```sql
-- Analyse sales performance over time
-- Quick Date Functions
SELECT
    YEAR(order_date) AS order_year,
    MONTH(order_date) AS order_month,
    SUM(sales_amount) AS total_sales,
    COUNT(DISTINCT customer_key) AS total_customers,
    SUM(quantity) AS total_quantity
FROM gold.fact_sales
WHERE order_date IS NOT NULL
GROUP BY YEAR(order_date), MONTH(order_date)
ORDER BY YEAR(order_date), MONTH(order_date);
```

136 %

Results | Messages

	order_year	order_month	total_sales	total_customers	total_quantity
1	2010	12	43419	14	14
2	2011	1	469795	144	144
3	2011	2	466307	144	144
4	2011	3	485165	150	150
5	2011	4	502042	157	157
6	2011	5	561647	174	174
7	2011	6	737793	230	230
8	2011	7	596710	188	188
9	2011	8	614516	193	193
10	2011	9	603047	185	185
11	2011	10	708164	221	221
12	2011	11	660507	208	208
13	2011	12	669395	222	222
14	2012	1	495363	252	252
15	2012	2	506992	260	260
16	2012	3	373478	212	212
17	2012	4	400324	219	219
18	2012	5	358866	207	207
19	2012	6	555142	318	318
20	2012	7	444533	246	246
21	2012	8	523887	294	294
22	2012	9	486149	269	269

Query executed successfully.

```sql
-- DATETRUNC()
SELECT
    DATETRUNC(month, order_date) AS order_date,
    SUM(sales_amount) AS total_sales,
    COUNT(DISTINCT customer_key) AS total_customers,
    SUM(quantity) AS total_quantity
FROM gold.fact_sales
WHERE order_date IS NOT NULL
GROUP BY DATETRUNC(month, order_date)
ORDER BY DATETRUNC(month, order_date);
```

136 %

Results | Messages

	order_date	total_sales	total_customers	total_quantity
1	2010-12-01	43419	14	14
2	2011-01-01	469795	144	144
3	2011-02-01	466307	144	144
4	2011-03-01	485165	150	150
5	2011-04-01	502042	157	157
6	2011-05-01	561647	174	174
7	2011-06-01	737793	230	230
8	2011-07-01	596710	188	188
9	2011-08-01	614516	193	193
10	2011-09-01	603047	185	185
11	2011-10-01	708164	221	221
12	2011-11-01	660507	208	208
13	2011-12-01	669395	222	222
14	2012-01-01	495363	252	252
15	2012-02-01	506992	260	260
16	2012-03-01	373478	212	212
17	2012-04-01	400324	219	219
18	2012-05-01	358866	207	207
19	2012-06-01	555142	318	318
20	2012-07-01	444533	246	246
21	2012-08-01	523887	294	294
22	2012-09-01	486149	269	269
23	2012-10-01	535125	313	313
24	2012-11-01	537918	324	324
25	2012-12-01	624454	354	483
26	2013-01-01	857758	627	1677
27	2013-02-01	771218	1373	3454

Query executed successfully.

```sql
-- FORMAT()
SELECT
    FORMAT(order_date, 'yyyy-MMM') AS order_date,
    SUM(sales_amount) AS total_sales,
    COUNT(DISTINCT customer_key) AS total_customers,
    SUM(quantity) AS total_quantity
FROM gold.fact_sales
WHERE order_date IS NOT NULL
GROUP BY FORMAT(order_date, 'yyyy-MMM')
ORDER BY FORMAT(order_date, 'yyyy-MMM');
```

136 %

Results | Messages

	order_date	total_sales	total_customers	total_quantity
1	2010-Dec	43419	14	14
2	2011-Apr	502042	157	157
3	2011-Aug	614516	193	193
4	2011-Dec	669395	222	222
5	2011-Feb	466307	144	144
6	2011-Jan	469795	144	144
7	2011-Jul	596710	188	188
8	2011-Jun	737793	230	230
9	2011-Mar	485165	150	150
10	2011-May	561647	174	174
11	2011-Nov	660507	208	208
12	2011-Oct	708164	221	221
13	2011-Sep	603047	185	185
14	2012-Apr	400324	219	219
15	2012-Aug	523887	294	294
16	2012-Dec	624454	354	483
17	2012-Feb	506992	260	260
18	2012-Jan	495363	252	252
19	2012-Jul	444533	246	246
20	2012-Jun	555142	318	318
21	2012-Mar	373478	212	212
22	2012-May	358866	207	207
23	2012-Nov	537918	324	324
24	2012-Oct	535125	313	313

Query executed successfully.

```sql
/*
===============================================================================
Purpose:
    - To calculate running totals or moving averages for key metrics.
    - To track performance over time cumulatively.
    - Useful for growth analysis or identifying long-term trends.

SQL Functions Used:
    - Window Functions: SUM() OVER(), AVG() OVER()
===============================================================================
*/

-- Calculate the total sales per month
-- and the running total of sales over time
SELECT
    order_date,
    total_sales,
    SUM(total_sales) OVER (ORDER BY order_date) AS running_total_sales,
    AVG(avg_price) OVER (ORDER BY order_date) AS moving_average_price
FROM
(
    SELECT
        DATETRUNC(year, order_date) AS order_date,
        SUM(sales_amount) AS total_sales,
        AVG(price) AS avg_price
    FROM gold.fact_sales
    WHERE order_date IS NOT NULL
    GROUP BY DATETRUNC(year, order_date)
) t
```

136 %

Results | Messages

	order_date	total_sales	running_total_sales	moving_average_price
1	2010-01-01	43419	43419	3101
2	2011-01-01	7075088	7118507	3146
3	2012-01-01	5842231	12960738	2670
4	2013-01-01	16344878	29305616	2080
5	2014-01-01	45642	29351258	1668

Query executed successfully.

```sql
/* Analyze the yearly performance of products by comparing their sales
to both the average sales performance of the product and the previous year's sales */
WITH yearly_product_sales AS (
    SELECT
        YEAR(f.order_date) AS order_year,
        p.product_name,
        SUM(f.sales_amount) AS current_sales
    FROM gold.fact_sales f
    LEFT JOIN gold.dim_products p
        ON f.product_key = p.product_key
    WHERE f.order_date IS NOT NULL
    GROUP BY
        YEAR(f.order_date),
        p.product_name
)
SELECT
    order_year,
    product_name,
    current_sales,
    AVG(current_sales) OVER (PARTITION BY product_name) AS avg_sales,
    current_sales - AVG(current_sales) OVER (PARTITION BY product_name) AS diff_avg,
    CASE
        WHEN current_sales - AVG(current_sales) OVER (PARTITION BY product_name) > 0 THEN 'Above Avg'
        WHEN current_sales - AVG(current_sales) OVER (PARTITION BY product_name) < 0 THEN 'Below Avg'
        ELSE 'Avg'
    END AS avg_change,
    -- Year-over-Year Analysis
    LAG(current_sales) OVER (PARTITION BY product_name ORDER BY order_year) AS py_sales,
    current_sales - LAG(current_sales) OVER (PARTITION BY product_name ORDER BY order_year) AS diff_py,
    CASE
        WHEN current_sales - LAG(current_sales) OVER (PARTITION BY product_name ORDER BY order_year) > 0 THEN 'Increase'
        WHEN current_sales - LAG(current_sales) OVER (PARTITION BY product_name ORDER BY order_year) < 0 THEN 'Decrease'
        ELSE 'No Change'
    END AS py_change
FROM yearly_product_sales
ORDER BY product_name, order_year;
```

93 %

Results | Messages

	order_year	product_name	current_sales	avg_sales	diff_avg	avg_change	py_sales	diff_py	py_change
1	2012	All-Purpose Bike Stand	159	13197	-13038	Below Avg	NULL	NULL	No Change
2	2013	All-Purpose Bike Stand	37683	13197	24486	Above Avg	159	37524	Increase
3	2014	All-Purpose Bike Stand	1749	13197	-11448	Below Avg	37683	-35934	Decrease
4	2012	AWC Logo Cap	72	6570	-6498	Below Avg	NULL	NULL	No Change
5	2013	AWC Logo Cap	18891	6570	12321	Above Avg	72	18819	Increase
6	2014	AWC Logo Cap	747	6570	-5823	Below Avg	18891	-18144	Decrease
7	2013	Bike Wash - Dissolver	6960	3636	3324	Above Avg	NULL	NULL	No Change
8	2014	Bike Wash - Dissolver	312	3636	-3324	Below Avg	6960	-6648	Decrease
9	2013	Classic Vest- L	11968	6240	5728	Above Avg	NULL	NULL	No Change

Query executed successfully.

```sql
/*Segment products into cost ranges and
count how many products fall into each segment*/
WITH product_segments AS (
    SELECT
        product_key,
        product_name,
        cost,
        CASE
            WHEN cost < 100 THEN 'Below 100'
            WHEN cost BETWEEN 100 AND 500 THEN '100-500'
            WHEN cost BETWEEN 500 AND 1000 THEN '500-1000'
            ELSE 'Above 1000'
        END AS cost_range
    FROM gold.dim_products
)
SELECT
    cost_range,
    COUNT(product_key) AS total_products
FROM product_segments
GROUP BY cost_range
ORDER BY total_products DESC;
```

136 %

Results | Messages

	cost_range	total_products
1	Below 100	110
2	100-500	101
3	500-1000	45
4	Above 1000	39

Query executed successfully.

```sql
/*Group customers into three segments based on their spending behavior:
    - VIP: Customers with at least 12 months of history and spending more than €5,000.
    - Regular: Customers with at least 12 months of history but spending €5,000 or less.
    - New: Customers with a lifespan less than 12 months.
And find the total number of customers by each group
*/
WITH customer_spending AS (
    SELECT
        c.customer_key,
        SUM(f.sales_amount) AS total_spending,
        MIN(order_date) AS first_order,
        MAX(order_date) AS last_order,
        DATEDIFF(month, MIN(order_date), MAX(order_date)) AS lifespan
    FROM gold.fact_sales f
    LEFT JOIN gold.dim_customers c
        ON f.customer_key = c.customer_key
    GROUP BY c.customer_key
)
SELECT
    customer_segment,
    COUNT(customer_key) AS total_customers
FROM (
    SELECT
        customer_key,
        CASE
            WHEN lifespan >= 12 AND total_spending > 5000 THEN 'VIP'
            WHEN lifespan >= 12 AND total_spending <= 5000 THEN 'Regular'
            ELSE 'New'
        END AS customer_segment
    FROM customer_spending
) AS segmented_customers
GROUP BY customer_segment
ORDER BY total_customers DESC;
```

112 %

Results | Messages

	customer_segment	total_customers
1	New	14631
2	Regular	2198
3	VIP	1655

Query executed successfully.

```sql
/*
===============================================================================
Part-to-Whole Analysis
===============================================================================
Purpose:
    - To compare performance or metrics across dimensions or time periods.
    - To evaluate differences between categories.
    - Useful for A/B testing or regional comparisons.

SQL Functions Used:
    - SUM(), AVG(): Aggregates values for comparison.
    - Window Functions: SUM() OVER() for total calculations.
===============================================================================
*/
-- Which categories contribute the most to overall sales?
WITH category_sales AS (
    SELECT
        p.category,
        SUM(f.sales_amount) AS total_sales
    FROM gold.fact_sales f
    LEFT JOIN gold.dim_products p
        ON p.product_key = f.product_key
    GROUP BY p.category
)
SELECT
    category,
    total_sales,
    SUM(total_sales) OVER () AS overall_sales,
    ROUND((CAST(total_sales AS FLOAT) / SUM(total_sales) OVER ()) * 100, 2) AS percentage_of_total
FROM category_sales
ORDER BY total_sales DESC;
```

112 %

Results | Messages

	category	total_sales	overall_sales	percentage_of_total
1	Bikes	28316272	29356250	96.46
2	Accessories	700262	29356250	2.39
3	Clothing	339716	29356250	1.16

Query executed successfully.

```
SELECT * FROM gold.report_customers
```

165 %

Results | Messages

	customer_key	customer_number	customer_name	age	age_group	customer_segment	last_order_date	recency	total_orders	total_sales	total_quantity	lifespan	avg_order_value	avg_monthly_spend
1	1	AW00011000	Jon Yang	54	50 and above	VIP	2013-05-03	151	3	8249	8	8	2749	294
2	2	AW00011001	Eugene Huang	49	40-49	VIP	2013-12-10	144	3	6384	11	10	2128	182
3	3	AW00011002	Ruben Torres	54	50 and above	VIP	2013-02-23	154	3	8114	4	4	2704	324
4	4	AW00011003	Christy Zhu	52	50 and above	VIP	2013-05-10	151	3	8139	9	9	2713	280
5	5	AW00011004	Elizabeth Johnson	46	40-49	VIP	2013-05-01	151	3	8196	6	6	2732	292
6	6	AW00011005	Julio Ruiz	49	40-49	VIP	2013-05-02	151	3	8121	6	6	2707	280
7	7	AW00011006	Janet Alvarez	49	40-49	VIP	2013-05-14	151	3	8119	5	5	2706	289
8	8	AW00011007	Marco Mehta	56	50 and above	VIP	2013-03-19	153	3	8211	8	8	2737	315
9	9	AW00011008	Rob Verhoff	50	50 and above	VIP	2013-03-02	153	3	8106	7	7	2702	311
10	10	AW00011009	Shannon Carlson	56	50 and above	VIP	2013-05-09	151	3	8091	5	5	2697	288
11	11	AW00011010	Jacquelyn Suarez	56	50 and above	VIP	2013-05-23	151	3	8088	4	4	2696	288
12	12	AW00011011	Curtis Lu	56	50 and above	VIP	2013-03-19	153	3	8133	4	4	2711	301
13	13	AW00011012	Lauren Walker	46	40-49	New	2013-10-15	146	2	81	5	5	40	11
14	14	AW00011013	Ian Jenkins	46	40-49	New	2014-01-21	143	2	114	5	5	57	12
15	15	AW00011014	Sydney Bennett	52	50 and above	New	2013-04-30	152	2	138	6	5	69	138
16	16	AW00011015	Chloe Young	41	40-49	New	2013-01-18	155	1	2501	3	3	2501	2501
17	17	AW00011016	Wyatt Hill	41	40-49	New	2013-02-09	154	1	2332	3	3	2332	2332
18	18	AW00011017	Shannon Wang	76	50 and above	VIP	2013-10-14	146	3	6434	4	4	2144	194
19	19	AW00011018	Clarence Rai	70	50 and above	VIP	2013-10-24	146	3	6533	7	7	2177	197
20	20	AW00011019	Luke Lal	42	40-49	New	2014-01-12	143	17	880	33	20	51	80
21	21	AW00011020	Jordan King	41	40-49	New	2012-12-29	156	1	2317	2	2	2317	2317
22	22	AW00011021	Destiny Wilson	41	40-49	New	2013-01-23	155	1	2372	3	3	2372	2372
23	23	AW00011022	Ethan Zhang	41	40-49	New	2013-01-20	155	1	2322	2	2	2322	2322
24	24	AW00011023	Seth Edwards	41	40-49	New	2014-01-14	143	2	122	6	6	61	11
25	25	AW00011024	Russell Xie	41	40-49	New	2013-07-26	149	2	56	6	5	28	56
26	26	AW00011025	Alejandro Beck	74	50 and above	VIP	2013-10-25	146	3	6577	6	6	2192	199
27	27	AW00011026	Harold Sai	74	50 and above	VIP	2013-10-15	146	3	6575	7	7	2191	199
28	28	AW00011027	Jessie Zhao	73	50 and above	VIP	2013-10-24	146	3	6591	9	9	2197	199
29	29	AW00011028	Jill Jimenez	74	50 and above	VIP	2013-10-07	146	3	6474	5	5	2158	196
30	30	AW00011029	Jimmy Moreno	73	50 and above	VIP	2013-11-11	145	3	6565	7	7	2188	193
31	31	AW00011030	Bethany Yuan	67	50 and above	VIP	2013-11-09	145	3	6471	4	4	2157	196
32	32	AW00011031	Theresa Ramos	72	50 and above	VIP	2013-11-13	145	3	6478	6	6	2159	196
33	33	AW00011032	Denise Stone	73	50 and above	VIP	2013-11-08	145	3	6525	10	10	2175	197
34	34	AW00011033	Jaime Nath	67	50 and above	VIP	2013-11-05	145	3	6495	7	7	2165	196
35	35	AW00011034	Ebony Gonzalez	73	50 and above	VIP	2013-11-10	145	3	6491	4	4	2163	196

Query executed successfully. LAPTOP-T4K7IGBT\SQLEXPRESS ... | LAPTOP-T4K7IGBT

```
SELECT * FROM gold.report_products
```

165 %

Results | Messages

	product_key	product_name	category	subcategory	cost	cost_range	product_segment	last_sale_date	receny_in_months	total_orders	total_sales	total_quantity	total_customers	lifespan	avg_selling_price	avg_order_revenue	avg_monthly_revenue
1	3	Mountain-100 Black- 38	Bikes	Mountain Bikes	1898	Above 1000	High-Performer	2011-12-27	168	49	165375	49	49	11	3375	3375	15034
2	4	Mountain-100 Black- 42	Bikes	Mountain Bikes	1898	Above 1000	High-Performer	2011-12-27	168	45	151875	45	45	11	3375	3375	13806
3	5	Mountain-100 Black- 44	Bikes	Mountain Bikes	1898	Above 1000	High-Performer	2011-12-21	168	60	202500	60	60	11	3375	3375	18409
4	6	Mountain-100 Black- 48	Bikes	Mountain Bikes	1898	Above 1000	High-Performer	2011-12-26	168	57	192375	57	57	12	3375	3375	16031
5	7	Mountain-100 Silver- 38	Bikes	Mountain Bikes	1912	Above 1000	High-Performer	2011-12-22	168	58	197200	58	58	12	3400	3400	16433
6	8	Mountain-100 Silver- 42	Bikes	Mountain Bikes	1912	Above 1000	High-Performer	2011-12-28	168	42	142800	42	42	11	3400	3400	12981
7	9	Mountain-100 Silver- 44	Bikes	Mountain Bikes	1912	Above 1000	High-Performer	2011-12-12	168	49	166600	49	49	12	3400	3400	13883
8	10	Mountain-100 Silver- 48	Bikes	Mountain Bikes	1912	Above 1000	High-Performer	2011-12-23	168	36	122400	36	36	11	3400	3400	11127
9	16	Road-150 Red- 44	Bikes	Road Bikes	2171	Above 1000	High-Performer	2011-12-28	168	281	1005418	281	281	12	3578	3578	83784
10	17	Road-150 Red- 48	Bikes	Road Bikes	2171	Above 1000	High-Performer	2011-12-28	168	337	1205786	337	337	12	3578	3578	100482
11	18	Road-150 Red- 52	Bikes	Road Bikes	2171	Above 1000	High-Performer	2011-12-27	168	302	1080556	302	302	12	3578	3578	90046
12	19	Road-150 Red- 56	Bikes	Road Bikes	2171	Above 1000	High-Performer	2011-12-27	168	295	1055510	295	295	12	3578	3578	87959
13	20	Road-150 Red- 62	Bikes	Road Bikes	2171	Above 1000	High-Performer	2011-12-28	168	336	1202208	336	336	12	3578	3578	100184
14	36	Road-650 Black- 44	Bikes	Road Bikes	487	100-500	High-Performer	2012-12-26	156	63	47565	63	63	23	755	755	2068
15	37	Road-650 Black- 48	Bikes	Road Bikes	487	100-500	High-Performer	2012-12-25	156	60	45552	60	60	21	759.2	759	2169
16	38	Road-650 Black- 52	Bikes	Road Bikes	487	100-500	High-Performer	2012-12-19	156	89	66915	89	89	23	751.9	751	2909
17	39	Road-650 Black- 58	Bikes	Road Bikes	487	100-500	High-Performer	2012-12-18	156	76	57996	76	76	23	763.1	763	2521
18	40	Road-650 Black- 60	Bikes	Road Bikes	487	100-500	High-Performer	2012-12-12	156	76	57156	76	76	22	752.1	752	2598
19	41	Road-650 Black- 62	Bikes	Road Bikes	487	100-500	High-Performer	2012-12-18	156	65	49047	65	65	24	754.6	754	2043
20	42	Road-650 Red- 44	Bikes	Road Bikes	487	100-500	High-Performer	2012-12-25	156	72	54528	72	72	23	757.3	757	2370
21	43	Road-650 Red- 48	Bikes	Road Bikes	487	100-500	High-Performer	2012-12-27	156	88	66720	88	88	23	758.2	758	2900
22	44	Road-650 Red- 52	Bikes	Road Bikes	487	100-500	High-Performer	2012-12-21	156	61	46083	61	61	24	755.5	755	1920
23	45	Road-650 Red- 58	Bikes	Road Bikes	487	100-500	High-Performer	2012-12-18	156	74	56346	74	74	22	761.4	761	2561
24	46	Road-650 Red- 60	Bikes	Road Bikes	487	100-500	High-Performer	2012-12-23	156	53	40071	53	53	23	756.1	756	1742
25	47	Road-650 Red- 62	Bikes	Road Bikes	487	100-500	High-Performer	2012-12-05	156	75	57381	75	75	23	765.1	765	2494
26	48	Road-250 Red- 44	Bikes	Road Bikes	1519	Above 1000	High-Performer	2012-12-25	156	144	351792	144	144	12	2443	2443	29316
27	49	Road-250 Red- 48	Bikes	Road Bikes	1519	Above 1000	High-Performer	2012-12-24	156	162	395766	162	162	12	2443	2443	32980
28	50	Road-250 Red- 52	Bikes	Road Bikes	1519	Above 1000	High-Performer	2012-12-25	156	133	324919	133	133	12	2443	2443	27076
29	104	Mountain Bottle Cage	Accessories	Bottles and Cages	4	Below 100	High-Performer	2014-01-28	143	2025	20340	2034	2004	13	10	10	1564
30	105	Road Bottle Cage	Accessories	Bottles and Cages	3	Below 100	High-Performer	2014-01-25	143	1711	15399	1711	1699	13	9	9	1184