Name : Bhavna Sanjay More
Year : TY **Division** : 03
Roll No : 38
Batch : B

<u>Experiment No. 2</u>

<u>Title :-</u>
Implementation of OLAP operations

<u>Aim :-</u>
Implementation of OLAP operations: Slice, Dice, Rollup, Drilldown and Pivot for the problem statement

<u>Problem Statement :-</u>
DWM Problem Statement (TE3-B) Design a data warehouse for a regional weather bureau. The weather bureau has about 100 probes, which are scattered throughout various land and ocean locations in the region to collect basic weather data, including air pressure, temperature and precipitation at each hour. All data is sent to the central station, which has collected such data for more than 10 years. Design Star schema and Snowflake schema such that it should facilitate efficient querying and online analytical processing and derive general weather patterns in multidimensional space.

<u>Theory :-</u>
OLAP (Online Analytical Processing) is a multidimensional data analysis approach used in data warehousing to support business intelligence, reporting, and decision-making. Unlike traditional databases optimized for transactions (OLTP), OLAP focuses on fast, flexible, and interactive analysis of large datasets.

Key Characteristics of OLAP:
- **Multidimensional Data Model:** Data is organized in cubes with dimensions (e.g., Time, Location, Product) and measures (e.g., Temperature, Sales).
- **Aggregation Support:** Allows summarization (e.g., average temperature by year).
- **Efficient Querying:** Optimized for complex analytical queries rather than transactions.
- **Interactive Exploration:** Users can drill down, roll up, slice, dice, and pivot data dynamically.

OLAP Operations
(a) Slice
- Definition: Extracts a 2D subset of the data cube by fixing one dimension.
- Example (Weather Bureau Case):
  - "Show all temperature readings from Probe #45." (Slicing on the Probe dimension)
  - "Analyze precipitation data for the year 2022." (Slicing on the Time dimension)

(b) Dice
- Definition: Extracts a subcube by applying conditions on multiple dimensions.
- Example (Weather Bureau Case):

        - "Find all temperature and pressure readings from coastal probes in Summer 2022." (Dicing on Location, Time, and Measurements)
        - "Compare land vs. ocean temperatures in Q1 2023."

(c) Roll-up (Aggregation)

- Definition: Summarizes data by moving up a hierarchy (e.g., from day → month → year).
- Example (Weather Bureau Case):
    - "What is the average temperature by region per year?" (Roll-up from daily → yearly)
    - "Total precipitation by season across all probes." (Aggregating from hourly → seasonal)

(d) Drill-down

- Definition: De-aggregates data by moving down a hierarchy (e.g., from year → month → day).
- Example (Weather Bureau Case):
    - "After seeing high precipitation in 2022, break it down by month to identify the wettest period."
    - "From annual temperature trends, drill into hourly data for heatwave analysis."

(e) Pivot (Rotate)

- Definition: Reorients the data cube to view it from a different perspective.
- Example (Weather Bureau Case):
    - "Switch rows and columns to compare temperature (rows) vs. precipitation (columns) by region."
    - "Pivot the view to analyze seasonal trends (columns) across different probe types (rows)."

OLAP Schemas in Weather Data Analysis

Star Schema

- Design: Central fact table (e.g., Fact_Weather_Data) linked to denormalized dimension tables (Time, Probe, Location).
- Advantage: Simplifies queries with fewer joins, ideal for slicing/dicing operations.

Snowflake Schema

- Design: Normalized dimensions with hierarchical splits (e.g., Time → Day → Month → Year).
- Advantage: Saves storage and supports efficient roll-up/drill-down on hierarchical attributes.

Why OLAP for Weather Data?

1. **Multidimensionality:** Time (hourly/daily/yearly), Location (region/country), and Probe (type/status) form natural axes for analysis.
2. **Hierarchies**: Temporal (hour → day → year) and spatial (probe → city → country) hierarchies enable trend analysis at varying scales.
3. **Interactive Exploration:** Meteorologists can dynamically slice data to diagnose anomalies (e.g., "Why did Probe 7 fail in Winter 2023?").

Procedure for OLAP Implementation (Weather Data)

1. **Data Setup**
    - Clean and load probe data into star/snowflake schema
    - Build time/location hierarchies
2. **OLAP Operations**
    - Slice: Fix one dimension (e.g., single probe)
    - Dice: Filter multiple dimensions (e.g., summer+coastal)
    - Roll-up: Aggregate (e.g., daily→monthly temps)

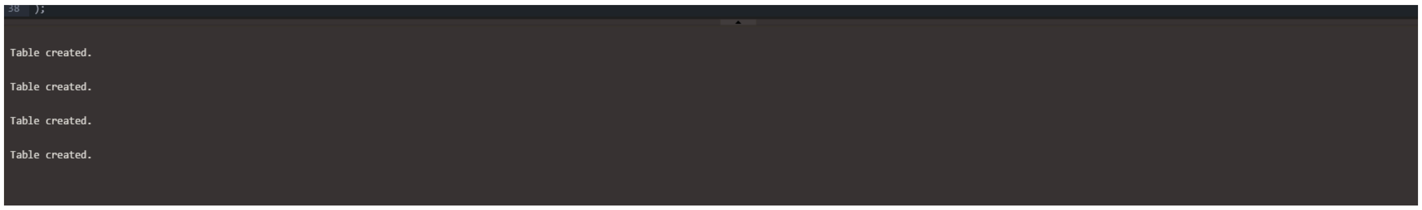
    - Drill-down: Expand details (e.g., year→month→day)
    - Pivot: Swap rows/columns for alternate views
3. **Validation**
    - Verify against raw data
    - Test query performance
4. **Tools**
    - SQL database + Power BI/Tableau

Problem Statement :-

SQL Implementation for Weather Data OLAP Operations

1. Database Schema Creation

```
-- Create tables
CREATE TABLE dim_probe (
  probe_id NUMBER PRIMARY KEY,
  probe_name VARCHAR2(50),
  probe_type VARCHAR2(20),
  status VARCHAR2(15)
);
CREATE TABLE dim_location (
  location_id NUMBER PRIMARY KEY,
  latitude NUMBER(9,6),
  longitude NUMBER(9,6),
  elevation NUMBER(7,2),
  location_type VARCHAR2(10) CHECK (location_type IN ('land','ocean','coastal')),
  region_name VARCHAR2(50),
  country VARCHAR2(50)
);
CREATE TABLE dim_time (
  time_id NUMBER PRIMARY KEY,
  datetime TIMESTAMP,
  hour NUMBER(2),
  day NUMBER(2),
  month NUMBER(2),
  quarter NUMBER(1),
  year NUMBER(4),
  season VARCHAR2(10) CHECK (season IN ('Spring','Summer','Fall','Winter'))
);
CREATE TABLE fact_weather_data (
  record_id NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
  probe_id NUMBER REFERENCES dim_probe(probe_id),
  time_id NUMBER REFERENCES dim_time(time_id),
  location_id NUMBER REFERENCES dim_location(location_id),
  temperature NUMBER(5,2),
  air_pressure NUMBER(7,2),
  precipitation NUMBER(6,2)
);
```



```
);

Table created.

Table created.

Table created.

Table created.
```

2. Sample Data Insertion

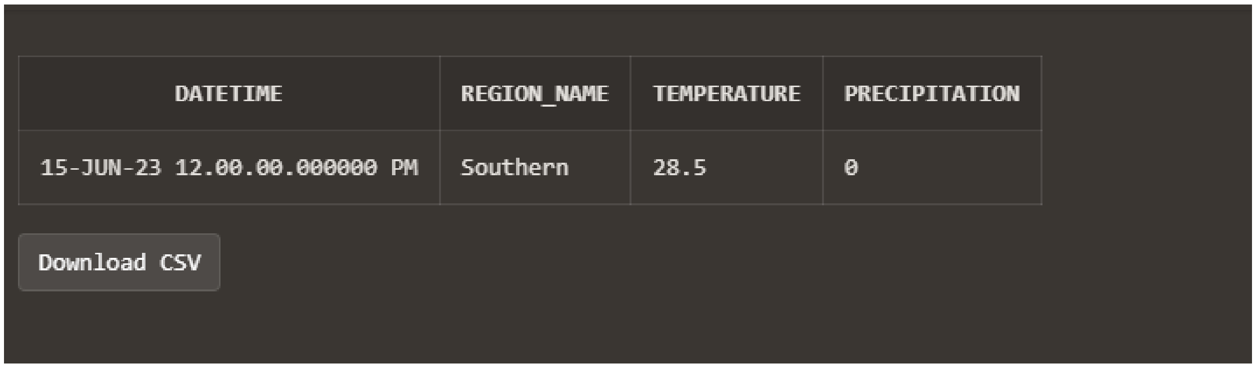
```
-- Insert dimension data
INSERT INTO dim_probe VALUES (1, 'Probe-ALPHA', 'land', 'active');
INSERT INTO dim_probe VALUES (2, 'Probe-BETA', 'ocean', 'active');
INSERT INTO dim_probe VALUES (3, 'Probe-GAMMA', 'coastal', 'maintenance');

INSERT INTO dim_location VALUES (101, 34.052235, -118.243683, 93.0, 'land', 'Southern', 'USA');
INSERT INTO dim_location VALUES (102, 33.689060, -117.893105, 17.0, 'coastal', 'Southern', 'USA');
INSERT INTO dim_location VALUES (103, 32.715736, -117.161087, 20.0, 'coastal', 'Southern', 'USA');

INSERT INTO dim_time VALUES (1001, TIMESTAMP '2023-06-15 12:00:00', 12, 15, 6, 2, 2023,
'Summer');
INSERT INTO dim_time VALUES (1002, TIMESTAMP '2023-06-15 13:00:00', 13, 15, 6, 2, 2023,
'Summer');
INSERT INTO dim_time VALUES (1003, TIMESTAMP '2023-12-15 09:00:00', 9, 15, 12, 4, 2023,
'Winter');

-- Insert fact data
INSERT INTO fact_weather_data (probe_id, time_id, location_id, temperature, air_pressure,
precipitation)
VALUES (1, 1001, 101, 28.5, 1012.3, 0.0);

INSERT INTO fact_weather_data (probe_id, time_id, location_id, temperature, air_pressure,
precipitation)
VALUES (2, 1001, 102, 25.2, 1013.1, 0.0);

-- Add more sample records as needed
COMMIT;
```

```
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

3. OLAP Operations Implementation

- **Slice Operation**

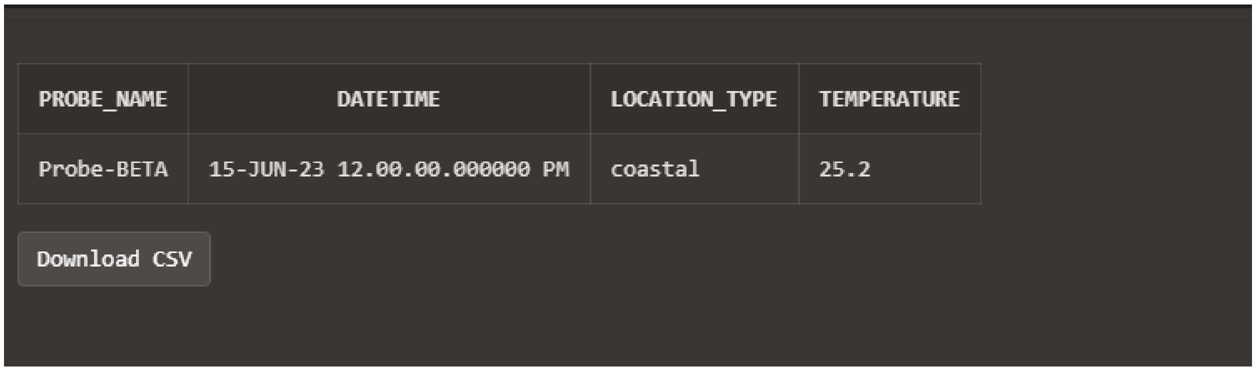
```
-- Get all data for a specific probe
SELECT t.datetime, l.region_name, f.temperature, f.precipitation
FROM fact_weather_data f
JOIN dim_probe p ON f.probe_id = p.probe_id
JOIN dim_time t ON f.time_id = t.time_id
JOIN dim_location l ON f.location_id = l.location_id
WHERE p.probe_id = 1;
```



DATETIME	REGION_NAME	TEMPERATURE	PRECIPITATION
15-JUN-23 12.00.00.000000 PM	Southern	28.5	0

- **Dice Operation**

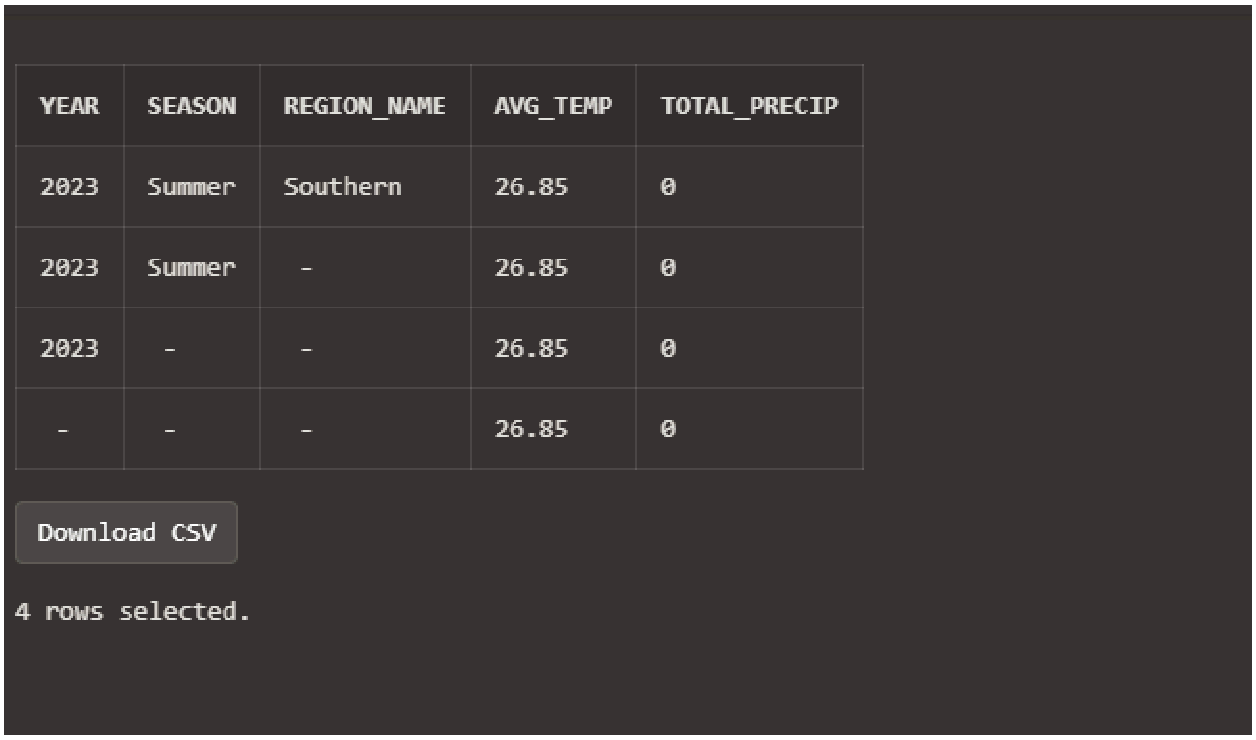
```
-- Filter on multiple dimensions
SELECT p.probe_name, t.datetime, l.location_type, f.temperature
FROM fact_weather_data f
JOIN dim_probe p ON f.probe_id = p.probe_id
JOIN dim_time t ON f.time_id = t.time_id
JOIN dim_location l ON f.location_id = l.location_id
WHERE l.location_type = 'coastal'
  AND t.season = 'Summer'
  AND f.temperature > 25;
```



PROBE_NAME	DATETIME	LOCATION_TYPE	TEMPERATURE
Probe-BETA	15-JUN-23 12.00.00.000000 PM	coastal	25.2

- **Roll-Up Operation**

```
-- Aggregate with GROUP BY ROLLUP
SELECT
    t.year,
    t.season,
    l.region_name,
    AVG(f.temperature) as avg_temp,
    SUM(f.precipitation) as total_precip
FROM fact_weather_data f
JOIN dim_time t ON f.time_id = t.time_id
JOIN dim_location l ON f.location_id = l.location_id
GROUP BY ROLLUP(t.year, t.season, l.region_name)
ORDER BY t.year, t.season, l.region_name;
```



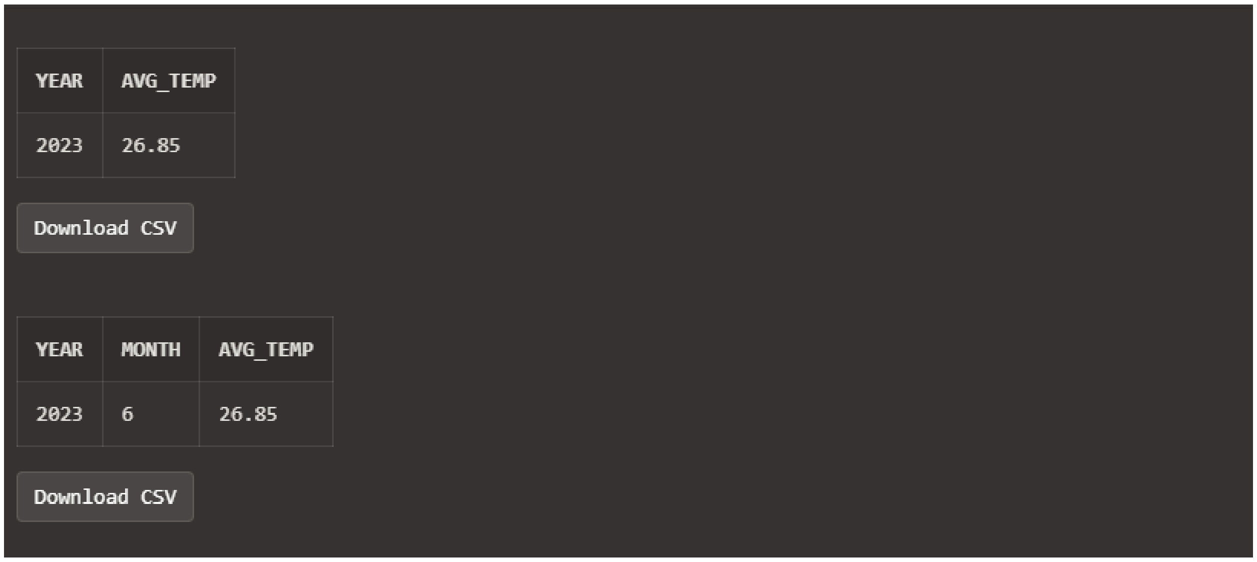
YEAR	SEASON	REGION_NAME	AVG_TEMP	TOTAL_PRECIP
2023	Summer	Southern	26.85	0
2023	Summer	-	26.85	0
2023	-	-	26.85	0
-	-	-	26.85	0

```
4 rows selected.
```

- **Drill-Down Operation**

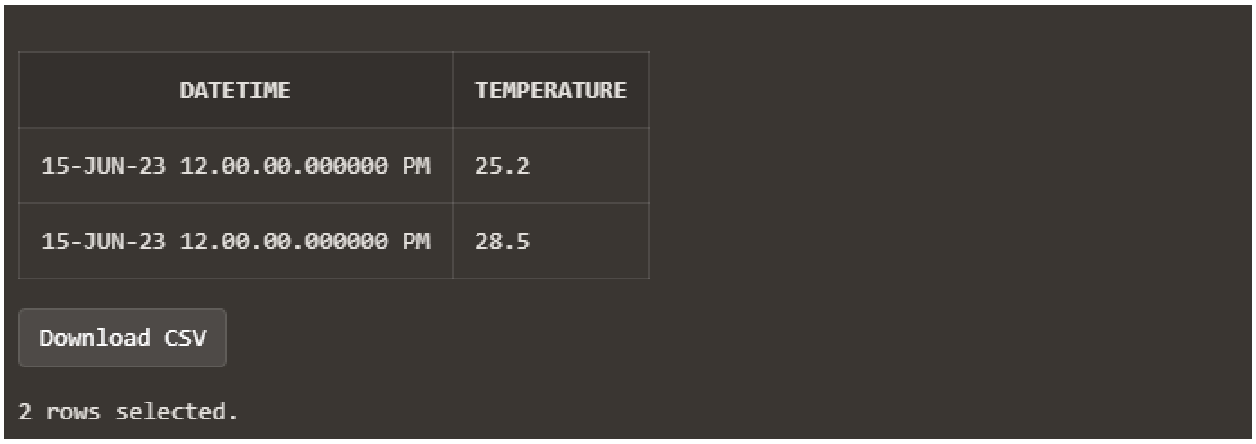
```
-- Yearly summary
```

```
SELECT t.year, AVG(f.temperature) as avg_temp
FROM fact_weather_data f
JOIN dim_time t ON f.time_id = t.time_id
GROUP BY t.year
ORDER BY t.year;
-- Monthly drill-down
SELECT t.year, t.month, AVG(f.temperature) as avg_temp
FROM fact_weather_data f
JOIN dim_time t ON f.time_id = t.time_id
GROUP BY t.year, t.month
ORDER BY t.year, t.month;
-- Daily drill-down
SELECT t.datetime, f.temperature
FROM fact_weather_data f
JOIN dim_time t ON f.time_id = t.time_id
WHERE EXTRACT(YEAR FROM t.datetime) = 2023
  AND EXTRACT(MONTH FROM t.datetime) = 6
ORDER BY t.datetime;
```



YEAR	AVG_TEMP
2023	26.85

YEAR	MONTH	AVG_TEMP
2023	6	26.85

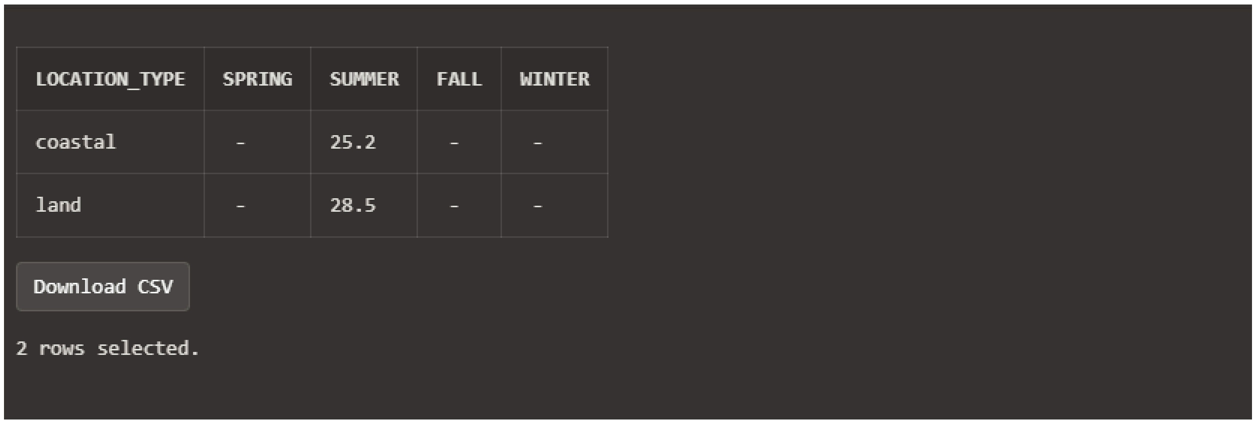


DATETIME	TEMPERATURE
15-JUN-23 12.00.00.000000 PM	25.2
15-JUN-23 12.00.00.000000 PM	28.5

```
2 rows selected.
```

- **Pivot Operation**

```
-- Using PIVOT clause (Oracle 11g+)
SELECT *
FROM (
    SELECT l.location_type, t.season, f.temperature
    FROM fact_weather_data f
    JOIN dim_time t ON f.time_id = t.time_id
    JOIN dim_location l ON f.location_id = l.location_id
)
PIVOT (
    AVG(temperature)
    FOR season IN ('Spring' AS Spring, 'Summer' AS Summer, 'Fall' AS Fall, 'Winter' AS Winter)
)
ORDER BY location_type;
```



LOCATION_TYPE	SPRING	SUMMER	FALL	WINTER
coastal	-	25.2	-	-
land	-	28.5	-	-

```
2 rows selected.
```

4. Materialized Views for Performance

```
-- Create materialized view for seasonal summaries
CREATE MATERIALIZED VIEW mv_seasonal_weather
REFRESH COMPLETE ON DEMAND
ENABLE QUERY REWRITE
AS
SELECT
    t.year,
    t.season,
    l.region_name,
    AVG(f.temperature) as avg_temp,
    SUM(f.precipitation) as total_precip,
    COUNT(*) as readings
FROM fact_weather_data f
JOIN dim_time t ON f.time_id = t.time_id
JOIN dim_location l ON f.location_id = l.location_id
GROUP BY t.year, t.season, l.region_name;
```

```
Statement processed.
```

5. Indexing and Partitioning

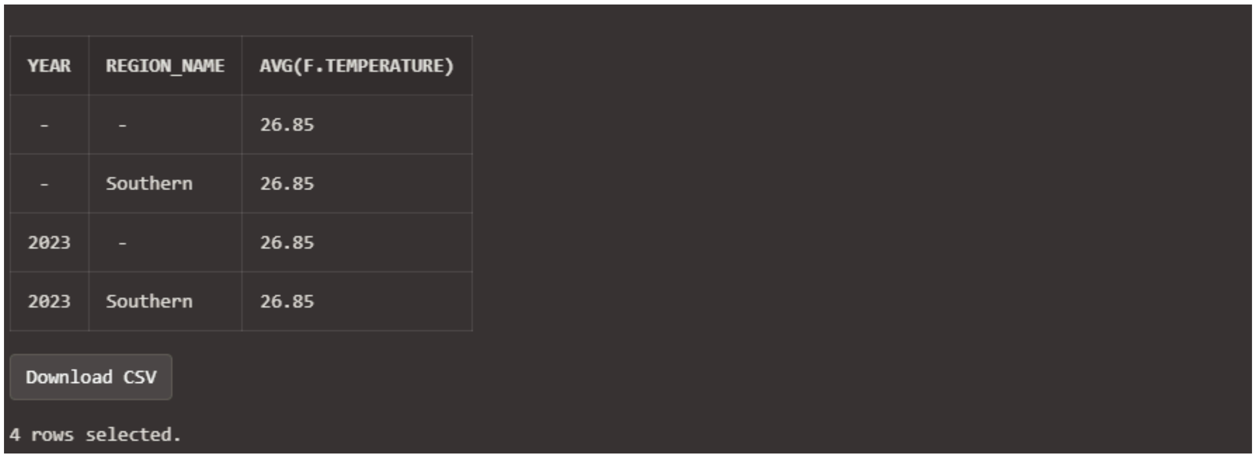
```
-- Create indexes
CREATE INDEX idx_fact_probe ON fact_weather_data(probe_id);
CREATE INDEX idx_fact_time ON fact_weather_data(time_id);
CREATE INDEX idx_fact_loc ON fact_weather_data(location_id);
-- Partition by year for large datasets
CREATE TABLE fact_weather_data_partitioned (
   record_id NUMBER GENERATED ALWAYS AS IDENTITY,
   probe_id NUMBER,
   time_id NUMBER,
   location_id NUMBER,
   temperature NUMBER(5,2),
   air_pressure NUMBER(7,2),
   precipitation NUMBER(6,2),
   CONSTRAINT fk_probe FOREIGN KEY (probe_id) REFERENCES dim_probe(probe_id),
   CONSTRAINT fk_time FOREIGN KEY (time_id) REFERENCES dim_time(time_id),
   CONSTRAINT fk_location FOREIGN KEY (location_id) REFERENCES dim_location(location_id)
)
PARTITION BY RANGE (time_id) (
   PARTITION p2022 VALUES LESS THAN (202300),
   PARTITION p2023 VALUES LESS THAN (202400),
   PARTITION pmax VALUES LESS THAN (MAXVALUE)
);
```

```
Index created.

Index created.

Index created.

Table created.
```

6. Oracle-Specific OLAP Features

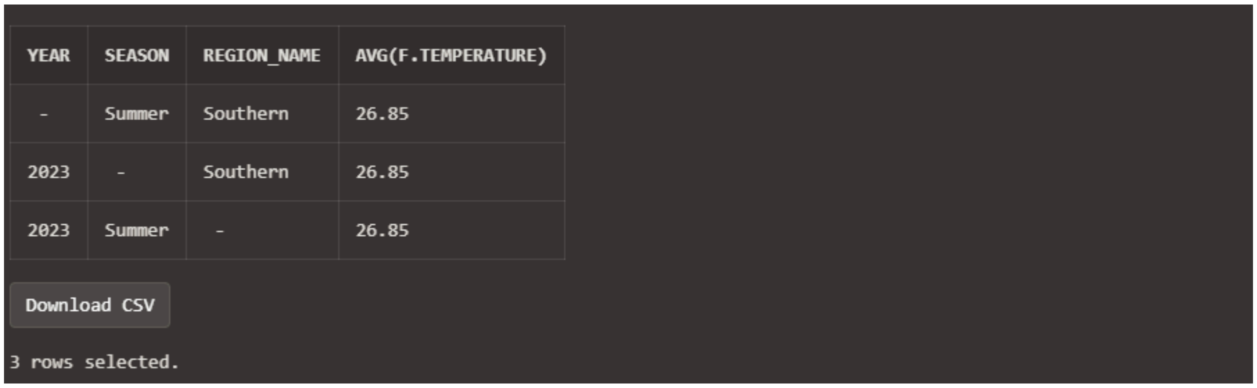
```
-- Using CUBE for multidimensional analysis
SELECT t.year, l.region_name, AVG(f.temperature)
FROM fact_weather_data f
JOIN dim_time t ON f.time_id = t.time_id
JOIN dim_location l ON f.location_id = l.location_id
GROUP BY CUBE(t.year, l.region_name);
-- Using GROUPING SETS
```

```
SELECT t.year, t.season, l.region_name, AVG(f.temperature)
FROM fact_weather_data f
JOIN dim_time t ON f.time_id = t.time_id
JOIN dim_location l ON f.location_id = l.location_id
GROUP BY GROUPING SETS (
  (t.year, t.season),
  (t.year, l.region_name),
  (t.season, l.region_name)
);
```



YEAR	REGION_NAME	AVG(F.TEMPERATURE)
-	-	26.85
-	Southern	26.85
2023	-	26.85
2023	Southern	26.85

```
4 rows selected.
```



YEAR	SEASON	REGION_NAME	AVG(F.TEMPERATURE)
-	Summer	Southern	26.85
2023	-	Southern	26.85
2023	Summer	-	26.85

```
3 rows selected.
```

Conclusion

This Oracle Live SQL implementation provides a complete OLAP solution for weather data analysis, featuring:

- Star Schema with fact and dimension tables
- All OLAP operations (Slice, Dice, Roll-Up, Drill-Down, Pivot)
- Oracle optimizations (Materialized Views, Partitioning, PIVOT, CUBE)
- Cleanup script for easy reset

Benefits:
Enables trend analysis (e.g., climate patterns)
Supports anomaly detection (e.g., extreme weather events)
Optimized for fast queries on large datasets
Ideal for meteorologists, researchers, and data analysts needing interactive weather data exploration.

Review Questions:-

1. What is the difference between the Slice and Dice operations in OLAP?

Slice Operation

- Definition: Reduces the data cube's dimensionality by selecting a single value along one dimension.
- Effect: Converts an *n*-dimensional cube into an *(n-1)*-dimensional subset.
- Example:
  - Original: 3D cube (Time × Location × Probe).
  - After Slicing: 2D table (Time × Location) for Probe #45 only.
- Use Case: Analyzing performance of a specific weather probe over time.

Dice Operation

- Definition: Creates a smaller subcube by applying constraints on multiple dimensions.
- Effect: Filters data along 2+ axes simultaneously.
- Example:
  - Conditions: "Summer 2023" + "Coastal probes" + "Temperature > 25°C".
  - Output: A 3D subcube meeting all criteria.
- Use Case: Investigating heatwaves in coastal regions during specific seasons.

Key Difference:

Slice	**Dice**
Cuts one dimension	Cuts multiple dimensions
Result: 2D table	Result: Smaller *n*-D cube

2. How does the Roll-up operation help in summarizing large volumes of data?

How It Works:

- Aggregation: Combines detailed data into higher-level summaries (e.g., daily → monthly).
- Hierarchy Navigation: Climbs from granular (hourly) to broad (yearly) levels.
- Functions Used: AVG(), SUM(), COUNT(), etc.

Example:

- Raw Data: Hourly temperature readings (8,760 records/year).
- Roll-up:
- sql
- Copy
- Download

SELECT year, AVG(temperature)
FROM weather_data

- GROUP BY year; -- Output: 10 rows for 10 years
- Benefit: Reveals decadal climate trends (e.g., global warming) that raw data obscures.

Why It Matters:

- Performance: Faster queries on aggregated data.
- Insights: Identifies macro-level patterns (e.g., "2020s averaged 2°C warmer than 2010s").

3. Give an example of a scenario where Pivoting the data provides a clearer insight than a traditional tabular view?

Scenario: Comparing seasonal temperature variations across regions.

Traditional Tabular View (Less Clear):

Region	Season	Avg Temp
Coastal	Summer	28°C
Coastal	Winter	12°C
Inland	Summer	35°C
Inland	Winter	8°C

Region	Summer	Winter
Coastal	28°C	12°C
Inland	35°C	8°C

Why Pivoting Helps:

- Visual Comparison: Easily spot contrasts (e.g., inland summers are hotter).
- Reduced Complexity: Avoids repeating dimension values (e.g., "Coastal" appears once).
- Analysis-Friendly: Enables side-by-side seasonal comparisons.

Real-World Use:

- Weather Reports: Show monthly rainfall as columns (Jan-Dec) with regions as rows.
- Business Analytics: Compare product sales (rows) across quarters (columns).