Database Systems – Assignment1

# q01

# q02

## Assumptions and remarks

1. assuming if there is a tie in sales numbers, all aligable stores should be returned.
2. first is the most rentals in july 2005, second is the least

## Query

SELECT store\_id AS good\_first\_bad\_second

FROM inventory JOIN

(

SELECT inventory\_id, rental\_id, rental\_date

FROM rental

WHERE YEAR(rental\_date) = 2005 AND MONTH(rental\_date) = 7

) AS july\_rentals

ON inventory.inventory\_id = july\_rentals.inventory\_id

GROUP BY inventory.store\_id

HAVING count(july\_rentals.rental\_id) >= ALL

(

SELECT count(rental\_id)

FROM inventory AS i

JOIN

(

SELECT inventory\_id, rental\_id, rental\_date

FROM rental

WHERE YEAR(rental\_date) = 2005 AND MONTH(rental\_date) = 7

) AS jr

ON i.inventory\_id = jr.inventory\_id

GROUP BY i.store\_id

)

UNION

SELECT store\_id

FROM inventory JOIN

(

SELECT inventory\_id, rental\_id, rental\_date

FROM rental

WHERE YEAR(rental\_date) = 2005 AND MONTH(rental\_date) = 7

) AS july\_rentals

ON inventory.inventory\_id = july\_rentals.inventory\_id

GROUP BY inventory.store\_id

HAVING count(july\_rentals.rental\_id) <= ALL

(

SELECT count(rental\_id)

FROM inventory AS i

JOIN

(

SELECT inventory\_id, rental\_id, rental\_date

FROM rental

WHERE YEAR(rental\_date) = 2005 AND MONTH(rental\_date) = 7

) AS jr

ON i.inventory\_id = jr.inventory\_id

GROUP BY i.store\_id

)

## Result

|  |
| --- |
| good\_first\_bad\_second |
| 2 |
| 1 |

# q03

## Assumptions and remarks

## Query

## Result

# q04

## Assumptions and remarks

None

## Query

SELECT LEFT(title, 1) AS most\_common\_letter

FROM film

GROUP BY LEFT(title, 1)

HAVING COUNT(LEFT(title, 1)) >= ALL

(

SELECT COUNT(LEFT(title, 1))

FROM film

GROUP BY LEFT(title, 1)

)

## Result

|  |
| --- |
| most\_common\_letter |
| S |

# q05

## Assumptions and remarks

## Query

## Result

# q06

## Assumptions and remarks

1. There are no store names, so the query shows the id

## Query

SELECT ps1.store\_id, MAX(ps1.sum\_per\_month - ps2.sum\_per\_month) AS earning\_difference

FROM

(

SELECT store\_id, MONTH(payment\_date) AS p\_month,

YEAR(payment\_date) AS p\_year, SUM(amount) AS sum\_per\_month

FROM staff AS s

JOIN payment AS p

ON s.staff\_id = p.staff\_id

GROUP BY store\_id, YEAR(payment\_date), MONTH(payment\_date)

) AS ps1,

(

SELECT store\_id, MONTH(payment\_date) AS p\_month,

YEAR(payment\_date) AS p\_year, SUM(amount) AS sum\_per\_month

FROM staff AS s

JOIN payment AS p

ON s.staff\_id = p.staff\_id

GROUP BY store\_id, YEAR(payment\_date), MONTH(payment\_date)

) AS ps2

WHERE ps1.store\_id = ps2.store\_id AND

ps1.p\_year = ps2.p\_year AND

ps1.p\_month = ps2.p\_month + 1

## Result

|  |  |
| --- | --- |
| store\_id | earning\_difference |
| 1 | 9514.83 |

# q07

## Assumptions and remarks

## Query

## Result

# q08

## Assumptions and remarks

1. assuming that if more than one actor answers the condition, we should return them all

## Query

SELECT CONCAT(first\_name, ' ', last\_name) AS actor\_name

FROM actor AS a

JOIN

(

SELECT fa.actor\_id, fa.film\_id, f.rating

FROM film\_actor AS fa

JOIN film AS f

ON fa.film\_id = f.film\_id

WHERE f.rating = 'R'

) AS faa

ON a.actor\_id = faa.actor\_id

GROUP BY actor\_name

HAVING COUNT(actor\_name) >= ALL

(

SELECT COUNT(a.actor\_id)

FROM actor AS a

JOIN

(

SELECT fa.actor\_id, fa.film\_id, f.rating

FROM film\_actor AS fa

JOIN film AS f

ON fa.film\_id = f.film\_id

WHERE f.rating = 'R'

) AS faa

ON a.actor\_id = faa.actor\_id

GROUP BY a.actor\_id

)

## Result

|  |
| --- |
| actor\_name |
| JULIANNE DENCH |
| MATTHEW LEIGH |
| SIDNEY CROWE |
| WARREN JACKMAN |

# q09

## Assumptions and remarks

## Query

## Result

# q10 – Our very own query

## Assumptions and remarks

1. "Diversity isn't all" – return the name of the actor who played the most kinds (categories) of movies, but didn't star in the most movies overall.
2. assuming one film only has one category

## Query

SELECT CONCAT(first\_name, ' ', last\_name) AS actor\_name

FROM actor

JOIN

(

SELECT actor\_id, category\_id AS category, fa.film\_id AS film

FROM film\_actor AS fa

JOIN

(

SELECT f.film\_id, fc.category\_id AS category\_id

FROM film AS f

JOIN film\_category AS fc

ON f.film\_id = fc.film\_id

) AS fc

ON fa.film\_id = fc.film\_id

) AS ac

ON ac.actor\_id = actor.actor\_id

GROUP BY actor\_name

HAVING COUNT(DISTINCT(category)) >= ALL

(

SELECT COUNT(DISTINCT(category\_id))

FROM film\_actor AS fa

JOIN

(

SELECT f.film\_id, fc.category\_id AS category\_id

FROM film AS f

JOIN film\_category AS fc

ON f.film\_id = fc.film\_id

) AS fc

ON fa.film\_id = fc.film\_id

GROUP BY actor\_id

) AND COUNT(film) < ANY

(

SELECT COUNT(fa.film\_id)

FROM film\_actor AS fa

JOIN

(

SELECT f.film\_id, fc.category\_id AS category\_id

FROM film AS f

JOIN film\_category AS fc

ON f.film\_id = fc.film\_id

) AS fc

ON fa.film\_id = fc.film\_id

GROUP BY actor\_id

)

## Result

|  |
| --- |
| actor\_name |
| DARYL WAHLBERG |
| EWAN GOODING |
| GROUCHO DUNST |
| HARVEY HOPE |
| IAN TANDY |
| KEVIN GARLAND |
| MICHAEL BOLGER |
| REESE WEST |
| SEAN WILLIAMS |
| UMA WOOD |