Insights into RDS Reserved Instances

Approach:

We saw in the documentation that we can use the *aws –describe* instances feature. Similarly with some scripting we can get the instance pricing description. A high level overview is that we can utilize scripts that can run in Cloudshell.

The main script:

```
#!/bin/bash

# --- Configuration ---
if [[ "$#" -gt 0 ]]; then
    INSTANCE_TYPES=("$@")
else
    INSTANCE_TYPES=("db.m5.large" "db.r5.large")
fi

# --- Fixed Script Parameters ---
REGION="us-east-1"
DEPLOYMENT_OPTION="Multi-AZ"
HOURS_IN_YEAR=8760

# --- Functions ---
get_ondemand_price() {
    local instance_type="$1"
    local db_engine="$2"
    local license_model="$3"
    local filters=("Type=TERM_MATCH,Field=regionCode,Value=$REGION"
"Type=TERM_MATCH,Field=instanceType,Value=$instance_type"
"Type=TERM_MATCH,Field=databaseEngine,Value=$db_engine"
"Type=TERM_MATCH,Field=licenseModel,Value=$license_model"
"Type=TERM_MATCH,Field=deploymentOption,Value=$DEPLOYMENT_OPTION"
"Type=TERM_MATCH,Field=termType,Value=OnDemand")
    local price_json=$(aws pricing get-products --service-code AmazonRDS --filters
"${filters[@]}" --region us-east-1 2>/dev/null)
    local hourly_price=$(echo "$price_json" | jq -r '.PriceList | .[]? | fromjson | .terms.OnDemand
| .. | .pricePerUnit?.USD? | select(. != null)' | head -n 1)
    echo "${hourly_price:-0.0}"
}

get_ri_details() {
    local instance_type="$1"
    local db_engine="$2"
    local license_model="$3"
```

```
    local lease_contract_length="$4"
    local purchase_option="$5"
    local filters=("Type=TERM_MATCH,Field=regionCode,Value=$REGION"
"Type=TERM_MATCH,Field=instanceType,Value=$instance_type"
"Type=TERM_MATCH,Field=databaseEngine,Value=$db_engine"
"Type=TERM_MATCH,Field=licenseModel,Value=$license_model"
"Type=TERM_MATCH,Field=deploymentOption,Value=$DEPLOYMENT_OPTION"
"Type=TERM_MATCH,Field=termType,Value=Reserved"
"Type=TERM_MATCH,Field=leaseContractLength,Value=$lease_contract_length")
    local price_json=$(aws pricing get-products --service-code AmazonRDS --filters
"${filters[@]}" --region us-east-1 2>/dev/null)
    local offer_term=$(echo "$price_json" | jq -c '.PriceList | .[]? | fromjson | .terms.Reserved |
.[] | select(.termAttributes.LeaseContractLength == """$lease_contract_length""" and
.termAttributes.PurchaseOption == """$purchase_option""")')
    if [[ -z "$offer_term" ]]; then
        echo "0.0 0.0"
        return
    fi
    local hourly_price=$(echo "$offer_term" | jq -r '.priceDimensions | .[] | select(.unit == "Hrs") |
.pricePerUnit.USD // "0.0"')
    local upfront_fee=$(echo "$offer_term" | jq -r '.priceDimensions | .[] | select(.unit ==
"Quantity") | .pricePerUnit.USD // "0.0"')
    echo "${hourly_price:-0.0} ${upfront_fee:-0.0}"
}

# --- Main Script ---
if ! command -v aws &> /dev/null || ! command -v jq &> /dev/null || ! command -v bc &>
/dev/null; then
    echo "Error: Missing dependencies." >&2; exit 1
fi
echo " Dependencies found. Fetching prices for Region: $REGION, Deployment:
$DEPLOYMENT_OPTION..."
echo ""
printf "%-18s | %-15s | %-20s | %-22s | %-22s | %-22s\n" "Instance Type" "DB Engine"
"On-Demand (Annual)" "1-Yr RI No Upfront" "1-Yr RI Partial Upfront" "1-Yr RI All Upfront"
printf "%-18s | %-15s | %-20s | %-22s | %-22s | %-22s\n" "" "" "(Annualized Cost)" "3-Yr RI No
Upfront" "3-Yr RI Partial Upfront" "3-Yr RI All Upfront"
printf '%s\n'
'-------------------+-----------------+----------------------+------------------------+------------------------+---------
---------------'

for instance in "${INSTANCE_TYPES[@]}"; do
    declare -A engine_license_map
    engine_license_map=(["MySQL"]="No License required" ["PostgreSQL"]="No License
required" ["MariaDB"]="No License required")
    for engine in "${!engine_license_map[@]}"; do
        license=${engine_license_map[$engine]}
        on_demand_price=$(get_ondemand_price "$instance" "$engine" "$license")
        if [[ $(bc <<<"$on_demand_price <= 0.0") -eq 1 ]]; then continue; fi
        on_demand_annual=$(bc <<<"scale=2; $on_demand_price * $HOURS_IN_YEAR / 1")
```

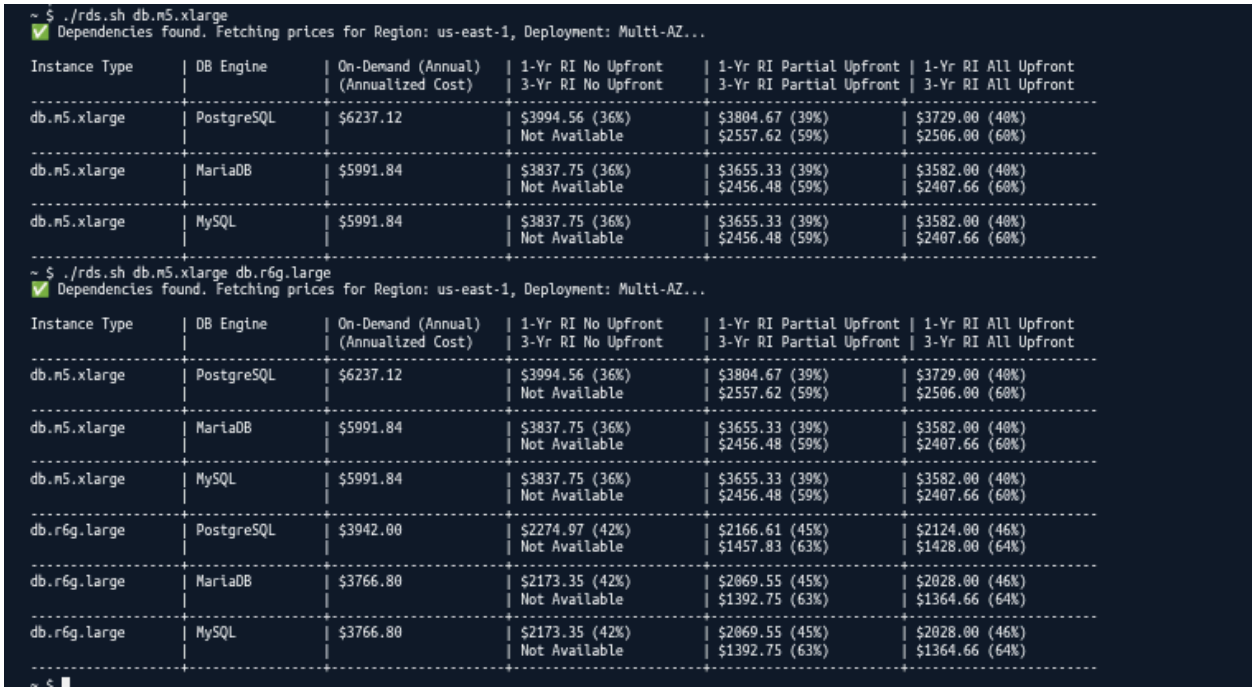
```
    declare -A ri_costs
    purchase_options=("No Upfront" "Partial Upfront" "All Upfront")
    term_lengths=("1yr" "3yr")

    for term in "${term_lengths[@]}"; do
        for po in "${purchase_options[@]}"; do
            read hourly upfront <<<$(get_ri_details "$instance" "$engine" "$license" "$term"
"$po")

            # CORRECTED: Removed the 'local' keyword, as it can only be used in a function.
            term_in_years=
            if [[ "$term" == "1yr" ]]; then term_in_years=1; else term_in_years=3; fi

            # 1. Calculate the total cost over the entire term
            total_term_cost=$(bc <<<"scale=2; ($hourly * $HOURS_IN_YEAR *
$term_in_years) + $upfront / 1")
            # 2. Calculate the effective annual cost
            effective_annual_cost=$(bc <<<"scale=2; $total_term_cost / $term_in_years")

            key="${term}_${po// /_}"
            ri_costs[$key]=$effective_annual_cost
        done
    done

    declare -A display_strings
    for term in "${term_lengths[@]}"; do
        for po in "${purchase_options[@]}"; do
            key="${term}_${po// /_}"
            cost=${ri_costs[$key]}
            display_key="${term}_${po}"
            if [[ $(bc <<<"$cost > 0.0") -eq 1 ]]; then
                discount=$(bc -l <<<"scale=2; 100 * ($on_demand_annual - $cost) /
$on_demand_annual")
                display_strings[$display_key]=$(printf "\$%.2f (%2.0f%%)" "$cost" "$discount")
            else
                display_strings[$display_key]="Not Available"
            fi
        done
    done

    printf "%-18s | %-15s | %-20s | %-22s | %-22s | %-22s\n" "$instance" "$engine"
"\$${on_demand_annual}" "${display_strings['1yr_No Upfront']}" "${display_strings['1yr_Partial
Upfront']}" "${display_strings['1yr_All Upfront']}"
    printf "%-18s | %-15s | %-20s | %-22s | %-22s | %-22s\n" "" "" ""
"${display_strings['3yr_No Upfront']}" "${display_strings['3yr_Partial Upfront']}"
"${display_strings['3yr_All Upfront']}"
    printf '%s\n'
'------------------+-----------------+----------------------+-----------------------+-----------------------+---------
---------------'
```

```
    done
done
```

Now we save this file as rds.sh in the Cloud shell, also install this dependency:
sudo yum install bc -y

To use this script:
First do: **`chmod +x rds.sh`**
Then we simply do: **`./rds.sh <space> <instance name(multiple)>`**

Example can be seen below:



Updated code for generating CSV files:

```
#!/bin/bash

# --- Configuration ---
CSV_MODE=false
if [[ "$1" == "--csv" ]]; then
    CSV_MODE=true
    shift # Remove the --csv flag to process the remaining arguments
fi

if [[ "$#" -gt 0 ]]; then
    INSTANCE_TYPES=("$@")
else
    INSTANCE_TYPES=("db.m5.large" "db.r5.large")
fi

# --- Fixed Script Parameters ---
REGION="us-east-1"
DEPLOYMENT_OPTION="Multi-AZ"
HOURS_IN_YEAR=8760

# --- Functions ---

# OPTIMIZED: Fetches all price data for an engine in just 2 API calls.
get_all_prices() {
    local instance_type="$1"
    local db_engine="$2"
    local license_model="$3"

    # 1. Get On-Demand Price
    local ondemand_filters=("Type=TERM_MATCH,Field=regionCode,Value=$REGION"
"Type=TERM_MATCH,Field=instanceType,Value=$instance_type"
"Type=TERM_MATCH,Field=databaseEngine,Value=$db_engine"
"Type=TERM_MATCH,Field=licenseModel,Value=$license_model"
"Type=TERM_MATCH,Field=deploymentOption,Value=$DEPLOYMENT_OPTION"
"Type=TERM_MATCH,Field=termType,Value=OnDemand")
    local ondemand_json=$(aws pricing get-products --service-code AmazonRDS --filters
"${ondemand_filters[@]}" --region us-east-1 2>/dev/null)
    local on_demand_price=$(echo "$ondemand_json" | jq -r '.PriceList | .[]? | fromjson |
.terms.OnDemand | .. | .pricePerUnit?.USD? | select(. != null)' | head -n 1)

    # If no On-Demand price, no need to look for RIs.
    if [[ -z "$on_demand_price" ]]; then echo "NoPrice"; return; fi

    # 2. Get All Reserved Instance Prices at once
    local ri_filters=("Type=TERM_MATCH,Field=regionCode,Value=$REGION"
"Type=TERM_MATCH,Field=instanceType,Value=$instance_type"
```

```
"Type=TERM_MATCH,Field=databaseEngine,Value=$db_engine"
"Type=TERM_MATCH,Field=licenseModel,Value=$license_model"
"Type=TERM_MATCH,Field=deploymentOption,Value=$DEPLOYMENT_OPTION"
"Type=TERM_MATCH,Field=termType,Value=Reserved")
    local ri_json=$(aws pricing get-products --service-code AmazonRDS --filters
"${ri_filters[@]}" --region us-east-1 2>/dev/null)

    # 3. Parse all data and return as a single line
    echo -n "$on_demand_price"
    local purchase_options=("No Upfront" "Partial Upfront" "All Upfront")
    local term_lengths=("1yr" "3yr")

    for term in "${term_lengths[@]}"; do
        for po in "${purchase_options[@]}"; do
            local offer_term=$(echo "$ri_json" | jq -c '.PriceList | .[]? | fromjson | .terms.Reserved |
.[] | select(.termAttributes.LeaseContractLength == """$term""" and
.termAttributes.PurchaseOption == """$po""")')
            if [[ -z "$offer_term" ]]; then
                echo -n " 0.0 0.0"
            else
                local hourly_price=$(echo "$offer_term" | jq -r '.priceDimensions | .[] | select(.unit ==
"Hrs") | .pricePerUnit.USD // "0.0"')
                local upfront_fee=$(echo "$offer_term" | jq -r '.priceDimensions | .[] | select(.unit ==
"Quantity") | .pricePerUnit.USD // "0.0"')
                echo -n " ${hourly_price:-0.0} ${upfront_fee:-0.0}"
            fi
        done
    done
    echo "" # Final newline
}

# --- Main Script ---
if ! command -v aws &> /dev/null || ! command -v jq &> /dev/null || ! command -v bc &>
/dev/null; then
    echo "Error: Missing dependencies." >&2; exit 1
fi

# --- Output Logic ---
if [[ "$CSV_MODE" == true ]]; then
    # CSV Header
    echo "Instance Type,DB Engine,License Model,Term,Purchase Option,Annualized
Cost,Discount vs On-Demand"
else
    # Pretty Table Header
    echo " Dependencies found. Fetching prices for Region: $REGION, Deployment:
$DEPLOYMENT_OPTION..."
    echo ""
    printf "%-18s | %-15s | %-20s | %-22s | %-22s | %-22s\n" "Instance Type" "DB Engine"
"On-Demand (Annual)" "1-Yr RI No Upfront" "1-Yr RI Partial Upfront" "1-Yr RI All Upfront"
    printf "%-18s | %-15s | %-20s | %-22s | %-22s | %-22s\n" "" "" "(Annualized Cost)" "3-Yr RI
```

```
No Upfront" "3-Yr RI Partial Upfront" "3-Yr RI All Upfront"
    printf '%s\n'
'------------------+----------------+--------------------+----------------------+----------------------+---------
---------------'
fi

for instance in "${INSTANCE_TYPES[@]}"; do
    declare -A engine_license_map
    engine_license_map=(["MySQL"]="No License required" ["PostgreSQL"]="No License
required" ["MariaDB"]="No License required")
    for engine in "${!engine_license_map[@]}"; do
        license=${engine_license_map[$engine]}

        # Read all price details at once
        read on_demand_hr \
        ri_1yr_no_hr ri_1yr_no_up \
        ri_1yr_part_hr ri_1yr_part_up \
        ri_1yr_all_hr ri_1yr_all_up \
        ri_3yr_no_hr ri_3yr_no_up \
        ri_3yr_part_hr ri_3yr_part_up \
        ri_3yr_all_hr ri_3yr_all_up \
        <<< $(get_all_prices "$instance" "$engine" "$license")

        if [[ "$on_demand_hr" == "NoPrice" ]]; then continue; fi

        on_demand_annual=$(bc <<<"scale=2; $on_demand_hr * $HOURS_IN_YEAR / 1")

        # --- Process and Store Results ---
        # Associative array to hold all results
        declare -A results
        results["On-Demand_Cost"]=$on_demand_annual

        # Helper function for calculation to avoid repetition
        calculate_annual_cost() {
            local term_years=$1
            local hourly=$2
            local upfront=$3
            local total_term_cost=$(bc <<< "scale=2; ($hourly * $HOURS_IN_YEAR *
$term_years) + $upfront / 1")
            echo $(bc <<< "scale=2; $total_term_cost / $term_years")
        }

        results["1yr_No Upfront_Cost"]=$(calculate_annual_cost 1 $ri_1yr_no_hr $ri_1yr_no_up)
        results["1yr_Partial Upfront_Cost"]=$(calculate_annual_cost 1 $ri_1yr_part_hr
$ri_1yr_part_up)
        results["1yr_All Upfront_Cost"]=$(calculate_annual_cost 1 $ri_1yr_all_hr $ri_1yr_all_up)
        results["3yr_No Upfront_Cost"]=$(calculate_annual_cost 3 $ri_3yr_no_hr $ri_3yr_no_up)
        results["3yr_Partial Upfront_Cost"]=$(calculate_annual_cost 3 $ri_3yr_part_hr
$ri_3yr_part_up)
        results["3yr_All Upfront_Cost"]=$(calculate_annual_cost 3 $ri_3yr_all_hr $ri_3yr_all_up)
```

```
      # --- Print based on selected mode ---
      if [[ "$CSV_MODE" == true ]]; then
         echo "$instance,$engine,$license,On-Demand,,${results['On-Demand_Cost']},"
         purchase_options=("No Upfront" "Partial Upfront" "All Upfront")
         term_lengths=("1yr" "3yr")
         for term in "${term_lengths[@]}"; do
            for po in "${purchase_options[@]}"; do
               cost=${results["${term}_${po}_Cost"]}
               if [[ $(bc <<< "$cost > 0.0") -eq 1 ]]; then
                  discount=$(bc <<< "scale=4; ($on_demand_annual - $cost) /
$on_demand_annual")
                  printf "%s,%s,%s,%s,%s,%.2f,%.2f%%\n" "$instance" "$engine" "$license"
"$term" "$po" "$cost" "$(bc <<< "$discount * 100")"
               fi
            done
         done
      else
         declare -A display_strings
         purchase_options=("No Upfront" "Partial Upfront" "All Upfront")
         term_lengths=("1yr" "3yr")
         for term in "${term_lengths[@]}"; do
            for po in "${purchase_options[@]}"; do
               cost=${results["${term}_${po}_Cost"]}
               if [[ $(bc <<< "$cost > 0.0") -eq 1 ]]; then
                  discount=$(bc -l <<< "scale=2; 100 * ($on_demand_annual - $cost) /
$on_demand_annual")
                  display_strings["${term}_${po}"]=$(printf "\$%.2f (%2.0f%%)" "$cost"
"$discount")
               else
                  display_strings["${term}_${po}"]="Not Available"
               fi
            done
         done
         printf "%-18s | %-15s | %-20s | %-22s | %-22s | %-22s\n" "$instance" "$engine"
"\$${on_demand_annual}" "${display_strings['1yr_No Upfront']}" "${display_strings['1yr_Partial
Upfront']}" "${display_strings['1yr_All Upfront']}"
         printf "%-18s | %-15s | %-20s | %-22s | %-22s | %-22s\n" "" "" ""
"${display_strings['3yr_No Upfront']}" "${display_strings['3yr_Partial Upfront']}"
"${display_strings['3yr_All Upfront']}"
         printf '%s\n'
'------------------+-----------------+----------------------+------------------------+------------------------+---------
---------------'
      fi
   done
done
```

To run in the cloud shell environment we do the same thing as before.
Additionally for a csv file, we do:

```
./rds.sh --csv db.m5.xlarge db.r5.large > rds_costs.csv
```