Linie Nummer 7 ist unrentabel geworden und wird eingestellt. Daher müssen u.a. aus der Tabelle Leitet alle betroffenen Zeilen entfernt werden.

```
DELETE FROM Leitet
WHERE fahrtNr IN ( SELECT fahrtNr FROM Fahrt WHERE 7 = linieNr ) ;
```

Aufgabe Nr. 102, Einfache Sichtendefinition

Erstelle eine Sicht FahrtMitUhrzeit fuer Fahrten an Heiligabend 2001, die linieNr, uhrzeitBeginn und busNr

```
CREATE VIEW FahrtMitUhrzeit AS SELECT linieNr , uhrzeitBeginn , busNr
FROM Fahrt WHERE '2001-12-24' = tag ;
```

Aufgabe Nr. 103, Komplexe Sichtendefinition

Die Frau des Fahrers Meier (einfach Fahrernummer 101 verwenden) stellt mindestens dreimal pro Woche die Frage, von wann bis wann ihr Mann morgen arbeiten muss. Um diese Fragen schneller beantworten zu koennen, soll eine Sicht MeiersFahrten definiert werden, die Tag, Uhrzeit, Haltestellen-Namen und Liniennummer seiner Fahrten chronologisch geordnet beinhaltet.
Formuliere diese Sicht und die Anfrage darauf. (Kein Fahrer wird mehrmals auf einer Fahrt eingesetzt).
Verwende NATURAL JOIN, ausserdem kann der Aliasname 'Verbund' für einen Tabellenverbund benutzt

```
CREATE VIEW MeiersFahrten AS
SELECT tag , fahrzeit + uhrzeitBeginn AS uhrzeit , haltestelleName , linieNr
FROM ( Erreicht NATURAL JOIN Fahrt NATURAL JOIN Haltestelle NATURAL JOIN Leitet ) AS Verbund
WHERE 101 = fahrerNr
 AND fahrzeit BETWEEN (
  SELECT Erreicht .fahrzeit FROM Erreicht
  WHERE Erreicht .haltestelleNr = leitetVonHaltestelleNr
   AND Erreicht .linieNr = Verbund .linieNr
 ) AND (
  SELECT Erreicht .fahrzeit FROM Erreicht
  WHERE Erreicht .haltestelleNr = leitetBisHaltestelleNr
   AND Erreicht .linieNr = Verbund .linieNr
 )
ORDER BY tag , uhrzeit ;
```

Aufgabe Nr. 104, Rechtevergabe

Die Frau des Fahrers Meier (einfach Fahrernummer 101 verwenden) stellt mindestens dreimal pro Woche die Frage, von wann bis wann ihr Mann morgen arbeiten muss. Um diese Fragen schneller beantworten zu koennen, soll eine Sicht MeiersFahrten definiert werden, die Tag, Uhrzeit, Haltestellen-Namen und Liniennummer seiner Fahrten chronologisch geordnet beinhaltet.
Frau Meier darf unter der Kennung 'Frau_Meier' ihre Anfragen selber an die DB stellen.

```
GRANT SELECT ON MeiersFahrten TO Frau_Meier;
```

Aufgabe Nr. 105, Sekundärindex

Für die Tabelle fahrt sollen Indices, möglichst UNIQUE, über die Spalten fahrtNr ('fahrtIdx') sowie tag und uhrzeitBeginn gemeinsam ('fahrtBeginnIdx') erstellt werden.

```
CREATE UNIQUE INDEX fahrtIdx   ON Fahrt ( fahrtNr ) ;
CREATE INDEX fahrtBeginnIdx   ON Fahrt ( tag , uhrzeitBeginn ) ;
```