

# Unicorn Mini-Projekt 2026 –

## Aufgabe 1: SQL Data Exploration

---

Dieses Dokument enthält die vollständige Bearbeitung von Aufgabe 1 (SQL Data Exploration). Für jede Fragestellung werden die Aufgabenstellung, der Rechenweg (SQL-Query), das Ergebnis sowie eine kurze Erkenntnis dokumentiert. Zusätzlich ist Platz für Screenshots aus Beekeeper vorgesehen.

---

### Q1 · Wie viele Kunden gibt es insgesamt?

#### SQL-Rechenweg:

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

#### Ergebnis:

795

#### Kurze Erkenntnis:

Überblick über die Größe der Kundenbasis.

#### Screenshot (Beekeeper):

The screenshot shows a Beekeeper SQL editor window. The code area contains a multi-line SQL query. The results area shows a single row with the value '795' in the 'anzahl\_kunden' column. The 'Auto Commit' button is visible at the bottom left, and the column header 'anzahl\_kunden' is at the top right of the results table.

```
/* -----  
* Q1 - Wie viele Kunden gibt es insgesamt?  
* ----- */  
SELECT  
    COUNT(*) AS anzahl_kunden  
FROM customers;  
  
/* -----  
* Q2 - Welche Stadt hatte 2015 den höchsten Profit?  
* ----- */  
SELECT  
    o.shipping_city,  
    SUM(od.order_profits) AS gesamtprofit  
FROM orders o  
JOIN order_details od  
ON o.order_id = od.order_id  
WHERE o.order_date >= DATE '2015-01-01'  
Auto Commit Manual  
anzahl_kunden ▾  
1 | 795 |
```

---

## Q2 – Welche Stadt hatte 2015 den höchsten Profit?

### SQL-Rechenweg:

```
SELECT o.shipping_city,
SUM(od.order_profits) AS gesamtprofit
FROM orders o
JOIN order_details od
ON o.order_id = od.order_id
WHERE o.order_date >= DATE '2015-01-01'
AND o.order_date < DATE '2016-01-01'
GROUP BY o.shipping_city
ORDER BY gesamtprofit DESC
LIMIT 1;
```

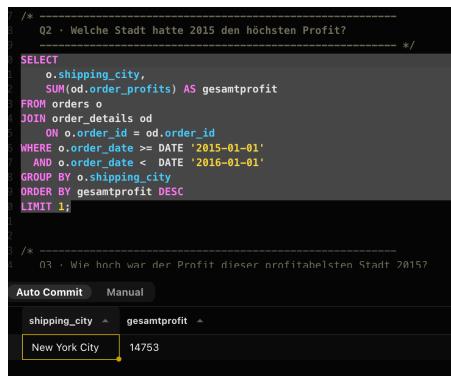
### Ergebnis:

New York City

### Kurze Erkenntnis:

Der Unternehmensprofit ist stark auf einzelne Metropolen konzentriert.

### Screenshot (Beekeeper):



The screenshot shows a Beekeeper application window. At the top, there is a code editor containing the SQL query for Q2. Below the code editor is a results table with one row. The results table has two columns: 'shipping\_city' and 'gesamtprofit'. The 'shipping\_city' column contains 'New York City' and the 'gesamtprofit' column contains '14753'. A yellow box highlights the '14753' value.

```
/*
Q2 - Welche Stadt hatte 2015 den höchsten Profit?
*/
SELECT
    o.shipping_city,
    SUM(od.order_profits) AS gesamtprofit
FROM orders o
JOIN order_details od
    ON o.order_id = od.order_id
WHERE o.order_date >= DATE '2015-01-01'
    AND o.order_date < DATE '2016-01-01'
GROUP BY o.shipping_city
ORDER BY gesamtprofit DESC
LIMIT 1;

/*
Q3 - Wie hoch war der Profit dieser profitabelsten Stadt 2015?
*/
Auto Commit Manual
shipping_city  gesamtprofit
New York City 14753
```

---

## Q3 – Wie hoch war der Profit dieser Stadt im Jahr 2015?

### SQL-Rechenweg:

```
SELECT SUM(od.order_profits)
FROM orders o
JOIN order_details od
ON o.order_id = od.order_id
WHERE o.shipping_city = 'New York City'
AND o.order_date >= DATE '2015-01-01'
AND o.order_date < DATE '2016-01-01';
```

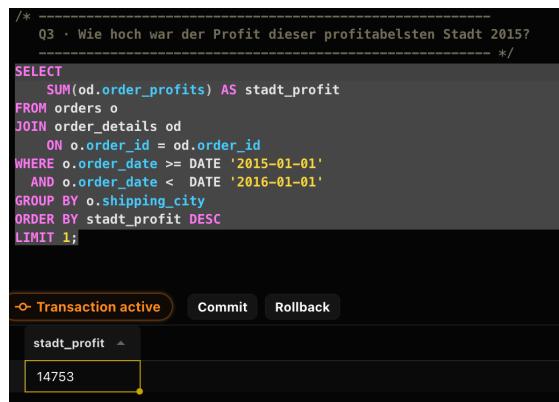
### Ergebnis:

14.753

### Kurze Erkenntnis:

New York City ist ein zentraler Profit-Treiber.

### Screenshot (Beekeeper):



The screenshot shows a SQL query in the Beekeeper application. The query is:

```
/*
  Q3 - Wie hoch war der Profit dieser profitabelsten Stadt 2015?
  -----
*/
SELECT
    SUM(od.order_profits) AS stadt_profit
FROM orders o
JOIN order_details od
    ON o.order_id = od.order_id
WHERE o.order_date >= DATE '2015-01-01'
    AND o.order_date < DATE '2016-01-01'
GROUP BY o.shipping_city
ORDER BY stadt_profit DESC
LIMIT 1;
```

Below the query, there is a table with one row showing the result:

| stadt_profit |
|--------------|
| 14753        |

At the bottom of the interface, there are buttons for Transaction active, Commit, and Rollback.

---

## Q4 – Wie viele unterschiedliche Städte gibt es?

### SQL-Rechenweg:

```
SELECT COUNT(DISTINCT shipping_city)
FROM orders;
```

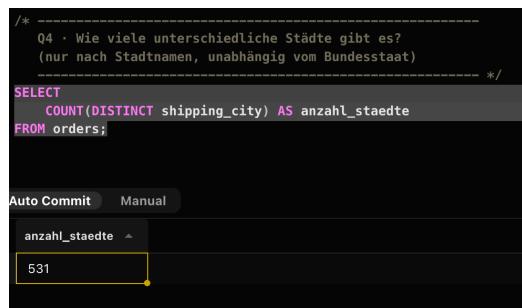
### Ergebnis:

531

### Kurze Erkenntnis:

Das Unternehmen ist geografisch breit aufgestellt.

### Screenshot (Beekeeper):



The screenshot shows a SQL query in the editor and its execution results. The query is:

```
/*
  Q4 - Wie viele unterschiedliche Städte gibt es?
  (nur nach Stadtnamen, unabhängig vom Bundesstaat)
*/
SELECT
  COUNT(DISTINCT shipping_city) AS anzahl_staedte
FROM orders;
```

The results table has one row with the value 531, which is highlighted with a yellow border.

| anzahl_staedte |
|----------------|
| 531            |

---

## Q5 – Welche Stadt in Tennessee ist am profitabelsten?

### SQL-Rechenweg:

```
SELECT o.shipping_city,
SUM(od.order_profits) AS gesamtprofit
FROM orders o
JOIN order_details od
ON o.order_id = od.order_id
WHERE o.shipping_state = 'Tennessee'
GROUP BY o.shipping_city
ORDER BY gesamtprofit DESC
LIMIT 1;
```

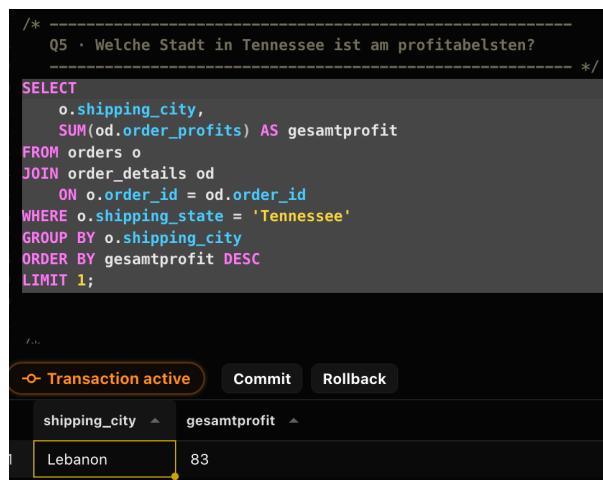
### Ergebnis:

[Lebanon \(83\)](#)

### Kurze Erkenntnis:

Tennessee trägt nur marginal zum Gesamtprofit bei.

### Screenshot (Beekeeper):



The screenshot shows the Beekeeper SQL interface. At the top, there is a code editor containing the SQL query for question 5. Below the code editor is a toolbar with buttons for 'Transaction active' (highlighted in orange), 'Commit', and 'Rollback'. The main area displays the results of the query in a table. The table has two columns: 'shipping\_city' and 'gesamtprofit'. There is one row of data: 'Lebanon' in the 'shipping\_city' column and '83' in the 'gesamtprofit' column. The entire row is highlighted with a yellow border.

```
/*
  Q5 - Welche Stadt in Tennessee ist am profitabelsten?
*/
SELECT
    o.shipping_city,
    SUM(od.order_profits) AS gesamtprofit
FROM orders o
JOIN order_details od
    ON o.order_id = od.order_id
WHERE o.shipping_state = 'Tennessee'
GROUP BY o.shipping_city
ORDER BY gesamtprofit DESC
LIMIT 1;
```

| shipping_city | gesamtprofit |
|---------------|--------------|
| Lebanon       | 83           |

---

## Q6 – Verteilung der Kundensegmente

### SQL-Rechenweg:

```
SELECT customer_segment,  
COUNT(*) AS anzahl_kunden  
FROM customers  
GROUP BY customer_segment;
```

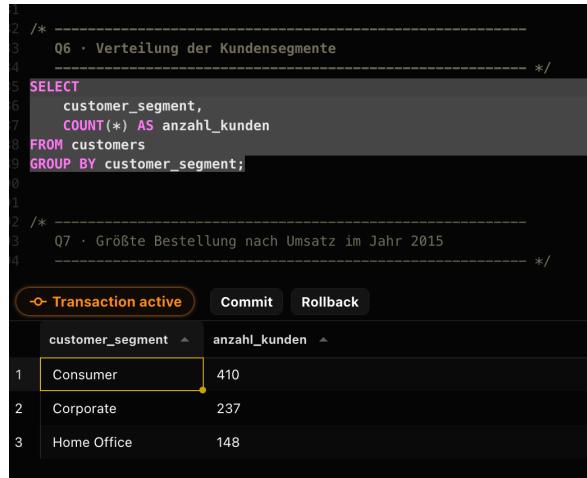
### Ergebnis:

Consumer 410 | Corporate 237 | Home Office 148

### Kurze Erkenntnis:

Consumer ist das größte Kundensegment.

### Screenshot (Beekeeper):



The screenshot shows a terminal window with an SQL query and its execution results. The query is:

```
/* -----  
 Q6 · Verteilung der Kundensegmente  
 ----- */  
SELECT  
    customer_segment,  
    COUNT(*) AS anzahl_kunden  
FROM customers  
GROUP BY customer_segment;  
  
/* -----  
 Q7 · Größte Bestellung nach Umsatz im Jahr 2015  
 ----- */
```

The results table has two columns: "customer\_segment" and "anzahl\_kunden". The data is:

| customer_segment | anzahl_kunden |
|------------------|---------------|
| Consumer         | 410           |
| Corporate        | 237           |
| Home Office      | 148           |

---

## Q7 – Größte Bestellung nach Umsatz 2015

### SQL-Rechenweg:

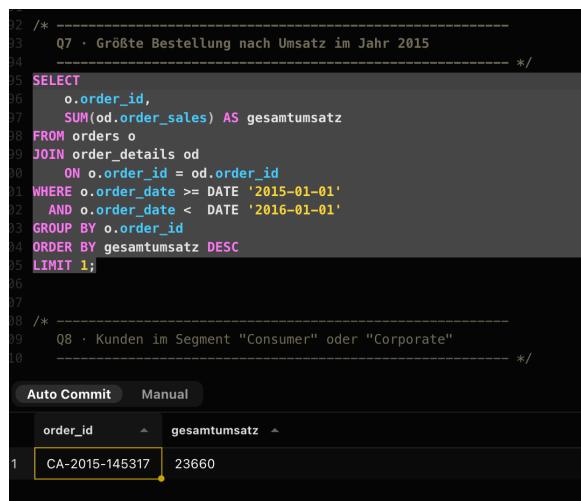
```
SELECT o.order_id,
       SUM(od.order_sales) AS gesamtumsatz
    FROM orders o JOIN order_details od
      ON o.order_id = od.order_id
   WHERE o.order_date >= DATE '2015-01-01'
     AND o.order_date < DATE '2016-01-01'
  GROUP BY o.order_id
 ORDER BY gesamtumsatz DESC
LIMIT 1;
```

### Ergebnis:

**CA-2015-145317 – 23.660**

### Kurze Erkenntnis:

### Screenshot (Beekeeper):



The screenshot shows a MySQL command-line interface window. The SQL query is displayed in the top half, and the results are shown in the bottom half. The results table has two columns: 'order\_id' and 'gesamtumsatz'. The single row contains the value 'CA-2015-145317' in the 'order\_id' column and '23660' in the 'gesamtumsatz' column.

| order_id       | gesamtumsatz |
|----------------|--------------|
| CA-2015-145317 | 23660        |

---

## Q8 – Kunden im Segment Consumer oder Corporate

**SQL-Rechenweg:**

```
SELECT COUNT(*)  
FROM customers  
WHERE customer_segment IN ('Consumer', 'Corporate');
```

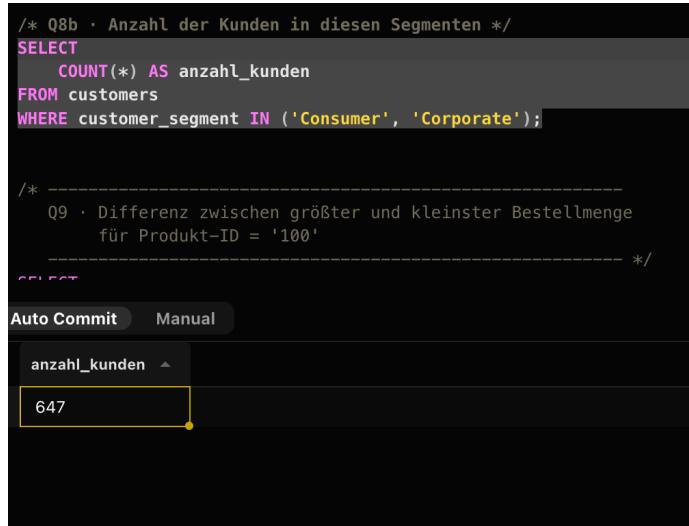
**Ergebnis:**

**647**

**Kurze Erkenntnis:**

**Consumer und Corporate dominieren die Kundenbasis.**

**Screenshot (Beekeeper):**



The screenshot shows a MySQL command-line interface. The query is displayed in the top pane:

```
/* Q8b · Anzahl der Kunden in diesen Segmenten */  
SELECT  
    COUNT(*) AS anzahl_kunden  
FROM customers  
WHERE customer_segment IN ('Consumer', 'Corporate');
```

The bottom pane shows the result of the query:

```
/* -----  
   Q9 · Differenz zwischen größter und kleinster Bestellmenge  
   für Produkt-ID = '100'  
----- */  
SELECT  
anzahl_kunden  
647
```

The result value '647' is highlighted with a yellow box.

---

## Q9 – Differenz größte/kleinste Bestellmenge für Produkt 100

**SQL-Rechenweg:**

```
SELECT MAX(quantity) - MIN(quantity)
FROM order_details
WHERE product_id = '100';
```

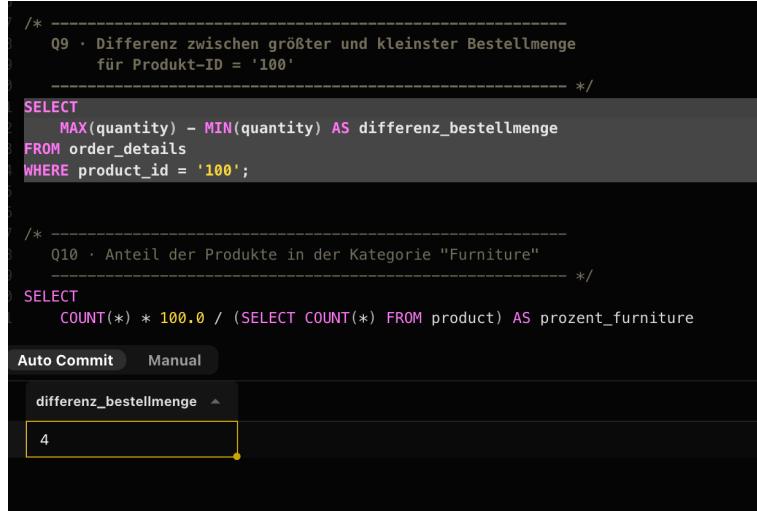
**Ergebnis:**

4

**Kurze Erkenntnis:**

Die Bestellmengen schwanken nur leicht.

**Screenshot (Beekeeper):**



The screenshot shows a SQL editor interface with two queries. The first query is the one we've been working on:

```
/*
    Q9 · Differenz zwischen größter und kleinster Bestellmenge
    für Produkt-ID = '100'
*/
SELECT
    MAX(quantity) - MIN(quantity) AS differenz_bestellmenge
FROM order_details
WHERE product_id = '100';
```

The second query is for Q10, calculating the percentage of products in the "Furniture" category:

```
/*
    Q10 · Anteil der Produkte in der Kategorie "Furniture"
*/
SELECT
    COUNT(*) * 100.0 / (SELECT COUNT(*) FROM product) AS prozent_furniture
```

Below the queries, there are two buttons: "Auto Commit" and "Manual". Underneath the buttons, the result of the first query is displayed in a table:

| differenz_bestellmenge |
|------------------------|
| 4                      |

---

## Q10 – Anteil der Produkte in der Kategorie Furniture

### SQL-Rechenweg:

```
SELECT COUNT(*) * 100.0 / (SELECT COUNT(*) FROM product) AS prozent_furniture
FROM product
WHERE product_category = 'Furniture';
```

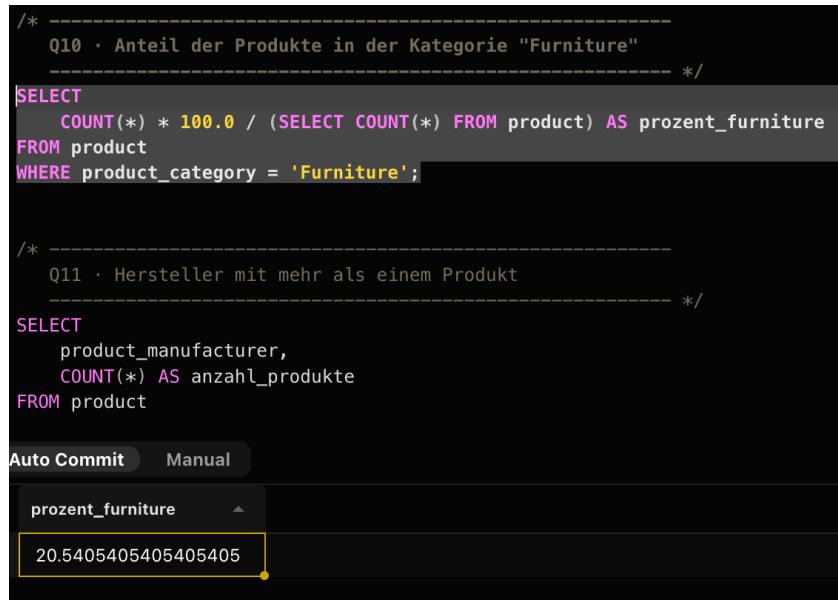
### Ergebnis:

≈ 20,54 %

### Kurze Erkenntnis:

Furniture ist eine wichtige, aber nicht dominante Kategorie.

### Screenshot (Beekeeper):



The screenshot shows a SQL editor interface with a dark theme. The code area contains two queries. The first query is for Q10, calculating the percentage of products in the 'Furniture' category. The second query is for Q11, listing manufacturers with more than one product. Below the code, there are two buttons: 'Auto Commit' and 'Manual'. A dropdown menu is open, showing the column name 'prozent\_furniture' and its value '20.5405405405405'.

```
/*
  Q10 · Anteil der Produkte in der Kategorie "Furniture"
  -----
*/
SELECT
    COUNT(*) * 100.0 / (SELECT COUNT(*) FROM product) AS prozent_furniture
FROM product
WHERE product_category = 'Furniture';

/*
  Q11 · Hersteller mit mehr als einem Produkt
  -----
*/
SELECT
    product_manufacturer,
    COUNT(*) AS anzahl_produkte
FROM product
```

Auto Commit Manual

prozent\_furniture

20.5405405405405

---

## Q11 – Hersteller mit mehr als einem Produkt

### SQL-Rechenweg:

```
SELECT product_manufacturer,  
COUNT(*) AS anzahl_produkte  
FROM product  
GROUP BY product_manufacturer  
HAVING COUNT(*) > 1;
```

### Ergebnis:

[z. B. Other \(356\), Xerox \(173\), Avery \(98\)](#)

### Kurze Erkenntnis:

[Einige Hersteller dominieren klar das Sortiment.](#)

### Screenshot (Beekeeper):

```
/* -----  
   Q11 · Hersteller mit mehr als einem Produkt  
----- */  
  
SELECT  
    product_manufacturer,  
    COUNT(*) AS anzahl_produkte  
FROM product  
GROUP BY product_manufacturer  
HAVING COUNT(*) > 1;  
  
/* -----  
   Q12 · Anzahl der Produkte pro Subkategorie  
----- */  
  
Auto Commit Manual  


| product_manufacturer | anzahl_produkte |
|----------------------|-----------------|
| Linden               | 2               |
| Iceberg              | 3               |
| SanDisk              | 8               |
| Memorex              | 13              |
| Bulldog              | 2               |
| Nortel               | 4               |


```

---

## Q12 – Anzahl der Produkte pro Subkategorie

### SQL-Rechenweg:

```
SELECT product_subcategory,  
COUNT() AS anzahl_produkte  
FROM product  
GROUP BY product_subcategory;  
ORDER BY count(*) DESC;
```

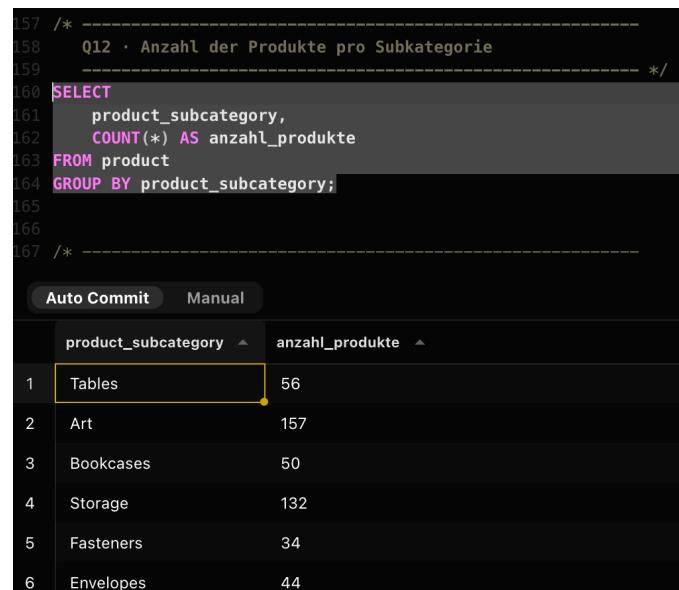
### Ergebnis:

Paper (277) und Binders (211) sind die größten Subkategorien → hohe Sortimentsbreite dort. Copiers (13) ist sehr klein → potenziell Nischenkategorie oder bewusst schmal gehaltenes Sortiment.

### Kurze Erkenntnis:

Paper (277) und Binders (211) sind die größten Subkategorien → hohe Sortimentsbreite dort. Copiers (13) ist sehr klein → potenziell Nischenkategorie oder bewusst schmal gehaltenes Sortiment.

### Screenshot (Beekeeper):



The screenshot shows the Beekeeper SQL IDE interface. At the top, there is a code editor window displaying the following SQL query:

```
157 /* -----  
158   Q12 · Anzahl der Produkte pro Subkategorie  
159 ----- */  
160 SELECT  
161   product_subcategory,  
162   COUNT(*) AS anzahl_produkte  
163 FROM product  
164 GROUP BY product_subcategory;  
165  
166  
167 /* -----
```

Below the code editor is a table result set. The table has two columns: "product\_subcategory" and "anzahl\_produkte". The data is as follows:

|   | product_subcategory | anzahl_produkte |
|---|---------------------|-----------------|
| 1 | Tables              | 56              |
| 2 | Art                 | 157             |
| 3 | Bookcases           | 50              |
| 4 | Storage             | 132             |
| 5 | Fasteners           | 34              |
| 6 | Envelopes           | 44              |

At the bottom of the interface, there are buttons for "Auto Commit" and "Manual".

---

## Q13 – Produkte mit Gesamtbestellmenge $\geq 100$

**SQL-Rechenweg:**

```
SELECT
p.product_name,
product_id,
SUM(quantity) AS gesamtbestellmenge
FROM order_details od
JOIN product p
ON p.product_id = od.product_id
GROUP BY product_id,
p.product_name
HAVING SUM(quantity) >= 100;
```

**Ergebnis:**

```
product_id 538 → gesamtmenge 150
product_id 812 → gesamtmenge 109
product_id 1216 → gesamtmenge 132
product_id 1507 → gesamtmenge 539
product_id 1501 → gesamtmenge 170
product_id 920 → gesamtmenge 155
product_id 1600 → gesamtmenge 221
product_id 122 → gesamtmenge 295
```

**Kurze Erkenntnis:**

Es gibt 8 Produkte mit sehr hoher Gesamtmenge ( $\geq 100$ ).  
Auffälligster Ausreißer: product\_id 1507 mit 539 Einheiten.

**Screenshot (Beekeeper):**

```
-- 013 - Produkt(e) mit Gesamtbestellmenge >= 100
SELECT
    product_id,
    SUM(quantity) AS gesamtmenge
FROM order_details
GROUP BY product_id
HAVING SUM(quantity) >= 100;
```

| product_id | gesamtmenge |
|------------|-------------|
| 538        | 150         |
| 812        | 109         |
| 1216       | 132         |
| 1507       | 539         |
| 1501       | 170         |
| 920        | 155         |
| 1600       | 221         |
| 122        | 295         |

---

## Q14 – Bonusaufgabe

**SQL-Rechenweg:**

```
SELECT
    o.order_id,
    o.order_date,
    o.shipping_date,
    o.shipping_city,
    o.shipping_state,
    o.shipping_region,
    o.shipping_country,
    o.shipping_postal_code,
    o.shipping_mode,
    c.customer_id,
    c.customer_name,
    c.customer_segment,
    od.order_details_id,
    od.product_id,
    od.quantity,
    od.order_discount,
    od.order_sales,
    od.order_profits,
    od.order_profit_ratio,
    p.product_name,
    p.product_category,
    p.product_subcategory,
    p.product_manufacturer
```

```
FROM
orders o
JOIN
order_details od
ON
o.order_id = od.order_id
JOIN customers c
ON
o.customer_id = c.customer_id
JOIN product p
ON od.product_id = p.product_id;
```

#### **Ergebnis:**

Die SQL-Abfrage erzeugt einen denormalisierten Datensatz, bei dem jede Zeile einem bestellten Produkt entspricht. Der Datensatz enthält sämtliche relevanten Informationen zu Bestellung, Kunde, Produkt, Umsatz und Versand.

Das Ergebnis wurde erfolgreich als CSV-Datei exportiert und dient als Grundlage für die Analysen im Spreadsheet- und Tableau-Teil des Projekts.

#### **Kurze Erkenntnis:**

Durch die Zusammenführung aller Tabellen entsteht eine einheitliche Datenbasis, die den Wechsel von SQL zu Spreadsheets und Tableau ermöglicht, ohne erneut Joins oder Datenaufbereitung durchführen zu müssen.

#### **Screenshot (Beekeeper):**

```

/*
Bonus – Alle Tabellen zu einem Datensatz zusammenführen (CSV-Export)
*/

```

**SELECT**

```

/* Orders */
o.order_id,
o.order_date,
o.shipping_date,
o.shipping_city,
o.shipping_state,
o.shipping_region,
o.shipping_country,
o.shipping_postal_code,
o.shipping_mode,
```

```

/* Customers */
.
```

Auto Commit Manual

| order_id       | order_date          | shipping_date       | shipping_city | shipping_state | shipping |
|----------------|---------------------|---------------------|---------------|----------------|----------|
| CA-2015-100004 | 2015-09-06 00:00:00 | 2015-09-06 00:00:00 | New York City | New York       | East     |
| CA-2015-100004 | 2015-09-06 00:00:00 | 2015-09-06 00:00:00 | New York City | New York       | East     |
| CA-2015-100006 | 2015-09-07 00:00:00 | 2015-09-13 00:00:00 | New York City | New York       | East     |
| CA-2015-100032 | 2015-09-07 00:00:00 | 2015-09-13 00:00:00 | New York City | New York       | East     |
| CA-2015-100032 | 2015-09-07 00:00:00 | 2015-09-13 00:00:00 | New York City | New York       | East     |
| CA-2015-100032 | 2015-09-07 00:00:00 | 2015-09-13 00:00:00 | New York City | New York       | East     |
| CA-2015-100051 | 2015-09-06 00:00:00 | 2015-09-12 00:00:00 | New York City | New York       | East     |
| CA-2015-100090 | 2015-07-08 00:00:00 | 2015-07-12 00:00:00 | San Francisco | California     | West     |
| CA-2015-100090 | 2015-07-08 00:00:00 | 2015-07-12 00:00:00 | San Francisco | California     | West     |
| CA-2015-100136 | 2015-09-05 00:00:00 | 2015-09-11 00:00:00 | New York City | New York       | East     |
| CA-2015-100136 | 2015-09-05 00:00:00 | 2015-09-11 00:00:00 | New York City | New York       | East     |
| CA-2015-100136 | 2015-09-05 00:00:00 | 2015-09-11 00:00:00 | New York City | New York       | East     |