Hochschule für Technik und Wirtschaft Berlin
University of Applied Sciences

Datenbanken

Übung 2

Lösungsvorschlag

Dozent: M. Sc. Burak Boyaci

Version: 03. November 2025

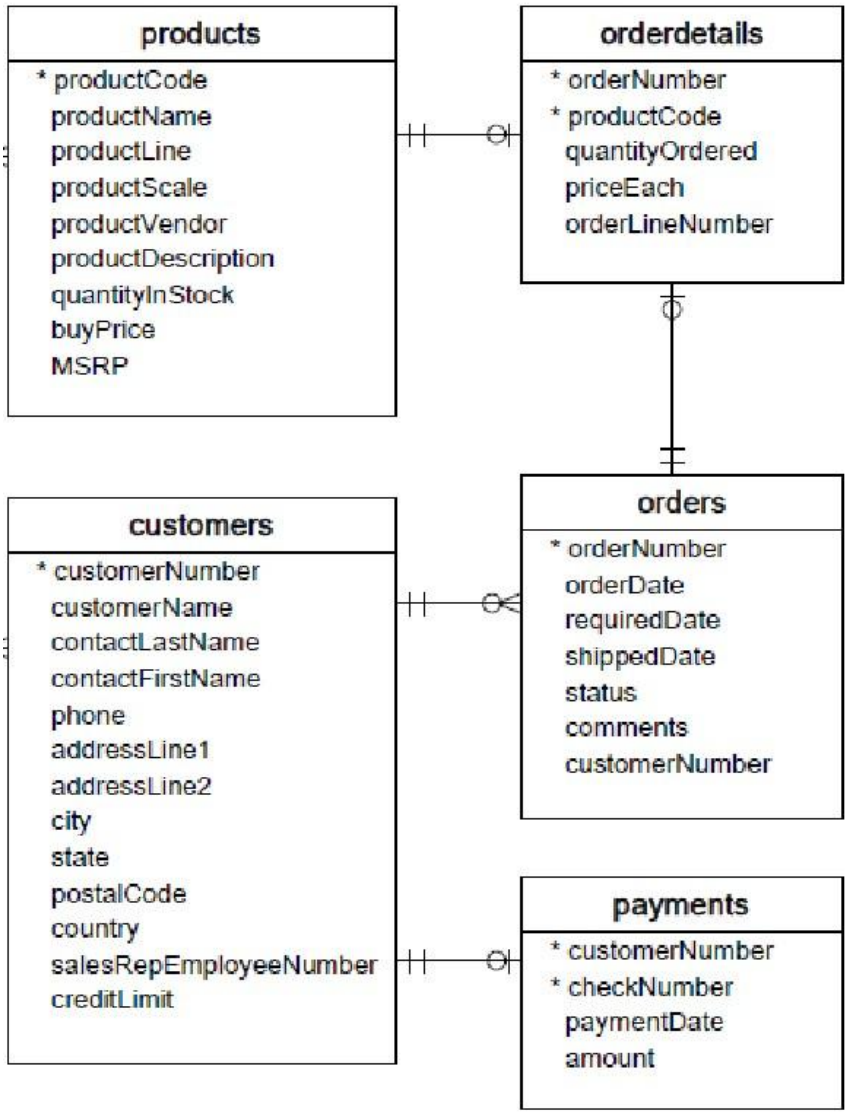
Wintersemester 25/26

1. Öffnen Sie pg4Admin 4 und verbinden Sie sich mit Ihrer Datenbank.

2. Gehen Sie in Ihre Datenbank und erstellen Sie ein neues Schema *classicmodels*.

3. Nachfolgend sehen Sie ein sogenanntes Datenmodell. Ein Datenmodell ist eine strukturierte Darstellung von Daten und deren Beziehungen zueinander. Ein Datenmodell dient dazu, Informationen systematisch zu organisieren, sodass sie von Datenbanken effizient gespeichert, verarbeitet und abgerufen werden können.

Datenmodell:



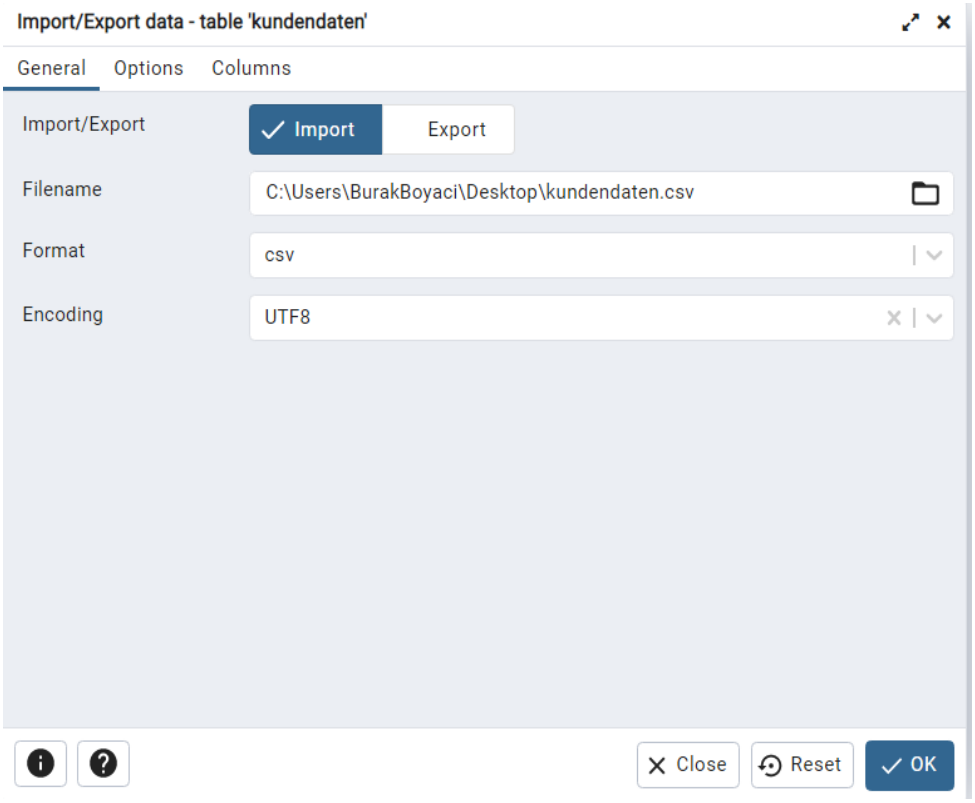
Beschreibung der Tabellen:

- **Customers**: stores customer's data.
- **Products**: stores a list of scale model cars.
- **Orders**: stores sales orders placed by customers.
- **OrderDetails**: stores sales order line items for each sales order.
- **Payments**: stores payments made by customers based on their accounts.

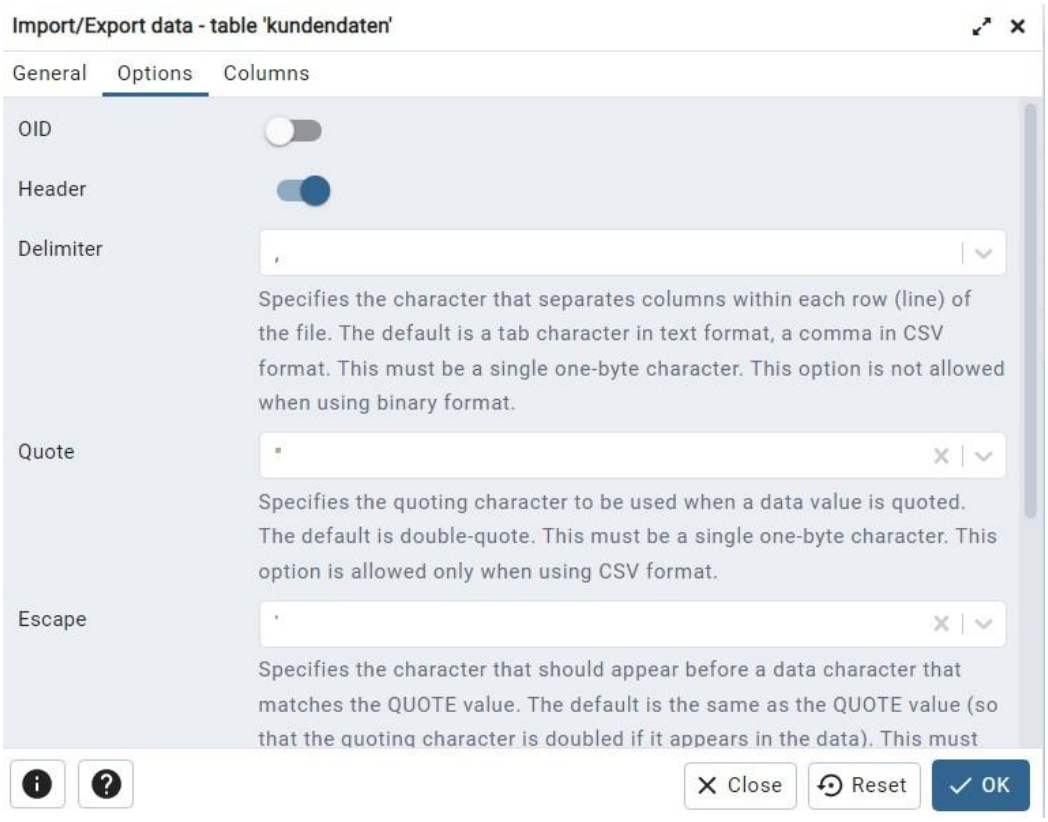
4. Erstellen Sie unter Ihrem neuen Verzeichnis *classicmodels* bitte alle im Datenmodell abgebildeten Tabellen mithilfe eines CREATE TABLE Statement. Die mit einem * markierten Spalten wie z.B. die customerNumber für die customers Tabelle sind jeweils die PRIMARY KEYS der Tabellen. Bitte vergeben Sie sinnvolle Datentypen. Grundlage für die Tabellen sind die mitgeschickten csv-Dateien im Moodle Kurs.

5. Wir möchten nun die Tabellen des Datenmodells mit Daten befüllen. Bitte schauen Sie sich im Moodle Kurs die zur Verfügung gestellten csv-Dateien an und laden diese in Ihren Benutzer Ordner runter. Wichtig ist, dass Sie den Pfad kennen, in dem Ihre Datei liegt.

6. Klicken Sie mit einem Rechtsklick auf die Tabelle wie z.B. customers und dort auf das Feld Import / Export Data… Sie erhalten nachfolgende Maske, in der Sie bitte den Zielpfad Ihrer csv-Datei unter Filename eintragen und als Encoding 'UTF8' wählen.



Im 2. Reiter 'Options' achten Sie bitte auf darauf, dass Sie die 'Header-Option' auswählen, der Rest sollte voreingestellt sein. (falls nicht, bitte wie auf dem folgenden **Screenshot** sichtbar **übernehmen**):



Im 3. Reiter *'Columns'* bitte alle Spalten vor der rownumber löschen, da die **rownumber** die **erste Spalte** der csv-Datei ist.

7. Wir möchten die Tabelle nun um eine weitere Zeile ergänzen, die noch nicht erfasst wurde. Bitte befüllen Sie die Tabelle *products mit folgendem Produkt durch eine INSERT-Statement*:

productcode: S18_1129
productName: 1993 Mazda RX-7
productLine: Classic Cars
productScale: 1:18
productVendor: Highway 66 Mini Classics
productDescription: "This model features, opening hood, opening doors, detailed engine, rear spoiler, opening trunk, working steering, tinted windows, baked enamel finish. Color red."
quantityInStock: 3975
buyPrice: 83.51
MSRP: 141.54

Lösungsvorschlag:

```
INSERT INTO products (
  productCode, productName, productLine, productScale, productVendor,
  productDescription, quantityInStock, buyPrice, MSRP
) VALUES (
  'S18_1129', '1993 Mazda RX-7', 'Classic Cars', '1:18', 'Highway 66 Mini Classics',
  'This model features, opening hood, opening doors, detailed engine, rear spoiler,
opening trunk, working steering, tinted windows, baked enamel finish. Color red.',
  3975, 83.51, 141.54
);
```

8. Geben Sie alle Daten der Tabelle *customers* aus.

Lösungsvorschlag:

```
SELECT * FROM customers;
```

9. Geben Sie alle Daten der Tabelle *customers* aus und ordnen Sie die Werte dabei nach aufsteigender customerNumber.

Lösungsvorschlag:

```
SELECT * FROM customers ORDER BY customerNumber ASC;
```

Hinweis: ASC ist optional, da Sortierung per default aufsteigend erfolgt

10. Geben Sie alle Daten der Tabelle *customers* aus und ordnen Sie die Werte dabei nach absteigendem customeName.

Lösungsvorschlag:

```
SELECT * FROM customers ORDER BY customerName DESC;
```

11. Geben Sie alle Spalten der Tabelle *customers* für die Kunden aus, die aus Deutschland kommen.

Lösungsvorschlag:

```
SELECT * FROM customers WHERE country = 'Germany';
```

12. Geben Sie alle Spalten der Tabelle Kundendaten für die Kunden aus, die ein Creditlimit von größer als 100.000 besitzen.

Lösungsvorschlag:

```
SELECT * FROM customers WHERE creditLimit > 100000;
```

13. Zeigen Sie alle eindeutigen Städte, aus denen Kunden stammen, alphabetisch sortiert.

Lösungsvorschlag:

```
SELECT DISTINCT city FROM customers ORDER BY city ASC;
```

Hinweis: ASC ist optional, da Sortierung per default aufsteigend ist.

14. Selektieren Sie alle Spalten für die Kunden, die entweder aus Frankreich oder Spanien stammen. Hinweis: Hier könnten Sie (müssen es aber nicht) den IN-Operator verwenden. Syntax: …. WHERE country IN ('France', 'Spain')

Lösungsvorschlag:

```
SELECT * FROM customers WHERE country IN ('France', 'Spain');
```

Hinweis: alternativ ginge auch z.B. das logische OR:

```
SELECT *
FROM customers
WHERE country = 'France' OR country = 'Spain';
```

15. Bestimmen Sie die customerNumber und customerName aller Kunden, deren Name mit 'T' beginnt, sortiert nach absteigender customerNumber.

Lösungsvorschlag:

```
SELECT customerNumber, customerName FROM customers
WHERE customerName LIKE 'T%' ORDER BY customerNumber DESC;
```

16. Prüfen Sie, ob die Spalte *Creditlimit* NULL-Werte enthält.

Lösungsvorschlag:

```
SELECT * FROM customers WHERE creditLimit IS NULL;
```

17. Selektieren Sie die *customerids* und Kundennamen jener Kunden, deren customerName mit der Zeichenkette 'Sig' beginnt, also z.B. customerName = Signal Gift Stores…

Lösungsvorschlag:

```
SELECT customerNumber, customerName FROM customers
WHERE customerName LIKE 'Sig%';
```

18. Geben Sie alle Daten aus der Tabelle *orders* aus und sortiere sie diese nach absteigendem orderDate.

Lösungsvorschlag:

```
SELECT * FROM orders ORDER BY orderDate DESC;
```

19. Zeigen Sie alle Bestellungen an, die den Status 'Shipped' haben.

Lösungsvorschlag:

```
SELECT * FROM orders WHERE status = 'Shipped';
```

20. Zeigen Sie alle Bestellungen an, deren requiredDate vor dem shippedDate liegt. Was bedeutet dies für die Bestellung?

Lösungsvorschlag:

```
SELECT * FROM orders WHERE requiredDate < shippedDate;
-- Bedeutung: Bestellung wurde später geliefert als ursprünglich geplant.
```

Hinweis: also wurde die Bestellung zu spät geliefert!

21. Zeigen Sie alle Bestellungen, die entweder den Status 'Cancelled' oder 'On Hold' haben.

Lösungsvorschlag:

```
SELECT * FROM orders WHERE status IN ('Cancelled', 'On Hold');
```

22. Welche eindeutigen Status gibt es in der Tabelle orders?

Lösungsvorschlag:

```
SELECT DISTINCT status FROM orders;
```

Weiterführende Aufgaben, deren Theorie wir noch nicht besprochen haben

23. Selektieren Sie das durchschnittliche creditlimit aller Kunden aus der Tabelle *customers*.

Lösungsvorschlag:

```
SELECT AVG(creditLimit) FROM customers;
```

24. Selektieren Sie das größte Creditlimit aus der Tabelle *customers*.

Lösungsvorschlag:

```
SELECT MAX(creditLimit) FROM customers;
```

25. Selektieren Sie das kleinste Creditlimit aus der Tabelle *customers*, das nicht 0 ist. (d.h. 0 explizit ausgenommen).

Lösungsvorschlag:

```
SELECT MIN(creditLimit) FROM customers WHERE creditLimit > 0;
```

26. Selektieren Sie die Anzahl aller Kunden aus der Tabelle *customers*.

Lösungsvorschlag:

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

27. Selektieren Sie die Anzahl aller Kunden aus der Tabelle Kundendaten, die einen creditscore von über 700 haben.

Lösungsvorschlag:

```
SELECT COUNT(*) FROM customers WHERE creditscore > 700;
```

Weitere weiterführende Aufgaben mit Kontext

a) Wir möchten wissen, wie viele Kunden (Anzahl) aus welchen Ländern kommen. Außerdem möchten wir das *durchschnittliche Kreditlimit* pro Land sehen. Das Ergebnis soll nach absteigender *Anzahl*, anschließend nach absteigendem *durchschnittlichen Creditlimit* sortiert werden.

Lösungsvorschlag:

```
SELECT country, COUNT(*) AS kundenanzahl, AVG(creditLimit) AS durchschnitt_creditlimit
FROM customers
GROUP BY country
ORDER BY kundenanzahl DESC, durchschnitt_creditlimit DESC;
```

b) Wir benötigen die Kundennummer, die Ordernumber, das Bestelldatum und den Status jener Kunden, deren Bestellung den Status 'Cancelled' haben. Diese sollen nach absteigendem Bestelldatum sortiert werden.

Lösungsvorschlag:

```
SELECT customerNumber, orderNumber, orderDate, status
FROM orders
WHERE status = 'Cancelled'
ORDER BY orderDate DESC;
```

c) Wir möchten eine spezielle Marketingaktion für alle Kunden aus USA und Deutschland organisieren. Allerdings kennen wir die Adressen der Kunden nicht. Bitte helfen Sie, indem Sie den Namen, das Land, die Stadt und die Adresse aller Kunden ausgeben, die aus diesen beiden Ländern kommen. Die Ausgabe soll dabei soll dabei so erfolgen, dass die Kunden aus Deutschland zuerst angezeigt werden.

Lösungsvorschlag:

```
SELECT customerName, country, city, addressLine1
FROM customers
WHERE country IN ('Germany', 'USA')
ORDER BY country ASC;
```