Building and Organizing Complex Queries: Takeaways

Syntax

- Using the WITH clause:

```
WITH track_info AS

(

 SELECT

     t.name,

     ar.name artist,

     al.title album_name,

 FROM track t

 INNER JOIN album al ON al.album_id = t.album_id

 INNER JOIN artist ar ON ar.artist_id = al.artist_id

)

SELECT * FROM track_info

WHERE album_name = "Jagged Little Pill";
```

- Creating a view:

```
CREATE VIEW chinook.customer_2 AS

SELECT * FROM chinook.customer;
```

- Dropping a view

```
DROP VIEW chinook.customer_2;
```

- Selecting rows that occur in one or more SELECT statements:

```
[select_statement_one]

UNION

[select_statement_two];
```

- Selecting rows that occur in both SELECT statements:

```
SELECT * from customer_usa

INTERSECT

SELECT * from customer_gt_90_dollars;
```

- Selecting rows that occur in the first SELECT statement but not the second SELECT statement:

```
SELECT * from customer_usa

EXCEPT

SELECT * from customer_gt_90_dollars;
```

- Chaining WITH statements:

```
WITH

usa AS

    (

    SELECT * FROM customer

    WHERE country = "USA"

    ),

last_name_g AS

    (

     SELECT * FROM usa

     WHERE last_name LIKE "G%"

    ),

state_ca AS

    (

    SELECT * FROM last_name_g

    WHERE state = "CA"

    )

SELECT
```

Concepts

- A few tips to help make your queries more readable:
  - If a select statement has more than one column: put each selected column on a new line, indented from the select statement.
  - Always capitalize SQL function names and keywords.
  - Put each clause of your query on a new line.
  - Use indenting to make subqueries appear logically separate.

```
    first_name,
    last_name,
    country,
    state
FROM study_cc
```

- A `WITH` statement helps a lot when your main query has some slight complexities.
- A view is a permanently defined `WITH` statement that you can use in all future queries.
- Redefining a view requires having to delete or drop the existing view.
- Statements before and after `UNION` clause must have the same number of columns, as well as compatible data types.
- Comparison of `UNION`, `INTERSECT`, and `EXCEPT`:

Operator	What it Does	Python Equivalent
`UNION`	Selects rows that occur in either statement.	`or`
`INTERSECT`	Selects rows that occur in both statements.	`and`
`EXCEPT`	Selects rows that occur in the first statement, but don't occur in the second statement.	`and not`

Resources

- SQL Style Guide
- Set Operations