DMV 13/02/25

Exam Date	@February 13, 2025
↗ Teaching	Data Management and Visualization

Exam (13/02/25)

1. Theory 1

Which of the following is a characteristic of synchronous replication in distributed databases?

1. **The master waits for all or a subset of slaves to commit before committing a transaction.**
2. The master commits locally without waiting for slaves.
3. It is faster and more reliable than asynchronous replication.
4. Slaves independently fetch updates from the master.
5. none of the other answers is correct.

---

2. Theory 2

Polyglot persistence refers to:

1. **Using different data stores in different circumstances.**
2. Using only NoSQL databases for all data storage needs.
3. Implementing standard RDBMS features in NoSQL databases.
4. Adopting NoSQL principles in RDBMS.
5. Using different languages to query the same database.
6. none of the other answers is correct.

---

3. Theory 3

Which one of the following guidelines is typically recommended for effective color usage in data visualization?

1. **Apply colors consistently across charts and dashboards to maintain clarity.**
2. Use every color of the rainbow to ensure each element stands out.
3. Randomly mix bright, contrasting colors to keep viewers' attention.
4. Select color palettes that clash significantly to emphasize differences.
5. Avoid color usage altogether and rely strictly on shape variations.

---

4. Conceptual design - Medical Clinic Analytics

A medical clinic needs to develop a data warehouse to analyse exam outcomes and patient health trends. The system will track various types of medical examinations, diseases, and patient information. You are required to design a data warehouse to analyse exam results and patient metrics according to the following specifications.

- For each exam the system tracks if the outcome is within the standard values or not.
- Exams are associated with **patient** classes. Patients are categorized by age groups (0-18, 19-40, 41-65, >65), gender, and insurance type (private, public, or none). For privacy compliance, personal identifying information is not stored in the data warehouse.
- Each exam belongs to a specific **exam type**. Exam types belong to one of the following categories: laboratory tests, imaging diagnostics, or clinical assessments. Each exam type may or may not require fast analysis, and may or may not need special preparation.
- Each exam is linked to one or more **diseases**. Diseases are organized in a hierarchical classification system, with main categories (e.g., cardiovascular, respiratory, and neurological) and subcategories. Each disease can be

associated with many different symptoms and many risk factors, that are tracked in the system. Both the list of symptoms and risk factors can grow over time.

- Each exam is performed at a specific **time**. The system tracks multiple time granularities: month, quarter, and year. The hour of day and if the exam hour is among the emergency hours are also tracked.

Write the textual formalism to describe the described conceptual schema.

▼ **Solution**

```
flowchart LR
F[[**EXAMS**
ExamOutcome
ExamCost
]]

F --- IndicatorInterpretation --- NO!

F --- PatientClass --- AgeGroup
PatientClass --- Gender
PatientClass --- InsuranceType

F --- ExamType --- ExamCategory
ExamType --- needsSpecialPreparation
ExamType --- requiresFastAnalysis

F === Disease --- DiseaseSubCategory --- DiseaseCategory
Disease === Symptoms
Disease === RiskFactors

F --- Month --- Quarter --- Year
F--- hourOfDay--- isEmergencyHour
```

5. Logical Schema

Given the following conceptual schema:

```
flowchart LR
F[[**VOTES**
number_of_votes
]]

F --- VoterJunk --- AgeGroup
VoterJunk --- Gender
VoterJunk --- ResidenceStatus

F --- Municipality --- Subregion --- Region --- Country
Subregion --- EUelectoralDistrict --- Country

F --- PoliticalParty --- EUGroup
PoliticalParty === PoliticalIssues

F --- Month --- Quarter --- Year
Quarter --- ElectionCycle
```

- Age group can be either “<3”, “3-12”, “12-18”, “18-65”, or “>65”
- The number of political issues to be tracked is not known in advance and the list of political issues to be tracked can grow over time.

Provide the logical design of the conceptual DW schema indicated in the picture.

Write each table on a new line.
Use
bold or underlining to identify primary-key attributes.

> Example: `TABLE_NAME(PrimaryKey, Attribute1, Attribute2)`

▼ **Solution**

```
Votes(ElectionJunk, VoterJunk, TimeID, LocationID, PoliticalPartyID, number_
VoterJunk(VoterJunk, AgeGroup, Gender, ResidenceStatus)
Location(LocationID, Municipality, Subregion, Region, Country, EUelectoralDi
PoliticalParty(PoliticalPartyID, EUGroup, PoliticalIssuesID)
Issues(PoliticalIssuesID, PoliticalIssues)
Time(TimeID, Month, Quarter, Year, ElectionCycle)
```

---

6. DW Query 1

```
Tickets(TimeID, CustomerConfID, AttractionID, TicketCount, TotalRevenues)
Time(TimeID, Date, isHoliday, isWeekDay, Month, 4m, 6m, Year)
CustomerConf(CostumerConfID, AgeGroup, LocationID)
Attraction(AttractionID, needsExtraCost, LocationID, ThemedSection, Amusemen
Location(LocationID, Region, Country)
```

For each Attraction **region**:

- select the monthly revenues
- the cumulative monthly revenues since the beginning of the year.
- The percentage of monthly revenues with respect to all Attractions in the same country.

Write the corresponding SQL query.

▼ **Solution**

```
SELECT Month, Region, Country,
  SUM(TotalRevenues) AS A,
    SUM(SUM(TotalRevenues)) OVER (PARTITION BY Year, Region ORDER BY
                                  ROWS UNBOUNDED PRECEDING) AS B,
    100 * SUM(TotalRevenues) / SUM(SUM(TotalRevenues)) OVER (PARTITION
```

```
FROM TICKETS F, TIME T, Location L
WHERE F.TimeID=T.TimeID and F.LocationID = L.LocationID
GROUP BY Month, Year, Region, Country;
```

---

7. DW Query 2

```
Tickets(TimeID, CustomerConfID, AttractionID, TicketCount, TotalRevenues)
Time(TimeID, Date, isHoliday, isWeekDay, Month, 4m, 6m, Year)
CstumerConf(CustomerConfID, AgeGroup, LocationID)
Attraction(AttractionID, needsExtraCost, LocationID, ThemedSection, Amusemen
Location(LocationID, Region, Country)
```

Consider the year 2023. Separately for each **amusement park** and **month**, analyse the:

- average income per ticket sold
- the monthly number of tickets with respect to the monthly number of tickets sold in all amusement parks
- Rank the amusement parks, separately for each month, based on their total number of tickets sold (rank 1st the highest).

Write the corresponding SQL query.

▼ **Solution**

```
SELECT AmusementPark, Month,
  SUM(TotalRevenues)/SUM(TicketCount) as A,
  SUM(TotalRevenues)/SUM(SUM(TicketCount)) OVER (PARTITION BY Month
  RANK() OVER (PARTITION BY month ORDER BY SUM(TicketCount) DESC) A

FROM TICKETS F, Time T, Attraction A
```

```
WHERE F.TimeID=T.TimeID and F.AttractionID = A.AttractionID
AND T.Year=2023
GROUP BY AmusementPark, Month, Year
```

---

8. NoSQL Design

You are required to design a MongoDB database that stores data about researchers who publish papers at conferences.

Each conference has a name (e.g., “International Conference on Big Data”), an organizer or host institution (e.g., “ACM”, “IEEE”), a location (e.g., “Rome, Italy”), a date range (e.g., 2024-06-10 to 2024-06-13), and a set of key topics (e.g., “machine learning”, “data analytics”, “cloud computing”, etc.).

A conference receives many papers from different researchers and accepts some of them for publication.

Each researcher is characterized by first name, last name, some email addresses, and its affiliation (e.g., “Politecnico di Torino”). Researchers write papers and submit them to conferences to get them published.

A paper is authored by one or more researchers, has a title, a set of keywords, the latest submission date, the latest status (e.g., either “submitted”, or “accepted”, or “rejected”).

Each paper is associated with exactly one conference, the last one the paper has been submitted to. Multiple contemporary submissions of the same paper to different conferences are not allowed.A researcher can have multiple papers in different conferences at the same time, whereas a conference can have many accepted papers.Typically most researchers have less than 100 papers. Only very few researchers have a very large number of papers (such as 10k, which exceeds the MongoDB max document size).

You are required to efficiently retrieve all papers authored by a specific researcher, together with the paper title, conference name, latest submission date, and latest status.

For each researcher, efficiently compute the average acceptance rate of their papers (e.g., out of X papers, Y were accepted).Given a specific conference,

efficiently provide the average acceptance rate of all submitted papers.

Indicate the collections you would use.

Write a sample document for each collection.

Important: In addition to the sample documents, explicitly state the design patterns used.

▼ **Solution**

```
conferences
{
  "_id": "ICBD2024",
  "name": "International Conference on Big Data",
  "organizer": "ACM",
  "location": "Rome, Italy",
  "start_date": "2024-06-10",
  "end_date": "2024-06-13",
  "topics": ["machine learning", "data analytics", "cloud computing"],

  // COMPUTED PATTERN
  "papers_submitted_count": 150,  // updated as new papers are submitted
  "papers_accepted_count": 60      // updated when papers are accepted
}

researchers
{
  "_id": "R123456",
  "first_name": "Alice",
  "last_name": "Johnson",
  "affiliation": "Politecnico di Torino",
  "emails": [
    "alice.johnson@polito.it",
    "alice_j@yahoo.com"
  ],
```

```
  "papers": [       // EXTENDED REFERENCE
    {
      "_id": "P9999",
      "title": "Big Data Analytics in Healthcare",
      "conference_id": "ICBD2024",
      "status": "submitted",
      "latest_submission_date": "2023-12-01"
    },
    ...
  ],
  "extra_papers": true,   // OUTLIER PATTERN

  // counters to compute acceptance ratio
  "papers_submitted_count": 10,   // COMPUTED PATTERN
  "papers_accepted_count": 6   // COMPUTED PATTERN
}

papers
{
  "_id": "P9999",
  "title": "Big Data Analytics in Healthcare",
  "keywords": ["big data", "analytics", "healthcare"],
  "latest_submission_date": "2023-12-01",
  "latest_status": "submitted",   // “accepted" or "rejected"

  // The last conference to which this paper was submitted
  "conference_id": "ICBD2024", // reference to conferences._id

  // List of authors
  "authors": [
    {
      "_id": "R123456",
      "first_name": "Alice",
      "last_name": "Johnson"
    },
    {
```

```
      "_id": "R789012",
      "first_name": "Bob",
      "last_name": "Smith"
    }
   ]
  }
```

---

9. NoSQL Query 1

Suppose you have a collection called buildings. Below is a sample document illustrating the structure and describing a building, its properties in terms of size, consumption, managers, and maintenance activities.

```
{
  "building_name": "PoliTo Tower",
  "floors": ["Office", "Laboratory", "Classroom"],
  "size_sqm": 20000,
  "consumption_records": [
    {
      "type": "electricity",
      "year": 2024,
      "consumption_kwh": 35000,
      "peak_usage_hours": 60
    },
    {
      "type": "water",
      "year": 2024,
      "consumption_liters": 70000,
      "peak_usage_hours": 20
    }
  ],
  "managers": [
    {
      "id": "m12345",
```

```
      "name": "Alice",
      "surname": "Green",
      "role": "agent"
    },
    {
      "id": "m67890",
      "name": "Carl",
      "surname": "White",
      "role": "administrator"
    }
  ],
  "maintenance_schedule": [
    {
      "date": "2024-09-16",
      "type": "regular"
    },
    {
      "date": "2025-02-11",
      "type": "emergency"
    }
  ]
}
```

Find all buildings having at least one manager whose role is “administrator”, having at least an “office” floor, and having at least one electricity consumption record in 2024 with consumption_kwh >= 30000. Show only the building name and the list of managers’ surnames in the output.

▼ **Solution**

```
db.buildings.find(
  {
    floors: "Office",
    "managers.role": "administrator",
```

```
    consumption_records: {
      $elemMatch: {
        type: "electricity",
        year: 2024,
        consumption_kwh: { $gte: 30000 }
      }
    }
  },
  {
    _id: 0,
    building_name: 1,
    “managers.surname”: 1
  }
);
```

10. NoSQL Query 1

Suppose you have a collection called buildings. Below is a sample document illustrating the structure and describing a building, its properties in terms of size, consumption, managers, and maintenance activities.

```
{
 "building_name": "PoliTo Tower",
 "floors": ["Office", "Laboratory", "Classroom"],
 "size_sqm": 20000,
 "consumption_records": [
  {
   "type": "electricity",
   "year": 2024,
   "consumption_kwh": 35000,
   "peak_usage_hours": 60
  },
  {
   "type": "water",
```

```
      "year": 2024,
      "consumption_liters": 70000,
      "peak_usage_hours": 20
    }
  ],
  "managers": [
    {
      "id": "m12345",
      "name": "Alice",
      "surname": "Green",
      "role": "agent"
    },
    {
      "id": "m67890",
      "name": "Carl",
      "surname": "White",
      "role": "administrator"
    }
  ],
  "maintenance_schedule": [
    {
      "date": "2024-09-16",
      "type": "regular"
    },
    {
      "date": "2025-02-11",
      "type": "emergency"
    }
  ]
}
```


Considering only buildings having at least one emergency maintenance event in 2024, separately for each year and type of the utility records, calculate the total number of peak hours. Sort the results in descending order of total peak hours.

▼ **Solution**

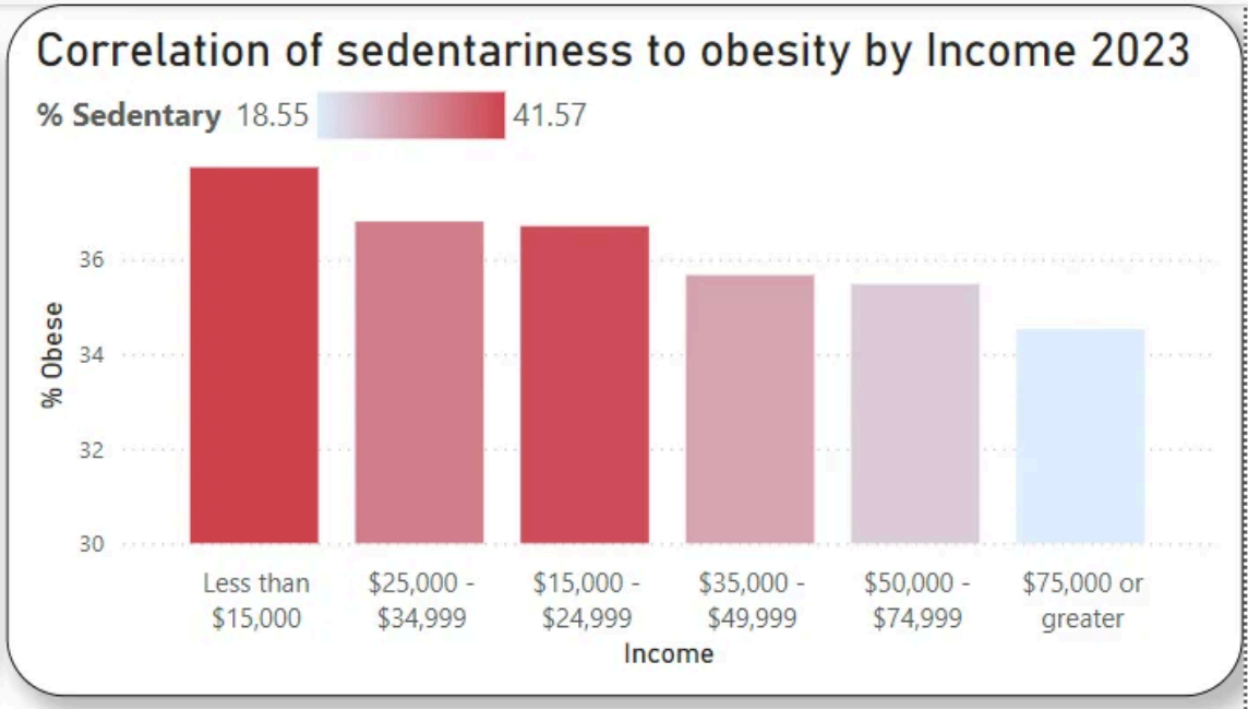
```
db.buildings.aggregate([
  // buildings having at least one "emergency" maintenance event in 2024
  {
    $match: {
      maintenance_schedule: {
        $elemMatch: {
          type: "emergency",
          date: { $gte: "2024-01-01", $lte: "2024-12-31" }
        }
      }
    }
  },

  {
    $unwind: "$consumption_records"
  },

  // Group by (year, type) to sum up total peak usage hours
  {
    $group: {
      _id: {
        year: "$consumption_records.year",
        type: "$consumption_records.type"
      },
      totalPeakHours: { $sum: "$consumption_records.peak_usage_hours" }
    }
  },

  {
    $sort: { totalPeakHours: -1 }
  }
]);
```

11. Data Visualization



Question

Which one of the following questions represents the purpose of this visualization?

1. **How do obesity rates vary by income bracket in 2023, and how do those rates relate to sedentary behavior?**
2. Which region in the country has the highest number of obesity cases, and how does that compare to other regions?
3. At which age group does obesity peak, and is that tied to higher levels of sedentariness?
4. What are the obesity and sedentary rates for men compared to women in 2023?
5. How does mental health correlate with obesity among various income brackets?

Data

Is the data quality appropriate? Select true answers only.

1. **The chart indicates the data is from 2023, which supports its currency.**
2. All individuals within each income bracket were surveyed, ensuring complete coverage for each bracket.
3. **The data provides decimal percentages for sedentariness, indicating a relatively high level of precision for cross-bracket comparison.**
4. Because the data comes from a local grocery store survey, its credibility for national trends is questionable.
5. The bar chart categorizes obesity rates in exact 1% increments, demonstrating high accuracy.
6. **Viewers can clearly match each bar to its income bracket label, indicating understandability.**
7. Because the sedentariness percentages range from 18.55% to 41.57%, the chart lacks clarity about how these values were derived, indicating low precision.
8. Including depression data in this visualization makes it more complete, since it links mental health to obesity.
9. **Each income bracket uses the same definitions and time frame for obesity and sedentariness, which enhances consistency.**
10. The dataset was compiled by a recognized health organization, which guarantees its credibility.

Visual Proportionality

Are the values encoded in a uniformly proportional way?

> No, they are not completely proportional. The bars for obesity rates do not start at zero on the Y-axis, which can distort how differences appear. Meanwhile, sedentariness relies on color encoding: while it may be proportional in theory, color

> gradations are harder to read precisely, making accurate comparisons more challenging.

Visual Utility

All the elements in the graph convey useful information?

> Most of the graph's main elements (such as the bars, the color shading, and the income labels) do convey relevant information about obesity, sedentariness, and income. However, some graphical details (e.g., background shading or the border) are more decorative than strictly informative.

Design data

Design the visualization based on the following data structure.

INCOME_BRACKET: Dimension
SEDENTARY_RATE:
Measure
OBESITY_RATE:
Measure
YEAR:
Dimension

Design schema & Sketch

Fill in the required schema elements; formulas can be used if required. Then describe in words the design proposal.

> Design proposal: This redesigned slope chart addresses the earlier issues by representing each income bracket with a line connecting its obesity rate and sedentary rate, thus enabling clear one-to-one comparisons without relying on a non-zero starting bar or complicated color gradients. Each bracket is represented by two points, one for obesity and one for

> sedentariness, with the slope of the line indicating how much higher or lower one measure is relative to the other. Income brackets are ordered logically or numerically to simplify reading, and each bracket is color-coded to distinguish its line from others, removing ambiguity. Numeric labels can appear on each point to clarify exact rates, while any additional decorative elements (such as borders or background shades) can be minimized to keep the focus on the slope lines themselves.

Schema	Details
Columns	MEASURE_NAMES
Rows	MEASURE_VALUES
Graph type	Line
Color	INCOME_BRACKET
Size	Default
Label	MEASURE_VALUES