Project

Part A

תאור המסד:

בית ההשקעות "יותר תשואה" מעוניין למדל מחדש את מסד הנתונים המיושן בו נשמר המידע הנחוץ לצרכי החברה. "יותר תשואה" מספקת ללקוחותיה פלטפורמה למסחר בבורסה ולצורך כך עליה לשמור מידע אודות לקוחותיה (המשקיעים בבורסה), עובדיה, החברות הנשכרות וערכי המניות שלהן.

כל משקיע (investor) מזוהה באמצעות מספר תעודת הזהות שלו (מספר בן 9 ספרות **שלא** יכול להתחיל בספרה 0), ובנוסף נשמרים שמו, תאריך לידתו, כתובת המייל שלו ותאריך הרשמתו למערכת. "יותר תשואה" לא מאפשר למשקיעים שנולדו בשנת 2006 ואילך להירשם למערכת. כמו כן, לא ייתכנו שני משקיעים שונים המשתמשים באותה כתובת מייל. משקיע חדש המתחיל להשתמש בשירותי החברה מוגדר בשלושת החודשים הראשונים שלאחר הרשמתו כמשקיע מתחיל (beginner), ולאחר מכן הוא הופך להיות משקיע פרימיום (premium). עבור כל משקיע פרימיום יש לשמור את יעדיו הפיננסיים (שדה מילולי). במסד מתועד מידע אודות עובדי החברה. כל אחד מהעובדים הוא בהכרח גם משקיע פרימיום המשתמש בעצמו בשירותי החברה. כל משקיע מתחיל מודרך על ידי (guided by) עובד אחד בדיוק, ולכל עובד יש לפחות משקיע מתחיל אחד אותו הוא מדריך.

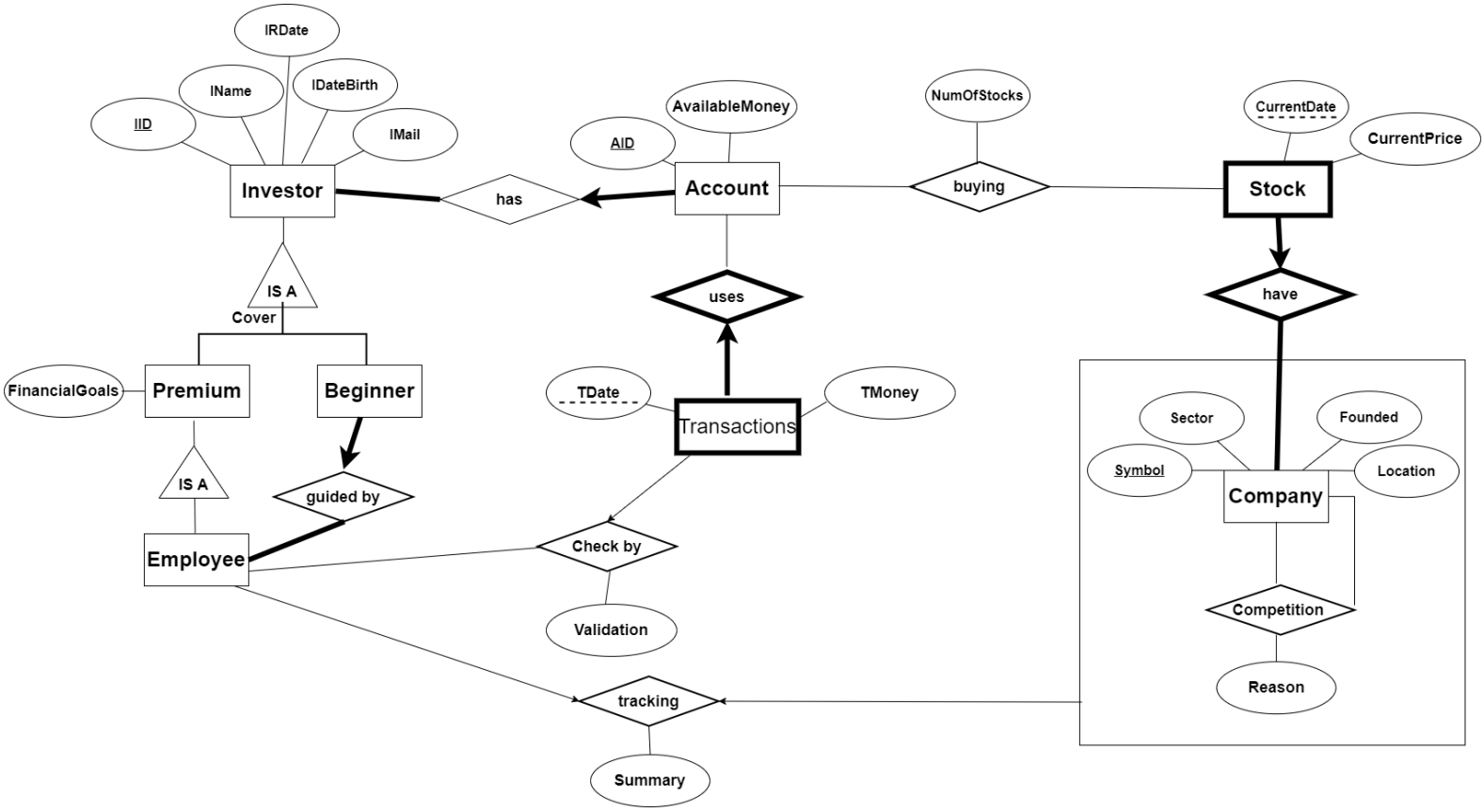
במסד נשמר מידע אודות החברות (company) שבמניותיהן ניתן לשכור. חברה מזוהה באמצעות קוד זיהוי (symbol, שדה טקסטואלי) ובנוסף נשמר התחום (sector) אליו היא משתייכת (נדל"ן, אנרגיה וכד'), שנת הקמתה (founded) והמדינה בה יושב המטה הראשי שלה (location). שתי חברות שונות עשויות להיות ביריבות אחת עם השנייה. במקרים אלו יש לתעד את הסיבה ליריבות. עובד בחברה עשוי לעקוב אחר יריבות בין שתי חברות ולכתוב דוח סיכום אודות היריבות ביניהן. לא ייתכן כי שני עובדים שונים או יותר עוקבים אחר אותה היריבות, ולא ייתכן כי עובד יחיד עוקב אחר יותר מיריבות אחת.

לכל אחד מהמשקיעים יש לפחות חשבון מסחר אחד. חשבון מסחר מזוהה באמצעות מחרוזת בת 10 תווים בדיוק, ושייך למשקיע אחד בדיוק. משקיע יכול להעביר כסף מחשבון הבנק הפרטי שלו לחשבון מסחר לצורך רכישת מניות. כל העברה (transaction) מזוהה באמצעות תאריך ומזהה חשבון המסחר אליו בוצעה ההעברה, ובנוסף נשמר סכום הכסף שהועבר. לא ניתן להעביר סכום כסף הקטן מ-1,000 דולר. לעיתים העברה כלשהי נראית למערכת חשודה, ובמקרים אלו היא מועברת לבדיקה של עובד החברה. יש לתעד את החלטת עובד החברה בנוגע לחוקיות ההעברה (משתנה בוליאני).

מניה של חברה (stock) מזוהה באמצעות תאריך והחברה אליה היא מתייחסת. כמו כן, יש לשמור את ערך המניה באותו יום (ניתן להניח לשם פשטות כי ערך המניה של חברה כלשהי לא משתנה במהלך היום). עבור כל אחת מהחברות המופיעות במסד קיים תיעוד של לפחות מניה אחת ביום כלשהו. פלטפורמת המסחר של "יותר תשואה" מאפשרת למשקיעים לבצע קניית מניות (buying) דרך חשבון המסחר שלהם. קניית מניות מוגדרת כרכישת כמות כלשהי של מניות שבוצעה מחשבון בנק לפי מחיר המניה ביום הקניה. עבור כל קניה יש לתעד את פרטי המניה וחשבון הבנק, בנוסף לכמות המניות שנקנו. לא ניתן לבצע יותר מקניה אחת של מניות השייכות לחברה כלשהי דרך אותו חשבון מסחר ביום יחיד (כן ניתן לבצע קניות מרובות באותו היום דרך אותו החשבון אם מדובר במניות של חברות שונות). עבור כל חשבון מסחר יש לתעד את סכום הכסף הזמין בו לרכישת מניות (סכום זה הוא סך הסכומים שהועברו לחשבון פחות הסכום שהושקע בקניית מניות).

מידול:

1) (40 נקי) צרו דיאגרמת ER של המערכת. ציינו את כל ההנחות עליהן אתם מתבססים. עבור דרישות שלא ניתנות למידול באמצעות הדיאגרמה, יש לציין מילולית מהי הדרישה ולהציע דרך התמודדות עימה (למשל ברמת ה-DDL). ניתן להיעזר לצורך יצירת התרשים באתר https://www.draw.io/. יש לבנות את הדיאגרמה באנגלית כך ששמות הישויות והתכונות יתאמו לאלו שבהם תשתמשו בשאלה 2.



הערות:

1. מכיוון שאחרי שלושה חודשים משקיע מתחיל חייב לעבור להיות משקיע פרמיום, אז כל משקיע בכל רגע מסוים הוא או משקיע מתחיל או משקיע פרמיום, ואין שום אופציה אחרת.
2. מכיוון שנתון שכל עובד הוא בהכרח משקיע פרמיום, אז יש לו את אותם שדות שיש למשקיע פרמיום, ולכן נניח שישות של עובד הוא "סוג של" משקיע פרמיום.
3. מבחינת הERD לא ניתן למנוע תיעוד כפול של יריבות[יוכלו להיות 2 רשומות (חברה1,חברה2) וגם (חברה2,חברה1)].
*אפשר למדל זאת בDDL שאי אפשר שיהיו שתי רשומות של יריבות על ידי בדיקה לקסיקוגרפית של סטרינג אחד גדול מהשני באמצעות אילוץ Check, אולם בחרנו לא לעשות זאת בDDL וזאת מכיוון שאילוץ Check שכזה יגרום לכך שלא נוכל להזין למסד נתונים צמד חברות, חברה 1, חברה 2, כך שהSymbol של חברה 2 יותר גדול(מבחינה לקסיקוגרפית) מזה של חברה 1.
4. אי-אפשר לוודא שחברה לא נמצאת ביריבות עם עצמה ברמת הERD, נוכל לפתור זאת ברמת הDDL באמצעות אילוץ Check.
5. בERD לא ניתן להגביל את הרשומות בשדה IDateBirth רק לרשומות עם תאריך לידה שלכל היותר 2005,
נוכל לפתור זאת ברמת הDDL באמצעות אילוץ Check.
6. בERD לא ניתן לאלץ שלשני משקיעים שונים אין את אותו מייל, נוכל לפתור זאת ברמת הDDL באמצעות אילוץ ייחודיות על שדה IMaiil.
7. בERD ובDDL אין אפשרות "להמיר" באופן דינמי משתמש שהתחיל מBeginner למשתמש Premium.
בSQL ניתן למחוק את המשקיע על ידי שאילתה שתמחק אותו מBeginner, ותשים אותו בתור משקיע Premium.
8. בERD ובDDL לא ניתן לעדכן את שדה AvailableMoney לאחר כל העברה של כסף לחשבון המסחר,
אולם בSQL ניתן לבצע שאילתה לעדכון יתרה אחרי כל העברה לאחר כל העברה לחשבון מסחר.
9. לא ניתן בERD ובDDL למדל את העדכון ביתרת הכסף שנשאר בחשבון המסחר לאחר כל רכישה.
בSQL נעשה עדכון של היתרה באמצעות SQL באמצעות הפרש של היתרה לפני הרכישה בחשבון פחות הסכום שבו קנה מניות.
10. בDDL לא ניתן לאכוף את אילוץ ההשתתפות של משקיע בקשר עם חשבון מסחר, נוכל לפתור זאת ברמת בSQL באמצעות שאילתה על הרלציות של Investor ו-Account.
11. לא ניתן לבטא ברמת ה-ERD את מגבלת 1000 הדולרים להעברה.

2) (30 נק') צרו סקריפט DDL עם פקודות Create Table מתאימות ליצירת מסד הנתונים. השתמשו בטיפוסים מתאימים לפי הערכים הנשמרים בכל שדה. בשדות בהם לדעתכם יש להשתמש בשדה מסוג מחרוזת אתם יכולים להניח כי לא ייתכן ערך שאורכו גדול מ-40 תווים, וכי בכל פעם שיש בסיפור התייחסות לתאריך הכוונה היא לפורמט DATE (YYYY-MM-DD). זכרו – יש חשיבות לסדר יצירת הטבלאות! הקפידו עליו. כמו כן, יש לוודא כי שמות הטבלאות והשדות תואמים לאלו שבהם השתמשתם בתרשים (שאלה 1).

```sql
CREATE TABLE Investor (
    IID INT CHECK(IID >= 100000000 AND IID <= 999999999) PRIMARY KEY,
    IName VARCHAR(40),
    IMail VARCHAR(40) UNIQUE,
    IRDate DATE,
    IDateBirth DATE CHECK (YEAR(IDateBirth) < 2006)
);
/*
In DDL it is not possible to enforce the participation constraint of an investor in connection with a trading account,
we can solve this at the SQL level by querying the relations of Investor and Account.
*/

CREATE TABLE Premium (
    IID INT PRIMARY KEY,
    FinancialGoals VARCHAR(40),
    FOREIGN KEY(IID) REFERENCES Investor(IID) ON DELETE CASCADE
);

CREATE TABLE Employee (
    IID INT PRIMARY KEY,
    FOREIGN KEY(IID) REFERENCES Premium(IID)
);

CREATE TABLE Beginner (
    IID INT PRIMARY KEY,
    IIDEmployee INT NOT NULL,
    FOREIGN KEY(IID) REFERENCES Investor(IID),
    FOREIGN KEY(IIDEmployee) REFERENCES Employee(IID) ON DELETE CASCADE
);
/*
In ERD and DDL there is no possibility to dynamically "convert" a user who started from Beginner to a Premium user.
In SQL, an update will be made to delete it from Beginner, and put it in Premium
*/

/*
In ERD and DDL it is not possible to update the AvailableMoney field after each transfer of money to the trading
account, however in SQL you can make a query to update the balance after each transfer,
after each transfer to a trading account.
*/
```

```sql

CREATE TABLE Account (
    AID CHAR(10) PRIMARY KEY,
    IID INT NOT NULL,
    AvailableMoney FLOAT,
    FOREIGN KEY(IID) REFERENCES Investor(IID) ON DELETE CASCADE
);
/*
It is not possible in ERD and DDL to update the balance of the money left in the trading account after each purchase.
In SQL we can update the balance using a query so that the result will be the difference between the balance before
the purchase in the account and the amount with which he bought stocks.
*/

CREATE TABLE Transactions (
    AID CHAR(10) NOT NULL,
    TDate DATE NOT NULL,
    TMoney FLOAT CHECK(TMoney >= 1000),
    PRIMARY KEY(AID, TDate),
    FOREIGN KEY (AID) REFERENCES Account(AID) ON DELETE CASCADE
);

```

```sql
CREATE TABLE CheckBy (
    IID INT,
    TDate DATE,
    AID CHAR(10),
    VaidaMoney BIT,
    PRIMARY KEY(IID, TDate, AID),
    UNIQUE (TDate, AID),
    FOREIGN KEY(IID) REFERENCES Employee(IID),
    FOREIGN KEY(AID, TDate) REFERENCES Transactions(AID, TDate)
);

CREATE TABLE Company (
    Symbol VARCHAR(40) PRIMARY KEY,
    Sector VARCHAR(40),
    Founded INT,
    Location VARCHAR(40)
);
/*
As far as the ERD is concerned, it is not possible to prevent double documentation of rivalries
[there could be 2 records (company 1, company 2) and (company 2, company 1)].
*It is possible to model in DDL that it is impossible to have two rival records by lexicographically
checking one string that is greater than the other using a Check constraint,
but we chose not to do this in DDL and this is because such a Check constraint would
mean that we cannot enter a pair of companies into the database,
so that a company 2 "larger" in terms of lexicography.
*/

CREATE TABLE Competition (
    Company1 VARCHAR(40),
    Company2 VARCHAR(40),
    Reason VARCHAR(40),
    PRIMARY KEY(Company1, Company2) ,
    FOREIGN KEY(Company1) REFERENCES Company(Symbol) ON DELETE CASCADE,
    FOREIGN KEY(Company2) REFERENCES Company(Symbol) /*ON DELETE CASCADE*/,
    CHECK(Company1 != Company2)
);
```

```sql

CREATE TABLE TrackingBy (
    IID INT UNIQUE NOT NULL,
    Company1 VARCHAR(40),
    Company2 VARCHAR(40),
    Summary VARCHAR(40),
    UNIQUE (Company1, Company2),
    PRIMARY KEY(Company1, Company2),
    FOREIGN KEY(IID) REFERENCES Employee(IID),
    FOREIGN KEY(Company1, Company2) REFERENCES Competition(Company1, Company2) ON DELETE CASCADE,
);

CREATE TABLE Stock (
    Symbol VARCHAR(40) NOT NULL,
    CurrentDate DATE NOT NULL,
    CurrentPrice FLOAT,
    PRIMARY KEY(Symbol, CurrentDate),
    FOREIGN KEY(Symbol) REFERENCES Company(Symbol) ON DELETE CASCADE
);

CREATE TABLE Buying (
    AID CHAR(10),
    Symbol VARCHAR(40),
    CurrentDate DATE,
    NumOfStocks INT CHECK(NumOfStocks >= 0),
    PRIMARY KEY(AID, Symbol, CurrentDate),
    FOREIGN KEY(AID) REFERENCES Account(AID),
    FOREIGN KEY(Symbol, CurrentDate) REFERENCES Stock(Symbol, CurrentDate)
);
```

שאילתות (Views):

בעולם מסדי הנתונים view הינו טבלה וירטואלית הנוצרת כתוצאה מהפעלת שאילתה על טבלאות רגילות במסד הנתונים. מכיוון ש-view דומה לטבלאות הרגילות במסד הנתונים בכך שגם הוא מורכב משורות ועמודות, ניתן לשלוף ממנו מידע ולעדכן אותו בדיוק כמו טבלה רגילה. במסד הנתונים, view מוגדר על ידי שאילתת SQL. כשהמידע בטבלאות עליהן ה-view בנוי משתנה, המידע ב-view משתנה אף הוא בהתאם. נדגים את יתרונות ה-view ואופן הגדרתו בעזרת הדוגמא הבאה:

נניח ובמסד הנתונים שלנו קיימת טבלה בשם Order Details המוגדרת כדלקמן:

OrderDetails: (<u>OrderNumber</u>, ProductNum, QuantityOrdered, PriceEach)

כאשר:

- OrderNumber: מספר הזמנה
- ProductNum: מק"ט המוצר שהוזמן
- QuantityOrdered: כמות שהוזמנה מאותו מוצר
- PriceEach: מחיר ליחידה

כעת, נוכל לבנות view מעל טבלה זו בו יוצג לכל מוצר סכום תשלומי כלל ההזמנות עבורו:

```
CREATE VIEW SalesPerProduct
AS
SELECT ProductNum, SUM (QuantityOrdered * PriceEach) as TotalPrice
FROM OrderDetails
GROUP by ProductNum
```

יצרנו טבלה וירטואלית בשם SalesPerProduct, ובכל פעם שנרצה לדעת מהו סך ההכנסות ממוצר מסוים נוכל להריץ את השאילתא הבאה (עבור מזהה המוצר המתאים, למשל 102 בדוגמה המופיעה כאן):

```
SELECT TotalPrice
FROM SalesPerProduct
WHERE ProductNum = 102
```

כדי להגדיר view נשתמש בפקודה CREATE VIEW ואחריה נרשום את שם טבלת ה-view (הטבלה הוירטואלית).

לאחר מכן נקבע איך view זה צריך להיראות באמצעות שאילתת SQL רגילה המופיעה לאחר האופרטור AS.

* מחיקת view מתבצעת בדומה למחיקת טבלה:

```
DROP VIEW viewname;
```

כאשר viewname הוא שם ה-view שברצוננו למחוק.

בשתי השאלות הבאות עליכם להשתמש אך ורק בשלוש הרלציות המוגדרות כדלקמן (שימו לב כי הרלציות הללו לא בהכרח תואמות בדיוק לאלו שמוגדרות בסיפור שניתן עבור שאלות 1 ו-2):

Company (Symbol, Sector, Founded, Location)

Stock (tDate, Symbol, Price)

Buying (ID, tDate, Symbol, BQuantity)

עבור כל אחת מהרלציות, ה-primary key מסומן באמצעות קו תחתון.

Company– רלציה הכוללת מידע אודות חברות.

- Symbol – מזהה החברה.
- Sector – הסקטור אליו משתייכת החברה.
- Founded- השנה בה הוקמה החברה.
- Location- המדינה בה יושב המטה הראשי של החברה.

Stock– רלציה הכוללת מידע אודות מניות של חברות.

- tDate – תאריך.
- Symbol – מזהה החברה.
- Price- ערך המניה של החברה בתאריך הנתון.

Buying– רלציה הכוללת תיעודי קניית מניות על ידי משקיע.

- ID – מספר זהות של משקיע.
- tDate – תאריך.
- Symbol – מזהה החברה.
- BQuantity – כמות המניות של החברה שרכש המשקיע בקניה זו.

3) (15 נק'):

"משקיע פעיל" הוא משקיע שרכש ב**כל** אחד מימי המסחר המתועדים במסד מניות של **לפחות** שתי חברות שונות.

עבור כל משקיע פעיל החזירו את הפרטים הבאים:

- מספר פעולות הקניה שביצע המשקיע.
- הסכום הכולל שהשקיע המשקיע בקניית מניות (עם דיוק של 3 ספרות אחרי הנקודה).
- הסקטור שמספר פעולות קניית המניות של חברות המשתייכות אליו הוא הגבוה ביותר עבור אותו משקיע. אם קיים שוויון בין שני סקטורים או יותר יש להחזיר את הסקטור הקטן יותר בהשוואה לקסיקוגרפית (מוקדם יותר ב-ABC).

יש להחזיר את התוצאה ממוינת בסדר יורד לפי מספר פעולות הקניה שביצע המשקיע (שבירת שוויון לפי שם הסקטור בהשוואה לקסיקוגרפית בסדר עולה, כפי שמפורט לעיל).

הערה: האופרטור ROUND(x, 3) מעגל את הערך של x לכדי דיוק של 3 ספרות אחרי הנקודה.

עבור כל אחת משתי השאילתות **מותר להשתמש ב-6 שאילתות VIEWS לכל היותר** (בנוסף לשאילתה המרכזית המחזירה את התשובה הסופית). כמובן שניתן להשתמש גם בפחות.

```
    "Q3":
        [
        """
CREATE VIEW ActiveInvestor
AS
SELECT B.ID
FROM Buying B, Buying B2
WHERE (B.ID = B2.ID) AND (B.tDate = B2.tDate) AND (B.Symbol <> B2.Symbol)
GROUP BY B.ID
HAVING (SELECT COUNT(DISTINCT S.tDate) FROM Stock S) = (COUNT(DISTINCT B.tDate))
        """,
        """
CREATE VIEW CountBuyings
AS
SELECT ActiveInvestor.ID ,COUNT(ActiveInvestor.ID) AS CountBuyings
FROM ActiveInvestor, Buying
WHERE ActiveInvestor.ID = Buying.ID
GROUP BY ActiveInvestor.ID
        """,
        """
CREATE VIEW SumOfInvestment
AS
SELECT AI.ID, ROUND(SUM(B.BQuantity * S.Price), 3) AS SumOfInvestment
FROM ActiveInvestor AI, Buying B, Stock S
WHERE AI.ID = B.ID AND B.Symbol = S.Symbol AND B.tDate = S.tDate
GROUP BY AI.ID
        """,
        """
CREATE VIEW ActionsPerSector
AS
SELECT AI.ID, C.Sector, COUNT(C.Sector) AS NumOfActionsPerSector
FROM ActiveInvestor AI, Buying B, Company C
WHERE AI.ID = B.ID AND B.Symbol = C.Symbol
GROUP BY AI.ID, C.Sector
        """
```

```
        """
CREATE VIEW MaxSector
AS
SELECT APS.ID, MIN(APS.Sector) AS Sector
FROM ActionsPerSector APS
WHERE APS.NumOfActionsPerSector = (SELECT MAX(NumOfActionsPerSector)
                                   FROM ActionsPerSector APS2
                                   WHERE APS2.ID = APS.ID)
GROUP BY APS.ID
        """
```

```
        """
SELECT ActiveInvestor.ID AS ID,
       CountBuyings.CountBuyings AS Actions,
       SumOfInvestment.SumOfInvestment AS TotalSum,
       MaxSector.Sector AS Sector
FROM ActiveInvestor, CountBuyings, SumOfInvestment, MaxSector
WHERE ActiveInvestor.ID = CountBuyings.ID AND
      CountBuyings.ID = SumOfInvestment.ID AND
      SumOfInvestment.ID = MaxSector.ID
ORDER BY CountBuyings.CountBuyings DESC
        """,
```

4) (15 נק'):

"א גרוייסע מציאה" מוגדרת כחברה המקיימת את שני התנאים הבאים:

- במסד מתועדת רק אירוע קנית מניות יחיד של אותה החברה.
- ביום המסחר העוקב (השדה tDate ברלציה Stock) לאירוע קניית מניות החברה ערכה של מנית החברה עלה **ביותר** מ-2%. על כן, התנאי לא יכול להתרחש עבור קניה שהתרחשה ביום המסחר האחרון המתועד במסד.

שימו לב- ייתכן כי קיים הפרש של יותר מיום אחד בין יום מסחר ליום המסחר העוקב שלו. למשל, יום המסחר העוקב ל-23.12.21 הוא 27.12.21.

"משקיע חד" הוא משקיע שקנה מניות של חברה המוגדרת כ"א גרוייסע מציאה".

עבור כל משקיע חד יש להחזיר את מספר פעולות קניית המניות שביצע המשקיע עבור חברות שנוסדו לפני שנת 2000 ומיקומן הוא ב-California. התוצאה לא צריכה להיות ממוינת.

```
    "Q4":
        [
            """
CREATE VIEW HasOnlyOne
AS
SELECT S.Symbol, S.tDate, S.Price
FROM Stock S, (SELECT B.Symbol, COUNT(*) AS NumOfBuyingStocks
               FROM Stock S, Buying B
               WHERE S.Symbol = B.Symbol AND S.tDate = B.tDate
               GROUP BY B.Symbol
               HAVING COUNT(*) = 1) NOBS
WHERE S.Symbol = NOBS.Symbol
            """,
            """
CREATE VIEW HigherOfBuyingDate
AS
SELECT HO01.Symbol, HO01.tDate AS AfterDate, HO01.Price AS AfterPrice,
       HO02.tDate AS BeforeDate, HO02.Price AS BeforePrice
FROM HasOnlyOne HO01, HasOnlyOne HO02, Buying B
WHERE HO01.Symbol = HO02.Symbol AND HO02.Symbol = B.Symbol AND
      HO01.tDate > B.tDate AND B.tDate = HO02.tDate AND HO01.Price > 1.02 * HO02.Price
            """,
            """
CREATE VIEW TradingDaysStock
AS
SELECT S.tDate AS CurrentTDay, MIN(S2.tDate) AS NextTDay
FROM Stock S, Stock S2
WHERE S2.tDate > S.tDate
GROUP BY S.tDate
            """,
            """
CREATE VIEW AfterBuyingDate
AS
SELECT DISTINCT HOBD.Symbol, MIN(AfterDate) AS AfterDate, MIN(BeforeDate) AS BeforeDate
FROM HigherOfBuyingDate HOBD, Buying B, Stock S, TradingDaysStock TDS
WHERE HOBD.Symbol = B.Symbol AND B.Symbol = S.Symbol AND
      HOBD.BeforeDate = B.tDate AND B.tDate < S.tDate AND TDS.CurrentTDay = B.tDate AND
      TDS.NextTDay = S.tDate AND S.Price > 1.02 * HOBD.BeforePrice
GROUP BY HOBD.Symbol;
            """,
```

```
"""
CREATE VIEW SharpInvestor
AS
SELECT DISTINCT B.ID, ABD.Symbol, AfterDate, BeforeDate
FROM Buying B, AfterBuyingDate ABD
WHERE B.Symbol = ABD.Symbol AND B.tDate = ABD.BeforeDate
"""
```

```
"""
SELECT SharpInvestor.ID ,COUNT(*) AS Actions
FROM SharpInvestor, Buying, Company
WHERE SharpInvestor.ID = Buying.ID AND Buying.Symbol = Company.Symbol AND
      Company.Founded < 2000 AND Company.Location = 'California'
GROUP BY SharpInvestor.ID
"""
```