תרגיל בית 2 חלק יבש

מגישים:

שם	ת.ז	מייל
לואי שקיר	212080162	loaishaqir@campus.technion.ac.il
אחמד עיאשה	212273981	Ahmad-ai@campus.technion.ac.il

עיצוב מסד הנתונים והסבר לפונקציות

עיצוב מסד הנתונים:

Tables

Owner
owner_id name

Customer
customer_id customer_name

Apartment
apartment_id address city country size

CustomerReviews
Customer_id
Apartment_id
Review_date
Rating
Review_text

CustomerReservations
customer_id
apartment_id
start_date
end_date
total_price

Views

ApartmentReviewsFullData
Owner_id
Apartment_id
Owner_name
Customer_id
Review_date
Rating
Review_text

ApartmentOwnersFullData
Owner_id
Owner_name
Apartment_id
Address
City
Country
size

OwnerCustomerReservations
Owner_id
Owner_name
Customer_id
Apartment_id

OwnerAvgRating
Owner_id
Avg_rating

ApartmentAvgRating
Owner_id
Apartment_id
Avg_rating

ApartmentPriceRatingAvg
Apartment_id
Address
City
Country
Size
Price_per_night
rating

CustomerReviewsProd
Customer_a_id
Customer_b_id
Apartment_id
Customer_a_rating
Customer_b_rating

CustomerRatingsAvgRatio
Customer_a_id
Customer_b_id
Avg_ratio

CustomersUnreviewedApartmentsAvgRatio
Customer_a_id
Customer_b_id
Avg_ratio
Unreviewed_apartment_id

UnreviewedApartment
Customer_id
Unreviewed_apartment_id

CustomerUnreviewedApartmentsFilter
Customer_id
Unreviewed_apartment_id
Expected_rating

סקירה כללית של הטבלאות:

בנינו טבלה לכל אחד מה business obejcts (Customer, Apartment, Owner) הנתונים לנו בתרגיל כך שכל טבלה שבנינו מכילה בדיוק את אותם attributes כפי שמוגדר בקובץ התרגיל בלי שינויים חוץ מהגדרת טיפוסי התכונות והגדרת המפתח כדי לשמור על אילוצי האובייקטים.

והוספנו עוד שתי טבלאות שהיו מתבקשות:

CustomerReservations שומרת ומסדרת לנו את כל ההזמנות שעשו לקוחות ואחנו צריכים את המידע הזה למימוש פונקציות שונות.

טבלה זו מכילה 5 שדות: customer_id, apartment_id, start_date, end_date, total_price שזה כל המידע שאנחנו מקבלים עבור הזמנה כלשהי שלקוח מבצע.

CustomerReviews שומרת ומסדרת לנו את כל החוות דעת שערכו לקוחות בדירות שכבר היו בהם וטבלה זו מכילה 5 שדות customer_id, apartment_id, review_date, rating, review_text

שזה כל המידע שאנחנו מקבלים עבור חוות דעת כלשהי שלקוח מבצע עבור דירה שכבר היה בה.

סקירה כללית של ה Views :

ApartmentOwnersFullData

המבט הזה עושה JOIN בין ApartmentOwners, Owner, Apartment מה שנותן לנו בסוף טבלה אחת שמאחדת את כל המידע שיש בידינו עבור בעל דירה ועל מזהה הדירה וכל הפרטים הנלווים.

ApartmentReviewsFullData

המבט הזה עושה right outer join בין ApartmentOwnersFullData לבין CustomerReviews ומבט זה משמש אותנו כאבן בניין בסיסית שמחברת לנו בין המידע על הבעלים של כל דירה והמידע על הדירוגים שדירות קיבלו ותכף נשתמש בו לצורך הגדרת שני המבטים הבאים,

הצירוף הוא חיצוני ימני בגלל שייתכן ויש לנו דירה שקיבלה דירוגים עוד לפני שאנחנו יודעים מי הבעל שלה, ודירות כאלו אנחנו כן רוצים לקחת בחשבון כמחשבים ממוצע דירוגים וכולי.

ApartmentAvgRating

מבט זה לוקח את המבט הקודם ופשוט מוסיף לו GROUP BY על apartment_id, owner_id ומבט זה נועד במיוחד לפי דרישות הפונקציה get_apartment_rating

OwnerAvgRating

מבט זה דומה מאוד למבט הקודם רק במקום לעשות GROUP BY על apartment_id, owner_id כאן אנחנו עושים GROUP BY רק על owner_id ומבט זה נועד במיוחד לפי דרישות הפונקציה get_owner_rating

OwnerCustomerReservations

מבט זה נחוץ לנו כדי לחבר את שמות בעלי הדירות לבין ההזמנות שנעשו בדירות שלהן,

ולכן כשמו כן הוא זה פשוט מבט שעושה RIGHT OUTER JOIN בין ApartmentOwnersFullData שטבלה זה מחברת בין שם בעל הדירה למזהה הדירה ומכאן נעשה צירוף עם CustomerReservations על apartment_id וכך נקבל את המידע שאנו רוצים שהוא החיבור המלא בין שם בעל בדירה לבין ההזמנות שנעשו בדירות שלו, הצירוף הוא חיצוני ימני בגלל שייתכן ויש לנו דירה שקיבלה דירוגים עוד לפני שאנחנו יודעים מי הבעל שלה, ודירות כאלו אנחנו כן רוצים לקחת בחשבון כמחשבים ממוצע דירוגים וכולי.

ApartmentPriceRatingAvg

מבט זה נועד במיוחד לעזור לנו לאחד מידע רלוונטי כדי לממש את הפונקציה best_value_for_money וזה נעשה באופן הבא: צריכים לדעת את כל הפרטים על Apartment ובנוסף אנחנו צריכים לדעת את המחיר הממוצע ללילה בכל דירה שאת זה אנחנו מקבלים מ CustomerReservations וגם אנחנו רוצים את ה ratings שאת אלה מקבלים מ ApartmentReviewsFullData ולכן עשיתי מבט שעושה JOIN בין הטבלאות האלה בשם ApartmentPriceRatingAvg וזה נותן לנו את כל המידע הנדרש

CustomerReviewsProd

מבט זה נועד בשביל לתת לנו מענה על השאלה על איזה דירות יש לנו לקוחות שונים שנתנו חוות דעת ואת מידע זה אנחנו צריכים כדי לקחת אותו מכאן ולעבד אותו יותר בשביל שבסוף נצליח לממש את הפונקציה get_apartment_recommendation ולכן מבט זה פשוט עושה מכפלה בין CustomerReviews לעצמו ומפלטר לשורות שיש שני לקוחות שנתנו חוות דעת על אותה דירה.

CustomerRatingsAvgRation

עוד מבט שמאפיין כעוד אבן בניין בסיסית לתוך המטרה של מימוש get_apartment_recommendation ולכן טבלה זו לוקחת את המבט הקודם CustomerReviewsProd עושה group by customer_a_id, customer_b_id כדי שנקבל "ממוצע של כמה שני לקוחות מסכימים על הדירוג שהם נותנים לדירות ששניהם עשו חוות דעת על" וזה בדיוק מה שמבט זה מחזיר לנו זה 3 עמודות מהצורה:

Customer_a_id, customer_b_id, avg_ratio

UnreviewedApartments

מבט זה מחזיר לנו שתי עמודות: customer_id, unreviewed_apartment_id

שכשמם כן הם זה מבט שנותן לנו מידע ועונה על השאלה "איזה לקוחות לא עשו review לאיזה apartments" ומידע זה אנחנו צריכים כעוד אבן בניין בדרך לבניית פתרון ומימוש לפונקציה get_apartment_recommendation.

משתמשים כאן במכפלה בין Customer, Apartment ומורידים ממנה את כל השורות של מזהה לקוח ו מזהה דירה שמופיעים בשורה ביחד בטבלה CustomerReviews ובסוף מורידים כפילויות.

CustomersUnreviewedApartmentsAvgRatio

מבט זה עושה צירוף בין CustomerRatingAvgRatio לבין UnreviewedApartments מה שמחזיר לנו טבלה עם 4 עמודות של:

Customer_a_id, customer_b_id, avg_ration, unreviewed_apartment_id

במידע זה נוכל להשתמש בהמשך כדי לשערך את חוות דעת של כל לקוח על דירה שעוד לא עשה לה חוות דעת לפי שמידע שיש לנו על לקוחות שונים שנתנו חוות דעת לדירה זו ויש לנו גם את AvgRatio. מה שחסר זה מה הדירוג שהלקוח השני נתן לכל דירה שלקוח ראשון לא לקח ובשביל זה יש לנו את המבט הבא

CustomersUnreviewedApartmentsFilter

מבט זה עושה צירוף בין המבט הקודם לבין CustomerReviews כדי לענות על הבקשה שחסרה כפי שהסברנו במבט הקודם וזה הדירוג שלקוח ב נתן לדירה ובנוסף על הדרך במבט זה אנחנו מחזירים עמודה בשם expected_rating שהיא בעצם הממוצע של rating/avg_ratio וכמובן שזה צריך להיעשות פר קבוצה כאשר כל קבוצה אנחנו מקבלים מ GROUP BY customer_a_id, unreviewed_apartment_id כי אנחנו צריכים לדעת מה יהיה הדירוג המשוערך לכל לקוח שעדיין לא דירג כל דירה.

מימוש CRUD API:

ReturnValue add_owner(owner: Owner)

נחלץ את השם והמזהה של ה Owner שנקבל כקלט ונוסיף שורה לטבלה שלנו Owner עם הנתונים הללו.

כמובן שכל הבדיקות על האילוצים נעשות אוטומטית על ידי המסד נתונים מבלי הצורך לבדוק כאן תקינות קלט.

Owner get_owner(owner_id: int)

נחפש את ה Owner בעל ה owner_id בטבלת Owner באמצעות שאילתה פשוטה שרק מחפשת את הכניסה בטבלה שבה ה owner_id כמו שקיבלנו בקלט, ומשם נקבל גם את השם ונעטוף אותם באובייקט מסוג Owner ונחזיר אותו. (במידה ויש).

ReturnValue delete_owner(owner_id: int)

נמחק כניסה מהטבלה Owner שמתאימה ל owner_id שקיבלנו, עושים זאת באמצעות שאילתה פשוטה שרק מחפשת את הכניסה בטבלה שבה ה owner_id כמו שקיבלנו בקלט.

ReturnValue add_apartment(apartment: Apartment)

נחלץ את הנתונים מהאובייקט Apartment שקיבלנו ונוסיף אותם כפי שהם לטבלה Apartment.

Apartment get_apartment(apartment_id: int)

נחפש את ה Apartment בעלת ה apartment_id בטבלת Apartment באמצעות שאילתה פשוטה שרק מחפשת את הכניסה בטבלה שבה ה apartment_id כמו שקיבלנו בקלט, ומשם נקבל גם את שאר הנתונים ונעטוף אותם באובייקט מסוג Apartment ונחזיר אותו. (במידה ויש).

ReturnValue delete_apartment(apartment_id: int)

נמחק כניסה מהטבלה Apartment שמתאימה ל apartment_id שקיבלנו, עושים זאת באמצעות שאילתה פשוטה שרק מחפשת את הכניסה בטבלה שבה ה apartment_id כמו שקיבלנו בקלט.

ReturnValue add_customer(customer: Customer)

נחלץ את הנתונים מהאובייקט Customer שקיבלנו ונוסיף אותם כפי שהם לטבלה Customer

Customer get_customer(customer_id: int)

נחפש את ה Customer בעל ה customer_id בטבלת Customer באמצעות שאילתה פשוטה שרק מחפשת את הכניסה בטבלה שבה ה customer_id כמו שקיבלנו בקלט, ומשם נקבל גם את שאר הנתונים ונעטוף אותם באובייקט מסוג Customer ונחזיר אותו. (במידה ויש).

ReturnValue delete_customer(customer_id: int)

נמחק כניסה מהטבלה Customer שמתאימה ל customer_id שקיבלנו, עושים זאת באמצעות שאילתה פשוטה שרק מחפשת את הכניסה בטבלה שבה ה customer_id כמו שקיבלנו בקלט.

ReturnValue customer_made_reservation(customer_id: int, apartment_id: int, start_date: date, end_date: date, total_price: float)

נוסיף כניסה לטבלה CustomerReservations בדיוק עם הפרמטרים שקיבלנו, אבל מה שמיוחד כאן הוא שבהוספה אנחנו עושים עוד בדיקות שלא יכולנו לדרוש ממסד הנתונים לעשות בצורה אוטומטית נשתמש ב nested queries כדי לבדוק האם יש כבר בטבלה שאנחנו רוצים להוסיף את הכניסה אליה איזשהי reservation שחופפת את ה reservation שאנחנו רוצים להוסיף, ובודקים כל צורות החפיפה האפשריות.

מה שמעניין אותנו מה subquery הזה הוא רק לדעת אם יש כניסה חופפת או לא ולכן מה שאנחנו מחזירים ממנה הוא רק SELECT 1 כי זה מספיק לנו כדי להחליט ועוטפים את כל זה ב WHERE NOT EXISTS .. (subquery) וכך עשינו את הבדיקה על reservations חופפים, אם הכל תקין אז נוסיף כניסה מתאימה.

ReturnValue customer_cancelled_reservation(customer_id: int, apartment_id: int, start_date: date)

נחפש את הכניסה המתאימה לפרמטרים שקיבלנו בטבלה CustomerReservations ונמחק אותה.

ReturnValue customer_reviewed_apartment(customer_id: int , apartment_id: int, review_date: date, rating: int, review_text: str)

נוסיף כניסה ל CustomerReviews עם הפרמטרים שקיבלנו, הבדיקה שכל לקוח עושה חוות דעת פעם אחת נבדק בצורה אוטומטית באמצעות מסד הנתונים שלנו וה constraints שהגדרנו לו, אבל הבדיקה שהיה לו רישום שהסתיים לפני חוות הדעת אנחנו עושים באמצעות שימוש ב subquery שתחפש לנו כניסה מתאימה ב CustomerReservations שבה יש את אותו מס לקוח ושהזימון שלו נגמר לפני review_date ובמידה וכן אנחנו עושים SELECT 1 כאילו פשוט להגיד לנו שיש כזאת כניסה ועוטפים את זה ב WHERE EXISTS וכך נוכל לדעת אם אפשר להוסיף את הכניסה או לא.

ReturnValue customer_updated_review(customer_id: int, apartment_id:int, update_date: date, new_rating: int, new_text: str)

פשוט נחפש את הכניסה בטבלה CustomerReviews המתאימה לפרמטרים שקיבלנו ומעדכנים את הנתונים שם, ורק עושים את הבדיקה ש update_date >= review_date ואם כן זה בפרט יגיד לנו שמותר ללקוח זה לתת חוות דעת על דירה זו בתאריך זה ולכן מאפשרים לו לעדכן.

ReturnValue owner_owns_apartment(owner_id: int, apartment_id: int)

פשוט מוסיפים כניסה בטבלה ApartmentOwners עם הטפל (owner_id, apartment_id).

ReturnValue owner_drops_apartment(owner_id: int, apartment_id: int)

מוחקים כניסה מהטבלה ApartmentOwners שמכילה את ה owner_id, apartment_id שקיבלנו בקלט (במידה ויש).

Owner get_apartment_owner(apartment_id: int)

נחפש את ה Owner המתאים ב VIEW שעשינו בשם ApartmentOwnersFullData שהוא בעצם JOIN בין הטבלאות ApartmentOwners JOIN Owner JOIN Apartment וזה מחזיר לנו ממש את כל המידע הנדרש החל מבעל הדירה למס שלו לשם שלו ועם הנתונים המלאים על הדירה עצמה, וכשיש לנו טבלה כזאת קל לחפש את הכניסה שמתאימה ל apartment_id ולהחזיר את הפרטים שמגדירים אובייקט Owner משם.

List[Apartment] get_owner_apartments(owner_id: int)

כעת שוב נשתמש בטבלה שלנו כמו בסעיף קודם שמאחדת את כל המידע הנדרש, לכן נחפש את כל הכניסות בטבלה ApartmentOwnersFullData שבהם יש את ה owner_id כמו שקיבלנו בקלט ומשם נעשה הטלה על האטריבוטים המגדירים דירה ונחזיר אותם בתור list of Apartments.

מימוש BASIC API:

float get_apartment_rating(apartment_id: int)

כפי שהסברנו כשהגדרנו את המבט ApartmentAvgRating למעלה מבט זה מכיל את כל המידע שאנחנו צריכים בשביל הפונקציה הזאת ולכן רק מה שנותר לנו זה לחפש את ה apartment_id במבט זה.

float get_owner_rating(owner_id: int)

כפי שהסברנו כשהגדרנו את המבט OwnerAvgRating למעלה מבט זה מכיל את כל המידע שאנחנו צריכים בשביל הפונקציה הזאת ולכן רק מה שנותר לנו זה לחפש את ה owner_id במבט זה.

Customer get_top_customer()

נעשה JOIN בין Customer, CustomerReservations כדי לקבל טבלה עם המידע הכולל של שם הלקוח והמזהה שלו והמזהים של כל הדירות שעשה עבורם הזמנות (יש עוד מידע אבל זה המידע הרלוונטי לנו) לאחר מכן נעשה GROUP BY על customer_id, customer_nameZ ואז נעשה ORDER BY COUNT(*) DESC, customer_id ASC שזה ימיין לנו את הקבוצות לפי מספר ההזמנות שכל לקוח עשה בסדר יורד (שורה ראשונה היא הלקוח שעשה הכי הרבה הזמנות) ואז לפי customer_id בסדר עולה וזה בשביל לענות על דרישות הפונקציה שבמקרה שיש כמה לקוחות שעשו את אותו מספר הזמנות אז להחזיר את הלקוח בעל המזהה הקטן.

List[Tuple[str, int]] reservations_per_owner()

לוקחים את המבט שהסברנו עליו לעיל OwnerCustomerReservations שדואג לחבר בין שמות בעלי הדירות לבין ההזמנות שנעשו בהם ומכאן נעשה GROUP BY על owner_name, owner_id ונספור את מספר השורות בכל קבוצה, ומכאן נסדר את המידע בצורה הנדרשת כlist של tuples בפייתון.

מימוש ADVANCED API:

List[Owners] get_all_location_owners()

נעשה GROUP BY על owner_id, owner_name מהטבלה ApartmentOwnersFullData ונפלטר עבור השורות עם מספר עירים ומדינות (מצורפות) בלי כפילויות ששווה לסך מספר העירים ומדינות (מצורפות) בלי כפיליות וזה יגיד שכל owner_id, owner_name שיישאר בפלט של שאילתה זה בדיוק אומר שמס הערים שיש לו בהם דירות = סך מספר הערים שיש בהם דירות ולכן זה אומר שיש לו דירה בכל עיר שמכילה דירות.

ובסוף נחזיר רשימה של כל הבעלים כדי שתתאים לפלט המצופה

Apartment best_value_for_money()

למימוש פונקציה זו צריך לשים לב שאנחנו צריכים להחזיר אובייקט מטיפוס Apartment כלומר אנחנו צריכים לדעת את כל הפרטים על Apartment ובנוסף אנחנו צריכים לדעת את המחיר

הממוצע ללילה בכל דירה שאת זה אנחנו מקבלים מ CustomerReservations וגם אנחנו רוצים את ה ratings שאת אלה מקבלים מ ApartmentReviewsFullData ולכן עשיתי מבט שעושה JOIN בין הטבלאות האלה בשם ApartmentPriceRatingAvg וזה נותן לנו את כל המידע הנדרש. ומכאן במימוש הפונקציה אני אסביר איך מעבדים את המידע הזה כדי לענות על דרישות הפונקציה.

עכשיו אחרי שיש לנו מבט שמאחד את כל המידע שאנחנו רוצים קודם כל נעשה GROUP BY apartment_id כדי לעשות ממוצע על rating ו price_per_night לכל דירה, וגם נסדר אותם לפי value_for_money בסדר יורד ונעשה LIMIT 1 כדי לקבל בשורה ראשונה את הדירה עם ה best_value_for_money ו value_for_money מחושב על ידי הממוצע של rating חלקי הממוצע של price_per_night.

List[Tuple[int, float] profit_per_month(year: int)

למימוש פונקציה זו יש לנו את כל המידע הנדרש בתוך הטבלה CustomerReservations אבל עשינו פילטור כדי שנקבל רק את השורות עם end_date שמתאים לשנה שקיבלנו כקלט נעשה GROUP BY month כדי לסכום את הרווח של כל חודש בנפרד.

List[Tuple[Apartment, float]] get_apartment_recommendation(customer_id: int)

פונקציה זו מורכבת וצריך לפרק אותה לכמה חלקים, בשביל להבין איך מימשנו את הפונקציה הזו צריך לקרוא את תיעוד המבטים האלה לפי הסדר (שם אנחנו הסברנו מה מהות כל מבט ואיך הוא עוזר לנו לעשות עוד צעד קדימה במימוש פונקציה זו)

1- CustomerReviewsProd
2- CustomerRatingsAvgRation
3- UnreviewedApartments
4- CustomersUnreviewedApartmentsAvgRatio
5- CustomersUnreviewedApartmentsFilter

אחרי שקוראים את ההסבר על המבטים האלה לפי הסדר, מה שנותר לנו זה פשוט לקחת את המבט האחרון CustomersUnreviewedApartmentsFilter ולחפש שם את הכניסה המתאימה ל customer_id וזהו זה נותן לנו את כל המידע שאנו צריכים