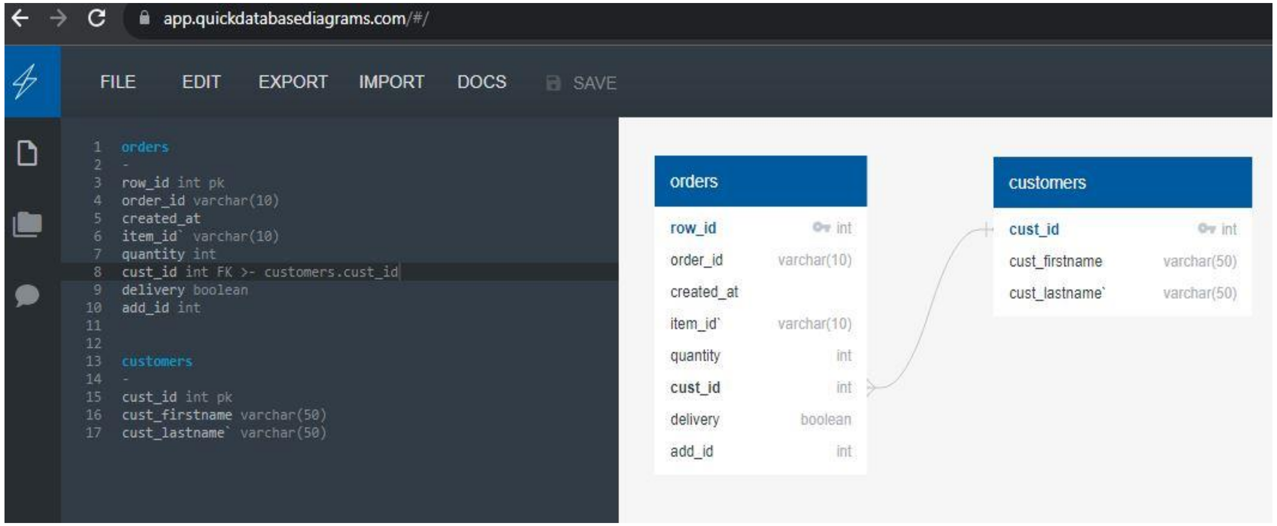
IMPLEMENTATION STEPS

[A] Create database schema in MySQL:

1. I used Quick DBD which allows for defining DB table names, field names, data type, primary keys and relationships to other tables. This essentially auto-generates database design diagram and creates table schemas which I ran in terminal.



```
orders
-
row_id int pk
order_id varchar(10)
created_at
item_id` varchar(10)
quantity int
cust_id int FK >- customers.cust_id
delivery boolean
add_id int


customers
-
cust_id int pk
cust_firstname varchar(50)
cust_lastname` varchar(50)
```

2. Next step was to load csv files into respective tables using below syntax:

```
LOAD DATA LOCAL INFILE '/home/training/Downloads/Pizzeria/rota.csv' INTO TABLE rota

FIELDS TERMINATED BY ','

ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 LINES;
```

This block of code specifies the path to get the csv file, how columns are terminated (by commas in this case) and when to load a new row record (‘\n’). ’Ignore 1 Lines’ instructs the load process to ignore first row as these are where field names are stored in the csv.

[B] Build Hive database on top of MySQL data:

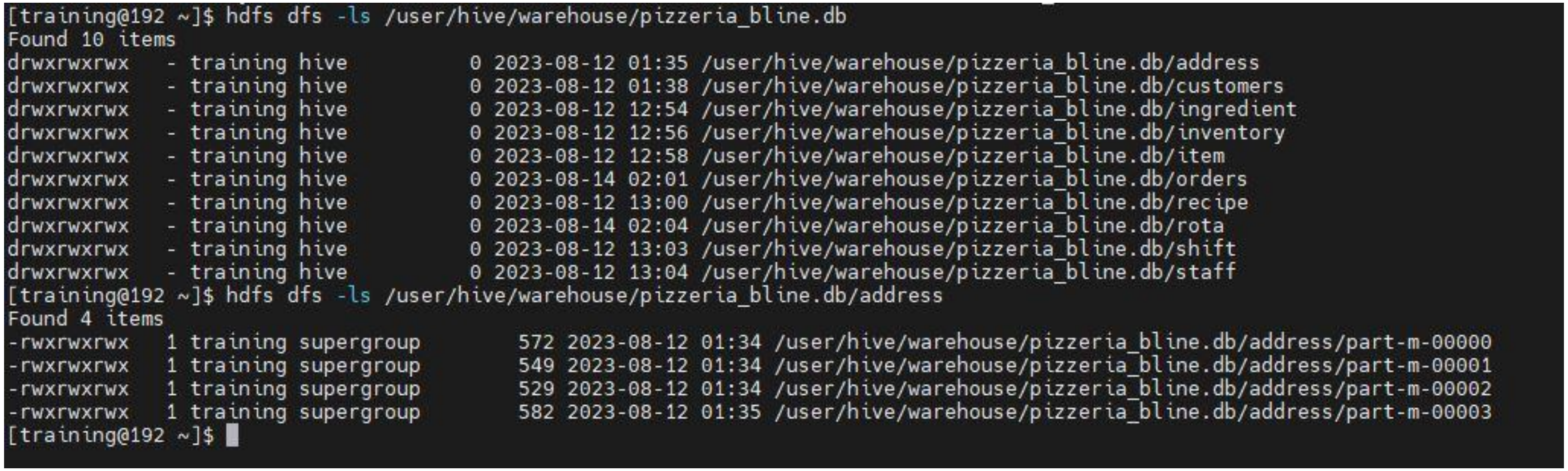
3. Upon loading data, I did a quick inspection to verify fields have accurate data and then proceeded to ingest tables to Hadoop hdfs. This was achieved with sqoop using syntax:

```
sqoop import \

--connect jdbc:mysql://localhost/Pizzeria \

--username training \

--password training \

--fields-terminated-by '\t' \

--table address \

--hive-import \

--hive-database pizzeria_bline
```

The sqoop command specifies the connection to MySQL database i.e., connect to Pizzeria DB.

This block is ingesting the 'address' table from MySQL into hive into the Hive pizzeria_bline database. By default, the corresponding hdfs files are created in Hadoop.

Confirmation that files are stored in Hadoop distributed file system is seen below.



```
[training@192 ~]$ hdfs dfs -ls /user/hive/warehouse/pizzeria_bline.db
Found 10 items
drwxrwxrwx   - training hive          0 2023-08-12 01:35 /user/hive/warehouse/pizzeria_bline.db/address
drwxrwxrwx   - training hive          0 2023-08-12 01:38 /user/hive/warehouse/pizzeria_bline.db/customers
drwxrwxrwx   - training hive          0 2023-08-12 12:54 /user/hive/warehouse/pizzeria_bline.db/ingredient
drwxrwxrwx   - training hive          0 2023-08-12 12:56 /user/hive/warehouse/pizzeria_bline.db/inventory
drwxrwxrwx   - training hive          0 2023-08-12 12:58 /user/hive/warehouse/pizzeria_bline.db/item
drwxrwxrwx   - training hive          0 2023-08-14 02:01 /user/hive/warehouse/pizzeria_bline.db/orders
drwxrwxrwx   - training hive          0 2023-08-12 13:00 /user/hive/warehouse/pizzeria_bline.db/recipe
drwxrwxrwx   - training hive          0 2023-08-14 02:04 /user/hive/warehouse/pizzeria_bline.db/rota
drwxrwxrwx   - training hive          0 2023-08-12 13:03 /user/hive/warehouse/pizzeria_bline.db/shift
drwxrwxrwx   - training hive          0 2023-08-12 13:04 /user/hive/warehouse/pizzeria_bline.db/staff
[training@192 ~]$ hdfs dfs -ls /user/hive/warehouse/pizzeria_bline.db/address
Found 4 items
-rwxrwxrwx   1 training supergroup        572 2023-08-12 01:34 /user/hive/warehouse/pizzeria_bline.db/address/part-m-00000
-rwxrwxrwx   1 training supergroup        549 2023-08-12 01:34 /user/hive/warehouse/pizzeria_bline.db/address/part-m-00001
-rwxrwxrwx   1 training supergroup        529 2023-08-12 01:34 /user/hive/warehouse/pizzeria_bline.db/address/part-m-00002
-rwxrwxrwx   1 training supergroup        582 2023-08-12 01:35 /user/hive/warehouse/pizzeria_bline.db/address/part-m-00003
[training@192 ~]$ █
```

[C] Develop Hive views for reporting:

4. For the Order Activity dashboard, view was created using below statement in Hive

```
CREATE VIEW Order_Activity AS
SELECT
        o.order_id,
        i.item_price,
        o.quantity,
        i.item_cat,
        i.item_name,
        o.created_at,
        a.delivery_address1,
        a.delivery_address2,
        a.delivery_city,
        a.delivery_zipcode,
        o.delivery
FROM
        orders o
        LEFT JOIN item i ON o.item_id = i.item_id
        LEFT JOIN address a ON o.add_id = a.add_id;
```

The HiveQL statement allows me to achieve the dashboard requirements by joining the required fields on orders, item and address tables.

```
CREATE VIEW Inv_Mgmt1 AS
SELECT
        s1.Menu,
        s1.ing_id,
        s1.ing_name,
        s1.ing_weight,
        s1.ing_price,
        s1.order_quantity,
        s1.recipe_quantity,
        s1.order_quantity*s1.recipe_quantity as ordered_weight,
        s1.ing_price/s1.ing_weight as unit_cost,
        (s1.order_quantity*s1.recipe_quantity)*(s1.ing_price/s1.ing_weight)
as ingredient_cost
FROM (SELECT
        o.item_id,
        i.sku,
        i.item_name AS Menu,
        r.ing_id,
        ing.ing_name,
        r.quantity AS recipe_quantity,
        sum(o.quantity) AS order_quantity,
        ing.ing_weight,
        ing.ing_price
FROM orders o
        LEFT JOIN item i ON o.item_id = i.item_id
        LEFT JOIN recipe r ON i.sku = r.recipe_id
        LEFT JOIN ingredient ing ON ing.ing_id = r.ing_id
GROUP BY
        o.item_id,
        i.sku,
        i.item_name,
        r.ing_id,
        r.quantity,
        ing.ing_name,
        ing.ing_weight,
        ing.ing_price) AS s1;
```

5. For the Inventory Management Dashboard, view was created using this statement in Hive. The subquery is written/built on existing statements as inventory dashboard requirements are fulfilled. The breakdown is shown in highlighted parts.

QUERY BREAK DOWN.

I. This query was written first to calculate order quantity per pizza by JOINING the orders and item tables. Total ordered quantity was aggregated as SUM of order_quantity to give total number of orders per pizza.

II. The next addition to the query breaks down pizza by ingredients from the recipe table by JOINING orders to recipe table

III. This section of the query adds the ingredient name, price and weight from the ingredient table by JOINING orders to ingredient table.

IV. To calculate ordered weight, product of order_quantity and recipe_quantity is needed. However, as order_quantity is already an aggregated field, it can not be used as a calculation inn same SELECT statement.

To resolve this, the statement was turned to a sub-query AS s1.

V. With the sub-query created, s1 can be queried with field names as seen in the un-highlighted parts and I can run calculation on the aggregated field order_quantity.

Hence, new fields are derived as

ordered_weight [order qty x recipe qty]

unit_cost [ing price/ing weight]

ingredient cost [ordered_weight x (ing price/ing weight)]

Another view was created for remaining inventory dashboard requirement to simplify the statement development process and aliased AS Inv_Mgmt2

```
CREATE VIEW Inv_Mgmt2 AS

SELECT

s2.ing_name,

s2.ordered_weight,

ing.ing_weight,

inv.quantity,

ing.ing_weight*inv.quantity AS total_inv_weight,

(ing.ing_weight*inv.quantity)-s2.ordered_weight AS remaining_weight

FROM

        ( SELECT ing_id, ing_name, sum(ordered_weight) AS ordered_weight
FROM Inv_Mgmt1 GROUP BY ing_name, ing_id ) AS s2

        LEFT JOIN inventory inv ON inv.item_id = s2.ing_id

        LEFT JOIN ingredient ing ON ing.ing_id = s2.ing_id;
```

6. To get the required view for percentage stock remaining and List of ingredients to re-order, the new view query's old view to get required fields.
VI. This query gets ingredient name, ordered weight and ingredient name from the Inv_Mgmt view. The result from the view is aliased as s2.

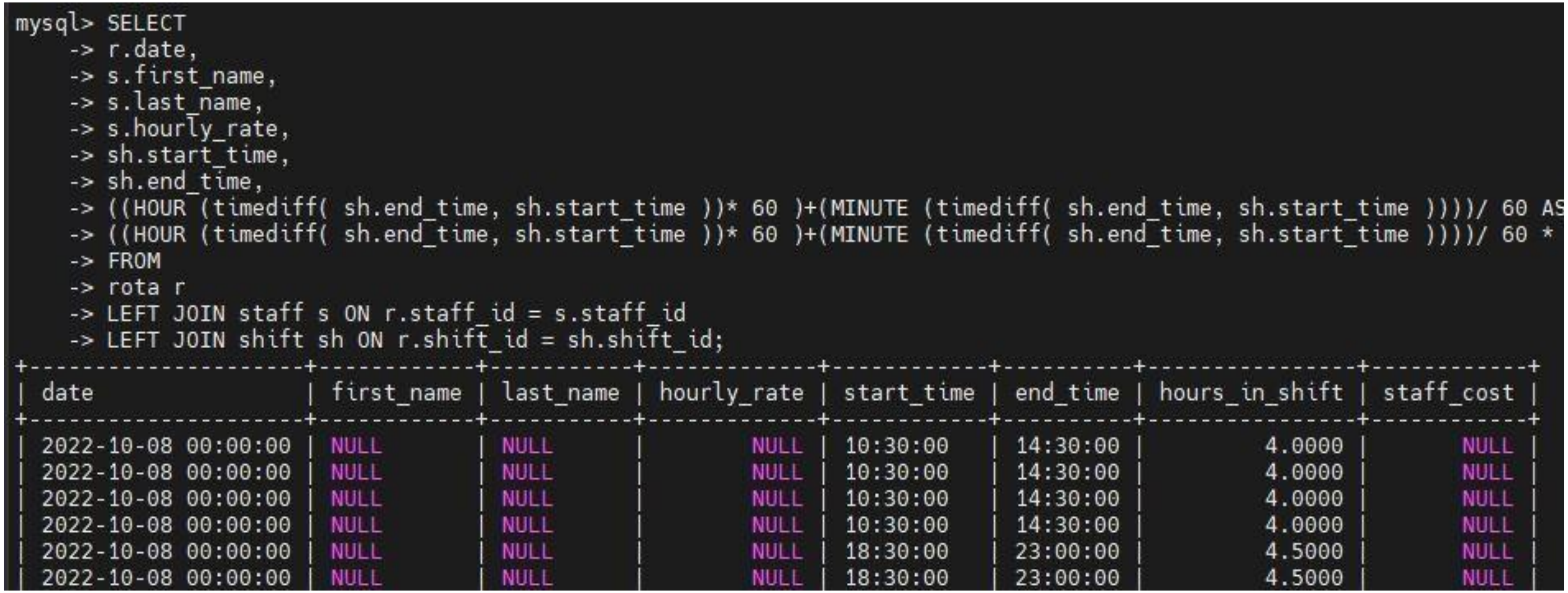
VII. As was done with previous view, s2 was queried to get fields FROM s2 and also calculate new fields as

Total_inv_weight [ing.ing_weight*inv.quantity]

Remaining weight [total_inv_weight – s2.ordered_weight]

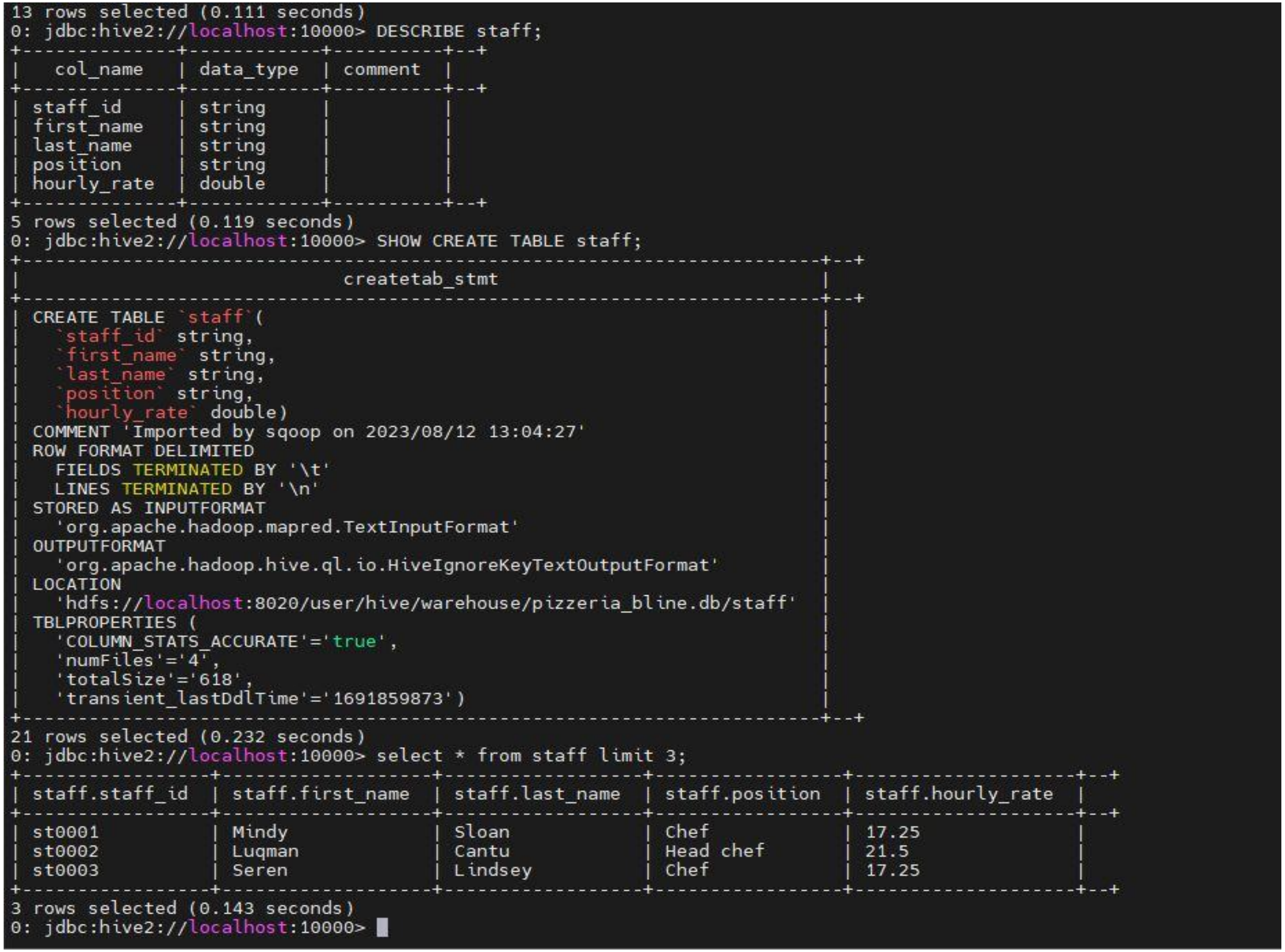
s2 is JOINED to inventory and ingredient tables ON s2's ing_id

7. For staff Management view, I encountered an issue where joining the rota, staff and shift tables produced a result with NULL values for staff names and hourly rate. And since staff cost is a product of hours_in_shift and hourly_rate, that calculated field also retuned as NULL.



```
mysql> SELECT
    -> r.date,
    -> s.first_name,
    -> s.last_name,
    -> s.hourly_rate,
    -> sh.start_time,
    -> sh.end_time,
    -> ((HOUR (timediff( sh.end_time, sh.start_time ))* 60 )+(MINUTE (timediff( sh.end_time, sh.start_time ))))/ 60 AS
    -> ((HOUR (timediff( sh.end_time, sh.start_time ))* 60 )+(MINUTE (timediff( sh.end_time, sh.start_time ))))/ 60 *
    -> FROM
    -> rota r
    -> LEFT JOIN staff s ON r.staff_id = s.staff_id
    -> LEFT JOIN shift sh ON r.shift_id = sh.shift_id;
+---------------------+------------+-----------+-------------+------------+----------+----------------+------------+
| date                | first_name | last_name | hourly_rate | start_time | end_time | hours_in_shift | staff_cost |
+---------------------+------------+-----------+-------------+------------+----------+----------------+------------+
| 2022-10-08 00:00:00 | NULL       | NULL      |        NULL | 10:30:00   | 14:30:00 |         4.0000 |       NULL |
| 2022-10-08 00:00:00 | NULL       | NULL      |        NULL | 10:30:00   | 14:30:00 |         4.0000 |       NULL |
| 2022-10-08 00:00:00 | NULL       | NULL      |        NULL | 10:30:00   | 14:30:00 |         4.0000 |       NULL |
| 2022-10-08 00:00:00 | NULL       | NULL      |        NULL | 10:30:00   | 14:30:00 |         4.0000 |       NULL |
| 2022-10-08 00:00:00 | NULL       | NULL      |        NULL | 18:30:00   | 23:00:00 |         4.5000 |       NULL |
| 2022-10-08 00:00:00 | NULL       | NULL      |        NULL | 18:30:00   | 23:00:00 |         4.5000 |       NULL |
```

I investigated using ‘DESCRIBE’ and ‘SHOW CREATE TABLE’ statements, but couldn’t find any issues there. Querying staff table shows the data both in MySQL and Hive



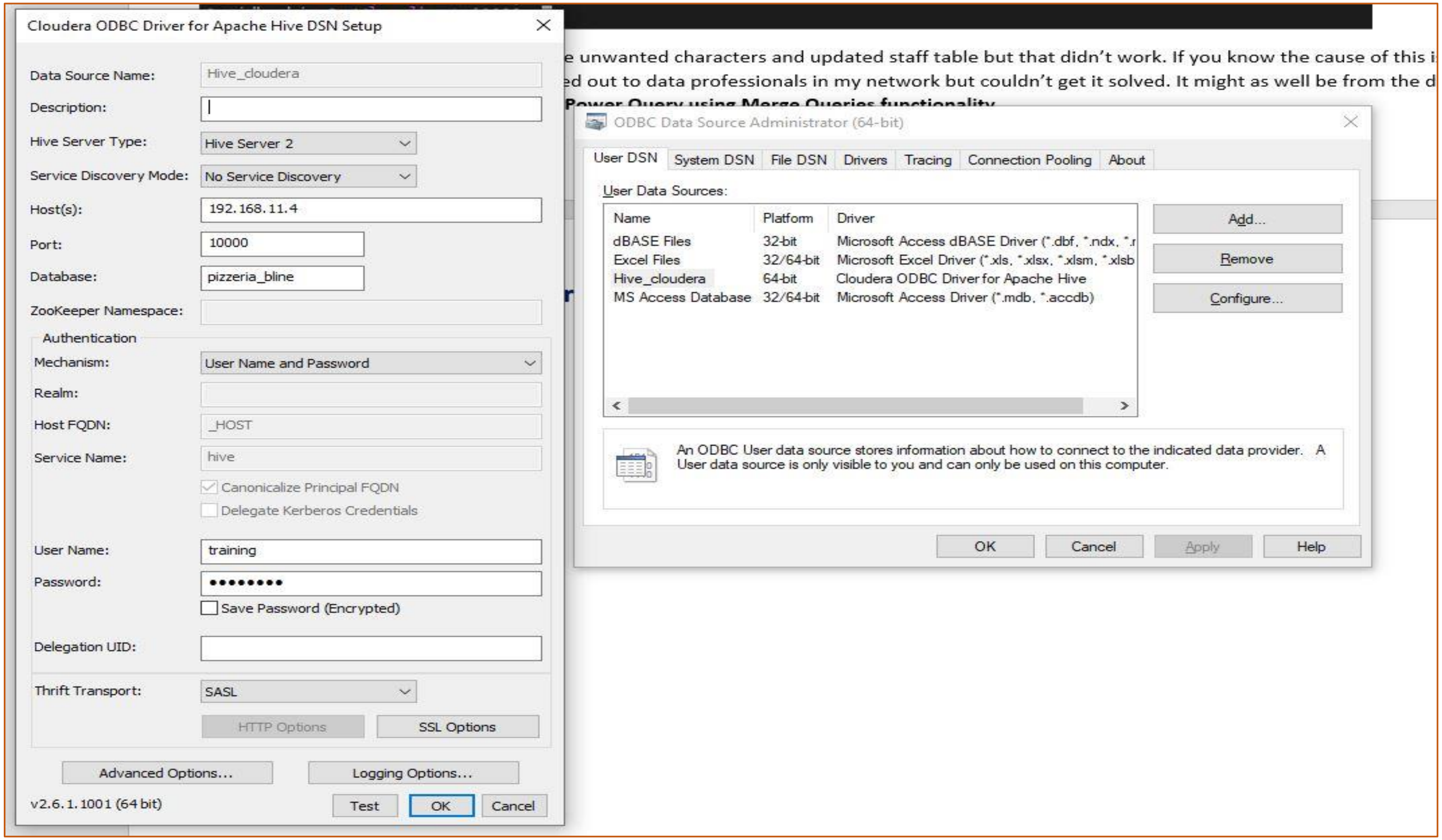
```
13 rows selected (0.111 seconds)
0: jdbc:hive2://localhost:10000> DESCRIBE staff;
+--------------+------------+----------+--+
|   col_name   | data_type  | comment  |
+--------------+------------+----------+--+
| staff_id     | string     |          |
| first_name   | string     |          |
| last_name    | string     |          |
| position     | string     |          |
| hourly_rate  | double     |          |
+--------------+------------+----------+--+
5 rows selected (0.119 seconds)
0: jdbc:hive2://localhost:10000> SHOW CREATE TABLE staff;
+---------------------------------------------------------------------+--+
|                            createtab_stmt                           |
+---------------------------------------------------------------------+--+
| CREATE TABLE `staff`(                                               |
|   `staff_id` string,                                                |
|   `first_name` string,                                              |
|   `last_name` string,                                               |
|   `position` string,                                                |
|   `hourly_rate` double)                                             |
| COMMENT 'Imported by sqoop on 2023/08/12 13:04:27'                  |
| ROW FORMAT DELIMITED                                                |
|   FIELDS TERMINATED BY '\t'                                         |
|   LINES TERMINATED BY '\n'                                          |
| STORED AS INPUTFORMAT                                               |
|   'org.apache.hadoop.mapred.TextInputFormat'                        |
| OUTPUTFORMAT                                                        |
|   'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'      |
| LOCATION                                                            |
|   'hdfs://localhost:8020/user/hive/warehouse/pizzeria_bline.db/staff'  |
| TBLPROPERTIES (                                                     |
|   'COLUMN_STATS_ACCURATE'='true',                                   |
|   'numFiles'='4',                                                   |
|   'totalSize'='618',                                                |
|   'transient_lastDdlTime'='1691859873')                             |
+---------------------------------------------------------------------+--+
21 rows selected (0.232 seconds)
0: jdbc:hive2://localhost:10000> select * from staff limit 3;
+-----------------+-------------------+------------------+-----------------+---------------------+--+
| staff.staff_id  | staff.first_name  | staff.last_name  | staff.position  | staff.hourly_rate   |
+-----------------+-------------------+------------------+-----------------+---------------------+--+
| st0001          | Mindy             | Sloan            | Chef            | 17.25               |
| st0002          | Luqman            | Cantu            | Head chef       | 21.5                |
| st0003          | Seren             | Lindsey          | Chef            | 17.25               |
+-----------------+-------------------+------------------+-----------------+---------------------+--+
3 rows selected (0.143 seconds)
0: jdbc:hive2://localhost:10000> 
```

I also tried TRIM() in the csv file to remove unwanted characters and updated staff table but that didn’t work. If you know the cause of this issue, kindly reach out to me, I would love to learn the fix as I also reached out to data professionals in my network but couldn’t get it solved. It might as well be from the data source. **The workaround for this was to perform a JOIN like action in Power Query using Merge Queries functionality**.

[D] Create interactive Power BI dashboards and reports:

8. To connect Hive views in Linux VM to Power BI, I installed the Cloudera ODBC driver, following steps on the left to specify host, port, DB name in Hive and user credentials.

Then on the right, I added Hive_cloudera as a user data source and applied changes.



Cloudera ODBC Driver for Apache Hive DSN Setup

Data Source Name: Hive_cloudera

Description:

Hive Server Type: Hive Server 2

Service Discovery Mode: No Service Discovery

Host(s): 192.168.11.4

Port: 10000

Database: pizzeria_bline

ZooKeeper Namespace:

Authentication

Mechanism: User Name and Password

Realm:

Host FQDN: _HOST

Service Name: hive

☑ Canonicalize Principal FQDN

☐ Delegate Kerberos Credentials

User Name: training

Password: ••••••••

☐ Save Password (Encrypted)

Delegation UID:

Thrift Transport: SASL

v2.6.1.1001 (64 bit)

e unwanted characters and updated staff table but that didn't work. If you know the cause of this i

ed out to data professionals in my network but couldn't get it solved. It might as well be from the d

Power Query using Merge Queries functionality

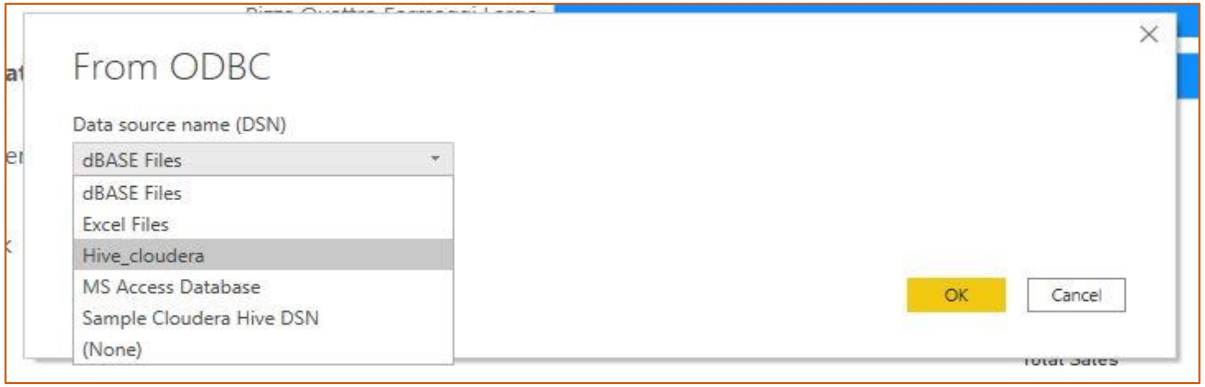
ODBC Data Source Administrator (64-bit)

User Data Sources:

Name	Platform	Driver
dBASE Files	32-bit	Microsoft Access dBASE Driver (*.dbf, *.ndx, *.r
Excel Files	32/64-bit	Microsoft Excel Driver (*.xls, *.xlsx, *.xlsm, *.xlsb
Hive_cloudera	64-bit	Cloudera ODBC Driver for Apache Hive
MS Access Database	32/64-bit	Microsoft Access Driver (*.mdb, *.accdb)

An ODBC User data source stores information about how to connect to the indicated data provider. A User data source is only visible to you and can only be used on this computer.

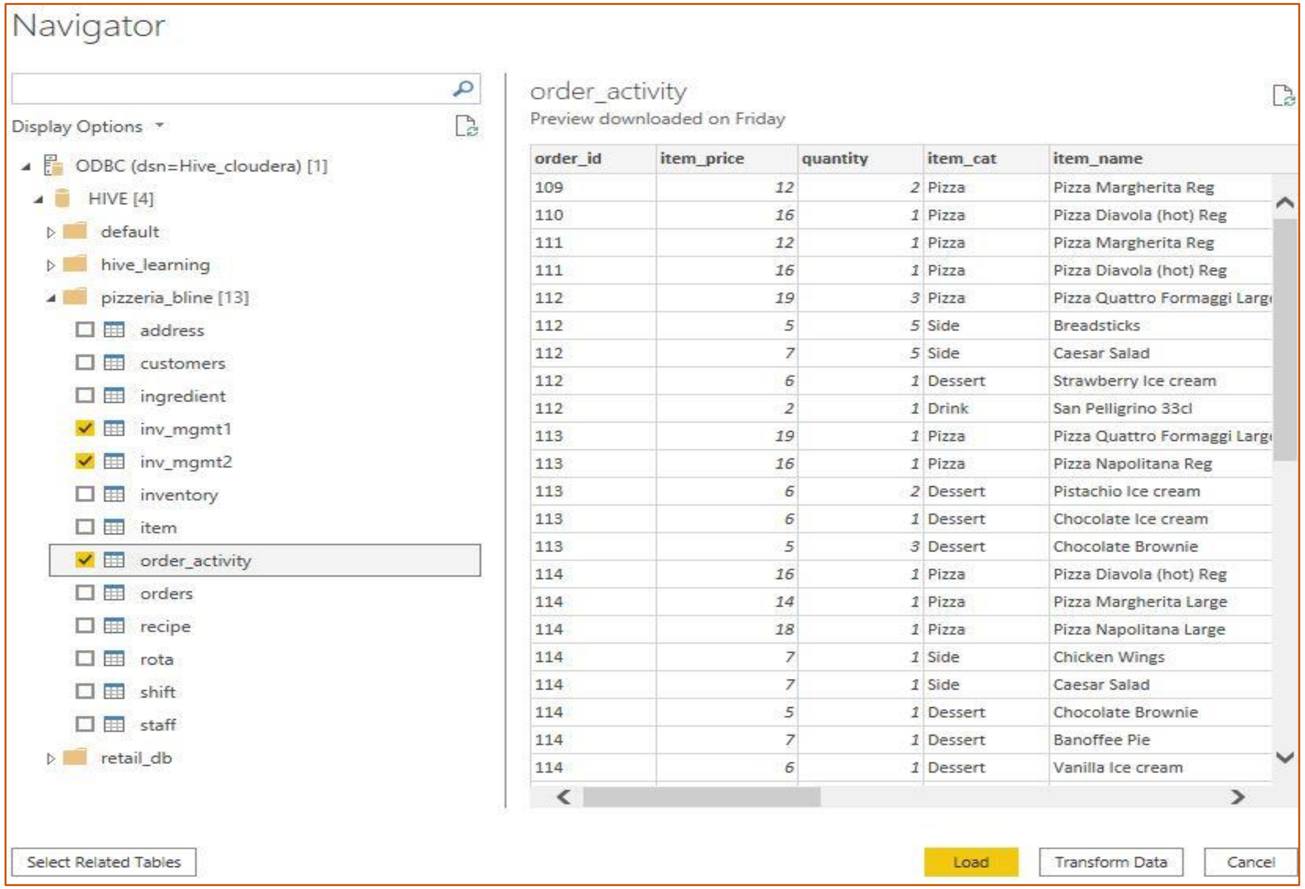
9. The final step of the connection is achieved by: Get Data > Other > ODBC > Hive_cloudera > OK. The views created can be seen in Hive DB name and loaded to Power BI.



From ODBC

Data source name (DSN)

dBASE Files



Navigator

Display Options

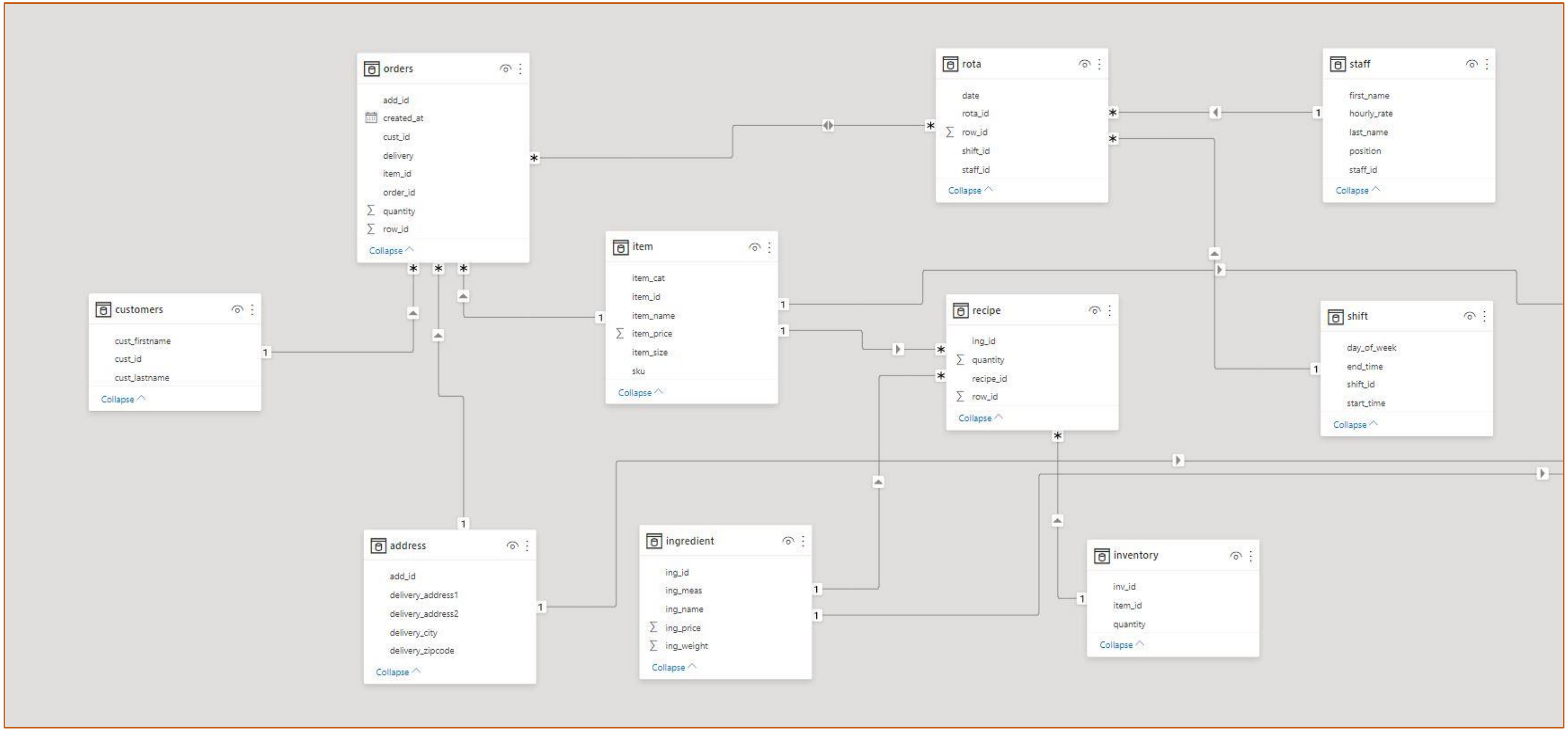
- ODBC (dsn=Hive_cloudera) [1]
  - HIVE [4]
    - default
    - hive_learning
    - pizzeria_bline [13]
      - □ address
      - □ customers
      - □ ingredient
      - ☑ inv_mgmt1
      - ☑ inv_mgmt2
      - □ inventory
      - □ item
      - ☑ order_activity
      - □ orders
      - □ recipe
      - □ rota
      - □ shift
      - □ staff
    - retail_db

order_activity

Preview downloaded on Friday

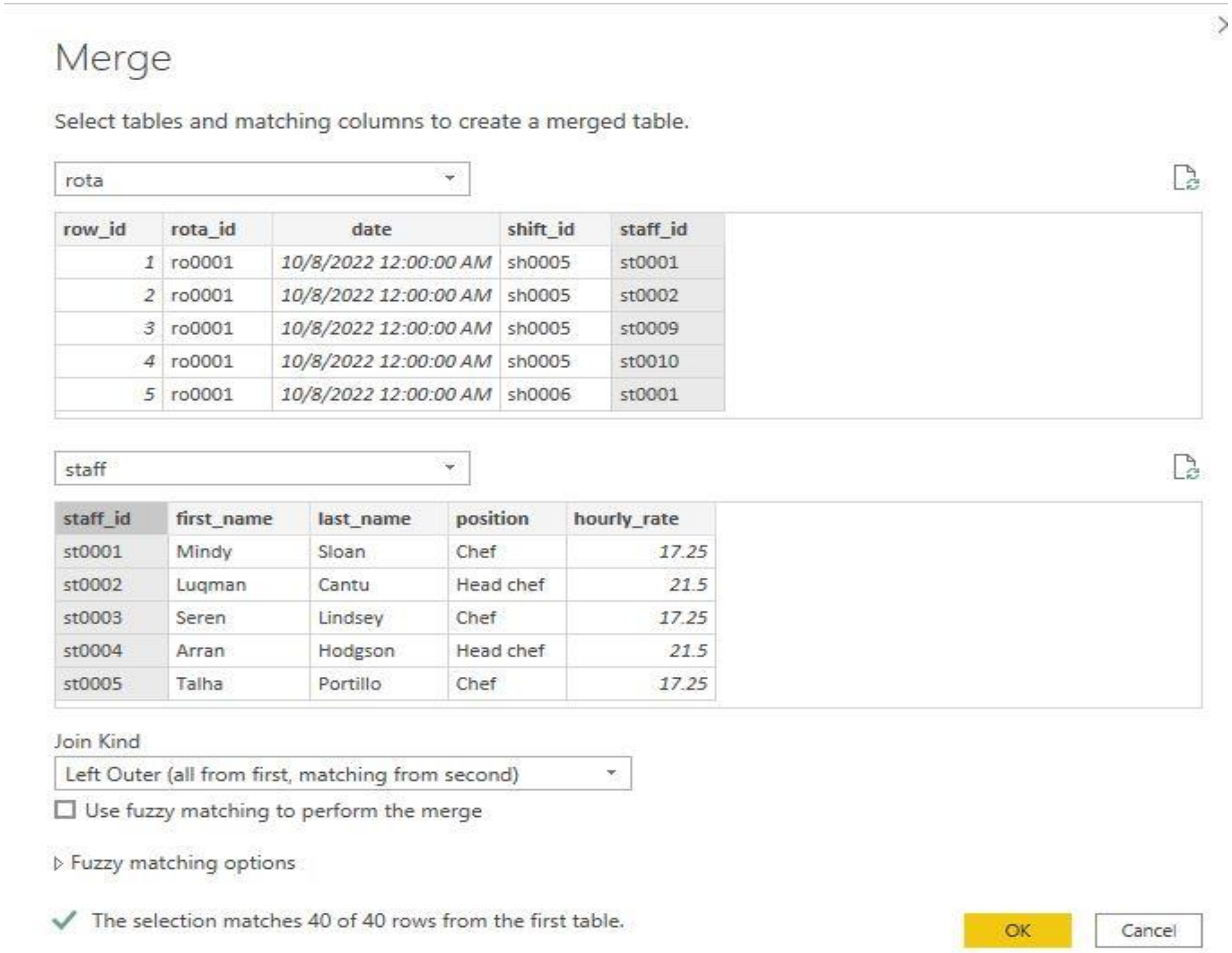
order_id	item_price	quantity	item_cat	item_name
109	12	2	Pizza	Pizza Margherita Reg
110	16	1	Pizza	Pizza Diavola (hot) Reg
111	12	1	Pizza	Pizza Margherita Reg
111	16	1	Pizza	Pizza Diavola (hot) Reg
112	19	3	Pizza	Pizza Quattro Formaggi Larg
112	5	5	Side	Breadsticks
112	7	5	Side	Caesar Salad
112	6	1	Dessert	Strawberry Ice cream
112	2	1	Drink	San Pelligrino 33cl
113	19	1	Pizza	Pizza Quattro Formaggi Larg
113	16	1	Pizza	Pizza Napolitana Reg
113	6	2	Dessert	Pistachio Ice cream
113	6	1	Dessert	Chocolate Ice cream
113	5	3	Dessert	Chocolate Brownie
114	16	1	Pizza	Pizza Diavola (hot) Reg
114	14	1	Pizza	Pizza Margherita Large
114	18	1	Pizza	Pizza Napolitana Large
114	7	1	Side	Chicken Wings
114	7	1	Side	Caesar Salad
114	5	1	Dessert	Chocolate Brownie
114	7	1	Dessert	Banoffee Pie
114	6	1	Dessert	Vanilla Ice cream

10. With my views now in Power BI, I created the required visuals for each dashboard. Nonetheless, I needed to load all tables from Hive and define relationships in Power BI's model so Merge Query functionality can be utilised for the staff management view which wasn't done in HiveQL.



After defining relationships, I check for data fields to ensure the data type are properly identified by the types, i.e. string, Boolean, datetime etc

Since the rota table is the primary table, I want all records to show (i.e., LEFT JOIN TABLE), I start from here. In Power Query Home > Merge Queries > Merge Queries as New, to create the new 'View" table. The table is names staff_mgmt



Merge

Select tables and matching columns to create a merged table.

rota

row_id	rota_id	date	shift_id	staff_id
1	ro0001	10/8/2022 12:00:00 AM	sh0005	st0001
2	ro0001	10/8/2022 12:00:00 AM	sh0005	st0002
3	ro0001	10/8/2022 12:00:00 AM	sh0005	st0009
4	ro0001	10/8/2022 12:00:00 AM	sh0005	st0010
5	ro0001	10/8/2022 12:00:00 AM	sh0006	st0001

staff

staff_id	first_name	last_name	position	hourly_rate
st0001	Mindy	Sloan	Chef	17.25
st0002	Luqman	Cantu	Head chef	21.5
st0003	Seren	Lindsey	Chef	17.25
st0004	Arran	Hodgson	Head chef	21.5
st0005	Talha	Portillo	Chef	17.25

Join Kind

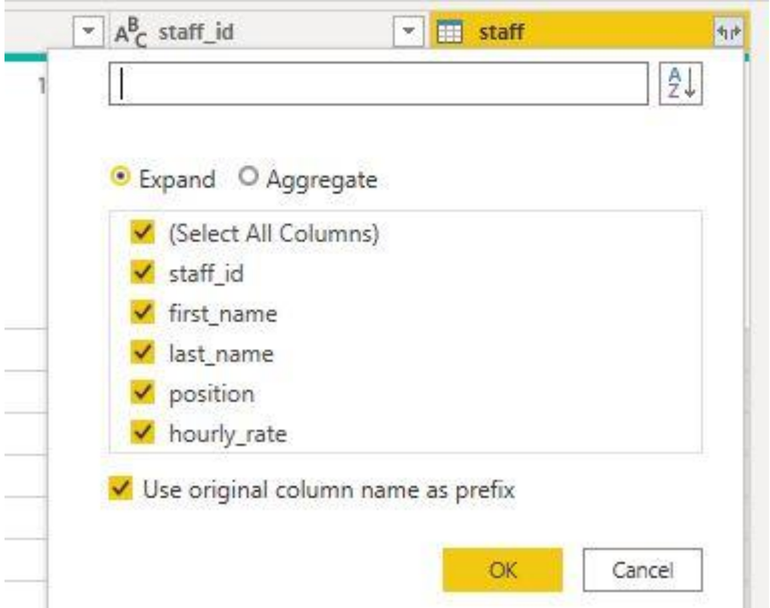
Left Outer (all from first, matching from second)

☐ Use fuzzy matching to perform the merge

▷ Fuzzy matching options

✓ The selection matches 40 of 40 rows from the first table.

Given the successful merge, I can now select fields from the staff table that are seen in staff-mgmt view



● Expand ○ Aggregate

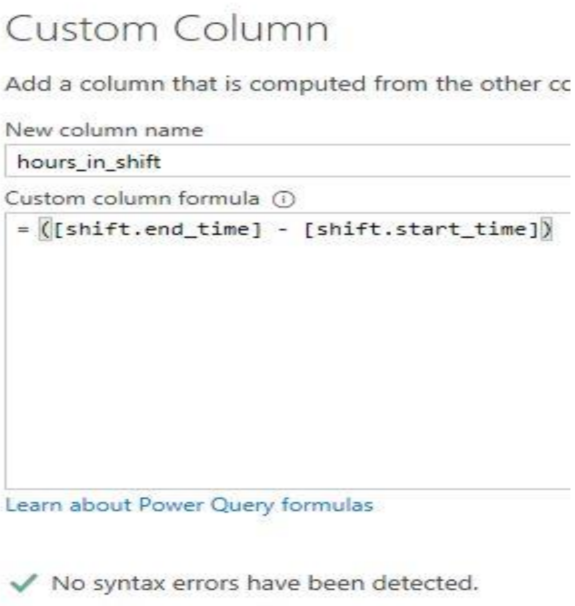
☑ (Select All Columns)
☑ staff_id
☑ first_name
☑ last_name
☑ position
☑ hourly_rate

☑ Use original column name as prefix

In the new table, I click on Merger Queries > Merge Queries (since I don’t need a new table to be created from staff_mgmt but merging it with shift table). Same process is repeated to bring in shift fields.

Finally, some transformation to create new columns for hours_in_shift and staff_cost.

Hours_in_shift data type was changed to duration and calculated duration in Total hours (changing 4hrs 30mins to 4.5hrs)



Custom Column

Add a column that is computed from the other cc

New column name

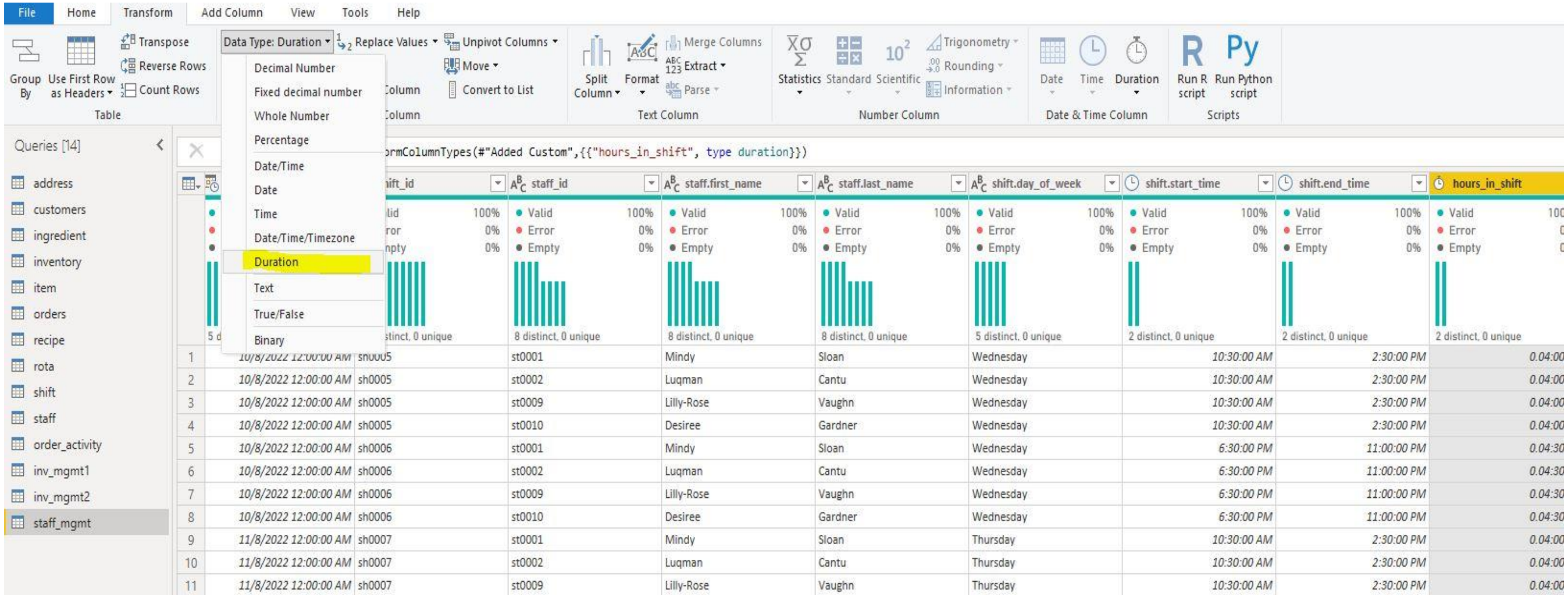
hours_in_shift

Custom column formula ⓘ

```
= ([shift.end_time] - [shift.start_time])
```

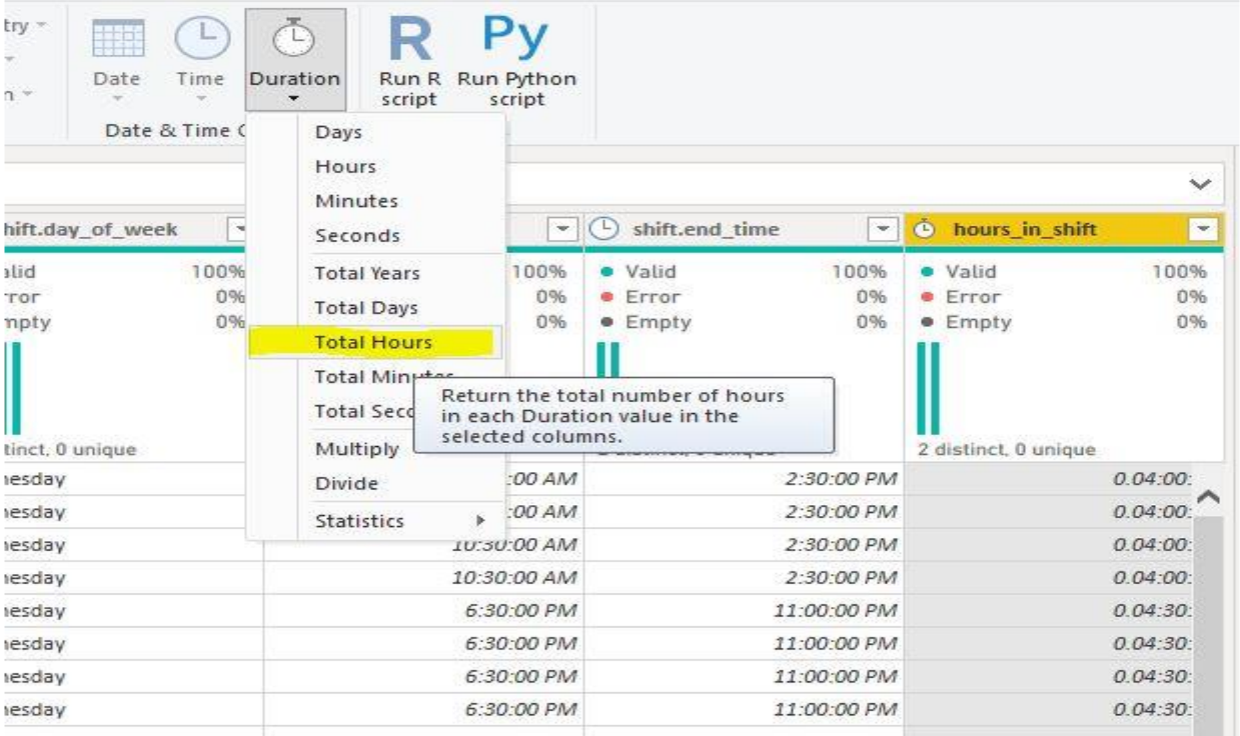
Learn about Power Query formulas

✓ No syntax errors have been detected.



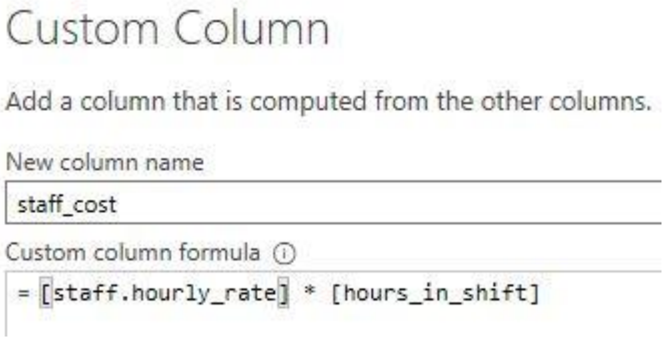
```
ormColumnTypes(#"Added Custom",{{"hours_in_shift", type duration}})
```

		ift_id	staff_id	staff.first_name	staff.last_name	shift.day_of_week	shift.start_time	shift.end_time	hours_in_shift
	5 d	tinct, 0 unique	8 distinct, 0 unique	8 distinct, 0 unique	8 distinct, 0 unique	5 distinct, 0 unique	2 distinct, 0 unique	2 distinct, 0 unique	2 distinct, 0 unique
1	10/8/2022 12:00:00 AM	sh0005	st0001	Mindy	Sloan	Wednesday	10:30:00 AM	2:30:00 PM	0.04:00
2	10/8/2022 12:00:00 AM	sh0005	st0002	Luqman	Cantu	Wednesday	10:30:00 AM	2:30:00 PM	0.04:00
3	10/8/2022 12:00:00 AM	sh0005	st0009	Lilly-Rose	Vaughn	Wednesday	10:30:00 AM	2:30:00 PM	0.04:00
4	10/8/2022 12:00:00 AM	sh0005	st0010	Desiree	Gardner	Wednesday	10:30:00 AM	2:30:00 PM	0.04:00
5	10/8/2022 12:00:00 AM	sh0006	st0001	Mindy	Sloan	Wednesday	6:30:00 PM	11:00:00 PM	0.04:30
6	10/8/2022 12:00:00 AM	sh0006	st0002	Luqman	Cantu	Wednesday	6:30:00 PM	11:00:00 PM	0.04:30
7	10/8/2022 12:00:00 AM	sh0006	st0009	Lilly-Rose	Vaughn	Wednesday	6:30:00 PM	11:00:00 PM	0.04:30
8	10/8/2022 12:00:00 AM	sh0006	st0010	Desiree	Gardner	Wednesday	6:30:00 PM	11:00:00 PM	0.04:30
9	11/8/2022 12:00:00 AM	sh0007	st0001	Mindy	Sloan	Thursday	10:30:00 AM	2:30:00 PM	0.04:00
10	11/8/2022 12:00:00 AM	sh0007	st0002	Luqman	Cantu	Thursday	10:30:00 AM	2:30:00 PM	0.04:00
11	11/8/2022 12:00:00 AM	sh0007	st0009	Lilly-Rose	Vaughn	Thursday	10:30:00 AM	2:30:00 PM	0.04:00



Return the total number of hours in each Duration value in the selected columns.

hift.day_of_week		shift.end_time	hours_in_shift
alid 100% rror 0% mpty 0%	100% 0% 0%	Valid 100% Error 0% Empty 0%	Valid 100% Error 0% Empty 0%
tinct, 0 unique			2 distinct, 0 unique
esday	:00 AM	2:30:00 PM	0.04:00:
esday	:00 AM	2:30:00 PM	0.04:00:
esday	10:30:00 AM	2:30:00 PM	0.04:00:
esday	10:30:00 AM	2:30:00 PM	0.04:00:
esday	6:30:00 PM	11:00:00 PM	0.04:30:
esday	6:30:00 PM	11:00:00 PM	0.04:30:
esday	6:30:00 PM	11:00:00 PM	0.04:30:
esday	6:30:00 PM	11:00:00 PM	0.04:30:



Custom Column

Add a column that is computed from the other columns.

New column name

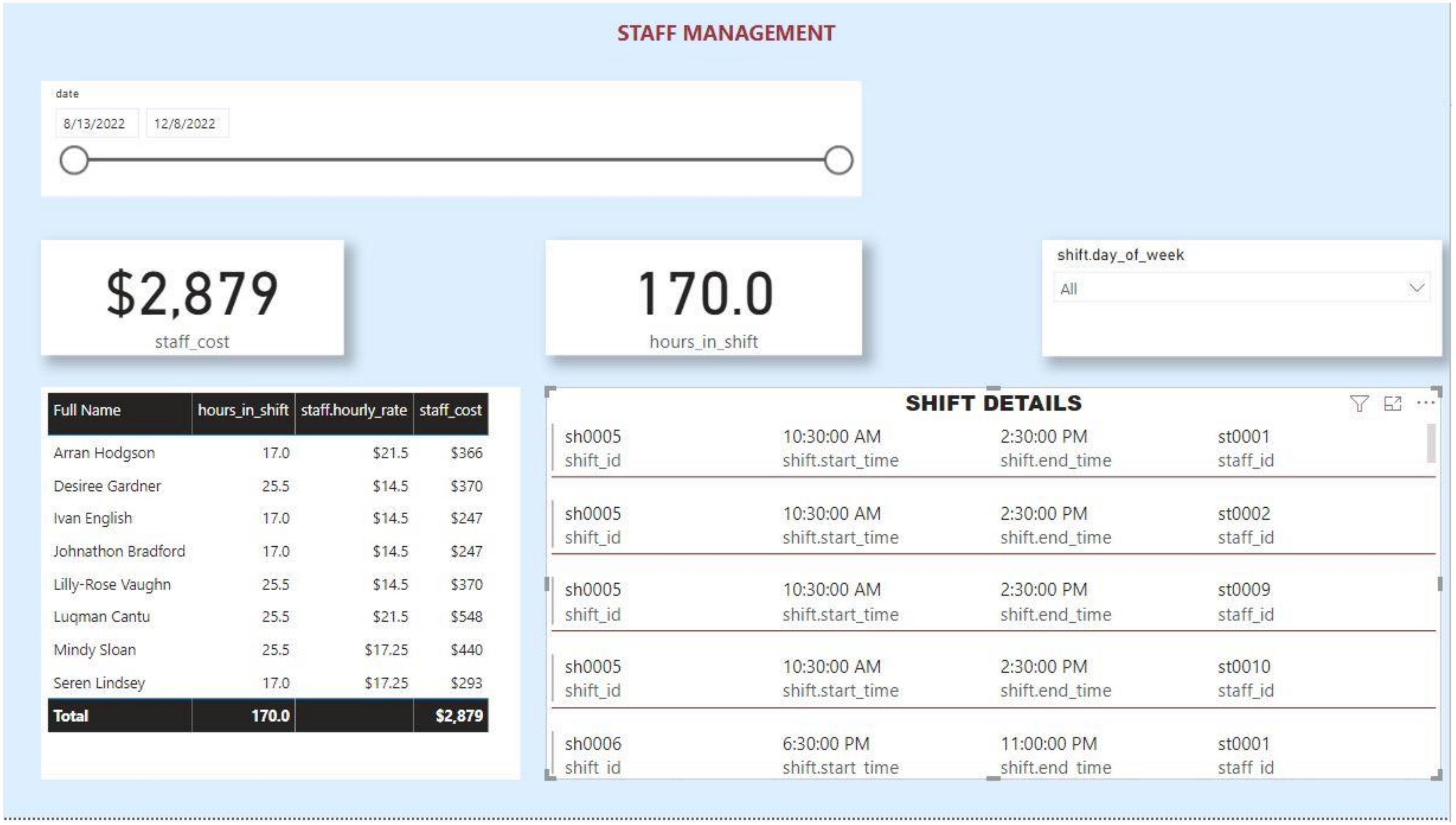
staff_cost

Custom column formula ⓘ

```
= [staff.hourly_rate] * [hours_in_shift]
```

11. **Staff Management Visual (Viz) Tiles** (*Kindly interact with pbix file or view pdf report*):

    Viz Tile 1 Slicer : Slider placed on the date field
    Viz Tile 2 Card : SUM staff_cost
    Viz Tile 3 Card : SUM hours_in_shift
    Viz Tile 4 Slicer: Day of Week Dropdown filter
    Viz Tile 5 Table: Staff Name BY hours_in_shift, staff.hourly_rate, staff_cost
    Viz Tile 6 Multi Row Card: Details on shift_id, start_time, end_time, staff_id



STAFF MANAGEMENT

date

8/13/2022 12/8/2022

$2,879
staff_cost

170.0
hours_in_shift

shift.day_of_week
All

Full Name	hours_in_shift	staff.hourly_rate	staff_cost
Arran Hodgson	17.0	$21.5	$366
Desiree Gardner	25.5	$14.5	$370
Ivan English	17.0	$14.5	$247
Johnathon Bradford	17.0	$14.5	$247
Lilly-Rose Vaughn	25.5	$14.5	$370
Luqman Cantu	25.5	$21.5	$548
Mindy Sloan	25.5	$17.25	$440
Seren Lindsey	17.0	$17.25	$293
Total	**170.0**		**$2,879**

SHIFT DETAILS

sh0005 shift_id | 10:30:00 AM shift.start_time | 2:30:00 PM shift.end_time | st0001 staff_id

sh0005 shift_id | 10:30:00 AM shift.start_time | 2:30:00 PM shift.end_time | st0002 staff_id

sh0005 shift_id | 10:30:00 AM shift.start_time | 2:30:00 PM shift.end_time | st0009 staff_id

sh0005 shift_id | 10:30:00 AM shift.start_time | 2:30:00 PM shift.end_time | st0010 staff_id

sh0006 shift_id | 6:30:00 PM shift.start_time | 11:00:00 PM shift.end_time | st0001 staff_id

12. **Order Activity Visual (Viz) Tiles** were created using following logic. (*Kindly interact with pbix file or view pdf report*):

Viz Tile 1 Card : Total Orders = Count (Distinct) order_id ‘Rename for Visual’ AS Total Orders

Viz Tile 2 Card : ‘New Measure’ Total Sales = `SUMX (order_activity, order_activity[item_price] * order_activity[quantity])`

Viz Tile 3 Card : Total Items = SUM quantity ‘Rename for Visual’ AS Total Items

Viz Tile 4 Card: ‘New Measure’ Average Order Value = (`SUMX (order_activity, order_activity[item_price] * order_activity[quantity])) / DISTINCTCOUNT (order_activity[order_id])`

Viz Tile 5 Donut Chart: ‘Measure’ Total Sales by Item Category

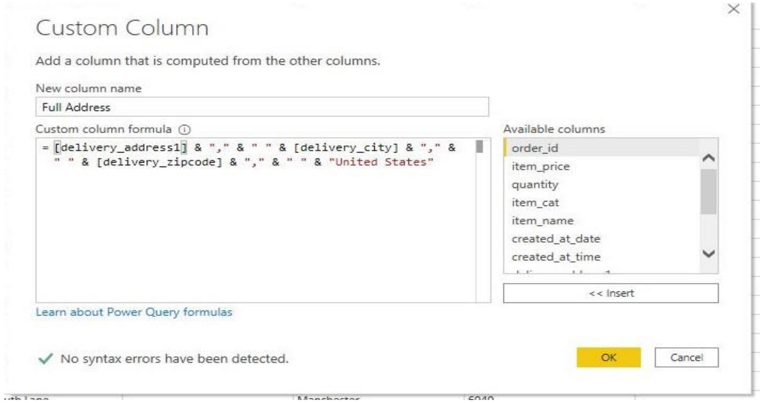
Viz Tile 6 Clustered Bar Chart: ‘Measure’ Total Sales by Item Name | Applied filter: ‘Top 5 By Value’ Total Sales

Viz Tile 7 Line Chart: ‘Measure’ Total Sales by [Count (Distinct) order_id] by created_at_time AS Total Sales and Orders by Hour

***Note created_at datetime has been transformed in Power BI by splitting to separate date and time columns*

Viz Tile 8 Map: ‘Measure’ Total Sales by Full Address

***Note Full address was created from CONCANCATE address fields and ‘United States’ text so map can recognize address data.*



Custom Column

Add a column that is computed from the other columns.

New column name

Full Address

Custom column formula

```
= [delivery_address1] & "," & " " & [delivery_city] & "," &
  " " & [delivery_zipcode] & "," & " " & "United States"
```

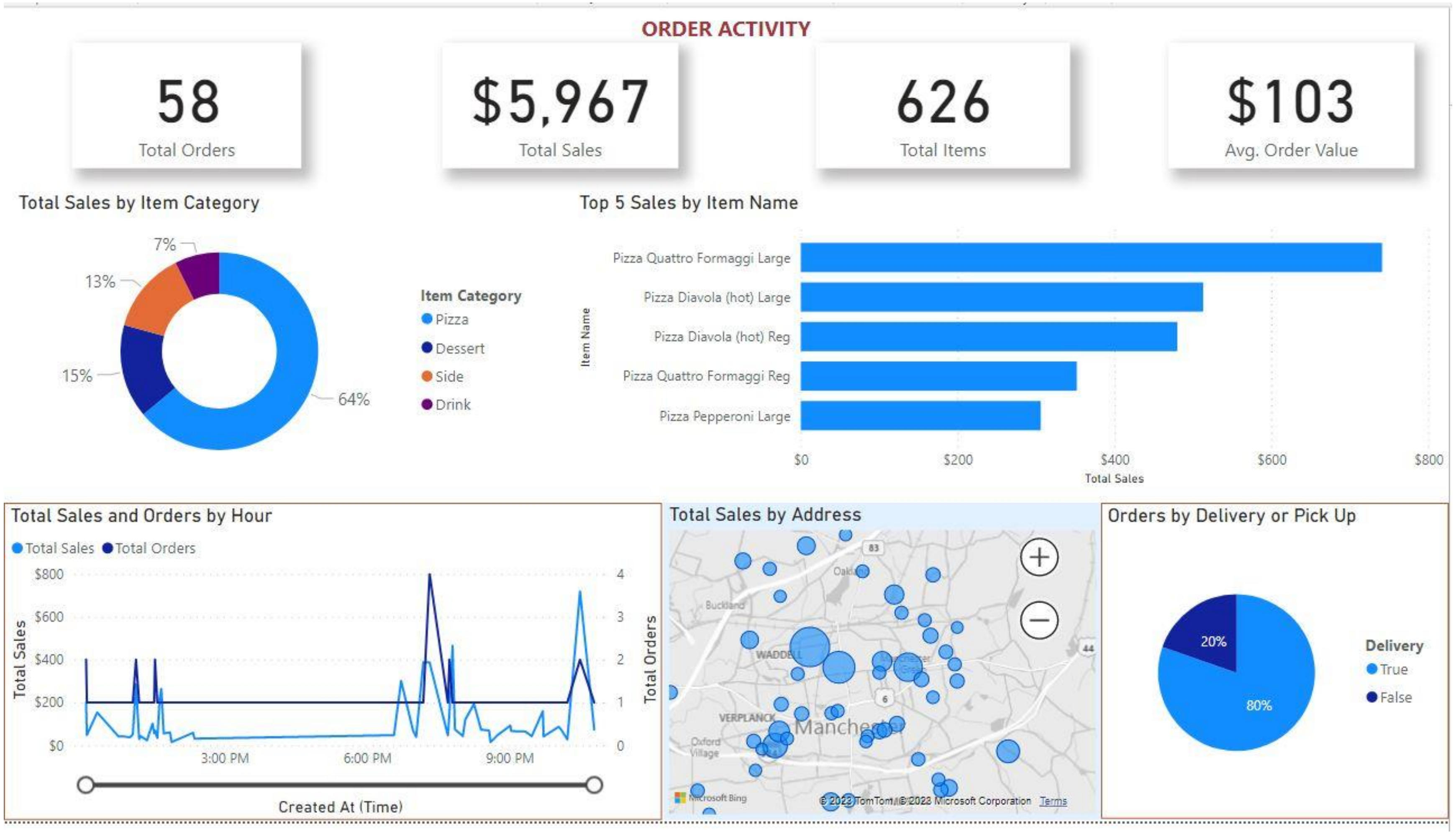
Available columns

order_id
item_price
quantity
item_cat
item_name
created_at_date
created_at_time

Learn about Power Query formulas

✓ No syntax errors have been detected.

Viz Tile 9 Pie Chart: Orders by delivery or Pick Up = Count (Distinct) order_id by delivery [True OR False]



13. **Inventory Management Visual (Viz) Tiles** were created with following Logic. (*Kindly interact with pbix file or view pdf report*):

Viz Tile 1 Card: Total Ingredient Cost = SUM ingredient_cost

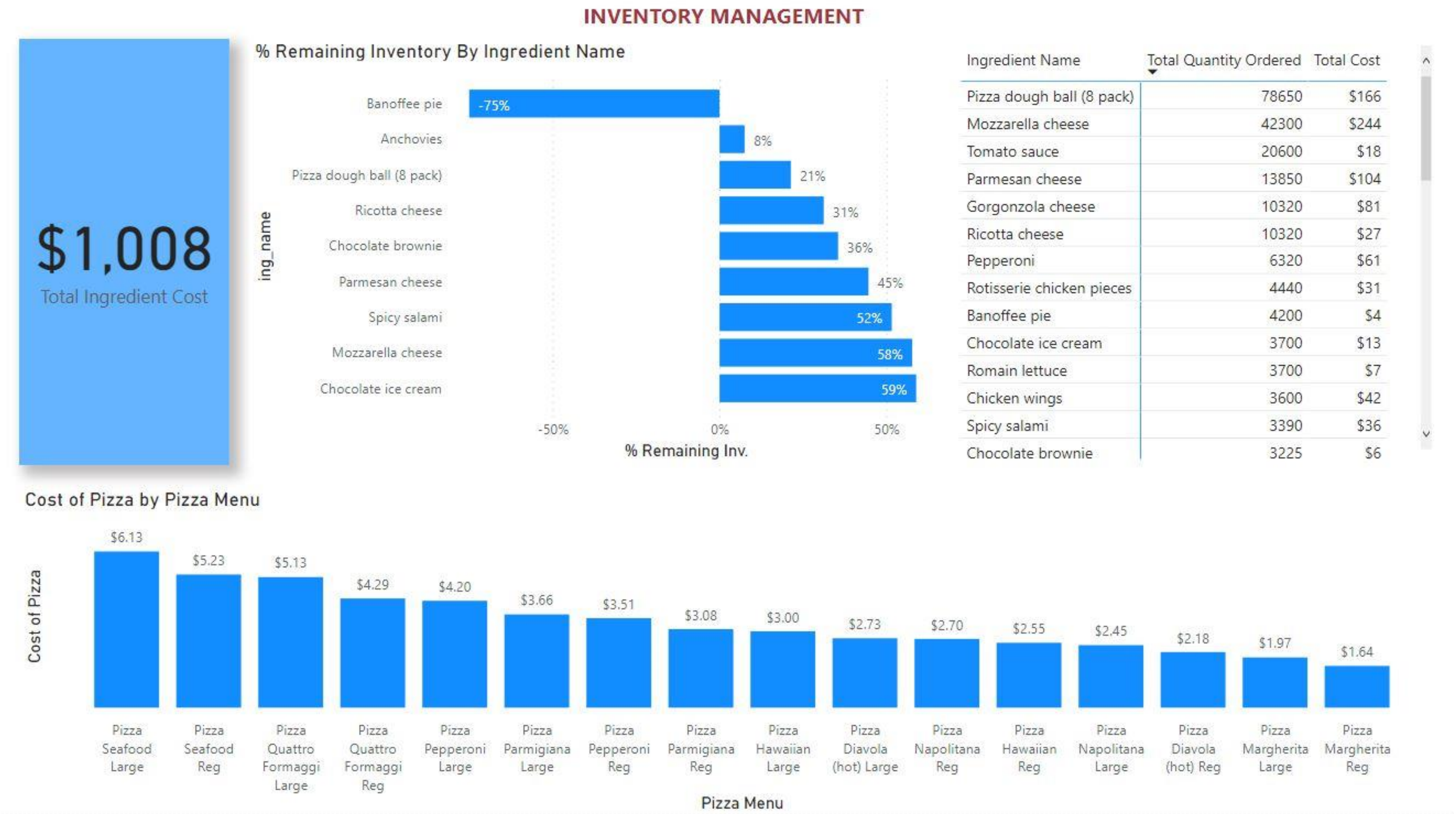
Viz Tile 2 Stacked Bar Chart: 'New Measure' % Remaining Inv = `SUMX (inv_mgmt2, (inv_mgmt2[total_inv_weight] - inv_mgmt2[ordered_weight]) / inv_mgmt2[total_inv_weight])` | Applied Filter: < 60%

BY ing_name AS % Remaining Inventory By Ingredient Name

Viz Tile 3 Matrix: Row-Ingredient Name; Values- SUM ingredient_cost, SUM ordered_weight

Viz Tile 4 Clustered Column Chart: ‘New Measure’ Cost of Pizza = `SUMX( inv_mgmt1, inv_mgmt1[recipe_quantity] * inv_mgmt1[unit_cost] )`

BY Menu | Applied Filter: Menu Contains Pizza AS Cost od Pizza by Pizza Menu



INVENTORY MANAGEMENT

$1,008

Total Ingredient Cost

Ingredient Name	Total Quantity Ordered	Total Cost
Pizza dough ball (8 pack)	78650	$166
Mozzarella cheese	42300	$244
Tomato sauce	20600	$18
Parmesan cheese	13850	$104
Gorgonzola cheese	10320	$81
Ricotta cheese	10320	$27
Pepperoni	6320	$61
Rotisserie chicken pieces	4440	$31
Banoffee pie	4200	$4
Chocolate ice cream	3700	$13
Romain lettuce	3700	$7
Chicken wings	3600	$42
Spicy salami	3390	$36
Chocolate brownie	3225	$6