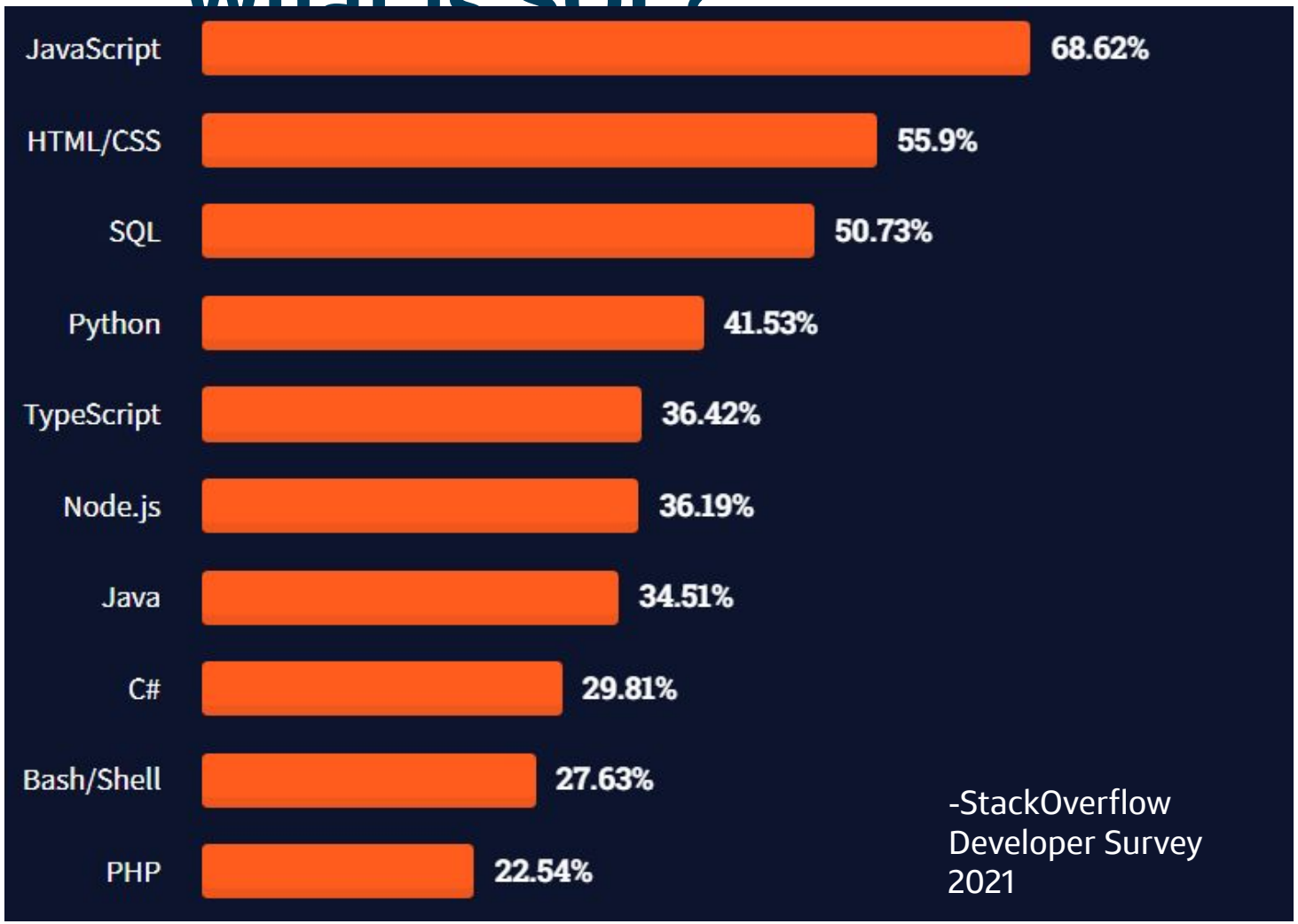
SQL 101

April 2022

What is SQL?



-StackOverflow Developer Survey 2021

SQL stands for Structured Query Language. SQL is used to communicate with a database.

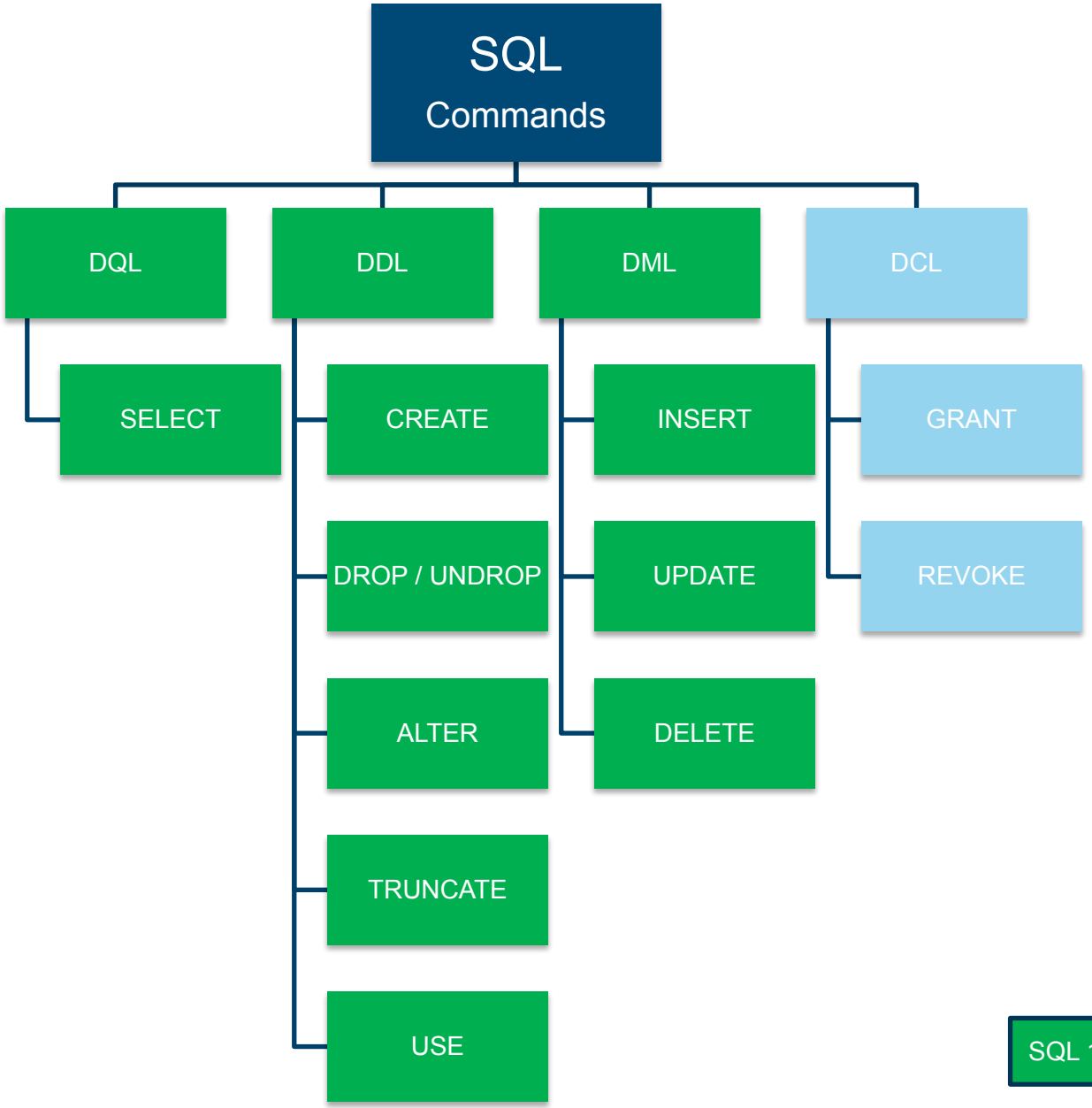
According to Stack Overflow's Developer 2021 survey, SQL is the third most commonly used programming language among Professional Developers.

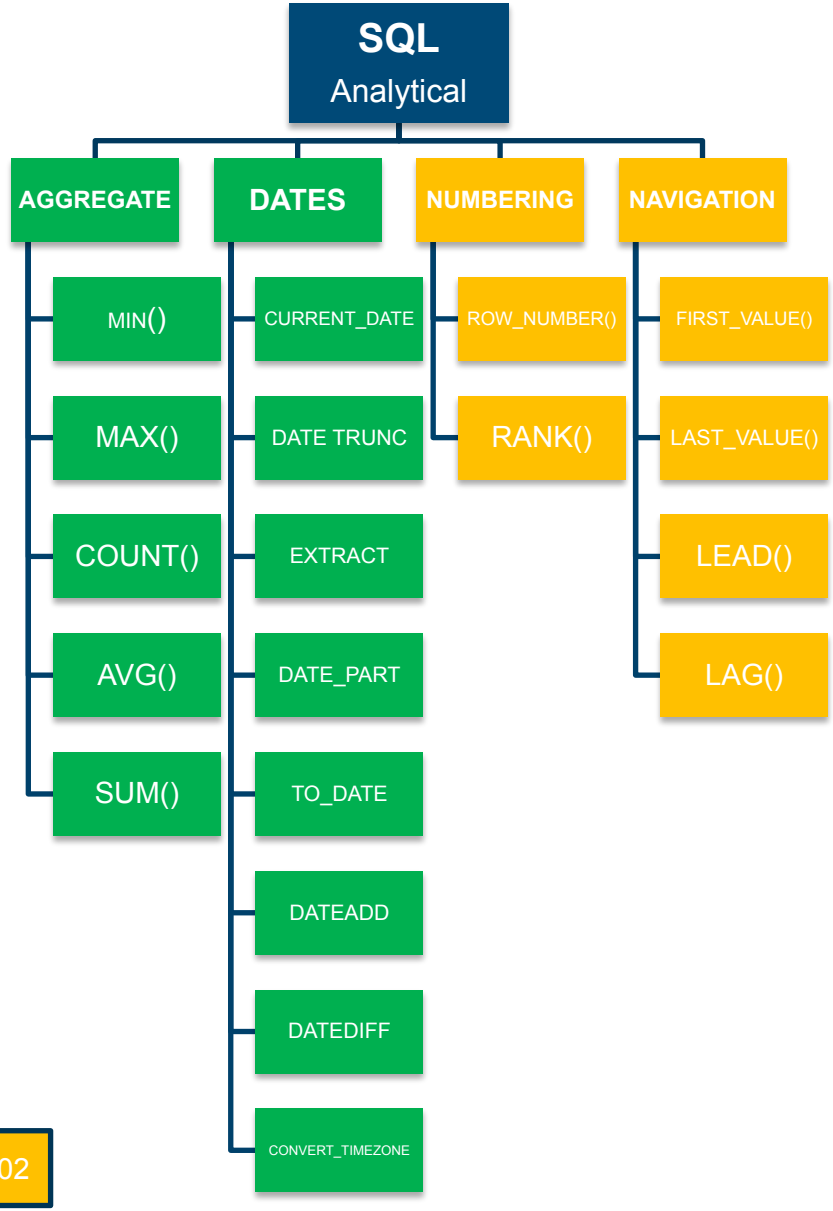
What is Snowflake?

Snowflake is a columnar MPP data warehousing solution built for the cloud with the same ANSI SQL that typical users are already familiar with. One of Snowflake's main advantages is the separation of storage and compute allowing for numerous concurrent users and processes to be scaled out indefinitely and on demand. Commonly referred to as Data-Warehouse-as-a-Service.

Fun Fact: Capital One was an early investor in Snowflake, which led to a $535 Million investment gain when Snowflake had its Initial Public Offering in September 2020. - Capital One 2020 Annual Report

Structured Query Language (SQL) Categories





SQL 101 SQL 102

0. Example Tables

Account

ACCT_ID	ACCT_OPEN_DT	ACCT_STATUS	CARD_TYPE
10000605865	2006-11-05	CLOSED	VISA
10001806270	2007-01-20	OPEN	DINERS
10075513038	2017-08-30	CHARGED OFF	VISA
10075513049	2017-08-30	OPEN	MASTER
10075513052	2017-08-30	OPEN	EXPRESS
10075513057	2017-08-30	OPEN	UNION

0. Example Tables (Contd.)

Customer

ACCT_ID	CUSTOMER_ NAME	DATA_OF_ BIRTH	HOME_PHN_ NUM	WORK_PHN_ NUM	MOBILE_PHN_ NUM	ADR_LINE_1	ADR_LINE_2	CITY	STATE	ZIP
10000605865	CHOSENICA	11/15/45				87324 PUNARUKU ESTUARY HIGHWAY	APT#1	PERRYOPOLIS	PA	15473300355
10001806270	HELVETICUS	7/10/43	7918588802			30398 BISMARK TERRACE RD	APT#4	BROOKLYN	NY	11229351207
10075513038	AMBLYRHYNCHOS	10/19/88	5407174767	9483165243	305139605	95820 TOLL BAR CREEK BLVD	APT#5	NEW YORK	NY	10040192341
10075513049	GODEFFROYI	8/10/63	7003859231			67491 MARCHWIEL DRIVE	APT#6	GROTON	MA	1450187081
10075513052	DIARDI	6/17/85	9166369005	8097866927	846760644	27296 AWAKERE STREET	APT#7	MIAMI	FL	33144557435
10075513057	COELURUS	1/2/62	7061141870	7061141870	7061141870	88094 CLARK STREAM WAY	APT#8	SARATOGA	CA	95070342638

0. Example Tables (Contd.)

Transactions

ACCT_ID	TRXN_POST_DT	TRXN_AMT	TRXN_DT	TRXN_DESC	MRCH_NM	MRCH_ST_CD	MRCH_CNTRY_CD
10001806270	2019-08-27	145.23	2019-08-23		GOLDIES DELI		ZA
10001806270	2020-02-12	844.18	2020-02-10		APPLE STORE #R087	FL	US
10001806270	2020-02-14	1085.49	2020-02-14		ELECTRONIC PAYMENT		
10075513049	2019-08-02	87.23	2019-07-31		FISH BONES IN THE VILLAGE	MA	US
10075513049	2020-01-06	75	2020-01-04		JETBLUE 2790613510395	UT	US
10075513049	2020-01-31	129	2020-01-30		KTP.COM	ME	US

Data Definition Language (DDL)

DDL: File Formats

- Allows users to describe the files to be imported into a stage
- Create Example:

```
CREATE OR REPLACE FILE FORMAT MY_CSV_FORMAT
TYPE = 'CSV'
FIELD_DELIMITER = ','
FIELD_OPTIONALLY_ENCLOSED_BY = '""'
SKIP_HEADER = 1;
```

- Drop Example:

```
DROP FILE FORMAT CSV_FORMAT;
```

DDL: CREATE TABLE

- Allows users to create a new table
- Examples:

```
CREATE TABLE DEMO(
ID INT,
TIMESTAMP DATETIME,
DESCRIPTION VARCHAR(64));

CREATE TABLE DEMO2 AS
SELECT * FROM DEMO;

CREATE TABLE DEMO3 CLONE
SELECT * FROM DEMO
AT (TIMESTAMP => TO_TIMESTAMP_TZ('04/15/2019', 'MM/DD/YYYY));
```

DDL: ALTER TABLE

- Allows users to modify existing tables
- Examples:

```
ALTER TABLE DEMO ADD TEST_FLAG VARCHAR(1);

ALTER TABLE DEMO DROP TEST_FLAG;

ALTER TABLE DEMO RENAME TO DEMO_RENAMED;

ALTER TABLE DEMO SWAP WITH DEMO2;
```

DDL: DROP/UNDROP/TRUNCATE TABLE

- Allows users to drop, restore, or empty existing tables
- Examples:

```
DROP TABLE DEMO;

UNDROP TABLE DEMO;

TRUNCATE TABLE DEMO;
```

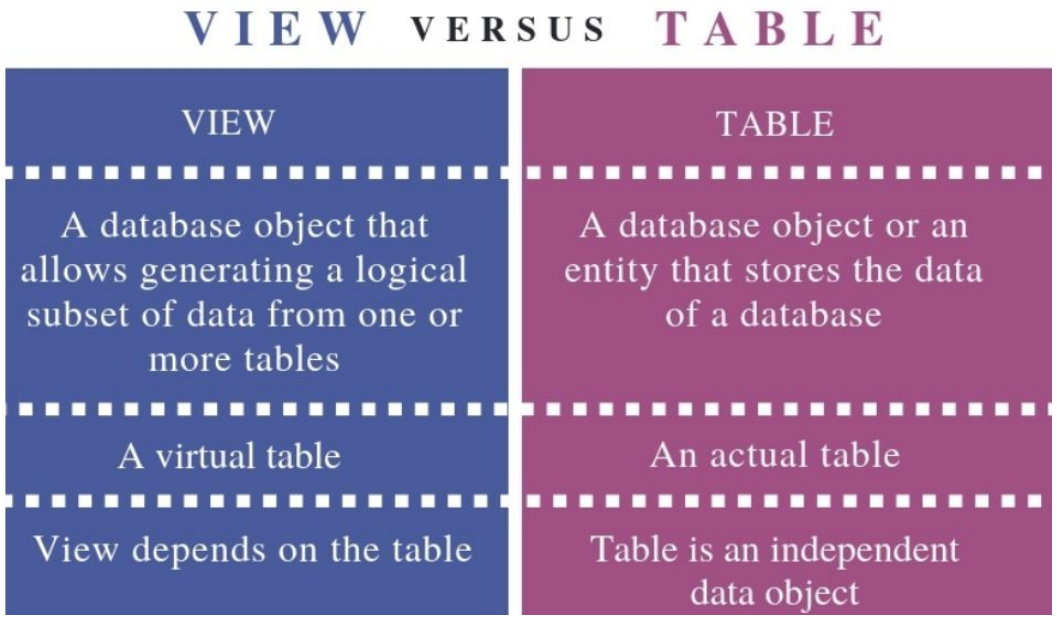
DDL: VIEWS

- Allows users to manage views
- Examples:

  Create: CREATE VIEW DEMO_VIEW AS SELECT acct_id, customer_name FROM DEMO;

  Rename: ALTER VIEW DEMO_VIEW RENAME TO DEMO_VIEW2;

  Drop: DROP VIEW DEMO_VIEW2;



Data Query Language (DQL)

SELECT - Allows users to select data from a specified table

- Examples:

```
SELECT * FROM ACCOUNT;

SELECT ACCT_ID, ACCT_STATUS
FROM ACCOUNT LIMIT 6;
```

ACCT_ID	ACCT_STATUS
10000605865	CLOSED
10001806270	OPEN
10075513038	CHARGED OFF
10075513049	OPEN
10075513052	OPEN
10075513057	OPEN

- Use DISTINCT to return unique values
- Example:

```
SELECT DISTINCT CARD_TYPE
FROM ACCOUNT ;
```

CARD_TYPE
DINERS DISCOVER EXPRESS JBC MASTER UNION VISA

SELECT (Cont.)

- Examples:

```
-- selects the first three records from the ACCOUNT table --
SELECT TOP 3 * FROM ACCOUNT;

SELECT * FROM ACCOUNT FETCH FIRST 3 ROWS ONLY;

-- selects 10 records after the first three records from the ACCOUNT table --

SELECT * FROM ACCOUNT LIMIT 3 OFFSET 10;
```

WHERE - Allows users to filter records that fulfill a specified condition

- Examples:

```
SELECT * FROM ACCOUNT WHERE ACCT_STATUS = 'CHARGED OFF';

SELECT ACCT_ID, ACCT_STATUS FROM ACCOUNT WHERE ACCT_ID = '10000605865';
```

ACCT_ID	ACCT_OPEN_DT	ACCT_STATUS	CARD_TYPE
10075513038	2017-08-30	CHARGED OFF	VISA
10075513125	2017-08-30	CHARGED OFF	JBC
10075513182	2017-08-30	CHARGED OFF	JBC
10075513213	2017-08-30	CHARGED OFF	EXPRESS
10075513250	2017-08-30	CHARGED OFF	DINERS
10075513294	2017-08-30	CHARGED OFF	MASTER

WHERE (cont.)

- Examples:

```
SELECT * FROM ACCOUNT WHERE ACCT_OPEN_DT > '2011-05-05';

SELECT ACCT_ID, ACCT_STATUS FROM ACCOUNT
WHERE ACCT_OPEN_DT >= '2011-05-01' AND ACCT_OPEN_DT < '2011-06-01';

SELECT * FROM CUSTOMER
WHERE STATE = 'VA' AND CITY = 'VIRGINIA BEACH';

SELECT * FROM CUSTOMER
WHERE HOME_PHN_NUM IS NULL AND MOBILE_PHN_NUM IS NULL AND
WORK_PHN_NUM IS NULL ;
```

ORDER BY - used to sort the result-set in ascending or descending order.

- Examples:

```
SELECT ACCT_ID, TRXN_AMT
FROM   TRANSACTIONS
WHERE ACCT_ID = '10001806270'
ORDER BY TRXN_AMT;

SELECT ACCT_ID, TRXN_AMT
FROM   TRANSACTIONS
WHERE ACCT_ID = '10001806270'
ORDER BY TRXN_AMT DESC;

SELECT ACCT_ID, TRXN_AMT
FROM   TRANSACTIONS
ORDER BY ACCT_ID ASC,
      TRXN_AMT DESC;
```

By default, the order will be ASCending. Use the DESC keyword for DESCending order.

ACCT_ID	TRXN_AMT
10001806270	0.37
10001806270	6.8
10001806270	7.99
10001806270	32.59
10001806270	32.72
10001806270	32.72
10001806270	32.78
10001806270	33.06
10001806270	33.06
10001806270	33.31
10001806270	33.34
10001806270	33.64
10001806270	35.12
10001806270	35.18
10001806270	35.73
10001806270	40.84
10001806270	54.76
10001806270	59.13
10001806270	60.36
10001806270	65.93
10001806270	81.14
10001806270	81.77
10001806270	98.56
10001806270	103.98
10001806270	128.45
10001806270	145.23
10001806270	208
10001806270	255.96
10001806270	844.18
10001806270	1029.18
10001806270	1085.49

Nulls - is different from a zero value or a field that contains spaces

Examples:

- SELECT * FROM CUSTOMER WHERE HOME_PHN_NUM IS NULL
- SELECT * FROM CUSTOMER WHERE HOME_PHN_NUM IS NOT NULL
- SELECT EQUAL_NULL(HOME_PHN_NUM,NULL) FROM CUSTOMER – gives TRUE/FALSE

NVL replaces null values with a value of your choice

Example:

```
SELECT NVL(HOME_PHN_NUM, ‘0000000000') FROM CUSTOMER;
```

Ranges and Sets

- IN, NOT IN

Examples:

```
WHERE CITY IN ('MIAMI', 'LAKELAND');
WHERE STATE NOT IN ('FL', 'CA', 'VA', 'DC');
WHERE TRXN_POST_DT = '2019-08-24'
```

- BETWEEN, NOT BETWEEN

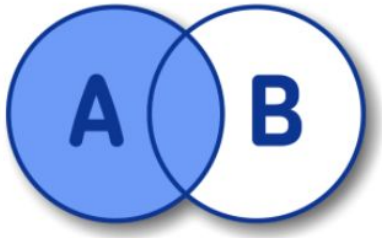
Examples:

```
WHERE TRXN_POST_DT BETWEEN '2020-09-01' AND '2020-09-30'
```

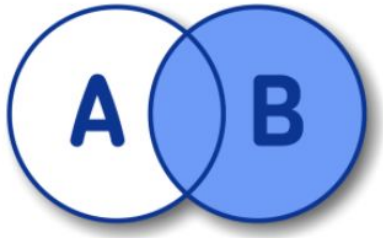
Both dates are inclusive values

SQL Joins

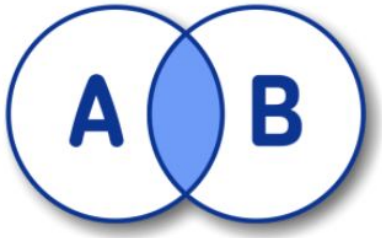
SQL JOINS



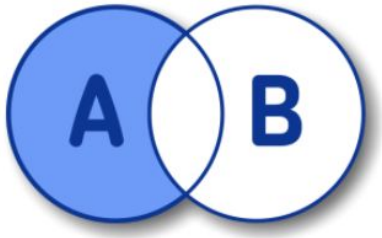
SELECT * FROM
A LEFT JOIN B
ON A.KEY = B.KEY



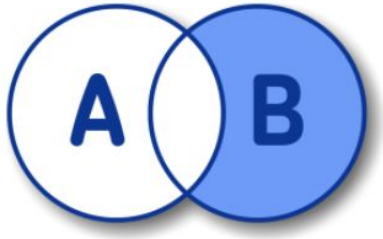
SELECT * FROM
A RIGHT JOIN B
ON A.KEY = B.KEY



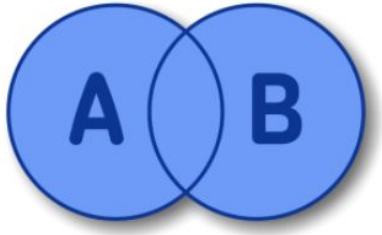
SELECT * FROM
A INNER JOIN B
ON A.KEY = B.KEY



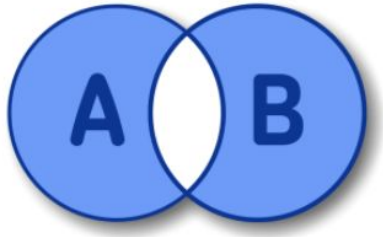
SELECT * FROM A
LEFT JOIN B
ON A.KEY = B.KEY
WHERE B.KEY IS NULL



SELECT * FROM A
RIGHT JOIN B
ON A.KEY = B.KEY
WHERE A.KEY IS NULL



SELECT * FROM A
FULL OUTER JOIN B
ON A.KEY = B.KEY



SELECT * FROM A FULL
OUTER JOIN B ON A.KEY =
B.KEY WHERE A.KEY IS
NULL OR B.KEY IS NULL

Inner Join - An inner join outputs selected columns of the records that exist in all the tables of the join statement based on join criteria.

ACCOUNT

ACCT_ID	ACCT_OPEN_DT	ACCT_STATUS	CARD_TYPE
10000605865	2006-11-05	CLOSED	VISA
10001806270	2007-01-20	OPEN	JBC
10075513038	2017-08-30	CHARGED OFF	MASTER

CUSTOMER

ACCT_ID	CUSTOMER_NAME	DATA_OF_BIRTH	HOME_PHN_NUM	WORK_PHN_NUM	MOBILE_PHN_NUM	ADR_LINE_1	ADR_LINE_2	CITY	STATE	ZIP
10000605865	CHOSENICA	11/15/45				87324 PUNARUKU ESTUARY HIGHWAY	APT#1	PERRYOPOLIS	PA	15473300355
10001806270	HELVETICUS	7/10/43	7918588802			30398 BISMARK TERRACE RD	APT#4	BROOKLYN	NY	11229351207
10075513049	GODEFFROYI	8/10/63	7003859231			67491 MARCHWIEL DRIVE	APT#6	GROTON	MA	1450187081

Inner Join (cont.)

- Example:

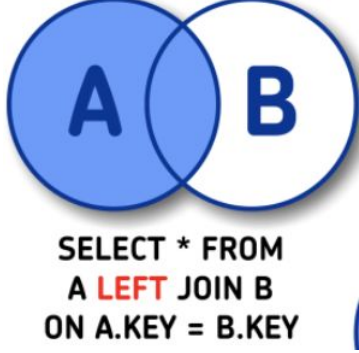
```
SELECT ac.ACCT_ID, ac.ACCT_STATUS, cm.CUSTOMER_NAME, cm.DATE_OF_BIRTH
FROM ACCOUNT ac
INNER JOIN CUSTOMER cm ON ac.ACCT_ID = cm. ACCT_ID;
```

- Output:

ACCT_ID	ACCT_STATUS	CUSTOMER_NAME	DATE_OF_BIRTH
10000605865	CLOSED	CHOSENICA	1945-07-10
10001806270	OPEN	HELVETICUS	1943-07-10

Note: The third record dropped out.

Left Join - A left join returns all records from the left table (table1), and the matching records from the right table (table2).



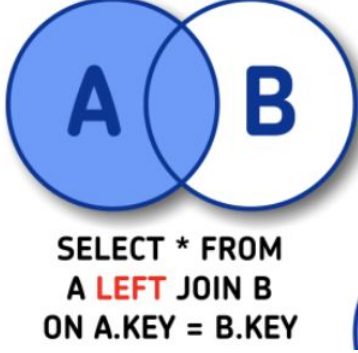
ACCOUNT

ACCT_ID	ACCT_OPEN_DT	ACCT_STATUS	CARD_TYPE
10000605865	2006-11-05	CLOSED	VISA
10001806270	2007-01-20	OPEN	JBC
10075513038	2017-08-30	CHARGED OFF	MASTER

CUSTOMER

ACCT_ID	CUSTOMER_NAME	DATA_OF_BIRTH	HOME_PHN_NUM	WORK_PHN_NUM	MOBILE_PHN_NUM	ADR_LINE_1	ADR_LINE_2	CITY	STATE	ZIP
10000605865	CHOSENICA	11/15/45				87324 PUNARUKU ESTUARY HIGHWAY	APT#1	PERRYOPOLIS	PA	15473300355
10001806270	HELVETICUS	7/10/43	7918588802			30398 BISMARK TERRACE RD	APT#4	BROOKLYN	NY	11229351207
10075513049	GODEFFROYI	8/10/63	7003859231			67491 MARCHWIEL DRIVE	APT#6	GROTON	MA	1450187081

Left Join (cont.)



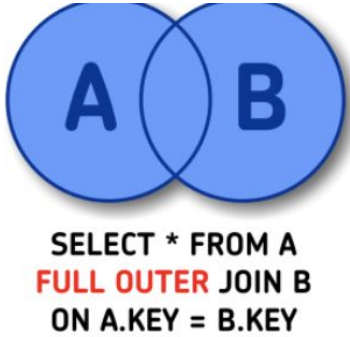
- Example:

```
SELECT ac.ACCT_ID, ac.ACCT_STATUS, cm.CUSTOMER_NAME, cm. DATE_OF_BIRTH
FROM ACCOUNT ac
LEFT JOIN CUSTOMER cm ON ac.ACCT_ID = cm. ACCT_ID;
```

- Output:

ACCT_ID	ACCT_STATUS	CUSTOMER_NAME	DATE_OF_BIRTH
10000605865	CLOSED	CHOSENICA	1945-07-10
10001806270	OPEN	HELVETICUS	1943-07-10
10075513038	CHARGED OFF		

Full (OUTER) Join - returns all records when there is a match in left (table1) or right (table2) table records



- Example:

```
SELECT ac.ACCT_ID, ac.ACCT_STATUS, cm.CUSTOMER_NAME, cm. DATE_OF_BIRTH
FROM ACCOUNT ac
FULL JOIN CUSTOMER cm ON ac.ACCT_ID = cm. ACCT_ID;
```

- Output:

ACCT_ID	ACCT_STATUS	CUSTOMER_NAME	DATE_OF_BIRTH
10000605865	CLOSED	CHOSENICA	1945-07-10
10001806270	OPEN	HELVETICUS	1943-07-10
10075513038	CHARGED OFF		
		GODEFFROYI	1963-08-10

SQL Analytics

Functions

- SUM adds the values for a group of rows

Example:

```
SELECT SUM(TRXN_AMT)
FROM TRANSACTIONS WHERE ACCT_ID = '10075513049';
```

- AVG finds the mean value for a group of rows

Example:

```
SELECT AVG(TRXN_AMT)
FROM TRANSACTIONS WHERE ACCT_ID = '10075513049';
```

NOTE: Records having Null on the aggregated field get ignored.

Functions (cont.)

- MIN returns the minimum value for a group of rows

Example:

```
SELECT MIN(TRXN_AMT)
FROM TRANSACTIONS WHERE
ACCT_ID = '10075513049';
```

- MAX returns the maximum of all values for a group of rows

Example:

```
SELECT MAX(TRXN_AMT)
FROM TRANSACTIONS WHERE
ACCT_ID = '10075513049';
```

- COUNT provides a count of the number of rows in a group

Example:

```
SELECT COUNT(*)
FROM    ACCOUNT;
```

- SUBSTR returns the substring of a string, starting at the index of the second parameter with a length of the third parameter.

Example:

```
SELECT SUBSTR(ZIP, 1, 5)
FROM CUSTOMER;
```

Functions (cont.)

- CONCAT joins two text fields

Examples:

```
SELECT CONCAT(‘First ', ‘Name’)
      OR
SELECT ‘First ’ || ‘Name’
```

Result: ‘First Name'

- IFF is the SQL version of an if/else statement

Example:

```
SELECT IFF(3<20, 'Math works', 'We have a problem')
```

Result: 'Math works'

GROUP BY - groups rows that have the same values into summary rows

- Example:

```
SELECT ACCT_STATUS, COUNT(*)
FROM  ACCOUNT
GROUP BY ACCT_STATUS
ORDER BY ACCT_STATUS;
```

ACCT_STATUS	COUNT(*)
CHARGED OFF	24
CLOSED	599
OPEN	499

- Example with alias:

```
SELECT ACCT_STATUS, COUNT(*) AS NO_OF_ACCOUNTS
FROM  ACCOUNT
GROUP BY ACCT_STATUS
ORDER BY ACCT_STATUS;
```

ACCT_STATUS	NO_OF_ACCOUNTS
CHARGED OFF	24
CLOSED	599
OPEN	499

GROUP BY (cont.)

- Example:

```
SELECT STATE, COUNT(*)
FROM   CUSTOMER
GROUP BY STATE
HAVING COUNT(*) >  50;
```

STATE	COUNT(*)
CA	126
FL	81
NY	95
TX	76

Pattern Matching

- The character “%” matches any string of zero or more characters except Null
- Example:

```
SELECT *
FROM CUSTOMER
WHERE CUSTOMER_NAME LIKE ‘C%';
```

- The character “_” matches any single character
- Example:

```
SELECT *
FROM TRANSACTIONS
WHERE MRCH_NM LIKE ‘Net_lix.com’;
```

Note: In Snowflake, LIKE is a case sensitive command. Use ILIKE if you want to ignore case.

Alternatively, you can use UPPER or LOWER to change the case of the field being compared.

Date Functions

- CURRENT_DATE() returns today's date

  Result: ‘2022-03-28’

- DATE_TRUNC returns the first date in that unit

  DATE_TRUNC('YEAR', CURRENT_DATE())
  Result: ‘2022-01-01’

  Also available: 'DAY', 'WEEK', 'MONTH', 'QUARTER'

- EXTRACT pulls the part of the date based on the unit

  EXTRACT('YEAR', CURRENT_DATE())
  Result: 2022

  Also available: 'DAY', 'WEEK', 'MONTH', 'QUARTER'

- DATE_PART extracts the specified date or time part from a date, time, or timestamp

  DATE_PART('YEAR', CURRENT_DATE())
  Result: 2022

  Also available: 'DAY', 'WEEK', 'MONTH', 'QUARTER'

- TO_DATE converts a string to a datetime:

  SELECT TO_DATE(‘4/26/2019', ‘MM/DD/YYYY')
  Results: ‘2019-04-26’

Date Functions (cont.)

- DATEADD adds specified amount of time to a date (you can also use a negative value)

Example:

```
SELECT DATEADD('MONTH', 6, CURRENT_DATE())
```

Results: 2022-09-27

Also available: 'DAY', 'WEEK', 'QUARTER', 'YEAR'

- DATEDIFF returns the difference between two dates in a specified format

Example:

```
SELECT DATEDIFF('QUARTER', '1970-01-01', '1985-01-01')
```

Results: 60

- CONVERT_TIMEZONE - converts a timestamp to another time zone

```
SELECT current_timestamp() as now_in_NY,
        convert_timezone('America/Los_Angeles', current_timestamp()) as now_in_LA,
convert_timezone('Europe/Paris', current_timestamp()) as now_in_paris,
convert_timezone('Asia/Tokyo', current_timestamp()) as now_in_tokyo;
```

Data Manipulation Language (DML)

Inserting and Deleting

- Example of how to insert hard-coded data values:

```
INSERT INTO BRAND (BRAND_NAME, COUNTRY)
VALUES ('Apple', 'USA');
```

- Example of how to insert data values from other tables:

```
INSERT INTO BRAND (BRAND_NAME, COUNTRY)
SELECT BRAND_NAME, COUNTRY
FROM   BRAND_BACKUP
WHERE BRAND_NAME  LIKE  'L%' ;
```

Inserting and Deleting (cont.)

- You have two options to delete data:

```
DELETE FROM table_name WHERE ...;
      OR
DELETE FROM table_name;
```

- To truncate a table, use this syntax:

```
TRUNCATE TABLE table_name;
```

Truncate is actually DDL because it frees up blocks on the filesystem. This means that it will automatically commit.

Update

- Update allows you to change the values for given columns in a table
- Example:

```
UPDATE PRODUCT
SET    BRAND_NAME = 'DeBeers'
WHERE  BRAND_NAME IS NULL
```

- Update will change the value for the column given for all rows matching the stated criteria.

- **Note:** Updating a table will usually lock the table. If a table is too big, it would impact online transaction.

Appendix

SQL Optimization

SQL queries don't run in the order in which they're written (or read). In fact, the SELECT statement is one of the last steps that's completed when a query is run.

The first part of the query that's run is the FROM/JOIN clause, meaning if we want to improve query performance (and who doesn't), we should look first at our old friend the JOIN.