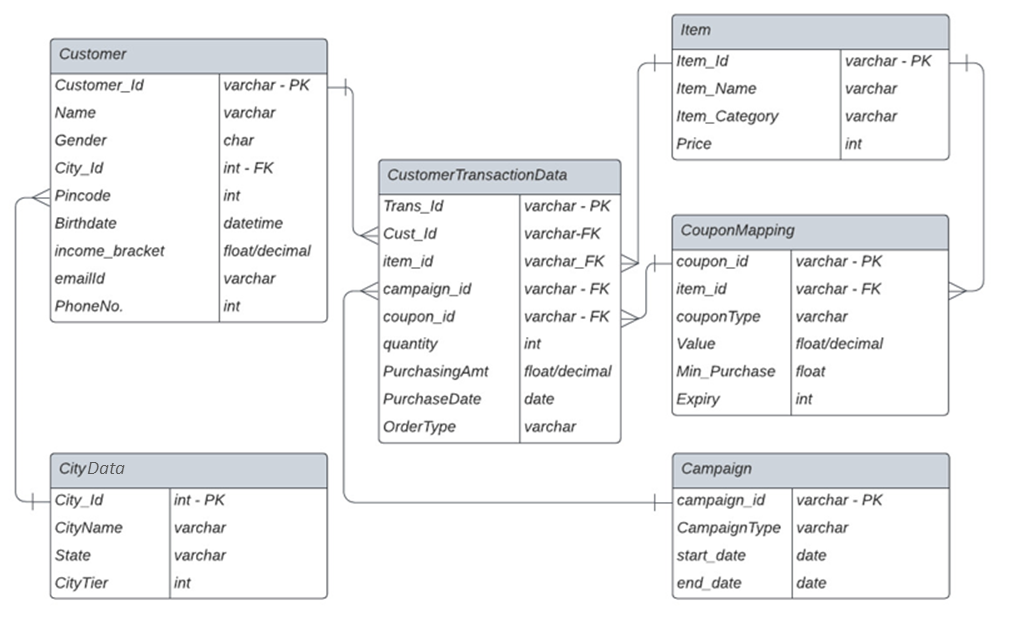
**SQL PROJECT SOLUTION TEMPLATE**

**Marketing Campaign Analysis**

**Section 1: Getting the overview of the data (Total Marks: 15)**

1. **Create the ERD diagram with the help of given schema.**

**Ans:**

****

**2.1.** **Check the cardinality of following columns:** **Different color segments (categories) provided by the company.**

**Ans: 5**

**2.2.** **Different Coupon Types that are offered.**

**Ans: 2**

**2.3.** **States where the company is currently delivering its products and services.**

**Ans: 21**

**2.4.** **Different Order Types.**

**Ans: 3**

**3.1.** **Identify total number of sales (transactions) happened by Yearly basis**

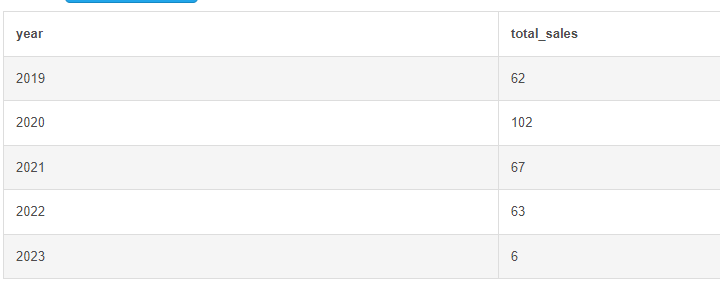
**Ans:**

SELECT YEAR(PurchaseDate) AS year, COUNT(Trans\_Id) AS total\_sales

FROM CustomerTransactionData

GROUP BY YEAR(PurchaseDate);

**Query Snapshot:**

****

**3.2.** **Quarterly basis**

**Ans:**

SELECT YEAR(PurchaseDate) AS year,

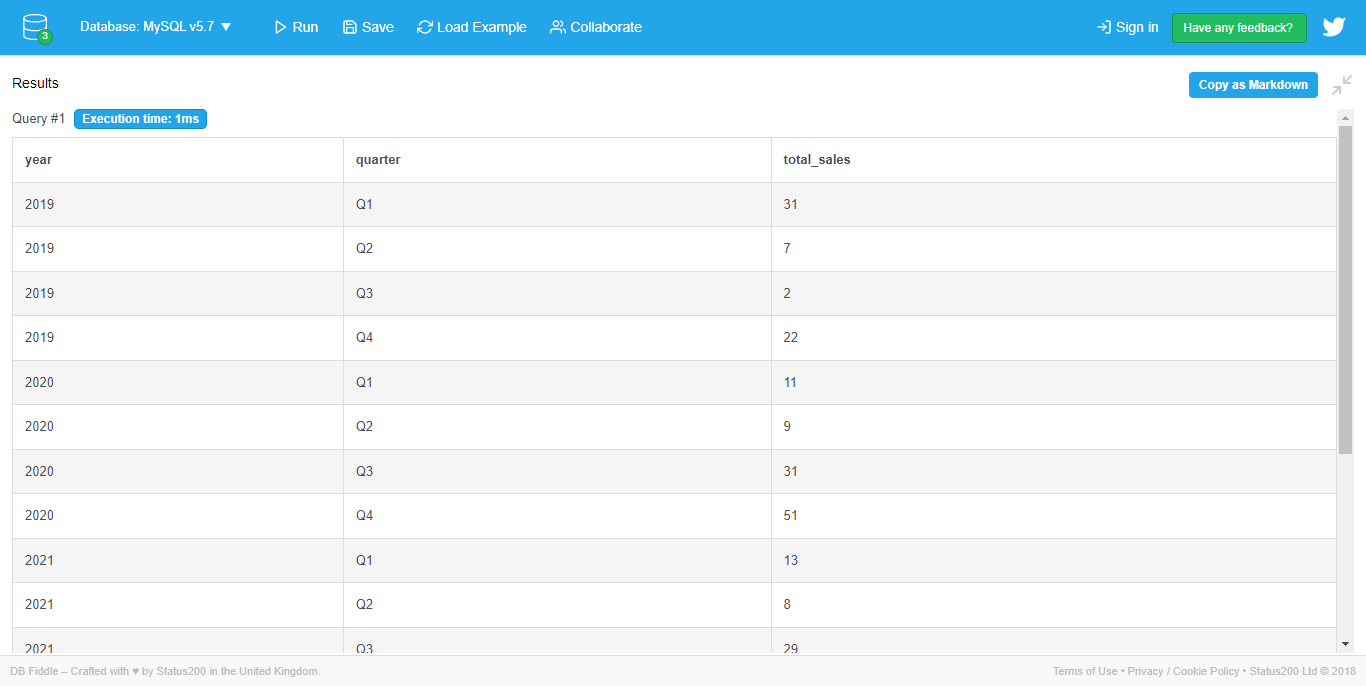
CONCAT('Q', QUARTER(PurchaseDate)) AS quarter,

COUNT(Trans\_Id) AS total\_sales

FROM CustomerTransactionData

GROUP BY YEAR(PurchaseDate), quarter;

**Query Snapshot:**

****

**3.3.** **Yearly and Monthly basis**

**Ans:**

SELECT YEAR(PurchaseDate) AS year,

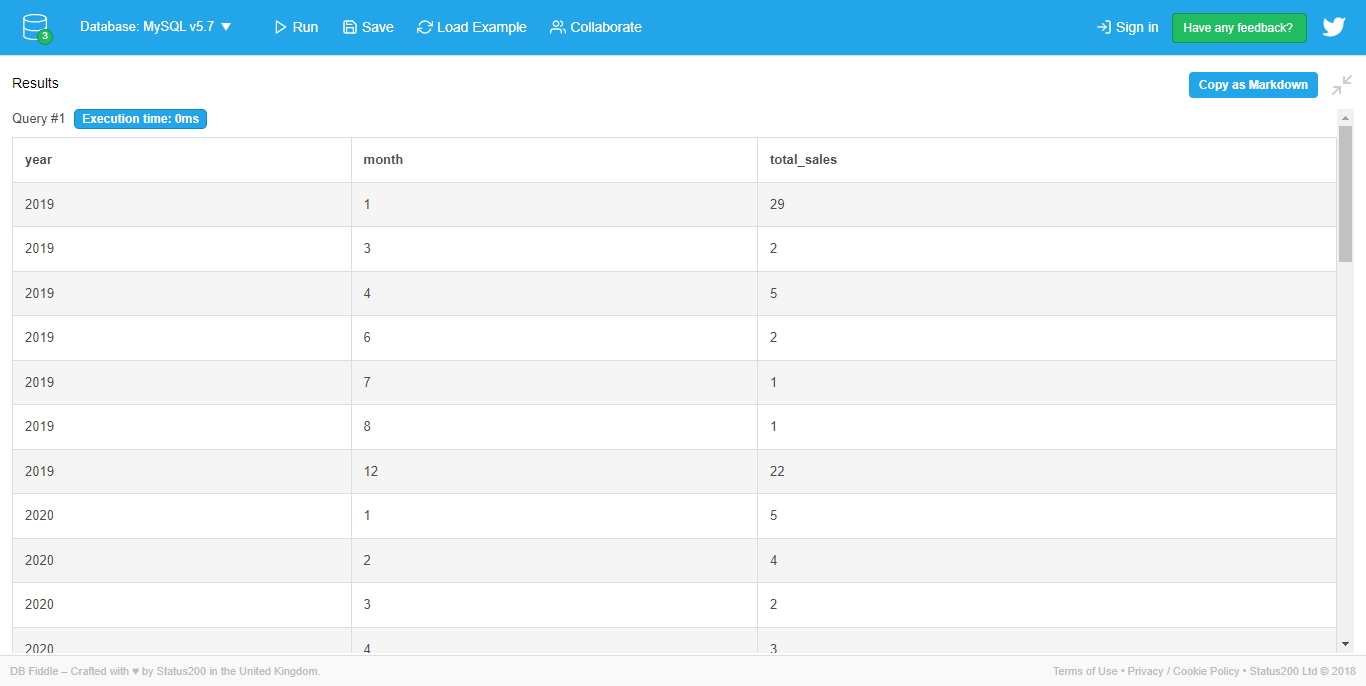
MONTH(PurchaseDate) AS month,

COUNT(Trans\_Id) AS total\_sales

FROM CustomerTransactionData

GROUP BY YEAR(PurchaseDate), MONTH(PurchaseDate);

**Query Snapshot:**

****

**4.1.** **Identify the total purchase order by Product category**

**Ans:**

SELECT Item\_Category, sum(quantity) AS total\_purchase\_orders

FROM CustomerTransactionData ctd

join Item i on ctd.Item\_Id = i.Item\_Id

GROUP BY Item\_Category;

**Query Snapshot:**

****

**4.2.**  **Yearly and Quarterly basis**

**Ans:**

SELECT

  YEAR(PurchaseDate) AS year,

  QUARTER(PurchaseDate) AS quarter,

  sum(quantity) AS total\_orders

FROM

  CustomerTransactionData

GROUP BY

  YEAR(PurchaseDate),

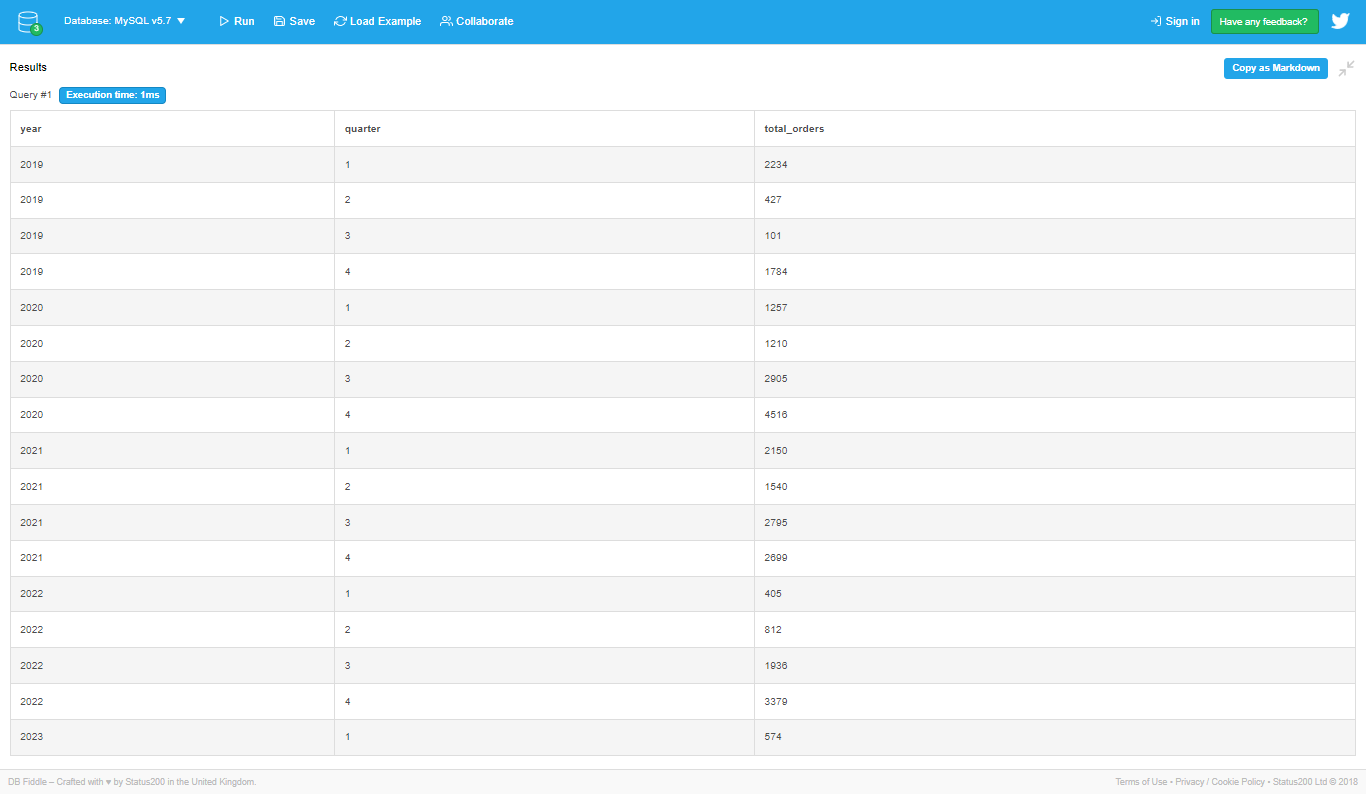
  QUARTER(PurchaseDate)

ORDER BY

  YEAR(PurchaseDate),

  QUARTER(PurchaseDate)

**Query Snapshot:**

****

**4.3.**  **Order Type**

**Ans:**

SELECT

  OrderType,

  SUM(quantity) AS total\_orders

FROM

  CustomerTransactionData

GROUP BY OrderType ;

**Query Snapshot:**

****

**4.4.**  **City Tier**

**Ans:**

SELECT

  CityTier,

  SUM(quantity) AS total\_orders

FROM

  CustomerTransactionData ctd JOIN Customer cu ON ctd.Cust\_Id = cu.Customer\_Id JOIN City c ON cu.City\_Id = c.City\_Id

GROUP BY CityTier ;

**Query Snapshot:**

****

**Section 2: Understanding lead conversions (Total Marks: 20)**

**1.** **1. Identify the total number of transactions with campaign coupon vs total number of transactions without campaign coupon.**

**Ans:**

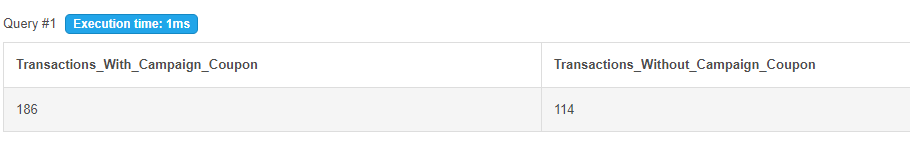
SELECT

SUM(CASE WHEN campaign\_id IS NOT NULL THEN 1 ELSE 0 END) AS Transactions\_With\_Campaign\_Coupon,

SUM(CASE WHEN campaign\_id IS NULL THEN 1 ELSE 0 END) AS Transactions\_Without\_Campaign\_Coupon

FROM CustomerTransactionData;

**Query Snapshot:**

****

**2. Identify the number of customers with first purchase done with or without campaign coupons.**

**Ans:**

WITH Occurences AS

(

    SELECT

        \*,

        ROW\_NUMBER () OVER (PARTITION BY Cust\_Id order by purchasedate ) AS "Occurence"

    FROM CustomerTransactionData

)

SELECT

    Count (\*) AS Count\_Value ,

    CASE

        WHEN coupon\_id is Null

     THEN 'Cust\_Without\_Coupon'

     ELSE 'Cust\_With\_Coupon'

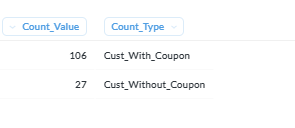
    END AS Count\_Type

FROM Occurences

WHERE Occurence = 1

Group By Count\_Type

**Query Snapshot:**

****

**3a. Identify the impact of campaigns on users.**

**Check the total number of unique users making purchases with or without campaign coupons.**

**Ans:**

SELECT

     Count ( distinct Cust\_Id) AS Count\_Value ,

    CASE

        WHEN coupon\_id is Null

            THEN 'Cust\_Without\_Coupon'

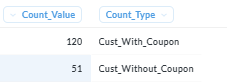
        ELSE 'Cust\_With\_Coupon'

    END AS Count\_Type

FROM CustomerTransactionData

Group By Count\_Type

**Query Snapshot:**

****

**3b.** **Identify the impact of campaigns on users.**

**Check the purchase amount with campaign coupons vs normal coupons vs no coupons.**

**Ans:**

SELECT

   SUM ( PurchasingAmt) AS Purchase\_Value ,

    CASE

        WHEN coupon\_id is Null and campaign\_id is NULL

            THEN 'No\_Coupon'

        WHEN coupon\_id is NOT Null and campaign\_id is NULL

        THEN 'Normal\_Coupon'

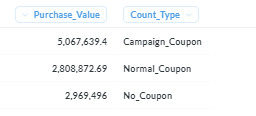
        ELSE 'Campaign\_Coupon'

    END AS Count\_Type

FROM CustomerTransactionData

Group By Count\_Type

**Query Snapshot:**

****

**Comment:**

**Section 3: Understanding company growth and decline (Total Marks: 20)**

**1a. Identify the total growth on an year by year basis excluding the current year Based on quantity of paint that is sold**

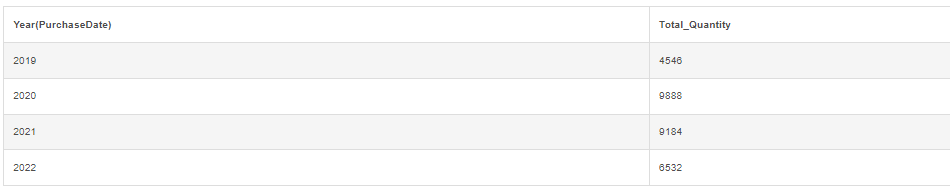
**Ans:**

select Year(PurchaseDate), sum(quantity) as Total\_Quantity from CustomerTransactionData where Year(PurchaseDate ) != 2023

group by Year(PurchaseDate)

order by Year(PurchaseDate);

**Query Snapshot:**

****

**1b.** **Based on amount of paint that is sold**

**Ans:**

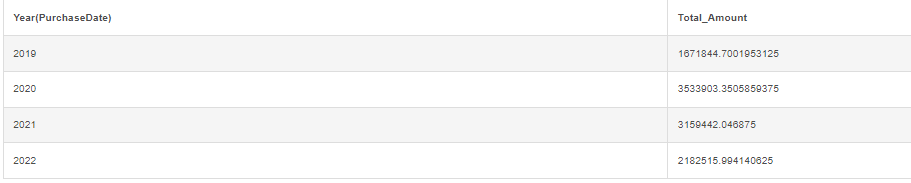
select Year(PurchaseDate), sum(PurchasingAmt) as Total\_Amount from CustomerTransactionData

where Year(PurchaseDate ) != 2023

group by Year(PurchaseDate)

 order by Year(PurchaseDate);

**Query Snapshot:**

****

**1c.** **Based on new customers that are acquired. (Hint: Get distinct new users every year before year by year analysis).**

**Ans:**

with

t1 as

(

select distinct count(cust\_id) as No\_Of\_New\_Customers\_2019 from CustomerTransactionData where Year(PurchaseDate)=2019

),

t2 as

(

select distinct count(cust\_id) as No\_Of\_New\_Customers\_2020 from CustomerTransactionData where Year(PurchaseDate)=2020 and cust\_id not in(

select distinct cust\_id from CustomerTransactionData where Year(PurchaseDate)=2019 )

),

t3 as

(

select distinct count(cust\_id) as No\_Of\_New\_Customers\_2021 from CustomerTransactionData where Year(PurchaseDate)=2021 and cust\_id not in(

select distinct cust\_id from CustomerTransactionData where Year(PurchaseDate)=2019 and Year(PurchaseDate)= 2020 )

),

t4 as

(

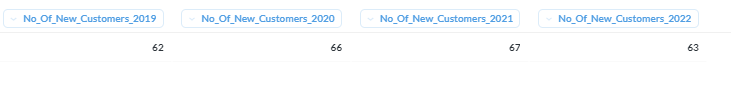
select distinct count(cust\_id) as No\_Of\_New\_Customers\_2022 from CustomerTransactionData where Year(PurchaseDate)=2022 and cust\_id not in(

select distinct cust\_id from CustomerTransactionData where Year(PurchaseDate)=2019 and Year(PurchaseDate)= 2020 and Year(PurchaseDate)= 2021)

  )

  select \* from t1,t2,t3,t4;

**Query Snapshot:**

****

**1c - i (subpart)** **. Segregate them By OrderType (Note: This is a new question, sub-part of 1c)**

**Ans:**

Select Years, NumberOfCustomers, NumberOfCustomers - Lag(NumberOfCustomers) Over ( partition by OrderType Order By Years) As YearByYear,OrderType

From

(Select Year(PurchaseDate) As Years, Count(Distinct Cust\_Id) As NumberOfCustomers,OrderType

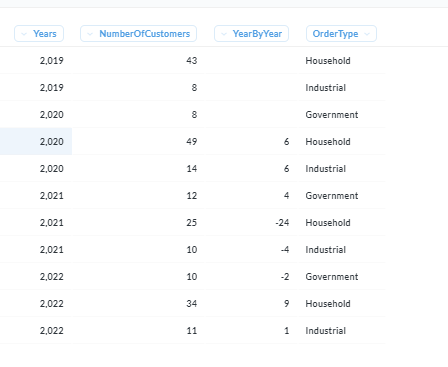
From CustomerTransactionData

Where Year(PurchaseDate) <> Year(Now())

Group By OrderType, Year(PurchaseDate))T1

Order by Years

**Query Snapshot:**

****

**Comment:**

**2. Identify the total decline, if any, within the total sales amount on an year by year basis excluding the current year. Comment on whether we need to launch a campaign for the consumers based on the recent pattern. What campaign type will be more appropriate for this scenario out of all the predefined distinct campaign types? [Note: Unlike previous question, get all the results for different years in their own records]**

**Ans:**

Select Campaign\_Id, Year(PurchaseDate) As Years,

Lag(Sum(PurchasingAmt)) Over (Partition By Year(PurchaseDate)) As YearByYear

From CustomerTransactionData

Group By Campaign\_Id,Years

**Query Snapshot:**

****

**Comment:**

**Section 4: Market basket analysis (Total Marks: 20)**

**1.**  **Please identify the dates when the same customer has purchased some product from the company outlets. Transactions from same order types and different products are only valid transactions here. [Hint: A Special type of Joins is required on customer id and don’t forget to exclude the exact same transactions.]**

**Ans:**

SELECT c1.Cust\_Id,c1.PurchaseDate as PurchaseDate1,

c2.PurchaseDate as PurchaseDate2

FROM CustomerTransactionData as c1

JOIN CustomerTransactionData as c2

ON c1.Cust\_Id = c2.Cust\_Id

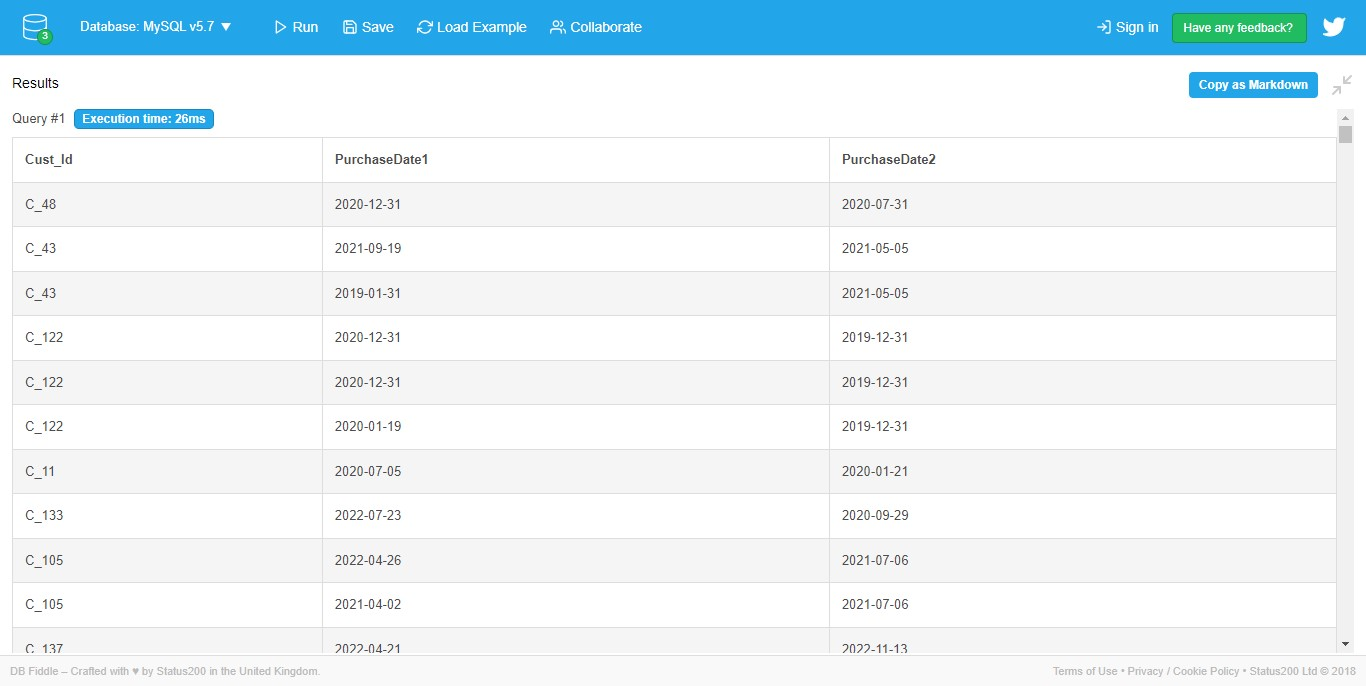
WHERE c1.OrderType = c2.OrderType AND

c1.item\_id != c2.item\_id AND

c1.Trans\_Id != c2.Trans\_Id

;

**Query Snapshot:**



**2.** **Out of the above, please identify the same combination of products coming at least thrice sorted in descending order of their appearance.**

**Ans:**

SELECT c1.Cust\_Id, c1.item\_id as ProductId1, c2.item\_id as ProductId2, c3.item\_id as ProductId3, COUNT(\*) as AppearanceCount

FROM CustomerTransactionData as c1

JOIN CustomerTransactionData as c2

ON c1.Cust\_Id = c2.Cust\_Id

JOIN CustomerTransactionData as c3

ON c1.Cust\_Id = c3.Cust\_Id

WHERE c1.OrderType = c2.OrderType

AND c1.OrderType = c3.OrderType

AND c1.item\_id != c2.item\_id

AND c1.item\_id != c3.item\_id

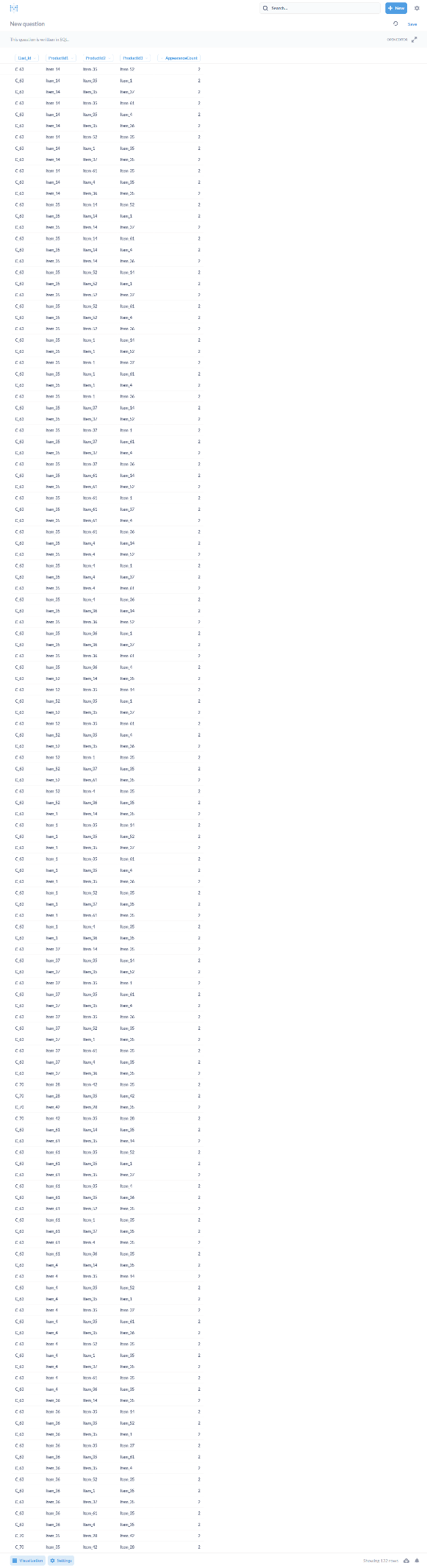
AND c2.item\_id != c3.item\_id

GROUP BY c1.Cust\_Id, c1.item\_id, c2.item\_id, c3.item\_id

HAVING COUNT(\*) >= 2

ORDER BY AppearanceCount DESC;

**Query Snapshot:**



**3.** **Out of the above combinations (coming thrice), please check which of these combinations are popular in different sectors (household, industrial and government).**

**Ans:**

SELECT pc.\*, c.OrderType

FROM (

SELECT c1.Cust\_Id, c1.item\_id as ProductId1, c2.item\_id as ProductId2, c3.item\_id as ProductId3, COUNT(\*) as AppearanceCount

FROM CustomerTransactionData as c1

JOIN CustomerTransactionData as c2

ON c1.Cust\_Id = c2.Cust\_Id

JOIN CustomerTransactionData as c3

ON c1.Cust\_Id = c3.Cust\_Id

WHERE c1.OrderType = c2.OrderType

AND c1.OrderType = c3.OrderType

AND c1.item\_id != c2.item\_id

AND c1.item\_id != c3.item\_id

AND c2.item\_id != c3.item\_id

GROUP BY c1.Cust\_Id, c1.item\_id, c2.item\_id, c3.item\_id

HAVING COUNT(\*) >= 2

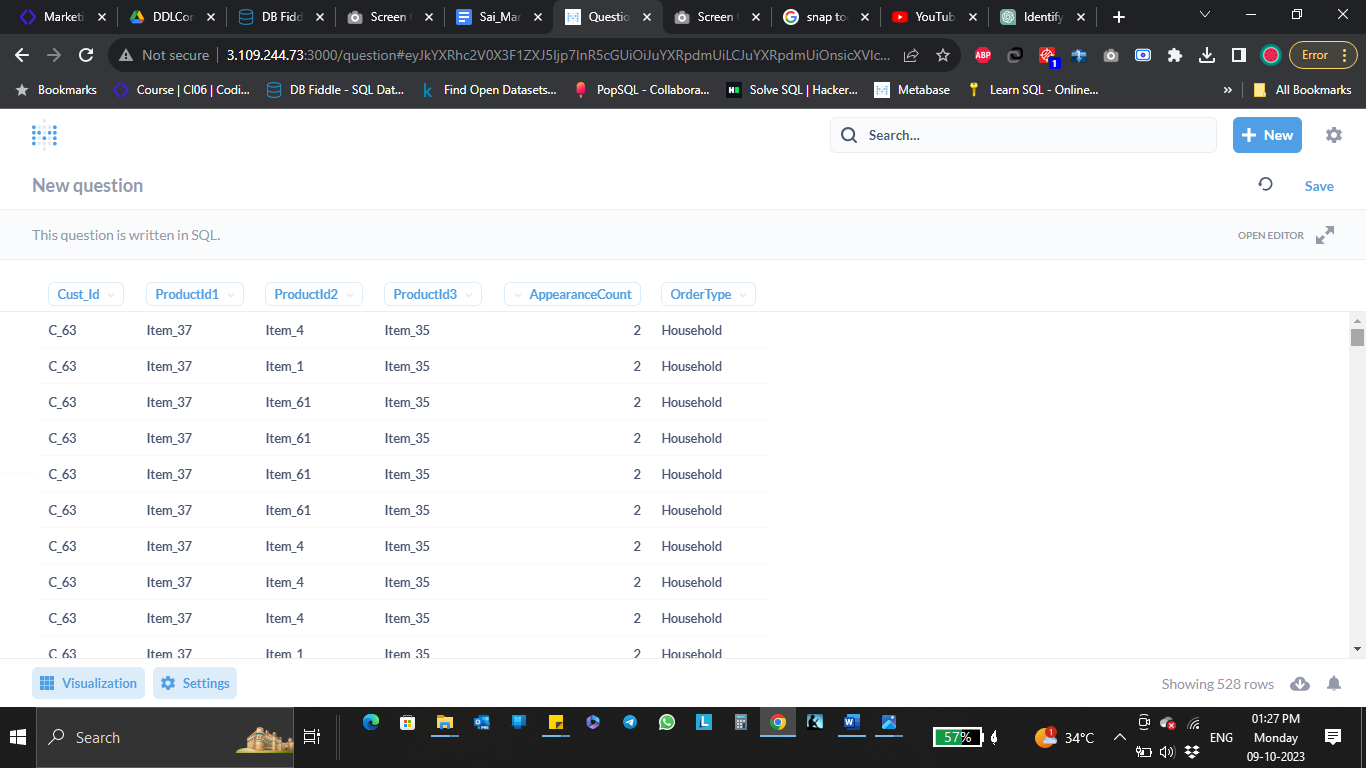
) AS pc

JOIN CustomerTransactionData c ON pc.Cust\_Id = c.Cust\_Id

WHERE c.item\_id IN (pc.ProductId1, pc.ProductId2, pc.ProductId3)

ORDER BY pc.AppearanceCount DESC, c.OrderType;

**Query Snapshot:**



**Comment:**

**Section 5: Automating tasks (Total Marks: 25)**

**1a.** **Create Functions for the following: Get the total discount, if any.**

**Ans:**

DELIMITER $$

CREATE FUNCTION Discount

(Quantity int, Price float, PurchasingAmt float)

RETURNS INT

DETERMINISTIC

BEGIN

DECLARE discount INT;

SET discount = Quantity \* Price - PurchasingAmt;

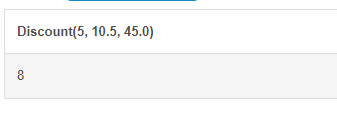
RETURN discount;

END$$

DELIMITER ;

SELECT Discount(5, 10.5, 45.0);

**Query Snapshot:**

****

**1b.** **Get the days/month/year elapsed since the last purchase of a customer depending on input from user. [Hint: Use If condition within the function]**

**Ans:**

DELIMITER $$

CREATE FUNCTION Time\_Elapsed (val VARCHAR(4), date\_last\_purchase DATE)

RETURNS INT

DETERMINISTIC

BEGIN

DECLARE time\_elapsed INT;

SET time\_elapsed = IF(val = 'day', DATEDIFF(NOW(), date\_last\_purchase), YEAR(NOW()) - YEAR(date\_last\_purchase));

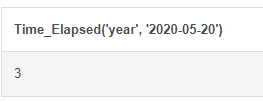
RETURN time\_elapsed;

END$$

DELIMITER ;

SELECT Time\_Elapsed('year', '2020-05-20');

**Query Snapshot:**

****

**2a. Create Views (using above functions) for the following:** **Identify the top 10 customers along with their demographic details from each sector based on their total discount.**

**Ans:**

DELIMITER $$

CREATE FUNCTION Discount

(Quantity int, Price float, PurchasingAmt float)

RETURNS INT

DETERMINISTIC

BEGIN

DECLARE discount INT;

SET discount = Quantity \* Price - PurchasingAmt;

RETURN discount;

END$$

DELIMITER ;

(SELECT c.Cust\_Id ,cu.Name,cu.City\_Id,

ci.City\_Name,

ci.State,

ci.CityTier,c.OrderType ,

SUM(Discount(c.quantity,i.Price,c.PurchasingAmt)) as total\_discount

FROM City ci JOIN

Customer cu ON cu.City\_Id = ci.City\_Id

JOIN CustomerTransactionData as c

ON cu.Customer\_Id = c.Cust\_Id

INNER JOIN Item as i

ON c.Item\_Id = i.Item\_Id

GROUP BY c.Cust\_Id , c.OrderType

HAVING c.OrderType = "Government"

ORDER BY c.OrderType,total\_discount DESC

LIMIT 10)

UNION

(SELECT c.Cust\_Id ,cu.Name,cu.City\_Id,

ci.City\_Name,

ci.State,

ci.CityTier,c.OrderType ,

SUM(Discount(c.quantity,i.Price,c.PurchasingAmt)) as total\_discount

FROM City ci JOIN

Customer cu ON cu.City\_Id = ci.City\_Id

JOIN CustomerTransactionData as c

ON cu.Customer\_Id = c.Cust\_Id

INNER JOIN Item as i

ON c.Item\_Id = i.Item\_Id

GROUP BY c.Cust\_Id , c.OrderType

HAVING c.OrderType = "Household"

ORDER BY c.OrderType,total\_discount DESC

LIMIT 10)

UNION

(SELECT c.Cust\_Id ,cu.Name,cu.City\_Id,

ci.City\_Name,

ci.State,

ci.CityTier,c.OrderType ,

SUM(Discount(c.quantity,i.Price,c.PurchasingAmt)) as total\_discount

FROM City ci JOIN

Customer cu ON cu.City\_Id = ci.City\_Id

JOIN CustomerTransactionData as c

ON cu.Customer\_Id = c.Cust\_Id

INNER JOIN Item as i

ON c.Item\_Id = i.Item\_Id

GROUP BY c.Cust\_Id , c.OrderType

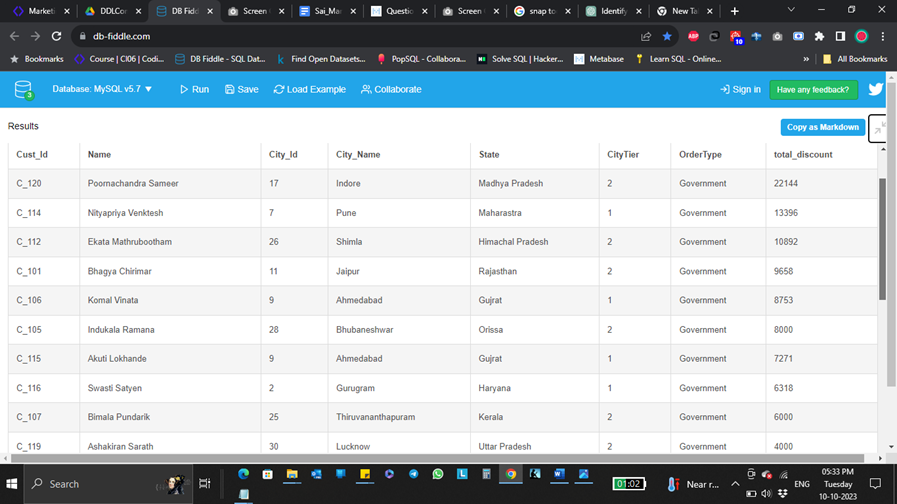
HAVING c.OrderType = "Industrial"

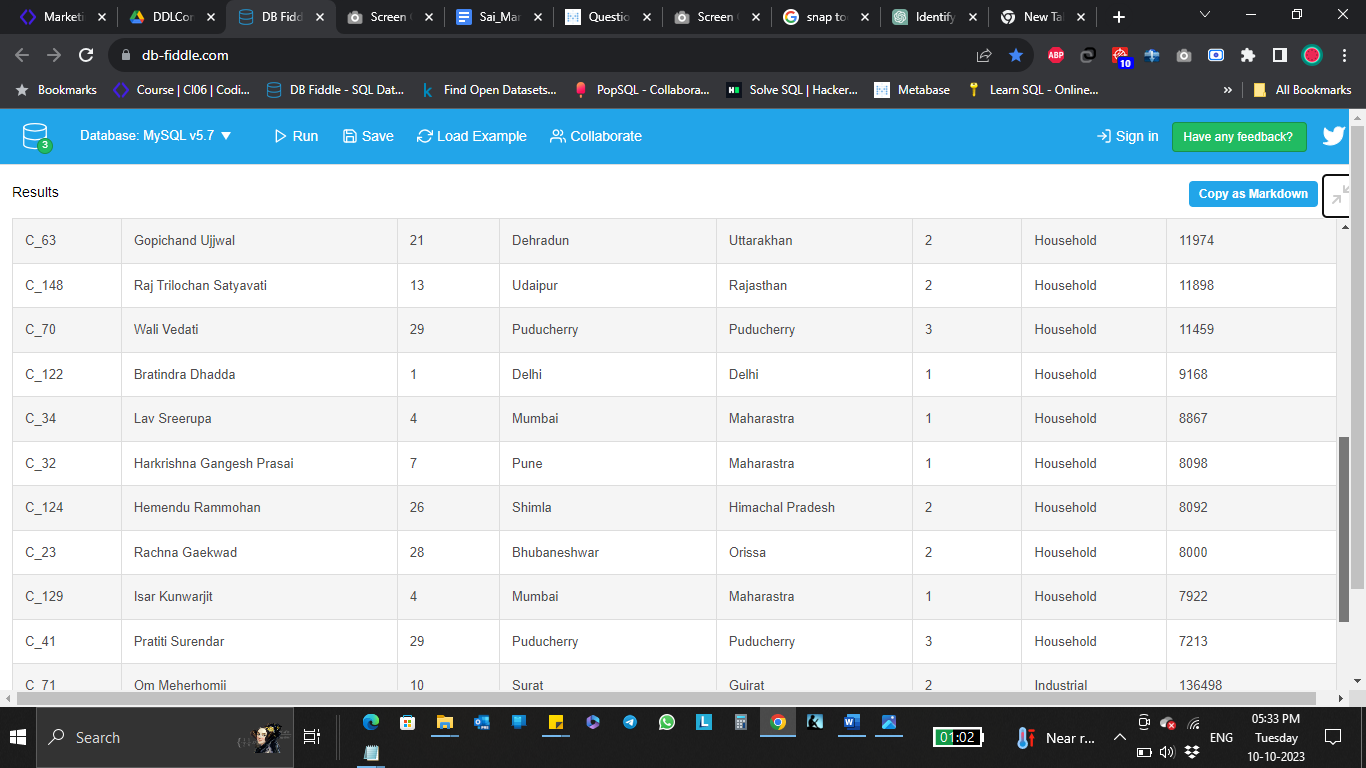
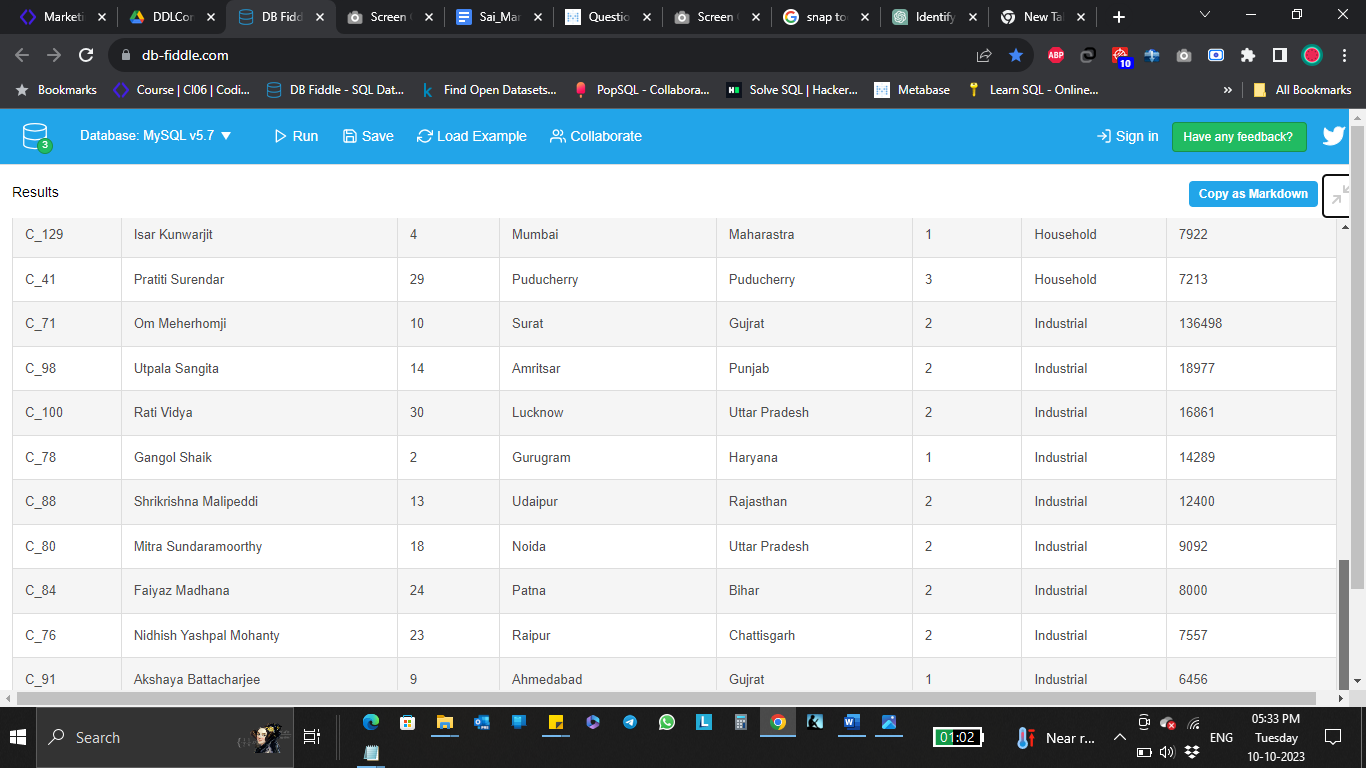
ORDER BY c.OrderType,total\_discount DESC

LIMIT 10)

;

**Query Snapshot:**

****



**2b.** **Identify the top 5 customers (from household and industrial sector) based on purchase amount and days elapsed in descending order. Do highlight if you think there is a data error.**

**Ans:**

CREATE VIEW Top5Customers AS

SELECT

C.Customer\_Id,C.Name,C.Gender,CI.City\_Name,CI.State,T.OrderType,

T.PurchasingAmt,

DATEDIFF(NOW(), T.PurchaseDate) AS DaysElapsed

FROM Customer AS C

JOIN City AS CI ON C.City\_Id = CI.City\_Id

JOIN CustomerTransactionData AS T ON C.Customer\_Id = T.Cust\_Id

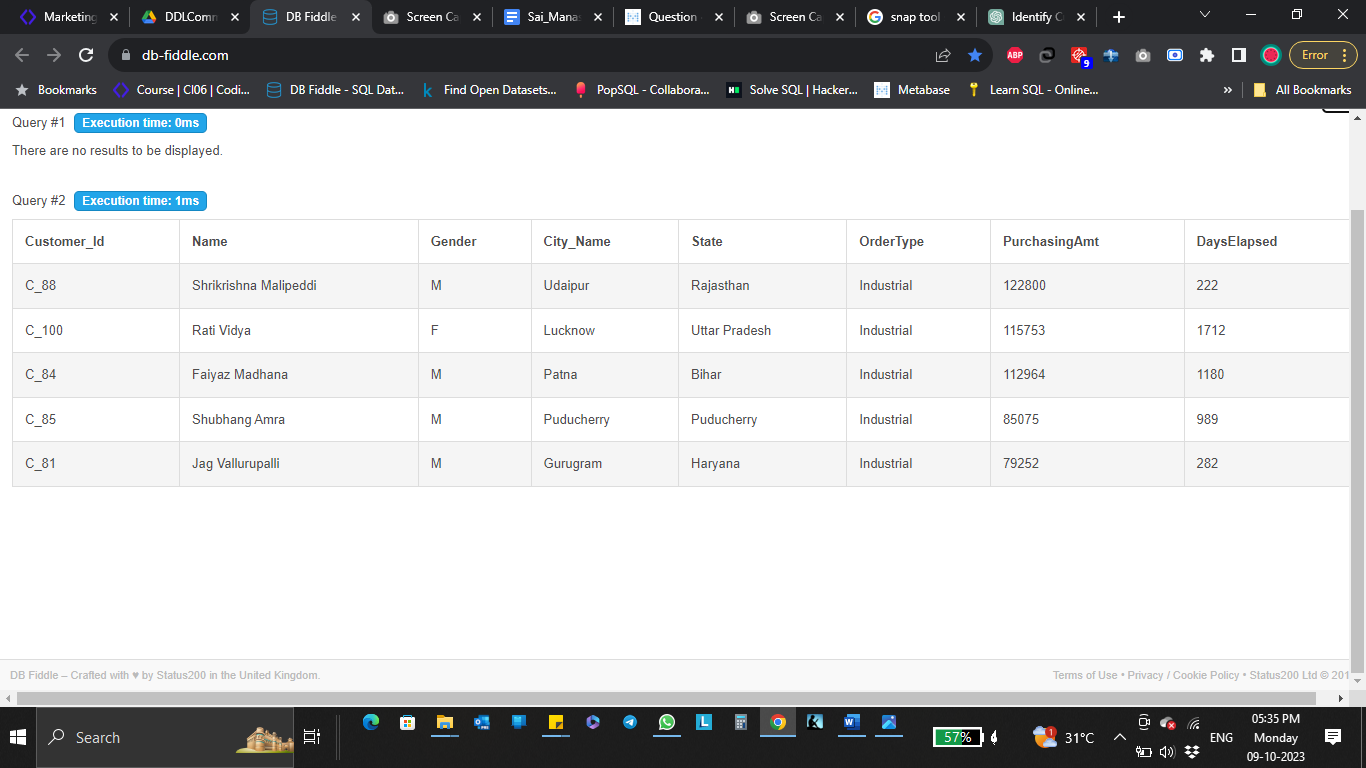
WHERE T.OrderType in ('Household','Industrial')

ORDER BY T.PurchasingAmt DESC,DaysElapsed DESC

LIMIT 5;

SELECT \* FROM Top5Customers;

**Query Snapshot:**



**2c.** **Identify the top 10 products that are sold last year based on sales amount along with the last 2 year details of the same.**

**Ans:**

SELECT

C.item\_id AS ProductId,

I.Item\_Name AS ProductName,

SUM(CASE WHEN YEAR(C.PurchaseDate) = YEAR(NOW()) - 1 THEN C.PurchasingAmt ELSE 0 END) AS LastYear,

SUM(CASE WHEN YEAR(C.PurchaseDate) = YEAR(NOW()) - 2 THEN C.PurchasingAmt ELSE 0 END) AS PreviousYear

FROM CustomerTransactionData C

JOIN Item I ON C.item\_id = I.Item\_Id

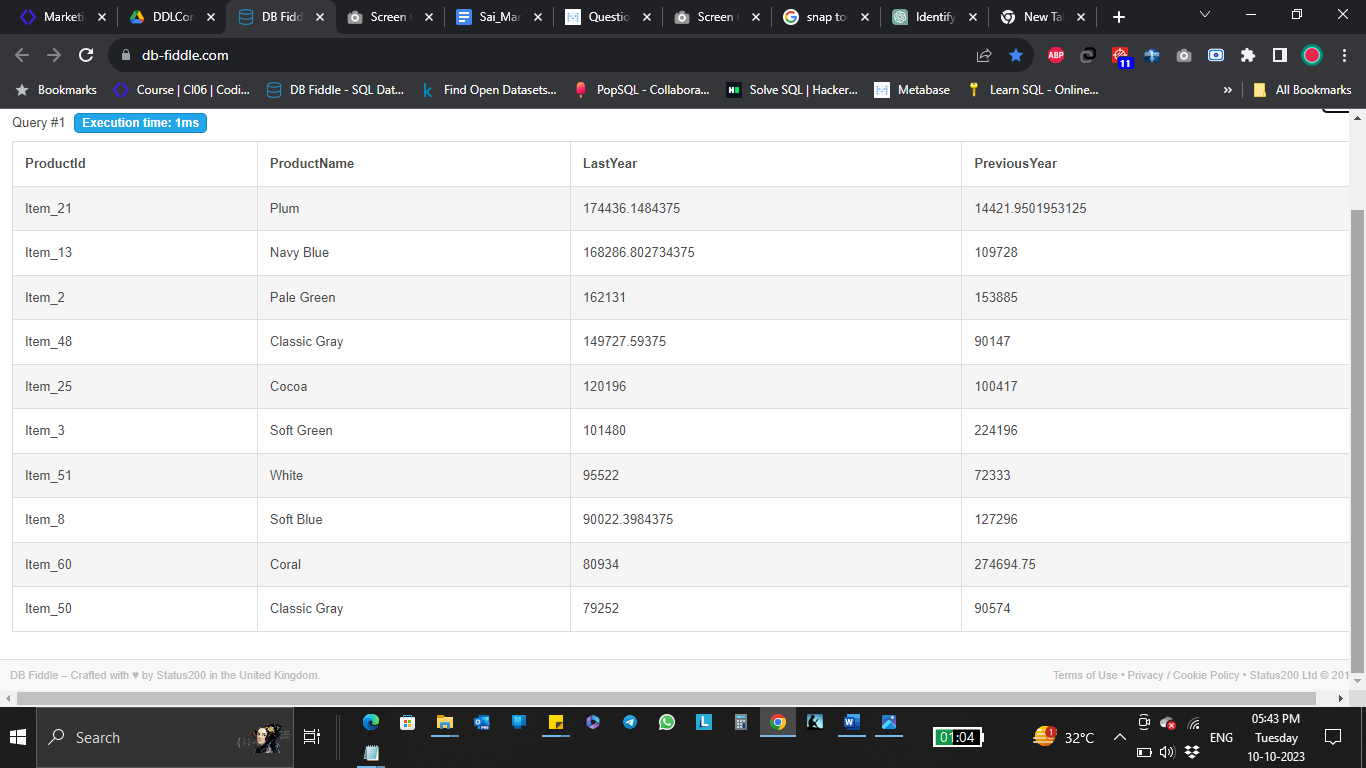
WHERE YEAR(C.PurchaseDate) IN (YEAR(NOW()) - 1, YEAR(NOW()) - 2)

GROUP BY C.item\_id, I.Item\_Name

ORDER BY LastYear DESC

LIMIT 10;

**Query Snapshot:**



**2d.** **Create 3 different income groups for household sector people - ‘high class’, ‘low class’, ‘middle class’ - based on their percent rank (33% each) and identify the top 2 products that are bought within these income class.**

**Ans:**

CREATE VIEW CLASS\_CATEGORY\_PURCHASE AS SELECT \* FROM((select Customer\_Id ,Name, PurchasingAmt, Item\_Id,Class\_Category from(

select

    Customer\_Id,Name, Item\_Id,PurchasingAmt,CASE

    WHEN ROUND(

    PERCENT\_RANK()

    OVER (

        ORDER BY income\_bracket

    ),2) <= 0.33 THEN 'Low\_Class'

    WHEN ROUND(

    PERCENT\_RANK()

    OVER (

        ORDER BY income\_bracket

    ),2) > 0.33 AND ROUND(

    PERCENT\_RANK()

    OVER (

        ORDER BY income\_bracket

    ),2) <= 0.66  THEN 'Middle\_Class'

    ELSE 'High\_Class'

END AS Class\_Category

    from Customer c Join CustomerTransactionData ctd

    where c.Customer\_id = ctd.Cust\_Id

    )as T1 where Class\_Category ="Low\_Class" Order By PurchasingAmt desc Limit 2)

    Union

    (select Customer\_Id , Name,PurchasingAmt, Item\_Id,Class\_Category from(

select

    Customer\_Id,Name, Item\_Id,PurchasingAmt,CASE

    WHEN ROUND(

    PERCENT\_RANK()

    OVER (

        ORDER BY income\_bracket

    ),2) <= 0.33 THEN 'Low\_Class'

    WHEN ROUND(

    PERCENT\_RANK()

    OVER (

        ORDER BY income\_bracket

    ),2) > 0.33 AND ROUND(

    PERCENT\_RANK()

    OVER (

        ORDER BY income\_bracket

    ),2) <= 0.66  THEN 'Middle\_Class'

    ELSE 'High\_Class'

END AS Class\_Category

    from Customer c Join CustomerTransactionData ctd

    where c.Customer\_id = ctd.Cust\_Id

    )as T1 where Class\_Category ="Middle\_Class" Order By PurchasingAmt desc Limit 2)

    UNION

    (select Customer\_Id ,Name, PurchasingAmt, Item\_Id,Class\_Category from(

select

    Customer\_Id,Name, Item\_Id,PurchasingAmt,CASE

    WHEN ROUND(

    PERCENT\_RANK()

    OVER (

        ORDER BY income\_bracket

    ),2) <= 0.33 THEN 'Low\_Class'

    WHEN ROUND(

    PERCENT\_RANK()

    OVER (

        ORDER BY income\_bracket

    ),2) > 0.33 AND ROUND(

    PERCENT\_RANK()

    OVER (

        ORDER BY income\_bracket

    ),2) <= 0.66  THEN 'Middle\_Class'

    ELSE 'High\_Class'

END AS Class\_Category

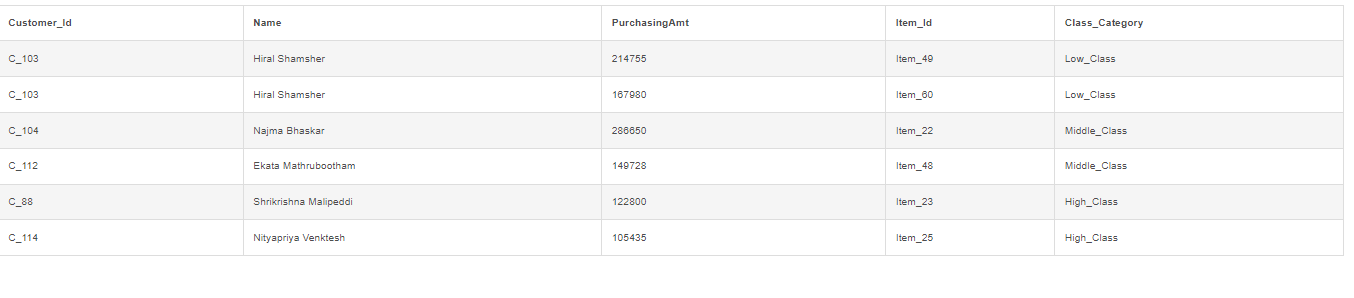
    from Customer c Join CustomerTransactionData ctd

    where c.Customer\_id = ctd.Cust\_Id

    )as T1 where Class\_Category ="High\_Class" Order By PurchasingAmt desc Limit 2)) AS TT1;

  SELECT \* FROM CLASS\_CATEGORY\_PURCHASE;

**Query Snapshot:**

****

**3a.** **Create Stored Procedures for following data validation tasks:**

**Identify whether a particular transaction amount (purchase amount) is ‘correct’ or ‘not correct’.**

**It is correct if price and quantity are used to calculate without a coupon. In case of a coupon, the coupon amount should be deducted from the original amount given the original amount is greater than equal to min purchase for a coupon; else you can simply calculate original amount based on quantity.**

**[Input will be transaction id] [Note: Look out for null coupon ids]**

**Ans:**

DELIMITER $$

CREATE PROCEDURE ValidateAmount(IN Id1 VARCHAR(32), OUT result VARCHAR(50))

BEGIN

SELECT

IF(PurchasingAmt != totalamt,'not correct','correct') as message

INTO result

FROM

(SELECT CT.PurchasingAmt,

IF(CT.coupon\_id IS NOT NULL AND Quantity \* Price >= Min\_Purchase,

Quantity \* Price =

IF(couponType != 'Flat',

Quantity \* Price \* Value\*0.01,Value), Quantity \* Price) AS totalamt

FROM Item as I

JOIN

CustomerTransactionData AS CT

ON I.Item\_id = CT.Item\_id

LEFT JOIN CouponMapping CM

ON CT.coupon\_id=CM.coupon\_id

where CT.Trans\_Id = Id1) as T;

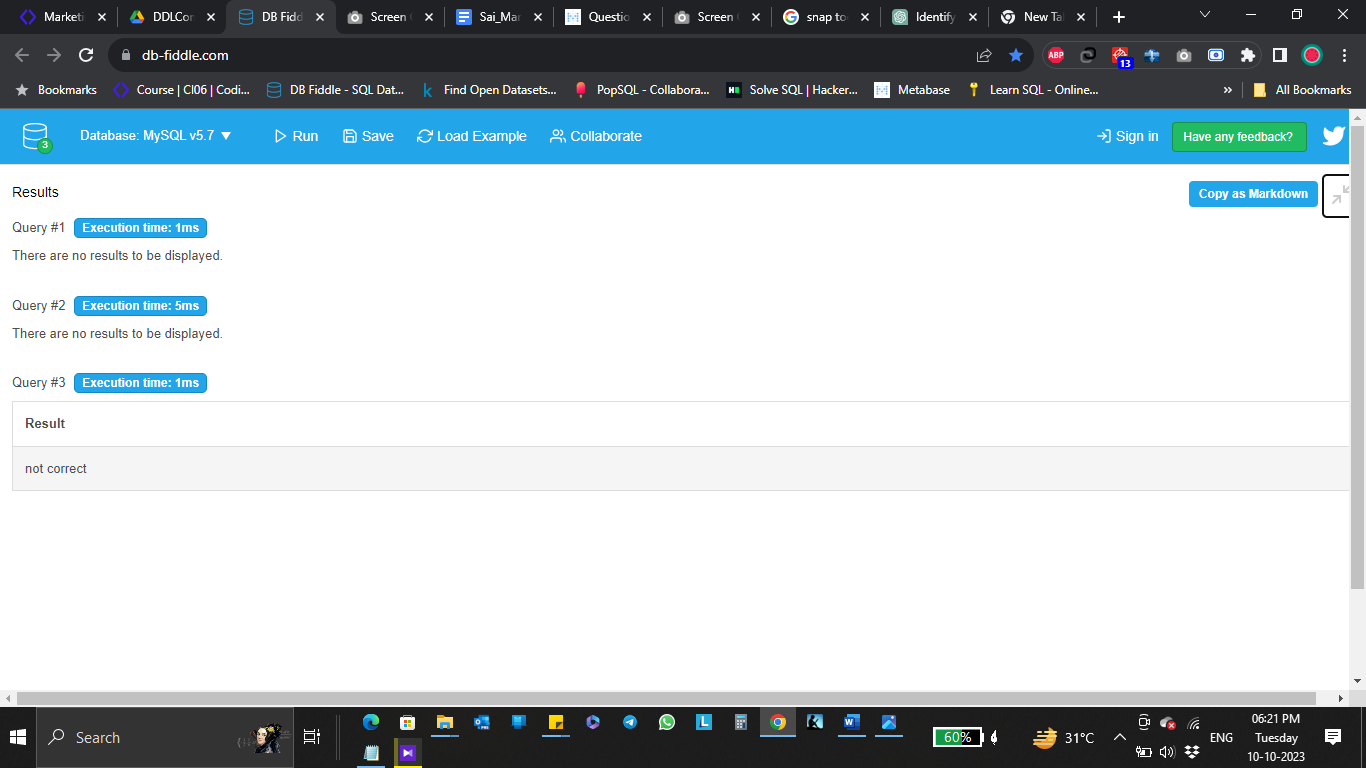
END$$

DELIMITER ;

CALL ValidateAmount('TID00026', @result);

SELECT @result AS Result;

**Query Snapshot:**



**3b.** **Check if there is any customer with age < 12. Print out the total such customers on-screen.**

**Ans:**

DELIMITER &&

CREATE PROCEDURE Check\_Age()

BEGIN

SELECT COUNT (\*) as Count\_Age\_Below\_12 FROM (

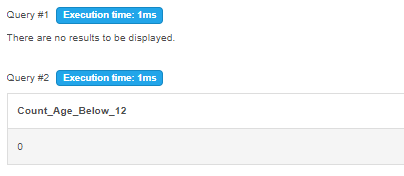
select Customer\_Id,Name,YEAR(NOW())-YEAR(Birthdate) AS AGE from Customer) AS T1 WHERE AGE < 12;

END &&

DELIMITER ;

Call Check\_Age();

**Query Snapshot:**

****