

Question 3 – Data Warehouse Design

Context:

The administration wants to build a data warehouse using operational data (students, subjects, timetables, enrolments) and attendance data from scanned student cards.

Task Summary:

You need to design the **conceptual schema** for this warehouse and describe the **dimensions, facts, hierarchies, and OLAP query**.

Question Parts:

1. **3.1** Create a conceptual schema for the above specification of a sample data warehouse domain.
Use notations from ISIT312/ISIT912.
2. **3.2** Use relational-algebraic notation to specify the OLAP query:

“Find the total time spent by each student in lecture classes per session and lecture.”

Professor’s Example Solution (as seen on slides)

Entities:

STUDENT (StudId, Name)
SUBJECT (SubCode, School)
TIMETABLE (SubCode, Date, Time)
ENROLMENT (StudId, SubCode, EnrolDate)

Fact Table:

LECTURE_ATTENDANCE
(timeSpentInClass)

Relationships:

- STUDENT ← ENROLMENT → SUBJECT
- LECTURE_ATTENDANCE references TIMETABLE, STUDENT, SUBJECT

Measurements (Facts):

- totalTimeSpent
- totalNumOfAttendance
- timeSpentInClass
- totalNumOfStudents

Dimensions:

- Time
- Subject

- DegreeEnrolled
- School

Hierarchy (under Time dimension):

Time → Day → Week → Month → Session → Year

Conceptual Schema Diagram (described):

```

STUDENT (StudId, Name)
  ↓
ENROLMENT (EnrolId, StudId, SubCode)
  ↓
SUBJECT (SubCode, SubjectId, School)
  ↓
LECTURE_ATTENDANCE (timeSpentInClass)
  ↓
TIMETABLE (timeofdayId, dayId, weekId, monthId, sessionId, yearId)

```

Question 4 – HQL Logical Schema and Query

Conceptual Schema:

DRIVER (licence, first_name, last_name, ID)
COMPANY (name, phone, fax, ID)
TRUCK (rego, capacity, ID)
Relationships:
DRIVER → Employs → COMPANY → Owns → TRUCK

Tasks:

1. **4.1** Draw a *logical schema* based on this conceptual schema.
2. **4.2** Write HQL statements to implement the logical schema in Hive as *internal tables*.
3. **4.3** Write an HQL query to find:

“The average truck capacity of drivers who are associated with a truck.”

Question 6 – Spark Programming

(6.1)

Given:

File: people.txt (in HDFS)
Path: hdfs://localhost:8080/people/
Contents:
Michael, 29, software engineer
Andy, 30, data scientist
Justin, 19, business analyst

Task:

Load `people.txt` into a Spark DataFrame `peopleDF` where:

- Column 1 (name) → String
- Column 2 (age) → Integer
- Column 3 (job) → String

Sample Solution (Scala):

```
case class Person(name: String, age: Long, job: String)

val personDF = spark.sparkContext.textFile("people/people.txt")
  .map(_.split(","))
  .map(attributes => (attributes(0), attributes(1).trim.toInt,
  attributes(2)))
  .toDF("name", "age", "job")
```

(6.2)

Given:

```
FlightsDF
|-- DEST_CITY: string
|-- DEST_COUNTRY_NAME: string
|-- ORIGIN_CITY: string
|-- ORIGIN_COUNTRY_NAME: string
```

Query:

Find the country (or countries) with the most international flights.

(International flights have different origin and destination countries.)

Sample Spark SQL Solution:

```
flightDF.createOrReplaceTempView("flightsView")

SELECT originCountry, COUNT(*) AS totalIntFlight
FROM flightsView
WHERE destCountry != originCountry
GROUP BY originCountry
ORDER BY totalIntFlight DESC
LIMIT 1;
```

Would you like me to transform all this into a **formatted study summary sheet** (ideal for quick exam review) — including schema diagrams and sample OLAP notation?

You said:

This one now

ChatGPT said:

Here's the extracted and cleaned-up content from your new set of images — this covers **Question 4** and **Question 5** from your *ISIT312 Data Warehouse Design* tutorial slides (as presented by Mr. Sionggo Japit).

Question 4

Conceptual Schema:

```
DRIVER (licence, first_name, last_name, ID)
COMPANY (name, phone, fax, ID)
TRUCK (rego, capacity, ID)
Relationships:
    DRIVER → Employs → COMPANY → Owns → TRUCK
```

Question 4.1

Task:

Present a drawing of a logical schema based on the above conceptual schema (notation per ISIT312).

Sample Solution (Logical Schema):

```
DRIVER (drvLicence, drvFirstName, drvLastName, drvCoName)
COMPANY (coName, coPhone, coFax)
TRUCK (trkRego, trkCapacity, trkCoName)
DRIVER-TRUCK (drvTrkLicence, drvTrkRego, drvTrkCoName)
```

Relationships:

- DRIVER-TRUCK joins DRIVER and TRUCK via drvTrkCoName (foreign key to COMPANY).
-

Question 4.2

Task:

Write HQL statements to implement the logical schema as internal Hive tables.

Sample Solutions:

Table 1 – DRIVER

```
CREATE TABLE DRIVER(
    drvLicence VARCHAR(30),
    drvFirstName VARCHAR(30),
    drvLastName VARCHAR(30),
    drvCoName VARCHAR(30)
)
```

```
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;
```

Table 2 – COMPANY

```
CREATE TABLE COMPANY(
    coName VARCHAR(30),
    coPhone VARCHAR(20),
    coFax VARCHAR(20)
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;
```

Table 3 – TRUCK

```
CREATE TABLE TRUCK(
    trkRego VARCHAR(30),
    trkCapacity DECIMAL(5),
    trkCoName VARCHAR(30)
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;
```

Question 4.3

Task:

Write an HQL query to find the average truck capacity of drivers associated with a truck.

Sample Solution:

```
CREATE TABLE DRIVER-TRUCK(
    drvTrkLicence VARCHAR(30),
    drvTrkRego VARCHAR(30),
    drvTrkCo VARCHAR(30)
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

SELECT drvTrkRego, AVG(*)
FROM DRIVER-TRUCK
GROUP BY drvTrkRego;
```

Question 5

Conceptual Schema:

```
PERSON (licence_number, first_name, last_name)
ACCIDENT (severity, damage($))
CAR (registration, model)
LOCATION (street, city)
```

```
DATE (day, month, year)
Relationships:
PERSON -involved_in-> ACCIDENT -car_involved-> CAR
ACCIDENT -where_happened-> LOCATION
ACCIDENT -when_happened-> DATE
```

Question 5.1

Task:

Write HBase shell commands that create the HBase table and load sample data (at least two accidents, two cars, one person involved in both).

Sample Solution (Creating Table):

```
create 'QUESTIONS5', 'PERSON', 'ACCIDENT', 'CAR', 'LOCATION', 'DATE'
```

Insert Data (Cars):

```
put 'QUESTIONS5', 'car:SLF1234L', 'CAR:regno', 'SLF1234L'
put 'QUESTIONS5', 'car:SLF1234L', 'CAR:make', 'Honda'
put 'QUESTIONS5', 'car:SLF1234L', 'CAR:model', 'Civic'

put 'QUESTIONS5', 'car:SLQ4321M', 'CAR:regno', 'SLQ4321M'
put 'QUESTIONS5', 'car:SLQ4321M', 'CAR:make', 'Toyota'
put 'QUESTIONS5', 'car:SLQ4321M', 'CAR:model', 'Camry'
```

Insert Data (Locations):

```
put 'QUESTIONS5', 'location:Bukit Timah|Singapore', 'LOCATION:street',
'Bukit Timah'
put 'QUESTIONS5', 'location:Bukit Timah|Singapore', 'LOCATION:city',
'Singapore'
put 'QUESTIONS5', 'location:Bukit Batok|Singapore', 'LOCATION:street',
'Bukit Batok'
put 'QUESTIONS5', 'location:Bukit Batok|Singapore', 'LOCATION:city',
'Singapore'
```

Question 5.2

Task:

Assume the table is populated. Write HBase shell commands for these queries:

- Find all accidents with damages higher than 1000.
- List first and last names of people involved in accidents in Sydney in 2019.

Sample Solution:

(a)

```
SCAN 'QUESTIONS5',
{ COLUMNS => ['ACCIDENT'],
```

```
    FILTER =>
"SingleColumnValueFilter('ACCIDENT','damage','>','binary:1000')" }
```

(b)

```
SCAN 'QUESTIONS5',
{ COLUMNS => ['PERSON:fName', 'PERSON:lName'],
  FILTER =>
"SingleColumnValueFilter('ACCIDENT','city','=', 'binary:Sydney') AND
  SingleColumnValueFilter('DATE','year','=', 'binary:2019')" }
```

Would you like me to now consolidate *Questions 3–6* (including the ones you uploaded earlier) into a **single formatted exam review sheet** — complete with schema diagrams (ASCII format) and short explanations for each SQL/HQL/Spark/HBase example? It'll look like a ready-to-print revision aid.

You said:

transform all this into a formatted study summary sheet (ideal for quick exam review) — including schema diagrams and sample OLAP notation

ChatGPT said:

Here's your **ISIT312 Exam Study Summary Sheet** — clean, compact, and designed for rapid recall. It consolidates **Questions 3–6** (Data Warehouse, Hive, HBase, and Spark) into one ready-to-review reference.

ISIT312 – Data Warehouse & Big Data Exam Summary

Question 3 – Data Warehouse Design

Scenario

Admin wants to analyze student attendance data in a data warehouse. Data comes from operational systems: students, subjects, timetables, enrolments.

Fact Table

Lecture_Attendance

- Measures:

- o totalTimeSpent
 - o totalNumAttendance
 - o timeSpentInClass
 - o totalNumStudents
-

Dimensions

Dimension	Attributes	Hierarchy
Time	Day, Week, Month, Session, Year	Day → Week → Month → Session → Year
Subject	SubCode, School	—
Degree Enrolled	DegreeID, Name	—
School	SchoolName, Faculty	—

Conceptual Schema (ASCII)

```

STUDENT (StudId, Name)
  ↓
ENROLMENT (EnrolId, StudId, SubCode)
  ↓
SUBJECT (SubCode, School)
  ↓
LECTURE_ATTENDANCE (timeSpentInClass)
  ↓
TIMETABLE (timeOfDayId, dayId, weekId, monthId, sessionId, yearId)

```

Sample OLAP Query (3.2)

Find the total time spent by each student in lecture classes per session and lecture.

Relational-Algebra / OLAP Notation

```
ROLLUP(STUDENT.name, SESSION, LECTURE, SUM(timeSpentInClass))
```

Question 4 – Hive (HQL) Logical Schema

Conceptual Schema

DRIVER —Employs—> COMPANY —Owns—> TRUCK

4.1 Logical Schema

```

DRIVER (drvLicence, drvFirstName, drvLastName, drvCoName)
COMPANY (coName, coPhone, coFax)

```

```
TRUCK (trkRego, trkCapacity, trkCoName)
DRIVER-TRUCK (drvTrkLicence, drvTrkRego, drvTrkCoName)
```

4.2 Hive Table Creation

```
CREATE TABLE DRIVER(
    drvLicence VARCHAR(30),
    drvFirstName VARCHAR(30),
    drvLastName VARCHAR(30),
    drvCoName VARCHAR(30)
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE;

CREATE TABLE COMPANY(
    coName VARCHAR(30),
    coPhone VARCHAR(20),
    coFax VARCHAR(20)
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE;

CREATE TABLE TRUCK(
    trkRego VARCHAR(30),
    trkCapacity DECIMAL(5),
    trkCoName VARCHAR(30)
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE;
```

4.3 Query

Find the average truck capacity of drivers associated with a truck.

```
CREATE TABLE DRIVER-TRUCK(
    drvTrkLicence VARCHAR(30),
    drvTrkRego VARCHAR(30),
    drvTrkCo VARCHAR(30)
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE;

SELECT drvTrkRego, AVG(*)
FROM DRIVER-TRUCK
GROUP BY drvTrkRego;
```

Question 5 – HBase

Conceptual Schema

```
PERSON --involved_in--> ACCIDENT --car_involved--> CAR
      |   where_happened--> LOCATION
      |   when_happened--> DATE
```

5.1 Create Table

```
create 'QUESTIONS5', 'PERSON', 'ACCIDENT', 'CAR', 'LOCATION', 'DATE'
```

Insert Data

Cars:

```
put 'QUESTIONS5', 'car:SLF1234L', 'CAR:regno', 'SLF1234L'  
put 'QUESTIONS5', 'car:SLF1234L', 'CAR:make', 'Honda'  
put 'QUESTIONS5', 'car:SLF1234L', 'CAR:model', 'Civic'  
put 'QUESTIONS5', 'car:SLQ4321M', 'CAR:make', 'Toyota'  
put 'QUESTIONS5', 'car:SLQ4321M', 'CAR:model', 'Camry'
```

Locations:

```
put 'QUESTIONS5', 'location:Bukit Timah|Singapore', 'LOCATION:street',  
'Bukit Timah'  
put 'QUESTIONS5', 'location:Bukit Timah|Singapore', 'LOCATION:city',  
'Singapore'  
put 'QUESTIONS5', 'location:Bukit Batok|Singapore', 'LOCATION:street',  
'Bukit Batok'  
put 'QUESTIONS5', 'location:Bukit Batok|Singapore', 'LOCATION:city',  
'Singapore'
```

5.2 Queries

(a) Accidents with damages > 1000

```
SCAN 'QUESTIONS5',  
{ COLUMNS => ['ACCIDENT'],  
 FILTER =>  
"SingleColumnValueFilter('ACCIDENT','damage','>','binary:1000')" }
```

(b) Names of people in accidents in Sydney (2019)

```
SCAN 'QUESTIONS5',  
{ COLUMNS => ['PERSON:fName', 'PERSON:lName'],  
 FILTER =>  
"SingleColumnValueFilter('ACCIDENT','city','=','binary:Sydney') AND  
SingleColumnValueFilter('DATE','year','=','binary:2019')" }
```

Question 6 – Spark

6.1 Load `people.txt` from HDFS

Input File (HDFS):

```
hdfs://localhost:8080/people/people.txt
```

```
Michael, 29, software engineer  
Andy, 30, data scientist  
Justin, 19, business analyst
```

Scala Code:

```
case class Person(name: String, age: Long, job: String)

val personDF = spark.sparkContext.textFile("people/people.txt")
  .map(_.split(","))
  .map(attributes => (attributes(0), attributes(1).trim.toInt,
  attributes(2)))
  .toDF("name", "age", "job")
```

6.2 Find Country with Most International Flights

Schema:

```
FlightsDF
|-- DEST_CITY: string
|-- DEST_COUNTRY_NAME: string
|-- ORIGIN_CITY: string
|-- ORIGIN_COUNTRY_NAME: string
```

Spark SQL:

```
flightDF.createOrReplaceTempView("flightsView")

SELECT originCountry, COUNT(*) AS totalIntFlight
FROM flightsView
WHERE destCountry != originCountry
GROUP BY originCountry
ORDER BY totalIntFlight DESC
LIMIT 1;
```



Quick Reference Summary

Topic	Tool	Key Concepts
Q3	OLAP / Data Warehouse	Fact & Dimension Tables, Hierarchy, Rollup
Q4	HiveQL	Schema Creation, AVG Aggregation
Q5	HBase	Column Families, Filters, Scans
Q6	Spark (Scala)	RDD → DataFrame, Transformations, Temp Views

This sheet captures schema diagrams, table definitions, and query syntax exactly as taught — formatted for quick visual recognition.

Would you like me to add **color-coded highlights (Hive / HBase / Spark / OLAP)** and export it as a **PDF cheat sheet** for print or mobile viewing?

You said:

ChatGPT said:

Perfect — let's expand your **ISIT312 Study Summary Sheet** with **Questions 1 and 2**, then fold them seamlessly into your previous formatted layout.

Below is your **full quick-revision sheet**, now covering **Q1–Q6** in clean, exam-ready form.

ISIT312 – Data Warehouse & Big Data Systems

Exam Study Summary Sheet

Question 1 – Apache Pig vs Apache Spark

(i) Similarity in Data Processing

Concept	Apache Pig	Apache Spark	Commonality
Data Flow	Uses <i>Pig Latin</i> scripts to describe transformations	Uses RDD/DataFrame transformations in <i>Scala/Python</i>	Both use declarative pipelines rather than procedural code
Execution Model	Converts Pig Latin into a series of MapReduce jobs	Optimizes DAG (Directed Acyclic Graph) of stages	Both follow parallel data-flow execution
Input/Output	Reads from HDFS, outputs to HDFS	Reads/writes from HDFS, Hive, NoSQL, etc.	Both integrate tightly with Hadoop
Processing	Distributed, scalable, batch-oriented	Distributed, scalable, supports batch & streaming	Both support cluster-wide computation

(ii) Advantages of Spark over Pig

- **In-Memory Processing:** Spark avoids writing intermediate data to disk → faster than MapReduce-based Pig.
 - **API Flexibility:** Supports multiple languages (Scala, Python, Java, R) while Pig uses only Pig Latin.
 - **Real-Time Processing:** Spark Streaming supports near-real-time data, Pig is purely batch.
 - **Advanced Libraries:** Includes MLlib (machine learning), GraphX (graph processing), SparkSQL (structured data).
 - **Fault Tolerance:** Uses resilient distributed datasets (RDDs) with lineage tracking, no recomputation needed.
 - **Ease of Integration:** Works natively with Hive, HBase, Kafka, and JDBC sources.
-

Question 2 – MapReduce Model

Question

A table X contains:

key value

K1	1
K1	2
K2	3
K2	4

Implement:

```
SELECT key, SUM(value)
FROM X
GROUP BY key;
```

MapReduce Implementation

Mapper:

```
# Input: (K1,1), (K1,2), (K2,3), (K2,4)
# Output: key-value pairs
map(key, value):
    emit(key, value)
```

Reducer:

```
# Input: key with list of values
reduce(key, values):
    sum_val = sum(values)
    emit(key, sum_val)
```

Output:

K1 → 3
K2 → 7

Explanation:

- Mapper emits each key–value pair as-is.
 - Reducer aggregates values per key → performs SQL-like GROUP BY SUM().
-

Question 3 – Data Warehouse Design

Scenario

Data warehouse built from operational systems containing student attendance and timetable data.

Fact Table

LECTURE_ATTENDANCE

- **Measures:** totalTimeSpent, totalNumAttendance, timeSpentInClass, totalNumStudents.
-

Dimensions

Dimension	Attributes	Hierarchy
Time	Day, Week, Month, Session, Year	Day → Week → Month → Session → Year
Subject	SubCode, School	—
DegreeEnrolled	DegreeID, Name	—
School	SchoolName, Faculty	—

Conceptual Schema

```
STUDENT (StudId, Name)
  ↓
ENROLMENT (EnrolId, StudId, SubCode)
  ↓
SUBJECT (SubCode, School)
  ↓
LECTURE_ATTENDANCE (timeSpentInClass)
  ↓
TIMETABLE (dayId, weekId, monthId, sessionId, yearId)
```

OLAP Query Example:

Find total time spent by each student in lecture classes per session and lecture.

```
ROLLUP(STUDENT.name, SESSION, LECTURE, SUM(timeSpentInClass))
```

Question 4 – Hive (HQL) Logical Schema

Conceptual Schema

```
DRIVER -Employs→ COMPANY -Owns→ TRUCK
```

Logical Schema

```
DRIVER(drvLicence, drvFirstName, drvLastName, drvCoName)
COMPANY(coName, coPhone, coFax)
TRUCK(trkRego, trkCapacity, trkCoName)
DRIVER-TRUCK(drvTrkLicence, drvTrkRego, drvTrkCoName)
```

HQL Table Creation

```
CREATE TABLE DRIVER(
    drvLicence VARCHAR(30),
    drvFirstName VARCHAR(30),
    drvLastName VARCHAR(30),
    drvCoName VARCHAR(30))
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE;

CREATE TABLE COMPANY(
    coName VARCHAR(30),
    coPhone VARCHAR(20),
    coFax VARCHAR(20))
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE;

CREATE TABLE TRUCK(
    trkRego VARCHAR(30),
    trkCapacity DECIMAL(5),
    trkCoName VARCHAR(30))
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE;
```

Query (4.3):

```
SELECT drvTrkRego, AVG(*)
FROM DRIVER-TRUCK
GROUP BY drvTrkRego;
```

Question 5 – HBase

Conceptual Schema

```
PERSON → ACCIDENT → CAR
      ↓           ↓
      LOCATION     DATE
```

Create Table

```
create 'QUESTIONS5', 'PERSON', 'ACCIDENT', 'CAR', 'LOCATION', 'DATE'
```

Insert Sample Data

```
put 'QUESTIONS5', 'car:SLF1234L', 'CAR:make', 'Honda'
put 'QUESTIONS5', 'car:SLQ4321M', 'CAR:model', 'Camry'
```

```
put 'QUESTIONS5', 'location:Bukit Timah|Singapore', 'LOCATION:city',
'Singapore'
```

Queries

(a) Accidents with damages > 1000

```
SCAN 'QUESTIONS5',
{ COLUMNS => ['ACCIDENT'],
  FILTER =>
"SingleColumnValueFilter('ACCIDENT','damage','>','binary:1000')"
```

(b) People in accidents in Sydney, 2019

```
SCAN 'QUESTIONS5',
{ COLUMNS => ['PERSON:fName','PERSON:lName'],
  FILTER =>
"SingleColumnValueFilter('ACCIDENT','city','=','binary:Sydney') AND
SingleColumnValueFilter('DATE','year','=','binary:2019')"
```

Question 6 – Spark

6.1 Load `people.txt` from HDFS

File Content

```
Michael, 29, software engineer
Andy, 30, data scientist
Justin, 19, business analyst
```

Scala Code

```
case class Person(name: String, age: Long, job: String)

val personDF = spark.sparkContext.textFile("people/people.txt")
  .map(_.split(","))
  .map(a => (a(0), a(1).trim.toInt, a(2)))
  .toDF("name","age","job")
```

6.2 Flights Query

Schema

```
DEST_CITY, DEST_COUNTRY_NAME, ORIGIN_CITY, ORIGIN_COUNTRY_NAME
```

Goal

Find country/countries with most international flights.

Spark SQL

```
flightDF.createOrReplaceTempView("flightsView")  
  
SELECT originCountry, COUNT(*) AS totalIntFlight  
FROM flightsView  
WHERE destCountry != originCountry  
GROUP BY originCountry  
ORDER BY totalIntFlight DESC  
LIMIT 1;
```



Quick Comparison Table

Question	Technology	Focus	Key Syntax/Concept
Q1	Pig & Spark	Batch vs in-memory frameworks	Data flow, DAGs
Q2	MapReduce	SQL aggregation logic	map(), reduce()
Q3	OLAP	Star schema & hierarchies	ROLLUP, SUM()
Q4	HiveQL	Data definition in Hadoop	CREATE TABLE, AVG()
Q5	HBase	Column families & filters	SCAN, put
Q6	Spark (Scala)	DataFrames & SQL	toDF(), createTempView

This sheet now covers **every core ISIT312 practical** — Pig, Spark, Hive, HBase, and OLAP — in a single digestible layout.

Would you like me to generate a **color-coded PDF** version next (with syntax highlighting and visual schema boxes) for print and offline use?