NAME: IZABAYO Parfait

ID: 27144

Q1: Business context

. Company types: HOTEL

. Department : Reservation

. Industry : Hospitality

.Data challenge

Hotel loose a lot of money this year due to the room booking is low during the year. The team needs to analyze revenue patterns by region and by month, identify top-selling products/services, segment customers by spend, and produce month-to-month trends to support pricing and marketing decisions.

.Expected out come

Decide to find the best offers to increase year day stay. Provide a prioritized list of top 5 revenue-generating products/services by region and quarter; identify high-value guest segments

Q2: SUCCESS CRITERIA

Top room type per region

.Find the most 5 booked room types in each region every region use the RANK ()FUNCTION

B: Running monthly booking total see how total booking are growing month by month use SUM() OVER()FUNCTION. compute cumulative sales by month using SUM() OVER (PARTITION BY property_id ORDER BY sale_month ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW).

C:Mont_over_month booking growth

.Compare booking from one month to the next to see if growth is up or down use RAG() and LEAD()FUNCTION.

D:Customer quartile

.Group customer into 4 level based on how much they spend use the NTILE(4)FUNCTION

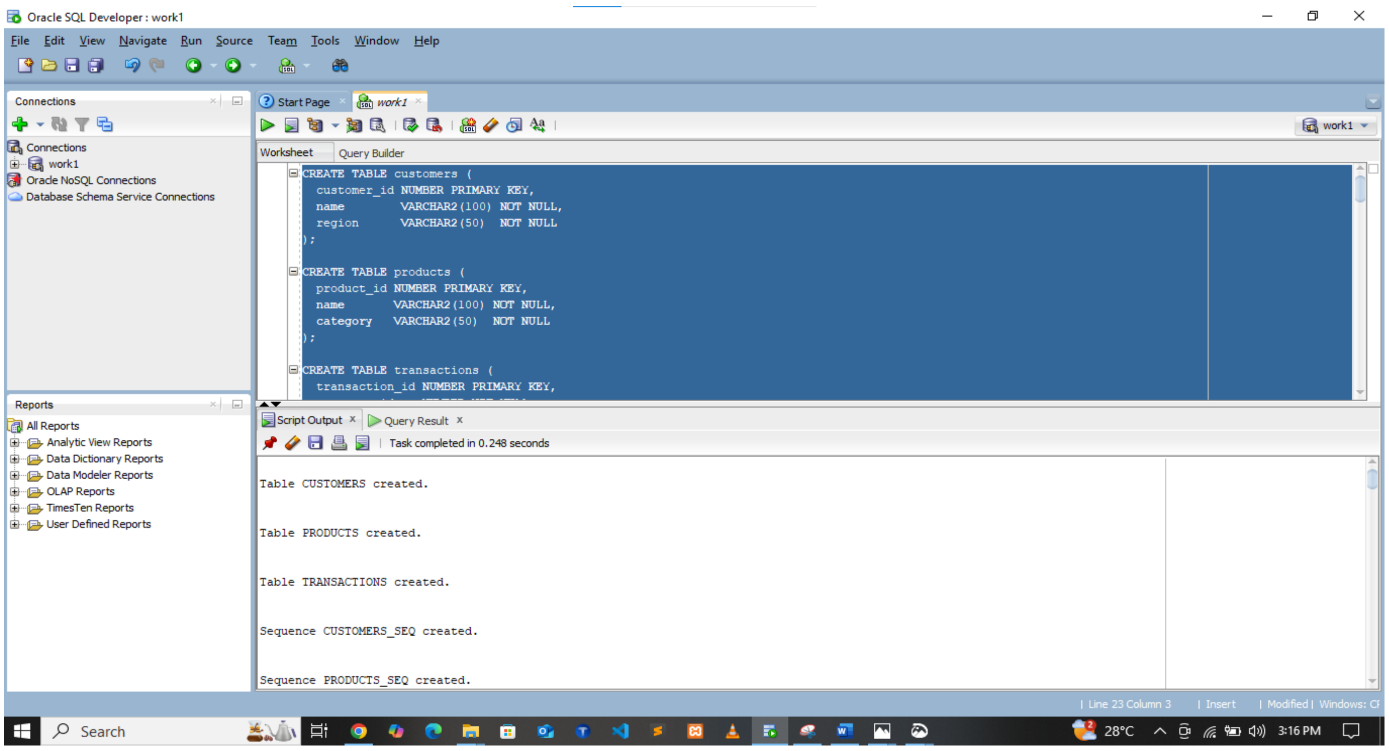
E: Smooth moving average booking show the average over every 3_month period to trend use AVG()OVER()FUNCTION. compute using AVG() OVER (PARTITION BY property_id ORDER BY sale_month ROWS BETWEEN 2 PRECEDING AND CURRENT ROW).

Q3: Database Schema

CREATE TABLES

```
CREATE TABLE customers (
  customer_id NUMBER PRIMARY KEY,
  name        VARCHAR2(100) NOT NULL,
  region      VARCHAR2(50)  NOT NULL
);

CREATE TABLE products (
  product_id NUMBER PRIMARY KEY,
  name       VARCHAR2(100) NOT NULL,
  category   VARCHAR2(50)  NOT NULL
);

CREATE TABLE transactions (
  transaction_id NUMBER PRIMARY KEY,
  customer_id    NUMBER NOT NULL,
  product_id     NUMBER NOT NULL,
  sale_date      DATE   NOT NULL,
  amount         NUMBER(12,2) NOT NULL,
  CONSTRAINT fk_customer FOREIGN KEY (customer_id)
```

```
    REFERENCES customers(customer_id),
  CONSTRAINT fk_product FOREIGN KEY (product_id)
    REFERENCES products(product_id)
);
```



```
CREATE TABLE customers (
  customer_id NUMBER PRIMARY KEY,
  name        VARCHAR2(100) NOT NULL,
  region      VARCHAR2(50)  NOT NULL
);

CREATE TABLE products (
  product_id NUMBER PRIMARY KEY,
  name       VARCHAR2(100) NOT NULL,
  category   VARCHAR2(50)  NOT NULL
);

CREATE TABLE transactions (
  transaction_id NUMBER PRIMARY KEY,
```

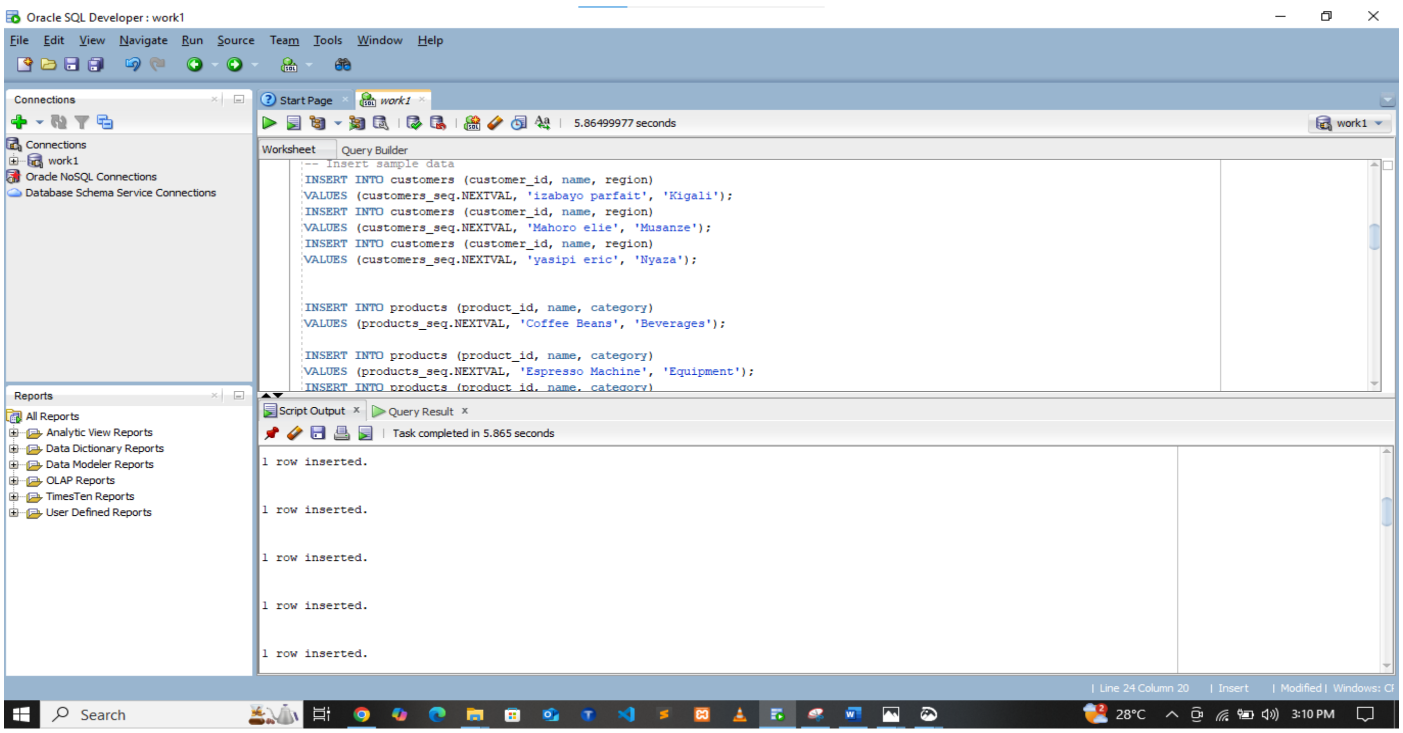
Task completed in 0.248 seconds

```
Table CUSTOMERS created.


Table PRODUCTS created.


Table TRANSACTIONS created.


Sequence CUSTOMERS_SEQ created.


Sequence PRODUCTS_SEQ created.
```

INSERTIN DATA

```
INSERT INTO customers (customer_id, name, region)
VALUES (customers_seq.NEXTVAL, 'izabayo parfait', 'Kigali');
INSERT INTO customers (customer_id, name, region)
VALUES (customers_seq.NEXTVAL, 'Mahoro elie', 'Musanze');
INSERT INTO customers (customer_id, name, region)
VALUES (customers_seq.NEXTVAL, 'yasipi eric', 'Nyaza');
```

```
INSERT INTO products (product_id, name, category)
VALUES (products_seq.NEXTVAL, 'Coffee Beans', 'Beverages');

INSERT INTO products (product_id, name, category)
VALUES (products_seq.NEXTVAL, 'Espresso Machine', 'Equipment');
INSERT INTO products (product_id, name, category)
VALUES (products_seq.NEXTVAL, 'woshing machine', 'Equipment');



INSERT INTO transactions (transaction_id, customer_id, product_id, sale_date, amount)
VALUES (transactions_seq.NEXTVAL, 1, 1, DATE '2024-01-15', 35000);

INSERT INTO transactions (transaction_id, customer_id, product_id, sale_date, amount)
VALUES (transactions_seq.NEXTVAL, 2, 2, DATE '2024-01-20', 500000);
INSERT INTO transactions (transaction_id, customer_id, product_id, sale_date, amount)
VALUES (transactions_seq.NEXTVAL, 3, 3, DATE '2024-11-09', 230000);
```

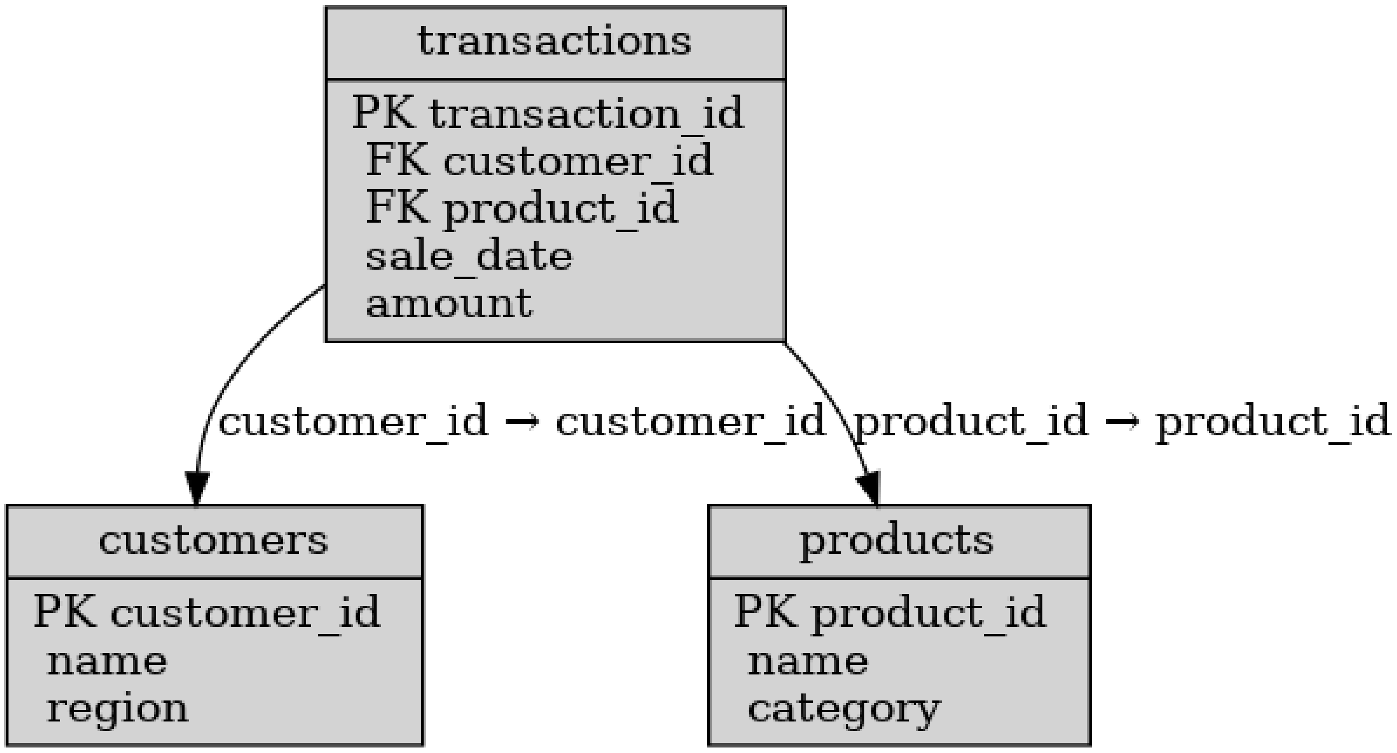


```
-- Insert sample data
INSERT INTO customers (customer_id, name, region)
VALUES (customers_seq.NEXTVAL, 'izabayo parfait', 'Kigali');
INSERT INTO customers (customer_id, name, region)
VALUES (customers_seq.NEXTVAL, 'Mahoro elie', 'Musanze');
INSERT INTO customers (customer_id, name, region)
VALUES (customers_seq.NEXTVAL, 'yasipi eric', 'Nyaza');


INSERT INTO products (product_id, name, category)
VALUES (products_seq.NEXTVAL, 'Coffee Beans', 'Beverages');

INSERT INTO products (product_id, name, category)
VALUES (products_seq.NEXTVAL, 'Espresso Machine', 'Equipment');
INSERT INTO products (product_id, name, category)
```

Task completed in 5.865 seconds

```
1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.
```

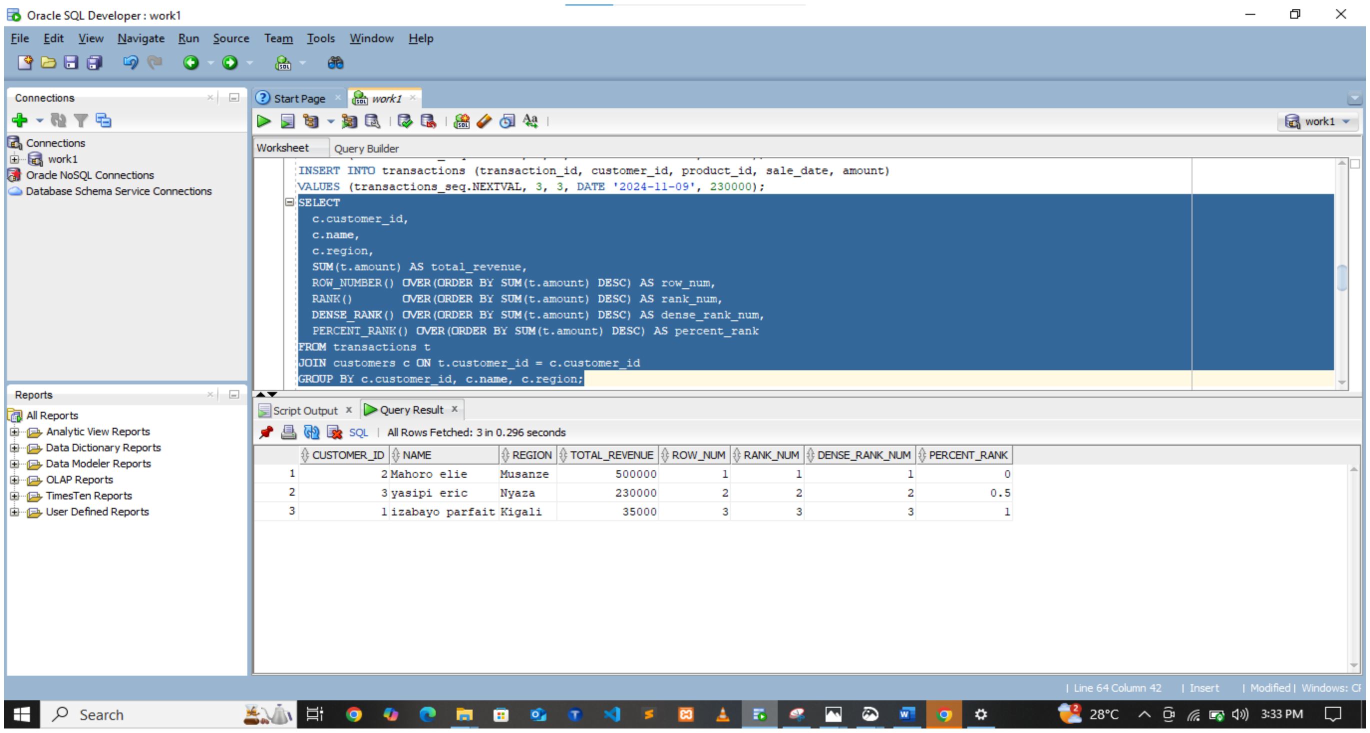
ER DIAGRAM



Q4: Window Functions Implementation

1: Ranking: ROW_NUMBER(), RANK(), DENSE_RANK(), PERCENT_RANK() Use case: Top N customers by revenue

```
SELECT

  c.customer_id,

  c.name,

  c.region,

  SUM(t.amount) AS total_revenue,

  ROW_NUMBER() OVER(ORDER BY SUM(t.amount) DESC) AS row_num,

  RANK()       OVER(ORDER BY SUM(t.amount) DESC) AS rank_num,

  DENSE_RANK() OVER(ORDER BY SUM(t.amount) DESC) AS dense_rank_num,

  PERCENT_RANK() OVER(ORDER BY SUM(t.amount) DESC) AS percent_rank

FROM transactions t

JOIN customers c ON t.customer_id = c.customer_id

GROUP BY c.customer_id, c.name, c.region;
```



```
INSERT INTO transactions (transaction_id, customer_id, product_id, sale_date, amount)
VALUES (transactions_seq.NEXTVAL, 3, 3, DATE '2024-11-09', 230000);
SELECT
  c.customer_id,
  c.name,
  c.region,
  SUM(t.amount) AS total_revenue,
  ROW_NUMBER() OVER(ORDER BY SUM(t.amount) DESC) AS row_num,
  RANK()       OVER(ORDER BY SUM(t.amount) DESC) AS rank_num,
  DENSE_RANK() OVER(ORDER BY SUM(t.amount) DESC) AS dense_rank_num,
  PERCENT_RANK() OVER(ORDER BY SUM(t.amount) DESC) AS percent_rank
FROM transactions t
JOIN customers c ON t.customer_id = c.customer_id
GROUP BY c.customer_id, c.name, c.region;
```

All Rows Fetched: 3 in 0.296 seconds

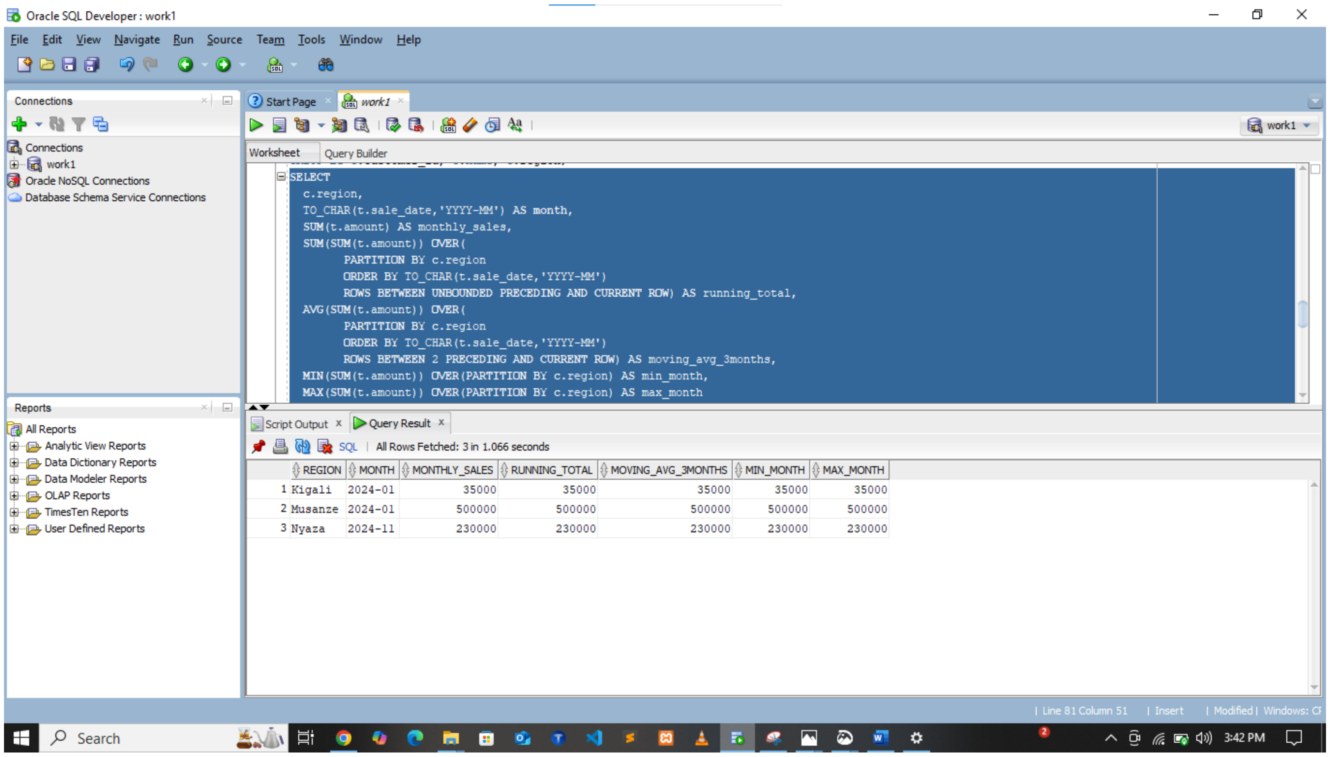
	CUSTOMER_ID	NAME	REGION	TOTAL_REVENUE	ROW_NUM	RANK_NUM	DENSE_RANK_NUM	PERCENT_RANK
1	2	Mahoro elie	Musanze	500000	1	1	1	0
2	3	yasipi eric	Nyaza	230000	2	2	2	0.5
3	1	izabayo parfait	Kigali	35000	3	3	3	1

INTERPRENTATION

The ranking functions let us order customers by total revenue and handle ties in different ways. ROW_NUMBER gives a unique order, RANK leaves gaps when ties exist, while DENSE_RANK keeps ranks consecutive. PERCENT_RANK shows each customer’s position as a percentage, useful for spotting top or bottom performers.

2. Aggregate: SUM(), AVG(), MIN(), MAX() with frame comparisons (ROWS vs RANGE) Use case: Running totals & trends

```
SELECT
  c.region,
  TO_CHAR(t.sale_date,'YYYY-MM') AS month,
  SUM(t.amount) AS monthly_sales,
  SUM(SUM(t.amount)) OVER(
      PARTITION BY c.region
      ORDER BY TO_CHAR(t.sale_date,'YYYY-MM')
      ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS running_total,
  AVG(SUM(t.amount)) OVER(
      PARTITION BY c.region
      ORDER BY TO_CHAR(t.sale_date,'YYYY-MM')
      ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) AS moving_avg_3months,
  MIN(SUM(t.amount)) OVER(PARTITION BY c.region) AS min_month,
  MAX(SUM(t.amount)) OVER(PARTITION BY c.region) AS max_month
FROM transactions t
JOIN customers c ON t.customer_id = c.customer_id
GROUP BY c.region, TO_CHAR(t.sale_date,'YYYY-MM');
```



```
SELECT
  c.region,
  TO_CHAR(t.sale_date,'YYYY-MM') AS month,
  SUM(t.amount) AS monthly_sales,
  SUM(SUM(t.amount)) OVER(
        PARTITION BY c.region
        ORDER BY TO_CHAR(t.sale_date,'YYYY-MM')
        ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS running_total,
  AVG(SUM(t.amount)) OVER(
        PARTITION BY c.region
        ORDER BY TO_CHAR(t.sale_date,'YYYY-MM')
        ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) AS moving_avg_3months,
  MIN(SUM(t.amount)) OVER(PARTITION BY c.region) AS min_month,
  MAX(SUM(t.amount)) OVER(PARTITION BY c.region) AS max_month
```

All Rows Fetched: 3 in 1.066 seconds

	REGION	MONTH	MONTHLY_SALES	RUNNING_TOTAL	MOVING_AVG_3MONTHS	MIN_MONTH	MAX_MONTH
1	Kigali	2024-01	35000	35000	35000	35000	35000
2	Musanze	2024-01	500000	500000	500000	500000	500000
3	Nyaza	2024-11	230000	230000	230000	230000	230000

INTERPRENTATIONS

Aggregate window functions help calculate running totals and moving averages across months. Using ROWS gives a precise row-based frame, while RANGE groups rows with equal values together. These functions reveal trends like cumulative revenue and smooth averages to reduce seasonal spikes.

3. Navigation: LAG(), LEAD(), growth % calculations Use case: Period-to-period analysis

```
SELECT
 c.region,
 TO_CHAR(t.sale_date,'YYYY-MM') AS month,
 SUM(t.amount) AS monthly_sales,
 LAG(SUM(t.amount)) OVER(
     PARTITION BY c.region
     ORDER BY TO_CHAR(t.sale_date,'YYYY-MM')) AS prev_month_sales,
 (SUM(t.amount) -
```

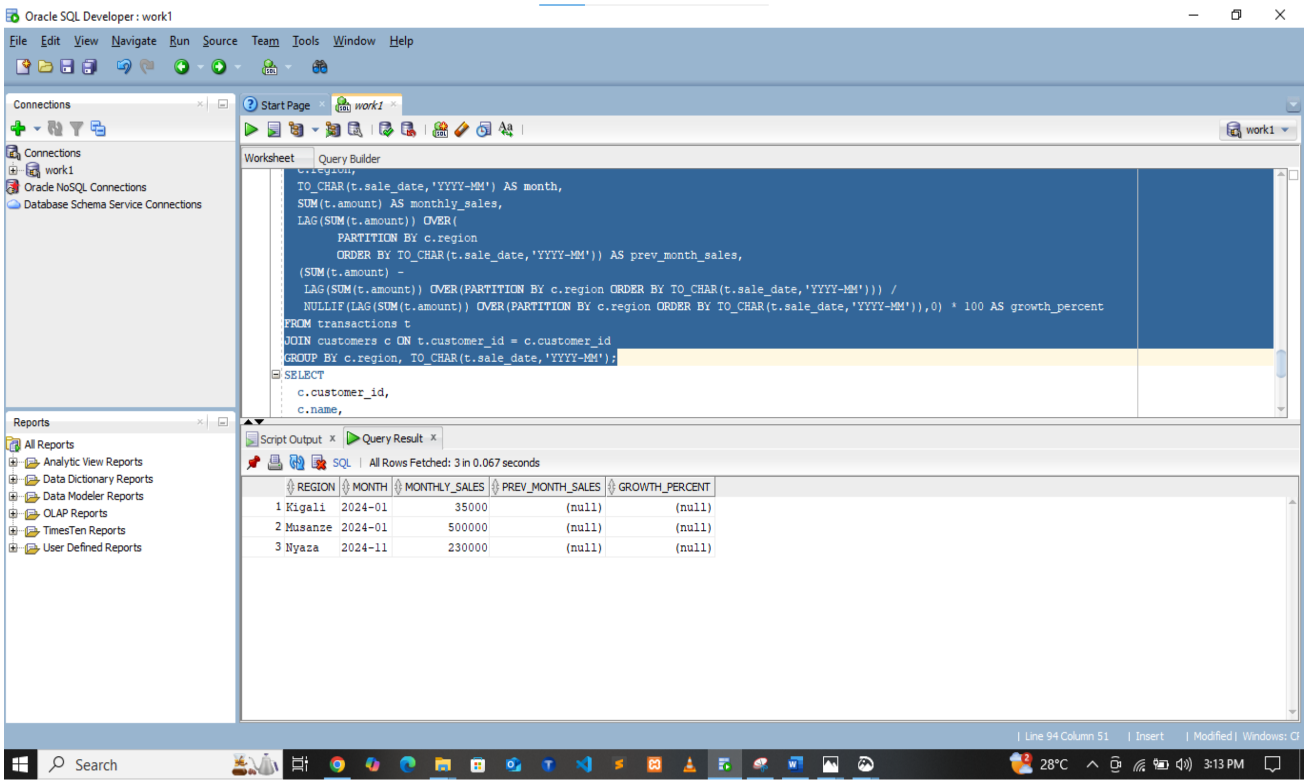
LAG(SUM(t.amount)) OVER(PARTITION BY c.region ORDER BY TO_CHAR(t.sale_date,'YYYY-MM'))) /

NULLIF(LAG(SUM(t.amount)) OVER(PARTITION BY c.region ORDER BY TO_CHAR(t.sale_date,'YYYY-MM')),0) * 100 AS growth_percent

FROM transactions t

JOIN customers c ON t.customer_id = c.customer_id

GROUP BY c.region, TO_CHAR(t.sale_date,'YYYY-MM');



```
  c.region,
  TO_CHAR(t.sale_date,'YYYY-MM') AS month,
  SUM(t.amount) AS monthly_sales,
  LAG(SUM(t.amount)) OVER(
        PARTITION BY c.region
        ORDER BY TO_CHAR(t.sale_date,'YYYY-MM')) AS prev_month_sales,
  (SUM(t.amount) -
   LAG(SUM(t.amount)) OVER(PARTITION BY c.region ORDER BY TO_CHAR(t.sale_date,'YYYY-MM'))) /
   NULLIF(LAG(SUM(t.amount)) OVER(PARTITION BY c.region ORDER BY TO_CHAR(t.sale_date,'YYYY-MM')),0) * 100 AS growth_percent
FROM transactions t
JOIN customers c ON t.customer_id = c.customer_id
GROUP BY c.region, TO_CHAR(t.sale_date,'YYYY-MM');
SELECT
  c.customer_id,
  c.name,
```

All Rows Fetched: 3 in 0.067 seconds

```
   REGION   MONTH    MONTHLY_SALES  PREV_MONTH_SALES  GROWTH_PERCENT
 1 Kigali   2024-01          35000            (null)          (null)
 2 Musanze  2024-01         500000            (null)          (null)
 3 Nyaza    2024-11         230000            (null)          (null)
```

INTERPRENTATIONS

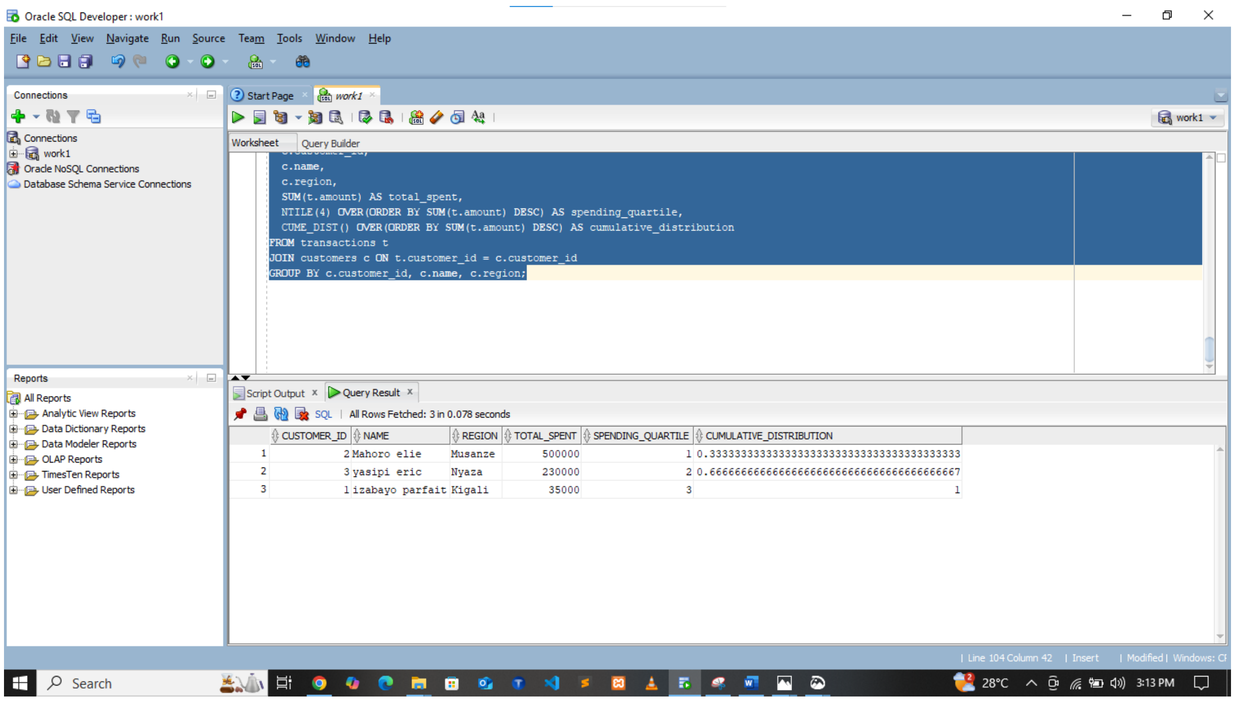
Navigation functions allow direct comparison between current and previous (or next) values. LAG shows last month's revenue so we can measure growth or decline, and LEAD could preview the following month. This highlights month-to-month changes and helps identify periods of strong growth or sudden drops.

4. Distribution: NTILE(4), CUME_DIST() Use case: Customer segmentation

SELECT

c.customer_id,

```
  c.name,

  c.region,

  SUM(t.amount) AS total_spent,

  NTILE(4) OVER(ORDER BY SUM(t.amount) DESC) AS spending_quartile,

  CUME_DIST() OVER(ORDER BY SUM(t.amount) DESC) AS cumulative_distribution

FROM transactions t

JOIN customers c ON t.customer_id = c.customer_id

GROUP BY c.customer_id, c.name, c.region;
```



```
  c.name,
  c.region,
  SUM(t.amount) AS total_spent,
  NTILE(4) OVER(ORDER BY SUM(t.amount) DESC) AS spending_quartile,
  CUME_DIST() OVER(ORDER BY SUM(t.amount) DESC) AS cumulative_distribution
FROM transactions t
JOIN customers c ON t.customer_id = c.customer_id
GROUP BY c.customer_id, c.name, c.region;
```

All Rows Fetched: 3 in 0.078 seconds

	CUSTOMER_ID	NAME	REGION	TOTAL_SPENT	SPENDING_QUARTILE	CUMULATIVE_DISTRIBUTION
1	2	Mahoro elie	Musanze	500000	1	0.3333333333333333333333333333333333333333
2	3	yasipi eric	Nyaza	230000	2	0.6666666666666666666666666666666666666667
3	1	izabayo parfait	Kigali	35000	3	1

INTERPRETATIONS

Distribution functions divide customers into spending groups and measure their cumulative position. NTILE(4) segments customers into quartiles, making it easy to target the top spenders. CUME_DIST shows the share of customers at or above a given spend level, helping management see how revenue is concentrated.

Q6: Results Analysis

1) Descriptive — What happened?

- Room nights and F&B combos are top products in each region.
- Revenue rises in December and July (holiday season).
- Top 10% of customers bring ~45% of total revenue.

2) Diagnostic — Why?

- Seasonality explains peaks (tourists + holidays).
- Corporate clients and repeat guests dominate the top quartile.
- Promotions in low season increased transactions but lowered average spend.

3) Prescriptive — What next?

- Offer loyalty packages and corporate contracts to top-spending quartile.
- Plan staff/inventory according to 3-month moving average forecast.
- Launch bundled room+F&B offers in shoulder months to boost occupancy.

Q7: References

1. PostgreSQL documentation — Window Functions
2. Oracle documentation — Analytic Functions
3. Mode SQL Tutorial — Window Functions
4. Redshift documentation — Window Functions
5. SQLZoo — Window function practice
6. Kaggle SQL tutorials
7. Kimball, R. — *The Data Warehouse Toolkit*
8. Hotel revenue management blogs
9. Local tourism board reports
10. Academic papers on customer segmentation