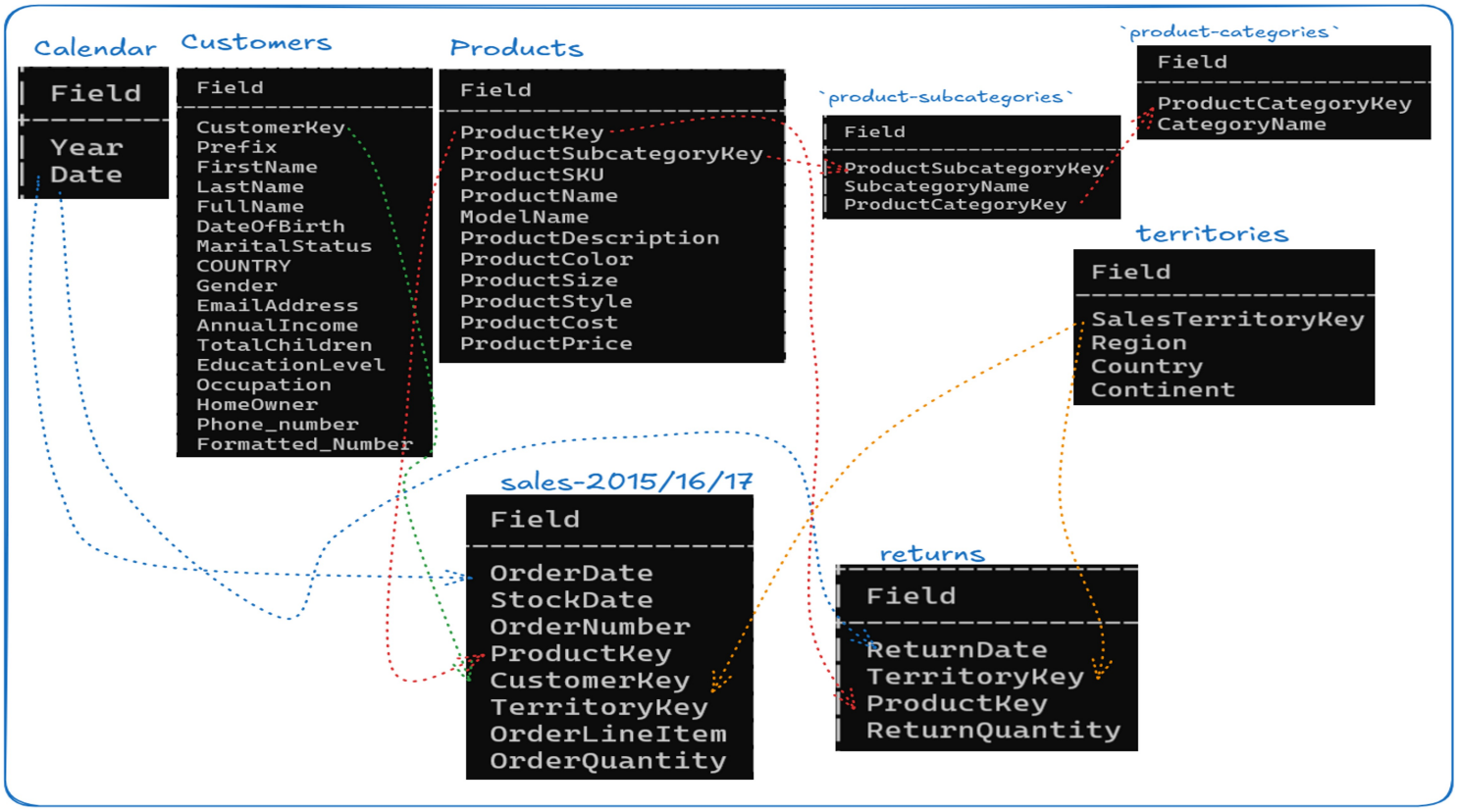
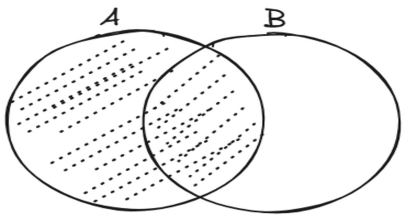
Advanced Concepts in Joins

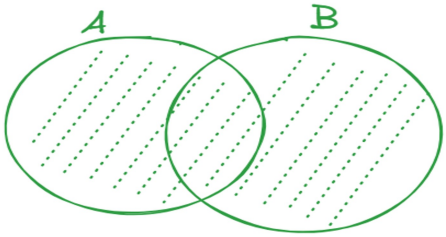


FULL JOIN

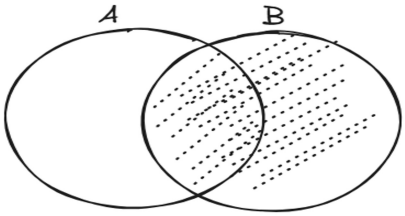


LEFT JOIN

A ∪ B



UNION ALL



RIGHT JOIN

```sql
-- UNION ALL [Append Queries]
-- FULL JOIN []
SELECT
    t.*,
    r.*
FROM territories t
LEFT JOIN returns r
ON t.SalesTerritoryKey = r.TerritoryKey
UNION ALL
SELECT
    t.*,
    r.*
FROM territories t
RIGHT JOIN returns r
ON t.SalesTerritoryKey = r.TerritoryKey;
```

UNION ALL [No Duplicates checked].

```sql
SELECT
    t.*,
    r.*
FROM territories t
LEFT JOIN returns r
ON t.SalesTerritoryKey = r.TerritoryKey
UNION
SELECT
    t.*,
    r.*
FROM territories t
RIGHT JOIN returns r
ON t.SalesTerritoryKey = r.TerritoryKey;
```

UNION [Remove Duplicates]

Challenge - 1

Display full customer names and the names of the products they bought by combining sales from all 3 years.

```
mysql> DESC Customers;
+------------------+--
| Field            | T
+------------------+--
| CustomerKey      | i
| Prefix           | t
| FirstName        | t
| LastName         | t
| FullName         | v
| DateOfBirth      | t
| MaritalStatus    | t
| COUNTRY          | v
| Gender           | t
| EmailAddress     | v
| AnnualIncome     | i
| TotalChildren    | i
| EducationLevel   | t
| Occupation       | t
| HomeOwner        | t
| Phone_number     | b
| Formatted_Number | v
+------------------+--
```

```
mysql> DESC Products;
+-------------------------+
| Field                   |
+-------------------------+
| ProductKey              |
| ProductSubcategoryKey   |
| ProductSKU              |
| ProductName             |
| ModelName               |
| ProductDescription      |
| ProductColor            |
| ProductSize             |
| ProductStyle            |
| ProductCost             |
| ProductPrice            |
+-------------------------+
```

```
mysql> DESC `sales-2015`
+---------------+------+
| Field         | Type |
+---------------+------+
| OrderDate     | text |
| StockDate     | text |
| OrderNumber   | text |
| ProductKey    | int  |
| CustomerKey   | int  |
| TerritoryKey  | int  |
| OrderLineItem | int  |
| OrderQuantity | int  |
+---------------+------+
```

```
10481 rows in set (0.48 sec)
```

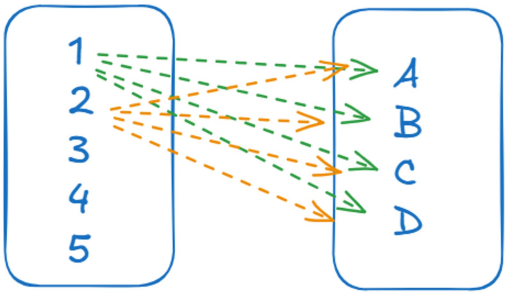
Picking common values from all 3 tables.

```sql
SELECT
    CONCAT(c.FirstName , " ", c.LastName) AS CustomerName,
    p.ProductName,
    s.OrderQuantity
FROM (
    SELECT * FROM `sales-2015`
    UNION ALL
    SELECT * FROM `sales-2016`
    UNION ALL
    SELECT * FROM `sales-2017`
) AS s
INNER JOIN Customers AS c
ON s.CustomerKey = c.CustomerKey
INNER JOIN Products AS p
ON s.ProductKey = p.ProductKey;
```

```
56046 rows in set (0.13 sec)
```

CROSS JOIN

Cardinality of M:M



Cross join : every record from left table will match with every record from right table

1A, 1B, 1C, 1D
2A, 2B, 2C, 2D
3A, 3B, 3C, 3D
4A, 4B, 4C, 4D
5A, 5B, 5C, 5D

Rule of Normalization

```
mysql> SELECT * FROM `product-categories`;
+---------------------+---------------+
| ProductCategoryKey | CategoryName |
+---------------------+---------------+
|                   1 | Bikes         |
|                   2 | Components    |
|                   3 | Clothing      |
|                   4 | Accessories   |
+---------------------+---------------+
4 rows in set (0.04 sec)
```

```
mysql> SELECT 37 * 4;
+--------+
| 37 * 4 |
+--------+
|    148 |
+--------+
1 row in set (0.00 sec)
```

```
mysql> SELECT * FROM `product-subcategories`;
+----------------------+--------------------+---------------------+
| ProductSubcategoryKey | SubcategoryName    | ProductCategoryKey |
+----------------------+--------------------+---------------------+
|                    1 | Mountain Bikes     |                   1 |
|                    2 | Road Bikes         |                   1 |
|                    3 | Touring Bikes      |                   1 |
|                    4 | Handlebars         |                   2 |
|                    5 | Bottom Brackets    |                   2 |
|                    6 | Brakes             |                   2 |
|                    7 | Chains             |                   2 |
|                    8 | Cranksets          |                   2 |
|                    9 | Derailleurs        |                   2 |
|                   10 | Forks              |                   2 |
|                   11 | Headsets           |                   2 |
|                   12 | Mountain Frames    |                   2 |
|                   13 | Pedals             |                   2 |
|                   14 | Road Frames        |                   2 |
|                   15 | Saddles            |                   2 |
|                   16 | Touring Frames     |                   2 |
|                   17 | Wheels             |                   2 |
|                   18 | Bib-Shorts         |                   3 |
|                   19 | Caps               |                   3 |
|                   20 | Gloves             |                   3 |
|                   21 | Jerseys            |                   3 |
|                   22 | Shorts             |                   3 |
|                   23 | Socks              |                   3 |
|                   24 | Tights             |                   3 |
|                   25 | Vests              |                   3 |
|                   26 | Bike Racks         |                   4 |
|                   27 | Bike Stands        |                   4 |
|                   28 | Bottles and Cages  |                   4 |
|                   29 | Cleaners           |                   4 |
|                   30 | Fenders            |                   4 |
|                   31 | Helmets            |                   4 |
|                   32 | Hydration Packs    |                   4 |
|                   33 | Lights             |                   4 |
|                   34 | Locks              |                   4 |
|                   35 | Panniers           |                   4 |
|                   36 | Pumps              |                   4 |
|                   37 | Tires and Tubes    |                   4 |
+----------------------+--------------------+---------------------+
37 rows in set (0.01 sec)
```

```
mysql> SELECT *
    -> FROM
    ->  `product-categories`
    -> CROSS JOIN `product-subcategories`;
```

ProductCategoryKey	CategoryName	ProductSubcategoryKey	SubcategoryName	ProductCategoryKey
4	Accessories	1	Mountain Bikes	1
3	Clothing	1	Mountain Bikes	1
2	Components	1	Mountain Bikes	1
1	Bikes	1	Mountain Bikes	1
4	Accessories	2	Road Bikes	1
3	Clothing	2	Road Bikes	1
2	Components	2	Road Bikes	1
1	Bikes	2	Road Bikes	1
4	Accessories	3	Touring Bikes	1
3	Clothing	3	Touring Bikes	1
2	Components	3	Touring Bikes	1
1	Bikes	3	Touring Bikes	1
4	Accessories	4	Handlebars	2
3	Clothing	4	Handlebars	2

```
148 rows in set (0.01 sec)
```

```
mysql> SELECT *
    -> FROM
    ->  `product-categories`pc
    -> CROSS JOIN `product-subcategories`ps
    -> WHERE pc.ProductCategoryKey = ps.ProductCategoryKey
    -> ORDER BY pc.ProductCategoryKey;
```

ProductCategoryKey	CategoryName	ProductSubcategoryKey	SubcategoryName	ProductCategoryKey
1	Bikes	1	Mountain Bikes	1
1	Bikes	2	Road Bikes	1
1	Bikes	3	Touring Bikes	1
2	Components	4	Handlebars	2
2	Components	5	Bottom Brackets	2
2	Components	6	Brakes	2
2	Components	7	Chains	2
2	Components	8	Cranksets	2
2	Components	9	Derailleurs	2
2	Components	10	Forks	2
2	Components	11	Headsets	2
2	Components	12	Mountain Frames	2
2	Components	13	Pedals	2
2	Components	14	Road Frames	2
2	Components	15	Saddles	2
2	Components	16	Touring Frames	2
2	Components	17	Wheels	2
3	Clothing	18	Bib-Shorts	3
3	Clothing	19	Caps	3
3	Clothing	20	Gloves	3
3	Clothing	21	Jerseys	3
3	Clothing	22	Shorts	3
3	Clothing	23	Socks	3
3	Clothing	24	Tights	3
3	Clothing	25	Vests	3
4	Accessories	26	Bike Racks	4
4	Accessories	27	Bike Stands	4
4	Accessories	28	Bottles and Cages	4
4	Accessories	29	Cleaners	4
4	Accessories	30	Fenders	4
4	Accessories	31	Helmets	4
4	Accessories	32	Hydration Packs	4
4	Accessories	33	Lights	4
4	Accessories	34	Locks	4
4	Accessories	35	Panniers	4
4	Accessories	36	Pumps	4
4	Accessories	37	Tires and Tubes	4

```
37 rows in set (0.01 sec)
```

```sql
SELECT *
FROM
    `product-categories`pc
CROSS JOIN `product-subcategories`ps
WHERE pc.ProductCategoryKey = ps.ProductCategoryKey
ORDER BY pc.ProductCategoryKey;
```

SELF- JOIN

- It connects itself with different aliases to do self-join, for finding some patterns or insights.

```
mysql> SELECT FirstName, LastName, Gender,Occupation,TotalChildren
    -> FROM Customers LIMIT 10;
+-----------+----------+--------+-------------+---------------+
| FirstName | LastName | Gender | Occupation  | TotalChildren |
+-----------+----------+--------+-------------+---------------+
| JON       | YANG     | M      | Professional |             2 |
| EUGENE    | HUANG    | M      | Professional |             3 |
| RUBEN     | TORRES   | M      | Professional |             3 |
| CHRISTY   | ZHU      | F      | Professional |             0 |
| ELIZABETH | JOHNSON  | F      | Professional |             5 |
| JULIO     | RUIZ     | M      | Professional |             0 |
| MARCO     | MEHTA    | M      | Professional |             3 |
| ROBIN     | VERHOFF  | F      | Professional |             4 |
| SHANNON   | CARLSON  | M      | Professional |             0 |
| JACQUELYN | SUAREZ   | F      | Professional |             0 |
+-----------+----------+--------+-------------+---------------+
10 rows in set (0.00 sec)
```

c1

```
mysql> SELECT FirstName, LastName, Gender,Occupation,TotalChildren
    -> FROM Customers LIMIT 10;
+-----------+----------+--------+-------------+---------------+
| FirstName | LastName | Gender | Occupation  | TotalChildren |
+-----------+----------+--------+-------------+---------------+
| JON       | YANG     | M      | Professional |             2 |
| EUGENE    | HUANG    | M      | Professional |             3 |
| RUBEN     | TORRES   | M      | Professional |             3 |
| CHRISTY   | ZHU      | F      | Professional |             0 |
| ELIZABETH | JOHNSON  | F      | Professional |             5 |
| JULIO     | RUIZ     | M      | Professional |             0 |
| MARCO     | MEHTA    | M      | Professional |             3 |
| ROBIN     | VERHOFF  | F      | Professional |             4 |
| SHANNON   | CARLSON  | M      | Professional |             0 |
| JACQUELYN | SUAREZ   | F      | Professional |             0 |
+-----------+----------+--------+-------------+---------------+
10 rows in set (0.00 sec)
```

c2

```sql
-- SELF JOIN
-- If I want to find a clan or community of same LastName.
SELECT
    c1.customerKey AS Customer1_ID,
    CONCAT(c1.FirstName, ' ', c1.LastName) as Customer1_Name,
    c2.customerKey AS Customer2_ID,
    CONCAT(c2.FirstName, ' ', c2.LastName) as Customer2_Name,
    c1.LastName
FROM customers c1
JOIN Customers c2
ON c1.LastName = c2.LastName
AND c1.customerKey < c2.customerKey
LIMIT 20;
```

Customer1_ID	Customer1_Name	Customer2_ID	Customer2_Name	LastName
11000	JON YANG	11000	JON YANG	YANG
11000	JON YANG	11110	CURTIS YANG	YANG
11000	JON YANG	12001	SHANNON YANG	YANG
11000	JON YANG	12117	CAMERON YANG	YANG
11000	JON YANG	12139	FRANKLIN YANG	YANG
11000	JON YANG	12396	SUSAN YANG	YANG
11000	JON YANG	12564	MARSHALL YANG	YANG
11000	JON YANG	12566	OMAR YANG	YANG
11000	JON YANG	12786	JAMES YANG	YANG
11001	EUGENE HUANG	11001	EUGENE HUANG	HUANG
11001	EUGENE HUANG	11220	ERICA HUANG	HUANG
11001	EUGENE HUANG	11464	ALEJANDRO HUANG	HUANG
11001	EUGENE HUANG	11515	SHANNON HUANG	HUANG
11001	EUGENE HUANG	11578	TRISHA HUANG	HUANG
11001	EUGENE HUANG	11943	DAWN HUANG	HUANG
11001	EUGENE HUANG	11945	DENNIS HUANG	HUANG
11001	EUGENE HUANG	11951	LACEY HUANG	HUANG
11001	EUGENE HUANG	12244	AMY HUANG	HUANG
11001	EUGENE HUANG	12463	TODD HUANG	HUANG
11001	EUGENE HUANG	12491	ALVIN HUANG	HUANG

```
mysql> SELECT
    ->  c1.customerKey AS Customer1_ID,
    ->     CONCAT(c1.FirstName, ' ', c1.LastName) as Customer1_Name,
    ->     c1.AnnualIncome AS customer1_Income,
    ->     c2.customerKey AS Customer2_ID,
    ->     CONCAT(c2.FirstName, ' ', c2.LastName) as Customer2_Name,
    ->     c2.AnnualIncome AS customer2_Income
    -> FROM customers c1
    -> JOIN Customers c2
    -> ON c1.AnnualIncome = c2.AnnualIncome
    -> AND c1.customerKey < c2.customerKey
    -> ORDER BY c2.AnnualIncome DESC
    -> LIMIT 20;
```

Customer1_ID	Customer1_Name	customer1_Income	Customer2_ID	Customer2_Name	customer2_Income
11080	DAMIEN CHANDER	170000	11181	DEVIN MARTIN	170000
11080	DAMIEN CHANDER	170000	11244	ALEXIS COLEMAN	170000
11181	DEVIN MARTIN	170000	11244	ALEXIS COLEMAN	170000
11080	DAMIEN CHANDER	170000	11250	SHANNON LIU	170000
11181	DEVIN MARTIN	170000	11250	SHANNON LIU	170000
11244	ALEXIS COLEMAN	170000	11250	SHANNON LIU	170000
11080	DAMIEN CHANDER	170000	11286	HUNTER GRIFFIN	170000
11181	DEVIN MARTIN	170000	11286	HUNTER GRIFFIN	170000
11244	ALEXIS COLEMAN	170000	11286	HUNTER GRIFFIN	170000
11250	SHANNON LIU	170000	11286	HUNTER GRIFFIN	170000
11080	DAMIEN CHANDER	170000	11422	DUSTIN DENG	170000
11181	DEVIN MARTIN	170000	11422	DUSTIN DENG	170000
11244	ALEXIS COLEMAN	170000	11422	DUSTIN DENG	170000
11250	SHANNON LIU	170000	11422	DUSTIN DENG	170000
11286	HUNTER GRIFFIN	170000	11422	DUSTIN DENG	170000
11080	DAMIEN CHANDER	170000	11434	ANDRE LOPEZ	170000
11181	DEVIN MARTIN	170000	11434	ANDRE LOPEZ	170000
11244	ALEXIS COLEMAN	170000	11434	ANDRE LOPEZ	170000
11250	SHANNON LIU	170000	11434	ANDRE LOPEZ	170000
11286	HUNTER GRIFFIN	170000	11434	ANDRE LOPEZ	170000

```
mysql> SELECT
    ->  c1.customerKey AS Customer1_ID,
    ->     CONCAT(c1.FirstName, ' ', c1.LastName) as Customer1_Name,
    ->     c2.customerKey AS Customer2_ID,
    ->     CONCAT(c2.FirstName, ' ', c2.LastName) as Customer2_Name,
    ->     c1.DateOfBirth
    -> FROM customers c1
    -> JOIN Customers c2
    -> ON c1.DateOfBirth = c2.DateOfBirth
    -> AND c1.customerKey < c2.customerKey
    -> LIMIT 20;
+-------------+------------------+-------------+--------------------+-------------+
| Customer1_ID | Customer1_Name    | Customer2_ID | Customer2_Name     | DateOfBirth |
+-------------+------------------+-------------+--------------------+-------------+
|        11004 | ELIZABETH JOHNSON |        11550 | DEB TORRES         | 08/08/1968  |
|        11010 | JACQUELYN SUAREZ  |        12002 | ADRIANA LOPEZ      | 02/06/1964  |
|        11017 | SHANNON WANG      |        12075 | CAMERON GRIFFIN    | 26/06/1944  |
|        11018 | CLARENCE RAI      |        12064 | JOHN WHITE         | 10/09/1944  |
|        11031 | THERESA RAMOS     |        12661 | DEBORAH PAL        | 22/08/1947  |
|        11041 | AMANDA CARTER     |        12029 | DAMIEN HU          | 16/10/1977  |
|        11053 | ANA PRICE         |        13039 | TRISHA MA          | 20/08/1980  |
|        11054 | DEANNA MUNOZ      |        12648 | LORI DOMINGUEZ     | 03/10/1952  |
|        11055 | GILBERT RAJE      |        11236 | JEREMY BUTLER      | 03/05/1952  |
|        11056 | MICHELE NATH      |        11312 | SARA RICHARDSON    | 04/03/1953  |
|        11068 | TIFFANY LIANG     |        12134 | HECTOR ALONSO      | 23/09/1955  |
|        11068 | TIFFANY LIANG     |        12770 | CHARLES EVANS      | 23/09/1955  |
|        11075 | FELICIA JIMENEZ   |        11529 | AUSTIN BRYANT      | 16/11/1957  |
|        11081 | SAVANNAH BAKER    |        11561 | BRIANA DOMINGUEZ   | 24/07/1966  |
|        11083 | ALYSSA COX        |        11572 | BILLY MORENO       | 15/03/1966  |
|        11091 | DALTON PEREZ      |        11982 | ALEXANDRA FOSTER   | 04/04/1957  |
|        11096 | ANDRÃ?S ANAND     |        11991 | FREDERICK MARTINEZ | 08/10/1962  |
|        11102 | JULIA NELSON      |        11575 | JACKSON CAMPBELL   | 21/04/1965  |
|        11105 | CANDACE FERNANDEZ |        12147 | TAYLOR PATTERSON   | 28/12/1964  |
|        11106 | JESSIE LIU        |        11263 | TRINITY RICHARDSON | 09/11/1964  |
+-------------+------------------+-------------+--------------------+-------------+
20 rows in set (0.16 sec)
```

```
|        12808 | MARC SCHMIDT      | 05/11/1968  |        12997 | NINA XIE           | 05/11/1968  |
|        12817 | THEODORE SUAREZ   | 02/09/1966  |        12820 | DEREK TANG         | 02/09/1966  |
|        12860 | JILLIAN PRASAD    | 25/03/1962  |        12861 | ROBYN JIMENEZ      | 25/03/1962  |
|        12928 | CAROL WRIGHT      | 11/10/1969  |        12932 | TYLER JOHNSON      | 11/10/1969  |
|        12940 | STEVEN ROGERS     | 24/03/1965  |        12945 | JESSICA BAILEY     | 24/03/1965  |
+-------------+------------------+-------------+--------------------+-------------+
231 rows in set (1.58 sec)
```

IF NULL() VS COALESCE()

- Both handles null value, but there is important difference in their working.

IFNULL(expr1, expr2).
- Takes exactly 2 expression.
- If Exp1 is Null, It picks Exp2 and vice - versa.
- It is simple for NULL Substitute.

COALESCE(expr1, expr2,..... exprN).
- Takes multiple expressions.
- It returns first occurrence of NON NULL Values from the list.

```
mysql> SELECT DISTINCT AnnualIncome FROM Customers;
+--------------+
| AnnualIncome |
+--------------+
|        90000 |
|        60000 |
|         NULL |
|        80000 |
|        70000 |
|       100000 |
|        30000 |
|        20000 |
|        40000 |
|        10000 |
|       160000 |
|       170000 |
|       130000 |
|       110000 |
|       120000 |
|       150000 |
|        50000 |
+--------------+
17 rows in set (0.01 sec)
```

```
mysql> SELECT
    ->  CustomerKey,
    ->     FullName,
    ->     AnnualIncome
    -> FROM Customers
    -> WHERE AnnualIncome IS NULL;
+-------------+------------------+--------------+
| CustomerKey | FullName         | AnnualIncome |
+-------------+------------------+--------------+
|       11003 | CHRISTY ZHU      |         NULL |
|       11015 | CHLOE YOUNG      |         NULL |
|       11081 | SAVANNAH BAKER   |         NULL |
|       11130 | CAROLINE RUSSELL |         NULL |
|       11227 | MARSHALL CHAVEZ  |         NULL |
|       11359 | JARROD PRASAD    |         NULL |
|       11552 | EDDIE RUBIO      |         NULL |
|       11673 | SUSAN YE         |         NULL |
|       12152 | ADRIAN RAMIREZ   |         NULL |
|       12756 | ANTONIO BUTLER   |         NULL |
+-------------+------------------+--------------+
10 rows in set (0.00 sec)
```

```
mysql> SELECT
    ->  CustomerKey,
    ->     FullName,
    ->     AnnualIncome,
    ->     IFNULL(AnnualIncome , "Not Available") As IncomeStatus
    -> FROM Customers
    -> WHERE AnnualIncome IS NULL;
+-------------+------------------+--------------+---------------+
| CustomerKey | FullName         | AnnualIncome | IncomeStatus  |
+-------------+------------------+--------------+---------------+
|       11003 | CHRISTY ZHU      |         NULL | Not Available |
|       11015 | CHLOE YOUNG      |         NULL | Not Available |
|       11081 | SAVANNAH BAKER   |         NULL | Not Available |
|       11130 | CAROLINE RUSSELL |         NULL | Not Available |
|       11227 | MARSHALL CHAVEZ  |         NULL | Not Available |
|       11359 | JARROD PRASAD    |         NULL | Not Available |
|       11552 | EDDIE RUBIO      |         NULL | Not Available |
|       11673 | SUSAN YE         |         NULL | Not Available |
|       12152 | ADRIAN RAMIREZ   |         NULL | Not Available |
|       12756 | ANTONIO BUTLER   |         NULL | Not Available |
+-------------+------------------+--------------+---------------+
10 rows in set (0.00 sec)
```

```
mysql> SELECT
    ->  CustomerKey,
    ->     FullName,
    ->     AnnualIncome,
    ->     IFNULL(AnnualIncome , "Not Available") As IncomeStatus
    -> FROM Customers
    -> LIMIT 10;
+-------------+------------------+--------------+---------------+
| CustomerKey | FullName         | AnnualIncome | IncomeStatus  |
+-------------+------------------+--------------+---------------+
|       11000 | JON YANG         |        90000 | 90000         |
|       11001 | EUGENE HUANG     |        60000 | 60000         |
|       11002 | RUBEN TORRES     |        60000 | 60000         |
|       11003 | CHRISTY ZHU      |         NULL | Not Available |
|       11004 | ELIZABETH JOHNSON |        80000 | 80000         |
|       11005 | JULIO RUIZ       |        70000 | 70000         |
|       11007 | MARCO MEHTA      |        60000 | 60000         |
|       11008 | ROBIN VERHOFF    |        60000 | 60000         |
|       11009 | SHANNON CARLSON  |        70000 | 70000         |
|       11010 | JACQUELYN SUAREZ |        70000 | 70000         |
+-------------+------------------+--------------+---------------+
10 rows in set (0.00 sec)
```

```
mysql> SELECT
    ->  CustomerKey,
    ->     FullName,
    ->     AnnualIncome,
    ->     COALESCE(AnnualIncome , 0) As IncomeStatus
    -> FROM Customers
    -> LIMIT 10;
+-------------+------------------+--------------+--------------+
| CustomerKey | FullName         | AnnualIncome | IncomeStatus |
+-------------+------------------+--------------+--------------+
|       11000 | JON YANG         |        90000 |        90000 |
|       11001 | EUGENE HUANG     |        60000 |        60000 |
|       11002 | RUBEN TORRES     |        60000 |        60000 |
|       11003 | CHRISTY ZHU      |         NULL |            0 |
|       11004 | ELIZABETH JOHNSON|        80000 |        80000 |
|       11005 | JULIO RUIZ       |        70000 |        70000 |
|       11007 | MARCO MEHTA      |        60000 |        60000 |
|       11008 | ROBIN VERHOFF    |        60000 |        60000 |
|       11009 | SHANNON CARLSON  |        70000 |        70000 |
|       11010 | JACQUELYN SUAREZ |        70000 |        70000 |
+-------------+------------------+--------------+--------------+
10 rows in set (0.00 sec)
```

```
mysql> SELECT
    ->  CustomerKey,
    ->     FullName,
    ->     AnnualIncome,
    ->     COALESCE(AnnualIncome , "Not Available") As IncomeStatus
    -> FROM Customers
    -> LIMIT 10;
+-------------+------------------+--------------+---------------+
| CustomerKey | FullName         | AnnualIncome | IncomeStatus  |
+-------------+------------------+--------------+---------------+
|       11000 | JON YANG         |        90000 | 90000         |
|       11001 | EUGENE HUANG     |        60000 | 60000         |
|       11002 | RUBEN TORRES     |        60000 | 60000         |
|       11003 | CHRISTY ZHU      |         NULL | Not Available |
|       11004 | ELIZABETH JOHNSON|        80000 | 80000         |
|       11005 | JULIO RUIZ       |        70000 | 70000         |
|       11007 | MARCO MEHTA      |        60000 | 60000         |
|       11008 | ROBIN VERHOFF    |        60000 | 60000         |
|       11009 | SHANNON CARLSON  |        70000 | 70000         |
|       11010 | JACQUELYN SUAREZ |        70000 | 70000         |
+-------------+------------------+--------------+---------------+
10 rows in set (0.00 sec)
```

COALESCE(exp1, exp2,exp3,..... expN).

COALESCE(AnnualIncome, previousYearIncome, 0).