

Feladatmegoldások

1. Kis feladatok

1. Gábor tud dolgozni péntek kivételével mindegyik napon, mert azokon a napokon dolgozik Péter vagy István. Richárd bármelyik nap tud dolgozni, mert Tamás dolgozik pénteken is, amikor Gábor nem dolgozik, tehát ők ketten lefedik az összes napot a héten.

2. Az A és D egymást követik, szóval a D szerepelhet 2. és 4. helyen. A B megelőzi A-t, tehát 1. vagy 2. helyen lehet. Az F nem szomszédos C-vel, szóval az is 1. vagy 2. helyen lehet. A kérdőjel a 6. helyen szerepel, és oda egyetlen lépés se volt az eddigiek közül említve, így kizárásos alapon az E lépés fog a ? helyére kerülni. A lehetséges sorrendek a következők:

F -> B -> A -> D -> C -> **E** vagy B -> F -> A -> D -> C -> **E**.

3. A prímszámok azok a számok amelyek csak 1-gyel és önmagukkal oszthatók. Definiáljunk egy függvényt, amely visszaadja a paraméterben kapott számról, hogy prímszám-e. Szükségünk lesz egy tömbre, amelyben eltároljuk a prímszámokat (\$prims), kezdetben ez a tömb üres. Egyesével fogjuk vizsgálni a számokat, hogy prímszám-e, amíg el nem jutunk a kérdéses számig, ezt egy ciklussal végezzük el, 2-től indítjuk a ciklust (mivel tudjuk az 1 nem prímszám), és addig fog menni amíg el nem érünk a paraméterben kapott számig (\$p). Ezen a cikluson belül indítunk még egy ciklust, amivel a prímszámok tömbjén megyünk végig, itt vizsgáljuk meg, hogy az aktuális vizsgálandó szám (\$i) maradékosan osztható-e az eddig felvett prímszámokkal. Ha bármelyikkel is maradékosan osztható, nem prímszám, szóval nem tároljuk el a prímekek tömbjében (ha prímszám, hozzáadjuk). Így ha végigértünk az első ciklussal, az azt jelenti, hogy megvizsgáltuk a paraméterben kapott számot is, szóval ellenőrizzük szerepel-e ez a szám a prímekek között, ez alapján meg tudjuk mondani, hogy prímszám-e vagy sem.

```
function getPrim(int $p){
    $prims = array();

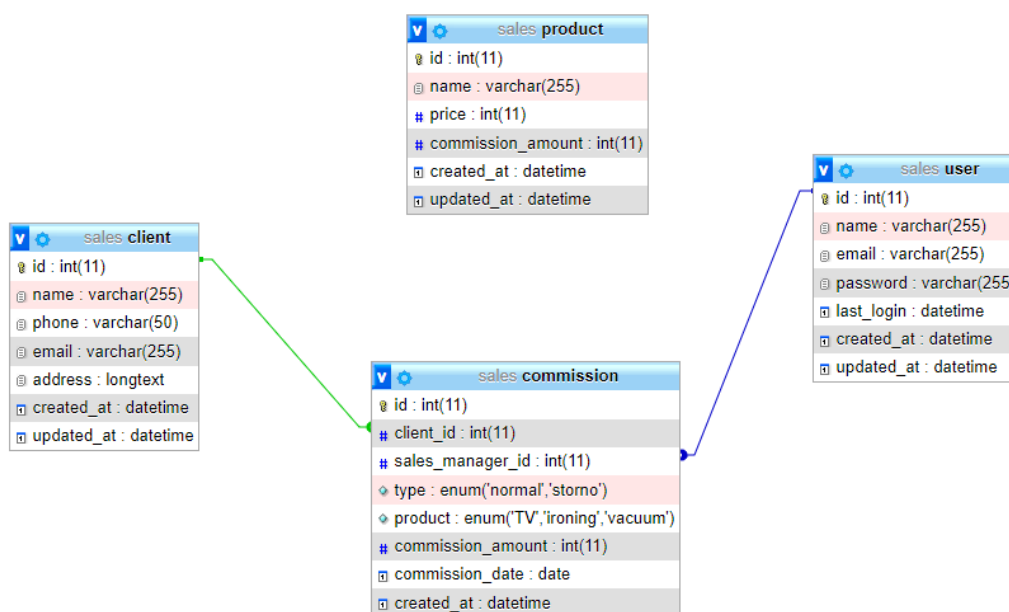
    for($i = 2; $i <= $p; $i++){
        $isPrim = true;
        foreach($prims as $onePrim){
            if($i % $onePrim === 0){
                $isPrim = false;
                break;
            }
        }

        if($isPrim){
            $prims[] = $i;
        }
    }

    if(in_array($p, $prims)){
        echo "Prim: ".$p;
    }else{
        echo "Not prim: ".$p;
    }
}
```

4. Deklaráljunk két változót, \$a és \$b. Az \$a értéke 20, a \$b értéke 10, ez 2 db egész szám. Először az \$a értékét egyenlővé teszem \$a + \$b összegnek értékével. Ekkor az \$a = 30, \$b = 10. Ezután a \$b értékét egyenlővé teszem az \$a – \$b különbségének értékével, ekkor \$a = 30, \$b = 20. Végül az \$a értékét egyenlővé tesszük az \$a – \$b különbségének értékével, vagyis \$a = 10, b = \$20.

5. Az UML ERD diagram:



Néhány dolog amit másként csinálnék:

- A commission táblába külső kulccsal bekötném a product táblát, mert most nincs kapcsolat a 2 tábla között. Szöveges értékek helyett egy productId oszlop lenne, amely a product.id oszlopára hivatkozik.

- A commission táblában a type mező határozza meg hogy normál vagy érvénytelenített az eladás. Beletennék egy plusz mezőt, hogy lehessen hivatkozni az érvénytelenített eladásra. Ez a mező mondjuk a referenced_commission_id lenne, amely egy commission.id-ra mutatna, pontosan arra amelyikhez készült az érvénytelenítés. Alapértelmezetten ez egy NULL értékű mező lenne, de ha készítenek egy storno type commissiont, akkor kereszthivatkozással megjelölném az eredeti eladásnál azt az eladást, amely érvényteleníti, az storno eladásnál meg azt az eladást amit érvénytelenít.

pl: 10-es id-val létrejön egy eladás, de a következő 11-es id-ra beszúrt eladás az ehhez tartozó storno típusú eladás. Ekkor a 10-es id-n lévő rekord referenced_commission_id mezőjének 11 lenne az értéke, a 11-es id lévő rekord referenced_commission_id mezőjének pedig 10 lenne az értéke.

- A user, client és product oszlopokba vennék fel további 2 mezőt: deleted_at és is_deleted, azért hogy soft delete-et lehessen csinálni velük.

- A client táblában az address mezőt szét lehetne bontani több részre, egy összetett adatszerkezetben jobban el lehetne tárolni a címadatokat, mondjuk ország, megye, irányítószám, város, közterület neve, közterület jellege, házszám. További egyéb mezők is felvehetők, pl emelet, épület, lépcsőház, azonban nem feltétlenül indokolt szétszedni, jelen esetben működhet egy mezőben eltárolva is.

6. Azért nincs updated_at mezője, mert nem engedik módosítani közvetlenül a jutalék tételt. Ha valamit módosítanának, akkor készítenek egy storno típusú eladást, erről a type mező árulkodik.

7. Elképzelhetőnek tartom, hogy azért mentik le mindkét táblában, mert változhatnak a jutalékok mértékei. A terméknel eltárolt jutalék az aktuális jutalékot mutatja amit a termék után kap az értékesítő ha eladja, de megeshet, hogy ez a jutalék korábban más értéket vett fel. Pl egy tv után most 20.000 ft a jutalék, de fél év múlva 25.000 lesz a jutalék egy TV eladása után. Ez jól látható a 6os feladat commission tábla adattartalmának szemléltetésénél. Dátum alapján nézve növekedtek a jutalékok. Redundáns adatnak tűnhet, de szükséges ahhoz, hogy az információ helyes és teljes legyen, így szerintem nem sért tervezési elvet. Megfelelő odafigyeléssel nem okoz anomáliát.

8. Top 3 értékesítő 2022. Q3, Q4

- legtöbb eladott termék összesen: `SELECT T1.product, COUNT(T1.id) AS salesQuantity, T2.name
from commission as T1
inner JOIN `user` AS T2 ON T2.id = T1.sales_manager_id
where T1.commission_date >= '2022-07-01'
and T1.commission_date <= '2022-12-31'
GROUP BY T1.sales_manager_id
ORDER BY salesQuantity DESC
LIMIT 3;`

- legtöbb eladott termék termékenként: `SELECT T1.product, COUNT(T1.id) AS salesQuantity, T2.name
from commission as T1
inner JOIN `user` AS T2 ON T2.id = T1.sales_manager_id
where T1.commission_date >= '2022-07-01'
and T1.commission_date <= '2022-12-31'
AND T1.product = 'TV'
GROUP BY T1.sales_manager_id
ORDER BY salesQuantity DESC
LIMIT 3;`

`SELECT T1.product, COUNT(T1.id) AS salesQuantity, T2.name
from commission as T1
inner JOIN `user` AS T2 ON T2.id = T1.sales_manager_id
where T1.commission_date >= '2022-07-01'
and T1.commission_date <= '2022-12-31'
AND T1.product = 'ironing'
GROUP BY T1.sales_manager_id
ORDER BY salesQuantity DESC
LIMIT 3;`

`SELECT T1.product, COUNT(T1.id) AS salesQuantity, T2.name
from commission as T1
inner JOIN `user` AS T2 ON T2.id = T1.sales_manager_id
where T1.commission_date >= '2022-07-01'
and T1.commission_date <= '2022-12-31'
AND T1.product = 'vacuum'
GROUP BY T1.sales_manager_id
ORDER BY salesQuantity DESC
LIMIT 3;`

- legtöbb eladott termék összesen, azok között akik legalább 10et eladtak belőlük: **SELECT**

```
T1.product, COUNT(T1.id) AS salesQuantity, T2.name
from commission as T1
inner JOIN `user` AS T2 ON T2.id = T1.sales_manager_id
where T1.commission_date >= '2022-07-01'
and T1.commission_date <= '2022-12-31'
GROUP BY T1.sales_manager_id
HAVING COUNT(*) >= 10
ORDER BY salesQuantity DESC
LIMIT 3;
```

- legtöbb eladott termék termékenként, azok között akik legalább 10et eladtak belőlük: **SELECT**

```
T1.product, COUNT(T1.id) AS salesQuantity, T2.name
from commission as T1
inner JOIN `user` AS T2 ON T2.id = T1.sales_manager_id
where T1.commission_date >= '2022-07-01'
and T1.commission_date <= '2022-12-31'
AND T1.product = 'TV'
GROUP BY T1.sales_manager_id
HAVING COUNT(*) >= 10
ORDER BY salesQuantity DESC
LIMIT 3;
```

```
SELECT T1.product, COUNT(T1.id) AS salesQuantity, T2.name
from commission as T1
inner JOIN `user` AS T2 ON T2.id = T1.sales_manager_id
where T1.commission_date >= '2022-07-01'
and T1.commission_date <= '2022-12-31'
AND T1.product = 'ironing'
GROUP BY T1.sales_manager_id
HAVING COUNT(*) >= 10
ORDER BY salesQuantity DESC
LIMIT 3;
```

```
SELECT T1.product, COUNT(T1.id) AS salesQuantity, T2.name
from commission as T1
inner JOIN `user` AS T2 ON T2.id = T1.sales_manager_id
where T1.commission_date >= '2022-07-01'
and T1.commission_date <= '2022-12-31'
AND T1.product = 'vacuum'
GROUP BY T1.sales_manager_id
HAVING COUNT(*) >= 10
ORDER BY salesQuantity DESC
LIMIT 3;
```

összes eladott termék értéke alapján: **SELECT SUM(T1.commission_amount) AS sumAmount,**
T2.name

```
from commission as T1
inner JOIN `user` AS T2 ON T2.id = T1.sales_manager_id
where T1.commission_date >= '2022-07-01'
and T1.commission_date <= '2022-12-31'
GROUP BY T1.sales_manager_id
ORDER BY sumAmount DESC
LIMIT 3;
```

9.

```
$sql = "SELECT sales_manager_id, product, SUM(commission_amount) as  
sumAmount  
FROM commission  
WHERE YEAR(commission_date) = 2022  
GROUP BY sales_manager_id, product";  
  
$stmt = $pdo->prepare($sql);  
$stmt->execute();  
$result = $stmt->fetchAll(PDO::FETCH_ASSOC);  
  
foreach ($result as $row) {  
    $bonus_amount = $row['total_commission'] * 0.05;  
    $sales_manager_id = $row['sales_manager_id'];  
    $product = $row['product'];  
  
    $sql = "INSERT INTO commission (sales_manager_id, product,  
commission_date, commission_type, commission_amount)  
VALUES (?, ?, '2022-12-24', 'premium', ?)";  
    $stmt = $pdo->prepare($sql);  
    $stmt->execute(array($sales_manager_id, $product, $bonus_amount));  
}
```

2. Nagy feladat

becslés: 10 óra, mivel nem tudom mennyi idő kell amíg a választott technológiákból elsajátítom az alapokat

eredmény: kb 4 óra volt a tudásom felhozni a megfelelő szintre, 5 óra az implementáció, a dokumentáció további 1 óra

A választáson a Symfonyra esett, még sohasem használtam, de azért választottam ezt, mivel ez volt az ajánlott keretrendszer.

A szükséges programok, technológiák: XAMMP (Apache, MySQL server), composer, node.js npm vagy yarn csomagtelepítő

Ezen a linken érhető el a forráskód: <https://github.com/fbence83/robot-fight.git>

A docs/db.sql tartalmazza a szükséges MySQL adatbázist.

A projekt root mappájában futtatni kell a következő parancsokat:

- composer install (előfordulhat, hogy szükséges a következő kapcsoló hozzá: --ignore-platform-req=ext-http)
- npm install
- npm run build
- symfony serve -d

ezek után a <http://localhost:8000> alatt elérhetővé válik a webes megoldás

az API végpont:

everest / fight


POST ▼ http://localhost:8000/api/robot/fight/?robotId=1&robotId2=3

Params ● Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

	KEY	VALUE
<input checked="" type="checkbox"/>	robotId1	1
<input checked="" type="checkbox"/>	robotId2	3
	Key	Value

Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize JSON ▼ 

```
1  {
2    "id": 1,
3    "name": "R2D2",
4    "type": "brawler",
5    "power": 100,
6    "created_at": {
7      "date": "2023-03-05 03:55:57.000000",
8      "timezone_type": 3,
9      "timezone": "Europe/Berlin"
10   },
11   "updated_at": {
12     "date": "2023-03-05 03:58:52.000000",
13     "timezone_type": 3,
14     "timezone": "Europe/Berlin"
15   },
16   "deleted_at": null,
17   "is_deleted": false
18 }
```