Data Warehouse — Final Exam Crash Guide

50 Marks | 5 Questions | Some split into (a) and (b)

HOW TO USE THIS DOC (read this first — 2 min)

You have 5 hours. Priority order:

1. **Data Warehouse basics + Schemas** (1 hr) — easy marks, always asked
2. **ETL Process** (30 min) — easy, definition + steps + diagram
3. **OLAP + operations** (45 min) — easy, mostly definitions
4. **K-Means, Apriori, FP-Growth** (1.5 hrs) — needs numerical understanding, highest effort
5. **Two scenario questions** (45 min) — memorize the answer structure below, don't just read it
6. **Data Visualization** (20 min) — light, mostly common sense

Write answers in **definition → diagram → example → advantages/disadvantages** format wherever possible. Examiners give marks for structure, not just content.

---

1. DATA WAREHOUSE CONCEPTS

Definition

A Data Warehouse (DW) is a **subject-oriented, integrated, time-variant, and non-volatile** collection of data used to support management decision-making. (This is Bill Inmon's definition — **memorize it word-for-word**, it's a guaranteed 2-mark question.)

The 4 Characteristics (explain each in 1-2 lines)

- **Subject-Oriented**: Organized around major subjects (sales, customer, product) rather than day-to-day operations.
- **Integrated**: Data pulled from multiple heterogeneous sources (databases, flat files, etc.) and made consistent (naming, formats, units).
- **Time-Variant**: Data is stored with a time dimension — you can see historical trends over months/years, unlike operational DBs which show current data only.
- **Non-Volatile**: Once data enters the warehouse, it is not updated or deleted — only appended. Stable for analysis.

DW vs Database (OLTP) — draw this table, it's a common Q

Feature	OLTP (Operational DB)	Data Warehouse (OLAP)
Purpose	Day-to-day transactions	Analysis & decision making
Data	Current, detailed	Historical, summarized

Feature	OLTP (Operational DB)	Data Warehouse (OLAP)
Operations	Insert/Update/Delete	Read/Query (mostly SELECT)
Design	Normalized (ER model)	Denormalized (Star/Snowflake)
Users	Clerks, operators	Managers, analysts
Size	Smaller, gigabytes	Larger, terabytes

The 5 Goals of a Data Warehouse (from your slides — likely a direct question)

1. Makes an organization's information **accessible**
2. Makes the organization's information **consistent**
3. Is an **adaptive and resilient** source of information
4. Is a **secure bastion** that protects our information asset
5. Is the **foundation for decision making**

Basic Elements of a Data Warehouse (Kimball model — YOUR SIR'S EXACT TERMINOLOGY, high-yield)

This is the architecture your lecture (Lecture 5, based on Kimball's "Data Warehouse Lifecycle Toolkit") actually uses. Learn these terms exactly — a generic "3-tier architecture" answer will lose marks if the question asks for these specific components.

```
Source System → Data Staging Area → Presentation Server → End User Data Access
                                                |
                                    (Dimensional Model:
                                     Fact + Dimension tables,
                                     organized into Data Marts)
```

1. **Source System**: An operational system of record whose function is to capture business transactions. Queries against it are narrow, account-based, and part of normal transaction flow — severely restricted. Maintains little historical data.
2. **Data Staging Area**: A storage area and set of processes that **clean, transform, combine, deduplicate, household, and archive** source data before it enters the warehouse. It's likely to be spread across multiple machines. **It does NOT provide query and presentation services** to end users.
3. **Presentation Server**: The target physical machine on which the data warehouse data is organized and stored for **direct querying** by end users, report writers, and applications. Data here should be presented and stored in a **dimensional framework**.
4. **Dimensional Model**: A specific discipline for modeling data — an alternative to the Entity-Relationship (E/R) model. Main components: **fact tables** and **dimension tables**. Better suited for decision support than E/R modeling.
5. **Data Mart**: A **logical subset** of the complete data warehouse. The full data warehouse = the union of all its data marts. Without conformed dimensions/facts, a data mart becomes an

isolated "stovepipe." A data mart can contain both summary data and atomic (detailed) data.

6. **Operational Data Store (ODS)**: Serves as the point of integration for operational systems — important for legacy systems that grew up independently. More geared toward real-time, account-based data queries. **Should NOT be considered part of the decision support system.**
7. **Metadata**: "Data about the data" — all information in the warehouse that is not the actual data itself. May be spread across several machines, in various formats.
8. **End User Data Access tools**:
    - **End User Application / Report Writers**: Tools that query, analyze, and present information for a specific business need.
    - **Ad hoc query tool**: Lets users form their own queries by directly manipulating relational tables and joins.
    - **Modeling Applications**: Sophisticated clients with analytic capabilities that transform/digest DW output — e.g. forecasting models, behavior scoring models, allocation models, data mining tools.

OLAP, ROLAP, MOLAP (your lecture's exact definitions)

- **OLAP**: The general activity of querying and presenting text and number data from the data warehouse in a **dimensional style**.
- **ROLAP** (Relational OLAP): A set of user interfaces and applications that give a relational database a dimensional flavor.
- **MOLAP** (Multidimensional OLAP): A set of user interfaces, applications, and proprietary database technology with a strong dimensional flavor.

Schemas — VERY commonly asked, draw both diagrams

Star Schema

- One central **Fact table** connected directly to multiple **Dimension tables**.
- Fact table = measures/numbers (sales_amount, quantity)
- Dimension tables = descriptive attributes (product, customer, time, location)
- Denormalized dimensions → simpler, faster queries, but some data redundancy.

```
        Dim_Time
           |
Dim_Product—Fact_Sales—Dim_Customer
           |
        Dim_Location
```

Snowflake Schema

- Extension of star schema where dimension tables are **normalized** into sub-dimensions.
- E.g., Dim_Product splits into Dim_Product → Dim_Category

- Saves storage, reduces redundancy, but queries need more joins (slower).

Star vs Snowflake — table form | Star Schema | Snowflake Schema | |---|---| | Denormalized dimensions | Normalized dimensions | | Simple structure, fewer joins | Complex structure, more joins | | Faster query performance | Slower due to joins | | More redundancy/storage | Less redundancy, saves storage |

Fact Constellation (Galaxy Schema): Multiple fact tables sharing dimension tables. Used for complex/multiple business processes.

Fact Table vs Dimension Table

- **Fact table**: contains quantitative data (measures) + foreign keys to dimensions. E.g., sales_id, product_id, customer_id, quantity, amount.
- **Dimension table**: contains descriptive/textual attributes used for filtering/grouping. E.g., product_name, category, customer_name, city.

---

2. ETL PROCESS

Definition

ETL = **Extract, Transform, Load**. The process of moving data from source systems into the data warehouse in usable form.

YOUR LECTURE'S FULL PROCESS LIST (broader than plain ETL — sir listed 10 processes, memorize the list even if you only detail a few)

1. Extracting
2. Transforming
3. Loading and Indexing
4. Quality Assurance Checking
5. Release/Publishing
6. Updating
7. Querying
8. Data Feedback
9. Auditing
10. Securing (+ Backing up and recovering)

The Core 3 — in your lecture's exact terms

1. Extracting

- The first step of getting data into the data warehouse environment.
- Involves reading and understanding the source data.

- Copying the needed parts from the source system into the **data staging area** for further work.

2. Transforming — has 5 sub-steps, know all 5 (commonly asked as a list):

- **Cleaning**: correcting misspellings, resolving domain conflicts, dealing with missing data elements.
- **Purging**: selecting fields from legacy data that are NOT useful for the data warehouse (discarding them).
- **Combining data sources**: matching the key across sources so records referring to the same entity are merged.
- **Creating surrogate keys**: system-generated keys that enforce referential integrity between dimension tables and fact tables.
- **Building aggregates**: pre-summarizing data for better query performance.

3. Loading and Indexing

- Usually replicates the dimension and fact tables from the data staging area into each data mart.
- Usually performed using the **bulk loading facility** of the data mart.
- **Indexing** should be done afterward for query performance.

The Rest of the Process (quick definitions — easy marks if asked to "list and explain")

- **Quality Assurance Checking**: The last step before publishing, after loading/indexing. Done by running a comprehensive **exception report** over the newly loaded data.
- **Release/Publishing**: Once a data mart is freshly loaded and QA'd, the user community is notified the new data is ready — including any changes to dimensions or new assumptions in measures/calculated facts.
- **Updating**: Modern data marts may update frequently via **"managed load updates"** (not transactional updates). Triggers include: data correction, label changes, hierarchy changes, status changes, corporate ownership changes.
- **Querying**: All activities requesting data from a data mart — ad hoc queries, model requests, data mining. **Always happens at the presentation server, never at the staging area.**
- **Data Feedback**: Uploading cleaned dimension descriptions from staging back to legacy source systems, or uploading query/model/mining results back into the data mart.
- **Auditing**: Critically important to know where data came from and what calculations were performed. Special audit records created during extraction and transformation.
- **Securing**: DW security must be managed **centrally**. Users need single sign-on access to all authorized data marts.

Diagram

```
Source System → [Extracting] → Data Staging Area → [Transforming: clean/purge/
combine/surrogate keys/aggregate] → [Loading & Indexing] → Presentation Server
→ [QA Check] → [Release/Publish] → End Users
```

Why these processes matter (if asked "importance/need of ETL")

- Ensures data quality and consistency across the organization
- Combines data from multiple disparate/legacy sources into one usable, dimensional format
- Enables historical, trustworthy analysis for decision support
- Auditing and QA build trust that numbers reaching decision-makers are correct

---

3. OLAP (Online Analytical Processing)

Definition

OLAP is a technology that enables multi-dimensional analysis of business data at high speed, allowing users to interactively analyze data from multiple perspectives.

OLAP vs OLTP (if not asked as DW vs DB, may be asked separately — same table as above basically)

OLAP Operations (VERY high-yield — draw examples for each)

1. **Roll-up**: Summarizing data by climbing up a concept hierarchy (e.g., city → state → country) or reducing dimensions.
2. **Drill-down**: Opposite of roll-up — going from summarized to detailed data (e.g., year → quarter → month).
3. **Slice**: Selecting one dimension from a cube to create a sub-cube (e.g., sales data for only "2024").
4. **Dice**: Selecting two or more dimensions to create a sub-cube (e.g., sales for "2024" AND "Product: Laptop").
5. **Pivot (Rotate)**: Rotating the data axes to provide an alternative view (e.g., swapping rows and columns — turning Product×Time into Time×Product).

```
        Roll-up ↑
Detailed data ⟷ Summarized data
        Drill-down ↓
```

Types of OLAP Servers

- **MOLAP** (Multidimensional OLAP): Data stored in multidimensional cube format. Fast, but limited storage capacity.
- **ROLAP** (Relational OLAP): Data stored in relational tables, uses SQL. Handles large data, but slower.

- **HOLAP** (Hybrid OLAP): Combination of both — detailed data in ROLAP, summary in MOLAP.

Type	Storage	Speed	Scalability
MOLAP	Multidimensional cube	Fast	Limited
ROLAP	Relational tables	Slower	High
HOLAP	Both	Balanced	Balanced

Data Cube

A data cube allows data to be modeled/viewed in multiple dimensions (e.g., Product × Time × Location × Sales). It's the core structure OLAP operations act on.

---

4. DATA VISUALIZATION

Definitions (from your lecture — memorize these three terms, they're often confused)

- **Visual**: something such as a picture, photograph, or piece of film used to explain something.
- **Visualization**: the act of creating a visual — representing something with a massive amount of data and higher complexity.
- **Visual Data Mining**: the process of discovering implicit but useful knowledge from large datasets **using visualization techniques**.

Why Data Visualization? (your lecture's list — good for a "benefits" question)

- Understand and analyze the problem domain faster
- Efficient data assimilation
- Fast access to business insight
- Fast identification of trends
- Direct interaction with data
- Offers different perspectives for the same data
- Easy to conduct business analytics
- Increases efficiency of AI-based expert systems

Table vs Graph — WHEN TO USE WHICH (draw this exact table, it's a classic exam question)

Tables are good when...	Graphs are good when...
User needs to look up specific values	User needs to absorb and process data fast

Tables are good when...	**Graphs are good when...**
User requires precise data	User wants to see relationships between values
User needs to precisely compare related values	User needs to see patterns and trends
Data has different sets of measurements	Message is contained in the *shape* of the values
—	Dataset is large

Common Display Types (from your lecture — 7 types, know them all)

Bar chart · Histogram · Pie chart · Line chart · Area chart · Scatter plot · Bubble chart

Bar Chart

- Presents categorical variables; height of the bar represents value.
- Can be stacked, grouped closer together, or 3D-rendered; horizontal or vertical.
- Very simple and easy to understand; can display values and percentages.

Pie Chart

- Summarizes a set of categorical/nominal data; emphasizes percentages relative to the whole.
- **Not good** for displaying exact values or when there are too many categories.
- Easily misinterpreted — use with care.

Line Chart

- Simple, fundamental representation of data; very good at showing **trends** and quantitative data.
- Can display multiple values (multiple lines); easily combined with bar charts, histograms, scatter plots.

Area Chart

- A variant of the line chart; good at showing trends, quantitative data, and relationships.
- Caution: sudden dips/spikes can distort the presentation.

Scatter Plot

- Represents the relationship between two variables; effective when a relationship exists.
- Very effective for continuous data and for **identifying cluster patterns**.
- Can extend to 3D scatter plots or bubble charts for multiple variables.

Other methods: other chart types, maps, animations, data models.

Bad Data Visualization / Common Pitfalls (your lecture spends multiple slides on this — expect a "what's wrong with this chart" or "list the pitfalls" question)

Common issues to be ready to critique: misleading/truncated axes, too many categories crammed into a pie chart, 3D effects that distort perception, no axis labels, poor color choice, cluttered charts, wrong chart type for the data/message.

Principles of Proper Data Visualization (your lecture's exact list — high-yield, likely a "how to design good visualization" question worth several marks)

1. **Arrange data clearly and presentably**: name the axes and values clearly, color-code different categories, organize data strategically (e.g., sort sales values), avoid cluttering.
2. **Understand what's vital**: identify the variables important to you/your organization, understand the association between them, identify the objective of the visualization. Don't present meaningless data just because you have it — avoid **data glut**.
3. **Pick the right chart**: know what you want to present and to whom (audience awareness). Make sure the chart doesn't alter, misinterpret, or give a wrong conclusion about the data.
4. **Keep it simple**: the main objective is to keep data simple, easy to understand, and analyze. Use simple language, avoid unnecessary adverbs/expressive language, and label charts clearly.

---

5. DATA MINING ALGORITHMS

This is the numerically-heaviest part. Understand the **logic + steps**, don't just memorize — if a numerical is given, you need to be able to execute it.

A. K-MEANS CLUSTERING

Type: Unsupervised learning, **clustering** algorithm (used for customer segmentation!)

Goal: Partition 'n' data points into 'k' clusters where each point belongs to the cluster with the nearest mean (centroid).

Algorithm Steps (memorize this exactly):

1. Choose the number of clusters **k**.
2. Randomly initialize **k centroids** (can be random data points).
3. **Assignment step**: Assign each data point to the nearest centroid (using Euclidean distance).
4. **Update step**: Recalculate each centroid as the mean of all points assigned to it.
5. Repeat steps 3-4 until centroids no longer change (convergence) or max iterations reached.

Euclidean Distance formula (know this):

```
d = √[(x2-x1)² + (y2-y1)²]
```

Simple worked example logic (be ready to solve a numerical):

- Given points, e.g.: A(2,3), B(3,3), C(6,8), D(8,8), k=2

- Pick 2 initial centroids (say A and C)
- Calculate distance from every point to each centroid → assign to nearest
- Recompute centroid = average of x's and average of y's of assigned points
- Repeat until stable

Advantages: Simple, fast, scales well to large datasets. **Disadvantages**: Must pre-specify k; sensitive to initial centroid choice; sensitive to outliers; struggles with non-spherical clusters.

How to choose k: Elbow method (plot within-cluster sum of squares against k, pick the "elbow" point).

B. APRIORI ALGORITHM

Type: **Association rule mining** (finds relationships between items — "customers who buy X also buy Y")

Core Concepts (memorize formulas — these are guaranteed numerical marks):

- **Support(A)** = (Transactions containing A) / (Total transactions)
- **Support(A→B)** = (Transactions containing both A and B) / (Total transactions)
- **Confidence(A→B)** = Support(A∪B) / Support(A)
- **Lift(A→B)** = Confidence(A→B) / Support(B)
  - Lift > 1 → positive correlation; Lift = 1 → independent; Lift < 1 → negative correlation

Algorithm Steps:

1. Set a **minimum support threshold**.
2. Generate **frequent 1-itemsets** (individual items meeting min support) — call this L1.
3. Generate candidate 2-itemsets (C2) by joining L1 with itself; calculate support; prune those below threshold → L2.
4. Repeat: generate C(k) from L(k-1), prune, get L(k) — continue until no more frequent itemsets can be formed.
5. **Apriori property (key idea)**: "All subsets of a frequent itemset must also be frequent" — this is used to prune candidates early (if any subset is infrequent, the superset is discarded without counting).
6. From final frequent itemsets, generate **association rules** meeting minimum confidence threshold.

Example logic to be ready for: Given a transaction table (TID, Items), you'll be asked to find frequent itemsets at a given min support (e.g., 50%), then generate rules with confidence.

Advantages: Easy to understand and implement, widely used (market basket analysis). **Disadvantages**: Computationally expensive — requires multiple database scans and generates a huge number of candidate sets for large datasets.

C. FP-GROWTH (Frequent Pattern Growth)

Type: Also association rule mining — an **improvement over Apriori**

Core idea: Avoids candidate generation (Apriori's biggest weakness) by using a compressed tree structure called the **FP-Tree**.

Algorithm Steps:

1. Scan the database once to find frequency of each item; discard items below min support.
2. Sort items in each transaction by descending frequency.
3. Build the **FP-Tree**: insert transactions into a tree structure, sharing common prefixes (paths) to compress data.
4. Use a **header table** to link all occurrences of the same item across the tree.
5. Mine the tree recursively using a **divide-and-conquer** approach: for each item, build its "conditional pattern base" and "conditional FP-tree" to extract frequent patterns — without generating candidates.

FP-Growth vs Apriori — high-yield comparison table: | Apriori | FP-Growth | |---|---| | Generates candidate itemsets | No candidate generation | | Multiple database scans | Only 2 database scans | | Slower on large datasets | Faster, more scalable | | Uses breadth-first search | Uses depth-first search (tree-based) |

Advantages: Much faster than Apriori, especially on large/dense datasets; no costly candidate generation. **Disadvantages**: FP-tree can be complex to build/implement; may not fit in memory for very large datasets.

---

6. SCENARIO-BASED QUESTIONS — FULL ANSWER TEMPLATES

These are worth the most marks per minute of prep since you can pre-structure the answer. **Use this format**: Identify technique → Justify why → Explain steps applied to scenario → Mention DW components involved (ETL/OLAP) if relevant.

SCENARIO 1: Customer Segmentation

Likely question: "A retail company wants to segment its customers based on purchasing behavior to design targeted marketing campaigns. Which data mining technique would you use? Explain how you would apply it."

Model Answer Structure:

1. Technique to use: K-Means Clustering Justification: Customer segmentation is an unsupervised learning problem — there are no predefined labels/categories, and the goal is to group similar customers together based on behavior. K-Means is ideal because it's efficient, scales to large customer datasets, and produces clearly interpretable clusters.

2. Relevant attributes/features (mention this — shows you understand data prep):

- Recency (days since last purchase), Frequency (number of purchases), Monetary value (total spend) — this is the classic **RFM model**, mention it, examiners love it.
- Also possibly: age, product category preference, average order value.

3. Steps applied to the scenario:

1. **Data collection/ETL**: Extract transaction data from the sales database (POS systems, e-commerce logs), transform it into RFM metrics per customer, load into the data warehouse.
2. **Choose k**: Use the elbow method to decide how many customer segments make sense (e.g., k=4: "high-value loyal", "at-risk", "new customers", "low-engagement").
3. **Run K-Means**: Assign each customer to the nearest centroid based on their RFM values (Euclidean distance), iteratively update centroids until convergence.
4. **Interpret clusters**: Label each resulting cluster based on its centroid characteristics (e.g., cluster with high frequency + high monetary = "VIP customers").
5. **Action**: Marketing team designs targeted campaigns per segment (e.g., loyalty rewards for VIPs, win-back discounts for at-risk customers).

4. Why not other techniques: Apriori/FP-Growth find item associations (what's bought together), not customer groupings — not suitable here. Classification isn't used because there's no pre-labeled "segment" to predict from.

SCENARIO 2: Examining Fraudulent Claims

Likely question: "An insurance company wants to identify potentially fraudulent claims from its historical data. How would you approach this using data mining?"

Model Answer Structure:

1. Technique to use: A combination approach — primarily Clustering (K-Means) for anomaly detection, and optionally Association Rule Mining (Apriori/FP-Growth) to detect suspicious co-occurring patterns.

Justification:

- Fraudulent claims are typically **rare and different from normal patterns (outliers)**. Clustering can group "normal" claims together, and claims that don't fit well into any cluster (or form a small, distant cluster) are flagged as **potential anomalies/fraud**.
- Association rule mining can additionally uncover suspicious **patterns of combinations —** e.g., certain claim types + certain hospitals + certain agents frequently appearing together in known fraud cases.

2. Relevant attributes/features:

- Claim amount, claim frequency by customer, time between policy purchase and claim, claim type, location, agent ID, prior claim history.

3. Steps applied to the scenario:

1. **ETL**: Extract historical claims data from operational systems, clean/transform (handle missing values, standardize claim categories), load into the warehouse.
2. **Clustering approach**:
   - Apply K-Means on claim attributes (amount, frequency, time-to-claim, etc.)
   - Most legitimate claims form dense, large clusters (normal behavior)
   - Claims that are far from all centroids, or form small sparse clusters, are flagged as **outliers → potential fraud**
3. **Association rule mining approach (supporting technique)**:
   - Apply Apriori/FP-Growth on claims data to find rules like: {Claim_Type=Theft, Location=X, Agent=Y} → high support/confidence of being historically fraudulent
   - High-confidence rules matching known fraud patterns flag new claims for investigation
4. **Output**: A ranked list of "suspicious" claims is passed to human fraud investigators for manual review (data mining flags, humans confirm — it's decision support, not full automation).

4. Why this combination: Fraud detection isn't a single clean clustering or association problem in practice — using clustering for **anomaly/outlier detection** plus association rules for **pattern-matching against known fraud signatures** gives a stronger, examiner-friendly answer that shows range.

---

7. QUICK-FIRE DEFINITIONS (memorize verbatim, likely to be a 1-2 mark sub-question)

- **Metadata**: "Data about data" — describes structure, source, and meaning of DW data.
- **Data Mart**: A subset of a data warehouse focused on a specific department/subject (e.g., Sales Data Mart).
- **Granularity**: The level of detail stored in the fact table (fine = transaction-level, coarse = summarized).
- **Data Cube**: Multidimensional structure allowing data to be viewed across multiple dimensions.
- **Surrogate Key**: A system-generated unique key used in dimension tables instead of natural/business keys.
- **Slowly Changing Dimension (SCD)**: A dimension where attribute values change slowly over time (e.g., customer address change) — handled via Type 1 (overwrite), Type 2 (new row/versioning), Type 3 (new column).

---

8. LAST-HOUR CHECKLIST

- ☐ Can you write the Inmon DW definition from memory?
- ☐ Can you draw Star schema AND Snowflake schema without looking?
- ☐ Can you list all 3 ETL steps with 1 sub-point each?
- ☐ Can you name all 5 OLAP operations and give an example of each?
- ☐ Can you write the K-Means steps 1-5 from memory?
- ☐ Can you write Support, Confidence, Lift formulas from memory?
- ☐ Can you explain Apriori property in one sentence?
- ☐ Can you give 2 differences between Apriori and FP-Growth?
- ☐ Can you explain, in your own words, why K-Means fits customer segmentation?
- ☐ Can you explain, in your own words, why clustering + association rules fit fraud detection?

If yes to all 10 — you're set for 90%+. Good luck. Go eat something before the exam, don't just cram till the bell.