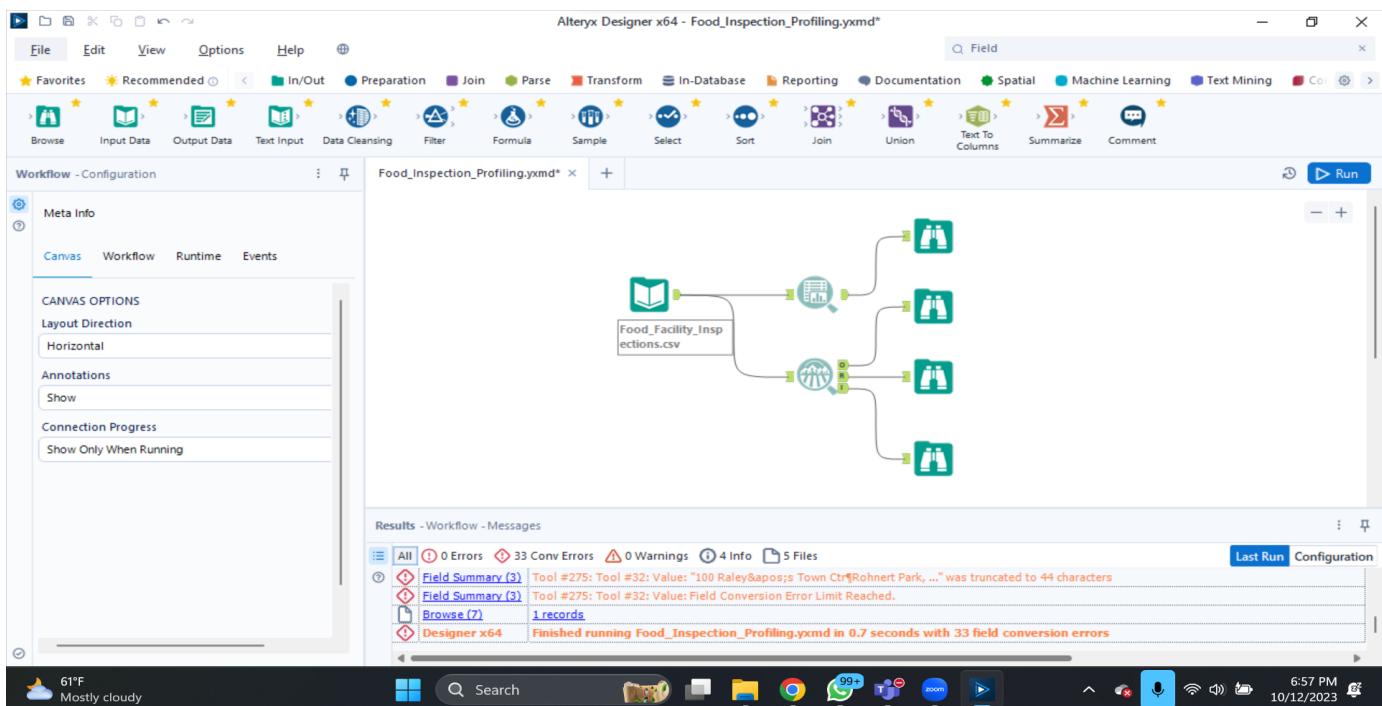


Food Inspection Documentation

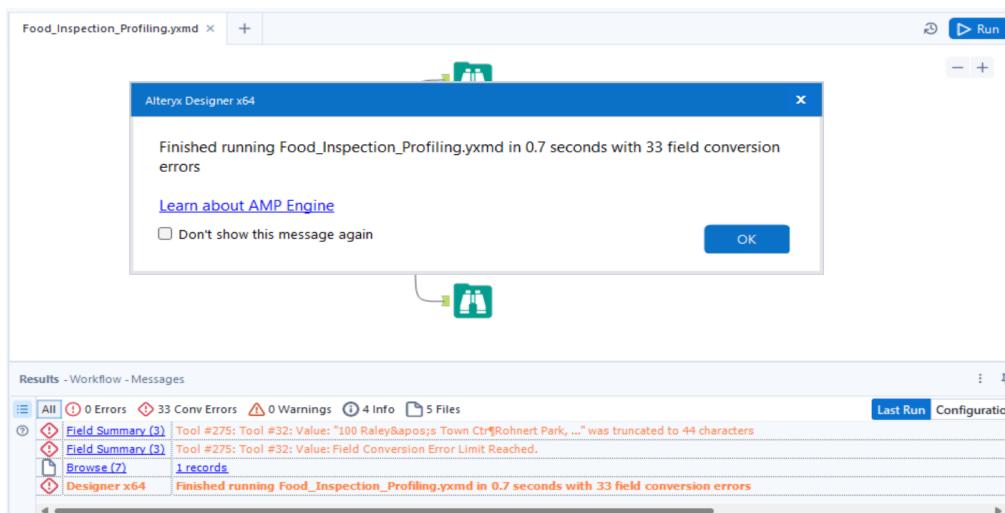
Team Member Group 13:

1. Shreyasi Wakankar : 002771284
2. Muskan Srivastava: 002794929
3. Pramita Dileep Sandhyan: 002766881
4. Rhea Bajpai: 002702927

Alteryx Profiling:



Time taken for execution:



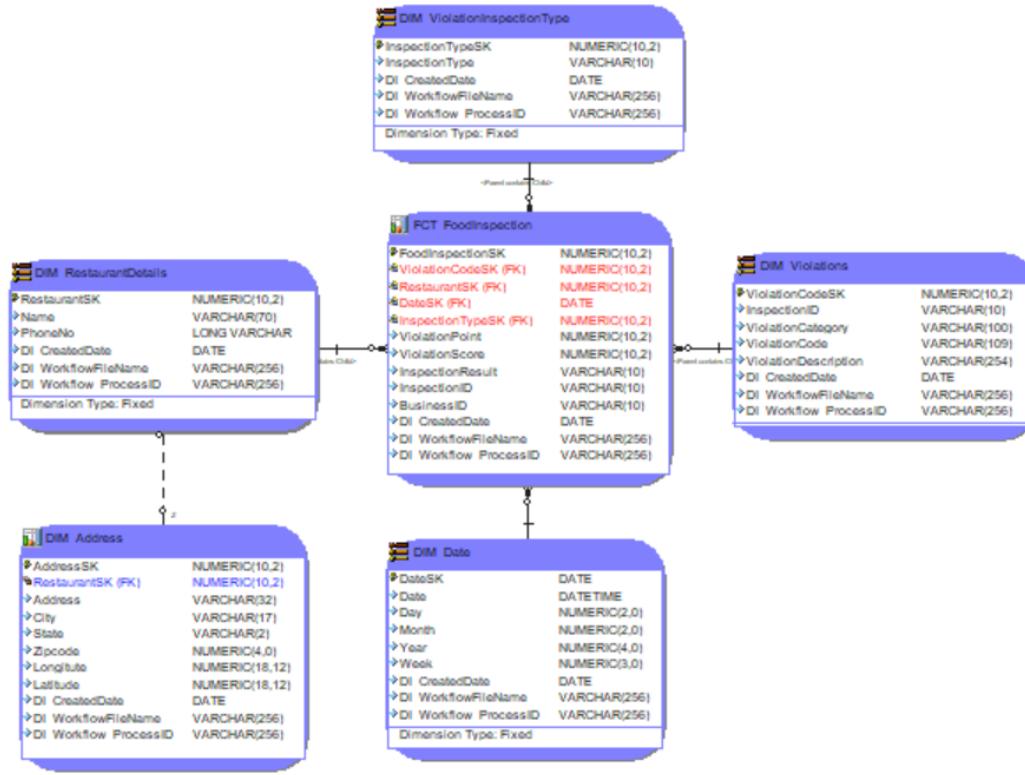
Observations and Changes Required:

Field Name	Data Type from website	Data Type after importing csv	Data Type Changes Required after profiling	Longest Length	Null Values
Business Id	String	String	String	9	0
Name	String	String	String	70	0
Address	String	String	String	32	0
City	String	String	String	17	0
State	String	String	String	2	0
ZipCode	String	String	Int	10	0
Phone No	String	String	Long	12	4307
InspectionID	String	String	String	9	0
Date	Date & Time	String	Date & Time	10	3
Inspection Type	String	String	String	9	3
Violation Code	String	String	String	109	10082
Violation Description	String	String	String	254	10132
Location	Location	String	String	86	3

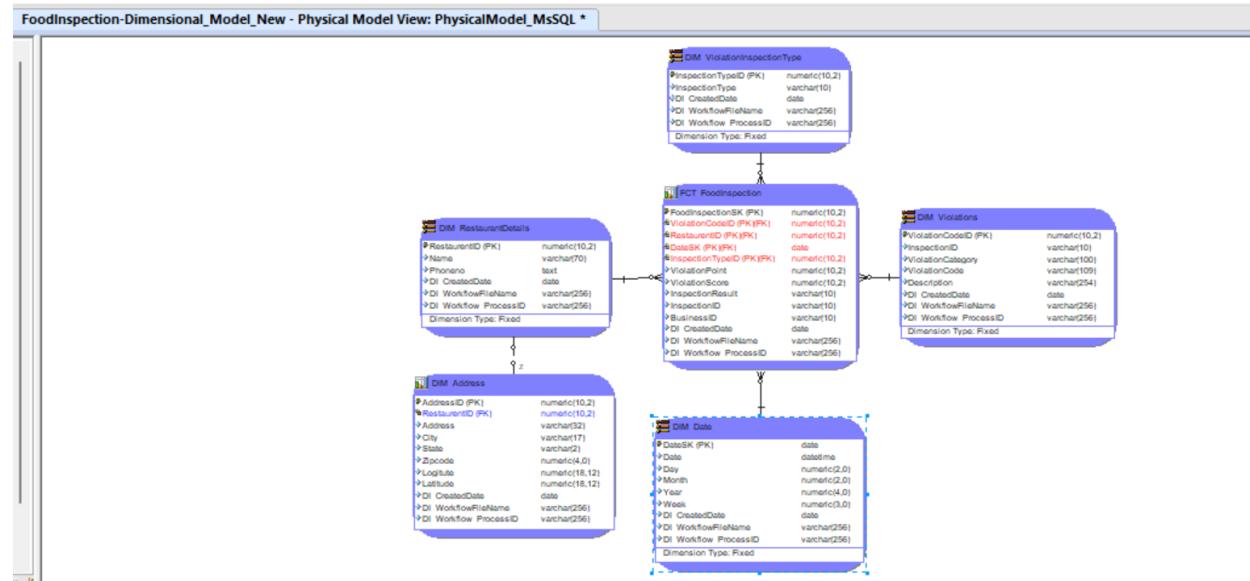
We observe that there should be three changes in data type for three columns: ZipCode, PhoneNumber and Date. There are three columns (PhoneNumber, ViolationCode and Violation Description) which have null values.

Part 2- ER Studio

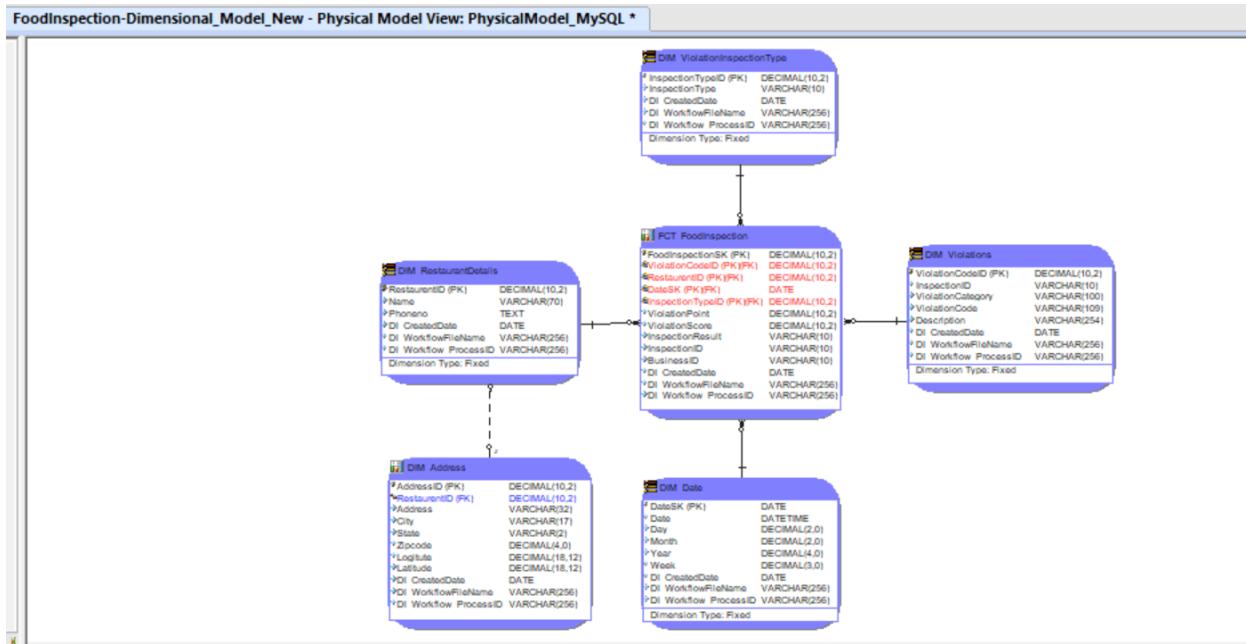
Logical Model:



MSSQL Physical Model



MYSQL Physical Model



TALEND:

Creation of Staging table:

The Talend job interface shows the following components and flow:

- SourceData**: The input component, connected to **tMap_1**.
- tMap_1**: A mapping component that processes 25312 rows in 3.13s.
- MySQL_p**: The target component, connected to **tMap_1**.
- MSSQL_New**: The target component, connected to the output of **tMap_1**.

Job Staging_Food_Inspection details:

- Basic Run**:
 - Execution: Run, Kill, Clear
 - Statistics: connecting to socket on port 3972, [statistics] connected, [statistics] disconnected
 - Log: Job Staging_Food_Inspection ended at 17:20 22/10/2023. [Exit code = 0]
- Component** palette on the right lists various Talend components like tLogRow, tMap, tAggregateRow, etc.
- Cloud Artifact** panel on the right shows connection settings for MySQL and MSSQL.

Purpose and Business Logic: The source data is cleaned and fed into MySQL and MSSQL through a Tmap so that we can join rows, alter the rows and perform transformations. The data has been taken from the source provided in the class and it has been profiled in Alterx as shown above.

Table in MsSQL:

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database structure, including the 'food_inspection' database. The central pane displays a query results grid for the 'stg_food_inspection' table. The grid contains 25,312 rows of data with columns: BusinessId, Name, Address, City, State, ZipCode, PhoneNumber, InspectionId, Date, and InspectionType. The 'Date' column is null for most entries. The 'InspectionType' column shows values like 'ROUTINE', 'DA0004NLT', 'DA000KOIO', etc. The bottom status bar indicates the query was executed successfully at 7:29 PM on 10/13/2023.

BusinessId	Name	Address	City	State	ZipCode	PhoneNumber	InspectionId	Date	InspectionType
PR0017321	Forget Me Not Cakes	1435 N McDowell Blvd Ste 100	Petaluma	CA	94954	17073643497	DA	NULL	
PR0000722	Rwera Restorante Inc	75 Montgomery Dr	Santa Rosa	CA	95404	17075792682	DA0004NLT	2016-09-08 00:00:00.000	ROUTINE
PR0020278	Golden Gate Bell, LLC Taco Bell #30934	1835 Mendocino Ave	Santa Rosa	CA	95401	17075718241	DA000KOIO	2017-08-09 00:00:00.000	ROUTINE
PR0014451	Marys Churros	2995 Wijan Ct	Santa Rosa	CA	95407	17079744801	DA0036FLR	2018-08-04 00:00:00.000	ROUTINE
PR0017756	Hooters	6099 Redwood Dr	Rohnert Park	CA	94928	17075859464	DA0058SQQ	2017-09-04 00:00:00.000	ROUTINE
PR0017167	Baskin Robbins Petaluma	60 E Washington St	Petaluma	CA	94952	17077633131	DA00BEEGO	2016-09-04 00:00:00.000	ROUTINE
PR0017795	Asian Market	1110 Petaluma Hill Rd #1	Santa Rosa	CA	95404	17075459618	DA00CQIPO	2018-08-03 00:00:00.000	ROUTINE
PR0001591	Burger King	5304 N Old Redwood Hwy	Petaluma	CA	94952	NULL	DA00EQQDJ	2017-04-07 00:00:00.000	ROUTINE
PR0020537	Mi Casita Mexican Restaurant	16380 Mill St #B	Guerneville	CA	95446	17078698016	DA00FAFTI	2016-01-08 00:00:00.000	ROUTINE
PR0000573	Hidden Valley School	3435 Bonita Vista Ln	Santa Rosa	CA	95404	NULL	DA00IBZEV	2018-12-04 00:00:00.000	ROUTINE

Row count:

The screenshot shows the Microsoft SQL Server Management Studio interface. On the left, the Object Explorer pane displays the database structure for 'MS\SQLEXPRESS'. In the center, the 'SQLQuery1.sql' window contains the following T-SQL code:

```
select * from stg_food_inspection

select COUNT(*) as row_count from stg_food_inspection
```

The results pane shows a single row with the value '25312' under the 'row_count' column. A status bar at the bottom indicates the query was executed successfully.

Table in MySQL:

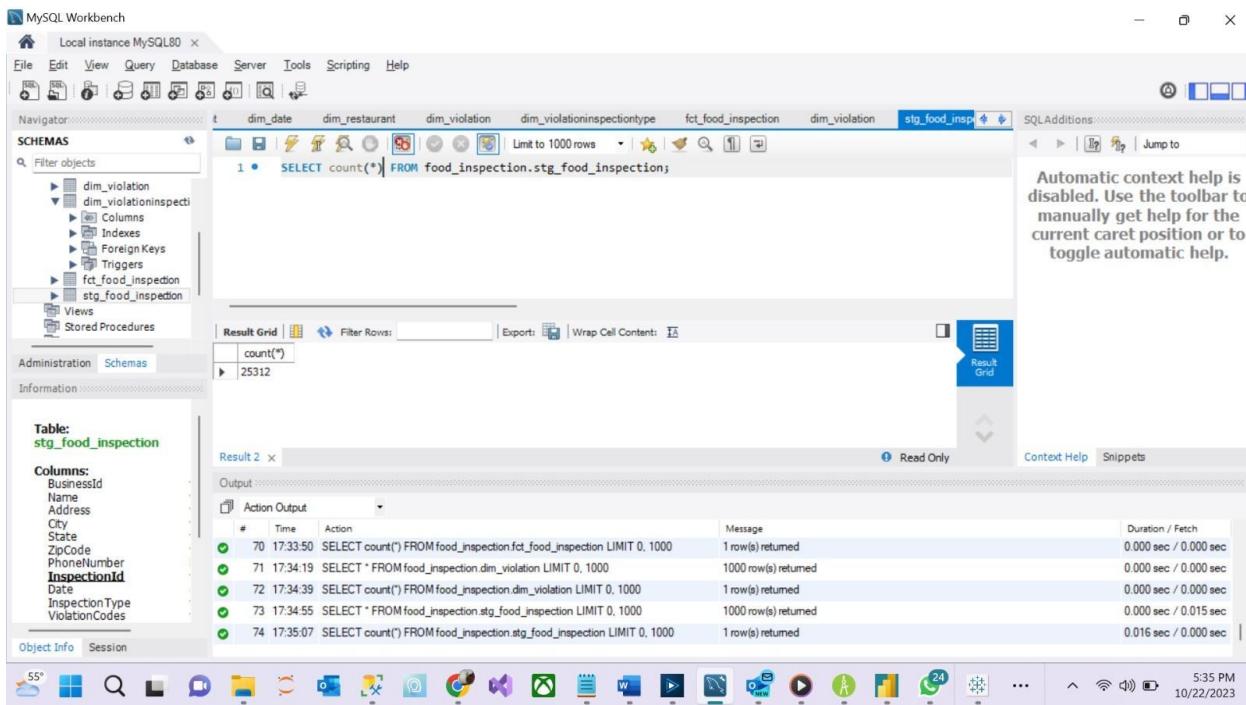
The screenshot shows the MySQL Workbench interface. The left sidebar shows the database schema with the 'stg_food_inspection' table selected. The main area displays the table data in a grid format:

BusinessId	Name	Address	City	State	ZipCode	PhoneNumber	Inspected
PR0017321	Forget Me Not Cakes	1435 N McDowell Blvd Ste 100	Petaluma	CA	94954	17073643497	DA
PR0000722	Riviera Restaurante Inc	75 Montgomery Dr	Santa Rosa	CA	95404	17075792682	DA0001
PR0020272	Golden Gate Bell, LLC Taco Bell #30934	1835 Mendocino Ave	Santa Rosa	CA	95401	170757928241	DA0001
PR0014451	Marys Churros	2995 Wiljan Ct	Santa Rosa	CA	95407	17079744801	DA003

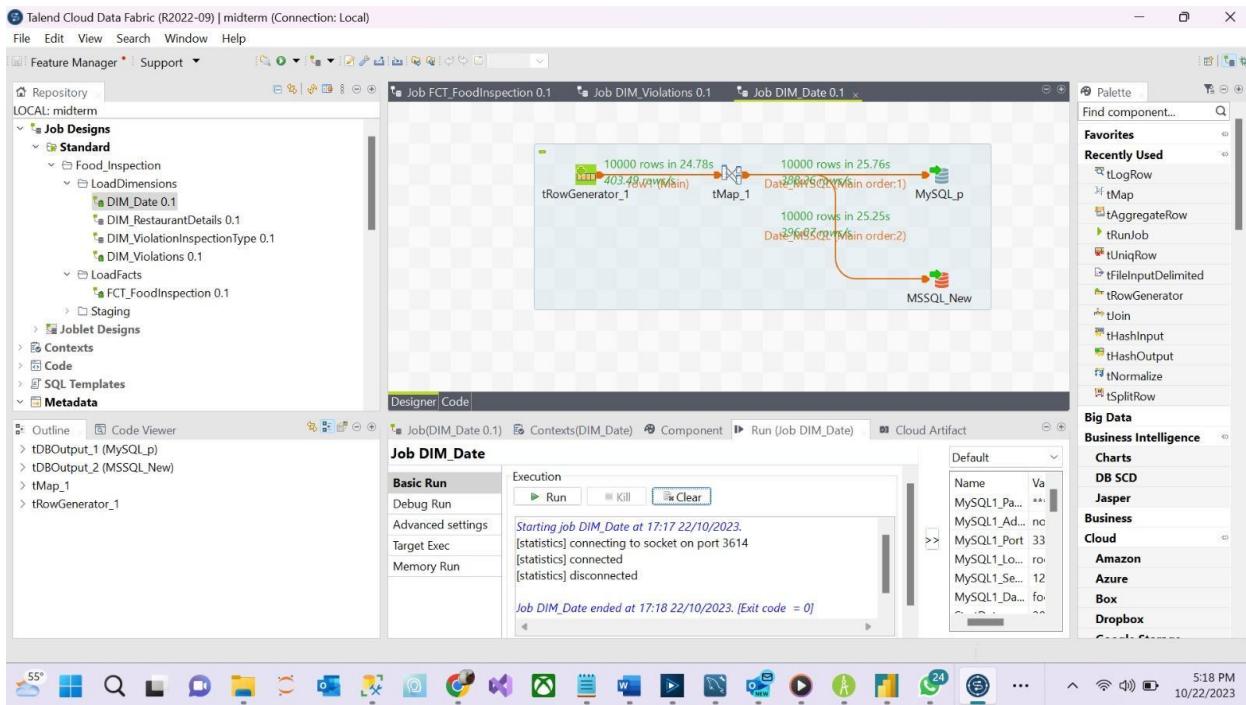
Below the table, the 'Output' pane shows the execution history of the previous query:

#	Time	Action	Message	Duration / Fetch
69	17:33:28	SELECT * FROM food_inspection.fct_food_inspection LIMIT 0, 1000	1000 row(s) returned	0.000 sec / 0.000 sec
70	17:33:50	SELECT count(*) FROM food_inspection.fct_food_inspection LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
71	17:34:19	SELECT * FROM food_inspection.dim_violation LIMIT 0, 1000	1000 row(s) returned	0.000 sec / 0.000 sec
72	17:34:39	SELECT count(*) FROM food_inspection.dim_violation LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
73	17:34:55	SELECT * FROM food_inspection.stg_food_inspection LIMIT 0, 1000	1000 row(s) returned	0.000 sec / 0.015 sec

Row Count:



Loading Date Dimension:



Purpose and Business Logic: We have used a tRowGenerator to load the data to MYSQL and MSSQL through a tmap. We have used a tRowGenerator so that we can generate rows and field as needed and feed each filed with a value taken randomly from the provided list. Tmap is used to put the fields and rows

generated from tRowGenerator to both the databases by manipulating/altering the input. We have given the grain of the date_dimension as Day, Month, Year, week, Quarter.

Table in MySQL:

The screenshot shows the MySQL Workbench interface with the 'dim_date' table selected in the 'Tables' section of the Navigator. The table has 20 columns: DateSK, Day_Num, Day_Str, Month_Num, Month_Str, Year_Num, Week, Quarter, dt, DI_CreatedDate, DayName, MonthName, YearName, WeekName, QuarterName, DayNum, MonthNum, YearNum, WeekNum, and QuarterNum. The Result Grid displays 20 rows of data from January 2009 to January 2020. The SQL pane contains the query: 'SELECT * FROM food_inspection.dim_date;'. The Output pane shows the execution log with the message '55 15:58:37 SELECT * FROM food_inspection.dim_date; 1000 row(s) returned'.

Row Count:

The screenshot shows the MySQL Workbench interface with the 'dim_date' table selected in the 'Tables' section of the Navigator. The SQL pane contains the query: 'SELECT count(*) FROM food_inspection.dim_date;'. The Result Grid displays a single row with the value '1000'. The Output pane shows the execution log with the message '55 15:58:37 SELECT count(*) FROM food_inspection.dim_date; 1000 row(s) returned'.

Table in MsSQL:

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the 'food_inspection' database is selected. In the center pane, a query window displays a script for generating a date dimension table. Below the script, the results grid shows 10 rows of data from 2009 to 2018, including columns like DateSK, Day_Num, Month_Num, Year_Num, and DI_CreatedDate. The status bar at the bottom indicates the query was executed successfully.

```

SELECT TOP (1000) [DateSK]
      ,[Day_Num]
      ,[Day_Str]
      ,[Month_Num]
      ,[Month_Str]
      ,[Year_Num]
      ,[Week]
      ,[Quarter]
      ,[dt]
      ,[DI_CreatedDate]
      ,[DI_WorkflowFileName]
      ,[DI_WorkflowProcessID]
  FROM [food_inspection].[dbo].[DIM_Date]
  
```

DateSK	Day_Num	Day_Str	Month_Num	Month_Str	Year_Num	Week	Quarter	dt	DI_CreatedDate	DI_WorkflowFileName
1012009	1	Thursday	0	January	2009	1	1	2009-01-01 00:00:00.000	2023-10-22 17:17:51.980	C:/Users/pramit/Downloads/Food_Facility_Inspe
1012010	1	Friday	0	January	2010	1	1	2010-01-01 00:00:00.000	2023-10-22 17:17:52.770	C:/Users/pramit/Downloads/Food_Facility_Inspe
1012011	1	Saturday	0	January	2011	1	1	2011-01-01 00:00:00.000	2023-10-22 17:17:53.360	C:/Users/pramit/Downloads/Food_Facility_Inspe
1012012	1	Sunday	0	January	2012	1	1	2012-01-01 00:00:00.000	2023-10-22 17:17:53.690	C:/Users/pramit/Downloads/Food_Facility_Inspe
1012013	1	Tuesday	0	January	2013	1	1	2013-01-01 00:00:00.000	2023-10-22 17:17:53.980	C:/Users/pramit/Downloads/Food_Facility_Inspe
1012014	1	Wednesday	0	January	2014	1	1	2014-01-01 00:00:00.000	2023-10-22 17:17:54.270	C:/Users/pramit/Downloads/Food_Facility_Inspe
1012015	1	Thursday	0	January	2015	1	1	2015-01-01 00:00:00.000	2023-10-22 17:17:54.523	C:/Users/pramit/Downloads/Food_Facility_Inspe
1012016	1	Friday	0	January	2016	1	1	2016-01-01 00:00:00.000	2023-10-22 17:17:54.830	C:/Users/pramit/Downloads/Food_Facility_Inspe
1012017	1	Sunday	0	January	2017	1	1	2017-01-01 00:00:00.000	2023-10-22 17:17:55.113	C:/Users/pramit/Downloads/Food_Facility_Inspe
1012018	1	Monday	0	January	2018	1	1	2018-01-01 00:00:00.000	2023-10-22 17:17:55.393	C:/Users/pramit/Downloads/Food_Facility_Inspe

Row Count:

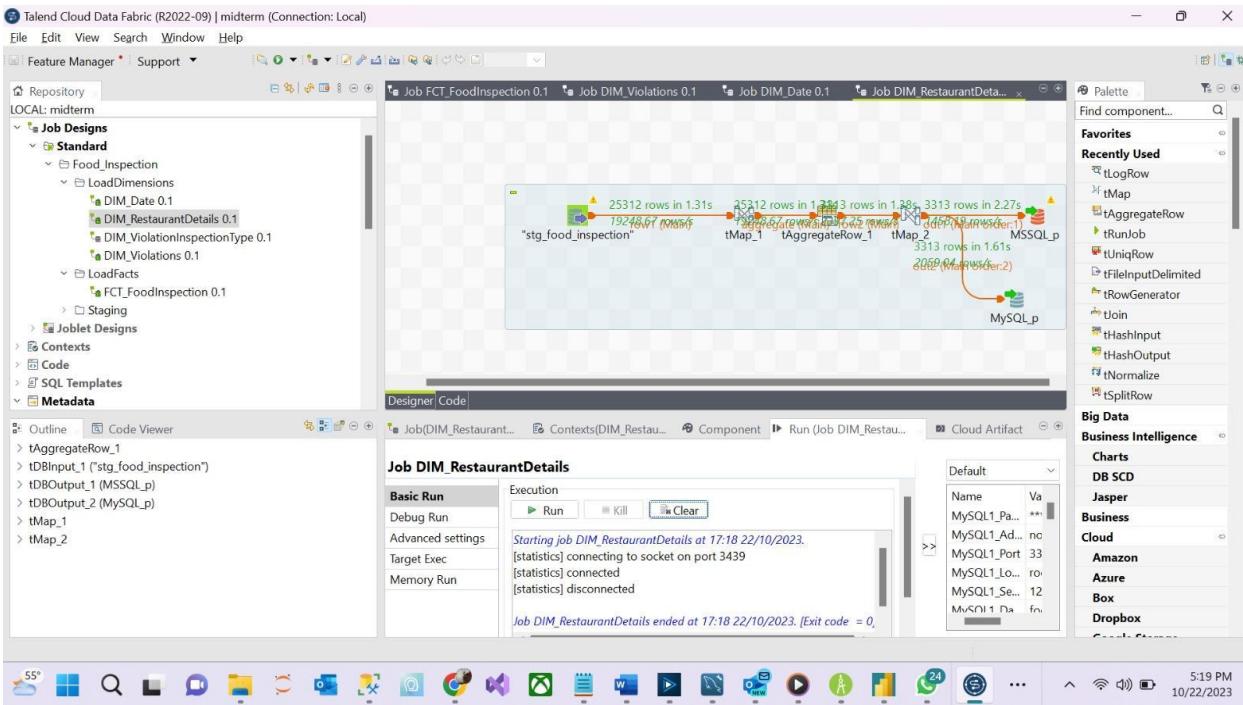
The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the 'food_inspection' database is selected. In the center pane, a query window displays a script for counting rows. Below the script, the results grid shows a single row with a count of 10000. The status bar at the bottom indicates the query was executed successfully.

```

SELECT count(*) as count_date
  FROM [food_inspection].[dbo].[DIM_Date]
  
```

count_date
10000

Loading Restaurant Dimension:



Purpose and Business Logic: The staging table is taken as an input and fed into the Tmap. The output from Tmap is taken as an input to TaggregatorRow so that we can perform function similar to SQL groupby. It receives the flow and aggregates it based on Name, Phone no, Address, City, State, Zipcode, Latitude and Longitude. We have taken the Latitude and Longitude as separate rows to have a more well defined grain. The output is fed into another row fed into the dbs.

Table in MySQL:

The screenshot shows the MySQL Workbench interface. The left sidebar shows the database structure with tables like dim_date, dim_restaurant, dimViolation, and dimViolationInspected. The main area shows the 'dim_restaurant' table with 10 rows of data. The table structure includes columns for RestaurantSK, Name, PhoneNumber, Address, City, State, Zipcode, Latitude, Longitude, DL_CreatedDate, DL_WorkflowFileName, and DL_Workflow_ProcessID. The data grid shows various restaurants like Acre Coffee, Pasteles Cristina Bakery, and Phyllis Giant Burgers, along with their addresses and coordinates. A tooltip on the right indicates that automatic context help is disabled and suggests using the toolbar for help.

Row count:

The screenshot shows the MySQL Workbench interface. In the top-left pane, the 'Schemas' tree is visible, showing tables like dim_date, dim_restaurant, dim_violation, and dim_violationinspec. The main pane displays a query results grid with one row containing the value '3313'. A status message on the right says: 'Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.'

```
SELECT count(*) FROM food_inspection.dim_restaurant;
```

count(*)
3313

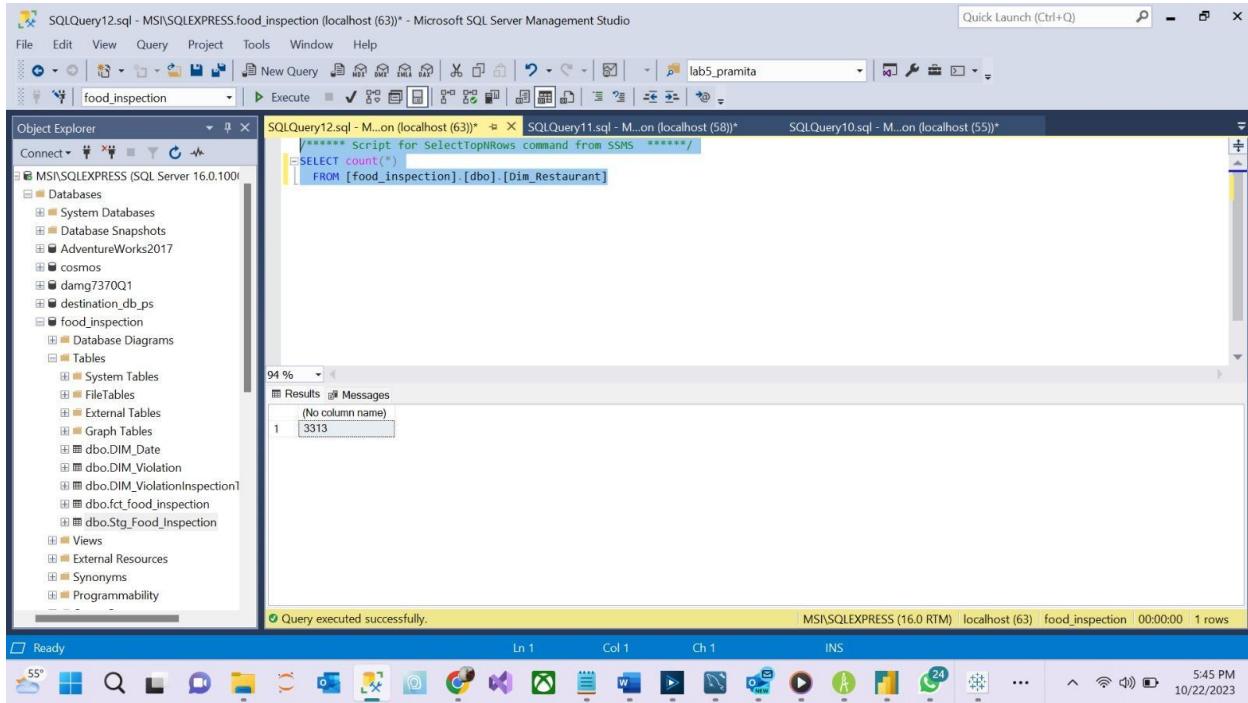
Table in MsSQL:

The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. The Object Explorer on the left shows databases like MSN\SQLEXPRESS and AdventureWorks2017. The central pane displays the results of a query against the Dim_Restaurant table, showing 3313 rows. The results grid includes columns: RestaurantSK, Name, PhoneNumber, Address, City, State, Zipcode, Latitude, Longitude, and DL_CreatedDate.

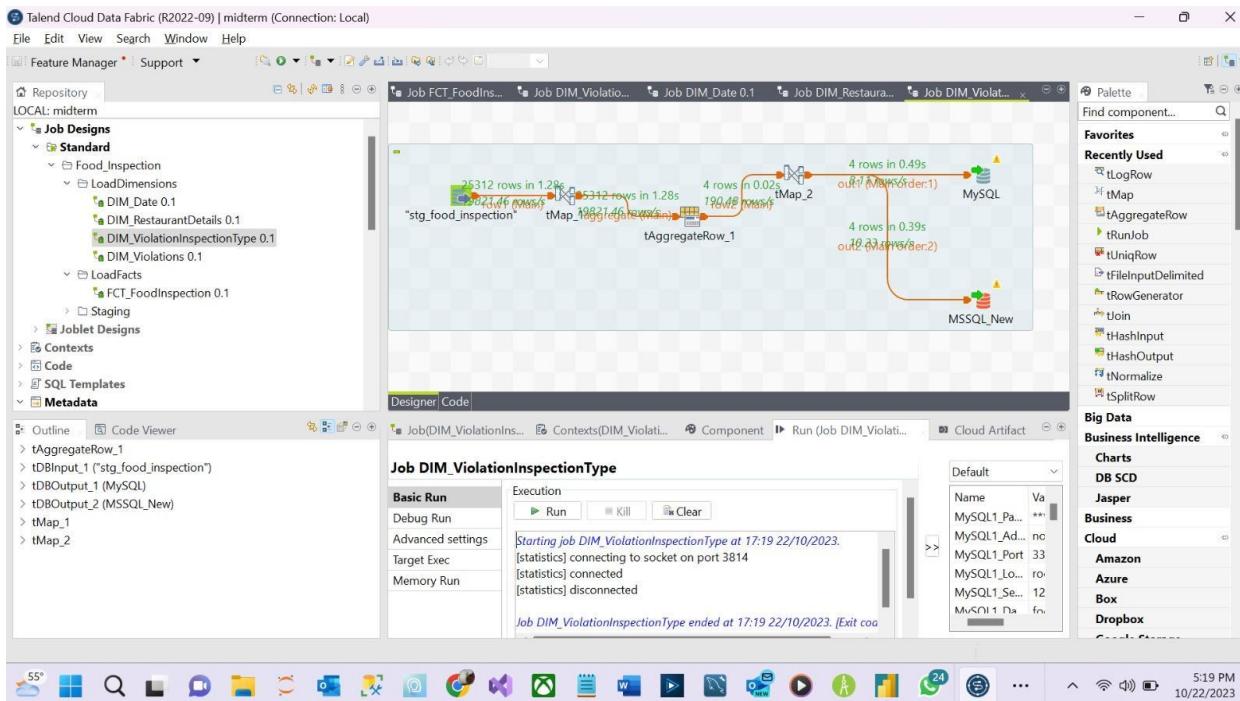
```
SELECT * FROM [food_inspection].[dbo].[Dim_Restaurant]
```

RestaurantSK	Name	PhoneNumber	Address	City	State	Zipcode	Latitude	Longitude	DL_CreatedDate
1	Acre Coffee	1415613729	621 4th St	Santa Rosa	CA	95404	38.4406630521	-122.7136648956	2023-10-22 17:18:47.4f
2	Pasteles Cristina Bakery	17075263055	320 W 3rd St #E	Santa Rosa	CA	95401	-1.0	(-1.0)	2023-10-22 17:18:47.4f
3	Phyllis Giant Burgers	17075384000	4910 Sonoma Hwy Ste B	Santa Rosa	CA	95409	38.4637213321	-122.6650818909	2023-10-22 17:18:47.4f
4	Rose Catering, Inc	17075849153	600 Martin Ave #105	Rohnert Park	CA	94928	-1.0	(-1.0)	2023-10-22 17:18:47.4f
5	Whole Foods Market	17078299801	6910 McKinley St	Sebastopol	CA	95472	38.403565816	-122.8237491806	2023-10-22 17:18:47.4f
6	Thumbs Up Burger & Deli	17078377443	8465 Old Redwood Hwy Ste 300	Windsor	CA	95492	38.5435142922	-122.8007296992	2023-10-22 17:18:47.4f
7	California Bagel, Deli & Cafe	17075382765	124 Calistoga Rd #B	Santa Rosa	CA	95409	-1.0	(-1.0)	2023-10-22 17:18:47.4f
8	Citi Coffee	17075665568	3975 Old Redwood Hwy	Santa Rosa	CA	95403	38.4842532231	-122.7349232329	2023-10-22 17:18:47.4f
9	Piner High School	17075285359	1700 Fulton Rd	Santa Rosa	CA	95403	38.4609136388	-122.7696069472	2023-10-22 17:18:47.4f
10	Chevron Fast Lane Gas & Food	17075751810	136 College Ave	Santa Rosa	CA	95401	38.4457531825	-122.7249410154	2023-10-22 17:18:47.4f

Row Count:



Loading Violation Inspection Type Dimension:



Purpose and Business Logic: The input is taken from stg table and fed into Tmap and Taggregaterow subsequently. We have fed Inspection type in the Taggregaterow to be taken as the field to be grouped. It is being fed as an input to tmap and further to the DBs. The output we receive from this dimension is the types of inspection which are: Routine, Followup, Complaint and Null.

We have done this so that we will get the information about the types of inspection and further used in visualization as shown below.

Table in MySQL:

The screenshot shows the MySQL Workbench interface with the following details:

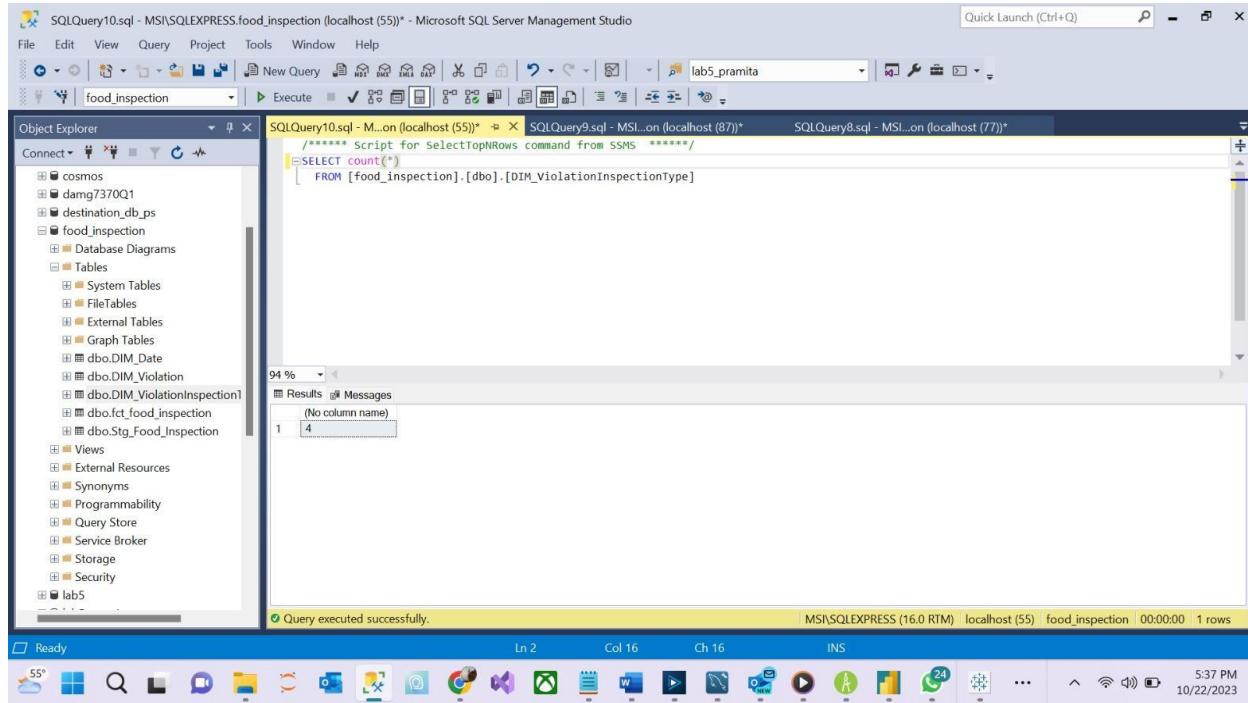
- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Schemas:** Local instance MySQL80, im_restaurant, fct_food_inspection, dim_violation, dim_restaurant, dim_date, dim_restaurant, dim_violation, dim_violationinspected.
- Query Editor:** A query window titled "1 • SELECT * FROM food_inspection.dim_violationinspectiontype;" is open.
- Result Grid:** Shows the results of the query, which are identical to the ones shown in the MSSQL screenshot below.
- Output Window:** Displays the execution log with 68 entries, showing the time, action, message, and duration for each query.
- System Bar:** Includes icons for file operations, search, and system status.

Table in MsSQL:

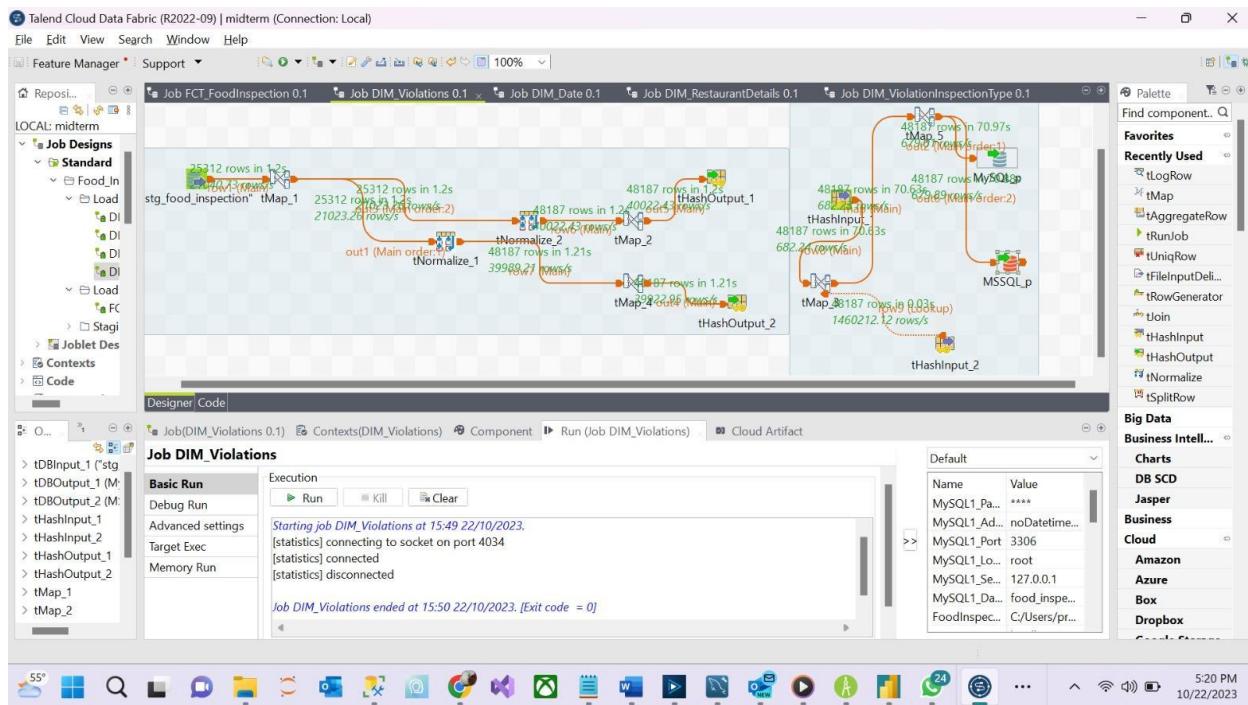
The screenshot shows the Microsoft SQL Server Management Studio interface with the following details:

- File Bar:** File, Edit, View, Query, Project, Tools, Window, Help.
- Object Explorer:** Shows the database structure, including the food_inspection database and its tables.
- Query Editor:** A query window titled "SQLQuery10.sql - MSL\SQLEXPRESS.food_inspection (localhost (55)) - Microsoft SQL Server Management Studio" is open, displaying the same query as the MySQL screenshot.
- Results Grid:** Shows the results of the query, which are identical to the ones shown in the MySQL screenshot above.
- System Bar:** Includes icons for file operations, search, and system status.

Row Count:



Loading Violation Dimension:



Purpose and Business Logic: We have used 2 Tnormalize components to normalize 2 multivalue columns: ViolationCode and ViolationDescription. On normalizing both the columns, we have passed the normalized values to Hashoutput 1 and Hashoutput2. Further, we are feeding the outputs to Hashinputs. We have then used Tmap to merge the data into the single table and store it in the DB. We have done this to normalize the

multivalued columns and store the data into one single table and were required to get the ViolationCategory and ViolationScore from the merged values.

Table in MySQL:

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the database schema with tables: dim_restaurant, dim_date, dim_violation, dim_violationinspectiontype, fct_food_inspection, and dim_violation.
- SQL Editor:** A query window with the SQL command: `SELECT * FROM food_inspection.dim_violation;`
- Result Grid:** Displays the results of the query, showing 5 rows of data with columns: ViolationCodeID, InspectionId, ViolationCodes, ViolationDescriptions, ViolationCategory, and ViolationPoint.
- Log:** Shows the execution history with 7 entries, each detailing a query execution time and message.
- Object Info:** Shows the detailed structure of the dim_violation table, including columns: ViolationCodeID (PK), InspectionId, ViolationCodes, ViolationDescriptions, ViolationCategory, and ViolationPoint.

Row Count:

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the database schema with tables: inspection, dim_restaurant, fct_food_inspection, dim_violation, dim_restaurant, dim_date, dim_restaurant, and dim_violation.
- SQL Editor:** A query window with the SQL command: `SELECT count(*) FROM food_inspection.dim_violation;`
- Result Grid:** Displays the result of the query, showing 1 row with the value 48187.
- Log:** Shows the execution history with 3 entries, including an error message at entry 66.
- Object Info:** Shows the detailed structure of the dim_violation table, including columns: ViolationCodeID (PK), InspectionId, ViolationCodes, ViolationDescriptions, ViolationCategory, and ViolationPoint.

Table in MsSQL:

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the database 'food_inspection' is selected. In the center pane, a query window displays the following SQL script:

```
***** Script for SelectTopNRows command from SSMS *****/
SELECT TOP (1000) [ViolationCodeID]
      ,[InspectionId]
      ,[ViolationCodes]
      ,[ViolationDescriptions]
      ,[ViolationCategory]
      ,[ViolationPoint]
  FROM [food_inspection].[dbo].[DIM_Violation]
```

The results grid shows 1,000 rows of data from the 'DIM_Violation' table. The columns are: ViolationCodeID, InspectionId, ViolationCodes, ViolationDescriptions, ViolationCategory, and ViolationPoint. A status bar at the bottom indicates "Query executed successfully." and "1,000 rows".

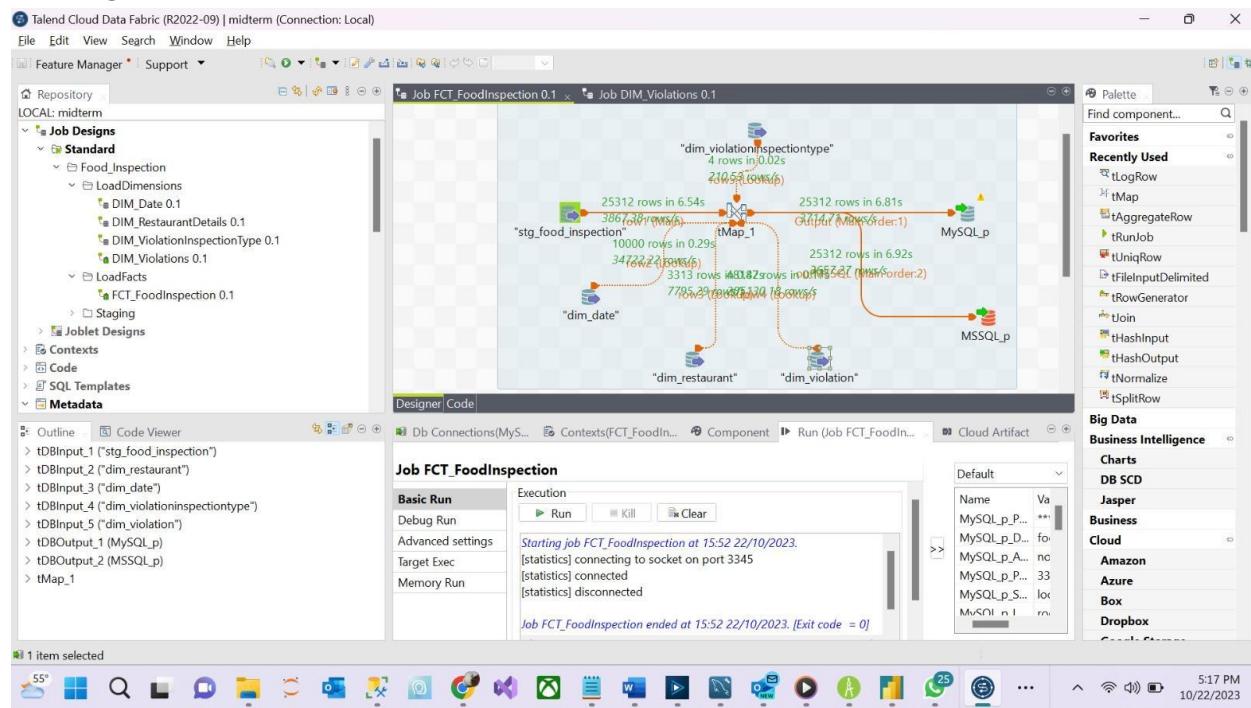
Row Count:

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the database 'food_inspection' is selected. In the center pane, a query window displays the following SQL script:

```
***** Script for SelectTopNRows command from SSMS *****/
SELECT count(*)
  FROM [food_inspection].[dbo].[DIM_Violation]
```

The results grid shows a single row with the value 48187. A status bar at the bottom indicates "Query executed successfully." and "1 rows".

Loading Fact table:



Purpose and Business Logic: The purpose of the fact table is to store all the Primary Keys together in a table as a Surrogate key and organize all the quantitative data in order to provide a solid foundation for decision-making in a business intelligence. We have used the inputs from staging table, and all the dimensions such as Date dimension, Violationinspectiontype dimension, violation dimension, restaurant inspection. We have fed all of them to tmap which is fed to the DBs. As the final output in the DB, we get FoodInspection SK, ViolationCodeID, Restaurant SK, Date SK, InspectionType SK, Violation score which is the addition of all the violation category, Inspection result which is pass or fail based on If violation score is between 0 and 60 is PASS and violation score > 60 is FAIL, InspectionID and BusinessID.

Table in MySQL:

The screenshot shows MySQL Workbench. The left sidebar shows the database schema with tables like 'dim_date', 'dim_restaurant', 'dim_violation', and 'fct_food_inspection'. The main area shows the 'fct_food_inspection' table with the following columns: FoodInspectionSK, ViolationCodeID, RestaurantSK, DateSK, InspectionTypeSK, ViolationScore, InspectionResult, InspectionID, BusinessID, and ResultGrid. The result grid displays 7 rows of data. Below the table, the 'Output' tab shows the execution history of several SQL queries, including SELECT statements and a COUNT(*) query. The bottom right corner shows the Windows taskbar with the date and time.

Row Count:

The screenshot shows the MySQL Workbench interface. In the top navigation bar, the database is set to "Local instance MySQL80". The main pane displays a query in the SQL editor:

```
1 • SELECT count(*) FROM food_inspection.fct_food_inspection;
```

The results grid shows a single row with the value "25312". Below the results, the "Logs" tab shows the execution history:

#	Time	Action	Message	Duration / Fetch
71	17:34:19	SELECT * FROM food_inspection.dimViolation LIMIT 0, 1000	1000 row(s) returned	0.000 sec / 0.000 sec
72	17:34:39	SELECT count(*) FROM food_inspection.dimViolation LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
73	17:34:55	SELECT * FROM food_inspection.stgFoodInspection LIMIT 0, 1000	1000 row(s) returned	0.000 sec / 0.015 sec
74	17:35:07	SELECT count(*) FROM food_inspection.stgFoodInspection LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.000 sec
75	18:51:07	SELECT count(*) FROM food_inspection.fctFoodInspection LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

Table in MsSQL:

The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. The Object Explorer on the left shows the database structure, including the "food_inspection" database and its tables.

In the center, a query window displays the following SQL script and results:

```
***** Script for SelectTopNRows command from SSMS *****/
SELECT TOP (1000) [FoodInspectionSK]
,[ViolationCodeID]
,[RestaurantSK]
,[DateSK]
,[InspectionTypeSK]
,[ViolationScore]
,[InspectionResult]
,[InspectionID]
,[BusinessID]
,[DI_CreatedDate]
,[DI_WorkflowFileName]
,[DI_Workflow_ProcessID]
FROM [food_inspection].[dbo].[fct_food_inspection]
```

The results grid shows 10 rows of data from the "fct_food_inspection" table:

FoodInspectionSK	ViolationCodeID	RestaurantSK	DateSK	InspectionTypeSK	ViolationScore	InspectionResult	InspectionID	BusinessID	DI_CreatedDate	DI_WorkflowFileName
1	2	2922	0	4	0	PASS	DA	PR0017321	2023-10-22 15:52:07.573	food_inspection
2	4	61	8092016	1	0	PASS	DA0004NLT	PR000722	2023-10-22 15:52:07.580	food_inspection
3	6	4	565	9082017	10	PASS	DA000K0IO	PR0020278	2023-10-22 15:52:07.583	food_inspection
4	8	5	1708	4082018	1	PASS	DA0036FLR	PR0014451	2023-10-22 15:52:07.587	food_inspection
5	10	8	2536	4092017	1	PASS	DA0058SQQ	PR0017756	2023-10-22 15:52:07.590	food_inspection
6	12	9	428	4092016	1	PASS	DA00BEEGO	PR0017167	2023-10-22 15:52:07.593	food_inspection
7	14	10	1340	3082018	1	PASS	DA00C0QJO	PR0001795	2023-10-22 15:52:07.593	food_inspection
8	16	11	2560	7042017	1	PASS	DA00E9QDJ	PR0001591	2023-10-22 15:52:07.597	food_inspection
9	18	12	2896	8012016	1	PASS	DA00FATI	PR0020537	2023-10-22 15:52:07.597	food_inspection
10	20	13	2339	4122018	1	PASS	DA008ZEV	PR0000573	2023-10-22 15:52:07.600	food_inspection

At the bottom, a message indicates the query was executed successfully.

Row count:

The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. On the left, the Object Explorer pane displays the database structure of 'food_inspection', including tables like 'cosmios', 'damg7370Q1', 'destination_db_ps', and 'food_inspection'. The 'food_inspection' table is expanded to show its columns: 'ID', 'InspectionDate', 'Location', 'InspectedBy', 'Status', 'ViolationCount', and 'ViolationDescription'. In the center, a query results window is open, showing the output of a SELECT COUNT(*) query from the 'food_inspection' table. The result is a single row with a value of 25312. The status bar at the bottom indicates the query was executed successfully.

```
SELECT count(*)
FROM [food_inspection].[dbo].[fct_food_inspection]
```

(No column name)
1
25312

Query executed successfully.