**Information Systems and Databases**

**Final Assignment**

**Students:**

Muhammad Taha 204720932

Rana Dakwar 204603211

**Amazon.com Inc.**

1. תארו אותה בפסקה או שתיים, התייחסו לתחומים בהם פועלת, אופן פריסתה והתמקדו בצורך שעליו היא עונה ללקוחותיה.

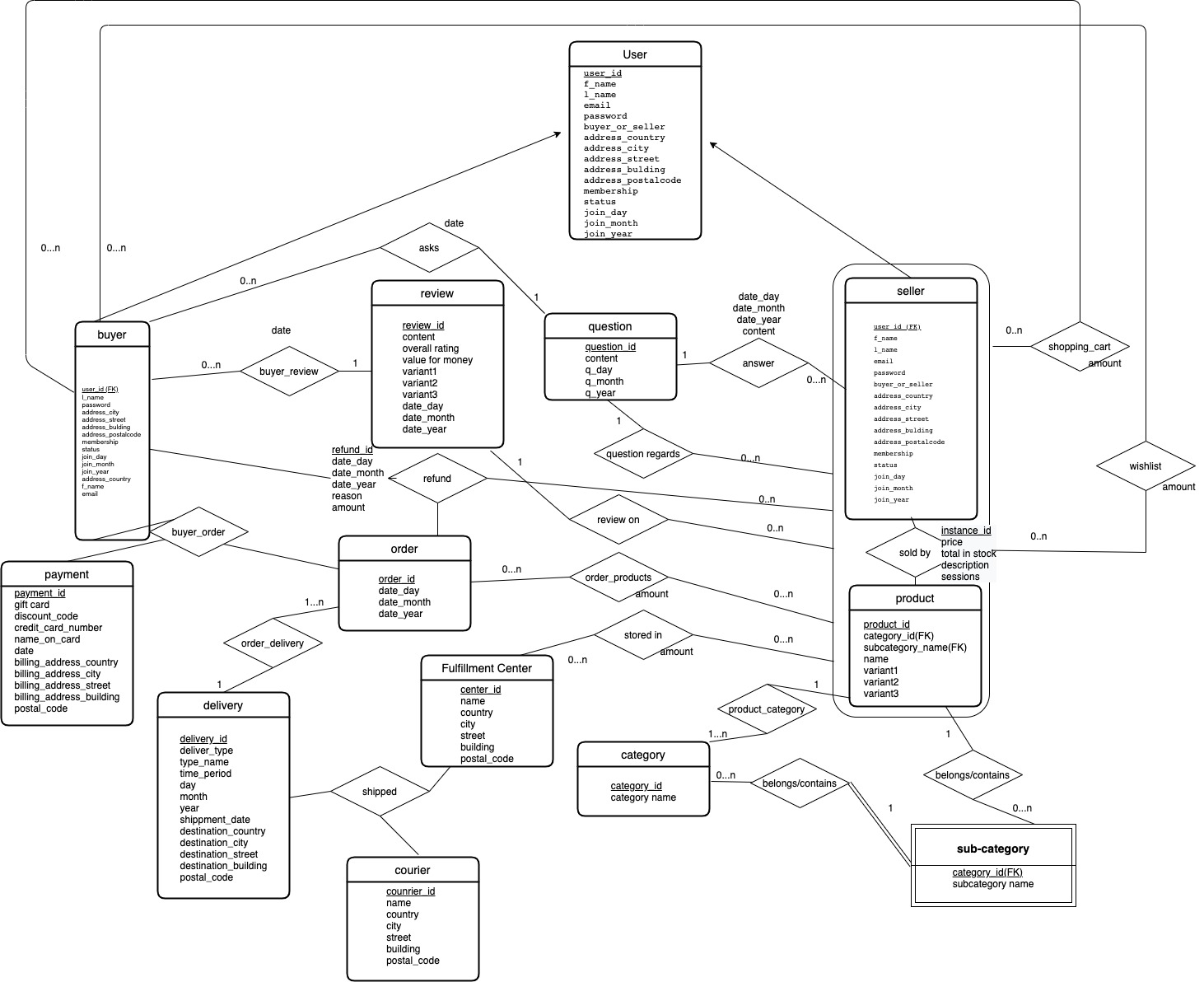
אמזון היא חברה אמריקאית למסחר אלקטרוני, שניתן לקנות באתר שלה מגוון רחב מאוד של מוצרים. נכון ל-2019, אמזון הוא האתר ה-14 הנצפה ביותר באינטרנט, ועל פי חברת המחקר EMarketer, נתח השוק של אמזון עומד על 38% מסך המסחר המקוון בארצות הברית.

הפלטפורמה של אמזון הפכה למעשה למערכת לוגיסטית-דיגיטלית שבה מוכרים שונים, שאינם אמזון, פותחים בה חנויות דיגיטליות ומציעים את מרכולתם. עם זאת, המשיכה אמזון למכור בתוך פלטפורמת המסחר שיצרה גם כחנות. נכון לשנת 2019, כ-53% מסחורות שנמכרות באמזון, נמכרות על ידי מוכרים חיצוניים. בנוסף לכך, החברה הציעה לסוחרים לאחסן את הסחורה, ולטפל באריזה ובשליחה, ונכון ל-2013 החזיקה החברה קרוב ל-100 מרכזי הפצה ברחבי העולם.

* בעבודה זו נרצה לנתח את מערך ההזמנות והקמעונאות הוירטואלית של חברת אמזון, וזאת במסגרת ובהיקף הנדרש והסביר. נדגיש כי בעבודה זו לא נעסוק ב-AWS או בתחום פעילות אחר של חברת אמזון.
* חלק מהסקריפטים של SQL נכתבו על מערכת הפעלה ווינדוז וחלק אחר במערכת הפעלה mac. הסקריפטים שצירפנו לעבודה עבדו באופן תקין.

2. צרו עבור חברה זאת תרשים ישויות-קשרים (ERD), שיתמקד בצורך עליו חברה עונה, ושיכיל לכל הפחות 10 ישויות. יש להשתמש באנגלית בסעיף זה. [24 נק']

**אפשר למצוא את ה-erd בקישור** [**כאן**](https://drive.google.com/file/d/18q-AtiX3G6WhZteLv-wjir-BJFV4Dw5t/view?usp=sharing)**.**



הערות:

* לכאורה, בקשר sold\_by מיותר להשתמש במפתח instance\_id מכיוון שניתן לזהות כל שדה ע"י המפתחות product\_id ו- seller\_id ביחד. אולם, עם יצירת הטבלאות נתקלנו בשגיאות ב- reference לטבלת sold\_by ע"י שני המפתחות האלה ביחד. לכן, החלטנו להוסיף מפתח ייחודי זה המספק לנו גם נוחות ויעילות.
* כפי שמתואר בתרשים, ישנה טבלת מוצרים וטבלה המקשרת בין המוצרים לבין המוכרים. לאורך כל העבודה בהמשך, הקנייה מתבצעת מתוך אותה טבלה במקשרת בין מוצרים ומוכרים (sold\_by) זאת מכיוון שכל קונה קונה מוצר ממוכר ולא יכול להיות מצב שהוא קונה מוצר ללא מוכר. נתון זה מחייב התייחסות לקשר בין המוצרים למוכרים כיישות הקיימת בפני עצמה. מאחר ומדובר בקשר הסגרנו אותו במלבן הגדול (כפי שלמדנו בהרצאות).
* כפי שניתן לראות בתרשים, ישנם לא מעט קשרים של יחיד לרבים. קשרים אלה חסכו לנו יצירת טבלאות נוספות מכיוון שביצירת טבלאות במקרים אלה, כפי שלמדנו בהרצאות, לוקחים את המפתח של טבלת היחיד ושותלים אותו בטבלת הרבים. דגומאות: בטבלה order שתלנו את המפתחות של buyer ו- payment ; בטבלה questions שתלנו את המפתחות של sellers וגם את התכונות של הקשר answer. כך גם ב-reviews ו- subcategories.
* למרות שהקונים מקושרים ב- refunds לא ראינו צורך בהמשך להכניס עמודה של הקונים בתוך טבלת ה- refunds זאת מכיוון שניתן לחזור לזהות הקונה ע"י טבלאות נוספות כמו orders. לכן מספיק שטבלה יש ביטוי לזהות ההזמנה.
* הערות נוספות:

1. התשלומים שוכנים ביישות נפרדת (payments), מה שמאפשר בין היתר ביטוי לכך שפרטי המשלם לא בהכרח זהים לפרטי הקונה.
2. כל הזמנה קשורה באחת או יותר delivery . כל delivery (בו מתואר סוג ותאריך שיגור) מקושר למבצע ההובלה ובמחסן בו נמצאים הפריטים.
3. כל מוצר יכול להיות מאוחסן ביותר ממחסן אחד.

3. צרו סכמה טבלאית עבור התרשים שיצרתם בסעיף הקודם, וודאו שהסכמה מציגה מפתחות ראשיים וזרים. יש להשתמש באנגלית בסעיף זה. [8 נק']

1. **users**(user\_id, f\_name, l\_name, email, password, buyer\_or\_seller, address\_country, address\_city, address\_street, address\_bulding, address\_postalcode, membership, status, join\_day, join\_month, join\_year)
2. **buyers**(user\_id(FK), f\_name, l\_name, email, password, address\_country, address\_city, address\_street, address\_no, address\_postalcode, membership, status, join\_day, join\_month, join\_year)
3. **sellers**(user\_id(FK), f\_name, l\_name, email, password, company name, company id, address\_country, address\_city, address\_street, address\_no, address\_postalcode, membership, status, join\_day, join\_month, join\_year)
4. **categories**(category\_id, name)
5. **sub\_categories**(category\_id(FK) ,name)
6. **products**(product\_id, name, category\_id(FK), subcategory\_name(FK), variant1, variant2, variant3)
7. **sold\_by**(instance\_id ,product\_id(FK), seller\_id(FK), price, total\_in\_stock, description, sessions)
8. **payments**(payment\_id, gift\_card, discount\_code, credit\_card\_num, name\_on\_card, billing\_address\_country, billing\_address\_city, billing\_address\_street, billing\_address\_no, billing\_address\_postal)
9. **orders**(order\_id, buyer\_id(FK), payment\_id(FK), date\_day, date\_month, day\_year)
10. **order\_products**(order\_id(FK), instance\_id(FK), amount)
11. **delivery**(delivery\_id, order\_id(FK), delivery\_type, time\_period, shipment\_date\_day, shipment\_date\_month, shipment\_date\_year, destination\_address\_country, destination\_address\_city, destination\_address\_street, destination\_address\_number, destination\_address\_postalcode)
12. **couriers**(courier\_id, name, address\_country, address\_city, address\_st, address\_no, address\_postalcode)
13. **fulfillment\_centers**(center\_id, name, address\_country, address\_city, address\_st, address\_no, address\_postalcode)
14. **shipped**(delivery\_id(FK), center\_id(FK), courier\_id(FK))
15. **stored\_in**(instance(FK), center\_id(FK), amount)
16. **refunds**(refund\_id ,order\_id(FK), instance\_id(FK), amount, reason, day, month, year)
17. **reviews**(review\_id, buyer\_id(FK), instance\_id(FK), content, overall\_rating, value\_for\_money, variant1, variant2, variant3, day, month, year)
18. **questions**(question\_id, buyer\_id(FK), instance\_id(FK), content, q\_day, q\_month, q\_year, answer\_content, a\_day, a\_month, a\_year)
19. **shopping\_carts**(buyer\_id(FK), instance\_id(FK), amount)
20. **wishlists**(buyer\_id(FK), instance\_id(FK), amount)

4. צרו את הטבלאות על-פי הסכימה שיצרתם בסעיף הקודם, וקלטו מספר רשומות לכל טבלה. הציגו את התחביר שבאמצעותו עניתם על סעיף זה. [10 נק'] שימו לב - אופן בדיקת הסעיף יהיה באמצעות העתקת התחביר שלכם as-is ל-MySQL והרצתו.

Create database amazon;

Use amazon;

Create table users(

User\_id char(9),

F\_name varchar(20),

L\_name varchar(20),

Email varchar(60),

Password varchar(30),

Buyer\_or\_seller char(7),

Country varchar(20),

City varchar(20),

Street varchar(20),

Building varchar(20),

Postalcode varchar(10),

Membership varchar(20),

Status varchar(15),

Join\_day int,

Join\_month int,

Join\_year int,

Primary key (user\_id)

);

Insert into users values

(“000000000”, “mohammad”, “taha”, “[mtaha220@gmail.com](mailto:mtaha220@gmail.com)”, “1234567”, “seller”, “czech republic”, “prague”, “richard”, “23”, “2018500”, “PremiumSeller”, “active”, 1, 12, 1998),

(“111111111”, “rana”, “dakwar”, “[rana.dakwar@gmail.com](mailto:rana.dakwar@gmail.com)”, “7654321”, “seller”, “spain”, “barcelona”, “saraya”, “87”, “789889”, “BasicSeller”, “active”, 2, 3, 2005),

(“222222222”, “michael”, “scofield”, “[michaels@yahoo.com](mailto:michaels@yahoo.com)”, “breaktheprison1”, “buyer”, “kenya”, “nairobi”, “lordson”, “43”, “14134545”, “R-Buyer”, “not active”, 4, 10, 2012),

(“333333333”, “sherlock”, “holmes”, “[office@holmes.com](mailto:office@holmes.com)”, “youdiebeforeyouknowit”, “buyer”, “england”, “london”, “alhambra”, “431”, “239843”, “P-Buyer”, “active”, 12, 6, 2018)

;

Create table buyers(

user\_id char(9),

F\_name varchar(20),

L\_name varchar(20),

Email varchar(60),

Password varchar(30),

Country varchar(20),

City varchar(20),

Street varchar(20),

Building varchar(20),

Postalcode varchar(10),

Membership varchar(20),

Status varchar(15),

Join\_day int,

Join\_month int,

Join\_year int,

Primary key (user\_id),

Foreign key (User\_id) references users (User\_id)

);

Insert into buyers values

(“222222222”, “michael”, “scofield”, “[michaels@yahoo.com](mailto:michaels@yahoo.com)”, “breaktheprison1”, “kenya”, “nairobi”, “lordson”, “43”, “14134545”, “R-Buyer”, “not active”, 4, 10, 2012),

(“333333333”, “sherlock”, “holmes”, “[office@holmes.com](mailto:office@holmes.com)”, “youdiebeforeyouknowit”, “england”, “london”, “alhambra”, “431”, “239843”, “P-Buyer”, “active”, 12, 6, 2018)

;

Create table sellers(

user\_id char(9),

F\_name varchar(20),

L\_name varchar(20),

Email varchar(60),

Password varchar(30),

Company varchar(30),

Company\_id varchar(10),

Country varchar(20),

City varchar(20),

Street varchar(20),

Building varchar(20),

Postalcode varchar(10),

Membership varchar(20),

Status varchar(15),

Join\_day int,

Join\_month int,

Join\_year int,

Primary key (user\_id),

Foreign key (User\_id) references users (User\_id)

);

Insert into sellers values

(“000000000”, “mohammad”, “taha”, “[mtaha220@gmail.com](mailto:mtaha220@gmail.com)”, “1234567”, “MT’s”, “90442“, “czech republic”, “prague”, “richard”, “23”, “2018500”, “PremiumSeller”, “active”, 1, 12, 1998),

(“111111111”, “rana”, “dakwar”, “[rana.dakwar@gmail.com](mailto:rana.dakwar@gmail.com)”, “7654321”, “ “ ,” “ , “spain”, “barcelona”, “saraya”, “87”, “789889”, “BasicSeller”, “active”, 2, 3, 2005)

;

Create table categories(

Category\_id char(9),

Name varchar(20),

Primary key (category\_id)

);

Insert into categories values

(“999999999”, “Cell Phones”),

(“888888888”, “Office Products”)

;

Create table sub\_categories(

Category\_id char(9),

Name varchar(20),

Primary key (category\_id, name),

Foreign key (category\_id) references categories (category\_id)

);

Insert into sub\_categories values

(“999999999”, “Accessories”),

(“999999999”, “Holsters & Sleeves”),

(“888888888”, “Office Electronics”),

(“888888888”, “Office Furniture”)

;

Create table products(

Product\_id char(9),

Name varchar(200),

Category\_id char(9),

Subcategory\_name varchar(20),

Variant1 varchar(100),

Variant2 varchar(100),

Variant3 varchar(100),

Primary key (product\_id),

Foreign key (category\_id, Subcategory\_name) references sub\_categories (category\_id, name)

);

Insert into products values

(“121212121”, “Apple AirPods”, “999999999”, “Accessories”, “brand: apple”, “color: white”, “Connectivity: wireless”),

(“131313131”, “Mkeke Compatible”, “999999999”, “Holsters & Sleeves”, “brand: Mkeke”, “color: clear”, “Material: TPU, Polycarbonate”),

(“141414141”, “HP OfficeJet 3830”, “888888888”, “Office Electronics”, “brand: HP”, “color: black”, “Printing Technology: Inkjet”),

(“151515151”, “Furinno 3-Tier Open Shelf”, “888888888”, “Office Furniture”, “Material: Particle Board”, “Furniture Finish: White”, “Brand: Furinno”)

;

Create table sold\_by(

Instance\_id char(9),

Product\_id char(9),

Seller\_id char(9),

Price int,

Total\_in\_stock int,

Description varchar(150),

Sessions int,

Primary key (instance\_id),

Foreign key (product\_id) references products (product\_id),

Foreign key (Seller\_id) references sellers (user\_id)

);

Insert into sold\_by values

(“aaaaaaaaa”, “121212121”, “000000000”, 100, 68, “apples original”, 3),

(“bbbbbbbbb”, “121212121”, “111111111”, 99, 233, “original product from apple”, 32),

(“ccccccccc“, “131313131”, “000000000”, 23, 14, “great sale”, 32),

(“ddddddddd”, “141414141”, “111111111”, 321, 124, “HP original”, 32),

(“eeeeeeeee”, “151515151”, “000000000”, 530, 43, “Furino new sale”, 21);

Create table payments(

Payment\_id char(9),

Giftcard char(9),

Discount\_code char(9),

creditcard\_num varchar(20),

Nameoncard varchar(50),

Country varchar(20),

City varchar(20),

Street varchar(20),

Building varchar(20),

Postalcode varchar(20),

Primary key (payment\_id)

);

Insert into payments values

(“909090909”, “218979293”, “ “, “1234567891011234”, “chong mi”, “US new york”, “NYC”, “5th avenue”, “788B”, “432143”),

(“898989898”, “ “, “111111111”, “1111222233334444”, “ben stiller”, “lithuania”, “capital city”, “main st”, “12h32”, “32144412”),

(“787878787”, “ “, “ “, “9999888877776666”, “james bond”, “italy”, “bologna”, “pedro”, “432”, “4321535”),

(“676767676”, “ “, “ “, “6666777755551111”, “judi abu khamis”, “um el tanafess”, “elfoaa”, “0”, “0”, “0000”);

Create table orders(

Order\_id char(9),

Buyer\_id char(9),

Payment\_id char(9),

Day int,

Month int,

Year int,

Primary key (order\_id),

Foreign key (buyer\_id) references buyers (user\_id),

Foreign key (payment\_id) references payments (payment\_id)

);

Insert into orders values

(“a11111111”, “222222222”, “909090909”, 1, 1, 2021),

(“a22222222”, “222222222”, “676767676”, 2, 2, 2021),

(“b11111111”, “333333333”, “898989898”, 3, 3, 2021);

Create table order\_products(

Order\_id char(9),

instance\_id char(9),

Amount int,

Primary key (order\_id, instance\_id),

Foreign key (instance\_id) references sold\_by (instance\_id),

Foreign key (order\_id) references orders (order\_id)

);

Insert into order\_products values

(“a11111111”, “aaaaaaaaa”, 2),

(“a11111111”, “bbbbbbbbb”, 1),

(“a22222222”, “ccccccccc”, 4),

(“a22222222”, “ddddddddd”, 3),

(“b11111111”, “eeeeeeeee”, 1),

(“b11111111”, “aaaaaaaaa”, 5);

Create table delivery(

Delivery\_id char(9),

Order\_id char(9),

Type varchar(50),

Time\_period int,

Day int,

Month int,

Year int,

Dest\_country varchar(50),

Dest\_city varchar(50),

Dest\_str varchar(50),

Dest\_building varchar(50),

Dest\_postalcode varchar(50),

Primary key (delivery\_id),

Foreign key (order\_id) references orders (order\_id)

);

Insert into delivery values

(“d11111111”, “a11111111”, “premium”, 5, 4, 2, 2021, “kenya”, “nairobi”, “lordson”, “43”, “14134545”),

(“d22222222”, “a22222222”, “standard”, 10, 5, 12, 2020, “kenya”, “nairobi”, “saint john”, “4321”, “43215”),

(“d33333333”, “b11111111”, “premium2”, 2, 3, 1, 2021, “england”, “london”, “alhambra”, “431”, “239843”);

Create table couriers(

Courier\_id char(9),

Name varchar(50),

Country varchar(50),

City varchar(50),

Street varchar(50),

Building varchar(50),

Postalcode varchar(50),

Primary key (courier\_id)

);

Insert into couriers values

(“co1111111”, “magnum shippings”, “germany”, “munich”, “st. peter”, “431” ,”3214”),

(“co2222222”, “premium shippings” , “south africa”, “capetown”, “detterss” , “e1431”, “4141f”),

(“co3333333”, “f class shippers”, “brazil”, “brazilia”, “frew”, “342”, “43242”);

Create table fulfillment\_centers(

Center\_id char(9),

Name varchar(50),

Country varchar(50),

City varchar(50),

Street varchar(50),

Building varchar(50),

Postalcode varchar(50),

Primary key (center\_id)

);

Insert into fulfillment\_centers values

(“c11111111”, “west china FC”, “china”, “beijing”, “hataaziya”, “3212B”, “968505”),

(“c22222222”, “cubian amazon FC”, “cuba”, “havana”, “loris beach”, “238d”, “9028437”),

(“c33333333”, “ott canada FC”, “canada”, “ottawa”, “frozen yogurt”, “43”, “09283”);

Create table shipped(

Delivery\_id char(9),

Center\_id char(9),

Courier\_id char(9),

Primary key (delivery\_id, center\_id, courier\_id),

Foreign key (delivery\_id) references delivery (delivery\_id),

Foreign key (center\_id) references fulfillment\_centers (center\_id),

Foreign key (courier\_id) references couriers (courier\_id)

);

Insert into shipped values

(“d11111111”, “c11111111”, “co1111111”),

(“d22222222”, “c22222222”, “co2222222”),

(“d33333333”, “c33333333”, “co3333333”);

Create table stored\_in(

instance\_id char(9),

Center\_id char(9),

Amount int,

Primary key (instance\_id, center\_id),

Foreign key (instance\_id) references sold\_by (instance\_id),

Foreign key (center\_id) references fulfillment\_centers (center\_id)

);

Insert into stored\_in values

(“aaaaaaaaa”, “c11111111”, 438279),

(“aaaaaaaaa”, “c22222222”, 398),

(“bbbbbbbbb”, “c22222222”, 38279),

(“ccccccccc”, “c33333333”, 92),

(“ddddddddd”, “c33333333”, 3920),

(“ddddddddd”, “c22222222”, 1123),

(“eeeeeeeee”, “c11111111”, 390);

Create table refunds(

Refund\_id char(9),

Order\_id char(9),

Instance\_id char(9),

Amount int,

Reason varchar(100),

Day int,

Month int,

Year int,

Primary key (refund\_id),

Foreign key (order\_id, instance\_id) references order\_products (order\_id, instance\_id)

);

Insert into refunds values

(“r11111111”, “a11111111”, “aaaaaaaaa”, 1, “regret”, 4, 2, 2021),

(“r22222222”, “a22222222”, “ccccccccc”, 2, “order slip”, 15, 2, 2021),

(“r33333333”, “a22222222”, “ccccccccc”, 2, “broken”, 17,2, 2021);

Create table reviews(

Review\_id char(9),

Buyer\_id char(9),

Instance\_id char(9),

Content varchar(250),

Overall\_rating int,

value\_for\_money int,

Variant1 int,

Variant2 int,

variant3 int,

Day int,

Month int,

Year int,

Primary key (review\_id),

Foreign key (buyer\_id) references buyers (user\_id),

Foreign key (instance\_id) references sold\_by (instance\_id)

);

Insert into reviews values

(“rev111111”, “222222222”, “aaaaaaaaa”, “great”, 5, 5, 4, 5, 5, 16, 2, 2021),

(“rev222222”, “222222222”, “bbbbbbbbb”, “bad”, 2, 2, 2, 1, 3, 16, 2, 2021),

(“rev333333”, “333333333”, “ccccccccc”, “not bad”, 3, 3, 2, 4, 3, 15, 2, 2021);

Create table questions(

Question\_id char(9),

Buyer\_id char(9),

Instance\_id char(9),

Content varchar(250),

Q\_day int,

Q\_month int,

Q\_year int,

Answer varchar(250),

A\_day int,

A\_month int,

A\_year int,

Primary key (question\_id),

Foreign key (buyer\_id) references buyers (user\_id),

Foreign key (instance\_id) references sold\_by (instance\_id)

);

Insert into questions values

(“q11111111”, “222222222”, “eeeeeeeee”, “is another colors?”, 1, 1, 2021, “look above idiot!”, 2, 2, 2021),

(“q22222222”, “333333333”, “ddddddddd”, “is it electric?”, 2, 1, 2021, “oh no! Yes nicholas tesla”, 1, 3, 1235);

Create table shopping\_carts(

Buyer\_id char(9),

Instance\_id char(9),

Amount int,

Primary key (buyer\_id, instance\_id),

Foreign key (buyer\_id) references buyers (user\_id),

Foreign key (instance\_id) references sold\_by (instance\_id)

);

Insert into shopping\_carts values

(“222222222”, “eeeeeeeee”, 2),

(“222222222”, “ddddddddd”, 6),

(“222222222”, “ccccccccc”, 5),

(“333333333”, “aaaaaaaaa”, 1),

(“333333333”, “bbbbbbbbb”, 2);

Create table wishlists(

Buyer\_id char(9),

Instance\_id char(9),

Amount int,

Primary key (buyer\_id, instance\_id),

Foreign key (buyer\_id) references buyers (user\_id),

Foreign key (instance\_id) references sold\_by (instance\_id)

);

Insert into wishlists values

(“222222222”, “bbbbbbbbb”, 4),

(“222222222”, “aaaaaaaaa”, 3),

(“333333333”, “ddddddddd”, 2),

(“333333333”, “eeeeeeeee”, 1);

5. נרמול טבלאות: קחו את 3 הטבלאות עם מספר העמודות הגדול ביותר, והסבירו עבור כל אחת מהן, באיזה רמה היא מנורמלת. [6 נק']

**buyers(user\_id(FK), f\_name, l\_name, email, password, address\_country, address\_city, address\_street, address\_no, address\_postalcode, membership, status, join\_day, join\_month, join\_year)**

* הטבלה מקיימת את רמת נרמול מספר 1 מכיוון שאין עמודות מרובות ערכים בטבלה.
* הטבלה מקיימת את רמת נרמול מספר 2 מכיוון שאין עמודות שתלויות רק בחלק מהעמודות המרכיבות את המפתח - המפתח מורכב מעמודה אחת בלבד.
* הטבלה אינה מקיימת את רמת נרמול מספר 3 מכיוון שקיימת תלות פונקציונלית בין עמודות שאינן חלק מהמפתח הראשי:

העמודה email יכולה בעצם לשמש כמפתח ראשי, והיא מגדירה את שאר העמודות בטבלה. בפרט העמודה ה- email מגדירה את עמודת password ואת העמודות membership, status, join\_day, join\_month, join\_year ועוד

העמודות address\_country ו- address\_postalcode, ביחד, מגדירות את העמודות address\_city, address\_street ו- address\_no.

נרמול הטבלה לפי דרישות רמה 3 יוביל ליצירת הטבלאות הבאות:

buyers(user\_id, email)

email\_datails(email, f\_name, l\_name, password, membership, status, join\_day, join\_month, join\_year)

email\_address(email, country, postalcode)

detailed\_address(country, postalcode, city, street, number)

* לאחר הפירוק לטבלאות לעיל, הטבלאות מקיימות את דרישות רמת נרמול BCNF.
* הטבלאות מקיימות את דרישות רמת נרמול מספר 4.

**sellers(user\_id(FK), f\_name, l\_name, email, password, company name, company id, address\_country, address\_city, address\_street, address\_no, address\_postalcode, membership, status, join\_day, join\_month, join\_year)**

* הטבלה מקיימת את רמת נרמול מספר 1 מכיוון שאין עמודות מרובות ערכים בטבלה.
* הטבלה מקיימת את רמת נרמול מספר 2 מכיוון שאין עמודות שתלויות רק בחלק מהעמודות המרכיבות את המפתח - המפתח מורכב מעמודה אחת בלבד.
* הטבלה אינה מקיימת את רמת נרמול מספר 3 מכיוון שקיימת תלות פונקציונלית בין עמודות שאינן חלק מהמפתח הראשי:

העמודה email יכולה בעצם לשמש כמפתח ראשי, והיא מגדירה את שאר העמודות בטבלה. בפרט העמודה ה- email מגדירה את עמודת password ואת העמודות membership, status, join\_day, join\_month, join\_year ועוד

העמודות address\_country ו- address\_postalcode, ביחד, מגדירות את העמודות address\_city, address\_street ו- address\_no.

נעיר כי אדם פרטי גם יכול להיות seller, במקרה זה company\_id הוא ת"ז של אותו אדם. אז לפי נתונים אלה, בנוסף ל-email גם העמודה company\_id מגדירה את company\_name, address\_country ו-address\_postalcode

נרמול הטבלה לפי דרישות רמה 3 יוביל ליצירת הטבלאות הבאות:

sellers(user\_id, email)

email\_details(email, password, company\_id, membership, status, join\_day, join\_month, join\_year)

co\_id\_details(company\_id, company\_name, country, postalcode)

detailed\_address(country, postalcode, city, street, number)

* לאחר הפירוק לטבלאות לעיל, הטבלאות מקיימות את דרישות רמת נרמול BCNF.
* הטבלאות מקיימות את דרישות רמת נרמול מספר 4.

**delivery(delivery\_id, order\_id(FK), delivery\_type, time\_period, shipment\_date\_day, shipment\_date\_month, shipment\_date\_year, destination\_address\_country, destination\_address\_city, destination\_address\_street, destination\_address\_number, destination\_address\_postalcode)**

* הטבלה מקיימת את רמת נרמול מספר 1 מכיוון שאין עמודות מרובות ערכים בטבלה.
* הטבלה מקיימת את רמת נרמול מספר 2 מכיוון שאין עמודות שתלויות רק בחלק מהעמודות המרכיבות את המפתח - המפתח מורכב מעמודה אחת בלבד.
* הטבלה אינה מקיימת את רמת נרמול מספר 3 מכיוון שקיימת תלות פונקציונלית בין עמודות שאינן חלק מהמפתח הראשי:

העמודה delivery\_type מגדירה את העמודה time\_period שכן משך הזמן נקבע לפי סוג המשלוח המבוקש;

העמודות העמודות address\_country ו- address\_postalcode, ביחד, מגדירות את העמודות address\_city, address\_street ו- address\_no.

נרמול הטבלה לפי דרישות רמה 3 יוביל ליצירת הטבלאות הבאות:

delivery(delivery\_id, order\_id(FK), delivery\_type, shipment\_date\_day, shipment\_date\_month, shipment\_date\_year, destination\_address\_country, destination\_address\_postalcode)

type\_period(delivery\_type, time\_period)

detailed\_address(country, postalcode, city, street, number)

* לאחר הפירוק לטבלאות לעיל, הטבלאות מקיימות את דרישות רמת נרמול BCNF.
* הטבלאות מקיימות את דרישות רמת נרמול מספר 4.

6. אתרו והציגו דף וובי למשל כמו זה, שמכיל את ה-KPIs שאליהם תתייחסו בסעיפים הבאים.

<https://helloprofit.com/amazon-seller-news/amazon-seller-analytics-10-kpis-winning-strategy/>

<https://www.sellerapp.com/blog/amazon-seller-kpi-measure/#3>

<https://salesfunnelhq.com/amazon-kpi/>

7. בסעיף זה עליכם ליצור 5 שאילתות

7.1. ציינו את חמשת המדדים החשובים ביותר למדידת ביצועי החברה (KPIs), למשל, יתכן שתרצו לבחור ב-" כמות הלקוחות הפעילים בכל חודש", או באנגלית " Monthly Active " Users.

[ענינו על שאלה זו במסגרת התשובה של שאלה 7.2.]

7.2. כעת, צרו 5 השאילתות שיענו על אותם KPIs שהגדרתם בסעיף הקודם, וצרפו לצד כל שאילתה את הפלט שלה. [15 נק']

1. **Revenue Value: Total amount of Payments per seller and Average Order Value: Total amount of Payments/ number of orders**

select seller\_id, F\_name as First\_name,L\_name as Last\_name, sum(price\*amount) as Revenue, sum(price\*amount)/count(amount) as "Average Order Value"

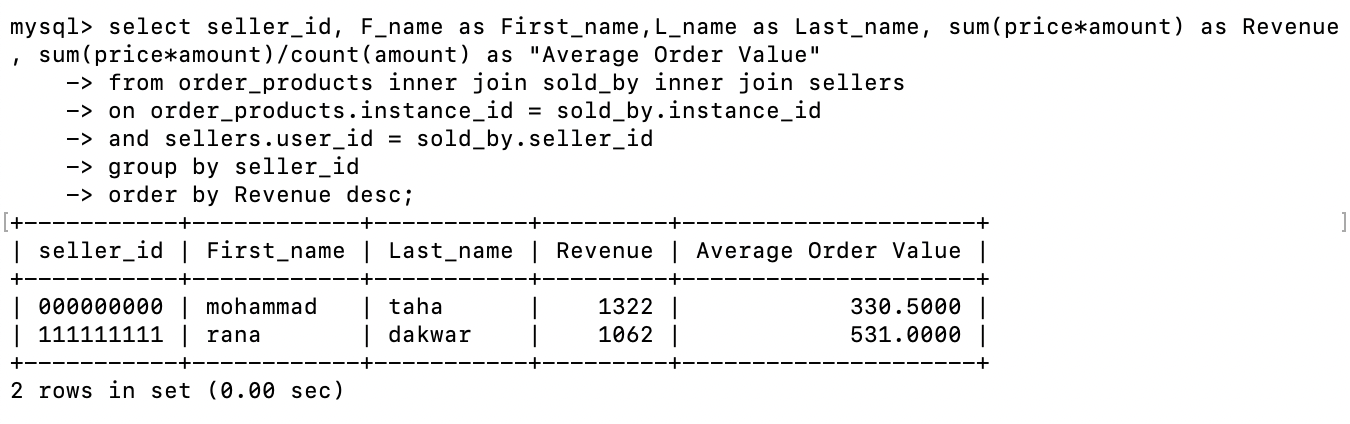
from order\_products inner join sold\_by inner join sellers

on order\_products.instance\_id = sold\_by.instance\_id

and sellers.user\_id = sold\_by.seller\_id

group by seller\_id

order by Revenue desc;

****

1. **Organic Conversion rate: number of units sold / number of sessions on the product seller page.** \*Values > 100% means users order more than 1 unit on avg per 1 order from a specific seller.

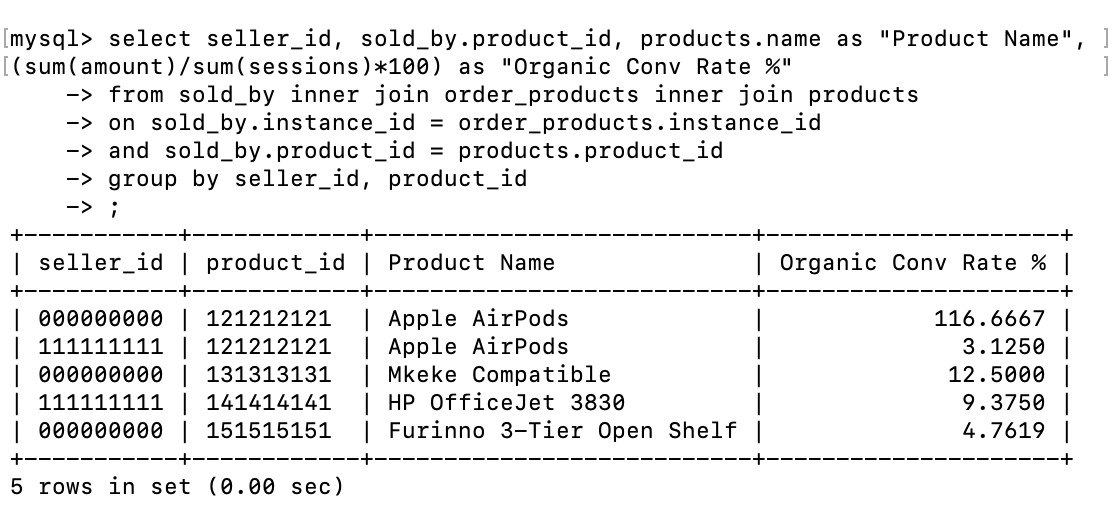
select seller\_id, sold\_by.product\_id, products.name as "Product Name", (sum(amount)/sum(sessions)\*100) as "Organic Conv Rate %"

from sold\_by inner join order\_products inner join products

on sold\_by.instance\_id = order\_products.instance\_id

and sold\_by.product\_id = products.product\_id

group by seller\_id, product\_id;

****

1. **Most popular categories**

select categories.name as Category , sum(amount) as "Sold Units"

from categories inner join products inner join sold\_by inner join order\_products

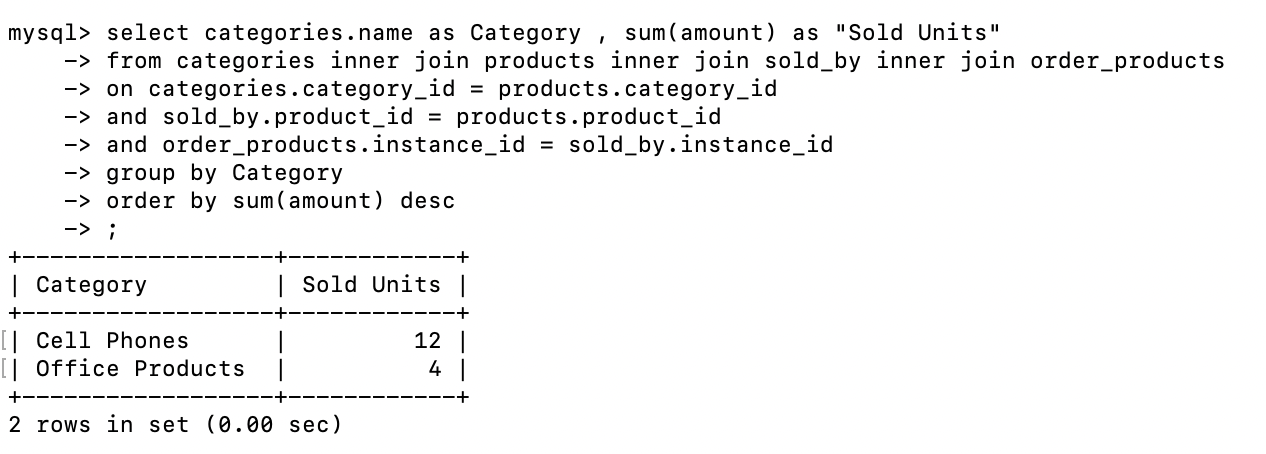
on categories.category\_id = products.category\_id

and sold\_by.product\_id = products.product\_id

and order\_products.instance\_id = sold\_by.instance\_id

group by Category

order by sum(amount) desc;

****

1. **Inventory Turns = AVG(Amount of Units Sold / number of units in stock). \***The higher the inventory turns, the better it is for business.

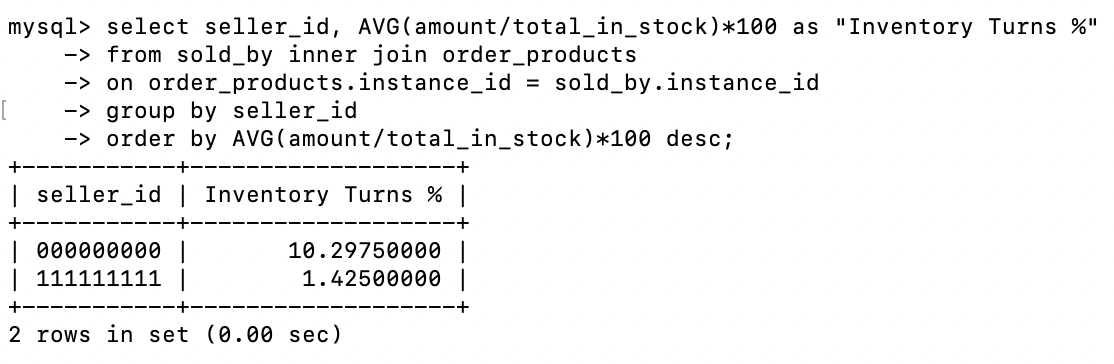
select seller\_id, AVG(amount/total\_in\_stock)\*100 as "Inventory Turns %"

from sold\_by inner join order\_products

on order\_products.instance\_id = sold\_by.instance\_id

group by seller\_id

order by AVG(amount/total\_in\_stock)\*100 desc;

****

1. **Quality of vendor by Shipping and Processing: Order rates according to the shipping center.**

select couriers.courier\_id, couriers.name as "Couriers Name", avg(Overall\_rating) as "Average Rating"

from couriers inner join shipped inner join stored\_in inner join reviews

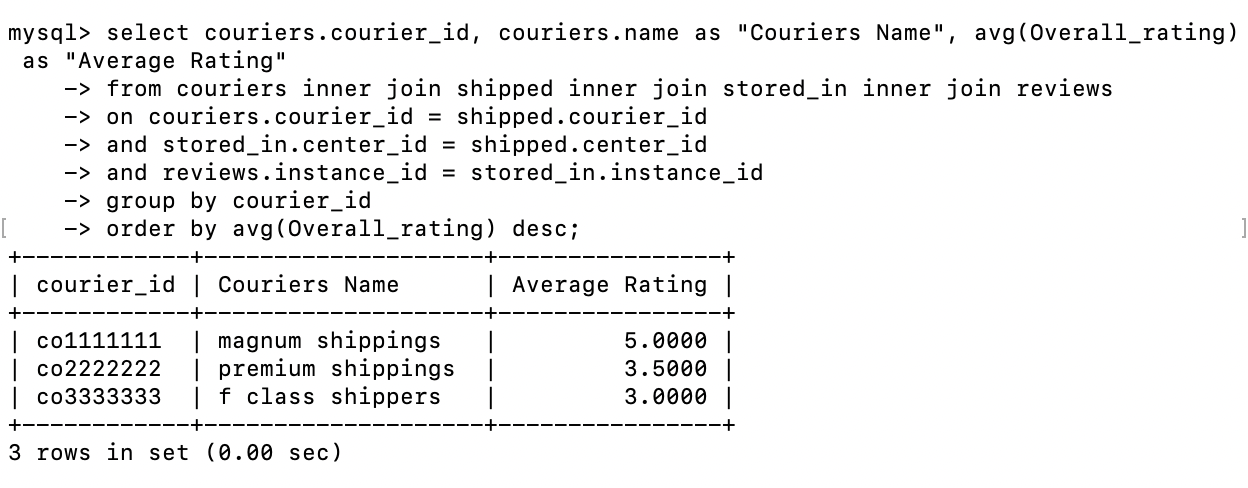
on couriers.courier\_id = shipped.courier\_id

and stored\_in.center\_id = shipped.center\_id

and reviews.instance\_id = stored\_in.instance\_id

group by courier\_id

order by avg(Overall\_rating) desc;

****

8. צרו 3 שאילתות באלגברה רלציונית עבור 3 ה-KPIs החשובים ביותר (מתוך ה-5 שטיפלתם בהם בסעיף הקודם) [9 נק']

1. **Revenue Value: Total amount of Payments per seller**

**τ revenue ↓**

**π seller\_id, f\_name → first\_name, l\_name → last\_name, SUM (price\*amount) → revenue**

**γ seller\_id, SUM (price\*amount) (order\_products ⋈ sellers . user\_id = sold\_by . seller\_id sold\_by ⋈ order\_products . instance\_id = sold\_by . instance\_id sellers)**

1. **Organic Conversion rate: number of units sold / number of sessions on the product seller page.**

**π seller\_id, sold\_by . product\_id, products . name → product\_name, SUM (amount) / SUM (sessions) \* 100 → organic\_conv\_rate**

**γ seller\_id, product\_id, (sold\_by ⋈ sold\_by . instance\_id = order\_products . instance\_id order\_products ⋈ sold\_by . product\_id = products . product\_id products)**

1. **Most popular categories**

**τ SUM (amount) ↓**

**π categories . name → category, SUM (amount) → sold\_units**

**γ category, SUM (amount) (categories ⋈ categories . category\_id = products . category\_id products ⋈ sold\_by . product\_id = products . product\_id sold\_by ⋈ order\_products . instance\_id = sold\_by . instance\_id order\_products)**

9. **בהמשך לסעיף 4, מהו שטח האחסון הנדרש, בהנחה שהטבלאות מכילות את כלל הנתונים (ולא רק הזנת נתונים מדגמית שביצעתם בסעיף 5)? נמקו והסבירו [5 נק']**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **table** | **col1** | **col2** | **col3** | **col4** | **col5** | **col6** | **col7** | **col8** | **col9** | **col10** | **col11** | **col12** | **col13** | **col14** | **col15** | **col16** | **col17** | **סה"כ נפח לרשומה אחת** |
| users | 9 | 20 | 20 | 60 | 30 | 7 | 20 | 20 | 20 | 20 | 10 | 20 | 15 | 4 | 4 | 4 |  | 283 |
| buyers | 9 | 20 | 20 | 60 | 30 | 20 | 20 | 20 | 20 | 10 | 20 | 15 | 4 | 4 | 4 |  |  | 276 |
| sellers | 9 | 20 | 20 | 60 | 30 | 30 | 10 | 20 | 20 | 20 | 20 | 10 | 20 | 15 | 4 | 4 | 4 | 316 |
| products | 9 | 200 | 9 | 20 | 100 | 100 | 100 |  |  |  |  |  |  |  |  |  |  | 538 |
| sold\_by | 9 | 9 | 9 | 4 | 4 | 150 | 4 |  |  |  |  |  |  |  |  |  |  | 189 |
| orders | 9 | 9 | 9 | 4 | 4 | 4 |  |  |  |  |  |  |  |  |  |  |  | 39 |
| payments | 9 | 9 | 9 | 20 | 50 | 20 | 20 | 20 | 20 | 20 |  |  |  |  |  |  |  | 197 |
| order\_products | 9 | 9 | 4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 22 |
| delivery | 9 | 9 | 50 | 4 | 4 | 4 | 4 | 50 | 50 | 50 | 50 | 50 |  |  |  |  |  | 334 |
| couriers | 9 | 50 | 50 | 50 | 50 | 50 | 50 |  |  |  |  |  |  |  |  |  |  | 309 |
| fulfillment\_centers | 9 | 50 | 50 | 50 | 50 | 50 | 50 |  |  |  |  |  |  |  |  |  |  | 309 |
| shipped | 9 | 9 | 9 |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 27 |
| stored\_in | 9 | 9 | 4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 22 |
| categories | 9 | 20 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 29 |
| sub\_categories | 9 | 20 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 29 |
| refunds | 9 | 9 | 9 | 4 | 100 | 4 | 4 | 4 |  |  |  |  |  |  |  |  |  | 143 |
| reviews | 9 | 9 | 9 | 250 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |  |  |  |  |  | 309 |
| questions | 9 | 9 | 9 | 250 | 4 | 4 | 4 | 250 | 4 | 4 | 4 |  |  |  |  |  |  | 551 |
| shopping\_carts | 9 | 9 | 4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 22 |
| wishlists | 9 | 9 | 4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 22 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **3966** |

כדי לחשב שטח האחסון הנדרש עבור תקופת זמן מסוימת, נחשב גודל של רשומה ונכפיל במספר הרשומות שיוזנו בתקופת זמן מסוימת. הטבלאות מחולקות לשלושה סוגי נתונים, טבלאות שגדלות לפי כמות המשתמשים, וטבלאות שגדלות לפי מספר העסקאות המבוצעות, טבלאות שגדלות לפי מספר המוצרים.

עבור טבלאות המוצרים: ממוצע גודל רשומה הוא 199, מספר הרשומות מוערך כ-350 מיליון\*, לכן צריך כ- 210GB עבור 3 הטבלאות: products + category + subcategory.

-עבור טבלאות המשתמשים:

גודל רשומה בטבלת הקונים הוא 276, מספר הרשומות מוערך כ-200 מיליון\*, לכן צריך כ-55GB.

גודל רשומה בטבלת המוכרים הוא 316, מספר הרשומות מוערך כ-5 מיליון\*, לכן צריך כ- 2GB.

גודל רשומה בטבלת המשתמשים הוא 283, מספר הרשומות מוערך כ-205 מיליון\*, לכן צריך כ- 58GB.

עבור טבלאות העסקאות : ממוצע גודל רשומה הוא 207, מספר הרשומות מוערך כ-144 מיליון\* בשנה. לכן צריך כ- 30GB בשנה, כפול 13 טבלאות = 650GB בשנה.

לכן, סה״כ, נדרש שטח אחסון של 210+55+2+58+650 = 975GB בשנה.

[\*https://www.oberlo.com/blog/amazon-statistics](https://www.oberlo.com/blog/amazon-statistics)

10. **קחו את הטבלה שלכם בעלת מספר השדות הגדול ביותר, הניחו כי יש בה כ-3,000,000 רשומות וכן שהיא נמצאת במצב של Index sequential file, על המפתח הראשי.**

10.1. **האם נכון לייצר עבור טבלה זאת אינדקס נוסף? נמקו והסבירו [3 נק']**

**sellers(user\_id(FK), f\_name, l\_name, email, password, company name, company id, address\_country, address\_city, address\_street, address\_no, address\_postalcode, membership, status, join\_day, join\_month, join\_year)**

האינדקס user\_id הינו אינדקס מספק עבור טבלה זו, זאת מכיוון שהוא מאפשר בין היתר יצירת קובץ sparse\_index וכך להקטין את שטח האחסון של קובץ האינדקס, כמו כן, אינדקס זה מאפשר לנו ליצור multivalued index וכך להקטין עוד את הזמן של הופסת או מחיקת רשומות מהקובץ וכן להקטין את זמן הגישה והחיפוש. הוספת אינדקס נוסף יהיה כדאי לאור אופן הפעילות הספציפי של הארגון, כפי שנשיב בסעיף הבא.

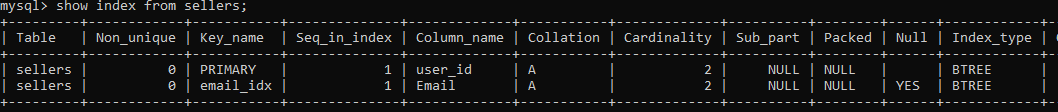
10.2. **ה-DBA הנחה אתכם להוסיף אינדקס אחד נוסף לטבלה זאת, מעבר למפתח הראשי. מהו האינדקס הכי משמעותי עבור הארגון, בטבלה זאת? נמקו והסבירו [3 נק']**

האינדקס המשמעותי בארגון הוא ה-email שכן פעולות רבות מחייבות שימוש ב-email לשם גישה לנתונים השמורים בטבלה:

10.3. **צרו עבור טבלה זאת את האינדקס שציינתם בסעיף הקודם באמצעות MySQL [3 נק']**

Create unique index email\_idx on sellers(email);

show index from sellers;



10.4. **בהנחה שהקובץ נמצא במצב של Index sequential file ושזמן הגישה 5ms, מה המשמעות על השיפור הצפוי בזמן השליפה, בשימוש באינדקס של המפתח הראשי, אל מול שליפה ללא אינדקס? [4 נק']**

**מצב 1:** שימוש באינדקס של המפתח הראשי, הקובץ שמור במצב של Index sequential file, ומדובר בשכבה אחד של אינדקס כך שמדובר בקובץ אינדקס גדול ששמור על הדיסק ולא על הזיכרון (לאור גודלו). במקרה זה כדי לשלוף מסוים תוך שימוש באינדקס של המפתח הראשי, ע"י ביצוע חיפוש בינארי, נצטרך ל- log2(3000000) = 21.51653107 גישות לדיסק, 5ms כל אחת, ובסה"כ ל- 107.58265535ms.

**מצב 2:** שימוש באינדקס של המפתח הראשי, הקובץ שמור במצב של Index sequential file, ומדובר אינדקס רב שכבתי המאפשר שמירת אינדקס על הזיכרון. במקרה זה חיפוש המפתח באינדקס יתבצע על ידי גישות לזיכרון ולא לדיסק, ורק לאחר מציאת המפתח המבוקש תתבצע גישה אחת לדיסק, כך שהעלות היא 5ms. האמור הינו בהנחה כי מדובר בשתי רמות, כך שניתן לקבל את המצביע לרשומה המבוקשת כבר בחיפוש על הקובץ השמור על הזיכרון ונעשה גישה אחת, אולם שימוש בשתי רמות אידנקס אינו אפשרי מכיוון שגם הרמה השניה של האינדקס תהיה גדולה מלהישמר על הזיכרון, כך נצטרך לעוד רמות, וכך גדל מספר הגישות לדיסק לשליפת נתון.נניח כי גודל רשומת אינדקס הוא 13 בתים (רשומת מפתח ראשי + מצביע), ונניח כי הנפח המקסימלי המוקצה לנו לשימוש מתוך הזיכרון הוא 13MB, אז ניתן לשמור על הזיכרון מיליון רשומות אינדקס, כך שכל רשומה תצביע לבלוק המונה 3 רשומות. מכיוון שלא כל רשומות האינדסק רשומות על הזיכרון, לא בהכרח נוכל למצוא את המצביע לרשומה המבוקשת כבר מחיפוש בינארי על הזיכרון, אולם, ניתן להשתמש באלגוריתם העובד בסיבוכיות( log2(N למציאת הבלוק הרלבנטי בו נמצאת הרשומה המבוקשת - אלגוריתם זה רץ על רשומות האינדקס השמורות על הזיכרון ונותן לנו את המצביע לבלוק הרלוונטי. בעת שנמצא את הבלוק הרלוונטי הצטרך לגשת (3)log2 פעמים לדיסק (מכיוון שבכל בלוק יש 3 רשומות שאחת מהן היא הרלבנטית). לכן: בסה"כ יש לנו log2(1,000,000) = 19.931568569 גישות לזיכרון ועוד 1.6 גישות לדיסק (8ms). יוער כי ככל שקטן הנפח המוקצה לנו בתוך הזיכרון יגדל מספר הגישות לדיסק.

**מצב 3:** ללא שימוש באינדקס של המפתח הראשי. במצב זה ברור כי נתונים שמורים על הדיסק, וכדי למצוא רשימה מסוימת יש לעבור על כל הרשומות ששמורות בדיסק. במקרה הגרוע נצטרך לגשת מספר פעמים לדיסק כמספר הרשומות 3,000,000 כפול 5 = 15,000,000ms.

10.5. **קיבלתם את המנדט להחליט באיזה סוג אינדקס נכון להשתמש בטבלה זאת עבור המפתח הראשי. מהו האינדקס שעליו תמליצו? הסבירו ונמקו, אל מול שימוש באינדקסים אחרים אותם תפשתם פחות מתאימים [6 נק']**

בשלב ראשון נבחן את סוגי אינדקסים העומדים לרשותנו:

**חלופה א':** indexed-sequential-file. את הטבלה נשמור את קובץ הטבלה בדיסק indexed-sequential-file dense, עבור המפתח הראשי. כפי שהזכרנו לעיל, רמה אחת של אינדקס אינה אופציה קיימת מכיוון ששמירת קובץ האינדקס על הזיכרון אינו אפשרי לאור גודלו. לכן נטה להשתמש במפתח רב שכבתי. נניח כי גודל רשומת אינדקס הוא 13 בתים (רשומת מפתח ראשי + מצביע), ונניח כי הנפח המקסימלי המוקצה לנו לשימוש מתוך הזיכרון הוא 13MB, אז ניתן לשמור על הזיכרון מיליון רשומות אינדקס, כך שכל רשומה תצביע לבלוק המונה 3 רשומות. מכיוון שלא כל רשומות האינדסק רשומות על הזיכרון, לא בהכרח נוכל למצוא את המצביע לרשומה המבוקשת כבר מחיפוש בינארי על הזיכרון, אולם, ניתן להשתמש באלגוריתם העובד בסיבוכיות( log2(N למציאת הבלוק הרלוונטי בו נמצאת הרשומה המבוקשת - אלגוריתם זה רץ על רשומות האינדקס השמורות על הזיכרון ונותן לנו את המצביע לבלוק הרלוונטי. בעת שנמצא את הבלוק הרלוונטי נצטרך לגשת עד( log2(3 פעמים לדיסק (מכיוון שבכל בלוק יש 3 רשומות שאחת מהן היא הרלוונטית). לכן: בסה"כ יש לנו log2(1,000,000) = 19.931568569 גישות לזיכרון ועוד 1.6 גישות לדיסק (8ms).

האמור הינו בקשר לגישה לנתון מסוים, ובהקשר זה יוער כי ככל שקטן הנפח המוקצה לנו בתוך הזיכרון או שגדל מספר הרשומות בטבלה, יגדל מספר הגישות לדיסק. ביחס להוספת רשומות לטבלה, זה יכול להיות מסבך לאור ה- overflows שייגרמו על בלוקים מסוימים כפי שלמדנו בהרצאה. מבחינת עלות זמן, הוספת ומחיקת רשומות לטבלה יכול להיות גדול מאוד לאור כך שנצטרך לבצע שינוי לרשומות רבות ולפיכך לבצע גישות רבות לדיסק. לאור עניין פעולות המחיקה וההופסה המרובות, מחייבת ארגון מחדש של ממסד הנתונים בכל תקופה.

**חלופה ב':** B+ TREE. ידוע כי גודלו של node הוא כגודל בלוק, 4KB. בהנחה כי גודלה של רשומת אינדקס היא 40 בתים, אז בכל node ניתן לאחסן 100 רשומות אינדקס. לפיכך, בהינתן כי יש לנו 3,000,000 רשומות בטבלה, גובהו של העץ הוא log50(3,000,000) = 3.8123811711. זה אומר כי בכל גישה לנתון בעץ אנחנו ניגש ל-4 nodes. בהינתן כי עלות הגישה היא 5ms אז כל אחת מהפעולות (גישה, מחיקה, הוספה) תעלה לנו 20ms. יודגש כי סיבוכיות הזמן גם במחיקה וגם בהוספה שווה לגובה העץ, עם זאת, ראינו בהרצאה כי זה יכול להיות פחות מכך לאור כך שלא כל פעולות ההוספה והמחיקה מחייבות עדכון כלשהו ב- nodes, ולרוב הדבר לא מחייב עדכון מלבד ההוספה/המחיקה עצמה.

נדגיש כי חלופה זו פותרת לנו את הקשיים הכרוכים בשימוש בחלופה א', זאת מכיוון שפעולות ההוספה והמחיקה אינן מחייבות שינוי רחב היקף בבסיס הנתונים, אלא בשינויים מקומיים עם סיבוכיות השווה לגובה העץ במקרה הגרוע. כמו כן, בעיית ה- overflow אינה שגרתית לשיטה זו. לכן, חלופה זו עדיפה על חלופה א'.

**חלופה ג':** static hash indexing. נניח ואנחנו עושים static hash indexing לפי המפתח הראשי. גם מצב זה תלוי בכמות הנתונים שמותר לנו לשמור על הזיכרון. על הזיכרון נרצה לשמור טבלה שממפה int למצביע בתוך הדיסק. Int זה הוא תוצאת פונקציית ההאש. מצב אחד הוא שבכל דלי בדיסק שמורה רשומה אחת מתוך 3 מיליון רשומות שיש לנו, במצב זה פונקציית ההאש תהיה f = mod 3,000,000, ועל הזיכרון נשמור טבלה בעלת 3,000,000 רשומות, כאשר כל רשומה מורכבת מעמודה int ששווה ל- 4BYTE ועמודה של מצביע שגם הוא 4BYTE. אז גודל הטבלה יהיה 3 מיליון כפול 8 בתים = 24 מיליון בתים = 24MB. אם סביר לנצל נפח זה מתוך הזיכרון זמן הגישה לנתונים בתוך הטבלה יושפע מאלה: ברגע שקיבלנו את תוצאת ההאש, נחפש את התוצאה בתוך הטבלה בזיכרון בעלות של log2 של 3 מיליון (שימוש בחיפוש בינארי), אז יהיו לנו 21.51653107 גישות לזיכרון. ברגע שמצאנו את הכניסה הרלבנטית בתוך הטבלה נצטרך לגישה אחת בלבד לדיסק לשליפה הנתוך בעלות של 5MS. נעיר כי במצב זה השתמשנו ב- Hash File Organization.

המצב ישתנה למשל אם נדע כי אין אפשרות להשתמש ב- 24MB מתוך הזיכרון, אז נשנה את פונקציית ההאש ואת הטבלה כך שתכיל פחות רשומות. במצב זה מספר הגישות לזיכרון לאיתור הכניסה הרלבנטית יקטן ואילו מספר הגישות לדיסק יגדל שכן במצב זה כל דלי מכיל יותר רשומות. יודגש כי במצב זה השימוש בהאש הוא ליצירת index-structure ולא רק ל- file-organization, ואינדקס ההאש ישמש כמפתח משני ולא ראשי. אם גודל הדלי הוא כגודל בלוק בדיסק = 4KB, וכל רשומה היא 316 בתים, אז בכל דלי ניתן לאחסן 12 רשומות נתונים. עבור 3 מיליון רשומות נצטרך ל- 250,000 דליים. במצב דברים זה, הטבלה על הזיכרון תכיל 250 אלף רשומות, 8 בתים כל אחת ובסה"כ 2MB; פונקציית ההאש תהיה mod 250,000. מבחינת עלות זמן: יש לנו ( log2(250000 גישות לזיכרון, ועוד( log2(12 גישות לדיסק לשם שליפת נתון.

למרות נתוני הגישה הסבירים, חלופה זו סובלת מבעיות רבות: אם מספר הדליים ההתחלתי קטן מדי והקובץ הולך וגדל, הביצועים יקטנו לאור ריבוי ה-overflows; אם מלכתחילה אנחנו משריינים מקום לטובת גידול עתידי בנתונים, אז אנחנו מבזבזים מקום רב ערך בממסד הנתונים; אם בסיס הנתונים מתכווץ שוב יהיה בזבוז מקום.

**חלופה ד':** dynamic hash indexing. חלופה הזו באה לגשר על הקשיים הדבוקים בשיטה הקודמת כך שהיא תופסת זיכרון באופן דינמי לפי הגידול או הקיטון במספר הרשומות. נניח ואנחנו עושים dynamic hash indexing לפי המפתח הראשי, ונניח כי הטבלה במצב hash-ordered גם לפי המפתח הראשי. עבור המפתח הראשי אנחנו צריכים לשמור, בתוך הזיכרון, טבלה הממפה את המפתח פריפקס לדלי הרלבנטי בתוך הדיסק, באמצעות מצביע. אם נמשיך בנתוני הדוגמא האחרונה לעיל, אז כדי למפות ל- 250 אלף דליים נצטרך לגודל פריפקס של 18 = ( log2(250,000. בזיכרון נשמור טבלה המורכבת מעמודת המפתח המשני פריפקס בגודל 18 ביטים ועוד עמודת מצביעים בגודל 4 בתים ובסה"ד 6.25 בתים כל רשומה כפול 250 אלף רשומות = 1.5MB בערך. זמן הגישה יהיה זהה לזמן הגישה ב- static עם אותה דוגמא.

מצב אחר יהיה כאשר נרצה למפות ישירות מתוך הטבלה בזיכרון לרשומה בדיסק. בשביל למפות 3,000,000 רשומות נצטרך לפריפקס בגודל של 22 = ( log2(3,000,000. צריך לאחסן פריפקס זה בטבלה עם מצביעים, כל רשומה בטבלה תהיה בגודל 6.75 בתים, ויש לנו עד 3 מיליון רשומות, לפיכך גודל הטבלה בסה"כ יהיה 20.25MB. הטבלה ממפה מפתח פריפקס (מפתח משני) לרשומה הרלבנטית בדיסק. זמן הגישה יהיה ( log2(3,000,000 גישות לזיכרון ועוד גישה אחת 5MS לדיסק.

**מה עדיף?** הסברנו לעיל כי חלופה ב' עדיפה על חלופה א', וכי חלופה ד' עדיפה על חלופה ג'. בהשוואה בין חלופה ב' לחלופה ד' נעדיף את חלופה ב', מכיוון שבהשוואה לחלופה ד', בחלופה ב' אנחנו לא צריכים לתפוס חלק מהזיכרון כדי לאחסן נתונים וכך נחסוך מקום וזמן גישה לזיכרון. ביחס לגישות לדיסק, מספר הגישות מאוד קרוב בין שתי החלופות, זאת לאור כך שסיבוכיות הזמן בשימוש בחלופה ב' היא במצב גרוע, ולכך שמצב גרוע זה הוא נדיר יחסית, ולכן בדרך כלל פעולות ההוספה, הגישה והמחיקה יעלו לנו מספר גישות לדיסק והוא קטן מגובה העץ.

10.6. **האם תשובתכם בסעיף הקודם תשתנה, אם תתבקשו למקסם את כלל הטבלאות? נמקו והסבירו [3 נק']**

לא, השיקולים שנלקחו בחשבון בסעיף הקודם הם מבוססים על נתוני האחסון, מקסום הטבלאות לא ישנה את זה, ולכן התשובה לא תשתנה.