**Practical 3:**

**Part 4:**

**SQL Numeric Functions**

SQL **Numeric Functions** are essential tools for performing **mathematical** and arithmetic operations on numeric data. These functions allow you to manipulate numbers, perform calculations, and **aggregate data** for reporting and analysis purposes.

Note: In this Lab Manual, examples are based on ORACLE SQLPLUS and MYSQL for your support but you are free to use any platform.

**Numeric Functions in SQL\*Plus (Oracle) and MySQL**

|  |  |
| --- | --- |
| **Function** | **Description** |
| **ABS(n)** | Returns the absolute value of n |
| **ACOS(n)** | Returns the arc cosine (inverse cosine) of n |
| **ASIN(n)** | Returns the arc sine (inverse sine) of n |
| **ATAN(n)** | Returns the arc tangent (inverse tangent) of n |
| **ATN2(y, x) (SQL**  **Server only)** | Returns the angle whose tangent is the quotient of two arguments (Not available in  Oracle/MySQL) |

**AVG(expression)** Returns the average of a set of values

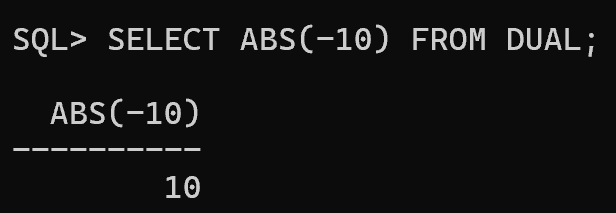
**CEILING(n)** Returns the smallest integer greater than or equal to n

|  |  |
| --- | --- |
| **COUNT(expression)** | Returns the number of rows matching a condition |
| **COS(n)** | Returns the cosine of n (in radians) |
| **COT(n)** | Returns the cotangent of n |
| **DEGREES(n)** | Converts radians to degrees |
| **EXP(n)** | Returns e raised to the power of n |
| **FLOOR(n)** | Returns the largest integer less than or equal to n |
| **LOG(n)** | Returns the natural logarithm (base e) of n |
| **LOG10(n)** | Returns the base-10 logarithm of n |
| **MAX(expression)** | Returns the maximum value in a column |
| **MIN(expression)** | Returns the minimum value in a column |
| **PI()** | Returns the value of π (pi) |
| **POWER(x, y)** | Returns x raised to the power of y |
| **RADIANS(n)** | Converts degrees to radians |
| **RAND()** | Returns a random number between 0 and 1 |
| **ROUND(n, d)** | Rounds n to d decimal places |
| **SIGN(n)** | Returns -1, 0, or 1 depending on the sign of n |
| **SIN(n)** | Returns the sine of n (in radians) |
| **SQRT(n)** | Returns the square root of n |
| **SQUARE(n)** | Returns the square of n (same as POWER(n, 2)) |

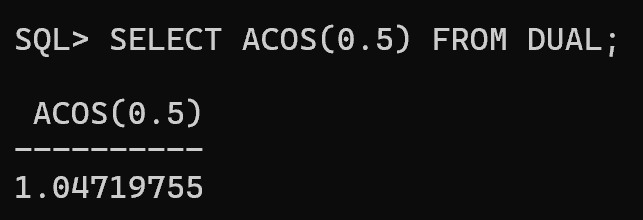
|  |  |
| --- | --- |
| **SUM(expression)** | Returns the sum of a set of values |
| **TAN(n)** | Returns the tangent of n (in radians) |

**2. Examples in SQL\*Plus (Oracle) /skip if you want to use mysql platform**

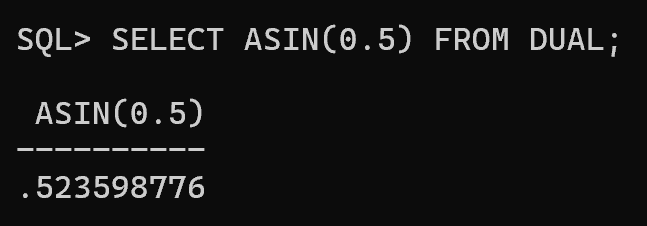
**2.1 Absolute Value (ABS)**

SELECT ABS(-10) FROM dual; -- Result: 10

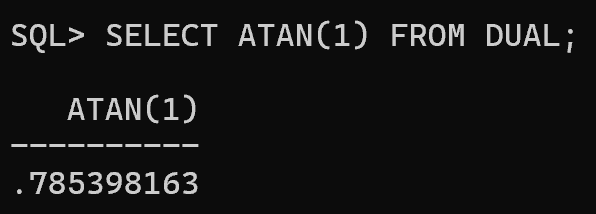
**2.2 Arc Cosine (ACOS)**

SELECT ACOS(0.5) FROM dual; -- Result: 1.04719755 (in radians)

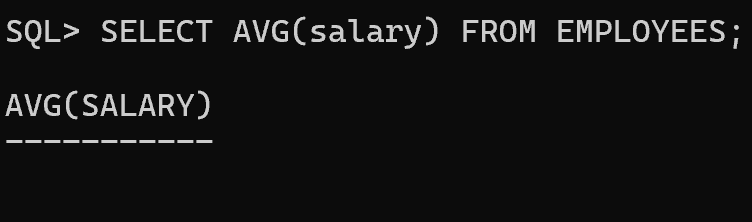
**2.3 Arc Sine (ASIN)**

SELECT ASIN(0.5) FROM dual; -- Result: 0.523598775 (in radians)

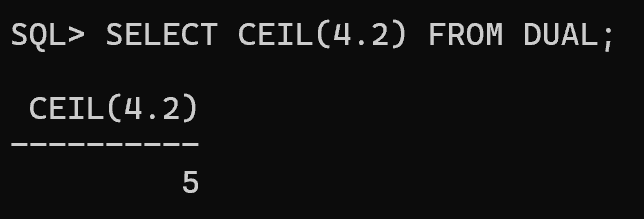
**2.4 Arc Tangent (ATAN)**

SELECT ATAN(1) FROM dual; -- Result: 0.785398163 (in radians) 

**2.5 Average (AVG)**

SELECT AVG(salary) FROM employees; -- Finds the average salary

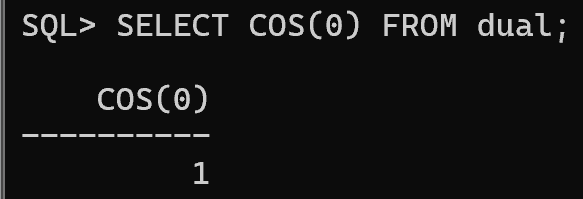
**2.6 Ceiling (CEILING)**

SELECT CEIL(4.2) FROM dual; -- Result: 5

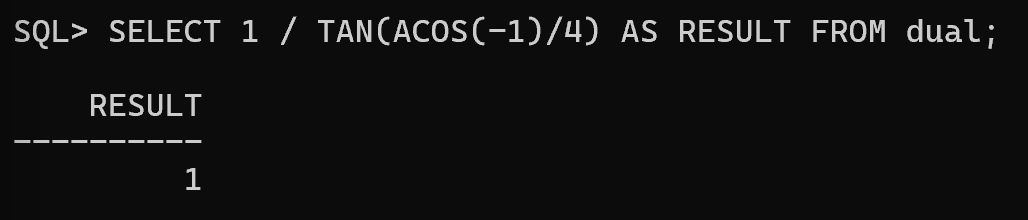
**2.7 Count (COUNT)**

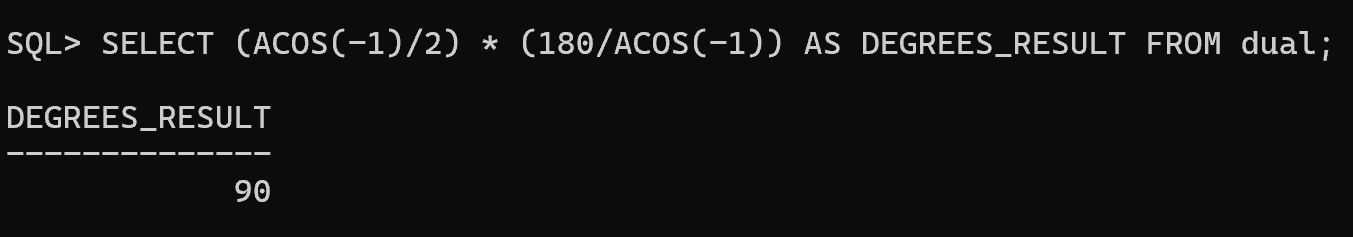
SELECT COUNT(\*) FROM employees WHERE department = 'Sales'; -- Counts employees in Sales

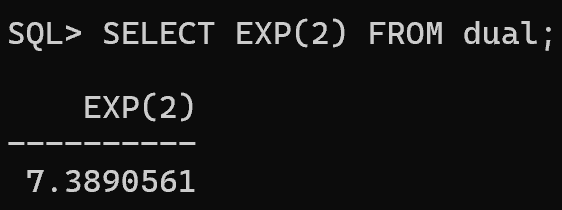
**2.8 Cosine (COS)**

SELECT COS(0) FROM dual; -- Result: 1 

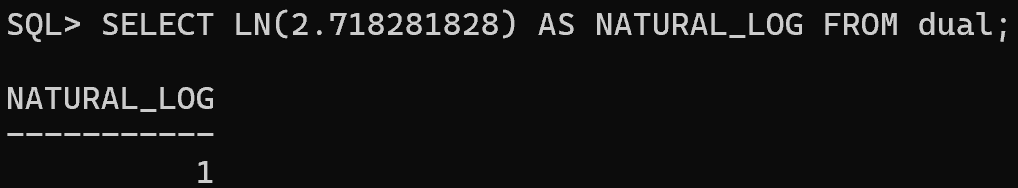
**2.9 Cotangent (COT)**

SELECT 1/TAN(PI()/4) FROM dual; -- Result: 1 **2.10 Convert Radians to Degrees (DEGREES)**

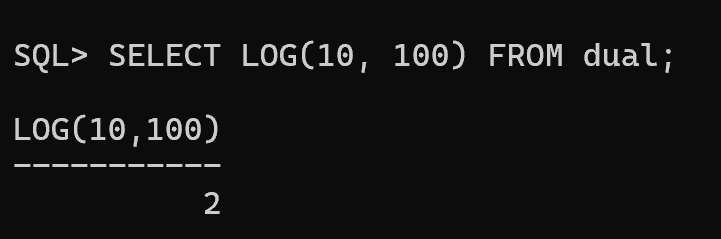
SELECT DEGREES(PI()/2) FROM dual; -- Result: 90 **2.11 Exponential (EXP)**

SELECT EXP(2) FROM dual; -- Result: 7.389056099 **2.12 Floor (FLOOR)**

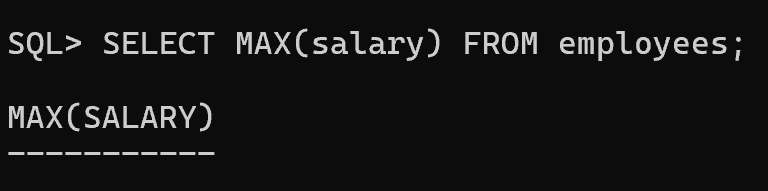
SELECT FLOOR(4.8) FROM dual; -- Result: 4 **2.13 Natural Logarithm (LOG)**

SELECT LOG(2.718281828) FROM dual; -- Result: 1 (since e^1 = e) 

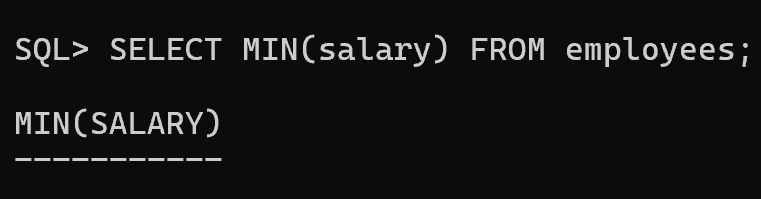
**2.14 Logarithm Base 10 (LOG10)**

SELECT LOG(10, 100) FROM dual; -- Result: 2 (since 10^2 = 100) 

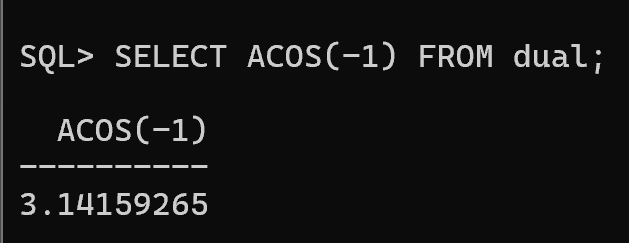
**2.15 Maximum (MAX)**

SELECT MAX(salary) FROM employees; -- Finds the highest salary 

**2.16 Minimum (MIN)**

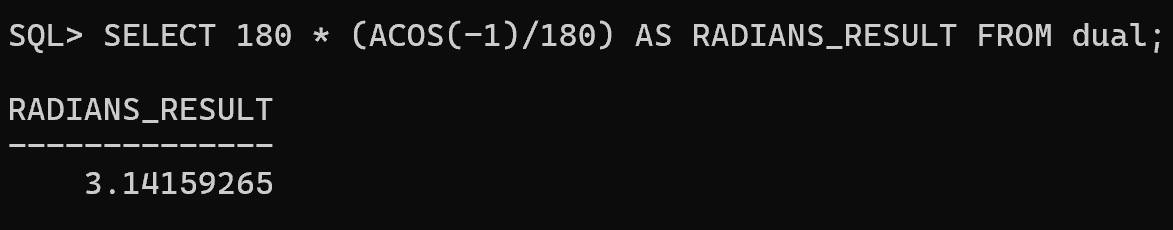
SELECT MIN(salary) FROM employees; -- Finds the lowest salary

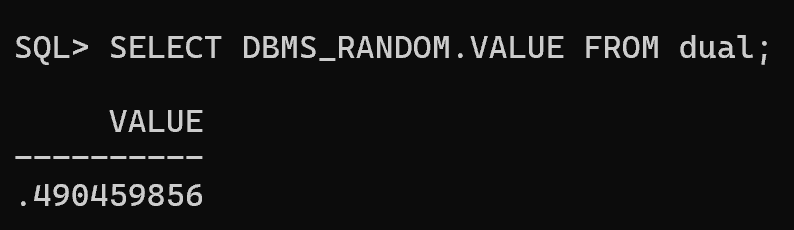
**2.17 Pi (PI)**

SELECT ACOS(-1) FROM dual; -- Result: 3.14159265 **2.18 Power (POWER)**

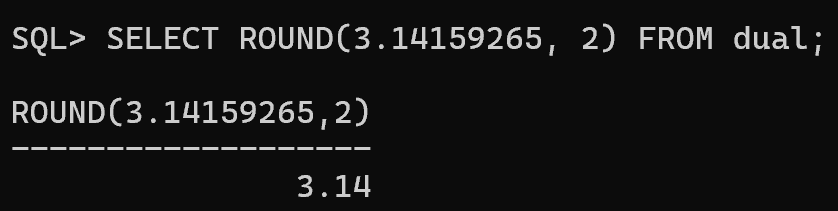
SELECT POWER(3, 2) FROM dual; -- Result: 9

**2.19 Convert Degrees to Radians (RADIANS)**

SELECT RADIANS(180) FROM dual; -- Result: 3.14159265 **2.20 Random Number (RAND)**

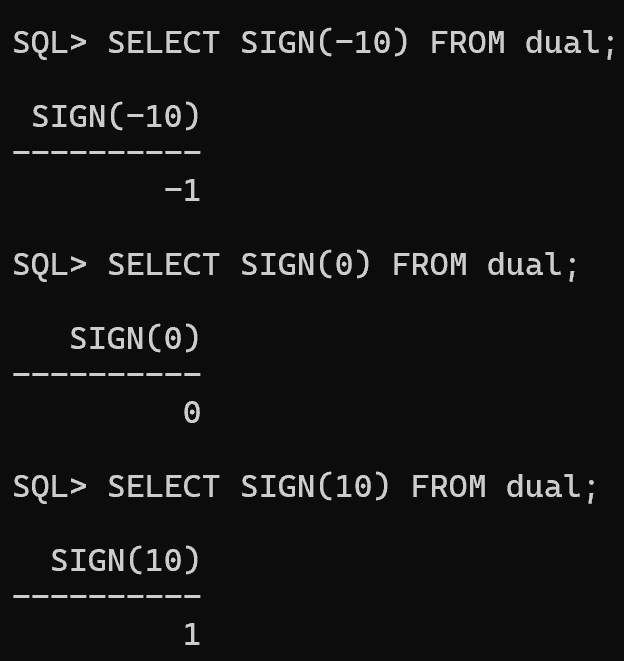
SELECT DBMS\_RANDOM.VALUE FROM dual; -- Returns a random number between 0 and 1 

**2.21 Round Number (ROUND)**

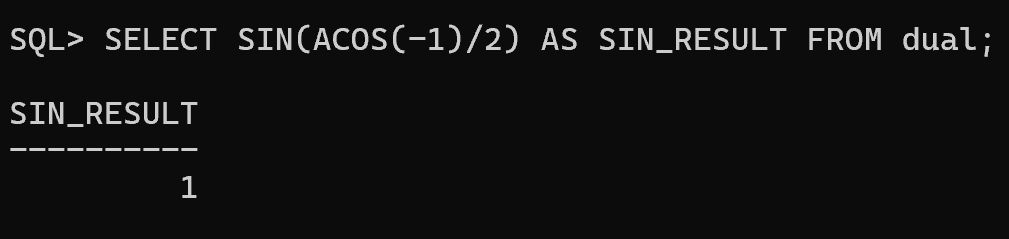
SELECT ROUND(3.14159265, 2) FROM dual; -- Result: 3.14 **2.22 Sign of Number (SIGN)**

SELECT SIGN(-10) FROM dual; -- Result: -1

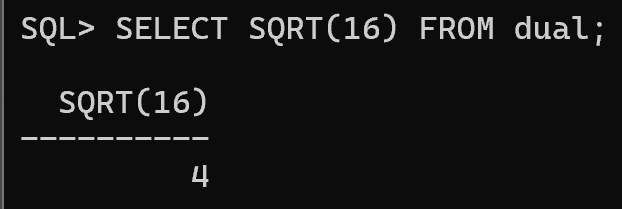
SELECT SIGN(0) FROM dual; -- Result: 0

SELECT SIGN(10) FROM dual; -- Result: 1 

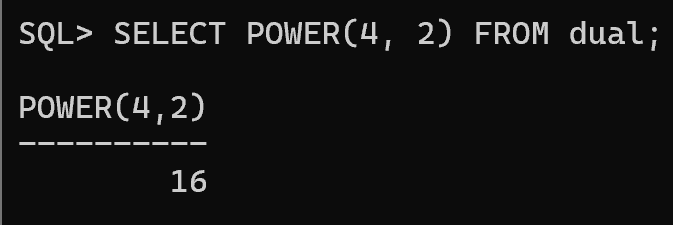
**2.23 Sine (SIN)**

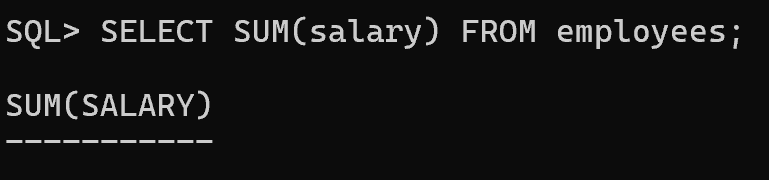
SELECT SIN(PI()/2) FROM dual; -- Result: 1

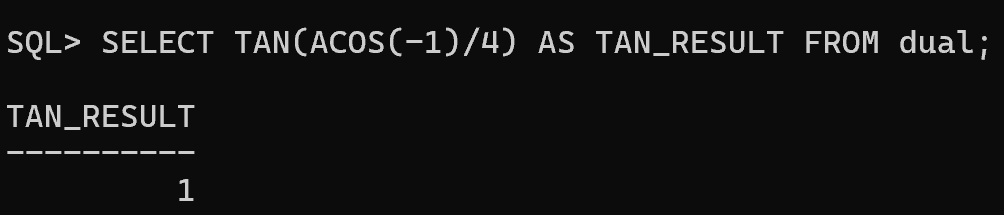
**2.24 Square Root (SQRT)**

SELECT SQRT(16) FROM dual; -- Result: 4 

**2.25 Square (SQUARE)**

SELECT POWER(4, 2) FROM dual; -- Result: 16 **2.26 Sum (SUM)**

SELECT SUM(salary) FROM employees; -- Sum of all salaries **2.27 Tangent (TAN)**

SELECT TAN(PI()/4) FROM dual; -- Result: 1 

**3. Examples in MySQL //SKIP IF DONE WITH ORACLE SQLPLUS**

�� The MySQL syntax is almost the same as Oracle, except for some functions.

SELECT ABS(-10); -- 10

SELECT ACOS(0.5); -- 1.04719755

SELECT ASIN(0.5); -- 0.523598775

SELECT ATAN(1); -- 0.785398163

SELECT AVG(salary) FROM employees;

SELECT CEIL(4.2); -- 5

SELECT COUNT(\*) FROM employees WHERE department = 'Sales';

SELECT COS(0); -- 1

SELECT COT(1); -- 0.6420926159

SELECT DEGREES(PI()/2); -- 90

SELECT EXP(2); -- 7.389056099

SELECT FLOOR(4.8); -- 4

SELECT LOG(2.718281828); -- 1

SELECT LOG10(100); -- 2

SELECT MAX(salary) FROM employees;

SELECT MIN(salary) FROM employees;

SELECT PI(); -- 3.1415926535

SELECT POWER(3, 2); -- 9

SELECT RADIANS(180); -- 3.1415926535

SELECT RAND(); -- Random number

SELECT ROUND(3.14159265, 2); -- 3.14

SELECT SIGN(-10); -- -1

SELECT SIN(PI()/2); -- 1

SELECT SQRT(16); -- 4

SELECT SUM(salary) FROM employees;

SELECT TAN(PI()/4); -- 1

**Advanced SQL Numeric Function Use Cases (Oracle & MySQL)**

Fllowing are **complex queries** using **numeric functions** in **real-world applications** for **financial analysis, scientific calculations, data analyt and system performance monitoring**.

**1️**⃣**Financial Analytics: Compound Interest Calculati Use Case:** Calculate compound interest for a bank’s customer accounts.

SELECT

account\_id,

principal,

interest\_rate,

years,

ROUND(principal \* POWER((1 + interest\_rate / 100), years), 2) AS future\_value

FROM savings\_accounts;

**Formula Used:**

FV=P×(1+r/n)(nt)FV = P \times (1 + r/n)^{(nt)}FV=P×(1+r/n)(nt) Where:

● principal: Initial deposit

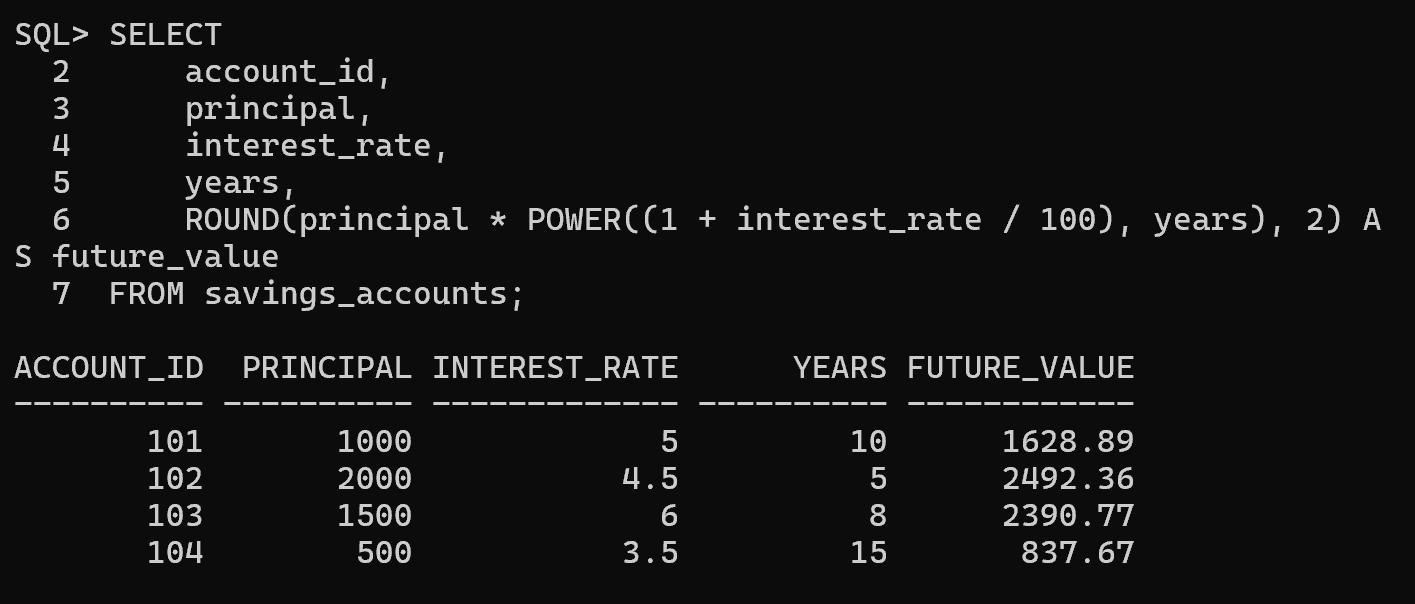
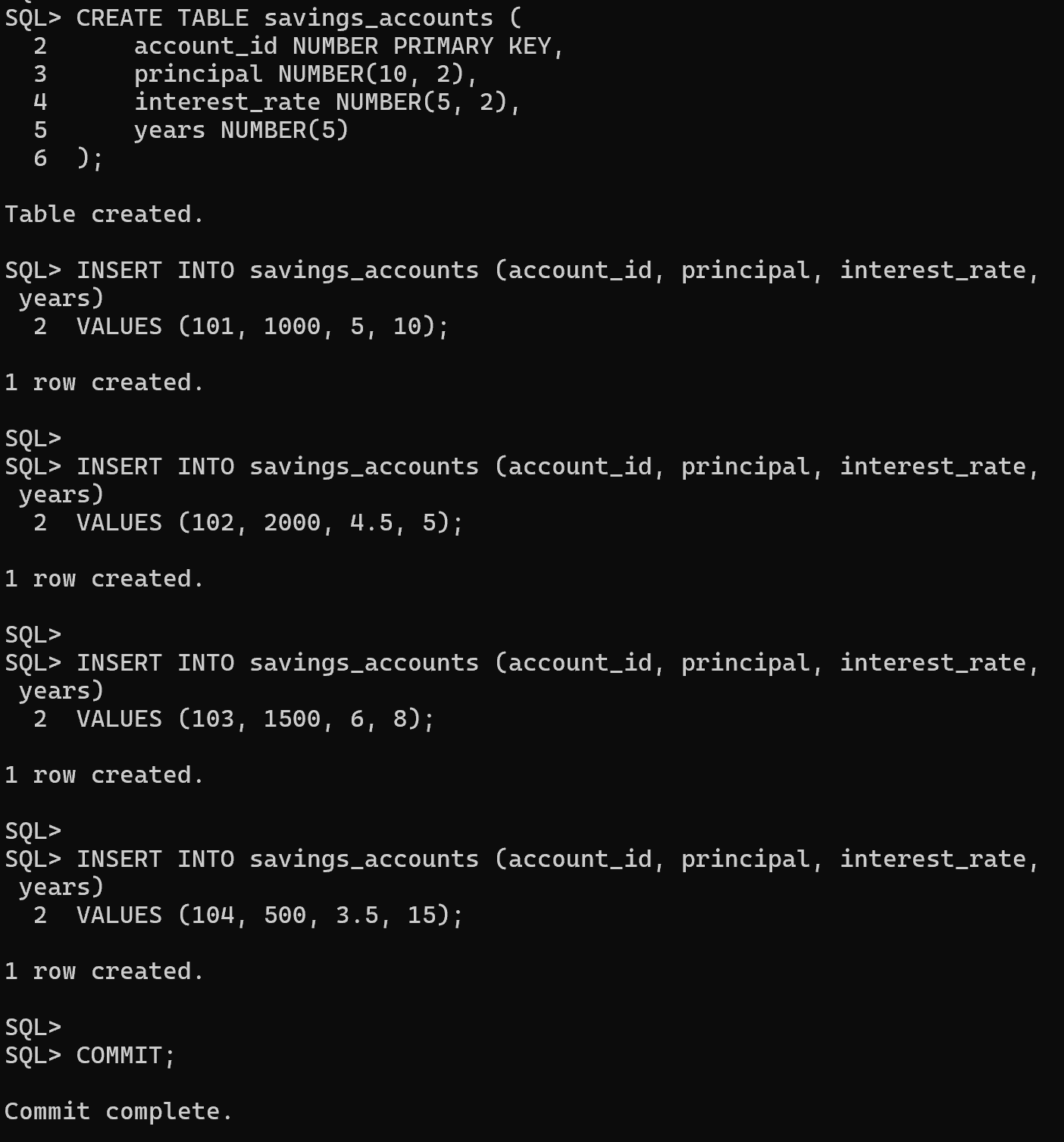
● interest\_rate: Annual interest rate

● years: Time period

● POWER() function computes exponentiation

**Result Example:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **account\_id** | **principal** | **interest\_rate** | **years** | **future\_value** |

101 1000 5 10 1628.89 

**2️**⃣**Customer Segmentation: Standard Deviation Purchases**

**Use Case:** Find **customer spending variability** to create better promotions.

SELECT

customer\_id,

ROUND(AVG(purchase\_amount), 2) AS avg\_spend, ROUND(STDDEV(purchase\_amount), 2) AS

spending\_variability

FROM orders

GROUP BY customer\_id

HAVING COUNT(\*) > 5; -- Filter frequent customers

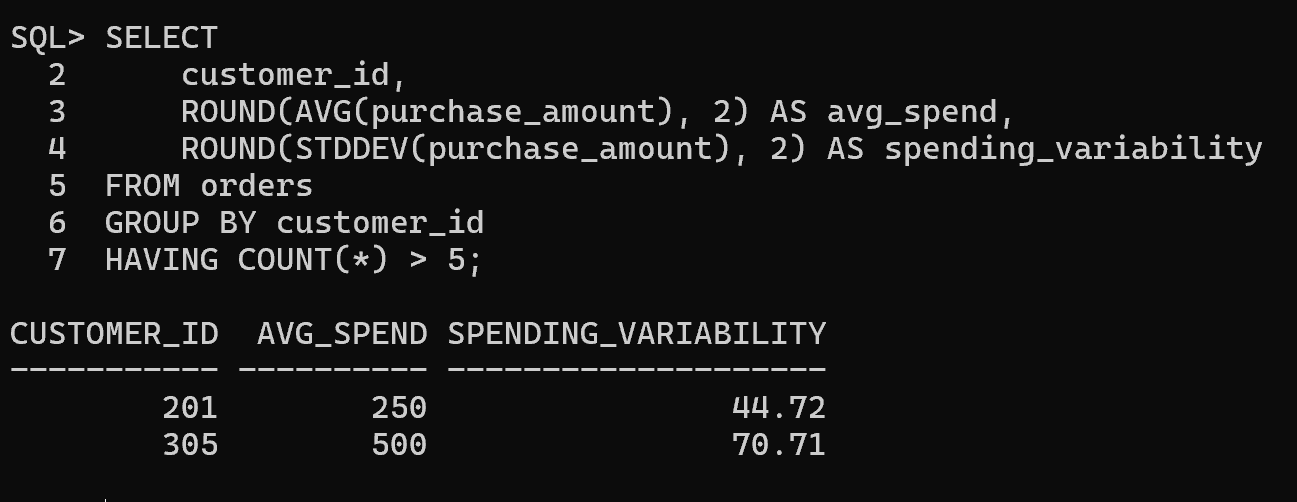
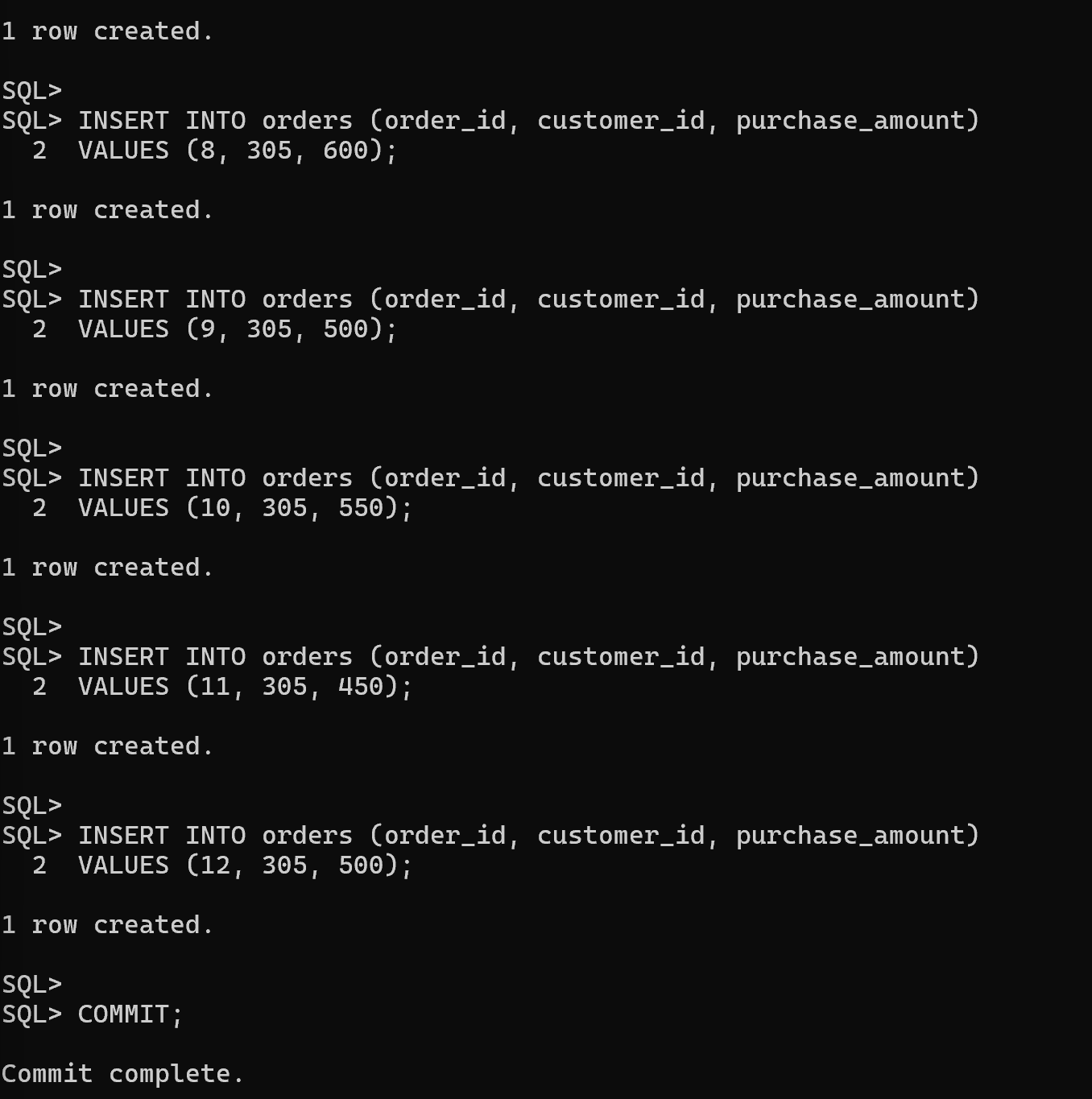
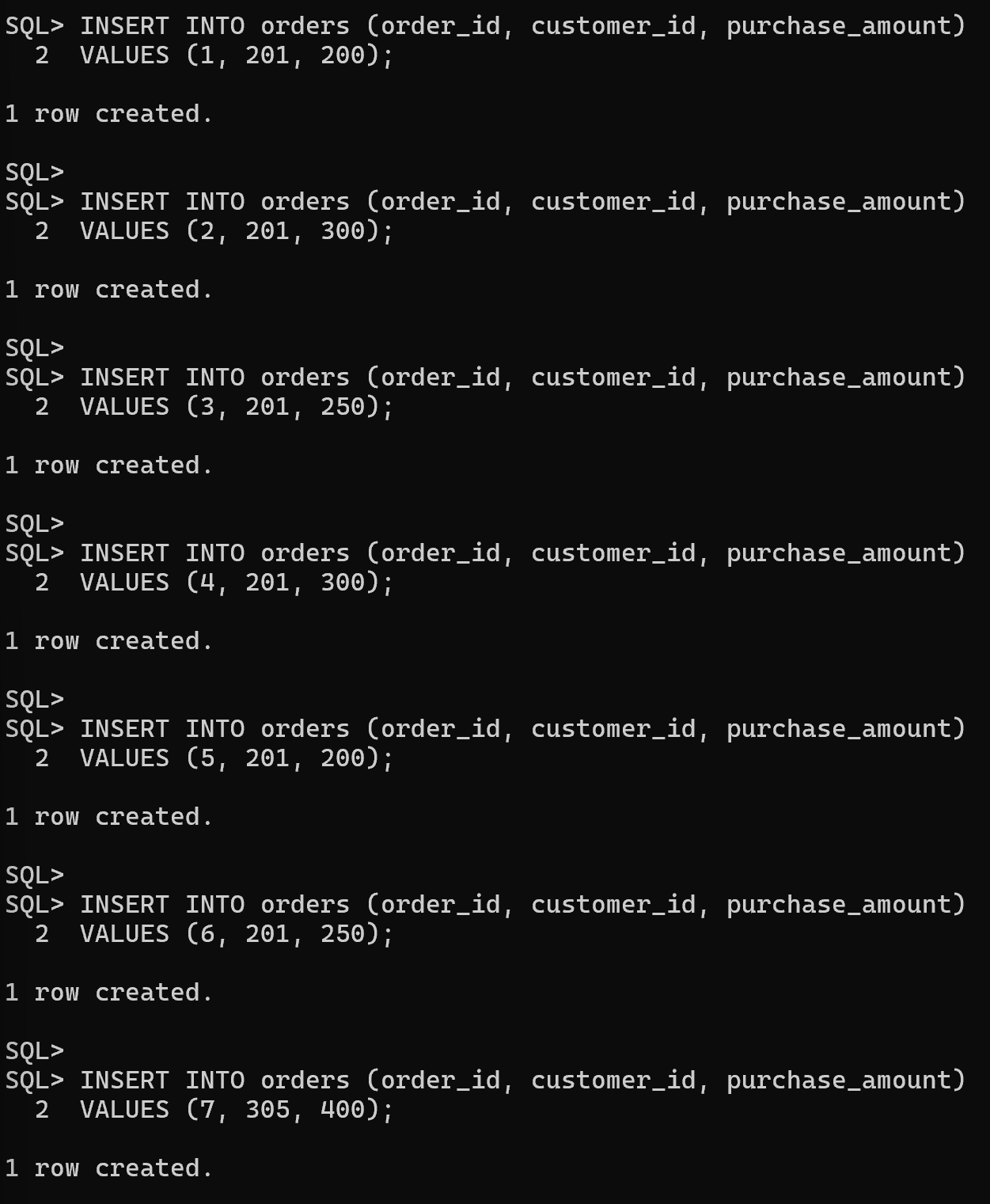
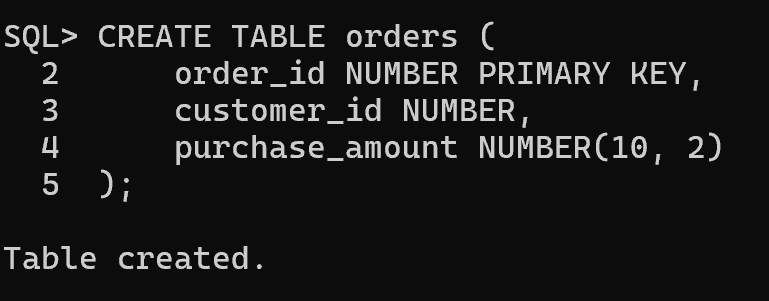
**Key Insights:**

● Uses STDDEV() to measure customer spending consistency.

● Filters for customers with at least **5 purchases** (HAVING COUNT(\*) > 5).

**Result Example:**

|  |  |  |
| --- | --- | --- |
| **customer\_id** | **avg\_spend** | **spending\_variability** |
| 201 | 250.00 | 50.25 |
| 305 | 500.00 | 120.75 |



**3️**⃣**Real-Time Performance Monitoring: CPU Load Analys Use Case:** Compute **CPU load trend** for a cloud server system.

SELECT

server\_id,

ROUND(AVG(cpu\_usage), 2) AS avg\_cpu,

ROUND(MAX(cpu\_usage), 2) AS max\_cpu,

ROUND(MIN(cpu\_usage), 2) AS min\_cpu,

ROUND(SQRT(POWER(MAX(cpu\_usage) - MIN(cpu\_usage), 2)), 2) AS load\_variance

FROM server\_logs

WHERE log\_date >= SYSDATE - INTERVAL '7' DAY -- Last 7 days

GROUP BY server\_id;

**Key Metrics:**

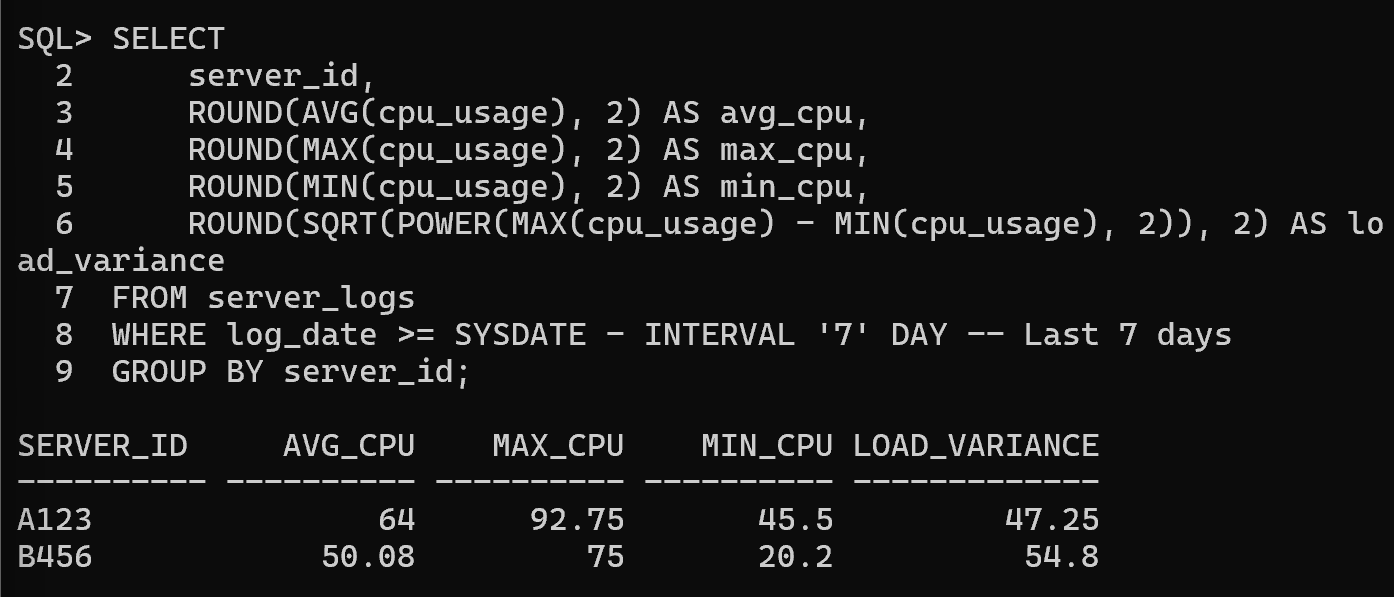
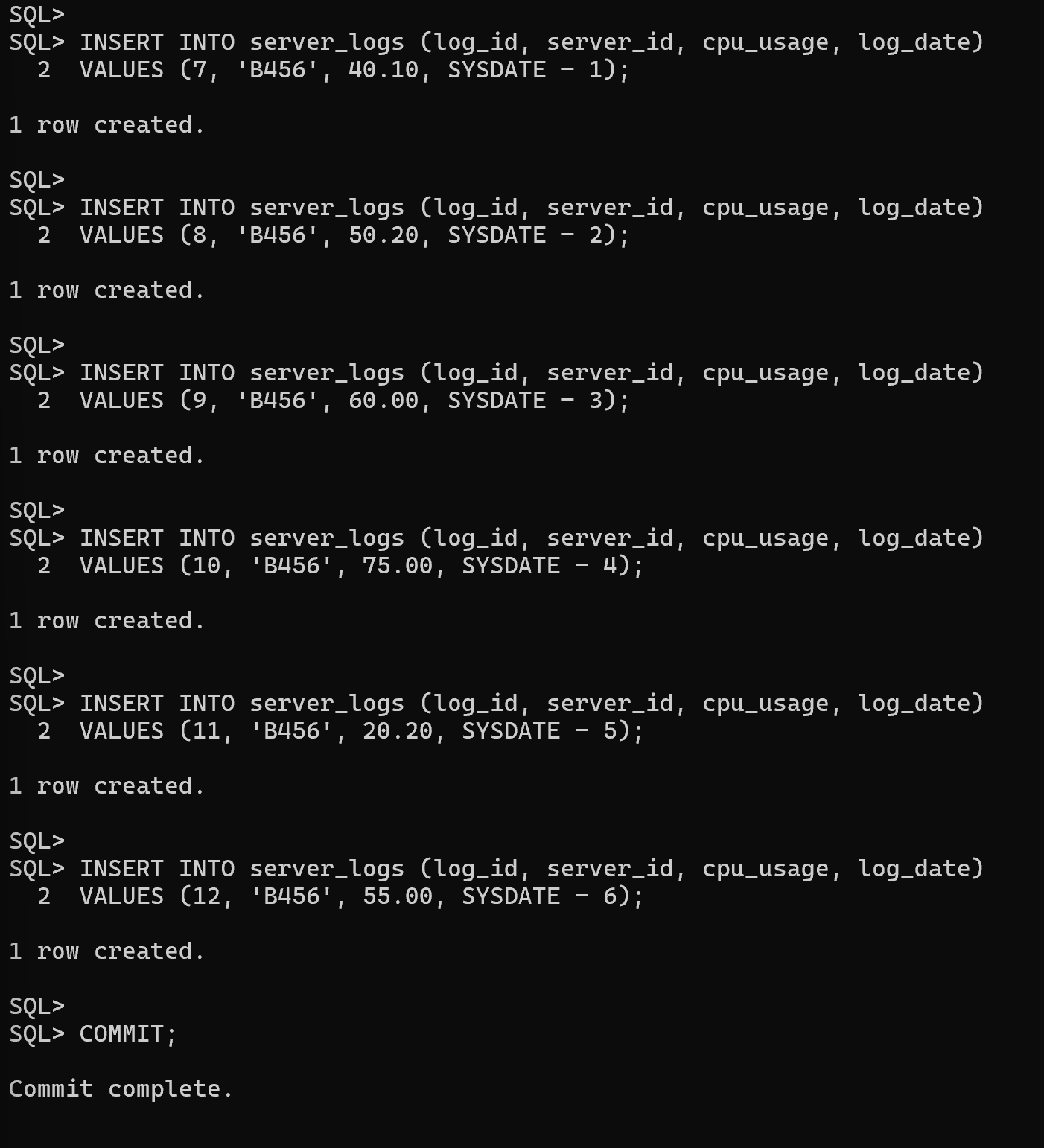
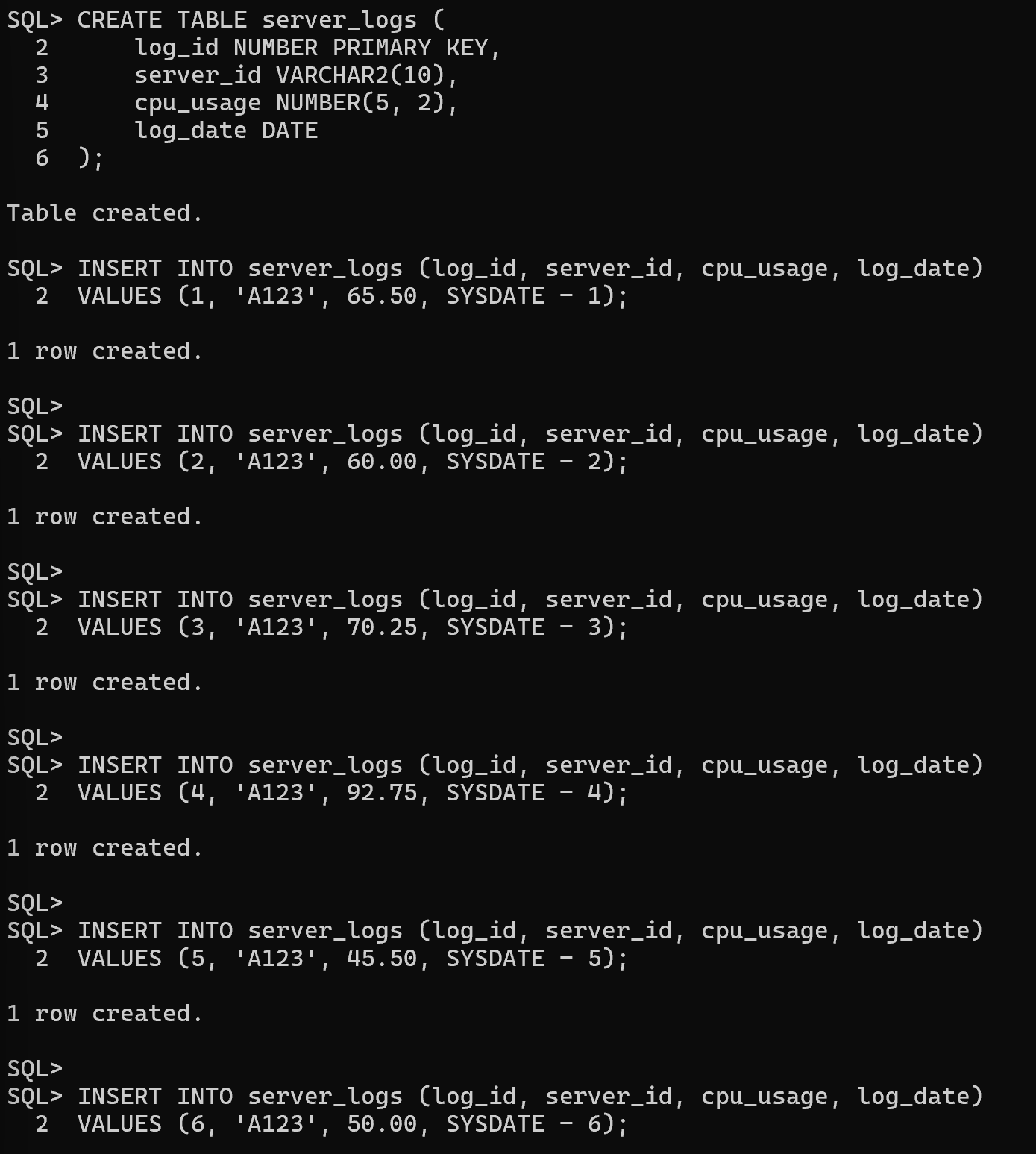
● AVG() to measure **average CPU usage**.

● MAX() & MIN() for **peak & lowest usage**.

● SQRT(POWER()) to calculate variance in load.

**Result Example:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **server\_id** | **avg\_cpu** | **max\_cpu** | **min\_cpu** | **load\_variance** |
| A123 | 65.25 | 92.75 | 45.50 | 47.25 |
| B456 | 40.10 | 75.00 | 20.20 | 54.80 |



**4️**⃣**Fraud Detection: Identifying Abnormal Transactio**

**Use Case:** Detect transactions that are significantly **higher** than the usu customer behavior.

SELECT

transaction\_id,

customer\_id,

amount,

(SELECT AVG(amount) FROM transactions WHERE customer\_id = t.customer\_id) AS avg\_amount, (SELECT STDDEV(amount) FROM transactions WHERE customer\_id = t.customer\_id) AS stddev\_amount FROM transactions t

WHERE amount > (SELECT AVG(amount) + 2 \* STDDEV(amount) FROM transactions WHERE customer\_id = t.customer\_id);

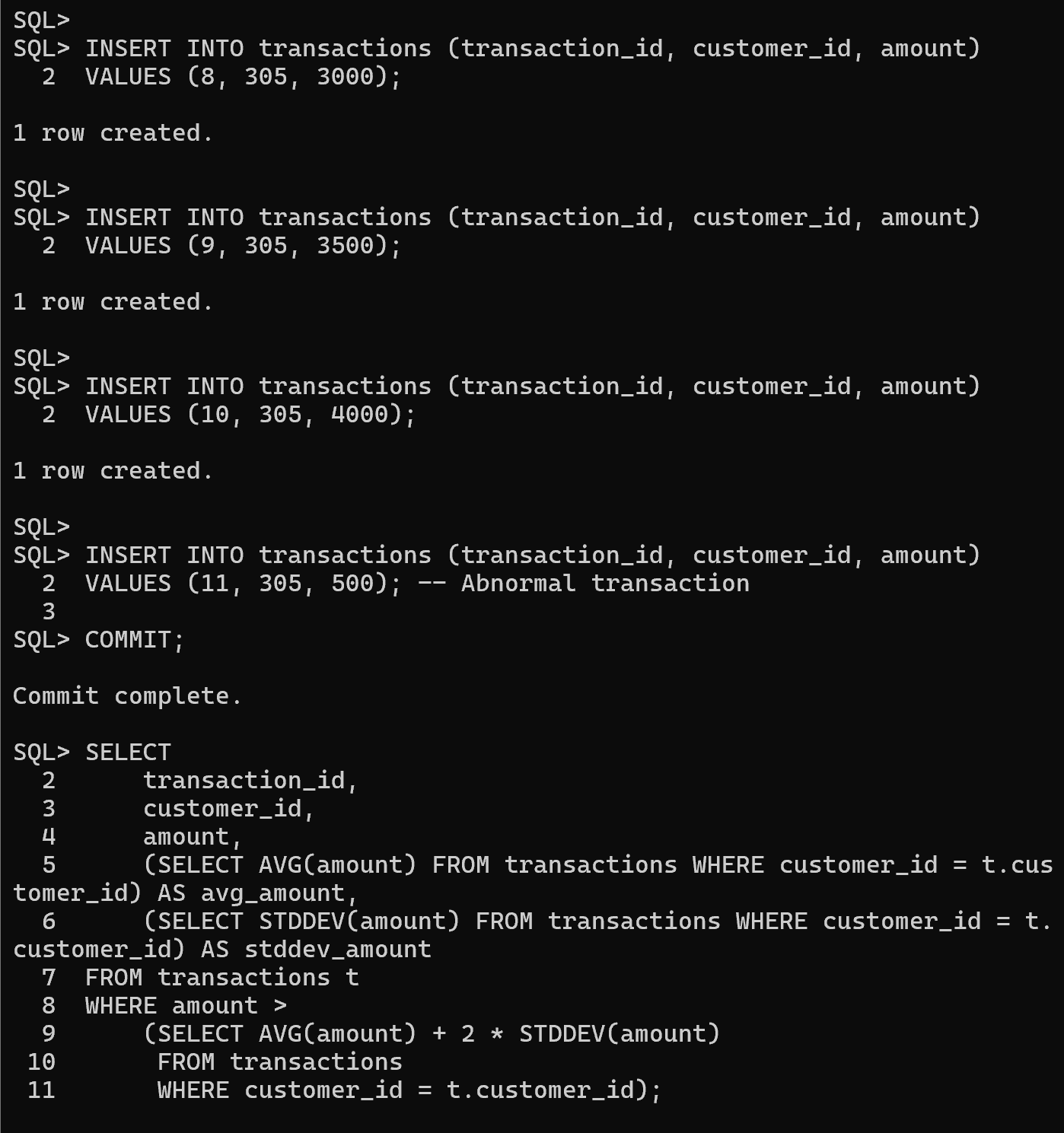
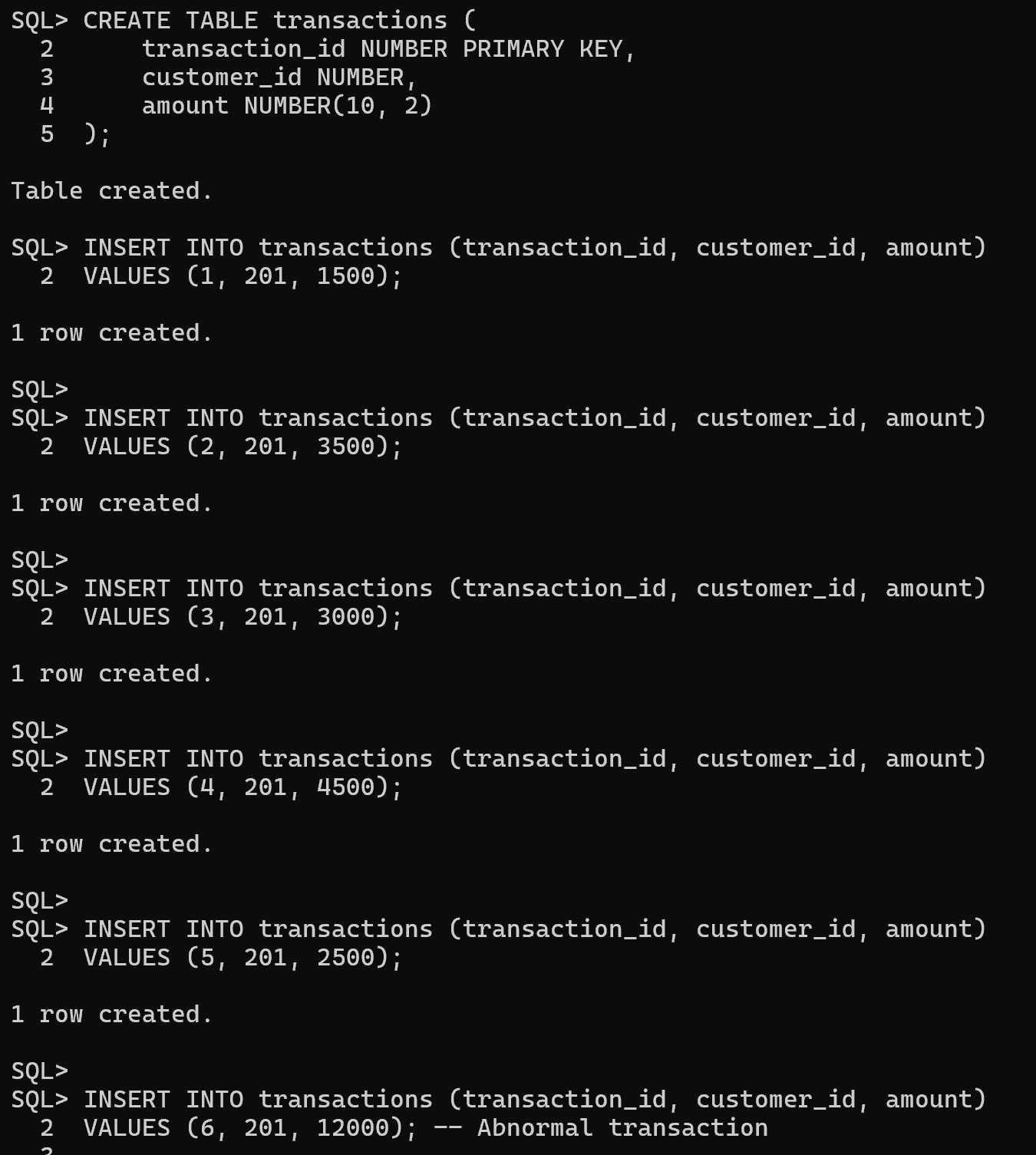
**Logic:**

● **Outlier transactions** are those **greater than 2 standard deviations** from the average.

● Uses AVG() and STDDEV() **per customer** to personalize fraud detection.

**Result Example:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **transaction\_i d** | **customer\_i d** | **amoun t** | **avg\_amou nt** | **stddev\_amou nt** |
| 90872 | 201 | 12000 | 3000 | 4500 |

�� **If a customer usually spends $3,000 ± $4,500, a $12,000 transaction is flagged as suspiciou **

**5️**⃣**Logistics: Estimating Delivery Time Based on Distan**

**Use Case:** Predict **delivery time (in hours)** for orders based on **distance and speed factors**.

SELECT

order\_id,

distance\_km,

ROUND(distance\_km / avg\_speed, 2) AS

estimated\_delivery\_time

FROM (

SELECT order\_id, distance\_km,

CASE

WHEN distance\_km < 50 THEN 60 -- Urban: 60 km/h

WHEN distance\_km BETWEEN 50 AND 200 THEN 80 -- Suburban: 80 km/h

ELSE 100 -- Highway: 100 km/h

END AS avg\_speed

FROM orders

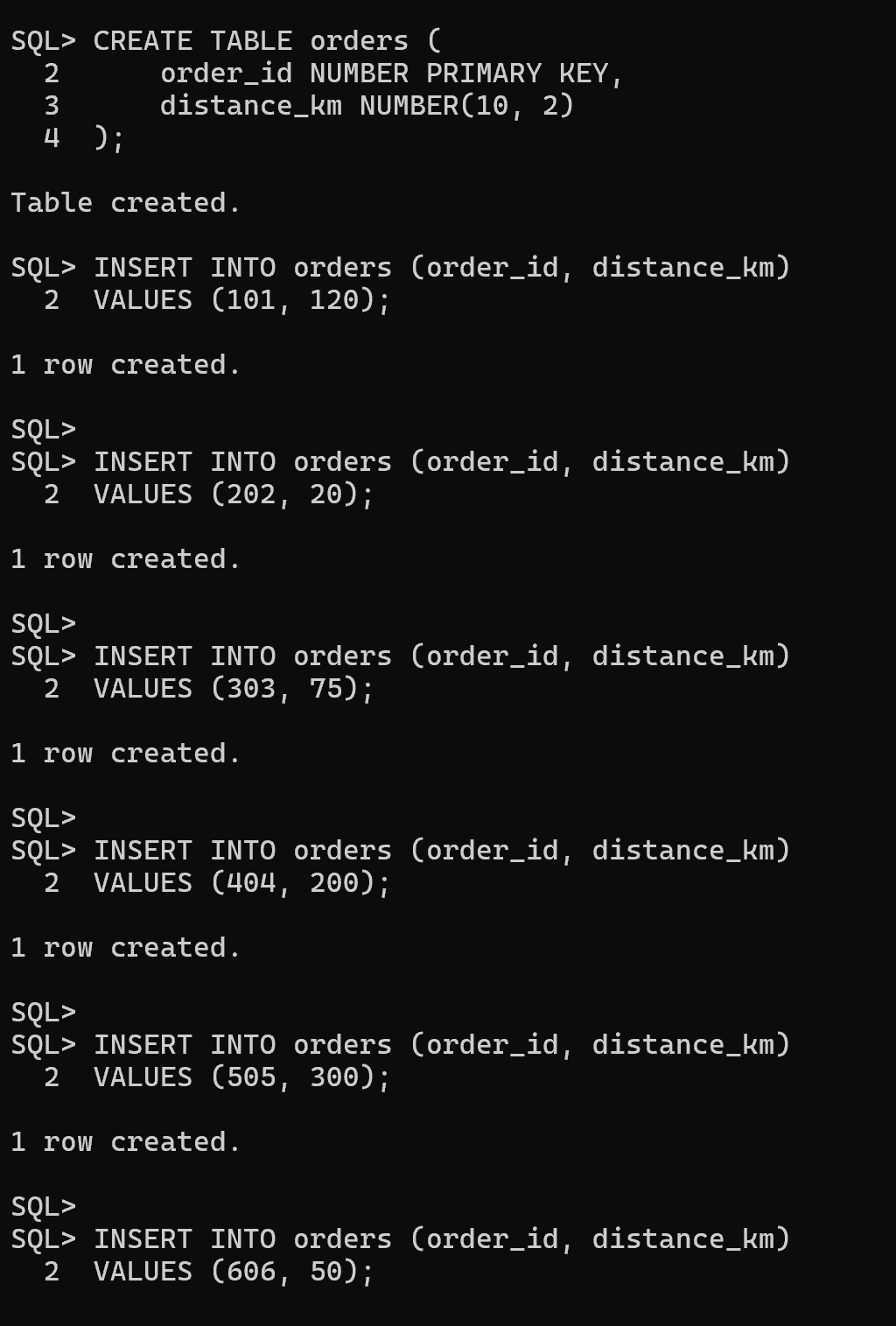
);

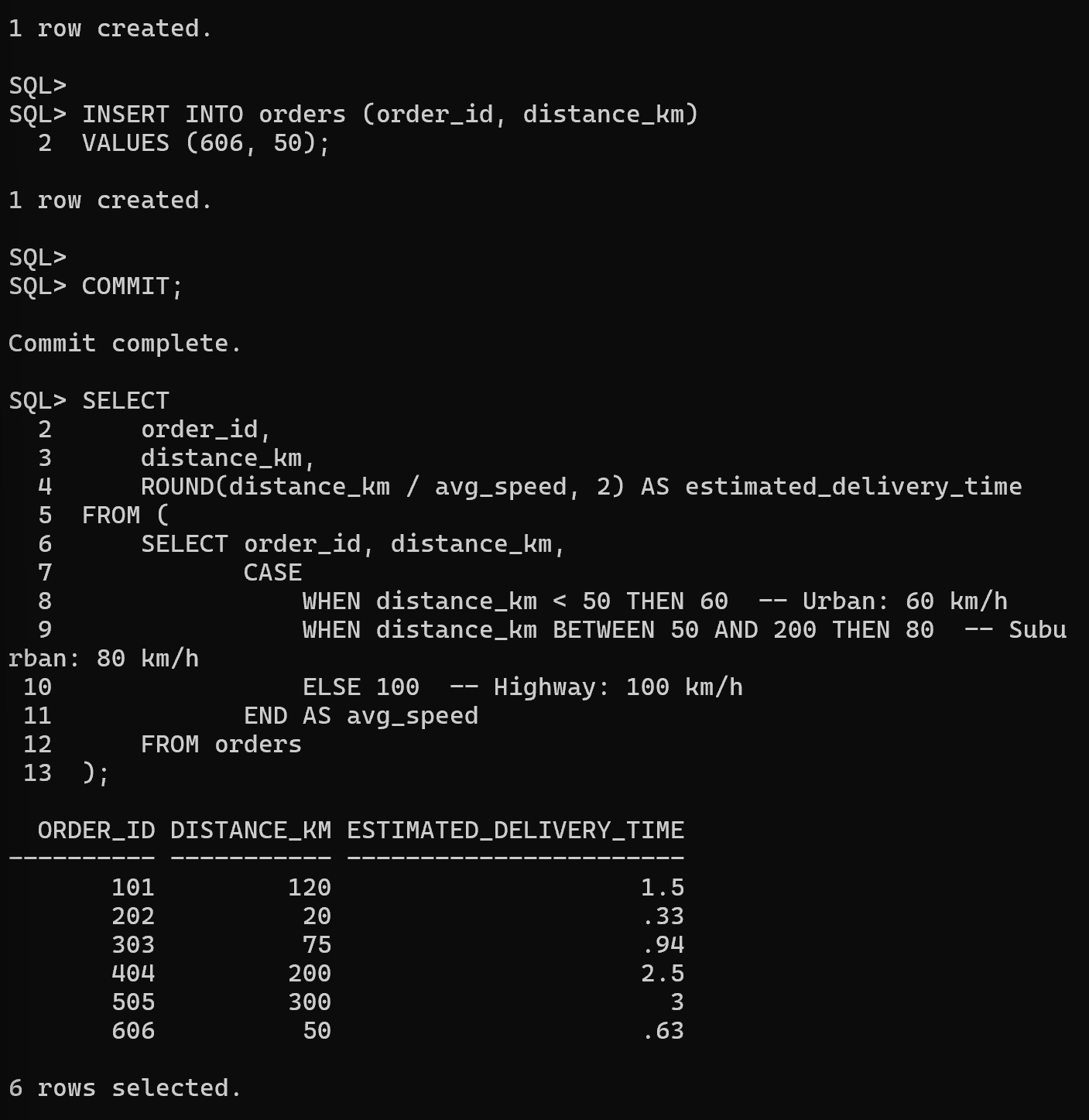
**Business Insight:**

● Uses **speed categories** (CASE) to calculate **realistic delivery estimates**. ● Uses ROUND() to **format** the estimated time.

**Result Example:**

|  |  |  |
| --- | --- | --- |
| **order\_id** | **distance\_km** | **estimated\_delivery\_time (hrs)** |
| 101 | 120 | 1.50 |
| 202 | 20 | 0.33 |





**6️**⃣**Astronomy/Physics: Calculating Earthquake Magnitu (Logarithmic Formula)**

**Use Case:** Compute earthquake **Richter magnitude** based on **seismic wave amplitude**.

SELECT

earthquake\_id,

station\_id,

amplitude,

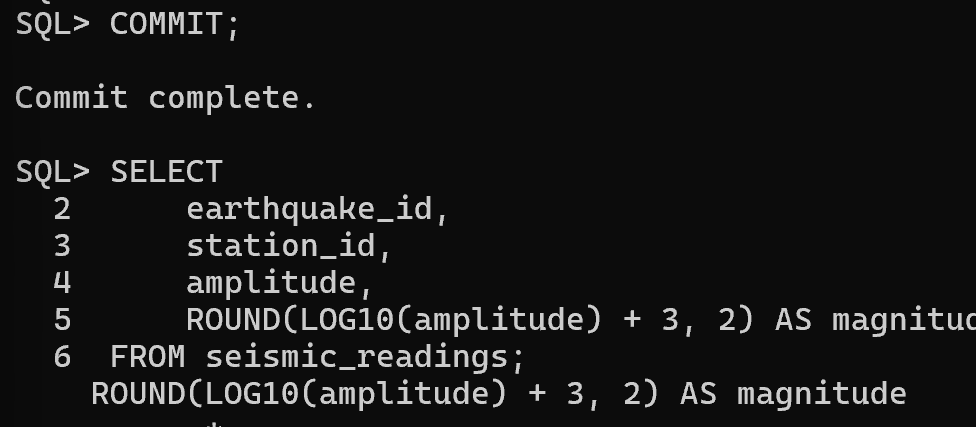
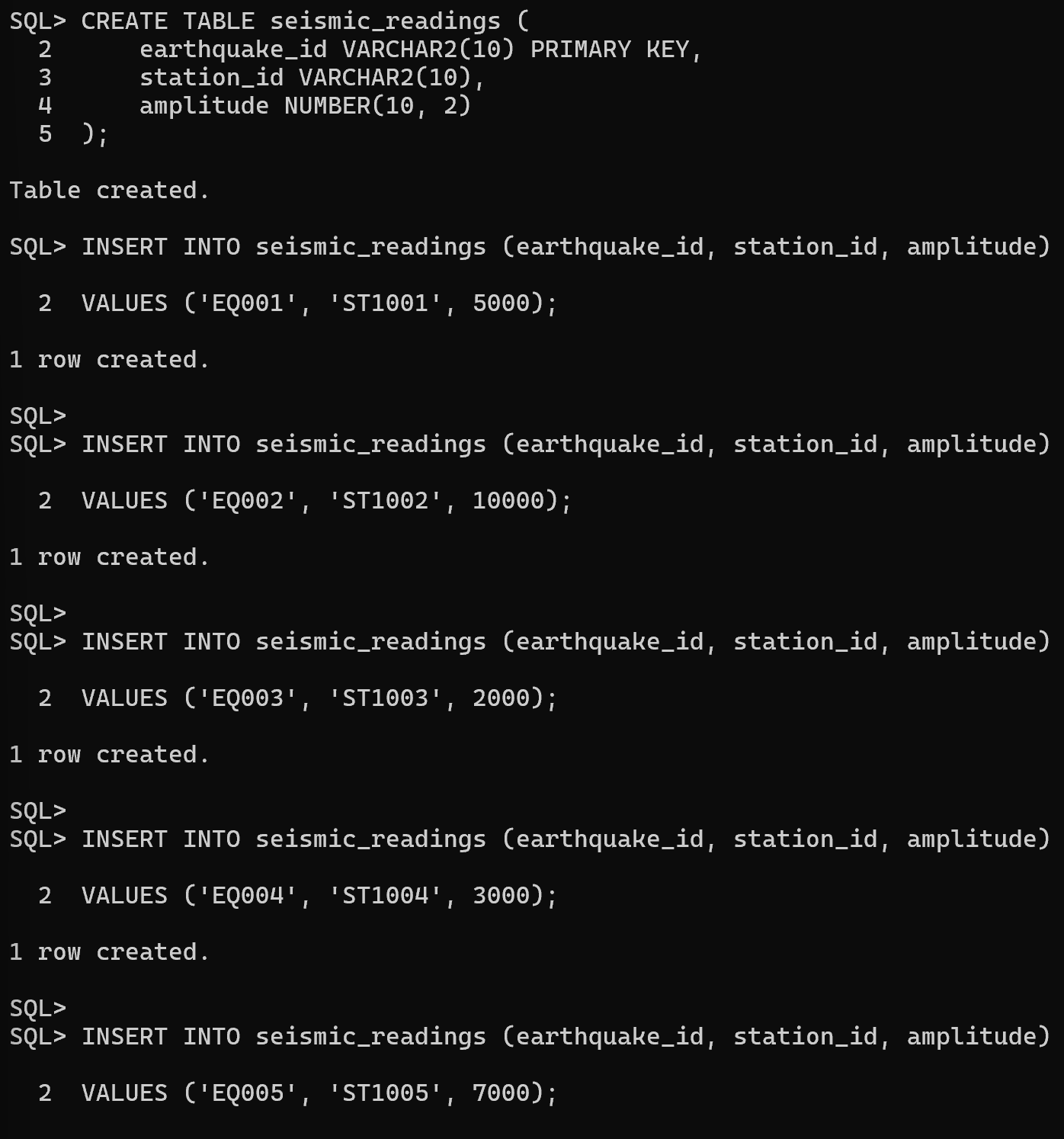
ROUND(LOG10(amplitude) + 3, 2) AS magnitude FROM seismic\_readings;

**Richter Scale Formula:**

M=log10(A)+3M = \log\_{10} (A) + 3M=log10(A)+3

● Uses LOG10() to calculate **magnitude from amplitude**. **Result Example:**

|  |  |  |  |
| --- | --- | --- | --- |
| **earthquake\_id** | **station\_id** | **amplitude** | **magnitude** |
| EQ001 | ST1001 | 5000 | 6.70 |



**7️**⃣**E-Commerce: Personalized Discount Calculati Use Case:** Apply dynamic **discount rates** based on **spending history**.

SELECT

customer\_id,

total\_spent,

CASE

WHEN total\_spent > 10000 THEN ROUND(total\_spent \* 0.15, 2)

WHEN total\_spent BETWEEN 5000 AND 10000 THEN ROUND(total\_spent \* 0.10, 2)

ELSE ROUND(total\_spent \* 0.05, 2)

END AS discount

FROM (

SELECT customer\_id, SUM(order\_value) AS total\_spent FROM orders GROUP BY customer\_id

);

**Discount Strategy:**

● **15% off** for VIP customers (> $10,000)

● **10% off** for mid-level ($5,000 - $10,000)

● **5% off** for casual shoppers (< $5,000)

**Result Example:**

|  |  |  |
| --- | --- | --- |
| **customer\_id** | **total\_spent** | **discount** |
| 101 | 12000 | 1800 |
| 202 | 7500 | 750 |

