

Abgabetermin: 24.10.2018, 13:30 Uhr
Abgabeform elektronisch

<input type="checkbox"/> DES31UE Niklas	Name	Niklas Vest	Aufwand in h	10
<input type="checkbox"/> DES32UE Niklas				
<input checked="" type="checkbox"/> DES33UE Traxler	Punkte		Kurzzeichen Tutor	

Hinweise und Richtlinien:

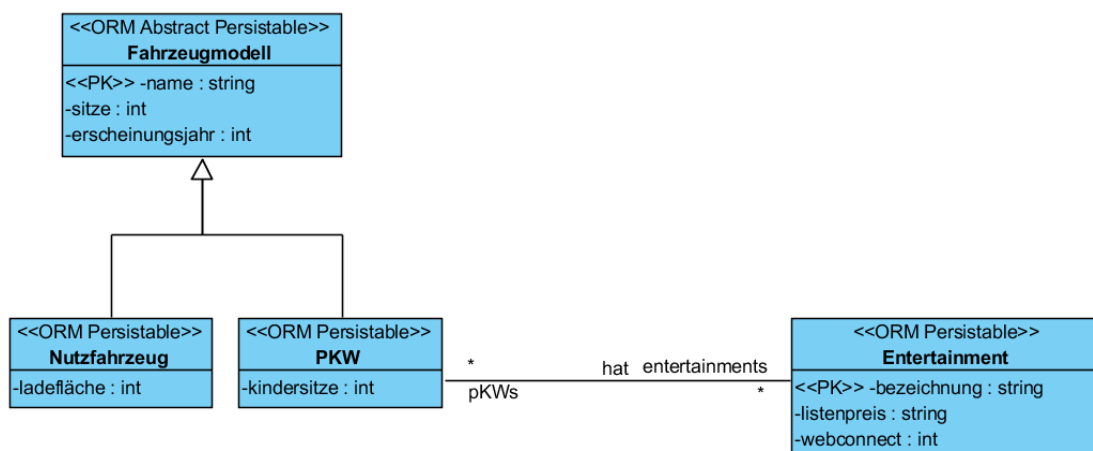
- Übungsausarbeitungen müssen den im eLearning angegebenen Formatierungsrichtlinien entsprechen – Nichtbeachtung dieser Formatierungsrichtlinien führt zu Punktabzug.
- Zusätzlich zu den allgemeinen Formatierungsrichtlinien sind für diese Übungsausarbeitung folgende zusätzlichen Richtlinien zu beachten:
 - Treffen Sie, falls notwendig, sinnvolle Annahmen und dokumentieren Sie diese nachvollziehbar in ihrer Lösung!
 - Recherchieren Sie eventuell unbekannte Elemente nach Bedarf.

Ziel dieser Übung ist es, ausgewählte Bereiche der Anfragesprache SQL zu wiederholen und zu vertiefen. Durch komplexe Beispiele wird demonstriert wie einzelne SQL Konstrukte kombiniert werden können. Verwenden Sie für die Ausarbeitung je nach Aufgabenstellung die zur Verfügung gestellte HR-Datenbank bzw. die Sakila-Datenbank.

1. Abbildung Generalisierung

(3 Punkte)

Erstellen Sie aus dem in angegebenen UML-Klassen-Diagramm ein **Relationenmodell**, um die Attribute in Tabellen in der Datenbank abzubilden, zB Tabelle 1 (PKAttr1, Attr2). Wählen Sie ein geeignetes Abbildungsmodell für die Generalisierungsbeziehung. Nennen Sie die Vor- und Nachteile des gewählten Abbildungsmodells und nehmen Sie dabei Bezug auf die Eigenschaften der Generalisierungsbeziehung, d.h. ob diese vollständig/unvollständig und überlappend/disjunkt ist. Dabei gilt: Ein Fahrzeughersteller verwaltet die Fahrzeuge von zwei Sparten Nutzfahrzeuge und PKWs. Die Sparten werden getrennt verwaltet, ein Modell wird nicht in beiden Sparten hergestellt.



2. Aggregate und Gruppierungen (Human Resources)

(3 Punkte)

Erstellen Sie die folgenden Statistikberichte für die Personalabteilung: Nehmen Sie die Abteilungsnummer, den Namen und die Anzahl der Angestellten für jede Abteilung auf, die folgende Bedingungen erfüllt:

- | | |
|--|---------|
| a. Weniger als 3 Angestellte: | 1 Punkt |
| b. Höchste Anzahl von Angestellten: | 1 Punkt |
| c. Niedrigste Anzahl von Angestellten: | 1 Punkt |

Bitte beachten Sie, dass es auch Abteilungen ohne Mitarbeiter geben kann.

3. Aggregate und Gruppierungen (Sakila-Datenbank)

(11 Punkte)

1. Welche Schauspieler/Schauspielerinnen sind in der Datenbank öfters als einmal eingetragen? (0,5 Punkt)
2. Ermitteln Sie jene Filme (Titel ausgeben), die zwar in der Filmdatenbank existieren, allerdings in keinem Geschäft angeboten werden. (42 Zeilen, 1 Punkt)
3. Geben Sie an, welcher Verkäufer in welchem Geschäft wie viele Verleihungen durchgeführt hat und wie viel Umsatz er (jeweils) erzielt hat – Achtung, der Verkäufer der die Verleihung eines Films durchgeführt hat, kann sich vom Verkäufer unterscheiden, der die Abrechnung (payment) durchgeführt hat, gehen Sie also davon aus, dass der Umsatz für jenen Verkäufer gezahlt wird, der die Verleihung durchgeführt hat! (1 Punkt)
4. Geben Sie pro Kunde (Vorname, Nachname) die Summe geliehener DVDs an. Sortieren Sie nach der Anzahl absteigend. (1 Punkt)
5. Geben Sie den besten Kunden (gemessen am Umsatz) pro Store an (Name, Umsatz, Store-ID des Kunden). (1,5 Punkte)
6. Geben Sie in Absteigender Reihenfolge an, welcher Kunde (Vorname, Nachname) bereits vier oder mehr Horror-Filme ausgeliehen hat. (34 Zeilen, 1,5 Punkte)
7. Geben Sie die Top 10 der Schauspieler aus (mit Vor- und Nachnamen), die in den meisten Filmen mitgespielt haben. (10 Zeilen, keine Ties berücksichtigen, 1,5 Punkte)
8. Geben Sie jene fünf Kategorien an, für die am wenigsten Filme zum Verleih (Inventory) stehen. (5 Zeilen, keine Ties berücksichtigen, 1,5 Punkte)
9. Bestimmen Sie den kürzesten Film (Titel ausgeben), der der längste Film einer Kategorie ist, zu der er gehört. (1,5 Punkte)

4. GROUP BY mit ROLLUP (Sakila-Datenbank)

(3 Punkte)

Erstellen Sie eine Filmabfrage und ermitteln Sie die Anzahl der Filme und die Summe der Längen in den Kategorien ‚Comedy‘ und ‚Music‘

- je Rating
- je Kategorie und Rating
- „Comedy“ und „Music“ gesamt

Entwickeln Sie drei Varianten, einmal mit der Verwendung des GROUP BY ROLLUP Operators (1 Punkt) und zum Vergleich dazu mit GROUPING SETS (1 Punkt) und komplett ohne ROLLUP und SETS (Hinweis: 3 Selects mit UNION) (1 Punkt). Vergleichen Sie die Zugriffspläne und kommentieren Sie diese.

Hinweis: SET AUTOTRACE ON erlaubt in SQL*PLUS die Ausgabe des Zugriffsplan sowie einiger Statistiken eines SQL-Statement und bietet eine einfache Möglichkeit zum Vergleich von Abfragen.

5. Group By mit CUBE (Sakila-Datenbank)

(4 Punkte)

- Erstellen Sie eine Abfrage, um für alle Manager, die in den Stores angestellt sind, folgendes anzuzeigen (1 Punkte):
 - Manager-Id
 - Store und Gesamterlös für jeden Manager
 - Gesamterlöse aller Manager
 - Kreuztabellenwerte für die Anzeige des Gesamterlöses für jeden Standort
 - Gesamterlös, unabhängig vom Standort

(Auszug aus der Lösung; Bitte vervollständigen Sie die Tabelle in Ihrer Abgabe)

MANAGER-ID	STORE	Gesamterlös
NULL	NULL	115657,58
NULL	1	44694,83
...
...
3	NULL	13074,71
3	3	6620,07
3	6	6454,64
...

- Prüfen Sie die Ausgabe der obigen Aufgabe. Schreiben Sie mit der GROUPING-Funktion eine Abfrage, um festzustellen, ob die Nullwerte in den Spalten, die den GROUP BY Ausdrücken entsprechen, von der CUBE-Operation verursacht werden. (0,5 Punkte)
- Erstellen Sie mit GROUPING SETS eine Abfrage, um folgende Gruppierungen anzuzeigen: (1 Punkt)
 - manager_staff_id, store_id, staff_id
 - manager_staff_id, store_id
 - store_id, staff_id

Die Abfrage soll die Summe der Erlöse für jede dieser Gruppen berechnen.

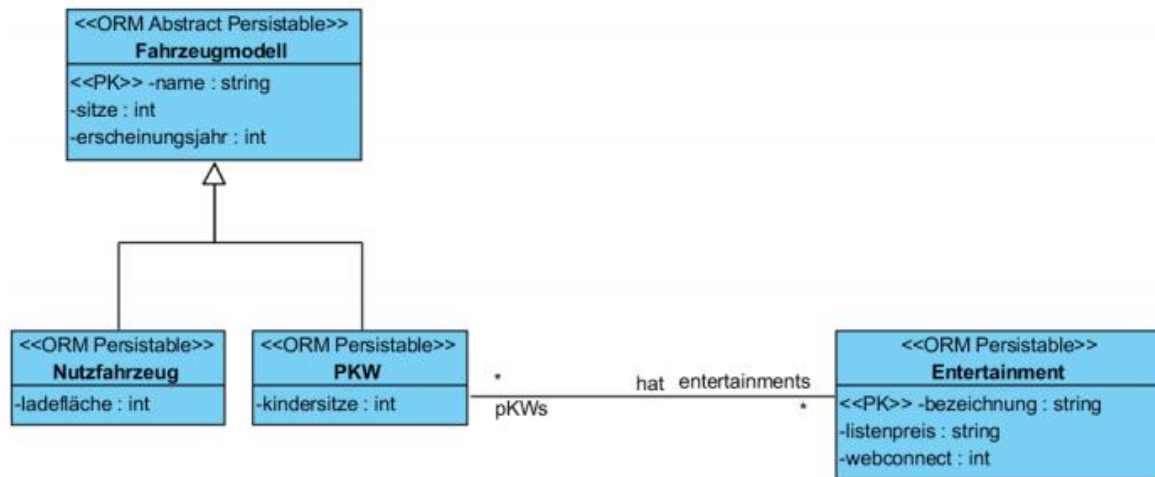
- Erstellen Sie für die Geschäftsleitung einen Bericht, der die Anzahl der Bezahlvorgänge und die Summe des Umsatzes pro Land und Jahr und pro Mitarbeiter-ID (der den Film verliehen hat) und Jahr zusammenfasst. Zusätzlich sollen Anzahl und Summe auch für alle Bezahlvorgänge angezeigt werden. Erstellen Sie ein SQL-Statement und verwenden Sie GROUPING SETS. (37 Zeilen) (1,5 Punkte)

(Auszug aus der Lösung; Bitte vervollständigen Sie die Tabelle in Ihrer Abgabe)

COUNTRY	YEAR	STAFF_ID	SUM(AMOUNT)	COUNT(*)
Japan	2013	NULL	626,80	73
Japan	2014	NULL	10402,46	1430
...
NULL	2013	1	1288,23	153
NULL	2013	2	1187,23	164
...
NULL	NULL	NULL	115647,75	16044
...

Ausarbeitung Übung 03

Abbildung Generalisierung



Die Generalisierung ist *complete*, da das Fahrzeugmodell als abstrakte Klasse modelliert ist. Sie ist außerdem *disjoint*, da ein Modell laut Angabe in beide Sparten PKW **und** Nutzfahrzeug fallen kann. Beim Einrelationenmodell wäre die Beziehung **hat** polymorph, was gegen die Modellierung spricht. Durch die Vollständigkeit der Generalisierung bietet sich die volle Redundanz eher schlecht an. Aus der Verbleibenden Auswahl empfinde ich das Partitionierungsmodell als umständlich da für eine Projektion mit allen Attributen mehrere Verbunde notwendig sind. So habe ich mich also für das Basisrelationenmodell entschieden:

Fahrzeugmodell: {[name: string, sitze: int, erscheinungsjahr: int]}

Nutzfahrzeug: {[name: string, sitze: int, erscheinungsjahr: int, ladefläche: int]}

Fahrzeugmodell: {[name: string, sitze: int, erscheinungsjahr: int, kindersitze: int]}

Entertainment: {[bezeichnung: string, listenpreis: string, webconnect: int]}

hat: {[name: string, bezeichnung: string]}

Aggregate und Gruppierungen

-- 2 a

```
SELECT department_id, department_name, Count(*)
FROM departments
    INNER JOIN employees USING (department_id)
GROUP BY department_id, department_name
HAVING Count(*) < 3;
```

	DEPARTMENT_ID ÷	DEPARTMENT_NAME ÷	COUNT(*) ÷
1	10	Administration	1
2	110	Accounting	2
3	20	Marketing	2

-- 2 b

```
SELECT department_id, department_name, Count(*)
FROM departments
    INNER JOIN employees USING (department_id)
GROUP BY department_id, department_name
HAVING Count(*) = (SELECT MAX(COUNT(employee_id))
    FROM departments
        INNER JOIN employees USING (department_id)
        GROUP BY department_id);
```

	DEPARTMENT_ID ÷	DEPARTMENT_NAME ÷	COUNT(*) ÷
1	50	Shipping	5

-- 2 c

```
SELECT department_id, department_name, Count(*)
FROM departments
    INNER JOIN employees USING (department_id)
GROUP BY department_id, department_name
HAVING Count(*) = (SELECT MIN(COUNT(employee_id))
    FROM departments
        INNER JOIN employees USING (department_id)
        GROUP BY department_id);
```

	DEPARTMENT_ID ÷	DEPARTMENT_NAME ÷	COUNT(*) ÷
1	10	Administration	1

Aggregate und Gruppierungen

-- 3.1 Assumption: The exercise asks for actors who played in multiple films

```
SELECT first_name, last_name, actor_id
FROM actor
      INNER JOIN film_actor USING (actor_id)
GROUP BY first_name, last_name, actor_id
HAVING (COUNT(*) > 1);
```

	FIRST_NAME	LAST_NAME	ACTOR_ID
1	JAYNE	NEESON	62
2	RIP	WINSLET	68
3	KIRK	JOVOVICH	43
4	VIVIEN	BASINGER	158
5	OLYMPIA	PFEIFFER	171
6	EMILY	DEE	148
7	GEOFFREY	HESTON	151
8	RIP	CRAWFORD	26
9	JUDY	DEAN	35
10	MORGAN	MCDORMAND	114
11	BURT	POSEY	75
12	ANGELINA	ASTAIRE	76
13	RALPH	CRUZ	80
14	BEN	WILLIS	83
15	MFG	HAUKE	97

-- 3.2

```
SELECT title
FROM film
WHERE film_id NOT IN (SELECT film_id FROM inventory);
```

	TITLE
1	ARSENIC INDEPENDENCE
2	DELIVERANCE MULHOLLAND
3	APOLLO TEEN
4	DAZED PUNK
5	TREASURE COMMAND
6	CRYSTAL BREAKING
7	PEARL DESTINY
8	SUICIDES SILENCE
9	PSYCHO SHRUNK
10	FRANKENSTEIN STRANGER
11	PERDITION FARGO
12	RAINBOW SHOCK
13	BOONDOCK BALLROOM
14	HATE HANDICAP
15	KENTUCKIAN GIANT

-- 3.3

```
SELECT r.staff_id, s.store_id, COUNT(*), SUM(amount)
FROM rental r
      INNER JOIN payment USING (rental_id)
      INNER JOIN staff s ON (s.staff_id = r.staff_id)
GROUP BY r.staff_id, s.store_id;
```

	STAFF_ID	STORE_ID	COUNT(*)	SUM(AMOUNT)
1	6	6	911	6454.64
2	5	5	930	6662.75
3	4	4	891	6657.14
4	1	1	6216	45287.33
5	2	2	6178	43965.82
6	3	3	918	6620.07

-- 3.4

```
SELECT first_name, last_name, COUNT(*)
FROM customer
      INNER JOIN rental USING (customer_id)
GROUP BY first_name, last_name
ORDER BY COUNT(*) DESC;
```

	FIRST_NAME	LAST_NAME	COUNT(*)
1	ELEANOR	HUNT	46
2	KARL	SEAL	45
3	MARCIA	DEAN	42
4	CLARA	SHAW	42
5	TAMMY	SANDERS	41
6	WESLEY	BULL	40
7	SUE	PETERS	40
8	MARION	SNYDER	39
9	RHONDA	KENNEDY	39
10	TIM	CARY	39
11	DAISY	BATES	38
12	ELIZABETH	BROWN	38
13	TOMMY	COLLAZO	38
14	CURTIS	IRBY	38
15	JUNIF	CARROLL	37

-- 3.5

```
SELECT customer_id, last_name, store_id, SUM(amount)
FROM payment INNER JOIN customer c1 USING (customer_id)
GROUP BY customer_id, last_name, store_id
HAVING SUM(amount) >= ALL (SELECT SUM(amount)
                           FROM payment INNER JOIN customer c2 USING (customer_id)
                           WHERE c1.store_id = c2.store_id
                           GROUP BY customer_id, last_name, store_id);
```

	CUSTOMER_ID	LAST_NAME	STORE_ID	SUM(AMOUNT)
1	75	SANDERS	3	305.88
2	236	DEAN	1	357.39
3	468	CARY	5	327.14
4	408	MURRELL	4	282.32
5	469	BULL	2	342.67
6	144	SHAW	6	322.62

-- 3.6

```
SELECT first_name, last_name, COUNT(*) AS number_of_horror_rental
FROM customer
      INNER JOIN rental USING (customer_id)
      INNER JOIN inventory USING (inventory_id)
      INNER JOIN film_category USING (film_id)
      INNER JOIN category USING (category_id)
WHERE name = 'Horror'
GROUP BY first_name, last_name
HAVING COUNT(*) >= 4
ORDER BY COUNT(*) DESC;
```

	FIRST_NAME	LAST_NAME	NUMBER_OF_HORROR_RENTAL
1	DUANE	TUBBS	5
2	KEN	PREWITT	5
3	KARL	SEAL	5
4	NANCY	THOMAS	5
5	THELMA	MURRAY	5
6	EMMA	BOYD	5
7	ROSEMARY	SCHMIDT	5
8	ANA	BRADLEY	5
9	JOSE	ANDREW	4
10	CARL	ARTIS	4
11	TINA	SIMMONS	4
12	KRISTINA	CHAMBERS	4
13	LOUIS	LEONE	4
14	ZACHARY	HITE	4
15	OSCAR	ANUTNO	4

-- 3.7

```
SELECT actor_id, last_name
FROM actor
      INNER JOIN film_actor USING (actor_id)
GROUP BY actor_id, last_name
ORDER BY COUNT(*) DESC
FETCH FIRST 10 ROWS ONLY;
```

	ACTOR_ID	LAST_NAME
1	107	DEGENERES
2	102	TORN
3	198	KEITEL
4	181	CARREY
5	23	KILMER
6	81	DAMON
7	158	BASINGER
8	60	BERRY
9	106	DUNST
10	13	WOOD

-- 3.8

```
SELECT name, category_id, COUNT(*)
FROM category
      INNER JOIN film_category USING (category_id)
      INNER JOIN film USING (film_id)
      INNER JOIN inventory USING (film_id)
GROUP BY category_id, name
ORDER BY COUNT(*) ASC
FETCH FIRST 5 ROWS ONLY;
```

	NAME	CATEGORY_ID	COUNT(*)
1	Music	12	232
2	Travel	16	235
3	Horror	11	248
4	Comedy	5	269
5	Children	3	269

-- 3.9

```
SELECT film_id, title, length, category_id
FROM film
      INNER JOIN film_category USING (film_id)
WHERE (category_id, length) IN (SELECT category_id, MAX(length)
                                FROM film
                                INNER JOIN film_category fc USING (film_id)
                                GROUP BY category_id)
ORDER BY length ASC
FETCH FIRST 1 ROWS WITH TIES;
```

	FILM_ID	TITLE	LENGTH	CATEGORY_ID
1	344	FURY MURDER	178	3
2	993	WRONG BEHAVIOR	178	3

GROUP BY und ROLLUP

	RATING	NAME	Nr. Of Films	Total Length (All Films)
1	G	Music	2	132
2	G	Comedy	11	1364
3	G	<null>	13	1496
4	R	Music	11	1088
5	R	Comedy	8	950
6	R	<null>	19	2038
7	PG	Music	10	1187
8	PG	Comedy	16	1869
9	PG	<null>	26	3056
10	NC-17	Music	20	2260
11	NC-17	Comedy	11	1255
12	NC-17	<null>	31	3515
13	PG-13	Music	8	1129
14	PG-13	Comedy	12	1280
15	PG-13	<null>	20	2409
16	<null>	<null>	109	12514

Figure 1 Ergebnis für alle drei Abfragen

```
-- 4.1
SELECT rating, name, COUNT(*) AS "Nr. Of Films", SUM(length) AS "Total Length (All Films)"
FROM film
    INNER JOIN film_category USING (film_id)
    INNER JOIN category USING (category_id)
WHERE name IN('Comedy', 'Music')
GROUP BY ROLLUP (rating, name);

-- 4.2
SELECT rating, name, COUNT(*) AS "Nr. Of Films", SUM(length) AS "Total Length (All Films)"
FROM film
    INNER JOIN film_category USING (film_id)
    INNER JOIN category USING (category_id)
WHERE name IN('Comedy', 'Music')
GROUP BY GROUPING SETS ( (rating),
                          (name, rating),
                          ());

-- 4.3
SELECT rating, name, COUNT(*) AS "Nr. Of Films", SUM(length) AS "Total Length (All Films)"
FROM film
    INNER JOIN film_category USING (film_id)
    INNER JOIN category USING (category_id)
WHERE name IN('Comedy', 'Music')
GROUP BY rating, name
UNION ALL
SELECT rating, NULL AS name, COUNT(*) AS "Nr. Of Films", SUM(length) AS "Total Length (All Films)"
FROM film
    INNER JOIN film_category USING (film_id)
    INNER JOIN category USING (category_id)
WHERE name IN('Comedy', 'Music')
GROUP BY rating, NULL
UNION ALL
SELECT NULL AS rating, NULL AS name, COUNT(*) AS "Nr. Of Films", SUM(length) AS "Total Length (All Films)"
FROM film
    INNER JOIN film_category USING (film_id)
    INNER JOIN category USING (category_id)
WHERE name IN('Comedy', 'Music');
```

GROUP BY mit CUBE

-- 5.1

```
SELECT manager_staff_id, store_id, SUM(amount)
FROM store s
      INNER JOIN staff USING (store_id)
      INNER JOIN payment USING (staff_id)
GROUP BY CUBE (manager_staff_id, store_id);
```

	MANAGER_STAFF_ID ÷	STORE_ID ÷	SUM(AMOUNT) ÷
1	<null>	<null>	115657.58
2	<null>	1	44694.83
3	<null>	2	44568.15
4	<null>	3	6620.07
5	<null>	4	6657.14
6	<null>	5	6662.75
7	<null>	6	6454.64
8	1	<null>	44694.83
9	1	1	44694.83
10	2	<null>	44568.15
11	2	2	44568.15
12	3	<null>	13074.71
13	3	3	6620.07
14	3	6	6454.64
15	4	<null>	6657.14
16	4	4	6657.14
17	5	<null>	6662.75
18	5	5	6662.75

-- 5.2

```
SELECT manager_staff_id, store_id, SUM(amount),
      GROUPING(manager_staff_id) grp_msi,
      GROUPING(store_id) grp_sid
FROM store s
      INNER JOIN staff USING (store_id)
      INNER JOIN payment USING (staff_id)
GROUP BY CUBE (manager_staff_id, store_id);
```

	MANAGER_STAFF_ID ÷	STORE_ID ÷	SUM(AMOUNT) ÷	GRP_MSI ÷	GRP_SID ÷
1	<null>	<null>	115657.58	1	1
2	<null>	1	44694.83	1	0
3	<null>	2	44568.15	1	0
4	<null>	3	6620.07	1	0
5	<null>	4	6657.14	1	0
6	<null>	5	6662.75	1	0
7	<null>	6	6454.64	1	0
8	1	<null>	44694.83	0	1
9	1	1	44694.83	0	0
10	2	<null>	44568.15	0	1
11	2	2	44568.15	0	0
12	3	<null>	13074.71	0	1
13	3	3	6620.07	0	0
14	3	6	6454.64	0	0
15	4	<null>	6657.14	0	1
16	4	4	6657.14	0	0
17	5	<null>	6662.75	0	1
18	5	5	6662.75	0	0

-- 5.3

```
SELECT manager_staff_id, store_id, staff_id, SUM(amount)
FROM store s
      INNER JOIN staff USING (store_id)
      INNER JOIN payment USING (staff_id)
GROUP BY GROUPING SETS ((manager_staff_id, store_id, staff_id),
                        (manager_staff_id, store_id),
                        (store_id, staff_id));
```

	MANAGER_STAFF_ID	STORE_ID	STAFF_ID	SUM(AMOUNT)
1	1	1	1	44694.83
2	2	2	2	44568.15
3	3	3	3	6620.07
4	4	4	4	6657.14
5	5	5	5	6662.75
6	3	6	6	6454.64
7	<null>	1	1	44694.83
8	<null>	2	2	44568.15
9	<null>	3	3	6620.07
10	<null>	4	4	6657.14
11	<null>	5	5	6662.75
12	<null>	6	6	6454.64
13	5	5	<null>	6662.75
14	3	6	<null>	6454.64
15	4	4	<null>	6657.14
16	1	1	<null>	44694.83
17	2	2	<null>	44568.15
18	3	3	<null>	6620.07

-- 5.4

```

SELECT country, EXTRACT(YEAR FROM payment_date), r.staff_id, sum(amount), count(*)
FROM payment
  INNER JOIN rental r USING (rental_id)
  INNER JOIN staff s ON r.staff_id = s.staff_id
  INNER JOIN inventory i USING (inventory_id)
  INNER JOIN store sto ON sto.store_id = i.store_id
  INNER JOIN address a ON sto.address_id = a.address_id
  INNER JOIN city USING (city_id)
  INNER JOIN country USING (country_id)
GROUP BY GROUPING SETS ( (country_id, country, EXTRACT(YEAR FROM payment_date)),
                          (r.staff_id, EXTRACT(YEAR FROM payment_date)),
                          ());

```

	COUNTRY	EXTRACT(YEAR FROM PAYMENT_DATE)	STAFF_ID	SUM(AMOUNT)	COUNT(*)
1	Japan	2013	<null>	626.8	73
2	Japan	2014	<null>	10402.46	1430
3	Japan	2015	<null>	8747.85	1219
4	Egypt	2013	<null>	521.95	62
5	Australia	2013	<null>	465.24	64
6	Canada	2015	<null>	8518.11	1172
7	Austria	2013	<null>	691.94	75
8	Egypt	2015	<null>	9011.33	1272
9	Australia	2014	<null>	9848.39	1383
10	United States	2013	<null>	382.58	55
11	United States	2014	<null>	10199.14	1400
12	Austria	2014	<null>	9544.62	1312
13	Austria	2015	<null>	8659.87	1241
14	Canada	2013	<null>	472.62	59
15	Canada	2014	<null>	9945.86	1371
16	Australia	2015	<null>	8519.39	1192
17	United States	2015	<null>	8648.36	1188
18	Egypt	2014	<null>	10441.24	1476
19	<null>	2013	1	1288.23	153
20	<null>	2013	2	1187.23	164
21	<null>	2013	3	166.54	18
22	<null>	2013	4	266.67	25
23	<null>	2013	5	173.32	18
24	<null>	2013	6	79.14	10
25	<null>	2014	1	23285.06	3201
26	<null>	2014	2	23607.95	3303
27	<null>	2014	3	3518.98	493
28	<null>	2014	4	3391.45	459
29	<null>	2014	5	3288.42	453
30	<null>	2014	6	3289.85	463
31	<null>	2015	1	20714.04	2862
32	<null>	2015	2	19170.64	2711
33	<null>	2015	3	2934.55	407
34	<null>	2015	4	2999.02	407
35	<null>	2015	5	3201.01	459
36	<null>	2015	6	3085.65	438
37	<null>	<null>	<null>	115647.75	16044