Business Analysis

of

India Cash and Carry

By

Ritu Ranjani Ravi Shankar

India Cash and Carry Business Analysis

Ritu Ranjani Ravi Shankar

Overview

The purpose of this project is to analyze a business location and observe the activities performed by the customers and the shopkeepers in the store, document the business scenario and use the data obtained from this scenario for data generation and analysis.

India Cash and Carry Scenario/Story

India Cash and Carry is a small local chain that sells Indian groceries, Indian specialty foods and takeaway food, located at El Camino Real.

A customer enters the India Cash and Carry (ICC) store on a Saturday at 8:30 PM. He then picks up the grocery cart from the entrance of the store. The isles are organized according to each category - snacks, spices, grains, rice, frozen food items, vegetables and fruits. He then navigates through each isle and picks up the items that he wants and places it in his cart. The customer is looking for red moong dhal in the grains section and is not able to find it. A helper is arranging items in the isle next to him. The customer reaches out to the helper enquiring about the dhal he is looking for. The helper didn't know if it was in stock and goes to the checkout lane to ask the billing person. The billing person then checks for stock in her computer, and then guided the helper to go to the warehouse, located inside the store towards the end of the checkout lane, and pick up the dhal from there. The helper then handed the dhal to the customer. The customer continued to navigate through each isle, until he had picked everything he wanted, then proceeded to the checkout lane. The line is quite long in the card payment lane and customer was checking if he had enough cash to move to the cash lane. The cash lane was shorter. The customer moved to the cash lane, the billing person scans his items, a helper next to the billing person helps to load the items into the paper bag. The billing person then told the total amount to the customer, he then paid for the items, collected his receipt and went to the restaurant section of the store.

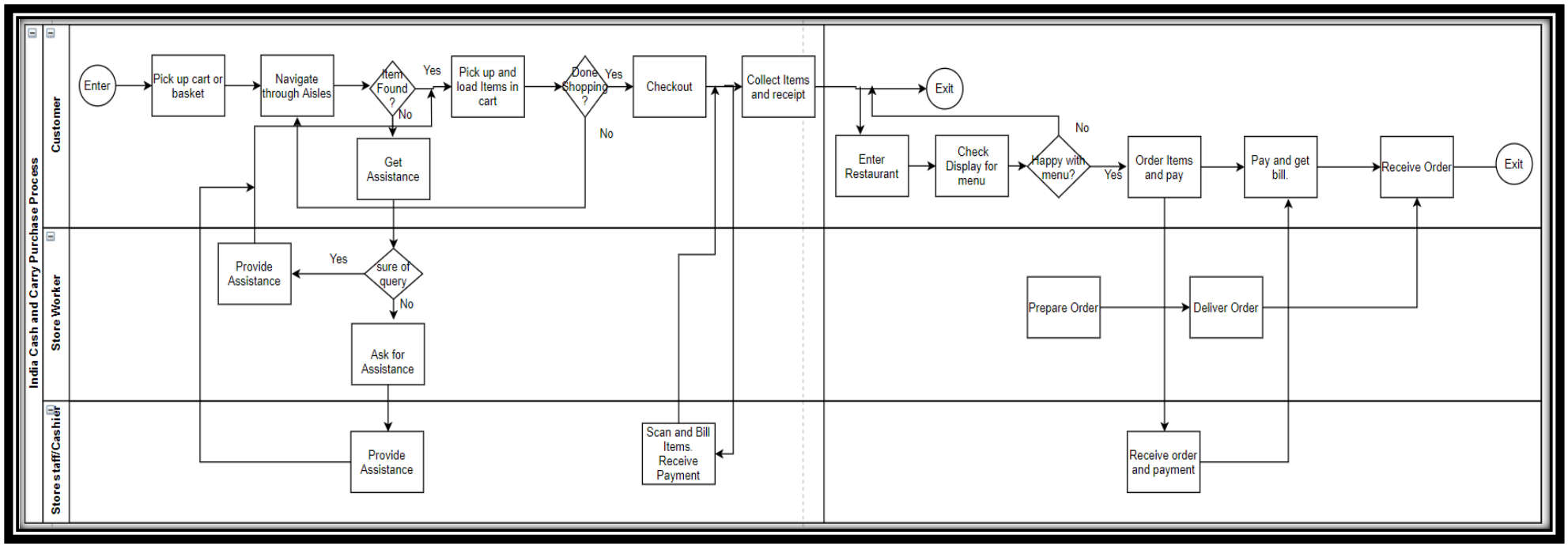
He looked at the menu at the TV display of the restaurant and called somebody to check what they wanted. He then stood in the queue to order his food. When it was his turn, he ordered the food, paid for the food items and waited on the side to collect his food. After 5 mins, the server called his name and he picked his food and left the store at 8:45 PM.

He unloaded all his items from the cart into his car, parked outside the store, then put the cart back in the cart return area and left.

Ritu Ranjani Ravi Shankar

Swim Lane Diagram

A pictorial representation of the job sharing, and responsibilities of the sub-processes described in the business scenario is shown by the swim lane diagram.



India Cash and Carry Business Analysis

Ritu Ranjani Ravi Shankar

Data Generation

Generated data using the receipt from the store. The receipt has information about the product purchased, the quantity of product purchased, the price of each product, the register the customer was served, Credit card authorization or cash, the discount on each item, store details.

Business Process & Data Assets

	Task	Entities
Customer Happy Path	Look for Items Check for Price Picks up item Picks one or two quantity of the item. Checks out	Item Price Item Description Item quantity Credit Card details Discount on Item Receive Receipt
Customer Exceptions	Doesn't Find the item	Store Inventory Log

From the above business process of happy and exception paths, we generate the data of the purchase made by the customer.

	Customer	Store
Data	Customer ID Credit Card Details/ Cash Payment	Transaction No Date/Time Register Served Transaction Authorization Item Discount Item Price Item Description Department ID/Description Order Quantity

Data Table Creation

Transforming the data assets to data tables in SQL, we create the tables first in a denormalized form, to understand the purchase process.

```
80    select * from denormalized_data;
```

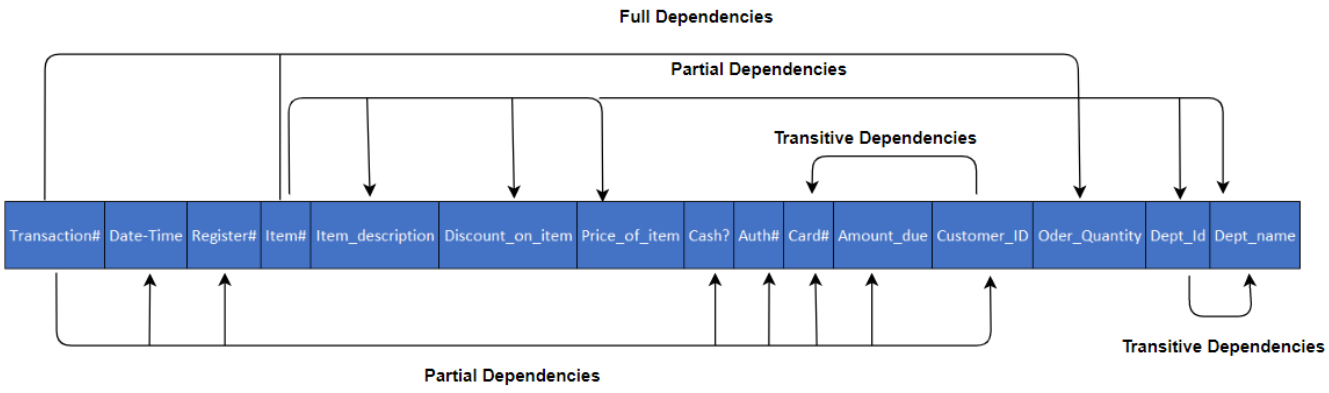
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Transaction_No	Date_Time	Register_no	Item_No	Item_Description	Discount_on_item	Price_of_item	Paid_by_cash	Auth_No	CreditCard_No
100	2020-01-04 00:00:00	6	4565	Limca	0.00	5.89	No	E5P 0O2	511655 830958
100	2020-01-04 00:00:00	6	4509	Kasoori Methi	0.99	2.99	No	E5P 0O2	511655 830958
100	2020-01-04 00:00:00	6	4232	Moong Dhal	0.90	1.99	No	E5P 0O2	511655 830958
100	2020-01-04 00:00:00	6	4132	Cumin Jeera	0.00	1.50	No	E5P 0O2	511655 830958
100	2020-01-04 00:00:00	6	4465	Britania Marie Gold	0.00	2.99	No	E5P 0O2	511655 830958
101	2020-01-05 00:00:00	1	4006	MTR Rava Dosa	0.99	2.49	Yes	O8F 5J8	NULL
101	2020-01-05 00:00:00	1	4465	Britania Marie Gold	0.00	2.99	Yes	O8F 5J8	NULL
102	2020-01-05 00:00:00	7	4194	Low Fat Yogurt	0.16	3.29	No	Q8I 8S8	538112 816888

CreditCard_No	Amount_Due	Customer_ID	Ordered_quantity	Dept_id	Dept_name
511655 830958 6615	0.00	2021	2	1	Beverages
511655 830958 6615	0.00	2021	1	3	Spices
511655 830958 6615	0.00	2021	1	5	Dal and Grains
511655 830958 6615	0.00	2021	1	3	Spices
511655 830958 6615	0.00	2021	1	2	Snacks
NULL	0.50	4347	1	8	Dry Batter
NULL	0.50	4347	2	2	Snacks
538112 8168881883	0.00	5159	1	6	Dairy

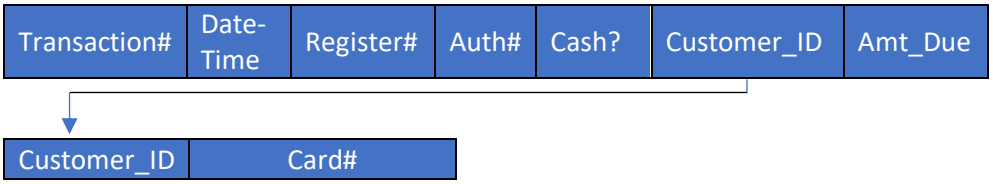
Here we see that; each transaction has multiple items that were purchased, and the data has duplicate values in few columns.

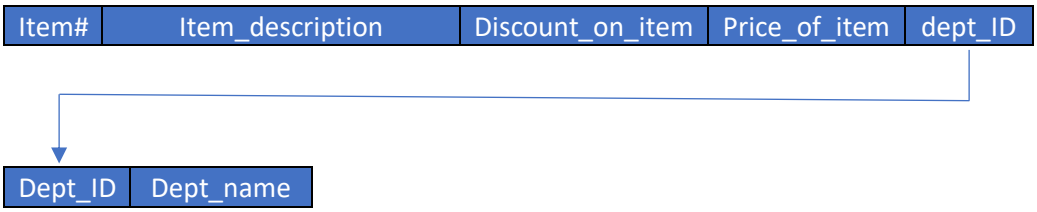
Identifying Dependencies among these columns



Converting the data table into 3rd Norm form:

By removing the transitive and partial dependencies, we could bring these tables into its 3rd normal form.





Transaction#	Item#	Order_Quantity

After converting to 3rd norm form, we now have 5 tables, whose relationship is defined by the primary and foreign key combination.

Table Creation Scripts:

```
CREATE TABLE Customer_Details
(Customer_ID   int NOT NULL,
CreditCard_No   CHAR(25) NULL,
CONSTRAINT PKCustomer_ID PRIMARY KEY (Customer_ID)   -- primary constraint
);
alter table customer_details drop index UniqSSN;

CREATE TABLE Order_details
(Transaction_No   INT NOT NULL,
Date_Time   datetime Not NULL,
Register_no   Int Not NULL,
Auth_No  CHAR(20) not null,
Paid_by_cash char(5) Not null,
Customer_ID int not null,
Amount_Due decimal(5,2),
CONSTRAINT FkCust_ID Foreign KEY (Customer_ID) References Customer_Details(Customer_ID),   -- Foreign Key constraint
CONSTRAINT pkTransaction_No PRIMARY KEY (Transaction_No),   -- primary constraint
CONSTRAINT UniqAuth_no UNIQUE (Auth_No)    -- unique constraint
);

CREATE TABLE Deptartment
(Dept_id   INT NOT NULL,
Dept_name CHAR(100) NOT NULL,
CONSTRAINT Pkdept_id Primary KEY (Dept_id)  -- primarykey constraint
);

CREATE TABLE Product_details
(Item_No   int NOT NULL,
Item_Description   CHAR(100) NOT NULL,
Discount_on_item   decimal(5,2) NULL,
Price_of_item Decimal(5,2) Not Null,
dept_Id int not null,
CONSTRAINT PkItem_No Primary KEY (Item_No) ,  -- primarykey constraint
CONSTRAINT Fkdept_ID Foreign KEY (dept_Id) References deptartment(dept_Id)  -- Foreign Key
);
Alter table Product_details modify Item_Description char(100);

CREATE TABLE Purchase_details
(Transaction_No   INT NOT NULL,
Item_No   int NOT NULL,
Ordered_quantity int not null,
CONSTRAINT FkTransac_No Foreign KEY (Transaction_No) References Order_details(Transaction_No),   -- Foreign Key constraint
CONSTRAINT Fk2Item_No Foreign KEY (Item_No) References Product_details(Item_No),  -- Foreign Key constraint
CONSTRAINT Pkpurchase Primary KEY (Transaction_No,Item_No)  -- primarykey constraint
);
```

Normalized Tables

We now have 5 tables:

1) order_details - contains the information such as the transaction ID – primary key, date and time of purchase, the register in which the customer was served, authorization code for the transaction, if the customer paid by cash or card (Boolean), customer id, and the amount due for the customer/transaction.

```
select * from order_details;
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History

Download

	transaction_no	date_time	register_no	auth_no	paid_by_cash	customer_id	amount_due
1	100	01/04/20 12:00:00 …	6	E5P 0O2	No	2021	0
2	101	01/04/20 12:00:00 …	1	O8F 5J8	Yes	4347	0.5
3	102	01/05/20 12:00:00 …	7	Q8I 8S8	No	5159	0
4	103	01/05/20 12:00:00 …	1	Z9G 7M1	No	9225	0
5	104	01/10/20 12:00:00 …	3	S6R 4T0	No	5217	0
6	105	01/11/20 12:00:00 …	4	M6O 3R8	Yes	9442	0.99
7	106	01/08/20 12:00:00	8	Z1N 5G8	No	8750	0

2) customer_details - contains the credit card details of the customer and the customer id (primary key).

```
select * from customer_details;
```

Query Result | Script Output | DBMS Output | Explain Pl

Download

	customer_id	creditcard_no
1	2021	511655 830958 6615
2	4347	NULL
3	5159	538112 8168881883
4	9225	546316 739497 2497
5	5217	511473 658099 4581
6	9442	NULL
7	8750	526123 758446 7010

3) product_details: contains the details of the item sold such as the item description, price, discount on the item, and which department the item belongs to. Here the item_no is the primary key and the dept_id is the foreign key.

```
12 select * from product_details;
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History

Download

	item_no	item_description	discount_on_item	price_of_item	dept_id
1	4565	Limca	0	5.89	1
2	4509	Kasoori Methi	0.99	2.99	3
3	4232	Moong Dhal	0.9	1.99	5
4	4132	Cumin Jeera	0	1.5	3
5	4465	Britania Marie Gold	0	2.99	2
6	4006	MTR Rava Dosa	0.99	2.49	8
7	4194	Low Fat Yogurt	0.16	3.29	6

4) department table: - contains the department name and the department id (primary key).

```
12 select * from department;
```

Query Result | Script Output | DBMS Output | Explain

Download

	dept_id	dept_name
1	1	Beverages
2	2	Snacks
3	3	Spices
4	4	Frozen
5	5	Dal and Grains
6	6	Dairy
7	7	Coffee

5) purchase_details: - contains the composite key(transaction_id and item_id) and ordered quantity.

```
12 select * from purchase_details;
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SC

Download

	transaction_no	item_no	ordered_quantity
1	100	4565	2
2	100	4509	1
3	100	4232	1

India Cash and Carry Business Analysis

Ritu Ranjani Ravi Shankar

Data Analysis

Using these data, we could answer the following questions

1) Which day of the week has the maximum sales?
2) What are the total sales of the grocery store by department?
3) What are the total sales of the restaurant?
4) Are the sales higher in the week of any Indian festival?
5) What percentage of the transaction was through cash?
6) Who is the most valuable customer?
7) Which register serves faster?
8) Is frozen food purchased more or fresh food from the restaurant purchased more?
9) Do the customers who buy from restaurant also buy frozen food?
10) Find the Avg sales per customer?
11) Find the top 10 most sold items?

Data Preprocessing:

Preprocessing data by checking the size of the dataset and seeing if there are any duplicates in these datasets.

Counting the number of *rows* in each table to know the size of the table.

```
148
149 select count(*) from order_Details;
150
```

Query Result | Script Output | DBMS Output | E

Download

	count(*)
1	19

```
152 select count(*) from product_Details;
```

Query Result | Script Output | DBMS Output | Ex

Download

	count(*)
1	21

```
154 select count(*) from purchase_Details;
```

Query Result | Script Output | DBMS Output | Ex

Download

	count(*)
1	60

```
159 select count(*) from customer_Details;
160
```

Query Result | Script Output | DBMS Output | Expl

Download

	count(*)
1	11

```
161 select count(*) from department;
162
```

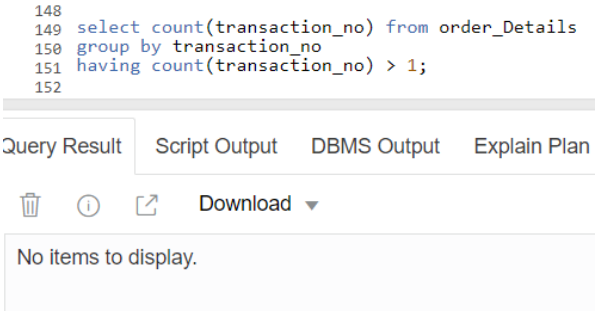
Query Result | Script Output | DBMS Output

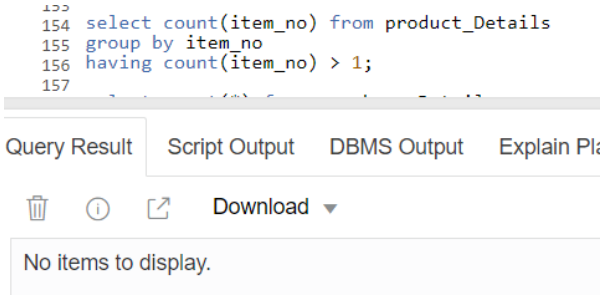
Download

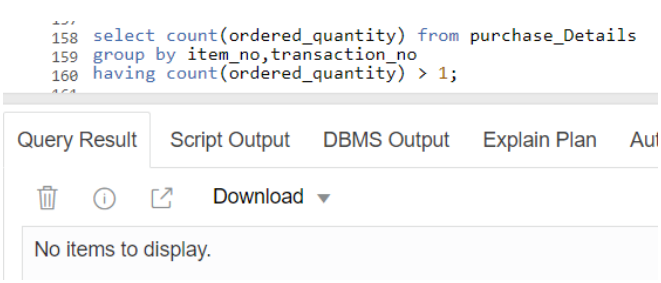
	count(*)
1	11

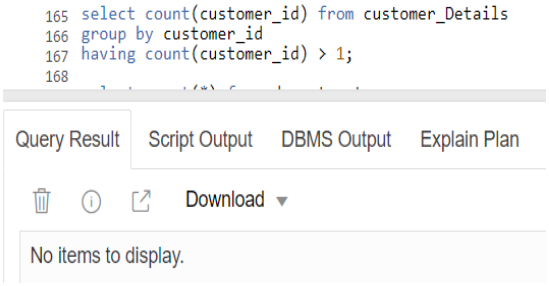
Checking for duplicate values in the table:

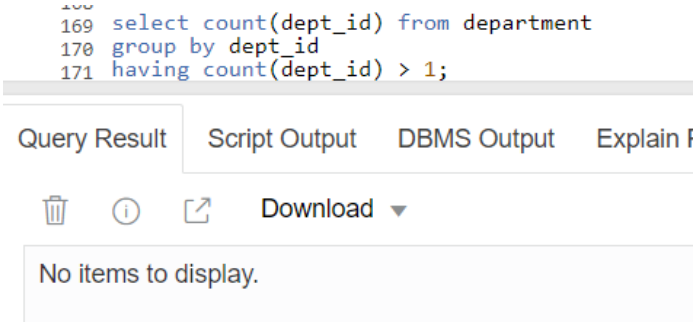
By using the group by function along with the count() function , we count the number of rows based on the primary key. If the count() is greater than one, that means there are more than 1 row with the same value. This indicates there are duplicates in the data. Below, we see that, our data has *no duplicates*.











After confirming that our table is clean and has no duplicates, we now proceed to our analyses.

Day of the week that has the maximum sales

```
105    -- Which day of the week has the maximum sales
106    select z.day_of_week,z.sales
107    from
108    (select y.day_of_week,y.sales,
109    rank() over(order by y.sales desc) as r
110    from
111    (select dayname(od.date_time) as day_of_week,
112     (sum(p.Price_of_item * purch.Ordered_quantity) - sum(p.Discount_on_item)) AS sales
113    from product_details p
114    join purchase_details purch
115    on p.Item_no = purch.Item_no
116    join order_details od
117    on od.transaction_no = purch.transaction_no
118    group by dayname(od.date_time))y)z
119    where r = 1;
120
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

day_of_week	sales
Saturday	195.07

Here, first we are calculating the total sales and grouping the data based on the date, to find the sales by each date. Then we are converting the date to day, i.e 01/11/2020 is converted to Saturday using the dayname() function. Similarly, all other dates are converted. Based on the sales highest to lowest, we rank these days as 1,2,3… etc. Fetching the highest sales day as the one with rank =1.

Result: From the query result above, we see that more customers visit the stores on a *Saturday.*

Interpretation: This could be attributed to the fact that, most customers who purchase items at this store could be working professionals, hence they find time only on weekends to do grocery shopping.

Sales of the grocery store based on dept

Given that India Cash and Carry also has a restaurant (take away food café), let's find out the total sales of just the grocery store, excluding the restaurant sales.

We must join product, purchase and department tables to calculate the sales and group by department to obtain the sales(revenue) of each department.

```
123    -- Total sales of Grocery store by dept
124    select d.dept_name as "Department",(sum(p.Price_of_item * purch.Ordered_quantity) - sum(p.Discount_on_item)) AS "Total Sales"
125    from product_details p
126    join purchase_details purch
127    on p.Item_no = purch.Item_no
128    join department d
129    on p.dept_ID = d.dept_ID
130    and dept_name not in ('restaurant')
131    group by 1
132    order by 2 desc;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Department	Total Sales
Frozen	39.54
Snacks	34.18
Batters	21.43
Dal and Grains	17.37
Beverages	11.78
Dairy	10.44
Coffee	8.98
Spices	8.00

Result Grid | Form Editor

Result: Sales of *Frozen* food department is more among all the other departments.

Interpretation: This gives us the lifestyle/the demography of the customers. The customers who purchase these frozen products could be millennials who are in their busiest lives. Hence preferring simple and quick food products.

Total sales from Restaurant

In the above analysis we saw that, sales have been maximum in the frozen department. Now let's try to find answers for these questions: Are people health conscious? Do they only rely on frozen items? Is there a segment of customers who eat healthy food?

```
22
23 -- Total sales of restaurant
24 select d.dept_name as "Restaurant",(sum(p.Price_of_item * purch.Ordered_quantity) - sum(p.Discount_on_item)) AS "Total Sales"
25 from product_details p
26 join purchase_details purch
27 on p.Item_no = purch.Item_no
28 join department d
29 on p.dept_ID = d.dept_ID
30 and d.dept_name = 'Restaurant'
31 group by d.dept_name;
32
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History

Download

	restaurant	total sales
1	Restaurant	262.02

By eyeballing the two results we see that, customers do buy fresh food from the restaurant and don't just depend on frozen food.

This now allows us to answer the question:

Is frozen food purchased more or fresh food from the restaurant purchased more.

In terms of total sales of frozen food from grocery and fresh food from restaurant, we are comparing which among them is sold more.

```
-- Is frozen food purchased more or fresh food from restaurant purchased more
delimiter $$
create procedure frozenvsfresh()
Begin
declare frozen,fresh decimal(5,2);
set @frozen := (select (sum(p.Price_of_item * purch.Ordered_quantity) - sum(p.Discount_on_item)) AS "Total sales of Frozen"
from product_details p
join purchase_details purch
on p.Item_no = purch.Item_no
join department d
on p.dept_ID = d.dept_ID and
dept_name like '%Frozen%');
set @fresh := (select (sum(p.Price_of_item * purch.Ordered_quantity) - sum(p.Discount_on_item)) AS "Total sales of Fresh"
from product_details p
join purchase_details purch
on p.Item_no = purch.Item_no
join department d
on p.dept_ID = d.dept_ID and
dept_name like '%restaurant%');
if frozen > fresh then
select concat('Frozen sales is more: ',@frozen) as Frozen, concat('Fresh food sales : ',@fresh) as Fresh;
elseif frozen = fresh then
select concat('Both the sales are equal. Fresh food sales: ',@fresh) as Fresh, concat('Frozen food sales : ',@frozen) as Frozen;
else
select concat('Fresh food sales is more: ',@fresh) as Fresh, concat('Frozen food sales : ',@frozen) as Frozen;
end if;
end $$
```

```
32      -- Call the stored procedure
33      call frozenvsfresh();
34
```

Result Grid | Filter Rows: | Export: | Wrap Cell Cont

Fresh	Frozen
Fresh food sales is more: 262.02	Frozen food sales : 39.54

We construct a stored procedure to calculate the sales(revenue) from frozen food and revenue from fresh food. We then use the if else construct to check which sales has been more.

Result: We see from the above query result that the sales of the fresh food from restaurant is almost 3X greater than that of the frozen food.

Interpretation: Customers prefer fresh food more compared to frozen food. However, they also buy more frozen food compared to other grocery items in the store.

We could interpret that people try to balance their lifestyle based on time and try to eat fresh.

However, we don't know if the customers who bought the fresh food are the ones who also bought frozen food.

Do the customers who buy from restaurant also buy frozen food

```
-- person who buys frozen also buying from restaurant
select rest.customer_id,rest.dept_name as Department1,froz.dept_name as Department2 from
(select p.transaction_no,p.item_no,d.dept_name,od.customer_id from purchase_details p join product_details pr
on p.item_no = pr.item_no join department d on d.dept_id = pr.dept_id
join order_details od on od.transaction_no = p.transaction_no
where d.dept_name = 'restaurant') as rest
join
(select p.transaction_no,p.item_no,d.dept_name,od.customer_id from purchase_details p join product_details pr
on p.item_no = pr.item_no join department d on d.dept_id = pr.dept_id
join order_details od on od.transaction_no = p.transaction_no
where d.dept_name = 'frozen') as froz
on rest.customer_id = froz.customer_id
group by 1,2,3;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	Department1	Department2
9225	Restaurant	Frozen

First, we find the customers who have purchased frozen food, and then find customers who purchased from restaurants. We then join the both these subqueries and make a temporary table/derived table that only returns the customers who are in both these groups.

Result: Customer 9225, is the only customer who has bought both frozen food and fresh food from restaurant.

Interpretation: We could interpret that people either prefer fresh food or frozen food, not both in most cases.

Are the sales higher in the week of any Indian festival

Finding the date that has the highest sale.

Here, first we calculate the total sales and grouping the data based on the date, to find the sales by each date. Based on the sales highest to lowest, we rank these days as 1,2,3… etc using the rank() function. Fetching the highest sales date as the one with rank =1.

```sql
-- The date that has most sales , is the following week an Indian festival?
select z.date_time,z.sales
from
(select y.date_time,y.sales,
rank() over(order by y.sales desc) as r
from
(select od.date_time,
 (sum(p.Price_of_item * purch.Ordered_quantity) - sum(p.Discount_on_item)) AS sales
from product_details p
join purchase_details purch
on p.Item_no = purch.Item_no
join order_details od
on od.transaction_no = purch.transaction_no
group by od.date_time)y)z
where r = 1;
```

ery Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History

Download

	date_time	sales
	01/11/20 12:00:00 …	142.52

Result: We see from the above query result that the sales has been maximum on 11th Jan 2020.

Interpretation: On googling for Indian festivals, the first festival in the year 2020 was Makar Sankranti/Lohri/Pongal, on 14th and 15th January 2020. We could interpret that, the sales are high in the Indian grocery store, if the following week has an Indian Festival.

Given that the sales are high during the weekends and during the week before any Indian festival, let's find **which register served faster on those days.**

```sql
-- Which register served faster

select Register_no, date_time, count(Register_no) as "Served Customers"
from order_details
group by date_time,Register_no
having count(Register_no) > 1
order by 3 desc;
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History

Download

	register_no	date_time	served customers
1	4	01/11/20 12:00:00 …	3
2	2	01/13/20 12:00:00 …	2
3	1	01/05/20 12:00:00 …	2
4	2	01/11/20 12:00:00 …	2

We group the order table based on date and register number and count how many times each register appeared. This tells us which register served more. We omit the counters that only served 1 customer.

Result: We see from the above query result that Register 4 has served more customers than other registers on the 11th Jan 2020, when the sales was high.

Interpretation: These registers could have had staff workers who were quick to scan items and give the receipt to the customers, than the other registers. Since these registers were quicker, more people might have preferred getting their items billed in these counters.

The most valuable customer

The customer who purchases more or has contributed more towards the store's revenue is the most valuable customer.

```
-- The customer who has contributed more towards the revenue of store.
delimiter $$
create procedure ValuableCustomer()
Begin
declare Customer int;
declare total decimal(5,2);
set @Customer = (select z.customer_id
from (select y.customer_id, y.Total_Sales,
 rank() over(order by y.Total_Sales desc) as r
 from (select od.customer_id, (sum(p.Price_of_item * purch.Ordered_quantity) - sum(p.Discount_on_item)) AS Total_Sales
from product_details p
join purchase_details purch
on p.Item_no = purch.Item_no
join order_details od
on od.transaction_no = purch.transaction_no
group by od.customer_id
order by 2 desc)y)z where r = 1);
set @total = (select z.Total_Sales
from (select y.customer_id, y.Total_Sales,
 rank() over(order by y.Total_Sales desc) as r
 from (select od.customer_id, (sum(p.Price_of_item * purch.Ordered_quantity) - sum(p.Discount_on_item)) AS Total_Sales
from product_details p
join purchase_details purch
on p.Item_no = purch.Item_no
join order_details od
on od.transaction_no = purch.transaction_no
group by od.customer_id
order by 2 desc)y)z where r=1);
select concat('The most valuable customer is : ' ,@Customer) as "Valuable Customer",
 concat('with sales: ', @total) as 'total sales';
End $$
```

```
96 •    call ValuableCustomer();
```

Result Grid | Filter Rows: | Export:

Valuable Customer	total sales
The most valuable customer is : 5159	with sales: 99.82

We construct a stored procedure to identify the customer who has contributed more towards sales by ranking each customer based on their total purchase.

Result: Customer 5159, is the most valuable customer with respect to sales.

Interpretation: We can analyze based on the sales and customer data, who has purchased more and how frequently do they visit the store. Here, customer 5159 has the most sales from our dataset.

India Cash and Carry Business Analysis

Ritu Ranjani Ravi Shankar

This analysis would help us to target customers who purchase less and lure them to buy more by giving offers and coupons.

Percentage of sales through cash

```
-- percentage of sales through cash

delimiter $$
create procedure salesthroughcash()
Begin
declare cash,total,percentage decimal(5,2);
set @cash := (select (sum(p.Price_of_item * purch.Ordered_quantity) - sum(p.Discount_on_item)) AS "sales through cash"
from product_details p
join purchase_details purch
on p.Item_no = purch.Item_no
join order_details od
on od.transaction_no = purch.transaction_no
and od.paid_by_cash = "yes");
set @total := (select (sum(p.Price_of_item * purch.Ordered_quantity) - sum(p.Discount_on_item)) AS "Total Sales"
from product_details p
join purchase_details purch
on p.Item_no = purch.Item_no);
set @percentage := round(((@cash/@total)*100));
select concat('% sales through cash is : ', @percentage) as "Cash Sales";
End $$
```

```
58      call salesthroughcash();
```

Result Grid | Filter Rows: | Export:

Cash Sales
% sales through cash is : 16

We construct a stored procedure to calculate the % of sales by cash. We first extract those transactions that are paid by cash and calculate their sales. Next, we calculate the total sales of the dataset, irrespective of being paid by cash or card. To calculate the %, we divide the sales through cash by total sales.

Result: Only 16% of the total sales was through cash.

Interpretation: This means that customers prefer purchasing through credit card and based on our business scenario, the customers who purchased through cash could be the ones who didn't want to stand in a longer queue and wanted to move to the cash queue to bill quickly and exit the shop. Also, these customers could have bought just one or two items, that costed less and preferred paying through cash.

Top 10 most sold/purchased Items

We calculate top 10 most purchased items, by grouping based on the item and counting the number of times each item was purchased by the customers. We create this as a view, as this information could be needed on a day to day basis.

```
190 -- View to find the top 10 products purchased
191 create view max_product_sold as
192 select * from
193 (select p.item_description, sum((purch.Ordered_quantity)) AS "Number_sold"
194 from product_details p
195 join purchase_details purch
196 on p.item_no = purch.item_no
197 group by item_description
198 order by 2 desc ) where rownum <=10;
199
200 select * from max_product_sold;
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History

Download

	item_description	number_sold
1	Fresh Roti	41
2	Britania Marie Gold	10
3	Toor Dal	6
4	Moong Dhal	6
5	Haldirams Home S…	6
6	Shastha Dosa Batter	6

Result: Fresh Roti is purchased more, followed by Biscuits and lentils.

Interpretation: We saw earlier that customers preferred fresh food and hence, the top 10 products list validates our earlier analysis. Also, this information would help the store manager to manage and stock items that are in most demand.

Trigger to insert into the purchase table.

When an order is placed by an existing customer, the purchase table should also be inserted with the corresponding order, item and quantity.

```
1  -- insert trigger
2
3  create trigger purchase_order
4  after insert on order_details
5  for each row
6  Begin
7  Insert into purchase_details values(:new.Transaction_No,4851,3);
8   end;
9
10 insert into order_details values(114,Current_date,4,'Lyg e5r','No',6330,0.00);
11
12 select * from order_details where transaction_no = 114;
13
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History

Download

	transaction_no	date_time	register_no	auth_no	paid_by_cash	customer_id	amount_due
1	114	02/17/20 04:24:15 …	4	Lyg e5r	No	6330	0

```
14  select * from purchase_details where Transaction_No = 114;
15
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL His

Download

	transaction_no	item_no	ordered_quantity
1	114	4851	3

Triggers maintain data integrity and hence if there is any error in the insertion into order table, the record won't be inserted into the purchase table.

Average Sales per Customer

Calculating the average sales per customer,

```
365     -- avg sales per customer per month
366     select MONTHNAME(od.date_time) as Month,(sum(p.Price_of_item * purch.Ordered_quantity) - sum(p.Discount_on_item)) AS "Total sales",
367     ((sum(p.Price_of_item * purch.Ordered_quantity) - sum(p.Discount_on_item))/ count(distinct od.customer_id))
368     as "Avg per customer"
369     from product_details p
370     join purchase_details purch
371     on p.Item_no = purch.Item_no
372     join order_details od
373     on od.transaction_no = purch.transaction_no group by month(od.date_time);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Month	Total sales	Avg per customer
January	477.51	43.410000
February	15.72	15.720000

We calculate, grouping by every month the average sales of each customer, by dividing the total sales by number of customers.

Result: The total sales in the month of January has been $477.51, with an average of $43.41 per customer and in the month of February, the sales has been low with a total sale of $15.72, with an average of $15.72.

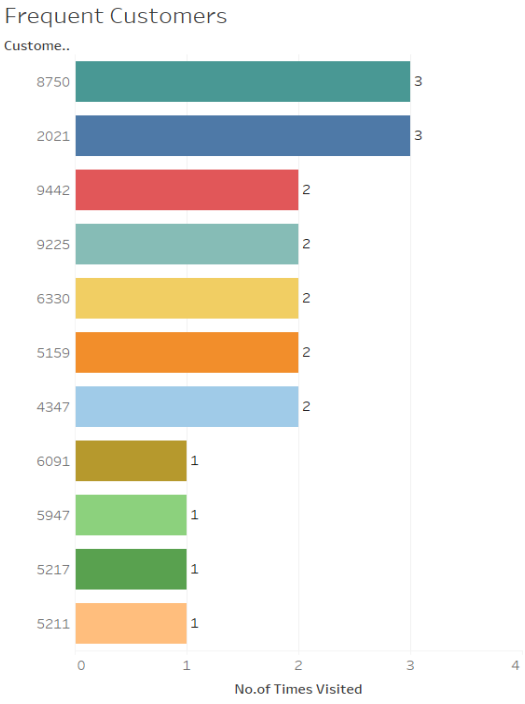
Interpretation: The revenue per customer has decreased from January to the month of February.

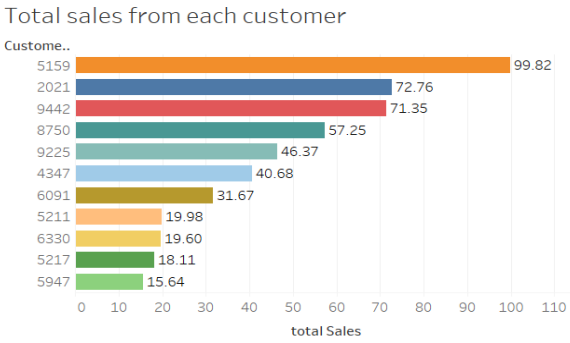
India Cash and Carry Business Analysis

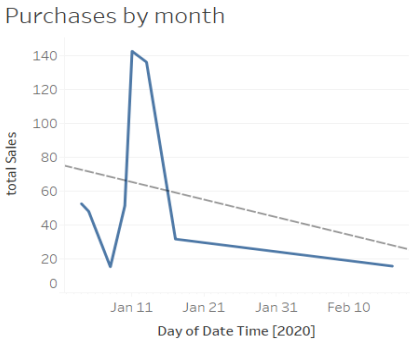
Ritu Ranjani Ravi Shankar

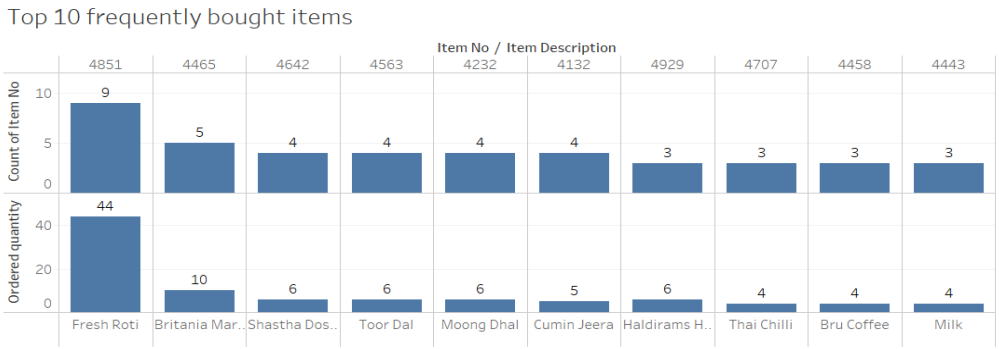
Data Visualization

Visualizing how frequently the customers have visited the store, total sales for each customer and top 10 frequently bought items and purchases by month using Tableau.

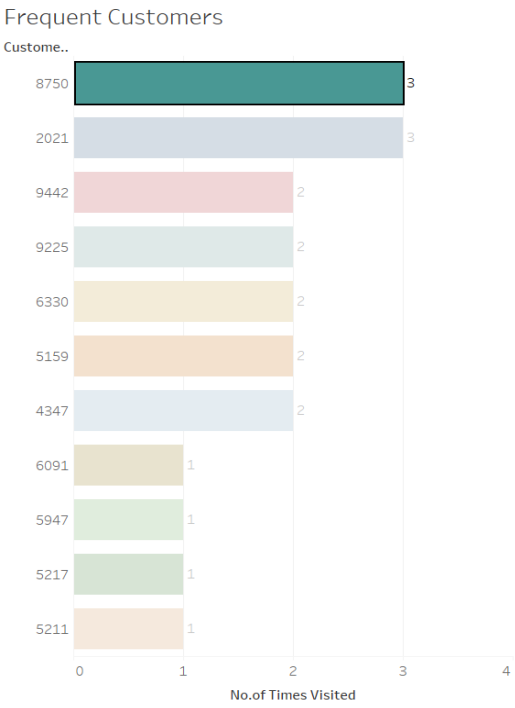


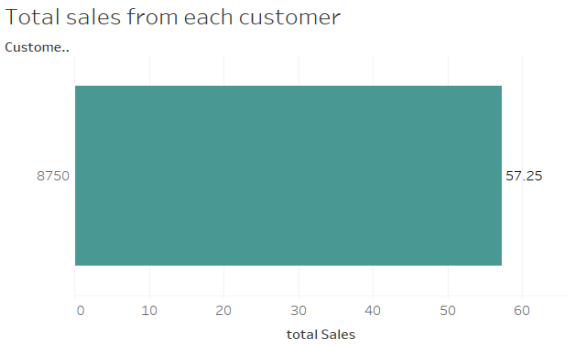


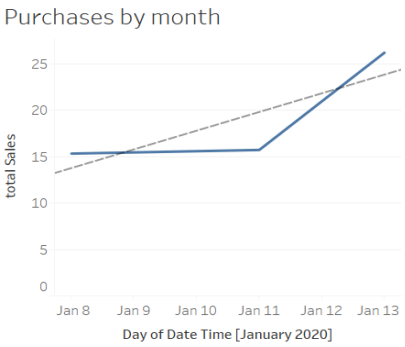


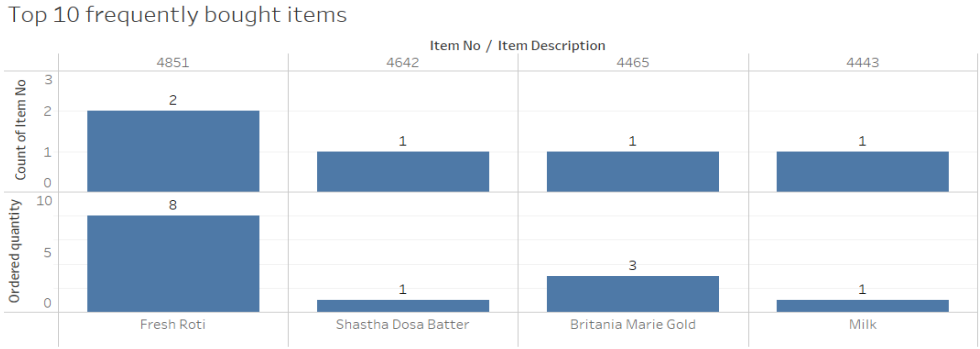


Customer 8750:









India Cash and Carry Business Analysis

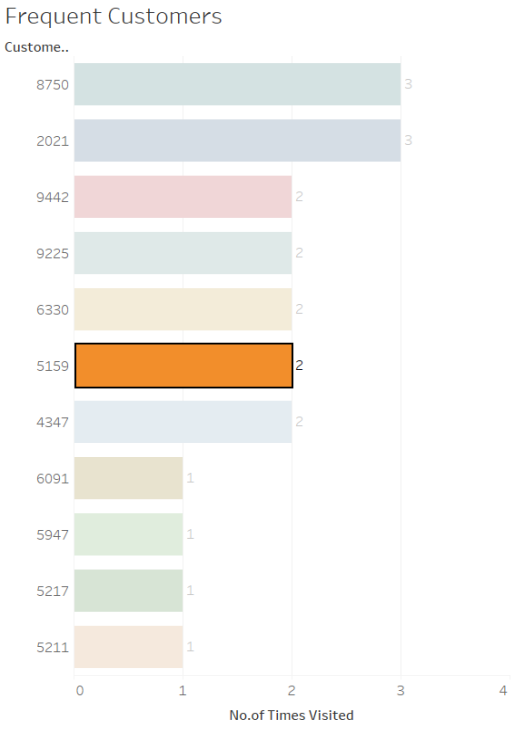
Ritu Ranjani Ravi Shankar

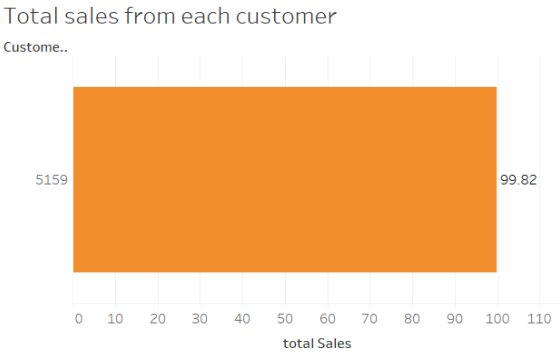
Fun Fact:

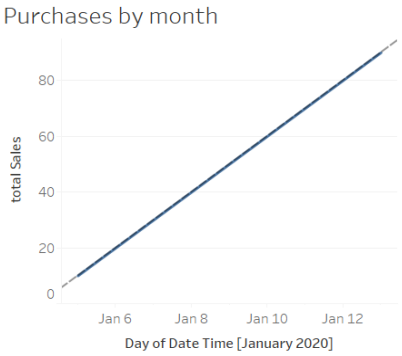
Even though customer 8750, has visited the store the most, his contribution to the revenue is only $57.25 for all his visits combined.

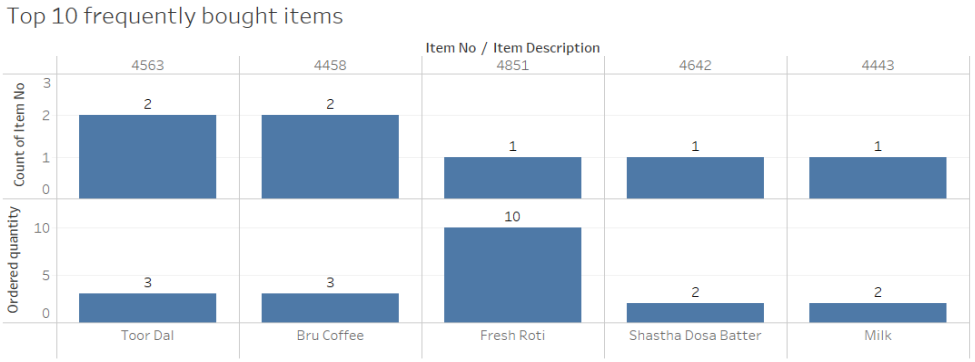
This customer has preferred buying from restaurant more than from the grocery store. His contribution to the revenue has increased over time, as indicated by the trend line in the purchase by month graph.

Customer 5159:









Fun fact

We see from the above graphs that, though customer 5159 has not visited that frequently, this customer has contributed the most in terms of sales. This could also be seen by the purchase by month chart, which is increasing.

Conclusion from analyses:

- As seen from the visualizations, the customer who visits more doesn't necessarily purchase more and contribute to the revenue and the customer who has visited less turned out to be more valuable.
- Hence, to analyze which customer is more valuable, we may need additional information to classify or segment these customers.
- Most customers prefer fresh food from the restaurant compared to frozen food.
- More customers visit the stores on weekends as opposed to weekdays.
- Sales is more in the weekend, leading to the week of any Indian Festival.

Suggestions to improve the business of ICC

- Using these purchase data, we could identify the customers who visit the stores but purchase less and try to turn them to loyal customers by giving them additional offers or through rewards program.
- We saw that most customers purchase during the weekends (Saturday) and the week leading to any Indian festival. The stores should allocate more employees to assist the customers and manage the counters. So that, the customers could buy in the shortest time possible.
- The stores can also give additional discounts to customers who purchase more than a certain amount, say $50. Thus, when a customer is close to that price, we could upsell more items to them.