

**מבחן במסדי נתונים**

**עמוס עזריה ומירב שקרון**

**7029210-1,3,4,5**

סמסטר ב' מועד ב' כ"ט בתמוז התשע"ט, 1.8.2019

**הנחיות כלליות:**

* משך הבחינה: 180 דקות.
* **יש לענות בגוף השאלון!** המחברת תשמש כטיוטא בלבד, מענה במחברת עלול לגרור ציון 0.
* אין להכניס שום חומר עזר.
* השימוש במחשבון **אסור**.
* בשאלות האמריקאיות רק תשובה אחת נכונה.
* בשאלות פתוחות נא לענות תשובה החלטית תחילה ואח"כ לנמק.
* בסיום הבחינה - נא למסור את השאלון ואת המחברת.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Total | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |  |
| 108 | 16 | 8 | 8 | 8 | 12 | 12 | 10 | 14 | 20 | Max points |
|  |  |  |  |  |  |  |  |  |  | Grade |

**ב ה צ ל ח ה!**

שאלה 1-SQL (20 נק')

נתונות הטבלאות הבאות לייצוג רשת בתי קולנוע:

cinema(cinema\_id, name, city)

hall(cinema\_id, hall\_num, seats)

film(film\_id, film\_name, director, genre, length)

film\_hall(show\_id, cinema\_id, hall\_num, film\_id, date, hour)

ticket(show\_id, customer\_id, num\_seats)

שימו לב: בSQL ניתן לעשות GROUP BY לפי שתי תכונות באמצעות הפרדה בפסיק (כמו שניתן לעשות בSELECT FROM וORDER BY)

1. כתבו שאילתא המציגה לכל בית קולנוע ולכל ז'אנר (genre) את מס' הפעמים שהוקרן סרט מז'אנר זה עד סוף שנת 2018 (יש להציג שם בית הקולנוע, ז'אנר ואת מס' הפעמים שהוקרן)

Select c.name, f.genre, count(fh.show\_id)

from film as j join film\_hall as fh

On(f.film\_id = fh.film\_id)

join cinema as c

on(fh.cinema\_id = c.cinema\_id)

where year(fh.date)<2019

group by (c.name, f.genre);

1. בהנחה שמחיר כרטיס הוא 35 ₪, כתבו שאילתא שתחזיר עבור כל הקרנה את שם בית הקולנוע, מס' האולם, תאריך ושעת ההקרנה ואת ההכנסה מאותה הקרנה, ובתנאי שההכנסה מאותה ההקרנה גדולה מ-100 ₪.

Select fh.show\_id, c.name, h.hall\_num,

fh.date, fh.hour, sum(t.num\_seats)\*35 as revenue

from Film as f join film\_hall as fh

on(f.film\_id = fh.film\_id)

join cinema as c on(c.cinema\_id = fh.cinema\_id)

join ticket as t on(fh.show\_id = t.show\_id)

group by fh.show\_id having revenue > 100;

לא ברור מהשאלה האם צריך להכפיל במספר המושבים או שכמה מושבים עולים אותו דבר כמו מושב אחד. בכל מקרה, הלוגיקה היא אותה לוגיקה

שאלה 2- נורמליזציה (14 נק')

תנו דוגמא לרלציה שרק אחת התכונות בה (attribute) היא non-prime (כל יתר התכונות הן prime), והרלציה ב2NF אך לא ב3NF. (רישמו את תכונות הרלציה כאותיות לדוגמא R(A,B,C) ואת רשימת התלויות שלה עם חיצים).

בידקו היטב את תשובתכם, לשאלה זו לא יינתן ניקוד חלקי.

R(A,B,C,D)

(A, C)->D

(A,B)->C

(B,C)->A

Candi-keys: (A,B),(B,C)

Non-prime: D.

D depends on (A,C) which is not a prime-key and not a subset of any candi-key.

שאלה 3- Cassandra (10 נק')

נתונות 2 הטבלאות הבאות- T1, T2.

T1 מוגדרת כך:

CREATE TABLE T1 (W INT, X INT, Y INT, Z INT, PRIMARY KEY((W, X, Y), Z));

T2 מוגדרת כך:

CREATE TABLE T2 (W INT, X INT, Y INT, Z INT, PRIMARY KEY((W, X), Y, Z));

סמנו עבור כל אחד מהשאילתות ה-CQL האם היא legal, illegal או syntax error (אם הסינטקס תקין, אבל בטבלאות הספציפיות האלו זה לא יעבוד (או יצריך ALLOW FILTERING), יש לכתוב illegal. אם הסינטקס לא תקין, כלומר, לא משנה איך היו מוגדרות הטבלאות, שאילתא כזו לא תקינה יש לכתוב syntax error)

1. SELECT \* FROM T2 WHERE W=3 and Y=-4 and X>6 \_\_\_illegal\_\_\_
2. SELECT \* FROM T2 INNER JOIN T1 ON T1.X=T2.X WHERE T2.W=45 \_\_\_syntax\_\_
3. SELECT \* FROM T1 WHERE W>54 \_illegal\_\_
4. SELECT Z FROM T1 WHERE W=145 AND X=356 AND Y>43 \_\_\_illegal\_\_
5. SELECT Z FROM T2 WHERE W=145 AND X=356 AND Y>43 \_\_\_\_legal\_\_

שאלה 4- MongoDB (12 נק')

נתון collection בשם orders בבסיס נתונים מסוג MongoDB המאחסן את רשימת הכרטיסים שנרכשו לסרטים שמוקרנים באולמות של רשת בתי הקולנוע "My Cinema".

להלן דוגמא למסמך ב-collection:

{

\_id: ObjectId("50a8240b927d5d8b5891743c"),

cinema: "My Cinema Eilat",

hall\_id: 8,

film\_name: “Aladin”,

film\_date: new Date("Oct 04, 2018"),

film\_time: “19:30”

num\_of\_tickets: 2,

amount: 78,

pay\_with: “paypal”

}

כתבו שאילתא ב-mongoDB שתחזיר את השדות הבאים- שם בית הקולנוע, סכום ששולם ומס' הכרטיסים שהוזמנו (ללא id), עבור מסמכים של הסרט Aladin בו הוזמנו 2 כרטיסים או שהסכום לתשלום (בסרט Aladin) היה 50 ₪.

db.orders.find({

film\_name:'Aladin',

$or:[

{num\_of\_tickets:2},

{amount:50}

]

},

{

cinema:1,

amount:1,

num\_of\_tickets:1,

\_id:0

});

שאלה 5- RDF + SPARQL (12 נק')

טליה ויונתן הם אנשים, כאשר טליה היא אישה ויונתן הוא גבר. יונתן אוהב סרטי מדע בדיוני. טליה אוהבת סרטים מצויירים. אלאדין, ספיידרמן ואווטאר הם סרטים, כאשר אלאדין וספיידרמן הם סרטים מצויירים, ואווטאר הוא סרט מדע בדיוני.

א. המירו את **כל המידע** המופיע בסיפור לעיל לטבלה בRDF, אך אין לכלול מידע מיותר הניתן להסקה מהטבלה: (לדוגמא, יונתן הוא גבר, וגבר זה סוג של אדם, אז אין לציין בנוסף שיונתן הוא אדם). (6 נק')

|  |  |  |
| --- | --- | --- |
| **object** | **predicate** | **subject** |
| male | gender | Jonathan |
| Sci-fi | likes | Jonathan |
| female | gender | Talia |
| cartoons | likes | Talia |
| cartoons | Genre | Aladin |
| cartoons | Genre | Spiderman |
| Sci-fi | Genre | Avatar |

ב. כתבו שאילתא בSPARQL שבהנתן טבלה מחזירה את כל האנשים יחד עם כל הסרטים שנכללים בקטגוריה שהם אוהבים. אם יש יותר מסרט אחד בקטגוריה שמישהו/י אוהב/ת, הוא/היא צריכ/ה להופיע מספר פעמים בתשובה, פעם אחת עם כל סרט. למשל בדוגמא לעיל, התשובה צריכה לכלול גם טליה אלאדין וגם טליה ספיידרמן. (6 נק')

Select ?name, ?movie where{

?name likes ?genre\_type.

?movie Genre ?genre\_type.

}

שאלה 6- JavaStreams (8 נק')

הניחו שקיימת לנו ב-java רשימה המכילה את פירוט ההזמנות שעשו אנשים שונים מאתר eBay.

מבנה אובייקט ההזמנה:

class Order {

int order\_id;

Date order\_date;

long customer\_id;

int order\_status;

List<Item> items;

{

מבנה אובייקט פריט בהזמנה:

class Item {

int item\_id;

double item\_price;

int amount;

}

כתבו קטע קוד ב-**java streams** המקבל רשימת הזמנות ומחזיר את מס' ההזמנות שבוצעו מאז תחילת השנה ושסטטוס ההזמנה גדול שווה 1 ושמחיר כל אחד מהמוצרים שהוזמנו גבוה מ- 5$.

Arrays.stream(dataset).parallel().filter(

ord->

ord.order\_date.getYear()>=2021

&& ord.order\_status > 0

&& (!ord.items.stream().anyMatch(item->item.item\_price <= 5.0))

).count();

שאלה 7- NoSQL (8 נק')

איזה מהמשפטים הבאים **אינו** מבטא הבדל **מהותי** בין Key-Value Store, לDocument Store?

1. בKey-Value Store הdatabase לא "מבין" את תוכן המסמכים המאוחסנים, אלא רק מאחסן אותם, בניגוד לDocument Store, שמכיל "הבנה" מסויימת של המסמכים.
2. בKey-Value Store יש אפשרות להגביל את הזמינות של מסמך מסויים (כלומר שהמסמך יימחק לאחר פרק זמן נתון), כמו שראינו שבRedis קיימת הפקודה Expire, בניגוד לDocument Store שלא תתכן תמיכה באופציה כזו.
3. בKey-Value Store, אין הגבלה על פורמט שמירת הנתונים בעוד בDocument Store הפורמט של המסמך מוגבל.
4. Key-Value Store, מיועד בעיקר לגישה מהירה למידע, ולא לעיבודו, בניגוד לDocument Store, שמאפשר שאילתות מורכבות לעיבוד המידע.

שאלה 8- Spark (8 נק')

איזה מהמשפטים הבאים **אינו נכון** לגבי היחס בין RDD לdataframes בSpark:

1. שימוש בRDD יהיה כמעט תמיד מהיר יותר מאשר שימוש בdataframes (כמו שקוד שנכתב בassembler כמעט תמיד מהיר יותר מקוד שנכתב בC).
2. ניתן להתייחס לdataframe כאל רמה מופשטת יותר, בה אומרים לSpark מה ברצוננו לבצע, אבל לא אומרים לה איך בדיוק לבצע את זה (בניגוד לRDD).
3. ניתן להשמש בRDD, גם כאשר הנתונים אינם מסודרים בשום מבנה, בניגוד לdataframes שמצריך מבנה כלשהו (לרוב מבנה רלציוני).
4. Spark מתרגמת את מבנה הdataframe (מאחרי הקלעים) ומבצעת את הפעולות על RDD.

שאלה 9- Linear Regression + Logistic Regression (16 נק')

תזכורת:

ב linear regression שמקבל כקלט xi עם מספר פיצ'רים מתקיים h(xi) = w1xi1+w2xi2+…+b

בהינתן m דוגמאות, פונקציית הטעות הינה:

הנגזרת של פונקציית הטעות לפי wj הינה:

ב logistic regression: h(xi) = 1/(1+exp(-(w1xi1+w2xi2+…+b)))

פונקציית הטעות בlogistic regression היא:

למדנו שהנגזרת של פונקציית הטעות בlogistic regression לפי wj היא

א. הניחו מודל של **Linear** regression, שמקבל כקלט k פיצ'רים, ומוציא מספר בודד כפלט. הניחו ריצה במשך t איטרציות (עד להתכנסות) על train דאטה בגודל m (הניחו שימוש בBatch gradient descent). הניחו test דאטה בגודל s. מה סדר הגודל של מספר הפעולות הנצרכות לצורך בניית מודל על הדאטה של הtrain והרצתו על הדאטה של הtest. השתמשו ב-O(). נמקו! (8 נק')

Top down approach:

Calculating final result: O(h(x))=O(k) (summing k+1 variables)

To reach h(x) we need to converge. To reach convergence we loop O(t) iterations.

In each iteration we calculate the gradient upon all dataset (to find the average).

So far:

Calculating gradient: and improving w,b:O(1).

This concludes: for training.

For testing its much easier since we already converged and holds w and b.

Assigning each test entry:

Conclusion:

ב. הניחו מודל של **Logistic** regression, שמקבל כקלט k פיצ'רים, ומנבא שייכות לאחת משתי קבוצות. הניחו ריצה במשך t איטרציות (עד להתכנסות) על train דאטה בגודל m (הניחו שימוש בBatch gradient descent). הניחו test דאטה בגודל s. מה סדר הגודל של מספר הפעולות הנצרכות לצורך בניית מודל והרצתו על הדאטה של הtest. השתמשו ב-O(). נמקו! (8 נק')

נזכר שמבנה האלגוריתם זהה מלבד פונקציית הניבוי/הכרעה (h) וההכרעה הסופית. נשים לב שהסיבוכייות של h הנוכחית זהה לh הקודמת כי גם כאן צריך לסכום k מכפלות של פיצרים בערכי w (הסכימה במעריך של e ).

גם ההכרעה בעלת סיבוכייות זהה כי אנו מחשבים את h(x) ומקבלים למעשה הסתברות של השתייכות לקבוצה המאופיינת כ1. ואז בד״כ מעל חצי או מתחת לחצי מכריע את הכף לסיווג. לסיכום, הסיבוכייות נשארת כמקודם