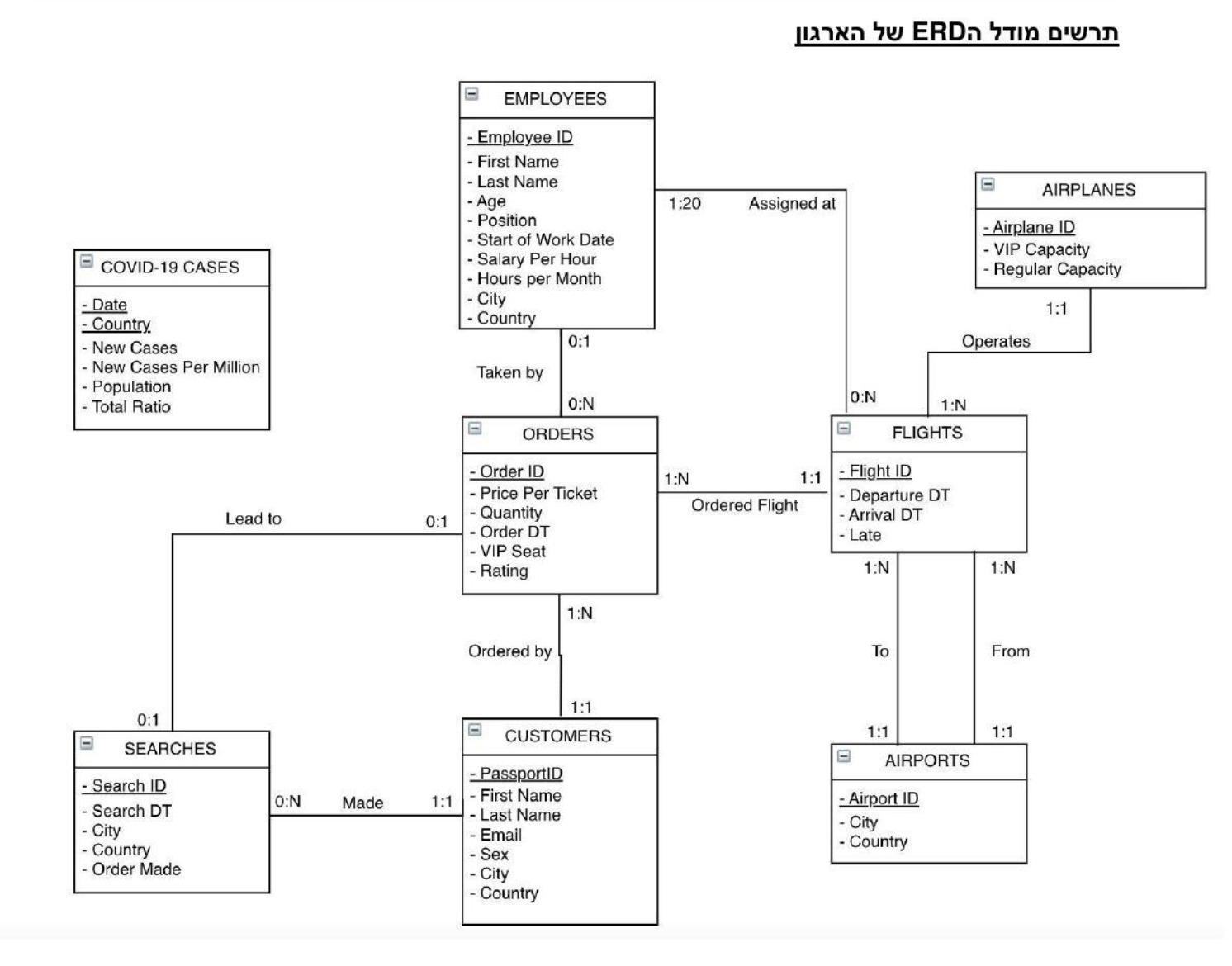
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **מס' קבוצה** | **שם הפרוייקט** | | | **תאריך הגשה** |
| 9 | חברת תעופה אל- על | | | 14/09/2022 |
| **מספרי תעודות הזהות של המגישים** | | | | |
| רותם חזן (315329839) | | נדב צברי(316462464) | ירדן ששון(318646171) | |

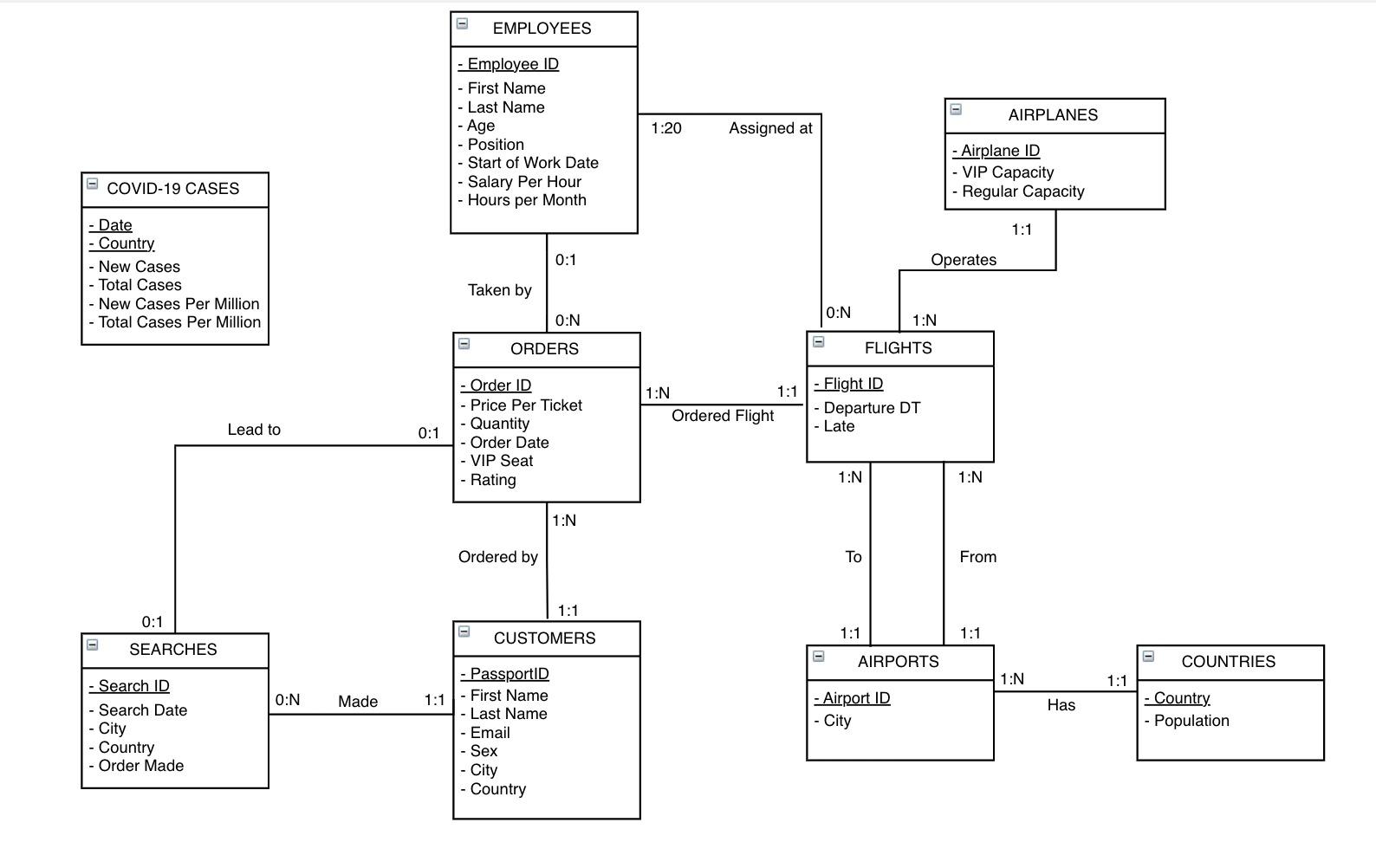
**חלק ב' – הכנת תשתית הנתונים**

# (20%) יישום מקורות הנתונים ומחסן הנתונים הטבלאי

**תרשים ERD לפני התיקון:**



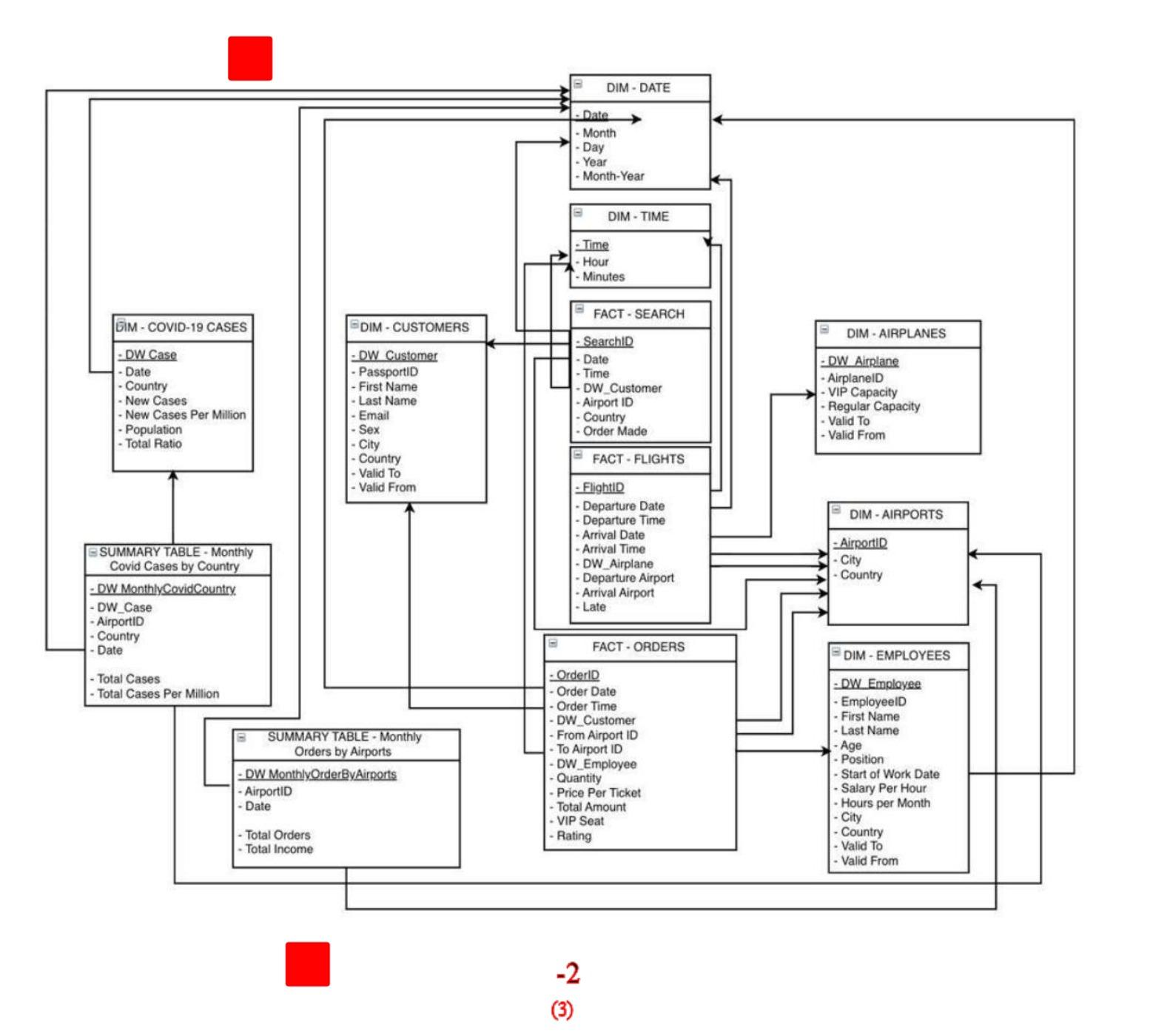
**תרשים ERD אחרי התיקון:**

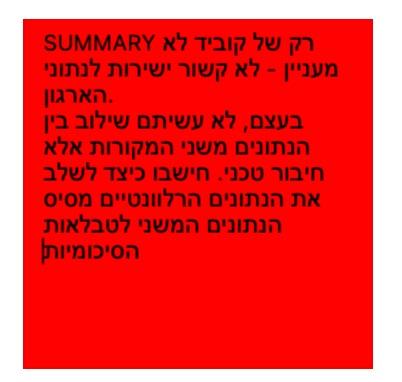


הוספנו יישות של מדינה , וקישרנו אותה לשדה התעופה בקרדינליות של יחיד לרבים.

השינוי בטבלת הנתונים:

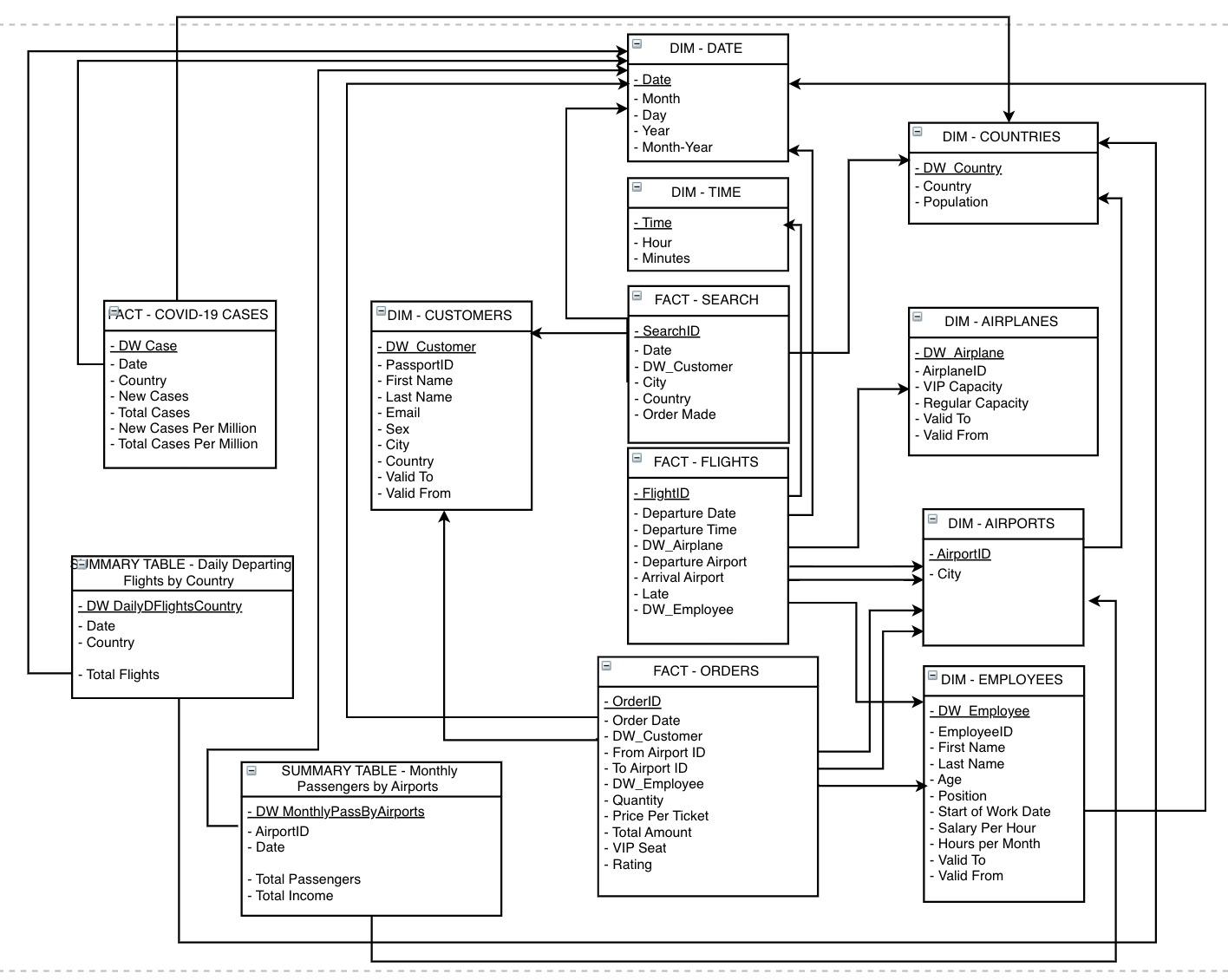
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **הטבלה** | **שם השדה** | **סוג הנתונים** | **סולם המדידה** | **תיאור השדה** |
| **EMPLOYEES**  רשימת עובדים | EmployeeID | Int | שמי | מזהה העובד |
| First Name | Varchar(20) | שמי | שם פרטי של העובד |
| Last name | Varchar(20) | שמי | שם משפחה של העובד |
| Age | Small int | שמי | גיל העובד |
| Position | Varchar(20) | סדר | תפקיד העובד |
| Start of work date | date | רווח | תאריך תחילת עבודה בחברה |
| Hourly per hour | money | שמי | משכורת שעתית |
| Hours per month | Int | שמי | שעות עבודה בחודש לעובד |
| City | Varchar(20) | שמי | עיר מגורים של העובד |
| Country | Varchar(20) | שמי | מדינת המגורים של העובד |
| **AIRPLANES**  רשימת המטוסים בידי החברה | AirplaneID | Varchar(3) | שמי | מזהה המטוס |
| VIPCapacity | Small int | רווח | מספר מושבים במחלקות יוקרה |
| RegCapacity | Small int | רווח | מספר מושבים במחלקת תיירים |
| **AIRPORTS**  רשימת שדות התעופה אליהם החברה טסה | AirportID | Varchar(3) | שמי | מזהה שדה התעופה |
| City | Varchar(20) | שמי | העיר בה נמצא שדה התעופה |
| Country(countries) | Varchar(20) | שמי | המדינה בה נמצא שדה התעופה |
| **FLIGHTS**  רשימת כלל הטיסות | FlightID | Varchar(5) | שמי | מזהה הטיסה |
| Airplane (AIRPLANES) | Varchar(3) | שמי | מזהה המטוס שמפעיל את הטיסה |
| From (AIRPORTS) | Varchar(3) | שמי | מזהה שדה התעופה ממנה יוצאת הטיסה |
| To (AIRPORTS) | Varchar(3) | שמי | מזהה שדה התעופה בה נוחתת הטיסה |
| Departure | datetime | רווח | תאריך ושעת המראה  שעון ישראל |
| Arrival | datetime | רווח | תאריך ושעת נחיתה  שעון ישראל |
| Late | Bit (1- late, 0- on time) | שמי | בוליאני – האם הטיסה יצאה בזמן או לא |
| **CUSTOMERS**  רשימת הלקוחות | PassportID | Int | שמי | מספר דרכון |
| First Name | Varchar(20) | שמי | שם פרטי |
| Last Name | Varchar(20) | שמי | שם משפחה |
| Email | Varchar(60) | שמי | אימייל |
| Sex | Varchar(6) | שמי | מין |
| City | Varchar(20) | שמי | עיר מגורים של הלקוח |
| Country | Varchar(20) | שמי | מדינת מגורים של הלקוח |
| **ORDERS**  רשימת ההזמנות של הלקוחות לטיסות | OrderID | Int | שמי | מזהה ההזמנה |
| FlightID (FLIGHTS) | Varchar(5) | שמי | מזהה מספר הטיסה |
| Customer (CUSTOMERS) | Int | שמי | מספר דרכון של הלקוח המזמין |
| Employee (EMPLOYEES) | Int | שמי | מזהה העובד שביצע את ההזמנה |
| Price per ticket | Money | שמי | מחיר לכרטיס |
| Quantity | Int | רווח | כמות כרטיסים בהזמנה |
| OrderDT | datetime | רווח | תאריך ושעה בה התבצעה ההזמנה |
| VIPSeat | Bit (1- has VIP seat, 0- reg seat) | שמי | האם הלקוח יושב במושב במחלקות היוקרה |
| Rating | Smallint | סדר | דירוג שביעות רצון הלקוח מהשירות |
| **SEARCHES**  רשימת החיפושים המתבצעים ע״י הלקוחות | SearchID | Int | שמי | מזהה החיפוש |
| SearchDT | Datetime | רווח | תאריך ושעה בה התבצע החיפוש |
| Customer (CUSTOMERS) | int | שמי | מזהה הלקוח שביצע את החיפוש |
| Country | Varchar(20) | שמי | מדינת היעד בחיפוש |
| City | Varchar(20) | שמי | עיר היעד בחיפוש |
| OrderMade | Bit (1- order was made, 0- no) | שמי | בוליאני – האם התבצעה הזמנה של טיסה מהחיפוש |
| Order id | Int | שמי | מזהה ההזמנה במידה ואחרי החיפוש בוצע הזמנה |
| **ASSIGNMENTS**  שיבוצי טיסות הדיילים | EmployeeID (EMPLOYEES) | Int | שמי | מזהה העובד |
| FlightID (FLIGHTS) | Varchar(5) | שמי | מזהה הטיסה |
| **COVID19 CASES**  רשימת מקרים קורונה לפי תאריך | Date | Date | רווח | תאריך תיעוד המקרים |
| Country | Varchar(20) | שמי | מדינה |
| NewCases | Int | יחס | סך מקרי קורונה חדשים לתאריך מסוים |
| NCPerMillion | Int | יחס | יחס מקרי קורונה חדשים לכל מיליון אנשים באוכלוסייה |
| Population | bigint | יחס | אוכלוסייה |
| TotalRatio | int | יחס | יחס חולי קורונה בהתאם לאוכלוסייה |
| **COUNTRIES**  רשימת המדינות שאיתן החברה עובדת | Country | Varchar(20) | שמי | מזהה מדינה |
| Population | Bigint | יחס | גודל האוכלוסיה במדינה |

**סכימת כוכב לפני התיקון :** תמונה שמכילה ריבוע

התיאור נוצר באופן אוטומטי

הנחות ושינויים עיקריים:

1. לכל מדינה יש כמה שדות תעופה , ולכן החלטנו להוסיף מימד של מדינה המכיל את המפתח הראשי – שם המדינה , ומספר אזרחים במדינה.
2. שינינו את טבלת הסיכום של נתוני הקורונה לטבלת סיכום לפי טיסות , מדינה ותאריך – אשר מקשרת בין בסיס הנתונים הראשי למשני.



**סכימת כוכב אחרי התיקון :**

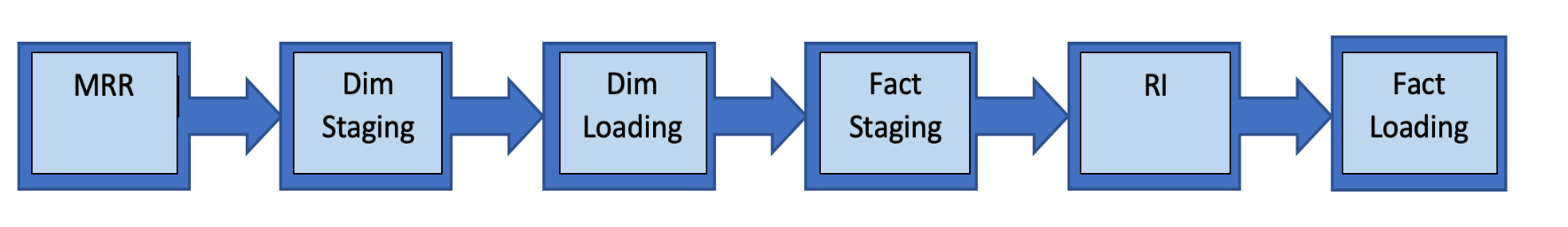
השינויים בטבלה :

|  |  |  |  |
| --- | --- | --- | --- |
| **הטבלה** | **שם השדה** | **סוג הנתונים** | **תיאור השדה** |
| **D\_CUSTOMERS**  טבלת מימד משתנה מסוג 2, המכילה את רשימות הלקוחות, תוך שמירת גרסאות | DW Customer | Int | מזהה לקוח  **מפתח עזר, כל ערך משקף גרסת לקוח בעלת טווח תאריכים מוגדר** |
| Passport id | Int | מזהה הלקוח |
| First Name | Varchar(20) | שם פרטי של הלקוח |
| Last name | Varchar(20) | שם משפחה של הלקוח |
| Email | Varchar(60) | מייל הלקוח |
| Sex | Varchar(6) | מין הלקוח |
| City | Varchar(20) | העיר בה מתגורר הלקוח  **עיר המגורים עשוייה להשתנות לעיתים, ומכאן הצורך בשמירת גרסאות** |
| Country | Varchar(20) | המדינה בה מתגורר הלקוח  **המדינה עשוייה להשתנות לעיתים, ומכאן הצורך בשמירת גרסאות** |
| Valid From | Date | תאריך תחילת תוקף הגרסה |
| Valid Until | Date | תאריך סיום תוקף הגרסה |
| **D\_EMPLOYEES**  טבלת מימד משתנה לאט מסוג 2, המכילה את פרטי העובדים בחברה, תוך שמירת גרסאות | DW employee | Int | מזהה עובד  **מפתח עזר, כל ערך משקף גרסת מוצר בעלת טווח תאריכים מוגדר** |
| Employee id | Int | מזהה העובד |
| First Name | Varchar(20) | שם פרטי של העובד |
| Last name | Varchar(20) | שם משפחה של העובד |
| Age | Int | גיל העובד |
| Position | Varchar(20) | תפקיד נוכחי של העובד  **תפקיד העובד עשוי להשתנות לעיתים, ומכאן הצורך בשמירת גרסאות** |
| Start of work | DT | תאריך קליטה לחברה |
| Salary per hour | Money | שכר שעתי של העובד |
| Hours per month | Int | מספר שעות של העובד בחודש |
| City | Varchar(20) | העיר בה מתגורר העובד  **עיר המגורים עשוייה להשתנות לעיתים, ומכאן הצורך בשמירת גרסאות** |
| Country | Varchar(20) | המדינה בה מתגורר העובד  **המדינה עשוייה להשתנות לעיתים, ומכאן הצורך בשמירת גרסאות** |
| Valid From | Date | תאריך תחילת תוקף הגרסה |
| Valid Until | Date | תאריך סיום תוקף הגרסה |
| **D\_AIRPLANES**  טבלת מימד משתנה לאט מסוג 2, המכילה את פרטי המטוסים בחברה, תוך שמירת גרסאות | DW\_airplanes | Int | מזהה פרטי המטוס  **מפתח עזר, משקף צירוף ייחודי של מזהה הזמנה ומזהה מוצר** |
| Airplane id | Varchar(3) | מזהה המטוס |
| VIP Capacity | Int | כמה מושבים מסוג VIP יש במטוס  **כמות המושבי VIP עשויה להשתנות לעיתים, ומכאן הצורך בשמירת גרסאות** |
| Regular Capacity | Int | כמה מושבים רגילים יש במטוס  **כמות המושבים הרגילים עשוייה להשתנות לעיתים, ומכאן הצורך בשמירת גרסאות** |
| Valid From | Date | תאריך תחילת תוקף הגרסה |
| Valid Until | Date | תאריך סיום תוקף הגרסה |
| **D\_AIRPORTS**  טבלת מימד ,המכילה את פרטי שדות התעופה בחברה | Airport id | Varchar(3) | מזהה שדה התעופה |
| City | Varchar(20) | העיר בה נמצא שדה התעופה |
| **D\_DATE**  טבלת מימד ,המכילה תאריכים שונים. | Date | Date | מזהה התאריך |
| Month | Int | מספר החודש בתאריך |
| Day | Int | מספר היום בתאריך |
| Year | Int | מספר השנה בתאריך |
| Month-year | Date | מספר החודש משולב עם השנה |
| **D\_TIME**  טבלת מימד המכילה את השעות . | Time | Time | מזהה השעה |
| Hour | Int | מספר השעה |
| Minutes | Int | מספר הדקות |
| **F\_COVID19**  טבלת מימד של מקרי הקורונה במדינות איתן החברה עובדת | DW\_Cases | Int | מזהה מקרה הקורונה  **מפתח עזר, משקף צירוף ייחודי של מזהה מדינה ומזהה תאריך** |
| Date | Date | מזהה של תאריך וזמן |
| Country | Varchar(20) | מזהה של מדינה |
| New Cases | Int | מספר מקרים חדשים בכל מדינה בתאריך מסויים |
| Total Cases | Int | סך מקרים חדשים עד תאריך זה |
| New Cases Per Million | Int | מספר יחסי של מקרים חדשים בכל מדינה בתאריך מסויים |
| Total Cases Per Million | int | סך מקרים עד תאריך זה |
| **F\_SEARCH**  טבלת עובדה פרטנית, המסכמת את פרטי החיפושים לרמת גרעיניות של חיפוש בודדת | Search id | Int | מזהה החיפוש |
| Search DT | DT | תאריך ושעת חיפוש |
| DW customers | Int | מזהה הלקוח שביצע את החיפוש |
| Airport id | Varchar(3) | מזהה שדה התעופה |
| Country | Varchar(20) | שם המדינה אותו הלקוח מחפש |
| Order made | Bit | 1. האם החיפוש הסתיים בהזמנה 0- החיפוש אינו הסתיים בהזמנה |
| **F\_FLIGHTS**  טבלת עובדה פרטנית, המסכמת את פרטי הטיסות לרמת גרעיניות של טיסה בודדת | Dw flights | Int | מפתח עזר של טיסה |
| Flight id | Varchar(5) | מזהה הטיסה |
| Departure DT | DT | תאריך ושעת יציאה של הטיסה |
| Arrival DT | DT | תאריך ושעת נחיתה של הטיסה |
| Departure airport | Varchar(3) | שדה התעופה ממנה הטיסה יצאה |
| Arrival airport | Varchar(3) | שדה התעופה אליה הטיסה נוחתת |
| Late | Bit | 1. הטיסה איחרה 0 – הטיסה הגיע בזמן |
| **F\_ORDERS**  טבלת עובדה פרטנית, המסכמת את פרטי הטיסות לרמת גרעיניות של טיסה בודדת | Order id | Int | מזהה הזמנה |
| Order DT | DT | תאריך ושעת הזמנה |
| DW customers | Int | מזהה הלקוח שביצע את ההזמנה |
| From (airport id) | Varchar(3) | שדה התעופה ממנו הטיסה עתידה לצאת |
| To (airport id) | Varchar(3) | שדה התעופה אליו הטיסה עתידה לנחות |
| DW employee | Int | מזהה העובד שלקח את ההזמנה , אם ההזמנה התבצעה באינטרנט השדה יהיה ריק |
| Quantity | Int | כמות כרטיסים שהוזמנו בהזמנה |
| Price per ticket | Money | מחיר לכרטיס |
| Total amount | Money | סכום כולל להזמנה  **שדה מחושב של סך כל ההזמנה – מחיר לכרטיס \* כמות הכרטיסים** |
| VIP seats | Int | שדה המציין כמה כרטיסי VIPנרכשו באותה ההזמנה |
| Rating | Int | לאחר קבלת השירות מהחברה ישלח ללקוח סקר שבו הוא ידרש לדרג את חווית השירות. דירוג מ1-5 |
| **F\_MONTHLY PASS\_BY\_AIRPORTS**  טבלת עובדה סיכומית שמציגה איזה שדה תעופה הוא הכי פעיל (טיסות יוצאות) | DW\_MonthlyPassByAirports | Int | מזהה ייחודי לכלל הנוסעים שאר נסעו לכל שדה תעופה בחודש מסויים |
| Date | Date | מפתח זר לטבלת מימד תאריך ומשם נחלץ את החודש והשנה |
| AirportID | Int | מזהה ייחודי של שדה התעופה |
| Total Passengers | Int | שדה המציג את סך הנוסעים אשר נסעו בחודש מסויים ונחתו בשדה תעופה מסויים  **שדה חישובי המתבצע ע"י סכימה של כלל הQUANTITY בכל ההזמנות שהתבצעו בחודש מסויים לשדה תעופה מסויים** |
| Total income | Int | שדה המציג את ההכנסות של כל שדה תעופה בכל חודש  **שדה חישובי המתבצע ע"י סכימה של הכנסות מכלל העסקאות בשדה תעופה מסויים בחודש מסויים** |
| **F\_DAILY DEPARTING FLIGHTS BY COUNTRY**  טבלת עובדה סיכומית , שמציגה את כמות המקרי הקורונה באופן חודשי | DW\_DailyDFlightsCountry | Int | מזהה מקרי קורונה בחודש לפי מדינה  **מפתח עזר, משקף צירוף ייחודי של מדינה וחודש** |
| Date | Date | מתוך טבלת מימד תאריך ניקח את התאריך לפי יום/חודש/שנה |
| Country | Varchar(20) | מזהה של המדינה |
| Total Flights | int | שדה המציג את סך הטיסות היוצאות ממדינה מסוימת בתאריך מסוים  **שדה חישובי המתבצע ע"י סכימה של כלל הטיסות היוצאות מאותה מדינה** |
| **D\_COUNTRIES**  טבלת מימד של מדינה | DW\_country | Int | מפתח עזר למדינה |
| Country | Varchar(20) | שם המדינה |
| Population | Bigint | מספר תושבים במדינה |

# (20%) אפיון תהליכי ה ETL

**היערכות ואתגרים**

סדר הרצת השלבים השונים בתהליך ה-ETL:

ch

* **Mirroring-** "שיקוף המידע"-העתקה/שכפול של בסיס הנתונים התפעולי.בתחילת שלב זה, לפני ההעתקה, נרוקן את הטבלאות על מנת לא ליצור מצב של כפילות נתונים.נרוקן את הטבלאות ע"י שימוש בפקודת Truncate table.

* **Dimensions Staging-** בשלב זה נבצע טרנפורמציות לטבלאות המימד על מנת להעביר את הנתונים ממבנה טבלאי לסכמת כוכב. בשלב זה גם מוסיפים לטבלאות המימד את מפתח העזר אך בעקבות ההוראות אנו לא עושים זאת בחלק הזה. הזנת הנתונים לטבלה זו תתבצע ע"י מחיקת הנתונים הישנים שהיו בטבלת המימד ע"י שימוש בפקודת Truncate table וטעינת הנתונים החדשים ב"טעינה מלאה".
* **Dimension Loading-** העברת הטבלאות המימד החדשות מהStaging לDW. שלב זה דומה לשלב הMRR אך לעיתים יהיו שינויים שיעשו בשלב זה (למשל עדכון הvalid from אך גם עפ"י ההוראות אנו לא עושים זאת בשלב הזה). בנוסף , בשלב זה תתבצע הוספה של נתונים חדשים ועדכון הנתונים הקיימים (UPSERT).
* **Fact Staging-** בשלב זה נבצע טרנספורמציות לטבלאות העובדה. נבצע שימוש בטבלאות המקור ובטבלאות הממד שהכנו בשלב הקודם. בשלב זה מתבצעות פעולות JOIN רבות בין הטבלאות הרלוונטיות וכן פעולות אגרגציה והעמסה, לערכי השדות שהגדרנו בטבלאות הFACT על מנת להביא את אותם הטבלאות לרמת הגרעיניות המתאימה.
* **Referential Integrity-** שלב אימות ה"יושרה" של הנתונים. בשלב זה מוודאים שיש תאימות מידע בין טבלאות העובדה לנתוני הטבלאות. לדוגמא , בטבלת העובדה מסויימת אשר יש לה מפתח זר לטבלת מימד , נוודא שאכן המפתח קיים וכן עוד בדיקות נוספות. במידה ונבחין באי התאמה, נדרש לפעול לפי אחת מהשיטות שלמדנו בהרצאה – שיטת א' הוספת טבלת נדחים , ושיטה ב' הוספת רשומה לטבלת המימד עם שדה UNKNOWN.
* **Fact Loading-** שלב הוספת נתונים חדשים בטעינה אינקרמנטלית לטבלאות העובדה. בשלב זה נעביר את טבלאות העובדה שיצרנו בשלב הFact Staging למחסן הנתונים DW.

**התלויות הקיימות בין שלבי התהליך**

שלבי התהליך חייבים להתבצע בסדר המוצג. יש להגדיר את טבלאות המימד לפני טבלאות העובדה, שכן טבלאות העובדה תלויות בהן והרשומות בטבלאות העובדה מאופיינות במפתחות זרים לטבלאות המימד אליהן הן מקושרות. כמו כן, את שלב בדיקת יושרת הנתונים יש לבצע לפני טעינת הטבלאות לDW, שכן בשלב זה מוודאים שלא קיימים מזהים של משתני מימד שאינם מופיעים בטבלאות המימד שכבר עודכנו בDW.

**שלבי הETL הצפויים לקחת זמן רב במיוחד** הם השלבים הנוגעים לטבלאות העובדה- Fact Staiging וFact Loading, שכן בשלבים אלו כוללים ביצוע פעולות על היקף נתונים גדול מאוד. בשלב הFact Staging בפרט, פעולות הJOIN הרבות ופעולות האגרגציה על כמות גדולה של נתונים צפויות לקחת זמן רב במיוחד.

בנוסף, שלב הRI צפוי גם הוא לקחת זמן רב, משום שבמסגרתו עוברים על כמות גדולה מאוד של רשומות הנמצאות בטבלאות העובדה ובצעים בדיקה של יושרת הנתונים.

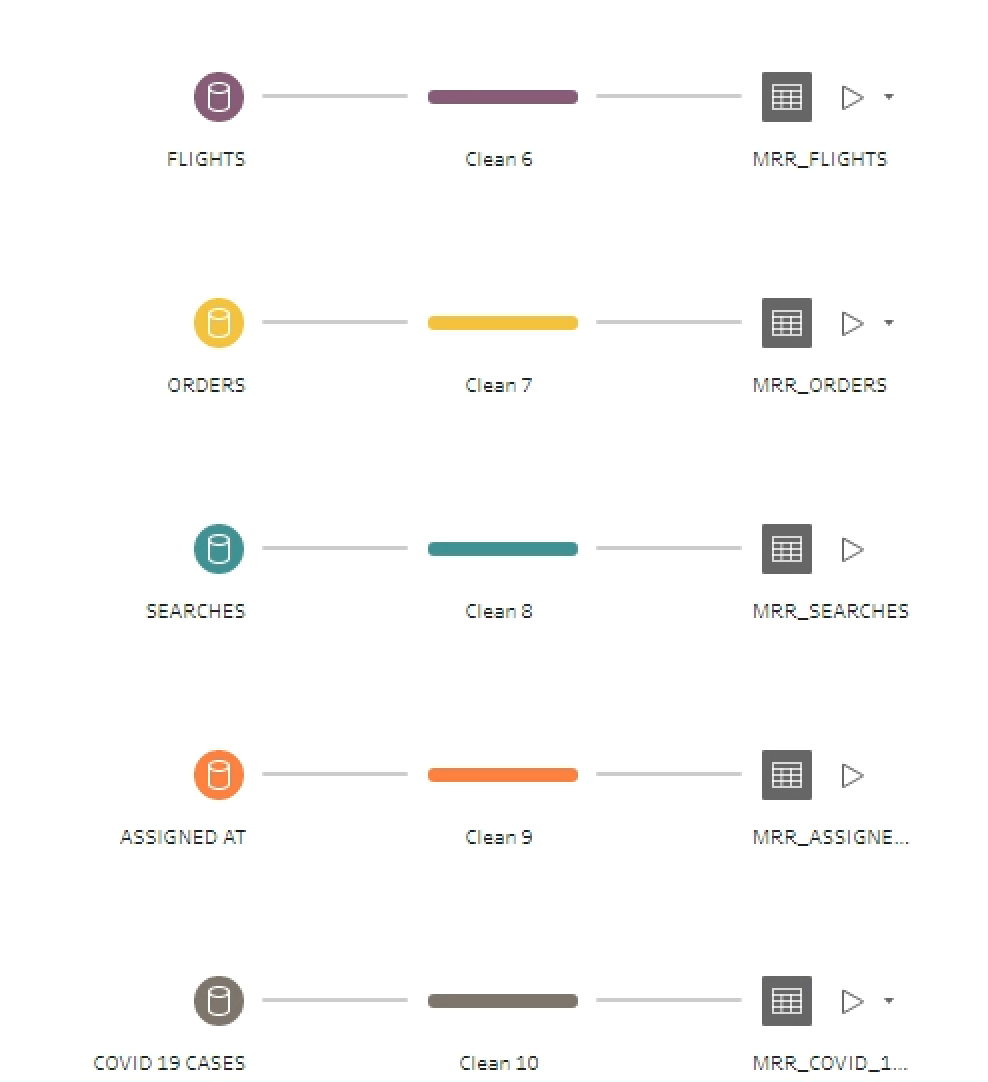
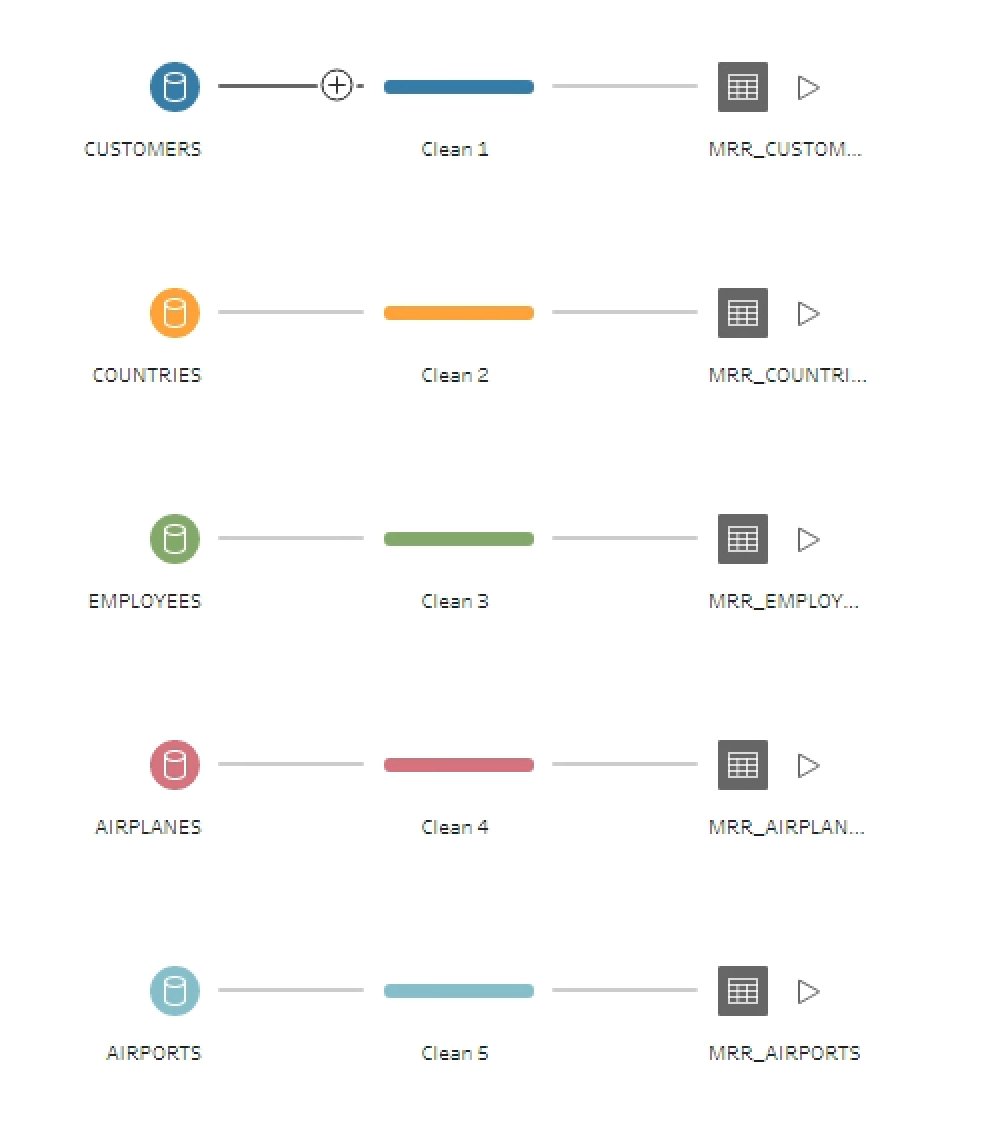
**שלבי תהליך הETL שעלולים להכשל** הם קודם כל שלב בדיקת יושרת הנתונים (RI), שכן בשלב זה עושים השוואה בין מפתחות זרים. אכיפה לא קפדנית של אילוצי המפתחות הזרים במקור הנתונים, ברשומות שמתארות טרנזקציות, עלולה להוביל לכך שבטבלאות העובדה יופיעו ערכי מפתח זר אל טבלאות המימד, לערכים שלא קיימים בטבלאות המימד. הדבר יוביל לכשלון בשלב הRI.

**ניתן להתמודד עם הבעיה הזו בשתי דרכים:**

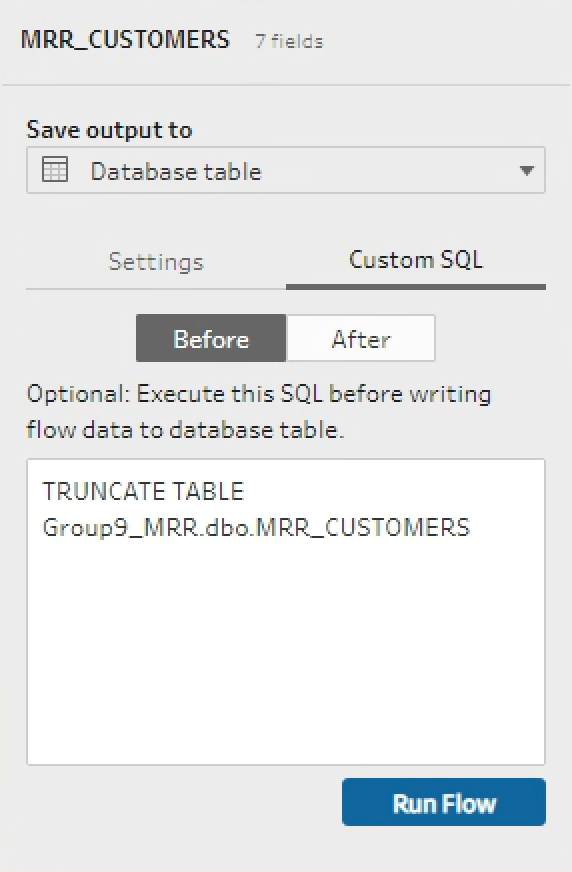
1. העברת הרשומות שהתגלה בהן כשל לטבלה נפרדת לצורך דיווח שגיאות ותחקור נוסף, ולא לטבלת הFACT.
2. בעת גילוי כשל, תתווסף לטבלת המימד רשומה מלאכותית עם הערך הבלתי מוכר. רשומת הפעולה שבה זוהה הערך הבלתי מוכר תקושר לרשומה המלאכותית ותוזן לטבלת הFACT.

בנוסף, שלב הMIRRORING עלול להכשל גם כן. בכדי למנוע את הכשל האפשרי, נבצע את שלב זה בזמנים בהם המערכת פחות עמוסה.

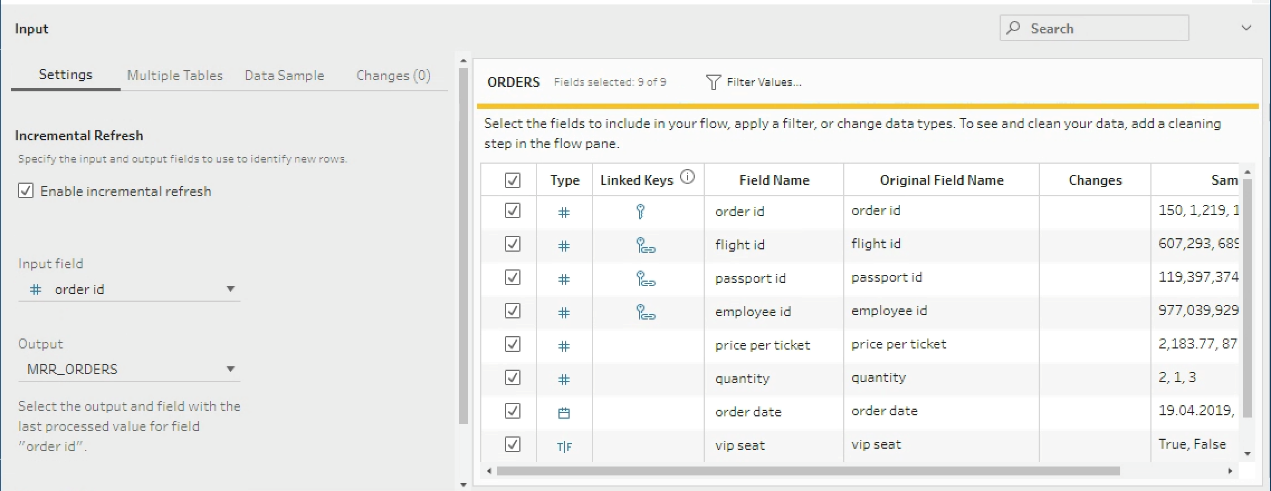
# (40%) מימוש תהליכי ה-ETL – יישומי הבסיס

**שלב 1 - MRR :** 

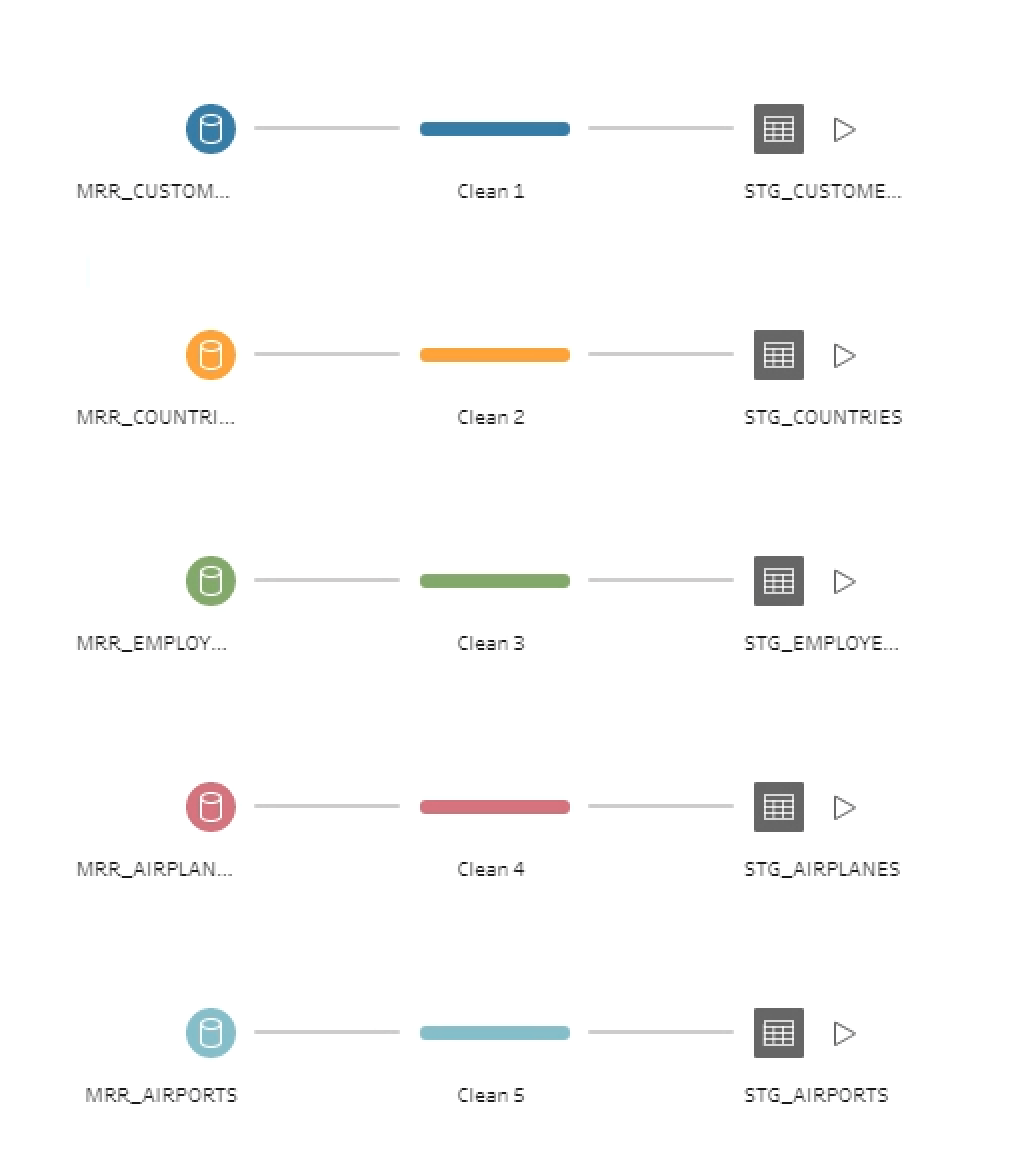
דוגמא לTRUNCATE:



דוגמא לטעינה אינקרמנטלית:



**שלב 2: Dimension Staging :**

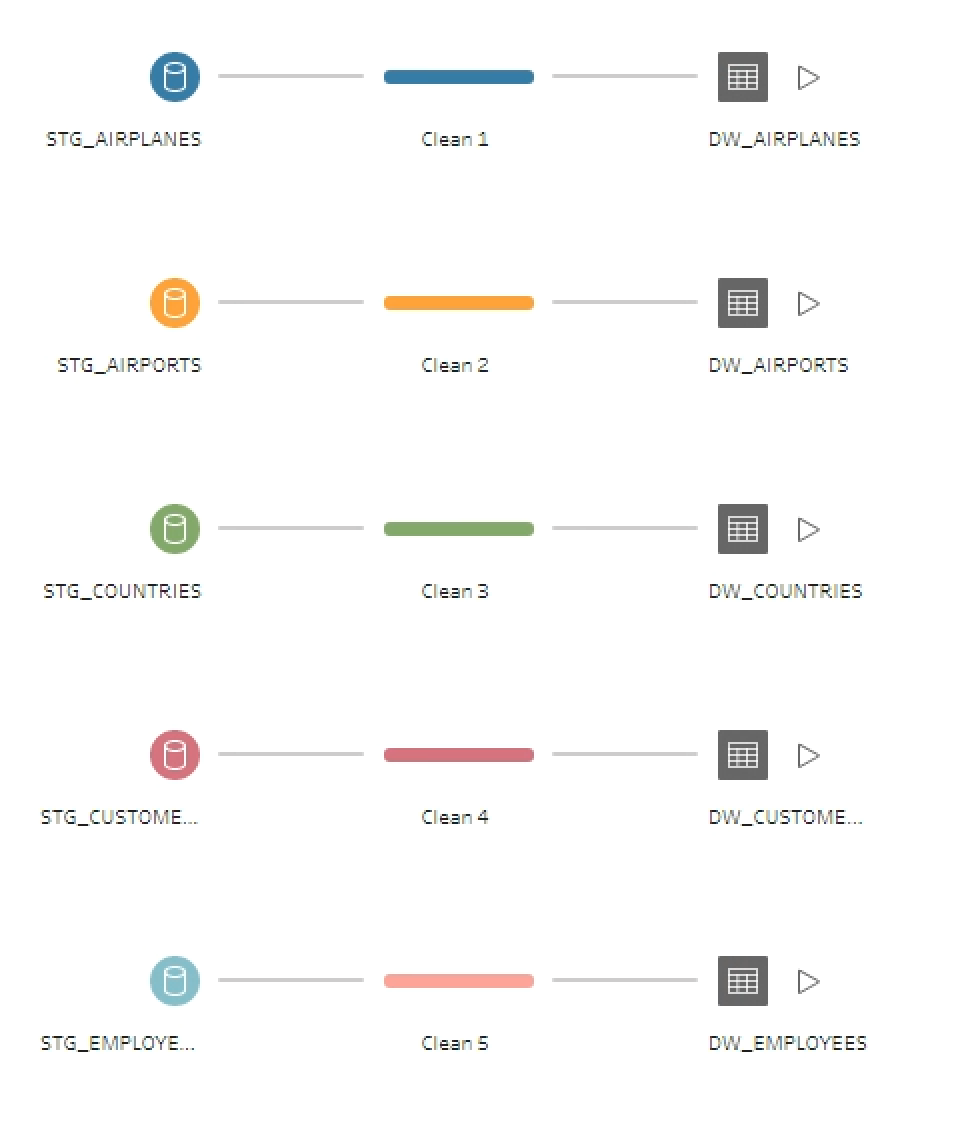


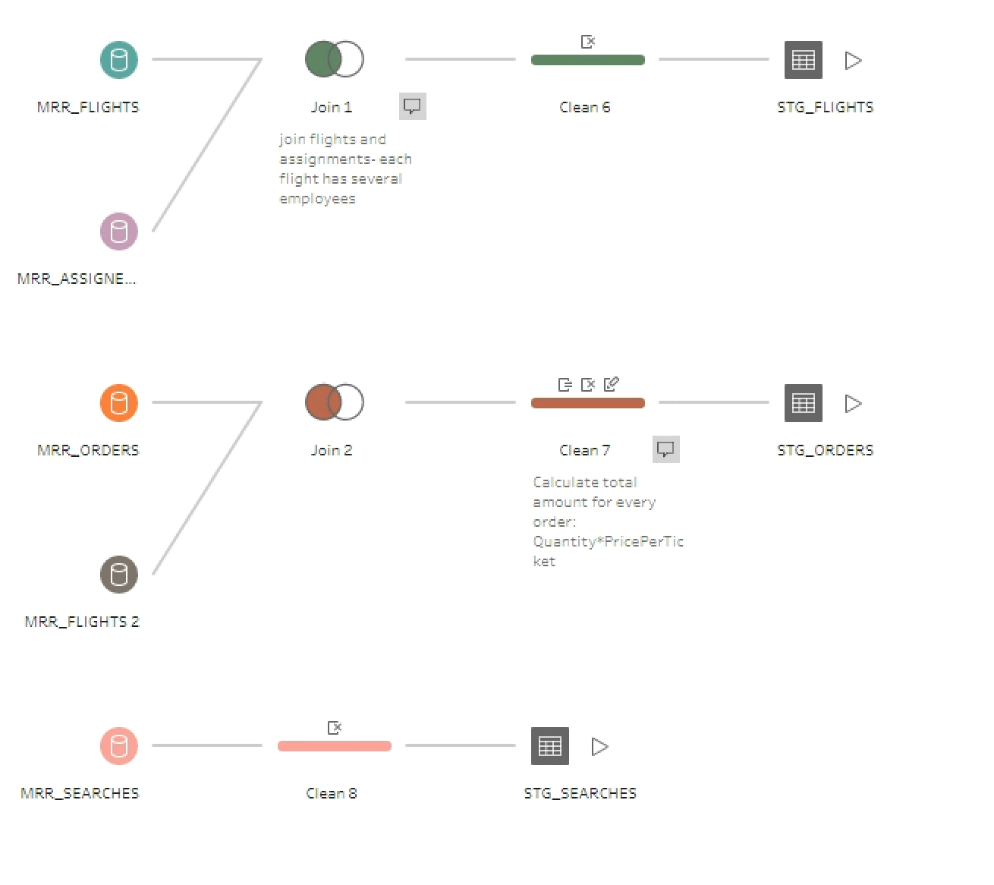
דוגמא לTRUNCATE:

תמונה שמכילה שולחן

התיאור נוצר באופן אוטומטי

**שלב 3 : Dimension Loading**



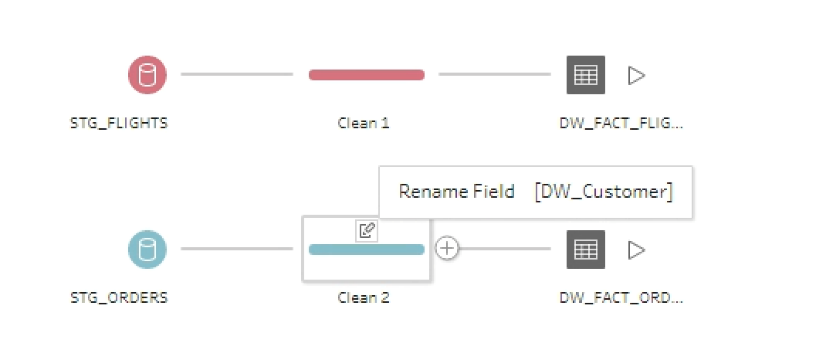
**שלב 4 – Fact Staging :**תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

**שלב 5 RI –** ביצענו במשימה 4 בדיקה בין טבלת העובדה חיפושים לבין טבלת המימד מדינות – מצב בו לקוח יכול לחפש לטוס ליעד שהחברה לא עובדת איתו , כלומר לא נמצא בטבלת המדינות ואז יווצר מצב של יושרת נתונים.

**שלב 6 Loading fact –**

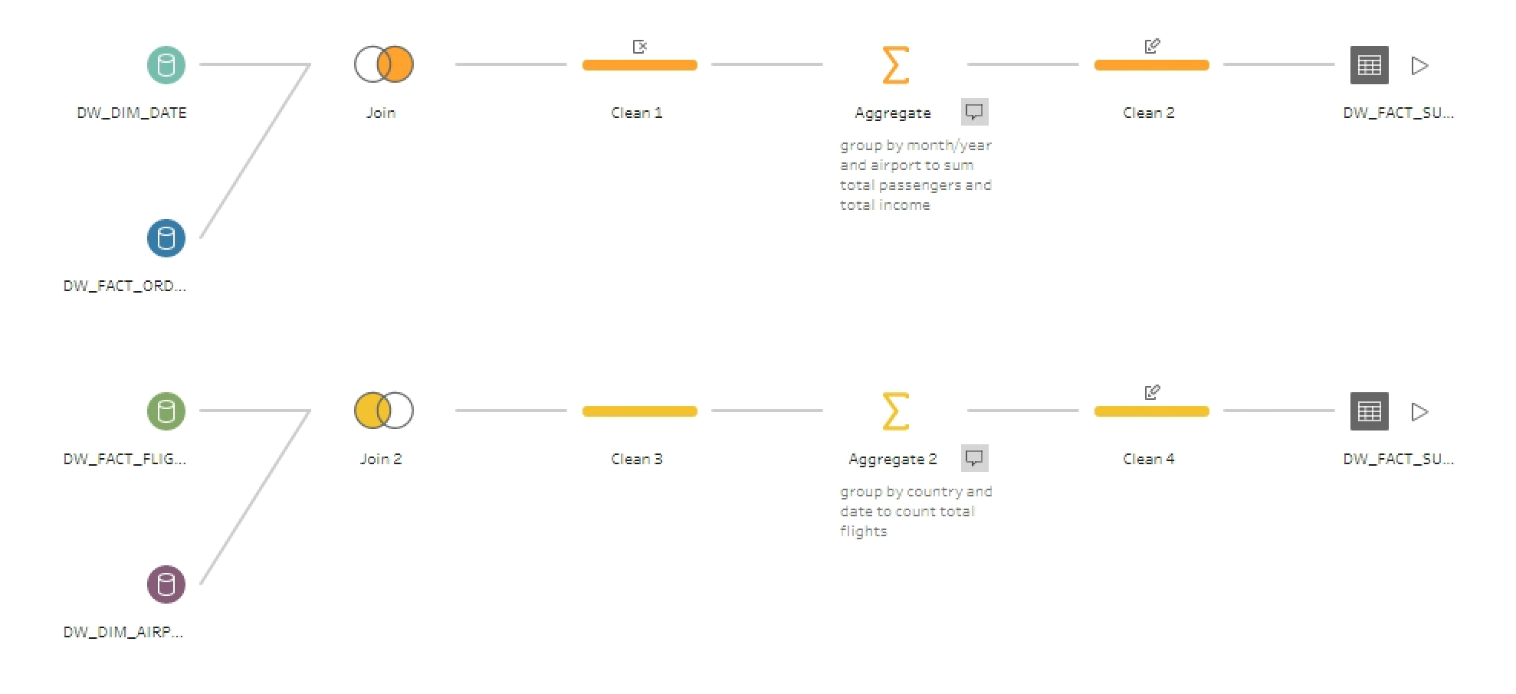
עשינו את התהליך עבור 2 טבלאות עובדה – הזמנות , טיסות . עבור טבלת העובדה חיפושים התהליך מתבצע בשלב הRI .





את הבסיס נתונים המשני נעביר ישירות לDW ללא כל שאר השלבים

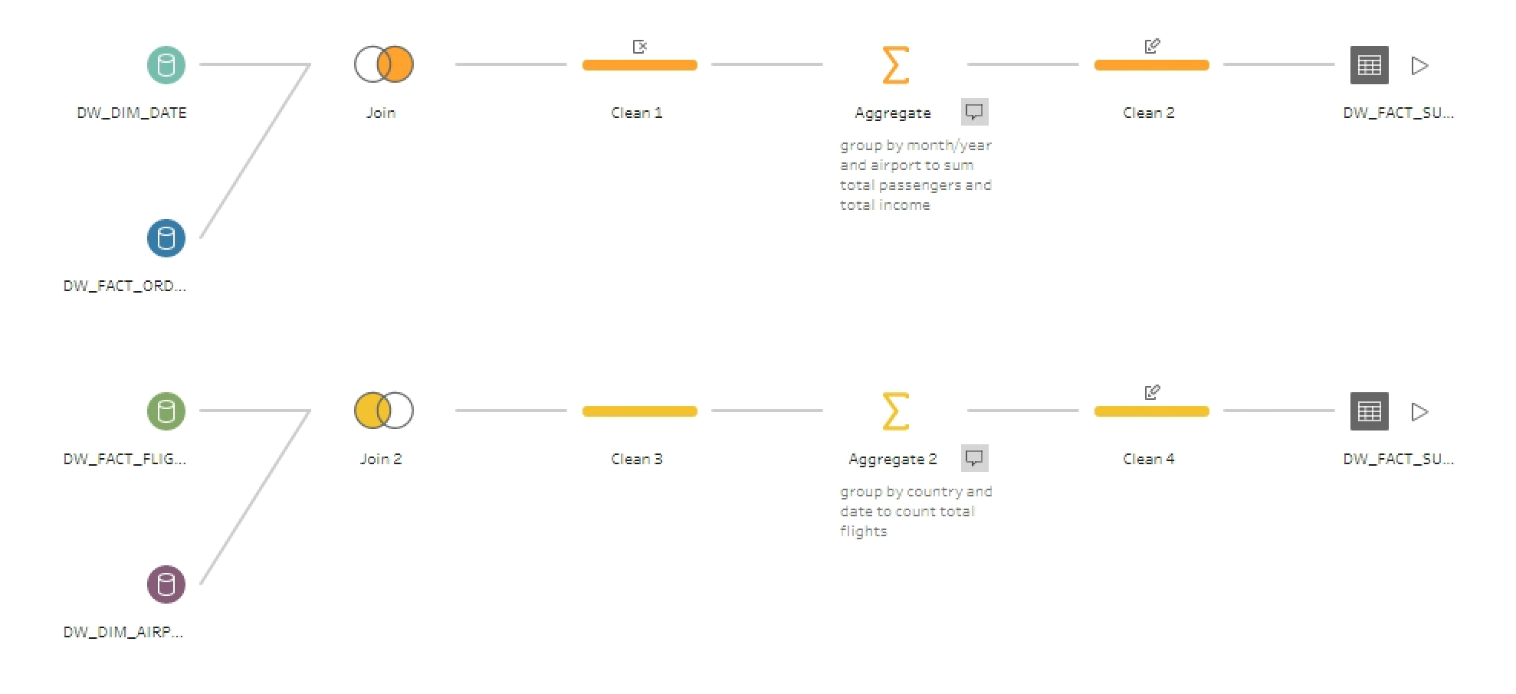
**טבלת סיכום של כמות נוסעים שנחתו בכל שדה תעופה לכל חודש:**

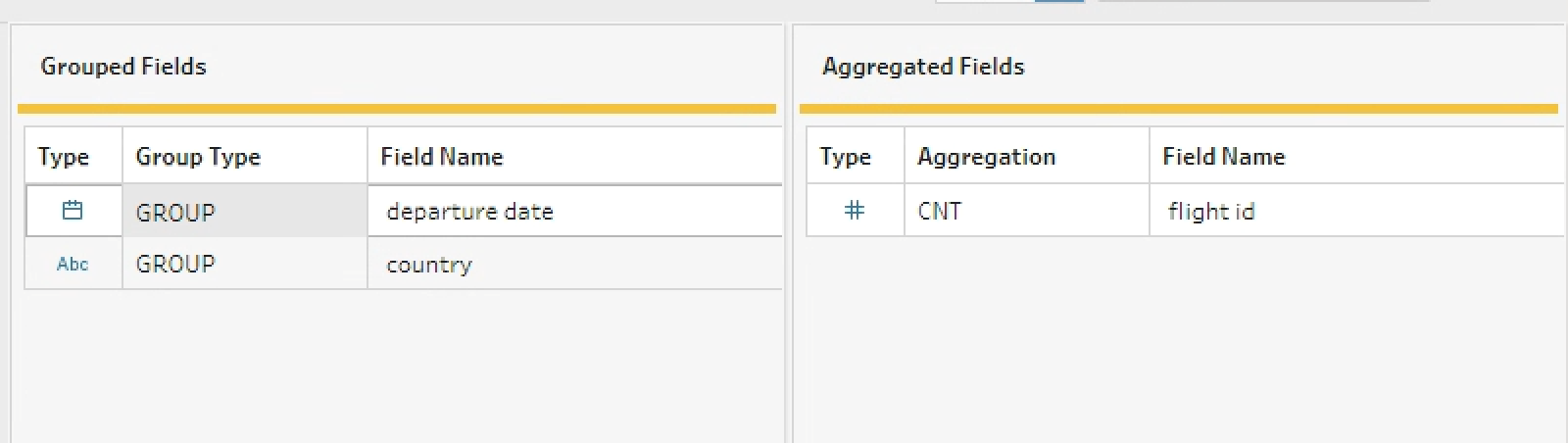


תמונה שמכילה שולחן

התיאור נוצר באופן אוטומטי

**טבלת סיכום המראה כמות טיסות יומיות אשר יוצאות מכל מדינה:**

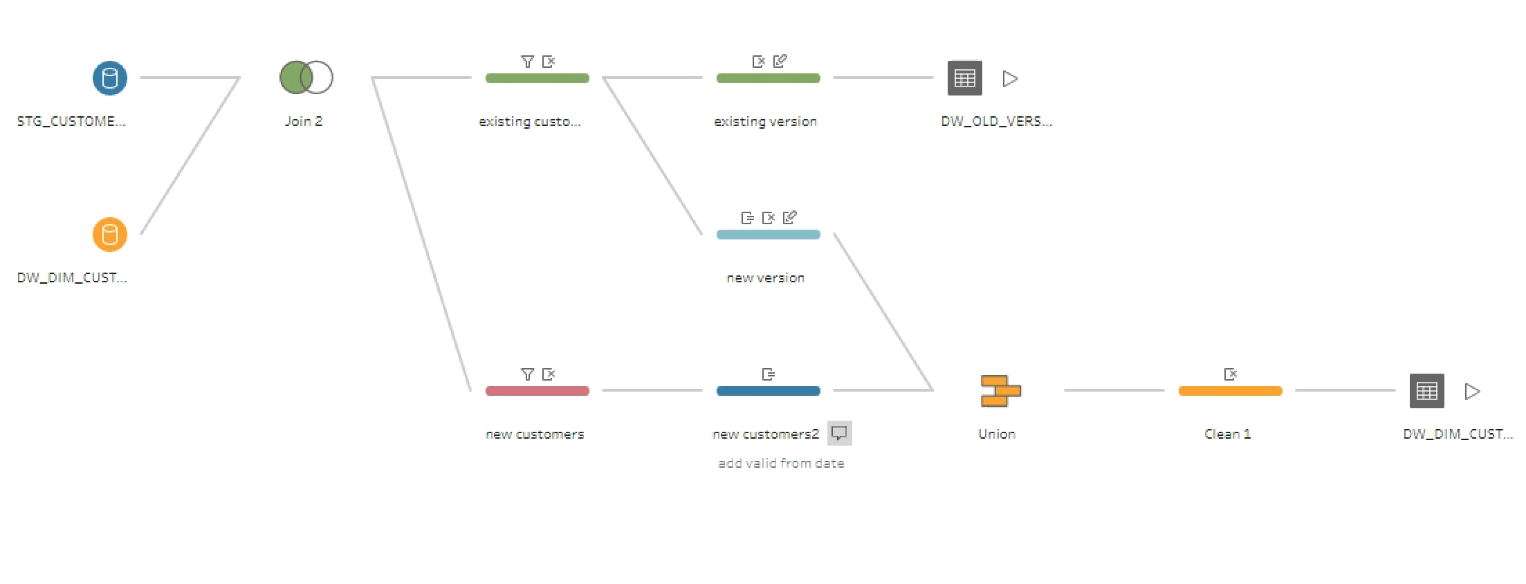




# (צבירה של עד 25%) מימוש תהליכי ה-ETL – יישומים מתקדמים

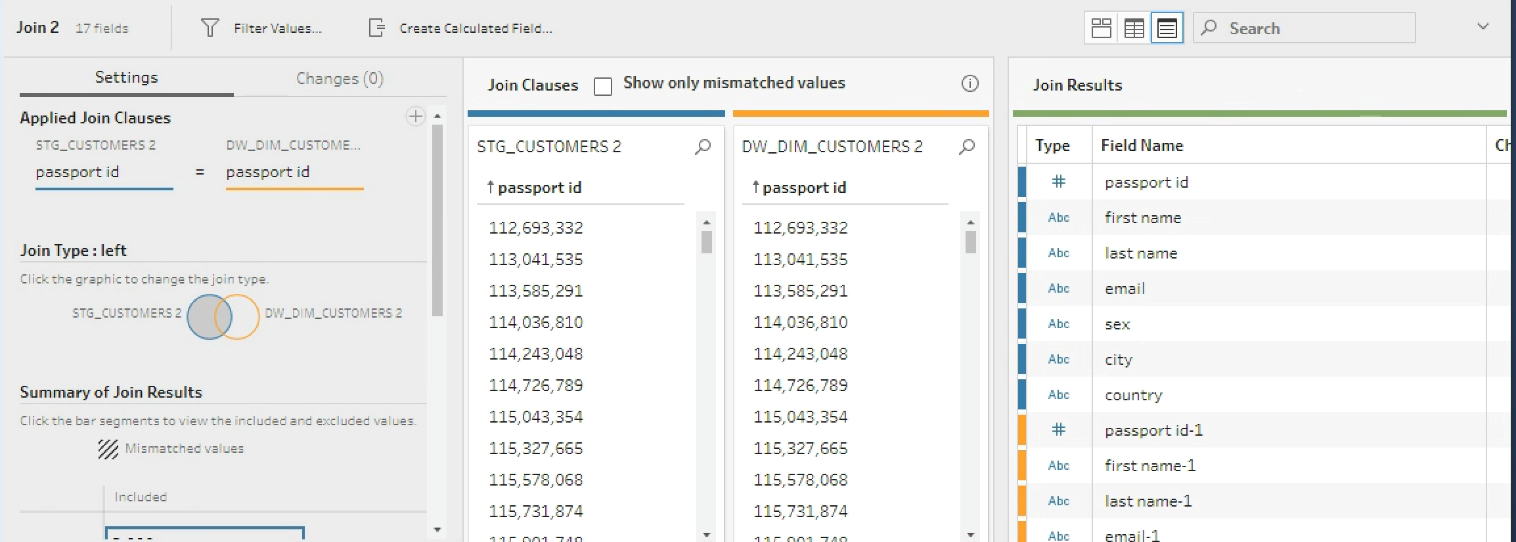
בחרנו לשנות את טבלת המימד של לקוחות לטבלת מימד משתנה מסוג 2 מאחר ובכדי להסיק מסקנות עסקיות נכונות ומדויקות המתבססות על תכונות של לקוחותינו, למשל שינוי אזור מגורים. נרצה ליצור לעיתים השוואות בין מצב נוכחי למצב קודם בכדי להבין מגמות של שינויים ולתת לך הסברים. לדוגמה: אם נבחין כי כמות גדולה מלקוחותינו הקבועים שגרים באירופה עברו לגור בארה"ב, נשקול להקטין את כמות הטיסות לאירופה ולהוסיף טיסות לארה"ב בהתאם.

על מנת ליצור את כל התהליך יצרנו טבלת חדש – טבלת גרסאות ישנה של הלקוחות אשר תגיד לנו איזה לקוחות אנו נדרשים לסגור עבורם את תאריך הגרסה.



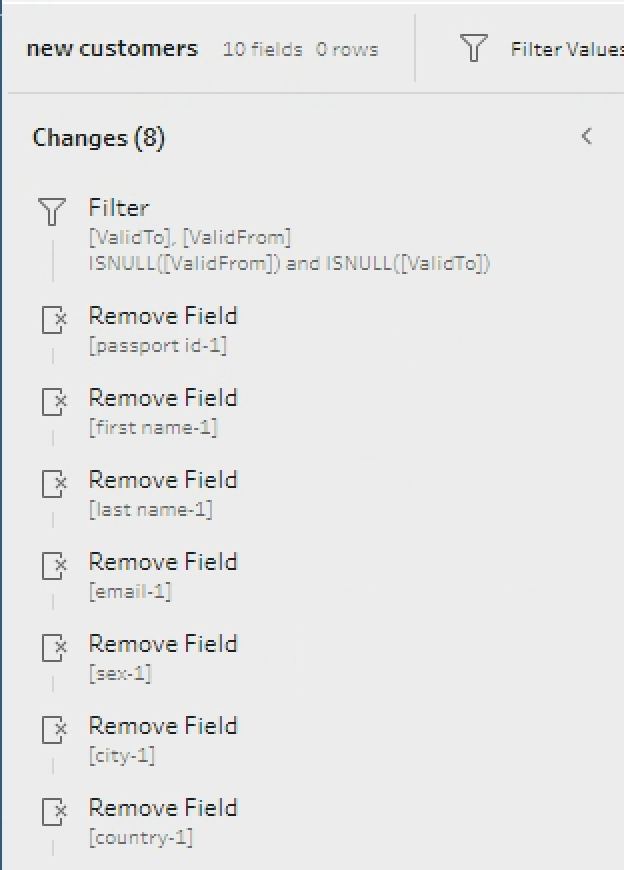
נציג את כל התהליכים שעשינו בPREP:

צעד ראשון – JOIN



צעד שני – רישום פילטרים בלקוחות החדשים ורישום פילטרים בלקוחות קיימים:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

צעד שלישי – רישום פליטרים בגרסאות חדשות של לקוחות ובגרסאות ישנות של לקוחות:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

מעבר לפילטרים הוספנו פרוצדורה אשר מעדכנת את תאריך הסיום של הגרסה רק ללקוחות אשר נמצאים בטבלת מימד לקוחות ובטבלת גרסאות ישנות של לקוחות . בסוף התהליך הפרוצדורה מוחקת מטבלת גרסאות ישנות של לקוחות את כל הרשומות המופיעות בה.

CREATE PROCEDURE updateOldVersion

as BEGIN

update DW\_DIM\_CUSTOMERS

SET ValidTo = GETDATE()-1

WHERE DW\_Customer in (Select DW\_Customer from Group9\_DW.dbo.OLD\_VERSION\_CUSTOMERS)

TRUNCATE TABLE Group9\_DW.dbo.OLD\_VERSION\_CUSTOMERS

END

קריאה לפרוצדורה מה-PREP , נשים לב כי קראנו לפרוצדורה לאחר שהFLOW מתבצע.

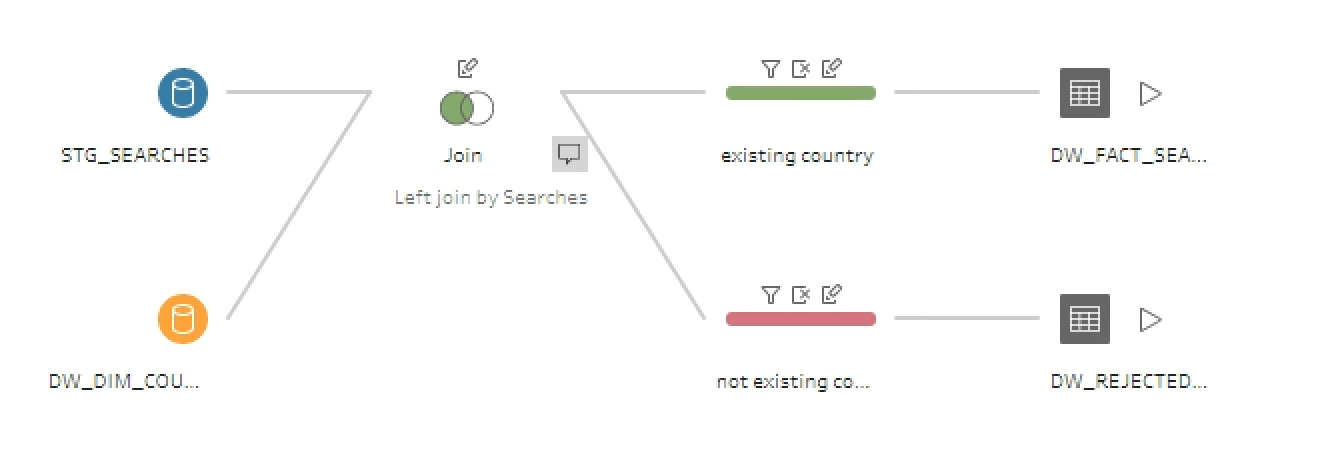
תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

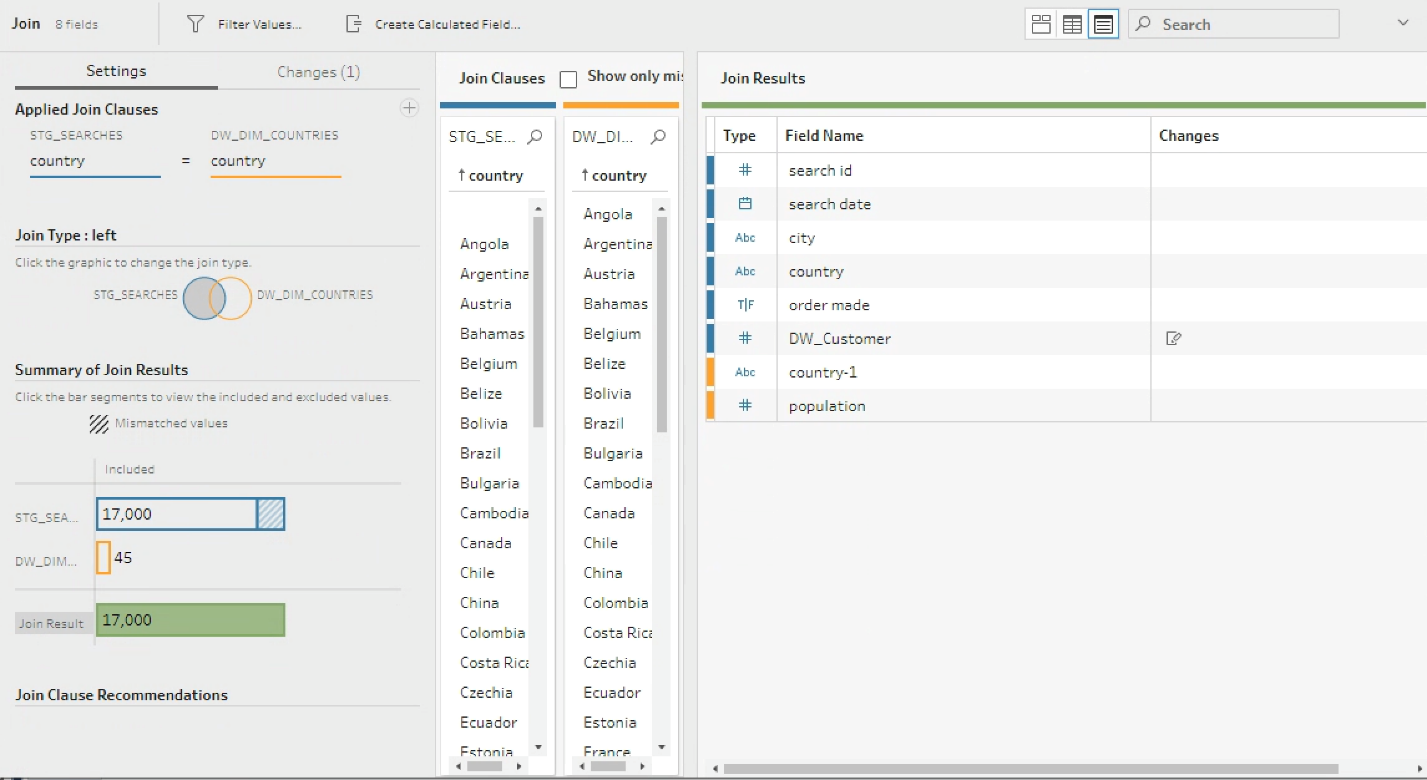
מצורף סרטון הדגמה בהמשך

**ג. (10%) בדיקת "יושרת היחס" (RI - Referential Integrity) של נתוני טבלאות העובדה**

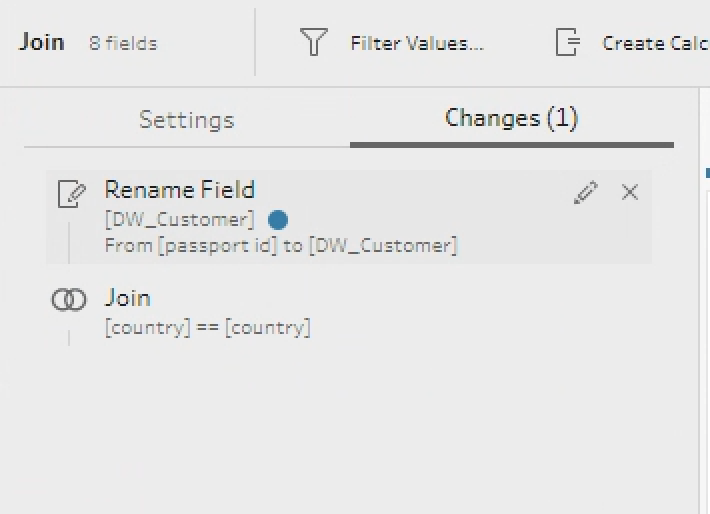
**גישה א'** – נציג רשומות שהתגלה בהן כשל RI , עמודות אלו לא יוזנו לטבלת ה-FACT, אלא יופנו לטבלה נפרדת לצורך דיווח שגיאות ותחקור נוסף. לצורך כך יצרנו טבלה חדשה של כלל הרשומות הנדחות.



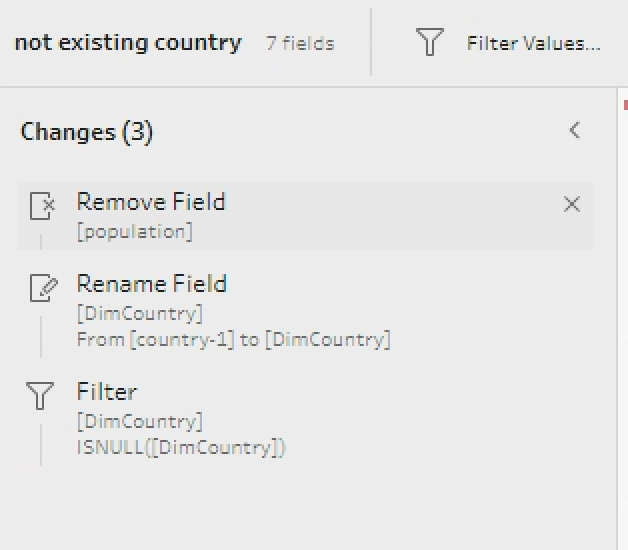
ביצענו LEFT JOIN בין הטבלאות על מנת לשמור חיפושים של לקוחות עבור מדינות שלא קיימות לנו בבסיס הנתונים – כלומר מדינות שהחברה לא עובדת עימן.



ביצענו פיטלר שמשנה את שם העמודה מpassport ID לdw\_customer על מנת לאפשר איחוד בין הטבלאות

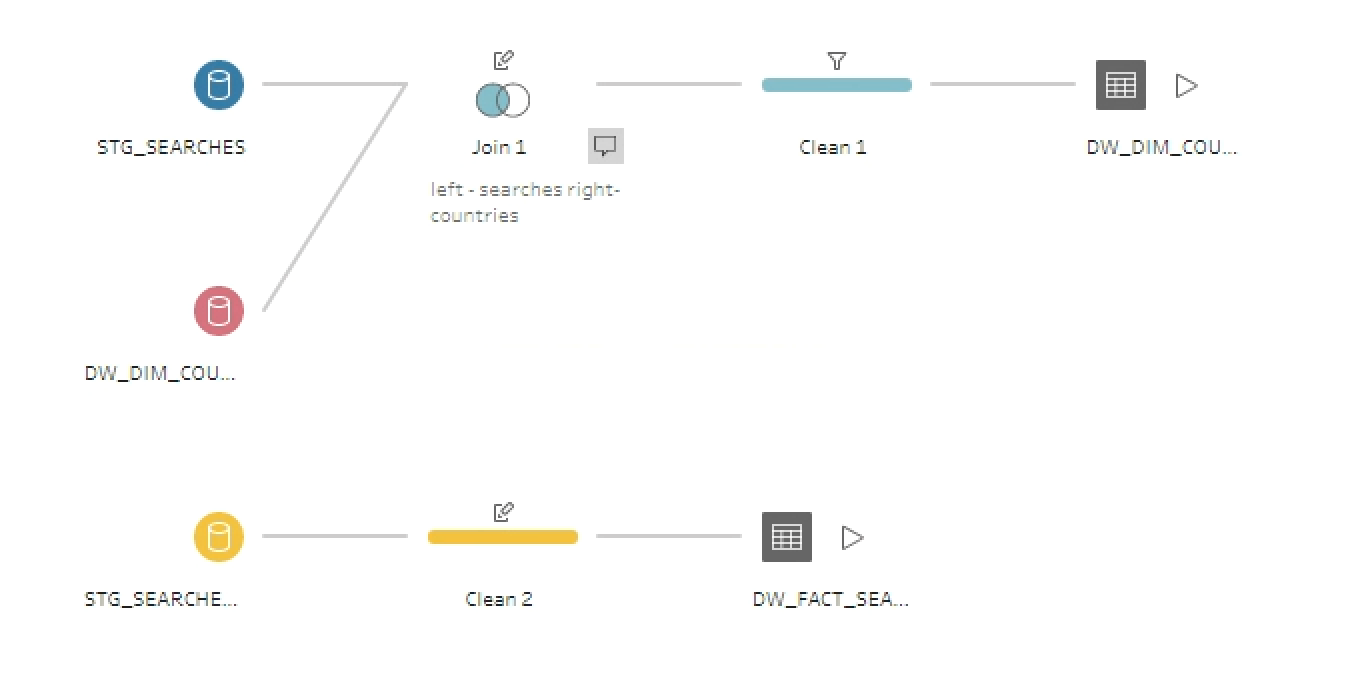


הוספה של פילטר שבודק האם השדה הוא ריק או שלא , כלומר האם המדינה קיימת בטבלת מדינות או שלא. במידה והשדה לא יהיה ריק , הרשומה תכנס לטבלת עובדה של חיפושים במידה והמדינה לא קיימת אז היא תכנס לטבלת הדחויים.

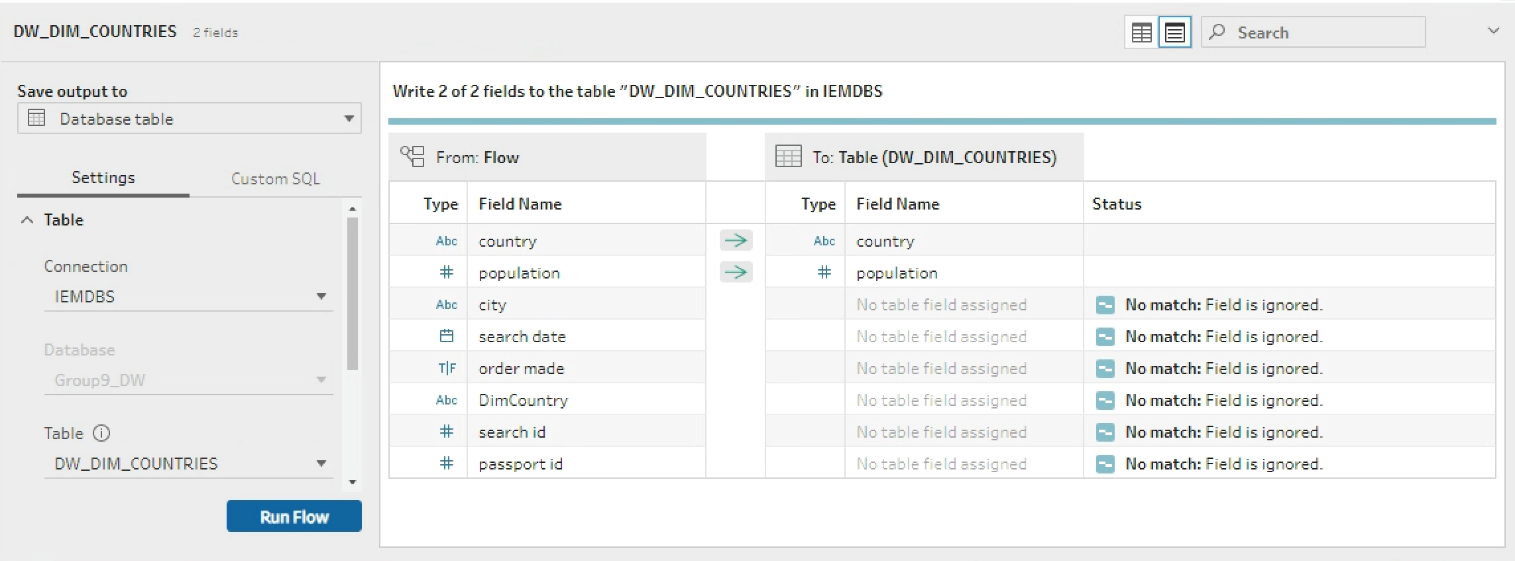
בעזרת תהליך זה , נוכל להבחין אילו מדינות יש להן חיפוש רב ואנו לא עובדים איתן , ובמידה וזה ישתלם לנו כחברה אולי נחליט לפתוח קו תעופה בין מדינת ישראל לבין מדינות אלו. מעבר לכך , טבלת הדחויים עוזרת לנו לעקוב אחר הרשומות שלא מתאימות , למצוא שגיאות במידה וקיימות ולחשוב על תהליכים עיסקיים.

**גישה ב' -** לטבלת המימד תתווסף רשומה מלאכותית, שתייצג את הערך הבלתי מוכר (עם קידוד "Unknown"). רשומת הפעולה שבה זוהה הערך הבלתי מוכר תקושר לרשומה המלאכותית, ותוזן לטבלת ה-FACT.



תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי



הסבר – כל מדינה שתחופש ולא תמצא בטבלת המדינות, תכנס לטבלת המימד של מדינות כאשר שם המדינה יהיה המדינה המחופשת והשדה של אוכלוסיה יהיה ריק , וכך נוכל לדעת על מדינה שחופשה אך לא נמצאת בבסיס הנתונים של החברה. מעבר לכך , הוספנו FLOW כך שכל רשומה של חיפוש תכנס לטבלת החיפושים גם אם המדינה לא נמצאת בטבלת מימד של המדינות.

מצורף סרטון הדגמה בהמשך

**נספחים – קטעי קוד :**

קטע קוד לחלק 1 – יצירת בסיסי הנתונים:

יצירת בסיס הנתונים התפעולי

--DROP TABLE Group9.dbo.CUSTOMERS

--TRUNCATE TABLE Group9.dbo.CUSTOMERS

CREATE TABLE Group9.dbo.CUSTOMERS(

[passport id] integer not null,

[first name] varchar(20) null,

[last name] varchar(20) null,

email varchar(60) null,

sex varchar(6) null,

city varchar(20) null,

country varchar(20) null,

constraint PK\_CUSTOMERS primary key([passport id])

)

--DROP TABLE Group9.dbo.COUNTRIES

--TRUNCATE TABLE Group9.dbo.COUNTRIES

CREATE TABLE Group9.dbo.COUNTRIES(

country varchar(20) not null,

[population] bigint null,

constraint PK\_COUNTRIES primary key(country)

)

--DROP TABLE Group9.dbo.EMPLOYEES

--TRUNCATE TABLE Group9.dbo.EMPLOYEES

CREATE TABLE Group9.dbo.EMPLOYEES(

[employee id] int not null,

[first name] varchar(20) null,

[last name] varchar(20) null,

age smallint null,

position varchar(20) null,

[start of work date] date null,

[salary per hour] money null,

[hours per month] int null,

constraint PK\_EMPLOYEES primary key([employee id])

)

--DROP TABLE Group9.dbo.AIRPLANES

--TRUNCATE TABLE Group9.dbo.AIRPLANES

CREATE TABLE Group9.dbo.AIRPLANES(

[airplane id] varchar(3) not null,

[vip capacity] smallint null,

[regular capacity] smallint null,

constraint PK\_AIRPLANES primary key([airplane id])

)

--DROP TABLE Group9.dbo.AIRPORTS

--TRUNCATE TABLE Group9.dbo.AIRPORTS

CREATE TABLE Group9.dbo.AIRPORTS(

[airport id] varchar(3) not null,

city varchar(20) null,

country varchar(20) null,

constraint PK\_AIRPORTS primary key([airport id]),

constraint FK\_AIRPORTS1 FOREIGN KEY (country) REFERENCES COUNTRIES(country)

)

--DROP TABLE Group9.dbo.FLIGHTS

--TRUNCATE TABLE Group9.dbo.FLIGHTS

CREATE TABLE Group9.dbo.FLIGHTS(

[flight id] int not null,

[airplane id] varchar(3) not null,

[from] varchar(3) not null,

[to] varchar(3) not null,

[departure date] date null,

[departure time] time null,

late bit null,

constraint PK\_FLIGHTS primary key([flight id]),

constraint FK\_FLIGHTS1 FOREIGN KEY ([from]) REFERENCES AIRPORTS([airport id]),

constraint FK\_FLIGHTS2 FOREIGN KEY ([to]) REFERENCES AIRPORTS([airport id]),

constraint FK\_FLIGHTS3 FOREIGN KEY ([airplane id]) REFERENCES AIRPLANES([airplane id])

)

--DROP TABLE Group9.dbo.ORDERS

--TRUNCATE TABLE Group9.dbo.ORDERS

CREATE TABLE Group9.dbo.ORDERS(

[order id] int not null,

[flight id] int not null,

[passport id] int not null,

[employee id] int null,

[price per ticket] money not null,

quantity int null,

[order date] date null,

[vip seat] bit null,

rating smallint null,

constraint PK\_ORDERS primary key([order id]),

constraint FK\_ORDERS1 FOREIGN KEY ([flight id]) REFERENCES FLIGHTS([flight id]),

constraint FK\_ORDERS2 FOREIGN KEY ([passport id]) REFERENCES CUSTOMERS([passport id]),

constraint FK\_ORDERS3 FOREIGN KEY ([employee id]) REFERENCES EMPLOYEES([employee id])

)

--DROP TABLE Group9.dbo.SEARCHES

--TRUNCATE TABLE Group9.dbo.SEARCHES

CREATE TABLE Group9.dbo.SEARCHES(

[search id] int not null,

[search date] date null,

city varchar(20) null,

country varchar(20) null,

[order made] bit null,

[passport id] int not null,

[order id] int null,

constraint PK\_SEARCHES primary key([search id]),

constraint FK\_SEARCHES1 FOREIGN KEY ([order id]) REFERENCES ORDERS([order id]),

constraint FK\_SEARCHES2 FOREIGN KEY ([passport id]) REFERENCES CUSTOMERS([passport id])

)

--DROP TABLE Group9.dbo.[ASSIGNED AT]

--TRUNCATE TABLE Group9.dbo.[ASSIGNED AT]

CREATE TABLE Group9.dbo.[ASSIGNED AT](

[flight id] int not null,

[employee id] int not null,

constraint PK\_ASSIGNED\_AT primary key([flight id],[employee id]),

constraint FK\_ASSIGNED\_AT1 FOREIGN KEY ([flight id]) REFERENCES FLIGHTS([flight id]),

constraint FK\_ASSIGNED\_AT2 FOREIGN KEY ([employee id]) REFERENCES EMPLOYEES([employee id])

)

--DROP TABLE Group9.dbo.[COVID 19 CASES]

--TRUNCATE TABLE Group9.dbo.[COVID 19 CASES]

CREATE TABLE Group9.dbo.[COVID 19 CASES](

id int identity(1,1),

[date] date not null,

country varchar(20) not null,

[new cases] int null,

[total cases] int null,

[new cases per million] int null,

[total cases per million] int null,

constraint pk\_covid primary key(id)

)

יצירת בסיס הנתונים של הMRR

--DROP TABLE [EL-AL MRR].dbo.MRR\_CUSTOMERS

--TRUNCATE TABLE [EL-AL MRR].dbo.MRR\_CUSTOMERS

CREATE TABLE [EL-AL MRR].dbo.MRR\_CUSTOMERS(

[passport id] integer not null,

[first name] varchar(20) null,

[last name] varchar(20) null,

email varchar(60) null,

sex varchar(6) null,

city varchar(20) null,

country varchar(20) null,

constraint PK\_CUSTOMERS primary key([passport id])

)

--DROP TABLE [EL-AL MRR].dbo.MRR\_COUNTRIES

--TRUNCATE TABLE [EL-AL MRR].dbo.MRR\_COUNTRIES

CREATE TABLE [EL-AL MRR].dbo.MRR\_COUNTRIES(

country varchar(20) not null,

[population] bigint null,

constraint PK\_COUNTRIES primary key(country)

)

--DROP TABLE [EL-AL MRR].dbo.MRR\_EMPLOYEES

--TRUNCATE TABLE [EL-AL MRR].dbo.MRR\_EMPLOYEES

CREATE TABLE [EL-AL MRR].dbo.MRR\_EMPLOYEES(

[employee id] int not null,

[first name] varchar(20) null,

[last name] varchar(20) null,

age smallint null,

position varchar(20) null,

[start of work date] date null,

[salary per hour] money null,

[hours per month] int null,

constraint PK\_EMPLOYEES primary key([employee id])

)

--DROP TABLE [EL-AL MRR].dbo.MRR\_AIRPLANES

--TRUNCATE TABLE [EL-AL MRR].dbo.MRR\_AIRPLANES

CREATE TABLE [EL-AL MRR].dbo.MRR\_AIRPLANES(

[airplane id] varchar(3) not null,

[vip capacity] smallint null,

[regular capacity] smallint null,

constraint PK\_AIRPLANES primary key([airplane id])

)

-- DROP TABLE [EL-AL MRR].dbo.MRR\_AIRPORTS

--TRUNCATE TABLE [EL-AL MRR].dbo.MRR\_AIRPORTS

CREATE TABLE [EL-AL MRR].dbo.MRR\_AIRPORTS(

[airport id] varchar(3) not null,

city varchar(20) null,

country varchar(20) null,

constraint PK\_AIRPORTS primary key([airport id]),

)

-- DROP TABLE [EL-AL MRR].dbo.MRR\_FLIGHTS

--TRUNCATE TABLE [EL-AL MRR].dbo.MRR\_FLIGHTS

CREATE TABLE [EL-AL MRR].dbo.MRR\_FLIGHTS(

[flight id] int not null,

[airplane id] varchar(3) not null,

[from] varchar(3) not null,

[to] varchar(3) not null,

[departure date] date null,

[departure time] time null,

late bit null,

constraint PK\_FLIGHTS primary key([flight id])

)

--DROP TABLE [EL-AL MRR].dbo.MRR\_ORDERS

--TRUNCATE TABLE [EL-AL MRR].dbo.MRR\_ORDERS

CREATE TABLE [EL-AL MRR].dbo.MRR\_ORDERS(

[order id] int not null,

[flight id] int not null,

[passport id] int not null,

[employee id] int null,

[price per ticket] money not null,

quantity int null,

[order date] date null,

[vip seat] bit null,

rating smallint null,

constraint PK\_ORDERS primary key([order id])

)

--DROP TABLE [EL-AL MRR].dbo.MRR\_SEARCHES

--TRUNCATE TABLE [EL-AL MRR].dbo.MRR\_SEARCHES

CREATE TABLE [EL-AL MRR].dbo.MRR\_SEARCHES(

[search id] int not null,

[search date] date null,

city varchar(20) null,

country varchar(20) null,

[order made] bit null,

[passport id] int not null,

[order id] int null,

constraint PK\_SEARCHES primary key([search id])

)

--DROP TABLE [EL-AL MRR].dbo.MRR\_[ASSIGNED AT]

--TRUNCATE TABLE [EL-AL MRR].dbo.MRR\_[ASSIGNED AT]

CREATE TABLE [EL-AL MRR].dbo.MRR\_ASSIGNED\_AT(

[flight id] int not null,

[employee id] int not null,

constraint PK\_ASSIGNED\_AT primary key([flight id],[employee id])

)

יצירת בסיס הנתונים של שלב הSTG

--DROP TABLE [EL-AL STG].dbo.STG\_CUSTOMERS

--TRUNCATE TABLE FandangoSTG.dbo.STG\_CUSTOMERS

CREATE TABLE [EL-AL STG].dbo.STG\_CUSTOMERS(

[passport id] integer,

[first name] varchar(20),

[last name] varchar(20) ,

email varchar(60) ,

sex varchar(6) ,

city varchar(20),

country varchar(20) ,

)

--DROP TABLE [EL-AL STG].dbo.COUNTRIES

--TRUNCATE TABLE [EL-AL STG].dbo.COUNTRIES

CREATE TABLE [EL-AL STG].dbo.STG\_COUNTRIES(

country varchar(20) ,

[population] bigint ,

)

--DROP TABLE [EL-AL STG].dbo.EMPLOYEES

--TRUNCATE TABLE [EL-AL STG].dbo.EMPLOYEES

CREATE TABLE [EL-AL STG].dbo.STG\_EMPLOYEES(

[employee id] int ,

[first name] varchar(20) ,

[last name] varchar(20) ,

age smallint ,

position varchar(20) ,

[start of work date] date ,

[salary per hour] money ,

[hours per month] int ,

)

--DROP TABLE [EL-AL STG].dbo.AIRPLANES

--TRUNCATE TABLE [EL-AL STG].dbo.AIRPLANES

CREATE TABLE [EL-AL STG].dbo.STG\_AIRPLANES(

[airplane id] varchar(3) ,

[vip capacity] smallint ,

[regular capacity] smallint,

)

--DROP TABLE [EL-AL STG].dbo.AIRPORTS

--TRUNCATE TABLE [EL-AL STG].dbo.AIRPORTS

CREATE TABLE [EL-AL STG].dbo.STG\_AIRPORTS(

[airport id] varchar(3) ,

city varchar(20) ,

country varchar(20) ,

)

--DROP TABLE [EL-AL STG].dbo.SEARCHES

--TRUNCATE TABLE [EL-AL STG].dbo.SEARCHES

CREATE TABLE [EL-AL STG].dbo.STG\_SEARCHES(

[search id] int ,

[search date] date ,

city varchar(20) ,

country varchar(20) ,

[order made] bit ,

[passport id] int ,

)

--DROP TABLE [EL-AL STG].dbo.FLIGHTS

--TRUNCATE TABLE [EL-AL STG].dbo.FLIGHTS

CREATE TABLE [EL-AL STG].dbo.STG\_FLIGHTS(

[flight id] int ,

[airplane id] varchar(3) ,

[from] varchar(3) ,

[to] varchar(3) ,

[departure date] date ,

[departure time] time ,

late bit ,

[employee id] int,

)

--DROP TABLE [EL-AL STG].dbo.ORDERS

--TRUNCATE TABLE [EL-AL STG].dbo.ORDERS

CREATE TABLE [EL-AL STG].dbo.STG\_ORDERS(

[order id] int ,

[passport id] int ,

[employee id] int ,

[price per ticket] money ,

quantity int ,

[order date] date ,

[vip seat] bit ,

rating smallint ,

[total amount] money,

[from airport id] varchar(3) ,

[to airport id] varchar(3),

)

יצירת בסיס הנתונים של DW

--TRUNCATE TABLE [EL-AL DW].dbo.DW\_DIM\_CUSTOMERS

CREATE TABLE [EL-AL DW].dbo.DW\_DIM\_CUSTOMERS(

[passport id] int,

[first name] varchar(20),

[last name] varchar(20) ,

email varchar(60) ,

sex varchar(6) ,

city varchar(20),

country varchar(20) ,

)

alter table [EL-AL DW].dbo.DW\_DIM\_CUSTOMERS

add DW\_Customer int identity(1,1),

ValidFrom date, ValidTo date

--TRUNCATE TABLE [EL-AL DW].dbo.DW\_DIM\_COUNTRIES

CREATE TABLE [EL-AL DW].dbo.DW\_DIM\_COUNTRIES(

country varchar(20) ,

[population] bigint ,

)

--TRUNCATE TABLE [EL-AL DW].dbo.DW\_DIM\_EMPLOYEES

CREATE TABLE [EL-AL DW].dbo.DW\_DIM\_EMPLOYEES(

[employee id] int ,

[first name] varchar(20) ,

[last name] varchar(20) ,

age smallint ,

position varchar(20) ,

[start of work date] date ,

[salary per hour] money ,

[hours per month] int ,

)

--TRUNCATE TABLE [EL-AL DW].dbo.DW\_DIM\_AIRPLANES

CREATE TABLE [EL-AL DW].dbo.DW\_DIM\_AIRPLANES(

[airplane id] varchar(3) ,

[vip capacity] smallint ,

[regular capacity] smallint,

)

--TRUNCATE TABLE [EL-AL DW].dbo.DW\_DIM\_AIRPORTS

CREATE TABLE [EL-AL DW].dbo.DW\_DIM\_AIRPORTS(

[airport id] varchar(3) ,

city varchar(20) ,

country varchar(20),

)

--drop table [EL-AL DW].dbo.DW\_DIM\_DATE

create table [EL-AL DW].dbo.DW\_DIM\_DATE(

TheDate date,

TheDay int,

TheMonth int,

TheYear int,

TheMonthYear int,

)

--drop table [EL-AL DW].dbo.DW\_DIM\_TIME

create table [EL-AL DW].dbo.DW\_DIM\_TIME(

TheTime time,

Thehour int,

TheMinutes int,

TheSecond int,

)

--TRUNCATE TABLE [EL-AL DW].dbo.DW\_FACT\_SEARCHES

CREATE TABLE [EL-AL DW].dbo.DW\_FACT\_SEARCHES(

[search id] int ,

[search date] date ,

city varchar(20) ,

country varchar(20) ,

[order made] bit ,

DW\_Customer int ,

)

--DROP TABLE [EL-AL DW].dbo.DW\_FACT\_FLIGHTS

--TRUNCATE TABLE [EL-AL DW].dbo.DW\_FACT\_FLIGHTS

CREATE TABLE [EL-AL DW].dbo.DW\_FACT\_FLIGHTS(

[flight id] int ,

[airplane id] varchar(3) ,

[from] varchar(3) ,

[to] varchar(3) ,

[departure date] date ,

[departure time] time ,

late bit ,

[employee id] int,

)

--DROP TABLE [EL-AL DW].dbo.FACT\_ORDERS

--TRUNCATE TABLE [EL-AL DW].dbo.FACT\_ORDERS

CREATE TABLE [EL-AL DW].dbo.DW\_FACT\_ORDERS(

[order id] int ,

DW\_Customer int ,

[employee id] int ,

[price per ticket] money ,

quantity int ,

[order date] date ,

[vip seat] bit ,

rating smallint ,

[total amount] money,

[from airport id] varchar(3) ,

[to airport id] varchar(3),

)

--DROP TABLE [EL-AL DW].dbo.DW\_COVID\_19\_CASES

--TRUNCATE TABLE [EL-AL DW].dbo.DW\_COVID\_19\_CASES

CREATE TABLE [EL-AL DW].dbo.DW\_COVID\_19\_CASES(

[DW Case] Int IDENTITY(1,1),

[date] date not null,

country varchar(20) not null,

[new cases] int null,

[total cases] int null,

[new cases per million] int null,

[total cases per million] int null,

)

--TRUNCATE TABLE [EL-AL DW].dbo.FACT\_SUMMARY\_MONTHLY\_PASSENGERS\_BY\_AIRPORTS

CREATE TABLE [EL-AL DW].dbo.DW\_FACT\_SUMMARY\_MONTHLY\_PASSENGERS\_BY\_AIRPORTS(

[DW monthly passanger by airport]Int IDENTITY(1,1),

[airport id] varchar(3),

[total passengers] int,

[total income] money,

MonthYear int,

)

--TRUNCATE TABLE [EL-AL DW].dbo.FACT\_SUMMARY\_DAILY\_DEPARTING\_FLIGHTS\_BY\_COUNTRY

CREATE TABLE [EL-AL DW].dbo.DW\_FACT\_SUMMARY\_DAILY\_DEPARTING\_FLIGHTS\_BY\_COUNTRY(

[DW daily flights by country] Int IDENTITY(1,1),

[date] date,

country varchar(20),

[total flights] int,

)

קטע קוד למימד תאריך

DECLARE @StartDate  date = '19700101';

DECLARE @CutoffDate date = DATEADD(DAY, -1, DATEADD(YEAR, 30, @StartDate));

;WITH seq(n) AS

(

  SELECT 0 UNION ALL SELECT n + 1 FROM seq

  WHERE n < DATEDIFF(DAY, @StartDate, @CutoffDate)

),

d(d) AS

(

  SELECT DATEADD(DAY, n, @StartDate) FROM seq

),

src AS

(

  SELECT

    TheDate         = CONVERT(date, d),

    TheDay          = DATEPART(DAY,       d),

    TheMonth        = DATEPART(MONTH,     d),

    TheYear         = DATEPART(YEAR,      d),

TheMonthYear = CAST(CONCAT(DATEPART(MONTH, d), DATEPART(YEAR, d)) as int)

  FROM d

),

dim AS

(

  SELECT

    TheDate,

    TheDay,

    TheMonth,

    TheYear,

TheMonthYear

  FROM src

)

insert into Group9\_DW.dbo.DW\_DIM\_DATE

SELECT \* FROM dim

  ORDER BY TheDate

OPTION (MAXRECURSION 0);

DECLARE @StartDate  date = '20000101';

DECLARE @CutoffDate date = DATEADD(DAY, -1, DATEADD(YEAR, 30, @StartDate));

;WITH seq(n) AS

(

  SELECT 0 UNION ALL SELECT n + 1 FROM seq

  WHERE n < DATEDIFF(DAY, @StartDate, @CutoffDate)

),

d(d) AS

(

  SELECT DATEADD(DAY, n, @StartDate) FROM seq

),

src AS

(

  SELECT

    TheDate         = CONVERT(date, d),

    TheDay          = DATEPART(DAY,       d),

    TheMonth        = DATEPART(MONTH,     d),

    TheYear         = DATEPART(YEAR,      d),

TheMonthYear = CAST(CONCAT(DATEPART(MONTH, d), DATEPART(YEAR, d)) as int)

  FROM d

),

dim AS

(

  SELECT

    TheDate,

    TheDay,

    TheMonth,

    TheYear,

TheMonthYear

  FROM src

)

insert into Group9\_DW.dbo.DW\_DIM\_DATE

SELECT \* FROM dim

  ORDER BY TheDate

OPTION (MAXRECURSION 0);

קטע קוד למימד הזמן

Declare

@StartTime time(0),

@EndTime time(0),

@Time time(0),

@Interval int

Set @StartTime = '00:00:00'

Set @EndTime = '23:59:59'

Set @Time = @StartTime

Set @Interval = 1

WHILE @Time <= @EndTime

BEGIN

    IF NOT EXISTS ( SELECT 'X' from Group9\_DW.dbo.DW\_DIM\_TIME (NOLOCK) where TheTime = @Time )

    BEGIN

    INSERT Into Group9\_DW.dbo.DW\_DIM\_TIME with(rowlock)

    Values

    (

    @Time,

    datename(hh,@Time),

    datename(minute,@time),

datename(second,@time)

    )

    END

    Select @Time = Dateadd(ss,1, @Time)

    Select @Interval = @Interval + 1

END

קטע קוד לחלק 4 – משימה 1 : יצירת טבלאות המדגימות את המשתנה מימד מסוג 2:

--TRUNCATE TABLE Group9\_DW.dbo.OLD\_VERSION\_CUSTOMERS

CREATE TABLE Group9\_DW.dbo.OLD\_VERSION\_CUSTOMERS (

DW\_Customer int IDENTITY(1,1),

[passport id] int,

[first name] varchar(20),

[last name] varchar(20) ,

email varchar(60) ,

sex varchar(6) ,

city varchar(20),

country varchar(20),

ValidFrom date,

ValidTo date

)

קטע קוד ליצירת טבלת נדחים – חלק 4 משימה 3 :

--TRUNCATE TABLE Group9\_DW.dbo.REJECTED\_SEARCHES

CREATE TABLE Group9\_DW.dbo.REJECTED\_SEARCHES

(

DWRejectedSearch int identity(1,1),

[search id] int,

[search date] date,

city varchar(20),

country varchar(20),

[order made] bit,

[passport id] int,

)