**Practical No 5**

**Q 1. Create a query to perform roll up operation using av.sales\_fact table.**

**(Note:-examine all data)**

**A.**

SELECT

month\_id,

category\_id,

state\_province\_id,

SUM(sales) AS total\_sales,

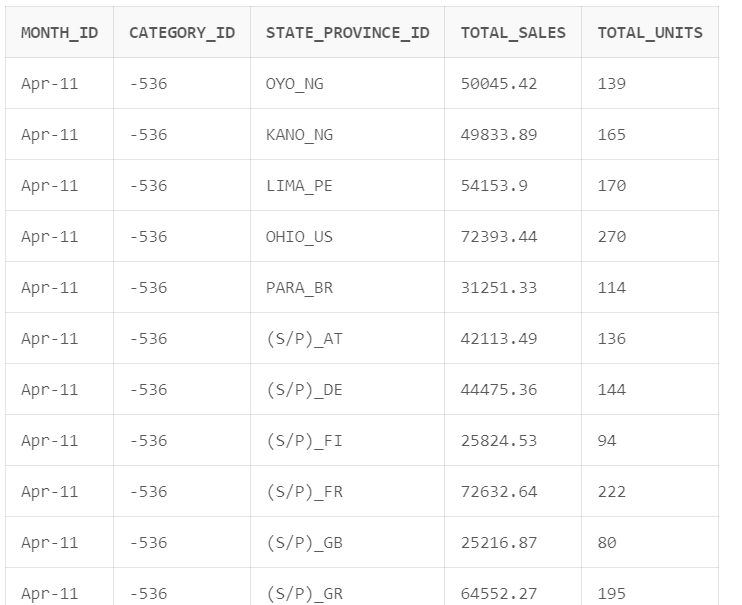
SUM(units) AS total\_units

FROM

av.sales\_fact

GROUP BY

ROLLUP (month\_id, category\_id, state\_province\_id);



**Q 2. Create a query to perform cube operation using av.sales\_fact table. (Note: Also examine the data)**

**A.**

SELECT

month\_id,

category\_id,

state\_province\_id,

SUM(sales) AS total\_sales,

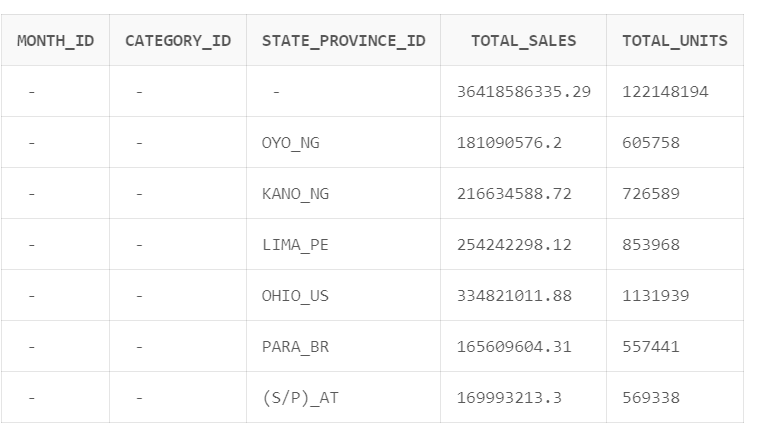
SUM(units) AS total\_units

FROM

av.sales\_fact

GROUP BY

CUBE(month\_id, category\_id, state\_province\_id);

****

**Q 3. Why we use roll up, cube and pivot function and what are the differences between them.**

**A.** Roll-up, cube, and pivot functions in SQL serve distinct purposes for data summarization. Roll-up is utilized to generate subtotals and a grand total for a specified set of columns, providing a hierarchical view. Cube, similar to roll-up, extends this by calculating subtotals and a grand total for all possible combinations of specified columns, offering a multidimensional perspective. Pivot, however, is employed to transform row data into a columnar format, enhancing readability, and is often used with aggregate functions for summarization. While roll-up and cube focus on hierarchical and dimensional analysis, providing subtotals at various levels, pivot primarily deals with restructuring data for better presentation. Each function is valuable in specific contexts, catering to diverse analytical needs.

Q 4. Perform pivot operation suing AV schema.

**A.**

SELECT

month\_id,

SUM(units) AS total\_units

FROM

av.sales\_fact

GROUP BY month\_id

ORDER BY month\_id;

|  |  |
| --- | --- |
| **MONTH\_ID** | **TOTAL\_UNITS** |
| Apr-11 | 2002303 |
| Apr-12 | 1998655 |
| Apr-13 | 2002657 |
| Apr-14 | 2003542 |
| Apr-15 | 2001030 |
| Aug-11 | 2042569 |
| Aug-12 | 2043836 |
| Aug-13 | 2043454 |
| Aug-14 | 2043335 |
| Aug-15 | 2043437 |
| Dec-11 | 2100209 |
| Dec-12 | 2100892 |
| Dec-13 | 2101709 |
| Dec-14 | 2101171 |
| Dec-15 | 2101992 |
| Feb-11 | 1866893 |
| Feb-12 | 1804964 |
| Feb-13 | 1804810 |
| Feb-14 | 1803959 |
| Feb-15 | 1869027 |
| Jan-11 | 1993656 |
| Jan-12 | 1990711 |
| Jan-13 | 1990777 |
| Jan-14 | 1991853 |
| Jan-15 | 1993658 |
| Jul-11 | 2077429 |
| Jul-12 | 2078935 |
| Jul-13 | 2080226 |
| Jul-14 | 2076820 |
| Jul-15 | 2078469 |
| Jun-11 | 2062078 |
| Jun-12 | 2064009 |
| Jun-13 | 2062598 |
| Jun-14 | 2061538 |
| Jun-15 | 2064798 |
| Mar-11 | 2031777 |
| Mar-12 | 2030877 |
| Mar-13 | 2030850 |
| Mar-14 | 2032628 |
| Mar-15 | 2032913 |
| May-11 | 2105096 |
| May-12 | 2105387 |
| May-13 | 2107216 |
| May-14 | 2106202 |
| May-15 | 2106537 |
| Nov-11 | 2044448 |
| Nov-12 | 2046881 |
| Nov-13 | 2044461 |
| Nov-14 | 2045638 |
| Nov-15 | 2046639 |

Q 5. Create pivot operation using SH schema.

**A.**

SELECT

prod\_id,

SUM(quantity\_sold) AS total\_items\_sold

FROM

sh.profits

GROUP BY prod\_id

ORDER BY prod\_id;

|  |  |
| --- | --- |
| **PROD\_ID** | **TOTAL\_ITEMS\_SOLD** |
| 13 | 6002 |
| 14 | 6010 |
| 15 | 5764 |
| 16 | 6928 |
| 17 | 6160 |
| 18 | 9589 |
| 19 | 10430 |
| 20 | 10903 |
| 21 | 4913 |
| 22 | 3441 |
| 23 | 19642 |
| 24 | 20948 |
| 25 | 19557 |
| 26 | 15651 |
| 27 | 11979 |
| 28 | 16796 |
| 29 | 7197 |
| 30 | 28979 |
| 31 | 23108 |
| 32 | 11253 |

Q 6. Create a view using cube operation.

**A.**

CREATE VIEW cube\_view AS

SELECT

month\_id,

category\_id,

SUM(units) AS total\_units

FROM

av.sales\_fact

GROUP BY

CUBE(month\_id, category\_id);