SQL Style Guide

Let's make our code look structured!

Basic SQL formatting example

```
WITH player_events AS (
    SELECT player_id,
           customer_id,
           CASE
             WHEN event_type = 'Card Game' THEN TRUE
             ELSE FALSE
           END AS is_casino_game,
           SUM(win) AS total_wins
      FROM dwh.player_events pe
     WHERE event_start >= '2022-01-01'
       AND event_end < '2022-07-01'
     GROUP BY player_id,
           customer_id,
           CASE
             WHEN event_type = 'Card Game' THEN TRUE
             ELSE FALSE
           END
    HAVING COUNT(*) > 10 )
, players AS (
    SELECT id,
                   name,
                   last_name,
                   gender,
           customer_id
      FROM dwh.players
     WHERE is_test IS FALSE
       AND COALESCE(age, 0) > 18 )
SELECT p.id AS player_id,
       p.name || p.last_name AS player_full_name,
       p.gender,
       pi.email,
       c.id AS customer_id,
       c.name AS customer_name
       pe.is_casino_game,
       pe.total_wins,
       DENSE_RANK() OVER (PARTITION BY is_casino_game ORDER BY total_wins DESC) AS rank_by_game_type
  FROM players p
  JOIN player_events pe
    ON p.player_id = pe.player_id
   AND p.customer_id = pe.customer_id
  JOIN dwh.customers c
    ON p.customer_id = c.id
  LEFT JOIN dwh.player_info pi
    ON p.id = pi.id
 ORDER BY pe.is_casino_game,
       pe.total_wins DESC,
       p.id ;
```

Rules:

- COLUMNS
    1. Each column new row

2. **Each column right under previous column**

```
SELECT p.id AS player_id,
       p.name || p.last_name AS player_full_name,
       c.id AS customer_id,
       c.name AS customer_name
       p.gender,
       pe.is_casino_game,
       pe.total_wins,
       DENSE_RANK() OVER (PARTITION BY is_casino_game ORDER BY total_wins DESC) AS
rank_by_game_type
```

- Aliases & Names

1. **Use table alias in every column if there is a join in the statement**

```
SELECT p.id AS player_id,
       p.name || p.last_name AS player_full_name,
       c.id AS customer_id,
       c.name AS customer_name
       p.gender,
       pe.is_casino_game,
       pe.total_wins,
       DENSE_RANK() OVER (PARTITION BY is_casino_game ORDER BY total_wins DESC) AS
rank_by_game_type
  FROM players p
  JOIN player_events pe
    ON p.player_id = pe.player_id
   AND p.customer_id = pe.customer_id
  JOIN dwh.customers c
    ON p.customer_id = c.id
```

2. **Use the "AS" operator when aliasing a column.**

```
p.name || p.last_name AS player_full_name,
```

3. **Don't use "AS" operator when aliasing a table.**

```
  FROM players p
  JOIN player_events pe
```

4. Use only "snake-case" convention ("player_id") not "camel-case" ("playerId") for naming
5. Table alias should represent first letters of the table name (players p, fact_wallet_transactions wt). If there are same aliases, use 2-4 letters short names (players p JOIN payments pmt).
   Predicates like "fact", "dimension", "datamart", ... are not used in alias

- JOIN

1. **Every join condition (ON / AND) on a new line**

```
  JOIN player_events pe
    ON p.player_id = pe.player_id
   AND p.customer_id = pe.customer_id
```

2. **ON condition always starts with main table (players in the example)**

```
  FROM players p
  JOIN player_events pe
    ON p.player_id = pe.player_id
   AND p.customer_id = pe.customer_id
  JOIN dwh.customers c
    ON p.customer_id = c.id
```

3. Use "JOIN" instead of ~~"INNER JOIN"~~

- GROUP BY

1. **Each column -> new line**

```
     GROUP BY player_id,
           customer_id,
           CASE
             WHEN event_type = 'Card Game' THEN TRUE
             ELSE FALSE
           END
```

2. Use only column names not numbers (~~GROUP BY 1,2,3~~)
3. Each column is under the previous one

- ORDER BY

1. **Each column -> new line**

```
 ORDER BY pe.is_casino_game,
       pe.total_wins DESC,
       p.id ;
```

2. Use only column names not numbers (~~ORDER BY 1,2,3~~)
3. Each column is under the previous one

- Functions

1. **Use one space delimiter between function parameters**

```
COALESCE(age, 0)
```

2. **CASE statement**

```
CASE
  WHEN event_type = 'Card Game' THEN TRUE
  ELSE FALSE
END AS is_casino_game

-- Long condition
CASE
  WHEN event_type = 'Card Game' AND event_time >= '2022-01-01 05:00:00' AND event_time < '2022-01-
01 08:00:00'
  THEN TRUE
  ELSE FALSE
END AS is_casino_game
```

- CTE & subqueries

1. **Each new CTE starts with ", "**

```
, players AS (
    SELECT *
      FROM dwh.players
     WHERE is_test IS FALSE
       AND COALESCE(age, 0) > 18 )
```

2. SELECT statement in CTE should have start in 4 spaces: "(space)(space)(space)(space)SELECT"
3. Naming of CTE tables should be clear for everyone. If there are multiple levels or aggregation/operations on the same table within few CTEs, they can be names like "wallet_transactions_lvl1", "wallet_transactions_lvl2", "wallet_transactions_lvl3", ...
4. CTE usage is preferable to subqueries
5. Both CTE & Subquery should have 1 space before SELECT and opening bracket and before closing bracket

- Spacing & Indents

Exceptions

1. **CASE statement inline can be used if it is simple / next columns have similar approach**

```
...
SUM(CASE WHEN account_change_type = 'commitBet'               THEN account_amount ELSE 0 END) AS
commit_bet,
SUM(CASE WHEN account_change_type = 'win'                     THEN account_amount ELSE 0 END) AS win,
SUM(CASE WHEN account_change_type = 'ticketCashOut'           THEN account_amount ELSE 0 END) AS
ticket_cash_out,
SUM(CASE WHEN account_change_type = 'jackpotWin'              THEN account_amount ELSE 0 END) AS
jackpot_win,
SUM(CASE WHEN account_change_type = 'revokeWin'               THEN account_amount ELSE 0 END) AS
revoke_win,
SUM(CASE WHEN account_change_type = 'deposit'                 THEN account_amount ELSE 0 END) AS
deposit,
...
```

2. **Subquery can be used only if there is a need to SELECT all columns but use some condition like (rn = 1):**

```
SELECT *
  FROM ( SELECT id,
                ROW_NUMBER() OVER (PARTITION BY type ODER BY created_at DESC) AS rn
           FROM payments )
 WHERE rn = 1 ;
```

Data Types

Use default data types and not aliases. Review the Snowflake summary of data types for more details. The defaults are:

- `NUMBER` instead of `DECIMAL`, `NUMERIC`, `INTEGER`, `BIGINT`, etc.
- `FLOAT` instead of `DOUBLE`, `REAL`, etc.
- `VARCHAR` instead of `STRING`, `TEXT`, etc.
- `TIMESTAMP` instead of `DATETIME`

Best practices

- Prefer `!=` to <>. This is because `!=` is more common in other programming languages and reads like "not equal" which is how we're more likely to speak.
- When making single line comments in a model use the `--` syntax
- When making multi-line comments in a model use the `/* */` syntax
- Time & Date columns should end with "_at" predicate ("created_at")
- Boolean columns should start with "is_" prefix ("is_test")
- No hardcoded IDs in objects (views, merges, tasks, ...)
- Always use column names instead of ~~SELECT * FROM~~

Uppercase / Lowercase?

Type	Case	Example
UPPERCASE	keywords	SELECT, UPDATE, JOIN ON, ORDER, LIMIT, AS, IN, ON, TASK, EXECUTE, SCHEDULE,...
	functions	SUM(), COALESCE(), COUNT(), ...
	booleans	TRUE, FALSE
	data types	STRING, VARCHAR, INT, FLOAT, ...
	constraints	NULL, NOT NULL, PRIMARY KEY()
LOWERCASE	object names	column names, table names, ...

Snowflake Objects

Task template

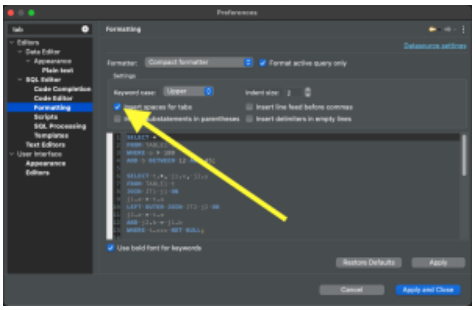
```
CREATE OR REPLACE TASK dwh.tsk_dm_liq_player_daily
AFTER dwh.tsk_dm_delete_liq_player_daily
AS
MERGE ...

CREATE OR REPLACE TASK dwh.tsk_dm_liq_player_daily
SCHEDULE = 'USING CRON 0 */6 * * * UTC'
AS
MERGE ...
```

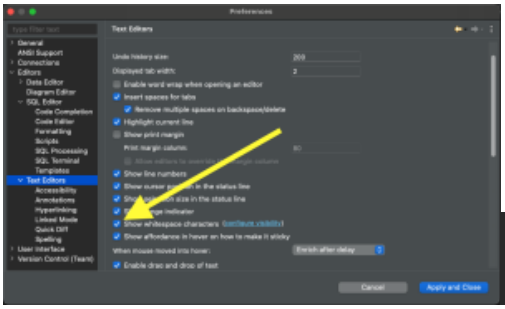
DBeaver approach

In DBeaver

- **TAB**. Switch to use spaces instead of tabs. Different code editors can use it's own tab indent and it can ruin your format.
  - Settings Editors SQL Editor Formatting "Insert spaces for tabs"



- **Whitespaces**. Showing whitespace characters makes it easier to manually format the code.
  - Settings Editors Text Editor "Show whitespace character"



```
SELECT·*
··FROM·player
·WHERE·name·=·'John';
```