**Data Analytics Tasks:**

**A. Basic Aggregation:**

1. Find the total number of purchases per product per year and list down the top selling products every year

**Use sql**

|  |
| --- |
| a.product\_id, |
| a.product\_name, |
| a.year1, |
| a.rank1 |
| FROM |
| (SELECT |
| YEAR(ph.purchase\_date) AS year1, |
| ph.product\_id, |
| p.product\_name, |
| COUNT(ph.quantity) AS total\_purchases, |
| DENSE\_RANK() OVER (PARTITION BY YEAR(ph.purchase\_date) ORDER BY COUNT(ph.quantity) DESC) AS rank1 |
| FROM |
| purchase\_history\_dataset ph |
| JOIN |
| products\_dataset p |
| ON |
| ph.product\_id = p.product\_id |
| GROUP BY |
| ph.product\_id, YEAR(ph.purchase\_date),p.product\_name |
| ORDER BY |
| year1 DESC, total\_purchases DESC) AS a |
| WHERE |
| a.rank1 = 1;  Ans:   |  |  |  |  | | --- | --- | --- | --- | | product\_id | product\_name | year1 | rank1 | | 36 | Milk | 2023 | 1 | | 274 | Butter | 2022 | 1 | | 150 | Butter | 2021 | 1 | | 5 | Rice | 2021 | 1 | | 426 | Pasta | 2020 | 1 | | 208 | Rice | 2019 | 1 | | 105 | Pasta | 2018 | 1 | | 91 | Cheese | 2018 | 1 | | 399 | Pasta | 2017 | 1 | | 405 | Bread | 2016 | 1 | | 96 | Rice | 2016 | 1 | | 392 | Apple | 2016 | 1 | | 325 | Banana | 2015 | 1 | | 137 | Bread | 2015 | 1 | | 62 | Pasta | 2015 | 1 | |

2.Calculate the average quantity purchased per product and list down the top 5 products with high avg quantity purchased

|  |
| --- |
| select  pd.product\_id,pd.product\_name, |
| AVG(ph.quantity) AS avg\_quantity\_purchased |
| FROM purchase\_history\_dataset ph join products\_dataset pd |
| on pd.product\_id = ph.product\_id |
| GROUP BY |
| pd.product\_id,pd.product\_name |
| ORDER BY |
| avg\_quantity\_purchased DESC |
| LIMIT 5;  Ans:   |  |  |  | | --- | --- | --- | | PRODUCT\_ID | PRODUCTS | AVG QTY | | 101 | Banana | 3.8421 | | 432 | Banana | 3.8235 | | 182 | Banana | 3.8182 | | 46 | Chicken | 3.7692 | | 43 | Milk | 3.7368 |   **B. Join Operations:**  1.Join purchase history with products dataset to get the product name for each purchase.   |  | | --- | | Select  pd.Product\_name,ph.purchase\_id, ph.product\_id  from purchase\_history\_dataset ph inner join products\_dataset pd | | on ph.product\_id = pd.product\_id;  Ans:   |  |  |  | | --- | --- | --- | | product name | purchase id | product id | | Milk | 1 | 42 | | Bread | 2 | 138 | | Bread | 3 | 403 | | Bread | 4 | 193 | | Bread | 5 | 26 | | Pasta | 6 | 357 | | Butter | 7 | 192 | | Cheese | 8 | 225 | | Eggs | 9 | 218 | | Chicken | 10 | 176 |   2.Join purchase history with customer profile dataset to include customer information for each purchase and  list top 5 customers with high purchases   |  | | --- | | select | | concat(cp.first\_name, ' ' ,cp.last\_name),gender,date\_of\_birth,email,phone\_number, | | signup\_date,address,city,state,zip\_code, | | sum(ph.total\_amount) as `top 5 high purchases` | | From  purchase\_history\_dataset ph inner join customer\_profile\_dataset cp | | on ph.customer\_id = cp.customer\_id  group by | | concat(cp.first\_name, ' ' ,cp.last\_name),gender,date\_of\_birth,email,phone\_number, | | signup\_date,address,city,state,zip\_code  order by `top 5 high purchases`desc limit 5;  Ans:   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | name | customer\_id | gender | | date\_of\_birth | | email | | | | | Alex Smith | 460 | Female | | 8/13/1995 9:31 | | jane.smith460@mail.com | | | | | Emily Brown | 573 | Female | | 4/12/1988 20:37 | | jessica.brown573@mail.com | | | | | Robert Jones | 403 | Male | | 8/25/1988 15:46 | | william.garcia403@mail.com | | | | | Michael Miller | 236 | Female | | 10/12/1965 22:20 | | jessica.jones236@mail.com | | | | | Robert Johnson | 192 | Male | | 1/11/1961 20:05 | | jane.jones192@mail.com | | | | | phone\_number | signup\_date | | address | | city | | state | zip\_code | top 5 high purchases | | | 220-657-3912 | 6/14/2015 2:14 | | 1685 Main St | | Chicago | | PA | 89184 | 2101.158435 | | | 320-765-2162 | 10/7/2021 13:34 | | 8461 Main St | | Los Angeles | | CA | 54244 | 2095.176859 | | | 989-345-4993 | 4/1/2020 22:19 | | 2479 Oak St | | Phoenix | | TX | 19425 | 2082.92056 | | | 722-896-3156 | 11/18/2017 22:33 | | 7301 Maple St | | Phoenix | | TX | 78590 | 2062.323178 | | | 278-361-3063 | 9/18/2015 4:03 | | 234 Maple St | | Philadelphia | | IL | 29474 | 2044.132281 | |   **C. Window Functions:**  1.Find the cumulative sum of purchases for each product category over year.   |  | | --- | | Select  ph.purchase\_id,pd.category,pd.product\_name,ph.total\_amount, | | YEAR(ph.purchase\_date) AS year\_only, | | ROUND(SUM(ph.total\_amount) OVER (PARTITION BY pd.category ORDER BY YEAR(ph.purchase\_date) ROWS  BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW), 2) AS commulative\_sum | | FROM purchase\_history\_dataset ph | | INNER JOIN products\_dataset pd | | ON ph.product\_id = pd.product\_id | | ORDER BY pd.category, year\_only;  Ans:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | purchase\_id | category | product\_name | total\_amount | year | cumulative\_sum | | 9750 | Dairy | Eggs | 44.97020243 | 2015 | 44.97 | | 9554 | Dairy | Pasta | 2.293200406 | 2015 | 47.26 | | 7513 | Dairy | Pasta | 193.0476688 | 2015 | 240.31 | | 7447 | Dairy | Banana | 164.1462534 | 2015 | 404.46 | | 9866 | Dairy | Milk | 42.03150966 | 2015 | 446.49 |   **D. Rank and Dense Rank:**  Rank customers based on their total expenditure   |  | | --- | | SELECT Customer\_ID, | | SUM(total\_amount) AS Total\_Expenditure, | | RANK() OVER (ORDER BY SUM(total\_amount) DESC) AS Rankcustomer | | FROM purchase\_history\_dataset | | GROUP BY Customer\_ID | | ORDER BY Rankcustomer;  Ans:   |  |  |  | | --- | --- | --- | | Customer\_ID | Total\_Expenditure | Rankcustomer | | 460 | 2101.158435 | 1 | | 573 | 2095.176859 | 2 | | 403 | 2082.92056 | 3 | | 236 | 2062.323178 | 4 | | 192 | 2044.132281 | 5 | | 199 | 2033.240175 | 6 | | 602 | 2003.596025 | 7 | | 705 | 1931.289448 | 8 | | 830 | 1892.015874 | 9 | | 748 | 1857.453403 | 10 |   2.Identify the top 10 customers by purchase frequency using DENSE\_RANK().   |  |  |  |  | | --- | --- | --- | --- | | dense rank |  |  |  | | select \* from (SELECT Customer\_ID, | |  |  | | sum(total\_amount) AS purchase\_frequency, | |  |  | | dense\_rank() OVER (order by sum(total\_amount) DESC) AS DenseRankcustomer | | | | | FROM purchase\_history\_dataset | |  |  | | GROUP BY Customer\_ID |  |  |  | | ORDER BY denseRankcustomer )as a where DenseRankcustomer <=10;  Ans:   |  |  |  | | --- | --- | --- | | Customer\_ID | purchase\_frequency | DenseRankcustomer | | 460 | 2101.158435 | 1 | | 573 | 2095.176859 | 2 | | 403 | 2082.92056 | 3 | | 236 | 2062.323178 | 4 | | 192 | 2044.132281 | 5 | | 199 | 2033.240175 | 6 | | 602 | 2003.596025 | 7 | | 705 | 1931.289448 | 8 | | 830 | 1892.015874 | 9 | | 748 | 1857.453403 | 10 | | | | |   **E. Percentiles:**  1.Calculate the 25th, 50th (median), and 75th percentiles of total purchase amounts for each customer  segment.   |  | | --- | | Select | | customer\_id, | | MAX(CASE WHEN percentile\_rank <= 25 THEN total\_purchase END) AS p25, | | MAX(CASE WHEN percentile\_rank <= 50 THEN total\_purchase END) AS p50, | | MAX(CASE WHEN percentile\_rank <= 75 THEN total\_purchase END) AS p75 | | FROM ( | | SELECT | | customer\_id, | | SUM(total\_amount) AS total\_purchase, | | NTILE(100) OVER (PARTITION BY customer\_id ORDER BY SUM(total\_amount)) AS percentile\_rank | | FROM purchase\_history\_dataset | | GROUP BY customer\_id | | ) AS ranked\_totals | | GROUP BY customer\_id;  Ans:   |  |  |  |  | | --- | --- | --- | --- | | 1 | 727.9147 | 727.9147 | 727.9147 | | 2 | 1039.335 | 1039.335 | 1039.335 | | 3 | 1392.96 | 1392.96 | 1392.96 | | 4 | 1272.246 | 1272.246 | 1272.246 | | 5 | 627.4875 | 627.4875 | 627.4875 |   **F. Median Calculation:**  1.Compute the median purchase amount per product category   |  | | --- | | WITH ranked\_purchases AS ( | | SELECT | | p.category, | | ph.total\_amount, | | ROW\_NUMBER() OVER (PARTITION BY p.category ORDER BY ph.total\_amount) AS row\_num, | | COUNT(\*) OVER (PARTITION BY p.category) AS total\_count | | FROM purchase\_history\_dataset ph | | JOIN products\_dataset p | | ON ph.product\_id = p.product\_id | | ) | | SELECT | | category, | | AVG(total\_amount) AS median\_purchase | | FROM ranked\_purchases | | WHERE row\_num IN ( | | FLOOR((total\_count + 1) / 2), | | FLOOR((total\_count + 2) / 2) | | ) | | GROUP BY category;  Ans:   |  |  | | --- | --- | | category | median\_purchase | | Dairy | 64.52905017 | | Fruits | 63.10525385 | | Grains | 60.52809418 | | Meats | 62.59683978 | |   **G. Complex Aggregation:**  1.Find the average, maximum, and minimum purchase value for each product type and customer age  group.   |  |  |  |  |  | | --- | --- | --- | --- | --- | | select category, | | | | | | AVG(ph.total\_amount) AS AveragePurchaseValue, | | | | | | MAX(ph.total\_amount) AS MaxPurchaseValue, | | | | | | MIN(ph.total\_amount) AS MinPurchaseValue | | | | | | FROM | | | | | | purchase\_history\_dataset ph join products\_dataset pd on ph.product\_id=pd.product\_id | | | | | | group by category;  Ans:   |  |  |  |  | | --- | --- | --- | --- | | category | AveragePurchaseValue | MaxPurchaseValue | MinPurchaseValue | | Fruits | 77.3866187 | 248.9685299 | 1.5266482 | | Grains | 76.28866119 | 249.4809101 | 1.556513223 | | Meats | 77.77868969 | 249.9635128 | 1.664455959 | | Dairy | 78.386581 | 249.862798 | 1.546734835 | | | | | | | 2.Customer age group | | | | | | SELECT category, | | | | | | CASE | | | | | | WHEN TIMESTAMPDIFF(YEAR, cp.date\_of\_birth, CURDATE()) < 20 THEN 'Under 20' | | | | | | WHEN TIMESTAMPDIFF(YEAR, cp.date\_of\_birth, CURDATE()) BETWEEN 20 AND 40 THEN '20 to 40' | | | | | | ELSE 'Above 40' | | | | | | END AS AgeGroup, | | | | | | AVG(ph.total\_amount) AS AveragePurchaseValue, | | | | | | MAX(ph.total\_amount) AS MaxPurchaseValue, | | | | | | MIN(ph.total\_amount) AS MinPurchaseValue | | | | | | FROM | | | | | | purchase\_history\_dataset ph | | | | | | LEFT JOIN customer\_profile\_dataset cp | | | | | | ON ph.customer\_id = cp.customer\_id | | | | | | INNER JOIN products\_dataset pd | | | | | | ON ph.product\_id = pd.product\_id | | | | | | GROUP BY category, | | | | | | CASE | | | | | | WHEN TIMESTAMPDIFF(YEAR, cp.date\_of\_birth, CURDATE()) < 20 THEN 'Under 20' | | | | | | WHEN TIMESTAMPDIFF(YEAR, cp.date\_of\_birth, CURDATE()) BETWEEN 20 AND 40 THEN '20 to 40' | | | | | | ELSE 'Above 40' | | | | | | END;  Ans: | | | | | | category | AgeGroup | AveragePurchaseValue | MaxPurchaseValue | MinPurchaseValue | | Fruits | 20 to 40 | 77.25740042 | 248.9685299 | 1.5266482 | | Grains | 20 to 40 | 76.15446409 | 248.5900399 | 1.920027886 | | Meats | 20 to 40 | 79.6495012 | 249.9635128 | 2.05013924 | | Dairy | 20 to 40 | 78.96594412 | 249.2274932 | 1.716753841 | | Fruits | Above 40 | 77.47256388 | 248.7455991 | 1.744938211 | | Dairy | Above 40 | 77.96434964 | 249.862798 | 1.546734835 | | Meats | Above 40 | 76.56216902 | 249.3645111 | 1.664455959 | | Grains | Above 40 | 76.3835789 | 249.4809101 | 1.556513223 |   **H. Grouping:**  1.Group purchases by day of the week and find the average number of purchases made on each day.   |  |  |  | | --- | --- | --- | | SELECT | | | | DAYOFWEEK(purchase\_date) AS day\_of\_week, | | | | AVG(daily\_total\_amount) AS avg\_purchases\_per\_day | | | | FROM ( | | | | SELECT | | | | purchase\_date, | | | | SUM(ph.total\_amount) AS daily\_total\_amount | | | | FROM | | | | purchase\_history\_dataset ph | | | | GROUP BY | | | | purchase\_date | | | | ) AS daily\_counts | | | | GROUP BY | | | | DAYOFWEEK(purchase\_date) | | | | ORDER BY | | | | day\_of\_week;  Ans: | | | | day\_of\_week | avg\_purchases\_per\_day | | 1 | 78.39979403 | | 2 | 76.29749145 | | 3 | 79.07887053 | | 4 | 78.17165367 | | 5 | 77.04593826 | | 6 | 76.96357485 | | 7 | 76.02708248 |   2.Group customers by city and find the total number of purchases and total revenue generated.  Select  c.city,  COUNT(ph.purchase\_id) AS total\_purchases,  SUM(ph.total\_amount) AS total\_revenue  FROM purchase\_history\_dataset ph  JOIN customer\_profile\_dataset c ON ph.customer\_id = c.customer\_id  JOIN products\_dataset pr ON ph.product\_id = pr.product\_id  GROUP BY c.city;  Ans:   |  |  |  | | --- | --- | --- | | city | total\_purchases | total\_revenue | | San Antonio | 1292 | 98815.29235 | | New York | 1497 | 118338.7228 | | Chicago | 1697 | 132182.1817 | | Los Angeles | 1283 | 101588.5172 | | Phoenix | 1557 | 116973.3683 | | Houston | 1548 | 116424.898 | | Philadelphia | 1434 | 113761.9771 |   **I. Case Statement:**  1.Classify customers as “High-Spending,” “Medium-Spending,” or “Low-Spending” based on their total  purchase amounts (using percentiles in multiples of 33).   |  |  |  |  |  | | --- | --- | --- | --- | --- | | SELECT | | | | | | c.customer\_id, | | | | | | concat(c.first\_name, ' ' , c.last\_name), | | | | | | SUM(ph.total\_amount) AS total\_spent, | | | | | | CASE | | | | | | WHEN NTILE(3) OVER (ORDER BY SUM(ph.total\_amount)) = 1 THEN 'Low-Spending' | | | | | | WHEN NTILE(3) OVER (ORDER BY SUM(ph.total\_amount)) = 2 THEN 'Medium-Spending' | | | | | | WHEN NTILE(3) OVER (ORDER BY SUM(ph.total\_amount)) = 3 THEN 'High-Spending' | | | | | | END AS spending\_classification | | | | | | FROM | | | | | | customer\_profile\_dataset c | | | | | | JOIN | | | | | | purchase\_history\_dataset ph ON c.customer\_id = ph.customer\_id | | | | | | GROUP BY | | | | | | c.customer\_id, c.first\_name,c.last\_name | | | | | | ORDER BY | | | | | | total\_spent DESC;  Ans: | | | | | | customer\_id | name | total\_spent | spending\_classification | | 460 | Alex Smith | 2101.158435 | High-Spending | | 573 | Emily Brown | 2095.176859 | High-Spending | | 403 | Robert Jones | 2082.92056 | High-Spending | | 236 | Michael Miller | 2062.323178 | High-Spending | | 192 | Robert Johnson | 2044.132281 | High-Spending | | 199 | Jane Smith | 2033.240175 | High-Spending | | 602 | Sarah Garcia | 2003.596025 | High-Spending | | 705 | Jane Brown | 1931.289448 | High-Spending |   **J. Join with Condition:**  1.Join purchase history with customer profile dataset where the customer’s age is above 30, and display  their purchase details.   |  | | --- | | SELECT | | ph.customer\_id, | | ph.total\_amount, | | ph.purchase\_date, | | ph.product\_id, | | ph.purchase\_id, | | ph.quantity, | | concat(cp.first\_name, ' ' , cp.last\_name), | | cp.date\_of\_birth, | | TIMESTAMPDIFF(YEAR, cp.date\_of\_birth, CURDATE()) AS age\_above\_30\_purch\_details | | FROM | | purchase\_history\_dataset ph | | JOIN | | customer\_profile\_dataset cp | | ON ph.customer\_id = cp.customer\_id | | WHERE | | TIMESTAMPDIFF(YEAR, cp.date\_of\_birth, CURDATE()) > 30 | | ORDER BY | | ph.purchase\_date; |   Ans:   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | customer\_id | | total\_amount | purchase\_date | product\_id | | purchase\_id | | | 847 | | 19.62920669 | 1/23/2015 5:08 | 283 | | 8624 | | | 886 | | 41.32591695 | 2/1/2015 17:17 | 411 | | 8993 | | | 701 | | 33.28022252 | 2/23/2015 8:26 | 164 | | 7130 | | | 989 | | 81.20679885 | 3/6/2015 13:59 | 430 | | 10187 | | | 886 | | 63.27535988 | 3/16/2015 12:59 | 356 | | 8994 | | | quantity | concat(cp.first\_name, ' ' , cp.last\_name) | | | | date\_of\_birth | | age\_above\_30\_purch\_details | | | 3 | Michael Rodriguez | | | | 7/27/1977 9:07 | | 47 | | | 1 | Michael Rodriguez | | | | 1/21/1962 6:59 | | 62 | | | 3 | Robert Johnson | | | | 11/10/1992 0:29 | | 32 | | | 2 | Emily Williams | | | | 5/8/1978 10:55 | | 46 | | | 4 | Michael Rodriguez | | | | 1/21/1962 6:59 | | 62 | |   **K. Top N Analysis:**   1. Find the top 5 products contributing to the highest revenue.  |  |  |  |  | | --- | --- | --- | --- | | SELECT | | | | | pd.product\_id, | | | | | pd.product\_name, | | | | | SUM(ph.quantity \* pd.price\_per\_unit) AS total\_revenue | | | | | FROM | | | | | purchase\_history\_dataset ph | | | | | JOIN | | | | | products\_dataset pd | | | | | ON ph.product\_id = pd.product\_id | | | | | GROUP BY | | | | | pd.product\_id, pd.product\_name | | | | | ORDER BY | | | | | total\_revenue DESC | | | | | LIMIT 5;  Ans: | | | | | product\_id | product\_name | total\_revenue | | 193 | Bread | 3263.7 | | 274 | Butter | 2703.88 | | 443 | Eggs | 2627.28 | | 116 | Butter | 2591.82 | | 298 | Eggs | 2487.98 |   2.Identify the top 3 cities with the most number of unique customers   |  |  |  |  | | --- | --- | --- | --- | | SELECT | | | | | city, | | | | | COUNT(DISTINCT cp.customer\_id) AS unique\_customers | | | | | FROM | | | | | customer\_profile\_dataset cp | | | | | GROUP BY | | | | | city | | | | | ORDER BY | | | | | unique\_customers DESC | | | | | LIMIT 3;  Ans: | | | | | city | unique\_customers | | | Chicago | 160 | | | Houston | 151 | | | New York | 151 | | | **L. Window Functions for Trend Analysis:**   1. Create a 7-day moving average of total purchases per product and which customer had high 7 day MA   WITH daily\_totals AS ( | | | | | | | | -- Step 1: Calculate total purchase per product per day | | | | | | | | SELECT | | | | | | | | ph.purchase\_date, | | | | | | | | ph.product\_id, | | | | | | | | ph.customer\_id, | | | | | | | | SUM(ph.total\_amount) AS total\_purchase\_per\_day | | | | | | | | FROM | | | | | | | | purchase\_history\_dataset ph | | | | | | | | GROUP BY | | | | | | | | ph.purchase\_date, ph.product\_id, ph.customer\_id | | | | | | | | ), | | | | | | | | moving\_avg AS ( | | | | | | | | -- Step 2: Calculate the 7-day moving average for each product and customer | | | | | | | | SELECT | | | | | | | | dt.purchase\_date, | | | | | | | | dt.product\_id, | | | | | | | | dt.customer\_id, | | | | | | | | dt.total\_purchase\_per\_day, | | | | | | | | AVG(dt.total\_purchase\_per\_day) OVER ( | | | | | | | | PARTITION BY dt.product\_id, dt.customer\_id | | | | | | | | ORDER BY dt.purchase\_date | | | | | | | | ROWS BETWEEN 6 PRECEDING AND CURRENT ROW | | | | | | | | ) AS moving\_avg\_7\_days | | | | | | | | FROM | | | | | | | | daily\_totals dt | | | | | | | | ) | | | | | | | | -- Step 3: Find the customer with the highest 7-day moving average for each product and enrich with customer  profile information | | | | | | | | SELECT | | | | | | | | ma.product\_id, | | | | | | | | ma.customer\_id, | | | | | | | | CONCAT(cp.first\_name, ' ',cp.last\_name), -- Assuming 'customer\_name' is the column in the  'customer\_profile\_dataset' | | | | | | | | MAX(ma.moving\_avg\_7\_days) AS highest\_7\_day\_moving\_avg | | | | | | | | FROM | | | | | | | | moving\_avg ma | | | | | | | | JOIN | | | | | | | | customer\_profile\_dataset cp ON ma.customer\_id = cp.customer\_id -- Joining the customer profile dataset  on customer\_id | | | | | | | | GROUP BY | | | | | | | | ma.product\_id, ma.customer\_id, CP.first\_name, CP.last\_name | | | | | | | | ORDER BY | | | | | | | | ma.product\_id, highest\_7\_day\_moving\_avg DESC;  Ans: | | | | | | | | product\_id | | customer\_id | CONCAT(cp.first\_name, ' ',cp.last\_name) | | highest\_7\_day\_moving\_avg | | 1 | | 656 | Michael Johnson | | 192.6416687 | | 1 | | 838 | Robert Jones | | 166.0823841 | | 1 | | 892 | Emily Miller | | 152.9514387 | | 1 | | 602 | Sarah Garcia | | 110.7370899 | | 1 | | 685 | Linda Smith | | 107.3675444 |   **M. Nested Queries:**   1. Find customers who have never purchased products from the top 5 most popular categories.  |  | | --- | | top 5 category not found data list only 4 category | | WITH top\_categories AS ( | | -- Step 1: Identify the top 5 most popular categories based on purchase count | | SELECT | | p.category, | | COUNT(ph.purchase\_id) AS purchase\_count | | FROM | | purchase\_history\_dataset ph | | JOIN | | products\_dataset p ON ph.product\_id = p.product\_id | | GROUP BY | | p.category | | ORDER BY | | purchase\_count DESC | | LIMIT 1 | | ) | | -- Step 2: Find customers who have never purchased from the top 1 categories | | SELECT | | cp.customer\_id, | | cp.first\_name | | FROM | | customer\_profile\_dataset cp | | WHERE | | NOT EXISTS ( | | -- Subquery to check if the customer has purchased from the top 2 categories | | SELECT 1 | | FROM purchase\_history\_dataset ph | | JOIN products\_dataset p ON ph.product\_id = p.product\_id | | WHERE | | ph.customer\_id = cp.customer\_id | | AND p.category IN (SELECT category FROM top\_categories) | | ) | | ORDER BY | | cp.customer\_id; |   Ans:   |  |  | | --- | --- | | 1 | never purchase found | | 2 | never purchase found | | 3 | never purchase found | | 4 | never purchase not found | | 5 | category not vailable |   2.Identify products purchased only once in the entire dataset.  Ans:No  **N. Date Analysis:**   1. Identify the month with the highest total sales volume.  |  |  |  | | --- | --- | --- | | SELECT | | | | EXTRACT(MONTH FROM purchase\_date) AS sale\_month, | | | | SUM(total\_amount) AS total\_sales\_volume | | | | FROM | | | | purchase\_history\_dataset | | | | GROUP BY | | | | EXTRACT(MONTH FROM purchase\_date) | | | | ORDER BY | | | | total\_sales\_volume DESC  Ans: | | | | sale\_month | high total\_sales\_volume | | 10 | 76338.88456 |   2.Calculate the year-over-year growth of total sales.   |  | | --- | | SELECT | | EXTRACT(YEAR FROM purchase\_date) AS sale\_year, | | SUM(total\_amount) AS total\_sales\_current\_year, | | LAG(SUM(total\_amount)) OVER (ORDER BY EXTRACT(YEAR FROM purchase\_date)) AS  total\_sales\_previous\_year, | | (SUM(total\_amount) - LAG(SUM(total\_amount)) OVER (ORDER BY EXTRACT(YEAR FROM purchase\_date))) / | | LAG(SUM(total\_amount)) OVER (ORDER BY EXTRACT(YEAR FROM purchase\_date)) \* 100 AS yoy\_growth\_percentage | | FROM | | purchase\_history\_dataset | | GROUP BY | | EXTRACT(YEAR FROM purchase\_date) | | ORDER BY | | sale\_year; |   Ans:   |  |  |  |  | | --- | --- | --- | --- | | sale\_year | total\_sales\_current\_year | total\_sales\_previous\_year | yoy\_growth\_percentage | | 2015 | 7636.187357 | NULL | NULL | | 2016 | 20761.45075 | 7636.187357 | 171.8824169 | | 2017 | 32783.69374 | 20761.45075 | 57.90656512 | | 2018 | 48696.42804 | 32783.69374 | 48.53856439 | | 2019 | 72876.8924 | 48696.42804 | 49.65551958 | | 2020 | 100255.4031 | 72876.8924 | 37.56816439 | | 2021 | 139867.1839 | 100255.4031 | 39.5108688 | | 2022 | 187221.3392 | 139867.1839 | 33.85651584 | | 2023 | 187986.3789 | 187221.3392 | 0.408628441 |   **O. Join with Aggregation:**   1. Join all three datasets to find the total revenue per product per customer.  |  | | --- | |  | | SELECT | | c.customer\_id, | | concat(c.first\_name, ' ', c.last\_name), | | p.product\_id, | | p.product\_name, | | SUM(ph.total\_amount) AS total\_revenue | | FROM | | customer\_profile\_dataset c | | JOIN | | purchase\_history\_dataset ph ON c.customer\_id = ph.customer\_id | | JOIN | | products\_dataset p ON ph.product\_id = p.product\_id | | GROUP BY | | c.customer\_id, | | c.first\_name, | | c.last\_name, | | p.product\_id, | | p.product\_name | | ORDER BY | | c.customer\_id, p.product\_id; |   Ans:   |  |  |  |  |  | | --- | --- | --- | --- | --- | | customer\_id | concat(c.first\_name, ' ', c.last\_name) | product\_id | product\_name | total\_revenue | | 1 | Robert Smith | 26 | Bread | 101.7788639 | | 1 | Robert Smith | 42 | Milk | 37.64207365 | | 1 | Robert Smith | 138 | Bread | 70.24710587 | | 1 | Robert Smith | 192 | Butter | 201.9883425 | | 1 | Robert Smith | 193 | Bread | 59.70505931 | | 1 | Robert Smith | 357 | Pasta | 167.3844015 | | 1 | Robert Smith | 403 | Bread | 89.16889586 | | 2 | Emily Garcia | 14 | Milk | 49.40745913 | | 2 | Emily Garcia | 55 | Apple | 42.61748689 |   **R. Subqueries:**   1. Identify customers whose total expenditure is above the average expenditure of all customers.  |  | | --- | | SELECT | | c.customer\_id, | | concat(c.first\_name, ' ' , | | c.last\_name), | | total\_purchase | | FROM ( | | SELECT | | ph.customer\_id, | | SUM(ph.total\_amount) AS total\_purchase | | FROM purchase\_history\_dataset ph | | GROUP BY ph.customer\_id | | ) AS total\_expenditure | | JOIN customer\_profile\_dataset c | | ON total\_expenditure.customer\_id = c.customer\_id | | WHERE total\_purchase > ( | | SELECT AVG(total\_amount) | | FROM purchase\_history\_dataset | | ); |   Ans:   |  |  |  | | --- | --- | --- | | customer\_id | name | total\_purchase | | 1 | Robert Smith | 727.9147426 | | 2 | Emily Garcia | 1039.334783 | | 3 | Jessica Brown | 1392.96016 | | 4 | Michael Brown | 1272.246149 | | 5 | Robert Jones | 627.4874904 |   2.Find products with a purchase amount higher than the average purchase amount across all products.   |  |  |  |  | | --- | --- | --- | --- | | SELECT p.product\_id, p.product\_name, SUM(ph.total\_amount) AS total\_purchase | | | | | FROM purchase\_history\_dataset ph | | | | | JOIN products\_dataset p | | | | | ON ph.product\_id = p.product\_id | | | | | GROUP BY p.product\_id, p.product\_name | | | | | HAVING total\_purchase > ( | | | | | SELECT AVG(total\_amount) | | | | | FROM purchase\_history\_dataset | | | | | );  Ans: | | | | | product\_id | product\_name | total\_purchase | | 42 | Milk | 1893.041628 | | 138 | Bread | 1332.837746 | | 403 | Bread | 1923.869817 | | 193 | Bread | 2520.797588 | | 26 | Bread | 2448.101133 |   **S. Correlated Subqueries:**   1. Find the purchase dates where a customer made a purchase larger than their average purchase   amount.   |  |  |  |  | | --- | --- | --- | --- | | SELECT ph.purchase\_date, concat(cp.first\_name, ' ' ,cp.last\_name), pr.product\_name | | | | | FROM purchase\_history\_dataset ph | | | | | JOIN customer\_profile\_dataset cp ON ph.customer\_id = cp.customer\_id | | | | | JOIN products\_dataset pr ON ph.product\_id = pr.product\_id | | | | | WHERE ph.total\_amount > ( | | | | | SELECT AVG(ph2.total\_amount) | | | | | FROM purchase\_history\_dataset ph2 | | | | | WHERE ph2.customer\_id = ph.customer\_id);  Ans: | | | | | purchase\_date | name | product\_name | | 2/4/2020 10:28 | Robert Smith | Pasta | | 9/21/2023 8:52 | Robert Smith | Butter | | 11/3/2023 8:21 | Emily Garcia | Eggs | | 10/22/2021 21:46 | Emily Garcia | Rice | | 3/20/2023 14:08 | Emily Garcia | Rice |   2.List all products that were purchased more frequently than the average frequency of products in their  category.   |  | | --- | | SELECT p.product\_id, p.product\_name, p.category | | FROM products\_dataset p | | JOIN purchase\_history\_dataset ph ON p.product\_id = ph.product\_id | | WHERE ( | | SELECT COUNT(\*) | | FROM purchase\_history\_dataset ph2 | | WHERE ph2.product\_id = p.product\_id | | ) > ( | | SELECT AVG(category\_purchase\_count) | | FROM ( | | SELECT p2.category, COUNT(\*) AS category\_purchase\_count | | FROM purchase\_history\_dataset ph2 | | JOIN products\_dataset p2 ON ph2.product\_id = p2.product\_id | | GROUP BY p2.category, p2.product\_id | | ) category\_avg | | WHERE category\_avg.category = p.category | | ) | | GROUP BY p.product\_id, p.product\_name, p.category |   Ans:   |  |  |  | | --- | --- | --- | | product\_id | product\_name | category | | 42 | Milk | Fruits | | 403 | Bread | Meats | | 193 | Bread | Grains | | 26 | Bread | Meats | | 357 | Pasta | Grains |   **T. Date Functions:**   1. Extract the day, month, and year from the purchase date, and group total purchases by month   across all years   |  |  |  |  | | --- | --- | --- | --- | | SELECT | | | | | MONTH(ph.purchase\_date) AS month, | | | | | YEAR(ph.purchase\_date) AS year, | | | | | SUM(ph.total\_amount) AS total\_purchases | | | | | FROM purchase\_history\_dataset ph | | | | | GROUP BY | | | | | YEAR(ph.purchase\_date), | | | | | MONTH(ph.purchase\_date) | | | | | ORDER BY | | | | | year, month | | | | | LIMIT 0, 1000;  Ans: | | | | | month | year | total\_purchases | | 1 | 2015 | 19.62920669 | | 2 | 2015 | 74.60613947 | | 3 | 2015 | 347.0150644 | | 4 | 2015 | 142.3131152 | | 5 | 2015 | 203.8237328 | | 6 | 2015 | 491.8604249 | | 7 | 2015 | 857.4687595 | | 8 | 2015 | 831.8061218 | | 9 | 2015 | 807.1426697 | | 10 | 2015 | 954.2470022 | | 11 | 2015 | 1622.308447 | | 12 | 2015 | 1283.966673 | | 1 | 2016 | 844.5590973 | | 2 | 2016 | 1072.80811 | | 3 | 2016 | 1810.991113 | | 4 | 2016 | 1259.271466 | | 5 | 2016 | 1279.933819 | | 6 | 2016 | 1783.258541 | | 7 | 2016 | 1371.804522 | | 8 | 2016 | 1626.242955 | | 9 | 2016 | 2280.125073 | | 10 | 2016 | 2628.136114 | | 11 | 2016 | 1716.19849 | | 12 | 2016 | 3088.121447 | | | | | | |
|  |