**Using joins to Analyze Sales Dealership**

The head of sales at the company would like a list of all customers who bought a car.

I'll create a query that will return all customer IDs, first names, last names and valid phone numbers of customers who purchased a car.

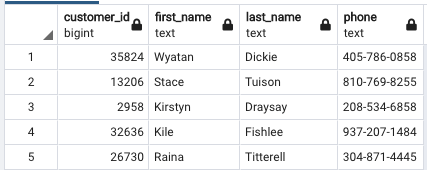
SELECT c.customer\_id, c.first\_name, c.last\_name, c.phone

FROM sales s

INNER JOIN customers c ON s.customer\_id=c.customer\_id

INNER JOIN products p ON s.product\_id=p.product\_id

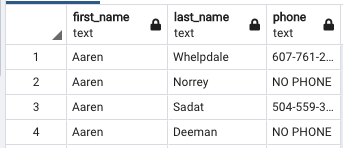
WHERE p.product\_type='automobile' AND c.phone IS NOT NULL



SELECT first\_name, last\_name, COALESCE(phone, 'NO PHONE') AS phone

FROM customers

ORDER BY 1;



**Generating an Elite Customer Party Guest List using UNION**

A query that will make a list of the company's customers and company employees

Who live in Los Angeles, CA. The guest list should contain the first and last name, and whether the guest is a customer or an employee.

(

SELECT first\_name, last\_name, 'Customer' AS guest\_type

FROM customers

WHERE city='Los Angeles' AND state='CA'

)

UNION

(

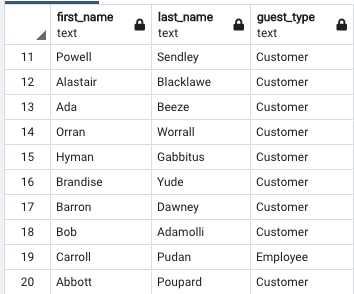
SELECT first\_name, last\_name, 'Employee' as guest\_type

FROM salespeople s

INNER JOIN dealerships d ON s.dealership\_id=d.dealership\_id

WHERE d.city ='Los Angeles' AND d.state = 'CA'

)



**Using the CASE WHEN Function to Get Regional Lists**

The aim is to create a query that will map various values in a column to other values.

SELECT c.customer\_id,

CASE WHEN c.state IN ('MA','NH','VT','ME','CT','RI') THEN 'New England'

WHEN c.state IN ('GA','FL','MS','AL','LA','KY','VA','NC','SC','TN','VI','WV','AR') THEN 'Southeast'

ELSE 'Other' END AS region

FROM customers c

ORDER BY 1;



**Building a Sales Model Using SQL Techniques**

Clean and prepare the data for analysis using SQL techniques.

The data science team wants to build a new model to help predict which customers are the best prospects for remarketing.

A new data scientist has joined their team and does not know the database well enough to pull a dataset for this new model.

The responsibility has fallen to the data analyst to help the data scientist prepare and build a dataset to be used to train a model.

SELECT c.\*,

p.\*,

COALESCE(s.dealership\_id,-1) as dealership\_id,

CASE WHEN (p.base\_msrp - s.sales\_amount > 500) THEN 1

ELSE 0 END AS high\_savings

FROM sales s

INNER JOIN customers c ON s.customer\_id = c.customer\_id

INNER JOIN products p ON p.product\_id = s.product\_id

LEFT JOIN dealerships d ON s.dealership\_id = d.dealership\_id;

**Using Aggregate Functions to Analyze Data**

Analyze the price of a products using different aggregate functions.

SELECT product\_type, MIN(base\_msrp), MAX(base\_msrp), AVG(base\_msrp), STDDEV(base\_msrp)

FROM products

GROUP BY product\_type;

**Data Replacement Techniques**

**COALESCE** – To replace NULL values with a standard value.

USE CASE: We want to query from the DB, fill N

ULL values of customers phone numbers with 'NO PHONE' value.

SELECT first\_name, last\_name, COALESCE(phone, 'NO PHONE') as phone

FROM customers

ORDER BY 1;

**NULLIF** – The opposite of COALESCE, will return NULL if the value is found.

USE CASE: We want to create a direct mail piece to send to the customer, but it cannot accept people's titles longer then 3 chars. In our DB the only title is 'Honorable'.

SELECT customer\_id,

NULLIF(title, 'Honorable') as title,

first\_name,

last\_name,

suffix,

email,

gender,

ip\_address,

phone,

street\_address,

city,

state,

postal\_code,

latitude,

longitude,

date\_added

FROM customers c

ORDER BY 1;

**LEAST/GREATEST** – Two functions that come in handy for data preparation.

Each function takes any number of values and returns the least or the greatest of the values, respectively.

USE CASE – Replace a value if it's too high or low.

SELECT product\_id,

model,

year,

product\_type,

LEAST(600, base\_msrp) as base\_msrp,

production\_start\_date,

production\_end\_date

FROM products

WHERE product\_type = 'scooter'

ORDER BY 1;

**CASTING** – Used for data transformation to change the data type of a column within a query.

USE CASE – Wanting to a apply a function available only on text data while working with a column that's numeric.

SELECT product\_id,

model,

year::TEXT,

product\_type,

base\_msrp,

production\_start\_date,

production\_end\_date

FROM products;

**DISTINCT and DISTINCT ON**

USE CASE – determining the unique values in a column or a group of columns.

SELECT DISTINCT(year)

FROM products

ORDER BY 1;

SELECT DISTINCT year, product\_type

FROM products

ORDER BY 1, 2;

**DISTINCT ON** – allows you to ensure that only one row is returned where one or more columns are always unique in the set.

I want to get a unique list of salespeople where each salesperson has a unique first name. In the case that two salespeople have the same first name, we will return the one who started earlier.

SELECT DISTINCT ON (first\_name)

\*

FROM salespeople

ORDER BY first\_name, hire\_date;

**GROUP BY** – A clause that divides the rows of a dataset into multiple groups based on some sort of key specified in the GROUP BY clause. An aggregate function is then applied to all the rows within a single group. To produce a single number.

SELECT {**KEY**}, {AGGFUNC{column1}} FROM {table} GROUP BY {**KEY**};