1. **Sales ----------------------**
   1. Total sales

select round((Sales),2) as TotSales from grocery;

* 1. Total sales by Fats

# Replacing LF with Low Fat

update grocery

set `Item Fat Content` = 'Low Fat'

where `Item Fat Content` = 'LF';

# Replacing reg with Regular

update grocer

set `Item Fat Content` = 'Regular'

where `Item Fat Content` = 'reg';

select `Item Fat Content`, sum(Sales) TotSales from grocery

group by `Item Fat Content`;

* 1. Sales by Item Type

with A AS

(

select `Item Type`, round(sum(Sales),0) as Tsales from grocery

group by 1

)

select \*, dense\_rank() over (order by Tsales desc) as Ranking

from A;

* 1. Sales by Year for each outlet size

select `Outlet Establishment Year` as OutletYear, `Outlet Size`, round(sum(Sales),0) Tsales from grocery

group by 1,2

order by 1 asc

* 1. Calculate the Sales Growth Rate Year-over-Year

alter table grocery **# Renamed Column**

rename column `Outlet Establishment Year` to OutletYear;

select \* from grocery;

with B as

(

with A as

(

select OutletYear, round(sum(Sales),0) Tsales from grocery

group by 1

order by 1

)

select OutletYear, Tsales, lag(Tsales) over (order by OutletYear) as Lsales from A

)

select \*, round(((Tsales -Lsales)/Lsales)\*100.0,2) Growth from B;

1. Identify Outlets with Sales Above 13%

with B as

(

with A as

(

select `Outlet Identifier`, round(sum(Sales),2) Tsales from grocery

group by 1

)

select \*, concat(round(Tsales/(select sum(Sales) from grocery)\*100.0,1), '%') as SalesPcnt from A

)

select \*,

case

when SalesPcnt >= 13 then 'Decent Performing'

else 'Under Performing'

end as Statuss

from B;

1. Calculate the Contribution of Each Item Type to Total Sales

with A as

(

select `Item Type`, round(sum(Sales),2) Tsales,

concat(round(round(sum(Sales),2)/(SELECT sum(Sales) from grocery)\*100.0 , 2), '%') as Contribution

from grocery

group by 1

)

Select \*, dense\_rank() over(order by Tsales desc) Ranking from A;

1. **Ratings ----------------------**
   1. **Calculate Avg ratings for each Item Type**

select `Item Type` ,

round(avg(Rating),2) Ratings, dense\_rank() over (order by round(avg(Rating),2) desc) Rankings

from grocery

group by 1;

* 1. **List the outlets with the highest average rating**

select `Outlet Identifier`, round(avg(Rating),3) Ratings from grocery

group by 1

order by 2 desc;

* 1. **Find the lowest rated item in each outlet**

with B as

(

With A as

(

select `Outlet Identifier`, `Item Type`,avg(Rating) as Ratingss from grocery

group by 1,2

order by 1,3

)

select \*, dense\_rank() over (partition by `Outlet Identifier` order by Ratingss) Rankings from A

)

select \* from B

where Rankings < 4;

* VIEWS ---

**Create a view to see the average rating for each item type**

**select `Item Type`, avg(Rating) as Ratingss from grocery -- COPY THIS SYNTAX IN VIEWS --**

**where `Item Type` = 'Fruits and Vegetables'**

**group by 1**

**;**

**-- VIEWS --**

**CREATE**

**ALGORITHM = UNDEFINED**

**DEFINER = `root`@`localhost`**

**SQL SECURITY DEFINER**

**VIEW `ratingbyitem` AS**

**SELECT**

**`grocery`.`Item Type` AS `Item Type`,**

**AVG(`grocery`.`Rating`) AS `Ratingss`**

**FROM**

**`grocery`**

**GROUP BY `grocery`.`Item Type`**

* **Stored Procedures**

1. Create a stored procedure to get the average rating for a specific item type

**select `Item Type`, avg(Rating) as Ratingss from grocery**

**where `Item Type` = Enter\_item**

**group by 1;**

**CREATE DEFINER=`root`@`localhost` PROCEDURE `ratings`(IN Enter\_item varchar(20))**

**BEGIN**

**select `Item Type`, avg(Rating) as Ratingss from grocery**

**where `Item Type` = Enter\_item**

**group by 1;**

**END**

**call ratings('Canned');**

**B.** Create a stored procedure to get the highest rated item in a specific outlet  
  
with A as

(

select `Outlet Identifier`,

`Item Type`,

avg(Rating) as Ratingss,

dense\_rank() over (partition by `Outlet Identifier` order by avg(Rating) desc) as Rankings

from grocery

group by 1,2

)

select \* from A

where `Outlet Identifier` = OI and Rankings = RK;

call highestratingBYstore('OUT017', 1);