



eduresourceza.com

- 1.1 She needs a list of movies displaying all the details of every movie sorted by release date from most recently released to last released and then by name alphabetically.

SELECT * FROM tblMovie
ORDER BY ReleaseDate DESC, MovieName (3)

- 1.2 Display the name and budget of the movie which had the highest budget. (2)

SELECT TOP 1 * FROM tblMovie
ORDER BY BUDGET DESC

- 1.3 She discovers there was a mistake in the calculations of box office takings for every movie. Update the box office column by multiplying every value by 0,95. (3)

UPDATE tblMovie
SET Budget = Budget * 0.89, BoxOffice = BoxOffice * 0.89
WHERE Language = 'French';

- 1.4 For her next movie she needs to select a director who is older than 50 in order to improve her age equity rating. Display a list of directors who are older than 50 as of the end of the year, also display their ages. (4)

SELECT FirstName, Surname, Year(Now) – Year(DateOfBirth) FROM tblDirector
WHERE Year(Now) – Year(DateOfBirth) > 50;

- 1.5 Steven Spielberg will not be directing any movies produced by Gigi Pritzker. Remove him from the table of directors. (3)

DELETE FROM tblDirector
WHERE FirstName = 'Steven' AND Surname = 'Spielberg';

- 1.6 Each director needs a username to login to the database. Generate a username for each director by joining their surname and the first letter of their first name. For example, Joseph Russo's username would be RussoJ. Display the director ID and their username. Sample output is below. (4)

SELECT DirectorID, Surname & LEFT(FirstName, 1)
FROM tblDirector

- 1.7 A sequel to the Avengers is planned. It will be called "Avengers: Still Not Dead". The release date will be 2021/09/01, the budget is 360, running time is 167, language is English. Put in an estimate for the box office takings which is the average of every movie which has a title STARTING with Avengers. (5)

INSERT
INTO tblMovie(BoxOffice, MovieName, ReleaseDate, Budget, RunningTime, Language)
SELECT AVG(BoxOffice), 'Avengers: Still Not Dead', #2021/09/01#, 360, 167, 'English'
FROM tblMovie
WHERE MovieName LIKE 'Avengers*';

- Downloaded from eduresourceza.com
- 1.8 Some movies have not yet been linked to directors. Display a list of the MovieIDs and MovieNames which have no director information yet. (4)

```
SELECT tblMovie.MovieID, MovieName
FROM tblMovie
LEFT JOIN tblMovieDirectors ✓✓
ON tblMovie.MovieID = tblMovieDirectors.MovieID ✓
WHERE tblMovieDirectors.MovieID IS NULL ✓
```

- 1.9 A new award is announced. All directors who have directed any movie NOT filmed in any variety English are eligible. Display the name and surname of all directors who are eligible. The director's name and surname should only be displayed once. Sample output is below. (6)

```
SELECT DISTINCT ✓ FirstName, Surname
FROM tblDirector
INNER JOIN (tblMovie
INNER JOIN tblMovieDirectors ✓ ON tblMovieDirectors.MovieID = tblMovie.MovieID) ✓
ON tblMovieDirectors.DirectorID = tblDirector.DirectorID
WHERE Language NOT ✓ LIKE '*English*' ✓
```

- 1.10 Gigi Pritzker wants to see how much profit each director in the database has made. Display a list with the first name, surname, number of movies made and total profit (box office takings – budget). (6)

```
SELECT FirstName, Surname, Count(*) ✓ AS NumMovies, SUM ✓ (BoxOffice – Budget) ✓ AS
PROFIT
FROM tblMovie
INNER JOIN (tblDirector ✓
INNER JOIN tblMovieDirectors ✓ ON tblMovieDirectors.DirectorID = tblDirector.DirectorID)
ON tblMovieDirectors.MovieID = tblMovie.MovieID
GROUP BY FirstName, Surname ✓
```

//2.1 [1]

```
public class Position✓ {
```

//2.2 [2]

```
private int✓ xPos;
```

```
private✓ int yPos;
```

//2.3 [2]

```
public Position(int inXPos, int inYPos) ✓ {
```

```
    xPos = inXPos;
```

```
    yPos = inYPos; ✓
```

```
}
```

//2.4 AND 2.5 [2]

```
public int getXPos() {✓
```

```
    return xPos;
```

```
}
```

```
public int getYPos() {✓
```

```
    return yPos;
```

```
}
```

//7.1 [5]

```
public double distanceFromPos (Position pos2) ✓ {
```

```
    int x = pos2.getXPos() - xPos;
```

```
    int y = pos2.getYPos() - yPos; ✓
```

```
    double distance = Math.pow(x, 2) + Math.pow(y, 2) ✓;
```

```
    return ✓ Math.sqrt(distance); ✓
```

```
}
```

//2.6 [2]

```
@Override
```

```
public String toString() {
```

```
    return✓ "Coordinates: (" + xPos + "; " + yPos + ")";✓
```

```
}
```

```
}
```

QUESTION 3

14 MARKS

//3.1 [1]

```
public class TrainingDummy✓ {
```

//3.2 [3]

```
private String name; ✓
```

```
private Position✓ pos;
```

```
private✓ int health;
```

//3.3 [3]

```
public TrainingDummy(String inName, Position inPos, ✓ int inHealth)
```

```
{
```

```
    name = inName; ✓
```

```
    pos = inPos; ✓
```

```
    health = inHealth;
```

```
}
```

//3.4 [1]

```

public String getName() {
    return name; ✓
}

//3.5 [1]
public Position getPos() {
    return pos; ✓
}

//3.6 [1]
public int getHealth() {
    return health; ✓
}

//3.7 [2]
public void takeDamage(int inDamage) {
    health -= inDamage; ✓
    if (health < 0) { ✓
        health = 0;
    }
}

//7.2 [3]
public double distanceFromPos (Position pos2) {
    return pos.distanceFromPos (pos2) ✓;
}

//3.8 [2]
@Override
public String toString() {
    return name + " located at " + pos.toString() ✓ + " with health "
+ health; ✓
}
}

```

QUESTION 4**13 MARKS**

```

//4.1 [1]
public class Superhero ✓ extends TrainingDummy {

    //4.2 [2]
    private int damage;
    private int range; ✓ SECOND ✓ GOES TO EXTRA FIELDS OR EXTENDS

    //4.3 [3]
    public Superhero(String inName, Position inPos, int inHealth, int
inDamage, int inRange) {
        super ✓ (inName, inPos, inHealth); ✓
        this.damage = inDamage; ✓
        this.range = inRange; IF NO SUPER THIRD ✓ GOES TO EXTRA FIELDS
    }

    //4.4 [1]
    public int getDamage() {
        return damage; ✓
    }

    //4.5 [1]
    public int getRange() {

```

```

        return range;
    }
    //4.6 [3]
    public boolean isAvenger() {
        if (this.getName().contains("Avenger")) {
            return true;
        }
        return false;
    }

    //4.7 [2]
    @Override
    public String toString() {
        return super.toString() + ", damage " + damage + " and range "
+ range;
    }
    }
    }

```

QUESTION 5**22 MARKS**

//5.1 [1]

```
public class FighterManager {
```

//5.2 [4]

```

    private TrainingDummy[] dummyArr = new TrainingDummy[100];
    int dummySize = 0;
    private Superhero[] superArr = new Superhero[100];
    int superSize = 0;

```

//5.3 [12]

```

    public FighterManager() {
        try {
            Scanner scFile = new Scanner(new File("fighters.txt"));
            while (scFile.hasNextLine()) {
                Scanner scLine = new
Scanner(scFile.nextLine()).useDelimiter("#");
                String name = scLine.next();
                Position position = new Position(scLine.nextInt(),
scLine.nextInt());
                int health = scLine.nextInt();
                if (scLine.hasNext()) {
                    int damage = scLine.nextInt();
                    int range = scLine.nextInt();
                    superArr[superSize] = new Superhero(name, position,
health, damage, range);
                    superSize++;
                } else {
                    dummyArr[dummySize] = new TrainingDummy(name,
position, health);
                    dummySize++;
                }
            }
        } catch (FileNotFoundException ex) {
            System.out.println("File not found: " + ex);
        }
    }

```

```

    }
//5.4 [5]
@Override
public String toString() ✓ {
    String ret;
    ret = "TrainingDummies:\n";
    for (int i = 0; i < dummySize; i++) ✓ {
        ret += dummyArr[i].toString() + "\n"; ✓
    }
    ret += "Superheroes:\n";
    for (int i = 0; i < superSize; i++) {
        ret += superArr[i].toString() + "\n"; ✓
    }
    return ✓ret;
}
//7.3.1 [2]
public String attack(String inName) {
    String ret = "";
    int i = 0;
    boolean found = false;
    while (i < superSize && !found) ✓ {
        if (superArr[i].getName().equals(inName)) ✓ {
//7.3.2 [7]
            for (int j = 0; j < dummySize; j++) ✓ {
                if (dummyArr[j].distanceFromPos ✓
(superArr[i].getPos()) < superArr[i].getRange()) ✓ {
                    dummyArr[j].takeDamage(superArr[i].getDamage()) ✓;
                    ret += dummyArr[j].getName() + " took " +
superArr[i].getDamage() + " damage. New health is " +
dummyArr[j].getHealth() + "."; ✓
                }
            }
            found = true; ✓
        }
        i++; ✓
    }
    return ret;
}
}

```

QUESTION 6**3 MARKS**

```

//6.1 [1]
public class TutorialApp ✓ {

    public static void main(String[] args) {
//6.2 [1]
        FighterManager fm = new FighterManager(); ✓
//6.3 [1]
        System.out.println(fm); ✓
//7.4 [2]
        System.out.println(fm.attack ✓ ("Avenger Captain America")) ✓;
    }
}

```