Using SQL Queries

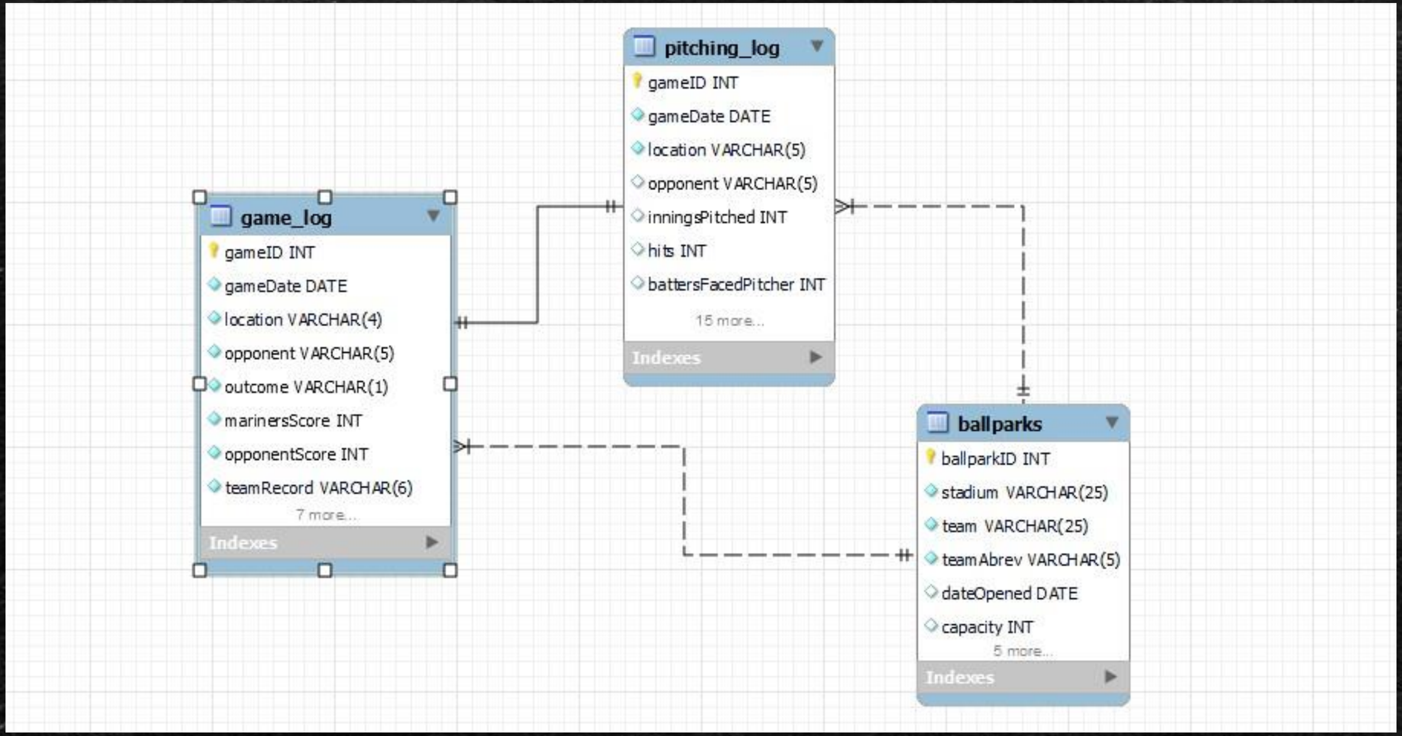
From analyzing the Seattle Mariner’s player statistics to running a small produce store. Databases are needed everywhere!

Sports have an endless amount of data…

- Sports data is always used by teams to make the toughest decisions.
- Teams want to be the best, and if they can't find their problems, how are they going to fix them?
- Want to pay players with incentives? Owners set statistical goals that players need to meet in order to make the big bucks!
- How else can you strategize against other teams? Find their weaknesses within the data!
- There's an endless amount of ways data can be used, **WE NEED IT!**

Looking at the 2022 Mariners:

The Mariners had a breakout season in 2022, making the playoffs for the first time since 2001! Let's analyze out how they made that happen.



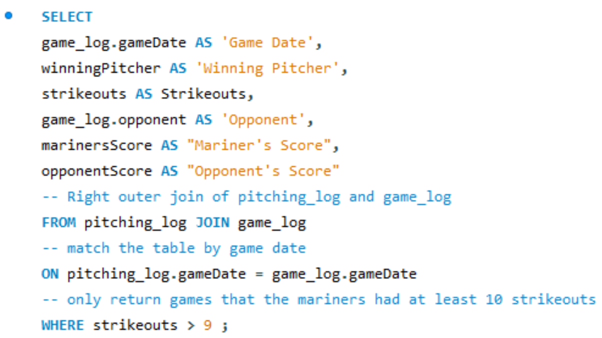
pitching_log
- gameID INT
- gameDate DATE
- location VARCHAR(5)
- opponent VARCHAR(5)
- inningsPitched INT
- hits INT
- battersFacedPitcher INT
- 15 more...
- Indexes

game_log
- gameID INT
- gameDate DATE
- location VARCHAR(4)
- opponent VARCHAR(5)
- outcome VARCHAR(1)
- marinersScore INT
- opponentScore INT
- teamRecord VARCHAR(6)
- 7 more...
- Indexes

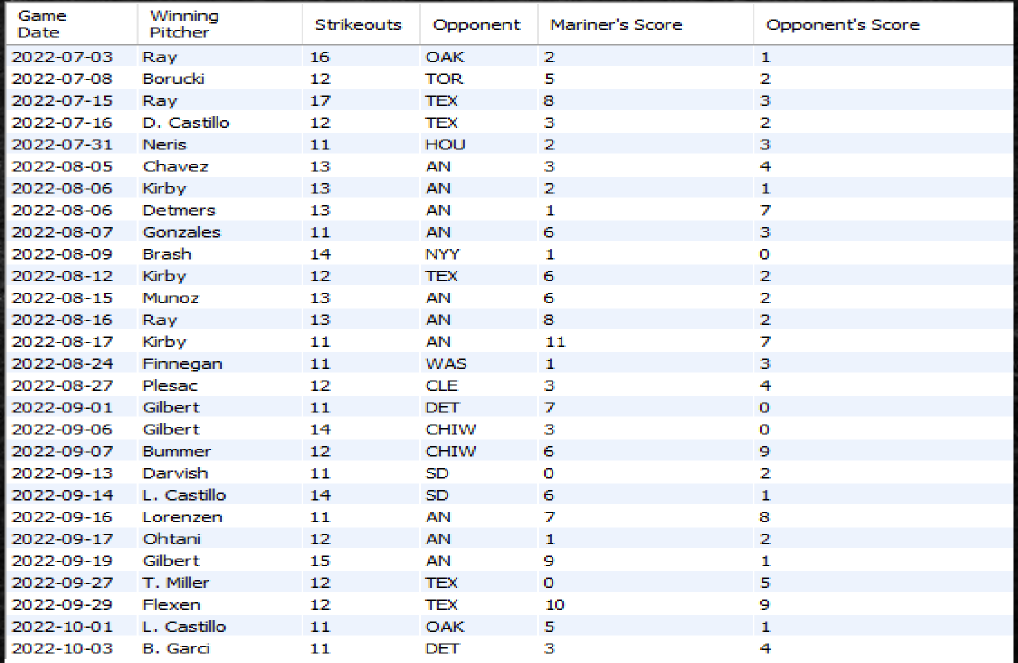
ballparks
- ballparkID INT
- stadium VARCHAR(25)
- team VARCHAR(25)
- teamAbrev VARCHAR(5)
- dateOpened DATE
- capacity INT
- 5 more...
- Indexes

Mariner’s pitching in 2022 was UNSTOPPABLE. Let’s take a look:

We can join our game_log and pitching_log to show games the Mariners pitched 10+ strikeouts.



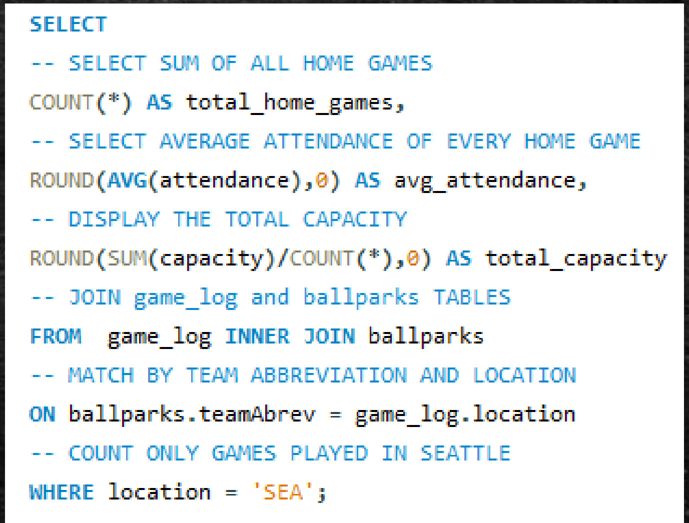
```sql
SELECT
game_log.gameDate AS 'Game Date',
winningPitcher AS 'Winning Pitcher',
strikeouts AS Strikeouts,
game_log.opponent AS 'Opponent',
marinersScore AS "Mariner's Score",
opponentScore AS "Opponent's Score"
-- Right outer join of pitching_log and game_log
FROM pitching_log JOIN game_log
-- match the table by game date
ON pitching_log.gameDate = game_log.gameDate
-- only return games that the mariners had at least 10 strikeouts
WHERE strikeouts > 9 ;
```



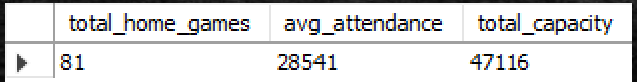
Game Date	Winning Pitcher	Strikeouts	Opponent	Mariner's Score	Opponent's Score
2022-07-03	Ray	16	OAK	2	1
2022-07-08	Borucki	12	TOR	5	2
2022-07-15	Ray	17	TEX	8	3
2022-07-16	D. Castillo	12	TEX	3	2
2022-07-31	Neris	11	HOU	2	3
2022-08-05	Chavez	13	AN	3	4
2022-08-06	Kirby	13	AN	2	1
2022-08-06	Detmers	13	AN	1	7
2022-08-07	Gonzales	11	AN	6	3
2022-08-09	Brash	14	NYY	1	0
2022-08-12	Kirby	12	TEX	6	2
2022-08-15	Munoz	13	AN	6	2
2022-08-16	Ray	13	AN	8	2
2022-08-17	Kirby	11	AN	11	7
2022-08-24	Finnegan	11	WAS	1	3
2022-08-27	Plesac	12	CLE	3	4
2022-09-01	Gilbert	11	DET	7	0
2022-09-06	Gilbert	14	CHIW	3	0
2022-09-07	Bummer	12	CHIW	6	9
2022-09-13	Darvish	11	SD	0	2
2022-09-14	L. Castillo	14	SD	6	1
2022-09-16	Lorenzen	11	AN	7	8
2022-09-17	Ohtani	12	AN	1	2
2022-09-19	Gilbert	15	AN	9	1
2022-09-27	T. Miller	12	TEX	0	5
2022-09-29	Flexen	12	TEX	10	9
2022-10-01	L. Castillo	11	OAK	5	1
2022-10-03	B. Garci	11	DET	3	4

Mariner’s Success Would Not Go Unnoticed

When using a join statement with some simple summary queries, we can find the average attendance for games played in Seattle.



```
SELECT
-- SELECT SUM OF ALL HOME GAMES
COUNT(*) AS total_home_games,
-- SELECT AVERAGE ATTENDANCE OF EVERY HOME GAME
ROUND(AVG(attendance),0) AS avg_attendance,
-- DISPLAY THE TOTAL CAPACITY
ROUND(SUM(capacity)/COUNT(*),0) AS total_capacity
-- JOIN game_log and ballparks TABLES
FROM  game_log INNER JOIN ballparks
-- MATCH BY TEAM ABBREVIATION AND LOCATION
ON ballparks.teamAbrev = game_log.location
-- COUNT ONLY GAMES PLAYED IN SEATTLE
WHERE location = 'SEA';
```

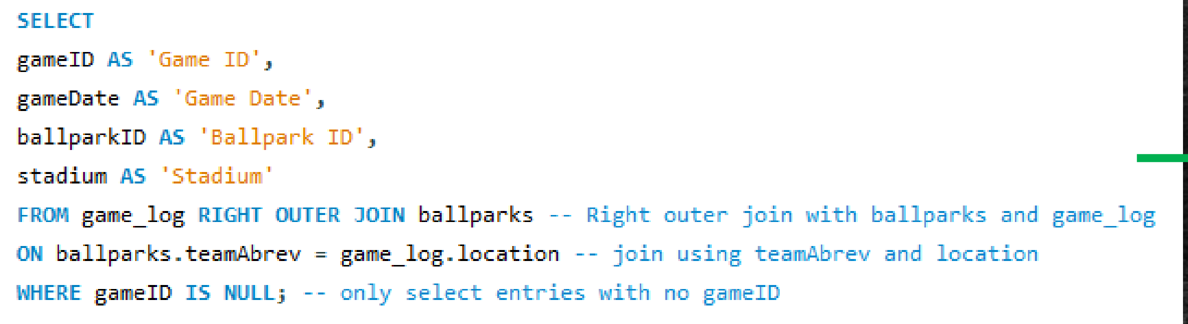


	total_home_games	avg_attendance	total_capacity
▶	81	28541	47116

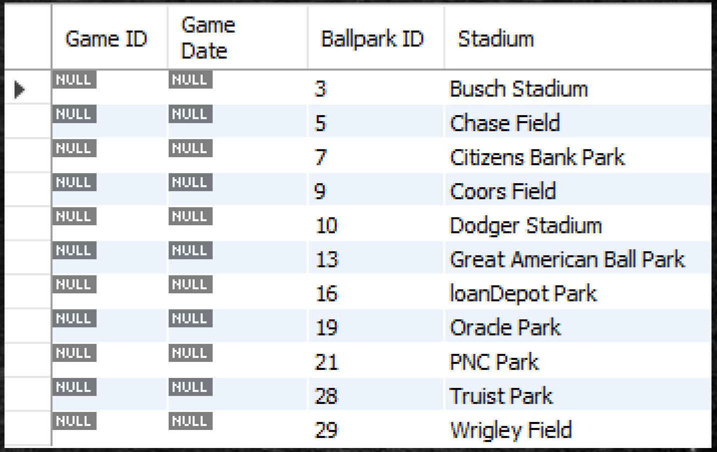
The Mariners finished top 15 in attendance for 2022!

Outer Join Queries

Using an outer join, we can find missing data between two tables (ballparks and game_log).



```
SELECT
gameID AS 'Game ID',
gameDate AS 'Game Date',
ballparkID AS 'Ballpark ID',
stadium AS 'Stadium'
FROM game_log RIGHT OUTER JOIN ballparks -- Right outer join with ballparks and game_log
ON ballparks.teamAbrev = game_log.location -- join using teamAbrev and location
WHERE gameID IS NULL; -- only select entries with no gameID
```



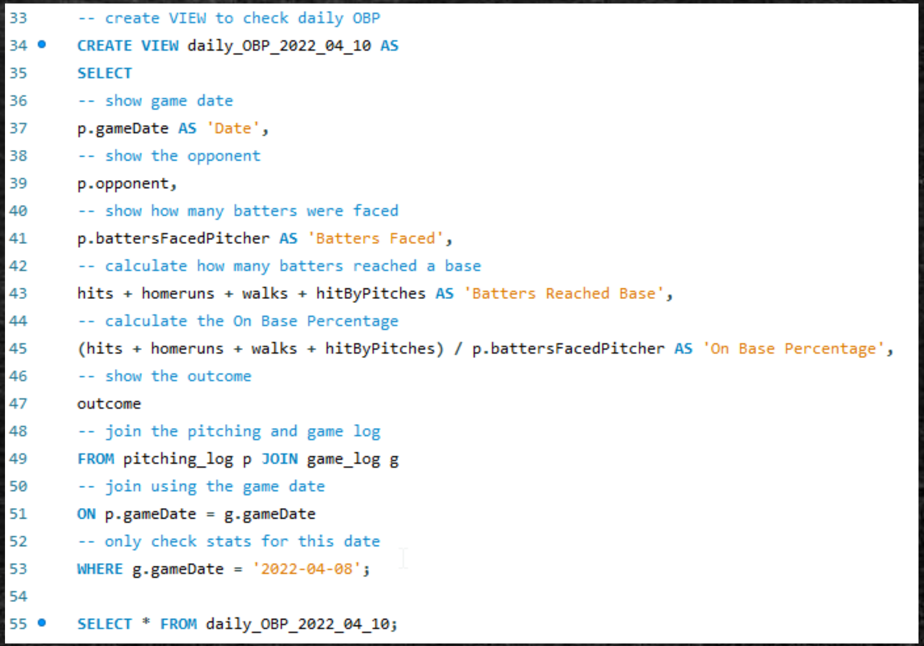
Game ID	Game Date	Ballpark ID	Stadium
NULL	NULL	3	Busch Stadium
NULL	NULL	5	Chase Field
NULL	NULL	7	Citizens Bank Park
NULL	NULL	9	Coors Field
NULL	NULL	10	Dodger Stadium
NULL	NULL	13	Great American Ball Park
NULL	NULL	16	loanDepot Park
NULL	NULL	19	Oracle Park
NULL	NULL	21	PNC Park
NULL	NULL	28	Truist Park
NULL	NULL	29	Wrigley Field

The results table shows us that we are missing game results for each of these stadiums. This concludes that the Mariners never played a game at those stadiums in 2022!

Keeping Track of Opponent’s On Base Percentage

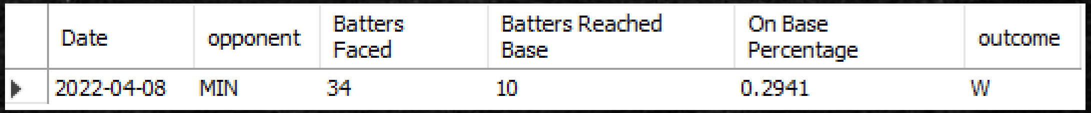
Using a View, we can keep track of daily OBP with the same select statements **AND** we can use previous views just like any other table to analyze more data!

I’ve been tasked of checking the daily opponent’s on base percentage and reporting it back to the club house after every game…



```sql
-- create VIEW to check daily OBP
CREATE VIEW daily_OBP_2022_04_10 AS
SELECT
-- show game date
p.gameDate AS 'Date',
-- show the opponent
p.opponent,
-- show how many batters were faced
p.battersFacedPitcher AS 'Batters Faced',
-- calculate how many batters reached a base
hits + homeruns + walks + hitByPitches AS 'Batters Reached Base',
-- calculate the On Base Percentage
(hits + homeruns + walks + hitByPitches) / p.battersFacedPitcher AS 'On Base Percentage',
-- show the outcome
outcome
-- join the pitching and game log
FROM pitching_log p JOIN game_log g
-- join using the game date
ON p.gameDate = g.gameDate
-- only check stats for this date
WHERE g.gameDate = '2022-04-08';

SELECT * FROM daily_OBP_2022_04_10;
```



Date	opponent	Batters Faced	Batters Reached Base	On Base Percentage	outcome
2022-04-08	MIN	34	10	0.2941	W

Database Importance In Businesses

- Provide a structured way to organize and manage business data.

- Allows businesses to personalize customer experience, target marketing campaigns, and enhance satisfaction.

- Provide scalability and growth allowing the ability to handle more data as time goes on.

- Provide support to management especially with adhering to employee needs.

Database Structure In Business

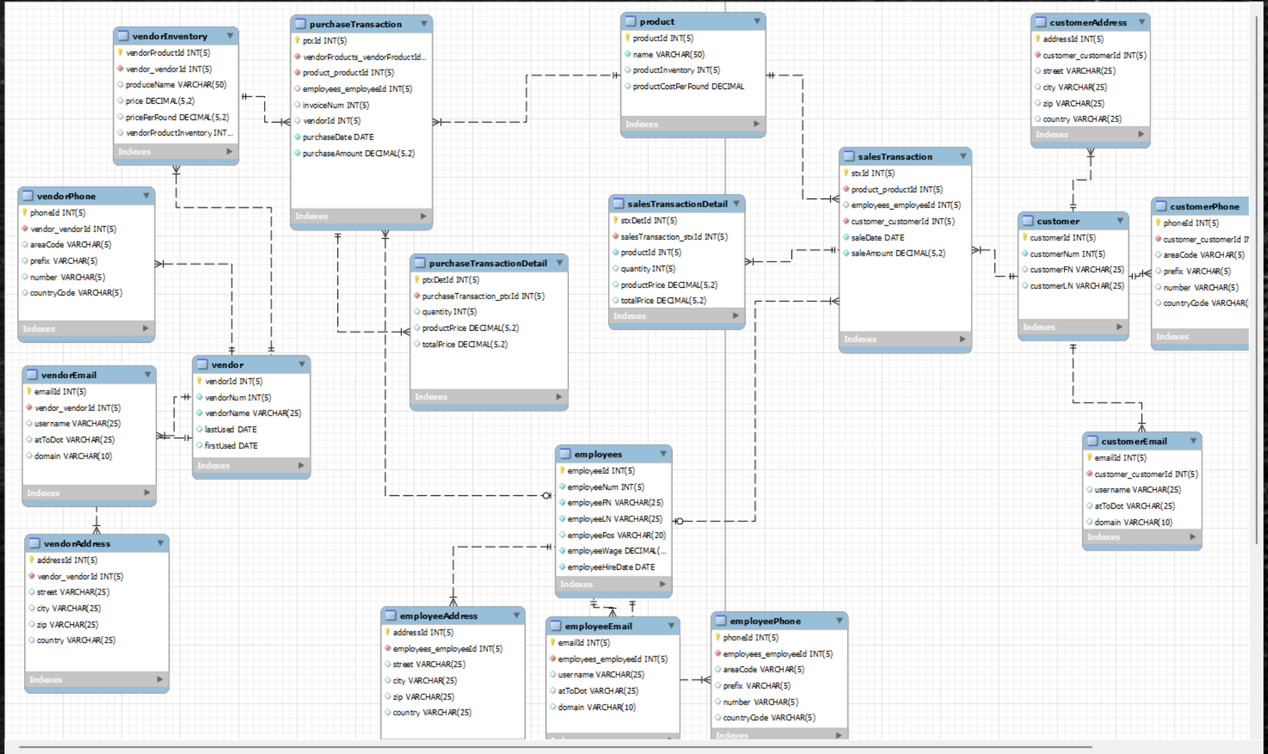
Database structure is determined by the needs of a business.

In the case of opening a small produce store, I need to keep track of lots of data including:

- Customers
- Employees
- Vendors
- Vendor Transactions
- Customer Transactions
- Product Inventory

And more…

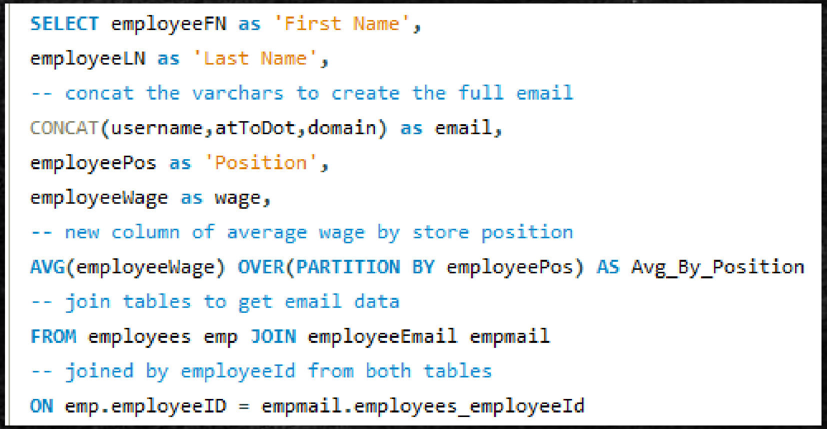
It's also important to make proper relationships between tables:



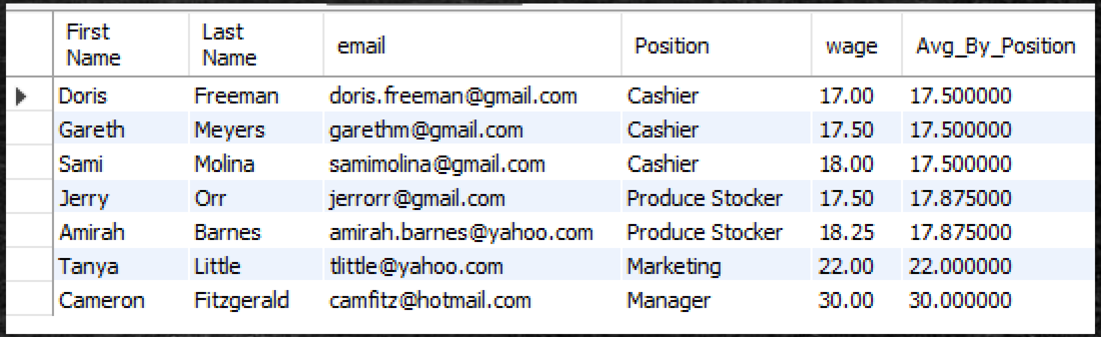
A Business Can’t Run Without Employees!

Employees are the heart of a business, so we need to take care of them!

With simple summary queries, joins, and an OVER statement…



```
SELECT employeeFN as 'First Name',
employeeLN as 'Last Name',
-- concat the varchars to create the full email
CONCAT(username,atToDot,domain) as email,
employeePos as 'Position',
employeeWage as wage,
-- new column of average wage by store position
AVG(employeeWage) OVER(PARTITION BY employeePos) AS Avg_By_Position
-- join tables to get email data
FROM employees emp JOIN employeeEmail empmail
-- joined by employeeId from both tables
ON emp.employeeID = empmail.employees_employeeId
```

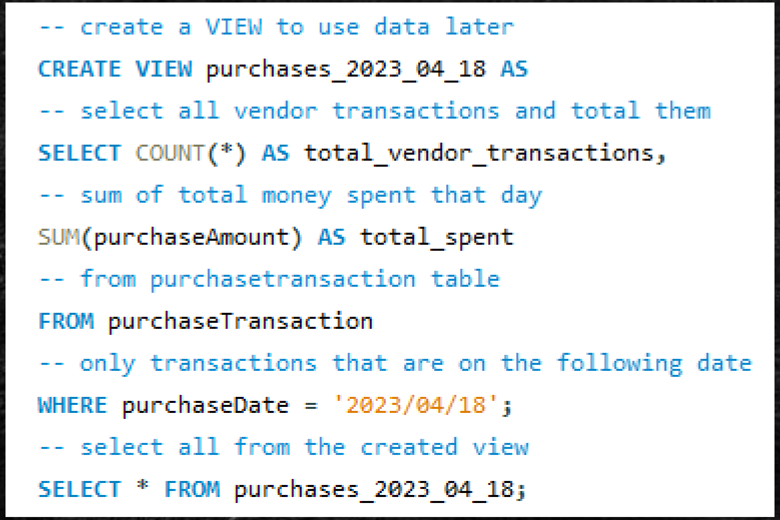


First Name	Last Name	email	Position	wage	Avg_By_Position
Doris	Freeman	doris.freeman@gmail.com	Cashier	17.00	17.500000
Gareth	Meyers	garethm@gmail.com	Cashier	17.50	17.500000
Sami	Molina	samimolina@gmail.com	Cashier	18.00	17.500000
Jerry	Orr	jerrorr@gmail.com	Produce Stocker	17.50	17.875000
Amirah	Barnes	amirah.barnes@yahoo.com	Produce Stocker	18.25	17.875000
Tanya	Little	tlittle@yahoo.com	Marketing	22.00	22.000000
Cameron	Fitzgerald	camfitz@hotmail.com	Manager	30.00	30.000000

I can check on my employees, and make sure they are being paid fairly. We can see average wages by position along side current personal wages!

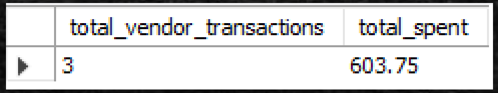
Managing Money With Databases

Without organization or attention to purchases/sales, businesses risk losing money!



```
-- create a VIEW to use data later
CREATE VIEW purchases_2023_04_18 AS
-- select all vendor transactions and total them
SELECT COUNT(*) AS total_vendor_transactions,
-- sum of total money spent that day
SUM(purchaseAmount) AS total_spent
-- from purchasetransaction table
FROM purchaseTransaction
-- only transactions that are on the following date
WHERE purchaseDate = '2023/04/18';
-- select all from the created view
SELECT * FROM purchases_2023_04_18;
```

Data like this can now be used for later queries/analysis!



	total_vendor_transactions	total_spent
▶	3	603.75

To Conclude…

- SQL is **powerful** and is the most popular/preferred language for databases worldwide.
- Databases have endless capabilities in the field of business/data management and serve as the backbone of modern information systems.
- Learning SQL is a valuable skill for anyone involved in data analysis, database administration, and software development!