**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**

**A: T**op 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,

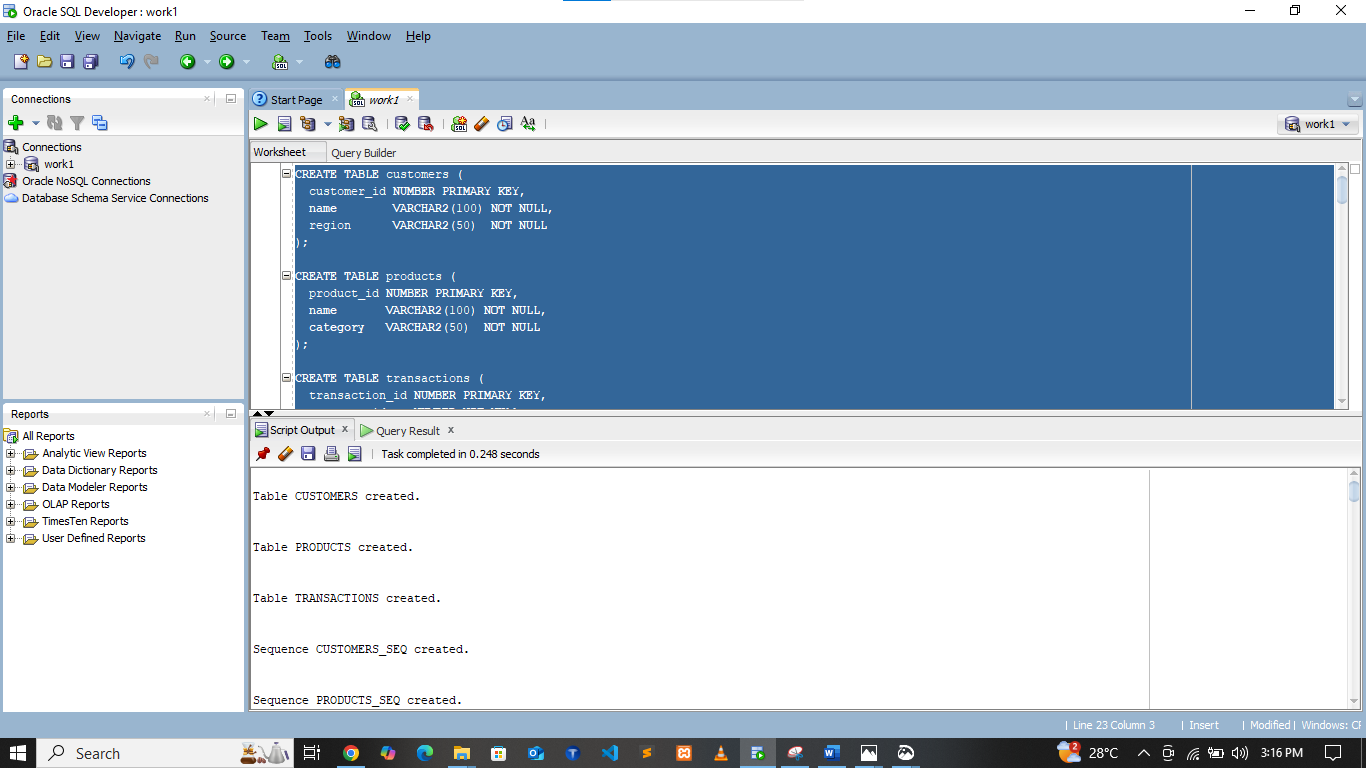
CONSTRAINT fk\_customer FOREIGN KEY (customer\_id)

REFERENCES customers(customer\_id),

CONSTRAINT fk\_product FOREIGN KEY (product\_id)

REFERENCES products(product\_id)

);



**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)

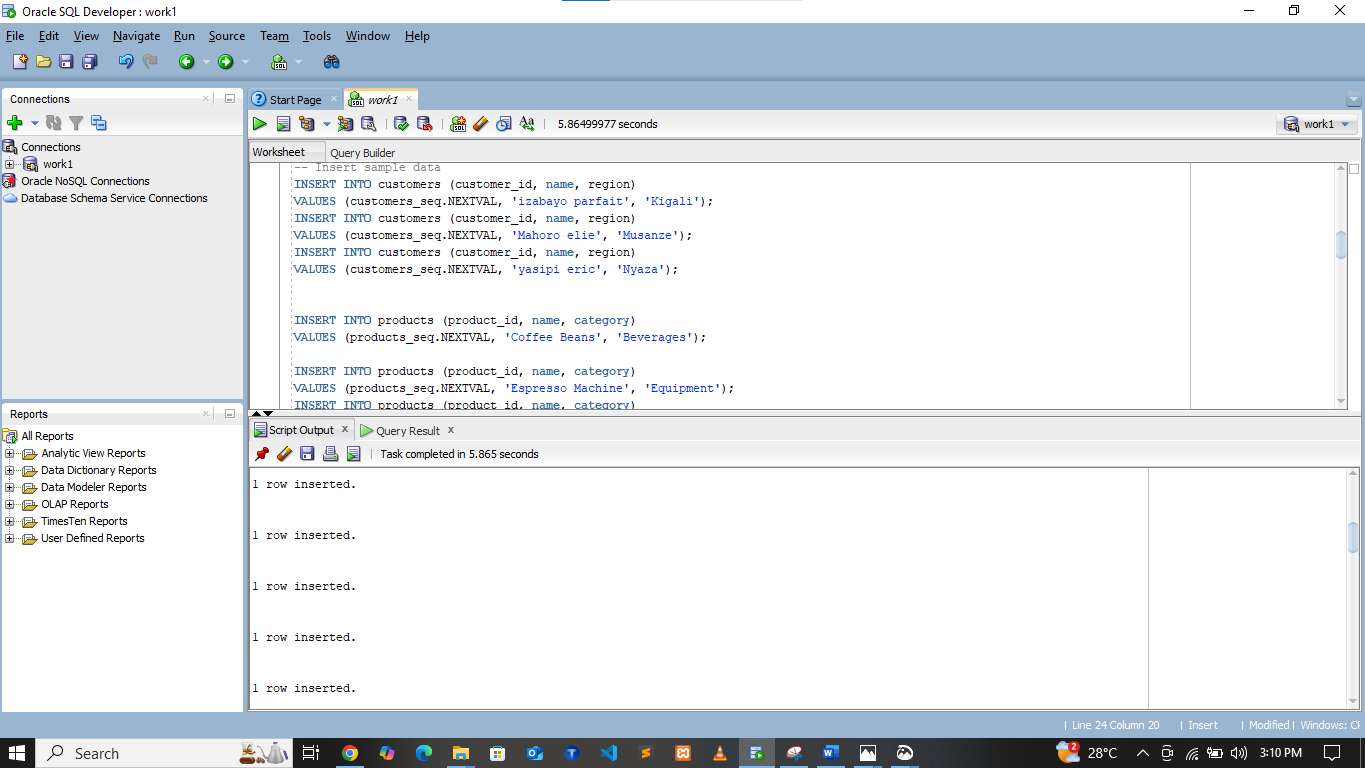
VALUES (transactions\_seq.NEXTVAL, 1, 1, DATE '2024-01-15', 35000);

INSERT INTO transactions (transaction\_id, customer\_id, product\_id, sale\_date, amount)

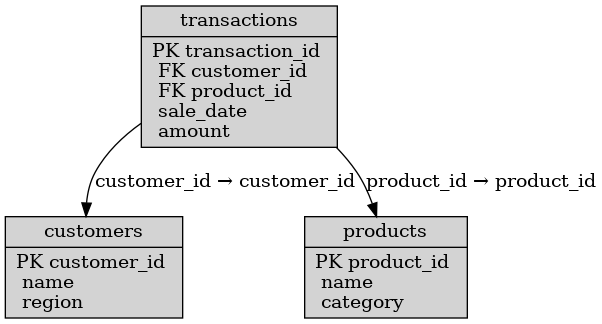
VALUES (transactions\_seq.NEXTVAL, 2, 2, DATE '2024-01-20', 500000);

INSERT INTO transactions (transaction\_id, customer\_id, product\_id, sale\_date, amount)

VALUES (transactions\_seq.NEXTVAL, 3, 3, DATE '2024-11-09', 230000);



**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,

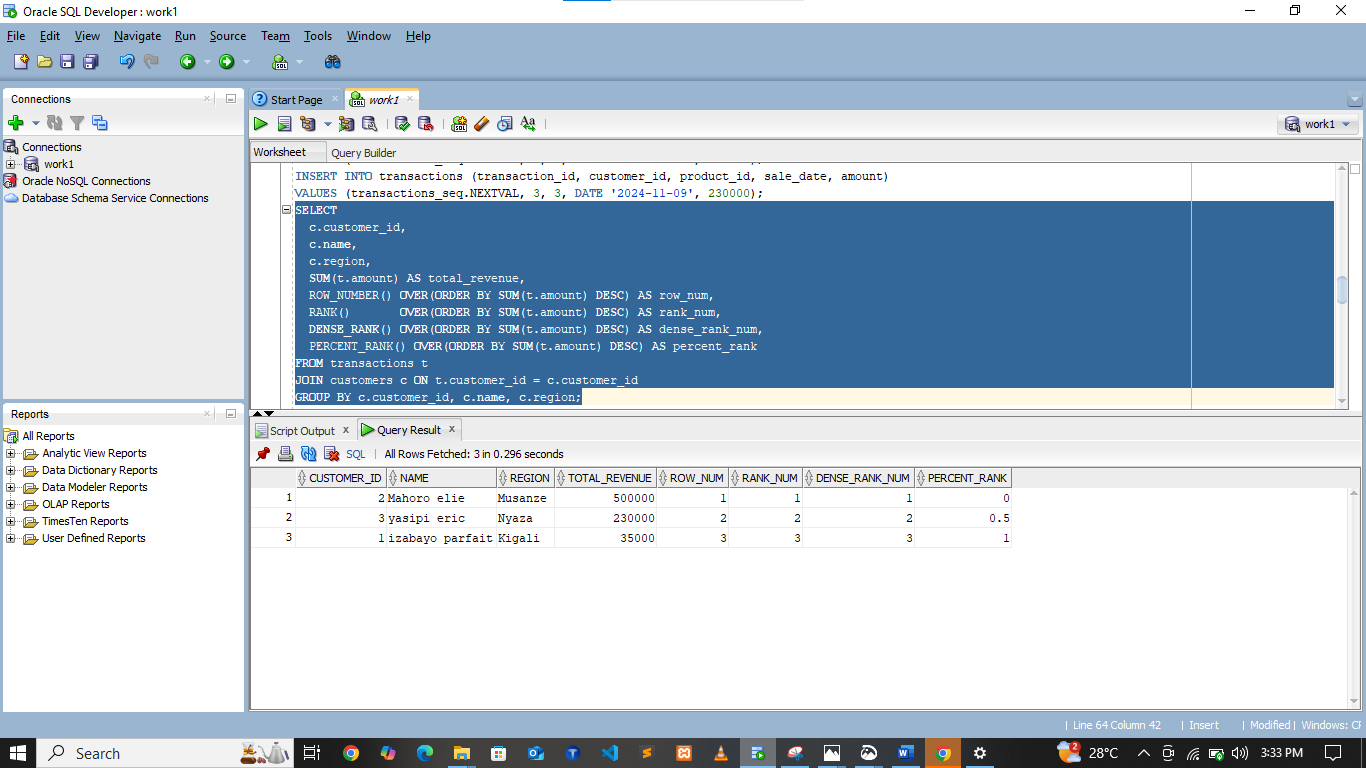
DENSE\_RANK() OVER(ORDER BY SUM(t.amount) DESC) AS dense\_rank\_num,

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;



**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,

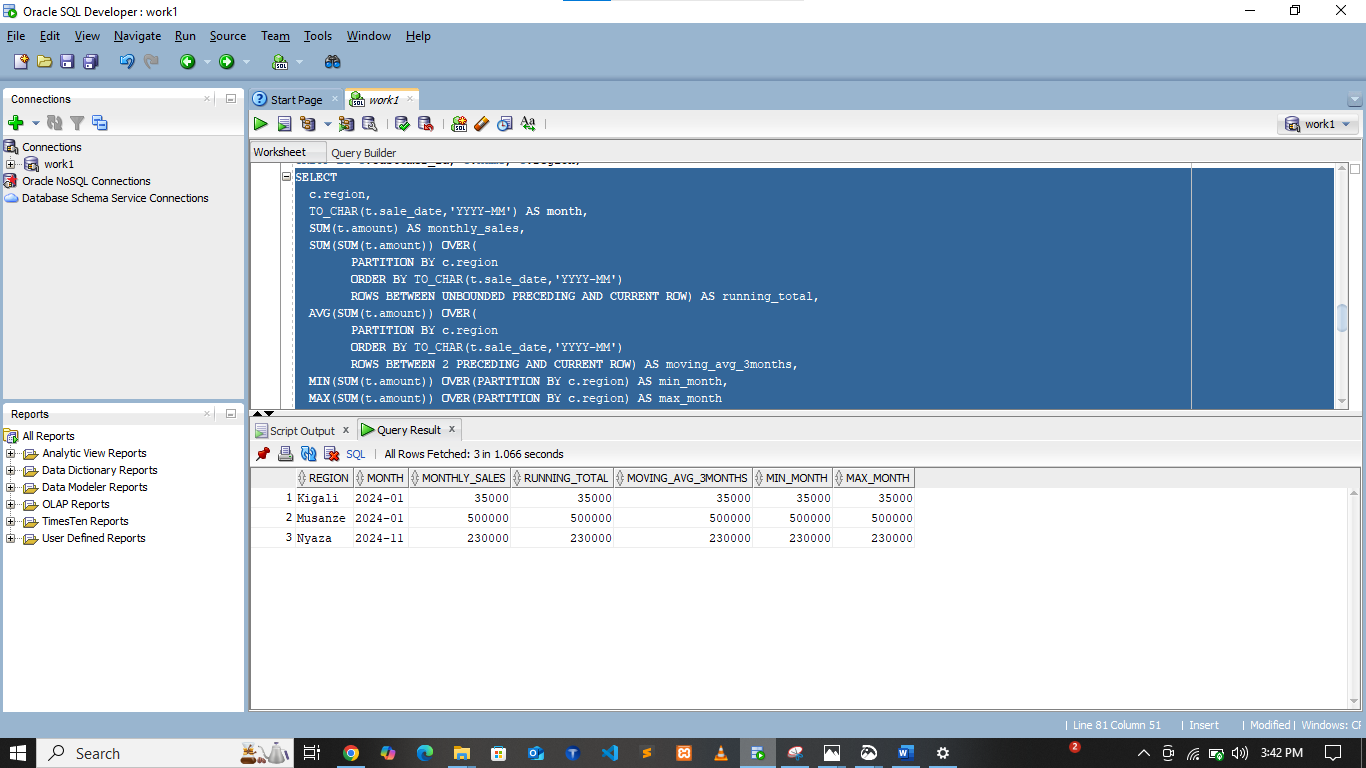
MIN(SUM(t.amount)) OVER(PARTITION BY c.region) AS min\_month,

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');



**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) -

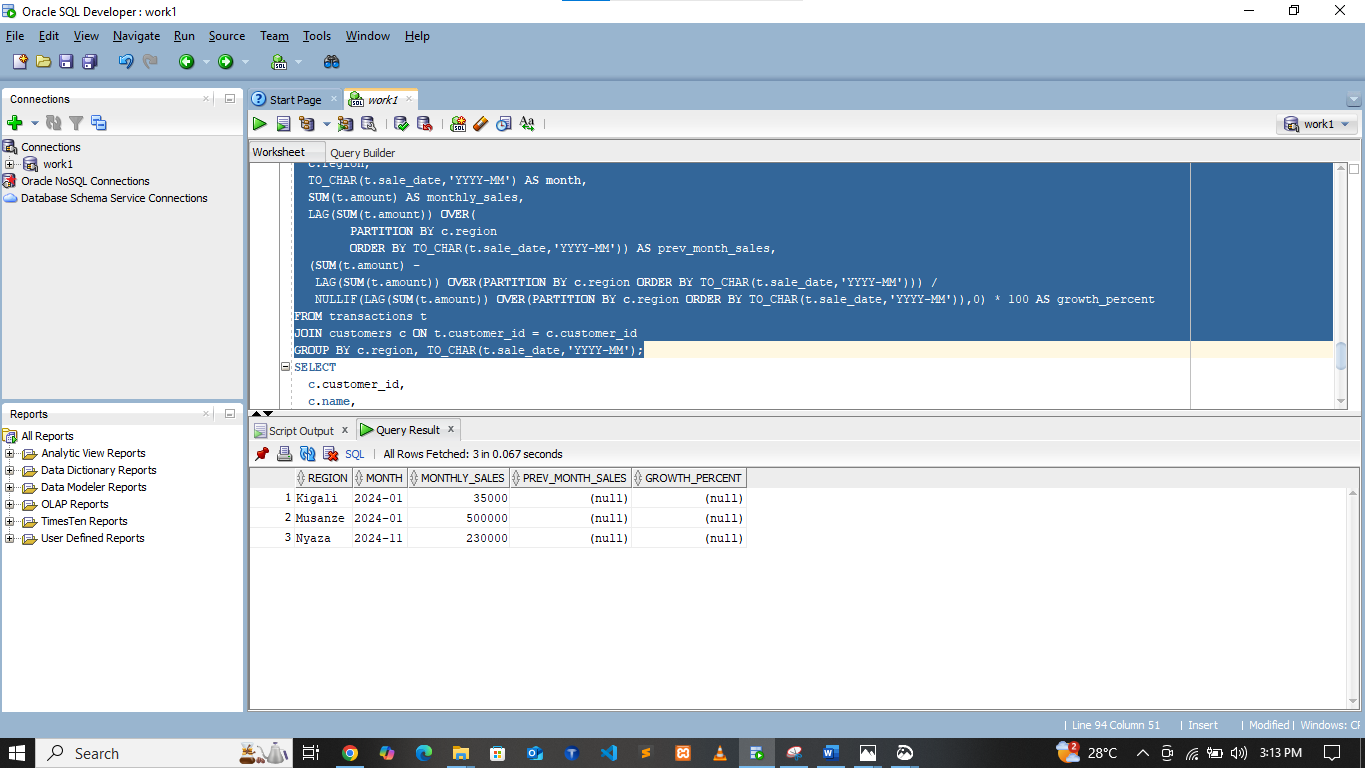
LAG(SUM(t.amount)) OVER(PARTITION BY c.region ORDER BY TO\_CHAR(t.sale\_date,'YYYY-MM'))) /

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');



**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,

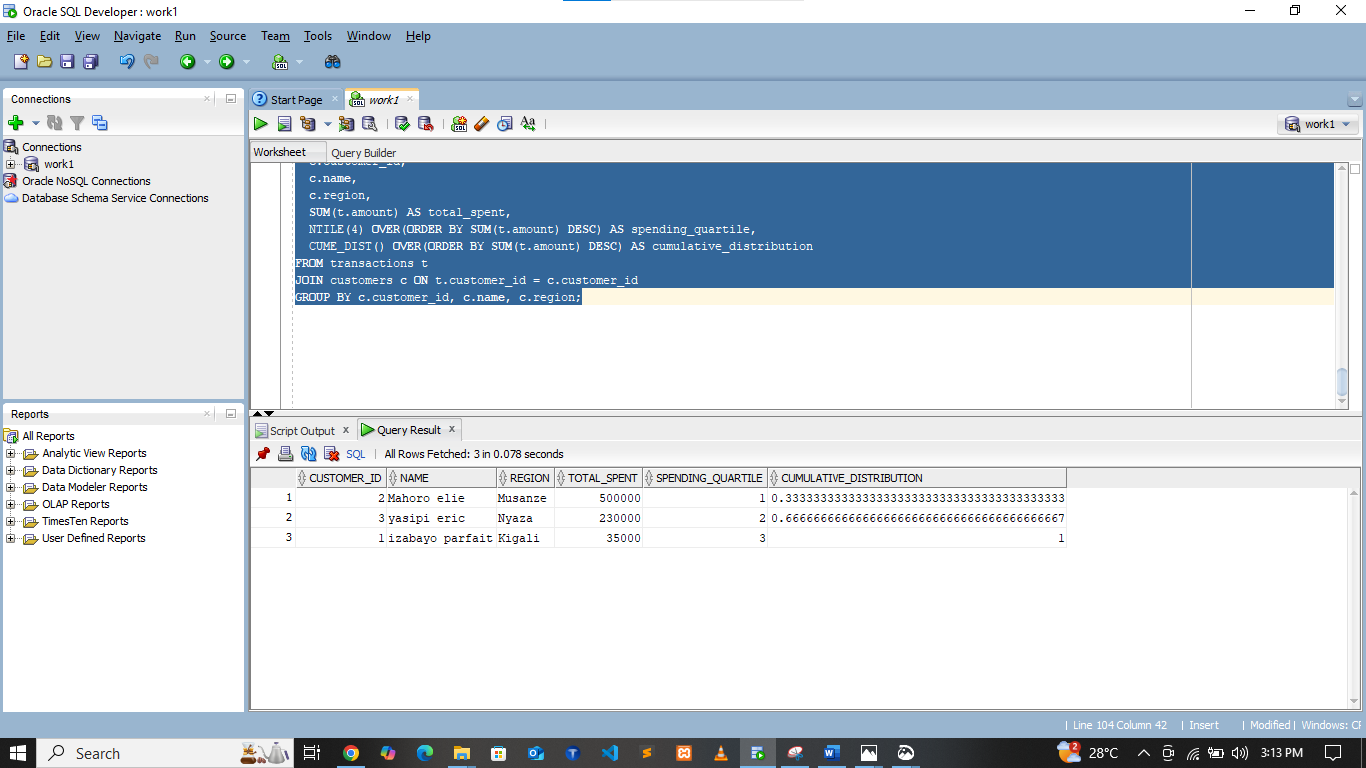
NTILE(4) OVER(ORDER BY SUM(t.amount) DESC) AS spending\_quartile,

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;



**INTERPRENTATIONS**

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