Eric Rodriguez

BA FALL 2024

Beazley's Last Statement

1. A First SQL Query

- *Original MySQL Query:*
  SELECT * FROM executions LIMIT 3;
- *Updated T-SQL Query:*
  SELECT TOP 3 * FROM tx_deathrow;

2. The select block

- *Original MySQL Query:*
  SELECT first_name, last_name FROM executions LIMIT 3;
- *UpdatedT-SQL Query:*
  SELECT TOP 3 first_name, last_name
  FROM tx_deathrow;

3.The From Block

- *Original MySQL Query:*
  SELECT first_name FROM execution LIMIT 3
- *Updated T-SQL Query:*
  SELECT TOP 3 first_name
  FROM tx_deathrow;

4. Modify the query to divide 50 and 51 by 2.

- *Original MySQL Query:*
  SELECT 50 + 2, 51 * 2
- *Updated T-SQL Query:*
  SELECT 50 + 2, 51.0 * 2

5.The Where Block

- *Original MySQL Query:*
  /* SELECT first_name, last_name, ex_age
  FROM executions WHERE ex_age <= 25

- *Updated T-SQL Query:*
  SELECT first_name, last_name, Age_at_Execution
  FROM tx_deathrow
  WHERE Age_at_Execution <= 25;

6. **Modify the query to find the result for Raymond Landry.**

- *Original MySQL Query:*

```
SELECT first_name, last_name, ex_number
FROM executions
WHERE first_name = 'Raymond'
  AND last_name = 'Landry'
```

- *Updated T-SQL Query:*

```
SELECT [First_Name], [Last_Name], [TDCJ_Number]
FROM [tx_deathrow]
WHERE [First_Name] LIKE '%Raymond%'
  AND [Last_Name] LIKE '%Landry%';
```

7. **Insert a pair of parentheses so that this statement returns 0**

- *Original MySQL Query:*

```
SELECT 0 AND 0 OR 1
```

- *Updated T-SQL Query:*

```
SELECT IIF(0 = 1 AND (0 = 1 OR 1 = 1), 1, 0);
```

8. Find Napoleon Beazley's last statement

- *Original MySQL Query:*

```
/* SELECT last_statement
FROM executions
WHERE first_name = 'Napoleon'
  AND last_name = 'Beazley'
```

- *Updated T-SQL Query:*

```
SELECT last_statement
FROM tx_deathrow
WHERE First_Name = 'Napoleon'
  AND Last_Name = 'Beazley';
```

Claims of Innocence

1.The count Function

- *Original MySQL Query:*

SELECT COUNT(first_name) FROM executions

/* SELECT COUNT(last_statement) FROM executions

- *Updated T-SQL Query:*

SELECT COUNT(last_statement) FROM tx_deathrow;

2.Nulls, Verify that 0 and the empty string are not considered NULL.

- *Original MySQL Query:*

  SELECT (0 IS NOT NULL) AND ('' IS NOT NULL)

- *Updated T-SQL Query:*

  SELECT

  CASE WHEN (0 IS NOT NULL) AND ('' IS NOT NULL) THEN 1 ELSE 0 END

3. Find the total number of executions in the dataset.

- *Original MySQL Query:*
  /* SELECT COUNT(ex_number) FROM executions
- *Updated T-SQL Query:*

  SELECT COUNT(TDCJ_Number) FROM tx_deathrow;

4.Variations on Count

- *Original MySQL Query:*
  SELECT COUNT(*) FROM executions
- *Updated T-SQL Query:*

  SELECT COUNT(*) FROM tx_deathrow;

5.This query counts the number of Harris and Bexar county executions. Replace SUMs with COUNTs and edit the CASE WHEN blocks so the query still works.

- *Original MySQL Query:*
  SELECT
  SUM(CASE WHEN county='Harris' THEN 1

```
    ELSE 0 END),
  SUM(CASE WHEN county='Bexar' THEN 1
    ELSE 0 END)
FROM executions
```

- *Updated T-SQL Query:*

```
SELECT
  SUM(CASE WHEN county = 'Harris' THEN 1 ELSE 0 END) AS Harris_Count,
  SUM(CASE WHEN county = 'Bexar' THEN 1 ELSE 0 END) AS Bexar_Count
FROM tx_deathrow;
```

6.Practice Find how many inmates were over the age of 50 at execution time.

- *Original MySQL Query:*
  / SELECT COUNT(*) FROM executions WHERE ex_age > 50*

- *Updated T-SQL Query:*

```
SELECT COUNT(*)

FROM tx_deathrow

WHERE Age_at_Execution > 50;
```

7. **Find the number of inmates who have declined to give a last statement.**

- *Original MySQL Query:*

```
/* SELECT COUNT(*) FROM executions WHERE last_statement IS NULL

SELECT COUNT(CASE WHEN last_statement IS NULL THEN 1 ELSE NULL END) FROM
executions

SELECT COUNT(*) - COUNT(last_statement) FROM executions
```

- *Updated T-SQL Query:*

```
SELECT COUNT(*)

FROM tx_deathrow

WHERE last_statement IS NULL;
```

8. **Find the minimum, maximum and average age of inmates at the time of execution.**

- *Original MySQL Query:*

```
/* SELECT MIN(ex_age), MAX(ex_age), AVG(ex_age) FROM executions
```

- *Updated T-SQL Query:*

  SELECT

  MIN(Age_at_Execution) AS Min_Age,

  MAX(Age_at_Execution) AS Max_Age,

  AVG(Age_at_Execution) AS Avg_Age

  FROM tx_deathrow;

9. **Find the average length (based on character count) of last statements in the dataset.**

- *Original MySQL Query:*
  /* SELECT AVG(LENGTH(last_statement)) FROM executions
- *Updated T-SQL Query:*

  SELECT AVG(LEN(last_statement)) AS Avg_Statement_Length
  FROM tx_deathrow;

10. **List all the counties in the dataset without duplication.**

- *Original MySQL Query:*

/* SELECT DISTINCT county FROM executions

- *Updated T-SQL Query:*

SELECT DISTINCT county

FROM tx_deathrow;

11. A strange Query

- *Original MySQL Query:*

SELECT first_name, COUNT(*) FROM executions

- *Updated T-SQL Query:*

SELECT first_name, COUNT(*) AS name_count

FROM tx_deathrow

GROUP BY first_name;

12.Conclusion & Recap

- *Original MySQL Query:*

/* SELECT

1.0 * COUNT(CASE WHEN last_statement LIKE '%innocent%'

THEN 1 ELSE NULL END) / COUNT(*)

FROM executions /* SELECT

1.0 * COUNT(CASE WHEN last_statement LIKE '%innocent%'

THEN 1 ELSE NULL END) / COUNT(*)

FROM executions

- *Updated T-SQL Query:*

  SELECT

  1.0 * COUNT(CASE WHEN last_statement LIKE '%innocent%' THEN 1 ELSE NULL END) / COUNT(*) AS Innocent_Statement_Percentage

  FROM tx_deathrow;

The Long Tail

1.The Group by block

- *Original MySQL Query:*

```
SELECT
  county,
  COUNT(*) AS county_executions
FROM executions
GROUP BY county
```

- *Updated T-SQL Query:*

```
SELECT
   county,
   COUNT(*) AS county_executions
FROM tx_deathrow
GROUP BY county;
```

2. **This query counts the executions with and without last statements. Modify it to further break it down by count**

- *Original MySQL Query:*

SELECT

last_statement IS NOT NULL AS has_last_statement, COUNT(*)

FROM executions GROUP BY has_last_statement

- *Updated T-SQL Query:*

```
SELECT
   CASE WHEN last_statement IS NOT NULL THEN 1 ELSE 0 END AS has_last_statement,
   COUNT(*) AS statement_count
FROM tx_deathrow
GROUP BY CASE WHEN last_statement IS NOT NULL THEN 1 ELSE 0 END;
```

3.The HAVING Block

- *Original MySQL Query:*

```
/* SELECT county, COUNT(*)
FROM executions
WHERE ex_age >= 50
GROUP BY county
```

- *Updated T-SQL Query:*

```
SELECT
   county,
   COUNT(*) AS executions_count
FROM tx_deathrow
WHERE Age_at_Execution >= 50
GROUP BY county;
```

4,.List the counties in which more than 2 inmates aged 50 or older have been executed.

- *Original MySQL Query:*

```
/*
SELECT county
FROM executions
WHERE ex_age >= 50
GROUP BY county
HAVING COUNT(*) > 2
```

- *Updated T-SQL Query:*

```
SELECT
   county,
   COUNT(*) AS executions_count
FROM tx_deathrow
WHERE Age_at_Execution >= 50
GROUP BY county;
```

5.List all the distinct counties in the dataset.

Original MySQL Query:
/* SELECT county FROM executions GROUP BY county

- *Updated T-SQL Query:*

  SELECT county

  FROM tx_deathrow

  GROUP BY county;

6. Find the first and last name of the inmate with the longest last statement (by character count).

- *Original MySQL Query:*
  /* SELECT first_name, last_name
  FROM executions
  WHERE LENGTH(last_statement) =
  (SELECT MAX(LENGTH(last_statement))
  FROM executions)

- *Updated T-SQL Query:*

  SELECT first_name, last_name

  FROM tx_deathrow

  WHERE LEN(last_statement) =

  (SELECT MAX(LEN(last_statement))

  FROM tx_deathrow);

7,Insert the <count-of-all-rows> query to find the percentage of executions from each county

Original MySQL Query:

```
SELECT
  county,
  100.0 * COUNT(*) / (<count-of-all-rows>)
   AS percentage
FROM executions
GROUP BY county
ORDER BY percentage DESC
```

- *Updated T-SQL Query:*

```
SELECT
   county,
   100.0 * COUNT(*) / (SELECT COUNT(*) FROM tx_deathrow) AS percentage
FROM tx_deathrow
```

```
GROUP BY county
ORDER BY percentage DESC;
```

Execution Hiatuses

1.Dates

- *Original MySQL Query:*

```
SELECT JULIANDAY('1993-08-10') - JULIANDAY('1989-07-07
```

- *Updated T-SQL Query:*

```
SELECT DATEDIFF(DAY, '1989-07-07', '1993-08-10') AS days_difference;
```

2.Self Joins

- *Original MySQL Query*

```
/*
SELECT
  ex_number + 1 AS ex_number,
  ex_date AS last_ex_date
FROM executions
WHERE ex_number < 553
```

- *Updated T-SQL Query:*

```
SELECT
   TDCJ_Number + 1 AS ex_number,
   Execution_Date AS last_ex_date
FROM tx_deathrow
WHERE TDCJ_Number < 553;
```

3. Nest the query which generates the previous table into the template.

- *Original MySQL Query*

```
/*
SELECT
  last_ex_date AS start,
  ex_date AS end,
  JULIANDAY(ex_date) - JULIANDAY(last_ex_date) AS day_difference
FROM executions
JOIN (
   SELECT
     ex_number + 1 AS ex_number,
     ex_date AS last_ex_date
```

```
   FROM executions
  ) previous
  ON executions.ex_number = previous.ex_number
ORDER BY day_difference DESC
LIMIT 10
```

- *Updated T-SQL Query:*

```
SELECT TOP 10
   executions.Execution_Date AS start_date,
   previous.Execution_Date AS end_date,
   DATEDIFF(DAY, previous.Execution_Date, executions.Execution_Date) AS day_difference
FROM tx_deathrow executions
JOIN (
   SELECT
      TDCJ_Number + 1 AS TDCJ_Number,
      Execution_Date
   FROM tx_deathrow
) previous
ON executions.TDCJ_Number = previous.TDCJ_Number
ORDER BY day_difference DESC;
```

4. **Fill in the JOIN ON clause to complete a more elegant version of the previous query.**

- *Original MySQL Query*

```
/*
SELECT
  previous.ex_date AS start,
  executions.ex_date AS end,
  JULIANDAY(executions.ex_date) - JULIANDAY(previous.ex_date)
    AS day_difference
FROM executions
JOIN executions previous
  ON executions.ex_number = previous.ex_number + 1
ORDER BY day_difference DESC
LIMIT 10
```

- *Updated T-SQL Query:*

```
SELECT TOP 10
   previous.Execution_Date AS start_date,
   executions.Execution_Date AS end_date,
   DATEDIFF(DAY, previous.Execution_Date, executions.Execution_Date) AS
day_difference
FROM tx_deathrow executions
JOIN tx_deathrow previous
```

```
    ON executions.TDCJ_Number = previous.TDCJ_Number + 1
ORDER BY day_difference DESC;
```