Applied Databases
Data Cleaning

HIGHER DIPLOMA IN DATA ANALYTICS

Raw Data

Firstname	Surname	Sex	IP Address	Clicks
		M	186.23.43.14	1
John		M	192.34.22.111	12
Alan	Johnson		152.12.21.242	8
Ben			111.123.23.4	4
Mrs.	Brown	F	132.23.98.323	8

The Data Analysis Process…

- … is the process of inspecting, cleaning, transforming, and interpreting data to discover valuable insights, draw conclusions, and support decision-making.

Data Cleaning

- "Every year, poor data quality costs organizations an average $12.9 million."
- Data cleaning addresses a range of errors and issues in data sets, including inaccurate, invalid, incompatible and corrupt data.
- Some of those problems are caused by human error during the data entry process, while others are the result of different data structures, formats and terminology in separate systems throughout an organization.

Data Cleaning Process

- Get rid of unwanted observations.
- Fix structural errors.
- Standardise the data.
- Handle missing values.

location	date	maxTemp
WH1	2023-06-01	18
WH2	2023-06-01	66.2
WH3	2023-06-01	18.5
WH1	2023-06-02	22
WH2	2023-06-02	72
WH3	2023-06-02	22

Benefits of Data Cleaning

- Invalid Data

```
+-------+-----------------------------------------+------+-----+---------+-------+
| Field | Type                                    | Null | Key | Default | Extra |
+-------+-----------------------------------------+------+-----+---------+-------+
| role  | enum('Manager','Engineer','Technician') | YES  |     | NULL    |       |
```

eid	role
1	Manager
2	Engineer
3	Manager
4	Technician
5	Eng
6	Engineer
7	Tech

Benefits of Data Cleaning

- Missing Data

```
+-------+-----------------------------------------+------+-----+---------+-------+
| Field | Type                                    | Null | Key | Default | Extra |
+-------+-----------------------------------------+------+-----+---------+-------+
| role  | enum('Manager','Engineer','Technician') | YES  |     | NULL    |       |
```

eid	role
1	Manager
2	Engineer
3	Manager
4	Technician
5	
6	Engineer
7	Technician

Benefits of Data Cleaning

- Inconsistent Data

```
+-----------+------+------+-----+---------+-------+
| Field     | Type | Null | Key | Default | Extra |
+-----------+------+------+-----+---------+-------+
| hire_date | date | YES  |     | NULL    |       |
```

eid	hire_date	
1	2024-01-05	
2	01/01/1998	✖
3	2010/11/04	
4	230914	
5	2005-11-17	
6	2021-12-11	
7	Nov 21st 2018	✖

Benefits of Data Cleaning

- Duplicate Data

eid	role
1	Manager
2	Engineer
6	Engineer
7	Technician

eid	role
2	Engineer
3	Manager
4	Technician
5	Engineer

orderNum	Pizza	Address
10045	Pepperoni Small	1 New St, Tuam
10343	2 Pepperoni Large	14 Main St. Tuam
13032	Margherita Medium	12 Pearce St, Ballinasloe
13259	1 Pepperoni Large 1 Margherita Large	Ivy Cottage, 14 Main St. Tuam

Benefits of Data Cleaning

- Irrelevant Data

eid	role	salary	status
1	Manager	200,134	permanent
2	Engineer	58,234	permanent
3	Manager	70,848	permanent
4	Technician	100,014	contractor
5	Engineer	60,003	permanent
6	Engineer	55,772	permanent
7	Technician	61,983	permanent

Benefits of Data Cleaning

- Indicator Variables
  - Have a value of 0 or 1
  - Used to indicate whether a given observation belongs to a discrete category
  - Useful in Data Analysis.

ID	Name	Age	Sex	Highest Education Level	County	isMale	fromLeinster	in30sOr40s
100	Ann	32	F	8	D	0	1	1
107	Thomas	23	M	8	WH	1	1	0
108	John	49	M	6	G	1	0	1
110	Chloe	18	F	5	C	0	0	0
115	Bernie	61	M	7	C	1	0	0

Benefits of Data Cleaning

- Binning
  - A way to group a number of continuous values into a smaller number of "bins".
  - Makes data more manageable, and easier to analyze.

ID	Name	Age	Sex	Salary	County	AgeRange	SalaryRange
100	Ann	32	F	23256.25	D	20-39	<50k
107	Thomas	23	M	28362.20	WH	20-39	<50k
108	John	49	M	61200.00	G	>40	50k-70k
110	Chloe	18	F	9535.11	C	<18	<50k
115	Bernie	61	M	77328.84	C	>40	70k-90k

Loading Data into MySQL

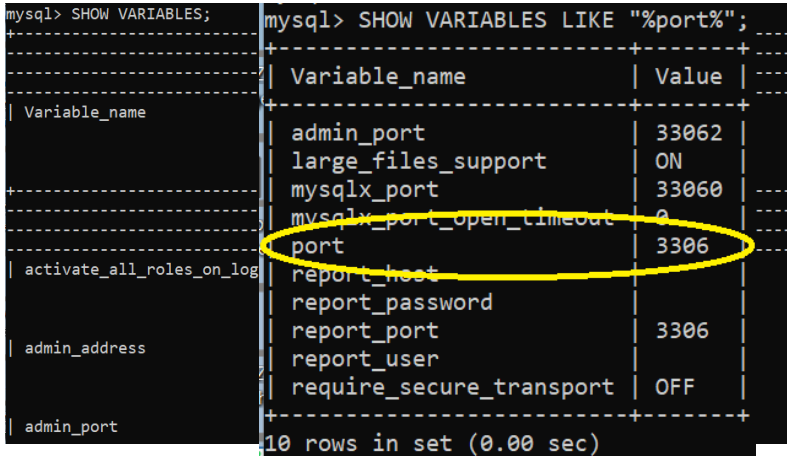
- LOAD
  - `"\Program Files\MySQL\MySQL Server 8.0\bin\mysql.exe" --local-infile=1 -u root -proot`
  - `SET GLOBAL local_infile=1;`
  - `secure-file-priv`

Cleaning Data

- Built-in Functions
  - String Functions
  - Numeric Functions
  - Date & Time Functions
  - Aggregate Functions
  - Control Flow Functions
- Stored Routines
- Stored Procedures
- User-Defined Variables

MySQL Variables

- 3 Types of Variable in MySQL:
  - System



  - Local
  - User-defined

User-Defined Variables

- Store a value in a user-defined variable in one statement and refer to it later in another statement.
- This enables you to pass values from one statement to another.

```
mysql> SELECT name, age,
    ->   CASE
    ->     WHEN age < (SELECT AVG(age) FROM people) THEN "<"
    ->     WHEN age = (SELECT AVG(age) FROM people) THEN "="
    ->     ELSE ">"
    ->   END AS diff
    -> FROM people;
+--------+------+------+
| name   | age  | diff |
+--------+------+------+
| Ann    |   32 | <    |
| Thomas |   23 | <    |
| John   |   49 | >    |
| Chloe  |   18 | <    |
| Bernie |   61 | >    |
+--------+------+------+
5 rows in set (0.00 sec)
```

```
mysql> SET @avg_age := (SELECT AVG(age) FROM people);
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> SELECT name, age,
    ->   CASE
    ->     WHEN age < @avg_age THEN "<"
    ->     WHEN age = @avg_age THEN "="
    ->     ELSE ">"
    ->   END AS diff
    -> FROM people;
+--------+------+------+
| name   | age  | diff |
+--------+------+------+
| Ann    |   32 | <    |
| Thomas |   23 | <    |
| John   |   49 | >    |
| Chloe  |   18 | <    |
| Bernie |   61 | >    |
+--------+------+------+
5 rows in set (0.00 sec)
```

```
mysql> SELECT @avg_age;
+--------------+
| @avg_age     |
+--------------+
| 36.600000000 |
+--------------+
1 row in set (0.00 sec)
```

name	age
Ann	32
Thomas	23
John	49
Chloe	18
Bernie	61

User-Defined Variables

```
mysql> SET @p_name := "John";
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @p_name;
+---------+
| @p_name |
+---------+
| John    |
+---------+
1 row in set (0.00 sec)
```

User-Defined Variables

- Can only have 1 value.

```
mysql> SELECT * FROM people;
+--------+------+
| name   | age  |
+--------+------+
| Ann    |   32 |
| Thomas |   23 |
| John   |   49 |
| Chloe  |   18 |
| Bernie |   61 |
+--------+------+
5 rows in set (0.00 sec)
```

```
mysql> SET @age := (SELECT age FROM people);
ERROR 1242 (21000): Subquery returns more than 1 row
```

```
mysql> select @age;
+------+
| @age |
+------+
| NULL |
+------+
1 row in set (0.00 sec)
```

User-Defined Variables

- Are only valid in the current session.
- Cannot be deleted by the user.
- Removed when the session ends.

```
mysql> SELECT * FROM performance_schema.user_variables_by_thread;
+-----------+---------------+--------------------------------+
| THREAD_ID | VARIABLE_NAME | VARIABLE_VALUE                 |
+-----------+---------------+--------------------------------+
|        54 | var_a         | 0x41                           |
+-----------+---------------+--------------------------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT @var_a;
+--------+
| @var_a |
+--------+
| A      |
+--------+
1 row in set (0.00 sec)
```

User-Defined Variables

ID	Name
100	Ann
107	Thomas
108	John
110	Chloe
115	Bernie

```
+--------+--------+--------+
| name   | Length | Length |
+--------+--------+--------+
| Ann    |      3 | Short  |
| Thomas |      6 | Long   |
| John   |      4 | Medium |
| Chloe  |      5 | Medium |
| Bernie |      6 | Long   |
+--------+--------+--------+
5 rows in set (0.00 sec)
```

```
SELECT name,
       CHAR_LENGTH(name) AS Length,
       CASE
           WHEN CHAR_LENGTH(name) < 3 THEN "Tiny"
           WHEN CHAR_LENGTH(name) < 4 THEN "Short"
           WHEN CHAR_LENGTH(name) < 6 THEN "Medium"
           ELSE "Long"
       END AS Length
FROM people;
```

```
SELECT name,
       @name_len := CHAR_LENGTH(name) AS Length,
       CASE
           WHEN @name_len < 3 THEN "Tiny"
           WHEN @name_len < 4 THEN "Short"
           WHEN @name_len < 6 THEN "Medium"
           ELSE "Long"
       END AS Length
FROM people;
```

Transposing Data

- Converting columns to rows, or rows to columns.
- Often for improved readability and data analysis.

Id	Name	Salary	Commission
1	John Ford	55000.00	2500.00
2	Anne Mulhern	34500.00	15500.00
3	Paddy O'Meara	NULL	76520.10
4	Mary Collins	58850.00	NULL

Id	Name	Type	Amount
1	John Ford	Salary	55000.00
1	John Ford	Commission	2500.00
2	Anne Mulhern	Salary	34500.00
2	Anne Mulhern	Commission	15500.00
3	Paddy O'Meara	Commission	76520.10
4	Mary Collins	Salary	58850.00

UNION

- The UNION operator is used to combine the result-set of two or more SELECT statements.
- Every SELECT statement within UNION must have the same number of columns.

UNION

Cid	Name	County
1	Jones Hardware	G
2	Murphy Contractors	G
3	Fitzpatrick Ltd	MO
4	Hanly Fabrication	WH

Sid	Name	County	Total
100	Smyth Doors	MO	25000
101	Westmeath Windows	WH	18700
102	Ballina Fabrication	MO	12800
103	Dublin Tiles	D	3900

```
mysql> SELECT county FROM customers;
+--------+
| county |
+--------+
| G      |
| G      |
| MO     |
| WH     |
+--------+
4 rows in set (0.00 sec)
```

```
mysql> SELECT county FROM customers
    -> UNION
    -> SELECT county FROM suppliers;
+--------+
| county |
+--------+
| G      |
| MO     |
| WH     |
| D      |
+--------+
4 rows in set (0.00 sec)
```

```
mysql> SELECT county FROM suppliers;
+--------+
| county |
+--------+
| MO     |
| WH     |
| MO     |
| D      |
+--------+
4 rows in set (0.00 sec)
```

Views

- A view is a virtual table based on the result-set of an SQL statement.
- A view contains rows and columns.
- The fields in a view are fields from one or more real tables in the database.
- A view always shows up-to-date data.

```
mysql> CREATE VIEW county_view AS
    -> SELECT county FROM customers
    -> UNION
    -> SELECT county FROM suppliers;
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> SELECT * FROM county_view;
+--------+
| county |
+--------+
| G      |
| MO     |
| WH     |
| D      |
+--------+
4 rows in set (0.00 sec)
```

View Management

- SHOW TABLES;
- DROP VIEW *view_name;*

```
mysql> SELECT TABLE_SCHEMA, TABLE_NAME, TABLE_TYPE
    -> FROM information_schema.tables
    -> WHERE TABLE_NAME = "county_view";
+--------------+-------------+------------+
| TABLE_SCHEMA | TABLE_NAME  | TABLE_TYPE |
+--------------+-------------+------------+
| appdbtest1   | county_view | VIEW       |
+--------------+-------------+------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT TABLE_SCHEMA, TABLE_NAME, VIEW_DEFINITION
    -> FROM information_schema.views
    -> WHERE TABLE_NAME = "county_view";
+--------------+-------------+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| TABLE_SCHEMA | TABLE_NAME  | VIEW_DEFINITION                                                                                                                                                               |
+--------------+-------------+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| appdbtest1   | county_view | select `appdbtest1`.`customers`.`county` AS `county` from `appdbtest1`.`customers` union select `appdbtest1`.`suppliers`.`county` AS `county` from `appdbtest1`.`suppliers` |
+--------------+-------------+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

Views

- Combine information from multiple tables.
- Focus, simplify and customise each user's view of the database.
- Allow users access data only through the view.

Id	Name	Role	Salary
1	John Ford	Engineer	55000.00
2	Anne Mulhern	Engineer	51500.20
3	Paddy O'Meara	Technician	48700.20
4	Mary Collins	Clerical	42460.99

```
mysql> CREATE VIEW employee_view AS
    -> SELECT id, name, role FROM employee_info;
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> SELECT * FROM employee_view;
+----+---------------+------------+
| id | name          | role       |
+----+---------------+------------+
|  1 | John Ford     | Engineer   |
|  2 | Anne Mulhern  | Engineer   |
|  3 | Paddy O'Meara | Technician |
|  4 | Mary Collins  | Clerical   |
+----+---------------+------------+
4 rows in set (0.00 sec)
```

Transposing Data

- Transpose rows to columns to see different summaries of the source data.

tid	Eid	Name	Type	Value
1	100	John Ford	Sell	252.35
2	100	John Ford	Sell	233.14
3	100	John Ford	Buy	133.14
4	101	Anne Mulhern	Buy	572.00
5	101	Anne Mulhern	Buy	32.00
6	102	Paddy O'Meara	Sell	45.00
7	102	Paddy O'Meara	Sell	168.00

Eid	Sell Total	Buy Total
100	485.49	133.14
101	NULL	604.00
102	213.00	NULL

```
mysql> SELECT eid,
    ->        SUM(CASE WHEN type = "Sell" THEN value END) AS "Sell Total",
    ->        SUM(CASE WHEN type = "Buy" THEN value END) AS "Buy Total"
    -> FROM stocks
    -> GROUP BY eid;
+-----+------------+-----------+
| eid | Sell Total | Buy Total |
+-----+------------+-----------+
| 100 |     485.49 |    133.14 |
| 101 |       NULL |    604.00 |
| 102 |     213.00 |      NULL |
+-----+------------+-----------+
3 rows in set (0.00 sec)
```