Name : Muhammad Awais

Basic Queries (5 queries)

Query 1: Simple SELECT: Retrieve all columns from a customers table

Purpose:
To view all customer records stored in the database. This helps the business understand the full customer base.

Concepts Used:

- SELECT → to specify the columns to return
- * → wildcard to select all columns
- FROM → specifies the table from which data is fetched

SQL Code: **SELECT * FROM Customers;**

Expected Output:

```
SELECT * from customers
```

14 %

Results | Messages

	customer_id	customer_unique_id	customer_zip_code_prefix	customer_city	customer_state
1	06b8999e2fba1a1fbc88172c00ba8bc7	861eff4711a542e4b93843c6dd7febb0	14409	franca	SP
2	18955e83d337fd6b2def6b18a428ac77	290c77bc529b7ac935b93aa66c333dc3	9790	sao bernardo do campo	SP
3	4e7b3e00288586ebd08712fdd0374a03	060e732b5b29e8181a18229c7b0b2b5e	1151	sao paulo	SP
4	b2b6027bc5c5109e529d4dc6358b12c3	259dac757896d24d7702b9acbbff3f3c	8775	mogi das cruzes	SP
5	4f2d8ab171c80ec8364f7c12e35b23ad	345ecd01c38d18a9036ed96c73b8d066	13056	campinas	SP
6	879864dab9bc3047522c92c82e1212b8	4c9374416667ad3b8f1fb645a3116a4	89254	jaragua do sul	SC
7	fd826e7cf63160e536e0908c76c3f441	addec96d2e059c80c30fe6871d30d177	4534	sao paulo	SP
8	5e274e7a0c3809e14aba7ad5aae0d407	57b2a98a40981cfe9618067b6b8ebe4f	35182	timoteo	MG
9	5adf08e34b2e993982a47070956c5c65	1175e95fb47ddff9de6b2b06188f7e0d	81560	curitiba	PR
10	4b7139f34592b3a31687243a302fa75b	9afe194fb833f79e300e37e580171f22	30575	belo horizonte	MG
11	9fb35e4ed6f0a14a4977cd9aea4042bb	2a7745e1ed516b289ed9b29c7d0539a5	39400	montes claros	MG
12	5aa9e4fdd4dfd20959cad2d772509598	2a49fb94aef5cbeeb850418118cee090	20231	rio de janeiro	RJ
13	b2d1536598b73a9abd18e0d75d92f0a3	918dc87cd72cd9f6ed4bd442ed785235	18682	lencois paulista	SP
14	eabebad39a88bb9f5b52376faec28612	295c05e81917928d76245e842748184d	5704	sao paulo	SP
15	1f1c7bf1c9b041b292af6c1c4470b753	3151a81801c8386361b62277d7fa5ecf	95110	caxias do sul	RS
16	206f3129c0e4d7d0b9550426023f0a08	21f748a16f4e1688a9014eb3ee6fa325	13412	piracicaba	SP
17	a7c125a0a07b75146167b7f04a7f8e98	5c2991dbd08bbf3cf410713c4de5a0b5	22750	rio de janeiro	RJ
18	c5c61596a3b6bd0cee5766992c48a9a1	b6e99561fe6f34a55b0b7da92f8ed775	7124	guarulhos	SP
19	9b8ce803689b3562defaad4613ef426f	7f3a72e8f988c6e735ba118d54f47458	5416	sao paulo	SP

Business Insight:
This query gives the business a full view of the customer database, which can be used as a starting point for more detailed data analysis.

Query 2: Column Selection: Select specific columns (name, email, city) from customers

Purpose:

If we want to fetch only specific records from the customers table, instead of fetching whole table.

Concepts Used:

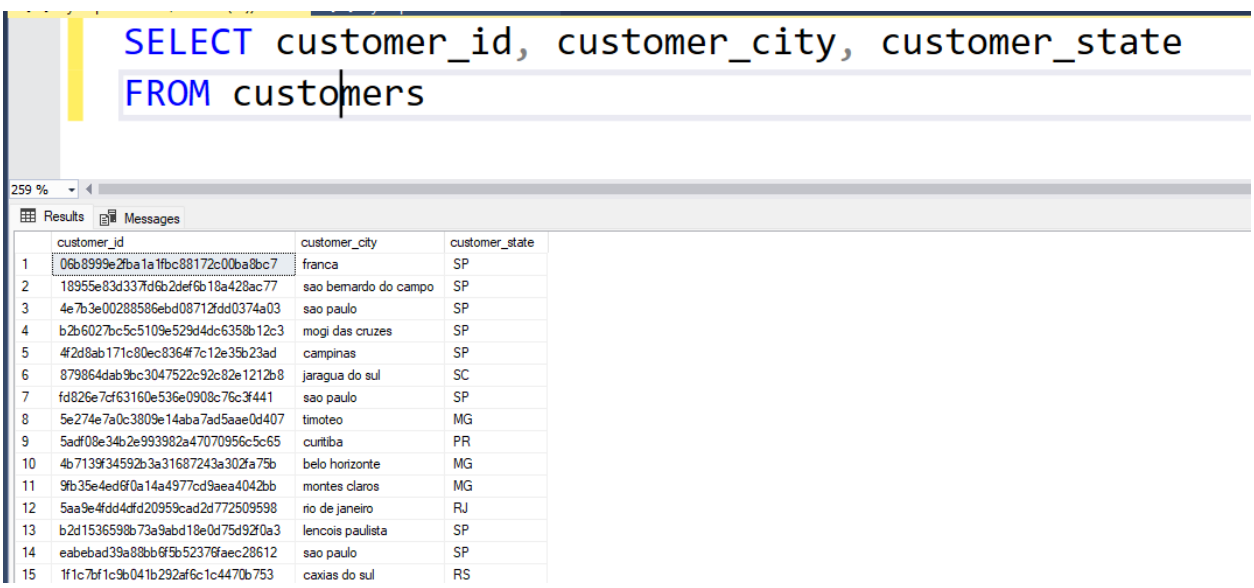
- SELECT > to specify what to return
- Specifying column name
- FROM, to specify the table from where data is fetched

SQL Code:

```
SELECT customer_id, customer_city, customer_state

FROM customers
```

Expected Output:



```
SELECT customer_id, customer_city, customer_state
FROM customers
```

259 %

Results | Messages

	customer_id	customer_city	customer_state
1	06b8999e2fba1a1fbc88172c00ba8bc7	franca	SP
2	18955e83d337fd6b2def6b18a428ac77	sao bernardo do campo	SP
3	4e7b3e00288586ebd08712fdd0374a03	sao paulo	SP
4	b2b6027bc5c5109e529d4dc6358b12c3	mogi das cruzes	SP
5	4f2d8ab171c80ec8364f7c12e35b23ad	campinas	SP
6	879864dab9bc3047522c92c82e1212b8	jaragua do sul	SC
7	fd826e7cf63160e536e0908c76c3f441	sao paulo	SP
8	5e274e7a0c3809e14aba7ad5aae0d407	timoteo	MG
9	5adf08e34b2e993982a47070956c5c65	curitiba	PR
10	4b7139f34592b3a31687243a302fa75b	belo horizonte	MG
11	9fb35e4ed6f0a14a4977cd9aea4042bb	montes claros	MG
12	5aa9e4fdd4dfd20959cad2d772509598	rio de janeiro	RJ
13	b2d1536598b73a9abd18e0d75d92f0a3	lencois paulista	SP
14	eabebad39a88bb6f5b52376faec28612	sao paulo	SP
15	1f1c7bf1c9b041b292af6c1c4470b753	caxias do sul	RS

Business Insight:
Shows where customers live, helping the business understand demand, plan marketing, and improve deliveries.

Query 3: WHERE Filtering: e.g. Find customers from a specific city

- **Purpose:**
  To list customers who live in a given city (in this case, *Franca*).

- **Concepts Used:**
  - SELECT → choose customer ID
  - FROM → source table (customers)
  - WHERE → filter rows by condition
- **SQL Code:**

```
SELECT customer_id
FROM customers
WHERE customer_city = 'franca';
```

- **Expected Output:**
  A list of customer IDs for customers whose city is **Franca**.

```
SELECT customer_id
FROM customers
WHERE customer_city='franca';
```

259 %

Results | Messages

	customer_id
1	06b8999e2fba1a1fbc88172c00ba8bc7
2	b9885472fa562ddc1cb5333a42b654fc
3	5dca924cc99eea2dc5ba40d11ec5dd0f
4	de3fa59dd658cd8b47a1634125047c01
5	0489975a325480c9e385e9f135bb13c3
6	33a0ede763ee223996e200f111e33592
7	9d1e06b0ab7ef29fa0e9cea88bdc7eeb
8	1a700c8b0ea5c7700237221567261552
9	5014401cb0faa8ae93727383cd39c424
10	6d82091ecad0c0c4848d8e1066550f20
11	661897d4968f1b59bfff74c7eb2eb4fc
12	d31d9ee6ccc7152f0782b24bc5570add
13	285b02dd357a06f32b97e0e1491c2317
14	693d02fe5397d7acda742a75d955610e
15	ce1798e7ceaec0a413a5ba767d658f24

- **Business Insight:**
  Helps the business see how many customers are in a **specific city**.

Query 4: Multiple Conditions (AND Filtering)

- **Purpose:**
  To list customers who are from a specific state and whose zip code prefix is 14409.
- **Concepts Used:**
  - SELECT → choose customer ID, city, and state
  - FROM → source table (Customers)

- WHERE → filter rows by multiple conditions (state AND zip code)

SQL Code:

Select customer_id, Customer_city, customer_state

FROM Customers

Where customer_state='SP' and customer_zip_code_prefix=14409

Expected Output:

SQLQuery1.sql - D...6HFL\Rehman (71))*

```
Select customer_id, Customer_city, customer_state

FROM Customers

Where customer_state='SP' and customer_zip_code_prefix=14409
```

214 %

Results | Messages

	customer_id	Customer_city	customer_state
1	06b8999e2fba1a1fbc88172c00ba8bc7	franca	SP
2	5dca924cc99eea2dc5ba40d11ec5dd0f	franca	SP
3	661897d4968f1b59bfff74c7eb2eb4fc	franca	SP
4	702b62324327ccba20f1be3465426437	franca	SP
5	bdf997bae7ca819b0415f5174d6b4302	franca	SP
6	48a9965057e1c75ed3a4d511714be18b	franca	SP
7	96fbbe8005e67d382b0fc453a288be02	franca	SP
8	7d58157ef9ab184d29d36f009c7ad786	franca	SP
9	d3c09791fc7ede91935f0488a3375a6	franca	SP

- **Business Insight:**
  This helps the business narrow down customers to a very specific location , useful for regional promotions, delivery planning, or market segmentation.

Query 5: ORDER BY Sorting

- **Purpose:**
  To list all customers and sort them alphabetically by their city.
- **Concepts Used:**
  - SELECT → choose customer ID, city, and state
  - FROM → source table (Customers)
  - ORDER BY → sort results by a specific column (here, customer_city)
  - ASC → ascending order (A → Z)

SQL Code:

Select customer_id, Customer_city, customer_state

FROM Customers

ORDER BY customer_city ASC

Expected Output:

```
Select customer_id, Customer_city, customer_state
FROM Customers
ORDER BY customer_city ASC
```

14 %

Results | Messages

	customer_id	Customer_city	customer_state
13	08528824266cd0720658ff01df662b6a	abaete	MG
14	24d97ea7bb12a40f87d701e17017d282	abaete	MG
15	e67618ef4120e6f73c343ef079dc405e	abaete	MG
16	3c46a6771b9833c7571cd2498f548f95	abaete	MG
17	3c12346ea1f05ce151ba3029c1cbf60d	abaetetuba	PA
18	987b15cf895b62d1fe337652f44f802f	abaetetuba	PA
19	ba1249c05a651c21b3e6a99caaad9525	abaetetuba	PA
20	305f736d2711641a06f6a9c9bd837b00	abaetetuba	PA
21	c7eb06383ae604616cb4c9d36fe6745e	abaetetuba	PA
22	7727e2cc9ec428ad3173cc812ce1781b	abaetetuba	PA

- **Business Insight:**
  Sorting customers by city makes it easier for the business to group customers geographically.

Intermediate Queries (5 queries)

Query 6: COUNT with GROUP BY: e.g. Count customers by city

- **Purpose:** Find out how many customers live in each city.
- **Concepts Used:**
  - COUNT() → counts number of customers
  - GROUP BY → groups rows by city before counting

SQL Code:

SELECT customer_city,

    COUNT(customer_id) AS total_customers

FROM Customers

GROUP BY customer_city;

Expected Output:

```sql
SELECT customer_city,
       COUNT(customer_id) AS total_customers
FROM Customers
GROUP BY customer_city;
```

214 %

Results | Messages

	customer_city	total_customers
1	claudia	4
2	santa salete	1
3	sao jose dos cordeiros	1
4	santa izabel do oeste	2
5	andradina	43
6	mostardas	11
7	buritis	16
8	cacule	3
9	santa quiteria	5
10	novo jardim	1
11	sud mennucci	5
12	deodapolis	2
13	lajeado	51
14	santo antonio de posse	20
15	planalto	9
16	quinta do sol	3
17	jardim do serido	3
18	santanesia	4
19	oliveira dos brejinhos	2
20	sao goncalo do abaete	2
21	goianinha	3

Business Insight: Helps the business see which cities have more or fewer customers.

Query 7: SUM with GROUP BY: e.g. Total sales by product category

- **Purpose:**
  To see how many products were sold in each category.
- **Concepts Used:**
  - COUNT() → counts sales (each product_id = 1 sale)
  - GROUP BY → groups by category

SQL Code:

```
SELECT product_category_name,
    COUNT(product_id) AS total_sales
FROM Products
GROUP BY product_category_name;
```

Expected Output:

```
SELECT product_category_name,
       COUNT(product_id) AS total_sales
FROM Products
GROUP BY product_category_name;
```

214 %

Results | Messages

	product_category_name	total_sales
1	casa_conforto_2	5
2	perfumaria	868
3	fashion_calcados	173
4	pcs	30
5	artigos_de_natal	65
6	bebidas	81
7	construcao_ferramentas_jardim	88
8	construcao_ferramentas_seguranca	91
9	telefonia	1134

Business Insight:
Helps the business see which product categories sell the most.

Query 8: AVG with GROUP BY: Average order value by customer segment

- **Purpose:**
  To find the average weight of products in each category.
- **Concepts Used:**
  - AVG() → calculates the average
  - GROUP BY → groups results by category

SQL Code:

```
SELECT product_category_name,
    AVG(product_weight_g) AS avg_weight
FROM Products
GROUP BY product_category_name;
```

Expected Output:

```sql
SELECT product_category_name,
       AVG(product_weight_g) AS avg_weight
FROM Products
GROUP BY product_category_name;
```

Results | Messages

product_category_name	avg_weight
alimentos_bebidas	1307
dvds_blu_ray	381
artes	1691
eletronicos	1275
moveis_sala	8934
brinquedos	1869
sinalizacao_e_seguranca	2969
utilidades_domesticas	3020
fashion_roupa_feminina	572

Business Insight:
Shows which categories usually have heavier or lighter products.

Query 9: Cities with more than 10 customers

- **Purpose:**
  To find cities that have a large customer base.
- **Concepts Used:**
  - COUNT() → counts customers per city
  - GROUP BY → groups by city
  - HAVING → filters groups after aggregation
- **SQL Code:**

```
SELECT customer_city,
    COUNT(customer_id) AS total_customers
```

FROM Customers
GROUP BY customer_city
HAVING COUNT(customer_id) > 10;

Expected Output:

```sql
SELECT customer_city,
       COUNT(customer_id) AS total_customers
FROM Customers
GROUP BY customer_city
HAVING COUNT(customer_id) > 10;
```

214 %

Results | Messages

	customer_city	total_customers
1	andradina	43
2	mostardas	11
3	buritis	16
4	lajeado	51
5	santo antonio de posse	20
6	aluminio	13
7	itauna	57
8	santa vitoria do palmar	12
9	laranjeiras do sul	20
10	arapongas	35
11	itapeva	58
12	volta redonda	232
13	botucatu	101
14	espera feliz	19
15	sapucaia do sul	40
16	araguaina	41
17	sao francisco de itabapoana	15
18	jaboatao dos guararapes	135
19	quirinopolis	14

- **Business Insight:**
  Helps the business see which cities have enough customers for focused marketing.

Query 10 : Highest and lowest product weight by category

- **Purpose:**
  To find the heaviest and lightest products in each category.
- **Concepts Used:**
  - MAX() → finds the largest value
  - MIN() → finds the smallest value

- GROUP BY → groups results by category

SQL Code:

```
SELECT product_category_name,
    MAX(product_weight_g) AS max_weight,
    MIN(product_weight_g) AS min_weight
FROM Products
GROUP BY product_category_name;
```

Expected Output:

	product_category_name	max_weight	min_weight
1	casa_conforto_2	3750	425
2	perfumaria	16050	50
3	fashion_calcados	9000	100
4	pcs	25950	200
5	artigos_de_natal	15400	50
6	bebidas	9900	100
7	construcao_ferramentas_jardim	25750	65
8	construcao_ferramentas_seguranca	9450	53
9	telefonia	8100	50
10	automotivo	30000	50
11	flores	6000	200
12	industria_comercio_e_negocios	30000	200
13	fashion_bolsas_e_acessorios	21100	50
14	beleza_saude	30000	50
15	telefonia_fixa	6050	50
16	alimentos_bebidas	10800	50
17	dvds_blu_ray	2600	90
18	artes	15400	100
19	eletronicos	24050	60
20	moveis_sala	30000	400

- **Business Insight:**
  Shows weight range per category, useful for logistics and shipping planning.

Advanced Queries (5 queries)

Query 11: INNER JOIN → Customers with their Orders

- **Purpose:** Show which customers placed which orders.

- **Concepts Used:**
    - INNER JOIN → matches customers with their orders using customer_id.

SQL Code:

```
SELECT c.customer_id, c.customer_city, o.order_id

FROM Customers c

INNER JOIN Orders o

 ON c.customer_id = o.customer_id;
```

Expected Output:

```
SELECT c.customer_id, c.customer_city, o.order_id
FROM Customers c
INNER JOIN Orders o
  ON c.customer_id = o.customer_id;
```

195 %

Results | Messages

	customer_id	customer_city	order_id
1	8b212b9525f9e74e85e37ed6df37693e	sao carlos	dd78f560c270f1909639c11b925620ea
2	456dc10730fbdba34615447ea195d643	sao paulo	f70a0aff17df5a6cdd9a7196128bd354
3	d9ef95f98d8da3b492bb8c0447910498	mesquita	d887b52c6516beb39e8cd44a5f8b60f7
4	5f16605299d698660e0606f7eae2d2f9	alvorada	8563039e855156e48fccee4d611a3196
5	3391c4bc11a817e7973e498b0b023158	sao paulo	7206b86ea789983f7a273ea7fa0bc2a8
6	148348ff65384b4249b762579532e248	paranavai	47aa4816b27ba60ec948cd019cc1afc1
7	2932d241d1f31e6df6c701d52370ae02	sao paulo	734e7d1bbaeb2ff82521ca0fe6fb6f79
8	a9d37ddc8ba4d9f6dbac7d8ec378cc95	joinville	fdf128b3630c21adc9ca4fb8a51b68ec
9	f5618502bee8eafdee72fb6955e2ebdf	carapicuiba	6abaad69b8b349c3a529b4b91ce18e46
10	48558a50a7ba1aab61891936d2ca7681	jequitai	20e0101b20700188cadb288126949685
11	3a897024068ed42a183de61d5727d866	sao paulo	0e782c3705510e717d28907746cbda82

- **Business Insight:** Helps see which customers are active and what orders they made.

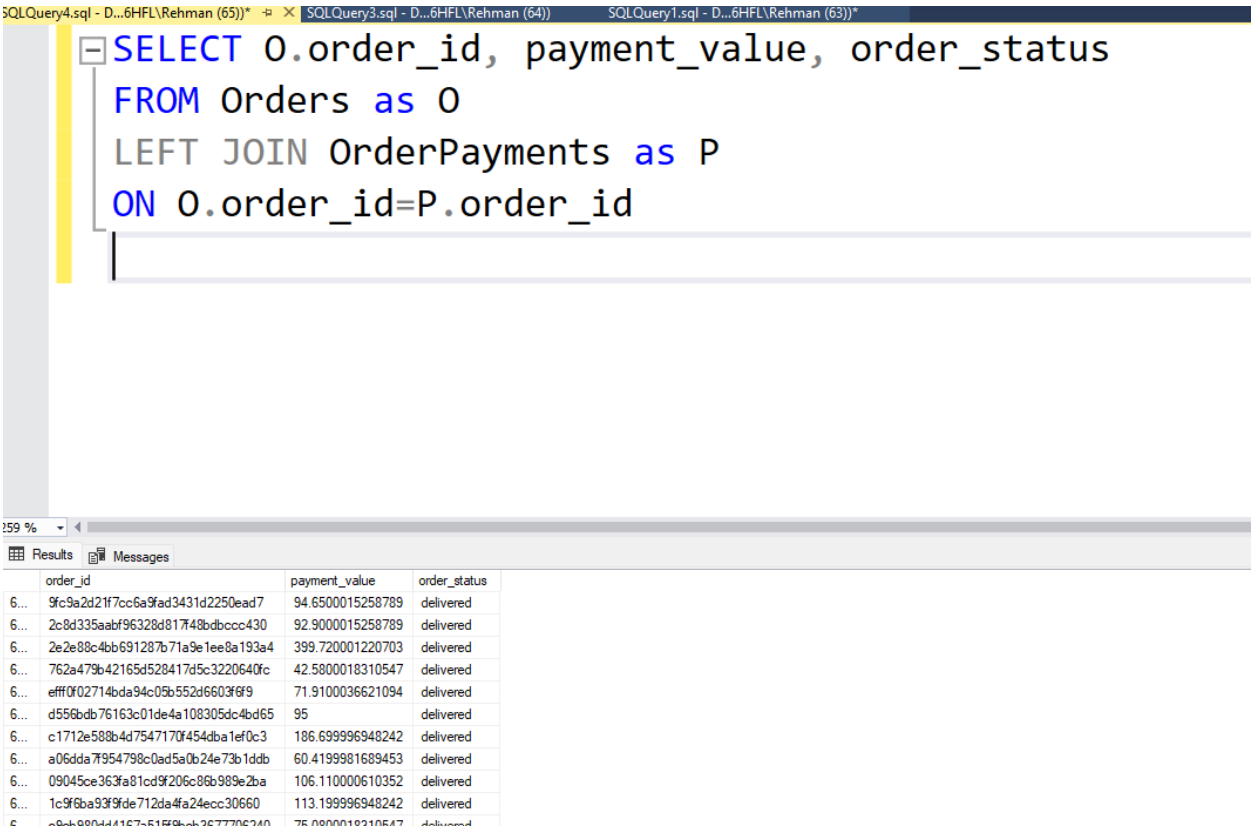
Query 12: LEFT JOIN: e.g. All customers and their orders (including customers without orders)

- **Purpose:** Show all orders, even if they don't have payment records.

- **Concepts Used:**
    - LEFT JOIN → keeps all rows from Orders, and only matches payments if available.
- **SQL Code:**

```
SELECT O.order_id, payment_value, order_status
FROM Orders as O
LEFT JOIN OrderPayments as P
ON O.order_id=P.order_id
```

Expected Output:



	order_id	payment_value	order_status
6...	9fc9a2d21f7cc6a9fad3431d2250ead7	94.6500015258789	delivered
6...	2c8d335aabf96328d817f48bdbccc430	92.9000015258789	delivered
6...	2e2e88c4bb691287b71a9e1ee8a193a4	399.720001220703	delivered
6...	762a479b42165d528417d5c3220640fc	42.5800018310547	delivered
6...	efff0f02714bda94c05b552d6603f6f9	71.9100036621094	delivered
6...	d556bdb76163c01de4a108305dc4bd65	95	delivered
6...	c1712e588b4d7547170f454dba1ef0c3	186.699996948242	delivered
6...	a06dda7f954798c0ad5a0b24e73b1ddb	60.4199981689453	delivered
6...	09045ce363fa81cd9f206c86b989e2ba	106.110000610352	delivered
6...	1c9f6ba93f9fde712da4fa24ecc30660	113.199996948242	delivered

- **Business Insight:** Helps check unpaid orders or missing payment records.

Query 13: Right JOIN: All Products and their Sales Data

- **Purpose:** Show all products, even if they were never sold.

- **Concepts Used:**
    - RIGHT JOIN → keeps all rows from Products, shows sales only if they exist.
- **SQL Code:**

```sql
SELECT P.product_id,
    P.product_category_name,
    OI.order_id
FROM [Order Items] AS OI
RIGHT JOIN Products AS P
 ON OI.product_id = P.product_id;
```

Expected Output

```sql
SELECT P.product_id,
       P.product_category_name,
       OI.order_id
FROM [Order Items] AS OI
RIGHT JOIN Products AS P
  ON OI.product_id = P.product_id;
```

14 %

Results | Messages

	product_id	product_category_name	order_id
1	e5f2d52b802189ee658865ca93d83a8f	pet_shop	00018f77f2f0320c557190d7a144bdd3
2	c777355d18b72b67abbeef9df44fd0fd	moveis_decoracao	000229ec398224ef6ca0657da4fc703e
3	557d850972a7d6f792fd18ae1400d9b6	ferramentas_jardim	000576fe39319847cbb9d288c5617fa6
4	4535b0e1091c278dfd193e5a1d63b39f	livros_tecnicos	0005f50442cb953dcd1d21e1fb923495
5	1c05e0964302b6cf68ca0d15f326c6ba	moveis_escritorio	000f25f4d72195062c040b12dce9a18a
6	5a419dbf24a8c9718fe522b81c69f61a	cool_stuff	0010b2e5201cc5f1ae7e9c6cc8f5bd00
7	21b1c2f67a9aafb5af0eb06c13b9dbda	esporte_lazer	00119ff934e539cf26f92b9ef0cdfed8
8	89321f94e35fc6d7903d36f74e351d40	alimentos	00130c0eee84a3d909e75bc08c5c3ca1
9	b10eba910a974df70b8a12d0665cdb9e	cama_mesa_banho	00169e31ef4b29deaae414f9a5e95929

- **Business Insight:** Helps identify unsold products and track demand by category.

Query 14: Multiple JOINs → Customer, Order, and Product Info

- **Purpose:** Combine customer details with their orders and the products they bought.
- **Concepts Used:**
    - INNER JOIN → links Customers → Orders → Order Items → Products.
- **SQL Code:**

```sql
SELECT C.customer_id,
    C.customer_city,
    O.order_id,
    P.product_id,
    P.product_category_name
FROM Customers AS C
INNER JOIN Orders AS O
 ON C.customer_id = O.customer_id
INNER JOIN [Order Items] AS OI
 ON O.order_id = OI.order_id
INNER JOIN Products AS P
 ON OI.product_id = P.product_id;
```

Expected Output

```sql
SELECT C.customer_id,
       C.customer_city,
       O.order_id,
       P.product_id,
       P.product_category_name
FROM Customers AS C
INNER JOIN Orders AS O
  ON C.customer_id = O.customer_id
INNER JOIN [Order Items] AS OI
  ON O.order_id = OI.order_id
INNER JOIN Products AS P
  ON OI.product_id = P.product_id;
```

77 %

Results | Messages

	customer_id	customer_city	order_id	product_id	product_category_name
1	f3476df115c5d26bdb6bfb71b5fb66af	sao paulo	00378c6c981f234634c0b9d6128df6dd	38fa750a3a3b3204f169c86a3284d387	esporte_lazer
2	f9173fb8dc039cb6acfcc3a7a2fc49eb	sao paulo	0533573d16b0c12cc9a3f9f97f67111b	1b28df0b54809d40f2bd2f2218c601a2	alimentos_bebidas
3	cfd03e2bdbef45d8d27ada56c4ae0348	goiania	05709f85ee94e226753ae0ed0886b69b	b19e331e05bee8900ed6e582b4c8dab6	cama_mesa_banho
4	cde010b04a18c2bd1acfdca70135a4e7	campo mourao	13e5cf1e067d58500141bfb2a638ba6b	1d7db62a21ffb4f220d3e8a05fa77e02	fashion_bolsas_e_acessorios
5	998671ecb4efd9e7e5a926705e142196	francisco morato	1ddbde4b8d3dc88ae08a92c7ecf6c2f7	5a7373ba4c9b3e13513ded06cd1986c4	esporte_lazer
6	430b306580959c99c25a228924eaede0	aguas formosas	18ca340d8c5cef7fba3c9c18efce43e9	3bbb1f94c6871212f10e8c25012a8e19	telefonia
7	9fd88b5e868210b0e4a9fed9889ea944	paranaiguara	13f747cacb95a747110cbb87bc2c75ce	e44f675b60b3a3a2453ec36421e06f0f	esporte_lazer
8	4fbd4b0638090bdcac4bd0a0add76ed5	sao paulo	13fd61236a6baeb00aba9df344275a78	e0cf79767c5b016251fe139915c59a26	beleza_saude
9	767e04f5c232aae9f6efb7bf1fb04cf1	ouro preto	14083f06e3e4bf1cd90b09251b0b7061	b72569983090c8bcd75fe577f5be5934	eletrodomesticos
10	32a999da0202254e1933631170777178	saquarema	149d32ba42826835e09c375b8a1fa923	dbb67791e405873b259e4656bf971246	informatica_acessorios

- **Business Insight:** Gives a full view of customer purchasing behavior, useful for product recommendations or city-level sales analysis.

Query 15: Complex – Monthly Sales Report with Customer Segments and Top Products

- **Purpose:** To see monthly sales, grouped by customer location (segment) and the top products sold.
- **Concepts Used:**
  - JOINs → combine Customers, Orders, OrderItems, Products.
  - DATEPART → extract month and year from order date.
  - GROUP BY → aggregate sales by month, city, and product.
  - SUM → calculate total sales.
- SQL Code:

```sql
SELECT
  DATEPART(YEAR, O.order_purchase_timestamp) AS order_year,
  DATEPART(MONTH, O.order_purchase_timestamp) AS order_month,
  C.customer_city,
  P.product_category_name,
  SUM(P.product_weight_g) AS total_sold_weight
FROM Orders AS O
INNER JOIN Customers AS C
 ON O.customer_id = C.customer_id
INNER JOIN [Order Items] AS OI
 ON O.order_id = OI.order_id
INNER JOIN Products AS P
 ON OI.product_id = P.product_id
GROUP BY
  DATEPART(YEAR, O.order_purchase_timestamp),
  DATEPART(MONTH, O.order_purchase_timestamp),
  C.customer_city,
  P.product_category_name
ORDER BY
  order_year, order_month, total_sold_weight DESC;
```

Expected Output:

```sql
SELECT
    DATEPART(YEAR, O.order_purchase_timestamp) AS order_year,
    DATEPART(MONTH, O.order_purchase_timestamp) AS order_month,
    C.customer_city,
    P.product_category_name,
    SUM(P.product_weight_g) AS total_sold_weight
FROM Orders AS O
INNER JOIN Customers AS C
  ON O.customer_id = C.customer_id
INNER JOIN [Order Items] AS OI
  ON O.order_id = OI.order_id
INNER JOIN Products AS P
  ON OI.product_id = P.product_id
GROUP BY
    DATEPART(YEAR, O.order_purchase_timestamp),
    DATEPART(MONTH, O.order_purchase_timestamp),
    C.customer_city,
    P.product_category_name
ORDER BY
    order_year, order_month, total_sold_weight DESC;
```

.0 %

Results | Messages

	order_year	order_month	customer_city	product_category_name	total_sold_weight
1	2016	9	boa vista	moveis_decoracao	3200
2	2016	9	sao joaquim da barra	beleza_saude	3000
3	2016	9	passo fundo	telefonia	700
4	2016	10	igarassu	esporte_lazer	30000
5	2016	10	rio de janeiro	beleza_saude	28750
6	2016	10	quissama	esporte_lazer	28500

- **Business Insight:** Helps track sales trends over time, find best-selling products, and see which cities drive demand each month.