Generate aggregation & change data type

```
SELECT CAST(AVG(<column_name>) AS INT) AS
<new_column_name>
FROM <table>;
```

Retrieve only the first occurrence of a value

```
WITH <cte> AS (
  SELECT <column_names>,
  ROW_NUMBER() OVER (PARTITION BY
  <column_name> ORDER BY <column_name>)
  AS <row_number>
  FROM <table>
)
SELECT <column_names>
FROM <cte>
WHERE <row_number> = 1;
```

Calculate the median of a column

```
SELECT
(
  (SELECT MAX(<column_name>)
  FROM
   (SELECT TOP 50 PERCENT <column_name>
   FROM <table> ORDER BY <column_name>)
   AS <bottomhalf>)
+
  (SELECT MIN(<column_name>)
  FROM
   (SELECT top 50 percent <column_name>
   FROM <table>
   ORDER BY <column_name> DESC)
   AS <tophalf>)
) / 2
AS <median>;
```

Categorization of distinct column values

```
SELECT <column_names>,
  CASE
    WHEN <column_name> = 'value_01' THEN
1
    WHEN <column_name> = 'value_02' THEN
2
    ELSE 3
  END <new_column_name>
FROM <table>;
```

Append two tables with the first occurrence of a row only

```
INSERT INTO <table_1>
SELECT <column_names>
FROM <table_2> B
WHERE NOT EXISTS (
  SELECT 1
  FROM <table_1> A
    WHERE B.<column_1> = A.<column_1>
    AND
    B.<column_2> = A.<column_2>
);
```

Append two tables and retrieve distinct rows only

```
SELECT <column_names>
FROM <table_1>
UNION
SELECT <column_names>
FROM <table_2>;
```

Retrieve rows based on conditions from another table

```
SELECT <column_1>, <column_2>
FROM <table_1>
WHERE (<column_2> =
        (SELECT MAX(<column_3>)
         FROM <table_2>)
);
```

Pivot a table with 2 columns and 2 rows

```
WITH <cte_1> AS (
  SELECT <column_1>,
    MAX(
      CASE
        WHEN <column_2> = 'value_1'
        THEN <column_1>
      END)
      OVER (PARTITION BY <column_1>
      ORDER BY <column_2> DESC)
      AS <new_column_1>,
    MAX(
      CASE
        WHEN <column_2> = 'value_2'
        THEN <column_1>
      END)
      OVER (PARTITION BY <column_1>
      ORDER BY <column_2> DESC)
      AS <new_column_2>
  FROM <table>
),
<cte_2> AS (
  SELECT ROW_NUMBER() OVER
    (ORDER BY <column_1>)
     AS <index_column>, <new_column_1>
  FROM <cte_1>
  WHERE <new_column_1> IS NOT NULL
),
<cte_3> AS (
  SELECT ROW_NUMBER() OVER
    (ORDER BY <column_1>)
     AS <index_column>, <new_column_2>
  FROM <cte_1>
  WHERE <new_column_2> IS NOT NULL
)
SELECT B.<new_column_1>, A.<new_column_2>
FROM <cte_3> A
LEFT JOIN
<cte_2> B
ON A.<index_column> = B.<index_column>
;
```

Legend

Title = Use Case of SQL Statement
BLUE = Standard SQL Command
PINK = Aggregation SQL Command
<red> = Column or Table Name