

1 select-Anfragen


Beantworten Sie mittels Datenbankabfragen folgende Fragen:

1. Welche Serien haben ein "a" im Namen ?
2. Geben Sie die Namen der Sender aus, welche aktuell ein Fernsehprogramm haben.
3. Geben Sie die Namen der Sender aus, welche aktuell ein Fernsehprogramm mit Ausstrahlungsbeginn nach 17:30 haben.
4. Geben Sie die Namen der Serien aus, bei denen Aaron Paul mitspielt.
5. Geben Sie die Gesamtsumme aus, welche die Pilsstube Herkules monatlich für Streamingabos ausgibt.
6. Geben Sie die nach durchschnittlichem Rating absteigend sortierten Episodenstaffeln aus.
7. Geben Sie aus, welche Serien aktuell nicht im TV ausgestrahlt werden.
8. Bei welchen Serien ist der selbe Künstler Schauspieler und Stoffentwickler?
9. Geben Sie alle Schauspieler aus, welche in mehreren Serien mitspielen.
10. Geben Sie für jede Episode den Unterschied im Rating zur Vorgängerepisode (nur innerhalb der jeweiligen Staffel) an.

1.)


```
SELECT serienname  
FROM Serie  
WHERE serienname ILIKE '%a%';
```

ILIKE Berücksichtigt Case nicht

	serienname [PK] character varying (254) 
1	Breaking Bad
2	The Path

2.)

```
SELECT DISTINCT name  
FROM Ausstrahlung;
```

	name character varying (254) 
1	SAT1
2	PRO7


DISTINCT extrahiert eindeutige Datensätze und verhindert doppelte Ergebnisse

name: Dies ist die Spalte, aus der die eindeutigen Werte ausgewählt werden sollen.

(Bei SELECT DISTINCT * werden doppelte Datensätze entfernt)

3.)


```
SELECT DISTINCT name
FROM Ausstrahlung
WHERE startzeit::time > '17:30:00';
```

	name character varying (254) 
1	PRO7

startzeit wird zu "time" gecasted

4.)

```
SELECT DISTINCT s.seriename
FROM Serie s
JOIN hat_Hauptschauspieler h ON s.seriename = h.seriename
JOIN Künstler k ON h.id = k.id
WHERE k.vorname = 'Aaron' AND k.nachname = 'Paul';
```

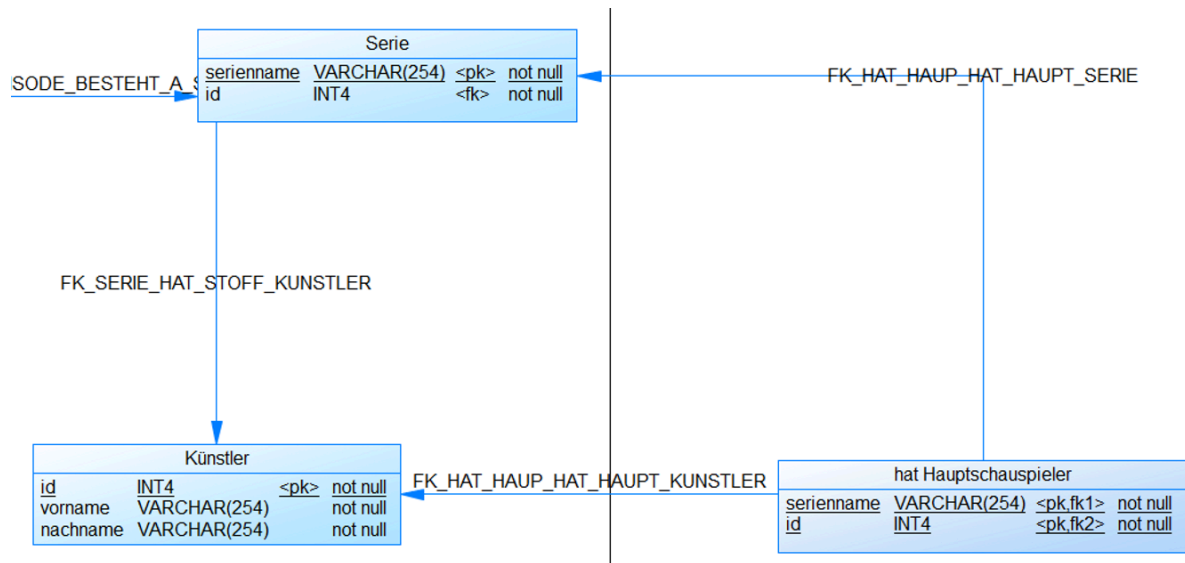
	seriename [PK] character varying (254) 
1	Breaking Bad
2	The Path

Alle Seriennamen, in denen der Schauspieler Aaron Paul als Hauptschauspieler mitspielt.

1. Jede Zeile wo Serie.seriename und hat_Hauptschauspieler.seriename gleich ist wird verknüpft
2. Jede Zeile wo hat_Hauptschauspieler.id und Künstler.id gleich ist wird verknüpft
3. Filterung nach Aaron Paul

(Ohne Filterung bekämen wir alle Serien, in denen ein Hauptschauspieler mitspielt =>


Verknüpfung der Serien mit Hauptschauspielern und Künstlern um Verknüpfungstabelle abzubilden)



5.) (Geänderter User)

```

SELECT SUM(monatspreis) AS Gesamtsumme
FROM Vertrag
WHERE email = 'peter.feldmann@awo.de';
  
```

	gesamtsumme	
	numeric	
1		75

Summiere die monatspreis-Werte aller Verträge in der Tabelle Vertrag, die der E-Mail-Adresse 'peter.feldmann@awo.de' zugeordnet sind.

6.)

```

SELECT serienname, staffel, AVG(imdbrating) AS avg_rating
FROM Episode
GROUP BY serienname, staffel
ORDER BY avg_rating DESC;
  
```

	seriename character varying (254)	staffel integer	avg_rating numeric
1	Breaking Bad	5	9.381250000000000
2	Breaking Bad	4	8.9615384615384615
3	Breaking Bad	2	8.7923076923076923
4	Breaking Bad	3	8.7307692307692308
5	Breaking Bad	1	8.700000000000000
6	The Orville	3	8.140000000000000
7	The Orville	2	8.0357142857142857
8	The Path	1	7.910000000000000
9	The Orville	1	7.8583333333333333
10	The Path	3	7.7615384615384615
11	Lupin	1	7.660000000000000
12	The Path	2	7.5384615384615385
13	Lupin	2	7.500000000000000

GROUP BY = Alle Zeilen wo Serienname und Staffel gleich sind werden in einer Gruppe zusammengefasst. Innerhalb dieser Gruppe wird der Durchschnitt berechnet

OHNE GROUP BY hätte AVG den Durchschnitt aller Einträge berechnet und den durchschnittlichen IMDB Rating aller Episoden als eine Zeile ausgegeben.

7.)

Welche Serien werden aktuell nicht ausgestrahlt?

```
SELECT seriename
FROM Serie
WHERE seriename NOT IN (
    SELECT DISTINCT seriename
    FROM Ausstrahlung
);
```

	seriename [PK] character varying (254)
1	Breaking Bad
2	The Path

Es werden die Serien zurückgegeben, die aktuell nicht im Fernsehen ausgestrahlt werden, da wir nach Einträgen suchen, die nicht in Ausstrahlung, aber in Serie sind.

Alternative Lösung:

```
SELECT s.seriename
FROM Serie s
LEFT JOIN Ausstrahlung a ON s.seriename = a.seriename
WHERE a.seriename IS NULL;
```

1. Ausstrahlung wird mit Serie verknüpft wobei Serie den primären Verknüpfungspartner (LEFT TABLE) darstellt.

2. Mit ON wird die Verknüpfungsbedingung dargestellt. Dies ist beim LEFT JOIN anders als beim NATURAL JOIN benötigt.
3. Zuletzt werden die Ergebnisse gefiltert und nur Serien ausgegeben, die bei serienname NULL, also keine Übereinstimmung mit Ausstrahlung haben

8.)

Alle Serien, wo der Hauptschauspieler auch gleichzeitig der Stoffentwickler ist.

```
SELECT DISTINCT s.serienname
FROM Serie s
JOIN hat_Hauptschauspieler h ON s.serienname = h.serienname
JOIN Künstler k ON s.id = k.id
WHERE h.id = k.id;
```

	serienname [PK] character varying (254)
1	The Orville

- Wähle Serien aus Serie S
- Verknüpfe beide Tabellen (Serie + hat_hauptschauspieler) wo der serienname übereinstimmt = Hauptschauspieler der Serien
- Verknüpfe Künstler mit Serie -> Erhalte ID + Serienzugehörigkeit des Stoffentwicklers.
- Überprüfe wo der Künstler Hauptschauspieler und Stoffentwickler ist = ID in Tabellen

9.) Schauspieler die in mehr als einer Serie vorkommen

```
SELECT k.id, k.vorname, k.nachname
FROM Künstler k
JOIN hat_Hauptschauspieler h ON k.id = h.id
GROUP BY k.id, k.vorname, k.nachname
HAVING COUNT(DISTINCT h.serienname) > 1;
```

	id [PK] integer	vorname character varying (254)	nachname character varying (254)
1	666739	Aaron	Paul

- Verknüpfe Tabelle hat_Hauptschauspieler und Künstler -> Wähle Künstler aus die in mind. einer Serie hauptschauspieler sind
- GROUP BY sorgt dafür, dass alle Zeilen, die denselben Künstler repräsentieren, zu einer einzigen Zeile zusammengefasst werden. Damit wird COUNT auf jedes Gruppenmitglied angewendet
- COUNT(DISTINCT h.serienname) zählt die eindeutigen Serientitel für jedes Tupel

- HAVING filtert die Gruppenergebnisse basierend auf der Bedingung, in wie vielen Serien der Schauspieler mitwirkt.

id	vorname	nachname	h.seriename
1082477	Omar	Sy	Lupin
425982	Penny	Jerald	The Orville
666739	Aaron	Paul	Breaking Bad
	Aaron	Paul	The Path

10.)

Eine Liste der Episoden mit ihren Rating-Differenzen zu vorangegangenen Episoden.

```
WITH EpisodenRatings AS (
    SELECT serienname, nummer, staffel, titel, imdbrating,
           LAG(imdbrating) OVER (PARTITION BY serienname, staffel ORDER
    BY nummer) AS prev_rating
    FROM Episode
)
SELECT serienname, nummer, staffel, titel,
       imdbrating - prev_rating AS rating_diff
FROM EpisodenRatings
WHERE prev_rating IS NOT NULL;
```

	serienname [PK] character varying (254)	nummer [PK] integer	staffel [PK] integer	titel character varying (254)	rating_diff numeric
1	Breaking Bad	2	1	Die Katze ist im Sack	-0.4
2	Breaking Bad	3	1	...And the Bags in the River	0.1
3	Breaking Bad	4	1	Cancer Man	-0.5
4	Breaking Bad	5	1	Gray Matter	0.1
5	Breaking Bad	6	1	Crazy Handful of Nothin	1.0
6	Breaking Bad	7	1	A No-Rough-Stuff-Type Deal	-0.5
7	Breaking Bad	2	2	Grilled	0.7
8	Breaking Bad	3	2	Bit by a Dead Bee	-1.0
9	Breaking Bad	4	2	Down	-0.1
10	Breaking Bad	5	2	Breakage	0.1
11	Breaking Bad	6	2	Peekaboo	0.5
12	Breaking Bad	7	2	Negro y Azul	-0.2
13	Breaking Bad	8	2	Better Call Saul	0.6
14	Breaking Bad	9	2	4 Days Out	-0.1
15	Breaking Bad	10	2	Over	-0.6

1. Definition einer CTE (Common Table Expression) mit serienname, nummer, staffel, titel und imdbrating. Zusätzlich wird eine Spalte prev_rating definiert, die das rating einer vorangegangenen Episode enthält.
 - a. LAG gibt das imdbrating einer vorangegangenen Zeile zurück,
 - b. PARTITION BY sorgt für die Anwendung von LAG auf Zeilen der selben Serie und Staffel. LAG wird also dann erneut ausgeführt, wenn wir auf eine neue Serie stoßen. Somit stellen wir sicher, dass LAG keine vorherigen ratings einer anderen Serie in die Berechnung miteinbezieht.
2. Aus der CTE werden die Spalten serienname, nummer, staffel, und titel ausgewählt. Zusätzlich wird eine Spalte rating_diff definiert, welche die Differenz aus CTE.prev_rating und imdbrating berechnet, um die Veränderung zu erhalten.

3. Das WHERE filtert die Ergebnisse so, dass nur Zeilen zurückgegeben werden, bei denen ein vorheriges IMDb-Rating (prev_rating) vorhanden ist. **Somit fliegt in diesem Beispiel die erste Episode von jeder Staffel aus dem Ergebnis raus.**

Non equi self join

2 Constraints

1. Fügen Sie der Tabelle Vertrag einen CHECKConstraint hinzu, so dass Abopreise niemals kleiner werden können als 10€. Verwenden Sie hierfür eine ALTER TABLE Anweisung.
2. Fügen Sie einen Constraint hinzu, der verbietet, dass negative Ratings für Episoden eingetragen werden. Benennen Sie den Constraint mit „keine_negativen_rating“.
3. Fügen Sie weiterhin einen DEFAULT-Wert von 8 für die Ratings hinzu. Verwenden Sie hierfür jeweils ALTER TABLE Anweisungen.
4. Testen Sie die zusätzlich erstellen Constraints.

1.)

CHECK Constraint für die Tabelle Vertrag, sodass Abopreise niemals kleiner als 10€ werden können.

```
ALTER TABLE Vertrag ADD CONSTRAINT check_monatspreis CHECK  
(monatspreis >= 10);
```

2.)

Constraint hinzufügen, der verbietet, dass negative Ratings für Episoden eingetragen werden, benannt als „keine_negativen_rating“.

```
ALTER TABLE Episode ADD CONSTRAINT keine_negativen_rating CHECK  
(imdbrating >= 0);
```

3.)

DEFAULT-Wert von 8 für die Ratings hinzufügen.

```
ALTER TABLE Episode ALTER COLUMN imdbrating SET DEFAULT 8;
```


4.)

Testen des CHECK Constraints für **Vertrag** (Abo-Preis >= 10):

```
INSERT INTO Vertrag (name, email, monatspreis, vertragslaufzeit) VALUES  
( 'Amazon Prime', 'fbi@h-da.de', 5, '2024-12-31');
```

```
ERROR: Failing row contains (Amazon Prime, fbi@h-da.de, 5, 2024-12-31).new row for relation "vertrag" violates check constraint  
"check_monatspreis"
```

```
ERROR: new row for relation "vertrag" violates check constraint "check_monatspreis"
```

```
SQL state: 23514
```

```
Detail: Failing row contains (Amazon Prime, fbi@h-da.de, 5, 2024-12-31).
```

Testen des CHECK Constraints für **Episode** (keine negativen Ratings):

```
INSERT INTO Episode (seriename, staffel, nummer, titel, imdbrating)  
VALUES ( 'Breaking Bad', 6, 2, 'Another Episode', -1);
```

```
ERROR: Failing row contains (Breaking Bad, 2, 6, Another Episode, -1).new row for relation "episode" violates check constraint  
"keine_negativen_rating"
```

```
ERROR: new row for relation "episode" violates check constraint "keine_negativen_rating"
```

```
SQL state: 23514
```

```
Detail: Failing row contains (Breaking Bad, 2, 6, Another Episode, -1).
```




Testen des DEFAULT-Werts für **imdbrating**:

```
-- Insert ohne Rating-Angabe, sollte den Default-Wert 8 setzen
```

```
INSERT INTO Episode (seriename, staffel, nummer, titel)  
VALUES ( 'Breaking Bad', 6, 3, 'Default Rating Episode');
```

```
-- Überprüfen, ob der Default-Wert gesetzt wurde
```

```
SELECT seriename, staffel, nummer, titel, imdbrating  
FROM Episode  
WHERE seriename = 'Breaking Bad' AND staffel = 6 AND nummer = 3;
```

	seriename [PK] character varying (254) 	staffel [PK] integer 	nummer [PK] integer 	titel character varying (254) 	imdbrating numeric 
1	Breaking Bad	6	3	Default Rating Episode	8