SQL

“Structured Query Language”

SQL

The swiss army knife for data

⚡ guest talk by DC Posch

https://dcpos.ch

SQL Introduction

Pros:

Universal. Declarative. Timeless, never goes out of fashion.

SQL Introduction

Pros:

Universal. Declarative. Timeless, never goes out of fashion.

A powerful tool both for production use and for data analysis.

SQL Introduction

Plan

- SQL Crash Course + challenge
- Core Concepts + challenge
- Advanced SQL
- Conclusion

SQL Crash Course

SQL Crash Course

SELECT to read (and transform, and summarize, and analyze) data.

INSERT, **UPDATE**, **DELETE** to write data.

Reading data

SQL Crash Course

```
SELECT *
  FROM plays;
```

SQL Crash Course

```
SELECT play_id, type, player,
       assist, points, description

  FROM plays;
```

SQL Crash Course

Try it! Please fork it:

https://www.db-fiddle.com/f/tzUp9abziH3mXxvhWoeKMm/0

Filtering

SQL Crash Course

```
SELECT *
  FROM plays
  WHERE player='Lebron James'
```

SQL Crash Course

```
SELECT *
  FROM plays
  WHERE player='Lebron James'
  AND event_type in ('shot', 'miss')
```

Grouping

SQL Crash Course

```
SELECT team, player, points
  FROM plays;
```

SQL Crash Course

```
SELECT team, player, sum(points)
  FROM plays
  GROUP BY team, player
  ORDER BY sum(points) desc;
```

SQL Crash Course

```
SELECT player, event_type, count(*)
  FROM plays
  WHERE event_type in ('shot', 'miss')
  GROUP BY player, event_type;
```

Challenge

player	shotsTaken	shotsMade
Anthony Davis	24	10
	...	

Questions so far?

SQL Concepts

SQL Concepts

Relation (n): fancy word for row

Foreign keys and constraints

Join: combining data from two tables

Transaction: updating data atomically

ACID: atomicity, consistency, isolation, and durability

Join

SQL Concepts

```
SELECT *
  FROM order_cart_items ci
  INNER JOIN items i ON i.id=ci.item_i;
```

SQL Concepts

```
SELECT brand, sum(quantity)
  FROM order_cart_items ci
  INNER JOIN items i ON i.id=ci.item_id
  GROUP BY i.brand;
```

SQL Concepts

```
SELECT brand, sum(quantity)
  FROM order_cart_items ci
  JOIN items i ON i.id=ci.item_id
  GROUP BY i.brand;
```

SQL Concepts

Quick about about minimizing roundtrips

SQL Concepts

```
insert into order_cart_items
(order_id,item_id,quantity)
  values (1, 1, 3),(1, 3, 2),(1, 4, 1),
  (2, 1, 1),(2, 4, 1);
```

SQL Concepts

Try it, this time with joins!

Please fork:

https://www.db-fiddle.com/f/pTChybKEAvb6krgJdo3sb3/3

Left join

*For when you want all the rows in A,
even when there's no match over in B*

SQL Concepts

```
SELECT name, sum(quantity)
  FROM items i
  JOIN order_cart_items ci
  ON i.id=ci.item_id
  GROUP BY name;
```

Challenge

Fix that query to show the missing item.

Advanced SQL

Transactions

SQL Advanced: Transactions

```
BEGIN TRANSACTION

UPDATE accounts SET bal=bal-amount
WHERE id=100;
UPDATE accounts SET bal=bal+amount
WHERE id=200;
COMMIT
```

SQL Advanced: Transactions

The core intuition for transactions: a transaction either happens all the way, or not at all. "Atomicity"

SQL Advanced: Transactions

ACID = Atomicity +

Isolation = you'll never see a "half-way" finished transaction.

Consistency = constraints still hold

Durability = what's done is done

Normalization

SQL Advanced: Normalization

Every bit of information should have a single source of truth. Don't paste the same stuff into two different tables.

Window functions

SQL Advanced: Window functions

Window functions let you do rankings, running totals, and similar. They allow a result row from a query to depend on the contents of *other* result rows.

SQL Advanced: Window Functions

```
SELECT …,
  RANK(points) OVER(PARTITION BY team)
  FROM plays;
```

JSON

SQL Advanced: JSON

- Lets you mix document-style with relational DB.

- Saves you a ton of tables.

- Makes it easier to atomically update a whole complex object.

- Rule of thumb: no ID references inside of JSON. Just plain data.

SQL Advanced: JSON

```
SELECT
  id, docId
  slideJson
FROM slides;
```

SQL Advanced: JSON

```
SELECT
  slideJson->'$.bgColor' as col,
  count(*)
FROM slides
GROUP BY bgColor;
```

Quick demo

Questions

dcposch@dcpos.ch