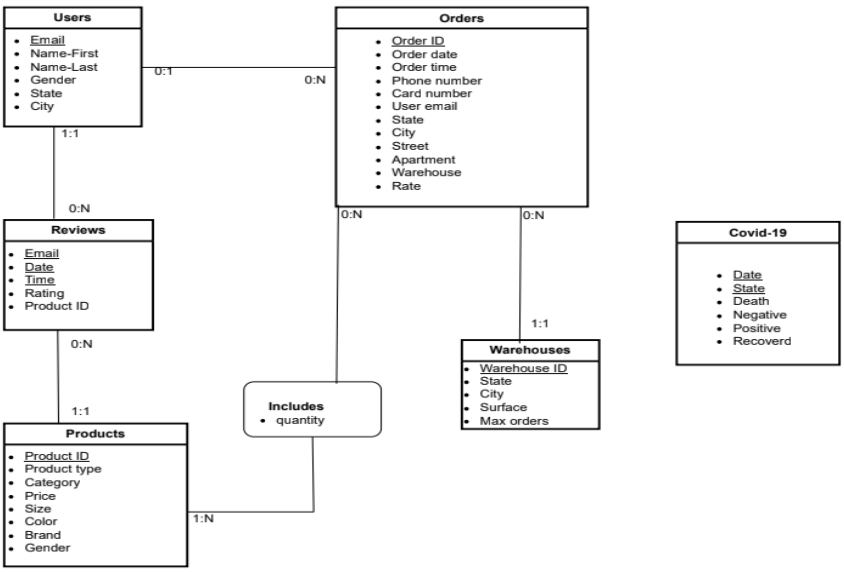
מס' קבוצה	שם הפרוייקט	תאריך הגשה
5	skate.com	15/09/2022
מספרי תעודות הזהות של המגישים		
314974155	318356995	316089200

חלק ב' – הכנת תשתית הנתונים

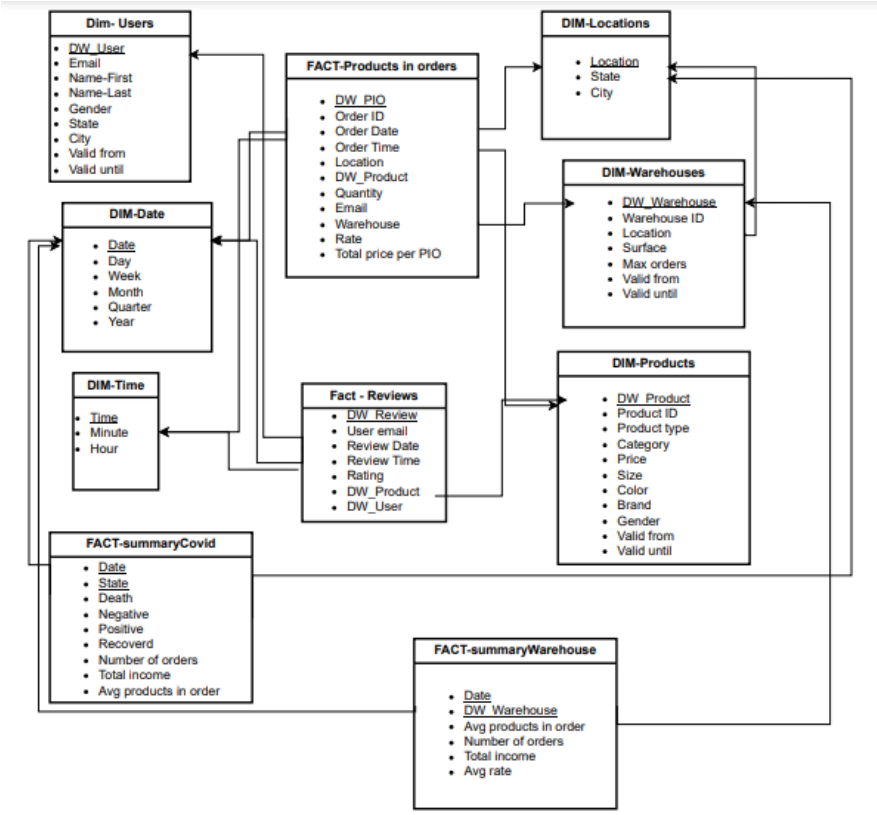
1. יישום מקורות הנתונים ומחסן הנתונים הטבלאי



ERD מתוקן:

מקור הנתונים הראשי: בטבלת Users הוספנו את השדות Gender, State, וCity.
מקור הנתונים המשני: לא בוצעו שינויים.

סכמת כוכב מתוקנת:



בסכמת הכוכב הורדנו את טבלת Fact-Covid-19. הנתונים מטבלה זו יכנסו לטבלת העובדה הסיכומית Fact-SummaryCovid. ובהתאם לכך הוספנו את השדות Death, Negative, וRecoverd.
בנוסף בטבלת Dim_Users הוספנו את השדות Gender, State, וCity.

הטבלה	שם השדה	סוג הנתונים	תיאור השדה
DIM-Users המשתמשים הרשומים באתר. טבלת מימד משתנה לאט מסוג 2, כיוון שישנם שדות שעלולים להשתנות (אימייל, שם פרטי ושם משפחה) ונרצה לשמור גם נתונים ישנים אודות המשתמשים.	DW_User	Int	מזהה משתמש. מפתח עזר, כל ערך משקף גרסה של משתמש לפי טווח תאריכים מוגדר.
	Email	Varchar(50)	מזהה המשתמש המקורי. האימייל שהמשתמש נרשם איתו לאתר
	Name-First	Varchar(20)	השם הפרטי של המשתמש
	Name-Last	Varchar(20)	שם המשפחה של המשתמש
	Gender	Varchar(10)	מין המשתמש
	State	Varchar(20)	מדינת מגורים
	City	Varchar(20)	עיר מגורים
	Valid From	Date	תאריך תחילת תוקף הגרסה
	Valid Until	Date	תאריך סיום תוקף הגרסה
FACT-summaryCovid טבלת עובדת סיכומית המחברת בין מקור הנתונים הראשי והמשני ובה מוצגים חישובים סיכומיים אודות ההזמנות בכל תאריך לפי מדינה.	Date	Date	תאריך הסיכום
	State	Varchar(20)	המדינה שעליה מתבצע הסיכום
	Death	Int	מספר המתים בתאריך ובמדינה הנתונים
	Negative	Int	מספר השליליים בתאריך ובמדינה הנתונים
	Positives	Int	מייצג את מספר התושבים שהתגלו כחיוביים במדינה ובתאריך הנתונים.
	Recovered	Int	מספר המחלימים בתאריך ובמדינה הנתונים
	Number of orders	Int	שדה סיכומי – מייצג את מספר ההזמנות שבוצעו במדינה ובתאריך הנתונים.
	Total income	Decimal(10, 2)	שדה סיכומי – סך ההכנסות מהזמנות במדינה ובתאריך
	Avg products in order	Decimal(10, 2)	שדה מחושב – ממוצע כמות המוצרים להזמנה במדינה ובתאריך

2. אפיון תהליכי ה ETL

שלבי ה-ETL:

1. Data Mirroring – בשלב זה מבצעים שכפול של בסיס הנתונים התפעולי. ניצור העתק של הטבלאות מבסיס הנתונים המקורי ומבנה הטבלאות יהיה זהה למבנה המקורי שלהן. מטרת השלב היא לעבוד על הנתונים העדכניים ביותר מבלי להעמיס על בסיס הנתונים המקורי ומבלי לבצע בו שינויים. שלב זה נבצע ריקון לטבלאות. את טבלאות העובדה נטען בצורה אינקרימנטלית ואת טבלאות המימד נטען בצורה מלאה.

2. Dimension Staging – יצירת שלד לטבלאות המימד. בשלב זה נעביר את המבנה הטבלאי לסכמת כוכב על ידי ביצוע טרנספורמציות ושבירת הנורמליזציה. את טבלאות המימד נטען לפי סדר היראכי אשר לא יפגע בתלויות השונות בין המימדים. הסדר שבו טענו את המימדים הוא: משתמשים, מוצרים, מיקומים, מחסנים. לפני השיקוף, נרוקן את הטבלאות מהנתונים הקודמים על מנת שנקבל את הגרסאות המעודכנות ביותר ונבצע טעינה מלאה.

3. Dimensions Warehousing – לאחר העברת הטבלאות לצורת סכמת הכוכב, נבצע טעינה של טבלאות המימדים משלב ה-STG לתוך מחסן הנתונים. עבור המימדים המשתנים לאט מסוג 2, מחסן הנתונים יכיל את הגרסאות השונות.

4. Fact Staging – בשלב זה נבצע טרנספורמציות על טבלאות העובדה ושילוב שלהן עם טבלאות המימד על מנת ליצור את טבלאות העובדה כפי שהגדרנו בסכמת הכוכב. בשלב זה, ניצור את השדות המחושבים הרלוונטיים לטבלאות העובדה על מנת שנוכל להפיק מידע שיסייע לנו לקבלת החלטות ולתחקור. בשלב זה נרוקן את הטבלאות מהנתונים הקודמים ונבצע טעינה אינרקימנטלית.

5. Referential Integrity – בשלב זה מאמתים את המידע בטבלאות העובדה מול המידע שמכילות טבלאות המימד על מנת לוודא שהמידע שיוזן לטבלאות העובדה אכן קיים בטבלאות המימד. במידה ולא, נצטרך להתמודד עם בעיית יושרת היחס בעזרת אחת משתי הדרכים שלמדנו בכיתה.

6. Fact Warehousing – ביצוע עדכון של כל טבלאות העובדה אל מחסן הנתונים על ידי העתקת המידע משלב ה-STG לשלב ה-DW. בשלב זה נרוקן את הטבלאות מהנתונים הקודמים ונבצע טעינה אינרקימנטלית.

תלויות קיימות בין שלבי התהליך

בין חלק מטבלאות המימד קיים קשר היראכי (למשל טבלת משתמשים ומיקומים, טבלת מחסנים ומיקומים). מאחר וקיימות תלויות אלה, יש משמעות לסדר בו נטען את טבלאות המימד על מנת להגיע למחסן נתונים תקין. בנוסף, עלינו לטעון את טבלאות המימד לפני טעינת טבלאות העובדה, מאחר ובטבלאות העובדה קיימים מפתחות זרים אל טבלאות המימד.

שלבי ETL הצפויים לקחת זמן רב

שלב 2 (dimension staging) עשוי לקחת זמן רב מאחר ואנו מרוקנים את הטבלאות ומבצעים טעינה מלאה של טבלאות המימד (ולא טעינה אינקרימנטלית), ובמידה וקיים בהן מספר גדול של רשומות, שלב זה עשוי לקחת זמן רב.

שלבים שעלולים להיכשל בתהליך

ייתכן ובעקבות כשלים של יושרת היחס, חלק משלבי תהליך הטעינה עשויים להיכשל. למשל במקרה בו נרצה להכניס רשומה לטבלת עובדה, שערך המפתח לטבלת המימד שלה אינו קיים. על מנת להתמודד עם כשלים אלו, עלינו לטפל במקרים של יושרת היחס באחת הדרכים שלמדנו בכיתה. בנוסף, בשלב ה-Fact Staging מבצעים איחוד של טבלאות המימד עם טבלאות העובדה ומכאן עשויים להיווצר בעיות של ספירה כפולה והטיית נתונים ועלינו לוודא את נכונות המידע המחושב.

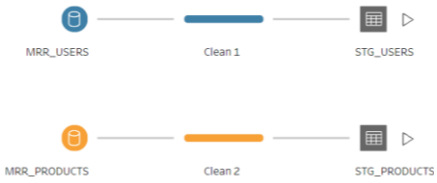
3. מימוש תהליכי ה-ETL – יישומי הבסיס

שלב 1 – Mirroring

בשלב זה מתבצע שכפול של בסיס הנתונים התפעולי אל הטבלאות המתאימות בבסיס הנתונים MRR. בטבלאות Users, Products, Warehouses בוצעה טעינה מלאה, ובטבלאות Orders, Quantities, Reviews, Covid-19 בוצעה טעינה אינקרמנטלית, מאחר וטבלאות אלו מכילות כמויות גבוהות של נתונים ומתעדכנות באופן תדיר, ובנוסף בטבלאות אלו לא צפויים שינויים ברשומות לאחר שאלה הוזנו, ולכן נרצה להימנע מטעינה מלאה שלהן. על מנת לאפשר טעינה אינקרמנטלית בטבלאות אלו, הוספנו שדה IncrementalNum ועל פיו בוצעה הטעינה האינקרמנטלית. טרם ההזנה בוצע ריקון של הטבלאות ב-MRR.

שלב 2 – Dimensions Staging

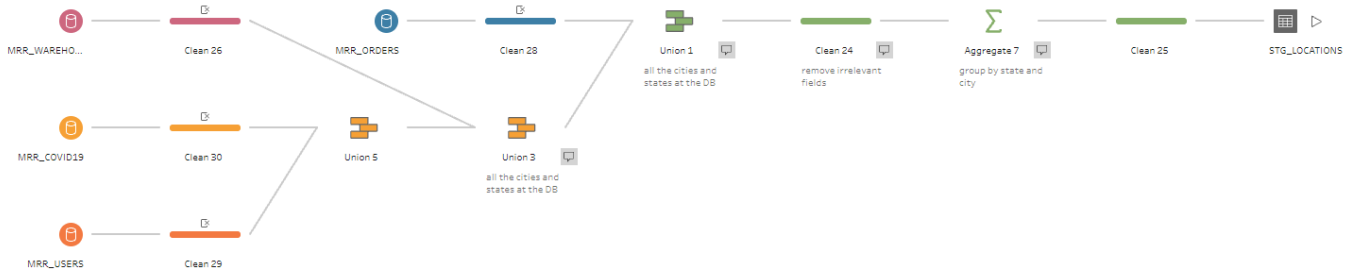
STG_Users, STG_Products



הטבלאות MRR_Users, MRR_Products הוזנו לתוך הטבלאות המתאימות בבסיס הנתונים STG ללא כל שינוי, וטרם ההזנה בוצע ריקון של הטבלאות ב-STG.

STG_Locations

על מנת ליצור את טבלת STG_Locations, ביצענו איחוד בין טבלאות בסיס הנתונים המכילות מיקומים (Orders, Users, Warehouses, Covid19) ובכך קיבלנו את כלל הצירופים של הערים והמדינות שמופיעים בבסיס הנתונים. על מנת לקבל רשומה יחידה מכל צירוף כזה, ביצענו group by לפי עיר ומדינה. טרם ההזנה בוצע ריקון של הטבלה ב-STG.

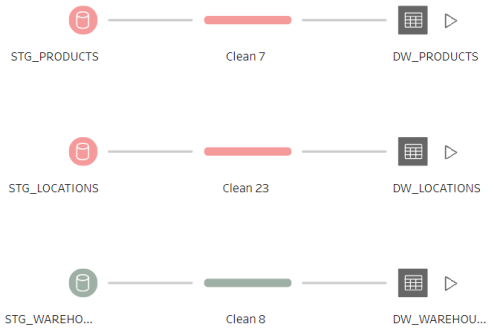


STG_Warehouses

על מנת ליצור את טבלת STG_Warehouses, ביצענו Join בין טבלת Warehouses בבסיס הנתונים MRR לבין טבלת Locations, וזאת כדי להכניס לתוך הטבלה החדשה את פרטי המחסן יחד עם ערך המפתח הזר של טבלת המיקומים. טרם ההזנה בוצע ריקון של הטבלה ב-STG.

שלב 3 – Dimensions Warehousing

DW_Products, DW_Locations, DW_Warehouses



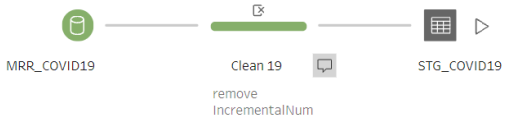
על מנת להכניס את טבלאות המימד אל מחסן הנתונים, העתקנו את טבלאות STG_Products, STG_Locations, STG_Warehouses לתוך הטבלאות המתאימות במחסן הנתונים DW ללא כל שינוי.

DW_Users

מומש כמימד משתנה לאט מסוג 2 ולכן נראה את ההכנסה שלו למחסן הנתונים בהמשך הדו"ח בחלק של היישומים המתקדמים.

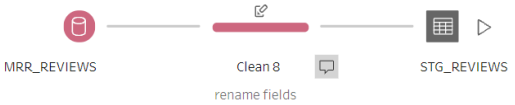
שלב 4 – Fact Staging

STG_Covid19



מחקנו את שדה העזר IncrementalNum מטבלת MRR_Covid והעברנו לטבלה המתאימה בבסיס הנתונים STG. טרם ההזנה בוצע ריקון של הטבלה ב-STG.

STG_Reviews



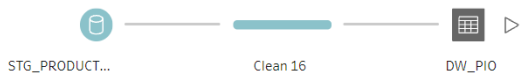
שינינו את שמות השדות ProductID ו-IncrementalNum ל- DW_Product, DW_Review בהתאמה והעברנו לטבלה המתאימה בבסיס הנתונים STG. טרם ההזנה בוצע ריקון של הטבלה ב-STG.

STG_Products_In_Orders

תחילה, מחקנו שדות מיותרים מטבלת Products בבסיס הנתונים MRR וביצענו Join בין טבלה זו לטבלת MRR_Quantities על השדה ProductID על מנת לקבל נתונים אודות המוצרים שנקנו בהזמנות. לאחר מכן, מחקנו שדות מיותרים מטבלת MRR_Orders וביצענו Join בינה לבין טבלת Locations על השדות עיר ומדינה בכדי לקבל את מזהה המיקום אליו נשלחה כל הזמנה. לבסוף, ביצענו Join בין המוצרים בהזמנה לבין ההזמנות ומיקומיהן על בסיס המזהה OrderID, מחקנו שדות מיותרים, שינינו שמות של שדות, חישבנו את השדה המחושב Total_Price_Per_PIO (מחיר כולל עבור המוצר בהזמנה) והעברנו לטבלת Products_In_Orders בבסיס הנתונים STG. טרם ההזנה בוצע ריקון של הטבלה ב-STG.

שלב 5 – Fact Warehousing

DW_Products_In_Orders



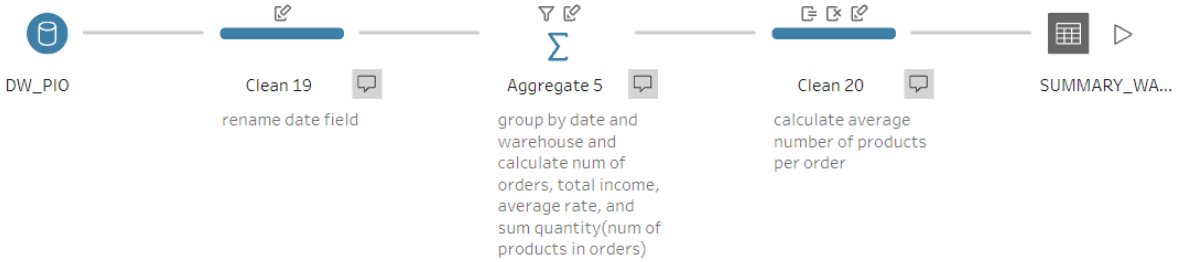
בשלב זה העברנו את טבלאות העובדה המסודרות מבסיס הנתונים STG לבסיס הנתונים DW ללא כל שינוי.

DW_Reviews

כחלק ממימוש המשתמשים כמימד משתנה מסוג 2, טבלת עובדה זו הושפעה ולכן נראה את המימוש של הכנסת טבלת Reviews למחסן הנתונים בחלק של היישומים המתקדמים.

Summary_Warehouses

ליצירת הטבלה הסיכומית לפי תאריך ומחסן, השתמשנו מטבלת המוצרים בהזמנות במחסן הנתונים וביצענו אגרגציה לפי תאריך ומחסן. עבור צירוף של תאריך ומחסן, חישבנו את מספר ההזמנות, את ההכנסה הכוללת ממכירת מוצרים, את הדירוג הממוצע עבור ההזמנות (רק עבור הזמנות שניתנה להן ביקורת כדירוג), ואת ממוצע מספר המוצרים בהזמנה (ממוצע מוצרים בהזמנה חושב על ידי סכימה של כמות המוצרים שנמכרו בהזמנות וחלוקה במספר ההזמנות). את התוצאות שהתקבלו הזנו לטבלה הסיכומית במחסן הנתונים.



Summary_Covid19

ליצירת הטבלה הסיכומית לפי תאריך ומדינה, ביצענו Join בין טבלאות המוצרים בהזמנות לבין טבלת המיקומים ממחסן הנתונים לפי השדה Location על מנת לקבל את המיקום אליו נשלחה כל הזמנה. לאחר מכן, בוצע Join בין הטבלה שהתקבלה לטבלת Covid19 מבסיס הנתונים STG לפי תאריך ומדינה, כדי לקבל את נתוני הקורונה המתאימים. מחקנו שדות מיותרים וביצענו אגרגציה לפי תאריך ומדינה. עבור צירוף של תאריך ומדינה, חישבנו את מספר ההזמנות, את ההכנסה הכוללת ממכירת מוצרים, ואת ממוצע מספר המוצרים בהזמנה (ממוצע מוצרים בהזמנה חושב על ידי סכימה של כמות המוצרים שנמכרו בהזמנות וחלוקה במספר ההזמנות). את התוצאות שהתקבלו הזנו לטבלה הסיכומית במחסן הנתונים.

4. מימוש תהליכי ה-ETL – יישומים מתקדמים

א. מימד משתנה לאט מסוג 2

בחרנו לממש את טבלת משתמשים כמימד משתנה לאט מסוג 2 מאחר ומשתמשי האתר עשויים לשנות את פרטיהם האישיים ולעדכן אותם, למשל שינוי שם משפחה בעקבות נישואין או שינוי עיר ומדינה בעקבות מעבר דירה.

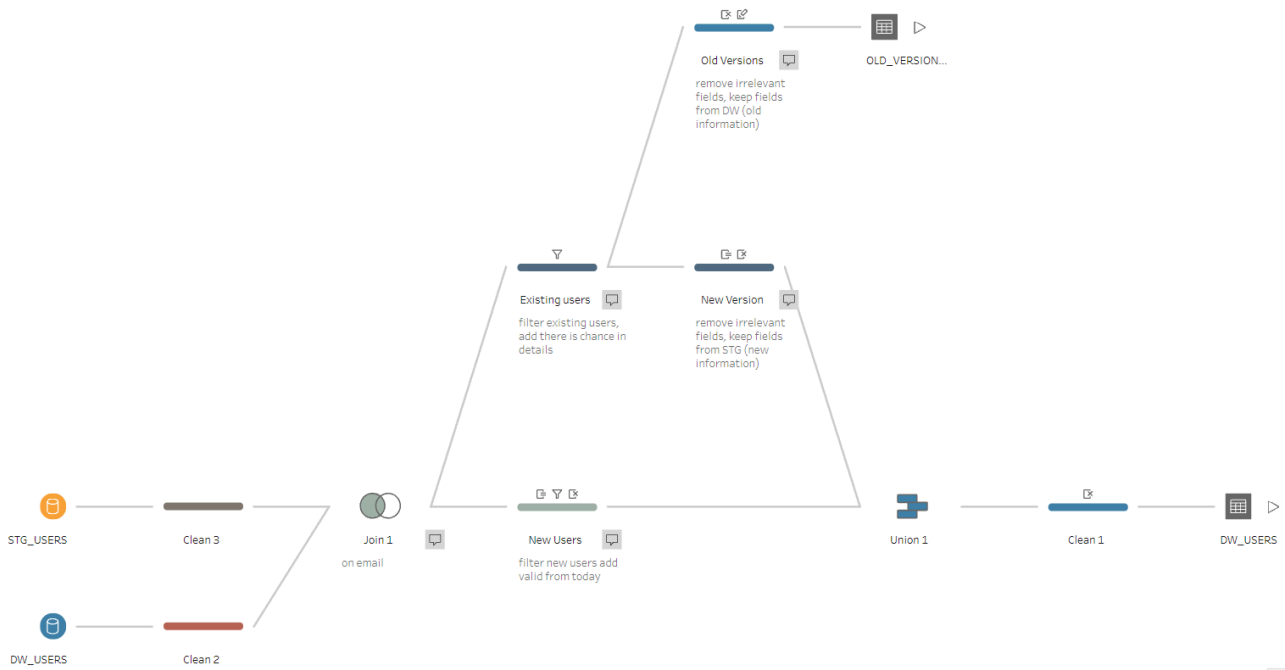
עדכון הטבלה כמימד משתנה לאט מסוג 2 בוצע בשלב שבין ה-STG לבין ה-DW. תחילה ביצענו Left Join בין טבלת המשתמשים ב-STG לבין טבלת המשתמשים ב-DW על מנת לבדוק האם המשתמשים ב-STG הם משתמשים ישנים שהתעדכנו (שדה DW_User בעל ערך שונה מ-NULL ושינוי בפרטיהם האישיים), או משתמשים חדשים שנוספו (שדה DW_User בעל ערך NULL).

עבור המשתמשים החדשים שנוספו, הוספנו להם תאריך Valid From בעל התאריך של היום ו-Valid Until בעל ערך NULL.

עבור המשתמשים הקיימים:

אם בוצע שינוי בשדות העיר, המדינה או שם המשפחה, יצרנו גרסה חדשה עם תאריך Valid From של היום, והעברנו אותם לטבלת Old Versions שמפעילה פרוצדורה שמורה לעדכון תאריך ה-Valid Until שלהם להיות התאריך של היום, על מנת לסגור את הגרסה הקודמת שלהם. בנוסף, נפתחת גרסה חדשה עם הפרטים המעודכנים כאשר השדה Valid From מקבל את התאריך של היום והשדה Valid Until הוא NULL.

לבסוף, בוצע איחוד בין המשתמשים הקיימים והמשתמשים החדשים והם הוכנסו לטבלת המשתמשים במחסן הנתונים. טרם ההכנסה למחסן הנתונים, מתבצע ריקון של טבלת DW_Reviews.

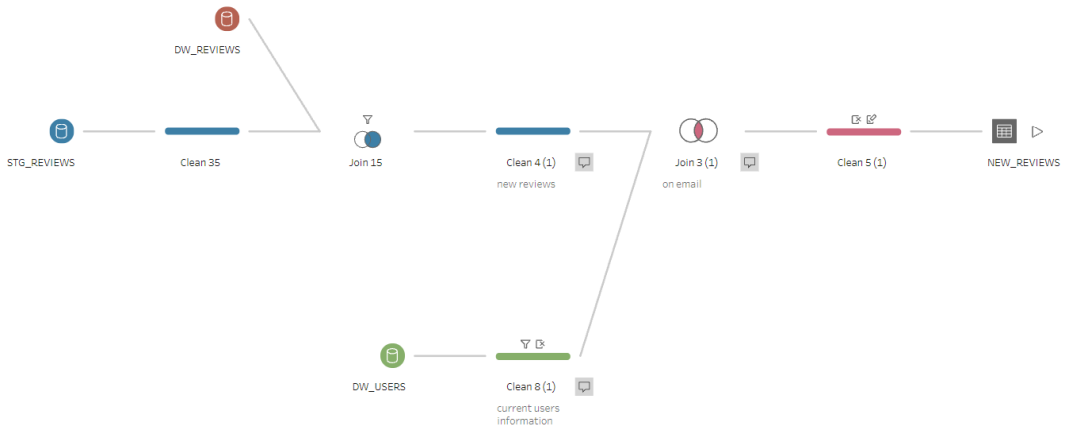


הפרוצדורה השמורה שמעדכנת את תאריך ה-Valid Until של המשתמשים בטבלת הגרסאות הישנות להיות התאריך של היום:

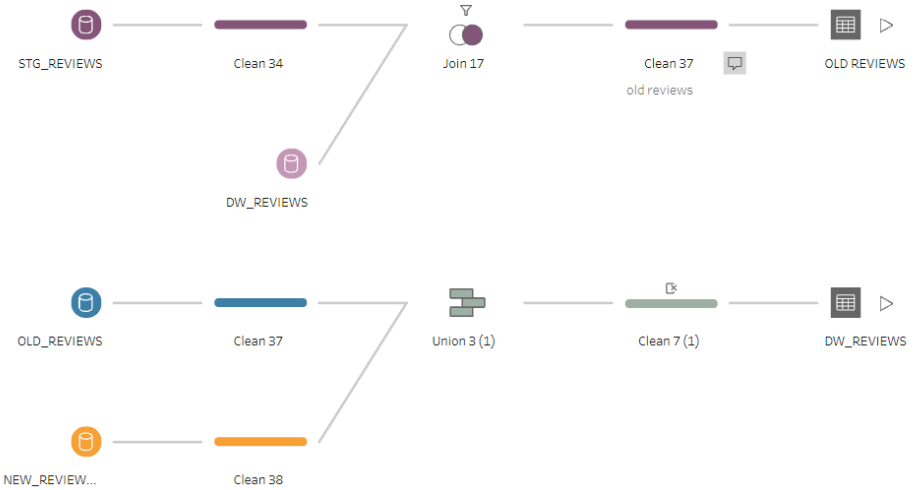
```
CREATE PROCEDURE [dbo].[updateOldUserVersions]
AS BEGIN
UPDATE DW_DIM_USERS SET ValidUntil = GETDATE() WHERE DW_User IN (SELECT DW_User FROM
OLD_VERSIONS_USERS)
TRUNCATE TABLE OLD_VERSION_JEWELS
END
```

בעקבות מימוש טבלת משתמשים כמימד משתנה לאט מסוג 2, יש להתאים את טבלת העובדה Reviews בהתאם.

תחילה, סיננו את הביקורות החדשות שנוספו וטרם קיימות במחסן הנתונים ואיחדנו אותם עם פרטי המשמשים המעודכנים על מנת להתאים את הגרסה העדכנית ביותר של המשתמשים, ואלו הוכנסו לטבלת New Reviews.



ביצענו Right Join בינן לבין הביקורות במחסן הנתונים על מנת לזהות את הביקורות הישנות שכבר קיימות שם, ואותן הכנסנו לטבלת Old Reviews. לבסוף, רוקנו את טבלת ביקורות במחסן הנתונים, ביצענו איחוד בין הביקורות החדשות לישנות והכנסנו אותן לטבלת Reviews במחסן הנתונים.

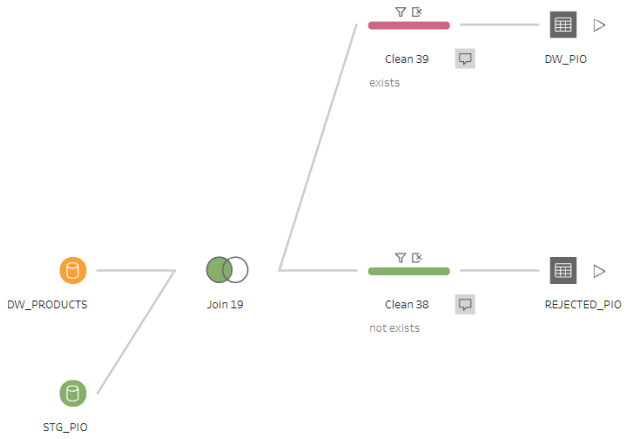


ג. בדיקת "יושרת היחס" (RI - Referential Integrity) של נתוני טבלאות העובדה

- <u>גישה א': הוספת רשומת עובדה עם כשל לטבלת העובדה שמציינת Rejected:</u>

ביצענו Left Join בין טבלת המוצרים במחסן הנתונים לבין טבלת המוצרים בהזמנה בבסיס הנתונים STG על מנת לזהות הזמנות בהן הוכנס ערך לא תקין של מוצר. ביצענו סינון לפי הערך של ProductID ובדיקה האם הוא

NULL על מנת לבדוק האם המוצר קיים בבסיס הנתונים או לא. במידה ולא קיים (כלומר הערך שלו הוא NULL), נכניס את הרשומות לטבלת Rejected לצורך תחקור מאחר ומדובר בכשל של יושרת היחס. במידה והוא לא NULL, אין כשל RI ולכן נכניס את הרשומות לטבלת המוצרים בהזמנות במחסן הנתונים.



פלט הרשומות הבעייתיות שנכנסו לטבלת Rejected:

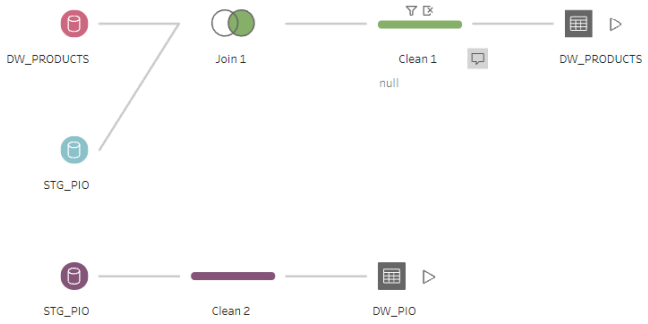
```
SELECT *
FROM Group5_DW.dbo.REJECTED_PIO
```

100 %

Results | Messages

	DW_PIO	Orde...	Order_Date	Order_Time	Locati...	DW_Prod...	Quant...	Email	DW_Wareho...	R...	Total_Price_Per_...	Price
1	1	8	2018-01-09	02:40:55.0000000	145	NULL	9	urna.et@icloud.couk	40	5	NULL	NULL
2	2	9	2018-01-09	20:19:28.0000000	38	NULL	9	donec.tempor@google.org	2	3	NULL	NULL
3	3	50	2018-02-07	00:59:03.0000000	14	NULL	9	interdum.enim.non@protonmail.net	50	5	NULL	NULL

- <u>גישה ב': הוספת רשומת ממד עם קידוד Unknown, והוספת רשמות עובדה בהתאם לטבלה הפרטנית:</u>



ביצענו Right Join בין טבלת המוצרים במחסן הנתונים לבין טבלת המוצרים בהזמנה בבסיס הנתונים STG. לאחר מכן בוצע סינון על שדה ProductID ובדיקה האם הוא NULL. עבור הרשומות בהן התקבל שערך זה הוא NULL, נכניס רשומה מלאכותית לטבלת המוצרים במחסן הנתונים בעלת NULL על כלל השדות, ולאחר מכן הכנסנו את הרשומות שזוהה בהן הערך הלא מוכר NULL לטבלת המוצרים בהזמנות בבסיס הנתונים.

פלט הרשומות הבעייתיות שנכנסו לטבלת המוצרים עם ערכי NULL ולטבלת המוצרים בהזמנות עם מוצר NULL:

```
SELECT *
FROM Group5_DW.dbo.DW_FACT_PRODUCTS_IN_ORDERS
WHERE DW_Product IS NULL
```

100 %

Results | Messages

	DW_PIO	Orde...	Order_Date	Order_Time	Locati...	DW_Prod...	Quant...	Email	DW_Wareho...	R...	Total_Price_Per_...	Price
1	3500	8	2018-01-09	02:40:55.0000000	145	NULL	9	urna.et@icloud.couk	40	5	NULL	NULL
2	3501	9	2018-01-09	20:19:28.0000000	38	NULL	9	donec.tempor@google.org	2	3	NULL	NULL
3	3502	50	2018-02-07	00:59:03.0000000	14	NULL	9	interdum.enim.non@protonmail.net	50	5	NULL	NULL

```
SELECT *
FROM Group5_DW.dbo.DW_DIM_PRODUCTS
WHERE ProductID IS NULL
```

100 %

Results | Messages

	DW_Prod...	Produc...	ProductTy...	Categ...	Price	Size	Color	Brand	Gen...
1	601	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
2	602	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
3	603	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

ERD

```sql
use Group5


--USERS
--DROP TABLE USERS
CREATE TABLE USERS(
	Email		varchar(50)		NOT NULL PRIMARY KEY,
	[Name-First]	varchar(20)		NOT NULL,
	[Name-Last]	varchar(20)		NOT NULL,
	Gender		varchar(10)		NOT NULL,
	State		varchar(20)		NOT NULL,
	City		varchar(20)		NOT NULL
	)

ALTER TABLE USERS
ADD CONSTRAINT CK1_USERS CHECK (Email LIKE '%@%.%')

--PRODUCTS
--DROP TABLE PRODUCTS
CREATE TABLE PRODUCTS(
	ProductID		Integer		NOT NULL PRIMARY KEY,
	ProductType		varchar(20)		NOT NULL,
	Category		varchar(20)		NOT NULL,
	Price		decimal(10,2)		NOT NULL,
	Size		varchar(4)		NULL,
	Color		varchar(20)		NOT NULL,
	Brand		varchar(20)		NULL,
	Gender		varchar(10)		NULL
	)


ALTER TABLE PRODUCTS
ADD CONSTRAINT CK1_PRODUCTS CHECK (Price > 0)

--WAREHOUSES

--DROP TABLE WAREHOUSES
CREATE TABLE WAREHOUSES(
	WarehouseID		Integer		NOT NULL PRIMARY KEY,
	State		varchar(20)		NOT NULL,
	City		varchar(20)		NOT NULL,
	Surface		Integer		NOT NULL,
	Max_Orders		Integer		NOT NULL
	)


ALTER TABLE WAREHOUSES
ADD CONSTRAINT CK1_WAREHOUSES CHECK (Max_Orders > 0)

ALTER TABLE WAREHOUSES
ADD CONSTRAINT CK2_WAREHOUSES CHECK (Surface > 0)

--ORDERS
--DROP TABLE ORDERS
```

```sql
CREATE TABLE ORDERS(
	OrderID		Integer		NOT NULL PRIMARY KEY,
	Order_Date	Date		NOT NULL,
	Order_Time	Time		NOT NULL,
	[Phone-Number]	Varchar(20)	NOT NULL,
	CardNumber	Varchar(20)	NOT NULL,
	Email		Varchar(50)	NULL,
	State		Varchar(20)	NOT NULL,
	City		Varchar(20)	NOT NULL,
	Street		Varchar(20)	NOT NULL,
	Apartment	Integer		NULL,
	Warehouse	Integer		NOT NULL,
	Rate		Integer		NULL
	)

ALTER TABLE ORDERS
ADD CONSTRAINT CK1_ORDERS CHECK (Email LIKE '%@%.%')


ALTER TABLE ORDERS
ADD CONSTRAINT FK_UserEmail_ORDERS
	FOREIGN KEY (Email)
	REFERENCES USERS (Email)


ALTER TABLE ORDERS
ADD CONSTRAINT FK_Warehouse_ORDERS
	FOREIGN KEY (Warehouse)
	REFERENCES WAREHOUSES (WarehouseID)


--REVIEWS
--DROP TABLE REVIEWS
CREATE TABLE REVIEWS(
	IncrementalNum	Integer		NOT NULL PRIMARY KEY,
	User_Email	varchar(50)	NOT NULL,
	Review_Date	Date		NOT NULL,
	Review_Time	Time		NOT NULL,
	Rating		Integer		NOT NULL,
	ProductID	Integer		NOT NULL
	)

ALTER TABLE REVIEWS
ADD CONSTRAINT CK1_REVIEWS CHECK (User_Email LIKE '%@%.%')

ALTER TABLE REVIEWS
ADD CONSTRAINT CK2_REVIEWS CHECK (Rating BETWEEN 1 AND 5)

ALTER TABLE REVIEWS
ADD CONSTRAINT FK_UserEmail_REVIEWS
	FOREIGN KEY (User_Email)
	REFERENCES USERS (Email)

ALTER TABLE REVIEWS
ADD CONSTRAINT FK_ProductID_REVIEWS
	FOREIGN KEY (ProductID)
	REFERENCES PRODUCTS (ProductID)


--QUANTITIES
```

```sql
--DROP TABLE QUANTITIES
CREATE TABLE QUANTITIES(
	IncrementalNum		Integer		NOT NULL PRIMARY KEY,
	OrderID		Integer		NOT NULL,
	ProductID		Integer		NULL,
	Quantity		Integer		NOT NULL
	)

ALTER TABLE QUANTITIES
ADD CONSTRAINT FK_OrderID_QUANTITIES
	FOREIGN KEY (OrderID)
	REFERENCES ORDERS (OrderID)

ALTER TABLE QUANTITIES
ADD CONSTRAINT FK_ProductID_QUANTITIES
	FOREIGN KEY (ProductID)
	REFERENCES PRODUCTS (ProductID)

--COVID19
--DROP TABLE COVID19
CREATE TABLE COVID19(
	IncrementalNum		Integer		NOT NULL PRIMARY KEY,
	Date		Date		NOT NULL,
	State		varchar(20)		NOT NULL,
	Death		Integer		NULL,
	Negative		Integer		NULL,
	Positive		Integer		NULL,
	Recovered		Integer		NULL
	)

ALTER TABLE COVID19
ADD CONSTRAINT CK1_COVID19 CHECK (Death >= 0)

ALTER TABLE COVID19
ADD CONSTRAINT CK2_COVID19 CHECK (Negative >= 0)

ALTER TABLE COVID19
ADD CONSTRAINT CK3_COVID19 CHECK (Positive >= 0)

ALTER TABLE COVID19
ADD CONSTRAINT CK4_COVID19 CHECK (Recovered > 0)

--DROP TABLES
DROP TABLE QUANTITIES
DROP TABLE REVIEWS
DROP TABLE ORDERS
DROP TABLE PRODUCTS
DROP TABLE USERS
DROP TABLE WAREHOUSES
DROP TABLE COVID19
```

MRR

```
use Group5_MRR
--USERS
--DROP TABLE MRR_USERS
--TRUNCATE TABLE MRR_USERS
CREATE TABLE MRR_USERS(
       Email                varchar(50)          NOT NULL PRIMARY KEY,
       [Name-First]         varchar(20)          NOT NULL,
       [Name-Last]          varchar(20)          NOT NULL,
       Gender               varchar(10)          NOT NULL,
       State                varchar(20)          NOT NULL,
       City                 varchar(20)          NOT NULL
       )

--PRODUCTS
--DROP TABLE MRR_PRODUCTS
--TRUNCATE TABLE MRR_PRODUCTS
CREATE TABLE MRR_PRODUCTS(
       ProductID            Integer              NOT NULL PRIMARY KEY,
       ProductType          varchar(20)          NOT NULL,
       Category             varchar(20)          NOT NULL,
       Price                decimal(10,2)        NOT NULL,
       Size                 varchar(4)           NULL,
       Color                varchar(20)          NOT NULL,
       Brand                varchar(20)          NULL,
       Gender               varchar(10)          NULL
       )

--WAREHOUSES
--DROP TABLE MRR_WAREHOUSES
--TRUNCATE TABLE MRR_WAREHOUSES
CREATE TABLE MRR_WAREHOUSES(
       WarehouseID          Integer              NOT NULL PRIMARY KEY,
       State                varchar(20)          NOT NULL,
       City                 varchar(20)          NOT NULL,
       Surface              Integer              NOT NULL,
       Max_Orders           Integer              NOT NULL
       )

--ORDERS
--DROP TABLE MRR_ORDERS
--TRUNCATE TABLE MRR_ORDERS
CREATE TABLE MRR_ORDERS(
       OrderID              Integer              NOT NULL PRIMARY KEY,
       Order_Date           Date                 NOT NULL,
       Order_Time           Time                 NOT NULL,
       [Phone-Number]       Varchar(20)          NOT NULL,
       CardNumber           Varchar(20)          NOT NULL,
       Email                Varchar(50)          NULL,
       State                Varchar(20)          NOT NULL,
       City                 Varchar(20)          NOT NULL,
       Street               Varchar(20)          NOT NULL,
       Apartment            Integer              NULL,
       Warehouse            Integer              NOT NULL,
       Rate                 Integer              NULL
       )
```

```sql
--REVIEWS

--DROP TABLE MRR_REVIEWS
--TRUNCATE TABLE MRR_REVIEWS
CREATE TABLE MRR_REVIEWS(
	IncrementalNum		Integer		NOT NULL PRIMARY KEY,
	User_Email		varchar(50)		NOT NULL,
	Review_Date		Date		NOT NULL,
	Review_Time		Time		NOT NULL,
	Rating		Integer		NOT NULL,
	ProductID		Integer		NOT NULL
	)

--QUANTITIES
--DROP TABLE MRR_QUANTITIES
--TRUNCATE TABLE MRR_QUANTITIES
CREATE TABLE MRR_QUANTITIES(
	IncrementalNum		Integer		NOT NULL PRIMARY KEY,
	OrderID		Integer		NOT NULL,
	ProductID		Integer		NULL,
	Quantity		Integer		NOT NULL
	)

--COVID19
--DROP TABLE MRR_COVID19
--TRUNCATE TABLE MRR_COVID19
CREATE TABLE MRR_COVID19(
	IncrementalNum		Integer		NOT NULL PRIMARY KEY,
	Date		Date		NOT NULL,
	State		varchar(20)		NOT NULL,
	Death		Integer		NULL,
	Negative		Integer		NULL,
	Positive		Integer		NULL,
	Recovered		Integer		NULL
	)

--DROP TABLES
DROP TABLE MRR_QUANTITIES
DROP TABLE MRR_REVIEWS
DROP TABLE MRR_ORDERS
DROP TABLE MRR_PRODUCTS
DROP TABLE MRR_USERS
DROP TABLE MRR_WAREHOUSES
DROP TABLE MRR_COVID19
```

STG

```sql
use Group5_STG

--USERS
--DROP TABLE DIM_USERS
CREATE TABLE STG_DIM_USERS(
       Email                varchar(50)          NOT NULL PRIMARY KEY,
       [Name-First]         varchar(20)          NOT NULL,
       [Name-Last]          varchar(20)          NOT NULL,
       Gender               varchar(10)          NOT NULL,
       State                varchar(20)          NOT NULL,
       City                 varchar(20)          NOT NULL
       )

--PRODUCTS

--DROP TABLE DIM_PRODUCTS
CREATE TABLE STG_DIM_PRODUCTS(
       ProductID            Integer              NOT NULL PRIMARY KEY,
       ProductType          varchar(20)          NOT NULL,
       Category             varchar(20)          NOT NULL,
       Price                decimal(10,2)        NOT NULL,
       Size                 varchar(4)           NULL,
       Color                varchar(20)          NOT NULL,
       Brand                varchar(20)          NULL,
       Gender               varchar(10)          NULL
       )

--WAREHOUSES
--DROP TABLE DIM_WAREHOUSES
CREATE TABLE STG_DIM_WAREHOUSES(
       WarehouseID          Integer              NOT NULL PRIMARY KEY,
       Location             Integer              NOT NULL,
       Surface              Integer              NOT NULL,
       Max_Orders           Integer              NOT NULL
       )


--LOCATIONS
--DROP TABLE STG_DIM_LOCATIONS
CREATE TABLE STG_DIM_LOCATIONS(
       State                Varchar(20)          NOT NULL,
       City                 Varchar(20)          NULL
       )

  --PRODUCTS_IN_ORDERS
--DROP TABLE STG_FACT_PRODUCTS_IN_ORDERS
CREATE TABLE STG_FACT_PRODUCTS_IN_ORDERS(
IncremenatalNum             Integer              NOT NULL,
       OrderID              Integer              NOT NULL,
       Order_Date           Date                 NOT NULL,
       Order_Time           Time                 NOT NULL,
       Location             Integer              NOT NULL,
       DW_Product           Integer              NULL,
       Quantity             Integer              NOT NULL,
       Email                Varchar(50)          NULL,
       DW_Warehouse         Integer              NOT NULL,
       Rate                 Integer              NULL,
       Total_Price_Per_PIO  Decimal(10,2)        NULL,
       Price                Decimal(10,2)        NULL
```

```sql
        )

--REVIEWS
--DROP TABLE STG_FACT_REVIEWS
CREATE TABLE STG_FACT_REVIEWS(
        IncrementalNum        Integer              NOT NULL PRIMARY KEY,
        User_Email            varchar(50)          NOT NULL,
        Review_Date           Date                 NOT NULL,
        Review_Time           Time                 NOT NULL,
        Rating                Integer              NOT NULL,
        DW_Product            Integer              NOT NULL,
        Loaded                varchar(5)           NULL

        )
  use Group5_STG

--COVID19
--DROP TABLE STG_FACT_COVID19
CREATE TABLE STG_FACT_COVID19(
        IncrementalNum        int            NOT NULL,
        Date                  Date                 NOT NULL,
        State                 varchar(20)          NOT NULL,
        Death                 Integer              NULL,
        Negative              Integer              NULL,
        Positive              Integer              NULL,
        Recovered             Integer              NULL
        )

ALTER TABLE STG_FACT_COVID19
ADD CONSTRAINT PK_STG_COVID19
        PRIMARY KEY (Date, State)

--DROP TABLES
DROP TABLE STG_FACT_REVIEWS
DROP TABLE STG_FACT_PRODUCTS_IN_ORDERS
DROP TABLE STG_FACT_COVID19
DROP TABLE STG_DIM_PRODUCTS
DROP TABLE STG_DIM_USERS
DROP TABLE STG_DIM_WAREHOUSES
DROP TABLE STG_DIM_LOCATIONS
```

DW

```
use Group5_DW

--USERS
--DROP TABLE DW_DIM_USERS
CREATE TABLE DW_DIM_USERS(
	DW_User		Int IDENTITY(1,1)	NOT NULL PRIMARY KEY,
	Email		varchar(50)		NOT NULL,
	[Name-First]	varchar(20)		NOT NULL,
	[Name-Last]	varchar(20)		NOT NULL,
	Gender		varchar(10)		NOT NULL,
	State		varchar(20)		NOT NULL,
	City		varchar(20)		NOT NULL,
	ValidFrom	Date			NULL,
	ValidUntil	Date			NULL
	)

--PRODUCTS

--DROP TABLE DW_DIM_PRODUCTS
CREATE TABLE DW_DIM_PRODUCTS(
	DW_Product		Int IDENTITY(1,1)	NOT NULL PRIMARY KEY,
	ProductID		Integer		NULL,
	ProductType		varchar(20)		NULL,
	Category		varchar(20)		NULL,
	Price		decimal(10,2)		NULL,
	Size		varchar(4)		NULL,
	Color		varchar(20)		NULL,
	Brand		varchar(20)		NULL,
	Gender		varchar(10)		NULL
	)

--WAREHOUSES
--DROP TABLE DIM_WAREHOUSES
CREATE TABLE DW_DIM_WAREHOUSES(
	DW_Warehouse		Int IDENTITY(1,1)	NOT NULL PRIMARY KEY,	WarehouseID
	Integer		NOT NULL,
	Location		Integer		NOT NULL,
	Surface		Integer		NOT NULL,
	Max_Orders		Integer		NOT NULL
	)


--LOCATIONS
--DROP TABLE DW_DIM_LOCATIONS
CREATE TABLE DW_DIM_LOCATIONS(
	Location		Int	NOT NULL PRIMARY KEY,
	State		Varchar(20)		NOT NULL,
	City		Varchar(20)		NULL
	)

--DROP TABLE dbo.DIM_DATE
create table DIM_DATE
(
	TheDate date,
	TheDay int,
	TheDayName varchar(20),
	TheDayOfWeek int,
	IsWeekend int,
	TheWeek int,
```

```
        TheWeekOfMonth int,
        TheMonth int,
        TheMonthName varchar(20),
        TheQuarter int,
        TheFirstOfQuarter date,
        TheLastOfQuarter date,
        TheYear int,
        IsLeapYear int
        )


DECLARE @StartDate  date = '20100101';
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),
    TheDayName      = DATENAME(WEEKDAY,   d),
    TheWeek         = DATEPART(WEEK,      d),
    TheISOWeek      = DATEPART(ISO_WEEK,  d),
    TheDayOfWeek    = DATEPART(WEEKDAY,   d),
    TheMonth        = DATEPART(MONTH,     d),
    TheMonthName    = DATENAME(MONTH,     d),
    TheQuarter      = DATEPART(Quarter,   d),
    TheYear         = DATEPART(YEAR,      d),
    TheFirstOfMonth = DATEFROMPARTS(YEAR(d), MONTH(d), 1),
    TheLastOfYear   = DATEFROMPARTS(YEAR(d), 12, 31),
    TheDayOfYear    = DATEPART(DAYOFYEAR, d)
  FROM d
),

dim AS
(
  SELECT
    TheDate,
    TheDay,
    TheDayName,
    TheDayOfWeek,
    IsWeekend           = CASE WHEN TheDayOfWeek IN (CASE @@DATEFIRST WHEN 1 THEN 6 WHEN 7
THEN 1 END,7)
                            THEN 1 ELSE 0 END,
    TheWeek,
    TheWeekOfMonth      = CONVERT(tinyint, DENSE_RANK() OVER
                            (PARTITION BY TheYear, TheMonth ORDER BY TheWeek)),
    TheMonth,
    TheMonthName,
    TheQuarter,
    TheFirstOfQuarter   = MIN(TheDate) OVER (PARTITION BY TheYear, TheQuarter),
    TheLastOfQuarter    = MAX(TheDate) OVER (PARTITION BY TheYear, TheQuarter),
    TheYear,
    IsLeapYear          = CONVERT(bit, CASE WHEN (TheYear % 400 = 0)
                            OR (TheYear % 4 = 0 AND TheYear % 100 <> 0)
```

```sql
                            THEN 1 ELSE 0 END)
  FROM src
)
insert into DIM_DATE
SELECT * FROM dim
  ORDER BY TheDate

OPTION (MAXRECURSION 0);


--DROP TABLE dbo.DIM_TIME
create table DIM_TIME
(
       TheTime time,
       TheHour int,
       TheMinute int
       )


DECLARE @StartTime  time = '00:00';
DECLARE @CutoffTime time = '23:59';WITH seq(n) AS
(
  SELECT 0 UNION ALL SELECT n + 1 FROM seq
  WHERE n < DATEDIFF(MINUTE, @StartTime, @CutoffTime)
),
t(t) AS
(
  SELECT DATEADD(MINUTE, n, @StartTime) FROM seq
),
src AS
(
  SELECT
       TheTime         = CONVERT(TIME, t),
       TheHour          = DATEPART(HOUR, t),
       TheMinute      = DATENAME(MINUTE, t)
  FROM T
),

dim AS
(
  SELECT
       TheTime,
       TheHour,
       TheMinute

  FROM src
)
insert into DIM_TIME
SELECT * FROM dim
  ORDER BY TheTime

OPTION (MAXRECURSION 0);



--PRODUCTS_IN_ORDERS
--DROP TABLE DW_FACT_PRODUCTS_IN_ORDERS
CREATE TABLE DW_FACT_PRODUCTS_IN_ORDERS(
       DW_PIO              Int IDENTITY(1,1)    NOT NULL PRIMARY KEY,
       OrderID             Integer              NOT NULL,
```

```
	Order_Date		Date		NOT NULL,
	Order_Time		Time		NOT NULL,
	Location		Integer		NOT NULL,
	DW_Product		Integer		NULL,
	Quantity		Integer		NOT NULL,
	Email			Varchar(50)	NULL,
	DW_Warehouse		Integer		NOT NULL,
	Rate			Integer		NULL,
	Total_Price_Per_PIO	Decimal(10,2)	NULL,
	Price			Decimal(10,2)	NULL
	)


--PRODUCTS_IN_ORDERS
--DROP TABLE REJECTED_PIO
CREATE TABLE REJECTED_PIO(
	DW_PIO		Int IDENTITY(1,1)	NOT NULL PRIMARY KEY,
	OrderID		Integer		NOT NULL,
	Order_Date		Date		NOT NULL,
	Order_Time		Time		NOT NULL,
	Location		Integer		NOT NULL,
	DW_Product		Integer		NULL,
	Quantity		Integer		NOT NULL,
	Email			Varchar(50)	NULL,
	DW_Warehouse		Integer		NOT NULL,
	Rate			Integer		NULL,
	Total_Price_Per_PIO	Decimal(10,2)	NULL,
	Price			Decimal(10,2)	NULL
	)









--REVIEWS
--DROP TABLE DW_FACT_REVIEWS
CREATE TABLE DW_FACT_REVIEWS(
	DW_Review		Int IDENTITY(1,1)	NOT NULL PRIMARY KEY,
	User_Email		varchar(50)	NOT NULL,
	Review_Date		Date		NOT NULL,
	Review_Time		Time		NOT NULL,
	Rating		Integer		NOT NULL,
	DW_User		Integer		NOT NULL,
	DW_Product		Integer		NOT NULL
	)



CREATE TABLE NEW_REVIEWS(
	User_Email		varchar(50)	NOT NULL,
	Review_Date		Date		NOT NULL,
	Review_Time		Time		NOT NULL,
	Rating		Integer		NOT NULL,
	DW_User		Integer		NOT NULL,
	DW_Product		Integer		NOT NULL
	)
```

```sql
CREATE TABLE OLD_REVIEWS(
	User_Email		varchar(50)		NOT NULL,
	Review_Date		Date			NOT NULL,
	Review_Time		Time			NOT NULL,
	Rating			Integer			NOT NULL,
	DW_User			Integer			NOT NULL,
	DW_Product		Integer			NOT NULL
	)




--SUMMARY_COVID
--DROP TABLE SUMMARY_COVID
CREATE TABLE DW_SUMMARY_COVID19(
	Date			Date			NOT NULL,
	State			varchar(20)		NOT NULL,
	Death			Integer			NULL,
	Negative		Integer			NULL,
	Positive		Integer			NULL,
	Recovered		Integer			NULL,
	NumOfOrders		Integer			NULL,
	TotalIncome		Decimal(10,2)		NULL,
	AvgPIO			Decimal(10,2)		NULL
	)

ALTER TABLE DW_SUMMARY_COVID19
ADD CONSTRAINT PK_SUMMARY_COVID19
	PRIMARY KEY (Date, State)




-- SUMMARY_WAREHOUSES

--DROP TABLE DW_SUMMARY_WAREHOUSES
CREATE TABLE DW_SUMMARY_WAREHOUSES(
	Date			Date			NOT NULL,
	DW_Warehouse		Int			NOT NULL,
	AvgPIO			Decimal(10,2)		NULL,
	NumOfOrders		Integer			NULL,
	TotalIncome		Decimal(10,2)		NULL,
	AvgRate			Decimal(10,2)		NULL
)
ALTER TABLE DW_SUMMARY_WAREHOUSES
ADD CONSTRAINT PK_DW_SUMMARY_WAREHOUSES
	PRIMARY KEY (Date, DW_Warehouse)


CREATE TABLE OLD_VERSIONS_USERS(
	DW_User			Int	NOT NULL PRIMARY KEY,
	Email			varchar(50)		NOT NULL,
	[Name-First]		varchar(20)		NOT NULL,
	[Name-Last]		varchar(20)		NOT NULL,
	Gender			varchar(10)		NOT NULL,
	State			varchar(20)		NOT NULL,
	City			varchar(20)		NOT NULL,
	ValidFrom		Date			NULL,
```

```
        ValidUntil          Date                NULL
        )


CREATE PROCEDURE [dbo].[updateOldUserVersions]

AS BEGIN
UPDATE DW_DIM_USERS SET ValidUntil = GETDATE() WHERE DW_User IN (SELECT DW_User FROM
OLD_VERSIONS_USERS)
TRUNCATE TABLE OLD_VERSION_JEWELS
END



--DROP TABLES
DROP TABLE DW_SUMMARY_COVID
DROP TABLE DW_SUMMARY_WAREHOUSES
DROP TABLE DW_FACT_REVIEWS
DROP TABLE DW_FACT_PRODUCTS_IN_ORDERS
DROP TABLE DW_FACT_COVID19
DROP TABLE DW_DIM_PRODUCTS
DROP TABLE DW_DIM_USERS
DROP TABLE DW_DIM_WAREHOUSES
DROP TABLE DW_DIM_LOCATIONS
```