<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>Futures Contracts Pricing</title>

</head>

<body>

<h1>Futures Contracts Pricing</h1>

<p>Open the browser console to view pricing outputs.</p>

<script>

// Base market prices

const COMEX\_Cu = 4.19; // Copper price per lb (COMEX)

const LME\_Al = 1.25; // Aluminum price per lb (LME)

const LME\_Lead = 2034 / 2000; // Lead price per lb (approx. $1.017)

const LME\_Zinc = 2879 / 2000; // Zinc price per lb (approx. $1.44)

const LME\_Ni = 15675 / 2000; // Nickel price per lb (approx. $7.84)

// ----------------------------

// Copper Pricing Formulas

// ----------------------------

// These functions return a price based on COMEX copper multiplied by a grade-specific factor.

function copperPrice(grade) {

switch(grade) {

case 'Barley': // Cu BB

return COMEX\_Cu \* 0.94;

case '#1': // Berry & Candy

return COMEX\_Cu \* 0.91;

case '#2': // Birch & Cliff

return COMEX\_Cu \* 0.85;

case '#3': // Sheet Copper

return COMEX\_Cu \* 0.83;

default:

return null;

}

}

// ----------------------------

// Aluminum Pricing Formulas

// ----------------------------

// Price = LME\_Al \* multiplier, where the multiplier is specific to each aluminum product.

function aluminumPrice(product) {

switch(product) {

case 'AL6061\_NewBareExtrusion':

return LME\_Al \* 0.84;

case 'AL6061\_ClipPlatePipe':

return LME\_Al \* 0.78;

case 'AL6063\_Secondary':

return LME\_Al \* 0.78;

case 'AL6063\_OldExtrusion':

return LME\_Al \* 0.95;

case 'AL6063\_Extrusion10/10':

return LME\_Al \* 0.85;

case 'AL\_PAS\_Baled':

return LME\_Al \* 0.82;

case 'AL\_Cast\_Clean':

return LME\_Al \* 0.72;

case 'AL\_RADS\_CleanBaled':

return LME\_Al \* 0.67;

case 'AL\_RADS\_DirtyBaled':

return LME\_Al \* 0.46;

case 'AL\_Sheet\_CleanBaled':

return LME\_Al \* 0.75;

case 'AL\_WheelAuto\_Clean': // Clean (Dirty 2¢ less)

return LME\_Al \* 0.93;

case 'AL\_WheelChrome\_Clean': // Clean (Dirty 2¢ less)

return LME\_Al \* 0.80;

case 'TruckWheel\_Clean':

return LME\_Al \* 0.72;

case 'AL\_Wire\_BareCATV':

return LME\_Al \* 0.43;

case 'AL\_Wire\_NeopreneACSR':

return LME\_Al \* 0.54;

case 'AL\_Wire\_ECPrepared':

return LME\_Al \* 1.03;

case 'AL\_MLC\_Prepared':

return LME\_Al \* 0.84;

default:

return null;

}

}

// ----------------------------

// Brass Pricing Formulas

// ----------------------------

// Brass prices are derived from COMEX copper using specific multipliers.

function brassPrice(product) {

switch(product) {

case 'Brass\_Hard': // Solid Clean

return COMEX\_Cu \* 0.81;

case 'Brass\_Red\_Ebony':

return COMEX\_Cu \* 0.77;

case 'Brass\_Red\_Semi':

return COMEX\_Cu \* 0.72;

case 'Brass\_Shaving\_AlBronze': // Brass Shaving (Al Bronze Shaving, C/D)

return COMEX\_Cu \* 0.57;

case 'Brass\_Shaving\_BrassRed': // Brass Shaving (Brass Red Shaving, C/D)

return COMEX\_Cu \* 0.61;

case 'Brass\_Shell': // Clean, No Chrome

return COMEX\_Cu \* 0.60;

case 'Brass\_Special': // Al Bronze Solid

return COMEX\_Cu \* 0.64;

case 'Brass\_YellowRegular': // Solid Clean

return COMEX\_Cu \* 0.61;

default:

return null;

}

}

// ----------------------------

// Lead Pricing Formulas

// ----------------------------

// Prices based on LME Lead.

function leadPrice(product) {

switch(product) {

case 'Lead\_Clean':

return LME\_Lead \* 0.718; // Approximates ~$0.73 per lb

case 'Lead\_Range':

return LME\_Lead \* 0.728; // Approximates ~$0.74 per lb

default:

return null;

}

}

// ----------------------------

// Stainless Steel Pricing Formulas

// ----------------------------

// Prices based on LME Nickel.

function stainlessPrice(product) {

switch(product) {

case 'SS304':

return LME\_Ni \* 0.0638; // Approximates ~$0.50 per lb

case 'SS316':

return LME\_Ni \* 0.1174; // Approximates ~$0.92 per lb

default:

return null;

}

}

// ----------------------------

// E‑Scrap Pricing Formula (Generic)

// ----------------------------

// Computes an E‑scrap board price based on recoverable metals.

// Parameters:

// goldContent: Troy ounces of gold per lb of board

// goldSpot: Gold spot price in $ per troy ounce

// silverContent: Troy ounces of silver per lb of board

// silverSpot: Silver spot price in $ per troy ounce

// copperContent: Fraction of a lb of copper per lb of board

// processingFee: Deduction per lb for processing

function eScrapPrice(goldContent, goldSpot, silverContent, silverSpot, copperContent, processingFee) {

return (goldContent \* goldSpot) +

(silverContent \* silverSpot) +

(copperContent \* COMEX\_Cu) -

processingFee;

}

// ----------------------------

// New: Insulated Wire Pricing Formulas

// ----------------------------

// These prices are derived using LME\_Al as the base.

// Multipliers are determined by: multiplier = (listed price) / 1.25

function insulatedWirePrice(product) {

switch(product) {

case 'ICW\_LowGrade\_ChristmasLights':

return LME\_Al \* 0.528; // 1.25 \* 0.528 ≈ 0.660

case 'ICW\_LowGrade\_ComputerWire25':

return LME\_Al \* 0.448; // ≈ 0.560

case 'ICW\_LowGrade\_CuCATV':

return LME\_Al \* 1.008; // ≈ 1.260

case 'ICW\_LowGrade\_ExtCords35Up':

return LME\_Al \* 0.92; // ≈ 1.150

case 'ICW\_LowGrade\_MixedWire40Up':

return LME\_Al \* 1.08; // ≈ 1.350

case 'ICW1\_Heliax57\_OpenEyeBaled':

return LME\_Al \* 1.536; // ≈ 1.920

case 'ICW1\_MCM85HG':

return LME\_Al \* 2.536; // ≈ 3.170

case 'ICW1\_Romex65\_NoWeatherProof':

return LME\_Al \* 1.904; // ≈ 2.380

case 'ICW1\_THHN80':

return LME\_Al \* 2.36; // ≈ 2.950

case 'ICW2\_50Cat5TelWire':

return LME\_Al \* 1.384; // ≈ 1.730

case 'ICW2\_HarnessWireNoFuseBoxes':

return LME\_Al \* 1.408; // ≈ 1.760

case 'ICW2\_BX\_Cable\_AL':

return LME\_Al \* 1.648; // ≈ 2.060

case 'ICW2\_BX\_Cable\_FE24':

return LME\_Al \* 0.6; // ≈ 0.750

default:

return null;

}

}

// ----------------------------

// New: Misc Product Pricing Formulas

// ----------------------------

// For most items here we use LME\_Al as base (1.25), except for Zinc Die Cast which uses LME\_Zinc.

function miscProductPrice(product) {

switch(product) {

case 'Ballast\_Electronic':

return LME\_Al \* 0.12; // 1.25 \* 0.12 = 0.150

case 'Ballast\_Regular':

return LME\_Al \* 0.216; // 1.25 \* 0.216 = 0.270

case 'BatteryAuto\_PbAcid':

return LME\_Al \* 0.216; // ≈ 0.270

case 'BatterySteelCase\_Industrial':

return LME\_Al \* 0.192; // 1.25 \* 0.192 = 0.240

case 'Compressor\_SealedUnitNoCastIron':

return LME\_Al \* 0.264; // 1.25 \* 0.264 = 0.330

case 'EMotorLowGrade\_CeilingFanMotors':

return LME\_Al \* 0.144; // 1.25 \* 0.144 = 0.180

case 'EMotorMix\_NoPumpsPowerToolsFans':

return LME\_Al \* 0.344; // 1.25 \* 0.344 = 0.430

case 'EMotorLarge\_LessThan1000lbs':

return LME\_Al \* 0.312; // 1.25 \* 0.312 = 0.390

case 'Transformer\_AluLargeOnly':

return LME\_Al \* 0.176; // 1.25 \* 0.176 = 0.220

case 'TransformerCu\_AlCu':

return LME\_Al \* 0.28; // 1.25 \* 0.28 = 0.350

case 'TransformerCu\_SmallPalmSize':

return LME\_Al \* 0.56; // 1.25 \* 0.56 = 0.700

case 'TransformerCu\_MediumOver200lbs':

return LME\_Al \* 0.592; // 1.25 \* 0.592 = 0.740

case 'TransformerCu\_Large':

return LME\_Al \* 0.624; // 1.25 \* 0.624 = 0.780

case 'Zinc\_DieCast':

return LME\_Zinc \* 0.5903; // 1.44 \* 0.5903 ≈ 0.850

default:

return null;

}

}

// ----------------------------

// New: Radiator Product Pricing Formulas

// ----------------------------

// These products are also derived using LME\_Al as base.

function radiatorProductPrice(product) {

switch(product) {

case 'ACReefer\_CleanTalkBaled':

return LME\_Al \* 1.608; // 1.25 \* 1.608 ≈ 2.010

case 'ACReefer\_CleanUnbaled':

return LME\_Al \* 1.592; // ≈ 1.990

case 'ACReefer\_DirtyTalkBaled':

return LME\_Al \* 1.496; // ≈ 1.870

case 'ACReefer\_DirtyUnbaled':

return LME\_Al \* 1.48; // ≈ 1.850

case 'Radiator\_CleanCar':

return LME\_Al \* 1.984; // ≈ 2.480

case 'CuBkg\_ACReeferEnd':

return LME\_Al \* 1.008; // ≈ 1.260

default:

return null;

}

}

// ----------------------------

// Material Name Arrays

// ----------------------------

// 1) Copper

const copperMaterials = [

"Barley", // Cu BB

"#1", // Berry & Candy

"#2", // Birch & Cliff

"#3" // Sheet Copper

];

// 2) Aluminum

const aluminumMaterials = [

"AL6061\_NewBareExtrusion",

"AL6061\_ClipPlatePipe",

"AL6063\_Secondary",

"AL6063\_OldExtrusion",

"AL6063\_Extrusion10/10",

"AL\_PAS\_Baled",

"AL\_Cast\_Clean",

"AL\_RADS\_CleanBaled",

"AL\_RADS\_DirtyBaled",

"AL\_Sheet\_CleanBaled",

"AL\_WheelAuto\_Clean",

"AL\_WheelChrome\_Clean",

"TruckWheel\_Clean",

"AL\_Wire\_BareCATV",

"AL\_Wire\_NeopreneACSR",

"AL\_Wire\_ECPrepared",

"AL\_MLC\_Prepared"

];

// 3) Brass

const brassMaterials = [

"Brass\_Hard",

"Brass\_Red\_Ebony",

"Brass\_Red\_Semi",

"Brass\_Shaving\_AlBronze",

"Brass\_Shaving\_BrassRed",

"Brass\_Shell",

"Brass\_Special",

"Brass\_YellowRegular"

];

// 4) Lead

const leadMaterials = [

"Lead\_Clean",

"Lead\_Range"

];

// 5) Stainless

const stainlessMaterials = [

"SS304",

"SS316"

];

// 6) E-Scrap (generic placeholder)

const eScrapMaterials = [

"E\_Scrap\_Board"

];

// 7) Insulated Wire

const insulatedWireMaterials = [

"ICW\_LowGrade\_ChristmasLights",

"ICW\_LowGrade\_ComputerWire25",

"ICW\_LowGrade\_CuCATV",

"ICW\_LowGrade\_ExtCords35Up",

"ICW\_LowGrade\_MixedWire40Up",

"ICW1\_Heliax57\_OpenEyeBaled",

"ICW1\_MCM85HG",

"ICW1\_Romex65\_NoWeatherProof",

"ICW1\_THHN80",

"ICW2\_50Cat5TelWire",

"ICW2\_HarnessWireNoFuseBoxes",

"ICW2\_BX\_Cable\_AL",

"ICW2\_BX\_Cable\_FE24"

];

// 8) Misc Product

const miscProductMaterials = [

"Ballast\_Electronic",

"Ballast\_Regular",

"BatteryAuto\_PbAcid",

"BatterySteelCase\_Industrial",

"Compressor\_SealedUnitNoCastIron",

"EMotorLowGrade\_CeilingFanMotors",

"EMotorMix\_NoPumpsPowerToolsFans",

"EMotorLarge\_LessThan1000lbs",

"Transformer\_AluLargeOnly",

"TransformerCu\_AlCu",

"TransformerCu\_SmallPalmSize",

"TransformerCu\_MediumOver200lbs",

"TransformerCu\_Large",

"Zinc\_DieCast"

];

// 9) Radiator Product

const radiatorProductMaterials = [

"ACReefer\_CleanTalkBaled",

"ACReefer\_CleanUnbaled",

"ACReefer\_DirtyTalkBaled",

"ACReefer\_DirtyUnbaled",

"Radiator\_CleanCar",

"CuBkg\_ACReeferEnd"

];

// ----------------------------

// Group all material names

// ----------------------------

const materialNames = {

copper: copperMaterials,

aluminum: aluminumMaterials,

brass: brassMaterials,

lead: leadMaterials,

stainless: stainlessMaterials,

eScrap: eScrapMaterials,

insulatedWire: insulatedWireMaterials,

miscProduct: miscProductMaterials,

radiatorProduct: radiatorProductMaterials

};

// ----------------------------

// Example Usage

// ----------------------------

console.log("Copper Barley Price:", copperPrice("Barley").toFixed(2));

console.log("AL6061 New Bare Extrusion Price:", aluminumPrice("AL6061\_NewBareExtrusion").toFixed(2));

console.log("Brass Hard Price:", brassPrice("Brass\_Hard").toFixed(2));

console.log("Lead Clean Price:", leadPrice("Lead\_Clean").toFixed(2));

console.log("SS304 Price:", stainlessPrice("SS304").toFixed(2));

// Example for E-Scrap:

// Assume: 0.001 troy oz gold, 0.01 troy oz silver, 0.05 lb copper per lb of board, and a processing fee of $0.10.

console.log("E-Scrap Board Price:", eScrapPrice(0.001, 1800, 0.01, 20, 0.05, 0.10).toFixed(2));

// New examples for additional categories:

console.log("ICW Low Grade Christmas Lights Price:", insulatedWirePrice("ICW\_LowGrade\_ChristmasLights").toFixed(3));

console.log("Ballast Electronic Price:", miscProductPrice("Ballast\_Electronic").toFixed(3));

console.log("AC Reeefer Clean Talk Baled Price:", radiatorProductPrice("ACReefer\_CleanTalkBaled").toFixed(3));

console.log("All Material Names:", materialNames);

</script>

</body>

</html>