Darbas su duomenimis

8. Grupavimas

11. Sąlygos

Grupavimas

- SELECT customer_id FROM rental;
- SELECT customer_id FROM rental GROUP BY customer_id;

```
SELECT customer_id, count(*)
    FROM rental
    GROUP BY customer_id;
```

```
SELECT customer_id, count(*)
    FROM rental
    GROUP BY customer_id
    ORDER BY 2 DESC;
```

Grupių filtravimas

```
SELECT customer_id, count(*)
    FROM rental
   WHERE count(*) >= 40
    GROUP BY customer_id;
```

```
SELECT customer_id, count(*)
    FROM rental
    GROUP BY customer_id
    HAVING count(*) >= 40;
```

Agregavimo funkcijos

- Max
- Min
- Avg
- Sum
- Count

```
SELECT MAX(amount) max_amt,
       MIN(amount) min_amt,
      AVG(amount) avg_amt,
       SUM(amount) tot_amt,
      COUNT(*) num_payments
     FROM payment;
```

Turime nurodyti grupavimo stulpelį

```
SELECT customer_id,
       MAX(amount) max_amt,
       MIN(amount) min_amt,
       AVG(amount) avg_amt,
       SUM(amount) tot_amt,
       COUNT(*) num_payments
     FROM payment
     GROUP BY customer_id;
```

Skirtingų reikšmių skaičiavimas

```
SELECT COUNT(customer_id) num_rows,
       COUNT(DISTINCT customer_id) num_customers
     FROM payment;
```

Papildomų funkcijų naudojimas

```
SELECT MAX(datediff(return_date,rental_date))
    FROM rental;
```

NULL reikšmės

```
CREATE TABLE number_tbl  (val SMALLINT);
INSERT INTO number_tbl VALUES (1);
INSERT INTO number_tbl VALUES (3);
INSERT INTO number_tbl VALUES (5);

SELECT COUNT(*) num_rows,
    COUNT(val) num_vals,
    SUM(val) total,
    MAX(val) max_val,
    AVG(val) avg_val
    FROM number_tbl;

INSERT INTO number_tbl VALUES (NULL);
```

Vieno stulpelio grupavimas

```
SELECT actor_id, count(*)
    FROM film_actor
    GROUP BY actor_id;
```

Kelių stulpelių grupavimas

```
SELECT fa.actor_id, f.rating, count(*)
    FROM film_actor fa
      INNER JOIN film f
      ON fa.film_id = f.film_id
    GROUP BY fa.actor_id, f.rating
    ORDER BY 1,2;
```

Grupavimas naudojantis papildoma funkcija

```
SELECT extract(YEAR FROM rental_date) year,
       COUNT(*) how_many
     FROM rental
     GROUP BY extract(YEAR FROM rental_date);
```

Roll up – suskaičiuoja kiekvienos grupės narių skaičių

```
SELECT fa.actor_id, f.rating, count(*)
    FROM film_actor fa
       INNER JOIN film f
       ON fa.film_id = f.film_id
     GROUP BY fa.actor_id, f.rating WITH ROLLUP
     ORDER BY 1,2;
```

Filtravimas

```
SELECT fa.actor_id, f.rating, count(*)
   FROM film_actor fa
     INNER JOIN film f
     ON fa.film_id = f.film_id
  WHERE f.rating IN ('G','PG')
  GROUP BY fa.actor_id, f.rating
  HAVING count(*) > 9;
```

```
SELECT fa.actor_id, f.rating, count(*)
   FROM film_actor fa
     INNER JOIN film f
     ON fa.film_id = f.film_id
   WHERE f.rating IN ('G','PG')
    AND count(*) > 9
  GROUP BY fa.actor_id, f.rating;
```

Užduotys

- Parašykite užklausą, kuri suskaičiuoja eilučių skaičių payment lentelėje.
- Suskaičiuokite kiekvieno kliento mokėjimų sumą.
- Suskaičiuokite kiekvieno kliento mokėjimų sumą ir lentelėje palikite tik tuos, kurie sumokėjo 40 ir daugiau kartų.

Sąlygos

Galimybė rinktis iš kelių variantų

Pvz.

```
SELECT first_name, last_name,
      CASE
         WHEN active = 1 THEN 'ACTIVE'
         ELSE 'INACTIVE'
       END activity_type
     FROM customer;
```

Search Sąlygos išraiška

```
CASE
  WHEN C1 THEN E1
  WHEN C2 THEN E2
  ...
  WHEN CN THEN EN
  [ELSE ED]
END
```

Search Sąlygos išraiškos pavyzdys

```
select
CASE
  WHEN category.name IN ('Children','Family','Sports','Animation')
    THEN 'All Ages'
  WHEN category.name = 'Horror'
    THEN 'Adult'
  WHEN category.name IN ('Music','Games')
    THEN 'Teens'
  ELSE 'Other'
end as category_group
from category ;
```

Sąlygos

- Sąlygos tikrinamos iš viršaus į apačią.
- Sąlygos gali gražinti bet kokio tipo duomenis, pvz. Subquery

```
SELECT c.first_name, c.last_name,
      CASE
         WHEN active = 0 THEN 0
         ELSE
          (SELECT count(*) FROM rental r
          WHERE r.customer_id = c.customer_id)
       END num_rentals
     FROM customer c;
```

Case sąlygos

```
CASE V0
  WHEN V1 THEN E1
  WHEN V2 THEN E2
  ...
  WHEN VN THEN EN
  [ELSE ED]
END
```

```
CASE category.name
  WHEN 'Children' THEN 'All Ages'
  WHEN 'Family' THEN 'All Ages'
  WHEN 'Sports' THEN 'All Ages'
  WHEN 'Animation' THEN 'All Ages'
  WHEN 'Horror' THEN 'Adult'
  WHEN 'Music' THEN 'Teens'
  WHEN 'Games' THEN 'Teens'
  ELSE 'Other'
END
```

Panaudojimo pavyzdžiai

```
SELECT monthname(rental_date) rental_month,
   count(*) num_rentals
      FROM rental
  WHERE rental_date BETWEEN '2005-05-01' AND '2005-08-01'
  GROUP BY monthname(rental_date);
```

```
SELECT
   SUM(CASE WHEN monthname(rental_date) = 'May' THEN 1
      ELSE 0 END) May_rentals,
   SUM(CASE WHEN monthname(rental_date) = 'June' THEN 1
      ELSE 0 END) June_rentals,
   SUM(CASE WHEN monthname(rental_date) = 'July' THEN 1
      ELSE 0 END) July_rentals
  FROM rental
  WHERE rental_date BETWEEN '2005-05-01' AND '2005-08-01';
```

Panaudojimo pavyzdžiai

```
SELECT a.first_name, a.last_name,
      CASE
         WHEN EXISTS (SELECT 1 FROM film_actor fa
                        INNER JOIN film f ON fa.film_id =
f.film_id
                       WHERE fa.actor_id = a.actor_id
AND f.rating = 'G') THEN 'Y'
ELSE 'N'
END g_actor,
CASE
WHEN EXISTS (SELECT 1 FROM film_actor fa
INNER JOIN film f ON fa.film_id = f.film_id
WHERE fa.actor_id = a.actor_id
AND f.rating = 'PG') THEN 'Y' ELSE 'N'
END pg_actor,
CASE
WHEN EXISTS (SELECT 1 FROM film_actor fa
INNER JOIN film f ON fa.film_id = f.film_id
WHERE fa.actor_id = a.actor_id
AND f.rating = 'NC-17') THEN 'Y'
ELSE 'N'
END nc17_actor
FROM actor a
WHERE a.last_name LIKE 'S%' OR a.first_name
LIKE 'S%';
```

Panaudojimo pavyzdžiai

```
SELECT f.title,
       CASE (SELECT count(*) FROM inventory i
             WHERE i.film_id = f.film_id)
         WHEN 0 THEN 'Out Of Stock'
         WHEN 1 THEN 'Scarce'
         WHEN 2 THEN 'Scarce'
         WHEN 3 THEN 'Available'
         WHEN 4 THEN 'Available'
         ELSE 'Common'
       END film_availability
     FROM film f ;
```

Panaudojimo pavyzdžiai

```
SELECT 100 / 0; -- dalyba iš nulio gražina null.

SELECT c.first_name, c.last_name,
       sum(p.amount) tot_payment_amt,
      count(p.amount) num_payments,
       sum(p.amount) /
         CASE WHEN count(p.amount) = 0 THEN 1
           ELSE count(p.amount)
        END avg_payment
     FROM customer c
      LEFT OUTER JOIN payment p
      ON c.customer_id = p.customer_id
     GROUP BY c.first_name, c.last_name;
```

Panaudojimo pavyzdžiai

```
UPDATE customer
SET active =
  CASE
    WHEN 90 <= (SELECT datediff(now(), max(rental_date))
                FROM rental r
                WHERE r.customer_id = customer.customer_id)
      THEN 0
    ELSE 1
  END
WHERE active = 1;
```

Panaudojimo pavyzdžiai

```
SELECT c.first_name, c.last_name,
  CASE
    WHEN a.address IS NULL THEN
'Unknown'
    ELSE a.address
  END address,
  CASE
    WHEN ct.city IS NULL THEN 'Unknown'
    ELSE ct.city
  END city,
  CASE
    WHEN cn.country IS NULL THEN
'Unknown'
    ELSE cn.country
  END country
FROM customer c
  LEFT OUTER JOIN address a
  ON c.address_id = a.address_id
  LEFT OUTER JOIN city ct
  ON a.city_id = ct.city_id
  LEFT OUTER JOIN country cn
  ON ct.country_id = cn.country_id;
```

Užduotys

Perrašykite užklausą naudodami search sąlygos išraišką. Panaudokite kuo mažiau when sąlygų

```
SELECT name,
  CASE name
    WHEN 'English' THEN 'latin1'
    WHEN 'Italian' THEN 'latin1'
    WHEN 'French' THEN 'latin1'
    WHEN 'German' THEN 'latin1'
    WHEN 'Japanese' THEN 'utf8'
    WHEN 'Mandarin' THEN 'utf8'
    ELSE 'Unknown'
  END character_set
FROM language;
```

Užduotys

Perrašykite užklausą taip, kad vietoje stulpelio rating turėtumėte eilutes

```
SELECT rating, count(*)
    FROM film
    GROUP BY rating;
```