List partitioning and range partitioning are two strategies in horizontal fragmentation in distributed database system. They split a large table into multiple smaller tables according to conditions. Partitioning improves query performance, provides efficient scanning, speeds up batch operations, and saves storage.

List partitioning:

List partitioning divides the table into partitions based on the key value(s) appear in each partition. Each partition contains rows that match one of the key values. When inserting data into the main table, system will automatically insert the data into the partition with corresponding key. It is important to note that if the inserted data has a key that does not belong to any sub-tables, the data will not be inserted.

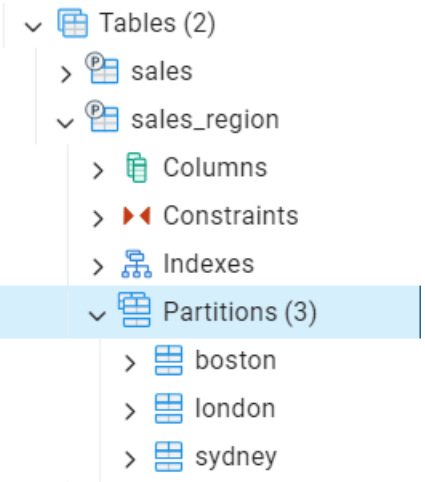
In this assignment, the sales_region table is divided into three regions: London, Boston, and Sydney. The structure of the table and the result of the query for all tables are shown below. Also, the query that generate partitions.

```
Sales Region Data:
(1, 337, 'London')
(2, 728, 'Boston')
(3, 181, 'London')
(4, 319, 'Boston')
(5, 740, 'London')
(6, 714, 'London')
(7, 182, 'Boston')
(8, 509, 'Boston')
(9, 493, 'Boston')
(10, 470, 'Sydney')
```

```
Boston Data:
(2, 728, 'Boston')
(4, 319, 'Boston')
(7, 182, 'Boston')
(8, 509, 'Boston')
(9, 493, 'Boston')
(11, 144, 'Boston')
(17, 288, 'Boston')
(19, 398, 'Boston')
(22, 339, 'Boston')
(24, 918, 'Boston')
```

```
Sydney Data:
(10, 470, 'Sydney')
(12, 647, 'Sydney')
(16, 669, 'Sydney')
(18, 926, 'Sydney')
(20, 496, 'Sydney')
(26, 277, 'Sydney')
(28, 154, 'Sydney')
(29, 529, 'Sydney')
(32, 123, 'Sydney')
(36, 442, 'Sydney')
```

```
London Data:
(1, 337, 'London')
(3, 181, 'London')
(5, 740, 'London')
(6, 714, 'London')
(13, 487, 'London')
(14, 483, 'London')
(15, 452, 'London')
(21, 972, 'London')
(23, 563, 'London')
(27, 282, 'London')
```




```
1  create table sales_region(
2      id int,
3      amount int,
4      region text
5  )partition by list (region);
6  create table boston partition of sales_region for values in('Boston');
7  create table london partition of sales_region for values in('London');
8  create table sydney partition of sales_region for values in('Sydney');
```

Data Output | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 45 msec.
```

Range Partitioning:

Range partitioning divides the table into partitions based on the value ranges of a key column. Each partition contains rows in the same range, and each range is inclusive at the lower end and exclusive at the upper end. Data with key in the range will be inserted into the corresponding table. Similarly, data that does not fall within any of the ranges will not be inserted.

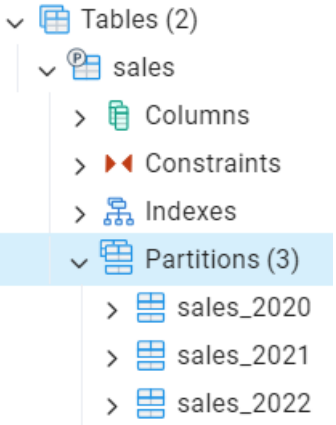
In this assignment, the sales table is divided into three partitions based on date range, recording data for the years 2020, 2021, and 2022 respectively. The structure of the table and the result of the query for all tables are shown below. Also, the query that generate partitions.

```
Sales Data:
(1, 'Product_C', 55, datetime.date(2022, 1, 30))
(2, 'Product_A', 6, datetime.date(2020, 12, 31))
(3, 'Product_B', 14, datetime.date(2020, 4, 6))
(4, 'Product_E', 52, datetime.date(2020, 4, 18))
(5, 'Product_A', 71, datetime.date(2021, 10, 1))
(6, 'Product_A', 6, datetime.date(2021, 3, 25))
(7, 'Product_D', 85, datetime.date(2022, 9, 21))
(8, 'Product_A', 45, datetime.date(2021, 1, 19))
(9, 'Product_B', 68, datetime.date(2020, 11, 14))
(10, 'Product_C', 25, datetime.date(2021, 2, 28))
```

```
Sales 2020 Data:
(2, 'Product_A', 6, datetime.date(2020, 12, 31))
(3, 'Product_B', 14, datetime.date(2020, 4, 6))
(4, 'Product_E', 52, datetime.date(2020, 4, 18))
(9, 'Product_B', 68, datetime.date(2020, 11, 14))
(13, 'Product_B', 69, datetime.date(2020, 7, 20))
(15, 'Product_B', 1, datetime.date(2020, 12, 24))
(20, 'Product_C', 53, datetime.date(2020, 2, 26))
(21, 'Product_C', 2, datetime.date(2020, 4, 11))
(23, 'Product_E', 71, datetime.date(2020, 1, 11))
(30, 'Product_B', 99, datetime.date(2020, 10, 31))
```

```
Sales 2021 Data:
(5, 'Product_A', 71, datetime.date(2021, 10, 1))
(6, 'Product_A', 6, datetime.date(2021, 3, 25))
(8, 'Product_A', 45, datetime.date(2021, 1, 19))
(10, 'Product_C', 25, datetime.date(2021, 2, 28))
(11, 'Product_A', 9, datetime.date(2021, 6, 26))
(14, 'Product_A', 43, datetime.date(2021, 2, 13))
(17, 'Product_E', 67, datetime.date(2021, 10, 24))
(22, 'Product_C', 26, datetime.date(2021, 7, 19))
(26, 'Product_E', 71, datetime.date(2021, 3, 22))
(27, 'Product_D', 39, datetime.date(2021, 12, 2))
```

```
Sales 2022 Data:
(1, 'Product_C', 55, datetime.date(2022, 1, 30))
(7, 'Product_D', 85, datetime.date(2022, 9, 21))
(12, 'Product_D', 69, datetime.date(2022, 11, 12))
(16, 'Product_E', 63, datetime.date(2022, 10, 17))
(18, 'Product_C', 48, datetime.date(2022, 7, 12))
(19, 'Product_E', 92, datetime.date(2022, 9, 5))
(24, 'Product_B', 82, datetime.date(2022, 1, 22))
(25, 'Product_E', 63, datetime.date(2022, 2, 2))
(29, 'Product_E', 13, datetime.date(2022, 4, 9))
(31, 'Product_B', 65, datetime.date(2022, 2, 6))
```




```
1  create table sales(
2      id int,
3      product_name text,
4      amount int,
5      sale_date date
6  )partition by range (sale_date);
7  create table sales_2020 partition of sales for values from ('2020-01-01') to ('2021-01-01');
8  create table sales_2021 partition of sales for values from ('2021-01-01') to ('2022-01-01');
9  create table sales_2022 partition of sales for values from ('2022-01-01') to ('2023-01-01');
```

Data Output | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 44 msec.
```