Cheat Sheet: Tech Essentials–Code & Commands

Below areconcise cheat sheetsforeach requestedtopic, arrangedfortabulation in Excel or PDF. Each topic isstructuredin a tableformatfor quickreferenceand copy-paste to your preferred tool.

1. Basic & Advanced SQL: Cheat Sheet

Operation	**SQL Command/Example**	**Notes**
Select Data	`SELECT * FROM employees;`	Retrieve all columns
Select Columns	`SELECT name, salary FROM employees;`	Specify columns
Where Filter	`SELECT * FROM employees WHERE salary > 50000;`	Filter rows
Insert Data	`INSERT INTO dept(id,name) VALUES (1,'HR');`	Add a row
Update Data	`UPDATE employees SET salary=60000 WHERE id=3;`	Modify values
Delete Data	`DELETE FROM employees WHERE id=5;`	Remove rows

Join Tables	SELECT e.name, d.dept FROM employees e JOIN deptd ON e.dept_id=d.id;	INNER JOIN example
Group By	SELECT dept, AVG(salary) FROM employees GROUP BY dept;	Aggregate groupings
Subquery	SELECT name FROM emp WHERE salary=(SELECT MAX(salary) FROM emp);	Nested queries
Create Table	CREATE TABLE prod(id INT, name VARCHAR(50));	New table
Alter Table	ALTER TABLE emp ADD COLUMN age INT;	Add column
Index	CREATE INDEX idx_sal ON emp(salary);	Speed up queries
View	CREATE VIEW high_earners AS SELECT * FROM emp WHERE salary > 90000;	Logical table
Window Function	SELECT name, salary, RANK() OVER (ORDER BY salary DESC) FROM emp;	Advanced analytics
CTE (WITH)	WITH t AS (SELECT * FROM dept) SELECT * FROM t WHERE id=2;	Common Table Exp.

2. Unix Fundamentals & Shell Scripting: Cheat Sheet

Operation	Command/Script	Description
List Directory	`ls -l`	Detailed file listing
PWD	`pwd`	Show current directory
Change Dir	`cd /var/log`	Navigate
Create File	`touch demo.txt`	New empty file
Remove File	`rm demo.txt`	Delete file
Copy File	`cp a.txt b.txt`	Copy file
Move/Rename	`mv file1.txt docs/`	Move/rename
View File	`cat file.txt`	Output contents
Edit File	`nano file.sh` / `vi file.sh`	Use nano or vi editor
Find File	`find . -name "*.sh"`	Search recursive

Pipe	`cat a.txt	grep foo`
Permission Change	**chmod 755 script.sh**	Set execute permissions
Make Executable	**chmod +x runme.sh**	
Variable Assign	**name=test**	Set variable
Print Variable	**echo $name**	Output variable
Read Input	**read user**	User input
For Loop	**for i in 1 2 3; do echo $i;** **done**	Iterate values
If Condition	**if [-f file]; then echo yes;** **fi**	Test file existence
Run Bg Process	**sleep 10 &**	Run in background

Schedule Job	`crontab -e`	Edit cron jobs
Script Header	`#!/bin/bash`	Start any bash script

3. Data Modeling for BI: Cheat Sheet

Term/Task	**Code / Diagram Equivalent**	**Explanation**
Entity	ENTITY: Customer, Product	Real-world objects
Attribute	ATTRIBUTES: name, price, contact_no	Entity properties
PK / FK	`PRIMARY KEY (id)`, `FOREIGN KEY (cust_id)` `REFERENCES customer(id)`	Relationships
Fact Table	Measures: sales_amount, qty_sold	Numeric, central
Dimension Table	Details: product, date, store	Lookup/context
Star Schema	Fact Table at center, Dimension tables as spokes	BI-friendly
Snowflake Schema	Normalized dimension tables (split "location" into	Storage efficient

	region/city)	
ER Diagram (draw.io)	Boxes for entities, lines for relationships	Visual in any diagram tool
Sample Star Schema SQL	`CREATE TABLE fact_sales (sale_id INT, prod_id INT, amt NUMERIC);`	Fact Table example

4. Data Warehouse Concepts: Cheat Sheet

Concept	**Definition / Example**
ETL	Extract, Transform, Load
Data Warehouse	Central repo for analytics
Data Mart	Dept.-level DW subset
OLAP	Multi-dimensional analytics
Fact Table	Stores measurable events
Dimension Table	Stores context attributes

Metadata	Data about data
Star Schema	Central fact, direct dims
Snowflake Schema	Normalized dimensions
Window Function (SQL)	`RANK() OVER (ORDER BY ...)`
Partitioning	Split warehouse tables

5. Scala Programming (Basic & Advanced): Cheat Sheet

Task	**Code Example**	**Note**
Variable Declaration	`val x = 10` (immut.), `var y = 5` (mut.)	Use `val` by default
Function	`def add(a: Int, b: Int): Int = a + b`	Works like a method
Main Method	`object App { def main(args: Array[String]) {...} }`	Program entry point

List	`val nums = List(1,2,3)`	Immutable seq.
Map & Filter	`nums.filter(_ > 1).map(_ * 2)`	Functional pattern
Case Class	`case class Book(t: String, p: Double)`	Immutable POJO
Pattern Matching	`x match { case 1 => ...; case _ => ... }`	Switch alternative
For Loop	`for(e <- nums) println(e)`	Iteration
Try/Catch	`Try( x / y ).getOrElse(-1)`	Exception handling
Parallel Collections	`nums.par.map(_ + 1)`	Auto parallelism

6. Maven for Packaging: Cheat Sheet

Command/Task	**Command Example**	**Purpose**
Create Maven Project	`mvn archetype:generate`	Interactive startup
Build Project	`mvn package`	Compile & JAR/WAR output

Run Tests	`mvn test`	Execute unit tests
Clean Build	`mvn clean`	Remove build artifacts
Install Locally	`mvn install`	Deploy to local repo
Add Dependency	`<dependency>...</dependency>` in POM	Add libs via `pom.xml`
Run Main Class	`mvn exec:java -Dexec.mainClass="pkg.Main"`	Run main app
Show Dependencies	`mvn dependency:tree`	List project dependencies

7. Unit Testing Framework (Scala: ScalaTest): Cheat Sheet

Feature	**Code Example**	**Notes**
Add Dependency (sbt)	`libraryDependencies += "org.scalatest" %% "scalatest" % "3.x" % Test`	`build.sbt`
Basic Test Class	`class MySpec extends AnyFunSuite { ... }`	Import style: `funsuite`

Simple Test	`test("should add") { assert(add(2,3)==5) }`	JUnit-like
FlatSpec	`"A List" should "return correct size" in { ... }`	Readable
Exception Testing	`intercept[ArithmeticException] { ... }`	Verify thrown errors
Property Testing	`forAll { (n: Int) => ... }`	Use ScalaCheck
Run Tests (sbt)	`sbt test`	CLI test run