

SQL SIMPLIFIED FOR ALL

By Bemnet Girma

[linkedin.com/in/bemnetdev](https://www.linkedin.com/in/bemnetdev)

15+ Topics

70+ Practice Queries

50 SQL Query Questions from 15+ Companies

110+ Frequently Asked SQL Interview Questions



Forbes



IN JUST 10 PAGES

KEYWORDS

Data is representation of raw facts, measurements, figures, or concepts in a formalized manner that have no specific meaning.

Information is processed (organized or classified) data, which has some meaningful values.

Database is an organized collection of data stored and accessed electronically in a computer system.

DBMS are software systems that enable users to store, retrieve, define and manage data in a database easily.

RDBMS is a type of DBMS that stores data in a row-based table structure which connects related data elements.

SQL is a database query language used for storing and managing data in RDBMS.

DATA TYPES

Data type is a data rule applicable to a particular column. NULL, CHAR, VARCHAR, INT, DATE, FLOAT, BOOLEAN

CONSTRAINTS

Constraints are limitations or restrictions applied to a column in a table, they are important to maintain data integrity among tables.

CHECK - control the values being inserted.

NOT NULL - ensure that NULL value has not inserted.

UNIQUE – ensure that every column value is unique.

PRIMARY KEY – ensure that all values are UNIQUE and NOT NULL

FOREIGN KEY - create a parent-child r/ship b/n tables

COMMANDS

Data Definition Language (DDL)

DCAT | DROP, CREATE, ALTER, TRUNCATE

Data Query Language (DQL)

SELECT

Data Manipulation Language (DML)

UMID | UPDATE, MERGE, INSERT, DELETE

Data Control Language (DCL)

GRANT and REVOKE

Transaction Control Language (TCL)

COMMIT, SAVEPOINT, ROLLBACK



PREREQUISITES

Watch the ff. video to install PostgreSQL DBMS
www.youtube.com/watch?v=CTOpjJPn9M&t=1058s

Use the ff. video to connect DB to Jupyter notebook
www.youtube.com/watch?v=LhKj-CCfY

You can get all the queries in my Github repo
www.github.com/bemnetdev/SQL/blob/main/s.ipynb



CREATE TABLE

CREATE - to create a new database object such as a table.

```
CREATE TABLE department (  
    did varchar(20),  
    name varchar(20) NOT NULL,  
    CONSTRAINT PK_DEPT PRIMARY KEY(did)  
);
```

INSERT DATA

```
INSERT INTO department VALUES  
( 'D1', 'Management'),  
( 'D2', 'IT'),  
( 'D3', 'Sales'),  
( 'D4', 'HR')
```

did	name
D1	Management
D2	IT
D3	Sales
D4	HR

CREATE “employee” TABLE

```
CREATE TABLE employee (  
    eid int,  
    name varchar(20) UNIQUE,  
    join_date date NOT NULL,  
    department char(2)  
    CHECK (dep IN ('D1', 'D2', 'D3')),  
    salary int,  
    manager int,  
    CONSTRAINT PK_ID PRIMARY KEY(eid),  
    CONSTRAINT FK_DID FOREIGN KEY(department)  
    REFERENCES department(did)  
);
```

eid	name	join_date	dep	salary	manager
101	David	2009-07-14	D1	50000	None
102	Sam	2010-06-24	D1	40000	101
103	Alicia	2011-05-11	D2	30000	102
104	Alex	2012-04-15	D2	20000	102
105	Robbi	2013-08-14	D2	20000	102
106	Jack	2014-09-19	D3	8000	101
107	Tom	2015-11-12	None	5000	116
108	Lily	2016-07-28	D3	1000	106

CREATE “project” TABLE

```
CREATE TABLE project (  
    person varchar(20),  
    proj_name varchar(20),  
    job_description varchar(100)  
);
```

person	proj_name	job description
David	Ecommerce	generate and manage sales via online channels
Sam	Inventory	manage location and pricing of inventory
Alicia	Inventory	manage products that are in storage or transit
Ryan	Ecommerce	advertising and marketing efforts of a company
Ellen	Inventory	manage overall operations and help employees

CREATE "sale" TABLE

```
CREATE TABLE sale (  
    category varchar(20),  
    brand varchar(20),  
    name varchar(50) NOT NULL,  
    quantity int CHECK (quantity >= 0),  
    price float NOT NULL,  
    stock boolean,  
    CONSTRAINT PK_CITY PRIMARY KEY(name)  
);
```

category	brand	name	quantity	price	stock
Phone	Apple	iPhone 13	4	1300.0	False
Phone	Apple	iPhone 12	6	1100.0	True
Phone	Samsung	Galaxy Note 20	5	1200.0	True
Phone	Samsung	Galaxy S21	4	1100.0	False
Laptop	Apple	MacBook Pro 13	3	2000.0	True
Laptop	Apple	MacBook Air	2	1200.0	True
Laptop	Dell	XPS 13	1	2000.0	False
Laptop	Dell	XPS 15	2	2300.0	True
Tablet	Apple	iPad 7th gen	3	560.0	False
Tablet	Samsung	Galaxy Tab A7	2	220.0	True

DUPLICATE TABLE

Duplicate employee Table with Data

```
CREATE table Backup AS  
SELECT *  
FROM employee;
```

Duplicate employee Table without Data

```
CREATE table Replica AS  
SELECT *  
FROM employee  
WHERE 1=2;
```

UPDATE DATA

Update manager of Tom

```
UPDATE employee  
SET manager = 106  
WHERE name = 'Tom';
```

Update department and salary of Lily

```
UPDATE employee  
SET dep = 'D3', salary = 5000  
WHERE name = 'Lily';
```

DELETE DATA

Delete Lily's record

```
DELETE FROM Backup  
WHERE name = 'Lily';
```

Delete Alex and Robbi's record

```
DELETE from Backup  
WHERE name IN ('Alex', 'Robbi');
```

Delete all records OR Truncate the whole backup data

```
DELETE FROM Backup;  
TRUNCATE TABLE Backup;
```

DROP TABLE

Drop Backup table

```
DROP TABLE Backup;
```

ALTER TABLE

Rename sale table to 'sales'

```
ALTER TABLE sale  
RENAME TO sales;
```

Rename dept column to 'dept'

```
ALTER TABLE employee  
RENAME COLUMN dep TO dept;
```

Alter dept column data type

```
ALTER TABLE employee  
ALTER COLUMN dept TYPE VARCHAR(2);
```

Add new column 'Gender'

```
ALTER TABLE employee  
ADD COLUMN Gender VARCHAR(20);
```

Add new constraint GEN to Gender column

```
ALTER TABLE employee  
ADD CONSTRAINT GEN  
CHECK (Gender IN ('M', 'F'));
```

Remove GEN constraint

```
ALTER TABLE employee  
DROP CONSTRAINT GEN;
```

Remove Gender column

```
ALTER TABLE employee  
DROP COLUMN Gender;
```

DCL COMMANDS

Grant single privilege

```
GRANT SELECT ON Users TO 'Test';
```

Grant multiple privilege

```
GRANT INSERT, UPDATE ON Users TO 'Test';
```

SQL SIMPLIFIED FOR ALL

Grant ALL privilege

GRANT ALL ON Users TO 'Test';

Revoke single privilege

REVOKE SELECT ON Users TO 'Test';

Revoke multiple privilege

REVOKE INSERT, UPDATE ON Users TO 'Test';

Revoke ALL privilege

REVOKE ALL ON Users TO 'Test';

TCL COMMANDS

BEGIN and COMMIT Transaction

BEGIN;

DELETE FROM employee

WHERE name = 'Lily';

COMMIT;

ROLLBACK to last commit

ROLLBACK;

SAVEPOINT Transaction

SAVEPOINT point;

ROLLBACK to specific saved point

ROLLBACK point;

OPERATORS

Fetch three employees who earn more than 10000

Comparison (=, >, <, <=, >=, !=), Order by and Limit

SELECT name, salary

FROM employee

WHERE salary > 10000

ORDER BY name

LIMIT 3;

name	salary
Alex	20000
Alicia	30000
David	50000

Fetch products in stock with price range 1000 to 1500

Between, Order by (Descending) AND, IN

SELECT name, brand, price stock

FROM sales

WHERE price BETWEEN 1000 AND 1500

AND stock IN ('1')

ORDER BY name DESC;

name	brand	price	stock
iPhone 12	Apple	1100.0	True
MacBook Air	Apple	1200.0	True
Galaxy Note 20	Samsung	1200.0	True

linkedin.com/in/bemnetdev



https://platform.stratascratch.com/coding/9688-churro-activity-date?code_type=1



https://platform.stratascratch.com/coding/10003-lyft-driver-wages?code_type=1



https://platform.stratascratch.com/coding/9924-find-libraries-who-havent-provided-the-email-address-in-2016-but-their-notice-preference-definition-is-set-to-email?code_type=1

Fetch employees not in department D2 and name either starts with 'j' or not end with 'y'

LIKE – STARTING, LIKE – ENDING, NOT IN, NOT LIKE, OR

SELECT name, dept

FROM employee

WHERE dept NOT IN ('D2')

AND (name LIKE ('j%') OR name NOT LIKE ('%y'));

name	dept
David	D1
Sam	D1
Jack	D3



https://platform.stratascratch.com/coding/9972-find-the-base-pay-for-police-captains?code_type=1



https://platform.stratascratch.com/coding/10026-find-all-wineries-which-produce-wines-by-possessing-aromas-of-plum-cherry-rose-or-hazelnut?code_type=1

DATE FUNCTIONS

Fetch employee data who join on April

Extract year, month, day, hour, minute, second from date

SELECT *

FROM employee

WHERE EXTRACT(MONTH FROM join_date) = '04';

eid	name	join_date	dep	salary	manager
104	Alex	2012-04-15	D2	20000	102

Fetch todays date

To_Char convert number & date to a character string.

SELECT TO_CHAR(CURRENT_DATE, 'Month dd, yyyy')

AS todays_date;

todays_date
October 10, 2022

DISTINCT

Fetch all brands in sales table

```
SELECT DISTINCT brand
FROM sales;
```

brand
Apple
Samsung
Dell



https://platform.stratascratch.com/coding/9650-find-the-top-10-ranked-songs-in-2010?code_type=1

CASE STATEMENT

Categorize employees based on their salary

```
SELECT name, salary,
CASE WHEN salary >= 30000 THEN 'High'
      WHEN salary BETWEEN 10000 AND 30000 THEN 'Mid'
      WHEN salary < 10000 THEN 'Low'
END AS Range
FROM employee
ORDER BY 2 DESC;
```

name	salary	Range
David	50000	High
Sam	40000	High
Alicia	30000	High
Alex	20000	Mid
Robbi	10000	Mid
Jack	8000	Low
Tom	5000	Low
Lily	5000	Low



https://platform.stratascratch.com/coding/9726-classify-business-type?code_type=1



https://platform.stratascratch.com/coding/9781-find-the-rate-of-processed-tickets-for-each-type?code_type=1

SET

Fetch employees from Management & involve on projects

```
SELECT name
FROM employee
WHERE dept = 'D1'
UNION
SELECT person
FROM project;
```

name
Alicia
David
Ellen
Ryan
Sam

Fetch only employees who work on projects

```
SELECT name
FROM employee
INTERSECT
SELECT person
FROM project;
```

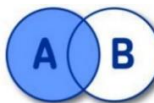
name
Alicia
David
Sam

Fetch person who is not an employee but work on project

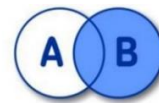
```
SELECT person
FROM project
EXCEPT
SELECT name
FROM employee;
```

person
Ellen
Ryan

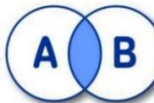
JOIN



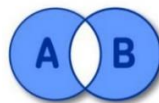
LEFT
JOIN



RIGHT
JOIN



INNER
JOIN



FULL
JOIN

INNER JOIN

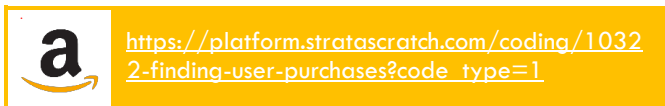
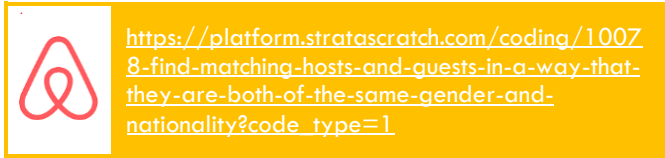
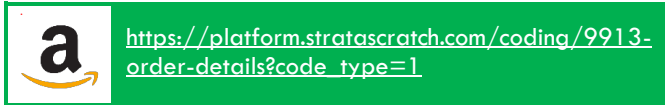
Fetch all IT employees name wrt their department

```
SELECT E.name, D.name as department
FROM employee E
INNER JOIN department D
ON E.dept = D.did
WHERE D.name = 'IT';
```

name	department
Alicia	IT
Alex	IT
Robbi	IT



https://platform.stratascratch.com/coding/1008-z-find-all-posts-which-were-reacted-to-with-a-heart?code_type=1



LEFT OUTER JOIN

Fetch all project name with respected employee name

```
SELECT E.name, P.proj_name
FROM project P
LEFT JOIN employee E
ON E.name = P.person;
```

name	proj_name
David	Ecommerce
Sam	Inventory
Alicia	Inventory
None	Ecommerce
None	Inventory



RIGHT OUTER JOIN

Fetch all employee name wrt projects they are working

```
SELECT E.name, P.proj_name
FROM project P
RIGHT JOIN employee E
ON E.name = P.person;
```

name	Proj_name
Alex	None
Alicia	Inventory
David	Ecommerce
Jack	None
Lily	None
Robbi	None
Sam	Inventory
Tom	None

FULL OUTER JOIN (LEFT JOIN UNION RIGHT JOIN)

Fetch all employee with their correlated projects

```
SELECT E.name, P.proj_name
FROM project P
FULL JOIN employee E
ON E.name = P.person;
```

name	Proj_name
None	Ecommerce
None	Inventory
Alex	None
Alicia	Inventory
David	Ecommerce
Jack	None
Lily	None
Robbi	None
Sam	Inventory
Tom	None

CROSS JOIN

Give 500 bonus for all employees

CREATE "Advance" TABLE

bonus
500

```
SELECT E.name, E.salary, A.bonus,
(E.salary + A.bonus) as Net_Salary
FROM employee E
CROSS JOIN Advance A;
```

name	salary	bonus	Net_Salary
David	50000	500	50500
Sam	40000	500	40500
Alicia	30000	500	30500
Alex	20000	500	20500
Robbi	10000	500	10500
Jack	8000	500	8500
Tom	5000	500	5500
Lily	5000	500	5500

SELF JOIN

Fetch all employee name with their manager

```
SELECT E.name, M.name as Manager
FROM employee as M
JOIN employee as E
ON M.eid = E.manager;
```

name	Manager
Sam	David
Alicia	Sam
Alex	Sam
Robbi	Sam
Jack	David
Tom	Jack
Lily	Jack

SQL SIMPLIFIED FOR ALL

[linkedin.com/in/bemnetdev](https://www.linkedin.com/in/bemnetdev)

NATURAL JOIN

SQL will Decide What Is the Join Condition by Itself.

- If there is no matching column name between two tables it will perform **CROSS JOIN**.
- If there is one common column between two tables It will perform **INNER JOIN**.
- If there is more than one common column between two tables It will perform **INNER JOIN** with **ALL** common columns.

CONCATINATE

Create a mail address for all employees using their name and department with lowercase & @tcs.in domain.

```
SELECT DISTINCT (LOWER(E.name)||'  
'||LOWER(SUBSTRING(D.name, 0, 6))||'@tcs.in')
```

AS Email, E.name

AS Emp_name, D.name

AS Department

FROM employee E

JOIN department D ON D.did = E.dept;

Email	Emp_name	Department
david.manag@tcs.in	David	Management
sam.manag@tcs.in	Sam	Management
alicia.it@tcs.in	Alicia	IT
alex.it@tcs.in	Alex	IT
robby.it@tcs.in	Robby	IT
jack.sales@tcs.in	Jack	Sales
lily.sales@tcs.in	Lily	Sales



https://platform.stratascratch.com/coding/9942-largest-olympics?code_type=1

GROUP BY AND AGGREGATION FUNCTIONS

Fetch total employee, min, max, average & total salary of each department which have less than 3 employees.

Count, Min, Max, Avg, Sum, Group by, Having

```
SELECT D.name, COUNT(1) AS emp,
```

```
MIN(E. salary) AS Min,
```

```
MAX(E. salary) AS Max,
```

```
AVG(E. salary) AS Avg,
```

```
SUM(E. salary) AS Total
```

```
FROM employee E
```

```
JOIN department D ON D.did = E.dept
```

```
GROUP BY D.name
```

```
HAVING COUNT(1) < 3;
```

Name	emp	Min	Max	Avg	Total
Management	2	40000	50000	45000	90000
Sales	2	5000	8000	6500	13000

ORDER OF EXECUTION

FROM -> JOIN -> WHERE -> GROUP BY -> HAVING

-> SELECT -> DISTINCT -> ORDER BY -> LIMIT



https://platform.stratascratch.com/coding/10128-count-the-number-of-movies-that-abigail-breslin-nominated-for-oscar?code_type=1



https://platform.stratascratch.com/coding/10299-finding-updated-records?code_type=1



https://platform.stratascratch.com/coding/10176-bikes-last-used?code_type=1



https://platform.stratascratch.com/coding/10061-popularity-of-hack?code_type=1



https://platform.stratascratch.com/coding/9992-find-artists-that-have-been-on-spotify-the-most-number-of-times?code_type=1



https://platform.stratascratch.com/coding/9622-number-of-bathrooms-and-bedrooms?code_type=1



https://platform.stratascratch.com/coding/9653-count-the-number-of-user-events-performed-by-macbookpro-users?code_type=1



https://platform.stratascratch.com/coding/10156-number-of-units-per-nationality?code_type=1



https://platform.stratascratch.com/coding/10048-top-businesses-with-most-reviews?code_type=1



https://platform.stratascratch.com/coding/9915-highest-cost-orders?code_type=1



https://platform.stratascratch.com/coding/10049-reviews-of-categories?code_type=1



https://platform.stratascratch.com/coding/9991-top-ranked-songs?code_type=1



https://platform.stratascratch.com/coding/9728-inspections-that-resulted-in-violations?code_type=1



https://platform.stratascratch.com/coding/9782-customer-revenue-in-march?code_type=1

UNNEST AND STRING_TO_ARRAY

Fetch occurrence of words in job description which is > 1

STRING_TO_ARRAY splits a string on a specified delimiter character and returns an array.

UNNEST convert array in to records.

SQL SIMPLIFIED FOR ALL

linkedin.com/in/bemnetdev

```
SELECT UNNEST(string_to_array(job_description, ' '))
AS word, COUNT(1) AS counter
FROM project
GROUP BY word
HAVING COUNT(1) > 1
ORDER BY counter DESC;
```

word	counter
manage	4
and	4
of	2

SUBQUERIES

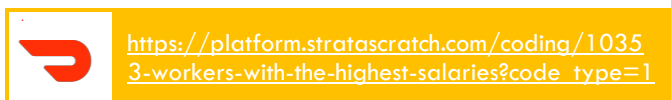
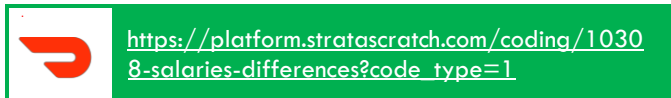
SCALAR SUB QUERY

Fetch name and salary of employee who earn more than average of total salary

The sub query always returns one row and one column

```
SELECT name, salary
FROM employee
WHERE salary > (
    SELECT AVG(salary)
    FROM employee
);
```

name	salary
David	50000
Sam	40000
Alicia	30000



MULTIPLE ROW SUB QUERY

The sub query always returns multiple row.

Find department which do not have any employees

One column multiple rows

```
SELECT *
FROM department
WHERE did NOT IN (
    SELECT DISTINCT(dept)
    FROM employee
    WHERE dept IS NOT NULL
);
```

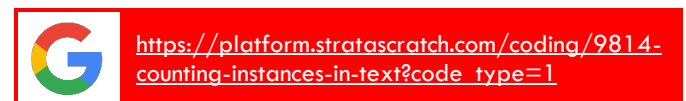
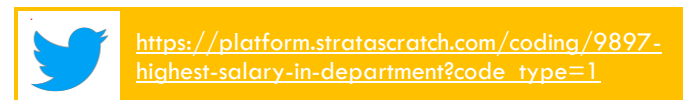
did	department
D4	HR

Employees who earn highest salary in each department

Multiple column multiple rows

```
SELECT name, salary, dept
FROM employee
WHERE (dept, salary) IN (
    SELECT dept, MAX(salary)
    FROM employee
    GROUP BY dept
);
```

name	salary	dept
David	50000	D1
Alicia	30000	D2
Jack	8000	D3



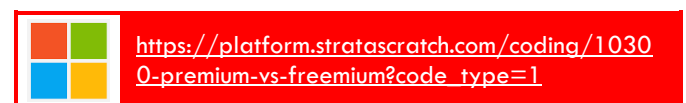
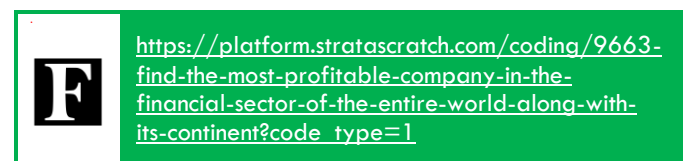
CORRELATED SUBQUERY

Find the employees in each department who earn more than the average salary in that department

sub query which is related to outer query like recursion.

```
SELECT name, dept, salary
FROM employee E1
WHERE salary > (
    SELECT AVG(salary)
    FROM employee E2
    WHERE E2.dept = E1.dept
);
```

name	dept	salary
David	D1	50000
Alicia	D2	30000
Jack	D3	8000



NESTED SUBQUERY

Find brand which sales are better than the average of total sales across all brands

The sub query has another sub query.

SQL SIMPLIFIED FOR ALL

linkedin.com/in/bemnetdev

```
SELECT *
FROM (
  SELECT brand, SUM(price) AS Total_Sales
  FROM sales
  GROUP BY brand) sales
JOIN (
  SELECT AVG(Total_Sales) AS SALES
  FROM (
    SELECT brand, SUM(price) AS Total_Sales
    FROM sales
    GROUP BY brand) X
  ) AVG_SALES
ON SALES.Total_Sales > AVG_SALES.Sales;
```

brand	Total_Sales	SALES
Apple	6160.0	4326.6666666666666

WE CAN USE DIFFERENT CLAUSES INSIDE SUB QUERIES
[WITH (CTE), SELECT, FROM, WHERE, HAVING]

Why do we write same query twice let's use "with" clause?

Step 1: Find total sales per each brand (TSPB)
Step 2: Find average sales wrt all brands (ASPAB)
Step 3: Find brand where TSPB > ASPAB

```
WITH Total_Sales (brand, TSPB) AS
(SELECT S.brand, SUM(price) AS TSPB
FROM sales S
GROUP BY S.brand),
AVG_SALES (ASPAB) AS
(SELECT AVG(TSPB) AS ASPAB
FROM Total_Sales)
```

```
SELECT *
FROM Total_Sales TS
JOIN AVG_SALES AV
ON TS.TSPB > AV.ASPAB;
```

brand	TSPB	ASPAB
Apple	6160.0	4326.6666666666666

ADVANTAGES OF USING WITH CLAUSE

- Easily readable
- Easier to maintain and debug
- Improvement of performance

USE WITH CLAUSE WHEN YOU ARE

- trying to use a particular subquery multiple times.
- writing a very complex SQL query and it becomes difficult to read and understand.
- interested in a particular record from big table data.

SQL COMMANDS WHICH ALLOW SUB QUERIES
[INSERT, UPDATE, DELETE]

https://platform.stratascratch.com/coding/10064-highest-energy-consumption?code_type=1

https://platform.stratascratch.com/coding/9905-highest-target-under-manager?code_type=1

https://platform.stratascratch.com/coding/10352-users-by-avg-session-time?code_type=1

https://platform.stratascratch.com/coding/10285-acceptance-rate-by-date?code_type=1

https://platform.stratascratch.com/coding/10284-popularity-percentage?code_type=1

https://platform.stratascratch.com/coding/9632-host-popularity-rental-prices?code_type=1

WINDOW /ANALYTIC/ FUNCTION

Whenever we are using a window function, we

1. can create partitions using by grouping then
2. apply any window function to each of those partitions.

ROW NUMBER

Give roll number to all employees with and without dept

```
SELECT E.eid, E.name, E.dept,
ROW_NUMBER() OVER() AS Roll,
ROW_NUMBER() OVER(PARTITION BY dept) AS Rp
FROM employee E;
```

eid	name	dept	Roll	Rp
101	David	D1	1	1
102	Sam	D1	2	2
103	Alicia	D2	3	1
104	Alex	D2	4	2
105	Robbi	D2	5	3
106	Jack	D3	6	1
108	Lily	D3	7	2
107	Tom	None	8	1

Fetch 1st employees from each dept to join the company

```
SELECT * FROM (
  SELECT E.eid, E.name, E.join_date, E.dept,
  ROW_NUMBER() OVER(PARTITION BY dept)
  AS RNO
  FROM employee E) X
WHERE X.RNO < 2;
```

eid	name	Join_date	dept	RNO
107	Tom	2015-11-12	None	1
101	David	2009-07-14	D1	1
103	Alicia	2011-05-11	D2	1
106	Jack	2014-09-19	D3	1

SQL SIMPLIFIED FOR ALL

[linkedin.com/in/bemnetdev](https://www.linkedin.com/in/bemnetdev)

RANK and DENSE_RANK

Rank all employees in each department based on their salary with and without duplicate rank skipping

SELECT E.eid, E.name, E.dept, E.salary,
RANK() OVER(ORDER BY salary DESC) **AS** RNK,
DENSE_RANK() OVER(ORDER BY salary DESC) **AS** Dr
FROM employee E;

eid	name	dept	salary	RNK	Dr
101	David	D1	50000	1	1
102	Sam	D1	40000	2	2
103	Alicia	D2	30000	3	3
104	Alex	D2	20000	4	4
105	Robbi	D2	20000	5	4
106	Jack	D3	8000	6	5
107	Tom	None	5000	7	6
108	Lily	D3	5000	7	6



https://platform.stratascratch.com/coding/9680-most-profitable-companies?code_type=1



https://platform.stratascratch.com/coding/10159-ranking-most-active-guests?code_type=1



https://platform.stratascratch.com/coding/10046-top-5-states-with-5-star-businesses?code_type=1



https://platform.stratascratch.com/coding/514-marketing-campaign-success-advanced?code_type=1

LAG and LEAD

Fetch a query to display if the salary of an employee is higher, lower or equal to previous and next employee

LAG extracts previous record.

LEAD extracts the next record.

WITH COMP AS (

SELECT E.eid, E.name, E.dept, E.salary,
LAG(salary) OVER(ORDER BY eid) **AS** P_SAL,
LEAD(salary) OVER(ORDER BY eid) **AS** N_SAL
FROM employee E)

SELECT *,

CASE WHEN salary > P_SAL THEN 'HIGH'
 WHEN salary < P_SAL THEN 'LOW'
 WHEN salary = P_SAL THEN 'SAME'

END P_Co,

CASE WHEN salary > N_SAL THEN 'HIGH'
 WHEN salary < N_SAL THEN 'LOW'
 WHEN salary = N_SAL THEN 'SAME'

END N_Co

FROM COMP;

name	dept	salary	P_SAL	N_SAL	P_Co	N_Co
David	D1	50000	None	40000	None	HIGH
Sam	D1	40000	50000	30000	LOW	HIGH
Alicia	D2	30000	40000	20000	LOW	HIGH
Alex	D2	20000	30000	20000	LOW	SAME
Robbi	D2	20000	20000	8000	SAME	HIGH
Jack	D3	8000	20000	5000	LOW	HIGH
Tom	None	5000	8000	5000	LOW	SAME
Lily	D3	5000	5000	None	SAME	None



https://platform.stratascratch.com/coding/10319-monthly-percentage-difference?code_type=1

FIRST_VALUE and LAST_VALUE

Query to display the most & least expensive product under each category (corresponding to each record)

FIRST_VALUE extracts first record of the partition.

LAST_VALUE extracts last record of the partition.

SELECT category, name, price,

FIRST_VALUE(name)

OVER(PARTITION BY category

ORDER BY price DESC) **AS** Expensive,

LAST_VALUE(name)

OVER(PARTITION BY category

ORDER BY price DESC) **AS** Cheap

FROM sales;

category	name	price	Expensive	Cheap
Laptop	XPS 15	2300	XPS 15	XPS 15
Laptop	MacBook Pro	2000	XPS 15	XPS 13
Laptop	XPS 13	2000	XPS 15	XPS 13
Laptop	MacBook Air	1200	XPS 15	MacBook Air
Phone	iPhone 13	1300	iPhone 13	iPhone 13
Phone	Galaxy Note	1200	iPhone 13	Galaxy Note
Phone	iPhone 12	1100	iPhone 13	iPhone 12
Phone	Galaxy S21	1100	iPhone 13	Galaxy S21
Tablet	ipad 7th gen	560	ipad 7th gen	ipad 7th gen
Tablet	Galaxy Tab A7	220	ipad 7th gen	Galaxy Tab A7

But, the above result is not correct for cheap column. That is b/c of default frame clause SQL is using. What is frame?

FRAME CLAUSE

Fetch least expensive product of each category

Inside each partitions we can again create some subset of records which is called as frames. So basically a frame is a subset of a partition.

NB: Default FRAME CLAUSE is RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW. This means our frame is all records between first record and current record.

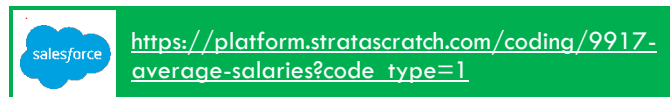
Not every window function will really be impacted by this default FRAME CLAUSE, It generally impacts the last_value, nth_value and almost all aggregate functions.

SQL SIMPLIFIED FOR ALL

[linkedin.com/in/bemnetdev](https://www.linkedin.com/in/bemnetdev)

```
SELECT category, brand, name, price,
FIRST_VALUE(name) OVER(PARTITION BY category
SELECT category, brand, name, price
FROM (
    SELECT *,
    LAST_VALUE(name)
    OVER(PARTITION BY category
    ORDER BY price DESC
    RANGE BETWEEN UNBOUNDED PRECEDING
    AND UNBOUNDED FOLLOWING) AS cheap
FROM sales) X
WHERE NAME = cheap;
```

category	brand	name	Price
Laptop	Apple	MacBook Air	1200
Phone	Samsung	Galaxy S21	1100
Tablet	Samsung	Galaxy Tab A7	220



DIFFERENCE BETWEEN ROWS AND RANGE

We can use the interchangeably unless that particular row has some other rows with duplicated values.

rows between unbounded preceding and current row

- For ROWS, LAST_VALUE is the exact current row

range between unbounded preceding and current row

- For RANGE, LAST_VALUE is the last row

category	name	Price	last_by_range	last_by_rows
Phone	iPhone 13	1300	iPhone 13	iPhone 13
Phone	Galaxy Note 20	1200	Galaxy Note 20	Galaxy Note 20
Phone	iPhone 12	1100	Galaxy S21	iPhone 12
Phone	Galaxy S21	1100	Galaxy S21	Galaxy S21

NTH_VALUE

Fetch the 2nd most expensive product of each category

It access Nth record of the partition

```
SELECT category, brand, name, price
FROM (
    SELECT *,
    NTH_VALUE(name, 2)
    OVER(PARTITION BY category
    ORDER BY price DESC
    RANGE BETWEEN UNBOUNDED PRECEDING
    AND UNBOUNDED FOLLOWING) AS cheap
FROM sales) X
WHERE NAME = cheap;
```

category	brand	name	Price
Laptop	Apple	MacBook Pro 13	2000
Phone	Samsung	Galaxy Note 20	1200
Tablet	Samsung	Galaxy Tab A7	220

NTILE

Segregate all phones into expensive, midrange & cheap

It segregate records of the partition into N no. of buckets

```
SELECT brand, name,
CASE WHEN X.BUCKETS = 1 THEN 'EXPENSIVE'
      WHEN X.BUCKETS = 2 THEN 'MIDRANGE'
      WHEN X.BUCKETS = 3 THEN 'CHEAP'
END PRICE_RANGE
FROM (
    SELECT *,
    NTILE(3)
    OVER(ORDER BY price DESC) AS BUCKETS
FROM sales
WHERE category = 'Phone') X;
```

brand	name	PRICE_RANGE
Apple	iPhone 13	EXPENSIVE
Samsung	Galaxy Note 20	EXPENSIVE
Apple	iPhone 12	MIDRANGE
Samsung	Galaxy S21	CHEAP

CUME_DIST / CUMMULATIVE DISTRIBUTION /

Fetch all products which are constituting the first 30% of the data in products table based on price

It identify the distribution percentage of each record wrt all the rows. Value of CUM_DIST is in range b/n 0 & 1.

```
SELECT brand, category, name, (cd||'%') AS cd
FROM (
    SELECT *, ROUND(
    CUME_DIST()
    OVER(ORDER BY price DESC)::numeric*100,2) AS cd
FROM sales) X
WHERE cd <= 30;
```

category	brand	name	cd
Laptop	Dell	XPS 15	10%
Laptop	Apple	MacBook Pro 13	30%
Laptop	Dell	XPS 13	30%

PERCENT_RANK - It is like percentile.

Write a Query to identify how much percentage more expensive is "iPhone 13" when compared to all products

```
SELECT category, brand, name, Perc
FROM (
    SELECT *,
    PERCENT_RANK()
    OVER(ORDER BY price) AS PER_RANK,
    ROUND(PERCENT_RANK()
    OVER(ORDER BY price)::numeric*100,2) AS Perc
FROM sales) X
WHERE name = 'iPhone 13';
```

category	brand	name	perc
Phone	Apple	iPhone 13	66.67



FREQUENTLY ASKED SQL INTERVIEW QUESTIONS

<https://www.edureka.co/blog/interview-questions/sql-interview-questions>