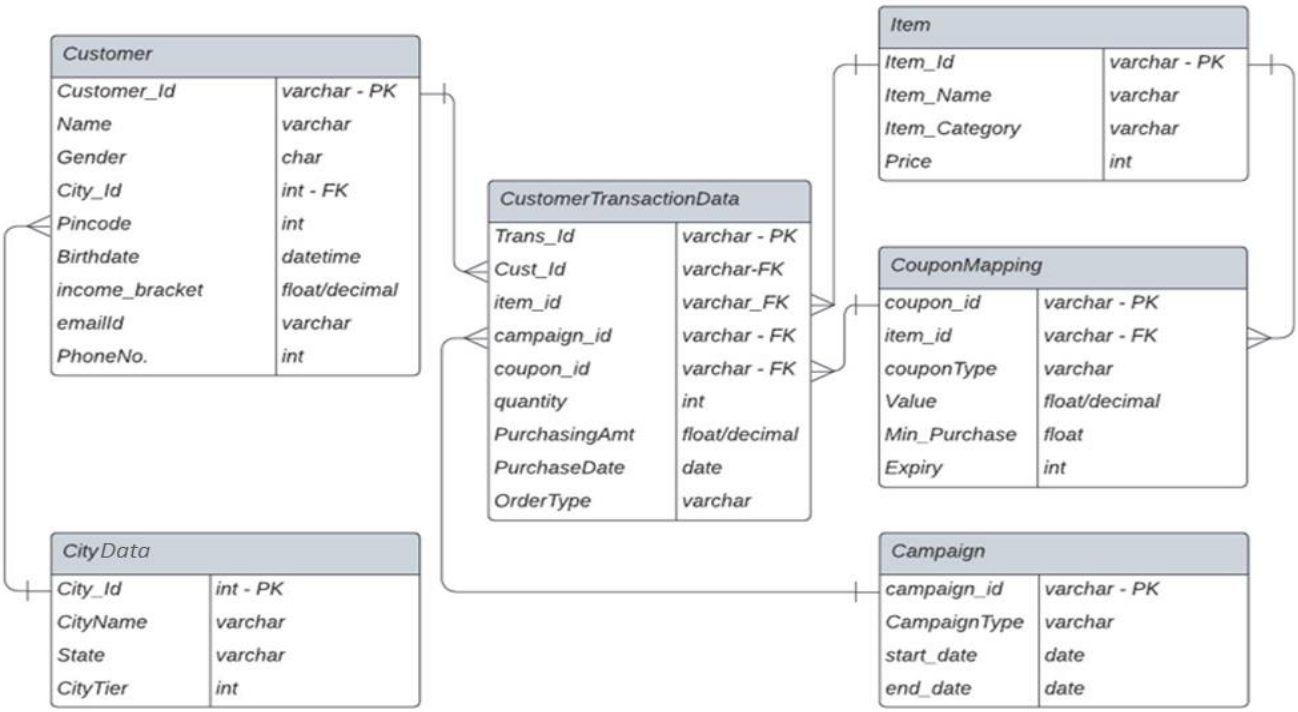
Marketing Campaign Analysis

Section 1: Getting the overview of the data

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



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:

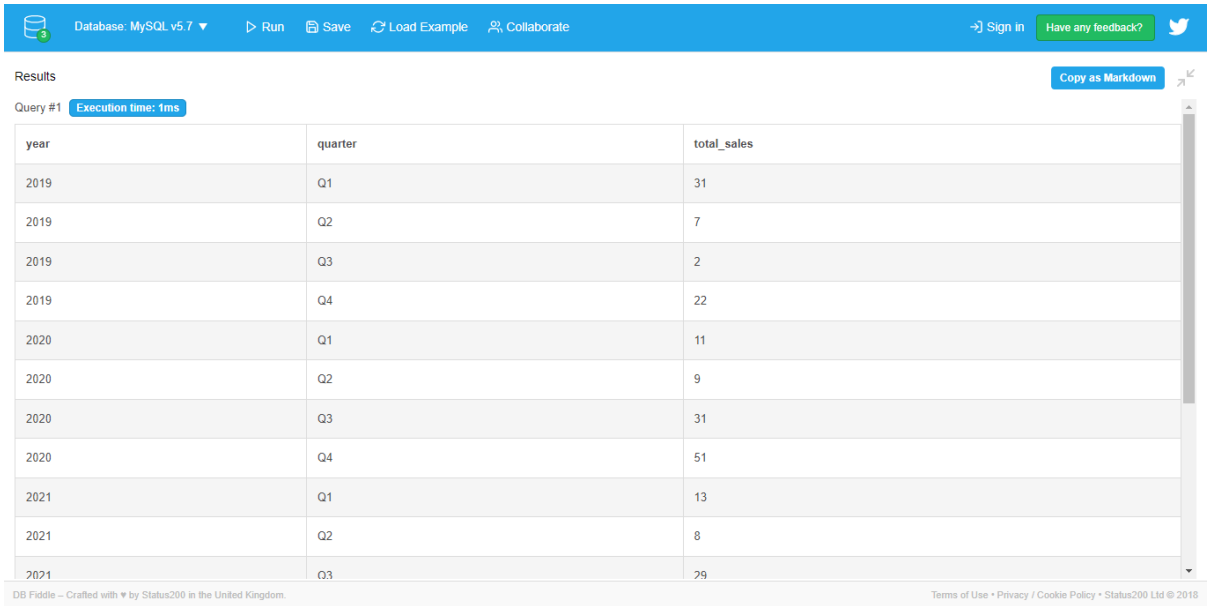
year	total_sales
2019	62
2020	102
2021	67
2022	63
2023	6

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:



year	quarter	total_sales
2019	Q1	31
2019	Q2	7
2019	Q3	2
2019	Q4	22
2020	Q1	11
2020	Q2	9
2020	Q3	31
2020	Q4	51
2021	Q1	13
2021	Q2	8
2021	Q3	29

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:

Database: MySQL v5.7 ▼ ▷ Run Save Load Example Collaborate Sign in Have any feedback?

Results Copy as Markdown

Query #1 Execution time: 0ms

year	month	total_sales
2019	1	29
2019	3	2
2019	4	5
2019	6	2
2019	7	1
2019	8	1
2019	12	22
2020	1	5
2020	2	4
2020	3	2
2020	4	3

DB Fiddle – Crafted with ♥ by Status200 in the United Kingdom. Terms of Use • Privacy / Cookie Policy • Status200 Ltd © 2018

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:

Item_Category	total_purchase_orders
Anti-Corrosive Paint	7524
Emulsion Paint	6593
Enamel Paint	3155
Oil Paint	9026
Synthetic paint	4426

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:

Database: MySQL v5.7 ▾ ▷ Run Save Load Example Collaborate Sign in Have any feedback?

Results

Copy as Markdown

Query #1 Execution time: 1ms

year	quarter	total_orders
2019	1	2234
2019	2	427
2019	3	101
2019	4	1784
2020	1	1257
2020	2	1210
2020	3	2905
2020	4	4516
2021	1	2150
2021	2	1540
2021	3	2795
2021	4	2699
2022	1	405
2022	2	812
2022	3	1936
2022	4	3379
2023	1	574

DB Fiddle – Crafted with ♥ by Status200 in the United Kingdom.

4.3. Order Type

Ans:

```
SELECT
  OrderType,
  SUM(quantity) AS total_orders
FROM
  CustomerTransactionData
GROUP BY OrderType ;
```

Query Snapshot:

OrderType	total_orders
Government	10580
Household	11856
Industrial	8288

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:

CityTier	total_orders
1	8088
2	19297
3	3339

Section 2: Understanding lead conversions

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:

Query #1 Execution time: 1ms

Transactions_With_Campaign_Coupon	Transactions_Without_Campaign_Coupon
186	114

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:

Count_Value	Count_Type
106	Cust_With_Coupon
27	Cust_Without_Coupon

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:

Count_Value	Count_Type
120	Cust_With_Coupon
51	Cust_Without_Coupon

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:

Purchase_Value	Count_Type
5,067,639.4	Campaign_Coupon
2,808,872.69	Normal_Coupon
2,969,496	No_Coupon

Comment:

Section 3: Understanding company growth and decline

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:

Year(PurchaseDate)	Total_Quantity
2019	4546
2020	9888
2021	9184
2022	6532

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:

Year(PurchaseDate)	Total_Amount
2019	1671844.7001953125
2020	3533903.3505859375
2021	3159442.046875
2022	2182515.994140625

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:

No_Of_New_Customers_2019	No_Of_New_Customers_2020	No_Of_New_Customers_2021	No_Of_New_Customers_2022
62	66	67	63

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:

Years	NumberOfCustomers	YearByYear	OrderType
2,019	43		Household
2,019	8		Industrial
2,020	8		Government
2,020	49	6	Household
2,020	14	6	Industrial
2,021	12	4	Government
2,021	25	-24	Household
2,021	10	-4	Industrial
2,022	10	-2	Government
2,022	34	9	Household
2,022	11	1	Industrial

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:

Campaign_Id	Years	YearByYear
CiD2	2,019	
CiD1	2,019	653,497.6
	2,019	811,693.1
CiD3	2,020	
	2,020	782,242.45
CiD4	2,020	1,596,563.15
CiD2	2,020	1,142,861.35
	2,021	
CiD5	2,021	2,635,026.4
CiD4	2,021	479,848.9
	2,022	
CiD6	2,022	1,340,125.14
CiD6	2,023	

Comment:

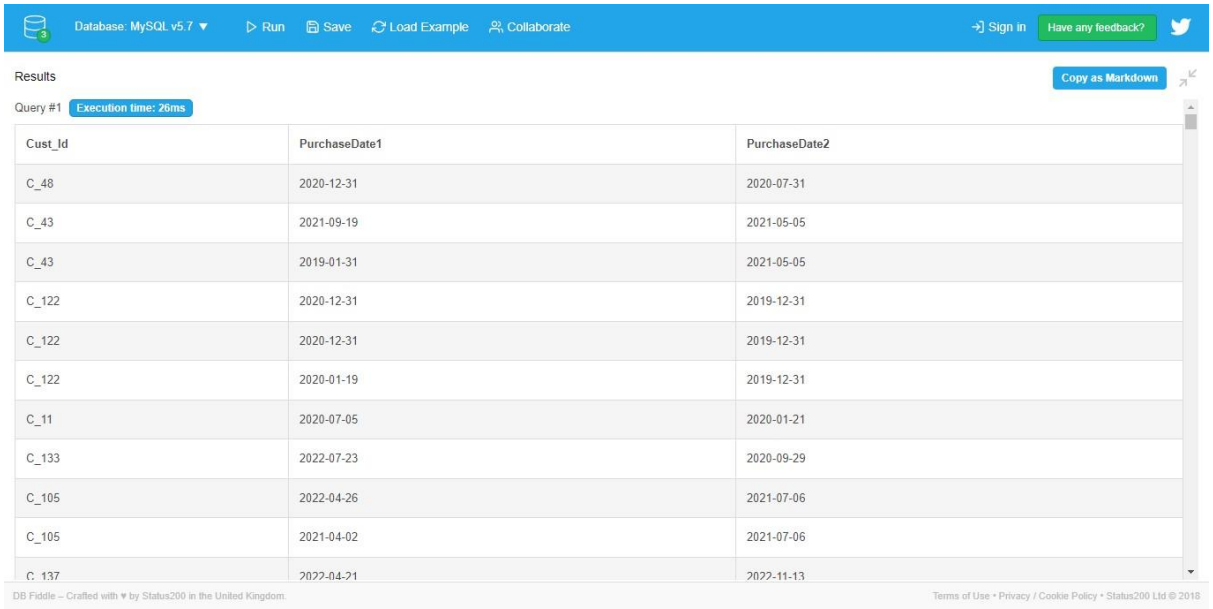
Section 4: Market basket analysis

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:



Cust_Id	PurchaseDate1	PurchaseDate2
C_48	2020-12-31	2020-07-31
C_43	2021-09-19	2021-05-05
C_43	2019-01-31	2021-05-05
C_122	2020-12-31	2019-12-31
C_122	2020-12-31	2019-12-31
C_122	2020-01-19	2019-12-31
C_11	2020-07-05	2020-01-21
C_133	2022-07-23	2020-09-29
C_105	2022-04-26	2021-07-06
C_105	2021-04-02	2021-07-06
C_137	2022-04-21	2022-11-13

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:

[+]

Search...

+ New

New question

Save

This question is written in SQL.

OPEN EDITOR

Cust_Id	ProductId1	ProductId2	ProductId3	AppearanceCount
C_63	Item_14	Item_35	Item_52	2
C_63	Item_14	Item_35	Item_5	2
C_63	Item_14	Item_35	Item_37	2
C_63	Item_14	Item_35	Item_63	2
C_63	Item_14	Item_35	Item_4	2
C_63	Item_14	Item_35	Item_26	2
C_63	Item_14	Item_52	Item_35	2
C_63	Item_14	Item_5	Item_35	2
C_63	Item_14	Item_37	Item_35	2
C_63	Item_14	Item_63	Item_35	2
C_63	Item_14	Item_4	Item_35	2
C_63	Item_14	Item_26	Item_35	2
C_63	Item_35	Item_14	Item_52	2
C_63	Item_35	Item_14	Item_5	2
C_63	Item_35	Item_14	Item_37	2
C_63	Item_35	Item_14	Item_63	2
C_63	Item_35	Item_14	Item_4	2
C_63	Item_35	Item_14	Item_26	2
C_63	Item_35	Item_52	Item_14	2
C_63	Item_35	Item_52	Item_5	2
C_63	Item_35	Item_52	Item_37	2
C_63	Item_35	Item_52	Item_63	2
C_63	Item_35	Item_52	Item_4	2
C_63	Item_35	Item_52	Item_26	2
C_63	Item_35	Item_5	Item_14	2
C_63	Item_35	Item_5	Item_52	2
C_63	Item_35	Item_5	Item_37	2
C_63	Item_35	Item_5	Item_63	2
C_63	Item_35	Item_5	Item_4	2
C_63	Item_35	Item_5	Item_26	2
C_63	Item_35	Item_37	Item_14	2
C_63	Item_35	Item_37	Item_52	2
C_63	Item_35	Item_37	Item_5	2
C_63	Item_35	Item_37	Item_63	2
C_63	Item_35	Item_37	Item_4	2
C_63	Item_35	Item_37	Item_26	2
C_63	Item_35	Item_63	Item_14	2
C_63	Item_35	Item_63	Item_52	2
C_63	Item_35	Item_63	Item_5	2
C_63	Item_35	Item_63	Item_37	2
C_63	Item_35	Item_63	Item_4	2
C_63	Item_35	Item_63	Item_26	2
C_63	Item_35	Item_4	Item_14	2
C_63	Item_35	Item_4	Item_52	2
C_63	Item_35	Item_4	Item_5	2
C_63	Item_35	Item_4	Item_37	2
C_63	Item_35	Item_4	Item_63	2
C_63	Item_35	Item_4	Item_26	2
C_63	Item_35	Item_26	Item_14	2
C_63	Item_35	Item_26	Item_52	2
C_63	Item_35	Item_26	Item_5	2
C_63	Item_35	Item_26	Item_37	2
C_63	Item_35	Item_26	Item_63	2
C_63	Item_35	Item_26	Item_4	2
C_63	Item_52	Item_35	Item_14	2
C_63	Item_52	Item_35	Item_5	2
C_63	Item_52	Item_35	Item_37	2
C_63	Item_52	Item_35	Item_63	2
C_63	Item_52	Item_35	Item_4	2
C_63	Item_52	Item_35	Item_26	2
C_63	Item_52	Item_5	Item_35	2
C_63	Item_52	Item_37	Item_35	2
C_63	Item_52	Item_63	Item_35	2
C_63	Item_52	Item_4	Item_35	2
C_63	Item_52	Item_26	Item_35	2
C_63	Item_5	Item_14	Item_35	2
C_63	Item_5	Item_35	Item_52	2
C_63	Item_5	Item_35	Item_37	2
C_63	Item_5	Item_35	Item_63	2
C_63	Item_5	Item_35	Item_4	2
C_63	Item_5	Item_35	Item_26	2
C_63	Item_5	Item_52	Item_35	2
C_63	Item_5	Item_37	Item_35	2
C_63	Item_5	Item_63	Item_35	2
C_63	Item_5	Item_4	Item_35	2
C_63	Item_5	Item_26	Item_35	2
C_63	Item_37	Item_14	Item_35	2
C_63	Item_37	Item_35	Item_52	2
C_63	Item_37	Item_35	Item_5	2
C_63	Item_37	Item_35	Item_63	2
C_63	Item_37	Item_35	Item_4	2
C_63	Item_37	Item_35	Item_26	2
C_63	Item_37	Item_52	Item_35	2
C_63	Item_37	Item_5	Item_35	2
C_63	Item_37	Item_63	Item_35	2
C_63	Item_37	Item_4	Item_35	2
C_63	Item_37	Item_26	Item_35	2
C_70	Item_28	Item_42	Item_35	2
C_70	Item_28	Item_42	Item_35	2
C_70	Item_42	Item_28	Item_35	2
C_70	Item_42	Item_35	Item_28	2
C_63	Item_63	Item_14	Item_35	2
C_63	Item_63	Item_35	Item_14	2
C_63	Item_63	Item_35	Item_52	2
C_63	Item_63	Item_35	Item_5	2
C_63	Item_63	Item_35	Item_37	2
C_63	Item_63	Item_35	Item_4	2
C_63	Item_63	Item_35	Item_26	2
C_63	Item_63	Item_37	Item_35	2
C_63	Item_63	Item_5	Item_35	2
C_63	Item_63	Item_37	Item_35	2
C_63	Item_63	Item_4	Item_35	2
C_63	Item_63	Item_26	Item_35	2
C_63	Item_4	Item_14	Item_35	2
C_63	Item_4	Item_35	Item_14	2
C_63	Item_4	Item_35	Item_52	2
C_63	Item_4	Item_35	Item_5	2
C_63	Item_4	Item_35	Item_37	2
C_63	Item_4	Item_35	Item_63	2
C_63	Item_4	Item_35	Item_26	2
C_63	Item_4	Item_5	Item_35	2
C_63	Item_4	Item_37	Item_35	2
C_63	Item_4	Item_63	Item_35	2
C_63	Item_4	Item_26	Item_35	2
C_63	Item_26	Item_14	Item_35	2
C_63	Item_26	Item_35	Item_52	2
C_63	Item_26	Item_35	Item_5	2
C_63	Item_26	Item_35	Item_37	2
C_63	Item_26	Item_35	Item_63	2
C_63	Item_26	Item_35	Item_4	2
C_63	Item_26	Item_52	Item_35	2
C_63	Item_26	Item_5	Item_35	2
C_63	Item_26	Item_37	Item_35	2
C_63	Item_26	Item_63	Item_35	2
C_63	Item_26	Item_4	Item_35	2
C_70	Item_35	Item_28	Item_42	2
C_70	Item_35	Item_42	Item_28	2

Visualization

Settings

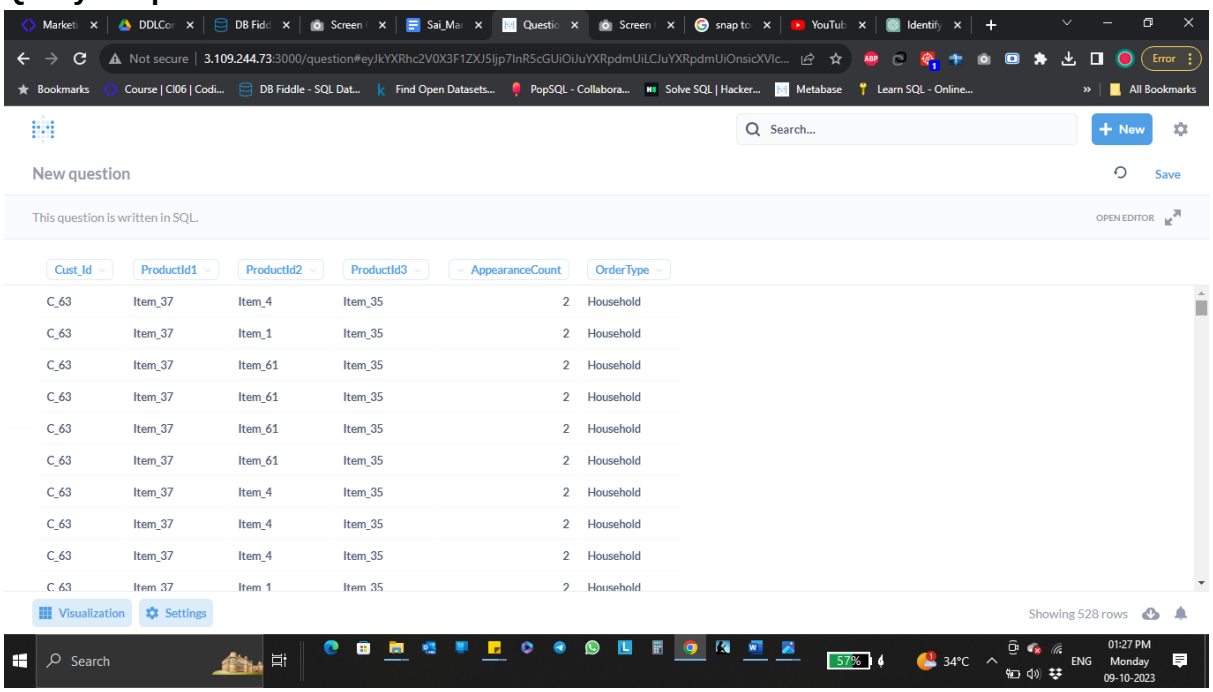
Showing 132 rows

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

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:

Discount(5, 10.5, 45.0)
8

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:

Time_Elapsed('year', '2020-05-20')
3

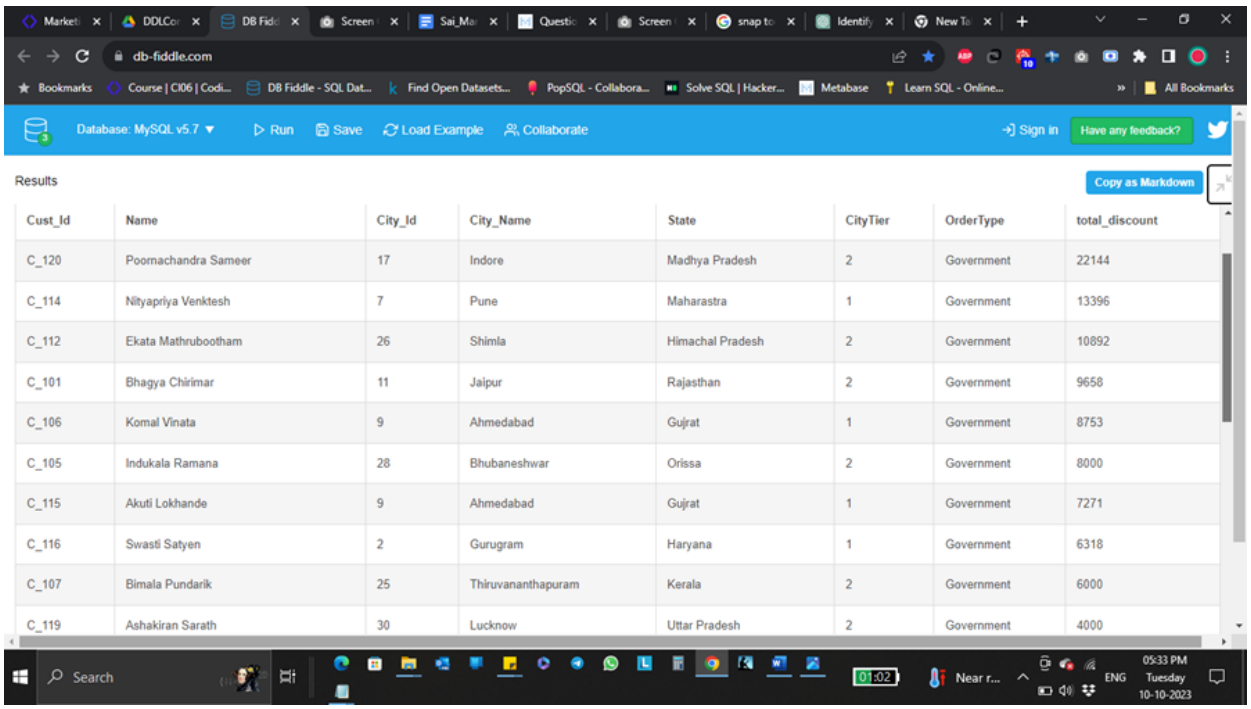
2a. Create Views (using above functions) for the following: Identify the top 10 customers along with their demographic details from each sector based on their total discount.

Ans:

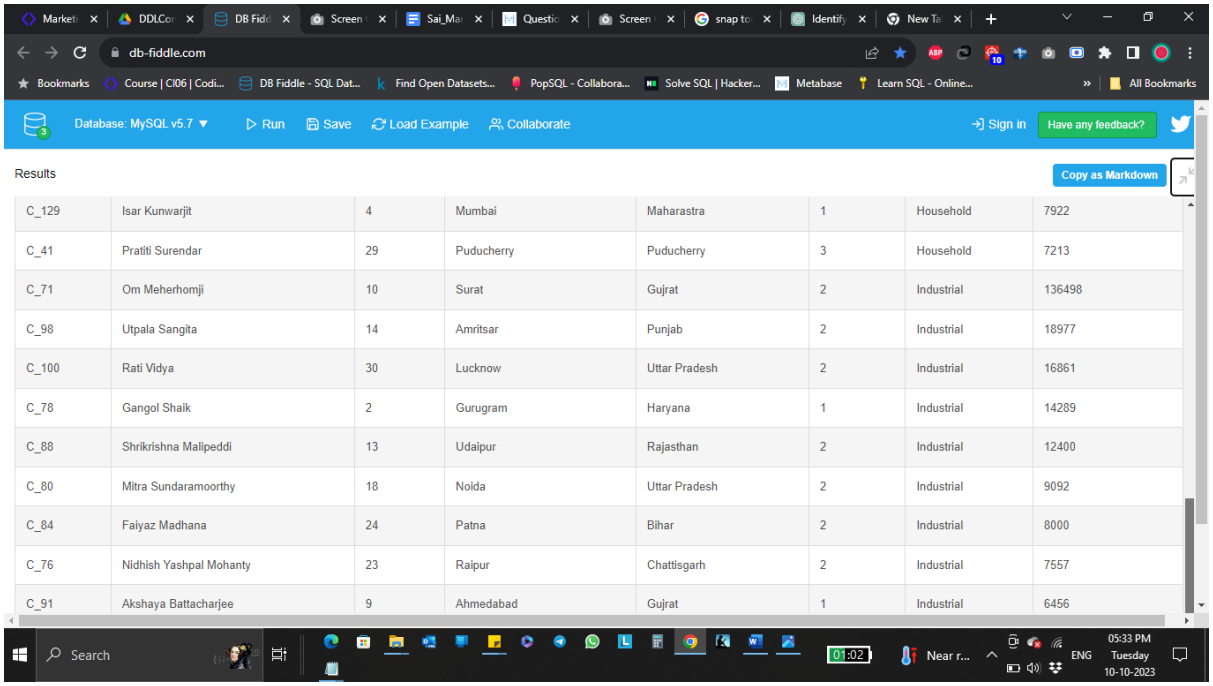
```
DELIMITER $$
CREATE FUNCTION Discount
(Quantity int, Price float, PurchasingAmt float)
RETURNS INT
DETERMINISTIC
BEGIN
   DECLARE discount INT;
   SET discount = Quantity * Price - PurchasingAmt;
   RETURN discount;
END$$
DELIMITER ;

(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:

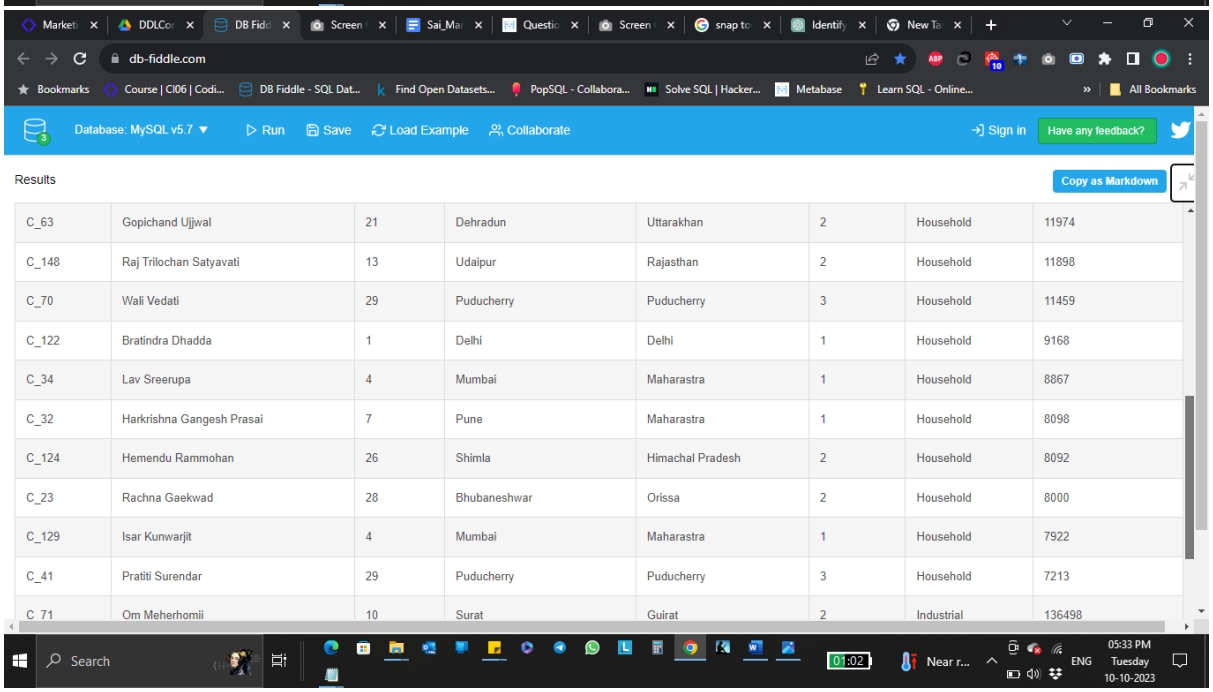


Cust_Id	Name	City_Id	City_Name	State	CityTier	OrderType	total_discount
C_120	Poornachandra Sameer	17	Indore	Madhya Pradesh	2	Government	22144
C_114	Nityapriya Venktesh	7	Pune	Maharastra	1	Government	13396
C_112	Ekata Mathrubootham	26	Shimla	Himachal Pradesh	2	Government	10892
C_101	Bhagya Chirimar	11	Jaipur	Rajasthan	2	Government	9658
C_106	Komal Vinata	9	Ahmedabad	Gujrat	1	Government	8753
C_105	Indukala Ramana	28	Bhubaneshwar	Orissa	2	Government	8000
C_115	Akuti Lokhande	9	Ahmedabad	Gujrat	1	Government	7271
C_116	Swasti Satyen	2	Gurugram	Haryana	1	Government	6318
C_107	Bimala Pundarik	25	Thiruvananthapuram	Kerala	2	Government	6000
C_119	Ashakiran Sarath	30	Lucknow	Uttar Pradesh	2	Government	4000



Results

C_129	Isar Kunwarjit	4	Mumbai	Maharastra	1	Household	7922
C_41	Pratiti Surendar	29	Puducherry	Puducherry	3	Household	7213
C_71	Om Meherhomji	10	Surat	Gujrat	2	Industrial	136498
C_98	Utpala Sangita	14	Amritsar	Punjab	2	Industrial	18977
C_100	Rati Vidya	30	Lucknow	Uttar Pradesh	2	Industrial	16861
C_78	Gangol Shaik	2	Gurugram	Haryana	1	Industrial	14289
C_88	Shrikrishna Malipeddi	13	Udaipur	Rajasthan	2	Industrial	12400
C_80	Mitra Sundaramoorthy	18	Noida	Uttar Pradesh	2	Industrial	9092
C_84	Faiyaz Madhana	24	Patna	Bihar	2	Industrial	8000
C_76	Nidhish Yashpal Mohanty	23	Raipur	Chattisgarh	2	Industrial	7557
C_91	Akshaya Battacharjee	9	Ahmedabad	Gujrat	1	Industrial	6456



Results

C_63	Gopichand Ujjwal	21	Dehradun	Uttarakhan	2	Household	11974
C_148	Raj Trilochan Satyavati	13	Udaipur	Rajasthan	2	Household	11898
C_70	Wali Vedati	29	Puducherry	Puducherry	3	Household	11459
C_122	Bratindra Dhadda	1	Delhi	Delhi	1	Household	9168
C_34	Lav Sreerupa	4	Mumbai	Maharastra	1	Household	8867
C_32	Harkrishna Gangesh Prasai	7	Pune	Maharastra	1	Household	8098
C_124	Hemendu Rammohan	26	Shimla	Himachal Pradesh	2	Household	8092
C_23	Rachna Gaekwad	28	Bhubaneshwar	Orissa	2	Household	8000
C_129	Isar Kunwarjit	4	Mumbai	Maharastra	1	Household	7922
C_41	Pratiti Surendar	29	Puducherry	Puducherry	3	Household	7213
C_71	Om Meherhomji	10	Surat	Gujrat	2	Industrial	136498

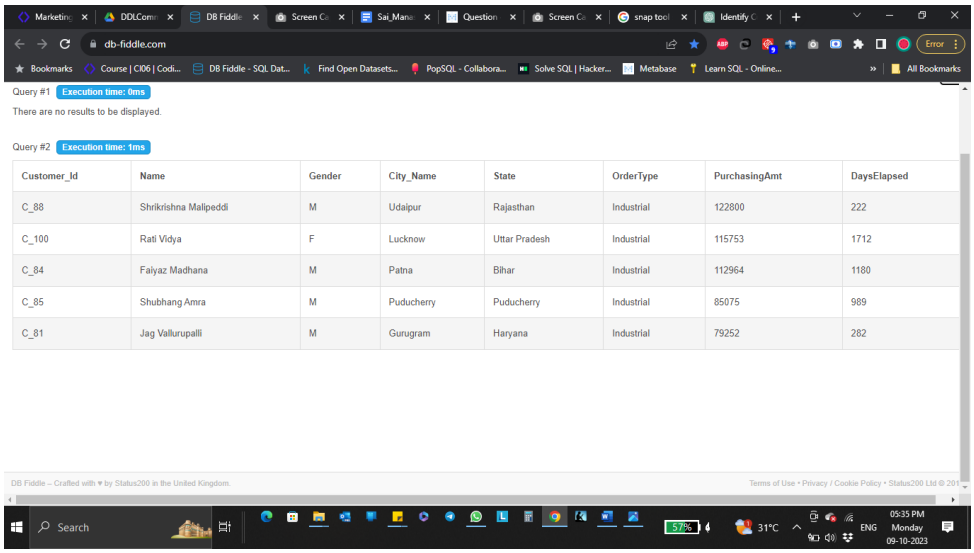
2b. Identify the top 5 customers (from household and industrial sector) based on purchase amount and days elapsed in descending order. Do highlight if you think there is a data error.

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:



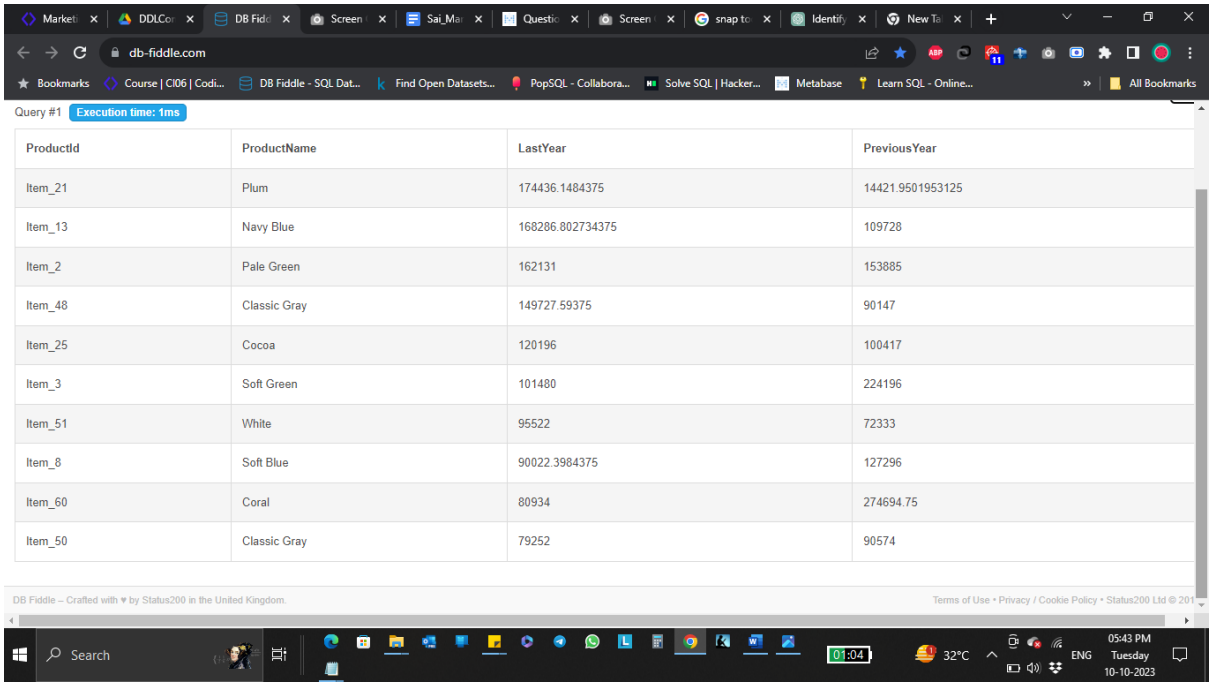
Customer_Id	Name	Gender	City_Name	State	OrderType	PurchasingAmt	DaysElapsed
C_88	Shrikrishna Malipeddi	M	Udaipur	Rajasthan	Industrial	122800	222
C_100	Rati Vidya	F	Lucknow	Uttar Pradesh	Industrial	115753	1712
C_84	Faiyaz Madhana	M	Patna	Bihar	Industrial	112964	1180
C_85	Shubhang Amra	M	Puducherry	Puducherry	Industrial	85075	989
C_81	Jag Vallurupalli	M	Gurugram	Haryana	Industrial	79252	282

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:



Query #1 Execution time: 1ms

ProductId	ProductName	LastYear	PreviousYear
Item_21	Plum	174436.1484375	14421.9501953125
Item_13	Navy Blue	168286.802734375	109728
Item_2	Pale Green	162131	153885
Item_48	Classic Gray	149727.59375	90147
Item_25	Cocoa	120196	100417
Item_3	Soft Green	101480	224196
Item_51	White	95522	72333
Item_8	Soft Blue	90022.3984375	127296
Item_60	Coral	80934	274694.75
Item_50	Classic Gray	79252	90574

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:

Customer_Id	Name	PurchasingAmt	Item_Id	Class_Category
C_103	Hiral Shamsher	214755	Item_49	Low_Class
C_103	Hiral Shamsher	167980	Item_60	Low_Class
C_104	Najma Bhaskar	286650	Item_22	Middle_Class
C_112	Ekata Mathrubootham	149728	Item_48	Middle_Class
C_88	Shrikrishna Malipeddi	122800	Item_23	High_Class
C_114	Nityapriya Venktesh	105435	Item_25	High_Class

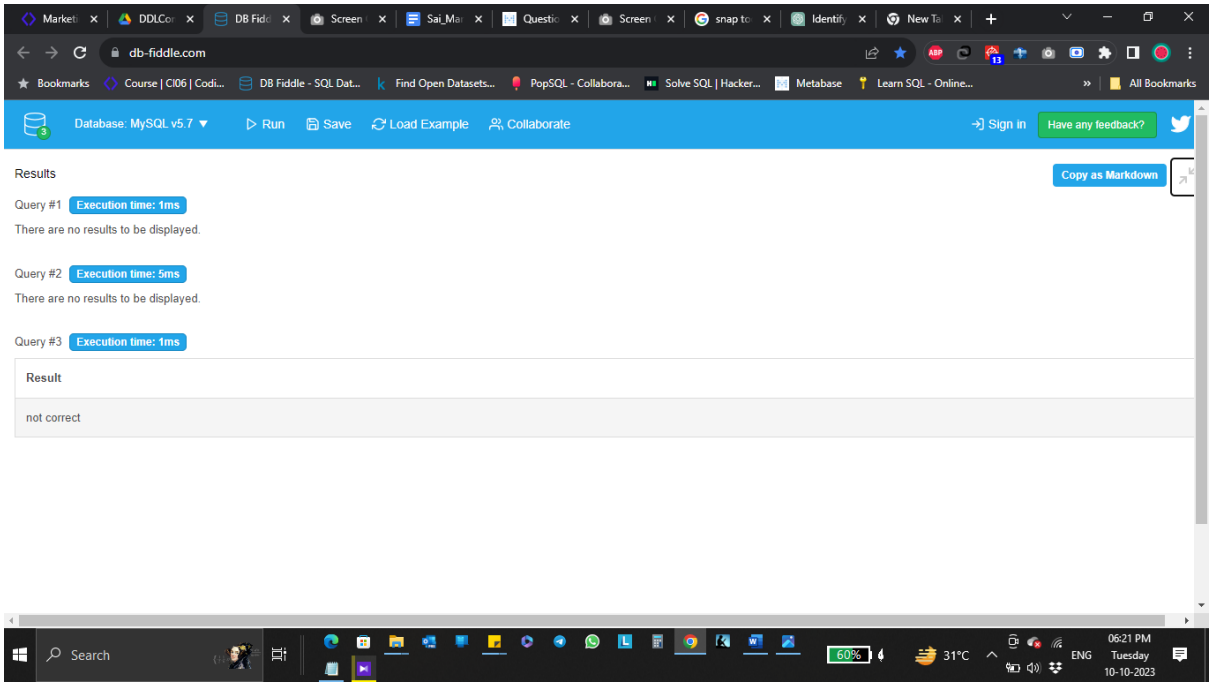
3a. Create Stored Procedures for following data validation tasks:
Identify whether a particular transaction amount (purchase amount) is 'correct' or 'not correct'.
It is correct if price and quantity are used to calculate without a coupon. In case of a coupon, the coupon amount should be deducted from 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:

Query #1 **Execution time: 1ms**

There are no results to be displayed.

Query #2 **Execution time: 1ms**

Count_Age_Below_12
0