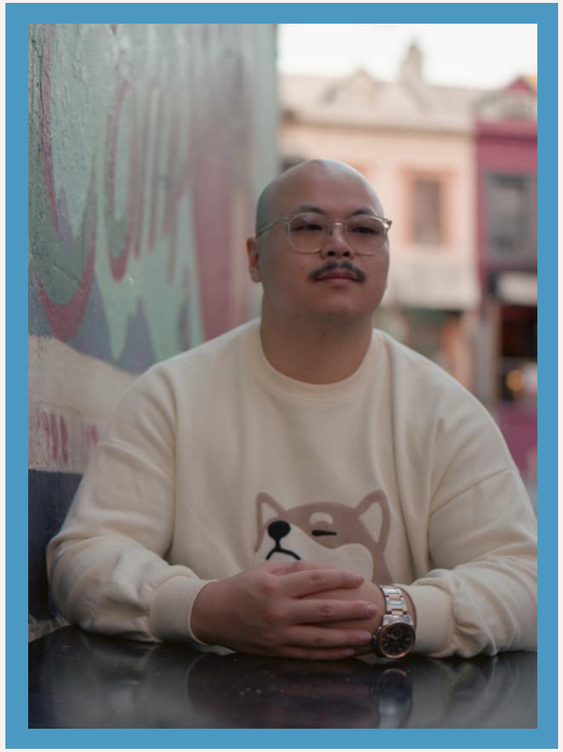
PostgreSQL for Data Analytics

Danny Ma
October 2021

Pearson

About me



Pearson

My Data Analytics Journey

Pearson

My Data Analytics Journey

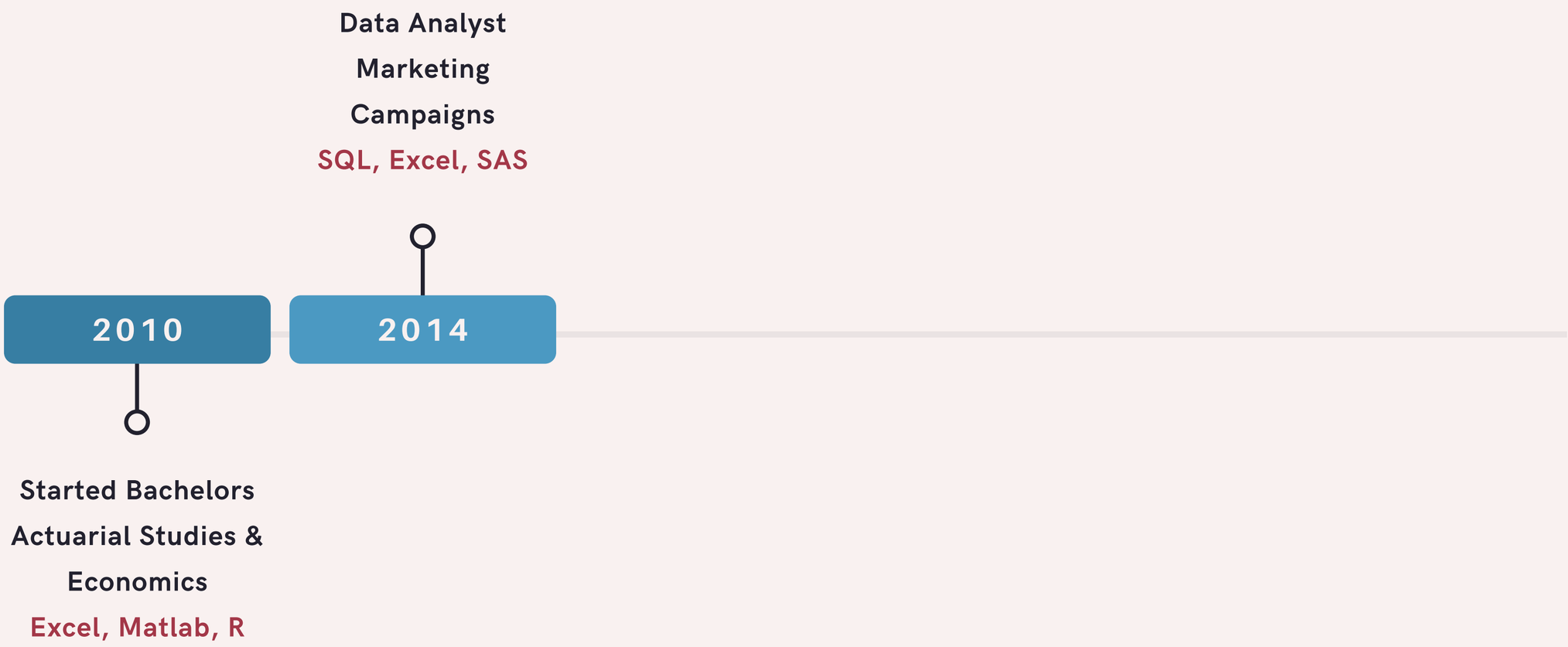
2010

Started Bachelors Actuarial Studies & Economics

Excel, Matlab, R

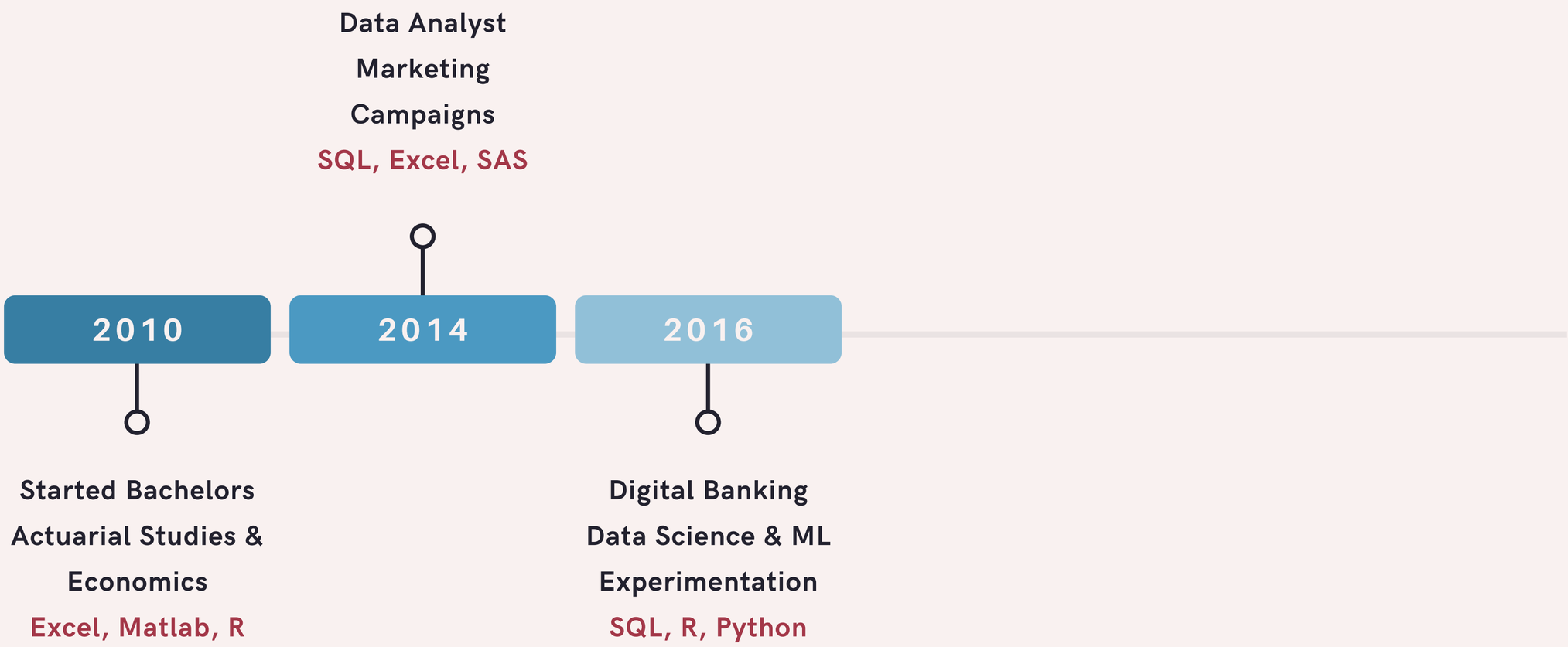
Pearson

My Data Analytics Journey



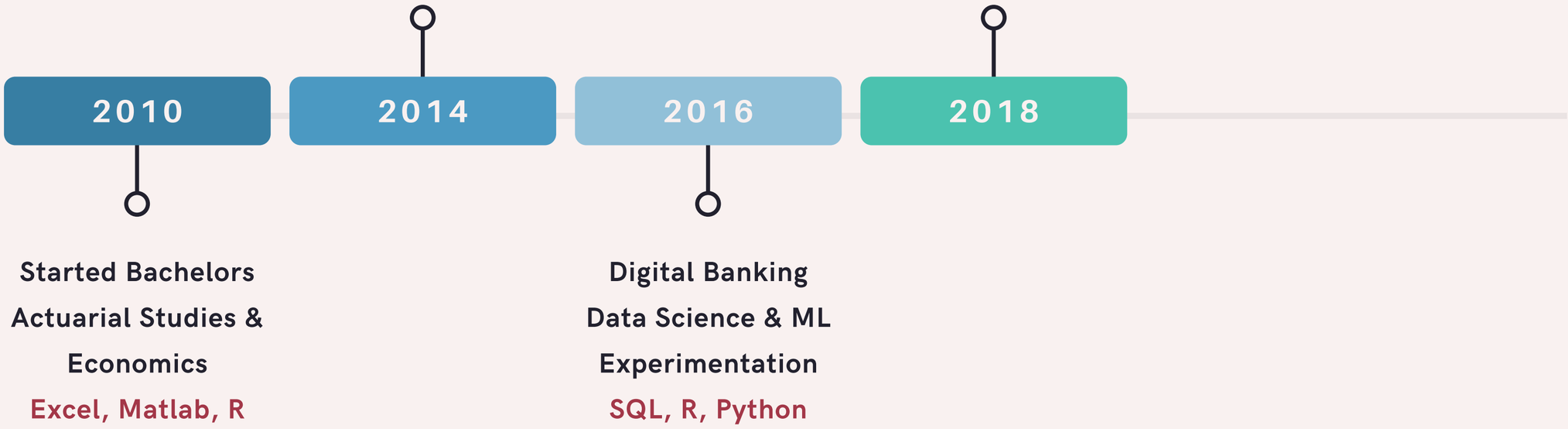
Pearson

My Data Analytics Journey



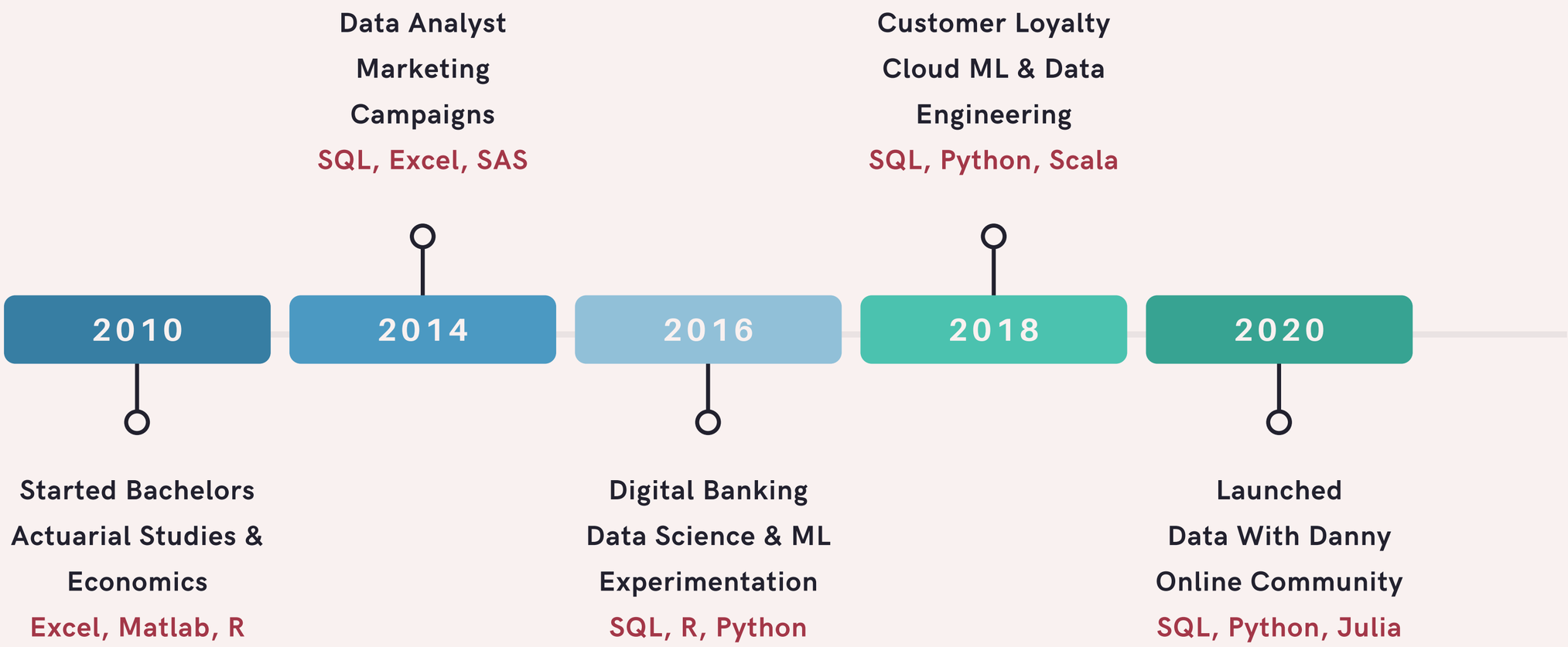
Pearson

My Data Analytics Journey



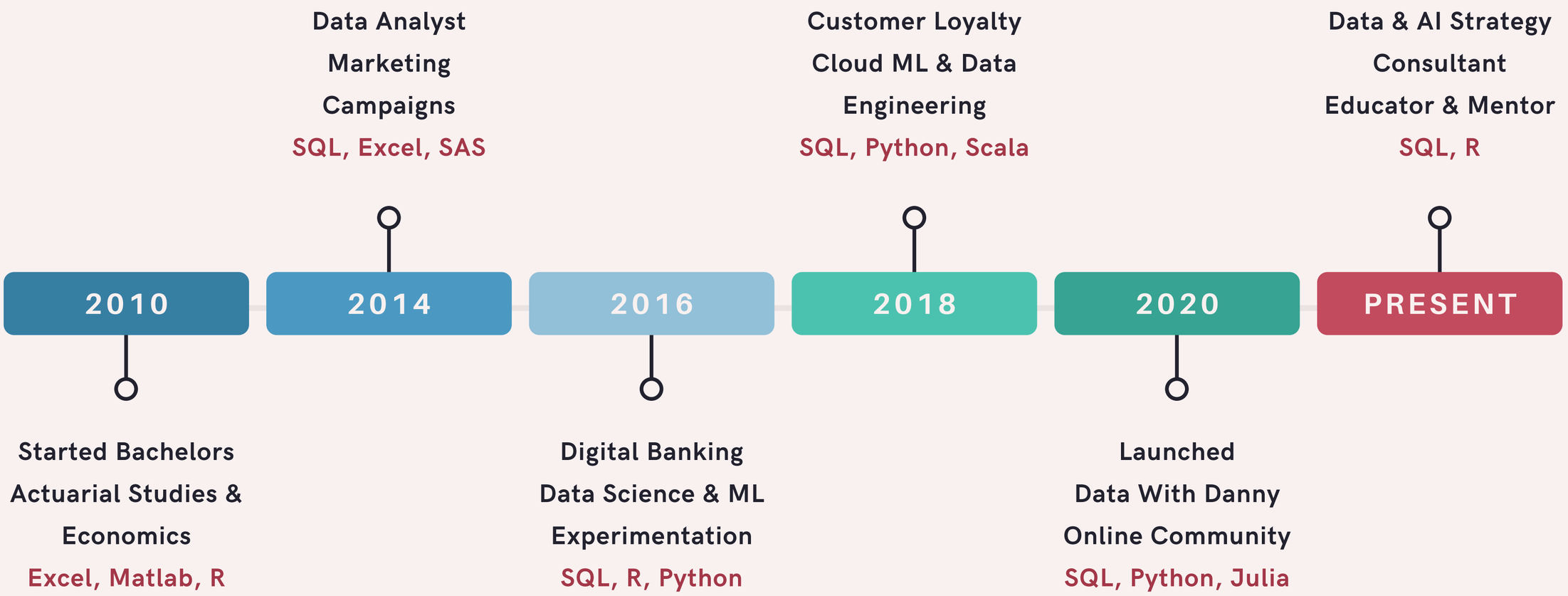
Pearson

My Data Analytics Journey

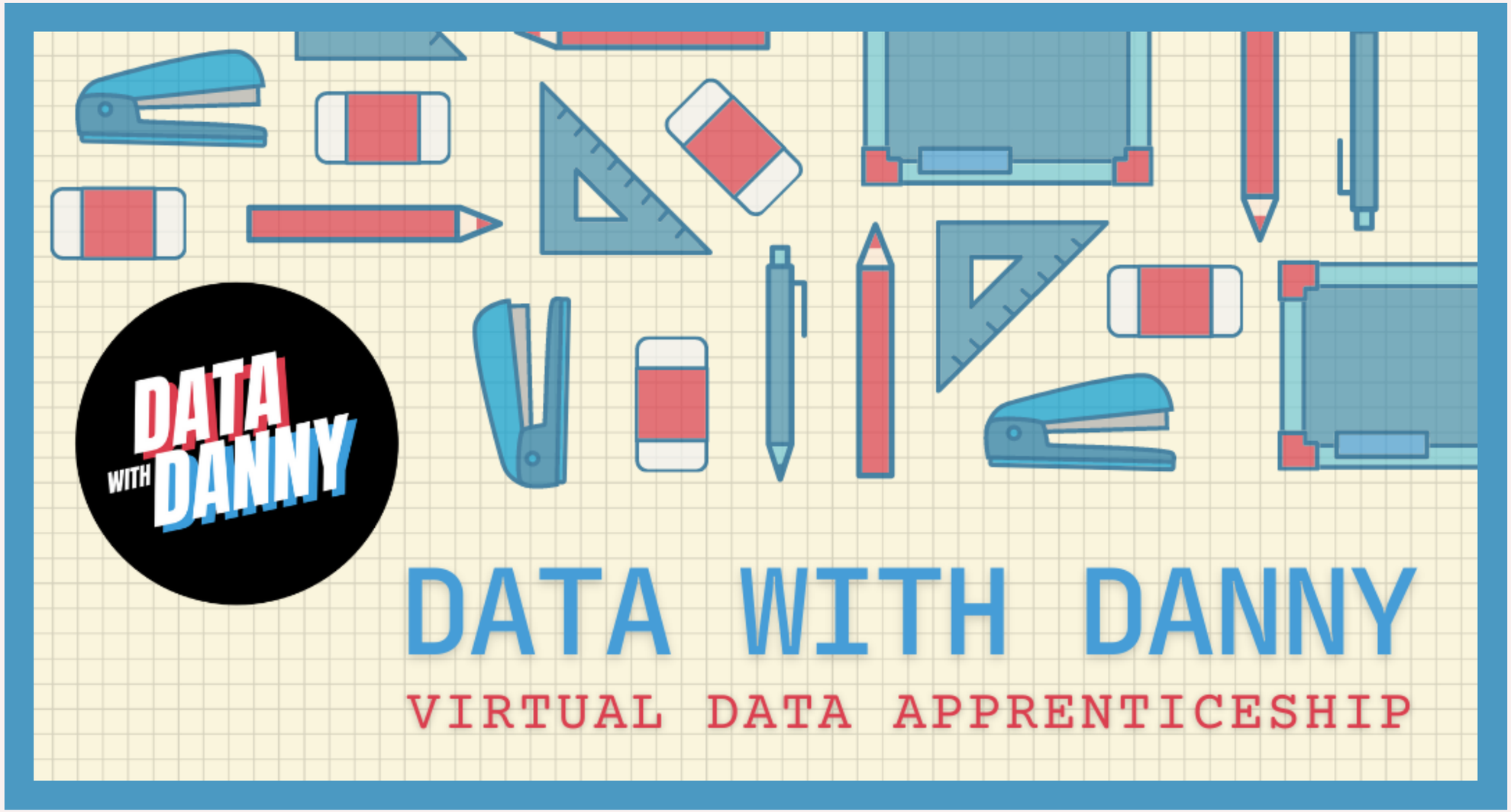


Pearson

My Data Analytics Journey



Pearson


DATA WITH DANNY
DATA WITH DANNY
VIRTUAL DATA APPRENTICESHIP

Pearson

How can I find out more about Data With Danny?

bit.ly/dwd-info

SQL FACTS






SQL FACTS

SQL appears in over 40% of all data job listings posted on Indeed in 2021

Pearson

SQL FACTS

65% of data analysts and data scientists use SQL according to a 2020 Stack Overflow survey

Pearson

SQL FACTS

SQL queries can be used with many languages and frameworks via objects and APIs to interact with databases

SQL FACTS

SQL is a standardised language making it a very versatile skill used across different domains and tech environments

Pearson

SQL FACTS

The modern data stack is moving to SQL-centric tools such as Snowflake, Amazon Redshift and Google BigQuery

SQL FACTS

SQL databases can store semi-structured data like JSON objects with expanding support for images, video and audio

SQL FACTS

SQL appears in over 40% of all data job listings posted on Indeed in 2021

SQL queries can be used with many languages and frameworks via objects and APIs to interact with databases

The modern data stack is moving to SQL-centric tools such as Snowflake, Amazon Redshift and Google BigQuery

65% of data analysts and data scientists use SQL according to a 2020 Stack Overflow survey

SQL is a standardised language making it a very versatile skill used across different domains and tech environments

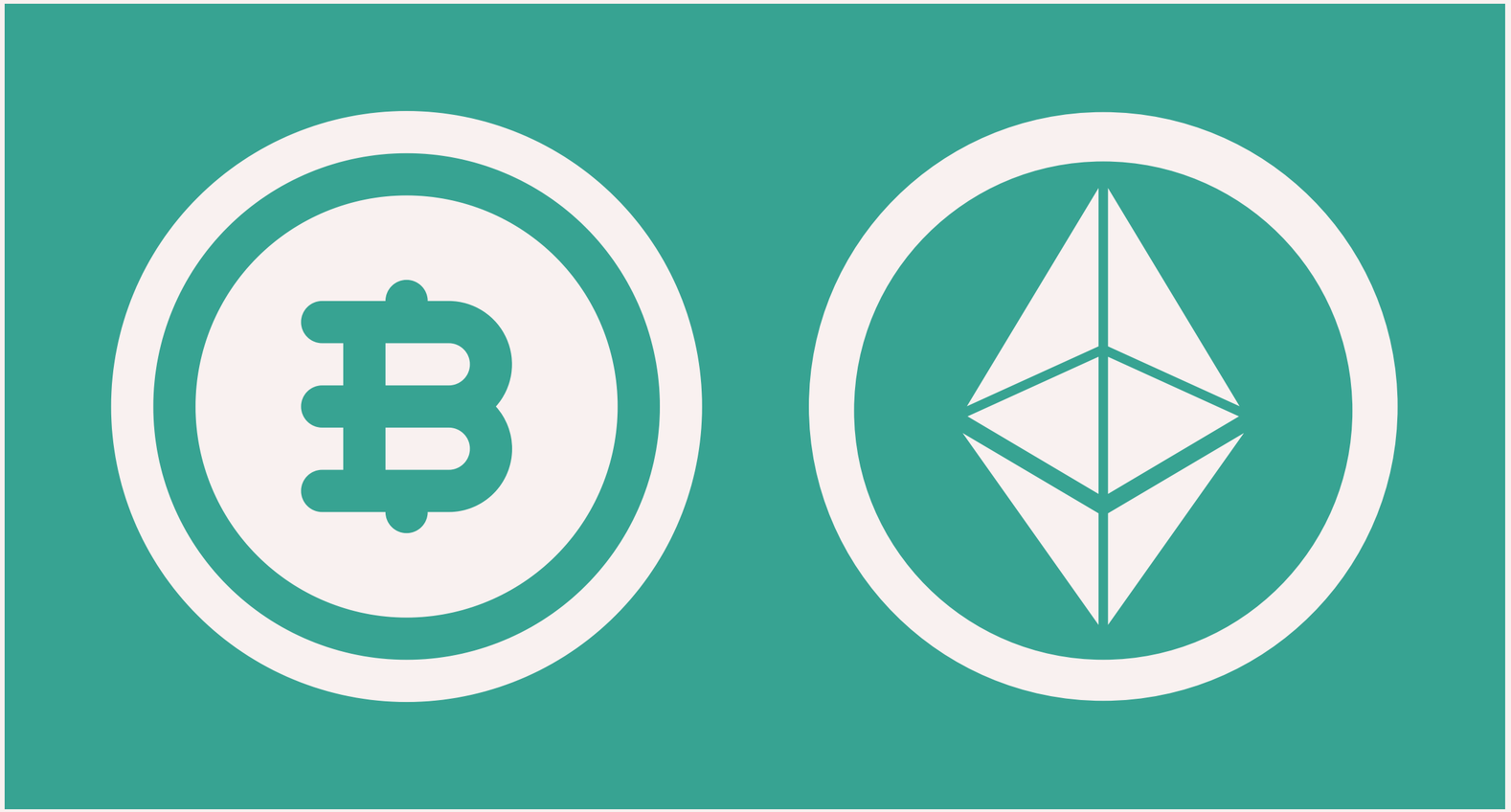
SQL databases can store semi-structured data like JSON objects with expanding support for images, video and audio

THE PLAN FOR TODAY

SQL
Simplified

General
Q&A

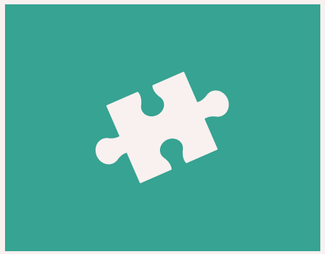
CRYPTO CASE STUDY



Pearson

SQL SIMPLIFIED

Session 1
Case Study Introduction



Session 2
Group By & Case When

Session 3
Window Functions

Session 4
Multiple Table Joins

Pearson

SQL SIMPLIFIED

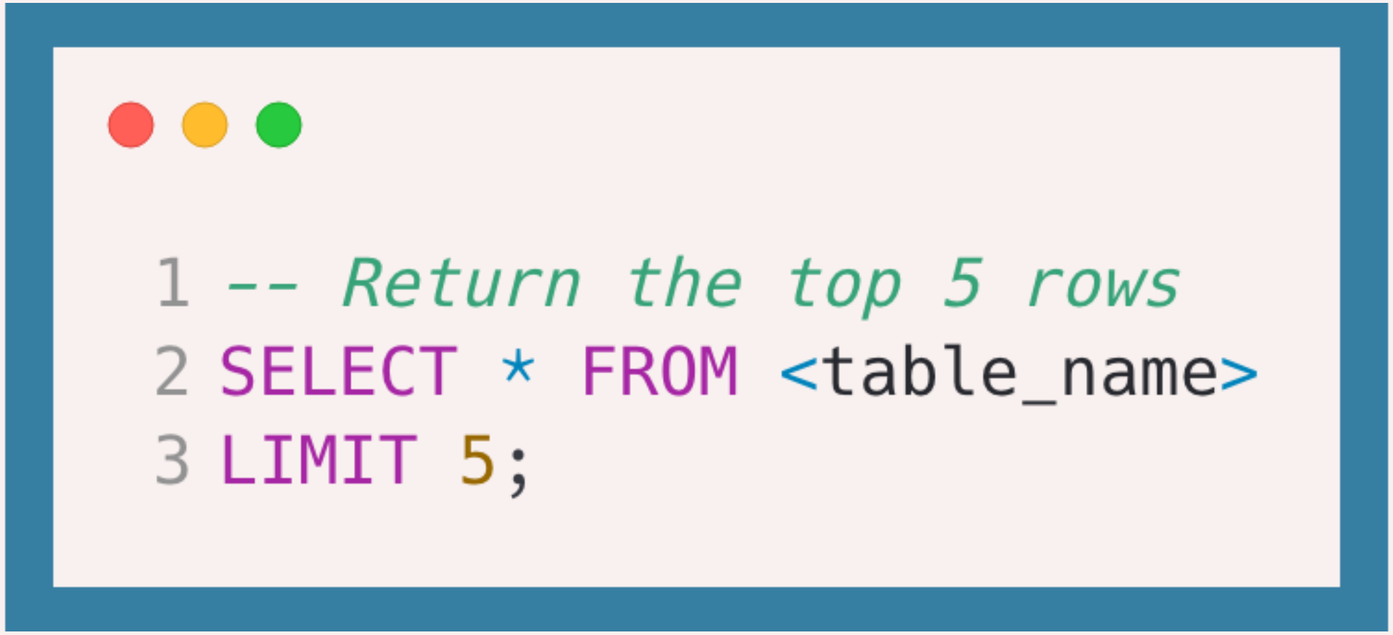
Session 1
Case Study Introduction

Pearson

SELECT STATEMENT

```sql
-- Return all columns and rows
SELECT * FROM <table_name>;
```

LIMIT



```sql
-- Return the top 5 rows
SELECT * FROM <table_name>
LIMIT 5;
```

ORDER BY

```sql
-- Order in alphabetical order
SELECT * FROM <table_name>
ORDER BY <string_column>;
```

COUNT

```sql
-- Count all rows including nulls
SELECT COUNT(*) FROM <table_name>;
```

```sql
-- Count non_null records in a column
SELECT COUNT(<column>) FROM <table_name>;
```

WHERE FILTERS

```sql
-- Where filter on string column
SELECT * FROM <table_name>
WHERE <string_column> = '<exact_string>';
```

```sql
-- Where filter to exclude records
SELECT * FROM <table_name>
WHERE <string_column> != '<exact_string>';
```

DISTINCT

```sql
-- Get unique values from a table
SELECT DISTINCT * FROM <table_name>;
```

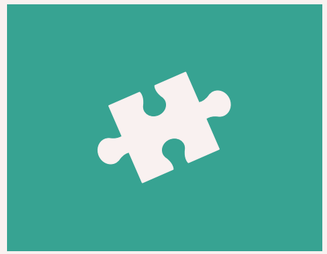
```sql
-- Get unique values from a column
SELECT DISTINCT <column> FROM <table_name>;
```

DISTINCT

```sql
-- Get unique values from 2+ columns
SELECT DISTINCT
  <column_1>,
  ...
  <column_n>
FROM <table_name>;
```

SQL SIMPLIFIED

Session 1
Case Study
Introduction



Session 2
Group By &
Case When

Pearson

BETWEEN

```sql
-- Where filter for date ranges
SELECT * FROM <table_name>
WHERE <date_column> BETWEEN '<start_date>' AND '<end_date>';
```

INEQUALITIES

```
1 -- Where filter using inequality
2 SELECT * FROM <table_name>
3 WHERE <column_name> {> or >= or < or <=} <exact_value>;
```

AGGREGATE FUNCTIONS

```sql
-- Min/max/average on a target column
-- using 1 single group by column
SELECT
  <group_by_column>,
  MIN(target_column),
  MAX(target_column),
  AVG(target_column)
FROM <table_name>
GROUP BY <group_by_column>;
```

EXTRACT DATE INFO

```sql
-- Extract a year date part from a date column
SELECT
  EXTRACT(YEAR FROM <date_column>)
FROM <table_name>;
```

DATE TRUNCATION

```sql
-- Extract 1st day of month from date column
SELECT
  DATE_TRUNC('MON', <date_column>)
FROM <table_name>;
```

CAST DATA TYPES

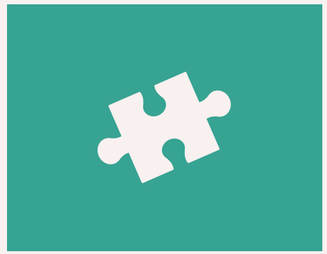
```sql
-- Cast float datatypes for rounding
SELECT
  ROUND(AVG(<numeric_column>)::NUMERIC, 2)
FROM <table_name>;
```

MULTI-LEVEL SORTING

```sql
-- Multi-level sort with more than 1 column
SELECT * FROM <table_name>
ORDER BY <column_1> [desc], <column_2> [desc];
```

SQL SIMPLIFIED

Session 1
Case Study
Introduction



Session 2
Group By &
Case When

Pearson

BASIC CASE WHEN

```sql
-- Basic case when statement structure
SELECT
  CASE
    WHEN <1st_if_condition> THEN <output_1>
    WHEN <2nd_if_condition> THEN <output_2>
    < Additional If-Then conditions go here >
    ELSE <output_3>
  END AS <alias_name>
```

SUM CASE WHEN: COUNTIF

```sql
-- Countif using sum case when
SELECT
  SUM(
    CASE
      WHEN <if_condition> THEN 1
      ELSE 0
    END
  )
FROM <table_name>;
```

SUM CASE WHEN: SUMIF

```sql
-- Sumif using sum case when
SELECT
  SUM(
    CASE
      WHEN <if_condition> THEN <then_output>
      ELSE 0
    END
  )
FROM <table_name>;
```

LEFT/RIGHT STRINGS

```sql
-- Find the n-th character from the left or right
SELECT
  LEFT(<string_column>, n) AS left_nth,
  RIGHT(<string_column>, n) AS right_nth
FROM <table_name>;
```

CHARACTER LENGTH

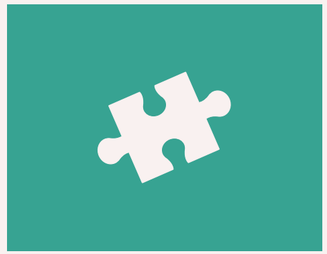
```sql
-- Find the character length of column
SELECT LENGTH(<string_column>) FROM <table_name>;
```

REMOVE LAST CHARACTER

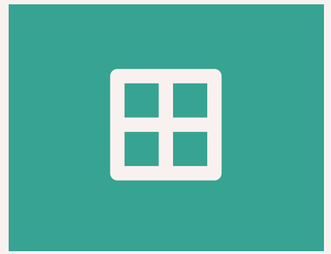
```sql
-- Remove the last character from string
SELECT
  LEFT(<string_column>, LENGTH(<string_column>) - 1))
FROM <table_name>;
```

SQL SIMPLIFIED

Session 1
Case Study
Introduction



Session 2
Group By &
Case When



Session 3
Window
Functions

Pearson

RANK WINDOW FUNCTION

```sql
-- Ranking window function
SELECT
  RANK() OVER (
    PARTITION BY <partition_column> ...
    ORDER BY <sort_column> ...
  ) AS ranking
FROM <table_name>;
```

7 DAY MOVING AVERAGE

```sql
-- 7 day moving average window function
SELECT
  AVG(<target_column>) OVER (
    PARTITION BY <partition_column> ...
    ORDER BY <date_column>  -- do not use DESC
    RANGE BETWEEN '7 DAYS' PRECEDING AND CURRENT ROW
  ) AS moving_avg
FROM <table_name>;
```

CUMULATIVE SUM

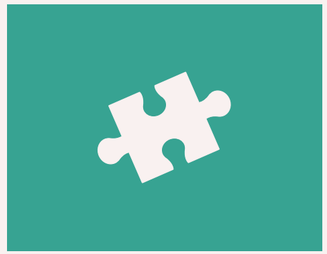
```sql
-- Cumulative sum window function
SELECT
  SUM(<target_column>) OVER (
    PARTITION BY <partition_column> ...
    ORDER BY <date_column>  -- do not use DESC
    RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
  ) AS cumulative_sum
FROM <table_name>;
```

LAG WINDOW FUNCTION

```sql
-- Lag window function
SELECT
  LAG(<target_column>) OVER (
    PARTITION BY <partition_column> ...
    ORDER BY <date_column>  -- do not use DESC
  ) AS previus_target_value
FROM <table_name>;
```

SQL SIMPLIFIED

Session 1
Case Study Introduction



Session 2
Group By & Case When

Session 3
Window Functions

Session 4
Multiple Table Joins

Pearson

INNER JOIN

```sql
-- Inner Join Between 2 Tables
SELECT
  members.region,
  SUM(transactions.quantity) AS bitcoin_quantity
FROM trading.transactions
INNER JOIN trading.members
  ON transactions.member_id = members.member_id
WHERE transactions.ticker = 'BTC'
  AND transactions.txn_type = 'BUY'
GROUP BY members.region
ORDER BY bitcoin_quantity DESC;
```

LEFT JOIN

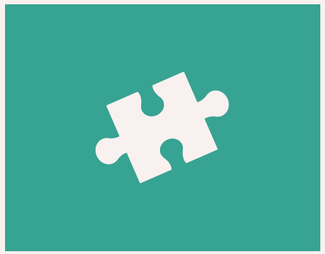
```sql
-- Left Join Between 2 Tables
SELECT
  prices.market_date,
  COUNT(transactions.txn_id) AS transaction_count
FROM trading.prices
LEFT JOIN trading.transactions
  ON prices.market_date = transactions.txn_date
  AND prices.ticker = transactions.ticker
GROUP BY prices.market_date
HAVING COUNT(transactions.txn_id) < 5
ORDER BY prices.market_date DESC;
```

MULTIPLE TABLE JOINS

```sql
-- Multiple Table Joins
SELECT
  members.region,
  SUM(transactions.quantity) AS btc_quantity,
  AVG(prices.price) AS avg_btc_price
FROM trading.transactions
INNER JOIN trading.prices
  ON transactions.ticker = prices.ticker
  AND transactions.txn_date = prices.market_date
INNER JOIN trading.members
  ON transactions.member_id = members.member_id
WHERE transactions.ticker = 'BTC'
  AND transactions.txn_type = 'BUY'
GROUP BY members.region
ORDER BY avg_btc_price DESC;
```

SQL SIMPLIFIED COMPLETED!

Session 1
Case Study Introduction



Session 2
Group By & Case When

Session 3
Window Functions

Session 4
Multiple Table Joins

Pearson

General Q&A

bit.ly/dwd-info