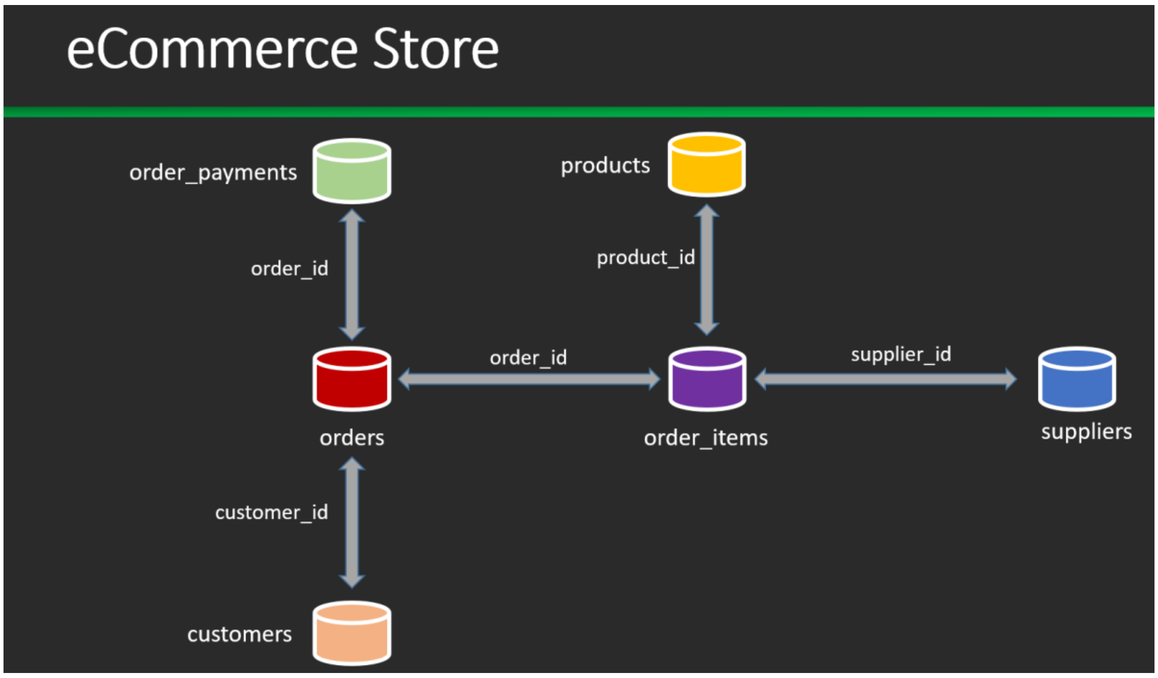
Level 2- Project Solution

Introduction

Welcome to our final project. In this project, we will use the eCommerce store datasets we already uploaded during the training. The following diagram will be helpful in quickly identifying the connections between the tables. To check your answers, please use the second file with the project solution.

Good luck!



Part 1 - Combining Data from Multiple Tables

Exercise #1

The list of all books (book title, year of publication) published between 1997 and 2005 and ordered in ascending by the year of publication (Tip – there are two separate tables for books: Book1 table is for books published until 1999 and the book2 table is for books published after 1999).

Answer:

```
SELECT book_title, year_of_publication

FROM books_schema.books1

WHERE year_of_publication >=1997

UNION

SELECT book_title, year_of_publication

FROM books_schema.books2

WHERE year_of_publication <=2005

ORDER BY year_of_publication ASC
```

Exercise #2

The list of publishers that published books in 1990 and 2004 (only publishers that published books in both years). (Tip – use the intersect operator).

Answer:

```
SELECT DISTINCT publisher

FROM books_schema.books1

WHERE year_of_publication = 1990

INTERSECT

SELECT DISTINCT publisher

FROM books_schema.books2

WHERE year_of_publication = 2004
```

Exercise #3

Answer:

The list of publishers that published books in 1990 but not in 2004. (Tip – use the except operator).

```
SELECT DISTINCT publisher
FROM books_schema.books1
WHERE year_of_publication = 1990
EXCEPT
SELECT DISTINCT publisher
FROM books_schema.books2
WHERE year_of_publication = 2004
```

Exercise #4

List all order items (order id, order item id, price) with the product information (product id, product category name), with a price higher than 3000, order by the order id and then order item id.

Answer:

```
SELECT oi.order_id, oi.order_item_id, oi.price, p.product_id, p.product_category_name
FROM ecommerce_schema.order_items AS oi
INNER JOIN ecommerce_schema.products AS p ON oi.product_id = p.product_id
WHERE price > 3000
ORDER BY 1,2
```

Exercise #5

List all orders (order id, order status) that order status is “shipped” with their order items (order item id, price) with the product information (product id, product category name) with a price between 50 and 60, order by the order id and then order item id. (tip – you will need more than one join).

Answer:

```
SELECT o.order_id, o.order_status, oi.order_item_id, oi.price, p.product_id,
p.product_category_name
FROM ecommerce_schema.orders as o
INNER JOIN ecommerce_schema.order_items as oi ON o.order_id = oi.order_id
INNER JOIN ecommerce_schema.products as p ON oi.product_id = p.product_id
WHERE
  order_status = 'shipped' AND
  price BETWEEN 50 AND 60
```

```
ORDER BY 1,2
```

Exercise #6

List of all products (product id, product category name) from the category “art” with their order items (order id, order item id, price) whether the product was ordered or not.

Answer:

```
SELECT p.product_id, p.product_category_name, oi.order_id, oi.order_item_id, oi.price

FROM ecommerce_schema.products AS p

LEFT JOIN ecommerce_schema.order_items AS oi ON p.product_id = oi.product_id

WHERE p.product_category_name = 'art'

ORDER BY order_id;
```

Part 2 - Subqueries

Exercise #7

All products from the "art" or "perfumery" category (product id, product category name, product price) that the product price is less than the average price of all products.

Answer:

```
SELECT product_id, product_category_name, product_price
FROM ecommerce_schema.products
WHERE product_category_name IN ('art', 'perfumery') AND
    product_price <
  (
        SELECT AVG(product_price)
        FROM ecommerce_schema.products
  );
```

Exercise #8

List all the names of customers that have orders with a "delivered" status with the products category "art" and price is more than 120 (Tip – use three sub-queries, break the problem into small steps, and check each sub-query before adding the outer query, start with the products table).

Answer:

```
SELECT customer_id, customer_name
FROM ecommerce_schema.customers
WHERE customer_id IN
(
  SELECT customer_id
  FROM ecommerce_schema.orders
  WHERE order_id IN
    (
      SELECT order_id
      FROM ecommerce_schema.order_items
      WHERE product_id IN
          (
                    SELECT product_id
                    FROM ecommerce_schema.products
                    WHERE product_category_name = 'art'
                  )
             AND price > 120
        )
     AND order_status ='delivered'
  )
```

Part 3 - Conditional Logic (CASE)

Exercise #9

List all orders reviews (order id, review score) and transform the review score (scale 1 to 5) into a wording scale ('Very Dissatisfied', 'Dissatisfied', 'Neutral', 'Satisfied', 'Very Satisfied')

Answer:

```
SELECT order_id,
        CASE review_score
                WHEN 1 THEN 'Very Dissatisfied'
                WHEN 2 THEN 'Dissatisfied'
                WHEN 3 THEN 'Neutral'
                WHEN 4 THEN 'Satisfied'
                WHEN 5 THEN 'Very Satisfied'
        END AS review_score
FROM ecommerce_schema.order_reviews;
```

Exercise #10

List all products (product id, product category, product price) and add a new column that will classify each product into a specific price category:

- Price<=50 → "Low Price"
- 50<Price<100 → "Medium Price"
- Price>100 → "High Price"

Answer:

```
SELECT product_id,product_category_name,product_price,
        CASE
                WHEN product_price<=50 THEN 'Low Price'
                WHEN product_price>50 AND product_price<100 THEN 'Medium Price'
                ELSE 'High Price'
        END AS price_category
FROM ecommerce_schema.products;
```

Exercise #11

Based on the classification in the previous answer, summarize the number of products per each price category.

Answer:

```
SELECT
	CASE
		WHEN product_price<=50 THEN 'Low Price'
		WHEN product_price>50 AND product_price<100 THEN 'Medium Price'
		ELSE 'High Price'
	END AS price_category, count(distinct product_id) AS amount_products
FROM ecommerce_schema.products
GROUP BY 1
```

Part 4 - Window Functions

Exercise #12

List of products (product id, product price, product category) and for each product, the average product price of the product category. Remove rows with NULL value in product price, and order the result based on product price descending.

Answer:

```
SELECT product_id, product_price, product_category_name, avg(product_price) OVER (PARTITION BY product_category_name) as avg_category_price

FROM ecommerce_schema.products

WHERE product_price IS NOT NULL

ORDER BY 2 DESC;
```

Exercise #13

List the five heaviest products (product id, product category, product weight) from each product category.

Answer:

```
SELECT *

FROM (

    SELECT product_id, product_category_name, product_weight_g, ROW_NUMBER()
    OVER (PARTITION BY product_category_name ORDER BY product_weight_g DESC) as
    row_num

    FROM ecommerce_schema.products

    WHERE product_price IS NOT NULL

  ) as t

WHERE row_num <= 5;
```

Part 5 - Simplify Queries (Views, CTEs)

Exercise #14

Create a view called "customers_shipped_vw" for all customers (customer id, customer name, customer city, order status) with orders that are in "shipped" status.

Answer:

```
CREATE VIEW ecommerce_schema.customers_shipped_vw AS

(

   SELECT c.customer_id, c.customer_name, c.customer_city, o.order_status

   FROM ecommerce_schema.customers as c

   INNER JOIN ecommerce_schema.orders as o ON c.customer_id = o.customer_id

   WHERE o.order_status = 'shipped'

);

SELECT *

FROM ecommerce_schema.customers_shipped_vw;
```

Exercise #15

Create a CTE ("with") that will create a sequential number based on the product price for each product category (product id, product category, product price, row num) and remove rows with NULL value in the product price.

Use the CTE to query for the three most expensive products from each category.

Answer:

```
WITH product_category_CTE

AS

(

  SELECT product_id, product_category_name, product_price, ROW_NUMBER() OVER (PARTITION
BY product_category_name ORDER BY product_price DESC) as row_num

  FROM ecommerce_schema.products

  WHERE product_price IS NOT NULL

)

SELECT *

FROM product_category_CTE

WHERE row_num <= 3
```